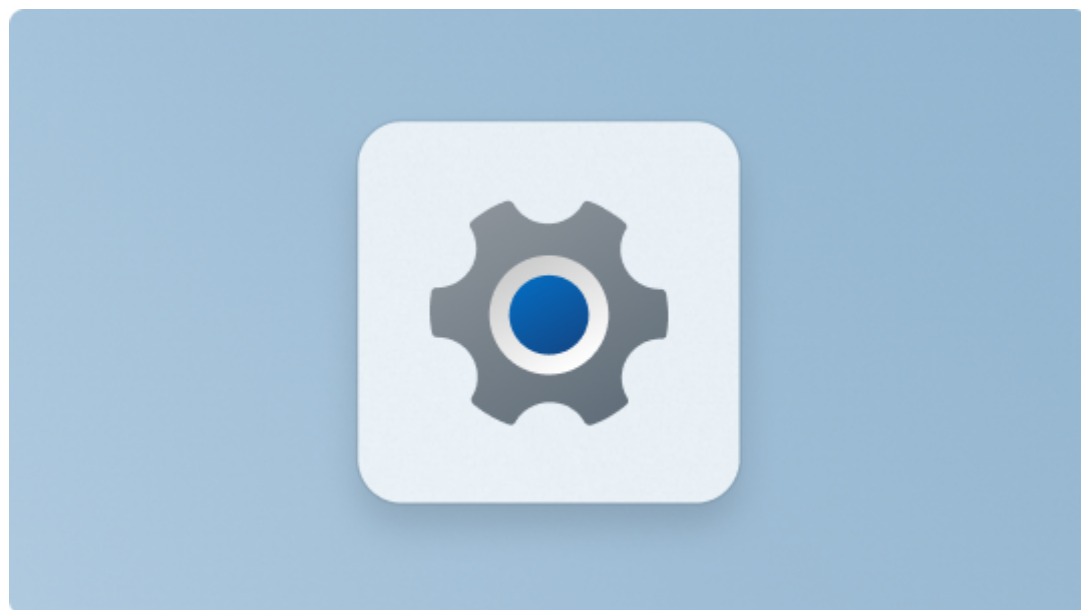


Windows 開発者ツール

Windows では、さまざまな開発ワークフローがサポートされています。Visual Studio Code や GitHub Copilot などのツールは Windows に十分に統合されており、エディターとターミナルに組み込まれている AI の支援により、作業を自動化するためのスクリプトやビルド ユーティリティを簡単に作成できます。Windows、Web、またはクロスプラットフォーム用に開発しているかどうかにかかわらず、このページでは、使用を開始するためのツールとセットアップ ガイドを示します。

作業を開始するためのツール



詳細設定

開発者モードやエクスプローラーのソース管理などのWindows設定を構成します。



CLI ツール

Coreutils、curl、Edit、sudo、tar など、Windows用の組み込みのコマンドラインユーティリティ。



Dev Configs

Windows開発環境を 1 つの手順で構成する、キュレーションされた宣言型セットアップ。



Microsoft PowerToys

Windows エクスペリエンスを調整および合理化するためのパワー ユーザー ユーティリティのセット。



WinGet

コマンド ライン マニフェストまたは宣言型マニフェストを使用してパッケージをインストールして構成します。



Linux 用 Windows サブシステム

Windows に完全に統合されたお気に入りの Linux ディストリビューションを実行します。



Windows ターミナル

複数のコマンド ライン シェルで動作するようにターミナル環境をカスタマイズします。

Microsoft Ignite の Windows 開発者ツールの概要を見て、これらのツールの動作を確認します。

セットアップ ガイド

次のガイドに従って、Windows で好みの言語またはフレームワークの開発環境を設定します。



JavaScript

Node.js、React、Vue などを使用して JavaScript 開発環境を設定します。



Python

Python をインストールし、Windows に開発環境を設定します。



ジャワ

JDK をインストールし、JAVA_HOMEを構成し、Windows で Java 開発環境を設定します。



C および C++

C++, C、およびアセンブリを使用してアプリ、サービス、およびツールを開発します。



C#

C# と .NET を使用してアプリをビルドします。



Docker

VS、VS Code、.NET、WSL のサポートを使用してリモート開発コンテナを作成します。



Powershell

PowerShell を使用したクロスプラットフォーム タスクの自動化と構成管理。



錆

Rust for Windows の設定方法など、Rust のプログラミングを開始します。



Windows アプリ開発

WinUI と Windows アプリ SDK を使用してデスクトップ アプリを構築します。



mac to Windows guide

Mac とWindows (またはLinux 用 Windows サブシステム) 開発環境の間の遷移。

Last updated on 2026/06/02

Windows の詳細設定

Windows の **[詳細設定]** ページ (旧称 :**開発者向け**) には、開発者とパワー ユーザーの生産性を高めるのに役立つ設定がまとめられています。 オープンソースプロジェクトであり、投稿やフィードバックは歓迎されています。 Windows設定で **システム > 詳細設定** を使用して直接開きます。

詳細設定を開く  [GitHubで表示](#)

内容

 **テーブルを展開する**

カテゴリ	Setting	[説明]
タスクバー	タスクの終了	タスク バーの終了タスクを右クリックして有効にする
ファイル エクスプロ ーラー	エクスプローラー	エクスプローラーを使用して開発者にとってわかりやすい エクスペリエンスを得るための設定を調整する
	↳ ファイル拡張子を表示する	エクスプローラーでファイル拡張子を表示する
	↳ 非表示ファイルとシステム ファイルを表示する	エクスプローラーで非表示ファイルとシステム ファイルを 表示する
	↳ タイトル バーに完全なパス を表示する	エクスプローラーのタイトル バーに完全なパスを表示する
	↳ スタート画面で別のユーザ ーとして実行するオプション を表示する	[スタート] メニューで別のユーザーとして実行するオプシ ョンを表示する
	↳ 空のドライブを表示する	エクスプローラーで空のドライブを表示する
ファイル エクスプロ ーラー	ファイル エクスプローラー + バージョン管理	エクスプローラーでブランチ、コミット、および作成者の 詳細を直接表示します。 エクスプローラーのバージョン管 理の統合に関するページ を参照してください
ファイル エクスプロ ーラー	長いパスを有効にする	一般的な Win32 ファイルおよびディレクトリ関数から MAX_PATH制限を削除する
仮想ワーク スペース	リモート デスクトップ	リモート デスクトップを有効にし、マシンの可用性を確保 する
仮想ワーク スペース	仮想ワークスペース	仮想ワークスペースの設定を調整する

カテゴリ	Setting	[説明]
ターミナル	Terminal	コマンド ライン アプリをホストする既定のターミナル アプリを選択する
ターミナル	PowerShell	これらの設定を有効にして PowerShell スクリプトを実行する
ターミナル	sudo を有効にする	sudo コマンドを有効にします。 Windows 用 Sudo を参照してください
開発者向け	開発者モード	ルーズ ファイルを含む任意のソースからアプリをインストールします。 開発者向けの設定 を参照してください
開発者向け	デバイスポータル	ローカル エリア ネットワーク接続経由でリモート診断を有効にする
Dev ドライブ	開発用ドライブ	開発者シナリオでのパフォーマンス用に最適化されたストレージ ボリューム

可用性に関する注意事項

- Windows 11 バージョン 25H2 以降では、以前の **開発者向け** 設定が **[詳細設定]** ページに表示されるようになりました。
- Windows 用 Sudo は、Windows 11 バージョン 24H2 以降で使用できます。 [「Windows 用 Sudo」](#) を参照してください。
- エクスプローラーのバージョン管理の統合は、現在、Windows Insider Beta チャネルを介して利用できます。登録とセットアップの詳細については、 [エクスプローラーのバージョン管理の統合](#) に関するページを参照してください。

トラブルシューティング

- オプションが見つからないか無効:** 組織によって管理されているデバイスでは、詳細設定の切り替えを無効にするポリシーが適用される場合があります。 関連する [グループ ポリシーのガイダンス](#)を参照してください。
- 管理者が必要:** 開発者モードを有効にするには、ローカル管理者特権が必要です。
- 機能が見つかりません:** デバイスがサポートされている Windows バージョンに更新されていることを確認します。 プレビュー機能では、Windows Insider チャネルへの参加が必要になる場合があります。

Windows 11 で Dev Drive を設定する

Dev Drive は、主要な開発者ワークロードのパフォーマンスを向上させるために使用できる新しい形式のストレージ ボリュームです。

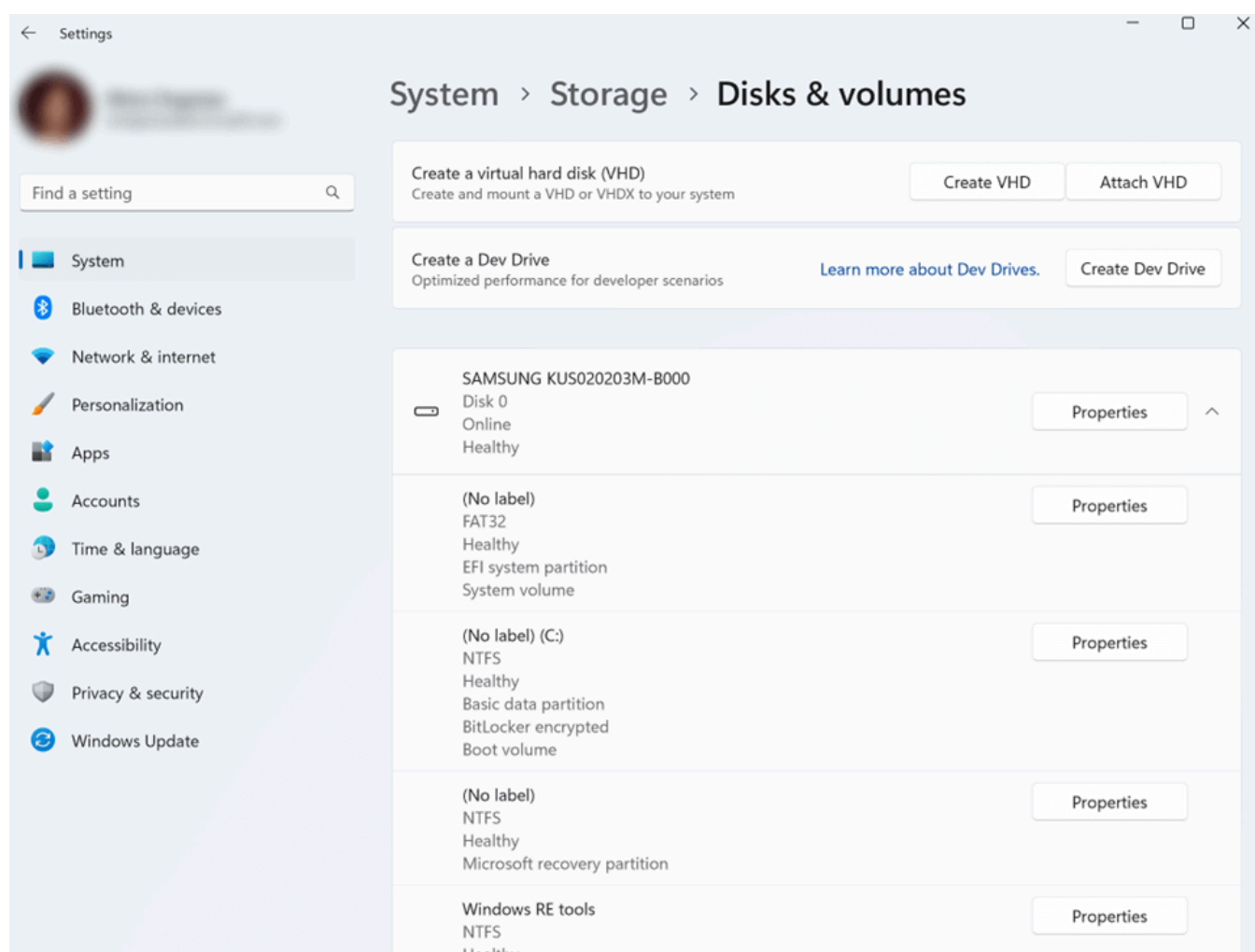
Dev Drive は [ReFS](#) テクノロジーに基づいて構築されており、対象となるファイル システムの最適化を採用し、信頼の指定、ウイルス対策の構成、アタッチされるフィルターの管理制御など、ストレージ ボリュームの設定とセキュリティをより詳細に制御できます。

ブログ記事「[Visual Studio のパフォーマンス向上のための開発ドライブ](#)」と、[☞](#) 一般的な開発操作全体の平均改善測定値については、Dev Boxes を参照してください。

Dev Drive の設定方法

新しい開発ドライブを設定するには、Windows **設定** を開き、**System>Storage>Advanced Storage Settings>Disks & volumes** に移動します。[**開発ドライブの作成**] を選択します。既存のストレージ ボリュームを Dev Drive に変換することはできません。Dev Drive の指定は、最初のフォーマット時にのみ行われます。

開発ドライブを設定する前 に、**前提条件が満たされていることを確認**します。



前提条件

- Windows 11 ビルド #10.0.22621.2338 以降 ([Windows 更新プログラムの確認](#))
- 16 GB のメモリ (最小 8 GB) をお勧めします
- 最小 50 GB の空きディスク領域
- 開発者ドライブ はすべての Windows SKU バージョンで使用可能
- ローカル管理者のアクセス許可。

最新の Windows 11 リリースに更新する場合は、Dev Drive 機能を利用できるようになる前に、追加の再起動が必要になる場合があります。ビジネス エンタープライズ環境で作業している場合、Dev Drive を有効にするには、セキュリティ管理者が [Dev Drive セキュリティ ポリシーを構成](#) する必要があります。

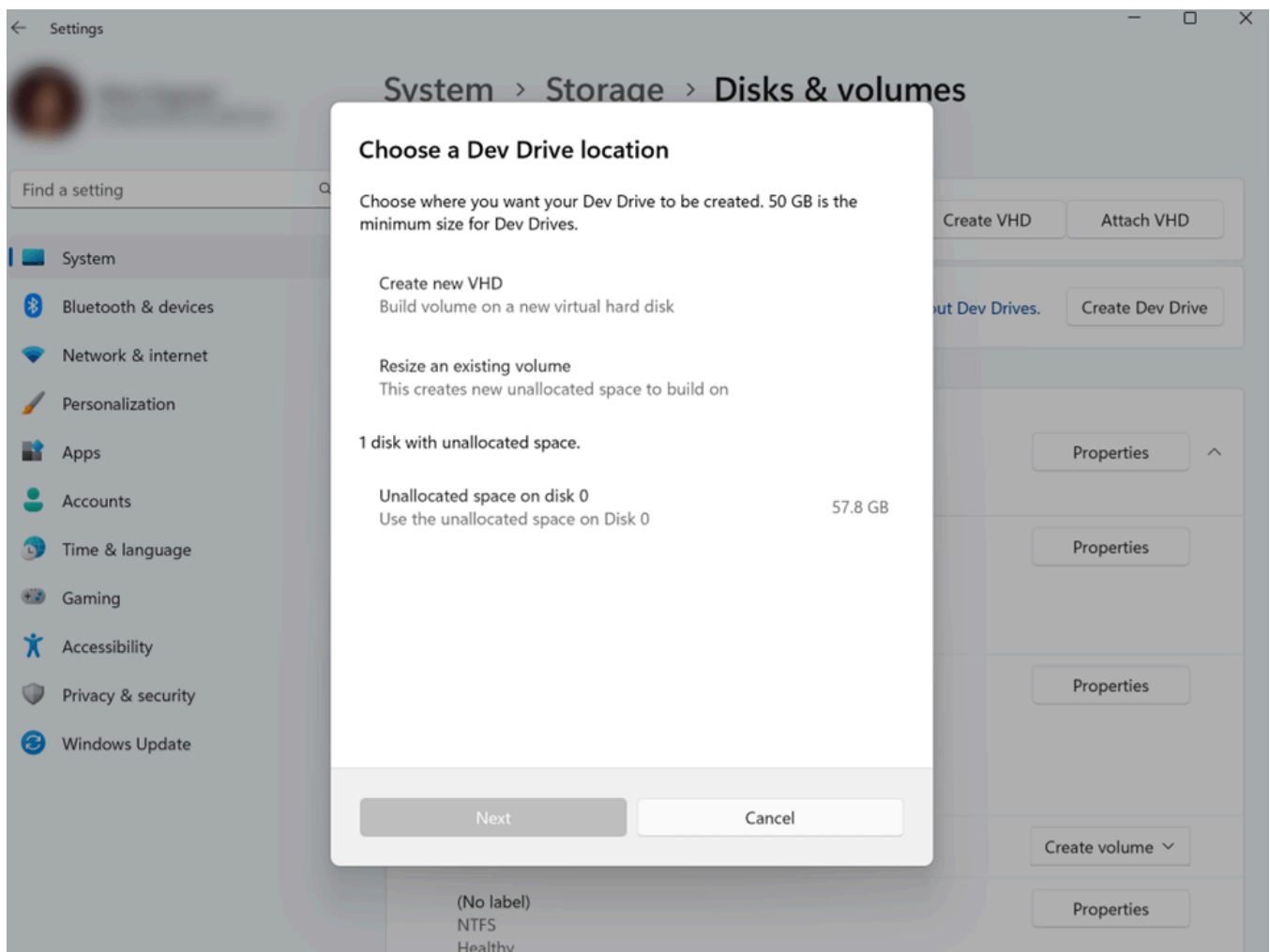
⚠ 警告

Dev Drive は 主要な開発者シナリオ のみを対象としており、カスタム設定は引き続きビジネスまたはエンタープライズの作業環境の グループ ポリシー 設定でカバーされます。Dev Drive のセキュリティ ポリシーを構成 する方法の詳細について説明します。

オプションを設定する

次の 3 つのオプションを使用できます。

1. **新しい VHD を作成** する - 新しい仮想ハード ディスクにボリュームをビルドする
2. **既存のボリュームのサイズを変更する** - 新しい未割り当て領域を作成して構築する
3. **ディスク上の未割り当て領域** - 既存のディスク上の未割り当て領域を使用します。" *このオプションは、以前にストレージで未割り当て領域を設定した場合にのみ表示されます。 "



ディスク パーティションまたは VHD のどちらを使用するかを選択する方法

[ディスク パーティションを作成](#)するか、開発ドライブを格納する[新しい VHD を作成](#)するかを選択する際に考慮すべき利点とトレードオフがあります。

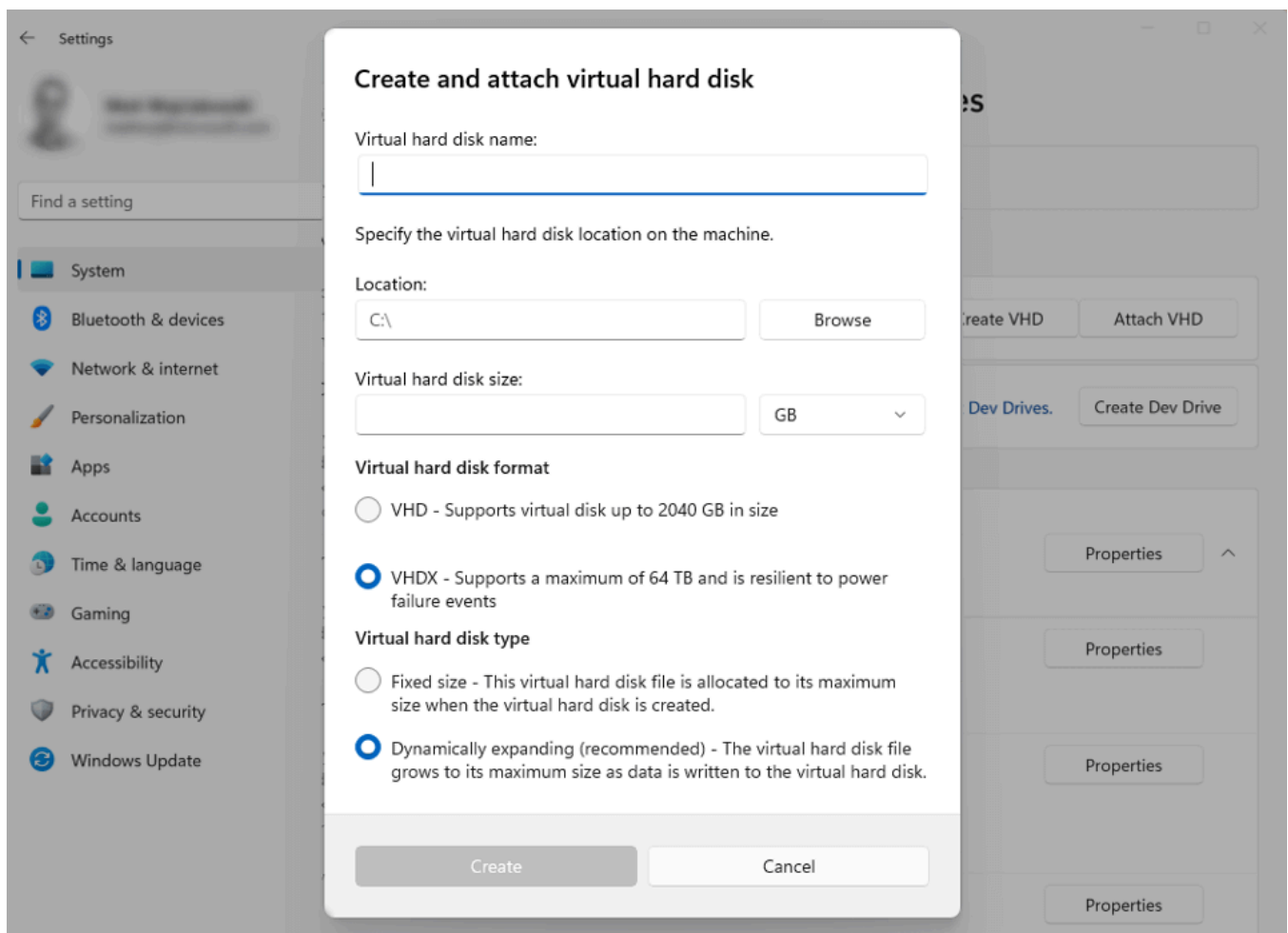
- **ディスク パーティションを作成**する: Dev Drive をディスク パーティションに格納すると、通常、追加のレイヤーなしで物理ディスクを直接使用するため、パフォーマンスが向上します。パーティションが物理ディスクに関連付けられているため、パーティションのサイズ変更は複雑でリスクが高く、移植性が低下する可能性があるため、パーティションされたディスクを使用する方が柔軟性が低くなります。
- **新しい VHD を作成**する: 仮想ハード ディスク (VHD) に開発ドライブを格納すると、仮想ディスク レイヤーを管理するオーバーヘッドが原因でパフォーマンスが若干低下する可能性があります。トレードオフとして、VHD は動的なサイズ変更 (ディスク領域を効率的に管理する必要がある場合)、データの移動、またはバックアップを柔軟に行うことができます。VHD も移植性が高く、VHD ファイルを別のコンピューターまたはバックアップの場所に転送できます。固定ディスク (HDD または SSD) 上で VHD がホストされる場合は、VHD をコピーして別のマシンに移動し、それを開発ドライブとして使用し続けることは推奨されません。

新しい VHD を作成する

[**新しい VHD の作成**] オプションを選択して開発ドライブを設定する場合は、次の内容を決める必要があります。

- **仮想ハード ディスク名:** VHD (Dev Drive) に名前を付けます。
- **場所:** Dev Drive VHD がコンピューター上に配置されるディレクトリ パスを割り当てます。既定の場所は `c:\` です。意図しない共有を回避するために、ユーザーごとのディレクトリ パスの場所を使用して Dev Drive を格納することをお勧めします。
- **仮想ハード ディスク サイズ:** 使用するボリュームに割り当てるディスク領域の量を割り当てます。最小サイズは 50 GB です。
- **仮想ハード ディスク形式:**
 - VHD: 最大 2040 GB のサイズの仮想ディスクをサポートします。
 - VHDX (推奨): 最大 64 TB をサポートし、停電などの問題によって発生する予期しない IO 障害に対する回復性の高い保護を提供します。 [VHD の管理の詳細を確認します](#)。
- **ディスクの種類:**
 - **固定サイズ** - この仮想ハード ディスク ファイルは、作成時に最大サイズに割り当てられます。
 - **動的に拡張する** - データがディスクに書き込まれると、仮想ハード ディスク ファイルは最大サイズに拡張されます。(おすすめ)

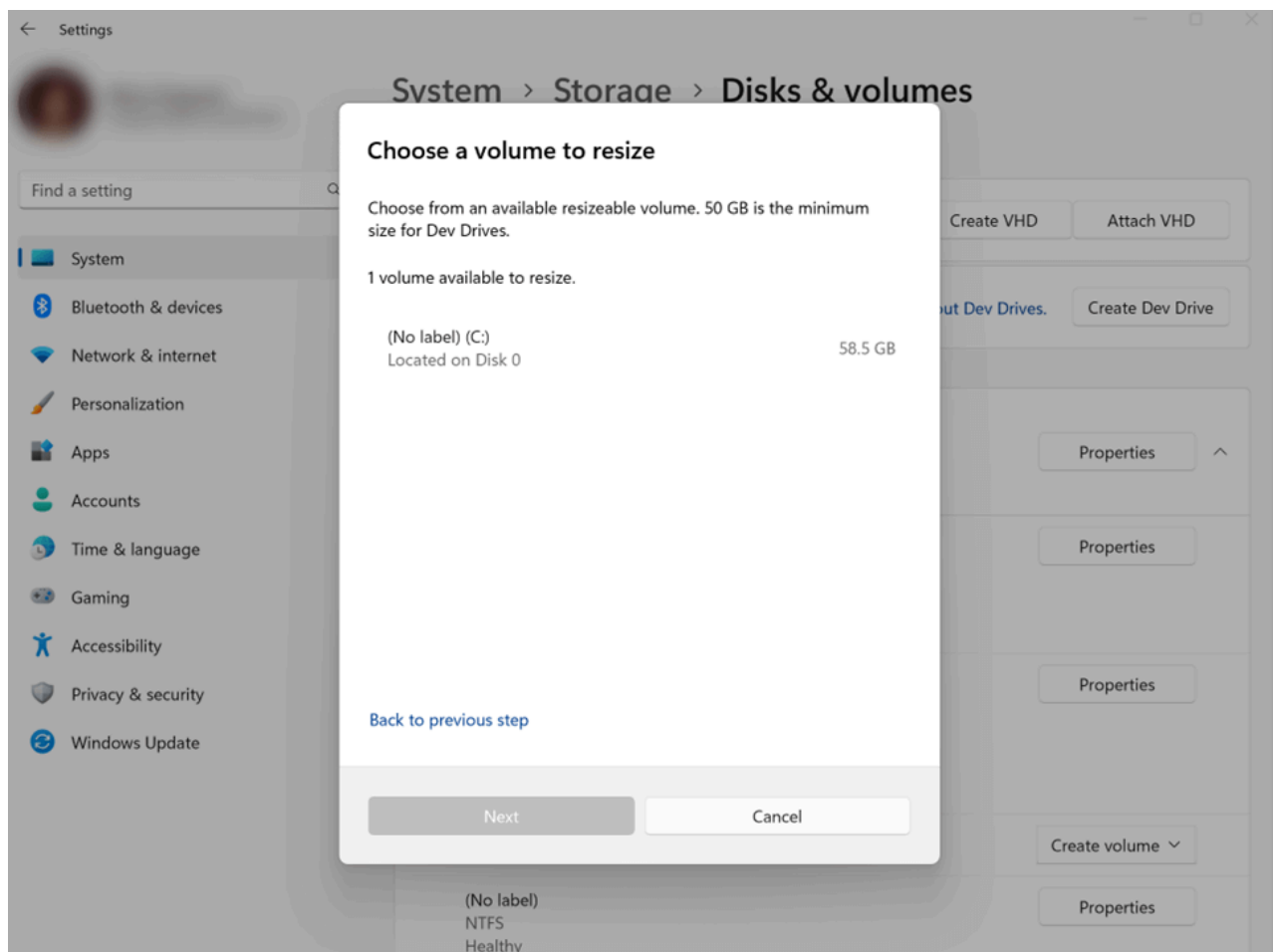
これらのオプションを選択するプロセスが完了すると、Dev Drive が作成されます。



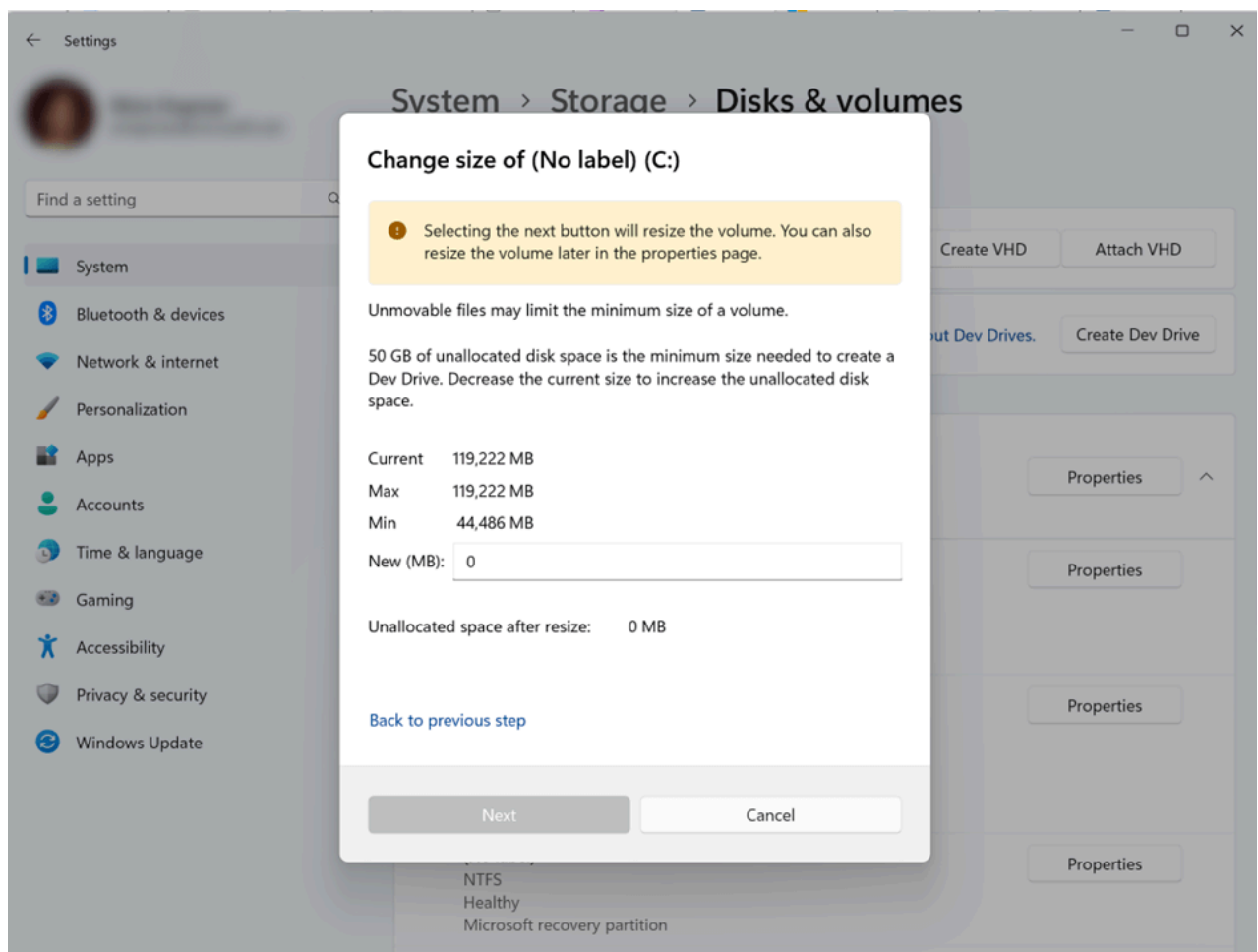
既存のボリュームのサイズを変更するか、既存のディスク上の未割り当て領域を使用する

既存のボリュームのサイズを変更するには:

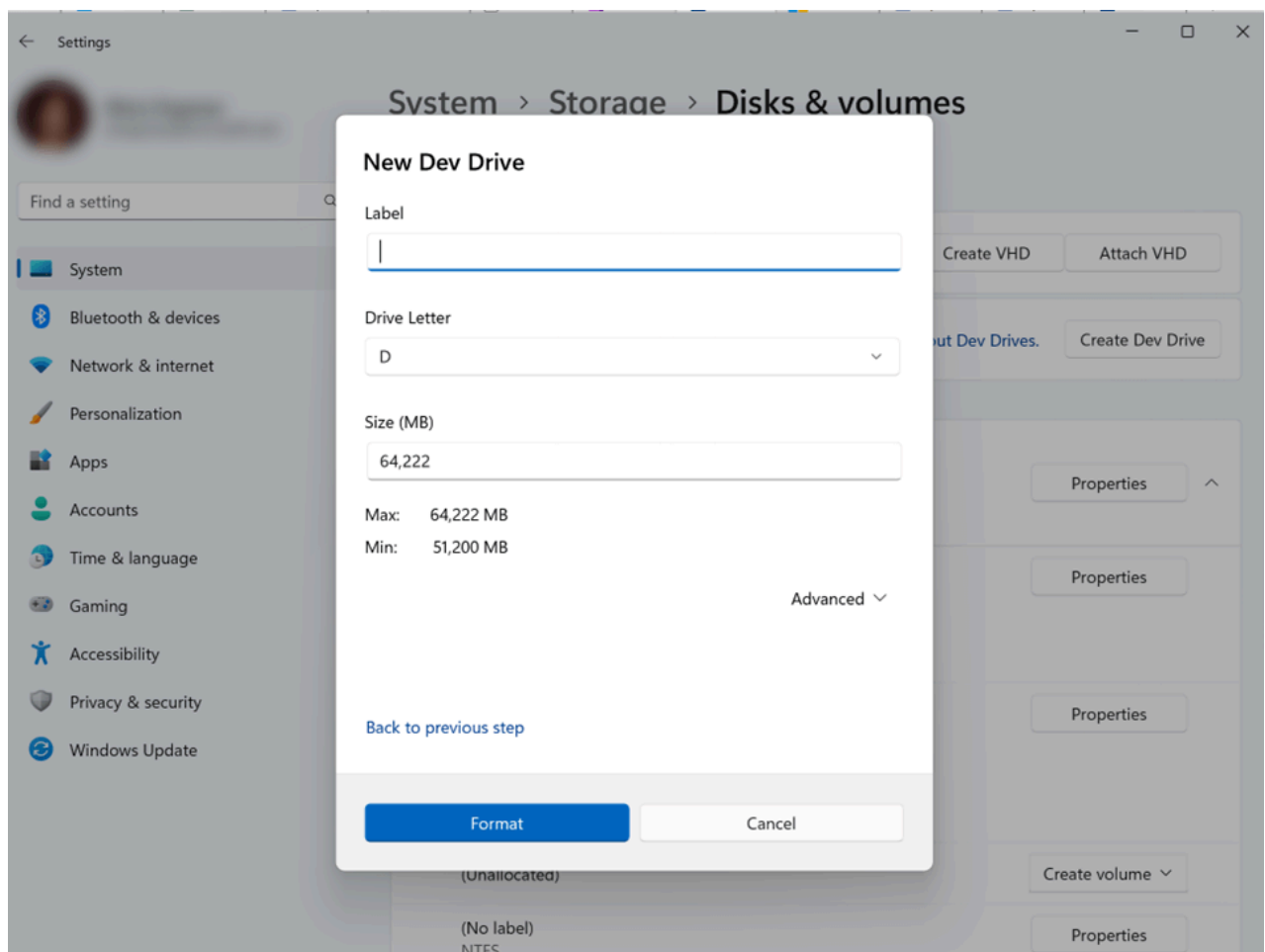
1. サイズを変更するボリュームを選択します。



2. ボリュームの新しいサイズを選択します。Dev Drive に必要な最小サイズである、少なくとも 50 GB の未割り当て領域が使用可能である必要があります。サイズが設定されたら、[次へ] を選択します。



3. 新しい空き領域に開発ドライブをフォーマットするには、**ラベル** (ドライブ名)、**ドライブ文字**、および **サイズ** の割り当てを指定します。最大サイズは、前の手順で割り当てた空き領域の量です。開発ドライブの最小サイズは 50 GB です。



おめでとうございます。これで、Dev Drive のサイズが変更されました。

既存のドライブで未割り当て領域を検索して使用するには、**System>Storage>Disks > ボリューム**を開き、ページを見て、記憶域スペースが "未割り当て" と表示されているかどうかを確認します。[**ボリュームの作成**] を選択すると、**簡易ボリューム** (標準の NTFS ストレージ ボリューム) または **開発ドライブの作成** を選択できます。開発ドライブを作成するには、上記と同じ手順を実行します。**ラベル** (ドライブ名)、**ドライブ文字**を追加し、**サイズ** の割り当てを確認する必要があります。

System > Storage > Disks & volumes

	SAMSUNG KUS020203M-B000 Disk 0 Online Healthy	Properties
	(No label) FAT32 Healthy EFI system partition System volume	Properties
	(No label) (C:) NTFS Healthy Basic data partition BitLocker encrypted Boot volume	Properties Create Simple Volume Create Dev Drive
	(Unallocated)	Create volume ▾

コマンド ラインからストレージ ボリュームを開発者ドライブとして書式設定する

Windows 設定を使用する代わりに、コマンド ラインから Dev Drive ストレージ ボリュームを作成する場合は 2 つのオプションがあります。いずれのオプションも、管理者としてコマンド ラインを開く必要があります。ハード ドライブをフォーマットするには、管理者グループのメンバーである必要があります。複数の開発者ドライブを作成する場合や、複数のコンピューターの管理者として使用する場合は、これらのコマンド ラインの書式設定方法が推奨されます。

1. Windows CMD または PowerShell の [Format](#) コマンド ライン ツールの使用:

CMD

```
Format D: /DevDrv /Q
```

2. PowerShell の [Format-Volume](#) コマンドレットの使用:

PowerShell

```
Format-Volume -DriveLetter D -DevDrive
```

これらのコード サンプルでは、ターゲットにするドライブの場所に **D:** を置き換える必要があります。その他のオプションとコマンド パラメーターの詳細については、リンクを参照してください。

Dev Drive のしくみ

ストレージ ボリュームは、特定の形式で、ディレクトリとファイルを介してファイル システムにデータを格納する方法を指定します。Windows ではシステム ドライブに NTFS を使用し、ほとんどの非リムーバブル ドライブに既定で NTFS を使用しています。**Resilient File System (ReFS)** は、データの可用性を最大化し、多様なワークロードにわたって大規模なデータ セットに効率的にスケーリングし、破損に対する回復性を備えたデータ整合性を提供するように設計された、新しい Microsoft ファイル システム形式です。拡大する一連のストレージ シナリオに対処し、将来のイノベーションの基盤を確立することを目指しています。

開発ドライブは ReFS を利用して、開発ワークロード専用のストレージ ボリュームを初期化し、より高速なパフォーマンスを提供し、開発シナリオ向けに最適化されたカスタマイズ可能な設定を提供します。ReFS には、主要な開発者シナリオのパフォーマンスを向上させるために、いくつかのファイル システム固有の最適化が含まれています。

[Dev Drive によるセキュリティの処理方法](#)の詳細について説明します。

Dev Drive に何を格納すべきですか？

Dev Drive は次を対象としています。

- ソース コード リポジトリとプロジェクト ファイル
- パッケージ キャッシュ
- ビルドの出力ファイルと中間ファイル

開発ドライブへの開発者ツールと SDK のインストールに関する考慮事項: 開発者ツールと SDK は、通常、管理者またはユーザーごとの場所に配置されます。これらの場所は、Windows での特定のセキュリティと分離の保証を提供し、Microsoft Defender の動作に影響を与えます。ただし、多くのツールでは、Dev Drive を含め、インストール場所を柔軟に選択できます。

Dev Drive に開発者ツールまたは SDK をインストールする前に、システムと非同期スキャンに関連付けられているトレードオフを評価して、デバイスと組織のセキュリティ要件と一致していることを確認します。Dev Drive に管理者またはユーザー単位のフォルダーを作成するオプションがあります。さらに、Microsoft Defender パフォーマンス モード (非同期スキャンなど) がバイナリを処理するためのニーズを満たしていることを確認することが重要です。

❗ Note

IT 管理者は、EOP 攻撃を回避するためのベスト プラクティスとして、マルチユーザー デバイス用のアクセス制御リスト (ACL) フォルダーをユーザーごとに作成する必要があります。

Dev Drive にパッケージ キャッシュを格納する

パッケージ キャッシュとは、インストールされているソフトウェアのファイルを格納するためにアプリケーションによって使用されるグローバル フォルダーの場所です。これらのソース ファイルは、インストールされているソフトウェアを更新、アンインストール、または修復する際に必要です。Visual Studio は、そのデータの大部分をパッケージ キャッシュに格納するアプリケーションの 1 つです。環境変数を変更した後、**開いているすべてのコンソール ウィンドウを再起動するか**、新しい値を適用するためにデバイスを再起動する必要があります。

- **Npm キャッシュ (NodeJS):**開発ドライブに npm キャッシュ ディレクトリを作成し (たとえば、`D:\packages\npm`)、グローバル環境変数 `npm_config_cache` そのパス (たとえば、`setx /M npm_config_cache D:\packages\npm`) に設定します。コンピューターに NodeJS を既にインストールしている場合は、`%AppData%\npm-cache` の内容をこのディレクトリに移動します。(一部のシステムでは、npm キャッシュは `%LocalAppData%\npm-cache` にあることがあります)。npm ドキュメントの詳細については、[npm-cache](#) と [npm config: cache](#) を参照してください。
- **NuGet グローバル パッケージ フォルダー:** NuGet グローバル パッケージ フォルダーは、dotnet、MSBuild、および Visual Studio によって使用されます。CopyOnWrite (CoW) ファイルシステムに、ユーザー固有の NuGet ディレクトリを作成します。たとえば、`D:\<username>\.nuget\packages` と指定します。次のいずれかの方法で、グローバル パッケージ フォルダーを既定の場所から新しく作成したフォルダーに変更します (グローバルにインストールされたパッケージを管理するため)。
 - グローバル環境変数 `NUGET_PACKAGES` を該当のパスに設定します。たとえば、`setx /M NUGET_PACKAGES D:\<username>\.nuget\packages` と指定します。
 - 構成設定で、`globalPackagesFolder` (`PackageReference` を使用する場合) または `repositoryPath` (`packages.config` を使用する場合) を該当のパスに設定します。
 - `RestorePackagesPath` MSBuild プロパティ (MSBuild のみ) を該当のパスに設定します。

global-packages フォルダーを確認するには、dotnet nuget locals コマンド `dotnet nuget locals global-packages --list` を実行します。復元により、パッケージが新しいパスにインストールされ、ダウンロードされます。既定の NuGet グローバルパッケージフォルダーは削除できます。詳細については、[NuGet のドキュメント「グローバルパッケージ、キャッシュ、および一時フォルダーの管理」](#)を参照してください。

❗ Note

現在、既知の問題があります。 [Dotnet ツール コマンドは nuget パッケージのパスを考慮しません](#)。 .NET チームは、.NET 10 の修正プログラムと 8.0 および 9.0 のサービスリリース更新プログラムを認識し、調査しています。

- **vcpkg キャッシュ:** dev Drive に vcpkg キャッシュ ディレクトリを作成します (たとえば、 `D:\packages\vcpkg`)。そのパスに `VCPKG_DEFAULT_BINARY_CACHE` グローバル環境変数を設定します (例: `setx /M VCPKG_DEFAULT_BINARY_CACHE D:\packages\vcpkg`)。パッケージを既にインストールしている場合は、 `%LOCALAPPDATA%\vcpkg\archives` または `%APPDATA%\vcpkg\archives` の内容をこのディレクトリに移動します。 vcpkg ドキュメントの詳細については、 [vcpkg バイナリ キャッシュ](#)を参照してください。
- **Pip cache (Python):**開発ドライブに pip キャッシュ ディレクトリを作成します (たとえば、 `D:\packages\pip`)。そのパスに `PIP_CACHE_DIR` グローバル環境変数 (`setx /M PIP_CACHE_DIR D:\packages\pip` など) を設定します。 コンピューターで pip パッケージと Wheel を既に復元している場合は、 `%LocalAppData%\pip\Cache` の内容をこのディレクトリに移動します。 pip ドキュメントの詳細については、 [pip キャッシュ](#) に関するページを参照し、 [Linux 上の pip キャッシュのディレクトリを変更](#) するための StackOverflow を参照してください。
- **Cargo Cache (Rust):**開発ドライブに Cargo キャッシュ ディレクトリを作成します (たとえば、 `D:\packages\cargo`)。そのパスに `CARGO_HOME` グローバル環境変数 (`setx /M CARGO_HOME D:\packages\cargo` など) を設定します。 コンピューターで Cargo パッケージを既に復元している場合は、 `%USERPROFILE%\cargo` の内容をこのディレクトリに移動します。 詳細については、Cargo ドキュメントの「[Cargo 環境変数](#)」を参照してください。
- **Maven キャッシュ (Java):**Dev Drive に Maven キャッシュ ディレクトリを作成します (たとえば、 `D:\packages\maven`)。次に、グローバル環境変数 `MAVEN_OPTS` を設定して、そのパスに構成設定を追加します (例: `setx /M MAVEN_OPTS "-Dmaven.repo.local=D:\packages\maven"`)。 `%USERPROFILE%\m2\repository` の内容をこのディレクトリに移動します (これには、Maven が `repository` フォルダーにダウンロードし

てプロジェクトに使用する依存関係、プラグイン、その他の成果物のみが含まれます)。
[Maven ドキュメント](#)の詳細については、「[.m2 フォルダーまたは settings.xml 永続的に別の場所を指定する方法](#)」の StackOverflow を参照してください。

- **Gradle キャッシュ (Java):**たとえば、`D:\packages\gradle` など、開発ドライブに Gradle キャッシュ ディレクトリを作成します。次に、そのパスを指すようにグローバル環境変数の `GRADLE_USER_HOME` を設定します。たとえば、コマンド ラインで `setx /M GRADLE_USER_HOME "D:\packages\gradle"` を使用してシステム全体にその変数を設定します。この変数を設定したら、Gradle は指定されたディレクトリ (`D:\packages\gradle`) をキャッシュと構成ファイルに使用します。既存の Gradle ファイルがある場合は、`%USERPROFILE%\gradle` のコンテンツをこの新しいディレクトリに移動します。詳細については、[Gradle のドキュメント](#)を参照し、[Gradle の構成とキャッシュ ディレクトリの管理に関するヒント](#)については、StackOverflow などのコミュニティ リソースを参照してください。

Dev Drive に関連するセキュリティ リスクと信頼を理解する

セキュリティと信頼は、プロジェクト ファイルを操作する際の重要な考慮事項です。通常、パフォーマンスとセキュリティの間にはトレードオフがあります。Dev Drive を使用すると、開発者とセキュリティ管理者の手でこのバランスを制御できます。その場合、追加するフィルターと、Microsoft Defender ウイルス対策スキャンの設定を選択する責任があります。

既定では、Microsoft Defender とサード パーティのウイルス対策フィルターの両方を含むウイルス対策フィルターが Dev Drive に追加されます。Microsoft Defender ウイルス対策は、既定で Dev Drives の新しい "[パフォーマンス モード](#)" 設定に設定され、速度とパフォーマンスを考慮しながら、フォルダーの除外に代わる安全な代替手段を提供します。保護レベルを上げるための Microsoft Defender には、"[リアルタイム保護モード](#)" も用意されています。

追加のフィルターを必要とする製品または機能は、フィルターが [Dev Drive に追加](#)されない限り機能しません。

⚠ 警告

Dev Drive は、ウイルス対策フィルターが追加されていない状態で実行できます。細心の注意を払ってください。ウイルス対策フィルターを削除することはセキュリティ上のリスクであり、ストレージ ドライブが標準のセキュリティ スキャンの対象にならないことを意味します。あなたは、ウイルス対策フィルターの削除に関連するリスクを評価する責任を負っています。これを行うのは、Dev Drive に保存されているファイルが悪意のある攻撃にさらされないことを確信している場合にのみしてください。

Microsoft では、[信頼できる開発ドライブ](#)を使用する場合は、既定のパフォーマンス モード設定を使用することをお勧めします。

"信頼済み" の Dev Drive とは

Dev Drive は、最初のフォーマット時にシステム レジストリに格納されたフラグを使用して "信頼済み" として自動的に指定され、可能な限り最高のパフォーマンスを既定で提供します。 "信頼済み" の Dev Drive とは、ボリュームを使用する開発者が、そこに格納されているコンテンツのセキュリティに高い信頼を置いていることを意味します。

開発者が [Windows セキュリティに除外を追加](#) することを選択した場合と同様に、開発者は、追加のパフォーマンスを得るために、格納されているコンテンツのセキュリティを管理する責任を負います。

信頼済みとしてマークされた開発ドライブは、Microsoft Defender が [パフォーマンス モード](#) で実行するためのシグナルです。 Microsoft Defender がパフォーマンス モードで実行されていると、脅威の防止とパフォーマンスのバランスが維持されます。 リアルタイム保護は、他のすべてのストレージ ボリュームで引き続き有効になります。

フィルターをデタッチするセキュリティ上の考慮事項により、コンピューター間で Dev Drive を転送すると、ボリュームは特別なフィルター アタッチ ポリシーのない通常のボリュームとして扱われます。 ボリュームは、新しいコンピューターへのアタッチ時に "信頼済み" とマークされる必要があります。 [Dev Drive を信頼済みとして指定する方法](#)を参照してください。

"信頼されていない" Dev Drive には、"信頼済み" の Dev Drive と同じ特権は付与されません。 開発ドライブが [信頼されていない](#) 場合、セキュリティはリアルタイム保護モードで実行されます。 最初の作成時以外に Dev Drive に信頼を指定する場合は、注意が必要です。

Dev Drive を信頼済みとして指定する方法

Dev Drive を "信頼済み" として指定するには:

1. PowerShell (または CMD) を右クリックして [管理者として実行] をクリックし、昇格されたアクセス許可で開きます。
2. Dev Drive を 信頼済みとして指定するには、次のコマンドを入力し、 `<drive-letter>` を信頼を指定するストレージ ドライブの文字に置き換えます。 たとえば、 `fsutil devdrv trust D:` のように指定します。

PowerShell

```
fsutil devdrv trust <drive-letter>:
```

Dev Drive が信頼済みになったかどうかを確認するには、次のコマンドを入力します。

PowerShell

```
fsutil devdrv query <drive-letter>:
```

コンピューター上の C: ドライブを Dev Drive として指定することはできません。Visual Studio、MSBuild、.NET SDK、Windows SDK などの開発者ツールは、Dev Drive ではなく C: ドライブに格納する必要があります。

Microsoft Defender パフォーマンス モードとは

新しい Microsoft Defender ウイルス対策機能として、パフォーマンス モードが Windows 11 で使用できるようになりました。この機能により、指定された Dev Drive に格納されているファイルに対する Microsoft Defender ウイルス対策スキャンのパフォーマンスへの影響が軽減されます。

パフォーマンス モードとリアルタイム保護との比較の詳細については、「[Microsoft Defender: パフォーマンス モードを使用した開発ドライブの保護](#)」を参照してください。

パフォーマンス モードを有効にするには、Dev Drive を“信頼済み”として指定し、Microsoft Defender のリアルタイム保護を“オン”に設定する必要があります。

Dev Drive での追加のフィルターの構成方法

既定では、[フィルター マネージャー](#) は、ウイルス対策フィルターを除き、開発ドライブ上のすべてのフィルターをオフにします。ウイルス対策フィルターとは、FSFilter Anti-Virus 高度範囲 (320000-329999) にアタッチされているフィルターです。FSFilter Anti-Virus には、ファイル I/O 中にウイルスを検出して駆除するフィルターが含まれています。

Dev Drive にウイルス対策フィルターを追加しないように既定のポリシーを構成できます。そのためには、`fsutil` を使用します。**注意:** このポリシーは、システム上のすべての開発ドライブに適用されます。

PowerShell

```
fsutil devdrv enable /disallowAv
```

コマンド `fsutil devdrv enable [/allowAv|/disallowAv]` には、次の 2 つのオプションがあります。

- `disallowAv`: Dev Drive にフィルター (ウイルス対策も) を追加しないことを指定します。
`fsutil devdrv setfiltersallowed <Filter-1>` コマンドを使用すると、フィルターを再度追加できます。(`<Filter-1>` を目的のフィルターの名前に置き換えてください。)

- `allowAv`: 既定のウイルス対策フィルターによって Dev Drive を保護することを指定します。

ヘルプを表示するには、`fsutil devdrv enable /?` コマンドを入力します。`/allowAv` と `/disallowAv` のどちらも指定されていない場合、Dev Drive のウイルス対策ポリシーは構成されず、システムの既定値でウイルス対策フィルターによって Dev Drive が保護されます。

⚠ 警告

フィルターを削除するときは細心の注意を払ってください。ウイルス対策フィルターを削除することはセキュリティ上のリスクであり、ストレージが標準のMicrosoft Defender リアルタイム保護またはパフォーマンス モード スキャンの対象にならないことを意味します。あなたは、ウイルス対策フィルターの削除に関連するリスクを評価する責任を負っています。これを行うのは、ファイルが悪意のある攻撃にさらされないことを確信している場合にのみにしてください。

フィルターの詳細については、「[ファイル システム フィルター ドライバーについて](#)」、[フィルター ドライバーのインストール](#)、[フィルター マネージャーの概念](#)、[ミニフィルター ドライバーの読み込み順序グループと高度](#)に関するページを参照してください。

Dev Drive で一部のフィルターの追加を許可する

ビジネスまたはエンタープライズの環境で作業している場合、上のポリシーに加えて、Dev Drive に一部のフィルターを追加するように会社のグループ ポリシーが構成されている場合があります。システム管理者は、**許可リスト**を使用して、特定の開発ドライブまたはすべての開発ドライブに追加のフィルターをアタッチすることもできます。

システム管理者は、"Foo" というフィルター (`FooFlt` と呼ぶ) を追加する場合があります。このフィルターを有効にするのは、**D:** としてマウントされた Dev Drive でのみの場合があります。このフィルターは、**E:** としてマウントした別の Dev Drive では必要ありません。管理者は、システム提供のコマンド ライン ユーティリティである `fsutil.exe` を使用して、Dev Drive のフィルターの許可リストに変更を加えることができます。

許可として特に設定されたフィルターは、上記で説明したウイルス対策フィルターポリシーに加えて、開発ドライブにアタッチできます。

許可リストによるフィルターの例

次の例は、許可リストを使用して、コンピューター上のすべての Dev Drive でフィルターを許可済みに設定する管理者機能を示しています。

`setfiltersallowed` コマンドを使用して `Filter-01` と `Filter-02` をすべての Dev Drive で許可するには、次のコマンドを使用します。

PowerShell

```
fsutil devdrv setfiltersallowed Filter-01, Filter-02
```

すべての Dev Drive のフィルター アタッチ ポリシーを表示するには、次のコマンドを使用します。

PowerShell

```
fsutil devdrv query
```

結果には、次の情報が表示されます。

- 開発者ボリュームが有効になっている。
- 開発者ボリュームが、ウイルス対策フィルターによって保護されている。
- すべての Dev Drive で `Filter-01` フィルターと `Filter-02` フィルターが許可されている。

Dev Drive で `Filter-03` のみを許可し、`Filter-01` と `Filter-02` のアタッチを許可しないようにこの Dev Drive の構成を変更するには、次のコマンドを使用します。

PowerShell

```
fsutil devdrv setfiltersallowed Filter-03
```

その他の関連コマンドについては、`fsutil devdrv /?` を参照してください。

一般的なシナリオのフィルター

Dev Drive では、次のフィルターを使用できます。

[🔗 テーブルを展開する](#)

シナリオ: 説明	フィルター名
GVFS: Windows のスパス参加リスト	PrjFlt
MSSense: EDR センサーの Microsoft Defender for Endpoint	MsSecFlt
Defender: Windows Defender フィルター	WdFilter
Docker: 開発者ドライブからコンテナを実行する	bindFlt、wcifs

シナリオ: 説明	フィルター名
Windows パフォーマンス レコーダー: ファイル システムの操作を測定する	ファイル情報
リソース モニター: リソースの使用状況を表示します ディスクアクティビティでファイル名を表示するために必要	ファイル情報
プロセス モニター - Sysinternals: ファイル システムのアクティビティを監視する	ProcMon24
Windows アップグレード: OS のアップグレード中に使用されます。ユーザーが TEMP 環境変数を Dev Drive に移動する場合は必須	WinSetupMon
Windows Defender アプリケーション制御 (WDAC): AppLocker ID サービスを使用した管理インストーラーの追跡	applockerfltr

WdFilter は既定でアタッチされます。次のコマンドは、これらの追加フィルターをすべて Dev Drive にアタッチする方法を示す例です。

PowerShell

```
fsutil devdrv setfiltersallowed "PrjFlt, MsSecFlt, WdFilter, bindFlt, wcifs, FileInfo, ProcMon24"
```

💡 ヒント

特定のシナリオに必要なフィルターを決定するには、Dev Drive を "信頼されていない" ものとして一時的にマークする必要があります。次に、シナリオを実行し、ボリュームにアタッチされているすべてのフィルターをメモします。Dev Drive をもう一度 **信頼済み** として指定し、その Dev Drive の [許可] リストにフィルターを追加して、シナリオが成功したことを確認します。最後に、必要ないフィルターを一度に 1 つずつ削除し、シナリオが想定どおりに動作することを確認します。

💡 ヒント

プロセス モニターのフィルター名は変更される場合があります。フィルター名 "ProcMon24" を追加しても開発ドライブのファイル システム アクティビティが取得されない場合は、コマンド `fltmc filters` を使用してフィルターを一覧表示し、プロセス モニターのフィルター名を見つけて、"ProcMon24" の代わりにその名前を使用します。

ブロックの複製サポート

Windows 11 24H2 & Windows Server 2025 以降、Dev Driveでブロック複製がサポートされるようになりました。Dev Drive は [ReFS](#) ファイル システム形式を利用するため、ブロック複製のサポートは、Dev Drive を使用してファイルをコピーするたびに、無料のパフォーマンス上の利点を意味します。ブロックの複製を使用すると、ファイル システムは、基盤となる物理データに対してコストのかかる読み取りおよび書き込み操作を実行するのではなく、低コストのメタデータ操作としてアプリケーションに代わってファイル バイトの範囲をコピーできます。これにより、複数のファイルが同じ論理クラスターを共有できるようにすることで、**コピーの完了が高速化され、基になるストレージへの I/O が少なくなり、ストレージ容量が向上します。** [ブロックの複製](#)について詳しくは、こちらをご覧ください。

Dev Drive でサポートされていないシナリオ 制限は何ですか？

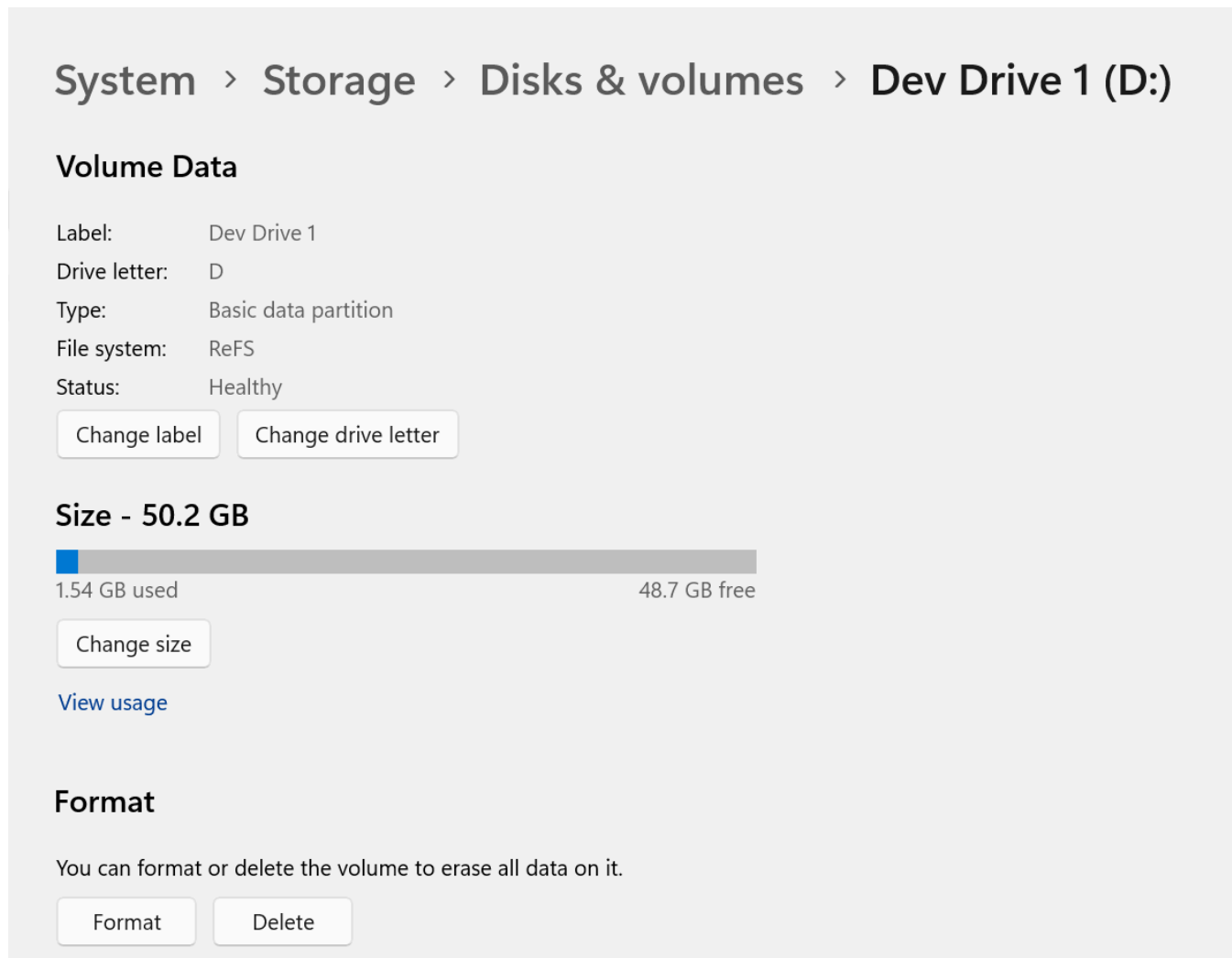
Dev Drive の使用を推奨しないシナリオがいくつかあります。具体的な内容は次のとおりです。

- 既存のストレージ ボリュームを再フォーマットして "Dev Drive" にすると、そのボリュームに格納されているすべてのコンテンツが破棄されます。既存のボリュームを再フォーマットする際に、保存されているコンテンツを保持することはサポートされていません。
- 固定ディスク (HDD または SSD) 上でホストされる仮想ハード ディスク (VHD) を作成する場合VHD をコピーして別のマシンに移動し、それを開発ドライブとして使用し続けることはお勧めしません。
- リムーバブル ディスクまたはホット プラグ可能ディスク (USB、HDD、SSD 外部ドライブなど) に格納されているボリュームでは、Dev Drive としての指定はサポートされていません。
- リムーバブル ディスクまたはホット プラグ可能ディスクによってホストされる VHD 内のボリュームでは、Dev Drive としての指定はサポートされていません。
- コンピューター上の C: ドライブを Dev Drive として指定することはできません。
- Dev Drive の目的は、リポジトリ、パッケージ キャッシュ、作業ディレクトリ、一時フォルダーを格納するように指定されたソフトウェア プロジェクトをビルドおよびデバッグするためのファイルをホストすることです。Dev Drive にアプリケーションをインストールすることはお勧めしません。
- [ダイナミック ディスク](#)での Dev Drive の使用はサポートされていません。代わりに記憶域スペースを使用します。記憶域 [スペース](#) を使用すると、ドライブの障害からデータを保護し、PC にドライブを追加するときに時間の経過と同時に記憶域を拡張できます。

Dev ドライブを削除する方法

Dev ドライブは Windows 11 システム設定 **System > Storage > Disks & volumes** で削除できます。

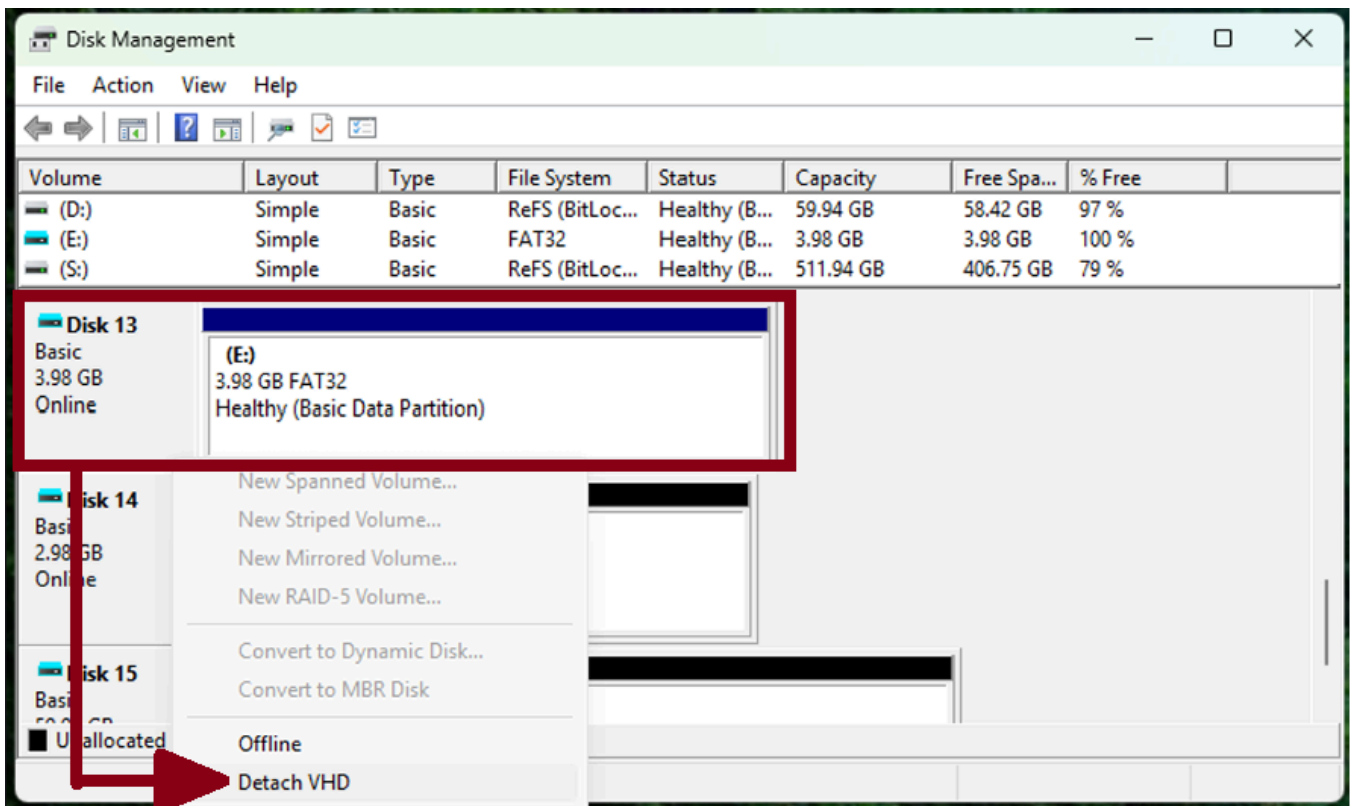
[Windows の設定] メニューを開き、[ストレージ]、[ストレージの詳細設定]、[ディスクとボリューム] の順に選択すると、デバイス上のストレージ ボリュームの一覧が表示されます。削除する Dev Drive ストレージ ボリュームの横にある [プロパティ] を選択します。ドライブのプロパティには、[書式] ラベルの下に [削除] オプションがあります。



これで開発ドライブは削除されます。ただし、開発ドライブが新しい VHD として作成された場合は、その VHD で使用されているストレージ領域を再利用するために VHD を削除する必要があります。これを行うには、仮想ディスクを切断して、Dev Drive をホストしている VHD ファイルを削除できるように次の手順に従う必要:

1. タスクバーの検索ボックスに「コンピューターの管理」と入力して、ディスク管理ツールを開きます。[ストレージ] 見出しの下にある [ディスクの管理] を選択します。開発ドライブのディスク (ボリュームではない) を選択します。Dev Drive をホストしている選択したディスクを右クリックし、結果のメニューから [VHD のデタッチ] を選択します。
2. 仮想ハード ディスクをデタッチすると使用できなくなる旨を伝えるポップアップ ウィンドウが表示されます。

3. デタッチされたら、VHD を削除できます。



Dev Drive に関する FAQ

Dev Drive に関してよく寄せられる質問には、次のようなものがあります。

ニーズに合わせて Dev Drive をカスタマイズするにはどうすればよいですか？

Dev Drive の既定の設定は一般的な開発シナリオ用に最適化されていますが、カスタマイズしてストレージ ボリュームで実行されるドライバーとサービスを制御できます。開発ドライブの設定をカスタマイズするには、[設定] メニューを開きます。[システム]>[ストレージ]>[ディスクとボリューム] で、[プロパティ] に移動します。

① 重要

ビジネスまたはエンタープライズで作業している場合、Dev Drive は引き続きエンタープライズ設定によって管理されます。そのため、会社のポリシーによっては、一部のカスタマイズが使用できない場合があります。

Dev Drive を使用するには、アプリケーションを再インストールする必要がありますか？

いいえ。コンピューターの C: ドライブにインストールされているアプリケーションやツールは、Dev Drive のファイルを利用できます。ただし、開発プロジェクトの場合は、プロジェクト固有のディレクトリ、ファイル、パッケージ キャッシュを Dev Drive 内に格納することをお勧めします。開発ドライブは、リマインダーとしてエクスプローラーの **クイック アクセス** にピン留めできます。

ReFS では NTFS よりも多くのメモリが使用されますか？

はい。ReFS では NTFS よりもわずかに多くのメモリが使用されます。少なくとも 8 GB のメモリ (理想的には 16 GB) のマシンをお勧めします。

マシンに複数の開発者ドライブをインストールできますか？

はい。領域がある場合は、Dev Drive を必要なだけ作成できます。ソフトウェア開発プロジェクトごとに個別の Dev Drive を使用すると、開発終了時にドライブを削除するだけで済みます。ディスクをもう一度パーティション分割する必要はありません。ただし、開発ドライブの最小サイズは 50 GB であることに注意してください。

Visual Studio で Dev Drive を使用する場合に確認しておくべきことはありますか？

Dev Drive が作成されると、新しいプロジェクトの作成時、または既存のプロジェクトを複製するときに、Visual Studio によって自動的に Dev Drive が認識され、既定でそのファイルパスが選択されます。Visual Studio を使用する場合はパフォーマンスを最適化するには、プロジェクトコード、[パッケージ キャッシュ](#)、MS ビルド タスク `Copy on write` を、以前に別の場所に保存されている可能性がある開発ドライブに移動することをお勧めします。(Visual Studio ドキュメント [でビルド出力ディレクトリを変更する方法](#) を参照してください)。また、`%TEMP%` と `%TMP%` envvars を Dev Drive にリダイレクトすることを検討することをお勧めします。これを行うには、Windows Update プロセスに必要な `WinSetupMon` フィルターも追加する必要があります。([一般的なシナリオについては、フィルターを参照してください](#)。これらは多くのプログラムで使用されているため、生じ得る副作用に注意してください。また、Dev Drive を使用することで非同期のパフォーマンス向上を図るために、[Microsoft Defender のパフォーマンス モード](#) を使用することをお勧めします。Microsoft Defender を完全にオフにすると、パフォーマンスが最大限に高まる可能性がありますが、これによりセキュリティリスクが増大する可能性があります。この設定はシステム管理者が制御します。

詳細については、ブログ投稿「[Dev Drive for Performance Improvements in Visual Studio and Dev Boxes](#)」を参照してください。

Dev Drive は WSL プロジェクト ファイルで動作しますか？

WSL 経由で実行されている Linux ディストリビューションから、Windows ファイル システムで実行される Dev Drive プロジェクト ファイルにアクセスできます。ただし、WSL は VHD で実行されます。最適なパフォーマンスを得るには、ファイルは Linux ファイル システムに格納する必要があります。WSL は Windows ファイル システムの範囲外であるため、WSL 経由で実行されている Linux ディストリビューションから Dev Drive のプロジェクト ファイルにアクセスするときにパフォーマンスが向上することはありません。

Windows ストレージ ボリュームのフォーマットにはどのような方法が使用されますか？

Windows ドライバーのドキュメント「[MSFT_Volume class](#)」を参照してください。

開発者ドライブで Live Unit Testing を構成して使用する方法

[Live Unit Testing を構成して使用する方法](#)については、Visual Studio のドキュメントを参照してください。ただし、[ProjFS](#) には依存関係があることに注意してください。Live Unit Testing ワークスペースのルートを開発ドライブに移動し、許可されたフィルター リストに Windows Projected File System を追加する必要があります。これを行うには、PowerShell で次のコマンドを使用します:

PowerShell

```
fsutil devdrv setfiltersallowed PrjFlt
```

開発ドライブとして使用するために作成された VHD は、それを格納するドライブで BitLocker が有効になっている場合、暗号化されますか？

はい。開発ドライブの VHD は、ホスティング ボリュームの BitLocker 暗号化に含まれます。取り付けた VHD で BitLocker を有効にする必要はありません。

Dev Drive を使用すると、Windows での Java 開発を高速化できますか？

はい、Dev Drive を使用すると、効率を高め、Java 開発プロジェクトで作業するときのビルド時間を短縮できます。ブログ記事「[開発ドライブを使用して Windows での Java 開発を高速化する](#)」[🔗](#)を参照してください。

Dev Drive パフォーマンス モードは、Microsoft Defender 以外のウイルス対策プログラムに適用できますか？

[Dev Drive パフォーマンス モード](#) は、具体的には、Defender のリアルタイム保護に関連する Microsoft Defender ウイルス対策機能です。開発ドライブで代替ウイルス対策プログラムを使用する場合、パフォーマンス モードは適用されませんが、開発作業のパフォーマンスとセキュリティの適切なバランスを見つけるために、開発ドライブにアタッチされているセキュリティ [フィルターの許可リストを調整](#) することができます。添付フィルター リストに変更を加えるときは、添付フィルターの機能を理解している必要があります。 [一般的なシナリオのフィルター](#) の説明を含む一覧を検索します。

作成して追跡しなくなった Dev Drive をを見つけるにはどうすればよいですか？

Dev Drive が取り付けられているが、その場所を忘れた場合は、次のメソッドを使用して見つけることができます。

- [DiskPart](#) と "[list vdisk](#)" コマンドを使用して、vhdx への完全なパスを表示します。1) コマンド ラインを開き、「`diskpart`」と入力します。2) DiskPart が開いたら、「`list vdisk`」と入力します。
- PowerShell と "[Get-Disk | Select-Object FriendlyName,Location](#)" を使用します。PowerShell を開き、「`Get-Disk | Select-Object FriendlyName,Location`」と入力します。

こうしたドキュメントや FAQ に投稿するにはどうしたらよいですか？

このドキュメントに問題がある場合、または追加の FAQ 候補を投稿したい場合は、[GitHub の Windows Dev Docs オープン ソース リポジトリ](#)を [参照](#)してください。

エンタープライズ ビジネス デバイスの Dev Drive セキュリティ ポリシーを構成する方法

エンタープライズ レベルの管理者は、多くの場合、組織内のさまざまな Windows デバイスのセキュリティを管理する責任があります。新しい Windows リリースで使用可能になったときに新機能を有効にするかどうかを制御するポリシーを構成する方法は複数あります。このガイドでは、Windows 11 [Dev Drive](#) ストレージ ボリュームの機能に関する重要な情報と、組織のグループ ポリシーを構成して、開発者がセキュリティ システム フィルターのアタッチに対するセキュリティと制御を維持しながら、このパフォーマンス最適化されたストレージ形式を使用できるようにする方法について説明します。

グループ ポリシーを有効にする方法に関するガイダンスは、任意のポリシー管理ツールを使用して以下にあります。

- [Microsoft Intune](#)、
- [Microsoft Configuration Manager](#) (ConfigMgr、以前の MEMCM/SCCM)、または
- [Windows 11 ローカル グループ ポリシー エディター](#)。

前提条件

- Windows 11、ビルド #10.0.22621.2338 以降 ([Windows 更新プログラムの確認](#))
- 16 GB のメモリを推奨 (最小 8 GB)
- 50 GB の最小空きディスク領域
- 開発者ドライブ はすべての Windows SKU バージョンで使用可能

一時的なエンタープライズ機能コントロールによって開発ドライブが無効になる

毎月の累積的な更新プログラムを通じて新機能と拡張機能が導入され、Windows 11に継続的なイノベーションが提供されます。組織に計画と準備の時間を与えるために、これらの新機能の一部は、[Windows 11の一時的なエンタープライズ機能コントロール](#)を使用して既定で一時的にオフになっています。

開発者ドライブ は、ポリシーによって管理されている Windows **更新プログラムを持つデバイス**に対して自動的に無効になります。開発ドライブを作成する機能を無効にするのは、セキュリティ管理者が新しいポリシーの更新を決定してロールアウトする時間を確保するための一時的なものです。これらのポリシーの更新を決定して構成するためのガイダンスの概要を以下に示します。

Dev Drive ストレージの有効化とウイルス対策フィルターのセキュリティに関するグループ ポリシーを決定する

グループ ポリシーは、エンタープライズ管理者が職場のデバイスの設定を管理し、ユーザー アカウント (ローカル管理者) がビジネス環境で行うことができる設定の変更をある程度制御できるようにする Windows の機能です。

既定では、Microsoft Defender とサード パーティのウイルス対策フィルターの両方を含むウイルス対策フィルターが Dev Drive に追加されます。Dev Drive ストレージ ボリュームの既定の設定では、ローカル デバイス管理者がアタッチするフィルターを制御することもできます。つまり、ローカル デバイス管理者は、既定のウイルス対策フィルターを削除するようにシステムを構成して、開発ドライブにウイルス対策フィルターがアタッチされないようにすることができます。これが問題になる場合は、開発ドライブが有効になっているときにウイルス対策フィルターがアタッチされたままになるようにグループ ポリシーを構成できます。さらに、許可されたファイル・システム・フィルタ・リストを定義できます。

グループ ポリシーを更新して開発ドライブを有効にする

[Dev Drive を有効にする] ポリシー設定には、次のものが含まれます。

- **未構成:** 既定では、グループ ポリシーでエンタープライズ管理者によって有効にされるまで、一時的なエンタープライズ機能の制御ポリシーでは [開発者ドライブ ストレージ ボリューム] オプションがオフになります。
- **有効:** 有効にすると、[Dev Drive](#) ストレージ ボリュームを作成するオプションがオンになります。
- **オプション - ウイルス対策フィルターで開発ドライブを保護させる:** 開発ドライブは、開発者シナリオのパフォーマンスに合わせて最適化されており、**ローカル**管理者 (ユーザー アカウント) は、アタッチするファイル システム フィルターを選択できます。これにより、ローカル管理者は、"ウイルス対策フィルターで開発ドライブを保護させる" オプションがオンになっていない限り、既定のウイルス対策機能をデタッチすることもできます。このオプションをオンにすると、既定のウイルス対策フィルターが強制的にアタッチされたままになります。
- **無効:** この設定を無効にすると、Dev Drive ストレージ ボリュームを作成して使用する機能がオフになります。

Dev Drive フィルターのアタッチ ポリシーを更新する

さらに、開発ドライブにアタッチできるフィルターをエンタープライズ管理者が制御できる Dev Drive **フィルター アタッチ ポリシー**設定があります。設定は次のとおりです。

- **[未構成]**: 既定では、Dev Drive はパフォーマンスのために最適化されており、Microsoft Defender と 3rdパーティのウイルス対策フィルターがアタッチされていますが、他のファイル システム フィルターは適用されていません。この既定の設定では、ローカル管理者は、既定のウイルス対策フィルターを含むフィルターをアタッチまたはデタッチできます。上記の [開発ドライブを有効にする] ポリシーでオプションの [ウイルス対策フィルターで開発ドライブを保護させる] をオンにすると、それ以上のフィルター ポリシーが定義されていない場合でも、ウイルス対策フィルターが強制的にアタッチされたままになります。
- **有効**: ローカル管理者 (ユーザー アカウント) は、フィルターをアタッチまたはデタッチできます。フィルター一覧を追加すると、エンタープライズ管理者 (グループ ポリシー ドメイン レベル) は、アタッチできるフィルタを定義できます。フィルター リストを含めない場合は、任意のフィルターをアタッチできます。
- **無効**: ローカル管理者 (ユーザー アカウント) は、フィルターをアタッチまたはデタッチできません。

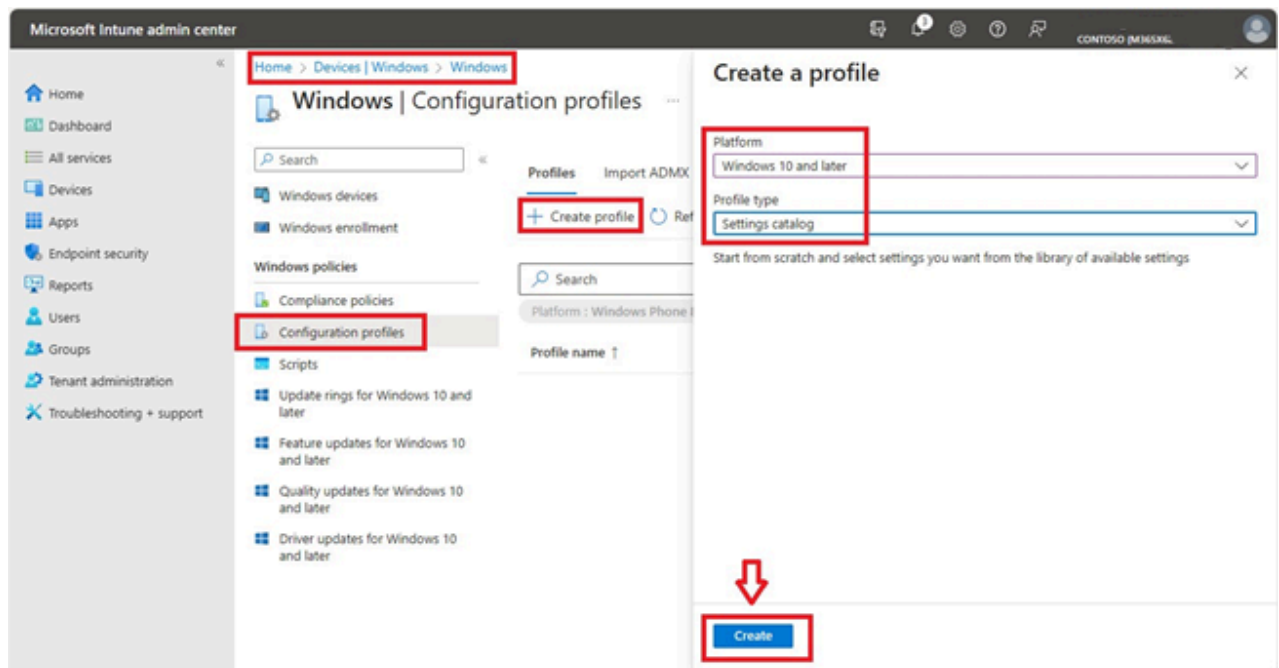
Dev Drive 機能を有効にし、グループ ポリシーを更新するには、いくつかの方法があります。

- [Microsoft Intune を使用してグループ ポリシーを更新する](#)
- [Microsoft Configuration Managerを使用してグループ ポリシーを更新する](#)
- [Windows11ローカルグループポリシーエディターを使用してグループポリシーを更新する](#)

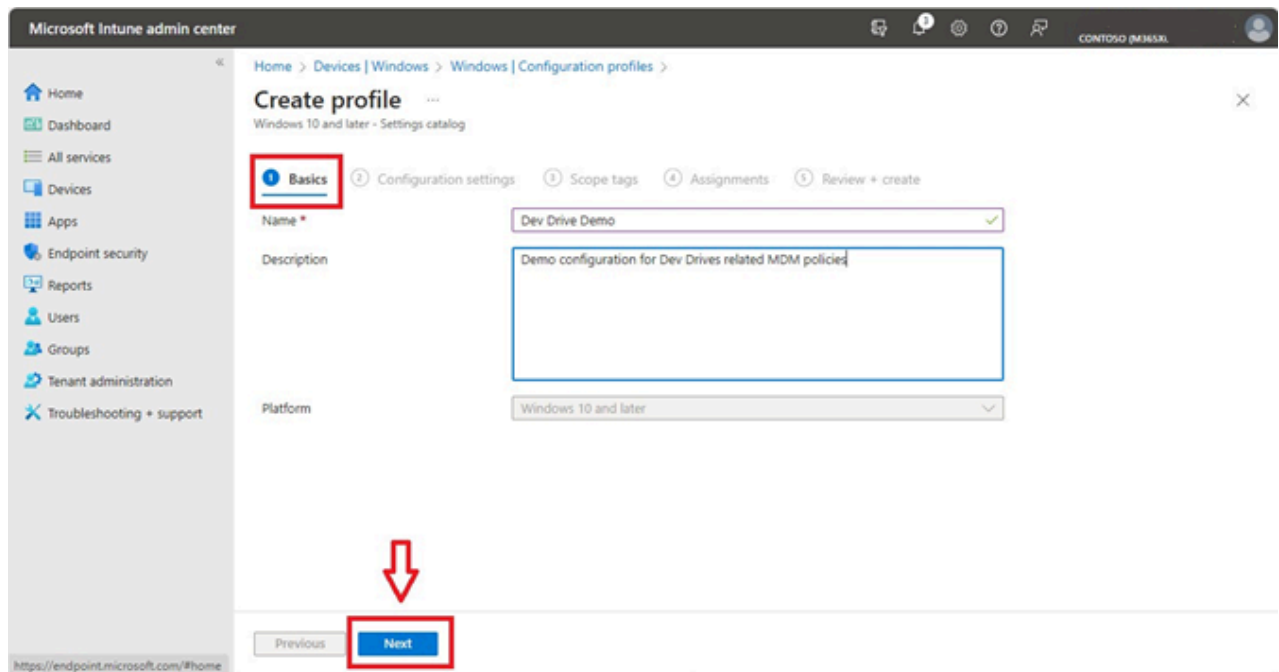
Microsoft Intuneを使用してDev Driveのグループ ポリシーを更新する

グループ ポリシーを更新し、[Microsoft Intune](#) を使用して [Dev Drive](#) を有効にするには:

1. Intune ポータル (<https://endpoint.microsoft.com>) を開き、資格情報を使用してログインします。
2. プロファイルを作成する:
 - a. [デバイス] > [Windows] > [構成プロファイル] > [プロファイルの作成]
 - b. プラットフォーム > の選択: Windows 10 以降
 - c. プロファイルの種類 > 設定カタログの選択

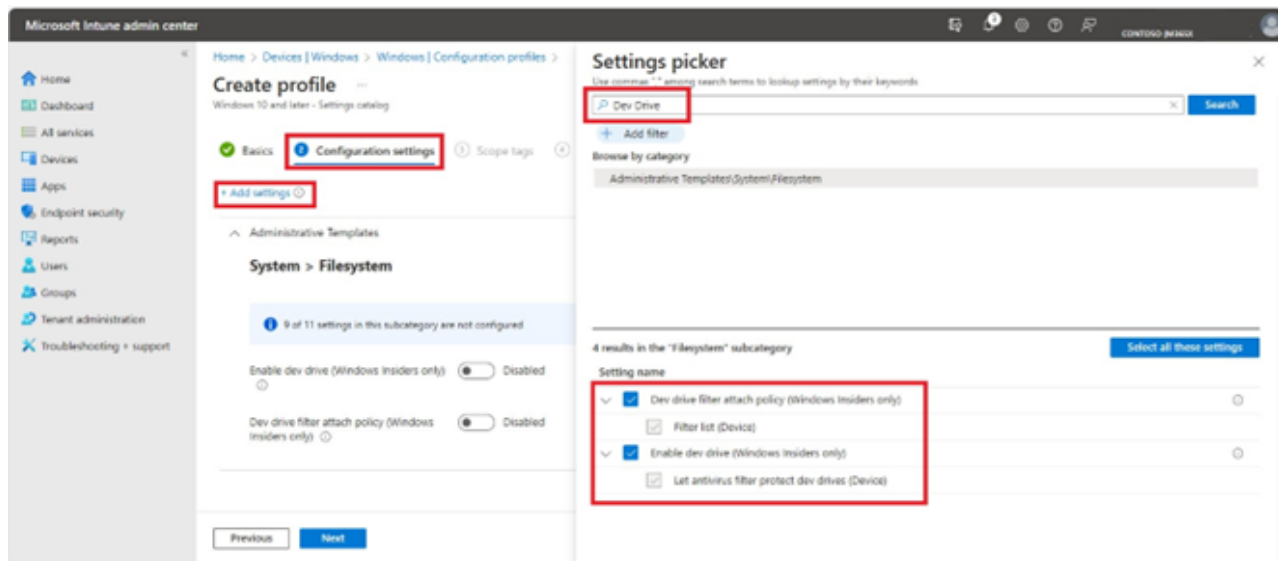


3. カスタムプロファイルの名前と説明を設定します。

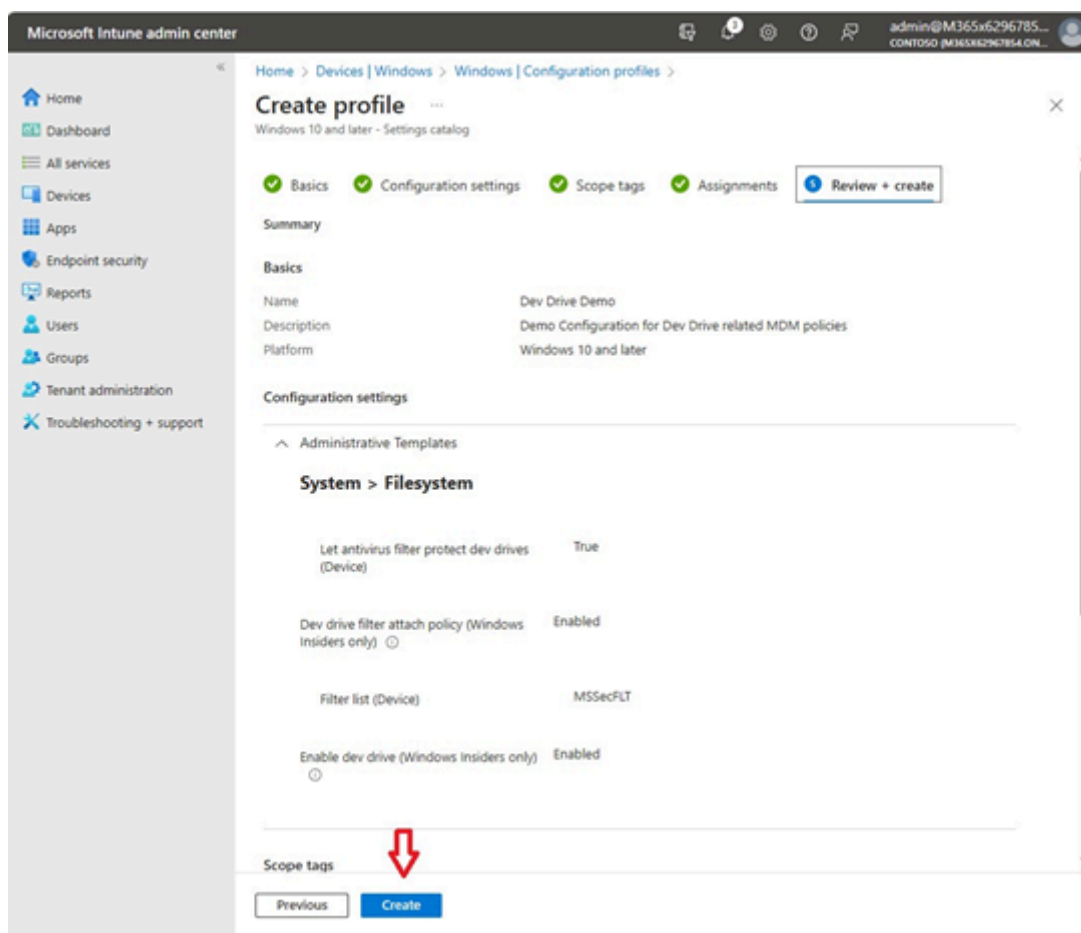


4. Dev Drive 関連の設定を構成します。

- 設定ピッカーで "Dev Drive" を検索するか、"管理用テンプレート\System\Filesystem" に移動します
- [Dev Drive 関連のポリシー] を選択します: [Dev Drive を有効にする] と [ウイルス対策フィルターで Dev Drives を保護させる]、[Dev Drive フィルターのアタッチ ポリシー] と [フィルター リスト] を選択します



5. Dev Drive ポリシー設定を構成し、スコープ タグと割り当ての残りの構成を完了し、**[作成]** を選択します

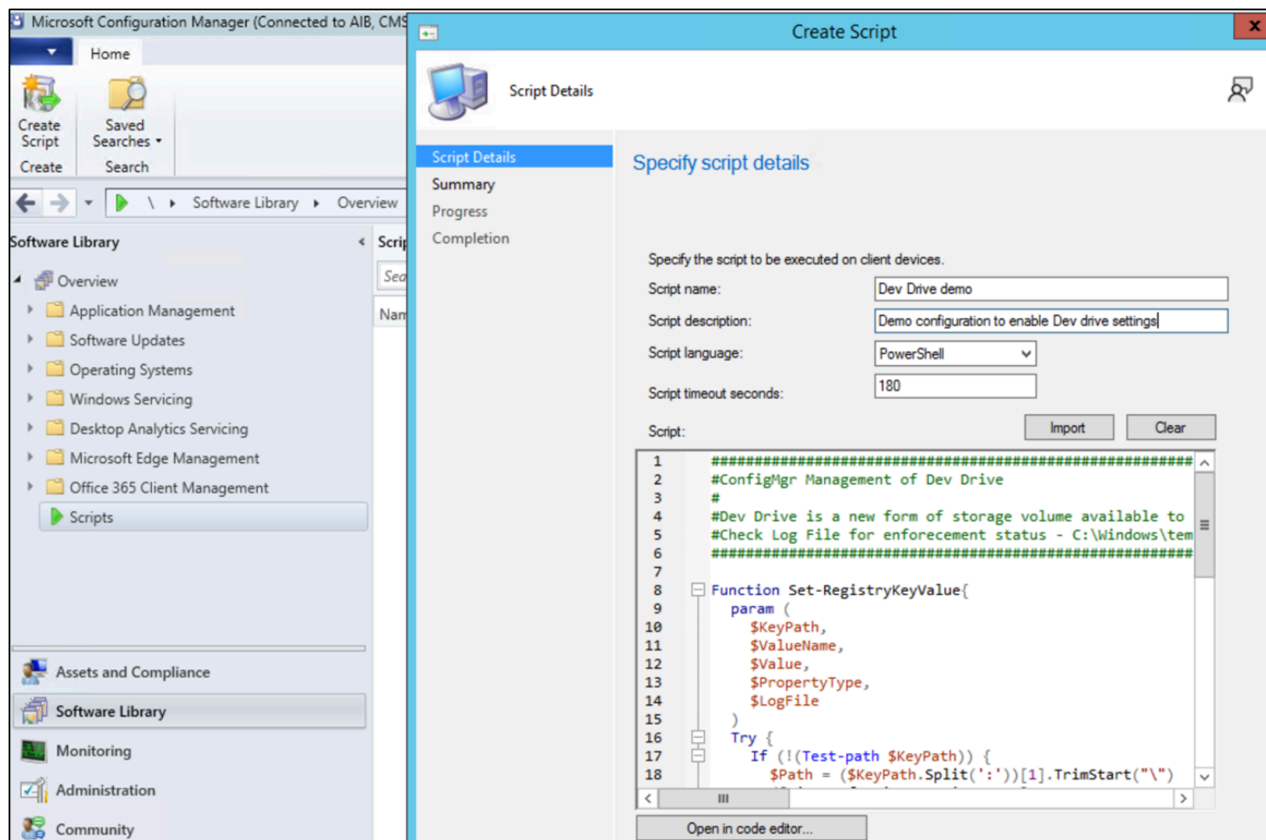


Microsoft Configuration Managerを使用してDev Driveのグループ ポリシーを更新する

Microsoft Configuration Manager (ConfigMgr、旧称 MEMCM/SCCM) を使用してグループ ポリシーを更新し、Dev Drive を有効にするには、次の PowerShell スクリプトを使用できます。
(Configuration Manager とは?)

Configuration Manager コンソールには、PowerShell スクリプトを実行して、ネットワーク内のすべてのコンピューターのグループ ポリシー設定を更新する機能が統合されています。

1. Microsoft Endpoint Configuration Manager コンソールを開きます。「ソフトウェア・ライブラリ>」の「スクリプト>」の「スクリプトの作成」を選択します。



2. スクリプト名 (例: Dev Drive demo)、説明 (Dev Drive 設定を有効にするためのデモ構成)、言語 (PowerShell)、タイムアウト秒 (180) を入力し、次の "Dev Drive demo" スクリプトの例を貼り付けてテンプレートとして使用します。

PowerShell

```
#####
#ConfigMgr Management of Dev Drive
#Dev Drive is a new form of storage volume available to improve performance for
key developer workloads.
#Check Log File for enforcement status - C:\Windows\temp\ConfigDevDrive-
<TimeStamp>.log
#####

Function Set-RegistryKeyValue{
param (
$KeyPath,
$ValueName,
$Value,
$PropertyType,
$LogFile
)
Try {
```

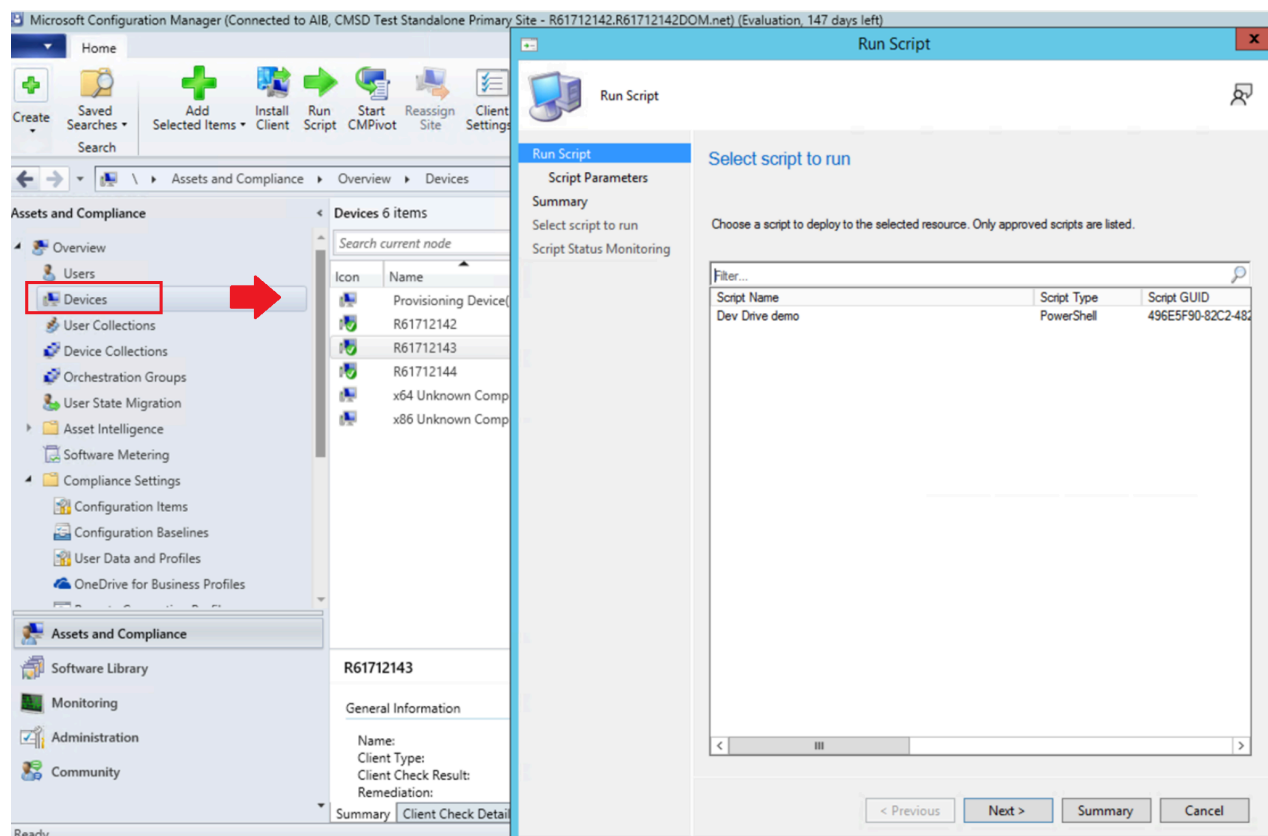
```

If (!(Test-path $KeyPath)) {
    $Path = ($KeyPath.Split(':'))[1].TrimStart("\")

    ([Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey([Microsoft.Win32.RegistryHive]::LocalMachine,$env:COMPUTERNAME)).CreateSubKey($Path)
        New-ItemProperty -path $KeyPath -name $ValueName -value $Value -
PropertyType $PropertyType -Force | Out-Null
    }
    Else {
        New-ItemProperty -path $KeyPath -name $ValueName -value $Value -
PropertyType $PropertyType -Force | Out-Null
    }
    $TestValue = (Get-ItemProperty -Path $KeyPath)."$ValueName"
    If ($TestValue -eq $Value){ Add-Content -Path $LogFile -Value
"$KeyPath,$ValueName,$Value,$PropertyType,$TestValue,Success" }
    Else { Add-Content -Path $LogFile -Value
"$KeyPath,$ValueName,$Value,$PropertyType,$TestValue,Failure" }
    }
    Catch {
        $ExceptionMessage = $($PSItem.ToString()) -replace
[Environment]::NewLine,"";
        Add-Content -Path $LogFile -Value
"$KeyPath,$ValueName,$Value,$PropertyType,$TestValue,Failure -
$ExceptionMessage"
    }
}
$ExecutionTime = Get-Date
$StartTime = Get-Date $ExecutionTime -Format yyyyMMdd-HH:mm:ss
$LogFile = "C:\Windows\temp\ConfigDevDrive-$StartTime.log"
Add-Content -Path $LogFile -Value "-----V 1.0
$ExecutionTime - Execution Starts -----"
Add-Content -Path $LogFile -Value
"RegistryKeyPath,ValueName,ExpectedValue,PropertyType,CurrentValue,ComparisonRe
sult"
#Set up a Dev Drive
Set-RegistryKeyValue -KeyPath "HKLM:\System\CurrentControlSet\Policies\" -
ValueName "FsEnableDevDrive" -Value "1" -PropertyType "Dword" -LogFile $LogFile
Set-RegistryKeyValue -KeyPath "HKLM:\System\CurrentControlSet\Policies\" -
ValueName "FltMgrDevDriveAllowAntivirusFilter" -Value "1" -PropertyType "Dword"
-LogFile $LogFile
Set-RegistryKeyValue -KeyPath "HKLM:\System\CurrentControlSet\Policies\" -
ValueName "FltMgrDevDriveAttachPolicy" -Value "PrjFlt, MsSecFlt, WdFilter,
bindFlt, wcifs, FileInfo" -PropertyType "MultiString" -LogFile $LogFile
$ExecutionTime = Get-Date
Add-Content -Path $LogFile -Value "-----
$ExecutionTime - Execution Ends -----
-----

```

3. 新しいスクリプトを追加するときは、スクリプトを選択して承認する必要があります。
承認状態が「承認待ち」から「承認済み」に変わります。
4. 承認されたら、1つのデバイスまたはデバイス コレクションを右クリックし、**[スクリプトの実行]** を選択します。



5. スクリプトの実行ウィザードのスク립ト ページで、一覧からスクリプトを選択します (この例では *Dev Drive demo*)。承認されたスクリプトのみが表示されます。[次へ] を選択し、ウィザードを完了します。

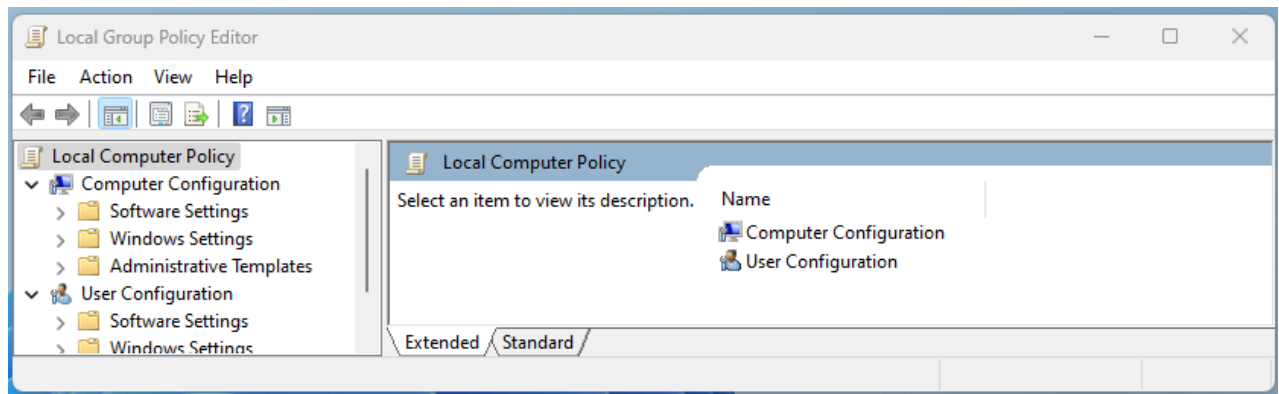
「[FsUtil を使用したポリシーのクエリ](#)」を参照して、グループ ポリシー設定が正確に更新されたことを確認します。

詳細については、「[Configuration Manager コンソールから PowerShell スクリプトを作成して実行する](#)」を参照してください。

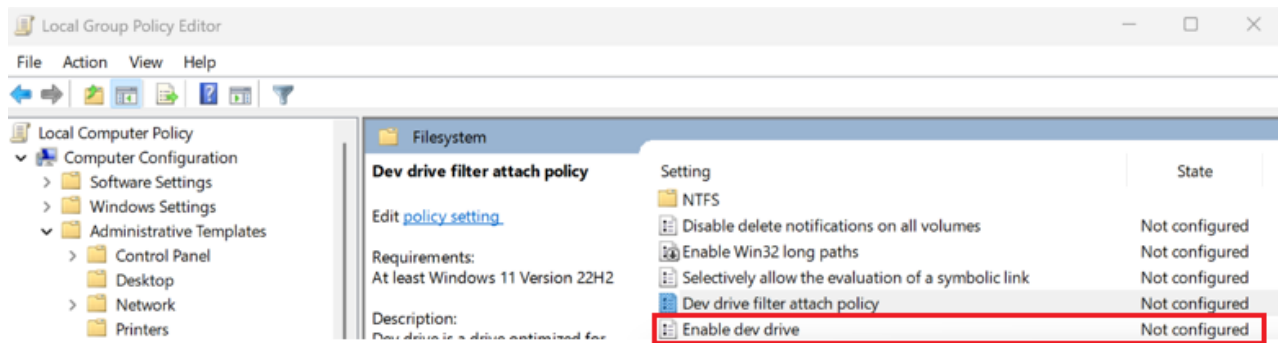
Windows 11 ローカル グループ ポリシー エディターを使用して、Dev Drive のグループ ポリシーを更新する

グループ ポリシーを更新し、Windows 11 ローカル グループ ポリシー エディターを使用して [Dev Drive](#) を有効にするには:

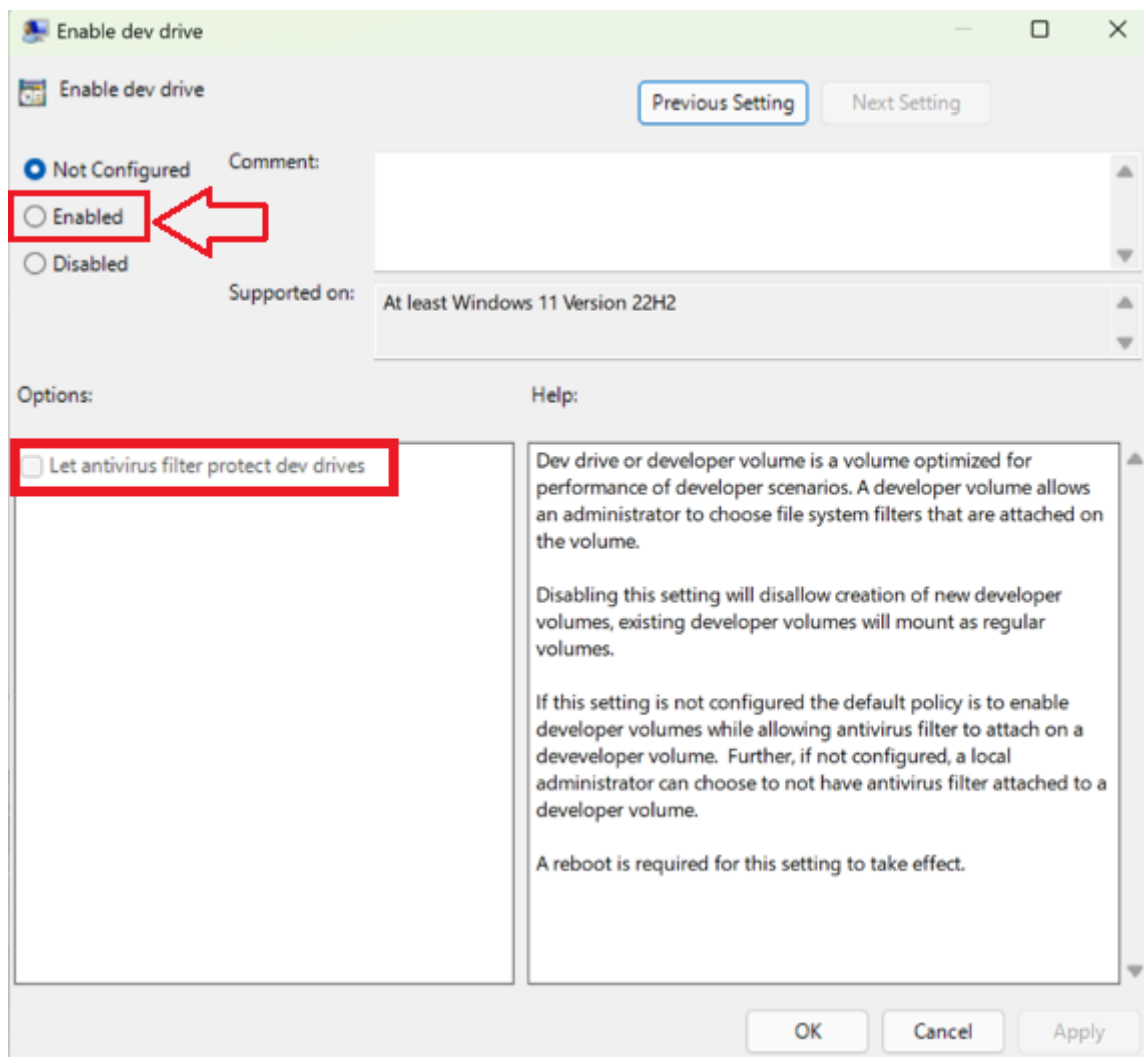
1. Windowsのコントロールパネルでローカルグループポリシーエディターを開きます。



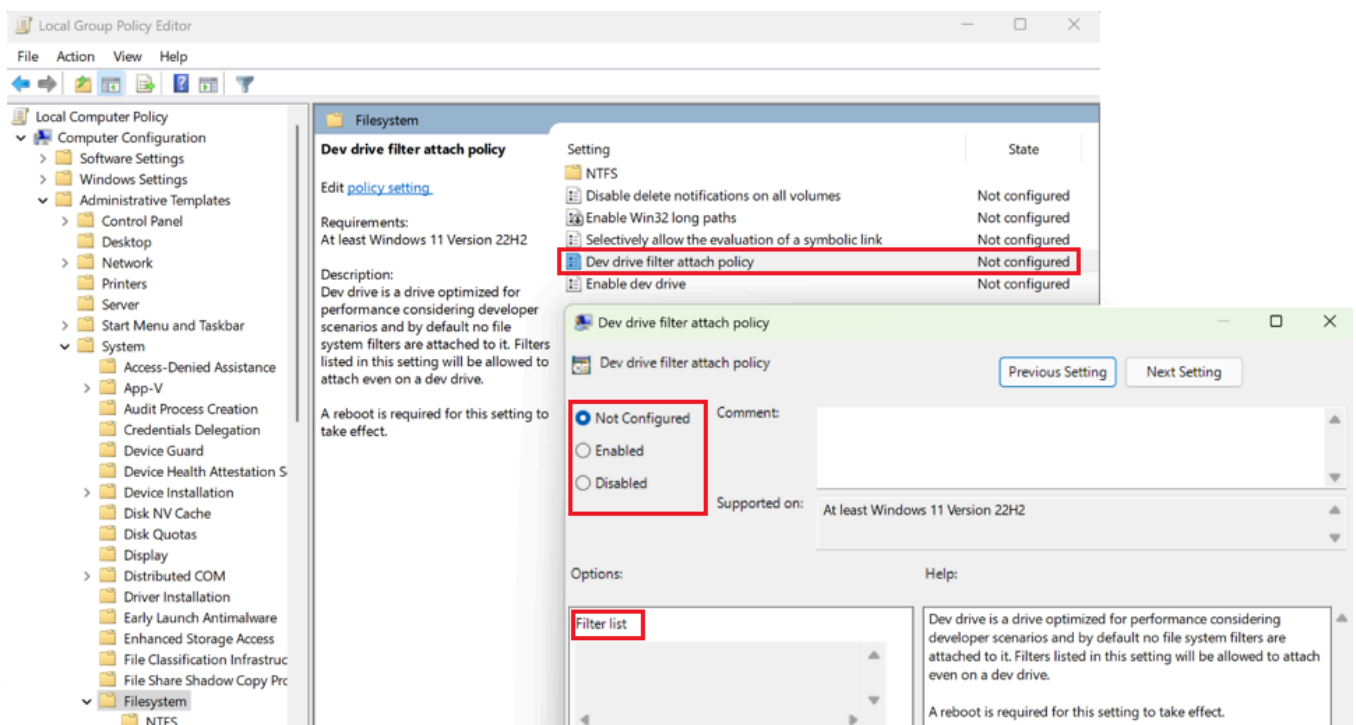
2. [コンピューターの構成] で [管理用テンプレート>システム > ファイルシステム] を選択し、[設定] の一覧で [開発ドライブを有効にする] を選択します。



3. [有効] を選択して、グループ ポリシーで Dev Drive を有効にします。



このフィルター アタッチ ポリシーを更新するには、Windows の [コントロールパネル] の [ローカル グループ ポリシー エディター] から [Dev Drive フィルター アタッチ ポリシー] を選択します。



FsUtil を使ってポリシーをクエリする

FsUtil を使用して、開発ドライブ用に構成されたグループ ポリシーに対してクエリを実行できます。次に、特定の設定で構成された Dev Drive グループ ポリシーの FsUtil クエリの出力を示します。

- Dev Drive を有効にする
- Microsoft Defender ウイルス対策フィルターで開発ドライブを保護する (WdFilter)
- FileInfo ミニフィルターが、許可されたフィルターとして [フィルター リスト] に追加されました

FsUtil コマンドを入力します。

PowerShell

```
fsutil devdrv query
```

結果:

PowerShell

```
Developer volumes are enabled.  
Developer volumes are protected by antivirus filter, by group policy.  
Filters allowed on any developer volume, by group policy:  
    MsSecFlt  
Filters allowed on any developer volume:  
    FileInfo
```

この同じクエリを特定の開発ドライブで実行して、アタッチされたフィルターを確認できます。特定の開発ドライブでコマンドを実行するには、次のコマンドを入力します。

PowerShell

```
fsutil devdrv query d:
```

結果:

PowerShell

```
This is a trusted developer volume.  
Developer volumes are protected by antivirus filter, by group policy.  
Filters allowed on any developer volume, by group policy:  
    MsSecFlt  
Filters allowed on any developer volume:  
    FileInfo
```

Filters currently attached to this developer volume:
MsSecFlt, WdFilter, FileInfo

その他のリソース

- [Windows 11 で継続的なイノベーションを提供する \(Microsoft サポート\)](#) 
- [Windows 11 の一時的なエンタープライズ機能制御](#)
- [追加の Windows Update 設定を管理する \(Windows 展開\)](#)

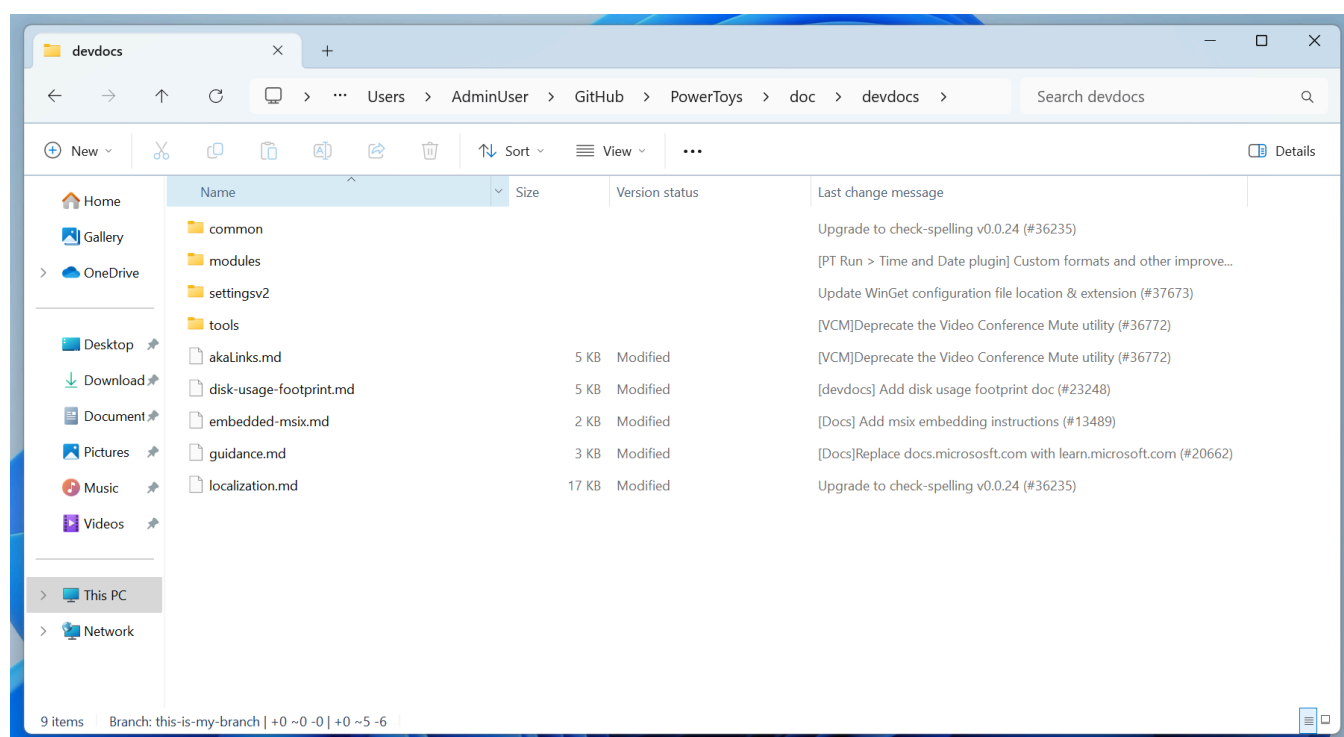
Last updated on 2025/04/06

エクスプローラーのバージョン管理の統合

エクスプローラーのバージョン管理統合 では、エクスプローラーでバージョン管理情報が直接提供されます。これには、ブランチ名、最終コミット作成者、最終コミット メッセージなどの情報が含まれます。

❗ Note

現時点では、エクスプローラーのバージョン管理統合では Git のみがサポートされています。[詳細設定] システム コンポーネントは拡張可能であり、追加のバージョン コントロールの種類を許可します。



❗ 重要

組織のポリシーでは、詳細設定コントロールを無効または非表示にすることができません。トグルが使用できない場合は、管理者に問い合わせてください。[グループ ポリシー](#) の関連ガイダンスを参照してください。

バージョン管理インジケータを有効にする

1. [設定] を開き、[システム > 詳細設定] に移動します。
2. [エクスプローラー + バージョン管理] で、[フォルダーの追加] を選択し、エクスプローラーで認識するリポジトリを選択します。

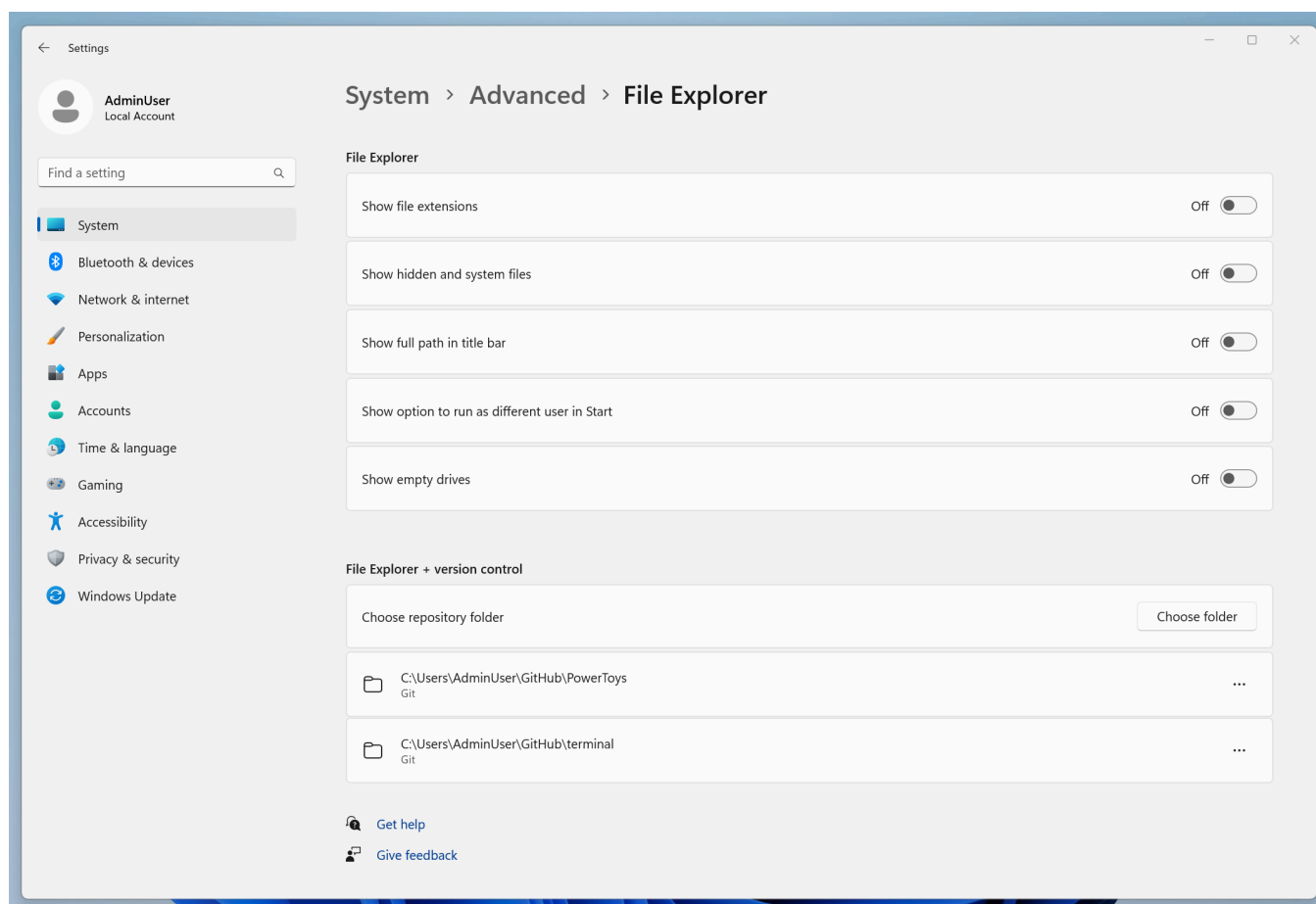
3. エクスプローラーに戻り、選択したフォルダーのいずれかを開いてリポジトリの詳細を表示します。

💡 ヒント

インジケータがすぐに表示されない場合は、エクスプローラーを閉じてから再度開きます。

リポジトリを識別する方法

エクスプローラーでバージョン管理情報を表示できるように、Windows はソース コード リポジトリであるフォルダーを認識する必要があります。[エクスプローラー + バージョン管理] ヘッダーの下にある [Windows 詳細設定] > エクスプローラーの設定で、リポジトリ フォルダーを選択できます。



"ファイル エクスプローラー"に表示される内容

エクスプローラーで選択したリポジトリ フォルダを開くと、次のようなバージョン コントロールの詳細がエクスプローラー UI に表示されます。

- ブランチ名

- 最後のコミット作成者
- 最後のコミット メッセージ
- コミット タイムスタンプ

制限事項

- Git リポジトリがサポートされています。他のシステムでは、将来の拡張機能が必要になる場合があります。
- 大量のコンテンツが生成された非常に大きなリポジトリまたはフォルダーは、インジケータを遅延させる可能性があります。
- 入れ子になったリポジトリとサブモジュールでは、詳細が一貫して表示されない場合があります。
- ネットワーク共有、シンボリックリンク されたパス、または WSL がマウントされた場所では、メタデータが表示されない場合があります。
- 大文字と小文字が区別されるファイルシステムと一般的でないファイル属性は、検出に影響を与える可能性があります。

❗ Note

パフォーマンスを向上させるには、リポジトリを選択するときに、生成された大きなフォルダー (`node_modules`、ビルド出力など) を除外することを検討してください。

トラブルシューティング

- インジケータが表示されない: フォルダーに `.git` ディレクトリが含まれていることを確認します。[詳細設定] の [エクスプローラーとバージョン管理] で選択されています。エクスプローラーを再起動します。
- 切り替えが無効化されています: デバイスはポリシーによって管理されている可能性があります。 [グループ ポリシー](#) を参照し、管理者に問い合わせてください。
- 低速または不整合な更新: 非常に大きなフォルダーを選択から減らすか、リポジトリをローカル ディスク パスに移動します。
- 別の SSH サービスとの競合: 開発者モードの SSH 機能を使用している場合は、ポートとサービスがリポジトリへのアクセスを妨げないようにします。

拡張性とフィードバック

Windows 詳細設定システム コンポーネントはオープン ソースであり、追加のバージョン管理プロバイダーをサポートするように設計されています。

- 詳細を確認するか、[GitHub](#) で機能を要求します。
 - 詳細設定に関するより広範なコンテキストについては、[Windows の詳細設定](#) のランディング ページを参照してください。
-

Last updated on 2026/01/23

[最新の実行] ダイアログ

最新の [実行] ダイアログでは、従来の Win + R エクスペリエンスが Fluent Design 上に構築された再設計されたインターフェイスに置き換えられます。ダークモードをサポートし、クラシックダイアログよりも高速に起動し、ホームディレクトリのクイックナビゲーションなどの新機能を追加します。

最新の実行ダイアログは WinUI 3 と C# でビルドされ、ネイティブパフォーマンスのために .NET AOT でコンパイルされます。PowerToys のコマンドパレット (CmdPal) とコードベースを共有します。

❗ 重要

組織のポリシーでは、詳細設定コントロールを無効または非表示にすることができます。トグルが使用できない場合は、管理者に問い合わせてください。 [グループポリシー](#) の関連ガイダンスを参照してください。

最新の [実行] ダイアログを有効にする

1. **[設定]** を開き、**[System > Advanced]** に移動します。
2. 画面の上部にある **[実行] ダイアログ** をオンに切り替えます。
3. **Win + R キー** を押して、新しい実行エクスペリエンスを開きます。

クラシックダイアログに戻すには、同じ設定ページに戻り、オプションをオフに切り替えます。

Features

最新の [実行] ダイアログでは、クラシックダイアログで実行されるすべての機能 (コマンドの実行、パスの開き方 (ローカルとネットワーク)、およびコマンド履歴の参照など) がサポートされます。

- **Fluent Design と dark モード:** インターフェイスは、Windows 11 のビジュアルスタイルと一致し、システムテーマを尊重します。
- **ホームディレクトリへのクイックアクセス:** `~\` 入力してユーザーディレクトリに直接移動し、コマンドラインから行うのと同じようにサブディレクトリ内を移動し続けます。

- **起動時間の短縮:** モダン ダイアログの表示時間の中央値は 94 ミリ秒で、クラシック ダイアログ ([ソース](#)) の場合は 103 ミリ秒です。
- **コマンド履歴:** 上方向キーと下方向キーを押して、クラシック ダイアログと同様に、以前に実行したコマンドを順番に切り替えます。

Requirements

最新の [実行] ダイアログには、詳細設定システム コンポーネントを使用した Windows 11 が必要です。現在、試験段階のチャンネル上の Windows Insider ビルドで利用できます。

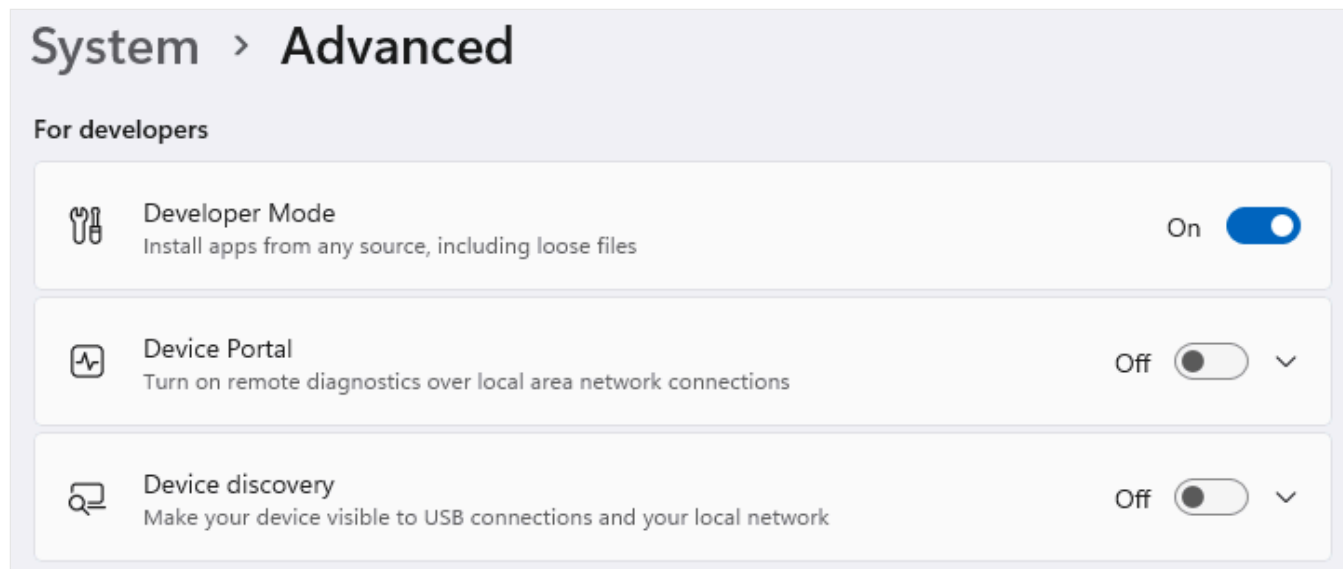
関連するコンテンツ

- [詳細設定の概要](#)
- [エクスプローラーのバージョン管理の統合](#)
- [新しい \[実行\] ダイアログ: 高速、クリーン、より高い機能 \(DevBlog\)](#)

Last updated on 2026/07/03

開発者向けの設定

Windows設定の [System > Advanced](#) ページには、開発者モードと開発者モードが有効になっているときに使用できる追加機能が含まれています。開発者モードでは、Windowsでのアプリのビルド、展開、テスト用に設計されたツール、設定、および機能のロックが解除されます。



① Note

Windows 11 25H2 より前のバージョンでは、これらの設定はWindows設定の **For developers** ページに表示されます。Windows 11 25H2 以降では、[詳細設定](#) ページの **開発者向け** セクションに表示されます。

開発者モードを有効にする

Visual Studioを使用してソフトウェアを記述している場合は、開発用 PC とコードのテストに使用するデバイスの両方で開発者モードを有効にする必要<>必要があります。

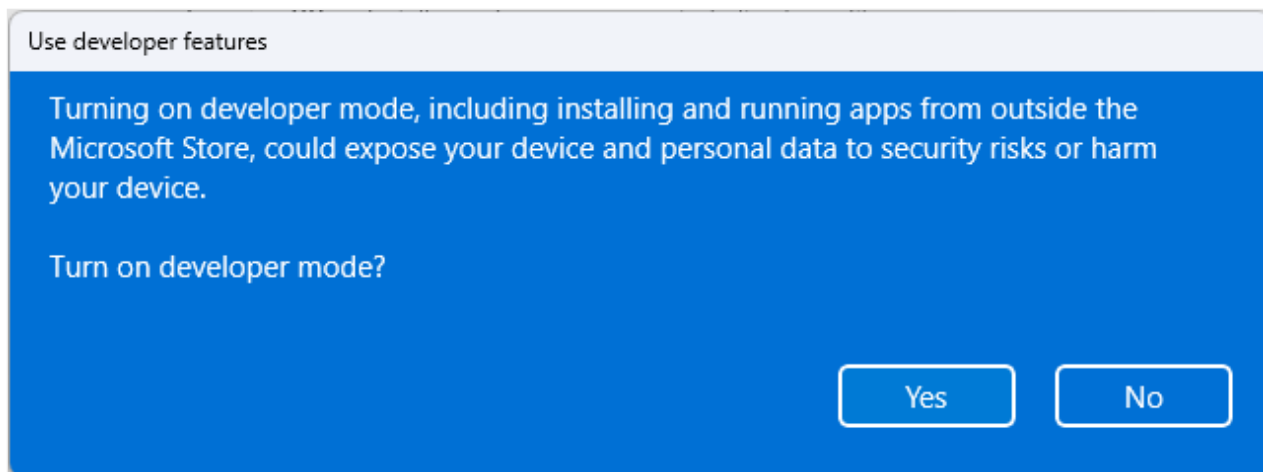
① 重要

コンピューターを日常的なアクティビティ (ゲーム、Web 閲覧、メール、Office アプリなど) に使用している場合は、開発者モードをアクティブにする必要はありません。コンピューターの問題を解決しようとしている場合は、[Windowsヘルプ](#)を確認してください。

開発者モードを有効にするか、他の設定にアクセスするには:

1. Windows設定を開きます。

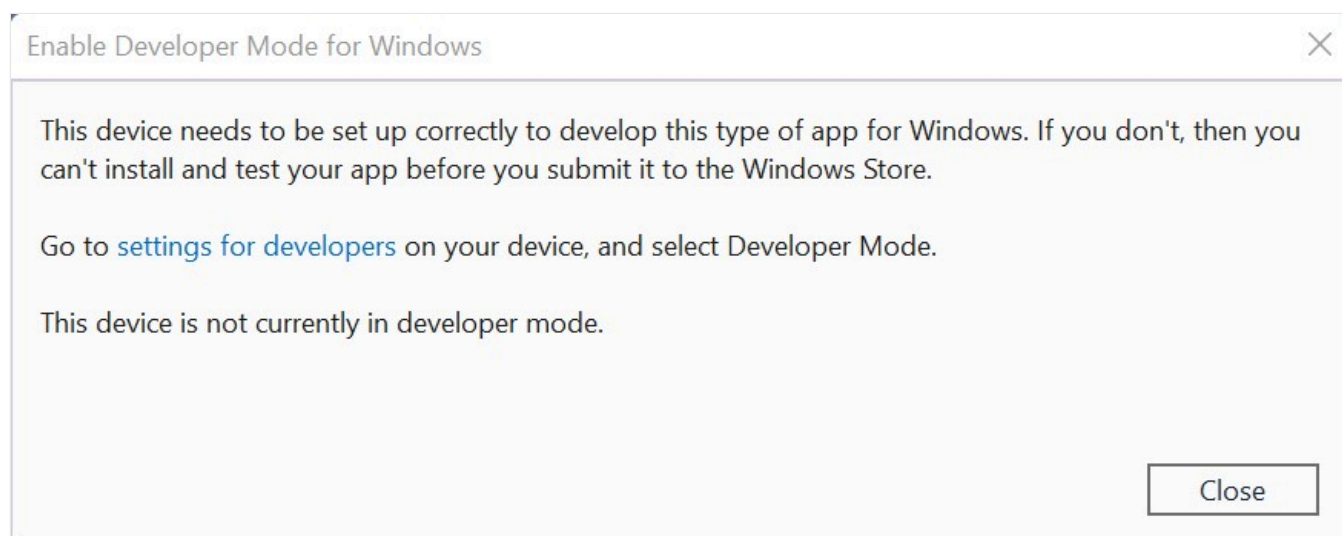
2. **[詳細設定]** を検索するか、**[System > Advanced]** に移動し、**[開発者向け]** セクションまでスクロールします。
3. **[開発者向け]** セクションの上部にある**開発者モードの設定**を切り替えます。
4. 免責事項をお読みください。 変更を受け入れるには、**[はい]** をクリックします。



❗ **Note**

開発者モードを有効にするには、管理者アクセス権が必要です。 デバイスが組織によって所有されている場合は、このオプションが無効になっている可能性があります。

開発者モードが有効になっていないときにWindows プロジェクトをVisual Studioでビルドしようとすると <>>、Visual Studioに次のダイアログが表示されます。



このダイアログが表示された場合は、**開発者向けの設定** を選択して、**[システム > 詳細設定]** ページを開きます。

❗ **Note**

開発者モードを**有効**または**無効**にするには、[詳細設定] ページにいつでも移動できます。

開発者モードの機能

開発者モードでは、開発者ライセンスの要件が置き換えられます。サイドローディングに加えて、開発者モードの設定では、デバッグと追加の展開オプションが有効になります。これには、このデバイスへのデプロイを許可する SSH サービスの開始が含まれます。このサービスを停止するには、開発者モードを無効にする必要があります。

デスクトップで開発者モードを有効にすると、次のような機能のパッケージがインストールされます。

- **Windows デバイス ポータル**: Enable Device Portal オプションがオンになっている場合にのみ、デバイス ポータルが有効になり、ファイアウォールの規則もそれに合わせて構成されます。
- アプリのリモート インストールを許可する SSH サービスのファイアウォール規則をインストールして構成します。Device Discovery を有効にすると、SSH サーバーが有効になります。

ⓘ Note

デバイス ポータル と デバイス検出 は、あるコンピューターで開発する必要があるが、テストのためにアプリを別のコンピューターにデプロイする必要がある場合に便利です。たとえば、タッチ最適化タブレット のユーザー インターフェイスをテストするために、アプリをタブレットにデプロイする必要がある場合です。

デバイスポータル

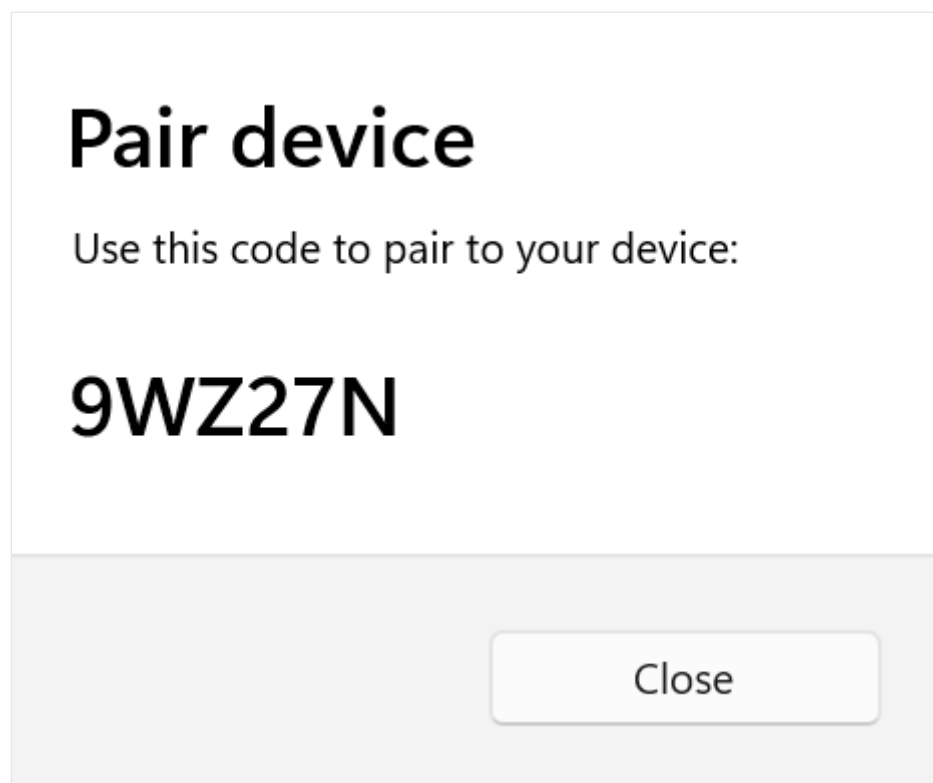
Device Portal の詳細については、「[Windows Device Portal の概要](#)」を参照してください。

具体的なセットアップ手順については、「[デスクトップ用デバイス ポータル](#)」を参照してください。

デバイスの検出

Device Discovery を有効にすると、mDNS を介してデバイスをネットワーク上の他のデバイスに表示できるようになります。また、この機能を使用すると、デバイス検出が有効になった直後に公開される [[ペアリング](#)] ボタンを押して、デバイスとのペアリング用の SSH PIN を取

得することもできます。デバイスを対象とする最初のVisual Studio展開を完了するには、このPIN プロンプトを画面に表示する必要があります。



デバイスを展開ターゲットにする場合にのみ、Device Discovery を有効にする必要があります。たとえば、Device Portal を使用してテスト用のタブレットにアプリを展開する場合は、タブレットで Device Discovery を有効にする必要がありますが、開発用 PC では有効にする必要はありません。

SSH

SSH サービスは、デバイスで Device Discovery を有効にすると有効になります。これは、デバイスが MSIX パッケージ アプリケーションのリモート展開ターゲットである場合に使用されます。サービスの名前は、SSH サーバー ブローカー と SSH サーバー プロキシ。

ⓘ Note

これは Microsoft の OpenSSH 実装ではなく、[GitHub](#) で確認できます。

SSH サービスを利用するために、デバイス検出を有効にしてピンペアリングを許可できます。別の SSH サービスを実行する場合は、別のポートでこれを設定するか、開発者モードの SSH サービスをオフにすることができます。SSH サービスをオフにするには、デバイス検出をオフにします。

SSH ログインは、認証用のパスワードを受け入れる *DevToolsUser* アカウントを使用して行われます。このパスワードは、[デバイスの検出] [ペアリング] ボタンを押した後にデバイスに

表示される PIN であり、PIN が表示されている間のみ有効です。SFTP サブシステムは、Visual Studioからルーズ ファイル展開がインストールされている `DevelopmentFiles` フォルダの手動管理にも使用できます。

SSH の使用に関する注意事項

Windowsで使用されている既存の SSH サーバーはまだプロトコルに準拠していません。SFTP または SSH クライアントを使用するには、特別な構成が必要な場合があります。特に、SFTP サブシステムはバージョン 3 以下で実行されるため、接続しているクライアントは、古いサーバーを想定するように構成する必要があります。古いデバイス上の SSH サーバーでは、公開キー認証に `ssh-dss` が使用されます (OpenSSH は非推奨になりました)。このようなデバイスに接続するには、`ssh-dss` を受け入れるように SSH クライアントを手動で構成する必要があります。

開発者モード パッケージのインストールに失敗する

ネットワークまたは管理上の問題により、開発者モードが正しくインストールされない場合があります。開発者モード パッケージは、この PC へのリモート展開 (ブラウザーまたは Device Discovery から Device Portal を使用して SSH を有効にする) には必要ですが、ローカル開発には必要ありません。これらの問題が発生した場合でも、Visual Studioを使用して (またはこのデバイスから別のデバイスに) アプリをローカルにデプロイできます。

開発者モードが正しくインストールされない場合は、フィードバック Hub アプリを使用してフィードバック要求を提出することをお勧めします。

❗ Note

1. [フィードバック ハブ アプリ](#) (まだインストールしていない場合) をインストールして開きます。
2. **[新しいフィードバックを追加する]**をクリックします。
3. **Developer Platform** カテゴリと **Developer Mode** サブカテゴリを選択します。
4. フィールドに入力し (必要に応じてスクリーンショットを添付することもあります)、そして「**送信**」をクリックします。

フィードバックの送信は、Microsoft が発生した問題を解決するのに役立ちます。

パッケージの検索に失敗しました

開発者モード パッケージをWindows Updateに配置できませんでした。 エラー コードの 0x80004005。 詳細については、こちらを参照してください。

このエラーは、ネットワーク接続の問題、エンタープライズ設定、またはパッケージが見つからない可能性があるために発生する可能性があります。

この問題を修正するには、次のようにします。

1. コンピューターがインターネットに接続されていることを確認します。
2. ドメインに参加しているコンピューターを使用している場合は、ネットワーク管理者にお問い合わせください。 開発者モード パッケージ (すべてのオンデマンド機能など) は、WSUS 2.1 では既定でブロックされます。 現在および以前のリリースの開発者モード パッケージのブロックを解除するには、WSUS で次の KB を許可する必要があります。
 - 4016509
 - 3180030
 - 3197985
3. Settings → **更新プログラムとセキュリティ** → Windows**更新プログラム**でWindows更新プログラムを確認します。
4. Windows Developer Mode パッケージが **設定 → システム → オプション機能 → 機能の追加** に存在することを確認します (Windows 10 22H2 より前のバージョンでは、**設定→アプリ→アプリと機能→オプション機能→ 機能の追加**の下を確認)。 見つからない場合、Windowsはコンピューターの正しいパッケージを見つけることができません。
5. 上記の手順を実行した後、を無効にし、その後開発者モードを再度有効にして修正を確認してください。

パッケージのインストールに失敗しました

開発者モード パッケージのインストールに失敗しました。 エラー コードの0x80004005。 詳細については、こちらを参照してください。

このエラーは、Windowsのビルドと開発者モード パッケージの間に互換性がないために発生する可能性があります。

この問題を修正するには、次のようにします。

1. Settings → Updates と Security → Windows Updates のWindows更新プログラムを確認します。

2. コンピューターを再起動して、すべての更新プログラムが適用されていることを確認します。

グループ ポリシーまたはレジストリ キーを使用してデバイスを有効にする

ほとんどの開発者は、Windows設定を使用して、デバイスのデバッグを有効にする必要があります。特定のシナリオ (自動テストなど) では、他の方法を使用して、Windows デスクトップ デバイスを開発用に有効にすることができます。

① Note

これらの手順では、SSH サーバーを有効にしたり、デバイスをリモート展開とデバッグの対象にしたりすることはできません。

`gpedit.msc` を使用して、*Windows 10 Home* または *Windows 11 Home* がない限り、デバイスを有効にするグループ ポリシーを設定できます。その場合は、`regedit` または PowerShell コマンドを使用してレジストリ キーを直接設定してデバイスを有効にする必要があります。

gpedit を使用してデバイスを有効にする

1. `gpedit.msc` を実行します。
2. ローカル コンピューター ポリシー → コンピューター構成 → 管理用テンプレート → Windows コンポーネント → アプリ パッケージの展開に移動します。
3. サイドローディングを有効にするには、次のポリシーを編集します。
 - すべての信頼されたアプリのインストールを許可します。

OR

サイドローディングと開発者モードの両方を有効にするには、次のポリシーを編集します。

- すべての信頼されたアプリのインストールを許可します。
- 統合開発環境 (IDE) から UWP アプリとインストールを開発できます。
- コンピューターを再起動します。

regedit を使用してデバイスを有効にする

1. `regedit` を実行します。
2. サイドローディングを有効にするには、この `DWORD` の値を `1` に設定します。

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock\AllowAllTrustedApps
```

OR

開発者モードを有効にするには、この `DWORD` の値を `1` に設定します。

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock\AllowDevelopmentWithoutDevLicense
```

PowerShell を使用してデバイスを有効にする

1. 管理者特権で PowerShell を実行します。
2. サイドローディングを有効にするには、次のコマンドを実行します。

PowerShell

```
PS C:\WINDOWS\system32> reg add  
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock"  
/t REG_DWORD /f /v "AllowAllTrustedApps" /d "1"
```

OR

開発者モードを有効にするには、次のコマンドを実行します。

PowerShell

```
PS C:\WINDOWS\system32> reg add  
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\AppModelUnlock"  
/t REG_DWORD /f /v "AllowDevelopmentWithoutDevLicense" /d "1"
```

Windows デバイス ポータルの概要

Windows デバイス ポータル (WDP) は、ネットワークまたは USB 接続経由でデバイスの設定を構成および管理できる Windows デバイスに含まれる Web サーバーです (ローカル接続は、Web ブラウザーを使用するデバイスでもサポートされます)。

WDP には、Windows デバイスのリアルタイム パフォーマンスのトラブルシューティングと表示のための高度な診断ツールも用意されています。

WDP 機能は、[REST API](#) のコレクションを通じてプログラムによって公開されます。

この記事では、Windows デバイス ポータルの一般的な説明と、各 Windows デバイス ファミリに関するより具体的な情報を含む記事へのリンクを示します。

ⓘ Note

デバイス ファミリは、デバイスのクラス全体で期待できる API、システム特性、および動作を識別します。

⚠ 警告

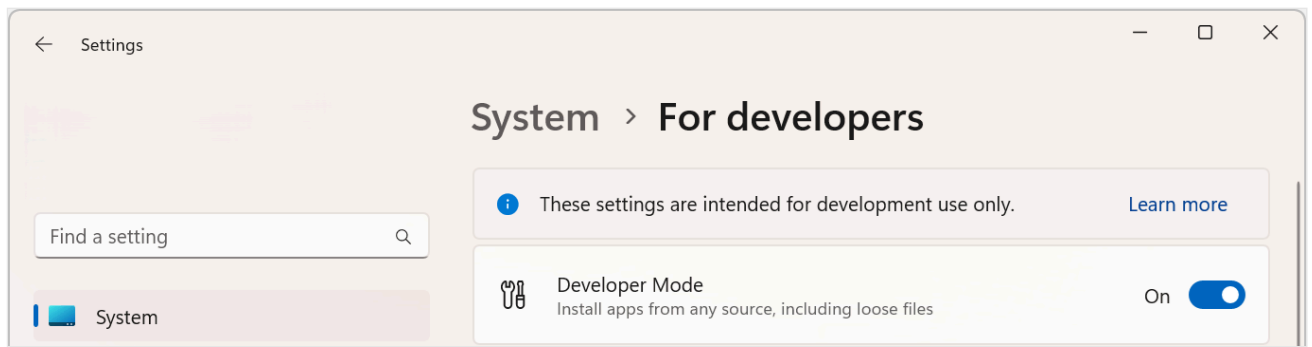
デバイス ポータルは既定で無効になっています。使用する前に、開発者モードを有効にしてから、[\[設定\] > \[システム > 詳細設定\]](#) でデバイス ポータルを明示的に有効にする必要があります。

セットアップ

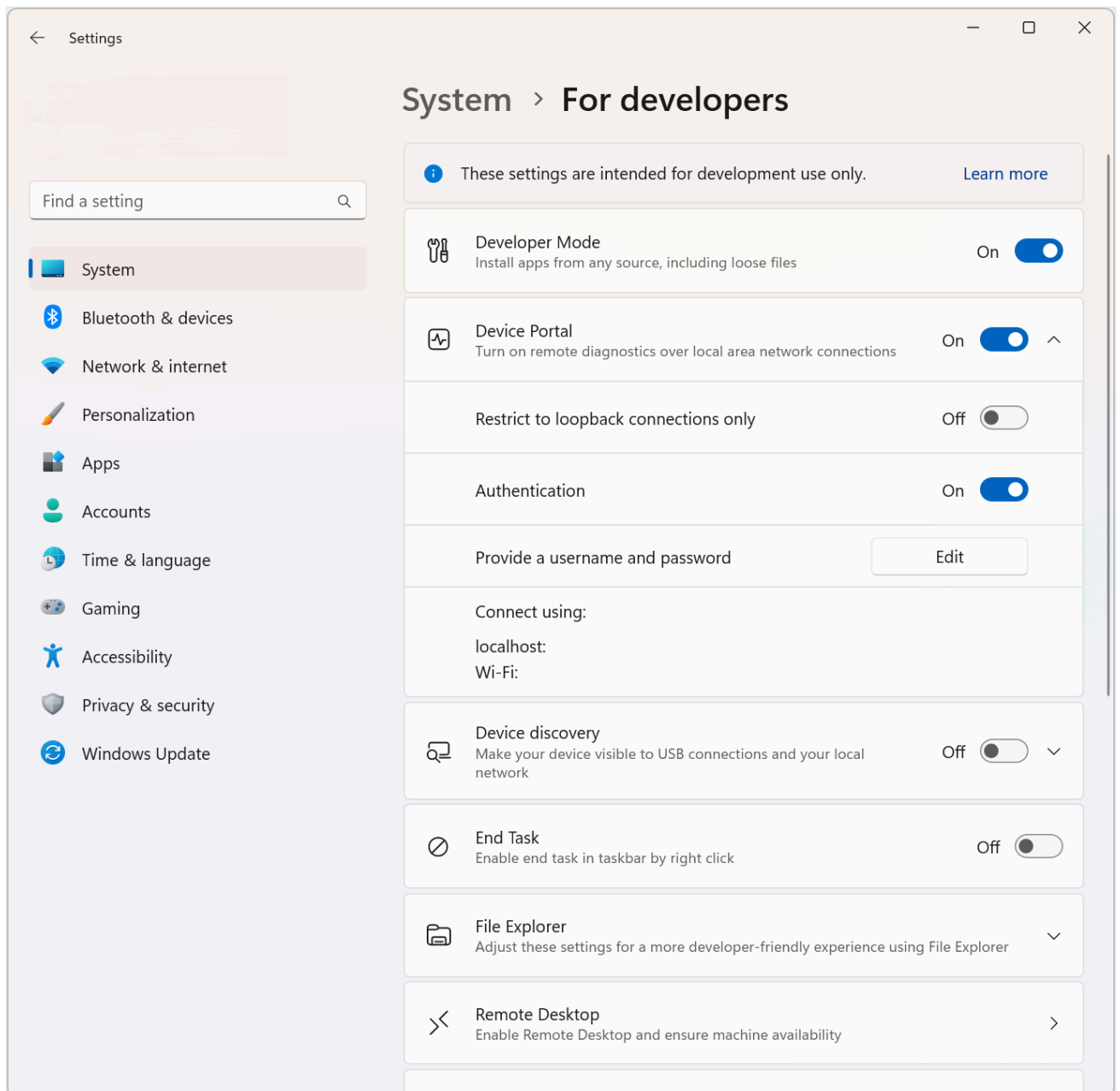
各デバイス ファミリには WDP のバージョンが用意されていますが、機能とセットアップはデバイスの要件によって異なります。

これらは、すべてのデバイスの基本的な手順です。

1. 開発者モードを有効にします。手順については、[開発者向けの設定](#) を参照してください。



2. デバイスで Device Portal を有効にします ([設定] > システム > 詳細設定)。



3. デバイスと PC をローカル ネットワークまたは USB 経由で接続します。

4. ブラウザーの [デバイス ポータル] ページに移動します。 次の表に、各デバイス ファミリで使用されるポートとプロトコルを示します。

次の表に、WDP のデバイス固有の詳細を示します。

デバイス ファミリ	既定ではオンですか?	HTTP	HTTPS	USB	手順
Desktop と IoT Enterprise	開発モード内で有効にする	50080*	50043*	N/A	Desktop または IoT Enterprise デバイス向け Device Portal

* デスクトップ上のデバイス ポータルが、デバイス上の既存のポート要求との競合を防ぐために一時的な範囲 (>50,000) のポートを要求するため、これは常に当てはまるとは限りません。詳細については、[デスクトップ用デバイス ポータルのレジストリ ベースの構成セクション](#) Windows参照してください。

Features

ツール バーとナビゲーション

ページの上部にあるツール バーから、よく使用される機能にアクセスできます。

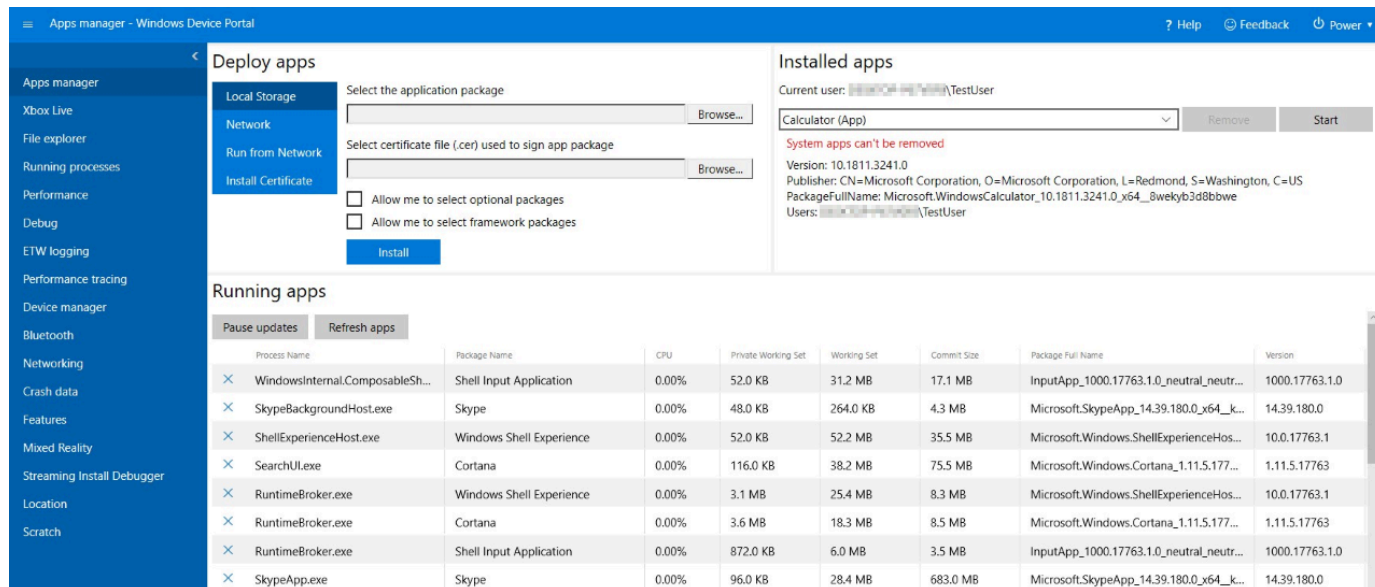
- **電源:** 電源オプションにアクセスします。
 - **シャットダウン:** デバイスをオフにします。
 - **再起動:** デバイスの電源を循環します。
- **ヘルプ:** ヘルプ ページを開きます。

ページの左側にあるナビゲーション ウィンドウのリンクを使用して、デバイスで使用可能な管理および監視ツールに移動します。

ここでは、デバイス ファミリ間で共通のツールについて説明します。デバイスによっては、その他のオプションを使用できる場合があります。詳細については、デバイスの種類の特定のページを参照してください。

アプリ マネージャー

Apps マネージャーは、ホスト デバイス上のアプリ パッケージとバンドルのインストール/アンインストールと管理機能を提供します。

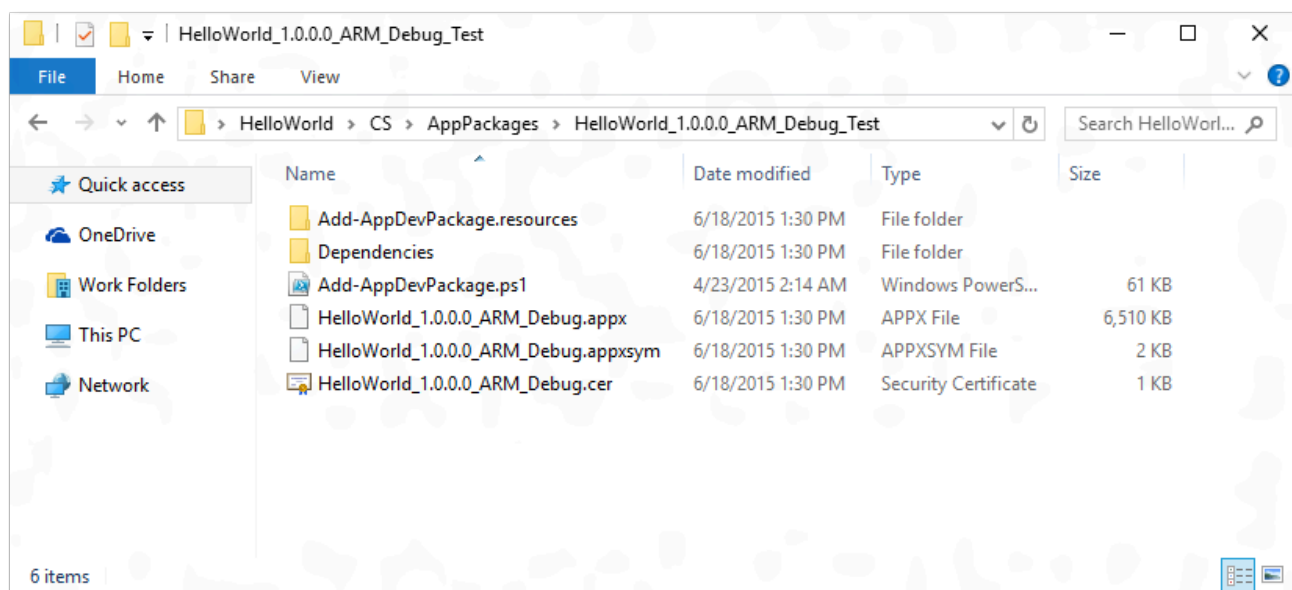


- **アプリの展開:** ローカル、ネットワーク、または Web ホストからパッケージ アプリを展開し、ネットワーク共有からルーズ ファイルを登録します。
- **インストールされているアプリ:** ドロップダウン メニューを使用して、デバイスにインストールされているアプリを削除または起動します。
- **実行中のアプリ:** 現在実行中のアプリに関する情報を取得し、必要に応じて閉じます。

アプリをインストール (サイドロード) する

Windows Device Portal を使用して、開発中にアプリをサイドロードできます。

1. アプリ パッケージを作成したら、デバイスにリモートでインストールできます。 Visual Studioでビルドすると、出力フォルダーが生成されます。



2. Windowsデバイス ポータルで、[アプリ マネージャー] ページに移動します。
3. [アプリのデプロイ] セクションで、[ローカル ストレージ] を選択します。

4. [**アプリケーション パッケージの選択**] で [**ファイルの選択**] を選択し、サイドロードするアプリ パッケージを参照します。
5. [**アプリ パッケージの署名に使用する証明書ファイルの選択 (.cer)**] で、[**ファイルの選択**] を選択し、そのアプリ パッケージに関連付けられている証明書を参照します。
6. アプリのインストールと共にオプションまたはフレームワーク パッケージをインストールする場合は、それぞれのボックスをオンにし、[**次へ**] を選択して選択します。
7. [**インストール**] を選択してインストールを開始します。
8. デバイスが S モードで Windows 10 実行されていて、指定された証明書がデバイスに初めてインストールされた場合は、デバイスを再起動します。

証明書をインストールする

または、Windows Device Portal を使用して証明書をインストールし、他の方法でアプリをインストールすることもできます。

1. Windows デバイス ポータルで、[**アプリ マネージャー**] ページに移動します。
2. [**アプリの展開**] セクションで、[**証明書のインストール**] を選択します。
3. **アプリ パッケージの署名に使用する証明書ファイルの選択 (.cer)** で、[**ファイルの選択**] を選択し、サイドロードするアプリ パッケージに関連付けられている証明書を参照します。
4. [**インストール**] を選択してインストールを開始します。
5. デバイスが S モードで Windows 10 実行されていて、指定された証明書がデバイスに初めてインストールされた場合は、デバイスを再起動します。

アプリのアンインストール

1. アプリが実行されていないことを確認します。
2. その場合は、[**実行中の アプリ**] に移動して閉じます。アプリの実行中にアンインストールしようとする、アプリを再インストールしようとしたときに問題が発生します。
3. ドロップダウンからアプリを選択し、[**削除**] をクリックします。

プロセスの実行

このページには、ホスト デバイスで現在実行されているプロセスの詳細が表示されます。これには、アプリとシステム プロセスの両方が含まれます。このページからアプリとシステム プロセスの両方を終了できます。

ファイル エクスプローラー

このページでは、サイドロードされたアプリによって保存されたファイルを表示および操作できます。 エクスプローラー ページでアプリのローカル ストレージを表示し、アプリ データを表示および管理します。

Performance

[パフォーマンス] ページには、電力使用量、フレーム レート、CPU 負荷などのシステム診断情報のリアルタイム グラフが表示されます。

使用可能なメトリックは次のとおりです。

- **CPU:** 使用可能な CPU 使用率の合計に対する割合
- **メモリ:** 合計メモリ、使用中のメモリ、使用可能なメモリ、コミットされたメモリ、ページメモリ、非ページ メモリ
- **I/O:** データ量の読み取りと書き込み
- **ネットワーク:** データの送受信
- **GPU:** 使用可能な GPU エンジンの使用率の合計に対する割合

Windows イベント トレーシング (ETW) ログ

ETW ログ ページでは、デバイス上のリアルタイムイベントトレーシングWindows (ETW) 情報が管理されます。

[**プロバイダーを非表示にする**] をオンにして、[イベント] リストのみを表示します。

- **登録済みプロバイダー:** イベント プロバイダーとトレース レベルを選択します。 トレース レベルは、次のいずれかの値です。
 1. 異常終了または強制終了
 2. 重大なエラー
 3. Warnings
 4. エラー以外の警告
 5. 詳細なトレース

[**有効にする**] をクリックまたはタップしてトレースを開始します。 プロバイダーが [**有効なプロバイダー**] ドロップダウンに追加されます。

- **カスタム プロバイダー:** カスタム ETW プロバイダーとトレース レベルを選択します。 GUID でプロバイダーを識別します。 GUID には角かっこを含めないでください。

- **有効なプロバイダー:** 有効なプロバイダーが一覧表示されます。ドロップダウンからプロバイダーを選択し、[**無効**] をクリックまたはタップしてトレースを停止します。[**すべて停止**] をクリックまたはタップして、すべてのトレースを中断します。
- **プロバイダー履歴:** 現在のセッション中に有効にされた ETW プロバイダーが表示されます。無効にされたプロバイダーをアクティブ化するには、[**有効**] をクリックまたはタップします。[**クリア**] をクリックまたはタップして履歴をクリアします。
- **フィルター/イベント:** [**イベント**] セクションには、選択したプロバイダーからの ETW イベントがテーブル形式で一覧表示されます。テーブルはリアルタイムで更新されます。[**フィルター**] メニューを使用して、イベントを表示するカスタム フィルターを設定します。[**クリア**] ボタンをクリックして、テーブルからすべての ETW イベントを削除します。これにより、プロバイダーは無効になりません。[**ファイルに保存**] をクリックすると、現在収集されている ETW イベントをローカル CSV ファイルにエクスポートできます。

パフォーマンス トレース

[Windows Performance Toolkit](#) にはパフォーマンス トレース ページが含まれており、ホストデバイスの [Windows Performance Recorder \(WPR\)](#) トレースを表示できます。

- **使用可能なプロファイル:** ドロップダウンから WPR プロファイルを選択し、[**開始**] をクリックまたはタップしてトレースを開始します。
- **カスタム プロファイル:** [**参照**] をクリックまたはタップして、PC から WPR プロファイルを選択します。[**アップロード**] をクリックまたはタップして **トレースを開始** します。

トレースを停止するには、[**停止**] をクリックします。トレース ファイル (.ETL) のダウンロードが完了するまで、このページを表示したままにしてください。

キャプチャ済み。ETLファイルは、[Windows パフォーマンス アナライザー](#)で分析するために開くことができます。

デバイス マネージャー

[デバイス マネージャー] ページには、デバイスに接続されているすべての周辺機器が列挙されます。設定アイコンをクリックすると、それぞれのプロパティを表示できます。

ネットワーク

[ネットワーク] ページでは、デバイス上のネットワーク接続を管理します。USB 経由で Device Portal に接続していない限り、これらの設定を変更すると、デバイス ポータルから切断される可能性があります。

- **使用可能なネットワーク:** デバイスで使用可能な WiFi ネットワークが表示されます。 ネットワークをクリックまたはタップすると、ネットワークに接続し、必要に応じてパスキーを指定できます。 Device Portal では、エンタープライズ認証はまだサポートされていません。 [**プロファイル**] ドロップダウンを使用して、デバイスに認識されている任意の WiFi プロファイルへの接続を試みることもできます。
- **IP 構成:** ホスト デバイスの各ネットワーク ポートに関するアドレス情報を表示します。

サービスの機能と注意事項

DNS-SD

Device Portal は、DNS-SD を使用してローカル ネットワーク上でそのプレゼンスをアドバタイズします。 デバイス の種類に関係なく、すべての Device Portal インスタンスが "WDP._wdp._tcp.local" でアドバタイズされます。 サービス インスタンスの TXT レコードには、次のものが用意されています。

 **テーブルを展開する**

Key	タイプ	Description
S	int	デバイス ポータルのセキュリティで保護されたポート。 0 (ゼロ) の場合、Device Portal は HTTPS 接続をリッスンしていません。
D	文字列	デバイスの種類。 これは "Windows.*" という形式になります。たとえば、Windows.Desktop または Windows.IoTEnterprise です。
A	文字列	デバイス アーキテクチャ。 これは、Arm、x86、または AMD64 になります。
T	ヌル文字で区切られた文字列のリスト	デバイスにユーザーが適用したタグ。 これを使用する方法については、Tags REST API を参照してください。 リストが double-null で終わる。

すべてのデバイスが、DNS-SD レコードによってアドバタイズされた HTTP ポートでリッスンしているわけではないので、HTTPS ポートでの接続が推奨されます。

CSRF の保護とスクリプト

[CSRF 攻撃](#) から保護するには、GET 以外のすべての要求で一意的トークンが必要です。 このトークン (X-CSRF-Token 要求ヘッダー) は、セッション Cookie である CSRF-Token から派生します。 Device Portal Web UI では、CSRF-Token Cookie が各要求の X-CSRF-Token ヘッダーにコピーされます。

重要

この保護により、スタンドアロン クライアント (コマンドライン ユーティリティなど) から REST API を使用できなくなります。これは、次の 3 つの方法で解決できます。

- "auto-" で始まるユーザー名を使用します。ユーザー名に "auto-" を付加するクライアントは、CSRF 保護をバイパスします。このユーザー名は、CSRF 攻撃に対するサービスを開くので、ブラウザー経由でデバイス ポータルにログインするために使用されないことが重要です。例: Device Portal のユーザー名が "admin" の場合は、CSRF 保護をバイパスするために `curl -u auto-admin:password <args>` を使用する必要があります。
- クライアントで Cookie からヘッダーへのスキームを実装します。これには、セッション Cookie を確立するための GET 要求と、後続のすべての要求にヘッダーと Cookie の両方を含める必要があります。
- 認証を無効にし、HTTP を使用します。CSRF 保護は HTTPS エンドポイントにのみ適用されるため、HTTP エンドポイント上の接続で上記のいずれかを行う必要はありません。

クロスサイト WebSocket ハイジャック (CSWSH) 対策

[CSWSH 攻撃](#) から保護するには、デバイス ポータルへの WebSocket 接続を開くすべてのクライアントも、ホスト ヘッダーに一致する Origin ヘッダーを提供する必要があります。これは、要求が Device Portal UI または有効なクライアント アプリケーションから送信されたことを Device Portal に証明します。Origin ヘッダーがない場合、要求は拒否されます。

こちらも参照ください

[Device Portal コア API リファレンス](#)

デスクトップ用 Windows デバイス ポータル

Windows Device Portal (WDP) は、デバイスの設定を構成および管理し、Web ブラウザーから HTTP 経由で診断情報を表示できるデバイス管理およびデバッグ ツールです。他のデバイスの WDP の詳細については、「[Windows Device Portal の概要](#)」を参照してください。

WDP は次の目的で使用できます。

- デバイス設定の管理 (Windows **設定** アプリと同様)
- 実行中のプロセスの一覧を表示および操作する
- アプリのインストール、削除、起動、終了
- Wi-Fi プロファイルの変更、信号強度の表示、ipconfig の詳細の表示
- CPU、メモリ、I/O、ネットワーク、GPU 使用率のライブ グラフを表示する
- プロセス ダンプを収集する
- ETW トレースを収集する
- サイドロードされたアプリの分離ストレージを操作する

デスクトップ デバイス上で Windows Device Portal を設定する

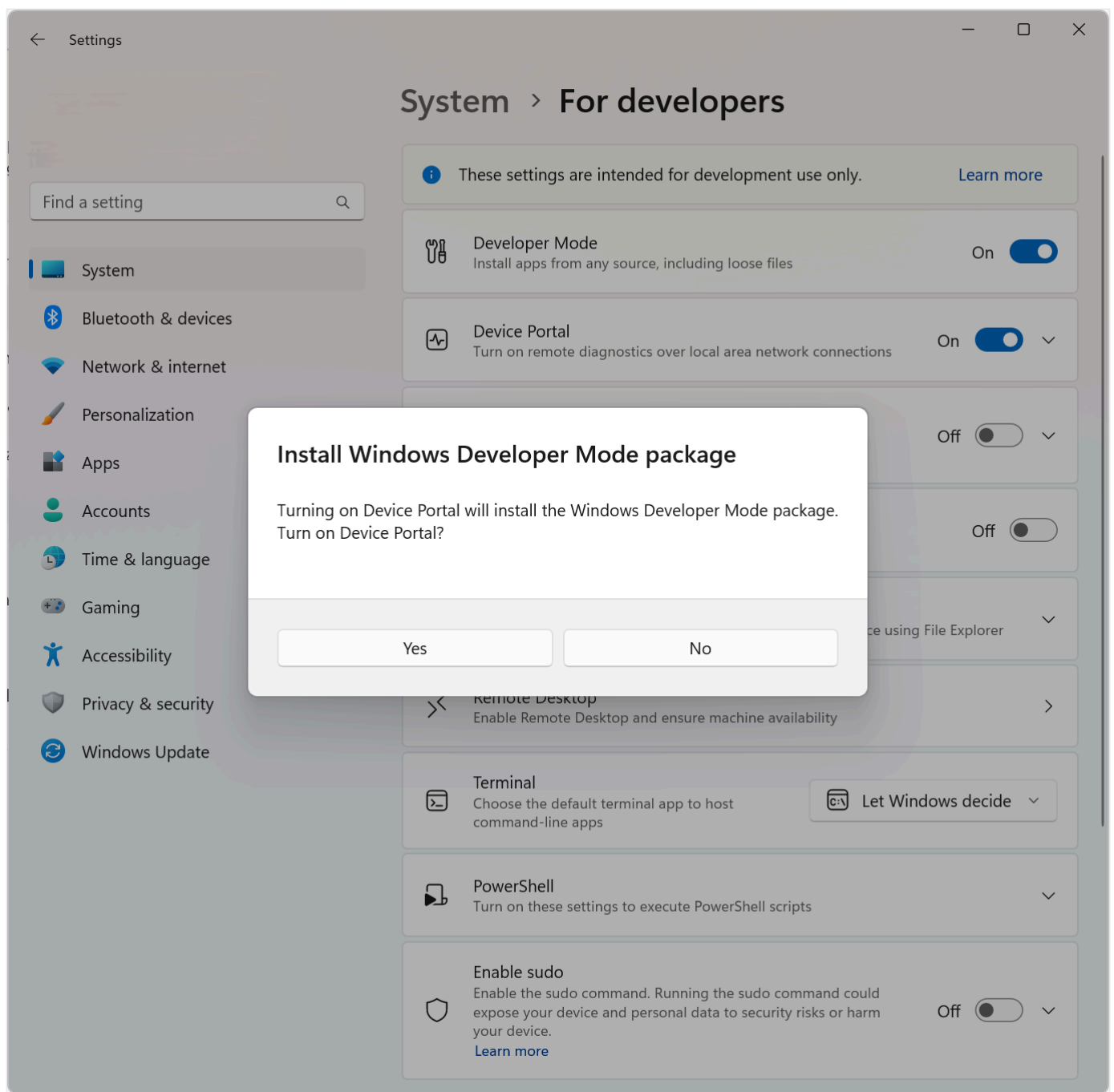
デバイス ポータルを有効にする前に、開発者モードが必要です。

ⓘ Note

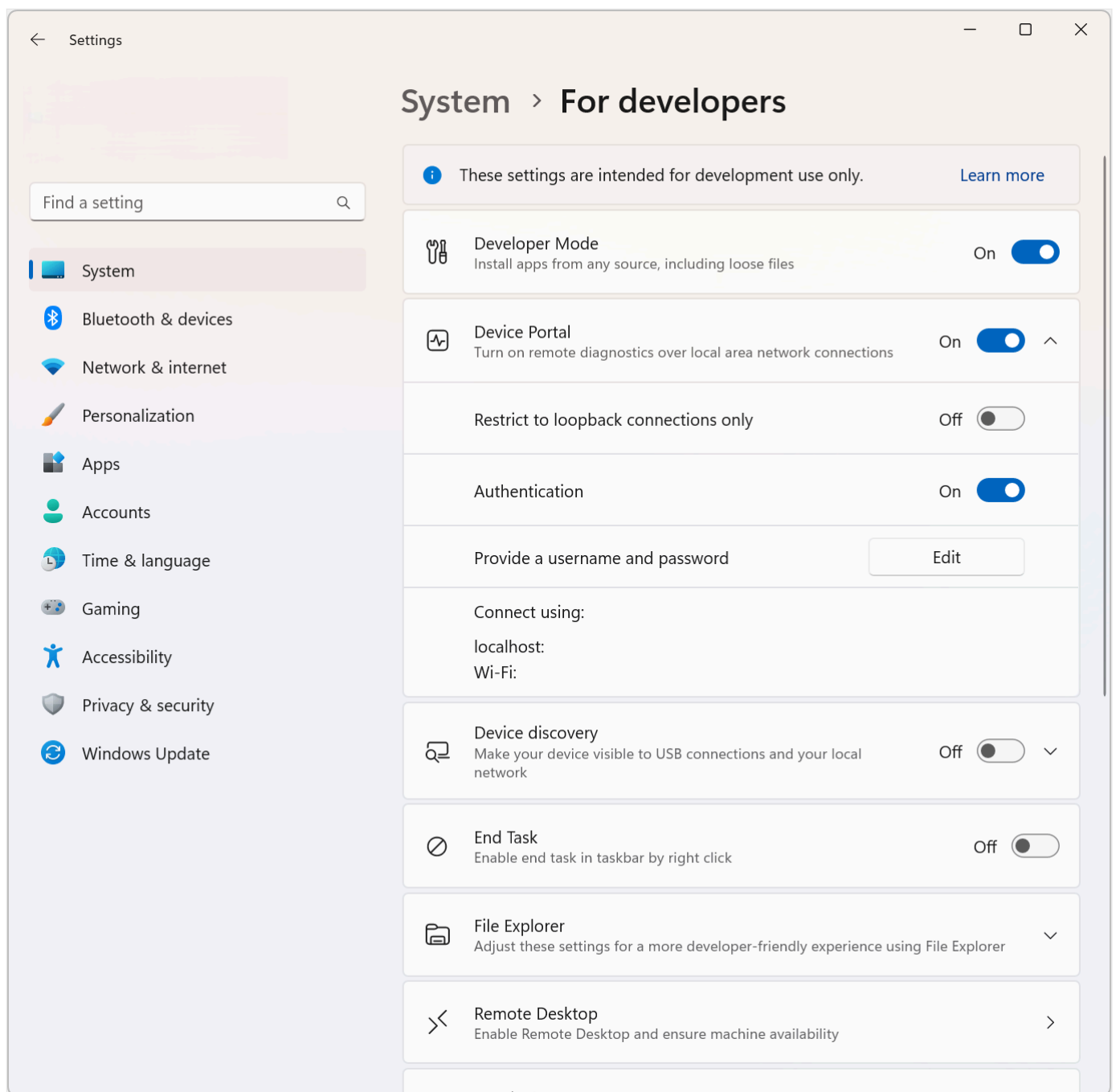
開発者モードを有効にする手順については、「[開発者向けの設定](#)」を参照してください。

Windows デバイス ポータルを有効にする

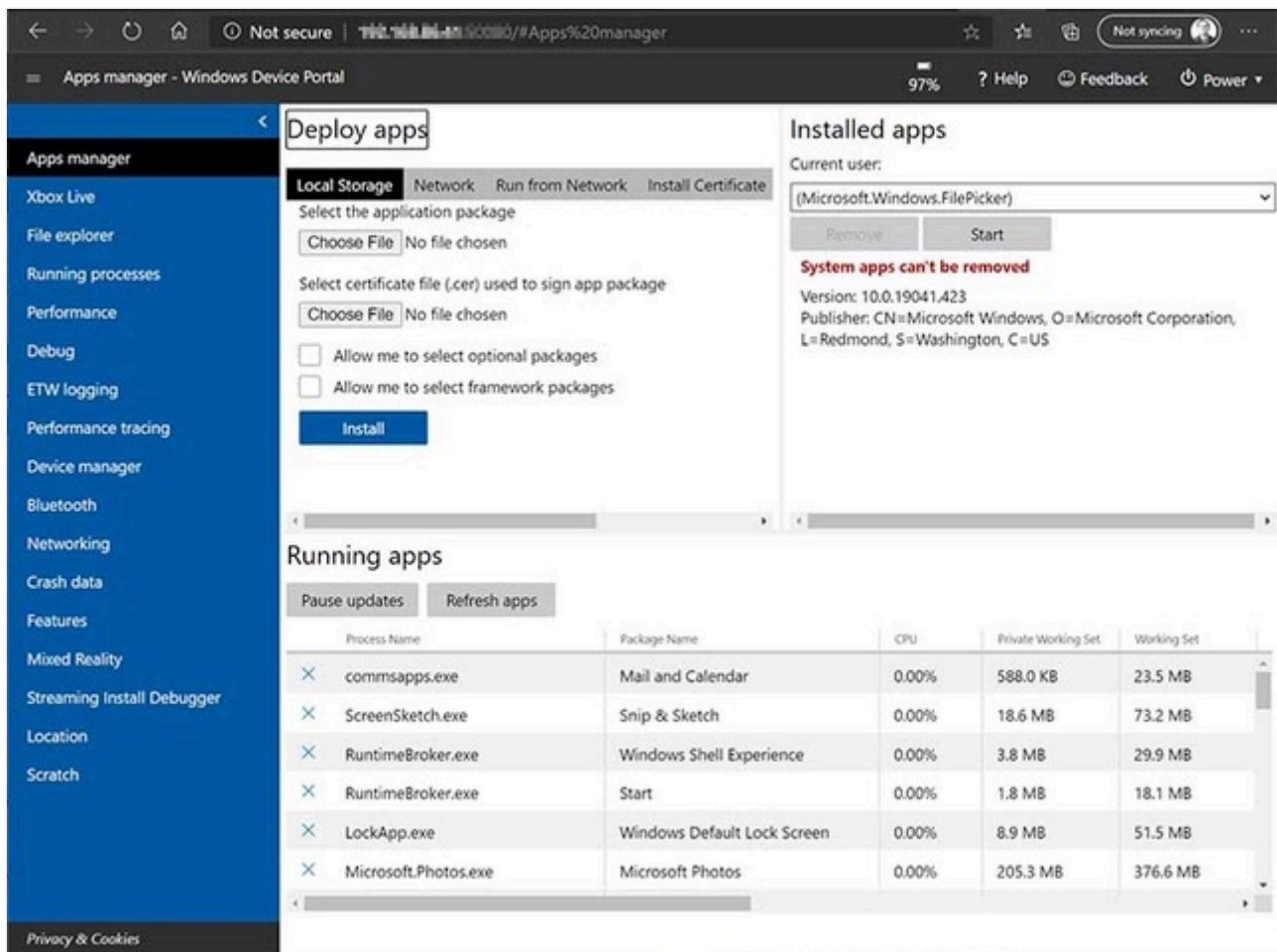
WDP は、**[設定]** の **[開発者向け]** セクションで有効にすることができます (**[設定]** > **システム** > **詳細設定**)。



WDP が有効になると、[デバイス ポータル] セクションの下部に URI が表示されます。一覧表示されている URL の末尾に追加されたポート番号を書き留めます。この数は、WDP が有効になっているときにランダムに生成されますが、デスクトップの再起動間で一貫性を保つ必要があります。



これらのリンクは、WDP に接続する方法として、ローカル ネットワーク (VPN を含む) 経由またはローカル ホスト経由の 2 つの方法を提供します。



Windows デバイス ポータルをオフにする

WDP は、Windows 設定の [開発者向け] セクションで無効にすることができます ([\[設定\] > システム > 詳細設定](#))。

Windows デバイス ポータルに接続する

ローカル ホスト経由で接続するには、ブラウザー ウィンドウを開き、ここに表示されている URI のいずれかを入力します (使用している接続の種類に基づいて)。

- Localhost: `http://127.0.0.1:<PORT>` または `http://localhost:<PORT>`
- ローカル ネットワーク: `https://<IP address of the desktop>:<PORT>`

認証とセキュリティで保護された通信には HTTPS が必要です。

WDP を保護された環境 (たとえば、ローカル ネットワーク上のすべてのユーザーを信頼し、デバイスに個人情報がなく、固有の要件があるテスト ラボ) で使用している場合は、[認証] オプションを無効にすることができます。これにより、暗号化されていない通信が可能になり、コンピュータの IP アドレスを持つすべてのユーザーが接続して制御できるようになります。

Windows デバイス ポータルのコンテンツ

WDP には、次の一連のページが用意されています。

- アプリ マネージャー
- ファイル エクスプローラー
- プロセスの実行
- Performance
- Debug
- ETW (Windowsのイベント トレース) のログ記録
- パフォーマンス トレース
- デバイス マネージャー
- Bluetooth
- ネットワーク
- クラッシュデータ
- Features
- 場所
- Scratch

Windows Device Portal を使用して MSIX アプリをテストおよびデバッグする

次のビデオでは、Windows Device Portal を使用して [MSIX アプリ](#) をリモートでテストおよびデバッグする方法について説明します。

<https://www.youtube-nocookie.com/embed/PdgXeOMt4hk> [↗](#)

デバイス ポータルのその他のWindowsオプション

次のセクションでは、デバイス ポータルのその他のWindowsオプションについて詳しく説明します。

レジストリ ベースの構成

WDP のポート番号 (80 や 443 など) を選択する場合は、次の regkey を設定できます。

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\WebManagement\Service` の下
 - `UseDynamicPorts`: 必須の DWORD 値。選択したポート番号を保持するには、これを 0 に設定します。

- `HttpPort`: 必須の DWORD 値。WDP が HTTP 接続をリッスンするポート番号を格納します。
- `HttpsPort`: 必須の DWORD 値。WDP が HTTPS 接続をリッスンするポート番号を格納します。

同じ regkey パスで、認証要件をオフにすることもできます。

- 無効の場合は `UseDefaultAuthorizer` - 0、有効の場合は 1。
 - これにより、各接続の基本的な認証要件と、HTTP から HTTPS へのリダイレクトの両方が制御されます。

Windows Device Portal のコマンド ライン オプション

管理コマンド プロンプトから、WDP の一部を有効にして構成できます。お使いのビルドでサポートされている最新のコマンド一覧を確認するには、`webmanagement /?` を実行できます。

- `sc start webmanagement` または `sc stop webmanagement`
 - サービスのオンとオフを切り替えます。そのためには、開発者モードを有効にする必要があります。
- `-Credentials <username> <password>`
 - WDP のユーザー名とパスワードを設定します。ユーザー名は基本認証標準に準拠している必要があるため、コロン (:) を含めることはできません。ブラウザーは標準の方法で完全な文字セットを解析しないため、[a-zA-Z0-9] など、標準の ASCII 文字から構築する必要があります。
- `-DeleteSSL`
 - これにより、HTTPS 接続に使用される SSL 証明書キャッシュがリセットされます。(予想される証明書の警告ではなく) バイパスできない TLS 接続エラーが発生した場合、このオプションによって問題が解決される可能性があります。
- `-SetCert <pfxPath> <pfxPassword>`
 - 詳細については、「[カスタム SSL 証明書を使用した Windows Device Portal のプロビジョニング](#)」を参照してください。
 - これにより、独自の SSL 証明書をインストールして、WDP で通常表示される SSL 警告ページを修正できます。
- `-Debug <various options for authentication, port selection, and tracing level>`
 - 特定の構成と表示されるデバッグ メッセージを使用して、WDP のスタンドアロンバージョンを実行します。これは、[パッケージ化されたプラグイン](#)を構築する場合に最も役立ちます。
 - パッケージ化されたプラグインを完全にテストするためにシステムとしてこれを実行する方法の詳細については、[MSDN マガジンの記事](#)を参照してください。

Troubleshooting

以下は、Windows Device Portal のセットアップ時に発生する可能性がある一般的なエラーです。

WindowsUpdateSearch が無効な数の更新プログラムを返す (0x800f0950)

Windowsのプレリリース ビルドに開発者パッケージをインストールしようとする、このエラーが発生することがあります。これらのオンデマンド機能 (FoD) パッケージは、Windows Update でホストされ、プレリリース ビルドでダウンロードするには、フライティングをオプトインする必要があります。インストールで適切なビルドとリングの組み合わせのフライティングをオプトインしていない場合、ペイロードはダウンロードできません。以下を再確認してください。

1. **設定 > 更新とセキュリティ > Windows Insider Program** に移動し、**Windows Insider アカウント** セクションに正しいアカウント情報が表示されていることを確認します。このセクションが表示されない場合は、**Windows Insider アカウントをリンク**し、メール アカウントを追加して、**Windows Insider アカウント**見出しの下に表示されることを確認します (新しく追加されたアカウントを実際にリンクするには、**Windows Insider アカウント**を 2 回目にリンクするを選択する必要がある場合があります)。
2. **[どのような種類のコンテンツを受信しますか?]** で、**[Windowsのアクティブな開発]** が選択されていることを確認します。
3. **[新しいビルドを取得するペース]** で、**[Windows Insider Fast]** が選択されていることを確認します。
4. これで FoD をインストールできるようになります。Windows Insider Fast を使用していて、FoD をインストールできない場合は、フィードバックを提供し、**C:\Windows\Logs\CBS** の下にログ ファイルを添付してください。

[SC]StartService: OpenService FAILED 1060: 指定されたサービスがインストール済みサービスとして存在しません

開発者パッケージがインストールされていない場合は、このエラーが発生する可能性があります。開発者パッケージがない場合、Web 管理サービスはありません。開発者パッケージをもう一度インストールしてみてください。

システムが従量制課金ネットワーク上にあるため、CBS はダウンロードを開始できません (CBS_E_METERED_NETWORK)

従量制課金インターネット接続を使用している場合は、このエラーが発生する可能性があります。 従量制課金接続で開発者パッケージをダウンロードすることはできません。

こちらも参照ください

- [Windows デバイス ポータルの概要](#)
- [Windows Device Portal のコア API リファレンス](#)

Last updated on 2026/07/17

カスタム SSL 証明書を使用して Windows デバイス ポータルをプロビジョニングする

Windows Device Portal (WDP) は、デバイス管理者が HTTPS 通信で使用するカスタム証明書をインストールする方法を提供します。

これは自分の PC で行うことができますが、この機能は主に既存の証明書インフラストラクチャが存在する企業を対象としています。

たとえば、企業は、HTTPS 経由で提供されるイントラネット Web サイトの証明書に署名するために使用する証明機関 (CA) を持っている場合があります。この機能は、そのインフラストラクチャの上に立っています。

Overview

既定では、WDP は自己署名ルート CA を生成し、それを使用して、リッスンしているすべてのエンドポイントの SSL 証明書に署名します。これには、`localhost`、`127.0.0.1`、`::1` (IPv6 localhost) が含まれます。

また、デバイスのホスト名 (`https://LivingRoomPC` など) と、デバイスに割り当てられた各リンクローカル IP アドレス (ネットワーク アダプターあたり最大 2 つの [IPv4、IPv6]) も含まれます。WDP のネットワーク ツールを見ると、デバイスのリンクローカル IP アドレスを確認できます。IPv4 の場合は `10.` または `192.`、IPv6 の場合は `fe80:` から始まります。

既定のセットアップでは、信頼されていないルート CA が原因で、証明書の警告がブラウザーに表示されることがあります。具体的には、WDP によって提供される SSL 証明書は、ブラウザーまたは PC が信頼しないルート CA によって署名されます。これは、新しい信頼されたルート CA を作成することで修正できます。

ルート CA を作成する

これは、会社 (または自宅) に証明書インフラストラクチャが設定されておらず、1 回だけ実行する必要がある場合にのみ行う必要があります。証明書ストアに書き込むため、`LocalMachine\My` プロンプトからこのスクリプトを実行します。このスクリプトは、`WdpTestCA.cer` という名前のルート CA を作成します。このファイルをローカル コンピューターの信頼されたルート証明機関にインストールすると、デバイスはこのルート CA によって署名された SSL 証明書を信頼しま

す。WDP に接続する各 PC に、この.cer ファイルをインストールできます (インストールする必要があります)。

PowerShell

```
$CN = "PickAName"
$OutputPath = "c:\temp\"

# Create root certificate authority (marked as a CA via Basic Constraints)
$FilePath = $OutputPath + "WdpTestCA.cer"
$Subject = "CN=" + $CN
$rootCA = New-SelfSignedCertificate -certstorelocation cert:\localmachine\my -Subject
$Subject -HashAlgorithm "SHA512" -KeyUsage CertSign,CRLSign -TextExtension
@"2.5.29.19={critical}{text}CA=true"
$rootCAFile = Export-Certificate -Cert $rootCA -FilePath $FilePath
```

ルート CA は、(秘密キーを使用して) LocalMachine\My に存在するようになりました。エクスポートされた WdpTestCA.cer は、接続している各 PC の信頼されたルート証明機関にインストールするパブリック証明書です。

ルート CA を使用して SSL 証明書を作成する

SSL 証明書には、暗号化による接続のセキュリティ保護と、悪意のあるサードパーティではなく、ブラウザーバー (Bing.com、192.168.1.37 など) に表示されているアドレスと実際に通信していることを確認する 2 つの重要な機能があります。

次の PowerShell スクリプトは、localhost エンドポイントの SSL 証明書を作成します。WDP がリッスンする各エンドポイントには、独自の証明書が必要です。スクリプトの \$IssuedTo 引数は、ホスト名、localhost、IP アドレスなど、デバイスの各異なるエンドポイントに置き換えることができます。このスクリプトには、**管理者特権での PowerShell** プロンプトも必要です。Get-ChildItem フィルターでは、手順 1 で選択したのと同じ CN (PickAName) が使用されます。同期したままにします。

PowerShell

```
$IssuedTo = "localhost"
$Password = "PickAPassword"
$OutputPath = "c:\temp\"

# Retrieve the root CA (with its private key) from the store where it was created.
# Match the CN you used above.
$rootCA = Get-ChildItem -Path cert:\localmachine\my | Where-Object { $_.Subject -eq
"CN=PickAName" -and $_.HasPrivateKey } | Select-Object -First 1

# Create SSL cert signed by certificate authority
$IssuedToClean = $IssuedTo.Replace(":", "-").Replace(" ", "_")
```

```
$FilePath = $OutputPath + $IssuedToClean + ".pfx"
$Subject = "CN=" + $IssuedTo
$cert = New-SelfSignedCertificate -certstorelocation cert:\localmachine\my -Subject
$Subject -DnsName $IssuedTo -Signer $rootCA -HashAlgorithm "SHA512"
$certFile = Export-PfxCertificate -cert $cert -FilePath $FilePath -Password
(ConvertTo-SecureString -String $Password -Force -AsPlainText)
```

複数のデバイスがある場合は、localhost.pfx ファイルを再利用できますが、デバイスごとに IP アドレス証明書とホスト名証明書を作成する必要があります。

.pfx ファイルのバンドルが生成されたら、WDP に読み込む必要があります。

証明書を使用して Windows Device Portal をプロビジョニングする

デバイス用に作成した .pfx ファイルごとに、管理者特権のコマンド プロンプトから次のコマンドを実行する必要があります。

Windows コマンド プロンプト

```
WebManagement.exe -SetCert <Path to .pfx file> <password for pfx>
```

使用例については、以下を参照してください。

Windows コマンド プロンプト

```
WebManagement.exe -SetCert localhost.pfx PickAPassword
WebManagement.exe -SetCert --1.pfx PickAPassword
WebManagement.exe -SetCert MyLivingRoomPC.pfx PickAPassword
```

証明書をインストールしたら、変更を有効にするためにサービスを再起動します。

Windows コマンド プロンプト

```
sc stop webmanagement
sc start webmanagement
```

💡 ヒント

IP アドレスは時間の経過と同時に変化する可能性があります。多くのネットワークでは DHCP を使用して IP アドレスが割り当てられます。そのため、デバイスは以前と同じ IP アドレスを常に取得するとは限りません。デバイス上の IP アドレスの証明書を作成し、そのデバイスのアドレスが変更された場合、WDP は既存の自己署名証明書を使用して新しい証

明書を生成し、作成した証明書の使用を停止します。これにより、証明書の警告ページがブラウザーに再び表示されます。このため、Windows デバイス ポータルで設定できるホスト名を使用してデバイスに接続することをお勧めします。IP アドレスに関係なく、これらは同じままです。

こちらも参照ください

- [Windows デバイス ポータルの概要](#)
- [デスクトップ用デバイス ポータル](#)
- [Windows Device Portal のコア API リファレンス](#)

Last updated on 2026/07/18

Windows Device Portal 用のカスタムプラグインを作成する

Windows Device Portal を使用して Web ページをホストし、診断情報を提供するパッケージ アプリ (Windows アプリ SDK を使用して UWP またはパッケージ化された Win32) を作成できます。Device Portal プラグイン機能は、UI フレームワークに関係なく、パッケージ化されたすべてのアプリで使用できる `windows.devicePortalProvider` 拡張機能を使用して実装されます。

ⓘ Note

この記事は、IoT Enterprise デバイス、キオスク展開、または Device Portal がリモート監視に使用されるその他のマネージド Windows 環境用の診断ツールまたは管理ツールを構築する開発者を対象としています。標準的な Windows 11 デスクトップ アプリを構築する場合は、Device Portal プラグインが必要な可能性はほとんどありません。代わりに、[アプリ診断](#)または Windows パフォーマンス アナライザーの使用を検討してください。

Device Portal プラグインのサポートは、Windows 10 以降で利用できます。

プロジェクトを作成する

Microsoft Visual Studio で新しいアプリ プロジェクトを作成します。[ファイル] > [新しい > Project] に移動し、次の project テンプレートのいずれかを選択します。

- UWP アプリの場合: **C# 用の空のアプリ** (Windows ユニバーサル)
- WinUI 3 デスクトップ アプリの場合: **パッケージ化された空のアプリ** (Desktop の WinUI 3)

[新しいプロジェクトの構成] ダイアログ ボックスで、プロジェクトに `DevicePortalProvider` という名前を付け、[作成] をクリックします。これは、アプリ サービスを含むアプリになります。

devicePortalProvider 機能を追加する

アプリの `rescap:devicePortalProvider` ファイルに `Package.appxmanifest` 機能を追加します。

ⓘ Note

rescap (制限付き機能) 名前空間は、使用する前に宣言する必要があります。詳細については、「[アプリ機能の宣言](#)」を参照してください。

ソリューション エクスプローラーで、`Package.appxmanifest` をダブルクリックしてマニフェスト デザイナーを開き、[機能] タブに切り替えて、**デバイス ポータル プロバイダー**が表示されている場合はオンにするか、XML を編集して手動で追加します。

XML

```
<Capabilities>
  <rescap:Capability Name="devicePortalProvider" />
</Capabilities>
```

アプリ サービスを設定する

Device Portal プラグインは、[アプリ サービス](#) を使用して、プラグイン ホスト プロセスと Device Portal の間で通信します。`Package.appxmanifest` 開き、[宣言] タブに切り替え、**Windows Device Portal プロバイダー**宣言を追加して、サービス名で構成します。

または、XML を直接編集します。

XML

```
<Applications>
  <Application Id="App" ...>
    ...
    <Extensions>
      <uap3:Extension Category="windows.appService"
EntryPoint="DevicePortalProvider.DevicePortalPluginService">
        <uap3:AppService Name="com.microsoft.myapp.devicePortalService"
SupportsRemoteSystems="true"/>
      </uap3:Extension>
      <uap4:Extension Category="windows.devicePortalProvider">
        <uap4:DevicePortalProvider DisplayName="My App Plugin"
AppServiceName="com.microsoft.myapp.devicePortalService"
HandlerRoute="/MyCustomApp"/>
      </uap4:Extension>
    </Extensions>
  </Application>
</Applications>
```

 **重要**

uap3 および uap4 XML 名前空間プレフィックスは、マニフェストのルート `<Package>` 要素で宣言し、`IgnorableNamespaces` 属性に追加する必要があります。

XML

```
<Package
...
xmlns:uap3="http://schemas.microsoft.com/appx/manifest/uap/windows10/3"
xmlns:uap4="http://schemas.microsoft.com/appx/manifest/uap/windows10/4"
IgnorableNamespaces="... uap3 uap4">
```

次の属性には、アプリに固有の構成が必要です。

 テーブルを展開する

Attribute	Description
EntryPoint	App Service を実装するクラスを指定します。
Name (AppService)	アプリ サービスの一意の名前。 この名前は、 <code>AppServiceName</code> 要素の <code>DevicePortalProvider</code> 属性として使用されます。
DisplayName	デバイス ポータル UI に表示される名前。
HandlerRoute	デバイス ポータルがプラグインに要求をルーティングするために使用する URL パス プレフィックス。 パスがこの値で始まるすべての要求は、アプリ サービスに転送されます。 ルート名は、先頭と末尾にスラッシュを付ける必要があります。

App Service コードを追加する

Windows ランタイム コンポーネント プロジェクトをソリューションに追加して、バックグラウンド アプリ サービスをホストします。

1. **ソリューション エクスプローラー**で、ソリューションを右クリックし、**[新しいProject>追加]** を選択します。
2. **Windows ランタイム コンポーネント (ユニバーサル Windows)** を選択し、**DevicePortalProvider** という名前を付けます。
3. 新しいプロジェクトを右クリックし、**[Add > New Item > App Service]** を選択します。

App Service コードの実装の詳細については、[App Service の作成と使用に関する記事](#)を 参照してください。

Device Portal ハンドラーを実装する

Device Portal プラグイン アプリ サービスの実装例を次に示します。

C#

```
using Windows.ApplicationModel.AppService;
using Windows.ApplicationModel.Background;
using Windows.Storage;
using Windows.System.Diagnostics.DevicePortal;
using Windows.Web.Http;

namespace DevicePortalProvider
{
    public sealed class DevicePortalPluginService : IBackgroundTask
    {
        private BackgroundTaskDeferral _taskDeferral;
        private DevicePortalConnection _connection;

        public void Run(IBackgroundTaskInstance taskInstance)
        {
            _taskDeferral = taskInstance.GetDeferral();
            taskInstance.Canceled += OnTaskCanceled;

            var triggerDetails = taskInstance.TriggerDetails as
AppServiceTriggerDetails;
            _connection =
DevicePortalConnection.GetForAppServiceConnection(triggerDetails.AppServiceConnection)
;
            _connection.RequestReceived += _connection_RequestReceived;
        }

        private void OnTaskCanceled(IBackgroundTaskInstance taskInstance,
BackgroundTaskCancellationReason reason)
        {
            _taskDeferral.Complete();
        }

        private async void _connection_RequestReceived(DevicePortalConnection sender,
DevicePortalConnectionRequestReceivedEventArgs args)
        {
            // Handle the incoming request and send a response
        }
    }
}
```

受信要求を処理する

プラグイン ハンドラーは、Device Portal が `HandlerRoute` にルーティングするすべての GET、POST、DELETE、および PUT 要求を受け取ります。 一般的な実装:

C#

```
private async void _connection_RequestReceived(
    DevicePortalConnection sender,
    DevicePortalConnectionRequestReceivedEventArgs args)
{
    var request = args.RequestMessage;
    var response = args.ResponseMessage;

    // Use the request's path to determine what data to return
    // The path must begin with the HandlerRoute declared in the manifest (e.g.
    // "/MyCustomApp/")
    if (request.RequestUri.LocalPath.Equals("/MyCustomApp/info",
        StringComparison.OrdinalIgnoreCase))
    {
        response.Content = new HttpStringContent(
            "{ \"appVersion\": \"1.0\" }",
            Windows.Storage.Streams.UnicodeEncoding.Utf8,
            "application/json");
        response.StatusCode = HttpStatusCode.Ok;
    }
}
```

静的コンテンツを提供する

アプリのパッケージ内から Web コンテンツ (HTML、CSS、JavaScript) を提供する場合は、それらのファイルをフォルダーに配置し、ファイル コンテンツを使用して要求に応答できます。

C#

```
private async void _connection_RequestReceived(
    DevicePortalConnection sender,
    DevicePortalConnectionRequestReceivedEventArgs args)
{
    var storageFile = await
        Windows.ApplicationModel.Package.Current.InstalledLocation.GetFilesAsync(@"data\content
        .html");
    string htmlContent = await FileIO.ReadTextAsync(storageFile);

    args.ResponseMessage.Content = new HttpStringContent(
        htmlContent,
        Windows.Storage.Streams.UnicodeEncoding.Utf8,
        "text/html");
    args.ResponseMessage.StatusCode = HttpStatusCode.Ok;
}
```

アプリ サービスをシステムとして実行する

場合によっては、Device Portal プラグインの完全なテストのために、プラグインをシステム アカウントとして実行することが必要になる場合があります。 詳細については、[MSDN マガジン](#)

[の記事](#) を参照してください。

こちらも参照ください

- [Windows デバイス ポータルの概要](#)
- [アプリ機能宣言](#)
- [アプリサービスを作成および使用する](#)
- [Windows Device Portal のコア API リファレンス](#)

Last updated on 2026/07/07

Windows デバイス ポータル コア REST API リファレンス

Windows デバイス ポータル (WDP) のすべての機能は、REST API を基にして構築されています。開発者は REST API を直接呼び出して、プログラムからリソースにアクセスし、デバイスを制御することができます。

アプリの展開

アプリをインストールする

要求

次の要求形式を使用して、アプリをインストールできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/app/packagemanager/package

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
パッケージ	(必須) インストールするパッケージのファイル名。

要求ヘッダー

- None

要求本文

- .appx または .appxbundle ファイル、およびアプリが必要とする依存関係。
- デバイスが IoT または Windows デスクトップの場合、アプリの署名に使う証明書。その他のプラットフォームでは、証明書は必要ありません。

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	展開要求は受け入れられ、処理されています。
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

⚠ Note

Windows Mixed Reality は通常のデスクトップで実行されるため、Desktop と同じポータルです。

関連セットをインストールする

要求

次の要求形式を使用して、[関連セット](#)をインストールできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/app/packagemanager/package

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
パッケージ	(必須) インストールするパッケージのファイル名。

要求ヘッダー

- None

要求本文

- オプション パッケージをパラメーターとして指定するとき
は、"foo.appx.opt"、"bar.appxbundle.opt" などのようにパッケージのファイル名に
".opt" を追加します。
- .appx または .appxbundle ファイル、およびアプリが必要とする依存関係。
- デバイスが IoT または Windows デスクトップの場合、アプリの署名に使う証明書。そ
の他のプラットフォームでは、証明書は必要ありません。

応答

状態コード

この API では次の状態コードが返される可能性があります。

[\[\] テーブルを展開する](#)

HTTP 状態コード	説明
200	展開要求は受け入れられ、処理されています。
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

アプリをルース フォルダーに登録する

要求

次の要求形式を使用して、アプリをルース フォルダーに登録できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/app/package manager/networkapp

URI パラメーター

- None

要求ヘッダー

- None

要求本文

JSON

```
{
  "mainpackage" :
  {
    "networkshare" : "\\some\share\path",
    "username" : "optional_username",
    "password" : "optional_password"
  }
}
```

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	展開要求は受け入れられ、処理されています。
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ

- Xbox
- HoloLens
- IoT

関連セットをルース ファイル フォルダーに登録する

要求

次の要求形式を使用して、[関連セット](#)をルース フォルダーに登録できます。

 テーブルを展開する

認証方法	要求 URI
POST	/api/app/packagemanager/networkapp

URI パラメーター

- None

要求ヘッダー

- None

要求本文

JSON

```
{
  "mainpackage" :
  {
    "networkshare" : "\\some\share\path",
    "username" : "optional_username",
    "password" : "optional_password"
  },
  "optionalpackages" :
  [
    {
      "networkshare" : "\\some\share\path2",
      "username" : "optional_username2",
      "password" : "optional_password2"
    },
    ...
  ]
}
```

応答

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	展開要求は受け入れられ、処理されています。
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- Xbox
- HoloLens
- IoT

アプリのインストール状態を取得する

要求

次の要求形式を使用して、現在進行中のアプリのインストールの状態を取得できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/app/package manager/state

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	最後の展開の結果
204	インストールは実行中です
404	インストール操作は見つかりませんでした

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

アプリをアンインストールする

要求

次の要求形式を使用して、アプリをアンインストールできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/app/packagemanager/package

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメータ	説明
—	
パッケージ	(必須) ターゲット アプリの PackageFullName (GET /api/app/packagemanager/packages から)

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

インストールされたアプリを取得する

要求

次の要求形式を使用して、システムにインストールされているアプリの一覧を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/app/packagemanager/packages

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、インストールされているパッケージの一覧と関連する詳細情報が含まれます。この応答のテンプレートは次のとおりです。

JSON

```
{
  "InstalledPackages": [
    {
      "Name": string,
      "PackageFamilyName": string,
      "PackageFullName": string,
      "PackageOrigin": int,
      (https://msdn.microsoft.com/library/windows/desktop/dn313167(v=vs.85).aspx)
      "PackageRelativeId": string,
      "Publisher": string,
      "Version": {
        "Build": int,
        "Major": int,
        "Minor": int,
        "Revision": int
      },
      "RegisteredUsers": [
        {
          "UserDisplayName": string,
          "UserSID": string
        }, ...
      ], ...
    }, ...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード

HTTP 状態コード	説明
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

Bluetooth

コンピューターの Bluetooth 無線を取得する

要求

次の要求形式を使用して、コンピューターにインストールされている Bluetooth 無線の一覧を取得できます。これは、同じ JSON データを使用して、WebSocket 接続にアップグレードすることもできます。

[🔗](#) テーブルを展開する

認証方法	要求 URI
GET	/api/bt/getradios
GET/WebSocket	/api/bt/getradios

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、デバイスにアタッチされている Bluetooth 無線の JSON 配列が含まれます。

JSON

```
{ "BluetoothRadios" : [
  {
    "BluetoothAddress" : int64,
    "DisplayName" : string,
    "HasUnknownUsbDevice" : boolean,
    "HasProblem" : boolean,
    "ID" : string,
    "ProblemCode" : int,
    "State" : string
  }, ...
]}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

Bluetooth 無線をオンまたはオフにします。

要求

特定の Bluetooth 無線をオンまたはオフに設定します。

 テーブルを展開する

認証方法	要求 URI
POST	/api/bt/setradio

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメータ	説明
—	
ID	(必須) Bluetooth 無線のデバイス ID であり、base 64 でエンコードされている必要があります。
State	(必須) これは、 <code>"On"</code> または <code>"Off"</code> にすることができます。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

ペアリングした Bluetooth デバイスの一覧を取得する

要求

現在ペアリングされている Bluetooth デバイスの一覧を取得するには、次の要求形式を使用します。これは、同じ JSON データを使用して、WebSocket 接続にアップグレードすることもできます。WebSocket 接続の有効期間中、デバイスの一覧が変更される可能性があります。更新が行われるたびに、WebSocket 接続を介してデバイスの完全な一覧が送信されます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/bt/getpaired
GET/WebSocket	/api/bt/getpaired

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、現在ペアリングされている Bluetooth デバイスの JSON 配列が含まれます。

JSON

```
{
  "PairedDevices": [
    {
      "Name" : string,
      "ID" : string,
      "AudioConnectionStatus" : string
    }, ...
  ]
}
```

デバイスをこのシステムのオーディオに使用できる場合は、*AudioConnectionStatus* フィールドが表示されます (ポリシーとオプションのコンポーネントがこの影響を受ける可能性があります)。 *AudioConnectionStatus* は、 "Connected" または "Disconnected" のいずれかです。

利用可能な Bluetooth デバイスの一覧を取得する

要求

ペアリングに使用できる Bluetooth デバイスの一覧を取得するには、次の要求形式を使用します。これは、同じ JSON データを使用して、WebSocket 接続にアップグレードすることもできます。WebSocket 接続の有効期間中、デバイスの一覧が変更される可能性があります。更新が行われるたびに、WebSocket 接続を介してデバイスの完全な一覧が送信されます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/bt/getavailable
GET/WebSocket	/api/bt/getavailable

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、現在ペアリングに利用可能な Bluetooth デバイスの JSON 配列が含まれます。

JSON

```
{
  "AvailableDevices": [
    {
      "Name" : string,
      "ID" : string
    }, ...
  ]
}
```

Bluetooth デバイスを接続する

要求

デバイスをこのシステムのオーディオに使用できる場合、デバイスに接続します (ポリシーとオプションのコンポーネントがこの影響を受ける可能性があります)。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/bt/connectdevice

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメーター	説明
ID	(必須) Bluetooth デバイスのアソシエーション エンドポイント ID。Base64 でエンコードされている必要があります。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens

- IoT

Bluetooth デバイスを切断する

要求

デバイスをこのシステムのオーディオに使用できる場合は、デバイスを切断します (ポリシーとオプションのコンポーネントがこの影響を受ける可能性があります)。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/bt/disconnectdevice

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメーター	説明
ID	(必須) Bluetooth デバイスのアソシエーション エンドポイント ID。Base64 でエンコードされている必要があります。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

HTTP 状態コード	説明
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

デバイス マネージャー

コンピューターにインストールされているデバイスを取得する

要求

次の要求形式を使用して、コンピューターにインストールされているデバイスの一覧を取得できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/devicemanager/devices

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、デバイスにアタッチされているデバイスの JSON 配列が含まれます。

JSON

```
{ "DeviceList": [
  {
    "Class": string,
    "Description": string,
    "ID": string,
    "Manufacturer": string,
    "ParentID": string,
    "ProblemCode": int,
    "StatusCode": int
  }, ...
]}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- IoT

接続された USB デバイス/ハブのデータを取得する

要求

次の要求形式を使用して、接続された USB デバイスおよびハブの USB 記述子の一覧を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/ext/devices/usbdevices

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答は、ハブの USB 記述子およびポート情報と共に、USB デバイスのデバイス ID が含まれる JSON です。

JSON

```
{
  "DeviceList": [
    {
      "ID": string,
      "ParentID": string, // Will equal an "ID" within the list, or be blank
      "Description": string, // optional
      "Manufacturer": string, // optional
      "ProblemCode": int, // optional
      "StatusCode": int // optional
    },
    ...
  ]
}
```

返されるデータのサンプル

JSON

```
{
  "DeviceList": [{
    "ID": "System",
    "ParentID": ""
  }, {
    "Class": "USB",
    "Description": "Texas Instruments USB 3.0 xHCI Host Controller",
    "ID": "PCI\\VEN_104C&DEV_8241&SUBSYS_1589103C&REV_02\\4&37085792&0&00E7",
    "Manufacturer": "Texas Instruments",
    "ParentID": "System",
    "ProblemCode": 0,
    "StatusCode": 25174026
  }, {
    "Class": "USB",
```

```
"Description": "USB Composite Device",
"DeviceDriverKey": "{36fc9e60-c465-11cf-8056-444553540000}\\0016",
"ID": "USB\\VID_045E&PID_00DB\\8&2994096B&0&1",
"Manufacturer": "(Standard USB Host Controller)",
"ParentID": "USB\\VID_0557&PID_8021\\7&2E9A8711&0&4",
"ProblemCode": 0,
"StatusCode": 25182218
    }
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

ダンプの収集

アプリのすべてのクラッシュ ダンプの一覧を取得する

要求

次の要求形式を使用して、サイドローディングされたすべてのアプリについて、利用可能なすべてのクラッシュ ダンプの一覧を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/usermode/dumps

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、サイドローディングされたアプリケーションごとにクラッシュ ダンプの一覧が含まれます。

状態コード

この API では次の状態コードが返される可能性があります。

[\[\] テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)
- Windows デスクトップ
- HoloLens
- IoT

アプリのクラッシュ ダンプ収集設定を取得する

要求

次の要求形式を使用して、サイドローディングされたアプリのクラッシュ ダンプ収集設定を取得できます。

[\[\] テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/usermode/crashcontrol

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
packageFullName	(必須) サイドローディングされたアプリのパッケージの完全な名前。

要求ヘッダー

- None

要求本文

- None

応答

応答は、次の形式になります。

JSON

```
{"CrashDumpEnabled": bool}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)

- Windows デスクトップ
- HoloLens
- IoT

サイドローディングされたアプリのクラッシュ ダンプを削除する

要求

次の要求形式を使用して、サイドローディングされたアプリのクラッシュ ダンプを削除できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/debug/dump/usermode/crashdump

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
packageFullName	(必須) サイドローディングされたアプリのパッケージの完全な名前。
fileName	(必須) 削除する必要があるダンプ ファイルの名前。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)
- Windows デスクトップ
- HoloLens
- IoT

サイドローディングされたアプリのクラッシュ ダンプを無効にする

要求

次の要求形式を使用して、サイドローディングされたアプリのクラッシュ ダンプを無効にすることができます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/debug/dump/usermode/crashcontrol

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
packageFullName	(必須) サイドローディングされたアプリのパッケージの完全な名前。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)
- Windows デスクトップ
- HoloLens
- IoT

サイドローディングされたアプリのクラッシュ ダンプをダウンロードする

要求

次の要求形式を使用して、サイドローディングされたアプリのクラッシュ ダンプをダウンロードできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/usermode/crashdump

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
packageFullName	(必須) サイドローディングされたアプリのパッケージの完全な名前。
fileName	(必須) ダウンロードするダンプ ファイルの名前。

要求ヘッダー

- None

要求本文

- None

応答

応答には、ダンプ ファイルが含まれます。 WinDbg または Visual Studio を使用して、ダンプ ファイルを検証できます。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)
- Windows デスクトップ
- HoloLens
- IoT

サイドローディングされたアプリのクラッシュ ダンプを有効にする

要求

次の要求形式を使用して、サイドローディングされたアプリのクラッシュ ダンプを有効にすることができます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/debug/dump/usermode/crashcontrol

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
packageFullName	(必須) サイドローディングされたアプリのパッケージの完全な名前。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Window Mobile (Windows Insider Program のみ)
- Windows デスクトップ
- HoloLens
- IoT

バグチェック ファイルの一覧を取得する

要求

次の要求形式を使用して、バグチェックのミニダンプ ファイルの一覧を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/kernel/dumplist

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、ダンプ ファイル名とこれらのファイルのサイズの一覧が含まれます。一覧は、次の形式になります。

JSON

```
{
  "DumpFiles": [
    {
      "FileName": string,
      "FileSize": int
    }, ...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

バグチェックのダンプ ファイルをダウンロードする

要求

次の要求形式を使用して、バグチェックのダンプ ファイルをダウンロードできます。

[\[\] テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/kernel/dump

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[\[\] テーブルを展開する](#)

URI パラメーター	説明
<ファイル名>	(必須) ダンプ ファイルのファイル名。API を使ってダンプの一覧を取得することによって確認できます。

要求ヘッダー

- None

要求本文

- None

応答

応答には、ダンプ ファイルが含まれます。WinDbg を使用してこのファイルを調べることができます。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

バグチェックのクラッシュ制御の設定を取得する

要求

次の要求形式を使用して、バグチェックのクラッシュ制御の設定を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/kernel/crashcontrol

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、クラッシュの制御の設定が含まれます。CrashControl について詳しくは、「[CrashControl](#)」をご覧ください。応答のテンプレートは次のとおりです。

JSON

```
{
  "autoreboot": bool (0 or 1),
  "dumptype": int (0 to 4),
  "maxdumpcount": int,
  "overwrite": bool (0 or 1)
}
```

ダンプの種類

0: 無効

1:完全なメモリ ダンプ (使用中のすべてのメモリを収集)

2:カーネル メモリ ダンプ (ユーザー モード メモリは無視)

3:制限付きカーネル ミニダンプ

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

ライブ カーネル ダンプを取得する

要求

次の要求形式を使用して、ライブ カーネル ダンプを取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/livekernel

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、カーネル モードの完全なダンプが含まれます。 WinDbg を使用してこのファイルを調べることができます。

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

ライブ ユーザー プロセスからダンプを取得する

要求

次の要求形式を使用して、ライブ ユーザー プロセスのダンプを取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/debug/dump/usermode/live

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
pid	(必須) 目的のプロセスの一意のプロセス ID。

要求ヘッダー

- None

要求本文

- None

応答

応答には、プロセス ダンプが含まれます。 WinDbg または Visual Studio を使用してこのファイルを調べることができます。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

バグチェックのクラッシュ制御の設定を行う

要求

次の要求形式を使用して、バグチェック データの収集に関する設定を行うことができます。

[\[\] テーブルを展開する](#)

認証方法	要求 URI
POST	/api/debug/dump/kernel/crashcontrol

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[\[\] テーブルを展開する](#)

URI パラメータ	説明
autoreboot	(オプション) true または false。これは、エラーやロックの発生後に、システムが自動的に再起動するかどうかを示します。
dumptype	(オプション) dump タイプ。サポートされる値については、「 CrashDumpType 列挙体 」をご覧ください。
maxdumpcount	(オプション) 保存するダンプの最大数。
overwrite	(オプション) true または false。これは、 <i>maxdumpcount</i> で指定されているダンプ カウンターの制限に達した場合に古いダンプを上書きするかどうかを示します。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[\[\] テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

ETW

websocket 経由でリアルタイムの ETW セッションを作成する

要求

次の要求形式を使用して、リアルタイムの ETW セッションを作成できます。これは、websocket 経由で管理されます。ETW イベントは、サーバーで一括処理され、1 秒に 1 回クライアントに送信されます。

[🔗](#) テーブルを展開する

認証方法	要求 URI
GET/WebSocket	/api/etw/session/realtime

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、有効なプロバイダーの ETW イベントが含まれます。以下の「ETW WebSocket コマンド」をご覧ください。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

ETW WebSocket コマンド

次のコマンドは、クライアントからサーバーに送信されます。

[🔗 テーブルを展開する](#)

コマンド	説明
provider {guid} enable {level}	{guid} で指定されたプロバイダー (括弧は不要) を指定されたレベルで有効にします。{level} は、1 (最小限の詳細) ~ 5 (詳細) の int です。
provider {guid} disable	{guid} で指定されたプロバイダー (括弧は不要) を無効にします。

この応答は、サーバーからクライアントに送信されます。これは、テキストとして送信され、JSON で解析すると次の形式になります。

JSON

```
{
  "Events": [
    {
      "Timestamp": int,
```



```

        "ProviderName": string,
        "ID": int,
        "TaskName": string,
        "Keyword": int,
        "Level": int,
        payload objects...
    },...
],
"Frequency": int
}

```

payload objects は、追加のキーと値のペア (文字列: 文字列) で、元の ETW イベントから提供されます。

例:

JSON

```

{
  "ID" : 42,
  "Keyword" : 9223372036854775824,
  "Level" : 4,
  "Message" : "UDPv4: 412 bytes transmitted from 10.81.128.148:510 to 132.215.243.34:510. ",
  "PID" : "1218",
  "ProviderName" : "Microsoft-Windows-Kernel-Network",
  "TaskName" : "KERNEL_NETWORK_TASK_UDPIP",
  "Timestamp" : 131039401761757686,
  "connid" : "0",
  "daddr" : "132.245.243.34",
  "dport" : "500",
  "saddr" : "10.82.128.118",
  "seqnum" : "0",
  "size" : "412",
  "sport" : "500"
}

```

登録済みの ETW プロバイダーを列挙する

要求

次の要求形式を使用して、登録済みプロバイダーを列挙できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/etw/providers

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、ETW プロバイダーの一覧が含まれます。一覧には、各プロバイダーのフレンドリ名と GUID が次の形式で含まれます。

JSON

```
{  
  "Providers": [  
    {  
      "GUID": string, (GUID)  
      "Name": string  
    }, ...  
  ]  
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗](#) テーブルを展開する

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

プラットフォームによって公開されているカスタム ETW プロバイダーを列挙します。

要求

次の要求形式を使用して、登録済みプロバイダーを列挙できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/etw/customproviders

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

200 OK。 応答には、ETW プロバイダーの一覧が含まれます。 一覧には、各プロバイダーのフレンドリ名と GUID が含まれます。

JSON

```
{ "Providers": [
  {
    "GUID": string, (GUID)
    "Name": string
  }, ...
]}
```

状態コード

- 標準の状態コード。

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

インストール先

場所の上書きモードを取得する

要求

次の要求型式を使用して、デバイスの場所スタック上書き状態を取得できます。 この呼び出しを成功させるには、開発者モードを有効にしておく必要があります。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/ext/location/override

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、デバイスの上書き状態が次の形式で含まれます。

JSON
<pre>{"Override" : bool}</pre>

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

HTTP 状態コード	説明
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

場所の上書きモードを設定する

要求

次の要求型式を使用して、デバイスの場所スタック上書き状態を設定できます。有効になっている場合は、場所スタックによって位置挿入が許可されます。この呼び出しを成功させるには、開発者モードを有効にしておく必要があります。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
PUT	/ext/location/override

URI パラメーター

- None

要求ヘッダー

- None

要求本文

JSON

```
{"Override" : bool}
```

応答

応答には、デバイスに設定されている上書き状態が次の形式で含まれます。

JSON

```
{"Override" : bool}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

挿入された位置を取得する

要求

次の要求型式を使用して、デバイスの挿入 (スプーフィング) された場所を取得できます。挿入された場所を設定する必要があります。設定されなかった場合は、エラーがスローされます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/ext/location/position

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、現在の挿入された緯度と経度の値が次の形式で含まれます。

JSON

```
{
  "Latitude" : double,
  "Longitude" : double
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[\[\] テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

挿入された位置を設定する

要求

次の要求型式を使用して、デバイスの挿入 (スプーフィング) された場所を設定できます。あらかじめデバイス上で場所の上書きモードが有効になっており、設定される場所も有効である必要があります。それ以外の場合はエラーがスローされます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
PUT	/ext/location/override

URI パラメーター

- None

要求ヘッダー

- None

要求本文

JSON

```
{
  "Latitude" : double,
  "Longitude" : double
}
```

応答

応答には、設定された場所の情報が次の形式で含まれます。

JSON

```
{
  "Latitude" : double,
  "Longitude" : double
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

OS 情報

コンピューター名を取得する

要求

次の要求形式を使用して、コンピューターの名前を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/os/machinename

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、コンピューター名が次の形式で含まれます。

JSON

```
{"ComputerName": string}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

オペレーティング システムの情報を取得する

要求

次の要求形式を使用して、コンピューターの OS 情報を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/os/info

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、OS 情報が次の形式で含まれます。

JSON

```
{
  "ComputerName": string,
  "OsEdition": string,
  "OsEditionId": int,
  "OsVersion": string,
  "Platform": string
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

デバイス ファミリを取得する

要求

次の要求形式を使用して、デバイス ファミリ (Xbox、携帯電話、デスクトップなど) を取得できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/os/devicefamily

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、デバイス ファミリ (SKU - デスクトップ、Xbox など) が含まれます。

JSON

```
{
  "DeviceType" : string
}
```

DeviceType は、"Windows.Xbox"、"Windows.Desktop" などのようになります。

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens

- IoT

コンピューター名を設定する

要求

次の要求形式を使用して、コンピューターの名前を設定できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/os/machinename

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメータ	説明
—	
name	(必須) コンピューターの新しい名前。 base64 でエンコードされている必要があります。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

ユーザー情報

アクティブ ユーザーを取得する

要求

次の要求形式を使用して、デバイスのアクティブ ユーザーの名前を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/users/activeuser

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、ユーザー情報が次の形式で含まれます。

成功した場合:

JSON

```
{
  "UserDisplayName" : string,
  "UserSID" : string
}
```

失敗した場合:

JSON

```
{
  "Code" : int,
  "CodeText" : string,
  "Reason" : string,
  "Success" : bool
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

パフォーマンス データ

実行中のプロセスの一覧を取得する

要求

次の要求形式を使用して、現在実行中のプロセスの一覧を取得できます。これは、WebSocket 接続にアップグレードすることもでき、1 秒に 1 度クライアントにプッシュされる同じ JSON データを取得できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/resourcemanager/processes
GET/WebSocket	/api/resourcemanager/processes

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、プロセスの一覧と各プロセスの詳細情報が含まれます。情報は JSON 形式で、テンプレートは次のとおりです。

JSON

```
{
  "Processes": [
    {
      "CPUUsage": float,
      "ImageName": string,
      "PageFileUsage": long,
      "PrivateWorkingSet": long,
      "ProcessId": int,
      "SessionId": int,
      "UserName": string,
      "VirtualSize": long,
      "WorkingSetSize": long
    }, ...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

システム パフォーマンスの統計情報を取得する

要求

次の要求形式を使用して、システム パフォーマンスの統計情報を取得できます。これには、読み取りと書き込みのサイクルや、使用されているメモリの量などの情報が含まれます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/resourcemanager/systemperf
GET/WebSocket	/api/resourcemanager/systemperf

これは、WebSocket 接続にアップグレードできます。1 秒に 1 度以下と同じ JSON データが提供されます。

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、CPU と GPU の使用量、メモリ アクセス、ネットワーク アクセスなど、パフォーマンスの統計情報が含まれます。 この情報は JSON 形式で、テンプレートは次のとおりです。

JSON

```
{
  "AvailablePages": int,
  "CommitLimit": int,
  "CommittedPages": int,
  "CpuLoad": int,
  "IOOtherSpeed": int,
  "IOReadSpeed": int,
  "IOWriteSpeed": int,
  "NonPagedPoolPages": int,
  "PageSize": int,
  "PagedPoolPages": int,
  "TotalInstalledInKb": int,
  "TotalPages": int,
  "GPUData":
  {
    "AvailableAdapters": [{ (One per detected adapter)
      "DedicatedMemory": int,
      "DedicatedMemoryUsed": int,
      "Description": string,
      "SystemMemory": int,
      "SystemMemoryUsed": int,
      "EnginesUtilization": [ float,... (One per detected engine)]
    }],...
  }},
  "NetworkingData": {
    "NetworkInBytes": int,
    "NetworkOutBytes": int
  }
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

Power

現在のバッテリー状態を取得する

要求

次の要求形式を使用して、バッテリーの現在の状態を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/battery

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

現在のバッテリー状態に関する情報が次の形式で返されます。

JSON

```
{
  "AcOnline": int (0 | 1),
  "BatteryPresent": int (0 | 1),
```

```
"Charging": int (0 | 1),
"DefaultAlert1": int,
"DefaultAlert2": int,
"EstimatedTime": int,
"MaximumCapacity": int,
"RemainingCapacity": int
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

アクティブな電源設定を取得する

要求

次の要求形式を使用して、アクティブな電源設定を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/activecfg

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

アクティブな電源設定の形式は、次のとおりです。

JSON

```
{"ActivePowerScheme": string (guid of scheme)}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

電源設定のサブ値を取得する

要求

次の要求形式を使用して、電源設定のサブ値を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/cfg/<電源設定のパス>

オプション:

- SCHEME_CURRENT

URI パラメーター

- None

要求ヘッダー

- None

要求本文

利用可能な電源状態の完全な一覧は、アプリケーションごとが基本で、バッテリー低下、重要なバッテリーといったさまざまな電源状態がフラグ設定されています。

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

システムの電源状態を取得する

要求

次の要求形式を使用して、システムの電源状態を確認できます。 これによって、低電力状態になっているかどうかを確認できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/state

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

電源状態の情報のテンプレートは次のとおりです。

JSON

```
{"LowPowerState" : false, "LowPowerStateAvailable" : true }
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

アクティブな電源設定を行う

要求

次の要求形式を使用して、アクティブな電源設定を設定できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/power/activecfg

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
scheme	(必須) システムのアクティブな電源設定として設定するスキームの GUID。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

電源設定のサブ値を設定する

要求

次の要求形式を使用して、電源設定のサブ値を設定できます。

[\[\] テーブルを展開する](#)

認証方法	要求 URI
POST	/api/power/cfg/<電源設定のパス>

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[\[\] テーブルを展開する](#)

URI パラメーター	説明
valueAC	(必須) AC 電源に使用する値。
valueDC	(必須) バッテリー電源に使用する値。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[\[\] テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

SleepStudy レポートを取得する

要求

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/sleepstudy/report

次の要求形式を使用して、SleepStudy レポートを取得できます。

URI パラメーター | URI パラメーター | 説明 | |:----- |:----- || FileName | **(必須)** ダウンロードするファイルのファイル名。この値は、hex64 エンコードされている必要があります。 |

要求ヘッダー

- None

要求本文

- None

応答

応答は、スリープ検査の結果が含まれているファイルです。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

利用可能な SleepStudy レポートを列挙する

要求

次の要求形式を使用して、利用可能な SleepStudy レポートを列挙できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/power/sleepstudy/reports

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

利用可能なレポートの一覧のテンプレートは次のとおりです。

JSON

```
{"Reports": [  
  {  
    "FileName": string  
  },...  
]}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

スリープ スタディ変換を取得する

要求

次の要求形式を使用して、スリープ スタディ変換を取得できます。 この変換は、SleepStudy レポートを、ユーザーが読み取ることができる XML 形式に変換する XSLT です。

 テーブルを展開する

認証方法	要求 URI
GET	/api/power/sleepstudy/transform

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、スリープ スタディ変換が含まれます。

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- IoT

リモート コントロール

ターゲット コンピューターを再起動する

要求

次の要求形式を使用して、ターゲット コンピューターを再起動できます。

 テーブルを展開する

認証方法	要求 URI
POST	/api/control/restart

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

ターゲット コンピューターをシャットダウンする

要求

次の要求形式を使用して、ターゲット コンピューターをシャット ダウンできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/control/shutdown

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

タスク マネージャー

最新のアプリを起動する

要求

次の要求形式を使用して、最新のアプリを起動できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/taskmanager/app

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
appid	(必須) 起動するアプリの PRAID。この値は、hex64 エンコードされている必要があります。

URI パラメーター	説明
パッケージ	(必須) 起動するアプリ パッケージの完全な名前。 この値は、hex64 エンコードされている必要があります。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

最新のアプリを停止する

要求

次の要求形式を使用して、最新のアプリを停止できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/taskmanager/app

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメータ	説明
—	
パッケージ	(必須) 停止するアプリ パッケージの完全な名前。 この値は、hex64 エンコードされている必要があります。
forcestop	(オプション) 値が yes の場合は、システムがすべてのプロセスを強制的に停止することを示します。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox

- HoloLens
- IoT

PID でプロセスを強制終了する

要求

次の要求形式を使用して、プロセスを強制終了できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/taskmanager/process

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
pid	(必須) 終了するプロセスの一意的プロセス ID。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

HTTP 状態コード	説明
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

ネットワーク

現在の IP 構成を取得する

要求

次の要求形式を使用して、現在の IP 構成を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/networking/ipconfig

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

応答には、次のテンプレートの IP 構成が含まれます。

JSON

```

{"Adapters": [
  {
    "Description": string,
    "HardwareAddress": string,
    "Index": int,
    "Name": string,
    "Type": string,
    "DHCP": {
      "LeaseExpires": int, (timestamp)
      "LeaseObtained": int, (timestamp)
      "Address": {
        "IpAddress": string,
        "Mask": string
      }
    },
    "WINS": {(WINS is optional)
      "Primary": {
        "IpAddress": string,
        "Mask": string
      },
      "Secondary": {
        "IpAddress": string,
        "Mask": string
      }
    },
    "Gateways": [{ (always 1+)
      "IpAddress": "10.82.128.1",
      "Mask": "255.255.255.255"
    },...
  ],
  "IpAddresses": [{ (always 1+)
    "IpAddress": "10.82.128.148",
    "Mask": "255.255.255.0"
  },...
]
},...
]}

```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

静的 IP アドレスを設定する (IPV4 構成)

要求

静的 IP と DNS を使用して IPV4 構成を設定します。静的 IP を指定しない場合、DHCP が有効になります。静的 IP を指定する場合、DNS も指定する必要があります。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
PUT	/api/networking/ipv4config

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメーター	説明
AdapterName	(必須) ネットワーク インターフェイスの GUID。
IPAddress	設定する静的 IP アドレス。
SubnetMask	(IPAddress が null でない場合は必須) 静的サブネット マスク。
DefaultGateway	(IPAddress が null でない場合は必須) 既定の静的ゲートウェイ。
PrimaryDNS	(IPAddress が null でない場合は必須) 設定する静的プライマリ DNS。
SecondayDNS	(PrimaryDNS が null でない場合は必須) 設定する静的セカンダリ DNS。

わかりやすくするために、インターフェイスを DHCP に設定するには、ネットワーク上の AdapterName のみをシリアル化します。

JSON

```
{
  "AdapterName": "{82F86C1B-2BAE-41E3-B08D-786CA44FEED7}"
}
```

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

ワイヤレス ネットワーク インターフェイスを列挙する

要求

次の要求形式を使用して、利用可能なワイヤレス ネットワーク インターフェイスを列挙できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wifi/interfaces

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

次の形式の利用可能なワイヤレス インターフェイスと詳細の一覧。

JSON

```
{
  "Interfaces": [
    {
      "Description": string,
      "GUID": string (guid with curly brackets),
      "Index": int,
      "ProfilesList": [
        {
          "GroupPolicyProfile": bool,
          "Name": string, (Network currently connected to)
          "PerUserProfile": bool
        },
        ...
      ]
    },
    ...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

ワイヤレス ネットワークを列挙する

要求

次の要求形式を使用して、指定されたインターフェイスのワイヤレス ネットワークの一覧を列挙できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wifi/networks

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
インターフェイス	(必須) ワイヤレス ネットワークの検索に使用するネットワーク インターフェイスの GUID (括弧は不要)。

要求ヘッダー

- None

要求本文

- None

応答

指定されたインターフェイスで見つかったワイヤレス ネットワークの一覧。これには、ネットワークの詳細が次の形式で含まれます。

JSON

```
{
  "AvailableNetworks": [
    {
      "AlreadyConnected": bool,
      "AuthenticationAlgorithm": string, (WPA2, etc)
      "Channel": int,
      "CipherAlgorithm": string, (for example, AES)
      "Connectable": int, (0 | 1)
      "InfrastructureType": string,
      "ProfileAvailable": bool,
      "ProfileName": string,
      "SSID": string,
      "SecurityEnabled": int, (0 | 1)
      "SignalQuality": int,
      "BSSID": [int,...],
      "PhysicalTypes": [string,...]
    },...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

Wi-Fi ネットワークを接続および切断する

要求

次の要求形式を使用して、Wi-Fi ネットワークを接続および切断できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/wifi/network

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
インターフェイス	(必須) ネットワークへの接続に使用するネットワーク インターフェイスの GUID。
op	(必須) 実行するアクションを示します。 設定可能な値は、connect または disconnect です。
ssid	(<i>op</i> == connect の場合は必須) 接続先 SSID。
key	(<i>op</i> == connect で、ネットワークで認証が必要な場合は必須) 共有キー。
createprofile	(必要) デバイスでネットワークのプロファイルを作成します。 これにより、今後、デバイスはネットワークに自動接続されます。 yes または no を指定できます。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

Wi-Fi のプロフィールを削除する

要求

次の要求形式を使用して、特定のインターフェイス上のネットワークに関連付けられたプロフィールを削除できます。

[\[\] テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/wifi/profile

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[\[\] テーブルを展開する](#)

URI パラメータ	説明
—	
インターフェイス	(必須) 削除するプロフィールに関連付けられたネットワーク インターフェイスの GUID。
プロフィール	(必須) 削除するプロフィールの名前。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

Windows エラー報告 (WER)

Windows エラー報告 (WER) ファイルをダウンロードする

要求

次の要求形式を使用して、WER 関連のファイルをダウンロードできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wer/report/file

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
ユーザー	(必須) レポートに関連付けられたユーザー名。
型	(必須) レポートの種類。 これは queried または archived のいずれかになります。

URI パラメーター	説明
name	(必須) レポートの名前。 base64 でエンコードされている必要があります。
ファイル	(必須) レポートからダウンロードするファイルの名前。 base64 でエンコードされている必要があります。

要求ヘッダー

- None

要求本文

- None

応答

- 応答には、要求したファイルが含まれています。

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

Windows エラー報告 (WER) レポート内のファイルを列挙する

要求

次の要求形式を使用して、WER レポート内のファイルを列挙できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wer/report/files

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
ユーザー	(必須) レポートに関連付けられたユーザー。
型	(必須) レポートの種類。これは queried または archived のいずれかになります。
name	(必須) レポートの名前。 base64 でエンコードされている必要があります。

要求ヘッダー

- None

要求本文

JSON

```
{
  "Files": [
    {
      "Name": string, (Filename, not base64 encoded)
      "Size": int (bytes)
    }, ...
  ]
}
```

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード

HTTP 状態コード	説明
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

Windows エラー報告 (WER) レポートを一覧表示する

要求

次の要求形式を使用して、WER レポートを取得できます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/wer/reports

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

WER 報告の形式は次のとおりです。

JSON

```
{
  "WerReports": [
    {
      "User": string,
      "Reports": [
        {
          "CreationTime": int,
```

```
        "Name": string, (not base64 encoded)
        "Type": string ("Queue" or "Archive")
    },
    ...
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows デスクトップ
- HoloLens
- IoT

Windows Performance Recorder (WPR)

カスタム プロファイルを使用してトレースを開始する

要求

次の要求形式を使用して、WPR プロファイルをアップロードし、そのプロファイルを使用してトレースを開始できます。一度に実行できるトレースは 1 つのみです。プロファイルはデバイス上に残りません。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/wpr/customtrace

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- カスタム WPR プロファイルが含まれる、原則に従ったマルチパートの http 本文。

応答

WPR セッション状態の形式は次のとおりです。

JSON

```
{
  "SessionType": string, (Running or Idle)
  "State": string (normal or boot)
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

起動パフォーマンス トレース セッションを開始する

要求

次の要求形式を使用して、WPR の起動トレース セッションを開始できます。これは、パフォーマンストレース セッションとも呼びます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/wpr/boottrace

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
プロファイル	(必須) このパラメーターは起動時に必要です。パフォーマンストレース セッションを開始する必要があるプロファイルの名前。指定可能なプロファイルは、perfprofiles/profiles.json に格納されています。

要求ヘッダー

- None

要求本文

- None

応答

この API は、開始時に次の形式で WPR セッション状態を返します。

JSON
<pre>{ "SessionType": string, (Running or Idle) "State": string (boot) }</pre>

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

起動パフォーマンス トレース セッションを停止する

要求

次の要求形式を使用して、WPR の起動トレース セッションを停止できます。これは、パフォーマンス トレース セッションとも呼びます。

 テーブルを展開する

認証方法	要求 URI
GET	/api/wpr/boottrace

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

- なし。 **注:** これは時間のかかる操作です。 ETL のディスクへの書き込みが終了すると、制御が戻ります。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

パフォーマンス トレース セッションを開始する

要求

次の要求形式を使用して、WPR のトレース セッションを開始できます。これは、パフォーマンス トレース セッションとも呼びます。一度に実行できるトレースは 1 つのみです。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/wpr/trace

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
プロファイル	(必須) パフォーマンス トレース セッションを開始する必要があるプロファイルの名前。指定可能なプロファイルは、perfprofiles/profiles.json に格納されています。

要求ヘッダー

- None

要求本文

- None

応答

この API は、開始時に次の形式で WPR セッション状態を返します。

JSON

```
{
  "SessionType": string, (Running or Idle)
  "State": string (normal)
}
```

状態コード

この API では次の状態コードが返される可能性があります。

 テーブルを展開する

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

パフォーマンスのトレース セッションを停止する

要求

次の要求形式を使用して、WPR のトレース セッションを停止できます。これは、パフォーマンス トレース セッションとも呼びます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wpr/trace

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

- なし。 **注:** これは時間のかかる操作です。 ETL のディスクへの書き込みが終了すると、制御が戻ります。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

トレース セッションの状態を取得する

要求

次の要求形式を使用して、現在の WPR セッションの状態を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wpr/status

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

WPR トレース セッションの状態の形式は次のとおりです。

JSON

```
{
  "SessionType": string, (Running or Idle)
  "State": string (normal or boot)
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

完了したトレース セッション (ETL) を一覧表示する

要求

次の要求形式を使用して、デバイス上の ETL トレースの一覧を取得できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wpr/tracefiles

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

完了したトレース セッションの一覧は次の形式で提供されます。

JSON

```
{
  "Items": [
    {
      "CurrentDir": string (filepath),
      "DateCreated": int (File CreationTime),
      "FileSize": int (bytes),
      "Id": string (filename),
      "Name": string (filename),
      "SubPath": string (filepath),
      "Type": int
    }
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

トレース セッション (ETL) をダウンロードする

要求

次の要求形式を使用して、トレースファイル (ブート トレースまたはユーザー モード トレース) をダウンロードできます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/wpr/tracefile

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメータ	説明
—	
<ファイル名>	(必須) ダウンロードする ETL トレースの名前。 これらは /api/wpr/tracefiles にあります。

要求ヘッダー

- None

要求本文

- None

応答

- トレース ETL ファイルを返します。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

トレース セッション (ETL) を削除する

要求

次の要求形式を使用して、トレースファイル (ブート トレースまたはユーザー モード トレース) を削除できます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/wpr/tracefile

URI パラメーター

次の追加パラメーターを要求 URI に指定できます。

[🔗 テーブルを展開する](#)

URI パラメーター	説明
<ファイル名>	(必須) 削除する ETL トレースの名前。これらは /api/wpr/tracefiles にあります。

要求ヘッダー

- None

要求本文

- None

応答

- トレース ETL ファイルを返します。

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- IoT

DNS SD タグ

タグを表示する

要求

デバイスに現在適用されているタグを表示します。これらのタグは、T キー内の DNS-SD TXT レコードを使用してアドバタイズされます。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/dns-sd/tags

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答 現在適用されているタグの形式は次のとおりです。

JSON

```
{
  "tags": [
    "tag1",
    "tag2",
    ...
  ]
}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
5XX	サーバー エラー

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

タグを削除する

要求

DNS-SD によって現在アドバタイズされているすべてのタグを削除します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/dns-sd/tags

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答

- None

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

HTTP 状態コード	説明
5XX	サーバー エラー

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

タグを削除する

要求

DNS-SD によって現在アドバタイズされている 1 つのタグを削除します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/dns-sd/tag

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメーター	説明
tagValue	(必須) 削除するタグ。

要求ヘッダー

- None

要求本文

- None

応答

- None

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

タグを追加する

要求

DNS-SD アドバタイズにタグを追加します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/dns-sd/tag

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメーター	説明
tagValue	(必要) 追加するタグ。

要求ヘッダー

- None

要求本文

- None

応答

- None

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
401	タグ領域のオーバーフロー。提供されたタグが、結果として生成される DNS-SD サービス レコードに対して長すぎます。

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- Xbox
- HoloLens
- IoT

アプリのエクスプローラー

既知のフォルダーを取得する

要求

アクセス可能なトップ レベル フォルダーの一覧を取得します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/filesystem/apps/knownfolders

URI パラメーター

- None

要求ヘッダー

- None

要求本文

- None

応答 利用可能なフォルダーの形式は次のとおりです。

JSON

```
{ "KnownFolders": [
  "folder0",
  "folder1", ...
]}
```

状態コード

この API では次の状態コードが返される可能性があります。

 **テーブルを展開する**

HTTP 状態コード	説明
200	展開要求は受け入れられ、処理されています。
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox
- IoT

ファイルを取得する

要求

フォルダー内のファイルの一覧を取得します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/filesystem/apps/files

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメータ	説明
knownfolderid	(必須) 必要なファイルの一覧の対象となるトップレベル ディレクトリ。サイドロードされたアプリにアクセスするには、 LocalAppData を使用します。
packagefullname	(knownfolderid == LocalAppData の場合は必須) 対象となるアプリのパッケージのフルネーム。
パス	(オプション) 上で指定したフォルダーまたはパッケージ内のサブディレクトリ。

要求ヘッダー

- None

要求本文

- None

応答 利用可能なフォルダーの形式は次のとおりです。

JSON

```
{"Items": [  
  {  
    "CurrentDir": string (folder under the requested known folder),  
    "DateCreated": int,  
    "FileSize": int (bytes),  
    "Id": string,  
    "Name": string,  
    "SubPath": string (present if this item is a folder, this is the name of the  
folder),  
    "Type": int  
  },...  
]}
```

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK
4XX	エラー コード
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox
- IoT

ファイルをダウンロードする

要求

既知のフォルダーまたは appLocalData からファイルを取得します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
GET	/api/filesystem/apps/file

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメータ	説明
knownfolderid	(必須) ファイルをダウンロードするトップレベルディレクトリ。サイドロードされたアプリにアクセスするには、 LocalAppData を使用します。
<ファイル名>	(必須) ダウンロードするファイルの名前。
packagefullname	(knownfolderid == LocalAppData の場合は必須) 対象となるパッケージのフルネーム。
パス	(オプション) 上で指定したフォルダーまたはパッケージ内のサブディレクトリ。

要求ヘッダー

- None

要求本文

- 要求するファイル (存在する場合)

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	要求したファイル
404	ファイルが見つからない
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox
- IoT

ファイルの名前の変更

要求

フォルダー内のファイルの名前を変更します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/filesystem/apps/rename

URI パラメーター

[🔍 テーブルを展開する](#)

URI パラメータ	説明
—	
knownfolderid	(必須) ファイルが存在するトップレベル ディレクトリ。 サイドロードされたアプリにアクセスするには、 LocalAppData を使用します。
<ファイル名>	(必須) 名前を変更するファイルの元の名前。
newfilename	(必須) ファイルの新しい名前。
packagefullname	(<i>knownfolderid</i> == LocalAppData の場合は必須) 対象となるアプリのパッケージのフルネーム。
パス	(オプション) 上で指定したフォルダーまたはパッケージ内のサブディレクトリ。

要求ヘッダー

- None

要求本文

- None

応答

- None

状態コード

この API では次の状態コードが返される可能性があります。

[🔍 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK です。 ファイルの名前が変更されました
404	ファイルが見つからない
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox

- IoT

ファイルを削除する

要求

フォルダー内のファイルを削除します。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
DELETE	/api/filesystem/apps/file

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメータ	説明
knownfolderid	(必須) ファイルを削除するトップレベルディレクトリ。サイドロードされたアプリにアクセスするには、 LocalAppData を使用します。
<ファイル名>	(必須) 削除するファイルの名前。
packagefullname	(<i>knownfolderid</i> == LocalAppData の場合は 必須) 対象となるアプリのパッケージのフルネーム。
パス	(オプション) 上で指定したフォルダーまたはパッケージ内のサブディレクトリ。

要求ヘッダー

- None

要求本文

- None

応答

- None

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK です。 ファイルが削除されます。
404	ファイルが見つからない
5XX	エラー コード

利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox
- IoT

ファイルをアップロードする

要求

フォルダーにファイルをアップロードします。 この場合、同じ名前を持つ既存のファイルは上書きされますが、新しいフォルダーは作成されません。

[🔗 テーブルを展開する](#)

認証方法	要求 URI
POST	/api/filesystem/apps/file

URI パラメーター

[🔗 テーブルを展開する](#)

URI パラメータ	説明
knownfolderid	(必須) ファイルをアップロードするトップレベルディレクトリ。 サイドロードされたアプリにアクセスするには、LocalAppData を使用します。
packagefullname	(<i>knownfolderid</i> == LocalAppData の場合は必須) 対象となるアプリのパッケージのフルネーム。
extract	(必須) true または false これは、アップロード後にファイルを抽出するかどうかを示

URI パラメータ	説明
—	
	します。
パス	(オプション) 上で指定したフォルダーまたはパッケージ内のサブディレクトリ。

要求ヘッダー

- None

要求本文

- None

応答

状態コード

この API では次の状態コードが返される可能性があります。

[🔗 テーブルを展開する](#)

HTTP 状態コード	説明
200	OK です。 ファイルがアップロードされます。
4XX	エラー コード
5XX	エラー コード

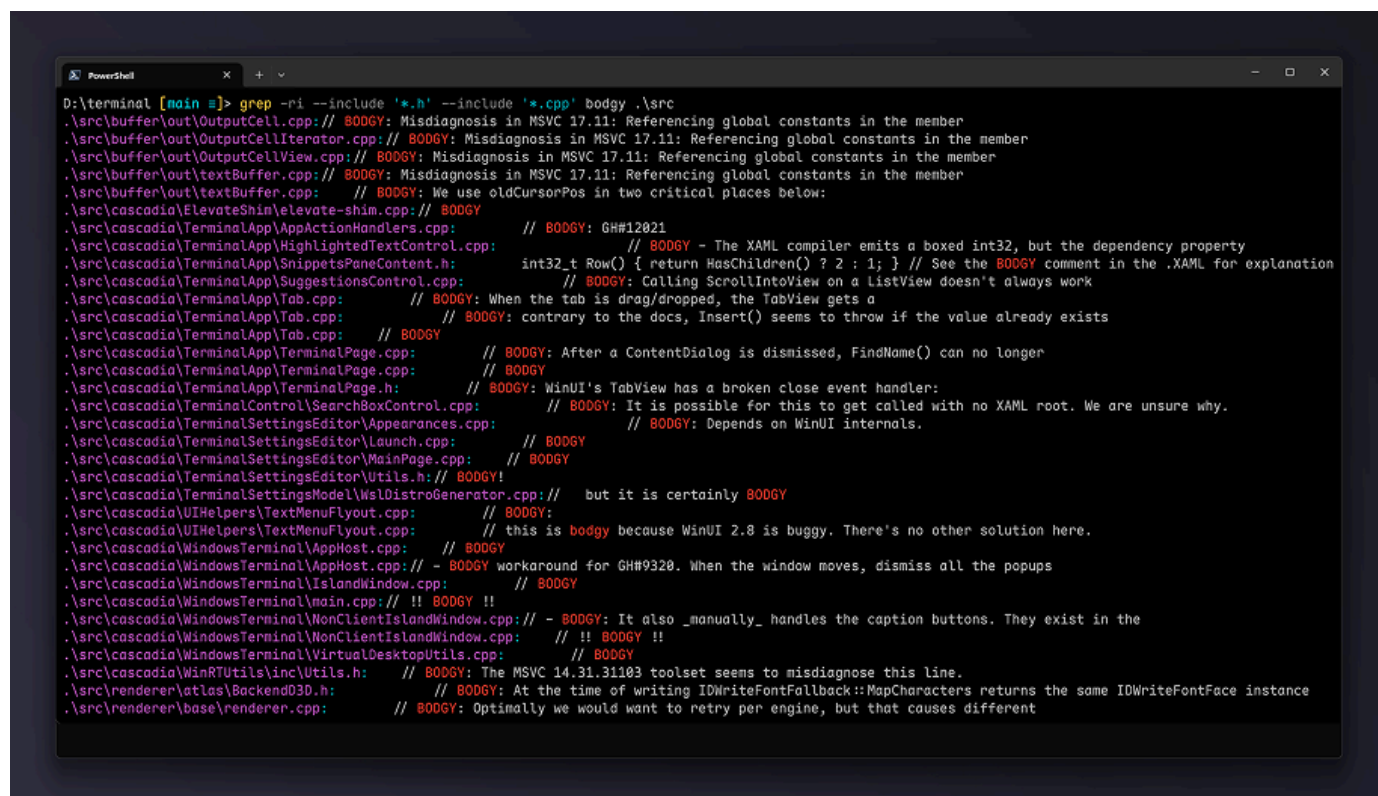
利用可能なデバイス ファミリ

- Windows Mobile
- Windows デスクトップ
- HoloLens
- Xbox
- IoT

Windowsの Coreutils

[Coreutils for Windows](#) は、linux、macOS、WSL で使用するのと同じコマンドとパイプラインをWindowsでネイティブに実行する、MICROSOFTで保守される UNIX スタイルのコマンドラインユーティリティのセットです。これは、標準名 (`cat.exe`、`grep.exe`、`find.exe` など) で各ユーティリティを公開する単一のマルチコール バイナリとして付属しており、開発者が他のプラットフォームで既にテキストのスクリプト作成、自動化、処理に使用している日常的なツールを提供します。完全な一覧については、「[コマンド](#)」を参照してください。

目標は、Linux、macOS、WSL、コンテナ、およびWindows間を移動するときの摩擦を解消することです。同じコマンド、フラグ、パイプラインが同じように動作するため、既存のスクリプトや習慣は翻訳なしで引き継がれます。各コマンドは、完全な構文とオプションの標準 `--help` フラグをサポートしています。



```
D:\terminal [main =>] grep -ri --include '*.h' --include '*.cpp' bodgy .\src
.\src\buffer\out\OutputCell.cpp: // BODGY: Misdiagnosis in MSVC 17.11: Referencing global constants in the member
.\src\buffer\out\OutputCellIterator.cpp: // BODGY: Misdiagnosis in MSVC 17.11: Referencing global constants in the member
.\src\buffer\out\OutputCellView.cpp: // BODGY: Misdiagnosis in MSVC 17.11: Referencing global constants in the member
.\src\buffer\out\textBuffer.cpp: // BODGY: Misdiagnosis in MSVC 17.11: Referencing global constants in the member
.\src\buffer\out\textBuffer.cpp: // BODGY: We use oldCursorPos in two critical places below:
.\src\cascaadia\ElevateShim\elevate-shim.cpp: // BODGY
.\src\cascaadia\TerminalApp\AppActionHandlers.cpp: // BODGY: GH#12021
.\src\cascaadia\TerminalApp\HighlightedTextControl.cpp: // BODGY - The XAML compiler emits a boxed int32, but the dependency property
.\src\cascaadia\TerminalApp\SnippetsPaneContent.h: int32_t Row() { return HasChildren() ? 2 : 1; } // See the BODGY comment in the .XAML for explanation
.\src\cascaadia\TerminalApp\SuggestionsControl.cpp: // BODGY: Calling ScrollIntoView on a ListView doesn't always work
.\src\cascaadia\TerminalApp\Tab.cpp: // BODGY: When the tab is drag/dropped, the TabView gets a
.\src\cascaadia\TerminalApp\Tab.cpp: // BODGY: contrary to the docs, Insert() seems to throw if the value already exists
.\src\cascaadia\TerminalApp\Tab.cpp: // BODGY
.\src\cascaadia\TerminalApp\TerminalPage.cpp: // BODGY: After a ContentDialog is dismissed, FindName() can no longer
.\src\cascaadia\TerminalApp\TerminalPage.cpp: // BODGY
.\src\cascaadia\TerminalApp\TerminalPage.h: // BODGY: WinUI's TabView has a broken close event handler:
.\src\cascaadia\TerminalControl\SearchBoxControl.cpp: // BODGY: It is possible for this to get called with no XAML root. We are unsure why.
.\src\cascaadia\TerminalSettingsEditor\Appearances.cpp: // BODGY: Depends on WinUI internals.
.\src\cascaadia\TerminalSettingsEditor\Launch.cpp: // BODGY
.\src\cascaadia\TerminalSettingsEditor\MainPage.cpp: // BODGY
.\src\cascaadia\TerminalSettingsEditor\Utils.h: // BODGY!
.\src\cascaadia\TerminalSettingsModel\WslDistroGenerator.cpp: // but it is certainly BODGY
.\src\cascaadia\UIHelpers\TextMenuFlyout.cpp: // BODGY:
.\src\cascaadia\UIHelpers\TextMenuFlyout.cpp: // this is bodgy because WinUI 2.8 is buggy. There's no other solution here.
.\src\cascaadia\WindowsTerminal\AppHost.cpp: // BODGY
.\src\cascaadia\WindowsTerminal\AppHost.cpp: // - BODGY workaround for GH#9320. When the window moves, dismiss all the popups
.\src\cascaadia\WindowsTerminal\IslandWindow.cpp: // BODGY
.\src\cascaadia\WindowsTerminal\main.cpp: // !! BODGY !!
.\src\cascaadia\WindowsTerminal\NonClientIslandWindow.cpp: // - BODGY: It also _manually_ handles the caption buttons. They exist in the
.\src\cascaadia\WindowsTerminal\NonClientIslandWindow.cpp: // !! BODGY !!
.\src\cascaadia\WindowsTerminal\VirtualDesktopUtils.cpp: // BODGY
.\src\cascaadia\WinRTUtils\inc\Utils.h: // BODGY: The MSVC 14.31.31103 toolset seems to misdiagnose this line.
.\src\renderer\atlas\BackendO3D.h: // BODGY: At the time of writing IDWriteFontFallback::MapCharacters returns the same IDWriteFontFace instance
.\src\renderer\base\renderer.cpp: // BODGY: Optimally we would want to retry per engine, but that causes different
```

 [Coreutils for Windows on GitHub](#)

Install

[WinGet](#) を使用して Coreutils をインストールします。

PowerShell

```
winget install Microsoft.Coreutils
```

または、GitHub。

どのように機能するのか

ユーティリティは、[uutils/coreutils](#) プロジェクトの上に Rust で実装されます。これは、最新の Linux ディストリビューションに付属する GNU coreutils のクロスプラットフォーム再実装と同じです。Microsoftでは、`coreutils`、`findutils` (`find`、`xargs`)、GNU 互換の `grep` を1つのパッケージとしてバンドルするWindowsに重点を置いたビルドが維持されます。また、元の DOS `sort` と `find` の統合ポートも含まれているため、`/switch` スタイルの構文に依存する既存の CMD スクリプトは、UNIX スタイルのバージョンと共に動作し続けます。詳細については、「[シェルの競合](#)」を参照してください。

関連するコンテンツ

- [Linux 用 Windows サブシステム](#)
- [Windows ターミナル](#)
- [Sudo for Windows](#)
- [WinGet](#)

Last updated on 2026/06/05

Commands

Windowsの Coreutils には、次のコマンドが付属しています。 各コマンドは、完全な構文とオプションの標準 `--help` フラグをサポートしています。 説明とリンクは、アップストリームのドキュメントに移動します。

- ほとんどのユーティリティは [utils/coreutils](#) から取得されます。
- `find` `xargs` は [utils/findutils](#) から取得されます。
- `grep` は [utils/grep](#) サブモジュールから取得されます。

シェル組み込みと競合するコマンドについては、 [シェルの競合](#) を参照してください。

 テーブルを展開する

命令	説明
arch	コンピューターのアーキテクチャを表示します。
b2sum	BLAKE2b チェックサムを印刷または確認します。
base32	base32 を使用してデータをエンコードまたはデコードし、標準出力に出力します。
base64	base64 を使用してデータをエンコードまたはデコードし、標準出力に出力します。
basename	先頭のディレクトリ コンポーネントが削除された印刷名。
basenc	いくつかの基本エンコードのいずれかを使用してデータをエンコードまたはデコードします。
cat	FILE または標準入力を標準出力に連結します。
cksum	各ファイルの CRC とサイズを印刷します。
comm	2 つの並べ替えられたファイルを 1 行ずつ比較します。
cp	SOURCE を DEST にコピーするか、複数のソースを DIRECTORY にコピーします。
csplit	コンテキスト行によって決定されるセクションにファイルを分割します。
cut	指定したバイト列またはフィールド列を入力各行から出力します。
date	システムの日付と時刻を印刷または設定します。
df	各 FILE が存在するファイル システムに関する情報を表示します。
dirname	ファイル名から最後のコンポーネントを削除します。

命令	説明
du	ファイル領域の使用量を見積もります。
echo	テキストの行を表示します。
env	環境内の各 NAME を VALUE に設定し、COMMAND を実行します。
expr	EXPRESSION の値を標準出力に出力します。
factor	指定された数値の素因数を出力します。
false	コマンドを常に 1 で終了します。
find	ディレクトリ階層内のファイルを検索します。
fmt	段落を入力 (または標準入力) から stdout に再フォーマットします。
fold	指定した幅に合わせて各入力行を折り返します。
<code>grep</code>	パターンに一致する行を印刷します。
head	各ファイルの最初の 10 行を標準出力に出力します。
hostname	システムのホスト名を表示または設定します。
join	共通フィールドで 2 つのファイルを結合します。
la	非表示のエントリ (<code>ls -A</code> のエイリアス) を含むディレクトリの内容を一覧表示します。
link	<code>link</code> 関数を呼び出して、既存の FILE1 への FILE2 という名前のリンクを作成します。
ln	ファイル間のリンクを作成します。
ls	ディレクトリの内容を一覧表示します。
md5sum	MD5 チェックサムを印刷または確認します。
mkdir	指定された DIRECTORY が存在しない場合は作成します。
mktemp	一時ファイルまたはディレクトリを作成します。
mv	SOURCE を DEST に移動するか、複数の SOURCE を DIRECTORY に移動します。
nl	ファイルの行数。
nproc	現在のプロセスで利用できるコアの数を出力します。
numfmt	人間が判読できる文字列から、または人間が判読できる文字列に数値を変換します。
od	ファイルを 8 進数やその他の形式でダンプします。
pathchk	ファイル名が有効か移植可能かを確認します。

命令	説明
pr	印刷用の改ページ位置または列付きファイル。
printenv	指定した環境変数の値を表示します。
printf	書式指定文字列と引数に基づいて出力を出力します。
ptx	ファイルの内容の順列インデックスを生成します。
pwd	現在の作業ディレクトリの完全なファイル名を表示します。
readlink	シンボリック リンクまたは正規ファイル名の値を出力します。
realpath	解決された絶対パスを出力します。
rm	ファイルを削除 (リンク解除) します。
rmdir	ディレクトリが空の場合は、ディレクトリを削除します。
seq	INCREMENT のステップで、FIRST から LAST までの数値を表示します。
sha1sum	SHA1 チェックサムを印刷または確認します。
sha224sum	SHA224 チェックサムを印刷または確認します。
sha256sum	SHA256 チェックサムを印刷または確認します。
sha384sum	SHA384 チェックサムを印刷または確認します。
sha512sum	SHA512 チェックサムを印刷または確認します。
shuf	入力行のランダムな順列を出力して、入力をシャッフルします。
sleep	NUMBER 秒数だけ一時停止します。
sort	すべての FILE の並べ替えられた連結を表示します。
split	入力の連続したセクションまたはインターリーブされたセクションを含む出力ファイルを作成します。
stat	ファイルまたはファイル システムの状態を表示します。
sum	ファイル内のブロックをチェックサムしてカウントします。
tac	各ファイルを標準出力に書き込み、最後の行を最初に書き込みます。
tail	各ファイルの最後の 10 行を標準出力に出力します。
tee	標準入力を各ファイルにコピーし、標準出力にもコピーします。
test	ファイルの種類を確認し、値を比較します。
touch	各 FILE のアクセス時間と変更時刻を更新します。

命令	説明
tr	文字を翻訳または削除します。
true	コマンドを常に 0 で終了します。
truncate	各ファイルのサイズを指定したサイズに縮小または拡張します。
tsort	FILE 内の文字列のトポロジ的な並べ替え。
unexpand	各ファイルの空白をタブに変換します。
uniq	繰り返し行を報告または省略します。
unlink	FILE でファイルのリンクを解除します。
uptime	現在の時刻、システムの稼働時間、ユーザー数、および負荷の平均を表示します。
wc	各ファイルの改行、単語、バイト数を出力します。
xargs	標準入力からコマンド ラインをビルドして実行します。
yes	文字列 (または y) で行を繰り返し表示します。

関連するコンテンツ

- [coreutils for Windows overview](#)
- [シェルの競合](#)

Windows での curl

curl は、サーバーとの間でデータを転送するためのコマンドライン ツールです。これは Windows に含まれており、HTTP、HTTPS、FTP、SFTP などの幅広いプロトコルをサポートしているため、追加のツールをインストールすることなく、REST API の呼び出し、ファイルのダウンロード、エンドポイントのテストを行う便利な方法になります。

Windows バージョンはアップストリームの [curl](#) プロジェクトからビルドされるため、Linux と macOS で認識されているのと同じフラグと動作が Windows で同じように動作します。

⚠ Note

Windows PowerShell 5.1 では、`curl` という名前の組み込みのエイリアスが定義されています。このエイリアスは、`Invoke-WebRequest` にマップされ、`curl.exe` をシャドウし、さまざまなパラメーターを受け取ります。Windows PowerShell 5.1 で実際の curl を使用するには、`Remove-Item Alias:curl` でエイリアスを削除するか、`curl.exe` として明示的に呼び出します (例: `curl.exe -O https://example.com/file.zip`)。PowerShell 7 以降では、このエイリアスは定義されません。

一般的なコマンド

ファイルをダウンロードします。

PowerShell

```
curl -O https://example.com/file.zip
```

GET 要求を行い、応答を出力します。

PowerShell

```
curl https://api.example.com/data
```

JSON POST 要求を送信します。

PowerShell

```
curl -X POST https://api.example.com/items `
  -H "Content-Type: application/json" `
```

```
-d '{"name":"widget"}'
```

オプションの完全な一覧については、`curl --help` 実行するか、[公式ドキュメント](#)  を参照してください。

Last updated on 2026/06/02

コマンド ライン ツールの編集

Edit は、Windows 11用の軽量でオープンソースのコマンドライン テキスト エディターです。従来の [MS-DOS エディター](#) に敬意を表しますが、VS Code に似た最新のインターフェイスと入力コントロールを使用します。

可用性

編集は、2025 年 9 月のオプション更新プログラム (またはそれ以降) の一部としてWindows 11 と、Windows 11 2025 更新プログラム (バージョン 25H2) に含まれています。システムで Edit が既に使用できるかどうかを確認するには、コマンド プロンプト、PowerShell、または Windows ターミナルを開き、「`edit`」と入力します。

Windowsのバージョンで Edit をまだ使用できない場合は、winget

PowerShell

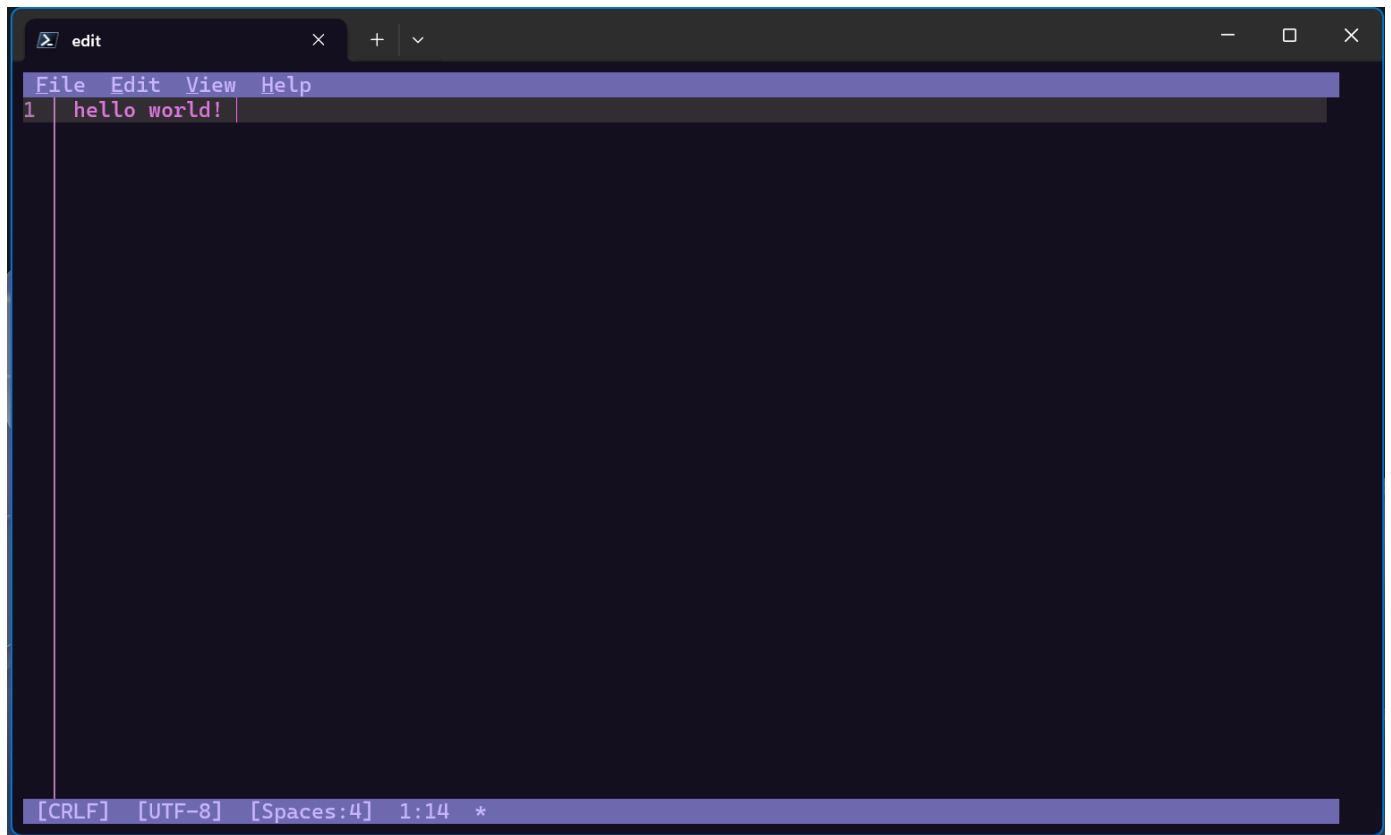
```
winget install Microsoft.Edit
```

GitHubから最新リリースを<>ダウンロードすることもできます。

編集は、Linux および macOS でも使用できます。これらのプラットフォームのインストール手順については、[Edit GitHub リポジトリ](#) を参照してください。

編集を使用する方法

編集を開くには、コマンド ラインに「`edit`」と入力するか、`edit <your-file-name>` を実行して特定のファイルを開きます。コンテキストを切り替えることなく、ターミナルで直接ファイルを編集できます。



Features

編集には、いくつかの機能が用意されています。

- **モードレス編集:** 編集では、テキスト ユーザー インターフェイス (TUI) を持つモードレス インターフェイスが使用されるため、学習するモードはありません。すべてのメニュー オプションには、事前に構成されたキー バインドもあります。
- **複数のファイル:** `Ctrl + P` キーを押しながら複数のファイルを開いて切り替えます。
- **検索と置換:** テキストを検索して `Ctrl + F` キーを押して置換するか、[TUI] メニューの [編集] > [検索] を選択します。大文字と小文字の区別、単語全体、および正規表現のオプションに対応しています。
- **Word Wrap:** `Alt+Z` で折り返しを切り替えるか、TUI メニューで [View > Word Wrap] を選択します。
- **マウスのサポート:** マウスを使用してメニュー内を移動し、テキストを選択し、スクロールします。

FAQ

別のコマンド ライン テキスト エディターを作成する理由

クラシック 16 ビット MS-DOS エディターが廃止された後、Windows の 64 ビットバージョンには既定の CLI テキスト エディターが含まれていませんでした。Edit は、初心者でも使い始めやすいモードレスエディターによって、そのギャップを埋めます。

多くの既存のモードレス エディターは、Windows に対するファーストパーティのサポートを持っていないか、OS とバンドルするには大きすぎるため、Edit は Rust で最初から構築されました。

オープン ソース リポジトリを編集する

編集は MIT ライセンスの下でオープンソースされ、投稿とフィードバックを歓迎します。ソースコードは、[GitHub](#) にあります。

Last updated on 2026/06/02

Sudo for Windows

Sudo for Windows は、ユーザーが管理者特権のコマンド (管理者として) を Windows 上の未選択のコンソール セッションから直接実行する方法です。

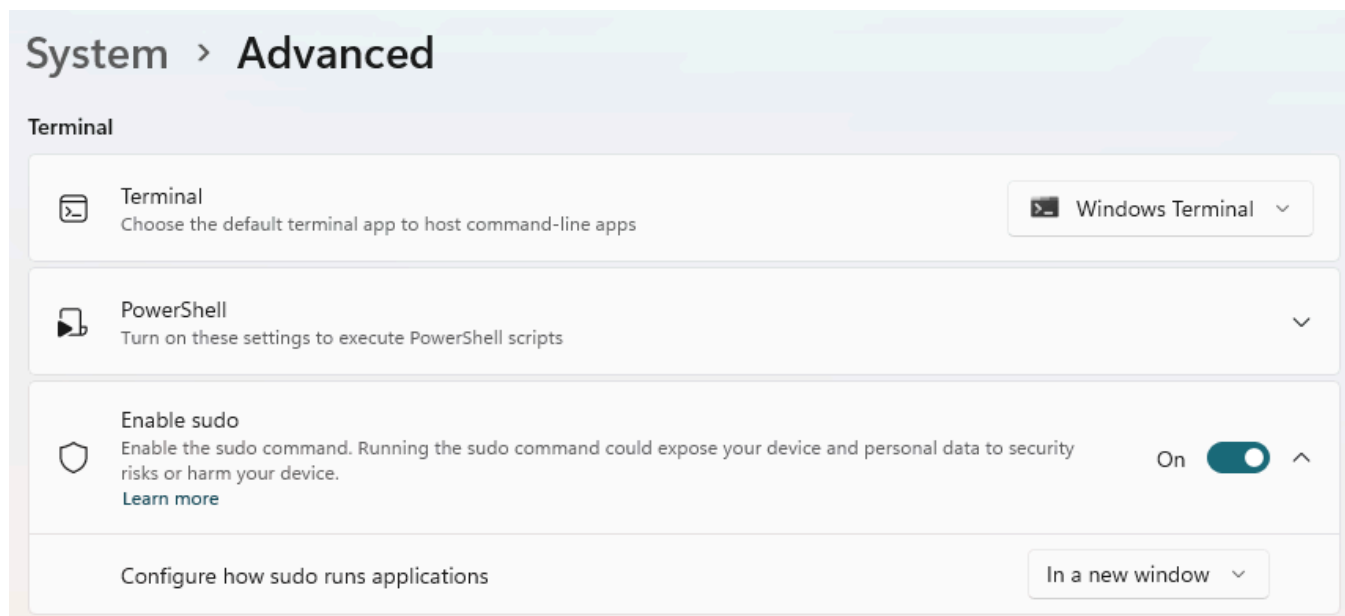
デモ ビデオと、Sudo for Windows のしくみの詳細については、[告知](#)をご覧ください。

[前提条件]

Sudo for Windows コマンドは、[Windows 11 バージョン 24H2](#) 以降で使用できます。
([Windows 更新プログラムをご確認ください](#)。)

Sudo for Windows を有効にする方法

Windows で Sudo を有効にするには、[Windows 設定] で [System > Advanced] を開き、[sudo の有効化] を [オン] に設定します。



⚠ 警告

特定の構成で有効にした場合は、特権ベクトルの潜在的な昇格として Sudo for Windows を使用できます。マシン上で sudo コマンドを有効にする際には、[セキュリティに関する考慮事項](#)に注意してください。

Sudo for Windows を構成する方法

Sudo for Windows では現在、3 つの異なる構成オプションがサポートされています。構成は、[システム > の \[詳細設定\]](#) メニューから、またはコマンド ラインを使用してプログラム

で設定できます。構成には以下のオプションがあります。

- **新しいウィンドウを使用する** (`forceNewWindow`): `forceNewWindow` 構成オプションは、Sudo for Windows の既定の構成オプションです。コマンドを新しいウィンドウで実行するには、この構成で `sudo` を使用します。これは、`runas /user:admin` コマンドの動作に似ています。
- **入力を閉じる** (`disableInput`): `disableInput` 構成オプションでは、昇格されたプロセスが現在のウィンドウで実行されます。入力ハンドルは閉じた状態です。つまり、この昇格されたプロセスでは、現在のコンソール ウィンドウから入力を受け取ることができません。このオプションは、管理者としてコマンドを実行する一方で、現在のコンソール ウィンドウからの入力をコマンドに渡したくない場合に便利です。この構成オプションでは、`inline` 構成オプションの利便性を活かしながら、関連するセキュリティ リスクを軽減できます。
- **インライン** (`normal`): `normal` 構成オプションは、他のオペレーティング システムでの `sudo` の動作に最も似ています。この構成では、昇格されたプロセスが現在のウィンドウで実行され、プロセスは現在のコンソール セッションから入力を受け取ることができます。このオプションは、管理者としてコマンドを実行し、現在のコンソール ウィンドウからの入力をコマンドに渡したい場合に便利です。この構成オプションは最も便利ですが、このオプションの使用は、関連するセキュリティ リスクに精通している場合のみに限定してください。

[[システム > 詳細設定](#)] メニューからこれらの構成を選択するか、管理者特権のコマンド ライン (管理コンソール) でプログラムで構成を変更できます。

- `sudo config --enable <configuration_option>`

`<configuration_option>` は `forceNewWindow`、`disableInput`、または `normal` に置き換えてください。

Sudo for Windows を使用する方法

Sudo for Windows は、管理者として実行するコマンドの先頭に `sudo` を追加するだけで使用できます。たとえば、`netstat -ab` を管理者として実行するには、コンソール ウィンドウで `sudo netstat -ab` を実行します。

一般的な開発者シナリオ

開発者が `sudo` を使用する一般的なシナリオを次に示します。

- **システム ファイルの編集:** `sudo notepad C:\Windows\System32\drivers\etc\hosts`

- パッケージをグローバルにインストールする: `sudo npm install -g package-name`
- 管理ツールの実行: `sudo diskpart`

ⓘ Note

`C:\Windows\`などのシステム ディレクトリを含む開発作業では、可能な場合は開発環境または代替アプローチの使用を検討してください。Sudo は、管理者特権のアクセス許可が本当に必要な場合にのみ、慎重に使用する必要があります。

`sudo` により、対象のプロセスが昇格され、管理者レベルのアクセス許可を使用して実行されるため、続行するかどうかを確認するプロンプトが開きます。

セキュリティに関する考慮事項

入力を閉じるオプション (`disableInput`) やインライン オプション (`normal`) の構成での `sudo` の実行には、リスクがあります。昇格されていない `sudo.exe` プロセスと昇格された `sudo.exe` プロセスが確立した接続を利用し、悪意のあるプロセスが昇格されたプロセスを操作しようとする可能性があります。

`disableInput` 構成オプションでは、入力ハンドルを閉じることでリスクが軽減されます。現在のコンソール ウィンドウから入力ハンドルを切断すると、昇格されていないプロセスは昇格されたプロセスに入力を送信できません。

`inline` 構成オプションでは、昇格されたプロセスが現在のウィンドウで実行され、プロセスが現在のコンソール セッションから入力を受け取ることができます。未選択のプロセスは、同じコンソール ウィンドウ内の昇格されたプロセスに入力を送信したり、この構成の現在のウィンドウの出力から情報を取得したりできます。

FAQ

Windows 用 Sudo と既存 `runas` のコマンドの違い

この `sudo` コマンドは、現在昇格されていないコマンド ライン コンテキストから、管理者としてのコマンド実行に簡単に昇格する方法として使用できます。他のオペレーティング システムを使用している一部のユーザーにとっては、馴染みのあるコマンドでもあります。 `runas` コマンドでは、管理者としての実行を含め、任意のユーザーとしてプログラムを実行する方法が提供されます。一方、現時点で Windows 上の `sudo` コマンドは、他のユーザーとしてのプログラムの実行をサポートしていません。これ以外に、`sudo` と `runas` には、主に次のような違いがあります。

- `runas` では、管理者を含む (管理者に限らず) 他のユーザーとしてプログラムを実行できます。この機能は、`sudo` コマンドのロードマップに含まれているものの、現時点では存在しません。
- `sudo` では、プロセスを管理者権限で簡単に昇格できます。
 - これを新しいウィンドウで行うことも選択できます。これは、`runas` の管理者フローに似ています。
 - 構成オプションとして `disableInput` および `normal` を使用すると、昇格されたプロセスを現在のコンソール ウィンドウに接続できます。これは、`runas` ではサポートされていません。
- `runas` では、コマンドラインでユーザーにパスワードの入力を求めることができます。
- `sudo` は、確認プロンプトの使用により、承認されていない変更からオペレーティングシステムを保護するように設計されているユーザー アカウント制御 (UAC) セキュリティ機能のみを通じて昇格できます。

個々のユースケースを考慮して、ニーズに最も適したコマンドの使用を計画する必要があります。また、`sudo` を `disableInput` モードおよび `normal` モードで実行する場合のセキュリティへの影響も考慮する必要があります。他の `forceNewWindow` 構成に関連するリスクに慣れている場合を除き、既定の `sudo` 構成オプションの使用をお勧めします。

Sudo for Windows オープン ソース リポジトリ

Sudo for Windows はオープン ソースです。皆様からのコントリビューションやフィードバックを歓迎いたします。Sudo for Windows のソース コードについては、[GitHub](#) をご覧ください。

その他の機能

Sudo for Windows で提供されていない追加機能をお探しの場合は、多数の追加機能と構成オプションが提供されている [gsudo by Gerardo Grignoli](#) をご覧ください。コミュニティで他のソリューションをチェックすることもできます。

Windows での tar

`tar` は、Windowsに含まれるコマンドラインアーカイブツールです。これにより、`.tar`、`.tar.gz`、`.zip`、`.7z` などのアーカイブファイルを、追加のソフトウェアをインストールすることなく、コマンドラインから直接作成、一覧表示、および抽出できます。

`tar` のWindowsビルドは、[libarchive](#) の `bsdtar` に基づいており、開発者が Linux と macOS で既に使用しているのと同じフラグをサポートしているため、既存のスクリプトと習慣は引き継がれます。

一般的なコマンド

アーカイブを抽出します。

PowerShell

```
tar -xf archive.tar.gz
```

フォルダーから `.tar.gz` を作成します。

PowerShell

```
tar -czf archive.tar.gz .\my-folder
```

フォルダーから `.zip` を作成します (`-a` フラグは、ファイル拡張子からアーカイブ形式を自動的に選択し、`-z`、`-j`、`-J`、`-Z` などの形式固有のフラグを置き換えます)。

PowerShell

```
tar -caf archive.zip .\my-folder
```

展開せずにアーカイブの内容を一覧表示します。

PowerShell

```
tar -tf archive.tar.gz
```

オプションの完全な一覧については、`tar --help` または [FreeBSD Tar マニュアル](#) を実行してください。

PowerToys: Windowsをカスタマイズするユーティリティ



Microsoft PowerToys は、パワー ユーザーと開発者が Windows からさらに多くを得るのを助けるために設計された無料のオープンソース ユーティリティのセットです。使い慣れた Windows エクスペリエンスに基づいて構築され、生産性を高め、ワークフローを合理化し、すぐには存在しないカスタマイズ オプションのロックを解除する、思慮深いツールが追加されます。ウィンドウ レイアウトの管理やキーの再マッピングから、コマンドの即時起動、日常のタスクの自動化まで、PowerToys はすべて摩擦を取り除き、より速く、よりスマートで、より多くの制御を行う必要のある作業に集中できるようにすることです。

取り付ける PowerToys

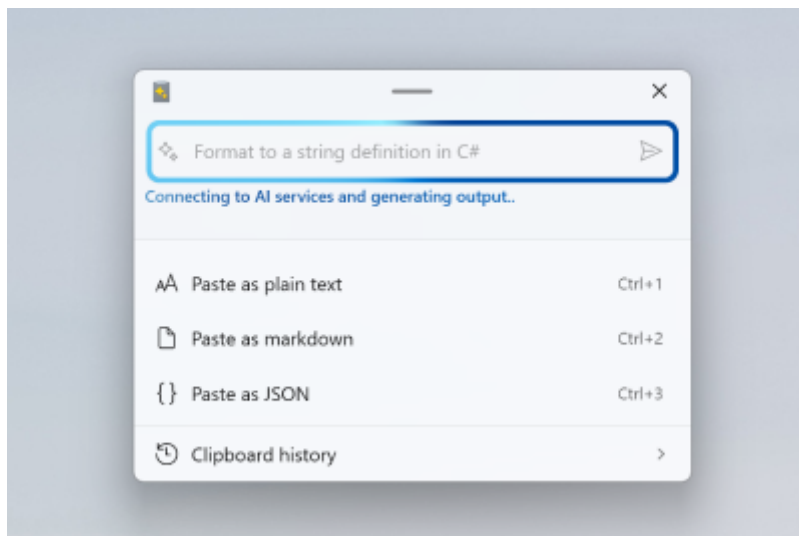
PowerToys はオープンソース プロジェクトです。GitHubにアクセスして、コードベースを調べ、問題を報告し、投稿し、各リリースの新機能を最新の状態に保ちます。

[PowerToysGitHub 上](#)

[GitHub での最新リリース](#)

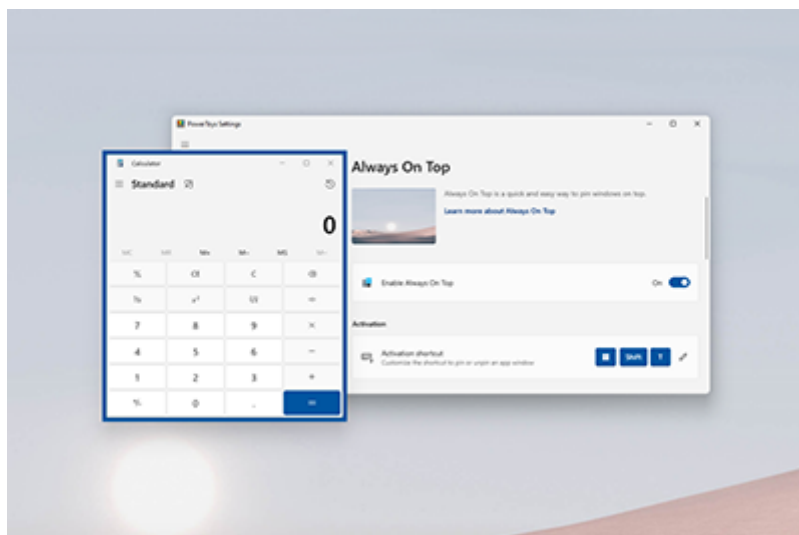
公共サービス

現在利用可能なユーティリティは、次のとおりです。



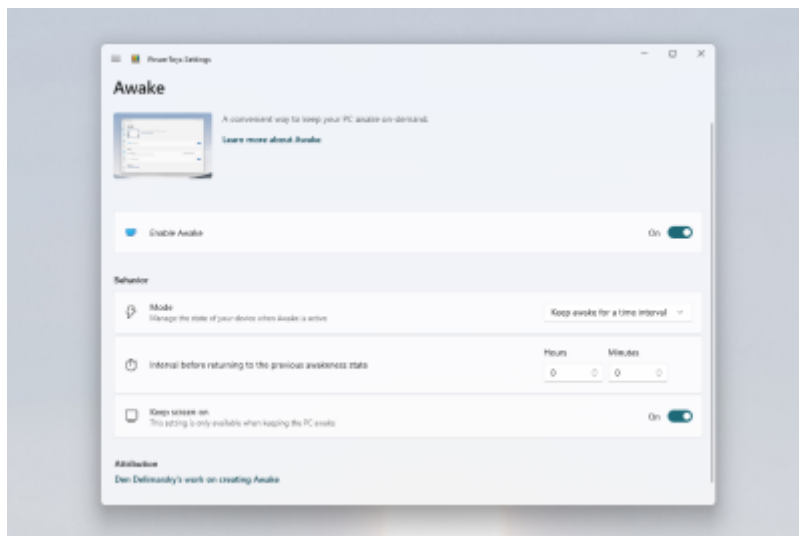
高度な貼り付け

クリップボードから必要な任意の形式にテキストを貼り付けます。 オプションの AI を利用した機能が含まれています。



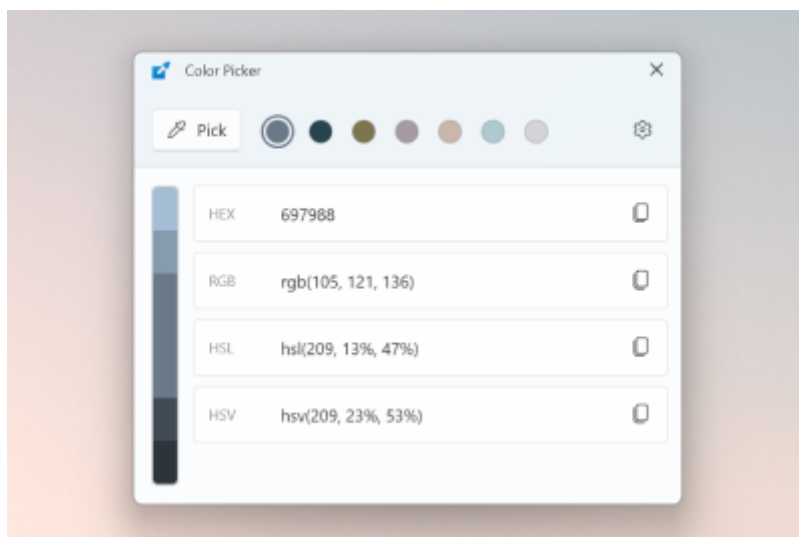
Always On Top

クイック キー ショートカットを使用して、ウィンドウを他のウィンドウの上にピン留めします。



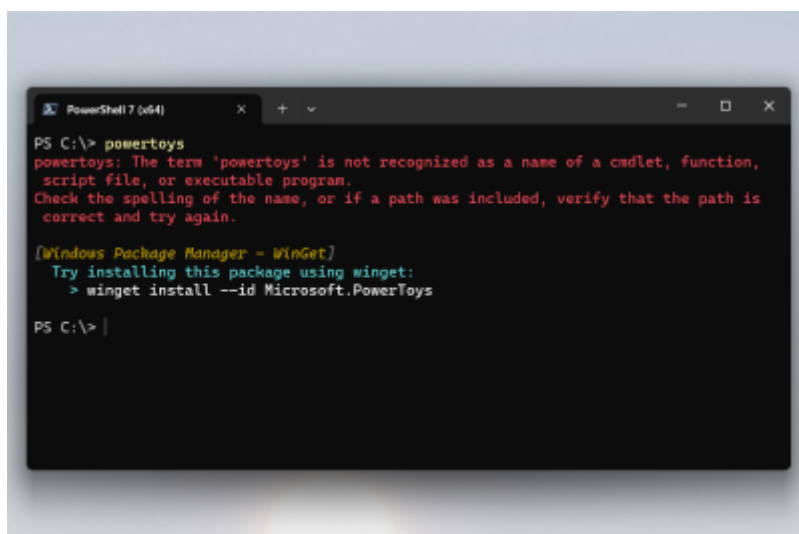
Awake

電源とスリープの設定を管理せずに、コンピューターを起動したままにします。



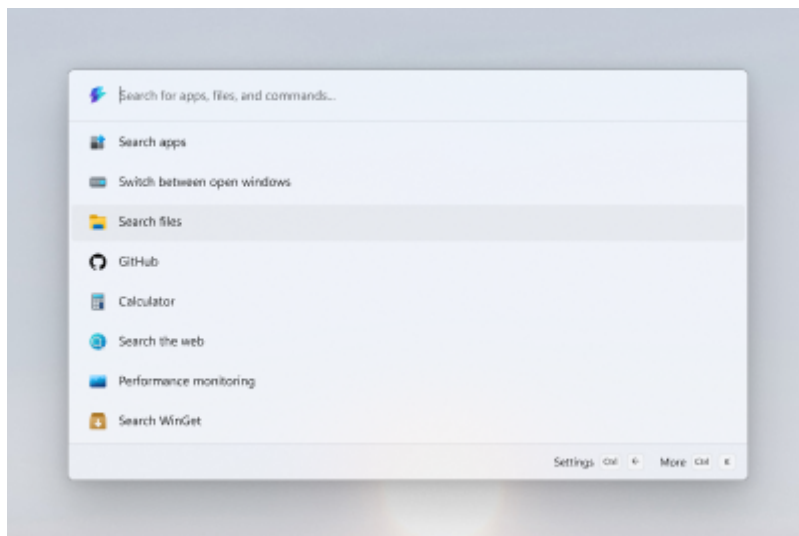
カラーピッカー

画面上のどこからでも色を選択するためのシステム全体のカラー 選択ユーティリティ。



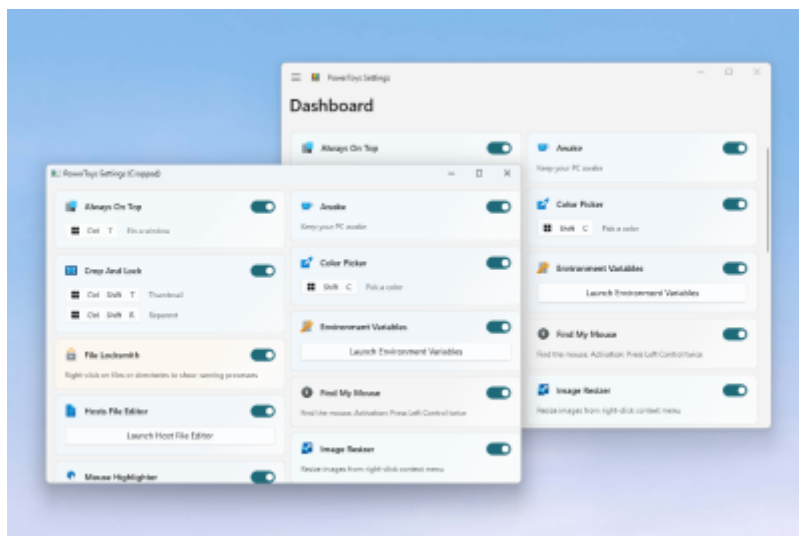
コマンドが見つかりません

コマンドが失敗したときに WinGet パッケージを提案する PowerShell 7 モジュール。



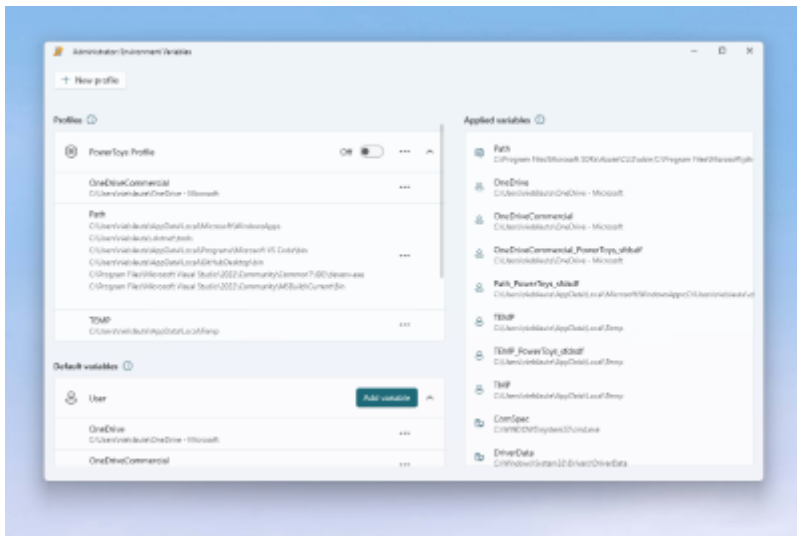
コマンド パレット

頻繁に使用されるコマンド、アプリ、ツールに、1つの高速でカスタマイズ可能なインターフェイスからアクセスします。



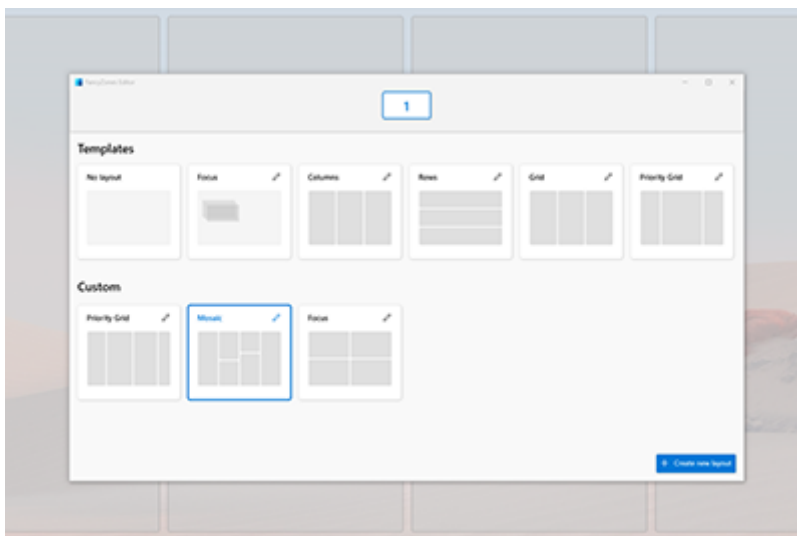
トリミングとロック

別のウィンドウの一部をインタラクティブなままにするために、クロップまたはサムネイルウィンドウを作成します。



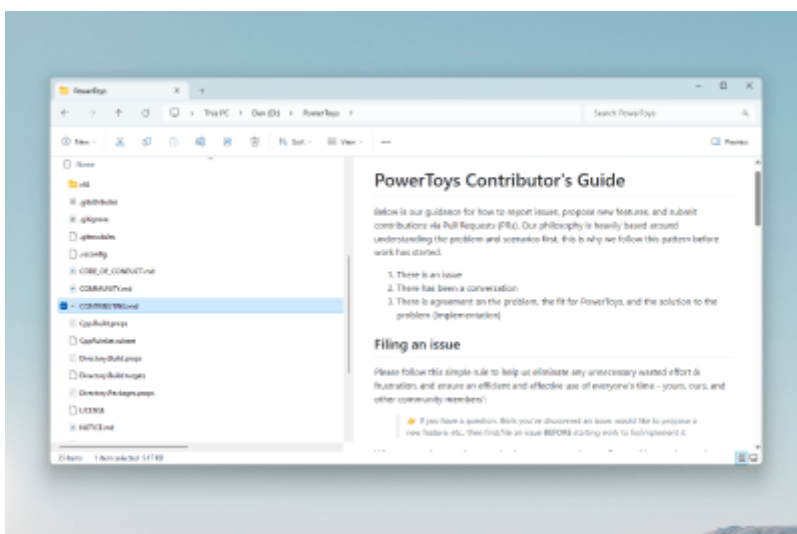
環境変数

プロファイルを使用して環境変数を管理し、変数をグループ化します。



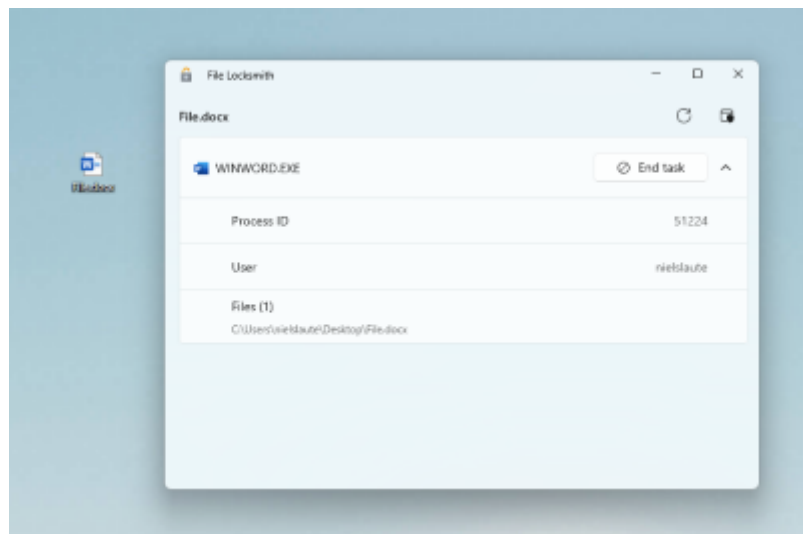
FancyZones

複雑なウィンドウ レイアウトを作成し、ウィンドウをすばやく配置するためのウィンドウ マネージャー。



File Explorer アドオン

File Explorerのさまざまなファイルの種類に対してプレビュー ウィンドウとサムネイルレンダリングを有効にします。



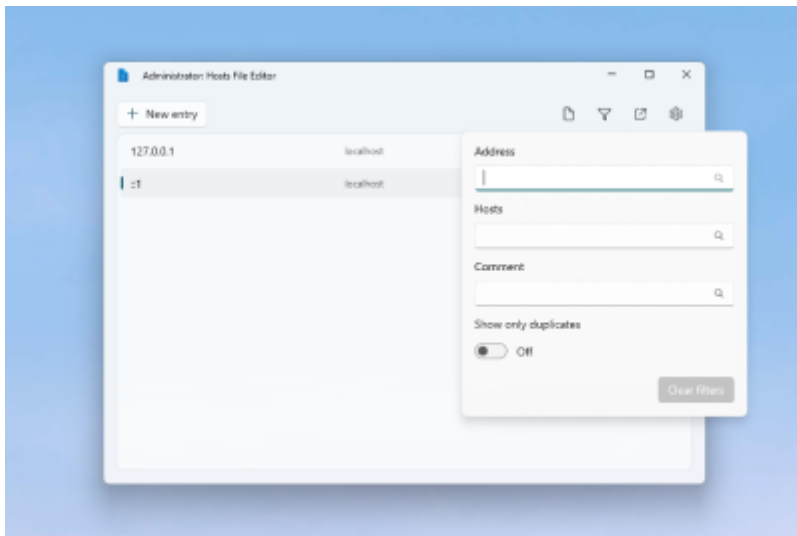
ファイルロックスミス

File Explorerコンテキスト メニューから、使用中のファイルとプロセスを確認します。



グラブアンドムーブ

修飾子キーを押したままウィンドウ内の任意の場所にドラッグして、ウィンドウを移動およびサイズ変更します。



ホストファイルエディタ

ドメイン名と IP アドレスを含む Windows Hosts ファイルを編集します。

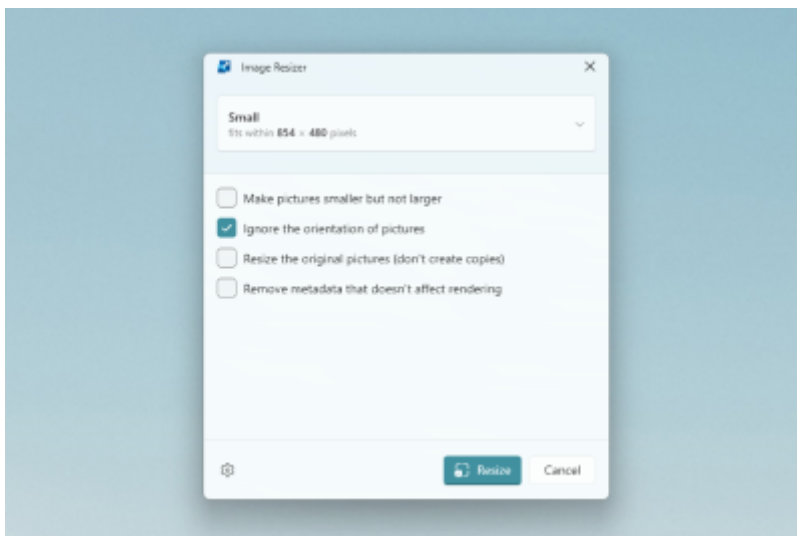
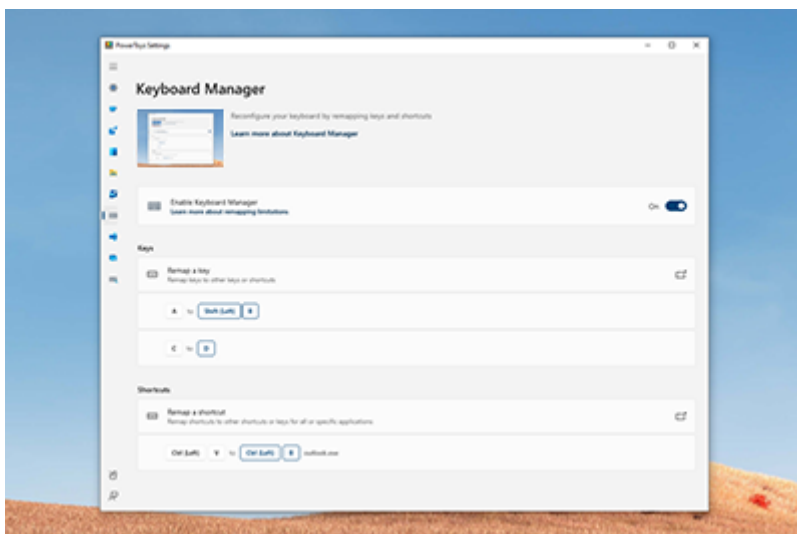


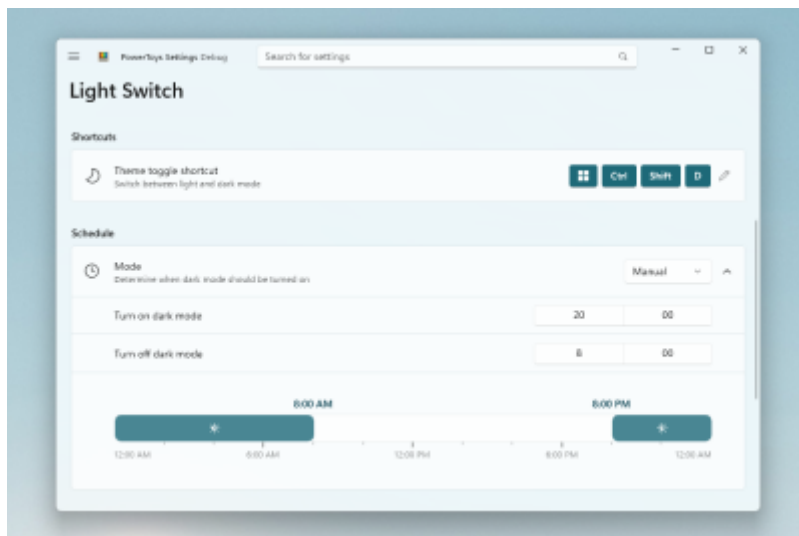
Image Resizer

File Explorerで右クリックすると、画像のサイズをすばやく変更できます。



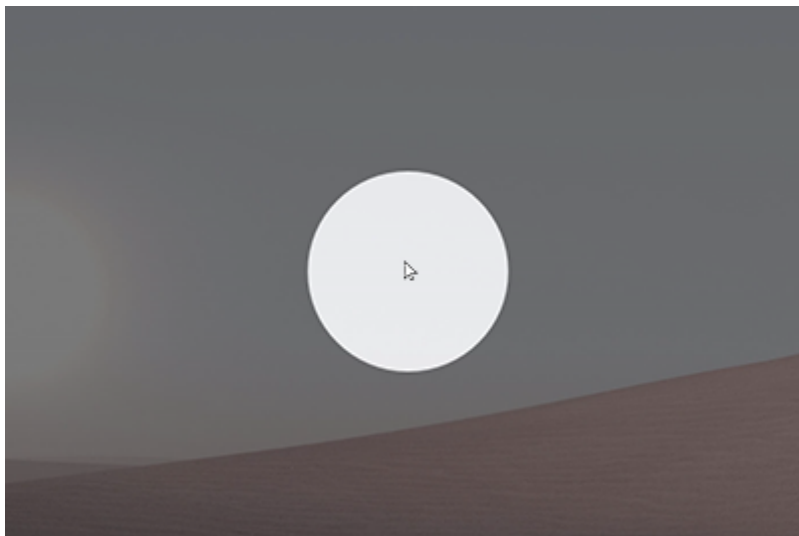
キーボード マネージャー

キーを再マップし、カスタム ショートカットを作成して、キーボードをカスタマイズします。



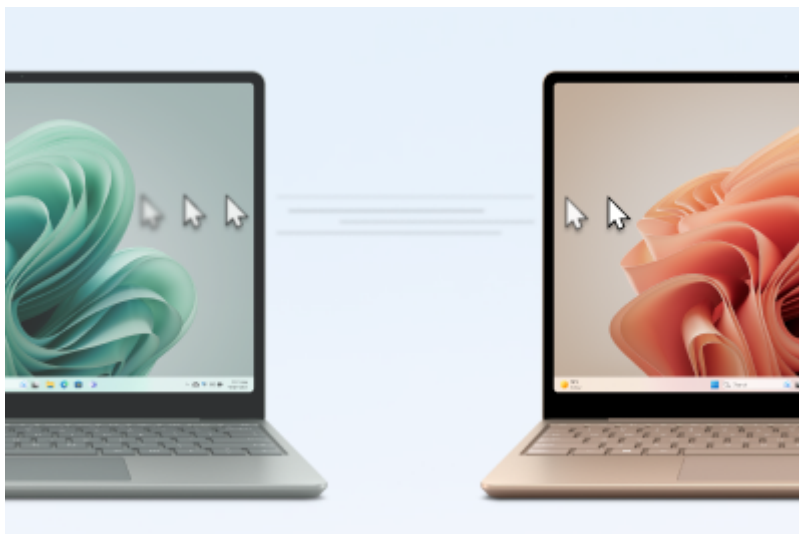
ライトスイッチ

時刻に基づいて、明るいテーマと暗いテーマを自動的に切り替えます。



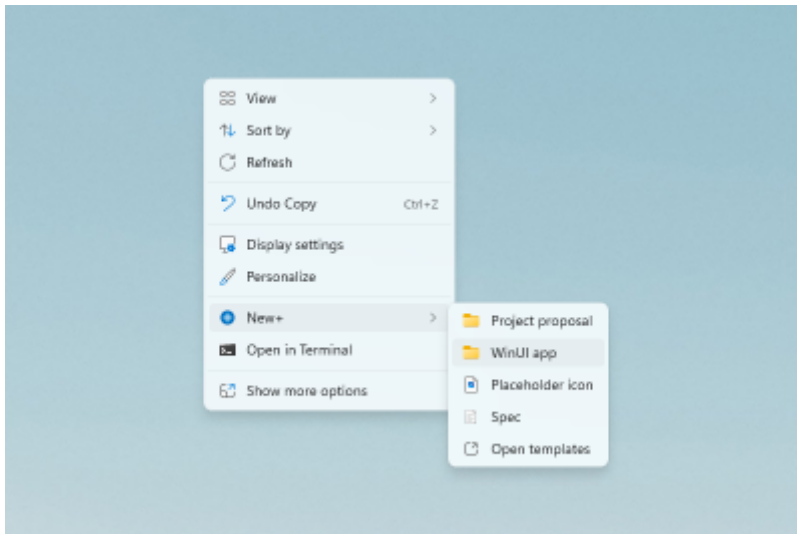
マウスユーティリティ

[マウスの検索]、[蛍光ペン]、[ジャンプ]、[十字線] を使用して、マウスとカーソルを強化する機能を追加します。



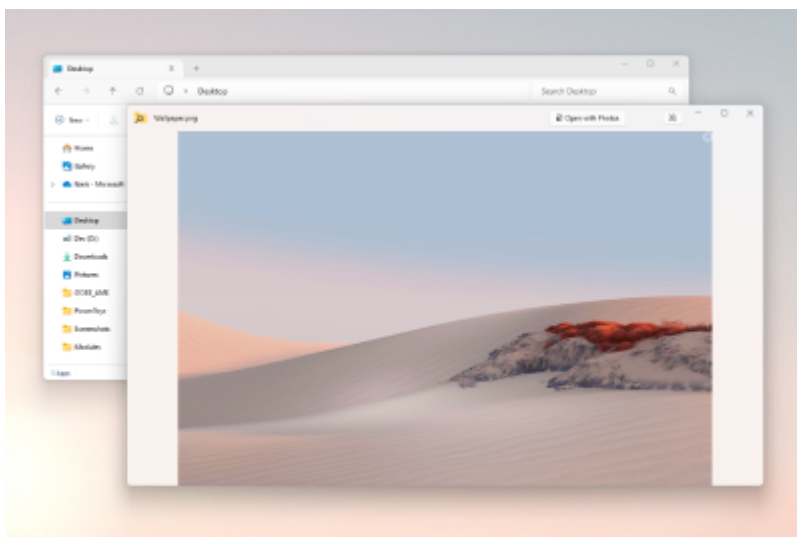
罫線のないマウス

1つのキーボードとマウスで複数のコンピューターを制御します。



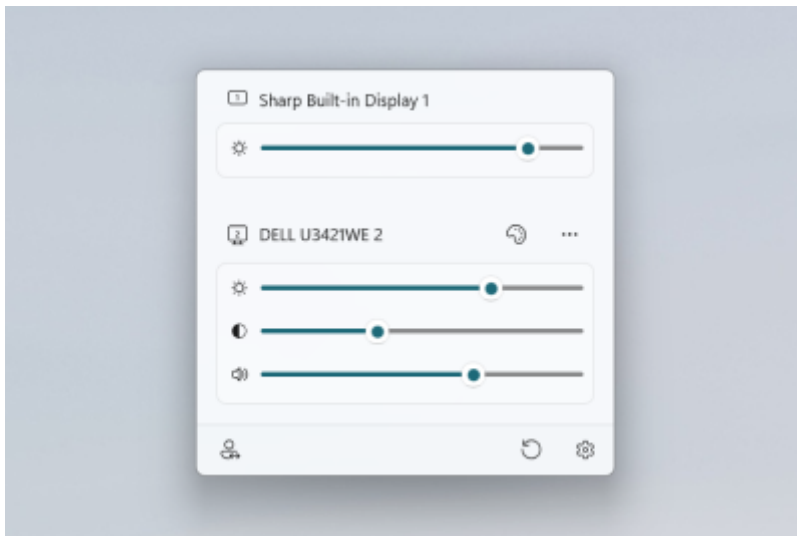
New+

File Explorerでカスタマイズされたテンプレート のセットからファイルとフォルダーを作成します。



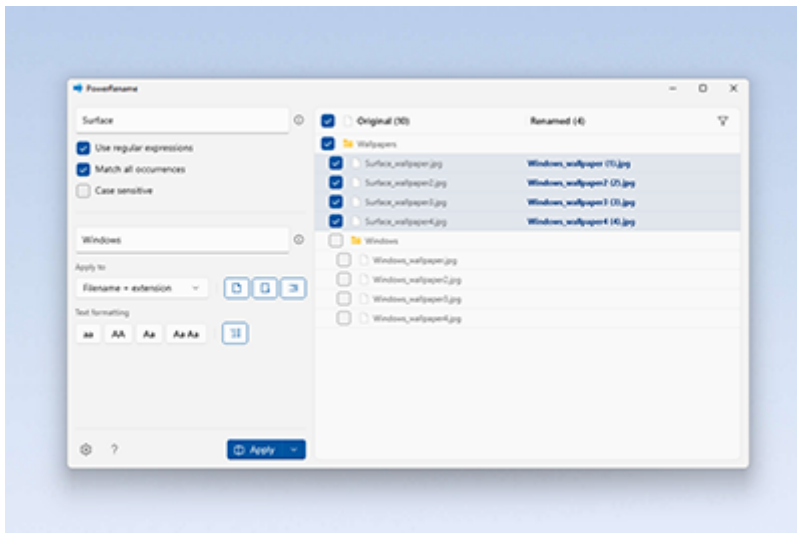
ピーク

アプリケーションを開いたり、ワークフローを中断したりせずに、ファイルコンテンツをプレビューします。



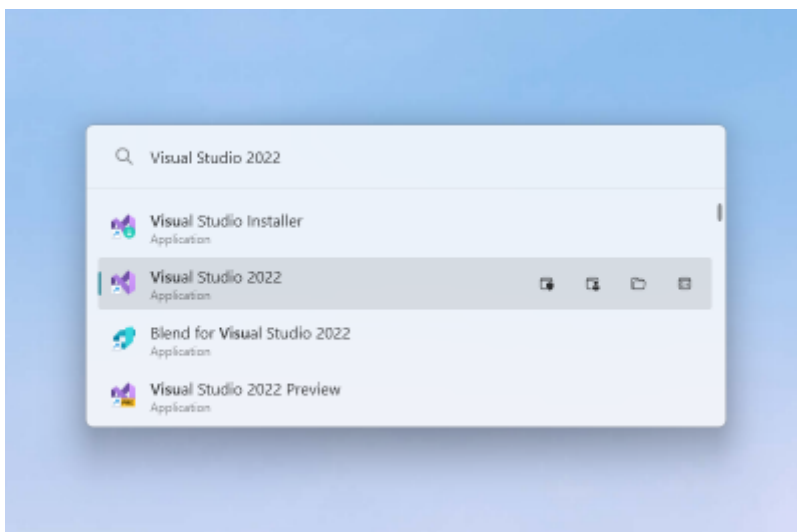
電源ディスプレイ

1つのポップアップから外部モニターの明るさ、コントラスト、ボリューム、入力ソース、色温度を制御します。



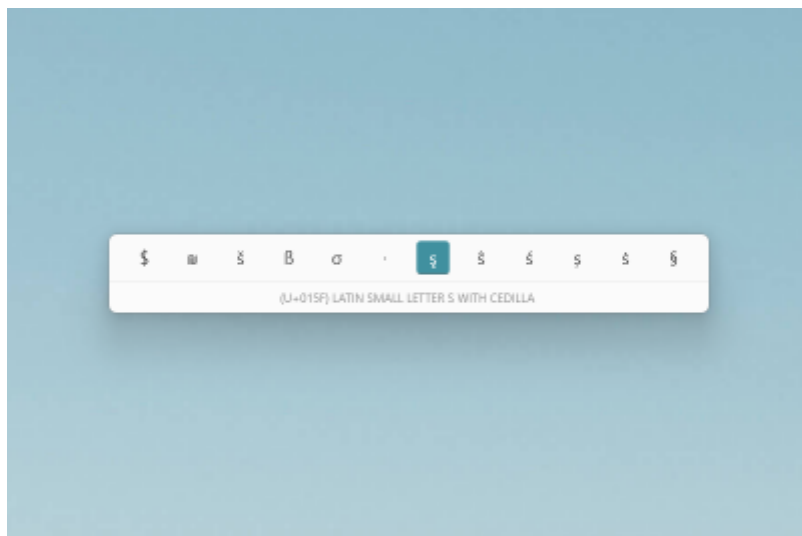
PowerRename

検索/置換、正規表現、および元に戻るサポートを使用して、一括名前変更を実行します。



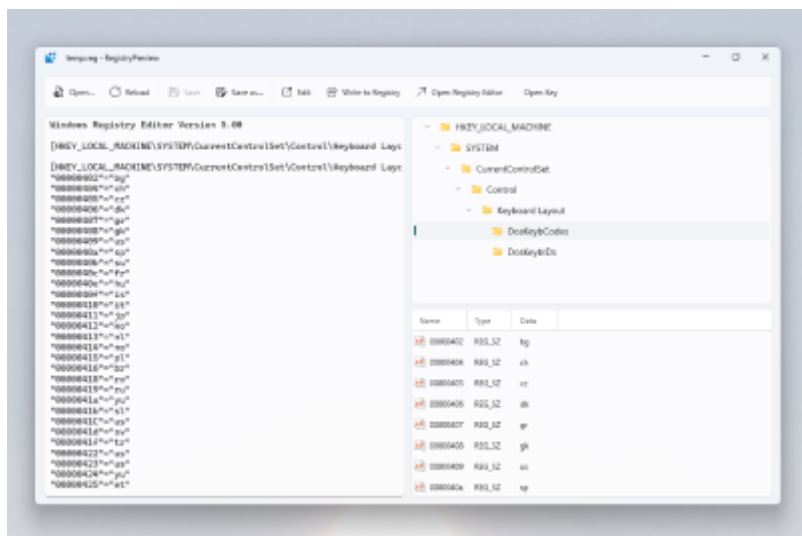
PowerToys Run

プラグインをサポートするクイックランチャーですぐにアプリを検索して開きます。



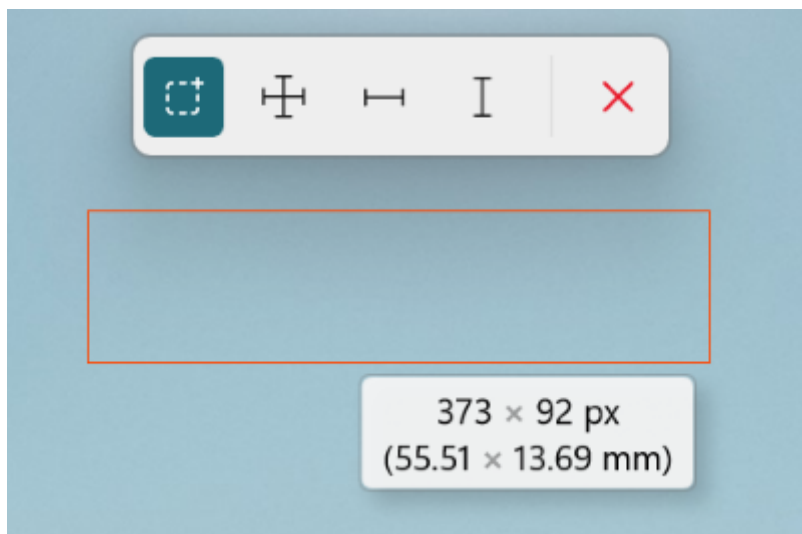
クイック アクセント

キーボードでサポートされていない場合にアクセント記号付き文字を入力する別の方法です。



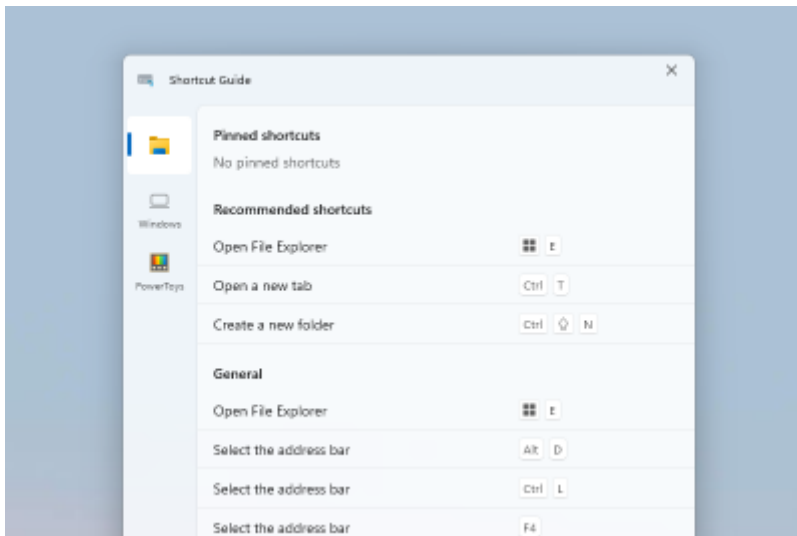
レジストリ プレビュー

Windows レジストリ ファイルを視覚化して編集します。



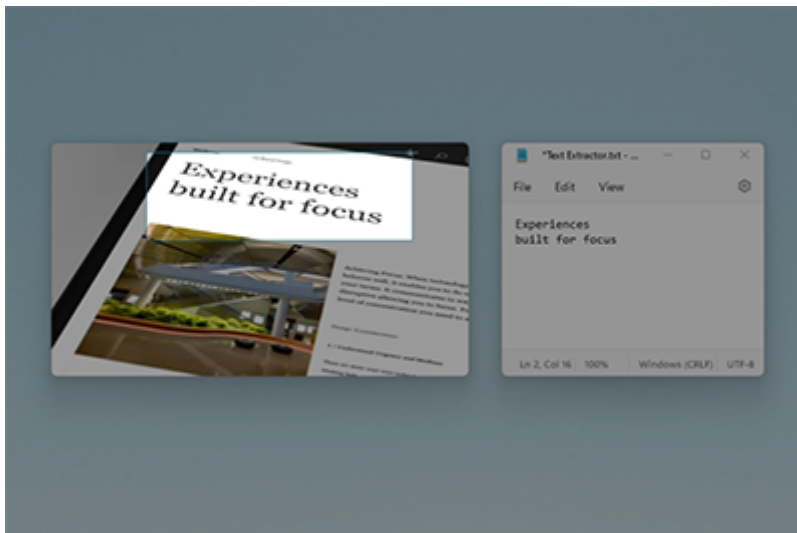
画面ルーラー

画像のエッジ検出に基づいて、画面上のピクセルをすばやく測定します。



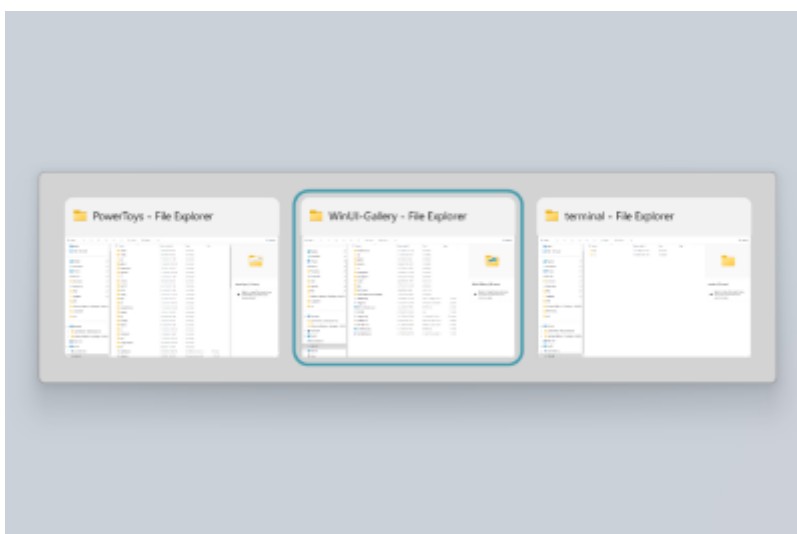
ショートカットガイド

Windowsとお気に入りのアプリのキーボード ショートカットを 1 か所で検索します。



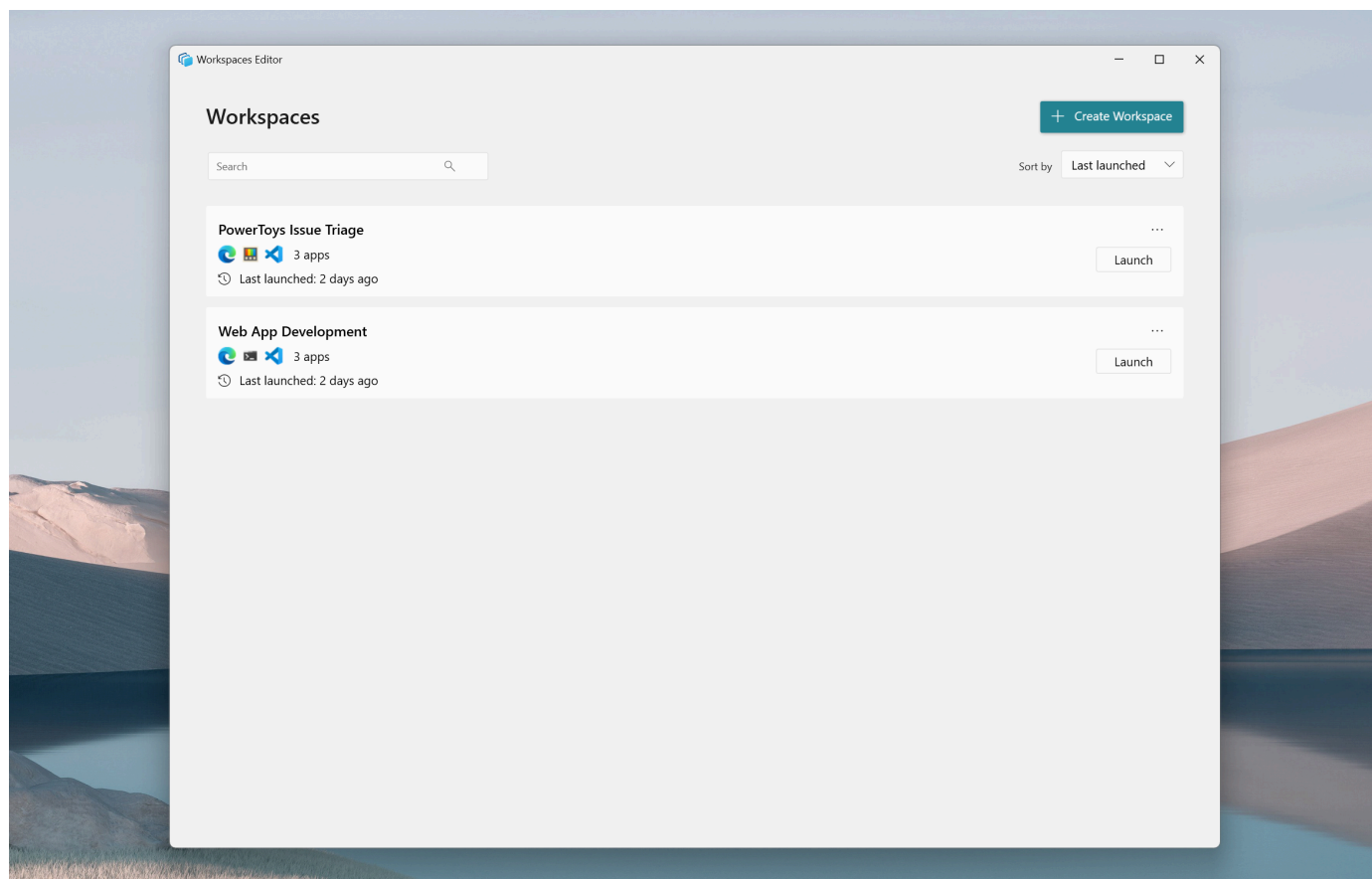
Text Extractor

OCR を使用して、画面上の任意の場所からテキストをコピーします。



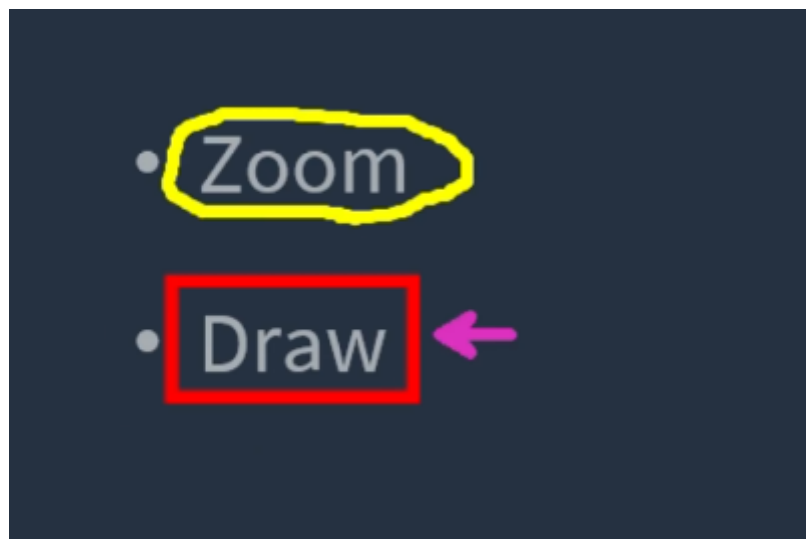
ウィンドウ ホッパー

現在フォーカスされているアプリに属しているウィンドウ間のサイクル。



ワークスペース

1 回のクリックでカスタムの位置にアプリケーションのセットを起動します。



Zoomit

技術的なプレゼンテーション用の画面ズーム、注釈、記録ツール。

PowerToys への貢献方法

PowerToys に関する投稿をお寄せください。PowerToys 開発チームは、パワー ユーザー コミュニティと協力して、ユーザーがワークフローを最大限に活用するのに役立つ最高の Windows ユ

ーティリティを構築することに興奮しています。投稿にはさまざまな方法があります。

- [技術仕様](#) を記述する
- [設計の概念または推奨事項](#) を送信する
- [ドキュメント](#) に投稿する
- [ソースコード](#) のバグを特定して修正する
- [新しい機能と PowerToy ユーティリティをコーディングする](#)

投稿する機能の作業を開始する前に、[共同作成者ガイド](#) を参照してください。PowerToys チームがお客様と協力して最適なアプローチを特定し、機能の開発全体のガイダンスを提供して指導するため、作業の無駄や重複を回避できます。

PowerToys開発を開始し、発生する可能性がある問題を解決する方法の詳細については、[PowerToys](#) を参照してください。これらのドキュメントは最近更新され、開発環境のセットアップ方法、コードのビルド方法、ユーティリティの実行方法など、さまざまな PowerToys ユーティリティの開発とデバッグのガイダンスが提供されています。

問題とトラブルシューティング

GitHubの リポジトリの Issues タブで、既知 <の問題を検索したり、新しい問題を提出したりできます。発生している問題が見つからない場合は、[製品リポジトリ](#)でPowerToysできます。

Last updated on 2026/08/25

WinGet

WinGet (Windows パッケージ マネージャー) は、次を含む包括的な [パッケージ マネージャー ソリューション](#) です。

- [WinGet](#): Windows パッケージ マネージャーのコマンド ライン ツールとクライアント インターフェイス。コマンド [winget search](#)を使用して使用可能なパッケージを表示し、winget コマンドをさらに見つけることができます。[WinGet ツールを使用して、アプリケーションをインストールして管理](#)。
- [Windows パッケージ マネージャー](#)にパッケージを送信する: Windows デバイスでアプリケーションをホストおよびインストールするためのパッケージ 化サービス。
- [WinGet 構成ファイル](#): Windows パッケージ マネージャーの一連の手順を作成して、デバイスを手動で設定し、新しいプロジェクトにオンボードする手順を、信頼性が高く反復可能な 1 つのコマンドに統合します。WinGet 構成ファイルは、PowerShell Desired State Configuration (DSC) を YAML 形式の命令と WinGet パッケージと組み合わせて使用して、コンピューターのセットアップを処理します。

Windows パッケージ マネージャーは、以下のために役立つツールです。

- コマンド ラインを使ってソフトウェア アプリケーションを管理したい[開発者](#)。
- ソフトウェアを配布したい[独立系ソフトウェア ベンダー \(ISV\)](#)。
- [デバイスのセットアップを自動化し、セキュリティで保護された作業環境を維持するエンタープライズ組織](#)。

パッケージ マネージャーについて

パッケージ マネージャーは、ソフトウェアのインストール、アップグレード、アンインストール、および必要に応じて構成するために使用されるシステムまたはツールのセットです。ほとんどのパッケージ マネージャーは、開発者ツールを検出およびインストールするために設計されています。

理想的には、開発者は、パッケージ マネージャーを使用して、特定のプロジェクトのソリューションを開発するために必要なツールの前提条件を指定します。次に、パッケージ マネージャーは、宣言型の指示に従ってツールをインストールし、構成します。パッケージ マネージャーを使用すると、環境の準備に費やす時間を短縮でき、確実にマシンに同じバージョンのパッケージをインストールすることができます。

サードパーティのパッケージ マネージャーは、[WinGet コミュニティ リポジトリ](#) を利用して、ソフトウェア カタログのサイズを増やします。

開発者向けの WinGet

開発者は、winget コマンド ライン ツールを使用して WinGet を使用して、キュレーションされた一連のアプリケーションを検出、インストール、アップグレード、削除、構成します。インストールが完了すると、開発者は Windows ターミナル、PowerShell、またはコマンドプロンプトを使用して winget にアクセスできます。

詳細については、「[winget ツールを使用したアプリケーションのインストールと管理](#)」を参照してください。

WinGet のビデオ デモについては、「[Windows パッケージ マネージャーの概要](#)」を参照してください。

[Windows コマンド ライン ブログ](#) で、最新の WinGet のお知らせとバージョンの更新プログラムを確認します。

ISV ソフトウェア配布用 WinGet

独立系ソフトウェア ベンダー (ISV) は、ツールとアプリケーションを含むソフトウェア パッケージの配布チャネルとして WinGet を使用できます。ソフトウェア パッケージを WinGet コミュニティ リポジトリに送信するために、GitHub のオープン ソースの **WinGet コミュニティ リポジトリ** を提供します。このリポジトリでは、ISV が [パッケージ マニフェスト](#) をアップロードして、ソフトウェア パッケージが既定の winget ソースに含まれると見なされます。参照するマニフェストとパッケージは自動的に検証され、手動で確認することもできます。

詳細については、「[パッケージを WinGet コミュニティ リポジトリに送信する](#)」を参照してください。

WinGet for Enterprise Security

WinGet は、コマンド ラインを使用して、複数のマシン間でアプリケーションをインストールおよび管理するために使用できます。IT 管理者やセキュリティ アナリストのように企業の作業環境を設定する担当者は、全員の作業マシンで一貫したレベルのセキュリティ設定を維持することを目標として [Microsoft Intune](#) を使い、「グループ ポリシー」設定を使ってセキュリティを管理することもできます。

継続的なセキュリティ更新プログラムを維持するために、WinGet は Microsoft Store を使用してリリースされ、["msstore" ソース](#) を使用して Microsoft Store からアプリケーションをインストールし、「証明書のピン留め」を適用して、接続がセキュリティで保護され、適切なエンドポイントで確立されるようにします。

企業組織によって適用されるグループ ポリシーは、WinGet と Microsoft Store ソースの間のファイアウォールを介して SSL 検査を使用している可能性があります。これにより、接続工

ラーが WinGet クライアントに表示されます。

このため、Winget (デスクトップ アプリ インストーラー経由) では、"BypassCertificatePinningForMicrosoftStore" というポリシー設定がサポートされています。このポリシーは、Microsoft Store ソースへの接続を開始するときに、WinGet が既知の Microsoft Store 証明書と一致する Microsoft Store 証明書ハッシュを検証するかどうかを制御します。このポリシーのオプションは次のとおりです。

- **構成されていません (既定値):** このポリシーを構成しない場合、Windows パッケージ マネージャーの管理者設定が適用されます。特に変更する必要がない場合は、このポリシーを構成しない既定値のままにすることをお勧めします。
- **有効:** このポリシーを有効にすると、Windows パッケージ マネージャーは Microsoft Store 証明書の検証をバイパスします。
- **無効:** このポリシーを無効にすると、Windows パッケージ マネージャーは Microsoft Store ソースと通信する前に、使われる Microsoft Store 証明書が有効であり、Microsoft Store に属していることを検証します。

"証明書のピン留め" により、パッケージ マネージャーから Microsoft Store への接続を確実にセキュリティで保護することができます。これは、クライアント (ユーザー) とサーバー (アプリケーション) の間に第三者を挿入して通信フローを密かに傍受し、ログイン資格情報などの機密データを盗む中間者 (MITM) 攻撃などの攻撃に関連するリスクを回避するために役立ちます。"証明書のピン留め" を無効にする (バイパスを有効にする) と、組織がこの領域のリスクにさらされる可能性があるため、避けてください。

企業組織のグループ ポリシー設定の詳細については、[Microsoft Intune のドキュメント](#)を参照してください。

WinGet の追加のグループ ポリシー設定

WinGet には、グループ ポリシーを通じて追加の構成オプションが用意されており、IT 管理者は複数のデバイスの機能を管理および制御できます。これらの設定は、コンプライアンスと一貫性が重要なエンタープライズ環境に特に役立ちます。

Windows 11 以降では、WinGet 用の追加のグループ ポリシー テンプレートが各リリースに含まれています。これらのテンプレートは複数のサブカテゴリに分けられ、IT 管理者はツールの動作の重要な側面を構成できます。次に例を示します。

- **ソース管理:** 許可またはブロックするソースを指定します。
- **ローカル開発:** ユーザーが試験的な機能またはローカル マニフェストのインストールを有効にできるかどうかを制御します。
- **実行ポリシー:** コマンド ライン インターフェイスとプロキシ オプションのポリシーを設定します。

グループ ポリシー テンプレートをダウンロードするには:

1. [WinGet GitHub リリース](#) にアクセスしてください。
2. 使用するリリース バージョンを見つけます。
3. リリースアセットに含まれる `DesktopAppInstallerPolicies.zip` ファイルをダウンロードします。

ZIP ファイルには、ポリシーを展開するために必要な `.admx` ファイルと `.adml` ファイルが含まれています。 `DesktopAppInstallerPolicies.zip` ファイルをダウンロードしたら、次の手順を実行します。

1. ローカル コンピューター上の ZIP ファイルの内容を抽出します。
2. `.admx` ファイルをターゲット デバイス上の `C:\Windows\PolicyDefinitions` フォルダーにコピーします。
3. 対応する言語固有の `.adml` ファイルを、 `C:\Windows\PolicyDefinitions\en-US` などの適切なサブディレクトリにコピーします。
4. グループ ポリシー管理コンソール (GPMC) を開き、ポリシーを構成します。

❗ Note

Windows ドメイン コントローラーで作業する場合は、グループ ポリシー テンプレートを中央ストアに格納できます。 詳細な手順については、「[Windows でグループ ポリシー管理用テンプレートの中央ストアを作成および管理する方法](#)」を参照してください。

WinGet の各リリースで、新しいグループ ポリシー設定が導入される場合があります。 環境を常に最新の状態に保つには、次の手順を実行します。

- [WinGet GitHub リポジトリ](#) ページで更新プログラムを定期的を確認します。
- リリース ノートで、ポリシー テンプレートに対する変更や追加について確認します。

Windows 開発者向け構成

Windows Developer Configurations は、新しい Windows コンピューターを 1 つのコマンドですぐにコードに変換できる状態にする、キュレーションされたオープンソースの構成ファイルのコレクションです。各構成は、再実行しても安全な宣言型ファイルです。特定のシナリオ (完全な開発者ワークステーション、快適な WSL シェル、または単一言語ツールチェーン) のパッケージ、OS 設定、およびインストール後の手順について説明しているため、インストーラーをクリックしたり、カスタム スクリプトを維持したりすることなく、任意のコンピューターで環境を再構築できます。

構成は github.com/microsoft/WindowsDeveloperConfig でオープンソースされ、変更が行われるたびに自動的にテストされます。サポートされているツールチェーンの最新のコマンド、オプション、完全な一覧については、リポジトリ README を参照してください。このページでは、基本について説明します。

🔗 [GitHub 上の Windows 開発者向け構成ファイル](#)

利用可能な構成

開発構成には 3 種類あります。必要なものに一致するものを選択します。正確なコマンドとオプションについては、[リポジトリ README](#) を参照してください。

Windows 開発構成

新しい Windows 11 を 1 つのコマンドでクリーンで気を散らさのない開発者ワークステーションに変換する 1 つの構成。開発者ツールの基本的な一式をインストールし、推奨される Windows の設定 (ダークテーマ、開発者モード、ファイル エクスプローラーやスタート メニュー/タスクバーの整理、同様の作業環境の整備など) を適用し、必要な再起動を含めて Ubuntu による WSL の初期設定を行います。非対話型で、既存のマシンで再実行しても安全です。

WSL コンフォート

より優れた Windows + WSL シェル エクスペリエンスのための対話型セットアップ。Windows 側では、WSL、ディストリビューション、Nerd Font、テーマ付き Windows ターミナル プロファイルがインストールされます。Linux 側では、オプション (シェル、プロンプト、最新の CLI ツール、クリップボード shim など) を選択してシェル自体を構成します。Linux ハーフはスタンドアロンであり、任意の Ubuntu ホストで直接実行できます。

ワークロード (単一言語ツールチェーン)

1 つの言語スタック (TypeScript、Python、.NET、Go、Java、Rust、PHP、WinForms、WinUI 3 など) が必要な場合は、各ワークロードに独自の構成と、それを適用して現在のセッションで `PATH` を更新する小さな shim が付属します。現在のリストとワークロードごとの詳細は、[リポジトリ README](#) に格納されます。

関連するコンテンツ

- [WinGet 構成の概要](#)
- [WinGet 構成ファイルを作成する](#)
- [Windows 上の WSL](#)
- [PowerToys](#)

Last updated on 2026/06/18

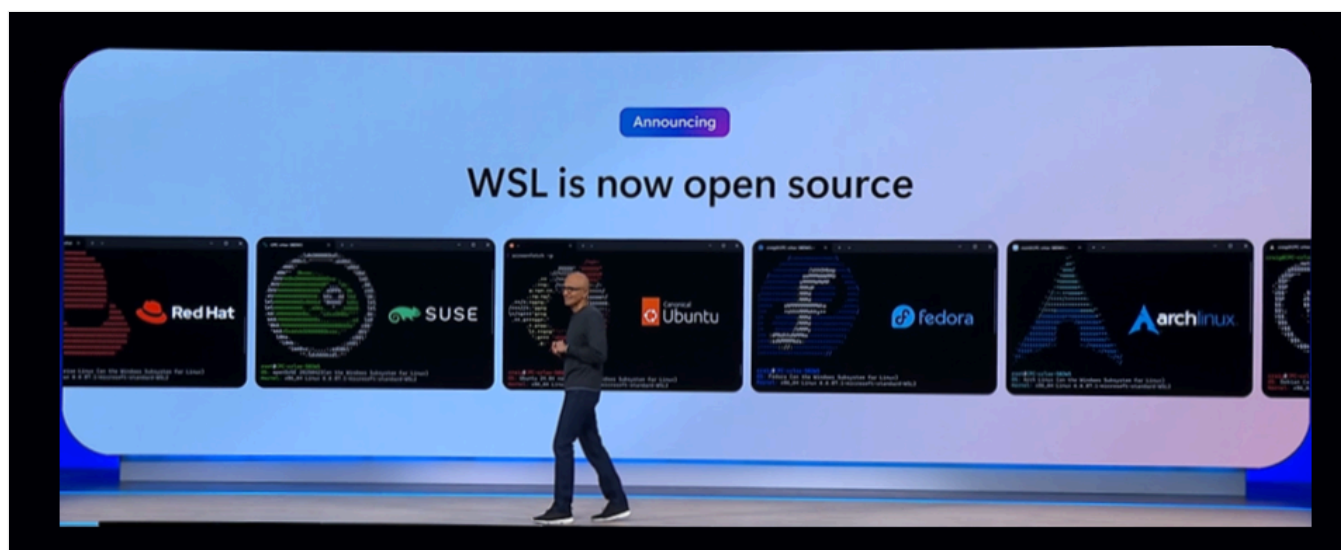
Windows Subsystem for Linux のドキュメント

Windows Subsystem for Linux (WSL) を使用すると、開発者は GNU/Linux 環境 (ほとんどのコマンドライン ツール、ユーティリティ、アプリケーションを含む) を Windows 上で直接実行でき、従来の仮想マシンやデュアルブートセットアップのオーバーヘッドは発生しません。

WSL のインストール

詳細情報

- [Windows Subsystem for Linux \(WSL\) とは](#)
- [Windows Subsystem for Linux がオープン ソースになりました](#) ↗
- [WSL 2 の新機能](#)
- [WSL 1 と WSL 2 の比較](#)
- [よく寄せられる質問](#)



概要

- [WSL のインストール](#)
- [Windows Server に Linux をインストールする](#)
- [手動インストール手順](#)
- [WSL 開発環境を設定するためのベスト プラクティス](#)

Windows Insider プログラムに参加して WSL プレビュー機能を試す

WSL の最新の機能または更新プログラムを試すには、[Windows Insider プログラム](#)に参加してください。Windows Insider に参加したら、Windows 設定メニュー内からプレビュー ビルドを受け取るチャンネルを選択できます。次の項目から選択できます。

- 開発チャンネル: 最新の更新プログラムですが、安定性は低いです。
- ベータ チャンネル: 開発チャンネルよりも信頼性の高いビルドを早期導入する場合に最適です。
- リリース プレビュー チャンネル: 次のバージョンの Windows で、一般に公開される直前のプレビュー修正プログラムと主な機能。

チーム ブログ

- [ビデオとブログのコレクションを含む概要の投稿](#)
- [Command-Line ブログ](#) (アクティブ)
- [Linux 用 Windows サブシステムのブログ](#) (履歴)

フィードバックを提供する

- [GitHub イシュー トラッカー: WSL](#)
- [GitHub イシュー トラッカー: WSL ドキュメント](#)

関連するビデオ

WSL の基本

1. [Linux 用 Windows サブシステム \(WSL\) とは](#) | 開発に関する 1 つの質問 (0:40)
2. [私は Windows 開発者です。WSL を使用する必要がある理由](#) | 開発に関する 1 つの質問 (0:58)
3. [私は Linux 開発者です。WSL を使用する必要がある理由](#) | 開発に関する 1 つの質問 (1:04)
4. [Linux とは](#) | 開発に関する 1 つの質問 (1:31)
5. [Linux ディストリビューションとは](#) | 開発に関する 1 つの質問 (1:04)
6. [WSL は仮想マシンまたはデュアル ブートとどのように異なりますか?](#) | 1 つの開発に関する質問
7. [Windows Subsystem for Linux が作成された理由](#) | 開発に関する 1 つの質問 (1:14)
8. [WSL でコンピューター上のファイルにアクセスするにはどうすればよいですか?](#) | 開発に関する 1 つの質問 (1:41)
9. [WSL と Windows の統合方法](#) | 開発に関する 1 つの質問 (1:34)

10. [ターミナルのホーム ディレクトリで起動するように WSL ディストリビューションを構成するにはどうすればよいですか?](#) [🔗](#) |開発に関する 1 つの質問 (0:47)
11. [スクリプトに WSL を使用できますか?](#) [🔗](#) |開発に関する 1 つの質問 (1:04)
12. [Windows で Linux ツールを使用する理由](#) [🔗](#) |開発に関する 1 つの質問 (1:20)
13. [WSL では、Microsoft Store 以外のディストリビューションを使用できますか?](#) [🔗](#) |開発に関する 1 つの質問 (1:03)

WSL デモ

1. [WSL2: Windows Subsystem for Linux でのコードの高速化](#) [🔗](#) |タブとスペース (13:42)
2. [WSL: Linux GUI アプリを実行する](#) [🔗](#) |タブとスペース (17:16)
3. [WSL 2: USB デバイスを接続する](#) [🔗](#) |タブとスペース (10:08)
4. [WSL 2 を使用した GPU 高速機械学習](#) [🔗](#) |タブとスペース (16:28)
5. [Visual Studio Code: SSH、VM、WSL を使用したリモート開発](#) [🔗](#) |タブとスペース (29:33)
6. [Windows 開発ツールの更新: WSL、ターミナル、パッケージ マネージャーなど](#) [🔗](#) |タブとスペース (20:46)
7. [WSL を使用してNodeJS アプリをビルド](#) [🔗](#) する |強調表示 (3:15)
8. [WSL 2 の新しいメモリ再利用機能](#) [🔗](#) |デモ (6:01)
9. [Windows での Web 開発 \(2019 年\)](#) [🔗](#) |デモ (10:39)

WSL の徹底解析

1. [Windows 11 の WSL - Craig Loewen と Scott Hanselman のデモ](#) [🔗](#) |Windows 水曜日 (35:48)
2. [WSL および Linux ディストリビューション - Hayden Barnes と Kayla シナモン](#) [🔗](#) | Windows Wednesday (37:00)
3. [Oh My Posh と WSL Linux ディストリビューションを使用してターミナルをカスタマイズ](#) [🔗](#) する |Windows 水曜日 (33:14)
4. [Web 開発、コンテンツ作成、WSL に関する Web 開発の Sarah Tamsin と Craig Loewen のチャット](#) [🔗](#) |開発パースペクティブ (12:22)
5. [WSL が Windows から Linux ファイルにアクセスする方法](#) [🔗](#) |詳細 (24:59)
6. [Linux 用 Windows サブシステムのアーキテクチャ: 詳細](#) [🔗](#) |ビルド 2019 (58:10)

Windows ターミナルとは

Windows ターミナルは、コマンド プロンプト、PowerShell、bash (Windows Subsystem for Linux (WSL) 経由) など、既に好きなコマンド ライン シェル用の最新のホスト アプリケーションです。その主な機能には、複数のタブ、ペイン、Unicode と UTF-8 文字のサポート、GPU 高速テキスト レンダリング エンジン、独自のテーマを作成し、テキスト、色、背景、ショートカットをカスタマイズする機能があります。

Windows ターミナルをインストールする

ⓘ Note

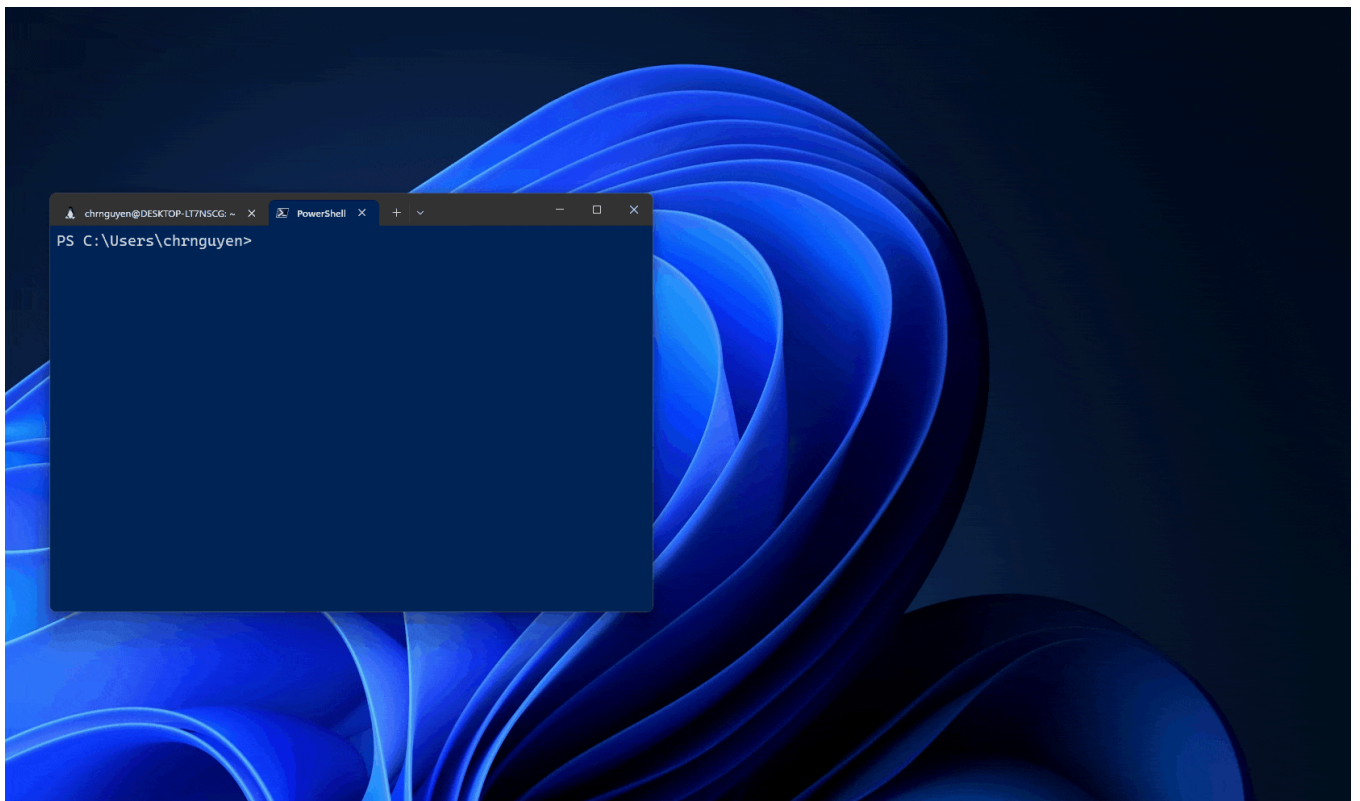
一般的な情報については、Scott Hanselman の記事を参照してください。 [コンソール、ターミナル、シェルの違いは何ですか。](#) または Rich Turner のビデオ [コマンドラインシェルとは何ですか](#)?

さまざまなコマンド ライン アプリケーションをサポートする複数のプロファイル

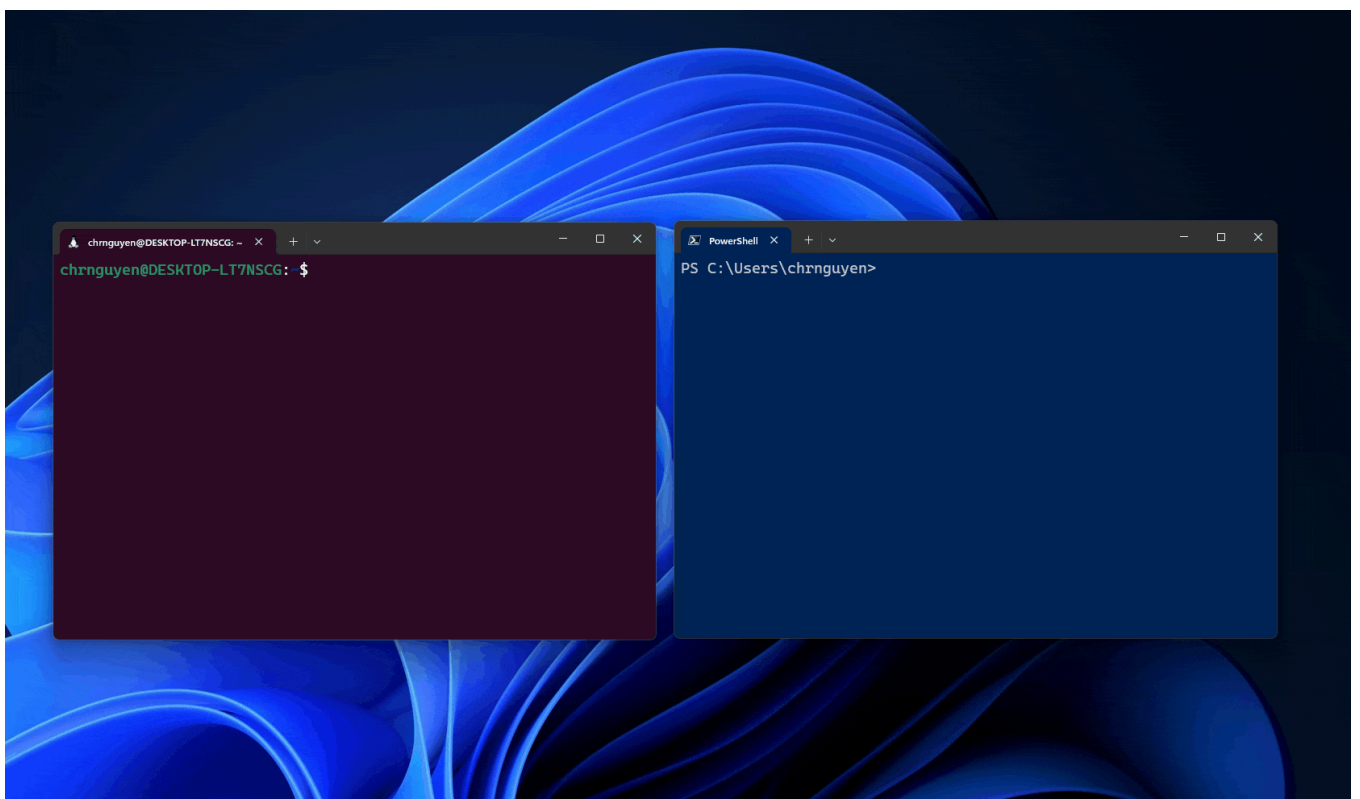
Windows ターミナル内のコマンド ライン インターフェイスを使用して、任意のアプリケーションを実行できます。この機能には、PowerShell とコマンド プロンプトから Azure Cloud Shell、および Ubuntu や Oh-My-Zsh などの WSL ディストリビューションまで、あらゆるものが含まれます。

タブのティアアウト

Windows ターミナルでタブを破棄し、新しいウィンドウを作成できます。



既存のウィンドウにタブをドラッグ アンド ドロップすることもできます。



カスタマイズされたスキームと構成

さまざまな配色と設定を使用するように Windows ターミナルを構成できます。クールなテーマでプロンプトをカスタマイズする方法については、「[チュートリアル: Oh My Posh を使用](#)

して [PowerShell または WSL のカスタム プロンプトを設定](#) する」を参照してください。独自の配色を作成する方法については、[[配色](#)] ページを参照してください。

カスタム アクション

Windows ターミナルには、より自然な感じにするために使用できるさまざまなカスタム コマンドが用意されています。特定のキーボード ショートカットが気に入らない場合は、任意のキーボード ショートカットに変更します。

たとえば、コマンド ラインからテキストをコピーするための既定のショートカットは、`Ctrl + Shift + C` です。このショートカットは、`Ctrl + 1` または任意に変更できます。新しいタブを開くには、既定のショートカットは `Ctrl + Shift + T` ですが、このショートカットを `Ctrl + 2` に変更したい場合があります。開いているタブ間を反転する既定のショートカットは、`Ctrl + Tab` です。このショートカットを `Ctrl + -` に変更し、それを使用して新しいタブを作成することもできます。

ショートカットのカスタマイズの詳細については、「[アクション](#)」ページを参照してください。

Unicode および UTF-8 文字のサポート

Windows ターミナルでは、さまざまな言語の絵文字や文字などの Unicode 文字と UTF-8 文字を表示できます。

GPU 高速テキスト レンダリング

Windows ターミナルは GPU を使用してそのテキストをレンダリングするため、既定の Windows コマンド ライン エクスペリエンスよりもパフォーマンスが向上します。

背景画像のサポート

Windows ターミナル ウィンドウ内に背景画像と GIF を含めることができます。プロファイルに背景画像を追加する方法については、「[プロファイル - 外観](#)」ページを参照してください。

コマンド ライン引数

コマンド ライン引数を使用して、特定の構成で起動するように Windows ターミナルを設定できます。新しいタブで開くプロファイル、選択するフォルダー ディレクトリを指定し、分割ウィンドウウィンドウでターミナルを開き、フォーカスするタブを選択できます。

たとえば、3つのウィンドウで PowerShell から Windows ターミナルを開くには、左側のウィンドウでコマンド プロンプト プロファイルを実行し、右側のウィンドウを PowerShell と WSL を実行している既定のプロファイルの間で分割して、次のように入力します。

PowerShell

```
wt -p "Command Prompt" `; split-pane -p "Windows PowerShell" `; split-pane -H  
wsl.exe
```

コマンド ライン引数の設定の詳細については、「[コマンド ライン引数](#)」ページを参照してください。

Last updated on 2025/11/20

Windowsに JavaScript フレームワークをインストールする

このガイドは、Node.js、React.js、Vue.jsなど、Windowsでの JavaScript フレームワークの使用を開始するのに役立ちます。

開発環境をインストールしてセットアップする JavaScript フレームワークを選択する



Node.js の概要

Node.js でできることと、Node.js 開発環境を設定する方法について説明します。

- Windows
- [WSL へのインストール](#)



React の概要

React でできることと、React 開発環境を設定する方法について説明します。

- [Web アプリを構築するためのWindowsにインストールします](#)
- [Web アプリを構築するために WSL にインストールする](#)
- [デスクトップ アプリをビルドするためのWindowsにインストールします](#)



Vue.js の概要

Vue.js でできることと、Vue.js 開発環境を設定する方法について説明します。

- Windows
- [WSL へのインストール](#)

Last updated on 2026/04/21

Windows で Node.js を始める

Node.js は、Chrome の V8 エンジン上に構築されたオープンソースのクロスプラットフォーム JavaScript ランタイムです。ブラウザーの外部 (サーバー、ローカル コンピューター、またはスクリプト) で JavaScript を実行できるため、最新の Web ツールとバックエンド開発の基盤となります。

Node.jsを使用する理由

- **フルスタック JavaScript:** クライアント コードとサーバー コードの両方を同じ言語で記述します。
- **高速でスケーラブル:** イベント ドリブンの非ブロッキング I/O により、Node.js リアルタイム アプリや API に最適です。
- **大規模なエコシステム:** npm は、ほぼすべての開発タスクに対して何百万ものパッケージをホストします。
- **最新のツールに必要:** ほとんどの JavaScript フレームワーク (React、Vue、Angular、Next.js) は、ビルド ツールを実行する Node.js に依存します。

Windows に Node.js をインストールする

好みの環境を選択します。

- **Node.js を Windows にインストール** する — バージョン マネージャー (nvm-windows) を使用して Windows に直接インストールします。
- **WSL に Node.js をインストール** する — Linux 用 Windows サブシステムにインストールします (Linux サーバーにデプロイする場合、または Linux ベースのツールを使用する場合に推奨)。

チュートリアルと学習リソース

- nodejs.org/en/learn — 公式 Node.js ファースト ステップ ガイドとチュートリアル。
- **Node.jsを使用して JavaScript アプリケーションを構築** する — 初心者向けの Microsoft Learn ラーニング パス。
- **VS Code のNode.js チュートリアル** — Visual Studio Code で簡単な Node.js アプリを構築します。

Windows Subsystem for Linux (WSL2) に Node.js をインストールする

Linux 環境で Node.js を使用する場合は、このガイドを使用して、Windows Subsystem for Linux に Node.js をインストールできます (WSL 2 が推奨バージョンです)。

インストールする場所と、ネイティブ Windows 環境と Linux (WSL 2) 環境で Node.js を使用して開発するかどうかを決定するときは、次の点を考慮してください。

- **スキル レベル:** Node.js を使った開発を初めて行い、学習できるようにすばやく稼働したい場合は、[Windows に Node.js をインストール](#)します。Windows に Node.js をインストールして使用すると、WSL を使用するよりも初心者向けの複雑な環境が提供されます。
- **コマンド ライン クライアント ツール:** PowerShell を使用する場合は、Windows で Node.js を使用します。Bash を使用する場合は、Node.js on Linux (WSL 2) を使用します。
- **運用サーバー:** Node.js アプリを Windows Server に展開する予定の場合は、Windows で Node.js を使用します。Linux Server にデプロイする予定の場合は、Node.js on Linux (WSL 2) を使用します。WSL を使用すると、好みの Linux ディストリビューション (既定として Ubuntu を使用) をインストールし、開発環境 (コードを記述する場所) と運用環境 (コードがデプロイされているサーバー) の間の一貫性を確保できます。
- **パフォーマンス速度とシステム呼び出しの互換性:** Linux と Windows のパフォーマンスでは継続的な議論と開発がありますが、Windows マシンを使用する場合の重要な点は、Node.js インストールしたのと同じファイル システムに開発プロジェクト ファイルを保持することです。Windows ファイル システムに Node.js をインストールする場合は、ファイルを Windows ドライブ (C:/ など) に保持します。Node.js を Linux ディストリビューション (Ubuntu など) にインストールする場合は、使用しているディストリビューションに関連付けられている Linux ファイル システム ディレクトリにプロジェクト ファイルを保持します。(WSL 配布コマンド ラインから「`explorer.exe` .」と入力し、Windows エクスプローラーを使用してディレクトリを参照します)。
- **Docker コンテナ:** Docker コンテナを使用して Windows でプロジェクトを開発する場合は、[Windows に Docker Desktop をインストール](#) [🔗](#) することをお勧めします。Linux ワークスペースで Docker を使用する場合は、Linux と Windows の両方のビルド スクリプトを維持する必要があるように、[WSL 2 で Windows 用 Docker Desktop を設定](#) する方法に関するページを参照してください。

Linux 用 Windows サブシステムをインストールする

Node.jsで Linux 開発環境を使用する場合は、[WSL インストールのドキュメント](#) を参照してください。これらの手順には、Linux ディストリビューションの選択 (Ubuntu が既定) と、Windows Subsystem for Linux (WSL 2 が既定の推奨バージョン) のバージョンが含まれます。必要に応じて、複数の Linux ディストリビューションをインストールできます。

WSL 2 と Linux ディストリビューションをインストールしたら、Linux ディストリビューション (Windows ターミナルの一覧または Windows のスタートメニューにあります) を開き、`lsb_release -dc` コマンドを使用してバージョンとコード名を確認します。

最新のパッケージを使用できるように、インストール直後を含め、Linux ディストリビューションを定期的に更新することをお勧めします。Windows はこの更新を自動的に処理しません。使用中のディストリビューションを更新するには、コマンド `sudo apt update && sudo apt upgrade` を使用します。

Windows ターミナル

Windows ターミナルは、Linux コマンドライン、Windows コマンドプロンプト、PowerShell、Azure CLI、または使用するものをすばやく切り替えることができるように、複数のタブを実行できる強化されたコマンドライン シェルです。また、カスタム キー バインド (タブを開いたり閉じたり、コピー+貼り付けしたりするためのショートカット キー) を作成したり、検索機能を使用したり、テーマ (配色、フォント スタイルとサイズ、背景画像/ぼかし/透明度) でターミナルをカスタマイズしたりすることもできます。[詳細については、Windows ターミナルのドキュメントを参照してください。](#)

nvm、node.js、および npm をインストールする

Windows または WSL にインストールするかどうかを選択するだけでなく、Node.jsをインストールするときに追加の選択肢があります。バージョンの変更が非常に早い場合は、バージョン マネージャーを使用することをお勧めします。作業しているさまざまなプロジェクトのニーズに基づいて、複数のバージョンの Node.js を切り替える必要があります。ノードバージョン マネージャーは、より一般的に nvm と呼ばれ、Node.jsの複数のバージョンをインストールする最も一般的な方法です。nvm をインストールし、それを使用して Node.js とノード パッケージ マネージャー (npm) をインストールする手順について説明します。次 [セクション](#)で説明するように、[検討すべき代替バージョン マネージャー](#) があります。

① 重要

バージョン マネージャーをインストールする前に、Node.js または npm の既存のインストールをオペレーティング システムから削除することをお勧めします。インストールの種類が異なると、奇妙で混乱する競合が発生する可能性があるためです。たとえば、

Ubuntu の `apt-get` コマンドでインストールできる Node のバージョンは現在古くなっています。以前のインストールを削除する方法については、「[ubuntu から nodejs を削除する方法](#)」を参照してください。

NVM のインストールに関する最新情報については、[GitHub の NVM リポジトリでのインストールと更新に関するページ](#)を参照してください。

1. Ubuntu コマンド ライン (または任意のディストリビューション) を開きます。
2. cURL (コマンド ラインでインターネットからコンテンツをダウンロードするために使用されるツール) をインストールします。 `sudo apt-get install curl`

3. nvm をインストールする (以下を使用して): `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/master/install.sh | bash`

⚠ Note

cURL を使用して新しいバージョンの NVM をインストールすると、古いものが置き換えられ、NVM を使用してインストールしたノードのバージョンはそのまま残ります。詳細については、[NVM の最新リリース情報については、GitHub プロジェクトページ](#)を参照してください。

4. インストールを確認するには、`command -v nvm` を入力します。これにより 'nvm' が返されるはずです。'コマンドが見つかりません' またはまったく応答がない場合は、現在のターミナルを閉じて再度開き、もう一度試してください。詳細については、[nvm github リポジトリを参照してください](#)。
5. 現在インストールされている Node のバージョンを一覧表示します (この時点ではなしにする必要があります)。 `nvm ls`

```
mattwojo@MININT-LOBGCR8: ~  
mattwojo@MININT-LOBGCR8:~$ nvm --version  
0.34.0  
mattwojo@MININT-LOBGCR8:~$ nvm ls  
->               system  
iojs -> N/A (default)  
node -> stable (-> N/A) (default)  
unstable -> N/A (default)  
lts/* -> lts/dubnium (-> N/A)  
lts/argon -> v4.9.1 (-> N/A)  
lts/boron -> v6.17.1 (-> N/A)  
lts/carbon -> v8.16.1 (-> N/A)  
lts/dubnium -> v10.16.3 (-> N/A)  
mattwojo@MININT-LOBGCR8:~$ nvm use node  
N/A: version "node -> N/A" is not yet installed.
```

6. Node.jsの現在の LTS バージョンと安定した LTS バージョンの両方をインストールします。後の手順では、`nvm` コマンドを使用してアクティブなバージョンの Node.js を切り替える方法について説明します。

- Node.js の現在の安定した LTS リリースをインストールします (運用環境のアプリケーションに推奨)。 `nvm install --lts`
- Node.js の現在のリリースをインストールします (最新の Node.js の機能と機能強化をテストするためですが、問題が発生する可能性が高くなります)。 `nvm install node`

7. インストールされている Node のバージョンを一覧表示します: `nvm ls` ...インストールした 2 つのバージョンが一覧表示されます。

```
mattwojo@MININT-LOBGCR8: ~  
mattwojo@MININT-LOBGCR8:~$ nvm ls  
->      v10.16.3  
        v12.9.0  
        system  
default -> node (-> v12.9.0)  
node    -> stable (-> v12.9.0) (default)  
stable  -> 12.9 (-> v12.9.0) (default)  
iojs    -> N/A (default)  
unstable -> N/A (default)  
lts/*   -> lts/dubnium (-> v10.16.3)  
lts/argon -> v4.9.1 (-> N/A)  
lts/boron -> v6.17.1 (-> N/A)  
lts/carbon -> v8.16.1 (-> N/A)  
lts/dubnium -> v10.16.3  
mattwojo@MININT-LOBGCR8:~$
```

8. Node.js がインストールされ、`node --version` によって現在のデフォルトバージョンであることを確認します。次に、npm があることを確認します。`npm --version` (`which node` または `which npm` を使用して、既定のバージョンで使用されているパスを確認することもできます)。
9. プロジェクトに使用する Node.js のバージョンを変更するには、新しいプロジェクトディレクトリ `mkdir NodeTest` を作成し、ディレクトリ `cd NodeTest` を入力してから、「`nvm use node`」と入力して現在のバージョンに切り替えるか、を入力して LTS バージョンに切り替えます。`nvm use v8.2.1` など、インストールした追加のバージョンに特定の番号を使用することもできます。(使用可能なすべてのバージョンの Node.js 一覧表示するには、次のコマンドを使用します: `nvm ls-remote`)。

NVM を使用して Node.js と NPM をインストールする場合は、SUDO コマンドを使用して新しいパッケージをインストールする必要はありません。

代替バージョン管理ツール

現在、nvm はノードの最も一般的なバージョン マネージャーですが、考慮すべき選択肢がいくつかあります。

- [n](#) は、少し異なるコマンドで同じことを実行し、bash スクリプトではなく `nvm` を使用してインストールされる、長年にわたる `npm` の代替手段です。
- [fnm](#) は新しいバージョン マネージャーであり、`nvm` よりもはるかに高速であると主張しています。([Azure Pipelines](#) も使用します)。

- [Volta](#) は LinkedIn チームの新しいバージョン マネージャーで、速度とクロスプラットフォームのサポートが向上していると主張しています。
- [asdf-vm](#) は、ike gvm、nvm、rbenv、pyenv (その他) など、複数の言語の単一の CLI です。
- [nvs](#) (ノード バージョン スイッチャー) は、`nvm` する機能を備えたクロスプラットフォームの代替手段です。

Visual Studio Code をインストールする

Node.js プロジェクト用の [リモート開発拡張機能パック](#) で Visual Studio Code を使用することをお勧めします。これにより、VS Code が "クライアント サーバー" アーキテクチャに分割され、クライアント (VS Code ユーザー インターフェイス) が Windows オペレーティングシステム上で実行され、サーバー (コード、Git、プラグインなど) が WSL Linux ディストリビューションで "リモート" で実行されます。

ⓘ Note

この "リモート" シナリオは、慣れている場合とは少し異なります。WSL では、Windows オペレーティングシステムとは別に、ローカル コンピューター上でプロジェクトコードが実行されている実際の Linux ディストリビューションがサポートされています。Remote-WSL 拡張機能は、Linux サブシステムとリモートサーバーの場合と同様に接続しますが、クラウドでは実行されていません。... これは、Windows と共に実行できるようにした WSL 環境のローカル コンピューター上で引き続き実行されます。

- Linux ベースの Intellisense と linting がサポートされています。
- プロジェクトは Linux で自動的にビルドされます。
- Linux で実行されているすべての拡張機能 ([ES Lint](#)、[NPM Intellisense](#)、[ES6 スニペットなど](#)) を使用できます。

IntelliJ、Sublime Text、Brackets などの他のコード エディターは、WSL 2 Node.js 開発環境でも動作しますが、VS Code と同じ種類のリモート機能を持たない場合があります。これらのコード エディターは、WSL 共有ネットワークの場所 (`\wsl$\Ubuntu\home`) へのアクセスに問題が発生する可能性があり、Windows ツールを使用して Linux ファイルをビルドしようとしても、これは望むものではありません。VS Code の Remote-WSL 拡張機能は、この互換性を自動的に処理します。他の IDE では、X サーバーのセットアップが必要になる場合があります。[WSL での GUI アプリの実行](#) (コード エディター IDE など) のサポートは近日公開予定です。

ターミナルベースのテキスト エディター (vim、emacs、nano) は、本体内からすばやく変更する場合にも役立ちます。記事 [Emacs、Nano、またはVim: あなたの Terminal-Based テキス](#)

[トエディタを賢く選択](#)、それぞれの違いや使用方法について少し説明しており、非常に分かりやすいです。

VS Code と Remote-WSL 拡張機能をインストールするには:

1. [Windows 用の VS Code をダウンロードしてインストールします](#)。VS Code は Linux でも使用できますが、Linux 用 Windows サブシステムは GUI アプリをサポートしていないため、Windows にインストールする必要があります。心配する必要はありません。リモート - WSL 拡張機能を使用して、Linux コマンドラインやツールと統合することができます。
2. [Remote - WSL 拡張機能](#) を VS Code にインストールします。これにより、統合開発環境として WSL を使用でき、互換性とパスを処理できます。[詳細については、こちらを参照してください](#)。

❗ 重要

VS Code が既にインストールされている場合は、[Remote - WSL 拡張機能](#) をインストールするために、[1.35 May リリース](#) 以降があることを確認する必要があります。オートコンプリート、デバッグ、lint などのサポートが失われるため、VS Code において、Remote - WSL 拡張機能なしで WSL を使用することはお勧めしません。豆知識: この WSL 拡張機能は `$HOME/.vscode-server/extensions` にインストールされます。

役に立つ VS Code 拡張機能

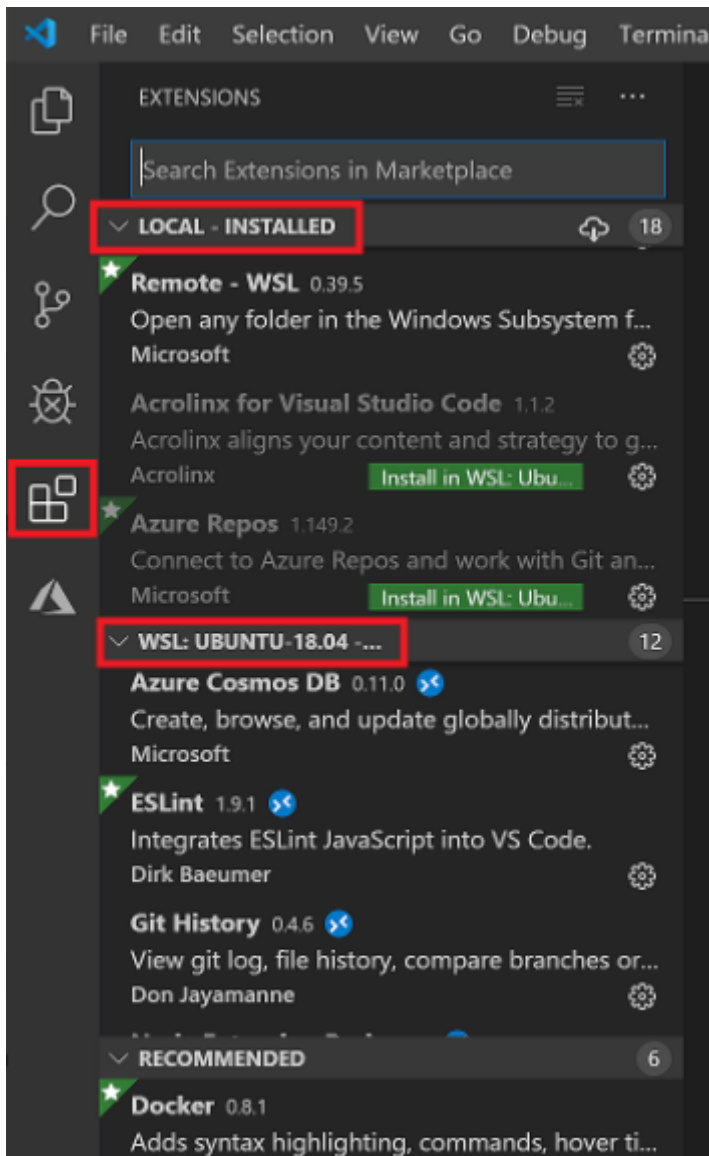
VS Code には、Node.js 開発用の多くの機能が付属していますが、[Node.js 拡張機能パック](#) で使用可能なインストールを検討する際に役立つ拡張機能がいくつかあります。それらをすべてインストールするか、最も役に立つと思われるかを選択してください。

Node.js 拡張機能パックをインストールするには:

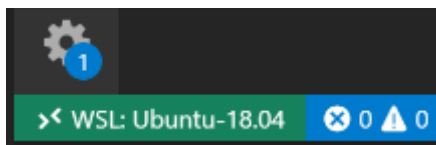
1. VS Code で **[拡張機能]** ウィンドウ (Ctrl + Shift + X) を開きます。

拡張機能ウィンドウは 3 つのセクションに分かれています (Remote-WSL 拡張機能をインストールしたため)。

- "ローカル - インストール済み": Windows オペレーティング システムで使用するためにインストールされた拡張機能。
- "WSL:Ubuntu-18.04-Installed": Ubuntu オペレーティング システム (WSL) で使用するためにインストールされた拡張機能。
- "推奨": 現在のプロジェクトのファイルの種類に基づいて VS Code によって推奨される拡張機能。



2. [拡張機能] ウィンドウの上部にある検索ボックスに、「**Node Extension Pack** (または探している拡張機能の名前)」と入力します。この拡張機能は、現在のプロジェクトを開いている場所にに応じて、VS Code のローカル インスタンスまたは WSL インスタンスにインストールされます。VS Code ウィンドウの左下隅にあるリモート リンク (緑色) を選択するとわかります。リモート接続を開いたり閉じたりするオプションが表示されます。"WSL:Ubuntu-18.04" 環境に Node.js 拡張機能をインストールします。



検討する必要がある追加の拡張機能をいくつか次に示します。

- [JavaScript デバッガー](#): Node.jsを使用してサーバー側での開発が完了したら、クライアント側を開発してテストする必要があります。この拡張機能は、DAP ベースの JavaScript デバッガーです。Node.js、Chrome、Edge、WebView2、VS Code 拡張機能などをデバッグします。

- [他のエディターからのキーマップ](#): これらの拡張機能は、別のテキスト エディター (Atom、Sublime、Vim、eMac、Notepad++ など) から移行する場合に、環境が自宅を感じるのに役立ちます。
- [設定の同期](#): GitHub を使用して、異なるインストール間で VS Code の設定を同期できます。異なるマシンで作業する場合は、環境の整合性を維持するのに役立ちます。重要なことに、この拡張機能は非推奨になりました。同等の同期ソリューションの場合は、Visual Studio Code の組み込みの **設定同期** を使用します。この同期は、チェックマークで示される **File>Preferences>Settings Sync is On** に移動することで確認できます。

Git を設定する (省略可能)

WSL で Node.js プロジェクト用に Git を設定するには、WSL ドキュメントの [Windows Subsystem for Linux で Git を使用する](#) 方法に関する記事を参照してください。

Last updated on 2025/06/11

Windows に Node.js をインストールする

[アーティクル] • 2025/01/02

このガイドは、Windows 開発環境に Node.js をインストールするのに役立ちます。

Linux 環境で Node.js を使用する場合は、「[Linux 用 Windows サブシステム \(WSL2\)](#)に Node.js をインストールする」を参照してください。

インストールする場所と、ネイティブ Windows 環境と Linux (WSL 2) 環境で Node.js を使用して開発するかどうかを決定するときは、次の点を考慮してください。

- **スキルレベルの:** Node.js を使った開発を初めて行い、すぐに起動して学習できるように実行したい場合は、Windows に Node.js をインストール。Windows に Node.js をインストールして使用すると、WSL を使用するよりも初心者向けの複雑な環境が提供されます。
- **コマンド ライン クライアント ツールの:** PowerShell を使用する場合は、Windows で Node.js を使用します。Bash を使用する場合は、Node.js on Linux (WSL 2) を使用します。
- **運用サーバー:** Windows Server に Node.js アプリを展開する予定の場合は、Windows で Node.js を使用します。Linux Server にデプロイする予定の場合は、Node.js on Linux (WSL 2) を使用します。WSL を使用すると、好みの Linux ディストリビューション (既定として Ubuntu を使用) をインストールし、開発環境 (コードを記述する場所) と運用環境 (コードがデプロイされているサーバー) の間の一貫性を確保できます。
- **パフォーマンスの速度とシステム呼び出しの互換性:** Linux と Windows のパフォーマンスでは継続的な議論と開発がありますが、Windows マシンを使用する場合の鍵は、Node.js をインストールしたのと同じファイル システムに開発プロジェクト ファイルを保持することです。Windows ファイル システムに Node.js をインストールする場合は、ファイルを Windows ドライブ (C:/など) に保持します。Node.js を Linux ディストリビューション (Ubuntu など) にインストールする場合は、使用しているディストリビューションに関連付けられている Linux ファイル システム ディレクトリにプロジェクト ファイルを保持します。(WSL 配布コマンド ラインから `explorer.exe` を入力し、Windows エクスプローラーを使用してディレクトリを参照します)。
- **Docker コンテナ:** Docker コンテナを使用して Windows でプロジェクトを開発する場合は、Windows に Docker Desktop をインストール することをお勧めします。Linux ワークスペースで Docker を使用するには、Linux と Windows の両方のビルド スクリプトを維持する必要がないように、WSL 2 を使用して Windows 用 Docker Desktop を設定する を参照してください。

nvm-windows、node.js、および npm をインストールする

Windows または WSL にインストールするかどうかを選択するだけでなく、Node.js をインストールするときに追加の選択肢があります。バージョンの変更が非常に早い場合は、バージョン マネージャーを使用することをお勧めします。作業しているさまざまなプロジェクトのニーズに基づいて、複数の Node.js バージョンを切り替える必要がある可能性があります。ノード バージョン マネージャーは、より一般的に nvm と呼ばれ、複数のバージョンの Node.js をインストールする最も一般的な方法ですが、Mac/Linux でのみ使用でき、Windows ではサポートされていません。代わりに、nvm-windows をインストールし、それを使用して Node.js とノード パッケージ マネージャー (npm) をインストールすることをお勧めします。[代替のバージョン マネージャー](#)についても検討する必要があります。次のセクションで説明します。

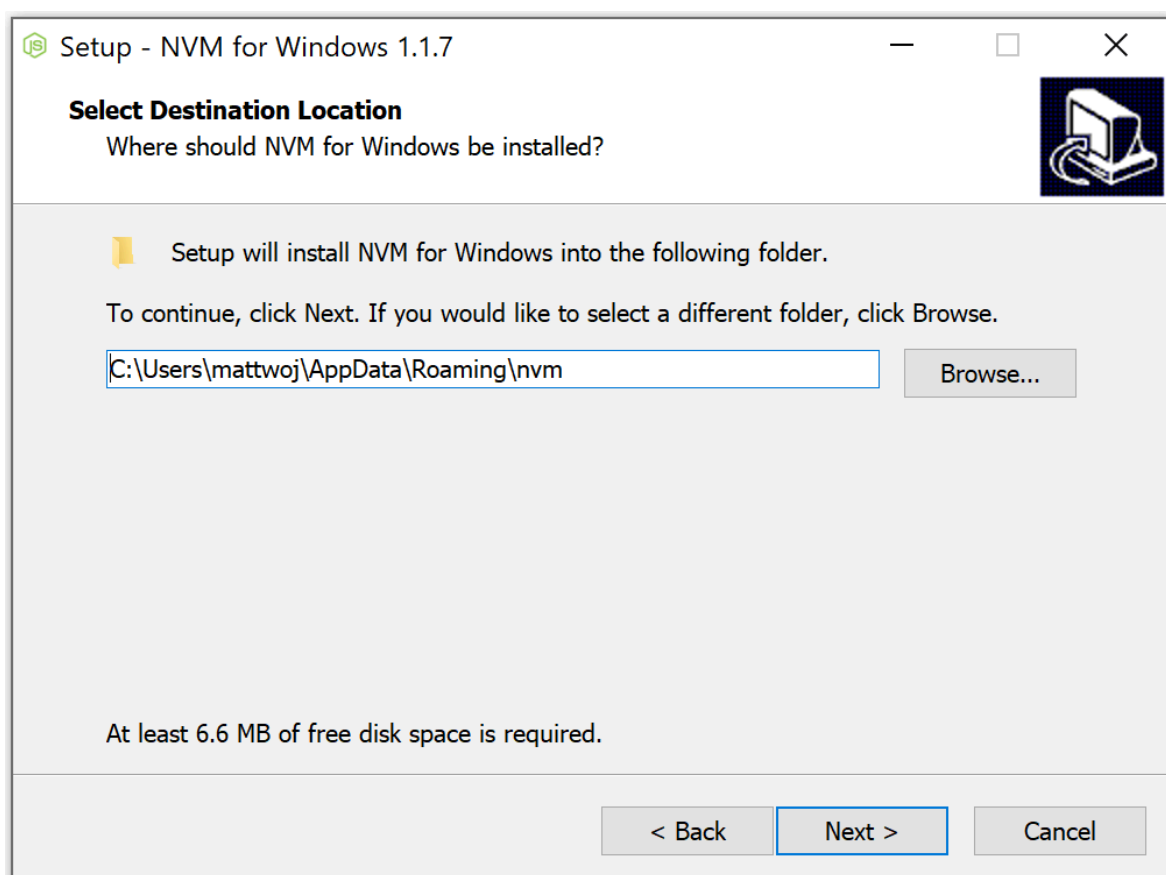
❗ 重要

バージョン マネージャーをインストールする前に、Node.js または npm の既存のインストールをオペレーティング システムから削除することをお勧めします。インストールの種類が異なると、奇妙で混乱する競合が発生する可能性があるためです。これには、残っている可能性がある既存の Node.js インストール ディレクトリ (例: "C:\Program Files\nodejs") の削除が含まれます。NVM で生成されたシンボリックリンクは、既存の (空の) インストール ディレクトリを上書きしません。以前のインストールを削除する方法については、「[Windows](#) から node.js を完全に削除する方法」を参照してください。

⚠ 警告

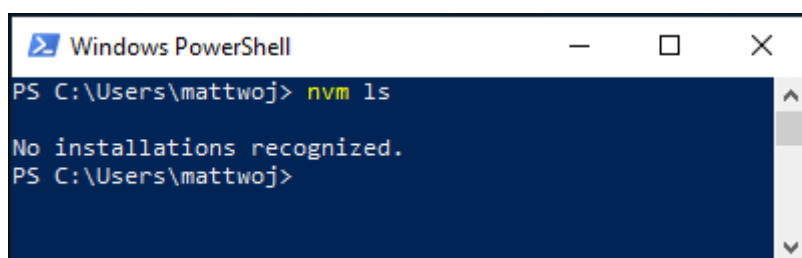
NVM は、ユーザーごとにインストールされ、シェルごとに呼び出されるように設計されています。これは、複数のビルド エージェントを持つ共有開発者ボックスやビルド サーバー用に設計されていません。NVM はシンボリックリンクを使用して動作します。共有シナリオで nvm を使用すると、そのリンクがユーザーのアプリ データ フォルダーを指しているため、問題が発生します。そのため、ユーザー x が `nvm use lts` 実行すると、リンクはボックス全体のノードをアプリ データ フォルダーにポイントします。ユーザー y がノードまたは npm を実行する場合、x のユーザー アカウントでファイルを実行するように指示され、`npm -g` の場合は x のファイルを変更しますが、既定では許可されません。したがって、nvm は 1 つの開発者ボックスにのみ規定されています。これはビルド サーバーにも当たります。2 つのビルド エージェントが同じ VM/ボックス上にある場合、それらは競合し、ビルドで奇妙な動作を引き起こす可能性があります。

1. [nvm-windows リポジトリ](#) のインストール手順に従います。 インストーラーを使用することをお勧めしますが、ニーズをより高度に理解している場合は、手動インストールを検討することをお勧めします。 インストーラーは最新バージョンの [リリース ページ](#) へ案内します。
2. 最新のリリースの **nvm-setup.zip** ファイルをダウンロードします。
3. ダウンロードしたら、zip ファイルを開き、**nvm-setup.exe** ファイルを開きます。
4. セットアップ NVM-for-Windows インストール ウィザードでは、nvm-windows と Node.js の両方がインストールされるディレクトリの選択など、セットアップ手順について説明します。



5. インストールが完了したら、PowerShell を開き (管理者特権で開くのを勧めします)、nvm-windows を使用して、現在インストールされている Node のバージョンを一覧表示してみてください (この時点では存在しない必要があります): `nvm`

`ls`

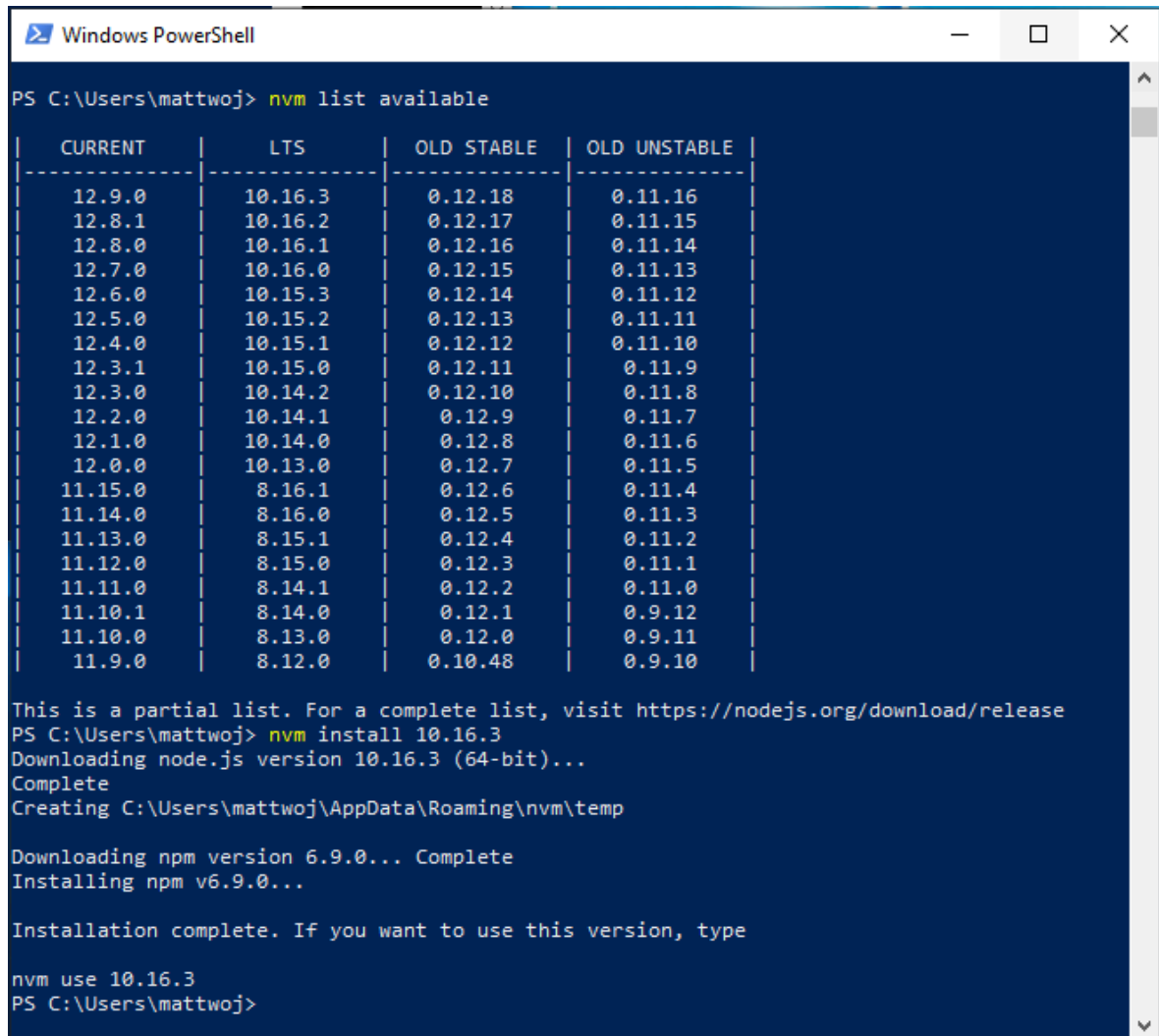


6. Node.js の現在のリリースをインストールします (最新の機能改善をテストするためですが、LTS バージョンよりも問題が発生する可能性が高くなります): `nvm`

```
install latest
```

7. Node.js の最新の安定版 LTS リリース (推奨) をインストールするには、まず `nvm list available` を使って現在の LTS バージョン番号を調べてから、そのバージョン番号を `nvm install <version>` でインストールします (<version> を該当の番号に置き換えます、例えば: `nvm install 12.14.0`)。

使用可能なバージョンの NVM 一覧を

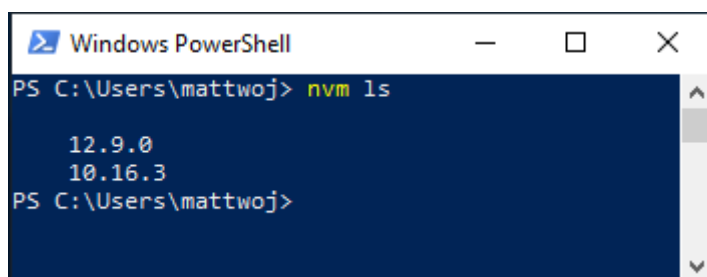


```
PS C:\Users\mattwoj> nvm list available
```

CURRENT	LTS	OLD STABLE	OLD UNSTABLE
12.9.0	10.16.3	0.12.18	0.11.16
12.8.1	10.16.2	0.12.17	0.11.15
12.8.0	10.16.1	0.12.16	0.11.14
12.7.0	10.16.0	0.12.15	0.11.13
12.6.0	10.15.3	0.12.14	0.11.12
12.5.0	10.15.2	0.12.13	0.11.11
12.4.0	10.15.1	0.12.12	0.11.10
12.3.1	10.15.0	0.12.11	0.11.9
12.3.0	10.14.2	0.12.10	0.11.8
12.2.0	10.14.1	0.12.9	0.11.7
12.1.0	10.14.0	0.12.8	0.11.6
12.0.0	10.13.0	0.12.7	0.11.5
11.15.0	8.16.1	0.12.6	0.11.4
11.14.0	8.16.0	0.12.5	0.11.3
11.13.0	8.15.1	0.12.4	0.11.2
11.12.0	8.15.0	0.12.3	0.11.1
11.11.0	8.14.1	0.12.2	0.11.0
11.10.1	8.14.0	0.12.1	0.9.12
11.10.0	8.13.0	0.12.0	0.9.11
11.9.0	8.12.0	0.10.48	0.9.10

```
This is a partial list. For a complete list, visit https://nodejs.org/download/release
PS C:\Users\mattwoj> nvm install 10.16.3
Downloading node.js version 10.16.3 (64-bit)...
Complete
Creating C:\Users\mattwoj\AppData\Roaming\nvm\temp
Downloading npm version 6.9.0... Complete
Installing npm v6.9.0...
Installation complete. If you want to use this version, type
nvm use 10.16.3
PS C:\Users\mattwoj>
```

8. インストールされている Node のバージョンを一覧表示します: `nvm ls` ...インストールした 2 つのバージョンが一覧表示されます。



```
PS C:\Users\mattwoj> nvm ls
```

12.9.0
10.16.3

```
PS C:\Users\mattwoj>
```

9. 必要な Node.js バージョン番号をインストールしたら、「`nvm use <version>`」と入力して使用するバージョンを選択します (`<version>` を番号に置き換えます。つまり、`nvm use 12.9.0`)。
10. プロジェクトに使用する Node.js のバージョンを変更するには、新しいプロジェクトディレクトリ `mkdir NodeTest` を作成し、ディレクトリ `cd NodeTest` に移動します。その後、`<version>` を希望するバージョン番号 (例: v10.16.3) に置き換えて、`nvm use <version>` を実行します。
11. インストールされている npm のバージョンを確認します。 `npm --version`、このバージョン番号は、現在のバージョンの Node.jsに関連付けられている npm バージョンに自動的に変更されます。

代替のバージョン管理ツール

NVM for Windows (nvm-windows) は現在、ノードで最も一般的なバージョン マネージャーですが、次の点を考慮する必要があります。

- nvs (Node Version Switcher) は、VS Codeと統合 機能を備えたクロスプラットフォーム の代替手段です。
- [Volta](#) は、LinkedIn チームの新しいバージョン マネージャーであり、速度とクロスプラットフォームのサポートが向上したと主張しています。

バージョン マネージャーとして Volta をインストールするには、[ファースト ステップ ガイド](#) の **Windows インストール** セクションに移動し、セットアップ手順に従って Windows インストーラーをダウンロードして実行します。

❗ 重要

Volta インストールする前に、Windows コンピューターで Developer Mode が有効になっていることを確認する必要があります。

Volta を使用して Windows に複数のバージョンの Node.js をインストールする方法の詳細については、[Volta Docs](#) を参照してください。

Visual Studio Code をインストールする

Windows で Node.js を使用して開発するには、Visual Studio Code をインストールすることを [お勧めします](#)。ヘルプについては、Visual Studio Codeのチュートリアル Node.js 参照してください。

代替コード エディター

Visual Studio Code 以外のコード エディターまたは IDE を使用する場合は、Node.js 開発環境に適したオプションも次のとおりです。

- [WebStorm](#)
- [Sublime Text](#)
- [Atom](#)
- [Brackets](#)
- [Notepad++](#)

Git をインストールする

他のユーザーと共同作業を行う場合、またはプロジェクトをオープンソース サイト (GitHub など) でホストする場合、VS Code では Gitを使用した バージョン管理がサポートされます。VS Code の [ソース管理] タブでは、すべての変更が追跡され、UI に一般的な Git コマンド (追加、コミット、プッシュ、プル) が組み込まれています。まず、ソース管理パネルに電源を供給するために Git をインストールする必要があります。

1. git-scm Web サイトから Git for Windows ダウンロードしてインストールします。
2. インストール ウィザードが含まれており、Git インストールの設定に関する一連の質問が表示されます。何かを変更する特定の理由がない限り、すべての既定の設定を使用することをお勧めします。
3. Git を使用したことがない場合は、[GitHub Guides](#) を使用して作業を開始できます。
4. [.gitignore ファイル](#) を Node プロジェクトに追加することをお勧めします。Node.js用の GitHub の既定の gitignore テンプレート 次を示します。

Windows Server での Node.js

Windows サーバーで Node.js アプリをホストする必要がある (ややまれな) 状況にある場合、最も一般的なシナリオは、リバース プロキシを使用して と思われます。これを行うには、2 つの方法があります: [iisnode を使う](#) か、[直接使用](#) します。Microsoft では、これらのリソースを保持していないので、[Linux サーバーを使用して Node.js アプリをホストする](#) ことをお勧めします。

Windows で React を始める

React は、ユーザー インターフェイスを構築するためのオープンソースの JavaScript ライブラリです。Meta によって作成された React を使用すると、**コンポーネント**と呼ばれる再利用可能な小さな部分から複雑な UI を作成できます。ビュー レイヤーのみに焦点を当てて、既存のプロジェクトに簡単に統合し、簡単に学習できるようにします。

React を使用する理由

- **コンポーネントベース**: カプセル化されたコンポーネントは独自の状態を管理するため、複雑な UI を作成してロジックを絡めずに作成します。
- **宣言型**: UI の外観を記述し、React が DOM の更新を効率的に処理します。
- **大規模なエコシステム**: アクティブなコミュニティに支えられたコミュニティ パッケージ、ツール、学習リソースの膨大なライブラリ。

Windows に React をインストールする

好みの環境を選択します。

- [Windows に React をインストールする](#) - Node.js で Windows 上で直接 Vite を使用します。
- [WSL への React のインストール](#) - Linux と互換性のある開発環境 (Linux サーバーへのデプロイに推奨) には、Linux 用 Windows サブシステム上の Vite を使用します。

React を使用してネイティブ Windows またはモバイル アプリを構築する

(Web アプリではなく) React を使用してネイティブ **のユニバーサル Windows プラットフォーム (UWP)** または **WPF デスクトップ アプリ**を構築するには、[React Native for Desktop](#) (Windows、Android、iOS、macOS を対象とする Microsoft オープンソース プロジェクト) を参照してください。

クロスプラットフォーム モバイル アプリについては、[reactnative.dev](#) を参照 [してください](#)。

チュートリアルと学習リソース

React コミュニティが維持しているものを複製するのではなく、次のことをお勧めします。

- [react.dev](#) — 公式 React のドキュメントと対話型のチュートリアル。

- [Microsoft Learn の React ラーニング パス](#) - 初心者向けの構造化されたオンライン モジュール。
 - [Next.js ドキュメント](#)  - 運用 Web アプリ用の React フレームワーク。
 - [Gatsby ドキュメント](#)  — React ベースの静的サイト ジェネレーター。
-

Last updated on 2026/03/24

Linux 用 Windows サブシステムに React をインストールする

このガイドでは、[Vite](#) フロントエンド ツールを使用して WSL (Linux 用 Windows サブシステム) に React 開発環境を設定する手順について説明します。

Linux サーバーへのデプロイ、Docker コンテナの使用、Bash ベースのツールの使用を計画している場合は、WSL をお勧めします。React を初めて使用し、すぐに開始したい場合は、代わりに [Windows に React を直接インストール](#) することを検討してください。

React の背景と、Web アプリ、モバイル アプリ (React Native)、ネイティブ デスクトップ アプリ (React Native for Desktop) など、さまざまなシナリオについては、React の [概要](#) を参照してください。

前提条件

- [WSL 2](#): Linux ディストリビューション (Ubuntu 推奨) を使用して WSL をインストールし、WSL 2 モードで実行されていることを確認します: `wsl -l -v`。Windows 10 バージョン 2004 以降または Windows 11 が必要です。
- [WSL 2 の Node.js](#): nvm を使用して WSL ディストリビューション内に Node.js をインストールします。WSL 内で Windows にインストールされているバージョンの Node.js を使用しないでください。

❗ 重要

マウントされた Windows ドライブ (`~/projects`) ではなく、WSL ファイルシステム (`/mnt/c/` など) 内にプロジェクト ファイルを格納します。ファイルシステムの境界を越えて作業すると、インストールとビルドの時間が大幅に遅くなります。

React アプリを作成する

1. WSL ターミナル (Ubuntu など) を開きます。
2. 新しいプロジェクト フォルダーを作成し、次のように入力します。

Bash

```
mkdir ~/ReactProjects
cd ~/ReactProjects
```

3. Vite を使用して新しい React アプリを作成します。

Bash

```
npm create vite@latest my-react-app -- --template react
```

Vite は、`my-react-app` フォルダー内の新しい React プロジェクトをスキャフォールディングします。

4. 新しいアプリ フォルダーに移動し、依存関係をインストールします。

Bash

```
cd my-react-app  
npm install
```

5. ローカル開発サーバーを起動します。

Bash

```
npm run dev
```

アプリは `http://localhost:5173` で利用できるようになります。 **Ctrl + C キー**を押してサーバーを停止します。

6. デプロイする準備ができたなら、運用バンドルをビルドします。

Bash

```
npm run build
```

出力は、`dist` フォルダーに配置されます。ホスティング オプションについては、「[静的サイトの展開](#)」を参照してください。

ⓘ Note

Vite は、シングルページ アプリ (SPA) に最適です。サーバー側のレンダリングまたは Node.js バックエンドが必要な場合は、[代わりにNext.js](#) することを検討してください。静的サイトの生成については、[Gatsby](#) を参照してください。

その他のリソース

- [React のドキュメント](#)
 - [Vite ドキュメント](#)
 - [Visual Studio Code での React の使用](#)
 - [Microsoft Learn の React ラーニング パス](#)
 - [WSL での NodeJS のインストール](#)
 - [デスクトップ用 React Native \(UWP/WPF\)](#)
-

Last updated on 2026/03/24

React を Windows に直接インストールする

このガイドでは、[Vite](#) フロントエンド ツールを使用して、Windows で React 開発環境を直接設定する手順について説明します。

React を初めて使用する場合や、Linux 環境を必要としないプロジェクトをビルドする場合は、次の手順をお勧めします。Bash ツールを広範囲に使用する場合、または Linux サーバーにデプロイする場合は、代わりに [REACT を WSL にインストール](#) することを検討してください。

React の背景と、Web アプリ、モバイル アプリ (React Native)、ネイティブ デスクトップ アプリ (React Native for Desktop) など、さまざまなシナリオについては、React の [概要](#) を参照してください。

前提条件

- [Node.js](#): Vite と npm を実行するために必要です。バージョン管理を容易にするための nvm-windows を使用してインストールします。

React アプリを作成する

1. ターミナル (Windows コマンド プロンプトまたは PowerShell) を開きます。
2. 新しいプロジェクト フォルダーを作成し、次のように入力します。

PowerShell

```
mkdir ReactProjects  
cd ReactProjects
```

3. Vite を使用して新しい React アプリを作成します。

PowerShell

```
npm create vite@latest my-react-app -- --template react
```

Vite は、`my-react-app` フォルダー内の新しい React プロジェクトをスキャフォールディングします。

4. 新しいアプリ フォルダーに移動し、依存関係をインストールします。

PowerShell

```
cd my-react-app
npm install
```

5. ローカル開発サーバーを起動します。

PowerShell

```
npm run dev
```

アプリは `http://localhost:5173` で利用できるようになります。 **Ctrl + C キー**を押してサーバーを停止します。

6. デプロイする準備ができれば、運用バンドルをビルドします。

PowerShell

```
npm run build
```

出力は、 `dist` フォルダーに配置されます。ホスティング オプションについては、「[静的サイトの展開](#)」を参照してください。

❗ Note

Vite は、シングルページ アプリ (SPA) に最適です。サーバー側のレンダリングまたは Node.js バックエンドが必要な場合は、[代わりにNext.js](#) することを検討してください。静的サイトの生成については、[Gatsby](#) を参照してください。

その他のリソース

- [React のドキュメント](#)
- [Vite ドキュメント](#)
- [Visual Studio Code での React の使用](#)
- [Microsoft Learn の React ラーニング パス](#)
- [Windows での NodeJS のインストール](#)
- [デスクトップ用 React Native \(UWP/WPF\)](#)

デスクトップ用 React Native

[React Native for Desktop](#) は、React と JavaScript を使用してネイティブ **のユニバーサル Windows プラットフォーム (UWP)** および WPF デスクトップ アプリケーションを構築できる Microsoft オープンソース プロジェクトです。ブラウザで実行される React Web アプリとは異なり、React Native for Desktop ではネイティブ Windows UI コントロールがレンダリングされます。同じ JavaScript コンポーネントで Android、iOS、macOS を対象にすることもできます。

概要

インストール手順、システム要件、チュートリアルについては、[React Native for Windows のドキュメント](#) を参照してください。

その他のリソース

- [React Native for Windows GitHub リポジトリ](#)
- [サンプル アプリ](#)
- [Fluent UI React ネイティブ コンポーネント](#)
- [React Native の概要](#)

Last updated on 2026/03/24

WindowsでVueを始める

Vue は、ユーザー インターフェイスを構築するためのオープンソースの JavaScript フレームワークです。Evan You によって作成され、2014 年にリリースされた Vue は、段階的に導入できるように設計されています。これは、既存のページを強化するための軽量ライブラリとして、またはシングルページ アプリケーションを構築するためのフル機能のフレームワークとして使用できます。

Vue を使用する理由

- **アプローチ可能:** HTML、CSS、JavaScript がわかっている場合は、Vue をすばやく選択できます。
- **汎用性:** 複雑な SPA まで、ページ上の単純なウィジェットに適しています。
- **パフォーマンス:** バンドル サイズが小さく、効率的な仮想 DOM。
- **コンポーネントベース:** クリーンなテンプレート構文を使用した、コンポーザブルで再利用可能なコンポーネント。

Windows に Vue をインストールする

好みの環境を選択します。

- **Windows に Vue をインストール** する - Windows で直接 `npm create vue@latest` (Vite ベース) を使用します。
- **WSL に Vue をインストール** する - Linux 用 Windows サブシステムで同じツールを使用します (Linux サーバーにデプロイする場合、または Linux ツールチェーンパリティが必要な場合に推奨)。

サーバーでレンダリングされた Vue アプリをビルドする

サーバー側のレンダリングと高度なルーティングについては、[公式の](#) [Vue](#) メタフレームワークである Nuxt.js を参照してください。

チュートリアルと学習リソース

- vuejs.org/guide/introduction [↗](#) — 公式の Vue ドキュメントとクイック スタート。
- vuejs.org/tutorial [↗](#) — Vue の基礎をカバーする対話型のブラウザー内チュートリアルです。
- Vue.js- Microsoft Learn ラーニング パス [を使用して、最初の手順を実行します](#)。

- [VS CodeVue.js チュートリアル](#)  - Visual Studio Code で単純な Vue アプリを構築します。
-

Last updated on 2026/03/24

Linux 用 Windows サブシステムに Vue.js をインストールする

Windows Subsystem for Linux (WSL) で Vue.js 開発環境を設定するためのガイド。背景については、Vue.js の [概要](#) を参照してください。

Vue は、Windows または WSL [に直接](#) インストールできます。Node.js バックエンドと対話する場合、Linux サーバーにデプロイする場合、または Bash コマンドを使用するチュートリアルに従う場合は、WSL をお勧めします。

[前提条件]

- WSL 2 モードで実行されている Linux ディストリビューション (Ubuntu など) を含む、Windows [Subsystem for Linux \(WSL\)](#) をインストールします。 `wsl -l -v` で確認する:
- [WSL 2](#) に Node.js をインストールする: これには、バージョン マネージャー、パッケージ マネージャー、Visual Studio Code、リモート開発拡張機能が含まれます。

❗ 重要

マウントされた Windows ドライブ (`~/projects`) ではなく、WSL ファイルシステム (`/mnt/c/` など) 内にプロジェクト ファイルを格納します。ファイルシステムの境界を越えて作業すると、インストールとビルドの時間が大幅に遅くなります。

Vue プロジェクトを作成する

新しい Vue 3 プロジェクトを開始するには、`npm create vue@latest` することをお勧めします。この方法では、公式の Vite ベースのスキャフォールディング ツールである [create-vue](#)  を使用します。

- WSL ターミナル (Ubuntu など) を開きます。
- プロジェクト ディレクトリに移動します。

Bash

```
mkdir -p ~/projects
cd ~/projects
```

- 新しい Vue プロジェクトを作成します。

Bash

```
npm create vue@latest
```

インストーラーによって、プロジェクトに名前を付け、オプションの機能 (TypeScript、JSX サポート、Vue ルーター、Pinia 状態管理、Vitest、ESLint) を選択するように求められます。

4. プロジェクト フォルダーに移動し、依存関係をインストールして、開発サーバーを起動します。

Bash

```
cd <your-project-name>  
npm install  
npm run dev
```

アプリは `http://localhost:5173` で利用できるようになります。

その他のリソース

- [Vue ドキュメント](#)
- [Vue.js の概要](#)
- [Windows に Vue.js をインストールする](#)
- [Vue.js ラーニング パスを使用して最初の手順を実行する](#)
- [VS Code を使用した Vue チュートリアル](#)

Last updated on 2026/03/24

Vue.js を Windows に直接インストールする

Windows で Vue.js 開発環境を設定するためのガイド。背景については、Vue.js の [概要](#) を参照してください。

Vue は、Windows または Windows Subsystem for Linux (WSL) に直接インストールできます。Node.js バックエンドと対話する場合、Linux サーバーにデプロイする場合、または Bash コマンドを使用するチュートリアルに従う場合は、代わりに [WSL に Vue をインストール](#) することを検討してください。

[前提条件]

- [Windows に Node.js をインストール](#)する: これには、バージョン マネージャー、パッケージ マネージャー、Visual Studio Code が含まれます。

Vue プロジェクトを作成する

新しい Vue 3 プロジェクトを開始するには、`npm create vue@latest` することをお勧めします。この方法では、公式の Vite ベースのスキャフォールディング ツールである [create-vue](#) を使用します。

1. PowerShell または Windows コマンド プロンプトを開きます。
2. プロジェクト ディレクトリに移動します。

PowerShell

```
cd C:\Users\YourName\Projects
```

3. 新しい Vue プロジェクトを作成します。

PowerShell

```
npm create vue@latest
```

インストーラーによって、プロジェクトに名前を付け、オプションの機能 (TypeScript、JSX サポート、Vue ルーター、Pinia 状態管理、Vitest、ESLint) を選択するように求められます。

4. プロジェクト フォルダーに移動し、依存関係をインストールして、開発サーバーを起動します。

PowerShell

```
cd <your-project-name>  
npm install  
npm run dev
```

アプリは `http://localhost:5173` で利用できるようになります。

その他のリソース

- [Vue ドキュメント](#)
- [Vue.js の概要](#)
- [WSL に Vue.js をインストールする](#)
- [Vue.js ラーニング パスを使用して最初の手順を実行する](#)
- [VS Code を使用した Vue チュートリアル](#)

Last updated on 2026/03/24

WindowsでPython開発環境を設定する

Python環境をWindowsで数分で準備します。winget を使用してコマンド ラインからインストールするか、手動で設定します。 以下のものがが必要です。

- **Python** – 現在の安定したPython ランタイム (3.14 以降)
- **Visual Studio Code** – 軽量で強力なコード エディター
- Visual Studio Code の Python 拡張機能 – Python言語のサポート、デバッグ、リンティングなどを追加します

WinGet

1. Windows ターミナルで PowerShell を開き、Pythonをインストールします。

PowerShell

```
winget install Python.Python.3.14
```

2. Visual Studio Codeをインストールします。

PowerShell

```
winget install Microsoft.VisualStudioCode
```

3. PowerShell を閉じて再度開き、Pythonがインストールされていることを確認します。

PowerShell

```
python --version
```

4. VS Code を開き、VS Code Marketplace から [Python 拡張機能](#) をインストールします。

これで、Python開発の準備ができました。

Python

よく寄せられる質問

▼ pip インストールでのパッケージのインストールに関する問題

インストールが失敗する理由は多数あります。多くの場合、適切なソリューションはパッケージ開発者に連絡することです。

問題の一般的な原因は、変更するアクセス許可がない場所にインストールしようとしている場合です。たとえば、既定のインストール場所には管理者特権が必要な場合がありますが、既定ではPythonには管理者特権は必要ありません。最適な解決策は、[仮想環境](#)を作成し、そこにインストールすることです。

一部のパッケージには、C または C++ コンパイラをインストールする必要があるネイティブコードが含まれています。一般に、パッケージ開発者はプリコンパイル済みバージョンを発行する必要があるますが、多くの場合は発行しません。これらのパッケージの一部は、[build Tools for Visual Studio](#) をインストールして C++ オプションを選択した場合に機能することがありますが、ほとんどの場合、パッケージ開発者に連絡する必要があります。

[StackOverflow](#) に関する説明に従う

▼ WSL での pip のインストールに関する問題

Linux 用 Windows サブシステム (WSL または WSL2) に pip を含むパッケージ (Flask など) をインストールすると、`python3 -m pip install flask` など、次のようなエラーが発生する可能性があります。

Bash

```
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None))
after connection broken by
'NewConnectionError(<urllib3.connection.VerifiedHTTPSConnection
object at 0x7f655471da30>: Failed to establish a new connection: [Errno -3]
Temporary failure in name resolution)': /simple/flask/
```

アフターマーケットファイアウォールを実行していない限り、おそらく解決策は pip を再インストールすることです。

Bash

```
sudo apt -y purge python3-pip
sudo python3 -m pip uninstall pip
sudo apt -y install python3-pip --fix-missing
```

▼ py.exeとは

さまざまな種類のPython プロジェクトに取り組んでいるため、コンピューターに複数のバージョンのPythonがインストールされる場合があります。これらはすべて `python` コマンドを使用するため、使用しているPythonのバージョンが明らかでない場合があります。標準として、`python3` コマンド (または `python3.7` を使用して特定のバージョンを選択する) を使用することをお勧めします。

[py.exeランチャー](#) は、インストールしたPythonの最新バージョンを自動的に選択します。`py -3.7`などのコマンドを使用して特定のバージョンを選択したり、`py --list` を使用して使用できるバージョンを確認したりすることもできます。HOWEVER、py.exe ランチャーは、[python.org](#) からインストールされたバージョンのPythonを使用している場合にのみ機能します。Microsoft StoreからPythonをインストールすると、`py` コマンドは**含まれません**。Linux、macOS、WSL、Microsoft Store バージョンのPythonの場合は、`python3` (または `python3.7`) コマンドを使用する必要があります。

▼ なぜ python.exe を実行すると Microsoft Store が開かれるのですか？

新しいユーザーがPythonの適切なインストールを見つけられるように、Microsoft Storeで公開されているコミュニティのパッケージの最新バージョンに直接移動するWindowsへのショートカットが追加されました。このパッケージは管理者権限なしで簡単にインストールでき、既定の `python` コマンドと `python3` コマンドを実際のコマンドに置き換えます。

コマンドライン引数を指定してショートカット実行可能ファイルを実行すると、Pythonがインストールされていないことを示すエラー コードが返されます。これは、意図されていない可能性がある場合に、バッチ ファイルとスクリプトがストア アプリを開くのを防ぐためです。

python.org "add to PATH" オプションを選択すると、新しい コマンドがショートカットよりも優先されます。他のインストーラーは、組み込みのショートカットよりも優先順位の `python` を追加する場合がありますことに注意してください。

スタートから [アプリの実行エイリアスの管理] を開き、"アプリ インストーラー" Pythonエントリを見つけて、"オフ" に切り替えることで、Pythonをインストールせずにショートカットを無効にすることができます。

▼ Pythonでファイルパスがコピー＆ペーストで動作しないのはなぜですか？

Python文字列では、特殊文字に "エスケープ" を使用します。たとえば、文字列に改行文字を挿入するには、「`\n`」と入力します。Windowsのファイルパスでは円記号が使用されるため、一部の部分が特殊文字に変換されている可能性があります。

パスを文字列としてPythonに貼り付けるには、`r` プレフィックスを追加します。これは、それが `raw` 文字列であり、" を除いてエスケープ文字が使用されていないことを示します (パス内の最後の円記号を削除する必要がある場合があります)。パスは次のようになります。 `r"C:\Users\MyName\Documents\Document.txt"`

Pythonでパスを操作する場合は、標準の `pathlib` モジュールを使用することをお勧めします。これにより、スラッシュや円記号を使用するかどうかにかかわらずパス操作を一貫して実行できる豊富な `Path` オブジェクトに文字列を変換できるため、異なるオペレーティングシステム間でコードの動作が向上します。

▼ PYTHONPATH とは

PYTHONPATH 環境変数は、モジュールをインポートできるディレクトリの一覧を指定するために、Pythonによって使用されます。実行中に、`sys.path` 変数を調べて、何かをインポートするときに検索されるディレクトリを確認できます。

コマンド プロンプトからこの変数を設定するには、`set PYTHONPATH=list;of;paths` を使用します。

PowerShell からこの変数を設定するには、Pythonを起動する直前に `$env:PYTHONPATH='list;of;paths'` を使用します。

Environment 変数の設定を使用してこの変数をグローバルに設定することは**推奨されません**。これは、意図したものではなく、任意のバージョンのPythonで使用される可能性があるためです。

▼ パッケージ化とデプロイに関するヘルプはどこで入手できますか？

[Docker](#) : [VSCode 拡張機能](#)  を使用すると、Dockerfile テンプレートとdocker-compose.yml テンプレートを使用して迅速にパッケージ化およびデプロイできます (プロジェクトに適した Docker ファイルを生成します)。

[Azure Kubernetes Service \(AKS\)](#) を使用すると、必要に応じてリソースをスケーリングしながら、コンテナ化されたアプリケーションをデプロイおよび管理できます。

▼ 異なるマシン間で作業する必要がある場合はどうなりますか？

VS Code には [Settings Sync](#) が組み込まれており、GitHubまたはMicrosoft アカウントを使用して、コンピューター間で設定、キーバインド、拡張機能などを共有できます。拡張機能は必要ありません。

▼ PyCharm、Atom、Sublime Text、Emacs、Vim の使用に慣れたりするとどうなるでしょうか。

VSCode 拡張機能 [キーマップ](#) は、作業環境を普段の環境のように快適にするのに役立ちます。

▼ Mac のショートカット キーをWindowsショートカット キーにマップする方法

一部のキーボード ボタンとシステム ショートカットは、WindowsコンピューターとMacintosh で若干異なります。この[Mac to Windows移行ガイド](#)では、基本について説明します。

▼ VS Code でのPythonの使用の詳細については、どこで確認できますか？

- [VS Code でのPythonの編集](#): VS Code のオートコンプリートと IntelliSense のPython のサポート (動作のカスタマイズ方法など) を利用する方法について説明します。または単にオフにします。
- [linting Python](#): Linting は、潜在的なエラーのコードを分析するプログラムを実行するプロセスです。VS Code がPythonに提供するさまざまな形式のリンティング サポートと、それを設定する方法について説明します。
- [debugging Python](#): デバッグは、コンピューター プログラムからエラーを識別して削除するプロセスです。この記事では、VS Code を使用してPythonのデバッグを初期化および構成する方法、ブレークポイントの設定と検証、ローカル スクリプトのアタッチ、さまざまなアプリの種類またはリモート コンピューターでのデバッグの実行、基本的なトラブルシューティングについて説明します。
- [Unit テスト Python](#): 単体テストの意味、チュートリアル例、テスト フレームワークの有効化、テストの作成と実行、テストのデバッグ、およびテスト構成設定の背景について説明します。

Windows で Java の使用を開始する

このガイドでは、Windows での Java のセットアップと Mac または Linux の設定の違いについて説明し、環境が正しく構成された動作する JDK のインストールについて説明します。

JDK を選択する

Windows では、いくつかの JDK ディストリビューションを使用できます。Microsoft は、OpenJDK に基づいて独自のビルドを発行します。

- [Microsoft Build of OpenJDK](#) — 無料のオープンソース。Microsoft によってサポートされ、Windows x64 および ARM64 で利用できます
- [Eclipse Temurin \(Adoptium\)](#) - 広く使用されているオープンソースディストリビューション
- [Oracle JDK](#) — 商用使用のためのライセンスが必要です

ほとんどの開発者にとって、Microsoft Build of OpenJDK または Temurin は適切な選択肢です。

winget を使用してインストールする

PowerShell またはコマンドプロンプトターミナルを開き、次のコマンドを実行します。

PowerShell

```
winget install Microsoft.OpenJDK.21
```

代わりに Temurin をインストールするには:

PowerShell

```
winget install EclipseAdoptium.Temurin.21.JDK
```

インストールを確認します。

PowerShell

```
java -version
```

JAVA_HOMEの設定

macOS (`/usr/libexec/java_home` でこれ进行处理する) とは異なり、Windows では、 `JAVA_HOME` を手動で設定する必要があります。Maven、Gradle、Android Studio など、多くのツールが必要です。

1. **[スタート]** を開き、**[環境変数]** を検索し、**[システム環境変数の編集]** を選択します。
2. **[環境変数]** をクリックします。
3. **[システム変数]** の **[新規]** をクリックして設定します。
 - 変数名: `JAVA_HOME`
 - 変数値: JDK へのパス (例: `C:\Program Files\Microsoft\jdk-21.0.x.x-hotspot`)
4. [システム変数] で **[パス]** 変数を探し、**[編集]** をクリックして、`%JAVA_HOME%\bin` を追加します。
5. [OK] をクリックしてすべてのダイアログを閉じ、新しいターミナルを開き、次のことを確認します。

PowerShell

```
echo $env:JAVA_HOME
java -version
javac -version
```

💡 ヒント

複数の JDK がインストールされている場合は、`Path` の最初に示されているものが優先されます。[SDKMAN](#) (WSL 経由) や [jEnv](#) などのツールは、複数のバージョンの管理に役立ちます。

WSL またはネイティブ Windows?

ほとんどの Java 開発 (Web バックエンド、Android、エンタープライズ アプリ) では、ネイティブ Windows が適切に機能します。ビルド ツールまたはデプロイ ターゲットが Linux 固有の場合、または他のすべてのユーザーが Linux/Mac 上にあり、環境パリティが必要なチームで作業している場合は、WSL を使用します。

推奨されるエディター

- [Visual Studio Code](#) と [Java 用拡張機能パック](#) - 軽量で、ほとんどのプロジェクトに適しています
- [IntelliJ IDEA](#) - フル機能の IDE、Community エディションは無料です

次のステップ

Java on Azure と Windows のドキュメント

- [Java for Beginners \(ビデオ シリーズ\)](#)
- [Learn Java \(java.com\)](#) 
- [5分でわかるMaven](#) 
- [Gradle クイックスタート](#) 
- [Visual Studio Code の Java](#) 

Last updated on 2026/03/28

Microsoft C++、C、およびアセンブラのドキュメント

C++、C、およびアセンブリ言語を使用して、お使いのプラットフォームやデバイス用のアプリケーション、サービス、ツールを開発する方法について説明します。



ダウンロード
**Visual Studio のインストール
と、C++ のワークロードの選択**



概要
Visual Studio での C++ へようこそ



作業の開始
Visual Studio と C++ の作業を開始する



新着情報
Visual Studio での C++ の新機能

C++ および C の作業を開始する

Visual Studio IDE の使用方法を学習する

-  Visual Studio のガイド ツアーを開始する
-  リポジトリからコードを開く
-  コードを作成して編集する
-  コードをビルドする
-  コードをデバッグする
-  コードをテストする

Visual Studio で C++ および C アプリを作成する

-  コンソール電卓アプリを作成する
-  Win32 を使用して Windows デスクトップ アプリを作成する
-  MFC を使用して Windows デスクトップ アプリを作成する
-  Windows DLL を作成する
-  スタティック ライブラリを作成する
-  .NET コンポーネントを作成する
-  ユニバーサル Windows プラットフォームのアプリを作成する

コマンド ライン ツールを使用する

-  [C++ コードのコンパイル](#)
-  [C コードのコンパイル](#)
-  [C++/CX のコンパイル](#)
-  [C++/CLI のコンパイル](#)

Visual Studio Code で C++ および C を使用する

-  [Visual Studio Code の作業を開始する](#)
-  [Microsoft C/C++ 拡張機能をインストールする](#)
-  [Windows での Microsoft C/C++ の使用](#)
-  [Linux 用 Windows サブシステムでの C++ の使用](#)
-  [Linux での C++ の使用](#)
-  [macOS で C++ を使用する](#)

言語とフレームワーク



C++



C



Microsoft アセンブラー



Windows ランタイム用 C++/CX



.NET 用 C++/CLI



アクティブ テンプレート ライブラリ (ATL)



Microsoft Foundation クラス (MFC)



Windows ランタイム用 C++/WinRT

C++ および C のワークロード、機能、およびライブラリ

Visual Studio ツールを使用して、選択したプラットフォーム向けに開発を行います。

ワークロード

特徴

ライブラリ

ユニバーサル Windows プラットフォーム開発

Windows デスクトップ開発

Linux 開発

埋め込み開発

モバイル開発

ゲーム開発

信頼性が高くセキュリティで保護されたプログラムを構築する

コードの編集とリファクタリング

コードプロジェクトのビルド

コードをデバッグする

コードの分析

アプリ パフォーマンスのプロファイル

コードの移植およびアップグレード

コードのバグをサニタイズする

C++ 標準ライブラリ リファレンス

C ランタイム ライブラリ リファレンス

MFC および ATL Windows デスクトップ ライブラリ

並列プログラミング ライブラリ

クラウドおよびネットワーク ライブラリ

Azure SDK for C++

ユニバーサル Windows プラットフォーム ライブラリ

vcpkg パッケージ マネージャ

Microsoft Learn Q&A - [C++ チーム ブログ](#) - [Twitter](#) - [Developer Community](#) - [Stack Overflow](#) - [問題を報告する方法](#) - [機能の提案](#) - C++ ドキュメントへの投稿する: 「[共同作成者ガイド](#)」をご覧ください。

C# 言語のドキュメント

C# ガイドには、C# および .NET プラットフォームの使用を開始するのに役立つ記事、チュートリアル、コード サンプルが含まれています。経験豊富な開発者は、「新機能」セクションで新機能について学習できます。経験豊富な開発者は、リファレンスと言語仕様から言語の動作の詳細について学習できます。



作業の開始
C# のツアー



概念
基礎



新着情報
新着情報



VIDEO
**C# ビデオ シリ
ーズを学習する**



トレーニング
**C# ラーニング
パス**



トレーニング
**基礎 C# 認定資
格 -
freeCodeCamp**

プログラミングの学習

C# 言語の主な概念と機能



概要

[C# の紹介ツアー](#)

[C を使用してビルドできる内容](#)

[初心者向け C# チュートリアル](#)

[Java 開発者向けの C#](#)

[初心者向け C# ビデオ シリーズ](#)

[基本的な C# 認定資格](#)



基礎

[型システム](#)

[オブジェクト指向プログラミング](#)

[関数の手法](#)

[例外](#)

[コーディング スタイル](#)

[C# 言語の戦略](#)



言語の概念

[プログラミングの概念](#)

[LINQ](#)

[非同期プログラミング](#)



高度な言語の概念

[リフレクションと属性](#)

[インターフェイスの実装](#)

[式ツリー](#)

[ネイティブ相互運用性](#)

[パフォーマンス エンジニアリング](#)

[.NET Compiler Platform \(Roslyn\) SDK](#)

リファレンス

C# 言語リファレンス資料と C# 言語仕様をお読みください。C# リファレンスでは、C# 言語に関する有益な参照情報を提供します。C# 言語仕様は、C# 言語の基準となる参照資料です。これは、C# 言語の構文およびセマンティクスを示す公式ソースです。機能仕様は、まだ標準で組み込まれていない機能に関するドキュメントです。

言語リファレンス

C# 言語リファレンスでは、C# 言語に関する有益な情報を提供します。

- [🔗 C# 言語リファレンス](#)
- [🔗 組み込み型](#)
- [🔗 キーワード](#)
- [🔗 演算子](#)
- [🔗 声明文](#)
- [🔗 特殊文字](#)
- [🔗 ドキュメントコメント](#)

言語リファレンス

[アンセーフコードとコンパイラ オプション](#)

- [🔗 特別な属性](#)
- [🔗 アンセーフコード](#)
- [🔗 プリプロセッサディレクティブ](#)
- [🔗 コンパイラ オプション](#)

言語仕様

[C# 言語の公式仕様](#)

- [🔗 標準と仕様](#)
- [🔗 仕様プロセス](#)
- [🔗 C# ドラフト標準](#)
- [🔗 機能仕様](#)

Visual Studio ファミリを使用して C# アプリをビルドする

Visual Studio または Visual Studio Code を選択して、C# アプリケーションをビルドします。



Visual Studio



Visual Studio Code [🔗](#)

アプリケーションを作成する

Web、モバイル、デスクトップ、ゲーム、IoT などを選択できます。

機械学習と AI

[ML.NET を使用してカスタム AI ソリューションを構築する](#)

[Azure Cognitive Services](#)

[Azure Machine Learning](#)

[機械学習での F#](#)

クラウド

[.NET 開発者向けの Azure](#)

[Aspire \[🔗\]\(#\)](#)

[を使用してクラウドネイティブ アプリを構築する Aspire \[🔗\]\(#\)](#)

[オンプレミスの .NET Web アプリまたはサービスを移行する](#)

[.NET 開発者向けの Azure サービス](#)

[.NETのAzure SDK](#)

[F# を使用して Azure リソースをデプロイする](#)

Web

[ASP.NET Core チュートリアル](#)

[ASP.NET Core とは何ですか?](#)

[Visual Studio での ASP.NET Core](#)

[Windows コンテナの ASP.NET MVC アプリ](#)

[Blazor: .NET を使用した対話型のクライアント側 Web UI](#)

[Web 開発での F#](#)

マイクロサービス

[.NET 開発者向け Dapr](#)

[クラウド ネイティブ .NET アプリ](#)

[Azure を使用したサーバーレス アプリ](#)

[コンテナ化された .NET アプリのアーキテクチャ](#)

[Worker サービスを Azure にデプロイする](#)

モバイル、デスクトップ、IoT

[Windows Presentation Foundation](#)

[Windows フォーム](#)

[.NET Multi-Platform App UI \(.NET MAUI\)](#)

[Azure でのモバイル アプリの開発](#)

[.NET IoT ライブラリ](#)

[5 分で IOT の使用を開始する](#)

[LED を点滅させる](#)

[.NET IoT 101 ビデオ シリーズ](#)

ゲーム開発

[Visual Studio を使用したゲーム開発](#)

[CRYENGINE を使用して C# でゲームを構築する方法について学習する](#)

[MonoGame ライブラリを使用して C# でゲームを作成する](#)

[Unity を使用して C# で 2D ゲームと 3D ゲームを作成する方法を学習する](#)

API と言語リファレンス

.NET API および言語リファレンスのドキュメントを検索します。

.NET API リファレンス

[.NET の API リファレンス ドキュメント](#)

ASP.NET Core API リファレンス

[ASP.NET Core の API リファレンス ドキュメント](#)

.NET Framework API リファレンス

[.NET Framework の API リファレンス ドキュメント](#)

.NET ドキュメントへの投稿に関心をお持ちですか。 詳細については、[共同作成者ガイド](#)を参照してください。

Windows上のコンテナの概要

Visual Studio、Visual Studio Code、.NET、さまざまなAzure サービスなど、Windowsの Microsoft ツールとサービスによって提供されるさまざまなコンテナ サポートについて説明します。

ⓘ Note

WSL では、組み込みの Linux コンテナのサポートも追加されています (パブリックプレビューが予定されています)。詳細と進行状況の更新については、[WSL GitHub リポジトリ](#) を参照してください。

Windows上の Docker



Windows 用 Docker Desktop のインストール

インストール手順、システム要件、インストーラーに含まれる内容、アンインストール方法、安定バージョンとエッジバージョンの違い、Windowsコンテナと Linux コンテナを切り替える方法について説明します。

" Dockerを実行しているスクリーンショット



Docker を始める

Docker の向きとセットアップに関するドキュメントと、ビデオ のチュートリアルなど、開始方法に関する詳細な手順が記載されています。



MODULE

Introduction to Docker containers

32 min ★★★★★ 4.6 (3K)

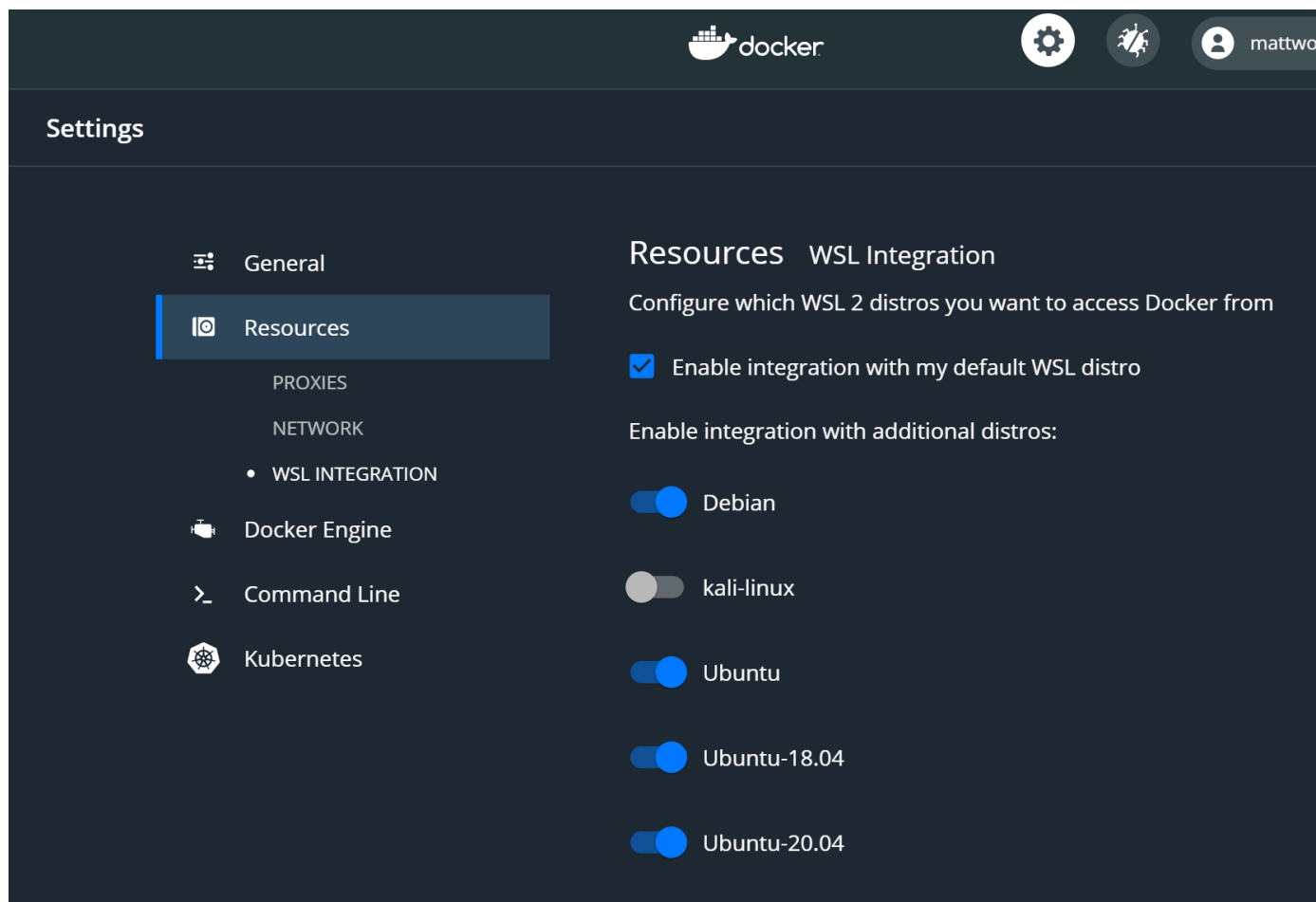
Azure

Developer

Beginner

MS Learn コース: Docker コンテナの概要

Microsoft Learn では、Docker の使用を開始し、Azure サービスとの接続に関する[コース](#)に加えて、Docker コンテナに関する無料のイントロ コースを提供しています。



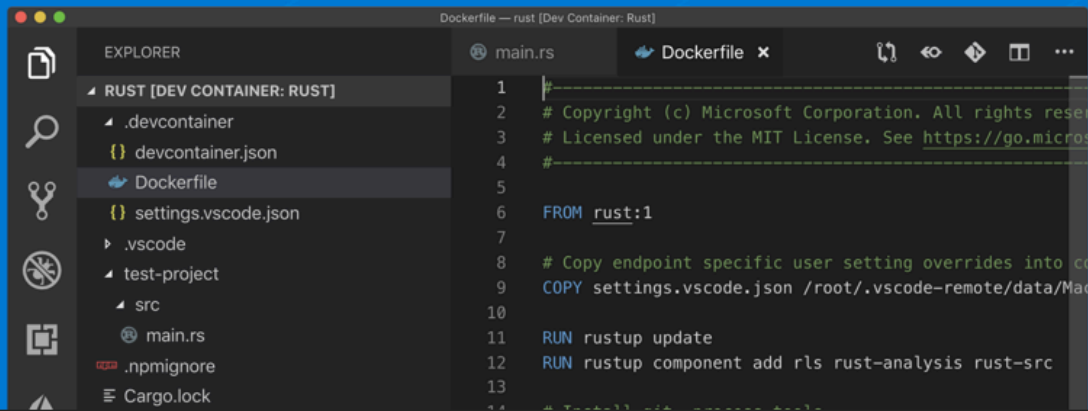
WSL 2 上で Docker リモート コンテナを始める方法

WSL 2 (Linux 用 Windows サブシステム、バージョン 2) を使用して Linux コマンド ライン (Ubuntu、Debian、SUSE など) で使用するWindows用に Docker Desktop を設定する方法について説明します。

VS Code と Docker



VS Code Remote Development in Containers

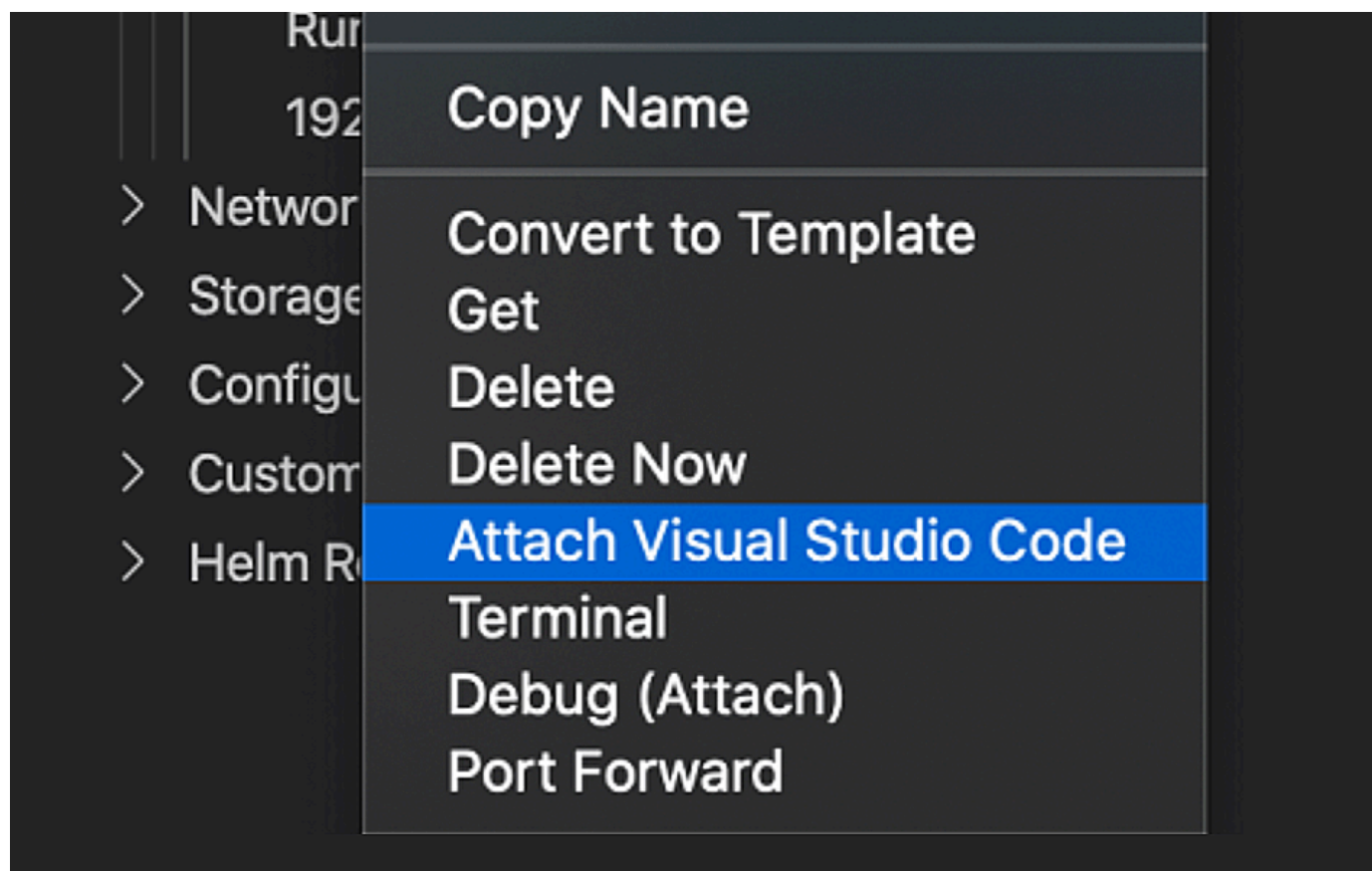


Dev Containers チュートリアル

Dev Containers 拡張機能を使用して、コンテナ内にフル機能の開発環境を設定します。

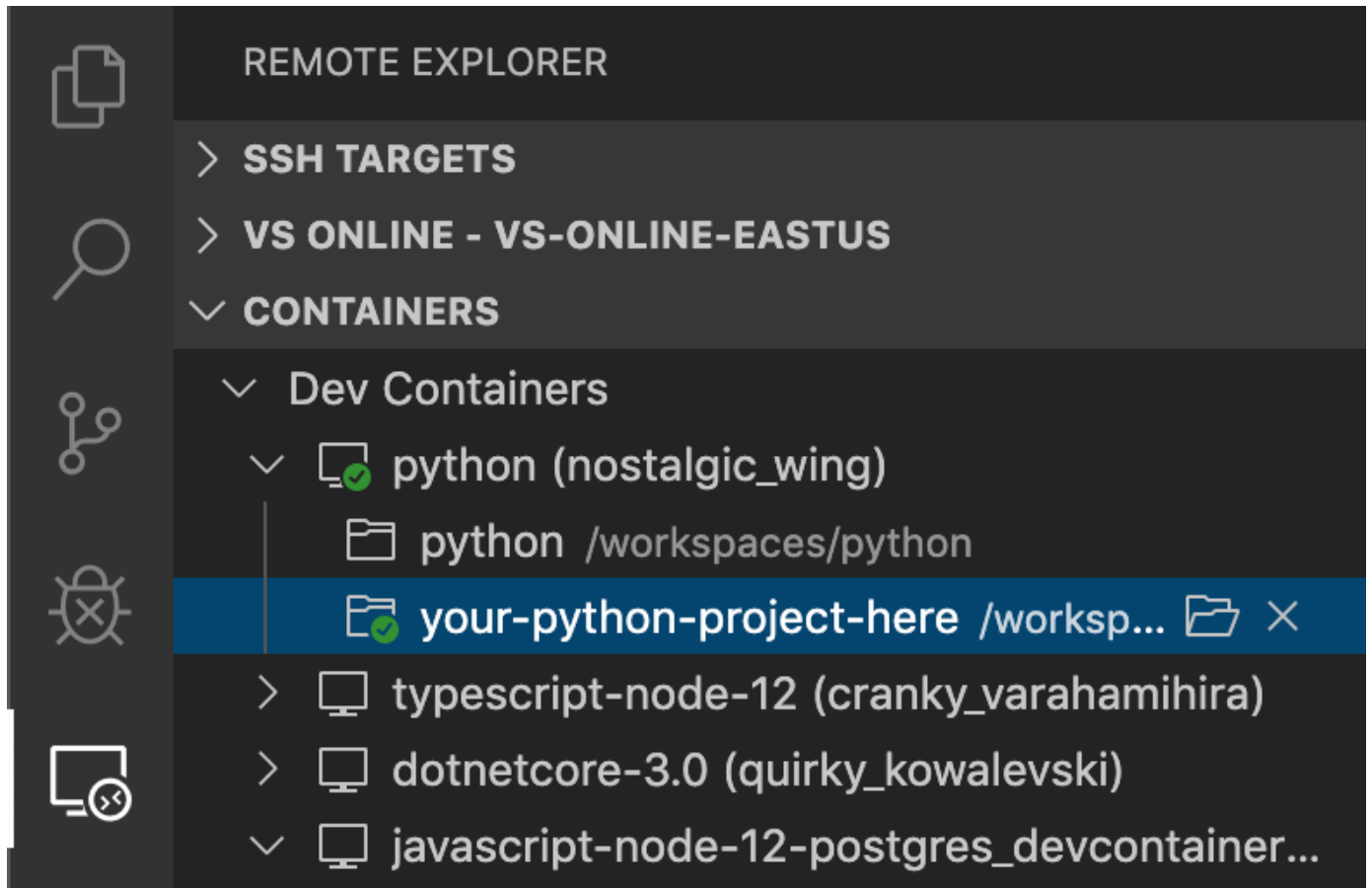
Node.js コンテナ、Python コンテナ、または ASP.NET Core コンテナを設定するチュートリアルを見つけます。

Docker のスクリーンショットを VSCode にアタッチ



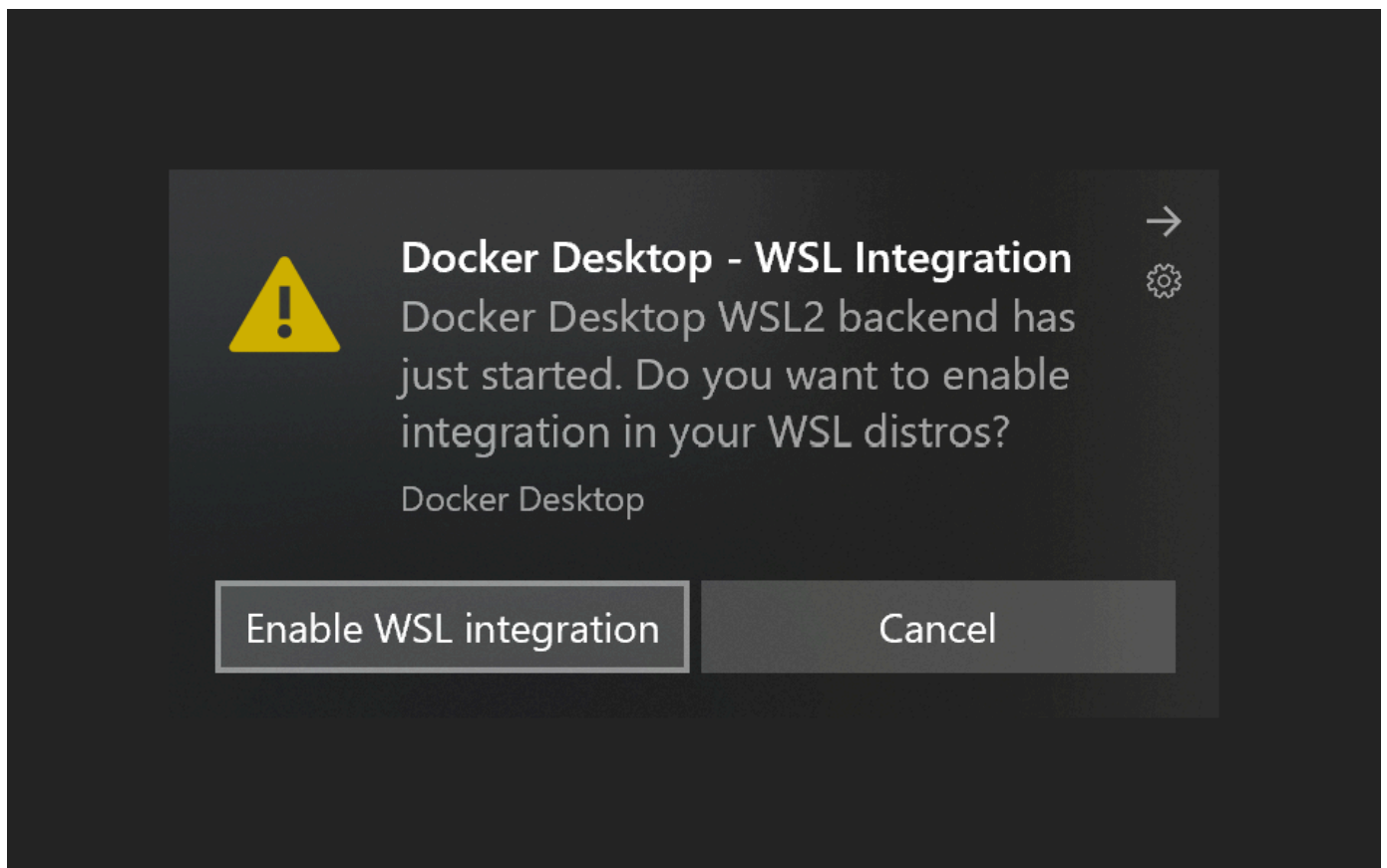
DOCKER コンテナに VS Code をアタッチする

既に実行されている Docker コンテナまたは Kubernetes クラスター内の [container](#) に Visual Studio Code をアタッチする方法について説明します。



Dev Containers のドキュメント

VS Code チームからの完全な Dev Containers リファレンス。詳細な構成、環境変数、ポートフォワーディングなどが含まれます。



Windows で Dev Containers (開発コンテナ) を設定します

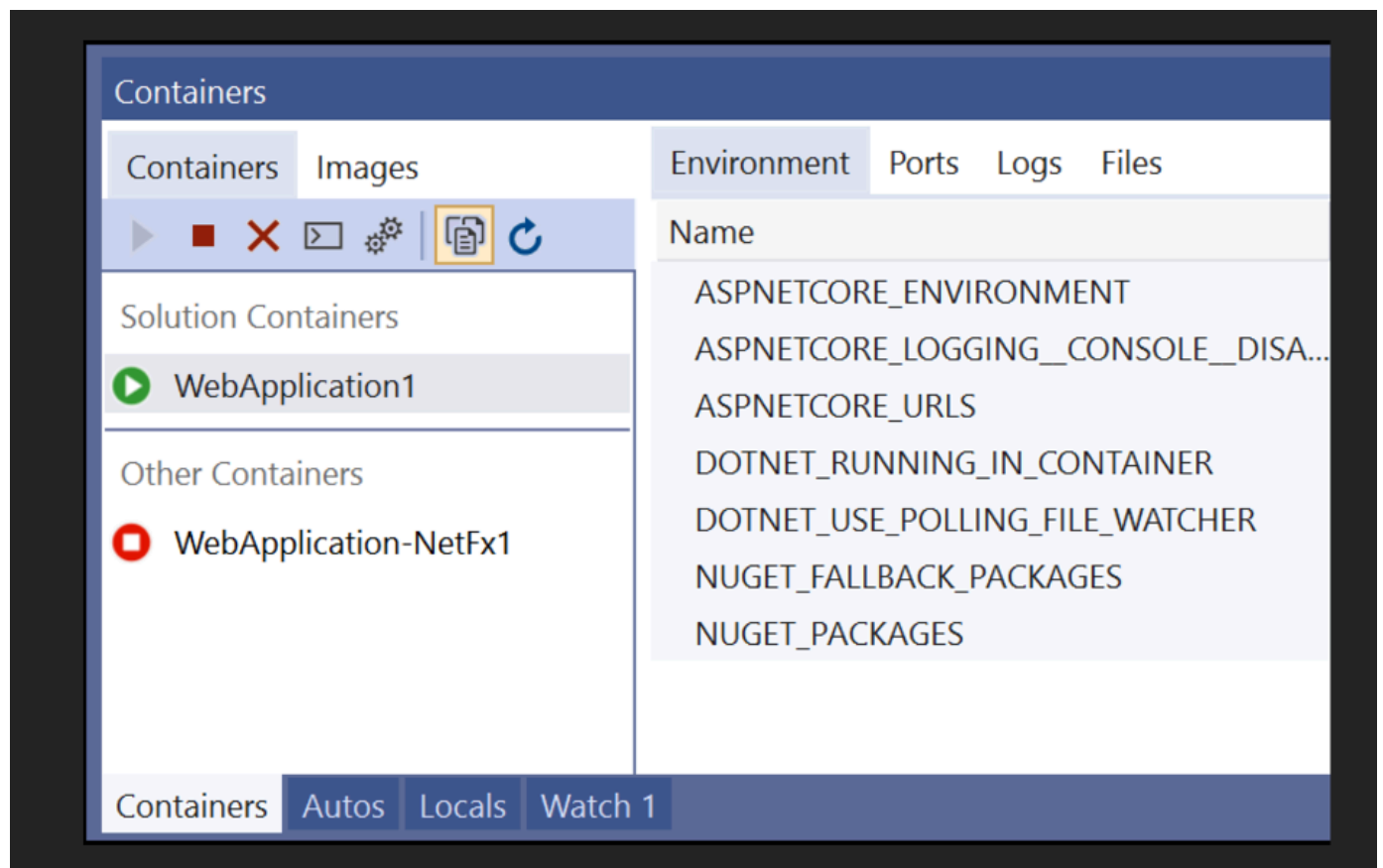
WSL 2 と Docker Desktop の構成に関するWindows固有のセットアップ ガイドと、適切なコンテナ パフォーマンスのためのファイル システムの配置要件について説明します。

Visual Studioと Docker



Visual Studio での Docker サポート

コンテナ オーケストレーションのサポートに加えて、ASP.NET プロジェクト、ASP.NET Core プロジェクト、および .NET Visual Studio のコア および .NET Framework コンソール プロジェクトで利用できる Docker のサポートについて説明します。



Quickstart: docker in Visual Studio

コンテナ化された.NET、ASP.NET、および ASP.NET Core アプリをビルド、デバッグ、実行し、それらを Azure Container Registry (ACR)、Docker Hub、Azure App Service、または独自のコンテナ レジストリに発行する方法について説明します Visual Studio。

Configure your new project

ASP.NET Core Web Application C# Windows Linux macOS Web

Project name

WebFrontEnd

Location

C:\Source\Repos\

Solution name ⓘ

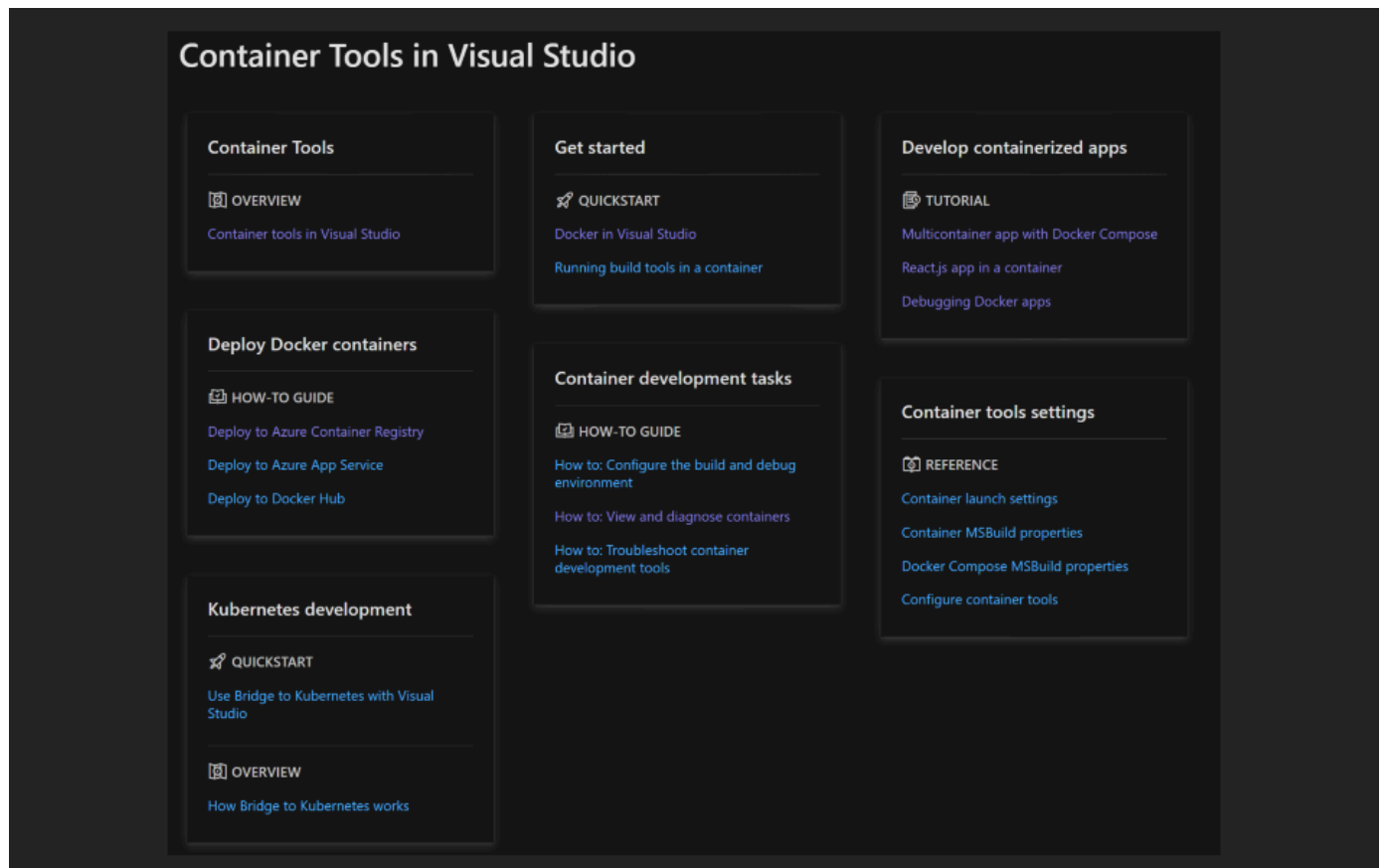
WebFrontEnd

☐ Place solution and project in the same directory

Back Create

チュートリアル: Docker Compose を使用してマルチコンテナ アプリを作成する

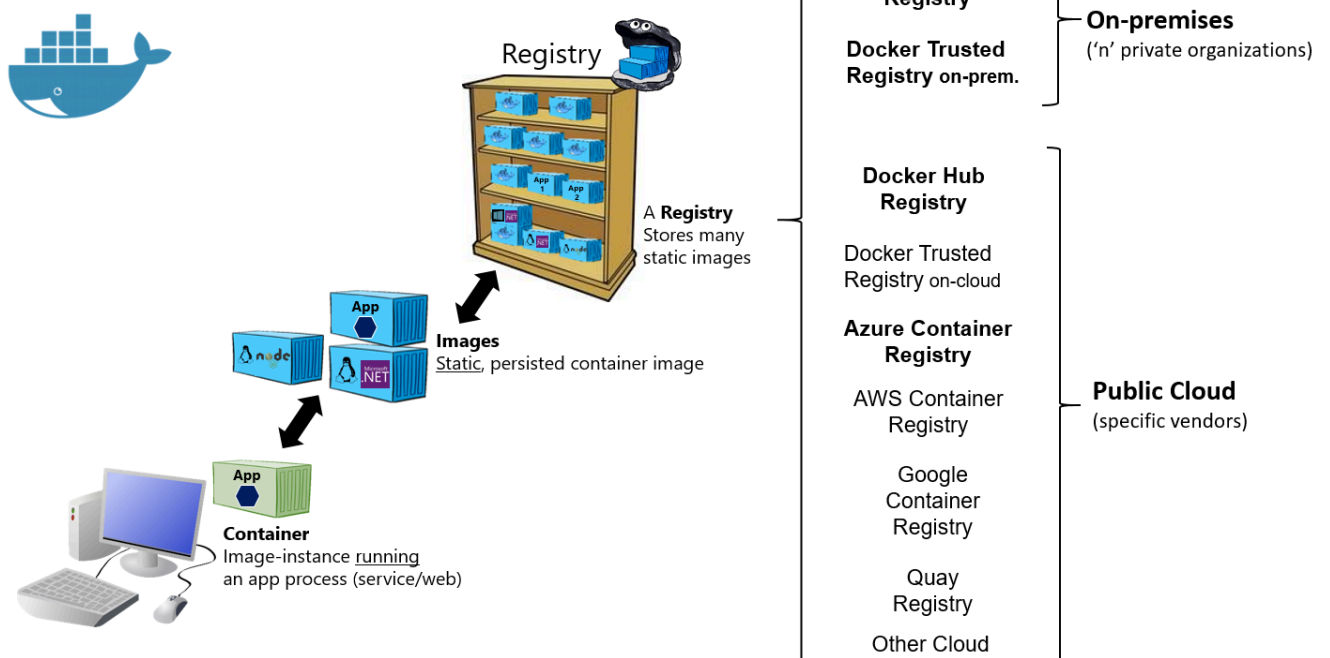
Visual Studioでコンテナ ツールを使用するときに、複数のコンテナを管理し、それらの間で通信する方法について説明します。 React シングルページ アプリで Docker を する方法などのチュートリアルへのリンクもあります。



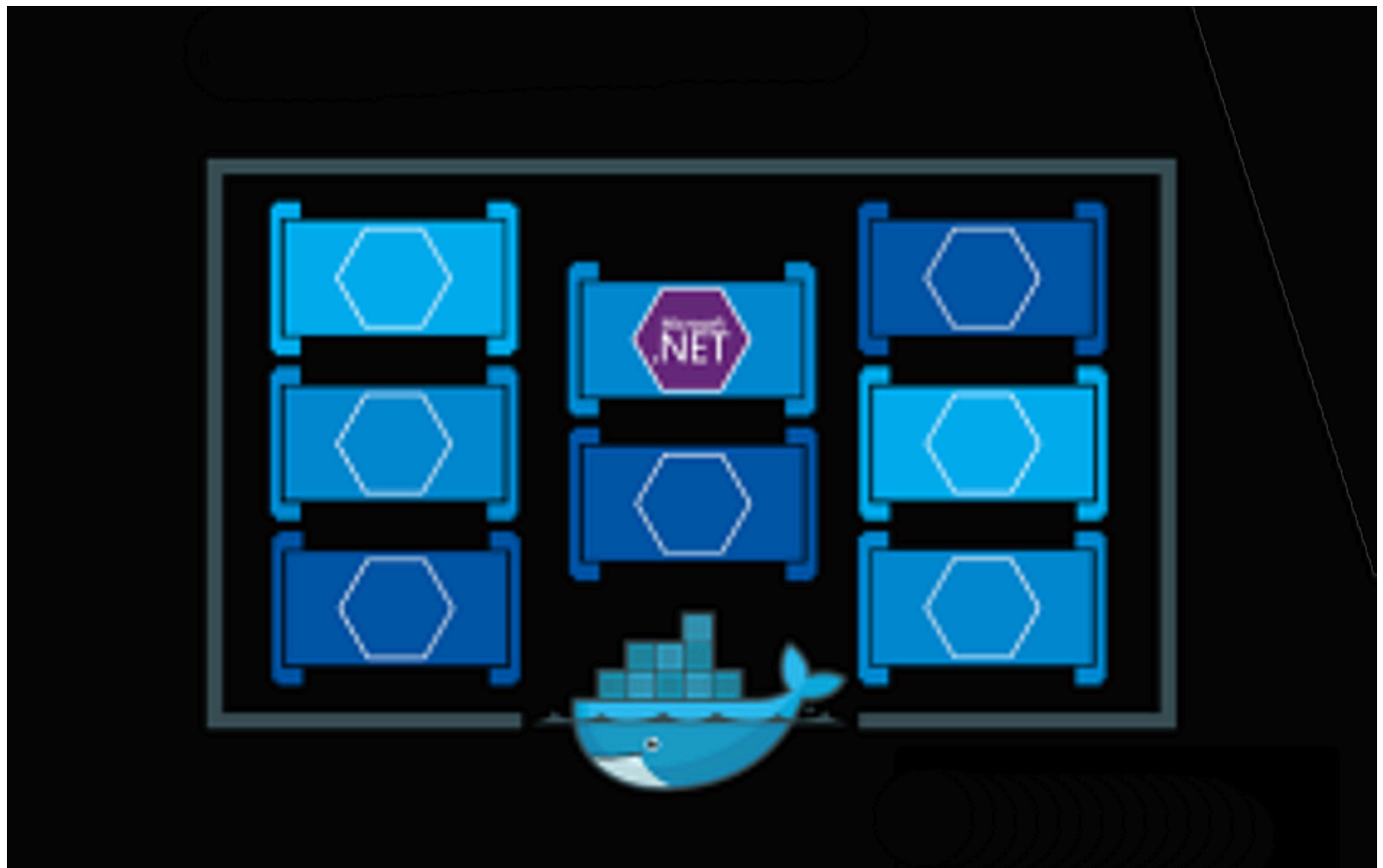
Visual Studio のコンテナツール

コンテナでビルド ツールを実行する方法、[Docker アプリ](#)のトラブルシューティング、開発ツールのトラブルシューティング、Docker コンテナのデプロイ、Visual Studioを使用した Kubernetes のブリッジ方法に関するトピックを紹介します。

Basic taxonomy in Docker

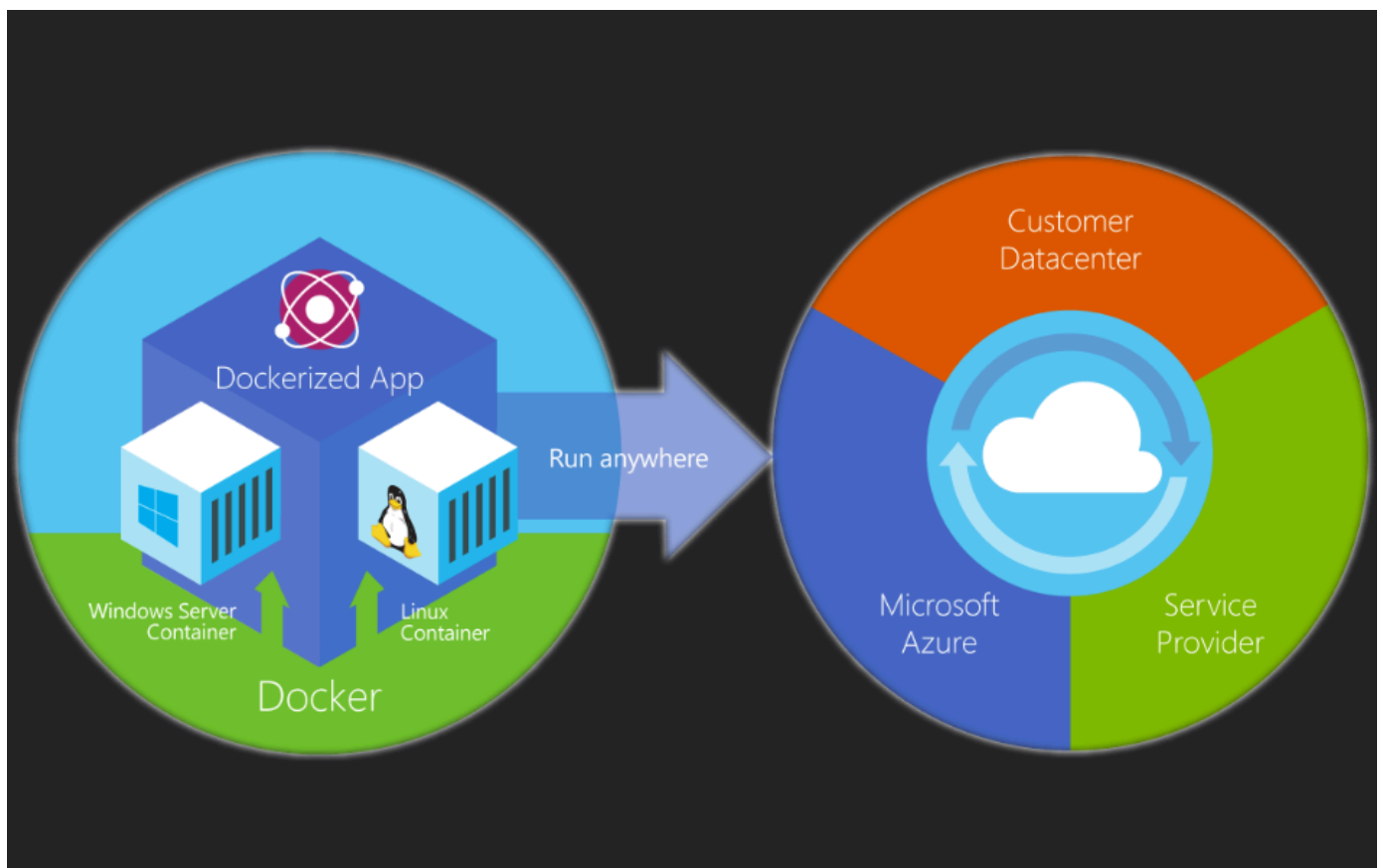


.NETと Docker



.NET ガイド: マイクロサービス アプリとコンテナ

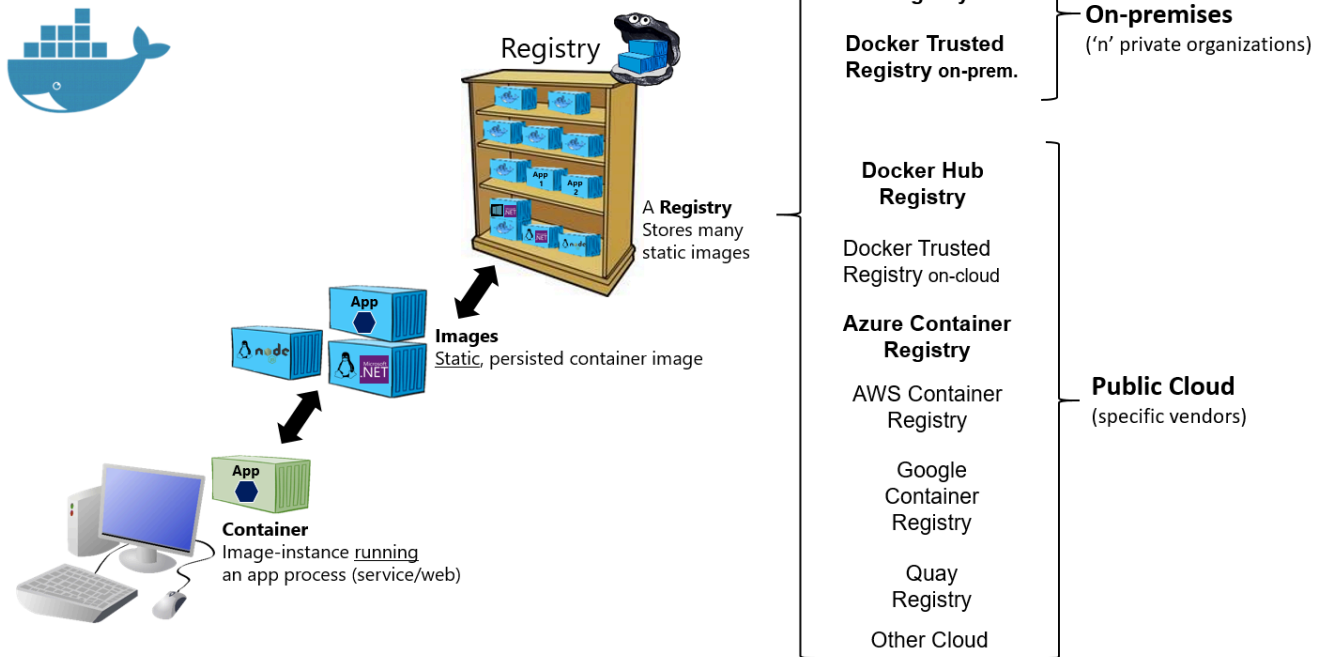
コンテナで管理されるマイクロサービス ベースのアプリの概要ガイド。



Docker とは

Docker コンテナと [仮想マシンの比較](#)、[コンテナ](#)、イメージ、レジストリの違いを説明する [Docker の用語と概念](#)の 基本的な分類など、Docker コンテナの基本的な説明。

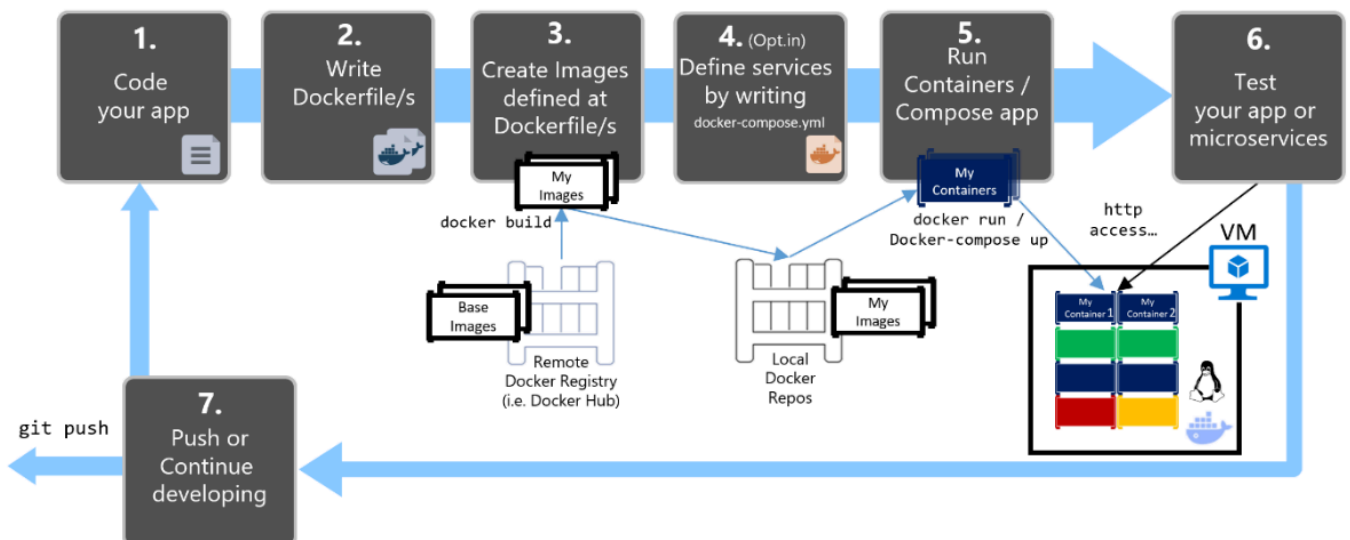
Basic taxonomy in Docker



Tutorial: .NET アプリをコンテナ化します

Dockerfile の作成、重要なコマンド、リソースのクリーンアップなど、Docker を使用して.NET アプリケーションをコンテナ化する方法について説明します。

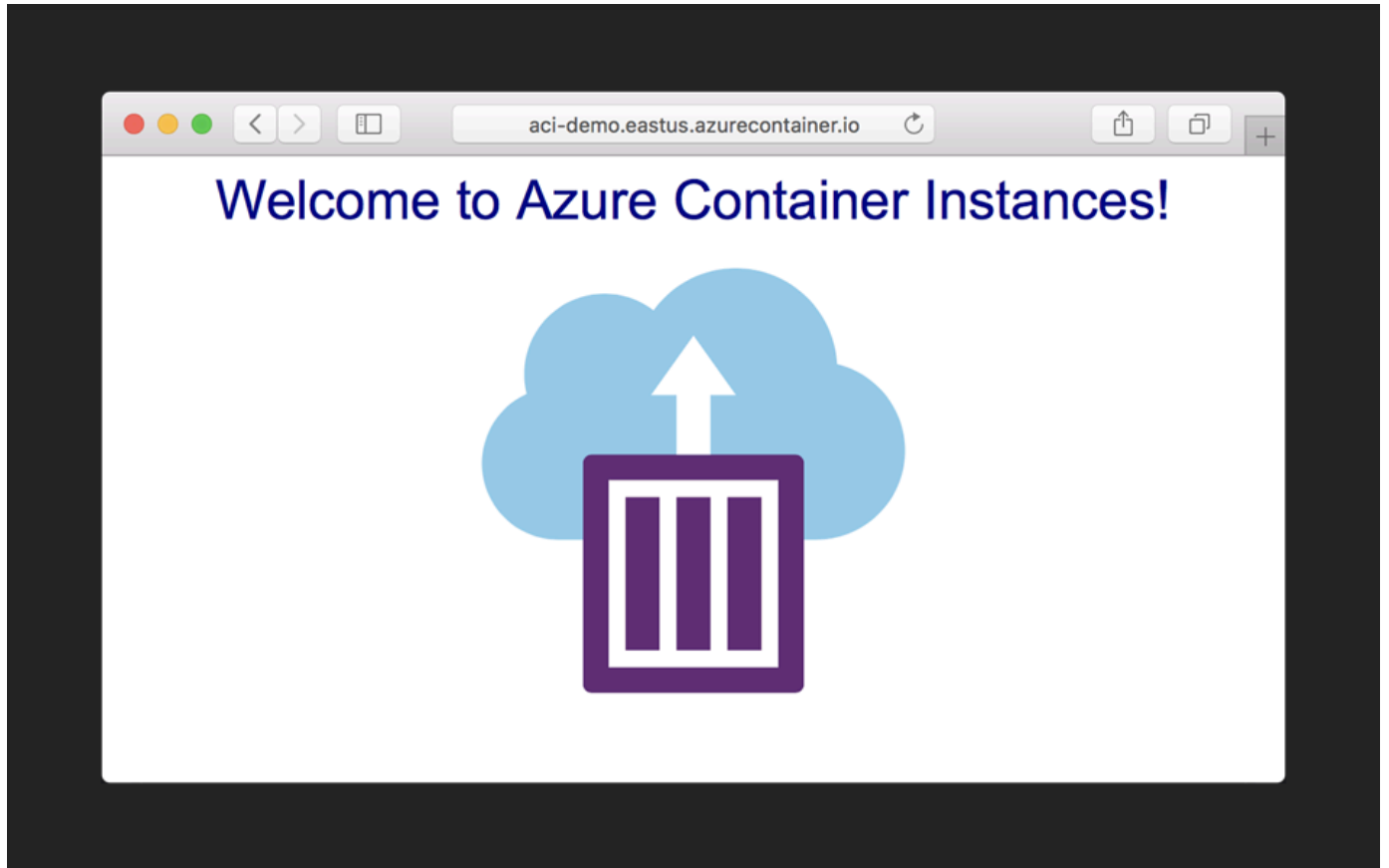
Inner-Loop development workflow for Docker apps



Docker アプリの開発ワークフロー

Docker コンテナ ベースのアプリケーションの内部ループ開発ワークフローについて説明します。

Azure Container Services



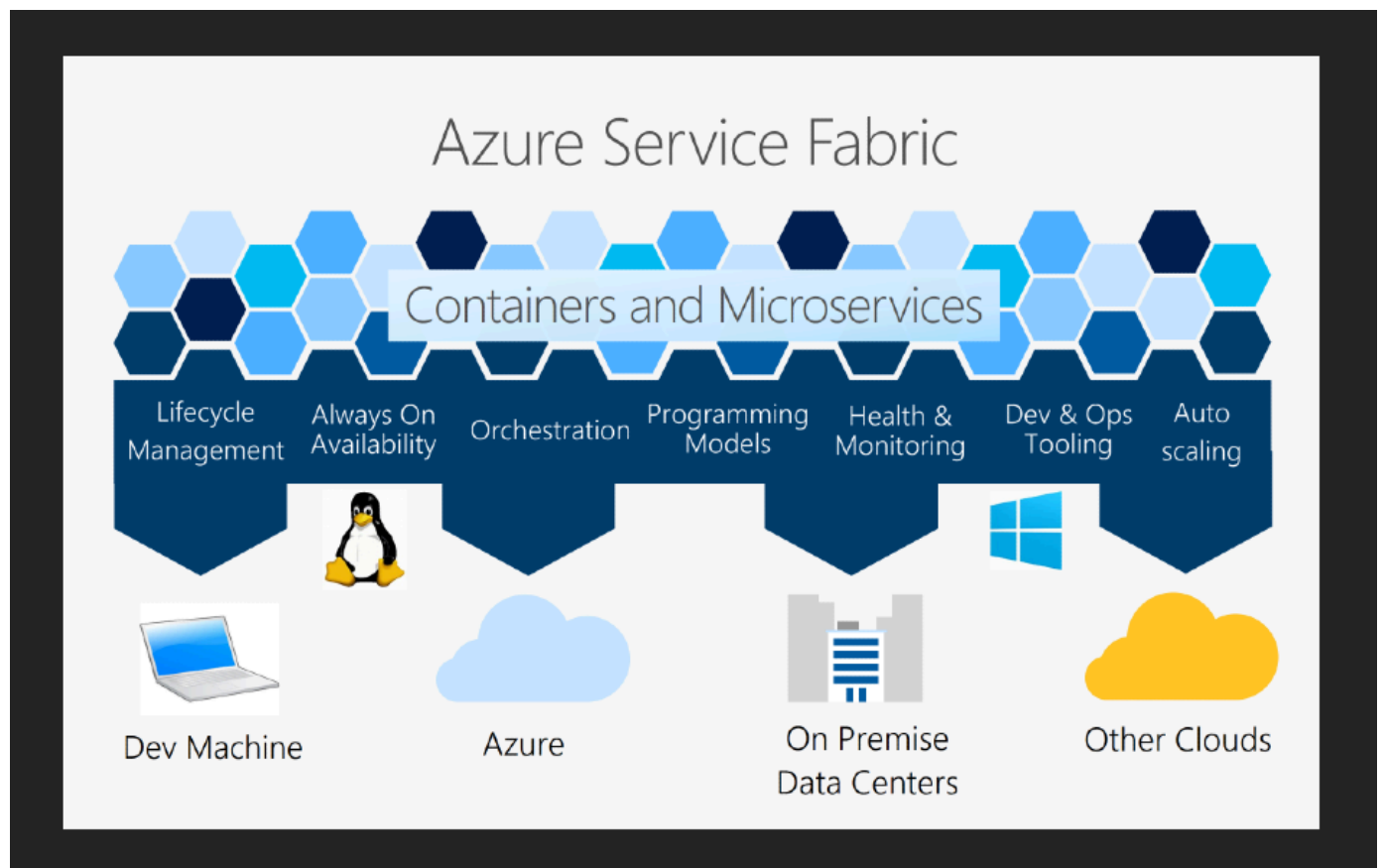
Azure Container Instances

管理されたサーバーレス Azure環境で Docker コンテナをオンデマンドで実行する方法について説明します。Docker CLI、ARM、Azure portalを使用してデプロイする方法、複数コンテナグループを作成する方法、コンテナ間でデータを共有する方法、仮想ネットワークに接続する方法などが含まれます。



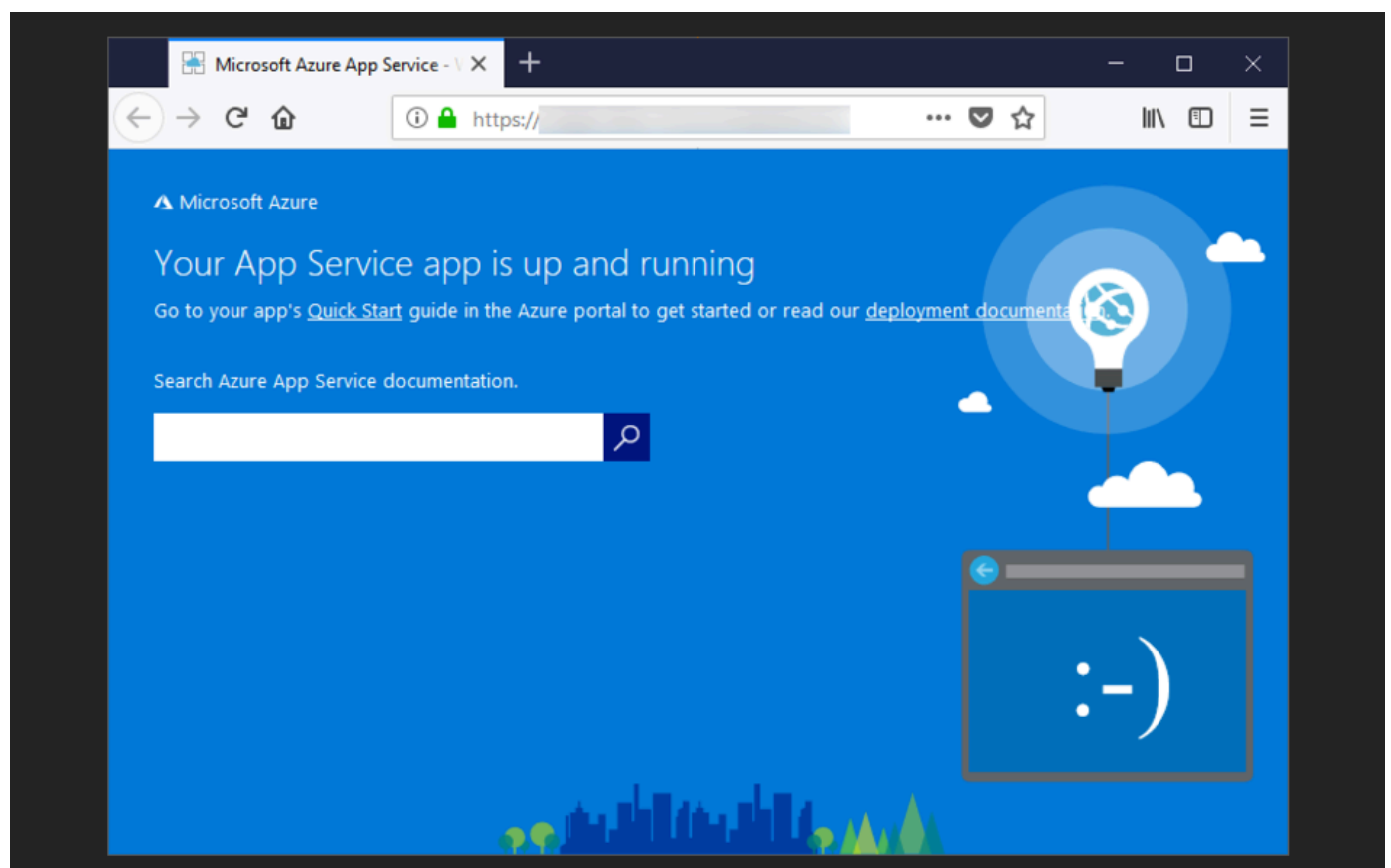
Azure Container Registry

すべての種類のコンテナ デプロイのプライベート レジストリでコンテナ イメージと成果物をビルド、格納、および管理する方法について説明します。 既存Azureコンテナ開発およびデプロイ パイプライン用のコンテナ レジストリを作成し、自動化タスクを設定し、geo レプリケーションやベスト プラクティスなど、レジストリを管理する方法について説明します。



Azure Service Fabric

スケーラブルで信頼性の高いマイクロサービスとコンテナをパッケージ化、デプロイ、管理するための分散システム プラットフォームであるAzure Service Fabricについて説明します。



Azure App Service

インフラストラクチャを管理せずに、任意のプログラミング言語で Web アプリ、モバイルバックエンド、RESTful API を構築してホストする方法について説明します。 [Azure App Service Learn](#) モジュールを試して、Docker イメージに基づいて Web アプリをデプロイし、継続的デプロイを構成します。

コンテナをサポートする [Azure サービス](#) について説明します。

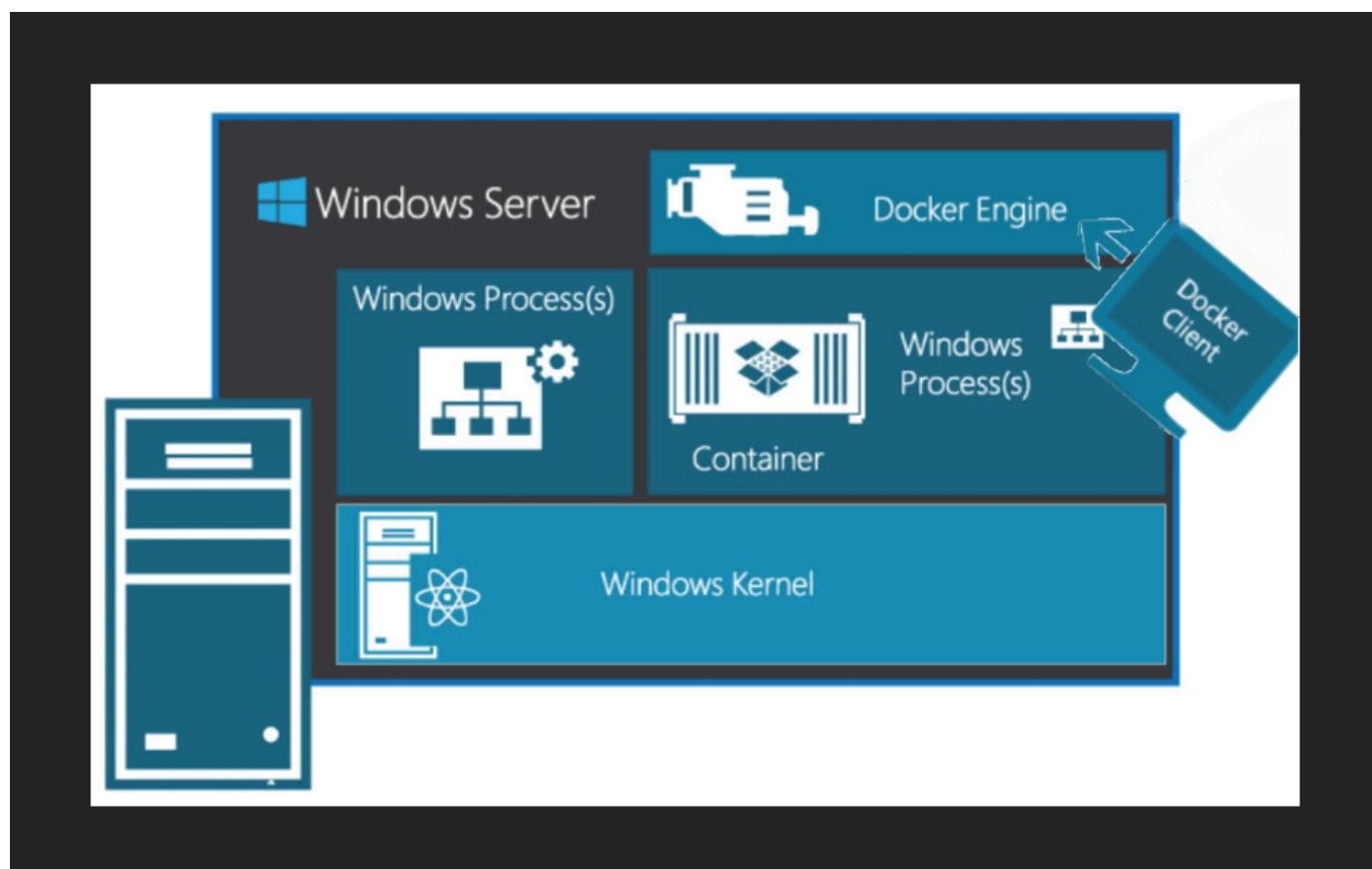
Docker コンテナの説明ビデオ

<https://www.youtube-nocookie.com/embed/0oEsMwSxBsk>

Kubernetes とコンテナオーケストレーションの解説ビデオ

<https://www.youtube-nocookie.com/embed/3RTvol-A7UQ>

Windows上のコンテナ



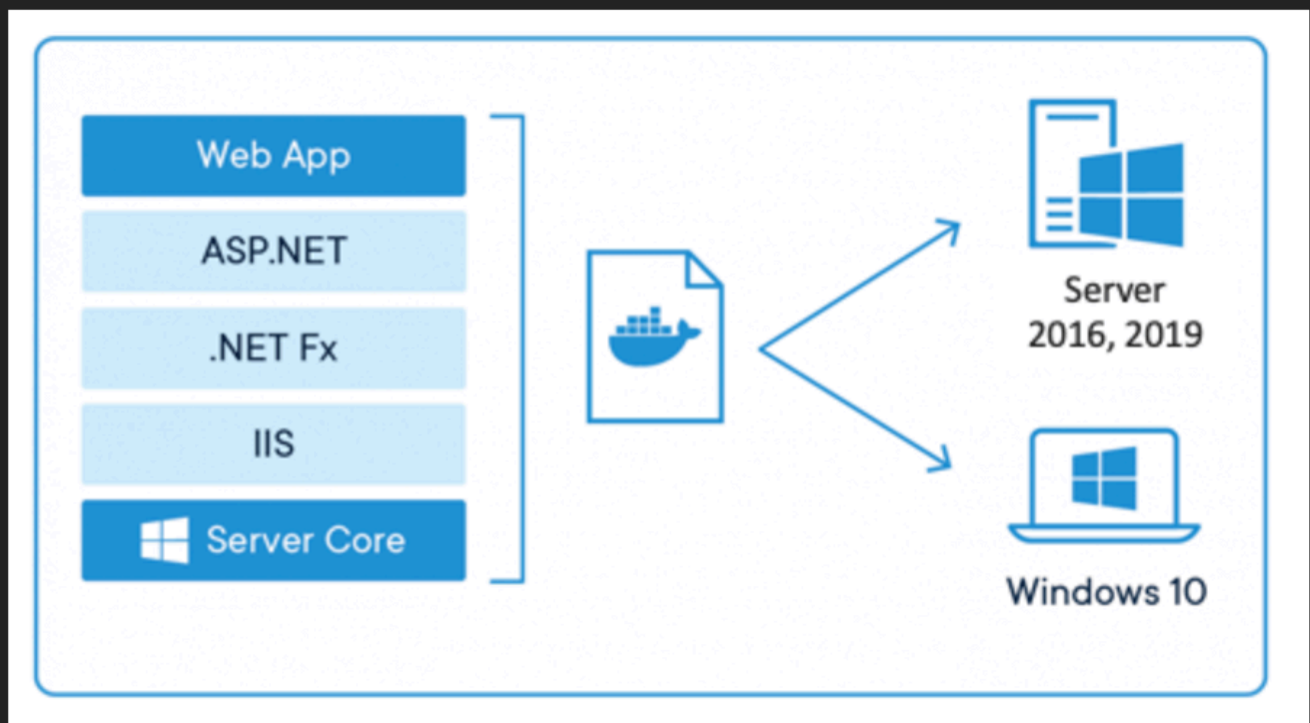
Windows コンテナ ドキュメント

アプリとその依存関係をパッケージ化し、オペレーティング システム レベルの仮想化を活用して、単一システム上の高速で完全に分離された環境を実現します。 掛け窓 コンテナ について説明します。 クイック スタート、デプロイ ガイド、サンプルなどを提供しています。



Windows コンテナに関する FAQ

コンテナに関してよく寄せられる質問を見つけます。また、StackOverflowの「Windows用 DockerとWindows上のDockerの違いは何ですか？」についての説明も参照してください。



環境を設定します

前提条件、Docker のインストール、[Windows コンテナ ベース イメージ](#)の操作など、コンテナを作成、実行、デプロイするためのWindows 11、Windows 10、またはWindows Serverを設定する方法について説明します。



[Azure Kubernetes Service \(AKS\) 上に Windows Server コンテナを作成します](#)

Azure CLIを使用して、Windows Server コンテナ内の ASP.NET サンプル アプリを AKS クラスターにデプロイする方法について説明します。

Last updated on 2026/06/13

Windowsで開発コンテナを設定する

開発コンテナを使用すると、Docker コンテナを完全な開発環境として使用できます。これは、リポジトリにチェックインされた `devcontainer.json` ファイルによって定義されます。プロジェクトを開くすべてのユーザーは、ローカル コンピューターにインストールされている内容に関係なく、同じツール、拡張機能、設定を取得します。

このページでは、Windows固有のセットアップについて説明します。開発コンテナの概要と動作の詳細については、VS Code Web サイトの [Dev Containers のドキュメント](#) を参照してください。

前提条件

Windows上の開発コンテナには、次のものがが必要です。

- WSL 2 - Linux 用 Windows サブシステムバージョン 2。 [WSL をまだインストール](#) していない場合はインストールします。
- WSL 2 バックエンドが有効になっている **Docker Desktop for Windows**。 [Docker Desktop をダウンロード](#) し、インストーラーに従います。セットアップ時に、Docker Desktop の設定 ([設定]>) で **[WSL 2 ベースのエンジンを使用する]** が選択されていることを確認します。
- Visual Studio Code — [DOWNload VS Code](#)。
- Dev Containers **拡張機能** - VS Code Marketplace から [Dev Containers 拡張機能](#) をインストールします。

ファイルを保存する場所が重要

📌 重要

Windowsでは、Dev Container のパフォーマンスはプロジェクト ファイルの場所によって大きく異なります。プロジェクトを **WSL 2 ファイル システム** (たとえば、`/home/yourname/projects/`) に格納します。Windows ファイル システム (たとえば、`C:\Users\yourname\projects\`) には格納しません。

ファイルが Windows ファイル システム (`C:\`) にある場合、Docker は OS 間のファイル共有を介してアクセスします。これは非常に低速です。ファイルが WSL 2 ファイル システムにある場

合、Docker ではネイティブ Linux I/O が使用され、パフォーマンスは大幅に向上します。特にビルドおよびファイル監視ツールの場合はパフォーマンスが大幅に向上します。

WSL 2 ファイル システムに複製するには、[スタート] メニューまたは Windows ターミナルから WSL ディストリビューション (Ubuntu など) を開き、そこで複製します。

Bash

```
cd ~  
mkdir projects && cd projects  
git clone https://github.com/your-org/your-repo.git
```

次に、WSL から VS Code でそのフォルダーを開きます。

Bash

```
code your-repo
```

VS Code は WSL に接続し、`devcontainer.json` が存在する場合は検出します。

開発コンテナでプロジェクトを開く

VS Code でプロジェクトを開くと (WSL に接続されます):

1. **F1** キーを押し、[開発コンテナ: コンテナで再度開く] を選択します。
2. VS Code は、`.devcontainer/devcontainer.json` で定義されているコンテナ イメージをビルドします (存在しない場合は追加するように求められます)。
3. ビルドが完了すると、VS Code は構成されたすべてのツールと拡張機能を使用してコンテナ内に再接続します。

ローカル環境に戻すには、**F1** キーを押し、[開発コンテナ: フォルダーをローカルに再度開く] を選択します。

既存のプロジェクトに開発コンテナを追加する

プロジェクトにまだ `devcontainer.json` がない場合:

1. **F1** キーを押し、[開発コンテナ: 開発コンテナ構成ファイルの追加] を選択します。
2. 基本イメージ (Node.js、Python、汎用 Debian/Ubuntu イメージなど) を選択します。
3. VS Code では、ソース管理にチェックインできる `.devcontainer/devcontainer.json` ファイルが作成されます。

`devcontainer.json` オプションの完全なリファレンスについては、Dev Container Specification サイト [devcontainer.json リファレンス](#) を参照してください。

Troubleshooting

Container は起動しますが、ファイルの変更は検出されませんプロジェクトはWindowsファイルシステムに格納されている可能性があります。WSL に移動します ([上記の「ファイルの保存場所」](#) を参照してください)。

Docker Desktop が起動しないか、WSL 統合が見つからない Docker Desktop を開き、 **設定** > **Resources** > **WSL 統合** に移動し、インストールされている WSL ディストリビューションの統合を有効にします。

VS Code がコンテナに接続できない VS Code を開く前に、Docker Desktop が実行されていることを確認します。Docker Desktop システム トレイ アイコンを確認します。

次のステップ

- [Dev Containers のドキュメント](#) - VS Code チームからの完全なリファレンス
- [devcontainer.json リファレンス](#) — すべての構成オプション
- [WSL 2 で Docker リモート コンテナを始める](#) - 全体的な WSL + Docker セットアップ ガイド
- [WSL をインストール](#) する — WSL をまだ設定していない場合

PowerShell のドキュメント

PowerShell の公式製品ドキュメント



作業の開始
PowerShell と
は



ダウンロード
セットアップと
インストール



攻略ガイド
サンプル スク
リプト



デプロイ
PowerShell ギ
ャラリー [🔗](#)



リファレンス
PowerShell モ
ジュールブラ
ウザー



ARCHITECTU...
GitHub の
PowerShell [🔗](#)

PowerShell エディション + ツール

PowerShell をサポートする利用可能なエディション、ツール、テクノロジー



PowerShell

Windows に PowerShell をイ
ンストールする

Linux に PowerShell をイン
ストールする

macOS に PowerShell をイン
ストールする

PowerShell の新機能



PowerShell ユーティ リティ モジュール

PowerShell ユーティリティ
モジュール

Crescendo の概要

PlatyPS の概要

PSScriptAnalyzer の概要

SecretManagement の概要



Desired State Configuration

PowerShell DSC の概要

マイクロソフト DSC 3.0

PowerShell DSC 2.0 (英語)

PowerShell DSC 1.1

DSC コミュニティ [🔗](#)



PowerShell ギャラリー

—

[PowerShell ギャラリー](#) 

[PowerShell のパッケージ管理](#)

[ベストプラクティス](#)

[FAQ](#)

開発者向けリソース

[Visual Studio Code PowerShell 拡張機能](#) 

[VS Code を使用したリモート編集とデバッグ](#)

[Windows PowerShell SDK のドキュメント](#)

[PowerShell SDK API リファレンス](#)

関連テクノロジー

[Azure Cloud Shell のクイックスタート](#)

[Azure PowerShell モジュール](#)

[Azure Automation Runbook](#)

[Azure で PowerShell 関数を作成する](#)

[Windows 管理モジュール](#)

コミュニティ リソース

他の PowerShell ユーザーと接続する



[ユーザー グループ](#) 



[PowerShell Tech Community](#) 



[DSC コミュニティ](#) 



[PowerShell.org](#) 



[Stack Overflow](#) 



[r/PowerShell](#) 



[Powershell Slack](#) 



[Powershell Discord](#) 



[Spiceworks](#) 



[デジタル アート](#)



[PowerShell ポッドキャスト](#) 



[PowerShell Summit のビデオ](#) 



PowerShell Conference EU の動画 [🔗](#)

PowerShell チーム

PowerShell チームと通信する

[PowerShell チームのブログ](#) [🔗](#)

公式ニュースとお知らせ

[PowerShell コミュニティ ブログ](#) [🔗](#)

コミュニティが PowerShell を学習し、分析情報を共有するための場所

[Twitter の PowerShell チーム](#) [🔗](#)

PowerShell ニュースを 280 文字ずつお届け

[GitHub の問題](#) [🔗](#)

現在のリリースのバグの場所

Rust を使用した Windows での開発の概要

[Rust](#) は、ガベージ コレクターを使用せずに、パフォーマンス、信頼性、メモリの安全性を確保するために設計されたシステム プログラミング 言語です。オペレーティング システム、コマンド ライン ツール、Web サーバー、および低レベルの制御が重要な場所で広く使用されています。

Rust は、長年にわたって最も賞賛されている言語として Stack Overflow の [年間開発者アンケート](#) を突破しており、Microsoft は [Rust Foundation](#) の設立メンバーです。

主な概念

- **Cargo** - Rust のパッケージ マネージャーとビルド ツール。プロジェクトの作成、依存関係の追加、ビルド、テストなど、ほぼすべての用途に使用します。
- **クレート** — Rust のコンパイルの基本単位。クレートには、バイナリ (実行可能) またはライブラリを指定できます。
- **rustup** — Rust ツールチェーンの公式インストーラーとバージョン マネージャー。コンパイラとツールを最新の状態に保ちます。
- **crates.io** — [crates.io](#) のコミュニティ パッケージ レジストリ。

次のステップ

- [Windows で Rust 開発環境を設定する](#)
- [Windows 用 Rust と windows クレート](#)
- [Visual Studio Code の Rust](#)

Last updated on 2026/03/27

Windows で Rust 開発環境を設定する

手順 1: MSVC C++ ビルド ツールをインストールする

Windows 上の Rust には、前提条件として Microsoft の C++ ビルド ツールが必要です。C++ ワークロードを使用した[デスクトップ開発](#)で Visual Studio が既にインストールされている場合は、この手順をスキップできます。

それ以外の場合は、[Microsoft C++ ビルド ツール](#)をインストールし、C++ ワークロードを使用した[デスクトップ開発](#)を選択します。

ⓘ Note

Microsoft C++ ビルド ツールを使用するには、有効な Visual Studio ライセンス (Community、Pro、または Enterprise) が必要です。Community エディションは、学生、オープンソースの共同作成者、個人向けに無料です。

手順 2: Rust をインストールする

公式の Rust インストーラー `rustup` は、コンパイラ、Cargo、将来の更新プログラムなど、すべてを処理します。

WinGet

PowerShell

```
winget install Rustlang.Rustup
```

インストール後、新しいターミナルを開き、セットアップを確認します。

PowerShell

```
cargo --version  
rustc --version
```

💡 ヒント

Rust を最新の状態に保つには、`rustup update` を定期的に実行します。

手順 3: Visual Studio Code と Rust 拡張機能をインストールする

1. [Visual Studio Code](#) をダウンロードしてインストールします。
2. VS Code Marketplace から [rust-analyzer 拡張機能](#) をインストールします。これにより、コード補完、インライン エラー、go-to-definition、デバッグのサポートが追加されます。

次のステップ

Rust について学習する

- [Windows 用 Rust と Windows クレート](#) - Rust から Windows API を直接呼び出す
- [Visual Studio Code の Rust](#) - VS Code Rust ワークフローの詳細
- [Rust プログラミング言語の本](#) - 公式無料の Rust ブック

Last updated on 2026/03/27

Windows 用 Rust と windows クレート

<https://www.youtube-nocookie.com/embed/-oZrsCPKsn4>

Windows 用 Rust の概要

Rust を使用した Windows での開発の概要 のトピックでは、*Hello, world!* というメッセージを出力する簡単なアプリを紹介しました。ただし、Windows では Rust を使用できるだけでなく、Rust を使用して Windows 用のアプリを作成することもできます。

最新の更新プログラムはすべて、GitHub の [Rust for Windows リポジトリのリリース ログ](#) で確認できます。

Windows 用 Rust を使用すると、[windows クレート](#) を介して、過去、現在、将来の任意の Windows API を直接シームレスに使用できるようになります (クレートは、バイナリ、ライブラリ、1 つにビルドされるソースコードを表す Rust の用語です)。

[CreateEventW](#) や [WaitForSingleObject](#) などの時代を超越した関数、[Direct3D](#) などの強力なグラフィックス エンジン、[CreateWindowExW](#) や [DispatchMessageW](#) などの従来のウィンドウ関数、または [コンポジション](#) などの最近のユーザー インターフェイス (UI) フレームワークのいずれであっても、[ウィンドウ クレート](#) で説明しました。

[win32metadata](#) プロジェクトは、Win32 API のメタデータを提供することを目的としています。このメタデータでは、API サーフェス (厳密に型指定された API シグネチャ、パラメーター、および型) について説明します。これにより、Windows API 全体を Rust (および C# や C++ などの言語) で自動的かつ完全な方法でできるようになります。また、「[Win32 API をより多くの言語にアクセスできるようにする](#)」も参照してください。

Rust 開発者は、Cargo (Rust のパッケージ管理ツール) と <https://crates.io> (Rust コミュニティのクレート レジストリ) を使用して、プロジェクトの依存関係を管理します。良いニュースは、Rust アプリから [Windows クレート](#) を参照し、すぐに Windows API の呼び出しを開始できることです。[windows クレートに関する Rust のドキュメント](#) は、<https://docs.rs> でも参照できます。

C++/WinRT と同様に、[Rust for Windows](#) は GitHub で開発されたオープン ソース言語プロジェクトです。[Windows 用 Rust](#) に関する質問がある場合、または問題を報告する場合は、Rust for Windows リポジトリを使用します。

Rust for Windows リポジトリには、従うことができる [いくつかの簡単な例](#) もあります。また、Robert Mikhayelyan の [マインスイーパー](#) の形式で優れたサンプル アプリがあります。

Windows 用 Rust に貢献する

[Rust for Windows](#) が投稿を歓迎します。

Windows API の Rust ドキュメント

Rust for Windows の利点は、Rust 開発者が楽しむ洗練されたツールチェーンから得られます。ただし、Windows API 全体をすぐに使うのが少し難しいと思われる場合は、[Windows API の Rust ドキュメント](#) もあります。

このリソースは基本的に、Windows API と型が慣用的な Rust にどのように投影されるかを説明します。これを使用して、知る必要がある API を参照または検索し、呼び出し方法を知る必要があります。

Windows 用 Rust を使用したアプリの作成

次のトピックは [RSS リーダー チュートリアル](#) です。ここでは、Rust for Windows を使用して簡単なアプリを記述する方法について説明します。

関連項目

- [Rust を使用した Windows での開発の概要](#)
- [RSS リーダーのチュートリアル](#)
- [windows クレート](#)
- [windows クレートに関するドキュメント](#)
- [Win32 メタデータ](#)
- [より多くの言語で Win32 API のアクセシビリティを高める](#)
- [Windows API 向けの Rust ドキュメント](#)
- [Windows 用 Rust](#)
- [Minesweeper サンプル アプリ](#)

RSS リーダー チュートリアル (VS Code を使用した Windows 用 Rust)

前のトピックでは、[Rust for Windows](#) と [Windows クレート](#)について説明しました。

次に、Really Simple Syndication (RSS) フィードからブログ投稿のタイトルをダウンロードする簡単なコンソール アプリを作成して、Windows 用 Rust を試してみましょう。

1. コマンド プロンプト (`cmd.exe`) を起動し、Rust プロジェクトを保持するフォルダーに `cd` します。
2. Cargo を使用して、`rss_reader`という名前 の新しい Rust プロジェクトを作成し、新しく作成したフォルダーに `cd` します。

コンソール

```
> cargo new rss_reader
> Created binary (application) `rss_reader` package
> cd rss_reader
```

3. 次に、VS Code で `rss_reader` プロジェクトを開きます。

コンソール

```
code .
```

4. メイン `rss_reader` プロジェクトを実装しましょう。まず、プロジェクトのルートにある `Cargo.toml` ファイルを開きます。 `Cargo.toml` ファイルは、Rust プロジェクトを記述するテキスト ファイルです。これには、依存関係も含まれます。

次の一覧に示すように、ウィンドウ クレートに依存関係を追加します。 `windows` クレートは大きいです。ビルド時間を短縮するために、このコードに必要な

`Foundation_Collections` 機能と `Web_Syndication` 機能のみを選択します。

toml

```
# Cargo.toml
...

[dependencies.windows]
version = "0.62.0"
```

```
features = [  
    "Foundation_Collections",  
    "Web_Syndication",  
]
```

❗ Note

`windows` クレーンのリリースが頻繁に発生します。最新バージョンの crates.io/crates/windows を確認し、それに応じて更新します。

5. 次に、`rss_reader` プロジェクトの `src/main.rs` ソースコードファイルを開きます。Cargo の既定の "Hello, world!" コードが表示されます。の先頭に次の `main.rs` ステートメントを追加します。

Rust

```
// src/main.rs  
use windows::{  
    core::*,  
    Foundation::Uri,  
    Web::Syndication::SyndicationClient  
};  
  
fn main() {  
    println!("Hello, world!");  
}
```

`use` 宣言によって、使用する型へのパスが短縮されます。前に説明した `Uri` 型があります。

6. 新しい **URI** を作成するには、Cargo の既定の `main` 関数を次のように置き換えます。

Rust

```
// src/main.rs  
...  
  
fn main() -> Result<> {  
    let uri = Uri::CreateUri(h!("https://blogs.windows.com/feed"))?;  
  
    Ok(())  
}
```

`main` 関数の戻り値の型が `windows::core::` からの `Result` であることに注意してください。オペレーティング システム (OS) API からのエラーに対処するのが一般的であるため、処理

が容易になります。 `windows::core::Result` は、エラー伝達と簡潔なエラー処理に役立ちます。

コード行の末尾に疑問符演算子が表示されます。 入力を節約するために、Rust のエラー伝達と短絡ロジックを使用します。 つまり、この簡単な例では、一連の手動エラー処理を行う必要はありません。 Rust のこの機能の詳細については、[エラー処理を容易にする？ 演算子](#)を参照してください。

また、`windows` クレートの `h!` マクロにも注目してください。 これを使用して、Rust 文字列リテラルから `HSTRING` 参照を構築します。 WinRT API は、文字列値に **対して** `HSTRING` を広範囲に使用します。

7. RSS フィードをダウンロードするには、新しい `SyndicationClient` を作成します。

Rust

```
// src\main.rs
...

fn main() -> windows::core::Result<()> {
    let uri = Uri::CreateUri(h!("https://blogs.windows.com/feed"))?;
    let client = SyndicationClient::new()?;

    Ok(())
}
```

新しい関数は Rust コンストラクターです。 ウィンドウ クレート内のすべてのオブジェクトは、Rust の規約に従って、コンストラクターを **new** と名付けています。

8. **これで**、`SyndicationClient` を使用してフィードを取得できます。

Rust

```
// src\main.rs
...

fn main() -> windows::core::Result<()> {
    let uri = Uri::CreateUri(h!("https://blogs.windows.com/feed"))?;
    let client = SyndicationClient::new()?;
    let feed = client.RetrieveFeedAsync(&uri)?.get()?;

    Ok(())
}
```

`RetrieveFeedAsync` は非同期 API であるため、ブロッキング `get` 関数を使用して例を単純に保ちます。 または、`await` 関数内で `async` 演算子を使用して、結果を協調的に待機する

こともできます。グラフィカル ユーザー インターフェイスを備えたより複雑なアプリでは、`async` が頻繁に使用されます。

9. これで、結果の項目を反復処理して、タイトルだけを出力してみましょう。また、一部の RSS フィードではこれを必要とするため、ユーザー エージェント ヘッダーを設定するための追加のコード行がいくつか以下に表示されます。

Rust

```
// src\main.rs
...

fn main() -> windows::core::Result<()> {
    let uri = Uri::CreateUri(h!("https://blogs.windows.com/feed"))?;
    let client = SyndicationClient::new()?;

    client.SetRequestHeader(
        h!("User-Agent"),
        h!("Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0)"),
    );

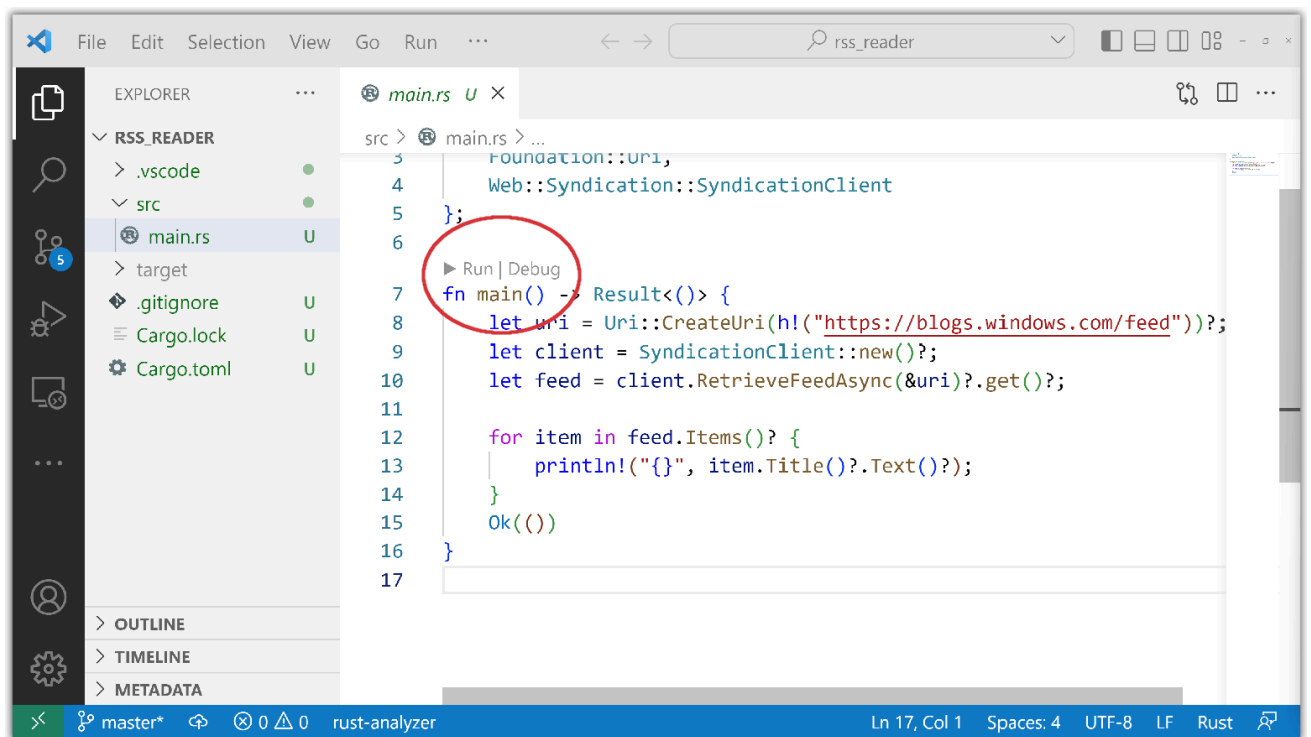
    let feed = client.RetrieveFeedAsync(&uri)?.get()?;

    for item in feed.Items()? {
        println!("{}", item.Title()?.Text()?);
    }

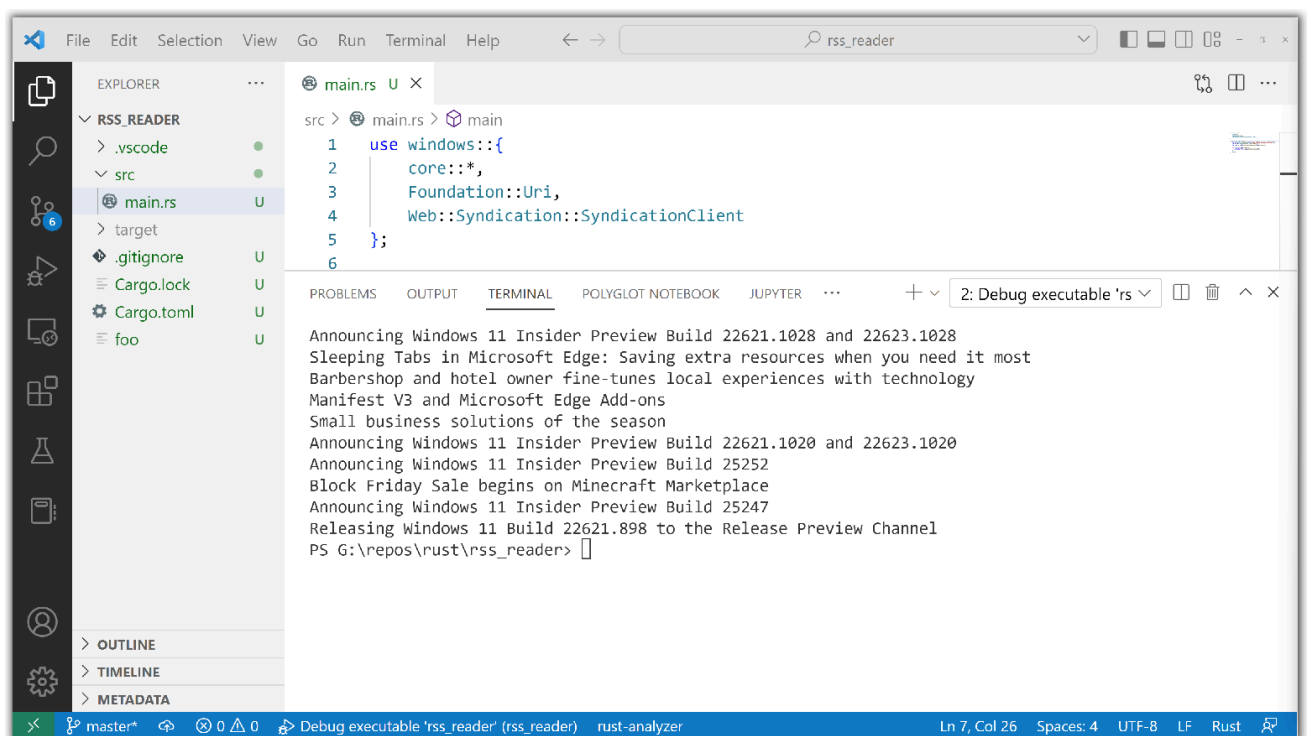
    Ok(())
}
```

10. 次に、[実行] をクリックしてビルドして実行できることを確認します (または **Ctrl + F5** キーを押します)。 予期しないメッセージが表示された場合は、[Hello, world! チュートリアル \(VS Code を使用した Rust\)](#) が正常に完了していることを確認してください。

また、テキストエディター内には、**デバッグ** コマンドと **実行** コマンドも埋め込まれています。または、`rss_reader` フォルダのコマンド プロンプトから「`cargo run`」と入力します。これにより、プログラムがビルドされて実行されます。



VS Code の [ターミナル] ペインの下方では、Cargo によって、windows クレートのダウンロードとコンパイルが正常に行われ、結果がキャッシュされて、それを使用して後続のビルドがより短い時間で完了したことを確認できます。次に、サンプルをビルドして実行し、ブログ投稿タイトルの一覧を表示します。



メッセージ ボックスの表示

Rust for Windows では、任意の Windows API (過去、現在、将来) を呼び出すことができます。このセクションでは、いくつかの Windows メッセージ ボックスを示します。

1. RSS プロジェクトの場合と同様に、コマンド プロンプトで Rust プロジェクトを含むフォルダーに `cd` します。
2. `message_box` という名前 の新しいプロジェクトを作成し、VS Code で開きます。

コンソール

```
> cargo new message_box
>     Created binary (application) `message_box` package
> cd message_box
> code .
```

3. VS Code で、`Cargo.toml` を開き、このプロジェクトの Windows 依存関係を追加します。

Rust

```
# message_box\Cargo.toml
...

[dependencies.windows]
version = "0.62.0"
features = [
    "Win32_Foundation",
    "Win32_UI_WindowsAndMessaging",
]
```

4. 次に、プロジェクトの `src/main.rs` ファイルを開き、新しい名前空間を使用して `use` 宣言を追加します (次に示すように)。最後に、[MessageBoxA 関数](#) と [MessageBoxW 関数](#) を呼び出すコード **を追加します**。Windows API ドキュメントは主に C/C++ を念頭に置いて記述されているため、`WINDOWS` クレートの Rust プロジェクション ([MessageBoxA \(Rust\)](#) と [MessageBoxW \(Rust\)](#)) のドキュメントと API ドキュメントを比較すると便利です。

Rust

```
// src/main.rs
use windows::{
    core::*,
    Win32::UI::WindowsAndMessaging::*
};
```

```
fn main() {
    unsafe {
        MessageBoxA(None, s!("Ansi"), s!("World"), MB_OK);
        MessageBoxW(None, w!("Wide"), w!("World"), MB_OK);
    }
}
```

ご覧のように、これらの Win32 API は `unsafe` ブロックで使用する必要があります ([安全でないブロック](#) を参照)。また、`s!` と `w!` のマクロにも注目してください。これらのマクロは、RustのUTF-8文字列リテラルから `LPCSTR` および `LPCWSTR` 引数を生成します。まるで `h!` マクロを用いて `HSTRING` を `rss_reader` で生成したのと同様に。Rust は UTF-8 文字列を使用したネイティブ Unicode であるため、ANSI (A サフィックス) API よりもワイド Unicode (W サフィックス) Windows API を使用することをお勧めします。これは、コードで英語以外のテキストを使用する場合に重要な場合があります。

今回ビルドして実行すると、Rust に 2 つの Windows メッセージ ボックスが表示されます。

関連項目

- [Windows 用 Rust と windows クレート](#)
- [ECMA-335](#)
- [? エラー処理を容易にする演算子](#)
- [安全でないブロック](#)

Windows アプリ開発のドキュメント

Windows アプリ SDKと WinUI 3 を使用してネイティブ Windows デスクトップ アプリを構築します。これは、Windows 10と 11 で C#、C++、XAML を使用した新しい開発に推奨されるプラットフォームです。



新着情報
Windows 開発
者向けの新機能



概念
Windows 開発
者プラットフォームの概要



チュートリアル
初めての WinUI
3 アプリをビルドする

Windows アプリを構築するためのパス

基礎から始め、優れたエクスペリエンスを設計し、アプリを構築し、配布用に準備します。



Essentials

Windows 開発者プラットフォームの概要

Windows アプリ開発の主要な概念

初めての WinUI 3 アプリの使用を開始する

開発環境を設定する



設計

Windows アプリの設計の概要

Windows 設計の原則

Windows アプリの設計ガイドライン

設計ツールとリソース



開発

Windows アプリ SDK 開発の概要

Windows アプリ SDK 機能と API

Windows アプリ SDK を使用して既存のアプリを最新化する

Windows 開発プラットフォームの概念

Windows アプリ SDK API リファレンス



パッケージ化と展開

[パッケージ化とデプロイの概要](#)

[Windows アプリ用の MSIX パッケージ化](#)

[アプリの配布パスを選択する](#)

[Windows アプリ SDK 展開オプション](#)

公開する

[Microsoft Store 公開の概要](#)

[Microsoft Store での公開を始める](#)

[Microsoft Store 開発者アカウントを開く](#)

[アプリを Microsoft Store に送信する](#)

Windows アプリに AI を追加する

[Windows AI 開発の概要](#)

[Windows AI Foundry ツールキットを試してみる](#)

[Copilot+ PC 向け Windows AI API](#)

どのプラットフォームを使用する必要がありますか？

シナリオに合ったフレームワークを選択します。新しいネイティブ Windows アプリには、Windows アプリ SDK を含む WinUI 3 をお勧めします。UWP はメンテナンス モードです。新しいプロジェクトでは WinUI 3 で Windows アプリ SDK を使用し、既存の UWP アプリを最新化するには移行ガイドを参照してください。

WinUI 3 (Windows アプリ SDK)

新しいネイティブ Windows デスクトップ アプリに推奨されます。C# または C++ と XAML を使用してビルドしま...

 [WinUI 3 と Windows アプリ SDK の概要](#)

 [初めての WinUI 3 アプリの使用を開始する](#)

 [Windows アプリ SDK のダウンロード](#)

WPF (.NET)

.NET デスクトップ アプリ用の成熟した XAML フレームワーク。Windows アプリ SDK 相互運用機能を使用して既存の...

 [WPF フレームワークの概要](#)

 [Windows アプリ SDK を使用して WPF アプリを最新化する](#)

Windows フォーム (.NET)

基幹業務アプリの迅速な UI 開発。 .NET 8 以降でもサポートされています。Windows アプリ SDK 相互運用機能を使用し...

 [Windows フォームの概要](#)

 [WinForms アプリに Windows アプリ SDK 機能を追加する](#)

.NET MAUI

1 つの C# と XAML コードベースから Windows、macOS、iOS、Android を対象とするクロスプラットフォームフレ...

C++ と Win32

高パフォーマンスのアプリ、ゲーム、システムユーティリティ用の直接 OS アクセス。Win32、COM、または...

 [Win32 および COM API リファレンス](#)

プログレッシブ Web Apps (PWA)

タスクバー、通知、オフラインサポートなど、Windows 統合された Web ファーストアプリ。HTML、CSS、JavaScript...

SDK リファレンスとダウンロード

リファレンス ドキュメント、リリース ノート、ダウンロード用の SDK ランディング ページに直接移動します。

[Windows アプリ SDK](#)[WinUI 3](#)[Windows AI ファウンダリー](#)[.NET デスクトップ開発](#)

作業の開始要件

Windows アプリ SDKで構築を開始するために必要なもの。バージョン 1809 (ビルド 17763) 以降 Windows 10ターゲット。

Windows アプリ SDK (最新の安定版)

現在の安定版リリースをダウンロードします。WinUI 3、アプリ ライフサイクル API、ウィンドウ化、デプロイ ツールが含まれています。ダウンロード ページで最新バージョン番号を確認します。

WinUI 3 (Windows アプリ SDKに含まれています)

Windows デスクトップ アプリ用のネイティブ UI フレームワーク。Windows アプリ SDKの一部として出荷されます。別のインストールは必要ありません。

Windows アプリ SDK ワークロードを含む Visual Studio 2026

".NET デスクトップ開発" ワークロードとWindows アプリ SDK拡張機能を使用して、Visual Studio 2026 をインストールします。または、C# 開発キットで VS Code を使用します。

サポートされている言語とランタイム

C# .NET 8 以降、C++ (Win32 および C++/WinRT)、Rust (windows-rs クレート経由)。XAML は、WinUI 3 および WPF の UI マークアップに使用されます。

ツールとリソース

GitHub 上の Windows アプリのコード サンプル

Windows アプリ SDK機能、WinUI 3 コントロール、プラットフォーム API を示すサンプル アプリとコード スニペットについて説明します。

Windows開発者環境のセットアップ

Visual Studio、Windows アプリ SDK、Dev Home、Linux 用 Windows サブシステムを使用して開発環境を設定します。

Windows開発者ブログ

Windows開発者チームからのお知らせ、リリースの更新、詳細を確認します。

UWP から Windows アプリ SDK に移行する

UWP はメンテナンス モードです。既存の UWP アプリをWindows アプリ SDKに移行して、現在の API と継続的なプラットフォームへの投資にアクセスします。

[@WindowsDocs](#) - [OneDevMinute \(YouTube\)](#) - [Windows 開発者向けサポート](#) - [Stack Overflow](#)

開発環境を Mac から Windows に変更するためのガイド

Mac と Windows（または WSL/Linux）の開発環境を移行する際に役立つ、次のヒントとコントロールの同等機能を紹介します。

アプリ開発の場合、Xcode と最も近いものは [Visual Studio](#) です。クロスプラットフォームのソースコード編集 (および膨大な数のプラグイン) では、[Visual Studio Code](#) が最も一般的な選択肢です。

キーボード ショートカット

💡 ヒント

[PowerToys キーボード マネージャー](#)を使用してWindowsショートカットを Mac で使用するショートカットにマップします。

 テーブルを展開する

操作	Mac	Windows
コピーする	Command + C	Ctrl + C
切る	Command + X	Ctrl + X
貼り付け	Command + V	Ctrl + V
元に戻す	Cmd+Z	Ctrl + Z
セーブ	コマンド+S	Ctrl + S
開け	Command + O	Ctrl + O
コンピューターをロックする	Command + Control + Q	WindowsKey + L
デスクトップを表示する	Command + F3	WindowsKey + D
ファイル ブラウザーを開く	Command + N	WindowsKey + E
ウィンドウを最小化する	Command + M	WindowsKey + M
Search	Command + Space	Windowsキー

操作	Mac	Windows
アクティブ ウィンドウを閉じる	Command + W	Control + W
現在のタスクを切り替える	Command + Tab	Alt + Tab
ウィンドウを全画面表示に最大化する	Control + Command + F	Windowsキー + 上矢印
スクリーンショットを保存	Command + Shift + 3	WindowsKey + PrtScn (ファイルに保存) または PrtScn (クリップボード)
リージョン/ウィンドウの保存 (スクリーンショット)	Command + Shift + 4	WindowsKey + Shift + S (切り取りツールが開きます)
アイテムの情報またはプロパティを表示する	Command + I	Alt + Enter
すべての項目を選択する	Command + A	Ctrl + A
一覧で複数の項目を選択する (連続しない)	コマンドを実行し、各項目をクリックします。	コントロールをクリックし、各項目をクリックします。
特殊文字を入力する	オプション+文字キー	Alt + 文字キー

トラックパッドのショートカット

ⓘ Note

これらのショートカットの一部には、Surfaceデバイス上のトラックパッドやその他のサードパーティ製ラップトップなど、"Precision Trackpad" が必要です。

両方のプラットフォームでトラックパッド オプションを構成できます。

 テーブルを展開する

操作	Mac	Windows
スクロール	2本の指の垂直方向のスイープ	2本の指の垂直方向のスイープ
ズーム	2本指でズームイン/アウト	2本指でズームイン/アウト
ビュー間で前後にスワイプする	2本の指を横にスワイプする	2本の指を横にスワイプする

操作	Mac	Windows
仮想ワークスペースの切り替え	4本の指を横にスワイプ	4本の指を横にスワイプ
現在開いているアプリを表示する	4本の指で上方向にスワイプ	3本指で上方向にスワイプ
アプリを切り替える	なし	ゆっくりと3本の指で横にスワイプする
デスクトップに移動する	4本の指を広げる	3本指で下方向にスワイプ
アクション センターを開く	右から2本の指でスライドする	3本指でタップ
追加情報を開く	3本指でタップ	なし
スタート パッドを表示する/アプリを起動する	4本の指で画面をつまむ	4本の指でタップする

コマンドライン シェルとターミナル

Windowsでは、いくつかのコマンドライン シェルとターミナルがサポートされています。これらのツールは、Mac の BASH シェルやターミナル エミュレーター アプリ (Terminal や iTerm など) とは少し異なる動作をする場合があります。

Windowsのシェル

Windowsには、次の2つの主要なコマンドライン シェルがあります。

- 1. **PowerShell** - PowerShell は、クロスプラットフォームのタスク自動化と構成管理フレームワークです。これは、.NET上に構築されたコマンドライン シェルとスクリプト言語で構成されます。PowerShell を使用すると、管理者、開発者、およびパワー ユーザーは、複雑なプロセスや環境やオペレーティング システムのさまざまな側面を管理するタスクをすばやく制御および自動化できます。PowerShell は [完全にオープンソース](#) であり、クロスプラットフォームであるため、[Mac と Linux でも使用できます](#)。

Mac および Linux BASH シェル ユーザー: PowerShell では、既に使い慣れている多くのコマンドエイリアスもサポートされています。例えば次が挙げられます。

- 現在のディレクトリの内容を次の内容で一覧表示します。 `ls`
- 次の方法でファイルを移動します。 `mv`
- 次を使用して新しいディレクトリに移動します。 `cd <path>`

PowerShell と BASH では、一部のコマンドと引数が異なります。詳細については、「PowerShell で [get-help](#) 」と入力するか、ドキュメントの [互換性エイリアス](#) を確認してください。

管理者として PowerShell を実行するには、Windows のスタートメニューに「PowerShell」と入力し、**管理者として実行**を選択します。

2. **Windows コマンド ライン (Cmd)** - Windows は従来のコマンド プロンプト (およびコンソール - 以下を参照) を提供し、現在および従来の MS-DOS 互換コマンドおよびバッチ ファイルとの互換性を提供します。Cmd は、既存または古いバッチ ファイルまたはコマンドライン操作を実行する場合に便利です。ただし、Cmd は現在メンテナンス中であり、今後は改善や新機能を受け取らないので、PowerShell を学習して使用してください。

ネイティブ Unix スタイルのコマンド

Mac または Linux のバックグラウンドから来ている場合は、WSL を必要とせずに、Windows でネイティブに使用できる使い慣れたコマンドが多数あります。

- **Coreutils for Windows:** GNU Coreutils のネイティブ移植版で、`ls`、`cat`、`cp`、`mv`、`grep`、`head`、`tail`、`wc`、`sort`、`uniq` などのコマンドを Windows で利用できるようにします。WinGet (`winget install utils.coreutils`) または GitHub からインストールします。PowerShell では、組み込みのエイリアスが優先される場合があります。coreutils バージョンを直接呼び出すには、`ls.exe`、`cp.exe` などを使用します。コマンドの [完全な一覧](#) を参照してください。
- **sudo:** macOS の場合と同様に、管理者特権でコマンドを実行します。Windows 11 24H2 以降で利用できます — **設定 > 開発者向け** で有効にできます。
- **curl** と **tar:** ファイルのダウンロードとアーカイブの操作にネイティブに使用できます。
- **編集:** コマンドラインから簡単に編集するための軽量のターミナルベースのテキストエディタ。2025 年 9 月の更新プログラム以降の Windows 11 に含まれるか、WinGet (`winget install Microsoft.Edit`) を使用してインストールします。

これらのツールは、基本的なタスクのために WSL に切り替えることなく、Windows で使い慣れた Unix ワークフローを直接使用できることを意味します。

Linux シェル

Linux 用 Windows サブシステム (WSL) をインストールして、Windows 内での Linux シェルの実行をサポートできるようになりました。つまり、**bash** を実行し、任意の特定の Linux ディストリビューションを選択して、Windows 内に統合することができます。WSL を使用すると、Mac ユーザーにとって最も使い慣れた環境が提供されます。たとえば、`ls` を使用して、従来の

Windows Cmd Shell と同様に、`dir` ではなく、現在のディレクトリ内のファイルを一覧表示します。WSL のインストールと使用については、[Linux 用 Windows サブシステム インストール ガイド](#)を参照してください。WSL を使用してWindowsにインストールできる Linux ディストリビューションは次のとおりです。

1. [Ubuntu 24.04 LTS](#) ↗
2. [Ubuntu 22.04 LTS](#) ↗
3. [Kali Linux](#) ↗
4. [Debian GNU/Linux](#) ↗
5. [openSUSE Leap 15.6](#) ↗
6. [SUSE Linux Enterprise Server 15 SP6](#) ↗

例えばいくつか挙げてみると。詳細については、[WSL install docs](#) を参照し、[Microsoft Store](#) ↗ から直接インストールします。

Windowsターミナル

多くのサードパーティ製品に加えて、Microsoftには2つのターミナルが用意されています。これらのターミナルは、コマンドラインシェルとアプリケーションへのアクセスを提供するGUIアプリケーションです。

1. **Windows ターミナル**: Windows ターミナルは、非常に高いパフォーマンス、待ち時間の短いコマンドライン ユーザー エクスペリエンス、複数のタブ、分割ウィンドウ ウィンドウ、カスタム テーマとスタイル、さまざまなシェルまたはコマンドラインアプリの複数の "プロファイル"、およびかなりの機会を提供する、高度に構成可能な最新のコマンドラインターミナルアプリケーションです。コマンドライン ユーザー エクスペリエンスの多くの側面を構成およびカスタマイズできます。Windows ターミナルは、Windows 11にプレインストールされています。Windows 10では、[Microsoft Store](#) ↗ からインストールできます。

Windows ターミナルを使用すると、PowerShell、WSL シェル (Ubuntu や Debian など)、従来の Windows コマンド プロンプト、その他のコマンドライン アプリ (SSH、Azure CLI、Git Bash など) に接続されたタブを開くことができます。

2. **コンソール**: macOS と Linux では、通常、ユーザーは優先ターミナルアプリケーションを起動し、ユーザーの既定のシェル (BASH など) を作成して接続します。

ただし、一風変わった歴史のため、Windowsユーザーは従来シェルを起動し、Windows自動的に GUI コンソール アプリを起動して接続します。

シェルを直接起動し、レガシ Windows コンソールを使用することはできますが、Windows ターミナルをインストールして使用して、最も高速で生産性の高いコマンド ライン エクスペリエンスを体験することを強くお勧めします。

アプリとユーティリティ

 テーブルを展開する

アプリ	macOS	Windows
設定と基本設定	システム環境設定	設定
タスク マネージャー	アクティビティ モニター	タスク マネージャ
ディスクの書式設定	ディスク ユーティリティ	ディスクの管理
テキスト編集	テキストエディット	メモ帳または 編集 (ターミナル)
イベントの表示	コンソール	イベント ビューアー
ファイル/アプリを検索する	Command + Space	Windows キー
パッケージ マネージャー	Homebrew	WinGet
開発者向けストレージ	なし	Dev ドライブ
Unix CLI ツール	組み込みコマンド (ls、cat、grep など)	Coreutils + Sudo
ウィンドウの管理	ミッション コントロール	スナップ レイアウト (WindowsKey + Z)
生産性ユーティリティ	なし	PowerToys

Last updated on 2026/06/05