TOPICS 技巧和窍门 ▼

远程开发技巧和窍门

(https://github.com/Microsoft/vscode-docs/blob/master/docs/remote/troubleshooting.md)

本文介绍了每个Visual Studio Code 远程开发 (https://aka.ms/vscode-remote/download/extension)扩展的疑难解答提示和技巧。有关设置和使用每个特定扩展的详细信息，请参见SSH (/docs/remote/ssh)，Containers (/docs/remote/containers)和WSL (/docs/remote/wsl)文章。或者尝试逐步教程， (/docs/remote/remote-tutorials)以帮助您在远程环境中快速运行。

可在服务的文档中 (https://aka.ms/vso-docs/troubleshooting)找到Visual Studio Codespaces的 (https://aka.ms/vso)疑难解答提示。 (https://aka.ms/vso-docs/troubleshooting)

SSH技巧

SSH功能强大且灵活，但这也增加了设置的复杂性。本节包括一些提示和技巧，这些提示和技巧可用于在不同环境中启动和运行Remote-SSH扩展。

如果仍然遇到问题，则可能需要尝试Visual Studio Codespaces免费的自托管环境选项 (https://aka.ms/vso-docs/vscode)的预览，因为它不需要SSH服务器，甚至不需要远程主机上的打开/直接访问的端口。该服务还允许您将其基于浏览器的编辑器与注册的主机一起使用。

配置基于密钥的身份验证

SSH公钥身份验证 (https://www.ssh.com/ssh/public-key-authentication)是一种便捷的高安全性身份验证方法，它将本地"私钥"与您与SSH主机上的用户帐户关联的"公钥"结合在一起。本节将引导您如何生成这些密钥并将其添加到主机。

> **提示：** Windows不支持 PuTTY 客户端，但是您可以转换PuTTYGen密钥。

快速入门：使用SSH密钥

为远程主机设置基于SSH密钥的身份验证。首先，我们将创建一个密钥对，然后将公共密钥复制到主机。

创建本地SSH密钥对

检查**本地**计算机上是否已经有SSH密钥。它通常位于 `~/.ssh/id_rsa.pub` macOS / Linux上，并且 `.ssh` 位于Windows上的用户配置文件文件夹中的目录（例如 `C:\Users\your-user\.ssh\id_rsa.pub` ）。

如果没有密钥，请在**本地**终端/ PowerShell中运行以下命令以生成SSH密钥对：

```
ssh-keygen -t rsa -b 4096
```

> **提示：** 没有 `ssh-keygen` ？ 安装受支持的SSH客户端。

授权您的macOS或Linux机器进行连接

在**本地终端窗口中**运行以下命令之一，以适当的方式替换用户名和主机名，以将本地公共密钥复制到SSH主机。

- 连接到**macOS或Linux** SSH主机：

```
export USER_AT_HOST="your-user-name-on-host@hostname"
export PUBKEYPATH="$HOME/.ssh/id_rsa.pub"

ssh-copy-id -i "$PUBKEYPATH" "$USER_AT_HOST"
```

- 连接到**Windows** SSH主机：

```
export USER_AT_HOST="your-user-name-on-host@hostname"
export PUBKEYPATH="$HOME/.ssh/id_rsa.pub"

ssh $USER_AT_HOST "powershell New-Item -Force -ItemType Directory -Path \"\$HOME\\.ssh\"; Add-Content -Force -Path \"\$HOME\\.ssh\\author
ized_keys\" -Value '$(tr -d '\n\r' < "$PUBKEYPATH")'"
```

  您可能需要验证**SSH主机上远程用户** `authorized_key` 的 `.ssh` 文件夹中的文件是否归您所有，并且没有其他用户有权访问该文件。有关详细信息，请参见OpenSSH Wiki (https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH#authorized_keys)。 (https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH#authorized_keys)

授权Windows计算机连接

在**本地PowerShell**窗口中运行以下命令之一，并根据需要替换用户名和主机名，以将本地公共密钥复制到SSH主机。

- 连接到**macOS或Linux** SSH主机：

```
$USER_AT_HOST="your-user-name-on-host@hostname"
$PUBKEYPATH="$HOME\.ssh\id_rsa.pub"

$pubKey=(Get-Content "$PUBKEYPATH" | Out-String); ssh "$USER_AT_HOST" "mkdir -p ~/.ssh && chmod 700 ~/.ssh && echo '${pubKey}' >> ~/.ssh/
authorized_keys && chmod 600 ~/.ssh/authorized_keys"
```

- 连接到**Windows** SSH主机：

```
$USER_AT_HOST="your-user-name-on-host@hostname"
$PUBKEYPATH="$HOME\.ssh\id_rsa.pub"

Get-Content "$PUBKEYPATH" | Out-String | ssh $USER_AT_HOST "powershell `"New-Item -Force -ItemType Directory -Path `"`$HOME\.ssh`"; Add-C
ontent -Force -Path `"`$HOME\.ssh\authorized_keys`" `""
```

验证**SSH主机上远程用户** `authorized_key` 的 `.ssh` 文件夹中的文件是否归您所有，并且没有其他用户有权访问该文件。有关详细信息，请参见OpenSSH Wiki (https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH#authorized_keys)。
(https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH#authorized_keys)

使用专用钥匙提高安全性

在所有SSH主机上使用单个SSH密钥虽然很方便，但是如果任何人都可以访问您的私钥，那么他们也将有权访问您的所有主机。您可以通过为开发主机创建单独的SSH密钥来防止这种情况。只需按照以下步骤操作：

1. 在其他文件中生成单独的SSH密钥。

   macOS / Linux：在**本地终端上**运行以下命令：

   ```
   ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa-remote-ssh
   ```

   Windows：在**本地PowerShell中**运行以下命令：

   ```
   ssh-keygen -t rsa -b 4096 -f "$HOME\.ssh\id_rsa-remote-ssh"
   ```

2. 请按照快速入门中的相同步骤对SSH主机上的密钥进行授权，但是将设置 `PUBKEYPATH` 为 `id_rsa-remote-ssh.pub` 文件。

3. 在VS Code中，运行**Remote-SSH：在命令面板（ F1 ）中打开"配置文件...**"，选择一个SSH配置文件，然后添加（或修改）主机条目，如下所示：

   ```
   Host name-of-ssh-host-here
       User your-user-name-on-host
       HostName host-fqdn-or-ip-goes-here
       IdentityFile ~/.ssh/id_rsa-remote-ssh
   ```

   > **提示**：您也可以将其 / 用于Windows路径。如果使用 \ ，则需要使用两个斜杠。例如， `C:\\path\\to\\my\\id_rsa` 。

重用PuTTYGen中生成的密钥

如果使用PuTTYGen为要连接的主机设置SSH公钥身份验证，则需要转换私钥，以便其他SSH客户端可以使用它。去做这个：

1. **在本地**打开PuTTYGen **，**然后加载要转换的私钥。
2. 从应用程序菜单中选择**转换>导出OpenSSH密钥。**将转换后的密钥保存到用户配置文件文件夹中目录下的**本地**位置 `.ssh` （例如 `C:\Users\youruser\.ssh` ）。
3. 验证此新**本地**文件归您所有，并且没有其他用户有权访问该文件。
4. 在VS Code中，运行**Remote-SSH：在命令面板（ F1 ）中打开"配置文件...**"，选择要更改的SSH配置文件，然后在配置文件中添加（或修改）主机条目，如下所示到文件：

   ```
   Host name-of-ssh-host-here
       User your-user-name-on-host
       HostName host-fqdn-or-ip-goes-here
       IdentityFile ~/.ssh/exported-keyfile-from-putty
   ```

从终端连接到远程主机

一旦配置了主机，就可以通过传递远程URI从终端直接连接到它。

例如，要连接 `remote_server` 并打开 `/code/my_project` 文件夹，请运行：

```
code --folder-uri "vscode-remote://ssh-remote+remote_server/code/my_project"
```

您也可以使用 `--file-uri` 开关打开特定文件。

解决挂起或连接失败的问题

如果您在尝试连接（并可能超时）时遇到VS Code挂起的问题，则可以采取一些措施来尝试解决此问题。

查看VS Code是否正在等待提示

在VS Code中启用 `remote.SSH.showLoginTerminal` 设置 (/docs/getstarted/settings)，然后重试。如果系统提示您输入密码或令牌，请参阅启用备用SSH身份验证方法以获取有关减少提示频率的详细信息。

如果仍然遇到问题，请在中设置以下属性， `settings.json` 然后重试：

```
"remote.SSH.showLoginTerminal": true,
"remote.SSH.useLocalServer": false
```

解决某些Windows OpenSSH服务器版本的错误

由于Windows的某些版本的OpenSSH服务器存在错误，因此默认检查（以确定主机是否正在运行Windows）可能无法正常工作。Windows 1909及更低版本附带的OpenSSH服务器不会发生这种情况。

幸运的是，您可以通过将以下内容添加到 `settings.json` ： 通过告诉VS Code SSH主机是否正在运行Windows来解决此问题：

```
"remote.SSH.useLocalServer": false
```

您还可以使用以下属性强制VS Code将特定主机标识为Windows：

```
"remote.SSH.remotePlatform": {
    "host-in-ssh-config-or-fqdn": "windows"
}
```

修复程序已合并，因此应在大于8.1.0.0的服务器版本中解决此问题。

在远程主机上启用TCP转发

远程-SSH扩展利用SSH隧道来促进与主机的通信。在某些情况下，这可能在您的SSH服务器上被禁用。要查看是否存在此问题，请在输出窗口中打开“ **远程-SSH**”类别，然后检查以下消息：

```
open failed: administratively prohibited: open failed
```

如果确实看到该消息，请按照以下步骤更新SSH服务器的sshd配置 (https://www.ssh.com/ssh/sshd_config/)：

1. 在**SSH主机**（非本地）上打开 `/etc/ssh/sshd_config` 或 `C:\ProgramData\ssh\sshd_config` 在其文本编辑器（如Vim，nano，Pico或记事本）中。
2. 添加设置 `AllowTcpForwarding yes` 。
3. 重启SSH服务器。 （在Ubuntu上，运行 `sudo systemctl restart sshd` 。在Windows上，在admin PowerShell运行中， `Restart-Service sshd` ）。
4. 重试。

在SSH配置文件中设置ProxyCommand参数

如果您位于代理后面，并且无法连接到SSH主机，则可能需要 `ProxyCommand` 在**本地** SSH配置文件中 (https://linux.die.net/man/5/ssh_config)为主机使用该参数。您可以阅读此SSH ProxyCommand文章 (https://www.cyberciti.biz/faq/linux-unix-ssh-proxycommand-passing-through-one-host-gateway-server/)以获取其用法示例。

确保远程机器可以访问互联网

远程计算机必须具有Internet访问权限，才能从市场上下载VS Code服务器和扩展。有关连接要求的详细信息 (/docs/remote/faq#_what-are-the-connectivity-requirements-for-vs-code-server)，请参见FAQ (/docs/remote/faq#_what-are-the-connectivity-requirements-for-vs-code-server)。

在远程主机上设置HTTP_PROXY / HTTPS_PROXY

如果远程主机位于代理之后，则可能需要在**SSH主机**上设置HTTP_PROXY或HTTPS_PROXY环境变量。打开 `~/.bashrc` 文件，添加以下内容（替换 `proxy.fqdn.or.ip:3128` 为适当的主机名/ IP和端口）：

```
export HTTP_PROXY=http://proxy.fqdn.or.ip:3128
export HTTPS_PROXY=$HTTP_PROXY

# Or if an authenticated proxy
export HTTP_PROXY=http://username:password@proxy.fqdn.or.ip:3128
export HTTPS_PROXY=$HTTP_PROXY
```

解决 `/tmp` 与安装 `noexec`

某些远程服务器设置为禁止从执行脚本 `/tmp` 。VS Code将其安装脚本写入系统temp目录，并尝试从该目录执行。您可以与系统管理员一起确定是否可以解决此问题。

检查安装过程中是否启动了其他外壳

一些用户 `.bash_profile` 从其**SSH主机**上的启动脚本或其他启动脚本启动其他Shell，因为他们想要使用与默认外壳不同的Shell。这可能会破坏VS Code的远程服务器安装脚本，因此不建议这样做。而是使用 `chsh` 更改远程计算机上的默认外壳程序。

连接到为每个连接动态分配机器的系统

某些系统会在每次建立SSH连接时将SSH连接动态路由到群集中的一个节点。这是VS Code的问题，因为它建立了两个连接来打开远程窗口：第一个连接用于安装或启动VS Code服务器（或查找已经运行的实例），第二个连接用于创建VS Code用于的SSH端口隧道。与服务器对话。如果VS Code在创建第二个连接时被路由到另一台计算机，则它将无法与VS Code服务器通信。

一种解决方法是使用 `ControlMaster` OpenSSH（仅适用于macOS / Linux客户端）中的选项，如启用替代SSH身份验证方法中所述，以便VS Code的两个连接将通过单个SSH连接复用到同一节点。

请与系统管理员联系以获取配置帮助

SSH是一种非常灵活的协议，并支持许多配置。如果您在登录终端或 `Remote-SSH` 输出窗口中看到其他错误，则可能是由于缺少设置。

请与系统管理员联系，以获取有关SSH主机和客户端所需设置的信息。可以将用于连接到SSH主机的特定命令行参数添加到SSH配置文件中 (https://linux.die.net/man/5/ssh_config)。

要访问您的配置文件，请运行**Remote-SSH：**在命令面板（ `F1` ）中**打开Configuration File...。**然后，您可以与管理员一起添加必要的设置。

启用备用SSH身份验证方法

如果要连接到SSH远程主机，并且是：

- 结合两因素验证，
- 使用密码验证，
- 当SSH代理未运行或不可访问时，将SSH密钥和密码一起使用，

... VS Code应该自动提示您输入所需的信息。如果没有看到提示，请在VS Code中启用 `remote.SSH.showLoginTerminal` 设置 (/docs/getstarted/settings)。每当VS Code运行SSH命令时，此设置都会显示终端。然后，当终端出现时，您可以输入验证码，密码或密码。

如果仍然遇到问题，则可能需要输入以下属性 `settings.json` 并重试：

```
"remote.SSH.showLoginTerminal": true,
"remote.SSH.useLocalServer": false
```

如果您使用的是macOS和Linux，并且希望减少输入密码或令牌的频率，则可以 `ControlMaster` 在**本地计算机**上启用该功能，以便OpenSSH通过单个连接运行多个SSH会话。

要启用 `ControlMaster`：

1. 将这样的条目添加到您的SSH配置文件中：

   ```
   Host *
       ControlMaster auto
       ControlPath  ~/.ssh/sockets/%r@%h-%p
       ControlPersist  600
   ```

2. 然后运行 `mkdir -p ~/.ssh/sockets` 以创建套接字文件夹。

设置SSH代理

如果要使用带有密码短语的密钥连接到SSH主机，则应确保SSH代理 (https://www.ssh.com/ssh/agent)在**本地**运行。VS Code将自动将密钥添加到代理，因此您不必在每次打开远程VS Code窗口时都输入密码。

要验证代理是否正在运行并且可以从VS Code的环境访问，请 `ssh-add -l` 在本地VS Code窗口的终端中运行。您应该在代理中看到密钥列表（或一条消息，它没有密钥）。如果代理未运行，请按照以下说明启动它。启动代理后，请确保重新启动VS Code。

视窗：

要在Windows上自动启用SSH代理，请启动**本地Administrator PowerShell**并运行以下命令：

```
# Make sure you're running as an Administrator
Set-Service ssh-agent -StartupType Automatic
Start-Service ssh-agent
Get-Service ssh-agent
```

现在，代理将在登录时自动启动。

Linux：

要在后台启动SSH代理，请运行：

```
eval "$(ssh-agent -s)"
```

要在登录时自动启动SSH代理，请将这些行添加到您的 `~/.bash_profile`：

```
if [ -z "$SSH_AUTH_SOCK" ]; then
   # Check for a currently running instance of the agent
   RUNNING_AGENT="`ps -ax | grep 'ssh-agent -s' | grep -v grep | wc -l | tr -d '[:space:]'`"
   if [ "$RUNNING_AGENT" = "0" ]; then
        # Launch a new instance of the agent
        ssh-agent -s &> .ssh/ssh-agent
   fi
   eval `cat .ssh/ssh-agent`
fi
```

苹果系统：

默认情况下，该代理应在macOS上运行。

修复SSH文件权限错误

SSH可能严格限制文件权限，如果设置不正确，您可能会看到诸如"警告：未保护的私有密钥文件！"之类的错误。有几种更新文件权限以解决此问题的方法，下面各节中将进行介绍。

本地SSH文件和文件夹权限

macOS / Linux：

在本地计算机上，确保设置了以下权限：

文件夹/文件	权限
`.ssh` 在您的用户文件夹中	`chmod 700 ~/.ssh`
`.ssh/config` 在您的用户文件夹中	`chmod 600 ~/.ssh/config`
`.ssh/id_rsa.pub` 在您的用户文件夹中	`chmod 600 ~/.ssh/id_rsa.pub`
任何其他密钥文件	`chmod 600 /path/to/key/file`

视窗：

特定的预期权限可能会有所不同，具体取决于您所使用的SSH实现。我们强烈建议使用开箱即用的Windows 10 OpenSSH Client (https://docs.microsoft.com/windows-server/administration/openssh/openssh_overview)。

在这种情况下，请确保 `.ssh` SSH主机上远程用户的文件夹中的所有文件均归您所有，并且没有其他用户有权访问它。有关详细信息，请参见Windows OpenSSH Wiki (https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH)。

对于所有其他客户，请参阅客户的文档以了解实施的期望。

服务器SSH文件和文件夹权限

macOS / Linux：

在要连接的远程计算机上，确保设置了以下权限：

文件夹/文件	Linux / macOS权限
`.ssh` 在服务器上的用户文件夹中	`chmod 700 ~/.ssh`
`.ssh/authorized_keys` 在服务器上的用户文件夹中	`chmod 600 ~/.ssh/authorized_keys`

请注意，当前仅支持Linux主机，这就是为什么省略了macOS和Windows 10权限的原因。

视窗：

有关为Windows OpenSSH服务器设置适当的文件权限的详细信息，请参见Windows OpenSSH Wiki (https://github.com/PowerShell/Win32-OpenSSH/wiki/Security-protection-of-various-files-in-Win32-OpenSSH)。

安装受支持的SSH客户端

操作系统	使用说明
Windows 10 1803 + /服务器2016/2019 1803+	安装Windows OpenSSH客户端 (https://docs.microsoft.com/windows-server/administration/openssh/openssh_install_firstuse)。
早期的Windows	安装Windows版Git (https://git-scm.com/download/win)。
苹果系统	预先安装。
Debian / Ubuntu	跑 `sudo apt-get install openssh-client`
RHEL / Fedora / CentOS	跑 `sudo yum install openssh-clients`

VS Code将 `ssh` 在PATH中查找命令。如果失败，它将在Windows上尝试查找 `ssh.exe` 默认的Windows Git安装路径。您还可以通过将该 `remote.SSH.path` 属性添加到来明确告诉VS Code SSH客户端的位置 `settings.json` 。

安装受支持的SSH服务器

操作系统	使用说明	细节
Debian 8以上版本/ Ubuntu 16.04以上版本	跑 `sudo apt-get install openssh-server`	有关详细信息，请参见Ubuntu SSH (https://help.ubuntu.com/community/SSH?action=show)文档。
RHEL / CentOS 7+	跑 `sudo yum install openssh-server && sudo systemctl start sshd.service && sudo systemctl enable sshd.service`	有关详细信息，请参见RedHat SSH (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/ch-openssh)文档。
SuSE 12+ / openSUSE 42.3+	在Yast中，转到"服务管理器"，在列表中选择" sshd"，然后单击"**启用**"。接下来转到防火墙，选择**永久**配置，然后在服务下检查**sshd**。	有关详细信息，请参见SuSE SSH (https://en.opensuse.org/OpenSSH)文档。
Windows 10 1803 + /服务器2016/2019 1803+	安装Windows OpenSSH服务器 (https://docs.microsoft.com/windows-server/administration/openssh/openssh_install_firstuse)。	
macOS 10.14+ （Mojave）	启用远程登录 (https://support.apple.com/guide/mac-help/allow-a-remote-computer-to-access-your-mac-mchlp1066/mac)。	

在SSH主机上执行Git推送或同步时解决挂起

如果您使用SSH克隆Git存储库，并且您的SSH密钥具有密码短语，则在远程运行时VS Code的提取和同步功能可能会挂起。

使用不带密码短语的SSH密钥，使用HTTPS克隆或从命令行运行 `git push` 以解决此问题。

使用SSHFS访问远程主机上的文件

SSHFS (https://en.wikipedia.org/wiki/SSHFS)是从SFTP建立的安全的远程文件系统访问协议。它提供了优于CIFS / Samba共享之类的优势，因为所需的全部是对计算机的SSH访问。

> **注意：**出于性能原因，SSHFS最适合用于单个文件编辑以及上载/下载内容。如果您需要使用一次可批量读取/写入多个文件的应用程序（例如本地源代码控制工具），则rsync是一个更好的选择。

macOS / Linux：

在Linux上，您可以使用发行版的软件包管理器来安装SSHFS。对于Debian / Ubuntu： `sudo apt-get install sshfs`

> **注意：** WSL 1不支持FUSE或SSHFS，因此当前Windows的说明有所不同。**WSL 2确实包含FUSE和SSHFS支持**，因此它将很快改变。

在macOS上，您可以使用Homebrew (https://brew.sh/)安装SSHFS： `brew install sshfs` 此外，如果您不想使用命令行安装远程文件系统，则还可以安装SSHFS GUI (https://github.com/dstuecken/sshfs-gui)。

要使用命令行，请从本地终端运行以下命令（替换 `user@hostname` 为远程用户和主机名/ IP）：

```
export USER_AT_HOST=user@hostname
# Make the directory where the remote filesystem will be mounted
mkdir -p "$HOME/sshfs/$USER_AT_HOST"
# Mount the remote filesystem
sshfs "$USER_AT_HOST:" "$HOME/sshfs/$USER_AT_HOST" -ovolname="$USER_AT_HOST" -p 22  \
    -o workaround=nonodelay -o transform_symlinks -o idmap=user  -C
```

这将使远程计算机上的主文件夹位于下方 `~/sshfs` 。完成后，可以使用操作系统的Finder /文件浏览器或使用命令行来卸载它：

```
umount "$HOME/sshfs/$USER_AT_HOST"
```

视窗：

跟着这些步骤：

1. 在Linux上，将 `.gitattributes` 文件添加到项目中以**强制** Linux和Windows之间的**行尾一致**，以避免由于两个操作系统之间的CRLF / LF差异而导致意外问题。有关详细信息，请参见解决Git行结束问题。
2. 接下来，使用Chocolatey (https://chocolatey.org/)安装SSHFS-Win (https://github.com/billziss-gh/sshfs-win)： (https://chocolatey.org/) `choco install sshfs`
3. 安装适用于Windows的SSHFS后，可以将File Explorer的**Map Network Drive ...**选项与path一起使用 `\\sshfs\user@hostname` ，其中path `user@hostname` 是您的远程用户和主机名/ IP。您可以使用命令提示符对此编写脚本，如下所示： `net use /PERSISTENT:NO X: \\sshfs\user@hostname`
4. 完成后，通过右键单击文件资源管理器中的驱动器并选择**断开连接**来**断开连接**。

使用rsync维护源代码的本地副本

到另一种使用SSHFS访问远程文件是使用 `rsync` (https://rsync.samba.org/)远程主机上的文件夹中的所有内容复制到本地机器。该 `rsync` 命令将确定每次运行时都需要更新哪些文件，这比使用诸如 `scp` 或那样更有效，更方便 `sftp` 。如果您确实需要使用多文件或性能密集的本地工具，则这是主要要考虑的事项。

该 `rsync` 命令在macOS上开箱即用，并且可以使用Linux软件包管理器进行安装（例如 `sudo apt-get install rsync` 在Debian / Ubuntu上）。对于Windows，您需要使用WSL (https://docs.microsoft.com/windows/wsl/install-win10)或Cygwin (https://www.cygwin.com/)来访问命令。

要使用该命令，请导航至要存储同步内容的文件夹，然后运行以下命令，将其替换 `user@hostname` 为远程用户和主机名/ IP以及 `/remote/source/code/path` 远程源代码位置。

在macOS，Linux或WSL内部：

```
rsync -rlptzv --progress --delete --exclude=.git "user@hostname:/remote/source/code/path" .
```

或**在Windows上**使用**PowerShell中的WSL**：

```
wsl rsync -rlptzv --progress --delete --exclude=.git "user@hostname:/remote/source/code/path" "`$(wslpath -a '$PWD')"
```

您每次想获取文件的最新副本时都可以重新运行此命令，并且仅更新将被传输。 `.git` 出于性能原因，该文件夹被有意排除在外，因此您可以使用本地Git工具，而不必担心远程主机上的状态。

要推送内容，请反转命令中的源参数和目标参数。但是，**在Windows上**，应在执行此操作之前将 `.gitattributes` 文件添加到项目中，以**强制行尾一致。**有关详细信息，请参见解决Git行结束问题。

```
rsync -rlptzv --progress --delete --exclude=.git . "user@hostname:/remote/source/code/path"
```

清理远程上的VS Code服务器

SSH的扩展，用于远程机器清理VS代码服务器提供了一个命令，**远程SSH：卸载VS代码服务器从主机...**。该命令有两个作用：杀死所有正在运行的VS Code Server进程，并删除服务器的安装文件夹。

如果要手动运行这些步骤，或者该命令对您不起作用，则可以运行如下脚本：

```
kill -9 `ps ax | grep "remoteExtensionHostAgent.js" | grep -v grep | awk '{print $1}'`
kill -9 `ps ax | grep "watcherService" | grep -v grep | awk '{print $1}'`
rm -rf ~/.vscode-server # Or ~/.vscode-server-insiders
```

VS Code Server以前安装在下面， `~/.vscode-remote` 因此您也可以检查该位置。

容器提示

本节包含一些提示和技巧，这些提示和技巧可用于在不同环境中启动和运行Remote-Containers扩展。

如果您遇到Docker问题，或者不想在本地运行Docker，则可能要尝试预览Visual Studio Codespaces受管基于云的环境 (https://aka.ms/vso-docs/vscode)。随着时间的流逝，该服务将支持越来越多的 `devcontainer.json` 属性，并且除了VS Code外，您还可以使用其基于浏览器的编辑器。

Windows版Docker桌面技巧

适用 (https://www.docker.com/products/docker-desktop)于Windows的Docker桌面 (https://www.docker.com/products/docker-desktop)在大多数设置中都能很好地工作，但是有一些"陷阱"可能会引起问题。以下是避免它们的一些技巧：

1. **考虑在Windows 10（2004+）上使用新的Docker WSL2后端。**如果您使用的是Docker Desktop的WSL2后端 (https://aka.ms/vscode-remote/containers/docker-wsl)，则可以在WSL内部以及本地打开文件夹。Windows和WSL内部也共享容器，并且此新引擎不易受到文件共享问题的影响。有关详细信息，请参见快速 (/docs/remote/containers#_open-a-wsl2-folder-in-a-container-on-windows)入门。
2. **退出" Windows上的Linux容器（LCOW）"模式。**默认情况下禁用，但最新版本的Docker支持Windows上的Linux容器（LCOW） (https://docs.microsoft.com/virtualization/windowscontainers/deploy-containers/linux-containers)，可让您同时使用Windows和Linux容器。但是，这是一项新功能，因此您可能会遇到问题，并且"远程-容器"扩展当前仅支持Linux容器。您可以随时通过右键单击Docker任务栏项目并从上下文菜单中选择" **切换到Linux容器...**"来退出LCOW模式。
3. **确保您的防火墙允许Docker设置共享驱动器。**Docker只需要在两个计算机本地IP之间进行连接，但是某些防火墙软件可能仍会阻止任何驱动器共享或所需的端口。有关解决此问题的后续步骤，请参见此Docker KB文章 (https://success.docker.com/article/error-a-firewall-is-blocking-file-sharing-between-windows-and-the-containers)。

这是一些适用于Windows的较旧版本Docker的技巧，但现在应该解决。如果由于可能的回归而遇到了暂存行为，则这些技巧过去已经解决了问题。

1. **共享驱动器时，请使用AD域帐户或本地管理员帐户。不要使用AAD（基于电子邮件）帐户。**AAD（基于电子邮件）帐户具有众所周知的问题，如Docker 问题132 (https://github.com/docker/for-win/issues/132)和问题＃1352中所述 (https://github.com/docker/for-win/issues/1352)。如果必须使用AAD帐户，请在您的计算机上创建一个单独的本地管理员帐户，纯粹用于共享驱动器。按照此博客文章中 (https://blogs.msdn.microsoft.com/stevelasker/2016/06/14/configuring-docker-for-windows-volumes/)的步骤进行 (https://blogs.msdn.microsoft.com/stevelasker/2016/06/14/configuring-docker-for-windows-volumes/)设置。
2. **坚持使用字母数字密码，以避免驱动器共享问题。**当要求您在Windows上共享驱动器时，系统将提示您输入具有计算机管理员权限的帐户的用户名和密码。如果警告您输入错误的用户名或密码，则可能是由于密码中的特殊字符所致。例如！，［和］已知会引起问题。将密码更改为字母数字字符即可解决。有关详细信息，请参见有关Docker卷安装问题的问题 (https://github.com/moby/moby/issues/23992#issuecomment-234979036)。
3. **使用您的Docker ID登录Docker（而不是您的电子邮件）。**Docker CLI仅支持使用您的Docker ID，因此使用电子邮件会引起问题。有关详细信息，请参见Docker 问题＃935 (https://github.com/docker/hub-feedback/issues/935#issuecomment-300361781)。

如果仍然遇到问题，请参阅《Docker Windows版桌面疑难解答指南》 (https://docs.docker.com/docker-for-windows/troubleshoot/#volumes)。

在Docker Desktop中启用文件共享

如果您的代码位于与Docker共享的文件夹或驱动器中，则VS Code Remote-Containers (https://aka.ms/vscode-remote/download/containers)扩展只能将您的源代码自动安装到容器中。如果从非共享位置打开开发容器，则容器将成功启动，但工作区将为空。

请注意，Docker Desktop的WSL2引擎 (https://aka.ms/vscode-remote/containers/docker-wsl)**不需要**执行此步骤。 (https://aka.ms/vscode-remote/containers/docker-wsl)

要更改Docker的驱动器和文件夹共享设置：

视窗：

1. 右键单击Docker任务栏项目，然后选择**设置**。
2. 转到**资源**>**文件共享**，然后检查源代码所在的驱动器。
3. 如果您看到有关本地防火墙阻止共享操作的消息，请参阅此Docker KB文章 (https://success.docker.com/article/error-a-firewall-is-blocking-file-sharing-between-windows-and-the-containers)以获取后续步骤。

苹果系统：

1. 单击Docker菜单栏项，然后选择**首选项**。
2. 转到**资源**>**文件共享**。确认包含源代码的文件夹在列出的共享文件夹之一下面。

解决容器中的Git行结束问题（导致许多修改的文件）

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，除了行尾外没有其他区别。为防止这种情况发生，您可以 `.gitattributes` 在Windows端使用文件或全局禁用行尾转换。

通常 `.gitattributes` ，在存储库中添加或修改 文件是解决此问题的最可靠方法。将此文件提交到源代码管理将帮助其他人，并允许您根据存储库的不同来更改行为。例如，将以下内容添加到 `.gitattributes` 文件库的根目录将强制所有内容为LF，但需要CRLF的Windows批处理文件除外：

```
* text=auto eol=lf
*.{cmd,[cC][mM][dD]} text eol=crlf
*.{bat,[bB][aA][tT]} text eol=crlf
```

请注意，这在**Git v2.10 +中有效**，因此，如果遇到问题，请确保已安装了最新的Git客户端。您可以将需要CRLF的其他文件类型添加到该文件中。

如果您仍然希望始终上传Unix样式的行尾（LF），则可以使用该 `input` 选项。

```
git config --global core.autocrlf input
```

如果您希望完全禁用行尾转换，请运行以下命令：

```
git config --global core.autocrlf false
```

最后，您可能需要再次克隆存储库以使这些设置生效。

使用Docker Compose时避免在容器中设置Git

有关解决此问题的信息，请参阅主容器文章中的与容器共享Git凭据 (/docs/remote/containers#_sharing-git-credentials-with-your-container)。

从容器执行Git推送或同步时解决挂起

如果您使用SSH克隆Git存储库，并且您的SSH密钥具有密码短语，则在远程运行时VS Code的提取和同步功能可能会挂起。

使用不带密码短语的SSH密钥，使用HTTPS克隆或从命令行运行 `git push` 以解决此问题。

解决有关缺少Linux依赖项的错误

一些扩展依赖于某些Docker映像中找不到的库。有关解决此问题的一些选项，请参见“ 容器” (/docs/remote/containers#_installing-additional-software)文章。

在Docker Desktop中加速容器

默认情况下，Docker Desktop仅为容器提供机器容量的一小部分。在大多数情况下，这就足够了，但是如果您要执行的操作需要更多容量，则可以增加内存，CPU或磁盘的使用量。

首先，尝试停止任何 (/docs/remote/containers#_managing-containers)不再使用的正在运行的容器 (/docs/remote/containers#_managing-containers)。

如果这不能解决您的问题，则可能需要查看CPU使用率实际上是否是问题所在，或者是否还有其他情况发生。一种简单的检查方法是安装资源监视器扩展 (https://marketplace.visualstudio.com/items?itemName=mutantdino.resourcemonitor&ssr=false#overview)。当安装在容器中时，它在状态栏中提供有关容器容量的信息。

如果您希望始终安装此扩展程序，请将其添加到您的 `settings.json`：

```
"remote.containers.defaultExtensions": [
    "mutantdino.resourcemonitor"
]
```

如果确定需要为容器提供更多机器容量，请执行以下步骤：

1. 右键单击Docker任务栏项目，然后选择**设置 / 首选项**。
2. 转到**高级**以增加CPU，内存或交换。
3. 在macOS上，转到“ **磁盘**”以增加Docker允许在您的计算机上使用的磁盘数量。在Windows上，它位于其他设置下的“高级”下。

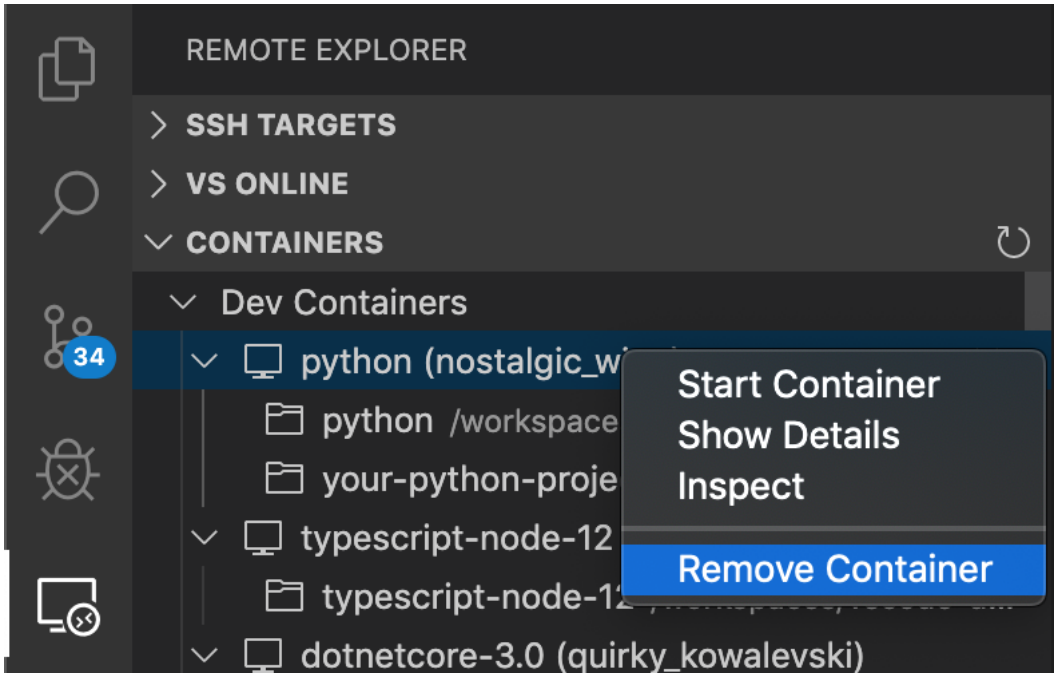
最后，如果您的容器正在**执行磁盘密集型**操作，或者您只是在寻找更快的响应时间，请参阅“ 提高容器磁盘性能” (/docs/remote/containers-advanced#_improving-container-disk-performance)以获取提示。VS Code的默认值为方便起见和通用支持而进行了优化，但是可以进行优化。

清理未使用的容器和图像

如果您从Docker报告中看到一个错误，指出您的磁盘空间不足，通常可以通过清除未使用的容器和映像来解决此问题。有几种方法可以做到这一点：

选项1：使用远程资源管理器

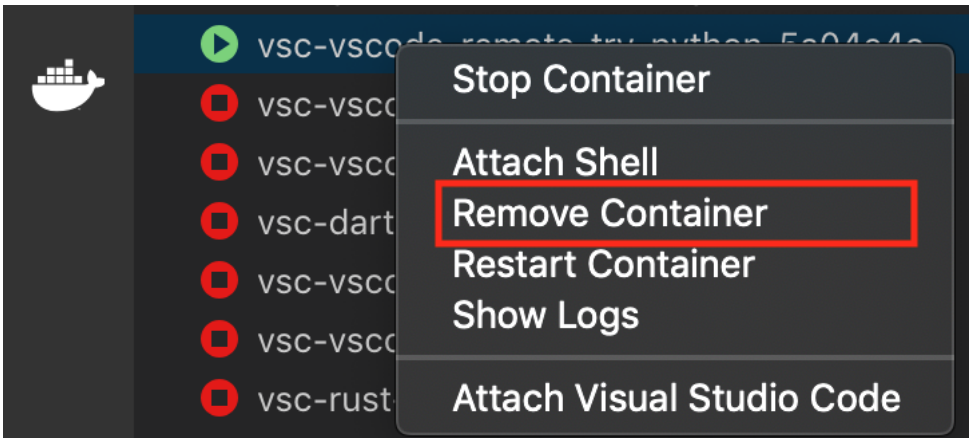
您可以通过选择**Remote Explorer**删除容器，右键单击要**删除的容器**，然后选择**Remove Container**。



但是，这不会清除您可能已下载的任何图像，这可能会使系统混乱。

选项2：使用Docker扩展

1. 在VS Code中打开一个**本地**窗口（**File> New Window**）。
2. 从扩展视图安装Docker扩展 (https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker)（如果尚不存在）。
3. 然后，您可以转到Docker视图并展开**Containers**或**Images**节点，右键单击并选择**Remove Container / Image**。



选项3：使用Docker CLI选择要删除的容器

1. 打开**本地**终端/命令提示符（或使用VS Code中的本地窗口）。
2. 键入 `docker ps -a` 以查看所有容器的列表。
3. `docker rm <Container ID>` 从此列表中键入以删除容器。
4. 键入 `docker image prune` 以删除所有未使用的图像。

如果 `docker ps` 没有提供足够的信息来标识要删除的容器，则以下命令将列出VS Code管理的所有开发容器以及用于生成它们的文件夹。

```
docker ps -a --filter="label=vsch.quality" --format "table {{.ID}}\t{{.Status}}\t{{.Image}}\tvscode-{{.Label \"vsch.quality\"}}\t{{.Label \"vsch.local.folder\"}}"
```

选项4：使用Docker Compose

1. 打开**本地**终端/命令提示符（或使用VS Code中的本地窗口）。
2. 转到包含 `docker-compose.yml` 文件的目录。
3. 键入 `docker-compose down` 以停止并删除容器。如果您有多个Docker Compose文件，则可以使用 `-f` 参数指定其他Docker Compose文件。

选项4：删除所有未运行的容器和图像：

1. 打开**本地**终端/命令提示符（或使用VS Code中的本地窗口）。
2. 输入 `docker system prune --all` 。

使用Debian 8解决映像的Dockerfile构建失败

当构建使用基于Debian 8 / Jessie的图像的容器（例如 `node:8` 图像的旧版本）时，您可能会遇到以下错误：

```
...
W: Failed to fetch http://deb.debian.org/debian/dists/jessie-updates/InRelease  Unable to find expected entry 'main/binary-amd64/Packages' in Release file (Wrong sources.list entry or malformed file)
E: Some index files failed to download. They have been ignored, or old ones used instead.
...
```

这是由"归档"的Debian 8引起的一个众所周知的问题 (https://github.com/debuerreotype/docker-debian-artifacts/issues/66)。映像的最新版本通常可以通过升级到Debian 9 / Stretch解决此问题。

有两种方法可以解决此错误：

- **选项1**：删除所有依赖于映像的容器，删除映像，然后再次尝试构建。这应该下载不受该问题影响的更新的映像。有关详细信息，请参见清理未使用的容器和图像。
- **选项2**：如果您不想删除容器或图像，请在任何 `apt` or `apt-get` 命令之前将此行添加到Dockerfile中。它为Jessie添加了所需的源列表：

  ```
  # Add archived sources to source list if base image uses Debian 8 / Jessie
  RUN cat /etc/*-release | grep -q jessie && printf "deb http://archive.debian.org/debian/ jessie main\ndeb-src http://archive.debian.org/debian/ jessie main\ndeb http://security.debian.org jessie/updates main\ndeb-src http://security.debian.org jessie/updates main" > /etc/apt/sources.list
  ```

解决使用电子邮件时的Docker Hub登录错误

Docker CLI仅支持使用您的Docker ID，因此使用电子邮件登录可能会导致问题。有关详细信息，请参见Docker 问题＃935 (https://github.com/docker/hub-feedback/issues/935#issuecomment-300361781)。

解决方法是，使用您的Docker ID登录Docker，而不是电子邮件。

macOS上Hyperkit的高CPU使用率

Mac的Docker (https://github.com/docker/for-mac/issues/1759)存在一个已知问题， (https://github.com/docker/for-mac/issues/1759)它可能导致CPU高峰运行。特别是，在观看文件和构建文件时，CPU使用率很高。如果 `com.docker.hyperkit` 在活动监视器中看到的CPU使用率很高，而在您的开发容器中却很少运行，则可能是此问题。请关注Docker问题 (https://github.com/docker/for-mac/issues/1759)以获取更新和修复。

使用SSH隧道连接到远程Docker主机

将在远程泊坞窗机或SSH主机容器内开发 (/docs/remote/containers-advanced#_developing-inside-a-container-on-a-remote-docker-host)与远程主机码头工人工作时，文章介绍如何设置VS代码。这通常很简单，就像将 `docker.host` 属性in `settings.json` 或 `DOCKER_HOST` 环境变量设置为 `ssh://` 或 `tcp://` URI 一样简单。

但是，由于SSH配置复杂性或其他限制，您可能会遇到这种情况在您的环境中不起作用的情况。在这种情况下，可以将SSH隧道用作备用。

使用SSH隧道作为后备选项

您可以设置SSH隧道，并将Docker套接字从远程主机转发到本地计算机。

跟着这些步骤：

1. 安装与OpenSSH兼容的SSH客户端 (/docs/remote/troubleshooting#_installing-a-supported-ssh-client)。
2. 如下更新 `docker.host` 用户或工作空间中的属性 `settings.json`：

```
"docker.host":"tcp://localhost:23750"
```

3. 从本地终端/ PowerShell运行以下命令（替换 `user@hostname` 为服务器的远程用户和主机名/ IP）：

```
ssh -NL localhost:23750:/var/run/docker.sock user@hostname
```

VS Code现在将能够附加到 (/docs/remote/containers#_attaching-to-running-containers)远程主机上任何正在运行的容器 (/docs/remote/containers#_attaching-to-running-containers)。您还可以使用专用的本地 `devcontainer.json` 文件来创建/连接到远程dev容器 (/docs/remote/containers-advanced#_converting-an-existing-or-predefined-devcontainerjson)。

完成后，在终端/ PowerShell中按 Ctrl + C 关闭隧道。

> **注意：**如果 `ssh` 命令失败，则可能需要 `AllowStreamLocalForwarding` 在SSH主机上。
>
> 1. 打开 `/etc/ssh/sshd_config` 在上一个编辑器（如Vim的，纳米，或微微）**SSH主机**（不在本地）。
> 2. 添加设置 `AllowStreamLocalForwarding yes` 。
> 3. 重新启动SSH服务器（在Ubuntu上，运行 `sudo systemctl restart sshd` ）。
> 4. 重试。

先进的容器配置技巧

请参阅“ 高级容器配置” (/docs/remote/containers-advanced)文章，以获取有关以下主题的信息：

- 添加环境变量 (/docs/remote/containers-advanced#_adding-environment-variables)
- 添加另一个卷挂载 (/docs/remote/containers-advanced#_adding-another-volume-mount)
- 更改或删除默认源代码安装 (/docs/remote/containers-advanced#_changing-the-default-source-code-mount)
- 向您的开发容器添加非root用户 (/docs/remote/containers-advanced#_adding-a-nonroot-user-to-your-dev-container)
- 改善容器磁盘性能 (/docs/remote/containers-advanced#_improving-container-disk-performance)
- 避免在容器重建时扩展安装 (/docs/remote/containers-advanced#_avoiding-extension-reinstalls-on-container-rebuild)
- 设置Docker Compose的项目名称 (/docs/remote/containers-advanced#_setting-the-project-name-for-docker-compose)
- 从容器内部使用Docker或Kubernetes (/docs/remote/containers-advanced#_using-docker-or-kubernetes-from-a-container)
- 一次连接到多个容器 (/docs/remote/containers-advanced#_connecting-to-multiple-containers-at-once)
- 在远程Docker Machine或SSH主机上的容器内部进行开发 (/docs/remote/containers-advanced#_developing-inside-a-container-on-a-remote-docker-host)
- 减少Dockerfile构建警告 (/docs/remote/containers-advanced#_reducing-dockerfile-build-warnings)

WSL技巧

首次启动：VS Code Server的先决条件

某些WSL Linux发行版缺少VS Code服务器启动所需的库。您可以使用其程序包管理器将其他库添加到Linux发行版中。

DEBIAN和UBUNTU

打开Debian或Ubuntu WSL shell进行添加 `wget` 和 `ca-certificates`：

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

高山的

以root（ `wsl -d Alpine -u root` ）的身份打开Alpine WSL shell 以添加 `libstdc++`：

```
apk update && apk add libstdc++
```

在Windows 10 April 2018更新（内部版本1803）和更早版本上， `/bin/bash` 是必需的：

```
apk update && apk add bash
```

选择Remote-WSL使用的分发

远程WSL：“新窗口”将打开已注册为默认值的WSL发行版。

要打开非默认发行版，请 `code .` 从发行版的WSL Shell 运行以使用或使用**Remote-WSL：使用Distro的新窗口。**

对于WSL版本早于Windows 10（2019年5月更新）（版本1903），WSL命令只能使用**默认发行版。**因此，如果您同意更改默认发行版，Remote-WSL可能会提示您。

您始终可以使用wslconfig.exe (https://docs.microsoft.com/windows/wsl/wsl-config)更改默认值。

例如：

```
wslconfig /setdefault Ubuntu
```

您可以通过运行以下命令查看已安装的发行版：

```
wslconfig /l
```

配置服务器启动的环境

当远程WSL扩展在WSL中启动VS Code服务器时，它不运行任何外壳程序配置脚本。这样做是为了避免自定义配置脚本会阻止启动。

如果需要配置启动环境，则可以使用此处 (/docs/remote/wsl#_advanced-environment-setup-script)所述的环境设置脚本。

为远程扩展主机配置环境

远程扩展主机和终端的环境基于默认Shell的配置脚本。为了评估远程扩展主机进程的环境变量，服务器将创建默认外壳程序的实例作为**交互式登录外壳程序。**它从中探查环境变量，并将其用作远程扩展主机进程的初始环境。因此，环境变量的值取决于将哪个Shell配置为默认外壳以及该外壳的配置脚本的内容。

有关每个shell的配置脚本的概述，请参见Unix shell初始化 (https://github.com/rbenv/rbenv/wiki/unix-shell-initialization)。大多数WSL发行版已 `/bin/bash` 配置为默认Shell。 `/bin/bash` 将寻找启动文件在 `/etc/profile` 第一和任何启动文件下 `~/.bash_profile` ， `~/.bash_login` ， `~/.profile` 。

要更改WSL发行版的默认外壳，请遵循此博客文章 (https://medium.com/@vinhp/set-and-use-zsh-as-default-shell-in-wsl-on-windows-10-the-right-way-4f30ed9592dc)的说明。

修复code命令无法正常工作的问题

如果 `code` 因为 `code` 找不到而无法从Window上的WSL终端键入内容，则可能是WSL中PATH中缺少某些关键位置。

通过打开WSL终端并键入进行检查 `echo $PATH` 。您应该看到列出了VS Code安装路径。默认情况下，这将是：

```
/mnt/c/Users/Your Username/AppData/Local/Programs/Microsoft VS Code/bin
```

但是，如果您使用了**System Installer**，则安装路径为：

```
/mnt/c/Program Files/Microsoft VS Code/bin
```

...要么...

```
/mnt/c/Program Files (x86)/Microsoft VS Code/bin
```

WSL的一个功能是从Windows中的PATH变量继承路径。要更改Windows PATH变量，**请**从Windows的“开始”菜单中使用“ **为您的帐户编辑环境变量**”命令。

如果您禁用了路径共享功能，请编辑 `.bashrc` ，添加以下内容，然后启动新的终端：

```
WINDOWS_USERNAME="Your Windows Alias"

export PATH="$PATH:/mnt/c/Windows/System32:/mnt/c/Users/${WINDOWS_USERNAME}/AppData/Local/Programs/Microsoft VS Code/bin"
# or...
# export PATH="$PATH:/mnt/c/Program Files/Microsoft VS Code/bin"
# or...
# export PATH="$PATH:/mnt/c/Program Files (x86)/Microsoft VS Code/bin"
```

> **注意：**请确保在目录名称中加引号或转义空格字符。

我看到EACCESS：尝试重命名打开的工作区中的文件夹时，权限被拒绝的错误

这是由VS Code激活的文件监视程序导致的WSL文件系统实现（Microsoft / WSL # 3395 (https://github.com/Microsoft/WSL/issues/3395)和Microsoft / WSL # 1956 (https://github.com/Microsoft/WSL/issues/1956)）的已知问题。该问题仅在WSL 2中得以解决。

为避免此问题，请将其设置 `remote.WSL.fileWatcher.polling` 为true。但是，基于轮询会对大型工作区产生性能影响。

对于较大的工作空间，您可能需要增加轮询间隔 `remote.WSL.fileWatcher.pollingInterval` ，并控制使用监视的文件夹 `files.watcherExclude` 。

WSL 2 (https://docs.microsoft.com/windows/wsl/wsl2-index)没有该文件监视程序问题，并且不受新设置的影响。

解决WSL中的Git行结尾问题（导致许多修改的文件）

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，除了行尾外没有其他区别。为防止这种情况发生，您可以 `.gitattributes` 在Windows端使用文件或全局禁用行尾转换。

通常 `.gitattributes` ，在存储库中添加或修改 文件是解决此问题的最可靠方法。将此文件提交到源代码管理将帮助其他人，并允许您根据存储库的不同来更改行为。例如，将以下内容添加到 `.gitattributes` 文件库的根目录将强制所有内容为LF，但需要CRLF的Windows批处理文件除外：

```
* text=auto eol=lf
*.{cmd,[cC][mM][dD]} text eol=crlf
*.{bat,[bB][aA][tT]} text eol=crlf
```

请注意，这在**Git v2.10 +中有效**，因此，如果遇到问题，请确保已安装了最新的Git客户端。您可以将需要CRLF的其他文件类型添加到该文件中。

如果您仍然希望始终上传Unix样式的行尾（LF），则可以使用该 `input` 选项。

```
git config --global core.autocrlf input
```

如果您希望完全禁用行尾转换，请运行以下命令：

```
git config --global core.autocrlf false
```

最后，您可能需要再次克隆存储库以使这些设置生效。

在Windows和WSL之间共享Git凭据

如果您使用HTTPS克隆存储库并**在Windows中配置 (https://help.github.com/en/articles/caching-your-github-password-in-git)了凭据帮助 (https://help.github.com/en/articles/caching-your-github-password-in-git)程序**，则可以与WSL共享该**帮助 (https://help.github.com/en/articles/caching-your-github-password-in-git)程序**，以便您输入的密码在两侧均保持不变。（请注意，这不适用于使用SSH密钥。）

只需按照以下步骤操作：

1. 通过在**Windows命令提示符**或**PowerShell中**运行以下**命令，**在Windows上配置凭据管理器：

   ```
   git config --global credential.helper wincred
   ```

2. 将WSL配置为使用相同的凭据帮助器，但在**WSL终端中**运行以下命令：

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

现在，WSL可以使用在Windows上使用Git时输入的任何密码，反之亦然。

从WSL执行Git推送或同步时解决挂起

如果您使用SSH克隆Git存储库，并且您的SSH密钥具有密码短语，则在远程运行时VS Code的提取和同步功能可能会挂起。

使用不带密码短语的SSH密钥，使用HTTPS克隆或从命令行运行 `git push` 以解决此问题。

Visual Studio代码空间提示

有关与服务或扩展有关的提示和技巧，请参阅Visual Studio代码空间疑难解答文章 (https://aka.ms/vso-docs/troubleshooting)。

扩展技巧

尽管许多扩展程序可以在未修改的情况下正常运行，但仍有一些问题可能会阻止某些功能按预期运行。在某些情况下，您可以使用其他命令来解决此问题，而在其他情况下，则可能需要修改扩展名。本部分提供常见问题的快速参考以及解决这些问题的提示。您也可以参考支持 (/api/advanced-topics/remote-extensions)扩展开发 (/api/advanced-topics/remote-extensions)上的主要扩展文章，以获取有关修改扩展以支持远程扩展主机的深入指南。

解决有关缺少依赖项的错误

一些扩展依赖于某些WSL Linux发行版的基本安装中找不到的库。您可以使用其程序包管理器将其他库添加到Linux发行版中。对于基于Ubuntu和Debian的发行版，运行 `sudo apt-get install <package>` 以安装所需的库。检查您的扩展程序文档或错误消息中提到的运行时，以获取其他安装详细信息。

远程应用时，本地绝对路径设置失败

当您连接到远程端点时，将重新使用VS Code的本地用户设置。虽然这可以使您的用户体验保持一致，但是由于目标位置不同，因此可能需要更改本地计算机与每个主机/容器/ WSL之间的绝对路径设置。

解决方法：在连接到远程端点之后，可以通过运行“ **首选项**”：从命令面板（ F1 ）中**打开**“**远程设置**”命令，或者在“设置”编辑器中选择“ **远程**”选项卡来设置特定于端点的设置。每当您连接时，这些设置将覆盖您已有的所有本地设置。

需要在远程端点上安装本地VSIX

有时，在开发过程中或扩展作者要求您尝试修复时，您都希望在远程计算机上安装本地VSIX。

解决方法：一旦连接到SSH主机，容器或WSL，就可以像在本地一样安装VSIX。运行命令面板（ F1 ）中的**Extensions：从VSIX安装**...命令。您可能还需要添加以防止自动更新到最新的Marketplace版本。有关在远程环境中开发和测试扩展的更多信息，请参见支持远程开发 (/api/advanced-topics/remote-extensions)。 `"extensions.autoUpdate": false` `settings.json` (/api/advanced-topics/remote-extensions)

浏览器无法在本地打开

一些扩展使用外部节点模块或自定义代码来启动浏览器窗口。不幸的是，这可能会导致扩展程序以远程方式而不是本地方式启动浏览器。

解决方法：该扩展程序可以使用 `vscode.env.openExternal` API来解决此问题。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#opening-something-in-a-local-browser-or-application)。

剪贴板不起作用

一些扩展使用节点模块，例如 `clipboardy` 与剪贴板集成。不幸的是，这可能会导致扩展无法正确地与远程剪贴板集成在一起。

解决方法：该扩展程序可以切换到VS Code剪贴板API来解决问题。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#using-the-clipboard)。

无法从浏览器或应用程序访问本地Web服务器

在容器，SSH主机中或通过Visual Studio Codespaces内部工作时，浏览器连接到的端口可能会被阻止。

解决方案：扩展程序可以使用 `vscode.env.openExternal` 或 `vscode.env.asExternalUri` API（自动转发localhost端口）来解决此问题。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#opening-something-in-a-local-browser-or-application)。解决方法是使用" **转发端口**"命令手动执行此操作。

Webview内容不出现

如果扩展程序的Webview内容使用 `iframe` 来连接到本地Web服务器，则该Webview连接到的端口可能会被阻止。此外，如果扩展名硬编码 `vscode-resource://` URI而不是使用URI `asWebviewUri` ，则内容可能不会出现在"代码空间"浏览器编辑器中。

解决方法：该扩展程序可以使用 `webview.asWebviewUri` 来解决 `vscode-resource://` URI 问题。

如果端口被阻止，最好的方法是改为使用webview消息传递 (/api/extension-guides/webview#scripts-and-message-passing) API。解决方法是，`vscode.env.asExternalUri` 可以使用允许Web视图从VS Code连接到生成的localhost Web服务器。但是，MicrosoftDocs / vsonline # 11 (https://github.com/MicrosoftDocs/vsonline/issues/11)目前仅针对基于代码空间浏览器的编辑器阻止了此操作。有关解决方法的详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#workarounds-for-using-localhost-from-a-webview)。

本地主机端口被阻塞

如果您尝试从外部应用程序连接到本地主机端口，则该端口可能被阻止。

解决： VS Code 1.40引入了新的 `vscode.env.asExternalUri` API，用于扩展以编程方式转发任意端口。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#forwarding-localhost)。解决方法是，可以使用" **转发端口**"命令手动执行此操作。

Websocket在基于Codespaces基于浏览器的编辑器中的端口转发内容中不起作用

当前，Visual Studio Codespaces基于浏览器的编辑器中的转发机制仅支持http和https请求。Web套接字即使在转发的Web内容中使用或在JavaScript代码中使用也不起作用。这可能会影响用户应用程序和使用来自Web视图的WebSocket的扩展。

但是，VS Code的远程开发和Visual Studio Codespaces扩展本身没有此限制。

解决方法：在使用需要Web套接字而不是基于浏览器的编辑器时，请使用VS Code的Codespaces扩展。Codespaces团队正在研究此问题的解决方案。有关详细信息，请参见MicrosoftDocs / vsonline # 19 (https://github.com/MicrosoftDocs/vsonline/issues/19)。

存储扩展数据时出错

扩展程序可能会通过 `~/.config/Code` 在Linux上查找文件夹来尝试保留全局数据。此文件夹可能不存在，这可能导致扩展名抛出错误，例如 `ENOENT: no such file or directory, open '/root/.config/Code/User/filename-goes-here`。

解决方案：扩展程序可以使用 `context.globalStoragePath` 或 `context.storagePath` 属性来解决此问题。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#persisting-extension-data-or-state)。

每次连接到新端点时都无法登录/必须登录

需要登录的扩展程序可以使用自己的代码保留秘密。由于缺少依赖关系，此代码可能会失败。即使成功，密钥也将被远程存储，这意味着您必须登录每个新端点。

解决：扩展程序可以使用 `keytar` 节点模块来解决此问题。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#persisting-secrets)。

不兼容的扩展名阻止VS Code连接

如果在远程主机，容器或WSL中安装了不兼容的扩展程序，我们将看到VS Code Server由于不兼容而挂起或崩溃的实例。如果扩展立即激活，这可能会阻止您连接并能够卸载该扩展。

解决方法：请按照以下步骤手动删除远程扩展文件夹：

1. 对于容器，请确保您 `devcontainer.json` 不再包含对错误扩展名的引用。
2. 接下来，使用单独的终端/命令提示符连接到远程主机，容器或WSL。
   - 如果是SSH或WSL，请相应地连接到环境（运行 `ssh` 以连接到服务器或打开WSL终端）。
   - 如果使用容器，请通过调用 `docker ps -a` 并在列表中查找具有正确名称的图像来标识容器ID 。如果容器已停止，请运行 `docker run -it <id> /bin/sh` 。如果正在运行，请运行 `docker exec -it <id> /bin/sh` 。
3. 连接后，请运行 `rm -rf ~/.vscode-server/extensions` VS Code稳定版和/或 `rm -rf ~/.vscode-server-insiders/extensions` VS Code Insiders以删除所有扩展。

运送或获取预建本机模块的扩展失败

与VS Code扩展捆绑在一起（或动态获取）的本机模块必须使用Electron的 `electron-rebuild` (https://electronjs.org/docs/tutorial/using-native-node-modules)重新编译。但是，VS Code Server运行Node.js的标准（非电子）版本，当远程使用二进制文件时可能会导致二进制文件失败。

解决：需要修改扩展名才能解决此问题。他们将需要为VS Code随附的Node.js中的"模块"版本包括（或动态获取）两组二进制文件（电子和标准Node.js），然后检查 `context.executionContext === vscode.ExtensionExecutionContext.Remote` 其激活功能中是否设置了正确的二进制文件。有关详细信息，请参见扩展作者指南 (/api/advanced-topics/remote-extensions#using-native-node.js-modules)。