告知我们有关下载 PDF 体验的信息。

适用于 Linux 的 Windows 子系统文档

项目 • 2023/03/21

适用于 Linux 的 Windows 子系统 (WSL) 可让开发人员直接在 Windows 上按原样运行 GNU/Linux 环境（包括大多数命令行工具、实用工具和应用程序），且不会产生传统虚拟机或双启动设置开销。

安装 WSL

了解更多

- 什么是适用于 Linux 的 Windows 子系统 (WSL)?
- WSL 2 的新增功能
- 比较 WSL 1 和 WSL 2
- 常见问题

入门

- 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 预览体验计划来试用 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，可以在 Windows“设置”菜单内选择希望从哪个渠道获取预览版。 可以选择：

- 开发频道：更新最新，但稳定性低。
- Beta 版频道：非常适合早期采用者，版本比开发频道的版本更可靠。
- Release Preview 版渠道：在公开发布前，预览下一版 Windows 上的修复和重要功能。

团队博客

- 概述文章以及一系列视频和博客

- 命令行博客（适用于现行版本）
- 适用于 Linux 的 Windows 子系统博客（适用于传统版本）

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 基本信息

1. 什么是适用于 Linux 的 Windows 子系统 (WSL)? | 一个开发人员问题 (0:40)
2. 我是一名 Windows 开发人员。 为什么要使用 WSL? | 一个开发人员问题 (0:58)
3. 我是一名 Linux 开发人员。 为什么要使用 WSL? | 一个开发人员问题 (1:04)
4. 什么是 Linux? | 一个开发人员问题 (1:31)
5. 什么是 Linux 发行版? | 一个开发人员问题 (1:04)
6. WSL 与虚拟机或双启动有何不同? | 一个开发人员问题
7. 为什么要创建适用于 Linux 的 Windows 子系统? | 一个开发人员问题 (1:14)
8. 如何在 WSL 中访问计算机上的文件? | 一个开发人员问题 (1:41)
9. WSL 如何与 Windows 集成? | 一个开发人员问题 (1:34)
10. 如何配置 WSL 发行版以在终端的主目录中启动? | 一个开发人员问题 (0:47)
11. 是否可以使用 WSL 编写脚本? | 一个开发人员问题 (1:04)
12. 为什么要在 Windows 上使用 Linux 工具? | 一个开发人员问题 (1:20)
13. 在 WSL 中，是否可以使用 Microsoft Store 中的发行版以外的发行版? | 一个开发人员问题 (1:03)

WSL 演示

1. WSL2：在适用于 Linux 的 Windows 子系统上更快地进行编码！ | 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 开发工具更新：WSL、终端、包管理器等 | 制表符与空格 (20:46)
7. 使用 WSL 构建 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 深入了解

1. Windows 11 上的 WSL - Craig Loewen 和 Scott Hanselman 的演示 | Windows Wednesday (35:48)
2. WSL 和 Linux 发行版 - Hayden Barnes 和 Kayla Cinnamon | Windows Wednesday (37:00)
3. 使用 Oh My Posh 和 WSL Linux 发行版自定义终端 | Windows Wednesday (33:14)
4. Web 开发人员 Sarah Tamsin 和 Craig Loewen 谈论 Web 开发、内容创建和 WSL | 开发者观点 (12:22)
5. WSL 如何从 Windows 访问 Linux 文件 | 深入了解 (24:59)
6. 适用于 Linux 的 Windows 子系统体系结构：深入了解 | Build 2019 (58:10)

什么是适用于 Linux 的 Windows 子系统?

项目 • 2023/12/05

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引导。 WSL 旨在为希望同时使用 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 发行版，例如 Ubuntu、Debian、Kali 等。 安装 Linux 发行版并从 Microsoft Store 接收自动更新、导入 Microsoft Store 中不可用的 Linux 发行版，或构建你自己的客户 Linux 发行版。
- 将文件存储在独立的 Linux 文件系统中，具体取决于安装的发行版。
- 运行命令行工具，例如 BASH。
- 运行常用的 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：NodeJSJavaScript、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。
- 使用类似于 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
- 使用你的设备 GPU 加速 Linux 上运行的机器学习工作负载。

安装 WSL

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

什么是 WSL 2?

安装 Linux 发行版时，WSL 2 是默认发行版类型。 WSL 2 使用虚拟化技术在轻量级实用工具虚拟机 (VM) 中运行 Linux 内核。 Linux 发行版作为独立的容器在 WSL 2 托管 VM 内运行。 通过 WSL 2 运行的 Linux 发行版将共享同一网络命名空间、设备树（而非 `/dev/pts`）、CPU/内核/内存/交换空间、`/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。

单个 Linux 分发版可以在 WSL 1 或 WSL 2 体系结构中运行。 每个分发版可随时升级或降级，并且你可以并行运行 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 热爱 Linux

详细了解 Microsoft 的 Linux 资源，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、活动和合作。 Microsoft 热爱 Linux！

比较 WSL 版本

项目 • 2023/12/21

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 发行版切换到早期 WSL 1 体系结构的特定场景或例外情况。

比较 WSL 1 和 WSL 2

本指南将比较 WSL 1 和 WSL 2，包括使用 WSL 1 而不是 WSL 2 的例外情况。 WSL 1 和 WSL 2 之间的主要区别在于，在托管 VM 内使用实际的 Linux 内核、支持完整的系统调用兼容性以及跨 Linux 和 Windows 操作系统的性能。 WSL 2 是安装 Linux 发行版时的当前默认版本，它使用最新最好的虚拟化技术在轻量级实用工具虚拟机 (VM) 内运行 Linux 内核。 WSL2 将 Linux 发行版作为托管 VM 内的隔离容器运行。 如果你的发行版当前运行的是 WSL 1，而你想要更新到 WSL 2，请参阅从 WSL 1 更新到 WSL 2。

比较功能

展开表

功能	WSL 1	WSL 2
Windows 和 Linux 之间的集成	✅	✅
启动时间短	✅	✅
与传统虚拟机相比，占用的资源量少	✅	✅
可以与当前版本的 VMware 和 VirtualBox 一起运行	✅	❌
托管 VM	❌	✅
完整的 Linux 内核	❌	✅
完全的系统调用兼容性	❌	✅
跨 OS 文件系统的性能	✅	❌
systemd 支持	❌	✅
IPv6 支持	❌	✅

从上面的比较表中可以看出，WSL 2 架构在几个方面优于 WSL 1，但跨 OS 文件系统的性能除外，对于这种情况，可通过将项目文件存储在与处理项目时运行的工具相同的操作系统上进行处理。

WSL 2 仅在 Windows 11 或 Windows 10 版本 1903、内部版本 18362 或更高版本中可用。 通过按 Windows 徽标键 + R，检查你的 Windows 版本，然后键入 **winver**，选择“确定”。 （或者在 Windows 命令提示符下输入 `ver` 命令）。 你可能需要更新到最新的 Windows 版本。 低于 14393 的版本完全不支持 WSL。

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 中提供了有关 IPv6 支持的详细信息。

> **备注**
>
> WSL 2 将与 VMware 15.5.5+ 配合使用，虽然 VirtualBox 6+ 声明支持 WSL，但仍存在重大挑战，导致其不受支持。 有关详细信息，请参阅**常见问题解答**。

WSL 2 中的新增功能

WSL 2 是对基础体系结构的一次重大改造，它使用虚拟化技术和 Linux 内核来实现其新功能。 此更新的主要目标是提高文件系统性能和添加完全的系统调用兼容性。

- WSL 2 系统要求
- 将 Linux 发行版的版本从 WSL 1 设置为 WSL 2
- 有关 WSL 2 的常见问题解答

WSL 2 体系结构

传统的 VM 体验可能启动速度慢，是独立的，消耗大量资源，需要你花费时间进行管理。 WSL 2 没有这些属性。

WSL 2 有 WSL 1 的优点，包括 Windows 和 Linux 之间的无缝集成，启动时间短，资源占用量少，并且无需 VM 配置或管理。 虽然 WSL 2 确实使用 VM，但 VM 是在幕后管理和运行的，因此你将具有与 WSL 1 相同的用户体验。

完整的 Linux 内核

WSL 2 中的 Linux 内核是 Microsoft 根据最新的稳定版分支（基于 kernel.org 上提供的源代码）构建的。此内核已专门针对 WSL 2 进行了调整，针对大小和性能进行了优化，以便在 Windows 上提供良好的 Linux 体验。 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功能，而无需自行管理它。

WSL 2 Linux 内核是开源的。 如果你想要了解详细信息，请查看由构建该内核的团队撰写的博客文章随 Windows 一起提供 Linux 内核。

有关详细信息，请参阅适用于 Linux 的 Windows 子系统内核发行说明

提升了文件 IO 性能

如果使用 WSL 2，文件密集型操作（如 git 克隆、npm 安装、apt 更新、apt 升级等）的速度都明显更快。

实际的速度提升将取决于你运行的应用以及它与文件系统的交互方式。 在对压缩的 tarball 进行解包时，WSL 2 的初始版本的运行速度比 WSL 1 快达 20 倍，在各种项目上使用 git 克隆、npm 安装和 cmake 时，大约快 2-5 倍。

完全的系统调用兼容性

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。 虽然 WSL 1 使用的是由 WSL 团队构建的转换层，但 WSL 2 包括了自己的 Linux 内核，具有完全的系统调用兼容性。 优点包括：

- 可以在 WSL 内部运行的一组全新应用，例如 **Docker** 等。
- 对 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）。

例外情况（使用 WSL 1 而不是 WSL 2）

我们建议使用 WSL 2，因为它提供更快的性能和100% 的系统调用兼容性。 但是，在某些特定情况下，你可能会更倾向于使用 WSL 1。 在以下情况下，请考虑使用 WSL 1：

- 你的项目文件必须存储在 Windows 文件系统中。 WSL 1 可以更快地访问从 Windows 装载的文件。
  - 如果你将使用 WSL Linux 分发版来访问 Windows 文件系统上的项目文件，并且这些文件无法存储在 Linux 文件系统上，那么，通过使用 WSL 1，你将跨 OS 文件系统实现更快的性能。
- 一个项目要求对相同的文件使用 Windows 和 Linux 工具进行交叉编译。
  - 在 WSL 1 中，跨 Windows 和 Linux 操作系统的文件性能比 WSL 2 中更快，因此如果要使用 Windows 应用程序来访问 Linux 文件，则目前通过 WSL 1 可实现更快的性能。
- 你的项目需要访问串行端口或 USB 设备。 但是，现在可通过 USBIPD-WIN 项目为 WSL 2 提供 USB 设备支持。 有关设置步骤，请参阅连接 USB 设备。
- WSL 2 不支持访问串行端口。 有关详细信息，请参阅常见问题解答或 WSL GitHub 存储库中有关串行支持的问题。
- 有严格的内存要求

    - WSL 2 的内存使用量会随使用而缩放。 当进程释放内存时，这会自动返回到 Windows。 但从现在开始，在关闭 WSL 实例前，WSL 2 还不会将内存中缓存的页面释放回 Windows。 如果你有长时间运行的 WSL 会话或访问非常大量的文件，此缓存可能会耗尽 Windows 内存。 我们通过 WSL GitHub 存储库问题 4166 跟踪工作以改善此体验。
- 使用 VirtualBox 的用户：请务必使用 VirtualBox 和 WSL 2 的最新版本。 请参阅相关的常见问题解答。
- 如果依赖 Linux 发行版在与主机相同的网络中拥有 IP 地址，则可能需要设置一种替代方法来运行 WSL 2。 WSL 2 作为 hyper-v 虚拟机运行。 这是对 WSL 1 中使用的桥接网络适配器的更改，这意味着 WSL 2 使用网络地址转换 (NAT) 服务作为其虚拟网络，而不是将其桥接到主机网络接口卡 (NIC)，从而生成唯一的将在重启时更改的 IP 地址。 要详细了解将 WSL 2 服务的 TCP 端口转发到主机 OS 的问题和缓解措施，请参阅 WSL GitHub 存储库问题 4150，NIC 桥接模式（TCP 缓解措施）。

> **备注**
>
> 请考虑尝试 VS Code **远程 WSL 扩展**，以便使你不仅能够使用 Linux 命令行工具将项目文件存储在 Linux 文件系统上，而且还可以使用 Windows 上的 VS Code 在 Internet 浏览器中创作、编辑、调试或运行项目，而不会造成任何与跨 Linux 和 Windows 文件系统工作相关联的性能下降。 **了解详细信息。**

Microsoft Store 中的 WSL

WSL 已将更新功能从 Windows OS 映像提取到一个包中，该包可通过 Microsoft Store 获得。 这意味着一旦更新和服务可用就会进行快速更新并提供服务，而无需等待 Windows 操作系统的更新。

WSL 最初作为可选组件包含在 Windows 操作系统中，需要启用该组件才能安装 Linux 发行版。 Microsoft Store 中的 WSL 具有相同的用户体验，并且是相同的产品，但作为商店包而不是整个 OS 更新接收更新和服务。 从 Windows 版本 19044 或更高版本开始，运行 `wsl.exe --install` 命令将从 Microsoft Store 安装 WSL 服务更新。 （请参阅宣布推出此更新的博客文章）。 如果你已经在使用 WSL，则可以更新以确保通过运行 `wsl.exe --update` 从商店接收最新的 WSL 功能和服务。

WSL 的基本命令

项目 • 2023/12/05

以下 WSL 命令以 PowerShell 或 Windows 命令提示符支持的格式列出。 若要通过 Bash/Linux 发行版命令行运行这些命令，必须将 `wsl` 替换为 `wsl.exe`。 若要获取完整的命令列表，请运行 `wsl --help`。 如果尚未执行此操作，我们建议更新到从 Microsoft Store 安装的 WSL 版本，以便尽快在 WSL 更新可用时接收更新。 （详细了解如何通过 Microsoft Store 安装 WSL）。

安装

PowerShell

```
wsl --install
```

安装 WSL 和 Linux 的默认 Ubuntu 发行版。 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 发行版。 若要获取发行版名称的有效列表，请运行 `wsl --list --online`。

选项包括：

- `--distribution`：指定要安装的 Linux 发行版。 可以通过运行 `wsl --list --online` 来查找可用的发行版。
- `--no-launch`：安装 Linux 发行版，但不自动启动它。
- `--web-download`：通过联机渠道安装，而不是使用 Microsoft Store 安装。

未安装 WSL 时，选项包括：

- `--inbox`：使用 Windows 组件（而不是 Microsoft Store）安装 WSL。 *（WSL 更新将通过 Windows 更新接收，而不是通过 Microsoft Store 中推送的可用更新来接收）*。
- `--enable-wsl1`：在安装 Microsoft Store 版本的 WSL 的过程中也启用“适用于 Linux 的 Windows 子系统”可选组件，从而启用 WSL 1。
- `--no-distribution`：安装 WSL 时不安装发行版。

> **备注**
>
> 如果在 Windows 10 或更低版本上运行 WSL，可能需要在 `--install` 命令中包含 `-d` 标志以指定发行版：`wsl --install -d <distribution name>`。

列出可用的 Linux 发行版

PowerShell

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

查看可通过在线商店获得的 Linux 发行版列表。 此命令也可输入为：`wsl -l -o`。

列出已安装的 Linux 发行版

PowerShell

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

查看安装在 Windows 计算机上的 Linux 发行版列表，其中包括状态（发行版是正在运行还是已停止）和运行发行版的 WSL 版本（WSL 1 或 WSL 2）。 比较 WSL 1 和 WSL 2。 此命令也可输入为：`wsl -l -v`。 可与 list 命令一起使用的其他选项包括：`--all`（列出所有发行版）、`--running`（仅列出当前正在运行的发行版）或 `--quiet`（仅显示发行版名称）。

将 WSL 版本设置为 1 或 2

PowerShell

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

若要指定运行 Linux 发行版的 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、内部版本 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，并且可能会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，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>
```

若要将默认版本设置为 WSL 1 或 WSL 2，请将 `<Version>` 替换为数字 1 或 2，表示对于安装新的 Linux 发行版，你希望默认使用哪个版本的 WSL。 例如，`wsl --set-default-version 2`。 比较 WSL 1 和 WSL 2。 WSL 2 仅在 Windows 11 或 Windows 10 版本 1903、内部版本 18362 或更高版本中可用。

设置默认 Linux 发行版

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>` 替换为你首选的 Linux 发行版的名称。

将目录更改为主页

PowerShell

```
wsl ~
```

`~` 可与 wsl 一起使用，以在用户的主目录中启动。 若要在 WSL 命令提示符中从任何目录跳回到主目录，可使用命令 `cd ~`。

通过 PowerShell 或 CMD 运行特定的 Linux 发行版

PowerShell

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

若要通过特定用户运行特定 Linux 发行版，请将 `<Distribution Name>` 替换为你首选的 Linux 发行版的名称（例如 Debian），将 `<User Name>` 替换为现有用户的名称（例如 root）。 如果 WSL 发行版中不存在该用户，你将会收到一个错误。 若要输出当前用户名，请使用 `whoami` 命令。

更新 WSL

PowerShell

```
wsl --update
```

将 WSL 版本更新到最新版本。 选项包括：

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

检查 WSL 状态

PowerShell

```
wsl --status
```

查看有关 WSL 配置的常规信息，例如默认发行版类型、默认发行版和内核版本。

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
```

查看 WSL 中可用的选项和命令列表。

以特定用户的身份运行

PowerShell

```
wsl -u <Username>`, `wsl --user <Username>
```

若要以指定用户身份运行 WSL，请将 `<Username>` 替换为 WSL 发行版中存在的用户名。

更改发行版的默认用户

PowerShell

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

更改用于发行版登录的默认用户。 用户必须已经存在于发行版中才能成为默认用户。

例如：`ubuntu config --default-user johndoe` 会将 Ubuntu 发行版的默认用户更改为“johndoe”用户。

> ⓘ **备注**
>
> 如果在确定发行版名称时遇到问题，请使用命令 `wsl -l`。

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

关闭

PowerShell

```
wsl --shutdown
```

立即终止所有正在运行的发行版和 WSL 2 轻量级实用工具虚拟机。 在需要重启 WSL 2 虚拟机环境的情形下，例如更改内存使用限制或更改 .wslconfig 文件，可能必须使用此命令。

Terminate

PowerShell

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

若要终止指定的发行版或阻止其运行，请将 `<Distribution Name>` 替换为目标发行版的名称。

标识 IP 地址

- `wsl hostname -i` 标识通过 WSL 2 安装的 Linux 分发版 IP 地址（WSL 2 VM 地址）
- `cat /etc/resolv.conf` 表示从 WSL 2 看到的 WINDOWS 计算机的 IP 地址 (WSL 2 VM)

导入和导出发行版

PowerShell

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

PowerShell

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

将指定 tar 文件导入和导出为新的发行版。 在标准输入中，文件名可以是 -。 选项包括：

- `--vhd`：指定导入/导出分发应为 .vhdx 文件而不是 tar 文件（这仅在使用 WSL 2 的情况下受支持）
- `--version`： （仅导入）指定将发行版导入为 WSL 1 还是 WSL 2 发行版

就地导入发行版

PowerShell

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

将指定的 .vhdx 文件导入为新的发行版。 虚拟硬盘必须采用 ext4 文件系统类型格式。

注销或卸载 Linux 发行版

尽管可以通过 Microsoft Store 安装 Linux 发行版，但无法通过 Store 将其卸载。

注销并卸载 WSL 发行版：

PowerShell

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

如果将 `<DistributionName>` 替换为目标 Linux 发行版的名称，则将从 WSL 取消注册该发行版，以便可以重新安装或清理它。 **警告：**取消注册后，与该分发版关联的所有数据、设置和软件将永久丢失。 从 Store 重新安装会安装分发版的干净副本。 例如：`wsl --unregister Ubuntu` 将从可用于 WSL 的发行版中删除 Ubuntu。 运行 `wsl --list` 将会显示它不再列出。

还可以像卸载任何其他应用商店应用程序一样卸载 Windows 计算机上的 Linux 发行版应用。 若要重新安装，请在 Microsoft Store 中找到该发行版，然后选择“启动”。

装载磁盘或设备

PowerShell

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

通过将 `<DiskPath>` 替换为物理磁盘所在的目录\文件路径，在所有 WSL2 发行版中附加和装载该磁盘。 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，但不进行装载。
- `--type <Filesystem>`：装载磁盘时使用的文件系统类型默认为 ext4（如果未指定）。 此命令也可输入为：`wsl --mount -t <Filesystem>`。可以使用 `blkid <BlockDevice>` 命令检测文件系统类型，例如：`blkid <dev/sdb1>`。
- `--partition <Partition Number>`：要装载的分区的索引号默认为整个磁盘（如果未指定）。
- `--options <MountOptions>`：装载磁盘时，可以包括一些特定于文件系统的选项。例如，`wsl --mount -o "data-ordered"` 或 `wsl --mount -o "data=writeback` 之类的 ext4 装载选项。 但是，目前仅支持特定于文件系统的选项。 不支持通用选项，例如 `ro`、`rw` 或 `noatime`。

> **备注**
>
> 如果你正在运行 32 位进程来访问 wsl.exe（一种 64 位工具），那么你可能需要按如下方式运行此命令：`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已装载的磁盘。

已弃用的 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 安装的 Linux 发行版的原始 wsl 语法，但已替换为 `wsl` 或 `wsl.exe` 命令语法。

如何使用 WSL 在 Windows 上安装 Linux

项目 • 2023/08/28

开发人员可以在 Windows 计算机上同时访问 Windows 和 Linux 的强大功能。 通过适用于 Linux 的 Windows 子系统 (WSL)，开发人员可以安装 Linux 发行版（例如 Ubuntu、OpenSUSE、Kali、Debian、Arch Linux 等），并直接在 Windows 上使用 Linux 应用程序、实用程序和 Bash 命令行工具，不用进行任何修改，也无需承担传统虚拟机或双启动设置的费用。

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。 如果使用的是更早的版本，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 所需的一切内容。 在管理员模式下打开 PowerShell 或 Windows 命令提示符，方法是右键单击并选择“以管理员身份运行”，输入 wsl --install 命令，然后重启计算机。

PowerShell

```
wsl --install
```

此命令将启用运行 WSL 并安装 Linux 的 Ubuntu 发行版所需的功能。 （可以更改此默认发行版）。

如果你运行的是旧版，或只是不想使用 install 命令并希望获得分步指引，请参阅旧版 WSL 手动安装步骤。

首次启动新安装的 Linux 发行版时，将打开一个控制台窗口，要求你等待将文件解压缩并存储到计算机上。 未来的所有启动时间应不到一秒。

> **备注**
>
> 上述命令仅在完全未安装 WSL 时才有效，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。 若要卸载 WSL，请参阅**卸载旧版 WSL** 或**注销或卸载 Linux 发行版**。

更改默认安装的 Linux 发行版

默认情况下，安装的 Linux 分发版为 Ubuntu。 可以使用 `-d` 标志进行更改。

- 若要更改安装的发行版，请输入：`wsl --install -d <Distribution Name>`。 将 `<Distribution Name>` 替换为要安装的发行版的名称。
- 若要查看可通过在线商店下载的可用 Linux 发行版列表，请输入：`wsl --list --online` 或 `wsl -l -o`。
- 若要在初始安装后安装其他 Linux 发行版，还可使用命令：`wsl --install -d <Distribution Name>`。

> **提示**
>
> 如果要通过 Linux/Bash 命令行（而不是通过 PowerShell 或命令提示符）安装其他发行版，必须在命令中使用 .exe：`wsl.exe --install -d <Distribution Name>` 或若要列出可用发行版，则使用：`wsl.exe -l -o`。

如果在安装过程中遇到问题，请查看疑难解答指南的安装部分。

要安装未列为可用版本的 Linux 发行版，可使用 TAR 文件导入任何 Linux 发行版。 在某些情况下，与 Arch Linux 一样，也可使用 `.appx` 文件进行安装。 还可以通过 WSL 创建自己的自定义 Linux 发行版，以供使用。

设置 Linux 用户信息

安装 WSL 后，需要为新安装的 Linux 发行版创建用户帐户和密码。 请参阅设置 WSL 开发环境的最佳做法指南来了解详细信息。

设置和最佳做法

建议遵循设置 WSL 开发环境的最佳做法这一指南，逐步了解以下操作：为已安装的 Linux 发行版设置用户名和密码、使用基本的 WSL 命令、安装和自定义 Windows 终端、针对 Git 版本控制进行设置、使用 VS Code 远程服务器进行代码编辑和调试、文件存储的最佳做法、设置数据库、装载外部驱动器、设置 GPU 加速等。

检查正在运行的 WSL 版本

可列出已安装的 Linux 发行版，并通过在 PowerShell 或 Windows 命令提示符中输入以下命令来检查每个发行版的 WSL 版本：`wsl -l -v`。

要在安装新的 Linux 发行版时将默认版本设置为 WSL 1 或 WSL 2，请使用命令 `wsl --set-default-version <Version#>`，将 `<Version#>` 替换为 1 或 2。

要设置与 `wsl` 命令一起使用的默认 Linux 发行版，请输入 `wsl -s <DistributionName>` 或 `wsl --setdefault <DistributionName>`，将 `<DistributionName>` 替换为要使用的 Linux 发行版的名称。 例如，从 PowerShell/CMD 输入 `wsl -s Debian`，将默认发行版设置为 Debian。 现在从 Powershell 运行 `wsl npm init` 将在 Debian 中运行 `npm init` 命令。

要在 PowerShell 或 Windows 命令提示符下运行特定的 WSL 发行版而不更改默认发行版，请使用命令 `wsl -d <DistributionName>`，将 `<DistributionName>` 替换为要使用的发行版的名称。

有关详细信息，请参阅 WSL 的基本命令指南。

将版本从 WSL 1 升级到 WSL 2

使用 `wsl --install` 命令安装的新 Linux 安装将默认设置为 WSL 2。

`wsl --set-version` 命令可用于从 WSL 2 降级到 WSL 1，或将以前安装的 Linux 发行版从 WSL 1 更新到 WSL 2。

要查看 Linux 发行版是设置为 WSL 1 还是 WSL 2，请使用命令 `wsl -l -v`。

要更改版本，请使用 `wsl --set-version <distro name> 2` 命令将 `<distro name>` 替换为要更新的 Linux 发行版的名称。 例如，`wsl --set-version Ubuntu-20.04 2` 会将 Ubuntu 20.04 发行版设置为使用 WSL 2。

如果在 `wsl --install` 命令可用之前手动安装了 WSL，则可能还需要启用 WSL 2 所使用的虚拟机可选组件并安装内核包（如果尚未这样做）。

如需了解更多信息，请参阅 WSL 命令参考以获取 WSL 命令列表，并参阅比较 WSL 1 和 WSL 2，获取有关用于你的工作场景的指南，或参阅设置 WSL 开发环境的最佳做法，了解有关使用 WSL 设置良好开发工作流的一般指南。

使用 WSL 运行多个 Linux 发行版的方法

WSL 支持运行想要安装的任意数量的不同 Linux 发行版。 这可能包括从 Microsoft Store 选择发行版、导入自定义发行版或生成自己的自定义发行版。

安装后，有几种方法可以运行 Linux 发行版：

- 安装 Windows 终端（推荐）：使用 Windows 终端支持你想要安装的任意数量的命令行，并允许你在多个标签或窗口窗格中打开它们并在多个 Linux 发行版或其他命

令行（PowerShell、命令提示符、PowerShell、Azure CLI 等）之间快速切换。 可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键来完全自定义终端。 了解详细信息。

- 通过访问 Windows“开始”菜单并键入已安装的发行版的名称，可以直接打开 Linux 发行版。 例如：“Ubuntu”。 这会在其自己的控制台窗口中打开 Ubuntu。
- 在 Windows 命令提示符或 PowerShell 中，可以输入已安装的发行版的名称。 例如：`ubuntu`
- 在 Windows 命令提示符或 PowerShell 中，可以在当前命令行中打开默认的 Linux 发行版，方法是输入：`wsl.exe`。
- 在 Windows 命令提示符或 PowerShell 中，可以在当前命令行中使用默认的 Linux 发行版，而无需输入新的发行版名称，方法是输入：`wsl [command]`。 将 `[command]` 替换为 WSL 命令，例如，替换为 `wsl -l -v` 以列出已安装的发行版，或 `wsl pwd` 以查看当前目录路径在 WSL 中的装载位置。 在 PowerShell 中，命令 `get-date` 将提供 Windows 文件系统中的日期，而 `wsl date` 将提供 Linux 文件系统中的日期。

选择的方法应取决于所执行的操作。 如果已在 Windows 提示符或 PowerShell 窗口中打开 WSL 命令行并想退出，请输入命令：`exit`。

想要试用最新的 WSL 预览功能?

加入 Windows 预览体验计划，试用 WSL 的最新功能或更新。 加入 Windows 预览体验成员后，可以在 Windows 设置菜单内选择希望从哪个频道接收预览版，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：

- 开发频道：更新最新，但稳定性低。
- Beta 版频道：非常适合早期采用者，版本比开发频道的版本更可靠。
- 发布预览频道：在下一版 Windows 即将公开发布之前的预览版修补程序和重要功能。

其他资源

- Windows 命令行博客：现可在 Windows 10 版本 2004 及更高版本中使用单个命令安装 WSL

旧版 WSL 的手动安装步骤

项目 • 2023/12/05

为简单起见，通常建议使用 wsl --install 安装适用于 Linux 的 Windows 子系统，但如果运行的是旧版 Windows，则可能不支持这种方式。 下面介绍了手动安装步骤。 如果在安装过程中遇到问题，请查看疑难解答指南的安装部分。

步骤 1 - 启用适用于 Linux 的 Windows 子系统

需要先启用“适用于 Linux 的 Windows 子系统”可选功能，然后才能在 Windows 上安装 Linux 分发。

以管理员身份打开 PowerShell（“开始”菜单 >“PowerShell” >单击右键 >“以管理员身份运行”），然后输入以下命令：

PowerShell

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

建议现在转到步骤 #2，更新到 WSL 2，但如果只想安装 WSL 1，现在可以重新启动计算机，然后继续执行步骤 6 - 安装所选的 Linux 发行版。 若要更新到 WSL 2，请等待重新启动计算机，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，需要运行 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。

> **备注**
>
> 低于 18362 的版本不支持 WSL 2。 使用 **Windows Update 助手** 更新 Windows 版本。 Windows 版本 1903 支持也仅适用于 x64 系统。 如果使用的是 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，选择 Windows 徽标键 + R，然后键入“winver”，选择“确定”。 更新到“设置”菜单中的最新 Windows 版本。

> **备注**
>
> 如果运行的是 Windows 10 版本1903 或 1909，请在 Windows 菜单中打开“设置”，导航到“更新和安全性”，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。 阅读详细信息：**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
```

重新启动计算机，以完成 WSL 安装并更新到 WSL 2。

步骤 4 - 下载 Linux 内核更新包

Linux 内核更新包会安装最新版本的 WSL 2 Linux 内核，以便在 Windows 操作系统映像中运行 WSL。 （若要运行 Microsoft Store 中的 WSL 并更频繁地推送更新，请使用 `wsl.exe --install` 或 `wsl.exe --update`。）

1. 下载最新包：

   - 适用于 x64 计算机的 WSL2 Linux 内核更新包

   > **备注**
   >
   > 如果使用的是 ARM64 计算机，请下载 **ARM64 包**。 如果不确定自己计算机的类型，请打开命令提示符或 PowerShell，并输入：`systeminfo | find "System Type"`。 **Caveat:** 在非英文版 Windows 上，你可能必须修改搜索文

本，对“System Type”字符串进行翻译。 你可能还需要对引号进行转义来用于 find 命令。 例如，在德语版中使用 `systeminfo | find '"Systemtyp"'`。

2. 运行上一步中下载的更新包。 （双击以运行 - 系统将提示你提供提升的权限，选择“是”以批准此安装。）

安装完成后，请继续执行下一步 - 在安装新的 Linux 分发时，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。）

> ⓘ **备注**
>
> 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

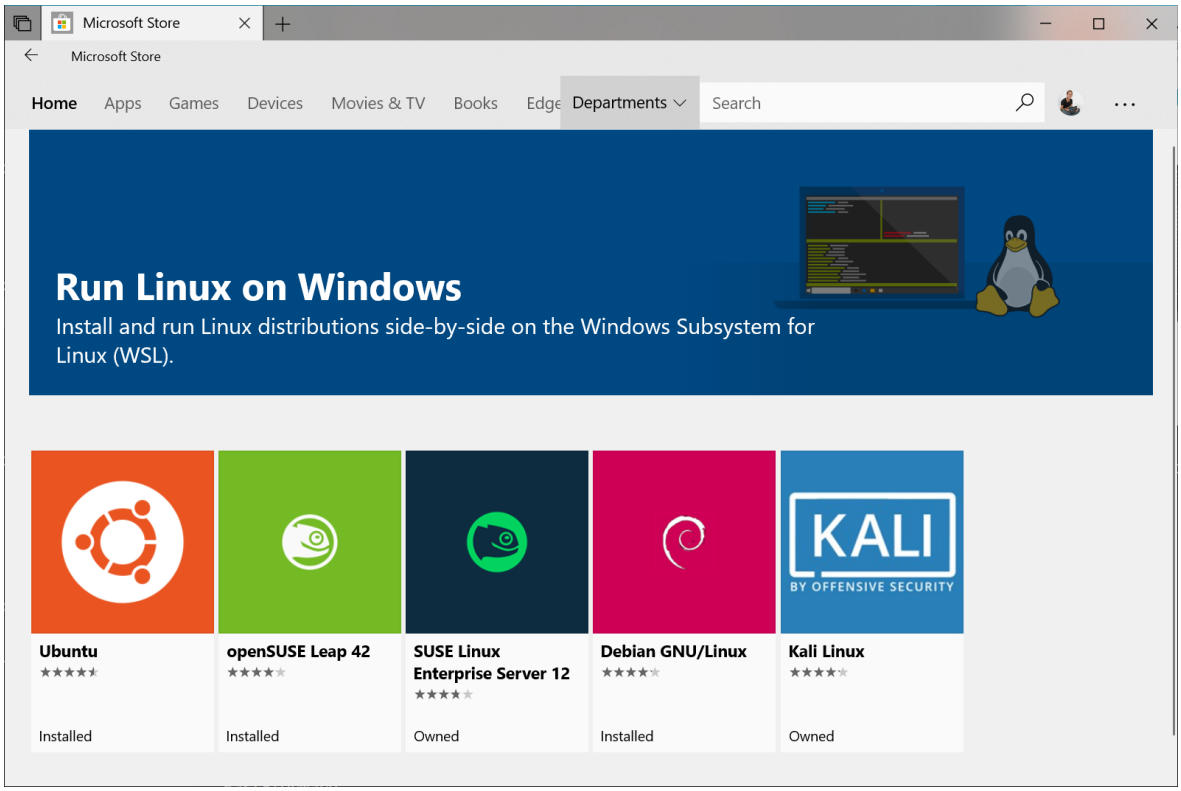
打开 PowerShell，然后在安装新的 Linux 发行版时运行以下命令，将 WSL 2 设置为默认版本：

PowerShell

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

步骤 6 - 安装所选的 Linux 分发

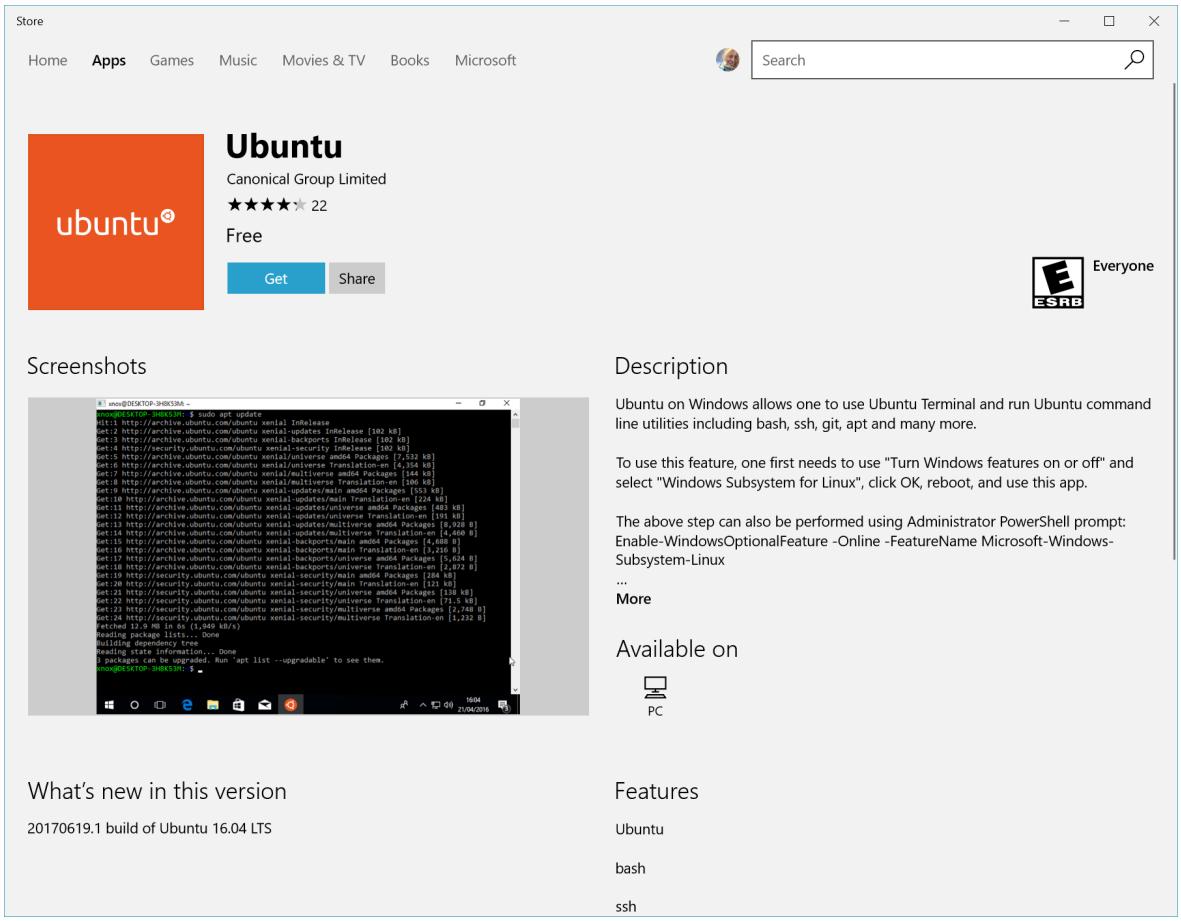
1. 打开 Microsoft Store，并选择你偏好的 Linux 分发版。



单击以下链接会打开每个分发版的 Microsoft Store 页面：

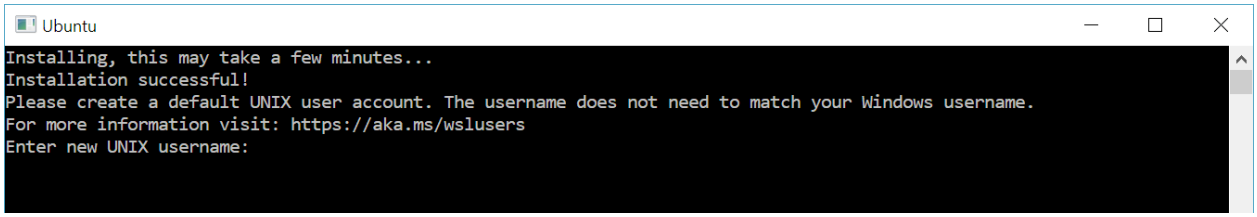
- Ubuntu 18.04 LTS
- Ubuntu 20.04 LTS
- Ubuntu 22.04 LTS
- openSUSE Leap 15.1
- SUSE Linux Enterprise Server 12 SP5
- SUSE Linux Enterprise Server 15 SP1
- Kali Linux
- Debian GNU/Linux
- Fedora Remix for WSL
- Pengwin
- Pengwin Enterprise
- Alpine WSL
- Raft（免费试用版）
- Alma Linux

2. 在分发版的页面中，选择“获取”。



首次启动新安装的 Linux 分发版时，将打开一个控制台窗口，系统会要求你等待一分钟或两分钟，以便文件解压缩并存储到电脑上。 未来的所有启动时间应不到一秒。

然后，需要为新的 Linux 分发版创建用户帐户和密码。



祝贺你！ 现已成功安装并设置了与 Windows 操作系统完全集成的 Linux 分发！

排查安装问题

如果在安装过程中遇到问题，请查看疑难解答指南的安装部分。

下载发行版

在某些情况下，你可能无法（或不想）使用 Microsoft Store 安装 WSL Linux 发行版。 你可能正在运行不支持 Microsoft Store 的 Windows Server 或长期服务 (LTSC) 桌面操作系统 SKU，或者你的公司网络策略和/或管理员不允许在你的环境中使用 Microsoft Store。 在这些情况下，虽然 WSL 本身可用，但你可能需要直接下载 Linux 发行版。

如果 Microsoft Store 应用不可用，可使用以下链接来下载并手动安装 Linux 发行版：

- Ubuntu
- Ubuntu 22.04 LTS
- Ubuntu 20.04
- Ubuntu 20.04 ARM
- Ubuntu 18.04
- Ubuntu 18.04 ARM
- Ubuntu 16.04
- Debian GNU/Linux
- Kali Linux
- SUSE Linux Enterprise Server 12
- SUSE Linux Enterprise Server 15 SP2
- SUSE Linux Enterprise Server 15 SP3
- openSUSE Tumbleweed
- openSUSE Leap 15.3
- openSUSE Leap 15.2
- Oracle Linux 8.5
- Oracle Linux 7.9
- Fedora Remix for WSL

这将导致 `<distro>.appx` 包下载到你选择的文件夹。

如果愿意，你也可通过命令行下载首选的发行版，可将 PowerShell 与 Invoke-WebRequest cmdlet 一起使用。 例如，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'` 来关闭进度栏

你还可以选择使用 curl 命令行实用程序来下载。 使用 curl 下载 Ubuntu 20.04：

控制台

```
curl.exe -L -o ubuntu-2004.appx https://aka.ms/wslubuntu2004
```

在本示例中，将执行 `curl.exe`（而不仅仅是 `curl`），以确保在 PowerShell 中调用真正的 curl 可执行文件，而不是调用 Invoke WebRequest 的 PowerShell curl 别名。

下载了发行版后，导航到包含下载内容的文件夹，并在该目录中运行以下命令，其中 `app-name` 是 Linux 发行版 .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 文件开始运行新发行版。 （命令 `wsl -l` 不会在此步骤完成之前显示发行版已安装）。

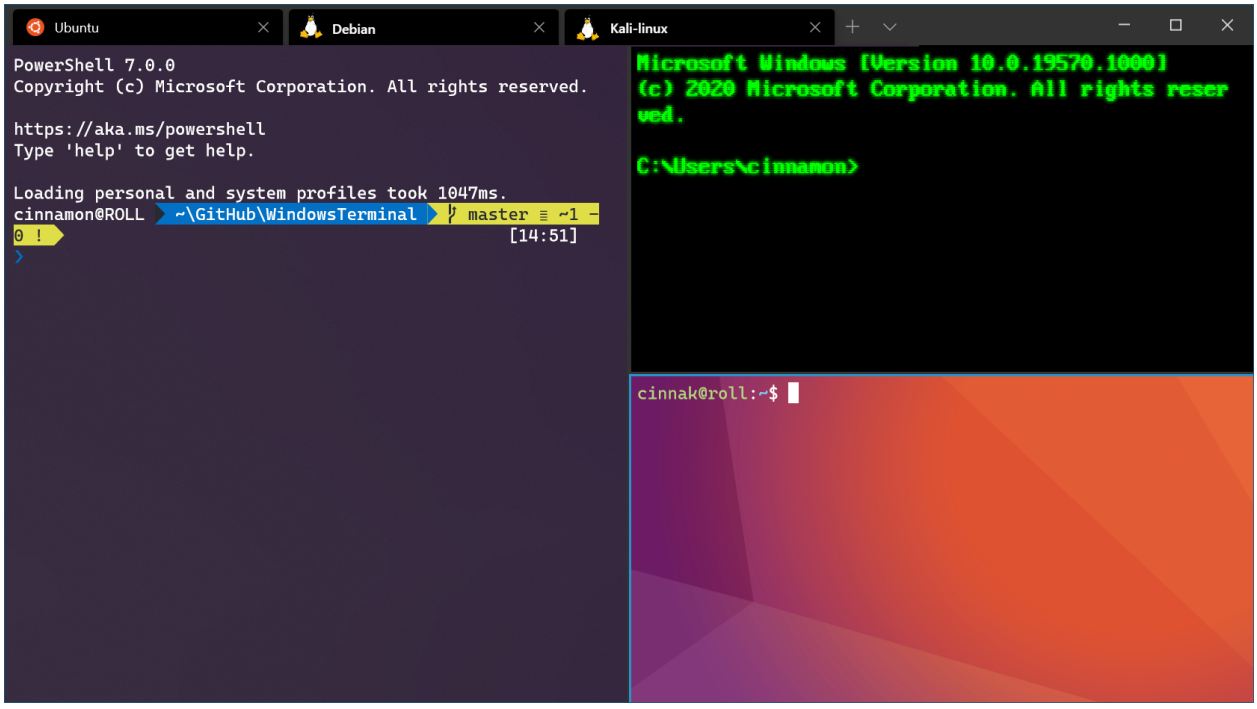
如果你使用 Windows Server，或在运行上述命令时遇到问题，可在 Windows Server 文档页上找到备用安装说明，以通过将 `.appx` 文件更改为 zip 文件来安装该文件。

安装了发行版后，请按照说明为新的 Linux 发行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，以显示多个 Linux 发行版或其他命令行（PowerShell、命令提示符、Azure CLI 等）并在它们之间快速切换。 可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键来完全自定义终端。 了解详细信息。

安装 Windows 终端。



Windows Server 安装指南

项目 • 2023/03/21

适用于 Linux (WSL) 的 Windows 子系统可供在 Windows Server 2019（版本 1709）和更高版本上安装。 本指南将指导你完成在计算机上启用 WSL 的步骤。

在 Windows Server 2022 上安装 WSL

Windows Server 2022 现在使用命令支持简单的 WSL 安装：

```
Bash

wsl --install
```

现在，可以在管理员 PowerShell 或 Windows 命令提示符中输入此命令，然后重启计算机来安装在 Windows Server 2022 上运行 WSL 所需的全部内容。

此命令将启用所需的可选组件，下载最新的 Linux 内核，将 WSL 2 设置为默认值，并安装 Linux 发行版（默认安装 Ubuntu）。

有关如何执行以下操作的详细信息，请参阅标准 WSL 文档：

- 更改默认安装的 Linux 发行版。
- 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。

在以前版本的 Windows Server 上安装 WSL

若要在 Windows Server 2019 （版本 1709+） 上安装 WSL，可以按照下面的手动安装步骤进行操作。

启用适用于 Linux 的 Windows 子系统

必须启用“适用于 Linux 的 Windows 子系统”可选功能并重启，然后才能在 Windows 上运行 Linux 发行版。

以管理员身份打开 PowerShell 并运行：

PowerShell

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

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版部分。

提取并安装 Linux 分发版

下载 Linux 分发版后，若要提取其内容并进行手动安装，请执行以下步骤：

1. 使用 PowerShell 提取 `<DistributionName>.appx` 包的内容：

   PowerShell

   ```
   Rename-Item .\Ubuntu.appx .\Ubuntu.zip
   Expand-Archive .\Ubuntu.zip .\Ubuntu
   ```

2. 下载了发行版后，导航到包含下载内容的文件夹，并在该目录中运行以下命令，其中 `app-name` 是 Linux 发行版 .appx 文件的名称。

Powershell

```
Add-AppxPackage .\app_name.appx
```

> ⊗ **注意**
>
> **安装失败并出现错误 0x8007007e**：如果收到此错误，则表明系统不支持 WSL。 请确保运行的是 Windows 版本 16215 或更高版本。 **检查内部版本**。 另外，请进行检查以**确认 WSL 已启用**，并且在启用此功能后重新启动了计算机。

3. 使用 PowerShell 将 Linux 发行版路径添加到 Windows 环境路径（在本例中为 `C:\Users\Administrator\Ubuntu`）：

PowerShell

```
$userenv = [System.Environment]::GetEnvironmentVariable("Path", "User")
[System.Environment]::SetEnvironmentVariable("PATH", $userenv +
";C:\Users\Administrator\Ubuntu", "User")
```

现在，可以通过键入 `<DistributionName>.exe` 从任何路径启动你的分发版。 例如：`ubuntu.exe`。

安装完成后，可为新的 Linux 发行版创建用户帐户和密码。

设置 WSL 开发环境

项目 • 2023/11/20

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。

开始使用

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+）。 若要检查 Windows 版本及内部版本号，选择 Windows 徽标键 + R，然后键入“winver”，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：

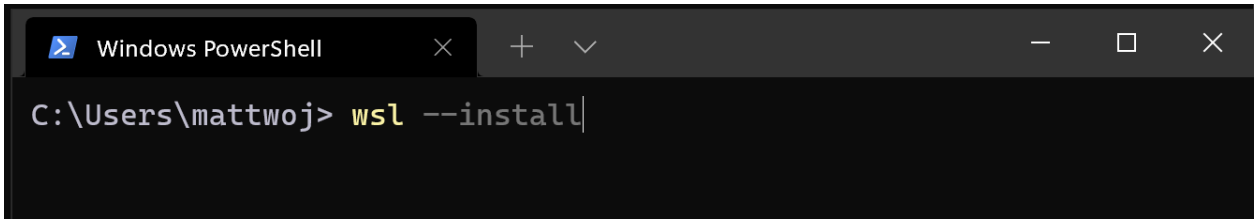
PowerShell

```
wsl --install
```

--install 命令执行以下操作：

- 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。



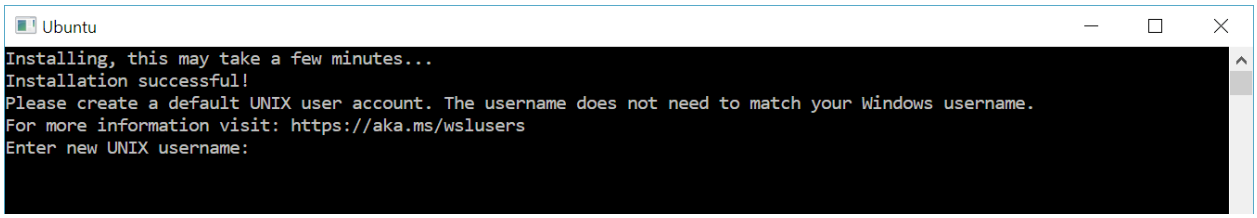
如果遇到任何问题，请查看排查安装问题一文。

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。

> **ⓘ 备注**
>
> 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？**



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`

   > 如果需要在非默认分发版中更新忘记的密码，请使用命令：`wsl -d Debian -u root`，并将 `Debian` 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。 在被告知密码已成功更新后，请使用以下命令在 PowerShell 内关闭 WSL：`exit`。

更新和升级包

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。 你可能还想要设置自定义 WSL 映像，以便在企业中分发。

设置 Windows Terminal

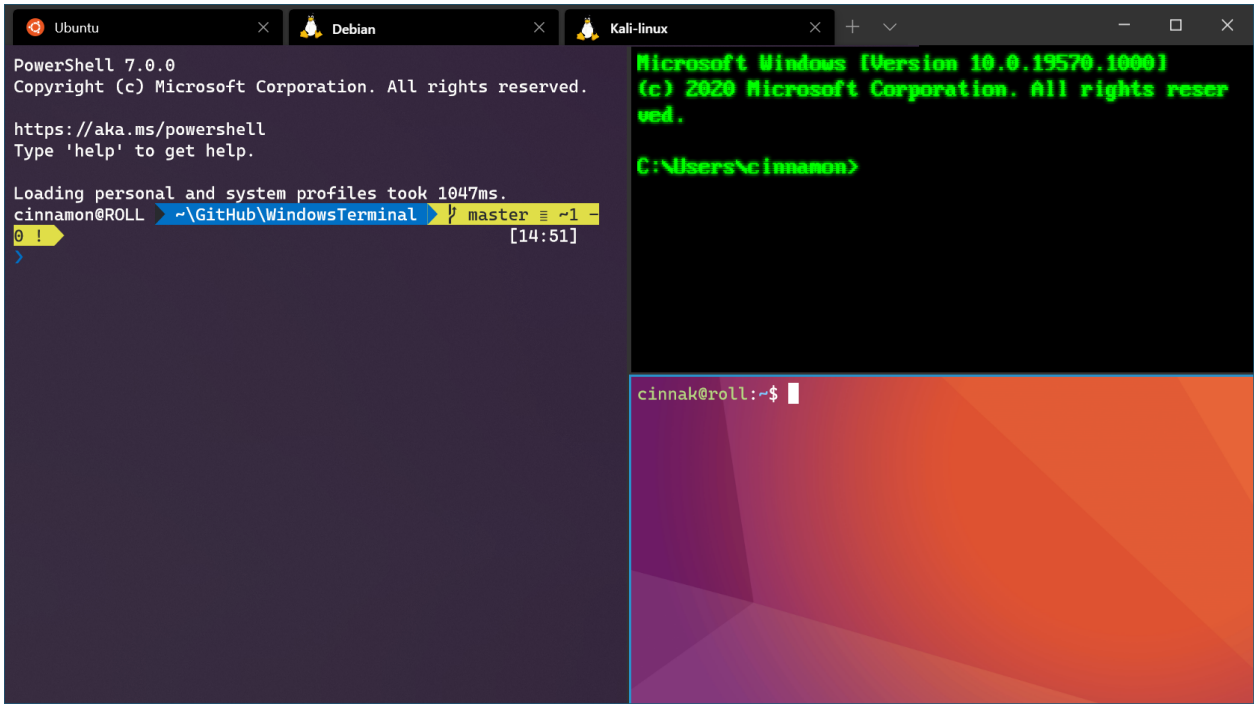
Windows Terminal 可以使用命令行接口运行任何应用程序。 它的主要功能包括多个选项卡、窗格、Unicode 和 UTF-8 字符支持、GPU 加速文本呈现引擎，你还可用它来创建你自己的主题并自定义文本、颜色、背景和快捷方式。

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。

建议将 WSL 与 Windows Terminal 配合使用，尤其是在计划使用多个命令行时。 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



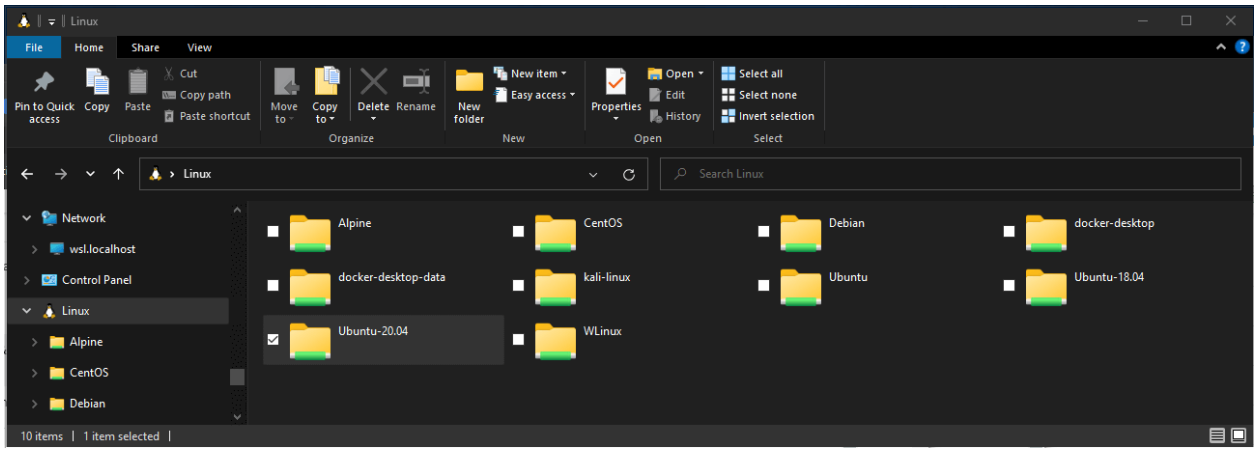
文件存储

- 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$\<DistroName>\home\<UserName>\Project`
- 而不使用 Windows 文件系统根目录：`C:\Users\<UserName>\Project` 或 `/mnt/c/Users/<UserName>/Project$`



设置你最喜欢的代码编辑器

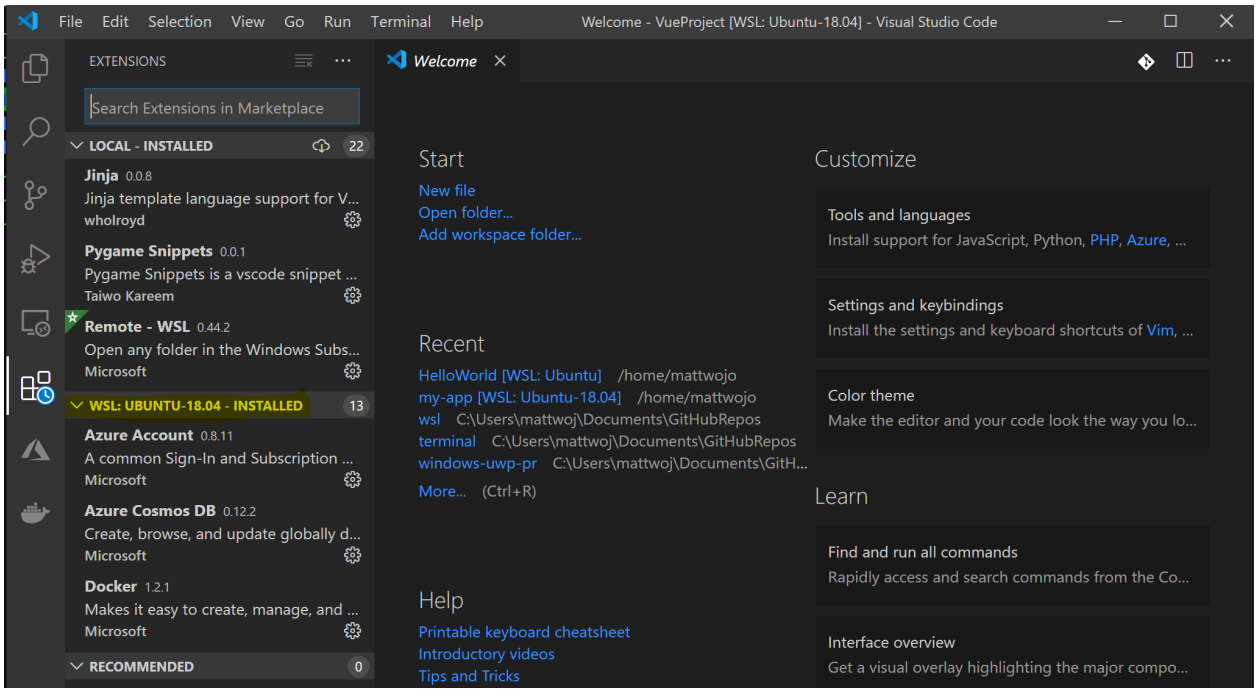
建议使用 Visual Studio Code 或 Visual Studio，因为它们直接支持使用 WSL 进行远程开发和调试。 Visual Studio Code 使你能够将 WSL 用作功能完备的开发环境。 Visual Studio 提供了对 C++ 跨平台开发的本机 WSL 支持。

使用 Visual Studio Code

按照此分步指南开始将 Visual Studio Code 与 WSL 配合使用，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。

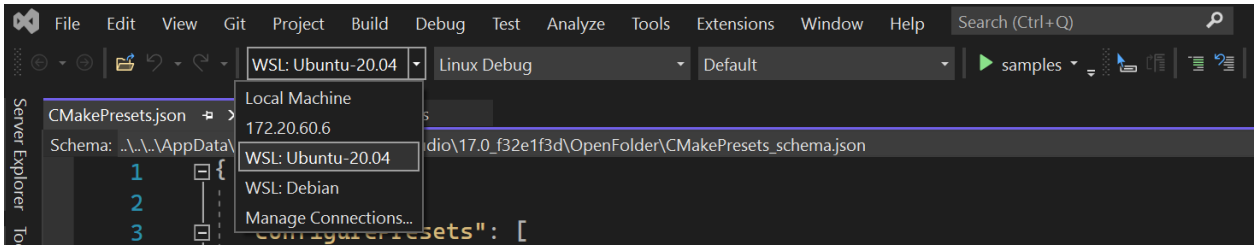
安装并设置 VS Code 后，可以通过输入以下内容使用 VS Code 远程服务器打开 WSL 项目：`code .`

请确保在命令的末尾添加句点以打开当前目录。



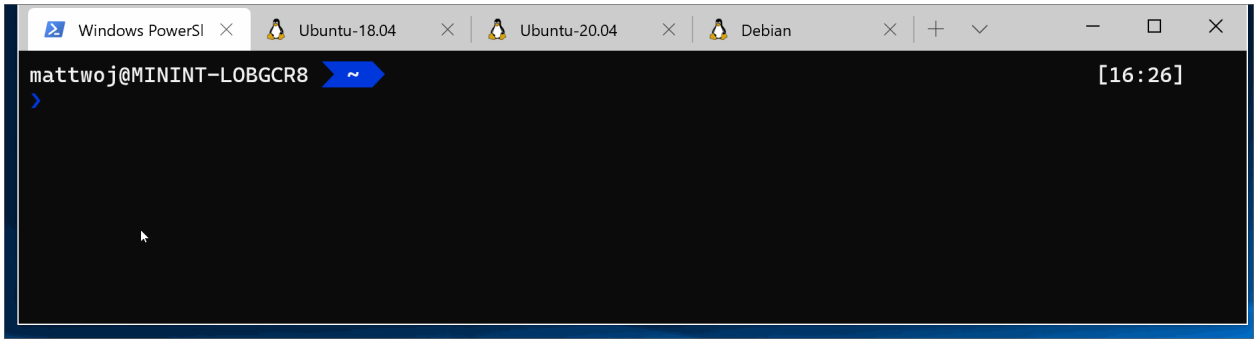
使用 Visual Studio

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。



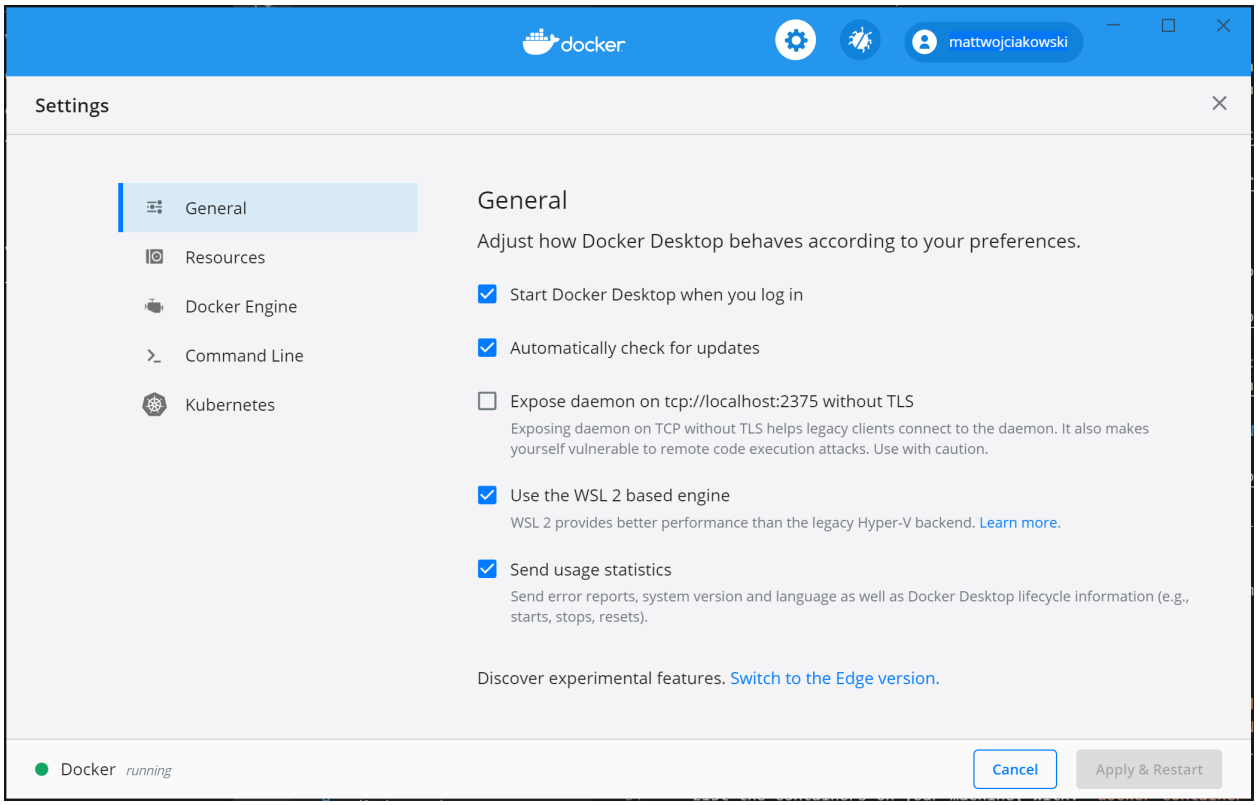
使用 Git 设置版本管理

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。



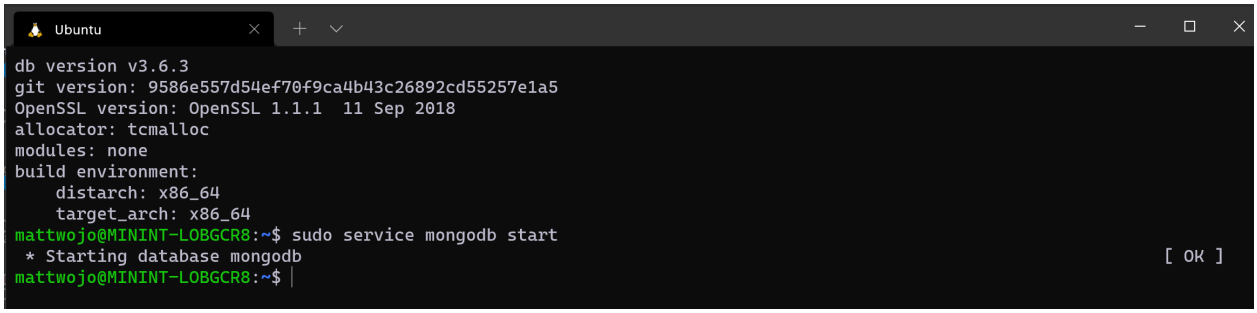
使用 Docker 设置远程开发容器

按照此分步指南开始使用 WSL 2 上的 Docker 远程容器，并使用 Docker Desktop for Windows 将项目连接到远程开发容器。



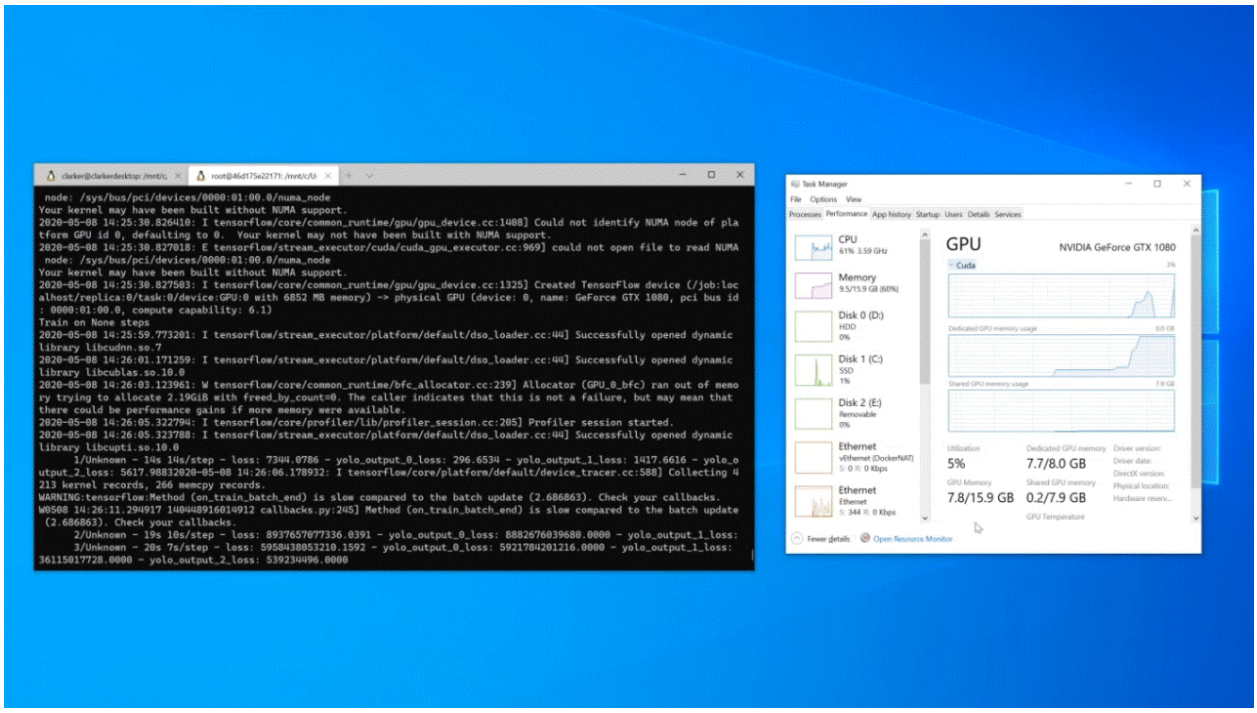
设置数据库

按照此分步指南开始使用 WSL 上的数据库，并将项目连接到 WSL 环境中的数据库。 开始使用 MySQL、PostgreSQL、MongoDB、Redis、Microsoft SQL Server 或 SQLite。



设置 GPU 加速以提高性能

按照此分步指南在 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）来加速性能繁重的工作负载。



基本 WSL 命令

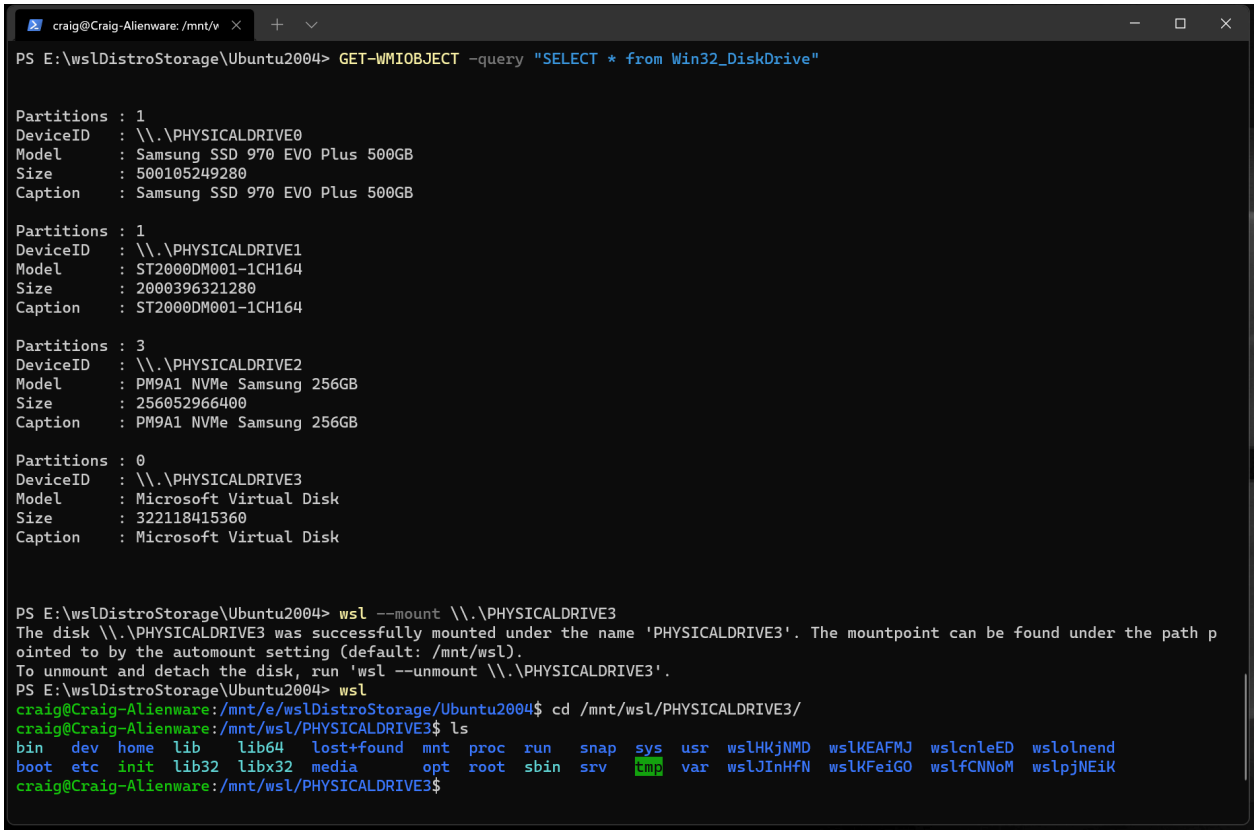
通过 WSL 安装的 Linux 发行版最好使用 PowerShell 或 Windows 命令提示符 (CMD) 进行管理。 有关使用 WSL 时需要熟悉的基本命令的列表，请参阅 WSL 命令参考指南。

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 `ls -la` 命令显示 `C:\temp>` 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

按照此分步指南开始在 WSL 2 中装载 Linux 磁盘。



运行 Linux GUI 应用

按照本教程了解如何设置和运行 WSL 上的 Linux GUI 应用。

其他资源

- 在 Windows 上设置开发环境：了解有关为首选语言或框架（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。

开始通过适用于 Linux 的 Windows 子系统使用 Visual Studio Code

项目 • 2024/02/24

Visual Studio Code 以及 WSL 扩展使你能够直接从 VS Code 使用 WSL 作为实时开发环境。 你可以：

- 在基于 Linux 的环境中进行开发
- 使用特定于 Linux 的工具链和实用程序
- 从 Windows 轻松地运行和调试基于 Linux 的应用程序，同时保持对 Outlook 和 Office 等生产力工具的访问
- 使用 VS Code 内置终端来运行选择的 Linux 发行版
- 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 或已装载的 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 难题

安装 VS Code 和 WSL 扩展

- 访问 VS Code 安装页，选择 32 位或 64 位安装程序。 在 Windows 上（不是在 WSL 文件系统中）安装 Visual Studio Code。
- 当在安装过程中系统提示“选择其他任务”时，请务必选中“添加到 PATH”选项，以便可以使用代码命令在 WSL 中轻松打开文件夹。
- 安装远程开发扩展包。 除了 Remote - SSH 和 Dev Containers 扩展之外，此扩展包还包含 WSL 扩展，使你能够打开容器中、远程计算机上或 WSL 中的任何文件夹。

> **ⓘ 重要**
>
> 若要安装 WSL 扩展，需要 VS Code 的 **1.35 5 月发行版**版本或更高版本。 建议不要在没有 WSL 扩展的情况下在 VS Code 中使用 WSL，因为会失去对自动完成、调试、linting 等的支持。有趣的事实：此 WSL 扩展安装在 $HOME/.vscode/extensions 中（在 PowerShell 中输入命令 `ls $HOME\.vscode\extensions\`）。

更新 Linux 发行版

某些 WSL Linux 发行版缺少启动 VS Code 服务器所需的库。 可以使用其他库的包管理器将其他库添加到 Linux 发行版中。

例如，要更新 Debian 或 Ubuntu，请使用：

```
Bash

sudo apt-get update
```

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

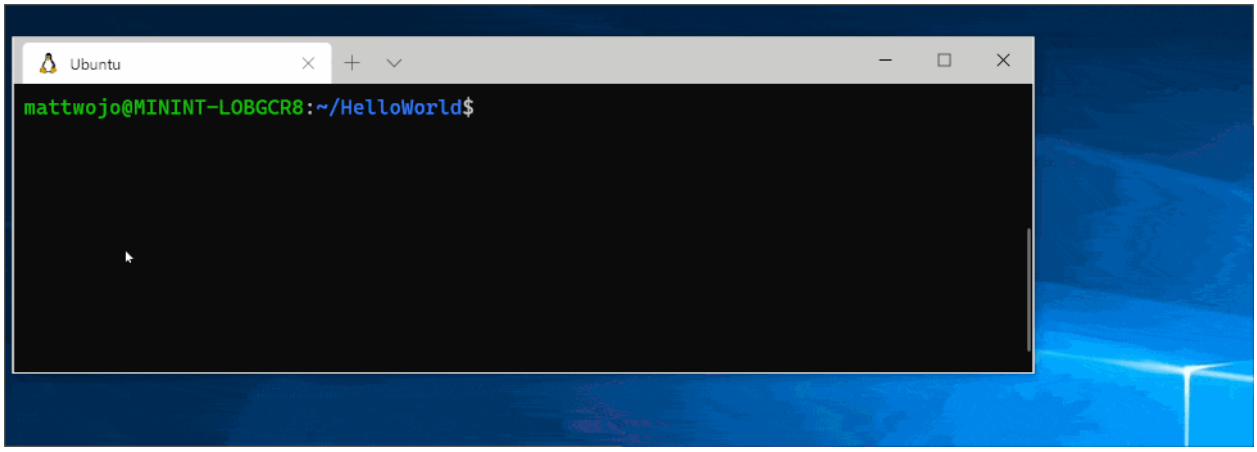
```
Bash

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

在 Visual Studio Code 中打开 WSL 项目

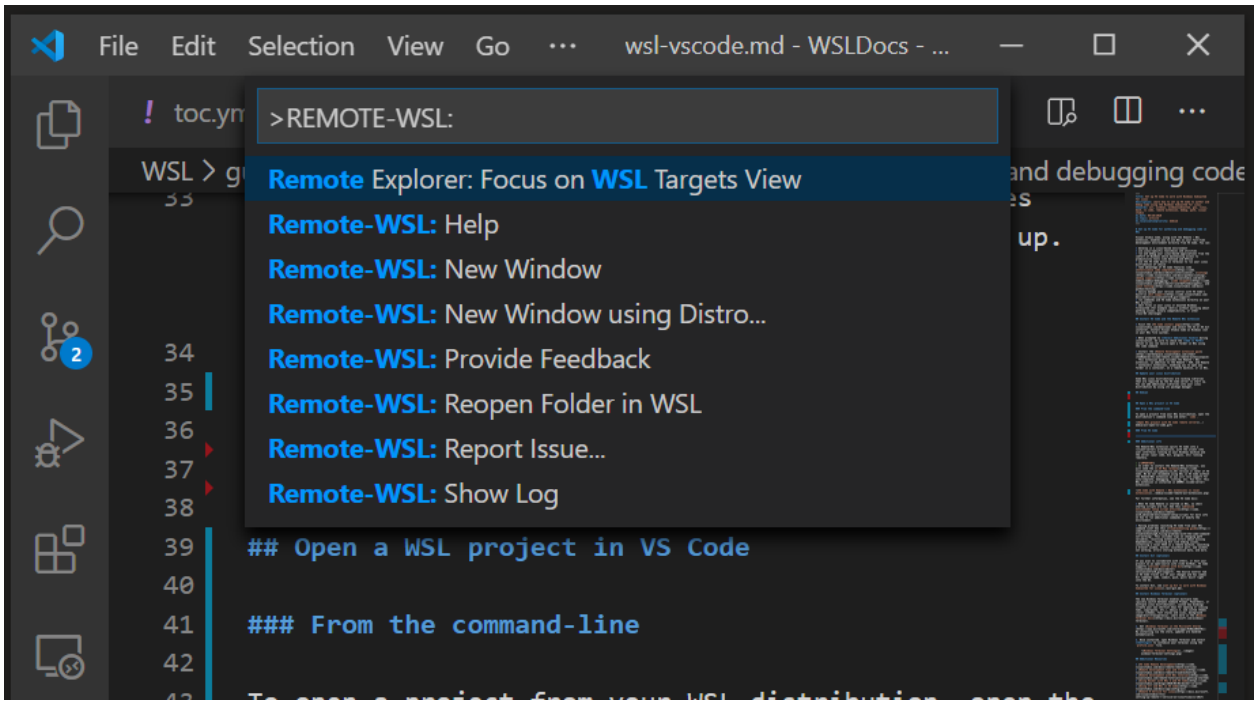
从命令行中

若要从 WSL 发行版打开项目，请打开发行版的命令行并输入：`code .`



从 VS Code 中

还可以通过使用 VS Code 中的快捷方式 `CTRL+SHIFT+P` 调出命令面板，以访问更多 VS Code WSL 选项。 如果随后键入 `WSL`，你将看到可用的选项列表，你可以在 WSL 会话中重新打开文件夹，指定要在哪个发行版中打开，等等。



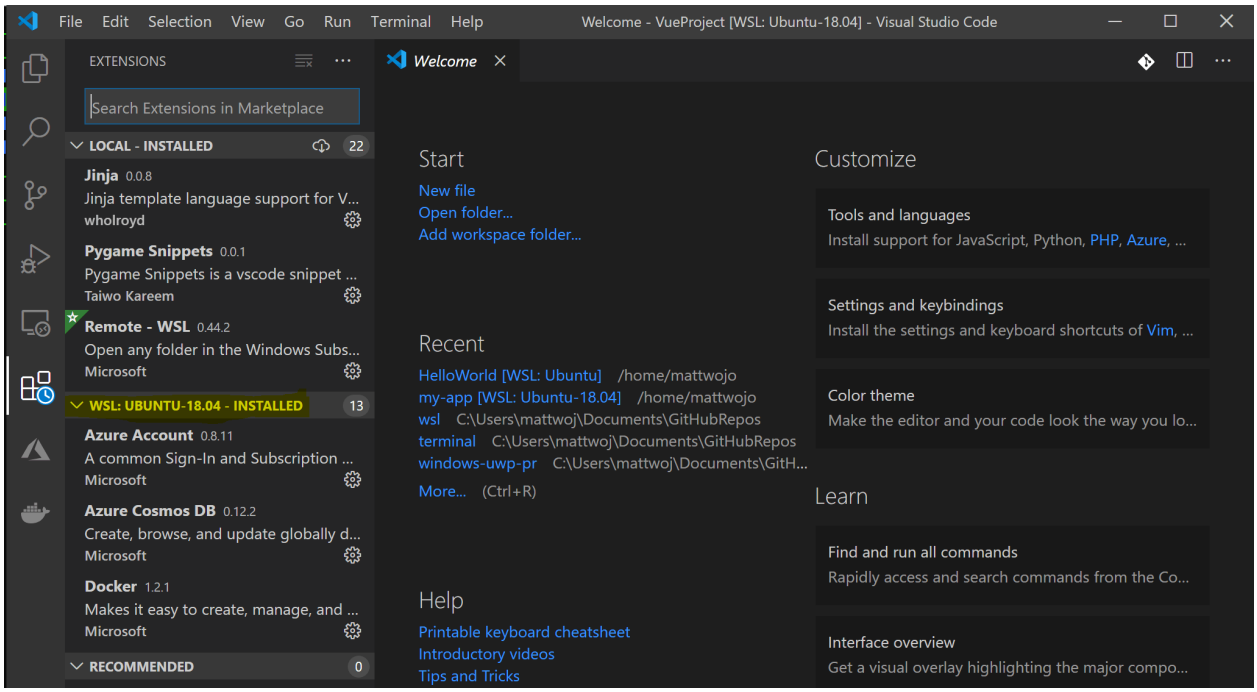
VS Code WSL 内部的扩展

WSL 扩展将 VS Code 拆分为"客户端-服务器"体系结构，使客户端（用户界面）在 Windows 计算机上运行，而使服务器（你的代码、Git、插件等）在你的 WSL 发行版中"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和 WSL 发行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（例如 Python 扩展或处理 linting 或调试等操作的任何扩展）必须单独安装在每个 WSL 发行版上。 如果在本地安装了一个未安装在 WSL 发行版上的扩展，则 VS Code 将显示一个警告图标 ⚠ 以及一个绿色的"在 WSL 中安装"按钮。



有关更多信息，请参阅以下 VS Code 文档：

- 在 WSL 中启动 VS Code 时，不会运行任何 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？ 此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 行尾问题、在远程计算机上安装本地 VSIX、启动浏览器窗口、阻止 localhost 端口、Web 套接字无法运行、存储扩展数据出错等的提示。

安装 Git（可选）

如果计划与其他人协作，或是在开放源代码站点（如 GitHub）上托管项目，则 VS Code 支持使用 Git 进行版本控制。 VS Code 中的“源代码管理”选项卡可跟踪所有更改，并直接在 UI 中内置了常见 Git 命令（add、commit、push、pull）。

若要安装 Git，请参阅设置 Git 以与适用于 Linux 的 Windows 子系统一起使用。

安装 Windows 终端（可选）

新的 Windows 终端启用多个选项卡（在命令提示符、PowerShell 或多个 Linux 发行版本之间快速切换）、自定义键绑定（创建自己的快捷键以打开或关闭选项卡、复制并粘贴等）、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 Store 中获取 Windows 终端：通过 Microsoft Store 进行安装时，将自动处理更新。

2. 安装完成后，打开 Windows 终端并选择“设置”以使用 `profile.json` 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 结合使用
- 在 VS Code 中使用 C++ 和 WSL
- 适用于 Linux 的远程 R 服务

可能需要考虑的几个附加扩展包括：

- 来自其他编辑器的键映射：如果是从另一个文本编辑器（如 Atom、Sublime、Vim、eMacs、Notepad++ 等）进行转换，则这些扩展可帮助你的环境对此进行适应。
- 设置同步：可以使用 GitHub 在不同安装之间同步 VS Code 设置。 如果在不同的计算机上工作，这有助于在它们之间保持一致的环境。

在 GitHub 上与我们协作

可以在 GitHub 上找到此内容的源，还可以在其中创建和查看问题和拉取请求。 有关详细信息，请参阅参与者指南。

Windows Subsystem for Linux 反馈

Windows Subsystem for Linux 是一个开放源代码项目。 选择一个链接以提供反馈：

提出文档问题

提供产品反馈

开始在适用于 Linux 的 Windows 子系统上使用 Git

项目 • 2023/10/07

Git 是最常用的版本控制系统。 使用 Git，可以跟踪对文件所做的更改，以便记录已完成的操作，并能够在需要时还原到文件的早期版本。 Git 还可以简化协作，使多个人员所做的更改全部合并到一个源中。

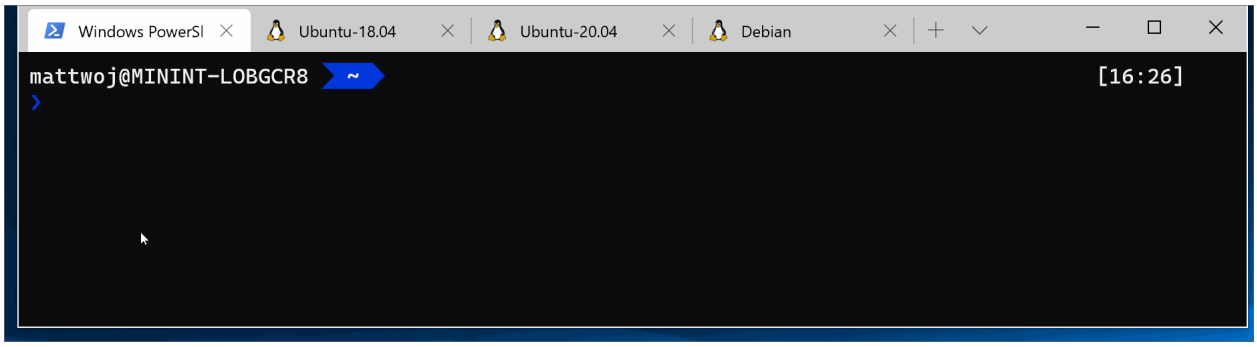
Git 可以安装在 Windows 和 WSL 上

一个重要的注意事项：启用 WSL 并安装 Linux 发行版时，将安装与计算机上的 Windows NTFS C:\ 驱动器分离的新文件系统。 在 Linux 中，驱动器没有字母。 将为它们提供装入点。 在 WSL 的情况下，文件系统 `/` 的根是根分区或文件夹的装入点。 并非 `/` 下的所有内容都是相同的驱动器。 例如，在我的笔记本电脑上，我安装了两个版本的 Ubuntu（20.04 和 18.04）以及 Debian。 如果我打开这些发行版，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发行版的目录路径。

Linux 发行版	Windows 访问主文件夹的路径
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**
>
> 如果想从 WSL 发行版命令行访问 Windows 文件目录，而不是使用 `C:\Users\username`，则需使用 `/mnt/c/Users/username` 访问该目录，因为 Linux 发行版将 Windows 文件系统视为已装载的驱动器。

需要在要使用它的每个文件系统上安装 Git。



安装 Git

大多数适用于 Linux 的 Windows 子系统发行版已安装了 Git，但是，可能需要将其更新到最新版本。 还需要设置 git 配置文件。

若要安装 Git，请参阅适用于 Linux 的 Git 下载站点。 每个 Linux 发行版都有自己的包管理器和安装命令。

对于 Ubuntu/Debian 中最新的稳定 Git 版本，请输入命令：

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

> **备注**
>
> 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，请打开正在使用的发行版的命令行，然后使用以下命令设置名称（将“Your Name”替换为你的首选用户名）：

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

使用以下命令设置电子邮件（用你喜欢的电子邮件替换“youremail@domain.com”）：

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

> **提示**
>
> 如果你还没有 GitHub 帐户，则可以**在 GitHub 上注册一个帐户**。 如果以前从未使用过 Git，则 **GitHub 指南**可以帮助入门。 如果需要编辑 Git 配置，则可以使用内置文本编辑器（如 nano：`nano ~/.gitconfig`）来执行此操作。

建议使用双因素身份验证 (2FA) 保护你的帐户。

Git 凭据管理器设置

Git 凭据管理器 (GCM) 是在 .NET 上构建的安全的 Git 凭据帮助程序，可用于 WSL1 和 WSL2。 它支持对 GitHub 存储库、Azure DevOps、Azure DevOps Server 和 Bitbucket 进行多重身份验证。

GCM 集成在 GitHub 等服务的身份验证流程中，在你通过托管提供程序身份验证后，它就会请求新的身份验证令牌。 然后，它会将令牌安全地存储在 Windows 凭据管理器中。 首次之后，可以使用 Git 与托管提供程序通信，而无需重新进行身份验证。 它将只需访问 Windows 凭据管理器中的令牌。

若要将 GCM 与 WSL 配合使用，必须使用 Windows 10 版本 1903 或更高版本。 这是 Windows 的第一个版本，其中包含 GCM 用于与 WSL 发行版中的 Git 互操作所需的 `wsl.exe` 工具。

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，系统将要求你选择一个凭据帮助程序，GCM 是默认选择。

如果你有理由不安装 Git for Windows，可以直接在 WSL 发行版中将 GCM 作为 Linux 应用程序安装，但请注意，这样做意味着 GCM 作为 Linux 应用程序运行，而不能利用主机 Windows 操作系统的身份验证或凭据存储功能。 有关如何在没有 Git for Windows 的情况下配置 WSL 的说明，请参阅 GCM 存储库。

若要设置 GCM 以与 WSL 发行版配合使用，请打开发行版，然后输入以下命令：

如果安装的 GIT 为 >= v2.39.0

Bash

```
git config --global credential.helper "/mnt/c/Program\
Files/Git/mingw64/bin/git-credential-manager.exe"
```

否则，如果安装的 GIT 为 >= v2.36.1

```
Bash

git config --global credential.helper "/mnt/c/Program\
Files/Git/mingw64/libexec/git-core/git-credential-manager.exe"
```

否则，如果版本为 < v2.36.1，请输入此命令：

```
Bash

git config --global credential.helper "/mnt/c/Program\
Files/Git/mingw64/bin/git-credential-manager-core.exe"
```

> **备注**
>
> 使用 GCM 作为 WSL Git 安装的凭据帮助程序意味着，默认情况下，GCM 不遵循 WSL Git 中的任何配置集。 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 GCM 的代理设置等内容需要在 Git for Windows 和 WSL Git 中设置，因为它们存储在不同文件中（`%USERPROFILE%\.gitconfig` 与 `\\wsl$\distro\home\$USER\.gitconfig`）。 你可以配置 WSL，使得 GCM 将使用 WSL Git 配置，但这意味着代理设置对于特定的 WSL 安装将是唯一的，并且不会与其他安装或 Windows 主机共享。

Git 与 SSH

Git 凭据管理器仅适用于 HTTP(S) 远程。 你仍然可以将 Git 与 SSH 配合使用：

- Azure DevOps SSH
- GitHub SSH
- Bitbucket SSH

Azure 的其他配置

若要使用 Azure Repos 或 Azure DevOps，需要一些额外配置：

```
Bash

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

现在，你在 WSL 发行版中执行的任何 git 操作都将使用 GCM。 如果已为主机缓存凭据，那么它会从凭据管理器访问这些凭据。 如果尚未缓存凭据，你将收到一个请求凭据的对话响应，即使你处于 Linux 控制台中也是如此。

> **提示**
>
> 如果使用 GPG 密钥来确保代码签名安全，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的推荐 .gitignore 文件设置。 例如，此处是 GitHub 用于 Node.js 项目的默认 gitignore 模板。

如果你选择使用 GitHub 网站创建新的存储库，则会出现可用于使用自述文件初始化存储库的复选框，设置用于特定项目类型的 .gitignore 文件，以及用于添加许可证（如果需要）的选项。

Git 和 VS Code

Visual Studio Code 内置了对 Git 的支持，包括一个源代码控制选项卡，用于显示更改和处理各种 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，请确保设置一致的行尾。

由于 Windows 和 Linux 使用不同的默认行尾，因此 Git 可能会报告大量修改后的文件，这些文件除了行尾之外没有任何区别。 为防止发生这种情况，可以使用 `.gitattributes` 文件或在 Windows 端全局禁用行尾转换。 请参阅此 VS Code 文档，了解如何解决 Git 行尾问题。

其他资源

- WSL & VS Code
- GitHub 学习实验室：在线课程
- Git 可视化工具
- Git 工具 - 凭据存储

开始使用适用于 Linux 的 Windows 子系统上的数据库

项目 • 2023/09/30

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，更新到版本 2004，**内部版本 19041** 或更高版本。
- 使用 WSL 安装 Linux 发行版，并创建 Linux 用户名和密码。

数据库系统之间的差异

数据库系统的一些最常用的选项 包括：

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

MySQL 是一种开源 SQL 关系数据库，可将数据组织到一个或多个表中，其中的数据类型可能相互关联。 它可垂直缩放，这意味着一台终极计算机将为你执行工作。 它是目前四种数据库系统中应用最广泛的。

PostgreSQL（有时称为 Postgres）也是一种开源 SQL 关系数据库，着重于扩展性和标准合规性。 现在，它也可以处理 JSON，但通常更适合处理结构化数据、进行垂直缩放以及符合 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。 这些也是建立在服务器上的关系数据库管理系统，其主要功能是根据软件应用程序的要求存储和检索数据。

SQLite 是一种开源自包含、基于文件的"无服务器"数据库，以其可移植性、可靠性和即使在低内存环境中也能有良好性能而著称。

MongoDB 是一种开源 NoSQL 文档数据库，旨在处理 JSON 和存储无架构数据。 它可水平缩放的，这意味着多台较小的计算机将为你执行工作。 它适用于实现灵活性和处理非结构化数据，以及缓存实时分析。

Redis 是一种开源 NoSQL 内存数据结构存储。 它使用键值对来存储而不是文档。

安装 MySQL

要在 WSL 上运行的 Linux 发行版上安装 MySQL，只需按照 MySQL 文档中的在 Linux 上安装 MySQL 说明进行操作。可能需要首先在 `wsl.conf` 配置文件中启用 systemd 支持。

使用 Ubuntu 发行版的示例：

1. 打开 Ubuntu 命令行并更新可用的包：`sudo apt update`
2. 更新该包后，使用以下命令安装 MySQL：`sudo apt install mysql-server`
3. 确认安装并获取版本号：`mysql --version`
4. 启动 MySQL Server / 检查状态：`systemctl status mysql`
5. 若要打开 MySQL 提示符，请输入：`sudo mysql`
6. 若要查看可用的数据库，请在 MySQL 提示符中输入：`SHOW DATABASES;`
7. 若要创建新数据库，请输入：`CREATE DATABASE database_name;`
8. 若要删除数据库，请输入：`DROP DATABASE database_name;`

有关使用 MySQL 数据库的更多信息，请参阅 MySQL 文档。

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

可能还想运行包含的安全脚本。 这会更改一些不太安全的默认选项，例如远程根登录名和示例用户。 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：`sudo service mysql start`
2. 启动安全脚本提示符：`sudo mysql_secure_installation`
3. 第一个提示符会询问是否要设置 VALIDATE PASSWORD COMPONENT，它可用于测试 MySQL 密码的强度。 如果要设置一些简单的密码，则不应设置此组件。
4. 然后你将为 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

要在 WSL（即 Ubuntu）上安装 PostgreSQL：

1. 打开 WSL 终端（即 Ubuntu）。
2. 更新 Ubuntu 包：`sudo apt update`
3. 更新该包后，使用以下命令安装 PostgreSQL（和 -contrib 包，其中包含一些有用的实用程序）：`sudo apt install postgresql postgresql-contrib`
4. 确认安装并获取版本号：`psql --version`

安装 PostgreSQL 后，需要知道以下 3 个命令：

- `sudo service postgresql status` 用于检查数据库的状态。
- `sudo service postgresql start` 用于开始运行数据库。
- `sudo service postgresql stop` 用于停止运行数据库。

默认管理员用户 `postgres` 需要分配的密码才能连接到数据库。 要设置密码，请执行以下操作：

1. 输入命令：`sudo passwd postgres`
2. 系统将提示你输入新密码。
3. 关闭并重新打开终端。

通过 psql shell 运行 PostgreSQL：

1. 启动 postgres 服务：`sudo service postgresql start`
2. 连接到 postgres 服务，并打开 psql shell：`sudo -u postgres psql`

成功输入 psql shell 后，将显示更改为如下所示的命令行：`postgres=#`

> **备注**
>
> 或者，也可以通过使用 `su - postgres` 切换为 postgres 用户，然后输入命令 `psql` 来打开 psql shell。

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

要查看在 PostgreSQL 安装上创建的用户帐户，请在 WSL 终端上使用 `psql -c "\du"`；如果已打开 psql shell，则仅使用 `\du`。 此命令将显示列：帐户用户名、角色属性列表和角色组成员。 要返回命令行，请输入：`q`。

有关使用 PostgreSQL 数据库的更多信息，请参阅 PostgreSQL 文档。

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

安装 MongoDB

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

根据用于安装的 Linux 发行版，安装 MongoDB 需要的步骤可能略有不同。 另请注意，MongoDB 安装可能因你要安装的版本而异。 使用 MongoDB 文档左上角的版本下拉列表选择符合目标的版本。 最后，可能需要在与 WSL 一起使用的 Linux 发行版的 `wsl.conf` 配置文件中启用 systemd 支持。 `systemctl` 命令是 systemd init 系统的一部分，如果发行版使用的是 systemv，则它可能无法正常工作。

VS Code 支持通过 Azure CosmosDB 扩展来处理 MongoDB 数据库，你可以在 VS Code 中创建、管理和查询 MongoDB 数据库。 有关详细信息，请访问 VS Code 文档：使用 MongoDB 。

有关详细信息，请参阅 MongoDB 文档：

- 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

要在 WSL 运行的 Linux 发行版上安装 SQL Server：Linux 上的 SQL Server。 若要在 VS Code 中使用 Microsoft SQL Server 数据库，请尝试使用 MSSQL 扩展 。

安装 SQLite

要在 WSL（即 Ubuntu）上安装 SQLite：

1. 打开 WSL 终端（即 Ubuntu）。
2. 更新 Ubuntu 包：`sudo apt update`
3. 更新该包后，使用以下命令安装 SQLite3：`sudo apt install sqlite3`
4. 确认安装并获取版本号：`sqlite3 --version`

若要创建名为“example.db”的测试数据库，请输入：`sqlite3 example.db`

若要查看 SQLite 数据库列表，请输入：`.databases`

若要查看数据库的状态，请输入：`.dbinfo ?DB?`

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

现在，输入 `.dbinfo ?DB?` 后将显示你已创建的数据库。

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，请尝试使用 SQLite 扩展。

安装 Redis

要在 WSL（即 Ubuntu）上安装 Redis：

1. 打开 WSL 终端（即 Ubuntu）。
2. 更新 Ubuntu 包：`sudo apt update`
3. 更新该包后，使用以下命令安装 Redis：`sudo apt install redis-server`
4. 确认安装并获取版本号：`redis-server --version`

开始运行 Redis 服务器：`sudo service redis-server start`

检查 redis 是否正常工作（redis-cli 是与 Redis 对话的命令行接口实用程序）：`redis-cli ping`。这应返回“PONG”的回复。

停止运行 Redis 服务器：`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，请尝试使用 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 发行版上运行的服务，请输入：`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，你可以考虑在 WSL 上的 `.profile` 文件中设置别名，使这些命令更便于使用、易于记忆。

要设置自己的自定义别名或快捷方式来执行这些命令，请执行以下操作：

1. 打开 WSL 终端并输入 `cd ~`以确保位于根目录中。
2. 使用终端文本编辑器 Nano 打开 `.profile` 文件，该文件可控制终端的设置：`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'
```

这样你就可以输入 `start-pg` 开始运行 postgresql 服务，并输入 `run-pg` 来打开 psql shell。 `start-pg` 和 `run-pg` 可更改为所需的任何名称，但是请注意不要覆盖 postgres 已经使用的命令！

4. 添加新别名后，请使用 Ctrl + X 退出 Nano 文本编辑器 - 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，然后尝试使用新的别名命令。

疑难解答

错误：目录同步 fdatasync 参数无效

确保在 WSL 2 模式下运行 Linux 发行版。 如需从 WSL 1 切换到 WSL 2 的帮助，请参阅将发行版本设置为 WSL 1 或 WSL 2。

其他资源

- 在 Windows 上设置开发环境

WSL 2 上的 Docker 远程容器入门

项目 • 2024/01/04

本分步指南将通过使用 WSL 2（适用于 Linux 的 Windows 子系统，版本 2）设置 Docker Desktop for Windows，帮助开始使用远程容器进行开发。

Docker Desktop for Windows 为生成、交付和运行 Docker 化的应用提供了一个开发环境。 通过启用基于 WSL 2 的引擎，可以在同一计算机上的 Docker Desktop 中运行 Linux 和 Windows 容器。 （Docker Desktop 免费供个人和小型企业使用，有关专业、团队或企业定价的信息，请参阅 Docker 站点常见问题解答 ）。

> **备注**
>
> 建议使用 Docker Desktop，因为它**与 Windows 和适用于 Linux 的 Windows 子系统集成** 。 但是，虽然 Docker Desktop 支持运行 Linux 和 Windows 容器，但**不**能同时运行它们两个。 若要同时运行 Linux 和 Windows 容器，需要在 WSL 中安装并运行单独的 Docker 实例。 如果需要同时运行容器，或者只是希望直接在 Linux 发行版中安装容器引擎，请按照该容器服务的 Linux 安装说明进行操作，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，用于创建、部署和运行应用程序（通过使用容器）。 容器使开发人员可以将应用与需要的所有部件（库、框架、依赖项等）打包为一个包一起交付。 使用容器可确保此应用的运行与之前相同，而不受任何自定义设置或运行该应用的计算机上先前安装的库的影响（运行应用的计算机可能与用于编写和测试应用代码的计算机不同）。 这使开发人员可以专注于编写代码，而无需操心将运行代码的系统。

Docker 容器与虚拟机类似，但不会创建整个虚拟操作系统。 相反，Docker 允许应用使用与运行它的系统相同的 Linux 内核。 这使得应用包能够仅要求主计算机上尚未安装的部件，从而降低包大小以及提高性能。

将 Docker 容器与 Kubernetes 等工具结合使用以实现持续可用性是容器普及的另一个原因。 这样就可以在不同的时间创建应用容器的多个版本。 每个容器（及其特定的微服务）均可以动态更换，而无需停止整个系统来进行更新或维护。 你可以准备一个包含所有更新的新容器，将该容器设置用于生产，并在新容器准备就绪后直接指向该容器。 你还可以使用容器对不同版本的应用进行存档，如有需要，还可将其作为安全回退保持运行。

若要了解详细信息，请查看 Docker 容器简介。

先决条件

- WSL 1.1.3.0 或更高版本。
- Windows 11 64 位：家庭版或专业版 21H2 或更高版本，或者企业版或教育版 21H2 或更高版本。
- Windows 10 64 位（推荐）：家庭版或专业版 22H2（内部版本 19045）或更高版本，或者企业版或教育版 22H2（内部版本 19045）或更高版本。 （最低要求）：家庭版或专业版 21H2（内部版本 19044）或更高版本，或者企业版或教育版 21H2（内部版本 19044）或更高版本。 更新 Windows
- 具有二级地址转换 (SLAT) 的 64 位处理器。
- 4GB 系统 RAM。
- 在 BIOS 中启用硬件虚拟化。
- 安装 WSL，并为在 WSL 2 中运行的 Linux 发行版设置用户名和密码。
- 安装 Visual Studio Code （可选）。 这将提供最佳体验，包括能够在远程 Docker 容器中进行编码和调试并连接到 Linux 发行版。
- 安装 Windows 终端（可选）。 这将提供最佳体验，包括能够在同一界面中自定义和打开多个终端（包括 Ubuntu、Debian、PowerShell、Azure CLI 或你喜欢使用的任何内容）。
- 在 Docker Hub 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。

> **备注**
>
> WSL 可以在 WSL 版本 1 或 WSL 2 模式下运行发行版。 可通过打开 PowerShell 并输入以下内容进行检查：`wsl -l -v`。 通过输入 `wsl --set-version <distro> 2`，确保发行版设置为使用 WSL 2。 将 `<distro>` 替换为发行版名称（例如 Ubuntu 18.04）。
>
> 在 WSL 版本 1 中，由于 Windows 和 Linux 之间的根本差异，Docker 引擎无法直接在 WSL 内运行，因此 Docker 团队使用 Hyper-V VM 和 LinuxKit 开发了一个替代解决方案。 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 容器可以在没有模拟的情况下以本机方式运行，从而在 Windows 和 Linux 工具之间实现更好的性能和互操作性。

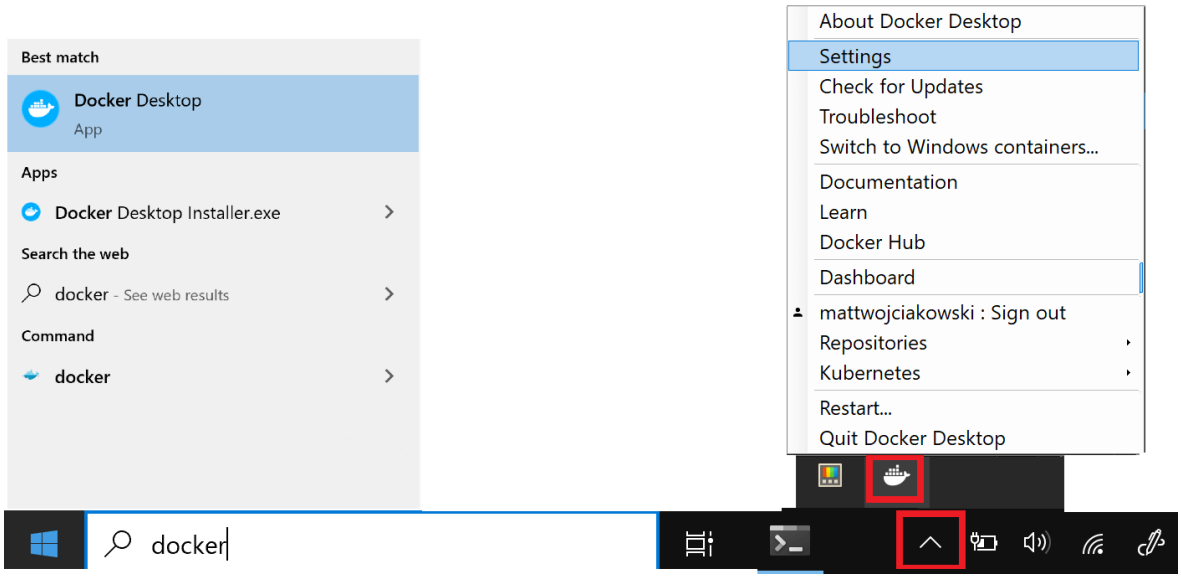
安装 Docker Desktop

借助 Docker Desktop for Windows 中支持的 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。

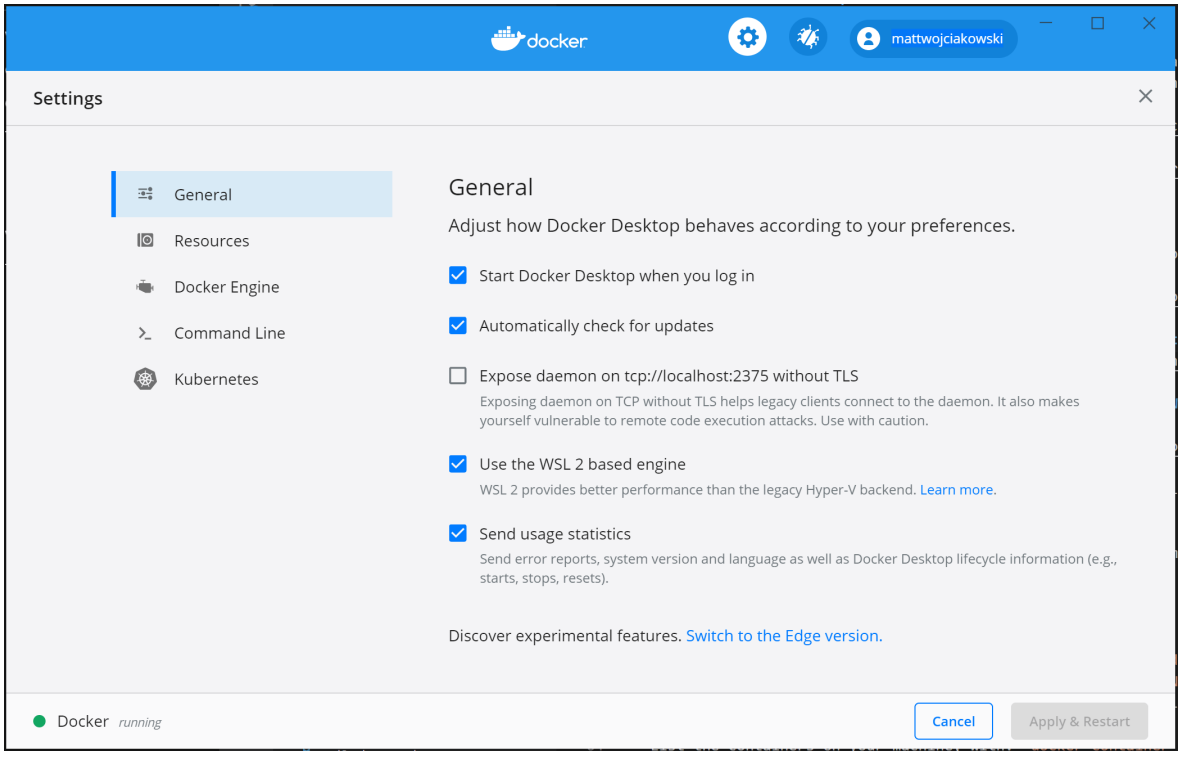
若要安装 Docker（在已安装 WSL 之后）：

1. 下载 Docker Desktop 并按照安装说明进行操作。

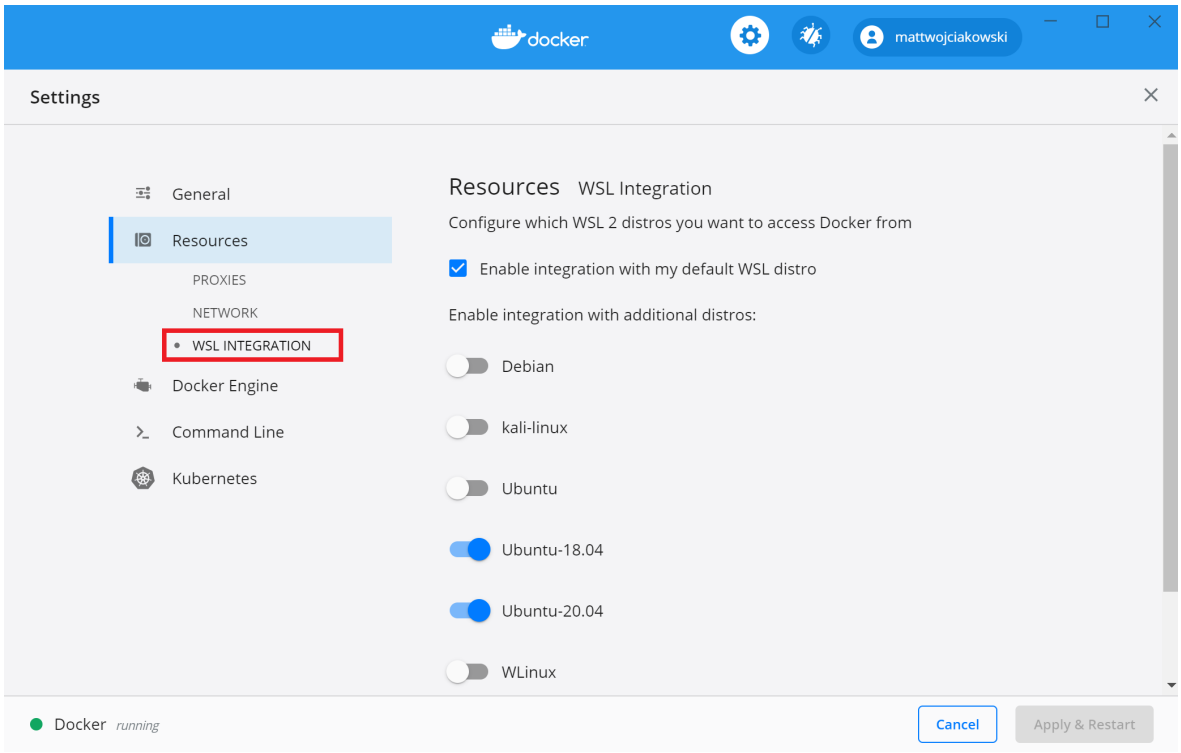
2. 安装后，从 Windows 开始菜单启动 Docker Desktop，然后从任务栏的隐藏图标菜单中选择 Docker 图标。 右键单击该图标以显示 Docker 命令菜单，然后选择“设置”。



3. 确保在“设置”>“常规”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 发行版中进行选择。



5. 若要确认已安装 Docker，请打开 WSL 发行版（例如 Ubuntu），并通过输入 `docker --version` 来显示版本和内部版本号

6. 通过使用 `docker run hello-world` 运行简单的内置 Docker 映像，测试安装是否正常工作

> **提示**
>
> 下面是一些需要了解的有用 Docker 命令：
>
> - 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`
> - 使用以下命令列出计算机上的 docker 映像（此时仅为 hello-world 映像）：`docker image ls --all`
> - 使用以下命令列出计算机上的容器：`docker container ls --all` 或 `docker ps -a`（如果没有 -a 显示全部标志，则仅显示正在运行的容器）
> - 使用以下命令列出有关 Docker 安装的系统范围的信息，包括 WSL 2 上下文中你可使用的统计信息和资源（CPU 和内存）：`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，建议使用 VS Code 以及 WSL、Dev Containers 和 Docker 扩展。

- 安装 VS Code WSL 扩展。 此扩展使你能够在 VS Code 中打开在 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 的难题）。
- 安装 VS Code Dev Containers 扩展。 此扩展使你能够打开容器内的项目文件夹或存储库，并利用 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 内生成、管理和部署容器化应用程序的功能。 （需要 Dev Containers 扩展才能使用容器作为你的开发环境。）

让我们使用 Docker 为现有应用项目创建开发容器。

1. 对于此示例，我将在 Python 开发环境设置文档中使用适用于 Django 的 Hello World 教程中的源代码。如果想要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）并输入：`git clone https://github.com/mattwojo/helloworld-django.git`

   > **备注**
   >
   > 始终将代码存储在使用工具的相同文件系统中。 这将提高文件访问性能。 在本例中，我们使用的是 Linux 发行版 (Ubuntu)，并且想要将项目文件存储在 WSL 文件系统 `\\wsl\` 上。 在 WSL 中使用 Linux 工具访问项目文件时，将项目文件存储在 Windows 文件系统上会明显降低速度。

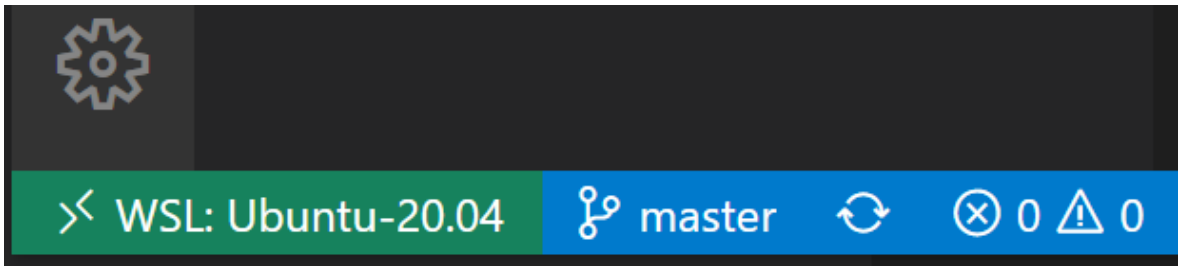
2. 在 WSL 终端中，将目录更改为此项目的源代码文件夹：

   ```Bash
   cd helloworld-django
   ```

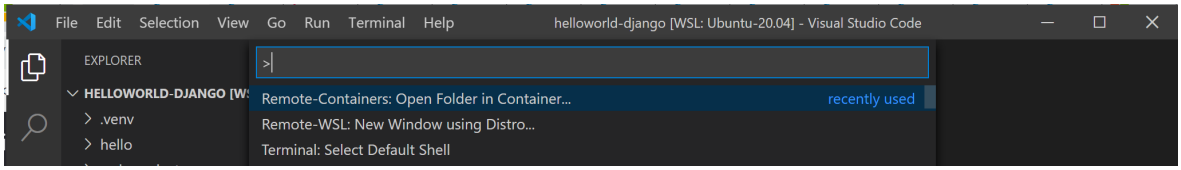
3. 通过输入以下内容，在本地 WSL 扩展服务器上运行的 VS Code 中打开项目：

   ```Bash
   code .
   ```

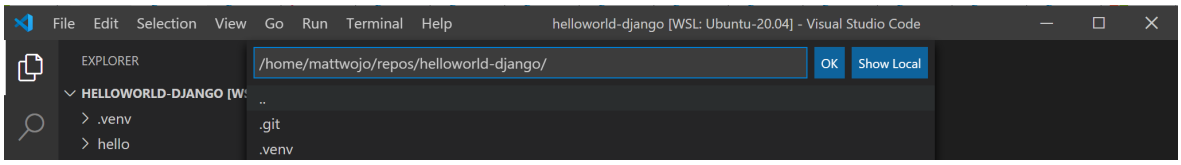
   通过检查 VS Code 实例左下角的绿色远程指示器，确认已连接到 WSL Linux 发行版。



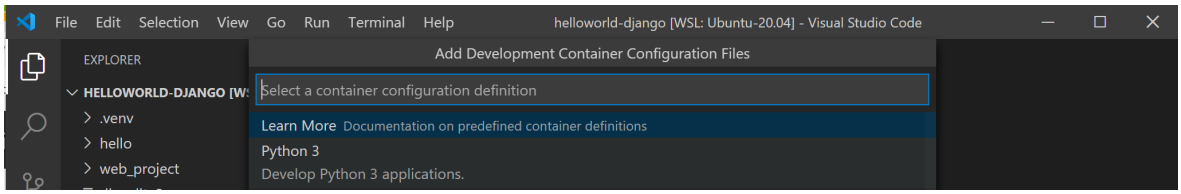
4. 从 VS Code 命令托盘 (Ctrl + Shift + P)，输入：**开发容器：在容器中重新打开**，因为我们使用的是已使用 WSL 扩展打开的文件夹。 也可以使用 **开发容器：在容器中打开文件夹**... 使用本地 `\\wsl$` 共享（从 Windows 端）选择 WSL 文件夹。 有关更多详细信息，请参阅 Visual Studio Code 快速入门：在容器中打开现有文件夹。 如果这些命令在你开始键入时未显示，请检查并确保你已安装上面链接的 Dev Containers 扩展。



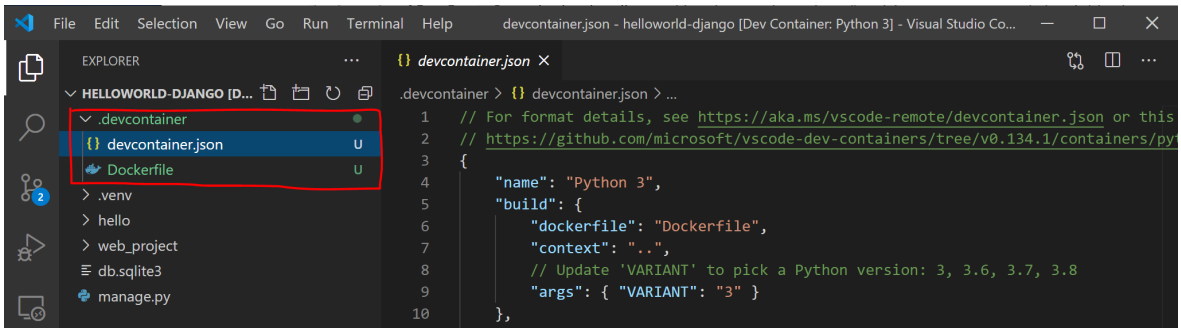
5. 选择要容器化的项目文件夹。 在我的示例中，它是 `\\wsl\Ubuntu-20.04\home\mattwojo\repos\helloworld-django\`



6. 将显示容器定义列表，因为项目文件夹（存储库）中还没有 dev container 配置。 显示的容器配置定义列表将根据项目类型进行筛选。 对于 Django 项目，我将选择 Python 3。

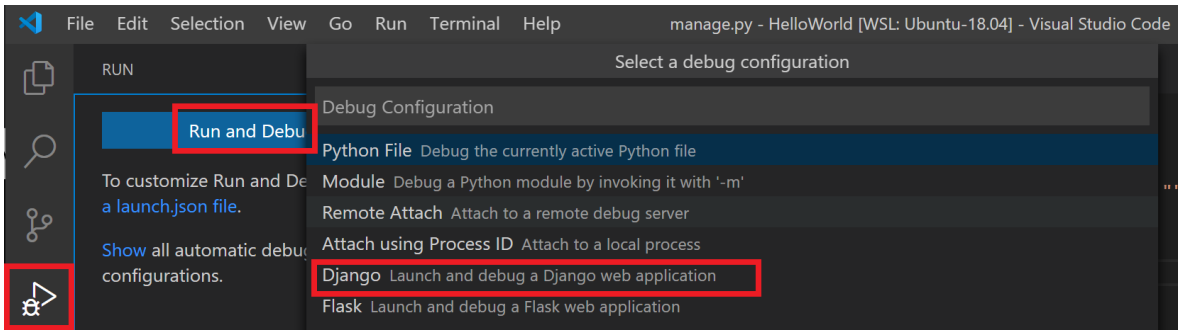


7. 系统将打开新的 VS Code 实例，开始生成新映像，生成完成后，将启动容器。 将看到出现新的 `.devcontainer` 文件夹，其中 `Dockerfile` 和 `devcontainer.json` 文件中包含容器配置信息。

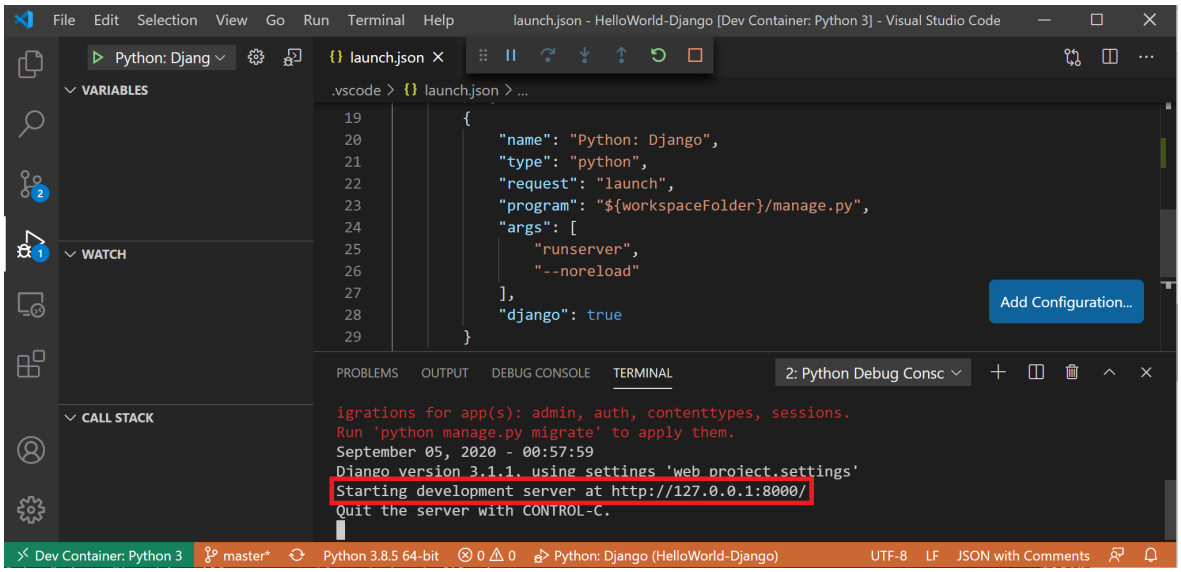


8. 若要确认项目仍然连接到 WSL 和容器中，请打开 VS Code 集成终端 (Ctrl + Shift + ~)。 通过输入 `uname` 检查操作系统，并通过 `python3 --version` 检查 Python 版本。 可以看到，uname 返回为“Linux”，因此你仍然连接到 WSL 2 引擎，Python 版本号将基于容器配置，该配置可能不同于 WSL 发行版上安装的 Python 版本。

9. 若要使用 Visual Studio Code 在容器内运行和调试应用，请首先打开“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”以选择调试配置，并选择最适合项目的配置（在我的示例中，这将是“Django”）。 这会在项目的 `launch.json` 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 中，选择“运行”>“开始调试”（或只按 F5 键）。 这会在 VS Code 中打开终端，并且你应会看到如下所示的结果：“正在 http://127.0.0.1:8000/ 启动开发服务器。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，并查看在其容器中运行的项目。



现在，你已使用 Docker Desktop 成功配置了远程开发容器，该容器由 WSL 2 后端提供支持，你可以使用 VS Code 对该容器进行编码、生成、运行、部署或调试！

疑难解答

已弃用的 WSL docker 上下文

如果你使用的是 Docker for WSL 的早期技术预览版，则可能有一个名为“wsl”的 Docker 上下文，该上下文现已弃用。 可以使用以下命令进行检查：`docker context ls`。 如果想要同时对 Windows 和 WSL2 使用默认上下文，可以使用 `docker context rm wsl` 命令删除此“wsl”上下文，以避免出现错误。

使用此已弃用的 wsl 上下文可能遇到的错误包括：`docker wsl open //./pipe/docker_wsl: The system cannot find the file specified.` 或 `error during connect: Get http://%2F%2F.%2Fpipe%2Fdocker_wsl/v1.40/images/json?all=1: open //./pipe/docker_wsl: The system cannot find the file specified.`

有关此问题的详细信息，请参阅如何在 Windows 10 上的适用于 Linux 的 Windows 系统 (WSL2) 中设置 Docker。

查找 docker 映像存储文件夹时遇到问题

Docker 创建了两个用于存储数据的发行版文件夹：

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

可以通过打开 WSL Linux 发行版并输入 `explorer.exe .` 以在 Windows 文件资源管理器中查看文件夹来查找这些文件夹。 输入：`\\wsl\<distro name>\mnt\wsl`，将 `<distro name>` 替换为你的发行版的名称（即 Ubuntu-20.04）以查看这些文件夹。

如需查找有关在 WSL 中查找 docker 存储位置的详细信息，请参阅 WSL 存储库中的问题或这篇 StackOverflow 帖子。

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：将 Docker Desktop 与 WSL 2 配合使用的最佳做法
- Docker Desktop for Windows 反馈：提交问题
- VS Code 博客：用于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 成为开发人员的宠儿

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

项目 • 2024/03/28

Visual Studio 2022 引入了一个本机 C++ 工具集来开发适用于 Linux 的 Windows 子系统版本 2 (WSL 2)。 现在可在 Visual Studio 2022 版本 17.0 或更高版本中使用此工具集。

WSL 2 是适用于 Linux 的 Windows 子系统 (WSL) 的新的推荐版本。 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。 借助 Visual Studio 的 WSL 2 工具集，无需添加 SSH 连接即可使用 Visual Studio 在 WSL 2 发行版上生成和调试 C++ 代码。 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。 建议在 Visual Studio 中对所有 C++ 跨平台开发使用 CMake。 建议使用 CMake，因为它在 Windows、WSL 和远程系统上生成和调试同一项目。

如需查看关于本主题中的信息的视频演示，请观看视频：使用 WSL 2 发行版和 Visual Studio 2022 调试 C++。

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 会执行一个本地 `rsync` 命令，来将文件从 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，会自动出现此行为。 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。 若要了解有关详细信息，请参阅 Visual Studio 和 Visual Studio Code 中的 CMake 预设集成和在 Visual Studio 中使用 CMake 预设进行配置和生成。 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。 在稍后的步骤中，你将使用 Visual Studio 的 CMake 二进制部署来安装最新版本的 CMake。

1. 按照安装 WSL 中的说明来安装 WSL 和 WSL 2 发行版。

2. 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 g++ gdb make ninja-build rsync zip
```

上述 `apt` 命令会安装：

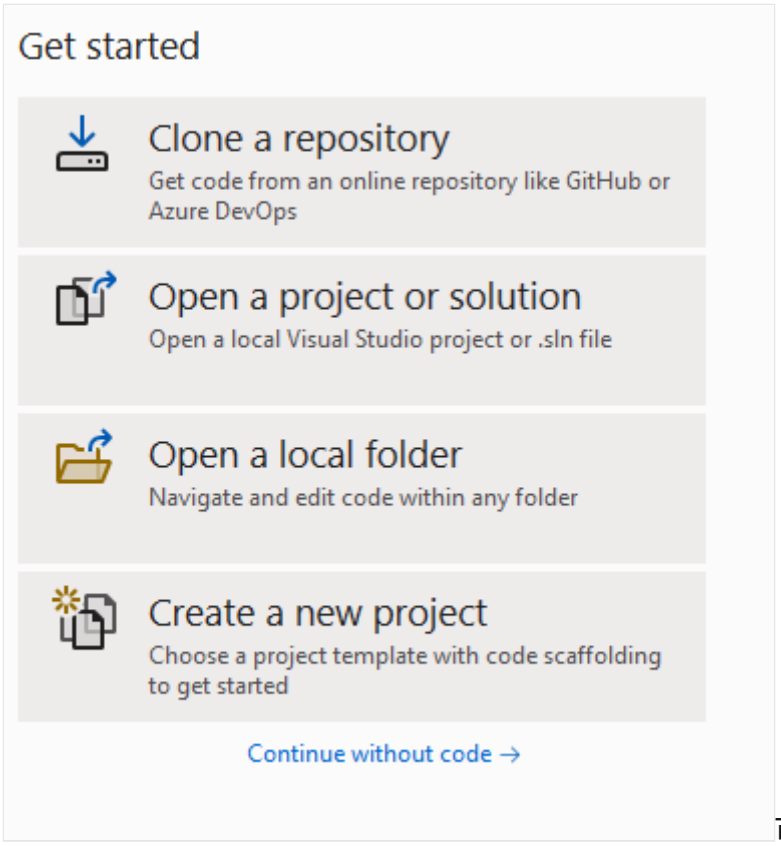
- C++ 编译器
- `gdb`
- `CMake`
- `rsync`
- `zip`
- 基础生成系统生成器

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。 还使用 Visual Studio 2022 版本 17.0 预览版 2 或更高版本。

Visual Studio 将 CMake 项目定义为在项目根目录中有一个 `CMakeLists.txt` 文件的文件夹。 在此演练中，你将使用 Visual Studio **CMake 项目**模板创建一个新的 CMake 项目：

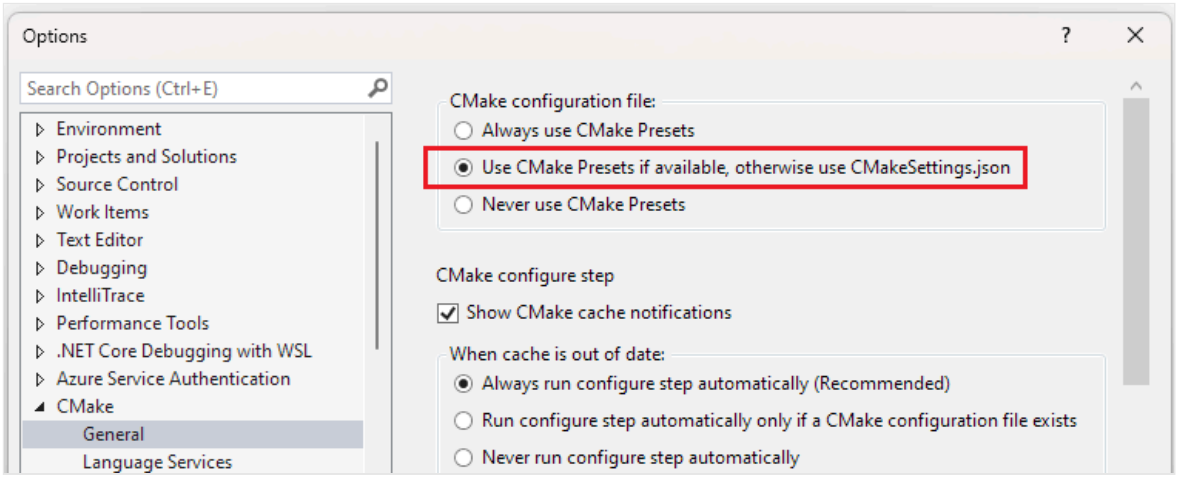
3. 在 Visual Studio 的“开始”屏幕中，选择“创建新项目”。



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。

4. 在“搜索模板”文本框中，键入“cmake”。 选择“CMake 项目”类型，然后选择“下一步”。 提供项目名称和位置，然后选择“创建”。

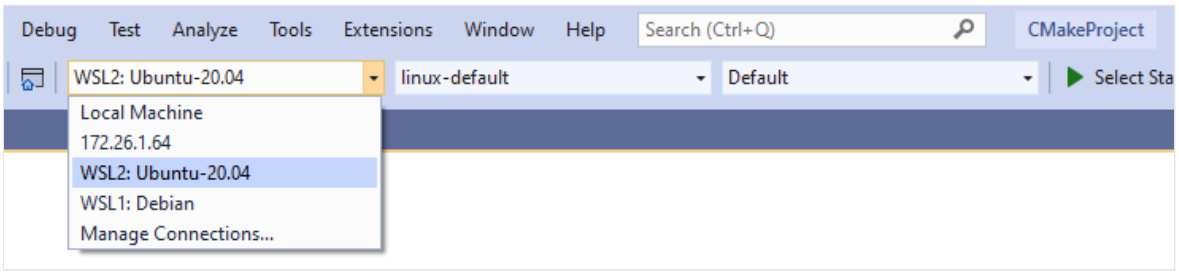
5. 启用 Visual Studio 的 CMake 预设集成。 选择“工具”>“选项”>“CMake”>“常规”。 选择“首选使用 CMake 预设进行配置、构建和测试”，然后选择“确定”。 相反，你可能已在项目的根目录下添加了一个 `CMakePresets.json` 文件。 有关详细信息，请参阅启用 CMake 预设集成。



6. 若要激活集成：在主菜单中，选择“文件”>“关闭文件夹”。 这会显示“开始”页面。 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。

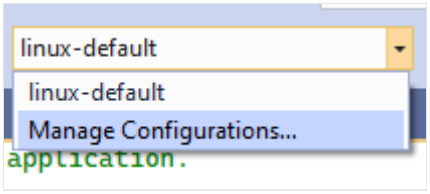
7. Visual Studio 主菜单栏上有 3 个下拉列表。 使用左侧的下拉列表可选择活动目标系统。 将在此系统上调用 CMake 来配置和生成项目。 Visual Studio 使用 `wsl -l -v`

查询 WSL 安装项。 在下图中，显示了 WSL2: Ubuntu-20.04，且已选中它作为目标系统。



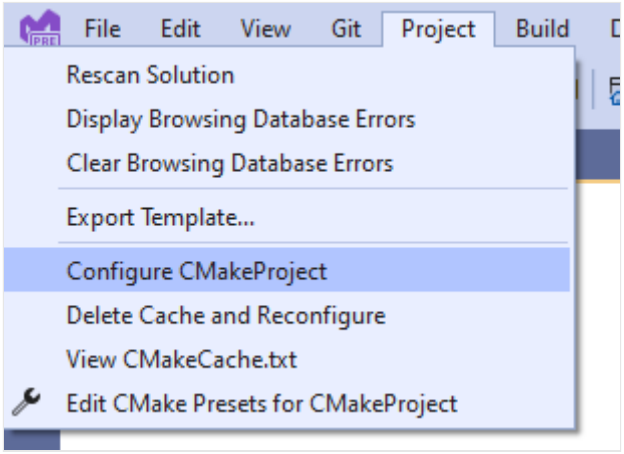
> **① 备注**
>
> 如果 Visual Studio 开始自动配置项目，请阅读步骤 11 来管理 CMake 二进制部署，然后继续执行以下步骤。 若要自定义此行为，请参阅**修改自动配置和缓存通知**。

8. 使用中间的下拉列表可选择活动配置预设。 配置预设会指示 Visual Studio 如何调用 CMake 和生成基础生成系统。 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。

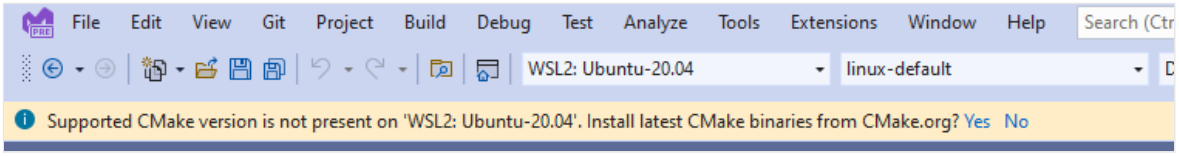


9. 使用右侧下拉列表可选择活动生成预设。 生成预设会指示 Visual Studio 如何调用生成。 在步骤 7 的插图中，活动生成预设是 Visual Studio 创建的默认生成预设。 若要详细了解生成预设，请参阅选择生成预设。

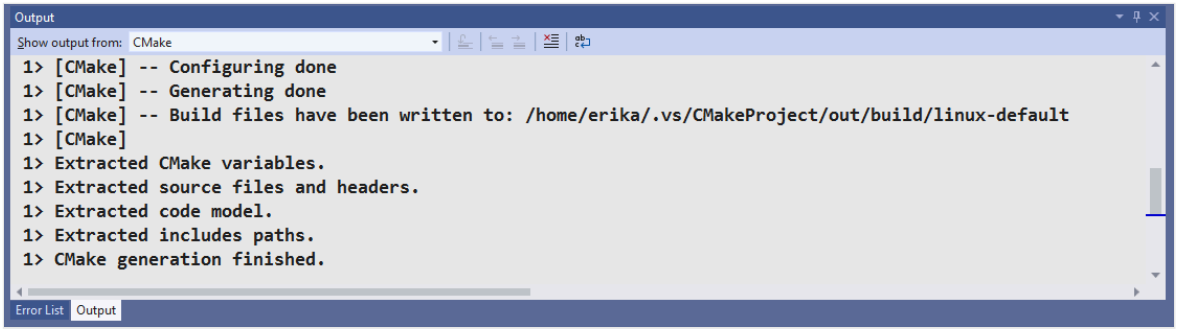
10. 在 WSL 2 上配置项目。 如果项目生成没有自动启动，请使用“项目”>“配置 <项目名称>”手动调用配置



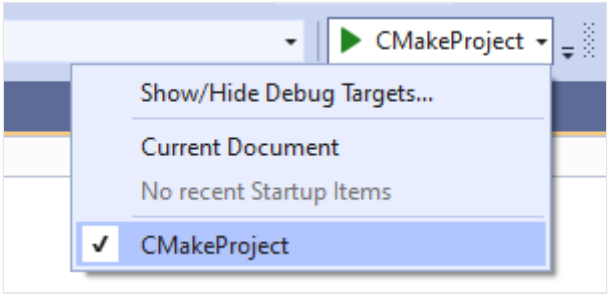
11. 如果未在 WSL 2 发行版上安装受支持的 CMake 版本，Visual Studio 将在主菜单功能区下提示部署最新版本的 CMake。 选择“是”，将 CMake 二进制文件部署到 WSL 2 发行版。



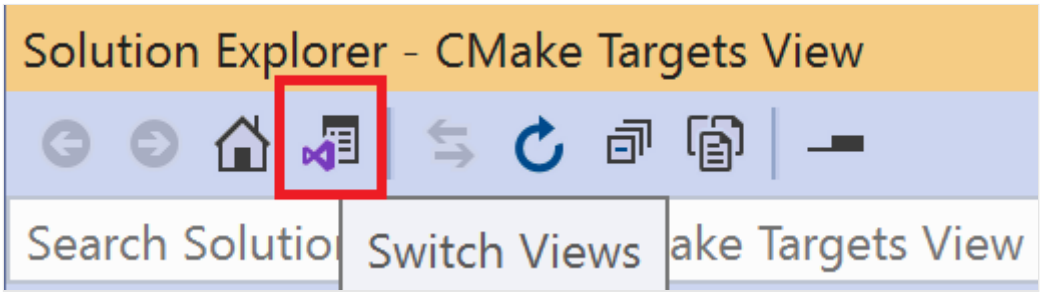
12. 确认已完成配置步骤，并且你在“**CMake**”窗格下的“**输出**”窗口中看到“**CMake 生成已完成**”消息。 生成文件会写入 WSL 2 发行版的文件系统中的某个目录。



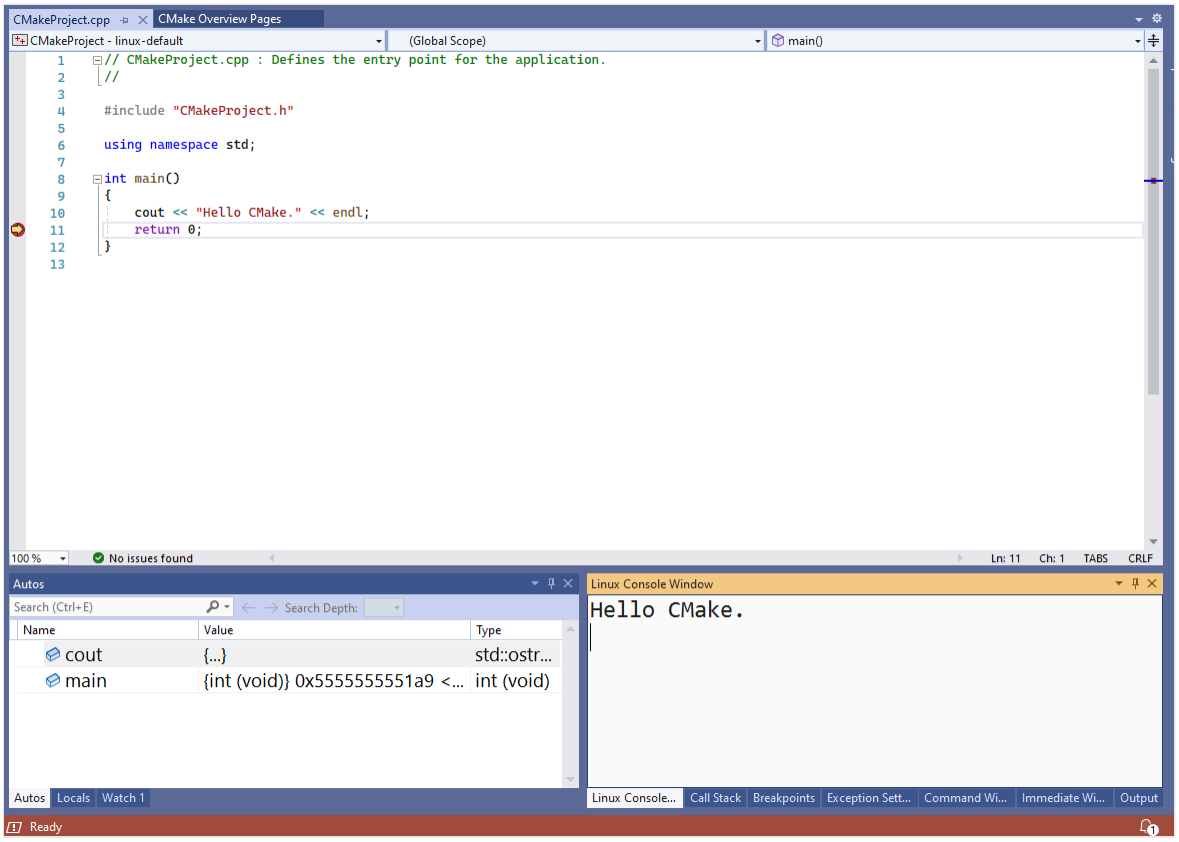
13. 选择活动调试目标。 调试下拉菜单列出了项目可使用的所有 CMake 目标。



14. 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。 还可在解决方案资源管理器中选择“视图选取器”按钮，导航到 CMake 目标视图，如以下屏幕截图中高亮所示：



15. 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."`）：



现在，你已使用 WSL 2 和 Visual Studio 2022 生成和调试了 C++ 应用。

高级 WSL 2 和 CMake 项目注意事项

Visual Studio 仅对将 `CMakePresets.json` 用作活动配置文件的 WSL 2 for CMake 项目提供本机支持。 若要从 `CMakeSettings.json` 迁移到 `CMakePresets.json`，请参阅启用 Visual Studio 中的 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。

如果 **forceWSL1Tooslet** 设置为 **true**，Visual Studio 不会在 WSL 文件系统中维护源文件的副本。 相反，它会访问已安装的 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 发行版一起使用，因为当项目文件改为存储在 Windows 文件系统中时，WSL 2 速度会变慢。 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` 中的 Visual Studio 远程设置供应商映射中，指定高级设置，例如 WSL 2 上项目要复制到的目录的路径、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，以提供本机 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。

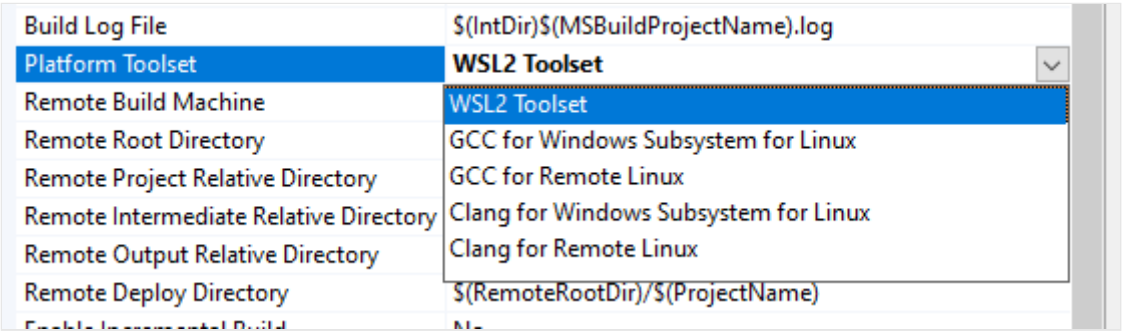
可在 `CMakePresets.json` 中的 Visual Studio 设置供应商映射中，更改 IntelliSense 模式或指定其他 IntelliSense 选项。 若要详细了解供应商映射，请参阅 Visual Studio 远程设置供应商映射。

基于 WSL 2 和 MSBuild 的 Linux 项目

建议在 Visual Studio 中对所有 C++ 跨平台开发使用 CMake，因为你可用它在 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，你可升级到 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”：



如果正在面向 WSL 2 发行版，但你不想使用 WSL 2 工具集，请在“平台工具集”下拉列表中，选择“适用于 Linux 的 Windows 子系统的 GCC”或“适用于 Linux 的 Windows 子系统的 Clang”工具集。 如果选择了上述任一工具集，则 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，来提供本机 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 发行版一起使用，因为当项目文件存储在 Windows 文件系统中时，WSL 2 速度会变慢。 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++
下载 Visual Studio 2022

在 Visual Studio 中创建 CMake Linux 项目
教程：在远程 Windows 计算机上调试 CMake 项目

反馈

此页面是否有帮助? 是 否

提供产品反馈 | 在 Microsoft Q&A 获取帮助

WSL 中 ML 的 GPU 加速入门

项目 • 2023/10/07

机器学习 (ML) 正在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，还是刚开始学习 ML，适用于 Linux 的 Windows 子系统 (WSL) 都为你提供了运行最常见和最常用的 GPU 加速 ML 工具的绝佳环境。

可通过多种不同的方法来设置这些工具。 例如，NVIDIA CUDA in WSL、TensorFlow-DirectML、PyTorch-DirectML 都提供了将 GPU 用于 ML 和 WSL 的不同方式。 若要详细了解选择其中一个而不是另一个的原因，请参阅 GPU 加速 ML 训练。

本指南介绍如何设置：

- 如果你有 NVIDIA 图形卡并运行示例 ML 框架容器，则为 NVIDIA CUDA
- 你的 AMD、Intel 或 NVIDIA 图形卡上的 TensorFlow-DirectML 和 PyTorch-DirectML

先决条件

- 确保你运行的是 Windows 11 或 Windows 10 版本 21H2 或更高版本。
- 安装 WSL，并为你的 Linux 发行版设置用户名和密码。

使用 Docker 设置 NVIDIA CUDA

1. 下载并安装适用于 NVIDIA GPU 的最新驱动程序

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

   ```
   Bash

   curl https://get.docker.com | sh
   ```

   ```
   Bash

   sudo service docker start
   ```

3. 如果你直接安装了 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)
```

Bash

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo gpg --
dearmor -o /usr/share/keyrings/nvidia-docker-keyring.gpg
```

Bash

```
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：

Bash

```
sudo apt-get update
```

Bash

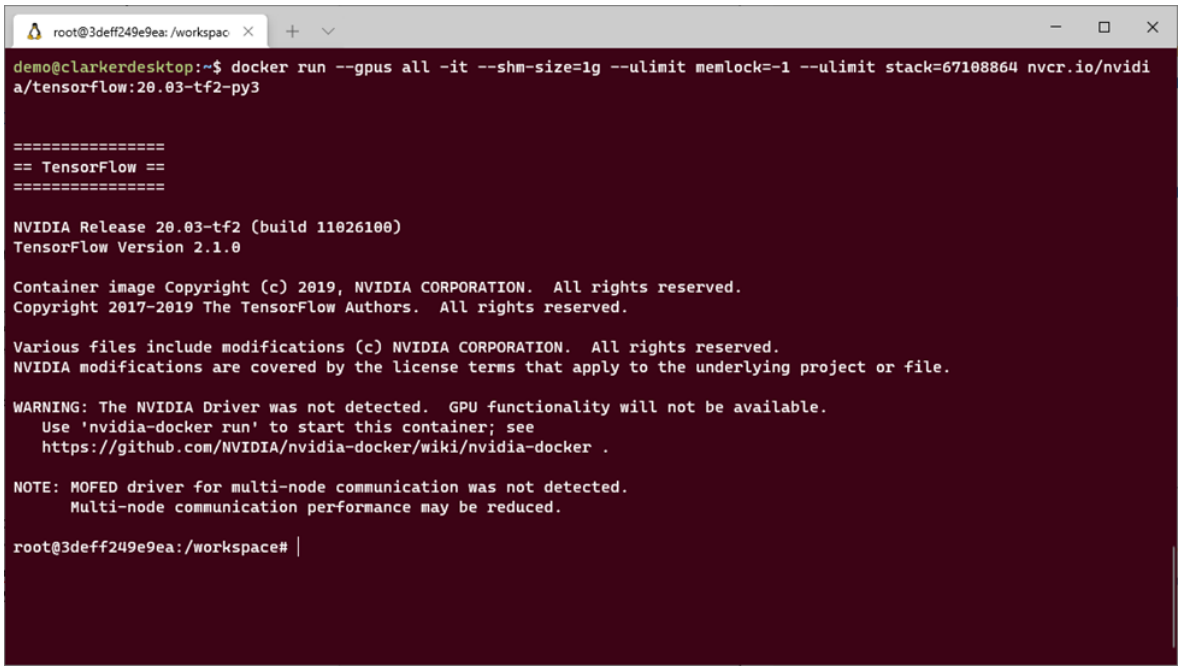
```
sudo apt-get install -y nvidia-docker2
```

4. 运行机器学习框架容器和示例。

若要运行机器学习框架容器并开始将 GPU 用于此 NVIDIA NGC TensorFlow 容器，请输入此命令：

Bash

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



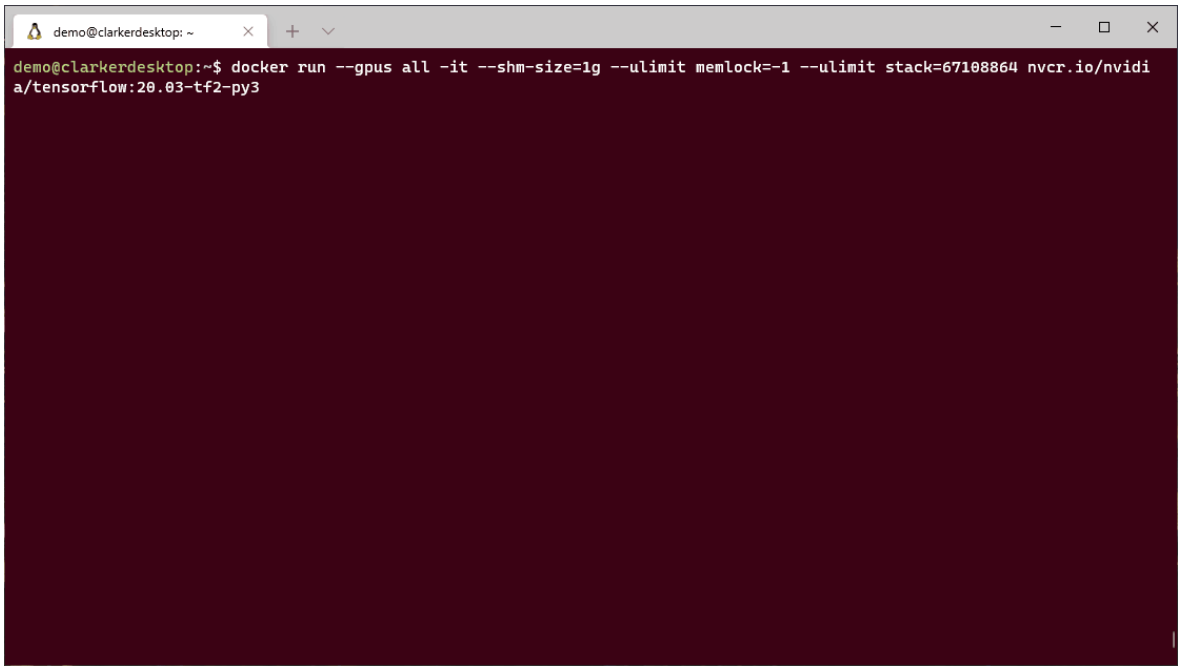
可以通过运行以下命令来运行内置在此容器中的预训练模型示例：

Bash

```
cd nvidia-examples/cnn/
```

Bash

```
python resnet.py --batch_size=64
```



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

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。 可以使用许多工具来设置虚拟 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

   bash Miniconda3-latest-Linux-x86_64.sh
   ```

   ```
   Bash

   conda create --name directml python=3.7 -y
   ```

   ```
   Bash

   conda activate directml
   ```

3. 安装由你选择的 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
   ```

   ```
   Bash

   pip install pytorch-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。

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

多个 GPU

如果计算机上有多个 GPU，你也可以在 WSL 中访问它们。 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你的 GPU 在设备管理器中显示的名称：

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

这将执行字符串匹配，因此，如果你将其设置为“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 中设置 TensorFlow 与 DirectML 的指南
- TensorFlow 与 DirectML 示例
- 在 WSL 中设置 PyTorch 与 DirectML 的指南
- PyTorch 与 DirectML 示例

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

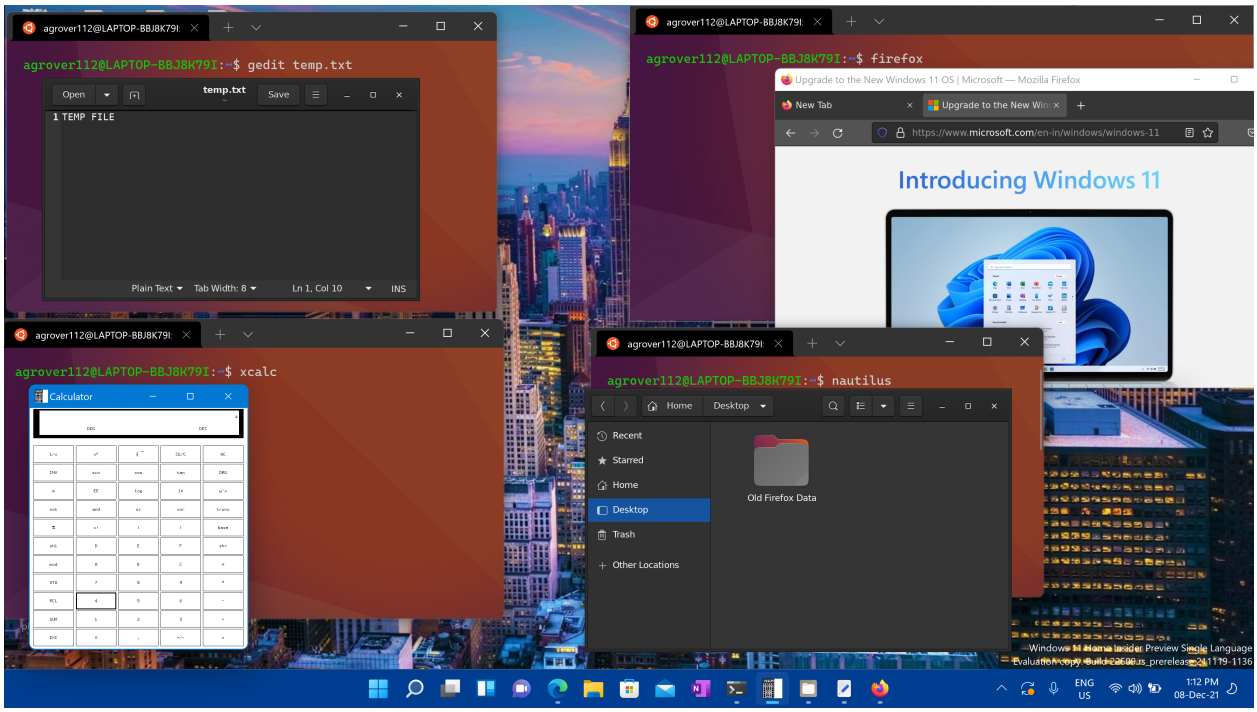
项目 • 2024/01/12

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

WSL 2 使 Linux GUI 应用程序在 Windows 上使用起来原生且自然。

- 从 Windows 的“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 应用和 Windows 应用之间切换
- 跨 Windows 应用和 Linux 应用剪切并粘贴

现在，可将 Windows 应用程序和 Linux 应用程序集成到工作流中，以获得无缝的桌面体验。



对 Linux GUI 应用的安装支持

先决条件

- 需要使用 Windows 10 **版本** 19044+ 或 Windows 11 才能使用此功能。
- 已安装适用于 vGPU 的驱动程序

若要运行 Linux GUI 应用，应首先安装下面与你的系统匹配的驱动程序。 这样，就可以使用虚拟 GPU (vGPU)，使你可受益于硬件加速 OpenGL 渲染。

- **Intel** GPU 驱动程序
- **AMD** GPU 驱动程序
- **NVIDIA** GPU 驱动程序

全新安装 - 没有以前的 WSL 安装

现在，可以在管理员 PowerShell 或 Windows 命令提示符中输入此命令，然后重启计算机来安装运行适用于 Linux 的 Windows 子系统 (WSL) 所需的全部内容。

PowerShell

```
wsl --install
```

计算机完成重启后，安装将继续进行，并要求你输入用户名和密码。 这将是 Ubuntu 发行版的 Linux 凭据。

现在，可开始在 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

如果已在计算机上安装 WSL，可通过从提升的命令提示符运行更新命令来更新到包含 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，更新才能生效。 可通过在 PowerShell 中运行关闭命令来重启 WSL。

   PowerShell

   ```
   wsl --shutdown
   ```

> **备注**
>
> Linux GUI 应用仅受 WSL 2 支持，并且不能用于为 WSL 1 配置的 Linux 发行版。 了解**如何将发行版从 WSL 1 更改为 WSL 2**。

运行 Linux GUI 应用

可从 Linux 终端运行以下命令，下载并安装这些常用的 Linux 应用程序。 如果使用的是不同于 Ubuntu 的发行版，则它可能使用与 apt 不同的包管理器。 安装 Linux 应用程序后，可在“开始”菜单中的发行版名称下找到它。 例如：`Ubuntu -> Microsoft Edge`。

> **备注**
>
> 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，因此可能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** 中提交问题。

更新发行版中的包

```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`

> **备注**
>
> **GNOME 文本编辑器** 取代 gedit 成为 Ubuntu 22.10 中 GNOME/Ubuntu 的默认文本编辑器。 如果运行的是较旧版本的 Ubuntu，并且想要使用 **gedit**（以前的默认文本编辑器），则使用 `sudo apt install gedit -y`。

安装 GIMP

GIMP 是一种免费的开源光栅图形编辑器，用于图像操作和图像编辑、自由形态绘图、不同图像文件格式之间的转码，以及更专业的任务。

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

若要启动，请输入：`gimp`

安装 Nautilus

Nautilus 也称为 GNOME Files，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 窗口管理系统，这是随它一起提供的各种应用和工具的集合，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`

安装适用于 Linux 的 Google Chrome

安装适用于 Linux 的 Google Chrome：

1. 将目录更改为 temp 文件夹：`cd /tmp`
2. 使用 wget 下载它：`wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb`
3. 安装包：`sudo apt install --fix-missing ./google-chrome-stable_current_amd64.deb`

*`--fix-missing` 选项用于修复安装过程中可能出现的缺少依赖项。 命令中的 `./` 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

在 Edge Insider 站点上查找有关如何使用命令行安装适用于 Linux 的 Microsoft Edge 浏览器的信息。 选择页面命令行安装部分下的“获取说明”。

若要启动，请输入：`microsoft-edge`

疑难解答

如果在启动 GUI 应用程序时遇到问题，请先查看此指南：诊断 WSLg 的“无法打开显示”一类的问题

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

项目 • 2023/07/13

如果你是使用 Node.js 的专业人士、发现性能速度和系统调用兼容性很重要、想要运行利用 Linux 工作区的 Docker 容器并避免维护 Linux 和 Windows 生成脚本，或只是倾向于使用 Bash 命令行，则需要在适用于 Linux 的 Windows 子系统（更具体地说就是 WSL 2）上安装 Node.js。

通过适用于 Linux 的 Windows 子系统 (WSL)，可以安装首选的 Linux 分发版（默认为 Ubuntu），以便在开发环境（编写代码的位置）和生产环境（部署代码的服务器）之间保持一致性。

> ⓘ **备注**
>
> 如果你不熟悉如何使用 Node.js 进行开发，但想要快速启动和运行以便学习，可**在 Windows 安装 Node.js**。 如果计划使用 Windows Server 生产环境，此建议同样适用。

安装 WSL 2

WSL 2 是适用于 Windows 的最新版本，我们建议用于专业的 Node.js 开发工作流。 要启用和安装 WSL 2，请按照 WSL 安装文档中的步骤操作。 这些步骤将包含选择 Linux 发行版（例如 Ubuntu）。

安装 WSL 2 和 Linux 发行版后，打开 Linux 发行版（可在 Windows 的开始菜单中找到），并使用命令 `lsb_release -dc` 查看版本和代码名称。

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`。

安装 Windows 终端（可选）

Windows 终端是经过改进的命令行 shell，使你可以运行多个选项卡，从而可以在 Linux 命令行、Windows 命令提示符、PowerShell、Azure CLI 或你喜欢使用的任何选项卡之间进行切换。 还可以创建自定义键绑定（打开或关闭选项卡、复制和粘贴的快捷键），使

用搜索功能，以及自定义终端的主题（配色方案、字体格式和大小以及背景图像/模糊/透明度）等。 在 Windows 终端文档中了解详细信息。

使用 Microsoft Store 安装 Windows 终端：通过 Microsoft Store 进行安装时，将自动处理更新。

安装 nvm、node.js 和 npm

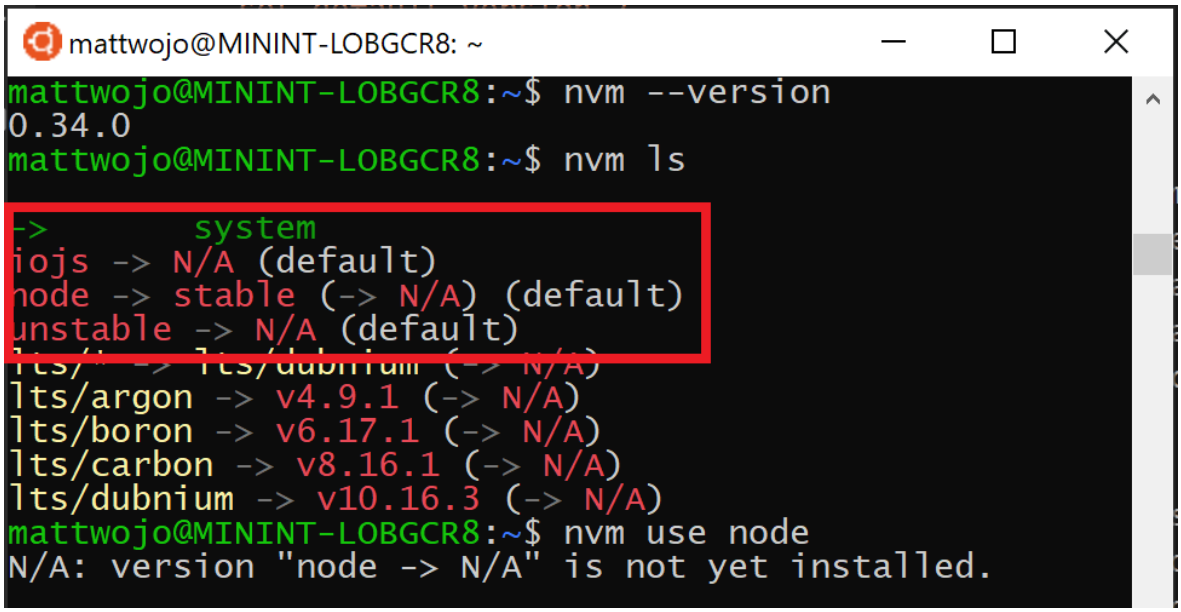
除了选择在 Windows 还是在 WSL 上安装，安装 Node.js 时还要作出其他选择。 建议使用版本管理器，因为版本变更速度非常快。 你可能需要根据所使用的不同项目的需求在多个版本的 Node.js 之间进行切换。 Node 版本管理器（通常称为 nvm）是安装多个版本的 Node.js 的最常见方法。 我们将演练安装 nvm 的步骤，然后使用它来安装 Node.js 和节点包管理器 (npm)。 下一节中还会介绍供考虑的替代版本管理器。

> **ⓘ 重要**
>
> 在安装版本管理器之前，始终建议从操作系统中删除 Node.js 或 npm 的任何现有安装，因为不同的安装类型可能会导致出现奇怪和混淆的冲突。 例如，可以使用 Ubuntu 的 `apt-get` 命令安装的 Node 版本当前已过时。 有关删除先前安装的帮助，请参阅**如何从 ubuntu 中删除 node.js**。

1. 打开 Ubuntu 命令行（或所选的发行版）。
2. 使用以下命令安装 cURL（用于在命令行中从 Internet 下载内容的工具）：`sudo apt-get install curl`
3. 使用以下命令安装 nvm：`curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/master/install.sh | bash`

   > **ⓘ 备注**
   >
   > 使用 cURL 安装较新版本的 NVM 将替换旧版本，并使已使用 NVM 进行安装的 Node 版本保持不变。 有关详细信息，请参阅**有关 NVM 最新版本信息的 GitHub 项目页面**。

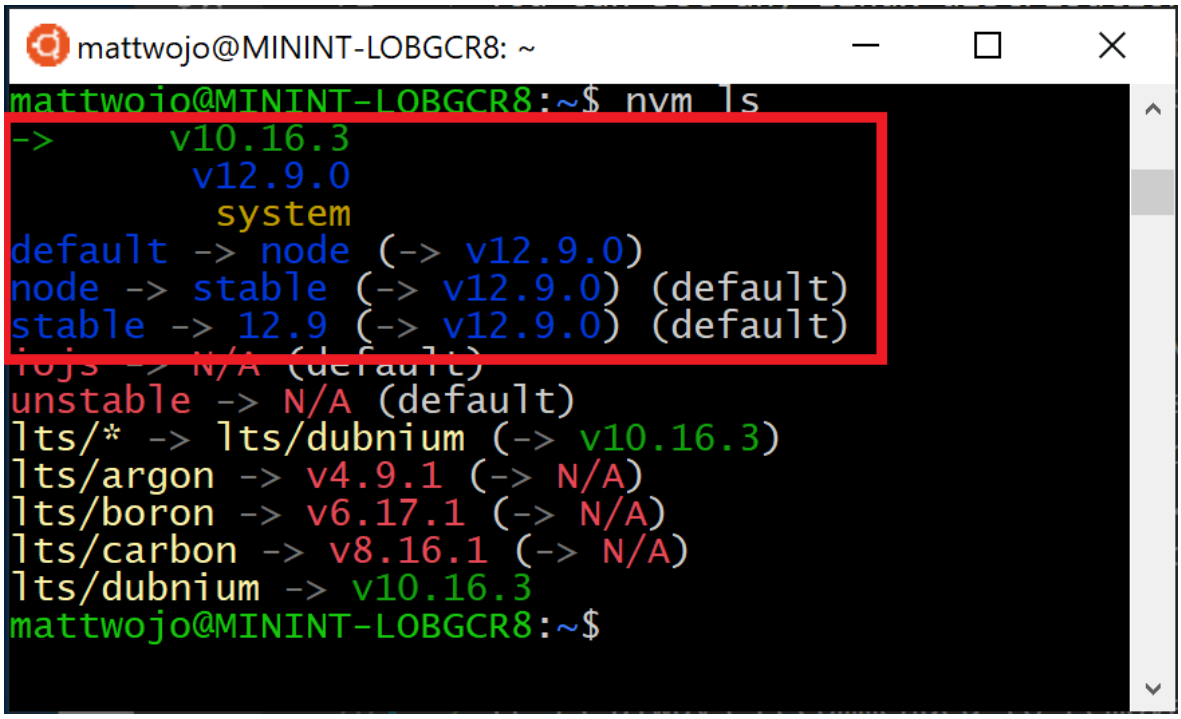
4. 若要验证安装，请输入：`command -v nvm`。此命令应返回“nvm”，如果你收到“找不到命令”或根本没有响应，请关闭当前终端，将其重新打开，然后重试。 在 nvm github 存储库中了解详细信息。
5. 列出当前安装的 Node 版本（此时应为无）：`nvm ls`



6. 安装 Node.js 的当前版本和稳定的 LTS 版本。 后面的步骤将介绍如何使用 `nvm` 命令在 Node.js 的活动版本之间切换。

   - 安装 Node.js 的当前稳定的 LTS 版本（推荐用于生产应用程序）：`nvm install --lts`
   - 安装 Node.js 的当前版本（用于测试最新的 Node.js 功能和改进，但更容易出现问题）：`nvm install node`

7. 列出安装的 Node 版本：`nvm ls`。现在应会看到刚安装的两个版本。



8. 使用以下命令验证 Node.js 是否已安装，以及是否为当前默认版本：`node --version`。 然后使用以下命令验证是否也有 npm：`npm --version`（还可以使用 `which node` 或 `which npm` 来查看用于默认版本的路径）。

9. 若要更改要用于项目的 Node.js 版本，请创建新的项目目录 `mkdir NodeTest`，输入目录 `cd NodeTest`，然后输入 `nvm use node` 切换到当前版本，或输入 `nvm use --lts` 切换到 LTS 版本。 你还可以使用已安装的任何其他版本的特定数量，如 `nvm use v8.2.1`。 （若要列出 Node.js 的所有可用版本，请使用以下命令：`nvm ls-remote`）。

如果要使用 NVM 安装 Node.js 和 NPM，则不需要使用 SUDO 命令来安装新包。

替代版本管理器

虽然 nvm 目前是最常用的节点版本管理器，但需要考虑一些替代版本管理器：

- n 是长期存在的 `nvm` 替代方法，该方法使用略微不同的命令来完成相同的操作，并通过 `npm` 而不是 bash 脚本来安装。
- fnm 是较新的版本管理器，它声称比 `nvm` 快得多。 （它还使用 Azure 管道。）
- Volta 是来自 LinkedIn 团队的新版本管理器，它声称改进了速度和跨平台支持。
- asdf-vm 是适用于多种语言的单个 CLI，例如将 ike gvm、nvm、rbenv & 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

建议将 Visual Studio Code 与 Node.js 项目的 Remote 开发扩展包一起使用。 这会将 VS Code 拆分为“客户端-服务器”体系结构，使客户端（VS Code 用户界面）在 Windows 操作系统上运行，而使服务器（你的代码、Git、插件等）在 WSL Linux 发行版中远程运行。

> **备注**
>
> 这种“远程”方案与你可能习惯的方案有点不同。 WSL 支持实际的 Linux 发行版，此发行版中的项目代码与 Windows 操作系统分开运行，但仍然在本地计算机上运行。 Remote-WSL 扩展与 Linux 子系统连接，就像它是一个远程服务器一样，尽管它没有在云中运行... 它仍在 WSL 环境中（与 Windows 一起运行）的本地计算机上运行。

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

其他代码编辑器（如 IntelliJ、Sublime Text 和 Brackets 等）也可在 WSL 2 Node.js 开发环境下工作，但是可能没有 VS Code 提供的那种远程功能。 这些代码编辑器在访问 WSL 共享网络位置 (\wsl$\Ubuntu\home) 时可能遇到问题，并将尝试使用 Windows 工具生成 Linux 文件，这可能不是你所希望的。 VS Code 中的 Remote-WSL 扩展为你处理此兼容性问题，而对于其他 IDE，你可能需要设置一个 X 服务器。 即将推出在 WSL 运行 GUI 应用的支持，如代码编辑器 IDE。

基于终端的文本编辑器（vim、emacs、nano）还有助于从控制台内部快速更改。 Emacs、Nano 或 Vim：明智地选择基于终端的文本编辑器一文清楚地介绍了它们之间的差异以及每个文本编辑器的使用方法。

安装 VS Code 和 Remote-WSL 扩展：

1. 下载和安装适用于 Windows 的 VS Code。 VS Code 也适用于 Linux，但适用于 Linux 的 Windows 子系统不支持 GUI 应用，因此需要在 Windows 上安装它。 不必担心，仍可以使用 Remote - WSL 扩展与 Linux 命令行和工具集成。

2. 在 VS Code 上安装 Remote - WSL 扩展。 这使你可以将 WSL 用作集成开发环境，并且会为你处理兼容性和路径。 了解详细信息。

> **ⓘ 重要**
>
> 如果已安装 VS Code，则需要确保具有 **1.35 5 月版本**或更高版本，以便安装 **Remote - WSL 扩展**。 建议不要在没有 Remote - WSL 扩展的情况下在 VS Code 中使用 WSL，因为会失去对自动完成、调试、linting 等的支持。趣味事实：此 WSL 扩展安装在 $HOME/.vscode-server/extensions 中。

有用的 VS Code 扩展

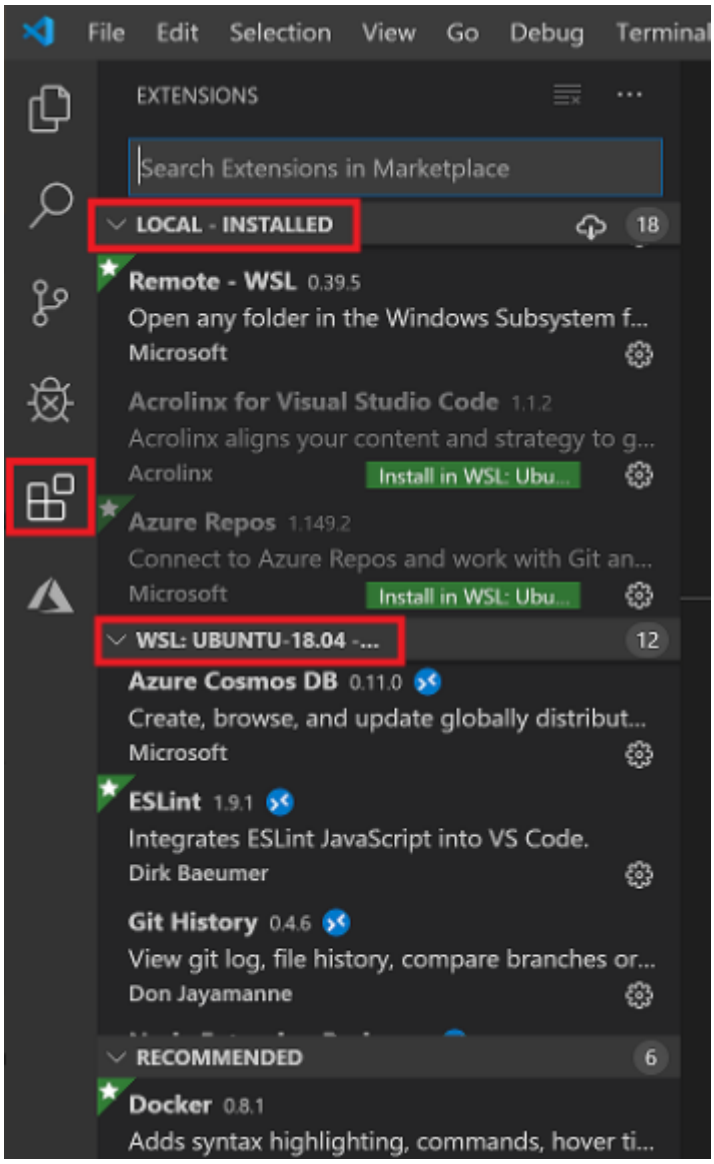
虽然 VS Code 附带了许多用于 Node.js 开发的功能，但有一些有用的扩展可考虑在 Node.js 扩展包中安装。 全部安装或选择对你最有用的包。

安装 Node.js 扩展包：

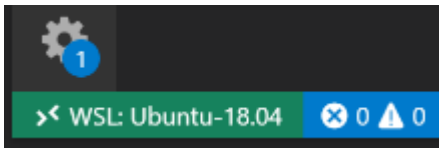
1. 在 VS Code 中打开“扩展”窗口 (Ctrl+Shift+X)。

   现在，“扩展”窗口分为三个部分（因为你已安装 Remote-WSL 扩展）。

   - “本地 - 已安装”：要与 Windows 操作系统一起使用而安装的扩展。
   - “WSL:Ubuntu-18.04-已安装”：要与 Ubuntu 操作系统 (WSL) 一起使用而安装的扩展。
   - “推荐”：根据当前项目中的文件类型由 VS Code 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）。 将为 VS Code 的本地实例或 WSL 实例安装扩展，具体取决于打开当前项目的位置。 可以通过选择“VS Code”窗口左下角的远程链接进行判断（以绿色显示）。 将向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。



可能需要考虑的几个附加扩展包括：

- JavaScript 调试程序：在服务器端通过 Node.js 完成开发后，需要开发并测试客户端。 此扩展是基于 DAP 的 JavaScript 调试程序。 它调试 Node.js、Chrome、Edge、WebView2、VS Code 扩展等。
- 来自其他编辑器的键映射：如果是从另一个文本编辑器（如 Atom、Sublime、Vim、eMacs、Notepad++ 等）进行转换，则这些扩展可帮助你的环境对此进行适应。

- 设置同步：可以使用 GitHub 在不同安装之间同步 VS Code 设置。 如果在不同的计算机上工作，这有助于在它们之间保持一致的环境。

设置 Git（可选）

要为 WSL 上的 Node.js 项目设置 Git，请参阅 WSL 文档中的文章开始在适用于 Linux 的 Windows 子系统上使用 Git。

开始使用 Linux 和 Bash

项目 • 2023/10/07

本教程将帮助 Linux 新手使用通过 WSL 默认安装的 Linux Ubuntu 发行版来安装和更新包，以及通过 Bash 命令行使用一些基本命令。

安装和更新软件

可以使用你正在运行的发行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”来更新可用软件的列表。 然后，可以使用“sudo apt-get install”命令并后跟要安装的程序的名称来直接获取软件：

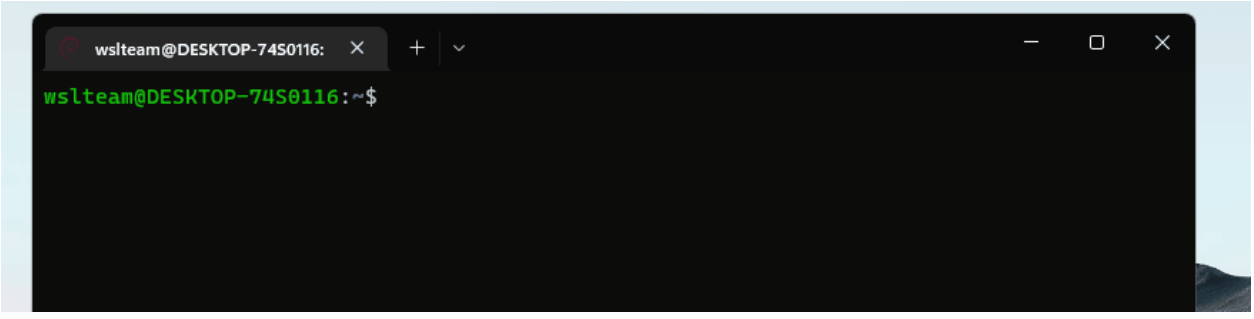
Bash

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

若要更新已安装的程序，你可以运行：

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的包管理器的安装命令。 例如，Arch Linux 的主要包管理器称为 **pacman**，它的安装命令应为 `sudo pacman -S <app_name>`。 OpenSuse 的主要包管理器称为 **Zypper**，它的安装命令应为 `sudo zypper install <app_name>`。 Alpine 的主要包管理器称为 **apk**，它的安装命令应为 `sudo apk add <app_name>`。 Red Hat 发行版（如 CentOS）的两个主要包管理器是 **YUM 和 RPM**，它们的安装命令可以是 `sudo yum install`

`<app_name>` 或 `sudo rpo -i <app_name>`。 请参阅你正在使用的发行版的文档，了解可用于安装和更新软件的工具。

处理文件和目录

若要查看你当前位于的目录的路径，请使用“pwd”命令：

Bash

```
pwd
```

若要创建新目录，请使用“mkdir”命令，后跟要创建的目录的名称：

Bash

```
mkdir hello_world
```

若要更改目录，请使用“cd”命令，后跟你要导航到的目录的名称：

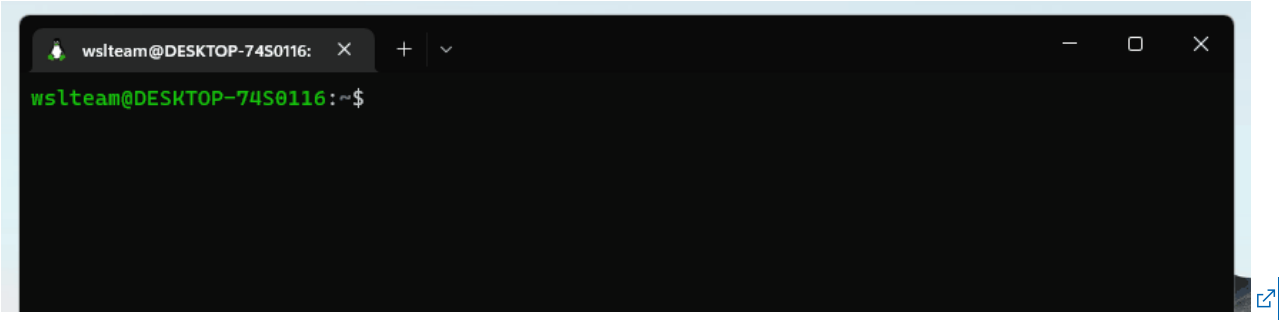
Bash

```
cd hello_world
```

若要查看当前目录中的内容，请在命令行中键入“ls”：

Bash

```
ls
```



默认情况下，“ls”命令仅输出所有文件和目录的名称。 若要获取其他信息，例如上次修改文件的时间或文件权限，请使用标志“-l”：

Bash

```
ls -l
```

可以通过“touch”命令并后跟要创建的文件的名称来创建新文件：

```Bash
touch hello_world.txt
```

可以使用任何下载的图形文本编辑器或 VS Code Remote–WSL 扩展来编辑文件。 可在此处了解有关 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。 许多发行版都安装了一个或多个这种程序，但你始终可以按照上面的指南中概述的安装说明来安装这些程序。

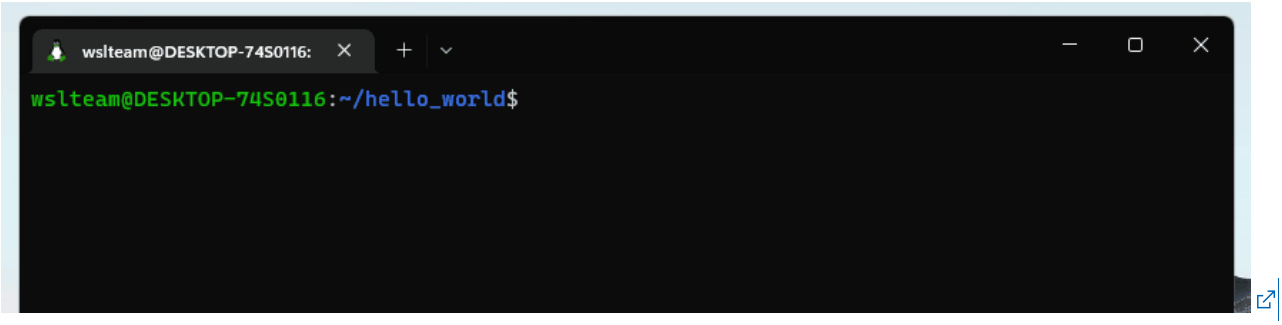
若要使用你偏好的编辑方法编辑文件，只需运行程序名称，后跟要编辑的文件的名称：

```Bash
code hello_world.txt
```

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

若要在命令行中查看文件的内容，请使用“cat”命令，后跟要读取的文件：

```Bash
cat hello_world.txt
```



使用管道和重定向操作符

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 会将输出从 lhscmd 定向到 rhscmd。 可以通过多种方式使用管道，以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。 以下面的 fruits.txt 为例：

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，“cat”命令的输出会被发送到标准输出，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”，它是一个实用的命令，用于搜索特定搜索字符串的输入。

Bash

```
cat fruits.txt | grep P

Pear

Plum

Peach
```

还可以使用重定向操作符（如“>”）将输出传递到文件或流。 例如，如果要使用 fruit.txt 排序后的内容创建一个新的 .txt 文件：

Bash

```
cat fruits.txt | sort > sorted_fruit.txt
```

Bash

```
$ cat sorted_fruit.txt

Apple

Banana

Kiwi

Orange

Peach

Pear

Plum

Strawberry
```

默认情况下，sort 命令的输出会被发送到标准输出，但是，“>”操作符允许我们改为将输出重定向到名为 sorted_fruits.txt 的新文件中。

你可以通过许多有趣的方式使用管道和重定向操作符，从而直接通过命令行更高效地完成任务。

推荐内容

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

跨 Windows 和 Linux 文件系统工作

项目 • 2023/03/21

在 Windows 和 Linux 文件系统之间工作时，需要牢记一些注意事项。 本指南介绍了一部分示例，包括一些混合使用基于 Windows 和 Linux 的命令的互操作性支持示例。

跨文件系统的文件存储和性能

建议不要跨操作系统使用文件，除非有这么做的特定原因。 若想获得最快的性能速度，请将文件存储在 WSL 文件系统中，前提是在 Linux 命令行（Ubuntu、OpenSUSE 等）中工作。 如果使用 Windows 命令行（PowerShell、命令提示符）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：`\\wsl$\Ubuntu\home\<user name>\Project`
- 而不使用 Windows 文件系统根目录：`/mnt/c/Users/<user name>/Project$` 或 `C:\Users\<user name>\Project`

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

在 Windows 文件资源管理器中查看当前目录

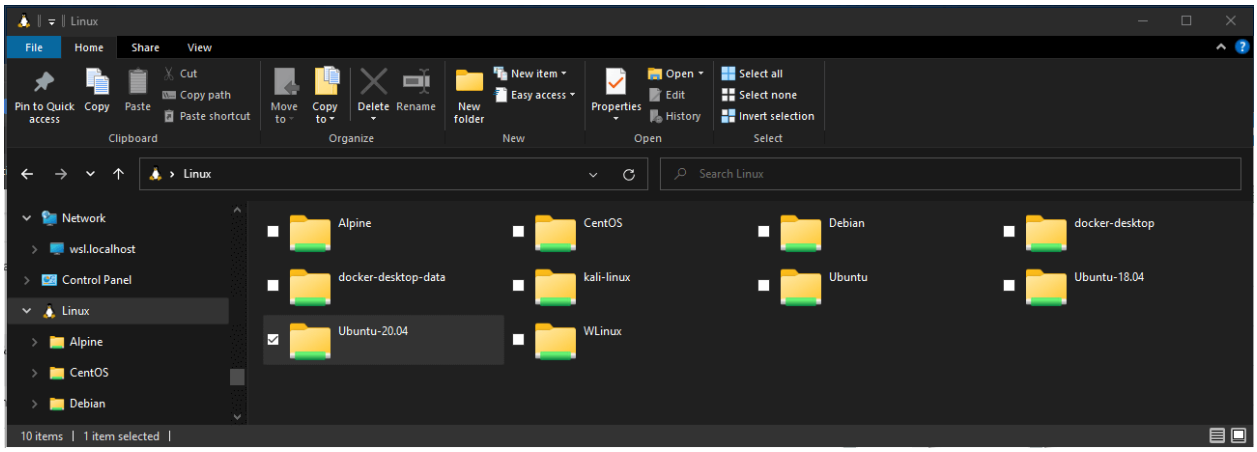
可使用以下命令从命令行打开 Windows 文件资源管理器，以查看存储文件的目录：

```
Bash

explorer.exe .
```

另外，还可以使用 `powershell.exe /c start .` 命令。请确保在命令的末尾添加句点以打开当前目录。

若要在 Windows 文件资源管理器中查看所有可用的 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

区分大小写确定在文件名或目录中是将大写 (FOO.txt) 和小写 (foo.txt) 字母作为不同项（区分大小写）还是等效项（不区分大小写）进行处理。 Windows 和 Linux 文件系统处理区分大小写的方式不同 - Windows 不区分大小写，而 Linux 区分大小写。 若要详细了解如何调整区分大小写（尤其是在使用 WSL 装载磁盘时），请参阅调整区分大小写操作说明文章。

Windows 和 Linux 命令之间的互操作性

借助 WSL，Windows 和 Linux 工具和命令可互换使用。

- 从 Linux 命令行（即 Ubuntu）运行 Windows 工具（即 notepad.exe）。
- 从 Windows 命令行（即 PowerShell）运行 Linux 工具（即 grep）。
- 在 Windows 与 Windows 之间共享环境变量。 （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 ->
```

以这种方式调用二进制文件：

- 使用当前 CMD 或 PowerShell 提示符中提到的同一工作目录。

- 以 WSL 默认用户的身份运行。
- 拥有与调用方进程和终端相同的 Windows 管理权限。

`wsl`（或 `wsl.exe`）后面的 Linux 命令的处理方式与 WSL 中运行的任何命令的处理方式类似。 可以执行 sudo、管道处理和文件重定向等操作。

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，将会列出默认的 Linux 分发版用户名，并将要求你输入密码。 正确输入密码后，分发版将下载更新。

混合 Linux 和 Windows 命令

下面是几个使用 PowerShell 混合 Linux 和 Windows 命令的示例。

若要使用 Linux 命令 `ls -la` 列出文件，并使用 PowerShell 命令 `findstr` 来筛选包含“git”的单词的结果，请组合这些命令：

PowerShell

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

若要使用 PowerShell 命令 `dir` 列出文件，并使用 Linux 命令 `grep` 来筛选包含“git”的单词的结果，请组合这些命令：

PowerShell

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

若要使用 Linux 命令 `ls -la` 列出文件，并使用 PowerShell 命令 `> out.txt` 将该列表输出到名为“out.txt”的文本文件，请组合这些命令：

PowerShell

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

传入 `wsl.exe` 的命令将按原样转发到 WSL 进程。 文件路径必须以 WSL 格式指定。

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

PowerShell

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

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

以这种方式运行的应用程序具有以下属性：

- 按 WSL 命令提示保留工作目录（大部分情况下是这样 -- 下面所述的情况除外）。
- 拥有与 WSL 进程相同的权限。
- 以活动 Windows 用户的身份运行。
- 显示在 Windows 任务管理器中，就如同直接从 CMD 提示符执行的一样。

在 WSL 中运行的 Windows 可执行文件的处理方式类似于本机 Linux 可执行文件 - 管道处理、重定向，甚至后台处理都可按预期方式工作。

若要运行 Windows 工具 `ipconfig.exe`，请使用 Linux 工具 `grep` 筛选“IPv4”结果，并使用 Linux 工具 `cut` 删除列字段，请从 Linux 分发版（例如 Ubuntu）输入：

Bash

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

让我们尝试一个混合使用 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，并使用 Windows PowerShell 工具 `findstr.exe` 来筛选结果，以便仅在结果中显示 `foo.txt` 文件：

Bash

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

Windows 工具必须包含文件扩展名，匹配文件大小写，并且可执行。 包含批处理脚本的不可执行文件。 `dir` 等 CMD 本机命令可与 `cmd.exe /C` 命令一起运行。

例如，通过输入以下命令列出 Windows 文件系统 C:\ 目录的内容：

```
Bash

cmd.exe /C dir
```

或者使用 `ping` 命令将回显请求发送到 microsoft.com 网站：

```
Bash

ping.exe www.microsoft.com
```

参数将按原样传递到 Windows 二进制文件。 例如，以下命令将通过 `notepad.exe` 打开 `C:\temp\foo.txt`：

```
Bash

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

以下命令也会起作用：

```
Bash

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

通过 WSLENV 在 Windows 与 WSL 之间共享环境变量

WSL 和 Windows 共享一个特殊环境变量 `WSLENV`（为了桥接 Windows 和 WSL 上运行的 Linux 分发版而创建）。

`WSLENV` 变量的属性：

- 它是共享的；它同时在 Windows 和 WSL 环境中存在。
- 它是要在 Windows 与 WSL 之间共享的环境变量列表。
- 它可以设置环境变量的格式，使其能够在 Windows 和 WSL 中正常运行。

- 它可以帮助 WSL 和 Win32 之间的流。

> **备注**
>
> 在 17063 以前，只有 WSL 可访问的 Windows 环境变量是 `PATH`（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

`WSLENV` 中有四个标志可以影响该环境变量的转换方式。

`WSLENV` 标志：

- `/p` - 在 WSL/Linux 样式路径与 Win32 路径之间转换路径。
- `/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。

可按需组合标志。

阅读有关 WSLENV 的详细信息，包括将 WSLENV 的值设置为其他预定义环境变量串联的常见问题解答和示例，每个示例都带有一个斜杠后缀，斜杠后跟标志，用于指定应如何转换值以及如何使用脚本传递变量。 本文还提供使用 编程语言 设置开发环境的示例，该示例配置为在 WSL 和 Win32 之间共享 GOPATH。

禁用互操作性

用户可以使用 root 身份运行以下命令，禁用针对单个 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，或者以 root 身份运行以下命令：

```
Bash

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

每次切换 WSL 会话后，禁用互操作的结果不会持久保留 -- 启动新会话后，会再次启用互操作。

WSL 中的高级设置配置

项目 • 2024/02/03

wsl.conf 和 .wslconfig 文件用于针对每个发行版（`wsl.conf`）和全局跨所有 WSL 2 发行版（`.wslconfig`）配置高级设置选项。 本指南将介绍每个设置选项、何时使用每种文件类型、存储文件的位置、示例设置文件和提示。

wsl.conf 和 .wslconfig 之间有什么差别?

可以为已安装的 Linux 发行版配置设置，使它们在你每次启动 WSL 时自动应用，有两种方法：

- .wslconfig 用于在 WSL 2 上运行的所有已安装发行版中配置**全局设置。**
- wsl.conf 用于为在 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。**

这两种文件类型都用于配置 WSL 设置，但存储文件的位置、配置的范围、可配置的选项类型，以及运行发行版的 WSL 版本都会影响应选择的文件类型。

WSL 1 和 WSL 2 使用不同的体系结构运行，并会影响配置设置。 WSL 2 作为轻型虚拟机 (VM) 运行，因此请使用让你能够控制内存或处理器使用量的虚拟化设置（使用 Hyper-V 或 VirtualBox 的话可能会比较熟悉）。 检查正在运行的 WSL 的版本。

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。 在重启之前，必须等子系统停止，给它足够的时间来获取你的更改。 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` 是重启 WSL 2 发行版的快速路径，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>` 来终止立即运行的特定发行版。

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。
- 若要访问已安装的发行版的 `/etc` 目录，请使用发行版的命令行和 `cd /` 访问根目录，然后使用 `ls` 列出文件或使用 `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`。

> ⓘ **备注**
>
> 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。 有关详细信息，请参阅博客公告。

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

Bash

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

然后，需要通过 PowerShell 使用 `wsl.exe --shutdown` 来关闭 WSL 发行版以重启 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` 命令进行确认，该命令会显示服务的状态。

自动装载设置

wsl.conf 节标签：`[automount]`

展开表

key	value	default	说明
enabled	boolean	是	`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	boolean	是	`true` 设置启动 WSL 时要处理的 `/etc/fstab`。 /etc/fstab 是可在其中声明其他文件系统的文件，类似于 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	string	`/mnt/`	设置固定驱动器要自动装载到的目录。 默认情况下，此项设置为 `/mnt/`，因此 Windows 文件系统 C 驱动器会装载到 `/mnt/c/`。 如果将 `/mnt/` 更改为 `/windir/`，则你应会看到固定的 C 驱动器装载到 `/windir/c`。
选项	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	空字符串	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，此项默认为 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，此项默认为 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	0.00
dmask	要对所有目录排除的权限的八进制掩码	0.00
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 会将 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。 如果此值用于指定不同的 gid 或 uid 选项，则默认用户值将被覆盖。 否则，将始终追加默认值。

用户文件创建模式掩码 (umask) 会为新创建的文件设置权限。 默认值为 022，只有你可以写入数据，但任何人都可以读取数据。 可以更改值以反映不同的权限设置。 例如，`umask=077` 会将权限更改为完全私密，其他用户无法读取或写入数据。 若要进一步指定权限，也可以使用 fmask（文件）和 dmask（目录）。

> **备注**
>
> 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

key	value	default	说明
generateHosts	boolean	`true`	`true` 将 WSL 设置为生成 `/etc/hosts`。 `hosts` 文件包含主机名对应的 IP 地址的静态映射。
generateResolvConf	boolean	`true`	`true` 将 WSL 设置为生成 `/etc/resolv.conf`。 `resolv.conf` 包含能够将给定主机名解析为其 IP 地址的 DNS 列表。
hostname	string	Windows 主机名	设置要用于 WSL 发行版的主机名。

互操作设置

wsl.conf 节标签：`[interop]`

这些选项在预览体验成员内部版本 17713 和更高版本中可用。

展开表

key	value	default	说明
enabled	boolean	`true`	设置此键可确定 WSL 是否支持启动 Windows 进程。
appendWindowsPath	boolean	`true`	设置此键可确定 WSL 是否会将 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

key	value	default	说明
默认值	string	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

key	value	default	说明
命令	string	""	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`。

示例 wsl.conf 文件

下面的 `wsl.conf` 示例文件演示了一些可用的配置选项。 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配置设置。
- 用于在作为 WSL 2 版本运行的所有已安装的 Linux 发行版中全局配置设置。
- **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 `%UserProfile%` 目录，请在 PowerShell 中使用 `cd ~` 访问主目录（通常是用户配置文件 `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 文件为使用 WSL 2 运行的所有 Linux 发行版全局配置设置。 *（有关针对发行版配置的信息，请参阅 wsl.conf）*。

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

> ⓘ **备注**

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。

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

key	value	default	说明
内核 (kernel)	path	Microsoft 内置内核提供的收件箱	自定义 Linux 内核的绝对 Windows 路径。
内存	大小	Windows 上总内存的 50% 或 8GB，以较小者为准；在 20175 之前的版本上：Windows 上总内存的 80%	要分配给 WSL 2 VM 的内存量。
处理器	number	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	boolean	`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	string	空白	其他内核命令行参数。
safeMode	boolean	`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

key	value	default	说明
			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	path	`%USERPROFILE%\AppData\Local\Temp\swap.vhdx`	交换虚拟硬盘的绝对 Windows 路径。
pageReporting	boolean	`true`	默认的 `true` 设置使 Windows 能够回收分配给 WSL 2 虚拟机的未使用内存。
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。 仅适用于 Windows 11。
debugConsole	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。 仅适用于 Windows 11。
nestedVirtualization	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。 仅适用于 Windows 11。

key	value	default	说明
vmIdleTimeout	数量*	`60000`	VM 在关闭之前处于空闲状态的毫秒数。 仅适用于 Windows 11。
networkingMode**	string	NAT	如果值为 `mirrored`，则会启用镜像网络模式。 默认或无法识别的字符串会生成 NAT 网络。
firewall**	bool	是	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**	bool	false	更改将 DNS 请求从 WSL 代理到 Windows 的方式
autoProxy*	bool	false	强制 WSL 使用 Windows 的 HTTP 代理信息

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

具有 `size` 值的条目后面必须跟上大小的单位，例如 `8GB` 或 `512MB`。

值类型后带有 * 的条目仅在 Windows 11 中可用。

值类型后显示 ** 的条目需要 Windows 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

设置名称	值	默认值	说明
`autoMemoryReclaim`	string	disabled	检测空闲 CPU 使用率后，自动释放缓存的内存。 设置为 `gradual` 以慢速释放，设置为 `dropcache` 以立即释放缓存的内存。
`sparseVhd`	bool	false	如果设置为 true，则任何新创建的 VHD 将自动设置为稀疏。
`useWindowsDnsCache`**	bool	false	仅当 `wsl2.dnsTunneling` 设置为 true 时才适用。 如果此选项设置为 false，则从 Linux 隧道传输的 DNS 请求将绕过 Windows 中的缓存名称，以始终将请求放在网络上。
`bestEffortDnsParsing`**	bool	false	仅当 `wsl2.dnsTunneling` 设置为 true 时才适用。 如果设置为 true，Windows 将从 DNS 请求中提取问题并尝试解决该问题，从而忽略未知记录。
`initialAutoProxyTimeout`*	string	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析，则必须重启 WSL 实例才能使用检索到的代理设置。
`ignoredPorts`**	string	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。 应在逗号分隔列表中设置格式，例如：`3000,9000,9090`
`hostAddressLoopback`**	bool	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，或允许主机通过此方式连接到容器。 请注意，始终可以使用 127.0.0.1 环回地址 - 此选项也允许使用所有额外分配的本地 IP 地址。

值类型后带有 * 的条目仅在 Windows 11 中可用。

值类型后显示 ** 的条目需要 Windows 版本 22H2 或更高版本。

示例 .wslconfig 文件

下面的 `.wslconfig` 示例文件演示了一些可用的配置选项。 在此示例中，文件路径为 `C:\Users\<UserName>\.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

# 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
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

# 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

在 GitHub 上与我们协作

可以在 GitHub 上找到此内容的源，还可以在其中创建和查看问题和拉取请求。 有关详细信息，请参阅参与者指南。

Windows Subsystem for Linux 反馈

Windows Subsystem for Linux 是一个开放源代码项目。 选择一个链接以提供反馈：

提出文档问题

提供产品反馈

WSL 的文件权限

项目 • 2023/03/21

本页详细说明了 Linux 文件权限在适用于 Linux 的 Windows 子系统上是如何解释的，尤其是在访问 NT 文件系统上 Windows 内部的资源时。 本文档假定读者基本了解 Linux 文件系统权限结构 和 umask 命令 。

从 WSL 访问 Windows 文件时，文件权限是根据 Windows 权限计算的，或者是从已由 WSL 添加到文件的元数据中读取的。 默认情况下，此元数据处于未启用状态。

Windows 文件上的 WSL 元数据

如果在 WSL 中以装载选项的形式启用了元数据，则可以在 Windows NT 文件上添加扩展属性并对其进行解释，从而提供 Linux 文件系统权限。

WSL 可以添加四个 NTFS 扩展属性：

属性名称	说明
$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数字和类型，例如：0777）
$LXDEV	设备（如果它是设备文件）

此外，任何并非常规文件或目录的文件（例如：符号链接、FIFO、块设备、unix 套接字和字符设备）也有 NTFS 重新分析点。 这样一来，就能够以快得多的速度确定给定目录中的文件类型，而不必查询其扩展属性。

文件访问方案

下面介绍了在使用适用于 Linux 的 Windows 子系统以不同的方式访问文件时，权限是如何确定的。

从 Linux 访问 Windows 驱动器文件系统 (DrvFS) 中的文件

在从 WSL 访问 Windows 文件（最有可能是通过 `/mnt/c`）时，会出现这些情况。

从现有 Windows 文件读取文件权限

结果取决于文件是否已具有现有元数据。

DrvFS 文件没有元数据（默认）

如果文件没有关联的元数据，则我们会将 Windows 用户的有效权限转换为读取/写入/执行位，并将其设置为对用户、组和其他用户而言相同的值。 例如，如果你的 Windows 用户帐户具有对该文件的读取和执行访问权限，但不具有对该文件的写入访问权限，则该帐户将对用户、组和其他对象显示为 `r-x`。 如果该文件在 Windows 中设有“只读”属性，则我们不会在 Linux 中授予写入权限。

文件有元数据

如果文件的元数据存在，则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具有现有元数据。

chmod 文件没有元数据（默认）

Chmod 只有一种效果，如果删除文件的所有写入属性，则会设置 Windows 文件上的“只读”属性，因为这是与 CIFS（通用 Internet 文件系统）相同的行为。CIFS 是 Linux 中的 SMB（服务器消息块）客户端。

chmod 文件有元数据

Chmod 将根据文件已有的现有元数据更改或添加元数据。

请记住，你为自己授予的访问权限不能多于你在 Windows 上具有的访问权限，即使元数据显示的情况与此相反，也是如此。 例如，你可以使用 `chmod 777` 将元数据设置为显示你对文件具有写入权限，但如果你尝试访问该文件，则你仍无法写入到该文件。 这是由于互操作性而导致的，因为对 Windows 文件的任何读取或写入命令均通过 Windows 用户权限进行路由。

在 DriveFS 中创建文件

结果取决于是否启用了元数据。

元数据未启用（默认）

新创建的文件的 Windows 权限将与你在未使用特定安全描述符的情况下在 Windows 中创建的文件相同，它将继承父级的权限。

元数据已启用

文件的权限位设置为遵循 Linux umask，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具有现有元数据。

用户文件没有元数据（默认）

在默认情况下，当自动装载 Windows 驱动器时，我们指定任何文件的用户 ID (UID) 将设置为 WSL 用户的用户 ID，并且组 ID (GID) 将设置为 WSL 用户的主体组 ID。

用户文件有元数据

在元数据中指定的 UID 和 GID 将应用为该文件的用户所有者和组所有者。

使用 `\\wsl$` 从 Windows 访问 Linux 文件

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

创建新文件

从 Windows 向 WSL 分发版内部创建新文件时，将应用默认 umask。 默认的 umask 是 `022`，也就是说，它允许除了组和其他用户的写入权限以外的所有权限。

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 驱动器内配置文件权限。 装载选项允许你设置 `umask`、`dmask` 和 `fmask` 权限掩码。 `umask` 应用于所有文件，`dmask` 只应用于目录，而 `fmask` 只应用于文件。 这些权限掩码在应用到文件时会通过一个“逻辑或”操作进行计

算，例如：如果 `umask` 值为 `023` 并且 `fmask` 值为 `022`，则为文件生成的权限掩码将为 `023`。

了解详细信息：wsl.conf 的每个发行版配置选项。

使用 WSL 访问网络应用程序

项目 • 2023/12/05

使用网络应用和 WSL 时需要了解一些注意事项。 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以获得最新的功能和改进。

默认网络模式： NAT

默认情况下，WSL 使用基于 NAT（网络地址转换）的网络体系结构。 使用基于 NAT 的网络体系结构时，请牢记以下注意事项：

从 Windows (localhost) 访问 Linux 网络应用

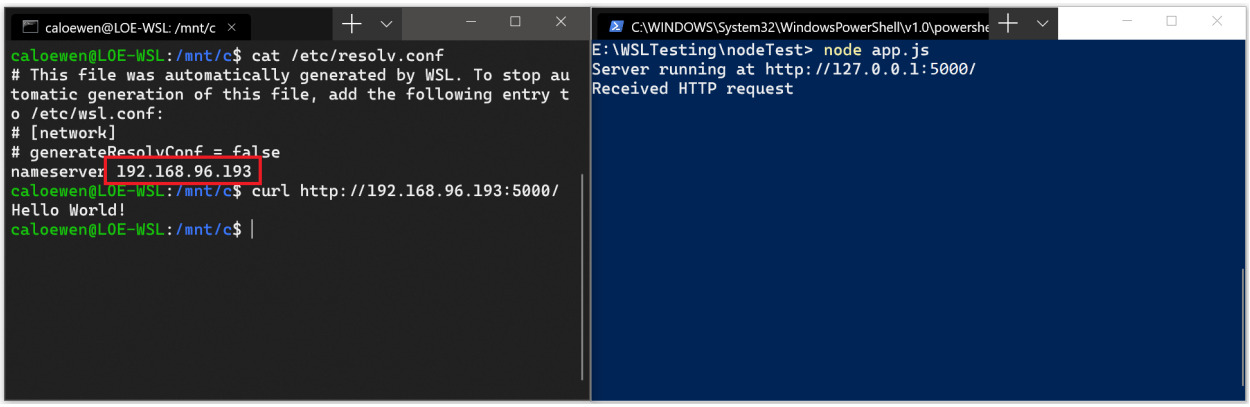
如果要在 Linux 分发版中构建网络应用（例如，在 NodeJS 或 SQL server 上运行的应用），可以使用 `localhost` 从 Windows 应用（如 Microsoft Edge 或 Chrome Internet 浏览器）访问它（就像往常一样）。

从 Linux（主机 IP）访问 Windows 网络应用

如果要从 Linux 分发版（即 Ubuntu）访问 Windows 上运行的网络应用（例如，在 NodeJS 或 SQL 服务器上运行的应用），则需要使用主机的 IP 地址。 虽然这不是一种常见方案，但你可以执行以下步骤来使其可行。

1. 通过在 Linux 发行版中运行以下命令来获取主机的 IP 地址：` 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。

下图显示了一个示例，该示例说明如何通过 curl 连接到在 Windows 中运行的 Node.js 服务器。



通过远程 IP 地址进行连接

当使用远程 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。 这意味着你需要确保你的应用程序可以接受 LAN 连接。

例如，你可能需要将应用程序绑定到 `0.0.0.0` 而非 `127.0.0.1`。 以使用 Flask 的 Python 应用为例，可以通过以下命令执行此操作：`app.run(host='0.0.0.0')`。 进行这些更改时请注意安全性，因为这将允许来自你的 LAN 的连接。

从局域网 (LAN) 访问 WSL 2 分发版

当使用 WSL 1 分发版时，如果计算机设置为可供 LAN 访问，那么在 WSL 中运行的应用程序也可供在 LAN 中访问。

这不是 WSL 2 中的默认情况。 WSL 2 有一个带有其自己独一无二的 IP 地址的虚拟化以太网适配器。 目前，若要启用此工作流，你需要执行与常规虚拟机相同的步骤。 （我们正在寻找改善此体验的方法。）

下面是使用 Netsh 接口 portproxy Windows 命令添加端口代理的示例，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` 表示将接受来自任何 IP 地址的传入请求。 侦听地址指定要侦听的 IPv4 地址，可以更改为以下值：IP 地址、计算机 NetBIOS 名称或计算机 DNS 名称。 如果未指定地址，则默认值为本地计算机。 需要将 `<yourPortToConnectToInWSL>` 值更新为希望 WSL 连接的端口号，例如 `connectport=4000`。 最后，`connectaddress` 值必须是通过 WSL 2 安装的 Linux 分发版的 IP 地址（WSL 2 VM 地址），可通过输入命令：`wsl.exe hostname -I` 找到。

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
```

要获取 IP 地址，请使用：

- `wsl hostname -I` 标识通过 WSL 2 安装的 Linux 分发版 IP 地址（WSL 2 VM 地址）

- `cat /etc/resolv.conf` 表示从 WSL 2 看到的 WINDOWS 计算机的 IP 地址 (WSL 2 VM)

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

> **备注**
>
> 在主机名命令中使用小写“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 是占位符诊断地址），而 `wsl hostname -I` 会返回其他计算机所看到的本地计算机的 IP 地址，应该用于识别通过 WSL 2 运行的 Linux 发行版的 `connectaddress`。

IPv6 访问

- `wsl hostname -i` 标识通过 WSL 2 安装的 Linux 分发版 IP 地址（WSL 2 VM 地址）
- `ip route show | grep -i default | awk '{ print $3}'` 表示从 WSL 2 看到的 WINDOWS 计算机的 IP 地址 (WSL 2 VM)

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

镜像模式网络

可以在 .wslconfig 文件中的 [wsl2] 下设置 networkingMode=mirrored，以启用镜像模式网络。 启用此功能会将 WSL 更改为全新的网络体系结构，其目标是将 Windows 上的网络接口“镜像”到 Linux 中，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
- 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

> **备注**
>
> 使用管理员权限在 PowerShell 窗口中运行以下命令，以**配置 Hyper-V 防火墙**设置，从而允许入站连接： `Set-NetFirewallHyperVVMSetting -Name '{40E0AC32-46A5-438A-A0B2-2B479E8F2E90}' -DefaultInboundAction Allow` 或 `New-NetFirewallHyperVRule -`

```
Name MyWebServer -DisplayName “My Web Server” -Direction Inbound -VMCreatorId “{40E0AC32-46A5-438A-A0B2-2B479E8F2E90}” -Protocol TCP -LocalPorts 80
```
。

这种新模式解决了使用基于 NAT（网络地址转换）的体系结构时出现的网络问题。 查找已知问题或针对 GitHub 上的 WSL 产品存储库 中发现的任何 bug 提交反馈。

DNS 隧道

在 .wslconfig 文件 中的 [wsl2] 下设置 dnsTunneling=true 可使 WSL 使用虚拟化功能来应答来自 WSL 内的 DNS 请求，而不是通过网络数据包请求它们。 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 .wslconfig 文件中的 [wsl2] 下设置 autoProxy=true 会强制 WSL 使用 Windows 的 HTTP 代理信息。 如果已在 Windows 中设置了代理，启用此功能会使该代理也在 WSL 中自动进行设置。

WSL 和防火墙

在运行 Windows 11 22H2 及更高版本以及运行 WSL 2.0.9 及更高版本的计算机上，Hyper-V 防火墙功能将默认打开。 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，详细了解如何在本地应用这些规则和设置，以及如何通过 Intune 之类的在线工具这样做。

使用 systemd 通过 WSL 管理 Linux 服务

项目 • 2023/10/07

适用于 Linux 的 Windows 子系统 (WSL) 现在支持 systemd，它是许多常用的 Linux 发行版（例如 Ubuntu、Debian 等）使用的 init 系统和服务管理器。 （什么是 systemd？）。

init 系统默认值最近已不再是 SystemV，Systemd 现在是使用 wsl --install 命令默认值安装的 Ubuntu 的当前版本默认值。 除当前版本的 Ubuntu 以外的 Linux 发行版仍然可能使用 WSL init，它类似于 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 主要是一个 init 系统和服务管理器，它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 控制组进行跟踪等功能。

大多数主要的 Linux 发行版现在都运行 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 wsl --install 命令默认值安装的 Ubuntu 的当前版本默认值。

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 init）：

1. 确保 WSL 版本为 0.67.6 或更高版本。 （若要检查，请运行 `wsl --version`。若要更新，请运行 `wsl --update` 或从 Microsoft Store 下载最新版本。）

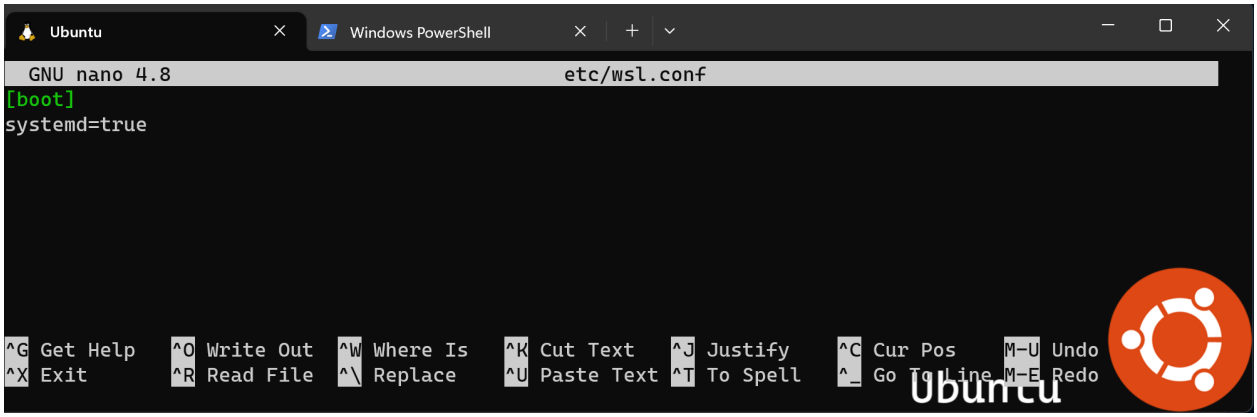
2. 打开 Linux 发行版的命令行并输入 `cd /` 来访问根目录，然后输入 `ls` 来列出文件。你将看到一个名为“etc”的目录，其中包含发行版的 WSL 配置文件。 打开此文件，以便可通过输入 `nano /etc/wsl.conf`，使用 Nano 文本编辑器进行更新。

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 init：

   ```
   Bash
   ```

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

4. 退出 Nano 文本编辑器（Ctrl + X，选择 Y 来保存更改）。 然后，需要关闭 Linux 发行版。 可以使用 PowerShell 中的 `wsl.exe --shutdown` 命令重启所有 WSL 实例。



Linux 发行版重启后，systemd 将会运行。 可使用 `systemctl list-unit-files --type=service` 命令进行确认，这将显示与 Linux 发行版关联的任何服务的状态。

详细了解 WSL 中的高级设置配置，其中包括 teh `wsl.conf`（特定于发行版）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 systemd 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 基于 Ubuntu 博客上的这些演示的教程 - 包括“在 WSL 上使用 snap 只需几分钟即可创建 Nextcloud 实例”、“使用 LXD 管理 Web 项目”和 Run a .Net Echo Bot as a systemd service on Ubuntu WSL （在 Ubuntu WSL 上将 .Net Echo Bot 作为 systemd 服务运行）
- GitHub 上的 Craig microk8s 演示

Systemd 示例

下面是一些依赖于 systemd 的 Linux 应用程序示例：

- snap：Canonical 为使用 Linux 内核和 systemd init 系统的操作系统开发的软件打包和部署系统。 这些包称为“snap”，用于生成 snap 的命令行工具称为“Snapcraft”，可下载/安装 snap 的中心存储库称为“Snap Store”，运行 snap 所需的守护程序（从应用商店下载、装载到位、进行限制并从中运行应用）称为“snapd”。 整个系统有时称为“snappy”。尝试运行游命令 `snap install spotify` 或 `snap install postman`。

- microk8s：一种开源、低运营、最低生产 Kubernetes，可自动部署、缩放和管理容器化应用。 按照说明在 WSL2 上安装 MicroK8s，查看入门教程或观看关于 Windows 上的 Kubernetes 与 MicroK8s 和 WSL 2 的视频。

- systemctl：用于控制和检查 systemd 的命令行实用工具，可帮助你与 Linux 发行版上的服务进行交互。 请尝试 `systemctl list-units --type=service` 命令来查看哪些服务可用及其状态。

一些相关教程演示了使用 systemd 的方法：

- Understanding and Using Systemd （了解和使用 Systemd）

- Systemd Essentials: Working with the Services, Units, and the Journal （Systemd 基础知识：使用服务、单元和日记）

- How To Sandbox Processes With Systemd On Ubuntu 20.04 （如何在 Ubuntu 20.04 上使用 Systemd 设置进程沙盒）

启用 systemd 如何影响 WSL 体系结构?

启用对 systemd 的支持需要更改 WSL 体系结构。 systemd 需要 PID 1，因此在 Linux 发行版中启动的 WSL init 进程将成为 systemd 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。 必须进行其他修改，以确保干净关闭（因为关闭现在由 systemd 控制），并与 WSLg、运行 Linux 图形用户界面 (GUI) 的 WSL 组件，或者 Windows 而不是命令行中显示的 Linux 应用兼容。

此外，请务必注意，通过这些更改，systemd 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。

企业环境：为公司设置适用于 Linux 的 Windows 子系统

项目 • 2023/11/30

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 将 WSL 作为 Windows 组件进行管理和管理推荐设置的指南，请参阅 WSL 的 Intune 设置。

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场景配置 .wslconfig 文件中 [wsl2] 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，使用适用于企业的 Microsoft Store 下载并安装适合你的 Linux 分发版。 使用适用于企业的 Microsoft Store 创建一个帐户。

导出 WSL 映像

导出自定义的 WSL 映像，方法是运行 wsl --export `<Distro> <FileName>`，将映像打包到 tar 文件中，并准备好将其分发到其他计算机。 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

从共享或存储设备分发 WSL 映像，方法是运行 wsl --import `<Distro> <InstallLocation> <FileName>`，将指定的 tar 文件作为新的分发版导入。

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

导入要与 WSL 一起使用的任何 Linux 发行版

项目 • 2023/10/07

通过使用 tar 文件导入任何 Linux 发行版，可在适用于 Linux 的 Windows 子系统 (WSL) 中使用该发行版（即使它不在 Microsoft Store 中提供）。

本文演示了如何通过使用 Docker 容器获取 Linux 发行版 CentOS 的 tar 文件来将它导入，以便与 WSL 一起使用。 此过程可应用于导入任何 Linux 发行版。

获取发行版的 tar 文件

首先，需要获取一个 tar 文件，其中包含发行版的所有 Linux 二进制文件。

可通过多种方式获取 tar 文件，其中两种方式包括：

- 下载提供的 tar 文件。 可在 Alpine Linux 下载站点的“微型根文件系统”部分找到 Alpine 的示例。
- 查找 Linux 发行版容器，将实例导出为 tar 文件。 以下示例将使用 CentOS 容器演示此过程。

获取 CentOS 的 tar 文件示例

在本示例中，使用 WSL 发行版中的 Docker 来获取 CentOS 的 tar 文件。

先决条件

- 必须在安装了运行 WSL 2 的 Linux 发行版的情况下启用 WSL。
- 必须安装了适用于 Windows 的 Docker Desktop，启用了 WSL 2 引擎并选中了集成，请参阅 Docker Desktop 许可协议，了解使用条款的更新。

从容器导出 tar

1. 打开已从 Microsoft Store 安装的 Linux 发行版（本例中是 Ubuntu）的命令行 (Bash)。

2. 启动 Docker 服务：

```
Bash
```

```
sudo service docker start
```

3. 在 Docker 中运行 CentOS 容器：

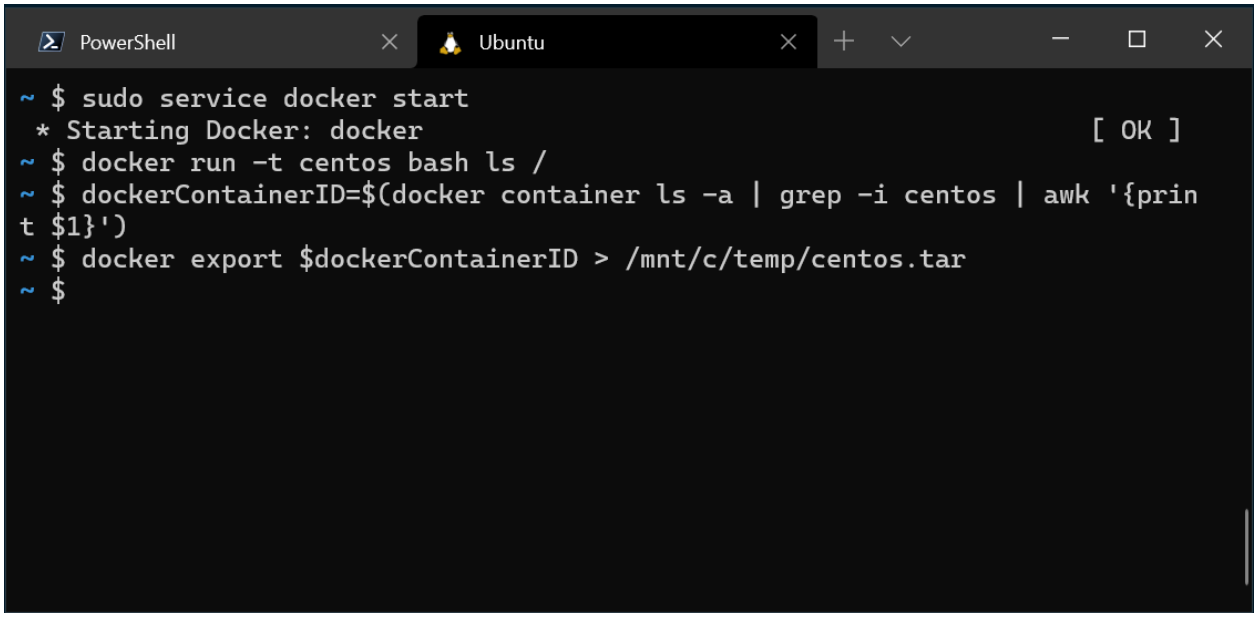
```Bash
docker run -t centos bash ls /
```

4. 使用 grep 和 awk 获取 CentOS 容器 ID：

```Bash
dockerContainerID=$(docker container ls -a | grep -i centos | awk
'{print $1}')
```

5. 将容器 ID 导出到装载的 C 盘上的 tar 文件：

```Bash
docker export $dockerContainerID > /mnt/c/temp/centos.tar
```



此过程从 Docker 容器导出 CentOS tar 文件，这样我们现在就可以导入它，以在本地将其与 WSL 一起使用。

将 tar 文件导入 WSL

准备好 tar 文件后，可使用以下命令导入它：`wsl --import <Distro> <InstallLocation> <FileName>`。

导入 CentOS 示例

将 CentOS 发行版 tar 文件导入 WSL：

1. 打开 PowerShell，并确保已创建一个要存储发行版的文件夹。

   PowerShell

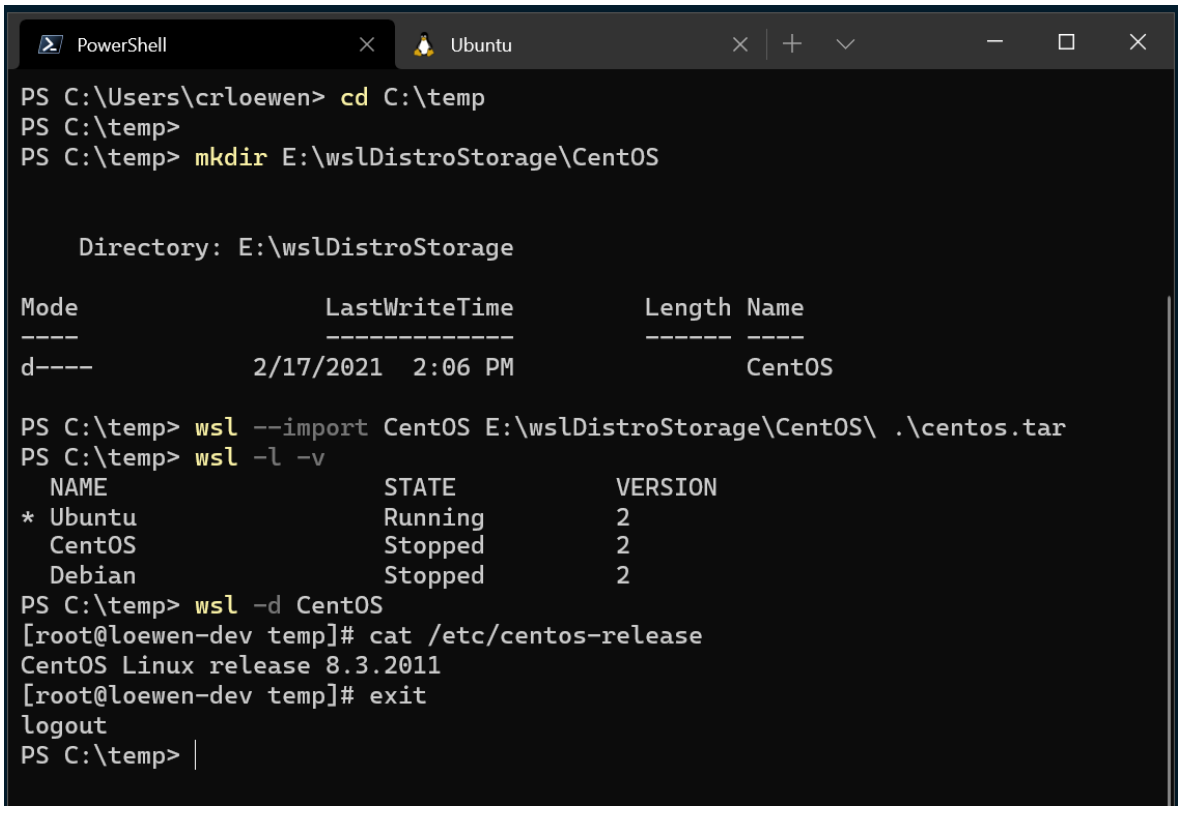
   ```
   cd C:\temp
   mkdir E:\wslDistroStorage\CentOS
   ```

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

   PowerShell

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

3. 使用命令 `wsl -l -v` 检查已安装的发行版。

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

添加特定于 WSL 的组件，例如默认用户

默认情况下，使用 --import 时，你总是作为根用户启动。 可设置自己的用户帐户，但请注意，根据每个不同的 Linux 发行版，设置过程略有不同。

若要使用刚导入的 CentOS 发行版设置用户帐户，首先打开 PowerShell 并使用以下命令引导到 CentOS：

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，用户名为“caloewen”。

> **备注**
>
> 你需要将用户名添加到 sudoers 文件，以便允许用户使用 sudo。 命令 `adduser -G wheel $myUsername` 会将用户 `myUsername` 添加到 wheel 组中。 wheel 组中的用户会被自动授予 sudo 权限，并且可以执行需要提升权限的任务。

Bash

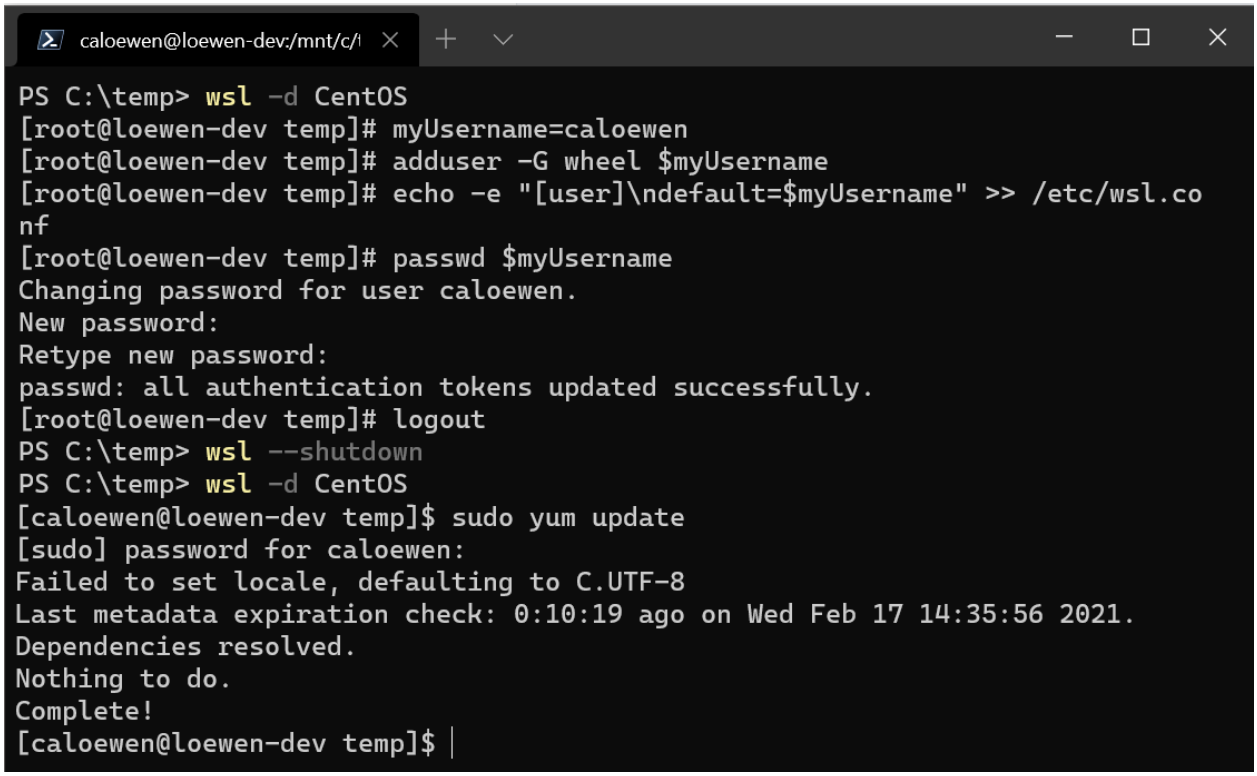
```
yum update -y && yum install passwd sudo -y
myUsername=caloewen
adduser -G wheel $myUsername
echo -e "[user]\ndefault=$myUsername" >> /etc/wsl.conf
passwd $myUsername
```

现在必须退出该实例，并确保所有 WSL 实例都已终止。 再次启动发行版，在 PowerShell 中运行以下命令以查看新的默认用户：

PowerShell

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

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



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 发行版

可创建自己的自定义 Linux 发行版（打包为 UWP 应用），其行为将与 Microsoft Store 中提供的 WSL 发行版完全一样。 若要了解如何操作，请参阅创建适用于 WSL 的自定义 Linux 发行版。

创建适用于 WSL 的自定义 Linux 发行版

项目 • 2023/07/07

使用开源 WSL 示例生成适用于 Microsoft Store 的 WSL 发行版包和/或创建用于旁加载的自定义 Linux 发行版包。 可在 GitHub 上查找发行版启动器存储库。

此项目可实现以下目的：

- 使 Linux 发行版维护人员能够打包并提交一个在 WSL 上运行的作为 appx 的 Linux 发行版
- 使开发人员能够创建可在其开发计算机上旁加载的自定义 Linux 发行版

背景

我们通过 Microsoft Store 将适用于 WSL 的 Linux 发行版分发为 UWP 应用程序。 可安装那些将在 WSL（位于 Windows 内核中的子系统）上运行的应用程序。 如之前的一篇博客文章中所述，这种交付机制有许多优势。

旁加载自定义 Linux 发行版包

可在个人计算机上将自定义 Linux 发行版包创建为应用程序进行旁加载。 请注意，除非你作为发行版维护人员提交自定义包，否则不会通过 Microsoft Store 分发自定义包。 若要将计算机设置为旁加载应用，需要在“面向开发人员”下的系统设置中启用此项。 请确保已选择“开发人员模式”或“旁加载应用”

面向 Linux 发行版维护人员

若要提交到 Store，你需要与我们合作以获得发布批准。 如果你是有兴趣将发行版添加到 Microsoft Store 的 Linux 发行版所有者，请联系 wslpartners@microsoft.com。

入门

按照发行版启动器 GitHub 存储库中的说明，创建自定义 Linux 发行版包。

团队博客

- 开源一个面向 Linux 发行版维护人员的 WSL 示例并旁加载自定义 Linux 发行版
- 命令行博客

提供反馈

- 发行版启动器 GitHub 存储库
- GitHub WSL 问题跟踪器

在 WSL 2 中装载 Linux 磁盘

项目 • 2023/10/07

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驱动器且使用这些装载说明操作失败，则可能需要尝试连接 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）。

> **备注**
>
> 将磁盘附加到 WSL 2 需要管理员访问权限。 WSL 2 `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

你需要使用 Windows 11 内部版本 22000 或更高版本，或者运行 WSL 的 Microsoft Store 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，方法是创建装载的目录（`sudo mkdir /mnt/d`，将“d”替换为要使用的驱动器号），然后使用 `drfs` 文件系统互操作插件并使用以下命令：

```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

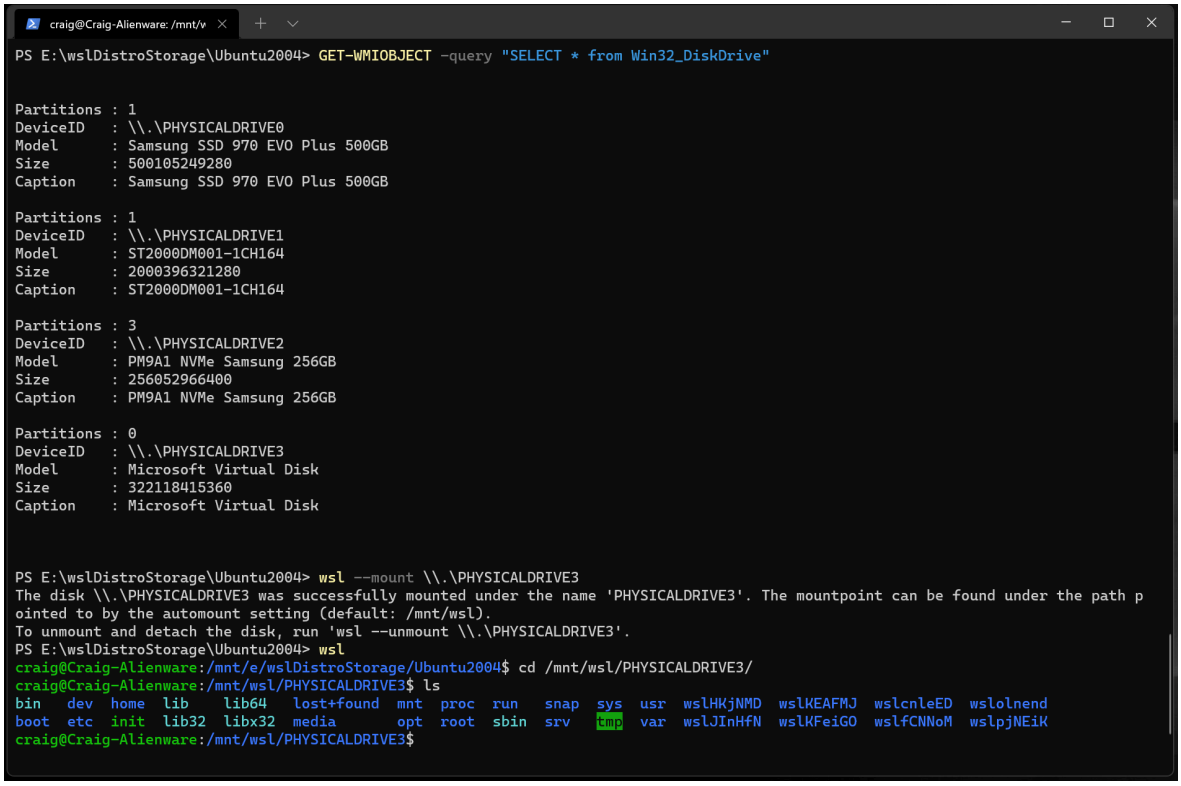
   ```
   GET-CimInstance -query "SELECT * from Win32_DiskDrive"
   ```

   磁盘路径位于“DeviceID”列下。 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 --mount <DiskPath>
   ```



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"
   ```

   磁盘路径列在“DeviceID”之后，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 --mount <DiskPath> --bare
   ```

   这将使磁盘在 WSL 2 中可用。 （在我们的示例中，`<DiskPath>` 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 --mount <DiskPath> --partition <PartitionNumber> --type <Filesystem>
```

> ⓘ **备注**
>
> 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 --unmount <DiskPath>
```

在 WSL 中装载 VHD

> ⓘ **备注**

Microsoft Store 中的 WSL 引入了直接装载 VHD 的新参数：`wsl --mount --vhd <pathToVHD>`

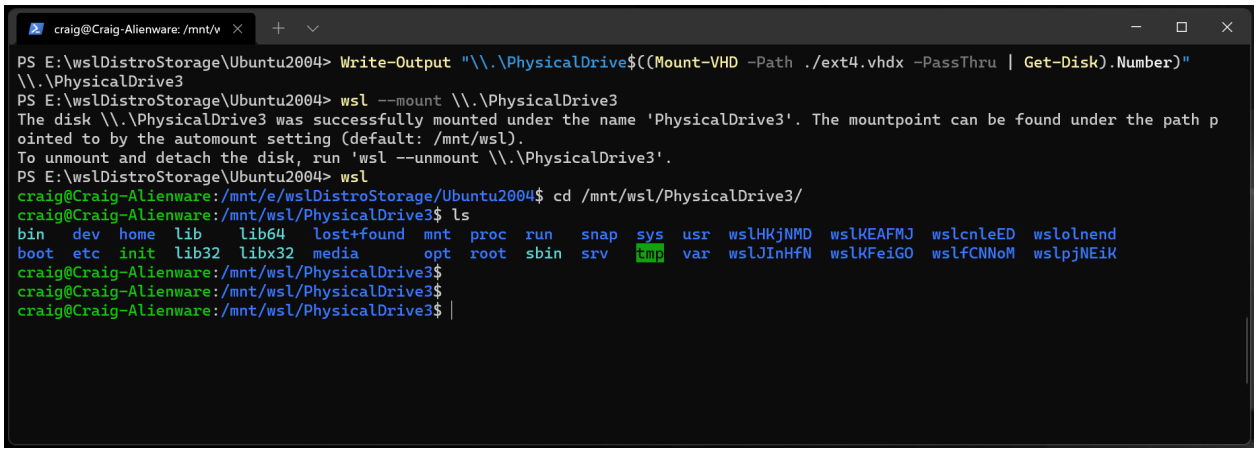
还可以使用 `wsl --mount` 将虚拟硬盘文件 (VHD) 装载到 WSL。 为此，首先需要使用 Windows 中的 Mount-VHD 命令将 VHD 装载到 Windows 中。 请确保以管理员权限运行此命令。 下面是一个示例，我们使用此命令并输出磁盘路径。 请务必将 `<pathToVHD>` 替换为实际 VHD 路径。

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：`C:\Users\[user]\AppData\Local\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 --mount <DiskPath> -t <FileSystem>
```

例如，若要以 fat 形式装载磁盘，请运行：

```
wsl --mount <Diskpath> -t vfat
```

> ⓘ **备注**
>
> 若要列出 WSL2 中的可用文件系统，请运行：`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：

```
wsl --mount <Diskpath> -p <PartitionIndex>
```

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 --mount <DiskPath> -o <MountOptions>
```

示例：

PowerShell

```
wsl --mount <DiskPath> -o "data=ordered"
```

> **备注**
>
> 目前仅支持特定于文件系统的选项。 不支持诸如 `ro, rw, noatime, ...` 之类的通用选项。

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 --mount <DiskPath> --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

> **备注**
>
> 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。示例：

PowerShell

```
wsl --mount <DiskPath> --name myDisk
```

分离磁盘

若要从 WSL 2 分离磁盘，请运行：

PowerShell

```
wsl --unmount [DiskPath]
```

如果省略 `Diskpath`，则将卸载和分离所有附加的磁盘。

> ⓘ **备注**
>
> 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。
- `wsl --mount` 只能装载内核中原生支持的文件系统。 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。

连接 USB 设备

项目 • 2024/02/04

本指南将演练使用 USB/IP 开源项目 usbipd-win 将 USB 设备连接到在 WSL 2 上运行的 Linux 发行版所必要的步骤。

在 Windows 计算机上设置 USB/IP 项目将启用常见的开发人员 USB 场景，例如刷写 Arduino 或访问智能卡读取器。

先决条件

- 运行 Windows 11（内部版本 22000 或更高版本）。 （可提供 Windows 10 支持，请参见下面的注释）
- 需要具有 x64 处理器的计算机。 （x86 和 Arm64 目前不支持 usbipd win）。
- WSL 已安装并使用最新版本进行设置。
- Linux 发行版已安装并设置为 WSL 2。

> **备注**
>
> 若要检查 Windows 版本及内部版本号，选择 Windows 徽标键 + R，然后键入“winver”，选择“确定”。 可通过选择“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 发行版并输入命令：`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 Store 同时支持 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 **Microsoft Store 中的 WSL 现已在 Windows 10 和 11 上正式发布**，了解如何更新到 Store 支持的 WSL 版本。 如果无法更新到 Store 支持的 WSL 版本并自动接收内核更新，请参阅 **USBIPD-WIN 项目存储库**，了解如何通过生成自己的已启用 USBIP 的 WSL 2 内核，将 USB 设备连接到在 WSL 2 上运行的 Linux 分发版。

安装 USBIPD-WIN 项目

WSL 本身并不支持连接 USB 设备，因此你需要安装开源 usbipd-win 项目。

1. 转到 usbipd-win 项目的最新发布页。
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. 运行下载的 usbipd-win_x.msi 安装程序文件。

> **备注**
>
> 或者，也可使用 Windows **程序包管理器程序** (winget) 来安装 usbipd-win 项目。 如果已安装 winget，只需使用命令 `winget install --interactive --exact dorssel.usbipd-win` 安装 usbipd-win 即可。 如果你省略了 --interactive，winget 可能会立即重启计算机（如果这是安装驱动程序所必需的操作）。

这将安装：

- 名为 `usbipd` 的服务（显示名称：USBIP 设备主机）。 可使用 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。

附加 USB 设备

在附加 USB 设备之前，请确保 WSL 命令行已打开。 这将使 WSL 2 轻型 VM 保持活动状态。

> **备注**
>
> 此文档假定已安装 **usbipd-win 4.0.0 或更高版本**

1. 通过以*管理员*模式打开 PowerShell 并输入以下命令，列出所有连接到 Windows 的 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` 验证设备是否已共享。

   ```
   ```

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。 附加到 WSL 后，任何作为 WSL 2 运行的分发版本都可以使用 USB 设备。 使用 `usbipd list` 验证设备是否已附加。 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（或首选的 WSL 命令行），使用以下命令列出附加的 USB 设备：

   Bash

   ```
   lsusb
   ```

   你应会看到刚刚附加的设备，并且能够使用常规 Linux 工具与之交互。 根据你的应用程序，你可能需要配置 udev 规则以允许非根用户访问设备。

5. 在 WSL 中完成设备使用后，可物理断开 USB 设备，或者从 PowerShell 运行此命令：

   PowerShell

   ```
   usbipd detach --busid <busid>
   ```

若要详细了解此操作的工作原理，请参阅 Windows 命令行博客和 GitHub 上的 usbipd-win 存储库。

有关视频演示，请参阅 WSL 2：连接 USB 设备（制表符与空格显示）。

在 GitHub 上与我们协作

可以在 GitHub 上找到此内容的源，还可以在其中创建和查看问题和拉取请求。 有关详细信息，请参阅参与者指南。

Windows Subsystem for Linux 反馈

Windows Subsystem for Linux 是一个开放源代码项目。 选择一个链接以提供反馈：

提出文档问题

提供产品反馈

调整区分大小写

项目 • 2023/03/21

区分大小写确定在文件名或目录中是将大写 (FOO.txt) 和小写 (foo.txt) 字母作为不同项（区分大小写）还是等效项（不区分大小写）进行处理。

- 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 区分大小写之间的差异

使用 Linux 和 Windows 文件和目录时，可能需要调整区分大小写的处理方式。

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 将被视为不同文件。

Windows 文件系统支持使用属性标志按目录设置区分大小写。 虽然标准行为是不区分大小写，但你可以分配属性标志来使目录区分大小写，以便它能够识别可能仅大小写不同的 Linux 文件和文件夹。

在将驱动器装载到适用于 Linux 的 Windows 子系统 (WSL) 文件系统时，尤其如此。 在 WSL 文件系统中工作时，运行的是 Linux，因此默认情况下，文件和目录被视为区分大小写。

> **备注**
>
> 在过去，如果文件的名称仅大小写不同，则 Windows 将无法访问这些文件，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大小写不同的文件。 虽然 Windows 文件资源管理器将同时显示这两个文件，但无论你选择哪一个文件，都只会打开一个文件。

更改文件和目录的区分大小写

以下步骤说明如何更改 Windows 文件系统上的目录，使其区分大小写，并能识别仅大小写不同的文件和文件夹。

> ⚠ **警告**
>
> 某些 Windows 应用程序（假定文件系统不区分大小写）不使用正确的大小写来引用文件。 例如，应用程序将文件名转换为全部使用大写或小写的情况并不罕见。 在标记为区分大小写的目录中，这意味着这些应用程序将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的文件的目录树中创建新目录，则这些目录不区分大小写。 这可能会导致难以在区分大小写的目录中使用 Windows 工具，因此，在更改 Windows 文件系统区分大小写设置时要小心谨慎。

检查当前区分大小写

若要检查 Windows 文件系统中的目录是否区分大小写，请运行以下命令：

PowerShell

```
fsutil.exe file queryCaseSensitiveInfo <path>
```

将 `<path>` 替换为文件路径。 对于 Windows (NTFS) 文件系统中的目录，`<path>` 将如下所示：`C:\Users\user1\case-test`，如果你已位于 `user1` 目录中，则可以直接运行：`fsutil.exe file setCaseSensitiveInfo case-test`

修改区分大小写

自 Windows 10 内部版本 17107 开始，支持按目录区分大小写。 在 Windows 10 内部版本 17692 中，支持已更新，以包括从 WSL 内检查和修改目录的区分大小写标志。 使用名为 `system.wsl_case_sensitive` 的扩展属性公开区分大小写。 对于不区分大小写的目录，此属性的值为 0；对于区分大小写的目录，此属性的值为 1。

更改目录的区分大小写需要运行提升的权限（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。 有关这方面的更多信息，请参阅疑难解答部分。

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默认设置不区分大小写 (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> disable
```

目录必须为空，才能更改该目录的区分大小写标志属性。 对于包含其名称仅大小写不同的文件夹/文件的目录，不能禁用区分大小写标志。

区分大小写的继承性

创建新目录时，这些目录将从其父目录继承区分大小写。

> ⚠ **警告**
>
> 在 WSL 1 模式下运行时，此继承策略会出现异常。 当发行版在 WSL 1 模式下运行时，不会继承每个目录的区分大小写标志；在区分大小写的目录中创建的目录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

使用 WSL 配置文件在适用于 Linux 的 Windows 子系统上装载驱动器时，可以管理区分大小写。 安装的每个 Linux 发行版都可以有自己的 WSL 配置文件，名为 `/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 (ie 的 Linux 分发版。Ubuntu) 。
2. 更改目录，直至看到 `etc` 文件夹（这可能需要从 `home` 目录 `cd ..`）。
3. 列出 `etc` 目录中的文件，查看 `wsl.conf` 文件是否已存在（使用 `ls` 命令，或使用 `explorer.exe .` 通过 Windows 文件资源管理器查看目录）。
4. 如果 `wsl.conf` 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` 创建它，后者将在从 Nano 编辑器保存时创建该文件。
5. 可以将以下选项添加到 `wsl.conf` 文件中：

默认设置： `dir` 用于为每个目录启用区分大小写。

Bash

```
[automount]
options = case = dir
```

(装载的 NTFS 驱动器上所有目录的区分大小写不可用，) 不区分大小写： `off`

```
Bash

[automount]
options = case = off
```

将 (NTFS) 驱动器上的所有目录视为区分大小写： `force`

```
Bash

[automount]
options = case = force
```

此选项仅在作为 WSL 1 运行的 Linux 发行版上安装驱动器时受支持，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）命令提示符中使用此命令：`reg.exe add HKLM\SYSTEM\CurrentControlSet\Services\lxss /v DrvFsAllowForceCaseSensitivity /t REG_DWORD /d 1`。

对 `wsl.conf` 文件进行任何更改之后，需要重新启动 WSL 才能使这些更改生效。 可以使用以下命令重新启动 WSL：`wsl --shutdown`

> **提示**
>
> 若要为所有驱动器装载具有特定区分大小写设置的驱动器（使用 DrvFs 文件系统插件使磁盘在 /mnt 下可用，如 /mnt/c、/mnt/d 等），请如上所述使用 `/etc/wsl.conf`。 若要为某个特定驱动器设置默认装载选项，请使用 **/etc/fstab 文件** 指定这些选项。 有关更多 WSL 配置选项，请参阅**使用 wslconf 配置每个发行版的启动设置**。

更改装载到 WSL 发行版的驱动器的区分大小写

默认情况下，装载到 WSL 发行版的 NTFS 格式的驱动器将不区分大小写。 更改装载到 WSL 分发 (的驱动器上的目录的区分大小写。Ubuntu)，请遵循与上面列出的 Windows 文件系统相同的步骤。 （默认情况下，EXT4 驱动器区分大小写）。

若要在目录上启用区分大小写 (FOO ≠ foo)，请使用以下命令：

```
Bash

fsutil.exe file setCaseSensitiveInfo <path> enable
```

若要在目录上禁用区分大小写并返回到默认设置不区分大小写 (FOO = foo)，请使用以下命令：

```
Bash

fsutil.exe file setCaseSensitiveInfo <path> disable
```

> **备注**
>
> 如果在 WSL 运行期间更改已装载驱动器的现有目录上的区分大小写标志，请确保 WSL 没有对该目录的引用，否则更改将无效。 这意味着任何 WSL 进程都不能打开该目录，包括使用该目录（或其子目录）作为当前工作目录。

使用 Git 配置区分大小写

Git 版本控制系统还具有一个配置设置，可用于调整处理的文件的区分大小写。 如果使用 Git，则可能需要调整 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、虚假冲突或文件重复。

有关详细信息，请参阅 Git 配置文档。

故障排除

我的目录包含大小写相混合的文件，需要区分大小写，但 Windows FS 工具无法识别这些文件

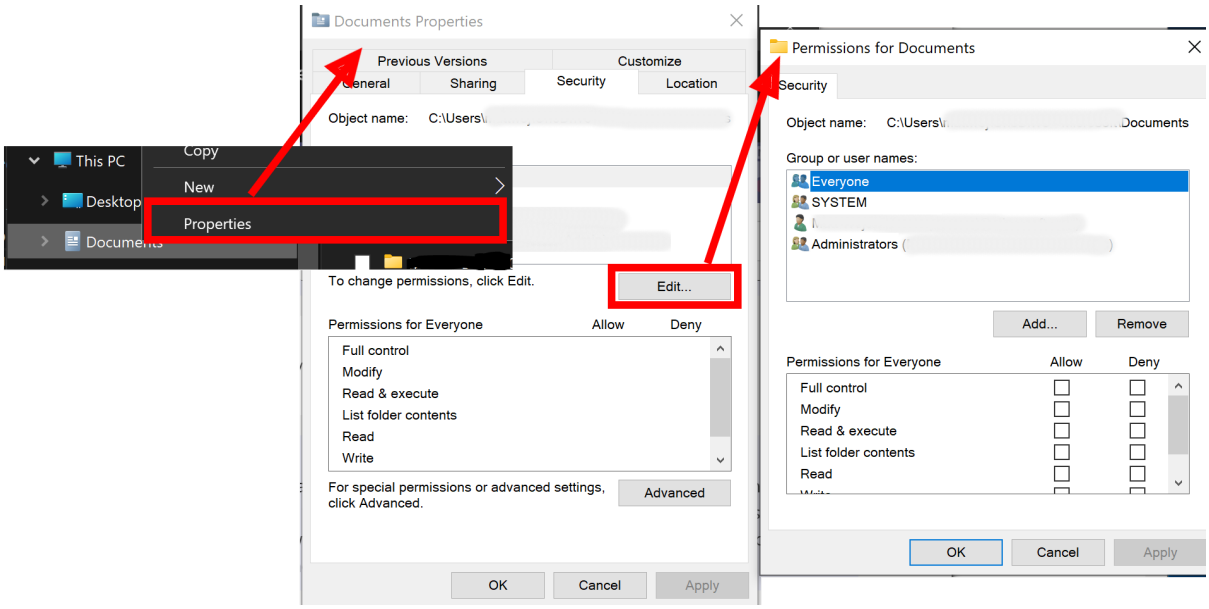
若要使用 Windows 文件系统工具处理包含混合大小写文件的 Linux 目录，需要创建一个全新的目录并将其设置为区分大小写，然后将这些文件复制到该目录中（使用 git clone 或 untar）。 这些文件仍使用的是混合大小写。 （请注意，如果已尝试将文件移动到不区分大小写的目录，并且存在冲突，则很可能某些文件已被覆盖且不再可用。）

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。 尝试创建一个新目录，更改设置，然后将混合大小写文件复制到该目录中。

错误：访问被拒绝

确保在需要更改区分大小写的目录上具有“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该目录（从命令行，使用命令：`explorer.exe .`）。 右键单击该目录，并选择“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

只能在 NTFS 格式的文件系统中的目录上设置区分大小写属性。 默认情况下，WSL (Linux) 文件系统中的目录区分大小写（不能使用 fsutil.exe 工具设置为不区分大小写）。

其他资源

- DevBlog：按目录区分大小写和 WSL
- DevBlog：WSL 中改进的按目录区分大小写支持

如何管理 WSL 磁盘空间

项目 • 2023/11/10

本指南介绍如何管理通过 WSL 2 安装的 Linux 发行版使用的磁盘空间，包括：

- 如何检查 VHD 中的可用磁盘空间量
- 如何扩展 VHD 的大小
- 发生错误时如何修复 VHD
- 如何查找任何已安装的 Linux 发行版的 .vhdx 文件和磁盘路径

适用于 Linux 的 Windows 子系统 (WSL 2) 使用虚拟化平台将 Linux 发行版与主机 Windows 操作系统一起安装，创建虚拟硬盘 (VHD) 来存储你安装的每个 Linux 发行版的文件。 这些 VHD 使用 ext4 文件系统类型，并在你的 Windows 硬盘驱动器上表示为 *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 文件最初被分配了 1TB 的最大磁盘空间量（在 WSL 版本 0.58.0 之前，此默认值设置为最大 512GB，在那之前为最大 256GB）。

如果 Linux 文件所需的存储空间超过此最大大小，你就会看到指示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操作。

如何检查可用磁盘空间

使用 Linux `Df` 命令检查随 WSL 2 安装的 Linux 发行版的 VHD 中可用的磁盘空间量。

若要检查可用磁盘空间，请打开 PowerShell 命令行并输入此命令（将 `<distribution-name>` 替换为实际的发行版名称）：

PowerShell

```
wsl.exe --system -d <distribution-name> df -h /mnt/wslg/distro
```

如果此命令对你没有效果，请使用 `wsl --update` 命令升级到 WSL 的 Microsoft Store 版本，或尝试 `wsl df -h /`。

该输出将包括：

- **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**：VHD 中剩余的空间量（已分配的大小减去使用的量）

- **使用百分比**：剩余磁盘空间的百分比（已用/已分配的大小）
- **装载位置**：装载磁盘的目录路径

如果发现即将达到分配给 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，请参阅下一节，了解如何扩展分配给与 Linux 发行版关联的 VHD 的最大磁盘空间量。 WSL 分配给 VHD 的磁盘空间量将始终显示默认的最大数量（在最新版本的 WSL 中为 1TB），即使实际 Windows 设备上的磁盘空间量小于此值。 WSL 会装载一个 VHD，该 VHD 将在你使用它的过程中扩展大小，因此 Linux 发行版会发现它可以增长到分配的大小上限 1TB。

如何扩展 WSL 2 虚拟硬盘的大小

若要将 Linux 发行版的 VHD 大小扩展到超过**默认的 1TB 上限**（所分配的磁盘空间量），请执行以下步骤。 *（对于尚未更新的早期 WSL 版本，此最大默认值可能设置为 512GB 或 256GB）。*

1. 使用 `wsl.exe --shutdown` 命令终止所有 WSL 实例

2. 将目录路径复制到与计算机上安装的 Linux 发行版关联的 *ext4.vhdx* 文件。 如需帮助，请参阅如何查找 Linux 发行版的 vhdx 文件和磁盘路径。

3. 使用管理员权限打开 Windows 命令提示符，然后输入以下命令来打开 diskpart 命令解释器：

   Windows 命令提示符

   ```
   diskpart
   ```

4. 现在会出现一个 `DISKPART>` 提示。 输入以下命令，将 `<pathToVHD>` 替换为与 Linux 发行版关联的 `ext4.vhdx` 文件的目录路径（步骤 2 中复制的）。

   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**，表示当前分配给 VHD 的大小上限：

   Windows 命令提示符

   ```
   detail vdisk
   ```

6. 需要将**虚拟大小**转换为 MB。 例如，如果**虚拟大小为 512 GB**，则等于 **512000 MB**。 你输入的新值必须大于此原始值。 要将 512 GB 的虚拟大小加倍到 1024 GB，需以 MB 为单位输入值：**1024000**。 请注意，不要输入高于实际需要的值，因为减小虚拟磁盘大小的过程要复杂得多。

7. 使用 Windows 命令提示符 `DISKPART>` 提示输入要分配给此 Linux 发行版的新的大小上限值：

   Windows 命令提示符

   ```
   expand vdisk maximum=<sizeInMegaBytes>
   ```

8. 退出 `DISKPART>` 提示：

   Windows 命令提示符

   ```
   exit
   ```

9. 启动此 Linux 发行版。 *（确保它在 WSL 2 中运行。可以使用以下命令确认这一点：`wsl.exe -l -v`。不支持 WSL 1）。*

10. 通过从 WSL 发行版命令行运行这些命令，让 WSL 知道它可以扩展此发行版的文件系统大小。 可能会看到以下消息，它响应第一个 mount 命令：“/dev: /dev 上未装载任何内容。”可以放心地忽略此消息。

    Bash

    ```
    sudo mount -t devtmpfs none /dev
    mount | grep ext4
    ```

11. 复制此项的名称，该名称类似于：`/dev/sdX`（X 表示任何其他字符）。 在下面的示例中，X 的值是 b：

Bash

```
   sudo resize2fs /dev/sdb <sizeInMegabytes>M
```

在上述示例中，我们将 vhd 大小更改为 **2048000**，因此命令将为：`sudo resize2fs /dev/sdb 2048000M`。

ⓘ 备注

可能需要安装 resize2fs。 如果是这样，可以使用此命令进行安装：`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 发行版的虚拟驱动器 (ext4.vhdx) 现已成功扩展到新的大小。

> **ⓘ 重要**
>
> 建议不要使用 Windows 工具或编辑器来修改、移动或访问 `AppData` 文件夹中与 WSL 相关的文件。 这样做可能会导致 Linux 分发版损坏。 如果要从 Windows 访问 Linux 文件，可通过路径 `\\wsl$\<distribution-name>\` 进行访问。 打开 WSL 分发版，然后输入 `explorer.exe .` 来查看此文件夹。 若要了解详细信息，请查看博客文章：**从 Windows 访问 Linux 文件**。

如何修复 VHD 装载错误

如果你遇到了与“装载发行版磁盘”相关的错误，这可能是由于突然关闭或断电，它可能导致 Linux 发行版 VHD 切换到*只读*，以避免数据丢失。 可以按照以下步骤使用 `e2fsck` Linux 命令来修复和还原发行版。

使用 lsblk 命令标识块设备名称

当 WSL 2 安装 Linux 发行版时，它会使用自己的文件系统将发行版装载为虚拟硬盘 (VHD)。 Linux 称这些硬盘驱动器为“块设备”，你可以使用 `lsblk` 命令查看有关它们的信息。

若要查找 WSL 2 当前正在使用的块设备的名称，请打开发行版并输入此命令：`lsblk`。（或打开 PowerShell 并输入命令：`wsl.exe lsblk`。）输出类似于以下：

```
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：

- **NAME**：分配给设备的名称将为 sd[a-z]，指的是 SCSI 磁盘，每个使用的磁盘都有指定的字母。 `sda` 始终是系统发行版。
- **MAJ:MIN**：表示 Linux 内核用于在内部标识设备的数字，其中第一个数字表示设备类型（8 用于表示小型计算机系统接口/SCSI 磁盘）。
- **RM**：让我们知道设备可以移动 (1) 与否 (0)。
- **SIZE**：卷的总大小。
- **RO**：让我们知道设备是只读 (1) 与否 (0)。
- **TYPE**：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 WSL 在打开 Linux 发行版时遇到了“装载错误”，则发行版可能会设置为只读，作为备用。 如果发生这种情况，则发行版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.
```

当发行版以只读方式启动时，写入文件系统的任何尝试都将失败，并会出现如下错误：

Bash

```
$ touch file
touch: cannot touch 'file': Read-only file system
```

若要修复 WSL 中的磁盘装载错误，并再次将其还原到可用/可写状态，可以使用 `wsl.exe --mount` 命令通过以下步骤重新装载磁盘：

1. 打开 PowerShell 并输入此命令来关闭所有 WSL 发行版：

   PowerShell

   ```
   wsl.exe --shutdown
   ```

2. （在提升的命令提示符中）以管理员身份打开 PowerShell，然后输入装载命令，将 `<path-to-ext4.vhdx>` 替换为发行版的 .vhdx 文件的路径。 有关查找此文件的帮助，请参阅如何查找 Linux 发行版的 VHD 文件和磁盘路径。

   PowerShell

   ```
   wsl.exe --mount <path-to-ext4.vhdx> --vhd --bare
   ```

3. 通过 PowerShell 使用 `wsl.exe lsblk` 命令来标识发行版的块设备名称 (sd[a-z])，然后输入以下命令修复磁盘（将 `<device>` 替换为正确的块设备名称，如“sdc”）。 `e2fsck` 命令会检查 ext4 文件系统（随 WSL 安装的发行版所使用的类型）的错误并相应地修复它们。

   PowerShell

   ```
   wsl.exe sudo e2fsck -f /dev/<device>
   ```

> ⓘ **备注**
>
> 如果只安装了一个 Linux 分发版，可能会遇到“ext 文件正在使用中”错误，并需要再**安装**一个分发版才能运行 `wsl.exe lsblk`。 修复完成后，可以**卸载**该分发版。

4. 修复完成后，输入以下命令在 PowerShell 中卸载磁盘：

   PowerShell

   ```
   wsl.exe --unmount
   ```

> ⚠ **警告**
>
> 可以使用 `sudo mount -o remount,rw /` 命令将只读发行版返回到可用/可写状态，但所有更改都将在内存中，因此会在重启发行版时丢失。 建议改用上面列出的步骤来装载和修复磁盘。

如何查找 Linux 发行版的 .vhdx 文件和磁盘路径

若要查找 Linux 发行版的 *.vhdx* 文件和目录路径，请打开 PowerShell 并使用以下脚本，将 `<distribution-name>` 替换为实际的发行版名称：

PowerShell

```
(Get-ChildItem -Path HKCU:\Software\Microsoft\Windows\CurrentVersion\Lxss |
Where-Object { $_.GetValue("DistributionName") -eq '<distribution-name>'
}).GetValue("BasePath") + "\ext4.vhdx"
```

结果将显示类似于 `%LOCALAPPDATA%\Packages\<PackageFamilyName>\LocalState\<disk>.vhdx` 的路径。 例如：

PowerShell

```
C:\Users\User\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_7
9rhkp1fndgsc\LocalState\ext4.vhdx
```

这是与你列出的 Linux 发行版关联的 `ext4.vhdx` 文件的路径。

有关适用于 Linux 的 Windows 子系统的常见问题解答

常见问题解答

常规

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 操作系统的一项功能，通过它可以直接在 Windows 上运行 Linux 文件系统以及 Linux 命令行工具和 GUI 应用，并可以运行传统的 Windows 桌面和应用。

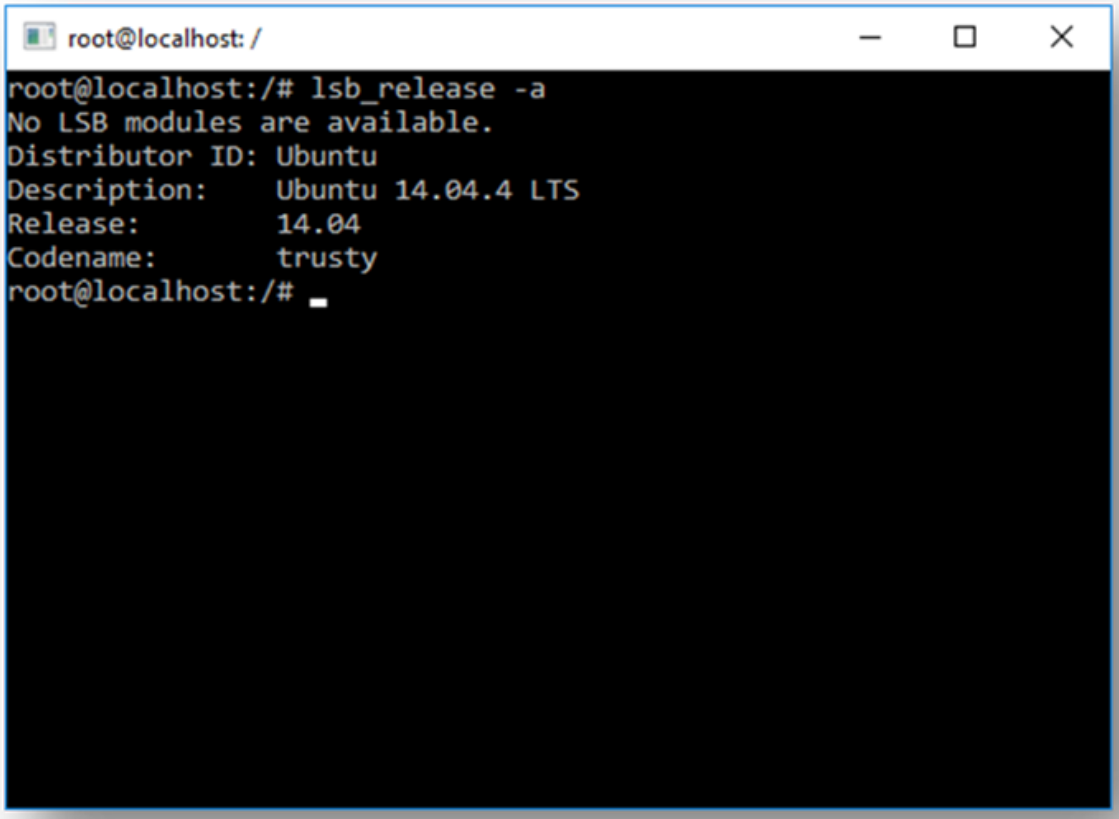
请参阅“关于”页了解更多详细信息。

WSL 面向哪些用户?

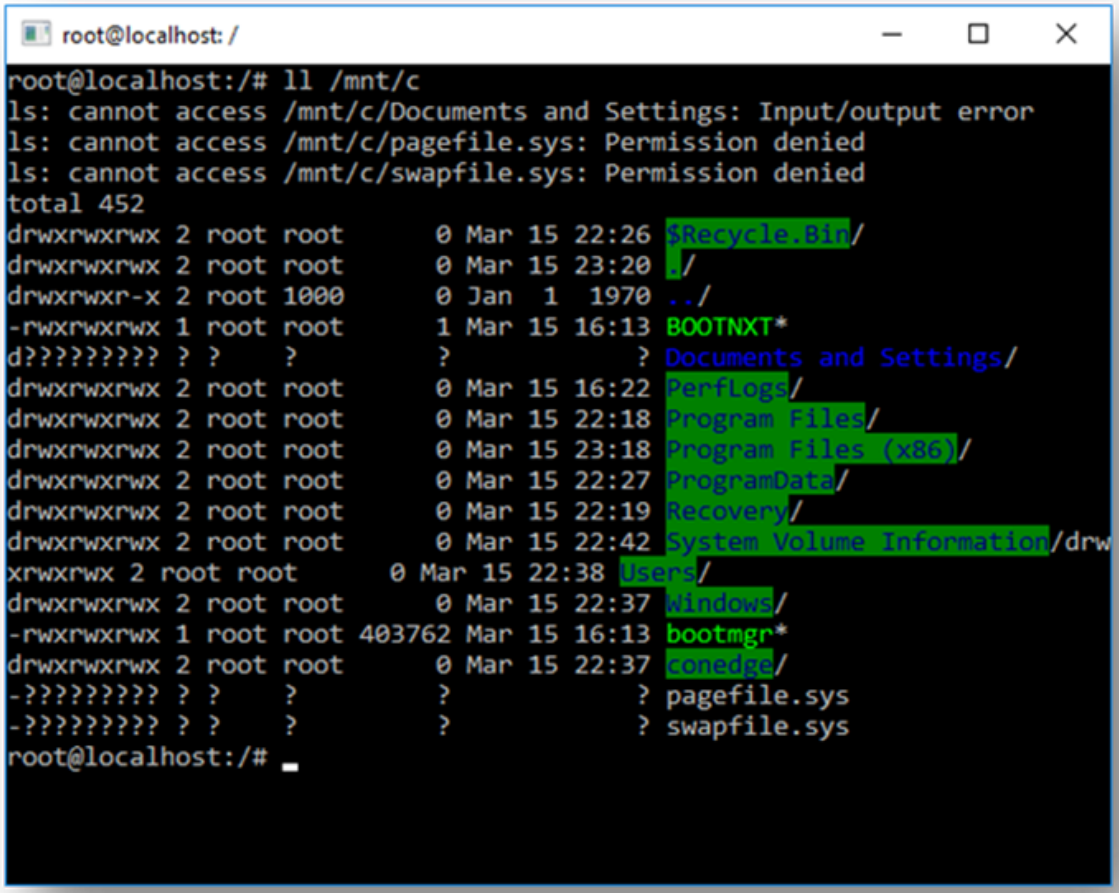
这主要是一种面向开发人员的工具，尤其是 Web 开发人员，他们从事开源项目，或者部署到 Linux 服务器环境。 WSL 适用于喜欢使用 Bash、常用 Linux 工具 (`sed` `awk`、 等) 和 Linux 优先框架 (Ruby、Python 等) 但也喜欢使用 Windows 生产力工具的任何人

WSL 有哪些作用?

通过 WSL，可使用选择的发行版（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 后，将会看到当前正在运行的 Linux 分发版的详细信息：



还可以从 Linux Bash shell 内部访问本地计算机的文件系统 - 你会发现本地驱动器装载在 `/mnt` 文件夹下。 例如，你的 `C:` 驱动器装载在 `/mnt/c` 下：



你能否描述一下整合了 WSL 的典型开发工作流?

WSL 面向开发人员受众，旨在用作内部开发流程的一部分。 假设何石要创建一个 CI/CD 管道（持续集成和持续交付），而且他想先在本地计算机（笔记本电脑）上测试它，然后再将它部署到云中。 何石可启用 WSL（和 WSL 2 来提高速度和性能），然后在本地（在笔记本电脑上）将正版 Linux Ubuntu 实例与所需的任何 Bash 命令和功能搭配使用。 在本地验证开发管道后，何石可将该 CI/CD 管道向上推送到云中（也就是 Azure 中），方法是将它放入 Docker 容器并将该容器推送到云实例，使其在生产就绪的 Ubuntu VM 上运行。

什么是 Bash?

Bash 是一个流行的基于文本的 shell，并且是一种命令语言。 它是 Ubuntu 和其他 Linux 发行版中包含的默认 shell。 用户在 shell 中键入命令，即可执行脚本和/或运行命令与工具来完成许多任务。

WSL 的工作原理是怎样的?

请查看 Windows 命令行博客上的这篇文章：深入探究 WSL 如何允许 Windows 访问 Linux 文件，其中详细介绍了基础技术。

在 VM 中为何要使用 WSL 而不是 Linux?

WSL 所需的资源（CPU、内存和存储）少于完整虚拟机所需的资源。 WSL 还允许结合 Windows 命令行、桌面和 Store 应用运行 Linux 命令行工具与应用，并允许从 Linux 内部访问 Windows 文件。 这样，你便可以根据需要针对相同的文件集使用 Windows 应用和 Linux 命令行工具。

举例而言，我为何要在 Linux（而不是 Windows）上使用 Ruby?

生成某些跨平台工具时，已假设其运行环境的行为类似于 Linux。 例如，某些工具假设它们能够访问很长的文件路径，或者特定的文件/文件夹存在。 这通常会在 Windows 上导致出现问题，因为 Windows 的行为通常与 Linux 不同。

许多语言（例如 Ruby 和 Node.js）通常已移植到 Windows，并且可以在 Windows 上非常顺利地运行。 但是，并非所有 Ruby Gem 或 Node/NPM 库所有者都会移植其库来支持 Windows，而许多库都与 Linux 之间存在特定的依赖关系。 这经常导致使用此类工具和库生成的系统在 Windows 上遭遇到生成错误（有时是运行时错误），或者出现不需要的行为。

这只是导致许多人要求 Microsoft 改进 Windows 命令行工具的一部分问题，也正是这些问题促使我们与 Canonical 展开合作，使得本机 Bash 和 Linux 命令行工具能够在 Windows 上运行。

这对于 PowerShell 而言意味着什么?

处理 OSS 项目时，在很多情况下，从 PowerShell 提示符切换到 Bash 极其有用。 Bash 支持是互补性的，可以增强 Windows 上的命令行的价值，使 PowerShell 和 PowerShell 社区能够利用其他流行技术。

请参阅 PowerShell 团队博客了解详细信息 -- Bash for Windows：Why it's awesome and what it means for PowerShell （为何它如此出色，它对 PowerShell 而言意味着什么）

WSL 支持哪些处理器?

WSL 支持 x64 和 Arm CPU。

如何访问我的 C: 驱动器?

系统会自动为本地计算机上的硬盘驱动器创建装入点，通过这些装入点可以轻松访问 Windows 文件系统。

/mnt/<drive letter>/

示例用法：运行 `cd /mnt/c` 访问 c:\

如何设置 Git 凭据管理器？ （如何在 WSL 中使用我的 Windows Git 权限？）

请参阅开始在适用于 Linux 的 Windows 子系统上使用 Git 这一教程，其中有一个关于设置 Git 凭据管理器和在 Windows 凭据管理器中存储身份验证令牌的部分。

如何在 Linux 应用中使用 Windows 文件?

WSL 的优势之一是可以通过 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 分发版中的 `/mnt/<drive>` 文件夹下。 例如，你的 `C:` 驱动器装载在 `/mnt/c/` 下

例如，使用装载的驱动器，可以使用 Visual Studio 或 VS Code 编辑 `C:\dev\myproj\` 中的代码，并通过 `/mnt/c/dev/myproj` 访问相同的文件，在 Linux 中生成/测试该代码。

有关详细信息，请参阅跨 Windows 和 Linux 文件系统工作一文。

Linux 驱动器中的文件是否不同于装载的 Windows 驱动器中的文件?

1. Linux 根目录（即 `/`）下的文件由符合 Linux 行为的 WSL 控制，控制的内容和操作包括但不限于：
    - 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 chown 更改文件属性
    - 文件/文件夹的区分大小写状态
2. 装载的驱动器中的文件由 Windows 控制，并具有以下行为：
    - 支持区分大小写
    - 设置的所有权限可以最好地反映 Windows 权限

如何卸载 WSL 分发版?

要从 WSL 中删除一个发行版并删除与该 Linux 发行版相关的所有数据，请运行 `wsl --unregister <distroName>`，其中 `<distroName>` 是你的 Linux 发行版的名称，可从 `wsl -l` 命令的列表中查看发行版的名称。

此外，可以像卸载任何其他应用商店应用程序一样，在计算机上卸载 Linux 发行版应用。

若要详细了解 wsl 命令，请参阅 WSL 的基本命令一文。

如何运行 OpenSSH 服务器?

OpenSSH 作为可选功能随 Windows 一起提供。 请参阅安装 OpenSSH 文档。在 WSL 中运行 OpenSSH 需要拥有 Windows 中的管理员特权。 若要运行 OpenSSH 服务器，请以管理员身份运行 WSL 发行版（如 Ubuntu）或 Windows 终端。 有几个资源涵盖了使用 WSL 的 SSH 场景。 请查看 Scott Hanselman 的博客文章：如何从 Linux 或 Windows 或任何位置通过 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，从外部机器通过 SSH 连接到 Windows 10 上的 Bash 和 WSL2 的简单方法，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 安装会尝试自动更改 Ubuntu 区域设置，使之与 Windows 安装的区域设置相匹配。如果你不希望出现此行为，可以在安装完成后，运行此命令来更改 Ubuntu 区域设置。必须重启 WSL 发行版才能使此更改生效。

以下示例将区域设置更改为 en-US：

```Bash
sudo update-locale LANG=en_US.UTF8
```

为何无法从 WSL 进行 Internet 访问?

某些用户已报告特定的防火墙应用程序会阻止 WSL 中的 Internet 访问的问题。 报告的防火墙包括：

1. Kaspersky
2. AVG
3. Avast
4. Symantec Endpoint Protection
5. F-Secure

在某些情况下，关闭防火墙即可进行访问。 在某些情况下，只需让安装的防火墙在表面上阻止访问。

如何从 Windows 中的 WSL 访问某个端口?

WSL 共享 Windows 的 IP 地址，因为它在 Windows 上运行。 因此，你可以访问 localhost 上的任何端口。例如，如果你在端口 1234 上提供 Web 内容，可以在 Windows 浏览器中输入 https://localhost:1234 。 有关详细信息，请参阅访问网络应用程序。

如何备份我的 WSL 发行版，或者如何将它们从一个驱动器移到另一个驱动器?

Windows 版本 1809 和更高版本中提供了备份或移动发行版的最佳方式：执行 export/import 命令。 可以使用 `wsl --export` 命令将整个分发版导出到 tarball。 然后，可以使用 `wsl --import` 命令（该命令可以指定一个新的驱动器位置，以用于导入）将此发行版导入回 WSL，从而可以备份和保存 WSL 发行版的状态或者移动 WSL 发行版。

请注意，用于备份 AppData 文件夹中的文件的传统备份服务（例如 Windows 备份）不会损坏 Linux 文件。

是否可以将 WSL 用于生产方案?

WSL 经过设计和构建，可与内部循环开发工作流配合使用。 WSL 中的一些设计功能使它非常适合用于此目的，但与其他产品相比，它可能会使其在生产相关方案中具有挑战性。我们的目标是明确 WSL 与常规 VM 环境的不同之处，以便你可以决定它是否符合业务需求。

WSL 与传统生产环境之间的main差异如下：

- WSL 具有可自动启动、停止和管理资源的轻型实用工具 VM。
- 如果 Windows 进程没有打开的文件句柄，则 WSL VM 将自动关闭。 这意味着，如果将其用作 Web 服务器，通过 SSH 连接到它以运行服务器，然后退出，VM 可能会关闭，因为它正在检测到用户已完成使用，并将清理其资源。
- 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 分发版等都可供用户访问，并且可以由用户修改。
- WSL 自动授予对 Windows 文件的文件访问权限。
- 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 `/dev/dxg` 设备进行，该设备将 GPU 呼叫路由到 Windows GPU。 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，随着内部循环开发工作流的优先顺序，预计将来会出现更多差异。

如何将 WSL 文件从一台计算机传输到另一台计算机?

有几种方法可以完成此任务：

- 最简单的方法是使用 `wsl --export --vhd` 命令将 WSL 分发导出到 VHD 文件。 然后，可以将此文件复制到另一台计算机，并使用 `wsl --import --vhd`导入它。 有关详细信息，请参阅 命令文档 。
- 上述实现需要大量磁盘空间。 如果磁盘空间不足，可以使用 Linux 技术将文件移到以下区域：
  - 使用 `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并运行 `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包的列表： `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的包，例如 `sudo apt install -y $(cat package_list.txt)` 在传输文件之后。

WSL 2

WSL 2 是否使用 Hyper-V？ 它在 Windows 10 家庭版 和 Windows 11 家庭版 上是否可用?

WSL 2 可用于提供 WSL 的所有桌面 SKU，包括Windows 10 家庭版和Windows 11 家庭版。

最新版本的 WSL 使用 Hyper-V 体系结构来实现其虚拟化。 此体系结构将在“虚拟机平台”可选组件中提供。 此可选组件在所有 SKU 上都将可用。 当我们更深入地了解 WSL 2 版本时，可以看到有关此体验的更多详细信息。

WSL 1 将发生什么情况？ 它是否将被弃用?

我们目前没有计划弃用 WSL 1。 你可以并行运行 WSL 1 和 WSL 2 发行版，还可以随时升级和降级任何发行版。 将 WSL 2 添加为新的体系结构为 WSL 团队提供了一个更好的平台来提供一些特性，使 WSL 成为在 Windows 中运行 Linux 环境的一种令人惊叹的方式。

我是否能够运行 WSL 2 和其他第三方虚拟化工具（例如 VMware 或 VirtualBox）？

当使用 Hyper-V 时，某些第三方应用程序无法工作，这意味着当启用了 WSL 2 时，这些应用程序（如 VMware 和 VirtualBox）将无法运行。 但最近，VirtualBox 和 VMware 都发布了支持 Hyper-V 和 WSL2 的版本。 在此处了解有关 VirtualBox 的更改的详细信息，并在此处了解有关 VMware 的更改的详细信息。 若要解决问题，请查看 GitHub 上 WSL 存储库中的 VirtualBox 问题讨论。

我们一直在开发解决方案以支持 Hyper-V 的第三方集成。 例如，我们向第三方虚拟化提供商公开了一组称为虚拟机监控程序平台的 API，可以用来使其软件与 Hyper-V 兼容。 这使得应用程序可以将 Hyper-V 体系结构用于其模拟，例如，现在都与 Hyper-V 兼容的 Google 安卓模拟器和 VirtualBox 6 及更高版本。

有关 VirtualBox 6.1 的 WSL 2 问题的更多背景和讨论，请参阅 WSL 问题存储库。

*如果要查找 Windows 虚拟机，可在 Windows 开发人员中心下载 VMWare、Hyper-V、VirtualBox 和 Parallels VM。

是否可以在 WSL 2 中访问 GPU？ 是否计划增加硬件支持?

我们发布了相关支持，可在 WSL 2 发行版内访问 GPU！ 这意味着，在涉及到大数据集时，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应用场景。 请查看 GPU 支持入门教程。 从现在开始，WSL 2 不包括串行支持和 USB 设备支持。 我们正在研究添加这些功能的最佳方法。 但是，USB 支持现在通过 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，一般情况下，网络应用程序在 WSL 2 中将工作得更好，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 的行为与虚拟机类似 -- WSL 2 分发版的 IP 地址与 windows OS) (主机地址不同。 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?

可以！ 你需要确保虚拟机已启用嵌套虚拟化。 可以在父 Hyper-V 主机中在 PowerShell 窗口中使用管理员权限运行以下命令来启用此功能：

```
Set-VMProcessor -VMName <VMName> -ExposeVirtualizationExtensions $true
```

请确保将“<VMName>”替换为你的虚拟机的名称。

是否可以在 WSL 2 中使用 wsl.conf?

WSL 2 支持 WSL 1 使用的同一 WSL 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、启用或禁用互操作、更改将装载 Windows 驱动器的目录等）在 WSL 2 中都可以工作。 还可以在发行版管理页面中详细了解 WSL 中的配置选项。 若要详细了解对装载驱动器、磁盘、设备或虚拟硬盘 (VHD) 的支持，请参阅在 WSL 2 中装载 Linux 磁盘一文。

可以在何处提供反馈?

借助 WSL 产品存储库问题，可进行以下操作：

- 搜索现有问题，查看是否存在任何与遇到的问题相关的问题。 请注意，在搜索栏中可以删除“is:open”，以便在搜索中包括已解决的问题。 请考虑对任何希望优先处理的未解决问题发表评论或点赞。
- 提交新问题。 如果发现 WSL 存在问题，并且该问题不像是已有的问题，你可以选择绿色的“新问题”按钮，然后选择“WSL - Bug 报告”。 需要在其中提供以下信息：问题的标题、Windows 内部版本号（运行 `cmd.exe /c ver` 查看当前的内部版本号）、运行的是 WSL 1 还是 2、当前的 Linux 内核版本号（运行 `wsl.exe --status` 或 `cat /proc/version`）、发行版的版本号（运行 `lsb_release -r`）、涉及的任何其他软件版本、重现步骤、预期行为、实际行为和诊断日志（如果可用且适当）。 有关详细信息，请参阅参与 WSL 工作。
- 提交功能请求，方法是选择绿色的“新问题”按钮，然后选择“功能请求”。 需要回答几个描述你的请求的问题。

也可执行以下操作：

- 使用 WSL 文档存储库提交文档问题。 若要参与编写 WSL 文档，请参阅 Microsoft Docs 参与者指南。
- 如果问题更多地与 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，可以访问：

- 我们的命令行团队博客
- Twitter。 请在 Twitter 上关注 @craigaloewen 来了解最新资讯、更新等。

反馈

此页面是否有帮助? 是 否

提供产品反馈

排查适用于 Linux 的 Windows 子系统问题

项目 • 2024/02/22

我们在下面介绍了一些与 WSL 相关的常见故障排除场景，但也请考虑在 GitHub 上的 WSL 产品存储库 中搜索已提交的问题。

提交问题、bug 报告、功能请求

借助 WSL 产品存储库问题，可进行以下操作：

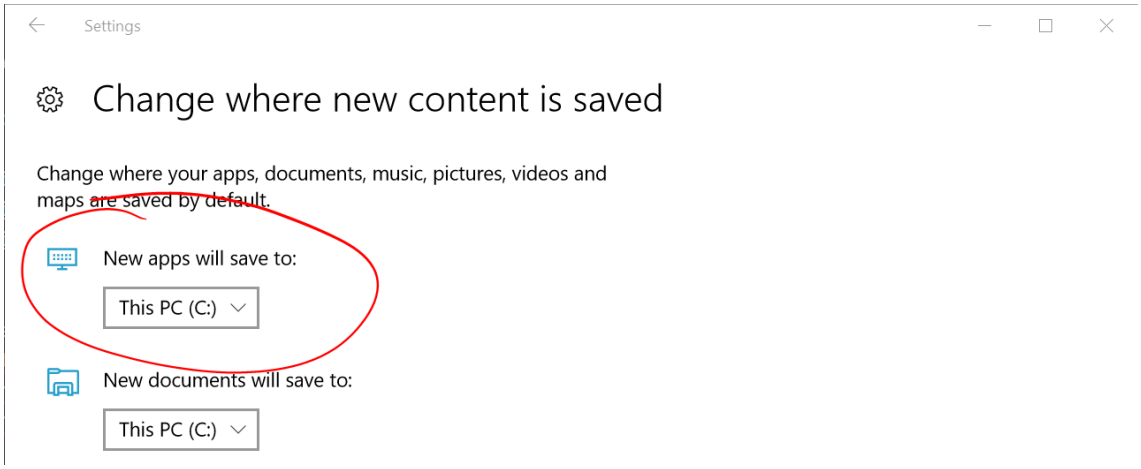
- 搜索现有问题，查看是否存在任何与遇到的问题相关的问题。 请注意，在搜索栏中可以删除“is:open”，以便在搜索中包括已解决的问题。 请考虑对任何希望优先处理的未解决问题发表评论或点赞。
- 提交新问题。 如果发现 WSL 存在问题，并且该问题不像是已有的问题，你可以选择绿色的“新问题”按钮，然后选择“WSL - Bug 报告”。 需要在其中提供以下信息：问题的标题、Windows 内部版本号（运行 `cmd.exe /c ver` 查看当前的内部版本号）、运行的是 WSL 1 还是 2、当前的 Linux 内核版本号（运行 `wsl.exe --status` 或 `cat /proc/version`）、发行版的版本号（运行 `lsb_release -r`）、涉及的任何其他软件版本、重现步骤、预期行为、实际行为和诊断日志（如果可用且适当）。有关详细信息，请参阅参与 WSL 工作。
- 提交功能请求，方法是选择绿色的“新问题”按钮，然后选择“功能请求”。 需要回答几个描述你的请求的问题。

也可执行以下操作：

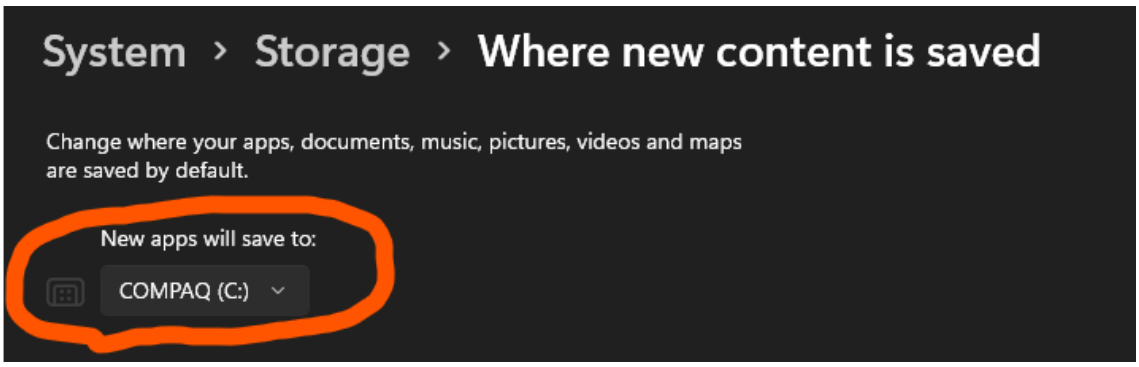
- 使用 WSL 文档存储库 提交文档问题。 若要参与编写 WSL 文档，请参阅 Microsoft Docs 参与者指南。
- 如果问题更多地与 Windows 终端、Windows 控制台或命令行 UI 相关，请使用 Windows 终端产品存储库 来提交 Windows 终端问题。

安装问题

- **安装失败并出现错误** 0x80070003
  - 适用于 Linux 的 Windows 子系统只能在系统驱动器（通常是 `C:` 驱动器）中运行。 请确保分发版存储在系统驱动器上：
  - 在 Windows 10 上打开“设置”->“系统”->“存储”->“更多存储设置: 更改新内容的保存位置”



  - 在 Windows 11 上打开“设置”->“系统”->“存储”->“高级存储设置”->“新内容的保存位置”



- **WslRegisterDistribution 失败并出现错误 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”-> 选中“适用于 Linux 的 Windows 子系统”，或使用本文开头所述的 PowerShell cmdlet。

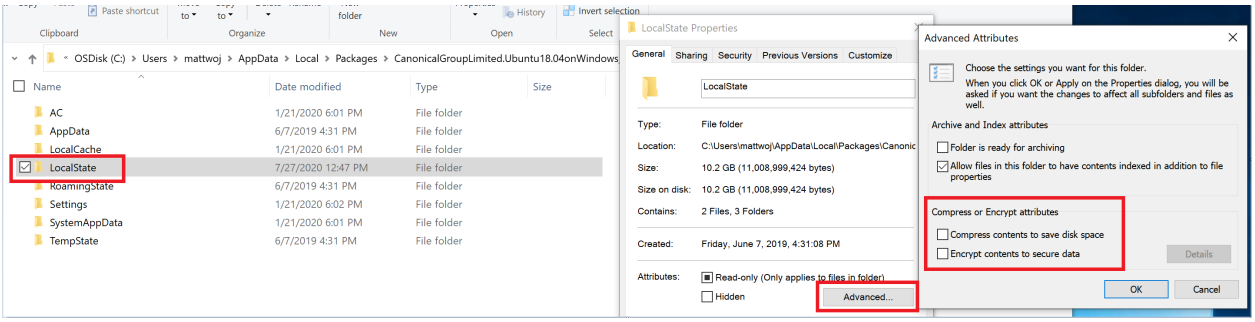
- **安装失败，出现错误 0x80070003 或错误 0x80370102**
  - 请确保在计算机的 BIOS 内已启用虚拟化。 有关如何执行此操作的说明因计算机而异，并且很可能在 CPU 相关选项下。
  - WSL2 要求 CPU 支持二级地址转换 (SLAT) 功能，后者已在 Intel Nehalem 处理器（Intel Core 第一代）和 AMD Opteron 中引入。 即使成功安装了虚拟机平台，旧版 CPU（例如 Intel Core 2 Duo）也无法运行 WSL2。

- **尝试升级时出错：** `Invalid command line option: wsl --set-version Ubuntu 2`
  - 请确保已启用适用于 Linux 的 Windows 子系统，并且你使用的是 Windows 内部版本 18362 或更高版本。 若要启用 WSL，请在 PowerShell 提示符下以具有管理员权限的身份运行此命令：`Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`。

- **由于虚拟磁盘系统的某个限制，无法完成所请求的操作。 虚拟硬盘文件必须是解压缩的且未加密的，并且不能是稀疏的。**
  - 打开 Linux 发行版的配置文件文件夹，取消选中“压缩内容”（如果已选中“加密内容”，请一并取消选中）。 它应位于 Windows 文件系统上的一个文件夹中，类似

于：`%USERPROFILE%\AppData\Local\Packages\CanonicalGroupLimited...`

- 在此 Linux 发行版配置文件中，应存在一个 LocalState 文件夹。 右键单击此文件夹可显示选项的菜单。 选择“属性”>“高级”，然后确保未选中（未勾选）“压缩内容以节省磁盘空间”和“加密内容以保护数据”复选框。 如果系统询问是要将此应用到当前文件夹还是应用到所有子文件夹和文件，请选择“仅此文件夹”，因为你只是要清除压缩标志。 完成此操作后，`wsl --set-version` 命令应正常工作。



> **① 备注**
>
> 在我的示例中，我的 Ubuntu 18.04 发行版的 LocalState 文件夹位于 C:\Users<my-user-name>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhkp1fndgsc
>
> 请检查 WSL Docs GitHub 主题 #4103，其中跟踪了此问题以提供更新的信息。

- **无法将词语“wsl”识别为 cmdlet、函数、脚本文件或可运行程序的名称。**
  - 请确保已安装“适用于 Linux 的 Windows 子系统”可选组件。 此外，如果你使用的是 ARM64 设备，并从 PowerShell 运行此命令，则会收到此错误。 请改为从 PowerShell Core 或从命令提示符运行 `wsl.exe`。

- **错误：适用于 Linux 的 Windows 子系统未安装发行版。**
  - 如果你在已安装 WSL 发行版之后收到此错误：

    1. 请先运行该发行版一次，然后在从命令行中调用它。
    2. 检查你是否正在运行单独的用户帐户。 运行具有提升权限（在管理员模式下）的主要用户帐户应该不会导致出现此错误，但你应确保你未在无意中运行 Windows 附带的内置管理员帐户。 这是单独的用户帐户，根据设计将不显示任何已安装的 WSL 发行版。 有关详细信息，请参阅启用和禁用内置 Administrator 帐户。
    3. WSL 可执行文件仅安装到本机系统目录中。 在 64 位 Windows 上运行 32 位进程（若在 ARM64 上，则为任何非本机组合）时，托管的非本机进程实际上会看到一个不同的 System32 文件夹。 （32 位进程在 x64 Windows 上看到的文件夹存储在磁盘上的 \Windows\SysWOW64。）可通过在虚拟文件夹

`\Windows\sysnative` 中查找，从托管进程访问“本机”system32。 请记住，它实际上不会存在于磁盘上，不过文件系统路径解析程序会找到它。

- **错误：此更新仅适用于装有适用于 Linux 的 Windows 子系统的计算机。**
  - 若要安装 Linux 内核更新 MSI 包，需要 WSL，应先启用它。 如果失败，将看到以下消息：`This update only applies to machines with the Windows Subsystem for Linux`。
  - 出现此消息有三个可能的原因：

    1. 你仍使用旧版 Windows，不支持 WSL 2。 有关版本要求和要更新的链接，请参阅步骤 #2。

    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，需要重新启动才能使其生效，请重新启动计算机，然后重试。

- **错误： WSL 2 要求对其内核组件进行更新。 若需了解相关信息，请访问 https://aka.ms/wsl2kernel 。**
  - 如果 %SystemRoot%\system32\lxss\tools 文件夹中缺少 Linux 内核包，会遇到此错误。 若要解决此问题，请在安装说明的步骤 #4 中安装 Linux 内核更新 MSI 包。 可能会需要从“添加或删除程序”卸载 MSI，然后重新安装。

常见问题

我在使用 Windows 10 版本 1903，但我还是没看见 WSL 2 选项

这可能是因为你的计算机尚未实现对 WSL 2 的向后移植。 要解决此问题，最简单的方法是转到 Windows 设置，然后打击“检查更新”，在你的系统上安装最新更新。 查看有关实现向后移植的完整说明 。

如果你点击了“检查更新”，但仍未收到更新，可以手动安装 KB KB4566116 。

错误：使用 `wsl --set-default-version 2` 时显示 0x1bc

当“显示语言”或“系统区域设置”未设为英语时，可能会发生此情况。

PowerShell

```
wsl --set-default-version 2
Error: 0x1bc
```

```
For information on key differences with WSL 2 please visit
https://aka.ms/wsl2
```

`0x1bc` 的实际错误为：

PowerShell

```
WSL 2 requires an update to its kernel component. For information please
visit https://aka.ms/wsl2kernel
```

有关详细信息，请查看问题 5749

无法从 Windows 访问 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。 如果在 Windows 上无法使用 `\\wsl$` 访问 WSL，则可能是因为 9P 无法正确启动。

要检查这一点，可以使用 `dmesg |grep 9p` 来检查启动日志，这将显示任何错误。 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
```

请参阅此 GitHub 主题获取有关此问题的进一步讨论。

无法启动 WSL 2 发行版，仅在输出中看到“WSL 2”

如果你的显示语言不是英语，则可能会看到截断的错误文本。

PowerShell

```
C:\Users\me>wsl
WSL 2
```

要解决此问题，请访问 `https://aka.ms/wsl2kernel`，按照该文档页面上的指示手动安装内核。

在 Linux 中执行 Windows .exe 时，显示 `command not found`

用户可以直接从 Linux 运行 notepad.exe 等 Windows 可执行文件。 有时，你可能会点击“找不到命令”，如下所示：

```Bash
$ notepad.exe
-bash: notepad.exe: command not found
```

如果 $PATH 中没有 Win32 路径，互操作将找不到 .exe。 可以通过在 Linux 中运行 `echo $PATH` 来进行验证。 应该会在输出中看到 Win32 路径（例如 /mnt/c/Windows）。 如果看不到任何 Windows 路径，则很可能是 Linux shell 覆盖了 PATH。

以下是 Debian 上的 /etc/profile 导致此问题的一个示例：

```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 上的正确方法是删除以上行。 还可在赋值过程中追加 $PATH（如下所示），但这可能导致 WSL 和 VSCode 的一些其他问题。

```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。

Windows 更新后出现“错误:0x80370102 由于未安装所需功能，因此无法启动虚拟机。”

请启用 Virtual Machine Platform Windows 功能，并确保在 BIOS 中启用了虚拟化。

1. 请查看 Hyper-V 系统要求
2. 如果你的计算机是 VM，请手动启用嵌套虚拟化。 以管理员身份启动 Powershell，然后运行以下命令，将 `<VMName>` 替换为主机系统上虚拟机的名称（可在 Hyper-V 管理器中找到）：

```
PowerShell

Set-VMProcessor -VMName <VMName> -ExposeVirtualizationExtensions $true
```

3. 请按照电脑制造商提供的虚拟化启用指南进行操作。 通常，这可能涉及使用系统 BIOS 来确保在 CPU 上启用了这些功能。 此过程的说明可能因计算机而异，有关示例，请参阅来自 Bleeping Computer 的本文。

4. 启用 `Virtual Machine Platform` 可选组件后，请重启计算机。

5. 确保在启动配置中启用了虚拟机监控程序启动。 可以通过运行（提升的 PowerShell）来对此进行验证：

   ```
   PowerShell

   bcdedit /enum | findstr -i hypervisorlaunchtype
   ```

   如果看到 `hypervisorlaunchtype Off`，则会禁用虚拟机监控程序。 使其在提升的 PowerShell 中运行：

   ```

   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），请确保安装其最新版本（VMware 15.5.5+ 和 VirtualBox 6+ 这些版本可以支持 HyperV）或关闭它们。

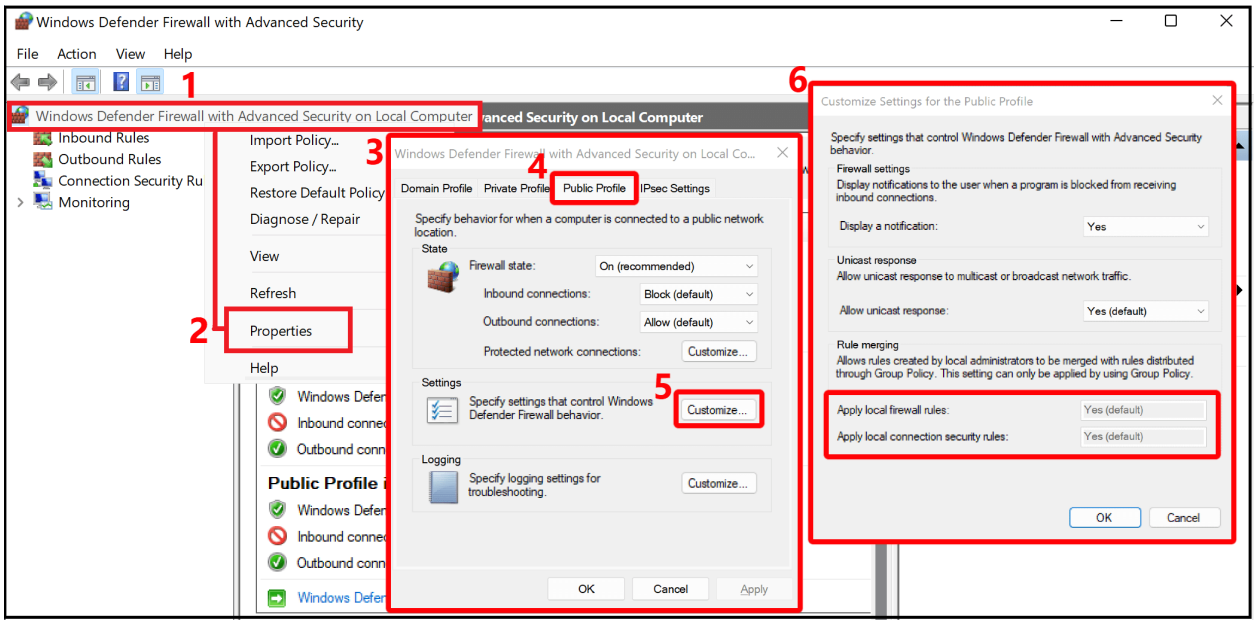
7. 如果在 Azure 虚拟机上收到此错误，检查以确保受信任的启动已禁用。 Azure 虚拟机不支持嵌套虚拟化。

详细了解在虚拟机中运行 Hyper-V 时如何配置嵌套虚拟化。

WSL 在工作计算机上或在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 WSL 网络在默认情况下无效，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的设置：



1. 打开“高级安全 Windows Defender 防火墙”（这不是“控制面板”中的“Windows Defender 防火墙”）
2. 右键单击“本地计算机上的高级安全 Windows Defender 防火墙”选项卡
3. 选择“属性”
4. 在打开的新窗口中选择“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”窗口中查看“规则合并”是否设置为“否”。 这将阻止对 WSL 的访问。

可在配置 Hyper-V 防火墙中找到有关如何更改此防火墙设置的说明。

WSL 在连接到 VPN 后没有网络连接

如果在连接到 Windows 上的 VPN 后 bash 断开网络连接，请尝试从 bash 内部解决问题。 此解决方法允许通过 `/etc/resolv.conf` 手动替代 DNS 解析。

1. 执行 `ipconfig.exe /all`，记下 VPN 的 DNS 服务器
2. 执行 `sudo cp /etc/resolv.conf /etc/resolv.conf.new`，创建现有 resolv.conf 的副本
3. 执行 `sudo unlink /etc/resolv.conf` 以便与当前 resolv.conf 取消链接
4. `sudo mv /etc/resolv.conf.new /etc/resolv.conf`
5. 编辑 `/etc/wsl.conf` 并将此内容添加到文件。 （有关此设置的详细信息，请参阅高级设置配置）

```
[network]
generateResolvConf=false
```

6. 打开 `/etc/resolv.conf` 并执行以下操作
   a. 从文件中删除第一行，其中包含描述自动生成的注释
   b. 将上面步骤 (1) 中记下的 DNS 条目添加为 DNS 服务器列表中的第一个条目。
   c. 关闭该文件。

断开 VPN 连接后，必须将更改还原为 `/etc/resolv.conf`。 为此，请执行以下命令：

1. `cd /etc`
2. `sudo mv resolv.conf resolv.conf.new`
3. `sudo ln -s ../run/resolvconf/resolv.conf resolv.conf`

NAT 模式下 WSL 的 Cisco Anyconnect VPN 问题

Cisco AnyConnect VPN 修改路由的方式阻止了 NAT 正常工作。 有一种特定于 WSL 2 的解决方法：请参阅 Cisco AnyConnect 安全移动客户端管理员指南，版本 4.10 - 排查 AnyConnect 问题。

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 传统的 NAT 网络体系结构可以更新为一种全新的网络模式，称为“镜像网络模式”。 当实验性的 `networkingMode` 设置为 `mirrored` 时，Windows 上的网络接口将镜像到 Linux 中以提高兼容性。 有关详细信息，请参阅命令行博客：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

可以使用 WSL 配置文件的实验部分中的 `autoProxy` 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，以下内容适用于 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` 设置为在 Windows HTTPS 代理配置中安装的一个或多个 HTTP 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。
- 除 `WSL_PAC_URL` 之外的每个环境变量都设置为小写和大写。 例如 `HTTP_PROXY` 和 `http_proxy`。

有关详细信息，请参阅命令行博客：WSL 2023 年 9 月更新。

DNS 隧道的网络注意事项

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 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，从而无需发送网络数据包即可解析 DNS 名称。 此功能应提高网络兼容性，并可支持实现更好的 Internet 连接，即使你拥有 VPN、特定防火墙设置或其他网络配置。

可以使用 WSL 配置文件的实验部分中的 `dnsTunneling` 设置来配置 DNS 隧道。 应用此设置时，请注意以下注意事项：

- 启用 DNS 隧道时，如果网络有阻止 DNS 流量到达 8.8.8.8 的策略，则 WSL 中的本机 Docker 可能会出现连接问题
- 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。
- Linux 分发版中的 `/etc/resolv.conf` 文件有 3 个 DNS 服务器的最大限制，而 Windows 可能使用超过 3 个 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 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. 每个接口的 DNS 后缀
    4. 如果在 Windows 上启用了 DNS 加密（DoH、DoT），加密将应用于来自 WSL 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 容器（Docker Desktop 或在 WSL 中运行的本机 Docker）的 DNS 查询会绕过 DNS 隧道。 无法利用 DNS 隧道将主机 DNS 设置和策略应用于 Docker DNS 流量。

- Docker Desktop 采用自己的方式（不同于 DNS 隧道）将主机 DNS 设置和策略应用于来自 Docker 容器的 DNS 查询。

有关详细信息，请参阅命令行博客：WSL 2023 年 9 月更新。

将 Windows 主机收到的入站流量引导至 WSL 虚拟机时出现问题

使用镜像网络模式（实验性 `networkingMode` 设置为 `mirrored`）时，Windows 主机收到的一些入站流量永远不会被引导到 Linux VM。 此流量如下所示：

- UDP 端口 68 (DHCP)
- TCP 端口 135（DCE 终结点解析）
- UDP 端口 5353 (mDNS)
- 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。 不支持在使用镜像网络模式时对这些设置进行任何用户配置。 下面是 WSL 将配置的设置列表：

展开表

设置名称	值
https://sysctl-explorer.net/net/ipv4/accept_local/	启用 (1)
https://sysctl-explorer.net/net/ipv4/route_localnet/	启用 (1)
https://sysctl-explorer.net/net/ipv4/rp_filter/	禁用 (0)
https://sysctl-explorer.net/net/ipv6/accept_ra/	禁用 (0)
https://sysctl-explorer.net/net/ipv6/autoconf/	禁用 (0)
https://sysctl-explorer.net/net/ipv6/use_tempaddr/	禁用 (0)
addr_gen_mode	禁用 (0)
disable_ipv6	禁用 (0)
https://sysctl-explorer.net/net/ipv4/arp_filter/	启用 (1)

在默认网络命名空间下运行时，启用了镜像网络模式的 WSL2 中出现 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 (docker run –publish/-p) 的 Docker Desktop 容器。 WSL 团队正在与 Docker Desktop 团队合作解决此问题。 要解决此问题，请在 Docker 容器中使用主机的网络命名空间。 通过“docker run”命令中使用的“--network host”选项设置网络类型。 另一种解决方法是在 WSL 配置文件中的实验部分的 `ignoredPorts` 设置中列出已发布的端口号。

WSL 中的 DNS 后缀

根据 .wslconfig 文件中的配置，WSL 将具有以下行为 wrt DNS 后缀：

当 networkingMode 设置为 NAT 时：

案例 1）默认情况下，Linux 中不配置 DNS 后缀

案例 2）如果启用 DNS 隧道（.wslconfig 中的 dnsTunneling 设置为 true），则在 /etc/resolv.conf 的“搜索”设置中，在 Linux 中配置所有 Windows DNS 后缀

后缀在 /etc/resolv.conf 中按照以下顺序进行配置，类似于 Windows DNS 尝试后缀的顺序，即先解析 name: global DNS 后缀，然后解析补充 DNS 后缀，再解析每个接口的 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 隧道以及 SharedAccess DNS 代理（.wslconfig 中的 dnsTunneling 和 dnsProxy 都设置为 false），则在 /etc/resolv.conf 的“域”中，在 Linux 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 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。

当 networkingMode 设置为“已镜像”时：

所有 Windows DNS 后缀都在 Linux 中的 /etc/resolv.conf 的“搜索”设置中进行配置

后缀在 /etc/resolv.conf 中的配置顺序与在 NAT 模式案例 2）中的顺序相同

当 Windows DNS 后缀发生更改时，该更改将自动反映在 Linux 中

注意：在 Windows 中，可以使用 SetInterfaceDnsSettings - Win32 apps | Microsoft Learn 配置补充 DNS 后缀，带有标记“**DNS_SETTING_SUPPLEMENTAL_SEARCH_LIST 在“设置”参数中设置**”

在 WSL 中排查 DNS 问题

当 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;响应以反向发送回 WSL 容器。 此数据包转发过程到共享访问需要防火墙规则才能允许入站 DNS 数据包，当 WSL 最初要求 HNS 为其 WSL 容器创建 NAT 虚拟网络时，该数据包由 HNS 服务创建。

由于采用这种 NAT - 共享访问设计，有几个已知配置可能会中断 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，则会忽略 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，则如果为 WSL vNIC 分配该配置文件（默认值为“公用”），则会阻止该数据包。 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 means the locally defined firewall rules, like that by HNS, will not be applied or used.`

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。

此类设置将覆盖任何允许入站防火墙规则。 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。 若要运行此配置，**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：

Computer Configuration \ Administrative Templates \ Network \ Network Connections \ Windows Defender Firewall \ Domain Profile | Standard Profile

“Windows Defender 防火墙：不允许例外” - 已启用

从 MDM 策略：

./Vendor/MSFT/Firewall/MdmStore/PrivateProfile/Shielded

./Vendor/MSFT/Firewall/MdmStore/DomainProfile/Shielded

./Vendor/MSFT/Firewall/MdmStore/PublicProfile/Shielded

可通过运行以下命令，查看是否配置为不允许任何入站防火墙规则（请参阅上述关于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 means that no inbound Firewall rules will be applied.`

3.用户通过Windows 安全设置应用并检查“阻止所有传入连接（包括允许的应用列表中的连接）”的控件

Windows 支持用户选择加入上述案例 2 中引用的企业可应用的相同设置。 用户可以打开“Windows 安全”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”的控件。

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。

必须取消选中 NAT DNS 代理配置才能从 WSL 工作，**或者可以将 WSL 设置为使用 DNS 隧道。**

4.允许 DNS 数据包共享访问的 HNS 防火墙规则可能无效，引用以前的 WSL 接口标识符。 这是 HNS 中的一处缺陷，在最新的 Windows 11 版本已得到修复。 在早期版本中

如果发生这种情况，虽然不容易发现，但有一个简单的解决方法：

- 停止 WSL

  ```
  wsl.exe -shutdown
  ```

- 删除旧的 HNS 防火墙规则。 大多数情况下，此 Powershell 命令应正常工作：

  ```
  Get-NetFirewallRule -Name "HNS*" | Get-NetFirewallPortFilter | where Protocol -eq UDP | where LocalPort -eq 53 | Remove-NetFirewallRule
  ```

- 移除所有 HNS 终结点。 注意：如果使用 HNS 管理其他容器（例如 MDAG 或 Windows 沙盒），则还应停止这些容器。

  ```
  hnsdiag.exe delete all
  ```

- 重新启动或重启 HNS 服务

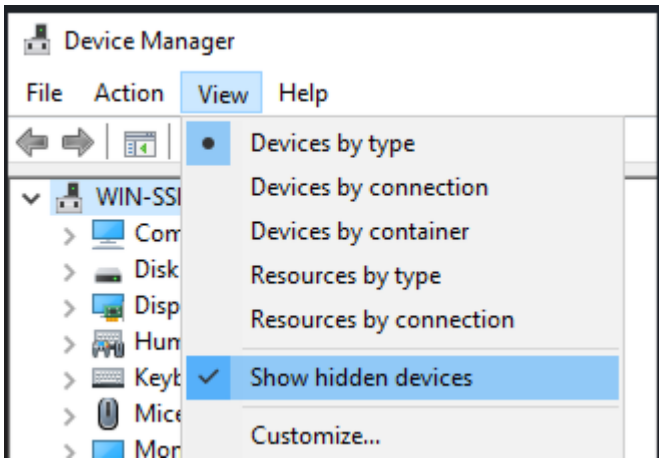
  ```
  Restart-Service hns
  ```

- 重启 WSL 后，HNS 将创建新的防火墙规则，并正确面向 WSL 接口。

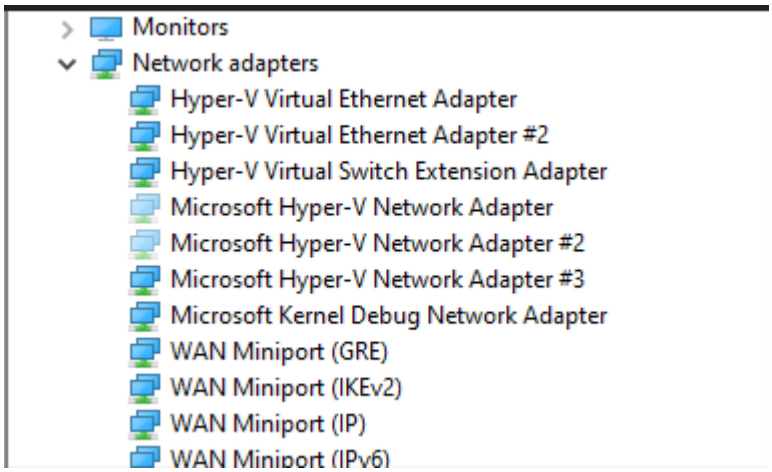
手动删除虚拟适配器的方法

幽灵适配器，或称虚拟即插即用 (PnP) 设备，指在系统中显示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 时看到幽灵适配器，请按照以下手动步骤查找和删除这些虚拟即插即用设备。 Microsoft 正在开发不需要手动干预的自动化解决方案。 更多信息即将推出。

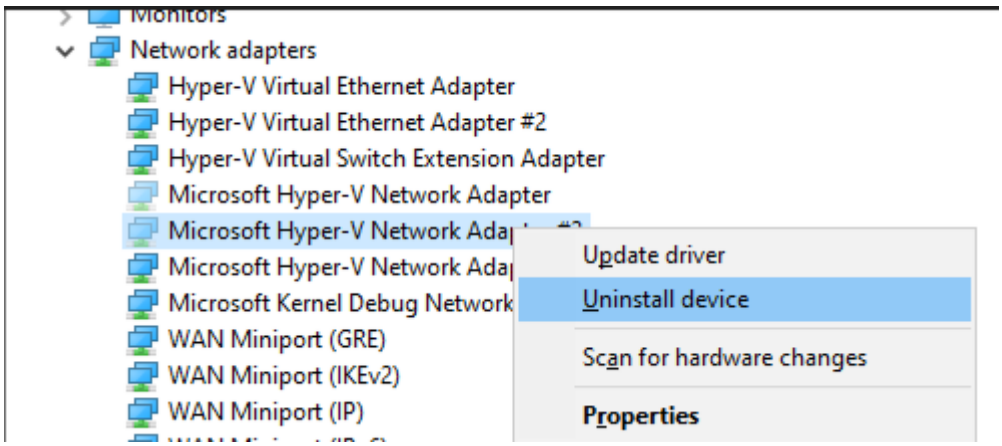
1. 打开“设备管理器”
2. 查看 >“显示隐藏的设备”



3. 打开网络适配器



4. 右键单击幽灵网络适配器并选择“卸载设备”



启动 WSL 或安装分发版会返回一个错误代码

按照 GitHub 上 WSL 存储库中收集 WSL 日志 的说明收集详细日志，并在我们的 GitHub 上提交问题。

更新 WSL

适用于 Linux 的 Windows 子系统有两个组件需要更新。

1. 若要更新适用于 Linux 的 Windows 子系统本身，请在 PowerShell 或 CMD 中使用命令 `wsl --update`。

2. 若要更新特定的 Linux 发行版用户二进制文件，请在要更新的 Linux 发行版中使用命令： `apt-get update | apt-get upgrade`。

Apt-get upgrade 错误

某些包使用我们尚未实现的功能。 例如，`udev` 尚不受支持，会导致多个 `apt-get upgrade` 错误。

若要解决与 `udev` 相关的问题，请执行以下步骤：

1. 将以下内容写入 `/usr/sbin/policy-rc.d` 并保存更改。

```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
```

Windows 更新后出现“错误:0x80040306”

此错误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：

1. 打开 cmd.exe
2. 右键单击标题栏 -> 选择“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”

在 Windows 更新期间可能禁用了“适用于 Linux 的 Windows 子系统”功能。 如果出现这种情况，则必须重新启用 Windows 功能。 在手动安装指南中可以找到有关启用“适用于 Linux 的 Windows 子系统”的说明。

更改显示语言

WSL 安装会尝试自动更改 Ubuntu 区域设置，使之与 Windows 安装的区域设置相匹配。 如果你不希望出现此行为，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 才能使此项更改生效。

以下示例将区域设置更改为 `en-US`：

```
Bash

sudo update-locale LANG=en_US.UTF8
```

Windows 系统还原后出现的安装问题

1. 删除 `%windir%\System32\Tasks\Microsoft\Windows\Windows Subsystem for Linux` 文件夹。
   注意：如果可选功能已完全安装且工作正常，请不要执行此操作。
2. 启用 WSL 可选功能（如果尚未这样做）
3. 重新启动
4. lxrun /uninstall /full
5. 安装 bash

在 WSL 中无法进行 Internet 访问

某些用户已报告特定的防火墙应用程序会阻止 WSL 中的 Internet 访问的问题。 报告的防火墙包括：

1. Kaspersky
2. AVG
3. Avast
4. Symantec Endpoint Protection

在某些情况下，关闭防火墙即可进行访问。 在某些情况下，只需让安装的防火墙在表面上阻止访问。

如果使用的是 Microsoft Defender 防火墙，取消选中“阻止所有传入连接，包括位于允许应用列表中的应用。”将允许访问。

使用 ping 时出现“权限被拒绝”错误

对于 Windows 周年更新版本 1607，在 WSL 中运行 ping 命令需要拥有 Windows 中的**管理员特权**。 若要运行 ping，请以管理员身份运行 Windows 上的 Ubuntu Bash，或使用管理员特权从 CMD/PowerShell 提示符运行 bash.exe。

对于更高版本的 Windows（内部版本 14926 及更高版本），则不再需要管理员特权。

Bash 挂起

如果使用 bash 时发现 bash 挂起（或死锁）且不响应输入，请收集并报告内存转储来帮助我们诊断问题。 请注意，这些步骤会导致系统崩溃。 如果你不熟悉此过程，请不要这样做，或者，请在执行此操作之前保存你的工作。

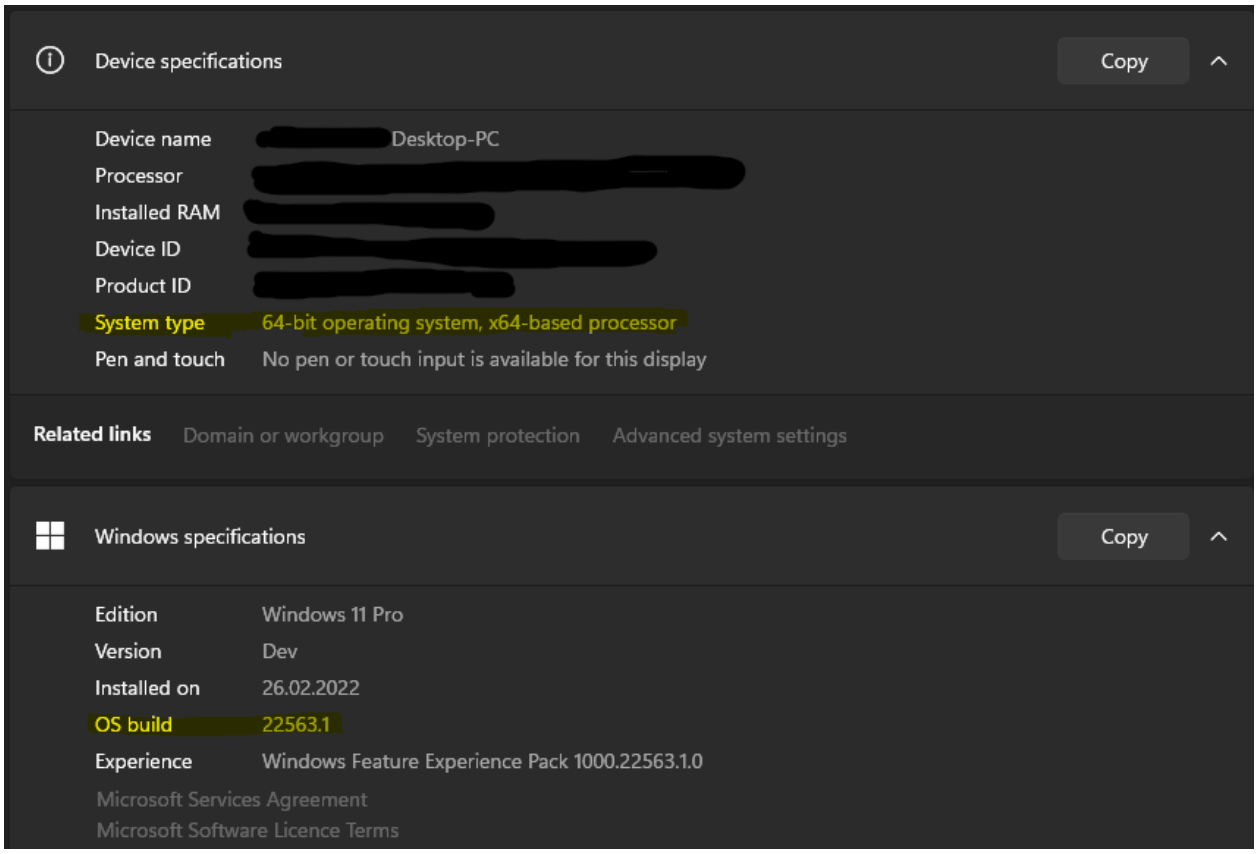
收集内存转储

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. 遵循这些步骤 使用键盘控制来配置崩溃。
3. 再现挂起或死锁场景。
4. 使用步骤 (2) 中的按键顺序来使系统崩溃。
5. 系统将会崩溃并收集内存转储。
6. 系统重新启动后，会将 memory.dmp 报告给 secure@microsoft.com。 转储文件的默认位置是 %SystemRoot%\memory.dmp 或 C:\Windows\memory.dmp（如果 C: 是系统驱动器）。 请注意，在电子邮件中，转储供 WSL 或 Windows 上的 Bash 团队参考。
7. 将内存转储类型还原为原始设置。

检查内部版本号

若要查找电脑的体系结构和 Windows 内部版本号，请打开
“设置”>“系统”>“关于”

查看“OS 内部版本”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

可通过在提升的 PowerShell 窗口中运行以下命令来确认是否已启用“适用于 Linux 的 Windows 子系统”：

PowerShell

```
Get-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

OpenSSH 服务器连接问题

尝试连接 SSH 服务器失败并出现以下错误：“127.0.0.1 端口 22 已关闭连接”。

1. 确保 OpenSSH 服务器正在运行：

   Bash

```
sudo service ssh status
```

并已按照此教程进行操作： https://ubuntu.com/server/docs/service-openssh

2. 停止 sshd 服务，然后在调试模式下启动 sshd：

```Bash
sudo service ssh stop
sudo /usr/sbin/sshd -d
```

3. 检查启动日志，确保已提供主机密钥，并且未看到如下所示的日志消息：

```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
```

如果确实看到了此类消息，并且 `/etc/ssh/` 下缺少主机密钥，则必须重新生成这些密钥，或者只是清除并安装 OpenSSH 服务器：

```BASH
sudo apt-get purge openssh-server
sudo apt-get install openssh-server
```

在启用 WSL 可选功能后出现“找不到引用的程序集。”

此错误与处于错误的安装状态相关。 请完成以下步骤来尝试解决此问题：

- 如果是从 PowerShell 运行了启用 WSL 功能的命令，请尝试改用 GUI，方法是打开“开始”菜单，搜索“启用或关闭 Windows 功能”，然后在列表中选择“适用于 Linux 的 Windows 子系统”，这将安装可选组件。

- 转到“设置”、“更新”，然后单击“检查更新”来更新 Windows 版本
- 如果这两个操作均失败，并且你需要访问 WSL，请考虑使用安装介质重新安装 Windows 并选择“保留所有内容”以确保保留你的应用和文件，从而就地升级。 可以在“重新安装 Windows 10”页面 中找到有关如何执行此操作的说明。

更正（SSH 相关）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` 文件：

```
Bash

[automount]
enabled = true
options = metadata,uid=1000,gid=1000,umask=0022
```

请注意，添加此命令将包含元数据并修改有关从 WSL 中看到的 Windows 文件的文件权限。 有关详细信息，请参阅文件系统权限。

无法通过在 Windows 上使用 OpenSSH 来远程使用 WSL

如果在 Windows 上使用 openssh-server 并尝试远程访问 WSL，则许多人会看到以下错误：

```
Windows 命令提示符

The file cannot be accessed by the system.
```

使用应用商店版本的 WSL 时，这是一个已知问题。 现在可以使用 WSL 1 或 Windows 内版本的 WSL 来暂时解决此问题。 有关详细信息，请参阅 https://aka.ms/wslstoreinfo 。

在分发中运行 Windows 命令失败

Microsoft Store 中提供的某些发行版尚不具备直接运行 Windows 命令的完全兼容性。 如果在运行 `powershell.exe /c start .` 或任何其他 Windows 命令时收到错误 `-bash: powershell.exe: command not found`，可按照以下步骤予以解决：

1. 在 WSL 分发中运行 `echo $PATH`。
   如果不包括：`/mnt/c/Windows/system32`，表示有什么重新定义了标准 PATH 变量。
2. 使用 `cat /etc/profile` 检查配置文件设置。
   如果其中包含 PATH 变量的分配，则使用 # 字符来编辑文件，以注释掉 PATH 分配块。
3. 检查 wsl.conf 是否存在 `cat /etc/wsl.conf`，并确保其中不包含 `appendWindowsPath=false`，否则请将其注释掉。
4. 通过键入 `wsl -t` 后跟分发名称，或者在 cmd 或 PowerShell 中运行 `wsl --shutdown` 来重启分发。

安装 WSL 2 后无法启动

我们注意到用户在安装 WSL 2 后无法启动的问题。 我们对该问题进行了全面诊断，而用户报告称更改缓冲区大小 或安装适当的驱动程序 可帮助解决此问题。 请查看此 GitHub 问题 了解有关此问题的最新更新。

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 服务来创建基础虚拟网络，WSL 2 依赖该网络来进行 NAT、DNS、DHCP 和主机连接共享。

禁用 ICS 服务 (SharedAccess) 或通过组策略禁用 ICS 会阻止创建 WSL HNS 网络。 这将导致在创建新的 WSL 版本 2 映像时失败，并在尝试将版本 1 映像转换为版本 2 时出现以下错误。

控制台

```
There are no more endpoints available from the endpoint mapper.
```

需要 WSL 2 的系统应将 ICS 服务 (SharedAccess) 保留在“手动(触发启动)”这一默认启动状态，并且应覆盖或删除禁用 ICS 的所有策略。 虽然禁用 ICS 服务会破坏 WSL 2，我们不建议禁用 ICS，但可以使用这些说明 禁用 ICS 部分。

使用旧版 Windows 和 WSL

如果运行的是旧版 Windows 和 WSL，例如 Windows 10 创意者更新（2017 年 10 月的内部版本 16299）或周年更新（2016 年 8 月的内部版本 14393），则有几处差异需要注意。 建议更新到最新的 Windows 版本，但如果无法执行此操作，请了解下面概述的一些差异。

互操作性命令差异：

- `bash.exe` 已替换为 `wsl.exe`。 可以从 Windows 命令提示或 PowerShell 运行 Linux 命令，但对于早期 Windows 版本，可能需要使用 `bash` 命令。 例如：`C:\temp> bash -c "ls -la"`。 传入 `bash -c` 的 WSL 命令将按原样转发到 WSL 进程。 必须以 WSL 格式指定文件路径，并且必须谨慎转义相关字符。 例如 `C:\temp> bash -c "ls -la /proc/cpuinfo"` 或 `C:\temp> bash -c "ls -la \"/mnt/c/Program Files\""`。
- 若要查看哪些命令适用于特定的分发版，请运行 `[distro.exe] /?`。 举个使用 Ubuntu 的例子：`C:\> ubuntu.exe /?`。
- Windows 路径包含在 WSL `$PATH` 中。
- 从早期的 Windows 10 版本中的 WSL 分发版调用 Windows 工具时，需要指定目录路径。 例如，若要通过 WSL 命令行调用 Windows 记事本应用，请输入：`/mnt/c/Windows/System32/notepad.exe`
- 若要将默认用户更改为 `root`，请在 PowerShell 中使用此命令：`C:\> lxrun /setdefaultuser root`，然后运行 Bash.exe 登录：`C:\> bash.exe`。 使用发行版密码命令（`$ passwd username`）重置密码，然后关闭 Linux 命令行（`$ exit`）。 在 Windows 命令提示符或 Powershell 中，将默认用户重置回常规的 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（2017 年 10 月的内部版本 16299）前的某个 Windows 10 版本上安装了 WSL，我们建议你将任何必需的文件、数据等从安装的旧 Linux 发行版迁移到通过 Microsoft Store 安装的新发行版。 若要从计算机中删除旧发行版，请通过命令行或 PowerShell 实例运行以下命令：`wsl --unregister Legacy`。 还可以选择手动删除旧发行版，方法是使用 Windows 文件资源管理器或 PowerShell 删除 `%localappdata%\lxss\` 文件夹（及其所有子内容）：`rm -Recurse $env:localappdata/lxss/`。

在 GitHub 上与我们协作

可以在 GitHub 上找到此内容的源，还可以在其中创建和查看问题和拉取请求。 有关详细信息，请参阅参与者指南。

Windows Subsystem for Linux 反馈

Windows Subsystem for Linux 是一个开放源代码项目。 选择一个链接以提供反馈：

提出文档问题

提供产品反馈

适用于 Linux 的 Windows 子系统发行说明

项目 • 2023/03/21

版本 21364

有关版本 21364 的一般 Windows 信息，请访问 Windows 博客。

- GUI 应用现已发布！ 有关详细信息，请参阅此博客文章。
- 解决通过 \\wsl.localhost\ 访问文件时发生的错误。
- 修复 LxssManager 服务中可能的死锁。

版本 21354

有关版本 21354 的一般 Windows 信息，请访问 Windows 博客。

- 当名为“wsl”的网络中存在计算机时，请将 \wsl 前缀切换到 \wsl.localhost 以避免出现问题。 \wsl$ 将继续有效。
- 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，使文件被标记为可执行文件 [GH 3847]
- 如果虚拟机平台未启用，请修复 wsl 服务故障。

内部版本 21286

有关内部版本 21286 的一般 Windows 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。
- 改进从 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 添加了一个用于启用启动命令的选项：

控制台

```
[boot]
command=<string>
```

版本 20226

有关内部版本 20226 的一般 Windows 信息，请访问 Windows 博客。

- 修复 LxssManager 服务中的故障。 [GH 5902]

内部版本 20211

有关内部版本 20211 的一般 Windows 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，请参阅访问 Windows 和 WSL 2 中的 Linux 文件系统。
- 解决检查 VM 是否处于空闲状态时 LxssManager 服务发生崩溃的问题。 [GH 5768]
- 支持压缩的 VHD 文件。 [GH 4103]
- 确保在 OS 升级期间保留安装到 c:\windows\system32\lxss\lib 的 Linux 用户模式库。 [GH 5848]
- 添加了功能：列出可使用 `wsl --install --list-distributions` 安装的可用分发。
- 现在，当用户注销时，WSL 实例会终止运行。

内部版本 20190

有关内部版本 20190 的一般 Windows 信息，请访问 Windows 博客。

- 修复阻止 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 选项以控制 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 wsl.exe 发行版的支持 - 安装。

内部版本 20175

有关内部版本 20175 的一般 Windows 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 8 GB（以较小者为准）[GH 4166]。
- 将 \\wsl$ 前缀更改为 \\wsl 以支持 URI 分析。 旧的 \\wsl$ 路径仍然受到支持。
- 在 amd64 上为 WSL2 默认启用嵌套虚拟化。 你可通过 %userprofile%\.wslconfig ([wsl2] nestedVirtualization=false) 来禁用此设置。
- 使 wsl.exe --update 要求启动 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 --install 命令行选项以管理对 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。

内部版本 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]
- 在未启用适用于 Linux 的 Windows 子系统可选组件时使 wsl.exe 存在，以提高功能的可发现性。
- 更改了 wsl.exe 以在未安装 WSL 可选组件时输出帮助文本
- 修复了创建实例时的争用条件
- 创建了包含所有命令行功能的 slclient.dll
- 防止了在 LxssManagerUser 服务停止期间发生崩溃
- 修复了当 distroName 参数为 NULL 时的 wslapi.dll 快速失败

内部版本 19041

有关内部版本 19041 的一般 Windows 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 内核更新到 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 内核更新到 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主机内存的 80%
- [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

有关内部版本 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

有关内部版本 19013 的一般 Windows 信息，请访问 Windows 博客。

- [WSL2] 提高 WSL 实用工具 VM 的内存性能。 不再处于使用状态的内存将释放回主机。
- [WSL2] 将内核版本更新到 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] 解决了有关处理某些 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

有关内部版本 18995 的一般 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] 在当前进程没有环境变量的情况下从互操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 内核版本更新到 4.19.72
- [WSL2] 添加了通过 .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

有关内部版本 18980 的一般 Windows 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 为了实现后向兼容而创建的所有符号链接（例如“C:\Document and Settings”），以及用户配置文件目录中的一些符号链接。
- 使意外的文件系统状态变得不严重 [GH 4334、4305]
- [WSL2] 添加当 CPU/固件支持虚拟化时对 arm64 的支持
- [WSL2] 允许无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 内核更新到 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] 修复大量 localhost 可靠性问题 [GH 4340]

内部版本 18970

有关内部版本 18970 的一般 Windows 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 Mount 命名空间中创建分发版。
- [WSL2] 修复当服务器直接绑定到 localhost 时的 localhost 端口中继 [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] 分发版未运行时弹出分发版 VHD。
- [WSL2] 修复配置分析库以正确处理空值。
- [WSL2] 通过创建跨分发版装入点来支持 Docker Desktop。 分发版可以通过将以下行添加到 /etc/wsl.conf 文件来启用此行为：

```
[automount]
crossDistro = true
```

内部版本 18945

有关内部版本 18945 的一般 Windows 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] 修复安装/转换失败和其他诊断，以跟踪将来的问题 [GH 4105]
- [WSL2] 改善 WSL2 网络问题的诊断
- [WSL2] 将内核版本更新到 4.19.55
- [WSL2] 使用 Docker 所需的配置选项更新了内核 [GH 4165]
- [WSL2] 增加分配给轻型实用程序 VM 的 CPU 数目，使其与主机相同（以前，内核配置中的 CONFIG_NR_CPUS 将数目限制为 8 个）[GH 4137]
- [WSL2] 为 WSL2 轻型 VM 创建交换文件
- [WSL2] 允许通过 \\wsl$\distro（例如 sshfs）显示用户装入点 [GH 4172]
- [WSL2] 改善 9p 文件系统的性能
- [WSL2] 确保 VHD ACL 不会无限增长 [GH 4126]
- [WSL2] 更新内核配置以支持 squashfs 和 xt_conntrack [GH 4107、4123]
- [WSL2] 修复 interop.enabled /etc/wsl.conf 选项 [GH 4140]
- [WSL2] 如果文件系统不支持 EA，则返回 ENOTSUP
- [WSL2] 修复 \\wsl$ 时 CopyFile 挂起的问题
- 将默认 umask 切换为 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]
- 引入 %UserProfile%\.wslconfig 文件用于调整 WSL2 设置

```
[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

有关内部版本 18917 的一般 Windows 信息，请访问 Windows 博客。

WSL

- WSL 2 现已推出！ 有关更多详细信息，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
- Plan 9：允许打开目录以使写入成功

内部版本 18890

有关内部版本 18890 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 非阻塞套接字泄露 [GH 2913]
- 在终端中输入 EOF 可能会阻塞后续读取 [GH 3421]
- 更新 resolv.conf 标头以引用 wsl.conf [在 GH 3928 中介绍]
- epoll delete 代码中的死锁 [GH 3922]
- 处理 --import 和 –export 的参数中的空格 [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

有关内部版本 18342 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 我们添加了相应的功能，使用户能够从 Windows 访问 WSL 分发版中的 Linux 文件。 可以通过命令行访问这些文件，此外，文件资源管理器、VSCode 等 Windows 应用可与这些文件交互。 通过导航到 \\wsl$\<distro_name> 访问文件，或通过导航到 \\wsl$ 来查看正在运行的发行版列表
- 添加额外的 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严重错误 [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

有关内部版本 18305 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 当主线程退出时，pthreads 失去对文件的访问权限 [GH 3589]
- TIOCSCTTY 应忽略"force"参数，除非该参数是必需的 [GH 3652]
- 改善 wsl.exe 命令行，并添加导入/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

有关内部版本 18272 的一般 Windows 信息，请访问 Windows 博客。

WSL

- **警告：** 此版本中存在一个导致 WSL 不可操作的问题。 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。 如果已安装此内部版本，可以使用“设置”->“更新和安全”->“恢复”中的“回退到 Windows 10 的上一个版本”回退到上一 Windows 内部版本。

内部版本 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]
- 在内部创建符号链接之前，确保 /bin 和 /sbin 目录存在 [GH 3584]
- 为 WSL 实例添加了实例终止超时机制。 超时目前设置为 15 秒，这意味着，实例将在上一个 WSL 进程退出 15 秒后终止。 若要立即终止分发版，请使用：

```
wslconfig.exe /terminate <DistributionName>
```

内部版本 17763 (1809)

有关内部版本 17763 的一般 Windows 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]
- 确保对启动时间使用无偏差的中断时间，以避免返回 clock_gettime(CLOCK_BOOTTIME) 的负值 [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]
- 修复以下问题：如果当前工作目录的长度少于 5 个字符，可能导致 WSL interop 失败 [GH 3379]
- 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 可执行文件不遵循符号链接命名约定 [GH 2909]
- 改善了 zombie 支持 [GH 1353]
- 添加 wsl.conf 项用于控制 Windows interop 行为 [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 应用也应该能够执行。
- 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 以停止正在运行的分发版。
- 修复 WSL shell 上下文菜单项无法正确处理包含空格的路径的问题。
- 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。

- 修复一个附加的问题：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 NETLINK 命令挂起问题。
- 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，以便在不使用 shell 的情况下调用单个二进制文件。
- 为 wsl.exe 添加 --distribution 选项，以选择特定的分发版。
- 对 dmesg 的有限支持。 现在，应用程序会将日志记录到 dmesg。 WSL 驱动程序会将有限的信息记录到 dmesg。 将来，此功能可能会扩展，以便从驱动程序发送其他信息/诊断数据。
  - 注意：目前通过 `/dev/kmsg` 设备接口支持 dmesg。 尚不支持 `syslog` syscall 接口。 此外，某些 `dmesg` 命令行选项（例如 `-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 对扩展属性的名称施加一些限制。 不允许使用某些字符（例如“/”、“:”和“*”），并且扩展属性名称在 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，移入单独的 tools 文件夹
- 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，但不为 S_IFREG 提供元数据支持
- 应为 DrvFs 上创建的只读文件设置只读属性 [GH 3411]
- 添加 /sbin/mount.drvfs 帮助器用于处理 DrvFs 装载
- 在 DrvFs 中使用 POSIX rename。
- 允许在无卷 GUID 的卷上执行路径转换。

内部版本 17738 (Fast)

有关内部版本 17738 的一般 Windows 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]
- 确保对启动时间使用无偏差的中断时间，以避免返回 clock_gettime(CLOCK_BOOTTIME) 的负值 [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)

有关内部版本 17728 的一般 Windows 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]
- 修复一些 AF_UNIX 相关的问题 [GH 3371]
- 修复以下问题：如果当前工作目录的长度少于 5 个字符，可能导致 WSL interop 失败 [GH 3379]

内部版本 18204 (Skip Ahead)

有关内部版本 18204 的一般 Windows 信息，请访问 Windows 博客。

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 可执行文件不遵循符号链接命名约定 [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 可执行文件不遵循符号链接命名约定 [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 wsl.conf 项用于控制 Windows interop 行为 [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 应用也应该能够执行。

内部版本 17704

有关内部版本 17704 的一般 Windows 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 以停止正在运行的分发版。

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 WSL shell 上下文菜单项无法正确处理包含空格的路径的问题。
- 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

有关内部版本 17686 的一般 Windows 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

有关内部版本 17677 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 修复一个附加的问题：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：某个问题会阻止 WSL 在某些 AMD 芯片组上运行 [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 NETLINK 命令挂起问题。
- 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，以便在不使用 shell 的情况下调用单个二进制文件。
- 为 wsl.exe 添加 --distribution 选项，以选择特定的分发版。

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 的有限支持。 现在，应用程序会将日志记录到 dmesg。 WSL 驱动程序会将有限的信息记录到 dmesg。 将来，此功能可能会扩展，以便从驱动程序发送其他信息/诊断数据。
  - 注意：目前通过 `/dev/kmsg` 设备接口支持 dmesg。 尚不支持 `syslog` sycall 接口。 此外，某些 `dmesg` 命令行选项（例如 `-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)

有关内部版本 17639 的一般 Windows 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 对扩展属性的名称施加一些限制。 具体而言，不允许使用某些字符（例如“/”、“:”和“*”），并且扩展属性名称在 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)

有关内部版本 17128 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 无

内部版本 17627 (Skip Ahead)

有关内部版本 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，WSL 功能目前不支持优先级，因此存在一些限制，但标准用法应该不会受到阻止。
- 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更多详细信息，请参阅链接
- 使用 wsl.exe 时遵循用户的默认 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：

正在测试。

内部版本 17618 (Skip Ahead)

有关内部版本 17618 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 引入用于 NT interop 的伪控制台功能 [GH 988、1366、1433、1542、2370、2406]。
- 传统的安装机制 (lxrun.exe) 已弃用。 支持用于安装分发版的机制是 Microsoft Store。

控制台

- 无修复措施。

LTP 结果：

正在测试。

版本 17110

有关内部版本 17110 的一般 Windows 信息，请访问 Windows 博客。

WSL

- 允许从 Windows 终止 /init [GH 2928]。
- 现在，DrvFs 默认使用按目录区分大小写（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：

正在测试。

内部版本 17107

有关内部版本 17107 的一般 Windows 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 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。
- 在从 17093 以前的内部版本迁移期间，升级 Linux 文件系统目录时会显示一条消息。 有关 17093 文件系统更改的更多详细信息，请参阅 17093 的发行说明。

控制台

- 无修复措施。

LTP 结果：

正在测试。

内部版本 17101

有关内部版本 17101 的一般 Windows 信息，请访问 Windows 博客。

WSL

- signalfd 支持。 [GH 129]
- 支持包含非法 NTFS 字符的文件名，现在会将这些字符编码为专用 Unicode 字符。 [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]
- 修复当注册表被篡改或损坏时，可能导致服务进入无限循环的问题。

- 修复 netlink 消息，以便能够使用更新版本（上游 4.14）的 iproute2。

控制台

- 无修复措施。

LTP 结果：

正在测试。

内部版本 17093

有关内部版本 17093 的一般 Windows 信息，请访问 Windows 博客 。

重要说明：

升级到此内部版本后，首次启动 WSL 时，需要执行一些操作来升级 Linux 文件系统目录。 这可能需要几分钟时间，因此 WSL 的启动速度看上去可能很慢。 对于从 Store 安装的每个分发版，只需执行此操作一次。

- 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：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 创建的新目录将标记为不区分大小写。
  - 注意：不会对旧版本中的 WSL 创建的目录设置此标志，因此，如果使用“case=dir”选项，则不会将这些目录视为区分大小写。 即将推出一种对现有目录设置此标志的方法。
  - 使用这些选项进行装载的示例（对于现有的驱动器，必须先卸载，然后才能使用不同选项装载）：sudo mount -t drvfs C: /mnt/c -o case=dir
  - 目前，case=force 仍是默认选项。 以后将更改为 case=dir。
- 现在，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 现在会在实例启动期间处理 /etc/fstab 文件 [GH 2636]。
  - 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 禁用此行为。
- /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。

可以使用 wsl.conf 更方便地配置 WSL

我们添加了一个方法，用于自动配置 WSL 中每次启动子系统时要应用的某些功能。 这包括自动装载选项和网络配置。 有关详细信息，请参阅博客文章：https://aka.ms/wslconf

AF_UNIX 允许在 WSL 上的 Linux 进程与 Windows 本机进程之间建立套接字连接

现在，WSL 和 Windows 应用程序可以通过 Unix 套接字相互通信。 假设你要在 Windows 中运行某个服务，并使其可在 Windows 和 WSL 应用中使用。 现在可以通过 Unix 套接字实现此目的。 有关详细信息，请参阅博客文章： https://aka.ms/afunixinterop

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]
- 加载包含偏移量非零的负载标头的 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 以静默方式终止进程的情况

控制台

- 无修复措施。

LTP 结果：

正在测试。

版本 17083

有关内部版本 17083 的一般 Windows 信息，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：

正在测试。

内部版本 17074

有关内部版本 17074 的一般 Windows 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：

正在测试。

内部版本 17063

有关内部版本 17063 的一般 Windows 信息，请访问 Windows 博客。

WSL

- DrvFs 支持其他 Linux 元数据。 这样，就可以使用 chmod/chown 设置文件的所有者和模式，并可以创建特殊文件，例如 fifos、Unix 套接字和设备文件。 默认情况下，此功能暂时处于禁用状态，因为它仍是试验性的。 注意：修复了 DrvFs 使用的元数据格式的 bug。 尽管试验性的元数据可在此内部版本中正常工作，但将来的内部版本无法正确读取此内部版本创建的元数据。 你可能需要手动更新已修改的文件的所有者，并且必须重新创建使用自定义设备 ID 的设备。

  若要启用元数据，请使用 metadata 选项装载 DrvFs（若要在现有装入点上启用，必须先将其卸载）：

  ```
  Bash

  mount -t drvfs C: /mnt/c -o metadata
  ```

  Linux 权限将作为附加元数据添加到文件；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，文件将还原为默认权限。

- 已将 mount 选项添加到 DrvFs，用于控制不包含元数据的文件。
  - uid：所有文件的所有者使用的用户 ID。
  - gid：所有文件的所有者使用的组 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
  ```

  与 metadata 选项相结合可以指定不包含元数据的文件的默认权限。

- 引入了新的环境变量 `WSLENV`，用于配置环境变量在 WSL 与 Win32 之间的流动方式。

  例如：

  ```
  Bash
  ```

```
WSLENV=GOPATH/l:USERPROFILE/pu:DISPLAY
```

`WSLENV` 是在从 Win32 启动 WSL 进程或者从 WSL 启动 Win32 进程时可以包含的环境变量的冒号分隔列表。 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：值是应在 WSL 路径与 Win32 路径之间进行转换的路径。
- l：值是路径列表。 在 WSL 中，它是冒号分隔的列表。 在 Win32 中，它是分号分隔的列表。
- u：仅当从 Win32 调用 WSL 时才应该包含该值
- w：仅当从 WSL 调用 Win32 时才应该包含该值

可以在 .bashrc 中或者在用户的自定义 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 适用于进程组 ID (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 bug 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。 请阅读此博客 来详细了解如何添加这两个新工具，以及它们如何打造 Windows 上的开发人员体验。

- `AF_UNIX` 适用于 Windows 预览体验成员 SDK (17061+)。 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 。

固定

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]
- 在满足特定的条件时，DrvFs 将创建 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 用于执行 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。

固定

WSL

- 自 17035 以来无修复措施。

控制台

- 自 17035 以来无修复措施。

LTP 结果：

正在测试。

内部版本 17035

有关内部版本 17035 的一般 Windows 信息，请访问 Windows 博客。

固定

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/删除行时丢失一些颜色。

LTP 结果：

正在测试。

内部版本 17025

有关内部版本 17025 的一般 Windows 信息，请访问 Windows 博客。

固定

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]。
- 如果未提供虚拟网桥名称，将生成默认名称 [GH 2497]。
- 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。

固定

WSL

- 忽略空的 ELF 程序标头 [GH 330]。
- 允许 LxssManager 为非交互式用户创建 WSL 实例（ssh 和计划任务支持）[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。
- tty/pty 文件的 O_ASYNC 支持。
- 其他改进和 bug 修复

控制台

- 此版本中没有控制台相关的更改。

LTP 结果：

正在测试。

Fall Creators Update

内部版本 16288

有关内部版本 16288 的一般 Windows 信息，请访问 Windows 博客。

固定

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

- 此版本中没有控制台相关的更改。

LTP 结果：

自 16273 以来无更改

内部版本 16278

有关内部版本 162738 的一般 Windows 信息，请访问 Windows 博客。

固定

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

- 此版本中没有控制台相关的更改。

LTP 结果：

自 16273 以来无更改

内部版本 16275

有关内部版本 162735 的一般 Windows 信息，请访问 Windows 博客。

固定

WSL

- 此版本中没有 WSL 相关的更改。

控制台

- 此版本中没有控制台相关的更改。

LTP 结果：

自 16273 以来无更改

内部版本 16273

有关内部版本 16273 的一般 Windows 信息，请访问 Windows 博客。

固定

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 套接字来取消阻止使用 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 实例时发生的拒绝访问错误 [GH 651、2095]
- 实现 futex FUTEX_REQUEUE 和 FUTEX_CMP_REQUEUE 操作 [GH 2242]
- 修复各种 SysFs 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，将从 RegisterDistribuiton 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 分发版
- 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

- 此版本中没有控制台相关的更改。

LTP 结果：

测试总数：1904
跳过的测试总数：209
失败总数：229

版本 16257

有关内部版本 16257 的一般 Windows 信息，请访问 Windows 博客。

固定

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 console/tty 的 copy/paste 行为，并添加更适当的完整缓冲区处理 [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]
- 重度使用后 /dev/ptmx 锁定 [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]
- 添加了新的色彩方案：https://blogs.msdn.microsoft.com/commandline/2017/08/02/updating-the-windows-console-colors/

LTP 结果：

自 16251 以来无更改

Syscall 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 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，修复程序将在 2017 年 8 月 14 日开始的周内，在预览体验成员内部版本中发布。

内部版本 16251

有关内部版本 16251 的一般 Windows 信息，请访问 Windows 博客。

固定

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

- 此版本中没有控制台相关的更改。

LTP 结果：

通过的测试数：768
失败的测试数：244
跳过的测试数：96

内部版本 16241

有关内部版本 16241 的一般 Windows 信息，请访问 Windows 博客。

固定

WSL

- 此版本中没有 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
- 修复代码页 65001 (utf8) 中不显示输出文本的问题
- 更改选择内容时，不会将对某种颜色的 RGB 值所做的更改传输到调色板的其他部分。 这在一定程度上使得控制台属性表变得更易于使用。
- Ctrl+S 似乎不起作用
- ANSI 转义代码中根本没有 Un-Bold/-Dim [GH 2174]
- 控制台不正常支持 Vim 色彩主题 [GH 1706]
- 无法粘贴特定的字符 [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 上显示 VT 时的控制台呈现 bug [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

版本 16237

有关内部版本 16237 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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版的支持
- 修复 getdents 和 getdents64 返回的项的填充
- 修复针对 shmctl SHM_STAT 系统调用的权限检查 [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。

固定

- 此版本中没有 WSL 相关的更改。

内部版本 16226

有关内部版本 16226 的一般 Windows 信息，请访问 Windows 博客。

固定

- xattr 相关的 syscall 支持（getxattr、setxattr、listxattr、removexattr）。
- security.capablity xattr 支持。
- 改善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]
- 改善了对 OneDrive 占位符、GVFS 占位符和精简 OS 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。

固定

- WSL 不再需要开发人员模式。
- 支持 drvfs 中的目录接合。
- 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分发版的功能。

- 添加了对 TCP 套接字的 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。

固定

- 这些版本中没有 WSL 相关的更改。

内部版本 16193

有关内部版本 16193 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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。

固定

- 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 现在会从 Windows hosts 文件 (%windir%\system32\drivers\etc\hosts) 继承项 [GH 1495]

内部版本 16179

有关内部版本 16179 的一般 Windows 信息，请访问 Windows 博客。

固定

- 本周没有 WSL 更改。

内部版本 16176

有关内部版本 16176 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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
  - 在 drvfs 中添加远程驱动器的支持
  - 在 drvfs 中启用 FAT 支持
- 其他修复和改进

LTP 结果

自 15042 以来无更改

内部版本 16170

有关内部版本 16170 的一般 Windows 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。

固定

- 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 以来无更改

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。

内部版本 15042

有关内部版本 15042 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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：

通过的测试数：737
未通过的测试数（失败、已跳过等）：255

内部版本 15031

有关内部版本 15031 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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。
- 现在会在 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：

通过的测试数：737
未通过的测试数（失败、已跳过等）：255

内部版本 15025

有关内部版本 15025 的一般 Windows 信息，请访问 Windows 博客。

固定

- 修复破坏 grep 2.27 的 bug [GH 1578]
- 为 eventfd2 syscall 实现了 EFD_SEMAPHORE 标志 [GH 452]
- 实现了 /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引用同一管道，则 syscall 会返回失败
- 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
- 做出更改以提高 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：

通过的测试数：732
未通过的测试数（失败、已跳过等）：255

内部版本 15019

有关内部版本 15019 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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
- 修复 unix 套接字 getsockopt SO_ERROR 以启用 postgress [GH 61、1354]
- 为 GO 语言实现 /proc/sys/net/core/somaxconn
- Apt-get package update 后台任务现在会在视图中以隐藏方式运行
- 清除 ipv6 localhost 的范围（Spring-Framework(Java) 失败）。
- 其他修复和改进

LTP 结果：

通过的测试数：714
未通过的测试数（失败、已跳过等）：249

内部版本 15014

有关内部版本 15014 的一般 Windows 信息，请访问 Windows 博客。

固定

- Ctrl+C 现在可按预期方式工作
- htop 和 ps auxw 现在会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：

通过的测试数：709
未通过的测试数（失败、已跳过等）：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

有关内部版本 15007 的一般 Windows 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。 这包括将充当普通“c”按键的 ctrl-c 命令。
  - 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：`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

固定

- 更正了运行 WSL 会消耗 100% 的 CPU 核心的问题
- 现在支持套接字选项 IP_PKTINFO、IPV6_RECVPKTINFO。 （GH #851、987）
- 在 lxcore 中将网络接口物理地址截断为 16 个字节（GH #1452、1414、1343、468、308）
- 其他修复和改进

LTP 结果：

通过的测试数：709
未通过的测试数（失败、已跳过等）：255

内部版本 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。 这包括将充当普通“c”按键的 ctrl-c 命令。
  - 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：`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。 根本原因已解决并已在内部修复。

固定

- 现在，必须在同一权限级别创建所有 bash 会话。 尝试在不同级别启动会话将遭到阻止。 这意味着，管理员和非管理员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)
- 修复 /dev/kmsg 的模式，以允许进行组访问/其他读取访问 (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)
- 已将默认 TERM 环境变量切换为 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)
- 修复了不正确本地化快捷方式名称的错误 (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 AptPackageIndexUpdate 窗口不会关闭的问题（GH #1184，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)
- 针对 patchelf 二进制文件的 Elf 分析不再失败。 (GH #471)
- VPN DNS 已传播到 /etc/resolv.conf（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，在打开过多的文件时可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：

通过的测试数：690
未通过的测试数（失败、已跳过等）：274

Syscall 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

有关内部版本 14986 的一般 Windows 信息，请访问 Windows 博客。

固定

- 修复了 Netlink 和 Pty IOCTL 的 bug 检查
- 内核版本现在会报告 4.4.0-43，以便与 Xenial 保持一致
- 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 0.000ms 的问题 (GH #1296)
- 打开过多的文件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，该实用工具将会崩溃。 这是“ip”（而不是 WSL）中的一个 bug。 “ip”实用工具会将用于输出硬件地址的字符串缓冲区的长度进行硬编码，而该缓冲区太小，无法输出 32 字节硬件地址。
- 其他修复和改进

LTP 结果：

通过的测试数：669
未通过的测试数（失败、已跳过等）：258

Syscall 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

有关内部版本 14971 的一般 Windows 信息，请访问 Windows 博客。

固定

- 由于这种情况超出了我们的控制，在此内部版本的适用于 Linux 的 Windows 子系统中未做更新。 定期计划的更新将在下一版本中恢复。

LTP 结果：

自 14965 以来无更改
通过的测试数：664
未通过的测试数（失败、已跳过等）：263

内部版本 14965

有关内部版本 14965 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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 用户路径现在默认会追加到 WSL 路径（即，现在可以键入 notepad.exe，无需将 System32 添加到 Linux 路径）
- 添加了对 /proc/sys/kernel/cap_last_cap 的支持
- 现在，如果当前工作目录包含非 ANSI 字符，则可以从 WSL 启动 NT 二进制文件 (GH #1254)
- 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
- 修复错误地假设解释器路径长度必须小于 64 个字符的 ELF 验证逻辑。 (GH #743)
- 打开文件描述符会使控制台窗口保持打开状态 (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
- 修复了计时器精度导致的 0.000ms ping 错误。
- 实现了 /proc/self/environ (GH #730)
- 其他 bug 修复和改进

LTP 结果：

通过的测试数：664
未通过的测试数（失败、已跳过等）：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。

固定

- 改善了 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
- 修复了在启用企业数据保护 (EDP) 后启动 bash.exe 时出现的错误 0x80070057
- 其他 bug 修复和改进

LTP 结果：

通过的测试数：665
未通过的测试数（失败、已跳过等）：263

内部版本 14955

有关内部版本 14955 的一般 Windows 信息，请访问 Windows 博客。

固定

- 由于这种情况超出了我们的控制，在此内部版本的适用于 Linux 的 Windows 子系统中未做更新。 定期计划的更新将在下一版本中恢复。

LTP 结果：

通过的测试数：665
未通过的测试数（失败、已跳过等）：263

内部版本 14951

有关内部版本 14951 的一般 Windows 信息，请访问 Windows 博客。

新功能：Windows/Ubuntu 互操作性

现在可以直接从 WSL 命令行调用 Windows 二进制文件。 这样，用户便可以通过前所未有的方式来与其 Windows 环境和系统交互。 举个简单的例子，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
```

可在以下资源中找到详细信息：

- WSL 团队的互操作博客
- WSL 文件系统文档

固定

- 现在将为所有新的 WSL 实例安装 Ubuntu 16.04 (Xenial)。 使用现有 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视大小，而不是缓冲区大小
- 其他 bug 修复和改进

LTP 结果：

通过的测试数：665
未通过的测试数（失败、已跳过等）：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。

固定

- 修复了阻止使用包含空格或引号的 NT 用户名为用户创建 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。 例如，可运行以下命令：
  - bash -c "echo data" | 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 现在会显示在 mountinfo 中，并显示正确的名称
- 其他 bug 修复和改进

LTP 结果：

通过的测试数：665
未通过的测试数（失败、已跳过等）：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。

固定

- 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 现在支持从 DrvFs 上的 Windows 应用程序生成通知
- 实现 mongod 的 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 bug 修复和改进

LTP 结果：

通过的测试数：665
未通过的测试数（失败、已跳过等）：263

Syscall 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：在即将发布的版本中，WSL 将会安装 Ubuntu 版本16.04 (Xenial) 而不是 Ubuntu 14.04 (Trusty)。 此更改将应用到安装新实例的预览体验版（lxrun /install 或首次运行 bash.exe）。 包含 Trusty 的现有实例不会自动升级。 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 检查将自身显示为崩溃，出现错误"尝试执行不可执行的内存"。 此问题的最常见表现形式是使用 ssh 时发生崩溃。 根本原因已在内部版本中修复，将会尽快推送到预览体验版。

固定

- 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：对源自 Windows 应用程序的更改的 Inotify 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)
- 在 .tar 文件中正确处理不包含 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 现在可以重定向到文件
- 其他 bug 修复和改进

LTP 结果：

通过的测试数：664
未通过的测试数（失败、已跳过等）：264

Syscall 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。

固定

- 由于这种情况超出了我们的控制，在此内部版本的适用于 Linux 的 Windows 子系统中未做更新。 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。

固定

- Ping 现在可以在没有管理员特权的控制台中正常运行
- 现在支持 Ping6，也无需管理员特权
- 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 事件：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：修改 Windows 应用程序中的文件不会生成任何事件
- Unix 套接字现在支持 SCM_CREDENTIALS

LTP 结果：

通过的测试数：651
未通过的测试数（失败、已跳过等）：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。

固定

- 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 send、recv 和 shutdown 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：功能键不起作用
- 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

有关内部版本 14905 的一般 Windows 信息，请访问 Windows 博客。

固定

- 现在支持可重启系统调用（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 bug 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。

固定

- 修复了尾随斜杠问题
  - `$ mv a/c/ a/b/` 等命令现在可正常运行
- 安装程序现在会提示是否应将 Ubuntu 区域设置指定为 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 中使用竖线分隔 bash -c 命令。 示例：>dir | bash -c "grep foo"
- Bash 现在可以安装在包含多个页面文件的系统上（GH #538、#358）
- 默认的 INET 套接字缓冲区大小应与默认 Ubuntu 安装的大小相匹配
- 将 xattr syscall 与 listxattr 对齐
- 仅返回使用 SIOCGIFCONF 中有效 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

Syscall 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

有关内部版本 14388 的一般 Windows 信息，请访问 Windows 博客。

固定

- 修复在 8/2 准备 Windows 10 周年更新时出现的问题
  - 可在我们的博客中找到有关周年更新中的 WSL 的详细信息。

内部版本 14376

有关内部版本 14376 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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）
- 修复了与网络相关的常见蓝屏问题
- 修复了访问 /proc/[pid]/task 时出现蓝屏的问题 (GH #523)
- 修复了在某些 pty 方案中 CPU 利用率偏高的问题（GH #488、#504）
- 其他 bug 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。

固定

- 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 bug 修复和改进

内部版本 14366

有关内部版本 14366 的一般 Windows 信息，请访问 Windows 博客。

固定

- 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 bug 修复和改进

Syscall 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

有关内部版本 14361 的一般 Windows 信息，请访问 Windows 博客。

固定

- 现在，在 Windows 上的 Ubuntu Bash 中运行时，DrvFs 区分大小写。
  - 用户可在其 /mnt/c 驱动器中保存 case.txt 和 CASE.TXT
  - 只有 Windows 上的 Ubuntu Bash 支持区分大小写。 在 Bash 外部，NTFS 会正确报告文件，但在与 Windows 中的文件交互时，可能会出现意外的行为。
  - 每个卷的根目录（即 /mnt/c）不区分大小写
  - 可在此处找到有关处理 Windows 中的这些文件的详细信息。
- 大幅增强了 pty/tty 支持。 现在支持 TMUX 等应用程序 (GH #40)
- 修复了不总会创建用户帐户的安装问题
- 优化了命令行参数结构，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)

- 现在可在 tty 设备上正常运行 chmod 和 chown
- 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）
- 用户现在可以选择禁用自动生成的 hosts 文件 (GH #398)
- 在安装期间自动将 Linux 区域设置与 NT 区域设置进行匹配 (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 ~。 现在支持类似于“bash ~ -c ls”的命令 (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)
- 改善了对 xEmacs 等 x11 应用的支持 (GH #481)
- 更新了初始线程堆栈大小，以匹配默认的 Ubuntu 设置，并正确地向 get_rlimit syscall 报告大小（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

Syscall 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。

固定

- 修复了不正常下载/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 中的问题，允许超过 255 个字符的文件名
- 改善了对非英语字符的支持
- 添加当前 Windows 时区数据并将其设为默认值
- 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)
- 更新了 resolv.conf 的格式，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)

Syscall 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 。

在 WSL 博客中可以找到有关 VolFs 和 DriveFs 的信息。

固定

- 修复了 Windows 用户在用户名中包含 Unicode 字符时出现的安装问题
- 现在，在首次运行时，默认会提供“常见问题解答”中的 apt-get update udev 解决方法
- 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 .// 现在可按预期方式工作
- DriveFs 上的 npm install 和 -g 选项现在可正常工作

- 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），以便与本机 Ubuntu 更匹配
- 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 可正常运行
- 现在支持预编译的 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 现在会返回正确的信息
- VFS 现在支持套接字
- /dev 现在装载为 tempfs
- 现在支持 Fifo
- 多核系统现在会在 /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
- Syscall 改进和 bug 修复。 下面列出了支持的 syscall。
- 其他 bug 修复和改进

已知问题

- 在某些情况下， 不会正确解析 DriveFs 上的“..”

Syscall 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`

内部版本 14332

有关内部版本 14332 的一般 Windows 信息，请访问 Windows 博客 。

固定

- 更好地生成 resolv.conf，包括指定 DNS 项的优先级
- 在 /mnt 与非 /mnt 驱动器之间移动文件和目录时出现的问题
- 现在可以使用符号链接创建 Tar 文件。
- 创建实例时会添加默认的 /run/lock 目录
- 更新 /dev/null 以返回正确的统计信息
- 首次运行期间下载内容时出现的其他错误
- Syscall 改进和 bug 修复。 下面列出了支持的 syscall。
- 其他 bug 修复和改进

Syscall 支持

下面是在 WSL 中具有某种实现的新 syscall。 此列表中的 syscall 至少在一种方案中受支持，但目前其所有参数不一定都受支持。

`READLINKAT`

内部版本 14328

有关内部版本 14332 的一般 Windows 信息，请访问 Windows 博客 。

新功能

- 现在支持 Linux 用户。 安装 Windows 上的 Ubuntu Bash 时会提示创建 Linux 用户。 有关详细信息，请访问 https://aka.ms/wslusers
- 现在，主机名将设置为 Windows 计算机名，而不再是 @localhost
- 有关内部版本 14328 的详细信息，请访问： https://aka.ms/wip14328

固定

- 非 `/mnt/<drive>` 文件的符号链接改进
  - 现在可正常运行 npm install
  - 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。 此功能将在以后的内部版本中可用
- 现在会显示 top 和 htop
- 有关某些安装失败的其他错误消息
- Syscall 改进和 bug 修复。 下面列出了支持的 syscall。

- 其他 bug 修复和改进

Syscall 支持

下面是在 WSL 中具有某种实现的 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`

适用于 Linux 的 Windows 子系统内核发行说明

项目 • 2023/03/21

我们添加了对 WSL 2 分发的支持，这些分发使用完整 Linux 内核。 此 Linux 内核是开源的，WSL2-Linux-Kernel 存储库中提供了其源代码。 此 Linux 内核通过 Microsoft 更新传递到计算机，并按单独的发布计划发布到适用于 Linux 的 Windows 子系统，该子系统作为 Windows 映像的一部分提供。

5.15.57.1

发布日期：预发行版 2022/08/02

官方 Github 版本链接

- 基于 v5.15 内核系列的 WSL2 内核的初始版本
- 版本 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
- 修复自上一个 v5.10 WSL2 内核以来的 9p 文件系统回归
- 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/
- 启用其他控制组 (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 版本链接

- 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[] 的边界外数组访问
- 将等待同步 VM 总线消息实现为“可终止”，以允许终止等待同步调用主机的进程
- 当进程被销毁时刷新设备以实现终止，以避免在来宾进程终止时出现死锁

5.10.93.2

发布日期：预发行版 2022/02/08

官方 Github 版本链接

- 版本 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 版本说明，以包含 pahole 的 dwarves 依赖项
- 已将 Dxgkrnl 版本切换到 2111
- 取消了现有和总 sysmem 分配的限制
- 在进程清理过程中正确刷新设备以实现终止
- 修复了 d3dkmthk.h 的 SPDX-License-Identifier

5.10.81.1

发布日期：预发行版 2022/02/01

官方 Github 版本链接

- 版本 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
- 启用非拱形特定的 ACPI 选项
- 启用与设备映射器 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 版本链接

- 版本 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 使用启用缓冲区共享和同步文件框架 (CONFIG_DMA_SHARED_BUFFER, CONFIG_SYNC_FILE)
- 修复了 8.1 之前的 GCC 版本的 Dxgkrnl 生成失败 (microsoft/WSL#7558)

5.10.60.1

发布日期：2021/11/02（预发行版 2021/10/05）

官方 Github 版本链接

- 版本 rolling-lts/wsl/5.10.60.1
- 更新到上游稳定内核版本 5.10.60
- 通过对 PCI BAR 相对地址的支持启用 virtio-pmem
- 在 Hyper-V 下启用对 arm64 的 vPCI 支持
- 启用 io_uring 支持
- 启用 USB over IP 支持
- 启用对 x86_64 的半虚拟化 spinlock 支持
- 刷新 dxgkrnl 驱动程序以选取 bug 修复和代码清理
- 启用对 NFSv4.1 的 NFS 客户端支持
- 启用 USB 内核配置选项来使用 Arduino over USB 进行交互
- 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 版本链接

- 版本 rolling-lts/wsl/5.10.43.3
- 更新到上游稳定内核版本 5.10.43
- 改进的 dxgkrnl 驱动程序
- 新版本的 arm64 Linux on Hyper-V 系列 (v9)
- 始终使用 arm64 来宾上的 Hyper-V 超级调用接口支持在所有版本的 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 版本链接

- 修复了 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 版本链接

5.4.72

发布日期：2021/01/21

官方 Github 版本链接

- 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 版本链接。

- 5\.4.51 的稳定版本

4.19.128-microsoft-standard

发布日期：2020/09/15

官方 Github 版本链接。

- 这是 4.19.128 的稳定版本
- 修复 dxgkrnl 驱动程序 IOCTL 内存损坏的问题

4.19.121-microsoft-standard

发布日期：预发行版

官方 Github 版本链接。

- Drivers: hv: vmbus: hook up 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 版本链接。

- 更新 4.19.104 的 WSL 配置

4.19.84-microsoft-standard

发布日期：2019/12/11

官方 Github 版本链接。

- 这是 4.19.84 稳定版本

Microsoft Store 中适用于 Linux 的 Windows 子系统的发行说明

项目 • 2023/03/21

在 Microsoft/WSL Github 存储库发行版页面 上可找到 Microsoft Store 内部的 WSL 的发行说明。 有关最新更新，请参阅该列表。

已知问题：

- 目前无法在 Linux 中通过 session zero 启动适用于 Linux 的 Windows 子系统（例如从 ssh 连接中）。