Azure Red Hat OpenShift documentation

Azure Red Hat OpenShift provides single-tenant, high-availability Kubernetes clusters on Azure, supported by Red Hat and Microsoft.

About Azure Red Hat OpenShift

OVERVIEW

What is Azure Red Hat OpenShift?

Get started

QUICKSTART

Create an Azure Red Hat OpenShift cluster

Connect to an Azure Red Hat OpenShift cluster

Delete an Azure Red Hat OpenShift cluster

Set up

HOW-TO GUIDE

Create a private Azure Red Hat OpenShift cluster

Deploy a Red Hat OpenShift cluster with an ARM template or Bicip file

Configure DNS Forwarding for an Azure Red Hat OpenShift cluster

Configure built-in container registry for Azure Red Hat OpenShift

Deploy an application from source

Deploy an application using OpenShift Serverless

Deploy an Open Liberty/WebSphere Liberty application on an Azure Red Hat OpenShift cluster

Reference

REFERENCE

Azure CLI OpenShift 4

REST API

Red Hat OpenShift docs

Azure Red Hat OpenShift

Article • 04/17/2025

The Microsoft *Azure Red Hat OpenShift* service enables you to deploy fully managed OpenShift clusters. Azure Red Hat OpenShift extends Kubernetes. Running containers in production with Kubernetes requires other tools and resources. This often includes the need to coordinate image registries, storage management, networking solutions, and logging and monitoring tools that must be versioned and tested together. Building container-based applications requires even more integration work with middleware, frameworks, databases, and CI/CD tools. Azure Red Hat OpenShift combines all those items into a single platform, bringing ease of operations to IT teams while giving application teams what they need to execute.

Azure Red Hat OpenShift is jointly engineered, operated, and supported by Red Hat and Microsoft to provide an integrated support experience. There are no virtual machines to operate, and no patching is required. Control plane, infrastructure, and application nodes are patched, updated, and monitored on your behalf by Red Hat and Microsoft. Your Azure Red Hat OpenShift clusters are deployed into your Azure subscription and are included on your Azure bill.

You can choose your own registry, networking, storage, and CI/CD solutions, or use the built-in solutions for automated source code management, container and application builds, deployments, scaling, health management, and more. Azure Red Hat OpenShift provides an integrated sign-on experience through Microsoft Entra ID.

Access, security, and monitoring

For improved security and management, Azure Red Hat OpenShift lets you integrate with Microsoft Entra ID and use Kubernetes role-based access control (Kubernetes RBAC). You can also monitor the health of your cluster and resources.

Cluster and node

Azure Red Hat OpenShift nodes run on Azure virtual machines. You can connect storage to nodes and pods and upgrade cluster components.

Service Level Agreement

Azure Red Hat OpenShift offers a Service Level Agreement to guarantee that the service is available 99.95% of the time. For more information about the SLA, see Azure Red Hat OpenShift SLA.

Next steps

Learn the prerequisites for Azure Red Hat OpenShift:

Create an OpenShift Cluster

What's new with Azure Red Hat OpenShift?

Summarize this article for me

Microsoft Azure Red Hat OpenShift receives improvements on an ongoing basis. To stay up to date with the most recent developments, this article provides you with information about the latest releases.

February 2026 - Updates

- We're pleased to announce that support for managed identity enabled clusters is now GA on Microsoft Azure Red Hat OpenShift. Managed identities allow usage of short-term, limited privilege credentials. For more information, see Understand managed identities in Azure Red Hat OpenShift. If you created managed identity clusters during the preview period, those clusters will now also be considered GA.
- Azure Red Hat OpenShift now supports updates to y-stream versions in the `fast` channel.
- Azure Red Hat OpenShift is released in Mexico Central, New Zealand North and Malaysia West.

December 2025 - Version 4.19

OpenShift version 4.19 is now available as an Azure Red Hat OpenShift install option. For more information, see OpenShift Container Platform 4.19 Documentation.

In addition to making version 4.19 available as an installable version, this release also makes the following features generally available:

- Support for additional Microsoft Azure instance types, and enabling ARO cluster deployments on the Dxv6 machine series.
- Support changing the MTU for the cluster network. This includes pod-to-pod, pod-to-service, and node-to-node communication over the OVN overlay network.

November 2025 - Updates

- OpenShift Virtualization is now Generally Available on Azure Red Hat OpenShift.
- Confidential containers is now Generally Available on Azure Red Hat OpenShift.

November 2025 - Version 4.18

OpenShift version 4.18 is now available as an Azure Red Hat OpenShift install option. For more information, see OpenShift Container Platform 4.18 Documentation.

In addition to making version 4.18 available as an installable version, this release also makes the following features generally available:

- Installing an ARO cluster with virtual network encryption is now supported from version 4.18+. You are required to use Azure virtual machines that have the `premiumIO` parameter set to `true`. See Microsoft's documentation about Creating a virtual network with encryption and Requirements and Limitations for more information.
- Using User-defined Network Segmentation (UDN) is supported from version 4.18+.

October 2025 - Updates

- All even minor versions of Azure Red Hat OpenShift from version 4.16+ will now support Extended Update Support Add-On - Term 1 (EUS).
- Setting persistence for Prometheus data is enabled which allows you to protect your metrics and alerting data from data loss, by storing them in a persistent volume (PV).

August 2025 - Updates

- Azure Red Hat OpenShift is released in UAE Central.
- Azure Red Hat OpenShift is live in US Gov Texas and the Azure Resource Manager APIs and CLI are available.

June 2025 - Version 4.17

- OpenShift version 4.17 is now available as an Azure Red Hat OpenShift install option. For more information, see OpenShift Container Platform 4.17 Documentation.
- Azure Red Hat OpenShift is live in US Gov Arizona and the Azure Resource Manager APIs and CLI are available.

May 2025 - Updates

We're pleased to announce that OpenShift Virtualization is now in public preview for Azure Red Hat OpenShift. For more information about the release, see Red Hat OpenShift Virtualization on Azure Red Hat OpenShift in Public Preview. For more information about how to install, see OpenShift Virtualization for Azure Red Hat OpenShift (preview).

April 2025 - Updates

We're pleased to announce that support for managed identity enabled clusters is now in public preview on Microsoft Azure Red Hat OpenShift. Managed identities allow usage of short-term, limited privilege credentials. For more information, see Understand managed identities in Azure Red Hat OpenShift.

March 2025 - Version 4.16

We're pleased to announce the availability of OpenShift 4.16 install on Azure Red Hat OpenShift. This release enables OpenShift Container Platform 4.16 as an installable version for new clusters creation. You can check the end of support date on the support lifecycle page for previous versions.

In addition to making version 4.16 available as an installable version, this release also makes the following features generally available:

- Support for configuring a cluster wide proxy
- Guidance for OpenShift SDN to OVN-Kubernetes migration

January 2025 - Updates

- Public preview of Confidential Containers support on Azure Red Hat OpenShift released at Microsoft Ignite 2024. Confidential Containers provide a secure enclave within the host system, isolating applications and their data from potential threats.
- Azure Red Hat OpenShift released in Spain Central region.
- New article released on purchasing Reserved Instances for Azure Red Hat OpenShift.
- SRE maintenance operations that previously required reboots of the nodes were improved. The reboots won't be initiated by SREs and will be paused until the next cluster upgrade operation.

September 2024 - Version 4.15

We're pleased to announce the launch of OpenShift 4.15 for Azure Red Hat OpenShift. This release enables OpenShift Container Platform 4.15 as an installable version. You can check the end of support date on the support lifecycle page for previous versions.

In addition to making version 4.15 available as an installable version, this release also makes the following features generally available:

- CLI for multiple public IP addresses for larger clusters up to 250 nodes

August 2024 - Updates

- You can now create up to 20 IP addresses per Azure Red Hat OpenShift cluster load balancer. This feature was previously in preview but is now generally available. See Configure multiple IP addresses per cluster load balancer for details. Azure Red Hat OpenShift 4.x has a 250 pod-per-node limit and a 250 compute node limit. For instructions on adding large clusters, see Deploy a large Azure Red Hat OpenShift cluster.
- There's a change in the order of actions performed by Site Reliability Engineers of Azure RedHat OpenShift. To maintain the health of a cluster, a timely action is necessary if control plane resources are over-utilized. Now the control plane is resized proactively to maintain cluster health. After the resize of the control plane, a notification is sent out to you with the details of the changes made to the control plane. Make sure you have the quota available in your subscription for Site Reliability Engineers to perform the cluster resize action.

May 2024 - Version 4.14

We're pleased to announce the launch of OpenShift 4.14 for Azure Red Hat OpenShift. This release enables OpenShift Container Platform 4.14. You can check the end of support date on the support lifecycle page for previous versions.

In addition to making version 4.14 available, this release also makes the following features generally available:

- Egress IP (v4.12.45+, 4.13.21+)
- Bring your own Network Security Group (NSG).
- Support for Azure Resource Health alerts
- Support in Azure Terraform provider

December 2023 - Version 4.13

We're pleased to announce the launch of OpenShift 4.13 for Azure Red Hat OpenShift. This release enables OpenShift Container Platform 4.13. Version 4.11 will be outside of support

after February 10, 2024. Existing clusters version 4.11 and below should be upgraded before then.

September 2023 - Updates

To create a private cluster without a public IP address, you can now add the parameter `--outbound-type UserDefinedRouting` to the `aro create` command. See Create a private cluster without a public IP address for details.

A cluster that is deployed with this feature and is running version 4.11 or higher can be scaled to 120 nodes and 30,000 pods.

August 2023 - Version 4.12

We're pleased to announce the launch of OpenShift 4.12 for Azure Red Hat OpenShift. This release enables OpenShift Container Platform 4.12 .

June 2023 - Updates

- Removed dependencies on service endpoints
  - The addition of the egress lockdown feature provided access to key Azure resources through the Private Link service, thus removing the need to access ACR and storage accounts through a service endpoint and using a private endpoint instead. With this release, dependencies on service endpoints were removed, and new clusters won't create service endpoints on VNets.

February 2023 - Version 4.11

We're pleased to announce the launch of OpenShift 4.11 for Azure Red Hat OpenShift. This release introduces the following features:

- Ability to deploy OpenShift 4.11
- Multi-version support:
  - Enables customers to select specific Y and Z version of the release. For more information about versions, see Red Hat OpenShift versions.
  - Customers can still deploy 4.10 clusters if that version is specified. For more information, see Select a different version.
- OVN as the CNI for clusters 4.11 and above
- Accelerated networking VMs
- UltraSSD support

- Gen2 VM support

Last updated on 02/25/2026

Azure Red Hat OpenShift service definition

The following sections provide service definitions to help you manage your Azure Red Hat OpenShift account.

Billing

Azure Red Hat OpenShift clusters are deployed into a customer's Azure subscription. A customer pays Azure directly for costs incurred by an Azure Red Hat OpenShift cluster.

Azure Red Hat OpenShift nodes run on Azure Virtual Machines. They're billed according to Azure Linux virtual machine pricing. Compute, networking, and storage resources consumed by an Azure Red Hat OpenShift cluster are billed according to usage.

In addition to the compute and infrastructure costs, application nodes have another cost for the Azure Red Hat OpenShift license component. This cost is based on the number of application nodes and the instance type.

All standard Azure purchasing options, including reservations and Azure prepayment apply. Standard Azure purchasing options can be used for Azure Red Hat OpenShift. Also, standard Azure purchasing options can be used for virtual machines, networking, and storage resources consumed by the Azure Red Hat OpenShift cluster.

For more information about pricing, see Azure Red Hat OpenShift pricing.

Cluster self-service

Customers can create and delete their clusters using the Azure command-line utility (CLI). Azure Red Hat OpenShift clusters deploy with a kubeadmin user whose credentials are available from the Azure CLI after a cluster is successfully deployed.

You can perform all other Azure Red Hat OpenShift cluster actions, such as scaling nodes, by interacting with the OpenShift API using tools such as the OpenShift web console or the OpenShift CLI (`oc`).

Azure resource architecture

An Azure Red Hat OpenShift deployment requires two resource groups within an Azure subscription. The first resource group is created by the customer and contains the virtual networking components for the cluster. Keeping the networking elements separate allows the

customer to configure Azure Red Hat OpenShift to meet requirements and to add any peering options.

The second resource group is created by the Azure Red Hat OpenShift resource provider. It contains Azure Red Hat OpenShift cluster components, including virtual machines, network security groups, and load balancers. Azure Red Hat OpenShift cluster components located within this resource group aren't modifiable by the customer. Cluster configuration must be performed via interactions with the OpenShift API using the OpenShift web console or OpenShift CLI or similar tools.

> ⓘ **Note**
>
> The service principal for the resource provider requires the Network Contributor role on the cluster's virtual network. This role is required for the resource provider to create resources such as the Private Link service and load balancers.

Red Hat operators

The recommendation is that a customer provides a Red Hat pull secret to the Azure Red Hat OpenShift cluster during cluster creation. The Red Hat pull secret enables your cluster to access Red Hat container registries, along with other content from the OpenShift Operator Hub.

Azure Red Hat OpenShift clusters can still serve applications without providing the Red Hat pull secret, but without the pull secret, they're unable to install operators from the Operator Hub.

The Red Hat pull secret can also be provided to the cluster post deployment.

Compute

Azure Red Hat OpenShift clusters are provisioned with three or more worker nodes.

- In regions consisting of multiple availability zones, a worker node machine set is created in each zone. Also, a worker node is provisioned from each machine set.
- When an Azure region doesn't support availability zones, the Azure Red Hat OpenShift cluster provisions the worker nodes from a single machine set. Customers have the ability to increase the node count and the permission in each region.

Azure Red Hat OpenShift clusters are provisioned with three control plane nodes. These nodes are responsible for etcd key-value store and API-related workloads. The control plane node can't be used for customer workloads. Control plane node deployment follows the same rules as worker nodes.

- In regions that consist of multiple availability zones, a control plane node machine set is created in each zone. A control plane node is provisioned from each machine set.
- Where an Azure region doesn't support availability zones, the Azure Red Hat OpenShift cluster provisions the control plane nodes from a single machine set.

Azure compute types

For a list of supported control plane and worker node types and sizes, see Supported virtual machine sizes.

Azure regions

For regions supported by Azure Red Hat OpenShift, see Products available by region.

From the Azure CLI, view a list of available regions by running the following command:

Azure CLI

```
az provider show -n Microsoft.RedHatOpenShift --query "resourceTypes[?resourceType
== 'OpenShiftClusters']".locations -o yaml
```

Once deployed, an Azure Red Hat OpenShift cluster can't be moved to a different region. Similarly, you can't transfer Azure Red Hat OpenShift clusters between subscriptions.

Service level agreement

For SLA details, see SLA for Azure Red Hat OpenShift.

Support

Support requests for Azure Red Hat OpenShift can be submitted by;

- Requesting support in the Azure portal
- Requesting support via Red Hat Customer Portal

Requests are triaged and addressed by Microsoft and Red Hat support engineers. Azure Red Hat OpenShift includes Red Hat Premium Support. Support can be accessed via the Microsoft Azure portal.

To open support tickets directly with Red Hat, your cluster needs to have a pull secret. You can add it during cluster creation, or add it or update it on an existing cluster.

Logging

The following sections provide information about Azure Red Hat OpenShift security.

Cluster operations and audit logging

Azure Red Hat OpenShift deploys with services for maintaining the health and performance of the cluster and its components. These services include cluster operations and audit logs. Cluster operations and audit logs are forwarded automatically to an Azure aggregation system for support and troubleshooting. This data is only accessible to authorized support staff via approved mechanisms.

Customer cluster administrators can deploy an optional logging stack to aggregate all logs from their Azure Red Hat OpenShift cluster. For example, node system audit logs and infrastructure logs can be aggregated. However, these logs consume another cluster resources.

Application logging

OpenShift Logging is an optional, deployable component for Azure Red Hat OpenShift that provides a centralized, cluster-level logging solution based on the open-source Loki and Vector projects. Its primary purpose is to collect, aggregate, store, forward, and visualize all logs generated across your OpenShift cluster. Logs can be stored in Loki or forwarded to other services, such as Azure Monitor Logs.

For more information, see the Red Hat OpenShift Logging documentation.

Monitoring

The following section provides information about Azure Red Hat OpenShift monitoring.

Cluster metrics

Azure Red Hat OpenShift deploys with services for maintaining the health and performance of the cluster and its components. These services include the streaming of important metrics to an Azure aggregation system for support and troubleshooting purposes. This data is only accessible to authorized support staff via approved mechanisms.

Azure Red Hat OpenShift clusters come with an integrated Prometheus/Grafana stack to enable customers to view cluster monitoring. The stack includes CPU, memory, and network-based metrics.

These metrics, which are accessible via the web console, can also be used to view cluster-level status and capacity/usage through a Grafana dashboard. These metrics also allow for horizontal pod autoscaling that is based on CPU or memory metrics provided by an Azure Red Hat OpenShift customer.

Network

The following sections provide information about the Azure Red Hat OpenShift network.

Domain-validated certificates

By default, Azure Red Hat OpenShift includes TLS security certificates needed for both internal and external services on the cluster. For external routes, a Transport Layer Security (TLS) wildcard certificate is provided and installed in the cluster. A TLS certificate is also used for the OpenShift API endpoint. DigiCert is the certificate authority (CA) used for these certificates.

Custom domains

During deployment, Azure Red Hat OpenShift allows you to specify a custom domain for your cluster. The custom domain is used for both cluster services and for applications. You must create two DNS A records in your DNS server for the specified domain:

- `api`, which points to the API server IP address
- `*.apps`, which points to the ingress IP address

By default, Azure Red Hat OpenShift uses self-signed certificates for all of the routes created on custom domains. If you choose to use custom domains, connect to the cluster. Next, follow the OpenShift documentation to configure a custom certificate authority CA for your ingress controller and a custom CA for your API server.

Custom CAs for builds

Azure Red Hat OpenShift supports the use of CAs to be trusted by builds when pulling images from an image registry.

Load balancers

Azure Red Hat OpenShift deploys with two Azure load balancers. The first is used for ingress traffic to applications and for the OpenShift and Kubernetes APIs. The second is used for internal communications between cluster components.

Cluster ingress

Project administrators can add route annotations for many different purposes, including ingress control via an IP allow list.

Ingress policies can be changed by using NetworkPolicy objects, which use the ovs-networkpolicy plugin. Using NetworkPolicy objects allows for full control over ingress network policy down to the pod level, including between pods on the same cluster and even in the same namespace.

All cluster ingress traffic traverses the defined load balancer.

Cluster egress

Pod egress traffic control via EgressNetworkPolicy objects can be used to prevent or limit outbound traffic in Azure Red Hat OpenShift. Currently all virtual machines must have outbound internet access.

Cloud network configuration

Azure Red Hat OpenShift enables configuration of private network connections through several cloud provider-managed technologies:

- Virtual network connections
- Azure virtual network peering
- Azure virtual network gateway
- Azure Express route

No monitoring of these private network connections is provided by Red Hat SRE. Monitoring these connections is the responsibility of the customer.

Customer-specified DNS

Azure Red Hat OpenShift customers can specify their own DNS servers. For more information, see Configure custom DNS for your Azure Red Hat OpenShift cluster.

Container Network Interface

Azure Red Hat OpenShift comes with OVN (Open Virtual Network) as the Container Network Interface (CNI). Replacing the CNI isn't a supported operation. For more information, see OVN-Kubernetes network provider for Azure Red Hat OpenShift clusters.

Storage

The following sections provide information about Azure Red Hat OpenShift storage.

Encryption-at-rest

Azure Storage uses server-side encryption (SSE) to automatically encrypt your data when it's persisted to the cloud. By default, data is encrypted with Microsoft platform-managed keys.

Block storage (RWO)

Persistent volumes are backed by Azure-Disk block storage, which is Read-Write-Once (RWO). 1024-GiB disks are dynamically created and attached to each Azure Red Hat OpenShift controller plane node. These disks are Premium SSD LRS Azure-managed disks. Disk sizes for the default worker node machine sets can be configured during cluster creation.

Customers have permissions for creating more machine sets to better suit their requirements.

Persistent volumes (PVs), which can only be attached to a single node at a time, are specific to the availability zone in which they were provisioned. They can be attached to any node in the availability zone.

Azure limits how many PVs of type block store can be attached to a single node. Azure limits depend on the type and size of the virtual machine the customer selects for worker nodes. For example, to see the max data disks for the Dasv4-series, see Dasv4.

Shared storage (RWX)

Shared storage for Azure Red Hat OpenShift clusters must be configured by the customer. For an example of how to configure a storage class for Azure files, see Create an Azure Files StorageClass on Azure Red Hat OpenShift 4

Platform

The following sections provide information about the Azure Red Hat OpenShift platform.

Cluster backup policy

> ⓘ **Important**

It's **critical** that you have a backup plan for your applications and application data.

Application and application data backups aren't an automated part of the Azure Red Hat OpenShift service. For a tutorial on how to perform manual application backup, see Create an Azure Red Hat OpenShift 4 cluster Application Backup.

DaemonSets

Customers can create and run DaemonSets on Azure Red Hat OpenShift. To restrict DaemonSets to only running on worker nodes, use the following nodeSelector:

```
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
```

Azure Red Hat OpenShift version

Azure Red Hat OpenShift is run as a service. It permits customers to keep up to date with the latest stable OpenShift Container Platform version. For the support and upgrade policy, see Support lifecycle for Azure Red Hat OpenShift 4.

Support lifecycle

For information about the Azure Red Hat OpenShift support lifecycle, see Support lifecycle for Azure Red Hat OpenShift 4.

Container engine

Azure Red Hat OpenShift runs on OpenShift 4 and uses the CRI-O implementation of the Kubernetes container runtime interface as the only available container engine.

Operating system

Azure Red Hat OpenShift runs on OpenShift 4 using Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system for all control plane and worker nodes. Windows workloads aren't supported on Azure OpenShift as the platform doesn't currently support Windows worker nodes.

Kubernetes operator support

Azure Red Hat OpenShift supports operators created by Red Hat and certified independent software vendors (ISVs). Operators provided by Red Hat are supported by Red Hat. ISV operators are supported by the ISV.

To use OperatorHub, your cluster must be configured with a Red Hat pull secret. For more information about using OperatorHub, see Understanding OperatorHub

Security

The following sections provide information about Azure OpenShift security.

Authentication provider

Azure Red Hat OpenShift clusters aren't configured with any authentication providers.

Customers need to configure their own providers, such as Microsoft Entra ID. For information about configuring providers, see the following articles:

- Microsoft Entra authentication
- OpenShift identity providers

Regulatory compliance

For details about Azure Red Hat OpenShift's regulatory compliance certifications, see Microsoft Azure Compliance Offerings.

Next Steps

For more information, see the support policies documentation.

Last updated on 11/25/2025

Azure Red Hat OpenShift 4.0 support policy

Certain configurations for Microsoft Azure Red Hat OpenShift 4 clusters can affect your cluster's supportability. Azure Red Hat OpenShift 4 allows cluster administrators to make changes to internal cluster components, but not all changes are supported. The support policy below shares which modifications violate the policy and void support from Microsoft and Red Hat.

> ⓘ **Note**
>
> Features marked Technology Preview in OpenShift Container Platform aren't supported in Azure Red Hat OpenShift.

Cluster configuration requirements

Compute

- The cluster must have a minimum of three worker nodes and three master nodes.
- Don't scale the cluster workers to zero, or attempt a cluster shutdown. Deallocating or powering down any virtual machine in the cluster resource group isn't supported.
- Don't create more than 250 worker nodes on a cluster. 250 is the maximum number of nodes that can be created on a cluster. For more information, see Configure multiple IP addresses per Azure Red Hat OpenShift cluster load balancer.
- If you're making use of infrastructure nodes, don't run any undesignated workloads on them because it can affect the Service Level Agreement and cluster stability. Also, the recommendation is to have three infrastructure nodes; one in each availability zone. For more information, see Deploy infrastructure nodes in an Azure Red Hat OpenShift cluster.
- Non-RHCOS compute nodes aren't supported. For example, you can't use a Red Hat Enterprise Linux (RHEL) compute node.
- Don't attempt to remove, replace, add, or modify a master node. Those tasks are high risk operations that can cause issues with `etcd`, permanent network loss, and loss of access and manageability by a Site Reliability Engineer (SRE). If you feel that a master node should be replaced or removed, contact support before making any changes.
- Ensure ample virtual machine quota is available in case control plane nodes need to be scaled up by keeping at least double your current control plane `vCPU` count available.

Operators

- All OpenShift Cluster operators must remain in a managed state. The list of cluster operators can be returned by running `oc get clusteroperators`.

Workload management

- Don't add taints that would prevent any default OpenShift components from being scheduled.
- To avoid disruption that results from cluster maintenance, in-cluster workloads should be configured with high availability practices. These practices include, but aren't limited to, pod affinity and anti-affinity, pod disruption budgets, and adequate scaling.
- Don't run extra workloads on the control plane nodes. While they can be scheduled on the control plane nodes, it causes extra resource usage and stability issues that can affect the entire cluster.
- Running custom workloads (including operators installed from Operator Hub or other operators provided by Red Hat) in infrastructure nodes isn't supported.

Logging and monitoring

- Don't remove or modify the default cluster Prometheus service, except to modify scheduling of the default Prometheus instance or to set up persistence. If setting up persistence, don't allow the persistent volume to become full.
- Don't remove or modify the default cluster `Alertmanager` service, default receiver, or any default alerting rules, except to add other receivers that notify external systems.
- Don't remove or modify Azure Red Hat OpenShift service logging (mdsd pods).

Network and security

- Unless you're using your own Network Security Group through the bring your own Network Security Group feature, the Azure Red Hat OpenShift provided Network Security Group (NSG) can't be modified or replaced. Any attempt to modify or replace the NSG is reverted.
- All cluster virtual machines must have direct outbound internet access, at least to the Azure Resource Manager (ARM) and service logging (Geneva) endpoints. Proxying of HTTPS traffic required to run the Azure Red Hat OpenShift service isn't supported. See cluster-wide proxy instructions for proxy related configuration.
- The Azure Red Hat OpenShift service accesses your cluster via Private Link Service. Don't remove or modify service access.

Cluster management

- Don't remove or modify the `arosvc.azurecr.io` cluster pull secret.
- Don't create new `MachineConfig` objects or modify existing objects, unless explicitly supported in the Azure Red Hat OpenShift documentation.
- Don't create new `KubeletConfig` objects or modify existing objects, unless explicitly supported in the Azure Red Hat OpenShift documentation.
- Don't set any `unsupportedConfigOverrides` options. Setting these options prevents minor version upgrades.
- Don't place policies within your subscription or management group that prevent SREs from performing normal maintenance against the Azure Red Hat OpenShift cluster. For example, don't require tags on the Azure Red Hat OpenShift RP-managed cluster resource group.
- Don't circumvent the deny assignment that is configured as part of the service, or perform administrative tasks normally prohibited by the deny assignment.
- OpenShift relies on the ability to automatically tag Azure resources. If you configured a tagging policy, don't apply more than 10 user-defined tags to resources in the managed resource group.
- If cluster is configured to use managed identities:
  - Don't modify or remove any required role assignments post-install.
  - Don't use custom role definitions as a substitute for ARO's built-in role definitions.
  - Don't remove or modify any federated identity credentials that were configured on the platform workload identities at install time.
  - Don't attempt to re-use the same managed identity across multiple clusters or platform workloads.

Incident management

An incident is an event that results in a degradation or outage of Azure Red Hat OpenShift services. Incidents are created by a customer or Customer Experience and Engagement (CEE) member through a support case, directly by the centralized monitoring and alerting system, or directly by a member of the Site Reliability Engineer (SRE) team.

Depending on the effect on the service and customer, the incident is categorized in terms of severity.

The general workflow of how a new incident is managed is described as follows:

1. An SRE first responder is alerted to a new incident and begins an initial investigation.

2. After the initial investigation, the incident is assigned to an incident lead, who coordinates the recovery efforts.

3. The incident lead manages all communication and coordination around recovery, including any relevant notifications or support case updates.

4. When the incident is resolved, a brief summary of the incident and resolution is provided in the customer-initiated support ticket. This summary helps the customer understand the incident and its resolution in more detail.

If more information is required in addition to what is provided in the support ticket:

1. The customer must make a request for more information within five business days of the incident resolution.

2. Depending on the severity of the incident, a root cause summary, or a root cause analysis (RCA) might be provided to the customer in the support ticket. The additional information is provided within seven business days for root cause summary and 30 business days for root cause analysis from the incident resolution.

Supported virtual machine sizes

Azure Red Hat OpenShift 4 supports node instances on the following virtual machine sizes:

Control plane nodes

Expand table

Series	Size	vCPU	Memory: GiB
Dsv3	Standard_D8s_v3	8	32
Dsv3	Standard_D16s_v3	16	64
Dsv3	Standard_D32s_v3	32	128
Dsv4	Standard_D8s_v4	8	32
Dsv4	Standard_D16s_v4	16	64
Dsv4	Standard_D32s_v4	32	128
Dsv5	Standard_D8s_v5	8	32
Dsv5	Standard_D16s_v5	16	64
Dsv5	Standard_D32s_v5	32	128
Dsv6	Standard_D8s_v6	8	32

Series	Size	vCPU	Memory: GiB
Dsv6	Standard_D16s_v6	16	64
Dsv6	Standard_D32s_v6	32	128
Ddsv6	Standard_D8ds_v6	8	32
Ddsv6	Standard_D16ds_v6	16	64
Ddsv6	Standard_D32ds_v6	32	128
Dasv4	Standard_D8as_v4	8	32
Dasv4	Standard_D16as_v4	16	64
Dasv4	Standard_D32as_v4	32	128
Dasv5	Standard_D8as_v5	8	32
Dasv5	Standard_D16as_v5	16	64
Dasv5	Standard_D32as_v5	32	128
Ddsv5	Standard_D8ds_v5	8	32
Ddsv5	Standard_D16ds_v5	16	64
Ddsv5	Standard_D32ds_v5	32	128
Easv4	Standard_E8as_v4	8	64
Easv4	Standard_E16as_v4	16	128
Easv4	Standard_E20as_v4	20	160
Easv4	Standard_E32as_v4	32	256
Easv4	Standard_E48as_v4	48	384
Easv4	Standard_E64as_v4	64	512
Easv4	Standard_E96as_v4	96	672
Easv5	Standard_E8as_v5	8	64
Easv5	Standard_E16as_v5	16	128
Easv5	Standard_E20as_v5	20	160
Easv5	Standard_E32as_v5	32	256
Easv5	Standard_E48as_v5	48	384

Series	Size	vCPU	Memory: GiB
Easv5	Standard_E64as_v5	64	512
Easv5	Standard_E96as_v5	96	672
Eisv3	Standard_E64is_v3	64	432
Eis4	Standard_E80is_v4	80	504
Eids4	Standard_E80ids_v4	80	504
Eisv5	Standard_E104is_v5	104	672
Eidsv5	Standard_E104ids_v5	104	672
Esv4	Standard_E8s_v4	8	64
Esv4	Standard_E16s_v4	16	128
Esv4	Standard_E20s_v4	20	160
Esv4	Standard_E32s_v4	32	256
Esv4	Standard_E48s_v4	48	384
Esv4	Standard_E64s_v4	64	504
Esv5	Standard_E8s_v5	8	64
Esv5	Standard_E16s_v5	16	128
Esv5	Standard_E20s_v5	20	160
Esv5	Standard_E32s_v5	32	256
Esv5	Standard_E48s_v5	48	384
Esv5	Standard_E64s_v5	64	512
Esv5	Standard_E96s_v5	96	672
Fsv2	Standard_F72s_v2	72	144
Mms `*`	Standard_M128ms	128	3892

`*` Standard_M128ms doesn't support encryption at host.

Note that Dsv6 SKUs are supported on Azure Red Hat OpenShift version 4.19 or higher.

Worker nodes

General purpose

Expand table

Series	Size	vCPU	Memory: GiB
Dasv4	Standard_D4as_v4	4	16
Dasv4	Standard_D8as_v4	8	32
Dasv4	Standard_D16as_v4	16	64
Dasv4	Standard_D32as_v4	32	128
Dasv4	Standard_D64as_v4	64	256
Dasv4	Standard_D96as_v4	96	384
Dasv5	Standard_D4as_v5	4	16
Dasv5	Standard_D8as_v5	8	32
Dasv5	Standard_D16as_v5	16	64
Dasv5	Standard_D32as_v5	32	128
Dasv5	Standard_D64as_v5	64	256
Dasv5	Standard_D96as_v5	96	384
Ddsv5	Standard_D4ds_v5	4	16
Ddsv5	Standard_D8ds_v5	8	32
Ddsv5	Standard_D16ds_v5	16	64
Ddsv5	Standard_D32ds_v5	32	128
Ddsv5	Standard_D48ds_v5	48	192
Ddsv5	Standard_D64ds_v5	64	256
Ddsv5	Standard_D96ds_v5	96	384
Dsv3	Standard_D4s_v3	4	16
Dsv3	Standard_D8s_v3	8	32
Dsv3	Standard_D16s_v3	16	64
Dsv3	Standard_D32s_v3	32	128
Dsv4	Standard_D4s_v4	4	16

Series	Size	vCPU	Memory: GiB
Dsv4	Standard_D8s_v4	8	32
Dsv4	Standard_D16s_v4	16	64
Dsv4	Standard_D32s_v4	32	128
Dsv4	Standard_D64s_v4	64	256
Dsv5	Standard_D4s_v5	4	16
Dsv5	Standard_D8s_v5	8	32
Dsv5	Standard_D16s_v5	16	64
Dsv5	Standard_D32s_v5	32	128
Dsv5	Standard_D64s_v5	64	256
Dsv5	Standard_D96s_v5	96	384
Dsv6	Standard_D4s_v6	4	16
Dsv6	Standard_D8s_v6	8	32
Dsv6	Standard_D16s_v6	16	64
Dsv6	Standard_D32s_v6	32	128
Dsv6	Standard_D48s_v6	48	192
Dsv6	Standard_D64s_v6	64	256
Dsv6	Standard_D96s_v6	96	384
Dsv6	Standard_D128s_v6	128	512
Dsv6	Standard_D192s_v6	192	768
Ddsv6	Standard_D4ds_v6	4	16
Ddsv6	Standard_D8ds_v6	8	32
Ddsv6	Standard_D16ds_v6	16	64
Ddsv6	Standard_D32ds_v6	32	128
Ddsv6	Standard_D48ds_v6	48	192
Ddsv6	Standard_D64ds_v6	64	256
Ddsv6	Standard_D96ds_v6	96	384

Series	Size	vCPU	Memory: GiB
Ddsv6	Standard_D128ds_v6	128	512
Ddsv6	Standard_D192ds_v6	192	768
Dlsv6	Standard_D4ls_v6	4	8
Dlsv6	Standard_D8ls_v6	8	16
Dlsv6	Standard_D16ls_v6	16	32
Dlsv6	Standard_D32ls_v6	32	64
Dlsv6	Standard_D48ls_v6	48	96
Dlsv6	Standard_D64ls_v6	64	128
Dlsv6	Standard_D96ls_v6	96	192
Dlsv6	Standard_D128ls_v6	128	256
Dldsv6	Standard_D4lds_v6	4	8
Dldsv6	Standard_D8lds_v6	8	16
Dldsv6	Standard_D16lds_v6	16	32
Dldsv6	Standard_D32lds_v6	32	64
Dldsv6	Standard_D48lds_v6	48	96
Dldsv6	Standard_D64lds_v6	64	128
Dldsv6	Standard_D96lds_v6	96	192
Dldsv6	Standard_D128lds_v6	128	256

Note that D*sv6 SKUs are supported on Azure Red Hat OpenShift version 4.19 or higher.

Memory optimized

Expand table

Series	Size	vCPU	Memory: GiB
Easv4	Standard_E4as_v4	4	32
Easv4	Standard_E8as_v4	8	64
Easv4	Standard_E16as_v4	16	128

Series	Size	vCPU	Memory: GiB
Easv4	Standard_E20as_v4	20	160
Easv4	Standard_E32as_v4	32	256
Easv4	Standard_E48as_v4	48	384
Easv4	Standard_E64as_v4	64	512
Easv4	Standard_E96as_v4	96	672
Easv5	Standard_E8as_v5	8	64
Easv5	Standard_E16as_v5	16	128
Easv5	Standard_E20as_v5	20	160
Easv5	Standard_E32as_v5	32	256
Easv5	Standard_E48as_v5	48	384
Easv5	Standard_E64as_v5	64	512
Easv5	Standard_E96as_v5	96	672
Esv3	Standard_E4s_v3	4	32
Esv3	Standard_E8s_v3	8	64
Esv3	Standard_E16s_v3	16	128
Esv3	Standard_E32s_v3	32	256
Esv4	Standard_E4s_v4	4	32
Esv4	Standard_E8s_v4	8	64
Esv4	Standard_E16s_v4	16	128
Esv4	Standard_E20s_v4	20	160
Esv4	Standard_E32s_v4	32	256
Esv4	Standard_E48s_v4	48	384
Esv4	Standard_E64s_v4	64	504
Esv5	Standard_E4s_v5	4	32
Esv5	Standard_E8s_v5	8	64
Esv5	Standard_E16s_v5	16	128

Series	Size	vCPU	Memory: GiB
Esv5	Standard_E20s_v5	20	160
Esv5	Standard_E32s_v5	32	256
Esv5	Standard_E48s_v5	48	384
Esv5	Standard_E64s_v5	64	512
Esv5	Standard_E96s_v5	96	672
Edsv5	Standard_E96ds_v5	96	672
Eisv3	Standard_E64is_v3	64	432
Eis4	Standard_E80is_v4	80	504
Eids4	Standard_E80ids_v4	80	504
Eisv5	Standard_E104is_v5	104	672
Eidsv5	Standard_E104ids_v5	104	672

Compute optimized

Expand table

Series	Size	vCPU	Memory: GiB
Fsv2	Standard_F4s_v2	4	8
Fsv2	Standard_F8s_v2	8	16
Fsv2	Standard_F16s_v2	16	32
Fsv2	Standard_F32s_v2	32	64
Fsv2	Standard_F72s_v2	72	144
FX4mds	Standard_FX4mds	4	84
FX48mds	Standard_FX48mds	48	1008
FX4mds_v2	Standard_FX4mds_v2	4	84
FX8mds_v2	Standard_FX8mds_v2	8	164
FX16mds_v2	Standard_FX16mds_v2	16	336
FX32mds_v2	Standard_FX32mds_v2	32	672

Series	Size	vCPU	Memory: GiB
FX48mds_v2	Standard_FX48mds_v2	48	1008
FX64mds_v2	Standard_FX64mds_v2	64	1344

Memory and compute optimized

Expand table

Series	Size	vCPU	Memory: GiB
Mms *	Standard_M128ms	128	3892

* Standard_M128ms doesn't support encryption at host

Storage optimized

Expand table

Series	Size	vCPU	Memory: GiB
L4s	Standard_L4s	4	32
L8s	Standard_L8s	8	64
L16s	Standard_L16s	16	128
L32s	Standard_L32s	32	256
L8s_v2	Standard_L8s_v2	8	64
L16s_v2	Standard_L16s_v2	16	128
L32s_v2	Standard_L32s_v2	32	256
L48s_v2	Standard_L48s_v2	48	384
L64s_v2	Standard_L64s_v2	64	512
L8s_v3	Standard_L8s_v3	8	64
L16s_v3	Standard_L16s_v3	16	128
L32s_v3	Standard_L32s_v3	32	256
L48s_v3	Standard_L48s_v3	48	384

Series	Size	vCPU	Memory: GiB
L64s_v3	Standard_L64s_v3	64	512
Lsv4	Standard_L4s_v4	4	32
Lsv4	Standard_L8s_v4	8	64
Lsv4	Standard_L16s_v4	16	128
Lsv4	Standard_L32s_v4	32	256
Lsv4	Standard_L48s_v4	48	384
Lsv4	Standard_L64s_v4	64	512
Lsv4	Standard_L80s_v4	80	640
Lsv4	Standard_L96s_v4	96	768

Note that Lsv4 SKUs are supported on Azure Red Hat OpenShift version 4.19 or higher.

GPU workload

Expand table

Series	Size	vCPU	Memory: GiB
NC4asT4v3	Standard_NC4as_T4_v3	4	28
NC6sV3	Standard_NC6s_v3	6	112
NC8asT4v3	Standard_NC8as_T4_v3	8	56
NC12sV3	Standard_NC12s_v3	12	224
NC16asT4v3	Standard_NC16as_T4_v3	16	110
NC24sV3	Standard_NC24s_v3	24	448
NC24rsV3	Standard_NC24rs_v3	24	448
NC64asT4v3	Standard_NC64as_T4_v3	64	440
ND96asr_v4*	Standard_ND96asr_v4	96	900
ND96amsr_A100_v4 *	Standard_ND96amsr_A100_v4	96	1924
NC24ads_A100_v4 *	Standard_NC24ads_A100_v4	24	220
NC48ads_A100_v4 *	Standard_NC48ads_A100_v4	48	440

Series	Size	vCPU	Memory: GiB
NC96ads_A100_v4 *	Standard_NC96ads_A100_v4	96	880

* Day-2 only (not supported as an install-time option)

Related content

For more information, see Support lifecycle for Azure Red Hat OpenShift 4.

Last updated on 03/17/2026

Overview of responsibilities for Azure Red Hat OpenShift

Article • 02/25/2025

This document outlines the responsibilities of Microsoft, Red Hat, and customers for Azure Red Hat OpenShift clusters. For more information about Azure Red Hat OpenShift and its components, see the Azure Red Hat OpenShift Service Definition.

While Microsoft and Red Hat manage the Azure Red Hat OpenShift service, the customer shares responsibility for the functionality of their cluster. While Azure Red Hat OpenShift clusters are hosted on Azure resources in customer Azure subscriptions, they are accessed remotely. Underlying platform and data security is owned by Microsoft and Red Hat.

Overview

Expand table

Resource	Incident and Operations Management	Change Management	Identity and Access Management	Security and Regulation Compliance
Customer data	Customer	Customer	Customer	Customer
Customer applications	Customer	Customer	Customer	Customer
Developer services	Customer	Customer	Customer	Customer
Platform monitoring	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat
Logging	Microsoft and Red Hat	Shared	Shared	Shared
Application networking	Shared	Shared	Shared	Microsoft and Red Hat
Cluster networking	Microsoft and Red Hat	Shared	Shared	Microsoft and Red Hat
Virtual networking	Shared	Shared	Shared	Shared

Control plane nodes	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat
Worker nodes	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat
Cluster Version	Microsoft and Red Hat	Shared	Microsoft and Red Hat	Microsoft and Red Hat
Capacity Management	Microsoft and Red Hat	Shared	Microsoft and Red Hat	Microsoft and Red Hat
Virtual Storage	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat
Physical Infrastructure and Security	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat	Microsoft and Red Hat

Table 1. Responsibilities by resource

Tasks for shared responsibilities by area

Incident and operations management

The customer, Microsoft, and Red Hat share responsibility for the monitoring and maintenance of an Azure Red Hat OpenShift cluster. The customer is responsible for incident and operations management of customer application data and any custom networking the customer may have configured.

Expand table

Resource	Microsoft and Red Hat responsibilities	Customer responsibilities
Application networking	• Monitor cloud load balancer(s) and native OpenShift router service, and respond to alerts.	• Monitor health of service load balancer endpoints. • Monitor health of application routes, and the endpoints behind them. • Report outages to Microsoft and Red Hat.
Virtual networking	• Monitor cloud load balancers, subnets, and Azure cloud components necessary for	• Monitor network traffic that is optionally configured via VNet to VNet connection, VPN connection,

	default platform networking, and respond to alerts.	or Private Link connection for potential issues or security threats.

Table 2. Shared responsibilities for incident and operations management

Change management

Microsoft and Red Hat are responsible for enabling changes to the cluster infrastructure and services that the customer controls, as well as maintaining versions available for the master nodes, infrastructure services, and worker nodes. The customer is responsible for initiating infrastructure changes and installing and maintaining optional services and networking configurations on the cluster, as well as all changes to customer data and customer applications.

Expand table

Resource	Microsoft and Red Hat responsibilities	Customer responsibilities
Logging	• Centrally aggregate and monitor platform audit logs. • Provide documentation for the customer to enable application logging using Log Analytics through Azure Monitor for containers. • Provide audit logs upon customer request.	• Install the optional default application logging operator on the cluster. • Install, configure, and maintain any optional app logging solutions, such as logging sidecar containers or third-party logging applications. • Tune size and frequency of application logs being produced by customer applications if they are affecting the stability of the cluster. • Request platform audit logs through a support case for researching specific incidents.

Application networking	• Set up public cloud load balancers • Set up the OpenShift Ingress cluster operator and the default IngressController. Provide the ability to add additional customer-managed IngressControllers and set the default IngressController as private. • Install, configure, and maintain the OVN-Kubernetes network plugin and related components for default internal pod traffic.	• Configure non-default pod network permissions for project and pod networks, pod ingress, and pod egress using NetworkPolicy objects. • Request and configure any additional service load balancers for specific services.
Cluster networking	• Set up cluster management components, such as public or private service endpoints and necessary integration with virtual networking components. • Set up internal networking components required for internal cluster communication between worker and master nodes.	• Provide optional non-default IP address ranges for machine CIDR, service CIDR, and pod CIDR if needed through OpenShift Cluster Manager when the cluster is provisioned. • Request that the API service endpoint be made public or private on cluster creation or after cluster creation through Azure CLI.
Virtual networking	• Set up and configure virtual networking components required to provision the cluster, including virtual private cloud, subnets, load balancers, internet gateways, NAT gateways, etc. • Provide the ability for the customer to manage VPN connectivity with on-premises resources, VNet to VNet connectivity, and Private Link connectivity as required through OpenShift Cluster Manager. • Enable customers to create and deploy public cloud load balancers for use with service load balancers.	• Set up and maintain optional public cloud networking components, such as VNet to VNet connection, VPN connection, or Private Link connection. • Request and configure any additional service load balancers for specific services.

Cluster Version	• Communicate schedule and status of upgrades for minor and maintenance versions • Publish changelogs and release notes for minor and maintenance upgrades	• Initiate Upgrade of cluster • Test customer applications on minor and maintenance versions to ensure compatibility
Capacity Management	• Monitor utilization of control plane (master nodes) resources including Network, Storage and Compute capacity • Proactively scale and/or resize control plane nodes to maintain quality of service	• Add or remove additional worker nodes as required. • Respond to Microsoft and Red Hat notifications regarding cluster resource requirements. • Ensure ample quota is available for larger control plane VMs in case of scaling operation

Table 3. Shared responsibilities for change management

Identity and Access Management

Identity and Access management includes all responsibilities for ensuring that only proper individuals have access to cluster, application, and infrastructure resources. This includes tasks such as providing access control mechanisms, authentication, authorization, and managing access to resources.

Expand table

Resource	Microsoft and Red Hat responsibilities	Customer responsibilities
Logging	• Adhere to an industry standards-based tiered internal access process for platform audit logs. • Provide native OpenShift RBAC capabilities.	• Configure OpenShift RBAC to control access to projects and by extension a project's application logs. • For third-party or custom application logging solutions, the customer is responsible for access management.
Application networking	• Provide native OpenShift RBAC capabilities.	• Configure OpenShift RBAC to control access to route configuration as required.

Cluster networking	• Provide native OpenShift RBAC capabilities.	• Manage Red Hat organization membership of Red Hat accounts. • Manage Org Admins for Red Hat organization to grant access to OpenShift Cluster Manager. • Configure OpenShift RBAC to control access to route configuration as required.
Virtual networking	• Provide customer access controls through OpenShift Cluster Manager.	• Manage optional user access to public cloud components through OpenShift Cluster Manager.

Table 4. Shared responsibilities for identity and access management

Security and compliance

Security and compliance includes any responsibilities and controls that ensure compliance with relevant laws, policies, and regulations.

Expand table

Resource	Microsoft and Red Hat responsibilities	Customer responsibilities
Logging	• Send cluster audit logs to a Microsoft and Red Hat SIEM to analyze for security events. Retain audit logs for a defined period of time to support forensic analysis.	• Analyze application logs for security events. Send application logs to an external endpoint through logging sidecar containers or third-party logging applications if longer retention is required than is offered by the default logging stack.
Virtual networking	• Monitor virtual networking components for potential issues and security threats. • Use additional public Microsoft and Red Hat Azure tools for additional monitoring and protection.	• Monitor optionally configured virtual networking components for potential issues and security threats. • Configure any necessary firewall rules or data center protections as required.

Table 5. Shared responsibilities for security and regulation compliance

Customer responsibilities when using Azure Red Hat OpenShift

Customer data and applications

The customer is responsible for the applications, workloads, and data they deploy to Azure Red Hat OpenShift. However, Microsoft and Red Hat provide various tools to help the customer manage data and applications on the platform.

Expand table

Resource	How Microsoft and Red Hat helps	Customer responsibilities
Customer Data	• Maintain platform-level standards for data encryption as defined by industry security and compliance standards. • Provide OpenShift components to help manage application data, such as secrets. • Enable integration with third-party data services (such as Azure SQL) to store and manage data outside of the cluster and/or Microsoft and Red Hat Azure.	• Maintain responsibility for all customer data stored on the platform and how customer applications consume and expose this data. • Etcd encryption
Customer Applications	• Provision clusters with OpenShift components installed so that customers can access the OpenShift and Kubernetes APIs to deploy and manage containerized applications. • Provide access to OpenShift APIs that a customer can use to set up Operators to add community, third-party, Microsoft and Red Hat,	• Maintain responsibility for customer and third-party applications, data, and their complete lifecycle. • If a customer adds Red Hat, community, third party, their own, or other services to the cluster by using Operators or external images, the customer is responsible for these services and for working with the appropriate provider (including Red Hat) to troubleshoot any issues. • Use the provided tools and features to configure and deploy ; keep up-to-date ; set up resource requests and limits ; size the cluster to have enough resources to run apps ; set up

	and Red Hat services to the cluster. • Provide storage classes and plug-ins to support persistent volumes for use with customer applications.	permissions ; integrate with other services; manage any image streams or templates that the customer deploys ; externally serve ; save, back up, and restore data; and otherwise manage their highly available and resilient workloads. • Maintain responsibility for monitoring the applications run on Azure Red Hat OpenShift; including installing and operating software to gather metrics and create alerts.

Table 6. Customer responsibilities for customer data, customer applications, and services

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Quickstart: Create an Azure Red Hat OpenShift 4 cluster

07/15/2025

In this Quickstart, you learn how to create a Microsoft Azure Red Hat OpenShift cluster using Azure CLI or the Azure portal. Azure Red Hat OpenShift is a managed OpenShift service that lets you deploy and manage clusters.

When you create the cluster, a service principal is created or you can use an existing service principal. A cluster that uses a service principal can't be migrated to use a managed identity. You need to create a new cluster that uses a managed identity on an Azure Red Hat OpenShift cluster. For more information, see Create an Azure Red Hat OpenShift cluster with managed identities (preview).

Prerequisites

Ensure you're using Azure CLI version 2.67.0 or higher. Use `az --version` to find the installed version of Azure CLI. If you need to install or upgrade, see Install Azure CLI.

Azure Red Hat OpenShift requires a minimum of 44 cores to create an OpenShift cluster. The default Azure resource quota for a new Azure subscription doesn't meet this requirement. To request an increase in your resource limit, see Increase VM-family vCPU quotas.

The 44 cores are used as follows:

- Bootstrap machine: 8 cores
- Control plane (master machines): 24 cores
- Compute (worker machines): 12 cores

When the installation is complete, the bootstrap machine is removed and your cluster uses a total of 36 cores. For more information, see Installing on Azure.

Run the following command to check your Azure subscription's quota for the *Standard DSv5* virtual machine size that's the default virtual machine size for the `az aro create` command.

Azure CLI

```
LOCATION=eastus
az vm list-usage --location $LOCATION \
  --query "[?contains(name.value, 'standardDSv5Family')]" \
  --output table
```

Verify your permissions

In this article, you create a resource group that contains the virtual network for the cluster. You need Contributor and User Access Administrator permissions or Owner permissions, either directly on the virtual network or on the resource group or subscription containing it.

You also need sufficient Microsoft Entra permissions, either a member user of the tenant, or a guest assigned with role Application administrator, for the tooling to create an application and service principal on your behalf for the cluster. For more information, see Member and guests and Assign Microsoft Entra roles.

Register the resource providers

The following resource providers must be registered in your Azure subscription:

- `Microsoft.RedHatOpenShift`
- `Microsoft.Compute`
- `Microsoft.Storage`
- `Microsoft.Authorization`

If you have multiple Azure subscriptions, specify the relevant subscription ID:

Azure CLI

```
az account set --subscription <SUBSCRIPTION ID>
```

To verify if a resource provider is registered, use the following command with the resource provider name. The command checks the `Microsoft.RedHatOpenShift` resource provider and returns a value of `Registered` or `NotRegistered`.

Azure CLI

```
az provider list --query "[?
namespace=='Microsoft.RedHatOpenShift'].registrationState" \
  --output table
```

If you need to register resource providers, use the following commands:

- Register the `Microsoft.RedHatOpenShift` resource provider:

  Azure CLI

  ```
  az provider register --namespace Microsoft.RedHatOpenShift --wait
  ```

- Register the `Microsoft.Compute` resource provider:

  Azure CLI

  ```
  az provider register --namespace Microsoft.Compute --wait
  ```

- Register the `Microsoft.Storage` resource provider:

  Azure CLI

  ```
  az provider register --namespace Microsoft.Storage --wait
  ```

- Register the `Microsoft.Authorization` resource provider:

  Azure CLI

  ```
  az provider register --namespace Microsoft.Authorization --wait
  ```

Get a Red Hat pull secret (optional)

A Red Hat pull secret enables your cluster to access Red Hat container registries, along with other content like operators from OperatorHub. A pull secret doesn't change the cost of the Red Hat OpenShift license. This step is optional but recommended. For example, if you plan to use OpenShift Virtualization, include the pull secret in your cluster deployment so that operators like OpenShift Virtualization can be installed.

If you decide to add the pull secret later, see add or update your pull secret. The field `cloud.openshift.com` is removed from your secret even if your pull-secret contains that field. This field enables an extra monitoring feature, which sends data to RedHat and is thus disabled by default. To enable this feature, see Enabling remote health reporting.

1. Sign in to the Red Hat OpenShift cluster manager portal.

   You need to sign in to your Red Hat account or create a new Red Hat account with your business email and accept the terms and conditions.

2. Select **Download pull secret** and download a pull secret to use with your cluster.

   - Keep the saved `pull-secret.txt` file somewhere safe. The file is used in each cluster creation if you need to create a cluster that includes samples or operators for Red Hat or certified partners.
   - When running the `az aro create` command, you can reference your pull secret using the `--pull-secret @pull-secret.txt` parameter. Execute `az aro create` from

the directory where you stored your `pull-secret.txt` file. Otherwise, replace `@pull-secret.txt` with `@/path/to/my/pull-secret.txt`.

- For an Azure portal cluster deployment, copy the pull secret and on the **Authentication** page, paste the pull secret into the **Red Hat pull secret** text box.
- If you copy your pull secret or reference it in other scripts, your pull secret should be formatted as a valid JSON string.

Prepare a custom domain for your cluster (optional)

When running the `az aro create` command, you can specify a custom domain for your cluster by using the `--domain foo.example.com` parameter.

> **Note**
>
> Although adding a domain name is optional when you create a cluster through Azure CLI, a domain name, or a prefix used as part of the autogenerated DNS name for OpenShift console and API servers, is needed when adding a cluster through the portal. For more information, see this article's **Azure portal** tab.

If you provide a custom domain for your cluster, note the following points:

- After creating your cluster, you must create two DNS `A` records in your DNS server for the `--domain` specified:
  - **api** - pointing to the API server IP address
  - ***.apps** - pointing to the ingress IP address
  - Retrieve these values by executing the following command after cluster creation: `az aro show -n -g --query '{api:apiserverProfile.ip, ingress:ingressProfiles[0].ip}'`.
- The OpenShift console is available at a URL like `https://console-openshift-console.apps.example.com`, instead of the built-in domain `https://console-openshift-console.apps.<random>.<location>.aroapp.io`.
- By default, OpenShift uses self-signed certificates for all of the routes created on custom domains `*.apps.example.com`. If you choose to use custom DNS after connecting to the cluster, you need to follow the OpenShift documentation to configure a custom CA for your ingress controller and a custom CA for your API server.

Create a virtual network containing two empty subnets

Next, you create a virtual network containing two empty subnets. If you have existing virtual network that meets your needs, you can skip this step.

For information about networking and requirements, see Networking for Azure Red Hat OpenShift.

1. Set the following variables in the shell environment in which you execute the `az` commands.

   Console

   ```
   LOCATION=eastus                 # the location of your cluster
   RESOURCEGROUP=aro-rg            # the name of the resource group where you
   want to create your cluster
   CLUSTER=cluster                 # the name of your cluster
   VIRTUALNETWORK=aro-vnet         # the name of the virtual network
   ```

2. Create a resource group.

   An Azure resource group is a logical group in which Azure resources are deployed and managed. When you create a resource group, you're asked to specify a location. This location is where resource group metadata is stored, and it's also where your resources run in Azure if you don't specify another region during resource creation. Create a resource group using the az group create command.

   > **Note**
   >
   > Azure Red Hat OpenShift isn't available in all regions where an Azure resource group can be created. See Available regions for information on where Azure Red Hat OpenShift is supported.

   Azure CLI

   ```
   az group create \
     --name $RESOURCEGROUP \
     --location $LOCATION
   ```

   The following example output shows the resource group created successfully:

   JSON

   ```
   {
     "id": "/subscriptions/<guid>/resourceGroups/aro-rg",
     "location": "eastus",
     "name": "aro-rg",
     "properties": {
       "provisioningState": "Succeeded"
     },
   ```

```
  "type": "Microsoft.Resources/resourceGroups"
}
```

3. Create a virtual network.

   Azure Red Hat OpenShift clusters running OpenShift 4 require a virtual network with two empty subnets, for the master and worker nodes. You can either create a new virtual network for this cluster, or use an existing virtual network.

   Create a new virtual network in the same resource group you created earlier:

   Azure CLI

```
az network vnet create \
  --resource-group $RESOURCEGROUP \
  --name $VIRTUALNETWORK \
  --address-prefixes 10.0.0.0/22
```

   The following example output shows the virtual network created successfully:

   JSON

```
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "10.0.0.0/22"
      ]
    },
    "dhcpOptions": {
      "dnsServers": []
    },
    "id": "/subscriptions/<guid>/resourceGroups/aro-
rg/providers/Microsoft.Network/virtualNetworks/aro-vnet",
    "location": "eastus",
    "name": "aro-vnet",
    "provisioningState": "Succeeded",
    "resourceGroup": "aro-rg",
    "type": "Microsoft.Network/virtualNetworks"
  }
}
```

4. Add an empty subnet for the master nodes.

   Azure CLI

```
az network vnet subnet create \
  --resource-group $RESOURCEGROUP \
  --vnet-name $VIRTUALNETWORK \
```

```
  --name master-subnet \
  --address-prefixes 10.0.0.0/23
```

5. Add an empty subnet for the worker nodes.

Azure CLI

```
az network vnet subnet create \
  --resource-group $RESOURCEGROUP \
  --vnet-name $VIRTUALNETWORK \
  --name worker-subnet \
  --address-prefixes 10.0.2.0/23
```

Create the cluster

To create a cluster, run the following command. If you choose to use either of the following options, modify the command accordingly:

- Optionally, you can pass your Red Hat pull secret, which enables your cluster to access Red Hat container registries along with other content. Add the `--pull-secret @pull-secret.txt` argument to your command.
- Optionally, you can use a custom domain. Add the `--domain foo.example.com` argument to your command, replacing `foo.example.com` with your own custom domain.
- The default master virtual machine size is `Standard D8s_v5`. If you need a different virtual machine size, use the `--master-vm-size` parameter. For example, `--master-vm-size Standard_D8s_v3`.
- The default worker virtual machine size is `Standard D4s_v5`. If you need a different virtual machine size, use the `--worker-vm-size` parameter. For example, `--worker-vm-size Standard_D4s_v3`.
- The cluster is created with an OpenShift Container Platform version with install availability. To ensure your cluster is created with the latest available install version see select a different version. In the `--version` parameter, replace `<x.y.z>` with a specific version like `4.17.27`.
- For more information about the command to create the cluster, see az aro create.

> **Note**
>
> The maximum number of worker nodes definable at creation time is 50. You can scale out up to 250 nodes after the cluster is created.

Azure CLI

```
az aro create \
  --resource-group $RESOURCEGROUP \
  --name $CLUSTER \
  --vnet $VIRTUALNETWORK \
  --master-subnet master-subnet \
  --worker-subnet worker-subnet \
  --version <x.y.z>
```

After you run the `az aro create` command, it takes about 45 minutes to create the cluster.

Large scale clusters

If you need to deploy an Azure Red Hat OpenShift cluster with more than 100 worker nodes, see Deploy a large Azure Red Hat OpenShift cluster.

Select a different version

Use the following command to list the versions available for the location where you want to deploy a cluster.

Azure CLI

```
az aro get-versions --location $LOCATION --output table
```

Output

```
Version
---------
4.14.38
4.14.51
4.15.35
4.15.49
4.16.30
4.16.39
4.17.27
```

After you decide which version you want to install, specify it in the `az aro create` command's `--version` parameter.

Next steps

Quickstart: Connect to an Azure Red Hat OpenShift 4 cluster

Quickstart: Connect to an Azure Red Hat OpenShift 4 cluster

07/07/2025

In this Quickstart, you learn how to connect to a Microsoft Azure Red Hat OpenShift cluster running OpenShift 4 with the `kubeadmin` user through the OpenShift web console.

Prerequisites

This article requires Azure CLI version 2.6.0 or later. To find the version, run the `az --version` command. If you need to install or upgrade, see Install Azure CLI.

Connect to the cluster

You can log into the cluster using the `kubeadmin` user. Run the following command to get the `kubeadmin` user's password.

Create variables for your cluster name and resource group name. Replace `<resourceGroupName>` and `<clusterName>` with your cluster's values.

Azure CLI

```
RESOURCEGROUP=<resourceGroupName>
CLUSTER=<clusterName>
```

Azure CLI

```
az aro list-credentials \
  --name $CLUSTER \
  --resource-group $RESOURCEGROUP
```

The following example output shows the password in `kubeadminPassword`.

JSON

```
{
  "kubeadminPassword": "<generated password>",
  "kubeadminUsername": "kubeadmin"
}
```

You can find the cluster console URL by running the following command, and outputs a URL like `https://console-openshift-console.apps.<random>.<region>.aroapp.io/`.

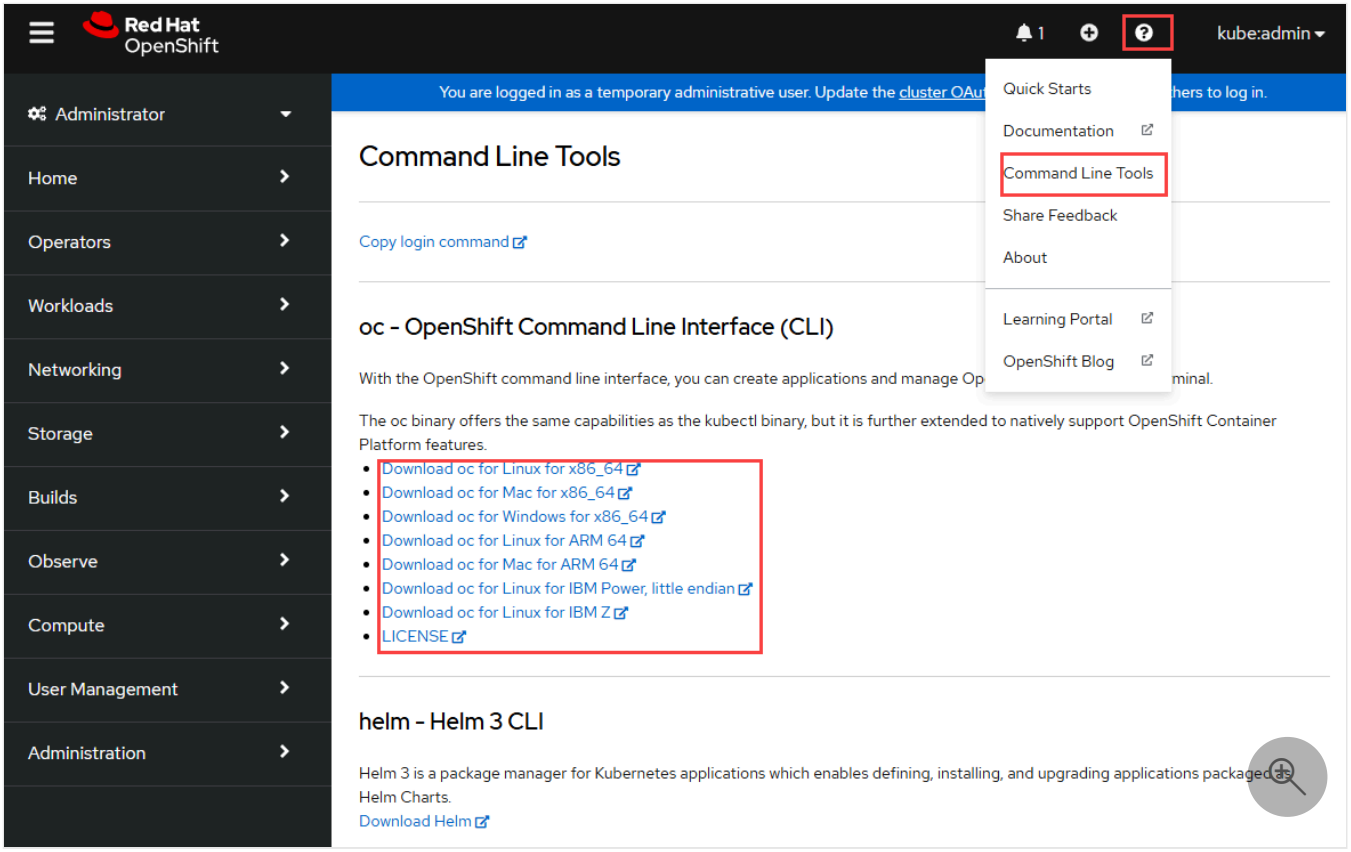
Azure CLI

```azurecli
az aro show \
  --name $CLUSTER \
  --resource-group $RESOURCEGROUP \
  --query "consoleProfile.url" --output tsv
```

Launch the console URL in a browser and sign in using the `kubeadmin` credentials.

Install the OpenShift CLI

After you're logged into the OpenShift Web Console, select the `?` at the top right and then on **Command Line Tools**. Download the release appropriate to your machine.



You can also download the latest release of the CLI appropriate to your machine.

If you're running the commands on the Azure Cloud Shell, download the latest OpenShift 4 CLI for Linux.

Azure CLI

```
cd ~
wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz

mkdir openshift
tar -zxvf openshift-client-linux.tar.gz -C openshift
echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

Connect using the OpenShift CLI

Retrieve the API server's address.

Azure CLI

```
apiServer=$(az aro show --resource-group $RESOURCEGROUP --name $CLUSTER --query apiserverProfile.url --output tsv)
```

Create the `kubevar` variable that stores the value of `kubeadminPassword` so that you don't need to type or copy and paste the credential.

Azure CLI

```
kubevar=$(az aro list-credentials --name $CLUSTER --resource-group $RESOURCEGROUP --query kubeadminPassword --output tsv)
```

Sign in to the OpenShift cluster's API server using the following command.

Azure CLI

```
oc login $apiServer --username kubeadmin --password $kubevar
```

After you sign in you should clear the `kubevar` variable's value.

Azure CLI

```
kubevar=""
```

Next steps

Quickstart: Delete an Azure Red Hat OpenShift 4 cluster

Quickstart: Delete an Azure Red Hat OpenShift 4 cluster

07/07/2025

In this Quickstart, you learn how to delete a Microsoft Azure Red Hat OpenShift cluster. When you delete a cluster, all the managed objects are deleted, but resources like the resource group, virtual network, and subnets must be manually deleted.

Prerequisites

This article requires Azure CLI version 2.6.0 or later. To find the version, run the `az --version` command. If you need to install or upgrade, see Install Azure CLI.

Sign in to Azure

If you're running the Azure CLI locally, run `az login` to sign in to Azure.

Azure CLI

```
az login
```

If you have access to multiple Azure subscriptions, run the following command and replace `{subscription ID}` with the subscription you want to use.

Azure CLI

```
az account set --subscription {subscription ID}
```

Delete the cluster

In previous articles to create and connect to a cluster, you created a resource group and cluster name. For the following variables, replace `<resourceGroupName>` and `<clusterName>` with the values you used to create your cluster:

Azure CLI

```
RESOURCEGROUP=<resourceGroupName>
CLUSTER=<clusterName>
```

Use these values in the following command to delete your cluster:

Azure CLI

```
az aro delete --resource-group $RESOURCEGROUP --name $CLUSTER
```

You're prompted to confirm if you're sure you want to perform this operation. After you confirm with `y`, it will take several minutes to delete the cluster. When the command finishes, the cluster is deleted and all the managed objects.

Delete the resource group

Resources like the resource group, virtual network, and subnets must be manually deleted.

Run the following command to delete the resource group and all the resources it contains. Respond with `y` to confirm you want to delete the resources.

Azure CLI

```
az group delete --name $RESOURCEGROUP
```

Next steps

Learn more about using OpenShift with the official Red Hat OpenShift documentation.

Azure Red Hat OpenShift FAQ

This article answers frequently asked questions (FAQs) about Microsoft Azure Red Hat OpenShift.

Installation and upgrade

Where can I find information about pricing and service level agreements?

For pricing information, see Azure Red Hat OpenShift pricing.

For Service Level Agreement (SLA) information, see Service Level Agreements for online services.

Which Azure regions are supported?

For a list of supported regions for Azure Red Hat OpenShift 4.x, see Available regions.

Can I migrate an existing cluster that uses a service principal to use a managed identity?

An existing cluster that uses a service principal can't be migrated to use a managed identity. You need to create a new cluster that uses a managed identity on an Azure Red Hat OpenShift cluster. For more information, see Create an Azure Red Hat OpenShift cluster with managed identities (preview).

What virtual machine sizes can I use?

For a list of supported virtual machine sizes for Azure Red Hat OpenShift 4, see Azure Red Hat OpenShift 4.0 support policy.

What is the maximum number of pods in an Azure Red Hat OpenShift cluster? What is the maximum number of pods per node in Azure Red Hat OpenShift?

The actual number of supported pods depends on an application's memory, CPU, and storage requirements.

Azure Red Hat OpenShift 4.x has a 250 pod-per-node limit and a 250 compute node limit. These limits cap the maximum number of pods supported in a cluster to 250×250 = 62,500. These limits are the same for clusters created using User Defined Routing (UDR) and running version 4.11 or higher.

Can a cluster have compute nodes across multiple Azure regions?

No. All nodes in an Azure Red Hat OpenShift cluster must originate in the same Azure region.

Can a cluster be deployed across multiple availability zones?

Yes. A cluster can be deployed across multiple availability zones automatically if your cluster is deployed to an Azure region that supports availability zones. For more information, see Availability zones.

Are control plane nodes abstracted away as they are with Azure Kubernetes Service (AKS)?

No. All resources, including the cluster control plane nodes, run in your customer subscription. These types of resources are put in a read-only resource group.

Does the cluster reside in a customer subscription?

The Azure Managed Application lives in a locked Resource Group with the customer subscription. Customers can view objects in that resource group but not modify them.

Is there any element in Azure Red Hat OpenShift shared with other customers? Or is everything independent?

Each Azure Red Hat OpenShift cluster is dedicated to a given customer and lives within the customer's subscription.

Are infrastructure nodes available?

Yes, Azure Red Hat OpenShift allows you to use infrastructure machine sets to create machines that only host infrastructure components, such as the default router, the integrated container registry, and the components for cluster metrics and monitoring. For more information, see Deploy infrastructure nodes in an Azure Red Hat OpenShift cluster.

How do I handle cluster upgrades?

For information on upgrades, maintenance, and supported versions, see the support lifecycle guide.

How are the host operating system and OpenShift software updated?

The host operating systems and OpenShift software are updated as Azure Red Hat OpenShift consumes minor release versions and patches from upstream OpenShift Container Platform.

What's the process to reboot the updated node?

Nodes are rebooted as a part of an upgrade.

Cluster operations

Can I use Prometheus to monitor my applications?

Prometheus comes preinstalled and configured for Azure Red Hat OpenShift 4.x clusters. Read more about cluster monitoring.

Can I use Prometheus to monitor metrics related to cluster health and capacity?

Yes, you can use Prometheus in Azure Red Hat OpenShift 4.x.

Can logs of underlying virtual machines be streamed out to a customer log analysis system?

Logs from underlying virtual machines are handled by the managed service and aren't exposed to customers.

How can a customer get access to metrics like CPU/memory at the node level to take action to scale, debug issues, etc.? I can't seem to run kubectl top on an Azure Red Hat OpenShift cluster.

For Azure Red Hat OpenShift 4.x clusters, the OpenShift web console contains all metrics at the node level. For more information, see the Red Hat documentation on viewing cluster information.

Is there a way to manage pod placement?

Customers have the ability to get nodes and view labels as the customer-admin.

Caution must be used when using specific labels:

- Hostname must not be used. Hostname gets rotated often with upgrades and updates and is guaranteed to change.
- If the customer has a request for specific labels or a deployment strategy, this could be accomplished. However, it would require engineering efforts, and it isn't supported today.

For more information, see Controlling pod placement.

Is the image registry available externally so I can use tools such as Jenkins?

For 4.x clusters, you need to expose a secure registry and configure authentication. For more information, see the following Red Hat documentation:

- Exposing a registry
- Accessing the registry

Can I move/migrate my cluster between Azure tenants?

Moving your cluster between tenants is currently unsupported.

Can I move my Azure Red Hat OpenShift clusters from the current Azure subscription to another?

Moving your cluster and its associated resources between subscriptions isn't supported.

Can I move my Azure Red Hat OpenShift clusters or infrastructure resources to other resource groups or rename them?

Moving or renaming your cluster and its associated resources isn't supported.

Networking

Can I deploy a cluster into an existing virtual network?

In 4.x clusters, you can deploy a cluster into an existing virtual network.

Is cross-namespace networking supported?

Customer and individual project admins can customize cross-namespace networking (including denying it) on a per-project basis using `NetworkPolicy` objects.

I'm trying to peer into a virtual network in a different subscription but getting Failed to get VNet CIDR error.

In the subscription that has the virtual network, make sure to register `Microsoft.ContainerService` provider with the following command: `az provider register -n Microsoft.ContainerService --wait`

Can we specify IP ranges for deployment on the private virtual network, avoiding clashes with other corporate virtual networks once peered?

In 4.x clusters, you can specify your own IP ranges.

Is the Software Defined Network module configurable?

The Software Defined Network is `openshift-ovs-networkpolicy` and isn't configurable.

What Azure Load balancer is used by Azure Red Hat OpenShift? Is it Standard or Basic and is it configurable?

Azure Red Hat OpenShift uses Standard Azure Load Balancer, and it isn't configurable.

Permissions

Can an admin manage users and quotas?

Yes. An Azure Red Hat OpenShift administrator can manage users and quotas in addition to accessing all user created projects.

Can I restrict a cluster to only certain Microsoft Entra users?

Yes. You can restrict which Microsoft Entra users can sign in to a cluster by configuring the Microsoft Entra Application. For details, see Restrict a Microsoft Entra app to a set of users.

Can I restrict users from creating projects?

Yes. Sign in to your cluster as an administrator and execute this command:

Bash

```
oc adm policy \
  remove-cluster-role-from-group self-provisioner \
  system:authenticated:oauth
```

For more information, see the OpenShift documentation on disabling self-provisioning for your cluster version: Disabling self-provisioning

Which UNIX rights (in IaaS) are available for master, infrastructure, and application nodes?

Node access is available through the cluster-admin role. For more information, see Kubernetes RBAC overview.

Which OCP rights do we have? Cluster-admin? Project-admin?

The cluster-admin role is available. For more information, see Kubernetes RBAC overview.

Which identity providers are available?

You configure your own identity provider. For more information, see the Red Hat documentation on configuring identity providers.

Storage

Is data on my cluster encrypted?

By default, data is encrypted at rest. The Azure Storage platform automatically encrypts your data before persisting it, and decrypts the data before retrieval. For more information, see Azure Storage Service Encryption for data at rest.

How are my storage accounts secured?

Storage accounts are set to private access only.

Storage accounts are encrypted (new clusters only). Existing clusters need to be re-created.

Storage accounts are created with general-purpose v2 for new clusters.

General-purpose v2 storage accounts support the latest Azure Storage features and incorporate all the functionality of general-purpose v1 and Blob storage accounts.

Storage accounts access is limited with firewall rules via Azure network security groups (NSGs), which filter network traffic to and from your storage accounts. For more information, see Azure network security groups overview.

Transport Layer Security (TLS) protocol version 1.2 provides secure communications, data privacy, and data integrity.

Is data stored in etcd encrypted on Azure Red Hat OpenShift?

Data isn't encrypted by default, but you can enable encryption. For more information, see the guide on encrypting etcd.

Can we choose any persistent storage solution, like OCS?

Azure Disk (Premium_LRS) is configured as the default storage class. For other storage providers, and for configuration details (including Azure File), see the Red Hat documentation on persistent storage.

Does Azure Red Hat OpenShift store any customer data outside of the cluster's region?

No. All data created in a cluster is maintained within the cluster's region.

Related content

- Azure Red Hat OpenShift 4.0 support policy
- Support lifecycle for Azure Red Hat OpenShift 4

Quickstart: Deploy an Azure Red Hat OpenShift cluster with an Azure Resource Manager template or Bicep file

Article • 02/25/2025

This article describes how to use either Azure Resource Manager template (ARM template) or Bicep to create an Azure Red Hat OpenShift cluster. You can deploy the Azure Red Hat OpenShift cluster with either PowerShell or the Azure command-line interface (Azure CLI).

An Azure Resource Manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax. You describe your intended deployment without writing the sequence of programming commands to create the deployment.

Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. In a Bicep file, you define the infrastructure you want to deploy to Azure, and then use that file throughout the development lifecycle to repeatedly deploy your infrastructure. Your resources are deployed in a consistent manner.

> **Note**
>
> For information on deploying Azure Red Hat OpenShift clusters using Terraform, see Microsoft.RedHatOpenShift openShiftClusters Terraform.

Prerequisites

- Install Azure CLI
- An Azure account with an active subscription is required. If you don't already have one, you can create an account for free.
- Ability to assign User Access Administrator and Contributor roles. If you lack this ability, contact your Microsoft Entra admin to manage roles.
- A Red Hat account. If you don't have one, you'll have to register for an account.

- A pull secret for your Azure Red Hat OpenShift cluster. Download the pull secret file from the Red Hat OpenShift Cluster Manager web site.
- If you want to run the Azure PowerShell code locally, Azure PowerShell.
- If you want to run the Azure CLI code locally:
  - A Bash shell (such as Git Bash, which is included in Git for Windows).
  - Azure CLI.

Create an ARM template or Bicep file

Choose either an Azure Resource Manager template (ARM template) or an Azure Bicep file. Then, you can deploy the template using either the Azure command line (azure-cli) or PowerShell.

Create an ARM template

The following example shows how your ARM template should look when configured for your Azure RedHat OpenShift cluster.

The template defines three Azure resources:

- **Microsoft.Network/virtualNetworks**
- **Microsoft.Network/virtualNetworks/providers/roleAssignments**
- **Microsoft.RedHatOpenShift/OpenShiftClusters**

More Azure Red Hat OpenShift template samples can be found on the Red Hat OpenShift web site.

Save the following example as *azuredeploy.json*:

JSON

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
      "location" : {
        "type": "string",
        "defaultValue": "eastus",
        "metadata": {
          "description": "Location"
        }
      },
      "domain": {
```

```
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "Domain Prefix"
            }
        },
        "pullSecret": {
            "type": "string",
            "metadata": {
                "description": "Pull secret from cloud.redhat.com. The json
should be input as a string"
            }
        },
        "clusterVnetName": {
            "type": "string",
            "defaultValue": "aro-vnet",
            "metadata": {
                "description": "Name of ARO vNet"
            }
        },
        "clusterVnetCidr": {
            "type": "string",
            "defaultValue": "10.100.0.0/15",
            "metadata": {
                "description": "ARO vNet Address Space"
            }
        },
        "workerSubnetCidr": {
            "type": "string",
            "defaultValue": "10.100.70.0/23",
            "metadata": {
                "description": "Worker node subnet address space"
            }
        },
        "masterSubnetCidr": {
            "type": "string",
            "defaultValue": "10.100.76.0/24",
            "metadata": {
                "description": "Master node subnet address space"
            }
        },
        "masterVmSize" : {
            "type": "string",
            "defaultValue": "Standard_D8s_v3",
            "metadata": {
                "description": "Master Node VM Type"
            }
        },
        "workerVmSize": {
            "type": "string",
            "defaultValue": "Standard_D4s_v3",
            "metadata": {
                "description": "Worker Node VM Type"
            }
        },
```

```
        "workerVmDiskSize": {
            "type" : "int",
            "defaultValue": 128,
            "minValue": 128,
            "metadata": {
                "description": "Worker Node Disk Size in GB"
            }
        },
        "workerCount": {
            "type": "int",
            "defaultValue": 3,
            "minValue": 3,
            "metadata": {
                "description": "Number of Worker Nodes"
            }
        },
        "podCidr": {
            "type": "string",
            "defaultValue": "10.128.0.0/14",
            "metadata": {
                "description": "Cidr for Pods"
            }
        },
        "serviceCidr": {
            "type": "string",
            "defaultValue": "172.30.0.0/16",
            "metadata": {
                "description": "Cidr of service"
            }
        },
        "clusterName" : {
          "type": "string",
          "metadata": {
            "description": "Unique name for the cluster"
          }
        },
        "tags": {
            "type": "object",
            "defaultValue" : {
                "env": "Dev",
                "dept": "Ops"
            },
            "metadata": {
                "description": "Tags for resources"
            }
        },
        "apiServerVisibility": {
            "type": "string",
            "allowedValues": [
                "Private",
                "Public"
            ],
            "defaultValue": "Public",
            "metadata": {
                "description": "Api Server Visibility"
```

```
            }
        },
        "ingressVisibility": {
            "type": "string",
            "allowedValues": [
                "Private",
                "Public"
            ],
            "defaultValue": "Public",
            "metadata": {
                "description": "Ingress Visibility"
            }
        },
        "aadClientId" : {
          "type": "string",
          "metadata": {
            "description": "The Application ID of an Azure Active Directory
client application"
          }
        },
        "aadObjectId": {
            "type": "string",
            "metadata": {
                "description": "The Object ID of an Azure Active Directory
client application"
            }
        },
        "aadClientSecret" : {
          "type":"securestring",
          "metadata": {
            "description": "The secret of an Azure Active Directory client
application"
          }
        },
        "rpObjectId": {
            "type": "String",
            "metadata": {
                "description": "The ObjectID of the Resource Provider Service
Principal"
            }
        }
    },
    "variables": {
        "contribRole": "[concat('/subscriptions/',
subscription().subscriptionId,
'/providers/Microsoft.Authorization/roleDefinitions/', 'b24988ac-6180-42a0-
ab88-20f7382dd24c')]"
    },
    "resources": [
        {
            "type": "Microsoft.Network/virtualNetworks",
            "apiVersion": "2020-05-01",
            "name": "[parameters('clusterVnetName')]",
            "location": "[parameters('location')]",
            "tags": "[parameters('tags')]",
```

```
            "properties": {
                "addressSpace": {
                "addressPrefixes": [
                    "[parameters('clusterVnetCidr')]"
                    ]
                },
                "subnets": [
                {
                    "name": "master",
                    "properties": {
                        "addressPrefix": "[parameters('masterSubnetCidr')]",
                        "serviceEndpoints": [
                            {
                                "service": "Microsoft.ContainerRegistry"
                            }
                        ],
                        "privateLinkServiceNetworkPolicies": "Disabled"
                    }
                },
                {
                    "name": "worker",
                    "properties": {
                        "addressPrefix": "[parameters('workerSubnetCidr')]",
                        "serviceEndpoints": [
                            {
                                "service": "Microsoft.ContainerRegistry"
                            }
                        ]
                    }
                }]
            }
        },
        {
            "type":
"Microsoft.Network/virtualNetworks/providers/roleAssignments",
            "apiVersion": "2018-09-01-preview",
            "name": "[concat(parameters('clusterVnetName'),
'/Microsoft.Authorization/', guid(resourceGroup().id, deployment().name,
parameters('aadObjectId')))]",
            "dependsOn": [
                "[resourceId('Microsoft.Network/virtualNetworks',
parameters('clusterVnetName'))]"
            ],
            "properties": {
                "roleDefinitionId": "[variables('contribRole')]",
                "principalId":"[parameters('aadObjectId')]"
            }
        },
        {
            "type":
"Microsoft.Network/virtualNetworks/providers/roleAssignments",
            "apiVersion": "2018-09-01-preview",
            "name": "[concat(parameters('clusterVnetName'),
'/Microsoft.Authorization/', guid(resourceGroup().id, deployment().name,
parameters('rpObjectId')))]",
```

```
            "dependsOn": [
                "[resourceId('Microsoft.Network/virtualNetworks',
parameters('clusterVnetName'))]"
            ],
            "properties": {
                "roleDefinitionId": "[variables('contribRole')]",
                "principalId":"[parameters('rpObjectId')]"
            }
        },
        {
            "type": "Microsoft.RedHatOpenShift/OpenShiftClusters",
            "apiVersion": "2020-04-30",
            "name": "[parameters('clusterName')]",
            "location": "[parameters('location')]",
            "tags": "[parameters('tags')]",
            "dependsOn": [
                "[resourceId('Microsoft.Network/virtualNetworks',
parameters('clusterVnetName'))]"
            ],
            "properties": {
                "clusterProfile": {
                    "domain": "[parameters('domain')]",
                    "resourceGroupId": "[concat('/subscriptions/',
subscription().subscriptionId,'/resourceGroups/aro-',
parameters('domain'))]",
                    "pullSecret": "[parameters('pullSecret')]"
                },
                "networkProfile": {
                    "podCidr": "[parameters('podCidr')]",
                    "serviceCidr": "[parameters('serviceCidr')]"
                },
                "servicePrincipalProfile": {
                    "clientId": "[parameters('aadClientId')]",
                    "clientSecret": "[parameters('aadClientSecret')]"
                },
                "masterProfile": {
                    "vmSize": "[parameters('masterVmSize')]",
                    "subnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('clusterVnetName'), 'master')]"
                },
                "workerProfiles": [
                    {
                        "name": "worker",
                        "vmSize": "[parameters('workerVmSize')]",
                        "diskSizeGB": "[parameters('workerVmDiskSize')]",
                        "subnetId": "
[resourceId('Microsoft.Network/virtualNetworks/subnets',
parameters('clusterVnetName'), 'worker')]",
                        "count": "[parameters('workerCount')]"
                    }
                ],
                "apiserverProfile": {
                    "visibility": "[parameters('apiServerVisibility')]"
                },
```

```
                "ingressProfiles": [
                    {
                        "name": "default",
                        "visibility": "[parameters('ingressVisibility')]"
                    }
                ]
            }
        }
    ],
    "outputs": {
         "clusterCredentials": {
            "type": "object",
            "value": "
[listCredentials(resourceId('Microsoft.RedHatOpenShift/OpenShiftClusters',
parameters('clusterName')), '2020-04-30')]"
         },
         "oauthCallbackURL": {
            "type": "string",
            "value": "[concat('https://oauth-openshift.apps.',
parameters('domain'), '.', parameters('location'),
'.aroapp.io/oauth2callback/AAD')]"
         }
    }
}
```

Deploy the azuredeploy.json template

The azuredeploy.json template is used to deploy an Azure Red Hat OpenShift cluster. The following parameters are required:

> **Note**
>
> For the `domain` parameter, specify the domain prefix that will be used as part of the auto-generated DNS name for OpenShift console and API servers. This prefix is also used as part of the name of the resource group that is created to host the cluster VMs.

Expand table

Property	Description	Valid Options	Default Value
`domain`	The domain prefix for the cluster.		none
`pullSecret`	The pull secret that you obtained from the Red Hat OpenShift Cluster Manager web site.		

Property	Description	Valid Options	Default Value
`clusterName`	The name of the cluster.		
`aadClientId`	The application ID (a GUID) of a Microsoft Entra client application.		
`aadObjectId`	The object ID (a GUID) of the service principal for the Microsoft Entra client application.		
`aadClientSecret`	The client secret of the service principal for the Microsoft Entra client application, as a secure string.		
`rpObjectId`	The object ID (a GUID) of the resource provider service principal.		

The template parameters below have default values. They can be specified, but they aren't explicitly required.

Expand table

Property	Description	Valid Options	Default Value
`location`	The location of the new ARO cluster. This location can be the same as or different from the resource group region.		eastus
`clusterVnetName`	The name of the virtual network for the ARO cluster.		aro-vnet
`clusterVnetCidr`	The address space of the ARO virtual network, in Classless Inter-Domain Routing (CIDR) notation.		10.100.0.0/15
`workerSubnetCidr`	The address space of the worker node subnet, in CIDR notation.		10.100.70.0/23
`masterSubnetCidr`	The address space of the control plane node subnet, in CIDR notation.		10.100.76.0/24
`masterVmSize`	The virtual machine type/size of the control plane node.		Standard_D8s_v3
`workerVmSize`	The virtual machine type/size of the worker node.		Standard_D4s_v3

Property	Description	Valid Options	Default Value
`workerVmDiskSize`	The disk size of the worker node, in gigabytes.		128
`workerCount`	The number of worker nodes.		3
`podCidr`	The address space of the pods, in CIDR notation.		10.128.0.0/14
`serviceCidr`	The address space of the service, in CIDR notation.		172.30.0.0/16
`tags`	A hash table of resource tags.		`@{env = 'Dev'; dept = 'Ops'}`
`apiServerVisibility`	The visibility of the API server (`Public` or `Private`).		Public
`ingressVisibility`	The ingress (entrance) visibility (`Public` or `Private`).		Public

The following sections provide instructions using PowerShell or Azure CLI.

PowerShell steps

Perform the following steps if you're using PowerShell.

Before you begin - PowerShell

Before running the commands in this article, you might need to run `Connect-AzAccount`. Check to determine whether you have connectivity to Azure before proceeding. To check whether you have connectivity, run `Get-AzContext` to verify whether you have access to an active Azure subscription.

> **Note**
>
> This template uses the pull secret text that was obtained from the Red Hat OpenShift Cluster Manager website. Before proceeding, ensure you have the pull secret saved locally as `pull-secret.txt`.

PowerShell

```
$rhosPullSecret= Get-Content .\pull-secret.txt -Raw # the pull secret text
that was obtained from the Red Hat OpenShift Cluster Manager website
```

Define the following parameters as environment variables - PowerShell

PowerShell

```
$resourceGroup="aro-rg"       # the new resource group for the cluster
$location="eastus"            # the location of the new ARO cluster
$domain="mydomain"            # the domain prefix for the cluster
$aroClusterName="cluster"     # the name of the cluster
```

Register the required resource providers - PowerShell

Register the following resource providers in your subscription: `Microsoft.RedHatOpenShift`, `Microsoft.Compute`, `Microsoft.Storage` and `Microsoft.Authorization`.

PowerShell

```
Register-AzResourceProvider -ProviderNamespace Microsoft.RedHatOpenShift
Register-AzResourceProvider -ProviderNamespace Microsoft.Compute
Register-AzResourceProvider -ProviderNamespace Microsoft.Storage
Register-AzResourceProvider -ProviderNamespace Microsoft.Authorization
```

Create the new resource group - PowerShell

PowerShell

```
New-AzResourceGroup -Name $resourceGroup -Location $location
```

Create a new service principal and assign roles - PowerShell

PowerShell

```
$suffix=Get-Random # random suffix for the Service Principal
$spDisplayName="sp-$resourceGroup-$suffix"
$azureADAppSp = New-AzADServicePrincipal -DisplayName $spDisplayName -Role
Contributor
```

```
New-AzRoleAssignment -ObjectId $azureADAppSp.Id -RoleDefinitionName 'User
Access Administrator' -ResourceGroupName $resourceGroup -ObjectType
'ServicePrincipal'
New-AzRoleAssignment -ObjectId $azureADAppSp.Id -RoleDefinitionName
'Contributor' -ResourceGroupName $resourceGroup -ObjectType
'ServicePrincipal'
```

Get the Service Principal password - PowerShell

PowerShell

```
$aadClientSecretDigest = ConvertTo-SecureString -String
$azureADAppSp.PasswordCredentials.SecretText -AsPlainText -Force
```

Get the service principal for the OpenShift resource provider - PowerShell

PowerShell

```
$rpOpenShift =  Get-AzADServicePrincipal -DisplayName 'Azure Red Hat
OpenShift RP' | Select-Object -ExpandProperty Id -Property Id -First 1
```

Check the parameters before deploying the cluster - PowerShell

PowerShell

```
# setup the parameters for the deployment
$templateParams = @{
    domain = $domain
    clusterName = $aroClusterName
    location = $location
    aadClientId = $azureADAppSp.AppId
    aadObjectId = $azureADAppSp.Id
    aadClientSecret = $aadClientSecretDigest
    rpObjectId = $rpOpenShift.Id
    pullSecret = $rhosPullSecret
}

Write-Verbose (ConvertTo-Json $templateParams) -Verbose
```

Deploy the Azure Red Hat OpenShift cluster using the ARM template - PowerShell

PowerShell

```
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroup
@templateParams `
    -TemplateFile azuredeploy.json
```

Connect to your cluster

To connect to your new cluster, review the steps in Connect to an Azure Red Hat OpenShift 4 cluster.

Clean up resources - PowerShell

Once you're done, run the following command to delete your resource group and all the resources you created in this article.

PowerShell

```
Remove-AzResourceGroup -Name $resourceGroup -Force
```

Azure CLI steps

Perform the following steps if you're using Azure CLI.

Before you begin - Azure CLI

You might need to run `az login` before running the commands in this article. Check whether you have connectivity to Azure before proceeding. To check whether you have connectivity, run `az account list` and verify that you have access to an active Azure subscription.

> ⓘ Note
>
> This template will use the pull secret text that was obtained from the Red Hat OpenShift Cluster Manager website. Before proceeding make sure you have that secret saved locally as `pull-secret.txt`.

Azure CLI

```
PULL_SECRET=$(cat pull-secret.txt)    # the pull secret text
```

Define the following parameters as environment variables - Azure CLI

Azure CLI

```
RESOURCEGROUP=aro-rg            # the new resource group for the cluster
LOCATION=eastus                 # the location of the new cluster
DOMAIN=mydomain                 # the domain prefix for the cluster
ARO_CLUSTER_NAME=aro-cluster    # the name of the cluster
```

Register the required resource providers - Azure CLI

Register the following resource providers in your subscription: `Microsoft.RedHatOpenShift`, `Microsoft.Compute`, `Microsoft.Storage` and `Microsoft.Authorization`.

Azure CLI

```
az provider register --namespace 'Microsoft.RedHatOpenShift' --wait
az provider register --namespace 'Microsoft.Compute' --wait
az provider register --namespace 'Microsoft.Storage' --wait
az provider register --namespace 'Microsoft.Authorization' --wait
```

Create the new resource group - Azure CLI

Azure CLI

```
az group create --name $RESOURCEGROUP --location $LOCATION
```

Create a service principal for the new Microsoft Entra application

- Azure CLI

Azure CLI

```
az ad sp create-for-rbac --name "sp-$RG_NAME-${RANDOM}" > app-service-principal.json
SP_CLIENT_ID=$(jq -r '.appId' app-service-principal.json)
SP_CLIENT_SECRET=$(jq -r '.password' app-service-principal.json)
SP_OBJECT_ID=$(az ad sp show --id $SP_CLIENT_ID | jq -r '.id')
```

Assign the Contributor role to the new service principal - Azure CLI

Azure CLI

```
az role assignment create \
    --role 'User Access Administrator' \
    --assignee-object-id $SP_OBJECT_ID \
    --scope $SCOPE \
    --assignee-principal-type 'ServicePrincipal'

az role assignment create \
    --role 'Contributor' \
    --assignee-object-id $SP_OBJECT_ID \
    --scope $SCOPE \
    --assignee-principal-type 'ServicePrincipal'
```

Get the service principal object ID for the OpenShift resource provider - Azure CLI

Azure CLI

```
ARO_RP_SP_OBJECT_ID=$(az ad sp list --display-name "Azure Red Hat OpenShift RP" --query [0].id -o tsv)
```

Deploy the cluster - Azure CLI

Azure CLI

```
az deployment group create \
    --name aroDeployment \
    --resource-group $RESOURCEGROUP \
    --template-file azuredeploy.json \
    --parameters location=$LOCATION \
    --parameters domain=$DOMAIN \
    --parameters pullSecret=$PULL_SECRET \
    --parameters clusterName=$ARO_CLUSTER_NAME \
    --parameters aadClientId=$SP_CLIENT_ID \
    --parameters aadObjectId=$SP_OBJECT_ID \
```

```
    --parameters aadClientSecret=$SP_CLIENT_SECRET \
    --parameters rpObjectId=$ARO_RP_SP_OBJECT_ID
```

Connect to your cluster - Azure CLI

To connect to your new cluster, review the steps in Connect to an Azure Red Hat OpenShift 4 cluster.

Clean up resources - Azure CLI

Once you're done, run the following command to delete your resource group and all the resources you created in this article.

Azure CLI

```
az aro delete --resource-group $RESOURCEGROUP --name $CLUSTER
```

> **Tip**
>
> Having issues? Let us know on GitHub by opening an issue in the **Azure Red Hat Openshift (ARO) repo**.

Next steps

In this article, you learned how to create an Azure Red Hat OpenShift cluster running OpenShift 4 using both ARM templates and Bicep.

Advance to the next article to learn how to configure the cluster for authentication using Microsoft Entra ID.

- Rotate service principal credentials for your Azure Red Hat OpenShift (ARO) Cluster
- Configure authentication with Microsoft Entra ID using the command line
- Configure authentication with Microsoft Entra ID using the Azure portal and OpenShift web consolei

Feedback

Update an Azure Red Hat OpenShift cluster

As part of the Microsoft Azure Red Hat OpenShift cluster lifecycle, you need to perform periodic updates to the latest version of the OpenShift platform. Updating your Azure Red Hat OpenShift clusters enables you to update to the latest features and functionalities and apply the latest security releases.

This article shows you how to update all components in an OpenShift cluster using the OpenShift web console, CLI, or the managed-upgrade-operator (MUO). For more details about OpenShift updates please see Understanding OpenShift updates

> **Important**
>
> Performing a Control Plane Only update is not supported for Azure Red Hat OpenShift and may result in cluster instability.

Prerequisites

- This article assumes you have access to an existing Azure Red Hat OpenShift cluster as a user with `admin` privileges.
- Make sure that the credentials for the service principal used for the cluster are valid/updated before starting the update. For more information, see Rotate service principal credentials for your Azure Red Hat OpenShift cluster.

Check for available cluster updates using the web console

1. From the left menu of the OpenShift web console, ensure you are in the **Administrator** perspective, which is the default when you sign as the kubeadmin.
2. Select the **Administration** tab.
3. Select **Cluster Settings** and open the **Details** tab. The version, update status, and channel are displayed. The channel isn't configured by default.
4. Select the **Channel** link, and at the prompt enter the desired update channel, for example `stable-4.19`. Once the desired channel is chosen, a graph showing available releases and channels is displayed. If the **Update Status** for your cluster shows **Updates Available**, you can update your cluster.

Update your cluster with the OpenShift web console

From the OpenShift web console in the previous step, set the **Channel** for the version that you want to update to, such as `stable-4.19`.

Selection a version to update to, and select **Update**. You see the update status change to: `Update to <product-version> in progress`. You can review the progress of the cluster update by watching the progress bars for the operators and nodes.

Schedule individual updates using the managed-upgrade-operator

Use the managed-upgrade-operator (MUO) to update your Azure Red Hat OpenShift cluster.

The managed-upgrade-operator manages automated cluster updates. The managed-upgrade-operator starts the cluster update, but it doesn't perform any activities of the cluster update process itself. The OpenShift Container Platform (OCP) is responsible for updating the clusters. The goal of the managed-upgrade-operator is to satisfy the operating conditions that a managed cluster must hold, both before and after starting the cluster update.

1. Prepare the configuration file, as shown in the following example for updating to OpenShift 4.19.

YAML

```yaml
apiVersion: upgrade.managed.openshift.io/v1alpha1
kind: UpgradeConfig
metadata:
  name: managed-upgrade-config
  namespace: openshift-managed-upgrade-operator
spec:
  type: "ARO"
  upgradeAt: "2025-09-08T03:20:00Z"
  PDBForceDrainTimeout: 60
  desired:
    channel: "stable-4.19"
    version: "4.19.15"
```

- `upgradeAT` is the time when the update occurs.
- `channel` is the channel the configuration file pulls from, according to the lifecycle policy. The channel used should be `stable-<version>` or `eus-<version>`.
- `version` is the version that you wish to update to, such as `4.19.15`.

2. Apply the configuration file. Replace `<file_name>` with your file's name.

Azure CLI

```
oc create -f <file_name>.yaml
```

Update your cluster using the CLI

Please see the following for Updating a cluster using the CLI.

Extended Update Support Add-on (EUS) Term 1 updates

When updating your cluster from one EUS version to another EUS version (ex: 4.16 to 4.18) you will need to update to the interim version and then to the target EUS version. For example, to update from 4.16 to 4.18, you must update to 4.17, then to 4.18. Control Plane Only updates are not supported. You must also select the relevant update channel for your target version, for example `eus-4.18`.

Next steps

- You can find information about available OpenShift Container Platform advisories and updates in the errata section of the Red Hat Customer Portal.

Last updated on 11/14/2025

Deploy a large Azure Red Hat OpenShift cluster

Article • 02/25/2025

This article provides the steps and best practices for deploying large scale Azure Red Hat OpenShift clusters up to 250 worker nodes. For clusters of that size, there are some recommendations regarding control plane nodes and infrastructure nodes.

> **⊗ Caution**
>
> Before deleting a cluster with more than 120 nodes, scale down the cluster to 120 nodes or less.

Recommendations

Control plane nodes

For clusters with over 100 worker nodes, the following virtual machine instance types (or similar, newer generation instance types) are recommended for control plane nodes:

- Standard_D32s_v3
- Standard_D32s_v4
- Standard_D32s_v5

Infrastructure nodes

For clusters with over 100 worker nodes, infrastructure nodes are required to separate cluster workloads (such as Prometheus) to minimize contention with other workloads. You should deploy three (3) infrastructure nodes per cluster for redundancy and scalability needs.

The following instance types are recommended for infrastructure nodes:

- Standard_E16as_v5
- Standard_E16s_v5

For instructions on configuring infrastructure nodes, see Deploy infrastructure nodes in an Azure Red Hat OpenShift cluster. This will be configured after cluster deployment.

Add IP addresses to the load balancer

Azure Red Hat OpenShift public clusters are created with a public load balancer that's used for outbound connectivity from inside the cluster. By default, one public IP address is configured on that public load balancer, and that limits the maximum node count of your cluster to 62. To be able to scale your cluster to the maximum supported number of 250 nodes, you need to assign multiple additional public IP addresses to the load balancer. You can configure up to 20 IP addresses per cluster. The outbound rules and frontend IP configurations are adjusted to accommodate the number of IP addresses.

For example, a cluster with 180 worker nodes needs at least (3) three IP addresses (180 nodes / 62 nodes per IP).

This can be accomplished as part of the cluster creation process or later, after the cluster is created.

Deploy a cluster

When deploying a large cluster, you must start with at most 50 worker nodes at creation time, then scale the cluster out to the desired number of worker nodes, up to 250 worker nodes.

> ⓘ **Note**
>
> While you can define up to 50 worker nodes at creation time, it's best to start with a small cluster (e.g, three (3) worker nodes) and then scale out to the desired number of worker nodes after the cluster is installed.

Follow the steps provided in Create an Azure Red Hat OpenShift cluster until the "Create the cluster" steps, then continue as instructed:

The sample command below using the Azure CLI can be used to deploy a cluster with Standard_D32s_v5 as the control plane nodes, requesting three public IP addresses, and defining nine worker nodes:

Azure CLI

```
az aro create \
    --resource-group $RESOURCEGROUP \
    --name $CLUSTER \
    --vnet aro-vnet \
    --master-subnet master-subnet \
    --worker-subnet worker-subnet \
```

```
    --master-vm-size Standard_D32s_v5 \
    --worker-count 9 \
    --lb-ip-count 3
```

To add IP addresses to the load balancer using the Azure CLI after the cluster is created, run the following command:

Azure CLI

```
az aro update
    --name <CLUSTER_NAME>
    –-resource-group <RESOURCE_GROUP>
    --lb-ip-count <PUBLIC_IP_COUNT>`
```

You can then configure the corresponding OpenShift MachineSets to obtain the number of worker nodes desired. See Manually scaling a compute machine set for more details.

Once the cluster is successfully installed, proceed to deploying infrastructure nodes as defined in the Infrastructure nodes section.

Feedback

Use Azure Spot Virtual Machines in an Azure Red Hat OpenShift (ARO) cluster

Article • 02/25/2025

This article provides the necessary details that allow you to configure your Azure Red Hat OpenShift cluster (ARO) to use Azure Spot Virtual Machines.

Using Azure Spot Virtual Machines allows you to take advantage of our unused capacity at a significant cost savings. At any point in time when Azure needs the capacity back, the Azure infrastructure will evict Azure Spot Virtual Machines. For more information around Spot Instances, see Spot Virtual Machines.

Before you begin

Before getting started, ensure that you have an Azure Red Hat Openshift cluster deployed. If you need an ARO cluster, see the ARO quickstart for a public cluster, or the private cluster tutorial for a private cluster. The steps to configure your cluster to use Spot VMs are the same for both private and public clusters.

An ARO cluster should always have at least three worker nodes that are non-Spot VMs, and three control nodes. An ARO cluster can't have any spot VM-based control nodes.

Add Spot VMs

Machine management in Azure Red Hat Openshift is accomplished by using MachineSet. MachineSet resources are groups of machines. MachineSets are to machines as ReplicaSets are to pods. If you need more machines or must scale them down, you change the *Replicas* field on the machine set to meet your compute need. To learn more, check out our OpenShift MachineSet documentation

The use of Spot VMs is specified by adding the `spotVMOptions` field within the template spec of a MachineSet. To get this MachineSet created, we will:

1. Get a copy of a MachineSet running on your cluster.
2. Create a modified MachineSet configuration.
3. Deploy this MachineSet to your cluster

First, connect to your OpenShift cluster using the CLI.

Azure CLI

```
oc login $apiServer -u kubeadmin -p <kubeadmin password>
```

Next, you'll list the MachineSets on your cluster. A default cluster will have 3 MachineSets deployed:

Azure CLI

```
oc get machinesets -n openshift-machine-api
```

The following shows a sample output from this command:

```
NAME                                DESIRED   CURRENT   READY
AVAILABLE   AGE
aro-cluster-5t2dj-worker-eastus1   1         1         1       1
2d22h
aro-cluster-5t2dj-worker-eastus2   1         1         1       1
2d22h
aro-cluster-5t2dj-worker-eastus3   1         1         1       1
2d22h
```

Next, you'll describe the MachineSet deployed. Replace <machineset> with one of the MachineSets listed above and output it to a file.

Azure CLI

```
oc get machineset <machineset> -n openshift-machine-api -o yaml >
spotmachineset.yaml
```

You'll need to change the following parameters in the MachineSet:

- `metadata.name`
- `spec.selector.matchLabels.machine.openshift.io/cluster-api-machineset`
- `spec.template.metadata.labels.machine.openshift.io/cluster-api-machineset`
- `spec.template.spec.providerSpec.value.spotVMOptions` (Add this field, and set it to `{}`.)

Below is an abridged example of Spot MachineSet YAML that highlights the key changes you need to make when basing a new Spot MachineSet on an existing worker MachineSet, including some additional information for context. (The example doesn't represent an entire, functional MachineSet; many fields have been omitted below.)

```
```

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: aro-cluster-abcd1-spot-eastus
spec:
  replicas: 2
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: aro-cluster-abcd1
      machine.openshift.io/cluster-api-machineset: aro-cluster-abcd1-spot-
eastus
  template:
    metadata:
        machine.openshift.io/cluster-api-machineset: aro-cluster-abcd1-spot-
eastus
    spec:
      providerSpec:
        value:
          spotVMOptions: {}
      taints:
        - effect: NoExecute
          key: spot
          value: 'true'
```

Once the file is updated, apply it.

Azure CLI

```
oc create -f spotmachineset.yaml
```

To validate that your MachineSet has been successfully created, run the following command:

Azure CLI

```
oc get machinesets -n openshift-machine-api
```

Here's a sample output. Your Machineset is ready once you have machines in the "Ready" state.

```
  NAME                                DESIRED   CURRENT   READY
AVAILABLE   AGE
aro-cluster-5t2dj-worker-eastus1      1         1         1       1
3d1h
aro-cluster-5t2dj-worker-eastus2      1         1         1       1
3d1h
```

```
aro-cluster-5t2dj-worker-eastus3           1         1         1       1
3d1h
spot                                       1         1         1       1
2m47s
```

Schedule interruptible workloads

It's recommended to add a taint to the Spot nodes to prevent non-interruptible nodes from being scheduled on them, and to add tolerations of this taint to any pods you want scheduled on them. You can taint the nodes via the MachineSet spec.

For example, you can add the following YAML to `spec.template.spec`:

```
      taints:
         - effect: NoExecute
           key: spot
           value: 'true'
```

This prevents pods from being scheduled on the resultant node unless they had a toleration for `spot='true'` taint, and it would evict any pods lacking that toleration.

To learn more about applying taints and tolerations, read Controlling pod placement using node taints.

Quota

Machines may go into a failed state due to quota issues if the quota for the machine type you're using is too low for a brief moment, even if it should eventually be enough (for example, one node is still deleting when another is being created). Because of this, it's recommended to set quota for the machine type you'll be using for Spot instances to be slightly higher than should be needed (maybe by 2*n, where n is the number of cores used by a machine). This overhead would avoid having to remedy failed machines, which, though relatively simple, is still manual intervention.

Node readiness

As is explained in the Spot VM documentation linked above, VMs go into Deallocated provisioning state when they're no longer available, or no longer available at the maximum price specified.

This will manifest itself in OpenShift as **Not Ready** nodes. The machines will remain healthy, in phase **Provisioned as node**.

They'll return to being **Ready** once the VMs are available again

Troubleshooting

Node stuck in Not Ready state, underlying VM deallocated

If a node is stuck for a long period of time in Not Ready state after its VM was deallocated, you can try deleting it, or deleting its corresponding OpenShift machine object.

Spot Machine stuck in Failed state

If a machine (OpenShift object) that uses a Spot VM is stuck in a Failed state, try deleting it manually. If it can't be deleted due to a 403 because the VM no longer exists, then edit the machine and remove the finalizers.

Feedback

Deploy infrastructure nodes in an Azure Red Hat OpenShift cluster

07/10/2025

Microsoft Azure Red Hat OpenShift allows you to use infrastructure machine sets to create machines that only host infrastructure components, such as the default router, the integrated container registry, and the components for cluster metrics and monitoring. These infrastructure machines don't incur OpenShift costs; they only incur Azure Compute costs.

In a production deployment, the recommendation is that you deploy three machine sets to hold infrastructure components. Each of these nodes can be deployed to different availability zones to increase availability. This type of configuration requires three different machines sets; one for each availability zone. For infrastructure node sizing guidance, see Recommended infrastructure practices.

Qualified workloads

The following infrastructure workloads don't incur Azure Red Hat OpenShift worker subscriptions:

- Kubernetes and Azure Red Hat OpenShift control plane services that run on masters
- The default router
- The integrated container image registry
- The HAProxy-based Ingress Controller
- The cluster metrics collection, or monitoring service, including components for monitoring user-defined projects
- Cluster-aggregated logging

> **Important**
>
> Running workloads other than the designated kinds on the infrastructure nodes might affect the Service Level Agreement (SLA) and the stability of the cluster.

Prerequisites

In order for Azure virtual machines added to a cluster to be recognized as infrastructure nodes, instead of worker nodes, and not be charged an OpenShift fee, the following criteria must be met:

- The nodes must be one of the following instance types only:
  - Standard_E4s_v5
  - Standard_E8s_v5
  - Standard_E16s_v5
  - Standard_E4as_v5
  - Standard_E8as_v5
  - Standard_E16as_v5
- There can be no more than three nodes. Any extra nodes are charged an OpenShift fee.
- The nodes must have an Azure tag of `node_role: infra`
- Only workloads designated for infrastructure nodes are allowed. All other workloads would consider these worker nodes and be subject to the fee. This designation might also invalidate the SLA and compromise the stability of the cluster.

Create infrastructure machine sets

1. Use the manifest definition template to create the manifest definition for your infrastructure machine set.
2. Replace all fields in between angle brackets (`<>`) with your specific values.

   For example, replace `location: <REGION>` with `location: westus2`
3. To get the required values for the manifest definition template, see Commands and values.
4. Create the machine set with the following command: `oc create -f <machine-set-filename.yaml>`
5. To verify the creation of the machine set, run the following command: `oc get machineset -n openshift-machine-api`

   The output of the verification command should look similar to following values:

   Output

   ```
   NAME                              DESIRED     CURRENT   READY   AVAILABLE   AGE
   ok0608-vkxvw-infra-westus21       1           1         1       1           165M
   ok0608-vkxvw-worker-westus21      1           1         1       1
   ```

```
4H24M
ok0608-vkxvw-worker-westus22    1           1        1       1
4H24M
ok0608-vkxvw-worker-westus23    1           1        1       1
4H24M
```

Manifest definition template

The following template was used in the previous procedure to create the manifest definition for your infrastructure machine set.

YAML

```yaml
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
    machine.openshift.io/cluster-api-machine-role: infra
    machine.openshift.io/cluster-api-machine-type: infra
  name: <INFRASTRUCTURE_ID>-infra-<REGION><ZONE>
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
      machine.openshift.io/cluster-api-machineset: <INFRASTRUCTURE_ID>-infra-
<REGION><ZONE>
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
        machine.openshift.io/cluster-api-machine-role: infra
        machine.openshift.io/cluster-api-machine-type: infra
        machine.openshift.io/cluster-api-machineset: <INFRASTRUCTURE_ID>-infra-
<REGION><ZONE>
    spec:
      metadata:
        creationTimestamp: null
        labels:
          machine.openshift.io/cluster-api-machineset: <OPTIONAL: Specify the
machine set name to enable the use of availability sets. This setting only applies
to new compute machines.>
          node-role.kubernetes.io/infra: ''
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
```

```
          image:
            offer: aro4
            publisher: azureopenshift
            sku: <SKU>
            version: <VERSION>
          kind: AzureMachineProviderSpec
          location: <REGION>
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: <NETWORK_RESOURCE_GROUP>
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          resourceGroup: <CLUSTER_RESOURCE_GROUP>
          tags:
            node_role: infra
          subnet: <SUBNET_NAME>
          userDataSecret:
            name: worker-user-data
          vmSize: <Standard_E4s_v5, Standard_E8s_v5, Standard_E16s_v5>
          vnet: <VNET_NAME>
          zone: <ZONE>
      taints:
      - key: node-role.kubernetes.io/infra
        effect: NoSchedule
```

Commands and values

The following are some common commands and values that are used to create and run the template.

List all machine sets:

Console

```
oc get machineset -n openshift-machine-api
```

Get details for a specific machine set:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o yaml
```

Cluster resource group:

Console

```
oc get infrastructure cluster -o
jsonpath='{.status.platformStatus.azure.resourceGroupName}'
```

Network resource group:

Console

```
oc get infrastructure cluster -o
jsonpath='{.status.platformStatus.azure.networkResourceGroupName}'
```

Infrastructure ID:

Console

```
oc get infrastructure cluster -o jsonpath='{.status.infrastructureName}'
```

Region:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o
jsonpath='{.spec.template.spec.providerSpec.value.location}'
```

SKU:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o
jsonpath='{.spec.template.spec.providerSpec.value.image.sku}'
```

Subnet:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o
jsonpath='{.spec.template.spec.providerSpec.value.subnet}'
```

Version:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o
```

```
jsonpath='{.spec.template.spec.providerSpec.value.image.version}'
```

Virtual network:

Console

```
oc get machineset <machineset_name> -n openshift-machine-api -o
jsonpath='{.spec.template.spec.providerSpec.value.vnet}'
```

Moving workloads to the new infrastructure nodes

Use the following instructions to move your infrastructure workloads to the infrastructure nodes previously created.

Ingress

Use this procedure for any other ingress controllers you might have in the cluster. If your application has high ingress resource requirements, it might be better to spread them across worker nodes or a dedicated machine set.

1. Set the `nodePlacement` on the `ingresscontroller` to `node-role.kubernetes.io/infra` and increase the `replicas` to match the number of infrastructure nodes:

   Console

   ```
   oc patch -n openshift-ingress-operator ingresscontroller default --type=merge
   \
    -p='{"spec":{"replicas":3,"nodePlacement":{"nodeSelector":{"matchLabels":
   {"node-role.kubernetes.io/infra":""}},"tolerations":
   [{"effect":"NoSchedule","key":"node-
   role.kubernetes.io/infra","operator":"Exists"}]}}}'
   ```

2. Verify that the Ingress Controller Operator is starting pods on the new infrastructure nodes:

   Console

   ```
   oc -n openshift-ingress get pods -o wide
   ```

   Output

   ```
   NAME                              READY   STATUS        RESTARTS   AGE   IP
   NODE                                                    NOMINATED NODE
   ```

```
READINESS GATES
router-default-69f58645b7-6xkvh   1/1     Running       0          66s
10.129.6.6    cz-cluster-hsmtw-infra-aro-machinesets-eastus-3-l6dqw   <none>
<none>
router-default-69f58645b7-vttqz   1/1     Running       0          66s
10.131.4.6    cz-cluster-hsmtw-infra-aro-machinesets-eastus-1-vr56r   <none>
<none>
router-default-6cb5ccf9f5-xjgcp   1/1     Terminating   0          23h
10.131.0.11   cz-cluster-hsmtw-worker-eastus2-xj9qx                   <none>
<none>
```

Registry

1. Set the `nodePlacement` on the registry to `node-role.kubernetes.io/infra`:

Console

```
oc patch configs.imageregistry.operator.openshift.io/cluster --type=merge \
-p='{"spec":{"affinity":{"podAntiAffinity":
{"preferredDuringSchedulingIgnoredDuringExecution":[{"podAffinityTerm":
{"namespaces":["openshift-image-
registry"],"topologyKey":"kubernetes.io/hostname"},"weight":100}]}},"logLevel
":"Normal","managementState":"Managed","nodeSelector":{"node-
role.kubernetes.io/infra":""},"tolerations":
[{"effect":"NoSchedule","key":"node-
role.kubernetes.io/infra","operator":"Exists"}]}}'
```

2. Verify that the Registry Operator is starting pods on the new infrastructure nodes:

Console

```
oc -n openshift-image-registry get pods -l "docker-registry" -o wide
```

Output

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
NODE                                                NOMINATED NODE
READINESS GATES
image-registry-84cbd76d5d-cfsw7   1/1     Running   0          3h46m
10.128.6.7   cz-cluster-hsmtw-infra-aro-machinesets-eastus-2-kljml   <none>
<none>
image-registry-84cbd76d5d-p2jf9   1/1     Running   0          3h46m
10.129.6.7   cz-cluster-hsmtw-infra-aro-machinesets-eastus-3-l6dqw   <none>
<none>
```

Cluster monitoring

1. Configure the cluster monitoring stack to use the infrastructure nodes.

   This overrides any other customizations to the cluster monitoring stack, so you might want to merge your existing customizations before running the command.

YAML

```yaml
cat << EOF | oc apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |+
    alertmanagerMain:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    prometheusK8s:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    prometheusOperator: {}
    grafana:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    kubeStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    telemeterClient:
      nodeSelector:
```

```
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    openshiftStateMetrics:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
    thanosQuerier:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      tolerations:
        - effect: "NoSchedule"
          key: "node-role.kubernetes.io/infra"
          operator: "Exists"
EOF
```

2. Verify that the OpenShift Monitoring Operator is starting pods on the new infrastructure nodes. Note that some nodes, such as `prometheus-operator`, remain on master nodes.

Console

```
oc -n openshift-monitoring get pods -o wide
```

Output

```
NAME                                           READY   STATUS    RESTARTS
AGE     IP            NODE
NOMINATED NODE   READINESS GATES
alertmanager-main-0                            6/6     Running   0
2m14s   10.128.6.11   cz-cluster-hsmtw-infra-aro-machinesets-eastus-2-kljml
<none>           <none>
alertmanager-main-1                            6/6     Running   0
2m46s   10.131.4.11   cz-cluster-hsmtw-infra-aro-machinesets-eastus-1-vr56r
<none>           <none>
cluster-monitoring-operator-5bbfd998c6-m9w62   2/2     Running   0
28h     10.128.0.23   cz-cluster-hsmtw-master-1
<none>           <none>
grafana-599d4b948c-btlp2                       3/3     Running   0
2m48s   10.131.4.10   cz-cluster-hsmtw-infra-aro-machinesets-eastus-1-vr56r
<none>           <none>
kube-state-metrics-574c5bfdd7-f7fjk            3/3     Running   0
2m49s   10.131.4.8    cz-cluster-hsmtw-infra-aro-machinesets-eastus-1-vr56r
<none>           <none>
```

DNS

1. Allow the DNS pods to run on the infrastructure nodes.

   Console

   ```
   oc edit dns.operator/default
   ```

   YAML

   ```yaml
   apiVersion: operator.openshift.io/v1
   kind: DNS
   metadata:
   name: default
   spec:
   nodePlacement:
     tolerations:
     - operator: Exists
   ```

2. Verify that DNS pods are scheduled onto all `infra` nodes.

   Console

   ```
   oc get ds/dns-default -n openshift-dns
   ```

   Output

   ```
   NAME          DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
   SELECTOR            AGE
   dns-default   7         7         7       7            7
   kubernetes.io/os=linux   35d
   ```

Related content

To upgrade your cluster, see Upgrade an Azure Red Hat OpenShift cluster.

Use GPU workloads with Azure Red Hat OpenShift

Article • 02/25/2025

This article shows you how to use NVIDIA GPU workloads with Azure Red Hat OpenShift (ARO).

Prerequisites

- OpenShift CLI
- jq, moreutils, and gettext package
- Azure Red Hat OpenShift 4.10

If you need to install an ARO cluster, see Tutorial: Create an Azure Red Hat OpenShift 4 cluster. ARO clusters must be version 4.10.x or higher.

> **Note**
>
> As of ARO 4.10, it is no longer necessary to set up entitlements to use the NVIDIA Operator. This has greatly simplified the setup of the cluster for GPU workloads.

Linux:

```bash
sudo dnf install jq moreutils gettext
```

macOS

```bash
brew install jq moreutils gettext
```

Request GPU quota

All GPU quotas in Azure are 0 by default. You will need to sign in to the Azure portal and request GPU quota. Due to competition for GPU workers, you may have to provision an ARO cluster in a region where you can actually reserve GPU.

ARO supports the following GPU workers:

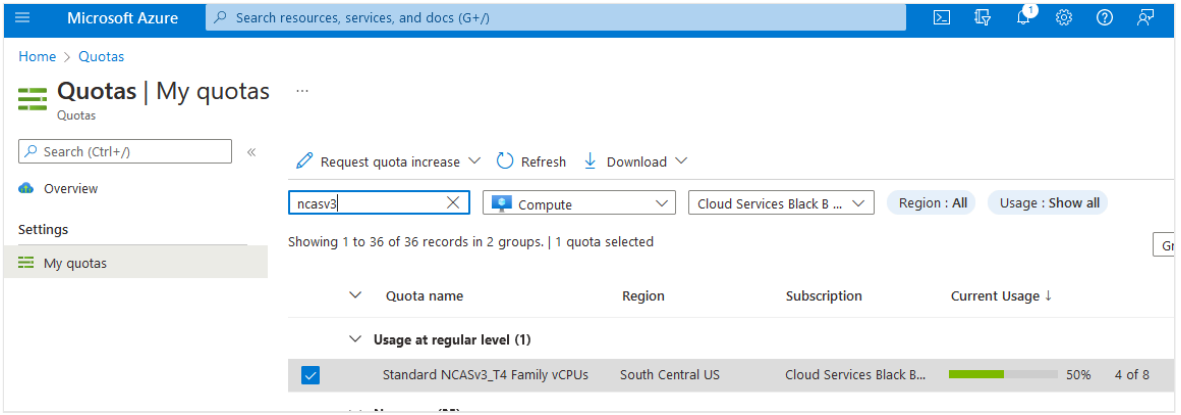
- NC4as T4 v3
- NC6s v3
- NC8as T4 v3
- NC12s v3
- NC16as T4 v3
- NC24s v3
- NC24rs v3
- NC64as T4 v3

The following instances are also supported in additional MachineSets:

- Standard_ND96asr_v4
- NC24ads_A100_v4
- NC48ads_A100_v4
- NC96ads_A100_v4
- ND96amsr_A100_v4

> **Note**
>
> When requesting quota, remember that Azure is per core. To request a single NC4as T4 v3 node, you will need to request quota in groups of 4. If you wish to request an NC16as T4 v3, you will need to request quota of 16.

1. Sign in to Azure portal.
2. Enter **quotas** in the search box, then select **Compute**.
3. In the search box, enter **NCAsv3_T4**, check the box for the region your cluster is in, and then select **Request quota increase**.
4. Configure quota.



Sign in to your ARO cluster

Sign in to OpenShift with a user account with cluster-admin privileges. The example below uses an account named **kubadmin**:

```
Bash

oc login <apiserver> -u kubeadmin -p <kubeadminpass>
```

Pull secret (conditional)

Update your pull secret to make sure you can install operators and connect to cloud.redhat.com.

> **Note**
>
> Skip this step if you have already recreated a full pull secret with cloud.redhat.com enabled.

1. Log into to cloud.redhat.com.
2. Browse to https://cloud.redhat.com/openshift/install/azure/aro-provisioned.
3. Select **Download pull secret** and save the pull secret as `pull-secret.txt`.

   > **Important**
   >
   > The remaining steps in this section must be run in the same working directory as `pull-secret.txt`.

4. Export the existing pull secret.

   ```
   Bash

   oc get secret pull-secret -n openshift-config -o json | jq -r
   '.data.".dockerconfigjson"' | base64 --decode > export-pull.json
   ```

5. Merge the downloaded pull secret with the system pull secret to add `cloud.redhat.com`.

   ```
   Bash
   ```

```
jq -s '.[0] * .[1]' export-pull.json pull-secret.txt | tr -d "\n\r" >
new-pull-secret.json
```

6. Upload the new secret file.

```
Bash

oc set data secret/pull-secret -n openshift-config --from-
file=.dockerconfigjson=new-pull-secret.json
```

> You may need to wait about 1 hour for everything to sync up with cloud.redhat.com.

7. Delete secrets.

```
Bash

rm pull-secret.txt export-pull.json new-pull-secret.json
```

GPU machine set

ARO uses Kubernetes MachineSet to create machine sets. The procedure below explains how to export the first machine set in a cluster and use that as a template to build a single GPU machine.

1. View existing machine sets.

   For ease of setup, this example uses the first machine set as the one to clone to create a new GPU machine set.

```
Bash

MACHINESET=$(oc get machineset -n openshift-machine-api -
o=jsonpath='{.items[0]}' | jq -r '[.metadata.name] | @tsv')
```

2. Save a copy of the example machine set.

```
Bash

oc get machineset -n openshift-machine-api $MACHINESET -o json >
gpu_machineset.json
```

3. Change the `.metadata.name` field to a new unique name.

```
Bash

jq '.metadata.name = "nvidia-worker-<region><az>"' gpu_machineset.json| sponge gpu_machineset.json
```

4. Ensure `spec.replicas` matches the desired replica count for the machine set.

```
Bash

jq '.spec.replicas = 1' gpu_machineset.json| sponge gpu_machineset.json
```

5. Change the `.spec.selector.matchLabels.machine.openshift.io/cluster-api-machineset` field to match the `.metadata.name` field.

```
Bash

jq '.spec.selector.matchLabels."machine.openshift.io/cluster-api-machineset" = "nvidia-worker-<region><az>"' gpu_machineset.json| sponge gpu_machineset.json
```

6. Change the `.spec.template.metadata.labels.machine.openshift.io/cluster-api-machineset` to match the `.metadata.name` field.

```
Bash

jq '.spec.template.metadata.labels."machine.openshift.io/cluster-api-machineset" = "nvidia-worker-<region><az>"' gpu_machineset.json| sponge gpu_machineset.json
```

7. Change the `spec.template.spec.providerSpec.value.vmSize` to match the desired GPU instance type from Azure.

   The machine used in this example is **Standard_NC4as_T4_v3**.

```
Bash

jq '.spec.template.spec.providerSpec.value.vmSize = "Standard_NC4as_T4_v3"' gpu_machineset.json | sponge gpu_machineset.json
```

8. Change the `spec.template.spec.providerSpec.value.zone` to match the desired zone from Azure.

```
Bash

jq '.spec.template.spec.providerSpec.value.zone = "1"'
gpu_machineset.json | sponge gpu_machineset.json
```

9. Delete the `.status` section of the yaml file.

```
Bash

jq 'del(.status)' gpu_machineset.json | sponge gpu_machineset.json
```

10. Verify the other data in the yaml file.

Ensure the correct SKU is set

Depending on the image used for the machine set, both values for `image.sku` and `image.version` must be set accordingly. This is to ensure if generation 1 or 2 virtual machine for Hyper-V will be used. See here for more information.

Example:

If using `Standard_NC4as_T4_v3`, both versions are supported. As mentioned in Feature support. In this case, no changes are required.

If using `Standard_NC24ads_A100_v4`, only **Generation 2 VM** is supported. In this case, the `image.sku` value must follow the equivalent `v2` version of the image that corresponds to the cluster's original `image.sku`. For this example, the value will be `v410-v2`.

This can be found using the following command:

```
Bash

az vm image list --architecture x64 -o table --all --offer aro4 --publisher
azureopenshift
```

```

Filtered output:

SKU      VERSION
-------  ---------------
v410-v2  410.84.20220125
aro_410  410.84.20220125
```

If the cluster was created with the base SKU image `aro_410`, and the same value is kept in the machine set, it will fail with the following error:

```
failure sending request for machine myworkernode: cannot create vm:
compute.VirtualMachinesClient#CreateOrUpdate: Failure sending request:
StatusCode=400 -- Original Error: Code="BadRequest" Message="The selected VM
size 'Standard_NC24ads_A100_v4' cannot boot Hypervisor Generation '1'.
```

Create GPU machine set

Use the following steps to create the new GPU machine. It may take 10-15 minutes to provision a new GPU machine. If this step fails, sign in to Azure portal and ensure there are no availability issues. To do so, go to **Virtual Machines** and search for the worker name you created previously to see the status of VMs.

1. Create the GPU Machine set.

   ```Bash
   oc create -f gpu_machineset.json
   ```

   This command will take a few minutes to complete.

2. Verify GPU machine set.

   Machines should be deploying. You can view the status of the machine set with the following commands:

   ```Bash
   oc get machineset -n openshift-machine-api
   oc get machine -n openshift-machine-api
   ```

   Once the machines are provisioned (which could take 5-15 minutes), machines will show as nodes in the node list:

   ```Bash
   oc get nodes
   ```

   You should see a node with the `nvidia-worker-southcentralus1` name that was created previously.

Install NVIDIA GPU Operator

This section explains how to create the `nvidia-gpu-operator` namespace, set up the operator group, and install the NVIDIA GPU operator.

1. Create NVIDIA namespace.

```yaml
cat <<EOF | oc apply -f -
apiVersion: v1
kind: Namespace
metadata:
  name: nvidia-gpu-operator
EOF
```

2. Create Operator Group.

```yaml
cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: nvidia-gpu-operator-group
  namespace: nvidia-gpu-operator
spec:
 targetNamespaces:
 - nvidia-gpu-operator
EOF
```

3. Get the latest NVIDIA channel using the following command:

```bash
CHANNEL=$(oc get packagemanifest gpu-operator-certified -n openshift-marketplace -o jsonpath='{.status.defaultChannel}')
```

> ⓘ **Note**
>
> If your cluster was created without providing the pull secret, the cluster won't include samples or operators from Red Hat or from certified partners. This will result in the following error message:
>
> *Error from server (NotFound): packagemanifests.packages.operators.coreos.com "gpu-operator-certified" not found.*

To add your Red Hat pull secret on an Azure Red Hat OpenShift cluster, follow this guidance.

1. Get latest NVIDIA package using the following command:

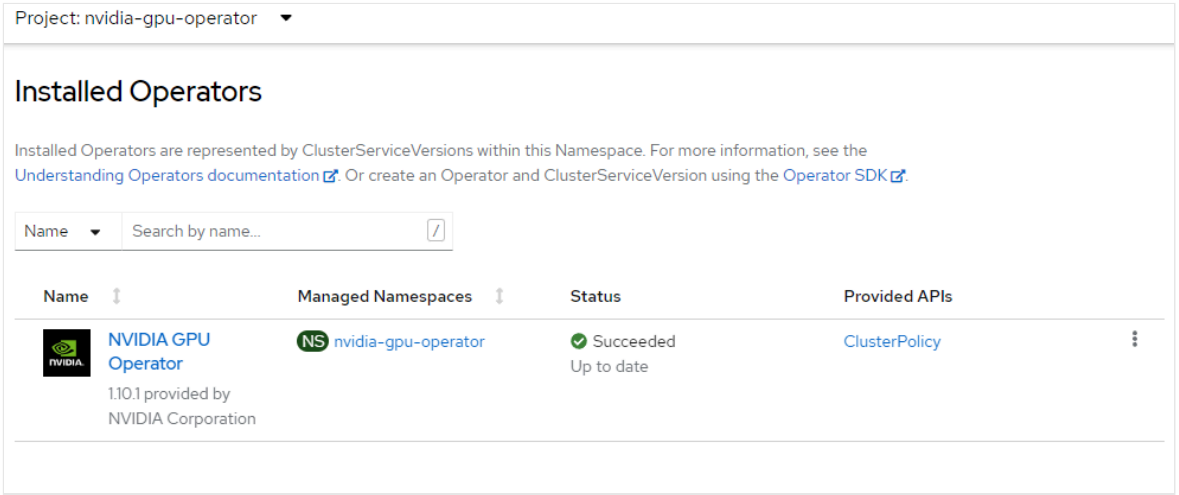
```bash
PACKAGE=$(oc get packagemanifests/gpu-operator-certified -n openshift-marketplace -ojson | jq -r '.status.channels[] | select(.name == "'$CHANNEL'") | .currentCSV')
```

2. Create Subscription.

```yaml
envsubst  <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: gpu-operator-certified
  namespace: nvidia-gpu-operator
spec:
  channel: "$CHANNEL"
  installPlanApproval: Automatic
  name: gpu-operator-certified
  source: certified-operators
  sourceNamespace: openshift-marketplace
  startingCSV: "$PACKAGE"
EOF
```

3. Wait for Operator to finish installing.

   Don't proceed until you have verified that the operator has finished installing. Also, ensure that your GPU worker is online.



Install node feature discovery operator

The node feature discovery operator will discover the GPU on your nodes and appropriately label the nodes so you can target them for workloads.

This example installs the NFD operator into the `openshift-ndf` namespace and creates the "subscription" which is the configuration for NFD.

Official Documentation for Installing Node Feature Discovery Operator.

1. Set up `Namespace`.

YAML

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-nfd
EOF
```

2. Create `OperatorGroup`.

YAML

```
cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  generateName: openshift-nfd-
  name: openshift-nfd
  namespace: openshift-nfd
EOF
```

3. Create `Subscription`.

YAML

```
cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: nfd
  namespace: openshift-nfd
spec:
  channel: "stable"
  installPlanApproval: Automatic
  name: nfd
  source: redhat-operators
```

```
  sourceNamespace: openshift-marketplace
EOF
```

4. Wait for Node Feature discovery to complete installation.

   You can log in to your OpenShift console to view operators or simply wait a few minutes. Failure to wait for the operator to install will result in an error in the next step.

5. Create NFD Instance.

YAML

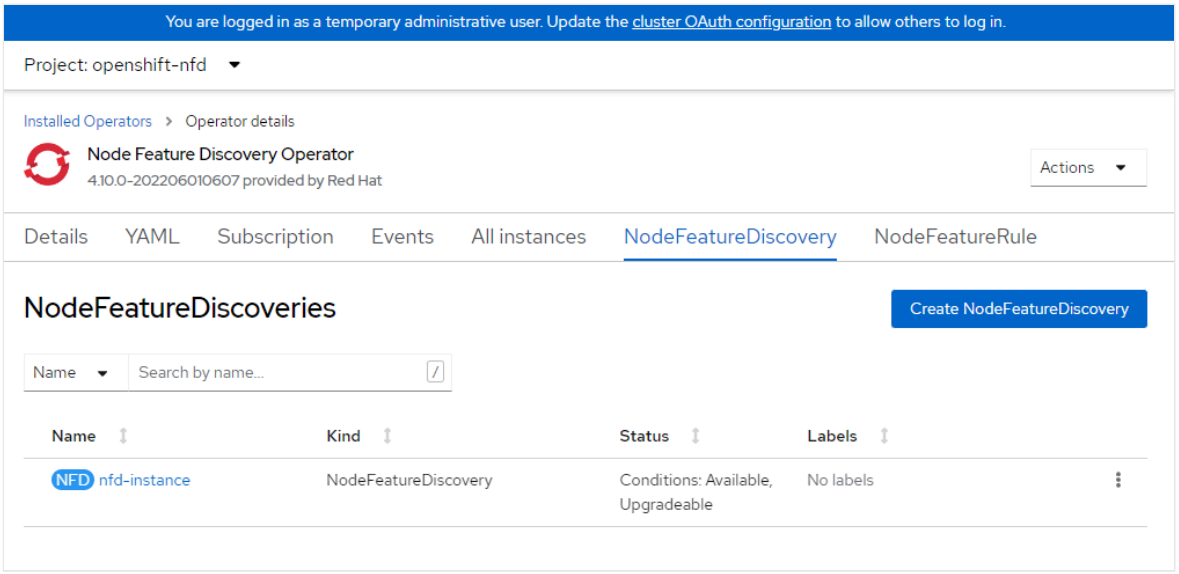
```
cat <<EOF | oc apply -f -
kind: NodeFeatureDiscovery
apiVersion: nfd.openshift.io/v1
metadata:
  name: nfd-instance
  namespace: openshift-nfd
spec:
  customConfig:
    configData: |
      #    - name: "more.kernel.features"
      #      matchOn:
      #      - loadedKMod: ["example_kmod3"]
      #    - name: "more.features.by.nodename"
      #      value: customValue
      #      matchOn:
      #      - nodename: ["special-.*-node-.*"]
  operand:
    image: >-
      registry.redhat.io/openshift4/ose-node-feature-
discovery@sha256:07658ef3df4b264b02396e67af813a52ba416b47ab6e1d2d08025a
350ccd2b7b
    servicePort: 12000
  workerConfig:
    configData: |
      core:
      #  labelWhiteList:
      #  noPublish: false
        sleepInterval: 60s
      #  sources: [all]
      #  klog:
      #    addDirHeader: false
      #    alsologtostderr: false
      #    logBacktraceAt:
      #    logtostderr: true
      #    skipHeaders: false
      #    stderrthreshold: 2
      #    v: 0
      #    vmodule:
      ##   NOTE: the following options are not dynamically run-time
```

```
      ##          configurable and require a nfd-worker restart to take
effect
      ##          after being changed
      #    logDir:
      #    logFile:
      #    logFileMaxSize: 1800
      #    skipLogHeaders: false
      sources:
      #  cpu:
      #    cpuid:
      ##     NOTE: attributeWhitelist has priority over
attributeBlacklist
      #      attributeBlacklist:
      #        - "BMI1"
      #        - "BMI2"
      #        - "CLMUL"
      #        - "CMOV"
      #        - "CX16"
      #        - "ERMS"
      #        - "F16C"
      #        - "HTT"
      #        - "LZCNT"
      #        - "MMX"
      #        - "MMXEXT"
      #        - "NX"
      #        - "POPCNT"
      #        - "RDRAND"
      #        - "RDSEED"
      #        - "RDTSCP"
      #        - "SGX"
      #        - "SSE"
      #        - "SSE2"
      #        - "SSE3"
      #        - "SSE4.1"
      #        - "SSE4.2"
      #        - "SSSE3"
      #      attributeWhitelist:
      #  kernel:
      #    kconfigFile: "/path/to/kconfig"
      #    configOpts:
      #      - "NO_HZ"
      #      - "X86"
      #      - "DMI"
        pci:
          deviceClassWhitelist:
            - "0200"
            - "03"
            - "12"
          deviceLabelFields:
      #      - "class"
            - "vendor"
      #      - "device"
      #      - "subsystem_vendor"
      #      - "subsystem_device"
      #  usb:
```

```
      #    deviceClassWhitelist:
      #      - "0e"
      #      - "ef"
      #      - "fe"
      #      - "ff"
      #    deviceLabelFields:
      #      - "class"
      #      - "vendor"
      #      - "device"
      #  custom:
      #    - name: "my.kernel.feature"
      #      matchOn:
      #        - loadedKMod: ["example_kmod1", "example_kmod2"]
      #    - name: "my.pci.feature"
      #      matchOn:
      #        - pciId:
      #            class: ["0200"]
      #            vendor: ["15b3"]
      #            device: ["1014", "1017"]
      #        - pciId :
      #            vendor: ["8086"]
      #            device: ["1000", "1100"]
      #    - name: "my.usb.feature"
      #      matchOn:
      #        - usbId:
      #          class: ["ff"]
      #          vendor: ["03e7"]
      #          device: ["2485"]
      #        - usbId:
      #          class: ["fe"]
      #          vendor: ["1a6e"]
      #          device: ["089a"]
      #    - name: "my.combined.feature"
      #      matchOn:
      #        - pciId:
      #            vendor: ["15b3"]
      #            device: ["1014", "1017"]
      #          loadedKMod : ["vendor_kmod1", "vendor_kmod2"]
EOF
```

6. Verify that NFD is ready.

   The status of this operator should show as **Available**.



Apply NVIDIA Cluster Config

This section explains how to apply the NVIDIA cluster config. Please read the NVIDIA documentation on customizing this if you have your own private repos or specific settings. This process may take several minutes to complete.

1. Apply cluster config.

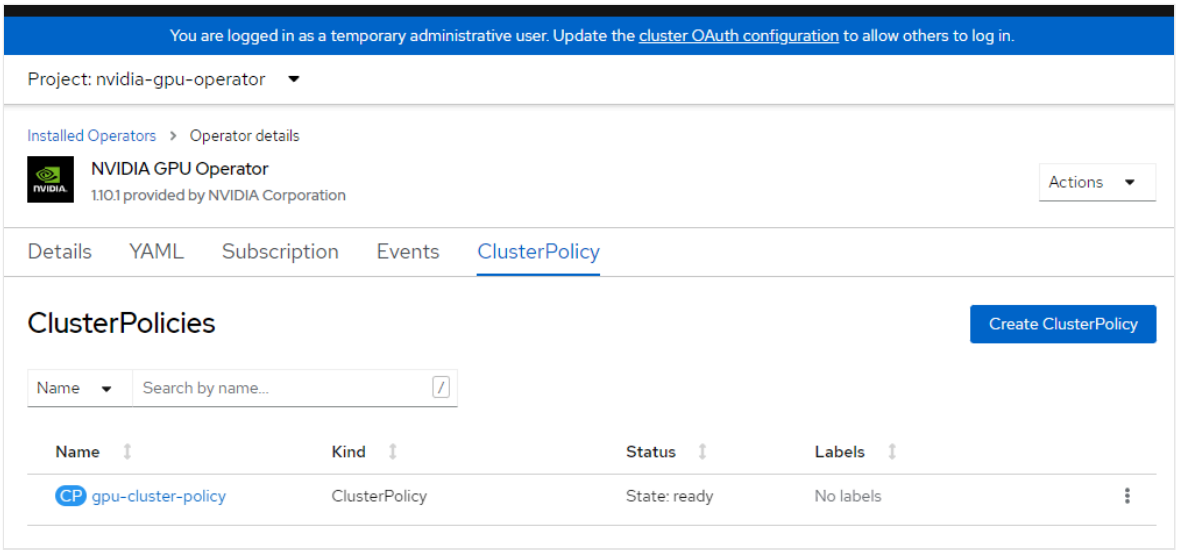
YAML

```
cat <<EOF | oc apply -f -
apiVersion: nvidia.com/v1
kind: ClusterPolicy
metadata:
  name: gpu-cluster-policy
spec:
  migManager:
    enabled: true
  operator:
    defaultRuntime: crio
    initContainer: {}
    runtimeClass: nvidia
    deployGFD: true
  dcgm:
    enabled: true
  gfd: {}
  dcgmExporter:
    config:
      name: ''
  driver:
    licensingConfig:
      nlsEnabled: false
      configMapName: ''
    certConfig:
      name: ''
    kernelModuleConfig:
```

```
      name: ''
    repoConfig:
      configMapName: ''
    virtualTopology:
      config: ''
    enabled: true
    use_ocp_driver_toolkit: true
  devicePlugin: {}
  mig:
    strategy: single
  validator:
    plugin:
      env:
        - name: WITH_WORKLOAD
          value: 'true'
  nodeStatusExporter:
    enabled: true
  daemonsets: {}
  toolkit:
    enabled: true
EOF
```

2. Verify cluster policy.

   Log in to OpenShift console and browse to operators. Ensure sure you're in the `nvidia-gpu-operator` namespace. It should say `State: Ready once everything is complete`.



Validate GPU

It may take some time for the NVIDIA Operator and NFD to completely install and self-identify the machines. Run the following commands to validate that everything is running as expected:

1. Verify that NFD can see your GPU(s).

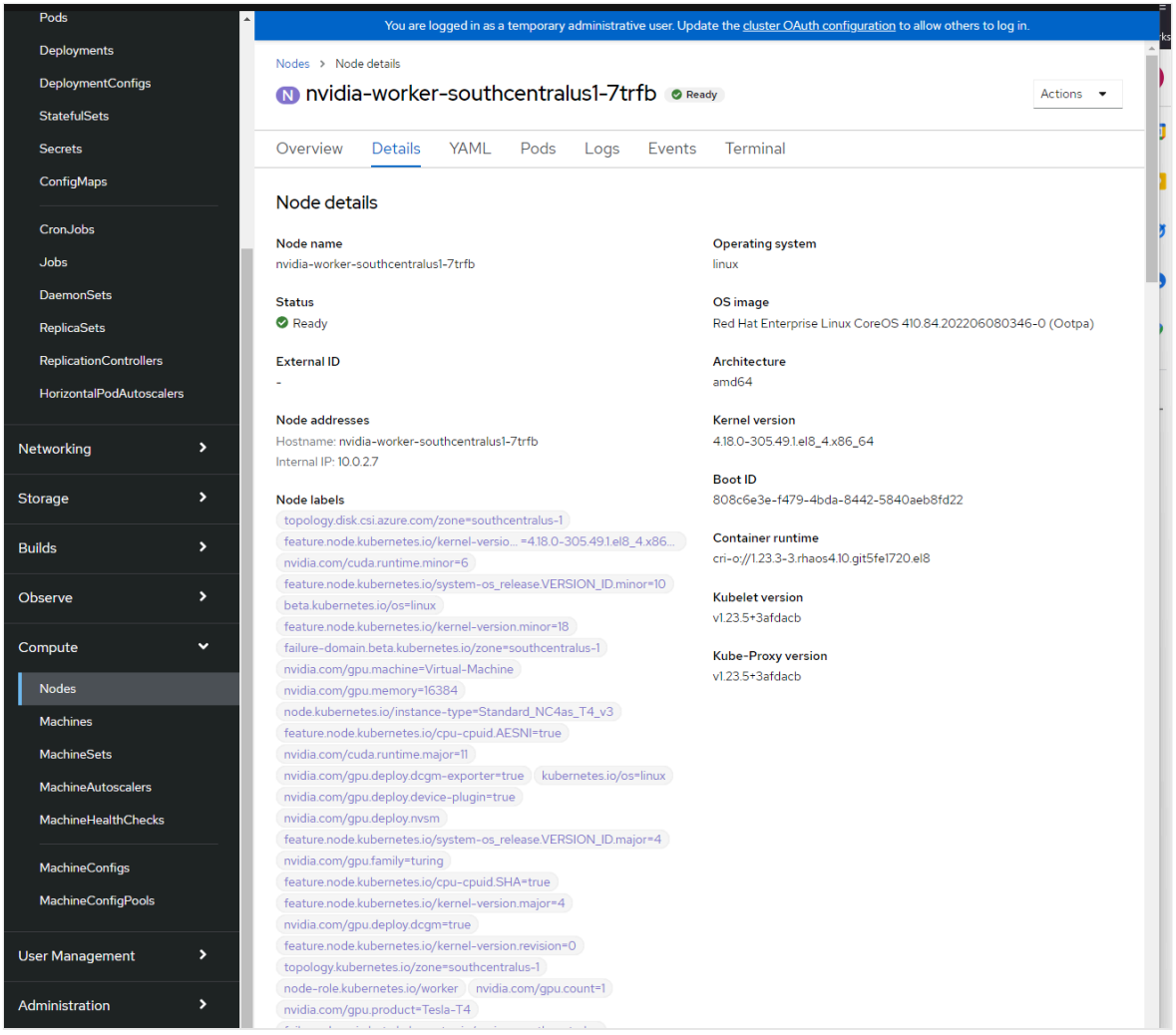
```
Bash

oc describe node | egrep 'Roles|pci-10de' | grep -v master
```

The output should appear similar to the following:

```
Bash

Roles:              worker
                    feature.node.kubernetes.io/pci-10de.present=true
```

2. Verify node labels.

You can see the node labels by logging into the OpenShift console -> Compute -> Nodes -> nvidia-worker-southcentralus1-. You should see multiple NVIDIA GPU labels and the pci-10de device from above.

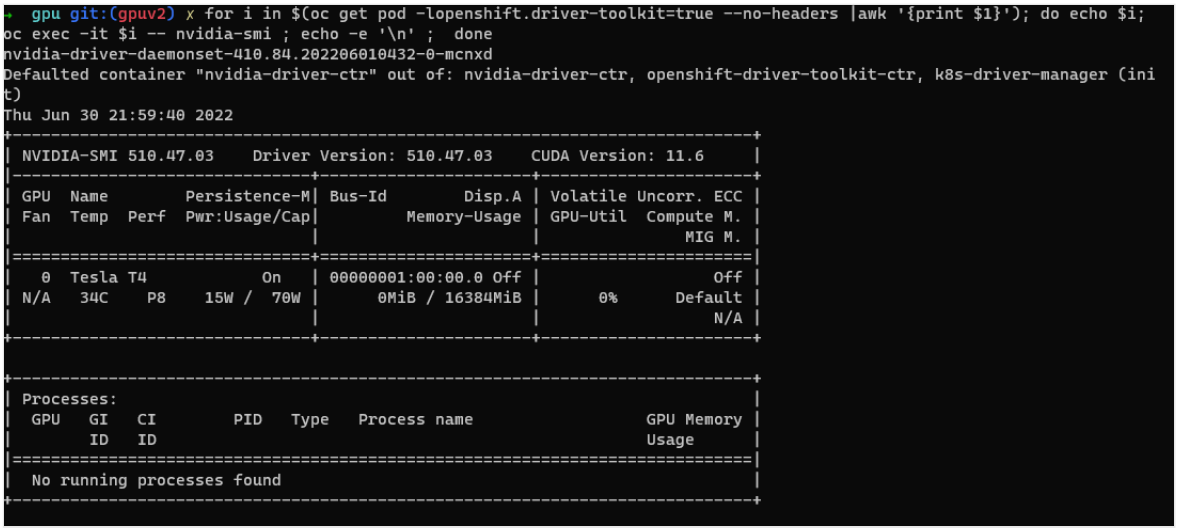


3. NVIDIA SMI tool verification.

```
Bash
```

```
oc project nvidia-gpu-operator
for i in $(oc get pod -lopenshift.driver-toolkit=true --no-headers |awk
'{print $1}'); do echo $i; oc exec -it $i -- nvidia-smi ; echo -e '\n'
;  done
```

You should see output that shows the GPUs available on the host such as this example screenshot. (Varies depending on GPU worker type)



4. Create Pod to run a GPU workload

YAML

```
oc project nvidia-gpu-operator
cat <<EOF | oc apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: cuda-vector-add
spec:
  restartPolicy: OnFailure
  containers:
    - name: cuda-vector-add
      image: "quay.io/giantswarm/nvidia-gpu-demo:latest"
      resources:
        limits:
          nvidia.com/gpu: 1
      nodeSelector:
        nvidia.com/gpu.present: true
EOF
```

5. View logs.

Bash

```
oc logs cuda-vector-add --tail=-1
```

> **Note**
>
> If you get an error `Error from server (BadRequest): container "cuda-vector-add" in pod "cuda-vector-add" is waiting to start: ContainerCreating`, try running `oc delete pod cuda-vector-add` and then re-run the create statement above.

The output should be similar to the following (depending on GPU):

Bash

```
[Vector addition of 5000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

If successful, the pod can be deleted:

Bash

```
oc delete pod cuda-vector-add
```

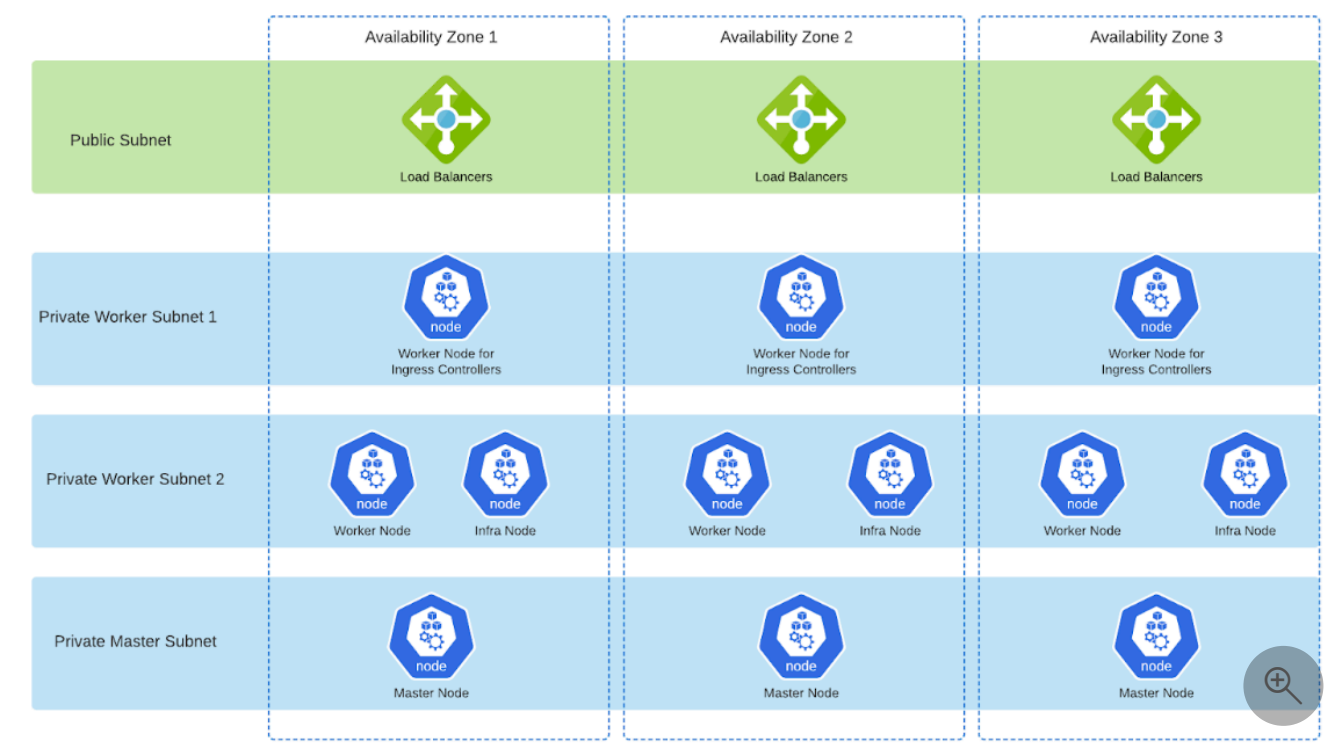
Feedback

Segregate worker nodes into different subnets with Azure Red Hat OpenShift

07/18/2025

This article shows you how to segregate worker nodes into different private subnets as part of a deployment. Separating worker nodes into different private subnets allows you to meet specific access control requirements for various services and applications deployed on Azure Red Hat OpenShif.

For example, you might want to run specific ingress controllers on dedicated worker nodes within a specific subnet, while the rest of the Kubernetes nodes for workloads (infra and other workers) are within a different subnet, as shown below:



> ⓘ **Note**
>
> As part of Azure Red Hat OpenShif, master and worker nodes cannot be deployed in the same private subnet.

In order to segregate worker nodes into different subnets, two main steps need to be performed:

1. Deploy a cluster.

2. Create the appropriate subnets and machine sets associated with those subnets.

Deploy a cluster

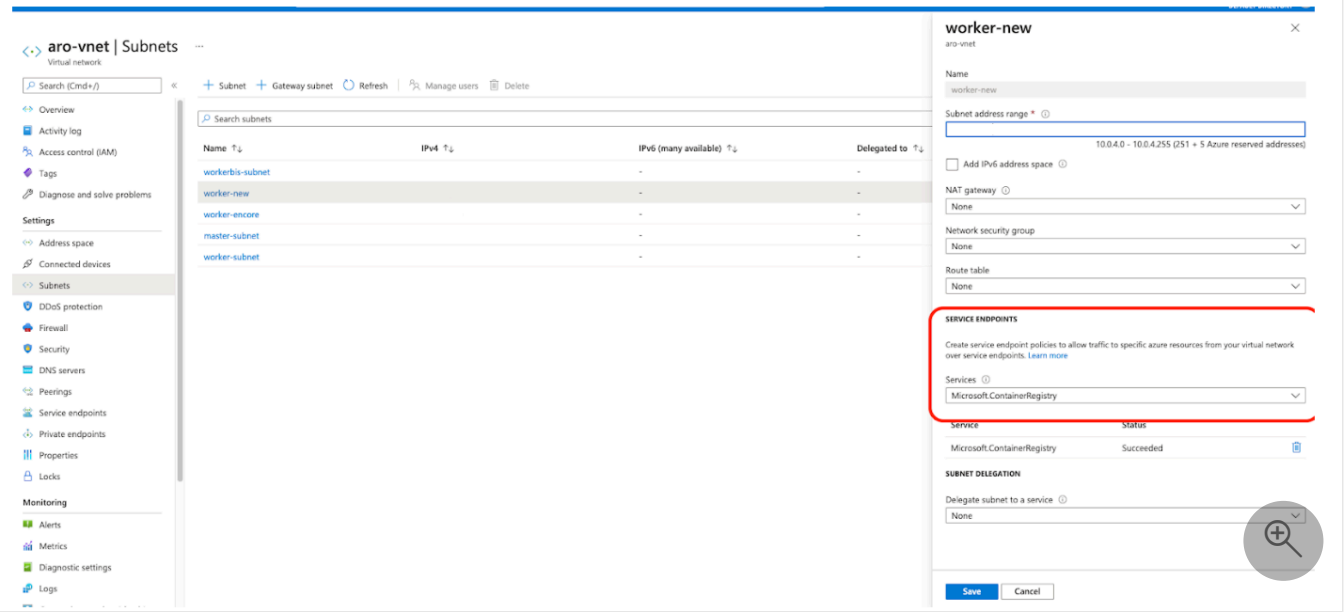
See Create an Azure Red Hat OpenShift 4 cluster for instructions on performing this step.

Create the subnets and associated machine sets

Once you've deployed your cluster, you'll need to create extra subnets as part of the same overall virtual network and create new machine sets for those subnets.

Step 1: Create the subnets

Create the subnets as part of the current virtual network in which is deployed. Make sure that all the subnets are updated to the `Microsoft.ContainerRegistry` for **Service Endpoints**.



Step 2: Sign-in to the jumphost

> **Note**
>
> This step is optional if you have an alternate method for logging into the cluster.

Use the following command to log into the jumphost:

```
oc login $apiServer -u kubeadmin -p <kubeadmin password>
```

Verify the number of nodes and machine sets using the `oc get nodes` and `oc get machineSets -n openshift-machine-api` commands, as shown in the following examples:

```
```

```
$ oc get nodes
NAME                                        STATUS   ROLES    AGE   VERSION
simon-aro-st5rm-master-0                    Ready    master   66m
v1.19.0+e405995
simon-aro-st5rm-master-1                    Ready    master   67m
v1.19.0+e405995
simon-aro-st5rm-master-2                    Ready    master   67m
v1.19.0+e405995
simon-aro-st5rm-worker-useast1-h6kzn   Ready    worker   59m   v1.19.0+e405995
simon-aro-st5rm-worker-useast2-48zsm   Ready    worker   59m   v1.19.0+e405995
simon-aro-st5rm-worker-useast3-rvzpn   Ready    worker   59m   v1.19.0+e405995
```

```
# oc get machineSets --all-namespaces
NAMESPACE               NAME                                DESIRED   CURRENT
READY   AVAILABLE   AGE
openshift-machine-api   simon-aro-st5rm-worker-useast1   1         1         1
1           69m
openshift-machine-api   simon-aro-st5rm-worker-useast2   1         1         1
1           69m
openshift-machine-api   simon-aro-st5rm-worker-useast3   1         1         1
1           69m
```

Step 3: Retrieve the machine sets in the `openshift-machine-api project/namespace`

Retrieving the machine sets allows you to get all of the relevant parameters into the machineSet template used in the following step.

`oc describe machineSet simon-aro-st5rm-worker-useast1 > aro-worker-az1.yaml`

Step 4: Create a new machineSet YAML file and apply it to the cluster

Use the template below for your machineSet YAML file. Change the parameters shown with **Xs** according to the values retrieved in the previous section. For example, `machine.openshift.io/cluster-api-cluster: XXX-XXX-XXX` might be `machine.openshift.io/cluster-api-cluster: machine-aro-st3mr`

yml

```yaml
==============MachineSet Template====================
apiVersion: machine.openshift.io/v1beta1
```

```
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
    machine.openshift.io/cluster-api-machine-role: worker
    machine.openshift.io/cluster-api-machine-type: worker
  name: XXX-XXX-XXX-XXX-XXX
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
      machine.openshift.io/cluster-api-machineset: <INFRASTRUCTURE_ID>-infra-
<REGION><ZONE>
  template:
    metadata:
      creationTimestamp: null
      labels:
        machine.openshift.io/cluster-api-cluster: <INFRASTRUCTURE_ID>
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: <INFRASTRUCTURE_ID>-infra-
<REGION><ZONE>
    spec:
      metadata:
        creationTimestamp: null
        labels:
          node-role.kubernetes.io/<role>: ""            #Example: worker,infra
      providerSpec:
        value:
          apiVersion: azureproviderconfig.openshift.io/v1beta1
          credentialsSecret:
            name: azure-cloud-credentials
            namespace: openshift-machine-api
          image:
            offer: aro4
            publisher: azureopenshift
            resourceID: ""
            sku: <SKU>
            version: <VERSION>
          kind: AzureMachineProviderSpec
          location: <REGION>
          metadata:
            creationTimestamp: null
          natRule: null
          networkResourceGroup: <NETWORK_RESOURCE_GROUP>
          osDisk:
            diskSizeGB: 128
            managedDisk:
              storageAccountType: Premium_LRS
            osType: Linux
          publicIP: false
          publicLoadBalancer: <LOADBALANCER_NAME>
          resourceGroup: <CLUSTER_RESOURCE_GROUP>
```

```
            subnet: <SUBNET_NAME>
            userDataSecret:
              name: worker-user-data
            vmSize: Standard_D4s_v3
            vnet: <VNET_NAME>
            zone: <ZONE>
```

Step 5: Apply the machine set

Apply the machine set created in the previous section using the `oc apply -f <filename.yaml>` command, as in the following example:

```
[root@jumphost-new ARO-cluster-Private]# oc apply -f aro-new-worker-az1.yaml
machineset.machine.openshift.io/simon-aro-qpsl5-worker-useast4 created
```

Step 6: Verify the machine set and nodes

Once you've applied the YAML file, you can verify the creation of the machine set and nodes using the `oc get machineSets` and `oc get nodes` commands, as shown in the following examples:

`[root@jumphost-new ARO-cluster-Private]# oc get machineSet`

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE
AGE
simon-aro-st5rm-worker-useast1   1         1         1       1           142m
simon-aro-st5rm-worker-useast2   1         1         1       1           142m
simon-aro-st5rm-worker-useast3   1         1         1       1           142m
simon-aro-st5rm-worker-useast4   1         1                             46s
```

After a few more minutes, the new machine set and nodes will appear:

`[root@jumphost-new ARO-cluster-Private]# oc get machineSet`

```
NAME                                DESIRED   CURRENT   READY   AVAILABLE
AGE
simon-aro-st5rm-worker-useast1   1         1         1       1           148m
simon-aro-st5rm-worker-useast2   1         1         1       1           148m
```

```
simon-aro-st5rm-worker-useast3   1         1         1       1           148m
simon-aro-st5rm-worker-useast4   1         1         1       1           6m11s
```

`[root@jumphost-new ARO-cluster-Private]# oc get nodes`

```
NAME                                        STATUS   ROLES    AGE    VERSION
simon-aro-st5rm-master-0                    Ready    master   147m
v1.19.0+e405995
simon-aro-st5rm-master-1                    Ready    master   147m
v1.19.0+e405995
simon-aro-st5rm-master-2                    Ready    master   147m
v1.19.0+e405995
simon-aro-st5rm-worker-useast1-h6kzn   Ready    worker   139m   v1.19.0+e405995
simon-aro-st5rm-worker-useast2-48zsm   Ready    worker   139m   v1.19.0+e405995
simon-aro-st5rm-worker-useast3-rvzpn   Ready    worker   139m   v1.19.0+e405995
simon-aro-st5rm-worker-useast4-qrsgx   Ready    worker   104s   v1.19.0+e405995
```

Tag Azure Red Hat OpenShift resources using Azure Policy

08/26/2025

This article shows you how to use Azure Policy to apply tags to the resources in an Azure Red Hat OpenShift cluster's managed resource group. You create three JSON files to create a policy definition and policy assignment at the subscription scope. You deploy a cluster that uses the policy's tags and use a policy remediation task to apply tags to cluster resources.

Prerequisites

- If you don't have an Azure account, create a free account before you begin.
- A text editor like Microsoft Visual Studio Code.
- Azure CLI installed on your computer to run the commands in this article. If you need to install or upgrade, see Install Azure CLI. You can also use Azure Cloud Shell in a browser.
- Review the prerequisites, permissions, and resource providers requirements in the Quickstart: Create an Azure Red Hat OpenShift 4 cluster because you create a cluster later in this article.

Create JSON files

Use a text editor like Microsoft Visual Studio Code, to copy and paste the following code and create three JSON files in your current working directory. These files are used to create your policy definition and policy assignment.

Create rules file

Create the *rules.json* file.

JSON

```
{
  "if": {
    "anyOf": [
      {
        "allOf": [
          {
            "value": "[resourceGroup().name]",
            "equals": "[parameters('resourceGroupName')]"
          }
        ]
      },
```

```
      {
        "allOf": [
          {
            "field": "name",
            "equals": "[parameters('resourceGroupName')]"
          },
          {
            "field": "type",
            "equals": "Microsoft.Resources/subscriptions/resourceGroups"
          }
        ]
      }
    ]
  },
  "then": {
    "details": {
      "operations": [
        {
          "condition": "[not(equals(parameters('tag0')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag0')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag0')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag1')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag1')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag1')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag2')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag2')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag2')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag3')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag3')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag3')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag4')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag4')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag4')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag5')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag5')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag5')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag6')['tag'][0], ''))]",
```

```
          "field": "[concat('tags[', parameters('tag6')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag6')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag7')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag7')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag7')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag8')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag8')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag8')['tag'][1]]"
        },
        {
          "condition": "[not(equals(parameters('tag9')['tag'][0], ''))]",
          "field": "[concat('tags[', parameters('tag9')['tag'][0], ']')]",
          "operation": "addOrReplace",
          "value": "[parameters('tag9')['tag'][1]]"
        }
      ],
      "roleDefinitionIds": [
        "/providers/Microsoft.Authorization/roleDefinitions/4a9ae827-6dc8-4573-
8ac7-8239d42aa03f"
      ]
    },
    "effect": "modify"
  }
}
```

Create parameter definitions

Create the *param-defs.json* file.

JSON

```
{
  "tag0": {
    "type": "Object",
    "metadata": {
      "displayName": "tag0"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
```

```
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag1": {
    "type": "Object",
    "metadata": {
      "displayName": "tag1"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag2": {
    "type": "Object",
    "metadata": {
      "displayName": "tag2"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
```

```
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag3": {
    "type": "Object",
    "metadata": {
      "displayName": "tag3"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag4": {
    "type": "Object",
    "metadata": {
      "displayName": "tag4"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
```

```
    }
  },
  "tag5": {
    "type": "Object",
    "metadata": {
      "displayName": "tag5"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag6": {
    "type": "Object",
    "metadata": {
      "displayName": "tag6"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag7": {
    "type": "Object",
    "metadata": {
```

```
      "displayName": "tag7"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag8": {
    "type": "Object",
    "metadata": {
      "displayName": "tag8"
    },
    "defaultValue": {
      "tag": [
        "",
        ""
      ]
    },
    "schema": {
      "type": "object",
      "properties": {
        "tag": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "maxItems": 2,
          "minItems": 2
        }
      }
    }
  },
  "tag9": {
    "type": "Object",
    "metadata": {
      "displayName": "tag9"
    },
    "defaultValue": {
      "tag": [
        "",
```

```
          ""
        ]
      },
      "schema": {
        "type": "object",
        "properties": {
          "tag": {
            "type": "array",
            "items": {
              "type": "string"
            },
            "maxItems": 2,
            "minItems": 2
          }
        }
      }
    },
    "resourceGroupName": {
      "type": "String",
      "metadata": {
        "displayName": "Resource Group Name",
        "description": "The name of the resource group whose resources you'd like to
require the tag on"
      }
    }
}
```

Create the *param-values.json* file.

JSON

```
{
  "tag0": {
    "value": {
      "tag": [
        "<your tag key>",
        "<your tag value>"
      ]
    }
  },
  "resourceGroupName": {
    "value": "<your cluster's managed resource group name>"
  }
}
```

This *param-values.json* file is the only one of the three files that must be modified. In the content, specify the values for the tags you want applied to the cluster's resources. For this example, replace the following values:

- `<your tag key>`: `demoKey`
- `<your tag value>`: `demoResourceGroupTag`

- `<your cluster's managed resource group name>`: `demoMRG`

The *param-values.json* file content only shows a value for the `tag0` parameter. The policy allows up to 10 tags. To set more than one tag, add values for parameters `tag1` through `tag 9`. Applying more than 10 tags to resources in the managed resource group isn't supported.

Set environment variables

Open a Bash shell, sign in to Azure, and create variables.

If you're using Azure CLI on your computer, from your Bash session, sign in to your Azure subscription. For more information, see Authenticate to Azure using Azure CLI.

Azure CLI

```
az login
```

Create environment variables used in later steps. You can use the values provided or create your own values.

Bash

```
export AZURE_SUBSCRIPTION_ID=$(az account list --query [].id --output tsv)
export POLICY_DEFINITION=demoPolicyDefName
export CLUSTER=demoCluster
export MANAGED_RESOURCE_GROUP=demoMRG
export POLICY_ASSIGNMENT="${POLICY_DEFINITION}-${CLUSTER}"
export LOCATION=centralus
```

Expand table

Variable name	Description
`AZURE_SUBSCRIPTION_ID`	The Azure subscription where resources are created. The command gets the ID from the Azure subscription you're signed into.
`POLICY_DEFINITION`	The name of your policy definition to apply resource tags. For this example, *demoPolicyDefName*.
`CLUSTER`	Your Azure Red Hat OpenShift cluster's name. For this example, *demoCluster*.
`MANAGED_RESOURCE_GROUP`	The managed resource group that contains your cluster's resources. For this example, *demoMRG*.
`POLICY_ASSIGNMENT`	Name created by joining the POLICY_DEFINITION and CLUSTER variable names.

Variable name	Description
`LOCATION`	Region where the policy assignment's managed identity is created. For this example, *centralus*. For more information about the managed identity, see Configure the managed identity.

Create the policy definition and assignment

1. To create the policy definition, run the following command.

   Azure CLI

   ```
   az policy definition create \
     --name $POLICY_DEFINITION \
     --mode All \
     --rules rules.json \
     --params param-defs.json
   ```

2. To create the policy assignment, run the following command.

   Azure CLI

   ```
    az policy assignment create \
      --name $POLICY_ASSIGNMENT \
      --policy $POLICY_DEFINITION \
      --scope "/subscriptions/${AZURE_SUBSCRIPTION_ID}" \
      --location $LOCATION \
      --mi-system-assigned \
      --role "Tag Contributor" \
      --identity-scope "/subscriptions/${AZURE_SUBSCRIPTION_ID}" \
      --params param-values.json
   ```

Create the cluster

Follow the instructions in create a cluster. In your deployment command, `az aro create`, include the parameter `--cluster-resource-group $MANAGED_RESOURCE_GROUP`. The command should look like the following example.

Azure CLI

```
az aro create \
  --resource-group $RESOURCEGROUP \
  --name $CLUSTER \
  --vnet $VIRTUALNETWORK \
  --master-subnet master-subnet \
  --worker-subnet worker-subnet \
```

```
    --version <x.y.z> \
    --cluster-resource-group $MANAGED_RESOURCE_GROUP
```

The policy doesn't apply any tags to resources in the cluster resource group. Tags are only applied to the resources in the managed resource group, in this example, *demoMRG*.

Remediate tags using Azure Policy

You can remediate previously assigned tags and add new tags using an Azure Policy remediation task. The *demoMRG* managed resource group and its resources have the tag `demoKey: demoResourceTag`.

Open the file `param-values.json` and modify existing parameters values and add new parameter values in the following example.

JSON

```
{
  "tag0": {
    "value": {
      "tag": [
        "demoKeyUpdate",
        "demoResourceGroupTagUpdate"
      ]
    }
  },
  "tag1": {
    "value": {
      "tag": [
        "demoKeyTag1",
        "demoResourceGroupTag1"
      ]
    }
  },
  "resourceGroupName": {
    "value": "demoMRG"
  }
}
```

When you run the commands to update the parameter values and run the remediation task, the policy applies the tags to resources in the managed resource group.

1. Run the following command to update the policy assignment's parameter values.

   Azure CLI

```
az policy assignment update \
  --name $POLICY_ASSIGNMENT \
  --params param-values.json
```

2. Trigger the remediation task.

Azure CLI

```
az policy remediation create \
  --name demoRemediation \
  --resource-group $MANAGED_RESOURCE_GROUP \
  --policy-assignment $POLICY_ASSIGNMENT
```

Allow the remediation task time to run and observe the tags being updated on the managed resource group and its resources. A remediation task doesn't remove tags from resources it only adds or updates tags.

Next steps

- Quickstart: Connect to an Azure Red Hat OpenShift 4 cluster
- Quickstart: Delete an Azure Red Hat OpenShift 4 cluster

Update Azure Red Hat OpenShift cluster certificates

09/09/2025

Azure Red Hat OpenShift uses cluster certificates stored on worker machines for API and application ingress. These certificates are normally updated in a transparent process during routine maintenance. In some cases, cluster certificates might fail to update during maintenance.

If you're experiencing certificate issues, you can manually update your certificates using the az aro update command:

Azure CLI

```
az aro update --name MyCluster --resource-group MyResourceGroup
```

where:

- `name` is the name of the cluster
- `resource-group` is the name of the resource group. You can configure the default group using `az-config --defaults group=<name>`.

Running this command restarts worker machines and updates the cluster certificates, setting the cluster to a known, proper state.

> **Note**
>
> Certificates for custom domains need to be updated manually. For more information, see the Red Hat OpenShift documentation.

Use Admin Kubeconfig to access an Azure Red Hat OpenShift cluster

07/18/2025

This article shows you how to regain access to a cluster using the Admin Kubeconfig feature. The Admin Kubeconfig feature lets you download and log in with the Admin Kubconfig file using the OpenShift CLI rather than the console, thus bypassing components that may not be functioning properly. This can be helpful in the following instances:

- The Azure Red Hat OpenShift console isn't responding, or not allowing a login.
- The OpenShift CLI isn't responding to requests.
- Cluster operators may not be available or accessible.
- An alternate cluster login method is required in order to fix the above issues.

The Admin Kubeconfig feature allows cluster access in scenarios where the kube-apiserver is available, but `openshift-ingress`, `openshift-console`, or `openshift-authentication` aren't allowing login.

> **Note**
>
> When using the Admin Kubeconfig feature in an environment with multiple clusters, make sure you are working in the correct context. For more information about contexts, see the Red Hat OpenShift documentation.

Before you begin

Ensure you're running Azure CLI version 2.50.0 or later.

To check the version of Azure CLI run:

Azure CLI

```
# Azure CLI version
az --version
```

To install or upgrade Azure CLI, see Install Azure CLI.

Retrieve Admin Kubeconfig

Run the following to retrieve Admin Kubeconfig:

```
export SUBSCRIPTION_ID=<your-subscription-ID>
export RESOURCE_GROUP=<your-resource-group-name>
export CLUSTER=<name-of-cluster>

az aro get-admin-kubeconfig --subscription $SUBSCRIPTION_ID --resource-group
$RESOURCE_GROUP --name $CLUSTER
```

Source and use the Kubeconfig

By default, the command used previously to retrieve Admin Kubeconfig saves it to the local directory under the name *kubeconfig*. To use it, set the environment variable `KUBECONFIG` to the path of that file:

```
export KUBECONFIG=/path/to/kubeconfig
oc get nodes
[output will show up here]
```

Now there's no need to use the OpenShift CLI (`oc`) login because the admin user is already logged in and the kubeconfig file is present.

Understand managed identities in Azure Red Hat OpenShift

In this article, learn about managed identities and how to use them to control or enable access to Azure resources.

What are managed identities

Managed identities are service principals of a special type, which can be used to control or enable access to Azure resources. They help reduce the risk of security breaches by eliminating the need to manually manage credentials, certificates, keys, or secrets. An application (or a service) utilizes Microsoft Entra ID authentication to request temporary, limited permission credentials in order to access other Azure services. To learn more about managed identities in Azure, see What are managed identities for Azure resources.

In Azure Red Hat OpenShift, the term workload identity is used to represent an application or workload that is running in the cluster that requires access to Azure services. For more information, see the documentation for What are workload identities?.

Managed identities and workload identities help minimize risk when securing workloads and applications by providing short-lived tokens rather than long-lived credentials such as a service principal with client secret credentials. This method provides a more secure authentication method for applications should credentials become compromised because of following factors:

- Short-term duration of the credentials.
- Refined minimal set of permissions, the principle of least privilege. The permissions are limited to only what the workload needs and only to the resources it needs to perform the task.

Understand identity role assignment architecture

Azure Red Hat OpenShift consists of core operators. Core operators are automated controllers responsible for managing an Azure Red Hat OpenShift cluster core infrastructure and ensuring its stability and operational health. An operator handles tasks like: deploy and maintain control plane components, manage cluster updates, oversee networking, and ensure essential cluster-level services are running correctly.

In an Azure Red Hat OpenShift cluster with managed identity support, core operators are assigned permissions from corresponding user-assigned managed identities. A user-assigned managed identity needs to be associated with each of the seven core Red Hat OpenShift

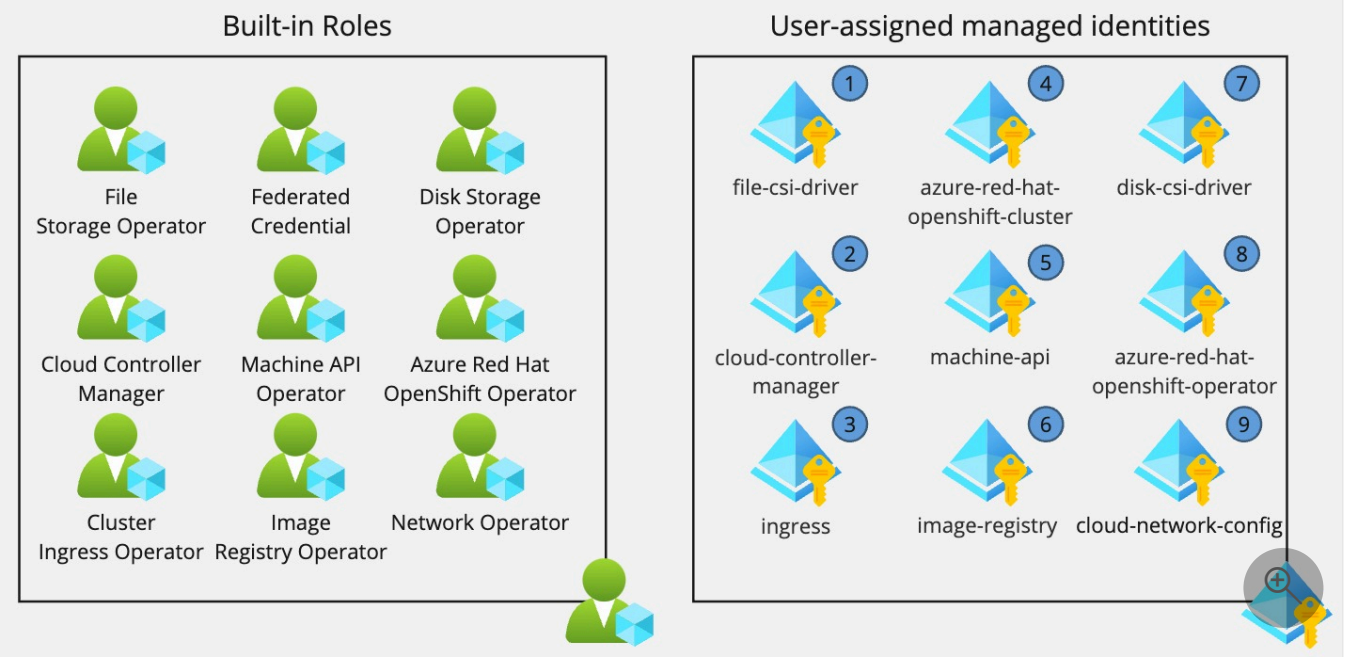
operators, and the Azure Red Hat OpenShift service operator. As of OpenShift 4.19, the Azure Red Hat OpenShift operators are:

- OpenShift Image Registry Operator (image-registry)
- OpenShift Network Operator (cloud-network-config)
- OpenShift Disk Storage Operator (disk-csi-driver)
- OpenShift File Storage Operator (file-csi-driver)
- OpenShift Cluster Ingress Operator (ingress)
- OpenShift Cloud Controller Manager (cloud-controller-manager)
- OpenShift Machine API Operator (machine-api)
- Azure Red Hat OpenShift Service Operator (aro-operator)

The Azure Red Hat OpenShift cluster requires one more user-assigned managed identity. The additional user-assigned managed identity is used to enable the use of the eight core Azure Red Hat OpenShift operator managed identities and perform federated credential creation for all the managed identities. The additional managed identity is the Azure Red Hat OpenShift Cluster Identity (aro-cluster). For more information about Red Hat OpenShift cluster operators, see Cluster Operators reference .

The following image shows the relation between managed identities and role assignment. Numbers are assigned to the managed identities to demonstrate role assignment in image three.

Image 1 - Roles and user-assigned identities (all diagrams are illustrative only)



Built-in Azure roles for Azure Red Hat OpenShift

Specific Azure Red Hat OpenShift built-in Azure roles grant the required permissions to the supporting managed identities to enable cluster operations. These roles follow Azure's model

by offering a standardized permission set for a common job function. The supporting built-in Azure roles are:

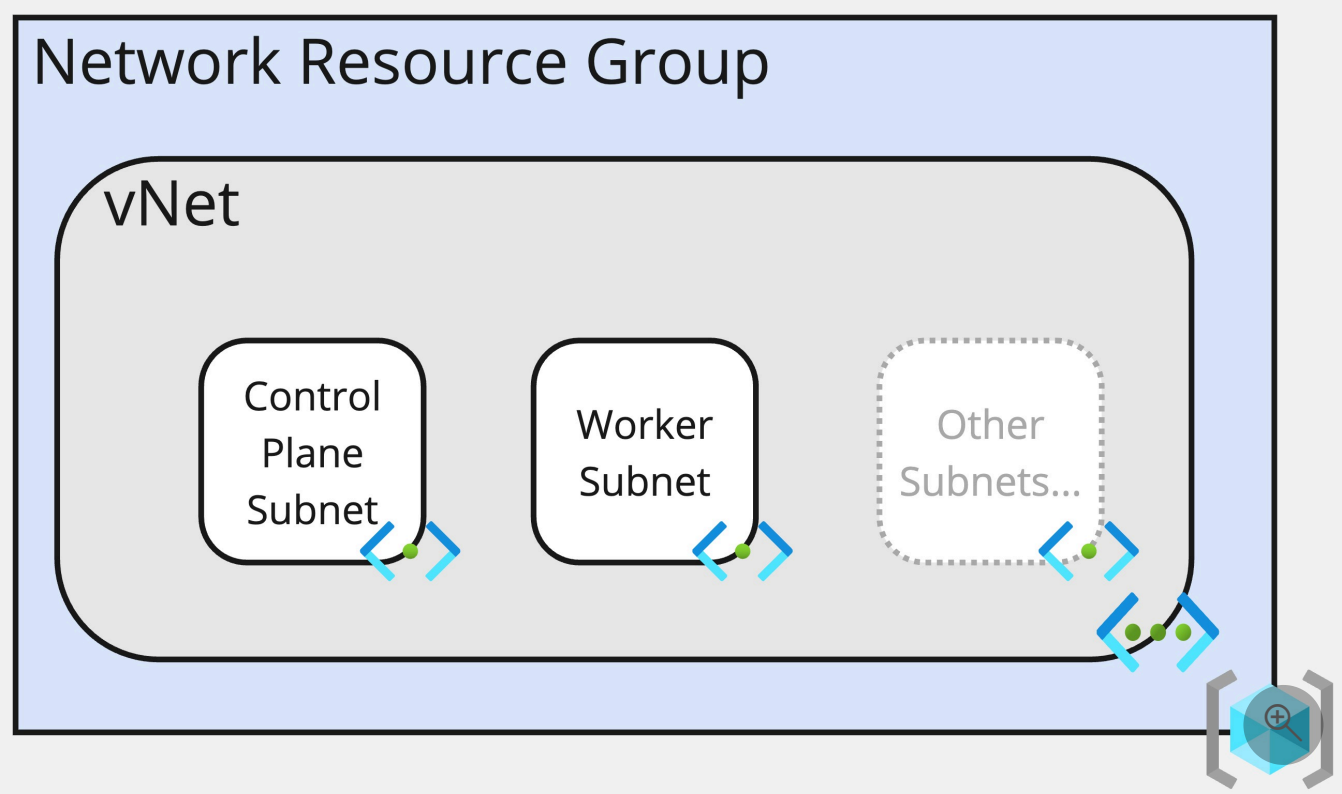
- Azure Red Hat OpenShift Cloud Controller Manager
- Azure Red Hat OpenShift Cluster Ingress Operator
- Azure Red Hat OpenShift Disk Storage Operator
- Azure Red Hat OpenShift Federated Credential
- Azure Red Hat OpenShift File Storage Operator
- Azure Red Hat OpenShift Image Registry Operator
- Azure Red Hat OpenShift Machine API Operator
- Azure Red Hat OpenShift Network Operator
- Azure Red Hat OpenShift Service Operator

For a detailed description of the built-in Azure Red Hat OpenShift roles see, Azure built-in roles.

Role assignment scope

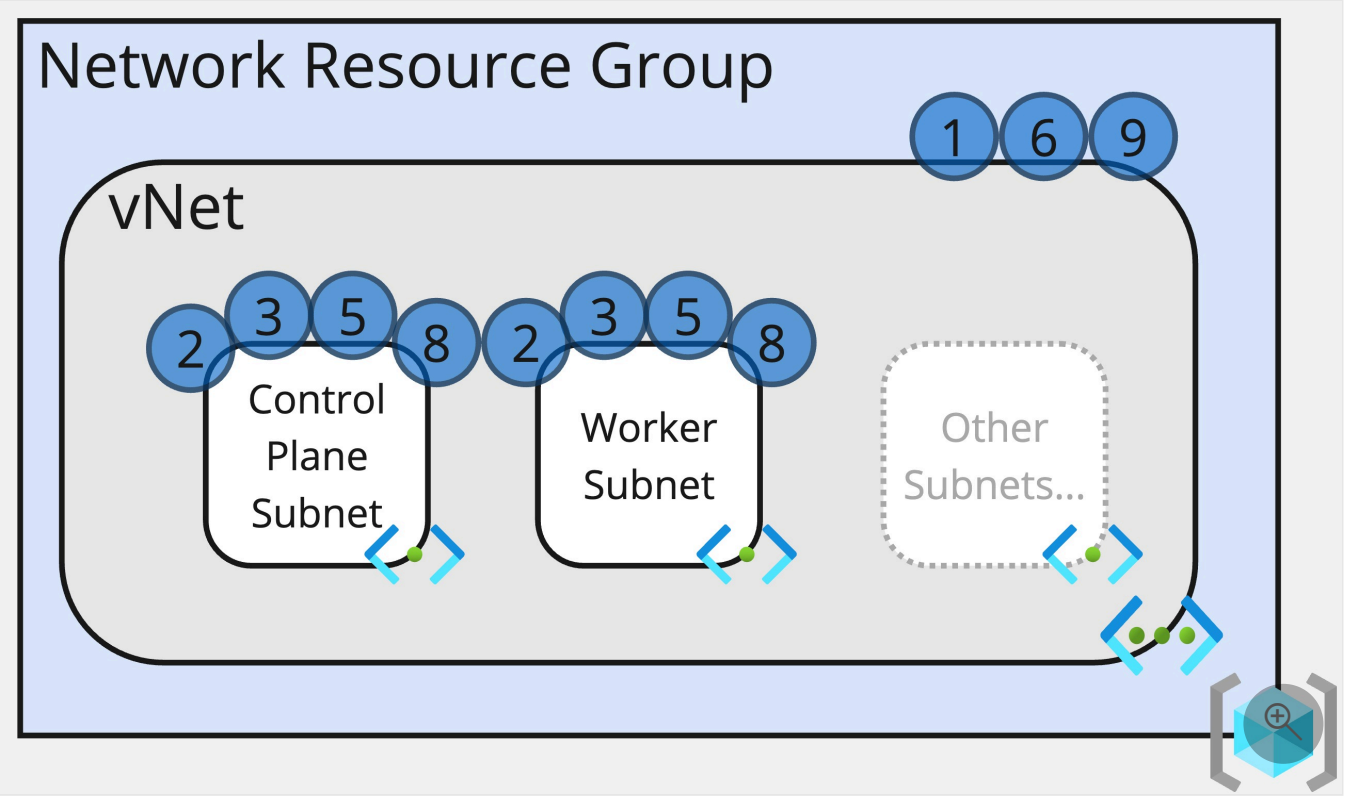
An Azure Red Hat OpenShift deployment requires that a virtual network with at least two empty subnets (one for control plane nodes and the other for worker nodes) exist in a resource group. For the following **Image 2** example, this resource group is called the **Network Resource Group**.

Image 2 - Network resource group



An Azure Red Hat OpenShift cluster with managed identities requires the creation of user-assigned managed identities with their corresponding role assignments. Follow the steps to Create an Azure Red Hat OpenShift cluster with managed identities enabled. After you created an Azure Red Hat OpenShift cluster, the role assignments in the Network Resource Group will look like the following example in Image 3.

Image 3 - Role assignments on network objects



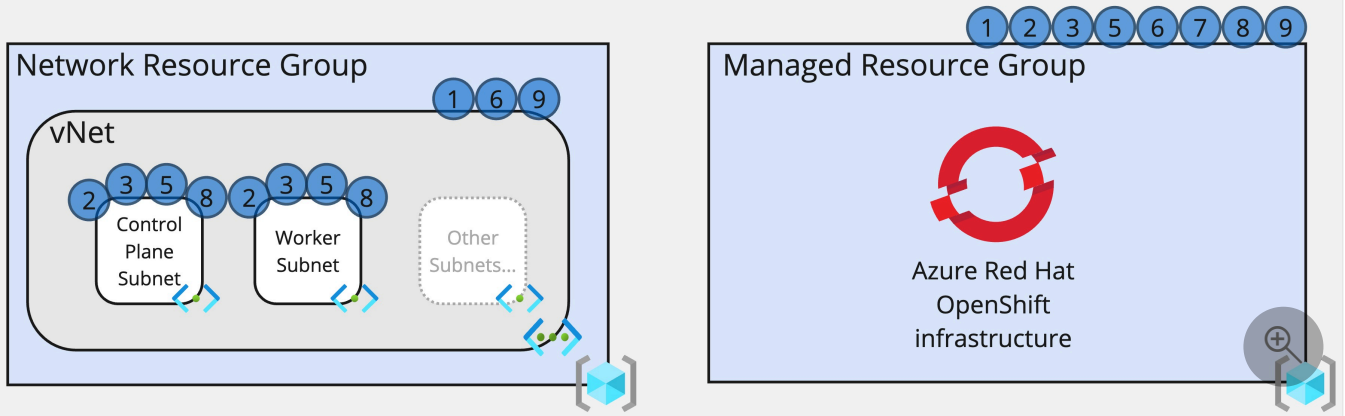
The identity role assignments are performed at the most limited scope possible. Most managed identities are on the specific subnets required, while the Network Operator, Image Registry, and Azure File Storage operator, which require permissions on the virtual network, are performed at the vNet scope.

There is one other assignment that isn't represented in Image 3:

- Cluster Identity - This assignment is set on the other identities themselves in order to be able to create the federated credentials for those components to operate.

The role assignment scopes are represented in the following image .

Image 4 - Role assignments



Thus putting it all together, the below table shows which components need a managed identity and role assignment created and at which scope:

Expand table

Key	Component	Example Identity Name	Built-in Role	Assignment Scope
1	OpenShift File Storage Operator	file-csi-driver	Azure Red Hat OpenShift File Storage Operator	vNET NSG (if using BYO NSG) Route table (if used) NAT GW (if used)
2	OpenShift Cloud Controller Manager	cloud-controller-manager	Azure Red Hat OpenShift Cloud Controller Manager	Control plane subnet Worker subnet NSG (if using BYO NSG) Route table (if used) NAT GW (if used)
3	OpenShift Cluster Ingress Operator	ingress	Azure Red Hat OpenShift Cluster Ingress Operator	Control plane subnet Worker subnet
4	Azure Red Hat OpenShift Cluster	aro-cluster	Azure Red Hat OpenShift Federated Credential	All other cluster identities
5	OpenShift Machine API Operator	machine-api	Azure Red Hat OpenShift Machine API Operator	Control plane subnet Worker subnet NSG (if using BYO NSG) Route table (if used)
6	OpenShift Image Registry Operator	image-registry	Azure Red Hat OpenShift Image Registry Operator	vNET
7	OpenShift Disk Storage Operator	disk-csi-driver	Azure Red Hat OpenShift Disk Storage Operator	None required (in managed resource group only)

Key	Component	Example Identity Name	Built-in Role	Assignment Scope
8	Azure Red Hat OpenShift Service Operator	aro-operator	Azure Red Hat OpenShift Service Operator	Control plane subnet Worker subnet NSG (if using BYO NSG) Route table (if used) NAT GW (if used)
9	OpenShift Network Operator	cloud-network-config	Azure Red Hat OpenShift Network Operator	vNET

Considerations

When you use managed identities and adhere to least privilege principles, it provides many benefits. However, there are some considerations or tradeoffs to keep in mind.

- Since role assignments should happen at the lowest scope possible, there are going to be more role assignments. When performing assignments at the subnet scope, if there are more subnets in use by the cluster like, segregating workloads into different subnets, the role assignments will need to happen on those subnets as well.
- Since Azure has a limit of 4,000 role assignments per subscription, if the limit is a concern, consider performing the role assignments at a higher level (such as the virtual network or resource group). Be sure to consider your security policies and the Azure inheritance hierarchy which *can have you granting permissions to unintended resources*. For example, performing a role assignment at the resource group would grant that identity access to every object in that resource group. For more information, see Understanding scope

> **ⓘ Important**
>
> When you perform role assignments at higher level scopes like, virtual network or resource group, it can have you granting permissions to unintended resources.

- Depending on which scope you decide to perform the role assignments at, as well as the roles required by each OpenShift version, the total number of role assignments will differ. As an example, for OpenShift version 4.19, the following are the total number of role assignments for a basic cluster install, depending on which scope is selected to perform the role assignments:
  - Subnet scope: 28 (11 on network objects + 8 on managed identities + 8 on managed resource group + 1 for the resource provider)

    - Virtual network scope: 24 (7 on network objects + 8 on managed identities + 8 on managed resource group + 1 for the resource provider)
    - Network resource group scope: 17 (8 on network resource group + 8 on managed resource group + 1 for the resource provider) (assuming the managed identities are in the same network resource group)
    - There can be more assignments on optional components/features
        - +4 for each added subnet
        - +4 if using bring your own (BYO) Network Security Group (NSG)
        - +4 if using route table
        - +3 if using NAT gateway

- Since the Azure Files Container Storage Interface (CSI) driver still has a dependency on shared access keys, the default Azure File StorageClass is disabled in clusters with managed / workload identity enabled by default and is optional to turn on for your cluster. If you want to use Azure File in Azure Red Hat OpenShift, create your own StorageClass to use shared keys to access the backing storage. For more information, see How to configure an Azure File StorageClass.

- When you update your cluster, ensure you set the `upgradeable-to` annotation on the cluster's CloudCredential resource. For more information, see Update an Azure Red Hat OpenShift cluster with managed identities enabled.

Next steps

- Learn more about managed identities.
- Create an Azure Red Had OpenShift cluster with managed identities.

Last updated on 02/02/2026

Create an Azure Red Hat OpenShift cluster with managed identities

This article shows you how to deploy an Azure Red Hat OpenShift cluster using managed identities. An existing cluster that uses a service principal can't be migrated to use a managed identity. You need to create a new cluster that uses a managed identity on an Azure Red Hat OpenShift cluster.

Azure Red Hat OpenShift is a managed OpenShift service that supports managed identities and workload identities. Managed identities and workload identities help minimize risk when securing workloads and applications by providing short-lived tokens rather than long-lived credentials such as a service principal with client secret credentials.

For more information, see:

- Understanding managed identities in Azure Red Hat OpenShift
- What are workload identities?
- What are managed identities for Azure resources?

Prerequisites

If you're not using the Azure portal, ensure that you're using Azure CLI version 2.84.0 or higher, which contains the fully supported CLI arguments for managed identity clusters. Use `az --version` to find the version of the Azure CLI you installed. If you need to install or upgrade, see Install Azure CLI.

Azure Red Hat OpenShift requires a minimum of 44 cores to create an OpenShift cluster. The default Azure resource quota for a new Azure subscription doesn't meet this requirement. To request an increase in your resource limit, see Increase VM-family vCPU quotas.

The 44 cores are used as follows:

- Bootstrap machine: 8 cores
- Control plane (master machines): 24 cores
- Compute (worker machines): 12 cores

When the installation is complete, the bootstrap machine is removed and your cluster uses a total of 36 cores. For more information, see Installing on Azure.

Run the following command to check your Azure subscription's quota for the Standard DSv5 virtual machine size that's the default virtual machine size for the CLI or Portal:

Azure CLI

```
LOCATION=eastus
az vm list-usage -l $LOCATION \
--query "[?contains(name.value, 'standardDSv5Family')]" -o table
```

Verify your permissions

In this article, you create a resource group that contains the virtual network and managed identities for the cluster. To create a resource group, you need Contributor and User Access Administrator permissions or Owner permissions on the resource group or subscription containing it.

You also need sufficient Microsoft Entra permissions (either a member user of the tenant, or a guest assigned with the role **Application administrator**) to create a set of managed identities and assign roles for the cluster to use. For more information, see Member and guests and Assign administrator and nonadministrator roles to users with Microsoft Entra ID.

Register the resource providers

Some Azure resource providers, including the Azure Red Hat OpenShift resource provider, require registration in order to function. Registering a resource provider creates a service principal inside of your subscription that authorizes the resource provider to perform certain actions, like resource creation. For instructions on registering resource providers using Azure portal, or for more information around resource provider registration, see Register resource provider.

1. If you have multiple Azure subscriptions, specify the relevant subscription ID:

   Azure CLI

   ```
   az account set --subscription <SUBSCRIPTION ID>
   ```

2. Register the `Microsoft.RedHatOpenShift` resource provider:

   Azure CLI

   ```
   az provider register -n Microsoft.RedHatOpenShift --wait
   ```

3. Register the `Microsoft.Compute` resource provider:

Azure CLI

```azurecli
az provider register -n Microsoft.Compute --wait
```

4. Register the `Microsoft.Storage` resource provider:

Azure CLI

```azurecli
az provider register -n Microsoft.Storage --wait
```

5. Register the `Microsoft.Authorization` resource provider:

Azure CLI

```azurecli
az provider register -n Microsoft.Authorization --wait
```

Get a Red Hat pull secret (optional)

An Azure Red Hat OpenShift pull secret doesn't change the cost of the Red Hat OpenShift license.

A Red Hat pull secret enables your cluster to access Red Hat container registries, along with other content such as operators from OperatorHub. This step is optional but recommended. If you decide to add the pull secret later, follow this guidance. The field `cloud.openshift.com` is removed from your secret even if your pull-secret contains that field. This field enables an extra monitoring feature, which sends data to RedHat and is thus disabled by default. To enable this feature, see Enabling remote health reporting.

1. Navigate to your Red Hat OpenShift cluster manager portal and sign-in.

   You need to sign in to your Red Hat account or create a new Red Hat account with your business email and accept the terms and conditions.

2. Select **Download pull secret**, then download a pull secret to be used with your Azure Red Hat OpenShift cluster.

   Keep the saved `pull-secret.txt` file somewhere safe. The file is used in each cluster creation if you need to create a cluster that includes samples or operators for Red Hat or certified partners.

When running the `az aro create` command, you can reference your pull secret using the `--pull-secret @pull-secret.txt` parameter. Execute `az aro create` from the directory where you stored your `pull-secret.txt` file. Otherwise, replace `@pull-secret.txt` with `@/path/to/my/pull-secret.txt`.

If you're copying your pull secret or referencing it in other scripts, your pull secret should be formatted as a valid JSON string.

Prepare a custom domain for your cluster (optional)

When running the `az aro create` command, you can specify a custom domain for your cluster by using the `--domain foo.example.com` parameter.

> **Note**
>
> Adding a domain name is optional when creating a cluster through Azure CLI. A domain name (or a prefix used as part of the autogenerated DNS name for OpenShift console and API servers) is needed when adding a cluster through the portal. For more information, see Quickstart: Deploy an Azure Red Hat OpenShift cluster using the Azure portal.

If you provide a custom domain for your cluster, note the following points:

- After creating your cluster, you must create two DNS A records in your DNS server for the `--domain` specified:
  - **api** - pointing to the API server IP address
  - ***.apps** - pointing to the ingress IP address
  - Retrieve these values by executing the following command after cluster creation: `az aro show -n -g --query '{api:apiserverProfile.ip, ingress:ingressProfiles[0].ip}'`.
- The OpenShift console is available at a URL such as `https://console-openshift-console.apps.example.com`, instead of the built-in domain `https://console-openshift-console.apps.<random>.<location>.aroapp.io`.
- By default, OpenShift uses self-signed certificates for all of the routes created on custom domains `*.apps.example.com`. If you choose to use custom DNS after connecting to the cluster, you need to follow the OpenShift documentation to configure a custom CA for your ingress controller and a custom CA for your API server.

Installation

You can use Azure CLI, Portal, Bicep, or an Azure Resource Manager template (ARM template) to deploy an Azure Red Hat OpenShift cluster that uses managed identities.

Install using Azure CLI

This section describes how to use Azure CLI to create an Azure Red Hat OpenShift cluster using managed identities.

Create a virtual network containing two empty subnets

Create a virtual network containing two empty subnets. If you have an existing virtual network that meets your needs, skip this step.

For information about networking and requirements, see Networking for Azure Red Hat OpenShift.

1. Set the following variables in the shell environment in which you execute the `az` commands.

   Console

   ```
   LOCATION=eastus                 # the location of your cluster
   RESOURCEGROUP=aro-rg            # the name of the resource group where you
   want to create your cluster
   CLUSTER=cluster                 # the name of your cluster
   ```

2. Create a resource group.

   An Azure resource group is a logical group in which Azure resources are deployed and managed. When you create a resource group, you're asked to specify a location. This location is where resource group metadata is stored and where your resources run in Azure if you don't specify another region during resource creation. Create a resource group using the az group create command.

   > **Note**
   >
   > Azure Red Hat OpenShift isn't available in all regions where an Azure resource group can be created. See **Available regions** for information on where Azure Red Hat OpenShift is supported.

   Azure CLI

```
az group create \
  --location $LOCATION \
  --name $RESOURCEGROUP
```

3. Create a virtual network, master, and worker subnets in the same resource group previously created.

   Azure Red Hat OpenShift clusters require a virtual network with two empty subnets for the master and worker nodes. You can either create a new virtual network or use an existing virtual network.

   Azure CLI

   ```
   az network vnet create \
      --resource-group $RESOURCEGROUP \
      --name aro-vnet \
      --address-prefixes 10.0.0.0/22
   ```

   Azure CLI

   ```
   az network vnet subnet create \
      --resource-group $RESOURCEGROUP \
      --vnet-name aro-vnet \
      --name master \
      --address-prefixes 10.0.0.0/23
   ```

   Azure CLI

   ```
   az network vnet subnet create \
      --resource-group $RESOURCEGROUP \
      --vnet-name aro-vnet \
      --name worker \
      --address-prefixes 10.0.2.0/23
   ```

Create the required user assigned managed identities

1. Create the following required identities. Azure Red Hat OpenShift requires nine managed identities, each needs to have an assigned, built-in role:

   - Seven managed identities related to core OpenShift operators.
   - One managed identity for the Azure Red Hat OpenShift service operator.
   - One other identity for the cluster to enable the use of these identities.

   The managed identity components are:

- OpenShift Image Registry Operator (image-registry)
- OpenShift Network Operator (cloud-network-config)
- OpenShift Disk Storage Operator (disk-csi-driver)
- OpenShift File Storage Operator (file-csi-driver)
- OpenShift Cluster Ingress Operator (ingress)
- OpenShift Cloud Controller Manager (cloud-controller-manager)
- OpenShift Machine API Operator (machine-api)
- Azure Red Hat OpenShift Service Operator (aro-operator)

There are eight different managed identities and corresponding built-in roles that represent the permissions needed for each component of Azure Red Hat OpenShift to perform its duties. In addition, the platform requires one other identity, the cluster identity, to perform federated credential creation for the previously listed managed identity components (aro-cluster).

For more information about Red Hat OpenShift cluster operators, see Cluster Operators reference.

For more information about managed identities in Azure Red Hat OpenShift, see Understanding managed identities in Azure Red Hat OpenShift.

Create the required identities:

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name aro-cluster
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name cloud-controller-manager
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name ingress
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name machine-api
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name disk-csi-driver
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name cloud-network-config
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name image-registry
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name file-csi-driver
```

Azure CLI

```
az identity create \
--resource-group $RESOURCEGROUP \
--name aro-operator
```

2. Create the required role assignments for each operator identity, cluster identity, and the first party service principal.

> **Note**
>
> This article assumes that only master and worker subnets are present. If you configured more cluster subnets at install time, you need to grant role assignment scope to those subnets, for operators that require it.

The following role assignments for master and worker subnets assume there's no network security group (NSG), route table, or network address translation (NAT) gateway attached. If you bring any of those network resources to the install, you need to create more role assignments that grant operator identities permissions for those extra network resources. For each operator that requires a role assignment for the following subnets or the virtual network, it also requires a role assignment for the extra network resource.

Azure CLI

```
SUBSCRIPTION_ID=$(az account show --query 'id' -o tsv)

# assign cluster identity permissions over identities previously created

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role "/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinitions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope "/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aro-operator"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role "/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinitions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope "/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.ManagedIdentity/userAssignedIdentities/cloud-controller-manager"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role "/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinitions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope "/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.ManagedIdentity/userAssignedIdentities/ingress"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name aro-cluster --query principalId -o tsv)" \
```

```
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Micro
soft.ManagedIdentity/userAssignedIdentities/machine-api"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Micro
soft.ManagedIdentity/userAssignedIdentities/disk-csi-driver"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Micro
soft.ManagedIdentity/userAssignedIdentities/cloud-network-config"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Micro
soft.ManagedIdentity/userAssignedIdentities/image-registry"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-cluster --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/ef318e2a-8334-4a05-9e4a-295a196c6a6e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Micro
soft.ManagedIdentity/userAssignedIdentities/file-csi-driver"

# assign vnet-level permissions for operators that require it, and subnets-
level permission for operators that require it
```

```
az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name cloud-controller-manager --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/a1f96423-95ce-4224-ab27-4e3dc72facd4" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/master"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name cloud-controller-manager --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/a1f96423-95ce-4224-ab27-4e3dc72facd4" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/worker"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name ingress --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/0336e1d3-7a87-462b-b6db-342b63f7802c" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/master"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name ingress --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/0336e1d3-7a87-462b-b6db-342b63f7802c" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/worker"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name machine-api --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/0358943c-7e01-48ba-8889-02cc51d78637" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/master"
```

```
az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name machine-api --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/0358943c-7e01-48ba-8889-02cc51d78637" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/worker"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name cloud-network-config --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/be7a6435-15ae-4171-8f30-4a343eff9e8f" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name file-csi-driver --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/0d7aedc0-15fd-4a67-a412-efad370c947e" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name image-registry --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/8b32b316-c2f5-4ddf-b05b-83dacd2d08b5" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet"

az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-operator --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/4436bae4-7702-4c84-919b-c4069ff25ee2" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/master"
```

```
az role assignment create \
    --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP -
-name aro-operator --query principalId -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/4436bae4-7702-4c84-919b-c4069ff25ee2" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet/subnets/worker"

az role assignment create \
    --assignee-object-id "$(az ad sp list --display-name "Azure Red Hat
OpenShift RP" --query '[0].id' -o tsv)" \
    --assignee-principal-type ServicePrincipal \
    --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinit
ions/42f3c60f-e7b1-46d7-ba56-6de681664342" \
    --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Micro
soft.Network/virtualNetworks/aro-vnet"
```

Create the cluster

To create a cluster, run the following command shown under the options. If you choose to use either of the following options, modify the command accordingly:

- Option 1: You can pass your Red Hat pull secret, which enables your cluster to access Red Hat container registries along with other content. Add the `--pull-secret @pull-secret.txt` argument to your command.
- Option 2: You can use a custom domain. Add the `--domain foo.example.com` argument to your command, replacing `foo.example.com` with your own custom domain.

Create the cluster with the required environment variables. For each `--assign-platform-workload-identity` flag, the first argument represents the key, which tells the Azure Red Hat OpenShift resource provider which OpenShift operator to use for a given identity. The second argument represents the reference to the identity itself.

Azure CLI

```
az aro create \
    --resource-group $RESOURCEGROUP \
    --name $CLUSTER \
    --vnet aro-vnet \
    --master-subnet master \
    --worker-subnet worker \
    --version <VERSION> \
    --enable-managed-identity \
```

```
    --assign-cluster-identity aro-cluster \
    --assign-platform-workload-identity file-csi-driver file-csi-driver \
    --assign-platform-workload-identity cloud-controller-manager cloud-controller-
manager \
    --assign-platform-workload-identity ingress ingress \
    --assign-platform-workload-identity image-registry image-registry \
    --assign-platform-workload-identity machine-api machine-api \
    --assign-platform-workload-identity cloud-network-config cloud-network-config \
    --assign-platform-workload-identity aro-operator aro-operator \
    --assign-platform-workload-identity disk-csi-driver disk-csi-driver
```

As an option, if identity resources exist in a different **region** or **resource group**, you can pass full resource IDs to create. See the following example:

Azure CLI

```
az aro create \
    --resource-group $RESOURCEGROUP \
    --name $CLUSTER \
    --vnet aro-vnet \
    --master-subnet master \
    --worker-subnet worker \
    --version <VERSION> \
    --enable-managed-identity \
    --assign-cluster-identity
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/aro-cluster \
    --assign-platform-workload-identity file-csi-driver
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/file-csi-driver \
    --assign-platform-workload-identity cloud-controller-manager
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/cloud-controller-manager \
    --assign-platform-workload-identity ingress
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/ingress \
    --assign-platform-workload-identity image-registry
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/image-registry \
    --assign-platform-workload-identity machine-api
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/machine-api \
    --assign-platform-workload-identity cloud-network-config
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/cloud-network-config \
    --assign-platform-workload-identity aro-operator
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/aro-operator \
    --assign-platform-workload-identity disk-csi-driver
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/disk-csi-driver
```

Select a different Azure Red Hat OpenShift version

You can choose to use a specific version of Azure Red Hat OpenShift when creating your cluster. First, use the CLI to query for available Azure Red Hat OpenShift versions:

Azure CLI

```
az aro get-versions --location <REGION>
```

Once the version is chosen, specify it using the `--version` parameter in the `az aro create` command.

Clean up

To delete a managed identity cluster, run the following command:

Azure CLI

```
az aro delete -n $CLUSTER -g $RESOURCEGROUP
```

The delete command doesn't clean up the cluster-assigned managed identities that were created as part of the installation. You need to manually delete the identities and role assignments.

Related content

For more information, see Understand managed identities in Azure Red Hat OpenShift.

Last updated on 04/02/2026

Replace platform workload identities and cluster identity

Summarize this article for me

You can replace platform workload identities and a cluster identity for your Azure Red Hat OpenShift cluster. For example, you might need to use an identity in a different resource group, or want to use a new identity with a different name, or to replace an identity that was accidentally deleted. The ability to replace identities is an important feature so a cluster doesn't need to be recreated if an identity is accidentally deleted.

Prerequisites

- Ensure you're using Azure CLI version 2.84.0 or higher, which contains the fully supported CLI arguments for managed identity clusters. Use `az--version` to find the version of Azure CLI you installed. If you need to install or upgrade, see Install Azure CLI.

Create variables

Set the following variables that are used to run the `az` commands in your Bash session. Replace the angle brackets (`<>`) and placeholder values with your cluster's values.

Bash

```
LOCATION=<location of your cluster>
RESOURCEGROUP=<your cluster resource group name>
CLUSTER=<name of your cluster>
SUBSCRIPTION_ID=<subscriptionID>
```

Replace platform workload identities

Use these instructions to replace platform workload identities.

Create platform workload identities

You can create one or more replacement platform workload identities.

Azure CLI

```
az identity create \
  --resource-group $RESOURCEGROUP \
```

```
  --name machine-api-replacement
```

Assign virtual network or subnet permissions

You can assign permissions to the virtual network or subnet. The role assignments are shown in the deployment article Create the required user assigned managed identities.

- If you replace the cluster identity, you have to recreate the eight cluster identity role assignments that are defined in deployment article.
- If you're replacing one operator identity, from the deployment article you need the role assignments that need to be made from that identity over either virtual network or subnets. The exception is the role assignment with parameter `--display-name "Azure Red Hat OpenShift RP"`.
- If you're replacing all identities, you need to recreate all of the role assignments shown in the deployment article.

Azure CLI

```
az role assignment create \
  --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name
machine-api-replacement --query principalId -o tsv)" \
  --assignee-principal-type ServicePrincipal \
  --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinitions/
0358943c-7e01-48ba-8889-02cc51d78637" \
  --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Microsoft.
Network/virtualNetworks/aro-vnet/subnets/master"

az role assignment create \
  --assignee-object-id "$(az identity show --resource-group $RESOURCEGROUP --name
machine-api-replacement --query principalId -o tsv)" \
  --assignee-principal-type ServicePrincipal \
  --role
"/subscriptions/$SUBSCRIPTION_ID/providers/Microsoft.Authorization/roleDefinitions/
0358943c-7e01-48ba-8889-02cc51d78637" \
  --scope
"/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RESOURCEGROUP/providers/Microsoft.
Network/virtualNetworks/aro-vnet/subnets/worker"
```

> **Important**
>
> It's important that the role assignments are redone before your run the `az aro update` command.

Run the following command to replace platform workload identities.

Azure CLI

```
az aro update -n $CLUSTER -g $RESOURCEGROUP \
  --assign-platform-workload-identity machine-api machine-api-replacement
```

In `--assign-platform-workload-identity machine-api machine-api-replacement` command flag, the first argument `machine-api` is the key that tells Azure Red Hat OpenShift which operator you're assigning the identity to and that value doesn't change.

The second argument `machine-api-replacement` represents the name of the identity that you're replacing the old one with. If it's in the same resource group as the Azure Red Hat OpenShift resource, you can just pass the name. If it's in a different resource group, you need to pass a full resource ID there. For more information about the functionality, see create the cluster.

Replace the cluster identity

Use these instructions to replace the cluster identity.

Create a replacement cluster identity

The name for the cluster identity is your choice and this example uses `aro-cluster-replacement`.

Azure CLI

```
az identity create \
  --resource-group $RESOURCEGROUP \
  --name aro-cluster-replacement
```

Assign cluster identity permissions

Assign cluster identity permissions for this replacement identity over the operator identities. For more information, see Create the required user assigned managed identities.

Replace the cluster identity

The name of the cluster identity in the parameter `--assign-cluster-identity` should match the name you created to replace the original cluster identity name.

Azure CLI

```
az aro update -n $CLUSTER -g $RESOURCEGROUP \
  --assign-cluster-identity aro-cluster-replacement
```

As an option, if the replacement identities exist in a different region or resource group, you can pass full resource IDs as shown in the following commands.

Azure CLI

```
az aro update -n $CLUSTER -g $RESOURCEGROUP \
  --assign-cluster-identity
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/aro-cluster-replacement

az aro update -n $CLUSTER -g $RESOURCEGROUP \
  --assign-platform-workload-identity machine-api
/subscriptions/$SUBSCRIPTION_ID/resourcegroups/$RESOURCEGROUP/providers/Microsoft.M
anagedIdentity/userAssignedIdentities/machine-api-replacement
```

Related content

- Learn more about managed identities.
- Create an Azure Red Had OpenShift cluster with managed identities.

Last updated on 03/11/2026

Deploy and configure an application using workload identity on an Azure Red Hat OpenShift managed identity cluster

This article shows how to:

- Create a Microsoft Entra Workload ID and Kubernetes service account.
- Configure the managed identity for token federation.
- Create a Microsoft Azure Key Vault with a new secret and role assignments (for a demo).
- Deploy the workload and verify authentication with the workload identity.
- Grant a pod in the cluster access to secrets in an Azure key vault.

The process entails:

- Obtain the OpenID Connect (OIDC) issuer URL.
- Create a user assigned managed identity for your application.
- Perform the role assignment on the desired resource.
- Create a Kubernetes service account.
- Set the annotation on the service account.
- Create the federated credential.
- Deploy the application and ensure that:
  - The `.spec.serviceAccountName` is set.
  - The label `azure.workload.identity/use: "true"` is set.

Prerequisites

You need an existing managed identity Azure Red Hat OpenShift cluster.

Verify pod-identity-webhook deployment

There's a mutating admission webhook which projects a signed service account token to a well-known path and injects authentication-related environment variables to the application pods. For more information, see Mutating admission webhook.

Verify that the annotation `target.workload.openshift.io/management` is set on the pod-identity-webhook deployment in the `openshift-cloud-credential-operator` namespace by running the following command:

Bash

```
oc describe deployment pod-identity-webhook -n openshift-cloud-credential-operator |
grep 'target.workload.openshift.io/management'
```

You should see a response like:

Bash

```
Annotations: target.workload.openshift.io/management: {"effect":
"PreferredDuringScheduling"}
```

If you find that the annotation is missing, open a support case.

Export environment variables

Begin by setting environment variables matching your existing managed identity Azure Red Hat OpenShift cluster. Make sure to set the correct `RESOURCE_GROUP`, `LOCATION`, `CLUSTER_NAME`, namespaces, and desired identity names:

Bash

```
export KEYVAULT_NAME="azwi-kv-$(openssl rand -hex 2)"
export KEYVAULT_SECRET_NAME="my-secret"
export KEYVAULT_LOCATION="eastus"
export RESOURCE_GROUP="<ARO_CLUSTER_RESOURCE_GROUP>"
export CLUSTER_NAME="<ARO_CLUSTER_NAME>"
export SUBSCRIPTION="$(az account show --query id --output tsv)"
export USER_ASSIGNED_IDENTITY_NAME="example-user-assigned-identity"
export FEDERATED_IDENTITY_CREDENTIAL_NAME="example-federated-identity"
export SERVICE_ACCOUNT_NAMESPACE="example-project"
export SERVICE_ACCOUNT_NAME="example-workload-identity-sa"
```

Retrieve the OIDC issuer URL

Obtain the Azure Red Hat OpenShift cluster OIDC issuer URL using OpenShift CLI (OC) and set the environment variable:

Bash

```
export ARO_OIDC_ISSUER="$(oc get authentication cluster -o
jsonpath='{.spec.serviceAccountIssuer}')"
```

Optionally, if you don't have cluster access, get the Azure Red Hat OpenShift cluster OIDC issuer URL by using the Azure CLI:

Bash

```
export ARO_OIDC_ISSUER="$(az aro show --subscription "${SUBSCRIPTION}" \
    --name "${CLUSTER_NAME}" \
    --resource-group "${RESOURCE_GROUP}" \
    --query "clusterProfile.oidcIssuer" \
    --output tsv)"
```

Verify the correct URL is set:

Bash

```
echo $ARO_OIDC_ISSUER
https://eastus.oic.aro.azure.net/00000000-0000-0000-0000-000000000000/11111111-1111-
1111-1111-111111111111
```

By default, the issuer is set to use the base URL `https://{region}.oic.aro.azure.net/{tenant_id}/{uuid}`, where the value for `{region}` matches the location to which the Azure Red Hat OpenShift cluster is deployed. The value `{uuid}` represents the OIDC key, which is an immutable, randomly generated, cluster-specific GUID.

Create an example resource

In this example, a new Microsoft Azure Key Vault with a new secret is created and later accessed via Microsoft Entra Workload ID. Use the Azure `az` command line tool to create the key vault that is later accessed.

Ensure the Azure role-based access control (Azure RBAC) Key Vault Secrets Officer role is assigned to yourself.

Bash

```
# If necessary, create the Resource Group
az group create --resource-group "${RESOURCE_GROUP}" --location
"${KEYVAULT_LOCATION}"

# Create the Keyvault and set a secret value
az keyvault create --resource-group "${RESOURCE_GROUP}" --location
"${KEYVAULT_LOCATION}" --name "${KEYVAULT_NAME}"
az keyvault secret set --vault-name "${KEYVAULT_NAME}" --name
"${KEYVAULT_SECRET_NAME}" --value "Hello world"

# Create an environment variable for the key vault URL:
export KEYVAULT_URL="$(az keyvault show --resource-group ${RESOURCE_GROUP} --name
${KEYVAULT_NAME} --query properties.vaultUri --output tsv)"

# Create an environment variable for the key vault Resource ID:
```

```
export KEYVAULT_RESOURCE_ID=$(az keyvault show --resource-group "${RESOURCE_GROUP}"
--name "${KEYVAULT_NAME}" | jq -r '.id')
```

Create a managed identity and grant permission to access Azure Key Vault

In the next steps, create a user-assigned managed identity and the corresponding role assignment:

Bash

```
# Create the identity
az identity create --name "${USER_ASSIGNED_IDENTITY_NAME}" --resource-group
"${RESOURCE_GROUP}"

# Assign Key Vault Secrets User to the identity
export USER_ASSIGNED_IDENTITY_OBJECT_ID="$(az identity show --name
"${USER_ASSIGNED_IDENTITY_NAME}" --resource-group "${RESOURCE_GROUP}" --query
'principalId' -o tsv)"
export USER_ASSIGNED_IDENTITY_CLIENT_ID="$(az identity show --name
"${USER_ASSIGNED_IDENTITY_NAME}" --resource-group "${RESOURCE_GROUP}" --query
'clientId' -o tsv)"

az role assignment create --assignee-object-id "${USER_ASSIGNED_IDENTITY_OBJECT_ID}"
--role "Key Vault Secrets User" --scope "${KEYVAULT_RESOURCE_ID}" --assignee-
principal-type ServicePrincipal
```

Create a Kubernetes service account

- In OpenShift Container Platform, create the Service Account with the `azure.workload.identity/client-id` set to the `USER_ASSIGNED_IDENTITY_CLIENT_ID` previously created.
- For more information about creating service accounts, see Creating service accounts.

Create the new project for this example app:

Bash

```
oc new-project ${SERVICE_ACCOUNT_NAMESPACE}
```

Create the service account:

Bash

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ${SERVICE_ACCOUNT_NAME}
  namespace: ${SERVICE_ACCOUNT_NAMESPACE}
  annotations:
    azure.workload.identity/client-id: ${USER_ASSIGNED_IDENTITY_CLIENT_ID}
EOF
```

Create the federated identity credential

Create an Azure federated identity credential that links the service account in OpenShift Container Platform to the Azure user-assigned managed identity:

Azure CLI

```
az identity federated-credential create \
--name "${FEDERATED_IDENTITY_CREDENTIAL_NAME}" \
--identity-name "${USER_ASSIGNED_IDENTITY_NAME}" \
--resource-group "${RESOURCE_GROUP}" \
--issuer "${ARO_OIDC_ISSUER}" \
--subject
"system:serviceaccount:${SERVICE_ACCOUNT_NAMESPACE}:${SERVICE_ACCOUNT_NAME}"
```

Deploy the application

The OpenShift Container Platform Service Account is now configured as expected. Note the `.spec.serviceAccountName` is set and the label `azure.workload.identity/use: "true"` is used:

YAML

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: quick-start
  namespace: ${SERVICE_ACCOUNT_NAMESPACE}
  labels:
    azure.workload.identity/use: "true"
spec:
  serviceAccountName: ${SERVICE_ACCOUNT_NAME}
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containers:
    - image: ghcr.io/azure/azure-workload-identity/msal-go
```

```
      name: oidc
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          drop: [ "ALL" ]
      env:
        - name: KEYVAULT_URL
          value: ${KEYVAULT_URL}
        - name: SECRET_NAME
          value: ${KEYVAULT_SECRET_NAME}
EOF
```

Verify the deployed workload

In the newly deployed workload, verify that the environment variables and projected volume are both present. Run the following command:

Bash

```
oc describe pod quick-start
```

See the following example output:

Bash

```
[..]
 Environment:
   KEYVAULT_URL:                https://azwi-kv-45ff.vault.azure.net/
   SECRET_NAME:                 my-secret
   AZURE_CLIENT_ID:             00001111-aaaa-2222-bbbb-3333cccc4444
   AZURE_TENANT_ID:             aaaabbbb-0000-cccc-1111-dddd2222eeee
   AZURE_FEDERATED_TOKEN_FILE:  /var/run/secrets/azure/tokens/azure-identity-token
   AZURE_AUTHORITY_HOST:        https://login.microsoftonline.com/
[..]
Volumes:
[..]
 azure-identity-token:
   Type:                    Projected (a volume that contains injected data from
multiple sources)
   TokenExpirationSeconds:  3600
```

Verify in the logs that the workload was able to access the Microsoft Azure Key Vault using the injected keys:

Bash

```
$ oc logs quick-start
I0816 09:43:37.961113       1 main.go:63] "successfully got secret" secret="Hello
```

```
world"
```

Next steps

- Deploy and configure workload identity on an Azure Kubernetes Service (AKS) cluster.
- Example for using Microsoft Entra Workload ID in OpenShift Container Platform .

Last updated on 02/02/2026

Upgrade a cluster with managed identities enabled

Summarize this article for me

This article shows you how to upgrade an Azure Red Hat OpenShift cluster with managed identities enabled using the OpenShift web console or the managed-upgrade-operator (MUO).

As part of the Azure Red Hat OpenShift cluster lifecycle, you need to perform periodic upgrades to the latest version of the OpenShift platform. Upgrading your Azure Red Hat OpenShift clusters enables you to obtain the latest features and functionalities and apply the latest security releases.

Prerequisites

Before you begin, verify you meet the following requirements needed to be successful in upgrading a cluster.

- You are using Azure CLI version 2.84.0 or higher, which contains the fully supported CLI arguments for managed identity clusters. Run `az --version` to find your current version. If you need to install or upgrade the Azure CLI, see Install Azure CLI.
- You have access to an existing Azure Red Hat OpenShift cluster as a user with admin privileges.
- You updated your Azure Red Hat OpenShift pull secret for an existing Azure Red Hat OpenShift 4.x cluster, including the **cloud.openshift.com** entry from your pull secret enables your cluster to start sending telemetry data to Red Hat.
  For more information, see Add or update your Red Hat pull secret on an Azure Red Hat OpenShift 4 cluster.
- Verify the federated identity credentials used by the cluster's managed identities are valid/updated before starting the upgrade. For more information, see Reconcile federated identity credentials for your Azure Red Hat OpenShift cluster with managed identities enabled.

Set the upgradeable-to annotation on the Azure Red Hat OpenShift cluster's CloudCredential resource

Before you can upgrade an Azure Red Hat OpenShift cluster with managed identity enabled, set the `upgradeable-to` annotation on the cluster's `CloudCredential` resource using the `az aro`

`update` command:

Azure CLI

```
az aro update --name <CLUSTER_NAME> --resource-group <RESOURCE_GROUP> --
upgradeable-to <VERSION>
```

Where:

- `--name` is the name of the cluster
- `--resource-group` is the name of the network resource group. You can configure the default group using `az-config --defaults group=<name>`.
- `--upgradeable-to` is the OpenShift version number you intend to upgrade to, specified in the format x.y.z

For more information about the upgradeable-to annotation, see Preparing to update a cluster.

> **Note**
>
> The previously shown `az aro update` command doesn't trigger the OpenShift version update. To complete the OpenShift version update, finish the remaining steps in this article.

Check for Azure Red Hat OpenShift cluster upgrades using the web console

1. From the left navigation menu of the OpenShift web console (the default when you sign in as the kubeadmin), select the **Administration** tab.
2. Select **Cluster Settings** and open the **Details** tab. You should see the version, update status, and channel. The channel isn't configured by default.
3. Select the **Channel** link, and at the prompt enter the desired update channel, for example **stable-4.16**. Once the desired channel is chosen, a graph that shows available releases and channels is displayed. If the **Update Status** for your cluster shows **Updates Available**, you can update your cluster.

Upgrade your Azure Red Hat OpenShift cluster with the OpenShift web console

From the OpenShift web console in the previous step, set the channel according to the version that you want to update to, such as `stable-4.16`. For more information, see Update channels.

Selection a version to update to, and select **Update**. You see the update status change to: `Update to <product-version> in progress`. You can review the progress of the cluster update by watching the progress bars for the operators and nodes.

Schedule individual upgrades using the managed-upgrade-operator

Use the managed-upgrade-operator (MUO) to upgrade your Azure Red Hat OpenShift cluster.

The MUO manages automated cluster upgrades. The MUO starts the cluster upgrade, but it doesn't perform any activities of the cluster upgrade process itself. The OpenShift Container Platform (OCP) is responsible for upgrading the clusters. The goal of the MUO is to satisfy the operating conditions that a managed cluster must hold, both before and after starting the cluster upgrade.

1. Prepare the configuration file, as shown in the following example for upgrading to OpenShift 4.16.

   YAML

   ```yaml
   apiVersion: upgrade.managed.openshift.io/v1alpha1
   kind: UpgradeConfig
   metadata:
     name: managed-upgrade-config
     namespace: openshift-managed-upgrade-operator
   spec:
     type: "ARO"
     upgradeAt: "2025-04-08T03:20:00Z"
     PDBForceDrainTimeout: 60
     desired:
       channel: "stable-4.16"
       version: "4.16.37"
   ```

   Where:

   - `channel` is the channel the configuration file pulls from, according to the lifecycle policy.
   - `version` is the version that you wish to upgrade to, such as `4.16.37`.
   - `upgradeAT` is the time when the upgrade takes place.

2. Apply the configuration file:

Bash

```
$ oc create -f <file_name>.yaml
```

Next steps

- Learn to upgrade an Azure Red Hat OpenShift cluster using the OC CLI.
- You can find information about available OpenShift Container Platform advisories and updates in the errata section of the Customer Portal.

Last updated on 03/11/2026

Reconcile federated identity credentials for your Azure Red Hat OpenShift cluster with managed identities enabled

Summarize this article for me

In this article, learn how to reconcile federated identity credentials for OpenShift operator managed identities in your Azure Red Hat OpenShift clusters with managed / workload identity enabled.

Reconciling the federated identity credentials for the OpenShift operator managed identities can be needed if any OpenShift operators are unable to authenticate to Azure. This procedure ensures the federated identity credentials for each of the OpenShift operators exist, are configured correctly, and are correctly deployed into the in-cluster secrets so that the OpenShift operators can use them.

Prerequisites

This article assumes that the following conditions are met.

- You have an existing Azure Red Hat OpenShift cluster with managed / workload identity enabled with the latest updates applied.
- You are using using Azure CLI version 2.84.0 or higher, which contains the fully supported CLI arguments for managed identity clusters.
- If you need to check the version of Azure CLI, run:

  Azure CLI

  ```
  az --version
  ```

- To install or upgrade the Azure CLI, follow Install Azure CLI.

Reconcile federated identity credentials

> **Important**
>
> The `az aro update` command can take up to two hours depending on the cluster state (for example, it can take longer for a larger cluster, etc.). Smaller clusters can take less time.

The command reconciles federated identity credentials without requiring any special arguments. There are no adverse effects of running this command on healthy clusters.

To update the resource and reconcile the federated identity credentials, run:

Azure CLI

```
az aro update --name <CLUSTER_NAME> --resource-group <RESOURCE_GROUP>
```

Next steps

- Learn more about managed identities.
- Create an Azure Red Had OpenShift cluster with managed identities.

Last updated on 03/11/2026

Configure an Azure File StorageClass (preview)

08/22/2025

In this article, learn how to configure an Azure File StorageClass.

The Azure File Container Storage Interface (CSI) driver has a dependency on shared access keys. The default Azure File `StorageClass` is disabled in clusters with managed identity enabled by default and is optional to turn on for your cluster. If you want to use Azure File in Azure Red Hat OpenShift, you need to create your own `storageclass` that uses shared keys to access the backing storage.

To use the Azure File in Azure Red Hat OpenShift, create a storage class with parameters `tag` and `matchTags`. The driver creates a new storage account for use with shared key access enabled. These parameters are required. If the operator attempts to use the existing cluster storage accounts for backing storage, it fails because the shared key access isn't enabled.

For more information, see Create an Azure Files StorageClass on Azure Red Hat OpenShift.

Use the following code to create a file for the Azure File `StorageClass` manifest.

> **Note**
>
> Don't use the name *azurefile-csi* for a custom storage class with managed identities because the service removes it.

YAML

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azure-file
provisioner: file.csi.azure.com
mountOptions:
  - dir_mode=0777
  - file_mode=0777
  - uid=0
  - gid=0
  - mfsymlinks
  - cache=strict
  - actimeo=30
  - noperm
parameters:
  location: $LOCATION
  secretNamespace: kube-system
```

```
  skuName: Standard_LRS
  resourceGroup: $AZURE_FILES_RESOURCE_GROUP
  tags: $TAG # must match key=value format
  matchTags: "true"
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Run the following command to create the storage class.

Bash

```
oc create -f azure-storageclass-azure-file.yaml
```

Next steps

- Learn more about managed identities.
- Create an Azure Red Had OpenShift cluster with managed identities.

OpenShift Virtualization for Azure Red Hat OpenShift

OpenShift Virtualization for Azure Red Hat OpenShift, a self-managed add-on to Red Hat OpenShift Container Platform (OCP), allows users to run and manage virtual machines (VM) alongside containers on the same OpenShift cluster. This integration provides a unified platform for diverse workloads, simplifying management and operations. This ability allows for easy migration and deployment of traditional virtual machines onto a trusted, consistent, and comprehensive cloud application platform. When deployed on Azure Red Hat OpenShift, OpenShift Virtualization uses Azure's robust infrastructure to deliver a scalable and resilient environment for your virtualized workloads.

Key Benefits

- **Unified platform:** Consolidate your containerized and virtualized applications on a single platform, reducing operational complexity.
- **Scalability:** Use Azure's elastic infrastructure to scale your virtual machines and containerized applications as needed.
- **Simplified management:** Use OpenShift's familiar tools and workflows to manage both containers and virtual machines.
- **Cost efficiency:** Optimize resource utilization by running VMs and containers on shared infrastructure.

OpenShift Container Platform version support

OpenShift Virtualization on Azure Red Hat OpenShift is supported only from **version 4.18 onwards**. Ensure your cluster meets this version requirement in order to utilize OpenShift Virtualization capabilities on Azure Red Hat OpenShift. Running on earlier versions isn't supported and might lead to unexpected behavior or functionality issues. If you need to update your cluster, see the Red Hat documentation for Updating clusters.

Important concepts

- OpenShift Virtualization requires a persistent volume storage location. For more information, see Supported persistent storage options. Azure NetApp Files support for OpenShift Virtualization is in preview.
- OpenShift Virtualization on Azure Red Hat OpenShift is only supported on the Dsv5 and Dsv6 family of nodes with eight (8) cores or higher. These instance types make use of Azure Boost technologies to improve performance.

- You need an Azure Red Hat OpenShift cluster with at least version 4.18.
- Dsv6 family of nodes are currently supported in preview and are supported starting with Azure Red Hat OpenShift version 4.19.
- For more information about available versions, see Azure Red Hat OpenShift release calendar.
- Performance of an application workload varies depending on the setup of the Virtualization environment. Refer to the best practices article for information on right-sizing your workloads and fine-tuning methods.

Supported Azure instance types

OpenShift Virtualization on Azure supports a range of instance types, allowing you to choose the appropriate compute resources for your virtual machines. The following list details the currently supported Azure instance types:

- Dsv5 Series (4.18+) - Use a SKU with minimum of eight (8) cores.
- Dsv6 Series (preview in 4.19+) - Use a SKU with minimum of eight (8) cores.

Supported persistent storage options

OpenShift Virtualization requires a persistent volume storage location. While you could use any storage available to the cluster, like Azure Files or Azure Disk, or any other storage options supporting the container storage interface (CSI), a storage class that provides block storage and/or supports the `ReadWriteMany` (RWX) access mode is recommended for storage. The following storage options are validated for use with OpenShift Virtualization:

- **OpenShift Data Foundation (ODF)**: OpenShift Data Foundation (ODF) provides persistent storage for applications running on OpenShift, offering advanced features like data replication, encryption, and disaster recovery.
- **Azure NetApp Files (ANF):** Azure NetApp Files (ANF) is a high-performance, enterprise-grade file storage service built on NetApp's ONTAP technology. ANF offers performance, scalability, and advanced data management features, which makes it ideal for demanding workloads like OpenShift Virtualization.
- **Azure Disk:** Azure Disk provides persistent, high-performance block storage offering a scalable and durable solution for applications and data. Live migration isn't supported with Azure Disks as your persistent storage.

Prerequisites

- Sufficient quota for an Azure Red Hat OpenShift cluster with at least one D8sv5 SKU in your subscription. For three worker nodes of eight (8) cores each, a **minimum of 52 cores** are needed to create a cluster.
- At least one worker node of D8sv5 or higher. For more information about creating worker nodes, see Overview of machine management.
- A running Azure Red Hat OpenShift cluster with at least version 4.18. For more information, see Create a cluster and Upgrade an Azure Red Hat OpenShift cluster.

Installation

1. After you have an Azure Red Hat OpenShift cluster, ensure that you have at least one worker node from the list of supported Azure instance types.

   > **Note**
   >
   > If you want to use the Dsv6 worker node, ensure that you provide the following values in the `MachineSet` YAML.
   >
   > - `spec.template.spec.providerSpec.value.image.sku` should be set to 419-v2.
   > - `spec.template.spec.providerSpec.value.image.version` should be set to 419.6.20250523.
   >
   > You can find this information using the following command:
   >
   > `az vm image list --architecture x64 -o table --all --offer aro4 --publisher azureopenshift`

2. Install the OpenShift Virtualization operator. For more information, see Installing OpenShift Virtualization.

3. (Recommended) Proceed to set up a persistent storage provider such as Red Hat OpenShift Data Foundation or Azure NetApp Files.

   > **Note**
   >
   > After installation, you might need to change the default storage class on the cluster to be one of the classes for your selected persistent storage. For more information, see Changing the default storage class.

Post installation steps and VM creation

There are typically a few procedures that are performed after installing OpenShift Virtualization. For more information, see Postinstallation configuration.

You're now ready to create a virtual machine. For more information, see Creating a virtual machine. If you're moving many workloads, see the Migration Toolkit for Virtualization to learn about large migrations of virtualization workloads to OpenShift Virtualization.

Licensing Windows VMs on OpenShift Virtualization

Windows Server VMs, licensed with Windows Server licenses with either active software assurance or subscription, are supported on OpenShift Virtualization on Azure Red Hat OpenShift through Azure Hybrid Benefit. If the license is for a Datacenter edition, it can only be used for one OpenShift Virtualization VM. This license does not allow you to run multiple VMs on OpenShift Virtualization host. If you're looking to procure new Windows server licenses, existing volume agreements with Microsoft or existing partner channels need to be utilized.

Subscriptions for RHEL VMs on OpenShift Virtualization

Red Hat Subscriptions can be consumed on OpenShift Virtualization in Azure through the Red Hat Cloud Access Program. For eligibility details, refer to this link: https://www.redhat.com/en/technologies/cloud-computing/cloud-access.

Additionally, OpenShift Virtualization on ARO includes guest subscriptions for Red Hat Enterprise Linux (RHEL) based on the number of vCPUs on the host:

- **Hosts with 96 or more vCPUs:** Unlimited RHEL guest subscriptions are included.
- **Hosts with fewer than 96 vCPUs:** You can run RHEL guests with a guest vCPU to host vCPU ratio of up to 8:1.
  - For example a host with 64 vCPUs (such as `Standard_D64s_v5`) can run up to 512 RHEL guest vCPUs (64 host vCPUs x 8 = 512 guest vCPUs).

Related content

Learn more about OpenShift Container Platform OpenShift Virtualization.

Last updated on 03/17/2026

Configure Azure NetApp Files for OpenShift Virtualization on Azure Red Hat OpenShift (preview)

Azure NetApp Files is an enterprise-class, high-performance, metered file storage service. It supports the most demanding enterprise file-workloads in the cloud, including databases and high-performance computing applications with no code changes.

Azure NetApp Files supports OpenShift Virtualization on Azure Red Hat OpenShift using the Trident CSI driver. The certified Trident Operator enables consumption and management of storage resources and can be deployed onto Azure Red Hat OpenShift from the OperatorHub. This allows Azure Red Hat OpenShift clusters to automatically create Azure NetApp Files volumes as persistent volumes for virtual machine (VM) disks. Azure NetApp Files offers fast VM provisioning, instant cloning, and live migration for OpenShift Virtualization.

When a new VM is deployed in Azure Red Hat OpenShift, Trident automatically provisions an NFS volume on Azure NetApp Files to store the VM's disks, tailoring capacity and performance on the selected Azure NetApp Files service level (Standard, Premium, Ultra, or Flexible). Multiple OpenShift nodes can simultaneously access the same volume, enabling seamless VM migration without any interruption in disk access.

Prerequisites

- A Microsoft Azure Red Hat OpenShift cluster running version 4.18 or greater

  > ⓘ **Note**
  >
  > Review the **upgrade guidance**, especially if you're running a version earlier than 4.17.x.

- OpenShift Virtualization for Azure Red Hat OpenShift, which can be deployed on OperatorHub or the OpenShift console

- NetApp Trident Version 25.6.2 or later Follow the instructions to deploy Trident operator from Red Hat OpenShift OperatorHub and deploy the Trident orchestrator into the OpenShift cluster. The examples on this page assume that Trident orchestrator is deployed into the `trident` namespace on the OpenShift cluster.

- Azure NetApp Files with at least one capacity pool using the Flexible, Premium, Standard, or Ultra service level.

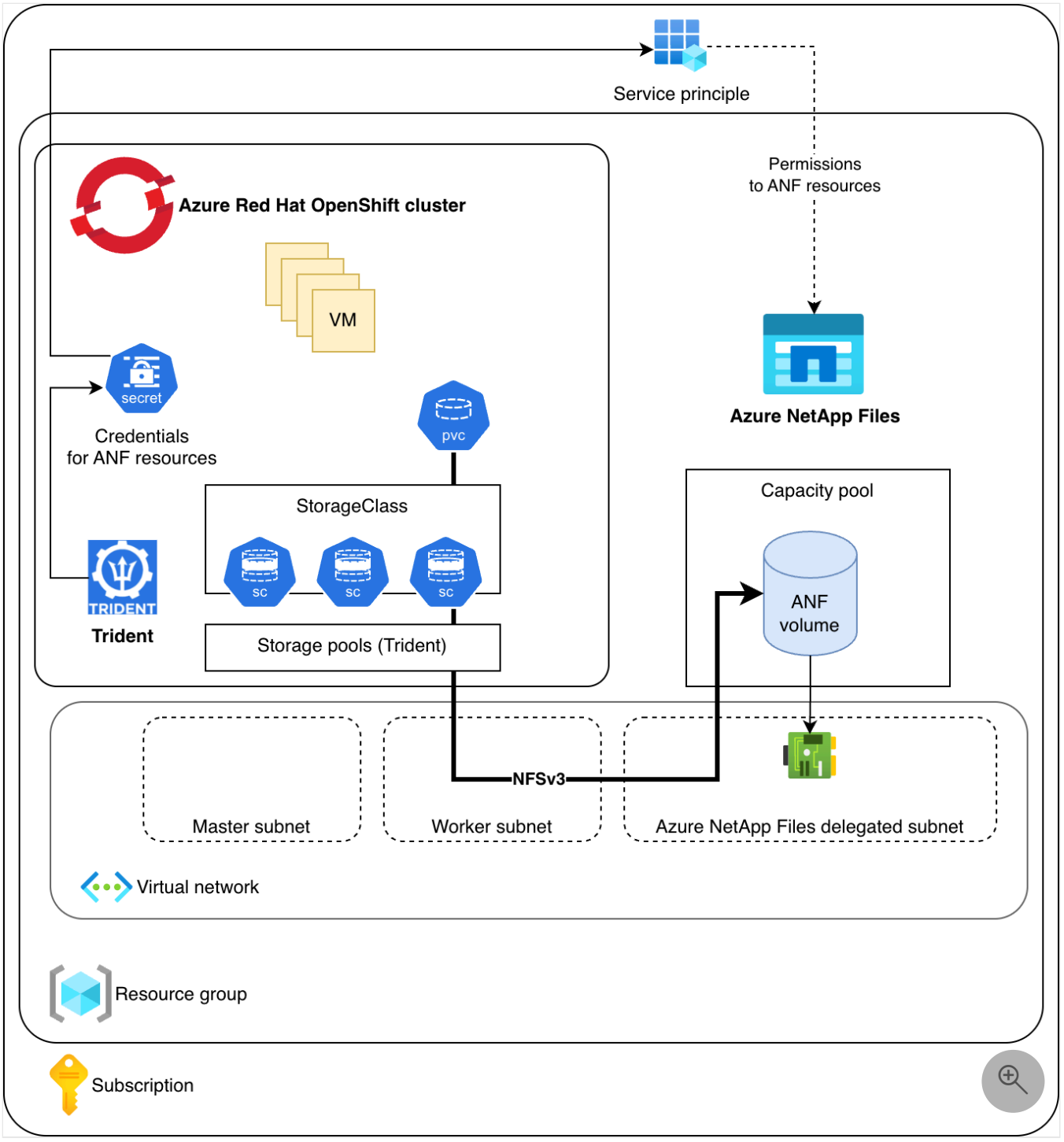
If this is your first time using Azure NetApp Files, see the quickstart guide.

The examples on this page assume that one Flexible service level capacity pool of Flexible an Azure NetApp Files delegated subnet exists on the virtual network used by Azure Red Hat OpenShift. The Flexible service level is recommended to control capacity and throughput for individual Azure NetApp Files volumes that contain the individual VM disk.

> **Note**
>
> Ensure there's sufficient capacity and throughput in your capacity pool for your VM disks. For more information, see Azure NetApp Files service levels and Azure NetApp Files performance calculator.

Architecture

This page details the setup of Azure NetApp Files for OpenShift Virtualization and the configuration steps for the Trident and its virtual storage pools as well as corresponding Kubernetes storage classes as shown in the diagram. It offers examples for one basic storage class with one throughput setting and for three storage classes with differing throughput characteristics.



Before you begin

This configuration process uses the built-in Contributor role for the service principle used by Trident. If you don't want to use the default Contributor role, you can create a custom role to grant only the required privileges to Trident.

Configure Trident for Azure NetApp Files

1. Create the service principal for the resource group that includes the Azure NetApp Files resources (NetApp account).

Azure CLI

```
az ad sp create-for-rbac --name trident --role Contributor --scopes
/subscriptions/<Subscription_ID>/resourceGroups/<Resource_Group>
```

The command outputs an `appId` and `password`. Make note of these outputs; they're required in the next step to create a secret for the Trident service principal.

JSON

```
{
  "appId": "<appID>",
  "displayName": "trident",
  "password": "<password>",
  "tenant": "<tenant>"
}
```

2. In the OpenShift console, create the secret with the credentials from the Trident service principal to manage the Azure NetApp Files resources.

Bash

```
oc create secret generic anf-credentials --from-literal=clientID=<appID> --
from-literal=clientSecret=<password> -n trident
```

3. Configure Azure NetApp Files backend for Trident. Import YAML using the OpenShift console.
   a. Log in to your OpenShift web console.
   b. Select the + icon in the masthead then **Import YAML**.
   c. Paste the YAML directly into the editor or create a file and upload it with the **Upload** button.

One storage class

This example configuration establishes one virtual storage pool in the Trident backend that is used by one `StorageClass` later. The virtual storage pool uses the Flexible service level capacity pool with manual QoS that's assigned 60 MB/s for every volume created.

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: <ANF_TridentBackendConfig_name>
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: azure-netapp-files
  credentials:
      name: anf-credentials
  subscriptionID: <Subscription_ID>
  tenantID: <Tenant_ID>
  location: <region>
  networkFeatures: Standard
  virtualNetwork: <Resource_Group/Virtual_Network_used_by_ARO>
  subnet:
<Resource_Group/Virtual_Network_used_by_ARO/Delegate_subnet_for_ANF>
  nfsMountOptions: nfsvers=3,nconnect=4
  defaults:
    unixPermissions: "0777"
    maxThroughput: "60"
    qosType: "Manual"
  labels:
    qos: manual60mbps
```

4. Confirm Azure NetApp Files backend configuration for Trident.

   a. Log in to your OpenShift console.

   b. In the sidebar, select **Home** then **Search**.

   c. Select your **TridentBackendConfig** resource.

   d. From the **Resources** drop-down, select **TridentBackendConfig**.

   e. From the **Projects** drop-down, select **All Projects**.

   f. From the **TridentBackendConfig** list, select `TridentBackendConfig_name`.

   g. Select **YAML**.

   h. Confirm the following `TridentBackendConfig` settings:

YAML

```
status:
    backendInfo:
    backendName: <TridentBackendConfig_name>
    backendUUID: <TridentBackendConfig_ID>
    deletionPolicy: delete
    lastOperationStatus: Success
    message: Backend '<TridentBackendConfig_name>' updated
    phase: Bound
```

5. Configure the storage class to use Azure NetApp Files.

a. Select the + icon in the masthead then **Import YAML**.
b. Paste the YAML directly into the editor, or create a file and upload it with the **Upload** button.

One storage class

This storage class uses the one virtual storage pool in the Trident backend based on the `qos` label.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <ANF_StorageClass_name>
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  selector: qos=manual60mbps
reclaimPolicy: Delete
allowVolumeExpansion: true
```

6. Configure the volume snapshot class for Azure NetApp Files. Select the + icon in the masthead then **Import YAML**.

   Paste the YAML directly into the editor, or create a file and upload it with the **Upload** button.

YAML

```yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: <ANF_VolumeSnapshotClass_name>
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

7. In the OpenShift console, modify the storage profile for the Azure NetApp Files based storage classes so that `ReadWriteMany (RWX)` is the default. This modification allows the VM that uses VM disks in those storage classes to use live migration.

Windows

Windows Command Prompt

```
# Windows:
oc patch storageprofile <ANF_StorageClass_name> --type=json -p "[{\"op\":
\"replace\", \"path\": \"/spec/claimPropertySets\", \"value\":
[{\"accessModes\": [\"ReadWriteMany\"], \"volumeMode\": \"Filesystem\"}]}]"
```

8. Set the `AzureNetAppFiles_StorageClass_name` as a default.
   a. In the OpenShift console's sidebar, select **Storage** then **Storage Classes**
   b. Select the Action menu ⋮ for the `AzureNetAppFiles_StorageClass_name` then **Set as default**.

Next steps

- Create a virtual machine from a template

Further resources

- Azure NetApp Files
- Learn about Trident
- Virtual storage pools in Trident
- Trident backend configuration options for Azure NetApp Files

---

Last updated on 11/18/2025

Best practices for virtual machine deployments on OpenShift Virtualization

Summarize this article for me

This document provides guidance for optimizing performance and cost efficiency when deploying virtual machines (VMs) using OpenShift Virtualization on Azure Red Hat OpenShift. This guidance also addresses any concerns around application performance and provides actionable steps for successful deployment.

Approach to optimization

> **Note**
>
> GPU-dependent workloads are currently not supported on OpenShift Virtualization on Azure. Plan your deployments accordingly.

Optimizing VM deployments begins with understanding your application workloads and aligning infrastructure choices accordingly. OpenShift Virtualization deployment on Azure Boost machines, the cluster's worker nodes, introduces architectural overhead compared to native VM or pod deployments. Planning for capacity and performance should account for this overhead.

Workload identification

Before provisioning VMs, categorize your workloads to determine their performance and resource requirements. Common workload types include:

- **General purpose:** Web servers, application servers, content management systems.
- **Database:** Relational and NoSQL databases requiring consistent input/output operations per second (IOPS) and memory.
- **Real-time analytics:** Low-latency data processing, operational dashboards.
- **AI/ML:** Compute-intensive workloads requiring high CPU/GPU and memory.
- **Data streaming & messaging:** High-throughput, low-latency event-driven architectures.
- **Batch processing:** Periodic or on-demand jobs processing large data volumes.
- **High-performance computing (HPC):** Scientific simulations, financial modeling.
- **Edge and IoT:** Aggregating and processing data from distributed sensors.
- **Media processing:** Video encoding/decoding, image transformation, streaming.
- **Dev/Test environments:** Temporary environments for development and testing.

Each workload type has unique characteristics that influence VM sizing, storage configuration, and performance tuning strategies.

Right sizing your application Workloads

Key considerations for right sizing

- **Minimum core requirement:** OpenShift Virtualization requires a minimum of eight (8) core Azure VMs for OpenShift worker nodes.
- **Architectural overhead:** Performance might vary depending on the architectural decisions taken while configuring the environment, including instance types, storage, and network characteristics.
- **Scaling out:** For demanding workloads, scaling out your Azure Red Hat OpenShift cluster by adding more nodes can help overcome resource contention and maintain throughput.
- **Benchmark your workloads:** Avoid relying solely on on-premises sizing references; benchmark your own workloads to inform right sizing.
- **Cost factors:** Consider Azure compute costs, OpenShift licensing, VM licensing, and scalability requirements.

Right sizing ensures that your VMs are provisioned with adequate resources to meet performance goals without overprovisioning. This process is critical in cloud environments, where resource efficiency directly impacts cost and performance.

Steps to right size workloads

1. **Define health metrics**

    - CPU Utilization: Target 60–70% average usage.
    - Memory Pressure: Monitor swap usage, memory saturation, and page faults.
    - IO Strain: Measure disk latency, throughput, and queue depth.

2. **Set up monitoring**

    - Use Prometheus and Grafana for real-time metric collection and visualization.
    - Enable KubeVirt metrics for VM-level insights.
    - Correlate infrastructure-level metrics with application performance by integrating with Azure Monitor, via Azure Arc.

3. **Analyze historical data**

    - Review performance trends over time.
    - Identify peak usage periods and resource saturation events.

- Use historical baselines to guide future autoscaling decisions.

4. **Adjust VM specifications**

   - Choose appropriate VM sizes from Dsv5 or Dsv6 series based on observed metrics.
   - Consider CPU, memory, disk IOPS, and network throughput needs.
   - Avoid overprovisioning by aligning resources with actual usage patterns.
   - Review the Red Hat guidance on right sizing - Announcing right-sizing for OpenShift Virtualization | Red Hat Developer.

5. **Test and validate**

   - Perform load testing using tools like Apache JMeter, Locust, or stress-ng.
   - Validate against defined health metrics and performance targets.
   - Iterate on configuration changes and retest to confirm improvements.

Observed performance characteristics

Below you can find observed performance metrics for OpenShift Virtualization workloads on Azure Red Hat OpenShift for specified configurations and workloads. Actual observed performance may vary depending on the workload chosen and specific cluster configuration, but the below tables will give you an idea of what can be expected. Due to the nature of running Virtual Machines in Azure Red Hat OpenShift, you will find that it will not be as performant as running the workloads in a pod. Performance will be improved in the near future with improved instance types becoming available and new technologies such as Direct Virtualization.

Compute

- Worker node type: Standard_D96ds_v5 (with Azure Boost)
- OpenShift version: 4.20
- Virtualization operator: 4.20

Expand table

	OCP Virtualization VM	Pod	With Direct Virtualization
Events	525,022	546,997	Coming soon
Latency (ms)	0.70	0.65	Coming soon

Storage

- Disk: Premium SSD v2 SCSI/SATA Disks
- Worker node type: Standard_D96ds_v5 (with Azure Boost)
- OpenShift version: 4.20
- Virtualization operator: 4.20
- ODF storage operator: 4.19
- Single AZ cluster

Expand table

Threads	OCP Virtualization VM (TPM)	Pod (TPM)	With Direct Virtualization
1	4,332	6,303	Coming soon
2	9,266	12,371	Coming soon
4	17,006	23,422	Coming soon
8	31,148	43,314	Coming soon
16	44,904	68,872	Coming soon
32	64,294	103,359	Coming soon

Network

- Worker node type: Standard_D96ds_v5 (with Azure Boost)
- NIC: 35 GB
- Single AZ Cluster

Expand table

Message size - threads	OCP Virtualization VM Latency (µs)	OCP Virtualization VM Throughput (Gbps)	Pod Latency (µs)	Pod Throughput (Gbps)	With Direct Virtualization
64B - 1 thr	94.58	0.4	45.84	0.9	Coming soon
64B - 8 thr	87.93	3.4	49.90	7.5	Coming soon
1024B - 1 thr	90.6	6.1	48.32	7.0	Coming soon
1024B - 8 thr	93.57	24.7	48.59	28.9	Coming soon
8192B - 1thr	151.4	7.6	104.43	10.9	Coming soon

Message size - threads	OCP Virtualization VM Latency (µs)	OCP Virtualization VM Throughput (Gbps)	Pod Latency (µs)	Pod Throughput (Gbps)	With Direct Virtualization
8192B - 8 thr	157.27	20.7	90.96	27.0	Coming soon

Fine tuning your environment

Fine tuning your OpenShift Virtualization environment is essential to achieving optimal performance, especially for demanding workloads. The following best practices are derived from extensive benchmarking and real-world experience on Azure Boost VM series (Dsv5/Dsv6).

Performance optimization strategies

- **Scale out or up for demanding workloads:** Add more nodes or upsize the nodes in your Azure Red Hat OpenShift cluster for high concurrency or resource-intensive applications.
- **Avoid strict resource limits:** Set only guest memory for VMs; avoid strict resource limits unless required for governance.
- **Tune storage and network configurations:** Select storage solutions and performance tiers that match your workload needs. For network-intensive workloads, tune settings such as NAPI and multiqueue, and monitor throughput and latency.
- **Monitor and benchmark regularly:** Use Prometheus, Grafana, and Azure Monitor to track key metrics. Benchmark your own workloads to validate performance and guide further tuning.
- **Expect architectural overhead:** Plan capacity and set expectations accordingly, especially for workloads with high I/O or network demands.

VM overcommit tuning

OpenShift Virtualization operator allows you to adjust CPU and memory overcommit ratios, letting you allocate more virtual resources than physically available. This change can improve density and resource utilization but might increase contention and affect performance.

Best practices for overcommit tuning:

- Use conservative overcommit for production workloads.
- Consider higher overcommit for dev/test environments.
- Monitor resource usage and adjust ratios as needed.

For more information, see Configuring higher VM workload density

Best practices based on benchmarking

- **Database workloads:** Avoid setting both resource requests and limits for VMs. Monitor performance closely when using fast storage and high concurrency. Scale out cluster nodes for large database deployments.
- **Network workloads:** Tune network settings for optimal throughput. Scale out as needed to achieve the desired network throughput.

Storage solution tuning

- **OpenShift Data Foundation (ODF):** Use SSD-backed storage for low-latency access. Configure replication and erasure coding policies based on workload needs. To prevent competition for your application compute resources, consider creating a separate worker pool for ODF with smaller Azure VM sizes, Ds16v5 is a good starting point, and use taints/tolerations to ensure ODF is the only workload scheduled there. Monitor storage performance and adjust replication factors as needed.
- **Azure NetApp Files (ANF):** Choose performance tiers based on IOPS and throughput requirements. Ensure proper mount options and network configuration for optimal performance. Use volume snapshots and backups to support data protection and recovery strategies.

Related content

OpenShift Virtualization for Azure Red Hat OpenShift.

Last updated on 02/16/2026

Create an Azure Red Hat OpenShift 4 private cluster

07/18/2025

In this article, you prepare your environment to create Azure Red Hat OpenShift private clusters running OpenShift 4. You learn how to:

- ✔ Setup the prerequisites and create the required virtual network and subnets
- ✔ Deploy a cluster with a private API server endpoint and a private ingress controller

If you choose to install and use the CLI locally, this tutorial requires that you're running the Azure CLI version 2.30.0 or later. To find the version, run the `az --version` command. If you need to install or upgrade, see Install Azure CLI.

Before you begin

Register the resource providers

1. If you have multiple Azure subscriptions, specify the relevant subscription ID:

   Azure CLI

   ```
   az account set --subscription <SUBSCRIPTION ID>
   ```

2. Register the `Microsoft.RedHatOpenShift` resource provider:

   Azure CLI

   ```
   az provider register -n Microsoft.RedHatOpenShift --wait
   ```

3. Register the `Microsoft.Compute` resource provider (if you haven't already):

   Azure CLI

   ```
   az provider register -n Microsoft.Compute --wait
   ```

4. Register the `Microsoft.Network` resource provider (if you haven't already):

   Azure CLI

```
az provider register -n Microsoft.Network --wait
```

5. Register the `Microsoft.Storage` resource provider (if you haven't already):

Azure CLI

```
az provider register -n Microsoft.Storage --wait
```

Get a Red Hat pull secret (optional)

A Red Hat pull secret enables your cluster to access Red Hat container registries along with other content. This step is optional but recommended.

1. Go to your Red Hat OpenShift cluster manager portal and sign in.

   Sign in to your Red Hat account or create a new Red Hat account with your business email and accept the terms and conditions.

2. Click Download pull secret.

Keep the saved `pull-secret.txt` file somewhere safe - it's used in each cluster creation.

When running the `az aro create` command, you can reference your pull secret using the `--pull-secret @pull-secret.txt` parameter. Execute `az aro create` from the directory where you stored your `pull-secret.txt` file. Otherwise, replace `@pull-secret.txt` with `@<path-to-my-pull-secret-file>`.

If you're copying your pull secret or referencing it in other scripts, your pull secret should be formatted as a valid JSON string.

Create a virtual network containing two empty subnets

Next, create a virtual network containing two empty subnets.

1. Set the following variables.

Console

```
LOCATION=eastus                 # the location of your cluster
RESOURCEGROUP="v4-$LOCATION"    # the name of the resource group where you
want to create your cluster
CLUSTER=aro-cluster             # the name of your cluster
```

2. Create a resource group

   An Azure resource group is a logical group in which Azure resources are deployed and managed. When you create a resource group, you specify a location. This location is where resource group metadata is stored, it's also where your resources run in Azure if you don't specify another region during resource creation. Create a resource group using the az group create command.

   Azure CLI

```
az group create --name $RESOURCEGROUP --location $LOCATION
```

   The following example output shows the resource group created successfully:

   JSON

```
{
"id": "/subscriptions/<guid>/resourceGroups/aro-rg",
"location": "eastus",
"managedBy": null,
"name": "aro-rg",
"properties": {
    "provisioningState": "Succeeded"
},
"tags": null
}
```

3. Create a virtual network.

   Azure Red Hat OpenShift clusters running OpenShift 4 require a virtual network with two empty subnets, for the control and worker nodes.

   Create a new virtual network in the same resource group you created earlier.

   Azure CLI

```
az network vnet create \
--resource-group $RESOURCEGROUP \
--name aro-vnet \
--address-prefixes 10.0.0.0/22
```

   The following example output shows the virtual network created successfully:

   JSON

```
{
  "newVNet": {
```

```
    "addressSpace": {
      "addressPrefixes": [
        "10.0.0.0/22"
      ]
    },
    "id": "/subscriptions/<guid>/resourceGroups/aro-
rg/providers/Microsoft.Network/virtualNetworks/aro-vnet",
    "location": "eastus",
    "name": "aro-vnet",
    "provisioningState": "Succeeded",
    "resourceGroup": "aro-rg",
    "type": "Microsoft.Network/virtualNetworks"
  }
}
```

4. Add an empty subnet for the master nodes.

Azure CLI

```
az network vnet subnet create \
--resource-group $RESOURCEGROUP \
--vnet-name aro-vnet \
--name master-subnet \
--address-prefixes 10.0.0.0/23 \
--service-endpoints Microsoft.ContainerRegistry
```

5. Add an empty subnet for the worker nodes.

Azure CLI

```
az network vnet subnet create \
--resource-group $RESOURCEGROUP \
--vnet-name aro-vnet \
--name worker-subnet \
--address-prefixes 10.0.2.0/23 \
--service-endpoints Microsoft.ContainerRegistry
```

6. Disable subnet private endpoint policies on the master subnet. This is required to be able to connect and manage the cluster.

Azure CLI

```
az network vnet subnet update \
--name master-subnet \
--resource-group $RESOURCEGROUP \
--vnet-name aro-vnet \
--private-link-service-network-policies Disabled
```

Create the cluster

Run the following command to create a cluster. Optionally, you can pass your Red Hat pull secret which enables your cluster to access Red Hat container registries along with other content.

> **Note**
>
> If you're copy/pasting commands and using one of the optional parameters, be sure delete the initial hashtags and the trailing comment text. As well, close the argument on the preceding line of the command with a trailing backslash.

Azure CLI

```
az aro create \
  --resource-group $RESOURCEGROUP \
  --name $CLUSTER \
  --vnet aro-vnet \
  --master-subnet master-subnet \
  --worker-subnet worker-subnet \
  --apiserver-visibility Private \
  --ingress-visibility Private
  # --domain foo.example.com # [OPTIONAL] custom domain
  # --pull-secret @pull-secret.txt # [OPTIONAL]
```

The `az aro create` command normally takes about 35 minutes to create a cluster.

> **Note**
>
> When attempting to create a cluster, if you receive an error message that your resource quota was exceeded, see Adding Quota to account to learn how to proceed.

> **Important**
>
> If you choose to specify a custom domain, for example `foo.example.com`, the OpenShift console is available at a URL such as `https://console-openshift-console.apps.foo.example.com`, instead of the built-in domain `https://console-openshift-console.apps.<random>.<location>.aroapp.io`.
>
> By default OpenShift uses self-signed certificates for all of the routes created on `*.apps.<random>.<location>.aroapp.io`. If you choose Custom DNS, after connecting to the

cluster, you'll need to follow the OpenShift documentation to configure a custom certificate for your ingress controller and custom certificate for your API server.

Create a private cluster without a public IP address

Typically, private clusters are created with a public IP address and load balancer, providing a means for outbound connectivity to other services. However, you can create a private cluster without a public IP address. This might be required in situations in which security or policy requirements prohibit the use of public IP addresses.

To create a private cluster without a public IP address, follow the procedure above, adding the parameter `--outbound-type UserDefinedRouting` to the `aro create` command, as in the following example:

```
az aro create \
  --resource-group $RESOURCEGROUP \
  --name $CLUSTER \
  --vnet aro-vnet \
  --master-subnet master-subnet \
  --worker-subnet worker-subnet \
  --apiserver-visibility Private \
  --ingress-visibility Private \
  --outbound-type UserDefinedRouting
```

> **Note**
>
> The UserDefinedRouting flag can only be used when creating clusters with `--apiserver-visibility Private` and `--ingress-visibility Private` parameters. Ensure you're using the latest Azure CLI. Clusters deployed with Azure CLI 2.52.0 and older will get deployed with public IPs.

This User Defined Routing option prevents a public IP address from being provisioned. User Defined Routing (UDR) allows you to create custom routes in Azure to override the default system routes or to add more routes to a subnet's route table. See Virtual network traffic routing to learn more.

> **Important**

Be sure to specify the correct subnet with the properly configured routing table when creating your private cluster.

For egress, the User Defined Routing option ensures that the newly created cluster has the egress lockdown feature enabled to allow you to secure outbound traffic from your new private cluster. See Control egress traffic for your Azure Red Hat OpenShift cluster to learn more.

> **Note**
>
> If you choose the User Defined Routing network type, you're completely responsible for managing the egress of your cluster's routing outside of your virtual network (for example, getting access to public internet). Azure Red Hat OpenShift can't manage this for you.

You can configure one or more egress IP addresses to a namespace or to specific pods in a namespace of a private cluster with no public IP address. To do so, follow the procedure above to create a private cluster without a public IP address, and then configure the egress IP as per this Red Hat OpenShift document. These egress IP addresses need to be from the subnets associated with the cluster.

Configuring an egress IP for a private cluster is only supported for clusters with the `--outbound-type UserDefinedRouting` parameter. It isn't supported for public clusters that have the `--outbound-type LoadBalancer` parameter.

Connect to the private cluster

You can log into the cluster using the `kubeadmin` user. Run the following command to find the password for the `kubeadmin` user.

Azure CLI

```
az aro list-credentials \
  --name $CLUSTER \
  --resource-group $RESOURCEGROUP
```

The following example output shows the password in `kubeadminPassword`.

JSON

```
{
  "kubeadminPassword": "<generated password>",
```

```
  "kubeadminUsername": "kubeadmin"
}
```

You can find the cluster console URL by running the following command, which looks like `https://console-openshift-console.apps.<random>.<region>.aroapp.io/`

Azure CLI

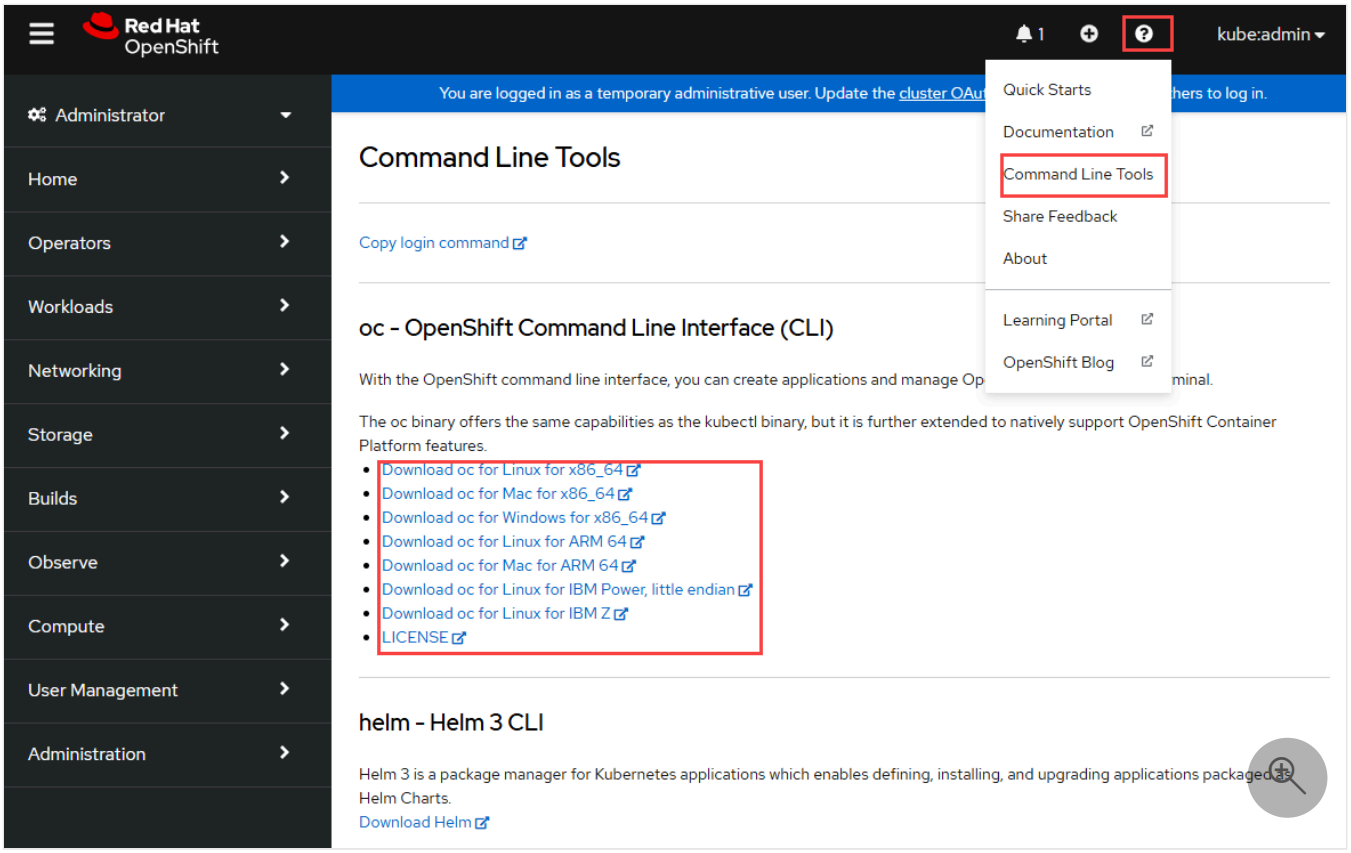
```
az aro show \
   --name $CLUSTER \
   --resource-group $RESOURCEGROUP \
   --query "consoleProfile.url" -o tsv
```

> **ⓘ Important**
>
> In order to connect to a private Azure Red Hat OpenShift cluster, you need to do the following step from a host that is either in the Virtual Network you created or in a Virtual Network that is peered with the Virtual Network the cluster was deployed to.

Launch the console URL in a browser and sign in using the `kubeadmin` credentials.

Install the OpenShift CLI

Once you're signed into the OpenShift Web Console, select the `?` at the top right and then on **Command Line Tools**. Download the release appropriate to your machine.



You can also download the latest release of the CLI appropriate to your machine.

Connect using the OpenShift CLI

Retrieve the API server's address.

Azure CLI

```
apiServer=$(az aro show --resource-group $RESOURCEGROUP --name $CLUSTER --query apiserverProfile.url --output tsv)
```

> **Important**
>
> In order to connect to a private Azure Red Hat OpenShift cluster, you'll need to perform the following step from a host that is either in the Virtual Network you created or in a Virtual Network that is peered with the Virtual Network the cluster was deployed to.

Sign in to the OpenShift cluster's API server using the following command. Replace `<kubeadmin password>` with the password you retrieved.

Azure CLI

```
oc login $apiServer --username kubeadmin --password <kubeadmin password>
```

Next steps

In this article, an Azure Red Hat OpenShift cluster running OpenShift 4 was deployed. You learned how to:

- ✓ Setup the prerequisites and create the required virtual network and subnets
- ✓ Deploy a cluster
- ✓ Connect to the cluster using the `kubeadmin` user

Advance to the next article to learn how to configure the cluster for authentication using Microsoft Entra ID.

- Configure authentication with Microsoft Entra ID using the command line
- Configure authentication with Microsoft Entra ID using the Azure portal and OpenShift web console

Configure multiple IP addresses per Azure Red Hat OpenShift cluster load balancer

Article • 02/25/2025

Azure Red Hat OpenShift public clusters are created with a public load balancer that's used for outbound connectivity from inside the cluster. By default, one public IP address is configured on that public load balancer, and that limits the maximum node count of your cluster to 62. To be able to scale your cluster to the maximum supported number of 250 nodes, you need to assign multiple additional public IP addresses to the load balancer.

You can configure up to 20 IP addresses per cluster. The outbound rules and frontend IP configurations are adjusted to accommodate the number of IP addresses.

> **⊗ Caution**
>
> Before deleting a cluster with more than 120 nodes, scale down the cluster to 120 nodes or less.

Requirements

The multiple public IPs feature is only available on the current network architecture used by ARO; older clusters don't support this feature. If your cluster was created before OpenShift Container Platform (OCP) version 4.5, this feature isn't available even if you upgraded your OCP version since then.

If you're unsure if your cluster was created before OCP version 4.5, use the following commands to check.

Get the cluster managed resource group:

```
RESOURCEGROUP=aro-rg   # the name of the resource group your cluster is in
CLUSTER=cluster        # the name of your cluster
CLUSTER_RESOURCEGROUP=$(az aro show -g $RESOURCEGROUP -n $CLUSTER --query
clusterProfile.resourceGroupId -o tsv | awk -F'/' '{print $NF}')
```

List the network load balancers:

```
az network lb list -g $CLUSTER_RESOURCEGROUP -o table
```

If you have a loadbalancer named `$CLUSTER-public-lb`, the cluster has the older network architecture and can't use the multiple public IP feature.

Create the cluster with multiple IP addresses

To create a new ARO cluster with multiple managed IPs on the public load balancer, use the following command with the desired number of IPs in the `--load-balancer-managed-outbound-ip-count` parameter. In the example below, seven (7) IP addresses are created:

```
az aro create \
  --resource-group aroResourceGroup \
  --name aroCluster \
  --load-balancer-managed-outbound-ip-count 7
```

See Deploy a large Azure Red Hat OpenShift cluster for more information about deploying a large cluster.

Update the number of IP addresses on existing clusters

To update the number of managed IPs on the public load balancer of an existing ARO cluster, use the following command with the desired number of IPs in the `--load-balancer-managed-outbound-ip-count` parameter. In the example below, the number of IPs for the cluster will be updated to four (4):

```
az aro update \
  --resource-group aroResourceGroup \
  --name aroCluster \
  --load-balancer-managed-outbound-ip-count 4
```

You can use this update method to either increase or decrease the number of IPs on a cluster to be between 1 and 20. Scaling down the number of clusters can interrupt the

outbound network traffic from the cluster.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Configure a custom DNS resolver for your Azure Red Hat OpenShift cluster

07/18/2025

This article provides the necessary details that allow you to configure your Azure Red Hat OpenShift cluster to use a custom DNS server. It contains the cluster requirements for a basic deployment.

Before you begin

This article assumes that you're creating a new cluster or have an existing cluster with latest updates applied. If you need a cluster, see the quickstart for a public cluster, or the private cluster tutorial for a private cluster. These steps to configure your cluster to use a custom DNS server are the same for both private and public clusters.

Confirm cluster compatibility with custom DNS

Confirm your cluster is eligible to support this feature by validating the existence of the `99-master-aro-dns` and `99-worker-aro-dns` `machineconfigs`.

```
oc get machineconfig
```

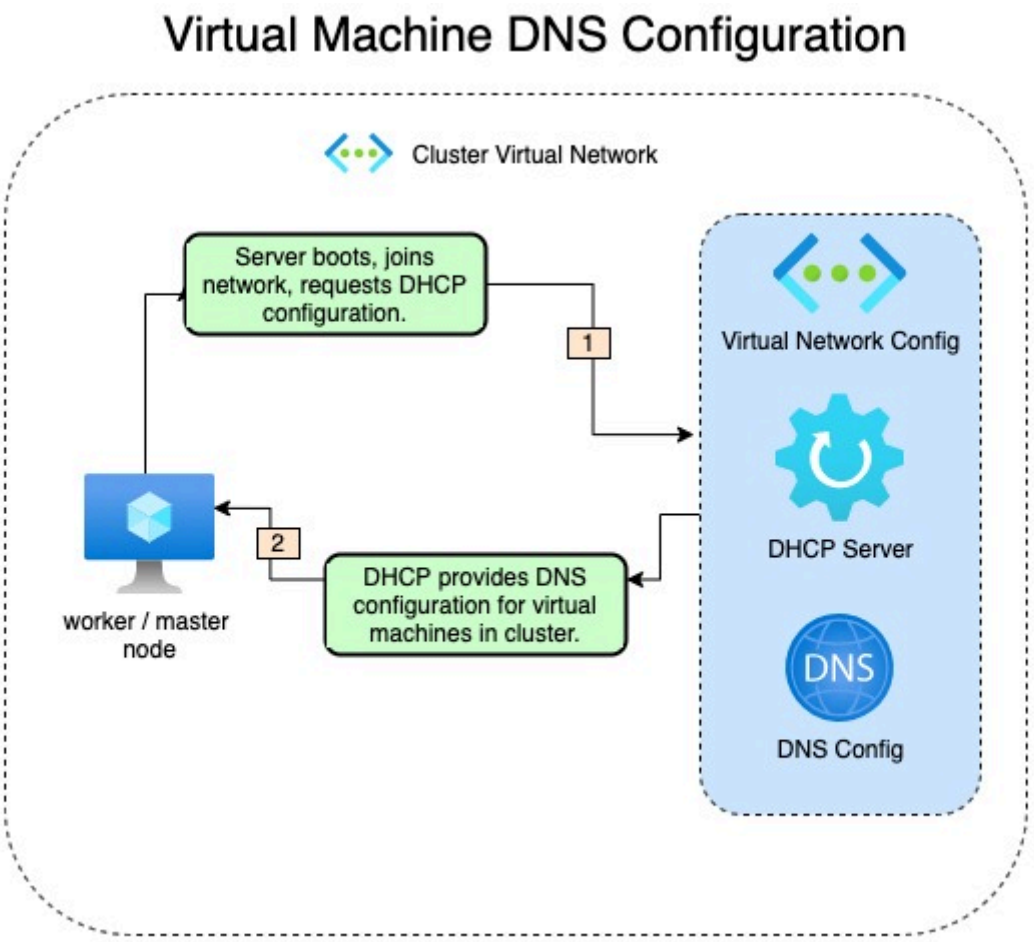
If the results of the above command include the following machineconfigs, your cluster is eligible for custom DNS support.

```
NAME                 GENERATEDBYCONTROLLER                      IGNITIONVERSION
AGE
...
99-master-aro-dns                                               2.2.0
54d
99-worker-aro-dns                                               2.2.0
54d
...
```

DNS architecture overview

As each node in the Azure Red Hat OpenShift cluster powers on and joins the network, DHCP configures the virtual machine with information such as IP address and which DNS server to use.

Below is the process flow overview of how the configuration is obtained:



An important trade-off of using your own DNS server instead of the default DNS server in the virtual network is that you lose the configuration that DNS server provided. The virtual machine names will no longer resolve through DNS on the network.

Update process overview

Configuring a custom dns server for the cluster is broken down into two steps.

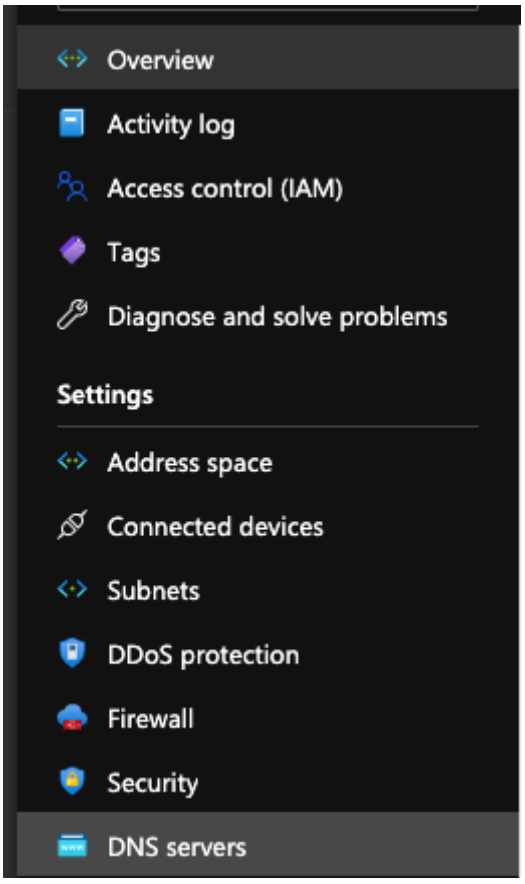
1. Modifying the Virtual Network DNS Servers configuration setting.
2. Restarting nodes in cluster to take changes.

Configure a custom DNS server

The following steps can be performed through the command line as well, but this documentation will be walking through using the portal web interface.

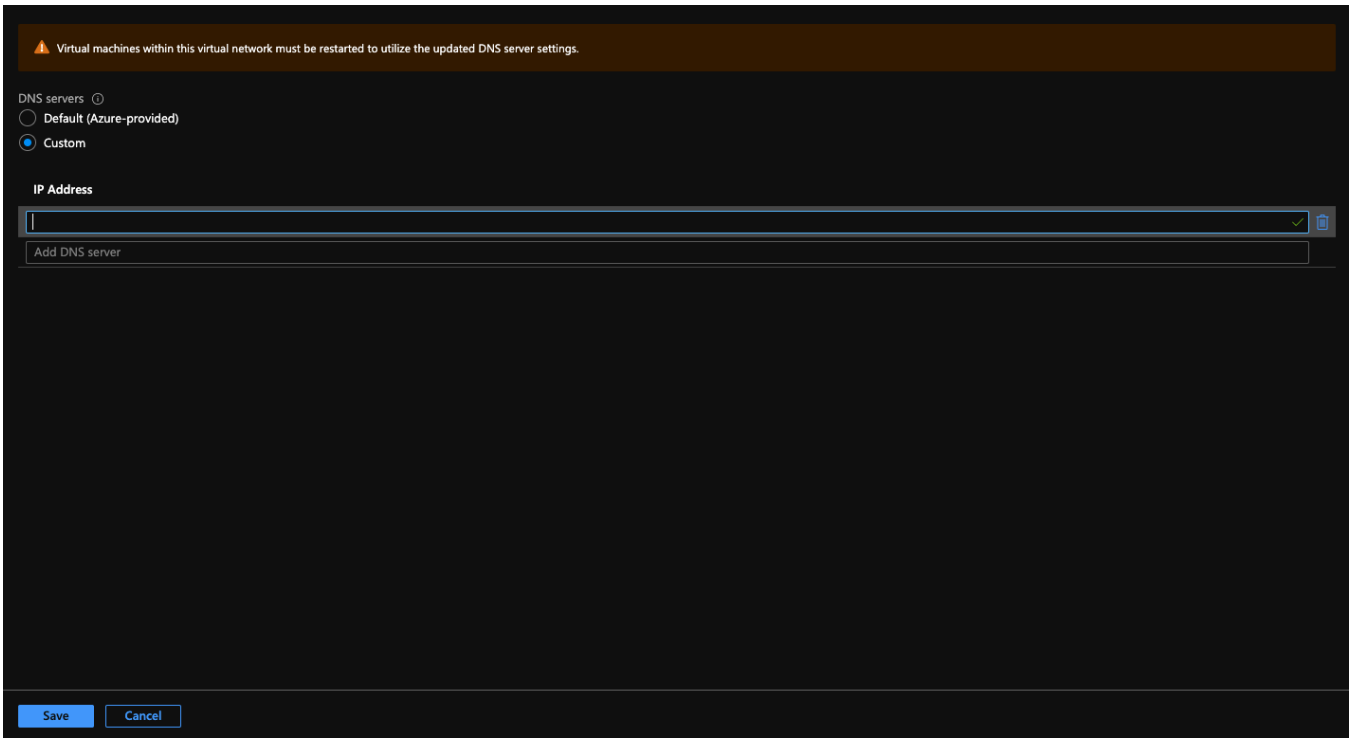
Update DNS configuration in virtual network

Log into the Azure portal and navigate to the desired virtual network you want to update. Select **DNS servers** from the virtual networks settings list.



Once you are at the DNS configuration screen, select **Custom** from the radial button configuration. Enter in the IP addresses for your DNS servers.

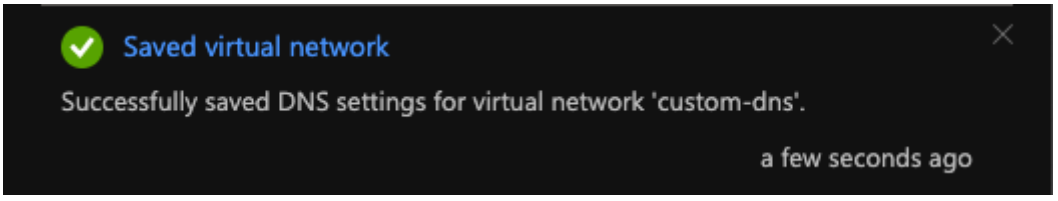
> **ⓘ Important**
>
> If you choose to specify a custom DNS server, you will no longer be able to resolve node names in the virtual network via DNS. Nodes will only be reachable via IP address.



Select **Save**.

> **Note**
>
> As shown in the portal interface, you must reboot all virtual machines for the changes to be in place.

You should receive a notification that your update was successful.



Gracefully reboot your cluster

These steps require having a valid kubeconfig to your cluster, see this tutorial for details on how to obtain a kubeconfig.

The following code snippets create noop `machineconfig`'s for master and worker nodes. This allows you to initiate rolling reboots for either the worker or master nodes. For more information about the Machine Config Operator (MCO), please see either the source code or the OpenShift docs for MCO.

MachineConfig definitions

Worker restarts:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 25-machineconfig-worker-reboot
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,cmVzdGFydAo=
        filesystem: root
        mode: 0644
        path: /etc/mco-noop-worker-restart.txt
```

Master restarts:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 25-machineconfig-master-reboot
spec:
  config:
    ignition:
      version: 2.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,cmVzdGFydAo=
        filesystem: root
        mode: 0644
        path: /etc/mco-master-noop-restart.txt
```

Reboot worker nodes

Create the worker restart file, this example calls the file `worker-restarts.yml`, and apply it.

```
```

```
[user@bastion ~]$ vim worker-restarts.yml
[user@bastion ~]$ oc apply -f worker-restarts.yml
machineconfig.machineconfiguration.openshift.io/25-machineconfig-worker-reboot
created
```

The MCO moves workloads and then reboots each node one at a time. Once the workers have come back online, we'll follow the same procedure to reboot the master nodes. You can verify the status of the workers by querying the nodes and validate they're all in the `Ready` state.

> ⓘ **Note**
>
> Depending on the size of the workload the cluster has, it can take several minutes for each node to reboot.

Example worker nodes not fully ready:

```
NAME                                STATUS                     ROLES    AGE
VERSION
dns-docs-tm45t-master-0             Ready                      master   5h40m
v1.19.0+a5a0987
dns-docs-tm45t-master-1             Ready                      master   5h40m
v1.19.0+a5a0987
dns-docs-tm45t-master-2             Ready                      master   5h40m
v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus1-8t6q8 Ready                      worker   5h35m
v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus2-ln2kq Ready,SchedulingDisabled   worker   5h34m
v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus3-gg75h Ready                      worker   5h35m
v1.19.0+a5a0987
```

As the node reboots, you'll see it change to the NotReady state:

```
dns-docs-tm45t-worker-eastus2-ln2kq   NotReady,SchedulingDisabled   worker   5h38m
v1.19.0+a5a0987
```

Fully ready:

```
[user@bastion ~]$ oc get nodes
NAME                                STATUS   ROLES    AGE     VERSION
```

```
dns-docs-tm45t-master-0               Ready    master   5h45m   v1.19.0+a5a0987
dns-docs-tm45t-master-1               Ready    master   5h46m   v1.19.0+a5a0987
dns-docs-tm45t-master-2               Ready    master   5h46m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus1-8t6q8   Ready    worker   5h41m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus2-ln2kq   Ready    worker   5h40m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus3-gg75h   Ready    worker   5h41m   v1.19.0+a5a0987
```

Reboot master nodes

Now, repeat the same process for the master nodes:

```
[user@bastion ~]$ vim master-restarts.yml
[user@bastion ~]$ oc apply -f master-restarts.yml
machineconfig.machineconfiguration.openshift.io/25-machineconfig-master-reboot
created
```

Confirm all nodes have returned to the `Ready` state:

```
[user@bastion ~]$ oc get nodes
NAME                                  STATUS   ROLES    AGE    VERSION
dns-docs-tm45t-master-0               Ready    master   6h8m   v1.19.0+a5a0987
dns-docs-tm45t-master-1               Ready    master   6h8m   v1.19.0+a5a0987
dns-docs-tm45t-master-2               Ready    master   6h8m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus1-8t6q8   Ready    worker   6h3m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus2-ln2kq   Ready    worker   6h2m   v1.19.0+a5a0987
dns-docs-tm45t-worker-eastus3-gg75h   Ready    worker   6h3m   v1.19.0+a5a0987
```

Confirm changes on a node (optional)

To validate the new DNS server on a node, we will use the `oc debug` pod.

```
[user@bastion ~]$ oc debug node/dns-docs-tm45t-worker-eastus2-ln2kq
Starting pod/dns-docs-tm45t-worker-eastus2-ln2kq-debug ...
To use host binaries, run `chroot /host`
chroot Pod IP: 10.0.2.6
If you don't see a command prompt, try pressing enter.
sh-4.4# chroot /host
sh-4.4# uptime
 18:40:16 up 1 min,  0 users,  load average: 0.82, 0.32, 0.12
sh-4.4# cat /etc/resolv.conf.dnsmasq
# Generated by NetworkManager
```

```
search reddog.microsoft.com
nameserver 192.168.0.1
```

Modifying the custom DNS server

The procedure for modifying the custom DNS on a cluster that already has custom DNS in place follows the same process.

Modify DNS

Follow the procedure outlined here to update the DNS configuration on the virtual network.

Reboot nodes

Instead of creating the `machineconfig`, we will instead delete the `machineconfig`s we created the first time. We'll start with the worker nodes.

```

oc delete machineconfig 25-machineconfig-worker-reboot
```

The output:

```

machineconfig.machineconfiguration.openshift.io "25-machineconfig-worker-reboot"
deleted
```

Wait for all of the worker nodes to reboot. This is similar to the reboot of worker nodes above.

Now we'll reboot the master nodes.

```

oc delete machineconfig 25-machineconfig-master-reboot
```

The output:

```

machineconfig.machineconfiguration.openshift.io "25-machineconfig-master-reboot"
deleted
```

Wait for all of the master nodes to reboot and return to a Ready state.

Configure DNS forwarding on an Azure Red Hat OpenShift 4 Cluster

Article • 02/25/2025

To configure DNS Forwarding on an Azure Red Hat OpenShift cluster, you'll need to modify the DNS operator. This modification allows the application pods running inside the cluster to resolve names hosted on a private DNS server outside the cluster. These steps are documented for OpenShift 4.6 here.

For example, if you want to forward all DNS requests for *.example.com to be resolved by a DNS server 192.168.100.10, edit the operator configuration by running:

Bash

```
oc edit dns.operator/default
```

This launches an editor sp you can replace `spec: {}` with:

YAML

```yaml
spec:
  servers:
  - forwardPlugin:
      upstreams:
      - 192.168.100.10
    name: example-dns
    zones:
    - example.com
```

Save the file and exit your editor.

Next steps

Check out more information on DNS forwarding for OpenShift 4.6 here.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Configure a cluster-wide proxy in an Azure Red Hat OpenShift (ARO) cluster

Article • 03/06/2025

This article describes the process for enabling a cluster-wide proxy on an Azure Red Hat OpenShift cluster. This feature allows production environments to deny direct access to the internet and instead have an HTTP or HTTPS proxy available. This article details the specific configuration steps necessary for an Azure Red Hat OpenShift cluster. For more information about how the cluster-wide proxy feature works for the OpenShift Container Platform, see the Red Hat documentation.

When configuring a cluster-wide proxy, it's important to understand the following impacts:

- **Node reboot:** Enabling the proxy causes nodes to reboot in a rolling fashion, similar to a cluster update. This is necessary as it applies new machine configurations.
- **Service disruptions:** To avoid any service disruptions during this process, it's crucial to prepare the `noProxy` list as described.

> **Important**
>
> Failure to adhere to the instructions outlined in this article might result in improper routing of cluster network traffic. This could lead to workload issues, such as image pull failures.

Scope of cluster-wide proxy configuration

- **OpenShift workloads:** The instructions in this article only apply to OpenShift workloads. Proxying application workloads is out of scope for this article.
- **OpenShift Container Platform versions:** Cluster-wide proxy is supported on OpenShift Container Platform versions outlined in the Azure Red Hat OpenShift support policy.

Following the instructions in this article and preparing the `noProxy` list will minimize disruptions and ensure a smooth transition when enabling the proxy.

Prerequisites and disclaimer

- Review the OpenShift documentation for Configuring the cluster-wide proxy for more information.
- Proxy server and certificates: You're expected to have a proxy server and certificates already in place.
- Azure Red Hat OpenShift SRE doesn't provide support for your proxy server or certificates.

Overview

1. Gather the required endpoint values for use in the `noProxy` list.
2. Enable the cluster-wide proxy using the gathered data for `noProxy`.
3. Verify that the `noProxy` list and the cluster-wide proxy were successfully configured.

Gather the required data for `noProxy`

1. Verify the cluster-wide proxy status by running the following command:

   ```
   oc get proxy cluster -o yaml
   ```

   The `spec` and `status` fields should be empty, showing it isn't enabled. If it isn't empty, then it may have been previously configured.

   YAML

   ```yaml
   apiVersion: config.openshift.io/v1
   kind: Proxy
   metadata:
     creationTimestamp: "xxxx-xx-xxTxx:xx:xxZ"
     generation:
     name: cluster
     resourceVersion:
     uid: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
   spec:
     trustedCA:
       name: ""
   status: {}
   ```

2. Note the IMDS IP: `169.254.169.254`

3. If you aren't using custom DNS, note the Azure DNS IP: `168.63.129.16`

4. Note the localhost and service domains:

   - `localhost`
   - `127.0.0.1`
   - `.svc`
   - `.cluster.local`

5. Retrieve the `gatewayDomains` by running the following command:

   ```
   oc get cluster cluster -o jsonpath='{.spec.gatewayDomains}'
   ```

   See the following example output:

   JSON

   ```
   [
       "agentimagestorews01.blob.core.windows.net",
       "agentimagestorecus01.blob.core.windows.net",
       "agentimagestoreeus01.blob.core.windows.net",
       "agentimagestoreeus01.blob.core.windows.net",
       "agentimagestoreeas01.blob.core.windows.net",
       "eastus-shared.prod.warm.ingest.monitor.core.windows.net",
       "...", // Many other endpoints
   ]
   ```

6. Get your cluster domain URLs.

   Create the cluster specific URLs for the API and application domains.

   a. Obtain the applications domain by running the following command:

   Azure CLI

   ```
   az aro show -n <CLUSTER_NAME> -g <RESOURCE_GROUP_NAME> --query
   "consoleProfile.url" -o tsv
   ```

   See the following example output:

   ```
   https://console-openshift-console.apps.xxxxxxxx.westus2.aroapp.io/
   ```

Only keep the part starting with `.apps.xxxxxxxx` for use in the `noProxy` list. Don't include the trailing "/".

See the following example:

```
.apps.xxxxxxxx.westus2.aroapp.io
```

b. Obtain the API domains.

Using the output of the previous command, replace `.apps` with `api` and `api-int` in the URL to get the API domains for the `noProxy` list.

See the following example:

```
api.xxxxxxxx.westus2.aroapp.io
api-int.xxxxxxxx.westus2.aroapp.io
```

7. Get the CIDR ranges.

a. Get the `addressPrefix` from the worker profile subnets by running the following command:

Azure CLI

```
SUBNET_ID=$(az aro show -n <CLUSTER_NAME> -g <RESOURCE_GROUP_NAME> --
query "workerProfiles[].subnetId" -o tsv)
az network vnet subnet show --ids "$SUBNET_ID" --query "addressPrefix
|| [].addressPrefix" -o tsv
```

Example output:

```
10.0.1.0/24
```

b. Get the `addressPrefix` from the master profile subnet by running the following command:

Azure CLI

```
SUBNET_ID=$(az aro show -n <CLUSTER_NAME> -g <RESOURCE_GROUP_NAME> --
query "masterProfile.subnetId" -o tsv)
az network vnet subnet show --ids "$SUBNET_ID" --query "addressPrefix"
-o tsv
```

Example output:

```
10.0.0.0/24
```

c. Get the `podCidr` by running the following command:

Azure CLI

```
az aro show -n <CLUSTER_NAME> -g <RESOURCE_GROUP_NAME> --query
"networkProfile.podCidr" -o tsv
```

Example output:

```
10.128.0.0/14
```

d. Get the `serviceCidr` by running the following command:

Azure CLI

```
az aro show -n <CLUSTER_NAME> -g <RESOURCE_GROUP_NAME> --query
"networkProfile.serviceCidr" -o tsv
```

Example output:

```
172.30.0.0/16
```

8. Combine the gathered data into your `noProxy` list which will be used for updating the proxy cluster object in the next section.

Enabling cluster-wide proxy

1. Create the `user-ca-bundle` configmap in the `openshift-config` namespace to use the correct certificate.

   a. Create a file called `user-ca-bundle.yaml` with the following contents, and provide the values of your PEM-encoded certificates:

   ```
   YAML
   ```

   ```yaml
   apiVersion: v1
   data:
     ca-bundle.crt: |
       <MY_PEM_ENCODED_CERTS>
   kind: ConfigMap
   metadata:
     name: user-ca-bundle
     namespace: openshift-config
   ```

   - `data.ca-bundle.crt`: This data key must be named **ca-bundle.crt**.
   - `data.ca-bundle.crt | <MY_PEM_ENCODED_CERTS>`: One or more PEM-encoded X.509 certificates used to sign the proxy's identity certificate.
   - `metadata.name`: The config map name referenced from the proxy object.
   - `metadata.namespace`: The config map must be in the `openshift-config` namespace.

   b. Create the ConfigMap by running the following command:

   ```
   oc create -f user-ca-bundle.yaml
   ```

   c. Confirm the creation of the `user-ca-bundle` ConfigMap by running the following command:

   ```
   oc get cm -n openshift-config user-ca-bundle -o yaml
   ```

   See the following example output:

   ```
   YAML
   ```

   ```yaml
   apiVersion: v1
   data:
     ca-bundle.crt: |
       -----BEGIN CERTIFICATE-----
   ```

```
    <CERTIFICATE_DATA>
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  creationTimestamp: "xxxx-xx-xxTxx:xx:xxZ"
  name: user-ca-bundle
  namespace: openshift-config
  resourceVersion: "xxxxxx"
  uid: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

2. Update the proxy cluster object using `oc edit`, then configure the proxy object using the previously gathered information.

   a. Run the following command:

```
oc edit proxy/cluster
```

   Update or add the following fields:

   - `spec.httpProxy`: A proxy URL to use for creating HTTP connections outside the cluster. The URL scheme must be `http`.
   - `spec.httpsProxy`: A proxy URL to use for creating HTTPS connections outside the cluster.
   - `spec.noProxy`: This will be the comma-separated list of endpoints obtained in the Gather the required data for noProxy steps above.
   - `spec.trustedCA`: A reference to the config map in the `openshift-config` namespace that contains other CA certificates required for proxying HTTPS connections. Note that the config map must already exist before referencing it here. In this case this is the name of the config map created above, which is **user-ca-bundle**.

   b. Confirm the configuration by running the following command:

```
oc get proxy cluster -o yaml
```

   See the following example output:

```
YAML
apiVersion: config.openshift.io/v1
kind: Proxy
```

```
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"config.openshift.io/v1","kind":"Proxy","metadata":
{"annotations":{},"name":"cluster"},"spec":
{"httpProxy":"http://10.0.0.15:3128","httpsProxy":"https://10.0.0.15:31
29","noProxy":"agentimagestorecus01.blob.core.windows.net,agentimagesto
reeus01.blob.core.windows.net,agentimagestorewus01.blob.core.windows.ne
t,agentimagestoreweu01.blob.core.windows.net,agentimagestoreeas01.blob.
core.windows.net,australiaeast-
shared.prod.warm.ingest.monitor.core.windows.net,gcs.prod.monitoring.co
re.windows.net,gsm1130809042eh.servicebus.windows.net,gsm1130809042xt.b
lob.core.windows.net,gsm119650579eh.servicebus.windows.net,gsm119650579
xt.blob.core.windows.net,gsm810972145eh.servicebus.windows.net,gsm81097
2145xt.blob.core.windows.net,maupdateaccount.blob.core.windows.net,maup
dateaccount2.blob.core.windows.net,maupdateaccount3.blob.core.windows.n
et,maupdateaccount4.blob.core.windows.net,production.diagnostics.monito
ring.core.windows.net,qos.prod.warm.ingest.monitor.core.windows.net,log
in.microsoftonline.com,management.azure.com,arosvc.azurecr.io,arosvc.au
straliaeast.data.azurecr.io,imageregistryvmxx7.blob.core.windows.net,.c
luster.local,.svc,api-
int.vlsi41ah.australiaeast.aroapp.io,localhost,10.0.0.0/8","trustedCA":
{"name":"user-ca-bundle"}}}
  creationTimestamp: "xxxx-xx-xxTxx:xx:xxZ"
  generation: 17
  name: cluster
  resourceVersion: "xxxxxxx"
  uid: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
spec:
  httpProxy: http://10.0.0.15:3128
  httpsProxy: https://10.0.0.15:3129
  noProxy:
agentimagestorecus01.blob.core.windows.net,agentimagestoreeus01.blob.co
re.windows.net,agentimagestorewus01.blob.core.windows.net,agentimagesto
reweu01.blob.core.windows.net,agentimagestoreeas01.blob.core.windows.ne
t,australiaeast-
shared.prod.warm.ingest.monitor.core.windows.net,gcs.prod.monitoring.co
re.windows.net,gsm1130809042eh.servicebus.windows.net,gsm1130809042xt.b
lob.core.windows.net,gsm119650579eh.servicebus.windows.net,gsm119650579
xt.blob.core.windows.net,gsm810972145eh.servicebus.windows.net,gsm81097
2145xt.blob.core.windows.net,maupdateaccount.blob.core.windows.net,maup
dateaccount2.blob.core.windows.net,maupdateaccount3.blob.core.windows.n
et,maupdateaccount4.blob.core.windows.net,production.diagnostics.monito
ring.core.windows.net,qos.prod.warm.ingest.monitor.core.windows.net,log
in.microsoftonline.com,management.azure.com,arosvc.azurecr.io,arosvc.au
straliaeast.data.azurecr.io,imageregistryvmxx7.blob.core.windows.net,.c
luster.local,.svc,api-
int.vlsi41ah.australiaeast.aroapp.io,localhost,10.0.0.0/8
  trustedCA:
    name: user-ca-bundle
status:
  httpProxy: http://10.0.0.15:3128
  httpsProxy: https://10.0.0.15:3129
  noProxy:
.cluster.local,.svc,10.0.0.0/8,10.128.0.0/14,127.0.0.0/8,127.0.0.1,169.
```

```
254.169.254,172.30.0.0/16,agentimagestorecus01.blob.core.windows.net,ag
entimagestoreeas01.blob.core.windows.net,agentimagestoreeus01.blob.core
.windows.net,agentimagestoreweu01.blob.core.windows.net,agentimagestore
wus01.blob.core.windows.net,api-
int.vlsi41ah.australiaeast.aroapp.io,arosvc.australiaeast.data.azurecr.
io,arosvc.azurecr.io,australiaeast-
shared.prod.warm.ingest.monitor.core.windows.net,gcs.prod.monitoring.co
re.windows.net,gsm1130809042eh.servicebus.windows.net,gsm1130809042xt.b
lob.core.windows.net,gsm119650579eh.servicebus.windows.net,gsm119650579
xt.blob.core.windows.net,gsm810972145eh.servicebus.windows.net,gsm81097
2145xt.blob.core.windows.net,imageregistryvmxx7.blob.core.windows.net,l
ocalhost,login.microsoftonline.com,management.azure.com,maupdateaccount
.blob.core.windows.net,maupdateaccount2.blob.core.windows.net,maupdatea
ccount3.blob.core.windows.net,maupdateaccount4.blob.core.windows.net,pr
oduction.diagnostics.monitoring.core.windows.net,qos.prod.warm.ingest.m
onitor.core.windows.net
```

3. Wait for the new machine-config to be rolled out to all the nodes and for the cluster operators to report healthy.

   a. Confirm node health by running the following command:

```
oc get nodes
```

   See the following example output:

```
NAME                                        STATUS   ROLES    AGE
VERSION
mycluster-master-0                          Ready    master   10d
v1.xx.xx+xxxxxxx
mycluster-master-1                          Ready    master   10d
v1.xx.xx+xxxxxxx
mycluster-master-2                          Ready    master   10d
v1.xx.xx+xxxxxxx
mycluster-worker-australiaeast1-mvzqr       Ready    worker   10d
v1.xx.xx+xxxxxxx
mycluster-worker-australiaeast2-l9fgj       Ready    worker   10d
v1.xx.xx+xxxxxxx
mycluster-worker-australiaeast3-pz9rw       Ready    worker   10d
v1.xx.xx+xxxxxxx
```

   b. Confirm cluster operator health by running the following command:

```
oc get co
```

See the following example output:

```
NAME                                       VERSION        AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
aro                                        vxxxxxxxx      True        False
False      10d
authentication                             4.xx.xx        True        False
False      8m25s
cloud-controller-manager                   4.xx.xx        True        False
False      10d
cloud-credential                           4.xx.xx        True        False
False      10d
cluster-autoscaler                         4.xx.xx        True        False
False      10d
... (Many other components) ...
storage                                    4.xx.xx        True        False
False      10d
```

> **Note**
>
> If you require the `user-ca-bundle`, it's located in the following directory (but it's not required for this process):
>
> `/etc/pki/ca-trust/source/anchors/openshift-config-user-ca-bundle.crt`

Verify `noProxy` configuration

To verify your proxy configuration, check the health status of the cluster operators. If the `noProxy` field is misconfigured, multiple cluster operators might enter a `Degraded: True` state. This can result from various issues, including, but not limited to, `ImagePullBack` errors, invalid certificates, or general connectivity problems. Additionally, some operators might remain in a `Progressing: True` state due to similar underlying causes.

1. Check the status of the cluster operators by running the following command:

```
oc get co
```

2. Interpreting the output (healthy state): If the `noProxy` field is **correctly configured**, the output should resemble the following example:

```
NAME                                       VERSION        AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
aro                                        vxxxxxxxx.xx   True
False         False      15d
authentication                             4.xx.xx        True
False         False      15d
cloud-controller-manager                   4.xx.xx        True
False         False      15d
cloud-credential                           4.xx.xx        True
False         False      15d
```

> **Note**
>
> The number and type of cluster operators may vary. The truncated example shown is provided to illustrate a healthy state for ARO-supported operators.

3. Interpreting the output (misconfigured): If the `noProxy` field is **misconfigured**, the output might resemble the following example:

```
NAME                           VERSION        AVAILABLE  PROGRESSING
DEGRADED  SINCE    MESSAGE
aro                            vxxxxxxxx.xx   True       False
False     45h
authentication                 4.xx.xx        False      True
True      24h
OAuthServerRouteEndpointAccessibleControllerAvailable: Get
"https://oauth-
openshift.apps.mm6osebam6b03b9df3.eastus2euap.aroapp.io/healthz": Not
Found
control-plane-machine-set      4.xx.xx        True       False
False     46h      SyncLoopRefreshProgressing: Working toward version
4.15.35, 1 replicas available
image-registry                 4.xx.xx        True       True
False     45h      NodeCADaemonProgressing: The daemon set node-ca is
deployed Progressing: The deployment has not completed
ingress                        4.xx.xx        True       True
True      83m      The "default" ingress controller reports
Degraded=True: DegradedConditions: One or more other status conditions
indicate a degraded state: CanaryChecksSucceeding=False
(CanaryChecksRepetitiveFailures: Canary route checks for the default
ingress controller are failing)
machine-config                 4.xx.xx        False      False
```

```
True      43h      Cluster not available for [{operator 4.15.35}]:
error during waitForControllerConfigToBeCompleted: [context deadline
exceeded, controllerconfig is not completed: status for
ControllerConfig machine-config-controller is being reported for 6,
expecting it for 13]
storage                     4.xx.xx        True       True
False     45h      AzureFileCSIDriverOperatorCRProgressing:
AzureFileDriverControllerServiceControllerProgressing: Waiting for
Deployment to deploy pods AzureFileCSIDriverOperatorCRProgressing:
AzureFileDriverNodeServiceControllerProgressing: Waiting for DaemonSet
to deploy node pods
```

> **Note**
>
> Shown is only a truncated sample output. Other cluster operators may also report a `Degraded: True` state with different errors resulting from the misconfiguration of `noProxy`.

Remove cluster-wide proxy

For information about removing the cluster-wide proxy, see the Red Hat OpenShift documentation.

Feedback

Control egress traffic for your Azure Red Hat OpenShift cluster

08/22/2025

This article provides the necessary details that allow you to secure outbound traffic from your Azure Red Hat OpenShift cluster. With the release of the Egress Lockdown Feature, all of the required connections for a cluster are proxied through the service. There are more destinations that you might want to allow to use features such as Operator Hub or Red Hat telemetry.

> **Important**
>
> Don't attempt these instructions on older clusters if those clusters don't have the Egress Lockdown feature enabled. To enable the Egress Lockdown feature on older clusters, see Enable Egress Lockdown.

Endpoints proxied through the service

The following endpoints are proxied through the service, and don't need other firewall rules. The following list is only for informational purposes.

Expand table

Destination FQDN	Port	Use
`arosvc.azurecr.io`	**HTTPS:443**	Global container registry for required system images.
`arosvc.$REGION.data.azurecr.io`	**HTTPS:443**	Regional container registry for required system images.
`management.azure.com`	**HTTPS:443**	Used by the cluster to access Azure APIs.
`login.microsoftonline.com`	**HTTPS:443**	Used by the cluster for authentication to Azure.
Specific subdomains of `monitor.core.windows.net`	**HTTPS:443**	Used for Microsoft Geneva Monitoring so that the team can monitor the customer's cluster.
Specific subdomains of `monitoring.core.windows.net`	**HTTPS:443**	Used for Microsoft Geneva Monitoring so that the team can monitor the customer's cluster.
Specific subdomains of `blob.core.windows.net`	**HTTPS:443**	Used for Microsoft Geneva Monitoring so that the team can monitor the customer's cluster.

Destination FQDN	Port	Use
Specific subdomains of `servicebus.windows.net`	**HTTPS:443**	Used for Microsoft Geneva Monitoring so that the team can monitor the customer's cluster.
Specific subdomains of `table.core.windows.net`	**HTTPS:443**	Used for Microsoft Geneva Monitoring so that the team can monitor the customer's cluster.

List of optional endpoints

You can configure your firewall allowlist with other endpoints. For more information, see Configuring your firewall.

Other container registry endpoints

Expand table

Destination FQDN	Port	Use
`registry.redhat.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat.
`quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn01.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn02.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn03.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn04.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn05.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`cdn06.quay.io`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.
`access.redhat.com`	**HTTPS:443**	Used to provide container images and operators from Red Hat and third-parties.

Destination FQDN	Port	Use
`registry.access.redhat.com`	**HTTPS:443**	Used to provide third-party container images and certified operators.
`registry.connect.redhat.com`	**HTTPS:443**	Used to provide third-party container images and certified operators.

Red Hat Telemetry and Red Hat Insights

By default, clusters are opted-out of Red Hat Telemetry and Red Hat Insights. If you wish to opt in to Red Hat telemetry, allow the following endpoints and update your cluster's pull secret.

Expand table

Destination FQDN	Port	Use
`cert-api.access.redhat.com`	**HTTPS:443**	Used for Red Hat telemetry.
`api.access.redhat.com`	**HTTPS:443**	Used for Red Hat telemetry.
`infogw.api.openshift.com`	**HTTPS:443**	Used for Red Hat telemetry.
`console.redhat.com/api/ingress`	**HTTPS:443**	Used in the cluster for the insights operator that integrates with Red Hat Insights.

For more information on remote health monitoring and telemetry, see the Red Hat OpenShift Container Platform documentation.

Other OpenShift endpoints

Expand table

Destination FQDN	Port	Use
`api.openshift.com`	**HTTPS:443**	Used by the cluster to check if updates are available for the cluster. Alternatively, users can use the OpenShift Upgrade Graph tool to manually find an upgrade path.
`mirror.openshift.com`	**HTTPS:443**	Required to access mirrored installation content and images.
`*.apps.<cluster_domain>*`	**HTTPS:443**	When you allowlist domains, this endpoint is used in your corporate network to reach applications deployed in Azure Red Hat OpenShift, or to access the OpenShift console.

Integrations

Azure Monitor container insights

Clusters can be monitored using the Azure Monitor container insights extension. For more information, see enable monitoring.

Related content

Overview of Azure Red Hat OpenShift egress lockdown

Migrate from OpenShift SDN to OVN-Kubernetes

Article • 02/25/2025

OpenShift SDN, a component of Red Hat OpenShift Networking, is a network plugin that uses software-defined networking (SDN) to create a unified network for your cluster. This network allows communication between pods across the OpenShift Container Platform. OpenShift SDN manages this network by configuring an overlay network using Open vSwitch (OVS).

OpenShift SDN has been deprecated since version 4.14 and will no longer be supported starting with version 4.17. Therefore, if your cluster is using OpenShift SDN, you must migrate to OVN-Kubernetes before upgrading to any minor OpenShift version beyond 4.16.

Migrating to OVN-Kubernetes for Azure Red Hat OpenShift

If your Azure Red Hat OpenShift (ARO) cluster is using the OpenShift SDN network plugin, you must migrate to the OVN-Kubernetes plugin before updating to version 4.17.

OVN-Kuberentes has been the default network plugin starting with ARO version 4.11. If you installed your cluster with version 4.11 or later, you likely don't need to perform a migration.

OpenShift SDN remains supported on Azure Red Hat OpenShift through version 4.16. See the Azure Red Hat OpenShift release calendar for end-of-life dates.

1. To determine which network plugin your cluster currently uses, run the following command:

   ```
   oc get network.operator.openshift.io cluster -o
   jsonpath='{.spec.defaultNetwork.type}'
   ```

   If you see an output such as `OpenShiftSDN`, proceed to the next step because you'll need to migrate.

2. See Limited live migration to the OVN-Kubernetes network plugin overview for steps to perform the migration.

> **Important**
>
> Azure Red Hat OpenShift only supports the limited live migration process. Don't use the offline migration process.

Next steps

- Learn more about OVN-Kubernetes network provider.
- Learn more about the OVN-Kubernetes network plugin.

Feedback

Change the MTU for the cluster network

This document details the supported process for enabling Jumbo MTU (Maximum Transmission Unit) on Azure Red Hat OpenShift (ARO) clusters. Enabling Jumbo MTU in ARO is strictly limited to intra-cluster network traffic, specifically covering pod-to-pod, pod-to-service, and node-to-node communication that utilizes the OVN overlay network. It is important to note that this configuration does not impact outbound or external network traffic, which continues to adhere to standard Azure networking MTU limits.

Overview

Changing the MTU may be beneficial for workloads that generate large volumes of east–west traffic within the cluster. Common use cases include high-throughput data processing pipelines, distributed databases, large-scale logging and monitoring systems, AI/ML training workloads, and storage-intensive applications. These are the workloads where reducing packet fragmentation can significantly improve throughput and CPU efficiency. Customers should consider enabling Jumbo MTU when network performance within the cluster is a known bottleneck or when running workloads that benefit from larger packet sizes.

Azure Red Hat OpenShift supports increasing the cluster network MTU and machine-level NIC MTU when the underlying Azure VM hardware uses the Microsoft Azure Network Adapter (MANA) driver. While Azure exposes a maximum NIC MTU of 9,000 bytes, the maximum configurable MTU is 8900 to account for 100 bytes of OVN overlay overhead.

Prerequisites

Jumbo MTU can only be enabled when every node in the cluster, including all control-plane and worker nodes, is running on Azure Virtual Machines (VM) that support the Microsoft Azure Network Adapter (MANA) driver.

- An ARO cluster running OpenShift 4.19 or higher.
- NICs with Accelerated Networking enabled.
- All cluster nodes must use VMs that support the MANA driver. Supported types include the Dv6 series (for example, D4as_v6, D8s_v6, D16s_v6, etc.).

> **Note**
>
> If the cluster has a mix of VMs where some support MANA and some do not, the MTU migration must be delayed until all nodes support MANA. Otherwise, the cluster uses the

default MTU of 1500, and may lead to fragmentation or connectivity issues for traffic between nodes with different MTU capabilities.

Validate the cluster nodes have the MANA driver

Validate that the MANA driver is in use by running the following commands on all nodes in the cluster (control plane and worker):

1. Get the node names in your cluster using this command.

   Console

   ```
   oc get nodes -o name
   ```

2. For each node, check what driver is used by the `enP*` interface.

   As the interface name may vary, first get the interface name:

   Console

   ```
   oc debug node/<NODE_NAME> -- chroot /host ip link | grep enP
   ```

   You see an output like:

   Output

   ```
   3: enP30xxxxx: <BROADCAST,MULTICAST,...,UP,LOWER_UP> mtu 1500 qdisc mq master
   eth0 state UP mode DEFAULT group default qlen 1000
   altname enP30xxxxxxx
   ```

   Use the interface name returned (like "enP30xxxxx") in the following command:

   Console

   ```
   oc debug node/<NODE_NAME> -- /bin/sh -c 'chroot /host ethtool -i
   <INTERFACE_NAME> | grep driver'
   ```

   You should see an output that shows that the `MANA` driver is in use.

ⓘ Important

If the driver is `mlx4`, `mlx5`, `hv_netvsc`, or anything other than `MANA`, the node does not support changing the MTU to 9,000.

Change the Maximum Transmission Unit (MTU)

Before changing the MTU, ensure that the cluster and all operators are healthy. Also ensure that all machine config pools are in a stable, fully updated, and healthy state. Plan for rolling reboots, as MTU changes trigger multiple MachineConfigPool rollouts.

1. To begin the MTU migration, specify the migration configuration by entering the following command. The Machine Config Operator performs a rolling reboot of the nodes in the cluster in preparation for the MTU change.

   Console

   ```
   oc patch Network.operator.openshift.io cluster --type=merge --patch \
   '{"spec": { "migration": { "mtu": { "network": { "from": 1400, "to": 8900 },
   "machine": { "to": 9000 } } } } }'
   ```

2. Monitor the rollout status by running the following command.

   Console

   ```
   oc get machineconfigpool
   ```

   Wait for all MachineConfigPool groups (master and worker) to reach a stable state, indicated by the following status values: `UPDATED=true`, `UPDATING=false`, `DEGRADED=false`. This takes some time to complete. The amount of time also depends on the size of the cluster.

   It should look similar to the following output.

   Output

   ```
   NAME     CONFIG                   UPDATED   UPDATING   DEGRADED   MACHINECOUNT
   READYMACHINECOUNT   UPDATEDMACHINECOUNT   DEGRADEDMACHINECOUNT   AGE
   master   rendered-master-xxxxx    True      False      False      3
   3                   3                     0                      2d4h
   worker   rendered-worker-xxxxx    True      False      False      3
   3                   3                     0                      2d4h
   ```

3. Verify MachineConfig rollout and MTU migration injection.

   After initiating the MTU migration in Step 1, verify that the Machine Config Operator (MCO) has successfully rendered and applied the updated MachineConfig to all nodes. Confirm that each node is transitioned to the expected rendered MachineConfig and that the configuration state is stable by running:

Console

```
oc describe node | egrep "hostname|machineconfig"
```

An example of the expected output is shown:

Output

```
kubernetes.io/hostname=master-0
[...]
machineconfiguration.openshift.io/currentConfig: rendered-master-xxxx
machineconfiguration.openshift.io/desiredConfig: rendered-master-xxxx
[...]
machineconfiguration.openshift.io/state: Done
```

Ensure that the value of `machineconfiguration.openshift.io/state` is `Done` and that the value of the `machineconfiguration.openshift.io/currentConfig` field is equal to the value of the `machineconfiguration.openshift.io/desiredConfig` field.

4. Verify the presence of the MTU migration script.

   During MTU migration, the Cluster Network Operator (CNO) injects a temporary systemd unit into the rendered MachineConfig. This unit runs the `mtu-migration.sh` script, which safely orchestrates the MTU transition across nodes and prevents network disruption during rolling reboots.

   To validate, inspect the MachineConfig referenced in the previous step (for example, `rendered-master-xxxx` or `rendered-worker-xxxx`).

   Console

   ```
   oc get machineconfig <CONFIG_NAME> -o yaml | grep mtu-migration.sh
   ```

   Where `<CONFIG_NAME>` specifies the name of the machine config from the `machineconfiguration.openshift.io/currentConfig` field. The expected output should include the following entry: "ExecStart=/usr/local/bin/mtu-migration.sh".

   > ⓘ **Note**
   >
   > The migration script is present only in the rendered MachineConfig generated by the MCO, not in user-created MachineConfigs. Always verify the specific `rendered-*` MachineConfig shown on the node.

5. Apply the new hardware MTU value.

After verifying that all previous steps are successful, create the following two MachineConfig files and apply them to the cluster.

a. Create the master MachineConfig (99-master-mtu.yaml) file

YAML

```yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
labels:
    machineconfiguration.openshift.io/role: master
name: 99-master-mtu
spec:
config:
    ignition:
    version: 3.5.0
    storage:
    files:
        - contents:
            compression: ""
            source: data:,%5Bconnection%5D%0Amatch-device%3Dinterface-
name%3Aeth0%0Aethernet.mtu%3D9000
        mode: 420
        path: /etc/NetworkManager/conf.d/99-eth0-mtu.conf
```

b. Apply the MachineConfig by running the following command:

Console

```
oc create -f 99-master-mtu.yaml
```

c. Create the worker MachineConfig (99-worker-mtu.yaml) file.

YAML

```yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
labels:
    machineconfiguration.openshift.io/role: worker
name: 99-worker-mtu
spec:
config:
    ignition:
    version: 3.5.0
    storage:
    files:
        - contents:
            compression: ""
            source: data:,%5Bconnection%5D%0Amatch-device%3Dinterface-
```

```
name%3Aeth0%0Aethernet.mtu%3D9000
        mode: 420
        path: /etc/NetworkManager/conf.d/99-eth0-mtu.conf
```

d. Apply the MachineConfig by running the following command:

Console

```
oc create -f 99-worker-mtu.yaml
```

e. Monitor the rollout status by running the following command.

Console

```
oc get machineconfigpool
```

Wait for all MachineConfigPool groups (master and worker) to reach a stable state, indicated by the following status values: `UPDATED=true`, `UPDATING=false`, `DEGRADED=false`. It will take some time to complete and also depends on the size of the cluster.

6. Reverify MachineConfig rollout.

Verify that the Machine Config Operator (MCO) is successfully rendered and applies the updated MachineConfig to all nodes. Confirm that each node is transitioned to the expected rendered MachineConfig and that the configuration state is stable by running:

Console

```
oc describe node | egrep "hostname|machineconfig"
```

An example of the expected output is shown:

Output

```
kubernetes.io/hostname=master-0
[...]
machineconfiguration.openshift.io/currentConfig: rendered-master-xxxx
machineconfiguration.openshift.io/desiredConfig: rendered-master-xxxx
[...]
machineconfiguration.openshift.io/state: Done
```

Ensure that the value of `machineconfiguration.openshift.io/state` is `Done` and that the value of the `machineconfiguration.openshift.io/currentConfig` field is equal to the value of the `machineconfiguration.openshift.io/desiredConfig` field.

7. Finalize the MTU migration.

   Clear the migration specification and apply the final MTU value by running the following command:

   Console

   ```
   oc patch Network.operator.openshift.io cluster --type=merge --patch \
   '{"spec": { "migration": null, "defaultNetwork": { "ovnKubernetesConfig": {
   "mtu": 8900 } } }}'
   ```

   Monitor the final rollout by checking the MachineConfigPool status, which should show UPDATED=true, UPDATING=false, DEGRADED=false.

   Console

   ```
   oc get machineconfigpool
   ```

Verify the change is completed

1. To verify that the Jumbo MTU configuration is successfully applied across the ARO cluster, you should begin by checking the cluster-wide network MTU. This can be done by inspecting the network configuration, where the expected value should reflect "Cluster Network MTU: 8900", confirming that the OVN overlay network is updated.

   Console

   ```
   oc describe network.config cluster | grep "Cluster Network MTU"
   ```

2. Verify the MTU at the node level by examining the primary network interface within a debug session. The interface should report an MTU of 9,000. OpenShift configures the overlay MTU conservatively (8900) to accommodate platform and overlay encapsulation and avoid packet fragmentation.

   Console

   ```
   oc debug node/<NODE_NAME> -- chroot /host ip -d link show eth0
   ```

3. For end-to-end pod connectivity validation, use a payload size of 8,872 bytes. This value is calculated to ensure the resulting packet size is exactly 8900 bytes (8,872 payload + 8 ICMP header + 20 IP header). This should succeed without fragmentation, indicating that the Jumbo MTU is consistently applied throughout the data path.

   You may select any pod to ping. For example:

Console

```
oc get pods -n openshift-monitoring -o wide
```

We can see an output like:

Output

```
NAME                                                  READY   STATUS
RESTARTS   AGE     IP            NODE                                NOMINATED
NODE   READINESS GATES
...
metrics-server-xxxxxxxxxx-xxxxx                       1/1     Running   0
2d16h   10.129.2.13   myarocluster-worker-westus2-xxxxx   <none>
<none>
metrics-server-xxxxxxxxxx-xxxxx                       1/1     Running   0
2d16h   10.131.0.7    myarocluster-worker-westus2-xxxxx   <none>
<none>
monitoring-plugin-xxxxxxxxxx-xxxxx                    1/1     Running   0
2d16h   10.129.2.8    myarocluster-worker-westus2-xxxxx   <none>
<none>
monitoring-plugin-xxxxxxxxxx-xxxxx                    1/1     Running   0
2d16h   10.131.0.16   myarocluster-worker-westus2-xxxxx   <none>
<none>
node-exporter-xxxxx                                   2/2     Running   10
3d      10.0.0.10     myarocluster-master-0               <none>
<none>
....
```

We can select the `metrics-server` pod.

Send an ICMP packet with a payload size of 8,872 bytes using the following command.

Console

```
oc debug node/<NODE_NAME> -- chroot /host ping -M do -s 8,872 <POD_IP>
```

We should see a result like:

Output

```
PING 10.129.2.13 (10.129.2.13) 8872(8900) bytes of data.
8880 bytes from 10.129.2.13: icmp_seq=1 ttl=62 time=5.26 ms
8880 bytes from 10.129.2.13: icmp_seq=2 ttl=62 time=0.408 ms
8880 bytes from 10.129.2.13: icmp_seq=3 ttl=62 time=0.198 ms
```

Troubleshooting

1. MachineConfigPool is stuck in an UPDATING state

   If the MachineConfigPool becomes stuck in the UPDATING state during the MTU migration process, you can begin troubleshooting by reviewing the Machine Config Operator (MCO) logs. This is done by running the command `oc logs -n openshift-machine-config-operator deploy/machine-config-operator`, which provides insight into what may be preventing the rollout from completing. Common issues that lead to an MCO stall include incorrectly formatted or improperly indented YAML within the applied MachineConfig, conflicts caused by multiple MachineConfigs attempting to modify the same files or settings, or nodes failing to reboot automatically after receiving updated configurations. Reviewing and correcting these issues typically allows the MachineConfigPool to resume progress and complete the update successfully.

2. MTU not updated on node NIC

   If the MTU does not appear to be applied correctly at the node level, begin troubleshooting by first identifying the physical network interface on the node (for example, `eth0` or `enp*`), as interface names may vary across environments. Verify the NetworkManager configuration file that sets the interface MTU by reviewing `/etc/NetworkManager/conf.d/99-eth0-mtu.conf` to ensure the expected MTU value is present. Next, confirm the MTU applied to the physical interface by inspecting its link settings rather than an OVS bridge such as `br-ex`. If the physical interface still reports an MTU of 1500, further investigation is required. In such cases, verify that the NIC driver in use is `MANA`, ensure that the underlying Azure VM size supports Jumbo MTU capabilities, and check whether the MachineConfigPool is still progressing by confirming whether `UPDATING=true`, which may indicate that the rollout has not yet completed.

3. Missing MANA driver

   Each NIC should list MANA as the active driver. If it is missing, the node's VM instance type may not support the Microsoft Azure Network Adapter (MANA). In such cases, you must resize the node to a Dv6-series or another MANA-supported Azure VM size. After resizing, the node may need to be rebuilt for the change to take effect which is typically done by draining and deleting the node, allowing ARO to automatically recreate it with the correct VM configuration.

4. OVN pods showing MTU mismatch

   If OVN pods report an MTU mismatch, you can begin troubleshooting by reviewing the OVN node logs using the command `oc logs -n openshift-ovn-kubernetes ds/ovnkube-node --all-containers=true | grep mtu`. This helps identify any discrepancies between the MTU values applied at the OVN layer and those configured through the

MachineConfigs. If mismatches appear in the logs, ensure that the Cluster Network Operator (CNO) MTU settings align with the values defined in the MachineConfig files, as inconsistencies between these components can prevent the MTU migration from completing correctly.

5. Connectivity issues after migration

   Connectivity issues after the MTU migration typically indicate that MTU values were not fully propagated across the cluster. To diagnose this, first verify that each node reports an MTU of 9,000, reflecting the expected hardware-level MTU on Azure. Next, ensure that the OVN overlay network is configured with an MTU of 8900, which aligns with the cluster-wide network settings. It is also critical to confirm that the Azure NIC is using the MANA driver, as non-MANA drivers do not support Jumbo MTU. If these values appear correct yet connectivity problems persist, you can further validate the end-to-end path MTU by running `tracepath <destination>`, which helps identify where packet fragmentation or MTU drops may be occurring in the network path.

Next steps

- View OpenShift documentation for Changing the MTU for the cluster network.

Last updated on 02/17/2026

Encrypt OS disks with a customer-managed key on Azure Red Hat OpenShift

Article • 02/25/2025

By default, the OS disks of the virtual machines in an Azure Red Hat OpenShift cluster were encrypted with auto-generated keys managed by Microsoft Azure. For additional security, customers can encrypt the OS disks with self-managed keys when deploying an Azure Red Hat OpenShift cluster. This feature allows for more control by encrypting confidential data with customer-managed keys (CMK).

Clusters created with customer-managed keys have a default storage class enabled with their keys. Therefore, both OS disks and data disks are encrypted by these keys. The customer-managed keys are stored in Azure Key Vault.

For more information about using Azure Key Vault to create and maintain keys, see Server-side encryption of Azure Disk Storage in the Microsoft Azure documentation.

With host-based encryption, the data stored on the VM host of your Azure Red Hat OpenShift agent nodes' VMs is encrypted at rest and flows encrypted to the Storage service. Host-base encryption means the temp disks are encrypted at rest with platform-managed keys.

The cache of OS and data disks is encrypted at rest with either platform-managed keys or customer-managed keys, depending on the encryption type set on those disks. By default, when using Azure Red Hat OpenShift, OS and data disks are encrypted at rest with platform-managed keys, meaning that the caches for these disks are also by default encrypted at rest with platform-managed keys.

You can specify your own managed keys following the encryption steps below. The cache for these disks also will be encrypted using the key that you specify in this step.

Limitation

It's the responsibility of customers to maintain the Key Vault and Disk Encryption Set in Azure. Failure to maintain the keys will result in broken Azure Red Hat OpenShift clusters. The VMs will stop working and, as a result, the entire Azure Red Hat OpenShift cluster will stop functioning.

The Azure Red Hat OpenShift Engineering team can't access the keys. Therefore, they can't back up, replicate, or retrieve the keys.

For details about using Disk Encryption Sets to manage your encryption keys, see Server-side encryption of Azure Disk Storage in the Microsoft Azure documentation.

Prerequisites

- Verify your permissions. You must have either Contributor and User Access Administrator permissions or Owner permissions.
- If you have multiple Azure subscriptions, register the resource providers. For registration details, see Register the resource providers.
- You will need to have the EncryptionAtHost feature enabled on your subscription. You can enable it by running:

  Azure CLI

  ```
  az feature register --namespace Microsoft.Compute --name
  EncryptionAtHost
  ```

- You can check the current status of the feature by running:

  Azure CLI

  ```
  az feature show --namespace Microsoft.Compute --name EncryptionAtHost
  ```

Create a virtual network containing two empty subnets

Create a virtual network containing two empty subnets. If you have an existing virtual network that meets your needs, you can skip this step. To review the procedure of creating a virtual network, see Create a virtual network containing two empty subnets.

Create an Azure Key Vault instance

You must use an Azure Key Vault instance to store your keys. Create a new Key Vault with purge protection enabled. Then, create a new key within the Key Vault to store your own custom key.

1. Set more environment permissions:

```
export KEYVAULT_NAME=$USER-enckv
export KEYVAULT_KEY_NAME=$USER-key
export DISK_ENCRYPTION_SET_NAME=$USER-des
```

2. Create a Key Vault and a key in the Key Vault:

Azure CLI

```
az keyvault create -n $KEYVAULT_NAME \
                -g $RESOURCEGROUP \
                -l $LOCATION \
                --enable-purge-protection true

az keyvault key create --vault-name $KEYVAULT_NAME \
                       -n $KEYVAULT_KEY_NAME \
                       --protection software

KEYVAULT_ID=$(az keyvault show --name $KEYVAULT_NAME --query "[id]" -o
tsv)

KEYVAULT_KEY_URL=$(az keyvault key show --vault-name $KEYVAULT_NAME \
                                        --name $KEYVAULT_KEY_NAME \
                                        --query "[key.kid]" -o tsv)
```

Create an Azure Disk Encryption Set

The Azure Disk Encryption Set is used as the reference point for disks in Azure Red Hat OpenShift clusters. It's connected to the Azure Key Vault that you created in the previous step, and pulls the customer-managed keys from that location.

Azure CLI

```
az disk-encryption-set create -n $DISK_ENCRYPTION_SET_NAME \
                              -l $LOCATION \
                              -g $RESOURCEGROUP \
                              --source-vault $KEYVAULT_ID \
                              --key-url $KEYVAULT_KEY_URL

DES_ID=$(az disk-encryption-set show -n $DISK_ENCRYPTION_SET_NAME -g
$RESOURCEGROUP --query 'id' -o tsv)

DES_IDENTITY=$(az disk-encryption-set show -n $DISK_ENCRYPTION_SET_NAME \
                                          -g $RESOURCEGROUP \
                                          --query "[identity.principalId]"
```

```
\
                                                -o tsv)
```

Grant permissions for the Disk Encryption Set to access the Key Vault

Use the Disk Encryption Set that you created in the previous step, and grant permission for the Disk Encryption Set to access and use the Azure Key Vault.

Azure CLI

```
az keyvault set-policy -n $KEYVAULT_NAME \
                       -g $RESOURCEGROUP \
                       --object-id $DES_IDENTITY \
                       --key-permissions wrapkey unwrapkey get
```

Create an Azure Red Hat OpenShift cluster

Create an Azure Red Hat OpenShift cluster to use the customer-managed keys.

> **Note**
>
> Enabling CMK on *existing* ARO clusters is only possible for worker nodes, not master nodes. You can achieve this using machine-API through machineset CRs. See Enabling customer-managed encryption keys for a machine set and Modifying a compute machine set for more information.

Azure CLI

```
az aro create --resource-group $RESOURCEGROUP \
              --name $CLUSTER  \
              --vnet aro-vnet  \
              --master-subnet master-subnet \
              --worker-subnet worker-subnet \
              --disk-encryption-set $DES_ID
```

After you create the Azure Red Hat OpenShift cluster, all VMs are encrypted with the customer-managed encryption keys.

To verify that you configured the keys correctly, run the following commands:

1. Get the name of the cluster Resource Group where the cluster VMs, disks, and so on are located:

   Azure CLI

   ```
   CLUSTERRESOURCEGROUP=$(az aro show --resource-group $RESOURCEGROUP --name $CLUSTER --query 'clusterProfile.resourceGroupId' -o tsv | cut -d '/' -f 5)
   ```

2. Check that the disks have the correct Disk Encryption Set attached:

   Azure CLI

   ```
   az disk list -g $CLUSTERRESOURCEGROUP --query '[].encryption'
   ```

   The field `diskEncryptionSetId` in the output must point to the Disk Encryption Set that you specified while creating the Azure Red Hat OpenShift cluster.

Feedback

Create an Azure Files StorageClass on Azure Red Hat OpenShift 4

Article • 04/10/2025

In this article, you create a StorageClass for Azure Red Hat OpenShift 4 that dynamically provisions ReadWriteMany (RWX) storage using Azure Files. You learn how to:

- ✓ Setup the prerequisites and install the necessary tools
- ✓ Create an Azure Red Hat OpenShift 4 StorageClass with the Azure File provisioner

If you choose to install and use the CLI locally, this tutorial requires that you're running the Azure CLI version 2.6.0 or later. To find the version, run the `az --version` command. If you need to install or upgrade, see Install Azure CLI.

Before you begin

Deploy an Azure Red Hat OpenShift 4 cluster into your subscription see Create an Azure Red Hat OpenShift 4 cluster.

Set up Azure storage account

This step creates a resource group outside of the Azure Red Hat OpenShift (ARO) cluster's resource group. This resource group contains the Azure Files shares that created Azure Red Hat OpenShift's dynamic provisioner.

Azure CLI

```
AZURE_FILES_RESOURCE_GROUP=aro_azure_files
LOCATION=eastus

az group create -l $LOCATION -n $AZURE_FILES_RESOURCE_GROUP

AZURE_STORAGE_ACCOUNT_NAME=aroazurefilessa

az storage account create \
  --name $AZURE_STORAGE_ACCOUNT_NAME \
  --resource-group $AZURE_FILES_RESOURCE_GROUP \
  --kind StorageV2 \
  --sku Standard_LRS
```

Set permissions

Set resource group permissions

The ARO service principal requires `listKeys` permission on the new Azure storage account resource group. Assign the Contributor role.

Azure CLI

```
ARO_RESOURCE_GROUP=aro-rg
CLUSTER=cluster
ARO_SERVICE_PRINCIPAL_ID=$(az aro show -g $ARO_RESOURCE_GROUP -n $CLUSTER --query
servicePrincipalProfile.clientId -o tsv)

az role assignment create --role Contributor --scope
/subscriptions/mySubscriptionID/resourceGroups/$AZURE_FILES_RESOURCE_GROUP --
assignee $ARO_SERVICE_PRINCIPAL_ID
```

Set ARO cluster permissions

The OpenShift persistent volume binder service account requires the ability to read secrets. Create and assign an OpenShift cluster role.

Azure CLI

```
ARO_API_SERVER=$(az aro list --query "[?contains(name,'$CLUSTER')].
[apiserverProfile.url]" -o tsv)

oc login -u kubeadmin -p $(az aro list-credentials -g $ARO_RESOURCE_GROUP -n
$CLUSTER --query=kubeadminPassword -o tsv) $ARO_API_SERVER

oc create clusterrole azure-secret-reader \
  --verb=create,get \
  --resource=secrets

oc adm policy add-cluster-role-to-user azure-secret-reader
system:serviceaccount:kube-system:persistent-volume-binder
```

Create StorageClass with Azure Files provisioner

This step creates a StorageClass with an Azure Files provisioner. Within the StorageClass manifest, the details of the storage account are required so that the ARO cluster knows to look at a storage account outside of the current resource group.

During storage provisioning, a secret named by secretName is created for the mounting credentials. In a multi-tenancy context, the recommendation is to set the value for

`secretNamespace` explicitly otherwise the storage account credentials might be read by other users.

Bash

```
cat << EOF >> azure-storageclass-azure-file.yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azure-file
provisioner: file.csi.azure.com
mountOptions:
  - dir_mode=0777
  - file_mode=0777
  - uid=0
  - gid=0
  - mfsymlinks
  - cache=strict
  - actimeo=30
  - noperm
parameters:
  location: $LOCATION
  secretNamespace: kube-system
  skuName: Standard_LRS
  storageAccount: $AZURE_STORAGE_ACCOUNT_NAME
  resourceGroup: $AZURE_FILES_RESOURCE_GROUP
reclaimPolicy: Delete
volumeBindingMode: Immediate
EOF

oc create -f azure-storageclass-azure-file.yaml
```

Mount options for Azure Files will generally be dependent on the workload that you're deploying and the requirements of the application. Specifically for Azure files, there are other parameters that you should consider using.

Mandatory parameters:

- `mfsymlinks` to map symlinks to a form the client can use
- `noperm` to disable permission checks on the client side

Recommended parameters:

- `nossharesock` to disable reusing sockets if the client is already connected via an existing mount point
- `actimeo=30` (or higher) to increase the time the CIFS client caches file and directory attributes

- `nobrl` to disable sending byte range lock requests to the server and for applications which have challenges with posix locks

Change the default StorageClass (optional)

The default StorageClass on ARO is called managed-premium and uses the azure-disk provisioner. Change this setting by issuing patch commands against the StorageClass manifests.

Bash

```
oc patch storageclass managed-premium -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"false"}}}'

oc patch storageclass azure-file -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

Verify Azure File StorageClass (optional)

Create a new application and assign storage to it.

> ⓘ **Note**
>
> To use the `httpd-example` template, you must deploy your ARO cluster with the pull secret enabled. For more information, see Get a Red Hat pull secret.

Bash

```
oc new-project azfiletest
oc new-app httpd-example

#Wait for the pod to become Ready
curl $(oc get route httpd-example -n azfiletest -o jsonpath={.spec.host})

#If you have set the storage class by default, you can omit the --claim-class
parameter
oc set volume dc/httpd-example --add --name=v1 -t pvc --claim-size=1G -m /data --
claim-class='azure-file'

#Wait for the new deployment to rollout
export POD=$(oc get pods --field-selector=status.phase==Running -o jsonpath=
{.items[].metadata.name})
oc exec $POD -- bash -c "echo 'azure file storage' >> /data/test.txt"
```

```
oc exec $POD -- bash -c "cat /data/test.txt"
azure file storage
```

The *test.txt* file is visible via the Storage Explorer in the Azure portal.

Next steps

In this article, you created dynamic persistent storage using Microsoft Azure Files and Azure Red Hat OpenShift 4. You learned how to:

- ✓ Create a Storage Account
- ✓ Configure a StorageClass on an Azure Red Hat OpenShift 4 cluster using the Azure Files provisioner

Advance to the next article to learn about Azure Red Hat OpenShift 4 supported resources.

- Azure Red Hat OpenShift support policy

Configure the built-in container registry for Azure Red Hat OpenShift 4

08/07/2025

Azure Red Hat OpenShift provides an integrated container image registry that adds the ability to automatically provision new image repositories on demand. This registry provides users with a built-in location for their application builds to push the resulting images.

In this article, you'll configure the built-in container image registry for an Azure Red Hat OpenShift 4 cluster. You'll learn how to:

- ✓ Authorize an identity to access to the registry
- ✓ Access the built-in container image registry from inside the cluster
- ✓ Access the built-in container image registry from outside the cluster

Before you begin

This article assumes you have an existing cluster (see Create an Azure Red Hat OpenShift 4 cluster). If you would like to configure Microsoft Entra integration, make sure to create the cluster with the `--pull-secret` argument to `az aro create`.

> ⓘ Note
>
> Configuring Microsoft Entra authentication for your cluster is the easiest way to interact with the internal registry from outside the cluster.

Once you have your cluster, connect to the cluster by authenticating as the `kubeadmin` user.

Configure authentication to the registry

For any identity (a cluster user, Microsoft Entra user, or ServiceAccount) to access the internal registry, it must be granted permissions inside the cluster:

As `kubeadmin`, execute the following commands:

Bash

```
# Note: replace "<user>" with the identity you need to access the registry
oc policy add-role-to-user -n openshift-image-registry registry-viewer <user>
oc policy add-role-to-user -n openshift-image-registry registry-editor <user>
```

> ⓘ Note
>
> For cluster users and Microsoft Entra users - this will be the same name you use to authenticate into the cluster. For OpenShift ServiceAccounts, format the name as `system:serviceaccount:<project>:<name>`

Access the registry

Now that you've configured authentication for the registry, you can interact with it:

From inside the cluster

If you need to access the registry from inside the cluster (for example, you're running a CI/CD platform as Pods that will push/pull images to the registry), you can access the registry via its ClusterIP Service at the fully qualified domain name `image-registry.openshift-image-registry.svc.cluster.local:5000`, which is accessible to all Pods within the cluster.

From outside the cluster

If your workflows require you access the internal registry from outside the cluster (for example, you want to push/pull images from a developer's laptop, external CI/CD platform, and/or a different cluster), you will need to perform a few more steps:

As `kubeadmin`, execute the following commands to expose the built-in registry outside the cluster via a Route:

Bash

```
oc patch config.imageregistry.operator.openshift.io/cluster --patch='{"spec":
{"defaultRoute":true}}' --type=merge
oc patch config.imageregistry.operator.openshift.io/cluster --patch='[{"op":
"add", "path": "/spec/disableRedirect", "value": true}]' --type=json
```

You can then find the registry's externally routable fully qualified domain name:

As `kubeadmin`, execute:

Bash

```
oc get route -n openshift-image-registry default-route --template='{{ .spec.host
}}'
```

Next steps

Now that you've set up the built-in container image registry, you can get started by deploying an application on OpenShift. For Java applications, check out Deploy a Java application with Open Liberty/WebSphere Liberty on an Azure Red Hat OpenShift 4 cluster.

Use Azure Container Registry with Azure Red Hat OpenShift

07/18/2025

Azure Container Registry (ACR) is a managed container registry service that you can use to store private Docker container images with enterprise capabilities such as geo-replication. To access the ACR from a cluster, the cluster can authenticate with ACR by storing Docker sign in credentials in a Kubernetes secret. Likewise, a cluster can use an imagePullSecret in the pod spec to authenticate against the registry when pulling the image. In this article, you learn how to set up an Azure Container Registry with an Azure Red Hat OpenShift cluster to store and pull private Docker container images.

Prerequisites

This guide assumes that you have an existing Azure Container Registry. If you don't, use the Azure portal or Azure CLI instructions to create a container registry.

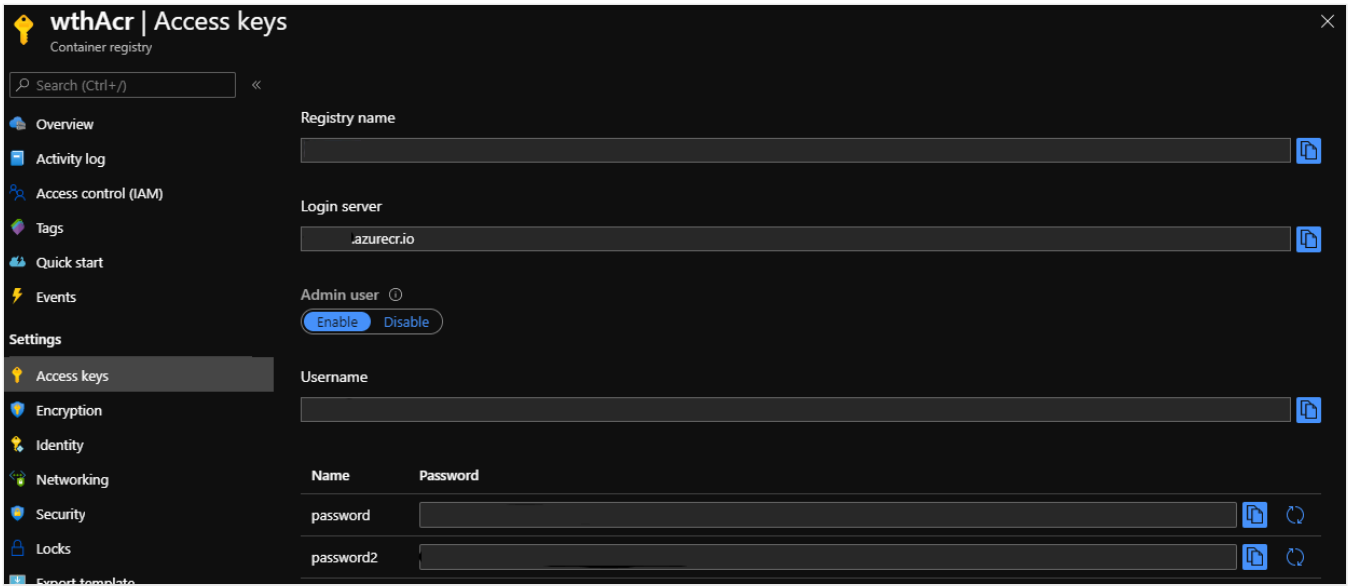
This article also assumes that you have an existing Azure Red Hat OpenShift cluster and have the `oc` CLI installed. If not, follow instructions in the Create cluster tutorial.

Get a pull secret

You need a pull secret from ACR to access the registry from your cluster.

To get your pull secret credentials, you can use either the Azure portal or the Azure CLI.

If using the Azure portal, navigate to your ACR instance, and select **Access Keys**. Your `docker-username` is the name of your container registry, use either password or password2 for `docker-password`.



Instead, you can use the Azure CLI to get these credentials:

Azure CLI

```
az acr credential show -n <your registry name>
```

Create the Kubernetes secret

Now, we use these credentials to create a Kubernetes secret. Execute the following command with your ACR credentials:

```
oc create secret docker-registry \
    --docker-server=<your registry name>.azurecr.io \
    --docker-username=<your registry name> \
    --docker-password=<password> \
    --docker-email=unused \
    acr-secret
```

> **Note**
>
> This secret is stored in the current OpenShift Project (Kubernetes Namespace) and will only be referenceable by pods created in that Project. See this document for further instructions on creating a cluster wide pull secret.

Link the secret to the service account

Next, link the secret to the service account that's used by the pod, so the pod can reach the container registry. The name of the service account should match the name of the service account used by the pod. `default` is the default service account:

```
oc secrets link default <pull_secret_name> --for=pull
```

Create a pod using a private registry image

Now that we connected your cluster to your ACR, let's pull an image from your ACR to create a pod.

Start with a podSpec and specify the secret you created as an imagePullSecret:

YAML

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-world
spec:
  containers:
  - name: hello-world
    image: <your registry name>.azurecr.io/hello-world:v1
  imagePullSecrets:
  - name: acr-secret
```

To test that your pod is up and running, execute this command and wait until the status is **Running**:

Bash

```
$ oc get pods --watch
NAME          READY   STATUS    RESTARTS   AGE
hello-world   1/1     Running   0          30s
```

Next steps

- Azure Container Registry
- Quickstart: Create a private container registry using the Azure CLI

Secure access to Azure Red Hat OpenShift with Azure Front Door

Article • 02/25/2025

This article explains how to use Azure Front Door Premium to secure access to Azure Red Hat OpenShift.

Prerequisites

The following prerequisites are required:

- You have an existing Azure Red Hat OpenShift cluster. Follow this guide to create a private Azure Red Hat OpenShift cluster.
- The cluster is configured with private ingress visibility.
- A custom domain name is used, for example:

  `example.com`

> **Note**
>
> The initial state doesn't have DNS configured. No applications are exposed externally from the Azure Red Hat OpenShift cluster.

Create an Azure Private Link service

This section explains how to create an Azure Private Link service. An Azure Private Link service is a reference to your own service that is powered by Azure Private Link.

Your service, which is running behind the Azure Standard Load Balancer, can be enabled for Private Link access so that consumers to your service can access it privately from their own VNets. Your customers can create a private endpoint inside their VNet and map it to this service.

For more information about the Azure Private Link service and how it's used, see Azure Private Link service.

Create an **AzurePrivateLinkSubnet**. This subnet includes a netmask that permits visibility of the subnet to the control plane and worker nodes of the Azure cluster. Don't delegate

this new subnet to any services or configure any service endpoints.

For example, if the virtual network is 10.10.0.0/16 and:

- Existing Azure Red Hat OpenShift control plane subnet = 10.10.0.0/24
- Existing Azure Red Hat OpenShift worker subnet = 10.10.1.0/24
- New AzurePrivateLinkSubnet = 10.10.2.0/24

Create a new Private Link at Azure Private Link service, as explained in the following steps:

1. On the **Basics** tab, configure the following options:

    - **Project Details**
        - Select your Azure subscription.
        - Select the resource group in which your Azure Red Hat OpenShift cluster was deployed.
    - **Instance Details**
        - Enter a **Name** for your Azure Private Link service, as in the following example: *example-com-private-link*.
        - Select a **Region** for your Private Link.

2. On the **Outbound Settings** tab:

    - Set the **Load Balancer** to the **-internal** load balancer of the cluster for which you're enabling external access. The choices are populated in the drop-down list.

    - Set the **Load Balancer frontend IP address** to the IP address of the Azure Red Hat OpenShift ingress controller, which typically ends in **.254**. If you're unsure, use the following command.

        Azure CLI

        ```
        az aro show -n <cluster-name> -g <resource-group> -o tsv --query ingressProfiles[].ip
        ```

    - The **Source NAT subnet** should be the **AzurePrivateLinkSubnet**, which you created.

    - No items should be changed in **Outbound Settings**.

3. On the **Access Security** tab, no changes are required.

- At the **Who can request access to your service?** prompt, select **Anyone with your alias**.
- Don't add any subscriptions for auto-approval.

4. On the **Tags** tab, select **Review + create**.

5. Select **Create** to create the Azure Private Link service, and then wait for the process to complete.

6. When your deployment is complete, select **Go to resource group** under **Next steps**.

In the Azure portal, enter the Azure Private Link service that was deployed. Retain the **Alias** that was generated for the Azure Private Link service. It will be used later.

Register domain in Azure DNS

This section explains how to register a domain in Azure DNS.

1. Create a global Azure DNS zone for example.com.

2. Create a global Azure DNS zone for apps.example.com.

3. Note the four nameservers that are present in Azure DNS for apps.example.com.

4. Create a new **NS** record set in the example.com zone that points to **apps** and specify the four nameservers that were present when the **apps** zone was created.

Create a new Azure Front Door Premium service

To create a new Azure Front Door Premium service:

1. On Microsoft Azure Compare offerings select **Azure Front Door**, and then select **Continue to create a Front Door**.

2. On the **Create a front door profile** page in the **Subscription** > **Resource group**, select the resource group in which your Azure Red Hat OpenShift cluster was deployed to house your Azure Front Door Premium resource.

3. Name your Azure Front Door Premium service appropriately. For example, in the **Name** field, enter the following name:

   `example-com-frontdoor`

4. Select the **Premium** tier. The Premium tier is the only choice that supports Azure Private Link.

5. For **Endpoint name**, choose an endpoint name that is appropriate for Azure Front Door.

   For each application deployed, a CNAME will be created in the Azure DNS to point to this hostname. Therefore, it's important to choose a name that is agnostic to applications. For security, the name shouldn't suggest the applications or architecture that you've deployed, such as **example01**.

   The name you choose will be prepended to the **.z01.azurefd.net** domain.

6. For **Origin type**, select **Custom**.

7. For **Origin Host Name**, enter the following placeholder:

   `changeme.com`

   This placeholder will be deleted later.

   At this stage, don't enable the Azure Private Link service, caching, or the Web Application Firewall (WAF) policy.

8. Select **Review + create** to create the Azure Front Door Premium resource, and then wait for the process to complete.

Initial configuration of Azure Front Door Premium

To configure Azure Front Door Premium:

1. In the Azure portal, enter the Azure Front Door Premium service that was deployed.

2. In the **Endpoint Manager** window, modify the endpoint by selecting **Edit endpoint**.

3. Delete the default route, which was created as **default-route**.

4. Close the **Endpoint Manager** window.

5. In the **Origin Groups** window, delete the default origin group that was named **default-origin-group**.

Exposing an application route in Azure Red Hat OpenShift

Azure Red Hat OpenShift must be configured to serve the application with the same hostname that Azure Front Door will be exposing externally (*.apps.example.com). In our example, we'll expose the Reservations application with the following hostname:

`reservations.apps.example.com`

Also, create a secure route in Azure Red Hat OpenShift that exposes the hostname.

Configure Azure DNS

To configure the Azure DNS:

1. Enter the public **apps** DNS zone previously created.
2. Create a new CNAME record set named **reservation**. This CNAME record set is an alias for our example Azure Front Door endpoint:

   `example01.z01.azurefd.net`

Configure Azure Front Door Premium

The following steps explain how to configure Azure Front Door Premium.

1. In the Azure portal, enter the Azure Front Door Premium service you created previously:

   `example-com-frontdoor`

In the Domains window:

1. Because all DNS servers are hosted on Azure, leave **DNS Management** set to **Azure managed DNS**.
2. Select the example domain:

   `apps.example.com`

3. Select the CNAME in our example:

   `reservations.apps.example.com`

4. Use the default values for **HTTPS** and **Minimum TLS version**.
5. Select **Add**.
6. When the **Validation stat** changes to **Pending**, select **Pending**.
7. To authenticate ownership of the DNS zone, for **DNS record status**, select **Add**.
8. Select **Close**.
9. Continue to select **Refresh** until the **Validation state** of the domain changes to **Approved** and the **Endpoint association** changes to **Unassociated**.

In the Origin Groups window:

1. Select **Add**.
2. Give your **Origin Group** an appropriate name, such as **Reservations-App**.
3. Select **Add an origin**.
4. Enter the name of the origin, such as **ARO-Cluster-1**.
5. Choose an **Origin type** of **Custom**.
6. Enter the fully qualified domain name (FQDN) hostname that was exposed in your Azure Red Hat OpenShift cluster, such as:

   `reservations.apps.example.com`
7. Enable the **Private Link** service.
8. Enter the **Alias** that was obtained from the Azure Private Link service.
9. Select **Add** to return to the origin group creation window.
10. Select **Add** to add the origin group and return to the Azure portal.

Grant approval in Azure Private Link

To grant approval to the **example-com-private-link**, which is the **Azure Private Link** service you created previously, complete the following steps.

1. On the **Private endpoint connections** tab, select the checkbox that now exists from the resource described as **do from AFD**.
2. Select **Approve**, and then select **Yes** to verify the approval.

Complete Azure Front Door Premium configuration

The following steps explain how to complete the configuration of Azure Front Door Premium.

1. In the Azure portal, enter the Azure Front Door Premium service you previously created:

   `example-com-frontdoor`

2. In the **Endpoint Manager** window, select **Edit endpoint** to modify the endpoint.

3. Select **+Add** under **Routes**.

4. Give your route an appropriate name, such as **Reservations-App-Route-Config**.

5. Under **Domains**, then under **Available validated domains**, select the fully qualified domain name, for example:

   `reservations.apps.example.com`

6. To redirect HTTP traffic to use HTTPS, leave the **Redirect** checkbox selected.

7. Under **Origin group**, select **Reservations-App**, the origin group you previously created.

8. You can enable caching, if appropriate.

9. Select **Add** to create the route. After the route is configured, the **Endpoint manager** populates the **Domains** and **Origin groups** panes with the other elements created for this application.

Because Azure Front Door is a global service, the application can take up to 30 minutes to deploy. During this time, you may choose to create a WAF for your application. When your application goes live, it can be accessed using the URL used in this example:

`https://reservations.apps.example.com`

Next steps

Create an Azure Web Application Firewall on Azure Front Door using the Azure portal:

Tutorial: Create a Web Application Firewall policy on Azure Front Door using the Azure portal

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Create a service principal to deploy an Azure Red Hat OpenShift cluster

07/18/2025

This article explains how to create a service principal using the Azure CLI or Azure portal. You can use the service principal when you deploy a Microsoft Azure Red Hat OpenShift cluster. A new cluster deployment also creates a service principal.

To interact with Azure APIs, a Microsoft Azure Red Hat OpenShift cluster requires a Microsoft Entra service principal. This service principal is used to dynamically create, manage, or access other Azure resources, such as an Azure load balancer or an Azure Container Registry. For more information, see Application and service principal objects in Microsoft Entra ID.

Service principals expire in one year unless configured for longer periods. For information about how to extend your service principal expiration period, see Rotate service principal credentials for your Azure Red Hat OpenShift cluster.

Create a service principal

The following sections explain how to create a service principal to deploy an Azure Red Hat OpenShift cluster.

Prerequisites

If you're using the Azure CLI, you need Azure CLI version 2.30.0 or later installed and configured. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI.

Create a resource group

To create a resource group for your Azure Red Hat OpenShift cluster, run the following Azure CLI command. The resource group name is stored in the `AZ_RG` variable.

Azure CLI

```
AZ_RG=$(az group create --name test-aro-rg --location eastus2 --query name --
output tsv)
```

Create a service principal and assign role-based access control

Service principals must be unique per Azure RedHat OpenShift cluster. The following commands create the `AZ_SUB_ID` variable to store your Azure subscription ID and assign the *Contributor* role and scope the service principal to the Azure Red Hat OpenShift resource group.

Azure CLI

```
AZ_SUB_ID=$(az account show --query id --output tsv)
az ad sp create-for-rbac --name "test-aro-sp" --role Contributor --scopes
"/subscriptions/${AZ_SUB_ID}/resourceGroups/${AZ_RG}"
```

The output is similar to the following example:

Output

```
{
  "appId": "55556666-ffff-7777-aaaa-8888bbbb9999",
  "displayName": "test-aro-sp",
  "password": "Gg7Hh~8Ii9.-Jj0Kk1Ll2Mm3Nn4Oo5_Pp6Qq7Rr8",
  "tenant": "bbbbcccc-1111-dddd-2222-eeee3333ffff"
}
```

Make note of this information and store it in a safe place. The password value is only displayed once. For more information about the command, see az ad sp create-for-rbac.

> **ⓘ Important**
>
> This service principal only allows a Contributor over the resource group that contains the Azure Red Hat OpenShift cluster. If your virtual network is in another resource group, you need to assign the service principal Contributor role to that resource group. You also need to create your Azure Red Hat OpenShift cluster in the resource group you created for the service principal.

To grant permissions to an existing service principal with the Azure portal, see Create a Microsoft Entra app and service principal in the portal.

Clean up resources

You can delete the app registration and service principal from Microsoft Entra ID if you don't need it.

The following commands get the application ID and delete the app registration and service principal.

Azure CLI

```
APP_ID=$(az ad app list --display-name test-aro-sp --query [].appId --output tsv)
az ad app delete --id $APP_ID
```

Related content

For more information about how to create a service principal in Azure CLI and assign roles, see Create an Azure service principal with Azure CLI and Assign Azure roles using Azure CLI.

Configure Microsoft Entra authentication for an Azure Red Hat OpenShift cluster using Azure portal

08/06/2025

In this article, you use Azure portal to set up Microsoft Entra authentication for an Azure Red Hat OpenShift cluster. You create variables that are used in commands that create an `OAuth` callback URL, create an application registration and client secret, and sign in to the cluster's web console using the Microsoft Entra authentication.

Prerequisites

- An existing Azure Red Hat OpenShift cluster. If you don't have a cluster, see create a new cluster.
- Azure CLI 2.30.0 or later installed on your computer. Use `az --version` to find the installed version of Azure CLI. If you need to install or upgrade, see Install Azure CLI. You can also use Azure Cloud Shell with Bash to run commands.

Create variables

Set the variables for resource group and cluster name. Replace `<resourceGroupName>` with your resource group's name and `<aroClusterName>` with your cluster's name.

Azure CLI

```
resourceGroup=<resourceGroupName>
aroCluster=<aroClusterName>
```

Create the cluster's `OAuth` callback URL and make note of it because you use it later. The `entraID` section in the `OAuth` callback URL must match the `OAuth` identity provider name you set up later.

Azure CLI

```
domain=$(az aro show --resource-group $resourceGroup --name $aroCluster --query
clusterProfile.domain --output tsv)
location=$(az aro show --resource-group $resourceGroup --name $aroCluster --query
location --output tsv)
```

```
echo "OAuth callback URL: https://oauth-
openshift.apps.$domain.$location.aroapp.io/oauth2callback/entraID"
```

Create a Microsoft Entra application for authentication

1. Sign in to the Azure portal.
2. Use the search field to find *Microsoft Entra ID* and select it from the list.
3. Select **App registrations** > **New registration**.
4. Complete the **Register an application** form.
    - **Name**: Enter an application name, for this example *aro-entraid-auth*.
    - **Supported account types**: Select *Accounts in this organizational directory only*.
    - **Redirect URI**: Select `Web` and enter your `OAuth` callback URL.
5. Select **Register**.

    After the application is created, you're taken to its **App registration** page that displays *aro-entraid-auth*.
6. Select **Certificates & secrets** and select **New client secret**.
    - **Description**: Enter a description for the client secret like *aro-client-secret*.
    - **Expires**: Accept the default 180 days or select a different expiration value.
7. Select **Add**.
8. Copy the **Value** and store it in a safe location. You need it later and can't retrieve the value again.
9. Select **Overview** and copy the **Application (client) ID** and **Directory (tenant) ID** because you use it later.

Configure optional claims

Application developers can use optional claims in their Microsoft Entra applications to specify which claims they want in tokens sent to their application.

You can use optional claims to:

- Select other claims to include in tokens for your application.

- Change the behavior of certain claims that Microsoft Entra ID returns in tokens.
- Add and access custom claims for your application.

You can configure OpenShift to use the `email` claim and fall back to `upn` to set the Preferred Username by adding the `upn` as part of the ID token returned by Microsoft Entra ID.

Go to **Token configuration** and select **Add optional claim**. Select **ID** then check the **email** and **upn** claims.

Assign users and groups to the cluster (optional)

Applications registered in a Microsoft Entra tenant are, by default, available to all users of the tenant who authenticate successfully. Microsoft Entra ID allows tenant administrators and developers to restrict an app to a specific set of users or security groups in the tenant.

For more information, see assign users and groups to the app.

Configure OpenShift OpenID authentication

Retrieve the `kubeadmin` credentials. Run the following command to find the password for the `kubeadmin` user.

Azure CLI

```
az aro list-credentials \
  --name $aroCluster \
  --resource-group $resourceGroup
```

The output displays the cluster's user name and password.

Output

```
{
  "kubeadminPassword": "<generated password>",
  "kubeadminUsername": "kubeadmin"
}
```

You can find the cluster console URL by running the following command.

Azure CLI

```
 az aro show \
  --name $aroCluster \
```

```
    --resource-group $resourceGroup \
    --query "consoleProfile.url" --output tsv
```

The output looks like the following example.

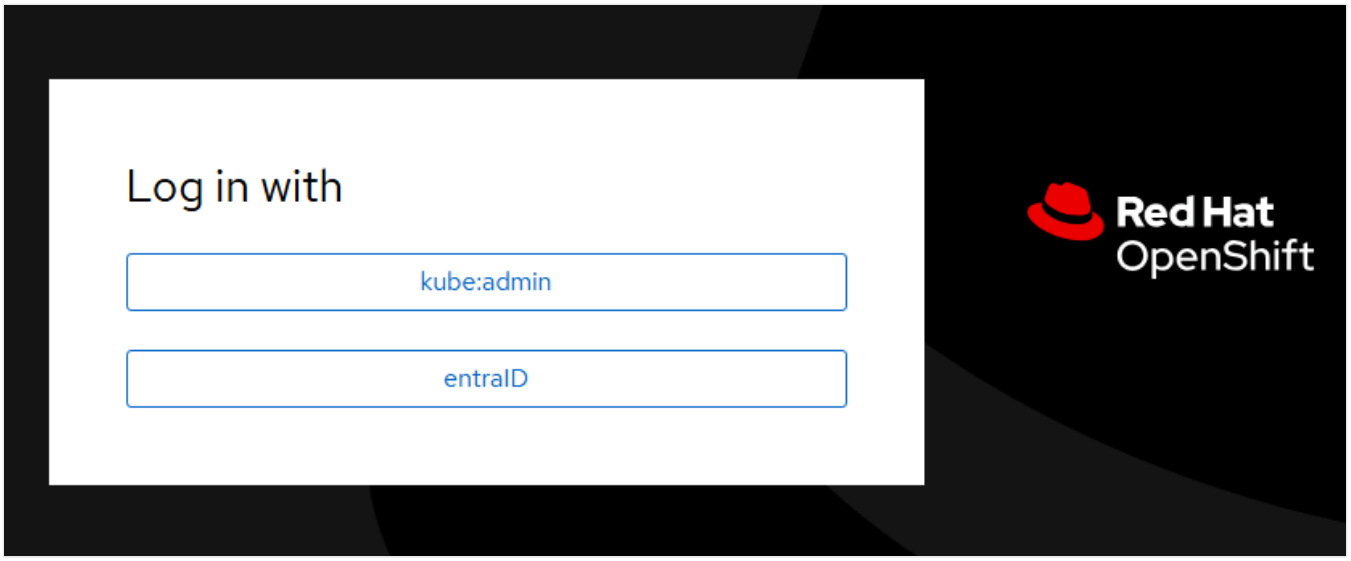
Output

```
https://console-openshift-console.apps.<domain>.<region>.aroapp.io/
```

1. Launch the console URL in a browser and sign in using the `kubeadmin` credentials.
2. Go to **Administration** > **Cluster Settings** and select the **Configuration** tab.
3. Scroll down and select **OAuth**.
4. Scroll down to **Identity Providers** > **Add** and select **OpenID Connect**.
    - **Name**: Enter `entraID`.
    - **Client ID**: Enter your app registrations Application (client) ID.
    - **Client secret**: Enter your app's client secret.
    - **Issuer URL**: Enter `https://login.microsoftonline.com/<tenantId>` and replace `<tenantId>` with your tenant ID.
5. Scroll down to the **Claims** section and update the **Preferred Username** to use the value *upn*.
6. Select **Add**.

Verify sign in using Microsoft Entra ID

Sign out of the OpenShift Web Console and sign in again, and you see a new option to sign in with **entraID**. You might need to wait for a few minutes for the option to become available.



If you get the error `AADSTS50011: The redirect URI https://oauth-openshift.apps.<domain>.<regions>.aroapp.io/oauth2callback/<identity> specified in the request doesn't match the redirect URIs configured`, you can follow the troubleshooting guide for the redirect URI and Error AADSTS50011 with OpenID authentication.

Related content

- Create a service principal to deploy an Azure Red Hat OpenShift cluster
- Rotate service principal credentials for your Azure Red Hat OpenShift cluster

Configure Microsoft Entra authentication for an Azure Red Hat OpenShift cluster

08/05/2025

In this article, you use Azure CLI to set up Microsoft Entra authentication for an Azure Red Hat OpenShift cluster. You create variables that are used in commands that create an `OAuth` callback URL, an application registration and client secret, and update the application's permissions.

Prerequisites

- An existing Azure Red Hat OpenShift cluster. If you don't have a cluster, see create a new cluster.
- Azure CLI 2.30.0 or later installed on your computer. Use `az --version` to find the installed version of Azure CLI. If you need to install or upgrade, see Install Azure CLI. You can also use Azure Cloud Shell with Bash to run commands.
- OpenShift CLI installed on your computer or in your Bash Cloud Shell. For more information, see Install the OpenShift CLI.

Create variables

Retrieve your cluster-specific URLs that are used to configure the Microsoft Entra application.

Set the variables for resource group and cluster name. Replace `<resourceGroupName>` with your resource group's name and `<aroClusterName>` with your cluster's name.

Azure CLI

```
resourceGroup=<resourceGroupName>
aroCluster=<aroClusterName>
```

Create the following variables that are used in other commands to complete the steps in this article.

Azure CLI

```
domain=$(az aro show --resource-group $resourceGroup --name $aroCluster --query
clusterProfile.domain --output tsv)
location=$(az aro show --resource-group $resourceGroup --name $aroCluster --query
location --output tsv)
apiServer=$(az aro show --resource-group $resourceGroup --name $aroCluster --query
```

```
apiserverProfile.url --output tsv)
webConsole=$(az aro show --resource-group $resourceGroup --name $aroCluster --
query consoleProfile.url --output tsv)
```

Create the cluster's `OAuth` callback URL and store it in the variable `oauthCallbackURL`. The `entraID` at the end of the `OAuth` callback URL must match the `OAuth` identity provider name that you create later in the *oidc.yaml* file.

The format of the `oauthCallbackURL` is dependent on if you're using a custom domain.

- If you have a custom domain like `contoso.com`, run the following command.

  Azure CLI

  ```
  oauthCallbackURL=https://oauth-openshift.apps.$domain/oauth2callback/entraID
  ```

- If you're not using a custom domain, the `$domain` is an eight character alphanumeric string that precedes the `$location.aroapp.io`.

  Azure CLI

  ```
  oauthCallbackURL=https://oauth-
  openshift.apps.$domain.$location.aroapp.io/oauth2callback/entraID
  ```

Create a Microsoft Entra application for authentication

Create a Microsoft Entra application and set a client secret for the application.

Azure CLI

```
appId=$(az ad app create \
  --display-name aro-auth \
  --web-redirect-uris $oauthCallbackURL \
  --query appId --output tsv)

clientSecret=$(az ad app credential reset --id $appId --query password --output
tsv)
```

The values are stored in variables and `clientSecret` displays a console message that the command output contains credentials. The value is used later in this article and when you're finished you can clear the variable using the `clientSecret=""` command.

For more information about the commands to create the app and credentials, see az ad app create and az ad app credential reset.

Retrieve the tenant ID of the subscription that owns the application.

Azure CLI

```
tenantId=$(az account show --query tenantId --output tsv)
```

Create a manifest file to define the optional claims to include in the ID Token

Application developers can use optional claims in their Microsoft Entra applications to specify which claims they want in tokens sent to their application.

You can use optional claims to:

- Select other claims to include in tokens for your application.
- Change the behavior of certain claims that Microsoft Entra ID returns in tokens.
- Add and access custom claims for your application.

You configure OpenShift to use the `email` claim and fall back to `upn` to set the Preferred Username by adding the `upn` as part of the ID token returned by Microsoft Entra ID.

Create a *manifest.json* file to configure the Microsoft Entra application.

Bash

```
cat > manifest-test-file.json<< EOF
{
  "idToken": [
    {
      "name": "email",
      "source": null,
      "essential": false,
      "additionalProperties": []
    },
    {
      "name": "upn",
      "source": null,
      "essential": false,
      "additionalProperties": []
    }
  ]
}
EOF
```

Update the Microsoft Entra application's optionalClaims with a manifest

To use the manifest file and update the application's `optionalClaims`, run the following command.

Azure CLI

```
az ad app update \
  --id $appId \
  --optional-claims @manifest.json
```

Update the Microsoft Entra application scope permissions

To be able to read the user information from Microsoft Entra ID, we need to define the proper scopes.

Add permission for the Microsoft Graph `User.Read` scope to enable sign in and read user profile. For more information, see User.Read.

Azure CLI

```
az ad app permission add \
  --api 00000003-0000-0000-c000-000000000000 \
  --api-permissions e1fe6dd8-ba31-4d61-89e7-88639da4683d=Scope \
  --id $appId
```

You can ignore the message to grant the consent unless you're authenticated as a Global Administrator for this Microsoft Entra ID. Standard domain users are asked to grant consent when they first sign in to the cluster with their Microsoft Entra credentials.

Assign users and groups to the cluster (optional)

Applications registered in a Microsoft Entra tenant are, by default, available to all users of the tenant who authenticate successfully. Microsoft Entra ID allows tenant administrators and developers to restrict an app to a specific set of users or security groups in the tenant.

Follow the instructions on the Microsoft Entra documentation to Restrict a Microsoft Entra app to a set of users.

Configure OpenShift OpenID authentication

Retrieve the `kubeadmin` credentials. Run the following command to find the password for the `kubeadmin` user.

Azure CLI

```
kubeadminPassword=$(az aro list-credentials \
  --name $aroCluster \
  --resource-group $resourceGroup \
  --query kubeadminPassword --output tsv)
```

Sign in to the OpenShift cluster's API server using the following command.

Azure CLI

```
oc login $apiServer --username kubeadmin --password $kubeadminPassword
```

Create an OpenShift secret to store the Microsoft Entra application secret.

Azure CLI

```
oc create secret generic openid-client-secret-azuread \
  --namespace openshift-config \
  --from-literal=clientSecret=$clientSecret
```

Create an *oidc.yaml* file to configure OpenShift OpenID authentication against Microsoft Entra ID.

Bash

```
cat > oidc.yaml<< EOF
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: entraID
    mappingMethod: claim
    type: OpenID
    openID:
      clientID: $appId
      clientSecret:
        name: openid-client-secret-azuread
      extraScopes:
      - email
      - profile
```

```
      extraAuthorizeParameters:
        include_granted_scopes: "true"
      claims:
        preferredUsername:
        - email
        - upn
        name:
        - name
        email:
        - email
      issuer: https://login.microsoftonline.com/$tenantId
EOF
```

Apply the configuration to the cluster.

Console

```
oc apply -f oidc.yaml
```

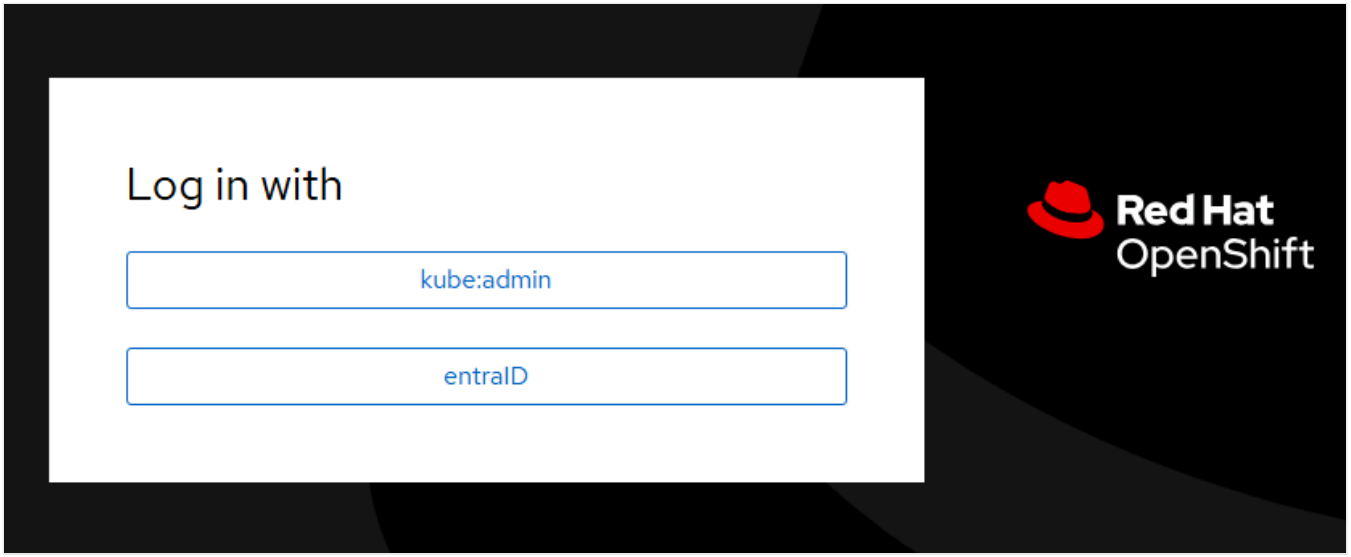
You get a response like the following example.

Output

```
oauth.config.openshift.io/cluster configured
```

Verify sign in using Microsoft Entra ID

Sign out of the OpenShift Web Console and sign in again, and you'll see a new option to sign in with **entraID**. You might need to wait for a few minutes for the option to become available.



Related content

- Create a service principal to deploy an Azure Red Hat OpenShift cluster
- Rotate service principal credentials for your Azure Red Hat OpenShift cluster

Add or update your Red Hat pull secret on an Azure Red Hat OpenShift 4 cluster

07/09/2025

This guide covers how to add or update your Red Hat pull secret for an existing Azure Red Hat OpenShift 4 cluster.

For a new cluster deployment, you can add your pull secret when you create the cluster. For more information about how to create cluster with a Red Hat pull secret, see Get a Red Hat pull secret (optional).

Prerequisites

This guide assumes you have an existing Azure Red Hat OpenShift 4 cluster. Ensure that you have administrator access to your cluster.

Prepare your pull secret

When you create a cluster without adding a Red Hat pull secret, a pull secret is still created on your cluster automatically. However, this pull secret isn't fully populated.

This section walks through updating that pull secret with other values from your Red Hat pull secret.

1. Fetch the secret named `pull-secret` in the `openshift-config` namespace and save it to a separate file by running the following command:

   Console

   ```
   oc get secrets pull-secret -n openshift-config -o template='{{index .data
   ".dockerconfigjson"}}' | base64 -d > pull-secret.json
   ```

   Your output should be similar to the following example and the actual secret value was removed.

   JSON

   ```
   {
     "auths": {
       "arosvc.azurecr.io": {
         "auth": "<my-aroscv.azurecr.io-secret>"
       }
   ```

```
  }
}
```

2. Go to your Red Hat OpenShift cluster manager portal and select **Download pull secret**. Your Red Hat pull secret looks like the following sample and the actual secret values were removed.

JSON

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<my-crc-secret>",
      "email": "violet@contoso.com"
    },
    "quay.io": {
      "auth": "<my-quayio-secret>",
      "email": "violet@contoso.com"
    },
    "registry.connect.redhat.com": {
      "auth": "<my-registry.connect.redhat.com-secret>",
      "email": "violet@contoso.com"
    },
    "registry.redhat.io": {
      "auth": "<my-registry.redhat.io-secret>",
      "email": "violet@contoso.com"
    }
  }
}
```

3. Edit the pull secret file you got from your cluster by adding in the entries found in your Red Hat pull secret.

> **ⓘ Important**
>
> If you include the `cloud.openshift.com` entry from your Red Hat pull secret your cluster sends telemetry data to Red Hat. Include this section only if you want to send telemetry data. Otherwise, exclude the following section.

JSON

```
{
  "cloud.openshift.com": {
    "auth": "<my-crc-secret>",
    "email": "violet@contoso.com"
  }
```

> **⊗ Caution**
>
> Don't remove or alter the `arosvc.azurecr.io` entry from your pull secret. This section is needed for your cluster to function properly.

JSON

```
"arosvc.azurecr.io": {
  "auth": "<my-aroscv.azurecr.io-secret>"
}
```

Your final file should look like the following example and the actual secret values were removed.

JSON

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "<my-crc-secret>",
      "email": "violet@contoso.com"
    },
    "quay.io": {
      "auth": "<my-quayio-secret>",
      "email": "violet@contoso.com"
    },
    "registry.connect.redhat.com": {
      "auth": "<my-registry.connect.redhat.com-secret>",
      "email": "violet@contoso.com"
    },
    "registry.redhat.io": {
      "auth": "<my-registry.redhat.io-secret>",
      "email": "violet@contoso.com"
    },
    "arosvc.azurecr.io": {
      "auth": "<my-aroscv.azurecr.io-secret>"
    }
  }
}
```

4. Ensure that the file is valid JSON. There are many ways to validate your JSON. The following example uses `jq` to validate the file.

JSON

```
cat pull-secret.json | jq
```

If there's an error in the file, it appears as `parse error`.

Add your pull secret to your cluster

Run the following command to update your pull secret. In version 4.9 or older, running this command causes your cluster nodes to restart one by one as they're updated. In version 4.10 or later, a restart isn't triggered.

Console

```
oc set data secret/pull-secret -n openshift-config --from-
file=.dockerconfigjson=./pull-secret.json
```

Verify that pull secret is in place

Console

```
oc exec -n openshift-apiserver $(oc get pod -n openshift-apiserver -o jsonpath="
{.items[0].metadata.name}") -- cat /var/lib/kubelet/config.json
```

After the secret is set, you're ready to enable Red Hat Certified Operators.

Modify the configuration files

Modify the following objects to enable Red Hat Operators.

Modify the Samples Operator configuration file. Then, you can run the following command to edit the configuration file:

Console

```
oc edit configs.samples.operator.openshift.io/cluster -o yaml
```

Change the `spec.managementState` value from `Removed` to `Managed`.

The following YAML snippet shows only the relevant sections of the edited YAML file:

YAML

```yaml
apiVersion: samples.operator.openshift.io/v1
kind: Config
metadata:
```

```
  ...

spec:
  architectures:
  - x86_64
  managementState: Managed
```

Run the following command to edit the Operator Hub configuration file:

Console

```
oc edit operatorhub cluster -o yaml
```

Change the `Spec.Sources.Disabled` value from `true` to `false` for any sources you want enabled.

The following YAML snippet shows only the relevant sections of the edited YAML file:

YAML

```
Name:         cluster

...
                dd3310b9-e520-4a85-98e5-8b4779ee0f61
Spec:
  Sources:
    Disabled:  false
    Name:      certified-operators
    Disabled:  false
    Name:      redhat-operators
```

Save the file to apply your edits.

Validate that your secret is working

After you add your pull secret and modify the correct configuration files, your cluster updates can take several minutes to finish. To check that your cluster was updated, run the following command to show the Certified Operators and Red Hat Operators sources available:

Console

```
$ oc get catalogsource -A
NAMESPACE               NAME                  DISPLAY               TYPE
PUBLISHER   AGE
openshift-marketplace   certified-operators   Certified Operators   grpc   Red Hat
10s
openshift-marketplace   community-operators   Community Operators   grpc   Red Hat
```

```
18h
openshift-marketplace   redhat-operators      Red Hat Operators     grpc   Red Hat
11s
```

If you don't see the Certified Operators and Red Hat Operators, wait a few minutes and try again.

To ensure that your pull secret was updated and is functional, open **OperatorHub** and check for any Red Hat verified Operator. For example, check to see if the OpenShift Container Storage Operator is available, and see if you have permissions to install.

Next steps

- To learn more about Red Hat pull secrets, see Using image pull secrets.
- To learn more about Red Hat OpenShift 4, see Red Hat OpenShift Container Platform Documentation.

Rotate service principal credentials for your Azure Red Hat OpenShift cluster

07/17/2025

The article explains how to to rotate Microsoft Entra ID service principal credentials for an Azure Red Hat OpenShift cluster. The Azure CLI commands use Bash syntax and can be run in Azure Cloud Shell.

Prerequisites

- An existing Azure Red Hat OpenShift cluster with the latest updates applied.
- Azure CLI version 2.24.0 is required to rotate service principal credentials. To check the version of Azure CLI run `az --version`. If you need to upgrade, see How to install the Azure CLI.

Service principal credential rotation

Service principal credential rotation can take two hours depending on cluster state. There are two methods for service principal credential rotation:

- Automated service principal credential rotation
- User provided client ID and client secret service principal credential rotation

For either method, create variables for your cluster's name and resource group. Replace `<clusterName>` and `<resourceGroupName>` with your cluster's values.

Azure CLI

```
CLUSTER=<clusterName>
RESOURCEGROUP=<resourceGroupName>
```

Automated service principal credential rotation

The method for automated service principal credential rotation requires that the cluster was created with Azure CLI version 2.24.0 or greater. Automated service principal credential rotation checks if the service principal exists and rotates or creates a new service principal.

You can automatically rotate service principal credentials with the following command.

Azure CLI

```
az aro update --refresh-credentials --name $CLUSTER --resource-group
$RESOURCEGROUP
```

User provided client ID and client secret service principal credential rotation

You can manually rotate service principal credentials with user provided client ID and client secret with the following instructions.

Retrieve the service principal client ID (`--client-id`) and set it as `SP_ID` environment variable.

Azure CLI

```
SP_ID=$(az aro show --name $CLUSTER --resource-group $RESOURCEGROUP \
    --query servicePrincipalProfile.clientId --output tsv)
```

Generate a new secure secret (`--client-secret`) for the service principal using the `SP_ID` variable. Store the new secure secret as `SP_SECRET` environment variable.

Azure CLI

```
SP_SECRET=$(az ad sp credential reset --id $SP_ID --query password --output tsv)
```

Rotate the service principal credentials using the environment variables.

Azure CLI

```
az aro update --client-id $SP_ID --client-secret $SP_SECRET \
    --name $CLUSTER --resource-group $RESOURCEGROUP
```

Troubleshoot

Service principal expiration date

Service principal credentials have a set expiration date of a year and should be rotated within that timeframe. If the credentials are expired, the following errors are possible.

Output

```
Failed to refresh the Token for request to <resourceGroupName> StatusCode=401
Original Error: Request failed. Status Code = '401'.
```

```
[with]
Response body: {"error":"invalid_client","error_description": The provided client
secret keys are expired.
[or]
Response body: {"error":"invalid_client","error_description": Invalid client
secret is provided.
```

To check the expiration date of service principal credentials run the following commands. The date is output in ISO 8601 UTC format.

Azure CLI

```
SP_ID=$(az aro show --name $CLUSTER --resource-group $RESOURCEGROUP \
    --query servicePrincipalProfile.clientId --output tsv)
az ad app credential list --id $SP_ID --query "[].endDateTime" --output tsv
```

If the service principal credentials are expired, update the credentials using one of the two credential rotation methods.

Cluster application contains a client secret with an empty description

The following error occurs when you use `az aro update` command for automated service principal credential rotation.

Output

```
Cluster application contains a client secret with an empty description.
Either manually remove the existing client secret and run `az aro update --
refresh-credentials`,
or manually create a new client secret and run `az aro update --client-secret
<clientSecret>`.
```

The cluster wasn't created using Azure CLI 2.24.0 or greater. Use the User provided client ID and client secret service principal credential rotation method instead.

Azure CLI commands

For more information about Azure CLI commands, see az aro documentation. You can also use commands like `az aro update --help` on the command line.

Clean up resources

When you're finished, you should clear your variables to remove any sensitive data.

Azure CLI

```
SP_SECRET=""
SP_ID=""
CLUSTER=""
RESOURCEGROUP=""
```

Related content

For more information about service principals, see Create and use a service principal to deploy an Azure Red Hat OpenShift cluster.

Enable FIPS for an Azure Red Hat OpenShift cluster

06/16/2025

This article explains how to enable Federal Information Processing Standard (FIPS) when you create an Azure Red Hat OpenShift cluster.

The Federal Information Processing Standard 140 is a US government standard that defines minimum security requirements for cryptographic modules in information technology products and systems. The Cryptographic Module Validation Program (CMVP), a joint effort between the US National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security, a branch of the Communications Security Establishment (CSE) of Canada maintain the testing against the FIPS 140 standard.

Support for FIPS cryptography

Starting with Release 4.10, you can deploy an Azure Red Hat OpenShift cluster in FIPS mode. FIPS mode ensures the control plane is using FIPS 140-2 cryptographic modules. To be FIPS compliant, all workloads and operators deployed on a cluster need to use FIPS 140-2.

You can install an Azure Red Hat OpenShift cluster that uses FIPS Validated or FIPS Modules in Process cryptographic libraries on the x86_64 architecture.

> **Note**
>
> If you're using Azure File storage, you can't enable FIPS mode.

Enable FIPS on your cluster

You must enable FIPs when you create an Azure Red Hat OpeShift cluster because FIPS can't be enabled or disabled after the cluster is created. The following example shows parameters that use values defined as environment variables for resource group, cluster name, and virtual network name. The `--fips` parameter enables FIPS mode and doesn't require a value.

Azure CLI

```
az aro create \
  --resource-group $RESOURCEGROUP \
  --name $CLUSTER \
  --vnet $VIRTUALNETWORK \
```

```
    --master-subnet master-subnet \
    --worker-subnet worker-subnet \
    --fips
```

For more information, see az-aro-create. To enable FIPS on the cluster, there are two options:

- `--fips`: Uses cryptographic modules that might be in the validation process.
- `--fips-validated-modules`: Ensures that only validated cryptographic modules are used.

Next steps

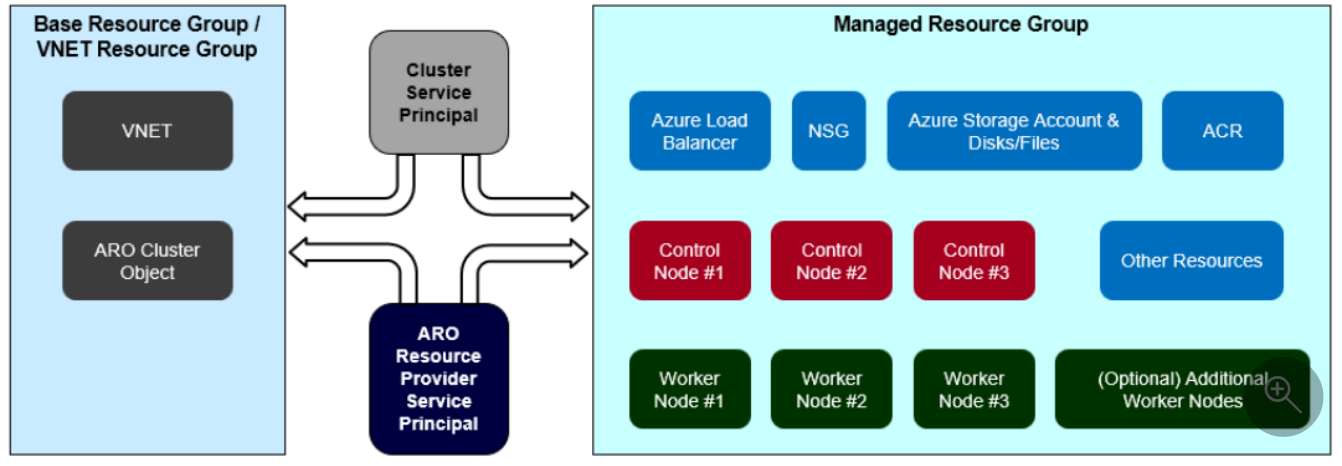
To learn how to deploy an Azure Red Hat OpenShift cluster, see Create an Azure Red Hat OpenShift 4 cluster.

Bring your own Network Security Group (NSG) to an Azure Red Hat OpenShift cluster

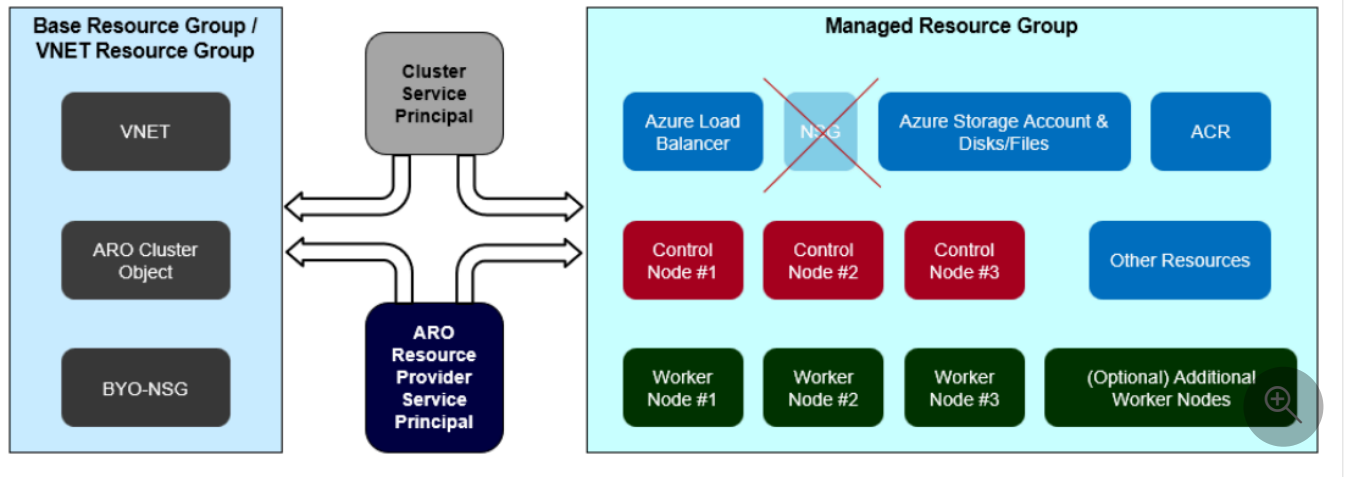
07/18/2025

Typically, when setting up a cluster, you must designate a resource group for deploying the cluster object (referred to as the Base Resource Group in the following diagram). In such scenarios, you can use either the same resource group for both the virtual network (VNET) and the cluster, or you can opt for a separate resource group solely for the VNET. Neither of these resource groups directly corresponds to a single cluster, granting you complete control over them. This means you can freely create, modify, or delete resources within these resource groups.

During the cluster creation process, the Resource Provider (RP) establishes a dedicated resource group specific to the cluster's needs. This group houses various cluster-specific resources like node VMs, load balancers, and Network Security Groups (NSGs), as depicted by the Managed Resource Group in the following diagram. The Managed Resource Group is tightly secured, prohibiting any modifications to its contents, including the NSG linked to the VNET subnets specified during cluster creation. In some situations, the NSG generated by the RP might not adhere to the security policies of certain organizations.



This article shows how to use the "bring your own" Network Security Group (NSG) feature to attach your own preconfigured NSG residing in the Base/VNET resource group (RG) (shown in the following diagram as BYO-NSG) to the cluster subnets. Since you own this preconfigured NSG, you can add/remove rules during the lifetime of the cluster.



General capabilities and limitations

- You need to attach your preconfigured NSGs to both master and worker subnets before you create the cluster. Failure to attach your preconfigured NSGs to both subnets results in an error.
- You can choose to use the same or different preconfigured NSGs for master and worker subnets.
- When using your own NSG, the RP still creates an NSG in the Managed Resource Group (default NSG), but that NSG isn't attached to the worker or master subnets.
- You can't enable the preconfigured NSG feature on an existing cluster. Currently, this feature can only be enabled at the time of cluster creation.
- The preconfigured NSG option isn't configurable from the Azure portal.
- If you used this feature during preview, your existing preconfigured clusters are now fully supported.

> ⓘ **Note**
>
> If you're using the "bring your own" NSG feature and want to use NSG flow logs, please refer to Flow logging for network security groups in Azure Network Watcher documentation, rather than the specific flow log documentation (which won't work with the bring your own NSG feature).

Using rules

> ⚠ **Warning**

Preconfigured NSGs aren't automatically updated with rules when you create Kubernetes LoadBalancer type services or OpenShift routes within the cluster. Therefore, you must update these rules manually, as required. This behavior is different from the original behavior wherein the default NSG is programmatically updated in such situations.

- The default cluster NSG (not attached to any subnet while using this feature) will still be updated with rules when you create Kubernetes LoadBalancer type services or OpenShift routes within the cluster.
- You can detach preconfigured NSGs from the subnets of the cluster created using this feature. It results in a cluster with subnets that have no NSGs. You can then attach a different set of preconfigured NSGs to the cluster. Alternatively, you can attach the default NSG to the cluster subnets (at which point your cluster becomes like any other cluster that's not using this feature).
- Your preconfigured NSGs shouldn't have INBOUND/OUTBOUND DENY rules of the following types, as these can interfere with the operation of the cluster and/or hinder the support/SRE teams from providing support/management. (Here, subnet indicates any or all IP addresses in the subnet and all ports corresponding to that subnet):
  - Master Subnet ←→ Master Subnet
  - Worker Subnet ←→ Worker Subnet
  - Master Subnet ←→ Worker Subnet
  - Misconfigured rules result in a signal used by Azure Monitor to help troubleshoot preconfigured NSGs.
- To allow incoming traffic to your public cluster, set the following INBOUND ALLOW rules (or equivalent) in your NSG. Refer to the default NSG of the cluster for specific details and to the example NSG shown in Deployment. You can create a cluster even without such rules in the NSG.
  - For API server access → From Internet (or your preferred source IPs) to port 6443 on the master subnet.
  - For access to OpenShift router (and hence to OpenShift console and OpenShift routes) → From Internet (or your preferred source IPs) to ports 80 and 443 on the default-v4 public IP on the public Load-balancer of the cluster.
  - For access to any Load-balancer type Kubernetes service → From Internet (or your preferred source IPs) to service ports on public IP corresponding to the service on the public Load-balancer of the cluster.

Deployment

Create VNET and create and configure preconfigured NSG

1. Create a VNET, and then create master and worker subnets within it.
2. Create preconfigured NSGs with default rules (or no rules at all) and attach them to the master and worker subnets.

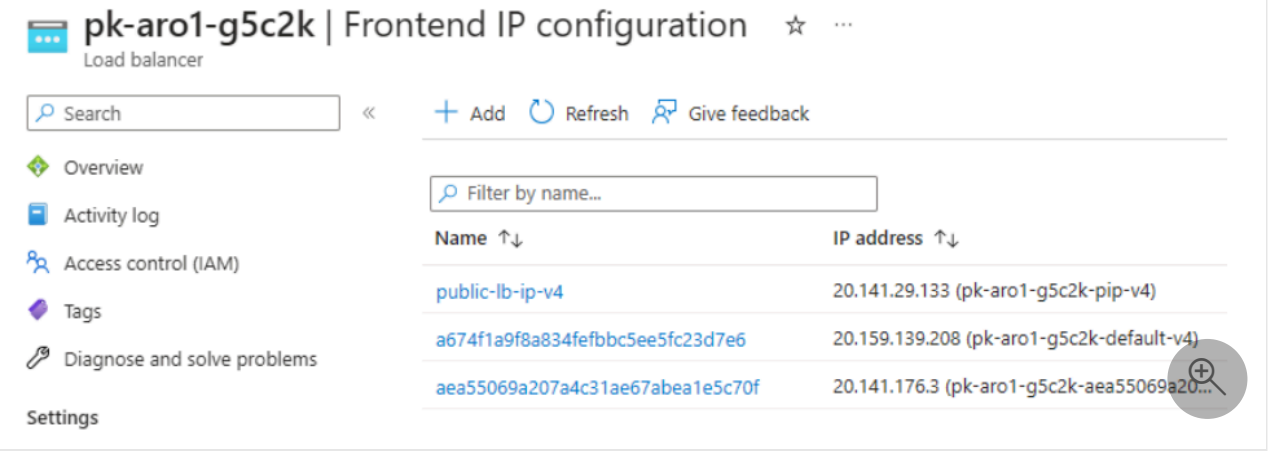
Create a cluster and update preconfigured NSGs

1. Create the cluster.

```
az aro create \
--resource-group BASE_RESOURCE_GROUP_NAME \
--name CLUSTER_NAME \
--vnet VNET_NAME \
--master-subnet MASTER_SUBNET_NAME \
--worker-subnet WORKER_SUBNET_NAME \
--client-id CLUSTER_SERVICE_PRINCIPAL_ID \
--client-secret CLUSTER_SERVICE_PRINCIPAL_SECRET \
--enable-preconfigured-nsg
```

2. Update the preconfigured NSGs with rules as per your requirements while also considering the points mentioned in Capabilities and limitations.

   The following example has the Cluster Public Load-balancer as shown in the screenshot/CLI output:



Output

```
$ oc get svc | grep tools
tools LoadBalancer 172.30.182.7 20.141.176.3 80:30520/TCP 143m
$ $ oc get svc -n openshift-ingress | grep Load
router-default LoadBalancer 172.30.105.218 20.159.139.208
80:31157/TCP,443:31177/TCP
5d20
```

Priority	Name	Port	Protocol	Source	Destination	Action
Inbound Security Rules						
111	AllowInToMasterSubnetForAPI	6443	Any	75.45.44.95	10.0.1.0/24	Allow
121	AllowInToOpenshiftRouter	80,443	Any	75.45.44.95	20.159.139.208	Allow
131	AllowInToServiceOnK8s	80,443	Any	75.45.44.95	20.141.176.3	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBou···	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny
Outbound Security Rules						
65000	AllowVnetOutBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowInternetOutBound	Any	Any	Any	Internet	Allow
65500	DenyAllOutBound	Any	Any	Any	Any	Deny

Authorize support requests for cluster access with Azure Lockbox

Article • 02/25/2025

In some circumstances, a support agent at Microsoft may need access to your OpenShift cluster resources. The Azure Lockbox feature works with Azure Red Hat OpenShift to provide customers a way to review and approve or reject requests from Microsoft support to access their cluster resources. This ability can be important for financial, government, or other regulatory industries where there's extra scrutiny regarding access to resources.

With Azure Lockbox, whenever a support ticket is created, you have the ability to grant consent to Microsoft support agents to access your cluster resources. The actions that the support engineer can take are limited to those listed below. Azure Lockbox will tell you exactly what action the support agent is trying to execute.

See Customer Lockbox for more information about the Lockbox feature.

Access request process

The Azure Lockbox workflow consists of the following main steps:

1. A support ticket is opened from the Azure portal. The ticket is assigned to a customer support engineer at Microsoft.
2. The customer support engineer reviews the request and determines the next steps to resolve the issue.
3. When the request requires the support engineer to perform one of the actions listed below, a Lockbox request is initiated. The request is now in a **Customer Notified** state, waiting for the customer's approval before granting access.
4. An email is sent from Microsoft to the customer, notifying them about the pending access request.
5. The customer signs in to the Azure portal to view the Lockbox request and can Approve or Deny the request.

As a result of the selection:

- Approve: Access is granted to the Microsoft support engineer. The access is granted for a default period of eight hours.
- Deny: The elevated access request by the support engineer is rejected and no further action is taken.

See Customer Lockbox--workflow for another details about the access request process.

Operating limitations

- The Lockbox feature works only with customer support tickets.
- Customers can only grant access through the Lockbox interface.
- No action can be taken until customer approval is granted.

Enable Lockbox for ARO

You can enable Lockbox from the Administration module in the Customer Lockbox blade. Once you enable Lockbox, it will apply to all the ARO clusters in that subscription.

> **Note**
>
> To enable Customer Lockbox, the user account needs to have the Global Administrator role assigned.

ARO Lockbox actions

The actions below require Lockbox authorization in order for a support engineer to proceed:

- Create Kubernetes object
- Update Kubernetes object
- Delete Kubernetes object
- Get logs from a pod in the OpenShift namespace
- List or get Kubernetes objects

Auditing logs

Lockbox logs are stored in activity logs. In the Azure portal, select Activity Logs to view auditing information related to Customer Lockbox requests. See Customer Lockbox, Auditing Logs for more information.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Use Azure Key Vault Provider for Secrets Store CSI Driver on Azure Red Hat OpenShift

07/18/2025

Azure Key Vault Provider for Secrets Store CSI Driver allows you to get secret contents stored in an Azure Key Vault instance and use the Secrets Store CSI Driver to mount them into Kubernetes pods. This article explains how to use Azure Key Vault Provider for Secrets Store CSI Driver on Azure Red Hat OpenShift.

> **Note**
>
> As an alternative to the open source solution presented in this article, you can use Azure Arc to manage your clusters along with its Azure Key Vault Provider for Secrets Store CSI Driver extension. This method is fully supported by Microsoft and is recommended instead of the open source solution below.

Prerequisites

The following prerequisites are required:

- An Azure Red Hat OpenShift cluster (See Create an Azure Red Hat OpenShift cluster to learn more.)
- Azure CLI (logged in)
- Helm 3.x CLI

Set environment variables

Set the following variables that will be used throughout this procedure:

```
export KEYVAULT_RESOURCE_GROUP=${AZR_RESOURCE_GROUP:-"openshift"}
export KEYVAULT_LOCATION=${AZR_RESOURCE_LOCATION:-"eastus"}
export KEYVAULT_NAME=secret-store-$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -
w 10 | head -n 1)
export AZ_TENANT_ID=$(az account show -o tsv --query tenantId)
```

Install the Kubernetes Secrets Store CSI Driver

1. Create a project; you'll deploy the CSI Driver into this project:

```
oc new-project k8s-secrets-store-csi
```

2. Set SecurityContextConstraints to allow the CSI Driver to run (otherwise, the CSI Driver will not be able to create pods):

```
oc adm policy add-scc-to-user privileged \
  system:serviceaccount:k8s-secrets-store-csi:secrets-store-csi-driver
```

3. Add the Secrets Store CSI Driver to your Helm repositories:

```
helm repo add secrets-store-csi-driver \
  https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
```

4. Update your Helm repositories:

```
helm repo update
```

5. Install the Secrets Store CSI Driver:

```
helm install -n k8s-secrets-store-csi csi-secrets-store \
   secrets-store-csi-driver/secrets-store-csi-driver \
   --version v1.3.1 \
   --set "linux.providersDir=/var/run/secrets-store-csi-providers"
```

   Optionally, you can enable autorotation of secrets by adding the following parameters to the command above:

   `--set "syncSecret.enabled=true" --set "enableSecretRotation=true"`

6. Verify that the CSI Driver DaemonSets are running:

```
kubectl --namespace=k8s-secrets-store-csi get pods -l "app=secrets-store-csi-driver"
```

After running the command above, you should see the following:

```
NAME                                                READY   STATUS    RESTARTS   AGE
 csi-secrets-store-secrets-store-csi-driver-cl7dv   3/3     Running   0          57s
 csi-secrets-store-secrets-store-csi-driver-gbz27   3/3     Running   0          57s
```

Deploy Azure Key Vault Provider for Secrets Store CSI Driver

1. Add the Azure Helm repository:

```
helm repo add csi-secrets-store-provider-azure \
   https://azure.github.io/secrets-store-csi-driver-provider-azure/charts
```

2. Update your local Helm repositories:

```
helm repo update
```

3. Install the Azure Key Vault CSI provider:

```
helm install -n k8s-secrets-store-csi azure-csi-provider \
   csi-secrets-store-provider-azure/csi-secrets-store-provider-azure \
   --set linux.privileged=true --set secrets-store-csi-driver.install=false \
   --set "linux.providersDir=/var/run/secrets-store-csi-providers" \
   --version=v1.4.1
```

4. Set SecurityContextConstraints to allow the CSI driver to run:

```
oc adm policy add-scc-to-user privileged \
   system:serviceaccount:k8s-secrets-store-csi:csi-secrets-store-provider-
azure
```

Create key vault and a secret

1. Create a namespace for your application.

```
oc new-project my-application
```

2. Create an Azure key vault in your resource group that contains Azure Red Hat OpenShift.

```
az keyvault create -n ${KEYVAULT_NAME} \
   -g ${KEYVAULT_RESOURCE_GROUP} \
   --location ${KEYVAULT_LOCATION}
```

3. Create a secret in the key vault.

```
az keyvault secret set \
   --vault-name ${KEYVAULT_NAME} \
   --name secret1 --value "Hello"
```

4. Create a service principal for the key vault.

> ⓘ **Note**
>
> If you receive an error when creating the service principal, you may need to upgrade your Azure CLI to the latest version.

```
export SERVICE_PRINCIPAL_CLIENT_SECRET="$(az ad sp create-for-rbac --skip-
assignment --name http://$KEYVAULT_NAME --query 'password' -otsv)"
export SERVICE_PRINCIPAL_CLIENT_ID="$(az ad sp list --display-name
http://$KEYVAULT_NAME --query '[0].appId' -otsv)"
```

5. Set an access policy for the service principal.

```
az keyvault set-policy -n ${KEYVAULT_NAME} \
   --secret-permissions get \
   --spn ${SERVICE_PRINCIPAL_CLIENT_ID}
```

6. Create and label a secret for Kubernetes to use to access the key vault.

```
kubectl create secret generic secrets-store-creds \
   -n my-application \
   --from-literal clientid=${SERVICE_PRINCIPAL_CLIENT_ID} \
   --from-literal clientsecret=${SERVICE_PRINCIPAL_CLIENT_SECRET}
kubectl -n my-application label secret \
   secrets-store-creds secrets-store.csi.k8s.io/used=true
```

Deploy an application that uses the CSI Driver

1. Create a `SecretProviderClass` to give access to this secret:

```
cat <<EOF | kubectl apply -f -
 apiVersion: secrets-store.csi.x-k8s.io/v1
 kind: SecretProviderClass
 metadata:
   name: azure-kvname
   namespace: my-application
 spec:
   provider: azure
   parameters:
     usePodIdentity: "false"
     useVMManagedIdentity: "false"
     userAssignedIdentityID: ""
     keyvaultName: "${KEYVAULT_NAME}"
     objects: |
       array:
         - |
           objectName: secret1
           objectType: secret
           objectVersion: ""
     tenantId: "${AZ_TENANT_ID}"
EOF
```

2. Create a pod that uses the `SecretProviderClass` created in the previous step:

```
cat <<EOF | kubectl apply -f -
 kind: Pod
 apiVersion: v1
 metadata:
   name: busybox-secrets-store-inline
   namespace: my-application
 spec:
   containers:
   - name: busybox
     image: k8s.gcr.io/e2e-test-images/busybox:1.29
     command:
       - "/bin/sleep"
       - "10000"
     volumeMounts:
     - name: secrets-store-inline
       mountPath: "/mnt/secrets-store"
       readOnly: true
   volumes:
     - name: secrets-store-inline
       csi:
         driver: secrets-store.csi.k8s.io
         readOnly: true
         volumeAttributes:
           secretProviderClass: "azure-kvname"
         nodePublishSecretRef:
           name: secrets-store-creds
EOF
```

3. Check that the secret is mounted:

```
kubectl exec busybox-secrets-store-inline -- ls /mnt/secrets-store/
```

   The output should match the following:

```
secret1
```

4. Print the secret:

```
kubectl exec busybox-secrets-store-inline \
   -- cat /mnt/secrets-store/secret1
```

The output should match the following:

Azure CLI

```
Hello
```

Cleanup

Uninstall the Key Vault Provider and the CSI Driver.

Uninstall the Key Vault Provider

1. Uninstall Helm chart:

   Azure CLI

   ```
   helm uninstall -n k8s-secrets-store-csi azure-csi-provider
   ```

2. Delete the app:

   ```
   oc delete project my-application
   ```

3. Delete the Azure key vault:

   ```
   az keyvault delete -n ${KEYVAULT_NAME}
   ```

4. Delete the service principal:

   ```
   az ad sp delete --id ${SERVICE_PRINCIPAL_CLIENT_ID}
   ```

Uninstall the Kubernetes Secret Store CSI Driver

1. Delete the Secrets Store CSI Driver:

```
helm uninstall -n k8s-secrets-store-csi csi-secrets-store
oc delete project k8s-secrets-store-csi
```

2. Delete the SecurityContextConstraints:

```
oc adm policy remove-scc-from-user privileged \
  system:serviceaccount:k8s-secrets-store-csi:secrets-store-csi-driver
```

Create an Azure Red Hat OpenShift 4 cluster application backup

07/26/2025

In this article, you prepare your environment to create an Azure Red Hat OpenShift 4 cluster application backup using Velero. You learn how to do the following tasks:

- ✔ Setup the prerequisites and install the necessary tools
- ✔ Create an Azure Red Hat OpenShift 4 application backup

If you choose to install and use the CLI locally, this tutorial requires that you're running the Azure CLI version 2.6.0 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI.

Prerequisites

Install Velero

To install Velero on your system, follow the recommended process for your operating system.

Set up Azure storage account and Blob container

This step creates a resource group outside of the cluster's resource group. This resource group allows the backups to persist and can restore applications to new clusters.

Azure CLI

```
AZURE_BACKUP_RESOURCE_GROUP=Velero_Backups
az group create -n $AZURE_BACKUP_RESOURCE_GROUP --location eastus

AZURE_STORAGE_ACCOUNT_ID="velero$(uuidgen | cut -d '-' -f5 | tr '[A-Z]' '[a-z]')"
az storage account create \
  --name $AZURE_STORAGE_ACCOUNT_ID \
  --resource-group $AZURE_BACKUP_RESOURCE_GROUP \
  --sku Standard_GRS \
  --encryption-services blob \
  --https-only true \
  --kind BlobStorage \
  --access-tier Hot

BLOB_CONTAINER=velero
```

```
az storage container create -n $BLOB_CONTAINER --public-access off --account-name
$AZURE_STORAGE_ACCOUNT_ID
```

Set permissions for Velero

Create service principal

Velero needs permissions to do backups and restores. When you create a service principal, you're giving Velero permission to access the resource group you define in the previous step. This step gets the cluster's resource group:

Bash

```
export AZURE_RESOURCE_GROUP=$(az aro show --name <name of cluster> --resource-
group <name of resource group> | jq -r .clusterProfile.resourceGroupId | cut -d
'/' -f 5,5)
```

Azure CLI

```
AZURE_SUBSCRIPTION_ID=$(az account list --query '[?isDefault].id' -o tsv)

AZURE_TENANT_ID=$(az account list --query '[?isDefault].tenantId' -o tsv)
```

Azure CLI

```
AZURE_CLIENT_SECRET=$(az ad sp create-for-rbac --name "velero" --role
"Contributor" --query 'password' -o tsv \
--scopes  /subscriptions/$AZURE_SUBSCRIPTION_ID)
AZURE_CLIENT_ID=$(az ad sp list --display-name "velero" --query '[0].appId' -o
tsv)
```

Bash

```
cat << EOF  > ./credentials-velero.yaml
AZURE_SUBSCRIPTION_ID=${AZURE_SUBSCRIPTION_ID}
AZURE_TENANT_ID=${AZURE_TENANT_ID}
AZURE_CLIENT_ID=${AZURE_CLIENT_ID}
AZURE_CLIENT_SECRET=${AZURE_CLIENT_SECRET}
AZURE_RESOURCE_GROUP=${AZURE_RESOURCE_GROUP}
AZURE_CLOUD_NAME=AzurePublicCloud
EOF
```

Install Velero on Azure Red Hat OpenShift 4 cluster

You need to install Velero into its own project and the custom resource definitions necessary to do backups and restores with Velero. Make sure you're successfully signed in to an Azure Red Hat OpenShift v4 cluster.

Bash

```
velero install \
--provider azure \
--plugins velero/velero-plugin-for-microsoft-azure:v1.1.0 \
--bucket $BLOB_CONTAINER \
--secret-file ~/path/to/credentials-velero.yaml \
--backup-location-config
resourceGroup=$AZURE_BACKUP_RESOURCE_GROUP,storageAccount=$AZURE_STORAGE_ACCOUNT_I
D \
--snapshot-location-config apiTimeout=15m \
--velero-pod-cpu-limit="0" --velero-pod-mem-limit="0" \
--velero-pod-mem-request="0" --velero-pod-cpu-request="0"
```

Create a backup with Velero

To create an application backup with Velero, you need to include the application's namespace. If you have a `nginx-example` namespace and want to include all the resources in that namespace in the backup, run the following command in the terminal:

Bash

```
velero create backup <name of backup> --include-namespaces=nginx-example
```

You can check the status of the backup by running:

Bash

```
oc get backups -n velero <name of backup> -o yaml
```

A successful backup outputs `phase:Completed` and the objects are stored in the container in the storage account.

Create a backup with Velero to include snapshots

To create an application backup with Velero to include the persistent volumes of your application, you need to include the application's namespace and include the `snapshot-`

`volumes=true` flag when you create the backup.

```
Bash

velero backup create <name of backup> --include-namespaces=nginx-example --
snapshot-volumes=true --include-cluster-resources=true
```

You can check the status of the backup by running:

```
Bash

oc get backups -n velero <name of backup> -o yaml
```

A successful backup outputs `phase:Completed` and the objects are stored in the container in the storage account.

For more information, see Backup OpenShift resources the native way

Next steps

In this article, an Azure Red Hat OpenShift 4 cluster application was backed up. You learned how to:

- ✓ Create a OpenShift v4 cluster application backup using Velero
- ✓ Create a OpenShift v4 cluster application backup with snapshots using Velero

Go to the next article to learn how to create an Azure Red Hat OpenShift 4 cluster application restore.

- Create a Azure Red Hat OpenShift 4 cluster application restore
- Create a Azure Red Hat OpenShift 4 cluster application restore including snapshots

Create an Azure Red Hat OpenShift 4 cluster application restore using Velero

07/26/2025

In this article, you prepare your environment to create a Microsoft Azure Red Hat OpenShift cluster application restore. You learn how to:

- ✔ Setup the prerequisites and install the necessary tools
- ✔ Create an Azure Red Hat OpenShift 4 application restore

If you choose to install and use the CLI locally, this article requires that you're running the Azure CLI version 2.6.0 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI.

Prerequisites

- An Azure Red Hat OpenShift 4 application backup. To create an Azure Red Hat OpenShift 4 application backup, see Create an Azure Red Hat OpenShift 4 backup.

Restore an Azure Red Hat OpenShift 4 Application

These steps allow you to restore an application that was backed up with Velero. You can check the list of cluster backups to see which backups are available for restore.

To list the backups, run the following command that assumes that you installed Velero in a project named `velero`.

Bash

```
oc get backups -n velero
```

After you have the backup that you want to restore, you do the restore with the following command.

Bash

```
velero restore create <name of restore> --from-backup <name of backup from above
output list>
```

This step creates the Kubernetes objects that were backed up from the previous step when creating a backup.

To see the status of the restore, run the following command.

```
Bash

oc get restore -n velero <name of restore created previously> -o yaml
```

When the phase says `Completed`, your Azure Red Hat OpenShift application should be restored.

Restore an Azure Red Hat OpenShift 4 application with included snapshots

To create a restore of an Azure Red Hat OpenShift 4 application with persistent volumes that use Velero, you do the restore with the following command.

```
Bash

velero restore create <name of the restore> --from-backup <name of backup from
above output list> --exclude-
resources="nodes,events,events.events.k8s.io,backups.ark.heptio.com,backups.velero
.io,restores.ark.heptio.com,restores.velero.io"
```

This following command creates the Kubernetes objects that were backed up from the previous command.

To see the status of the restore, execute the following command.

```
Bash

oc get restore -n velero <name of restore created previously> -o yaml
```

When the phase says `Completed`, your Azure Red Hat OpenShift application should be restored.

For more information, see Backup OpenShift resources the native way.

Next steps

In this article, an Azure Red Hat OpenShift cluster application was restored. You learned how to:

- ✓ Create a OpenShift v4 cluster application restore using Velero
- ✓ Create a OpenShift v4 cluster application restore with snapshots using Velero

For information about Azure Red Hat OpenShift supported resources, see Azure Red Hat OpenShift v4 supported resources.

Configure Azure Resource Health alerts for Azure Red Hat OpenShift (ARO) clusters

Article • 02/25/2025

Azure Resource Health is a component of Azure Monitor that can be configured to generate alerts based on signals from Azure Red Hat OpenShift clusters. These alerts help you prepare for events such as planned and unplanned maintenance.

Resource Health signals can generate one or more of the following alerts:

- **Cluster maintenance operation pending:** This signal indicates that your Azure Red Hat OpenShift cluster will undergo a maintenance operation within the next two weeks. This may cause rolling reboots of nodes resulting in workload pod restarts.
- **Cluster maintenance operation in progress:** This signal indicates one of the following operation types:
  - **Planned:** A planned maintenance operation has started on your Azure Red Hat OpenShift cluster. This may cause rolling reboots of nodes resulting in workload pod restarts.
  - **Unplanned:** An unplanned maintenance operation has started on your Azure Red Hat OpenShift cluster. This may cause rolling reboots of nodes resulting in workload pod restarts.
- **Action needed to complete maintenance operation:** This signal indicates that action is needed to complete an ongoing maintenance operation of your Azure Red Hat OpenShift cluster. Contact Azure Support to complete the ongoing maintenance operation of your Azure Red Hat OpenShift cluster.
- **Cluster API server is unreachable:** This signal indicates that the Azure Red Hat OpenShift service Resource Provider is unable to reach your cluster's API server. Your cluster is hence unable to be monitored and is unmanageable.

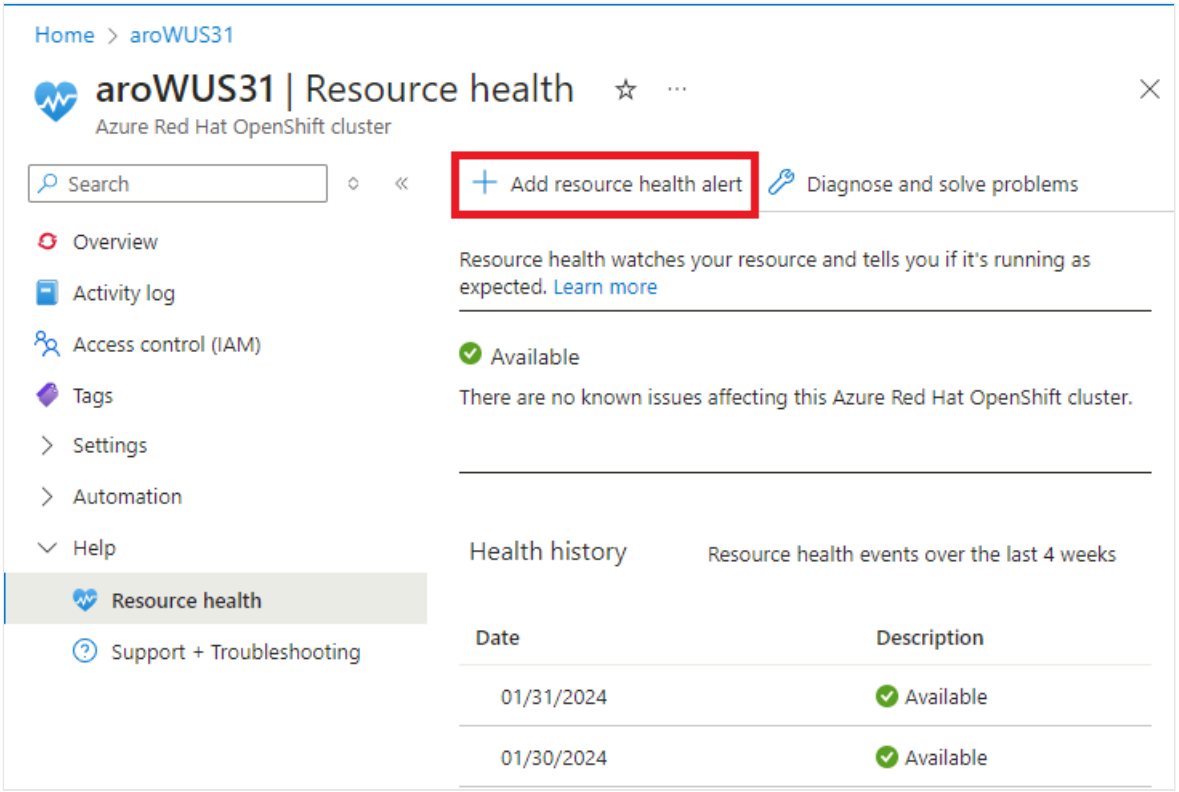
Once the underlying condition causing an alert is remediated, the alert is cleared and the Resource Health is reported as *Available*.

> **Note**
>
> This feature is not currently available for Azure Government cloud.

Creating alert rules

Configuring Resource Health alerts for an ARO cluster requires an alert rule. Alert rules define the conditions in which alert signals are generated.

1. In the Azure portal, go to the ARO cluster for which you want to configure alerts.
2. Select **Resource health**, then select **Add resource health alert**.



3. Enter all applicable parameters for the alert rule in the various tabs of the window, including an **Alert rule name** in the **Details** tab.
4. Select **Review + Create**.

Cluster alert notifications

When Azure Monitor detects a signal related to the cluster, it generates an alert. For detailed instructions on using and creating alert rules, see What are Azure Monitor alerts?

View cluster alerts in Azure portal

You can view the status of the cluster at any time from the Azure portal. Go the applicable cluster in the portal and select **Resource health** to see if the cluster is

available or unavailable and any events associated with it. For more information, see Resource Health overview.

You can also view the alert rule you created for the cluster. Viewing the alert rule in the portal allows you to see any alerts fired against the rule.

Feedback

Enable monitoring for Azure Kubernetes Service (AKS) clusters

As described in Kubernetes monitoring in Azure Monitor, multiple features of Azure Monitor work together to provide complete monitoring of your Azure Kubernetes Service (AKS) clusters. This article describes how to enable the following features for AKS clusters:

- Prometheus metrics
- Managed Grafana
- Container logging
- Control plane logs

Prerequisites

- You need at least Contributor access to the cluster for onboarding.
- To link an Azure Monitor Workspace with an existing Managed Grafana workspace as part of onboarding, you need either Owner access or at least Contributor and User Access Administrator roles.
- You need Monitoring Reader or Monitoring Contributor roles to view data after monitoring is enabled.

> **Important**
>
> If your clusters connect to the Azure Monitor workspace or Log Analytics workspace by using Azure private link, see **Enable private link for monitoring virtual machines and Kubernetes clusters in Azure Monitor**.

Create workspaces

The following table describes the workspaces that are required to support the Azure Monitor features enabled in this article. If you don't already have an existing workspace of each type, you can create them as part of the onboarding process. See Design a Log Analytics workspace architecture for guidance on how many workspaces to create and where they should be placed.

Expand table

Feature	Workspace	Notes
Managed Prometheus	Azure Monitor workspace	If you don't specify an existing Azure Monitor workspace when onboarding, the default workspace for the resource group is used. If a default workspace doesn't already exist in the cluster's region, one with a name in the format `DefaultAzureMonitorWorkspace-<mapped_region>` is created in a resource group with the name `DefaultRG-<cluster_region>`. `Contributor` permission is enough for enabling the addon to send data to the Azure Monitor workspace. You need `Owner` level permission to link your Azure Monitor Workspace to view metrics in Azure Managed Grafana. This is required because the user executing the onboarding step, needs to be able to give the Azure Managed Grafana System Identity `Monitoring Reader` role on the Azure Monitor Workspace to query the metrics.
Container logging Control plane logs Container insights	Log Analytics workspace	You can attach a cluster to a Log Analytics workspace in a different Azure subscription in the same Microsoft Entra tenant, but you must use the Azure CLI or an Azure Resource Manager template. You can't currently perform this configuration with the Azure portal. If you're connecting an existing cluster to a Log Analytics workspace in another subscription, the *Microsoft.ContainerService* resource provider must be registered in the subscription with the Log Analytics workspace. For more information, see Register resource provider. If you don't specify an existing Log Analytics workspace, the default workspace for the resource group is used. If a default workspace doesn't already exist in the cluster's region, one is created with a name in the format `DefaultWorkspace-<GUID>-<Region>`. For a list of the supported mapping pairs to use for the default workspace, see Region mappings supported by Container insights. See Configure Azure Monitor with Network Security Perimeter for guidance on how to configure the workspace with network security perimeter.
Managed Grafana	Azure Managed Grafana workspace	Link your Grafana workspace to your Azure Monitor workspace to make the Prometheus metrics collected from your cluster available to Grafana dashboards.

Prometheus metrics and container insights

> **Note**
>
> Using Application insights to monitor the applications running on your AKS cluster by using the OpenTelemetry Protocol (OTLP) for instrumentation and data collection is now in

public preview. See Monitor AKS applications with OpenTelemetry Protocol (OTLP) Limited Preview.

When you enable Prometheus and container insights on a cluster, a containerized version of the Azure Monitor agent is installed in the cluster. You can configure these features at the same time on a new or existing cluster, or enable each feature separately.

Enable Managed Grafana for your cluster at the same time that you enable scraping of Prometheus metrics. See Link a Grafana workspace for options to connect your Azure Monitor workspace and Azure Managed Grafana workspace.

Prerequisites

Azure CLI

- The cluster must use managed identity authentication.
- The following resource providers must be registered in the subscription of the cluster and the Azure Monitor workspace:
  - Microsoft.ContainerService
  - Microsoft.Insights
  - Microsoft.AlertsManagement
  - Microsoft.Monitor
- The following resource providers must be registered in the subscription of the Grafana workspace subscription:
  - Microsoft.Dashboard
- Managed identity authentication is default in CLI version 2.49.0 or higher.
- The `aks-preview` extension must be uninstalled from AKS clusters using the command `az extension remove --name aks-preview`.

Enable Prometheus metrics on an AKS cluster

Azure CLI

Enable Prometheus metrics on an AKS cluster using the `--enable-azure-monitor-metrics` option with the az aks create command for a new cluster or the az aks update command for an existing cluster. This option uses the configuration described in Default Prometheus

metrics configuration in Azure Monitor. To modify this configuration, see Customize scraping of Prometheus metrics in Azure Monitor managed service for Prometheus.

Example commands:

Azure CLI

```
### Use default Azure Monitor workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group>

### Use existing Azure Monitor workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --azure-monitor-workspace-resource-id
<workspace-name-resource-id>

### Use an existing Azure Monitor workspace and link with an existing Grafana
workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --azure-monitor-workspace-resource-id
<azure-monitor-workspace-name-resource-id> --grafana-resource-id  <grafana-
workspace-name-resource-id>

### Use optional parameters
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --ksm-metric-labels-allow-list
"namespaces=[k8s-label-1,k8s-label-n]" --ksm-metric-annotations-allow-list
"pods=[k8s-annotation-1,k8s-annotation-n]"
```

Optional parameters

The example commands allow the following optional parameters. The parameter name is different for each, but their use is the same.

Expand table

Parameter	Name and description
Annotation keys	`--ksm-metric-annotations-allow-list` Comma-separated list of Kubernetes annotations keys used in the resource's `kube_resource_annotations` metric. For example, kube_pod_annotations is the annotations metric for the pods resource. By default, this metric contains only name and namespace labels. To include more annotations, provide a list of resource names in their plural form and Kubernetes annotation keys that you want to allow for them. A single `*` can be provided for each resource to allow any annotations, but this has severe performance implications. For example, `pods=[kubernetes.io/team,...],namespaces=[kubernetes.io/team],...`.

Parameter	Name and description
Label keys	`--ksm-metric-labels-allow-list` Comma-separated list of more Kubernetes label keys that is used in the resource's kube_resource_labels metric kube_resource_labels metric. For example, kube_pod_labels is the labels metric for the pods resource. By default this metric contains only name and namespace labels. To include more labels, provide a list of resource names in their plural form and Kubernetes label keys that you want to allow for them A single `*` can be provided for each resource to allow any labels, but i this has severe performance implications. For example, `pods=[app],namespaces=[k8s-label-1,k8s-label-n,...],...`.
Recording rules	`--enable-windows-recording-rules` Lets you enable the recording rule groups required for proper functioning of the Windows dashboards.

> **Note**
>
> The parameters set using `--ksm-metric-annotations-allow-list` and `--ksm-metric-labels-allow-list` can be overridden or alternatively set using the ama-metrics-settings-configmap.

Enable container insights and logging on an AKS cluster

Azure CLI

Enable container insights and container logging on an AKS cluster using the `--addon monitoring` option with the az aks create command for a new cluster or the az aks enable-addon command to update an existing cluster.

Example commands:

Azure CLI

```
### Use default Log Analytics workspace
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
<cluster-resource-group-name>

### Use existing Log Analytics workspace
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
```

```
<cluster-resource-group-name> --workspace-resource-id <workspace-resource-id>

### Use custom log configuration file
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
<cluster-resource-group-name> --workspace-resource-id <workspace-resource-id> -
-data-collection-settings dataCollectionSettings.json
```

Log configuration file

To customize log collection settings for the cluster, you can provide the configuration as a JSON file using the following format. If you don't provide a configuration file, the default settings identified in the table are used.

JSON

```
{
  "interval": "1m",
  "namespaceFilteringMode": "Include",
  "namespaces": ["kube-system"],
  "enableContainerLogV2": true,
  "streams": ["Microsoft-Perf", "Microsoft-ContainerLogV2"]
}
```

The following table describes each of the settings in the log configuration file:

Expand table

Name	Description
`interval`	Determines how often the agent collects data. Valid values are *1m - 30m* in *1m* intervals If the value is outside the allowed range, then it defaults to *1m*. Default: 1m.
`namespaceFilteringMode`	*Include*: Collects only data from the values in the *namespaces* field. *Exclude*: Collects data from all namespaces except for the values in the *namespaces* field. *Off*: Ignores any *namespace* selections and collect data on all namespaces. Default: Off
`namespaces`	Array of comma separated Kubernetes namespaces to collect inventory and perf data based on the *namespaceFilteringMode*. For example, *namespaces = ["kube-system", "default"]* with an *Include* setting collects only these two namespaces. With an *Exclude* setting, the agent collects data from all other namespaces except for *kube-system* and *default*. With an *Off* setting, the agent collects data from all namespaces

Name	Description
	including *kube-system* and *default*. Invalid and unrecognized namespaces are ignored. None.
`enableContainerLogV2`	Boolean flag to enable ContainerLogV2 schema. If set to true, the stdout/stderr Logs are ingested to ContainerLogV2 table. If not, the container logs are ingested to **ContainerLog** table, unless otherwise specified in the ConfigMap. When specifying the individual streams, you must include the corresponding table for ContainerLog or ContainerLogV2. Default: True
`streams`	An array of table streams to collect. See Stream values for a list of the valid streams and their corresponding tables. Default: Microsoft-ContainerInsights-Group-Default

Stream values

When you specify the tables to collect using CLI or BICEP/ARM, you specify stream names that correspond to particular tables in the Log Analytics workspace. The following table lists the stream names and their corresponding table.

> **Note**
>
> If you're familiar with the structure of a data collection rule, the stream names in this table are specified in the Data flows section of the DCR.

Expand table

Stream	Container insights table
Microsoft-ContainerInventory	ContainerInventory
Microsoft-ContainerLog	ContainerLog
Microsoft-ContainerLogV2	ContainerLogV2
Microsoft-ContainerLogV2-HighScale	ContainerLogV2 (High scale mode)[1]
Microsoft-ContainerNodeInventory	ContainerNodeInventory

Stream	Container insights table
Microsoft-InsightsMetrics	InsightsMetrics
Microsoft-KubeEvents	KubeEvents
Microsoft-KubeMonAgentEvents	KubeMonAgentEvents
Microsoft-KubeNodeInventory	KubeNodeInventory
Microsoft-KubePodInventory	KubePodInventory
Microsoft-KubePVInventory	KubePVInventory
Microsoft-KubeServices	KubeServices
Microsoft-Perf	Perf
Microsoft-ContainerInsights-Group-Default	Group stream that includes all of the above streams.[2]

[1] Don't use both Microsoft-ContainerLogV2 and Microsoft-ContainerLogV2-HighScale together. This will result in duplicate data. [2] Use the group stream as a shorthand to specifying all the individual streams. If you want to collect a specific set of streams then specify each stream individually instead of using the group stream.

Applicable tables and metrics

The settings for **collection frequency** and **namespace filtering** don't apply to all log data. The following tables list the tables in the Log Analytics workspace along with the settings that apply to each.

Expand table

Table name	Interval?	Namespaces?	Remarks
ContainerInventory	Yes	Yes	
ContainerNodeInventory	Yes	No	Data collection setting for namespaces isn't applicable since Kubernetes Node isn't a namespace scoped resource
KubeNodeInventory	Yes	No	Data collection setting for namespaces isn't applicable Kubernetes Node isn't a namespace scoped resource
KubePodInventory	Yes	Yes	
KubePVInventory	Yes	Yes	

Table name	Interval?	Namespaces?	Remarks
KubeServices	Yes	Yes	
KubeEvents	No	Yes	Data collection setting for interval isn't applicable for the Kubernetes Events
Perf	Yes	Yes	Data collection setting for namespaces isn't applicable for the Kubernetes Node related metrics since the Kubernetes Node isn't a namespace scoped object.
InsightsMetrics	Yes	Yes	Data collection settings are only applicable for the metrics collecting the following namespaces: `container.azm.ms/kubestate`, `container.azm.ms/pv`, and `container.azm.ms/gpu`.

> **Note**
>
> Namespace filtering doesn't apply to ama-logs agent records. As a result, even if the kube-system namespace is listed among excluded namespaces, records associated to ama-logs agent container are still ingested.

Expand table

Metric namespace	Interval?	Namespaces?	Remarks
Insights.container/nodes	Yes	No	Node isn't a namespace scoped resource
Insights.container/pods	Yes	Yes	
Insights.container/containers	Yes	Yes	
Insights.container/persistentvolumes	Yes	Yes	

Special scenarios

Use the following resources for configuration requirements for particular scenarios:

- If you're using private link, see Enable private link for Kubernetes monitoring in Azure Monitor.
- To enable container logging with network security perimeter see Configure Azure Monitor with Network Security Perimeter to configure your Log Analytics workspace.

- To enable high scale mode, follow the onboarding process at Enable high scale mode for Monitoring add-on. You must also update the ConfigMap as described in Update ConfigMap, and the DCR stream needs to be changed from `Microsoft-ContainerLogV2` to `Microsoft-ContainerLogV2-HighScale`.

Enable control plane logs on an AKS cluster

Control plane logs are implemented as resource logs in Azure Monitor. To collect these logs, create a diagnostic setting for the cluster. Send them to the same Log Analytics workspace as your container logs.

Azure CLI

Use the az monitor diagnostic-settings create command to create a diagnostic setting with the Azure CLI. See the documentation for this command for descriptions of its parameters.

The following example creates a diagnostic setting that sends all Kubernetes categories to a Log Analytics workspace. This includes resource-specific mode to send the logs to specific tables listed in Supported resource logs for Microsoft.ContainerService/fleets.

Azure CLI

```
az monitor diagnostic-settings create \
--name 'Collect control plane logs' \
--resource  /subscriptions/<subscription ID>/resourceGroups/<resource group name>/providers/Microsoft.ContainerService/managedClusters/<cluster-name> \
--workspace /subscriptions/<subscription ID>/resourcegroups/<resource group name>/providers/microsoft.operationalinsights/workspaces/<log analytics workspace name> \
--logs '[{"category": "karpenter-events","enabled": true},{"category": "kube-audit","enabled": true},
{"category": "kube-apiserver","enabled": true},{"category": "kube-audit-admin","enabled": true},{"category": "kube-controller-manager","enabled": true},{"category": "kube-scheduler","enabled": true},{"category": "cluster-autoscaler","enabled": true},{"category": "cloud-controller-manager","enabled": true},{"category": "guard","enabled": true},{"category": "csi-azuredisk-controller","enabled": true},{"category": "csi-azurefile-controller","enabled": true},{"category": "csi-snapshot-controller","enabled": true},{"category": "fleet-member-agent","enabled": true},{"category": "fleet-member-net-controller-manager","enabled": true},{"category": "fleet-mcs-controller-manager","enabled": true}]'
--metrics '[{"category": "AllMetrics","enabled": true}]' \
--export-to-resource-specific true
```

Enable Windows metrics (Preview)

Windows metric collection is enabled for AKS clusters as of version 6.4.0-main-02-22-2023-3ee44b9e of the Managed Prometheus addon container. Onboarding to the Azure Monitor Metrics add-on enables the Windows DaemonSet pods to start running on your node pools. Both Windows Server 2019 and Windows Server 2022 are supported. Follow these steps to enable the pods to collect metrics from your Windows node pools.

> **Note**
>
> There's no CPU/Memory limit in `windows-exporter-daemonset.yaml` so it might overprovision the Windows nodes. For details see Resource reservation
>
> As you deploy workloads, set resource memory and CPU limits on containers. This also subtracts from NodeAllocatable and helps the cluster-wide scheduler in determining which pods to place on which nodes. Scheduling pods without limits might overprovision the Windows nodes and in extreme cases can cause the nodes to become unhealthy.

Install Windows exporter

Manually install windows-exporter on AKS nodes to access Windows metrics by deploying the windows-exporter-daemonset YAML file. Enable the following collectors. For more collectors, see Prometheus exporter for Windows metrics.

- `[defaults]`
  - `container`
  - `memory`
  - `process`
  - `cpu_info`

Deploy the windows-exporter-daemonset YAML file. If there are any taints applied in the node, you need to apply the appropriate tolerations.

Bash

```
kubectl apply -f windows-exporter-daemonset.yaml
```

Enable Windows metrics

Set the `windowsexporter` and `windowskubeproxy` Booleans to `true` in your metrics settings ConfigMap and apply it to the cluster. See Customize collection of Prometheus metrics from your Kubernetes cluster using ConfigMap.

Enable recording rules

Enable the recording rules that are required for the out-of-the-box dashboards:

- If onboarding using CLI, include the option `--enable-windows-recording-rules`.
- If onboarding using an ARM template, Bicep, or Azure Policy, set `enableWindowsRecordingRules` to `true` in the parameters file.
- If the cluster is already onboarded, use this ARM template and this parameter file to create the rule groups. This adds the required recording rules and isn't an ARM operation on the cluster and doesn't impact current monitoring state of the cluster.

Verify deployment

Use the kubectl command line tool to verify that the agent is deployed properly.

Managed Prometheus

Verify that the DaemonSet was deployed properly on the Linux node pools

Bash

```
kubectl get ds ama-metrics-node --namespace=kube-system
```

The number of pods should be equal to the number of Linux nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-metrics-node --namespace=kube-system
NAME               DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
ama-metrics-node   1         1         1       1            1           <none>
10h
```

Verify that Windows nodes were deployed properly

Bash

```
kubectl get ds ama-metrics-win-node --namespace=kube-system
```

The number of pods should be equal to the number of Windows nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-metrics-node --namespace=kube-system
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
ama-metrics-win-node   3         3         3       3            3           <none>
10h
```

Verify that the two ReplicaSets were deployed for Prometheus

Bash

```
kubectl get rs --namespace=kube-system
```

The output should resemble the following example:

Output

```
User@aksuser:~$kubectl get rs --namespace=kube-system
NAME                            DESIRED   CURRENT   READY   AGE
ama-metrics-5c974985b8          1         1         1       11h
ama-metrics-ksm-5fcf8dffcd      1         1         1       11h
```

Container insights and logging

Verify that the DaemonSets were deployed properly on the Linux node pools

Bash

```
kubectl get ds ama-logs --namespace=kube-system
```

The number of pods should be equal to the number of Linux nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-logs --namespace=kube-system
NAME       DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR
AGE
```

```
ama-logs    2         2         2         2            2           <none>
1d
```

Verify that Windows nodes were deployed properly

Bash

```
kubectl get ds ama-logs-windows --namespace=kube-system
```

The number of pods should be equal to the number of Windows nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-logs-windows --namespace=kube-system
NAME                  DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE
SELECTOR     AGE
ama-logs-windows           2         2         2         2            2
<none>            1d
```

Verify deployment of the container logging solution

Bash

```
kubectl get deployment ama-logs-rs --namespace=kube-system
```

The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get deployment ama-logs-rs --namespace=kube-system
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
ama-logs-rs   1/1     1            1           24d
```

View configuration with CLI

Use the `aks show` command to find out whether the solution is enabled, the Log Analytics workspace resource ID, and summary information about the cluster.

Azure CLI

```
az aks show --resource-group <resourceGroupofAKSCluster> --name <nameofAksCluster>
```

The command returns JSON-formatted information about the solution. The `addonProfiles` section should include information on the `omsagent` as in the following example:

Output

```
"addonProfiles": {
    "omsagent": {
        "config": {
            "logAnalyticsWorkspaceResourceID": "/subscriptions/aaaa0a0a-bb1b-cc2c-
dd3d-eeeeee4e4e4e/resourcegroups/my-resource-
group/providers/microsoft.operationalinsights/workspaces/my-workspace",
            "useAADAuth": "true"
        },
        "enabled": true,
        "identity": null
    },
}
```

Related content

- If you experience issues attempting to onboard, review the Troubleshooting guide.
- Learn how to Analyze Kubernetes monitoring data in the Azure portal Container insights.

Last updated on 04/14/2026

Enable monitoring for Azure Kubernetes Service (AKS) clusters

As described in Kubernetes monitoring in Azure Monitor, multiple features of Azure Monitor work together to provide complete monitoring of your Azure Kubernetes Service (AKS) clusters. This article describes how to enable the following features for AKS clusters:

- Prometheus metrics
- Managed Grafana
- Container logging
- Control plane logs

Prerequisites

- You need at least Contributor access to the cluster for onboarding.
- To link an Azure Monitor Workspace with an existing Managed Grafana workspace as part of onboarding, you need either Owner access or at least Contributor and User Access Administrator roles.
- You need Monitoring Reader or Monitoring Contributor roles to view data after monitoring is enabled.

> **Important**
>
> If your clusters connect to the Azure Monitor workspace or Log Analytics workspace by using Azure private link, see **Enable private link for monitoring virtual machines and Kubernetes clusters in Azure Monitor**.

Create workspaces

The following table describes the workspaces that are required to support the Azure Monitor features enabled in this article. If you don't already have an existing workspace of each type, you can create them as part of the onboarding process. See Design a Log Analytics workspace architecture for guidance on how many workspaces to create and where they should be placed.

Expand table

Feature	Workspace	Notes
Managed Prometheus	Azure Monitor workspace	If you don't specify an existing Azure Monitor workspace when onboarding, the default workspace for the resource group is used. If a default workspace doesn't already exist in the cluster's region, one with a name in the format `DefaultAzureMonitorWorkspace-<mapped_region>` is created in a resource group with the name `DefaultRG-<cluster_region>`. `Contributor` permission is enough for enabling the addon to send data to the Azure Monitor workspace. You need `Owner` level permission to link your Azure Monitor Workspace to view metrics in Azure Managed Grafana. This is required because the user executing the onboarding step, needs to be able to give the Azure Managed Grafana System Identity `Monitoring Reader` role on the Azure Monitor Workspace to query the metrics.
Container logging Control plane logs Container insights	Log Analytics workspace	You can attach a cluster to a Log Analytics workspace in a different Azure subscription in the same Microsoft Entra tenant, but you must use the Azure CLI or an Azure Resource Manager template. You can't currently perform this configuration with the Azure portal. If you're connecting an existing cluster to a Log Analytics workspace in another subscription, the *Microsoft.ContainerService* resource provider must be registered in the subscription with the Log Analytics workspace. For more information, see Register resource provider. If you don't specify an existing Log Analytics workspace, the default workspace for the resource group is used. If a default workspace doesn't already exist in the cluster's region, one is created with a name in the format `DefaultWorkspace-<GUID>-<Region>`. For a list of the supported mapping pairs to use for the default workspace, see Region mappings supported by Container insights. See Configure Azure Monitor with Network Security Perimeter for guidance on how to configure the workspace with network security perimeter.
Managed Grafana	Azure Managed Grafana workspace	Link your Grafana workspace to your Azure Monitor workspace to make the Prometheus metrics collected from your cluster available to Grafana dashboards.

Prometheus metrics and container insights

> **Note**
>
> Using Application insights to monitor the applications running on your AKS cluster by using the OpenTelemetry Protocol (OTLP) for instrumentation and data collection is now in

public preview. See Monitor AKS applications with OpenTelemetry Protocol (OTLP) Limited Preview.

When you enable Prometheus and container insights on a cluster, a containerized version of the Azure Monitor agent is installed in the cluster. You can configure these features at the same time on a new or existing cluster, or enable each feature separately.

Enable Managed Grafana for your cluster at the same time that you enable scraping of Prometheus metrics. See Link a Grafana workspace for options to connect your Azure Monitor workspace and Azure Managed Grafana workspace.

Prerequisites

Azure CLI

- The cluster must use managed identity authentication.
- The following resource providers must be registered in the subscription of the cluster and the Azure Monitor workspace:
  - Microsoft.ContainerService
  - Microsoft.Insights
  - Microsoft.AlertsManagement
  - Microsoft.Monitor
- The following resource providers must be registered in the subscription of the Grafana workspace subscription:
  - Microsoft.Dashboard
- Managed identity authentication is default in CLI version 2.49.0 or higher.
- The `aks-preview` extension must be uninstalled from AKS clusters using the command `az extension remove --name aks-preview`.

Enable Prometheus metrics on an AKS cluster

Azure CLI

Enable Prometheus metrics on an AKS cluster using the `--enable-azure-monitor-metrics` option with the az aks create command for a new cluster or the az aks update command for an existing cluster. This option uses the configuration described in Default Prometheus

metrics configuration in Azure Monitor. To modify this configuration, see Customize scraping of Prometheus metrics in Azure Monitor managed service for Prometheus.

Example commands:

Azure CLI

```
### Use default Azure Monitor workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group>

### Use existing Azure Monitor workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --azure-monitor-workspace-resource-id
<workspace-name-resource-id>

### Use an existing Azure Monitor workspace and link with an existing Grafana
workspace
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --azure-monitor-workspace-resource-id
<azure-monitor-workspace-name-resource-id> --grafana-resource-id  <grafana-
workspace-name-resource-id>

### Use optional parameters
az aks create/update --enable-azure-monitor-metrics --name <cluster-name> --
resource-group <cluster-resource-group> --ksm-metric-labels-allow-list
"namespaces=[k8s-label-1,k8s-label-n]" --ksm-metric-annotations-allow-list
"pods=[k8s-annotation-1,k8s-annotation-n]"
```

Optional parameters

The example commands allow the following optional parameters. The parameter name is different for each, but their use is the same.

Expand table

Parameter	Name and description
Annotation keys	`--ksm-metric-annotations-allow-list` Comma-separated list of Kubernetes annotations keys used in the resource's `kube_resource_annotations` metric. For example, kube_pod_annotations is the annotations metric for the pods resource. By default, this metric contains only name and namespace labels. To include more annotations, provide a list of resource names in their plural form and Kubernetes annotation keys that you want to allow for them. A single `*` can be provided for each resource to allow any annotations, but this has severe performance implications. For example, `pods=[kubernetes.io/team,...],namespaces=[kubernetes.io/team],...`.

Parameter	Name and description
Label keys	`--ksm-metric-labels-allow-list` Comma-separated list of more Kubernetes label keys that is used in the resource's kube_resource_labels metric kube_resource_labels metric. For example, kube_pod_labels is the labels metric for the pods resource. By default this metric contains only name and namespace labels. To include more labels, provide a list of resource names in their plural form and Kubernetes label keys that you want to allow for them A single `*` can be provided for each resource to allow any labels, but i this has severe performance implications. For example, `pods=[app],namespaces=[k8s-label-1,k8s-label-n,...],...`.
Recording rules	`--enable-windows-recording-rules` Lets you enable the recording rule groups required for proper functioning of the Windows dashboards.

> **Note**
>
> The parameters set using `--ksm-metric-annotations-allow-list` and `--ksm-metric-labels-allow-list` can be overridden or alternatively set using the ama-metrics-settings-configmap.

Enable container insights and logging on an AKS cluster

Azure CLI

Enable container insights and container logging on an AKS cluster using the `--addon monitoring` option with the az aks create command for a new cluster or the az aks enable-addon command to update an existing cluster.

Example commands:

Azure CLI

```
### Use default Log Analytics workspace
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
<cluster-resource-group-name>

### Use existing Log Analytics workspace
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
```

```
<cluster-resource-group-name> --workspace-resource-id <workspace-resource-id>

### Use custom log configuration file
az aks enable-addons --addon monitoring --name <cluster-name> --resource-group
<cluster-resource-group-name> --workspace-resource-id <workspace-resource-id> -
-data-collection-settings dataCollectionSettings.json
```

Log configuration file

To customize log collection settings for the cluster, you can provide the configuration as a JSON file using the following format. If you don't provide a configuration file, the default settings identified in the table are used.

JSON

```
{
  "interval": "1m",
  "namespaceFilteringMode": "Include",
  "namespaces": ["kube-system"],
  "enableContainerLogV2": true,
  "streams": ["Microsoft-Perf", "Microsoft-ContainerLogV2"]
}
```

The following table describes each of the settings in the log configuration file:

Expand table

Name	Description
`interval`	Determines how often the agent collects data. Valid values are *1m - 30m* in *1m* intervals If the value is outside the allowed range, then it defaults to *1m*. Default: 1m.
`namespaceFilteringMode`	*Include*: Collects only data from the values in the *namespaces* field. *Exclude*: Collects data from all namespaces except for the values in the *namespaces* field. *Off*: Ignores any *namespace* selections and collect data on all namespaces. Default: Off
`namespaces`	Array of comma separated Kubernetes namespaces to collect inventory and perf data based on the *namespaceFilteringMode*. For example, *namespaces = ["kube-system", "default"]* with an *Include* setting collects only these two namespaces. With an *Exclude* setting, the agent collects data from all other namespaces except for *kube-system* and *default*. With an *Off* setting, the agent collects data from all namespaces

Name	Description
	including *kube-system* and *default*. Invalid and unrecognized namespaces are ignored. None.
`enableContainerLogV2`	Boolean flag to enable ContainerLogV2 schema. If set to true, the stdout/stderr Logs are ingested to ContainerLogV2 table. If not, the container logs are ingested to **ContainerLog** table, unless otherwise specified in the ConfigMap. When specifying the individual streams, you must include the corresponding table for ContainerLog or ContainerLogV2. Default: True
`streams`	An array of table streams to collect. See Stream values for a list of the valid streams and their corresponding tables. Default: Microsoft-ContainerInsights-Group-Default

Stream values

When you specify the tables to collect using CLI or BICEP/ARM, you specify stream names that correspond to particular tables in the Log Analytics workspace. The following table lists the stream names and their corresponding table.

> **Note**
>
> If you're familiar with the structure of a data collection rule, the stream names in this table are specified in the Data flows section of the DCR.

Expand table

Stream	Container insights table
Microsoft-ContainerInventory	ContainerInventory
Microsoft-ContainerLog	ContainerLog
Microsoft-ContainerLogV2	ContainerLogV2
Microsoft-ContainerLogV2-HighScale	ContainerLogV2 (High scale mode)[1]
Microsoft-ContainerNodeInventory	ContainerNodeInventory

Stream	Container insights table
Microsoft-InsightsMetrics	InsightsMetrics
Microsoft-KubeEvents	KubeEvents
Microsoft-KubeMonAgentEvents	KubeMonAgentEvents
Microsoft-KubeNodeInventory	KubeNodeInventory
Microsoft-KubePodInventory	KubePodInventory
Microsoft-KubePVInventory	KubePVInventory
Microsoft-KubeServices	KubeServices
Microsoft-Perf	Perf
Microsoft-ContainerInsights-Group-Default	Group stream that includes all of the above streams.[2]

[1] Don't use both Microsoft-ContainerLogV2 and Microsoft-ContainerLogV2-HighScale together. This will result in duplicate data. [2] Use the group stream as a shorthand to specifying all the individual streams. If you want to collect a specific set of streams then specify each stream individually instead of using the group stream.

Applicable tables and metrics

The settings for **collection frequency** and **namespace filtering** don't apply to all log data. The following tables list the tables in the Log Analytics workspace along with the settings that apply to each.

Expand table

Table name	Interval?	Namespaces?	Remarks
ContainerInventory	Yes	Yes	
ContainerNodeInventory	Yes	No	Data collection setting for namespaces isn't applicable since Kubernetes Node isn't a namespace scoped resource
KubeNodeInventory	Yes	No	Data collection setting for namespaces isn't applicable Kubernetes Node isn't a namespace scoped resource
KubePodInventory	Yes	Yes	
KubePVInventory	Yes	Yes	

Table name	Interval?	Namespaces?	Remarks
KubeServices	Yes	Yes	
KubeEvents	No	Yes	Data collection setting for interval isn't applicable for the Kubernetes Events
Perf	Yes	Yes	Data collection setting for namespaces isn't applicable for the Kubernetes Node related metrics since the Kubernetes Node isn't a namespace scoped object.
InsightsMetrics	Yes	Yes	Data collection settings are only applicable for the metrics collecting the following namespaces: `container.azm.ms/kubestate`, `container.azm.ms/pv`, and `container.azm.ms/gpu`.

> **Note**
>
> Namespace filtering doesn't apply to ama-logs agent records. As a result, even if the kube-system namespace is listed among excluded namespaces, records associated to ama-logs agent container are still ingested.

Expand table

Metric namespace	Interval?	Namespaces?	Remarks
Insights.container/nodes	Yes	No	Node isn't a namespace scoped resource
Insights.container/pods	Yes	Yes	
Insights.container/containers	Yes	Yes	
Insights.container/persistentvolumes	Yes	Yes	

Special scenarios

Use the following resources for configuration requirements for particular scenarios:

- If you're using private link, see Enable private link for Kubernetes monitoring in Azure Monitor.
- To enable container logging with network security perimeter see Configure Azure Monitor with Network Security Perimeter to configure your Log Analytics workspace.

- To enable high scale mode, follow the onboarding process at Enable high scale mode for Monitoring add-on. You must also update the ConfigMap as described in Update ConfigMap, and the DCR stream needs to be changed from `Microsoft-ContainerLogV2` to `Microsoft-ContainerLogV2-HighScale`.

Enable control plane logs on an AKS cluster

Control plane logs are implemented as resource logs in Azure Monitor. To collect these logs, create a diagnostic setting for the cluster. Send them to the same Log Analytics workspace as your container logs.

Azure CLI

Use the az monitor diagnostic-settings create command to create a diagnostic setting with the Azure CLI. See the documentation for this command for descriptions of its parameters.

The following example creates a diagnostic setting that sends all Kubernetes categories to a Log Analytics workspace. This includes resource-specific mode to send the logs to specific tables listed in Supported resource logs for Microsoft.ContainerService/fleets.

Azure CLI

```
az monitor diagnostic-settings create \
--name 'Collect control plane logs' \
--resource  /subscriptions/<subscription ID>/resourceGroups/<resource group name>/providers/Microsoft.ContainerService/managedClusters/<cluster-name> \
--workspace /subscriptions/<subscription ID>/resourcegroups/<resource group name>/providers/microsoft.operationalinsights/workspaces/<log analytics workspace name> \
--logs '[{"category": "karpenter-events","enabled": true},{"category": "kube-audit","enabled": true},
{"category": "kube-apiserver","enabled": true},{"category": "kube-audit-admin","enabled": true},{"category": "kube-controller-manager","enabled": true},{"category": "kube-scheduler","enabled": true},{"category": "cluster-autoscaler","enabled": true},{"category": "cloud-controller-manager","enabled": true},{"category": "guard","enabled": true},{"category": "csi-azuredisk-controller","enabled": true},{"category": "csi-azurefile-controller","enabled": true},{"category": "csi-snapshot-controller","enabled": true},{"category": "fleet-member-agent","enabled": true},{"category": "fleet-member-net-controller-manager","enabled": true},{"category": "fleet-mcs-controller-manager","enabled": true}]'
--metrics '[{"category": "AllMetrics","enabled": true}]' \
--export-to-resource-specific true
```

Enable Windows metrics (Preview)

Windows metric collection is enabled for AKS clusters as of version 6.4.0-main-02-22-2023-3ee44b9e of the Managed Prometheus addon container. Onboarding to the Azure Monitor Metrics add-on enables the Windows DaemonSet pods to start running on your node pools. Both Windows Server 2019 and Windows Server 2022 are supported. Follow these steps to enable the pods to collect metrics from your Windows node pools.

> **Note**
>
> There's no CPU/Memory limit in `windows-exporter-daemonset.yaml` so it might overprovision the Windows nodes. For details see Resource reservation
>
> As you deploy workloads, set resource memory and CPU limits on containers. This also subtracts from NodeAllocatable and helps the cluster-wide scheduler in determining which pods to place on which nodes. Scheduling pods without limits might overprovision the Windows nodes and in extreme cases can cause the nodes to become unhealthy.

Install Windows exporter

Manually install windows-exporter on AKS nodes to access Windows metrics by deploying the windows-exporter-daemonset YAML file. Enable the following collectors. For more collectors, see Prometheus exporter for Windows metrics.

- `[defaults]`
  - `container`
  - `memory`
  - `process`
  - `cpu_info`

Deploy the windows-exporter-daemonset YAML file. If there are any taints applied in the node, you need to apply the appropriate tolerations.

Bash

```
kubectl apply -f windows-exporter-daemonset.yaml
```

Enable Windows metrics

Set the `windowsexporter` and `windowskubeproxy` Booleans to `true` in your metrics settings ConfigMap and apply it to the cluster. See Customize collection of Prometheus metrics from your Kubernetes cluster using ConfigMap.

Enable recording rules

Enable the recording rules that are required for the out-of-the-box dashboards:

- If onboarding using CLI, include the option `--enable-windows-recording-rules`.
- If onboarding using an ARM template, Bicep, or Azure Policy, set `enableWindowsRecordingRules` to `true` in the parameters file.
- If the cluster is already onboarded, use this ARM template and this parameter file to create the rule groups. This adds the required recording rules and isn't an ARM operation on the cluster and doesn't impact current monitoring state of the cluster.

Verify deployment

Use the kubectl command line tool to verify that the agent is deployed properly.

Managed Prometheus

Verify that the DaemonSet was deployed properly on the Linux node pools

Bash

```
kubectl get ds ama-metrics-node --namespace=kube-system
```

The number of pods should be equal to the number of Linux nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-metrics-node --namespace=kube-system
NAME               DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
ama-metrics-node   1         1         1       1            1           <none>
10h
```

Verify that Windows nodes were deployed properly

Bash

```
kubectl get ds ama-metrics-win-node --namespace=kube-system
```

The number of pods should be equal to the number of Windows nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-metrics-node --namespace=kube-system
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
ama-metrics-win-node   3         3         3       3            3           <none>
10h
```

Verify that the two ReplicaSets were deployed for Prometheus

Bash

```
kubectl get rs --namespace=kube-system
```

The output should resemble the following example:

Output

```
User@aksuser:~$kubectl get rs --namespace=kube-system
NAME                            DESIRED   CURRENT   READY   AGE
ama-metrics-5c974985b8          1         1         1       11h
ama-metrics-ksm-5fcf8dffcd      1         1         1       11h
```

Container insights and logging

Verify that the DaemonSets were deployed properly on the Linux node pools

Bash

```
kubectl get ds ama-logs --namespace=kube-system
```

The number of pods should be equal to the number of Linux nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-logs --namespace=kube-system
NAME       DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR
AGE
```

```
ama-logs    2         2         2         2            2           <none>
1d
```

Verify that Windows nodes were deployed properly

Bash

```
kubectl get ds ama-logs-windows --namespace=kube-system
```

The number of pods should be equal to the number of Windows nodes on the cluster. The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get ds ama-logs-windows --namespace=kube-system
NAME                  DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE
SELECTOR     AGE
ama-logs-windows           2         2         2         2            2
<none>            1d
```

Verify deployment of the container logging solution

Bash

```
kubectl get deployment ama-logs-rs --namespace=kube-system
```

The output should resemble the following example:

Output

```
User@aksuser:~$ kubectl get deployment ama-logs-rs --namespace=kube-system
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
ama-logs-rs   1/1     1            1           24d
```

View configuration with CLI

Use the `aks show` command to find out whether the solution is enabled, the Log Analytics workspace resource ID, and summary information about the cluster.

Azure CLI

```
az aks show --resource-group <resourceGroupofAKSCluster> --name <nameofAksCluster>
```

The command returns JSON-formatted information about the solution. The `addonProfiles` section should include information on the `omsagent` as in the following example:

Output

```
"addonProfiles": {
    "omsagent": {
        "config": {
            "logAnalyticsWorkspaceResourceID": "/subscriptions/aaaa0a0a-bb1b-cc2c-
dd3d-eeeeee4e4e4e/resourcegroups/my-resource-
group/providers/microsoft.operationalinsights/workspaces/my-workspace",
            "useAADAuth": "true"
        },
        "enabled": true,
        "identity": null
    },
}
```

Related content

- If you experience issues attempting to onboard, review the Troubleshooting guide.
- Learn how to Analyze Kubernetes monitoring data in the Azure portal Container insights.

Last updated on 04/14/2026

Set up remote write of Prometheus monitoring data to an Azure Monitor workspace

09/05/2025

Azure Red Hat OpenShift is preinstalled with a default Prometheus server. As detailed in the Azure Red Hat OpenShift support policy, this default Prometheus server shouldn't be removed.

In some scenarios, you might want to centralize data from self-managed Prometheus clusters for long-term data retention to create a centralized view across your clusters. You can use Azure Monitor managed service for Prometheus to collect and analyze metrics at scale by using a Prometheus-compatible monitoring solution based on the Prometheus project from the Cloud Native Computing Foundation. You can use remote write to send data from Prometheus servers in your cluster to the Azure managed service.

Prerequisites

To send data from a Prometheus server by using remote write, you need:

- An Azure Red Hat OpenShift cluster. If you need to create a cluster, see Quickstart: Create an Azure Red Hat OpenShift 4 cluster.
- An Azure Monitor workspace. If you don't have a workspace, see Create an Azure Monitor workspace.
- OpenShift CLI. After you're logged into the OpenShift Web Console, select the `?` at the top right and then on **Command Line Tools**. Download the release appropriate to your computer. You can also download the latest release of the CLI appropriate to your computer.

The article assumes you used the names and locations in the examples and that you're running commands in a Bash session. If you used different names or locations, you need to adjust the instructions.

Register an application with Microsoft Entra ID

Register a Microsoft Entra ID application and create a client secret.

1. Sign in to the Azure portal.
2. In the search box, enter *Microsoft Entra ID* and select it from the list of services.
3. Select **App registrations**.

4. Select **New registration**.

    - Enter a name for the application like *demo-monitoring-app*.
    - Select *Accounts in this organizational directory only*.

5. Select **Register** to create the app.

After the app is created, the **Overview** page is displayed. Copy and save the values for the **Application (client) ID** and **Directory (tenant) ID** because you use them later in this article.

Create a client secret for your registered app.

1. Select **Certificates & secrets**.
2. From **Client secrets** select **New client secret**.

    - **Description**: Enter a description like *monitoring app secret*.
    - **Expires**: Select 90 days or custom to select a start and end date.

3. Select **Add**.
4. Copy the client secret value because you use it later in this article. The client secret value is only shown when the client secret is created.

In your Bash session, create the following variables and replace `<tenant ID>`, `<client ID>`, and `<client-secret>` with the values from the previous steps. These variables are used when you set up remote write later in this article.

Bash

```
export TENANT_ID=<tenant ID>
export CLIENT_ID=<client ID>
export CLIENT_SECRET=<client-secret>
```

For more information, see Register a Microsoft Entra app and create a service principal and create a new client secret.

Assign the Monitoring Metrics Publisher role to the application

The application must have the Monitoring Metrics Publisher role for the data collection rule that is associated with your Azure Monitor workspace.

1. In the Azure portal, enter *Monitor* in the search box and select *Monitor* from the services list.
2. Select **Settings**> **Data Collection Rules**.

3. Select the data collection rule that is associated with your Azure Monitor workspace.
4. On the **Overview** page for the data collection rule, select **Access control (IAM)**.
5. Select **Add**, and then select **Add role assignment**.
6. Select the **Monitoring Metrics Publisher** role, and then select **Next**.
7. Select **User, group, or service principal**, and then choose **Select members**. Select the application that you registered, and then choose **Select**.
8. To complete the role assignment, select **Review + assign**.

Create a secret in your Azure Red Hat OpenShift cluster

To authenticate by using a remote write endpoint, you use the OAuth 2.0 authentication method from the supported remote write authentication settings.

To begin, create a secret using the `CLIENT_ID` and `CLIENT_SECRET` variables.

YAML

```
cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: oauth2-credentials
  namespace: openshift-monitoring
stringData:
  id: "${CLIENT_ID}"
  secret: "${CLIENT_SECRET}"
EOF
```

In the YAML code, `oc apply -f -` refers to `stdin` (standard input) to accept the YAML input from the command line. The variable values from `CLIENT_ID` and `CLIENT_SECRET` are used to create the secret. You can view the secret in the cluster's web console **Workloads** > **Secrets** with the name *oauth2-credentials*.

Set up remote write

To set up remote write, you need a `ConfigMap` object named *cluster-monitoring-config*. For more information, see Creating a cluster monitoring config map.

Check if your cluster has a `ConfigMaps` object named *cluster-monitoring-config*.

Bash

```
oc -n openshift-monitoring get configmap cluster-monitoring-config
```

If *cluster-monitoring-config* doesn't exist, the following message is displayed.

Output

```
Error from server (NotFound): configmaps "cluster-monitoring-config" not found
```

If you need to create the *cluster-monitoring-config* object, copy the following YAML and save it to a file named *cluster-monitoring-config.yaml*.

YAML

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      remoteWrite:
        - url: "<INGESTION-URL>"
          oauth2:
            clientId:
              secret:
                name: oauth2-credentials
                key: id
            clientSecret:
              name: oauth2-credentials
              key: secret
            tokenUrl:
"https://login.microsoftonline.com/<TENANT_ID>/oauth2/v2.0/token"
            scopes:
              - "https://monitor.azure.com/.default"
```

- Replace `<INGESTION-URL>` with the value for **Metrics ingestion endpoint** from the **Overview** page for the Azure Monitor workspace.
- Replace `<TENANT_ID>` with the tenant ID of the service principal. You can use the command `echo TENANT_ID` to display the value you saved in the app registration.

Apply the YAML file to create the *cluster-monitoring-config* object.

Bash

```
oc apply -f cluster-monitoring-config.yaml
```

You can verify the object was created in the cluster's web console **Workloads** > **ConfigMaps** with the name *cluster-monitoring-config*.

If your cluster has a *cluster-monitoring-config* object, you can use the following command to edit the file.

```
Bash

oc edit -n openshift-monitoring configmap cluster-monitoring-config
```

Visualize metrics by using Azure Managed Grafana

You can use community Grafana dashboards to visualize the captured metrics, or you can create contextual dashboards. This example imports the Grafana dashboard *Openshift/K8 Cluster Overview*.

Create an Azure Managed Grafana workspace

Go to Quickstart: Create an Azure Managed Grafana workspace using the Azure portal and use the following values to create the workspace.

- Subscription ID: The subscription that contains your cluster and Azure Monitor workspace.
- Resource group name: Use *demo-workspace-rg* resource group that contains your Azure Monitor workspace.
- Location: *centralus*
- Name: *demo-grafana-workspace*
- Pricing plan: Standard
- Size: X1
- Grafana version: 11

Link the Azure Managed Grafana workspace

Link the Azure Managed Grafana workspace to your Azure Monitor workspace.

1. In Azure portal search box, enter *Azure Monitor workspace*.
2. Select *demo-workspace*.
3. Select **Linked Grafana workspaces**.
4. Select **+Link**.
5. Select the **Grafana workspace** *demo-grafana-workspace*.
6. Select **Link**.

For more information, see Link a Grafana workspace.

Import a Grafana dashboard

Import the dashboard *Openshift/K8 Cluster Overview* with ID 3870.

1. In Azure portal search box, enter *Azure Managed Grafana*, select it from the list of services, and select *demo-grafana-workspace*.
2. On the **Overview** page, select the **Endpoint** URL.
3. On the **Grafana** page, select **Dashboards**.
4. Select **New** > **Import**.
5. Enter *3870* in the **dashboard URL or ID** field and select **Load**. The name displayed is *Openshift/K8 Cluster Overview*
6. In the **Folder** dropdown, select *Dashboards*.
7. In the **Openshift** dropdown to select a Prometheus data source, select your Azure Monitor workspace, *Managed_Prometheus_demo-workspace*.
8. Select **Import**. The Grafana dashboard is displayed for your Azure Red Hat OpenShift cluster.

To access the dashboard, in your Azure Managed Grafana workspace, go to **Home** > **Dashboards** and then select the *Openshift/K8 Cluster Overview* dashboard.

For more information, see Import the community Grafana dashboard Openshift/K8 Cluster Overview with ID 3870 to the Grafana workspace.

Troubleshoot

For troubleshooting information, see Troubleshoot remote write.

Related content

For more information about monitoring, see Azure Monitor and Prometheus.

Configure Prometheus persistence for Azure Red Hat OpenShift

Prometheus is an open-source monitoring tool that's designed for reliability and scalability. When integrated with OpenShift, Prometheus provides insights into the health and performance of your containerized applications and the underlying cluster infrastructure.

Azure Red Hat OpenShift comes with Prometheus preinstalled as part of the OpenShift Container Platform monitoring stack for your cluster. For more information, see About OpenShift Container Platform monitoring.

Considerations

To protect your metrics and alerting data from data loss, configure cluster monitoring with persistent storage to store it in a persistent volume (PV). As a result, the data can survive pods being restarted or recreated.

> **Caution**
>
> A persistent volume that becomes full can significantly affect the ability to monitor your cluster effectively, potentially impacting your SLA. Ensure that you maintain a buffer of available storage on your persistent volumes to prevent any issues.

- **Set a proper retention size**: Ensuring that there's ample free space in the volume is imperative to the proper monitoring of the cluster. The recommendation is to set the proper retention size to ensure the allocated storage for Prometheus doesn't become full. For more information, see Retention time and size for Prometheus metrics and Modifying retention time and size for Prometheus metrics data.
- **Monitor persistent volume usage:** Monitor your persistent volume usage. You can set up notifications for alerts, for example `KubePersistentVolumeFillingUp`, to notify you when a persistent volume is approaching its capacity limit. These alerts help prevent data loss and ensure the stability of your Prometheus instance. See Sending notifications to external systems.
- **Consider data forwarding:** For longer term storage and to prevent the persistent volume from becoming overloaded, consider forwarding the data to a separate solution like Azure Monitor.

Set up persistence

To set up persistence, see Storing and recording data for core platform monitoring.

Next steps

- Learn about OpenShift Container Platform monitoring.
- Learn about Retention time and size for Prometheus metrics.
- Modify retention time and size for Prometheus metrics data so that your persistent volumes don't fill up.

Last updated on 11/25/2025

Deploy an application from source to Azure Red Hat OpenShift

Article • 02/25/2025

In this article, you deploy an application to an Azure Red Hat OpenShift cluster from source code using a source-to-image (S2I) build. Source-to-Image (S2I) is a build process for building reproducible container images from source code. S2I produces ready-to-run images by injecting source code into a container image and letting the container prepare that source code for execution. You can have OpenShift build an application from source to deploy it, so you don't have to construct a container by hand with every change. OpenShift can then build and deploy new versions automatically when notified of source code changes.

Before you begin

> **Note**
>
> This article assumes you have set up a pull secret. If you do not have a pull secret for your cluster, you can follow the documentation to Add or update your Red Hat pull secret.

Create a cluster

Follow the tutorial to create an Azure Red Hat OpenShift cluster. If you choose to install and use the *command-line interface* (CLI) locally, this tutorial requires you to use Azure CLI version 2.6.0 or later. Run `az --version` to find your current version. If you need to install or upgrade, see Install Azure CLI.

Connect to the cluster

To manage an Azure Red Hat OpenShift cluster, you need to use oc, the OpenShift command-line client.

> **Note**
>
> We recommend that you install OpenShift command line on Azure Cloud Shell and that you use it for all of the command-line operations in this article. Open your

shell from shell.azure.com or select the link:

Follow the tutorial to install your CLI, to retrieve your cluster credentials and to connect to the cluster with the web console and the OpenShift CLI.

Once you're logged in, you should see a message saying you're using the `default` project.

Output

```
Login successful.

You have access to 61 projects, the list has been suppressed. You can list
all projects with 'oc projects'

Using project "default".
```

Create a project

To create a new project called `demoproject`, run the command:

Azure CLI

```
oc new-project demoproject
```

You should see an output similar to:

Output

```
Now using project "demoproject" on server
"https://api.wzy5hg7x.eastus.aroapp.io:6443".

You can add applications to this project with the 'new-app' command. For
example, try:

    oc new-app django-psql-example

to build a new example application in Python. Or use kubectl to deploy a
simple Kubernetes application:

    kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-
install/hello-node
```

Launch the web console

Find out the cluster web console URL by running:

Azure CLI

```
az aro show \
   --name <cluster name> \
   --resource-group <resource group> \
   --query "consoleProfile.url" -o tsv
```

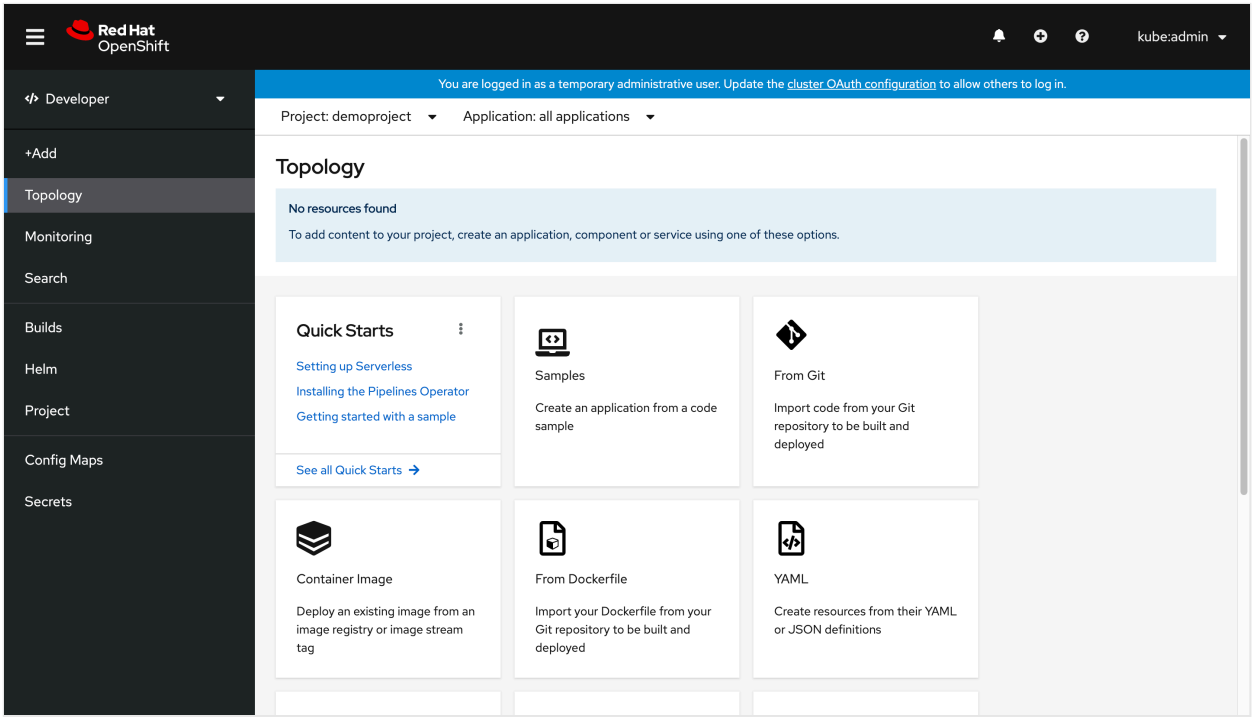
You should get a URL similar to this one.

Output

```
https://console-openshift-console.apps.wzy5hg7x.eastus.aroapp.io/
```

Launch the console URL in a browser and login using the `kubeadmin` credentials.

Switch to the *Developer* perspective instead of the *Administrator* perspective in the left-hand side menu and select `demoproject` in the list of projects. You should then be at the *Topology* page for the project.



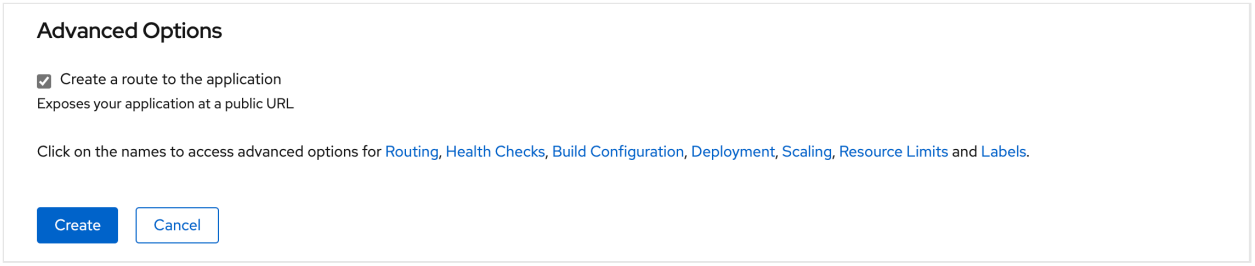
As the project is empty, no workloads should be found and you'll be presented with various options for how you can deploy an application.

Deploying using the web console

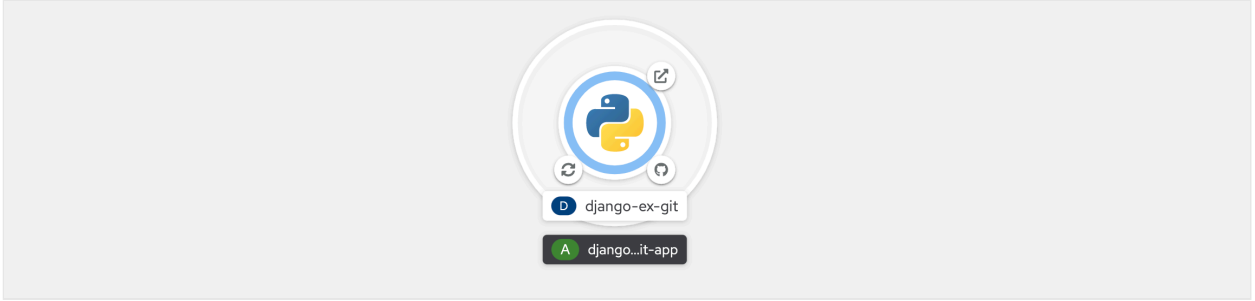
From the options presented for deploying an application, select *From Git*. This will land you on the *Import from Git* page. Use `https://github.com/sclorg/django-ex.git` as the **Git Repo URL**. The sample web application is implemented using the Python programming language.

> **Note**
>
> OpenShift detects that this is a Python project and selects the appropriate builder image.

Scroll down to *Advanced Options* and make sure that **Create a route to the application** is checked. This action will create an OpenShift route, a way to expose a service by giving it an externally reachable hostname.



When you're ready, at the bottom of the page click on **Create**. This will create resources to manage the build and deployment of the application. You'll then be redirected to the topology overview for the project.

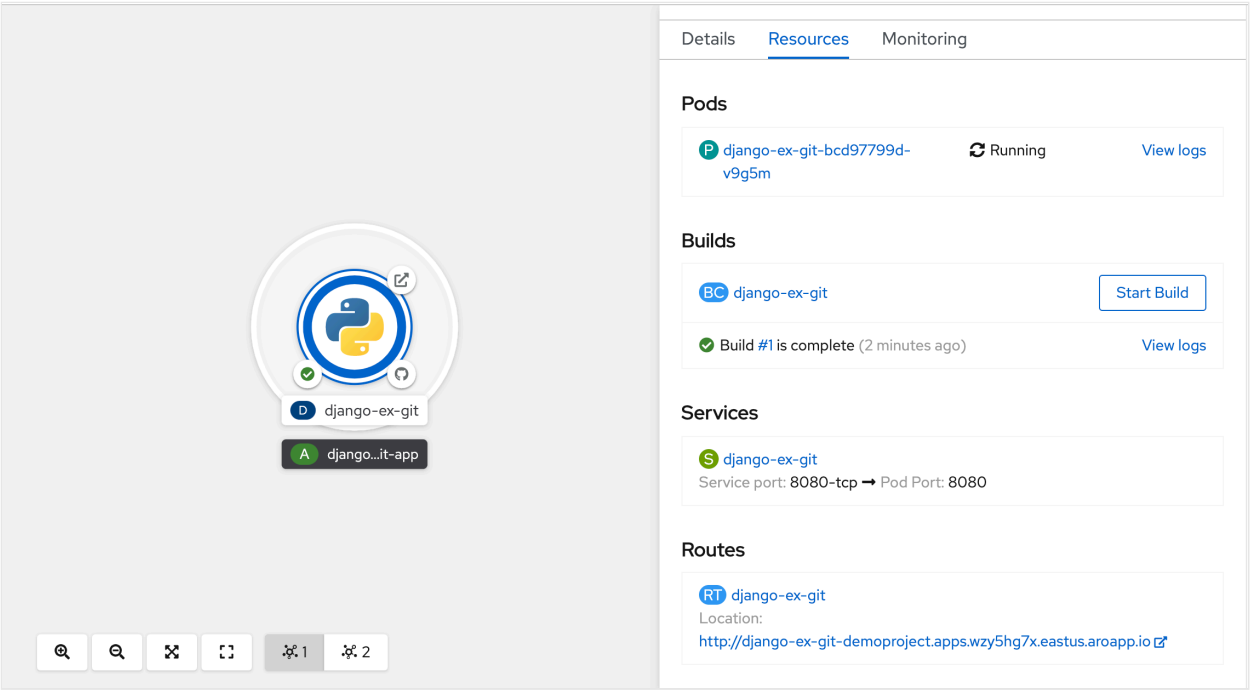


The topology overview provides a visual representation of the application you've deployed. With this view, you can see the overall application structure.

The Git icon can be clicked to take you to the Git repository from which the source code for the application was built. The icon shown on the bottom left shows the build status of the application. Clicking this icon will take you to the build details section. If the application has exposed routes, the icon on the top right can be clicked to open the URL for the application route that was created.

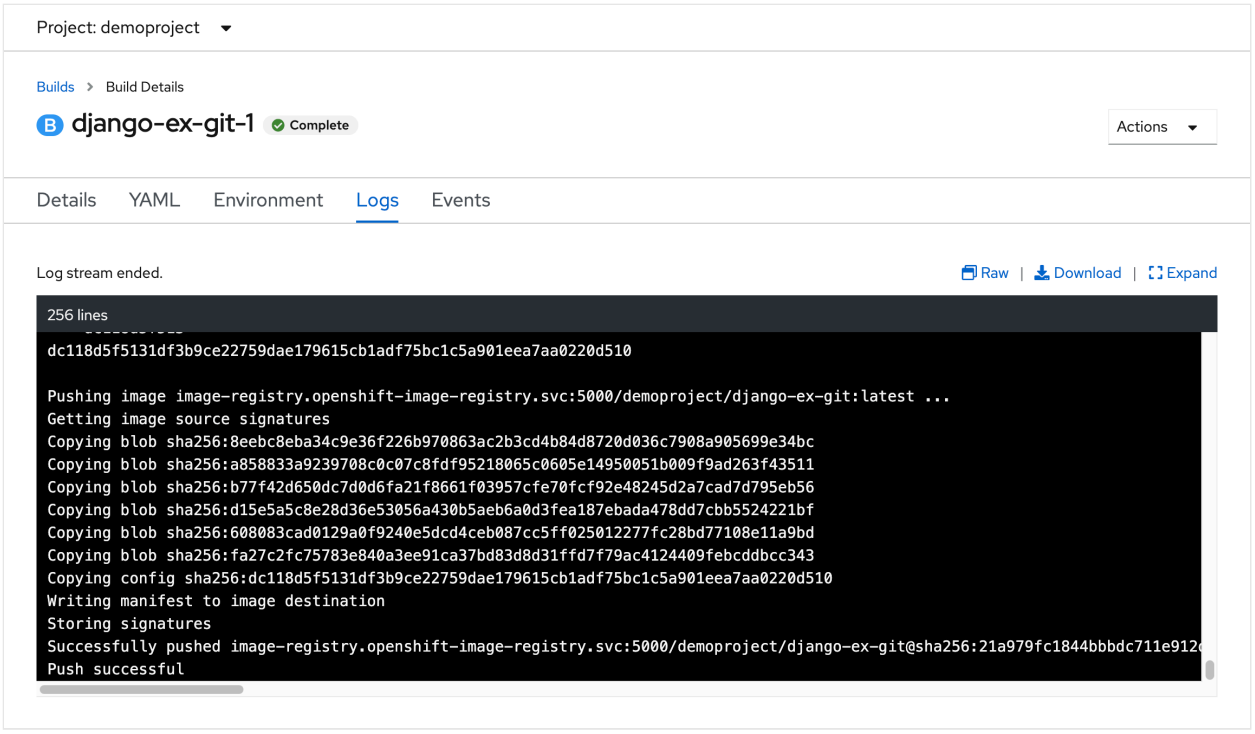
While the application is scaling up or down, starting rollouts and recreating pods, the application representation in the topology view will be animated to give you a real-time view of what's going on.

Clicking on the application icon will bring up more details as shown below.



Viewing the builder logs

Once the build has started, click on the *View Logs* link shown on the *Resources* panel.



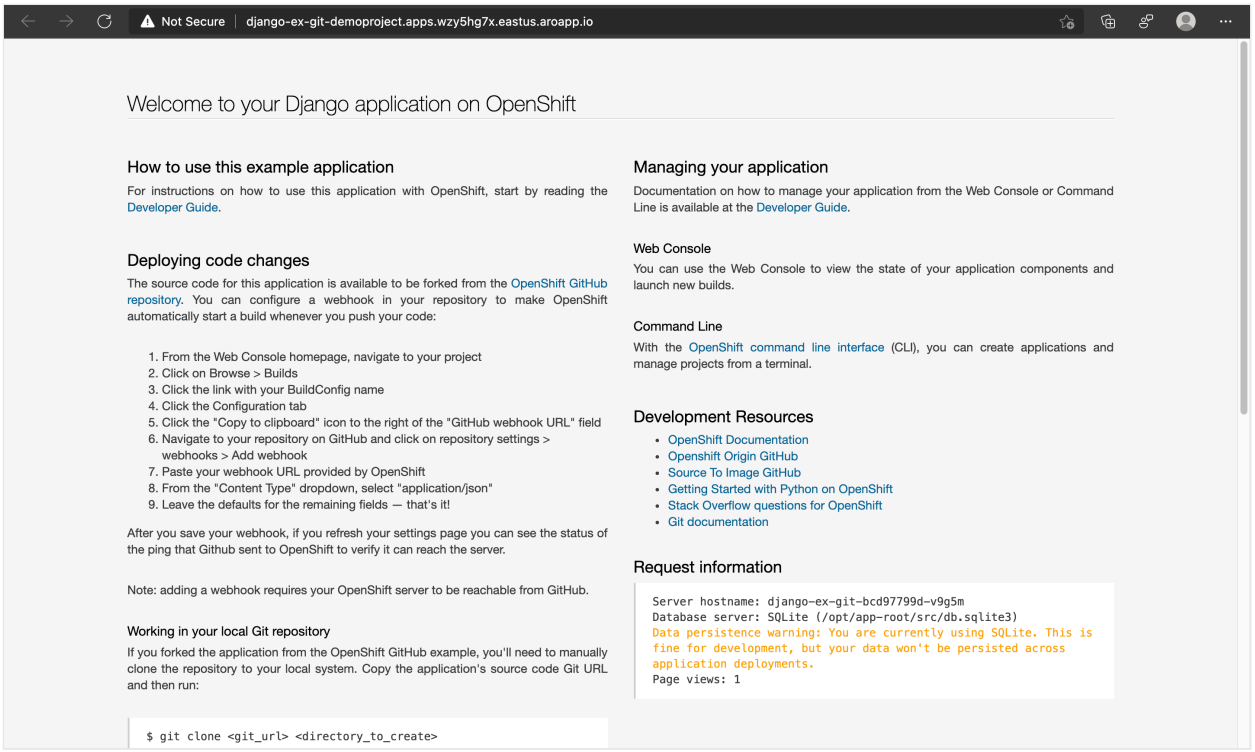
This will allow you to monitor the progress of the build as it runs. The builder image, Python in this case, will inject the application source code into the final image before it pushes it to the OpenShift internal image registry. The build will have completed successfully when you see a final message of "Push successful".

Accessing the application

Once the build of the application image has completed, it will be deployed.

Click on *Topology* in the left-hand menu bar to return to the topology view for the project. When you created the application using the web console, a *Route* was automatically created for the application and it will be exposed outside of the cluster. The URL that can be used to access the application from a web browser was visible on the *Resources* tab for the application you viewed previously.

From the topology view, you can get to the URL for the deployed application by clicking on the icon top right of the ring. When the deployment is complete, click on the icon and you should see the application you deployed.



Deploying using the command-line

You've learnt how to deploy an application using the web console, now lets deploy the same web application, but this time using the `oc` command-line tool.

Run the following command to delete the project and start over:

Azure CLI

```
oc delete project demoproject
```

You should get a confirmation message that the `demoproject` was deleted.

Output

```
project.project.openshift.io "demoproject" deleted
```

Create the `demoproject` again by running:

Azure CLI

```
oc new-project demoproject
```

Within the project, create a new application from the source on GitHub, specifying the S2I builder for the latest version of Python provided.

Azure CLI

```
oc new-app python:latest~https://github.com/sclorg/django-ex.git
```

This should display output similar to:

Output

```
--> Found image 8ec6f0d (4 weeks old) in image stream "openshift/python"
under tag "latest" for "python:latest"

    Python 3.8
    ----------
   [...]

    Tags: builder, python, python38, python-38, rh-python38

    * A source build using source code from
https://github.com/sclorg/django-ex.git will be created
      * The resulting image will be pushed to image stream tag "django-
ex:latest"
      * Use 'oc start-build' to trigger a new build
    * This image will be deployed in deployment config "django-ex"
    * Port 8080/tcp will be load balanced by service "django-ex"
      * Other containers can access this service through the hostname
"django-ex"

--> Creating resources ...
    imagestream.image.openshift.io "django-ex" created
    buildconfig.build.openshift.io "django-ex" created
    deploymentconfig.apps.openshift.io "django-ex" created
    service "django-ex" created
--> Success
    Build scheduled, use 'oc logs -f bc/django-ex' to track its progress.
    Application is not exposed. You can expose services to the outside world
by executing one or more of the commands below:
     'oc expose svc/django-ex'
    Run 'oc status' to view your app.
```

OpenShift will use the name of the Git repository as the name for the application. Review the status of the build and deployment by running:

Azure CLI

```
oc status
```

When the build and deployment is completed, you should see a similar output to.

Output

```
In project demoproject on server https://api.wzy5hg7x.eastus.aroapp.io:6443

svc/django-ex - 172.30.200.50:8080
  dc/django-ex deploys istag/django-ex:latest <-
    bc/django-ex source builds https://github.com/sclorg/django-ex.git on
openshift/python:latest
    deployment #1 deployed about a minute ago - 1 pod


2 infos identified, use 'oc status --suggest' to see details.
```

To expose the application outside the OpenShift cluster, you'll need to create a route by running:

Azure CLI

```
oc expose service/django-ex
```

You should get a confirmation.

Output

```
route.route.openshift.io/django-ex exposed
```

Retrieve the URL by running:

Azure CLI

```
oc get route django-ex
```

You should get back the hostname assigned to the route that you can use to browse to the deployed application.

Output

```
NAME        HOST/PORT                                                PATH
SERVICES    PORT       TERMINATION   WILDCARD
django-ex   django-ex-demoproject.apps.wzy5hg7x.eastus.aroapp.io
django-ex   8080-tcp                 None
```

Triggering a new binary build

As you work on the application, you'll likely want to make changes and see them deployed. You can easily setup a webhook that will trigger a new build and deployment

with every code commit. However, this may not be desirable as sometimes you'd like to see the changes without having to push every code change to the repository.

In cases where you're rapidly iterating on changes, you can use what is called a binary build. To demonstrate this scenario, clone the Git repository for the application locally by running:

Azure CLI

```
git clone https://github.com/sclorg/django-ex.git
```

This will create a sub directory `django-ex` containing the source code for the application:

Output

```
Cloning into 'django-ex'...
remote: Enumerating objects: 980, done.
remote: Total 980 (delta 0), reused 0 (delta 0), pack-reused 980
Receiving objects: 100% (980/980), 276.23 KiB | 4.85 MiB/s, done.
Resolving deltas: 100% (434/434), done.
```

Change into the sub directory:

Azure CLI

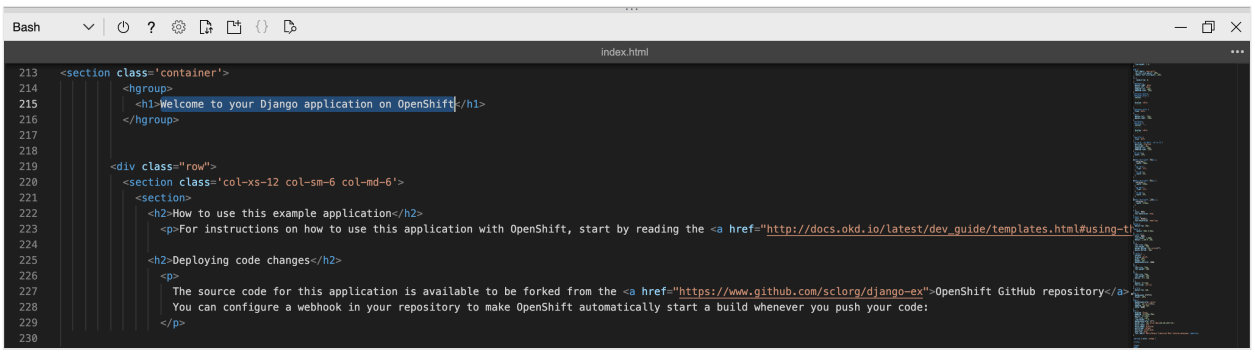
```
cd django-ex
```

Open the integrated Azure Cloud Shell editor:

Azure CLI

```
code welcome/templates/welcome/index.html
```

Scroll down and change the line that says `Welcome to your Django application on OpenShift` to say `Welcome to Azure Red Hat OpenShift`. Save the file and close the editor through the `...` menu on the top right.



Start a new build by running the command:

Azure CLI

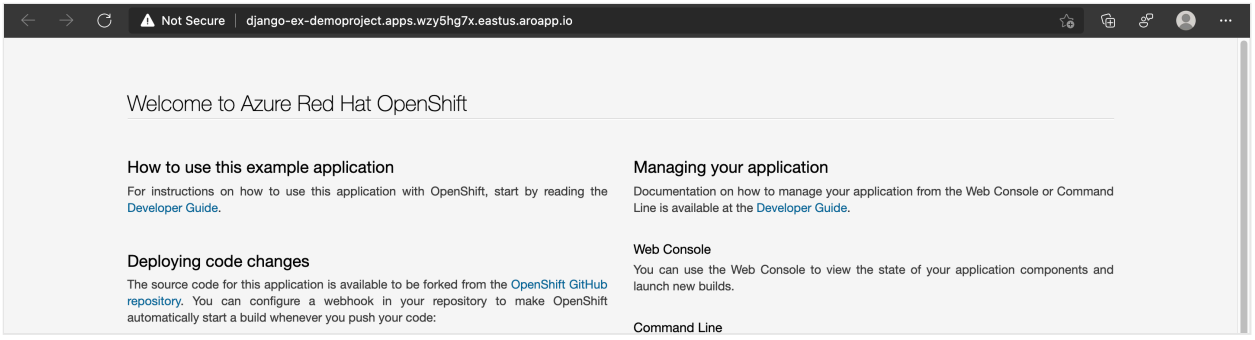
```
oc start-build django-ex --from-dir=. --wait
```

By passing the `--from-dir=.` flag, the OpenShift command-line will upload the source code from the specified directory then initiate the build and deployment process. You should get an output similar to the below, and after a few minutes, the build should be completed.

Output

```
Uploading directory "." as binary input for the build ...
.
Uploading finished
build.build.openshift.io/django-ex-2 started
```

If you refresh the browser with the application, you should see the updated title.



Clean up resources

When you're done with the application, you can run the following command to delete the project:

Azure CLI

```
oc delete project demoproject
```

You can also delete the cluster by following the instructions in Tutorial: Delete an Azure Red Hat OpenShift 4 cluster.

Next steps

In this guide, you learned how to:

- ✓ Create a project
- ✓ Deploy an application from source code using the web console
- ✓ Deploy an application from source code using the OpenShift command-line
- ✓ Trigger a binary build using the OpenShift command-line

Learn more about how to build and deploy applications using source-to-image and other build strategies.

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

Deploy applications to Azure Red Hat OpenShift using OpenShift Serverless

08/07/2025

In this article, you deploy an application to an Azure Red Hat OpenShift cluster with OpenShift Serverless. OpenShift Serverless helps developers to deploy and run applications that scale up or scale to zero on demand. This scalability eliminates consumption of resources when they're not in use.

Application code can be packaged in a container along with the appropriate runtimes. Serverless functionality starts the application containers when they're triggered by an event. You can trigger applications through various events: from your own applications, from multiple cloud service providers, software as a service (SaaS) systems and other services.

You can use built-in OpenShift interface features to manage all aspects of serverless container deployment. Developers can visually identify which events are driving the launch of containerized applications. There are also multiple ways to modify event parameters. OpenShift Serverless applications can be integrated with other OpenShift services, such as OpenShift Pipelines, Service Mesh, and Monitoring. This integration provides a complete serverless application development and deployment experience.

Before you start

Create a cluster

Follow the tutorial to create an Azure Red Hat OpenShift cluster. If you choose to install and use the *command-line interface* (CLI) locally, this tutorial requires you to use Azure CLI version 2.6.0 or later. Run `az --version` to find your current version. If you need to install or upgrade, see Install Azure CLI.

Connect to the cluster

To manage an Azure Red Hat OpenShift cluster, you need to use oc, the OpenShift command-line client.

> **Note**
>
> We recommend that you install OpenShift command line on Azure Cloud Shell and that you use it for all of the command-line operations in this article. Open your shell from

shell.azure.com or select the link:

Follow the tutorial to install your CLI, to retrieve your cluster credentials and to connect to the cluster with the web console and the OpenShift CLI.

Once you're logged in, you should see a message saying you're using the `default` project.

Output

```
Login successful.

You have access to 61 projects, the list has been suppressed. You can list all
projects with 'oc projects'

Using project "default".
```

Install the Knative command-line interface (kn)

Download the latest release of the *command-line interface* (CLI) that's appropriate for your machine, from https://github.com/knative/client/releases/

If you run commands on Azure Cloud Shell, download the latest Knative CLI for Linux.

Azure CLI

```
cd ~
wget https://github.com/knative/client/releases/download/v0.22.0/kn-linux-amd64

mkdir knative
chmod +x kn-linux-amd64
mv kn-linux-amd64 knative/kn
echo 'export PATH=$PATH:~/knative' >> ~/.bashrc && source ~/.bashrc
```

Open the OpenShift web console

Find your cluster web console URL by running the following script:

Azure CLI

```
az aro show \
   --name <cluster name> \
   --resource-group <resource group> \
   --query "consoleProfile.url" -o tsv
```

You should get a URL similar to the following.

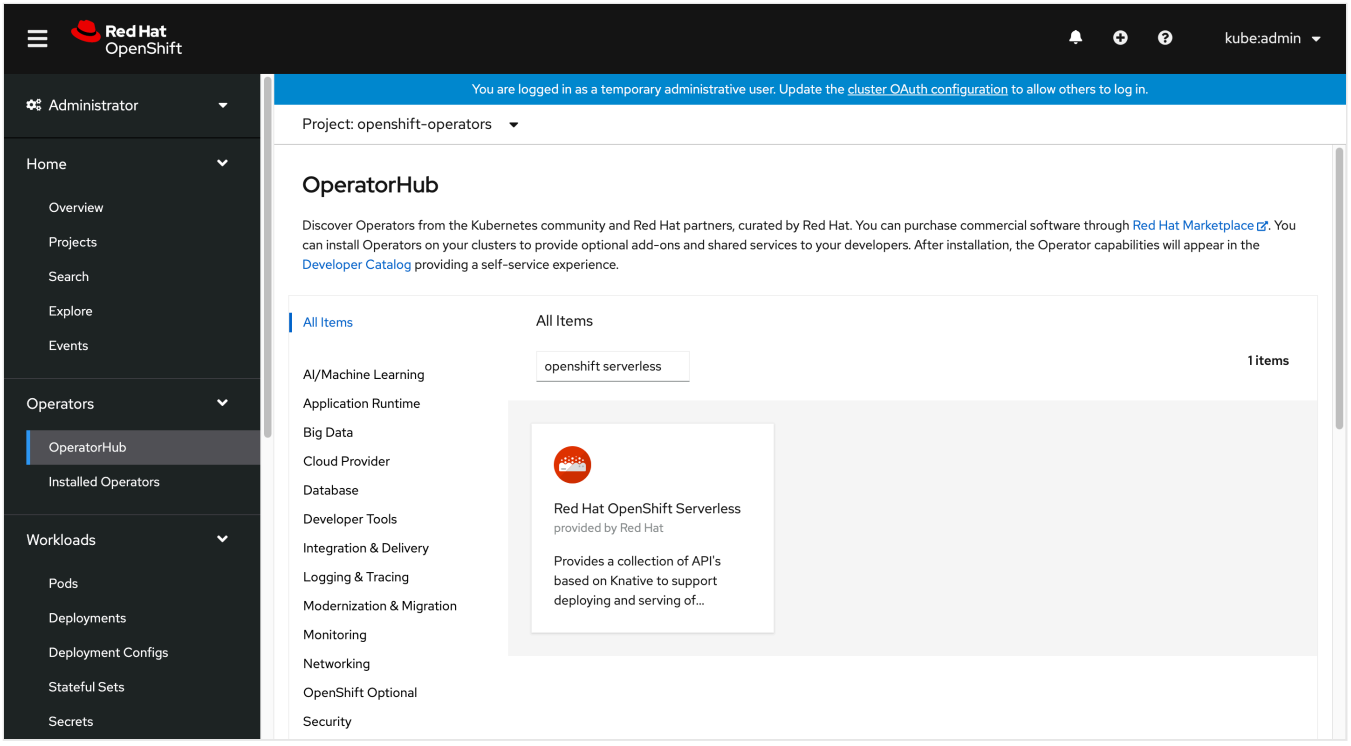
Output

```
https://console-openshift-console.apps.wzy5hg7x.eastus.aroapp.io/
```

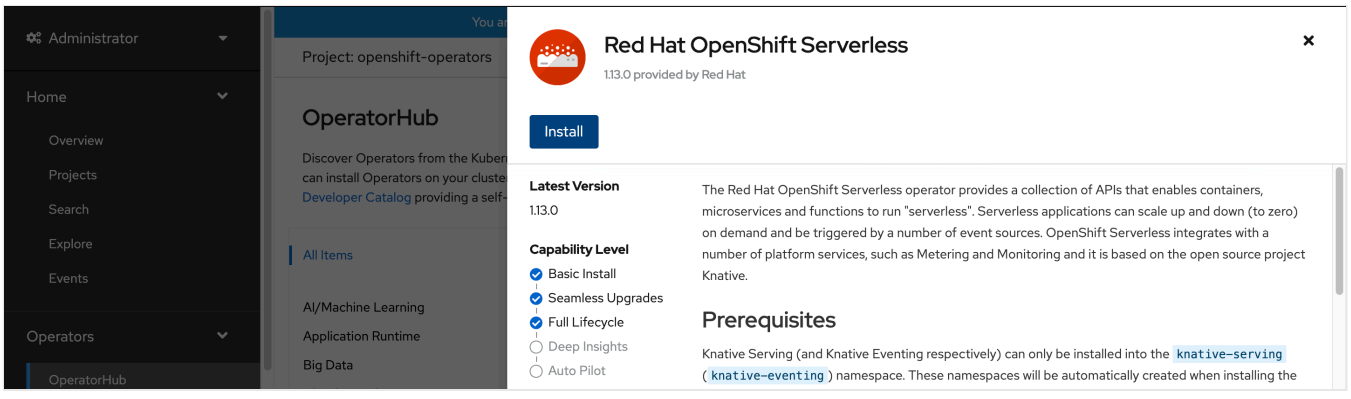
Open a web browser and open the console URL. Sign in using `kubeadmin` credentials.

Install the OpenShift Serverless operator

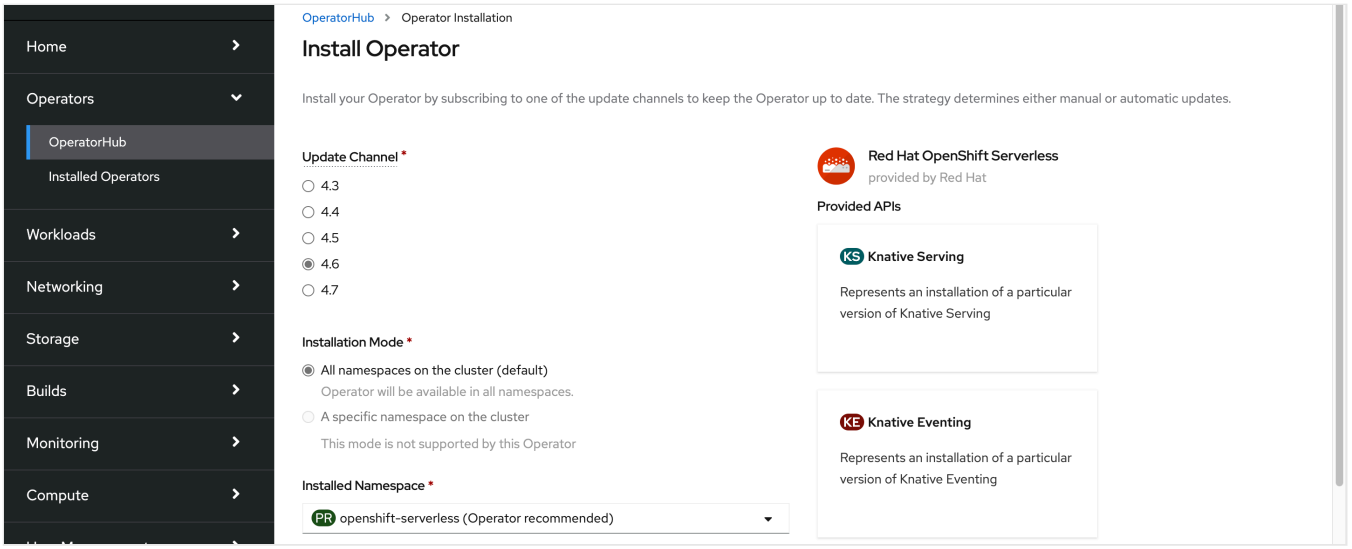
When you're logged into the OpenShift web console, confirm that you're in *Administrator* view. Open the *Operator Hub* and select the **OpenShift Serverless** operator.



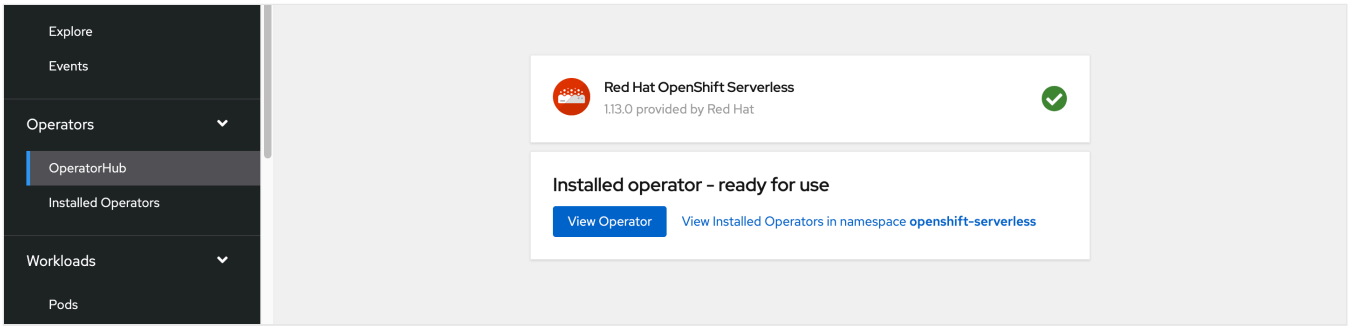
Next, open the operator installation page by selecting **Install**.



Choose the appropriate *Update Channel* for your Azure Red Hat OpenShift's cluster version and install the operator in the `openshift-serverless` namespace. Scroll down and select **Install**.



Within a few minutes, the status page reflects that the operator is installed and is ready for use. Select the **View Operator** button to proceed.

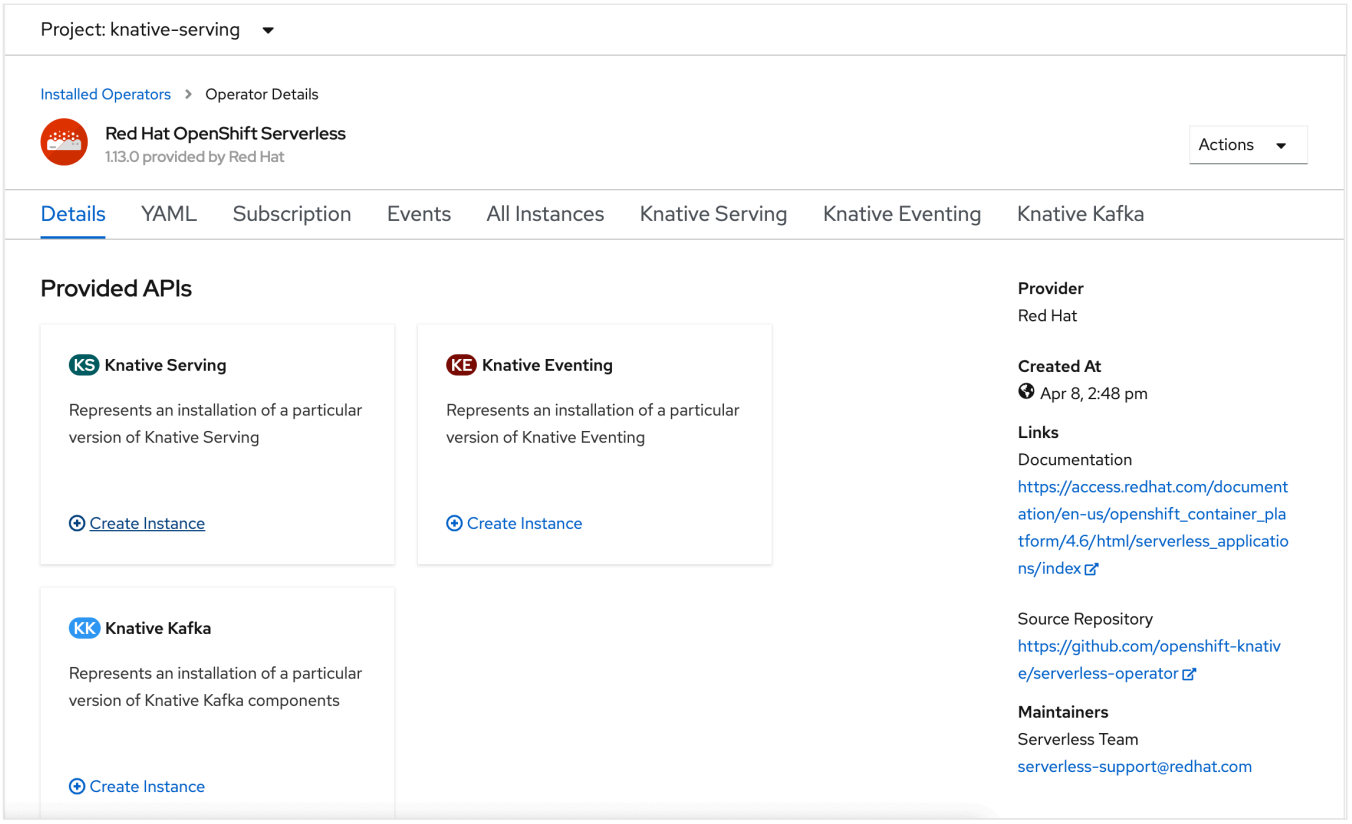


Install Knative Serving

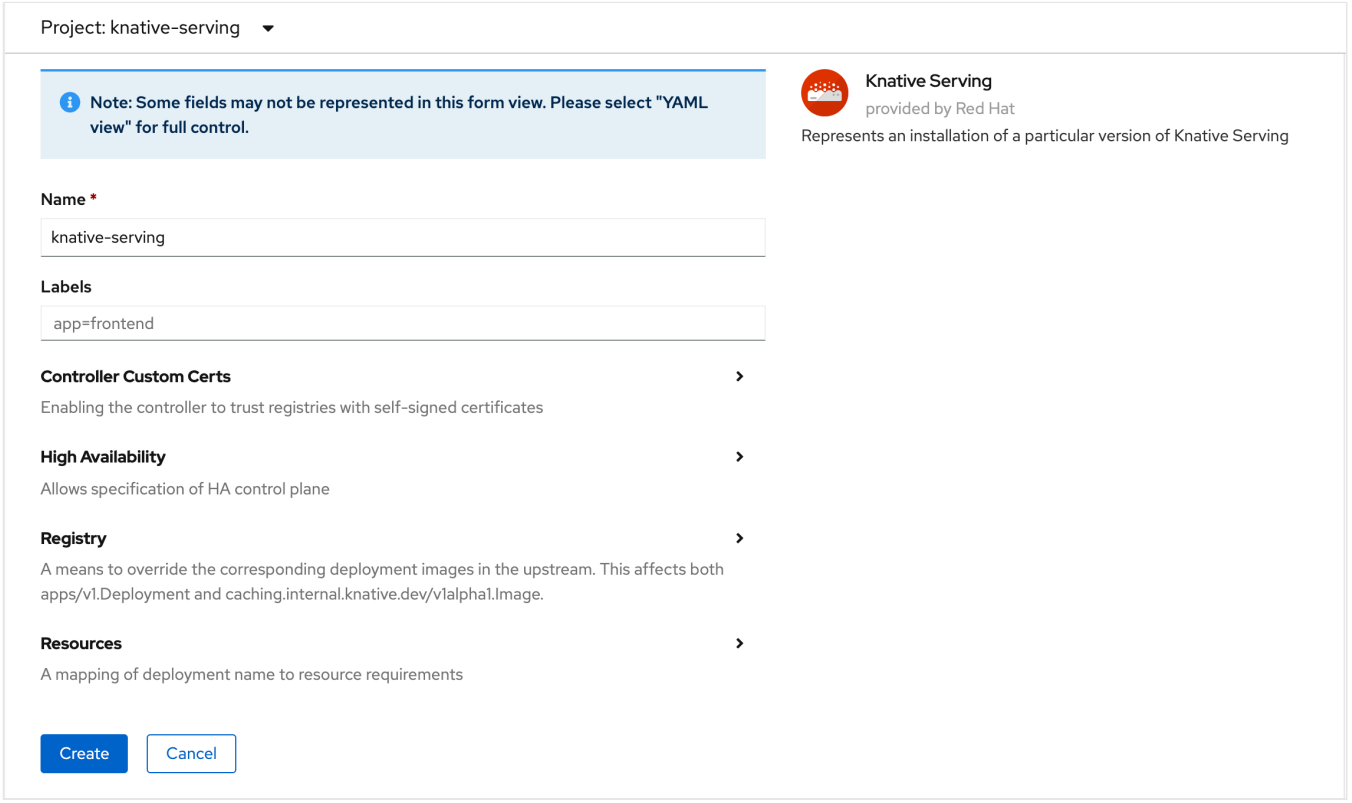
The option to run a container in a serverless fashion on OpenShift Serverless is possible by using upstream Knative. Knative extends Kubernetes to provide a set of components that deploy, run, and manage modern applications through its serverless methodology.

Create an instance of the Knative Serving

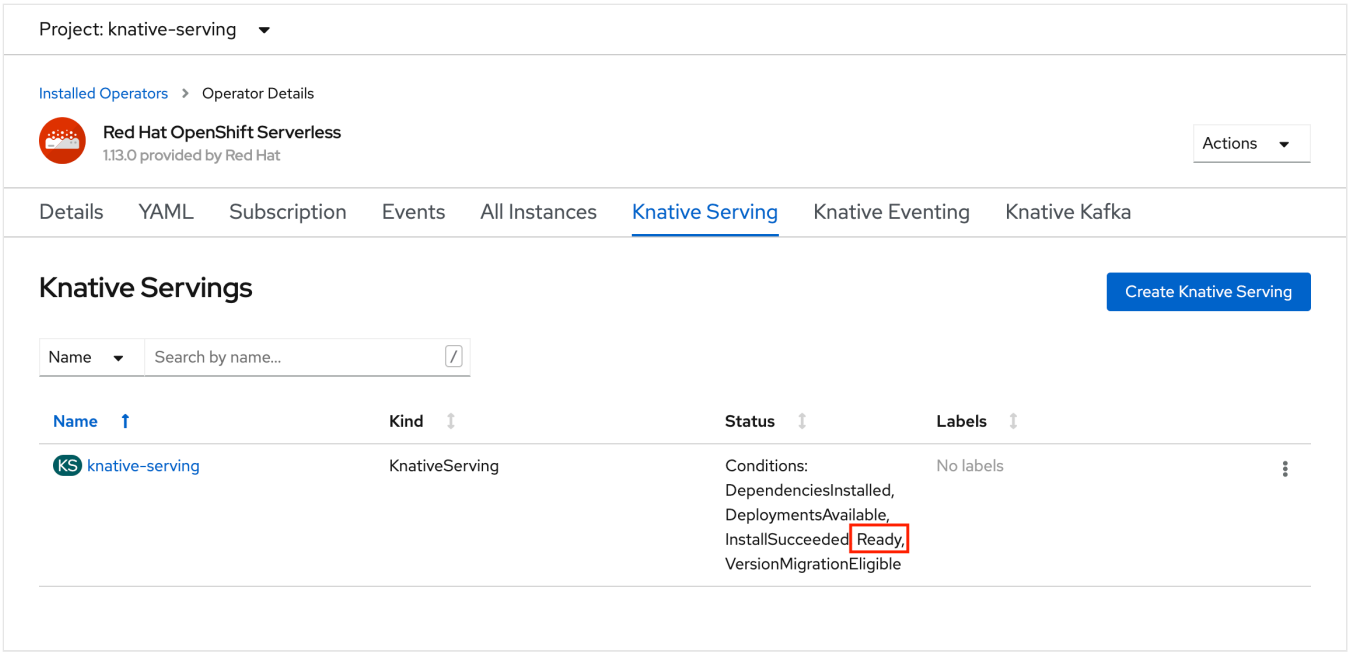
In the upper-left corner of the window, in the **Project** list, select `knative-server`. Then in the **Provided APIs** pane, select **Create Instance** within the *Knative Serving* card.



On the *Create Knative Serving* page, keep all of the default values. Scroll down and select the **Create** button.



OpenShift Serverless is installed when the *Status* column shows **Ready**. Now you're ready to create an OpenShift Serverless project.



Create a serverless project

To create a new project called `demoserverless`, run the following command:

Azure CLI

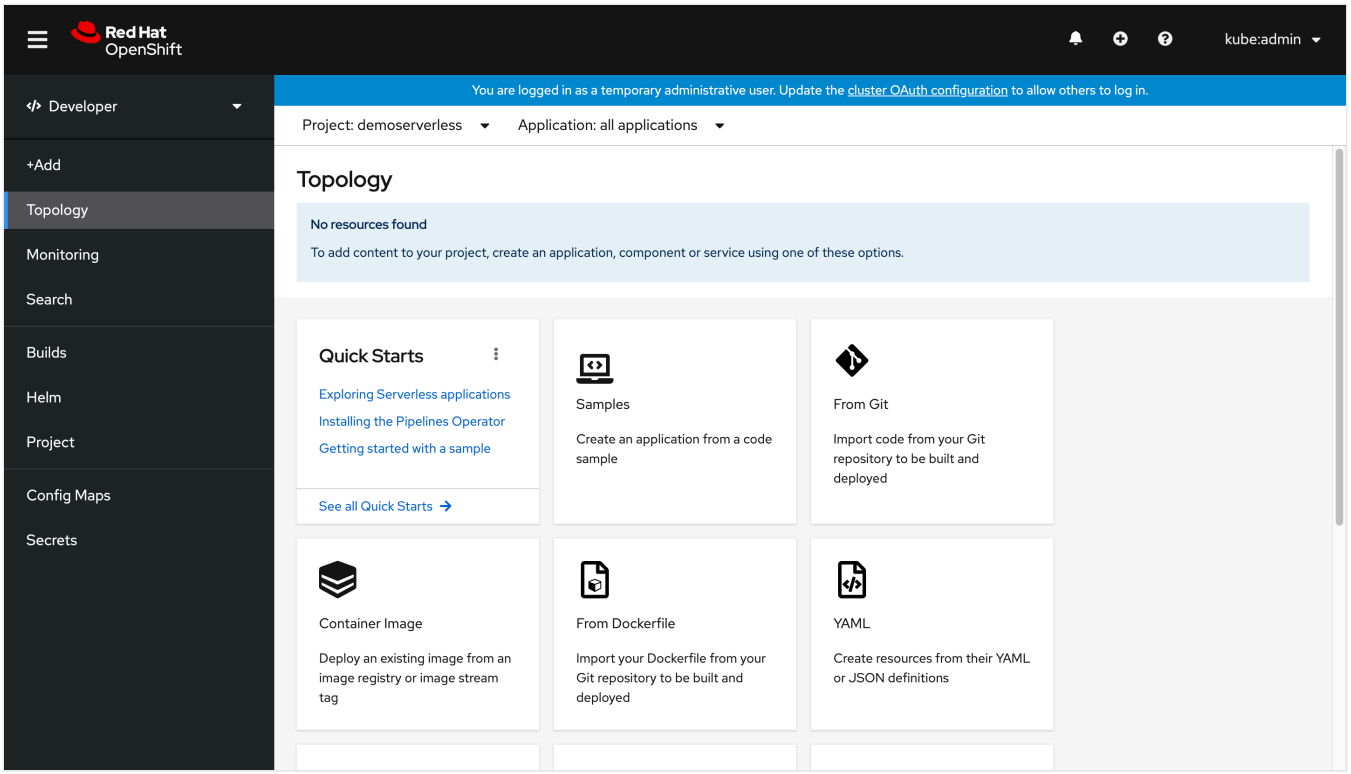
```
oc new-project demoserverless
```

The output should be similar to the following example:

Output

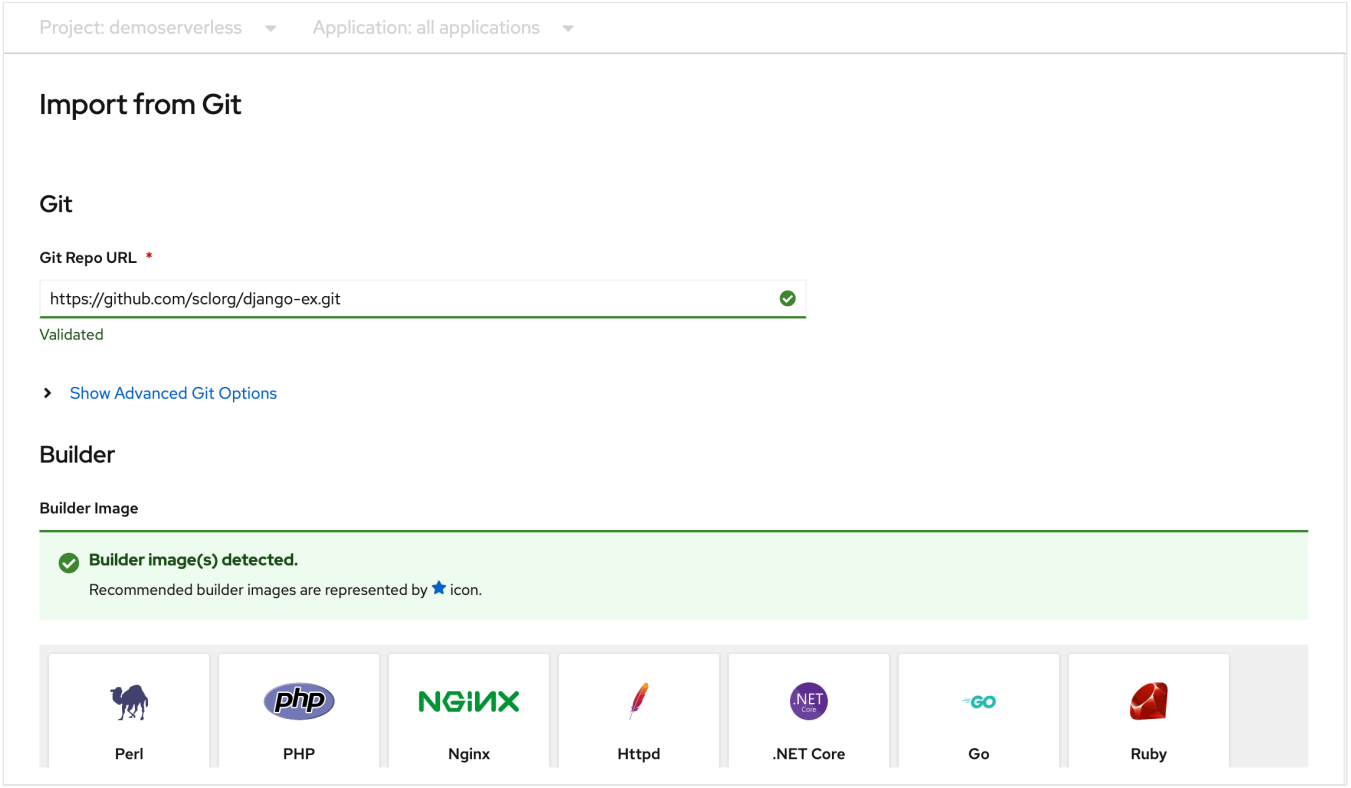
```
Now using project "demoserverless" on server
"https://api.wzy5hg7x.eastus.aroapp.io:6443".

You can add applications to this project with the 'new-app' command. For example,
build a new example application in Python with the following:

    oc new-app django-psql-example

Or use kubectl to deploy a simple Kubernetes application:

    kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-
install/hello-node
```

Let's switch from the Administrator view to the Developer view. Go to your list of projects in the left menu and select `demoserverless`. You're now at the **Topology** page for the project.



Deploy using the web console

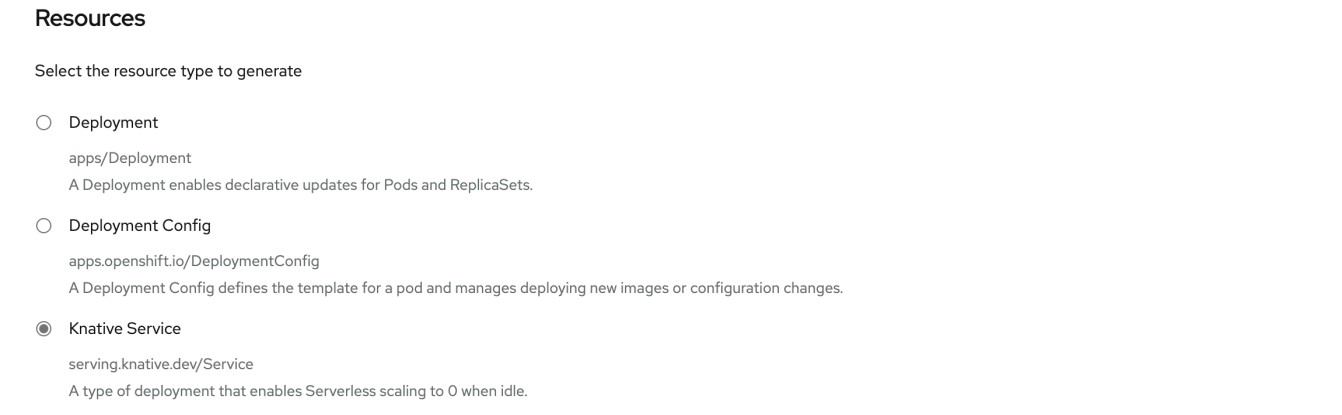
On the **Topology** page, select **From Git**. On the *Import from Git* page, use `https://github.com/sclorg/django-ex.git` as the **Git Repo URL**. A sample web application is implemented with Python programming language.



> ⓘ Note

OpenShift detects that this is a Python project and selects the appropriate builder image.

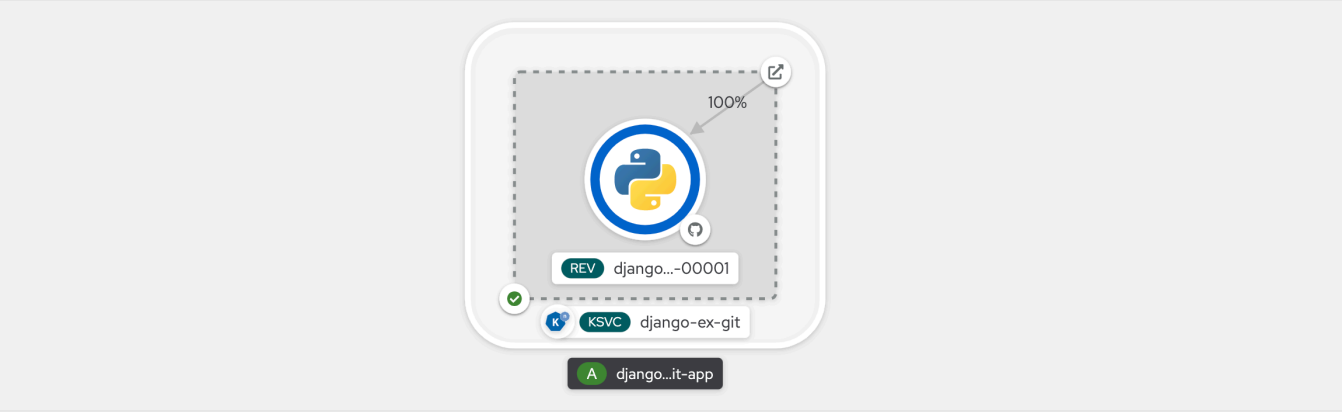
Scroll to **Resources** and confirm that **Knative Service** is selected as the resource type to generate. This creates a Knative Service, a type of deployment that enables OpenShift Serverless scaling to zero when idle.



At the bottom of the page, select **Create**. This creates resources to manage the build and deployment of the application. You're then redirected to the topology overview for the project.

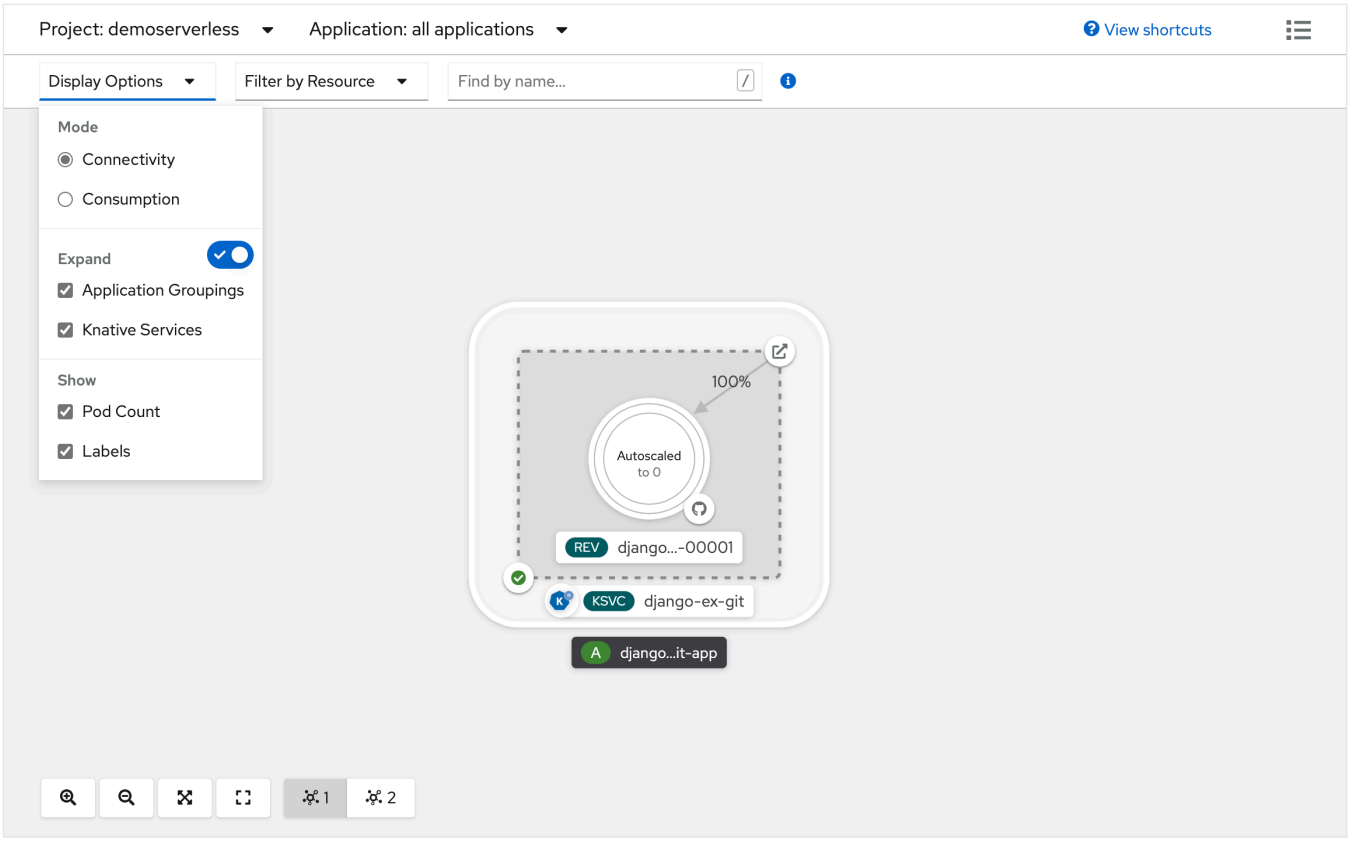
The topology overview provides a visual representation of the application you've deployed. You can see the overall application structure.

Wait for the build to complete. It might take a few minutes. When the build completes, a green checkmark appears in the lower-left corner of the service.

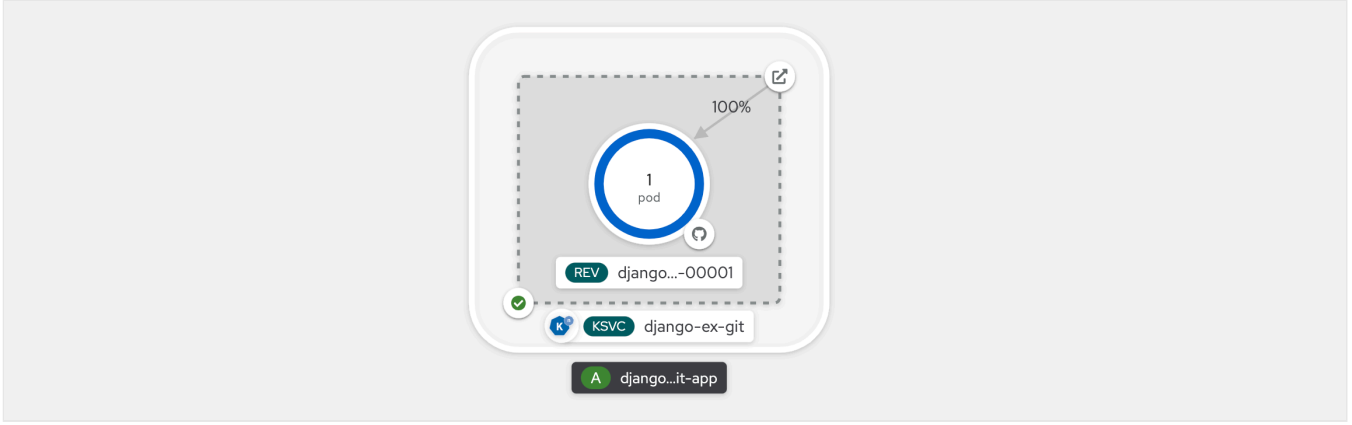


See your application scale

At the top of the Topology view, in the **Display Options** list, select *Pod Count*. Wait for the Pod count to scale down to zero Pods. Scaling down might take a few minutes.



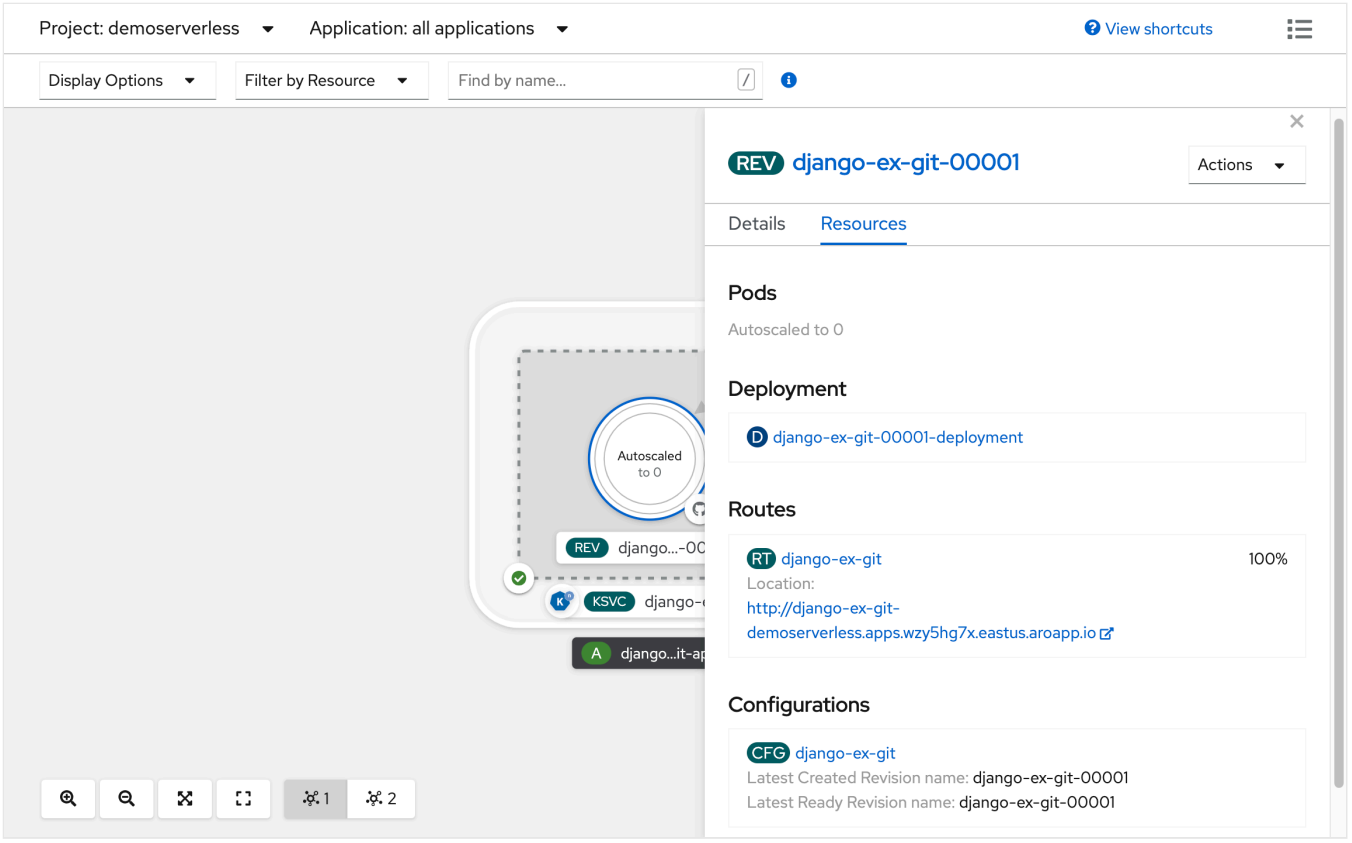
In the upper-right corner of the Knative Service panel, select the **Open URL** icon. The application opens in a new browser tab. Close the tab and return to the Topology view. There you can see that your application scaled up to one Pod, to accommodate your request. After a few minutes, your application scales back down to zero Pods.



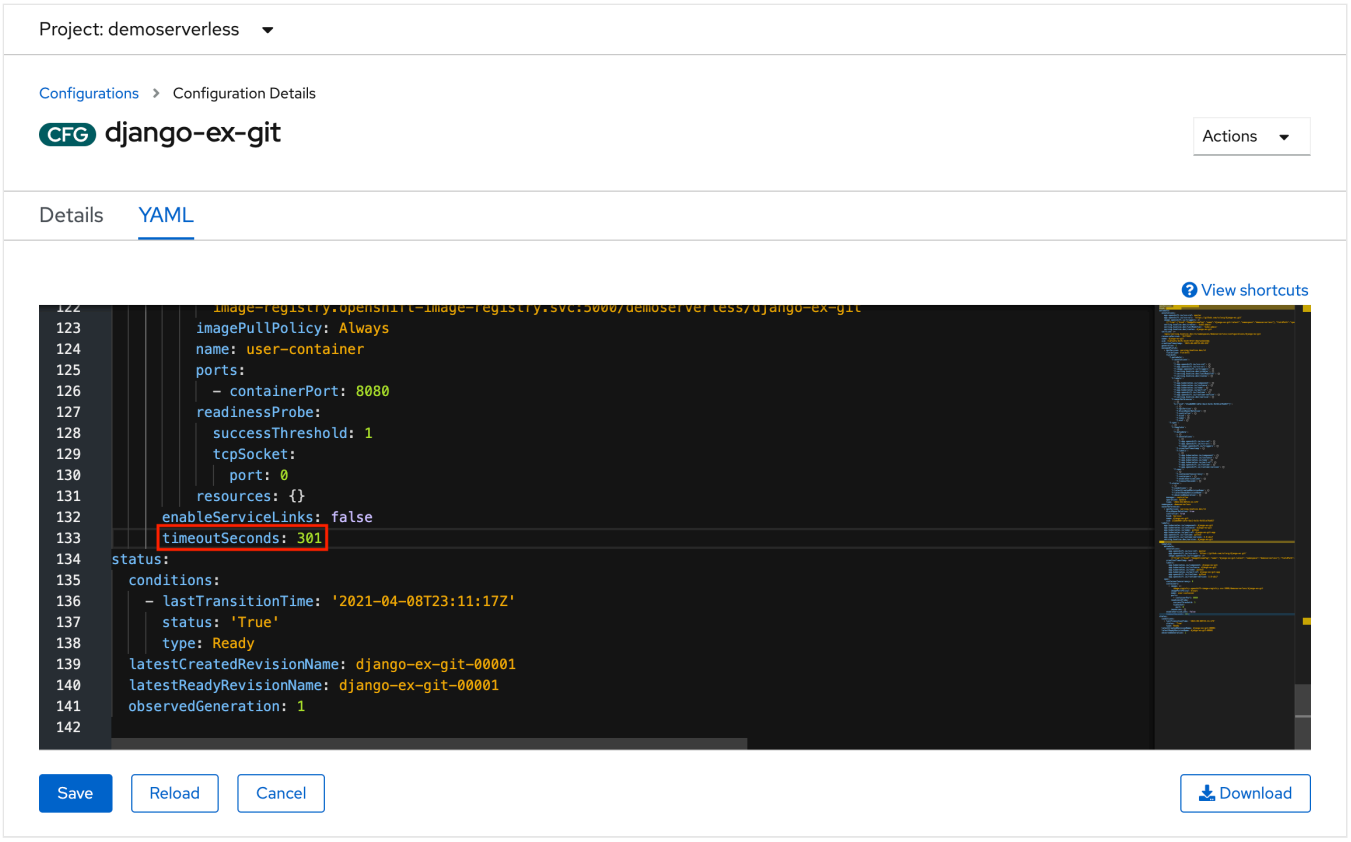
Forcing a new revision and setting traffic distribution

Knative services allow traffic mapping, which means that revisions of a service can be mapped to an allocated portion of traffic. With each service configuration update, a new revision is created. The service route then points all traffic to the latest ready revision by default. You can change this behavior by defining which revision gets portions of the traffic. Traffic mapping also provides the option to create unique URLs for individual revisions.

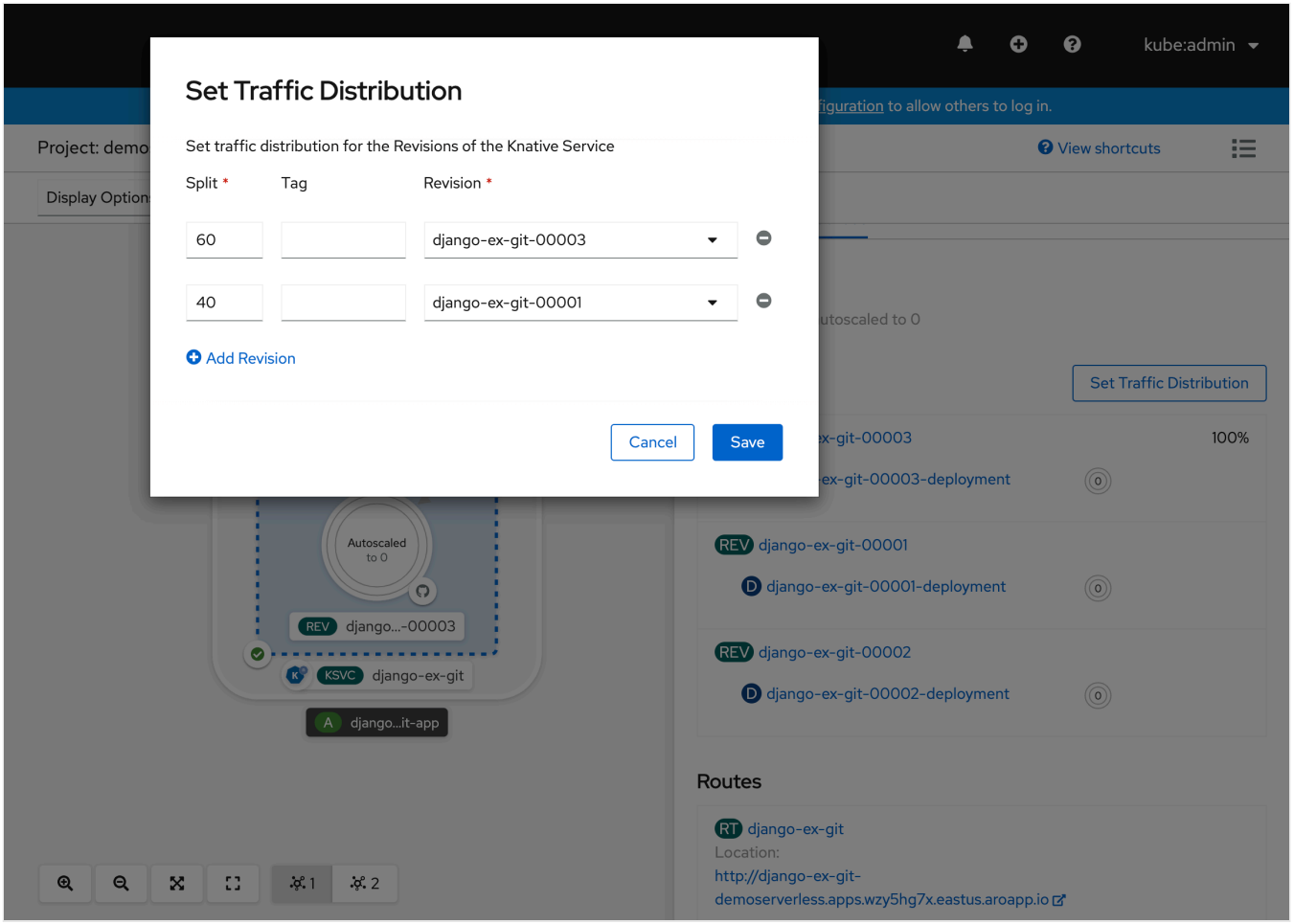
In the topology created, select the revision displayed inside your service to view its details. The badges under the Pod ring and at the top of the detail panel should be `(REV)`. In the side panel, within the **Resources** tab, scroll down and select the configuration associated with your service.



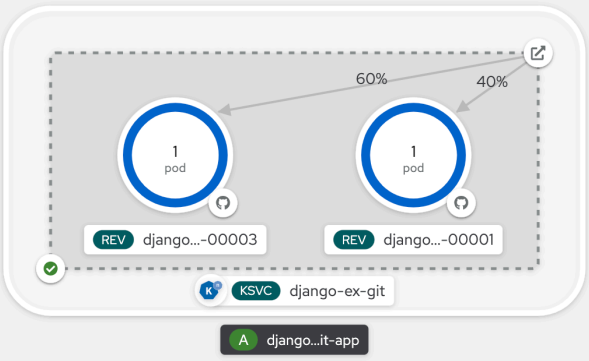
Force a configuration update by switching to the *YAML* tab and scrolling down to edit the value of `timeoutSeconds`. Change the value to `301`. Select **Save**. In a real world scenario, configuration updates can also be triggered by updating the container image tag.



Return to the *Topology* view, you see that a new revision was deployed. Select the service ending with the badge `(KSVC)` and select the **Set Traffic Distribution** button. You should now be able to divide traffic between the revisions in the service.



The **Topology** view shows you how traffic is distributed between the revisions.



Using the Knative command-line interface (kn)

In prior steps, you used the OpenShift web console to create and deploy an application to OpenShift Serverless. Since OpenShift Serverless is running Knative underneath, you can also use the Knative command-line interface (kn) to create Knative services.

> **Note**
>
> If you haven't installed the `kn` CLI already, make sure to follow the steps in the prerequisites section of this article. Also make sure that you signed in using the OpenShift command-line interface `oc`.

We're going to use a container image that is already built at `quay.io/rhdevelopers/knative-tutorial-greeter`.

Deploy a service

To deploy the service, run the following command:

Azure CLI

```
kn service create greeter \
--image quay.io/rhdevelopers/knative-tutorial-greeter:quarkus \
--namespace demoserverless \
--revision-name greeter-v1
```

You see an output similar to the following.

Output

```
Creating service 'greeter' in namespace 'demoserverless':

  0.044s The Route is still working to reflect the latest desired specification.
  0.083s ...
  0.114s Configuration "greeter" is waiting for a Revision to become ready.
 10.420s ...
 10.489s Ingress has not yet been reconciled.
 10.582s Waiting for load balancer to be ready
 10.763s Ready to serve.

Service 'greeter' created to latest revision 'greeter-v1' is available at URL:
http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
```

You can retrieve a list of routes in the project by running:

Azure CLI

```
kn route list
```

You get a list of routes in the namespace. Open the URL in a web browser to view the deployed service.

Output

```
NAME      URL                                                    READY
greeter   http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io   True
```

Deploy a new version of the service

Deploy a new version of the application by running the following command, and by passing the `:latest` image tag and an environment variable `MESSAGE_PREFIX`:

Azure CLI

```
kn service update greeter \
 --image quay.io/rhdevelopers/knative-tutorial-greeter:latest \
 --namespace demoserverless \
 --env MESSAGE_PREFIX=GreeterV2 \
 --revision-name greeter-v2
```

You get a confirmation that a new revision, `greeter-v2` was deployed.

Output

```
Updating Service 'greeter' in namespace 'demoserverless':

  5.029s Traffic is not yet migrated to the latest revision.
  5.086s Ingress has not yet been reconciled.
  5.190s Waiting for load balancer to be ready
  5.332s Ready to serve.

Service 'greeter' updated to latest revision 'greeter-v2' is available at URL:
http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
```

To view a list of all revisions and their traffic distributions, run the following command:

Azure CLI

```
kn revision list
```

You get a list similar to the following output. In this instance, the new revision receives 100% of the traffic.

Output

```
NAME           SERVICE    TRAFFIC   TAGS   GENERATION   AGE     CONDITIONS   READY
REASON
greeter-v2     greeter    100%             2            90s     3 OK / 4     True
greeter-v1     greeter                     1            5m32s   3 OK / 4     True
```

Blue/green and canary deployments

When a new revision is deployed, by default it's assigned 100% of the traffic. Let's say you want to implement a blue/green deployment strategy where you can quickly roll back to the older version of the application. Knative makes this process easy.

You can update the service to create three traffic tags, while assigning 100% of traffic to them.

- **current**: points to the currently deployed version
- **prev**: points to the previous version
- **latest**: always points to the latest version

Azure CLI

```
kn service update greeter \
   --tag greeter-v2=current \
   --tag greeter-v1=prev \
   --tag @latest=latest
```

You get a confirmation similar to the following.

Output

```
Updating Service 'greeter' in namespace 'demoserverless':

  0.037s Ingress has not yet been reconciled.
  0.121s Waiting for load balancer to be ready
  0.287s Ready to serve.

Service 'greeter' with latest revision 'greeter-v2' (unchanged) is available at
URL:
http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
```

List routes using the following command:

Azure CLI

```
kn route describe greeter
```

You receive output showing the URLs for each of the tags, along with their traffic distribution.

Output

```
Name:       greeter
Namespace:  demoserverless
Age:        10m
URL:        http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
Service:    greeter

Traffic Targets:
  100%  @latest (greeter-v2) #latest
        URL:  http://latest-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
    0%  greeter-v1 #prev
        URL:  http://prev-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
    0%  greeter-v2 #current
        URL:  http://current-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io

[..]
```

Let's say you want to quickly roll back to the previous version, you can update traffic distribution to send 100% of the traffic to the previous tag:

Azure CLI

```
kn service update greeter --traffic current=0 --traffic prev=100
```

List the routes and check again, by using the following command:

Azure CLI

```
kn route describe greeter
```

You see output showing that 100% of traffic distribution is going to the previous version.

Output

```
Name:       greeter
Namespace:  demoserverless
Age:        19m
URL:        http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
Service:    greeter

Traffic Targets:
    0%  @latest (greeter-v2) #latest
        URL:  http://latest-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
  100%  greeter-v1 #prev
        URL:  http://prev-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io
    0%  greeter-v2 #current
        URL:  http://current-greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io

[..]
```

Play around with the traffic distribution while refreshing the main route in your browser (`http://greeter-demoserverless.apps.wzy5hg7x.eastus.aroapp.io` in this case).

Clean up resources

When you're finished with the application, you can run the following command to delete the project:

Azure CLI

```
oc delete project demoserverless
```

You can also delete the cluster by following the instructions in Tutorial: Delete an Azure Red Hat OpenShift 4 cluster.

Next steps

In this guide, you learned how to:

- ✓ Install the OpenShift Serverless operator and Knative Serving
- ✓ Deploy a serverless project using the web console

- ✓ Deploy a serverless project using the Knative CLI (kn)
- ✓ Configure blue/green deployments and canary deployments using the Knative CLI (kn)

Learn more about how to build and deploy serverless, event-driven applications on Azure Red Hat OpenShift using OpenShift Serverless. Follow the Getting started with OpenShift Serverless documentation and the Creating and managing serverless applications documentation.

Deploy WebSphere Liberty and Open Liberty on Azure Red Hat OpenShift

Article • 05/06/2025

This article shows you how to quickly stand up IBM WebSphere Liberty and Open Liberty on Azure Red Hat OpenShift using the Azure portal.

This article uses the Azure Marketplace offer for Open/WebSphere Liberty to accelerate your journey to Azure Red Hat OpenShift. The offer automatically provisions several resources including an Azure Red Hat OpenShift cluster, the Liberty Operators, and optionally a container image including Liberty and your application. To see the offer, visit the Azure portal. If you prefer manual step-by-step guidance for running Liberty on Azure Red Hat OpenShift that doesn't utilize the automation enabled by the offer, see Manually deploy a Java application with Open Liberty/WebSphere Liberty on an Azure Red Hat OpenShift cluster.

This article is intended to help you quickly get to deployment. Before going to production, you should explore Tuning Liberty.

If you're interested in providing feedback or working closely on your migration scenarios with the engineering team developing WebSphere on Azure solutions, fill out this short survey on WebSphere migration and include your contact information. The team of program managers, architects, and engineers will promptly get in touch with you to initiate close collaboration.

> **Important**
>
> While Azure Red Hat OpenShift is jointly engineered, operated, and supported by Red Hat and Microsoft to provide an integrated support experience, the software you run on top of Azure Red Hat OpenShift, including that described in this article, is subject to its own support and license terms. For details about support of Azure Red Hat OpenShift, see Support lifecycle for Azure Red Hat OpenShift 4. For details about support of the software described in this article, see the main pages for that software as listed in the article.

> **Note**
>
> Azure Red Hat OpenShift requires a minimum of 40 cores to create and run an OpenShift cluster. The default Azure resource quota for a new Azure subscription doesn't meet this requirement. To request an increase to your resource limit, see the section Request an increase for non-adjustable quotas in Increase VM-family vCPU quotas. Because the kind of quota for which you need to request an increase is "non-adjustable", you must file

a support ticket. The steps in **Request an increase for non-adjustable quotas** show you exactly how to file the ticket with the correct content.

The free trial subscription isn't eligible for a quota increase. Upgrade to a Pay-As-You-Go subscription before you request a quota increase. For more information, see **Upgrade your Azure free account or Azure for Students Starter account**.

Prerequisites

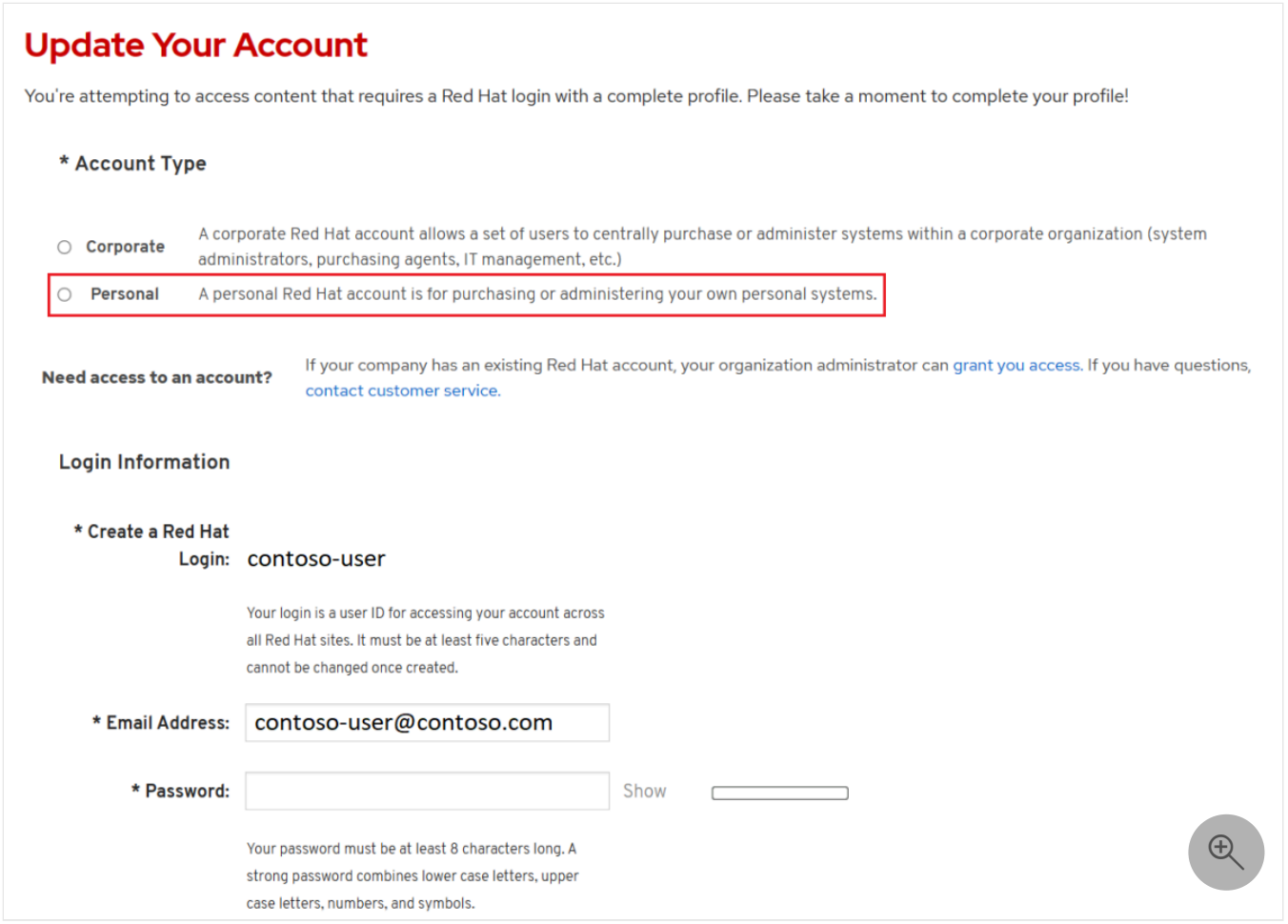
- An Azure subscription.
- A local machine with a Unix-like operating system installed (for example, Ubuntu, macOS, or Windows Subsystem for Linux).
- The Azure CLI. If you're running on Windows or macOS, consider running Azure CLI in a Docker container. For more information, see How to run the Azure CLI in a Docker container.
  - Sign in to the Azure CLI by using the az login command. To finish the authentication process, follow the steps displayed in your terminal. For other sign-in options, see Sign into Azure with Azure CLI.
  - When you're prompted, install the Azure CLI extension on first use. For more information about extensions, see Use and manage extensions with the Azure CLI.
  - Run az version to find the version and dependent libraries that are installed. To upgrade to the latest version, run az upgrade. This article requires at least version 2.61.0 of Azure CLI.
- A Java Standard Edition (SE) implementation, version 17 (for example, Eclipse Open J9).
- Maven version 3.9.8 or higher.
- Docker for your OS.
- The Azure identity you use to sign in has either the Contributor role and the User Access Administrator role or the Owner role in the current subscription. For an overview of Azure roles, see What is Azure role-based access control (Azure RBAC)?

Get a Red Hat pull secret

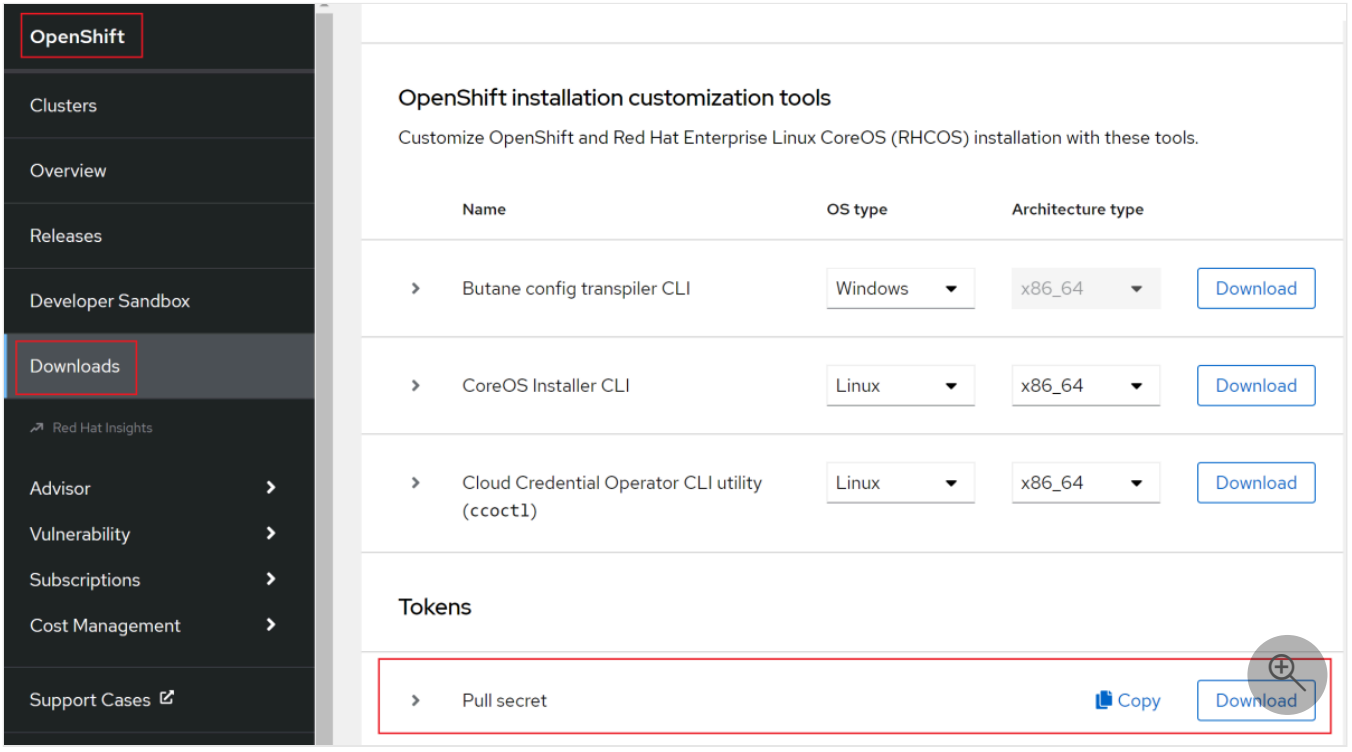
The Azure Marketplace offer you're going to use in this article requires a Red Hat pull secret. This section shows you how to get a Red Hat pull secret for Azure Red Hat OpenShift. To learn about what a Red Hat pull secret is and why you need it, see the Get a Red Hat pull secret section of Create an Azure Red Hat OpenShift 4 cluster. To get the pull secret for use, follow the steps in this section.

Use your Red Hat account to sign in to the OpenShift cluster manager portal, by visiting the Red Hat OpenShift Hybrid Cloud Console . You might need to accept more terms and update

your account as shown in the following screenshot. Use the same password as when you created the account.



After you sign in, select **OpenShift**, then **Downloads**. Select the **All categories** dropdown list and then select **Tokens**. Under **Pull secret**, select **Copy** or **Download**, as shown in the following screenshot.



The following content is an example that was copied from the Red Hat console portal, with the auth codes replaced with `xxxx...xxx`.

JSON

```
{"auths":{"cloud.openshift.com":{"auth":"xxxx...xxx","email":"contoso-
user@contoso.com"},"quay.io":{"auth":"xxx...xxx","email":"contoso-
user@test.com"},"registry.connect.redhat.com":
{"auth":"xxxx...xxx","email":"contoso-user@contoso.com"},"registry.redhat.io":
{"auth":"xxxx...xxx","email":"contoso-user@contoso.com"}}}
```

Save the secret to a file so you can use it later.

Create a Microsoft Entra service principal from the Azure portal

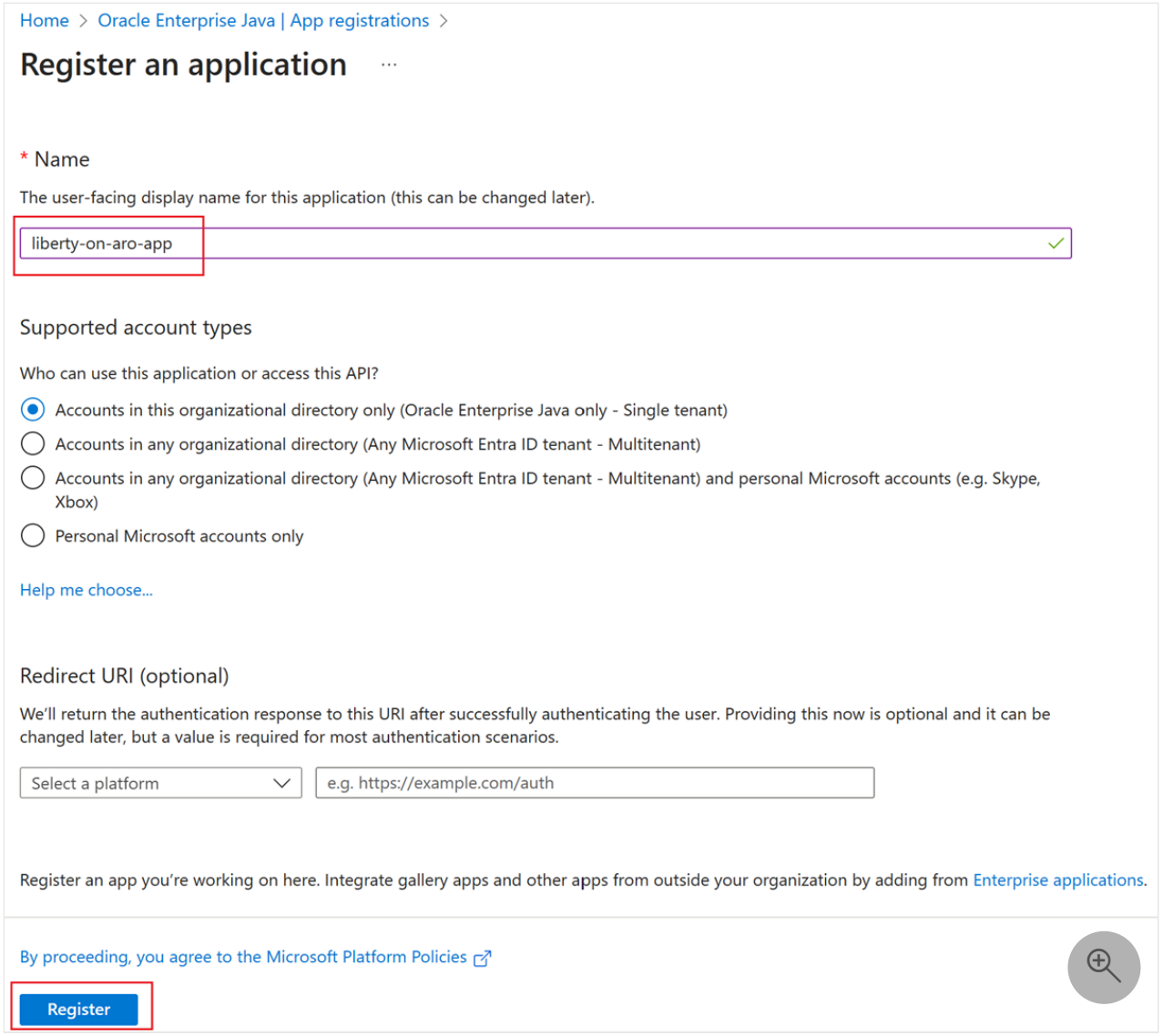
The Azure Marketplace offer you're going to use in this article requires a Microsoft Entra service principal to deploy your Azure Red Hat OpenShift cluster. The offer assigns the service principal with proper privileges during deployment time, with no role assignment needed. If you have a service principal ready to use, skip this section and move on to the next section, where you deploy the offer.

Use the following steps to deploy a service principal and get its Application (client) ID and secret from the Azure portal. For more information, see Create and use a service principal to deploy an Azure Red Hat OpenShift cluster.

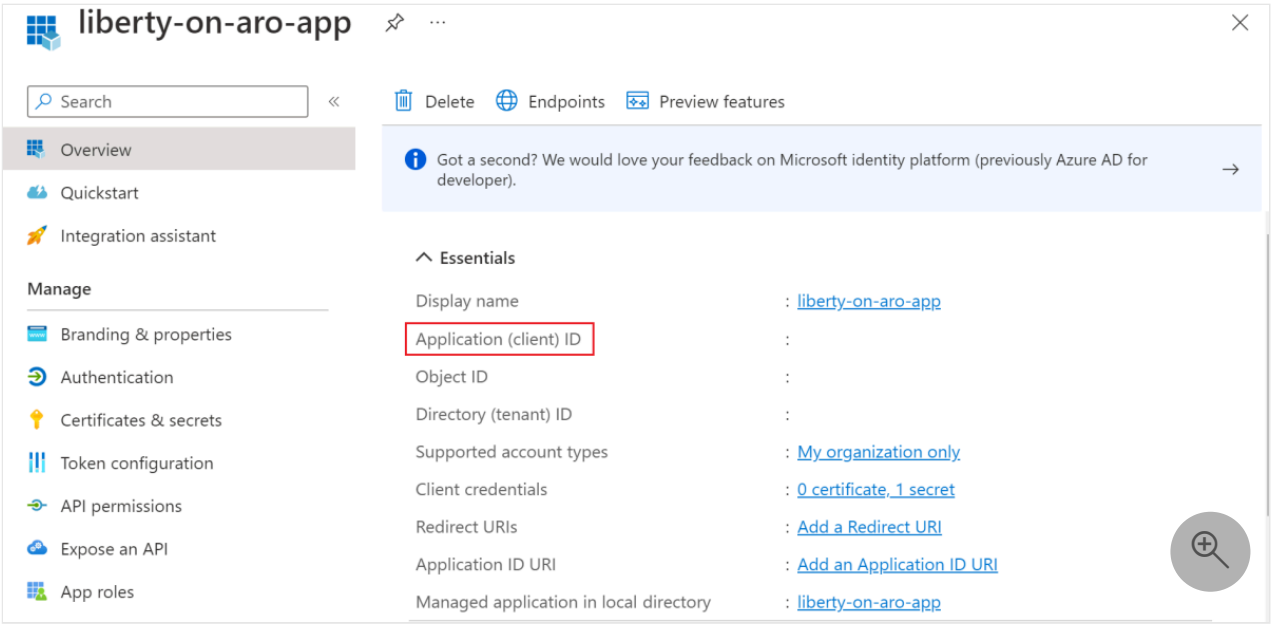
> **Note**
>
> You must have sufficient permissions to register an application with your Microsoft Entra tenant. If you run into a problem, check the required permissions to make sure your account can create the identity. For more information, see Register a Microsoft Entra app and create a service principal.

1. Sign in to your Azure account through the Azure portal.
2. Select **Microsoft Entra ID**.
3. Select **Manage**, then **App registrations**.
4. Select **New registration**.
5. Name the application, for example "liberty-on-aro-app". Select a supported account type, which determines who can use the application. After setting the values, select **Register**, as

shown in the following screenshot. It takes several seconds to provision the application. Wait for the deployment to complete before proceeding.



6. Save the Application (client) ID from the overview page, as shown in the following screenshot. Hover the pointer over the value (redacted in the screenshot) and select the copy icon that appears. The tooltip says **Copy to clipboard**. Be careful to copy the correct value, since the other values in that section also have copy icons. Save the Application ID to a file so you can use it later.



7. Create a new client secret by following these steps:
   a. Select **Manage**, then **Certificates & secrets**.
   b. Select **Client secrets**, then **New client secret**.
   c. Provide a description of the secret and a duration. When you're done, select **Add**.
   d. After the client secret is added, the value of the client secret is displayed. Copy this value because you can't retrieve it later.

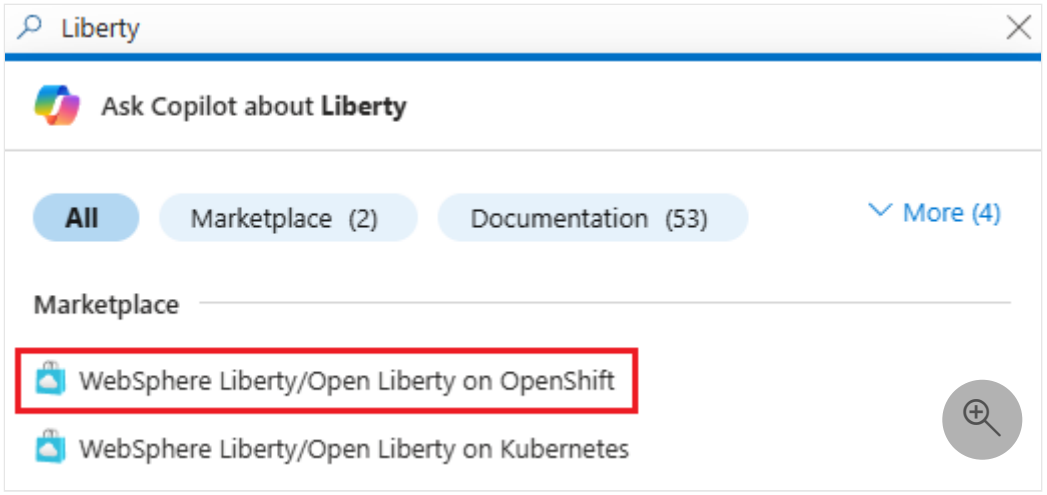
You now have a Microsoft Entra application, service principal, and client secret.

Deploy IBM WebSphere Liberty or Open Liberty on Azure Red Hat OpenShift

The steps in this section direct you to deploy IBM WebSphere Liberty or Open Liberty on Azure Red Hat OpenShift.

The following steps show you how to find the offer and fill out the **Basics** pane.

1. In the search bar at the top of the Azure portal, enter *Liberty*. In the autosuggested search results, in the **Marketplace** section, select **WebSphere Liberty/Open Liberty on OpenShift**, as shown in the following screenshot.



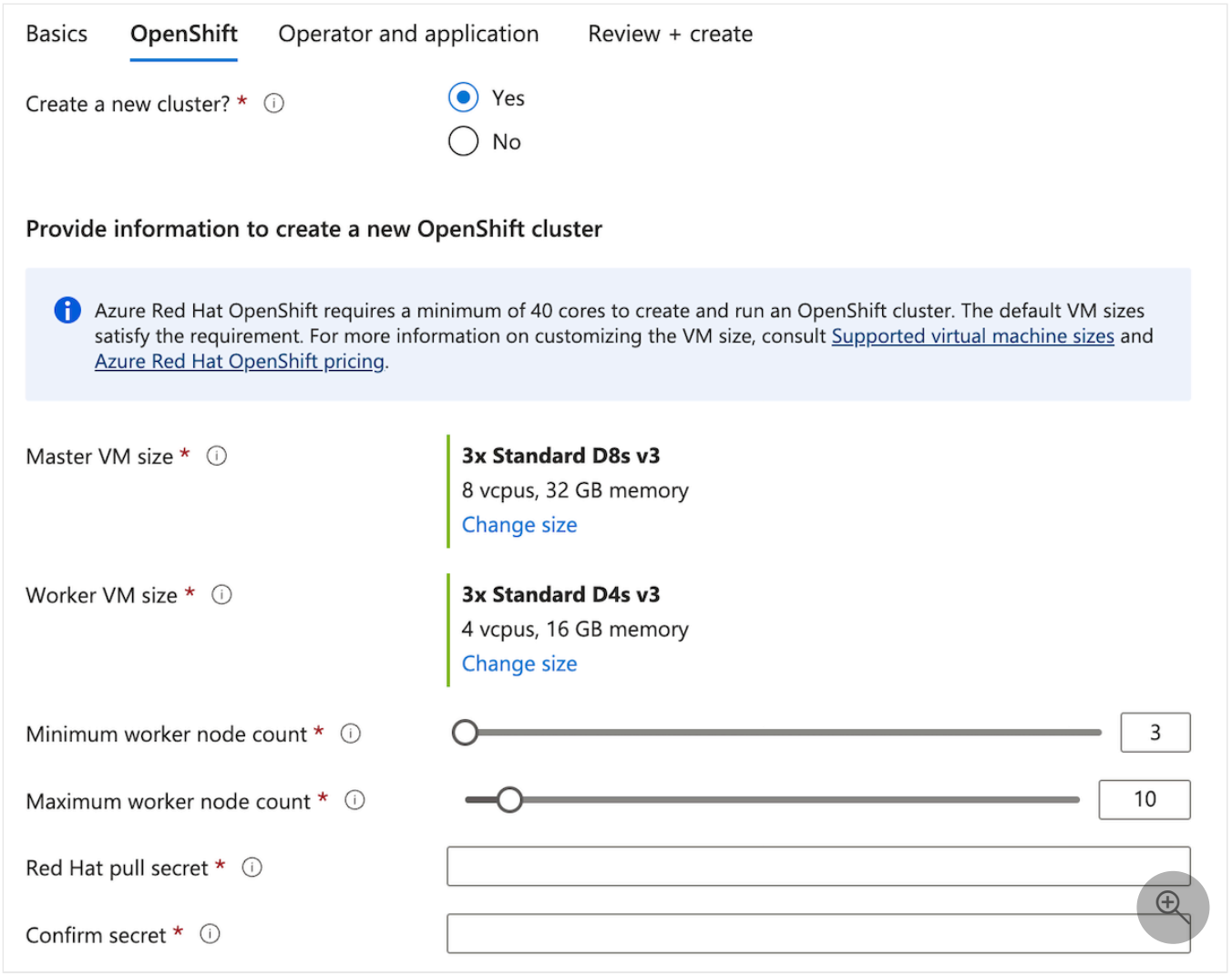
You can also go directly to the offer with this portal link.

2. On the offer page, select **Create**.
3. On the **Basics** pane, ensure that the value shown in the **Subscription** field is the same one that has the roles listed in the prerequisites section.
4. In the **Resource group** field, select **Create new** and fill in a value for the resource group. Because resource groups must be unique within a subscription, pick a unique name. An easy way to have unique names is to use a combination of your initials, today's date, and some identifier - for example, *abc1228rg*.
5. Create an environment variable in your shell for the resource group name.

```bash
export RESOURCE_GROUP_NAME=<your-resource-group-name>
```

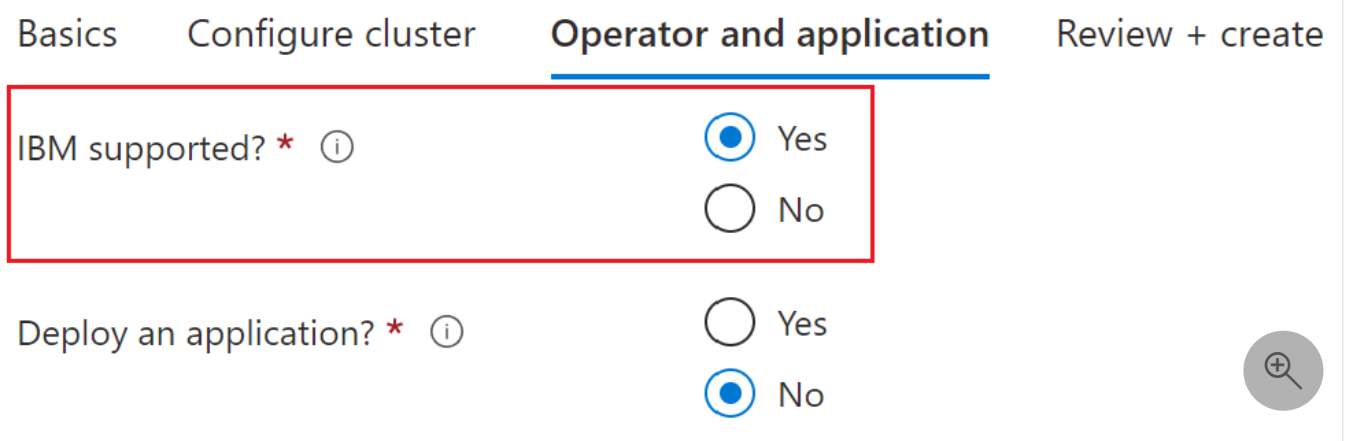
6. Under **Instance details**, select the region for the deployment. For a list of Azure regions where OpenShift operates, see Regions for Red Hat OpenShift 4.x on Azure.
7. After selecting the region, select **Next**.

The following steps show you how to fill out the **OpenShift** pane shown in the following screenshot:



1. Under **Create a new cluster**, select **Yes**.
2. Under **Provide information to create a new OpenShift cluster**, for **Master VM size**, **Worker VM size**, **Minimum worker node count**, and **Maximum worker node count**, leave the values at their defaults. For more information on how to use these advanced options, see OpenShift Virtualization - Tuning & Scaling Guide.
3. For **Red Hat pull secret**, fill in the Red Hat pull secret that you obtained in the Get a Red Hat pull secret section. Use the same value for **Confirm secret**.
4. Fill in **Service principal client ID** with the service principal Application (client) ID that you obtained in the Create a Microsoft Entra service principal from the Azure portal section.
5. Fill in **Service principal client secret** with the service principal Application secret that you obtained in the Create a Microsoft Entra service principal from the Azure portal section. Use the same value for **Confirm secret**.
6. After filling in the values, select **Next**.

The following steps show you how to fill out the **Operator and application** pane shown in the following screenshot, and start the deployment.



1. Under **IBM supported?**, select **Yes**.

   > **Note**
   >
   > This quickstart deploys the IBM-supported WebSphere Liberty Operator, but you can select **No** to deploy the Open Liberty Operator instead.

2. Leave the default option of **No** for **Deploy an application?**.

   > **Note**
   >
   > This quickstart manually deploys a sample application later, but you can select **Yes** for **Deploy an application?** if you prefer.

3. Select **Review + create**. Fix any validation problems and then select **Review + create** again.

4. Select **Create**.

5. Track the progress of the deployment on the **Deployment is in progress** page.

Depending on network conditions and other activity in your selected region, the deployment might take up to 40 minutes to complete.

Verify the functionality of the deployment

The steps in this section show you how to verify that the deployment completed successfully.

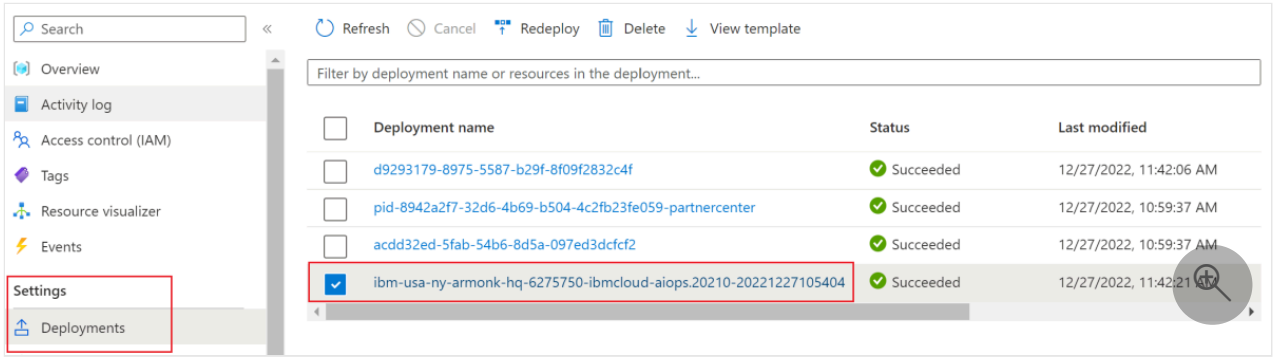
If you navigated away from the **Deployment is in progress** page, the following steps show you how to get back to that page. If you're still on the page that shows **Your deployment is complete**, you can skip to step 5.

1. In the corner of any portal page, select the hamburger menu and then select **Resource groups**.

2. In the box with the text **Filter for any field**, enter the first few characters of the resource group you created previously. If you followed the recommended convention, enter your initials, then select the appropriate resource group.

3. In the navigation pane, in the **Settings** section, select **Deployments**. You see an ordered list of the deployments to this resource group, with the most recent one first.

4. Scroll to the oldest entry in this list. This entry corresponds to the deployment you started in the preceding section. Select the oldest deployment, as shown in the following screenshot.



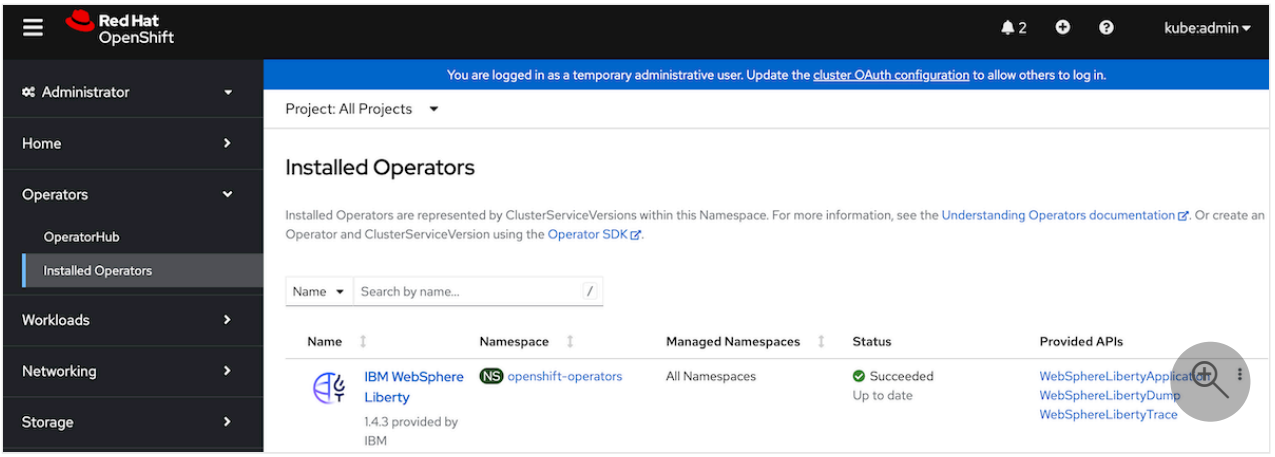
5. In the navigation pane, select **Outputs**. This list shows the output values from the deployment, which includes some useful information.

6. Open your terminal and paste the value from the **cmdToGetKubeadminCredentials** field. You see the admin account and credential for logging in to the OpenShift cluster console portal. The following content is an example of an admin account.

```Bash
az aro list-credentials --resource-group abc1228rg --name clusterf9e8b9
{
  "kubeadminPassword": "xxxxx-xxxxx-xxxxx-xxxxx",
  "kubeadminUsername": "kubeadmin"
}
```

7. Paste the value from the **clusterConsoleUrl** field into an Internet-connected web browser, and then press `Enter`. Fill in the admin user name and password and sign in.

8. Verify that the appropriate Kubernetes operator for Liberty is installed. In the navigation pane, select **Operators**, then **Installed Operators**, as shown in the following screenshot:



Take note if you installed the WebSphere Liberty operator or the Open Liberty operator. The operator variant matches what you selected at deployment time. If you selected **IBM Supported**, you have the WebSphere Liberty operator. Otherwise you have the Open Liberty operator. This information is important to know in later steps.

9. Download and install the OpenShift CLI `oc` by following steps in tutorial Install the OpenShift CLI, then return to this documentation.

10. Switch to **Outputs** pane, copy the value from the **cmdToLoginWithKubeadmin** field, and then paste it in your terminal. Run the command to sign in to the OpenShift cluster's API server. You should see output similar to the following example in the console:

    Output

    ```
    Login successful.

    You have access to 73 projects, the list has been suppressed. You can list
    all projects with 'oc projects'

    Using project "default".
    ```

Create an Azure SQL Database

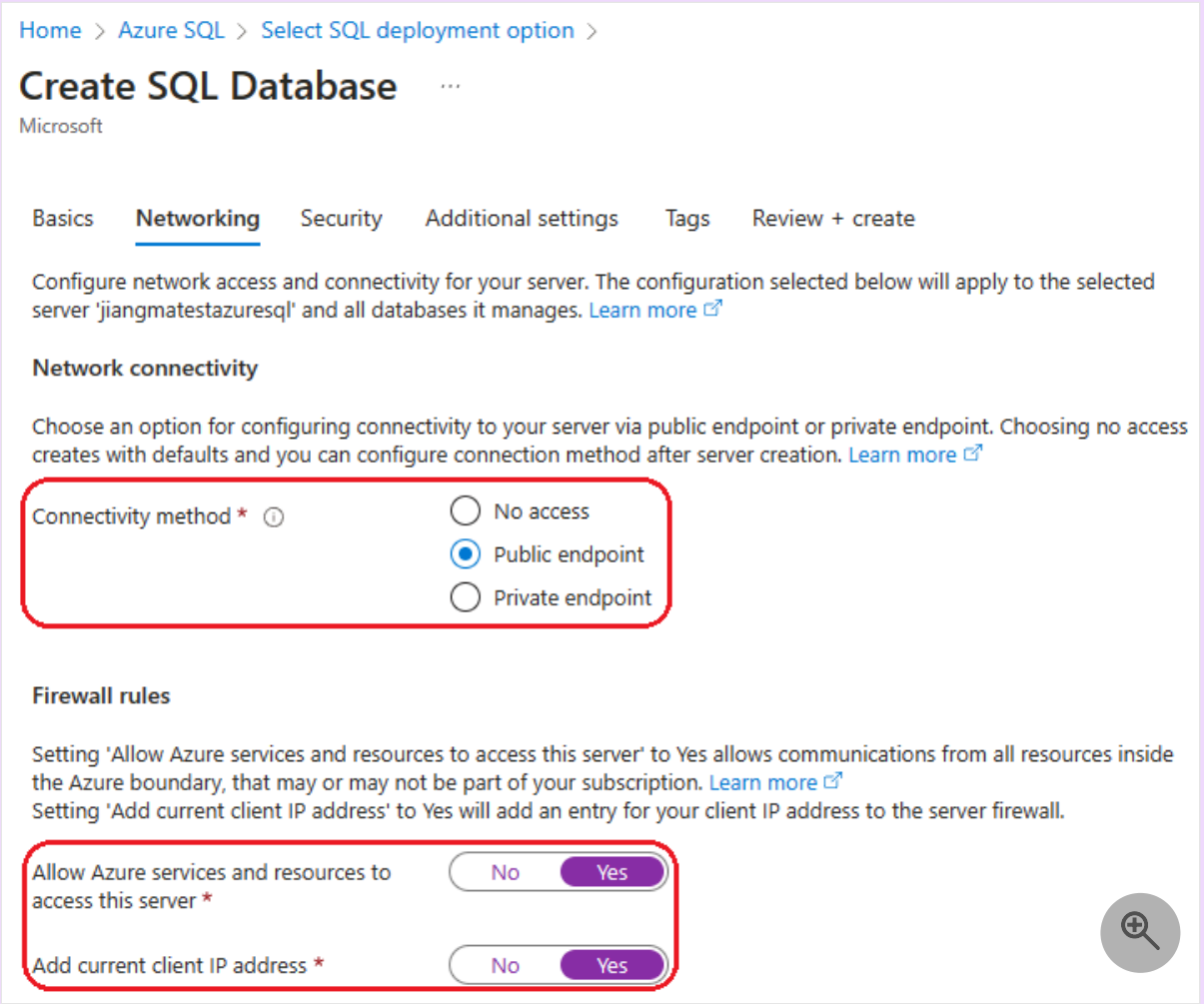
The following steps guide you through creating an Azure SQL Database single database for use with your app:

1. Create a single database in Azure SQL Database by following the steps in Quickstart: Create an Azure SQL Database single database, carefully noting the differences described in the following note. You can deploy the database to the same resource group as the OpenShift cluster. Return to this article after creating and configuring the database server.

    > **Note**

At the **Basics** step, write down the values for **Resource group**, **Database name**, ***<server-name>*.database.windows.net**, **Server admin login**, and **Password**. The database **Resource group** is referred to as `<db-resource-group>` later in this article.

At the **Networking** step, set **Connectivity method** to **Public endpoint**, **Allow Azure services and resources to access this server** to **Yes**, and **Add current client IP address** to **Yes**.



2. Create an environment variable in your shell for the resource group name for the database.

```bash
export DB_RESOURCE_GROUP_NAME=<db-resource-group>
```

> **Note**
>
> This article guides you to create an Azure SQL Database single database with SQL authentication. A more secure practice is to use Microsoft Entra authentication for Azure SQL for authenticating the database server connection. Azure Red Hat OpenShift doesn't

currently support Microsoft Entra Workload ID, so SQL authentication is the only available option.

Now that you created the database and Azure Red Hat OpenShift cluster, you can prepare the Azure Red Hat OpenShift cluster to host your WebSphere Liberty application.

Configure and deploy the sample application

Follow the steps in this section to deploy the sample application on the Liberty runtime. These steps use Maven.

Check out the application

Clone the sample code for this guide by using the following commands. The sample is on GitHub.

Bash

```
git clone https://github.com/Azure-Samples/open-liberty-on-aro.git
cd open-liberty-on-aro
export BASE_DIR=$PWD
git checkout 20250501
cd 3-integration/connect-db/mssql
```

If you see a message about being in "detached HEAD" state, this message is safe to ignore. It just means you checked out a tag.

There are a few samples in the repository. We use *3-integration/connect-db/mssql/*. Here's the file structure of the application:

```
mssql
├─ src/main/
│  ├─ aro/
│  │  ├─ db-secret.yaml
│  │  ├─ openlibertyapplication.yaml
│  │  ├─ webspherelibertyapplication.yaml
│  ├─ docker/
│  │  ├─ Dockerfile
│  │  ├─ Dockerfile-ol
│  ├─ liberty/config/
│  │  ├─ server.xml
│  ├─ java/
│  ├─ resources/
```

```
│   ├─ webapp/
├─ pom.xml
```

The directories *java*, *resources*, and *webapp* contain the source code of the sample application. The code declares and uses a data source named `jdbc/JavaEECafeDB`.

In the *aro* directory, there are three deployment files. *db-secret.xml* is used to create Kubernetes Secrets with database connection credentials. The file *websphereliberty application.yaml* is used in this quickstart to deploy the WebSphere Liberty Application. Use the file *openlibertyapplication.yaml* to deploy the Open Liberty Application if you deployed Open Liberty Operator in section Deploy IBM WebSphere Liberty or Open Liberty on Azure Red Hat OpenShift.

In the *docker* directory, there are two files to create the application image with either Open Liberty or WebSphere Liberty. These files are *Dockerfile* and *Dockerfile-ol*, respectively. You use the file *Dockerfile* to build the application image with WebSphere Liberty in this quickstart. Similarly, use the file *Dockerfile-ol* to build the application image with Open Liberty if you deployed Open Liberty Operator in section Deploy IBM WebSphere Liberty or Open Liberty on Azure Red Hat OpenShift.

In directory *liberty/config*, the *server.xml* file is used to configure the database connection for the Open Liberty and WebSphere Liberty cluster.

Build the project

Now that you gathered the necessary properties, you can build the application by using the following commands. The POM file for the project reads many variables from the environment. As part of the Maven build, these variables are used to populate values in the YAML files located in *src/main/aro*. You can do something similar for your application outside Maven if you prefer.

Bash

```
cd ${BASE_DIR}/3-integration/connect-db/mssql

# The following variables are used for deployment file generation into target.
export DB_SERVER_NAME=<server-name>.database.windows.net
export DB_NAME=<database-name>
export DB_USER=<server-admin-login>@<server-name>
export DB_PASSWORD='<server-admin-password>'

mvn clean install
```

(Optional) Test your project locally

You can now run and test the project locally before deploying to Azure by using the following steps. For convenience, we use the `liberty-maven-plugin`. To learn more about the `liberty-maven-plugin`, see Building a web application with Maven. For your application, you can do something similar using any other mechanism, such as your local IDE. You can also consider using the `liberty:devc` option intended for development with containers. You can read more about `liberty:devc` in the Liberty docs.

1. Start the application by using `liberty:run`, as shown in the following example. `liberty:run` also uses the environment variables defined in the previous section.

   Bash

   ```
   cd ${BASE_DIR}/3-integration/connect-db/mssql
   mvn liberty:run
   ```

2. Verify that the application works as expected. You should see a message similar to `[INFO] [AUDIT] CWWKZ0003I: The application javaee-cafe updated in 1.930 seconds.` in the command output if successful. Go to `http://localhost:9080/` or `https://localhost:9443/` in your browser and verify the application is accessible and all functions are working.

3. Press `Ctrl` + `C` to stop.

Next, use the following steps to containerize your project using Docker and run it as a container locally before deploying to Azure:

1. Use the following commands to build the image:

   Bash

   ```
   cd ${BASE_DIR}/3-integration/connect-db/mssql/target
   docker buildx build --platform linux/amd64 -t javaee-cafe:v1 --pull --
   file=Dockerfile .
   ```

2. Run the image using the following command. Note we're using the environment variables defined previously.

   Bash

   ```
   docker run -it --rm -p 9080:9080 -p 9443:9443 \
       -e DB_SERVER_NAME=${DB_SERVER_NAME} \
       -e DB_NAME=${DB_NAME} \
   ```

```
    -e DB_USER=${DB_USER} \
    -e DB_PASSWORD=${DB_PASSWORD} \
    javaee-cafe:v1
```

3. Once the container starts, go to `http://localhost:9080/` or `https://localhost:9443/` in your browser to access the application.

4. Press `Ctrl` + `C` to stop.

Build image and push to the image stream

When you're satisfied with the state of the application, you build the image remotely on the cluster by using the following steps.

1. Use the following commands to identity the source directory and the Dockerfile:

```
Bash

cd ${BASE_DIR}/3-integration/connect-db/mssql/target

# If you are deploying the application with WebSphere Liberty Operator, the
existing Dockerfile is ready for you

# If you are deploying the application with Open Liberty Operator, uncomment
and execute the following two commands to rename Dockerfile-ol to Dockerfile
# mv Dockerfile Dockerfile.backup
# mv Dockerfile-ol Dockerfile
```

2. Use the following command to create an image stream:

```
Bash

oc create imagestream javaee-cafe
```

3. Use the following command to create a build configuration that specifies the image stream tag of the build output:

```
Bash

oc new-build --name javaee-cafe-config --binary --strategy docker --to
javaee-cafe:v1
```

4. Use the following command to start the build to upload local contents, containerize, and output to the image stream tag specified before:

```
```

Bash

```
oc start-build javaee-cafe-config --from-dir . --follow
```

Deploy and test the application

Use the following steps to deploy and test the application:

1. Use the following command to apply the database secret:

   Bash

   ```
   cd ${BASE_DIR}/3-integration/connect-db/mssql/target
   oc apply -f db-secret.yaml
   ```

   You should see the output `secret/db-secret-mssql created`.

2. Use the following command to apply the deployment file:

   Bash

   ```
   oc apply -f websphereliberty application.yaml
   ```

3. Wait until all pods are started and running successfully by using the following command:

   Bash

   ```
   oc get pods -l app.kubernetes.io/name=javaee-cafe --watch
   ```

   You should see output similar to the following example to indicate that all the pods are running:

   Output

   ```
   NAME                          READY   STATUS    RESTARTS   AGE
   javaee-cafe-67cdc95bc-2j2gr   1/1     Running   0          29s
   javaee-cafe-67cdc95bc-fgtt8   1/1     Running   0          29s
   javaee-cafe-67cdc95bc-h47qm   1/1     Running   0          29s
   ```

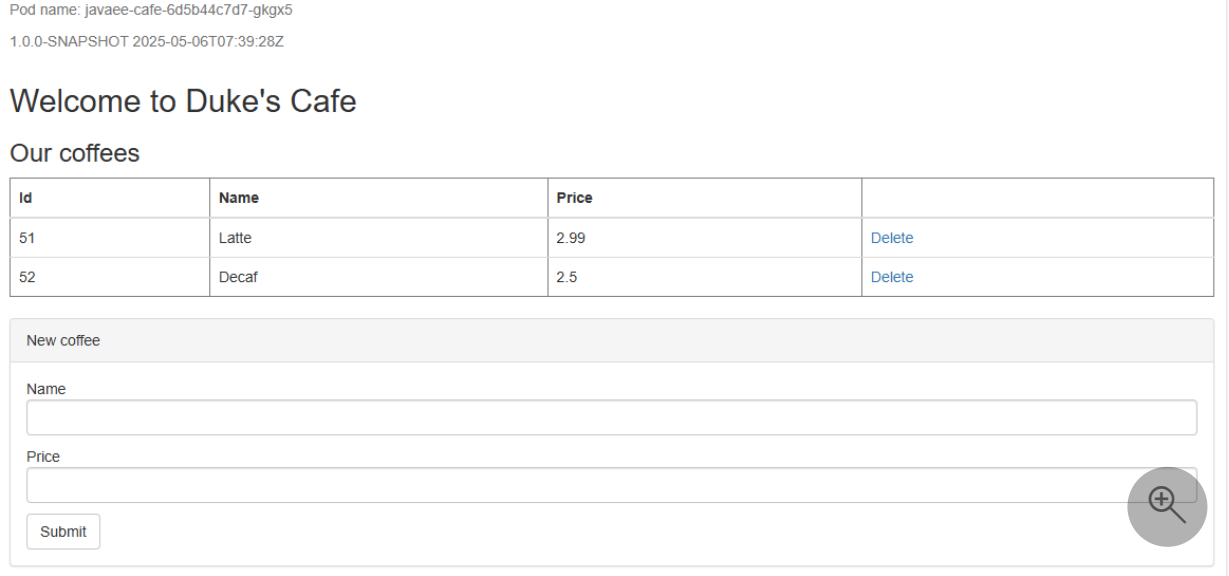
4. Use the following steps to verify the results:

   a. Use the following command to get the *host* of the Route resource deployed with the application:

```bash
echo "route host: https://$(oc get route javaee-cafe --template='{{
.spec.host }}')"
```

b. Copy the value of `route host` from the output, open it in your browser, and test the application. If the web page doesn't render correctly, that's because the app is still starting in the background. Wait for a few minutes and then try again.

c. Add and delete a few coffees to verify the functionality of the app and the database connection.



Clean up resources

To avoid Azure charges, you should clean up unnecessary resources. When the cluster is no longer needed, use the az group delete command to remove the resource group, Azure Red Hat OpenShift cluster, Azure SQL Database, and all related resources.

```bash
az group delete --name $RESOURCE_GROUP_NAME --yes --no-wait
az group delete --name $DB_RESOURCE_GROUP_NAME --yes --no-wait
```

Next steps

For more information about deploying the IBM WebSphere family on Azure, see What are solutions to run the WebSphere family of products on Azure?

Quickstart: Deploy JBoss EAP on Azure Red Hat OpenShift

This article shows you how to quickly set up JBoss Enterprise Application Platform (EAP) on Azure Red Hat OpenShift using the Azure portal.

This article uses the Azure Marketplace offer for JBoss EAP to accelerate your journey to Azure Red Hat OpenShift. The offer automatically provisions resources including an Azure Red Hat OpenShift cluster with a built-in OpenShift Container Registry (OCR), the JBoss EAP Operator, and optionally a container image including JBoss EAP and your application using Source-to-Image (S2I). To see the offer, visit the Azure portal. If you prefer manual step-by-step guidance for running JBoss EAP on Azure Red Hat OpenShift that doesn't use the automation enabled by the offer, see Deploy a Java application with Red Hat JBoss Enterprise Application Platform (JBoss EAP) on an Azure Red Hat OpenShift 4 cluster.

If you're interested in providing feedback or working closely on your migration scenarios with the engineering team developing JBoss EAP on Azure solutions, fill out this short survey on JBoss EAP migration and include your contact information. The team of program managers, architects, and engineers will promptly get in touch with you to initiate close collaboration.

Prerequisites

- An Azure subscription. If you don't have an Azure account, create a free account before you begin.
- A Red Hat account with complete profile. If you don't have one, you can sign up for a free developer subscription through the Red Hat Developer Subscription for Individuals.
- A local developer command line with a UNIX-like command environment - for example, Ubuntu, macOS, or Windows Subsystem for Linux - and Azure CLI installed. To learn how to install the Azure CLI, see How to install the Azure CLI.
- The `mysql` CLI. For example, you can install the CLI by using the following commands on Ubuntu or Debian-based systems:

  Bash

  ```
  sudo apt update
  sudo apt install mysql-server
  ```

- An Azure identity that you use to sign in that has either the Contributor role and the User Access Administrator role or the Owner role in the current subscription. For an overview

of Azure roles, see What is Azure role-based access control (Azure RBAC)?

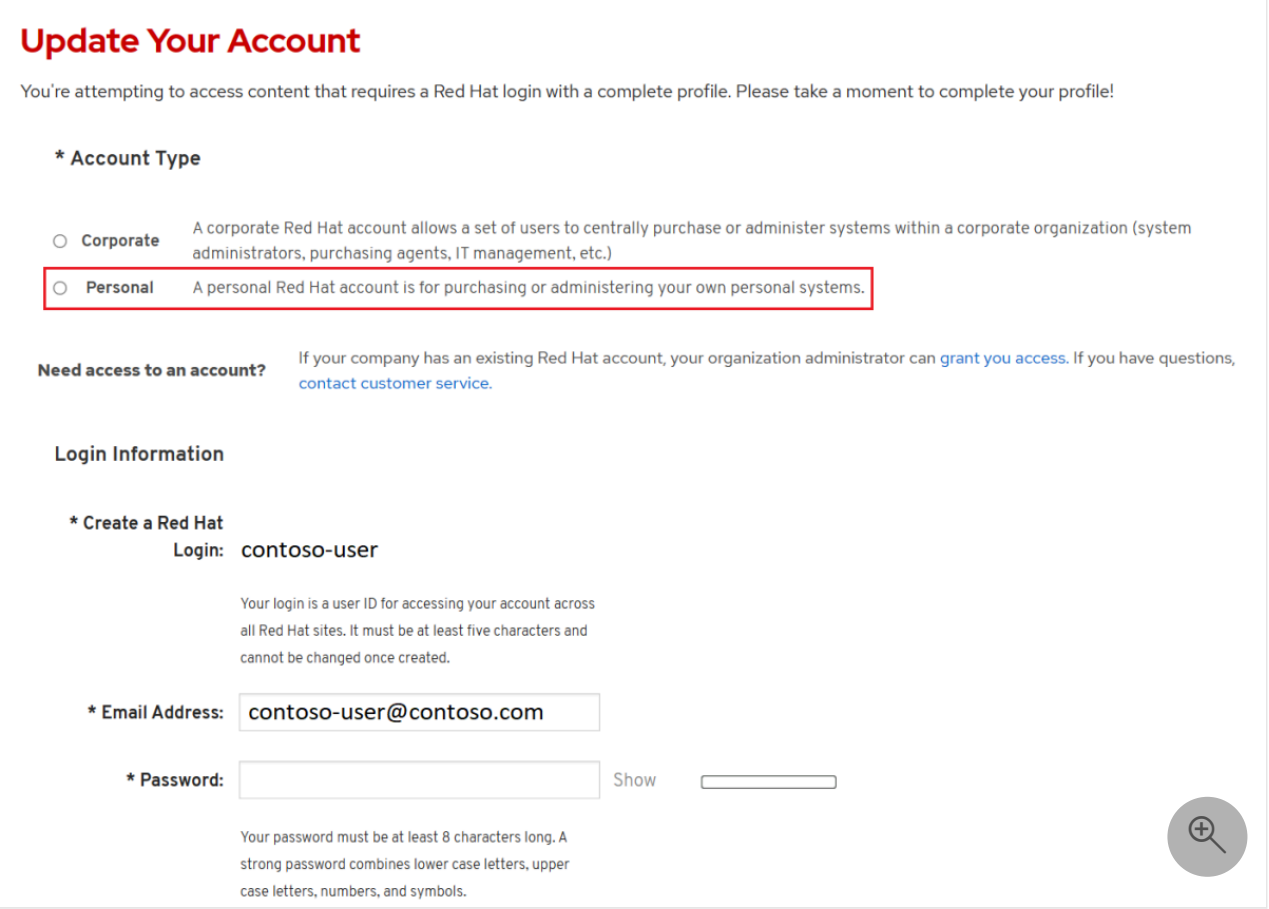
> **Note**
>
> Azure Red Hat OpenShift requires a minimum of 40 cores to create and run an OpenShift cluster. The default Azure resource quota for a new Azure subscription doesn't meet this requirement. To request an increase in your resource limit, see Increase VM-family vCPU quotas. Note that the free trial subscription isn't eligible for a quota increase. Upgrade to a Pay-As-You-Go subscription before you request a quota increase. For more information, see Upgrade your Azure free account or Azure for Students Starter account.

Get a Red Hat pull secret

The Azure Marketplace offer used in this article requires a Red Hat pull secret. This section shows you how to get a Red Hat pull secret for Azure Red Hat OpenShift. To learn about what a Red Hat pull secret is and why you need it, see the Get a Red Hat pull secret section of Tutorial: Create an Azure Red Hat OpenShift 4 cluster.

Use the following steps to get the pull secret:

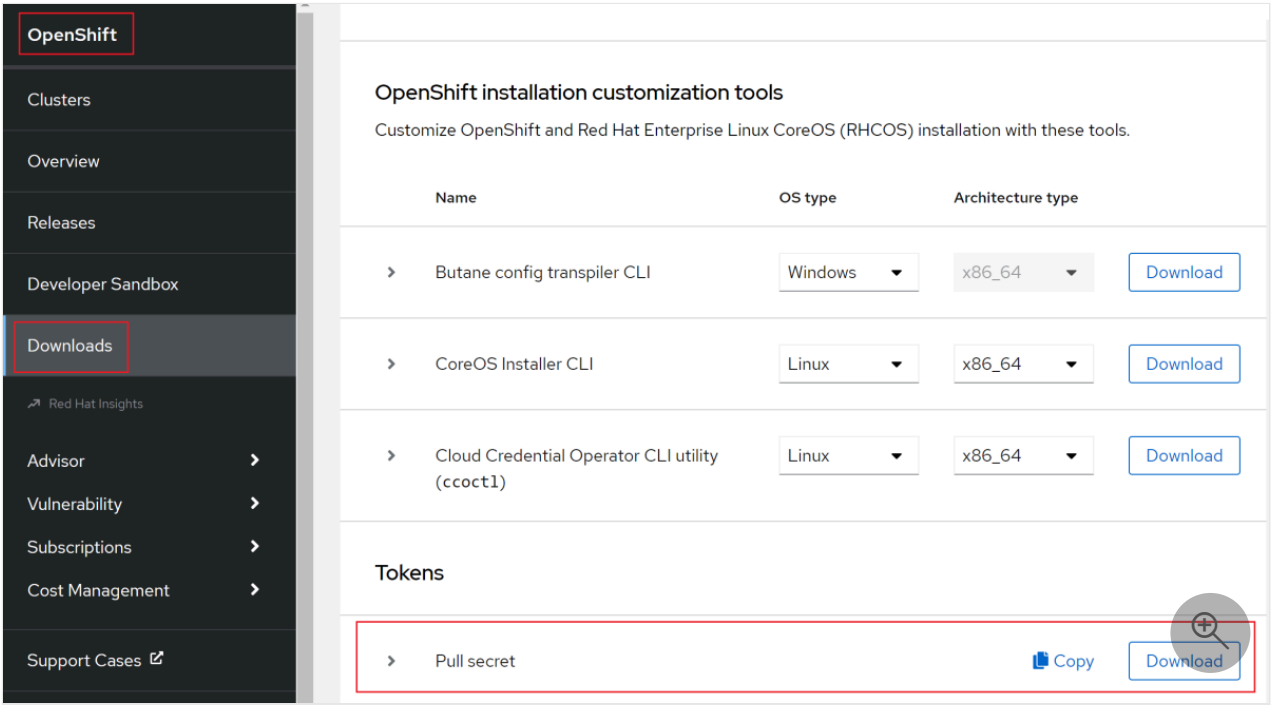
1. Open the Red Hat OpenShift Hybrid Cloud Console, then use your Red Hat account to sign in to the OpenShift cluster manager portal. You may need to accept more terms and update your account as shown in the following screenshot. Use the same password as when you created the account.



2. After you sign in, select **OpenShift** then **Downloads**.

3. Select the **All categories** dropdown list and then select **Tokens**.

4. Under **Pull secret**, select **Copy** or **Download** to get the value, as shown in the following screenshot.



The following content is an example that was copied from the Red Hat console portal, with the auth codes replaced with `xxxx...xxx`.

JSON

```
{"auths":{"cloud.openshift.com":{"auth":"xxxx...xxx","email":"contoso-user@contoso.com"},"quay.io":{"auth":"xxx...xxx","email":"contoso-user@test.com"},"registry.connect.redhat.com":{"auth":"xxxx...xxx","email":"contoso-user@contoso.com"},"registry.redhat.io":{"auth":"xxxx...xxx","email":"contoso-user@contoso.com"}}}
```

5. Save the secret to a file so you can use it later.

Create a Red Hat Container Registry service account

Later, this article shows you how to manually deploy an application to OpenShift using Source-to-Image (S2I). A Red Hat Container Registry service account is necessary to pull the container image for JBoss EAP on which to run your application. If you have a Red Hat Container Registry service account ready to use, skip this section and move on to the next section, where you deploy the offer.

Use the following steps to create a Red Hat Container Registry service account and get its username and password. For more information, see Creating Registry Service Accounts in the Red Hat documentation.

1. Use your Red Hat account to sign in to the Registry Service Account Management Application.
2. From the **Registry Service Accounts** page, select **New Service Account**.
3. Provide a name for the Service Account. The name is prepended with a fixed, random string.

    - Enter a description.
    - Select **create**.

4. Navigate back to your Service Accounts.
5. Select the Service Account you created.

    - Note down the **username**, including the prepended string (that is, `XXXXXXX|username`). Use this username when you sign in to `registry.redhat.io`.
    - Note down the **password**. Use this password when you sign in to `registry.redhat.io`.

You created your Red Hat Container Registry service account.

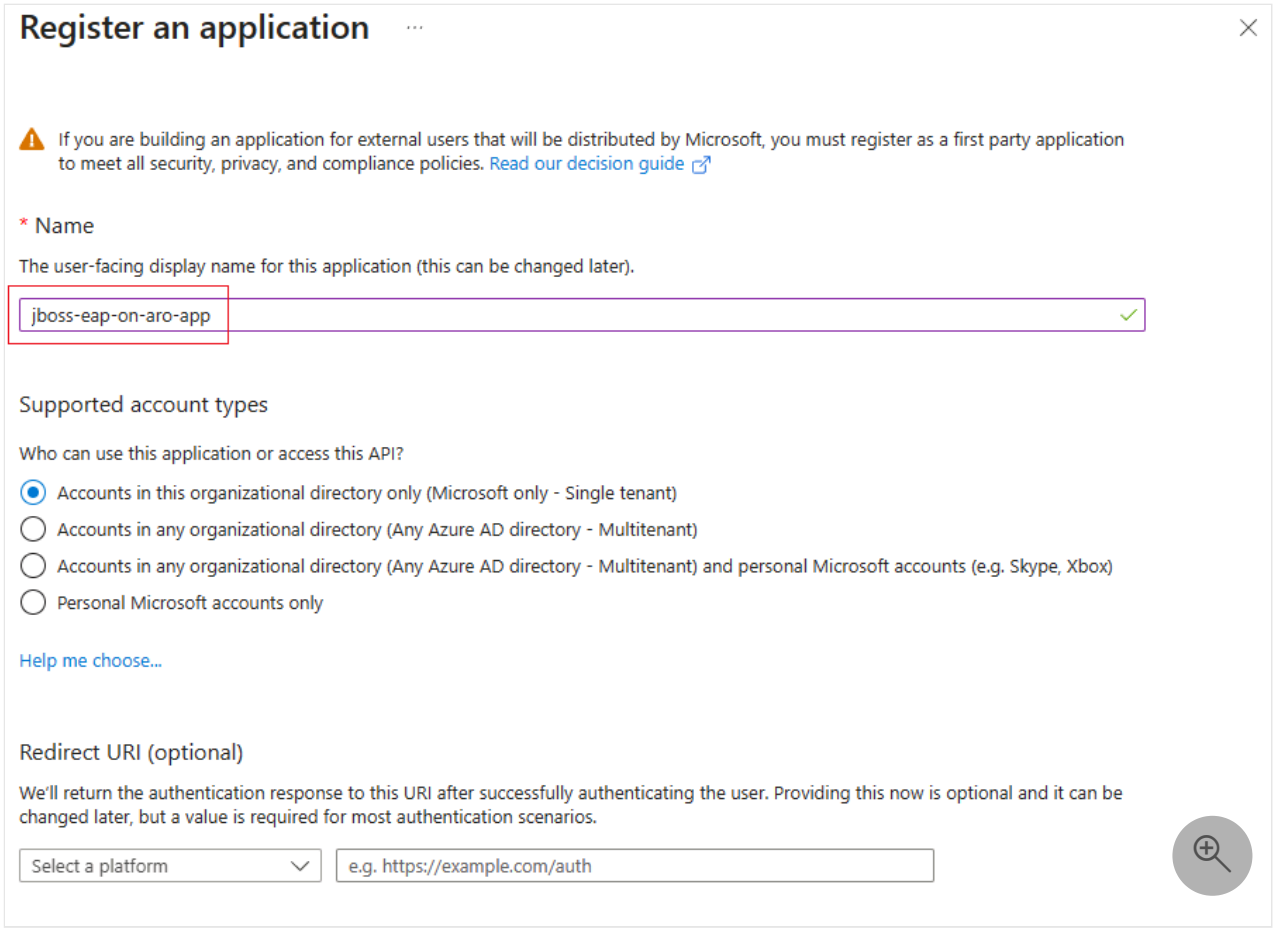
Create a Microsoft Entra service principal from the Azure portal

The Azure Marketplace offer used in this article requires a Microsoft Entra service principal to deploy your Azure Red Hat OpenShift cluster. The offer assigns the service principal with proper privileges during deployment time, with no role assignment needed. If you have a service principal ready to use, skip this section and move on to the next section, where you create a Red Hat Container Registry service account.

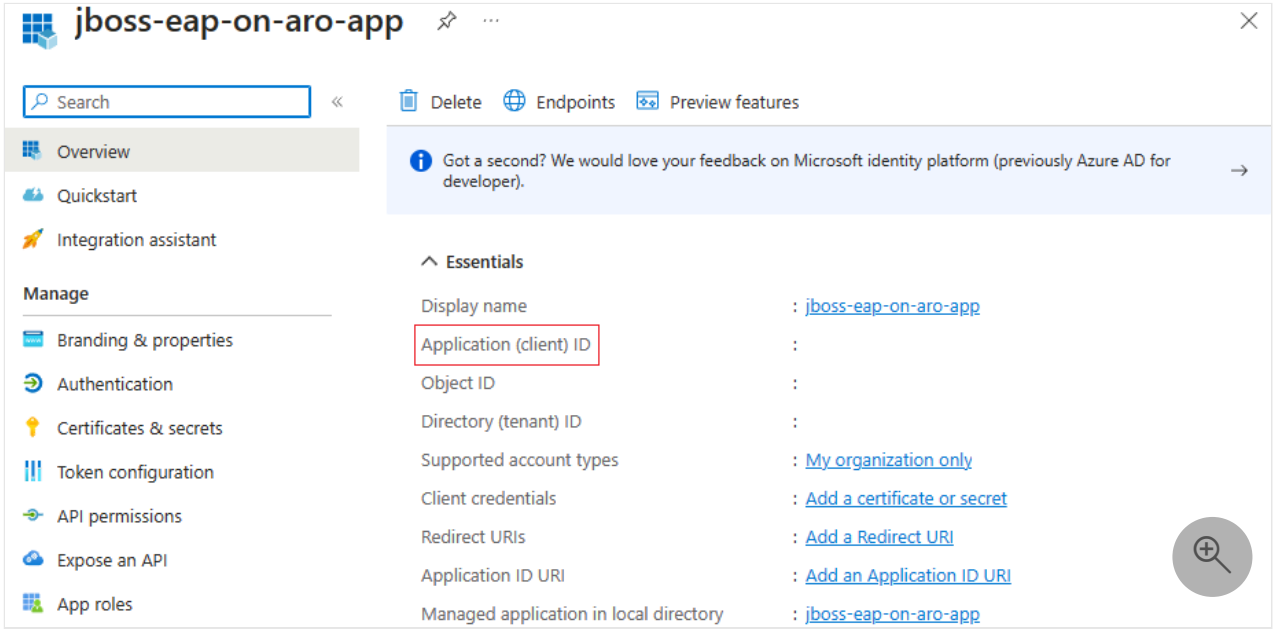
Use the following steps to deploy a service principal and get its Application (client) ID and secret from the Azure portal. For more information, see Create and use a service principal to deploy an Azure Red Hat OpenShift cluster.

> **Note**
>
> You must have sufficient permissions to register an application with your Microsoft Entra tenant. If you run into a problem, check the required permissions to make sure your account can create the identity. For more information, see Register a Microsoft Entra app and create a service principal.

1. Sign in to your Azure account through the Azure portal.
2. Select **Microsoft Entra ID**.
3. Select **App registrations**.
4. Select **New registration**.
5. Name the application - for example, `jboss-eap-on-aro-app`. Select a supported account type, which determines who can use the application. After setting the values, select **Register**, as shown in the following screenshot. It takes several seconds to provision the application. Wait for the deployment to complete before proceeding.



6. Save the Application (client) ID from the overview page, as shown in the following screenshot. Hover the pointer over the value, which is redacted in the screenshot, and select the copy icon that appears. The tooltip says **Copy to clipboard**. Be careful to copy the correct value, since the other values in that section also have copy icons. Save the Application ID to a file so you can use it later.



7. Create a new client secret by following these steps:
   a. Select **Certificates & secrets**.
   b. Select **Client secrets**, then **New client secret**.

c. Provide a description of the secret and a duration. When you're done, select **Add**.
d. After the client secret is added, the value of the client secret is displayed. Copy this value because you can't retrieve it later. Be sure to copy the **Value** and not the **Secret ID**.

You created your Microsoft Entra application, service principal, and client secret.

Validate the service principal

Use the following command to validate the service principal:

Azure CLI

```
az login \
    --service-principal \
    --username <service-principal-client-id> \
    --password <service-principal-client-secret> \
    --tenant <tenant-id>
az account show
```

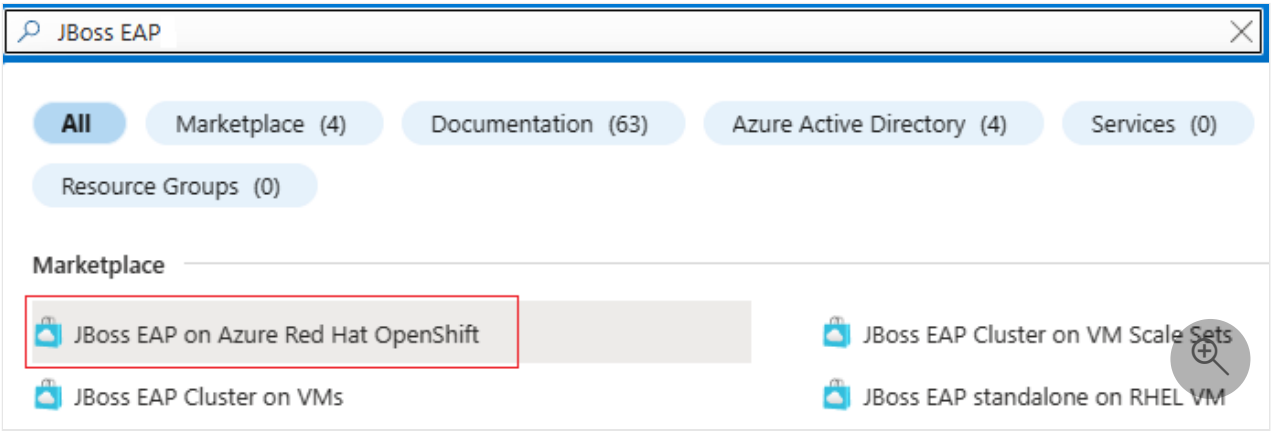
Replace `<service-principal-client-id>`, `<service-principal-client-secret>`, and `<tenant-id>` with the values you obtained in the previous steps. If you see the account information, the service principal is valid.

Deploy JBoss EAP on Azure Red Hat OpenShift

The steps in this section direct you to deploy JBoss EAP on Azure Red Hat OpenShift.

The following steps show you how to find the offer and fill out the **Basics** pane.

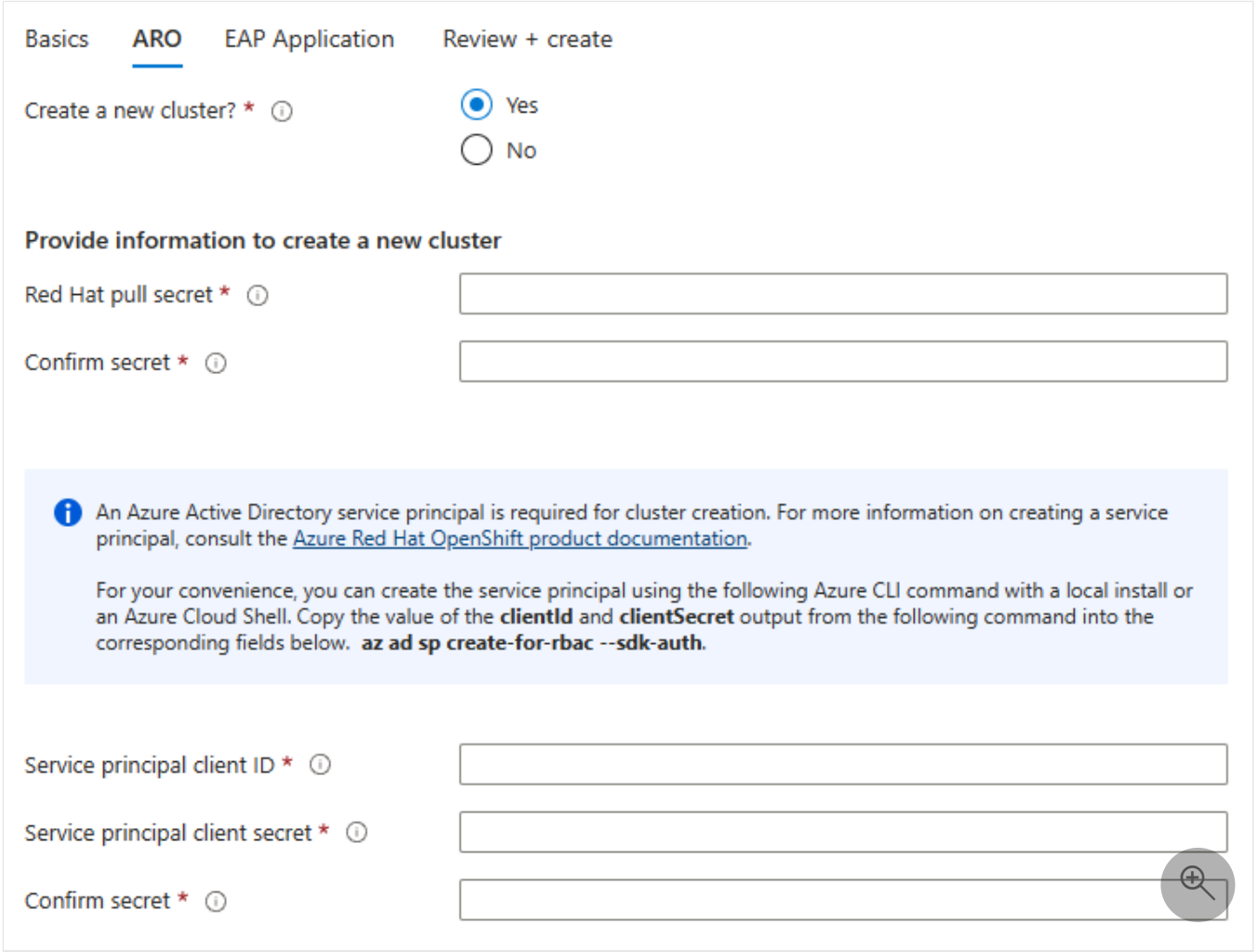
1. In the search bar at the top of the Azure portal, enter *JBoss EAP*. In the search results, in the **Marketplace** section, select **JBoss EAP on Azure Red Hat OpenShift**, as shown in the following screenshot.



You can also go directly to the JBoss EAP on Azure Red Hat OpenShift offer on the Azure portal.

2. On the offer page, select **Create**.
3. On the **Basics** pane, ensure that the value shown in the **Subscription** field is the same one that has the roles listed in the prerequisites section.
4. In the **Resource group** field, select **Create new** and fill in a value for the resource group. Because resource groups must be unique within a subscription, pick a unique name. An easy way to have unique names is to use a combination of your initials, today's date, and some identifier. For example, *eaparo033123rg*.
5. Under **Instance details**, select the region for the deployment. For a list of Azure regions where OpenShift operates, see Regions for Red Hat OpenShift 4.x on Azure.
6. Select **Next: ARO**.

The following steps show you how to fill out the **ARO** pane shown in the following screenshot:



1. Under **Create a new cluster**, select **Yes**.

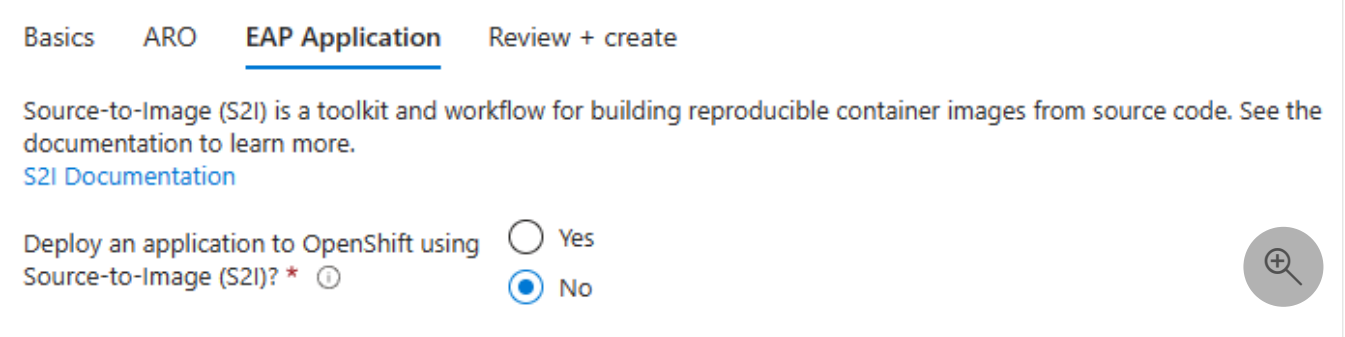
2. Under **Provide information to create a new cluster**, for **Red Hat pull secret**, fill in the Red Hat pull secret that you obtained in the Get a Red Hat pull secret section. Use the same value for **Confirm secret**.

3. Fill in **Service principal client ID** with the service principal Application (client) ID that you obtained in the Create a Microsoft Entra service principal from the Azure portal section.

4. Fill in **Service principal client secret** with the service principal Application secret that you obtained in the Create a Microsoft Entra service principal from the Azure portal section. Use the same value for **Confirm secret**.

5. Select **Next EAP Application**.

The following steps show you how to fill out the **EAP Application** pane shown in the following screenshot, and then start the deployment.



1. Leave the default option of **No** for **Deploy an application to OpenShift using Source-to-Image (S2I)?**.

   > **Note**
   >
   > Later, this quickstart shows you how to manually deploy an application with a database connection.

2. Select **Next: Review + create**.

3. Select **Review + create**. Ensure that the green **Validation Passed** message appears at the top. If the message doesn't appear, fix any validation problems, and then select **Review + create** again.

4. Select **Create**.

5. Track the progress of the deployment on the **Deployment is in progress** page.

Depending on network conditions and other activity in your selected region, the deployment might take up to 35 minutes to complete.

While you wait, you can set up the database.

Set up Azure Database for MySQL - Flexible Server

The following sections show you how to set up Azure Database for MySQL - Flexible Server.

Set environment variables in the command line shell

The sample is a Java application backed by a MySQL database, and is deployed to the OpenShift cluster using Source-to-Image (S2I). For more information about S2I, see the S2I Documentation.

> **Note**
>
> Because Azure Workload Identity is not yet supported by Azure OpenShift, this article still uses username and password for database authentication instead of using passwordless database connections.

Open a shell and set the following environment variables. Replace the substitutions as appropriate.

Bash

```
RG_NAME=<resource-group-name>
SERVER_NAME=<database-server-name>
DB_DATABASE_NAME=testdb
ADMIN_USERNAME=myadmin
ADMIN_PASSWORD=Secret123456
DB_USERNAME=testuser
DB_PASSWORD=Secret123456
PROJECT_NAME=eaparo-sample
CON_REG_SECRET_NAME=eaparo-sample-pull-secret
CON_REG_ACC_USER_NAME="<red-hat-container-registry-service-account-username>"
CON_REG_ACC_PWD="<red-hat-container-registry-service-account-password>"
APPLICATION_NAME=javaee-cafe
APP_REPLICAS=3
```

Replace the placeholders with the following values, which are used throughout the remainder of the article:

- `<resource-group-name>`: The name of resource group you created previously - for example, `eaparo033123rg`.
- `<database-server-name>`: The name of your MySQL server, which should be unique across Azure - for example, `eaparo033123mysql`.

- `ADMIN_PASSWORD`: The admin password of your MySQL database server. This article was tested using the password shown. Consult the database documentation for password rules.
- `<red-hat-container-registry-service-account-username>` and `<red-hat-container-registry-service-account-password>`: The username and password of the Red Hat Container Registry service account you created before.

It's a good idea to save the fully filled out name/value pairs in a text file, in case the shell exits before you're done executing the commands. That way, you can paste them into a new instance of the shell and easily continue.

These name/value pairs are essentially "secrets". For a production-ready way to secure Azure Red Hat OpenShift, including secret management, see Security for the Azure Red Hat OpenShift landing zone accelerator.

Create and initialize the database

Next, use the following steps to create an Azure Database for MySQL - Flexible Server, and create a user with permissions to read/write from/to the specific database.

1. Use the following command to create an Azure Database for MySQL - Flexible Server:

   Azure CLI

   ```
   az mysql flexible-server create \
       --resource-group ${RG_NAME} \
       --name ${SERVER_NAME} \
       --database-name ${DB_DATABASE_NAME} \
       --public-access 0.0.0.0  \
       --admin-user ${ADMIN_USERNAME} \
       --admin-password ${ADMIN_PASSWORD} \
       --yes
   ```

   This command might take ten or more minutes to complete. When the command successfully completes, you see output similar to the following example:

   Output

   ```
   {
     "connectionString": "mysql testdb --host
   ejb010406adb.mysql.database.azure.com --user myadmin --
   password=Secret#123345",
     "databaseName": "testdb",
     "firewallName": "AllowAllAzureServicesAndResourcesWithinAzureIps_2023-4-
   6_21-21-3",
     "host": "ejb010406adb.mysql.database.azure.com",
   ```

```
  "id":
"/subscriptions/redacted/resourceGroups/ejb010406a/providers/Microsoft.DBforMy
SQL/flexibleServers/ejb010406adb",
  "location": "East US",
  "password": "Secret#123345",
  "resourceGroup": "ejb010406a",
  "skuname": "Standard_B1ms",
  "username": "myadmin",
  "version": "5.7"
}
```

2. Use the following commands to get the host of the created MySQL server:

Azure CLI

```
DB_HOST=$(az mysql flexible-server show \
    --resource-group ${RG_NAME} \
    --name ${SERVER_NAME} \
    --query "fullyQualifiedDomainName" \
    --output tsv)
echo $DB_HOST
```

Save the name/value pair to your text file.

3. Use the following command to create a temporary firewall rule to allow connection to the MySQL server from the public internet:

Azure CLI

```
az mysql flexible-server firewall-rule create \
    --resource-group ${RG_NAME} \
    --name ${SERVER_NAME} \
    --rule-name "AllowAllIPs" \
    --start-ip-address 0.0.0.0 \
    --end-ip-address 255.255.255.255
```

4. Use the following command to create a new database user with permissions to read and write the specific database. This command is useful to send SQL directly to the database.

Bash

```
mysql --host ${DB_HOST} --user ${ADMIN_USERNAME} --password=${ADMIN_PASSWORD}
<< EOF
CREATE USER '${DB_USERNAME}'@'%' IDENTIFIED BY '${DB_PASSWORD}';
GRANT ALL PRIVILEGES ON ${DB_DATABASE_NAME} . * TO '${DB_USERNAME}'@'%';
FLUSH PRIVILEGES;
EOF
```

5. Use the following command to delete the temporary firewall rule:

Azure CLI

```
az mysql flexible-server firewall-rule delete \
    --resource-group ${RG_NAME} \
    --name ${SERVER_NAME}  \
    --rule-name "AllowAllIPs" \
    --yes
```

You now have a MySQL database server running and ready to connect with your app.

Verify the functionality of the deployment

The steps in this section show you how to verify that the deployment completes successfully.

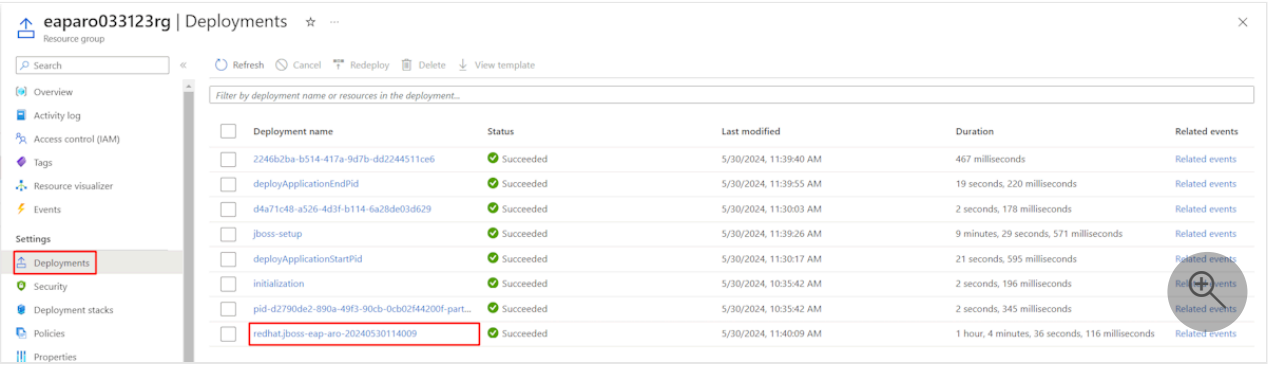
If you navigated away from the **Deployment is in progress** page, the following steps show you how to get back to that page. If you're still on the page that shows **Your deployment is complete**, you can skip to step 5.

1. In the corner of any Azure portal page, select the hamburger menu and then select **Resource groups**.

2. In the box with the text **Filter for any field**, enter the first few characters of the resource group you created previously. If you followed the recommended convention, enter your initials, then select the appropriate resource group.

3. In the navigation pane, in the **Settings** section, select **Deployments**. You see an ordered list of the deployments to this resource group, with the most recent one first.

4. Scroll to the oldest entry in this list. This entry corresponds to the deployment you started in the preceding section. Select the oldest deployment, as shown in the following screenshot.



5. In the navigation pane, select **Outputs**. This list shows the output values from the deployment, which includes some useful information.

6. Open the shell, paste the value from the **cmdToGetKubeadminCredentials** field, and execute it. You see the admin account and credential for signing in to the OpenShift cluster console portal. The following example shows an admin account:

Azure CLI

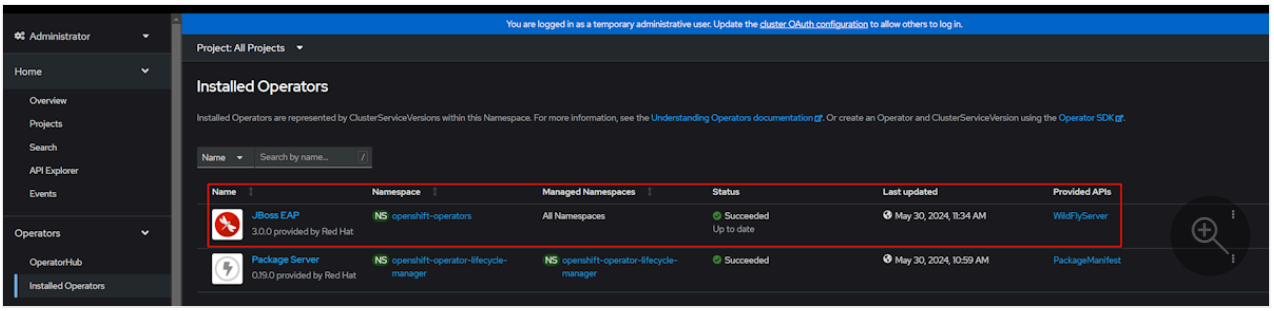
```
az aro list-credentials --resource-group eaparo033123rg --name clusterf9e8b9
```

This command produces output similar to the following example:

Output

```
{
  "kubeadminPassword": "xxxxx-xxxxx-xxxxx-xxxxx",
  "kubeadminUsername": "kubeadmin"
}
```

7. Paste the value from the **consoleUrl** field into an Internet-connected web browser, and then press `Enter`. Fill in the admin user name and password, then select **Log in**. In the admin console of Azure Red Hat OpenShift, select **Operators** > **Installed Operators**, where you can find that the **JBoss EAP** operator is successfully installed, as shown in the following screenshot.



Next, use the following steps to connect to the OpenShift cluster using the OpenShift CLI:

1. In the shell, use the following commands to download the latest OpenShift 4 CLI for GNU/Linux. If running on an OS other than GNU/Linux, download the appropriate binary for that OS.

Bash

```
cd ~
wget https://mirror.openshift.com/pub/openshift-
v4/clients/ocp/latest/openshift-client-linux.tar.gz

mkdir openshift
tar -zxvf openshift-client-linux.tar.gz -C openshift
echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

2. Paste the value from the **cmdToLoginWithKubeadmin** field into the shell, and execute it. You should see the `login successful` message and the project you're using. The following content is an example of the command to connect to the OpenShift cluster using the OpenShift CLI.

Azure CLI

```
oc login \
    $(az aro show \
        --resource-group ${RG_NAME} \
        --name aro-cluster \
        --query apiserverProfile.url \
        --output tsv) \
    -u $(az aro list-credentials \
        --resource-group ${RG_NAME} \
        --name aro-cluster \
        --query kubeadminUsername \
        --output tsv) \
    -p $(az aro list-credentials \
        --resource-group ${RG_NAME} \
        --name aro-cluster \
        --query kubeadminPassword \
        --output tsv)
```

This command produces output similar to the following example:

Output

```
Login successful.

You have access to 68 projects, the list has been suppressed. You can list all
projects with 'oc projects'

Using project "default".
```

Deploy a JBoss EAP app to the OpenShift cluster

The steps in this section show you how to deploy an app on the cluster.

Build the image to deploy

Use the following steps to deploy the app to the cluster. The app is hosted in the GitHub repo rhel-jboss-templates/eap-coffee-app.

1. Install Helm by using the following commands:

Bash

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
# Add the JBoss EAP Helm chart repository
helm repo add jboss-eap https://jbossas.github.io/eap-charts/
```

2. Prepare the environment variables for the deployment by using the following commands:

Bash

```
export SRC_REPO_URL="https://github.com/azure-javaee/rhel-jboss-templates.git"
export SRC_REPO_REF="20250609"
export SRC_REPO_DIR="eap-coffee-app"
```

3. Create a new project in the OpenShift cluster by using the following command:

Bash

```
oc new-project ${PROJECT_NAME}
```

4. Create a pull secret for the Red Hat Container Registry by using the following command:

Bash

```
oc create secret docker-registry ${CON_REG_SECRET_NAME} \
     --docker-server=registry.redhat.io \
     --docker-username=${CON_REG_ACC_USER_NAME} \
     --docker-password=${CON_REG_ACC_PWD}
```

5. Build the image by using the following commands:

Bash

```
helmDeployment=helm.yaml
cat <<EOF >${helmDeployment}
build:
   uri: ${SRC_REPO_URL}
   ref: "${SRC_REPO_REF}"
   contextDir: ${SRC_REPO_DIR}
   pullSecret: ${CON_REG_SECRET_NAME}
deploy:
   enabled: false
EOF

helm install ${APPLICATION_NAME} -f helm.yaml jboss-eap/eap8 --namespace
${PROJECT_NAME}
```

Create a secret for the database password

Next, use the following steps to create a secret:

1. Use the following command to create a secret for holding the password of the database:

   Bash

   ```
   oc create secret generic db-secret --from-literal=password=${DB_PASSWORD}
   ```

2. Use the following commands to deploy and run three replicas of the containerized app in the cluster:

   Bash

   ```
   cat <<EOF | oc apply -f -
   apiVersion: wildfly.org/v1alpha1
   kind: WildFlyServer
   metadata:
     name: ${APPLICATION_NAME}
   spec:
     applicationImage: ${APPLICATION_NAME}:latest
     replicas: ${APP_REPLICAS}
     env:
       - name: DB_SERVICE_PREFIX_MAPPING
         value: TEST-MYSQL=DS1
       - name: TEST_MYSQL_SERVICE_HOST
         value: ${DB_HOST}
       - name: TEST_MYSQL_SERVICE_PORT
         value: '3306'
       - name: DS1_JNDI
         value: java:jboss/datasources/JavaEECafeDB
       - name: DS1_URL
         value: jdbc:mysql://${DB_HOST}:3306/${DB_DATABASE_NAME}
       - name: DS1_DRIVER
         value: mysql
       - name: DS1_DATABASE
         value: ${DB_DATABASE_NAME}
       - name: DS1_USERNAME
         value: ${DB_USERNAME}
       - name: DS1_PASSWORD
         valueFrom:
           secretKeyRef:
             name: db-secret
             key: password
     secrets:
       - db-secret
   EOF
   ```

If the command completed successfully, you should see `wildflyserver.wildfly.org/javaee-cafe created`. If you don't see this output, troubleshoot and resolve the problem before proceeding.

3. Run `oc get pod -w | grep 1/1` to monitor whether all pods of the app are running. When you see output similar to the following example, press `Ctrl` + `C` to stop the monitoring:

Output

```
javaee-cafe-2          1/1     Running             0          31s
javaee-cafe-1          1/1     Running             0          30s
javaee-cafe-0          1/1     Running             0          30s
```

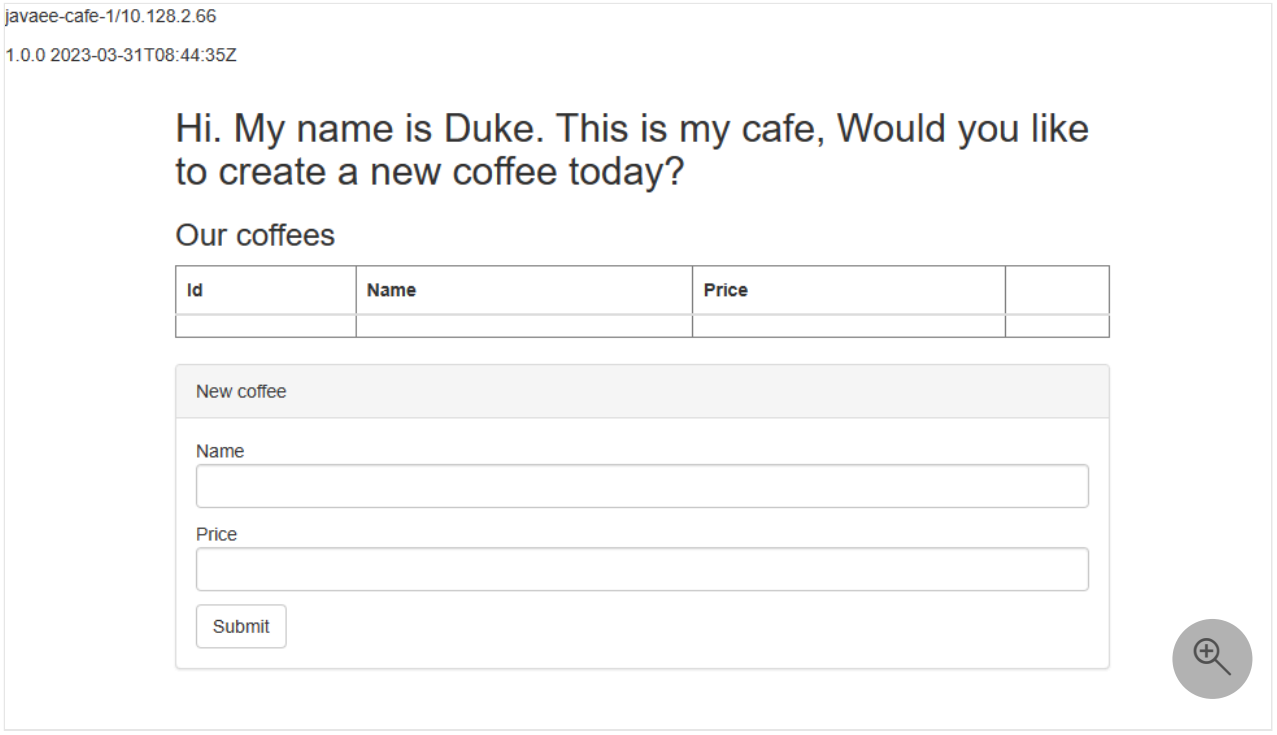
It might take a few minutes to reach the proper state. You might even see `STATUS` column values including `ErrImagePull` and `ImagePullBackOff` before `Running` is shown.

4. Run the following command to return the URL of the application. You can use this URL to access the deployed sample app. Copy the output to the clipboard.

Bash

```
echo http://$(oc get route ${APPLICATION_NAME}-route -
o=jsonpath='{.spec.host}')/javaee-cafe
```

5. Paste the output into an Internet-connected web browser, and then press `Enter`. You should see the UI of **Java EE Cafe** app similar to the following screenshot:



6. Add and delete some rows to verify the database connectivity is correctly functioning.

Clean up resources

If you're not going to continue to use the OpenShift cluster, navigate back to your working resource group. At the top of the page, under the text **Resource group**, select the resource group. Then, select **Delete resource group**.

Next step

For more information about deploying JBoss EAP on Azure, see Red Hat JBoss EAP on Azure.

Last updated on 03/26/2026

Confidential containers with Azure Red Hat OpenShift

Summarize this article for me

Confidential containers, a feature of Red Hat OpenShift sandboxed containers, offer a robust solution to protect sensitive data within cloud environments. When you use hardware-based trusted execution environments (TEEs), confidential containers provide a secure enclave within the host system, isolating applications and their data from potential threats. This isolation ensures that even if the host system is compromised, the confidential data remains protected.

This article describes the benefits of using confidential containers to safeguard sensitive data and explains how confidential containers function within Azure Red Hat OpenShift.

Benefits of Using confidential containers

Confidential containers offer several key benefits:

- Enhanced Data Security: By isolating applications and their data within a secure enclave, confidential containers protect sensitive information from unauthorized access, even if the host system is compromised.
- Regulatory Compliance: Industries such as healthcare, finance, and government are subject to stringent data privacy regulations. Confidential containers can help organizations meet these compliance requirements by providing a robust mechanism for protecting sensitive data.
- Improved Trust and Confidence: confidential containers can foster trust between cloud service providers and their customers by demonstrating commitment to data security and privacy.
- Reduced Risk of Data Breaches: The use of confidential containers can significantly reduce the risk of data breaches, which can have devastating consequences for organizations.
- Increased Efficiency: confidential containers can streamline the development and deployment of applications by providing a secure and efficient environment for running sensitive workloads.

Typical use cases

The following table describes the most common use cases for deploying confidential containers.

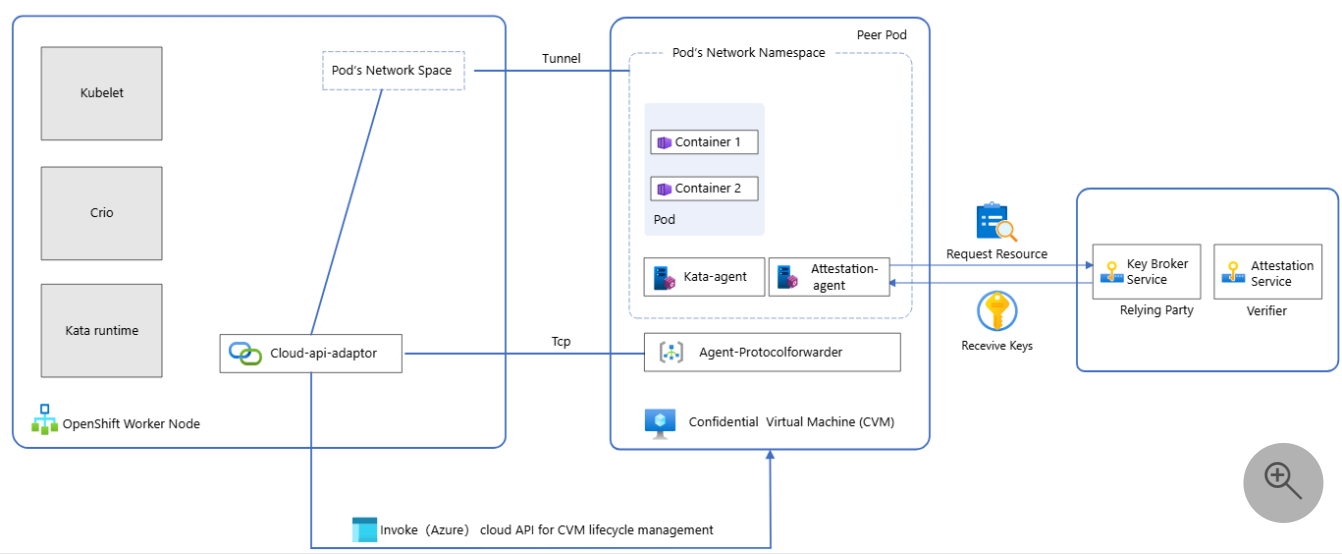
Expand table

Use case	Industry	Example
Regulator compliance Meeting strict data protection and privacy regulations.	Government, Finance, Healthcare	A healthcare provider using Confidential containers to process and store patient data in compliance with HIPAA regulations.
Multi-tenancy environments Hosting multiple clients' applications and data with strong isolation.	SaaS providers, Cloud service providers	A cloud service provider offering isolated environments for different clients within the same infrastructure.
Confidential Analytics	Any industry using sensitive data processing.	Safely process proprietary or regulated data where confidentiality is essential.

How confidential containers work

Confidential containers are a feature of Red Hat OpenShift sandboxed containers, which provide an isolated environment for running containerized applications. The core of confidential containers is the Confidential Virtual Machine (CVM). This specialized virtual machine, operating within a Trusted Execution Environment (TEE), establishes a secure enclave for applications and their associated data. TEEs, hardware-based isolated environments fortified with enhanced security features, ensure that even if the host system is compromised, the data residing within the CVM remains protected.

Azure Red Hat OpenShift serves as the orchestrator, overseeing the sandboxing of workloads (pods) through the utilization of virtual machines. With CVMs, Azure Red Hat OpenShift empowers Confidential Container capabilities for your workloads. After a confidential containers workload is created, Azure Red Hat OpenShift deploys it within a CVM executing within the TEE, providing a secure and isolated environment for your sensitive data.



The diagram shows the three main steps for using confidential containers on a cluster:

1. The OpenShift sandboxed containers Operator is deployed on the cluster.
2. Kata Runtime container on a worker node uses the cloud-api-adapter to create a peer pod on a confidential virtual machine.
3. The remote attestation agent on the peer pod initiates the attestation of the container image before the kata-agent deploys it, ensuring the integrity of the image.

> ⓘ **Note**
>
> The confidential containers feature is supported on Azure Red Hat OpenShift clusters with version 4.15 or higher.

Attestation

Attestation constitutes a fundamental component of confidential containers, particularly within the context of zero-trust security. Before you deploy a workload as a confidential containers workload, it's imperative to verify the trustworthiness of the TEE where the workload is executed. Attestation ensures that the TEE is indeed secure and possesses the capability to safeguard your confidential data.

The Red Hat build of Trustee project

The Red Hat build of Trustee project provides the attestation capabilities essential for confidential containers. It executes attestation operations and delivers secrets to the TEE following successful verification. Key components of the Red Hat build of Trustee include the following items:

- Guest agents: These components operate within the CVM, including the Attestation Agent (AA) responsible for transmitting evidence to substantiate the environment's trustworthiness.
- Key Broker Service (KBS): This service functions as an entrypoint for remote attestation, forwarding evidence to the Attestation Service (AS) for verification.
- Attestation Service (AS): This service validates the TEE evidence.

Red Hat build of Trustee Operator

The Red Hat build of Trustee Operator, an integral component of the Azure Red Hat OpenShift confidential containers solution, facilitates the deployment and management of Red Hat build

of Trustee services within a cluster. It streamlines the configuration of Red Hat build of Trustee services and the management of secrets for confidential containers workloads.

A Unified perspective

A typical confidential containers deployment involves Azure Red Hat OpenShift working with the Red Hat build of Trustee Operator deployed in a separate, trusted environment. The workload is executed within a CVM operating inside a TEE, benefiting from the encrypted memory and integrity guarantees provided by the TEE. Red Hat build of Trustee agents that reside within the CVM perform attestation and acquire requisite secrets, safeguarding the security and confidentiality of your data.

Next steps

To deploy confidential containers on your Azure Red Hat OpenShift cluster, see the following articles:

- Deploy Red Hat build of Trustee .
- Deploy confidential containers .

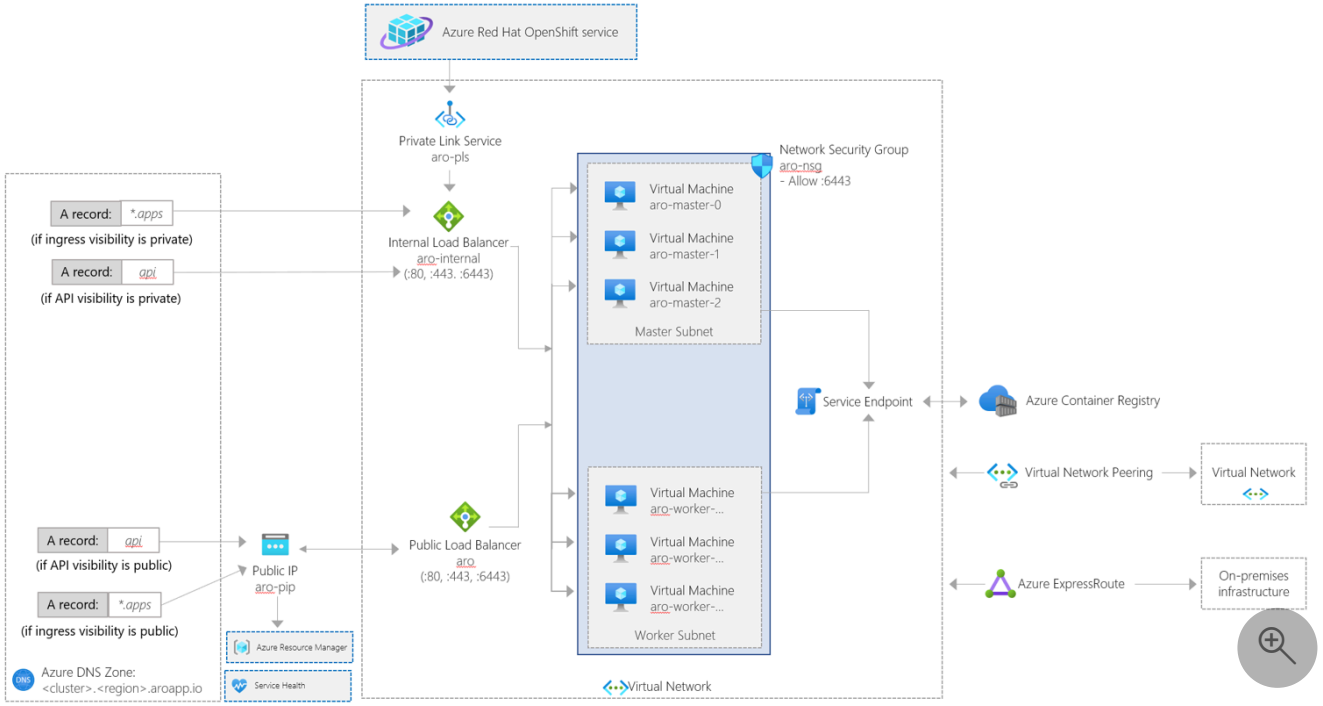
Last updated on 02/11/2026

Network concepts for Azure Red Hat OpenShift

06/30/2025

This guide covers an overview of Azure Red Hat OpenShift (ARO) networking on OpenShift 4 clusters, along with a diagram and a list of important endpoints. For more information on core OpenShift networking concepts, see the Azure Red Hat OpenShift 4 networking documentation.

Azure Red Hat OpenShift network architecture



When you deploy Azure Red Hat OpenShift on OpenShift 4, your entire cluster is contained within a virtual network. Within this virtual network, your control plane nodes and worker nodes each live in their own subnet. Each subnet uses an internal load balancer and a public load balancer.

> **Note**
>
> For information on the latest changes introduced to ARO, check out What's new with Azure Red Hat OpenShift.

Networking components

The following list covers important networking components in an Azure Red Hat OpenShift cluster.

- **aro-pls**
  - This Azure Private Link endpoint is used by Microsoft and Red Hat site reliability engineers to manage the cluster.

- **aro-internal**
  - This endpoint balances traffic to the API server and internal service traffic. Control plane nodes and worker nodes are in the backend pool.
  - This load balancer isn't created by default. It's created after you create a service of type `LoadBalancer` with the correct annotations. For example: `service.beta.kubernetes.io/azure-load-balancer-internal: "true"`.

- **aro**
  - This endpoint is used for any public traffic. When you create an application and a route, this endpoint is the path for ingress traffic.
  - This endpoint also routes and balances traffic to the API server (if the API is public). This endpoint assigns a public outgoing IP so control planes can access Azure Resource Manager and report back on cluster health.
  - This load balancer also covers egress Internet connectivity from any pod running in the worker nodes through Azure Load Balancer outbound rules.
    - Currently outbound rules aren't configurable. They allocate 1,024 TCP ports to each node.
    - DisableOutboundSnat isn't configured in the LB rules, so pods could get as egress IP any public IP configured in this ALB.
    - As a consequence of the two previous points, the only way of adding ephemeral SNAT ports is by adding public LoadBalancer-type services to ARO.

- **aro-nsg**
  - When you expose a service, the API creates a rule in this network security group so traffic flows through and reaches the control plane and nodes through port 6443.
  - By default this network security group allows all outbound traffic. Currently, outbound traffic can only be restricted to the Azure Red Hat OpenShift control plane.

- **Azure Container Registry**
  - The container registry is provided and used by Microsoft internally. It's read-only and not intended for use by Azure Red Hat OpenShift users.
    - This registry provides host platform images and cluster components. For example, monitoring or logging containers.
    - Connections to this registry occur over the service endpoint (internal connectivity between Azure services).

    - By default, this internal registry isn't available outside of the cluster.

- **Private Link**
    - A Private Link allows network connectivity from the management plane into a cluster. This is used by Microsoft and Red Hat site reliability engineers to help manage your cluster.

Networking policies

- **Ingress**: The ingress networking policy is supported as a part of OpenShift SDN. This network policy is enabled by default, and the enforcement is carried out by users. While the ingress network policy is V1 NetworkPolicy compliant, the Egress and IPBlock Types aren't supported.

- **Egress**: The egress network policies are supported by using the egress firewall feature in OpenShift. There's only one egress policy per namespace/project. Egress policies aren't supported on the "default" namespace and are evaluated in order (first to last).

Networking basics in OpenShift

OpenShift Software Defined Networking (SDN) is used to configure an overlay network using Open vSwitch (OVS), an OpenFlow implementation based on Container Network Interface (CNI) specification. The SDN supports different plugins. Network Policy is the plugin used in Azure Red Hat on OpenShift 4. The SDN manages all network communication, so no extra routes are needed on your virtual networks to achieve pod to pod communication.

Networking for Azure Red Hat OpenShift

The following networking features are specific to Azure Red Hat OpenShift:

- Users can create their Azure Red Hat OpenShift cluster in an existing virtual network or create a new virtual network when creating their cluster.
- Pod and Service Network CIDRs are configurable.
- Nodes and control planes are in different subnets.
- Nodes and control plane virtual network subnets should be minimum /27.
- The default Pod CIDR is 10.128.0.0/14.
- The default Service CIDR is 172.30.0.0/16.
- Pod and Service Network CIDRs shouldn't overlap with other address ranges in use on your network. They must not be within the virtual network IP address range of your cluster.

- Pod CIDRs should be minimum /18 in size. (The pod network is nonroutable IPs, and is only used inside the OpenShift SDN.)
- Each node is allocated /23 subnet (512 IPs) for its pods. This value can't be changed.
- You can't attach a pod to multiple networks.
- For private ARO clusters using OVN-Kubernetes network plugin, it's possible to configure egress IPs. For information, see configuring an egress IP address.

Network settings

The following network settings are available for Azure Red Hat OpenShift 4 clusters:

- **API Visibility** - Set the API visibility when running the az aro create command.
  - *Public* - API Server is accessible by external networks.
  - *Private* - API Server assigned a private IP from the control plane subnet, only accessible using connected networks (peered virtual networks and other subnets in the cluster).
- **Ingress Visibility** - Set the API visibility when running the az aro create command.
  - *Public* routes default to a public Standard Load Balancer. (The default can be changed.)
  - *Private* routes default to an internal load balancer. (The default can be changed.)

Network security groups

Network security groups are created in the node's resource group, which is locked to users. The network security groups are assigned directly to the subnets, not on the node's NICs. The network security groups are immutable. Users don't have the permissions to change them.

With a publicly visible API server, you can't create network security groups and assign them to the NICs.

Domain forwarding

Azure Red Hat OpenShift uses CoreDNS. Domain forwarding can be configured. You can't bring your own DNS to your virtual networks. For more information, see the documentation on using DNS forwarding.

Next steps

For more information on outbound traffic and what Azure Red Hat OpenShift supports for egress, see the support policies documentation.

Overview of Azure Red Hat OpenShift egress lockdown

Article • 02/25/2025

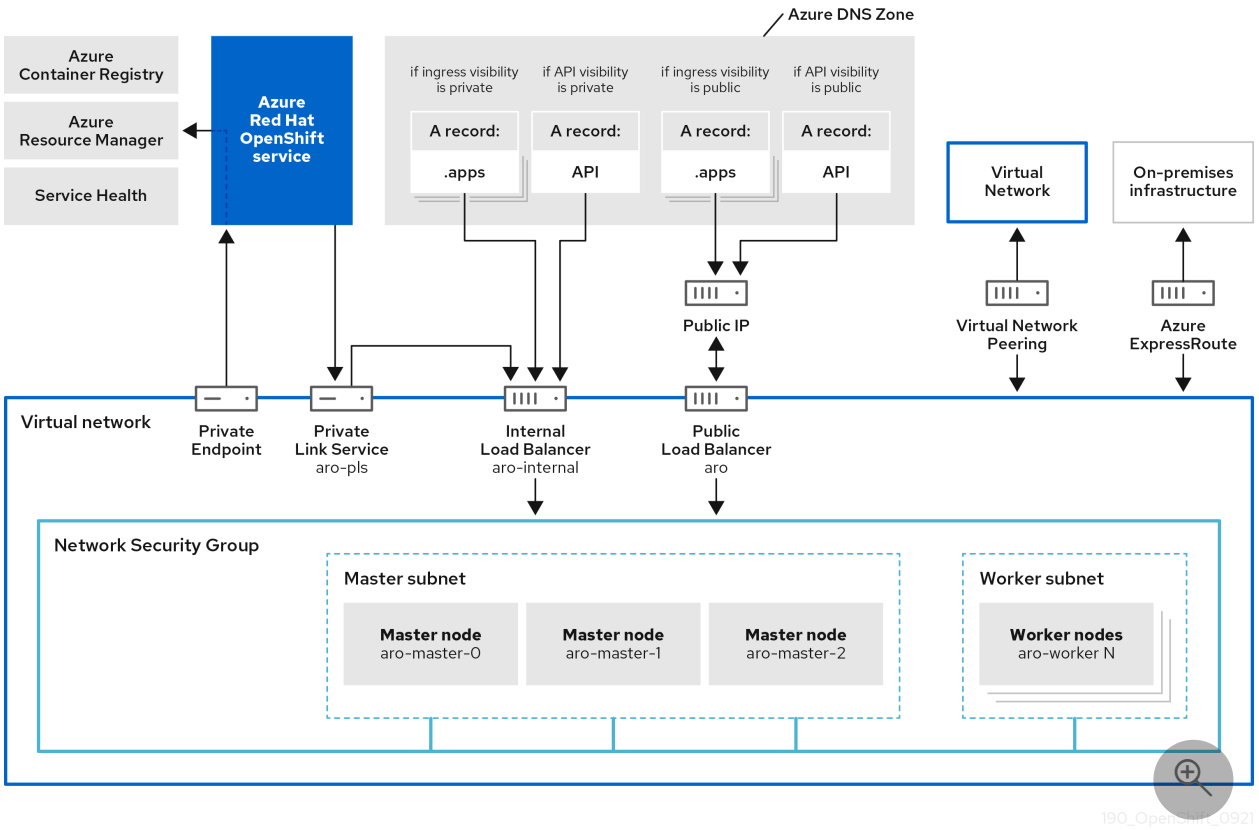
Egress lockdown provides access to the URLs and endpoints an Azure Red Hat OpenShift cluster needs to function effectively.

Egress lockdown ensures that you have access to URLs, such as management.azure.com, so you can create another worker node backed by Azure VMs. Egress lockdown ensures access even if the outbound (egress) traffic is restricted by a firewall appliance or other means.

Egress lockdown takes a collection of domains required for an Azure Red Hat OpenShift cluster to function and proxies calls to these domains through the Azure Red Hat OpenShift service. The domains, which are region-specific, can't be configured by customers.

Egress lockdown doesn't rely on customer internet access for Azure Red Hat OpenShift services to work. In order for clusters to reach any Azure Red Hat OpenShift service, cluster traffic exits through an Azure private endpoint created within the cluster resource group where all of the Azure Red Hat OpenShift resources are available.

The following image displays the architecture changes that encompass egress lockdown.



A well-known subset of domains (that the Azure Red Hat OpenShift clusters need to function) validates the destination of the cluster traffic. Finally, the traffic passes through the Azure Red Hat OpenShift service to connect to these URLs and endpoints.

Enable egress lockdown

In order to function, egress lockdown relies on the Server Name Indication (SNI) extension to the Transport Layer Security (TLS). All customer workloads that communicate with the well-known subset of domains must have SNI enabled.

Egress lockdown is enabled by default for new cluster creation. However, to enable egress lockdown on existing clusters, you must have SNI enabled on the customer workloads. To enable egress lockdown on your existing clusters, submit a support case to either Microsoft Support or Red Hat Support.

Verify egress lockdown is enabled on a cluster

To verify whether egress lockdown is enabled on a cluster, sign in to your Azure cluster and run the following command:

Azure CLI

```
$ oc get cluster.aro.openshift.io cluster -o go-template='{{ if
.spec.gatewayDomains }}{{ "Egress Lockdown Feature Enabled" }}{{ else }}{{
"Egress Lockdown Feature Disabled" }}{{ end }}{{ "\n" }}'
```

Depending on whether egress lockdown is enabled or disabled, you'll see one of the following messages:

- `Egress Lockdown Feature Enabled`
- `Egress Lockdown Feature Disabled`

Relation to storage lockdown

Storage lockdown is another feature of Azure Red Hat OpenShift that enhances cluster security. Storage Accounts created with the cluster are configured to restrict any public access. Exceptions are added for the Azure Red Hat OpenShift Resource Provisioner subnets as well as the subnet of egress lockdown gateway. Cluster components that utilize this storage, for example, OpenShift Image Registry, rely on egress lockdown functionality instead of accessing the storage accounts directly.

Next steps

For more information on controlling egress traffic on your Azure Red Hat OpenShift cluster, see Control egress traffic for your Azure Red Hat OpenShift (ARO) cluster (preview).

Feedback

OVN-Kubernetes network provider for Azure Red Hat OpenShift clusters

Article • 02/25/2025

The OpenShift Container Platform cluster uses a virtualized network for pod and service networks. The OVN-Kubernetes Container Network Interface (CNI) plug-in is a network provider for the default cluster network. OVN-Kubernetes, which is based on the Open Virtual Network (OVN), provides an overlay-based networking implementation.

A cluster that uses the OVN-Kubernetes network provider also runs Open vSwitch (OVS) on each node. OVN configures OVS on each node to implement the declared network configuration.

OVN-Kubernetes features

The OVN-Kubernetes CNI cluster network provider offers the following features:

- Uses OVN to manage network traffic flows. OVN is a community developed, vendor-agnostic network virtualization solution.
- Implements Kubernetes network policy support, including ingress and egress rules.
- Uses the Generic Network Virtualization Encapsulation (Geneve) protocol rather than the Virtual Extensible LAN (VXLAN) protocol to create an overlay network between nodes.

> **Important**
>
> Since ARO 4.11, OVN-Kubernetes has been the CNI for all new Azure Red Hat OpenShift clusters. For clusters created before then, migrating from the previous SDN standard to OVN is recommended. SDN has been deprecated since version 4.14 and will no longer be supported after version 4.16. You must migrate to OVN-Kubernetes prior to updating to 4.17.

For more information about OVN-Kubernetes CNI network provider, see About the OVN-Kubernetes default Container Network Interface (CNI) network provider.

For information about migrating from SDN to OVN-Kubernetes, see Migrate from OpenShift SDN to OVN-Kubernetes.

Recommended content

Tutorial: Create an Azure Red Hat OpenShift 4 cluster

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

az aro

Manage Azure Red Hat OpenShift clusters.

Commands

Expand table

Name	Description	Type	Status
az aro create	Create a cluster.	Core	GA
az aro delete	Delete a cluster.	Core	GA
az aro get-admin-kubeconfig	List admin kubeconfig of a cluster.	Core	GA
az aro get-versions	List versions available for installation.	Core	GA
az aro list	List clusters.	Core	GA
az aro list-credentials	List credentials of a cluster.	Core	GA
az aro show	Get the details of a cluster.	Core	GA
az aro update	Update a cluster.	Core	GA
az aro validate	Validate permissions required to create a cluster.	Core	GA
az aro wait	Wait for a cluster to reach a desired state.	Core	GA

az aro create

Edit

Create a cluster.

Azure CLI

```
az aro create --master-subnet
              --name
              --resource-group
              --worker-subnet
              [--acquire-policy-token]
              [--apiserver-visibility {Private, Public}]
              [--assign-cluster-identity --mi-user-assigned]
```

```
            [--assign-platform-wi --assign-platform-workload-identity]
            [--change-reference]
            [--client-id]
            [--client-secret]
            [--cluster-resource-group]
            [--disk-encryption-set]
            [--domain]
            [--enable-managed-identity --enable-mi {false, true}]
            [--enable-preconfigured-nsg {false, true}]
            [--fips --fips-validated-modules {false, true}]
            [--ingress-visibility {Private, Public}]
            [--lb-ip-count --load-balancer-managed-outbound-ip-count]
            [--location]
            [--master-enc-host --master-encryption-at-host {false, true}]
            [--master-vm-size]
            [--no-wait]
            [--outbound-type]
            [--pod-cidr]
            [--pull-secret]
            [--service-cidr]
            [--tags]
            [--version]
            [--vnet]
            [--vnet-resource-group]
            [--worker-count]
            [--worker-enc-host --worker-encryption-at-host {false, true}]
            [--worker-vm-disk-size-gb]
            [--worker-vm-size]
```

Examples

Create a cluster.

Azure CLI

```
az aro create --resource-group MyResourceGroup --name MyCluster --vnet MyVnet --
master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet
```

Create a cluster with a supported OpenShift version.

Azure CLI

```
az aro create --resource-group MyResourceGroup --name MyCluster --vnet MyVnet --
master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet --version X.Y.Z
```

Create a cluster with 5 compute nodes and Red Hat pull secret.

Azure CLI

```
az aro create --resource-group MyResourceGroup --name MyCluster --vnet MyVnet --
master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet --worker-count 5 --
pull-secret pullsecret.txt
```

Create a private cluster.

Azure CLI

```
az aro create --resource-group MyResourceGroup --name MyCluster --vnet MyVnet --
master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet --apiserver-visibility
Private --ingress-visibility Private
```

Required Parameters

`--master-subnet`

Name or ID of master vnet subnet. If name is supplied, `--vnet` must be supplied.

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--worker-subnet`

Name or ID of worker vnet subnet. If name is supplied, `--vnet` must be supplied.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--apiserver-visibility`

API server visibility. [Default: Public].

Expand table

Property	Value
Accepted values:	Private, Public

`--assign-cluster-identity --mi-user-assigned`

Set the user managed identity on the cluster. Value must be an identity name or resource ID.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--assign-platform-wi --assign-platform-workload-identity`

Assign a platform workload identity used within the cluster. Requires two values: an operator name and either the name or resource ID of the Azure identity to use for it.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--client-id`

Client ID of cluster service principal.

`--client-secret`

Client secret of cluster service principal.

`--cluster-resource-group`

Resource group of cluster.

`--disk-encryption-set`

ResourceID of the DiskEncryptionSet to be used for master and worker VMs.

`--domain`

Domain of cluster.

`--enable-managed-identity --enable-mi`

Enable managed identity for this cluster.

Expand table

Property	Value
Parameter group:	Identity Arguments
Default value:	False
Accepted values:	false, true

`--enable-preconfigured-nsg`

Use Preconfigured NSGs. Allowed values: false, true. [Default: false].

Expand table

Property	Value
Accepted values:	false, true

`--fips --fips-validated-modules`

Use FIPS validated cryptography modules. [Default: false].

Expand table

Property	Value
Accepted values:	false, true

`--ingress-visibility`

Ingress visibility. [Default: Public].

Expand table

Property	Value
Accepted values:	Private, Public

`--lb-ip-count --load-balancer-managed-outbound-ip-count`

The desired number of IPv4 outbound IPs created and managed by Azure for the cluster public load balancer.

`--location -l`

Location. Values from: `az account list-locations`. You can configure the default location using `az configure --defaults location=<location>`.

`--master-enc-host --master-encryption-at-host`

Encryption at host flag for master VMs. [Default: false].

Expand table

Property	Value
Default value:	False

Property	Value
Accepted values:	false, true

`--master-vm-size`

Size of master VMs. [Default: Standard_D8s_v5].

`--no-wait`

Do not wait for the long-running operation to finish.

Expand table

Property	Value
Default value:	False

`--outbound-type`

Outbound type of cluster. Must be "Loadbalancer" or "UserDefinedRouting". [Default: Loadbalancer].

`--pod-cidr`

CIDR of pod network. Must be a minimum of /18 or larger. [Default: 10.128.0.0/14].

`--pull-secret`

Pull secret of cluster.

`--service-cidr`

CIDR of service network. Must be a minimum of /18 or larger. [Default: 172.30.0.0/16].

`--tags`

Space-separated tags: key[=value] [key[=value] ...]. Use "" to clear existing tags.

`--version`

OpenShift version to use for cluster creation.

`--vnet`

Name or ID of vnet. If name is supplied, `--vnet-resource-group` must be supplied.

`--vnet-resource-group`

Name of vnet resource group.

`--worker-count`

Count of worker VMs. [Default: 3].

`--worker-enc-host --worker-encryption-at-host`

Encryption at host flag for worker VMs. [Default: false].

Expand table

Property	Value
Default value:	False
Accepted values:	false, true

`--worker-vm-disk-size-gb`

Disk size in GB of worker VMs. [Default: 128].

`--worker-vm-size`

Size of worker VMs. [Default: Standard_D4s_v5].

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro delete

Edit

Delete a cluster.

Azure CLI

```
az aro delete --name
              --resource-group
              [--acquire-policy-token]
              [--change-reference]
              [--delete-identities {false, true}]
              [--no-wait]
              [--yes]
```

Examples

Delete a cluster.

Azure CLI

```
az aro delete --name MyCluster --resource-group MyResourceGroup
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--delete-identities`

Delete the cluster's associated managed identities together with the cluster.

Expand table

Property	Value
Parameter group:	Identity Arguments
Accepted values:	false, true

`--no-wait`

Do not wait for the long-running operation to finish.

Expand table

Property	Value
Default value:	False

`--yes -y`

Do not prompt for confirmation.

Expand table

Property	Value
Default value:	False

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro get-admin-kubeconfig

Edit

List admin kubeconfig of a cluster.

Azure CLI

```
az aro get-admin-kubeconfig --name
                            --resource-group
                            [--acquire-policy-token]
                            [--change-reference]
                            [--file]
```

Examples

List admin kubeconfig of a cluster. The default is to save it in a file named "kubeconfig".

Azure CLI

```
az aro get-admin-kubeconfig --name MyCluster --resource-group MyResourceGroup
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--file -f`

Path to the file where kubeconfig should be saved. Default: kubeconfig in local directory.

Expand table

Property	Value
Default value:	kubeconfig

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json

Property	Value
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro get-versions

Edit

List versions available for installation.

Azure CLI

```
az aro get-versions --location
                    [--acquire-policy-token]
                    [--change-reference]
```

Examples

List install versions available for the East US region.

Azure CLI

```
az aro get-versions --location eastus
```

List install versions available for the East US region with table formatted output.

Azure CLI

```
az aro get-versions --location eastus -o table
```

Required Parameters

`--location -l`

Location. Values from: `az account list-locations`. You can configure the default location using `az configure --defaults location=<location>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro list

Edit

List clusters.

Azure CLI

```
az aro list [--resource-group]
```

Examples

List clusters.

Azure CLI

```
az aro list
```

List clusters with table view.

Azure CLI

```
az aro list -o table
```

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro list-credentials

Edit

List credentials of a cluster.

Azure CLI

```
az aro list-credentials --name
                        --resource-group
                        [--acquire-policy-token]
                        [--change-reference]
```

Examples

List credentials of a cluster.

Azure CLI

```
az aro list-credentials --name MyCluster --resource-group MyResourceGroup
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro show

Edit

Get the details of a cluster.

Azure CLI

```
az aro show --name
            --resource-group
```

Examples

Get the details of a cluster.

Azure CLI

```
az aro show --name MyCluster --resource-group MyResourceGroup
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro update

Edit

Update a cluster.

Azure CLI

```
az aro update --name
              --resource-group
              [--acquire-policy-token]
              [--assign-cluster-identity --mi-user-assigned]
              [--assign-platform-wi --assign-platform-workload-identity]
              [--change-reference]
              [--client-id]
              [--client-secret]
              [--lb-ip-count --load-balancer-managed-outbound-ip-count]
              [--no-wait]
              [--refresh-credentials {false, true}]
              [--upgradeable-to]
```

Examples

Update a cluster.

Azure CLI

```
az aro update --name MyCluster --resource-group MyResourceGroup
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--assign-cluster-identity --mi-user-assigned`

Set the user managed identity on the cluster. Value must be an identity name or resource ID.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--assign-platform-wi --assign-platform-workload-identity`

Assign a platform workload identity used within the cluster. Requires two values: an operator name and either the name or resource ID of the Azure identity to use for it.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--client-id`

Client ID of cluster service principal.

`--client-secret`

Client secret of cluster service principal.

`--lb-ip-count --load-balancer-managed-outbound-ip-count`

The desired number of IPv4 outbound IPs created and managed by Azure for the cluster public load balancer.

`--no-wait`

Do not wait for the long-running operation to finish.

Expand table

Property	Value
Default value:	False

`--refresh-credentials`

Refresh cluster application credentials.

Expand table

Property	Value
Default value:	False

Property	Value
Accepted values:	false, true

`--upgradeable-to`

OpenShift version to upgrade to.

Expand table

Property	Value
Parameter group:	Identity Arguments

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro validate

Edit

Validate permissions required to create a cluster.

Azure CLI

```
az aro validate --master-subnet
                --name
                --resource-group
                --worker-subnet
                [--acquire-policy-token]
                [--assign-cluster-identity --mi-user-assigned]
                [--assign-platform-wi --assign-platform-workload-identity]
                [--change-reference]
                [--client-id]
                [--client-secret]
                [--cluster-resource-group]
```

```
                [--disk-encryption-set]
                [--enable-managed-identity --enable-mi {false, true}]
                [--location]
                [--pod-cidr]
                [--service-cidr]
                [--version]
                [--vnet]
                [--vnet-resource-group]
```

Examples

Validate permissions.

Azure CLI

```
az aro validate --resource-group MyGroup --name MyName --vnet MyVnet --master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet
```

Validate permissions and OpenShift version

Azure CLI

```
az aro validate --resource-group MyGroup --name MyName --vnet MyVnet --master-subnet MyMasterSubnet --worker-subnet MyWorkerSubnet --version X.Y.Z
```

Required Parameters

`--master-subnet`

Name or ID of master vnet subnet. If name is supplied, `--vnet` must be supplied.

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

`--worker-subnet`

Name or ID of worker vnet subnet. If name is supplied, `--vnet` must be supplied.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--assign-cluster-identity --mi-user-assigned`

Set the user managed identity on the cluster. Value must be an identity name or resource ID.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--assign-platform-wi --assign-platform-workload-identity`

Assign a platform workload identity used within the cluster. Requires two values: an operator name and either the name or resource ID of the Azure identity to use for it.

Expand table

Property	Value
Parameter group:	Identity Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--client-id`

Client ID of cluster service principal.

`--client-secret`

Client secret of cluster service principal.

`--cluster-resource-group`

Resource group of cluster.

`--disk-encryption-set`

ResourceID of the DiskEncryptionSet to be used for master and worker VMs.

`--enable-managed-identity --enable-mi`

Enable managed identity for this cluster.

Expand table

Property	Value
Parameter group:	Identity Arguments
Default value:	False
Accepted values:	false, true

`--location -l`

Location. Values from: `az account list-locations`. You can configure the default location using `az configure --defaults location=<location>`.

`--pod-cidr`

CIDR of pod network. Must be a minimum of /18 or larger. [Default: 10.128.0.0/14].

`--service-cidr`

CIDR of service network. Must be a minimum of /18 or larger. [Default: 172.30.0.0/16].

`--version`

OpenShift version to use for cluster creation.

`--vnet`

Name or ID of vnet. If name is supplied, `--vnet-resource-group` must be supplied.

`--vnet-resource-group`

Name of vnet resource group.

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

az aro wait

Edit

Wait for a cluster to reach a desired state.

If an operation on a cluster was interrupted or was started with `--no-wait`, use this command to wait for it to complete.

Azure CLI

```
az aro wait --name
            --resource-group
            [--acquire-policy-token]
            [--change-reference]
```

```
            [--created]
            [--custom]
            [--deleted]
            [--exists]
            [--interval]
            [--timeout]
            [--updated]
```

Required Parameters

`--name -n`

Name of cluster.

`--resource-group -g`

Name of resource group. You can configure the default group using `az configure --defaults group=<name>`.

Optional Parameters

The following parameters are optional, but depending on the context, one or more might become required for the command to execute successfully.

`--acquire-policy-token`

Acquiring an Azure Policy token automatically for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--change-reference`

The related change reference ID for this resource operation.

Expand table

Property	Value
Parameter group:	Global Policy Arguments

`--created`

Wait until created with 'provisioningState' at 'Succeeded'.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	False

`--custom`

Wait until the condition satisfies a custom JMESPath query. E.g. provisioningState!='InProgress', instanceView.statuses[?code=='PowerState/running'].

Expand table

Property	Value
Parameter group:	Wait Condition Arguments

`--deleted`

Wait until deleted.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	False

`--exists`

Wait until the resource exists.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	False

`--interval`

Polling interval in seconds.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	30

`--timeout`

Maximum wait in seconds.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	3600

`--updated`

Wait until updated with provisioningState at 'Succeeded'.

Expand table

Property	Value
Parameter group:	Wait Condition Arguments
Default value:	False

Global Parameters

`--debug`

Increase logging verbosity to show all debug logs.

Expand table

Property	Value
Default value:	False

`--help -h`

Show this help message and exit.

`--only-show-errors`

Only show errors, suppressing warnings.

Expand table

Property	Value
Default value:	False

`--output -o`

Output format.

Expand table

Property	Value
Default value:	json
Accepted values:	json, jsonc, none, table, tsv, yaml, yamlc

`--query`

JMESPath query string. See http://jmespath.org/ for more information and examples.

`--subscription`

Name or ID of subscription. You can configure the default subscription using `az account set -s NAME_OR_ID`.

`--verbose`

Increase logging verbosity. Use --debug for full debug logs.

Expand table

Property	Value
Default value:	False

Azure Command-Line Interface (CLI) documentation

Official product documentation for Azure command-line interface (Azure CLI). Azure CLI is a cross-platform command-line tool for managing Azure resources with interactive commands or scripts.

About Azure CLI

OVERVIEW

Get started

What is Azure CLI?

Support lifecycle

TRAINING

Create Azure resources

Control Azure services

Manage virtual machines

Installation

DOWNLOAD

Overview

Install - Windows

Install - Linux

Install - macOS

Run in Azure Cloud Shell

Run in a Docker container

What's new

WHAT'S NEW

Overview

Release notes

Upcoming breaking changes

Impact of MFA in automation

Azure CLI reference

REFERENCE

Command reference

Identity and authentication

HOW-TO GUIDE

Authentication methods

Sign in with Web Account Manager (WAM)

Concepts

HOW-TO GUIDE

Manage subscriptions

Use the Azure REST API

Query command output

Change output formats

Script Azure resources at scale

TUTORIAL

Learn to use the Azure CLI

Work with service principals

Create virtual machines

Configuration

HOW-TO GUIDE

Configure settings

Protect sensitive information

Fundamentals

GET STARTED

Conceptual articles

Onboarding cheat sheet

Tips to use Azure CLI successfully

Samples

SAMPLE

Samples

Samples repo

Environment-specific guidance

CONCEPT

Work in interactive mode

How-to use Azure CLI in a Bash environment

Considerations in a PowerShell environment

Help & support

OVERVIEW

Report product issues

Troubleshoot

Get help from the community

Follow Azure CLI on X

Azure Tools Blog

Azure Red Hat OpenShift REST API reference

Article • 10/31/2023

Microsoft Azure Red Hat OpenShift enables you to deploy fully managed OpenShift clusters, allowing you to maintain regulatory compliance while you stay focused on application development.

Troubleshooting for Azure Red Hat OpenShift

08/07/2025

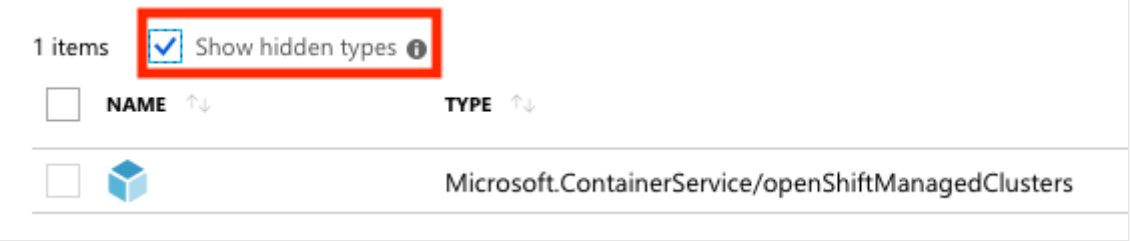
This article details some common issues encountered while creating or managing Microsoft Azure Red Hat OpenShift clusters.

Retrying the creation of a failed cluster

If you create an Azure Red Hat OpenShift cluster using the `az` CLI command and it fails, don't attempt to retry the deployment. Use `az aro delete` to delete the failed cluster, then create a new cluster.

Hidden Azure Red Hat OpenShift cluster resource group

Currently, the `RedHatOpenShift/OpenShiftClusters` resource that's automatically created by the Azure CLI (`az aro create` command) is hidden in the Azure portal. In the **Resource group** view, check **Show hidden types** to view the resource group.



Creating a cluster results in error that no registered resource provider found

If creating a cluster results in an error that `No registered resource provider found for location '<location>' and API version '2019-04-30' for type 'openShiftManagedClusters'. The supported api-versions are '2018-09-30-preview'.`, then you were part of the preview and now need to purchase Azure virtual machine reserved instances to use the generally available product. A reservation reduces your spend by pre-paying for fully managed Azure services. For more information about reservations and how they save you money, see What are Azure Reservations?

Exceeding Azure storage limits

If requests are being throttled due to Azure storage limits being exceeded, it might be due to one of the following reasons:

- There's a maximum of approximately 50 clusters per subscription ID + region. Create fewer than 50 clusters per subscription + region. For example: 25 clusters in subscription + eastus and 25 clusters in subscription + eastus2.
- Avoid creating multiple clusters within a single subscription + region at the same time. If you need to create multiple clusters in a short period of time, federate over multiple subscriptions or regions.

If the issue persists, create a support ticket for investigation.

Next steps

- Visit the OpenShift documentation
- Azure Support or Red Hat Support to open a support case.
- Find answers to frequently asked questions about Azure Red Hat OpenShift.

Support lifecycle for Azure Red Hat OpenShift 4

Red Hat releases minor versions of Red Hat OpenShift Container Platform (OCP) approximately every four months. These releases include new features and improvements. Patch releases are more frequent (typically weekly) and might include fixes for security vulnerabilities or bugs.

Azure Red Hat OpenShift is built from specific releases of OCP. This article covers the versions of OCP that are supported for Azure Red Hat OpenShift and details about updates, deprecations, and the support policy.

Red Hat OpenShift versions

Red Hat OpenShift Container Platform uses semantic versioning. Semantic versioning uses different levels of numbers to specify different versions. The following table illustrates the different parts of a semantic version number, in this case using the example version number `4.19.16`.

Expand table

Major version (x)	Minor version (y)	Patch version (z)
4	19	16

- **Major version**: No major version releases are planned at this time. Major versions involve significant changes to the core service such as large-scale additions of new features and functions, architectural changes, and removal of existing functions.
- **Minor version**: Released approximately every four months. Minor version updates can include feature additions, enhancements, deprecations, removals, bug fixes, security enhancements, and other improvements.
- **Patch version**: Typically released each week, or as needed. Patch version updates can include bug fixes, security enhancements, and other improvements.

You should aim to run the latest minor release of the major version you're running. For example, if your production cluster is on 4.18, and 4.19 is the latest generally available minor version for the 4 series, you should update to 4.19 as soon as you can.

Update channels

Update channels are the mechanism by which users state the OpenShift Container Platform minor version they intend to update their clusters to. Update channels are tied to a minor

version of Red Hat OpenShift Container Platform. The version number in the channel represents the target minor version that the cluster will eventually be updated to. An update channel doesn't recommend updates to a version above the selected channel's version. For instance, the OCP `stable-4.18` update channel doesn't include an update to a 4.19 release. Update channels only control release selection and don't modify the current version of the cluster. For more information, see Understanding update channels and releases.

> **Important**
>
> Azure Red Hat OpenShift provides support for `stable` and `eus` channels only. For example, `stable-4.19` or, `eus-4.18`.

You can use the `stable` or `eus` channel to update from a previous minor version of Azure Red Hat OpenShift. Clusters updated using `fast` or `candidate` channels could put your cluster in a Limited Support state.

Azure Red Hat OpenShift version support policy

Azure Red Hat OpenShift version availability

An Azure Red Hat OpenShift release is available through one of two mechanisms:

- When an update to a newer version is available for an existing cluster
- When a new version is available as an install target for a new cluster

Update availability

Azure Red Hat OpenShift supports generally available (GA) minor versions of Red Hat OpenShift Container Platform from when an update is available in the OpenShift `stable` channel. Update availability can be checked at the following page, Red Hat OpenShift Container Platform Update Graph.

Install availability

Installable versions can be validated by using the Azure Red Hat OpenShift release calendar or by running the following Azure CLI command:

Azure CLI

```
az aro get-versions --location [region]
```

Supported versions policy exceptions

The Azure Red Hat OpenShift SRE team reserves the right to add or remove new or existing versions, or delay upcoming minor release versions that were identified to have one or more critical production impacting bugs or security issues without advance notice.

Specific patch releases might be skipped, or rollout might be accelerated depending on the severity of the bug or security issue.

Mandatory updates

In extreme circumstances and based on the assessment of the Common Vulnerabilities and Exposures (CVE) criticality to the environment, you're notified that you have 72 hours to update your cluster to the latest, secure patch release. In the case that the update isn't done after 72 hours, a critical patch update might be applied to clusters automatically by Azure Red Hat OpenShift Site Reliability Engineers (SRE) which are then followed with a notification that informs you of the change. It's best practice to install patch (z-stream) updates as soon as they're available.

Version end-of-life

End-of-life means that a version is no longer supported in a `stable` channel for odd minor versions, nor in a `eus` channel for even minor versions. The end-of-life date for a version of Azure Red Hat OpenShift can be found in the Azure Red Hat OpenShift release calendar.

> **Note**
>
> If you're running an unsupported Red Hat OpenShift version, you might be asked to update when requesting support for the cluster. Clusters running unsupported Red Hat OpenShift releases aren't covered by the Azure Red Hat OpenShift service-level agreement (SLA).

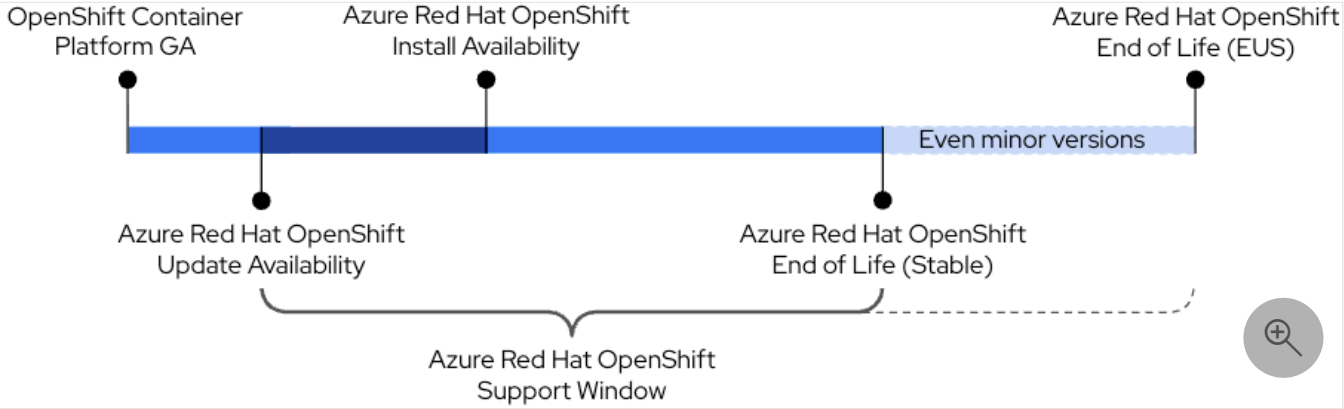
Azure Red Hat OpenShift release calendar

See the following guide for the past Red Hat OpenShift Container Platform (upstream) release history.

Expand table

Version	OCP GA Availability	Azure Red Hat Openshift Install Availability	Azure Red Hat Openshift End of Life (Stable channel)	Azure Red Hat OpenShift End of Life (EUS Term 1)
4.20	October 2025	Coming soon	April 21, 2027	October 21, 2027
4.19	June 2025	Coming soon	December 17, 2026	N/A
4.18	February 2025	November 6, 2025	August 25, 2026	February 25, 2027
4.17	October 2024	June 5, 2025	April 1, 2026	N/A
4.16	June 2024	March 10, 2025	December 27, 2025	June 27, 2026
4.15	February 2024	September 4, 2024	August 27, 2025	N/A

Review the following image to learn about the Azure Red Hat OpenShift support window.



- The support window for an OCP version begins with **Azure Red Hat Openshift Update Availability**.
- **Azure Red Hat Openshift Update Availability** is the date when the OCP version is available in a stable channel for an update from a previous version.
- **Azure Red Hat Openshift Install Availability** is the date when the version is available for a new cluster installation. For example when you create a new cluster with Azure portal or Azure CLI.
- **Azure Red Hat Openshift End of Life** is the date when a version is no longer supported in the `stable` channel for odd minor versions.
- **Azure Red Hat OpenShift End of Life (EUS)** is the date when a version is no longer supported in the `eus` channel for even minor versions.

For more information about stable update channels, see Understanding update channels and releases.

Extended Update Support Add-on Term 1

Extended Update Support Add-on (EUS) Term 1 is available on even-numbered minor versions starting with version 4.16, and is included with your Azure Red Hat OpenShift subscription. This provides the key benefit of extending the support lifecycle for an additional 6 month period.

To apply EUS Term 1 to your Azure Red Hat OpenShift cluster you must change your support channel to `eus-4.y`. For more information about update channels see Understanding update channels and releases.

> **Note**
>
> Obtaining updates and support during the EUS period requires you to change your update channel to `eus-4.y`

For more information about EUS Term 1 see Extended Update Support Add-On - Term 1.

Limited support status

When a cluster transitions to a limited support status, also called outside of support, Azure Red Hat OpenShift SREs no longer proactively monitor the cluster. And, the SLA is no longer applicable and credits requested against the SLA are denied, though it doesn't mean that you no longer have product support.

A cluster might transition to a Limited Support status for many reasons, including the following scenarios:

- If you don't update a cluster to a supported version before the end-of-life date.
  - There are no runtime or SLA guarantees for versions after their end-of-life date. To avoid this and continue receiving full support, update the cluster to a supported version before the end-of-life date. If you don't update the cluster before the end-of-life date, the cluster transitions to a Limited Support status until it's updated to a supported version.
  - Azure Red Hat OpenShift SREs provide commercially reasonable support to update from an unsupported version to a supported version. However, if a supported update path is no longer available, you might have to create a new cluster and migrate your workloads.
- If you remove or replace any native Azure Red Hat OpenShift components or any other component that's installed and managed by the service.
  - If admin permissions were used, Azure Red Hat OpenShift isn't responsible for any of your or your authorized user's actions, including those actions that affect infrastructure services, service availability, or data loss. If any such actions are detected, the cluster

might transition to a Limited Support status. You should then either revert the action or create a support case to explore remediation steps.
    - In some cases, the cluster can return to a fully supported status if you remediate the violating factors. However, in other cases, you might have to delete and recreate the cluster.
    - For more information, see the Azure Red Hat OpenShift support policy about cluster configuration requirements.

FAQ

What happens when a user updates an OpenShift cluster with a minor version that isn't supported?

Azure Red Hat OpenShift supports installing minor versions consistent with the dates in the previous table. A version is supported as soon as an update path to that version is available in the stable channel. If you're running a version past the End of Life date, you're outside of support and might be asked to update to continue receiving support. Updating from an older version to a supported version can be challenging, and in some cases not possible. We recommend you keep your cluster on the latest OpenShift version to avoid potential update issues.

For example, if the oldest supported Azure Red Hat OpenShift version is 4.16 and you are on 4.15 or older, you're outside of support. When the update from 4.15 to 4.16 or higher succeeds, you're back within our support policies.

Reverting your cluster to a previous version, or a rollback, isn't supported. Only updating to a newer version is supported.

What does "Outside of Support" or "Limited Support" mean?

If your cluster is running an OpenShift version that isn't on the supported versions list, or is using an unsupported cluster configuration, your cluster is *outside of support*. As a result:

- When you open a support ticket for your cluster, you might be asked to update the cluster to a supported version before receiving support.
- Any runtime or SLA guarantees for clusters outside of support are voided.
- Clusters outside of support are patched only on a best effort basis.
- Clusters outside of support aren't monitored.

Next steps

For more support information, see Azure Red Hat OpenShift 4.0 support policy.

Last updated on 11/14/2025