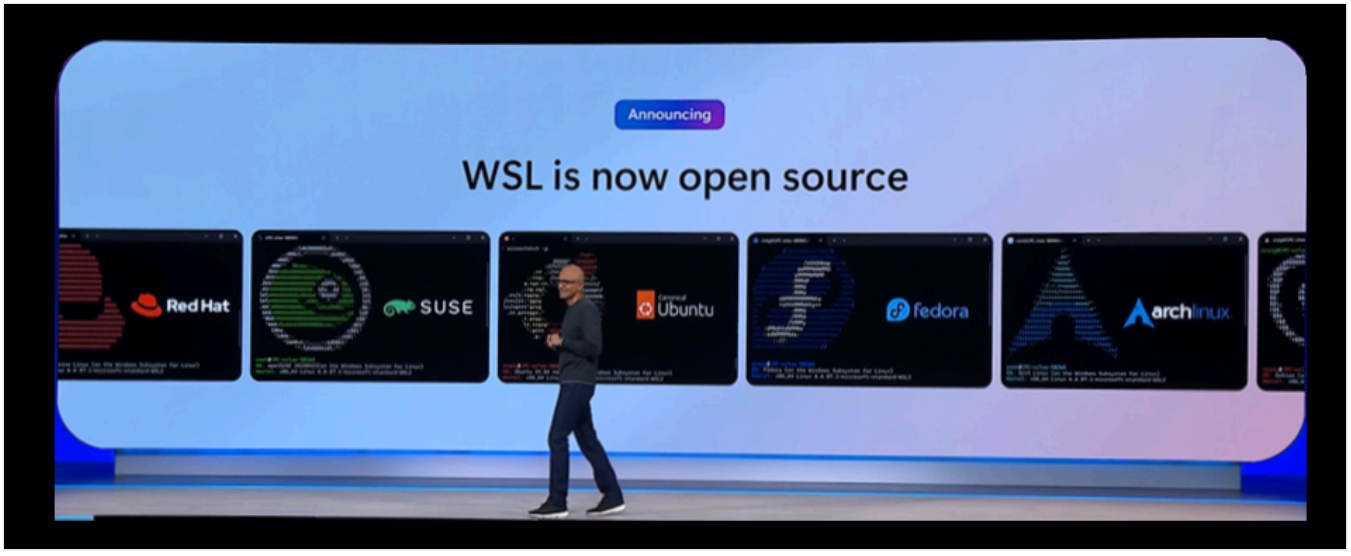
Windows Subsystem for Linux 文档

适用于 Linux 的 Windows 子系统（WSL）允许开发人员直接在 Windows 上运行 GNU/Linux 环境（包括大多数命令行工具、实用工具和应用程序），无需传统虚拟机或双启动设置的开销。

安装 WSL

了解详细信息

- 什么是适用于 Linux 的 Windows 子系统（WSL）？
- 适用于 Linux 的 Windows 子系统现已开放源代码
- WSL 2 的新增功能有哪些?
- 比较 WSL 1 和 WSL 2
- 常见问题解答



开始

- 安装 WSL
- 在 Windows Server 上安装 Linux
- 手动安装步骤
- 设置 WSL 开发环境的最佳做法

通过加入 Windows 预览体验计划试用 WSL 预览版功能

若要尝试 WSL 的最新功能或更新，请加入 Windows 预览体验计划。 加入 Windows 预览体验成员后，可以选择希望从 Windows 设置菜单中接收预览版的频道。 可以选择：

- 开发通道：最新的更新，但稳定性较低。
- Beta 通道：非常适合早期采用者，比开发通道更可靠。
- 发布预览频道：在 Windows 下一个版本正式提供给公众之前，预览修补程序和主要功能。

团队博客

- 包含视频和博客集合的概述文章
- Command-Line 博客 （活动）
- 适用于 Linux 的 Windows 子系统博客 （历史）

提供反馈

- GitHub 问题跟踪器：WSL
- GitHub 问题跟踪器：WSL 文档

相关视频

WSL 基础知识

1. 什么是适用于 Linux 的 Windows 子系统 (WSL)? |一个开发问题 （0：40）
2. 我是 Windows 开发人员。 为何应使用 WSL? | 一个开发问题 （0：58）
3. 我是 Linux 开发人员。 为何应使用 WSL? |一个开发问题 （1：04）
4. 什么是 Linux? |一个开发问题 （1：31）
5. 什么是 Linux 发行版? |一个开发问题 （1：04）
6. WSL 与虚拟机或双启动有何不同? |一个开发问题
7. 为什么创建了适用于 Linux 的 Windows 子系统? |一个开发问题 （1：14）
8. 如何在 WSL 中访问计算机上的文件? |一个开发问题 （1：41）
9. WSL 如何与 Windows 集成? |一个开发问题 （1：34）
10. 如何将 WSL 发行版配置为在终端的主目录中启动? |一个开发问题 （0：47）
11. 是否可以使用 WSL 编写脚本? |一个开发问题 （1：04）
12. 为什么我想在 Windows 上使用 Linux 工具? |一个开发问题 （1：20）
13. 在 WSL 中，是否可以使用除 Microsoft 应用商店中的发行版以外的发行版? |一个开发问题 （1：03）

WSL 演示

1. WSL2：在适用于 Linux 的 Windows 子系统上更快地编写代码！ |制表符与空格 （13：42）
2. WSL：运行 Linux GUI 应用 |制表符与空格 （17：16）

3. WSL 2：连接 USB 设备 | 制表符与空格（10:08）
4. 使用 WSL 2 的 GPU 加速机器学习 | 制表符与空格（16：28）
5. Visual Studio Code：使用 SSH、VM 和 WSL 进行远程开发 |制表符与空格 （29：33）
6. Windows 开发人员工具更新：WSL、终端、包管理器等 |制表符与空格 （20：46）
7. 使用 WSL 生成Node.JS应用 |突出显示（3：15）
8. WSL 2 中的新内存回收功能 |演示 （6：01）
9. Windows 上的 Web 开发 （2019 年） |演示（10：39）

WSL 深度探索

1. Windows 11 上的 WSL - Craig Loewen 和 Scott Hanselman 的演示 | Windows 周三 （35：48）
2. WSL 和 Linux 发行版 – 海登·巴恩斯和凯拉·辛纳蒙 |Windows 星期三 （37：00）
3. 使用 Oh My Posh 和 WSL Linux 发行版自定义终端 | Windows 星期三 （33：14）
4. Web 开发者 Sarah Tamsin 和 Craig Loewen 讨论关于 Web 开发、内容创建和 WSL | 开发视角 (12:22)
5. WSL 如何从 Windows 访问 Linux 文件 |深入探讨 （24：59）
6. 适用于 Linux 的 Windows 子系统体系结构：深入探讨 |内部版本 2019 （58：10）

Last updated on 2025/05/21

什么是适用于 Linux 的 Windows 子系统?

适用于 Linux 的 Windows 子系统（WSL）是 Windows 的一项功能，可用于在 Windows 计算机上运行 Linux 环境，而无需单独的虚拟机或双重启动。 WSL 旨在为想要同时使用 Windows 和 Linux 的开发人员提供无缝高效的体验。

- 使用 WSL 安装和运行各种 Linux 分发版，例如 Ubuntu、Debian、Kali 等。 安装 Linux 分发版 并从 Microsoft 应用商店接收自动更新， 导入 Microsoft 应用商店中不可用的 Linux 分发版，或 生成自己的自定义 Linux 分发版。
- 将文件存储在独立的 Linux 文件系统中，特定于已安装的分发版。
- 运行命令行工具，例如 BASH。
- 运行常见的 BASH 命令行工具，例如`grep`，`sed` `awk`或其他 ELF-64 二进制文件。
- 运行 Bash 脚本和 GNU/Linux 命令行应用程序，包括：
  - 工具：vim、emacs、tmux
  - 语言： NodeJS、JavaScript、 Python、Ruby、C/C++、C# & F#、Rust、Go 等。
  - 服务：SSHD、 MySQL、Apache、lighttpd、 MongoDB、 PostgreSQL。
- 使用自己的 GNU/Linux 分发包管理器安装其他软件。
- 使用类似 Unix 的命令行 shell 调用 Windows 应用程序。
- 在 Windows 上调用 GNU/Linux 应用程序。
- 运行直接集成到 Windows 桌面的 GNU/Linux 图形应用程序
- 使用设备 GPU 加速 Linux 上运行的机器学习工作负载。

WSL 是一个开放源代码工具，提供可供下载和贡献的源代码：

- 详细了解 WSL 开源组件
- WSL 开源文档网站： wsl.dev
- GitHub 上的 WSL 存储库： github.com/Microsoft/wsl

开始使用 WSL：

安装 WSL

https://www.youtube-nocookie.com/embed/48k317kOxqg

什么是 WSL 2?

安装 Linux 分发版时，WSL 2 是默认发行版类型。 WSL 2 使用虚拟化技术在轻型实用工具虚拟机（VM）内运行 Linux 内核。 Linux 分发版作为 WSL 2 托管 VM 内的独立容器运行。 通过 WSL 2 运行的 Linux 分发版将共享同一网络命名空间、设备树（而非`/dev/pts`）、CPU/内核/内存/交换、`/init`二进制文件，但有自己的 PID 命名空间、装载命名空间、用户命名空间、Cgroup 命名空间和`init`进程。

WSL 2 **提高了文件系统性能**，并且与 WSL 1 体系结构相比增加了 **完整的系统调用兼容性**。 详细了解 WSL 1 和 WSL 2 的比较方式。

可以使用 WSL 1 或 WSL 2 体系结构运行单个 Linux 分发版。 可以随时升级或降级每个分发，并且可以并行运行 WSL 1 和 WSL 2 分发版。 请参阅 “设置 WSL 版本”命令。

https://www.youtube-nocookie.com/embed/MrZolfGm8Zk

Microsoft喜欢 Linux

在Microsoft了解有关 Linux 资源的详细信息，包括在 Linux 上运行的Microsoft工具、Linux 培训课程、适用于 Linux 的云解决方案体系结构，以及 Microsoft + Linux 新闻、事件和合作关系。 **Microsoft喜欢 Linux！**

Last updated on 2025/05/21

比较 WSL 版本

详细了解不同的 WSL 版本，包括为什么 WSL 2 现在是默认版本，以及可能需要将已安装的 Linux 分发版切换到早期 WSL 1 体系结构的特定方案或异常。

比较 WSL 1 和 WSL 2

本指南将比较 WSL 1 和 WSL 2，包括 选择使用 WSL 1 而非 WSL 2 的例外情况。 WSL 1 和 WSL 2 之间的主要区别是使用托管 VM 内的实际 Linux 内核、支持完整的系统调用兼容性，以及跨 Linux 和 Windows作系统的性能。 WSL 2 是安装 Linux 分发版时的当前默认版本，使用最新的虚拟化技术在轻型实用工具虚拟机（VM）内运行 Linux 内核。 WSL2 以托管 VM 中的独立容器的形式运行 Linux 分发版。 如果分发版当前正在运行 WSL 1，并且想要更新到 WSL 2，请参阅 从 WSL 1 更新到 WSL 2。

比较功能

展开表

功能 / 特点	WSL 1	WSL 2
Windows 与 Linux 之间的集成	✅	✅
快速启动时间	✅	✅
与传统虚拟机相比，资源占用很小	✅	✅
使用 VMware 和 VirtualBox 的当前版本运行	✅	❌
托管 VM	❌	✅
完整 Linux 内核	❌	✅
完整的系统调用兼容性	❌	✅
跨 OS 文件系统的性能	✅	❌
systemd 支持	❌	✅
IPv6 支持	✅	✅

如上面的比较表所示，WSL 2 体系结构在多种方面优于 WSL 1，但在不同操作系统的文件系统的性能方面除外，这可以通过将项目文件存储在与运行项目工具相同的操作系统上来解决。

WSL 2 仅在 Windows 11 或 Windows 10 版本 1903 版本 18362 或更高版本中可用。 通过选择 **Windows 徽标键** + **R** 来检查 Windows 版本，键入 **winver**，选择“ **确定**”。 （或在 Windows 命

令提示符中输入 `ver` 命令）。 可能需要 更新到最新的 Windows 版本。 对于低于 14393 的版本，WSL 根本不支持。

有关最新的 WSL 2 更新的详细信息，请参阅 Windows 命令行博客，包括 Systemd 支持现已在 WSL 和 WSL 2023 年 9 月更新 中提供，了解有关 IPv6 支持的详细信息。

> **注意**
>
> WSL 2 将与 VMware 15.5.5+ 配合使用，尽管 VirtualBox 6+ 指出存在 WSL 支持，但仍存在重大挑战，因此不支持它。 在我们的**常见问题解答中**了解详细信息。

WSL 2 中的新增功能

WSL 2 是对基础体系结构的重大改革，它使用虚拟化技术和 Linux 内核来启用新功能。 此更新的主要目标是提高文件系统性能并添加完整的系统调用兼容性。

- WSL 2 系统要求
- 将 Linux 分发版本从 WSL 1 设置为 WSL 2
- 有关 WSL 2 的常见问题

WSL 2 体系结构

传统的 VM 体验启动速度可能很慢，是隔离的，会消耗大量资源，并且需要时间来管理它。 WSL 2 没有这些属性。

WSL 2 提供 WSL 1 的优势，包括 Windows 和 Linux 之间的无缝集成、快速启动时间、少量的资源占用，并且不需要 VM 配置或管理。 虽然 WSL 2 确实使用 VM，但它在后台进行管理和运行，使你拥有与 WSL 1 相同的用户体验。

完整 Linux 内核

WSL 2 中的 Linux 内核由 Microsoft 基于最新稳定分支构建，该分支的源代码可从 kernel.org 获取。此内核已专门针对 WSL 2 进行优化，以便在 Windows 上提供卓越的 Linux 体验，特别是在尺寸和性能方面进行了优化。 内核将由 Windows 更新提供服务，这意味着你将获得最新的安全修补程序和内核改进，而无需自行管理它。

WSL 2 Linux 内核是开放源代码。 若要了解更多内容，请查看由构建该内核的团队编写的博客文章《Windows 发布 Linux 内核》。

在 适用于 Linux 的 Windows 子系统内核的发行说明中了解详细信息。

提高了文件 IO 性能

WSL 2 的文件密集型操作（例如`git clone`、`npm install`、`apt update`、`apt upgrade`等）都明显更快。

实际速度提高取决于正在运行的应用以及它与文件系统的交互方式。 WSL 2 的初始版本在解压缩压缩的 tarball 时运行速度比 WSL 1 快最多达到 20 倍，在使用`git clone`、`npm install`和`cmake`处理各种项目时速度提高约为 2-5 倍。

完整的系统调用兼容性

Linux 二进制文件使用系统调用来执行访问文件、请求内存、创建进程等功能。 而 WSL 1 使用了由 WSL 团队生成的转换层，但 WSL 2 包含其自己的 Linux 内核，具有完整的系统调用兼容性。 优点包括：

- 可以在 WSL 内部运行的一组全新的应用，例如 **Docker** 等。
- Linux 内核的任何更新都立即可供使用（无需等待 WSL 团队实现更新并添加更改）。

使用 WSL 1 而不是 WSL 2 的异常

建议使用 WSL 2，因为它提供更快的性能和 100% 系统调用兼容性。 但是，在某些情况下，你可能更喜欢使用 WSL 1。 如果以下情况，请考虑使用 WSL 1：

- 项目文件必须存储在 Windows 文件系统中。 WSL 1 可更快地访问从 Windows 装载的文件。
  - 如果你将使用 WSL Linux 分发版访问 Windows 文件系统上的项目文件，并且这些文件不能存储在 Linux 文件系统上，则通过使用 WSL 1，你将在 OS 文件系统中实现更快的性能。
- 需要在同一文件中使用 Windows 和 Linux 工具进行交叉编译的项目。
  - 在 WSL 1 中，跨 Windows 和 Linux作系统的文件性能比 WSL 2 更快，因此，如果使用 Windows 应用程序访问 Linux 文件，则目前 WSL 1 的性能将更快。
- 项目需要访问串行端口或 USB 设备。 *然而* USB 设备支持现已通过`USBIPD-WIN`项目用于 WSL 2。 有关设置步骤，请参阅 连接 USB 设备 。
- WSL 2 不包括对访问串行端口的支持。 在 常见问题解答 或 WSL GitHub 存储库中详细了解串行支持问题 。
- 内存要求严格
  - WSL 2 的内存使用量在使用它时会增大和收缩。 当进程释放内存时，会自动返回到 Windows。 但是，截至现在，WSL 2 在关闭 WSL 实例之前，尚未将内存中的缓存页释放回 Windows。 如果长时间运行 WSL 会话或访问大量文件，此缓存可能会占用

Windows 上的内存。 我们正在跟踪改进 WSL GitHub 存储库问题 4166 上的此体验的工作。

- 对于使用 VirtualBox 的用户，请务必使用最新版本的 VirtualBox 和 WSL 2。 请参阅 相关的常见问题解答。
- 如果您依赖 Linux 发行版在与主机相同的网络中获取 IP 地址，那么可能需要设置一个解决方案才能运行 WSL 2。 WSL 2 作为 Hyper-V 虚拟机运行。 这是 WSL 1 中使用的桥接网络适配器的更改，这意味着 WSL 2 对其虚拟网络使用网络地址转换（NAT）服务，而不是将其桥接到主机网络接口卡（NIC），从而导致在重启时更改的唯一 IP 地址。 若要详细了解将 WSL 2 服务的 TCP 端口转发到主机 OS 的问题和解决方法，请参阅 WSL GitHub 存储库问题 4150、NIC 桥模式（TCP 解决方法）。

> **注意**
>
> 请考虑尝试使用 VS Code **远程 WSL 扩展** 将项目文件存储在 Linux 文件系统上，使用 Linux 命令行工具，还可以在 Windows 上使用 VS Code 在 Internet 浏览器中创作、编辑、调试或运行项目，而无需与在 Linux 和 Windows 文件系统中工作相关的任何性能降低。 **了解详细信息**。

Microsoft 商店中的 WSL

WSL 已将更新功能从 Windows OS 映像中移出，并包含在微软应用商店提供的包中。 这意味着，一旦更新和服务可用，您将可以立即获得更新和服务，而无需等待您的Windows操作系统更新。

WSL 最初作为需要启用的可选组件包含在 Windows作系统中，以便安装 Linux 分发版。 应用商店中的 WSL 具有相同的用户体验，并且是相同的产品，但接收更新和服务作为应用商店包，而不是作为整个 OS 更新。 从 Windows 版本 19044 或更高版本开始，运行 `wsl.exe --install` 此命令将从 Microsoft 应用商店安装 WSL 服务更新。 （请参阅宣布此更新的博客文章）。 如果已在使用 WSL，则可以更新以确保通过运行 `wsl.exe --update`，从应用商店接收最新的 WSL 功能和服务。

> **注意**
>
> 如果组织中无法访问Microsoft应用商店，则仍可以通过追加 `--web-download` 到 `--update` 命令来利用此 WSL 版本，例如 `wsl --update --web-download`。 每次有新版本推出时，您需要手动使用此方法更新 WSL。

Last updated on 2025/06/10

WSL 的基本命令

下面的 WSL 命令以 PowerShell 或 Windows 命令提示符支持的格式列出。 要从 Bash/Linux 发行版的命令行运行这些命令，必须将 `wsl` 替换为 `wsl.exe`。 完整的命令列表，请运行 `wsl --help`。 如果尚未这样做，我们建议 更新到从 Microsoft Store 安装的 WSL 版本 ，以便在可用后立即接收 WSL 更新。 （了解有关通过 Microsoft Store 安装 WSL 的详细信息。

安装

PowerShell

```
wsl --install
```

安装 WSL 和 Linux 的默认 Ubuntu 分发版。 了解详细信息。 还可以使用此命令通过运行 `wsl --install <Distribution Name>` 来安装其他 Linux 分发版。 对于有效的发行版名称列表，请运行 `wsl --list --online`。

选项包括：

- `--distribution`：指定要安装的 Linux 分发版。 可以通过运行 `wsl --list --online` 来查找可用的分发版。
- `--no-launch`：安装 Linux 分发版，但不自动启动它。
- `--web-download`：从联机源安装，而不是使用 Microsoft Store。
- `--location`：指定要将 WSL 分发版安装到哪个文件夹。

未安装 WSL 时的选项包括：

- `--inbox`：使用 Windows 组件而不是使用 Microsoft 应用商店安装 WSL。 *（WSL 更新将通过 Windows 更新接收，而不是通过应用商店按可用方式推送）*。
- `--enable-wsl1`：在安装 Microsoft Store 版本的 WSL 时，启用 "适用于 Linux 的 Windows 子系统" 可选组件，从而启用 WSL 1。
- `--no-distribution`：安装 WSL 时不要安装分发版。

> **注意**
>
> 如果在 Windows 10 或更早版本上运行 WSL，则可能需要将 `-d` 标志包含在命令中 `--install` 以指定分发版： `wsl --install -d <distribution name>`

列出可用的 Linux 分发版

PowerShell

```
wsl --list --online
```

请参阅通过在线商店提供的 Linux 分发版列表。 此命令也可以输入为： `wsl -l -o`。

列出已安装的 Linux 分发版

PowerShell

```
wsl --list --verbose
```

查看 Windows 计算机上安装的 Linux 分发版的列表，包括状态（分发正在运行还是已停止），以及运行分发版的 WSL 版本（WSL 1 或 WSL 2）。 比较 WSL 1 和 WSL 2。 此命令也可以输入为： `wsl -l -v`。 可用于列表命令的其他选项包括： `--all` 列出所有通讯组、 `--running` 仅列出当前正在运行的分发版或 `--quiet` 仅显示分发名称。

将 WSL 版本设置为 1 或 2

PowerShell

```
wsl --set-version <distribution name> <versionNumber>
```

若要指定 Linux 发行版运行的 WSL 版本（1 或 2），请将 `<distribution name>` 替换为发行版的名称，并将 `<versionNumber>` 替换为 1 或 2。 比较 WSL 1 和 WSL 2。 WSL 2 仅在 Windows 11 或 Windows 10 版本 1903 版本 18362 或更高版本中可用。

> **⚠ 警告**
>
> 在 WSL 1 和 WSL 2 之间切换可能非常耗时，并且由于两种体系结构之间的差异而导致失败。 对于包含大型项目的分发版，建议在尝试转换之前备份文件。

设置默认 WSL 版本

PowerShell

```
wsl --set-default-version <Version>
```

若要设置 WSL 1 或 WSL 2 的默认版本，请将 `<Version>` 替换为数字 1 或 2。 例如， `wsl --set-default-version 2`。 该数字表示默认用于新 Linux 发行版安装的 WSL 版本。 比较 WSL 1 和

WSL 2。 WSL 2 仅在 Windows 11 或 Windows 10 版本 1903 版本 18362 或更高版本中可用。

设置默认 Linux 分发版

```
PowerShell
wsl --set-default <Distribution Name>
```

若要设置 WSL 命令将用于运行的默认 Linux 发行版，请将 `<Distribution Name>` 替换为您首选的 Linux 发行版名称。

在用户主目录中启动 WSL

```
PowerShell
wsl ~
```

`~`可与 wsl 一起使用，以在用户的主目录中启动。 若要从 WSL 命令提示符内从任何目录跳转回主页，可以使用以下命令： `cd ~`

从 PowerShell 或 CMD 运行特定的 Linux 分发版

```
PowerShell
wsl --distribution <Distribution Name> --user <User Name>
```

若要使用特定用户运行特定 Linux 发行版，请将 `<Distribution Name>` 替换为您首选的 Linux 发行版的名称（即 Debian），并将 `<User Name>` 替换为现有用户的名称（即 root）。 如果 WSL 分发中不存在用户，将收到错误。 若要打印当前用户名，请使用命令 `whoami`。

更新 WSL

```
PowerShell
wsl --update
```

将 WSL 版本更新为最新版本。 选项包括：

- `--web-download`：从 GitHub 下载最新更新，而不是 Microsoft 商店。

检查 WSL 状态

PowerShell

```
wsl --status
```

请参阅有关 WSL 配置的常规信息，例如默认分发类型、默认分发和内核版本。

检查 WSL 版本

PowerShell

```
wsl --version
```

检查有关 WSL 及其组件的版本信息。

Help 命令

PowerShell

```
wsl --help
```

请参阅 WSL 可用的选项和命令列表。

以特定用户身份运行

PowerShell

```
wsl --user <Username>
```

为以指定用户身份运行 WSL，请将 `<Username>` 替换为 WSL 分发中存在的用户的名称。

更改分发版的默认用户

PowerShell

```
<DistributionName> config --default-user <Username>
```

更改分发版登录的默认用户。 用户必须已存在于分发中，才能成为默认用户。

例如： `ubuntu config --default-user johndoe` 将 Ubuntu 分发的默认用户更改为“johndoe”用户。

> ⓘ **注意**
>
> 如果在找出分发名称时遇到问题，请使用命令 `wsl -l`。

> ⚠ **警告**
>
> 此命令不适用于导入的分发版，因为这些分发版没有可执行启动器。 可以改为使用 `/etc/wsl.conf` 文件更改导入分发的默认用户。 请参阅 **“高级设置配置”** 文档中的“自动装载”选项。

关机

PowerShell

```
wsl --shutdown
```

立即终止所有正在运行的分发版和 WSL 2 轻型实用工具虚拟机。 在需要重启 WSL 2 虚拟机环境的实例中，可能需要此命令，例如 更改内存使用限制 或更改 .wslconfig 文件。

终止

PowerShell

```
wsl --terminate <Distribution Name>
```

若要终止指定的分发或阻止其运行，请将 `<Distribution Name>` 替换为目标分发的名称。

标识 IP 地址

- `wsl hostname -I`：返回通过 WSL 2 安装的 Linux 分发版的 IP 地址（WSL 2 VM 地址）
- `ip route show | grep -i default | awk '{ print $3}'`：返回从 WSL 2（WSL 2 VM）中看到的 Windows 计算机的 IP 地址

有关更详细的说明，请参阅 使用 WSL 访问网络应用程序：识别 IP 地址。

导出分发

PowerShell

```
wsl --export <Distribution Name> <FileName>
```

将指定分发的快照导出为新的分发文件。 默认为 tar 格式。 文件名可以是`-`，用于标准输入。 选项包括：

- `--vhd`：指定导出分发应为 .vhdx 文件而不是 tar 文件（仅使用 WSL 2 支持）

导入发行版

PowerShell

```
wsl --import <Distribution Name> <InstallLocation> <FileName>
```

将指定的 tar 文件导入为新的分发版。 文件名可以是`-`，用于标准输入。 选项包括：

- `--vhd`：指定导入分发应为 .vhdx 文件而不是 tar 文件（仅使用 WSL 2 支持）
- `--version <1/2>`：指定是否将分发导入为 WSL 1 还是 WSL 2

就地导入分发包

PowerShell

```
wsl --import-in-place <Distribution Name> <FileName>
```

将指定的 .vhdx 文件导入为新的分发版。 虚拟硬盘必须在 ext4 文件系统类型中格式化。

注销或卸载 Linux 分发版

若要注销和卸载 WSL 分发版，

PowerShell

```
wsl --unregister <DistributionName>
```

将`<DistributionName>`替换为您目标的 Linux 发行版名称，会使该发行版从 WSL 中注销，以便您可以重新安装或清理它。 **谨慎：** 注销后，与该分发关联的所有数据、设置和软件都将永久丢

失。 从应用商店重新安装将安装分发版的干净副本。 例如， `wsl --unregister Ubuntu` 将从 WSL 中可用的分发中删除 Ubuntu。 运行 `wsl --list` 将显示它不再列出。

还可以像任何其他应用商店应用程序一样卸载 Windows 计算机上的 Linux 分发应用。 若要重新安装，请在 Microsoft 应用商店中找到分发版，然后选择“启动”。

装载磁盘或设备

PowerShell

```
wsl --mount <DiskPath>
```

在所有 WSL2 分发版中，通过将 `<DiskPath>` 替换为磁盘所在的目录\文件路径来附加和装载物理磁盘。 请参阅 在 WSL 2 中装载 Linux 磁盘。 选项包括：

- `--vhd`：指定 `<Disk>` 引用虚拟硬盘。
- `--name`：使用装入点的自定义名称装载磁盘
- `--bare`：将磁盘附加到 WSL2，但不装载它。
- `--type <Filesystem>`：在装载磁盘时使用的文件系统类型，如果未指定，则默认为 ext4。此命令也可以输入为： `wsl --mount -t <Filesystem>`。可以使用以下命令检测文件系统类型，例如： `blkid <BlockDevice>` `blkid <dev/sdb1>`。
- `--partition <Partition Number>`：要装载的分区的索引号（如果未指定）默认为整个磁盘。
- `--options <MountOptions>`：装载磁盘时，可以包含一些特定于文件系统的选项。 例如，ext4 装载选项 ，例如： `wsl --mount -o "data-ordered"` 或 `wsl --mount -o "data=writeback`。 但是，目前仅支持特定于文件系统的选项。 不支持泛型选项，例如 `ro`， `rw` 或 `noatime`。

> **注意**
>
> 如果要运行 32 位进程才能访问 wsl.exe（64 位工具），则可能需要以下列方式运行命令：`C:\Windows\Sysnative\wsl.exe --command`

卸载磁盘

PowerShell

```
wsl --unmount <DiskPath>
```

卸载磁盘路径中给定的磁盘，如果未提供磁盘路径，则此命令将卸载并分离所有装载的磁盘。

弃用的 WSL 命令

PowerShell

```
wslconfig.exe [Argument] [Options]
```

PowerShell

```
bash [Options]
```

PowerShell

```
lxrun /[Argument]
```

这些命令是用于配置随 WSL 一起安装的 Linux 分发版的原始 wsl 语法，但已替换为 `wsl` 或 `wsl.exe` 命令语法。

Last updated on 2025/12/01

WSL 和开放源代码资源

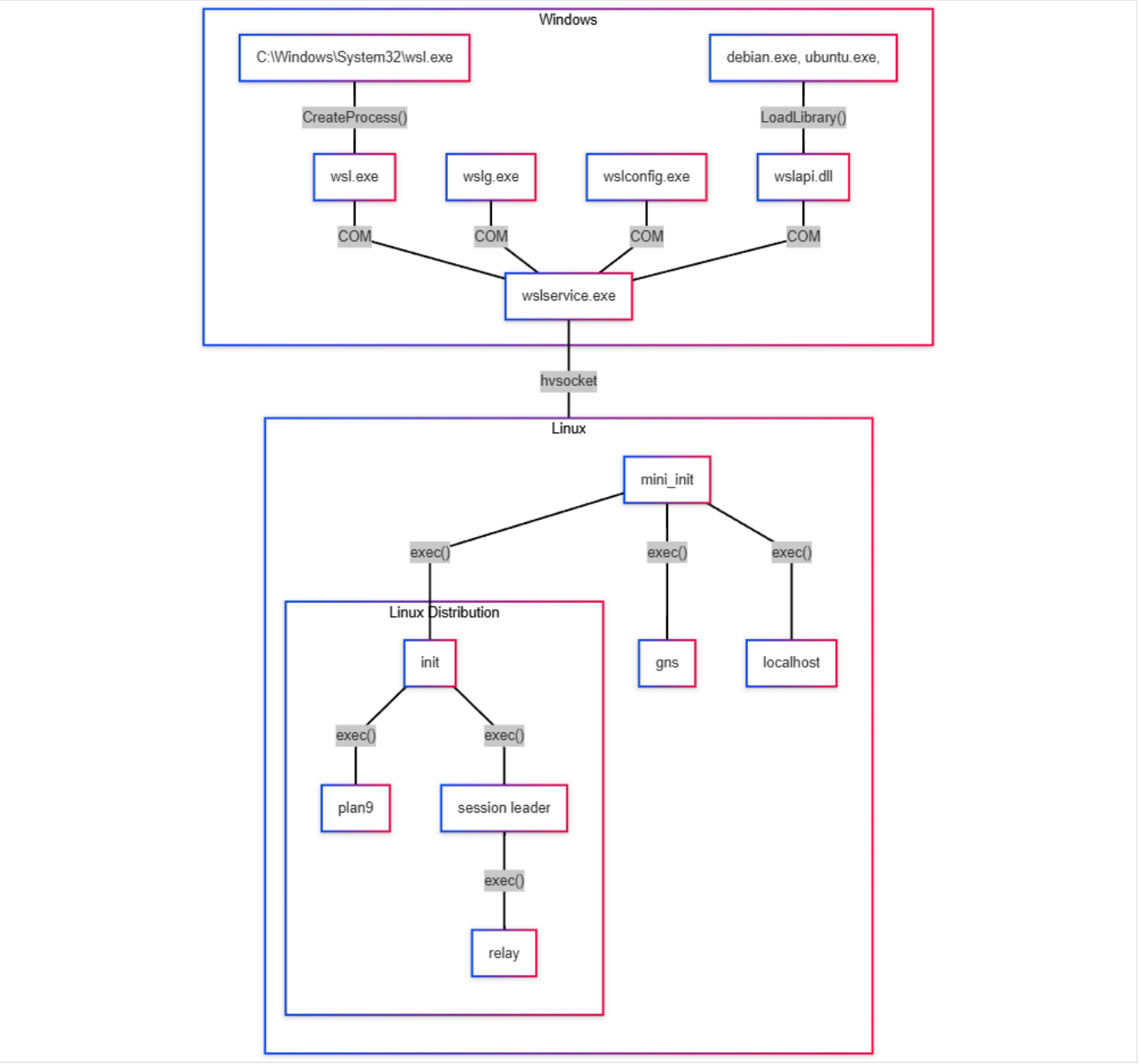
适用于 Linux 的 Windows 子系统（WSL）现在是开放源代码。

支持 WSL 的代码在 GitHub 上提供：microsoft/WSL 版本。

了解这意味着什么，如何下载 WSL、从源代码生成以及 WSL 组件的概述等。

组件概述

WSL 由一组分发组件组成。 有些在 Windows 中运行，有些在 WSL 2 虚拟机中运行。



WSL 的代码可以分为以下主要区域：

- 要与 WSL 交互的入口点是命令行可执行文件：`wsl.exe`、`wslconfig.exe`和`wslg.exe`。 这些是从 `C:\\Windows\System32\wsl.exe` > `CreateProcess()`中运行的进程。

- 启动 WSL 虚拟机（VM）、启动已安装的 Linux 发行版、挂载文件共享访问等的 WSL 服务：`wslservice.exe`
- Linux init 和守护进程，以及在 Linux 中运行的二进制文件用于提供 WSL 功能：`init` 用于启动，`gns` 用于网络连接，`localhost` 用于端口转发等。
- 使用 WSL 的计划 9 服务器实现将 Linux 文件共享到 Windows： `plan9`

详细了解 WSL 开源文档站点中的每个组件： wsl.dev 。

这些其他开源组件支持 WSL 源代码：

- microsoft/wslg：启用适用于 Linux 的 Windows 子系统以包括对 Wayland 和 X 服务器相关方案的支持：图形用户界面（GUI）通过 UI（而不仅仅是命令行）运行 Linux 应用。
- microsoft/WSL2-Linux-Kernel：Windows Subsystem for Linux 2（WSL2）中使用的 Linux 内核的源：WSL2 Linux 内核。

以下组件仍然是 Windows 映像的一部分，目前不开放源代码：

- `Lxcore.sys`：支持 WSL 1 的内核端驱动程序
- `P9rdr.sys` 和 `p9np.dll`：运行 `\\wsl.localhost` 文件系统重定向的组件（从 Windows 到 Linux）

有关开放源代码 WSL 背后的历史和社区的详细信息，请参阅来自 Microsoft 内部版本 2025 的博客公告： 适用于 Linux 的 Windows 子系统现在由皮埃尔·布莱（Pierre Boulay）开放源代码

访问 GitHub 上的 WSL 存储库以参与： microsoft/WSL 。

Last updated on 2025/08/07

如何使用 WSL 在 Windows 上安装 Linux

开发人员可以在 Windows 计算机上同时访问 Windows 和 Linux 的强大功能。 借助适用于 Linux 的 Windows 子系统（WSL），开发人员可以安装 Linux 分发版（如 Ubuntu、OpenSUSE、Kali、Debian、Arch Linux 等），并在 Windows 上直接使用 Linux 应用程序、实用工具和 Bash 命令行工具（未经修改），无需传统虚拟机或双包设置的开销。

先决条件

必须运行 Windows 10 版本 2004 及更高版本（内部版本 19041 及更高版本）或 Windows 11 才能使用以下命令。 如果使用的是早期版本，请参阅 手动安装页。

安装 WSL 命令

现在，可以使用单个命令安装运行 WSL 所需的所有内容。 右键单击并选择“以 **管理员** 身份运行”，在管理员模式下打开 PowerShell，输入 wsl --install 命令，然后重新启动计算机。

PowerShell

```
wsl --install
```

此命令将启用运行 WSL 并安装 Linux 的 Ubuntu 分发所需的功能。 （可以更改此默认分布）。

如果运行的是较旧的版本，或者只是不想使用安装命令，并且想要分步说明，请参阅 **适用于旧版本的 WSL 手动安装步骤**。

首次启动新安装的 Linux 分发版时，控制台窗口将打开，系统会要求你等待文件取消压缩并存储在计算机上。 所有未来的发射应该需要不到一秒钟的时间。

> **注意**
>
> 仅当 WSL 根本不安装时，上述命令才有效。 如果运行 `wsl --install` 并查看 WSL 帮助文本，请尝试运行 `wsl --list --online` 以查看可用发行版列表并运行 `wsl --install -d <DistroName>` 以安装发行版。 如果安装过程停在 0.0%，请先运行 `wsl --install --web-download -d <DistroName>` 下载发行版，然后再进行安装。 若要卸载 WSL，请参阅卸载旧版 WSL 或 **注销或卸载 Linux 分发版**。

更改安装的默认 Linux 分发版

默认情况下，已安装的 Linux 分发版将为 Ubuntu。 可以通过使用`-d`标志来更改这一点。

- 若要更改安装的分发版，请输入：

  PowerShell

  ```
  wsl.exe --install [Distro]
  ```

  将 `[Distro]` 替换为您想要安装的分发版名称。

- 若要查看可通过在线商店下载的可用 Linux 分发版列表，请输入：

  PowerShell

  ```
  wsl.exe --list --online
  ```

如果在安装过程中遇到问题，请查看故障排除指南 安装部分。

若要安装未列为可用的 Linux 分发版，可以使用 TAR 文件 导入任何 Linux 分发 版。 或者在某些情况下，与 Arch Linux 一样，可以使用文件进行安装 `.appx` 。 还可以创建自己的 自定义 Linux 分发版，以便与 WSL 一起使用。

设置 Linux 用户信息

安装 WSL 后，需要为新安装的 Linux 分发版创建用户帐户和密码。 请参阅 设置 WSL 开发环境指南的最佳做法，了解详细信息。

配置和最佳实践

建议遵循我们的 设置 WSL 开发环境的最佳实践 指南，通过逐步演示来了解如何为已安装的 Linux 发行版设置用户名和密码，使用基本 WSL 命令，安装和自定义 Windows 终端，为 Git 版本控制、代码编辑和调试使用 VS Code 远程服务器，好的文件存储实践，设置数据库，装载外部驱动器，设置 GPU 加速等。

检查你正在运行的 WSL 的版本

可以通过输入命令列出已安装的 Linux 分发版并检查每个版本的 WSL：

PowerShell

```
wsl.exe --list --verbose
```

若要在安装新的 Linux 分发版时将默认版本设置为 WSL 1 或 WSL 2，请使用以下命令：

PowerShell

```
wsl.exe --set-default-version <1|2>
```

若要设置与命令一起使用 `wsl` 的默认 Linux 分发版，请输入：

PowerShell

```
wsl.exe --set-default <Distro>
```

替换为 `<Distro>` 要使用的 Linux 分发版的名称。 例如，在 PowerShell 中，输入： `wsl -s Debian` 将默认分发设置为 Debian。 现在，从 Powershell 运行 `wsl npm init` 将在 Debian 中运行 `npm init` 命令。

若要在不更改默认分发的情况下从 PowerShell 中运行特定的 wsl 分发版，请使用以下命令：

PowerShell

```
wsl.exe --distribution <DistroName>
```

替换为 `<DistroName>` 要使用的分发的名称。

在 WSL 基本命令指南中了解详细信息。

将版本从 WSL 1 升级到 WSL 2

默认情况下，使用 `wsl --install` 命令安装的新 Linux 安装将设置为 WSL 2。

若要查看 Linux 分发版是否设置为 WSL 1 或 WSL 2，请使用以下命令： `wsl -l -v` 可以使用以下命令从 WSL 1 升级到 WSL 2 或从 WSL 2 降级到 WSL 1：

PowerShell

```
wsl.exe --set-version <Distro> <1|2>
```

替换为 `<Distro>` 要更新的 Linux 分发版的名称。 例如， `wsl --set-version Ubuntu 2` 将 Ubuntu 分发设置为使用 WSL 2。

如果在命令可用之前 `wsl --install` 手动安装了 WSL，则还可能需要启用 WSL 2 使用的 虚拟机可选组件 ，并 安装内核包 （如果尚未这样做）。

若要了解更多信息，请参阅 WSL 的命令参考以获取 WSL 命令列表，查看 WSL 1 和 WSL 2 的比较以获得关于您工作场景适用版本的指导，或参考 设置 WSL 开发环境的最佳实践以获取关于设置高效开发工作流程的一般指导。

使用 WSL 运行多个 Linux 分发版的方法

WSL 支持运行您想安装的许多不同的 Linux 发行版。 这包括从 Microsoft 应用商店 中选择分发版、 导入自定义分发版或 生成自己的自定义分发版。

安装 Linux 分发版后，可通过多种方式运行：

- 从 Windows 终端 ***（建议）*** 使用 Windows 终端支持任意多的命令行，并且允许你在多个选项卡或窗口窗格中打开它们，并在多个 Linux 分发版或其他命令行（PowerShell、命令提示符、Azure CLI 等）之间快速切换。 可以使用独特的配色方案、字体样式、大小、背景图像和自定义键盘快捷方式完全自定义终端。 了解详细信息。
- 可以通过访问 Windows 开始菜单并键入已安装的分发版的名称来直接打开 Linux 分发版。 例如：“Ubuntu”。 这将在一个独立的控制台窗口中打开 Ubuntu。
- 在 PowerShell 中，可以输入已安装的分发版的名称。 例如：`ubuntu`
- 在 PowerShell 中，可以通过输入： `wsl.exe` 在当前命令行中打开默认 Linux 分发版。
- 在 PowerShell 中，可以通过输入：`wsl [command]`，在当前命令行中使用默认 Linux 分发版，而无需输入新发行版。 你可以用 `[command]` 替换为 WSL 命令，例如：`wsl -l -v` 列出已安装的发行版，或 `wsl pwd` 查看当前目录路径在 WSL 中装载的位置。 在 PowerShell 中，该命令 `Get-Date` 将提供 Windows 文件系统中的日期，并提供 `wsl date` Linux 文件系统中的日期。

选择的方法应取决于你正在做的事情。 如果在 PowerShell 窗口中打开了 WSL 命令行并想要退出，请输入以下命令： `exit`

想要试用最新的 WSL 预览功能?

通过加入 Windows 预览体验计划 来试用 WSL 的最新功能或更新。 加入 Windows 预览体验成员后，可以选择希望从 Windows 设置菜单中接收预览版的频道，以自动接收与该版本关联的任何 WSL 更新或预览功能。 可以选择：

- Canary 频道：
  - 适合高度技术性用户。
  - 在开发周期的早期预览最新的平台更改。
  - 这些版本可能不稳定，并且仅发布任何文档。
- 开发频道：
  - 非常适合爱好者。
  - 访问最新的 Windows 11 预览版，因为我们培育新想法并开发长期领先功能。
  - 会有一些粗糙的边缘和低稳定性。
- Beta 频道：
  - 非常适合早期采用者。
  - 预览并提供有关稳定环境中的 Windows 11 预发行功能的反馈。

- 发布预览频道：
  - 如果想要预览修补程序和某些关键功能，并在正式发布之前获取对下一版本的 Windows 的可选访问权限，则理想。
  - 此外，建议将此频道用于商业用户。

如果不希望将 Windows 安装切换到预览频道，仍可以通过发出命令来测试 WSL 的最新预览：

PowerShell

```
wsl.exe --update --pre-release
```

有关详细信息，请查看 GitHub 上的 WSL 发布页。

后续步骤

接下来，让我们了解 WSL 的基本命令。

基本 WSL 命令

脱机安装

若要脱机安装 WSL，需要执行以下步骤：

- 从 GitHub 发布页下载并安装最新的 WSL MSI 包
- 使用管理员权限打开 PowerShell 窗口，并运行 `dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart` 以启用虚拟机平台可选组件。 可能需要重新启动计算机才能生效。
- 通过 .wsl 文件安装分发版。 可以在所选发行版 的DistributionInfo.json 中找到用于下载这些文件的 URL。

其他资源

- Windows 命令行博客：使用 Windows 10 版本 2004 及更高版本中提供的单个命令安装 WSL

Last updated on 2025/08/15

旧版 WSL 的手动安装步骤

为简单起见，我们通常建议使用 wsl --install 安装适用于 Linux 的 Windows 子系统，但如果运行的是较旧的 Windows 版本或 Windows Server Core，则可能不受支持。 我们包括了下面的手动安装步骤。 如果在安装过程中遇到问题，请查看故障排除指南 安装部分。

步骤 1 - 启用适用于 Linux 的 Windows 子系统

必须先启用“适用于 Linux 的 Windows 子系统”可选功能，然后才能在 Windows 上安装任何 Linux 分发版。

以管理员身份打开 PowerShell （> **PowerShell** > **右键单击** > **以管理员身份运行**） 并输入以下命令：

PowerShell

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all
/norestart
```

建议现在继续执行 步骤 2 - 检查运行 WSL 2 的要求，更新到 WSL 2，但如果只想安装 WSL 1，现在可以 **重启** 计算机并转到 步骤 6 - 安装所选的 Linux 分发版。 若要更新到 WSL 2，**等待重启** 计算机，然后继续执行下一步。

步骤 2 - 检查运行 WSL 2 的要求

若要更新到 WSL 2，必须运行 Windows 10...

- 对于 x64 系统：版本 1903 或更高版本，内部版本为 18362.1049 或更高版本。
- 对于 ARM64 系统：版本 2004 或更高版本，内部版本为 19041 或更高版本。

或 Windows 11。

> **注意**
>
> 低于 18362 的内部版本不支持 WSL 2。 使用 **Windows 更新助手** 更新 Windows 版本。 Windows 版本 1903 支持也仅适用于 x64 系统。 如果使用 Arm64 版本的 Windows，则需要升级到 Windows 10 版本 2004 或更高版本才能完全访问 WSL 2。 有关详细信息，请参阅 **WSL 2 即将支持 Windows 10 版本 1903 和 1909** 。

若要检查 Windows 版本及内部版本号，请按 Windows 徽标键 + R，键入“winver”，然后选择“确定”。 在设置菜单中更新到最新的 Windows 版本。

> ⓘ **注意**
>
> 如果运行的是 Windows 10 版本 1903 或 1909，请从 Windows 菜单中打开“设置”，导航到“更新 & 安全性”，然后选择“检查更新”。 内部版本号必须是 18362.1049+ 或 18363.1049+，次要内部版本号需要高于 .1049。 **WSL 2 即将支持 Windows 10 版本 1903 和 1909** 。

步骤 3 - 启用虚拟机功能

安装 WSL 2 之前，必须启用 **虚拟机平台** 可选功能。 计算机将需要 虚拟化功能 才能使用此功能。

以管理员身份打开 PowerShell 并运行：

PowerShell

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

重启 计算机以完成 WSL 安装和更新到 WSL 2。

步骤 4 - 下载 Linux 内核更新包

Linux 内核更新包安装最新版本的 WSL 2 Linux 内核 ，以便在 Windows 操作系统映像中运行 WSL。 要从 Microsoft Store 运行WSL，并使用更频繁发布的更新，请使用 `wsl.exe --install` 或 `wsl.exe --update`。

1. 下载最新的包：

   - WSL2 Linux 内核更新包适用于 x64 计算机
   - 适用于 ARM64 计算机的 WSL2 Linux 内核更新包

   > ⓘ **注意**
   >
   > 如果不知道系统体系结构，请按 Win + X 键，在菜单中查找系统，或按 Y 键打开“关于系统”设置（或控制面板）界面，并查找当前系统的体系结构。

2. 运行在上一步中下载的更新包。 （双击以运行 - 系统会提示你输入提升的权限，选择“是”以批准此安装。

安装完成后，转到下一步 - 在安装新的 Linux 分发版时将 WSL 2 设置为默认版本。 （如果希望将新的 Linux 安装设置为 WSL 1，请跳过此步骤）。

> ⓘ **注意**
>
> 有关详细信息，请参阅 Windows 命令行博客上的文章对更新 WSL2 Linux 内核的更改。

步骤 5 - 将 WSL 2 设置为默认版本

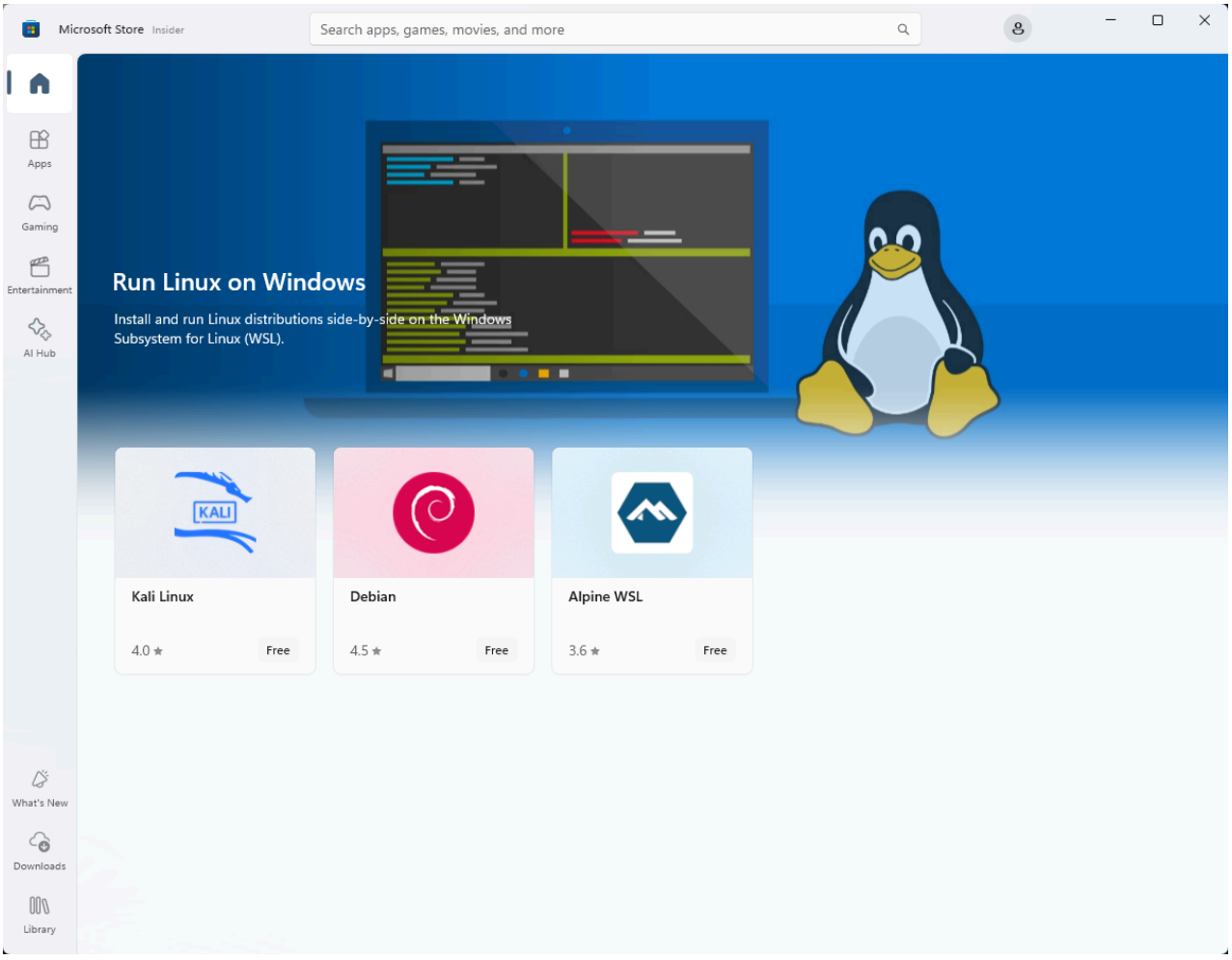
打开 PowerShell 并运行以下命令，在安装新的 Linux 分发版时将 WSL 2 设置为默认版本：

PowerShell

```
wsl --set-default-version 2
```

步骤 6 - 安装所选 Linux 分发版

1. 打开 Microsoft Store 并选择你喜欢的 Linux 分发版。



中 Linux 分发版的视图

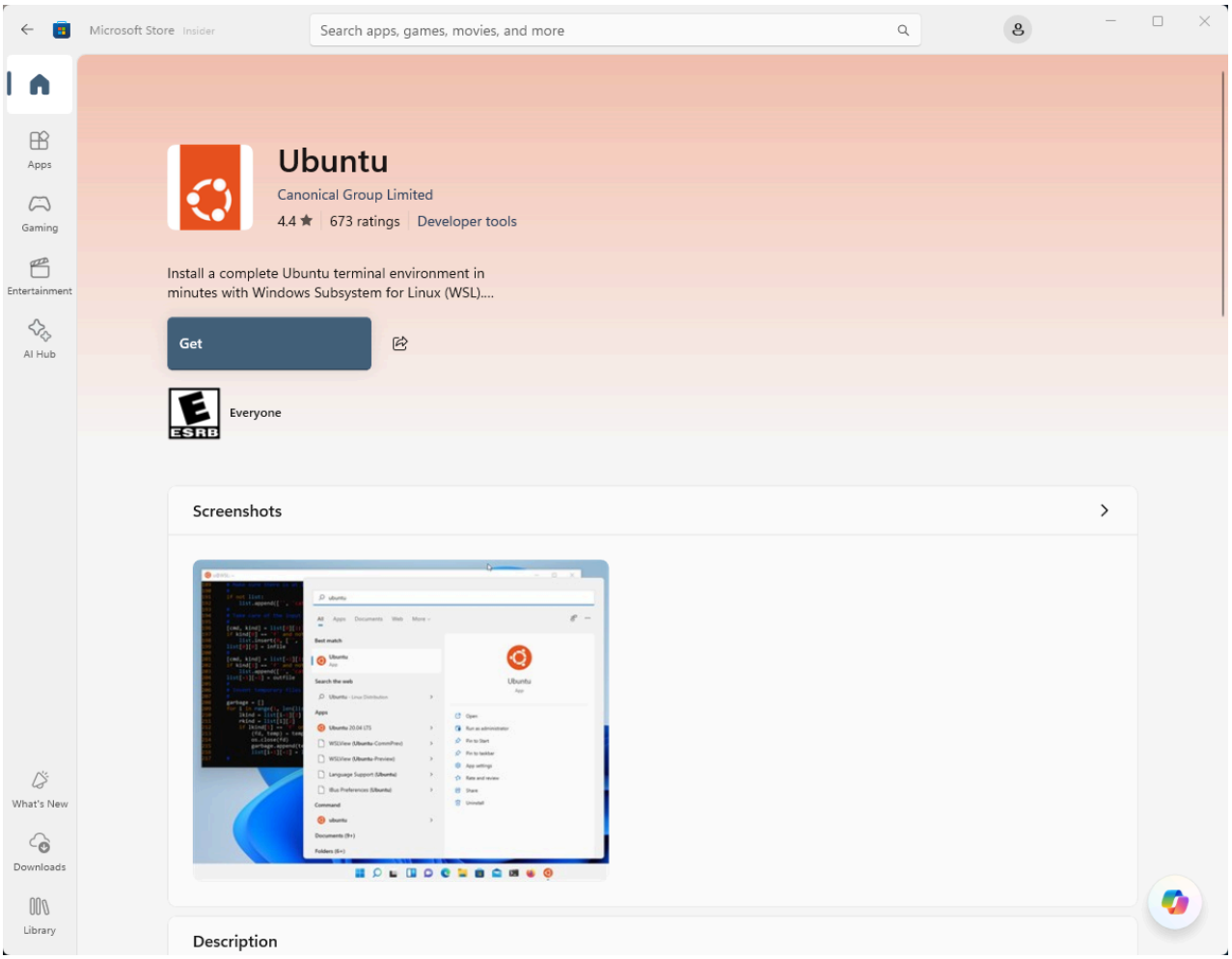
以下链接将打开每个发行版的 Microsoft 商店页面：

- Ubuntu：
  - Ubuntu
  - Ubuntu 24.04.1 LTS
  - Ubuntu 22.04.5 LTS
  - Ubuntu 20.04.6 LTS
  - Ubuntu 20.04 LTS
  - Ubuntu 18.04.6 LTS
  - Ubuntu 18.04 LTS
  - Ubuntu （预览版）
- Debian：
  - Debian
- Arch Linux：
  - Arch WSL
- Fedora：
  - Fedora WSL
- deepin：
  - deepin WSL
- Alpine Linux：
  - Alpine WSL
- openEuler：
  - openEuler 24.09
  - openEuler 24.03
  - openEuler 23.09
  - openEuler 23.03
  - openEuler 22.09
  - openEuler 22.03
  - openEuler 20.03
- SUSE：
  - openSUSE：
    - openSUSE Tumbleweed
    - openSUSE Leap 15.6
    - openSUSE Leap 15.5
    - openSUSE-Leap-15-1
  - SUSE Linux：
    - SUSE Linux Enterprise 15 SP6
    - SUSE Linux Enterprise 15 SP5
    - SUSE Linux Enterprise Server 15 SP1
    - SUSE Linux Enterprise Server 12 SP5
- Pistachio Linux：
  - Pistachio Linux
- Kali Linux：

    - Kali Linux
- Oracle Linux：
    - Oracle Linux 9.4
    - Oracle Linux 9.3
    - Oracle Linux 9.2
    - Oracle Linux 9.1
    - Oracle Linux 9
    - Oracle Linux 8.10
    - Oracle Linux 8.9
    - Oracle Linux 8.8
    - Oracle Linux 8.7
    - Oracle Linux 8.6
    - Oracle Linux 8.5
    - Oracle Linux 7.9
- AlmaLinux OS：
    - AlmaLinux OS 9
    - AlmaLinux OS 8
- AOSC OS：
    - WSL 上的 AOSC OS
- Athena OS
    - Athena OS
- Slackware：
    - WSLackware （非官方）
- Fedora Remix：
    - Fedora Remix for WSL （付费）
- 彭温：
    - 彭温 （付费）
    - 彭温企业 9 （付费）
    - 彭温企业 8 （付费）
    - 彭温企业 7
- Rocky Linux：
    - RLU 9 （非官方， 付费）
    - 落基 8 RC 非官方 （非官方， 付费）
- Azure Linux：
    - 斯瓦比 （非官方， 付费）
    - 斯瓦比2 （非官方， 付费）

*付费仅表示它在 Microsoft 应用商店中付费，但这并不意味着它没有其他免费下载频道。

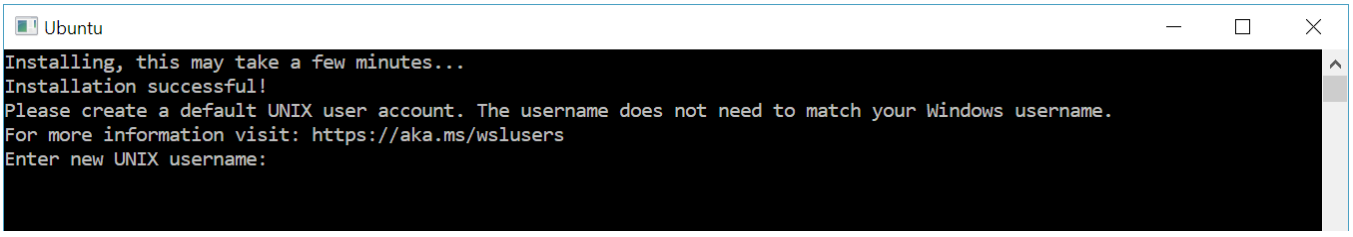
2. 在分发页中，选择“获取”。



中的 Linux 发行版

首次启动新安装的 Linux 分发版时，控制台窗口将打开，系统将要求你等待一到两分钟，以便文件取消压缩并存储在电脑上。 所有未来的发射应该需要不到一秒钟的时间。

然后，需要 为新的 Linux 分发版创建用户帐户和密码。



祝贺！ 您已成功安装并设置了与 Windows 操作系统完全集成的 Linux 发行版！

排查安装问题

如果在安装过程中遇到问题，请查看故障排除指南 安装部分。

下载发行版

在某些情况下，你可能无法（或不想）使用 Microsoft 商店安装 WSL 的 Linux 发行版。 你可能正在运行不支持 Microsoft 应用商店的 Windows Server 或 Long-Term 服务 （LTSC） 桌面 OS SKU，或者公司网络策略和/或管理员不允许在环境中使用 Microsoft 应用商店。 在这些情况下，虽然 WSL 本身可用，但可能需要直接下载 Linux 分发版。

如果 Microsoft 应用商店应用不可用，则可以使用以下链接下载并手动安装 Linux 分发版：

- Ubuntu：
  - Ubuntu （x64，arm64）
  - Ubuntu 24.04 LTS （x64，arm64）
  - Ubuntu 22.04 LTS （x64，arm64）
  - Ubuntu 20.04 LTS （x64，arm64）
  - Ubuntu 18.04 LTS （x64）
  - Ubuntu 18.04 LTS ARM （arm64）
  - Ubuntu 16.04 （x64）
- Debian：
  - Debian GNU/Linux （x64， arm64）
- Kali Linux：
  - Kali Linux 滚动
- OracleLinux：
  - Oracle Linux 9.1 （x64）
  - Oracle Linux 8.7 （x64）
  - Oracle Linux 8.5 （x64）
  - Oracle Linux 7.9 （x64）
  - SUSE：
    - openSUSE：
      - openSUSE Tumbleweed （x64）
      - openSUSE Leap 15.6 （x64）
      - openSUSE Leap 15.3 （x64）
      - openSUSE Leap 15.2 （x64）
    - SUSE Linux：
      - SUSE Linux Enterprise Server 15 SP6 （x64）
      - SUSE Linux Enterprise Server 15 SP5 （x64）
      - SUSE Linux Enterprise Server 15 SP3 （x64）
      - SUSE Linux Enterprise Server 15 SP2 （x64）
      - SUSE Linux Enterprise Server 12 （x64）
- Fedora Remix：
  - 用于 WSL 的 Fedora Remix （x64， arm64）

这将导致 `<distro>.appx` 包下载到所选文件夹。

如果愿意，还可以通过命令行下载首选分发版，可以将 PowerShell 与 Invoke-WebRequest cmdlet 配合使用。 例如，若要下载 Ubuntu 20.04，

PowerShell

```
Invoke-WebRequest -Uri https://aka.ms/wslubuntu2004 -OutFile Ubuntu.appx -
UseBasicParsing
```

> **提示**
>
> 如果下载需要很长时间，请通过设置 `$ProgressPreference = 'SilentlyContinue'` 关闭进度栏

还可以选择使用 curl 命令行实用工具 进行下载。 使用 curl 下载 Ubuntu 20.04：

PowerShell

```
curl.exe -LR -o ubuntu-2004.Appx https://aka.ms/wslubuntu2204
```

在此示例中，执行 `curl.exe`（而不仅仅是 `curl`），以确保在 PowerShell 中调用实际 curl 可执行文件，而不是 Invoke-WebRequest的 PowerShell curl 别名。

使用 Add-AppxPackage 安装 Appx 包

注意 以下命令在服务器核心安装上不起作用

下载分发版后，导航到包含下载的文件夹，并在该目录中运行以下命令，其中 `app-name` 是 Linux 分发.appx文件的名称。

PowerShell

```
Add-AppxPackage .\app_name.Appx
```

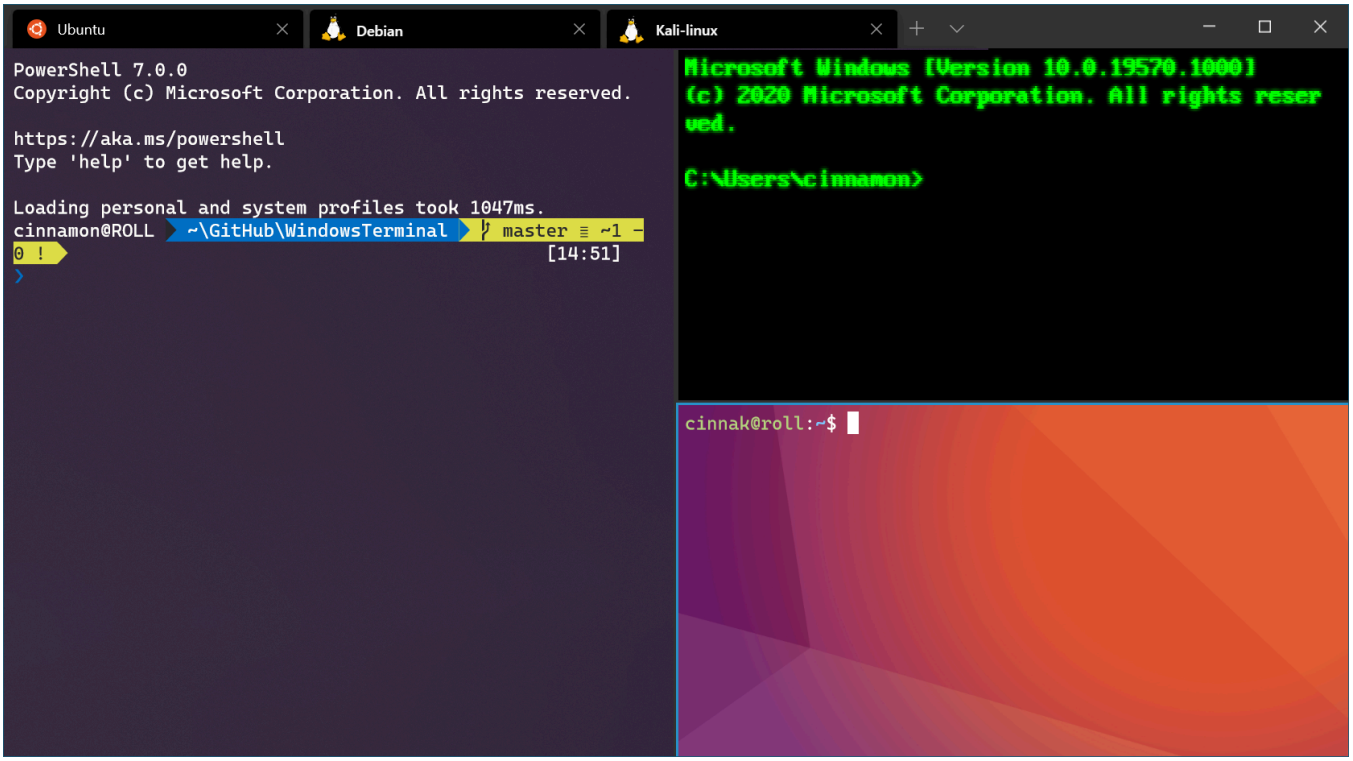
Appx 包下载完成后，可以通过双击 appx 文件开始运行新分发版。 （在此步骤完成之前，命令 `wsl -l` 不会显示分发版已安装）。

如果使用 Windows Server，或者运行上述命令时遇到问题，可以在 Windows Server 文档页上找到备用安装说明，通过将其更改为 zip 文件来安装 `.Appx` 文件。

安装分发版后，请按照说明 为新的 Linux 分发版创建用户帐户和密码。

安装 Windows 终端（可选）

使用 Windows 终端可以打开多个选项卡或窗口窗格，以便显示多个 Linux 分发版或其他命令行（PowerShell、命令提示符、Azure CLI 等）并快速切换。 可以使用独特的配色方案、字体样式、大小、背景图像和自定义键盘快捷方式完全自定义终端。 了解更多信息。



安装 Windows 终端

Last updated on 2025/08/15

Windows Server 安装指南

适用于 Linux 的 Windows 子系统（WSL）可用于在 Windows Server 2019（版本 1709）及更高版本上安装。 本指南将演练在计算机上启用 WSL 的步骤。

在 Windows Server 2022 和 2025 桌面体验上安装 WSL

Windows Server 2022 现在使用以下命令支持简单的 WSL 安装：

PowerShell

```
wsl.exe --install
```

现在，可以在 **管理员** PowerShell 中输入此命令，然后重启计算机，安装在 Windows Server 2022 上运行 WSL 所需的所有内容。

此命令将启用所需的可选组件、下载最新的 Linux 内核、将 WSL 2 设置为默认组件，并为你安装 Linux 分发版*（默认情况下为 Ubuntu）*。

有关如何执行以下操作的详细信息，请参阅 WSL 标准文档。

- 更改安装的默认 Linux 分发版。
- 设置 Linux 用户名和密码。
- 检查正在运行的 WSL 版本
- 更新和升级包。
- 添加其他发行版
- 将 Git 与 WSL 配合使用。

在早期版本的 Windows Server 和 Server Core 上安装 WSL

若要在 Windows Server 2019（版本 1709+）以及 Server Core for 2019 及更高版本上安装 WSL，可以按照下面的手动安装步骤进行作。

启用适用于 Linux 的 Windows 子系统

在 Windows 上运行 Linux 分发版之前，必须启用“适用于 Linux 的 Windows 子系统”可选功能和重新启动。

以管理员身份打开 PowerShell 并运行：

PowerShell

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux, VirtualMachinePlatform
```

安装 WSL 2 的 WSL 内核更新

Server Core 2025 不需要这样。

PowerShell

```
Invoke-WebRequest -Uri "https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi" -OutFile ".\wsl_update_x64.msi"
Start-Process "msiexec.exe" -ArgumentList "/i .\wsl_update_x64.msi /quiet" -NoNewWindow -Wait
```

下载 Linux 分发版

有关下载首选 Linux 分发版的说明和链接，请参阅手动安装页的" 下载分发 "部分。

提取并安装 Linux 分发版

下载 Linux 分发版后，若要提取其内容并手动安装，请执行以下步骤。 请注意，你将下载一个包含多个 appx 文件的 appx 文件。 在此示例中，我们将使用 debian。

1. 使用 tar.exe列出 appx 的内容：

   Windows 命令提示符

   ```
   > tar -tf .\debian.appx
   DistroLauncher-Appx_1.12.2.0_ARM64.appx
   DistroLauncher-Appx_1.12.2.0_scale-100.appx
   DistroLauncher-Appx_1.12.2.0_scale-125.appx
   DistroLauncher-Appx_1.12.2.0_scale-150.appx
   DistroLauncher-Appx_1.12.2.0_scale-400.appx
   DistroLauncher-Appx_1.12.2.0_x64.appx
   ```

   在我们的示例中，我们有一个 x64 位服务器，因此我们想要安装 `DistroLauncher-Appx_1.12.2.0_x64.appx`。

2. 将内容解压缩到名为 `\%USERPROFILE%\AppData\Local\DebianWSL\` 的新文件夹中。

   PowerShell

```
$debianWSLPath = Join-Path -Path $env:LocalAppData -ChildPath DebianWSL
New-Item -Path $debianWSLPath -ItemType Directory | Out-Null
Expand-Archive -Path ".\DistroLauncher-Appx_1.12.2.0_x64.appx" -DestinationPath
$debianWSLPath
```

3. 使用 PowerShell 将 Linux 分发路径添加到 Windows 环境路径（`C:\Users\Administrator\Ubuntu` 在此示例中）：

PowerShell

```
$userenv = [System.Environment]::GetEnvironmentVariable("Path", "User")
[System.Environment]::SetEnvironmentVariable("PATH", $userenv +
";$env:USERPROFILE\AppData\Local\DebianWSL", "User")
```

现在，您可以通过在任何路径上键入 `<DistributionName>.exe` 来启动您的发布版本。 例如：`ubuntu.exe`。 请注意，在服务器核心的情况下，需要从“开始”菜单启动新的 Powershell 实例，或注销并再次登录以刷新路径

安装完成后，可以为 新的 Linux 分发版创建用户帐户和密码。

Last updated on 2025/10/04

设置 WSL 开发环境

设置 WSL 开发环境的最佳做法分步指南。 了解如何运行命令以安装默认的 Bash Shell，它使用 Ubuntu，或者可以设置为安装其他 Linux 发行版、使用基本 WSL 命令、设置 Visual Studio Code 或 Visual Studio、Git、Windows 凭据管理器、MongoDB、Postgres 或 MySQL 等数据库、设置 GPU 加速、运行 GUI 应用等。

开始

适用于 Linux 的 Windows 子系统随 Windows 操作系统一起提供，但必须先启用它并安装 Linux 发行版，然后才能开始使用它。

若要使用简化的 --install 命令，必须运行最新版本的 Windows（内部版本 20262+）。 要检查您的 Windows 版本和内部版本号，请按 Windows 徽标键 + R，输入“winver”，然后选择“确定”。 可以使用“设置”菜单或 Windows 更新助手 进行更新。

如果希望安装除 Ubuntu 以外的 Linux 发行版，或者希望手动完成这些步骤，请参阅 WSL 安装页了解更多详细信息。

打开 PowerShell（或 Windows 命令提示符）并输入：

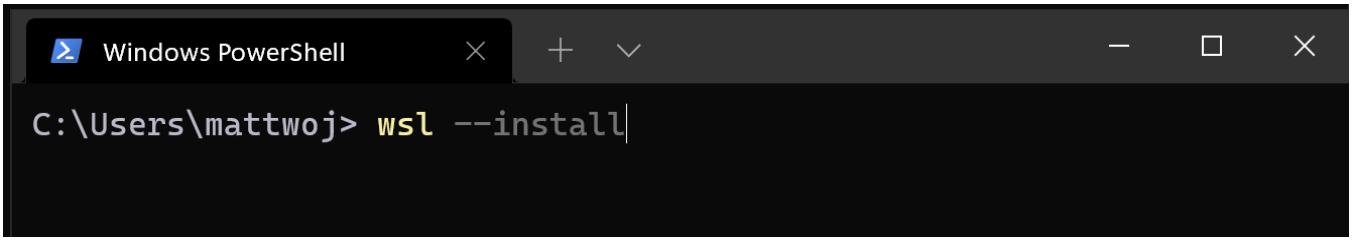
PowerShell

```
wsl --install
```

该 `--install` 命令执行以下作：

- 启用可选的 WSL 和虚拟机平台组件
- 下载并安装最新 Linux 内核
- 将 WSL 2 设置为默认值
- 下载并安装 Ubuntu Linux 发行版（可能需要重新启动）

在此安装过程中，你将需要重启计算机。



运行 wsl --install 的 PowerShell 命令行

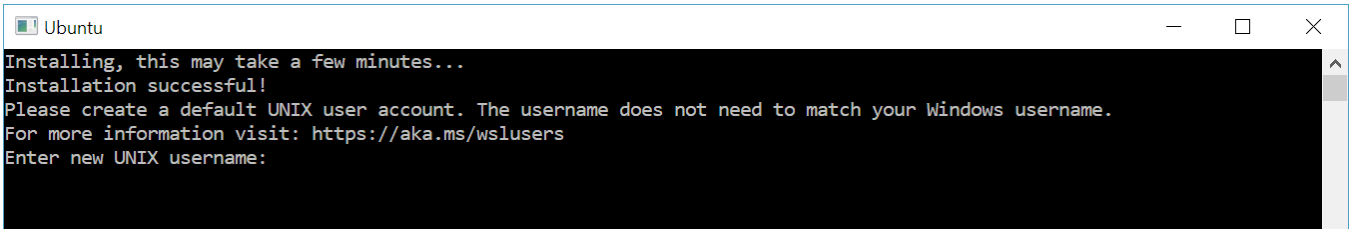
如果遇到任何问题，请查看排查安装问题一文。

设置 Linux 用户名和密码

使用 WSL 安装 Linux 发行版的过程完成后，使用“开始”菜单打开该发行版（默认情况下为 Ubuntu）。 系统将要求你为 Linux 发行版创建“用户名”和“密码”。

- 此**用户名**和**密码**特定于安装的每个单独的 Linux 分发版，与 Windows 用户名无关。
- 请注意，输入**密码**时，屏幕上不会显示任何内容。 这称为盲人键入。 你不会看到你正在键入的内容，这是完全正常的。
- 创建**用户名**和**密码**后，该帐户将是分发版的默认用户，并将在启动时自动登录。
- 此帐户将被视为 Linux 管理员，能够运行 `sudo` (Super User Do) 管理命令。
- 在 WSL 上运行的每个 Linux 发行版都有其自己的 Linux 用户帐户和密码。 每当添加分发版、重新安装或重置时，都必须配置一个 Linux 用户帐户。

> **注意**
>
> 随 WSL 一起安装的 Linux 发行版是按用户安装，不可与其他 Windows 用户帐户共享。 遇到用户名错误？ StackExchange：在 Linux 上的用户名中，应使用或不使用哪些字符？



在 Ubuntu 命令行中输入 UNIX 用户名

若要更改或重置密码，请打开 Linux 发行版并输入命令：`passwd`。 系统会要求你输入当前密码，然后要求输入新密码，之后再确认新密码。

如果忘记了 Linux 分发版的密码：

1. 请打开 PowerShell，并使用以下命令进入默认 WSL 分发版的根目录：`wsl -u root`

   > 如果需要在非默认的分发版中更新忘记的密码，请使用命令：`wsl -d <DistroName> -u root`，并将 `<DistroName>` 替换为目标分发版的名称。

2. 在 PowerShell 内的根级别打开 WSL 发行版后，可使用此命令更新密码：`passwd <username>`，其中 `<username>` 是发行版中帐户的用户名，而你忘记了它的密码。

3. 系统将提示你输入新的 UNIX 密码，然后确认该密码。 在您被告知密码已正确更新后，请在 PowerShell 内使用以下命令关闭 WSL：`exit`。

更新和升级软件包

建议使用发行版的首选包管理器定期更新和升级包。 对于 Ubuntu 或 Debian，请使用以下命令：

```bash
sudo apt update && sudo apt upgrade
```

Windows 不会自动更新或升级 Linux 分发版。 大多数 Linux 用户往往倾向于自行控制此任务。

添加其他发行版

若要添加其他 Linux 发行版，可以通过 Microsoft Store、通过 --import 命令或通过旁加载你自己的自定义发行版进行安装。 你可能还希望设置自定义 WSL 映像，以便在企业公司中分发。

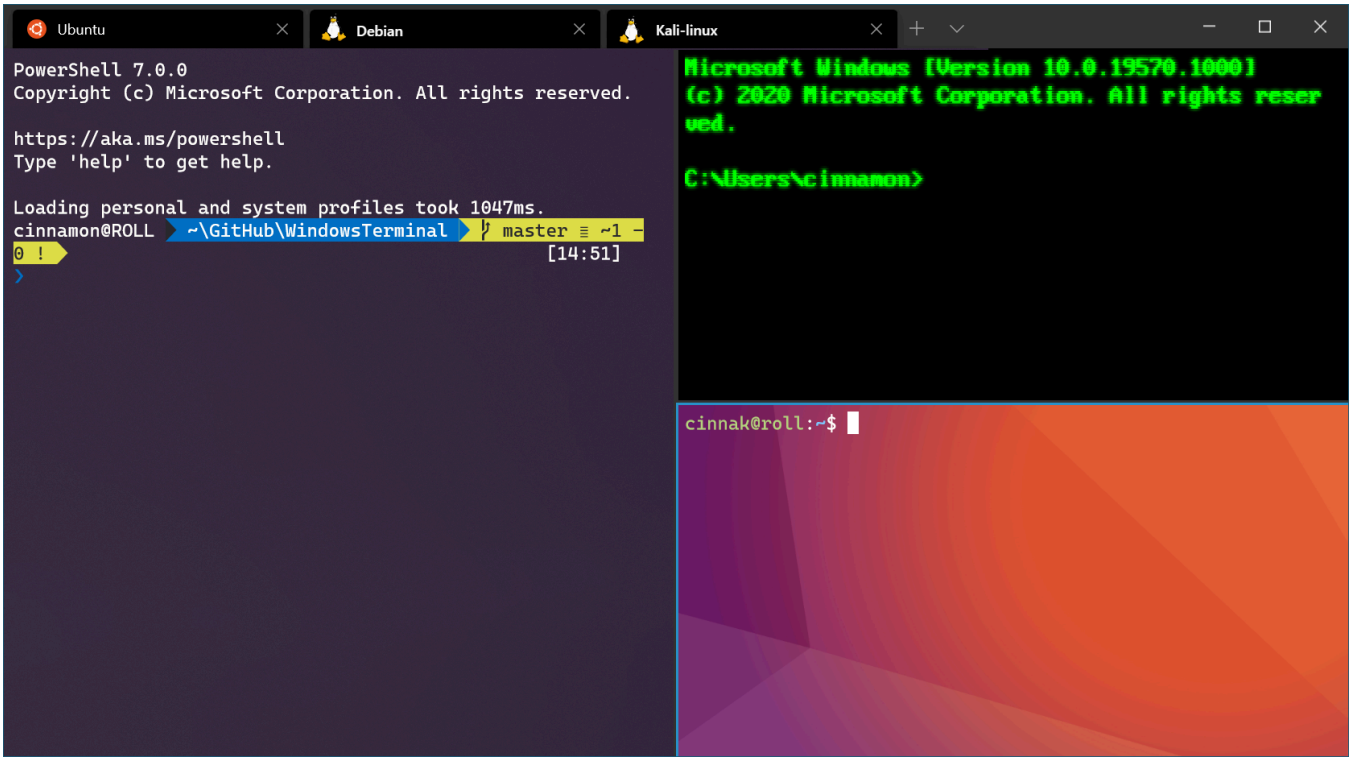
设置 Windows Terminal

Windows Terminal 可以运行任何具有命令行界面的应用程序。 它的主要功能包括多个选项卡、窗格、Unicode 和 UTF-8 字符支持、GPU 加速文本呈现引擎，你还可用它来创建你自己的主题并自定义文本、颜色、背景和快捷方式。

每当安装新的 WSL Linux 发行版时，都会在 Windows Terminal 中为其创建一个新实例，该实例可根据你的偏好进行自定义。

我们建议将 WSL 与 Windows Terminal 配合使用，特别是在计划同时使用多个命令行界面时。请参阅 Windows Terminal 文档，了解如何对其进行设置以及如何自定义首选项，包括：

- 从 Microsoft Store 安装 Windows Terminal 或 Windows Terminal（预览版）
- 使用命令面板
- 设置键盘快捷方式等自定义操作，使终端适应你的首选项
- 设置默认启动配置文件
- 自定义外观：主题、配色方案、名称和起始目录、背景图像等
- 了解如何使用命令行参数，例如使用拆分为窗口窗格或选项卡的多个命令行打开终端
- 了解搜索功能
- 查找提示和技巧，例如如何重命名选项卡或为其着色、使用鼠标交互或启用“Quake 模式”
- 查找有关如何设置自定义命令提示符、SSH 配置文件或选项卡标题的教程
- 查找自定义终端库和故障排除指南



Windows Terminal 屏幕截图

文件存储

- 若要在 Windows 文件资源管理器中打开 WSL 项目，请输入：`explorer.exe .`

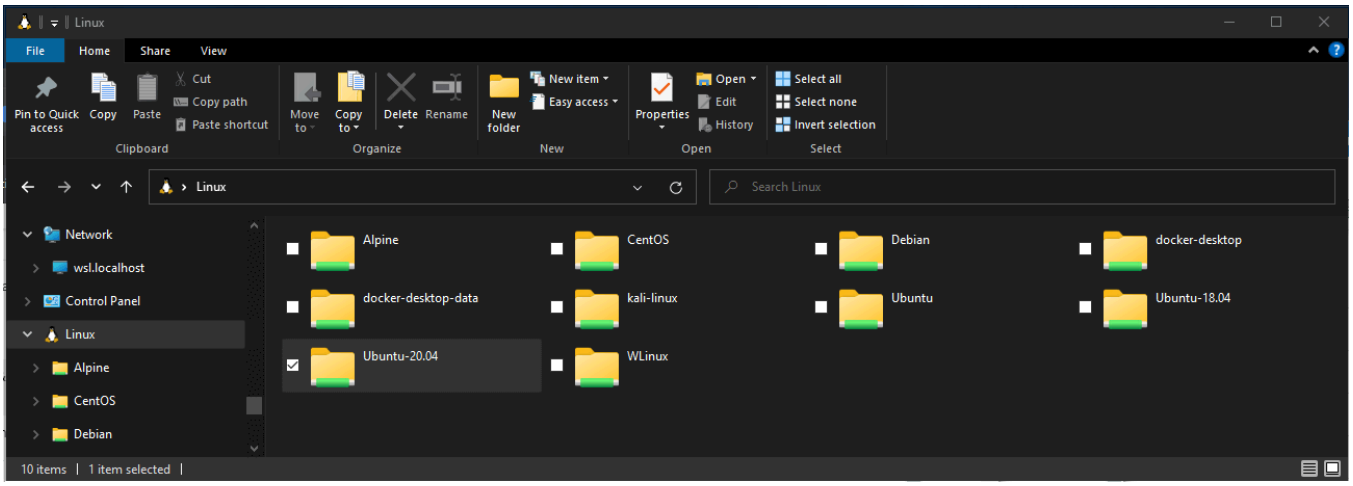
  请确保在命令的末尾添加句点以打开当前目录。

- 将项目文件与计划使用的工具存储在相同的操作系统上。

  若想获得最快的性能速度，请将文件存储在 WSL 文件系统中，前提是使用 Linux 工具在 Linux 命令行（Ubuntu、OpenSUSE 等）中处理这些文件。 如果是使用 Windows 工具在 Windows 命令行（PowerShell、命令提示符）中工作，请将文件存储在 Windows 文件系统中。 可以跨操作系统访问文件，但这可能会显著降低性能。

例如，在存储 WSL 项目文件时：

- 使用 Linux 文件系统根目录：`\\wsl$\<DistroName>\home\<UserName>\Project`
- 不是 Windows 文件系统的根目录：`C:\Users\<UserName>\Project` 或 `/mnt/c/Users/<UserName>/Project$`



设置你最喜欢的代码编辑器

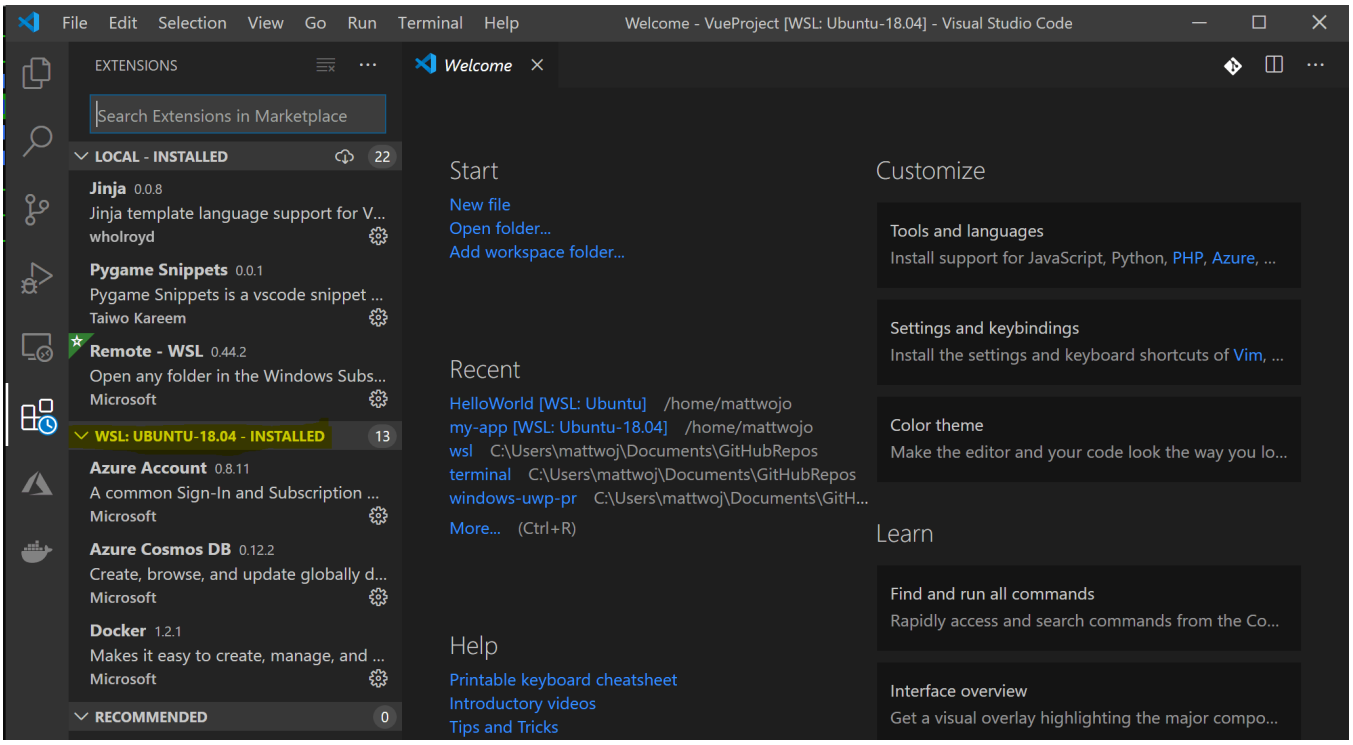
建议使用 Visual Studio Code 或 Visual Studio，因为它们直接支持使用 WSL 进行远程开发和调试。 Visual Studio Code 使你能够将 WSL 用作功能完备的开发环境。 Visual Studio 提供了对 C++ 跨平台开发的原生 WSL 支持。

使用 Visual Studio Code

按照此分步指南开始在 WSL 中使用 Visual Studio Code，其中包括安装远程开发扩展包。 使用此扩展，能够运行 WSL、SSH 或开发容器，以使用整套 Visual Studio Code 功能进行编辑和调试。 在不同的独立开发环境之间快速切换并进行更新，而无需担心会影响本地计算机。

安装并设置 VS Code 后，可以通过输入以下内容使用 VS Code 远程服务器打开 WSL 项目：`code .`

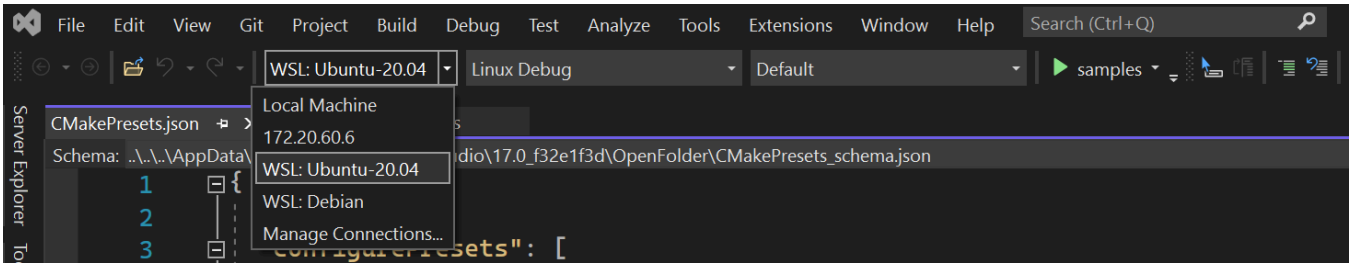
请确保在命令的末尾添加句点以打开当前目录。



VS Code 显示的 WSL 扩展

使用 Visual Studio

按照此分步指南开始将 Visual Studio 与 WSL 一起用于 C++ 跨平台开发。 Visual Studio 2022 使你能够从 Visual Studio 的同一实例在 Windows、WSL 发行版和 SSH 连接上生成和调试 CMake 项目。



在 Visual Studio 2022 中选择目标系统

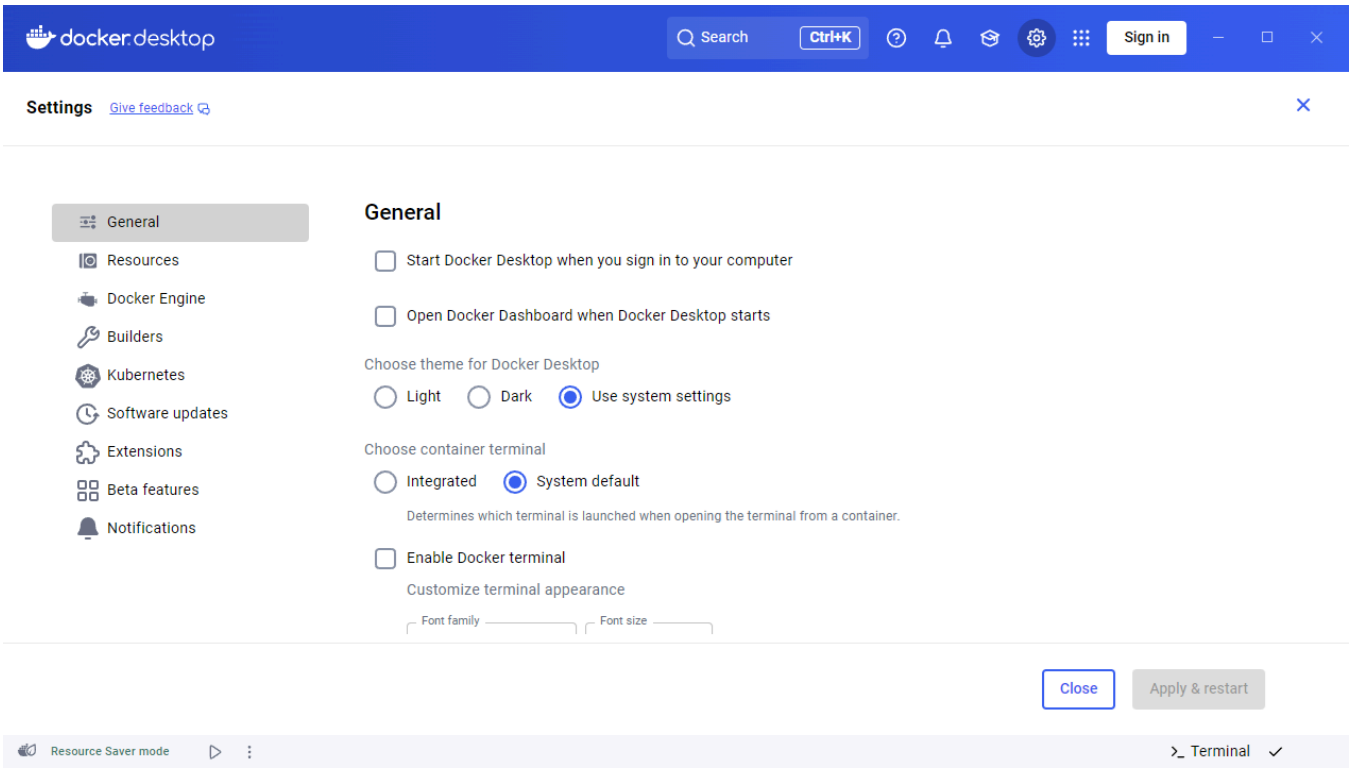
使用 Git 设置版本管理

按照此分步指南开始在 WSL 上使用 Git，并将项目连接到 Git 版本控制系统，同时使用凭据管理器进行身份验证，使用 Git Ignore 文件，了解 Git 行尾，以及使用内置到 VS Code 的 Git 命令。

在命令行中显示 git 版本

使用 Docker 设置远程开发容器

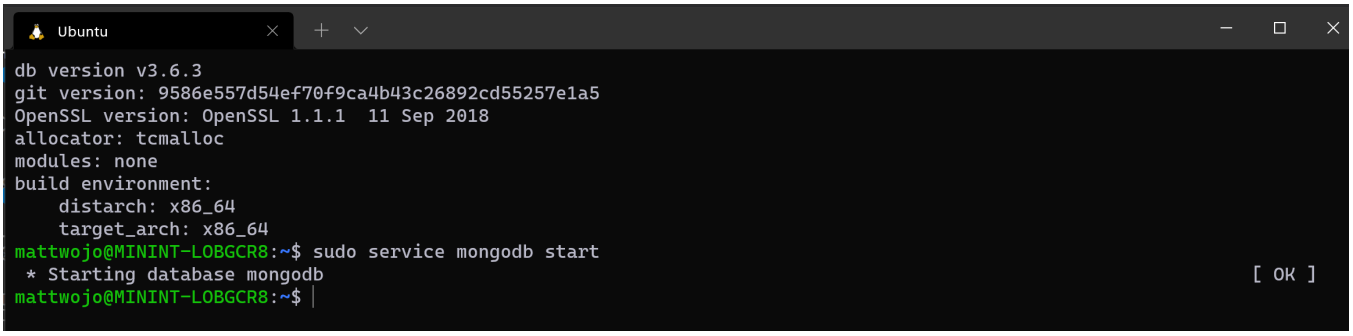
开始按照此分步指南来使用 WSL 2 上的 Docker 远程容器，并通过 Windows 的 Docker Desktop 将您的项目连接到远程开发容器。



Docker Desktop 屏幕截图

设置数据库

按照这份分步指南开始使用 WSL 上的数据库，并将您的项目连接到 WSL 环境中的数据库。 开始使用 MySQL、PostgreSQL、MongoDB、Redis、Microsoft SQL Server 或 SQLite。



在 Ubuntu 中通过 WSL 运行 MongoDB

设置 GPU 加速以提高性能

按照这份分步指南，在 WSL 中设置 GPU 加速的机器学习训练，并利用计算机的 GPU（图形处理单元）来加速繁重的性能工作负载。

使用 WSL 进行 GPU 加速运行

基本 WSL 命令

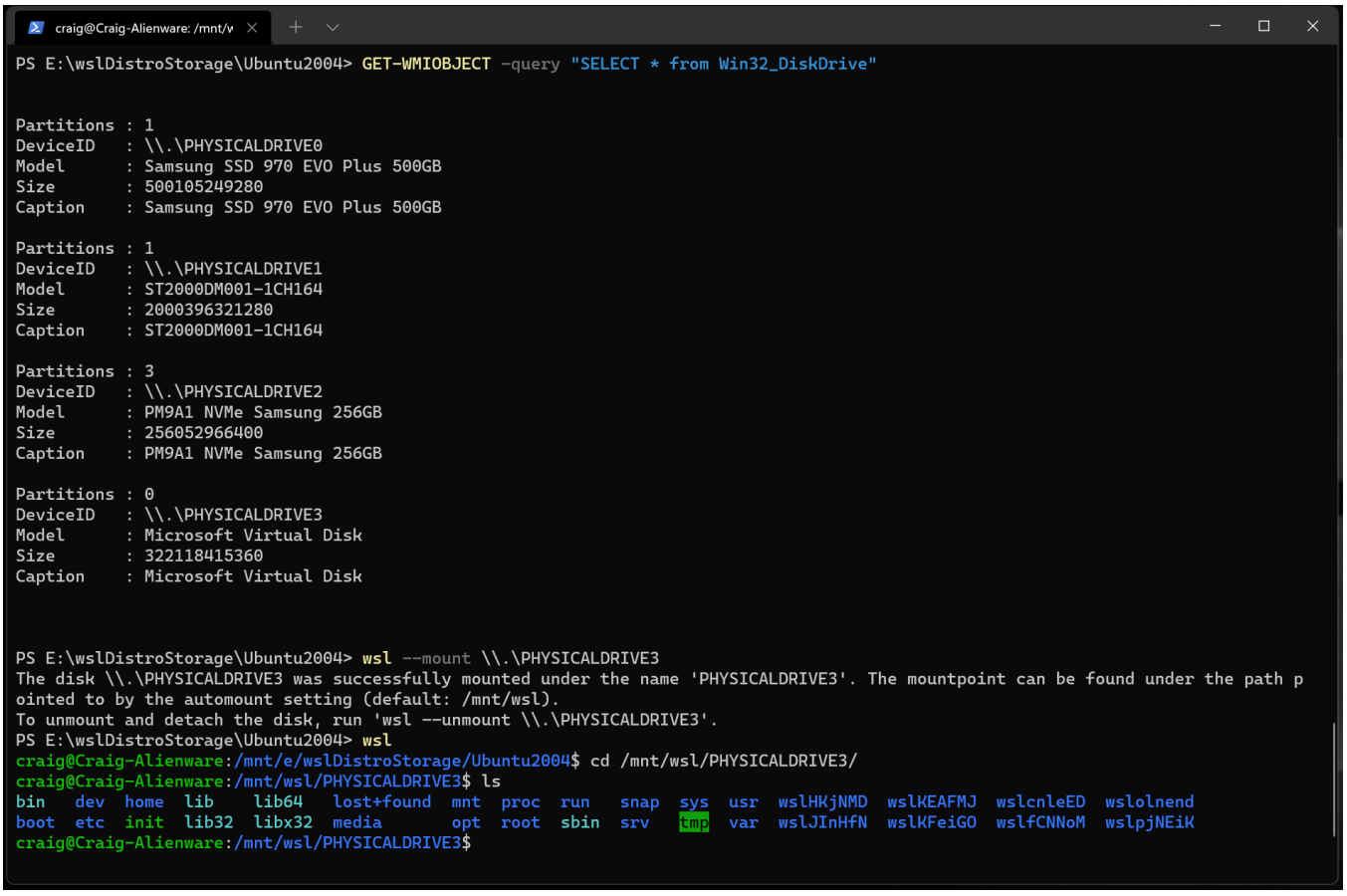
通过 WSL 安装的 Linux 发行版最好使用 PowerShell 或 Windows 命令提示符 (CMD) 进行管理。有关使用 WSL 时需要熟悉的基本命令的列表，请参阅 WSL 命令参考指南。

此外，许多命令在 Windows 和 Linux 之间都具有互操作性。 下面是几个示例：

- 从 Windows 命令行运行 Linux 工具：打开 PowerShell，通过输入以下内容使用 Linux `C:\temp>` 命令显示 `ls -la` 的目录内容：`wsl ls -la`
- 混合 Linux 和 Windows 命令：在此示例中，使用 Linux 命令 `ls -la` 列出目录中的文件，然后使用 PowerShell 命令 `findstr` 筛选包含“git”的单词的结果：`wsl ls -la | findstr "git"`。 这还可以通过混合使用 Windows `dir` 命令和 Linux `grep` 命令来实现：`dir | wsl grep git`。
- 直接从 WSL 命令行运行 Windows 工具：`<tool-name>.exe`。例如，若要打开 .bashrc 文件（启动 Linux 命令行时运行的 shell 脚本），请输入：`notepad.exe .bashrc`
- 使用 Linux Grep 工具运行 Windows ipconfig.exe 工具：从 Bash 输入命令 `ipconfig.exe | grep IPv4 | cut -d: -f2` 或从 PowerShell 输入 `ipconfig.exe | wsl grep IPv4 | wsl cut -d: -f2`。此示例演示了 Windows 文件系统上的 ipconfig 工具，该工具先是用于显示当前 TCP/IP 网络配置值，然后通过 Linux 工具 grep 被筛选为仅显示 IPv4 结果。

装载外部驱动器或 USB 设备

按照此分步指南开始在 WSL 2 中装载 Linux 磁盘。



wsl 装载命令屏幕截图

运行 Linux GUI 应用

按照本教程了解如何在 WSL 上设置和运行 Linux GUI 应用。

其他资源

- 设置开发环境：了解有关为您首选的语言或框架（如 React、Python、NodeJS、Vue 等）设置开发环境的详细信息。
- 疑难解答：查找常见问题、在何处报告错误、在何处请求新功能以及如何参与文档编撰工作。
- 常见问题解答：查找常见问题列表。
- 发行说明：查看 WSL 发行说明，了解过去版本更新的历史记录。 还可以查找 WSL Linux 内核的发行说明。

Last updated on 2025/03/31

开始将 Visual Studio Code 与适用于 Linux 的 Windows 子系统配合使用

Visual Studio Code 以及 WSL 扩展使你可以直接从 VS Code 使用 WSL 作为全职开发环境。 您可以：

- 在基于 Linux 的环境中进行开发
- 使用特定于 Linux 的工具链和实用工具
- 在 Windows 上运行和调试基于 Linux 的应用程序，同时保持对 Outlook 和 Office 等生产力工具的访问权限。
- 使用 VS Code 内置终端运行所选 Linux 分发版
- 利用 VS Code 功能，例如 Intellisense 代码完成、 linting、 调试支持、 代码片段和 单元测试
- 使用 VS Code 的内置 Git 支持轻松管理版本控制
- 直接在 WSL 项目上运行命令和 VS Code 扩展
- 在 Linux 或装载的 Windows 文件系统（例如 `/mnt/c`）中编辑文件，而无需担心路径问题、二进制兼容性或其他跨 OS 挑战

安装 VS Code 和 WSL 扩展

- 访问 VS Code 安装页，并选择当前系统体系结构的 **Windows 安装程序** 。 在 Windows 上安装 Visual Studio Code（不在 WSL 文件系统中）。
- 当系统提示在安装过程中 **选择其他任务** 时，请务必选中 “**添加到 PATH** ”选项，以便使用代码命令轻松在 WSL 中打开文件夹。
- 安装 远程开发扩展包。 除了远程 - SSH 和开发容器扩展之外，此扩展包还包括 WSL 扩展，使你能够在容器、远程计算机上或 WSL 中打开任何文件夹。

> **重要**
>
> 若要安装 WSL 扩展，需要 **1.35 年 5 月版本** 或更高版本的 VS Code。 不建议在没有 WSL 扩展的情况下在 VS Code 中使用 WSL，因为你将失去对自动完成、调试、linting 等的支持。有趣的事实：此 WSL 扩展安装在 $HOME/.vscode/extensions（在 PowerShell 中输入命令 `ls $HOME\.vscode\extensions\` ）。

更新 Linux 分发版

某些 WSL Linux 分发版缺少 VS Code 服务器启动所需的库。 可以使用其包管理器将其他库添加到 Linux 分发版中。

例如，若要更新 Debian 或 Ubuntu，请使用：

```Bash
sudo apt-get update
```

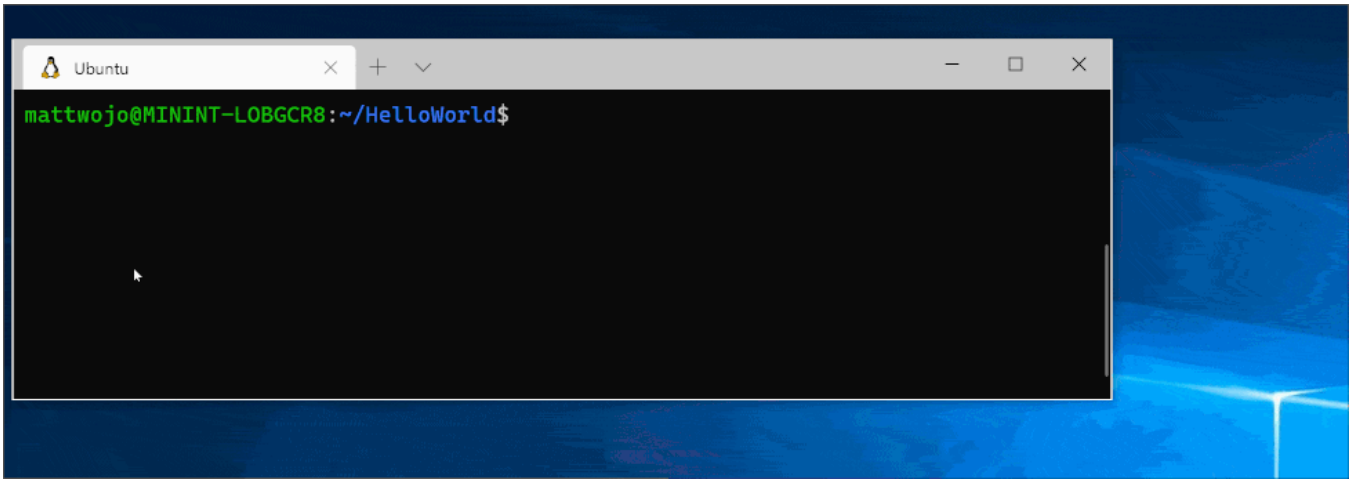
若要添加 wget（要从 Web 服务器检索内容）和 ca-certificates（若要允许基于 SSL 的应用程序检查 SSL 连接的真实性），请输入：

```Bash
sudo apt-get install wget ca-certificates
```

在 Visual Studio Code 中打开 WSL 项目

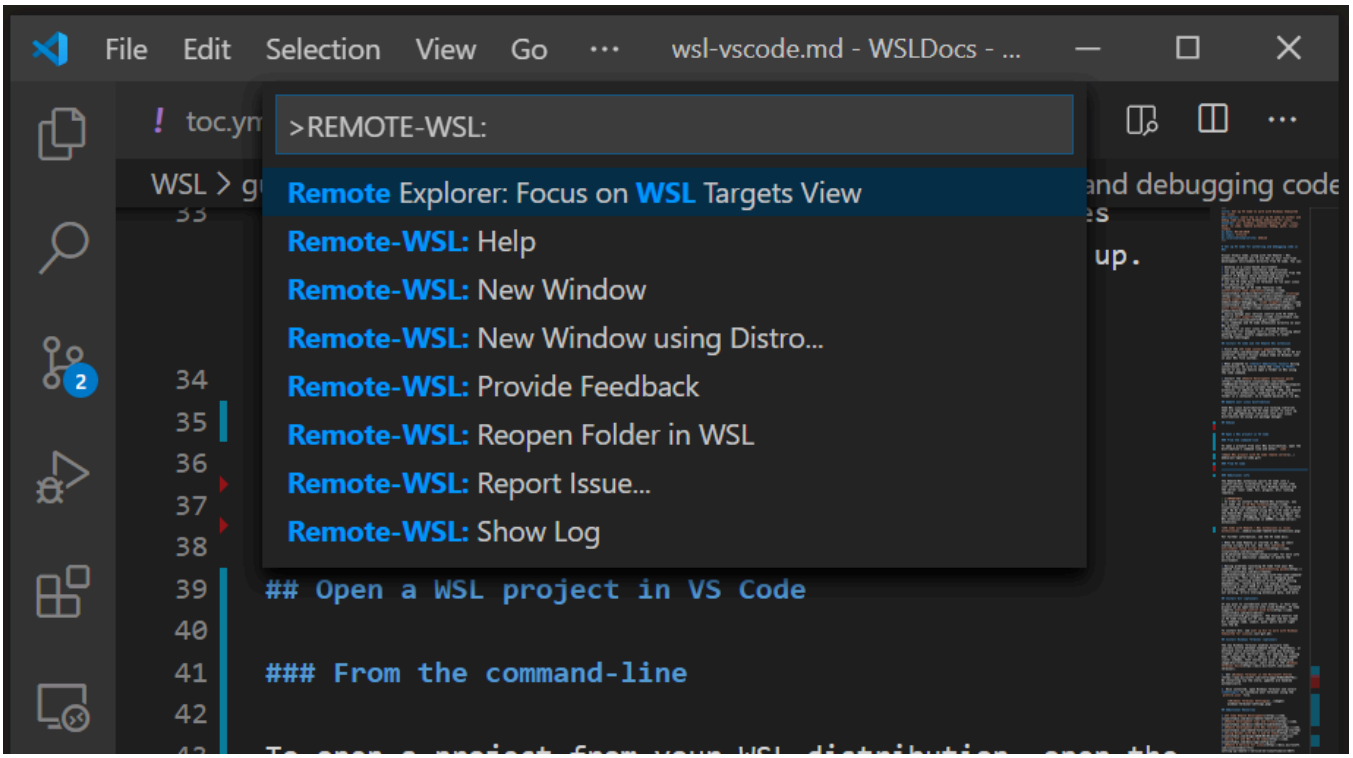
从命令行

若要从 WSL 分发版打开项目，请打开分发的命令行并输入： `code .`



从 VS Code 中

还可以使用快捷方式访问更多 VS Code WSL 选项：VS Code 中的 **Ctrl** + **Shift** + **P** 来打开命令面板。 如果键入 `WSL` ，你将看到可用的选项列表，允许你在 WSL 会话中重新打开文件夹，指定要在其中打开哪个分发版，等等。



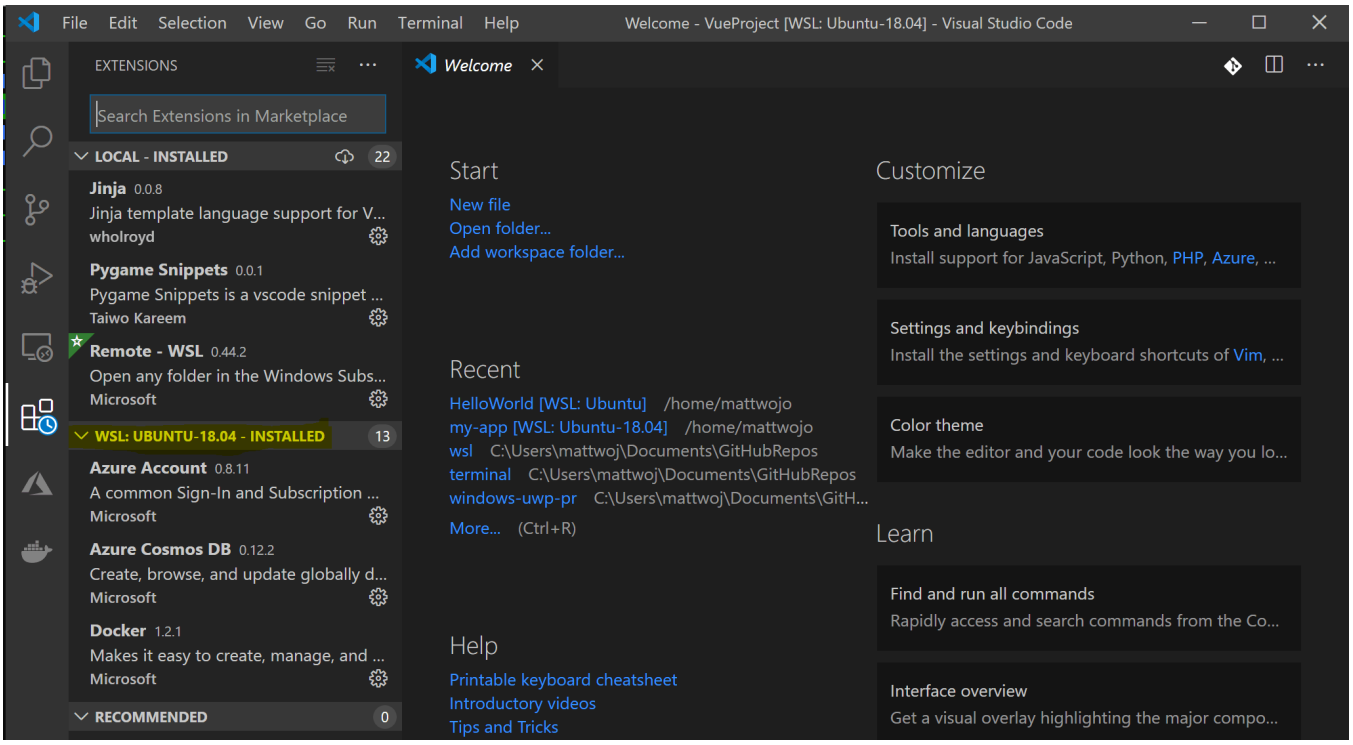
VS Code WSL 中的扩展

WSL 扩展将 VS Code 拆分为“客户端-服务器”体系结构，客户端（用户界面）在 Windows 计算机上运行，服务器（代码、Git、插件等）在 WSL 分发版中“远程”运行。

运行 WSL 扩展时，选择“扩展”选项卡将显示本地计算机与 WSL 分发版之间拆分的扩展列表。

安装本地扩展（如 主题）只需安装一次。

某些扩展（如 Python 扩展）或用于处理代码检查或调试任务的任何扩展必须单独安装在每个 WSL 发行版上。 如果本地安装了未安装在 WSL 分发版上的扩展，VS Code 将显示警告图标 ⚠ 以及绿色的“在 WSL 中安装”按钮。



有关详细信息，请参阅 VS Code 文档：

- 在 WSL 中启动 VS Code 时，不会运行 shell 启动脚本。 有关如何运行其他命令或修改环境的详细信息，请参阅此 高级环境设置脚本文章 。
- 从 WSL 命令行启动 VS Code 时遇到问题？ 本 故障排除指南 包括有关更改路径变量、解决有关缺少依赖项的扩展错误、解决 Git 行结束问题、在远程计算机上安装本地 VSIX、启动浏览器窗口、阻止器 localhost 端口、Web 套接字不起作用、存储扩展数据的错误等的提示。

安装 Git （可选）

如果计划与他人协作，或在开源站点（如 GitHub）上托管项目，VS Code 支持使用 Git版本控制。 VS Code 中的“源代码管理”选项卡跟踪所有更改，并具有内置于 UI 中的常见 Git 命令（添加、提交、推送、拉取）。

若要安装 Git，请参阅 设置 Git 以使用适用于 Linux 的 Windows 子系统。

安装 Windows 终端（可选）

新的 Windows 终端支持多个选项卡（在命令提示符、PowerShell 或多个 Linux 分发版之间快速切换）、自定义键绑定（为打开或关闭选项卡、复制+粘贴等创建自己的快捷键）、表情符号 ☺ 和自定义主题（配色方案、字体样式和大小、背景图像/模糊/透明度）。 在 Windows 终端文档 中了解更多信息。

1. 在 Microsoft 应用商店中获取 Windows 终端：通过应用商店安装，会自动处理更新。

2. 安装后，打开 Windows 终端并选择 “**设置**” 以使用 `profile.json` 该文件自定义终端。

其他资源

- VS Code WSL 文档
- VS Code WSL 教程
- 远程开发提示和技巧
- 将 Docker 与 WSL 2 和 VS Code 配合使用
- 在 VS Code 中使用 C++ 和 WSL
- 适用于 Linux 的远程 R 服务

可能需要考虑的一些附加扩展包括：

- 来自其他编辑器的键映射：如果你从另一个文本编辑器（如 Atom、Sublime、Vim、Emacs、Notepad++等）转换，这些扩展可以帮助你的环境更适应你的使用习惯。
- 设置同步：使你能够使用 GitHub 跨不同安装同步 VS Code 设置。 如果你在不同的计算机上工作，这有助于在这些计算机之间保持你的工作环境一致。

Last updated on 2025/06/10

在适用于 Linux 的 Windows 子系统上使用 Git 入门

Git 是最常用的版本控制系统。 借助 Git，可以跟踪对文件的更改，以便记录已完成的作，并能够在需要的情况下还原到早期版本的文件。 Git 还可以简化协作，允许多人更改全部合并到一个源中。

Git 可以安装在 Windows 和 WSL 上

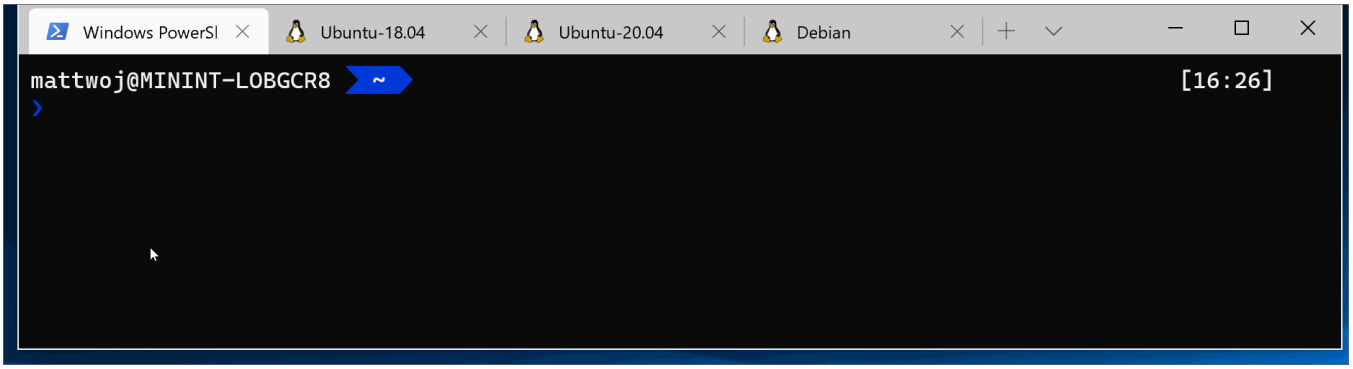
一个重要注意事项：启用 WSL 并安装 Linux 分发版时，将安装一个新的文件系统，并将其与计算机上的 Windows NTFS C：\ 驱动器分开。 在 Linux 系统中，驱动器不会被分配字母。 将为它们提供装载点。 在 WSL 的情况下，您的文件系统根 `/` 是您的根分区或者文件夹的挂载点。 `/` 下的所有内容并非都是同一个驱动器。 例如，在笔记本电脑上，我安装了两个版本的 Ubuntu（20.04 和 18.04），以及 Debian。 如果打开这些分发版，请使用命令 `cd ~`选择主目录，然后输入命令 `explorer.exe .`，Windows 文件资源管理器将打开并显示该分发的目录路径。

展开表

Linux 发行版和 Windows	用于访问主文件夹的 Windows 路径
Ubuntu 20.04	`\\wsl$\Ubuntu-20.04\home\username`
Ubuntu 18.04	`\\wsl$\Ubuntu-18.04\home\username`
Debian	`\\wsl$\Debian\home\username`
Windows PowerShell	`C:\Users\username`

> **提示**
>
> 如果要从 WSL 分发命令行访问 Windows 文件目录，而不是 `C:\Users\username`，将使用 `/mnt/c/Users/username`访问该目录，因为 Linux 分发版将 Windows 文件系统视为装载的驱动器。

需要在每个要与之一起使用的文件系统上安装 Git。



显示 Git 版本

安装 Git

Git 已随适用于 Linux 分发版的大多数 Windows 子系统一起安装，但可能需要更新到最新版本。 还需要设置 git 配置文件。

若要安装 Git，请参阅适用于 Linux 站点的 Git 下载。 每个 Linux 分发版都有自己的包管理器并安装命令。

对于 Ubuntu/Debian 中最新的稳定 Git 版本，请输入以下命令：

Bash

```
sudo apt-get install git
```

> **注意**
>
> 如果尚未**安装适用于 Windows 的 Git**，可能需要进行安装。

Git 配置文件设置

若要设置 Git 配置文件，请为正在使用的分发打开命令行，并使用此命令设置名称（将“你的名称”替换为首选用户名）：

Bash

```
git config --global user.name "Your Name"
```

使用此命令设置电子邮件（将“`youremail@domain.com`”替换为你喜欢的电子邮件）：

Bash

```
git config --global user.email "youremail@domain.com"
```

> **提示**
>
> 如果还没有 GitHub 帐户，可以在 GitHub 上注册一个帐户。 如果你以前从未使用过 Git，GitHub Guides 可以帮助你入门。 如果需要编辑 Git 配置，可以使用内置的文本编辑器（如 nano：`nano ~/.gitconfig`）执行此作。

建议 使用双重身份验证（2FA） 保护帐户。

Git 凭据管理器设置

Git 凭据管理器（GCM） 是一个安全的 Git 凭据帮助程序，构建在 .NET 上，可用于 WSL1 和 WSL2。 它为 GitHub 存储库、Azure DevOps、Azure DevOps Server 和 Bitbucket 启用多重身份验证支持。

GCM 集成到 GitHub 等服务的身份验证流中，在向托管提供商进行身份验证后，请求新的身份验证令牌。 然后，它将令牌安全地存储在 Windows 凭据管理器 中。 第一次之后，可以使用 Git 与托管提供商通信，而无需重新进行身份验证。 它将仅访问 Windows 凭据管理器中的令牌。

若要将 GCM 与 WSL 配合使用，必须位于 Windows 10 版本 1903 或更高版本上。 这是第一个包含 GCM 在您的 WSL 发行版中与 Git 互操作所需的 `wsl.exe` 工具的 Windows 版本。

建议安装 最新的 Git for Windows，以便在 WSL 和 Windows 主机之间共享凭据 & 设置。 Git 凭据管理器包含在 Git for Windows 中，最新版本包含在每个新的 Git for Windows 版本中。 在安装过程中，系统会要求你选择凭据帮助程序，并将 GCM 设置为默认值。

如果你有理由不安装适用于 Windows 的 Git，则可以在 WSL 分发版中将 GCM 作为 Linux 应用程序直接安装，但请注意，这样做意味着 GCM 作为 Linux 应用程序运行，并且不能利用主机 Windows作系统的身份验证或凭据存储功能。 有关如何在没有适用于 Windows 的 Git 的情况下配置 WSL 的说明，请参阅 GCM 存储库。

用于检查和设置 WSL 的 GCM 的命令

如果已安装适用于 Windows 的 Git，则已安装并配置 GCM。 可以通过在 WSL 分发版中运行以下命令来验证这一点：

PowerShell

```
git --version; git credential-manager --version
```

> **注意**

使用 GCM 作为 WSL Git 安装的凭据帮助程序意味着 WSL Git 中的任何配置集都不受 GCM（默认情况下）的尊重。 这是因为 GCM 作为 Windows 应用程序运行，因此将使用 Git for Windows 安装来查询配置。 这意味着需要在 Git for Windows 和 WSL Git 中设置 GCM 的代理设置等内容，因为它们存储在不同的文件中（`%UserProfile%\.gitconfig` 与 `\\wsl$\distro\home\$USER\.gitconfig`）。 你可以配置 WSL，以便 GCM 将使用 WSL Git 配置，但这意味着代理设置对特定 WSL 安装是唯一的，不会与其他或 Windows 主机共享。

> **⚠ 警告**
>
> **GitHub CLI 中当前存在使用 keyring 的问题**。

将 Git 与 SSH 搭配使用

Git 凭据管理器仅适用于 HTTP(S) 远程。 仍可将 Git 与 SSH 配合使用：

- Azure DevOps SSH
- GitHub SSH
- Bitbucket SSH

Azure 的其他配置

如果要使用 Azure Repos 或 Azure DevOps，则需要进行一些其他配置：

Bash

```
git config --global credential.https://dev.azure.com.useHttpPath true
```

现在，您在 WSL 发行版中执行的任何 git 操作都将使用 GCM。 如果您已经为某个主机系统缓存了凭据，系统将从凭据管理器中访问它们。 如果没有，即使你位于 Linux 控制台中，你也会收到请求凭据的对话框响应。

> **提示**
>
> 如果使用 GPG 密钥来确保代码签名的安全性，可能需要**将 GPG 密钥与 GitHub 电子邮件相关联**。

添加 Git Ignore 文件

我们建议向项目添加 .gitignore 文件。 GitHub 提供了一系列有用的 .gitignore 模板，其中包含根据你的用例组织推荐的 .gitignore 文件设置。 例如，下面是 gitHub 的默认 gitignore 模板，用于 Node.js 项目。

如果选择 使用 GitHub 网站 创建新存储库，可以选中复选框以初始化存储库，包括添加一个 README 文件、根据特定项目类型设置的 .gitignore 文件，以及必要时添加许可证的选项。

Git 和 VS Code

Visual Studio Code 附带了对 Git 的内置支持，包括源代码管理选项卡，用于显示更改并处理各种 git 命令。 详细了解 VS Code 的 Git 支持。

Git 行尾

如果在 Windows、WSL 或容器之间使用相同的存储库文件夹，请务必设置一致的行尾。

由于 Windows 和 Linux 使用不同的默认行尾，Git 可能会报告大量文件被修改，但这些文件除了行尾之外没有其他差异。 若要防止发生这种情况，可以使用 `.gitattributes` 文件或在 Windows 端全局禁用行结束转换。 请参阅此 VS Code 文档，了解如何解决 git 行结束问题。

其他资源

- WSL 和 VS Code
- GitHub Learning Lab：联机课程
- Git 可视化工具
- Git 工具 - 凭据存储

Last updated on 2025/10/04

适用于 Linux 的 Windows 子系统上的数据库入门

本分步指南将帮助你开始将 WSL 中的项目连接到数据库。 开始使用 MySQL、PostgreSQL、MongoDB、Redis、Microsoft SQL Server 或 SQLite。

先决条件

- 运行 Windows 11 或 Windows 10，更新至版本号 2004，**编译号 19041**或更高。
- 使用 WSL 安装 Linux 分发版，并创建 Linux 用户名和密码。

数据库系统之间的差异

数据库系统的一些 常用选项 包括：

- MySQL （SQL）
- PostgreSQL （SQL）
- Microsoft SQL Server （SQL）
- SQLite （SQL）
- MongoDB （NoSQL）
- Redis （NoSQL）

MySQL 是一个开源 SQL 关系数据库，将数据组织成一个或多个表，其中数据类型可能相互关联。 它可垂直缩放，这意味着一台最终计算机将为你完成工作。 它目前是四个数据库系统中最广泛使用的。

PostgreSQL（有时称为 Postgres）也是一个开源 SQL 关系数据库，强调扩展性和标准符合性。它现在也可以处理 JSON，但通常更适合结构化数据、垂直缩放和符合 ACID 的需求，例如电子商务和金融交易。

Microsoft SQL Server 包括 Windows 上的 SQL Server、Linux 上的 SQL Server 和 Azure 上的 SQL。 这些也是在服务器上设置的关系数据库管理系统，其主要功能是存储和检索软件应用程序请求的数据。

SQLite 是一个开源自包含的、基于文件的"无服务器"数据库，以其可移植性、可靠性和高性能而闻名，即使在低内存环境中也是如此。

MongoDB 是一个开源 NoSQL 文档数据库，旨在处理 JSON 并存储无架构数据。 它可水平缩放，这意味着多个较小的计算机将为你完成工作。 它非常适合实现灵活性，处理非结构化数据，并缓存实时分析。

Redis 是一种开源 NoSQL 内存中数据存储。 它使用键值对进行存储而不是文档。

安装 MySQL

若要在 WSL 上运行的 Linux 分发版上安装 MySQL，只需按照 MySQL 文档中的 Linux 上安装 MySQL 说明进行作。可能需要首先在配置文件中`wsl.conf`。

使用 Ubuntu 分发的示例：

1. 打开 Ubuntu 命令行并更新可用的包：

   ```Bash
   sudo apt update
   ```

2. 更新包后，使用以下命令安装 MySQL：

   ```Bash
   sudo apt install mysql-server
   ```

3. 确认安装并获取版本号：

   ```Bash
   mysql --version
   ```

4. 启动 MySQL 服务器/检查状态：

   ```Bash
   systemctl status mysql
   ```

5. 若要打开 MySQL 提示符，请输入：

   ```Bash
   sudo mysql
   ```

6. 若要查看可用的数据库，请在 MySQL 提示符下输入：

   ```Bash
   SHOW DATABASES;
   ```

7. 若要创建新数据库，请输入：

```bash
CREATE DATABASE database_name;
```

8. 若要删除数据库，请输入：

```bash
DROP DATABASE database_name;
```

有关使用 MySQL 数据库的详细信息，请参阅 MySQL 文档。

若要在 VS Code 中使用 MySQL 数据库，请尝试 MySQL 扩展。

可能还需要运行包含的安全脚本。 这会更改远程根登录和示例用户等一些不太安全的默认选项。 此脚本还包括更改 MySQL 根用户密码的步骤。 运行安全脚本：

1. 启动 MySQL 服务器：

```bash
sudo service mysql start
```

2. 启动安全脚本提示：

```bash
sudo mysql_secure_installation
```

3. 第一个提示将询问是否要设置 VALIDATE PASSWORD COMPONENT，该组件可用于测试 MySQL 密码强度。 如果要设置一些简单密码，则不应设置此组件。

4. 然后，将为 MySQL 根用户设置/更改密码，决定是否删除匿名用户，决定是否允许根用户在本地和远程登录，决定是否删除测试数据库，最后决定是否立即重新加载特权表。

安装 PostgreSQL

在 WSL 上安装 PostgreSQL（即Ubuntu）：

1. 打开 WSL 终端（即Ubuntu）。

2. 更新 Ubuntu 包：

```bash
```

```
sudo apt update
```

3. 更新包后，请安装 PostgreSQL（以及具有一些有用实用工具的 PostgreSQL Contrib 包）：
<

```Bash
sudo apt install postgresql postgresql-contrib
```

4. 确认安装并获取版本号：

```Bash
psql --version
```

安装 PostgreSQL 后，需要知道 3 个命令：

- 检查数据库状态

```Bash
sudo service postgresql status
```

- 启动数据库

```Bash
sudo service postgresql start
```

- 停止数据库

```Bash
sudo service postgresql stop
```

默认管理员用户 `postgres`需要分配密码才能连接到数据库。 设置密码：

1. 输入以下命令：

```Bash
sudo passwd postgres
```

2. 系统会提示输入新密码。

3. 关闭并重新打开终端。

若要使用 psql shell 运行 PostgreSQL，请执行以下作：

1. 启动 postgres 服务：

```Bash
sudo service postgresql start
```

2. 连接到 postgres 服务并打开 psql shell：

```Bash
sudo -u postgres psql
```

成功输入 psql shell 后，会看到命令行更改如下所示：

```psqlshell
postgres=#
```

> **注意**
>
> 或者，可以通过使用以下命令切换到 postgres 用户 `su - postgres` ，然后输入以下命令 `psql` 来打开 psql shell。

若要退出 postgres=# Enter： `\q` 或使用快捷键：Ctrl+D

若要查看 PostgreSQL 安装上已创建的用户帐户，请使用 WSL 终端中的命令：`psql --command="\du"` ...或如果已打开 psql shell 仅使用 `\du`。 此命令将显示列：帐户用户名、角色属性列表和角色组的成员。 若要退出命令行，请输入： `q`

有关使用 PostgreSQL 数据库的详细信息，请参阅 PostgreSQL 文档。

若要在 VS Code 中使用 PostgreSQL 数据库，请尝试 PostgreSQL 扩展。

安装 MongoDB

若要安装 MongoDB，请参阅 Mongodb 文档： 在 Linux 上安装 MongoDB 社区版

安装 MongoDB 可能需要略有不同的步骤，具体取决于用于安装的 Linux 分发版。 另请注意，MongoDB 安装可能因要安装的版本 #而异。 使用 MongoDB 文档左上角的版本下拉列表选择与

目标相符的版本。 最后，可能需要在与 WSL 一起使用的 Linux 分发版配置文件中`wsl.conf`。 该`systemctl`命令是 systemd init 系统的一部分，如果分发使用的是 systemv，则可能无法正常工作。

VS Code 支持通过 Azure CosmosDB 扩展使用 MongoDB 数据库，可以从 VS Code 中创建、管理和查询 MongoDB 数据库。 若要了解详细信息，请访问 VS Code 文档： 使用 MongoDB。

在 MongoDB 文档中了解详细信息：

- 使用 MongoDB 简介
- 创建用户
- CRUD：创建、读取、更新、删除
- 参考文档

安装 Microsoft SQL Server

快速入门：在适用于 Linux 的 Windows 子系统上安装 SQL Server 并创建数据库（WSL 2）。

安装 SQLite

在 WSL 上安装 SQLite（即Ubuntu）：

1. 打开 WSL 终端（即Ubuntu）。
2. 更新 Ubuntu 包：

   ```Bash
   sudo apt update
   ```

3. 包更新后，使用以下命令安装 SQLite3：

   ```Bash
   sudo apt install sqlite3
   ```

4. 确认安装并获取版本号：

   ```Bash
   sqlite3 --version
   ```

若要创建名为“example.db”的测试数据库，请输入：

```bash
sqlite3 example.db
```

若要查看 SQLite 数据库的列表，请输入： `.databases`

若要查看数据库的状态，请输入： `.dbinfo ?DB?`

创建后，数据库将为空。 可以使用 `CREATE TABLE empty (kol INTEGER);` 为数据库创建新表。

现在，输入 `.dbinfo ?DB?` 将显示已创建的数据库。

若要退出 SQLite 提示符，请输入： `.exit`

有关使用 SQLite 数据库的详细信息，请参阅 SQLite 文档。

若要在 VS Code 中使用 SQLite 数据库，请尝试 SQLite 扩展。

安装 Redis

在 WSL 上安装 Redis（即Ubuntu）：

1. 打开 WSL 终端（即Ubuntu）。
2. 更新 Ubuntu 包：

   ```bash
   sudo apt update
   ```

3. 更新包后，使用以下命令安装 Redis：

   ```bash
   sudo apt install redis-server
   ```

4. 确认安装并获取版本号：

   ```bash
   redis-server --version
   ```

开始运行 Redis 服务器：

```bash
```

```
sudo service redis-server start
```

检查 redis 是否正常工作（redis-cli 是命令行接口实用工具，用于与 Redis 通信）：

```bahs
redis-cli ping
```

这应返回“PONG”的答复。

停止运行 Redis 服务器：

```Bash
sudo service redis-server stop
```

有关使用 Redis 数据库的详细信息，请参阅 Redis 文档。

若要在 VS Code 中使用 Redis 数据库，请尝试 Redis 扩展。

查看正在运行的服务并设置配置文件别名

若要查看当前在 WSL 分发版上运行的服务，请输入：

```Bash
service --status-all
```

键入`sudo service mongodb start`或`sudo service postgres start`和`sudo -u postgrest psql`可能会很繁琐。 但是，可以考虑在 WSL 的 `.profile` 文件中设置别名，使这些命令更快使用，且更容易记住。

若要设置自己的自定义别名或快捷方式，用于执行以下命令：

1. 打开 WSL 终端并输入 `cd ~` ，确保你位于根目录中。
2. `.profile`使用终端文本编辑器 Nano 打开用于控制终端设置的文件：

   ```Bash
   sudo nano .profile
   ```

3. 在文件底部（请勿更改 `# set PATH` 设置），添加以下内容：

Bash

```
# My Aliases
alias start-pg='sudo service postgresql start'
alias run-pg='sudo -u postgres psql'
```

这将允许你输入 `start-pg` 以开始运行 postgresql 服务并 `run-pg` 打开 psql shell。 你可以将 `start-pg` 和 `run-pg` 更改为任何你想要的名称，只需注意不要覆盖 PostgreSQL 已经在使用的命令！

4. 添加新别名后，使用 **Ctrl+X** 退出 Nano 文本编辑器 -- 在系统提示保存并输入时选择 `Y`（是）（将文件名保留为 `.profile`）。

5. 关闭并重新打开 WSL 终端，然后尝试新的别名命令。

故障排除

错误：目录同步 fdatasync 参数无效

确保以 WSL 2 模式运行 Linux 分发版。 有关从 WSL 1 切换到 WSL 2 的帮助，请参阅 将分发版本设置为 WSL 1 或 WSL 2。

其他资源

- 在 Windows 上设置开发环境

Last updated on 2025/08/15

WSL 2 上的 Docker 远程容器入门

本分步指南将帮助你开始使用远程容器进行开发，方法是 **使用 WSL 2 设置适用于 Windows 的 Docker Desktop** （适用于 Linux 的 Windows 子系统，版本 2）。

适用于 Windows 的 Docker Desktop 提供用于生成、传送和运行 dockerized 应用的开发环境。 通过启用基于 WSL 2 的引擎，可以在同一台计算机上运行 Docker Desktop 中的 Linux 和 Windows 容器。 （Docker Desktop 免费供个人使用，小型企业，有关 Pro、Team 或 Business 定价的信息，请参阅 Docker 站点常见问题解答）。

> **注意**
>
> 建议使用 Docker Desktop，因为它 **与适用于 Linux 的 Windows 和 Windows 子系统集成**。 但是，虽然 Docker Desktop 支持同时运行 Linux 和 Windows 容器，但 **不能** 同时运行这两个容器。 若要同时运行 Linux 和 Windows 容器，需要在 WSL 中安装和运行单独的 Docker 实例。 如果需要同时运行容器，或者只想直接在 Linux 分发版中安装容器引擎，请遵循该容器服务的 Linux 安装说明，例如在 **Ubuntu 上安装 Docker 引擎** 或 **安装 Podman 以运行 Linux 容器**。

Docker 容器概述

Docker 是一种工具，用于使用容器创建、部署和运行应用程序。 使用容器，开发人员可以使用它所需的所有部件（库、框架、依赖项等）打包应用，并将其作为一个包交付。 使用容器可确保无论运行应用的任何自定义设置或以前安装的库都运行相同，而该库可能与用于编写和测试应用代码的计算机不同。 这样，开发人员就可以专注于编写代码，而无需担心代码将运行的系统。

Docker 容器类似于虚拟机，但不创建整个虚拟作系统。 相反，Docker 使应用能够使用与运行它的系统相同的 Linux 内核。 这允许应用包仅下载主计算机上尚未存在的部件，减少包大小并提高性能。

容器能够保持持续可用性的一个重要原因是通过使用 Docker 容器与 Kubernetes 这样的工具配合使用。 这样，就可以在不同时间创建应用容器的多个版本。 无需关闭整个系统进行更新或维护，而是可以即时替换每个容器（及其特定的微服务）。 可以使用所有的更新内容准备新容器，为生产环境配置容器，并在容器准备就绪后只需指向新的容器。 还可以使用容器存档您的应用的不同版本，并根据需要将其作为安全的备份回退运行。

若要了解详细信息，请查看 Docker 容器简介。

先决条件

- WSL 版本 1.1.3.0 或更高版本。
- Windows 11 家庭版和专业版、 企业和教育版、Windows 10 22H2（内部版本 19045）64 位 家庭版和专业版，或 企业和教育 版（推荐）。

  Windows 10 21H2（内部版本 19044）64 位 家庭版和专业版，或 企业和教育 版（最低）。 更新 Windows
- 具有 二级地址转换（SLAT） 的 64 位处理器。
- 4GB 系统 RAM 或更高版本。
- 在 BIOS 中启用硬件虚拟化。
- 安装 WSL 并为在 WSL 2 中运行的 Linux 分发设置用户名和密码。
- 安装 Visual Studio Code *（可选）*。 这将提供最佳体验，包括能够在远程 Docker 容器内编码和调试并连接到 Linux 分发版。
- 安装 Windows 终端 *（可选）*。 这将提供最佳体验，包括在同一接口中自定义和打开多个终端（包括 Ubuntu、Debian、PowerShell、Azure CLI 或喜欢使用的任何终端）。
- 在 Docker 中心注册 Docker ID *（可选）*。
- 有关使用条款的更新，请参阅 Docker Desktop 许可协议 。

有关详细信息，请参阅 在 Windows 上安装 Docker Desktop 的 Docker 文档系统要求。

若要了解如何在 Windows Server 上安装 Docker，请参阅 入门：为容器准备 Windows。

> **注意**
>
> WSL 可以在 WSL 版本 1 或 WSL 2 模式下运行分发版。 可以通过打开 PowerShell 并输入：`wsl -l -v`来检查此项。 确保你的发行版设置为使用 WSL 2，然后输入：`wsl --set-version <Distro> 2`。 替换为 `<Distro>` 发行版名称（例如 Ubuntu 18.04）。
>
> 在 WSL 版本 1 中，由于 Windows 和 Linux 之间存在根本差异，Docker 引擎无法直接在 WSL 内部运行，因此 Docker 团队使用 Hyper-V VM 和 LinuxKit 开发了替代解决方案。 但是，由于 WSL 2 现在在具有完整系统调用容量的 Linux 内核上运行，因此 Docker 可以在 WSL 2 中完全运行。 这意味着 Linux 容器可以在不仿真的情况下本机运行，从而提高 Windows 和 Linux 工具之间的性能和互作性。

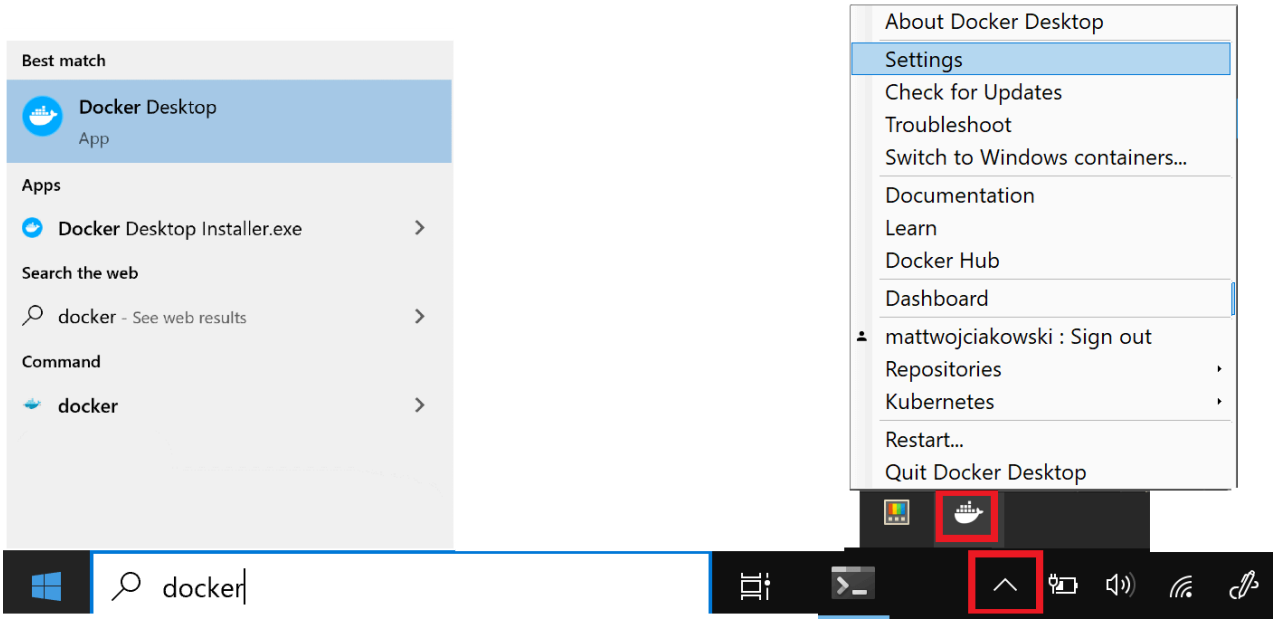
安装 Docker Desktop

使用适用于 Windows 的 Docker Desktop 支持的 WSL 2 后端，可以在基于 Linux 的开发环境中工作并生成基于 Linux 的容器，同时使用 Visual Studio Code 进行代码编辑和调试，并在 Windows 上的 Microsoft Edge 浏览器中运行容器。

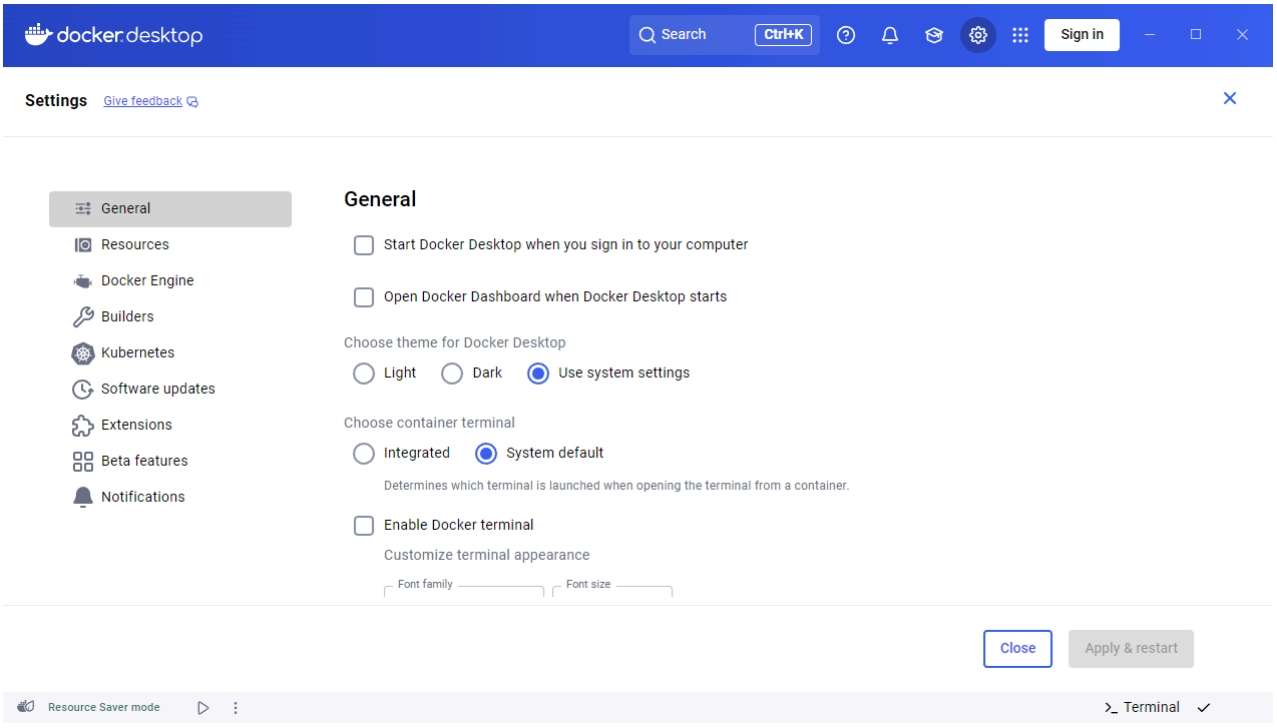
若要安装 Docker（安装 WSL 后）：

1. 下载 Docker Desktop 并按照安装说明进行作。

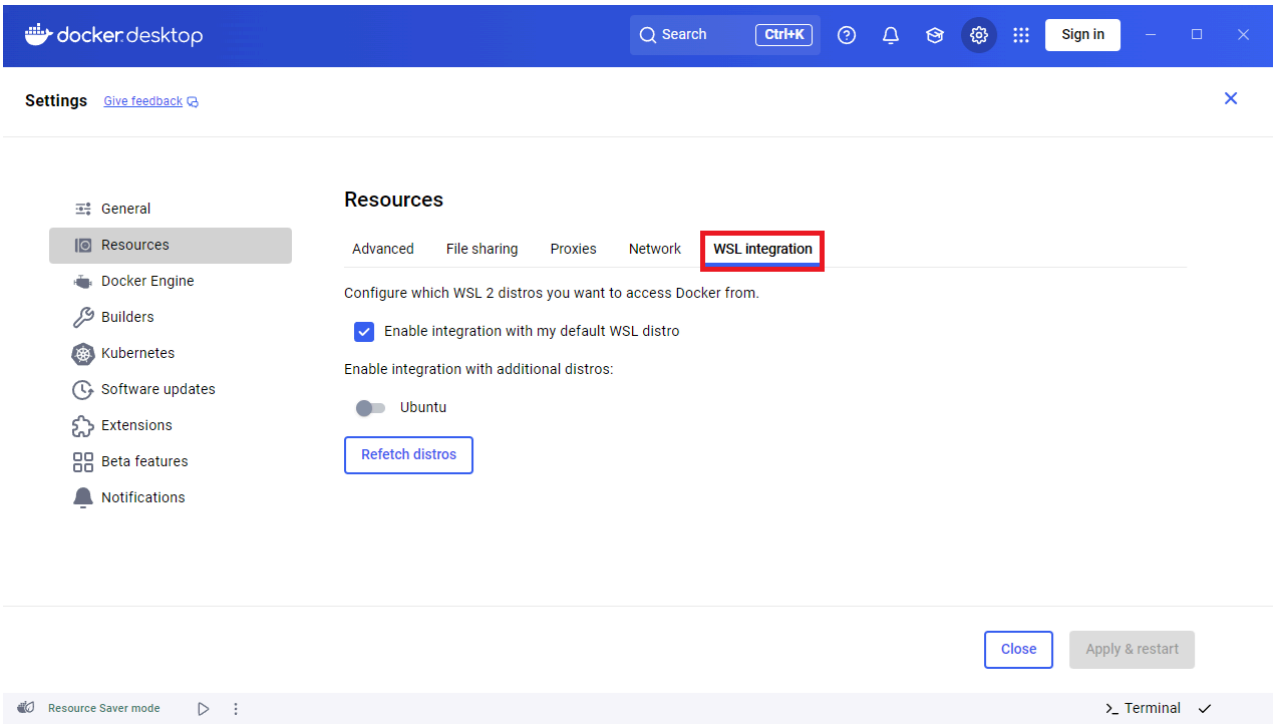
2. 安装后，从 Windows"开始"菜单启动 Docker Desktop，然后从任务栏的隐藏图标菜单中选择 Docker 图标。 右键单击图标以显示 Docker 命令菜单，然后选择"设置"。



3. 确保在"**设置**>"中选中"使用基于 WSL 2 的引擎"。



4. 通过转到 "**设置**>**资源**>**WSL 集成**"，从要启用 Docker 集成的已安装 WSL 2 分发版中进行选择。



5. 要确认 Docker 是否已安装，请打开 WSL 发行版（如 Ubuntu），并输入以下命令以显示版本及内部版本号：`docker --version`

6. 使用以下方法测试安装是否正常工作：运行简单的内置 Docker 映像：`docker run hello-world`

> **提示**
>
> 下面是一些有用的 Docker 命令，需要了解：
>
> - 通过输入以下命令列出 Docker CLI 中可用的命令：`docker`
> - 使用以下命令列出特定命令的信息：`docker <COMMAND> --help`
> - 列出您计算机上的 Docker 镜像（此时仅有 hello-world 镜像）：`docker image ls --all`
> - 列出计算机上的容器，其中包含：`docker container ls --all` 或 `docker ps -a`（如果没有 -a 显示所有标志，将仅显示正在运行的容器）
> - 列出有关 Docker 安装的系统范围信息，包括 WSL 2 上下文中可用的资源（CPU 和内存）和统计信息，以及：`docker info`

使用 VS Code 在远程容器中开发

若要开始使用 Docker 和 WSL 2 开发应用，我们建议使用 VS Code 以及 WSL、开发容器和 Docker 扩展。

- 安装 VS Code WSL 扩展。 此扩展使你能够打开在 VS Code 中的 WSL 上运行的 Linux 项目（无需担心路径问题、二进制兼容性或其他跨 OS 挑战）。
- 安装 VS Code 开发容器扩展。 通过此扩展，可以在容器内打开项目文件夹或存储库，利用 Visual Studio Code 的完整功能集在容器中执行开发工作。
- 安装 VS Code Docker 扩展。 此扩展添加了从 VS Code 内部生成、管理和部署容器化应用程序的功能。 （需要开发容器扩展才能实际使用容器作为开发环境。

让我们使用 Docker 为现有应用项目创建开发容器。

1. 在本示例中，我将使用 Python 开发环境中 Django 的 Hello World 教程 中的源代码来设置文档。如果希望使用自己的项目源代码，可以跳过此步骤。 若要从 GitHub 下载 HelloWorld-Django Web 应用，请打开 WSL 终端（例如 Ubuntu），然后输入：

   ```Bash
   git clone https://github.com/mattwojo/helloworld-django.git
   ```

   > **注意**
   >
   > 始终将代码存储在正在使用的工具所在的同一文件系统中。 这将导致文件访问性能更快。 在此示例中，我们使用 Linux 发行版（Ubuntu），并希望将项目文件存储在 WSL 文件系统 `\\wsl\` 上。 使用 WSL 中的 Linux 工具访问这些文件时，在 Windows 文件系统上存储项目文件会显著降低速度。

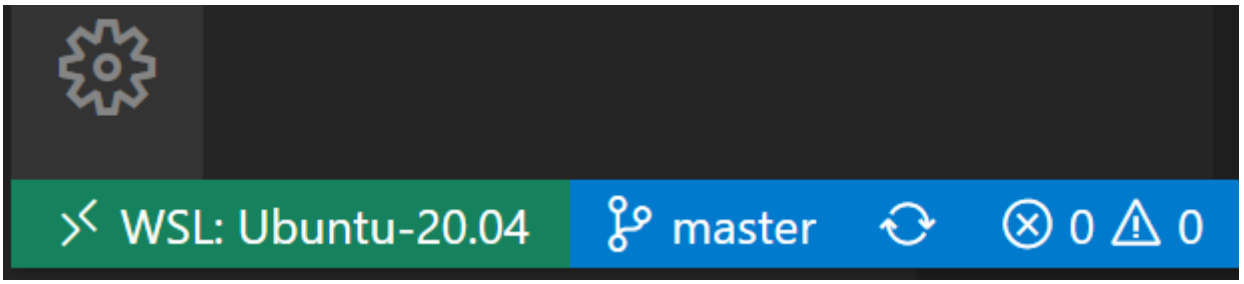
2. 在 WSL 终端中，切换到此项目的源代码目录：

   ```Bash
   cd helloworld-django
   ```

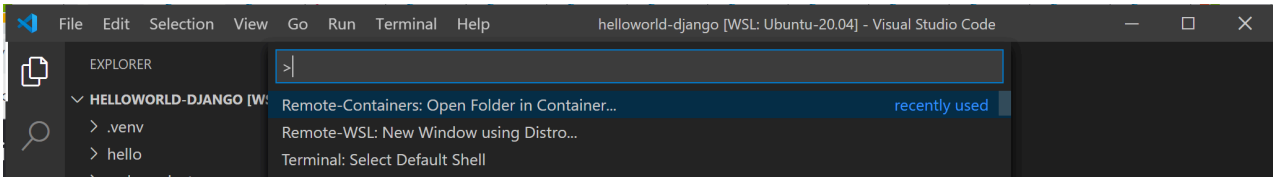
3. 输入以下命令，打开在本地 WSL 扩展服务器上运行的 VS Code 中的项目：

   ```Bash
   code .
   ```

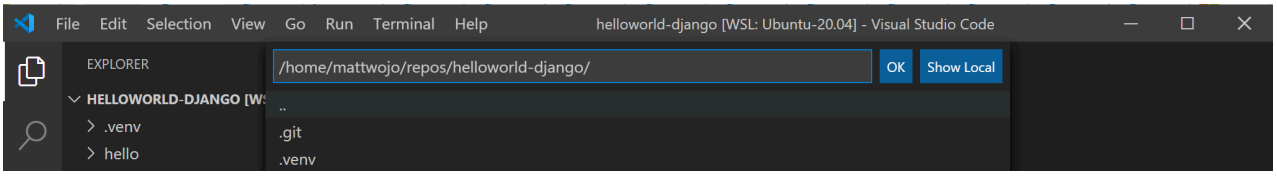
   通过检查 VS Code 实例左下角的绿色远程指示器，确认已连接到 WSL Linux 发行版。



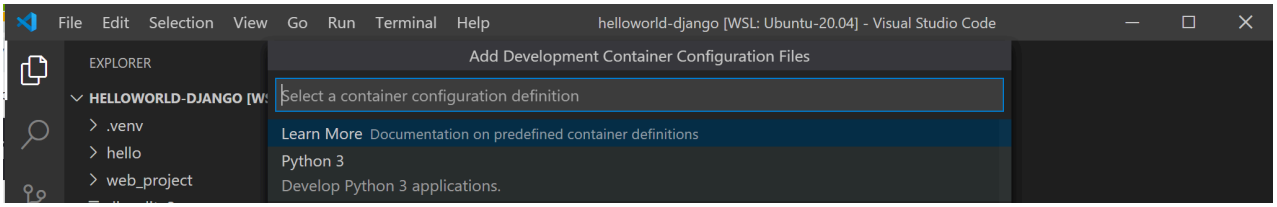
4. 在 VS Code 命令面板（Ctrl + Shift + P）中，输入：**开发者容器：重新以容器方式打开**，因为我们使用 WSL 扩展已经打开了该文件夹。 或者，使用 **开发容器：在容器中打开文件夹**... 使用本地 `\\wsl$` 共享（从 Windows 端）选择 WSL 文件夹。 有关更多详细信息，请参阅 Visual Studio Code 快速入门：在容器中打开现有文件夹 。 如果这些命令在开始键入时未显示，请检查以确保已安装上面链接的开发容器扩展。



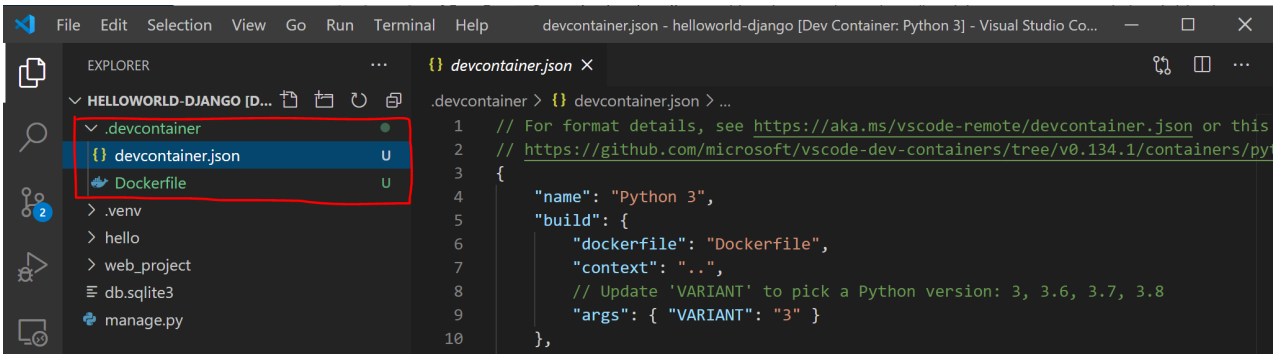
5. 选择要容器化的项目文件夹。 在本例中，这是 `\\wsl\Ubuntu-20.04\home\mattwojo\repos\helloworld-django\`



6. 将显示容器定义列表，因为项目文件夹（存储库）中尚没有开发容器配置。 显示的容器配置定义列表根据项目类型进行筛选。 对于 Django 项目，我将选择 Python 3。

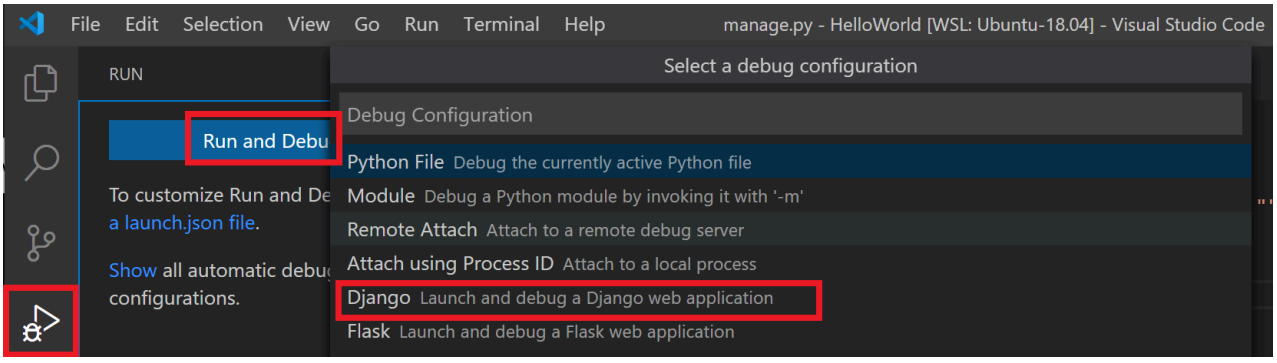


7. VS Code 的新实例将打开，开始生成新映像，生成完成后，将启动容器。 你将看到一个新的 `.devcontainer` 文件夹，其中包含容器配置信息，位于 `Dockerfile` 和 `devcontainer.json` 文件中。

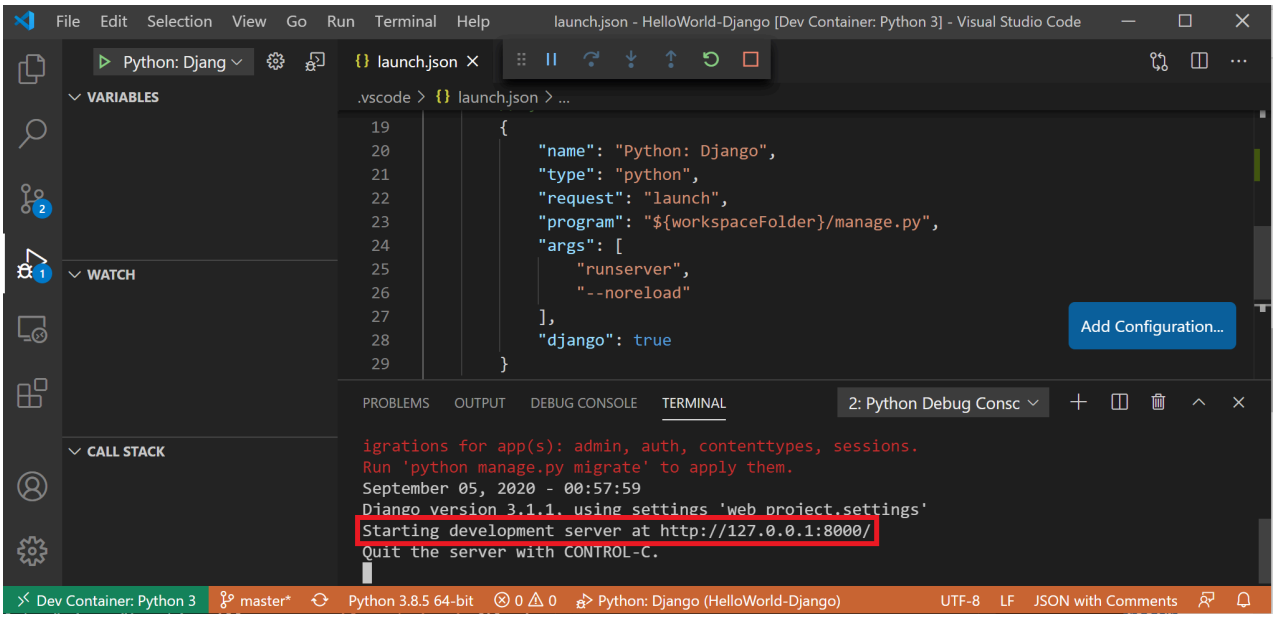


8. 若要确认项目仍连接到 WSL 和容器中，请打开 VS Code 集成终端（Ctrl + Shift + ~）。 输入以下代码检查作系统：`uname` 和 Python 版本：`python3 --version`。 可以看到，uname 显示为“Linux”，这意味着你仍然连接到 WSL 2 引擎，而 Python 版本号将根据容器配置而定，并可能与您在 WSL 发行版上安装的 Python 版本不同。

9. 若要使用 Visual Studio Code 在容器内运行和调试应用，请先打开 “**运行** ”菜单（Ctrl+Shift+D 或选择最左侧菜单栏上的选项卡）。 然后选择“ **运行”和“调试** ”以选择最适合你的项目的配置（在本示例中，这是“Django”）。 这将在`launch.json`项目的文件夹中创建一个`.vscode`文件，其中包含有关如何运行应用的说明。



10. 在 VS Code 内部，选择“ **运行**>**开始调试** ”（或只按 **F5** 键）。 这将在 VS Code 中打开一个终端。你应该看到如下所示的结果：“开发服务器正在 `http://127.0.0.1:8000/` 启动。使用 CONTROL-C 退出服务器。” 按住 Control 键并选择显示的地址，以在默认 Web 浏览器中打开应用程序，并查看项目在其容器中运行。



现在，你已使用由 WSL 2 后端提供支持的 Docker Desktop 成功配置远程开发容器，可以使用 VS Code 编写代码、生成、运行、部署或调试！

故障排除

WSL docker 上下文已弃用

如果使用的是适用于 WSL 的 Docker 早期技术预览版，则可能具有一个名为“wsl”的 Docker 上下文，该上下文现已弃用，不再使用。 可以使用以下命令进行检查： `docker context ls` 可以删除此“wsl”上下文以避免命令错误： `docker context rm wsl` 因为你想要对 Windows 和 WSL2 使用默认上下文。

可能会遇到此已弃用 wsl 上下文的可能错误包括：“docker wsl open //./pipe/docker_wsl：系统找不到指定的文件。”或“连接期间出错：获取 http://%2F%2F.%2Fpipe%2Fdocker_wsl/v1.40/images/json?all=1: 打开 //./pipe/docker_wsl：系统找不到指定的文件。

有关此问题的详细信息，请参阅 如何在 Windows 10 上的 Windows System for Linux（WSL2）中设置 Docker。

查找 docker 映像存储文件夹时遇到问题

Docker 创建两个发行版文件夹来存储数据：

- \wsl$\docker-desktop
- \wsl$\docker-desktop-data

可以通过打开 WSL Linux 分发版并输入： `explorer.exe .` 在 Windows 文件资源管理器中查看文件夹来查找这些文件夹。 Enter： `\\wsl\<Distro>\mnt\wsl` 替换为 `<Distro>` 分发的名称（例如 Ubuntu-24.04），以查看这些文件夹。

在 WSL 中查找 Docker 存储位置的详细信息，请参阅 WSL 存储库 或此 StackOverflow 文章 中的此问题。

有关 WSL 中常规故障排除问题的更多帮助，请参阅 故障排除 文档。

其他资源

- Docker 文档：使用 WSL 2 的 Docker Desktop 最佳做法
- 适用于 Windows 的 Docker Desktop 的反馈：提出问题
- VS Code Docs：选择开发环境的指南
- VS Code 博客：在 WSL 2 中使用 Docker
- VS Code 博客：在 WSL 2 中使用远程容器
- Hanselminutes 播客：与 Simon Ferquel 一起让 Docker 更贴近开发人员的需求

Last updated on 2025/08/15

演练：使用 WSL 2 和 Visual Studio 2022 生成和调试 C++

Visual Studio 2022 引入了一套用于开发 适用于 Linux 的 Windows 子系统 版本 2 (WSL 2) 的本机 C++ 工具集。 此工具集现已在 Visual Studio 2022 版本 17.0 或更高版本中提供。

WSL 2 是新的推荐版本的 适用于 Linux 的 Windows 子系统 （WSL）。 它提供更好的 Linux 文件系统性能、GUI 支持和完整的系统调用兼容性。 Visual Studio的 WSL 2 工具集允许使用Visual Studio在 WSL 2 发行版上生成和调试 C++ 代码，而无需添加 SSH 连接。 可使用 Visual Studio 2019 版本 16.1 中引入的本机 WSL 1 工具集在 WSL 1 发行版上生成和调试 C++ 代码。

Visual Studio的 WSL 2 工具集支持基于 CMake 和 MSBuild 的 Linux 项目。 CMake 是使用 Visual Studio 进行所有 C++ 跨平台开发的建议。 建议使用 CMake，因为它在 Windows、WSL 和远程系统上生成和调试相同的项目。

有关本主题中信息的视频演示，请参阅 Video：使用 WSL 2 分发版调试 C++ 和 Visual Studio 2022.

WSL 2 工具集背景信息

Visual Studio中的 C++ 跨平台支持假定所有源文件都源自Windows文件系统。 面向 WSL 2 发行版时，Visual Studio执行本地 `rsync` 命令，将文件从 Windows 文件系统复制到 WSL 文件系统。 本地 `rsync` 副本不需要任何用户干预。 当Visual Studio检测到你正在使用 WSL 2 发行版时，它会自动发生。 若要详细了解 WSL 1 与 WSL 2 之间的差异，请参阅比较 WSL 1 和 WSL 2。

Visual Studio中的 CMake 预设集成支持 WSL 2 工具集。 若要了解详细信息，请参阅 Visual Studio 和 Visual Studio Code 中的 CMake 预设集成 和 在 Visual Studio 中使用 CMake 预设进行配置和构建。 本文高级 WSL 2 和 CMake 项目注意事项下还有更高级的信息。

安装生成工具

安装在 WSL 2 上进行生成和调试所需的工具。 在后续步骤中使用 Visual Studio 的 CMake 二进制部署安装最新版本的 CMake。

1. 按照安装 WSL 中的说明来安装 WSL 和 WSL 2 发行版。

2. 在Visual Studio安装程序中，验证是否已安装适用于 Linux 的 **C++ CMake 工具。** 为此，请选择 Visual Studio 版本的 **Modify。** 在 “**单个组件** ”选项卡下，搜索 **适用于 Linux 和 Mac 的 C++ CMake 工具** ，并确保它已选中并安装。 Visual Studio 需要此组件来检测 WSL 安装。

3. 假设你的发行版使用 `apt` （本演练使用 Ubuntu），此时请使用以下命令在 WSL 2 发行版上安装所需的生成工具：

   Bash

   ```
   sudo apt update
   sudo apt install cmake g++ gdb make ninja-build rsync zip
   ```

   安装方法：

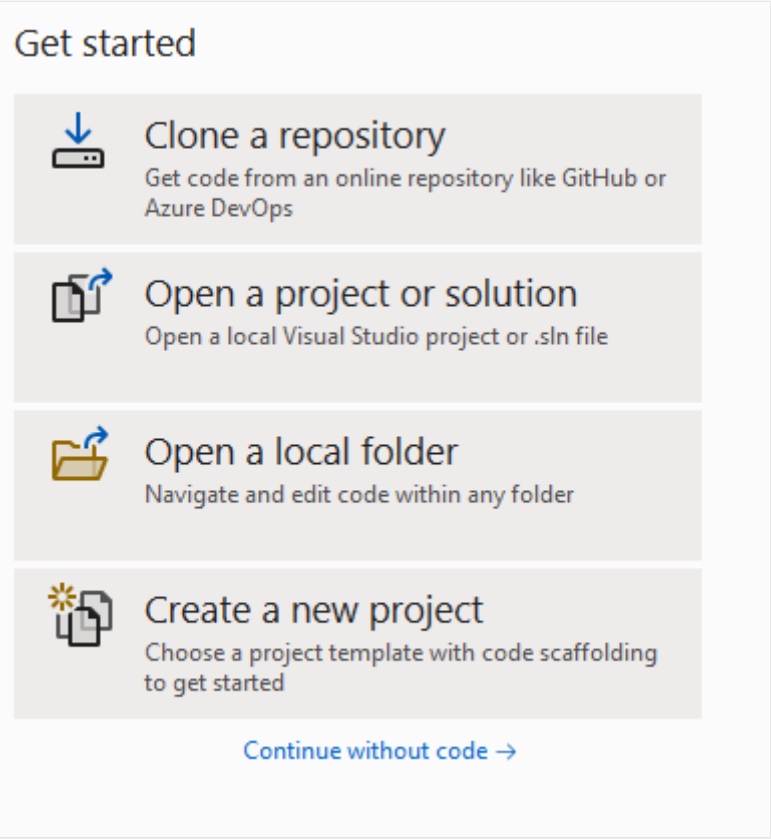
   - C++ 编译器
   - `gdb`
   - `CMake`
   - `rsync`
   - `zip`
   - 底层构建系统生成器

使用 WSL 2 发行版进行跨平台 CMake 开发

本演练使用 Ubuntu 上的 GCC 和 Ninja。 Visual Studio 2022 版本 17.0 预览版 2 或更高版本。

Visual Studio将 CMake 项目定义为项目根目录中具有 `CMakeLists.txt` 文件的文件夹。 在本演练中，你将使用 Visual Studio **CMake Project** 模板创建新的 CMake project：

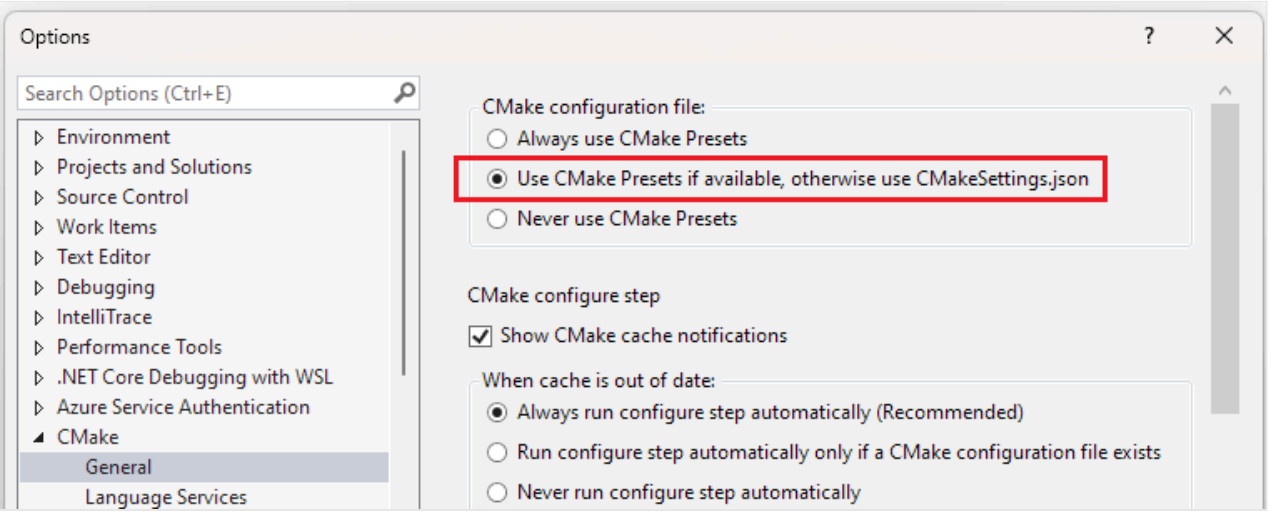
1. 从 Visual Studio **Get started** 屏幕中，选择**创建一个新项目。**



可用选项包括：克隆存储库、打开项目或解决方案、打开本地文件夹、创建新项目或不使用代码继续。

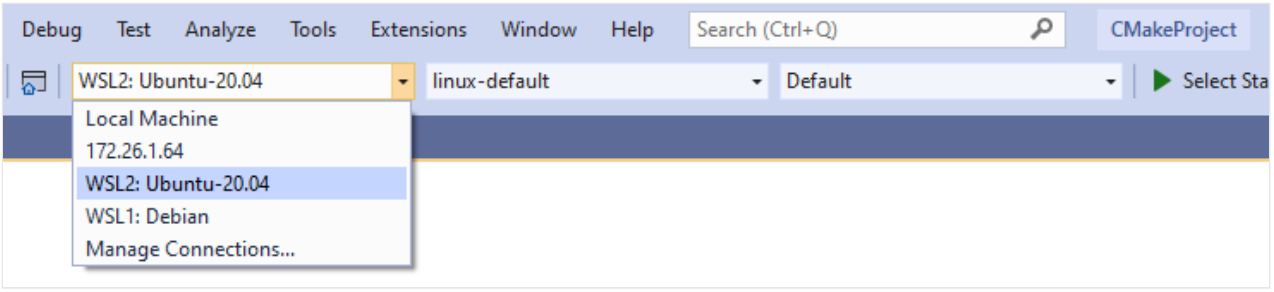
2. 在“搜索模板”文本框中，键入“cmake”。 选择 **CMake Project** 类型，然后选择 **Next**。 提供项目名称和位置，然后选择“创建”。

3. 启用Visual Studio的 CMake 预设集成。 选择**工具**>**选项**>**CMake**>**常规**。 选择“首选使用 CMake 预设进行配置、构建和测试”，然后选择“确定”。 或者，可以将文件添加到 `CMakePresets.json` 项目的根目录。 有关详细信息，请参阅启用 CMake 预设集成。



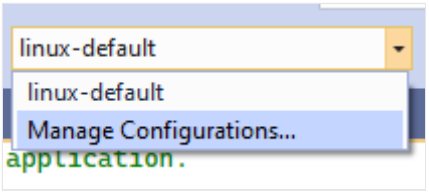
4. 若要激活集成：在主菜单中，选择“文件”“关闭文件夹”>。 ""入门”页面会出现。 " 在“打开最近使用的文件”下，选择刚关闭的文件夹来重新打开它。

5. Visual Studio主菜单栏有三个下拉列表。 使用左侧的下拉列表选择当前的活动目标系统。 将在此系统上调用 CMake 来配置和生成项目。 Visual Studio 使用 `wsl -l -v` 查询 WSL 安装项。 在下图中，WSL2: Ubuntu-20.04 被选为目标系统。



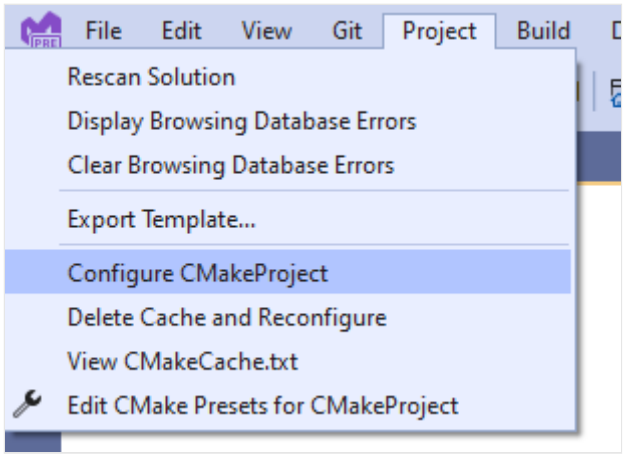
> **注意**
>
> 如果Visual Studio开始自动配置项目，请阅读步骤 11 以管理 CMake 二进制部署，然后继续执行以下步骤。 若要自定义此行为，请参阅**修改自动配置和缓存通知**。

6. 使用中间的下拉列表可选择活动配置预设。 配置预设配置告知 Visual Studio 如何调用 CMake 并生成底层构建系统。 在步骤 7 中，活动配置预设是 Visual Studio 创建的 **linux-default** 预设。 若要创建自定义配置预设，请选择“管理配置...”；有关配置预设的详细信息，请参阅选择配置预设和编辑预设。

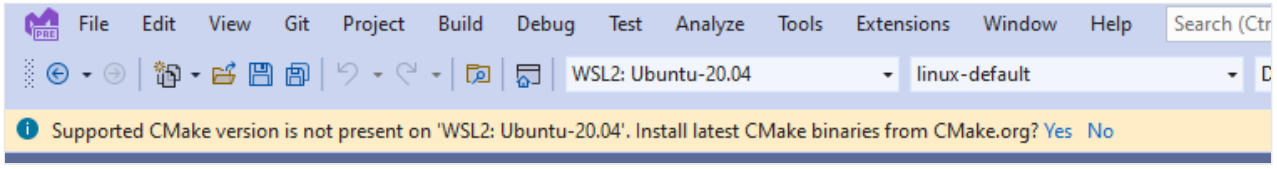


7. 使用右侧的下拉列表选择你的当前构建预设。 生成预设用于告知Visual Studio如何执行构建。 在步骤 7 的图中，活动生成预设是由 Visual Studio 创建的 **Default** 生成预设。 若要详细了解生成预设，请参阅选择生成预设。

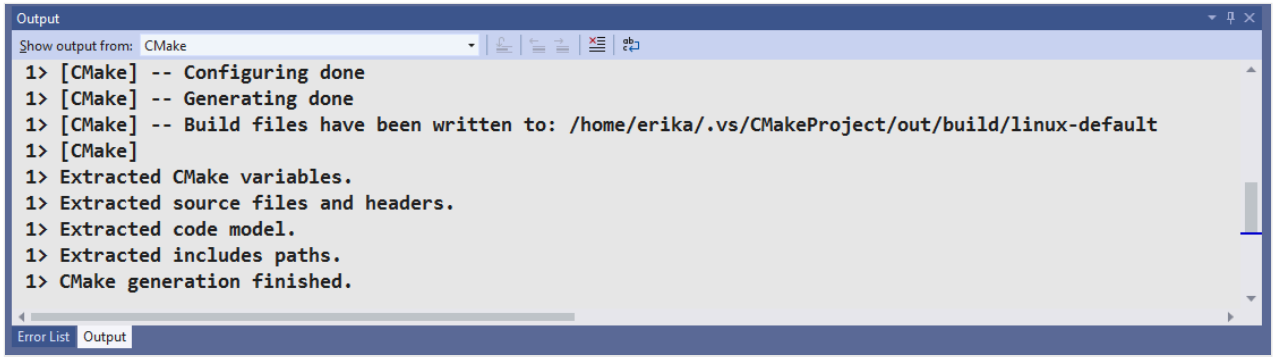
8. 在 WSL 2 上配置项目。 如果项目生成未能自动启动，请使用 **Project**>**Configure***project-name* 手动调用配置。



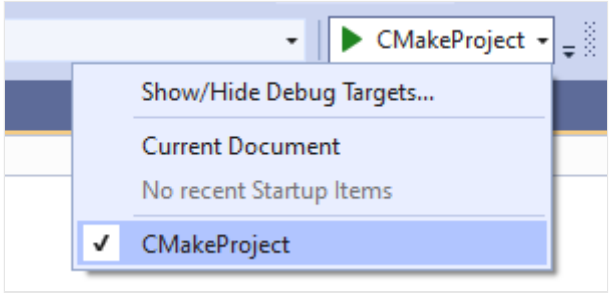
9. 如果 WSL 2 发行版上未安装受支持的 CMake 版本，Visual Studio提示在主菜单功能区下部署最新版本的 CMake。 选择**是**以将 CMake 二进制文件部署到 WSL 2 发行版。



10. 确认已完成配置步骤，并且你在“**CMake**”窗格下的“**输出**”窗口中看到“**CMake 生成已完成**”消息。 生成文件会写入 WSL 2 发行版的文件系统中的某个目录。

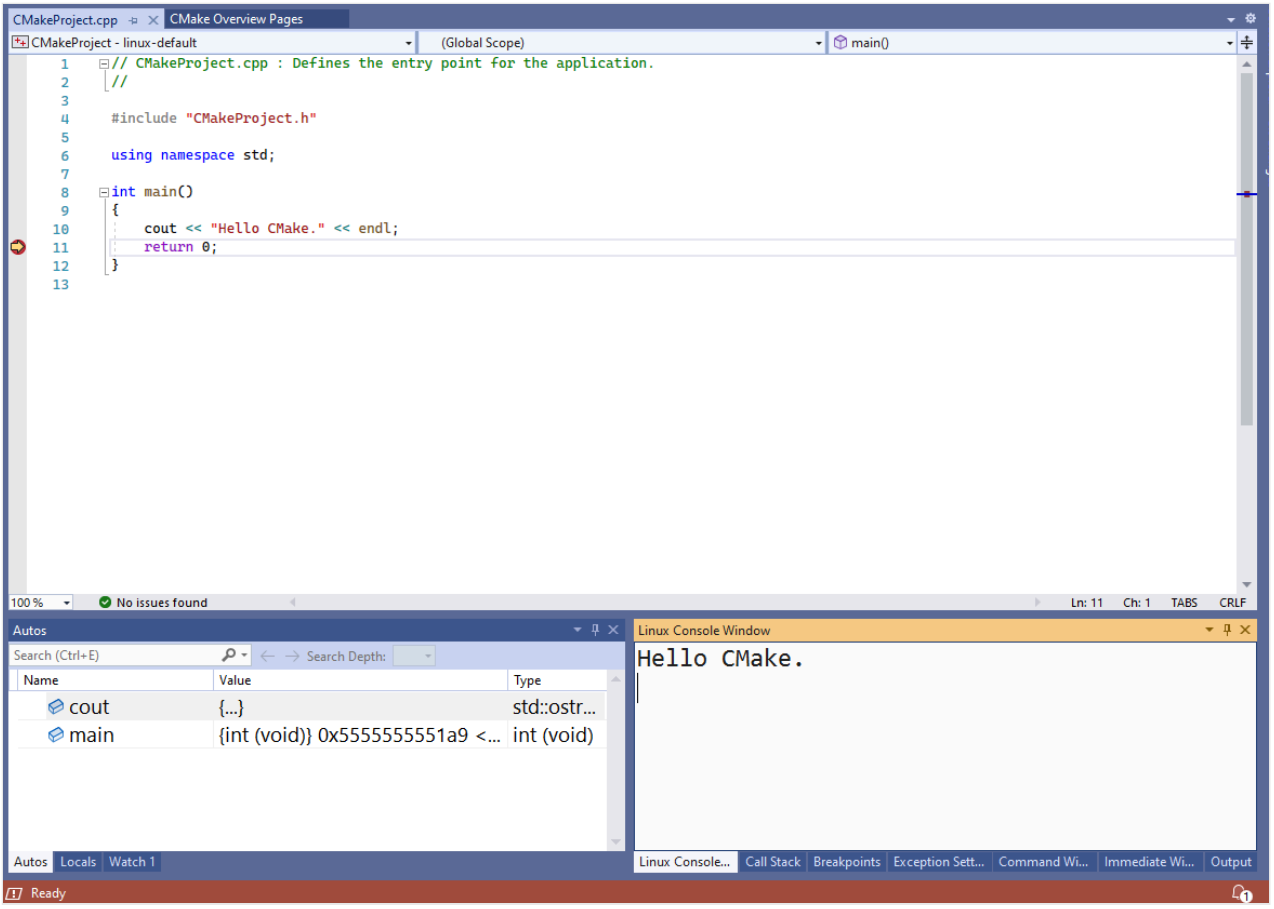


11. 选择活动调试目标。 调试下拉菜单列出了项目可使用的所有 CMake 目标。



12. 在 **解决方案资源管理器** 中展开项目子文件夹。 在 `CMakeProject.cpp` 文件中，在 `main()` 中设置断点。 您也可以通过选择**解决方案资源管理器**中突出显示的“视图选取器”按钮来导航到 CMake 目标视图，如下面的屏幕截图所示。

13. 选择**调试**>**开始**，或者按**F5**。 项目会执行生成操作，可执行文件将在 WSL 2 发行版上启动，而 Visual Studio 将在断点处停止执行。 Linux 控制台窗口会显示程序的输出（在本例中为 `"Hello CMake."`）：



现已使用 WSL 2 和 Visual Studio 2022 生成和调试 C++ 应用。

进阶 WSL 2 和 CMake 项目注意事项

Visual Studio 仅为使用 `CMakePresets.json` 作为活动配置文件的 CMake 项目提供对 WSL 2 的本机支持。 若要从 `CMakeSettings.json` 迁移到 `CMakePresets.json`，请参阅 在 Visual Studio 中启用 CMake 预设集成。

如果面向 WSL 2 发行版，但你不想使用 WSL 2 工具集，请在 `CMakePresets.json` 中的 Visual Studio 远程设置供应商映射中，将 forceWSL1Toolset 设置为 true。 有关详细信息，请参阅 Visual Studio 远程设置供应商映射图。

如果 **forceWSL1Tooslet** 设置为 **true**，则Visual Studio不会维护 WSL 文件系统中的源文件的副本。 而是访问装载的Windows驱动器（`/mnt/`...）中的源文件。

在大多数情况下，最好将 WSL 2 工具集与 WSL 2 分发版配合使用，因为当项目文件存储在Windows文件系统中时，WSL 2 速度较慢。 若要详细了解 WSL 2 中的文件系统性能，请参阅比较 WSL 1 和 WSL 2。

在 `CMakePresets.json` 的 Visual Studio 远程设置供应商映射中，指定高级设置，如 WSL 2 上项目被复制到的目录路径、复制源选项和 rsync 命令参数。 有关详细信息，请参阅Visual Studio远程设置供应商映射。

系统标头仍会自动复制到 Windows 文件系统，以提供原生 IntelliSense 体验。 可在 `CMakePresets.json` 中的 Visual Studio 远程设置供应商映射中，自定义要包含或不包含在此副本中的标头。

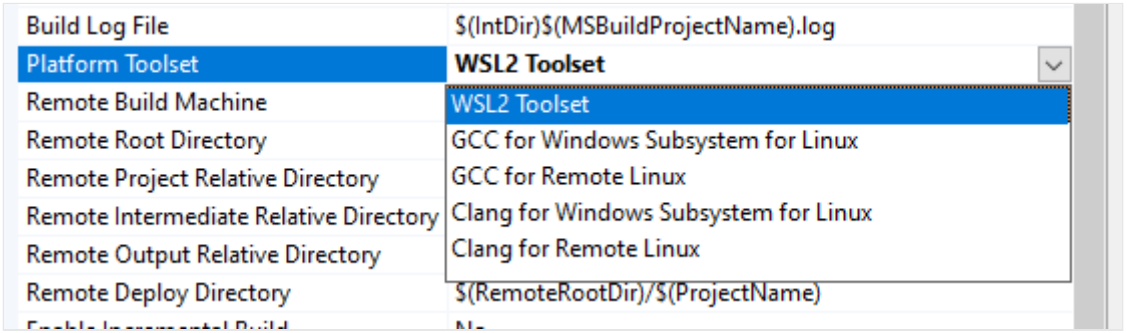
可以在 `CMakePresets.json` 的Visual Studio设置供应商映射中更改 IntelliSense 模式或指定其他 IntelliSense 选项。 有关供应商映射的详细信息，请参阅 Visual Studio 远程设置供应商映射。

基于 WSL 2 和 MSBuild 的 Linux 项目

建议对所有使用 Visual Studio 的 C++ 跨平台开发使用 CMake，因为它允许在 Windows、WSL 和远程系统上生成和调试同一项目。

但是，你可能有一个基于 MSBuild 的 Linux 项目。

如果你有基于 MSBuild 的 Linux 项目，则可以升级到 Visual Studio 中的 WSL 2 工具集。 请在解决方案资源管理器中右键单击项目，然后选择“属性”“常规”>“平台工具集”：>



如果您要针对 WSL 2 分发版并且不想使用 WSL 2 工具集，请在 **Platform Toolset** 下拉列表中选择 **用于 适用于 Linux 的 Windows 子系统 的 GCC 工具集** 或 **用于 适用于 Linux 的 Windows 子系统 的 Clang 工具集**。 如果选择上述任一工具集，Visual Studio不会在 WSL 文件系统中维护源文件的副本，而是通过装载的Windows驱动器（`/mnt/`...）访问源文件。 系统标头仍会自动复制到 Windows 文件系统，以提供本地 IntelliSense 体验。 在“属性页”“常规”中，自定义要包含或不包含在此副本中的标头。>

在大多数情况下，最好将 WSL 2 工具集与 WSL 2 分发版配合使用，因为当项目文件存储在Windows文件系统中时 WSL 2 速度较慢。 要了解详细信息，请参阅比较 WSL 1 和 WSL 2。

另请参阅

视频：使用 WSL 2 发行版和 Visual Studio 2022 调试 C++

Download Visual Studio 2022

在 Visual Studio

Tutorial：在远程Windows计算机上调试 CMake 项目

Last updated on 2026/04/17

WSL 中 ML 的 GPU 加速入门

机器学习（ML）正成为许多开发工作流的关键部分。 无论你是数据科学家、ML 工程师还是开始使用 ML for Linux 的 Windows 子系统（WSL）开始学习之旅，都提供了一个运行最常见和常用的 GPU 加速 ML 工具的绝佳环境。

设置这些工具的方法有很多不同。 例如，WSL、TensorFlow-DirectML 和 PyTorch-DirectML 中的 NVIDIA CUDA 都提供了将 GPU 用于 ML 和 WSL 的不同方式。 若要详细了解选择一个或另一个原因，请参阅 GPU 加速 ML 训练。

本指南介绍如何设置：

- 如果您有 NVIDIA 图形卡并运行示例的 ML 框架容器，则可以使用 NVIDIA CUDA。
- AMD、Intel 或 NVIDIA 图形卡上的 TensorFlow-DirectML 和 PyTorch-DirectML

先决条件

- 确保运行的是 Windows 11 或 Windows 10 版本 21H2 或更高版本。
- 安装 WSL 并为 Linux 分发版设置用户名和密码。

使用 Docker 设置 NVIDIA CUDA

1. 下载并安装 NVIDIA GPU 的最新驱动程序

2. 运行以下命令，安装 Docker Desktop 或直接在 WSL 中安装 Docker 引擎：

   Bash

   ```
   curl https://get.docker.com | sh
   sudo service docker start
   ```

3. 如果直接安装 Docker 引擎，请按照以下步骤安装 NVIDIA 容器工具包。

   通过运行以下命令为 NVIDIA 容器工具包设置稳定存储库：

   Bash

   ```
   distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
   curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo gpg --dearmor -
   o /usr/share/keyrings/nvidia-docker-keyring.gpg
   curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
   docker.list | sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-
   docker-keyring.gpg] https://#g' | sudo tee /etc/apt/sources.list.d/nvidia-
   docker.list
   ```

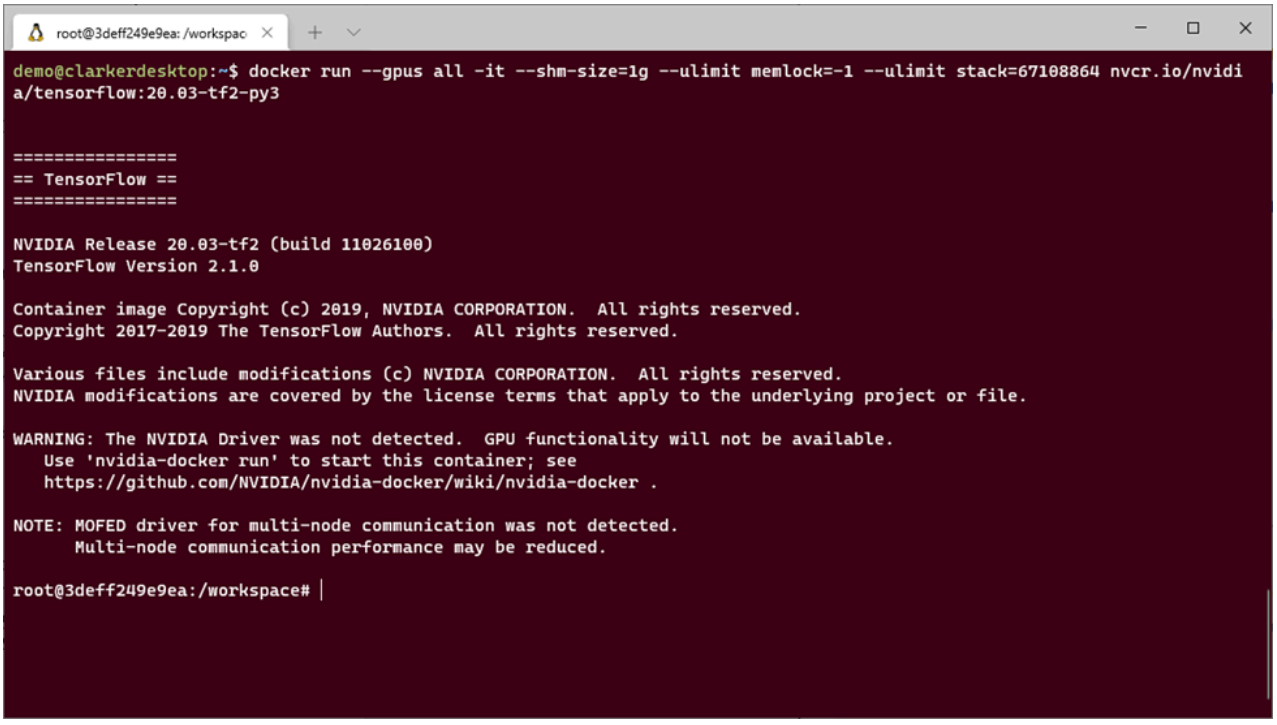
通过运行以下命令安装 NVIDIA 运行时包和依赖项：

```
sudo apt-get update
sudo apt-get install -y nvidia-docker2
```

4. 运行机器学习框架容器和示例。

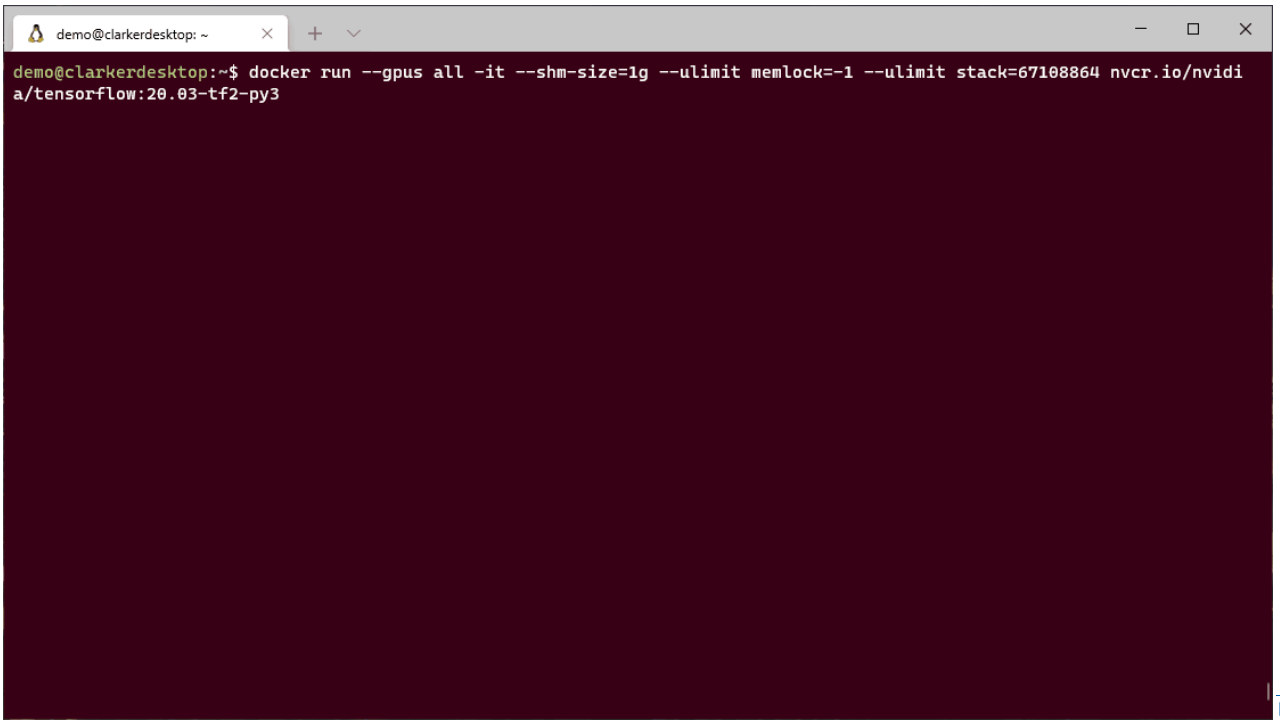
   要运行机器学习框架容器并开始使用 GPU，请输入以下命令以使用 NVIDIA NGC TensorFlow 容器：

```
docker run --gpus all -it --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 nvcr.io/nvidia/tensorflow:20.03-tf2-py3
```



可以通过运行命令来运行内置在此容器中的预先训练的模型示例：

```
cd nvidia-examples/cnn/
python resnet.py --batch_size=64
```



有关获取设置和使用 NVIDIA CUDA 的其他方法，请参阅 WSL 用户指南上的 NVIDIA CUDA。

设置 TensorFlow-DirectML 或 PyTorch-DirectML

1. 从 GPU 供应商网站下载并安装最新的驱动程序： AMD、 Intel 或 NVIDIA。

2. 设置 Python 环境。

   建议设置虚拟 Python 环境。 有许多工具可用于设置虚拟 Python 环境， 对于这些说明，我们将使用 Anaconda 的 Miniconda。

   Bash

   ```
   wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
   bash Miniconda3-latest-Linux-x86_64.sh
   conda create --name directml python=3.7 -y
   conda activate directml
   ```

3. 安装由所选 DirectML 支持的机器学习框架。

   TensorFlow-DirectML：

   Bash

   ```
   pip install tensorflow-directml
   ```

   PyTorch-DirectML：

Bash

```
sudo apt install libblas3 libomp5 liblapack3
pip install torch-directml
```

4. 在 TensorFlow-DirectML 或 PyTorch-DirectML 的交互式 Python 会话中运行快速添加示例，以确保一切正常运行。

如果有疑问或遇到困难，请访问 GitHub 上的 DirectML 存储库。

多个 GPU

如果计算机上有多个 GPU，还可以在 WSL 内部访问它们。 但是，一次只能访问一个。 若要选择特定的 GPU，请将下面的环境变量设置为 GPU 的名称，因为它显示在设备管理器中：

Bash

```
export MESA_D3D12_DEFAULT_ADAPTER_NAME="<NameFromDeviceManager>"
```

这将进行字符串匹配，因此，如果将它设置为“NVIDIA”，它将匹配以“NVIDIA”开头的第一个 GPU。

其他资源

- 在 WSL 中设置 NVIDIA CUDA 的指南
- 在 WSL 中使用 DirectML 设置 TensorFlow 的指导
- TensorFlow 与 DirectML 示例
- 在 WSL 中使用 DirectML 设置 PyTorch 的指导
- 包含 DirectML 示例的 PyTorch

Last updated on 2025/06/10

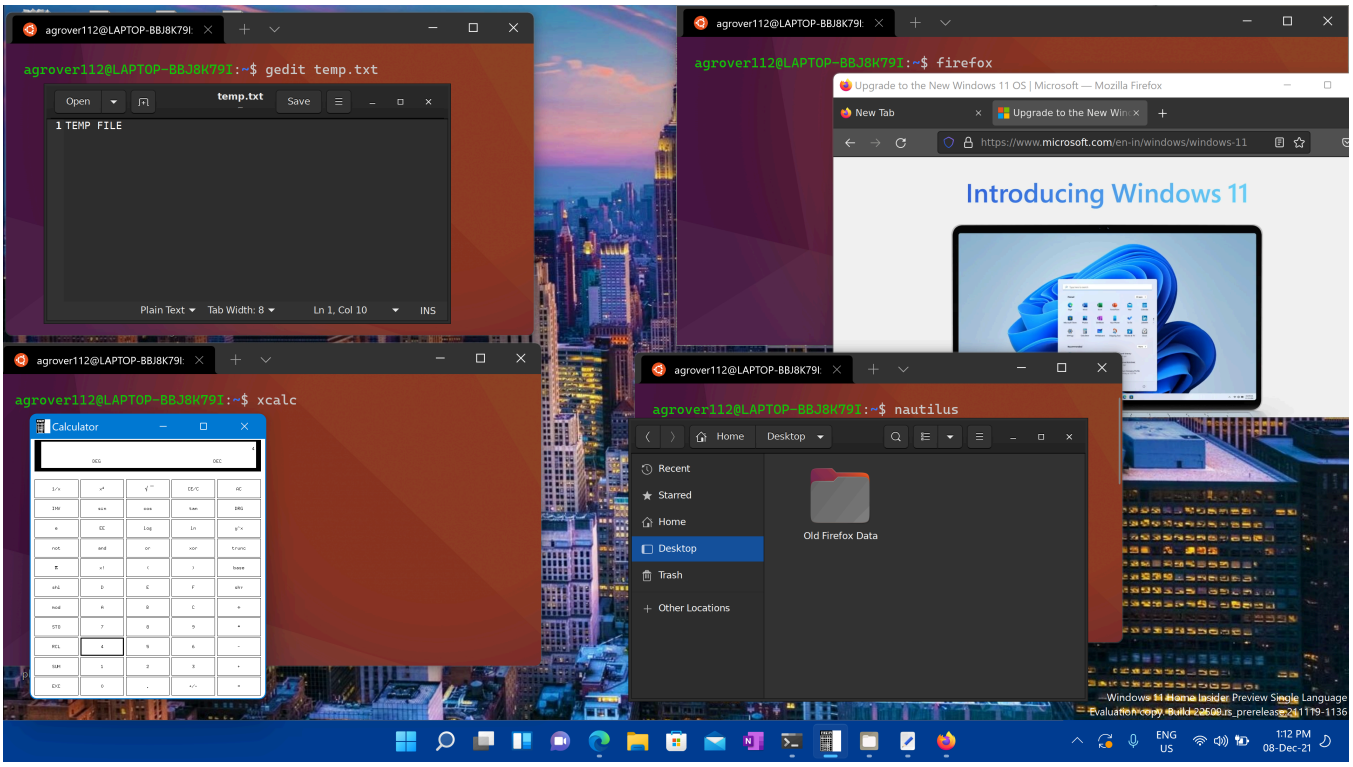
在适用于 Linux 的 Windows 子系统上运行 Linux GUI 应用

适用于 Linux 的 Windows 子系统（WSL）现在支持在完全集成的桌面体验中在 Windows 上运行 Linux GUI 应用程序（X11 和 Wayland）。

WSL 2 使 Linux GUI 应用程序在 Windows 上运行时感觉像原生应用程序一样自然。

- 从 Windows"开始"菜单启动 Linux 应用
- 将 Linux 应用固定到 Windows 任务栏
- 使用 alt-tab 在 Linux 和 Windows 应用之间切换
- 跨 Windows 和 Linux 应用剪切 + 粘贴

现在可以将 Windows 和 Linux 应用程序集成到工作流中，实现无缝桌面体验。



安装对 Linux GUI 应用的支持

先决条件

- 需要位于 **Windows 10 内部版本** 19044+ 或 Windows 11 上才能访问此功能。
- **已安装适用于 vGPU 的驱动程序**

  若要运行 Linux GUI 应用，应首先安装与以下系统匹配的驱动程序。 这样，便可以使用虚拟 GPU （vGPU），以便从硬件加速的 OpenGL 渲染中获益。

- **英特尔** GPU 驱动程序
- **AMD** GPU 驱动程序
- **NVIDIA** GPU 驱动程序

全新安装 - 未曾安装过 WSL

现在可以在管理员 PowerShell 中输入此命令，然后重启计算机，安装运行适用于 Linux 的 Windows 子系统（WSL）所需的所有内容。

PowerShell

```
wsl --install
```

计算机重启后，安装将继续，系统会要求你输入用户名和密码。 这是 Ubuntu 发行版的 Linux 凭据。

现在可以开始在 WSL 上使用 Linux GUI 应用了！

有关详细信息，请查看 安装 WSL。

现有 WSL 安装

如果计算机上已安装 WSL，可以通过从提升的命令提示符运行更新命令，更新到包含 Linux GUI 支持的最新版本。

1. 选择 “**开始**”，键入 **PowerShell**，右键单击 **Windows PowerShell**，然后选择“ **以管理员身份运行**”。
2. 输入 WSL 更新命令：

   PowerShell

   ```
   wsl --update
   ```

3. 需要重启 WSL 才能使更新生效。 可以通过在 PowerShell 中运行关闭命令来重启 WSL。

   PowerShell

   ```
   wsl --shutdown
   ```

> **注意**

Linux GUI 应用仅受 WSL 2 支持，不适用于为 WSL 1 配置的 Linux 分发版。 了解如何 **将分发版从 WSL 1 更改为 WSL 2**。

运行 Linux GUI 应用

可以从 Linux 终端运行以下命令，下载并安装这些常用的 Linux 应用程序。 如果使用的是与 Ubuntu 不同的分发版，则它可能会使用与 apt 不同的包管理器。 安装 Linux 应用程序后，可以在分发名称下的 “**开始**” 菜单中找到它。 例如：Ubuntu -> Microsoft Edge。

> ⓘ **注意**
>
> 对 WSL 上的 GUI 应用的支持不提供完整的桌面体验。 它依赖于 Windows 桌面，因此不支持安装以桌面为中心的工具或应用。 若要请求其他支持，可以在 **GitHub 上的 WSLg 存储库** 中提出问题。

更新您发行版中的软件包

```Bash
sudo apt update
```

安装 Gnome 文本编辑器

Gnome 文本编辑器是 GNOME 桌面环境的默认文本编辑器。

```Bash
sudo apt install gnome-text-editor -y
```

若要在编辑器中启动 bashrc 文件，请输入： `gnome-text-editor ~/.bashrc`

> ⓘ **注意**
>
> **GNOME 文本编辑器** 将 gedit 替换为 Ubuntu 22.10 中的 GNOME/Ubuntu 的默认文本编辑器。 如果你正在运行旧版本的 Ubuntu，并且想要使用以前的默认文本编辑器 **gedit**，请使用： `sudo apt install gedit -y`。

安装 GIMP

GIMP 是一种免费的开源光栅图形编辑器，用于图像作和图像编辑、自由格式绘图、不同图像文件格式之间的转码以及更专门的任务。

```
Bash
sudo apt install gimp -y
```

若要启动，请输入： `gimp`

安装 Nautilus

Nautilus 也称为 GNOME 文件，是 GNOME 桌面的文件管理器。 （类似于 Windows 文件资源管理器）。

```
Bash
sudo apt install nautilus -y
```

若要启动，请输入： `nautilus`

安装 VLC

VLC 是一种免费的开源跨平台多媒体播放器和框架，可播放大多数多媒体文件。

```
Bash
sudo apt install vlc -y
```

若要启动，请输入： `vlc`

安装 X11 应用

X11 是 Linux 开窗系统，这是随其附带的应用和工具的杂项集合，例如 xclock、xcalc 计算器、用于剪切和粘贴的 xclipboard、用于事件测试的 xev 等。有关详细信息，请参阅 x.org 文档 。

```
Bash
sudo apt install x11-apps -y
```

若要启动，请输入要使用的工具的名称。 例如：

- `xcalc`、`xclock`、`xeyes`

安装 Google Chrome for Linux

安装 Google Chrome for Linux：

1. 将目录更改为临时文件夹：

```Bash
cd /tmp
```

2. 使用 wget 下载它：

```Bash
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
```

3. 安装包：

```Bash
sudo apt install -f ./google-chrome-stable_current_amd64.deb
```

*此选项 `-f` 用于修复安装过程中可能出现的损坏依赖项。 该 `./` 命令指定.deb文件所在的当前目录。 如果.deb文件位于其他目录中，则需要在命令中指定该文件的路径。

若要启动，请输入： `google-chrome`

安装适用于 Linux 的 Microsoft Edge 浏览器

查找有关如何使用页面上的命令行安装 Edge for Linux 的信息，以便 成为 Edge 预览体验成员。 在页面的"命令行安装"部分下选择 "**命令行** 安装"。

若要启动，请输入： `microsoft-edge`

故障排除

如果在启动 GUI 应用程序时遇到问题，请先检查本指南： 诊断 WSLg 的"无法打开显示"类型问题

Last updated on 2025/08/15

在适用于 Linux 的 Windows 子系统上安装 Node.js （WSL2）

对于喜欢在 Linux 环境中使用 Node.js 的用户，本指南将帮助你在适用于 Linux 的 Windows 子系统上安装 Node.js（WSL 2 是推荐的版本）。

在决定在原生 Windows 环境还是在 Linux（WSL 2）环境中安装和使用 Node.js 进行开发时，请考虑以下事项：

- **技能级别**：如果你不熟悉使用 Node.js 进行开发，并希望快速启动并运行，以便你可以学习，在 Windows上安装 Node.js。 在 Windows 上安装和使用 Node.js 将为初学者提供比使用 WSL 更复杂的环境。
- **命令行客户端工具**：如果你更喜欢 PowerShell，请使用 Windows 上的 Node.js。 如果你更喜欢 Bash，请使用 Linux 上的 Node.js （WSL 2）。
- **生产服务器**：如果计划在 Windows Server 上部署 Node.js 应用，请在 Windows 上使用 Node.js。 如果打算在 Linux 服务器上部署，请使用 Linux 上的 Node.js（WSL 2）。 使用 WSL 可以安装首选的 Linux 分发版（默认使用 Ubuntu），确保开发环境（编写代码的位置）与生产环境（部署代码的服务器）之间的一致性。
- **性能速度和系统调用兼容性**：Linux 与 Windows 性能存在持续辩论和开发，但使用 Windows 计算机时的关键是将开发项目文件保留在已安装 Node.js的同一文件系统中。 如果在 Windows 文件系统上安装 Node.js，请将文件保存在 Windows 驱动器上（例如 C：/）。 如果在 Linux 分发版（如 Ubuntu）上安装 Node.js，请将项目文件保留在与所使用的分发关联的 Linux 文件系统目录中。 （输入 WSL 分发命令行中的 `explorer.exe .`，以使用 Windows 文件资源管理器浏览目录。
- **Docker 容器**：如果要使用 Docker 容器在 Windows 上开发项目，建议 在 Windows 上安装 Docker Desktop。 若要在 Linux 工作区中使用 Docker，请参阅 使用 WSL 2 为 Windows 设置 Docker Desktop，以避免同时维护 Linux 和 Windows 生成脚本。

安装适用于 Linux 的 Windows 子系统

如果计划将 Linux 开发环境与 Node.js配合使用，请参阅 WSL 安装文档 。 这些步骤将包括选择 Linux 分发版（Ubuntu 是默认值）和适用于 Linux 的 Windows 子系统版本（WSL 2 是默认版本和建议的版本）。 如果需要，可以安装多个 Linux 分发版。

安装 WSL 2 和 Linux 分发版后，打开 Linux 分发版（可在 Windows 终端列表或 Windows 开始菜单中找到），并使用以下命令检查版本和代码名： `lsb_release -dc`

建议定期更新 Linux 发行版，包括在安装之后立即更新，以确保具有最新的包。 Windows 不会自动处理此更新。 要更新发行版，请使用命令：`sudo apt update && sudo apt upgrade`。

Windows 终端

Windows 终端是一个改进的命令行 shell，可用于运行多个选项卡，以便你可以在 Linux 命令行、Windows 命令提示符、PowerShell、Azure CLI 或你喜欢使用的任何内容之间快速切换。 还可以创建自定义键绑定（用于打开或关闭选项卡、复制+粘贴等的快捷键），使用搜索功能，使用主题自定义终端（配色方案、字体样式和大小、背景图像/模糊/透明度），等等。 在 Windows 终端文档中了解详细信息。

安装 nvm、node.js和 npm

除了选择是安装在 Windows 还是 WSL 上，在安装 Node.js时还有其他选择。 建议使用版本管理器，因为版本更改非常快。 您可能需要根据您正在处理的不同项目的需求，在多个版本的 Node.js 之间切换。 节点版本管理器（更常见的称为 nvm）是安装多个版本的 Node.js的最常用方法。 我们将逐步完成安装 nvm 的步骤，然后使用它安装 Node.js 和 Node 包管理器（npm）。 备用版本管理器 也在下一部分中进行了介绍。

> ⓘ **重要**
>
> 通常建议在安装版本管理器之前，先从操作系统中删除任何现有的 Node.js 或 npm 安装，因为不同类型的安装可能会导致奇怪且令人困惑的冲突。 例如，可以使用 Ubuntu 命令 `apt-get` 安装的 Node 版本当前已过时。 有关删除以前安装的帮助，请参阅 **如何从 ubuntu 中删除 nodejs**。

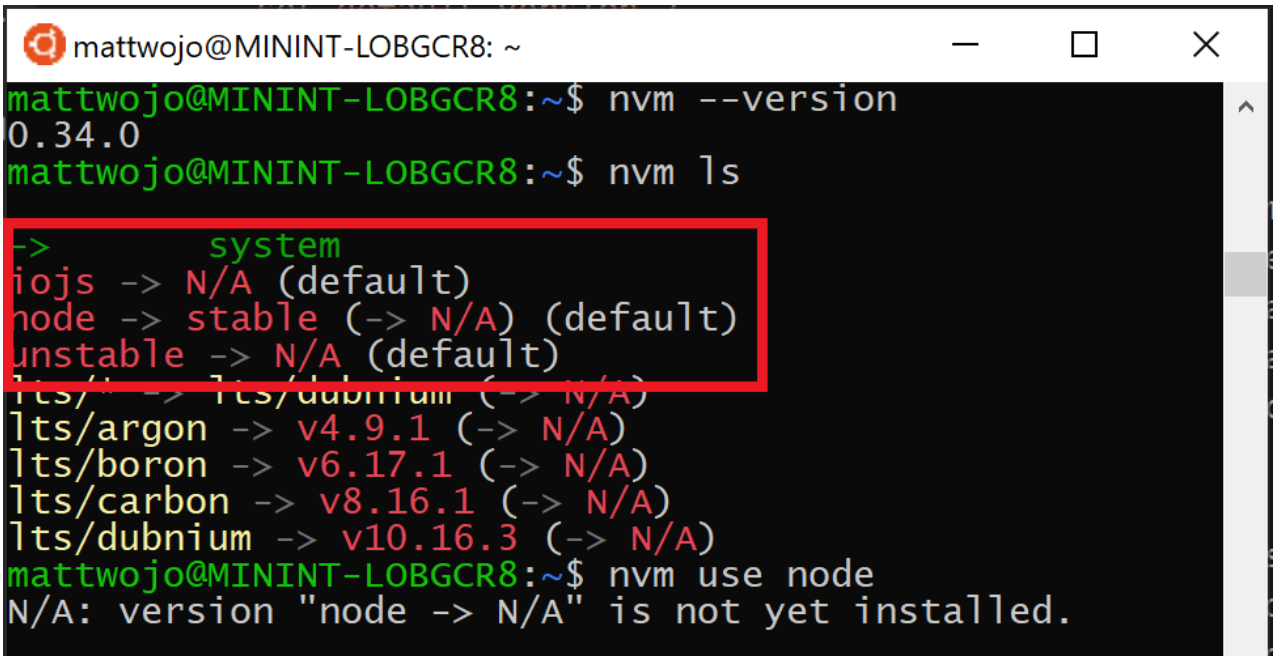
有关安装 NVM 的最新信息，请参阅 GitHub 上的 NVM 存储库中的安装和更新。

1. 打开 Ubuntu 命令行（或你选择的发行版）。
2. 安装 cURL（一个在命令行用于从互联网下载内容的工具）使用：`sudo apt-get install curl`
3. 安装 nvm，使用 `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/master/install.sh | bash`：

   > ⓘ **注意**
   >
   > 使用 cURL 安装较新版本的 NVM 将替换旧版本，使使用 NVM 安装的节点版本保持不变。 有关详细信息，请参阅 **GitHub 项目页，了解有关 NVM 的最新版本信息**。

4. 若要验证安装，请输入：“`command -v nvm`”……如果收到“找不到命令”或根本没有响应，请关闭当前终端，重新打开它，然后重试。 在 nvm github 存储库中了解详细信息。

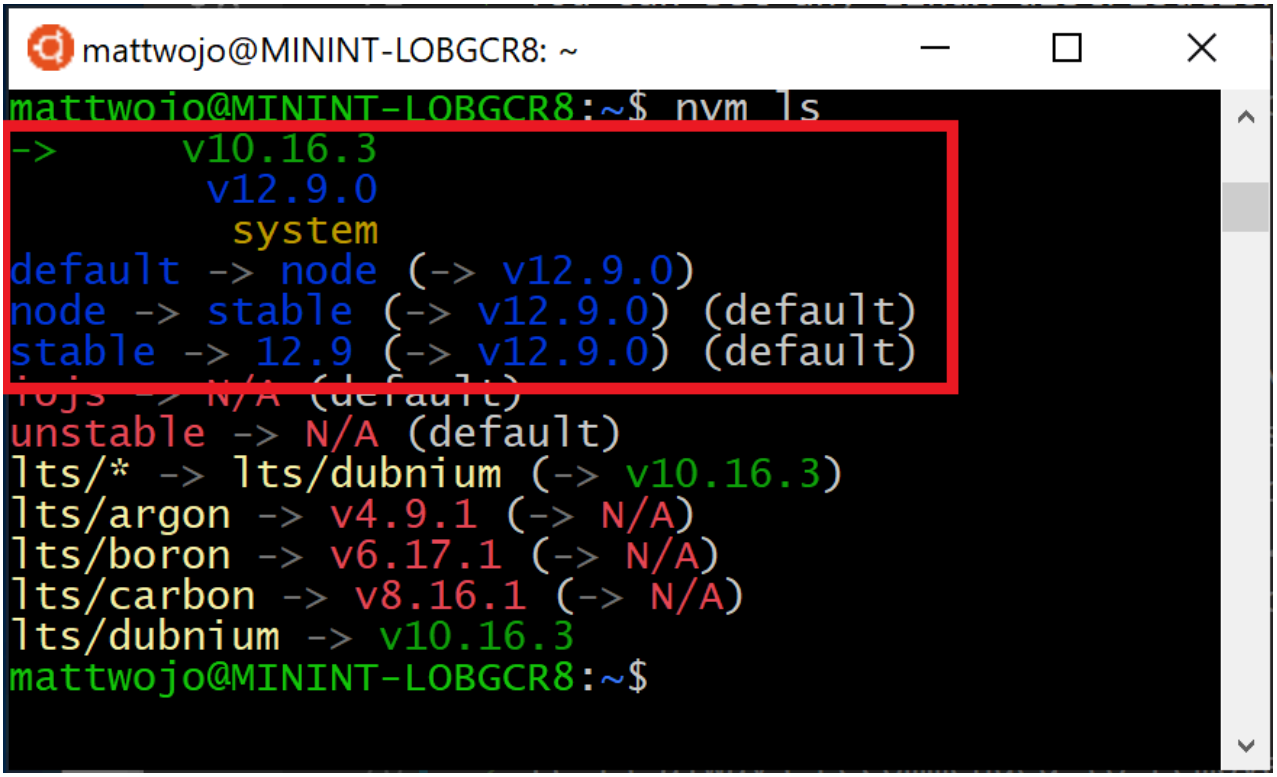
5. 列出当前安装的 Node 版本（此时应不安装）： `nvm ls`



6. 安装 Node.js的当前和稳定的 LTS 版本。 在后面的步骤中，你将了解如何使用 `nvm` 命令在 Node.js 的活动版本之间进行切换。

   - 安装 Node.js 的当前稳定 LTS 版本（建议用于生产应用程序）： `nvm install --lts`
   - 安装当前版本的 Node.js（用于测试最新的 Node.js 功能和改进，但更有可能出现问题）： `nvm install node`

7. 列出已安装的 Node 版本： `nvm ls` ...现在，应会看到刚刚安装的两个版本。



8. 验证是否已安装 Node.js，并检查其是否为当前默认版本：`node --version`。 然后验证你是否安装了 npm，并确认：`npm --version` （你还可以使用 `which node` 或 `which npm` 查看默认版本的路径）。

9. 若要更改要用于项目的 Node.js 版本，请创建新的项目目录，然后输入该目录`mkdir NodeTest` `cd NodeTest`，然后输入`nvm use node`以切换到当前版本或`nvm use --lts`切换到 LTS 版本。 还可以对已安装的任何其他版本使用特定编号，例如 `nvm use v8.2.1`。 （若要列出所有可用的 Node.js 版本，请使用以下命令： `nvm ls-remote`

如果使用 NVM 安装 Node.js 和 NPM，则无需使用 SUDO 命令来安装新包。

替代版本管理器

虽然 nvm 目前是节点最受欢迎的版本管理器，但有一些可考虑的替代方法：

- n 是一种长期 `nvm` 替代方法，它使用略有不同的命令完成相同的工作，并通过 `npm` 而不是 bash 脚本进行安装。
- fnm 是一个较新的版本管理器，声称比它快得多 `nvm`。 （它还使用 Azure Pipelines。）
- Volta 是LinkedIn团队推出的新版本管理器，声称具有更快的速度和跨平台支持。
- asdf-vm 是适用于多种语言的单个 CLI，例如将 ike gvm、nvm、rbenv 和 pyenv（等）整合在一起。
- nvs （Node 版本切换器）是跨平台的 `nvm` 替代方法，可与 VS Code 集成。

安装 Visual Studio Code

建议使用 Visual Studio Code 配合远程开发扩展包，用于 Node.js 项目。 这会将 VS Code 拆分为“客户端-服务器”体系结构，客户端（VS Code 用户界面）在 Windows作系统上运行，服务器（代码、Git、插件等）在 WSL Linux 分发版上运行“远程”。

> **注意**
>
> 此“远程”方案与你可能习惯的情况略有不同。 WSL 支持实际的 Linux 发行版用于运行您的项目代码，与 Windows 操作系统独立，但仍在本地计算机上运行。 Remote-WSL 扩展与 Linux 子系统连接，就像它是远程服务器一样，尽管它未在云中运行... 它仍在你启用了的 WSL 环境中运行，与 Windows 一起在本地计算机上运行。

- 支持基于 Linux 的 Intellisense 和 linting。
- 你的项目将在 Linux 中自动生成。
- 可以使用 Linux 上运行的所有扩展（ES Lint、NPM Intellisense、ES6 代码片段等）。

其他代码编辑器（如 IntelliJ、Sublime Text、Brackets 等）也将使用 WSL 2 Node.js 开发环境，但可能没有 VS Code 提供的相同远程功能。 这些代码编辑器在访问 WSL 共享网络位置（\wsl$\Ubuntu\home）时可能会遇到问题，并会尝试通过 Windows 工具处理 Linux 文件，这可能不是你所想要的。 在 VS Code 中，Remote-WSL 扩展可以帮助你处理这种兼容性，而使用其他 IDE 时，你可能需要设置一个 X Server。 即将推出支持在 WSL（如代码编辑器 IDE）中运行的 GUI 应用。

基于终端的文本编辑器（vim、emacs、nano）也有助于从控制台内部快速更改。 本文 Emacs、Nano 或 Vim：明智地选择你的 Terminal-Based 文本编辑器，很好地解释了一些差异，并且讲述了如何使用每个编辑器。

安装 VS Code 和 Remote-WSL 扩展：

1. 下载和安装适用于 Windows 的 VS Code。 VS Code 也可用于 Linux，但适用于 Linux 的 Windows 子系统不支持 GUI 应用，因此我们需要在 Windows 上安装它。 不必担心，你仍然可以使用远程 - WSL 扩展与 Linux 命令行和工具集成。

2. 在 VS Code 上安装 Remote - WSL 扩展。 这使您可以使用 WSL 作为您的集成开发环境，并将为您处理兼容性和路径问题。 了解详细信息。

> **重要**
>
> 如果您已安装 VS Code，需要确保拥有 **1.35 版本 5月发布** 或更高版本，才能安装 **Remote - WSL 扩展**。 建议不要在 VS Code 中使用没有 Remote-WSL 扩展的 WSL，因为这样会失去对自动完成、调试、代码分析等功能的支持。趣味小知识：这个 WSL 扩展安装在 $HOME/.vscode-server/extensions 中。

有用的 VS 代码扩展

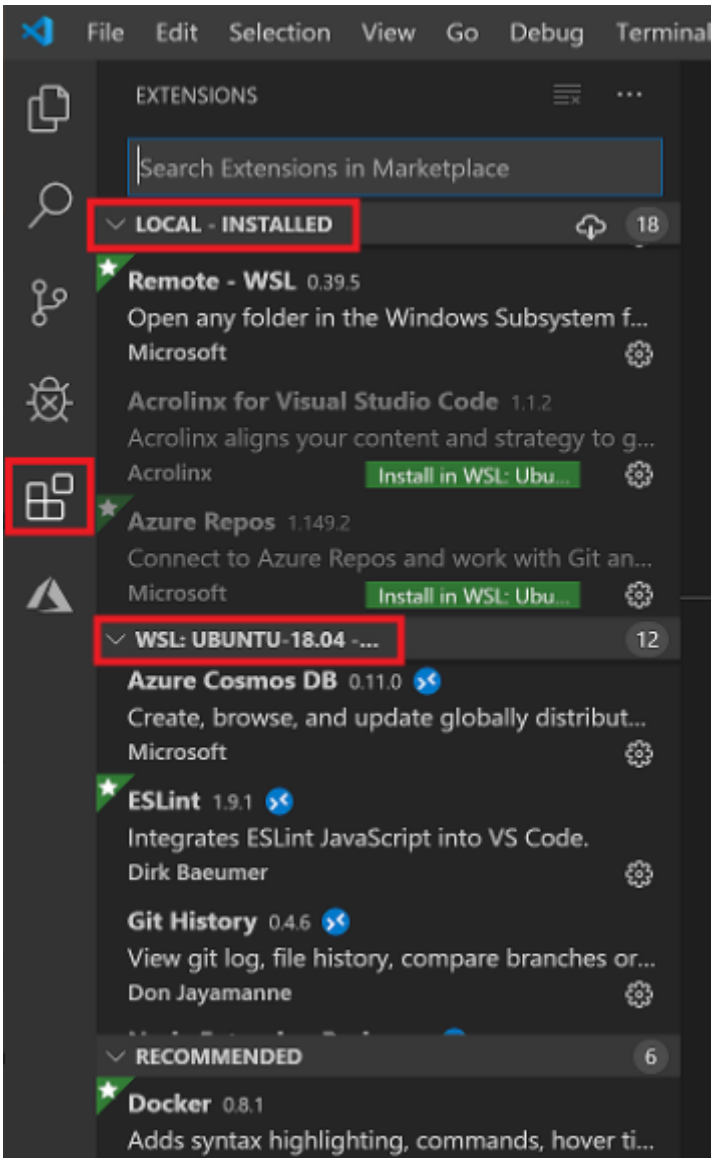
虽然 VS Code 自带了许多适用于 Node.js 开发的功能，但你可以考虑在 Node.js 扩展包中安装一些有用的扩展工具。 你可以全部安装，或者挑选出你觉得最有用的功能来安装。

若要安装 Node.js 扩展包，请执行以下作：

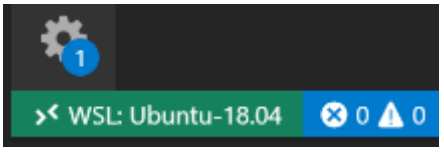
1. 在 VS Code 中打开 “**扩展** ”窗口（Ctrl+Shift+X）。

   “扩展”窗口现在分为三个部分（因为你安装了 Remote-WSL 扩展）。

   - “本地 - 已安装”：为 Windows 操作系统安装的扩展。
   - “WSL：Ubuntu-18.04-Installed”：用于您的 Ubuntu 操作系统 (WSL) 的已安装扩展。
   - “推荐”：VS Code 根据当前项目中的文件类型推荐的扩展。



2. 在“扩展”窗口顶部的搜索框中，输入： **节点扩展包** （或要查找的任何扩展的名称）。 根据您当前项目打开的位置，将在 VS Code 的本地实例或 WSL 实例之一安装该扩展。 可以通过选择 VS Code 窗口左下角的远程链接（绿色）来判断。 它将为你提供打开或关闭远程连接的选项。 在“WSL：Ubuntu-18.04”环境中安装 Node.js 扩展。



可能需要考虑的一些附加扩展包括：

- JavaScript 调试器：使用 Node.js在服务器端完成开发后，需要开发和测试客户端。 此扩展是基于 DAP 的 JavaScript 调试器。 它调试 Node.js、Chrome、Edge、WebView2、VS Code 扩展等。
- 来自其他编辑器的键映射：如果你从另一个文本编辑器（如 Atom、Sublime、Vim、Emacs、Notepad++等）转换，这些扩展可以帮助你的环境更适应你的使用习惯。
- 设置同步：使你能够使用 GitHub 跨不同安装同步 VS Code 设置。 如果你在不同的计算机上工作，这有助于在这些计算机之间保持你的工作环境一致。 重要的是，此扩展现已弃

用。 对于类似的同步解决方案，请使用 Visual Studio Code 内置的 **设置同步**，可以通过导航到 **文件**>**首选项**>，看到勾选标记表示**设置同步已开启。**

设置 Git （可选）

要在 WSL 上为 Node.js 项目设置 Git，请参阅 WSL 文档中的文章 “在 Linux 的 Windows 子系统上开始使用 Git”。

Last updated on 2025/06/11

Linux 和 Bash 入门

本教程将帮助 Linux 新手开始在默认通过 WSL 安装的 Linux 中的 Ubuntu 发行版上安装和更新软件包，并使用 Bash 命令行中的一些基本命令。

安装和更新软件

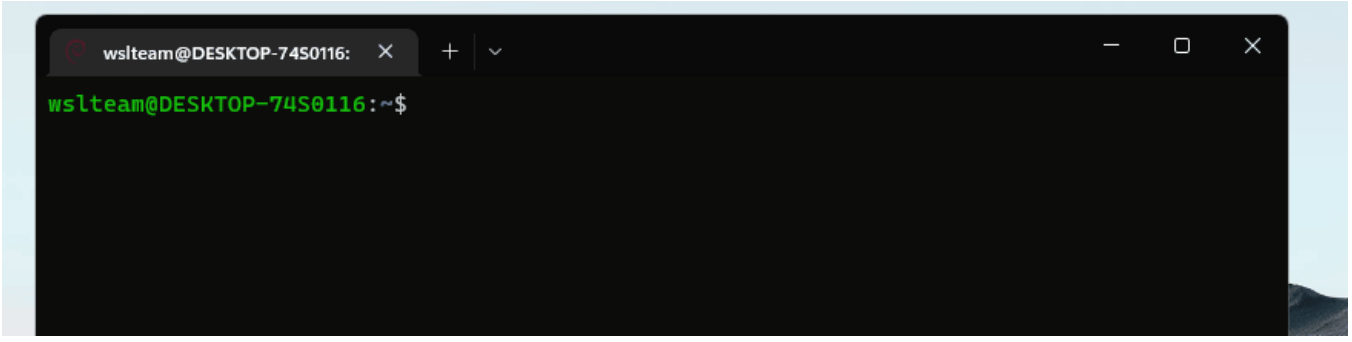
可以使用运行分发版的首选包管理器直接从命令行安装和更新软件程序。

例如，在 Ubuntu 中，首先通过运行 `sudo apt update` 来更新可用的软件列表。 然后，您可以使用 `sudo apt-get install` 命令直接获取软件，然后输入您要安装的程序名称:

```bash
sudo apt-get install <app_name>
```

若要更新已安装的程序，可以运行：

```bash
sudo apt update && sudo apt upgrade
```



> **提示**
>
> Linux 的不同发行版通常具有不同的包管理器，需要使用特定于关联包管理器的安装命令。 例如，Arch Linux 的主包管理器被称为 pacman，安装命令将是 `sudo pacman -S <app_name>`。 openSUSE 的主包管理器叫做 Zypper，安装命令是 `sudo zypper install <app_name>`。 Alpine 的主包管理器称为 Alpine Package Keeper （apk），安装命令将是 `sudo apk add <app_name>`。 Red Hat 分发版（如 CentOS）的两个主要包管理器是 YUM 和 RPM，安装命令可以是 `sudo yum install <app_name>` 或 `sudo rpo -i <app_name>`。 请参阅你正在使用的分发文档，了解可用于安装和更新软件的工具。

处理文件和目录

若要查看当前正在使用的目录的路径，请使用 `pwd` 以下命令：

```Bash
pwd
```

若要创建新目录，请使用命令 `mkdir`，后接您要创建的目录名称。

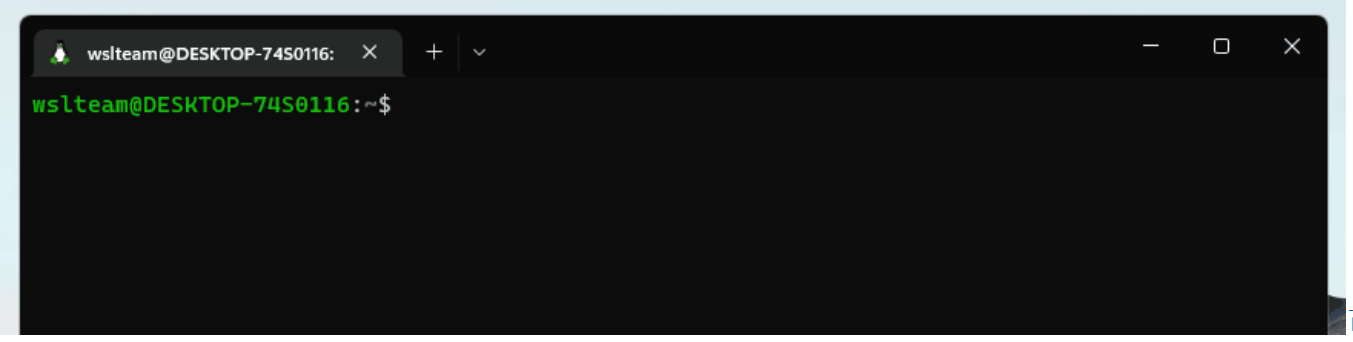
```Bash
mkdir hello_world
```

若要更改目录，请使用 `cd` 命令后跟要导航到的目录的名称：

```Bash
cd hello_world
```

若要查看当前正在使用的目录中的内容，请键入 `ls` 命令行：

```Bash
ls
```



默认情况下，该 `ls` 命令将仅打印所有文件和目录的名称。 若要获取其他信息，例如上次修改文件时或文件权限，请使用标志 `-l`：

```Bash
ls -l
```

可以通过 `touch` 命令创建一个新文件，后面接要创建的文件名称：

```Bash
```

```
touch hello_world.txt
```

可以使用任何下载的图形文本编辑器或 VS Code 远程 – WSL 扩展编辑文件。 可以 在此处了解有关 VS Code 入门的详细信息。

如果希望直接从命令行编辑文件，则需要使用命令行编辑器，例如 `vim`， `emacs` 或 `nano`。 许多分发版都安装了其中一个或多个程序，但你始终可以按照 上述 指南中概述的安装说明来安装这些程序。

若要使用首选的编辑方法编辑文件，只需运行程序名称，然后运行要编辑的文件的名称：

Bash

```
code hello_world.txt
```

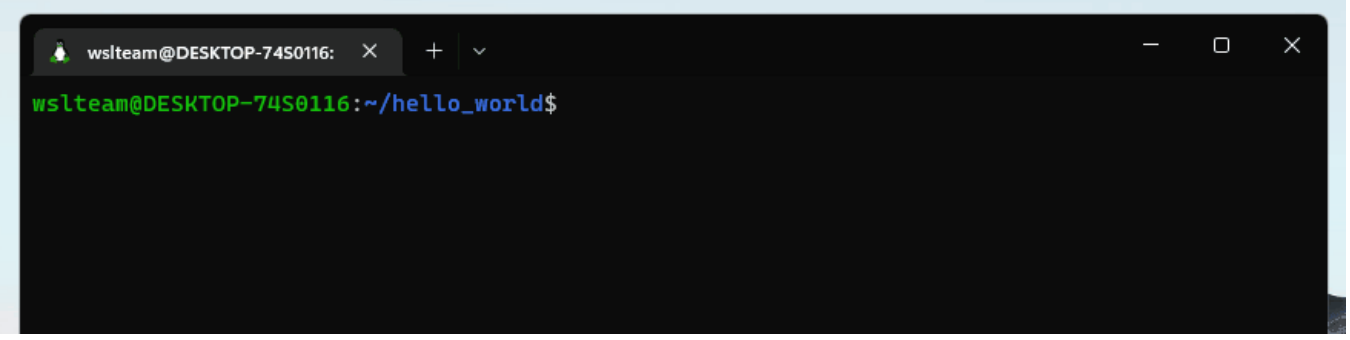
Bash

```
notepad.exe hello_world.txt
```

要在命令行中查看文件的内容，请使用 `cat` 命令，然后指定您要读取的文件：

Bash

```
cat hello_world.txt
```



使用管道和重定向运算符

管道 `|` 将一个命令的输出作为输入重定向到另一个命令。 例如， `lhscmd | rhscmd` 将输出定向 `lhscmd` 到 `rhscmd`。 可以通过多种方式使用管道，通过命令行快速完成任务。 下面是有关如何使用管道的几个简单示例。

假设你想要快速对文件的内容进行排序。 请看下面的 fruits.txt 示例：

Bash

```
$ cat fruits.txt
Orange
Banana
Apple
Pear
Plum
Kiwi
Strawberry
Peach
```

可以使用管道快速对此列表进行排序：

Bash

```
$ cat fruits.txt | sort
Apple
Banana
Kiwi
Orange
Peach
Pear
Plum
Strawberry
```

默认情况下，命令的 `cat` 输出将发送到标准输出;但是， `|` 允许我们改为将输出作为输入重定向到另一个命令 `sort`。

另一个用例是搜索。 可以使用 `grep` 一个有用的命令来搜索特定搜索字符串的输入。

Bash

```
cat fruits.txt | grep P
Pear
Plum
Peach
```

还可以使用重定向运算符，例如 `>` 将输出传递给文件或流。 例如，如果要使用 fruit.txt的已排序内容创建新的 .txt 文件：

Bash

```
$ cat fruits.txt | sort > sorted_fruit.txt
$ cat sorted_fruit.txt
Apple
Banana
Kiwi
Orange
Peach
Pear
```

```
Plum
Strawberry
```

默认情况下，命令的 `sort` 输出将发送到标准输出;但是， `>` 运算符允许我们改为将输出重定向到名为 sorted_fruits.txt的新文件中。

可以通过多种有趣的方式使用管道和重定向运算符，以便直接从命令行完成任务。

建议的内容

- Microsoft Learn：Bash 入门
- 面向初学者的命令行
- Microsoft Learn：WSL 入门

Last updated on 2025/08/07

跨 Windows 和 Linux 文件系统工作

在 Windows 和 Linux 文件系统之间工作时，需要牢记一些注意事项。 本指南概述了其中的一些内容，包括混合使用 Windows 和基于 Linux 的命令的互作性支持的一些示例。

跨文件系统的文件存储和性能

我们建议避免在不同操作系统之间使用文件，除非您有特定原因这样做。 为了获得最快的性能速度，如果在 Linux 命令行（Ubuntu、OpenSUSE 等）中工作，请将文件存储在 WSL 文件系统中。 如果在 Windows 命令行（PowerShell、命令提示符）中工作，请将文件存储在 Windows 文件系统中。

例如，在存储 WSL 项目文件时：

- 使用 Linux 文件系统根目录：`/home/<user name>/Project`
- 不是 Windows 文件系统的根目录：`/mnt/c/Users/<user name>/Project$` 或 `C:\Users\<user name>\Project`

在 WSL 命令行的文件路径中看到 `/mnt/` 时，这意味着您正在使用已挂载的驱动器。 因此，在 WSL 命令行中装载时，Windows 文件系统 C：\ 驱动器 （`C:\Users\<user name>\Project`） 将如下所示： `/mnt/c/Users/<user name>/Project$` 可以将项目文件存储在装载的驱动器上，但如果将它们直接存储在驱动器上 `\\wsl$` ，性能速度会提高。

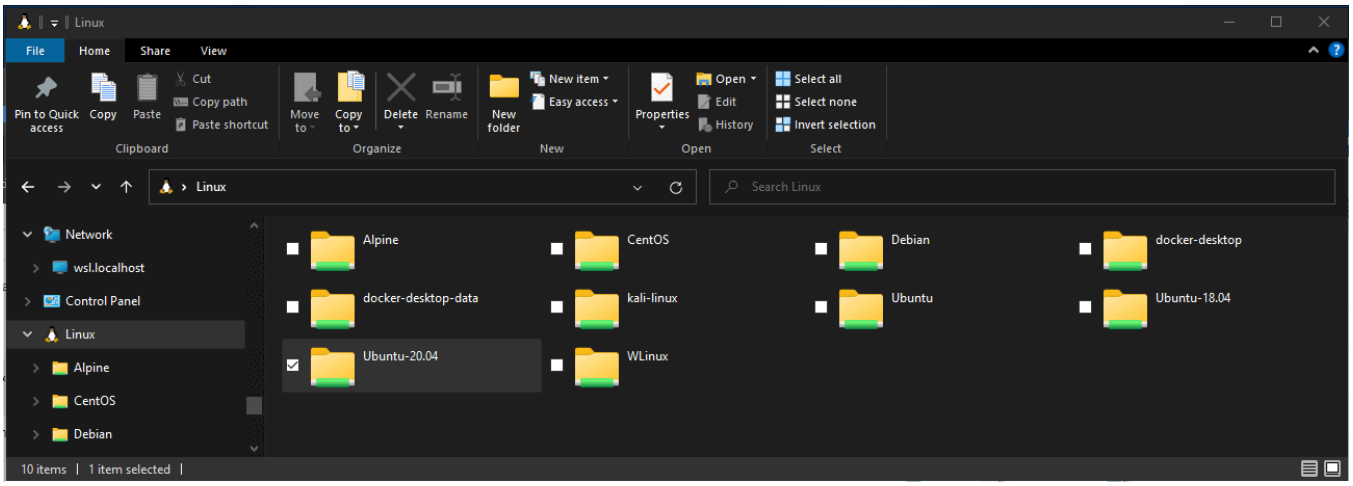
在 Windows 文件资源管理器中查看当前目录

可以通过从命令行打开 Windows 文件资源管理器来查看存储文件的目录，方法是：

```Bash
explorer.exe .
```

或者，也可以使用命令： `powershell.exe /c start .` 请务必在命令末尾添加句点以打开当前目录。

若要查看 Windows 文件资源管理器中所有可用的 Linux 发行版及其根文件系统，请在地址栏中输入： `\\wsl$`



文件名和目录区分大小写

区分大小写决定了在文件名或目录中，大写字母（FOO.txt）和小写字母（foo.txt）是作为区分（区分大小写）处理的，还是作为等效（不区分大小写）处理的。 Windows 和 Linux 文件系统在处理大小写敏感性时采用不同的方式，其中 Windows 不区分大小写，而 Linux 区分大小写。 若要详细了解如何调整大小写敏感度，尤其是在使用 WSL 装载磁盘时，请参阅 调整大小写敏感度 指南。

Windows 和 Linux 命令之间的互作性

Windows 和 Linux 工具和命令可与 WSL 互换使用。

- 从 Linux 命令行（例如 Ubuntu）运行 Windows 工具（即 notepad.exe）。
- 从 Windows 命令行（例如 PowerShell）运行 Linux 工具（例如 grep）。
- 在 Linux 和 Windows 之间共享环境变量。 （内部版本 17063+）

从 Windows 命令行运行 Linux 工具

使用 `wsl <command>` （或 `wsl.exe <command>`） 从 Windows 命令提示符（CMD）或 PowerShell 运行 Linux 二进制文件。

例如：

PowerShell

```
C:\temp> wsl ls -la
<- contents of C:\temp ->
```

以这种方式调用的二进制文件：

- 使用与当前 CMD 或 PowerShell 提示符相同的工作目录。

- 以 WSL 默认用户身份运行。
- 具有与调用进程和终端相同的 Windows 管理权限。

以下 `wsl` Linux 命令（或 `wsl.exe`），将像在 WSL 中运行的任何命令一样处理。 sudo、管道和文件重定向等功能运行正常。

使用 sudo 更新默认 Linux 分发版的示例：

PowerShell

```
C:\temp> wsl sudo apt-get update
```

运行此命令后，将列出默认的 Linux 分发用户名，系统会要求你输入密码。 正确输入密码后，分发版将下载更新。

混合 Linux 和 Windows 命令

下面是使用 PowerShell 混合 Linux 和 Windows 命令的几个示例。

若要使用 Linux 命令 `ls -la` 列出文件和 PowerShell 命令 `findstr` 来筛选包含“git”的字词的结果，请合并这些命令：

PowerShell

```
wsl ls -la | findstr "git"
```

若要使用 PowerShell 命令 `dir` 列出文件和 Linux 命令 `grep` 来筛选包含“git”的单词的结果，请合并这些命令：

PowerShell

```
C:\temp> dir | wsl grep git
```

若要使用 Linux 命令 `ls -la` 列出文件和 PowerShell 命令 `> out.txt` 将该列表打印到名为“out.txt”的文本文件，请合并这些命令：

PowerShell

```
C:\temp> wsl ls -la > out.txt
```

传入 `wsl.exe` 的命令将转发到 WSL 进程，无需修改。 文件路径必须以 WSL 格式指定。

若要使用 Linux 命令 `ls -la` 列出 Linux 文件系统路径中的 `/proc/cpuinfo` 文件，请使用 PowerShell：

PowerShell

```
C:\temp> wsl ls -la /proc/cpuinfo
```

若要使用 Linux 命令 `ls -la` 列出 Windows 文件系统路径中的 `C:\Program Files` 文件，请使用 PowerShell：

PowerShell

```
C:\temp> wsl ls -la "/mnt/c/Program Files"
```

从 Linux 运行 Windows 工具

WSL 可以使用 `[tool-name].exe` 直接从 WSL 命令行运行 Windows 工具。 例如，`notepad.exe`。

应用程序以这种方式运行具有以下属性：

- 保留工作目录作为 WSL 命令提示符（在大多数情况下 -- 异常如下所述）。
- 具有与 WSL 进程相同的权限。
- 以活动 Windows 用户身份运行。
- 在 Windows 任务管理器中显示为直接从 CMD 提示符执行。

在 WSL 中运行的 Windows 可执行文件的处理方式与本机 Linux 可执行文件类似，管道、重定向，甚至后台处理都能如预期正常工作。

要运行 Windows 工具 `ipconfig.exe`，请先在 Linux 发行版（例如 Ubuntu）中使用 Linux 工具 `grep` 筛选“IPv4”结果，再使用 Linux 工具 `cut` 去除列字段。输入：

Bash

```
ipconfig.exe | grep IPv4 | cut -d: -f2
```

让我们尝试一个混合 Windows 和 Linux 命令的示例。 打开 Linux 分发版（即Ubuntu） 并创建文本文件： `touch foo.txt`。 现在，使用 Linux 命令 `ls -la` 列出直接文件及其创建详细信息，以及 Windows PowerShell 工具 `findstr.exe` 来筛选结果，因此只有 `foo.txt` 文件显示在结果中：

Bash

```
ls -la | findstr.exe foo.txt
```

在 Windows 操作系统中，工具必须包含文件扩展名，匹配文件名的大小写，并且为可执行文件。 非可执行文件，包括批处理脚本。 可以用 `dir` 命令来运行`cmd.exe /C`这样的 CMD 本机命

令。

例如，通过输入以下内容列出 Windows 文件系统 C：\ 目录的内容：

```Bash
cmd.exe /C dir
```

或使用`ping`命令向 microsoft.com 网站发送回显请求：

```Bash
ping.exe www.microsoft.com
```

参数将传递给未修改的 Windows 二进制文件。 例如，以下命令将在`C:\temp\foo.txt`中打开`notepad.exe`：

```Bash
notepad.exe "C:\temp\foo.txt"
```

这也会起作用。

```Bash
notepad.exe C:\\temp\\foo.txt
```

使用 WSLENV 在 Windows 和 WSL 之间共享环境变量

WSL 和 Windows 共享一个特殊的环境变量， `WSLENV`用于桥接在 WSL 上运行的 Windows 和 Linux 分发版。

`WSLENV`变量的属性：

- 它共享;它存在于 Windows 和 WSL 环境中。
- 它是在 Windows 和 WSL 之间共享的环境变量列表。
- 它可以设置环境变量的格式，以在 Windows 和 WSL 中正常工作。
- 它可以协助 WSL 与 Win32 之间的交互。

> **注意**

在 17063 之前，只有 WSL 可以访问 `PATH` 的 Windows 环境变量（因此可以从 WSL 下启动 Win32 可执行文件）。 从 17063 开始， `WSLENV` 开始受支持。 WSLENV 区分大小写。

WSLENV 标志

在 `WSLENV` 中，有四个标志可以影响环境变量的翻译方式。

`WSLENV` 标志：

- `/p` - 转换 WSL/Linux 样式路径和 Win32 路径之间的路径。
- `/l` - 指示环境变量是路径列表。
- `/u` - 指示仅当从 Win32 运行 WSL 时，才应包含此环境变量。
- `/w` - 指示仅当从 WSL 运行 Win32 时，才应包含此环境变量。

可以根据需要组合标志。

阅读有关 WSLENV 的详细信息，包括常见问题解答和将 WSLENV 的值设置为其他预定义环境 var 的串联的示例，每个后缀都带有斜杠后跟标志，以指定应如何转换值并使用脚本传递变量。 本文还包括使用 Go 编程语言 设置开发环境的示例，该语言配置为在 WSL 和 Win32 之间共享 GOPATH。

禁用互操作性

用户可以通过以 root 身份运行以下命令来禁用为单个 WSL 会话运行 Windows 工具的功能：

Bash

```
echo 0 > /proc/sys/fs/binfmt_misc/WSLInterop
```

若要重新启用 Windows 二进制文件，请退出所有 WSL 会话并重新运行 bash.exe 或以 root 身份运行以下命令：

Bash

```
echo 1 > /proc/sys/fs/binfmt_misc/WSLInterop
```

禁用互操作不会在 WSL 会话之间持续有效 —— 启动新会话时将再次启用互操作。

Last updated on 2025/06/10

WSL 中的高级设置配置

wsl.conf 和 .wslconfig 文件用于在 WSL 中配置高级设置，这些设置将在 WSL VM 启动时应用。 `wsl.conf` 用于根据 WSL 发行版应用设置，`.wslconfig` 用于将全局设置应用于 WSL。 可以阅读有关以下差异的详细信息。

展开表

方面	`.wslconfig`	`wsl.conf`
Scope	适用于所有 WSL 的常规设置	仅限 WSL 分发的设置
配置	WSL 中的功能启用、为 WSL 2 提供支持的虚拟机设置（RAM、要启动的内核、CPU 数等）	WSL 中的分发设置，例如启动选项、DrvFs 自动装载、网络、与 Windows 系统的互作性、系统使用情况和默认用户
位置	`%UserProfile%\.wslconfig`，在 WSL 分发之外	`/etc/wsl.conf`，而在 WSL 分发中

目前，所有 `.wslconfig` 设置仅适用于 WSL 2 分发版。 了解如何检查正在运行的 WSL 版本。

配置更改的 8 秒规则

必须等到运行你的 Linux 发行版的子系统完全停止运行并重启，配置设置更新才会显示。 这通常需要关闭发行版 shell 的所有实例后大约 8 秒。

如果启动分发版（例如 Ubuntu），请修改配置文件，关闭分发版，然后重启它，你可能会假设配置更改已立即生效。 但当前情况并非如此，因为子系统可能仍在运行。 在重新启动之前，必须等待子系统停止，以便有足够的时间让系统识别你的更改。 可以通过使用 PowerShell 和以下命令来检查关闭 Linux 发行版 (shell) 后其是否仍在运行：`wsl --list --running`。 如果分发版未运行，则会收到响应："没有正在运行的分发版。"现在可以重启分发版，以查看应用的配置更新。

命令 `wsl --shutdown` 是重新启动 WSL 2 发行版的快速途径，但它会关闭所有正在运行的发行版，因此请谨慎使用。 还可以使用 `wsl --terminate <distroName>` 来终止正在运行的特定发行版。

wsl.conf

使用 **wsl.conf** 为 WSL 1 或 WSL 2 上运行的每个 Linux 发行版按各个发行版配置**本地设置**。

- 作为 unix 文件存储在发行版的 `/etc` 目录中。
- 用于针对每个发行版配置设置。 在此文件中配置的设置将仅应用于包含存储此文件的目录的特定 Linux 发行版。

- 可用于由 WSL 1 或 WSL 2 版本运行的发行版。
- 若要访问已安装发行版的`/etc`目录，请使用发行版的命令行通过`cd /`访问根目录，然后使用`ls`列出文件或者`explorer.exe .`在Windows文件资源管理器中查看。 该目录路径应类似于：`/etc/wsl.conf`。

> ⓘ **注意**
>
> 使用 wsl.conf 文件调整每个发行版设置的功能仅适用于 Windows 版本 17093 及更高版本。

wsl.conf 的配置设置

wsl.conf 文件会针对每个发行版配置设置。 *（有关 WSL 2 发行版的全局配置，请参阅 .wslconfig）*。

wsl.conf 文件支持四个部分：`automount`、`network`、`interop` 和 `user`。 *（在 .ini 文件约定之后建模，密钥在 .gitconfig 文件等部分下声明。）*有关存储 wsl.conf 文件位置的信息，请参阅 wsl.conf。

systemd 支持

许多 Linux 发行版（包括 Ubuntu）默认运行“systemd”，WSL 最近添加了对此系统/服务管理器的支持，因此 WSL 更类似于在裸机上使用你最爱的 Linux 发行版。 需要 WSL 的 0.67.6+ 版本才能启用 systemd。 使用命令 `wsl --version` 检查 WSL 版本。 如果需要更新，可以在 Microsoft Store 中获取最新版本的 WSL。 了解更多信息，请参阅博客公告。

若要启用 systemd，请使用 `wsl.conf` 通过管理员权限在文本编辑器中打开 `sudo` 文件，并将以下行添加到 `/etc/wsl.conf`：

Bash

```
[boot]
systemd=true
```

然后，您需要从 PowerShell 使用 `wsl.exe --shutdown` 来关闭您的 WSL 发行版，以便重启 WSL 实例。 发行版重启后，systemd 应该就会运行了。 可以使用 `systemctl list-unit-files --type=service` 命令确认，这将显示您的服务状态。

自动装载设置

wsl.conf 节标签：`[automount]`

展开表

密钥	值	默认	说明
`enabled`	布尔	`true`	`true` 导致固定驱动器（即 `C:/` 或 `D:/`）自动装载到 DrvFs 中的 `/mnt` 下。 `false` 表示驱动器不会自动装载，但你仍可以手动或通过 `fstab` 装载驱动器。
`mountFsTab`	布尔	`true`	`true` 设置启动 WSL 时要处理的 `/etc/fstab`。 `/etc/fstab` 是一个文件，可在其中声明其他文件系统，例如 SMB 共享。 因此，在启动时，可以在 WSL 中自动装载这些文件系统。
`root`	字符串	`/mnt/`	设置固定驱动器要自动装载到的目录。 默认情况下，此设置设置为 `/mnt/`，因此 Windows 文件系统 `C:\` 已装载到 `/mnt/c/`。 如果将`/mnt/`更改为`/windir/`，您应会看到固定`C:\`装载到`/windir/c`。
`options`	以逗号分隔的值的列表，例如 uid、gid 等，请参阅下面的自动装载选项	Null	下面列出了自动装载选项值，它们追加到了默认的 DrvFs 装载选项字符串中。 **只能指定特定于 DrvFs 的选项。**

对于所有自动装载的驱动器，这些自动装载选项会应用为装载选项。 若要仅更改特定驱动器的选项，请改用 `/etc/fstab` 文件。 通常由装载二进制文件分析成标志的选项不受支持。 若要显式指定这些选项，必须在 `/etc/fstab` 中包含要对其执行此操作的每个驱动器。

自动装载选项

为 Windows 驱动器 (DrvFs) 设置不同的装载选项可以控制为 Windows 文件计算文件权限的方式。 可使用以下选项：

展开表

密钥	说明	默认
`uid`	用于所有文件的所有者的用户 ID	WSL 发行版的默认用户 ID（首次安装时，默认为 `1000`）
`gid`	用于所有文件的所有者的组 ID	WSL 发行版的默认组 ID（首次安装时，默认为 `1000`）
`umask`	要对所有文件和目录排除的权限的八进制掩码	`022`
`fmask`	要对所有文件排除的权限的八进制掩码	`000`

密钥	说明	默认
`dmask`	要对所有目录排除的权限的八进制掩码	`000`
`metadata`	是否将元数据添加到 Windows 文件以支持 Linux 系统权限	`disabled`
`case`	确定被视为区分大小写的目录以及使用 WSL 创建的新目录是否将设置标志。 有关选项的详细说明，请参阅区分大小写。 选项包括 `off`、`dir` 或 `force`。	`off`

默认情况下，WSL 将 `uid` 和 `gid` 设置为默认用户的值。 例如，在 Ubuntu 中，默认用户为 `uid=1000`： `gid=1000` 如果此值用于指定其他 `gid` 或 `uid` 选项，将覆盖默认用户值。 否则，默认值将始终被追加。

上述 umask、fmask 等选项仅在 Windows 驱动器以元数据方式挂载时适用。 默认情况下，未启用元数据。 可 在此处找到有关此内容的详细信息。

> ⓘ **注意**
>
> 权限掩码在应用到文件或目录之前通过一个逻辑或操作进行设置。

什么是 DrvFs?

DrvFs 是 WSL 的文件系统插件，旨在支持 WSL 和 Windows 文件系统之间的互操作。 DrvFs 支持 WSL 在 /mnt 下装载包含支持的文件系统的驱动器，例如 /mnt/c、/mnt/d 等。有关在装载 Windows 或 Linux 驱动器或目录时指定默认区分大小写行为的详细信息，请参阅区分大小写页。

网络设置

wsl.conf 节标签： `[network]`

展开表

密钥	值	默认	说明
`generateHosts`	布尔	`true`	`true` 将 WSL 设置为生成 `/etc/hosts`。 `hosts` 文件包含主机名与 IP 地址对应的静态映射。
`generateResolvConf`	布尔	`true`	`true` 将 WSL 设置为生成 `/etc/resolv.conf`。 `resolv.conf` 包含一个 DNS 列表，能够将给定的主机名解析为其 IP 地址。
`hostname`	字符	Windows 主机名	设置要用于 WSL 发行版的主机名。

密钥	值	默认	说明
	串		

互操作设置

wsl.conf 节标签：`[interop]`

这些选项在 Insider 版本 17713 和更高版本中可用。

展开表

密钥	值	默认	说明
`enabled`	布尔	`true`	设置此键可确定 WSL 是否支持启动 Windows 进程。
`appendWindowsPath`	布尔	`true`	设置此键将确定 WSL 是否将 Windows 路径元素添加到 `$PATH` 环境变量。

用户设置

wsl.conf 节标签：`[user]`

这些选项在版本 18980 及更高版本中可用。

展开表

密钥	值	默认	说明
`default`	字符串	首次运行时创建的初始用户名	设置此键指定在首次启动 WSL 会话时以哪个用户身份运行。

启动设置

启动设置仅在 Windows 11 和 Server 2022 上可用。

wsl.conf 节标签：`[boot]`

展开表

密钥	值	默认	说明
`command`	字符串	Null	你希望在 WSL 实例启动时运行的命令字符串。 此命令以根用户身份运行。 例如`service docker start`。
`protectBinfmt`	布尔	`true`	防止 WSL 在启用 systemd 时生成 systemd 单元。

GPU 设置

wsl.conf 节标签：`[gpu]`

展开表

密钥	值	默认	说明
`enabled`	布尔	`true`	允许 Linux 应用程序通过准虚拟化访问 Windows GPU。

时间设置

wsl.conf 节标签：`[time]`

展开表

密钥	值	默认	说明
`useWindowsTimezone`	布尔	`true`	设置此密钥将使 WSL 使用 Windows 中设置的时区并与之同步。

示例 wsl.conf 文件

下面的 `wsl.conf` 示例文件展示了一些可以使用的配置选项。 在此示例中，分发版为 Ubuntu-20.04，文件路径为 `\\wsl.localhost\Ubuntu-20.04\etc\wsl.conf`。

Bash

```
# Automatically mount Windows drive when the distribution is launched
[automount]

# Set to true will automount fixed drives (C:/ or D:/) with DrvFs under the root
directory set above. Set to false means drives won't be mounted automatically, but
need to be mounted manually or with fstab.
enabled=true

# Sets the directory where fixed drives will be automatically mounted. This example
changes the mount location, so your C-drive would be /c, rather than the default
/mnt/c.
```

```
root = /

# DrvFs-specific options can be specified.
options = "metadata,uid=1003,gid=1003,umask=077,fmask=11,case=off"

# Sets the `/etc/fstab` file to be processed when a WSL distribution is launched.
mountFsTab=true

# Network host settings that enable the DNS server used by WSL 2. This example
changes the hostname, sets generateHosts to false, preventing WSL from the default
behavior of auto-generating /etc/hosts, and sets generateResolvConf to false,
preventing WSL from auto-generating /etc/resolv.conf, so that you can create your
own (ie. nameserver 1.1.1.1).
[network]
hostname=DemoHost
generateHosts=false
generateResolvConf=false

# Set whether WSL supports interop processes like launching Windows apps and adding
path variables. Setting these to false will block the launch of Windows processes
and block adding $PATH environment variables.
[interop]
enabled=false
appendWindowsPath=false

# Set the user when launching a distribution with WSL.
[user]
default=DemoUser

# Set a command to run when a new WSL instance launches. This example starts the
Docker container service.
[boot]
command=service docker start
```

.wslconfig

使用 **.wslconfig** 为 WSL 上运行的所有已安装的发行版配置**全局设置**。

- 默认情况下，.wslconfig 文件不存在。 它必须创建并存储在`%UserProfile%`目录中才能应用这些设置。
- 用于在所有运行 WSL 2 版本的已安装 Linux 发行版中配置全局设置。
- **只能用于 WSL 2 运行的发行版**。 作为 WSL 1 运行的发行版不受此配置的影响，因为它们不作为虚拟机运行。
- 要访问您的 `%UserProfile%` 目录，可以在 PowerShell 中使用 `cd ~` 进入您的主目录（通常是您的用户配置文件 `C:\Users\<UserName>`），或者可以打开 Windows 文件资源管理器并在地址栏中输入 `%UserProfile%`。 该目录路径应类似于：`C:\Users\<UserName>\.wslconfig`。

WSL 将检测这些文件是否存在，读取内容，并在每次启动 WSL 时自动应用配置设置。 如果文件缺失或格式错误（标记格式不正确），则 WSL 将继续正常启动，而不应用配置设置。

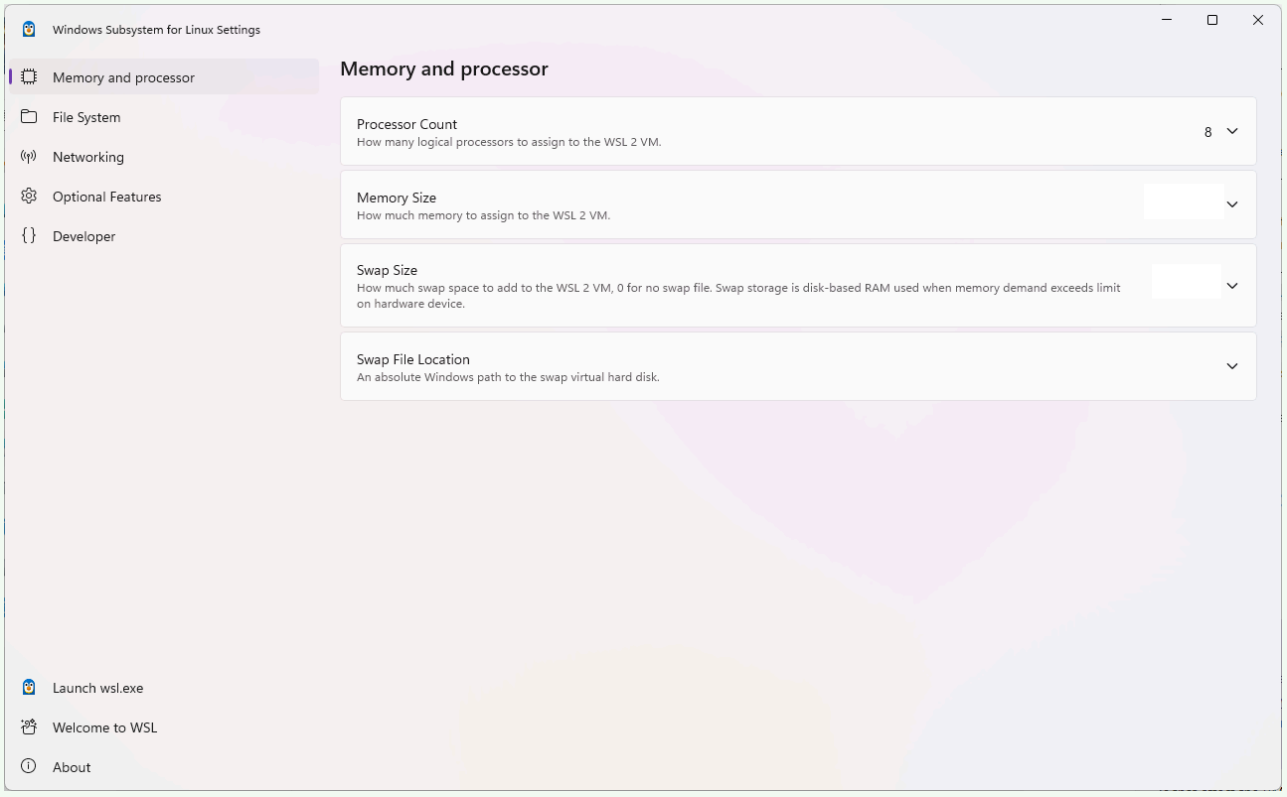
.wsl.conf 的配置设置

.wslconfig 文件对所有在 WSL 2 上运行的 Linux 发行版进行全局设置配置。 （有关每个发行版配置的信息，请参阅 wsl.conf）。

有关 .wslconfig 文件的存储位置的信息，请参见 .wslconfig。

> **注意**
>
> 使用 `.wslconfig` 进行全局配置的选项仅适用于在 Windows 版本 19041 及更高版本中作为 WSL 2 运行的发行版。 请记住，可能需要运行 `wsl --shutdown` 来关闭 WSL 2 VM，然后重启 WSL 实例以使这些更改生效。

> **提示**
>
> 建议直接在 WSL 设置中修改 WSL 配置，而不是手动编辑 .wslconfig 文件。 可以在“开始”菜单中找到 WSL 设置。



此文件可以包含以下选项，它们会影响为任何 WSL 2 发行版提供支持的 VM：

主要 WSL 设置

.wslconfig 节标签：`[wsl2]`

展开表

密钥	值	默认	说明
`kernel`	路径	Microsoft 内置内核提供的收件箱	自定义 Linux 内核的绝对 Windows 路径。
`kernelModules`	路径	自定义 Linux 内核模块 VHD 的绝对 Windows 路径。	
`memory`	大小	Windows 上总内存的 50%	要分配给 WSL 2 VM 的内存量。
`processors`	数字	Windows 上相同数量的逻辑处理器	要分配给 WSL 2 VM 的逻辑处理器数量。
`localhostForwarding`	布尔	`true`	一个布尔值，用于指定绑定到 WSL 2 VM 中的通配符或 localhost 的端口是否应可通过 `localhost:port` 从主机连接。
`kernelCommandLine`	字符串	没有	其他内核命令行参数。
`safeMode`	布尔	`false`	在“安全模式”中运行 WSL，这会禁用许多功能，应用于恢复处于错误状态的发行版。 仅适用于 Windows 11 和 WSL 版本 0.66.2+。
`swap`	大小	Windows 上 25% 的内存大小四舍五入到最接近的 GB	要向 WSL 2 VM 添加的交换空间量，0 表示无交换文件。 交换存储是当内存需求超过硬件设备上的限制时使用的基于磁盘的 RAM。
`swapFile`	路径	`%Temp%\swap.vhdx`	交换虚拟硬盘的绝对 Windows 路径。
`pageReporting`	布尔	`true`	设置使 Windows 能够回收分配给 WSL 2 虚拟机的未使用的内存。
`guiApplications`	布尔	`true`	一个布尔值，用于在 WSL 中打开或关闭对 GUI 应用程序 (WSLg) 的支持。
`debugConsole`[1]	布尔	`false`	一个布尔值，用于在 WSL 2 发行版实例启动时打开显示 `dmesg` 内容的输出控制台窗口。
`maxCrashDumpCount`	数字	`10`	设置将出于调试目的保留的最大故障转储文件数。 WSL 保留的默认数字为 10。 超过此限制后，将自动删除较旧的故障转储，为新转储腾出空间。 设置最大值有助于减少这些崩溃文件使用的磁盘空间量。

密钥	值	默认	说明
`nestedVirtualization`[1]	布尔	`true`	用于打开或关闭嵌套虚拟化的布尔值，使其他嵌套 VM 能够在 WSL 2 中运行。
`vmIdleTimeout`[1]	数字	`60000`	VM 在关闭之前处于空闲状态的毫秒数。
`dnsProxy`	布尔	`true`	`networkingMode = NAT`仅适用于 . 布尔值，通知 WSL 将 Linux 中的 DNS 服务器配置为主机上的 NAT。 `false`设置为将 DNS 服务器从 Windows 镜像到 Linux。
`networkingMode`[1][2]	字符串	`NAT`	可用值为： `none`、、 （ `net` `bridged`已弃用） `mirrored`和`virtioproxy`。 如果值为 `none`，则 WSL 网络断开连接。 如果值为 `net` 或未知值，则使用 NAT 网络模式（从 WSL 2.3.25 开始，如果 NAT 网络模式失败，则会回退到使用 VirtioProxy 网络模式）。 如果值为 `bridged`，则使用桥接网络模式（此模式已标记为自 WSL 2.4.5 以来已弃用）。 如果值为， `mirrored`则使用镜像网络模式。 如果值为 `virtioproxy`，则使用 VirtioProxy 网络模式。
`firewall`[1][2]	布尔	`true`	如果设置为 true，则 Windows 防火墙规则以及特定于 Hyper-V 流量的规则可以筛选 WSL 网络流量。
`dnsTunneling`[1][2]	布尔	`true`	更改将 DNS 请求从 WSL 代理到 Windows 的方式
`autoProxy`[1]	布尔	`true`	强制 WSL 使用 Windows 的 HTTP 代理信息
`defaultVhdSize`	大小	`1099511627776` （1 TB）	设置存储 Linux 发行版（例如 Ubuntu）文件系统的虚拟硬盘 (VHD) 大小。 可用于限制分发文件系统允许占用的最大大小。

具有“**path**”值的条目必须是带转义反斜杠的 Windows 路径，例如： `C:\\Temp\\myCustomKernel`

默认情况下，具有`size`值的条目默认为B（字节），单位可以省略。 若要使用其他单位，必须追加大小单位，例如： `8GB` 或 `512MB`。

[1]：仅在 Windows 11 上可用。

}：需要 Windows 11 版本 22H2 或更高版本。

实验性设置

这些设置是试验性功能的选择加入预览，我们的目标是将来将其设为默认设置。

.wslconfig 节标签：`[experimental]`

展开表

密钥	值	默认	说明
`autoMemoryReclaim`	字符串	`dropCache`	可用值为： `disabled`、 `gradual`和 `dropCache`。 如果值为 `disabled`，则将禁用 WSL 自动内存回收。 如果值为此值 `gradual`，则缓存的内存将缓慢且自动回收。 如果值为 `dropCache` 或未知值，将立即回收缓存内存。
`sparseVhd`	布尔	`false`	当设置为 `true` 时，任何新创建的 VHD 将自动设置为稀疏。
`bestEffortDnsParsing`[1][2]	布尔	`false`	仅当 `wsl2.dnsTunneling` 设置为 `true` 时才适用。 设置为`true`时，Windows 将从 DNS 请求中提取查询，并尝试解析它，忽略未知记录。
`dnsTunnelingIpAddress`[1][2]	字符串	`10.255.255.254`	仅当 `wsl2.dnsTunneling` 设置为 `true` 时才适用。 指定在启用 DNS 隧道时将在 Linux `resolv.conf` 文件中配置的名称服务器。
`initialAutoProxyTimeout`[1]	字符串	`1000`	仅当 `wsl2.autoProxy` 设置为 `true` 时才适用。 配置启动 WSL 容器时，WSL 等待检索 HTTP 代理信息的时长（以毫秒为单位）。 如果代理设置在此时间之后解决，则必须重新启动 WSL 实例才能使用检索到的代理设置。
`ignoredPorts`[1][2]	字符串	Null	仅当 `wsl2.networkingMode` 设置为 `mirrored` 时才适用。 指定 Linux 应用程序可以绑定到哪些端口（即使该端口已在 Windows 中使用）。 通过此设置，应用程序能够仅侦听 Linux 中的流量端口，因此即使该端口在 Windows 上用于其他用途，这些应用程序也不会被阻止。 例如，WSL 将允许绑定到 Linux for Docker Desktop 中的端口 53，因为它只侦听来自 Linux 容器中的请求。 应在逗号分隔的列表中设置格式，例如：`3000,9000,9090`
`hostAddressLoopback`[1][2]	布尔	`false`	仅当 `wsl2.networkingMode` 设置为 `mirrored` 时才适用。 如果设置为 `true`，将会允许容器通过分配给主机的 IP 地址连接到主机，或允许主机通过该地址连接到容器。 始终可以使用`127.0.0.1` 环回地址，此选项还允许使用所有额外分配的本地 IP 地址。 仅支持分配给主机的 IPv4 地址。

[1]：仅在 Windows 11 上可用。

}：需要 Windows 11 版本 22H2 或更高版本。

示例 .wslconfig 文件

下面的 `.wslconfig` 示例文件展示了一些可以使用的配置选项。 在此示例中，文件路径为 `%UserProfile%\.wslconfig`。

Bash

```
# Settings apply across all Linux distros running on WSL 2
[wsl2]

# Limits VM memory to use no more than 4 GB, this can be set as whole numbers using
GB or MB
memory=4GB

# Sets the VM to use two virtual processors
processors=2

# Specify a custom Linux kernel to use with your installed distros. The default
kernel used can be found at https://github.com/microsoft/WSL2-Linux-Kernel
kernel=C:\\temp\\myCustomKernel

# Specify the modules VHD for the custum Linux kernel to use with your installed
distros.
kernelModules=C:\\temp\\modules.vhdx

# Sets additional kernel parameters, in this case enabling older Linux base images
such as Centos 6
kernelCommandLine = vsyscall=emulate

# Sets amount of swap storage space to 8GB, default is 25% of available RAM
swap=8GB

# Sets swapfile path location, default is %UserProfile%\AppData\Local\Temp\swap.vhdx
swapfile=C:\\temp\\wsl-swap.vhdx

# Disable page reporting so WSL retains all allocated memory claimed from Windows
and releases none back when free
pageReporting=false

# Turn on default connection to bind WSL 2 localhost to Windows localhost. Setting
is ignored when networkingMode=mirrored
localhostforwarding=true

# Disables nested virtualization
nestedVirtualization=false

# Turns on output console showing contents of dmesg when opening a WSL 2 distro for
debugging
debugConsole=true

# Sets the maximum number of crash dump files to retain (default is 5)
maxCrashDumpCount=10
```

```
# Enable experimental features
[experimental]
sparseVhd=true
```

其他资源

- Windows 命令行博客：自动配置 WSL
- Windows 命令行博客：Chmod/Chown、DrvFs、文件元数据

Last updated on 2025/08/07

WSL 的文件权限

本页详细介绍了如何在适用于 Linux 的 Windows 子系统中解释 Linux 文件权限，尤其是在 NT 文件系统上的 Windows 内部访问资源时。 本文档假定你基本了解 Linux 文件系统权限结构和 umask 命令 。

从 WSL 访问 Windows 文件时，文件权限是从 Windows 权限计算的，或者从 WSL 添加到文件的元数据中读取。 默认情况下不启用此元数据。

Windows 文件中的 WSL 元数据

在 WSL 中启用元数据作为装载选项时，可以添加和解释 Windows NT 文件上的扩展属性以提供 Linux 文件系统权限。

WSL 可以添加四个 NTFS 扩展属性：

展开表

属性名称	DESCRIPTION
$LXUID	用户所有者 ID
$LXGID	组所有者 ID
$LXMOD	文件模式（文件系统权限八进制和类型，例如：0777）
$LXDEV	设备（如果是设备文件）

此外，任何不是常规文件或目录的文件（例如符号链接、FIDO、块设备、unix 套接字和字符设备）也具有 NTFS 重新分析点。 这样就可以更快地确定给定目录中的文件类型，而无需查询其扩展属性。

文件访问方案

下面是使用适用于 Linux 的 Windows 子系统以不同方式访问文件时如何确定权限的说明。

从 Linux 访问 Windows 驱动器文件系统中的文件（DrvFS）

从 WSL 访问 Windows 文件时，很可能通过 `/mnt/c` 这些方案。

从现有 Windows 文件读取文件权限

结果取决于文件是否已有现有元数据。

DrvFS 文件没有元数据（默认值）

如果文件没有与之关联的元数据，我们将 Windows 用户的有效权限转换为读/写/执行位，并将其设置为与用户、组和其他值相同的值。 例如，如果你的 Windows 用户帐户具有读取和执行访问权限，但对文件没有写入访问权限，则这会显示为 `r-x` 用户、组和其他帐户。 如果该文件在 Windows 中设置了“只读”属性，则我们不会在 Linux 中授予写入访问权限。

该文件具有元数据

如果文件存在元数据，我们只需使用这些元数据值，而不是转换 Windows 用户的有效权限。

使用 chmod 更改现有 Windows 文件的文件权限

结果取决于文件是否已有现有元数据。

chmod 文件没有元数据（默认值）

如果删除文件的所有写入属性，则设置 Windows 文件中的“只读”属性，因为这是与 Linux 中的 SMB（服务器消息块）客户端 CIFS（Common Internet File System）相同的行为。

chmod 文件具有元数据

Chmod 将根据文件现有的元数据更改或添加元数据。

请记住，即使元数据显示这种情况，你也不能给自己更多的访问权限，而不是你在 Windows 上拥有的访问权限。 例如，可以将元数据设置为显示你有权使用 `chmod 777`的文件写入权限，但如果尝试访问该文件，你仍无法写入该文件。 这要归功于互作性，因为对 Windows 文件的任何读取或写入命令都通过 Windows 用户权限路由。

在 DriveFS 中创建文件

结果取决于是否启用了元数据。

未启用元数据（默认值）

新创建文件的 Windows 权限与在 Windows 中创建文件时没有特定安全描述符时的权限相同，它将继承父级的权限。

已启用元数据

文件的权限位设置为遵循 Linux umask，该文件将随元数据一起保存。

哪个 Linux 用户和 Linux 组拥有该文件?

结果取决于文件是否已有现有元数据。

用户文件没有元数据（默认值）

在默认方案中，自动装载 Windows 驱动器时，我们指定任何文件的用户 ID（UID）设置为 WSL 用户的用户 ID，并将组 ID （GID） 设置为 WSL 用户的主体组 ID。

用户文件具有元数据

元数据中指定的 UID 和 GID 将作为文件的用户所有者和组所有者应用。

使用 从 Windows 访问 Linux 文件

通过 `\\wsl$` 访问 Linux 文件将使用 WSL 分发版的默认用户。 因此，任何访问 Linux 文件的 Windows 应用都将具有与默认用户相同的权限。

创建新文件

在 Windows 中的 WSL 分发版内创建新文件时，将应用默认的 umask。 默认的 umask 是 `022`，或者换句话说，它允许除对组和其他组的写入权限之外的所有权限。

从 Linux 访问 Linux 根文件系统中的文件

在 Linux 根文件系统中创建、修改或访问的任何文件都遵循标准 Linux 约定，例如将 umask 应用到新创建的文件。

配置文件权限

可以使用 wsl.conf 中的装载选项在 Windows 驱动器中配置文件权限。 装载选项允许设置 `umask` `dmask`和`fmask`权限掩码。 将 `umask` 应用于所有文件，将 `dmask` 仅应用于目录， `fmask` 并且仅应用于文件。 然后，这些权限掩码将在应用于文件时通过逻辑 OR作进行，例如：如果你有一个 `umask` 值 `023` 和一个 `fmask` 值 `022` ，则生成的文件权限掩码将是 `023`。

了解详细信息： 使用 wsl.conf 按分发配置选项。

Last updated on 2025/08/15

使用 WSL 访问网络应用程序

使用网络应用和 WSL 时，需要注意一些注意事项。 默认情况下，WSL 使用 基于 NAT 的体系结构，建议尝试新的 镜像网络模式 来获取最新的功能和改进。

标识 IP 地址

确定用于通过 WSL 运行的 Linux 分发的 IP 地址时，需要考虑两种情况：

方案一： 从 Windows 主机的角度来看，你想要查询通过 WSL2 运行的 Linux 分发版 IP 地址，以便 Windows 主机上的程序可以连接到分发（实例）中运行的服务器程序。

Windows 主机可以使用命令：

PowerShell

```
wsl.exe --distribution <DistroName> hostname -i
```

如果查询默认分布，则可以省略指定分发的命令的此部分： `-d <DistroName>` 请务必使用小写 `-i` 标志。

在后台，主机命令 wsl.exe 启动目标实例并执行 Linux 命令 `hostname --ip-addresses`。 然后，此命令会将 WSL 实例的 IP 地址打印到 `STDOUT`。 然后，文本 `STDOUT` 内容将中继回 wsl.exe。 最后，wsl.exe 显示该输出到命令行。

典型的输出可能是：

PowerShell

```
172.30.98.229
```

方案二： 通过 WSL2（实例）在 Linux 分发中运行的程序想要知道 Windows 主机的 IP 地址，以便 Linux 程序可以连接到 Windows 主机服务器程序。

WSL2 Linux 用户可以使用命令：

Bash

```
ip route show | grep -i default | awk '{ print $3}'
```

典型的输出可能是：

PowerShell

```
172.30.96.1
```

因此， `172.30.96.1` 此示例中是 Windows 的主机 IP 地址。

> ⓘ **注意**
>
> 当 WSL2 使用默认 **NAT 网络模式**运行时，通常需要上述 IP 地址查询作。 当 WSL2 运行在新的 **镜像模式**下时，Windows 主机和 WSL2 虚拟机 (VM) 可以相互连接，使用（127.0.0.1）作为目标地址，因此不需要使用查询对方的 IP 地址。

默认网络模式：NAT

默认情况下，WSL 使用基于 NAT（网络地址转换）的体系结构进行网络。 使用基于 NAT 的网络体系结构时，请记住以下注意事项：

从 Windows 访问 Linux 网络应用（localhost）

如果要在 Linux 分发版中构建网络应用（例如，在 NodeJS 或 SQL Server 上运行的应用），则可以使用 `localhost` Windows 应用（如 Edge 或 Chrome Internet 浏览器）访问它（就像平时一样）。

从 Linux 访问 Windows 网络应用（主机 IP）

如果想要从 Linux 分发版（即 ie Ubuntu）访问在 Windows 上运行的网络应用（例如 NodeJS 或 SQL Server 上运行的应用），则需要使用主机的 IP 地址。 虽然这不是一种常见情况，但可以按照以下步骤使其发挥作用。

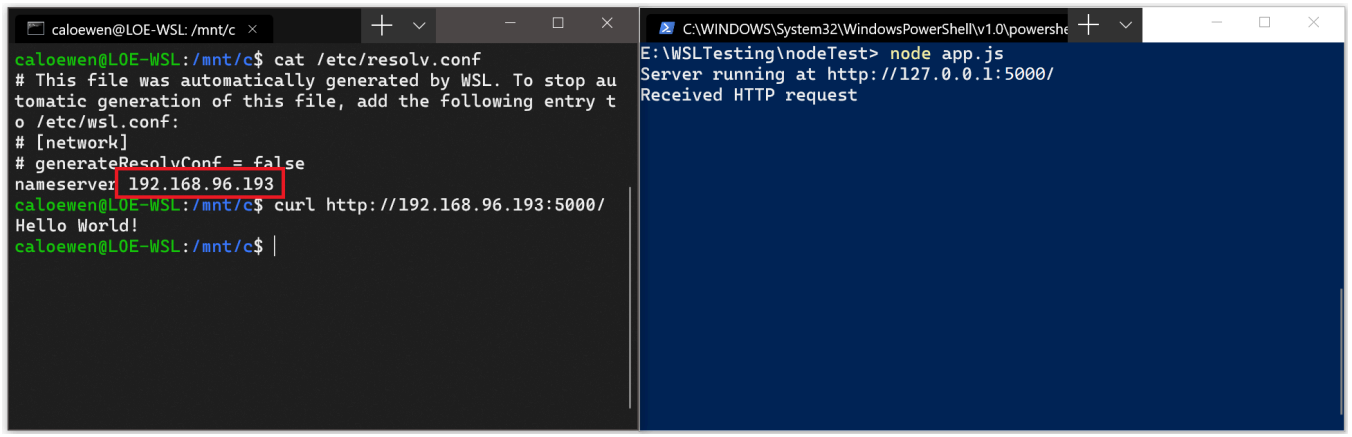
1. 通过从 Linux 分发版运行以下命令来获取主机的 IP 地址：

   Bash

   ```
   ip route show | grep -i default | awk '{ print $3}'`
   ```

2. 使用复制的 IP 地址连接到任何 Windows 服务器。

下图显示了通过 curl 连接到在 Windows 中运行的 Node.js 服务器的示例。



通过远程 IP 地址进行连接

使用远程 IP 地址连接到应用程序时，它们将被视为来自局域网（LAN）的连接。 这意味着需要确保应用程序可以接受 LAN 连接。

例如，可能需要将应用程序绑定到`0.0.0.0`，而不是`127.0.0.1`。 在使用 Flask 的 Python 应用示例中，可以使用以下命令完成此作。 `app.run(host='0.0.0.0')` 在进行这些更改时，请记住安全性，因为这将允许来自 LAN 的连接。

从局域网（LAN）访问 WSL 2 分发版

使用 WSL 1 分发版时，如果计算机已设置为由 LAN 访问，则可以在 LAN 上访问在 WSL 中运行的应用程序。

这不是 WSL 2 中的默认情况。 WSL 2 具有具有其自己的唯一 IP 地址的虚拟化以太网适配器。 目前，若要启用此工作流，需要执行与常规虚拟机相同的步骤。 （我们正在研究改善此体验的方法。

下面是使用 Netsh 接口 portproxy Windows 命令添加一个端口代理的示例，该代理侦听主机端口并将该端口代理连接到 WSL 2 VM 的 IP 地址。

PowerShell

```
netsh interface portproxy add v4tov4 listenport=<yourPortToForward>
listenaddress=0.0.0.0 connectport=<yourPortToConnectToInWSL> connectaddress=(wsl
hostname -I)
```

在此示例中，需要更新 `<yourPortToForward>` 到端口号，例如 `listenport=4000`。 `listenaddress=0.0.0.0` 表示将从 ANY IP 地址接受传入请求。 侦听地址指定要侦听的 IPv4 地址，并且可以更改为包括：IP 地址、计算机 NetBIOS 名称或计算机 DNS 名称的值。 如果未指定地址，则默认值为本地计算机。 需要将 `<yourPortToConnectToInWSL>` 值更新为要在其中连接 WSL 的端口号，例如 `connectport=4000`。 最后，`connectaddress`该值必须是通过 WSL 2（WSL

2 VM 地址）安装的 Linux 分发版的 IP 地址，可通过输入以下命令找到该值： `wsl.exe hostname -I`

因此，此命令可能如下所示：

PowerShell

```
netsh interface portproxy add v4tov4 listenport=4000 listenaddress=0.0.0.0
connectport=4000 connectaddress=192.168.101.100
```

若要获取 IP 地址，请使用：

- `wsl hostname -I` 用于通过 WSL 2 安装的 Linux 发行版的 IP 地址（WSL 2 VM 地址）
- `cat /etc/resolv.conf` 是从 WSL 2 (WSL 2 VM) 视角看到的 Windows 计算机的 IP 地址

使用 `listenaddress=0.0.0.0` 将侦听所有 IPv4 端口。

> **注意**
>
> 使用带主机名命令的小写“i”将生成与使用大写“I”不同的结果。 `wsl hostname -i` 是本地计算机（127.0.1.1 是用于诊断的占位符地址），而 `wsl hostname -I` 将返回其他计算机看到的本地计算机的 IP 地址，并应用于标识通过 WSL 2 运行的 Linux 发行版的 `connectaddress`。

IPv6 访问

- `wsl hostname -i` 用于通过 WSL 2 安装的 Linux 发行版的 IP 地址（WSL 2 VM 地址）
- `ip route show | grep -i default | awk '{ print $3}'` 是从 WSL 2 (WSL 2 VM) 视角看到的 Windows 计算机的 IP 地址

使用 `listenaddress=0.0.0.0` 将侦听所有 IPv4 端口。

镜像模式网络

在运行 Windows 11 22H2 及更高版本的计算机上，你可以在networkingMode=mirrored文件中[wsl2].wslconfig进行设置，以启用镜像模式网络。 启用此项会将 WSL 更改为全新的网络体系结构，该体系结构的目标是将 Windows 上的网络接口“镜像”到 Linux，以添加新的网络功能并提高兼容性。

以下是启用此模式的当前优势：

- IPv6 支持

- 使用 localhost 地址 `127.0.0.1`从 Linux 内部连接到 Windows 服务器。 不支持 IPv6 本地主机地址`::1`
- 改进了 VPN 的网络兼容性
- 多播支持
- 直接从局域网 (LAN) 连接到 WSL

> **注意**
>
> 在 PowerShell 窗口中以管理员权限运行以下命令，以配置 Hyper-V 防火墙设置，使其允许入站连接：
>
> PowerShell
>
> ```
> Set-NetFirewallHyperVVMSetting -Name '{40E0AC32-46A5-438A-A0B2-2B479E8F2E90}' -DefaultInboundAction Allow
> ```
>
> 或
>
> PowerShell
>
> ```
> New-NetFirewallHyperVRule -Name "MyWebServer" -DisplayName "My Web Server" -Direction Inbound -VMCreatorId '{40E0AC32-46A5-438A-A0B2-2B479E8F2E90}' -Protocol TCP -LocalPorts 80
> ```

此新模式解决了使用基于 NAT（网络地址转换）体系结构的网络问题。 在 GitHub 上的 WSL 产品存储库 中查找已知问题或提交反馈。

DNS 隧道

在运行 Windows 11 22H2 或更高版本的机器上，`dnsTunneling`功能默认启用（可以在文件中的[wsl2].wslconfig找到），它通过 WSL 内的虚拟化功能响应 DNS 请求，而不是通过网络数据包请求 DNS。 此功能旨在提高与 VPN 和其他复杂网络设置的兼容性。

自动代理

在运行 Windows 11 22H2 及更高版本的计算机上，在文件中的设置 autoProxy=true[wsl2].wslconfig会强制 WSL 使用 Windows 的 HTTP 代理信息。 如果已在 Windows 中设置了代理，则启用此功能也会使该代理在 WSL 中自动设置。

WSL 和防火墙

在运行 Windows 11 22H2 及更高版本（WSL 2.0.9 及更高版本）的计算机上，默认情况下将打开 Hyper-V 防火墙功能。 这将确保：

- 请参阅 具有高级安全性的 Windows Defender 防火墙，详细了解将自动应用于 WSL 的 Windows 安全功能。
- 请参阅 “配置 Hyper-V 防火墙”，了解有关在本地和通过 Intune 等联机工具应用这些规则和设置的详细信息。

Last updated on 2025/08/07

使用 systemd 通过 WSL 管理 Linux 服务

适用于 Linux 的 Windows 子系统（WSL）现在支持 systemd，它是一种 init 系统和服务管理器，许多流行的 Linux 发行版，例如 Ubuntu、Debian 等都在使用。 （什么是系统？）。

初始系统默认值最近已不再是 SystemV，使用 默认值安装的 `wsl --install`。 除当前版本的 Ubuntu 以外的 Linux 分发版仍可能使用 WSL init，类似于 SystemV init。 若要更改为 systemd，请参阅 如何启用 systemd。

Linux 中的 systemd 是指什么?

根据 systemd.io：“systemd 是 Linux 系统的基本构建基块套件。 它提供一个系统和服务管理器，该管理器作为 PID 1 运行并启动系统的其余部分。”

Systemd 主要是 init 系统和服务管理器，包括按需启动守护程序、装载和自动装载点维护、快照支持以及使用 Linux 控制组进行进程跟踪等功能。

大多数主要的 Linux 分发版现在都运行 systemd，因此在 WSL 上启用它可使体验更接近于使用裸机 Linux。 请参阅带 systemd 演示的视频公告或下面的 systemd 使用示例，详细了解 systemd 提供的功能。

如何启用systemd?

systemd 现在是将使用 默认值安装的 `wsl --install`。

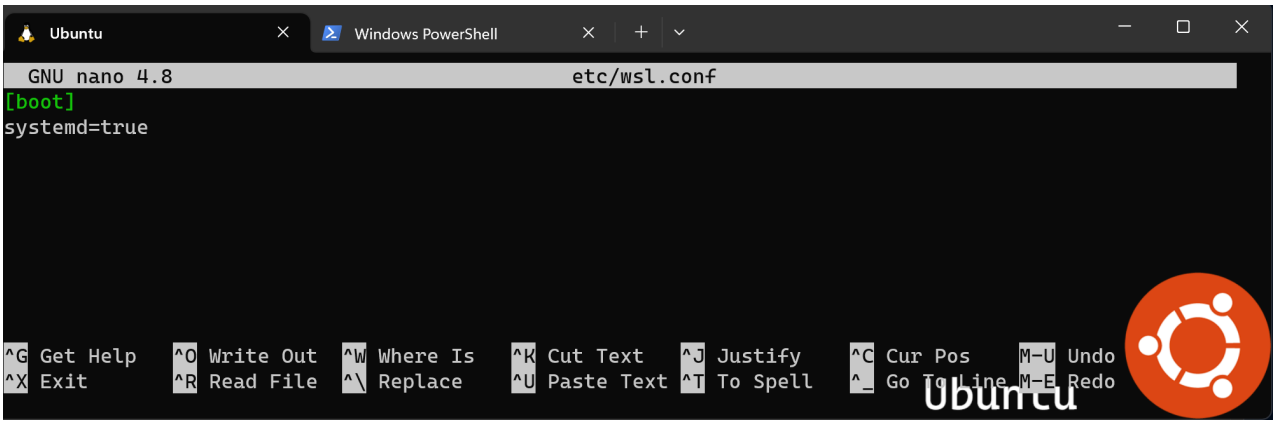
若要为 WSL 2 上运行的任何其他 Linux 发行版启用 systemd（更改默认值，使其不再使用 systemv 初始值）：

1. 确保 WSL 版本为 0.67.6 或更高版本：
   - 若要检查，请运行 `wsl --version`;如果命令引发 `Invalid command line option: --version` 错误，则必须更新 WSL;
   - 若要更新、运行 `wsl --update` 或 从 Microsoft Store 下载最新版本。
2. 打开 Linux 分发版的命令行并输入 `cd /` 以访问根目录，然后 `ls` 列出文件。 你将看到一个名为“etc”的目录，其中包含分发的 WSL 配置文件。 打开此文件，以便通过输入：`nano /etc/wsl.conf`，使用 Nano 文本编辑器进行更新。
3. 在 `wsl.conf` 文件中添加以下行，你现在已打开此文件来更改用于 systemd 的初始值：

```
Bash
```

```
[boot]
systemd=true
```

在 WSL 2 上启用 systemd



在 WSL 2 上启用 systemd

4. 退出 Nano 文本编辑器（Ctrl + X，键入 Y 以保存更改，并使用 `enter` 键进行确认）。

5. 然后，需要关闭 Linux 分发版。 可以使用 PowerShell 中的命令 `wsl.exe --shutdown` 重新启动所有 WSL 实例。

6. 重启 Linux 分发版后，systemd 将运行。 可以使用命令 `systemctl status` 来验证它，以显示运行 *状态的* 和命令 `systemctl list-unit-files --type=service`，这将显示与 Linux 分发版关联的任何服务的状态。

如果 Linux 分发版为 Debian/Ubuntu/Kali Rolling，则不仅应该已安装 systemd 包，而且还确保已安装 systemd-sysv 包。

Bash

```
sudo apt-get update -y && sudo apt-get install systemd systemd-sysv -y
```

详细了解 WSL 中的高级设置配置，包括 `wsl.conf`（特定于分发）和 `.wslconfig`（全局）配置文件之间的差异、如何更新自动装载设置等。

Systemd 演示视频

Microsoft与 Canonical 合作，为 WSL 提供系统支持。 请查看 Craig Loewen（Microsoft 的 WSL 项目经理）和 Oliver Smith（Canonical 的 Ubuntu on WSL 项目经理）宣布 systemd 支持并展示其启用功能的一些演示。

https://learn-video.azurefd.net/vod/player?show=tabs-vs-spaces&ep=wsl-partnering-with-canonical-to-support-systemd&locale=zh-cn&embedUrl=%2Fwindows%2Fwsl%2Fsystemd

- systemd 支持博客公告
- Oliver 的教程基于这些在 Ubuntu 博客上的演示 - 包括“在 WSL 上使用 snap 几分钟内创建 Nextcloud 实例”、“使用 LXD 管理您的 Web 项目”，以及 “在 Ubuntu WSL 上以 systemd 服务运行 .Net Echo Bot”
- GitHub 上的 Craig microk8s 演示

Systemd 示例

依赖于系统的 Linux 应用程序的几个示例包括：

- snap：Canonical 开发的软件打包和部署系统，适用于使用 Linux 内核和 systemd init 系统的操作系统。 这些包称为“snaps”，用于生成快照的命令行工具称为“Snapcraft”，可以下载/安装快照的中心存储库称为“Snap Store”，运行快照所需的守护程序（从应用商店下载、装载到位置、限制和运行应用）称为“snapd”。 整个系统有时称为“snappy”。请尝试运行命令：`snap install spotify`。
- microk8s：一种开源、低运维、简化生产环境的 Kubernetes，可自动部署、扩展和管理容器化应用。 按照说明 在 WSL2 上安装 MicroK8s，查看 入门教程，或观看关于 在 Windows 上使用 MicroK8s 和 WSL 2 的 Kubernetes 视频。
- systemctl：一个命令行实用工具，用于控制和检查系统，并帮助与 Linux 分发版上的服务进行交互。 请尝试以下命令：`systemctl list-units --type=service` 查看哪些服务可用及其状态。

一些相关教程演示了使用系统的方法：

- 了解和使用 systemd
- systemd 基础知识：使用服务、单元和日记
- 如何在 Ubuntu 20.04 上使用 Systemd 对进程进行沙箱化

启用 systemd 如何影响 WSL 架构?

启用对 systemd 的支持需要更改 WSL 架构。 由于 systemd 需要 PID 1，在 Linux 分发中启动的 WSL 初始化进程将成为系统的子进程。 由于 WSL init 进程负责为 Linux 和 Windows 组件之间的通信提供基础结构，因此更改此层次结构需要重新考虑 WSL init 进程做出的一些假设。 必须进行额外修改，以确保实现正常关闭（因为关闭现在由 systemd 控制），并与 WSLg 兼容，该组件用于运行 Linux 图形用户界面 (GUI)，或者显示在窗口中而非命令行中的 Linux 应用。

此外，请务必注意，在这些更改中，系统服务不会使 WSL 实例保持活动状态。 WSL 实例将保持活动状态，就像在此更新之前一样，你可以在这篇 2017 年的后台任务支持博客文章 中阅读详细信息。

Last updated on 2025/04/03

导入要与 WSL 一起使用的任何 Linux 分发版

可以使用适用于 Linux 的 Windows 子系统（WSL）内的任何 Linux 分发版（即使它在 Microsoft 应用商店 中不可用），方法是使用 tar 文件导入它。

本文介绍如何导入 Linux 发行版 CentOS，以便通过 Docker 容器获取其 tar 文件来与 WSL 一起使用。 此过程可以应用于导入任何 Linux 分发版。

获取用于分发的 tar 文件

首先，需要获取包含分发的所有 Linux 二进制文件的 tar 文件。

可以通过多种方式获取 tar 文件，其中包括：

- 下载提供的 tar 文件。 可以在 Alpine Linux 下载 站点的“**Mini Root Filesystem**”部分找到 Alpine 的示例。
- 查找 Linux 分发容器，并将实例导出为 tar 文件。 以下示例将使用 CentOS 容器 显示此过程。

获取 CentOS 示例的 tar 文件

在此示例中，我们将使用 WSL 分发中的 Docker 获取 CentOS 的 tar 文件。

先决条件

- 必须启用 WSL，并安装运行 WSL 2 的 Linux 发行版。
- 必须安装已启用了 WSL 2 引擎和集成的 Docker Desktop for Windows，请参阅 Docker Desktop 许可协议，了解使用条款的更新。

从容器导出 tar 文件

1. 打开已从 Microsoft 应用商店（此示例中的 Ubuntu）安装的 Linux 分发版的命令行（Bash）。

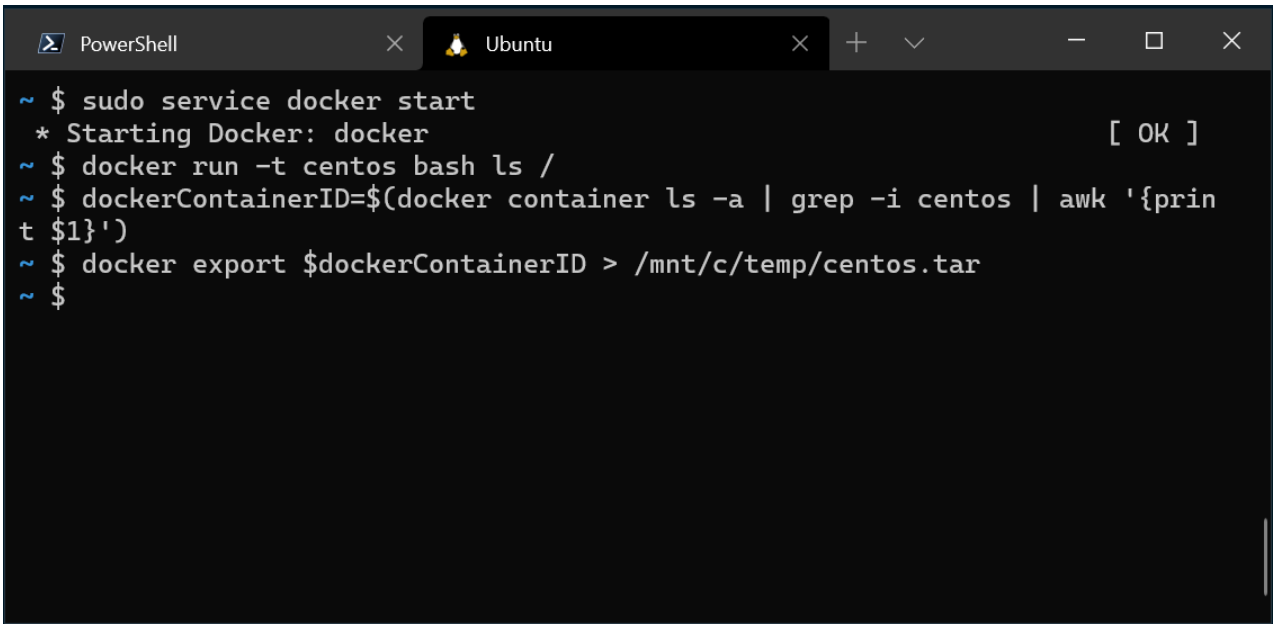
2. 确保 Docker Desktop 正在运行（或者，如果在 WSL 发行版中安装了 Docker，请使用 `sudo service docker start` 以下命令启动服务）

3. 在 Docker 中运行 CentOS 容器：

   ```Bash
   docker run -t --name wsl_export centos ls /
   ```

4. 将容器 ID 导出到装载的 c 驱动器上的 tar 文件：

```bash
docker export wsl_export > /mnt/c/temp/centos.tar
```



5. 清理容器

```bash
docker rm wsl_export
```

此过程从 Docker 容器中导出 CentOS tar 文件，以便现在可以导入它以在本地与 WSL 一起使用。

将 tar 文件导入 WSL

准备好 tar 文件后，可以使用以下命令导入它：

```powershell
wsl.exe --import <Distro> <InstallLocation> <FileName> [Options]
Options:
    --version <Version>
    --vhd
```

导入 CentOS 示例

将 CentOS 分发 tar 文件导入 WSL：

1. 打开 PowerShell 并确保已创建一个要在其中存储分发的文件夹。

   PowerShell

   ```
   cd C:\temp
   mkdir E:\wslDistroStorage\CentOS
   ```

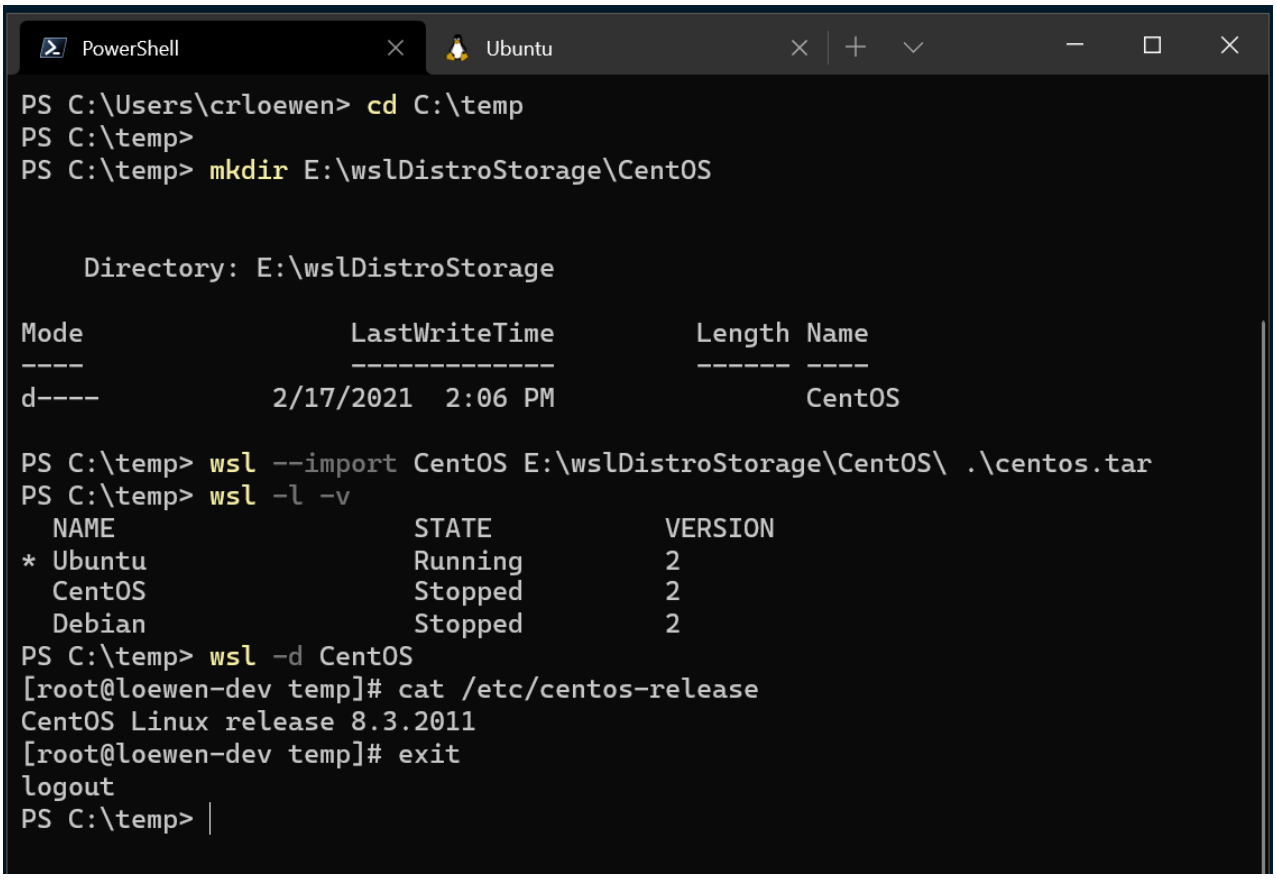
2. 使用命令 `wsl --import <Distro> <InstallLocation> <FileName>` 导入 tar 文件。

   PowerShell

   ```
   wsl --import CentOS E:\wslDistroStorage\CentOS .\centos.tar
   ```

3. 使用命令 `wsl -l -v` 检查已安装的分发版。

4. 最后，使用命令 `wsl -d CentOS` 运行新导入的 CentOS Linux 分发版。



添加特定于 WSL 的组件，例如默认用户

默认情况下，使用 `--import` 时，始终作为根用户启动。 可以设置自己的用户帐户，但请注意，设置过程会根据每个不同的 Linux 分发略有不同。

若要使用刚刚导入的 CentOS 分发设置用户帐户，请先打开 PowerShell 并启动到 CentOS，然后使用以下命令：

PowerShell

```
wsl -d CentOS
```

接下来，打开 CentOS 命令行。 使用此命令将 sudo 和密码设置工具安装到 CentOS 中，创建用户帐户，并将其设置为默认用户。 在此示例中，用户名将为“caloewen”。

> ⓘ **注意**
>
> 需要将用户名添加到 sudoers 文件，使用户能够使用 sudo。 该命令 `adduser -G wheel $myUsername` 将用户 `myUsername` 添加到滚轮组。 滚轮组中的用户会自动被授予 sudo 权限，并且可以执行需要提升权限的任务。

Bash

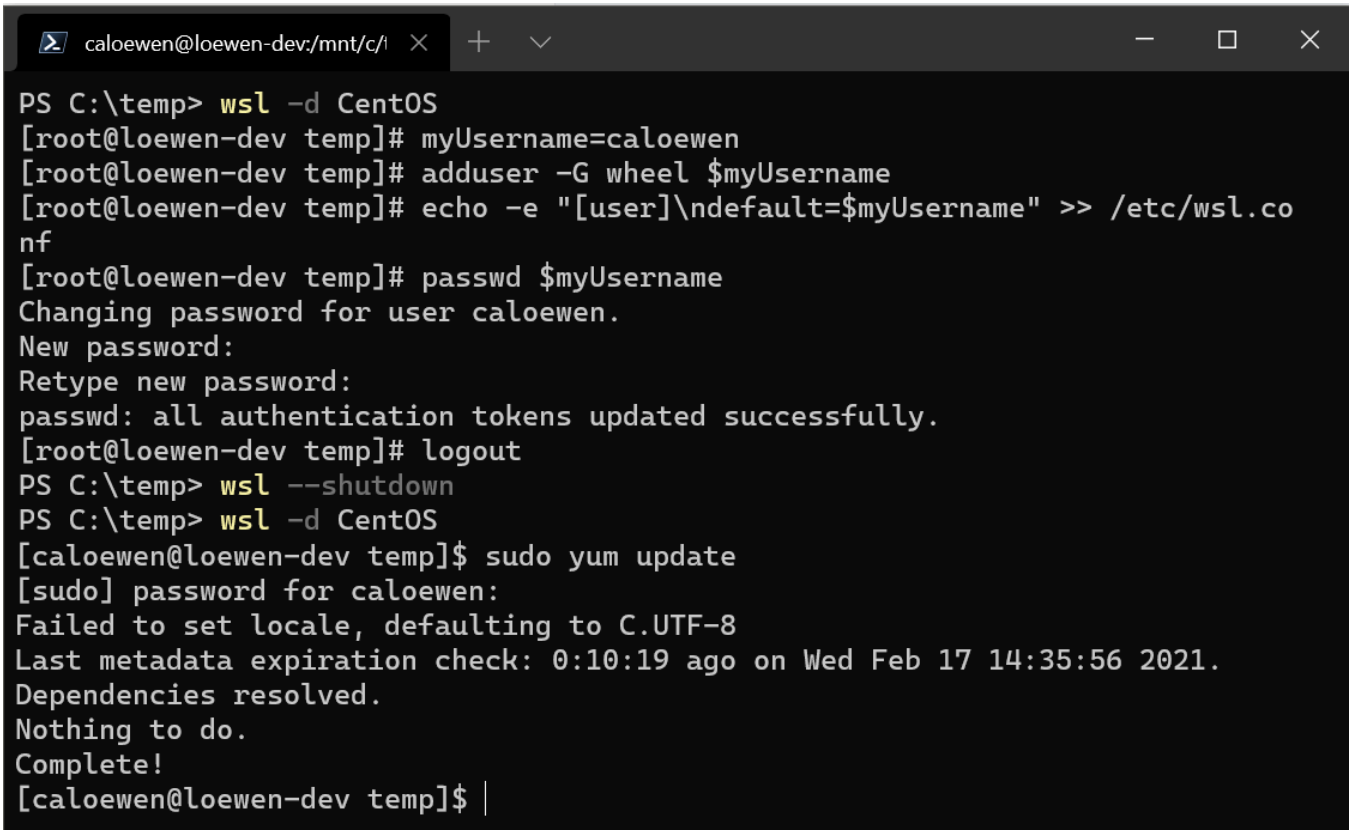
```
yum update -y && yum install passwd sudo -y
myUsername=caloewen
adduser -G wheel $myUsername
echo -e "[user]\ndefault=$myUsername" >> /etc/wsl.conf
passwd $myUsername
```

现在必须退出该实例，并确保所有 WSL 实例都终止。 再次启动分发，通过在 PowerShell 中运行以下命令来查看新的默认用户：

PowerShell

```
wsl --terminate CentOS
wsl -d CentOS
```

现在，你将看到 `[caloewen@loewen-dev]$` 基于此示例的输出。



若要详细了解如何配置 WSL 设置，请参阅 使用 .wslconfig 和 wsl.conf 配置设置。

使用自定义 Linux 分发版

可以创建自己的自定义 Linux 分发版（打包为 UWP 应用），该分发版的行为与 Microsoft 应用商店中提供的 WSL 分发版完全相同。 若要了解如何作，请参阅 为 WSL 创建自定义 Linux 分发版。

Last updated on 2025/06/10

为 WSL 生成自定义 Linux 分发版

本指南将演练创建和分发 WSL 分发（文件 `.wsl` ）的步骤。

WSL 分发有两个部分：

1. 根文件系统（以 tar 文件的形式分发）
2. 清单条目（其中包含分发元数据）

本指南仅适用于 WSL 版本 2.4.4 及更高版本。

> **注意**
>
> 有关上一个基于 appx 的分发打包说明，请参阅 **此存储库** 。

什么是 WSL 根文件系统 tar 文件?

WSL 分发版由具有 Windows 文件扩展名的 `.wsl` tar 文件定义。

TAR 文件（磁带存档简称）是一种存档文件，用于将多个文件存储在单个文件中，以便更轻松地分发或备份。 TAR 文件包含 Linux 分发版（所有分发文件）以及 WSL 配置文件的根文件系统。 WSL 配置文件告知 WIndows 操作系统如何安装和启动分发版。

拥有想要进入 WSL 分发版的 Linux 系统后，请按照以下步骤开始操作。

创建 WSL 配置文件

分发应包括两个配置文件：

1. `/etc/wsl-distribution.conf`：由分发维护者创建的文件，负责控制首次使用 WSL 启动时应如何配置 Linux 分发版。
2. `/etc/wsl.conf`：包含特定于用户的全局系统设置的文件，并控制分发的启动方式。 详细了解 WSL 配置文件。

添加 WSL 分发配置文件

分发配置文件 `/etc/wsl-distribution.conf`定义在用户首次启动时应如何配置 Linux 分发版。 此文件可用于以交互方式创建用户帐户、显示许可协议等。

下面是一个示例 `/etc/wsl-distribution.conf` 文件：

Bash

```
# /etc/wsl-distribution.conf

[oobe]
command = /etc/oobe.sh
defaultUid = 1000
defaultName = my-distro

[shortcut]
enabled = true
icon = /usr/lib/wsl/my-icon.ico

[windowsterminal]
enabled = true
ProfileTemplate = /usr/lib/wsl/terminal-profile.json
```

WSL 分发文件配置选项：

展开表

密钥	价值	违约	注释
`oobe.command`	字符串	没有	OOBE 代表现装体验。 此命令在用户首次在分发版中打开交互式 shell 时运行。 如果该命令返回非零，则被视为失败，并且用户将无法打开 shell。
`oobe.defaultUid`	整数	没有	分发以默认 UID 开头。 当脚本创建新用户时 `oobe.command` ，这非常有用。
`oobe.defaultName`	字符串	没有	分发所注册的默认名称。 此默认名称可以替换为命令： `wsl.exe --install <distro> --name <name>`
`shortcut.icon`	字符串	默认 WSL 图标	分发的“开始”菜单快捷方式中的图标。 必须 `.ico` 采用最大大小为 的格式 `10MB`
`shortcut.enabled`	布尔	是	安装分发版时是否应创建开始菜单快捷方式。
`windowsterminal.profileTemplate`	字符串	没有	用于为此分发生成Windows 终端配置文件的 JSON 模板。
`windowsterminal.enabled`	布尔	是	是否应在安装分发版时创建终端配置文件。 如果未设置`profileTemplate`，将生成默认配置文件。
`windowsterminal.profileTemplate`	字符	终端模板文件	用于为此分发生成Windows 终端配置文件的 JSON 模板。

密钥	价值	违约	注释
	串	的路径	

你需要为分发创建现装体验（OOBE） 首次运行体验。 下面是可以使用的示例 bash 脚本。 此脚本假定设置为`oobe.defaultUid` `1000`：

Bash

```
#!/bin/bash

set -ue

DEFAULT_GROUPS='adm,cdrom,sudo,dip,plugdev'
DEFAULT_UID='1000'

echo 'Please create a default UNIX user account. The username does not need to match
your Windows username.'
echo 'For more information visit: https://aka.ms/wslusers'

if getent passwd "$DEFAULT_UID" > /dev/null ; then
  echo 'User account already exists, skipping creation'
  exit 0
fi

while true; do

  # Prompt from the username
  read -p 'Enter new UNIX username: ' username

  # Create the user
  if /usr/sbin/adduser --uid "$DEFAULT_UID" --quiet --gecos ''  "$username"; then

    if /usr/sbin/usermod "$username" -aG "$DEFAULT_GROUPS"; then
      break
    else
      /usr/sbin/deluser "$username"
    fi
  fi
done
```

生成Windows 终端配置文件

安装分发版时，WSL 自动生成Windows 终端配置文件。 分发维护人员可以通过在 WSL 配置文件`windowsterminal.profileTemplate`中设置`/etc/wsl-distribution.conf`来自定义生成的配置文件。

json 文件遵循 终端配置文件 json 格式。 下面是一个示例配置文件：

JSON

```
{
  "profiles": [
    {
      "antialiasingMode": "aliased",
      "fontWeight": "bold",
      "colorScheme": "Postmodern Tango Light"
    }
  ],
  "schemes": [
    {
      "name": "Postmodern Tango Light",
      "black": "#0C0C0C",
      "red": "#C50F1F",
      "green": "#13A10E",
      "yellow": "#C19C00",
      "blue": "#0037DA",
      "purple": "#881798",
      "cyan": "#3A96DD",
      "white": "#CCCCCC",
      "brightBlack": "#767676",
      "brightRed": "#E74856",
      "brightGreen": "#16C60C",
      "brightYellow": "#F9F1A5",
      "brightBlue": "#3B78FF",
      "brightPurple": "#B4009E",
      "brightCyan": "#61D6D6",
      "brightWhite": "#F2F2F2"
    }
  ]
}
```

此文件不需要指定配置文件 `name`，也 `commandLine`不需要指定配置文件。 生成终端配置文件时，WSL 会自动添加这些配置文件。

为每个分布区添加本地设置的 WSL 配置

在分发根文件系统的上下文中，我们建议你配置系统设置，包括默认是否以按分布方式在本地设置中`/etc/wsl.conf`启动系统设置。 请参阅以下示例。

Bash

```
# /etc/wsl.conf

[boot]
systemd=true|false
```

分发作者通过将值设置为 `boot.systemd` `true` （enabled）或 `false` （未启用）来确定系统是默认启用的。

如果选择默认启用 Systemd，请参阅 Systemd 建议 。

有关所有受支持的设置，请参阅 `/etc/wsl.conf` 中的高级设置配置。

创建 tar 文件

分发和配置文件到位后，可以在 tar 文件中捕获根文件系统。

创建 tar 文件的建议方法：

```Bash
cd /path/to/rootfs
tar --numeric-owner --absolute-names -c  * | gzip --best > ../install.tar.gz
```

tar 的根目录应该是文件系统的根（而不是包含根文件系统的目录）。

建议的压缩格式为 gzip。 其他压缩格式可能会破坏与旧版 WSL 版本的兼容性。

有关应且不应包含在配置中的文件列表，请参阅配置文件 建议 。

若要获取现有 Linux 分发版的 tar 文件，请查找有关如何在 导入要用于 WSL 的任何 Linux 分发版中导出 Docker 容器的指导。

tar 文件存档准备就绪后，请参阅 在本地测试分发版 以在本地试用。

创建 .wsl 文件扩展名

最后一步是将创建的表示自定义 Linux 分发版的 tar 文件的文件扩展名通过重命名从 `.tar` 文件扩展名更改为 `.wsl` 文件扩展名。 重命名此文件扩展名会将它标记为 WSL 分发版。 将 tar 重命名 `.tar` 为 `.wsl` 后，文件将在文件资源管理器中打开（双击）时正确安装在 Windows 上。 要使此双击操作能够正常工作，`oobe.defaultName`文件中需要一个`/etc/wsl-distribution.conf`条目。

分发 WSL 分发

WSL 用户可以通过运行 `wsl --list --online` 查看可用的发行版，并可以直接使用 `wsl --install <distroName>` 安装它们（将 `<distroName>` 替换为 Linux 发行版的实际名称）。 此过程由分发清单文件控制。 可以将此清单文件添加到客户 Linux 分发版，以便将其包含在命令选项中 `wsl --install` 。

您创建并重命名为`.wsl`文件扩展名的自定义 Linux 分发 tar包可以按照您的意愿分发。 下载后，用户可以使用命令行 `wsl --install --from-file <fileLocation>` 直接安装它（将

`<fileLocation>` 替换为文件的实际位置）。 或者， `.wsl` 可以通过双击它打开自定义 WSL 分发的文件。

分发清单详细信息

分发 清单 包含有关可通过 `wsl --install <distribution>` 安装分发版的元数据。

下面列出了 `ModernDistribution` 基于 tar 的 Linux 分发版，格式如下：

JSON

```
{
    "ModernDistributions": {
        "<flavor>": [
            {
                "Name": "<version name>",
                "FriendlyName": "<friendly name>",
                "Default": true | false,
                "Amd64Url": {
                    "Url": "<tar url>",
                    "Sha256": "<tar sha256 hash>"
                },
                "Arm64Url": {
                    "Url": "<tar url>",
                    "Sha256": "<tar sha256 hash>"
                }
            },
            {
                ...
            }
        ],
        "<flavor>": [
            ...
        ]
    }
}
```

每个 `flavor` 条目都包含一个可安装分发版的列表。 可以通过风格名称（即安装默认条目）或版本名称来安装分发版。

查看命令如何使用 `wsl --install` 以下清单：

JSON

```
{
    "ModernDistributions": {
        "my-distro": [
            {
                "Name": "my-distro-v3",
```

```
                "Default": true,
                "FriendlyName": "My distribution version 3 (latest)"
                [...]
            },
            {
                "Name": "my-distro-v2",
                "Default": false,
                "FriendlyName": "My distribution version 2"
                [...]
            }
        ]
    }
}
```

示例安装命令：

PowerShell

```
wsl --install my-distro # Installs 'my-distro-v3' since it's the default for 'my-distro' flavor
wsl --install my-distro-v3 # Installs 'my-distro-v3' explicitly
wsl --install my-distro-v2 # Installs 'my-distro-v2' explicitly
```

将您的分发添加到 `wsl --install` 以供所有 WSL 用户使用

为了将您的发行版包括在命令`wsl --list --online`列表中，分发版必须满足在分发邮件列表中概述的成员资格条件。 这可确保所有列出的分发版都遵循必要的安全标准。

如果分发符合条件，并且想要将其添加到 `--install` 列表中，请在 WSL GitHub 存储库（https://github.com/microsoft/WSL）上提交拉取请求，以使用分发的详细信息更新 DistributionInfo.json 文件。 WSL 团队将审查此拉取请求。

将您的分发添加到 `wsl --install` 用于您的企业或业务团队

还可以通过在所选的 Windows 设备上编辑注册表项，使分发仅对 `wsl --install` 选择的用户组可用。

可以通过在其中 `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Lxss`创建注册表值来重写 WSL 分发清单。

- `DistributionListUrl`：重写分发清单 URL
- `DistributionListUrlAppend`：将清单 URL 中的分发版添加到可安装分发版列表

这两个注册表值都是字符串（REG_SZ），应采用 URL 格式。

从 WSL 版本 2.4.4 开始，`file://` 支持协议，以便更轻松地进行本地测试。 预期格式为：`file:///C:/path/to/file`。

在本地测试分发

若要测试分发 tar，可以使用以下示例 powershell 脚本通过新的分发替代分发清单。 首先将以下脚本保存为 `override-manifest.ps1`：

PowerShell

```powershell
#Requires -RunAsAdministrator

[cmdletbinding(PositionalBinding = $false)]
param (
    [Parameter(Mandatory = $true)][string]$TarPath,
    [string]$Flavor = "test-distro",
    [string]$Version = "test-distro-v1",
    [string]$FriendlyName = "Test distribution version 1")

Set-StrictMode -Version latest

$TarPath = Resolve-Path $TarPath
$hash = (Get-Filehash $TarPath -Algorithm SHA256).Hash


$manifest= @{
    ModernDistributions=@{
        "$Flavor" = @(
            @{
                "Name" = "$Version"
                Default = $true
                FriendlyName = "$FriendlyName"
                Amd64Url = @{
                    Url = "file://$TarPath"
                    Sha256 = "0x$hash"
                }
            })
        }
    }

$manifestFile = "$PSScriptRoot/manifest.json"
$manifest | ConvertTo-Json -Depth 5 | Out-File -encoding ascii $manifestFile


Set-ItemProperty -Path "HKLM:SOFTWARE\Microsoft\Windows\CurrentVersion\Lxss" -Name
DistributionListUrl -Value "file://$manifestFile" -Type String -Force
```

然后在提升的 Powershell 中运行以下命令来配置本地清单：

PowerShell

```
.\override-manifest.ps1 -TarPath /path/to/tar
```

完成后，您应会看到来自`wsl.exe --list --online`的以下输出：

PowerShell

```
$ wsl --list --online
The following is a list of valid distributions that can be installed.
Install using 'wsl.exe --install <Distro>'.

NAME              FRIENDLY NAME
test-distro-v1    Test distribution version 1
```

然后，可以运行 `wsl.exe --install test-distro-v1` 以尝试安装新分发版。

完成后，可以删除 `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Lxss\DistributionListUrl` 以还原到官方清单。

WSL 自定义 Linux 分发建议

配置文件建议

- 确保自定义分发包含 `/etc/wsl.conf` 配置文件和 `/etc/wsl-distribution.conf` 配置文件。这两个文件都应归 `root:root` 其所有，其权限应为 `0644`。
- 用于创建新用户的`oobe.command`设置时， `uid`和`oobe.defaultUid`应同时设置为`1000`。
- 请确保在分发配置文件中同时设置`oobe.defaultName`和`shortcut.icon`： `/etc/wsl-distribution.conf`
- 不要在根文件系统中包含该文件 `/etc/resolv.conf` 。
- 在 `/etc/passwd` 中包含根用户。 此 `uid` 根用户应为 `0`。
- 在 `/etc/shadow` 中不应有密码哈希。
- 存档不应包含内核或 initramfs。

系统建议

如果启用了 systemd，则应禁用或屏蔽可能导致 WSL 问题的单元。 已知以下单位会导致 WSL 分发中的问题（适用于系统和用户单位）：

- systemd-resolved.service
- systemd-networkd.service

- NetworkManager.service
- systemd-tmpfiles-setup.service
- systemd-tmpfiles-clean.service
- systemd-tmpfiles-clean.timer
- systemd-tmpfiles-setup-dev-early.service
- systemd-tmpfiles-setup-dev.service
- tmp.mount

Last updated on 2025/10/04

在 WSL 2 中装载 Linux 磁盘

如果要访问 Windows 不支持的 Linux 磁盘格式，可以使用 WSL 2 装载磁盘并访问其内容。 本教程将介绍标识要附加到 WSL2 的磁盘和分区的步骤、如何装载它们以及如何对其进行访问。

如果你正在连接外部硬盘，且按照这些装载说明操作不成功，则可能需要参考连接 USB 设备的说明。 `wsl --mount` 命令目前不支持 USB/闪存驱动器/SD 读卡器（详细了解此问题 ）。

> **注意**
>
> 将磁盘附加到 WSL 2 需要管理员访问权限。 WSL 2 `wsl --mount` 命令不支持装载当前正在使用的磁盘（或属于该磁盘的分区）。 即使只请求一个分区，`wsl --mount` 也始终会附加整个磁盘。 无法装载 Windows 安装磁盘。

先决条件

需要位于 Windows 11 上，或运行 WSL Microsoft 应用商店版本。 若要检查 WSL 和 Windows 版本，请使用以下命令：`wsl.exe --version`

使用 Windows 格式设置与 Linux 格式设置装载外部驱动器之间的差异

为 Windows 设置格式的外部驱动器通常使用 NTFS 文件系统格式。 为 Linux 设置格式的外部驱动器通常使用 Ext4 文件系统格式。

如果在 Windows 文件系统上装载了 NTFS 格式的驱动器，则可以使用 WSL 从 Linux 分发版访问该驱动器，方法是创建一个装载的目录（`sudo mkdir /mnt/d`替换为 `d` 要使用的驱动器号），然后使用 `drvfs` 文件系统互作插件和命令：

```Bash
sudo mount -t drvfs D: /mnt/d
```

详细了解装载场景 。

如果你有 Ext4 格式的驱动器，则无法将其装载到 Windows 文件系统上。 若要使用 WSL 在 Linux 分发版中装载 Ext4 格式的驱动器，可以按照下面的说明使用 `wsl --mount` 命令。

装载未分区的磁盘

如果磁盘没有任何分区，则可以使用 `wsl --mount` 命令直接装载该磁盘。 首先需要标识磁盘。

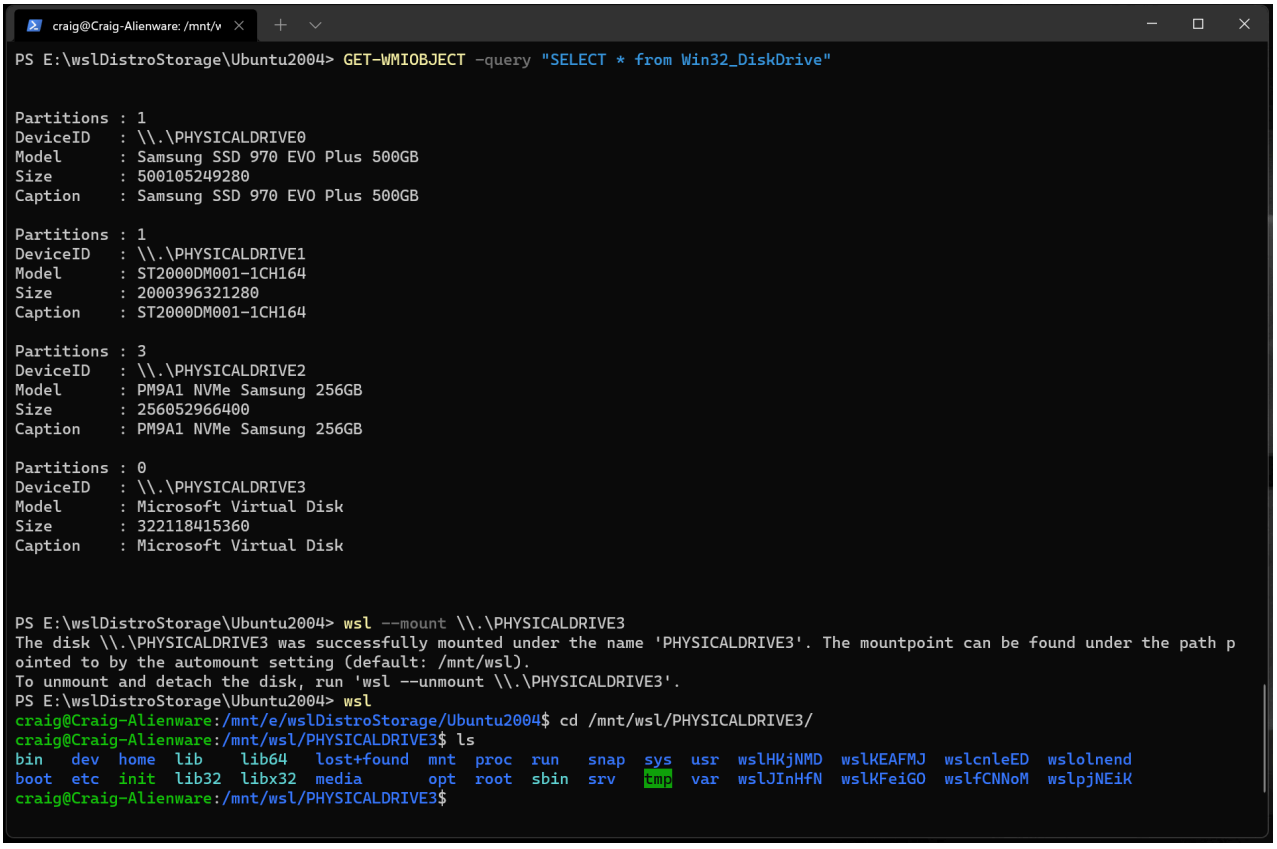
1. **标识磁盘** - 要列出 Windows 中的可用磁盘，请运行：

   ```PowerShell
   Get-CimInstance -Query "SELECT * from Win32_DiskDrive"
   ```

   磁盘路径在“**DeviceID**”列下可用。 通常采用 `\\.\PHYSICALDRIVE*` 格式。

2. **装载磁盘** - 使用 PowerShell，可以使用上面发现的磁盘路径装载磁盘，请运行：

   ```PowerShell
   wsl.exe --mount <Disk>
   ```



装载分区磁盘

如果不确定磁盘的文件格式或其中的分区，可以按照以下步骤进行装载。

1. **标识磁盘** - 要列出 Windows 中的可用磁盘，请运行：

   ```PowerShell
   Get-CimInstance -Query "SELECT * from Win32_DiskDrive"
   ```

磁盘路径在“DeviceID”列下可用。 通常采用 `\\.\PHYSICALDRIVE*` 格式。

2. **列出并选择要在 WSL 2 中装载的分区** - 确定磁盘后，运行：

```
PowerShell
wsl.exe --mount <Disk> --bare
```

这将使磁盘在 WSL 2 中可用。 （在我们的示例中，`<Disk>` 为 `\\.\PHYSICALDRIVE*`。

3. 附加后，可以通过在 WSL 2 中运行以下命令来列出分区：

```
Bash
lsblk
```

这会显示可用的块设备及其分区。

在 Linux 中，块设备被标识为 `/dev/<Device><Partition>`。 例如，`/dev/sdb3`是磁盘`sdb`的分区号`3`。

示例输出：

```
Bash
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sdb      8:16   0    1G  0 disk
├─sdb2   8:18   0   50M  0 part
├─sdb3   8:19   0  873M  0 part
└─sdb1   8:17   0  100M  0 part
sdc      8:32   0  256G  0 disk /
sda      8:0    0  256G  0 disk
```

标识文件系统类型

如果不知道磁盘或分区的文件系统类型，可以使用以下命令：

```
Bash
blkid <BlockDevice>
```

这将输出检测到的文件系统类型（采用 `TYPE="<Filesystem>"` 格式）。

装载所选分区

确定要装载的分区后，请在每个分区上运行以下命令：

```
PowerShell
wsl.exe --mount <Disk> --partition <Index> --type <Type>
```

> ⓘ **注意**
>
> 如果希望将整个磁盘装载为单个卷（即如果磁盘未分区），则可以省略 `--partition`。
>
> 如果 `--type` 省略，则默认文件系统类型为“ext4”。

访问磁盘内容

装载后，可以通过配置值指向的路径访问磁盘：`automount.root`。 默认值为 `/mnt/wsl`。

在 Windows 中，可以通过导航到以下位置从文件资源管理器访问磁盘：`\\wsl$\<Distro>\<Mountpoint>`（选择任何 Linux 发行版）。

卸载磁盘

如果要从 WSL 2 卸载和分离磁盘，请运行：

```
PowerShell
wsl.exe --unmount [DiskPath]
```

在 WSL 中装载 VHD

> ⓘ **注意**
>
> Microsoft Store 中的 WSL 引入了直接装载 VHD 的新参数：`wsl --mount --vhd <pathToVHD>`

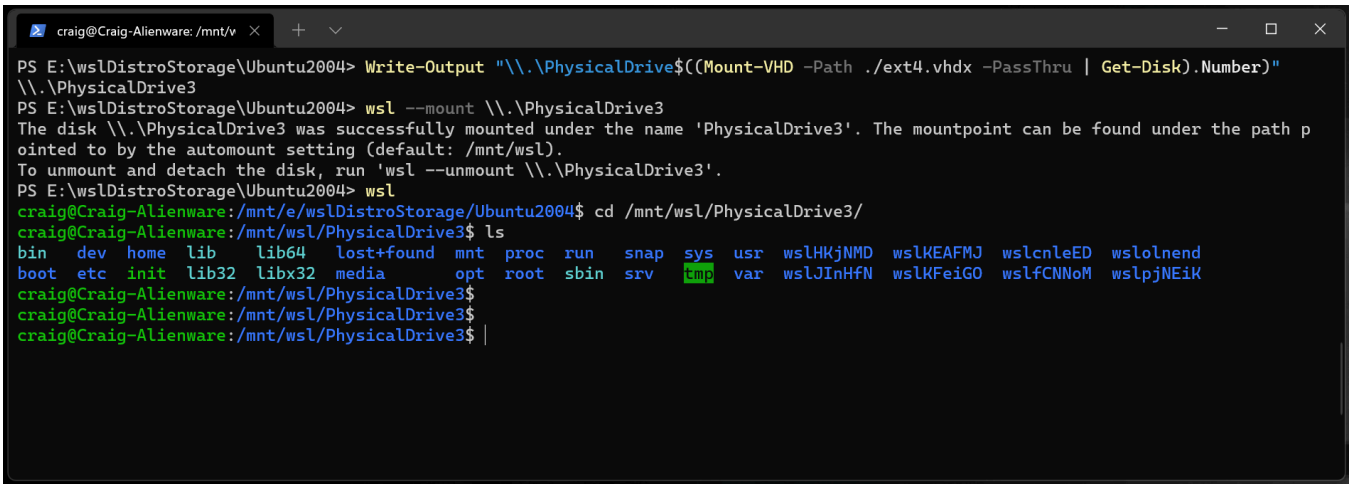
还可以使用 `wsl --mount` 将虚拟硬盘文件 (VHD) 装载到 WSL。 为此，首先需要使用 Windows 中的 Mount-VHD 命令将 VHD 装载到 Windows 中。 请确保以管理员权限运行此命令。 下面是一个示例，我们使用此命令并输出磁盘路径。 请务必将 `<pathToVHD>` 替换为实际 VHD 路径。

```
PowerShell
```

```
Write-Output "\\.\PhysicalDrive$((Mount-VHD -Path <pathToVHD> -PassThru | Get-Disk).Number)"
```

可以使用上面的输出获取此 VHD 的磁盘路径，然后按照上一部分中的说明将其装载到 WSL 中。

还可以使用此技术装载其他 WSL 发行版的虚拟硬盘并与之交互，因为每个 WSL 2 发行版都通过名为 `ext4.vhdx` 的虚拟硬盘文件进行存储。 默认情况下，WSL 2 发行版的 VHD 存储在以下路径中：`%LocalAppData%\Packages\[distro]\LocalState\[distroPackageName]`，请谨慎访问这些系统文件，这是一个高级用户工作流。 确保在与该磁盘交互之前运行 `wsl --shutdown` 以确保该磁盘未被使用。



命令行参考

装载特定文件系统

默认情况下，WSL 2 将尝试将设备装载为 ext4。 若要指定其他文件系统，请运行：

PowerShell

```
wsl.exe --mount <Disk> --type <Type>
```

例如，若要将磁盘装载为 FAT，请运行：

PowerShell

```
wsl.exe --mount <Disk> --type vfat
```

> ⓘ **注意**
>
> 若要列出 WSL2 中的可用文件系统，请运行：

Bash

```
cat /proc/filesystems
```

如果磁盘已通过 WSL2（Linux 文件系统）进行装载，则无法再通过 Windows 文件系统上的 ext4 驱动程序进行装载。

装载特定的分区

默认情况下，WSL 2 将尝试装载整个磁盘。 若要装载特定分区，请运行：

PowerShell

```
wsl.exe --mount <Disk> --partition <Index>
```

仅当磁盘是 MBR（主启动记录）或 GPT（GUID 分区表）时，此操作才有效。 了解分区样式 - MBR 和 GPT。

指定装载选项

若要指定装载选项，请运行：

PowerShell

```
wsl.exe --mount <Disk> --options <Options>
```

示例：

PowerShell

```
wsl.exe --mount <Disk> --options "data=ordered"
```

注意

目前仅支持特定于文件系统的选项。 泛型选项，例如 `ro`， `rw`，... `noatime` 不支持。

附加磁盘而不装载它

如果上述任何选项都不支持磁盘方案，则可以通过运行以下内容将磁盘附加到 WSL 2 而不对其进行装载：

PowerShell

```
wsl.exe --mount <Disk> --bare
```

这将使块设备在 WSL 2 内可用，以便可以从那里手动装载。 使用 `lsblk` 列出 WSL 2 中可用的块设备。

指定装载名称

> ⓘ **注意**
>
> 此选项仅适用于 Microsoft Store 中的 WSL

默认情况下，装入点名称是根据物理磁盘或 VHD 名称生成的。 这可以用 `--name` 覆盖此项。

PowerShell

```
wsl.exe --mount <Disk> --name <Name>
```

示例：

PowerShell

```
wsl.exe --mount <Disk> --name myDisk
```

分离磁盘

若要从 WSL 2 分离磁盘，请运行：

PowerShell

```
wsl.exe --unmount [DiskPath]
```

如果省略 `Diskpath`，则将卸载和分离所有附加的磁盘。

> ⓘ **注意**
>
> 如果无法卸载某个磁盘，可以通过运行 `wsl --shutdown` 强制退出 WSL 2，这将分离磁盘。

限制

- 目前，只能将整个磁盘附加到 WSL 2，这意味着不可能只附加一个分区。 具体而言，这意味着无法使用 `wsl --mount` 读取启动设备上的分区，因为该设备无法与 Windows 分离。
- 只有内核原生支持的文件系统可以被`wsl --mount`装载。 这意味着无法通过调用 `wsl --mount` 来使用已安装的文件系统驱动程序（例如 ntfs-3g）。
- 内核不直接支持的文件系统可以通过 `--bare` 附加然后调用相关的 FUSE 驱动程序来装载。

Last updated on 2025/08/15

连接 USB 设备

本指南将演练使用 USB/IP 开源项目 usbipd-win 将 USB 设备连接到 WSL 2 上运行的 Linux 分发版所需的步骤。

在 Windows 计算机上配置 USB/IP 项目可以实现常见的开发者 USB 场景，例如刷写 Arduino 或访问智能卡读卡器。

先决条件

- 运行 Windows 11（内部版本 22000 或更高版本）。 *(Windows 10 支持是可能的，请参阅以下说明）。*
- 需要具有 x64 或 ARM64 处理器的计算机。 （x86 目前不支持 usbipd-win）。
- WSL 已安装并设置为最新版本。
- 已安装并 设置为 WSL 2 的 Linux 分发版。

> **注意**
>
> 若要检查 Windows 版本和内部版本号，请选择 **Windows 徽标键** + **R**，键入 **winver**，选择 “ **确定**”。 您可以通过选择 **开始**>**设置**>**Windows 更新**>**检查更新**来更新到最新的 Windows 版本。 若要检查 Linux 内核版本，请打开 Linux 分发版并输入以下命令： `uname -a` 若要手动更新到最新内核，请打开 PowerShell 并输入命令： `wsl --update`

> **重要**
>
> WSL 现在通过 Microsoft 应用商店支持 Windows 10 和 Windows 11，这意味着 Windows 10 用户现在可以访问最新的内核版本，而无需从源进行编译。 请参阅 **适用于 Windows 10 和 11 的 WSL，现已在微软应用商店中普遍可用** ，了解如何更新到受微软应用商店支持的 WSL 版本。 如果无法更新到支持应用商店的 WSL 版本并自动接收内核更新，请参阅 **USBIPD-WIN 项目存储库** ，了解如何通过生成启用了自己的 USBIP 的 WSL 2 内核将 USB 设备连接到 WSL 2 上运行的 Linux 分发版。

安装 USBIPD-WIN 项目

WSL不提供本机连接USB设备的支持，因此需要安装开源项目usbipd-win。

内核要求

若要将 USBIPD 与适用于 Linux 的 Windows 子系统配合使用（WSL），需要具有 5.10.60.1 或更高版本的 Linux 内核版本。 如果已安装的内核版本早于 5.10.60.1，则可以通过先关闭 WSL `wsl --shutdown` 的任何正在运行的实例来更新它，然后运行以下命令： `wsl --update`

在 WSL 上安装 USBIPD

1. 转到 usbipd-win 项目的最新发布页面。
2. 选择 .msi 文件，该文件将下载安装程序。 （你可能会收到一条警告，要求你确认你信任此下载）。
3. 运行下载 usbipd-win_x.msi 安装程序文件。

> **注意**
>
> 或者，也可以使用 **Windows 程序包管理器** （winget）安装 usbipd-win 项目。 如果已安装 winget，只需使用以下命令： `winget install --interactive --exact dorssel.usbipd-win` 安装 usbipd-win。 如果省略 --interactive，则当需要安装驱动程序时，winget 可能会立即重启计算机。

这将安装：

- 名为 `usbipd` 的服务，（显示名称：USBIP 设备主机）。 可以使用 Windows 中的服务应用检查此服务的状态。
- 命令行工具 `usbipd`。 此工具的位置将添加到 PATH 环境变量。
- 名为 `usbipd` 的防火墙规则，用于允许所有本地子网连接到服务。 可修改此防火墙规则以微调访问控制。

连接 USB 设备

在附加 USB 设备之前，请确保 WSL 命令行处于打开状态。 这会使 WSL 2 轻型 VM 保持运行。

> **注意**
>
> 此文档假定已安装 **usbipd-win 5.0.0 或更高版本**

1. 通过以 *管理员* 模式打开 PowerShell 并输入以下命令列出连接到 Windows 的所有 USB 设备。 列出设备后，选择并复制要附加到 WSL 的设备总线 ID。

   PowerShell

   ```
   usbipd list
   ```

2. 在附加 USB 设备之前，必须使用该命令 `usbipd bind` 来共享设备，从而允许它附加到 WSL。 这需要管理员权限。 选择要在 WSL 中使用的设备的总线 ID，然后运行以下命令。 运行命令后，请再次使用命令 `usbipd list` 验证设备是否共享。

   PowerShell
   ```
   usbipd bind --busid 4-4
   ```

3. 若要附加 USB 设备，请运行以下命令。 （不再需要使用提升的管理员提示。确保 WSL 命令提示符处于打开状态，以使 WSL 2 轻型 VM 保持活动状态。 请注意，只要 USB 设备连接到 WSL，Windows 将无法使用它。 一旦连接到 WSL，任何在 WSL 2 上运行的发行版都可以使用该 USB 设备。 请确认设备是否已连接 `usbipd list`。 在 WSL 提示符下，运行 `lsusb` 以验证 USB 设备是否已列出，并且可以使用 Linux 工具与之交互。

   PowerShell
   ```
   usbipd attach --wsl --busid <busid>
   ```

4. 打开 Ubuntu（或首选 WSL 命令行），并使用以下命令列出附加的 USB 设备：

   Bash
   ```
   lsusb
   ```

   应会看到刚刚附加的设备，并且能够使用普通 Linux 工具与之交互。 根据应用程序，可能需要配置 udev 规则，以允许非根用户访问设备。

5. 在 WSL 中使用设备后，可以物理断开 USB 设备的连接，或者从 PowerShell 运行以下命令：

   PowerShell
   ```
   usbipd detach --busid <busid>
   ```

若要详细了解此作的工作原理，请参阅 GitHub 上的 Windows 命令行博客 和 usbipd-win 存储库。

有关视频演示，请参阅 WSL 2：连接 USB 设备（选项卡与空格显示）。

Last updated on 2025/06/10

调整区分大小写

区分大小写决定了在文件名或目录中，大写字母（FOO.txt）和小写字母（foo.txt）是作为区分（区分大小写）处理的，还是作为等效（不区分大小写）处理的。

- 区分大小写：FOO.txt ≠ foo.txt ≠ Foo.txt
- 不区分大小写：FOO.txt = foo.txt = Foo.txt

Windows 和 Linux 的大小写敏感性差异

使用 Linux 和 Windows 文件和目录时，可能需要调整大小写敏感性的处理方式。

标准行为：

- Windows 文件系统将文件和目录名称视为不区分大小写。 FOO.txt 和 foo.txt 将被视为等效文件。
- Linux 文件系统将文件和目录名称视为区分大小写。 FOO.txt 和 foo.txt 将被视为不同的文件。

Windows 文件系统支持使用每个目录的属性标志设置区分大小写。 虽然标准操作是不区分大小写的，但你可以分配一个属性标志，使目录设置为区分大小写，这样它将识别仅大小写不同的 Linux 文件和文件夹。

将驱动器装载到适用于 Linux 的 Windows 子系统（WSL）文件系统时，这尤其如此。 在 WSL 文件系统中工作时，由于运行的是 Linux，文件和目录默认情况下被视为区分大小写。

> ⓘ **注意**
>
> 过去，如果你的文件名称只因大小写而不同，则 Windows 无法访问这些文件，因为 Windows 应用程序将文件系统视为不区分大小写，并且无法区分名称仅在大小写中不同的文件。 虽然 Windows 文件资源管理器将显示这两个文件，但无论选择哪个文件，只有一个文件将打开。

更改文件和目录的大小写敏感性

以下步骤说明如何更改 Windows 文件系统上的目录，使其具备区分大小写的功能，并识别仅因大小写不同的文件和文件夹。

> ⚠ **警告**

某些 Windows 应用程序基于文件系统不区分大小写的假设，未使用正确的大小写来引用文件。 例如，应用程序转换文件名以使用所有大写或小写并不罕见。 在标记为区分大小写的目录中，这意味着这些应用程序无法再访问这些文件。 此外，如果 Windows 应用程序在使用区分大小写文件的目录树中创建新目录，这些目录将不再区分大小写。 这会使在区分大小写的目录中使用 Windows 工具变得困难，因此在更改 Windows 文件系统区分大小写设置时要小心。

检查当前区分大小写

若要检查 Windows 文件系统中目录是否区分大小写，请运行以下命令：

PowerShell

```
fsutil.exe file queryCaseSensitiveInfo <path>
```

将 `<path>` 替换为您的文件路径。 对于 Windows （NTFS） 文件系统中的目录， `<path>` 如下所示： `C:\Users\user1\case-test` 或者如果已在 `user1` 目录中，可以运行： `fsutil.exe file setCaseSensitiveInfo case-test`

修改区分大小写

Windows 10 内部版本 17107 开始支持每个目录区分大小写。 在 Windows 10 内部版本 17692 中，支持已更新为包括检查和修改 WSL 中目录的区分大小写标志。 使用名为 `system.wsl_case_sensitive` 的扩展属性来公开大小写敏感性。 对于不区分大小写的目录，此属性的值将为 0，对于区分大小写的目录为 1。

更改目录的区分大小写需要运行 **更高权限** （以管理员身份运行）。 更改区分大小写标志还需要对目录具有“写入属性”、“创建文件”、“创建文件夹”和“删除子文件夹和文件”权限。 有关此问题的详细信息，请参阅故障排除部分。

若要更改 Windows 文件系统中的目录，使其区分大小写（FOO ≠ foo），请以管理员身份运行 PowerShell 并使用以下命令：

PowerShell

```
fsutil.exe file setCaseSensitiveInfo <path> enable
```

若要将 Windows 文件系统中的目录更改回不区分大小写的默认（FOO = foo），请以管理员身份运行 PowerShell 并使用以下命令：

PowerShell

```
fsutil.exe file setCaseSensitiveInfo <path> disable
```

目录必须为空，才能更改该目录上的区分大小写标志属性。 不能在包含名称仅大小写不同的文件夹或文件的目录中禁用大小写敏感性标志。

大小写敏感性继承

创建新目录时，这些目录将继承父目录的大小写敏感性。

> ⚠ **警告**
>
> 在 WSL 1 模式下运行时，此继承策略例外。 当分发在 WSL 1 模式下运行时，不会继承每个目录区分大小写的标志;在区分大小写的目录中创建的目录本身不会自动区分大小写。 必须将每个目录显式标记为区分大小写

用于在 WSL 配置文件中装载驱动器的区分大小写选项

在使用 WSL 配置文件在适用于 Linux 的 Windows 子系统上装载驱动器时，可以管理大小写敏感性。 已安装的每个 Linux 分发版都可以有它自己的 WSL 配置文件（称为 `/etc/wsl.conf`）。 有关如何装载驱动器的详细信息，请参阅 开始在 WSL 2 中装载 Linux 磁盘。

若要在装载驱动器时在 `wsl.conf` 文件中配置区分大小写选项，

1. 打开将使用的 Linux 分发版（即Ubuntu）。
2. 更改目录直到看到 `etc` 文件夹（这可能需要你从 `cd ..` 目录向上 `home`）。
3. 列出 `etc` 目录中的文件，以查看 `wsl.conf` 文件是否已存在（使用 `ls` 命令或使用 Windows 文件资源管理器通过 `explorer.exe .` 查看目录）。
4. `wsl.conf`如果文件尚不存在，则可以使用： `sudo touch wsl.conf`或通过运行`sudo nano /etc/wsl.conf`来创建该文件，这将在从 Nano 编辑器保存时创建该文件。
5. 可使用以下选项添加到 `wsl.conf` 文件中：

默认设置： `off` 对于区分大小写不可用（装载的 NTFS 驱动器上的所有目录都将不区分大小写）。

Bash

```
[automount]
options = case = off
```

为每个目录启用区分大小写： `dir`

```
Bash
[automount]
options = case = dir
```

将(NTFS)磁盘上的所有目录视为区分大小写： `force`

```
Bash
[automount]
options = case = force
```

此选项仅支持在作为 WSL 1 运行的 Linux 分发版上装载驱动器，并且可能需要注册密钥。 若要添加注册密钥，可以从提升的（管理员）命令提示符处使用此命令： `reg.exe add HKLM\SYSTEM\CurrentControlSet\Services\lxss /v DrvFsAllowForceCaseSensitivity /t REG_DWORD /d 1`

在对文件进行任何更改 `wsl.conf` 后，需要重启 WSL 才能使这些更改生效。 可以使用以下命令重启 WSL： `wsl --shutdown`

> **提示**
>
> 若要装载驱动器（这些驱动器使用 DrvFs 文件系统插件，可以在 /mnt 下访问，例如 /mnt/c、/mnt/d 等），并为所有驱动器设置特定的区分大小写，请按上述说明使用 `/etc/wsl.conf`。 若要为一个特定驱动器设置默认装载选项，请使用 **/etc/fstab 该文件** 指定这些选项。 有关更多 WSL 配置选项，请参阅 **使用 wslconf 配置每个发行版启动设置**。

更改装载到 WSL 分发版的驱动器上的区分大小写

默认情况下，装载到 WSL 分发版的 NTFS 格式驱动器将不区分大小写。 要更改装载到 WSL 分发版（例如 Ubuntu）的驱动器上的目录如何处理大小写，请按照与 Windows 文件系统中所列相同的步骤操作。 （默认情况下，EXT4 驱动器区分大小写）。

若要在目录（FOO ≠ foo）上启用区分大小写，请使用以下命令：

```
Bash
fsutil.exe file setCaseSensitiveInfo <path> enable
```

若要禁用目录上的区分大小写并返回到不区分大小写的默认（FOO = foo），请使用以下命令：

```
Bash
fsutil.exe file setCaseSensitiveInfo <path> disable
```

> **注意**
>
> 在 WSL 运行时，如果要更改已装载驱动器目录的区分大小写标志，请确保 WSL 没有链接到该目录，否则更改将不会生效。 这意味着目录不能由任何 WSL 进程打开，包括使用目录（或其后代）作为当前工作目录。

使用 Git 配置区分大小写

Git 版本控制系统还有一个配置设置，可用于调整正在使用的文件的区分大小写。 如果使用 Git，可能需要调整 git config core.ignorecase 设置。

若要将 Git 设置为区分大小写（FOO.txt ≠ foo.txt），请输入：

```
git config core.ignorecase false
```

若要将 Git 设置为不区分大小写（FOO.txt = foo.txt），请输入：

```
git config core.ignorecase true
```

在不区分大小写的文件系统上将此选项设置为 false 可能会导致混淆错误、错误冲突或重复文件。

有关详细信息，请参阅 Git 配置文档。

故障排除

我的目录包含混合大小写的文件，需要区分大小写，但 Windows FS 工具无法识别这些文件

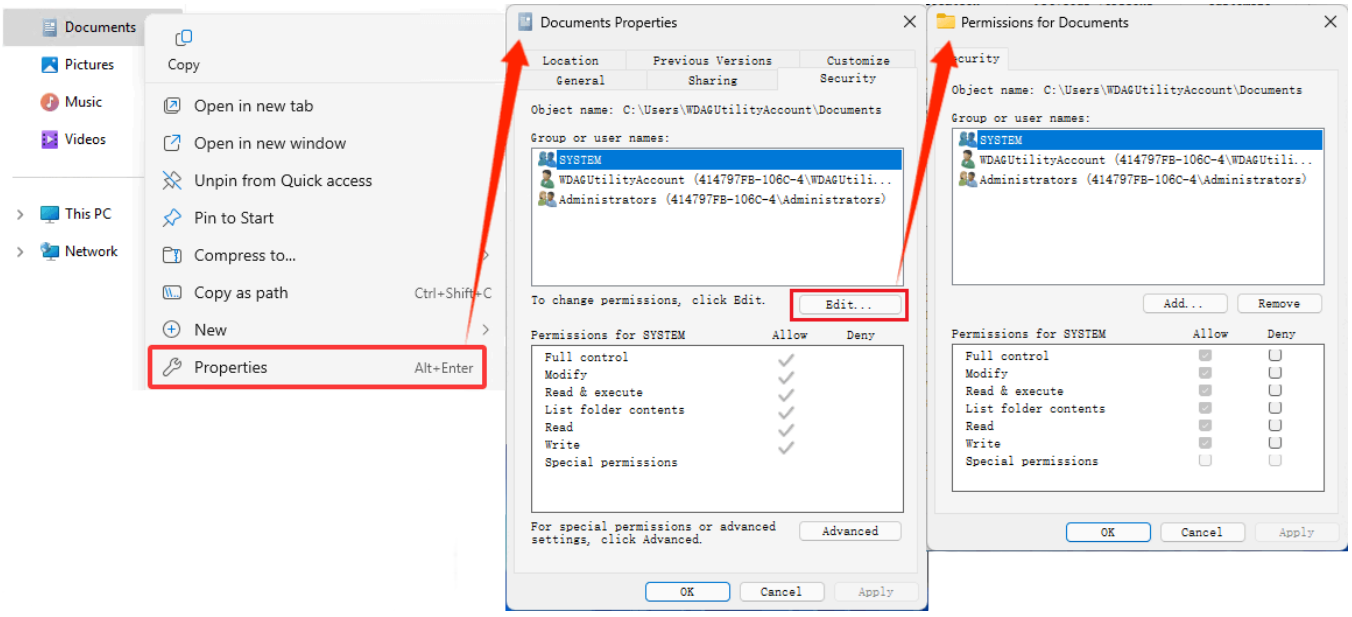
若要使用 Windows 文件系统工具处理包含混合大小写文件的 Linux 目录，您需要创建一个全新的目录并将其设置为区分大小写，然后将文件复制到该目录中（可以使用 git 克隆或解压缩）。 这些文件将保持混合大小写。 （请注意，如果已尝试将文件移动到不区分大小写的目录，并且存在冲突，则可能存在一些被覆盖且不再可用的文件。

错误：目录不为空

不能更改包含其他文件或目录的目录上的区分大小写设置。 尝试创建新目录，更改设置，然后将混合大小写文件复制到其中。

错误：访问被拒绝

确保对更改区分大小写所需的目录具有“写入属性”、“创建文件”、“创建文件夹”和“删除子文件夹和文件”权限。 若要检查这些设置，请在 Windows 文件资源管理器中打开目录（从命令行使用以下命令）： `explorer.exe .` 右键单击目录，然后选择“ **属性** ”以打开“文档属性”窗口，然后选择“ **编辑** ”以查看或更改目录的权限。



错误：此操作需要本地 NTFS 卷

区分大小写属性只能在 NTFS 格式的文件系统中的目录上设置。 默认情况下，WSL（Linux）文件系统中的目录对大小写敏感（无法通过 fsutil.exe 工具设置为不区分大小写）。

其他资源

- 开发博客: 按目录区分大小写敏感度与 WSL
- DevBlog：在 WSL 中改进了每个目录的区分大小写支持

Last updated on 2025/06/10

如何管理 WSL 磁盘空间

本指南介绍如何管理使用 WSL 2 安装的 Linux 分发版使用的磁盘空间，包括：

- 如何检查 VHD 中可用的磁盘空间量
- 如何扩展 VHD 的大小
- 如果发生错误，如何修复 VHD
- 如何查找任何已安装 Linux 分发版的 .vhdx 文件和磁盘路径

适用于 Linux 的 Windows 子系统（WSL 2）使用虚拟化平台与主机 Windows作系统一起安装 Linux 分发版，创建虚拟硬盘（VHD）来存储所安装的每个 Linux 分发版的文件。 这些 VHD 使用 ext4 文件系统类型 ，在 Windows 硬盘上表示为 *ext4.vhdx* 文件。

WSL 2 会自动调整这些 VHD 文件的大小以满足存储需求。 默认情况下，WSL 2 使用的每个 VHD 文件最初分配的最大磁盘空间量为 1TB（在 WSL 版本 0.58.0 之前，此默认值设置为 512GB 最大值和 256GB 最大值之前）。

如果 Linux 文件所需的存储空间超过此最大大小，则会看到指出磁盘空间不足的错误。 若要修复此错误，请按照以下有关如何 扩展 WSL 2 虚拟硬盘大小的指南进行作。

如何检查可用磁盘空间

使用 Linux `Df` 命令检查随 WSL 2 一起安装的 Linux 分发版的 VHD 中可用的磁盘空间量。

若要检查可用的磁盘空间，请打开 PowerShell 命令行并输入此命令（替换为 `<distribution-name>` 实际的分发名称）：

PowerShell

```
wsl.exe --system -d <distribution-name> df -h /mnt/wslg/distro
```

如果此命令不起作用，请使用 `wsl --update` 命令升级到 WSL 的 Store 版本，或尝试 `wsl df -h /`。

输出将包括：

- **文件系统**：VHD 文件系统的标识符
- **大小**：磁盘的总大小（分配给 VHD 的最大空间量）
- **已用**：VHD 中当前使用的空间量
- **可用性**：VHD 中剩余的空间量（已分配大小减去使用的量）
- **使用%**：剩余磁盘空间百分比（已用/已分配大小）
- **装载在**：装载磁盘的目录路径

如果看到即将达到分配给 VHD 的可用磁盘空间量，或者由于没有剩余磁盘空间而收到错误，请参阅下一部分，了解如何扩展分配给与 Linux 分发关联的 VHD 的最大磁盘空间量的步骤。 WSL 分配给 VHD 的磁盘空间量将始终显示默认的最大量（最新版本的 WSL 中为 1TB），即使实际 Windows 设备上的磁盘空间量小于此量。 WSL 会挂载一个 VHD，该 VHD 会随着使用逐渐扩展，因此 Linux 发行版会发现它可以增长到分配的最大大小为 1TB。

如何扩展 WSL 2 虚拟硬盘的大小

若要将 Linux 分发版的 VHD 大小扩展到默认 1TB 最大分配磁盘空间量之外，可以使用 `wsl --manage` 该命令，或按照下面的手动步骤作。 （早期 WSL 版本最大默认值可能设置为 512GB 或 256GB）。

使用 `wsl --manage` 扩展 VHD 大小

该 `wsl --manage` 命令仅适用于 WSL 版本 2.5 及更高版本。

若要重设虚拟磁盘上分配的空间大小，请执行以下作：

1. 使用命令终止所有 WSL 实例 `wsl.exe --shutdown`
2. 运行 `wsl --manage <distribution name> --resize <memory string>`。 支持的内存字符串采用形式 `<Memory Value>B/M/MB/G/GB/T/TB`。 小数值当前不受支持（例如 `2.5TB`）。

输出应如下所示：

```
Bash
e2fsck 1.46.5 (30-Dec-2021)
Pass 1: Checking inodes, blocks, and sizes
resize2fs 1.46.5 (30-Dec-2021)
The operation completed successfully.
```

此 Linux 分发版的虚拟驱动器（ext4.vhdx）现已成功扩展到新的大小。

手动扩展

若要使用手动步骤扩展 Linux 分发版的 VHD 大小，请执行以下作：

1. 使用以下命令终止所有 WSL 实例： `wsl.exe --shutdown`
2. 将目录路径复制到与计算机上安装的 Linux 分发版关联的 *ext4.vhdx* 文件。 有关帮助，请参阅 如何查找 Linux 分发版的 vhdx 文件和磁盘路径。

3. 使用管理员权限打开 Windows 命令提示符，然后输入以下命令打开 diskpart 命令解释器：

   Windows 命令提示符
   ```
   diskpart
   ```

4. 你现在会有一个 `DISKPART>` 提示。 输入以下命令，将 `<pathToVHD>` 替换为步骤 2 中复制的与 Linux 分发版关联文件的目录路径 `ext4.vhdx`。

   Windows 命令提示符
   ```
   Select vdisk file="<pathToVHD>"
   ```

5. 显示与此虚拟磁盘关联的详细信息，包括 **虚拟大小**，表示分配 VHD 的当前最大大小：

   Windows 命令提示符
   ```
   detail vdisk
   ```

6. 需要将 **虚拟大小** 转换为兆字节。 例如，如果 **虚拟大小为 512 GB**，则等于 **512000 MB**。 输入的新值必须大于此原始值。 若要将虚拟大小加倍为 512 GB 到 1024 GB，需以 MB 为单位输入值： **1024000**。 请注意不要输入高于实际想要的值，因为减少虚拟磁盘大小的过程要复杂得多。

7. 使用 Windows 命令提示符提示 `DISKPART>` 输入要分配给此 Linux 分发的新最大大小的值：

   Windows 命令提示符
   ```
   expand vdisk maximum=<sizeInMegaBytes>
   ```

8. 退出`DISKPART>`的提示：

   Windows 命令提示符
   ```
   exit
   ```

9. 启动此 Linux 分发版。 *（确保它在 WSL 2 中运行。可以使用以下命令确认此作：* `wsl.exe -l -v` *不支持 WSL 1。*

10. 让 WSL 知道它可以通过从 WSL 分发命令行运行这些命令来扩展此分发的文件系统大小。 你可能会看到此消息，以响应第一个 **装载** 命令：“/dev：尚未装载在 /dev 上。”可以安全地忽略此消息。

```
Bash
sudo mount -t devtmpfs none /dev
mount | grep ext4
```

11. 复制此项的名称，如下所示： `/dev/sdX` （X 表示任何其他字符）。 在以下示例中， **X** 的值为 **b**：

    ```
    Bash
    sudo resize2fs /dev/sdb <sizeInMegabytes>M
    ```

    使用上面的示例，我们将 vhd 大小更改为 **2048000**，因此命令将为： `sudo resize2fs /dev/sdb 2048000M`

    > ⓘ **注意**
    >
    > 可能需要安装 **resize2fs**。 如果是这样，可以使用此命令来安装它： `sudo apt install resize2fs`.

    输出将类似于以下内容：

    ```
    Bash
    resize2fs 1.44.1 (24-Mar-2021)
    Filesystem at /dev/sdb is mounted on /; on-line resizing required
    old_desc_blocks = 32, new_desc_blocks = 38
    The filesystem on /dev/sdb is now 78643200 (4k) blocks long.
    ```

此 Linux 分发版的虚拟驱动器（ext4.vhdx）现已成功扩展到新的大小。

> ⓘ **重要**
>
> 建议不要使用 Windows 工具或编辑器修改、移动或访问文件夹中的 `AppData` WSL 相关文件。 这样做可能会导致 Linux 分发版损坏。 如果要从 Windows 访问 Linux 文件，可以通过路径 `\\wsl$\<distribution-name>\` 访问它。 打开 WSL 分发版并输入 `explorer.exe .` 以查看该文件夹。 若要了解详细信息，请参阅博客文章： **从 Windows 访问 Linux 文件** 。

如何修复 VHD 装载错误

如果遇到与“装载分发磁盘”相关的错误，这可能是由于突然关闭或停电，并可能导致 Linux 分发 VHD 切换到 *只读* ，以避免数据丢失。 您可以按照以下步骤使用 Linux 命令修复和还原 `e2fsck`

发行版。

使用 lsblk 命令标识块设备名称

当 WSL 2 安装 Linux 发行版时，它会将该发行版与其自身的文件系统一起挂载为虚拟硬盘（VHD）。 Linux 将这些硬盘驱动器称为“阻止设备”，你可以使用命令查看有关它们 `lsblk` 的信息。

若要查找 WSL 2 当前正在使用的块设备的名称，请打开分发版并输入命令： `lsblk` （或打开 PowerShell 并输入命令： `wsl.exe lsblk`.）输出如下所示：

```
Bash
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 363.1M  1 disk
sdb    8:16   0     8G  0 disk [SWAP]
sdc    8:32   0   1.5T  0 disk
sdd    8:48   0     1T  0 disk /mnt/wslg/distro
```

有关块设备的信息包括：

- **名称**：分配给设备的名称将为 sd[a-z]，其中每个使用中的 SCSI 磁盘都会被分配一个字母标识。 `sda` 始终是系统发行版。
- **MAJ：MIN**：表示 Linux 内核用于在内部识别设备的编号，首先的数字表示设备类型（例如，8 表示小型计算机系统接口/SCSI 磁盘）。
- **RM**：让我们知道设备是可移动的（1）还是不可移动（0）。
- **大小**：卷的总大小。
- **RO**：让我们知道设备是只读的（1）还是不是（0）。
- **类型**：指设备类型（在本例中为磁盘）。
- **MOUNTPOINTS**：指块设备所在的文件系统上的当前目录（SWAP 用于预配置的非活动内存，因此没有装入点）。

只读回退错误

如果在打开 Linux 发行版时 WSL 遇到“装载错误”，则可能会将发行版设置为只读模式以作为回退措施。 如果发生这种情况，则分发可能会在启动期间显示以下错误：

```
PowerShell
An error occurred mounting the distribution disk, it was mounted read-only as a
fallback.
```

当发行版以只读模式启动时，所有尝试写入文件系统的操作都会失败，并显示如下错误：

Bash

```
$ touch file
touch: cannot touch 'file': Read-only file system
```

若要修复 WSL 中的磁盘装载错误，并再次将其还原回可用/可写状态，可以使用 `wsl.exe --mount` 该命令通过以下步骤重新装载磁盘：

1. 通过以管理员身份打开 PowerShell（在提升的命令提示符中）并输入命令来关闭所有 WSL 分发版：

   PowerShell

   ```
   wsl.exe --shutdown
   ```

2. 输入挂载命令，并将 `<path-to-ext4.vhdx>` 替换为此分发版的 .vhdx 文件路径。 有关查找此文件的帮助，请参阅 如何查找 Linux 分发版的 VHD 文件和磁盘路径。

   PowerShell

   ```
   wsl.exe --mount <path-to-ext4.vhdx> --vhd --bare
   ```

3. 使用 PowerShell 的`wsl.exe lsblk`命令来识别分发的阻止设备名称（sd[a-z]），然后输入以下命令进行磁盘修复（将`<device>`替换为正确的块设备名称，如“sdc”）。 该 `e2fsck` 命令检查 ext4 文件系统（随 WSL 一起安装的分发版所使用的类型）是否存在错误，并相应地修复它们。

   PowerShell

   ```
   wsl.exe sudo e2fsck -f /dev/<device>
   ```

   > ⓘ **注意**
   >
   > 如果只安装了单个 Linux 分发版，可能会遇到“正在使用的 ext 文件”错误，并且需要 **安装** 其他分发版才能运行 `wsl.exe lsblk`。 修复完成后，可以 **卸载** 分发版。 此外，可能需要关闭 Windows 上的 Docker Desktop 以避免在运行命令 `Wsl/Service/CreateInstance/MountVhd/HCS/ERROR_SHARING_VIOLATION`时出错`wsl.exe sudo e2fsck -f /dev/sdc`。

4. 修复完成后，通过输入以下命令在 PowerShell 中卸载磁盘：

   PowerShell

```
wsl.exe --unmount
```

> **⚠ 警告**
>
> 可以使用以下命令： `sudo mount -o remount,rw /` 将只读分发返回到可用/可写状态，但所有更改都将在内存中，因此在重新启动分发时会丢失。 建议改用上面列出的步骤来装载和修复磁盘。

如何查找 Linux 分发版的 .vhdx 文件和磁盘路径

若要查找 Linux 分发版的 *.vhdx* 文件和目录路径，请打开 PowerShell 并使用以下脚本，替换为 `<distribution-name>` 实际的分发名称：

PowerShell

```
(Get-ChildItem -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Lxss | Where-Object { $_.GetValue("DistributionName") -eq '<distribution-name>' }).GetValue("BasePath") + "\ext4.vhdx"
```

结果将显示一个类似于 `%LOCALAPPDATA%\Packages\<PackageFamilyName>\LocalState\<disk>.vhdx` 的路径。 例如：

PowerShell

```
C:\Users\User\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\LocalState\ext4.vhdx
```

这是与您列出的 Linux 发行版关联的文件 `ext4.vhdx` 的路径。

Last updated on 2025/06/10

WSL 插件

Windows 应用程序现在可以使用 WSL 插件创建和与在适用于 Linux 的 Windows 子系统（WSL）内运行的 Linux 进程进行交互。 本文概述了工作原理以及如何开始使用它们。

了解插件功能

WSL 插件提供以下核心功能：

- 允许应用程序指定在 WSL 虚拟机启动时启动的 Windows 可执行文件
- Windows 可执行文件可以在 WSL 内部创建 Linux 进程，并且可以使用虚拟化套接字直接与它们通信

使用这些应用程序，Windows 应用程序可以基于 WSL 体验构建，并提供与适用于 Linux 的 Windows 子系统相关的附加功能。

安装插件

作为 WSL 插件创建者，可以通过将注册表项设置为指向插件的 DLL 文件，在计算机上安装插件。

作为 WSL 用户，使用的任何应用程序都会在其正常安装过程中自动安装 WSL 插件。

创建自己的插件

若要启动插件项目，需要生成 Win32 dll。 使用此方法进行设置的最简单方法是试用 WSL 插件示例项目。 为此，可以将 WSL 插件示例存储库 克隆到本地文件夹 `git clone` ，并在 Visual Studio 中将其打开。

打开项目时，请导航到 dllmain.cpp 该文件，你将看到可用于 WSL 插件的函数列表。

然后，可以按“生成”选项卡并生成项目，这将输出可供使用的 DLL，很可能在下方 `wsl-plugin-sample\x64\Debug\WSLPluginSample.dll`。

测试插件

仅当 WSL 插件经过 数字签名时，才会运行它们。 若要测试此项，需要在计算机上启用测试签名。

启用测试签名和创建测试认证

打开提升的 PowerShell 窗口，并通过运行以下命令 启用测试签名：

```PowerShell
## If this command results in "The value is protected by Secure Boot policy and cannot be modified or deleted"
## Then reboot the PC, go into BIOS settings, and disable Secure Boot. BitLocker may also affect your ability to modify this setting.
Bcdedit.exe -set TESTSIGNING ON
```

启用测试签名（可能需要重启）后，在上面创建的 WSLPluginSample.dll 文件的目录中的提升 PowerShell 命令提示符下，我们将创建 WSL 测试证书：

```PowerShell
# Create the cert
$certname = "WSLPluginTestCert"
$cert = New-SelfSignedCertificate -Subject "CN=$certname" -CertStoreLocation "Cert:\CurrentUser\My" -KeyExportPolicy Exportable -KeySpec Signature -KeyLength 2048 -KeyAlgorithm RSA -HashAlgorithm SHA256 -Type CodeSigningCert

# Export it to a local path
Export-Certificate -Cert $cert -FilePath ".\$certname.cer"

# Sign the DLL file
Set-AuthenticodeSignature -FilePath "C:\dev\Path\To\Your\WSLPlugin.dll" -Certificate $cert
```

最后一次将证书导入到受信任的根证书颁发机构：

```PowerShell
Import-Certificate -FilePath ".\$certname.cer" -CertStoreLocation Cert:\LocalMachine\Root"
```

有关详细信息，请参阅 如何创建自签名证书 文档页。

安装插件

在同一提升的 PowerShell 窗口中，运行以下命令以安装插件，并确保将插件 DLL 的路径更改为现有路径：

```PowerShell
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Lxss\Plugins" -Name "demo-plugin" -Value "C:\Path\to\plugin.dll" -Force
```

若要使用插件，请通过以下方法重启 wsl 服务：

```PowerShell
Stop-Service -Name "wslservice" -Force
wsl.exe echo "test"
```

现在应加载插件。 有关插件加载失败的详细信息，请参阅 故障排除和其他信息 部分。

完成后，可以运行以下命令来删除插件（请记住，需要重启 WSL 服务才能生效）：

```PowerShell
Remove-ItemProperty -Path
"HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Lxss\Plugins" -Name "demo-plugin" -
Force
```

故障排除和其他信息

常见错误代码：

- Wsl/Service/CreateInstance/CreateVm/Plugin/ERROR_MOD_NOT_FOUND -> 无法加载插件 DLL。 检查插件注册路径是否正确
- Wsl/Service/CreateInstance/CreateVm/Plugin/TRUST_E_NOSIGNATURE -> 插件 DLL 未签名，或者计算机不信任其签名
  - 请 启用测试签名 ，并查看 上面的签名部分，了解如何设置测试证书。
- Wsl/Service/CreateInstance/CreateVm/Plugin/* -> 插件 DLL 在 WSLPLUGINAPI_ENTRYPOINTV1 或 OnVmStarted 中返回了错误（）
- Wsl/Service/CreateInstance/Plugin/* -> 插件 DLL 在 OnDistributionStarted 中返回错误（）

插件 Linux 用户空间

通过 ExecuteBinary（） 创建的 Linux 进程将在 WSL2 虚拟机的根命名空间中运行。 此命名空间不与任何分发相关联，并且具有非常最小的基于 Mariner 的根文件系统。

文件系统是可写 tmpfs，这意味着在关闭 WSL2 虚拟机时，将删除对它所做的所有更改。

可以通过在 WSL 运行时运行 `wsl --debug-shell` 来检查根命名空间的内容。

其他注意事项

- 所有 WSL 插件挂钩都是同步的，这意味着 WSL 将等待插件挂钩完成，然后再继续。

- 插件返回的任何错误都被视为 WSL 致命错误（这意味着用户的分发不会启动）
- 插件代码与 WSL 服务在同一地址空间中运行。 插件中的任何崩溃都会使整个 WSL 服务崩溃，这可能会导致数据丢失

Last updated on 2025/08/14

企业环境：为公司设置适用于 Linux 的 Windows 子系统

本指南适用于负责设置企业工作环境的 IT 管理员或安全分析师，其目标是跨多台计算机分发软件，并跨这些工作计算机维护一致的安全设置级别。

许多公司使用 Microsoft Intune 和 Microsoft Defender 来管理这些安全设置。 但是，在这种情况下设置 WSL 和访问 Linux 发行版需要一些特定的设置。 本指南提供了在企业环境中实现 Linux 和 WSL 的安全使用所需的知识。

建议使用 Microsoft Defender for Endpoint、Intune 和高级网络控制进行企业设置

有多种方法可以设置安全的企业环境，但我们建议使用以下方法来设置利用 WSL 的安全环境。

先决条件

若要开始，请确保所有企业设备都安装了以下最低版本：

- Windows 10 22H2 或更高版本，或者 Windows 11 22H2 或更高版本
  - 高级网络功能仅适用于 Windows 11 22H2 或更高版本。
- WSL 2.0.9 或更高版本
  - 可通过运行 `wsl --version` 来检查 WSL 版本。

启用 Microsoft Defender for Endpoint (MDE) 集成

Microsoft Defender for Endpoint 是一个企业终结点安全平台，专门用于帮助企业网络防御、检测、调查和响应高级威胁。 MDE 现在作为 WSL 插件与 WSL 集成，这样安全团队就可使用 Defender for Endpoint 在所有正在运行的 WSL 发行版中查看并持续监视安全事件，同时将对开发人员工作负载的性能影响降低最小。

若要详细了解如何入门，请参阅适用于 WSL 的 Microsoft Defender for Endpoint 插件。

使用 Intune 配置推荐的设置

Microsoft Intune 是基于云的终结点管理解决方案。 它管理用户对组织资源的访问，并简化了跨多台设备（包括移动设备、台式计算机和虚拟终结点）的应用和设备管理。 可使用 Microsoft Intune 来管理组织内部的设备，这现在还包括管理对 WSL 及其关键安全设置的访问。

有关使用 Intune 管理 WSL 作为 Windows 组件的指导以及推荐的设置，请参阅 Intune 设置用于 WSL。

使用高级网络功能和控件

从 Windows 11 22H2 和 WSL 2.0.9 或更高版本开始，Windows 防火墙规则将自动应用于 WSL。 这可确保 Windows 主机上设置的防火墙规则默认自动应用于所有 WSL 发行版。 有关自定义 WSL 防火墙设置的指南，请访问配置 Hyper-V 防火墙。

此外，建议根据你的特定企业方案配置 [wsl2] 文件中 .wslconfig 下的设置。

镜像模式网络

`networkingMode=mirrored` 启用镜像模式网络。 这种新的网络模式提高了与复杂网络环境的兼容性（尤其是 VPN 等），还增添了对默认 NAT 模式（例如 IPv6）中不可用的新网络功能的支持。

DNS 隧道

`dnsTunneling=true` 更改 WSL 获取 DNS 信息的方式。 此设置改进了不同网络环境中的兼容性，并利用虚拟化功能获取 DNS 信息而不是网络数据包。 如果遇到任何连接问题，建议启用此功能，在使用 VPN、高级防火墙设置等时尤其有用。

自动代理

`autoProxy=true` 强制 WSL 使用 Windows 的 HTTP 代理信息。 建议在 Windows 上使用代理时启用此设置，因为这会使该代理自动应用于 WSL 发行版。

创建自定义的 WSL 映像

通常所说的“映像”只是软件及其组件保存到文件中的快照。 对于适用于 Linux 的 Windows 子系统，映像将由子系统、其分发版以及分发版上安装的所有软件和包组成。

若要开始创建 WSL 映像，请先安装适用于 Linux 的 Windows 子系统。

安装完成后，使用 Microsoft Store 下载并安装适合你的 Linux 分发版。

导出 WSL 映像

通过运行 `wsl --export <Distro> <FileName> [Options]` 导出自定义 WSL 映像，这会将映像包装在 tar 文件中，并使其准备好分发到其他计算机。 可按照自定义发行版指南创建自定义分发版，包括 CentOS、RedHat 等。

分发 WSL 映像

通过运行 `wsl --import <Distro> <InstallLocation> <FileName> [Options]`从共享或存储设备分发 WSL 映像，这将将指定的 tar 文件导入为新的分发版。

更新和修补 Linux 分发版和包

强烈建议使用 Linux 配置管理器工具来监视和管理 Linux 用户空间。 可以从许多 Linux 配置管理器的主机进行选择。 请参阅此博客文章，了解如何在 WSL 2 中快速运行 Puppet 。

Windows 文件系统访问

当 WSL 内的 Linux 二进制文件访问 Windows 文件时，它使用运行 `wsl.exe` 的 Windows 用户的用户权限执行此操作。 因此，即使 Linux 用户在 WSL 中具有根访问权限，如果 Windows 用户没有这些权限，他们也无法在 Windows 上执行 Windows 管理员级操作。 对于来自 WSL 的 Windows 文件和 Windows 可执行文件访问，运行 shell（例如 `bash`）与以该用户身份从 Windows 运行 `powershell` 具有相同的安全级别权限。

支持

- 使用 `wsl --import` 和 `wsl --export` 在内部共享已批准的映像
- 使用 WSL Distro Launcher 存储库 为企业创建自己的 WSL 分发版
- 使用 Microsoft Defender for Endpoint (MDE) 监视 WSL 发行版中的安全事件
- 使用防火墙设置控制 WSL 中的网络（包括将 Windows 防火墙设置同步到 WSL）
- 使用 Intune 或组策略控制对 WSL 及其密钥安全设置的访问

下面是我们尚不支持但正在研究的功能列表。

目前不支持

下面是 WSL 中当前不支持但常被要求提供的功能的列表。 这些请求已成为我们的积压工作(backlog)，我们正在设法添加它们。

- 使用 Windows 工具管理 Linux 分发版和包的更新和修补程序
- Windows 更新也会更新 WSL 分发版的内容
- 控制企业中的用户可以访问哪些分发版
- 控制用户的根访问权限

Last updated on 2025/08/16

WSL 的 Intune 设置

现在可以使用 Intune 等管理工具将 WSL 作为 Windows 组件进行管理。

若要访问这些设置，请导航到 Microsoft Intune 管理中心门户，然后选择： **设备** -> **配置文件** -> **创建** -> **新建策略** -> **Windows 10 及更高版本** -> **设置目录**，为新配置文件创建名称，并搜索“适用于 Linux 的 Windows 子系统”以查看和添加可用设置的完整列表。

推荐设置

若要最大程度地提高企业环境中的安全性，建议指定以下设置：

展开表

设置名称	价值	DESCRIPTION
允许适用于 Linux 的 Windows 子系统的收件箱版本	禁用	设置为“已禁用”时，此策略将禁用适用于 Linux 的 Windows 子系统的收件箱版本（可选组件）。 如果禁用此策略，则只能使用 WSL 的存储版本。 在此处详细了解 Microsoft Store WSL 与收件箱 WSL 之间的差异
允许 WSL1	禁用	设置为禁用时，此策略将禁用 WSL1。 禁用后，只能使用 WSL2 分发版。
允许调试 shell	禁用	设置为禁用时，此策略将禁用调试 shell（wsl.exe --debug-shell）。 此策略仅适用于存储 WSL。
允许自定义内核配置	禁用	设置为禁用时，此策略通过 .wslconfig （wsl2.kernel） 禁用自定义内核配置。 此策略仅适用于存储 WSL。
允许内核命令行配置	禁用	设置为禁用时，此策略通过 .wslconfig（wsl2.kernelCommandLine）禁用内核命令行配置。 此策略仅适用于存储 WSL。
允许自定义系统分发配置	禁用	设置为禁用时，此策略通过 .wslconfig （wsl2.systemDistro） 禁用自定义系统分发配置。 此策略仅适用于存储 WSL。
允许自定义网络配置	禁用	设置为禁用时，此策略通过 .wslconfig （wsl2.networkingmode） 禁用自定义网络配置。 此策略仅适用于存储 WSL。
允许用户设置防火墙配置	禁用	设置为禁用时，此策略通过 .wslconfig （wsl2.firewall） 禁用防火墙配置。 此策略仅适用于存储 WSL。
允许嵌套虚拟化	禁用	设置为禁用时，此策略通过 .wslconfig（wsl2.nestedVirtualization）禁用嵌套虚拟化配置。 此策略仅适用于存储 WSL。
允许内核调试	禁用	设置为禁用时，此策略通过 .wslconfig（wsl2.kernelDebugPort）禁用内核调试配置。 此策略仅适用于存储 WSL。

控制对 WSL 的访问

和`AllowWSL` `AllowInboxWSL` `AllowWSL1`设置控制用户对 WSL 的访问。 可以将这些设置配置为启用或禁用对 WSL、WSL 1 发行版或 WSL 本身的 Windows 中版本的访问权限。

这样，就可以配置 WSL，以确保用户仅使用最新版本的 WSL 和 Enterprise 功能支持。

控制 WSL 命令

`AllowDebugShell` 并 `AllowDiskMount` 控制用户是否可以运行 `wsl --debug-shell` 和 `wsl --mount` 命令。 详细了解如何使用命令 在 WSL 2`wsl --mount` 中装载磁盘。

控制对 WSL 设置的访问 `.wslconfig`

最后一组设置，这些设置以控制对 WSL 高级设置的访问`*UserSettingConfigurable`结尾`.wslconfig`。 当这些设置为禁用时，用户只能使用该设置的默认值，并且无法将其配置为自定义值。 详细了解 .wslconfig 的配置设置，包括可为使用 WSL 2 运行的所有 Linux 分发版全局配置的设置列表。

可用设置的完整列表

展开表

设置名称	DESCRIPTION
允许适用于 Linux 的 Windows 子系统	设置为“已禁用”时，此策略将禁用对计算机上的所有用户对适用于 Linux 的 Windows 子系统的访问。
允许适用于 Linux 的 Windows 子系统的收件箱版本	设置为“已禁用”时，此策略将禁用适用于 Linux 的 Windows 子系统的收件箱版本（可选组件）。 如果禁用此策略，则只能使用 WSL 的存储版本。
允许 WSL1	设置为禁用时，此策略将禁用 WSL1。 禁用后，只能使用 WSL2 分发版。
允许调试 shell	设置为禁用时，此策略将禁用调试 shell（wsl.exe --debug-shell）。 此策略仅适用于存储 WSL。
允许直通磁盘装载	设置为禁用时，此策略将禁用 WSL2（wsl.exe --mount）中的直通磁盘装载。 此策略仅适用于存储 WSL。
允许自定义内核配置	设置为禁用时，此策略通过 .wslconfig （wsl2.kernel） 禁用自定义内核配置。 此策略仅适用于存储 WSL。

设置名称	DESCRIPTION
允许内核命令行配置	设置为禁用时，此策略通过 .wslconfig（wsl2.kernelCommandLine）禁用内核命令行配置。 此策略仅适用于存储 WSL。
允许自定义系统分发配置	设置为禁用时，此策略通过 .wslconfig （wsl2.systemDistro） 禁用自定义系统分发配置。 此策略仅适用于存储 WSL。
允许自定义网络配置	设置为禁用时，此策略通过 .wslconfig （wsl2.networkingmode） 禁用自定义网络配置。 此策略仅适用于存储 WSL。
允许用户设置防火墙配置	设置为禁用时，此策略通过 .wslconfig （wsl2.firewall） 禁用防火墙配置。 此策略仅适用于存储 WSL。
允许嵌套虚拟化	设置为禁用时，此策略通过 .wslconfig（wsl2.nestedVirtualization）禁用嵌套虚拟化配置。 此策略仅适用于存储 WSL。
允许内核调试	设置为禁用时，此策略通过 .wslconfig（wsl2.kernelDebugPort）禁用内核调试配置。 此策略仅适用于存储 WSL。

使用组策略进行设置

WSL 策略由可从 GitHub 下载的 WSL ADMX 文件 定义。

ADMX 文件可以手动导入，并用于在计算机上本地管理组策略， 请参阅 ADMX 文档页 ，了解有关该过程的详细信息。

Last updated on 2025/08/15

有关适用于 Linux 的 Windows 子系统的常见问题解答

概况

什么是适用于 Linux 的 Windows 子系统（WSL）？

适用于 Linux 的 Windows 子系统（WSL）是 Windows作系统的一项功能，可用于运行 Linux 文件系统，以及直接在 Windows 上的 Linux 命令行工具和 GUI 应用，以及传统的 Windows 桌面和应用。

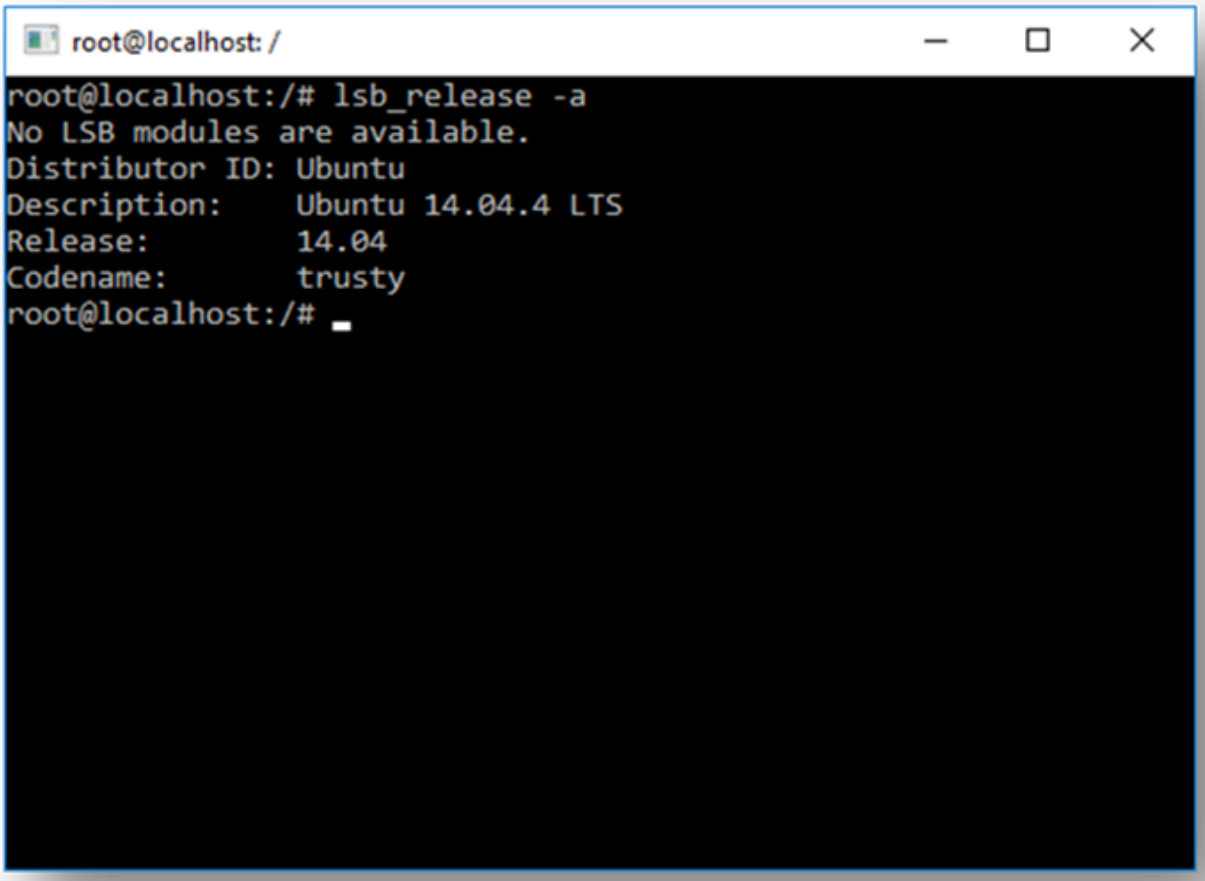
有关更多详细信息，请参阅 “关于”页 。

谁是 WSL?

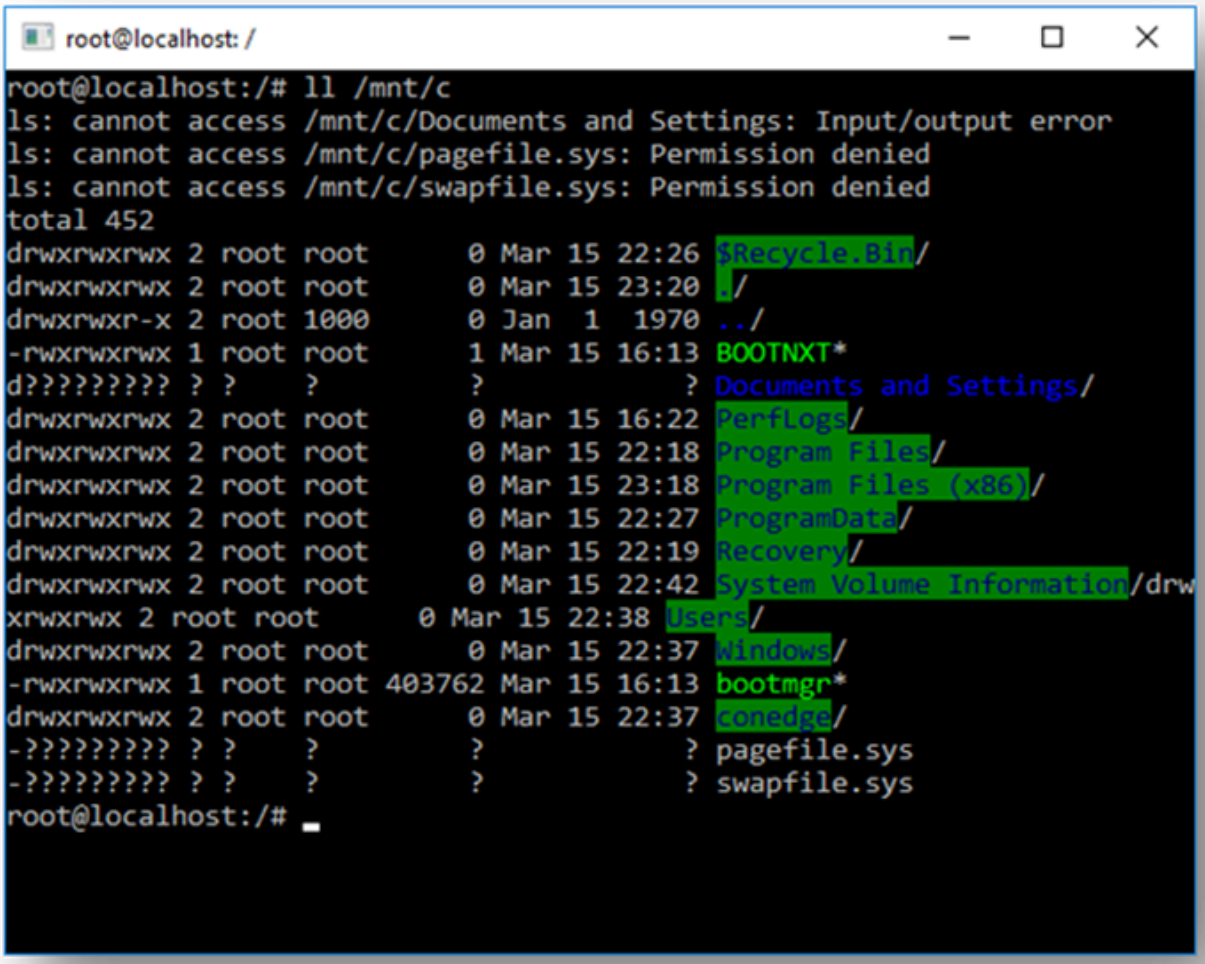
这主要是面向开发人员的工具，尤其是 Web 开发人员、那些处理开源项目或部署到 Linux 服务器环境的工具。 WSL 适用于喜欢使用 Bash、常用 Linux 工具（等）和 Linux 优先框架（`sed` `awk`Ruby、Python 等）的任何人，但也喜欢使用 Windows 生产力工具。

可以使用 WSL 做什么?

使用 WSL，可以使用所选分发版（Ubuntu、Debian、OpenSUSE、Kali、Alpine 等）在 Bash shell 中运行 Linux。 使用 Bash，可以运行命令行 Linux 工具和应用。 例如，键入 `lsb_release -a` 并按 Enter;你将看到当前正在运行的 Linux 发行版的详细信息：



还可以从 Linux Bash shell 内部访问本地计算机的文件系统 - 可以在文件夹下找到装载的 `/mnt` 本地驱动器。 例如，驱动器 `C:` 装载在 `/mnt/c`：



是否可以描述包含 WSL 的典型开发工作流?

WSL 面向开发人员受众，目的是用作内部开发循环的一部分。 假设 Sam 正在创建 CI/CD 管道（持续集成和持续交付），并希望先在本地计算机（笔记本电脑）上对其进行测试，然后再将其部署到云。 Sam 可以启用 WSL （& WSL 2 以提高速度和性能），然后在本地（在笔记本电脑上）使用真正的 Linux Ubuntu 实例，以及他们喜欢的任何 Bash 命令和工具。 在本地验证开发管道后，Sam 可以将该 CI/CD 管道推送到云（即 Azure），方法是将其推送到 Docker 容器，并将容器推送到在生产就绪的 Ubuntu VM 上运行的云实例。

什么是 Bash?

Bash 是基于文本的常用 shell 和命令语言。 它是 Ubuntu 和其他 Linux 发行版中包含的默认 shell。 用户将命令键入 shell 以执行脚本和/或运行命令和工具来完成许多任务。

这是如何运作的?

请查看 Windows 命令行博客上的本文： 深入了解 WSL 如何允许 Windows 访问有关基础技术的 Linux 文件 。

为什么在 VM 中使用 WSL 而不是 Linux?

WSL 所需的资源（CPU、内存和存储）比完整虚拟机少。 WSL 还允许你与 Windows 命令行、桌面和应用商店应用一起运行 Linux 命令行工具和应用，以及从 Linux 内部访问 Windows 文件。 这使你可以在同一组文件上使用 Windows 应用和 Linux 命令行工具（如果需要）。

为什么使用 Ruby on Linux 而不是 Windows?

假设运行它们的环境类似于 Linux，则生成了一些跨平台工具。 例如，某些工具假定它们能够访问很长的文件路径或存在特定文件/文件夹。 这通常会导致 Windows 上的行为与 Linux 不同。

许多语言（如 Ruby 和 Node.js）通常移植到 Windows 上并运行出色。 但是，并非所有 Ruby Gem 或 node/NPM 库所有者都移植其库以支持 Windows，并且许多库具有特定于 Linux 的依赖项。 这通常会导致使用此类工具和库生成的系统，并且有时在 Windows 上出现运行时错误或不需要的行为。

这只是一些问题，导致许多人要求Microsoft改进 Windows 的命令行工具，以及是什么促使我们与 Canonical 合作，使本机 Bash 和 Linux 命令行工具能够在 Windows 上运行。

这对 PowerShell 意味着什么?

使用 OSS 项目时，有许多场景，从 PowerShell 提示符进入 Bash 非常有用。 Bash 支持是互补的，增强了 Windows 上命令行的价值，使 PowerShell 和 PowerShell 社区能够利用其他常用技术。

在 PowerShell 团队博客上阅读详细信息 - Bash for Windows：为什么它很棒，以及 PowerShell 的含义

WSL 支持哪些处理器?

WSL 支持 x64 和 Arm64 CPU。

如何访问 C： 驱动器?

本地计算机上的硬盘驱动器的装入点会自动创建，并提供对 Windows 文件系统的轻松访问。

/mnt/<drive letter>/

示例用法是 `cd /mnt/c` 访问 `c:\`

如何设置 Git 凭据管理器? （如何在 WSL 中使用 Windows Git 权限? ）

请参阅教程： 开始在适用于 Linux 的 Windows 子系统上使用 Git，其中包含有关在 Windows 凭据管理器中设置 Git 凭据管理器和存储身份验证令牌的部分。

如何将 Windows 文件与 Linux 应用配合使用?

WSL 的优势之一是通过 Windows 和 Linux 应用或工具访问文件。

WSL 将计算机的固定驱动器装载到 Linux 发行版中的文件夹下 `/mnt/<drive>` 。 例如，驱动器 `C:` 装载在 `/mnt/c/`.

使用装载的驱动器，可以通过通过`/mnt/c/dev/myproj`访问相同的文件来编辑代码，例如 `C:\dev\myproj\`，使用 Visual Studio 或 VS Code 编辑代码，并在 Linux 中生成/测试代码。

详细了解 如何跨 Windows 和 Linux 文件系统 工作文章。

Linux 驱动器中的文件是否不同于装载的 Windows 驱动器?

1. Linux 根下的文件（即 `/`）由 WSL 控制，与 Linux 行为保持一致，包括但不限于：

    - 包含无效 Windows 文件名字符的文件
    - 为非管理员用户创建的符号链接
    - 通过 `chmod` 和更改文件属性 `chown`
    - 文件/文件夹区分大小写

2. 装载驱动器中的文件由 Windows 控制，并具有以下行为：

    - 支持区分大小写
    - 所有权限都设置为最能反映 Windows 权限

如何卸载 WSL 分发版?

若要从 WSL 中删除分发版并 *删除与该 Linux 分发版关联的所有数据*，请运行 `wsl --unregister <Distro>` `<Distro>` Linux 发行版的名称，该发行版可从命令中的 `wsl -l` 列表中查看。

此外，还可以像任何其他应用商店应用程序一样卸载计算机上的 Linux 发行版应用。

若要了解有关 wsl 命令的详细信息，请参阅文章： WSL 的基本命令。

如何运行 OpenSSH 服务器?

OpenSSH 附带 Windows 作为可选功能。 请参阅 安装 OpenSSH 文档。在 Windows 中，需要管理员权限才能在 WSL 中运行 OpenSSH。 若要运行 OpenSSH 服务器，请以管理员身份运行

WSL 分发（即 Ubuntu）或 Windows 终端。 有几种资源涵盖 WSL 的 SSH 方案。 查看 Scott Hanselman 的博客文章： 如何从 Linux 或 Windows OR 随处通过 SSH 连接到 Windows 10 计算机， 如何从外部计算机通过 SSH 连接到 Windows 10 上的 WSL2，以及如何从外部计算机通过 WINDOWS 10 通过 SSH 连接到 Bash 和 WSL2， 以及如何使用 Windows 10 的内置 OpenSSH 自动通过 SSH 连接到远程 Linux 计算机。

如何更改 WSL 的显示语言?

WSL 安装将尝试自动更改 Ubuntu 区域设置以匹配 Windows 安装的区域设置。 如果不希望此行为，可以在安装完成后运行此命令来更改 Ubuntu 区域设置。 必须重新启动 WSL 分发才能使此更改生效。

以下示例将区域设置更改为 en-US：

Bash

```
sudo update-locale LANG=en_US.UTF8
```

为什么我没有来自 WSL 的 Internet 访问?

某些用户报告了特定防火墙应用程序阻止 WSL 中的 Internet 访问的问题。 报告的防火墙包括：

1. 卡巴斯基
2. AVG
3. Avast
4. Symantec Endpoint Protection
5. F-Secure

在某些情况下，关闭防火墙允许访问。 在某些情况下，只需安装防火墙即可阻止访问。

如何从 Windows 中的 WSL 访问端口?

WSL 共享 Windows 的 IP 地址，因为它在 Windows 上运行。 因此，可以在 localhost 上访问任何端口，例如，如果端口 1234 上有 Web 内容，则可以 https://localhost:1234 进入 Windows 浏览器。 有关详细信息，请参阅 访问网络应用程序。

如何备份 WSL 分发版?

备份或移动分发版的最佳方式是通过 Windows 版本 1809 及更高版本中提供的 导出/导入 命令。 可以使用命令将整个分发导出到 tarball `wsl --export` 。 然后，可以使用命令将此分发重新导入到 WSL `wsl --import` 中，该命令可以命名导入的新驱动器位置，从而允许备份和保存

WSL 分发版（或移动）状态。 若要了解有关移动 WSL 分发版的详细信息，请参阅如何将 WSL 文件从一台计算机传输到另一台计算机?

在 AppData 文件夹中备份文件（如 Windows 备份）的传统备份服务不会损坏 Linux 文件。

是否可以将 WSL 用于生产方案?

是的，但是 WSL 已设计并构建为与内部循环开发工作流一起使用。 WSL 中有一些设计功能，使其非常适合此目的，但与其他产品相比，它可能会使生产相关方案具有挑战性。 我们的目标是明确 WSL 与常规 VM 环境的不同之处，以便你可以就它是否符合业务需求做出决定。

WSL 与传统生产环境之间的主要区别包括：

- WSL 具有一个轻型实用工具 VM，可自动启动、停止和管理资源。
- 如果 Windows 进程没有打开的文件句柄，WSL VM 将自动关闭。 这意味着，如果将它用作 Web 服务器，请通过 SSH 连接到它以运行服务器，然后退出，VM 可能会关闭，因为它正在检测用户已完成使用它，并清理其资源。
- WSL 用户对其 Linux 实例具有完全访问权限。 VM 的生存期、已注册的 WSL 分发版等都可以由用户访问，可由用户修改。
- WSL 自动授予文件对 Windows 文件的访问权限。
- 默认情况下，Windows 路径会追加到路径中，这可能会导致某些 Linux 应用程序与传统 Linux 环境相比出现意外行为。
- WSL 可以从 Linux 运行 Windows 可执行文件，这也可能导致不同于传统 Linux VM 的环境。
- WSL 使用的 Linux 内核会自动更新。
- WSL 中的 GPU 访问通过将 `/dev/dxg` GPU 调用路由到 Windows GPU 的设备进行。 此设置不同于传统的 Linux 设置。
- 与裸机 Linux 相比，还有其他较小的差异，预计将来会出现更多的差异，因为内部循环开发工作流是优先顺序的。

如何将 WSL 文件从一台计算机传输到另一台计算机?

可通过几种方法完成此任务：

- 最简单的方法是使用 `wsl --export <Distro> <FileName> --format vhd` 命令将 WSL 分发版导出到 VHD 文件。 然后，可以将此文件复制到另一台计算机，并使用它导入它 `wsl --import <Distro> <InstallLocation> <FileName> --vhd`。 有关详细信息，请参阅 WSL 基本命令文档中的导入和导出命令。
- 上述实现需要大量的磁盘空间。 如果没有大量磁盘空间，则可以使用 Linux 技术将文件移出：
  - 用于 `tar -czf <tarballName> <directory>` 创建文件的 tarball。 然后，可以将这些特定文件复制到新计算机，然后运行 `tar -xzf <tarballName>` 以提取它们。

- 还可以使用 `apt` 如下所示的命令导出已安装的包列表：`dpkg --get-selections | grep -v deinstall | awk '{print $1}' > package_list.txt` 然后使用命令在另一台计算机上重新安装这些相同包，例如 `sudo apt install -y $(cat package_list.txt)` 在传输文件之后。

如何将 WSL 分发版移动到其他驱动器或位置?

可以使用 PowerShell 执行此作。 下面是每个步骤的必要命令和说明。 请打开 PowerShell 窗口并调整标记之间的 `<>` 值以适应特定的用例：

PowerShell

```
# Export your distro to that folder as a VHD
wsl --export <Distro, e.g: Ubuntu> <FileName, e.g: D:\WSLDistros\Ubuntu\ext4.vhdx> --format vhd

# Unregister your old distro
# Please note this will erase your existing distro's file contents, please ensure the backup file you created in the 2nd step is present at the location and that the export operation completed successfully.
# Please exercise caution when using this command, as it is destructive and could cause data loss.
wsl --unregister <Distro, e.g: Ubuntu>

# Import your VHD backup
wsl --import-in-place <Distro, e.g: Ubuntu> <FileName, e.g: D:\WSLDistros\Ubuntu\ext4.vhdx>
```

如何设置默认用户帐户

可以通过设置wsl.conf值user.default=<name>来设置默认用户帐户

WSL 2

WSL 2 在 Windows 10 家庭版和 Windows 11 家庭版上是否可用?

是的。 WSL 2 在 WSL 可用的所有桌面 SKU 上可用，包括 Windows 10 家庭版和 Windows 11 家庭版。

具体而言，WSL2 需要启用两个功能：

1. “虚拟机平台”（Hyper-V子集）
2. “适用于 Linux 的 Windows 子系统”

WSL 2 是否使用 Hyper-V?

最新版本的 WSL 使用一 *部分* Hyper-V 体系结构来实现其虚拟化。 此子集作为名为“虚拟机平台”的可选组件提供，可在所有桌面 SKU 上使用。

WSL 1 会发生什么情况？ 会放弃吗?

我们目前没有计划弃用 WSL 1。 可以并行运行 WSL 1 和 WSL 2 发行版，并且可以随时升级和降级任何发行版。 将 WSL 2 添加为新体系结构为 WSL 团队提供了更好的平台，以提供使 WSL 成为在 Windows 中运行 Linux 环境的绝佳方法的功能。

能否运行 WSL 2 和其他第三方虚拟化工具，例如 VMware 或 VirtualBox?

在使用 Hyper-V 时，某些第三方应用程序无法正常工作，这意味着当启用了 WSL 2（如 VMware 和 VirtualBox）时，它们将无法运行。 但是，最近，VirtualBox 和 VMware 都发布了支持 Hyper-V 和 WSL2 的版本。 在此处详细了解 VirtualBox 的更改，以及此处的 VMware 更改。 若要排查问题，请查看 GitHub 上的 WSL 存储库中的 VirtualBox 问题讨论。 StackOverflow 还提供了一个有用的提示： 如何同时使 VirtualBox 6.0 和 WSL 正常工作。

我们一贯致力于支持 Hyper-V 的第三方集成的解决方案。 例如，我们公开了一组称为 虚拟机监控程序平台 的 API，第三方虚拟化提供程序可以使用这些 API 使其软件与 Hyper-V 兼容。 这使应用程序可以使用 Hyper-V 体系结构进行仿真，例如 Google Android Emulator 和 VirtualBox 6 及更高版本，它们现在都与 Hyper-V 兼容。

有关 VirtualBox 6.1 的 WSL 2 问题 的详细信息和讨论，请参阅 WSL 问题存储库。

*如果要查找 Windows 虚拟机、VMWare、Hyper-V、VirtualBox 和 Parallels VM 下载，可在 Windows 开发人员中心获取。

是否可以在 WSL 2 中访问 GPU？ 是否有计划增加硬件支持?

我们发布了在 WSL 2 分发版中访问 GPU 的支持！ 这意味着，在涉及大数据集时，现在可以更轻松地将 WSL 用于机器学习、人工智能和数据科学方案。 查看 GPU 支持教程入门 。 截至目前，WSL 2 不包括串行支持或 USB 设备支持。 我们正在调查添加这些功能的最佳方式。 但是，USB 支持现已通过 USBIPD-WIN 项目提供。 有关设置 USB 设备支持的步骤，请参阅 “连接 USB 设备”。

WSL 2 是否可以使用网络应用程序?

是的，在一般网络应用程序中，WSL 2 的运行效果更好且速度更快，因为它提供完整的系统调用兼容性。 但是，WSL 2 体系结构使用虚拟化网络组件，这意味着 WSL 2 的行为类似于虚拟机 -- WSL 2 分发版将具有不同于主机（Windows OS）的 IP 地址。 有关详细信息，请参阅 使用 WSL 访问网络应用程序。

是否可以在虚拟机中运行 WSL 2?

是的！ 需要确保虚拟机已启用嵌套虚拟化。 可以通过在具有管理员权限的 PowerShell 窗口中运行以下命令，在父 Hyper-V 主机中启用此功能：

PowerShell

```
Set-VMProcessor -VMName <VMName> -ExposeVirtualizationExtensions $true
```

请确保将“VMName>”<替换为虚拟机的名称。

是否可以在 WSL 2 中使用 wsl.conf?

WSL 2 支持 WSL 1 使用的相同 wsl.conf 文件。 这意味着你在 WSL 1 发行版中设置的任何配置选项（例如自动装载 Windows 驱动器、启用或禁用互作、更改将装载 Windows 驱动器的目录等）都将在 WSL 2 内部工作。 可以在 “分发管理 ”页中详细了解 WSL 中的配置选项。 有关在 WSL 2 中装载 Linux 磁盘 中装载驱动器、磁盘、设备或虚拟硬盘（VHD）的支持的详细信息。

1：VirtualBox 6.0 2 的更改日志：Hyper-V API3：Hyper-V Android 模拟器支持4：VMware 工作站技术预览版 20H1 — Hyper-V/主机 VBS 支持

我可以在哪里提供反馈?

WSL 产品仓库问题可以让你：

- **搜索现有问题** ，查看是否存在与你存在的问题相关的问题。 请注意，在搜索栏中，可以删除“is：open”以包括已在搜索中解决的问题。 请考虑对任何未解决的问题进行评论或点赞，表达你希望将其作为优先事项推动的兴趣。
- **提交新问题**。 如果发现 WSL 存在问题且不存在现有问题，则可以选择绿色 *的“新建问题”* 按钮，然后选择 *“WSL - Bug 报告”*。 你将需要包含问题的标题、Windows 内部版本号（运行 `cmd.exe /c ver` 以查看当前内部版本号）、你是运行 WSL 1 还是 2、当前 Linux 内核版本号（运行 `wsl.exe --status` 或 `cat /proc/version`）、你使用的发行版版本号（运行 `lsb_release -r`）、涉及的任何其他软件版本、复现步骤、预期行为、实际行为，以及诊断日志（如果可用且适用）。 有关详细信息，请参阅 参与 WSL。
- 选择绿色**的“新建问题”**按钮，然后选择*“功能请求”来提交功能请求*。 你需要回答几个描述您请求的问题。

您还可以：

- 使用 **WSL 文档存储库**提交文档问题。 若要参与 WSL 文档，请参阅 Microsoft Docs 参与者指南。
- 如果你的问题与 Windows 终端、Windows 控制台或命令行 UI 相关，请使用 Windows 终端产品存储库提交 Windows 终端问题。

如果想要随时了解最新的 WSL 新闻，可以执行以下作：

- 命令行团队博客
- X. 请按照 X 上的@craigaloewen 了解新闻、更新等信息。

如何解决“错误：0x800704ec组策略阻止了此程序。有关详细信息，请与系统管理员联系。'?

此错误是由阻止 WSL 的组策略引起的。 若要解决此问题，请先运行命令 `wsl --update` 以更新到最新的应用商店版本。 如果这无法解决问题，请与管理员联系。 详细了解收件箱 WSL 与 WSL 的应用商店版本之间的差异： Microsoft 应用商店中的 WSL。

LxssManager 是否被 WSLService 取代?

是的，当 WSL 从收件箱组件转换到由 Microsoft 应用商店提供服务时，LxssManager 已替换为 WSLService。

排查适用于 Linux 的 Windows 子系统问题

下面介绍了一些与 WSL 关联的常见故障排除方案，但请考虑在 GitHub 上的 WSL 产品存储库 中搜索问题。

提交问题、bug 报告、功能请求

WSL 产品仓库问题 可以让你：

- **搜索现有问题**，查看是否存在与你存在的问题相关的问题。

  请注意，在搜索栏中，可以删除“**state:open**”以包括已在搜索中解决的问题。

  请考虑对任何未解决的问题进行评论或点赞，表达你希望将其作为优先事项推动的兴趣。

- **提交新问题。** 如果您发现 WSL 出现了问题，并且似乎没有现有问题，则可以选择绿色的“**New issue**”按钮，然后选择“**WSL - Bug Report**”。

  您需要提供以下信息：
  - 问题标题
  - Windows 版本：请运行 `cmd.exe /c ver` 以获取你正在使用的 Windows 版本。
  - WSL 版本：如果要从 Microsoft 应用商店运行适用于 Linux 的 Windows 子系统，请运行 `wsl.exe -v`。
  - 是否使用 WSL 1 或 WSL 2：告诉我们问题是否在 WSL 2 和/或 WSL 1 上。 可以通过运行 `wsl.exe -l -v`来判断。
  - 内核版本：请告诉我们所使用的 Linux 内核版本，或者使用的是自定义内核。 如果该命令可供你使用，你可以运行 `wsl.exe --status`，或者在你的发行版中运行 `cat /proc/version`。
  - 发行版版本：请告诉我们你正在使用哪些发行版（如果适用）。 可以在可能的情况下获取有关版本的其他信息，例如在 Debian/Ubuntu `run lsb_release -r`上。
  - 其他软件：如果要报告涉及 WSL 和其他应用程序交互的 bug，请告知我们。 哪些应用程序？ 哪些版本?
  - 重现步骤：请列出重现错误的步骤。
  - 预期行为：你期望看到什么？ 包括任何相关的示例或文档链接。
  - 实际情况：究竟发生了什么?
  - 诊断日志：如果需要，请提供其他诊断。 有关如何收集反馈中心日志、网络日志等的信息，请参阅指南。

  有关详细信息，请参阅 参与 WSL。

- 通过选择绿色的“”按钮，然后选择“”**New issue**来**Feature request**。

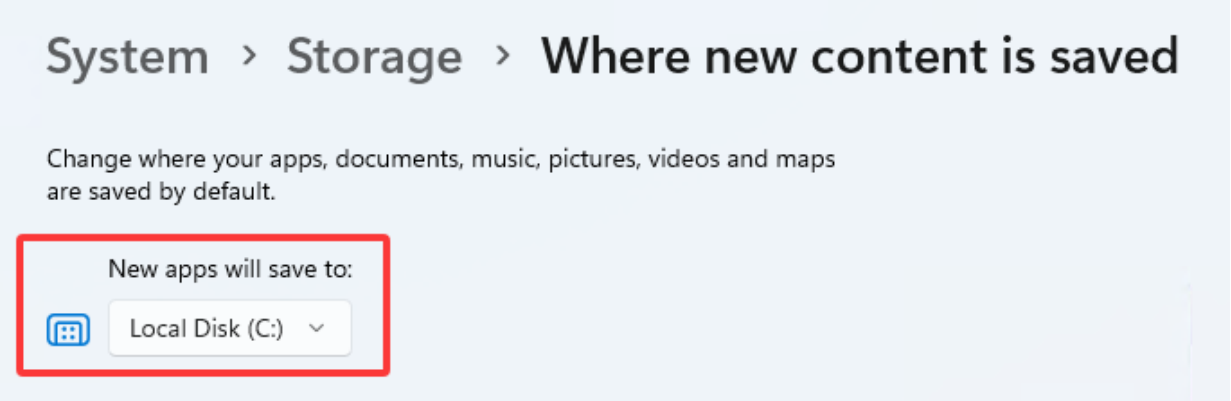
  你需要回答几个描述您请求的问题。

您还可以：

- 使用 **WSL 文档存储库**提交文档问题。 若要参与 WSL 文档，请参阅 Microsoft Docs 参与者指南。
- 如果问题与 Windows 终端、Windows 控制台或命令行 UI 相关，请使用 **Windows 终端产品存储库**提交 Windows 终端问题。

安装问题

- **安装失败，出现错误0x80070003**
  - 适用于 Linux 的 Windows 子系统仅在系统驱动器上运行（通常是你的 `C:` 驱动器）。 确保分发存储在系统驱动器上：
  - 在 Windows 10 上打开**设置** -> -**存储** ->
  - 在 Windows 11 上打开"**设置**-> ->**存储** ->**高级存储设置** "->**其中保存新内容的**



- **WslRegisterDistribution 失败，出现错误0x8007019e**
  - 未启用适用于 Linux 的 Windows 子系统可选组件：
  - 打开**控制面板** ->**程序和功能** ->**打开或关闭 Windows 功能** -> 检查**适用于 Linux 的 Windows 子系统**，或使用步骤 1 中提到的 PowerShell cmdlet。

- **安装失败，出现错误0x80070003或错误0x80370102**
  - 请确保在计算机的 BIOS 中启用虚拟化。 如何执行这一过程的说明因计算机的不同而有所不同，很可能在 CPU 相关选项下找到。
  - WSL2 要求 CPU 支持第二级地址转换（SLAT）功能，该功能是在 Intel Nehalem 处理器（Intel Core 1st Generation）和 AMD Opteron 中引入的。 旧的 CPU（如 Intel Core 2 Duo）将无法运行 WSL2，即使虚拟机平台已成功安装。

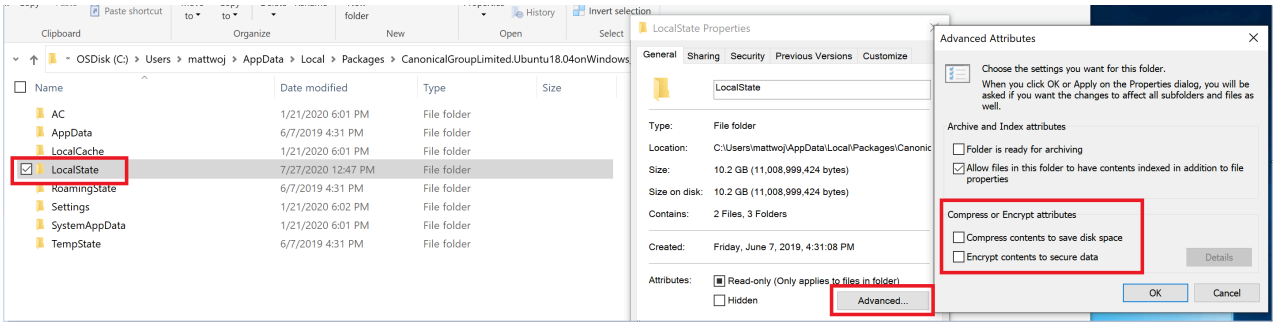
- **尝试升级时出错，命令行选项无效：** `wsl --set-version Ubuntu 2`

  - 确保已启用适用于 Linux 的 Windows 子系统，并且使用的是 Windows 内部版本 18362 或更高版本。 若要启用 WSL，请在具有管理员权限的 PowerShell 提示符中运行以下命令：

    PowerShell

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

- **由于虚拟磁盘系统限制，无法完成请求的作。 虚拟硬盘文件必须未压缩且未加密，并且不得稀疏。**
  - 通过打开 Linux 分发版的配置文件夹，取消选择“**压缩内容**”（如果勾选了“**加密内容**”，也取消选择该项）。 它应位于 Windows 文件系统上的文件夹中，如下所示：`%LocalAppData%\Packages\CanonicalGroupLimited...`
  - 在此 Linux 发行版配置文件中，应有 LocalState 文件夹。 右键单击此文件夹以显示选项菜单。 选择 “**属性**>**高级** ”，并确保“**压缩内容以节省磁盘空间**”和“**加密内容以保护数据**”复选框未选中（未选中）。 如果系统询问是将此应用于当前文件夹还是应用于所有子文件夹和文件，请选择“**仅此文件夹**”，因为你只是清除压缩标志。 之后，该 `wsl --set-version` 命令应正常工作。



> **ⓘ 注意**
>
> 在本例中，Ubuntu 18.04 分发版的 LocalState 文件夹位于 `C:\Users\<my-user-name>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc`
>
> 检查 WSL Docs GitHub 线程 #4103 ，其中正在跟踪此问题以获取最新信息。

- **术语“wsl”无法识别为 cmdlet、函数、脚本文件或可执行程序的名称。**
  - 确保 已安装适用于 Linux 的 Windows 子系统可选组件。 此外，如果使用 ARM64 设备并从 PowerShell 运行此命令，将收到此错误。 而是从 `wsl.exe` 或命令提示符运行。

- **错误：适用于 Linux 的 Windows 子系统未安装分发版。**
  - 如果已安装 WSL 分发版后收到此错误：

    1. 在从命令行调用程序之前，至少运行一次该程序。
    2. 检查是否正在运行单独的用户帐户。 使用提升的权限运行主用户帐户（在管理员模式下）不应导致此错误，但应确保不会意外运行 Windows 附带的内置管理员帐户。 这是一个单独的用户帐户，不会按设计显示任何已安装的 WSL 分发版。 有关详细信息，请参阅 “启用和禁用内置管理员帐户”。

3. WSL 可执行文件仅安装到本机系统目录。 当你在 64 位 Windows 上运行 32 位进程（或在 ARM64 上运行任何非本机组合），托管的非本机进程实际上会看到不同的 System32 文件夹。 在 x64 Windows 上，32 位进程所看到的版本存储在磁盘的 %SystemRoot%\SysWOW64 中。你可以通过查看虚拟文件夹，从托管进程中访问“原生”的 system32。 它实际上不会出现在磁盘上，但文件系统路径解析程序会发现它。

- **错误：此更新仅适用于具有适用于 Linux 的 Windows 子系统的计算机。**
  - 若要安装 Linux 内核更新 MSI 包，需要 WSL，并且应首先启用。 如果失败，将看到消息：此更新仅适用于具有适用于 Linux 的 Windows 子系统的计算机。
  - 有三种可能的原因可以看到此消息：

    1. 你仍处于不支持 WSL 2 的旧版 Windows 中。 请参阅 步骤 2 - 检查运行 WSL 2 的要求。

    2. 未启用 WSL。 需要返回到 步骤 1 ，并确保在计算机上启用了可选的 WSL 功能。

    3. 启用 WSL 后，需要重新启动才能生效，请重新启动计算机，然后重试。

- **错误：WSL 2 需要更新其内核组件。 有关信息，请访问 步骤 4**
  - 如果文件夹中缺少 `%SystemRoot%\system32\lxss\tools` Linux 内核包，将遇到此错误。 通过在安装说明 的步骤 4 中安装 Linux 内核更新 MSI 包来解决此问题。 可能需要从“添加或删除程序”卸载 MSI，然后再次安装它。

常见问题

我在 Windows 10 版本 1903 上，我仍然看不到 WSL 2 的选项

这可能是因为您的计算机尚未获取 WSL 2 的补丁。 解决此问题的最简单方法是转到 Windows 设置 并单击“**检查更新**”，在系统上安装最新更新。 请参阅 有关获取回溯的完整说明 。

如果点击“**检查更新**”，但仍未收到更新，则可以 手动安装 KB KB4566116 。

错误：0x1bc 当 `wsl --set-default-version 2`

当“显示语言”或“系统区域设置”不是英语时，可能会发生这种情况。

PowerShell

```
PS C:\> wsl.exe --set-default-version 2
Error: 0x1bc
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
```

实际错误 `0x1bc` 为：WSL 2 需要更新其内核组件。 有关信息，请访问 https://aka.ms/wsl2kernel

有关详细信息，请参阅问题 #5749

无法访问 Windows 中的 WSL 文件

9p 协议文件服务器在 Linux 端提供服务，以允许 Windows 访问 Linux 文件系统。 如果无法在 Windows 上访问 WSL `\\wsl$` ，可能是因为 9P 未正确启动。

要检查此项，可以通过 `dmesg | grep 9p` 查看启动日志，这会显示任何错误。 成功的输出如下所示：

Bash

```
[    0.363323] 9p: Installing v9fs 9p2000 file system support
[    0.363336] FS-Cache: Netfs '9p' registered for caching
[    0.398989] 9pnet: Installing 9P2000 support
```

有关此问题的进一步讨论，请参阅问题 #5307 。

无法启动 WSL 2 发行版，并且在输出中只能看到“WSL 2”

如果显示语言不是英语，则可能会看到错误文本的截断版本。

PowerShell

```
PS C:\> wsl.exe
WSL 2
```

若要解决此问题，请参阅 步骤 4 ，并按照该文档页上的说明手动安装内核。

`command not found` 在 Linux 中执行 Windows .exe 时

用户可以直接从 Linux 运行 windows 可执行文件，例如 notepad.exe。 有时，您可能会遇到“找不到命令”的情况，如下所示：

Bash

```
$ notepad.exe
-bash: notepad.exe: command not found
```

如果在 $PATH 中没有 Win32 路径，系统将找不到 .exe。 可以通过在 Linux 中运行 `echo $PATH` 来验证它。 预期会在输出中看到 Win32 路径（例如 `/mnt/c/Windows`）。 如果看不到任何

Windows 路径，则很可能你的 PATH 被 Linux shell 覆盖。

下面是 Debian 上导致该问题的示例 `/etc/profile` ：

```Bash
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
```

Debian 的正确方法是删除上述行。 您也可以在进行以下配置时追加$PATH，但这可能导致 WSL 和 VSCode 的其他问题。

```Bash
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$PATH"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:$PATH"
fi
```

有关详细信息，请参阅问题 #5296 和问题 #5779 。

“错误：0x80370102无法启动虚拟机，因为未安装必需的功能。

请启用虚拟机平台 Windows 功能，并确保在 BIOS 中启用虚拟化。

1. 检查 Hyper-V 系统要求
2. 如果计算机是 VM，请手动启用 嵌套虚拟化 。 使用管理员启动 powershell，并运行以下命令，将其替换为 `<VMName>` 主机系统上虚拟机的名称（可以在 Hyper-V Manager 中找到该名称）：

   ```PowerShell
   Set-VMProcessor -VMName <VMName> -ExposeVirtualizationExtensions $true
   ```

3. 请按照电脑制造商关于如何启用虚拟化的指南进行作。 一般情况下，这可能需要使用系统 BIOS 来确保在 CPU 上启用这些功能。 有关此过程的说明可能因计算机而异，请参阅 “在 Windows 上启用虚拟化 ”。
4. 启用 **虚拟机平台**可选组件后重启计算机。

5. 请确保在启动配置中启用了虚拟化程序。 可以通过运行具有管理员权限的 PowerShell 来验证这一点：

   ```
   PowerShell
   bcdedit /enum | findstr -i hypervisorlaunchtype
   ```

   如果看到 `hypervisorlaunchtype Off`，则说明虚拟机监控程序已被禁用。 若要启用它，请在以管理员身份运行的 PowerShell 中运行：

   ```
   PowerShell
   bcdedit /set hypervisorlaunchtype Auto
   ```

6. 此外，如果安装了第三方虚拟机监控程序（例如 VMware 或 VirtualBox），请确保在支持 HyperV（VMware 15.5.5+ 和 VirtualBox 6+）的最新版本上安装了这些虚拟机监控程序，或者已关闭。

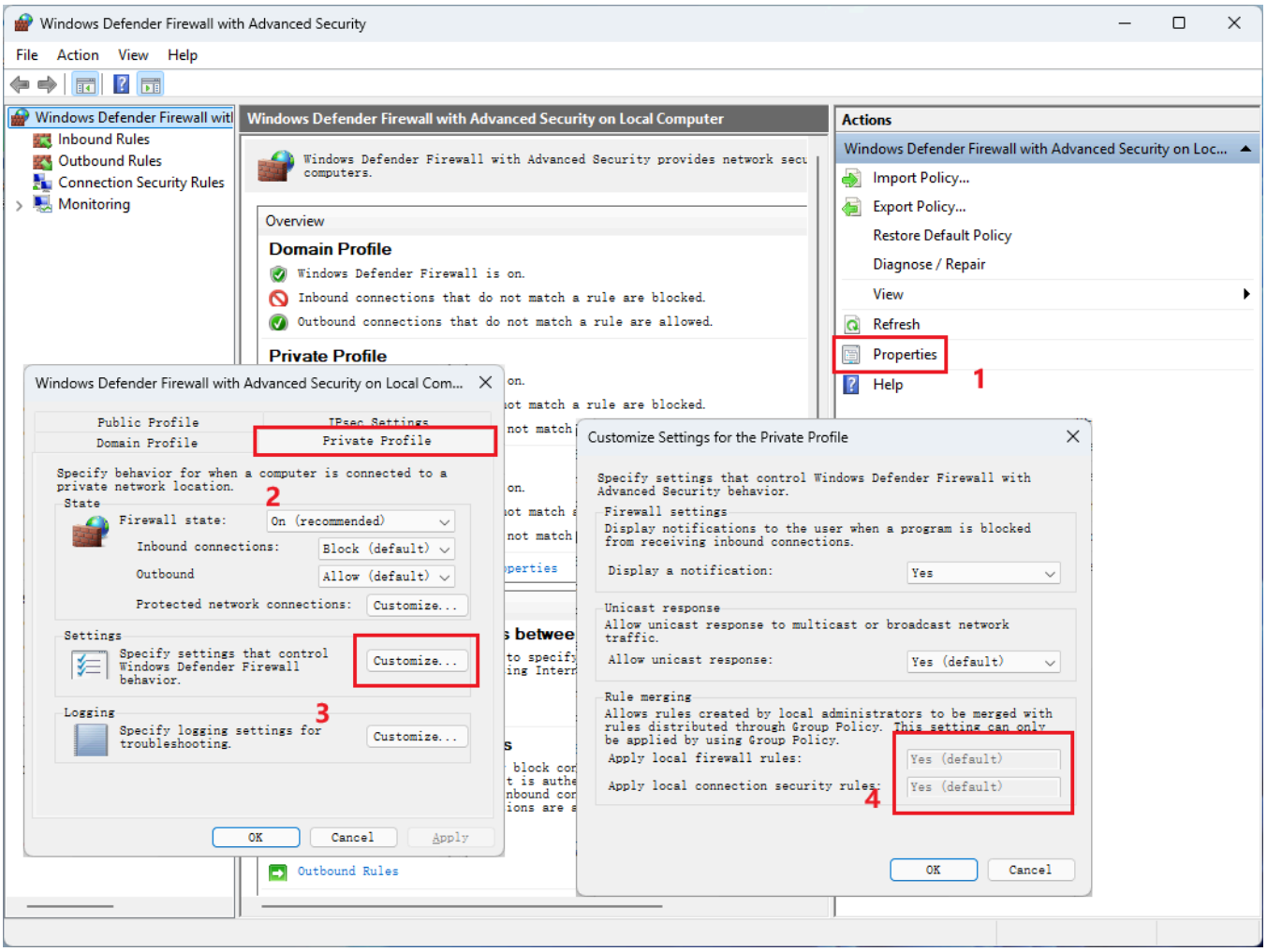
7. 如果在 Azure 虚拟机上收到此错误，请检查以确保禁用 受信任的启动 。 Azure 虚拟机在受信任启动下不支持嵌套虚拟化。

详细了解如何在虚拟机中运行 Hyper-V 时 配置嵌套虚拟化 。

WSL 在工作计算机或企业环境中没有网络连接

业务或企业环境可能 已将 Windows Defender 防火墙设置配置为 阻止未经授权的网络流量。 如果 本地规则合并 设置为“否”，则默认情况下 WSL 网络将不起作用，并且管理员需要添加防火墙规则以允许它。

可以按照以下步骤确认本地规则合并设置：



1. 打开“具有高级安全性的 Windows Defender 防火墙” *(这与控制面板中的“Windows Defender 防火墙”不同)*
2. 右键单击“本地计算机上具有高级安全性的 Windows Defender 防火墙”选项卡
3. 选择“属性”
4. 选择打开的新窗口中的“公共配置文件”选项卡
5. 在“设置”部分下选择“自定义”
6. 在打开的“自定义公共配置文件的设置”窗口中检查“规则合并”是否设置为“否”。 这会阻止访问 WSL。

可以在 “配置 Hyper-V 防火墙”中找到有关如何更改此防火墙设置的说明。

禁用 IPv6 时，WSL 没有网络连接

如果使用注册表值 `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip6\Parameters` `(REG_DWORD) DisabledComponents`禁用 IPv6，则 WSL 网络连接可能会失败。 请参阅 有关在 Windows 中为高级用户配置 IPv6 的指南。

如果必须在主机系统上禁用 IPv6，我们建议使用 Powershell 直接删除 IPv6 协议绑定来执行此作。 例如： `Disable-NetAdapterBinding -Name "<MyAdapter>" -ComponentID ms_tcpip[6]`。

WSL 连接到 VPN 后没有网络连接

如果在 Windows 上连接到 VPN 后，bash 会失去网络连接，请尝试在 bash 中解决此问题。 此解决方法允许你通过手动方式替代 DNS 解析 `/etc/resolv.conf`。

1. 记下 VPN 的 DNS 服务器，执行以下操作：

   PowerShell
   ```
   ipconfig.exe /all
   ```

2. 创建现有 `resolv.conf` 副本：

   Bash
   ```
   sudo cp /etc/resolv.conf /etc/resolv.conf.bak
   ```

3. 取消当前 `resolv.conf` 的链接：

   Bash
   ```
   sudo unlink /etc/resolv.conf
   ```

4. 运行：

   Bash
   ```
   sudo mv /etc/resolv.conf.bak /etc/resolv.conf
   ```

5. 编辑 `/etc/wsl.conf` 此内容并将其添加到文件。 （有关此设置的详细信息，请参阅 高级设置配置）

   Bash
   ```
   [network]
   generateResolvConf=false
   ```

6. 打开 `/etc/resolv.conf` 和
   a. 从包含描述自动生成的注释的文件中删除第一行。
   b. 将上面的 DNS 条目添加为 DNS 服务器列表中的第一个条目。
   c. 关闭 文件。

断开 VPN 连接后，必须将更改还原为 `/etc/resolv.conf`。 为此，您可以执行以下步骤：

Bash

```
sudo mv /etc/resolv.conf /etc/resolv.conf.bak
sudo ln -s /run/resolvconf/resolv.conf /etc/resolv.conf
```

WSL 的全局安全访问客户端问题

全局安全访问客户端（/entra/global-secure-access/how-to-install-windows-client）可能会影响 WSL 连接，因为它具有在解析名称时返回临时地址的功能。 然后在建立网络连接时将地址替换为实际地址。 这可能会破坏 WSL，因为 WSL 流量被转发到大量 GSA 客户端挂钩的以下层。

建议禁用 DNS 隧道（`dnsTunneling=false`）或禁用镜像模式（`networkingMode=nat`）。

Cisco Anyconnect VPN 在 WSL 的 NAT 模式下存在问题

Cisco AnyConnect VPN 以阻止 NAT 工作的方式修改路由。 有一种特定于 WSL 2 的解决方法：请参阅 Cisco AnyConnect 安全移动性客户端管理员指南（版本 4.10 - AnyConnect 疑难解答）。

镜像网络模式处于打开状态时 VPN 的 WSL 连接问题

镜像网络模式目前是 WSL 配置中的实验性设置。 WSL 的传统 NAT 网络体系结构可以更新为名为“镜像网络模式”的全新网络模式。 当实验`networkingMode`设置为`mirrored`后，Windows 上已有的网络接口将被映射到 Linux 以提高兼容性。 在命令行博客中了解详细信息： WSL 2023 年 9 月更新。

已测试并确认某些 VPN 与 WSL 不兼容，包括：

- “Bitdefender”版本 26.0.2.1
- “OpenVPN”版本 2.6.501
- “Mcafee Safe Connect”版本 2.16.1.124

在 WSL 中对 HttpProxy 镜像使用 autoProxy 时的注意事项

可以在`autoProxy`中使用 设置来配置 HTTP/S 代理镜像。 应用此设置时，请注意以下注意事项：

- **PAC 代理**：WSL 将通过设置`WSL_PAC_URL`环境变量在 Linux 中配置设置。 默认情况下，Linux 不支持 PAC 代理。
- **与 WSLENV 的交互**：用户定义的环境变量优先于此功能指定的环境变量。

启用后，以下项适用于 Linux 分发版上的代理设置：

- Linux 环境变量 `HTTP_PROXY` 设置为在 Windows HTTP 代理配置中安装的一个或多个 HTTP 代理。
- Linux 环境变量 `HTTPS_PROXY` 设置为 Windows HTTPS 代理配置中安装的一个或多个 HTTPS 代理。
- Linux 环境变量 `NO_PROXY` 设置为绕过在 Windows 配置目标中找到的任何 HTTP/S 代理。
- 每个环境变量（除外 `WSL_PAC_URL`）都设置为小写和大写。 例如：`HTTP_PROXY` 和 `http_proxy`。

ZScaler 配置导致已知问题，其中 ZScaler 重复启用和禁用 Windows 代理配置，导致 WSL 反复显示“主机上检测到 Http 代理更改”通知。

在命令行博客中了解详细信息： WSL 2023 年 9 月更新 。

DNS 隧道中的网络考量因素

当 WSL 无法连接到 Internet 时，可能是因为对 Windows 主机的 DNS 调用被阻止。 这是因为现有网络配置阻止了 WSL VM 发送到 Windows 主机的 DNS 的网络数据包。 DNS 隧道修复了此问题，方法是使用虚拟化功能直接与 Windows 通信，从而允许解析 DNS 名称，而无需发送网络数据包。 此功能应提高网络兼容性，并允许你获得更好的 Internet 连接，即使你拥有 VPN、特定的防火墙设置或其他网络配置。

可以使用 `dnsTunneling` 设置在 WSL 配置文件实验部分中配置 DNS 隧道。 应用此设置时，请注意以下注意事项：

- 如果将 VPN 与 WSL 配合使用，请启用 DNS 隧道。 许多 VPN 使用 NRPT 策略，这些策略仅在启用 DNS 隧道时应用于 WSL DNS 查询。
- `/etc/resolv.conf` Linux 分发中的文件有 3 个 DNS 服务器的最大限制，而 Windows 可能使用 3 个以上的 DNS 服务器。 使用 DNS 隧道可消除此限制 - Linux 现在可以使用所有 Windows DNS 服务器。
- WSL 将按以下顺序使用 Windows DNS 后缀（类似于 Windows DNS 客户端使用的顺序）：

    1. 全局 DNS 后缀
    2. 补充 DNS 后缀
    3. 接口专用 DNS 后缀
    4. 如果在 Windows 上启用了 DNS 加密（DoH、DoT），加密将应用于 WSL 中的 DNS 查询。 如果用户想要在 Linux 中启用 DoH、DoT，则需要禁用 DNS 隧道。

- 来自 Docker Desktop 管理的 Docker 容器的 DNS 查询将绕过 DNS 隧道。 Docker Desktop 采用自己的方式（不同于 DNS 隧道），用于将主机 DNS 设置和策略应用到 Docker 容器中的 DNS 查询。
- 若要成功启用 DNS 隧道，不应禁用 wsl.conf 文件中的 generateResolvConf 选项。

- 启用 DNS 隧道后， `generateHosts` 将忽略 wsl.conf 文件中的选项（Windows DNS 主机文件未在 Linux /etc/hosts 文件中复制）。 Windows 主机文件中的策略将应用于 Linux 中的 DNS 查询，而无需在 Linux 中复制该文件。

在命令行博客中了解详细信息： WSL 2023 年 9 月更新 。

指导 Windows 主机收到的入站流量到 WSL 虚拟机的问题

使用镜像网络模式（实验性功能 `networkingMode` 设置为 `mirrored`），Windows 主机接收到的某些入站流量将不会被传递到 Linux 虚拟机。 此流量如下所示：

- UDP 端口 68 （DHCP）
- TCP 端口 135 （DCE 终结点解析）
- TCP 端口 1900 （UPnP）
- TCP 端口 2869 （SSDP）
- TCP 端口 5004 （RTP）
- TCP 端口 3702 （WSD）
- TCP 端口 5357 （WSD）
- TCP 端口 5358 （WSD）

使用镜像网络模式时，WSL 将自动配置某些 Linux 网络设置。 不支持使用镜像网络模式时这些设置的任何用户配置。 下面是 WSL 将配置的设置列表：

展开表

设置名称	价值
`https://sysctl-explorer.net/net/ipv4/accept_local/`	已启用 （1）
`https://sysctl-explorer.net/net/ipv4/route_localnet/`	已启用 （1）
`https://sysctl-explorer.net/net/ipv4/rp_filter/`	已禁用 (0)
`https://sysctl-explorer.net/net/ipv6/accept_ra/`	已禁用 (0)
`https://sysctl-explorer.net/net/ipv6/autoconf/`	已禁用 (0)
`https://sysctl-explorer.net/net/ipv6/use_tempaddr/`	已禁用 (0)
地址生成模式	已禁用 (0)
disable_ipv6（禁用IPv6）	已禁用 (0)
`https://sysctl-explorer.net/net/ipv4/arp_filter/`	已启用 （1）

在默认网络命名空间下运行时启用镜像网络模式的 WSL2 中的 Docker 容器问题

存在一个已知问题，即无法创建具有已发布端口的 Docker Desktop 容器（`docker run -publish/-p`）。 WSL 团队正在与 Docker 桌面团队合作解决此问题。 若要解决此问题，请使用 Docker 容器中的主机网络命名空间。 通过 `--network host` 命令中使用的 `docker run` 选项设置网络类型。 另一种解决方法是在 `ignoredPorts` 的设置中列出已发布的端口号。

当其网络管理器正在运行时，Docker 容器出现问题

Docker 容器运行网络管理器服务时存在已知问题。 症状包括尝试与主机建立环回连接时失败。 建议停止网络管理器服务，以便正确配置 WSL 网络。

```Bash
sudo systemctl disable network-manager.service
```

解析 WSL 中的 .local 名称

若要将主机名解析为本地网络中的 IP 地址，无需使用传统的 DNS 服务器，通常使用 .local 名称。 这是通过 mDNS（多播 DNS）协议实现的，该协议依赖于多播流量来运行。

`networkingMode` 设置为 `NAT`：

目前，启用 DNS 隧道时不支持此功能。 若要启用 .local 名称的解析，建议使用以下解决方案：

- 禁用 DNS 隧道。
- 使用镜像网络模式。

`networkingMode` 设置为 `Mirrored`：

注意：您需要在 WSL 版本 2.3.17 或更高版本上，以便使用以下功能。

由于镜像模式支持多播流量，因此 mDNS（多播 DNS）协议可用于解析 .local 名称。 Linux 必须配置为支持 mDNS，因为它默认不这样做。 配置它的一种方法是使用以下两个步骤：

1. 安装 `libnss-mdns` 包

   ```Bash
   sudo apt-get install libnss-mdns
   ```

*该 `libnss-mdns` 包是 GNU C 库 （glibc） 的 GNU 名称服务交换机 （NSS） 功能的插件，它通过多播 DNS （mDNS） 提供主机名解析。 此包实际上允许常见的 Unix/Linux 程序解析临时 mDNS 域 .local 中的名称。

2. 配置 `/etc/nsswitch.conf` 文件以在 `mdns_minimal` 部分启用 `hosts` 设置。 文件的示例内容：

```
Bash
/>cat /etc/nsswitch.conf
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the `glibc-doc-reference' and `info' packages installed, try:
# `info libc "Name Service Switch"' for information about this file.

passwd:         compat systemd
group:          compat systemd
shadow:         compat
gshadow:        files

hosts:          files mdns_minimal [NOTFOUND=return] dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

WSL 中的 DNS 后缀

根据 .wslconfig 文件中的配置，WSL 在 DNS 后缀方面将有以下行为：

何时 `networkingMode` 设置为 `NAT`：

案例 1：默认情况下，Linux 中未配置 DNS 后缀

情况 2：如果启用了 DNS 隧道（`dnsTunneling` 在 .wslconfig 中设置为 `true` ），则在 Linux 中将配置所有 Windows DNS 后缀，并在 /etc/resolv.conf 的 `search` 设置中体现。

后缀按以下顺序在 /etc/resolv.conf 中配置，类似于 Windows DNS 客户端在解析名称时尝试后缀的顺序：首先全局 DNS 后缀，然后补充 DNS 后缀，然后按接口 DNS 后缀。

当 Windows DNS 后缀发生更改时，该更改将自动反映在 Linux 中

案例 3：如果禁用了 DNS 隧道，并且禁用了 SharedAccess DNS 代理（`dnsTunneling` 并在 `dnsProxy` .wslconfig 中设置为 `false` ）。 在 Linux 中，在 /etc/resolv.conf 的“域”设置中配置单个 DNS 后缀

当 Windows DNS 后缀发生更改时，该更改不会反映在 Linux 中

Linux 中配置的单个 DNS 后缀是从每个接口 DNS 后缀中选择的（忽略全局和补充后缀）

如果 Windows 具有多个接口，则会使用启发式选择将在 Linux 中配置的单个 DNS 后缀。 例如，如果 Windows 上有 VPN 接口，则从该接口中选择后缀。 如果没有 VPN 接口，则从最有可能提供 Internet 连接的接口中选择后缀。

何时 `networkingMode` 设置为 `Mirrored`：

在 /etc/resolv.conf 的设置中，所有 Windows DNS 后缀都配置在 Linux 中`search`

后缀在 /etc/resolv.conf 中按与 NAT 模式的第2种情况相同的顺序进行配置。

当 Windows DNS 后缀发生更改时，该更改将自动反映在 Linux 中

> ⚠ **注意**
>
> 可以在 Windows 中使用补充 DNS 后缀进行配置。
>
> SetInterfaceDnsSettings 函数（netioapi.h），其参数**中设置了 `DNS_SETTING_SUPPLEMENTAL_SEARCH_LIST` 标志 `Settings`**。

排查 WSL 中的 DNS 问题

WSL 在 NAT 模式下启动容器时的默认 DNS 配置是让 Windows 主机上的 NAT 设备充当 WSL 容器的 DNS“服务器”。 当 DNS 查询从 WSL 容器发送到 Windows 主机上的 NAT 设备时，DNS 数据包将从 NAT 设备转发到主机上的共享访问服务;响应以反向方向发送回 WSL 容器。 为使数据包转发过程到共享访问正常进行，防火墙规则需要允许入站的 DNS 数据包。该数据包是在 WSL 最初请求 HNS 为其 WSL 容器创建 NAT 虚拟网络时，由 HNS 服务创建的。

由于这种 NAT - 共享访问设计，有几个已知的配置可能会破坏 WSL 的名称解析。

1. **企业可以推送不允许本地定义的防火墙规则的策略，只允许企业策略定义的规则。**

   如果这是企业设置的，则忽略 HNS 创建的防火墙规则，因为它是本地定义的规则。

   若要使此配置正常工作，企业必须创建防火墙规则，以允许 UDP 端口 53 到共享访问服务，或者 WSL 可以设置为使用 DNS 隧道。

可以通过运行以下内容来查看这是否配置为不允许本地定义的防火墙规则。 请注意，这将显示所有 3 个配置文件的设置：域、专用和公用。 如果已在任何配置文件上设置，则在为 WSL vNIC 分配该配置文件（默认值为 `Public`）时，将阻止数据包。 这只是 Powershell 中返回的第一个防火墙配置文件的代码片段：

PowerShell

```
PS C:\> Get-NetFirewallProfile -PolicyStore ActiveStore
Name                  : Domain
Enabled               : True
DefaultInboundAction  : Block
DefaultOutboundAction : Allow
AllowInboundRules     : True
AllowLocalFirewallRules : False
...
```

`AllowLocalFirewallRules: False` 表示不会应用或使用本地定义的防火墙规则，如 HNS 规则。

2. **企业可以向下推送阻止所有入站规则的组策略和 MDM 策略设置。**

   这些设置将优先于任何 Allow-Inbound 防火墙规则。 因此，此设置将阻止 HNS 创建的 UDP 防火墙规则，从而阻止 WSL 解析名称。

   要使此配置正常工作， **必须将 WSL 设置为使用 DNS 隧道。** 此设置将始终阻止 NAT DNS 代理。

   从组策略：

   计算机配置 \ 管理模板 \ 网络 \ 网络连接 \ Windows Defender 防火墙 \ 域配置文件 |标准配置文件

   “Windows Defender 防火墙：不允许例外”- 已启用

   MDM 策略来源：

   ./Vendor/MSFT/Firewall/MdmStore/PrivateProfile/Shielded

   ./Vendor/MSFT/Firewall/MdmStore/DomainProfile/Shielded

   ./Vendor/MSFT/Firewall/MdmStore/PublicProfile/Shielded

   可以通过运行以下命令来查看是否已配置为不允许任何入站防火墙规则（请参阅上述防火墙配置文件的注意事项）。 这只是 Powershell 中返回的第一个防火墙配置文件的代码片段：

   powerShell

```
PS C:\> Get-NetFirewallProfile -PolicyStore ActiveStore
Name                  : Domain
Enabled               : True
DefaultInboundAction  : Block
DefaultOutboundAction : Allow
AllowInboundRules     : False
...
```

`AllowInboundRules: False` 表示不会应用任何入站防火墙规则。

3. **用户在 Windows 安全设置应用中检查控件，查看是否“阻止所有传入连接，包括在允许的应用列表中的连接”。**

   Windows 支持用户选择加入上述 #2 中引用的企业可应用的相同设置。 用户可以打开“Windows 安全性”设置页，选择“防火墙和网络保护”选项，选择要配置的防火墙配置文件（域、专用或公共），并在“传入连接”下检查标记为“阻止所有传入连接（包括允许的应用列表中的连接）”的控制。

   如果为公共配置文件设置了此设置（这是 WSL vNIC 的默认配置文件），则 HNS 创建的防火墙规则将阻止 UDP 数据包进行共享访问。

   为让 NAT DNS 代理配置在使用 WSL 时有效，必须取消选中此项，**或者将 WSL 设置为使用 DNS 隧道。**

4. **允许 DNS 数据包共享访问的 HNS 防火墙规则可能变得无效，引用以前的 WSL 接口标识符。**

   这是 HNS 中已修复的最新 Windows 11 版本的缺陷。 在早期版本中，如果发生这种情况，则不容易发现，但它有一个简单的解决方法：

   - 停止 WSL

     PowerShell
     ```
     wsl.exe –shutdown
     ```

   - 删除旧的 HNS 防火墙规则。 在大多数情况下，此 Powershell 命令应正常工作：

     PowerShell
     ```
     Get-NetFirewallRule -Name "HNS*" | Get-NetFirewallPortFilter | where Protocol -eq UDP | where LocalPort -eq 53 | Remove-NetFirewallRule
     ```

   - 删除所有 HNS 终结点。 注意：如果使用 HNS 管理其他容器（如 MDAG 或 Windows 沙盒），则还应停止这些容器。

PowerShell

```
hnsdiag.exe delete all
```

- 重新启动或重启 HNS 服务

PowerShell

```
Restart-Service hns
```

- 重启 WSL 后，HNS 将创建新的防火墙规则，正确针对 WSL 接口。

排查 Windows 上的网络访问问题

如果没有网络访问权限，则可能是因为配置错误。 请查看 FSE 驱动程序是否正在运行：

PowerShell

```
Get-Service FSE
```

如果未显示 FSE 正在运行，请检查此注册表项下的注册表值 `PortTrackerEnabledMode` 是否存在：

PowerShell

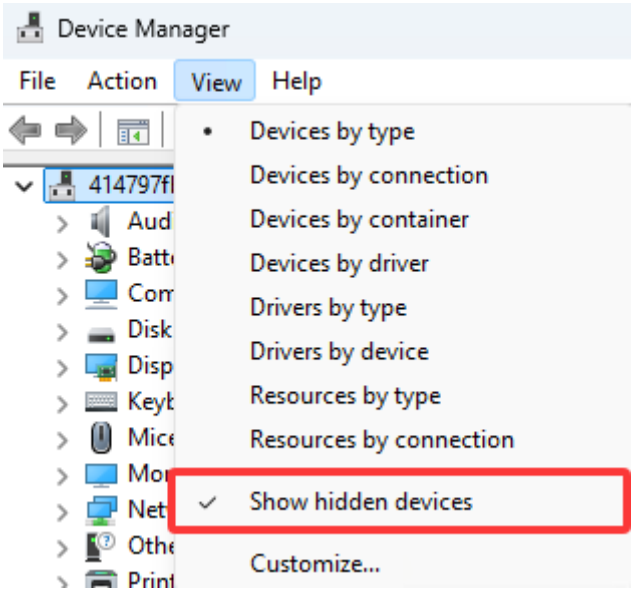
```
Get-ItemProperty HKLM:\System\CurrentControlSet\Services\Tcpip\Parameters
```

如果 FSE 未运行或安装，并且 `PortTrackerEnabledMode` 存在，请删除该注册表值并重新启动

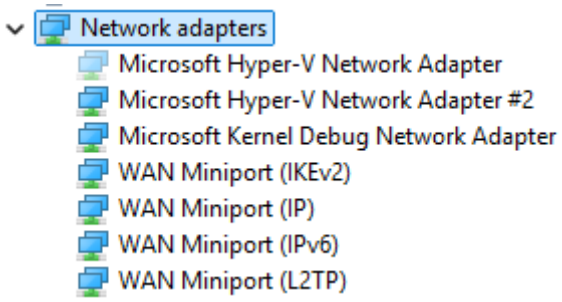
手动删除虚拟适配器的方法

虚影适配器，或虚拟即插即用（PnP）设备，指显示在系统中但未物理连接的硬件组件。 这些“幽灵”设备可能会导致系统设置中的混乱和杂乱。 如果在虚拟机（VM）中运行 WSL 时看到虚影适配器，请按照以下手动步骤查找和删除这些虚拟 PnP 设备。 Microsoft正在处理不需要手动干预的自动化解决方案。 更多信息即将推出。

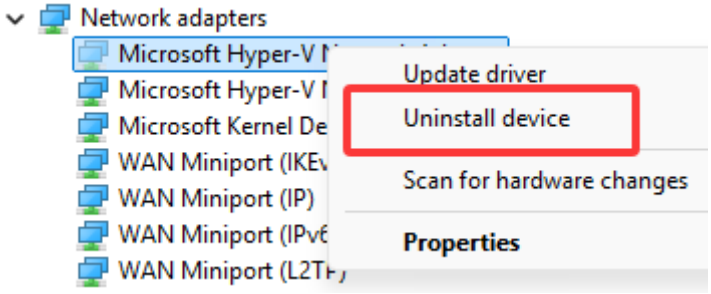
1. 打开“设备管理器”
2. 查看 > 显示隐藏的设备



3. 打开网络适配器



4. 右键单击虚影网络适配器并选择“ **卸载设备**”



启动 WSL 或安装分发版将返回错误代码

按照说明在 GitHub 上的 WSL 存储库中收集 WSL 日志 ，以收集详细的日志并在 GitHub 上提出问题。

更新 WSL

适用于 Linux 的 Windows 子系统有两个组件，可能需要更新。

1. 若要更新适用于 Linux 的 Windows 子系统本身，请使用以下命令。

PowerShell

```
wsl.exe --update
```

2. 若要更新特定的 Linux 分发用户二进制文件，请在要更新的 Linux 分发版中使用以下命令。

Bash

```
apt-get update | apt-get upgrade
```

通过 Apt-get 升级时发生的错误

某些包使用尚未实现的功能。 例如，`udev` 目前尚不支持，并导致多个 `apt-get upgrade` 错误。

若要修复与 `udev` 此问题相关的问题，请执行以下步骤：

1. 将以下内容 `/usr/sbin/policy-rc.d` 写入并保存所做的更改。

Bash

```
#!/bin/sh
exit 101
```

2. 将执行权限添加到 `/usr/sbin/policy-rc.d`：

Bash

```
chmod +x /usr/sbin/policy-rc.d
```

3. 运行以下命令：

Bash

```
dpkg-divert --local --rename --add /sbin/initctl
ln -s /bin/true /sbin/initctl
```

安装时出现“错误：0x80040306”

这与我们不支持旧版控制台这一事实有关。 若要关闭旧版控制台，请执行以下作：

1. 打开 cmd.exe
2. 右键单击标题栏 - 属性 ->> 取消选中“使用旧版控制台”
3. 单击“确定”

Windows 更新后出现“错误：0x80040154”

Windows Update 期间可能会禁用适用于 Linux 的 Windows 子系统功能。 如果发生这种情况，则必须重新启用 Windows 功能。 有关启用适用于 Linux 的 Windows 子系统的说明，请参阅 手动安装指南。

更改显示语言

WSL 安装将尝试自动更改 Ubuntu 区域设置以匹配 Windows 安装的区域设置。 如果不希望此行为，可以在安装完成后运行此命令来更改 Ubuntu 区域设置。 必须重新启动 bash.exe 才能使此更改生效。

以下示例将区域设置更改为`en-US`：

```bash
sudo update-locale LANG=en_US.UTF8
```

Windows 系统还原后的安装问题

1. 删除 `%SystemRoot%\System32\Tasks\Microsoft\Windows\Windows Subsystem for Linux` 文件夹。

   > ⚠ **警告**
   >
   > 如果可选功能已完全安装且正常工作，请不要执行此作。

2. 启用 WSL 可选功能（如果尚未启用）
3. 重新启动
4. 此命令用于完全卸载 lxrun 环境：lxrun /uninstall /full
5. 安装 bash

WSL 中无法连接到互联网

某些用户报告了特定防火墙应用程序阻止 WSL 中的 Internet 访问的问题。 报告的防火墙包括：

1. 卡巴斯基
2. AVG
3. Avast

4. Symantec Endpoint Protection

在某些情况下，关闭防火墙允许访问。 在某些情况下，只需安装防火墙即可阻止访问。

如果你正在使用 Microsoft Defender 防火墙，取消选中“阻止所有传入连接，包括允许的应用列表中的连接”选项，即可允许访问。

使用 ping 时出现权限被拒绝的错误

对于 Windows 周年更新版本 1607，在 WSL 中运行 需要 Windows 的 `ping`。 若要运行 `ping`，请以管理员身份在 Windows 上的 Ubuntu 上运行 Bash，或者从具有管理员权限的 PowerShell 提示符运行 `bash.exe` 。

对于更高版本的 Windows（ 内部版本 14926+），不再需要管理员权限。

Bash 挂起

如果在使用 bash 时，你发现 bash 卡住（或发生死锁），并且不响应输入，请通过收集并报告内存转储来帮助我们诊断问题。 请注意，这些步骤会使系统崩溃。 如果你对此不感到舒服，或在这样做之前没有保存你的工作，请不要这样做。

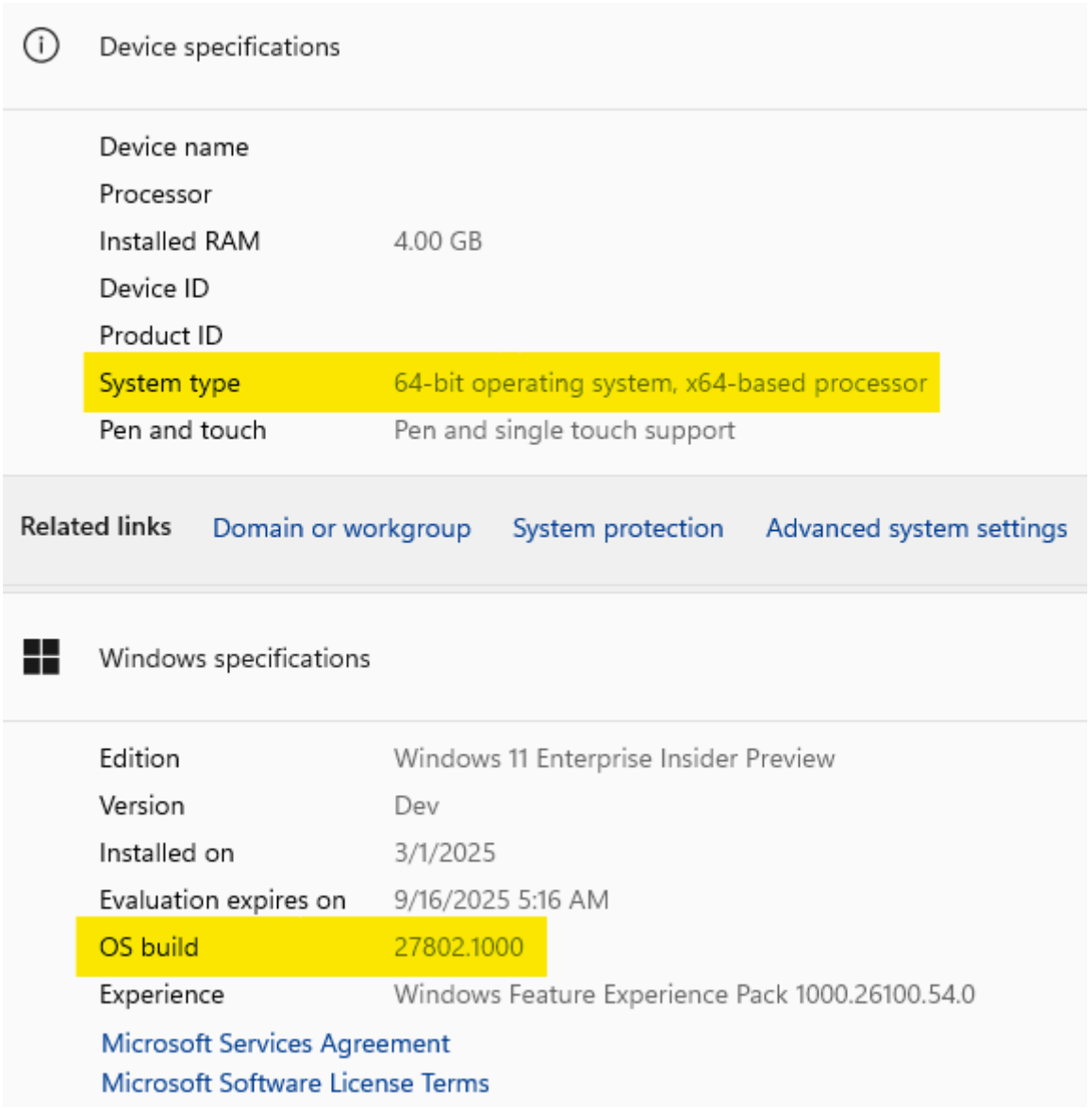
收集内存转储：

1. 将内存转储类型更改为“完整内存转储”。 更改转储类型时，请记下当前类型。
2. 使用 步骤 通过键盘控件设置崩溃管理。
3. 重新设置挂起或死锁。
4. 用按键序列（2）中的键使系统崩溃。
5. 系统将崩溃并收集内存转储。
6. 系统重新启动后，请将 `memory.dmp` 报告给 secure@microsoft.com。 转储文件的默认位置是 `%SystemRoot%\memory.dmp`，如果 `C:\Windows\memory.dmp` 是系统驱动器，则为 `C:`。 在电子邮件中，请注明该转储是给 WSL 或 Bash on Windows 团队的。
7. 将内存转储类型还原到原始设置。

检查版本号

若要查找电脑的体系结构和 Windows 内部版本号，请打开“**设置**>>”

查找 “OS **生成** ”和 “**系统类型**” 字段。



若要查找 Windows Server 内部版本号，请在 PowerShell 中运行以下命令：

PowerShell

```
systeminfo | Select-String "^OS Name","^OS Version"
```

确认已启用 WSL

可以通过在提升的 PowerShell 窗口中运行以下命令来确认已启用适用于 Linux 的 Windows 子系统：

PowerShell

```
Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

OpenSSH-Server 连接问题

尝试连接 SSH 服务器时失败，出现以下错误：“连接被 127.0.0.1 端口 22 关闭”。

1. 确保 OpenSSH 服务器正在运行：

   ```Bash
   sudo service ssh status
   ```

   你已遵循本教程进行操作：https://ubuntu.com/server/docs/service-openssh

2. 停止 sshd 服务并在调试模式下启动 sshd：

   ```Bash
   sudo service ssh stop
   sudo /usr/sbin/sshd -d
   ```

3. 检查启动日志并确保 HostKey 可用，并且看不到日志消息，例如：

   ```Bash
   debug1: sshd version OpenSSH_7.2, OpenSSL 1.0.2g  1 Mar 2016
   debug1: key_load_private: incorrect passphrase supplied to decrypt private key
   debug1: key_load_public: No such file or directory
   Could not load host key: /etc/ssh/ssh_host_rsa_key
   debug1: key_load_private: No such file or directory
   debug1: key_load_public: No such file or directory
   Could not load host key: /etc/ssh/ssh_host_dsa_key
   debug1: key_load_private: No such file or directory
   debug1: key_load_public: No such file or directory
   Could not load host key: /etc/ssh/ssh_host_ecdsa_key
   debug1: key_load_private: No such file or directory
   debug1: key_load_public: No such file or directory
   Could not load host key: /etc/ssh/ssh_host_ed25519_key
   ```

如果看到此类消息且`/etc/ssh/`下的密钥缺失，则您必须重新生成密钥，或者仅清除并安装 openssh-server。

```BASH
sudo apt-get purge openssh-server
sudo apt-get install openssh-server
```

“找不到引用的程序集。”启用 WSL 可选功能时

此错误与处于错误安装状态有关。 请完成以下步骤来尝试并解决此问题：

- 如果您在 PowerShell 中运行启用 WSL 功能的命令，可以尝试使用图形界面。在开始菜单中搜索“打开或关闭 Windows 功能”，然后在列表中选择“适用于 Linux 的 Windows 子系统”，以进行可选组件的安装。
- 转到“设置”、“更新”并单击“检查更新”来更新 Windows 版本
- 如果这两个都失败，并且你需要访问 WSL，请考虑使用安装媒体重新安装 Windows，然后选择“保留所有内容”以确保你的应用和文件保持原样。 可以在 “重新安装 Windows 10”页面上 找到有关如何执行此作的说明。

更正（与 SSH 相关的）权限错误

如果看到此错误：

Bash

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@         WARNING: UNPROTECTED PRIVATE KEY FILE!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0777 for '/home/user/.ssh/private-key.pem' are too open.
```

若要解决此问题，请将以下内容追加到 `/etc/wsl.conf` 该文件：

Bash

```
[automount]
enabled = true
options = metadata,uid=1000,gid=1000,umask=0022
```

请注意，添加此命令将包括元数据，并修改从 WSL 看到的 Windows 文件的文件权限。 有关详细信息，请参阅 文件系统权限 。

无法通过 Windows 上的 OpenSSH 远程使用 WSL

如果在 Windows 上使用 openssh-server 并尝试远程访问 WSL，则许多人会看到此错误：系统无法访问该文件。

使用 WSL 的应用商店版本时，这是一个 已知问题。 可以使用 WSL 1 或 Windows 中的 WSL 版本解决此问题。 有关详细信息，请参阅 Microsoft Store 中的 WSL 。

运行 Windows 命令在分发版中失败

某些Microsoft应用商店中提供的分发版尚未开箱即用地运行 Windows 命令。 如果您在运行 `powershell.exe /c start .` 或任何其他 Windows 命令时遇到“-bash：powershell.exe：找不到

命令”错误，可以按照以下步骤解决此问题：

1. 在 WSL 分发版中运行 `echo $PATH`。 如果不包含：`/mnt/c/Windows/system32`，那么某些东西正在重新定义标准 PATH 变量。
2. 使用`cat /etc/profile`检查配置文件设置。 如果它包含 PATH 变量的赋值，请编辑该文件以注释掉带 # 字符的 PATH 赋值块。
3. 检查 wsl.conf 是否存在 `cat /etc/wsl.conf` 并确保它不包含 `appendWindowsPath=false`，否则将其注释掉。
4. 通过键入 `wsl -t <Distro>` 后跟分发名称或在 PowerShell 中运行 `wsl --shutdown` 来重启分发。

安装 WSL 2 后无法启动

我们注意到了影响用户在安装 WSL 2 后无法启动的问题。 在我们对这些问题进行全面诊断的同时，用户报告称，更改缓冲区大小或安装正确的驱动程序可以帮助解决此问题。 请查看此 GitHub 问题 ，查看此问题的最新更新。

禁用 ICS 时出现 WSL 2 错误

Internet 连接共享（ICS）是 WSL 2 的必需组件。 主机网络服务 （HNS） 使用 ICS 服务创建 WSL 2 依赖的 NAT、DNS、DHCP 和主机连接共享的基础虚拟网络。

禁用 ICS 服务（SharedAccess）或通过组策略禁用 ICS 将阻止创建 WSL HNS 网络。 这将导致创建新的 WSL 版本 2 映像时失败，尝试将版本 1 映像转换为版本 2 时出现以下错误：终结点映射器中没有其他可用的终结点。

需要 WSL 2 的系统应将 ICS 服务（SharedAccess）保留为默认启动状态、手动（触发器启动）以及禁用 ICS 的任何策略都应被覆盖或删除。 禁用 ICS 服务会导致 WSL 2 出现问题，因此我们不建议禁用 ICS。不过，您可以使用这些说明来禁用 ICS 的某些功能。

使用较旧版本的 Windows 和 WSL

如果运行的是较旧版本的 Windows 和 WSL，如 Windows 10 创意者更新（2017 年 10 月版本 16299）或周年更新（2016 年 8 月版本 14393），则有几个区别需要注意。 我们建议 你更新到最新的 Windows 版本，但如果这是不可能的，我们概述了以下一些差异。

互作性命令差异：

- `bash.exe` 已替换为 `wsl.exe`。 Linux 命令可以从 PowerShell 运行，但对于早期 Windows 版本，可能需要使用此命令 `bash` 。 例如： `C:\temp> bash -c "ls -la"`。 传入 `bash -c` 的 WSL 命令将转发到 WSL 进程，而无需修改。 文件路径必须以 WSL 格式指定，并且需

要仔细注意转义相关字符。 例如：`C:\temp> bash -c "ls -la /proc/cpuinfo"` 或 `C:\temp> bash -c "ls -la \"/mnt/c/Program Files\""`。

- 若要查看哪些命令可用于特定分发版，请运行 `[distro.exe] /?`。 例如，使用 Ubuntu：`C:\> ubuntu.exe /?`。
- Windows 路径包含在 WSL `$PATH`中。
- 在早期版本的 Windows 10 中从 WSL 分发版调用 Windows 工具时，需要指定目录路径。 例如，若要从 WSL 命令行调用 Windows 记事本应用，请输入：`/mnt/c/Windows/System32/notepad.exe`
- 若要更改默认用户以 `root` 在 PowerShell 中使用此命令： `C:\> lxrun /setdefaultuser root` 然后运行 Bash.exe 以登录： `C:\> bash.exe`。 使用分发密码命令重置密码： `$ passwd username` 然后关闭 Linux 命令行： `$ exit`。 在 Windows 命令提示符或 Powershell 中，将默认用户重置回正常的 Linux 用户帐户： `C:\> lxrun.exe /setdefaultuser username`

卸载旧版 WSL

如果最初在创意者更新之前的 Windows 10 版本（2017 年 10 月内部版本 16299）上安装 WSL，建议将任何必要的文件、数据等从安装的较旧的 Linux 分发版迁移到通过 Microsoft 应用商店安装的较新的分发版。 若要从计算机中删除旧分发版，请从命令行或 PowerShell 实例运行以下命令： `wsl --unregister Legacy` 您还可以选择使用 Windows 文件资源管理器或 PowerShell，手动删除旧版遗留发行版的 `%LocalAppData%\lxss\` 文件夹及其所有子内容。 `Remove-Item -Recurse $env:localappdata/lxss/`

错误代码0x8000FFFF意外失败

此错误代码通常意味着在尝试安装或使用带有 WSL 的 Linux 分离（如 Ubuntu）时，系统作期间出现意外或“灾难性”故障。 有许多原因可能导致此失败。 首先检查以下内容：

- 这是权限问题吗？ 请检查您是否以预期的用户身份登录到命令行，并在通过 WSL 安装 Linux 发行版时拥有必要的管理员权限。 （右键单击终端或命令行任务栏图标以选择“以管理员身份运行”。
- 是否已将 WSL 更新到最新版本？ 使用命令： `wsl --update` 更新到最新版本。 你可能还需要 更新到最新版本的 Windows。
- 确保正确使用 wsl --install 命令并指定要安装的 Linux 发行版。
- 尝试使用以下命令关闭并重启 WSL。 `wsl --shutdown`
- 如果你使用的是 Windows Server，请确保你的版本是最新的，并遵循 Windows Server 安装指南。
- 如果您怀疑这可能与系统文件丢失或损坏有关，请以管理员权限运行命令提示符，然后您可以扫描并修复系统文件，和/或修复 Windows 操作系统映像。 若要扫描和修复损坏或缺

少的 Windows 系统文件，请使用以下命令： `SFC /SCANNOW` 若要修复 Windows 映像本身，请使用以下命令： `DISM /Online /Cleanup-Image /RestoreHealth`

- 请参阅相关的 WSL 产品存储库问题 9420 。

Last updated on 2025/10/03

适用于 Linux 的 Windows 子系统发行说明

版本 21364

关于版本 21364 的一般 Windows 信息，请访问 Windows 博客。

- GUI 应用现已推出！ 有关详细信息，请参阅 此博客文章。
- 解决通过 \\wsl.localhost\ 访问文件时出现错误。
- 修复 LxssManager 服务中潜在的死锁。

版本 21354

有关版本 21354 的一般 Windows 信息，请访问 Windows 博客。

- 将 \wsl 前缀切换到 \wsl.localhost，以避免在网络上有名为“wsl”的计算机时出现问题。 \wsl$ 将继续工作。
- 为“WoW”进程启用 Linux 快速访问图标。
- 更新总是通过 wslapi RegisterDistribution 传递版本 2 的问题。
- 将 /usr/lib/wsl/lib 目录的 fmask 更改为 222，以便将文件标记为可执行文件 [GH 3847]
- 修复虚拟机平台未启用时导致的 WSL 服务崩溃。

内部版本 21286

有关 Windows 版本 21286 的常规信息，请访问 Windows 博客。

- 引入 wsl.exe --cd 命令以设置命令的当前工作目录。
- 改进 NTSTATUS 到 Linux 错误代码的映射。 [GH 6063]
- 改进 wsl.exe --装载错误报告。
- 向 /etc/wsl.conf 添加了用于启用启动命令的选项：

控制台

```
[boot]
command=<string>
```

版本 20226

有关构建 20226 的通用 Windows 信息，请访问 Windows 博客。

- 修复 LxssManager 服务中的崩溃问题。 [GH 5902]

内部版本 20211

有关 Windows 内部版本 20211 的通用信息，请访问 Windows 博客。

- 用于装载物理或虚拟磁盘的 `wsl.exe --mount` 简介。 有关详细信息，请参阅 在 Windows 和 WSL 2 中访问 Linux 文件系统。
- 修复 LxssManager 服务在检查 VM 是否空闲时的崩溃。 [GH 5768]
- 支持压缩的 VHD 文件。 [GH 4103]
- 确保在 OS 升级中保留安装到 c：\windows\system32\lxss\lib 的 Linux 用户模式库。 [GH 5848]
- 添加了列出随 `wsl --install --list-distributions` 一起安装的可用分发版的功能。
- 当用户注销时，WSL 实例现在将终止。

内部版本 20190

有关 Windows 版本 20190 的一般信息，请访问 Windows 博客。

- 修复了阻止 WSL1 实例启动的 bug。 [GH 5633]
- 修复重定向 Windows 进程输出时出现的挂起。 [GH 5648]
- 添加 %userprofile%\.wslconfig 选项来控制 VM 空闲超时（wsl2.vmIdleTimeout=<time_in_ms>）。
- 支持从 WSL 启动应用执行别名。
- 添加了对安装 WSL2 内核和分发版到 wsl.exe --install 的支持。

版本 20175

有关 Windows 版本 20175 的一般信息，请访问 Windows 博客。

- 将 WSL2 VM 的默认内存分配调整为主机内存的 50% 或 8GB（以较小者为准）[GH 4166]。
- 将 \\wsl$ 前缀更改为 \\wsl 以支持 URI 分析。 仍支持旧的 \\wsl$ 路径。
- 默认情况下，在 amd64 上为 WSL2 启用嵌套虚拟化。 可以通过 %userprofile%\.wslconfig（[wsl2] nestedVirtualization=false）禁用此功能。
- 执行 wsl.exe --update 命令以启动 Microsoft 更新。
- 支持在 DrvFs 中对只读文件进行重命名。
- 确保始终在正确的代码页中打印错误消息。

内部版本 20150

有关内部版本 20150 的一般 Windows 信息，请访问 Windows 博客。

- 有关 WSL2 GPU 计算，请参阅 Windows 博客 以了解详细信息。
- 引入 wsl.exe --install 命令行选项以轻松设置 WSL。
- 引入 wsl.exe --update 命令行选项来管理 WSL2 内核的更新。
- 将 WSL2 设置为默认值。
- 增加 WSL2 VM 正常关闭超时。
- 修复映射设备内存 virtio-9p 争用情况。
- 如果禁用了 UAC，请勿运行提升的 9p 服务器。

版本 19640

有关内部版本 19640 的一般 Windows 信息，请访问 Windows 博客 。

- [WSL2] virtio-9p (drvfs) 的稳定性改进。

内部版本 19555

有关内部版本 19555 的一般 Windows 信息，请访问 Windows 博客 。

- [WSL2] 使用 memory cgroup 限制了安装和转换操作使用的内存量 [GH 4669]
- 如果未启用适用于 Linux 的 Windows 子系统可选组件，则应使 wsl.exe 可见以提高功能的可发现性。
- 如果未安装 WSL 可选组件，请更改 wsl.exe 以打印帮助文本
- 修复了创建实例时的争用条件
- 创建包含所有命令行功能的 wslclient.dll
- 在 LxssManagerUser 服务停止期间防止崩溃
- 修复当 distroName 参数为 NULL 时 wslapi.dll 快速失败的问题。

内部版本 19041

有关 Windows 版本 19041 的常规信息，请访问 Windows 博客 。

- [WSL2]在启动进程之前清除信号掩码
- [WSL2]将 Linux 内核更新为 4.19.84
- 当 symlink 非相关时，处理 /etc/resolv.conf symlink 的创建

内部版本 19028

有关内部版本 19028 的一般 Windows 信息，请访问 Windows 博客 。

- [WSL2]将 Linux 内核更新为 4.19.81
- [WSL2]将 /dev/net/tun 的默认权限更改为 0666 [GH 4629]

- [WSL2]将分配给 Linux VM 的默认内存量调整为 80% 主机内存
- [WSL2] 修复互操作服务器以便使用“超时”功能处理请求，从而使不良调用方无法挂起服务器

内部版本 19018

有关版本 19018 的一般 Windows 信息，请访问 Windows 博客。

- [WSL2] 使用 cache=mmap 作为 9p 装入点的默认值来修复 dotnet 应用
- [WSL2] localhost 中继的修补程序 [GH 4340]
- [WSL2] 引入了用于在发行版之间共享状态的跨发行版共享 tmpfs 装入点
- 修复了 \\wsl$ 的永久网络驱动器还原

内部版本 19013

有关版本 19013 的 Windows 详细信息，请访问 Windows 博客。

- [WSL2]提高 WSL 实用工具 VM 的内存性能。 不再使用的内存将释放回主机。
- [WSL2]将内核版本更新为 4.19.79。 （添加 CONFIG_HIGH_RES_TIMERS、CONFIG_TASK_XACCT、CONFIG_TASK_IO_ACCOUNTING、CONFIG_SCHED_HRTICK 和 CONFIG_BRIDGE_VLAN_FILTERING）。
- [WSL2] 修复了输入中继，以处理 stdin 为未关闭管道句柄的情况 [GH 4424]
- 检查 \\wsl$ 是否不区分大小写。

控制台

```
[wsl2]
pageReporting = <bool>    # Enable or disable the free memory page reporting feature
(default true).
idleThreshold = <integer> # Set the idle threshold for memory compaction, 0 disables
the feature (default 1).
```

版本 19002

有关版本 19002 的一般 Windows 信息，请访问 Windows 博客。

- [WSL]修复了处理某些 Unicode 字符的问题：https://github.com/microsoft/terminal/issues/2770
- [WSL] 解决了在版本到版本升级后立即启动时可能会注销发行版的罕见情况。
- [WSL] 解决了 wsl.exe --shutdown 的以下小问题：无法取消实例空闲计时器。

内部版本 18995

有关版本 18995 的常规 Windows 信息，请访问 Windows 博客。

- [WSL2] 修复了 DrvFs 装载在某项操作被中断（例如 ctrl-c）后失效的问题 [GH 4377]
- [WSL2] 修复了处理极大型 hvsocket 消息的问题 [GH 4105]
- [WSL2] 修复了当 stdin 为文件时互操作出现的问题 [GH 4475]
- [WSL2] 修复了当遇到意外网络状态时服务崩溃的问题 [GH 4474]
- [WSL2]如果当前进程没有环境变量，请从互作服务器查询发行版名称
- [WSL2] 修复了当 stdin 为文件时互操作出现的问题
- [WSL2]将 Linux 内核版本更新为 4.19.72
- [WSL2]添加通过 .wslconfig 指定其他内核命令行参数的功能

```
[wsl2]
kernelCommandLine = <string> # Additional kernel command line arguments
```

版本 18990

有关版本 18990 的一般 Windows 信息，请访问 Windows 博客。

- 提高 \\wsl$ 中目录列表的性能
- [WSL2] 注入额外的启动熵 [GH 4416]
- [WSL2] 修复使用 su/sudo 时的 Windows 互操作 [GH 4465]

内部版本 18980

有关 Windows 版本 18980 的一般信息，请访问 Windows 博客。

- 修复拒绝 FILE_READ_DATA 的读取符号链接。 这包括 Windows 创建的所有符号链接以实现向后兼容性，例如“C：\Document and Settings”和用户配置文件目录中的一堆符号链接
- 使意外的文件系统状态非致命 [GH 4334， 4305]
- [WSL2]如果 CPU/固件支持虚拟化，请添加对 arm64 的支持
- [WSL2]允许未特权用户查看内核日志
- [WSL2] 修复关闭 stdout/stderr 套接字后的输出中继 [GH 4375]
- [WSL2] 支持电池和交流适配器直通
- [WSL2]将 Linux 内核更新为 4.19.67
- 添加在 /etc/wsl.conf 中设置默认用户名的功能：

```
[user]
default=<string>
```

内部版本 18975

有关内部版本 18975 的一般 Windows 信息，请访问 Windows 博客。

- [WSL2]修复了许多 localhost 可靠性问题 [GH 4340]

内部版本 18970

有关版本 18970 的 Windows 常规信息，请访问 Windows 博客。

- [WSL2] 当系统从睡眠状态恢复时，使时间与主机时间同步 [GH 4245]
- [WSL2] 在可能的情况下，在 Windows 卷上创建 NT 符号链接。
- [WSL2] 在 UTS、IPC、PID 和挂载命名空间中创建 Linux 发行版。
- [WSL2] 修复服务器直接绑定到 localhost 时的本地端口转发问题 [GH 4353]
- [WSL2] 修复重定向输出时的 interop [GH 4337]
- [WSL2] 支持转换绝对 NT 符号链接。
- [WSL2]将内核更新到 4.19.59
- [WSL2]正确设置 eth0 的子网掩码。
- [WSL2] 发出退出事件信号时更改逻辑，以中断控制台工作线程循环。
- [WSL2] 发行版未运行时弹出发行版 VHD。
- [WSL2]修复配置分析库以正确处理空值。
- [WSL2] 通过创建跨发行版装入点来支持 Docker Desktop。 发行版可以通过将以下行添加到 /etc/wsl.conf 文件来选择加入此行为：

```
[automount]
crossDistro = true
```

内部版本 18945

有关 Windows 内部版本 18945 的一般信息，请访问 Windows 博客。

WSL

- [WSL2] 允许侦听可使用 localhost:port 通过主机访问的 WSL2 中的 TCP 套接字
- [WSL2] 修复了安装/转换失败的情况，并提供额外的诊断工具以跟踪将来的问题 [GH 4105]

- [WSL2]提高 WSL2 网络问题的可诊断性
- [WSL2]将内核版本更新为 4.19.55
- [WSL2] 使用 Docker 所需的配置选项更新了内核 [GH 4165]
- [WSL2]将分配给轻型实用工具 VM 的 CPU 数增加为与主机相同（以前在内核配置中 CONFIG_NR_CPUS限制为 8）[GH 4137]
- [WSL2]为 WSL2 轻型 VM 创建交换文件
- [WSL2] 允许通过 \\wsl$\distro（例如 sshfs）显示用户装入点 [GH 4172]
- [WSL2]提高 9p 文件系统性能
- [WSL2] 确保 VHD ACL 不会无限增长 [GH 4126]
- [WSL2]更新内核配置以支持 squashfs 和 xt_conntrack [GH 4107， 4123]
- [WSL2] 修复 interop.enabled /etc/wsl.conf 选项 [GH 4140]
- [WSL2] 如果文件系统不支持 EA，则返回 ENOTSUP
- [WSL2] 修复 \\wsl$ 时 CopyFile 挂起的问题
- 将默认 umask 切换到 0022，并将 filesystem.umask 设置添加到 /etc/wsl.conf
- 修复 wslpath 以正确解析符号链接，这是19h1 中的回归 [GH 4078]
- 介绍用于调整 WSL2 设置的 %UserProfile%\.wslconfig 文件

```
[wsl2]
kernel=<path>               # An absolute Windows path to a custom Linux kernel.
memory=<size>               # How much memory to assign to the WSL2 VM.
processors=<number>         # How many processors to assign to the WSL2 VM.
swap=<size>                 # How much swap space to add to the WSL2 VM. 0 for no
swap file.
swapFile=<path>             # An absolute Windows path to the swap vhd.
localhostForwarding=<bool> # Boolean specifying if ports bound to wildcard or
localhost in the WSL2 VM should be connectable from the host via localhost:port
(default true).

# <path> entries must be absolute Windows paths with escaped backslashes, for
example C:\\Users\\Ben\\kernel
# <size> entries must be size followed by unit, for example 8GB or 512MB
```

内部版本 18917

有关 Windows 内部版本 18917 的一般信息，请访问 Windows 博客 。

WSL

- WSL 2 现已推出！ 有关详细信息，请参阅 博客 。
- 修复一个回归问题：无法通过符号链接启动 Windows 进程 [GH 3999]
- 将 wsl.exe --list --verbose、wsl.exe --list --quiet 和 wsl.exe --import --version 选项添加到 wsl.exe

- 添加 wsl.exe --shutdown 选项
- 计划 9：允许打开目录以使写入成功

内部版本 18890

有关 Windows 版本 18890 的一般信息，请访问 Windows 博客。

WSL

- 非阻塞套接字泄露 [GH 2913]
- 终端的 EOF 输入可以阻止后续读取 [GH 3421]
- 更新 resolv.conf 标头以引用 wsl.conf [在 GH 3928 中讨论]
- epoll delete 代码中的死锁 [GH 3922]
- 处理 --import 和 --export 参数中的空格 [GH 3932]
- 无法正常扩展 mmap'd 文件 [GH 3939]
- 修复了 ARM64 \\wsl$ 访问不正常的问题
- 为 wsl.exe 添加更好的默认图标

内部版本 18342

有关 Windows 版本 18342 的一般信息，请访问 Windows 博客。

WSL

- 我们添加了用户从 Windows 访问 WSL 发行版中的 Linux 文件的功能。 可以通过命令行访问这些文件，还可以通过 Windows 应用（如文件资源管理器、VSCode 等）与这些文件进行交互。 通过导航到 \\wsl$\<distro_name>来访问文件，或通过导航到 \\wsl$ 查看正在运行的分发版的列表
- 添加其他 CPU 信息标记并修复Cpus_allowed[_list] 值 [GH 2234]
- 支持从非领先线程执行 [GH 3800]
- 将配置更新失败视为非致命 [GH 3785]
- 更新 binfmt 以正确处理偏移 [GH 3768]
- 为 Plan 9 启用映射网络驱动器 [GH 3854]
- 支持对绑定载入点执行“Windows -> Linux”和“Linux -> Windows”路径转换
- 为以只读方式打开的文件中的映射创建只读节

内部版本 18334

有关内部版本 18334 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 重新设计 Windows 时区映射到 Linux 时区的方式 [GH 3747]
- 修复内存泄漏并添加新的字符串转换函数 [GH 3746]
- 不包含任何线程的线程组上的 SIGCONT 是一个 no-op [GH 3741]
- 在 /proc/self/fd 中正确显示套接字和 epoll 文件描述符

内部版本 18305

有关版本 18305 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 当主线程退出 [GH 3589] 时，pthread 将失去对文件的访问权限
- TIOCSCTTY 应忽略“force”参数，除非它是必需的 [GH 3652]
- wsl.exe 命令行改进和添加导入/导出功能。

```
Usage: wsl.exe [Argument] [Options...] [CommandLine]

Arguments to run Linux binaries:

    If no command line is provided, wsl.exe launches the default shell.

    --exec, -e <CommandLine>
        Execute the specified command without using the default Linux shell.

    --
        Pass the remaining command line as is.

Options:
    --distribution, -d <DistributionName>
        Run the specified distribution.

    --user, -u <UserName>
        Run as the specified user.

Arguments to manage Windows Subsystem for Linux:

    --export <DistributionName> <FileName>
        Exports the distribution to a tar file.
        The filename can be - for standard output.

    --import <DistributionName> <InstallLocation> <FileName>
        Imports the specified tar file as a new distribution.
        The filename can be - for standard input.
```

```
--list, -l [Options]
    Lists distributions.

    Options:
        --all
            List all distributions, including distributions that are currently
            being installed or uninstalled.

        --running
            List only distributions that are currently running.

-setdefault, -s <DistributionName>
    Sets the distribution as the default.

--terminate, -t <DistributionName>
    Terminates the distribution.

--unregister <DistributionName>
    Unregisters the distribution.

--upgrade <DistributionName>
    Upgrades the distribution to the WslFs file system format.

--help
    Display usage information.
```

内部版本 18277

有关内部版本 18277 的一般 Windows 信息，请访问 Windows 博客 。

WSL

- 修复内部版本 18272 中引入的“不支持此类接口”错误 [GH 3645]
- 忽略 umount syscall 的 MNT_FORCE 标志 [GH 3605]
- 切换 WSL interop 以使用官方的 CreatePseudoConsole API
- FUTEX_WAIT 重启时不保留超时值

内部版本 18272

若要获取有关 Windows 内部版本 18272 的一般信息，请访问 Windows 博客 。

WSL

- **警告：** 此版本存在一个问题，使得 WSL 无法运行。 尝试启动分发版时，会看到“不支持此类接口”错误。 该问题已修复，下周发布的 Insider Fast 内部版本将会应用修复程序。 如果

您已安装此版本，可以通过“设置->更新 & 安全性->恢复”中的“返回到以前的 Windows 10 版本”功能来回退到以前的 Windows 版本。

版本 18267

有关内部版本 18267 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 修复 zombie 进程不会回收，而是无限期保留的问题。
- 如果错误消息超过最大长度，WslRegisterDistribution 将会挂起 [GH 3592]
- 允许 fsync 针对 DrvFs 上的只读文件成功运行 [GH 3556]
- 在 [GH 3584] 中创建符号链接之前，请确保 /bin 和 /sbin 目录存在
- 为 WSL 实例添加了实例终止超时机制。 超时当前设置为 15 秒，这意味着实例将在最后一个 WSL 进程退出后终止 15 秒。 若要立即终止分发，请使用：

```
wslconfig.exe /terminate <DistributionName>
```

版本 17763 （1809）

有关 Windows 17763 版本的一般信息，请访问 Windows 博客。

WSL

- Setpriority syscall 权限检查过于严格，导致无法更改同一线程的优先级 [GH 1838]
- 确保使用无偏见的中断时间来计算启动时间，以避免clock_gettime(CLOCK_BOOTTIME)返回负值 [GH 3434]。
- 在 WSL binfmt 解释器中处理符号链接 [GH 3424]
- 更好地处理线程组领先者文件描述符清理。
- 切换 WSL 以使用 KeQueryInterruptTimePrecise 而不是 KeQueryPerformanceCounter，以避免溢出 [GH 3252]
- Ptrace attach 可能导致系统调用返回错误值 [GH 1731]
- 修复多个AF_UNIX相关问题 [GH 3371]
- 修复了以下问题：如果当前工作目录长度小于 5 个字符 [GH 3379]，则可能导致 WSL 互作失败
- 避免导致无法与不存在的端口建立环回连接的一秒延迟 [GH 3286]
- 添加 /proc/sys/fs/file-max 存根文件 [GH 2893]
- 更准确的 IPV6 范围信息。

- PR_SET_PTRACER 支持 [GH 3053]
- 管道文件系统意外清除边缘触发的 epoll 事件 [GH 3276]
- 通过 NTFS 符号链接启动的 Win32 可执行文件不尊重符号链接名称 [GH 2909]
- 改善了 zombie 支持 [GH 1353]
- 添加用于控制 Windows 互作行为的 wsl.conf 条目 [GH 1493]

```
  [interop]

  enabled=false # enable launch of Windows binaries; default is true

  appendWindowsPath=false # append Windows path to $PATH variable; default is
true
```

- 修复 getsockname 不是始终返回 UNIX 套接字系列类型的问题 [GH 1774]
- 添加对 TIOCSTI 的支持 [GH 1863]
- 连接进程中的非阻塞套接字应返回写入尝试的 EAGAIN [GH 2846]
- 支持已装载的 VHD 上的 interop [GH 3246、3291]
- 修复根文件夹的权限检查问题 [GH 3304]
- 对 TTY 键盘 ioctl KDGKBTYPE、KDGKBMODE 和 KDSKBMODE 的有限支持。
- 即使在后台启动，Windows UI 应用也应执行。
- 添加 wsl -u 或 --user 选项 [GH 1203]
- 修复启用快速启动时的 WSL 启动问题 [GH 2576]
- Unix 套接字需要保留断开连接的对等凭据 [GH 3183]
- 使用 EAGAIN 时非阻塞 Unix 套接字无限期失败 [GH 3191]
- case=off 是新的默认 drvfs 装入点类型 [GH 2937、3212、3328]
  - 有关详细信息，请参阅 博客。
- 添加 wslconfig /terminate 以停止运行分发。
- 修复了 WSL shell 上下文菜单条目无法正确处理路径中含空格的问题。
- 公开按目录区分大小写作为扩展属性
- ARM64：模拟缓存维护操作。 解决 .NET 问题。
- DrvFs：只取消转义专用范围中与已转义字符对应的字符。
- 修复 ELF 分析程序解释器长度验证中的一位偏移错误 [GH 3154]
- 包含过去时间的 WSL 绝对计时器不会激发 [GH 3091]
- 确保新建的重分析点在父目录中以此类类型列出。
- 以原子方式在 DrvFs 中创建区分大小写的目录。
- 修复了一个附加问题，即即使文件存在，多线程操作也可能返回 ENOENT。 [GH 2712]
- 修复了启用 UMCI 时 WSL 启动失败的问题。 [GH 3020]
- 添加浏览器上下文菜单用于启动 WSL [GH 437、603、1836]。 若要使用此菜单，请在资源管理器窗口中按住 Shift 键的同时单击右键。
- 修复 Unix 套接字非阻塞行为 [GH 2822、3100]

- 修复 GH 2026 中报告的卡住的 NETLINK 命令。
- 添加对装载传播标志的支持 [GH 2911]。
- 修复截断后不会导致 inotify 事件的问题 [GH 2978]。
- 为 wsl.exe 添加 --exec 选项，以在没有 shell 的情况下调用单个二进制文件。
- 为 wsl.exe 添加 --distribution 选项以选择特定的发行版。
- 对 dmesg 的支持有限。 应用程序现在可以登录到 dmesg。 WSL 驱动程序将有限的信息记录到 dmesg。 将来，可以扩展这一功能以传递来自驱动程序的其他信息和诊断数据。
  - 注意：dmesg 当前通过 `/dev/kmsg` 设备接口受支持。 尚不支持 `syslog` syscall 接口。 因此，某些 `dmesg` 命令行选项（如 `-S`），`-C` 不起作用。
- 更改串行设备的默认 gid 和模式，以匹配本机 [GH 3042]
- DrvFs 现在支持扩展属性。
  - 注意：DrvFs 对扩展属性的名称有一些限制。 不允许某些字符（如"/"、":"和"*"），扩展属性名称在 DrvFs 上不区分大小写

内部版本 18252 (Skip Ahead)

有关内部版本 18252 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 将 init 和 bsdtar 二进制文件移出 lxssmanager dll，并移动到单独的工具文件夹中
- 修复在使用 CLONE_FILES 的情况下，关闭文件描述符时出现的争用
- 处理转换 DrvFs 路径时 /proc/pid/mountinfo 中的可选字段
- 允许 DrvFs mknod 成功，而无需对 S_IFREG 提供元数据支持
- 在 DrvFs 上创建的只读文件应具有只读属性集 [GH 3411]
- 添加 /sbin/mount.drvfs 帮助器用于处理 DrvFs 装载
- 在 DrvFs 中使用 POSIX 重命名。
- 允许在无卷 GUID 的卷上执行路径转换。

内部版本 17738 (Fast)

有关 Windows 内部版本 17738 的一般信息，请访问 Windows 博客。

WSL

- Setpriority syscall 权限检查过于严格，导致无法更改同一线程的优先级 [GH 1838]
- 确保使用无偏见的中断时间来计算启动时间，以避免clock_gettime(CLOCK_BOOTTIME)返回负值 [GH 3434]。
- 在 WSL binfmt 解释器中处理符号链接 [GH 3424]
- 更好地处理线程组领先者文件描述符清理。

内部版本 17728 (Fast)

关于 Windows 内部版本 17728 的一般信息，请访问 Windows 博客。

WSL

- 切换 WSL 以使用 KeQueryInterruptTimePrecise 而不是 KeQueryPerformanceCounter，以避免溢出 [GH 3252]
- Ptrace attach 可能导致系统调用返回错误值 [GH 1731]
- 修复许多AF_UNIX相关问题 [GH 3371]
- 修复了以下问题：如果当前工作目录长度小于 5 个字符 [GH 3379]，则可能导致 WSL 互作失败

内部版本 18204 (Skip Ahead)

请访问 Windows 博客，以获取有关 Windows 内部版本 18204 的一般信息。

WSL

- 管道文件系统意外清除边缘触发的 epoll 事件 [GH 3276]
- 通过 NTFS 符号链接启动的 Win32 可执行文件不尊重符号链接名称 [GH 2909]

内部版本 17723 (Fast)

有关内部版本 17723 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 避免导致无法与不存在的端口建立环回连接的一秒延迟 [GH 3286]
- 添加 /proc/sys/fs/file-max 存根文件 [GH 2893]
- 更准确的 IPV6 范围信息。
- PR_SET_PTRACER 支持 [GH 3053]
- 管道文件系统意外清除边缘触发的 epoll 事件 [GH 3276]
- 通过 NTFS 符号链接启动的 Win32 可执行文件不尊重符号链接名称 [GH 2909]

内部版本 17713

有关内部版本 17713 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 改善了 zombie 支持 [GH 1353]
- 添加用于控制 Windows 互作行为的 wsl.conf 条目 [GH 1493]

```
  [interop]

  enabled=false # enable launch of Windows binaries; default is true

  appendWindowsPath=false # append Windows path to $PATH variable; default is
true
```

- 修复 getsockname 不是始终返回 UNIX 套接字系列类型的问题 [GH 1774]
- 添加对 TIOCSTI 的支持 [GH 1863]
- 连接进程中的非阻塞套接字应返回写入尝试的 EAGAIN [GH 2846]
- 支持已装载的 VHD 上的 interop [GH 3246、3291]
- 修复根文件夹的权限检查问题 [GH 3304]
- 对 TTY 键盘 ioctl KDGKBTYPE、KDGKBMODE 和 KDSKBMODE 的有限支持。
- 即使在后台启动，Windows UI 应用也应执行。

内部版本 17704

有关 Windows 内部版本 17704 的一般信息，请访问 Windows 博客 。

WSL

- 添加 wsl -u 或 --user 选项 [GH 1203]
- 修复启用快速启动时的 WSL 启动问题 [GH 2576]
- Unix 套接字需要保留断开连接的对等凭据 [GH 3183]
- 使用 EAGAIN 时非阻塞 Unix 套接字无限期失败 [GH 3191]
- case=off 是新的默认 drvfs 装入点类型 [GH 2937、3212、3328]
  - 有关详细信息，请参阅 博客 。
- 添加 wslconfig /terminate 以停止运行分发。

版本 17692

有关内部版本 17692 的一般 Windows 信息，请访问 Windows 博客 。

WSL

- 修复了 WSL shell 上下文菜单条目无法正确处理路径中含空格的问题。
- 公开按目录区分大小写作为扩展属性
- ARM64：模拟缓存维护操作。 解决 .NET 问题。
- DrvFs：只取消转义专用范围中与已转义字符对应的字符。

版本 17686

有关 Windows 内部版本 17686 的一般信息，请访问 Windows 博客。

WSL

- 修复 ELF 分析程序解释器长度验证中的一位偏移错误 [GH 3154]
- 包含过去时间的 WSL 绝对计时器不会激发 [GH 3091]
- 确保新建的重分析点在父目录中以此类类型列出。
- 以原子方式在 DrvFs 中创建区分大小写的目录。

内部版本 17677

有关 Windows 内部版本 17677 的一般信息，请访问 Windows 博客。

WSL

- 修复了一个附加问题，即即使文件存在，多线程操作也可能返回 ENOENT。 [GH 2712]
- 修复了启用 UMCI 时 WSL 启动失败的问题。 [GH 3020]

内部版本 17666

有关内部版本 17666 的一般 Windows 信息，请访问 Windows 博客。

WSL

警告：存在阻止 WSL 在某些 AMD 芯片集上运行的问题 [GH 3134]。 修复程序已准备就绪，即将在 Insider Build 分支中发布。

- 添加浏览器上下文菜单用于启动 WSL [GH 437、603、1836]。 若要使用此菜单，请在资源管理器窗口中按住 Shift 键的同时单击右键。
- 修复 Unix 套接字非阻塞行为 [GH 2822、3100]
- 修复 GH 2026 中报告的卡住的 NETLINK 命令。
- 添加对装载传播标志的支持 [GH 2911]。

- 修复截断后不会导致 inotify 事件的问题 [GH 2978]。
- 为 wsl.exe 添加 --exec 选项，以在没有 shell 的情况下调用单个二进制文件。
- 为 wsl.exe 添加 --distribution 选项以选择特定的发行版。

内部版本 17655 (Skip Ahead)

有关内部版本 17655 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 对 dmesg 的支持有限。 应用程序现在可以登录到 dmesg。 WSL 驱动程序将有限的信息记录到 dmesg。 将来，可以扩展这一功能以传递来自驱动程序的其他信息和诊断数据。
  - 注意：dmesg 当前通过 `/dev/kmsg` 设备接口受支持。 尚不支持 `syslog` sycall 接口。 因此，某些 `dmesg` 命令行选项（如 `-S`），`-C` 不起作用。
- 修复了即使文件存在，多线程操作也可能返回 ENOENT 的问题。 [GH 2712]

内部版本 17639 (Skip Ahead)

有关 Windows 版本 17639 的常规信息，请访问 Windows 博客。

WSL

- 更改串行设备的默认 gid 和模式，以匹配本机 [GH 3042]
- DrvFs 现在支持扩展属性。
  - 注意：DrvFs 对扩展属性的名称有一些限制。 具体而言，不允许某些字符（如“/”、“：”和“*”），扩展属性名称在 DrvFs 上不区分大小写

内部版本 17133 (Fast)

有关内部版本 17133 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 修复 WSL 中的挂起问题。 [GH 3039， 3034]

内部版本 17128 (Fast)

有关 Windows 版本 17128 的一般信息，请访问 Windows 博客。

WSL

- 没有

内部版本 17627 (Skip Ahead)

有关版本 17627 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 添加对 futex pi 感知操作的支持。 [GH 1006]
  - 请注意，优先级当前不是受支持的 WSL 功能，因此存在限制，但应取消阻止标准使用。
- 对 WSL 进程的 Windows 防火墙支持。 [GH 1852]
  - 例如，若要允许 WSL python 进程侦听任何端口，请使用提升的 Windows cmd：`netsh.exe advfirewall firewall add rule name=wsl_python dir=in action=allow program="C:\users\<username>\appdata\local\packages\canonicalgrouplimited.ubuntuonwindows_79rhkp1fndgsc\localstate\rootfs\usr\bin\python2.7" enable=yes`
  - 有关如何添加防火墙规则的其他详细信息，请参阅 链接
- 当使用 wsl.exe时，应尊重用户的默认 shell。 [GH 2372]
- 将所有网络接口报告为以太网。 [GH 2996]
- 更好地处理损坏的 /etc/passwd 文件。 [GH 3001]

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17618 (Skip Ahead)

有关 Windows 内部版本 17618 的一般性信息，请访问 Windows 博客。

WSL

- 为 NT 互操作引入伪终端功能 [GH 988、1366、1433、1542、2370、2406]。

- 旧安装机制（lxrun.exe）已弃用。 安装分发版支持的机制是通过 Microsoft 应用商店。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

版本 17110

有关 Windows 内部版本 17110 的综合信息，请访问 Windows 博客。

WSL

- 允许在 Windows 上终止 /init [GH 2928]。
- 现在，DrvFs 默认按目录区分大小写（相当于使用“case=dir”装载选项）。
  - 使用“case=force”（旧行为）需要设置注册表项。 如果需要使用“case=dir”，请运行以下命令来启用它：reg add HKLM\SYSTEM\CurrentControlSet\Services\lxss /v DrvFsAllowForceCaseSensitivity /t REG_DWORD /d 1
  - 如果在旧版 Windows 中使用 WSL 创建的现有目录需要区分大小写，请使用 fsutil.exe 将其标记为区分大小写：fsutil.exe file setcasesensitiveinfo `<path>` enable
- NULL 终止从 uname syscall 返回的字符串。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17107

关于 Windows 内部版本 17107 的一般信息，请访问 Windows 博客。

WSL

- 支持主 pty 终结点上的 TCSETSF 和 TCSETSW [GH 2552]。
- 启动同步 interop 进程可能会导致 EINVAL [GH 2813]。
- 修复了PTRACE_ATTACH，以在 /proc/pid/status 中显示正确的跟踪状态。
- 修复以下争用问题：在不清除 TID 地址的情况下，通过 CLEARTID 和 SETTID 标志克隆的生存期较短的进程可能会退出。
- 从 17093 预版本迁移时，在升级 Linux 文件系统目录时显示消息。 有关 17093 文件系统更改的更多详细信息，请参阅 17093 发行说明。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17101

有关版本 17101 的 Windows 一般信息，请访问 Windows 博客。

WSL

- signalfd 支持。 [GH 129]
- 通过将文件名编码为专用 Unicode 字符来支持包含非法 NTFS 字符的文件名。 [GH 1514]
- 不支持写入时，自动装载将回退到只读。 [GH 2603]
- 允许粘贴 Unicode 代理项对（类似于表情符号）。 [GH 2765]
- /proc 和/sys 中的伪文件应从 select、poll、epoll 等命令返回 read 和 write ready [GH 2838]
- 修复了导致服务在注册表被篡改或损坏时进入无限循环的问题。
- 修复 netlink 消息，以兼容较新的（上游 4.14）版本的 iproute2。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17093

有关内部版本 17093 的一般 Windows 信息，请访问 Windows 博客。

重要：

升级到此版本后首次启动 WSL 时，它需要执行一些步骤来升级 Linux 文件系统目录。 这可能需要几分钟时间，因此 WSL 似乎会慢慢启动。 对于从应用商店安装的每个分发版，这应该只发生一次。

- 改善了 DrvFs 中的区分大小写支持。
  - DrvFs 现在支持按目录区分大小写。 这是一个可对目录设置的新标志，用于指示应将这些目录中的所有操作视为区分大小写，使得 Windows 应用程序能够正确打开按大小写区分的文件。
  - DrvFs 提供新的装载选项用于按目录控制区分大小写状态
    - case=force：将所有目录视为区分大小写（驱动器根目录除外）。 使用 WSL 创建的新目录将标记为区分大小写。 这也是一种传统行为，不过，它会将新目录标记为区分大小写。
    - case=dir：仅将具有目录专用区分大小写标志的目录视为区分大小写；其他目录不区分大小写。 使用 WSL 创建的新目录将标记为区分大小写。
    - case=off：只将带有按目录区分大小写标志的目录视为区分大小写；其他目录不区分大小写。 使用 WSL 创建的新目录被标记为不区分大小写。
  - 注意：以前版本中 WSL 创建的目录未设置此标志，因此，如果使用“case=dir”选项，则不会被视为区分大小写。 即将推出一种方法来在现有目录上设置此标志。
  - 使用这些选项进行装载的示例（对于现有驱动器，必须先卸载，然后才能使用不同的选项进行装载）：sudo mount -t drvfs C： /mnt/c -o case=dir
  - 目前，case=force 仍然是默认选项。 这将在未来更改为 case=dir。
- 现在，装载 DrvFs 时，可以在 Windows 路径中使用正斜杠，例如：sudo mount -t drvfs //server/share /mnt/share
- WSL 现在在实例启动 [GH 2636] 期间处理 /etc/fstab 文件。
  - 这种处理是在自动装载 DrvFs 驱动器之前完成的；fstab 已装载的任何驱动器不会自动重新装载，使你可以更改特定驱动器的装入点。
  - 可以使用 wsl.conf 关闭此行为。
- /proc 中的 mount、mountinfo 和 mountstats 文件会正确转义反斜杠和空格等特殊字符 [GH 2799]
- 在启用元数据的情况下使用 DrvFs 创建的特殊文件（例如 WSL 符号链接，或 fifos 和 sockets）现在可以从 Windows 复制和移动。

WSL 更具可配置性，可以通过 wsl.conf 进行配置。

我们添加了一种方法，用于在 WSL 中自动配置某些功能，每次启动子系统时都会应用该功能。 这包括自动装载选项和网络配置。 在我们的博客文章中详细了解它：https://aka.ms/wslconf

AF_UNIX允许在WSL上的Linux进程和Windows本机进程之间建立套接字连接

WSL 和 Windows 应用程序现在可以通过 Unix 套接字相互通信。 假设你想要在 Windows 中运行服务，并使该服务同时可用于 Windows 和 WSL 应用。 现在，可以使用 Unix 套接字实现此目的。 在 https://aka.ms/afunixinterop 的博客文章中阅读详细信息

WSL

- 支持使用 MAP_NORESERVE 的 mmap() [GH 121、2784]
- 支持CLONE_PTRACE和CLONE_UNTRACED [GH 121， 2781]
- 处理克隆中的非 SIGCHLD 终止信号 [GH 121、2781]
- 存根 /proc/sys/fs/inotify/max_user_instances 和 /proc/sys/fs/inotify/max_user_watches [GH 1705]
- 加载包含具有非零偏移量的加载头的 ELF 二进制文件时出错 [GH 1884]
- 加载映像时将尾随页字节归零。
- 减少 execve 以无提示方式终止进程的情况

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

版本 17083

有关 Windows 的一般信息（版本 17083），请访问 Windows 博客。

WSL

- 修复了与 epoll 相关的 bug 检查 [GH 2798、2801、2857]
- 修复了关闭 ASLR 时挂起的问题 [GH 1185、2870]
- 确保 mmap 操作显示原子性 [GH 2732]

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

版本 17074

有关 Windows 内部版本 17074 的一般信息，请访问 Windows 博客。

WSL

- 修复了 DrvFs 元数据的存储格式 [GH 2777]
  重要提示：在此内部版本之前创建的 DrvFs 元数据将显示错误或根本不显示。 若要修复受影响的文件，请使用 chmod 和 chown 重新应用元数据。
- 修复了多个信号和可重启 syscall 的问题。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17063

有关 Windows 内部版本 17063 的通用信息，请访问 Windows 博客。

WSL

- DrvFs 支持其他 Linux 元数据。 这允许使用 chmod/chown 设置文件的所有者和模式，以及创建特殊文件（如 fifos、unix 套接字和设备文件）。 这是暂时禁用的，因为它仍然是实验性的。 **注意：** 我们修复了 DrvFs 使用的元数据格式的 bug。 虽然元数据适用于此版本进行试验，但将来的版本将无法正确读取由此生成创建的元数据。 可能需要手动更新已修改文件的所有者，具有自定义设备 ID 的设备将需要重新创建。

若要启用，请使用元数据选项装载 DrvFs（若要在现有装载上启用 DrvFs，必须先卸载它）：

```
Bash
mount -t drvfs C: /mnt/c -o metadata
```

将 Linux 权限添加为文件的其他元数据;它们不会影响 Windows 权限。 请记住，使用 Windows 编辑器编辑文件可能会删除元数据。 在这种情况下，该文件将还原为其默认权限。

- 向 DrvFs 添加了装载选项，用于控制没有元数据的文件。
  - uid：用于所有文件的所有者的用户 ID。
  - gid：用于所有文件的所有者的组 ID。
  - umask：要对所有文件和目录排除的权限的八进制掩码。
  - fmask：要对所有常规文件排除的权限的八进制掩码。
  - dmask：要对所有目录排除的权限的八进制掩码。

  例如：

```
mount -t drvfs C: /mnt/c -o uid=1000,gid=1000,umask=22,fmask=111
```

  结合元数据选项，为没有元数据的文件指定默认权限。

- 引入了一个新的环境变量，`WSLENV`，用于配置环境变量在 WSL 和 Win32 之间的流动方式。

  例如：

```
Bash
WSLENV=GOPATH/l:USERPROFILE/pu:DISPLAY
```

  `WSLENV` 是一个以冒号分隔的环境变量列表，可以在启动 WSL 进程或从 WSL 启动 Win32 进程时包含这些变量。 每个变量可以使用斜杠后接一个用于指定其转换方式的标志作为后缀。
  - p：该值是应在 WSL 路径与 Win32 路径之间转换的一个路径。
  - l：该值是路径列表。 在 WSL 中，它是以冒号分隔的列表。 在 Win32 中，它是以分号分隔的列表。
  - u：仅当从 Win32 调用 WSL 时，才应包含该值
  - w：仅当从 WSL 调用 Win32 时，才应包含该值

可以在 .bashrc 或用户的自定义 Windows 环境中设置 `WSLENV`。

- drvfs 装入点会正确保留 tar、cp -p 中的时间戳 (GH 1939)
- drvfs 符号链接会报告正确的大小 (GH 2641)
- read/write 适用于极大的 IO 大小 (GH 2653)
- waitpid 可与进程组 ID 一起使用（GH 2534）
- 极大改善了大型保留区域的 mmap 性能；改善了 ghc 性能 (GH 1671)
- READ_IMPLIES_EXEC 的个性化支持；修复 maxima 和 clisp (GH 1185)
- mprotect 支持 PROT_GROWSDOWN；修复 clisp (GH 1128)
- overcommit 模式的页面错误修复；修复 sbcl (GH 1128)
- clone 支持更多标志组合
- 支持 epoll 文件的 select/epoll（以前为 no-op）。
- 通知未实现的 syscall 的 ptrace。
- 忽略生成 resolv.conf 名称服务器时不启动的接口 [GH 2694]
- 枚举没有物理地址的网络接口。 [GH 2685]
- 其他错误修复和改进。

适用于 Windows 上的开发人员的 Linux 工具

- Windows 命令行工具链包括 bsdtar （tar） 和 curl。 阅读 此博客，了解有关添加这两个新工具的更多信息，并了解它们如何影响 Windows 上的开发人员体验。
- 在 Windows 预览版 SDK (17061+) 中可以找到 `AF_UNIX`。 请阅读此博客来详细了解 `AF_UNIX`，以及 Windows 上的开发人员如何使用它。

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17046

有关内部版本 17046 的一般 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 允许进程在没有活动终端的情况下运行。 [GH 709、1007、1511、2252、2391 等]
- 更好地支持CLONE_VFORK和CLONE_VM。 [GH 1878， 2615]
- 跳过 WSL 网络操作的 TDI 筛选器驱动程序。 [GH 1554]
- 满足某些条件时，DrvFs 会创建 NT 符号链接。 [GH 353， 1475， 2602]
  - 链接目标必须是相对性的，不能跨任何装入点或符号链接，并且必须存在。
  - 除非已启用开发人员模式，否则用户必须具有 SE_CREATE_SYMBOLIC_LINK_PRIVILEGE（这通常需要以提升的权限启动 wsl.exe）。
  - 在所有其他情况下，DrvFs 仍会创建 WSL 符号链接。
- 允许同时运行提升和未提升的 WSL 实例。
- 支持 /proc/sys/kernel/yama/ptrace_scope
- 添加 wslpath 以执行 WSL<->Windows 路径转换。 [GH 522、1243、1834、2327 等]

```
wslpath usage:
  -a    force result to absolute path format
  -u    translate from a Windows path to a WSL path (default)
  -w    translate from a WSL path to a Windows path
  -m    translate from a WSL path to a Windows path, with '/' instead of '\\'

  EX: wslpath 'c:\users'
```

控制台

- 无修复措施。

LTP 结果：

正在进行测试。

内部版本 17040

有关内部版本 17040 的一般 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 自 17035 年以来没有修复。

控制台

- 自 17035 年以来没有修复。

LTP 结果：

正在进行测试。

内部版本 17035

关于 Windows 内部版本 17035 的一般信息，请访问 Windows 博客 。

已修复

WSL

- 访问 DrvFs 上的文件偶尔会失败并出现 EINVAL。 [GH 2448]

控制台

- 在 VT 模式下插入/删除行时，某些颜色丢失。

LTP 结果：

正在进行测试。

内部版本 17025

有关 Windows 内部版本 17025 的一般信息，请访问 Windows 博客 。

已修复

WSL

- 在新的前台进程组中启动初始进程 [GH 1653， 2510]。
- SIGHUP 传递修复 [GH 2496]。
- 如果没有提供 [GH 2497]，则生成默认虚拟桥名称。
- 实现 /proc/sys/kernel/random/boot_id [GH 2518]。
- 更多 interop stdout/stderr 管道修复措施。
- 存根 syncfs 系统调用。

控制台

- 修复第三方控制台的输入 VT 转换 [GH 111]

LTP 结果：

正在进行测试。

内部版本 17017

有关内部版本 17017 的一般 Windows 信息，请访问 Windows 博客 。

已修复

WSL

- 忽略空 ELF 程序标头 [GH 330]。
- 允许 LxssManager 为非交互式用户（ssh 和计划的任务支持）创建 WSL 实例 [GH 777，1602]。
- 支持 WSL->Win32->WSL（“起始”）方案 [GH 1228]。
- 有限支持终止通过 interop 调用的控制台应用 [GH 1614]。
- 支持 devpts 的装载选项 [GH 1948]。
- Ptrace 阻止子级启动 [GH 2333]。
- EPOLLET 缺少某些事件 [GH 2462]。
- 返回PTRACE_GETSIGINFO的更多数据。
- 结合 lseek 运行 Getdents 会提供错误的结果。
- 修复某些 Win32 interop 应用挂起，并等待在没有更多数据的管道中提供输入的问题。
- O_ASYNC 对 tty/pty 文件的支持。
- 其他改进和 bug 修复

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

正在进行测试。

Fall Creators Update

内部版本 16288

有关版本 16288 的一般 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 正确初始化和报告套接字文件描述符的 uid、gid 和模式 [GH 2490]
- 其他改进和 bug 修复

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

自 16273 年以来没有更改

版本 16278

有关 Windows 版本 162738 的一般信息，请访问 Windows 博客。

已修复

WSL

- 分解 LX MM 状态时显式取消映射文件后备节的映射视图 [GH 2415]

- 其他改进和 bug 修复

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

自 16273 年以来没有更改

内部版本 16275

有关 Windows 版本 162735 的一般信息，请访问 Windows 博客。

已修复

WSL

- 此版本中没有与 WSL 相关的更改。

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

自 16273 年以来没有更改

版本 16273

有关内部版本 16273 的一般 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 修复了 DrvFs 有时报告目录的错误文件类型的问题 [GH 2392]
- 允许创建 NETLINK_KOBJECT_UEVENT 套接字，以解锁使用 uevent 的程序 [GH 1121, 2293, 2242, 2295, 2235, 648, 637]

- 添加对非阻塞连接的支持 [GH 903、1391、1584、1585、1829、2290、2314]
- 实现 CLONE_FS 克隆系统调用标志 [GH 2242]
- 修复有关在 NT interop 中不会正确处理制表符或引号的问题 [GH 1625、2164]
- 解决尝试重新启动 WSL 实例时的访问被拒绝错误 [GH 651， 2095]
- 实现 futex FUTEX_REQUEUE 和 FUTEX_CMP_REQUEUE 操作 [GH 2242]
- 修复各种 SysF 文件的权限 [GH 2214]
- 修复设置过程中 Haskell 堆栈挂起的问题 [GH 2290]
- 实现 binfmt_misc“C”、“O”和“P”标志 [GH 2103]
- 添加 /proc/sys/kernel /shmmax /shmmni 和 /threads-max [GH 1753]
- 添加对 ioprio_set 系统调用的部分支持 [GH 498]
- 存根 SO_REUSEPORT 和添加 netlink 套接字的 SO_PASSCRED 支持 [GH 69]
- 如果当前正在安装或卸载分发版，则从 RegisterDistribution 返回不同的错误代码。
- 允许通过 wslconfig.exe 取消注册部分安装的 WSL 发行版
- 修复 udp::msg_peek 的 python 套接字测试挂起问题
- 其他改进和 bug 修复

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

总测试数：1904
跳过的测试总数：209
总失败数：229

版本 16257

关于 Windows 内部版本 16257 的一般信息，请访问 Windows 博客。

已修复

WSL

- 实现 prlimit64 系统调用
- 添加对 ulimit -n 的支持 (setrlimit RLIMIT_NOFILE) [GH 1688]
- TCP 套接字的存根 MSG_MORE [GH 2351]
- 修复无效的 AT_EXECFN 辅助矢量行为 [GH 2133]
- 修复控制台/tty 的复制/粘贴行为，并添加更好的完整缓冲区处理 [GH 2204， 2131]
- 在 set-user-ID 和 set-group-ID 程序的辅助矢量中设置 AT_SECURE [GH 2031]

- 伪终端主终结点不处理 TIOCPGRP [GH 1063]
- 修复 lseek 在 LxFs 中的回退目录行为 [GH 2310]
- /dev/ptmx 在大量使用后会死机 [GH 1882]
- 其他改进和 bug 修复

控制台

- 修复横线/下划线四处可见的问题 [GH 2168]
- 修复进程顺序更改，使 NPM 更难以关闭的问题 [GH 2170]
- 添加了我们的新配色方案：https://blogs.msdn.microsoft.com/commandline/2017/08/02/updating-the-windows-console-colors/

LTP 结果：

自 16251 年以来没有更改

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`prlimit64`

已知问题

GitHub 问题 2392：WSL 无法识别的 Windows 文件夹...

在内部版本 16257 中，WSL 在通过 `/mnt/c/...` 枚举 Windows 文件/文件夹时会出现问题。 此问题已修复，应在 2017 年 8 月 14 日当周发布到 Insiders 版本中。

内部版本 16251

有关版本 16251 的常规 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 从 WSL 可选组件中删除 beta 标记，有关详细信息，请参阅 博客文章。
- 在执行时正确初始化 set-user-ID 和 set-group-ID 二进制文件的 saved-set uid 和 gid [GH 962、1415、2072]
- 添加了对 ptrace PTRACE_O_TRACEEXIT 的支持 [GH 555]
- 添加了对包含 NT_FPREGSET 的 ptrace PTRACE_GETFPREGS 和 PTRACE_GETREGSET 的支持 [GH 555]
- 修复了 ptrace 在忽略信号时停止的问题
- 其他改进和 bug 修复

控制台

- 此版本中没有与控制台相关的更改。

LTP 结果：

通过测试数：768
失败测试数：244 个
跳过的测试数：96 个

内部版本 16241

有关内部版本 16241 的一般 Windows 信息，请访问 Windows 博客。

已修复

WSL

- 此版本中没有与 WSL 相关的更改。

控制台

- 修复输出跨行 DEC 的错误字符的问题，最初报告位于此处
- 修复了字符集 65001 (UTF-8) 中未显示任何输出文本的问题。
- 不要在选择更改时将一种颜色的 RGB 值所做的更改传输到调色板的其他部分。 这会使控制台属性表更易于使用。
- Ctrl+S 似乎无法正常工作
- ANSI 转义代码中根本没有 Un-Bold/-Dim [GH 2174]
- 控制台无法正确支持 Vim 颜色主题 [GH 1706]
- 无法粘贴特定字符 [GH 2149]

- 当编辑/命令行上有内容时，重复流的调整大小操作以奇怪的方式与 bash 窗口调整大小操作交互 [GH ConEmu 1123]
- 按 Ctrl-L 不会清屏 [GH 1978]
- 在 HDPI 显示器上显示 VT 时遇到控制台渲染错误 [GH 1907]
- 日文字符出现乱码和 Unicode 字符 U+30FB [GH 2146]
- 其他改进和 bug 修复

内部版本 16237

有关内部版本 16237 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 对 lxfs 中不包含 EA 的文件使用默认属性 (root, root, 0000)
- 添加了对使用扩展属性的分发的支持
- 修复 getdents 和 getdents64 返回的项的填充
- 修复shmctl SHM_STAT系统调用的权限验证 [GH 2068]
- 修复了 tty 的错误初始 epoll 状态 [GH 2231]
- 修复 DrvFs readdir 不返回所有项的问题 [GH 2077]
- 修复取消链接文件时的 LxFs readdir [GH 2077]
- 允许通过 procfs 重新打开未链接的 drvfs 文件
- 添加了用于禁用 WSL 功能的全局注册表项重写（interop/驱动器装载）
- 修复 DrvFs（和 LxFs）的“统计信息”中的错误块计数 [GH 1894]
- 其他改进和 bug 修复

内部版本 16232

有关内部版本 16232 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 此版本中没有与 WSL 相关的更改。

内部版本 16226

有关 Windows 版本 16226 的常规信息，请访问 Windows 博客。

已修复

- xattr 相关的 syscall 支持（getxattr、setxattr、listxattr、removexattr）。
- security.capability xattr 支持。
- 改进了与某些文件系统和筛选器（包括非 MS SMB 服务器）的兼容性。 [GH #1952]
- 改进了对 OneDrive 占位符、GVFS 占位符以及紧凑操作系统压缩文件的支持。
- 其他改进和 bug 修复

内部版本 16215

有关内部版本 16215 的一般 Windows 信息，请访问 Windows 博客。

已修复

- WSL 不再需要开发人员模式。
- 支持 drvfs 中的目录交接点。
- 处理 WSL appx 分发包的卸载。
- 更新 procfs 以显示专用映射和共享映射。
- 为 wslconfig.exe 添加清理部分安装或已卸载的发行版的功能。
- 添加了对 TCP 套接字 IP_MTU_DISCOVER 的支持。 [GH 1639， 2115， 2205]
- 推断 AF_INADDR 路由的协议系列。
- 串行设备改进 [GH 1929]。

内部版本 16199

有关内部版本 16199 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 这些版本中没有与 WSL 相关的更改。

内部版本 16193

有关版本 16193 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 发送 SIGCONT 与终止线程组之间的争用状态 [GH 1973]
- 更改 tty 和 pty 设备以报告 FILE_DEVICE_NAMED_PIPE 而不是 FILE_DEVICE_CONSOLE [GH 1840]
- IP_OPTIONS 的 SSH 修复
- 已将 DrvFs 装载移到初始化守护程序 [GH 1862、1968、1767、1933]
- 在 DrvFs 中添加了遵循 NT 符号链接的支持。

内部版本 16184

有关内部版本 16184 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 删除了 apt 包维护任务（lxrun.exe /update）
- 修复了 node.js 中的 Windows 进程不显示输出的问题 [GH 1840]
- 放宽 lxcore 中的对齐要求 [GH 1794]
- 修复了许多系统调用中处理 AT_EMPTY_PATH 标志的问题。
- 修复了删除存在已打开句柄的 DrvFs 文件时，导致文件出现未定义的行为的问题 [GH 544、966、1357、1535、1615]
- /etc/hosts 现在将从 Windows 主机文件继承条目（%windir%\system32\drivers\etc\hosts） [GH 1495]

内部版本 16179

关于 Windows 内部版本 16179 的一般信息，请访问 Windows 博客。

已修复

- 本周没有 WSL 变动。

版本 16176

有关 Windows 的常规信息，请访问版本 16176 的 Windows 博客。

已修复

- 已启用串行支持
- 添加了 IP 套接字选项 IP_OPTIONS [GH 1116]
- 实现了 pwritev 函数（将文件上传到 nginx/PHP-FPM 时）[GH 1506]
- 添加了 IP 套接字选项 IP_MULTICAST_IF 和 IPV6_MULTICAST_IF [GH 990]
- 支持套接字选项 IP_MULTICAST_LOOP 和 IPV6_MULTICAST_LOOP [GH 1678]
- 为应用节点、traceroute、dig、nslookup、主机添加了 IP(V6)_MTU 套接字选项
- 添加了 IP 套接字选项 IPV6_UNICAST_HOPS
- 文件系统改进
  - 允许装载 UNC 路径
  - 在 drvfs 中启用 CDFS 支持
  - 正确处理 drvfs 中网络文件系统的权限
  - 向 drvfs 添加对远程驱动器的支持
  - 在 drvfs 中启用 FAT 支持
- 其他修复和改进

LTP 结果

自 15042 年以来没有更改

内部版本 16170

有关 Windows 内部版本 16170 的一般信息，请访问 Windows 博客。

我们发布的新博客文章中介绍了我们在测试 WSL 方面所做的努力。

已修复

- 支持套接字选项 IP_ADD_MEMBERSHIP 和 IPV6_ADD_MEMBERSHIP [GH 1678]
- 添加对 PTRACE_OLDSETOPTIONS 的支持。 [GH 1692]
- 其他修复和改进

LTP 结果

自 15042 年以来没有更改

内部版本 15046 到 Windows 10 创意者更新

我们未计划在 Windows 10 创意者更新中包含其他 WSL 修复或功能。 WSL 的发行说明将在未来几周恢复发布，其中补充了面向下一个 Windows 更新主要版本的信息。 有关内部版本 15046 和将来的预览体验版的一般 Windows 信息，请访问 Windows 博客。

版本 15042

有关版本 15042 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 修复删除以“...”结尾的路径时出现死锁的问题
- 修复了 FIONBIO 在成功时不返回 0 的问题 [GH 1683]
- 修复了 inet 数据报套接字的零长度读取问题
- 修复 drvfs inode 查找中的争用状况可能导致死锁的问题 [GH 1675]
- 扩展了对 unix 套接字辅助数据的支持；SCM_CREDENTIALS 和 SCM_RIGHTS [GH 514、613、1326]
- 其他修复和改进

LTP 结果：

通过测试数：737
非通过次数（失败、跳过等...）：255

内部版本 15031

有关 Windows 内部版本 15031 的一般信息，请访问 Windows 博客。

已修复

- 修复了 time(2) 偶尔行为异常的 bug。
- 修复了 *SIGPROCMASK syscall 可能损坏信号掩码的问题。

- 现在，在 WSL 进程创建通知中返回完整的命令行长度。 [GH 1632]
- WSL 现在会针对 GDB 挂起通过 ptrace 报告线程退出。 [GH 1196]
- 修复了在收到繁重 tmux IO 后 ptys 挂起的 bug。 [GH 1358]
- 修复了许多系统调用中的超时验证（futex、semtimedop、ppoll、sigtimedwait、itimer、timer_create）
- 添加了 eventfd EFD_SEMAPHORE 支持 [GH 452]
- 其他修复和改进

LTP 结果：

通过测试数：737
非通过次数（失败、跳过等）：255

内部版本 15025

有关版本 15025 的 Windows 一般信息，请访问 Windows 博客。

已修复

- 修复破坏 grep 2.27 的 bug [GH 1578]
- 为 eventfd2 syscall 实现了 EFD_SEMAPHORE 标志 [GH 452]
- 实现 /proc/[pid]/net/ipv6_route [GH 1608]
- 针对 unix 流套接字的信号驱动 IO 支持 [GH 393、68]
- 支持F_GETPIPE_SZ和F_SETPIPE_SZ [GH 1012]
- 实现 recvmmsg() syscall [GH 1531]
- 修复了 epoll_wait() 不等待的 bug [GH 1609]
- 实现 /proc/version_signature
- 现在，如果两个文件描述符都引用同一个管道，Tee 系统调用将返回失败。
- 为 Unix 套接字的 SO_PEERCRED 实现了正确的行为
- 修复了 tkill syscall 处理无效参数的方法
- 更改以提高 drvfs 的性能
- 针对 Ruby IO 阻塞的次要修复
- 修复了 recvmsg() 对 inet 套接字的 MSG_DONTWAIT 标志返回 EINVAL 的问题 [GH 1296]
- 其他修复和改进

LTP 结果：

通过测试数：732
非通过次数（失败、跳过等）：255

内部版本 15019

有关 Windows 版本 15019 的一般信息，请访问 Windows 博客。

已修复

- 修复了 htop 等工具在 procfs 中错误报告 CPU 使用率的 bug（GH 823、945、971）
- 对现有的文件结合 O_TRUNC 调用 open() 时，inotify 现在会在 IN_OPEN 的前面生成 IN_MODIFY
- 对 unix 套接字 getsockopt SO_ERROR 的修复，使 postgres 可以启用 [GH 61， 1354]
- 为 GO 语言实现 /proc/sys/net/core/somaxconn
- Apt-get 软件包更新后台任务现在在用户不可见的情况下运行
- 清除 ipv6 localhost 的范围（Spring-Framework(Java) 失败）。
- 其他修复和改进

LTP 结果：

通过测试数：714
非通过数（失败、跳过等...）：249

内部版本 15014

有关 Windows 内部版本 15014 的一般信息，请访问 Windows 博客。

已修复

- Ctrl+C 现在按预期工作
- htop 和 ps auxw 现在显示正确的资源利用率 （GH #516）
- NT 异常到信号的基本转换。 （GH #513）
- 当空间耗尽时，fallocate 现在会失败并返回 ENOSPC，而不是返回 EINVAL (GH #1571)
- 添加了 /proc/sys/kernel/sem。
- 实现了 semop 和 semtimedop 系统调用
- 修复了 IP_RECVTOS 和 IPV6_RECVTCLASS 套接字选项的 nslookup 错误 (GH 69)
- 支持套接字选项 IP_RECVTTL 和 IPV6_RECVHOPLIMIT
- 其他修复和改进

LTP 结果：

通过测试的数量：709
非通过数（失败、跳过等...）：255

Syscall 摘要

Syscall 总数：384
已实现总数：235
已存根总数：22
未实现总数：127

内部版本 15007

有关 Windows 内部版本 15007 的一般信息，请访问 Windows 博客。

已知问题

- 已知 bug：控制台无法识别某些 Ctrl + `<key>` 输入。 这包括 ctrl-c 命令，该命令将像普通的“c”键按下那样工作。
  - 解决方法：将备用键映射到 Ctrl+C。 例如，若要将 Ctrl+K 映射到 Ctrl+C，请执行以下作：`stty intr \^k`。 这种映射是按终端进行的，*每次*启动 bash 都必须执行。 用户可以探索该选项，以将此映射包含在其 `.bashrc` 中

已修复

- 更正了运行 WSL 会消耗 100% 的 CPU 核心的问题
- 套接字选项IP_PKTINFO和IPV6_RECVPKTINFO现在都支持。 （GH #851，987）
- 将网络接口物理地址截断为 lxcore 中的 16 字节（GH #1452、1414、1343、468、308）
- 其他修复和改进

LTP 结果：

通过测试的数量：709
非通过数（失败、跳过等...）：255

版本 15002

有关内部版本 15002 的一般 Windows 信息，请访问 Windows 博客。

已知问题

两个已知问题：

- 已知 bug：控制台无法识别某些 Ctrl + `<key>` 输入。 这包括 ctrl-c 命令，该命令将像普通的“c”键按下那样工作。
  - 解决方法：将备用键映射到 Ctrl+C。 例如，若要将 Ctrl+K 映射到 Ctrl+C，请执行以下作：`stty intr \^k`。 这种映射是按终端进行的，*每次*启动 bash 都必须执行。 用户可以探索该选项，以将此映射包含在其 `.bashrc` 中

- 当 WSL 运行时，系统线程将消耗 100% 的 CPU 核心。 根本原因已在内部得到解决并修复。

已修复

- 现在必须在同一权限级别创建所有 bash 会话。 尝试在不同的级别启动会话将被阻止。 这意味着管理员和非管理控制台不能同时运行。 （GH #626）
- 实现了以下NETLINK_ROUTE消息（需要 Windows 管理员）
  - RTM_NEWADDR（支持 `ip addr add`）
  - RTM_NEWROUTE（支持 `ip route add`）
  - RTM_DELADDR（支持 `ip addr del`）
  - RTM_DELROUTE（支持 `ip route del`）
- 用于检查要更新的包的计划任务将不再在按流量计费的连接上运行 (GH #1371)
- 修复了运行 bash -c "ls -alR /" | bash -c "cat" 时管道停滞的错误 (GH #1214)
- 实现了 TCP_KEEPCNT 套接字选项 (GH #843)
- 实现了 IP_MTU_DISCOVER INET 套接字选项（GH #720、717、170、69）
- 删除了旧功能，以通过 NT 路径查找从 init 运行 NT 二进制文件。 （GH #1325）
- 更改 /dev/kmsg 的模式以允许用户组和其他用户读取访问权限（0644） （GH #1321）
- 实现了 /proc/sys/kernel/random/uuid (GH #1092)
- 更正了进程开始时间显示为 2432 年的错误（GH #974）
- 将默认 TERM 环境变量切换到 xterm-256color （GH #1446）
- 修改了进程分叉期间进程提交的计算方式。 （GH #1286）
- 实现了 /proc/sys/vm/overcommit_memory。 （GH #1286）
- 实现了 /proc/net/route 文件 （GH #69）
- 修复了快捷方式名称未正确本地化的错误（GH #696）
- 修复了错误地验证程序标头必须小于（或等于）PATH_MAX 的 elf 分析逻辑。 （GH #1048）
- 为 procfs、sysfs、cgroupfs 和 binfmtfs 实现了 statfs 回调 (GH #1378)

- 修复了不会关闭的 AptPackageIndexUpdate 窗口（GH #1184，在 GH #1193 中也进行了讨论）
- 添加了 ASLR 个性化 ADDR_NO_RANDOMIZE 支持。 （GH #1148， 1128）
- 改善了 PTRACE_GETSIGINFO、SIGSEGV，在 AV 期间会提供正确的 gdb 堆栈跟踪 (GH #875)
- 对于 patchelf 二进制文件，Elf 分析不再失败。 （GH #471）
- 传播到 /etc/resolv.conf 的 VPN DNS （GH #416， 1350）
- TCP 关闭改进，可以更可靠地传输数据。 （GH #610， 616， 1025， 1335）
- 现在，当打开的文件过多时，将返回正确的错误代码（EMFILE）。 （GH #1126， 2090）
- Windows 审核日志现在会在进程创建审核中报告映像名称。
- 现在，在从 bash 窗口内部启动 bash 时会正常失败
- 添加了在 interop 无法访问 LxFs 下的工作目录（即 notepad.exe .bashrc）时显示的错误消息
- 修复了 Windows 路径在 WSL 中截断的问题
- 其他修复和改进

LTP 结果：

通过测试的数量：690
非通过数（失败、跳过等...）：274

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`shmctl`

`shmget`

`shmdt`

`shmat`

内部版本 14986

有关 Windows 版本 14986 的通用信息，请访问 Windows 博客。

已修复

- 修复了 Netlink 和 Pty IOCTL 的 bug 检查

- 内核版本现在报告 4.4.0-43，以便与 Xenial 保持一致性
- 现在，当输入定向到 'nul:' 时会启动 Bash.exe (GH #1259)
- 现在，procfs 中会正确报告线程 ID (GH #967)
- 现在，inotify_add_watch() 中支持 IN_UNMOUNT | IN_Q_OVERFLOW | IN_IGNORED | IN_ISDIR 标志 (GH #1280)
- 实现 timer_create 和相关的系统调用。 这会启用 GHC 支持 (GH #307)
- 修复了 ping 返回时间为 0.000 毫秒的问题（GH #1296）
- 打开的文件过多时，返回正确的错误代码。
- 修复了 WSL 中的问题：如果网络接口的硬件地址为 32 字节（例如 Teredo 接口），则针对该网络接口数据的 Netlink 请求将会失败并返回 EINVAL
  - 请注意，Linux“ip”实用工具包含一个 bug，如果 WSL 报告 32 字节硬件地址，它将崩溃。 这是“ip”（而不是 WSL）中的一个 bug。 “ip”实用工具硬编码用于打印硬件地址的字符串缓冲区的长度，并且该缓冲区太小，无法打印 32 字节的硬件地址。
- 其他修复和改进

LTP 结果：

通过测试数：669
非通过次数（失败、跳过等）：258

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`timer_create`

`timer_delete`

`timer_gettime`

`timer_settime`

内部版本 14971

有关版本 14971 的通用 Windows 信息，请访问 Windows 博客。

已修复

- 由于无法控制的情况，适用于 Linux 的 Windows 子系统在此版本中没有更新。 定期计划的更新将在下一个版本中恢复。

LTP 结果：

与14965
通过测试的数量不变：664
非通过次数（失败、跳过等）：263

内部版本 14965

有关 Windows 版本 14965 的一般信息，请访问 Windows 博客。

已修复

- 支持 Netlink 套接字 NETLINK_ROUTE 协议的 RTM_GETLINK 和 RTM_GETADDR (GH #468)
  - 为网络枚举启用 ifconfig 和 ip 命令
- /sbin 现在默认位于用户的路径中
- NT 用户路径现在默认追加到 WSL 路径（即现在可以键入 notepad.exe 而不将 System32 添加到 Linux 路径）
- 添加了对 /proc/sys/kernel/cap_last_cap的支持
- 现在，当当前工作目录包含非ANSI字符（GH #1254）时，可以从Windows子系统（WSL）中启动NT二进制文件。
- 允许在断开连接的 unix 流套接字上关闭。
- 添加了对PR_GET_PDEATHSIG的支持。
- 添加了对CLONE_PARENT的支持
- 修复了运行 bash -c "ls -alR /" | bash -c "cat" 时管道停滞的错误 (GH #1214)
- 处理连接到当前终端的请求。
- 将 `/proc/<pid>/oom_score_adj` 标记为可写。
- 添加 /sys/fs/cgroup 文件夹。
- sched_setaffinity 应返回关联位掩码的数目
- 修复了错误地假定解释器路径长度必须小于 64 个字符的 ELF 验证逻辑。 （GH #743）
- 打开文件描述符可以保持控制台窗口打开（GH #1187）

- 修复了当目标名称包含尾随斜杠时 rename() 失败的错误 (GH #1008)
- 实现 /proc/net/dev 文件
- 修复了由于计时器分辨率导致的 0.000 毫秒的 ping 问题。
- 实现了 /proc/self/environ (GH #730)
- 其他问题修复和改进

LTP 结果：

通过测试数：664
非通过次数（失败、跳过等）：263

内部版本 14959

有关内部版本 14959 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 改进了适用于 Windows 的 Pico 进程通知。 在 WSL 博客中可以找到更多信息。
- 改善了 Windows 互操作的稳定性
- 修复了启用企业数据保护（EDP）时启动 bash.exe 时出现的错误代码 0x80070057。
- 其他问题修复和改进

LTP 结果：

通过测试数：665
非通过次数（失败、跳过等）：263

内部版本 14955

有关版本 14955 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 由于无法控制的情况，适用于 Linux 的 Windows 子系统在此版本中没有更新。 定期计划的更新将在下一个版本中恢复。

LTP 结果：

通过测试数：665
非通过次数（失败、跳过等）：263

内部版本 14951

有关 Windows 版本 14951 的一般信息，请访问 Windows 博客。

新功能：Windows/Ubuntu 互作性

现在可以直接从 WSL 命令行调用 Windows 二进制文件。 这样，用户就可以以不可能的方式与其 Windows 环境和系统进行交互。 作为一个快速示例，用户现在可以运行以下命令：

```Bash
$ export PATH=$PATH:/mnt/c/Windows/System32
$ notepad.exe
$ ipconfig.exe | grep IPv4 | cut -d: -f2
$ ls -la | findstr.exe foo.txt
$ cmd.exe /c dir
```

有关详细信息，请参阅：

- WSL 团队的互操作博客
- WSL 文件系统文档

已修复

- 现已为所有新的 WSL 实例安装 Ubuntu 16.04 （Xenial）。 具有现有 14.04（Trusty） 实例的用户不会自动升级。
- 现在会显示安装过程中指定的区域设置
- 终端改进，包括修复了不是总能将 WSL 进程重定向到文件的 bug
- 控制台生存期应与 bash.exe 生存期密切关联
- 控制台窗口大小应使用可见大小，而不是缓冲区大小
- 其他问题修复和改进

LTP 结果：

通过测试数：665
非通过次数（失败、跳过等）：263

内部版本 14946

有关内部版本 14946 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 修复了阻止为包含空格或引号的 NT 用户名的用户创建 WSL 用户帐户的问题。
- 更改 VolFs 和 DrvFs，以在统计信息中返回 0 作为目录的链接计数
- 支持 IPV6_MULTICAST_HOPS 套接字选项。
- 限制为每个 tty 只有一个控制台 I/O 循环。 示例：以下命令可能：
  - bash -c“回显数据”|bash -c“ssh user@example.com 'cat > foo.txt'”
- 在 /proc/cpuinfo 中将空格替换为制表符（GH #1115）
- DrvFs 现在会显示在 mountinfo 中，其中包含一个与已装载的 Windows 卷匹配的名称
- /home 和 /root 现在显示在 mountinfo 中，名称正确
- 其他问题修复和改进

LTP 结果：

通过测试数：665
非通过次数（失败、跳过等）：263

内部版本 14942

有关内部版本 14942 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 解决了一些 bug 检查问题，包括阻止 SSH 的“尝试执行不可执行的内存”网络崩溃
- inotifiy 对从 DrvFs 上的 Windows 应用程序生成通知的支持目前处于
- 为 mongod 实现TCP_KEEPIDLE和TCP_KEEPINTVL。 （GH #695）
- 实现 pivot_root 系统调用
- 实现 SO_DONTROUTE 的套接字选项

- 其他问题修复和改进

LTP 结果：

通过测试数：665
非通过次数（失败、跳过等）：263

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`pivot_root`

内部版本 14936

有关内部版本 14936 的一般 Windows 信息，请访问 Windows 博客。

注意：WSL 将在即将发布的版本中安装 Ubuntu 版本 16.04（Xenial），而不是 Ubuntu 14.04（Trusty）。 对于安装新实例（lxrun.exe /install 或首次运行 bash.exe）的预览体验成员，此更改将适用。 不会自动升级具有 Trusty 的现有实例。 用户可以使用 do-release-upgrade 命令将其 Trusty 映像升级到 Xenial。

已知问题

WSL 遇到了某些套接字实现的问题。 bug 检查将崩溃，出现错误“尝试执行不可执行的内存”。 此问题的最常见表现是使用 ssh 时崩溃。 根本原因已在内部版本中修复，并将在第一时间推送给内测用户。

已修复

- 实现了 chroot 系统调用
- inotifiy 的改进，~~包括支持从 DrvFs 上的 Windows 应用程序生成通知~~
  - 更正：Inotify 对源自 Windows 应用程序的更改的支持目前不可用。
- 与 `IPV6::<port n>` 的套接字绑定现在支持 `IPV6_V6ONLY`（GH #68、#157、#393、#460、#674、#740、#982、#996）
- 实现了 waitid 系统调用的 WNOWAIT 行为 (GH #638)
- 支持 IP 套接字选项 IP_HDRINCL 和 IP_TTL
- 零长度 read() 应立即返回 (GH #975)

- 正确处理在.tar文件中不包含 NULL 终止符的文件名和文件名前缀。
- 对 /dev/null 的 epoll 支持
- 修复 /dev/alarm 时间源
- Bash -c 现在能够重定向到文件
- 其他问题修复和改进

LTP 结果：

通过测试数：664
非通过数（失败、跳过等...）：264

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`chroot`

内部版本 14931

有关内部版本 14931 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 由于无法控制的情况，适用于 Linux 的 Windows 子系统在此版本中没有更新。 定期计划的更新将在下一版本中恢复。

内部版本 14926

有关内部版本 14926 的一般 Windows 信息，请访问 Windows 博客。

已修复

- Ping 现在可在没有管理员权限的控制台中工作
- Ping6 现在也受支持，也不支持管理员权限
- 对通过 WSL 修改的文件的 Inotify 支持。 （GH #216）
  - 支持的标志：

    - inotify_init1：LX_O_CLOEXEC、LX_O_NONBLOCK
    - inotify_add_watch 活动：LX_IN_ACCESS、LX_IN_MODIFY、LX_IN_ATTRIB、LX_IN_CLOSE_WRITE、LX_IN_CLOSE_NOWRITE、LX_IN_OPEN、LX_IN_MOVED_FROM、LX_IN_MOVED_TO、LX_IN_CREATE、LX_IN_DELETE、LX_IN_DELETE_SELF、LX_IN_MOVE_SELF
    - inotify_add_watch 属性：LX_IN_DONT_FOLLOW、LX_IN_EXCL_UNLINK、LX_IN_MASK_ADD、LX_IN_ONESHOT、LX_IN_ONLYDIR
    - 读取输出：LX_IN_ISDIR、LX_IN_IGNORED
  - 已知问题：从 Windows 应用程序修改文件不会生成任何事件
- Unix 套接字现在支持 SCM_CREDENTIALS

LTP 结果：

通过测试数：651
非通过次数（失败、跳过等）：258

内部版本 14915

有关内部版本 14915 的一般 Windows 信息，请访问 Windows 博客。

已修复

- Unix 数据报套接字的套接字对 (GH #262)
- 对 SO_REUSEADDR 的 Unix 套接字支持
- 对 SO_BROADCAST 的 UNIX 套接字支持 (GH #568)
- 对 SOCK_SEQPACKET 的 Unix 套接字支持（GH #758、#546）
- 添加对 Unix 数据报套接字发送、接收和关闭的支持
- 修复对非固定地址的无效 mmap 参数验证导致的 bug 检查。 （GH #847）
- 支持暂停/恢复终端状态
- 支持使用 TIOCPKT ioctl 阻止 Screen 实用工具 (GH #774)
  - 已知问题：功能键无法使用
- 更正了 TimerFd 中的一种争用状态，该状态可能导致 LxpTimerFdWorkerRoutine 访问已释放的成员“ReaderReady” (GH #814)
- 为 futex、poll 和 clock_nanosleep 启用可重启的系统调用支持
- 添加了绑定装载支持
- 装载命名空间支持的取消共享
  - 已知问题：使用 `unshare(CLONE_NEWNS)` 创建新的装载命名空间时，当前工作目录将继续指向旧命名空间
- 其他改进和 bug 修复

内部版本 14905

有关版本 14905 的常规 Windows 信息，请访问 Windows 博客。

已修复

- 现在支持可重启的系统调用 （GH #349， GH #520）
- 现在可正常使用指向以 / 结尾的目录的符号链接 (GH #650)
- 为 /dev/random 实现了 RNDGETENTCNT ioctl
- 实现了 /proc/[pid]/mounts、/proc/[pid]/mountinfo 和 /proc/[pid]/mountstats 文件
- 其他问题修复和改进

内部版本 14901

Windows 10 周年更新版的第一个预览体验内部版本。

有关内部版本 14901 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 修复了尾随斜杠问题
  - 命令（如 `$ mv a/c/ a/b/`）现在正常工作
- 现在安装会提示是否应将 Ubuntu 区域设置设置为 Windows 区域设置
- 对 ns 文件夹的 Procfs 支持
- 添加了 tmpfs、procfs 和 sysfs 文件系统的装入点和卸载点
- 修复 mknod[at] 32 位 ABI 签名
- Unix 套接字已移到调度模型
- 应遵循使用 setsockopt 设置的 INET 套接字接收缓冲区大小
- 实现 MSG_CMSG_CLOEXEC unix 套接字接收消息标志
- Linux 进程 stdin/stdout 管道重定向 （GH #2）
  - 允许在 CMD 中通过管道传输 bash -c 命令。 示例：>dir | bash -c "grep foo"
- Bash 现在可以安装在具有多个页文件的系统 （GH #538， #358）
- 默认 INET 套接字缓冲区大小应与默认 Ubuntu 设置的缓冲区大小匹配
- 将 xattr syscalls 与 listxattr 对齐
- 仅返回具有来自 SIOCGIFCONF 的有效 IPv4 地址的接口
- 修复 ptrace 注入时的信号默认操作

- 实现 /proc/sys/vm/min_free_kbytes
- 在 sigreturn 中还原上下文时使用计算机上下文寄存器值
  - 这解决了某些用户遇到的 java 和 javac 挂起问题
- 实现 /proc/sys/kernel/hostname

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`waitid`

`epoll_pwait`

Windows 10 周年更新的内部版本 14388

有关 Windows 版本 14388 的一般信息，请访问 Windows 博客。

已修复

- 为 8 月 2 日的 Windows 10 周年更新做准备的修复事项
  - 有关周年更新中的 WSL 的详细信息，请参阅我们的 博客

版本 14376

有关版本 14376 的 Windows 一般信息，请访问 Windows 博客。

已修复

- 删除了一些存在 apt-get 挂起问题的实例 (GH #493)
- 修复了不正确处理空装入点的问题
- 修复了不正确装载 ramdisk 的问题
- 更改 unix 套接字接受行为以支持标志（在 GH #451 中做了部分描述）
- 修复了常见的网络相关蓝屏问题
- 修复了访问 /proc/[pid]/task 时出现蓝屏问题 （GH #523）
- 修复了某些 pty 场景中的高 CPU 占用率（GH #488, #504）
- 其他问题修复和改进

内部版本 14371

有关内部版本 14371 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 更正了使用 ptrace 时 SIGCHLD 和 wait() 出现计时争用的问题
- 更正了当路径包含尾随 / 时出现的某种行为 (GH #432)
- 修复了由于打开子级句柄导致重命名/取消链接失败的问题
- 其他问题修复和改进

内部版本 14366

有关 Windows 内部版本 14366 的一般信息，请访问 Windows 博客。

已修复

- 修复通过符号链接创建文件的问题
- 添加了 Python 的 listxattr (GH 385)
- 其他问题修复和改进

系统调用支持

- 下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`listxattr`

内部版本 14361

有关 Windows 内部版本 14361 的一般信息，请访问 Windows 博客 。

已修复

- 现在，在 Windows 上的 Ubuntu Bash 中运行时，DrvFs 区分大小写。
  - 用户可以在其 /mnt/c 驱动器上使用 case.txt 和 CASE.TXT

    - 只有 Windows 上的 Ubuntu Bash 支持区分大小写。 当在 Bash 之外时，NTFS 将正确报告文件，但在 Windows 中与文件交互时可能会发生意外行为。
    - 每个卷的根目录（即 /mnt/c）不区分大小写
    - 有关在 Windows 中处理这些文件的详细信息，请参阅此处 。
- 大幅增强了 pty/tty 支持。 现在支持 TMUX 等应用程序（GH #40）
- 修复了安装过程中用户帐户未总是被创建的问题
- 优化的命令行参数结构，允许极长的参数列表。 （GH #153）
- 现在可对 DrvFs 中的只读文件执行删除和 chmod
- 修复了一些在断开连接时终端会挂起的实例 (GH #43)
- chmod 和 chown 现在适用于 tty 设备
- 允许连接到作为 localhost 的 0.0.0.0 和 :: (GH #388)
- Sendmsg/recvmsg 现在可处理 >1 的 IO 矢量长度（在 GH #376 中做了部分描述）
- 用户现在可以选择不使用自动生成的主机文件（GH #398）
- 在安装期间自动匹配 Linux 区域设置与 NT 区域设置（GH #11）
- 添加了 /proc/sys/vm/swappiness 文件 (GH #306)
- strace 现在会正常退出
- 允许通过 /proc/self/fd 重新打开管道 （GH #222）
- 在 DrvFs 中隐藏 %LOCALAPPDATA%\lxss 下的目录 (GH #270)
- 更好地处理 bash.exe ~。 现在支持“bash ~ -c ls”等命令（GH #467）
- 现在，在关闭期间，套接字会通知 epoll read 可用 (GH #271)
- lxrun /uninstall 可以更好地删除文件和文件夹
- 更正了 ps -f (GH #246)
- 改进了对 x11 应用（如 xEmacs）的支持 （GH #481）
- 更新了初始线程堆栈大小以匹配默认 Ubuntu 设置，并将大小正确报告到 get_rlimit syscall （GH #172， #258）
- 改善了 pico 进程映像名称的报告（例如，用于审核）
- 实现了 df 命令的 /proc/mountinfo
- 修复了子名称 . 和 .. 的符号链接错误代码
- 其他 bug 修复和改进

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`GETTIMER`

`MKNODAT`

`RENAMEAT`

`SENDFILE`

`SENDFILE64`

`SYNC_FILE_RANGE`

内部版本 14352

有关内部版本 14352 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 修复了未正确下载/创建大型文件的问题。 这应该可以解除 npm 和其他方案的阻碍（GH #3、GH #313）
- 删除了一些存在套接字挂起情况的实例
- 更正了一些 ptrace 错误
- 修复了 WSL 允许文件名超过 255 个字符的问题
- 改进了对非英语字符的支持
- 添加当前的 Windows 时区数据并设置为默认值
- 每个装入点的唯一设备 ID（JRE 修复 – GH #49）
- 更正了路径包含“.”和“..”的问题
- 添加了 Fifo 支持 (GH #71)
- 已更新的 resolv.conf 格式以匹配本机 Ubuntu 格式
- 一些 procfs 清理
- 为管理员控制台启用了 ping (GH #18)

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`FALLOCATE`

`EXECVE`

`LGETXATTR`

`FGETXATTR`

内部版本 14342

有关内部版本 14342 的一般 Windows 信息，请访问 Windows 博客。

有关 VolFs 和 DriveF 的信息，请参阅 WSL 博客。

已修复

- 修复了 Windows 用户用户名中存在 Unicode 字符时的安装问题
- 常见问题解答中的 apt-get update udev 解决方法现在默认在首次运行时提供
- 在 DriveFs (`/mnt/<drive>`) 目录中启用了符号链接
- 现在可以在 DriveFs 和 VolFs 之间使用符号链接
- 解决了顶级路径分析问题：ls .// 现在将按预期工作
- 在 DriveF 上安装 npm，-g 选项现已正常工作
- 修复了阻止 PHP 服务器启动的问题
- 更新了默认环境值，例如$PATH以更紧密地匹配本机 Ubuntu
- 在 Windows 中添加了每周维护任务以更新 apt 包缓存
- 修复了 ELF 标头验证的问题，WSL 现在支持所有 Melkor 选项
- Zsh shell 功能正常
- 现在支持预编译 Go 二进制文件
- 现已正确本地化首次运行 Bash 时出现的提示
- /proc/meminfo 现在返回正确的信息
- VFS 现在支持套接字
- /dev 现在装载为 tempfs
- Fifo 现在受支持
- 多核系统现在在 /proc/cpuinfo 中正确显示
- 其他改进以及首次运行期间下载内容时显示的错误消息
- 系统调用改进和漏洞修复。 下面列出了支持的 syscall。
- 其他问题修复和改进

已知问题

- 在某些情况下， 不会正确解析 DriveFs 上的“..”

系统调用支持

下面是在 WSL 中具有某种实现的新的或增强的 syscall 列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一都受支持。

`FCHOWNAT`

`GETEUID`

`GETGID`

`GETRESUID`

`GETXATTR`

`PTRACE`

`SETGID`

`SETGROUPS`

`SETHOSTNAME`

`SETXATTR`

版本 14332

关于版本 14332 的一般 Windows 信息，请访问 Windows 博客。

已修复

- 更优化的 resolv.conf 文件生成，侧重于 DNS 条目的优先级排序
- 在 /mnt 和非/mnt 驱动器之间移动文件和目录时出现问题
- 现在可以使用符号链接创建 Tar 文件
- 在创建实例时添加了默认 /run/lock 目录
- 更新 /dev/null 以返回正确的状态信息
- 首次运行期间下载时出现其他错误
- 系统调用改进和漏洞修复。 下面列出了支持的 syscall。
- 其他 bug 修复和改进

系统调用支持

下面是在 WSL 中具有某种实现的新 syscall。 此列表中的 syscall 至少在一个方案中受支持，但目前可能不支持所有参数。

`READLINKAT`

内部版本 14328

关于版本 14332 的一般 Windows 信息，请访问 Windows 博客。

新功能

- 现在支持 Linux 用户。 在 Windows 上的 Ubuntu 上安装 Bash 将提示创建 Linux 用户。 有关详细信息，请访问 https://aka.ms/wslusers
- 主机名现在设置为 Windows 计算机名称，不再 @localhost
- 有关内部版本 14328 的详细信息，请访问：https://aka.ms/wip14328

已修复

- 非 `/mnt/<drive>` 文件的符号链接改进
  - npm 安装现在有效
  - 现在可以根据此处的说明安装 jdk/jre。
  - 已知问题：符号链接不适用于 Windows 装载。 功能将在后续版本中提供。
- 现在会显示 top 和 htop
- 有关某些安装失败的其他错误消息
- 系统调用改进和漏洞修复。 下面列出了支持的 syscall。
- 其他 bug 修复和改进

系统调用支持

下面是在 WSL 中具有某些实现的系统调用列表。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一定都受支持。

`ACCEPT`

`ACCEPT4`

`ACCESS`

`ALARM`

`ARCH_PRCTL`

`BIND`

`BRK`

`CAPGET`

`CAPSET`

`CHDIR`

`CHMOD`

`CHOWN`

`CLOCK_GETRES`

`CLOCK_GETTIME`

`CLOCK_NANOSLEEP`

`CLONE`

`CLOSE`

`CONNECT`

`CREAT`

`DUP`

`DUP2`

`DUP3`

`EPOLL_CREATE`

`EPOLL_CREATE1`

`EPOLL_CTL`

`EPOLL_WAIT`

`EVENTFD`

`EVENTFD2`

`EXECVE`

`EXIT`

`EXIT_GROUP`

`FACCESSAT`

`FADVISE64`

`FCHDIR`

`FCHMOD`

`FCHMODAT`

`FCHOWN`

`FCHOWNAT`

`FCNTL64`

`FDATASYNC`

`FLOCK`

`FORK`

`FSETXATTR`

`FSTAT64`

`FSTATAT64`

`FSTATFS64`

`FSYNC`

`FTRUNCATE`

`FTRUNCATE64`

`FUTEX`

`GETCPU`

`GETCWD`

`GETDENTS`

`GETDENTS64`

`GETEGID`

`GETEGID16`

`GETEUID`

`GETEUID16`

`GETGID`

`GETGID16`

`GETGROUPS`

`GETPEERNAME`

`GETPGID`

`GETPGRP`

`GETPID`

`GETPPID`

`GETPRIORITY`

`GETRESGID`

`GETRESGID16`

`GETRESUID`

`GETRESUID16`

`GETRLIMIT`

`GETRUSAGE`

`GETSID`

`GETSOCKNAME`

`GETSOCKOPT`

`GETTID`

`GETTIMEOFDAY`

`GETUID`

`GETUID16`

`GETXATTR`

`GET_ROBUST_LIST`

`GET_THREAD_AREA`

`INOTIFY_ADD_WATCH`

`INOTIFY_INIT`

`INOTIFY_RM_WATCH`

`IOCTL`

`IOPRIO_GET`

`IOPRIO_SET`

`KEYCTL`

`KILL`

`LCHOWN`

`LINK`

`LINKAT`

`LISTEN`

`LLSEEK`

`LSEEK`

`LSTAT64`

`MADVISE`

`MKDIR`

`MKDIRAT`

`MKNOD`

`MLOCK`

`MMAP`

`MMAP2`

`MOUNT`

`MPROTECT`

`MREMAP`

`MSYNC`

`MUNLOCK`

`MUNMAP`

`NANOSLEEP`

`NEWUNAME`

`OPEN`

`OPENAT`

`PAUSE`

`PERF_EVENT_OPEN`

`PERSONALITY`

`PIPE`

`PIPE2`

`POLL`

`PPOLL`

`PRCTL`

`PREAD64`

`PROCESS_VM_READV`

`PROCESS_VM_WRITEV`

`PSELECT6`

`PTRACE`

`PWRITE64`

`READ`

`READLINK`

`READV`

`REBOOT`

`RECV`

`RECVFROM`

`RECVMSG`

`RENAME`

`RMDIR`

`RT_SIGACTION`

`RT_SIGPENDING`

`RT_SIGPROCMASK`

`RT_SIGRETURN`

`RT_SIGSUSPEND`

`RT_SIGTIMEDWAIT`

`SCHED_GETAFFINITY`

`SCHED_GETPARAM`

`SCHED_GETSCHEDULER`

`SCHED_GET_PRIORITY_MAX`

`SCHED_GET_PRIORITY_MIN`

`SCHED_SETAFFINITY`

`SCHED_SETPARAM`

`SCHED_SETSCHEDULER`

`SCHED_YIELD`

`SELECT`

`SEND`

`SENDMMSG`

`SENDMSG`

`SENDTO`

`SETDOMAINNAME`

`SETGID`

`SETGROUPS`

`SETHOSTNAME`

`SETITIMER`

`SETPGID`

`SETPRIORITY`

`SETREGID`

`SETRESGID`

`SETRESUID`

`SETREUID`

`SETRLIMIT`

`SETSID`

`SETSOCKOPT`

`SETTIMEOFDAY`

`SETUID`

`SETXATTR`

`SET_ROBUST_LIST`

`SET_THREAD_AREA`

`SET_TID_ADDRESS`

`SHUTDOWN`

`SIGACTION`

`SIGALTSTACK`

`SIGPENDING`

`SIGPROCMASK`

`SIGRETURN`

`SIGSUSPEND`

`SOCKET`

`SOCKETCALL`

`SOCKETPAIR`

`SPLICE`

`STAT64`

`STATFS64`

`SYMLINK`

`SYMLINKAT`

`SYNC`

`SYSINFO`

`TEE`

`TGKILL`

`TIME`

`TIMERFD_CREATE`

`TIMERFD_GETTIME`

`TIMERFD_SETTIME`

`TIMES`

`TKILL`

`TRUNCATE`

`TRUNCATE64`

`UMASK`

`UMOUNT`

`UMOUNT2`

`UNLINK`

`UNLINKAT`

`UNSHARE`

`UTIME`

`UTIMENSAT`

`UTIMES`

`VFORK`

`WAIT4`

`WAITPID`

`WRITE`

`WRITEV`

Last updated on 2025/03/31

适用于 Linux 内核的 Windows 子系统发行说明

我们添加了对 WSL 2 分发版的支持， 这些分发版使用完整的 Linux 内核。 此 Linux 内核是开放源代码，其源代码在 WSL2-Linux-Kernel 存储库中提供。 此 Linux 内核通过 Microsoft 更新传送到您的计算机，并遵循与作为 Windows 映像一部分提供的 Windows Subsystem for Linux 不同的发布计划。

5.15.57.1

发布日期：预发行版 2022/08/02

官方 GitHub 发布链接

- 基于 v5.15 内核系列的 WSL2 内核的初始版本
- 发布 rolling-lts/wsl/5.15.57.1
- 更新到稳定内核版本 v5.15.57
- 在 x86_64 版本中启用 Retbleed 缓解措施
- 启用 nftables 和流量控制
- 启用 VGEM 驱动程序
- 修复自上次发布的 v5.10 WSL2 内核以来的 9p 文件系统回归问题。
- 启用对精度时间协议 （PTP） 时钟设备的支持
- 启用 Landlock Linux 安全模块 （LSM）
  - https://landlock.io/
- 启用杂项控制组 （CGroup）
  - https://www.kernel.org/doc/html/v5.15/admin-guide/cgroup-v2.html#misc
- 禁用对 Ceph 分布式文件系统的支持

5.10.102.1

发布日期：预发行版 2022/05/09

官方 Github 发布链接

- 发布版本：rolling-lts/wsl/5.10.102.1
- 更新到上游稳定内核版本 5.10.102
- 默认情况下禁用非特权 BPF
- 通过将 kernel.unprivileged_bpf_disabled sysctl 设置为 0，可以重新启用它
- 将 Dxgkrnl 版本更新为 2216
- 修复 ioctls[] 的越界数组访问
- 将同步 VM 总线消息实现为“可终止”，以允许终止等待对主机进行同步调用的进程

- 在进程被销毁时刷新设备，以避免在来宾进程被终止时发生死锁问题。

5.10.93.2

发布日期：预发行版 2022/02/08

官方 Github 发布链接

- 发布 rolling-lts/wsl/5.10.93.2
- 更新到上游稳定内核版本 5.10.93
- 启用 CH341 和 CP210X USB 串行驱动程序
- 修复了 README.md 生成说明，以包含对 pahole 的矮人依赖项
- 已将 Dxgkrnl 版本切换到 2111
- 删除了对现有和总 sysmem 分配的限制
- 在进程清理过程中正确刷新设备以终止
- 修复了 d3dkmthk.h 中的 SPDX-License-Identifier

5.10.81.1

发布日期：预发行版 2022/02/01

官方 Github 发布链接

- 发布 rolling-lts/wsl/5.10.81.1
- 更新到上游稳定内核版本 5.10.81
- 通过在 arm64 上启用缺少的选项来统一内核配置
- 启用非架构特定的 ACPI 选项
- 启用与设备映射器 (device-mapper) RAID 相关的选项
- 启用 Btrfs
- 启用 LZO 和 ZSTD 压缩

5.10.74.3

发布日期：预发行版 2021/11/10

官方 Github 发布链接

- 发布版本为 rolling-lts/wsl/5.10.74.3
- 更新到上游稳定内核版本 5.10.74
- 启用 BPF 类型格式（CONFIG_DEBUG_INFO_BTF）以供 eBPF 工具使用（microsoft/WSL#7437）
- 已将 Dxgkrnl 版本更新为 2110

- 为 Dxgkrnl 启用缓冲区共享和同步文件框架（CONFIG_DMA_SHARED_BUFFER、CONFIG_SYNC_FILE）
- 修复 GCC 版本低于 8.1 时导致 Dxgkrnl 构建失败的问题（microsoft/WSL#7558）

5.10.60.1

发布日期：2021/11/02 （预发行版 2021/10/05）

官方 Github 发布链接

- 发布 rolling-lts/wsl/5.10.60.1
- 更新到上游稳定内核版本 5.10.60
- 启用 virtio-pmem，支持 PCI BAR 相对地址
- 在 arm64 的 Hyper-V 目录下启用 vPCI 功能支持
- 启用io_uring支持
- 启用 USB over IP 支持
- 为x86_64启用半虚拟化旋转锁支持
- 更新 dxgkrnl 驱动程序以获取 bug 修复和代码清理
- 为 NFSv4.1 启用 NFS 客户端支持
- 启用 USB 内核配置选项，以便通过 USB 与 Arduino 交互
- 提供特定于 WSL2 的 README.md

5.10.43.3

发布日期：预发行版 2021/07/12

官方 Github 发布链接

- 版本 rolling-lts/wsl/5.10.43.3
- 更新到上游稳定内核版本 5.10.43
- 改进了 dxgkrnl 驱动程序
- Hyper-V 系列上的 arm64 Linux 的新修订版 （v9）
- 始终在 arm64 客户机上使用 Hyper-V 超级调用接口，以支持在所有版本的 Windows 上运行。

5.10.16.3

发布日期：2021/07/20 （预发行版 2021/04/16）

官方 Github 发布链接

- 修复 GH 5324

- 添加了对使用 wsl --mount 的 LUKS 加密磁盘的支持

5.4.91

发布日期：预发行版 2021/02/22

官方 Github 发布链接

5.4.72

发布日期：2021/01/21

官方 Github 发布链接

- 修复 5.4.72 的配置

5.4.51-microsoft-standard

发布日期：预发行版 - 2020/10/22

官方 Github 发布链接。

- 稳定版本 5.4.51

4.19.128-microsoft-standard

发布日期：2020/09/15

官方 Github 发布链接。

- 这是 4.19.128 的稳定版本
- 修复 dxgkrnl 驱动程序 IOCTL 内存损坏

4.19.121-microsoft-standard

发布日期：预发行版

官方 Github 发布链接。

- 驱动程序：hv：vmbus：连接 dxgkrnl
- 添加了对 GPU 计算的支持

4.19.104-microsoft-standard

发布日期：2020/06/09

官方 Github 发布链接。

- 更新 4.19.104 的 WSL 配置

4.19.84-microsoft-standard

发布日期：2019/12/11

官方 Github 发布链接。

- 这是 4.19.84 稳定版

Last updated on 2025/06/10

Microsoft 应用商店中适用于 Linux 的 Windows 子系统发行说明

可以在 Microsoft/WSL GitHub 存储库发布页上找到 Microsoft 应用商店内的 WSL 发行说明。 有关最新更新，请参阅该列表。

已知问题：

- 从零会话启动 Windows Linux 子系统当前不起作用（例如，通过 ssh 连接）。

Last updated on 2025/06/10