

# Contents

## WSL Documentation

### Overview

- What is WSL?

- Comparing WSL 2 and WSL 1

### Quickstart

- Install WSL & update to WSL 2

- Install on Windows Server

- Create a user account & password

### Tutorials

- Get started with VS Code

- Get started with Git

- Get started with databases

- Get started with Docker remote containers

- Set up GPU acceleration (NVIDIA CUDA/DirectML)

- Run Linux GUI apps

### How-to

- Interoperability commands

- Launch commands & configurations

- File permissions

- Set up WSL for your company

- Import any Linux distribution

- Build a custom distribution

- Mount a disk in WSL 2 (Preview)

- Command Line Reference

### Frequently Asked Questions

### WSL 2 FAQ

### Troubleshooting

### Release Notes

- General Release Notes



# Windows Subsystem for Linux Documentation

4/14/2021 • 2 minutes to read • [Edit Online](#)

The Windows Subsystem for Linux lets developers run a GNU/Linux environment -- including most command-line tools, utilities, and applications -- directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup.

[Install WSL](#)

## Learn more

- [What is the Windows Subsystem for Linux?](#)
- [What's new with WSL 2?](#)
- [Compare WSL 2 and WSL 1](#)
- [Read frequently asked questions](#)

## Get started

- [Install WSL1](#)
- [Check requirements for WSL2](#)
- [Update from WSL 1 to WSL 2](#)
- [Install Linux on Windows Server](#)
- [Create a user account and password for your new Linux distribution](#)

## Team blogs

- [Overview post with a collection of videos and blogs](#)
- [Command-Line blog](#) (Active)
- [Windows Subsystem for Linux Blog](#) (Historical)

## Posts and articles

- [Run Bash on Ubuntu on Windows](#)
- [Developers Can Run Bash And Usermode Ubuntu Linux Binaries On Windows 10](#)
- [Ubuntu on Windows – The Ubuntu Userspace for Windows Developers](#)

## Provide feedback

- [GitHub issue tracker](#)

# What is the Windows Subsystem for Linux?

4/14/2021 • 2 minutes to read • [Edit Online](#)

The Windows Subsystem for Linux lets developers run a GNU/Linux environment -- including most command-line tools, utilities, and applications -- directly on Windows, unmodified, without the overhead of a traditional virtual machine or dualboot setup.

You can:

- Choose your favorite GNU/Linux distributions [from the Microsoft Store](#).
- Run common command-line tools such as `grep`, `sed`, `awk`, or other ELF-64 binaries.
- Run Bash shell scripts and GNU/Linux command-line applications including:
  - Tools: vim, emacs, tmux
  - Languages: [NodeJS](#), Javascript, [Python](#), Ruby, C/C++, C# & F#, Rust, Go, etc.
  - Services: SSHD, [MySQL](#), Apache, lighttpd, [MongoDB](#), [PostgreSQL](#).
- Install additional software using your own GNU/Linux distribution package manager.
- Invoke Windows applications using a Unix-like command-line shell.
- Invoke GNU/Linux applications on Windows.

[Install WSL](#)

## What is WSL 2?

WSL 2 is a new version of the Windows Subsystem for Linux architecture that powers the Windows Subsystem for Linux to run ELF64 Linux binaries on Windows. Its primary goals are to **increase file system performance**, as well as adding **full system call compatibility**.

This new architecture changes how these Linux binaries interact with Windows and your computer's hardware, but still provides the same user experience as in WSL 1 (the current widely available version).

Individual Linux distributions can be run with either the WSL 1 or WSL 2 architecture. Each distribution can be upgraded or downgraded at any time and you can run WSL 1 and WSL 2 distributions side by side. WSL 2 uses an entirely new architecture that benefits from running a real Linux kernel.

## Next steps

- [Install WSL 1 and update to WSL 2](#)
- [Compare WSL 2 and WSL 1](#)
- [Create a user account and password for your new Linux distribution](#)
- [Read Frequently Asked Questions](#)
- [Read Frequently Asked Questions about WSL 2](#)
- [Troubleshooting](#)

- [Run interoperability commands between Windows and Linux](#)
- [Run launch commands and configurations](#)
- [Set up file permissions](#)
- [Set up WSL for Enterprise](#)
- [Reference WSL commands](#)
- [Build custom distributions](#)
- [Read the WSL Release Notes](#)

# Comparing WSL 1 and WSL 2

6/3/2021 • 12 minutes to read • [Edit Online](#)

The primary difference and reasons for updating the Windows Subsystem for Linux from WSL 1 to WSL 2 are to:

- **increase file system performance,**
- **support full system call compatibility.**

WSL 2 uses the latest and greatest in virtualization technology to run a Linux kernel inside of a lightweight utility virtual machine (VM). However, WSL 2 is not a traditional VM experience.

[Install WSL 1 and update to WSL 2](#)

## Comparing features

FEATURE	WSL 1	WSL 2
Integration between Windows and Linux	✓	✓
Fast boot times	✓	✓
Small resource foot print compared to traditional Virtual Machines	✓	✓
Runs with current versions of VMware and VirtualBox	✓	✓
Managed VM	✗	✓
Full Linux Kernel	✗	✓
Full system call compatibility	✗	✓
Performance across OS file systems	✓	✗

As you can tell from the comparison table above, the WSL 2 architecture outperforms WSL 1 in several ways, with the exception of performance across OS file systems.

## Performance across OS file systems

We recommend against working across operating systems with your files, unless you have a specific reason for doing so. For the fastest performance speed, store your files in the WSL file system if you are working in a Linux command line (Ubuntu, OpenSUSE, etc). If you're working in a Windows command line (PowerShell, Command Prompt), store your files in the Windows file system.

For example, when storing your WSL project files:

- Use the Linux file system root directory: `\\wsl$\\Ubuntu-18.04\\home\\<user name>\\Project`
- Not the Windows file system root directory: `C:\\Users\\<user name>\\Project`

All currently running distributions (`ws1 -l`) are accessible via network connection. To get there run a command [WIN+R] (keyboard shortcut) or type in File Explorer address bar `\\ws1$` to find respective distribution names and access their root file systems.

You can also use windows commands inside WSL's Linux [Terminal](#). Try opening a Linux distribution (ie Ubuntu), be sure that you are in the Linux home directory by entering this command: `cd ~`. Then open your Linux file system in File Explorer by entering *(don't forget the period at the end)*: `powershell.exe /c start .`

#### IMPORTANT

If you experience an error `-bash: powershell.exe: command not found` please refer to the [WSL troubleshooting page](#) to resolve it.

WSL 2 is only available in Windows 10, Version 1903, Build 18362 or higher. Check your Windows version by selecting the **Windows logo key + R**, type `winver`, select **OK**. (Or enter the `ver` command in Windows Command Prompt). You may need to [update to the latest Windows version](#). For builds lower than 18362, WSL is not supported at all.

#### NOTE

WSL 2 will work with [VMware 15.5.5+](#) and [VirtualBox 6+](#). Learn more in our [WSL 2 FAQs](#).

## What's new in WSL 2

WSL 2 is a major overhaul of the underlying architecture and uses virtualization technology and a Linux kernel to enable new features. The primary goals of this update are to increase file system performance and add full system call compatibility.

- [WSL 2 system requirements](#)
- [Update from WSL 1 to WSL 2](#)
- [Frequently Asked Questions about WSL 2](#)

### WSL 2 architecture

A traditional VM experience can be slow to boot up, is isolated, consumes a lot of resources, and requires your time to manage it. WSL 2 does not have these attributes.

WSL 2 provides the benefits of WSL 1, including seamless integration between Windows and Linux, fast boot times, a small resource footprint, and requires no VM configuration or management. While WSL 2 does use a VM, it is managed and run behind the scenes, leaving you with the same user experience as WSL 1.

### Full Linux kernel

The Linux kernel in WSL 2 is built by Microsoft from the latest stable branch, based on the source available at [kernel.org](#). This kernel has been specially tuned for WSL 2, optimizing for size and performance to provide an amazing Linux experience on Windows. The kernel will be serviced by Windows updates, which means you will get the latest security fixes and kernel improvements without needing to manage it yourself.

The [WSL 2 Linux kernel is open source](#). If you'd like to learn more, check out the blog post [Shipping a Linux Kernel with Windows](#) written by the team that built it.

### Increased file IO performance

File intensive operations like git clone, npm install, apt update, apt upgrade, and more are all noticeably faster with WSL 2.

The actual speed increase will depend on which app you're running and how it is interacting with the file system.

Initial versions of WSL 2 run up to 20x faster compared to WSL 1 when unpacking a zipped tarball, and around 2-5x faster when using git clone, npm install and cmake on various projects.

### Full system call compatibility

Linux binaries use system calls to perform functions such as accessing files, requesting memory, creating processes, and more. Whereas WSL 1 used a translation layer that was built by the WSL team, WSL 2 includes its own Linux kernel with full system call compatibility. Benefits include:

- A whole new set of apps that you can run inside of WSL, such as [Docker](#) and more.
- Any updates to the Linux kernel are immediately ready for use. (You don't have to wait for the WSL team to implement updates and add the changes).

## Exceptions for using WSL 1 rather than WSL 2

We recommend that you use WSL 2 as it offers faster performance and 100% system call compatibility. However, there are a few specific scenarios where you might prefer using WSL 1. Consider using WSL 1 if:

- Your project files must be stored in the Windows file system. WSL 1 offers faster access to files mounted from Windows.
  - If you will be using your WSL Linux distribution to access project files on the Windows file system, and these files cannot be stored on the Linux file system, you will achieve faster performance across the OS file systems by using WSL 1.
- A project which requires cross-compilation using both Windows and Linux tools on the same files.
  - File performance across the Windows and Linux operating systems is faster in WSL 1 than WSL 2, so if you are using Windows applications to access Linux files, you will currently achieve faster performance with WSL 1.
- Your project needs access to a serial port or USB device.
  - According to the [WSL 2 FAQ](#), WSL 2 does not include support for accessing serial ports. The [open issue on serial support](#) indicates that support has not been added yet.
- You have strict memory requirements
  - WSL 2's memory usage grows and shrinks as you use it. When a process frees memory this is automatically returned to Windows. However, as of right now WSL 2 does not yet release cached pages in memory back to Windows until the WSL instance is shut down. If you have long running WSL sessions, or access a very large amount of files, this cache can take up memory on Windows. We are tracking the work to improve this experience on [the WSL Github repository issue 4166](#).

#### NOTE

Consider trying the VS Code [Remote WSL Extension](#) to enable you to store your project files on the Linux file system, using Linux command line tools, but also using VS Code on Windows to author, edit, debug, or run your project in an internet browser without any of the performance slow-downs associated with working across the Linux and Windows file systems. [Learn more](#).

## Accessing network applications

### Accessing Linux networking apps from Windows (localhost)

If you are building a networking app (for example an app running on a NodeJS or SQL server) in your Linux distribution, you can access it from a Windows app (like your Edge or Chrome internet browser) using

`localhost` (just like you normally would).

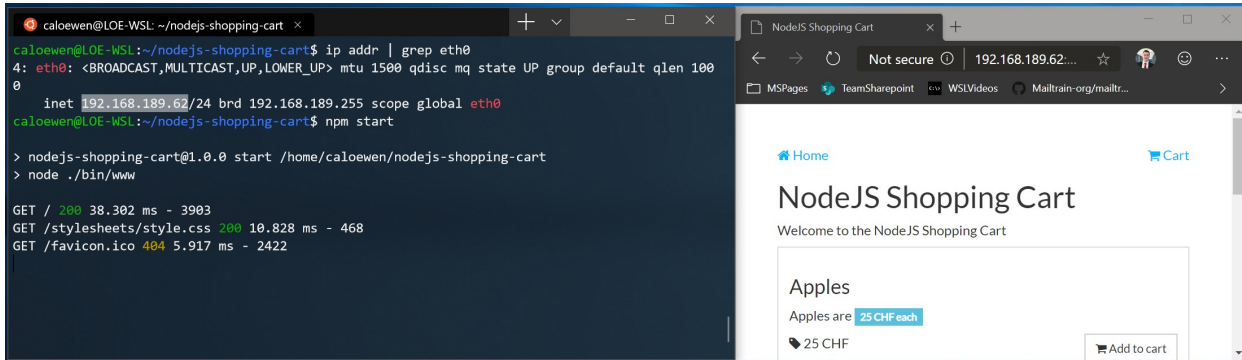
However, if you are running an older version of Windows (Build 18945 or less), you will need to get the IP address of the Linux host VM (or [update to the latest Windows version](#)).



To find the IP address of the virtual machine powering your Linux distribution:

- From your WSL distribution (ie Ubuntu), run the command: `ip addr`
- Find and copy the address under the `inet` value of the `eth0` interface.
- If you have the `grep` tool installed, find this more easily by filtering the output with the command:  
`ip addr | grep eth0`
- Connect to your Linux server using this IP address.

The picture below shows an example of this by connecting to a Node.js server using the Edge browser.

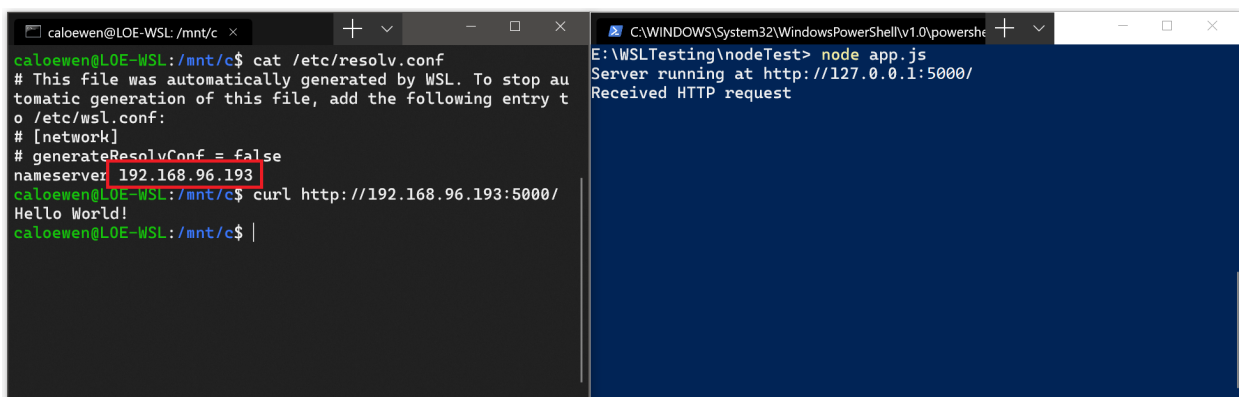


### Accessing Windows networking apps from Linux (host IP)

If you want to access a networking app running on Windows (for example an app running on a NodeJS or SQL server) from your Linux distribution (ie Ubuntu), then you need to use the IP address of your host machine. While this is not a common scenario, you can follow these steps to make it work.

1. Obtain the IP address of your host machine by running this command from your Linux distribution:  
`cat /etc/resolv.conf`
2. Copy the IP address following the term: `nameserver`.
3. Connect to any Windows server using the copied IP address.

The picture below shows an example of this by connecting to a Node.js server running in Windows via `curl`.



### Additional networking considerations

#### Connecting via remote IP addresses

When using remote IP addresses to connect to your applications, they will be treated as connections from the Local Area Network (LAN). This means that you will need to make sure your application can accept LAN connections.

For example, you may need to bind your application to `0.0.0.0` instead of `127.0.0.1`. In the example of a Python app using Flask, this can be done with the command: `app.run(host='0.0.0.0')`. Please keep security in mind when making these changes as this will allow connections from your LAN.

#### Accessing a WSL 2 distribution from your local area network (LAN)

When using a WSL 1 distribution, if your computer was set up to be accessed by your LAN, then applications

run in WSL could be accessed on your LAN as well.

This isn't the default case in WSL 2. WSL 2 has a virtualized ethernet adapter with its own unique IP address. Currently, to enable this workflow you will need to go through the same steps as you would for a regular virtual machine. (We are looking into ways to improve this experience.)

Here's an example PowerShell command to add a port proxy that listens on port 4000 on the host and connects it to port 4000 to the WSL 2 VM with IP address 192.168.101.100.

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

#### IPv6 access

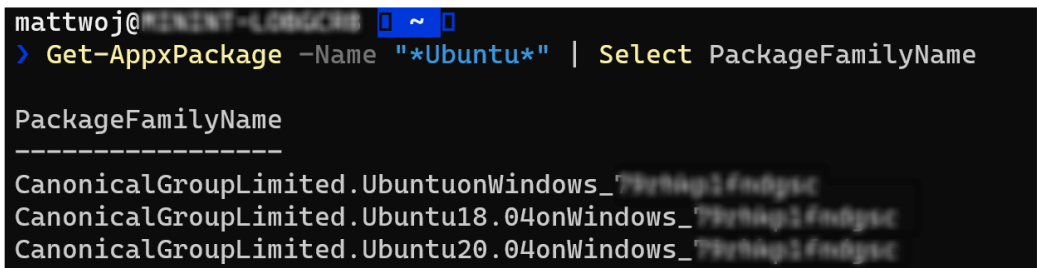
WSL 2 distributions currently cannot reach IPv6-only addresses. We are working on adding this feature.

## Expanding the size of your WSL 2 Virtual Hard Disk

WSL 2 uses a Virtual Hard Disk (VHD) to store your Linux files. In WSL 2, a VHD is represented on your Windows hard drive as a `.vhd` file.

The WSL 2 VHD uses the ext4 file system. This VHD automatically resizes to meet your storage needs and has an initial maximum size of 256GB. If the storage space required by your Linux files exceeds this size you may need to expand it. If your distribution grows in size to be greater than 256GB, you will see errors stating that you've run out of disk space. You can fix this error by expanding the VHD size.

To expand your maximum VHD size beyond 256GB:

1. Terminate all WSL instances using the command: `ws1 --shutdown`
2. To find your distribution installation package name ('PackageFamilyName'):
  - Using PowerShell (where 'distro' is your distribution name) enter the command:
  - `Get-AppxPackage -Name "*<distro>*" | Select PackageFamilyName`
  - For example: `Get-AppxPackage -Name "*Ubuntu*" | Select PackageFamilyName`

```
mattwoj@C:\Users\mattwoj> Get-AppxPackage -Name "*Ubuntu*" | Select PackageFamilyName  
PackageFamilyName  
-----  
CanonicalGroupLimited.UbuntuonWindows_799566888-bfb66d46-8282-4645-b8c8-000000000000  
CanonicalGroupLimited.Ubuntu18.04onWindows_799566888-bfb66d46-8282-4645-b8c8-000000000000  
CanonicalGroupLimited.Ubuntu20.04onWindows_799566888-bfb66d46-8282-4645-b8c8-000000000000
```
3. Use the resulting `PackageFamilyName` to locate the VHD file `fullpath` used by your WSL 2 installation, this will be your `pathToVHD`. To find the full path:
  - In your Start menu, enter: `%LOCALAPPDATA%` and select to open the `%LOCALAPPDATA%` file folder.
  - Next, open the "Packages" folder and search for the `PackageFamilyName` of your distribution. Open that folder (ie. `CanonicalGroupLimited.Ubuntu20.04onWindows_79xxxx`).
  - Inside the `PackageFamilyName` folder, open the "LocalState" folder and find the `<disk>.vhd` file.
  - Copy the path to that file, it should look something like:  
`%LOCALAPPDATA%\Packages\<PackageFamilyName>\LocalState\<disk>.vhd`
  - For example, the `<pathToVHD>` for Ubuntu 20.04 should look something like:  
`%LOCALAPPDATA%\Packages\CanonicalGroupLimited.Ubuntu20.04onWindows_79xxxx\LocalState\ext4.vhd`.
4. Resize your WSL 2 VHD by completing the following commands:

- Open Windows Command Prompt with admin privileges and enter:

```
diskpart
```

```
DISKPART> Select vdisk file="<pathToVHD>"
```

```
DISKPART> detail vdisk
```

- Examine the output of the **detail** command. The output will include a value for **Virtual size**. This is the current maximum. Convert this value to megabytes. For example, if the **detail** output shows **Virtual size: 256 GB**, convert this to **256000**.
- The new value you enter must be greater than this original value. As an example, to double the virtual size listed above, you could enter the value: **512000**. Once you have determined the number you would like to set for your new size (in megabytes), enter the following command in your Windows Command Prompt **diskpart** prompt:

```
DISKPART> expand vdisk maximum=<sizeInMegabytes>
```

- Exit **diskpart**

```
DISKPART> exit
```

5. Launch your WSL distribution (Ubuntu, for example).
6. Make WSL aware that it can expand its file system's size by running these commands from your WSL distribution command line.

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

- You may see this message in response to the first **mount** command: `"/dev: none already mounted on /dev."` This message can safely be ignored.
- Copy the name of this entry, which will look like: `/dev/sdX` (with the X representing any other character). In the following example the value of X is b:

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

- Using the example from above, we changed the vhd size to **512000**, so the command would be:

```
sudo resize2fs /dev/sdb 512000M .
```

#### NOTE

You may need to install **resize2fs**. If so, you can use this command to install it: `sudo apt install resize2fs` .

The output will look similar to the following:

```
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.
```

### IMPORTANT

We recommend that you do not modify, move, or access the WSL related files located inside of your AppData folder using Windows tools or editors. Doing so could cause your Linux distribution to become corrupted. If you would like to access your Linux files from Windows, that is possible via the path `\\ws1$\<<distroName>\`. Open your WSL distribution and enter `explorer.exe .` to view that folder. To learn more, see the blog post: [Accessing Linux files from Windows](#).

# Windows Subsystem for Linux Installation Guide for Windows 10

6/3/2021 • 10 minutes to read • [Edit Online](#)

There are two options available for installing Windows Subsystem for Linux (WSL):

- **Simplified install** (*preview release*): `wsl --install`

The `wsl --install` simplified install command requires that you join the [Windows Insiders Program](#) and install a preview build of Windows 10 (OS build 20262 or higher), but eliminates the need to follow the manual install steps. All you need to do is open a command window with administrator privileges and run `wsl --install`, after a restart you will be ready to use WSL.

- **Manual install**: Follow the six steps listed below.

The manual install steps for WSL are listed below and can be used to install Linux on any version of Windows 10.

## NOTE

If you run into an issue during the install process, check the [Troubleshooting installation](#) section at the bottom of this page.

## Simplified Installation for Windows Insiders

The installation process for Windows Subsystem for Linux has been significantly improved in the latest Windows Insiders preview builds of Windows 10, replacing the manual steps below with a single command.

In order to use the `wsl --install` simplified install command, you must:

- Join the [Windows Insiders Program](#)
- Install a preview build of Windows 10 (OS build 20262 or higher).
- Open a command line windows with Administrator privileges

Once those requirements are met, to install WSL:

- Enter this command in the command line you've opened in Admin mode: `wsl.exe --install`
- Restart your machine

The first time you launch a newly installed Linux distribution, a console window will open and you'll be asked to wait for files to de-compress and be stored on your PC. All future launches should take less than a second.

You will then need to [create a user account and password for your new Linux distribution](#).

**CONGRATULATIONS! You've successfully installed and set up a Linux distribution that is completely integrated with your Windows operating system!**

The `--install` command performs the following actions:

- Enables the optional WSL and Virtual Machine Platform components
- Downloads and installs the latest Linux kernel
- Sets WSL 2 as the default

- Downloads and installs a Linux distribution (*reboot may be required*)

By default, the installed Linux distribution will be Ubuntu. This can be changed using

`wsl --install -d <Distribution Name>`. (Replacing `<Distribution Name>` with the name of your desired distribution.) Additional Linux distributions may be added to your machine after the initial install using the `wsl --install -d <Distribution Name>` command.

To see a list of available Linux distributions, enter `wsl --list --online`.

## Manual Installation Steps

If you are not on a Windows Insiders build, the features required for WSL will need to be enabled manually following the steps below.

### Step 1 - Enable the Windows Subsystem for Linux

You must first enable the "Windows Subsystem for Linux" optional feature before installing any Linux distributions on Windows.

Open PowerShell as Administrator and run:

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

We recommend now moving on to step #2, updating to WSL 2, but if you wish to only install WSL 1, you can now **restart** your machine and move on to [Step 6 - Install your Linux distribution of choice](#). To update to WSL 2, **wait to restart** your machine and move on to the next step.

### Step 2 - Check requirements for running WSL 2

To update to WSL 2, you must be running Windows 10.

- For x64 systems: **Version 1903** or higher, with **Build 18362** or higher.
- For ARM64 systems: **Version 2004** or higher, with **Build 19041** or higher.
- Builds lower than 18362 do not support WSL 2. Use the [Windows Update Assistant](#) to update your version of Windows.

To check your version and build number, select **Windows logo key + R**, type **winver**, select **OK**. [Update to the latest Windows version](#) in the Settings menu.

#### NOTE

If you are running Windows 10 version 1903 or 1909, open "Settings" from your Windows menu, navigate to "Update & Security" and select "Check for Updates". Your Build number must be 18362.1049+ or 18363.1049+, with the minor build # over .1049. Read more: [WSL 2 Support is coming to Windows 10 Versions 1903 and 1909](#). See the [troubleshooting instructions](#).

### Step 3 - Enable Virtual Machine feature

Before installing WSL 2, you must enable the **Virtual Machine Platform** optional feature. Your machine will require [virtualization capabilities](#) to use this feature.

Open PowerShell as Administrator and run:

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

**Restart** your machine to complete the WSL install and update to WSL 2.

## Step 4 - Download the Linux kernel update package

1. Download the latest package:

- [WSL2 Linux kernel update package for x64 machines](#)

### NOTE

If you're using an ARM64 machine, please download the [ARM64 package](#) instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: `systeminfo | find "System Type"`.

**Caveat:** On non-English Windows versions, you might have to modify the search text, for example, in German it would be `systeminfo | find "Systemtyp"`.

2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)

Once the installation is complete, move on to the next step - setting WSL 2 as your default version when installing new Linux distributions. (Skip this step if you want your new Linux installs to be set to WSL 1).

### NOTE

For more information, read the article [changes to updating the WSL2 Linux kernel](#), available on the [Windows Command Line Blog](#).

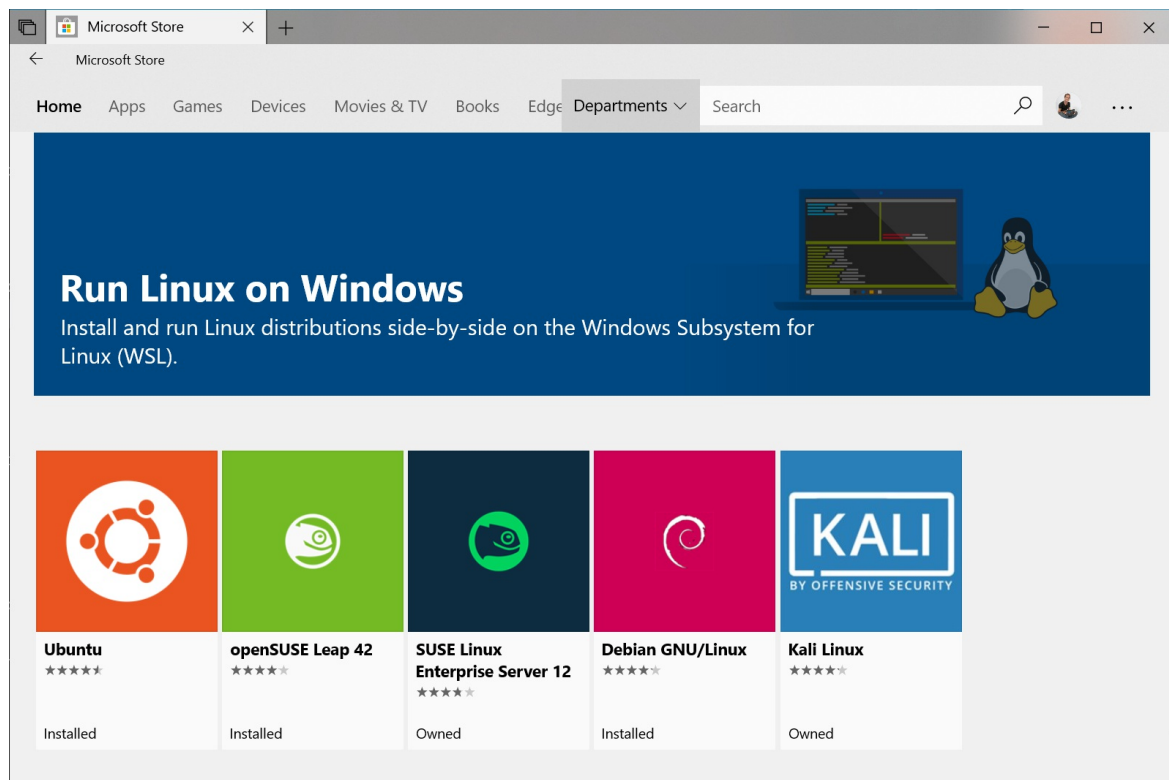
## Step 5 - Set WSL 2 as your default version

Open PowerShell and run this command to set WSL 2 as the default version when installing a new Linux distribution:

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

## Step 6 - Install your Linux distribution of choice

1. Open the [Microsoft Store](#) and select your favorite Linux distribution.



The following links will open the Microsoft store page for each distribution:


- [Ubuntu 18.04 LTS](#)
- [Ubuntu 20.04 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](#)
- [Penguin](#)
- [Penguin Enterprise](#)
- [Alpine WSL](#)


2. From the distribution's page, select "Get".



Store

Home Apps Games Music Movies & TV Books Microsoft






# Ubuntu

Canonical Group Limited

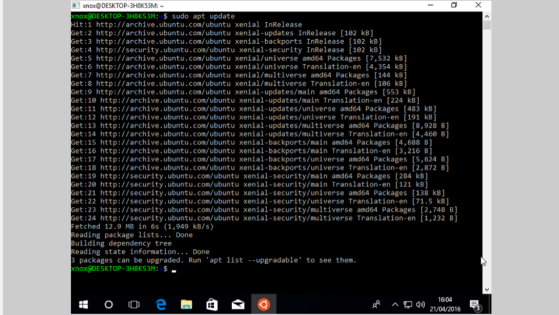
★★★★☆ 22

Free

GetShare

 Everyone

### Screenshots



### Description

Ubuntu on Windows allows one to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.


To use this feature, one first needs to use "Turn Windows features on or off" and select "Windows Subsystem for Linux", click OK, reboot, and use this app.

The above step can also be performed using Administrator PowerShell prompt: `Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`

...

More

### Available on

 PC

### Features

- Ubuntu
- bash
- ssh

### What's new in this version

20170619.1 build of Ubuntu 16.04 LTS

The first time you launch a newly installed Linux distribution, a console window will open and you'll be asked to wait for a minute or two for files to de-compress and be stored on your PC. All future launches should take less than a second.

You will then need to [create a user account and password for your new Linux distribution](#).

Ubuntu

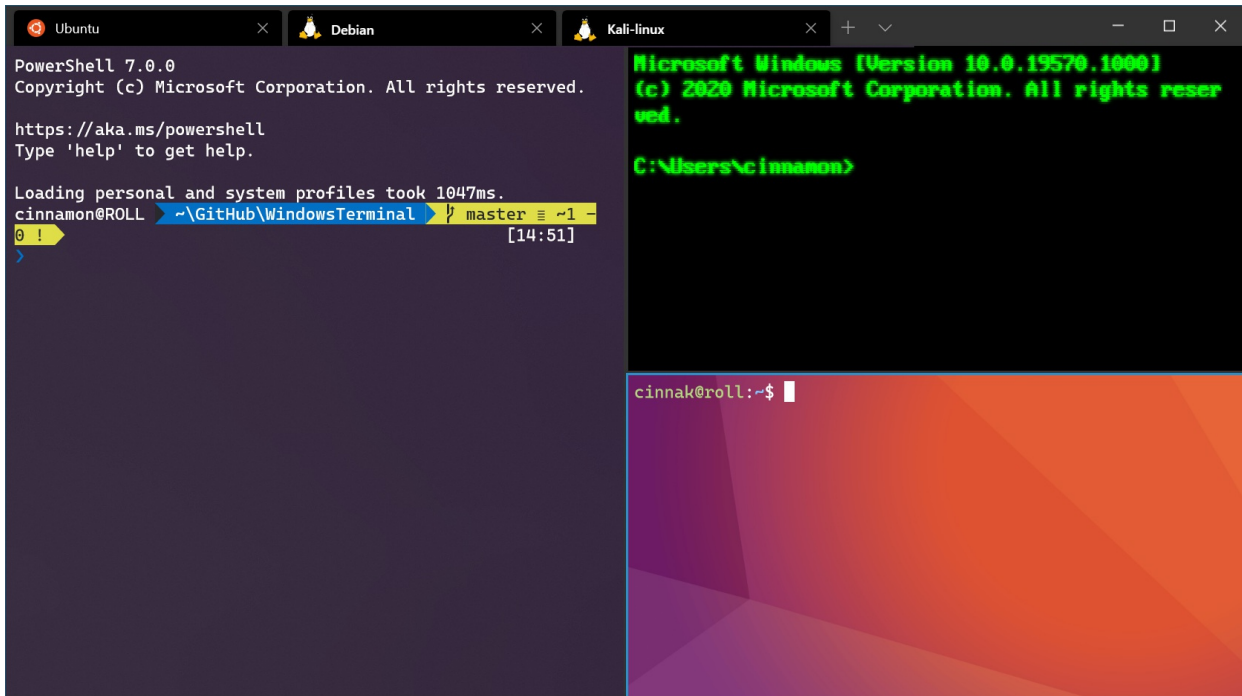
```
Installing, this may take a few minutes...
Installation successful!
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

CONGRATULATIONS! You've successfully installed and set up a Linux distribution that is completely integrated with your Windows operating system!

## Install Windows Terminal (optional)

Windows Terminal enables multiple tabs (quickly switch between multiple Linux command lines, Windows Command Prompt, PowerShell, Azure CLI, etc), create custom key bindings (shortcut keys for opening or closing tabs, copy+paste, etc.), use the search feature, and custom themes (color schemes, font styles and sizes, background image/blur/transparency). [Learn more](#).

[Install Windows Terminal](#).



## Set your distribution version to WSL 1 or WSL 2

You can check the WSL version assigned to each of the Linux distributions you have installed by opening the PowerShell command line and entering the command (only available in [Windows Build 18362 or higher](#)):

```
wsl -l -v
```

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

To set a distribution to be backed by either version of WSL please run:

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

Make sure to replace `<distribution name>` with the actual name of your distribution and `<versionNumber>` with the number '1' or '2'. You can change back to WSL 1 at anytime by running the same command as above but replacing the '2' with a '1'.

## NOTE

The update from WSL 1 to WSL 2 may take several minutes to complete depending on the size of your targeted distribution. If you are running an older (legacy) installation of WSL 1 from Windows 10 Anniversary Update or Creators Update, you may encounter an update error. Follow these instructions to [uninstall and remove any legacy distributions](#).

If `wsl --set-default-version` results as an invalid command, enter `wsl --help`. If the `--set-default-version` is not listed, it means that your OS doesn't support it and you need to update to version 1903, Build 18362 or higher. If you are on Build 19041 for ARM64, this command may fail when using PowerShell in which case you can use a *Command Prompt* instead to issue the `wsl.exe` command.

If you see this message after running the command:

```
WSL 2 requires an update to its kernel component. For information please visit https://aka.ms/ws12kernel .  
You still need to install the MSI Linux kernel update package.
```

Additionally, if you want to make WSL 2 your default architecture you can do so with this command:

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

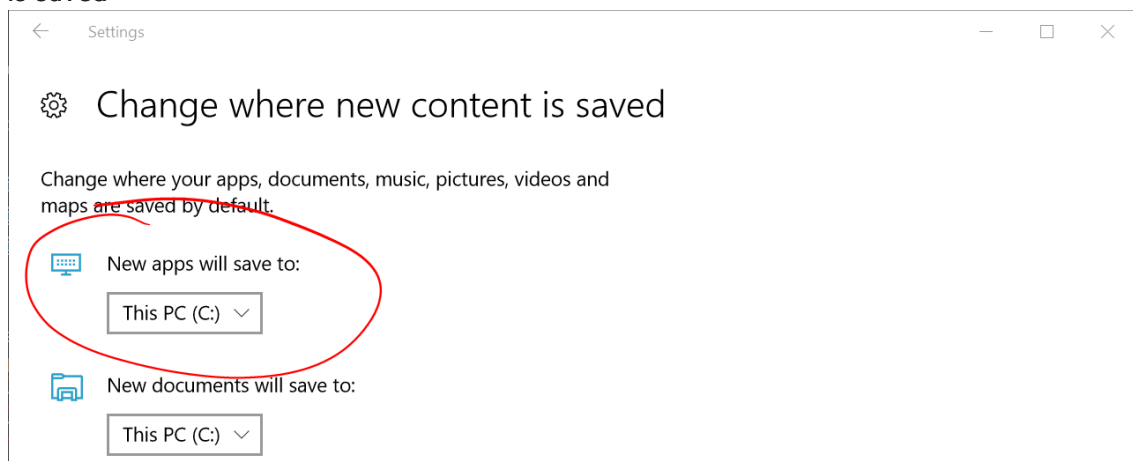
This will set the version of any new distribution installed to WSL 2.

## Troubleshooting installation

Below are related errors and suggested fixes. Refer to the [WSL troubleshooting page](#) for other common errors and their solutions.

### • Installation failed with error 0x80070003

- The Windows Subsystem for Linux only runs on your system drive (usually this is your `C:` drive). Make sure that distributions are stored on your system drive:
- Open **Settings** -> **System** --> **Storage** -> **More Storage Settings: Change where new content is saved**



### • WslRegisterDistribution failed with error 0x8007019e

- The Windows Subsystem for Linux optional component is not enabled:
- Open **Control Panel** -> **Programs and Features** -> **Turn Windows Feature on or off** -> Check **Windows Subsystem for Linux** or using the PowerShell cmdlet mentioned at the beginning of this article.

### • Installation failed with error 0x80070003 or error 0x80370102

- Please make sure that virtualization is enabled inside of your computer's BIOS. The instructions on how to do this will vary from computer to computer, and will most likely be under CPU related

options.

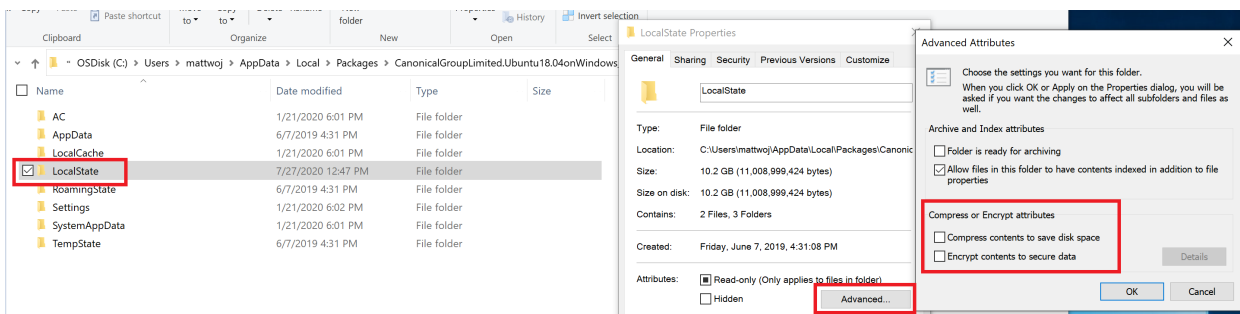
- WSL2 requires that your CPU supports the Second Level Address Translation (SLAT) feature, which was introduced in Intel Nehalem processors (Intel Core 1st Generation) and AMD Opteron. Older CPUs (such as the Intel Core 2 Duo) will not be able to run WSL2, even if the Virtual Machine Platform is successfully installed.

- **Error when trying to upgrade:** `Invalid command line option: wsl --set-version Ubuntu 2`

- Enure that you have the Windows Subsystem for Linux enabled, and that you're using Windows Build version 18362 or higher. To enable WSL run this command in a PowerShell prompt with admin privileges: `Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux`.

- **The requested operation could not be completed due to a virtual disk system limitation. Virtual hard disk files must be uncompressed and unencrypted and must not be sparse.**

- Deselect "Compress contents" (as well as "Encrypt contents" if that's checked) by opening the profile folder for your Linux distribution. It should be located in a folder on your Windows file system, something like: `USERPROFILE\AppData\Local\Packages\CanonicalGroupLimited...`
- In this Linux distro profile, there should be a LocalState folder. Right-click this folder to display a menu of options. Select Properties > Advanced and then ensure that the "Compress contents to save disk space" and "Encrypt contents to secure data" checkboxes are unselected (not checked). If you are asked whether to apply this to just to the current folder or to all subfolders and files, select "just this folder" because you are only clearing the compress flag. After this, the `wsl --set-version` command should work.



#### NOTE

In my case, the LocalState folder for my Ubuntu 18.04 distribution was located at `C:\Users<my-user-name>\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu18.04onWindows_79rhk1fndgsc`

Check [WSL Docs GitHub thread #4103](#) where this issue is being tracked for updated information.

- **The term 'wsl' is not recognized as the name of a cmdlet, function, script file, or operable program.**

- Ensure that the [Windows Subsystem for Linux Optional Component is installed](#). Additionally, if you are using an ARM64 device and running this command from PowerShell, you will receive this error. Instead run `wsl.exe` from [PowerShell Core](#), or Command Prompt.

- **Error: This update only applies to machines with the Windows Subsystem for Linux.**

- To install the Linux kernel update MSI package, WSL is required and should be enabled first. If it fails, it you will see the message: `This update only applies to machines with the Windows Subsystem for Linux`.
- There are three possible reason you see this message:
  - You are still in old version of Windows which doesn't support WSL 2. See step #2 for version requirements and links to update.
  - WSL is not enabled. You will need to return to step #1 and ensure that the optional WSL feature is

enabled on your machine.

3. After you enabled WSL, a reboot is required for it to take effect, reboot your machine and try again.

- **Error: WSL 2 requires an update to its kernel component. For information please visit <https://aka.ms/wsl2kernel> .**
  - If the Linux kernel package is missing in the %SystemRoot%\system32\lxss\tools folder, you will encounter this error. Resolve it by installing the Linux kernel update MSI package in step #4 of these installation instructions. You may need to uninstall the MSI from '[Add or Remove Programs](#)', and install it again.

# Windows Server Installation Guide

4/14/2021 • 2 minutes to read • [Edit Online](#)

The Windows Subsystem for Linux is available for installation on Windows Server 2019 (version 1709) and later. This guide will walk through the steps of enabling WSL on your machine.

## Enable the Windows Subsystem for Linux

Before you can run Linux distros on Windows, you must enable the "Windows Subsystem for Linux" optional feature and reboot.

Open PowerShell as Administrator and run:

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

## Download a Linux distribution

Follow [these instructions](#) to download your favorite Linux distribution.

## Extract and install a Linux distribution

Now that you've downloaded a Linux distribution, in order to extract its contents and manually install, follow these steps:

1. Extract the `<distro>.appx` package's contents, using PowerShell:

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

2. Run the distribution launcher application in the target folder. The launcher is typically named `<distro>.exe` (for example, `ubuntu.exe`).

```
Administrator: C:\Windows\system32\cmd.exe - PowerShell
d-r---      8/2/2017    3:50 PM          Favorites
d-r---      8/2/2017    3:50 PM          Links
d-r---      8/2/2017    3:50 PM          Music
d-----      8/2/2017    4:15 PM        OpenSUSE
d-r---      8/2/2017    3:50 PM          Pictures
d-r---      8/2/2017    3:50 PM          Saved Games
d-r---      8/2/2017    3:50 PM          Searches
d-----      8/3/2017   12:27 PM          Ubuntu
d-r---      8/2/2017    3:50 PM          Videos
-a-----      8/3/2017   12:05 PM    200970356 OpenSuse.zip
-a-----      8/3/2017   12:20 PM    201749452 Ubuntu.zip

PS C:\Users\Administrator> cd .\Ubuntu\
PS C:\Users\Administrator\Ubuntu> ls

Directory: C:\Users\Administrator\Ubuntu

Mode                LastWriteTime         Length Name
----                -
d-----      8/3/2017   12:27 PM              AppxMetadata
d-----      8/3/2017   12:27 PM              Assets
d-----      8/3/2017   12:27 PM              images
-a-----      7/11/2017    6:10 PM         190434 AppxBlockMap.xml
-a-----      7/11/2017    6:10 PM          2475 AppxManifest.xml
-a-----      7/11/2017    6:11 PM         10554 AppxSignature.p7x
-a-----      7/11/2017    6:10 PM        201254783 install.tar.gz
-a-----      7/11/2017    6:10 PM          4840 resources.pri
-a-----      7/11/2017    6:10 PM         222208 ubuntu.exe
-a-----      7/11/2017    6:10 PM           809 [Content_Types].xml

PS C:\Users\Administrator\Ubuntu>
```

#### Caution

**Installation failed with error 0x8007007e:** If you receive this error, then your system doesn't support WSL. Ensure that you're running Windows build 16215 or later. [Check your build](#). Also check to [confirm that WSL is enabled](#) and your computer was restarted after the feature was enabled.

3. Add your distro path to the Windows environment PATH ( `C:\Users\Administrator\Ubuntu` in this example), using PowerShell:

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

You can now launch your distribution from any path by typing `<distro>.exe`. For example: `ubuntu.exe`.

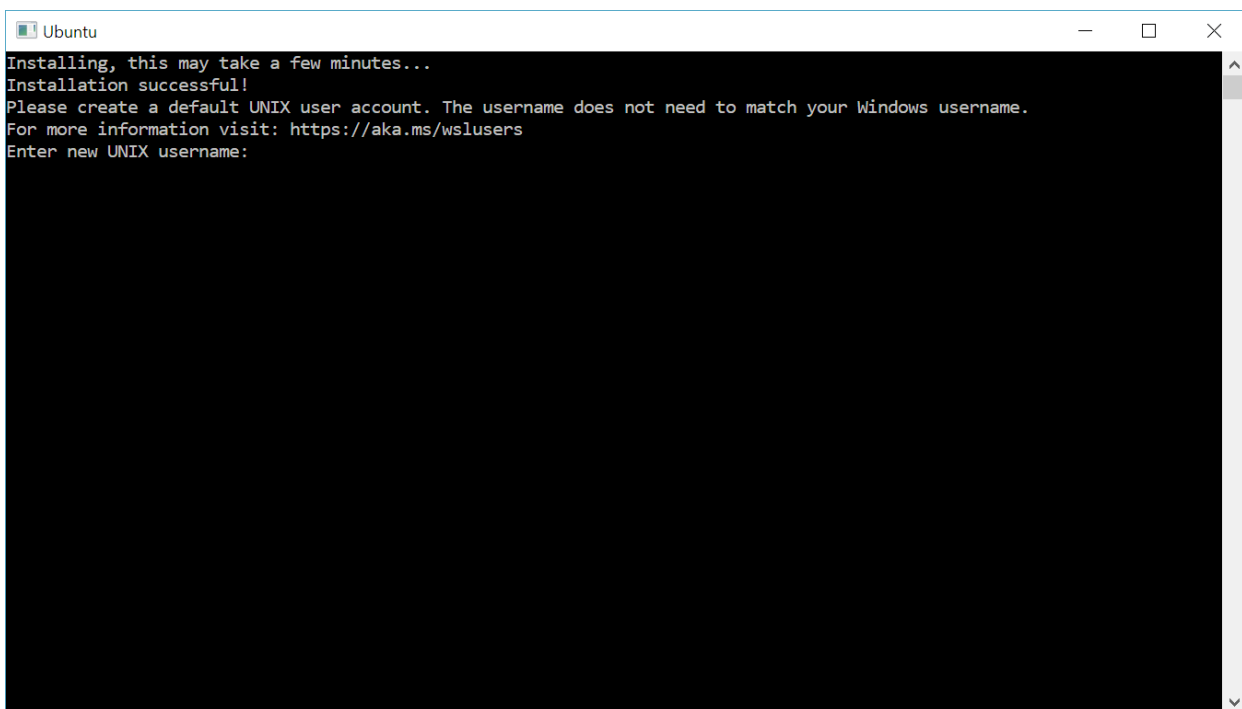
Now that it is installed, you must [initialize your new distribution instance](#) before using it.

# Create a user account and password for your new Linux distribution

4/14/2021 • 2 minutes to read • [Edit Online](#)

Once you have [enabled WSL and installed a Linux distribution from the Microsoft Store](#), the first step you will be asked to complete when opening your newly installed Linux distribution is to create an account, including a **User Name** and **Password**.

- This **User Name** and **Password** is specific to each separate Linux distribution that you install and has no bearing on your Windows user name.
- Once you create a **User Name** and **Password**, the account will be your default user for the distribution and automatically sign-in on launch.
- This account will be considered the Linux administrator, with the ability to run `sudo` (Super User Do) administrative commands.
- Each Linux distribution running on the Windows Subsystem for Linux has its own Linux user accounts and passwords. You will have to configure a Linux user account every time you add a distribution, reinstall, or reset.



## Update and upgrade packages

Most distributions ship with an empty or minimal package catalog. We strongly recommend regularly updating your package catalog and upgrading your installed packages using your distribution's preferred package manager. For Debian/Ubuntu, use `apt`:

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

Windows does not automatically update or upgrade your Linux distribution(s). This is a task that most Linux users prefer to control themselves.



# Reset your Linux password

To change your password, open your Linux distribution (Ubuntu for example) and enter the command: `passwd`

You will be asked to enter your current password, then asked to enter your new password, and then to confirm your new password.

## Forgot your password

If you forgot the password for your Linux distribution:

1. Open PowerShell and enter the root of your default WSL distribution using the command: `wsl -u root`

If you need to update the forgotten password on a distribution that is not your default, use the command: `wsl -d Debian -u root`, replacing `Debian` with the name of your targeted distribution.

2. Once your WSL distribution has been opened at the root level inside PowerShell, you can use this command to update your password: `passwd <WSLUsername>` where `<WSLUsername>` is the username of the account in the DISTRO whose password you've forgotten.
3. You will be prompted to enter a new UNIX password and then confirm that password. Once you're told that the password has updated successfully, close WSL inside of PowerShell using the command: `exit`

### NOTE

If you are running an early version of Windows operating system, like 1703 (Creators Update) or 1709 (Fall Creators Update), see the [archived version of this user account update doc](#).

# Get started using Visual Studio Code with Windows Subsystem for Linux

4/14/2021 • 4 minutes to read • [Edit Online](#)

Visual Studio Code, along with the Remote - WSL extension, enables you to use WSL as your full-time development environment directly from VS Code. You can:

- develop in a Linux-based environment
- use Linux-specific toolchains and utilities
- run and debug your Linux-based applications from the comfort of Windows while maintaining access to productivity tools like Outlook and Office
- use the VS Code built-in terminal to run your Linux distribution of choice
- take advantage of VS Code features like [Intellisense code completion](#), [linting](#), [debug support](#), [code snippets](#), and [unit testing](#)
- easily manage your version control with VS Code's built-in [Git support](#)
- run commands and VS Code extensions directly on your WSL projects
- edit files in your Linux or mounted Windows filesystem (for example /mnt/c) without worrying about pathing issues, binary compatibility, or other cross-OS challenges

## Install VS Code and the Remote WSL extension

- Visit the [VS Code install page](#) and select the 32 or 64 bit installer. Install Visual Studio Code on Windows (not in your WSL file system).
- When prompted to **Select Additional Tasks** during installation, be sure to check the **Add to PATH** option so you can easily open a folder in WSL using the code command.
- Install the [Remote Development extension pack](#). This extension pack includes the Remote - WSL extension, in addition to the Remote - SSH, and Remote - Containers extensions, enabling you to open any folder in a container, on a remote machine, or in WSL.

### IMPORTANT

In order to install the Remote-WSL extension, you will need the [1.35 May release](#) version or later of VS Code. We do not recommend using WSL in VS Code without the Remote-WSL extension as you will lose support for auto-complete, debugging, linting, etc. Fun fact: this WSL extension is installed in \$HOME/.vscode/extensions (enter the command

```
ls $HOME\.vscode\extensions\
```

 in PowerShell).

## Update your Linux distribution

Some WSL Linux distributions are lacking libraries that are required by the VS Code server to start up. You can add additional libraries into your Linux distribution by using its package manager.

For example, to update Debian or Ubuntu, use:

```
sudo apt-get update
```

To add wget (to retrieve content from web servers) and ca-certificates (to allow SSL-based applications to check

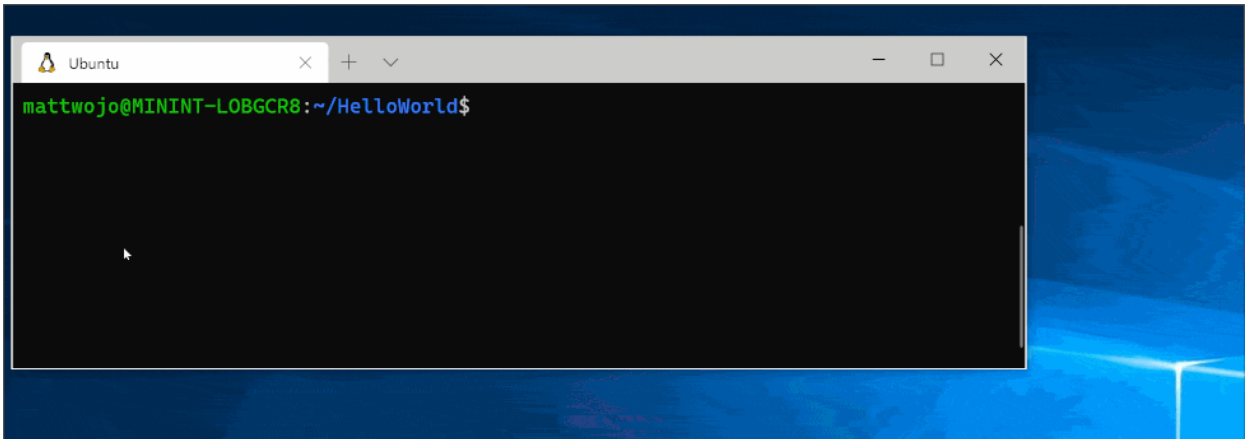
for the authenticity of SSL connections), enter:

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

## Open a WSL project in Visual Studio Code

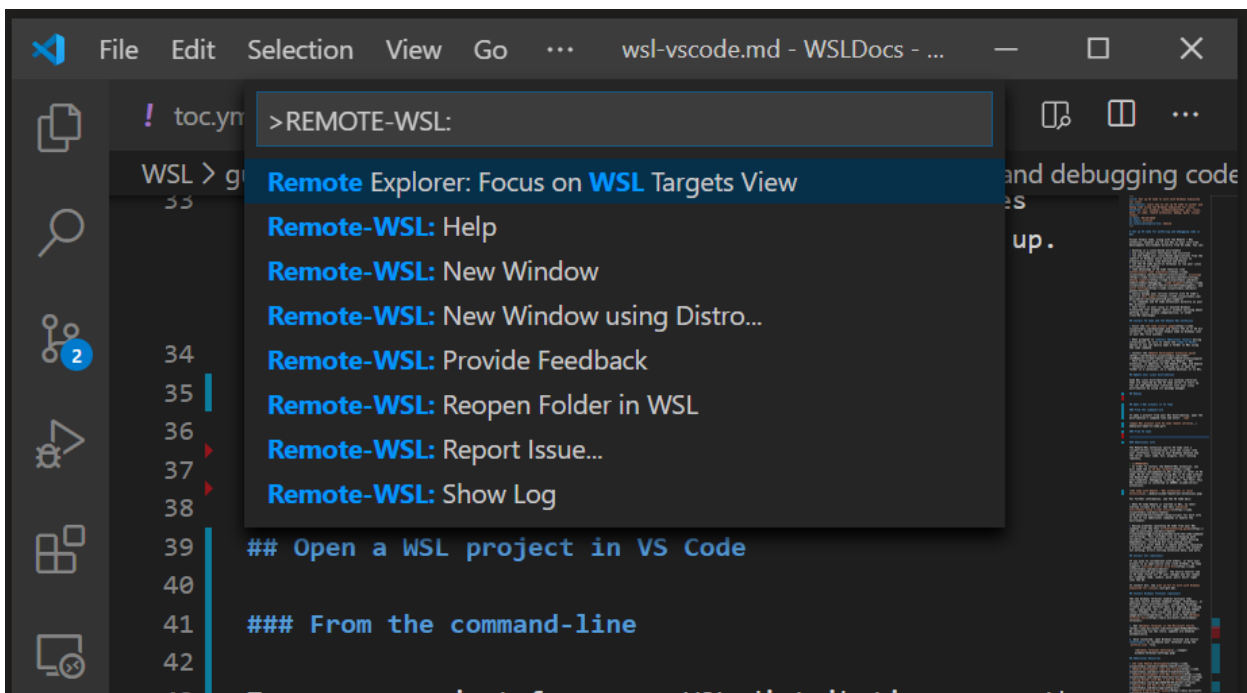
### From the command-line

To open a project from your WSL distribution, open the distribution's command line and enter: `code .`



### From VS Code

You can also access more VS Code Remote options by using the shortcut: `CTRL+SHIFT+P` in VS Code to bring up the command palette. If you then type `Remote-WSL` you will see a list of the VS Code Remote options available, allowing you to reopen the folder in a remote session, specify which distribution you want to open in, and more.




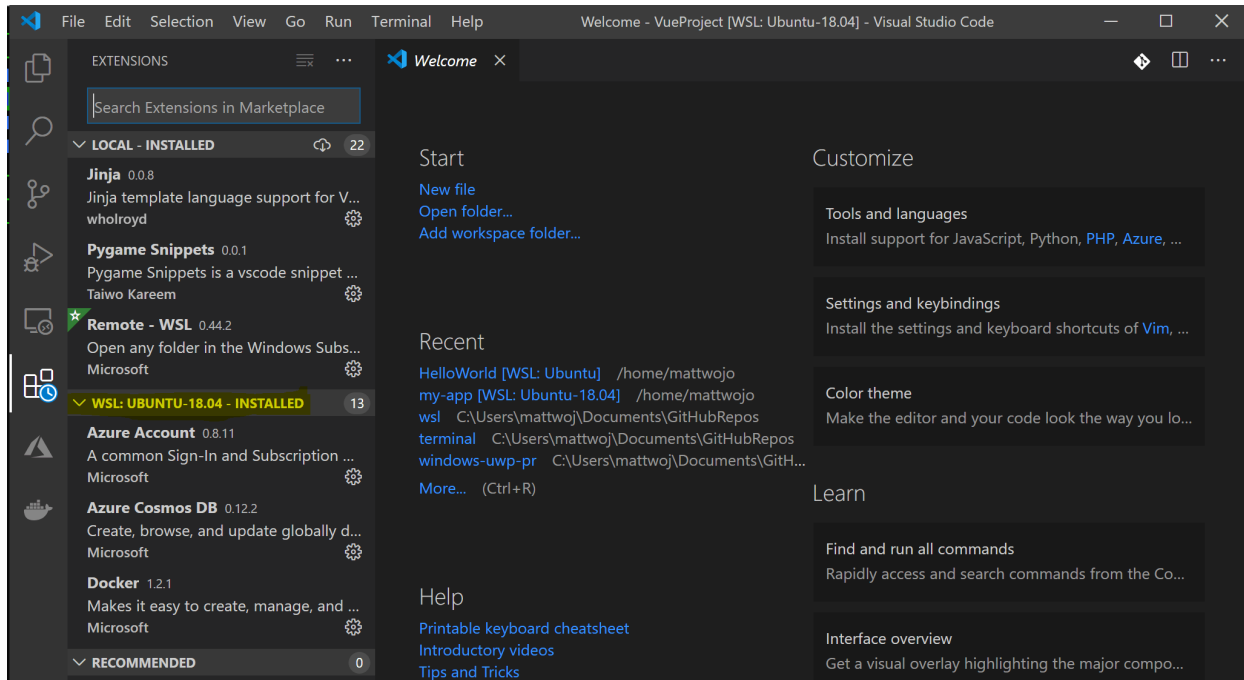
## Extensions inside of VS Code Remote

The Remote-WSL extension splits VS Code into a "client-server" architecture, with the client (the user interface) running on your Windows machine and the server (your code, Git, plugins, etc) running remotely.

When running VS Code Remote, selecting the 'Extensions' tab will display a list of extensions split between your local machine and your WSL distribution.

Installing a local extension, like a [theme](#), only needs to be installed once.

Some extensions, like the [Python extension](#) or anything that handles things like linting or debugging, must be installed separately on each remote WSL distributions. VS Code will display a warning icon , along with a green "Install in WSL" button, if you have an extension locally installed that is not installed on your WSL Remote.



For further information, see the VS Code docs:

- When VS Code Remote is started in WSL, no shell startup scripts are run. See this [advanced environment setup script article](#) for more info on how to run additional commands or modify the environment.
- Having problems launching VS Code from your WSL command line? This [troubleshooting guide](#) includes tips on changing path variables, resolving extension errors about missing dependencies, resolving Git line ending issues, installing a local VSIX on a remote machine, launching a browser window, blocker localhost port, web sockets not working, errors storing extension data, and more.

## Install Git (optional)

If you plan to collaborate with others, or host your project on an open-source site (like GitHub), VS Code supports [version control with Git](#). The Source Control tab in VS Code tracks all of your changes and has common Git commands (add, commit, push, pull) built right into the UI.

To install Git, see [set up Git to work with Windows Subsystem for Linux](#).

## Install Windows Terminal (optional)

The new Windows Terminal enables multiple tabs (quickly switch between Command Prompt, PowerShell, or multiple Linux distributions), custom key bindings (create your own shortcut keys for opening or closing tabs, copy+paste, etc.), emojis 🤖, and custom themes (color schemes, font styles and sizes, background image/blur/transparency). Learn more in the [Windows Terminal docs](#).

1. Get [Windows Terminal in the Microsoft Store](#): By installing via the store, updates are handled automatically.
2. Once installed, open Windows Terminal and select **Settings** to customize your terminal using the `profile.json` file.

## Additional Resources

- [VS Code Remote Development](#)
- [Remote development tips and tricks](#)
- [Remote development with WSL tutorial](#)
- [Using Docker with WSL 2 and VS Code](#)
- [Using C++ and WSL in VS Code](#)
- [Remote R Service for Linux](#)

A few additional extensions you may want to consider include:

- [Keymaps from other editors](#): These extensions can help your environment feel right at home if you're transitioning from another text editor (like Atom, Sublime, Vim, eMacs, Notepad++, etc).
- [Settings Sync](#): Enables you to synchronize your VS Code settings across different installations using GitHub. If you work on different machines, this helps keep your environment consistent across them.
- [Debugger for Chrome](#): Once you finish developing on the server side with Linux, you'll need to develop and test the client side. This extension integrates your VS Code editor with your Chrome browser debugging service, making things a bit more efficient.

# Get started using Git on Windows Subsystem for Linux

6/3/2021 • 4 minutes to read • [Edit Online](#)

Git is the most commonly used version control system. With Git, you can track changes you make to files, so you have a record of what has been done, and have the ability to revert to earlier versions of the files if needed. Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

## Git can be installed on Windows AND on WSL

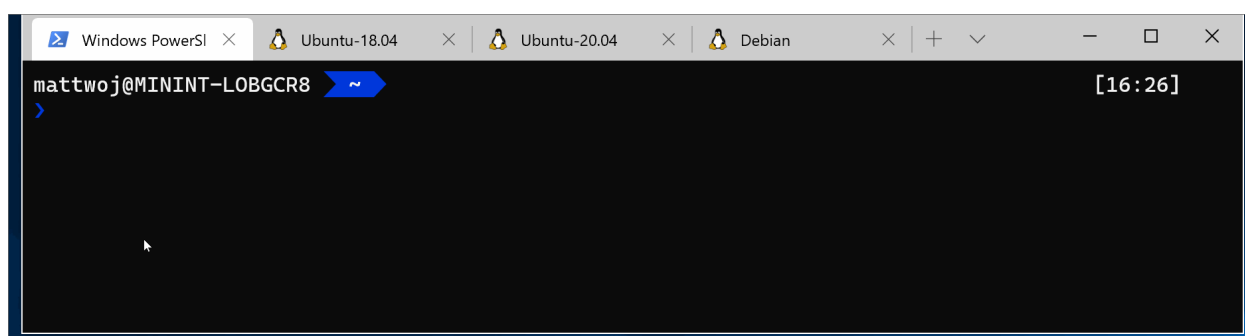
An important consideration: when you enable WSL and install a Linux distribution, you are installing a new file system, separated from the Windows NTFS C:\ drive on your machine. In Linux, drives are not given letters. They are given mount points. The root of your file system `/` is the mount point of your root partition, or folder, in the case of WSL. Not everything under `/` is the same drive. For example, on my laptop, I've installed two version of Ubuntu (20.04 and 18.04), as well as Debian. If I open those distributions, select the home directory with the command `cd ~`, and then enter the command `explorer.exe .`, Windows File Explorer will open and show me the directory path for that distribution.

LINUX DISTRO	WINDOWS PATH TO ACCESS HOME FOLDER
Ubuntu 20.04	<code>\\wsl\$\Ubuntu-20.04\home\username</code>
Ubuntu 18.04	<code>\\wsl\$\Ubuntu-18.04\home\username</code>
Debian	<code>\\wsl\$\Debian\home\username</code>
Windows PowerShell	<code>C:\Users\username</code>

### TIP

If you are seeking to access the Windows file directory from your WSL distribution command line, instead of `C:\Users\username`, the directory would be accessed using `/mnt/c/Users/username`, because the Linux distribution views your Windows file system as a mounted drive.

You will need to install Git on each file system that you intend to use it with.



## Installing Git

Git comes already installed with most of the Windows Subsystem for Linux distributions, however, you may want to update to the latest version. You also will need to set up your git config file.

To install Git, see the [Git Download for Linux](#) site. Each Linux distribution has their own package manager and install command.

For the latest stable Git version in Ubuntu/Debian, enter the command:

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

#### NOTE

You also may want to [install Git for Windows](#) if you haven't already.

## Git config file setup

To set up your Git config file, open a command line for the distribution you're working in and set your name with this command (replacing "Your Name" with your preferred username):

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

Set your email with this command (replacing "youremail@domain.com" with the email you prefer):

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

#### TIP

If you don't yet have a GitHub account, you can [sign-up for one on GitHub](#). If you've never worked with Git before, [GitHub Guides](#) can help you get started. If you need to edit your Git config, you can do so with a built-in text editor like nano: `nano ~/.gitconfig`.

We recommend that you [secure your account with two-factor authentication \(2FA\)](#).

## Git Credential Manager setup

[Git Credential Manager \(GCM\) Core](#) enables you to authenticate a remote Git server, even if you have a complex authentication pattern like two-factor authentication, Azure Active Directory, or using SSH remote URLs that require an SSH key password for every Git push. GCM Core integrates into the authentication flow for services like GitHub and, once you're authenticated to your hosting provider, requests a new authentication token. It then stores the token securely in the [Windows Credential Manager](#). After the first time, you can use Git to talk to your hosting provider without needing to re-authenticate. It will just access the token in the Windows Credential Manager.

To set up GCM Core for use with a WSL distribution, open your distribution and enter this command:

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

Now any git operation you perform within your WSL distribution will use GCM Core. If you already have credentials cached for a host, it will access them from the credential manager. If not, you'll receive a dialog response requesting your credentials, even if you're in a Linux console.

#### NOTE

If you are using a GPG key for code signing security, you may need to [associate your GPG key with your GitHub email](#).

## Adding a Git Ignore file

We recommend adding a [.gitignore file](#) to your projects. GitHub offers [a collection of useful .gitignore templates](#) with recommended .gitignore file setups organized according to your use-case. For example, here is [GitHub's default gitignore template for a Node.js project](#).

If you choose to [create a new repo using the GitHub website](#), there are check boxes available to initialize your repo with a README file, .gitignore file set up for your specific project type, and options to add a license if you need one.

## Git and VS Code

Visual Studio Code comes with built-in support for Git, including a source control tab that will show your changes and handle a variety of git commands for you. Learn more about [VS Code's Git support](#).

## Git line endings

If you are working with the same repository folder between Windows, WSL, or a container, be sure to set up consistent line endings.

Since Windows and Linux use different default line endings, Git may report a large number of modified files that have no differences aside from their line endings. To prevent this from happening, you can disable line ending conversion using a `.gitattributes` file or globally on the Windows side. See this [VS Code doc about resolving Git line ending issues](#).

## Additional resources

- [WSL & VS Code](#)
- [GitHub Learning Lab: Online courses](#)
- [Git Visualization Tool](#)
- [Git Tools - Credential Storage](#)



# Get started with databases on Windows Subsystem for Linux

4/14/2021 • 9 minutes to read • [Edit Online](#)

This step-by-step guide will help you get started connecting your project in WSL to a database. Get started with MySQL, PostgreSQL, MongoDB, Redis, Microsoft SQL Server, or SQLite.

## Prerequisites

- Running Windows 10, [updated to version 2004](#), **Build 19041** or higher.
- [WSL enabled and installed, and updated to WSL 2](#).
- [Linux distribution installed](#) (Ubuntu 18.04 for our examples).
- Ensure your Ubuntu 18.04 distribution is [running in WSL 2 mode](#). (WSL can run distributions in both v1 or v2 mode.) You can check this by opening PowerShell and entering: `wsl -l -v`

## Differences between database systems

The most [popular choices](#) for a database system include:

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

**MySQL** is an open-source SQL relational database, organizing data into one or more tables in which data types may be related to each other. It is vertically scalable, which means one ultimate machine will do the work for you. It is currently the most widely used of the four database systems.

**PostgreSQL** (sometimes referred to as Postgres) is also an open-source SQL relational database with an emphasis on extensibility and standards compliance. It can handle JSON now too, but it is generally better for structured data, vertical scaling, and ACID-compliant needs like eCommerce and financial transactions.

**Microsoft SQL Server** includes SQL Server on Windows, SQL Server on Linux, and SQL on Azure. These are also relational database management systems set up on servers with primary function of storing and retrieving data as requested by software applications.

**SQLite** is an open-source self-contained, file-based, "serverless" database, known for its portability, reliability, and good performance even in low-memory environments.

**MongoDB** is an open-source NoSQL document database designed to work with JSON and store schema-free data. It is horizontally scalable, which means multiple smaller machines will do the work for you. It's good for flexibility and unstructured data, and caching real-time analytics.

**Redis** is an open-source NoSQL in-memory data structure store. It uses key-value pairs for storage instead of documents. Redis is known for its flexibility, performance, and wide language support. It's flexible enough to be used as a cache or message broker and can use data structures like lists, sets, and hashes.

The sort of database you choose should depend on the type of application you will be using the database with.

We recommend that you look up the advantages and disadvantages of structured and unstructured databases and choose based on your use case.

## Install MySQL

To install MySQL on WSL (Ubuntu 18.04):

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Update your Ubuntu packages: `sudo apt update`
3. Once the packages have updated, install MySQL with: `sudo apt install mysql-server`
4. Confirm installation and get the version number: `mysql --version`

You may also want to run the included security script. This changes some of the less secure default options for things like remote root logins and sample users. To run the security script:

1. Start a MySQL server: `sudo /etc/init.d/mysql start`
2. Start the security script prompts: `sudo mysql_secure_installation`
3. The first prompt will ask whether you'd like to set up the Validate Password Plugin, which can be used to test the strength of your MySQL password. You will then set a password for the MySQL root user, decide whether or not to remove anonymous users, decide whether to allow the root user to login both locally and remotely, decide whether to remove the test database, and, lastly, decide whether to reload the privilege tables immediately.

To open the MySQL prompt, enter: `sudo mysql`

To see what databases you have available, in the MySQL prompt, enter: `SHOW DATABASES;`

To create a new database, enter: `CREATE DATABASE database_name;`

To delete a database, enter: `DROP DATABASE database_name;`

For more about working with MySQL databases, see the [MySQL docs](#).

To work with with MySQL databases in VS Code, try the [MySQL extension](#).

## Install PostgreSQL

To install PostgreSQL on WSL (Ubuntu 18.04):

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Update your Ubuntu packages: `sudo apt update`
3. Once the packages have updated, install PostgreSQL (and the -contrib package which has some helpful utilities) with: `sudo apt install postgresql postgresql-contrib`
4. Confirm installation and get the version number: `psql --version`

There are 3 commands you need to know once PostgreSQL is installed:

- `sudo service postgresql status` for checking the status of your database.
- `sudo service postgresql start` to start running your database.
- `sudo service postgresql stop` to stop running your database.

The default admin user, `postgres`, needs a password assigned in order to connect to a database. To set a password:

1. Enter the command: `sudo passwd postgres`
2. You will get a prompt to enter your new password.

3. Close and reopen your terminal.

To run PostgreSQL with `psql` shell:

1. Start your postgres service: `sudo service postgresql start`
2. Connect to the postgres service and open the psql shell: `sudo -u postgres psql`

Once you have successfully entered the psql shell, you will see your command line change to look like this:

```
postgres=#
```

#### NOTE

Alternatively, you can open the psql shell by switching to the postgres user with: `su - postgres` and then entering the command: `psql`.

To exit postgres=# enter: `\q` or use the shortcut key: Ctrl+D

To see what user accounts have been created on your PostgreSQL installation, use from your WSL terminal:

`psql -c "\du"` ...or just `\du` if you have the psql shell open. This command will display columns: Account User Name, List of Roles Attributes, and Member of role group(s). To exit back to the command line, enter: `q`.

For more about working with PostgreSQL databases, see the [PostgreSQL docs](#).

To work with with PostgreSQL databases in VS Code, try the [PostgreSQL extension](#).

## Install MongoDB

To install MongoDB on WSL (Ubuntu 18.04):

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Update your Ubuntu packages: `sudo apt update`
3. Once the packages have updated, install MongoDB with: `sudo apt-get install mongodb`
4. Confirm installation and get the version number: `mongod --version`

There are 3 commands you need to know once MongoDB is installed:

- `sudo service mongodb status` for checking the status of your database.
- `sudo service mongodb start` to start running your database.
- `sudo service mongodb stop` to stop running your database.

#### NOTE

You might see the command `sudo systemctl status mongodb` used in tutorials or articles. In order to remain lightweight, WSL does not include `systemd` (a service management system in Linux). Instead, it uses SysVinit to start services on your machine. You shouldn't notice a difference, but if a tutorial recommends using `sudo systemctl`, instead use: `sudo /etc/init.d/`. For example, `sudo systemctl status mongodb`, for WSL would be `sudo /etc/init.d/mongodb status` ...or you can also use `sudo service mongodb status`.

To run your Mongo database in a local server:

1. Check the status of your database: `sudo service mongodb status` You should see a [Fail] response, unless you've already started your database.
2. Start your database: `sudo service mongodb start` You should now see an [OK] response.
3. Verify by connecting to the database server and running a diagnostic command:

`mongo --eval 'db.runCommand({ connectionStatus: 1 })'` This will output the current database version, the server address and port, and the output of the status command. A value of `1` for the "ok" field in the response indicates that the server is working.

4. To stop your MongoDB service from running, enter: `sudo service mongod stop`

#### NOTE

MongoDB has several default parameters, including storing data in `/data/db` and running on port 27017. Also, `mongod` is the daemon (host process for the database) and `mongo` is the command-line shell that connects to a specific instance of `mongod`.

VS Code supports working with MongoDB databases via the [Azure CosmosDB extension](#), you can create, manage and query MongoDB databases from within VS Code. To learn more, visit the VS Code docs: [Working with MongoDB](#).

Learn more in the MongoDB docs:

- [Introduction to using MongoDB](#)
- [Create users](#)
- [Connect to a MongoDB instance on a remote host](#)
- [CRUD: Create, Read, Update, Delete](#)
- [Reference Docs](#)

## Install Microsoft SQL Server

To install SQL Server on WSL (Ubuntu 18.04), follow this quickstart: [Install SQL Server and create a database on Ubuntu](#).

To work with Microsoft SQL Server databases in VS Code, try the [MSSQL extension](#).

## Install SQLite

To install SQLite on WSL (Ubuntu 18.04):

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Update your Ubuntu packages: `sudo apt update`
3. Once the packages have updated, install SQLite3 with: `sudo apt install sqlite3`
4. Confirm installation and get the version number: `sqlite3 --version`

To create a test database, called "example.db", enter: `sqlite3 example.db`

To see a list of your SQLite databases, enter: `.databases`

To see the status of your database, enter: `.dbinfo ?DB?`

To exit the SQLite prompt, enter: `.exit`

For more information about working with a SQLite database, see the [SQLite docs](#).

To work with SQLite databases in VS Code, try the [SQLite extension](#).

## Install Redis

To install Redis on WSL (Ubuntu 18.04):

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Update your Ubuntu packages: `sudo apt update`
3. Once the packages have updated, install Redis with: `sudo apt install redis-server`
4. Confirm installation and get the version number: `redis-server --version`

To start running your Redis server: `sudo service redis-server start`

Check to see if redis is working (redis-cli is the command line interface utility to talk with Redis): `redis-cli ping`  
this should return a reply of "PONG".

To stop running your Redis server: `sudo service redis-server stop`

For more information about working with a Redis database, see the [Redis docs](#).

To work with Redis databases in VS Code, try the [Redis extension](#).

## See services running and set up profile aliases

To see the services that you currently have running on your WSL distribution, enter: `service --status-all`

Typing out `sudo service mongodb start` or `sudo service postgres start` and `sudo -u postgres psql` can get tedious. However, you could consider setting up aliases in your `.profile` file on WSL to make these commands quicker to use and easier to remember.

To set up your own custom alias, or shortcut, for executing these commands:

1. Open your WSL terminal and enter `cd ~` to be sure you're in the root directory.
2. Open the `.profile` file, which controls the settings for your terminal, with the terminal text editor, Nano:  
`sudo nano .profile`
3. At the bottom of the file (don't change the `# set PATH` settings), add the following:

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

This will allow you to enter `start-pg` to start running the postgresql service and `run-pg` to open the psql shell. You can change `start-pg` and `run-pg` to whatever names you want, just be careful not to overwrite a command that postgres already uses!

4. Once you've added your new aliases, exit the Nano text editor using **Ctrl+X** -- select `y` (Yes) when prompted to save and Enter (leaving the file name as `.profile`).
5. Close and re-open your WSL terminal, then try your new alias commands.

## Additional resources

- [Setting up your development environment on Windows 10](#)

# Get started with Docker remote containers on WSL

## 2

4/14/2021 • 9 minutes to read • [Edit Online](#)

This step-by-step guide will help you get started developing with remote containers by **setting up Docker Desktop for Windows with WSL 2** (Windows Subsystem for Linux, version 2).

Docker Desktop for Windows is available for free and provides a development environment for building, shipping, and running dockerized apps. By enabling the WSL 2 based engine, you can run both Linux and Windows containers in Docker Desktop on the same machine.

## Overview of Docker containers

Docker is a tool used to create, deploy, and run applications using containers. Containers enable developers to package an app with all of the parts it needs (libraries, frameworks, dependencies, etc) and ship it all out as one package. Using a container ensures that the app will run the same regardless of any customized settings or previously installed libraries on the computer running it that could differ from the machine that was used to write and test the app's code. This permits developers to focus on writing code without worrying about the system that code will be run on.

Docker containers are similar to virtual machines, but don't create an entire virtual operating system. Instead, Docker enables the app to use the same Linux kernel as the system that it's running on. This allows the app package to only require parts not already on the host computer, reducing the package size and improving performance.

Continuous availability, using Docker containers with tools like [Kubernetes](#), is another reason for the popularity of containers. This enables multiple versions of your app container to be created at different times. Rather than needing to take down an entire system for updates or maintenance, each container (and it's specific microservices) can be replaced on the fly. You can prepare a new container with all of your updates, set up the container for production, and just point to the new container once it's ready. You can also archive different versions of your app using containers and keep them running as a safety fallback if needed.

To learn more, checkout the [Introduction to Docker containers](#) on Microsoft Learn.

## Prerequisites

- Ensure your machine is running Windows 10, [updated to version 2004](#), **Build 18362** or higher.
- [Enable WSL, install a Linux distribution, and update to WSL 2.](#)
- [Download and install the Linux kernel update package.](#)
- [Install Visual Studio Code](#) (*optional*). This will provide the best experience, including the ability to code and debug inside a remote Docker container and connected to your Linux distribution.
- [Install Windows Terminal](#) (*optional*). This will provide the best experience, including the ability to customize and open multiple terminals in the same interface (including Ubuntu, Debian, PowerShell, Azure CLI, or whatever you prefer to use).
- [Sign up for a Docker ID at Docker Hub](#) (*optional*).

## NOTE

WSL can run distributions in both WSL version 1 or WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`. Ensure that the your distribution is set to use WSL 2 by entering: `wsl --set-version <distro> 2`. Replace `<distro>` with the distro name (e.g. Ubuntu 18.04).

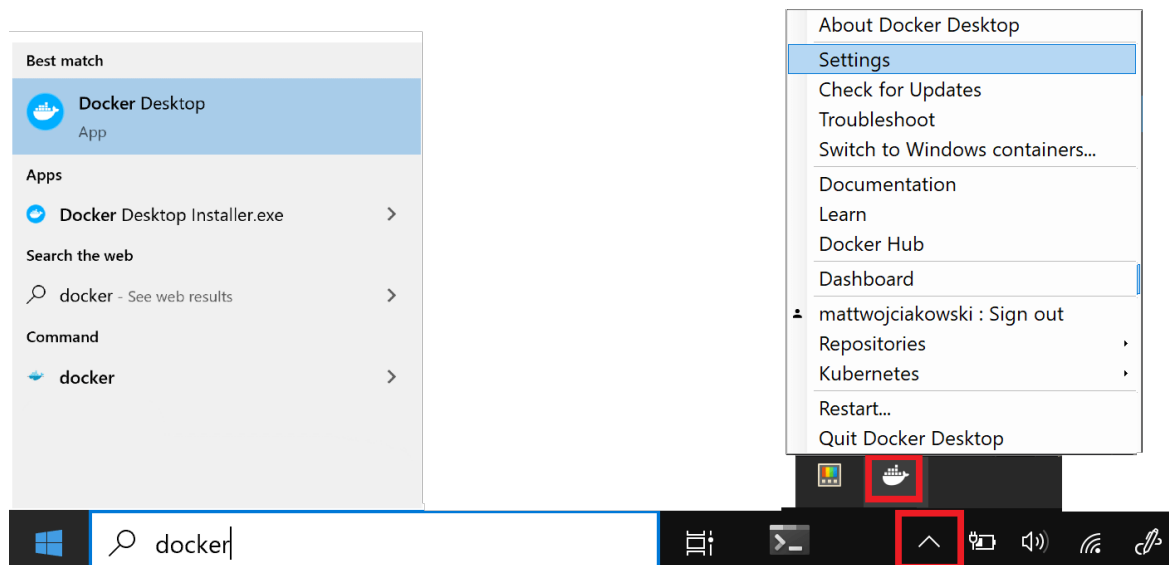
In WSL version 1, due to fundamental differences between Windows and Linux, the Docker Engine couldn't run directly inside WSL, so the Docker team developed an alternative solution using Hyper-V VMs and LinuxKit. However, since WSL 2 now runs on a Linux kernel with full system call capacity, Docker can fully run in WSL 2. This means that Linux containers can run natively without emulation, resulting in better performance and interoperability between your Windows and Linux tools.

## Install Docker Desktop

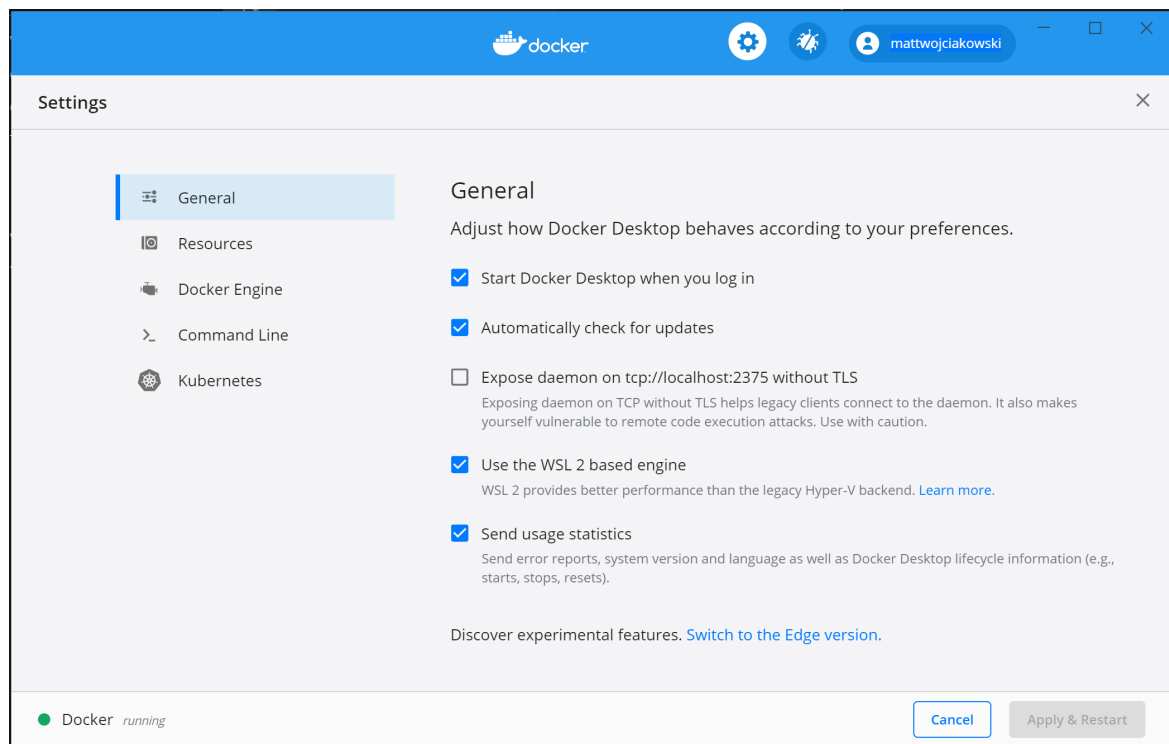
With the WSL 2 backend supported in Docker Desktop for Windows, you can work in a Linux-based development environment and build Linux-based containers, while using Visual Studio Code for code editing and debugging, and running your container in the Microsoft Edge browser on Windows.

To install Docker (after already [installing WSL 2](#)):

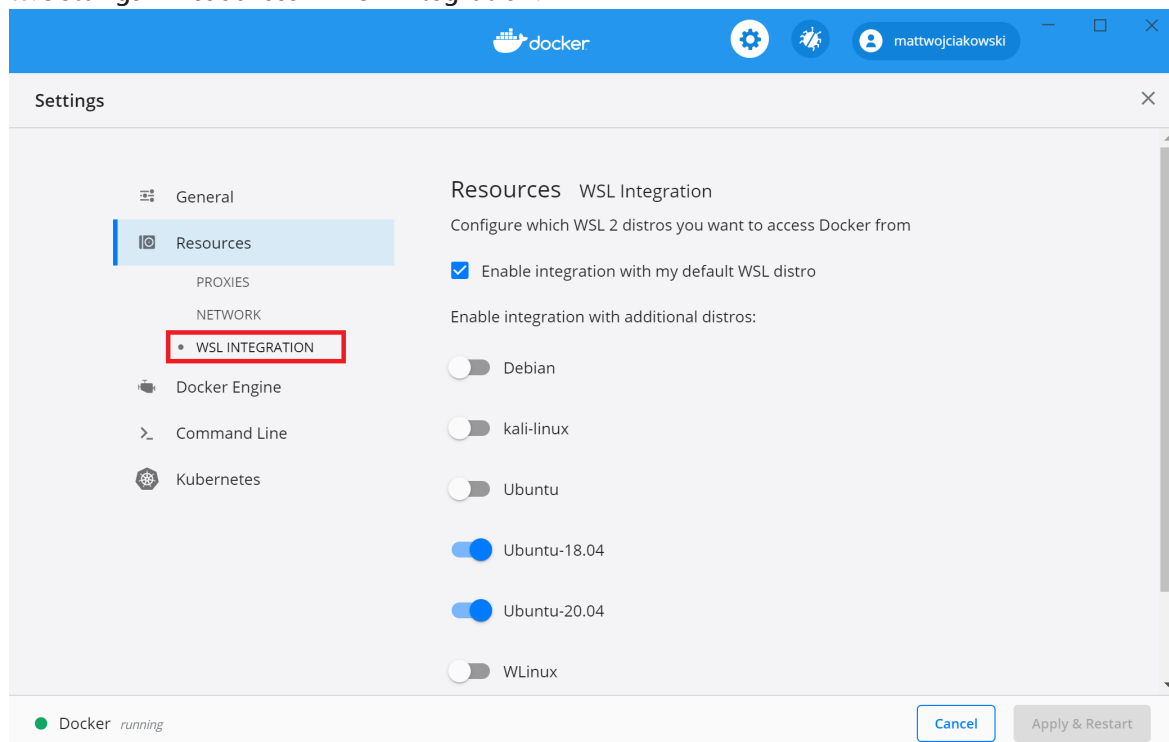
1. Download [Docker Desktop](#) and follow the installation instructions.
2. Once installed, start Docker Desktop from the Windows Start menu, then select the Docker icon from the hidden icons menu of your taskbar. Right-click the icon to display the Docker commands menu and select "Settings".



3. Ensure that "Use the WSL 2 based engine" is checked in **Settings > General**.



4. Select from your installed WSL 2 distributions which you want to enable Docker integration on by going to: **Settings > Resources > WSL Integration**.



5. To confirm that Docker has been installed, open a WSL distribution (e.g. Ubuntu) and display the version and build number by entering: `docker --version`
6. Test that your installation works correctly by running a simple built-in Docker image using: `docker run hello-world`



### TIP

Here are a few helpful Docker commands to know:

- List the commands available in the Docker CLI by entering: `docker`
- List information for a specific command with: `docker <COMMAND> --help`
- List the docker images on your machine (which is just the hello-world image at this point), with:  
`docker image ls --all`
- List the containers on your machine, with: `docker container ls --all` or `docker ps -a` (without the -a show all flag, only running containers will be displayed)
- List system-wide information regarding the Docker installation, including statistics and resources (CPU & memory) available to you in the WSL 2 context, with: `docker info`

## Develop in remote containers using VS Code

To get started developing apps using Docker with WSL 2, we recommend using VS Code, along with the Remote-WSL extension and Docker extension.

- [Install the VS Code Remote-WSL extension](#). This extension enables you to open your Linux project running on WSL in VS Code (no need to worry about pathing issues, binary compatibility, or other cross-OS challenges).
- [Install the VS code Remote-Containers extension](#). This extension enables you to open your project folder or repo inside of a container, taking advantage of Visual Studio Code's full feature set to do your development work within the container.
- [Install the VS Code Docker extension](#). This extension adds the functionality to build, manage, and deploy containerized applications from inside VS Code. (You need the Remote-Container extension to actually use the container as your dev environment.)

Let's use Docker to create a development container for an existing app project.

1. For this example, I'll use the source code from my [Hello World tutorial for Django](#) in the Python development environment set up docs. You can skip this step if you prefer to use your own project source code. To download my HelloWorld-Django web app from GitHub, open a WSL terminal (Ubuntu for example) and enter: `git clone https://github.com/mattwojo/helloworld-django.git`

### NOTE

Always store your code in the same file system that you're using tools in. This will result in faster file access performance. In this example, we are using a Linux distro (Ubuntu) and want to store our project files on the WSL file system `\\wsl\\`. Storing project files on the Windows file system would significantly slow things down when using Linux tools in WSL to access those files.

2. From your WSL terminal, change directories to the source code folder for this project:

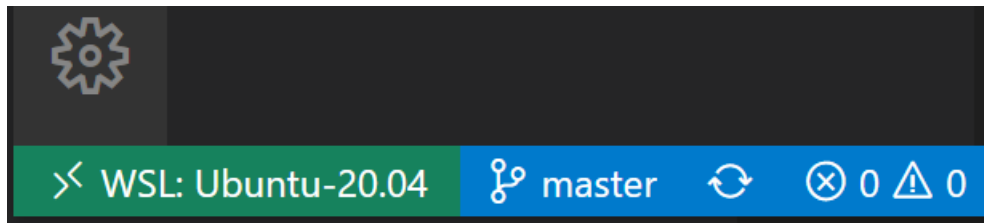
```
cd helloworld-django
```

3. Open the project in VS Code running on the local Remote-WSL extension server by entering:

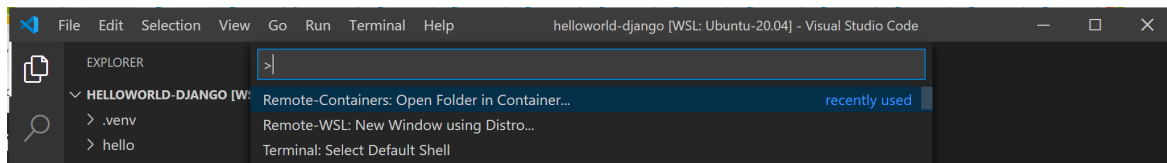
```
code .
```

Confirm that you are connected to your WSL Linux distro by checking the green remote indicator in the

bottom-left corner of your VS Code instance.

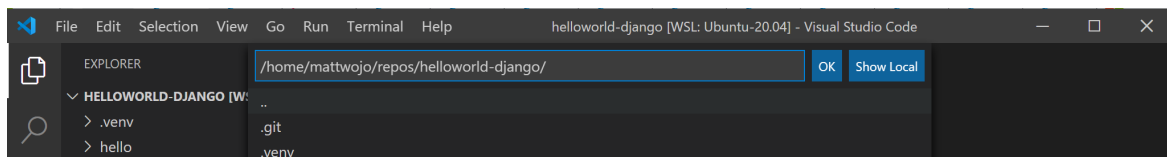


- From the VS Code command palette (Ctrl + Shift + P), enter: **Remote-Containers: Open Folder in Container...** If this command doesn't display as you begin to type it, check to ensure that you've installed the Remote Container extension linked above.

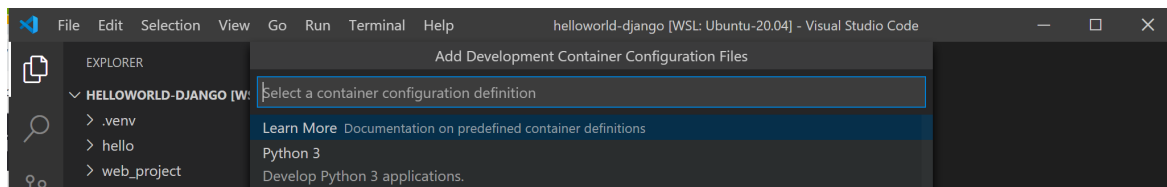


- Select the project folder that you wish to containerize. In my case, this is

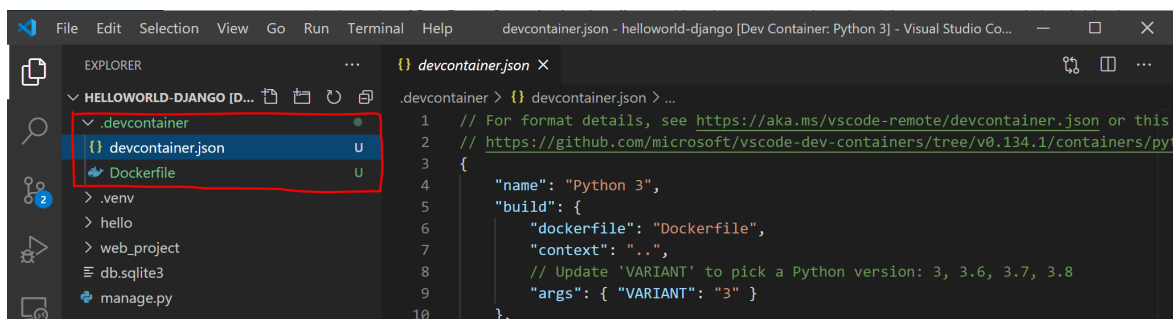
```
\\wsl\Ubuntu-20.04\home\mattwojo\repos\helloworld-django\
```



- A list of container definitions will appear, since there is no DevContainer configuration in the project folder (repo) yet. The list of container configuration definitions that appears is filtered based on your project type. For my Django project, I'll select Python 3.

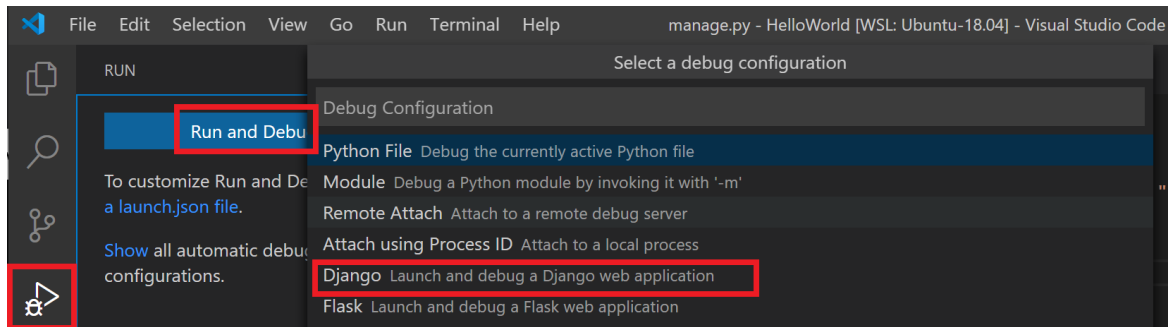


- A new instance of VS Code will open, begin building our new image, and once the build completed, will start our container. You will see that a new `.devcontainer` folder has appeared with container configuration information inside a `Dockerfile` and `devcontainer.json` file.

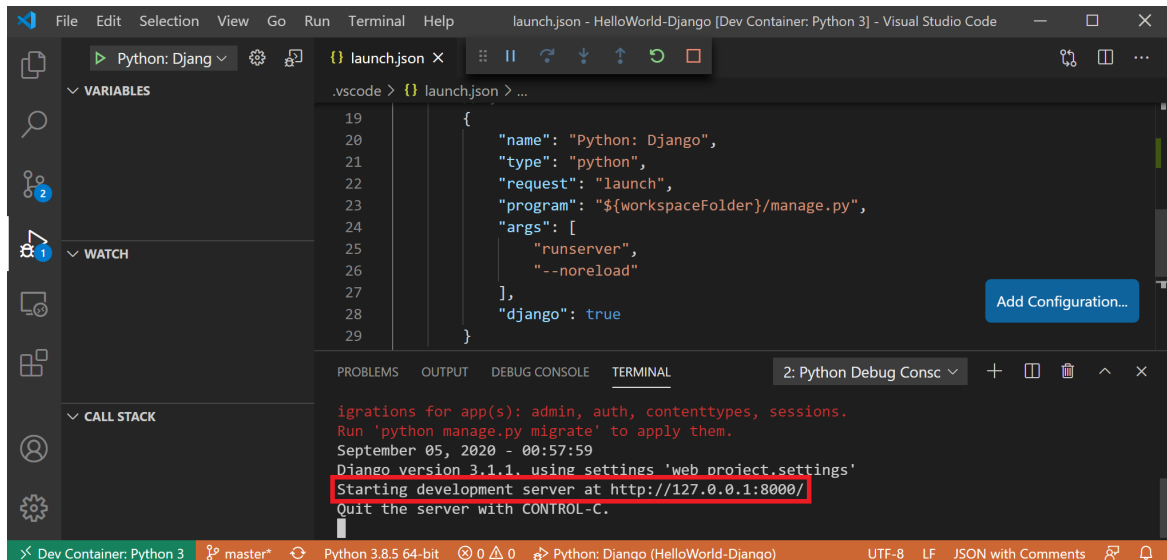


- To confirm that your project is still connected to both WSL and within a container, open the VS Code integrated terminal (Ctrl + Shift + ~). Check the operating system by entering: `uname` and the Python version with: `python3 --version`. You can see that the uname came back as "Linux", so you are still connected to the WSL 2 engine, and Python version number will be based on the container config that may differ from the Python version installed on your WSL distribution.
- To run and debug your app inside of the container using Visual Studio Code, first open the Run menu (Ctrl+Shift+D or select the tab on the far left menu bar). Then select **Run and Debug** to select a debug

configuration and choose the configuration that best suites your project (in my example, this will be "Django"). This will create a `launch.json` file in the `.vscode` folder of your project with instructions on how to run your app.



10. From inside VS Code, select **Run > Start debugging** (or just press the **F5** key). This will open a terminal inside VS Code and you should see a result saying something like: "Starting development server at <http://127.0.0.1:8000/> Quit the server with CONTROL-C." Hold down the Control key and select the address displayed to open your app in your default web browser and see your project running inside of its container.



You have now successfully configured a remote development container using Docker Desktop, powered by the WSL 2 backend, that you can code in, build, run, deploy, or debug using VS Code!

## Troubleshooting

### WSL docker context deprecated

If you were using an early Tech Preview of Docker for WSL, you may have a Docker context called "wsl" that is now deprecated and no longer used. You can check with the command: `docker context ls`. You can remove this "wsl" context to avoid errors with the command: `docker context rm wsl` as you want to use the default context for both Windows and WSL2.

Possible errors you might encounter with this deprecated wsl context include:

```
docker wsl open ../pipe/docker_wsl: The system cannot find the file specified. OR
error during connect: Get http://%2F%2Fpipe%2Fdocker_wsl/v1.40/images/json?all=1: open
../pipe/docker_wsl: The system cannot find the file specified.
```

For more on this issue, see [How to set up Docker within Windows System for Linux \(WSL2\) on Windows 10](#).

### Trouble finding docker image storage folder

Docker creates two distro folders to store data:

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

You can find these folders by opening your WSL Linux distribution and entering: `explorer.exe .` to view the folder in Windows File Explorer. Enter: `\\wsl\<distro name>\mnt\wsl` replacing `<distro name>` with the name of your distribution (ie. Ubuntu-20.04) to see these folders.

Find more on locating docker storage locations in WSL, see this [issue from the WSL repo](#) or this [StackOverflow post](#).

For more help with general troubleshooting issues in WSL, see the [Troubleshooting](#) doc.

## Additional resources

- [Docker docs: Best practices for Docker Desktop with WSL 2](#)
- [Feedback for Docker Desktop for Windows: File an issue](#)
- [VS Code Blog: Guidelines for choosing a development environment](#)
- [VS Code Blog: Using Docker in WSL 2](#)
- [VS Code Blog: Using Remote Containers in WSL 2](#)
- [Hanselminutes Podcast: Making Docker lovely for Developers with Simon Ferquel](#)

# GPU accelerated machine learning training in the Windows Subsystem for Linux

4/14/2021 • 2 minutes to read • [Edit Online](#)

Support for GPU compute, the #1 most requested WSL feature, is now available for preview via the Windows Insider program. [Read the blog post](#).

## What is GPU compute?

Leveraging GPU acceleration for compute-intensive tasks is generally referred to as "GPU compute". GPU computing leverages the GPU (graphics processing unit) to accelerate math heavy workloads and uses its parallel processing to complete the required calculations faster, in many cases, than utilizing only a CPU. This parallelization enables significant processing speed improvements for these math heavy workloads then when running on a CPU. Training machine learning models is a great example in which GPU compute can significantly accelerate the time to complete this computationally expensive task.

## Install and set up

Learn more about WSL 2 support and how to start training machine learning models in the [GPU Accelerated Training guide](#) inside the DirectML docs. This guide covers:

- Guidance for beginners or students to set up TensorFlow with DirectML
- Guidance for professionals to start running their existing CUDA ML workflows

# Run Linux GUI apps on the Windows Subsystem for Linux (preview)

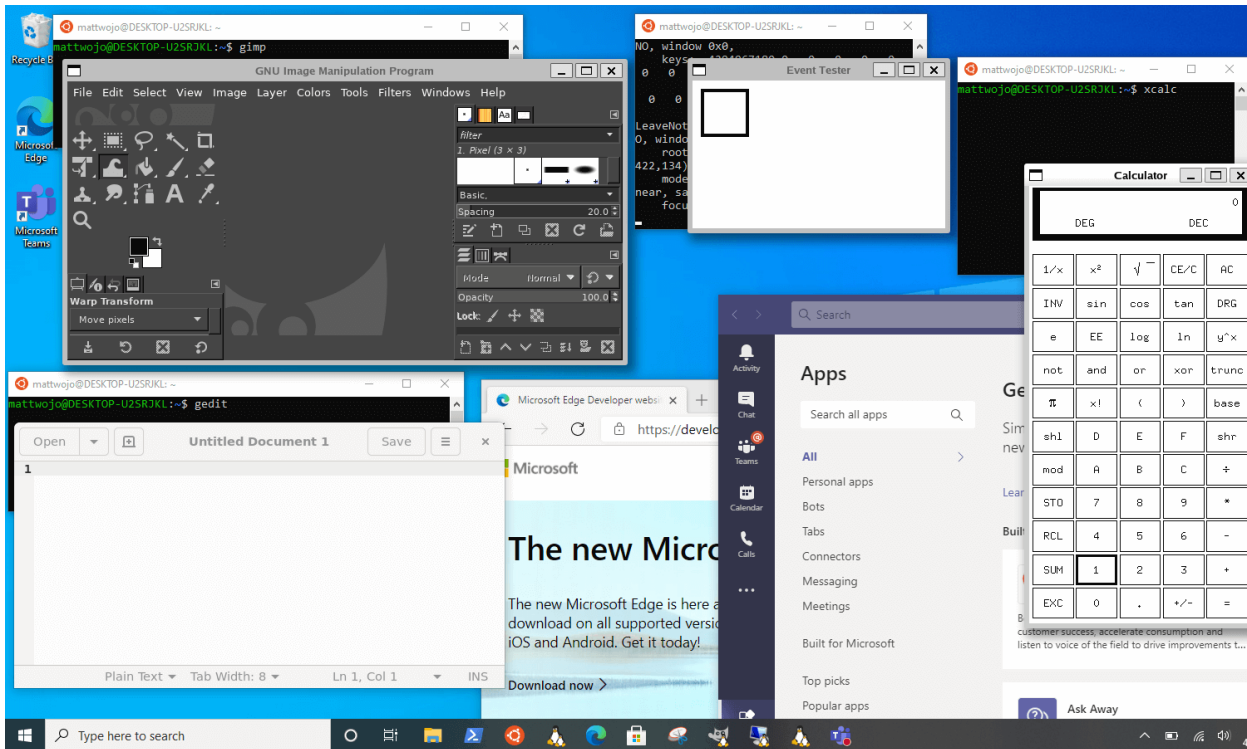
5/25/2021 • 4 minutes to read • [Edit Online](#)

You can now preview Windows Subsystem for Linux (WSL) support for running Linux GUI applications (X11 and Wayland) on Windows in a fully integrated desktop experience.

WSL 2 enables Linux GUI applications to feel native and natural to use on Windows.

- Launch Linux apps from the Windows Start menu
- Pin Linux apps to the Windows task bar
- Use alt-tab to switch between Linux and Windows apps
- Cut + Paste across Windows and Linux apps

You can now integrate both Windows and Linux applications into your workflow for a seamless desktop experience.



## Install support for Linux GUI apps

### Prerequisites

- [Windows 10 Insider Preview build 21362+](#)

Support for Linux GUI apps via WSL will become generally available in the next major release of Windows. To access to the preview build of Windows and try running Linux GUI apps now, you'll need to join the [Windows Insider Program](#) and run a Windows 10 Insider Preview build from the dev channel.

- **Installed driver for vGPU**

To run Linux GUI apps, you should first install the preview driver matching your system below. This will enable you to use a virtual GPU (vGPU) so you can benefit from hardware accelerated OpenGL rendering.

- [Intel GPU driver for WSL](#)
- [AMD GPU driver for WSL](#)
- [NVIDIA GPU driver for WSL](#)

### Fresh install - No prior WSL installation

If you have not already followed the steps to [install WSL](#), you can use the new simplified install command.

1. Open a command prompt with administrator privileges.

*Select **Start**, type **PowerShell**, right-click **Windows PowerShell**, and then select **Run as administrator**.*

2. Run this command and reboot your machine when prompted:

```
ws1 --install -d Ubuntu
```

3. Once your machine has finished rebooting, installation will continue and you will be asked to enter a username and password. This will be your Linux credential for the Ubuntu distribution.

You're now ready to begin using Linux GUI apps on WSL!

### Existing WSL install

If you already have WSL installed on your machine, you can update to the latest version that includes Linux GUI support by running the update command from an elevated command prompt.

1. Select **Start**, type **PowerShell**, right-click **Windows PowerShell**, and then select **Run as administrator**.
2. Enter the WSL update command:

```
ws1 --update
```

3. You will need to restart WSL for the update to take effect. You can restart WSL by running the shutdown command in PowerShell.

```
ws1 --shutdown
```

#### NOTE

Linux GUI apps are only supported with WSL 2 and will not work with a Linux distribution configured for WSL 1. Read about [how to change your distribution from WSL 1 to WSL 2](#).

## Run Linux GUI apps

You can run the following commands from your Linux terminal to download and install these popular Linux applications. If you are using a different distribution than Ubuntu, it may use a different package manager than apt. Once the Linux application is installed, you can find it in your **Start** menu under the distribution name. For example: `Ubuntu -> Microsoft Edge`.

### Update the packages in your distribution

```
sudo apt update
```

## Install Gedit

Gedit is the default text editor of the GNOME desktop environment.

```
sudo apt install gedit -y
```

To launch your bashrc file in the editor, enter: `gedit ~/.bashrc`

## Install GIMP

GIMP is a free and open-source raster graphics editor used for image manipulation and image editing, free-form drawing, transcoding between different image file formats, and more specialized tasks.

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

To launch, enter: `gimp`

## Install Nautilus

Nautilus, also known as GNOME Files, is the file manager for the GNOME desktop. (Similiar to Windows File Explorer).

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

To launch, enter: `nautilus`

## Install VLC

VLC is a free and open source cross-platform multimedia player and framework that plays most multimedia files.

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

To launch, enter: `vlc`

## Install X11 apps

X11 is the Linux windowing system and this is a miscellaneous collection of apps and tools that ship with it, such as the xclock, xcalc calculator, xclipboard for cut and paste, xev for event testing, etc. See the [x.org docs](https://x.org/docs) for more info.

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

To launch, enter the name of the tool you would like to use. For example:

- `xcalc` , `xclock` , `xeyes`

## Install Google Chrome for Linux

To install the Google Chrome for Linux:

1. Change directories into the temp folder: `cd /tmp`

2. Use wget to download it:

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

3. Get the current stable version: `sudo dpkg -i google-chrome-stable_current_amd64.deb`

4. Fix the package: `sudo apt install --fix-broken -y`

5. Configure the package: `sudo dpkg -i google-chrome-stable_current_amd64.deb`



To launch, enter: `google-chrome`

### Install Microsoft Teams for Linux

To install Microsoft Teams for Linux:

1. Change directories into the temp folder: `cd /tmp`

2. Use curl to download the package:

```
sudo curl -L -o "./teams.deb" "https://teams.microsoft.com/downloads/desktopurl?env=production&plat=linux&arch=x64&download=true&linuxArchiveType=deb"
```

3. Use apt to install it: `sudo apt install ./teams.deb -y`

To launch, enter: `teams`

### Install Microsoft Edge browser for Linux

To install Microsoft Edge for Linux:

1. Use curl to download the package:

```
sudo curl https://packages.microsoft.com/repos/edge/pool/main/m/microsoft-edge-dev/microsoft-edge-dev_91.0.852.0-1_amd64.deb -o /tmp/edge.deb
```

2. Use apt to install it: `sudo apt install /tmp/edge.deb -y`

To launch, enter: `microsoft-edge`

# Windows interoperability with Linux

3/18/2021 • 6 minutes to read • [Edit Online](#)

The Windows Subsystem for Linux (WSL) is continuously improving integration between Windows and Linux. You can:

- Run Windows tools (ie. notepad.exe) from a Linux command line (ie. Ubuntu).
- Run Linux tools (ie. grep) from a Windows command line (ie. PowerShell).
- Share environment variables between Linux and Windows. (Build 17063+)

## NOTE

If you're running Creators Update (Oct 2017, Build 16299) or Anniversary Update (Aug 2016, Build 14393), jump to the [Earlier versions of Windows 10](#).

## Run Linux tools from a Windows command line

Run Linux binaries from the Windows Command Prompt (CMD) or PowerShell using `wsl <command>` (or `wsl.exe <command>`).

For example:

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

Binaries invoked in this way:

- Use the same working directory as the current CMD or PowerShell prompt.
- Run as the WSL default user.
- Have the same Windows administrative rights as the calling process and terminal.

The Linux command following `wsl` (or `wsl.exe`) is handled like any command run in WSL. Things such as `sudo`, piping, and file redirection work.

Example using `sudo` to update your default Linux distribution:

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

Your default Linux distribution user name will be listed after running this command and you will be asked for your password. After entering your password correctly, your distribution will download updates.

## Mixing Linux and Windows commands

Here are a few examples of mixing Linux and Windows commands using PowerShell.

To use the Linux command `ls -la` to list files and the PowerShell command `findstr` to filter the results for words containing "git", combine the commands:

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

To use the PowerShell command `dir` to list files and the Linux command `grep` to filter the results for words containing "git", combine the commands:

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

To use the Linux command `ls -la` to list files and the PowerShell command `> out.txt` to print that list to a text file named "out.txt", combine the commands:

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

The commands passed into `wsl.exe` are forwarded to the WSL process without modification. File paths must be specified in the WSL format.

To use the Linux command `ls -la` to list files in the `/proc/cpuinfo` Linux file system path, using PowerShell:

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

To use the Linux command `ls -la` to list files in the `C:\Program Files` Windows file system path, using PowerShell:

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

## Run Windows tools from Linux

WSL can run Windows tools directly from the WSL command line using `[tool-name].exe`. For example, `notepad.exe`.

Applications run this way have the following properties:

- Retain the working directory as the WSL command prompt (for the most part -- exceptions are explained below).
- Have the same permission rights as the WSL process.
- Run as the active Windows user.
- Appear in the Windows Task Manager as if directly executed from the CMD prompt.

Windows executables run in WSL are handled similarly to native Linux executables -- piping, redirects, and even backgrounding work as expected.

To run the Windows tool `ipconfig.exe`, use the Linux tool `grep` to filter the "IPv4" results, and use the Linux tool `cut` to remove the column fields, from a Linux distribution (for example, Ubuntu) enter:

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

Let's try an example mixing Windows and Linux commands. Open your Linux distribution (ie. Ubuntu) and create a text file: `touch foo.txt`. Now use the Linux command `ls -la` to list the direct files and their creation details, plus the Windows PowerShell tool `findstr.exe` to filter the results so only your `foo.txt` file shows in the results:

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

Windows tools must include the file extension, match the file case, and be executable. Non-executables including

batch scripts. CMD native commands like `dir` can be run with `cmd.exe /C` command.

For example, list the contents of your Windows files system C:\ directory, by entering:

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

Or use the `ping` command to send an echo request to the microsoft.com website:

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

Parameters are passed to the Windows binary unmodified. As an example, the following command will open

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

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

This will also work:

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

## Share environment variables between Windows and WSL

WSL and Windows share a special environment variable, `WSLENV`, created to bridge Windows and Linux distributions running on WSL.

Properties of `WSLENV` variable:

- It is shared; it exists in both Windows and WSL environments.
- It is a list of environment variables to share between Windows and WSL.
- It can format environment variables to work well in Windows and WSL.
- It can assist in the flow between WSL and Win32.

### NOTE

Prior to 17063, only Windows environment variable that WSL could access was `PATH` (so you could launch Win32 executables from under WSL). Starting in 17063, `WSLENV` begins being supported. `WSLENV` is case sensitive.

## WSLENV flags

There are four flags available in `WSLENV` to influence how the environment variable is translated.

`WSLENV` flags:

- `/p` - translates the path between WSL/Linux style paths and Win32 paths.
- `/l` - indicates the environment variable is a list of paths.
- `/u` - indicates that this environment variable should only be included when running WSL from Win32.
- `/w` - indicates that this environment variable should only be included when running Win32 from WSL.

Flags can be combined as needed.

[Read more about WSL](#), including FAQs and examples of setting the value of `WSLENV` to a concatenation of other pre-defined environment vars, each suffixed with a slash followed by flags to specify how the value should

be translated and passing variables with a script. This article also includes an example for setting up a dev environment with the [Go programming language](#), configured to share a GOPATH between WSL and Win32.

## Disable interoperability

Users may disable the ability to run Windows tools for a single WSL session by running the following command as root:

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

To re-enable Windows binaries, exit all WSL sessions and re-run bash.exe or run the following command as root:

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

Disabling interop will not persist between WSL sessions -- interop will be enabled again when a new session is launched.

## Earlier versions of Windows 10

There are several differences for the interoperability commands on earlier Windows 10 versions. If you're running a Creators Update (Oct 2017, Build 16299), or Anniversary Update (Aug 2016, Build 14393) version of Windows 10, we recommend you [update to the latest Windows version](#), but if that's not possible, we have outlined some of the interop differences below.

Summary:

- `bash.exe` has been replaced with `ws1.exe`.
- `-c` option for running a single command isn't needed with `ws1.exe`.
- Windows path is included in the WSL `$PATH`.
- The process for disabling interop is unchanged.

Linux commands can be run from the Windows Command Prompt or from PowerShell, but for early Windows versions, you may need to use the `bash` command. For example:

```
C:\temp> bash -c "ls -la"
```

Things such as input, piping, and file redirection work as expected.

The WSL commands passed into `bash -c` are forwarded to the WSL process without modification. File paths must be specified in the WSL format and care must be taken to escape relevant characters. Example:

```
C:\temp> bash -c "ls -la /proc/cpuinfo"
```

Or...

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

When calling a Windows tool from a WSL distribution in an earlier version of Windows 10, you will need to specify the directory path. For example, from your WSL command line, enter:

```
/mnt/c/Windows/System32/notepad.exe
```

In WSL, these executables are handled similar to native Linux executables. This means adding directories to the Linux path and piping between commands works as expected. For example:

```
export PATH=$PATH:/mnt/c/Windows/System32
```

Or

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

The Windows binary must include the file extension, match the file case, and be executable. Non-executables including batch scripts and command like `dir` can be run with `/mnt/c/Windows/System32/cmd.exe /C` command. For example:

```
/mnt/c/Windows/System32/cmd.exe /C dir
```

## Additional resources

- [WSL blog post on interoperability from 2016](#)

# WSL commands and launch configurations

6/3/2021 • 11 minutes to read • [Edit Online](#)

## Ways to run WSL

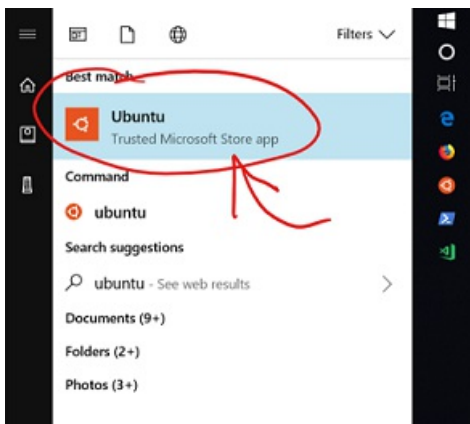
There are several ways to run a Linux distribution with WSL once it's [installed](#).

1. Open your Linux distribution by visiting the Windows Start menu and typing the name of your installed distributions. For example: "Ubuntu".
2. From Windows Command Prompt or PowerShell, enter the name of your installed distribution. For example:  
`ubuntu`
3. From Windows Command Prompt or PowerShell, to open your default Linux distribution inside your current command line, enter: `ws1.exe`.
4. From Windows Command Prompt or PowerShell, to open your default Linux distribution inside your current command line, enter: `ws1 [command]`.

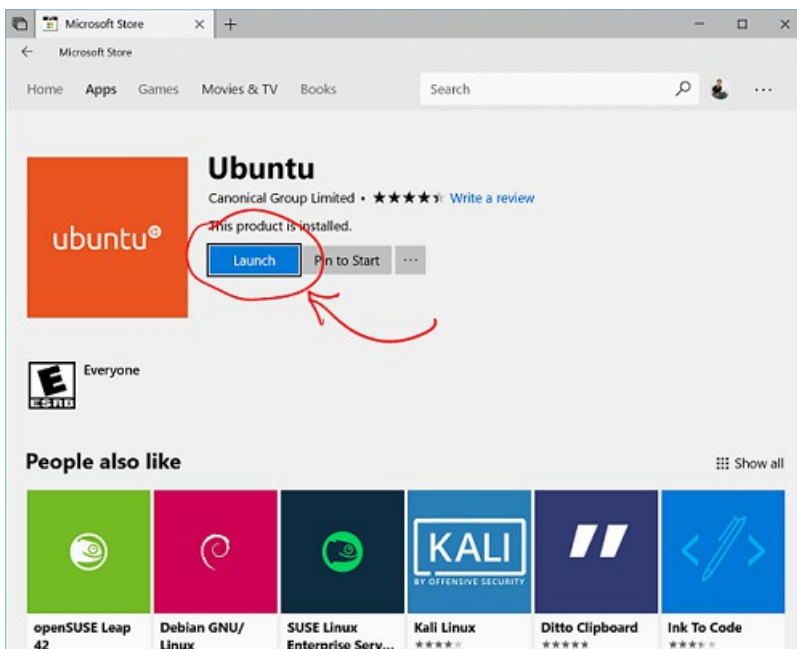
Which method you should use depends on what you're doing. If you've opened a WSL command line within a Windows Prompt or PowerShell window and want to exit, enter the command: `exit`.

## Launch WSL by distribution

Running a distribution using its distro-specific application launches that distribution in its own console window.



It is the same as clicking "Launch" in the Microsoft store.



You can also run the distribution from the command line by running `[distribution].exe`.

The disadvantage of running a distribution from the command line in this way is that it will automatically change your working directory from the current directory to the distribution's home directory.

#### Example: (using PowerShell)

```
PS C:\Users\sarah> pwd

Path
----
C:\Users\sarah

PS C:\Users\sarah> ubuntu

scooley@scooley-elmer:~$ pwd
/home/scooley
scooley@scooley-elmer:~$ exit
logout

PS C:\Users\sarah>
```

#### WSL and wsl [command]

The best way to run WSL from the command line is using `wsl.exe`.

#### Example: (using PowerShell)

```
PS C:\Users\sarah> pwd

Path
----
C:\Users\sarah

PS C:\Users\sarah> wsl

scooley@scooley-elmer:/mnt/c/Users/sarah$ pwd
/mnt/c/Users/sarah
```

Not only does `wsl` keep the current working directory in place, it lets you run a single command along side Windows commands.



## Example: (using PowerShell)

```
PS C:\Users\sarah> Get-Date

Sunday, March 11, 2018 7:54:05 PM

PS C:\Users\sarah> wsl
scooley@scooley-elmer:/mnt/c/Users/sarah$ date
Sun Mar 11 19:55:47 DST 2018
scooley@scooley-elmer:/mnt/c/Users/sarah$ exit
logout

PS C:\Users\sarah> wsl date
Sun Mar 11 19:56:57 DST 2018
```

## Example: (using PowerShell)

```
PS C:\Users\sarah> Get-VM

Name                State CPUUsage(%) MemoryAssigned(M) Uptime   Status
----                -
Server17093         Off   0             0                  00:00:00 Opera...
Ubuntu              Off   0             0                  00:00:00 Opera...
Ubuntu (bionic)     Off   0             0                  00:00:00 Opera...
Windows             Off   0             0                  00:00:00 Opera...

PS C:\Users\sarah> Get-VM | wsl grep "Ubuntu"
Ubuntu              Off   0             0                  00:00:00 Opera...
Ubuntu (bionic)     Off   0             0                  00:00:00 Opera...
PS C:\Users\sarah>
```

# Managing multiple Linux Distributions

In Windows 10 Version 1903 [and later](#), you can use `ws1.exe` to manage your distributions in the Windows Subsystem for Linux (WSL), including listing available distributions, setting a default distribution, and uninstalling distributions.

Each Linux distribution independently manages its own configurations. To see distribution-specific commands, run `[distro.exe] /?`. For example `ubuntu /?`.

## List distributions

```
ws1 -l , wsl --list
```

Lists available Linux distributions available to WSL. If a distribution is listed, it's installed and ready to use.

```
ws1 --list --all
```

 Lists all distributions, including ones that aren't currently usable. They may be in the process of installing, uninstalling, or are in a broken state.

```
ws1 --list --running
```

 Lists all distributions that are currently running.

## Set a default distribution

The default WSL distribution is the one that runs when you run `ws1` on a command line.

```
ws1 -s <DistributionName> , wsl --setdefault <DistributionName>
```

Sets the default distribution to `<DistributionName>`.

### Example: (using PowerShell)

`wsl -s Ubuntu` would set my default distribution to Ubuntu. Now when I run `wsl npm init` it will run in Ubuntu. If I run `wsl` it will open an Ubuntu session.

## Unregister and reinstall a distribution

While Linux distributions can be installed through the Microsoft store, they can't be uninstalled through the store. WSL Config allows distributions to be unregistered/uninstalled.

Unregistering also allows distributions to be reinstalled.

**Caution:** Once unregistered, all data, settings, and software associated with that distribution will be permanently lost. Reinstalling from the store will install a clean copy of the distribution.

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

Unregisters the distribution from WSL so it can be reinstalled or cleaned up.

For example: `wsl --unregister Ubuntu` would remove Ubuntu from the distributions available in WSL. When I run `wsl --list` it will not be listed.

To reinstall, find the distribution in the Microsoft store and select "Launch".

## Run as a specific user

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

Run WSL as the specified user. Please note that user must exist inside of the WSL distribution.

## Change the default user for a distribution

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

Change the default user that for your distribution log-in. The user has to already exist inside the distribution in order to become the default user.

For example: `ubuntu config --default-user johndoe` would change the default user for the Ubuntu distribution to the "johndoe" user.

### NOTE

If you are having trouble figuring out the name of your distribution, see [List distributions](#) for the command to list the official name of the installed distributions.

## Run a specific distribution

```
wsl -d <DistributionName> , wsl --distribution <DistributionName>
```

Run a specified distribution of WSL, can be used to send commands to a specific distribution without having to change your default.

## Managing multiple Linux Distributions in earlier Windows versions

In Windows 10 prior to version 1903, the WSL Config ( `wslconfig.exe` ) command-line tool should be used to manage Linux distributions running on the Windows Subsystem for Linux (WSL). It lets you list available distributions, set a default distribution, and uninstall distributions.

While WSL Config is helpful for settings that span or coordinate distributions, each Linux distribution independently manages its own configurations. To see distribution-specific commands, run `[distro.exe] /?`. For example `ubuntu /?`.

To see all available options for `wslconfig`, run: `wslconfig /?`

```
wslconfig.exe
Performs administrative operations on Windows Subsystem for Linux

Usage:
  /l, /list [/all] - Lists registered distributions.
  /all - Optionally list all distributions, including distributions that
        are currently being installed or uninstalled.
  /s, /setdefault <DistributionName> - Sets the specified distribution as the default.
  /u, /unregister <DistributionName> - Unregisters a distribution.
```

To list distributions, use:

```
wslconfig /list
```

Lists available Linux distributions available to WSL. If a distribution is listed, it's installed and ready to use.

```
wslconfig /list /all
```

Lists all distributions, including ones that aren't currently usable. They may be in the process of installing, uninstalling, or are in a broken state.

To set a default distribution that runs when you run `wsl` on a command line:

```
wslconfig /setdefault <DistributionName> Sets the default distribution to <DistributionName>.
```

### Example: (using PowerShell)

`wslconfig /setdefault Ubuntu` would set my default distribution to Ubuntu. Now when I run `wsl npm init` it will run in Ubuntu. If I run `wsl` it will open an Ubuntu session.

To unregister and reinstall a distribution:

```
wslconfig /unregister <DistributionName>
```

Unregisters the distribution from WSL so it can be reinstalled or cleaned up.

For example: `wslconfig /unregister Ubuntu` would remove Ubuntu from the distributions available in WSL. When I run `wslconfig /list` it will not be listed.

To reinstall, find the distribution in the Microsoft store and select "Launch".

## Configure per distro launch settings with `wslconf`

Available in Windows Build 17093 and later

Automatically configure functionality in WSL that will be applied every time you launch the subsystem using `wsl.conf`. This includes automount options and network configuration.

`wsl.conf` is located in each Linux distribution in `/etc/wsl.conf`. If the file is not there, you can create it yourself. WSL will detect the existence of the file and will read its contents. If the file is missing or malformed (that is, improper markup formatting), WSL will continue to launch as normal.

Here is a sample `wsl.conf` file you could add into your distributions:

```
# Enable extra metadata options by default
[automount]
enabled = true
root = /windir/
options = "metadata,umask=22,fmask=11"
mountFsTab = false

# Enable DNS - even though these are turned on by default, we'll specify here just to be explicit.
[network]
generateHosts = true
generateResolvConf = true
```

When launching multiple Linux shells for the same distribution, you must wait until the Linux subsystem stops running, this can take approximately 8 seconds after closing the last instance of the distribution shell. If you launch a distribution (ie. Ubuntu), modify the wsl.conf file, close the distribution, and then re-launch it. You might assume that your changes to the wsl.conf file have immediately gone into effect. This is not currently the case as the subsystem could still be running. You must wait ~8 seconds for the subsystem to stop before relaunching in order to give enough time for your changes to be picked up. You can check to see whether your Linux distribution (shell) is still running after closing it by using PowerShell with the command: `wsl --list --running`. If no distributions are running, you will receive the response: "There are no running distributions." You can now restart the distribution to see your wsl.conf updates applied.

## Configuration Options

In keeping with .ini conventions, keys are declared under a section.

WSL supports four sections: `automount`, `network`, `interop`, and `user`.

### automount

Section: `[automount]`

KEY	VALUE	DEFAULT	NOTES
enabled	boolean	true	<code>true</code> causes fixed drives (i.e. <code>C:/</code> or <code>D:/</code> ) to be automatically mounted with DrvFs under <code>/mnt</code> . <code>false</code> means drives won't be mounted automatically, but you could still mount them manually or via <code>fstab</code> .
mountFsTab	boolean	true	<code>true</code> sets <code>/etc/fstab</code> to be processed on WSL start. <code>/etc/fstab</code> is a file where you can declare other filesystems, like an SMB share. Thus, you can mount these filesystems automatically in WSL on start up.

KEY	VALUE	DEFAULT	NOTES
root	String	<code>/mnt/</code>	Sets the directory where fixed drives will be automatically mounted. For example, if you have a directory in WSL at <code>/windir/</code> and you specify that as the root, you would expect to see your fixed drives mounted at <code>/windir/c</code>
options	comma-separated list of values	empty string	This value is appended to the default DrvFs mount options string. <b>Only DrvFs-specific options can be specified.</b> Options that the mount binary would normally parse into a flag are not supported. If you want to explicitly specify those options, you must include every drive for which you want to do so in <code>/etc/fstab</code> .

By default, WSL sets the uid and gid to the value of the default user (in Ubuntu distro, the default user is created with uid=1000,gid=1000). If the user specifies a gid or uid option explicitly via this key, the associated value will be overwritten. Otherwise, the default value will always be appended.

**Note:** These options are applied as the mount options for all automatically mounted drives. To change the options for a specific drive only, use `/etc/fstab` instead.

#### Mount options

Setting different mount options for Windows drives (DrvFs) can control how file permissions are calculated for Windows files. The following options are available:

KEY	DESCRIPTION	DEFAULT
uid	The User ID used for the owner of all files	The default User ID of your WSL distro (On first installation this defaults to 1000)
gid	The Group ID used for the owner of all files	The default group ID of your WSL distro (On first installation this defaults to 1000)
umask	An octal mask of permissions to exclude for all files and directories	000
fmask	An octal mask of permissions to exclude for all files	000
dmask	An octal mask of permissions to exclude for all directories	000

KEY	DESCRIPTION	DEFAULT
metadata	Whether metadata is added to Windows files to support Linux system permissions	disabled
case	Determines directories treated as case sensitive and whether new directories created with WSL will have the flag set. See <a href="#">Per-directory case sensitivity and WSL</a> for a detailed explanation of the options.	<code>off</code>

**Note:** The permission masks are put through a logical OR operation before being applied to files or directories.

#### network

Section label: `[network]`

KEY	VALUE	DEFAULT	NOTES
generateHosts	boolean	<code>true</code>	<code>true</code> sets WSL to generate <code>/etc/hosts</code> . The <code>hosts</code> file contains a static map of hostnames corresponding IP address.
generateResolvConf	boolean	<code>true</code>	<code>true</code> set WSL to generate <code>/etc/resolv.conf</code> . The <code>resolv.conf</code> contains a DNS list that are capable of resolving a given hostname to its IP address.

#### interop

Section label: `[interop]`

These options are available in Insider Build 17713 and later.

KEY	VALUE	DEFAULT	NOTES
enabled	boolean	<code>true</code>	Setting this key will determine whether WSL will support launching Windows processes.
appendWindowsPath	boolean	<code>true</code>	Setting this key will determine whether WSL will add Windows path elements to the \$PATH environment variable.

#### user

Section label: `[user]`

These options are available in Build 18980 and later.

KEY	VALUE	DEFAULT	NOTES
default	string	The initial username created on first run	Setting this key specifies which user to run as when first starting a WSL session.

## Configure global options with .wslconfig

Available in Windows Build 19041 and later

You can configure global WSL options by placing a `.wslconfig` file into the root directory of your users folder:

`C:\Users\<yourUserName>\.wslconfig`. Many of these files are related to WSL 2, please keep in mind you may need to run `wsl --shutdown` to shut down the WSL 2 VM and then restart your WSL instance for these changes to take affect.

Here is a sample .wslconfig file:

```
[wsl2]
kernel=C:\\temp\\myCustomKernel
memory=4GB # Limits VM memory in WSL 2 to 4 GB
processors=2 # Makes the WSL 2 VM use two virtual processors
```

This file can contain the following options:

### WSL 2 Settings

Section label: `[wsl2]`

These settings affect the VM that powers any WSL 2 distribution.

KEY	VALUE	DEFAULT	NOTES
kernel	string	The Microsoft built kernel provided inbox	An absolute Windows path to a custom Linux kernel.
memory	size	50% of total memory on Windows or 8GB, whichever is less; on builds before 20175: 80% of your total memory on Windows	How much memory to assign to the WSL 2 VM.
processors	number	The same number of processors on Windows	How many processors to assign to the WSL 2 VM.
localhostForwarding	boolean	<code>true</code>	Boolean specifying if ports bound to wildcard or localhost in the WSL 2 VM should be connectable from the host via localhost:port.
kernelCommandLine	string	Blank	Additional kernel command line arguments.
swap	size	25% of memory size on Windows rounded up to the nearest GB	How much swap space to add to the WSL 2 VM, 0 for no swap file.

KEY	VALUE	DEFAULT	NOTES
swapFile	string	%USERPROFILE%\AppData\Local\Temp\swap.vhdx	An absolute Windows path to the swap virtual hard disk.

- Note: This value is true for Windows Build 19041 and may be different in Windows builds in the Insiders program

Entries with the `path` value must be Windows paths with escaped backslashes, e.g: `C:\\Temp\\myCustomKernel`

Entries with the `size` value must be a size followed by a unit, for example `8GB` or `512MB` .



# File Permissions for WSL

4/14/2021 • 4 minutes to read • [Edit Online](#)

This page details how Linux file permissions are interpreted across the Windows Subsystem for Linux, especially when accessing resources inside of Windows on the NT file system. This documentation assumes a basic understanding of the [Linux file system permissions structure](#) and the [umask command](#).

When accessing Windows files from WSL the file permissions are either calculated from Windows permissions, or are read from metadata that has been added to the file by WSL. This metadata is not enabled by default.

## WSL metadata on Windows files

When metadata is enabled as a mount option in WSL, extended attributes on Windows NT files can be added and interpreted to supply Linux file system permissions.

WSL can add four NTFS extended attributes:

ATTRIBUTE NAME	DESCRIPTION
\$LXUID	User Owner ID
\$LXGID	Group Owner ID
\$LXMOD	File mode (File systems permission octals and type, e.g: 0777)
\$LXDEV	Device, if it is a device file

Additionally, any file that is not a regular file or directory (e.g: symlinks, FIFOs, block devices, unix sockets, and character devices) also have an NTFS [reparse point](#). This makes it much faster to determine the kind of file in a given directory without having to query its extended attributes.

## File Access Scenarios

Below is a description of how permissions are determined when accessing files in different ways using the Windows Subsystem for Linux.

### Accessing Files in the Windows drive file system (DrvFS) from Linux

These scenarios occur when you are accessing your Windows files from WSL, most likely via `/mnt/c`.

#### Reading file permissions from an existing Windows file

The result depends on if the file already has existing metadata.

**DrvFS file does not have metadata (default)**

If the file has no metadata associated with it then we translate the effective permissions of the Windows user to read/write/execute bits and set them to the this as the same value for user, group, and other. For example, if your Windows user account has read and execute access but not write access to the file then this will be shown as `r-x` for user, group and other. If the file has the 'Read Only' attribute set in Windows then we do not grant write access in Linux.

**The file has metadata**

If the file has metadata present, we simply use those metadata values instead of translating effective permissions of the Windows user.

### Changing file permissions on an existing Windows file using chmod

The result depends on if the file already has existing metadata.

**chmod file does not have metadata (default)**

Chmod will only have one effect, if you remove all the write attributes of a file then the 'read only' attribute on the Windows file will be set, since this is the same behaviour as CIFS (Common Internet File System) which is the SMB (Server Message Block) client in Linux.

**chmod file has metadata**

Chmod will change or add metadata depending on the file's already existing metadata.

Please keep in mind that you cannot give yourself more access than what you have on Windows, even if the metadata says that is the case. For example, you could set the metadata to display that you have write permissions to a file using `chmod 777`, but if you tried to access that file you would still not be able to write to it. This is thanks to interoperability, as any read or write commands to Windows files are routed through your Windows user permissions.

### Creating a file in DriveFS

The result depends on if metadata is enabled.

**Metadata is not enabled (default)**

The Windows permissions of the newly created file will be the same as if you created the file in Windows without a specific security descriptor, it will inherit the parent's permissions.

**Metadata is enabled**

The file's permission bits are set to follow the Linux umask, and the file will be saved with metadata.

### Which Linux user and Linux group owns the file?

The result depends on if the file already has existing metadata.

**User file does not have metadata (default)**

In the default scenario, when automounting Windows drives, we specify that the user ID (UID) for any file is set to the user ID of your WSL user and the group ID (GID) is set to the principal group ID of your WSL user.

**User file has metadata**

The UID and GID specified in the metadata is applied as the user owner and group owner of the file.

### Accessing Linux files from Windows using `\\wsl$`

Accessing Linux files via `\\wsl$` will use the default user of your WSL distribution. Therefore any Windows app accessing Linux files will have the same permissions as the default user.

### Creating a new file

The default umask is applied when creating a new file inside of a WSL distribution from Windows. The default umask is `022`, or in other words it allows all permissions except write permissions to groups and others.

### Accessing files in the Linux root file system from Linux

Any files created, modified, or accessed in the Linux root file system follow standard Linux conventions, such as applying the umask to a newly created file.

## Configuring file permissions

You can configure your file permissions inside of your Windows drives using the mount options in `wsl.conf`. The mount options allow you to set `umask`, `dmask` and `fmask` permissions masks. The `umask` is applied to all files, the `dmask` is applied just to directories and the `fmask` is applied just to files. These permission masks are then put through a logical OR operation when being applied to files, e.g: If you have a `umask` value of `023` and an `fmask` value of `022` then the resulting permissions mask for files will be `023`.

Please see the [Configure per distro launch settings with wslconf](#) article for instructions on how to do this.

# Windows Subsystem for Linux for Enterprise

4/14/2021 • 2 minutes to read • [Edit Online](#)

As an administrator or manager, you may require all developers to use the same approved software. This consistency helps to create a well-defined work environment. The Windows Subsystem for Linux aids in this consistency by allowing you to import and export custom WSL images from one machine to the next. Read the guide below to learn more about:

- [Creating a custom WSL image](#)
- [Distributing a WSL image](#)
- [Update and patch Linux distributions and packages](#)
- [Enterprise security and control options](#)

## Creating a custom WSL image

What is commonly referred to as an "image", is simply a snapshot of your software and its components saved to a file. In the case of the Windows Subsystem for Linux, your image would consist of the subsystem, its distributions, and whatever software and packages are installed on the distribution.

To begin creating your WSL image, first [install the Windows Subsystem for Linux](#).

Once installed, use The Microsoft Store for Business to download and install the Linux distribution that's right for you. Create an account with the [Microsoft Store for Business](#)

### Exporting your WSL image

Export your custom WSL image by running `wsl --export <Distro> <FileName>`, which will wrap your image in a tar file and make it ready for distribution on to other machines.

## Distributing your WSL image

Distribute the WSL image from a share or storage device by running `wsl --import`

`<Distro> <InstallLocation> <FileName>`, which will import the specified tar file as a new distribution.

## Update and patch Linux distributions and packages

Using Linux configuration manager tools is strongly recommended for monitoring and managing Linux user space. There are a host of Linux configuration managers to choose from. Check out this [blog post](#) on how to install Puppet in WSL 2.

## Enterprise security and control options

Currently, WSL offers limited control mechanisms in regard to modifying the user experience in an Enterprise scenario. Enterprise features continue in development however, below are the areas of supported and unsupported features. To request a new feature not covered in this list, file an issue in our [GitHub repo](#).

### Supported

- Sharing an approved image internally using `wsl --import` and `wsl --export`
- Creating your own WSL distro for your Enterprise using the [WSL Distro Launcher repo](#)

Here's a list of features for which we don't yet have support for, but are investigating.

## **Currently unsupported**

Below is a list of commonly asked features that are currently unsupported within WSL. These requests are on our backlog and we are investigating ways to add them.

- Synchronizing the user inside WSL with the Windows user on the host machine
- Managing updates and patching of the Linux distributions and packages using Windows tools
- Having Windows update also update WSL distro contents
- Controlling which distributions users in your Enterprise can access
- Running mandatory services (logging or monitoring) inside of WSL
- Monitoring Linux instances using Windows configuration manager tools such as SCCM or Intune
- McAfee support

# Import any Linux distribution to use with WSL

6/8/2021 • 3 minutes to read • [Edit Online](#)

You can use any Linux distribution inside of the Windows Subsystem for Linux (WSL), even if it is not available in the [Microsoft Store](#), by importing it with a tar file.

This article shows how to import the Linux distribution, [CentOS](#), for use with WSL by obtaining its tar file using a Docker container. This process can be applied to import any Linux distribution.

## Obtain a tar file for the distribution

First you'll need to obtain a tar file that contains all the Linux binaries for the distribution.

You can obtain a tar file in a variety of ways, two of which include:

- Download a provided tar file. You can find an example for Alpine in the "Mini Root Filesystem" section of the [Alpine Linux downloads](#) site.
- Find a Linux distribution container and export an instance as a tar file. The example below will show this process using the [CentOS container](#).

### Obtaining a tar file for CentOS example

In this example, we'll use Docker inside of a WSL distribution to obtain the tar file for CentOS.

#### Prerequisites

- You must have [WSL enabled with a Linux distribution installed running WSL 2](#).
- You must have [Docker Desktop for Windows installed with the WSL 2 engine enabled and integration checked](#) for the distribution you will use in the next steps.

#### Export the tar from a container

1. Open the command line (Bash) for a Linux distribution that you've already installed from the Microsoft Store (Ubuntu in this example).
2. Start the Docker service:

```
sudo service docker start
```

3. Run the CentOS container inside Docker:

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

4. Grab the CentOS container ID using grep and awk:

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

5. Export the container ID to a tar file on your mounted c-drive:

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

```
PowerShell x Ubuntu x + v - □ x
~ $ sudo service docker start
* Starting Docker: docker [ OK ]
~ $ docker run -t centos bash ls /
~ $ dockerContainerID=$(docker container ls -a | grep -i centos | awk '{prin
t $1}')
~ $ docker export $dockerContainerID > /mnt/c/temp/centos.tar
~ $
```

This process exports the CentOS tar file from the Docker container so that we can now import it for use locally with WSL.

## Import the tar file into WSL

Once you have a tar file ready, you can import it using the command:

```
ws1 --import <Distro> <InstallLocation> <FileName> .
```

### Importing CentOS example

To import the CentOS distribution tar file into WSL:

1. Open PowerShell and ensure that you have a folder created where you'd like the distribution to be stored.

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

2. Use the command `ws1 --import <DistroName> <InstallLocation> <InstallTarFile>` to import the tar file.

```
ws1 --import CentOS E:\wslDistroStorage\CentOS .\centos.tar
```

3. Use the command `ws1 -l -v` to check which distributions you have installed.

```
PowerShell | Ubuntu
PS C:\Users\crloewen> cd C:\temp
PS C:\temp>
PS C:\temp> mkdir E:\wslDistroStorage\CentOS

Directory: E:\wslDistroStorage

Mode                LastWriteTime         Length Name
----                -
d-----           2/17/2021   2:06 PM             CentOS

PS C:\temp> wsl --import CentOS E:\wslDistroStorage\CentOS\ .\centos.tar
PS C:\temp> wsl -l -v
    NAME                STATE          VERSION
*  Ubuntu                Running         2
   CentOS                Stopped        2
   Debian                Stopped        2
PS C:\temp> wsl -d CentOS
[root@loewen-dev temp]# cat /etc/centos-release
CentOS Linux release 8.3.2011
[root@loewen-dev temp]# exit
logout
PS C:\temp> |
```

4. Finally, use the command `wsl -d CentOS` to run your newly imported CentOS Linux distribution.

## Add WSL specific components like a default user

By default when using `--import`, you are always started as the root user. You can set up your own user account, but note that the set up process will vary slightly based on each different Linux distribution.

To set up user account with the CentOS distribution we just imported, first open PowerShell and boot into CentOS, using the command:

```
wsl -d CentOS
```

Next, open your CentOS command line. Use this command to install sudo and password setting tools into CentOS, create a user account, and set it as the default user. In this example, the username will be 'caloewen'.

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

You must now quit out of that instance and ensure that all WSL instances are terminated. Start your distribution again to see your new default user by running this command in PowerShell:

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

You will now see `[caloewen@loewen-dev]$` as the output based on this example.

```
caloewen@loewen-dev:/mnt/c/  × + ∨
PS C:\temp> wsl -d CentOS
[root@loewen-dev temp]# myUsername=caloewen
[root@loewen-dev temp]# adduser -G wheel $myUsername
[root@loewen-dev temp]# echo -e "[user]\ndefault=$myUsername" >> /etc/wsl.conf
[root@loewen-dev temp]# passwd $myUsername
Changing password for user caloewen.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@loewen-dev temp]# logout
PS C:\temp> wsl --shutdown
PS C:\temp> wsl -d CentOS
[caloewen@loewen-dev temp]$ sudo yum update
[sudo] password for caloewen:
Failed to set locale, defaulting to C.UTF-8
Last metadata expiration check: 0:10:19 ago on Wed Feb 17 14:35:56 2021.
Dependencies resolved.
Nothing to do.
Complete!
[caloewen@loewen-dev temp]$ |
```

To learn more about configuring WSL settings, see [Launch commands & configurations](#).

## Use a custom Linux distribution

You can create your own customized Linux distribution, packaged as a UWP app, that will behave exactly like the WSL distributions available in the Microsoft Store. To learn how, see [Creating a Custom Linux Distribution for WSL](#).



# Creating a Custom Linux Distribution for WSL

3/18/2021 • 2 minutes to read • [Edit Online](#)

Use our open source WSL sample to build WSL distro packages for the Microsoft Store and/or to create custom Linux distro packages for sideloading. You can find the [distro launcher repo](#) on GitHub.

This project enables:

- Linux distribution maintainers to package and submit a Linux distribution as an appx that runs on WSL
- Developers to create custom Linux distributions that can be sideloaded onto their dev machine

## Background

We distribute Linux distros for WSL as UWP applications through the Microsoft Store. You can install those applications that will then run on WSL - the subsystem that sits in the Windows kernel. This delivery mechanism has many benefits as discussed in an [earlier blog post](#).

## Sideloading a Custom Linux Distro Package

You can create a custom Linux distro package as an application to sideload on your personal machine. Please note that your custom package would not be distributed through the Microsoft Store unless you submit as a distribution maintainer. To set up your machine to sideload apps, you will need to enable this in the system settings under "For Developers". Be sure to either have developer mode, or sideload apps selected

## For Linux Distro Maintainers

To submit to the Store, you will need to work with us to receive publishing approval. If you are a Linux distribution owner interested in adding your distribution to the Microsoft Store, please contact [wslpartners@microsoft.com](mailto:wslpartners@microsoft.com).

## Getting Started

Follow the instructions on the [Distro Launcher GitHub repo](#) to create a custom Linux distro package.

## Team Blogs

- [Open Sourcing a WSL Sample for Linux Distribution Maintainers and Sideloaded Custom Linux Distributions](#)
- [Command-Line blog](#)

## Provide Feedback

- [Distro Launcher GitHub repo](#)
- [GitHub issue tracker for WSL](#)

# Get started mounting a Linux disk in WSL 2 (preview)

6/8/2021 • 5 minutes to read • [Edit Online](#)

If you want to access a Linux disk format that isn't supported by Windows, you can use WSL 2 to mount your disk and access its content. This tutorial will cover the steps to identify the disk and partition to attach to WSL2, how to mount them, and how to access them.

## NOTE

Administrator access is required to attach a disk to WSL 2. The WSL 2 `mount` command does not support mounting a disk (or partitions that belong to the disk) that is currently in use. `wsl --mount` always attaches the entire disk even if only a partition is requested. You can't mount the Windows installation disk.

## Prerequisites

You will need to be on Windows 10 Build 20211 or higher to access this feature. You can join the [Windows Insiders Program](#) to get the latest preview builds.

## Mounting an unpartitioned disk

In this simplest case, if you have a disk that doesn't have any partitions, you can mount it directly using the `wsl --mount` command. First you need to identify the disk.

### Identify the disk

To list the available disks in Windows, run:

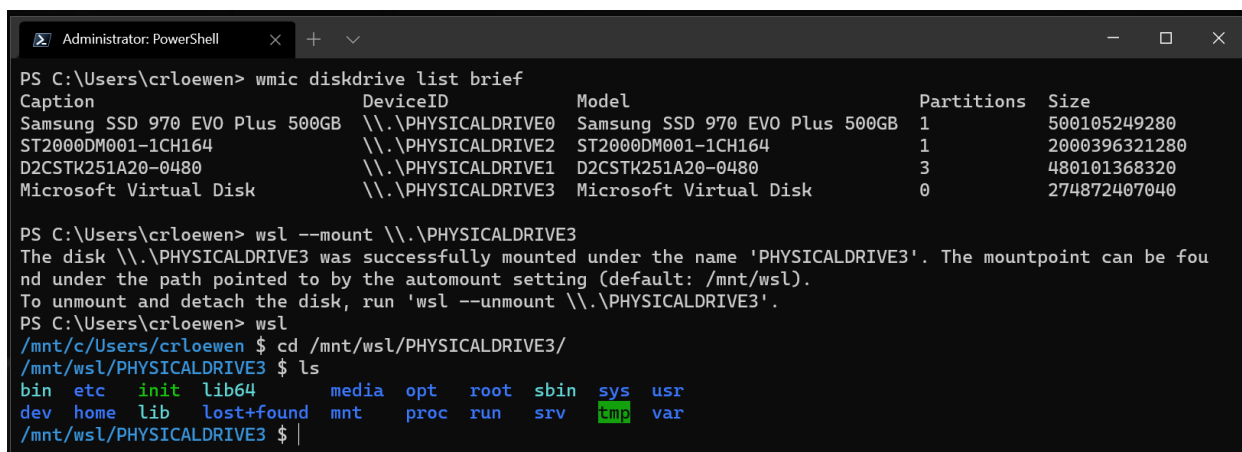
```
wmic diskdrive list brief
```

The disks paths are available under the 'DeviceID' columns. Usually under the `\\.\PHYSICALDRIVE*` format.

### Mount the disk

Then in Powershell you can mount the disk using the Disk path discovered above.

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



```
PS C:\Users\crloewen> wmic diskdrive list brief
Caption                                DeviceID                                Model                                Partitions  Size
-----                                -
Samsung SSD 970 EVO Plus 500GB        \\.\PHYSICALDRIVE0                    Samsung SSD 970 EVO Plus 500GB        1           500105249280
ST2000DM001-1CH164                    \\.\PHYSICALDRIVE2                    ST2000DM001-1CH164                    1           2000396321280
D2CSTK251A20-0480                     \\.\PHYSICALDRIVE1                    D2CSTK251A20-0480                    3           480101368320
Microsoft Virtual Disk                 \\.\PHYSICALDRIVE3                    Microsoft Virtual Disk                 0           274872407040

PS C:\Users\crloewen> wsl --mount \\.\PHYSICALDRIVE3
The disk \\.\PHYSICALDRIVE3 was successfully mounted under the name 'PHYSICALDRIVE3'. The mountpoint can be found under the path pointed to by the automount setting (default: /mnt/wsl).
To unmount and detach the disk, run 'wsl --unmount \\.\PHYSICALDRIVE3'.
PS C:\Users\crloewen> wsl
/mnt/c/Users/crloewen $ cd /mnt/wsl/PHYSICALDRIVE3/
/mnt/wsl/PHYSICALDRIVE3 $ ls
bin  etc  init  lib64  media  opt  root  sbin  sys  usr
dev  home  lib  lost+found  mnt  proc  run  srv  tmp  var
/mnt/wsl/PHYSICALDRIVE3 $ |
```

# Mounting a partitioned disk

If you have a disk that you aren't sure what file format it is in, or what partitions it has, you can follow the steps below to mount it.

## Identify the disk

To list the available disks in Windows, run:

```
wmic diskdrive list brief
```

The disks paths are available under the 'DeviceID' columns. Usually under the `\\.\PHYSICALDRIVE*` format

## List and select the partitions to mount in WSL 2

Once the disk is identified, run:

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

This will make the disk available in WSL 2.

Once attached, the partition can be listed by running the following command inside WSL 2:

```
lsblk
```

This will display the available block devices and their partitions.

Inside Linux, a block device is identified as `/dev/<Device><Partition>`. For example, `/dev/sdb3`, is the partition number 3 of disk `sdb`.

Example output:

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	

## Identifying the filesystem type

If you don't know the type of filesystem of a disk or partition, you can use this command:

```
blkid <BlockDevice>
```

This will output the detected filesystem type (under the `TYPE=<Filesystem>` format).

## Mount the selected partitions

Once you have identified the partitions you want to mount, run this command on each partition:

```
wsl --mount <DiskPath> --partition <PartitionNumber> --type <Filesystem>
```

#### NOTE

If you wish to mount the entire disk as a single volume (i.e. if the disk isn't partitioned), `--partition` can be omitted.

If omitted, the default filesystem type is "ext4".

### Access the disk content

Once mounted, the disk can be accessed under the path pointed to by the config value: `automount.root`. The default value is `/mnt/wsl`.

From Windows, the disk can be accessed from File Explorer by navigating to: `\\wsl$\\<Distro>\\<Mountpoint>` (pick any Linux distribution).

## Unmount the disk

If you want to unmount and detach the disk from WSL 2, run:

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

## Mount a VHD in WSL

You can also mount virtual hard disk files (VHD) into WSL using `wsl --mount`. To do this, you first need to mount the VHD into Windows using the `Mount-VHD` command in Windows. Be sure to run this command with administrator privileges. Below is an example where we use this command, and also output the disk path. Be sure to replace `<pathToVHD>` with your actual VHD path.

```
Write-Output "\\.\PhysicalDrive$((Mount-VHD -Path <pathToVHD> -PassThru | Get-Disk).Number)"
```

You can use the output above to obtain the disk path for this VHD and mount that into WSL following the instructions in the previous section.

You can also use this technique to mount and interact with the virtual hard disks of other WSL distros, as each WSL 2 distro is stored via a virtual hard disk file called: `ext4.vhdx`. By default the VHDs for WSL 2 distros are stored in this path: `C:\Users\[user]\AppData\Local\Packages\[distro]\LocalState\[distroPackageName]`, please exercise caution accessing these system files, this is a power user workflow. Make sure to run `wsl --shutdown` before interacting with this disk to ensure the disk is not in use.

```
Administrator: PowerShell
PS E:\wslDistroStorage\CustomWSLDistro> ls

Directory: E:\wslDistroStorage\CustomWSLDistro

Mode                LastWriteTime         Length Name
----                -
-a---             2/17/2021  4:25 PM         401604608 ext4.vhdx

PS E:\wslDistroStorage\CustomWSLDistro> Write-Output "\\.\PhysicalDrive$((Mount-VHD -Path .\ext4.vhdx -PassThru | Get-Disk).Number)"
\\.\PhysicalDrive3
PS E:\wslDistroStorage\CustomWSLDistro> wmic diskdrive list brief
Caption                                DeviceID           Model              Partitions  Size
-----                                -
Samsung SSD 970 EVO Plus 500GB        \\.\PHYSICALDRIVE0 Samsung SSD 970 EVO Plus 500GB 1           500105249280
ST2000DM001-1CH164                    \\.\PHYSICALDRIVE2 ST2000DM001-1CH164            1           2000396321280
D2CSTK251A20-0480                     \\.\PHYSICALDRIVE1 D2CSTK251A20-0480            3           480101368320
Microsoft Virtual Disk                 \\.\PHYSICALDRIVE3 Microsoft Virtual Disk        0           274872407040

PS E:\wslDistroStorage\CustomWSLDistro> wsl --mount \\.\PHYSICALDRIVE3
The disk \\.\PHYSICALDRIVE3 was successfully mounted under the name 'PHYSICALDRIVE3'. The mountpoint can be found under the path pointed to by the automount setting (default: /mnt/wsl).
To unmount and detach the disk, run 'wsl --unmount \\.\PHYSICALDRIVE3'.
PS E:\wslDistroStorage\CustomWSLDistro> wsl
/mnt/e/wslDistroStorage/CustomWSLDistro $ cd /mnt/wsl/PHYSICALDRIVE3/
/mnt/wsl/PHYSICALDRIVE3 $ ls
bin  etc  init  lib64  media  opt  root  sbin  sys  usr
dev  home  lib  lost+found  mnt  proc  run  srv  tmp  var
/mnt/wsl/PHYSICALDRIVE3 $
```

## Command line reference

### Mounting a specific filesystem

By default, WSL 2 will attempt to mount the device as ext4. To specify another filesystem, run:

```
wsl --mount <DiskPath> -t <FileSystem>
```

For example, to mount a disk as fat, run:

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

#### NOTE

To list the available filesystems in WSL2, run: `cat /proc/filesystems`

When a disk has been mounted via WSL2 (Linux file system), it is no longer available to mount via an ext4 driver on the Windows file system.

### Mounting a specific partition

By default, WSL 2 attempts to mount the entire disk. To mount a specific partition, run:

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

This only works if the disk is either MBR (Master Boot Record) or GPT (GUID Partition Table). [Read about partition styles - MBR and GPT.](#)

### Specifying mount options

To specify mount options, run:

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

Example:

```
ws1 --mount <DiskPath> -o "data=ordered"
```

#### NOTE

Only filesystem specific options are supported at this time. Generic options such as `ro, rw, noatime, ...` are not supported.

### Attaching the disk without mounting it

If the disk scheme isn't supported by any of the above options, you can attach the disk to WSL 2 without mounting it by running:

```
ws1 --mount <DiskPath> --bare
```

This will make the block device available inside WSL 2 so it can be mounted manually from there. Use `lsblk` to list the available block devices inside WSL 2.

### Detaching a disk

To detach a disk from WSL 2, run:

```
ws1 --unmount [DiskPath]
```

If `Diskpath` is omitted, all attached disks are unmounted and detached.

#### NOTE

If one disk fails to unmount, WSL 2 can be forced to exit by running `ws1 --shutdown`, which will detach the disk.

## Limitations

- At this time, only entire disks can be attached to WSL 2, meaning that it's not possible to attach only a partition. Concretely, this means that it's not possible to use `ws1 --mount` to read a partition on the boot device, because that device can't be detached from Windows.
- USB flash drives and SD cards are not supported at this time and will fail to attach to WSL 2. USB disks are supported though.
- Only filesystems that are natively supported in the kernel can be mounted by `ws1 --mount`. This means that it's not possible to use installed filesystem drivers (such as ntfs-3g for example) by calling `ws1 --mount .`

# Command Reference for Windows Subsystem for Linux

4/14/2021 • 3 minutes to read • [Edit Online](#)

The best way to interact with the Windows Subsystem for Linux is to use the `wsl.exe` command.

## Set WSL 2 as your default version

Run the following command in Powershell to set WSL 2 as the default version when installing a new Linux distribution:

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

### NOTE

If you're running a 32-bit process in order to access `wsl.exe` (a 64-bit tool), you may have to run the above command in the following manner: `C:\Windows\Sysnative\wsl.exe --set-default-version 2`

## Set your distribution version to WSL 1 or WSL 2

You can check the WSL version assigned to each of the Linux distributions you have installed by opening the PowerShell command line and entering the command (only available in [Windows Build 19041 or higher](#)):

```
wsl -l -v
```

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

To set a distribution to be backed by either version of WSL please run:

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

Make sure to replace `<distribution name>` with the actual name of your distribution and `<versionNumber>` with the number '1' or '2'. You can change back to WSL 1 at anytime by running the same command as above but replacing the '2' with a '1'.

Additionally, if you want to make WSL 2 your default architecture you can do so with this command:

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

This will set the version of any new distribution installed to WSL 2.

```
wsl.exe
```

Below is a list containing all options when using `wsl.exe` as of Windows Version 1903.

Using: `wsl [Argument] [Options...] [CommandLine]`

### Arguments for running Linux commands

- **Without arguments**

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.

The above commands also accept the following options:

- **--distribution, -d <Distro>**

Run the specified distribution.

- **--user, -u <UserName>**

Run as the specified user.

### **Arguments for managing Windows Subsystem for Linux**

- **--export <Distro> <FileName>**

Exports the distribution to a tar file. The filename can be - for standard output.

- **--import <Distro> <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.

- **--set-default, -s <Distro>**

Sets the distribution as the default.

- **--terminate, -t <Distro>**

Terminates the specified distribution.

- **--unregister <Distro>**

Un-register the distribution.

- **--help** Display usage information.

## **Additional Commands**

There are also historic commands to interact with the Windows Subsystem for Linux. Their functionality is encompassed within `wsl.exe`, but they are still available for use.



`wslconfig.exe`

This command lets you configure your WSL distribution. Below is a list of its options.

Using: `wslconfig [Argument] [Options...]`

#### Arguments

- **/l, /list [Options]**

Lists registered distributions.

Options:

- **/all** Optionally list all distributions, including distributions that are currently being installed or uninstalled.
- **/running** List only distributions that are currently running.
- **/s, /setdefault <Distro>** Sets the distribution as the default.
- **/t, /terminate <Distro>** Terminates the distribution.
- **/u, /unregister <Distro>** Un-registers the distribution.
- **/upgrade <Distro>** Upgrades the distribution to the WslFs file system format.

`bash.exe`

This command is used to start a bash shell. Below are the options you can use with this command.

Using: `bash [Options...]`

- **No Option given**

Launches the Bash shell in the current directory. If the Bash shell is not installed automatically runs

`lxrun /install`

- **~**

`bash ~` launches the bash shell into the user's home directory. Similar to running `cd ~`.

- **-c "<command>"**

Runs the command, prints the output and exits back to the Windows command prompt.

Example: `bash -c "ls"`.

## Deprecated Commands

The `lxrun.exe` was the first command used to install and manage the Windows Subsystem for Linux. It is deprecated as of Windows 10 1803 and later.

The command `lxrun.exe` can be used to interact with the Windows Subsystem for Linux (WSL) directly. These commands are installed into the `\Windows\System32` directory and may be run within a Windows command prompt or in PowerShell.

COMMAND	DESCRIPTION
<code>lxrun</code>	The lxrun command is used to manage the WSL instance.

COMMAND	DESCRIPTION
<code>lxrun /install</code>	Starts the download and install process. <i>/y</i> may be added to bypass all prompts. The confirmation prompt is automatically accepted and the default user is set to root.
<code>lxrun /uninstall</code>	Uninstalls and deletes the Ubuntu image. By default this does not remove the user's Ubuntu home directory. <i>/y</i> may be added to automatically accept the confirmation prompt <i>/full</i> uninstalls and deletes the user's Ubuntu home directory
<code>lxrun /setdefaultuser &lt;userName&gt;</code>	Sets the default Bash on Ubuntu user. Will prompt for a password if the specified user does not exist. For more information visit: <a href="https://aka.ms/wslusers">https://aka.ms/wslusers</a> . <i>/y</i> Bypasses prompting for the password. The user will be created without a password.
<code>lxrun /update</code>	Updates the subsystem's package index

# Troubleshooting Windows Subsystem for Linux

6/3/2021 • 11 minutes to read • [Edit Online](#)

For support with issues related to WSL, please see our [WSL product repo on GitHub](#).

## Search for any existing issues related to your problem

For technical issues, use the [product repo](#).

For issues related to the contents of this documentation, use the [docs repo](#).

## Submit a bug report

For bugs related to WSL functions or features, file an issue in the product repo:

<https://github.com/Microsoft/wsl/issues>

## Submit a feature request

To request a new feature related to WSL functionality or compatibility, [file an issue in the product repo](#).

## Contribute to the docs

To contribute to the WSL documentation, submit a pull request in the docs repo:

<https://github.com/MicrosoftDocs/wsl/issues>

## Terminal or Command Line

Lastly, if your issue is related to the Windows Terminal, Windows Console, or the command-line UI, use the Windows Terminal repo: <https://github.com/microsoft/terminal>

## Common issues

### I'm on Windows 10 version 1903 and I still do not see options for WSL 2

This is likely because your machine has not yet taken the backport for WSL 2. The simplest way to resolve this is by going to Windows Settings and clicking 'Check for Updates' to install the latest updates on your system. See [the full instructions on taking the backport](#).

If you hit 'Check for Updates' and still do not receive the update you can [install KB KB4566116 manually](#).

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

This may happen when 'Display Language' or 'System Locale' setting is not English.

```
wsl --set-default-version 2
Error: 0x1bc
For information on key differences with WSL 2 please visit https://aka.ms/wsl2
```

The actual error for `0x1bc` is:

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

For more information, please refer to issue [5749](#)

### Cannot access WSL files from Windows

A 9p protocol file server provides the service on the Linux side to allow Windows to access the Linux file system. If you cannot access WSL using `\\wsl$` on Windows, it could be because 9P did not start correctly.

To check this, you can check the start up logs using: `dmesg |grep 9p`, and this will show you any errors. A successful output looks like the following:

```
[ 0.363323] 9p: Installing v9fs 9p2000 file system support
[ 0.363336] FS-Cache: Netfs '9p' registered for caching
[ 0.398989] 9pnet: Installing 9P2000 support
```

Please see [this Github thread](#) for further discussion on this issue.

### Can't start WSL 2 distribution and only see 'WSL 2' in output

If your display language is not English, then it is possible you are seeing a truncated version of an error text.

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

To resolve this issue, please visit <https://aka.ms/wsl2kernel> and install the kernel manually by following the directions on that doc page.

### command not found when executing windows .exe in linux

Users can run Windows executables like notepad.exe directly from Linux. Sometimes, you may hit "command not found" like below:

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

If there are no win32 paths in your \$PATH, interop isn't going to find the .exe. You can verify it by running `echo $PATH` in Linux. It's expected that you will see a win32 path (for example, /mnt/c/Windows) in the output. If you can't see any Windows paths then most likely your PATH is being overwritten by your Linux shell.

Here is an example that /etc/profile on Debian contributed to the problem:

```
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
```

The correct way on Debian is to remove above lines. You may also append \$PATH during the assignment like below, but this leads to some [other problems](#) with WSL and VSCode..

```
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
```

For more information, see issue [5296](#) and issue [5779](#).

**"Error: 0x80370102 The virtual machine could not be started because a required feature is not installed."**

Please enable the Virtual Machine Platform Windows feature and ensure virtualization is enabled in the BIOS.

1. Check the [Hyper-V system requirements](#)
2. If your machine is a VM, please enable [nested virtualization](#) manually. Launch powershell with admin, and run:

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

3. Please follow guidelines from your PC's manufacturer on how to enable virtualization. In general, this can involve using the system BIOS to ensure that these features are enabled on your CPU. Instructions for this process can vary from machine to machine, please see [this article](#) from Bleeping Computer for an example.
4. Restart your machine after enabling the `Virtual Machine Platform` optional component.

### Bash loses network connectivity once connected to a VPN

If after connecting to a VPN on Windows, bash loses network connectivity, try this workaround from within bash. This workaround will allow you to manually override the DNS resolution through `/etc/resolv.conf`.

1. Take a note of the DNS server of the VPN from doing `ipconfig.exe /all`
2. Make a copy of the existing resolv.conf `sudo cp /etc/resolv.conf /etc/resolv.conf.new`
3. Unlink the current resolv.conf `sudo unlink /etc/resolv.conf`
4. `sudo mv /etc/resolv.conf.new /etc/resolv.conf`
5. Open `/etc/resolv.conf` and
  - a. Delete the first line from the file, which says "# This file was automatically generated by WSL. To stop automatic generation of this file, remove this line."
  - b. Add the DNS entry from (1) above as the very first entry in the list of DNS servers.
  - c. Close the file.

Once you have disconnected the VPN, you will have to revert the changes to `/etc/resolv.conf`. To do this, do:

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

### Starting WSL or installing a distribution returns an error code

Follow [these instructions](#) to collect detailed logs and file an issue on our GitHub.

### Updating Bash on Ubuntu on Windows

There are two components of Bash on Ubuntu on Windows that can require updating.

1. The Windows Subsystem for Linux

Upgrading this portion of Bash on Ubuntu on Windows will enable any new fixes outlines in the [release notes](#). Ensure that you are subscribed to the Windows Insider Program and that your build is up to date. For finer grain control including resetting your Ubuntu instance check out the [command reference page](#).

2. The Ubuntu user binaries

Upgrading this portion of Bash on Ubuntu on Windows will install any updates to the Ubuntu user binaries including applications that you have installed via apt-get. To update run the following commands in Bash:

- a. `apt-get update`
- b. `apt-get upgrade`

## Apt-get upgrade errors

Some packages use features that we haven't implemented yet. `udev`, for example, isn't supported yet and causes several `apt-get upgrade` errors.

To fix issues related to `udev`, follow the following steps:

1. Write the following to `/usr/sbin/policy-rc.d` and save your changes.

```
#!/bin/sh
exit 101
```

2. Add execute permissions to `/usr/sbin/policy-rc.d`:

```
chmod +x /usr/sbin/policy-rc.d
```

3. Run the following commands:

```
dpkg-divert --local --rename --add /sbin/initctl
ln -s /bin/true /sbin/initctl
```

## "Error: 0x80040306" on installation

This has to do with the fact that we do not support legacy console. To turn off legacy console:

1. Open cmd.exe
2. Right click title bar -> Properties -> Uncheck Use legacy console
3. Click OK

## "Error: 0x80040154" after Windows update

The Windows Subsystem for Linux feature may be disabled during a Windows update. If this happens the Windows feature must be re-enabled. Instructions for enabling the Windows Subsystem for Linux can be found in the [Installation Guide](#).

## Changing the display language

WSL install will try to automatically change the Ubuntu locale to match the locale of your Windows install. If you do not want this behavior you can run this command to change the Ubuntu locale after install completes. You will have to relaunch bash.exe for this change to take effect.

The below example changes to locale to en-US:

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

## Installation issues after Windows system restore

1. Delete the `%windir%\System32\Tasks\Microsoft\Windows\Windows Subsystem for Linux` folder.

**Note:** Do not do this if your optional feature is fully installed and working.

2. Enable the WSL optional feature (if not already)
3. Reboot
4. `lxrun /uninstall /full`
5. Install bash

## No internet access in WSL

Some users have reported issues with specific firewall applications blocking internet access in WSL. The firewalls reported are:

1. Kaspersky
2. AVG
3. Avast
4. Symantec Endpoint Protection

In some cases turning off the firewall allows for access. In some cases simply having the firewall installed looks to block access.

### Permission Denied error when using ping

For [Windows Anniversary Update, version 1607](#), **administrator privileges** in Windows are required to run ping in WSL. To run ping, run Bash on Ubuntu on Windows as an administrator, or run bash.exe from a CMD/PowerShell prompt with administrator privileges.

For later versions of Windows, [Build 14926+](#), administrator privileges are no longer required.

### Bash is hung

If while working with bash, you find that bash is hung (or deadlocked) and not responding to inputs, help us diagnose the issue by collecting and reporting a memory dump. Note that these steps will crash your system. Do not do this if you are not comfortable with that or save your work prior to doing this.

To collect a memory dump

1. Change the memory dump type to "complete memory dump". While changing the dump type, take a note of your current type.
2. Use the [steps](#) to configure crash using keyboard control.
3. Repro the hang or deadlock.
4. Crash the system using the key sequence from (2).
5. The system will crash and collect the memory dump.
6. Once the system reboots, report the memory.dmp to [secure@microsoft.com](mailto:secure@microsoft.com). The default location of the dump file is %SystemRoot%\memory.dmp or C:\Windows\memory.dmp if C: is the system drive. In the email, note that the dump is for the WSL or Bash on Windows team.
7. Restore the memory dump type to the original setting.

### Check your build number

To find your PC's architecture and Windows build number, open

**Settings > System > About**

Look for the **OS Build** and **System Type** fields.

The screenshot shows the Windows Settings application, specifically the 'System' > 'About' section. The 'Settings' title bar is at the top. On the left, there is a navigation pane with 'Home' selected. Below it is a search bar labeled 'Find a setting'. The main content area displays system information in a table-like format. The 'Edition' is 'Windows 10 Home' and the 'Version' is '1607'. The 'OS Build' is '14393.0', which is highlighted with a red box. Below this, the 'Product ID' is '00326-10000-00000-AA728'. The 'Processor' is 'Intel(R) Xeon(R) CPU W3520 @ 2.67GHz'. The 'Installed RAM' is '1.00 GB'. The 'System type' is '64-bit operating system, x64-based processor', which is also highlighted with a red box. At the bottom, there are links for 'Change product key or upgrade your edition of Windows' and 'Read the Privacy Statement for Windows and Microsoft services'.

Edition	Windows 10 Home
Version	1607
OS Build	14393.0
Product ID	00326-10000-00000-AA728
Processor	Intel(R) Xeon(R) CPU W3520 @ 2.67GHz
Installed RAM	1.00 GB
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

[Change product key or upgrade your edition of Windows](#)

[Read the Privacy Statement for Windows and Microsoft services](#)

To find your Windows Server build number, run the following in PowerShell:

```
systeminfo | Select-String "^OS Name","^OS Version"
```

### Confirm WSL is enabled

You can confirm that the Windows Subsystem for Linux is enabled by running the following in PowerShell:

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

### OpenSSH-Server connection issues

Trying to connect your SSH server is failed with the following error: "Connection closed by 127.0.0.1 port 22".

1. Make sure your OpenSSH Server is running:

```
sudo service ssh status
```

and you've followed this tutorial: <https://ubuntu.com/server/docs/service-openssh>

2. Stop the sshd service and start sshd in debug mode:

```
sudo service ssh stop  
sudo /usr/sbin/sshd -d
```

3. Check the startup logs and make sure HostKeys are available and you don't see log messages such as:

```
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
```

If you do see such messages and the keys are missing under `/etc/ssh/`, you will have to regenerate the keys or just purge&install openssh-server:

```
sudo apt-get purge openssh-server  
sudo apt-get install openssh-server
```

### "The referenced assembly could not be found." when enabling the WSL optional feature

This error is related to being in a bad install state. Please complete the following steps to try and fix this issue:

- If you are running the enable WSL feature command from PowerShell, try using the GUI instead by opening the start menu, searching for 'Turn Windows features on or off' and then in the list select 'Windows Subsystem for Linux' which will install the optional component.
- Update your version of Windows by going to Settings, Updates, and clicking 'Check for Updates'



- If both of those fail and you need to access WSL please consider upgrading in place by reinstalling Windows 10 using installation media and selecting 'Keep Everything' to ensure your apps and files are preserved. You can find instructions on how to do so at the [Reinstall Windows 10 page](#).

### Correct (SSH related) permission errors

If you're seeing this error:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                WARNING: UNPROTECTED PRIVATE KEY FILE!                @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Permissions 0777 for '/home/artur/.ssh/private-key.pem' are too open.
```

To fix this, append the following to the the `/etc/wsl.conf` file:

```
[automount]
enabled = true
options = metadata,uid=1000,gid=1000,umask=0022
```

Please note that adding this command will include metadata and modify the file permissions on the Windows files seen from WSL. Please see the [File System Permissions](#) for more information.

### Running Windows commands fails inside a distribution

Some distributions [available in Microsoft Store](#) are yet not fully compatible to run Windows commands in [Terminal](#) out of the box. If you get an error `-bash: powershell.exe: command not found` running

`powershell.exe /c start .` or any other Windows command, you can resolve it following these steps:

1. In your WSL distribution run `echo $PATH`.  
If it does not include: `/mnt/c/Windows/system32` something is redefining the standard PATH variable.
2. Check profile settings with `cat /etc/profile`.  
If it contains assignment of the PATH variable, edit the file to comment out PATH assignment block with a `#` character.
3. Check if `wsl.conf` is present `cat /etc/wsl.conf` and make sure it does not contain `appendWindowsPath=false`, otherwise comment it out.
4. Restart distribution by typing `wsl -t` followed by distribution name or run `wsl --shutdown` either in cmd or PowerShell.

### Unable to boot after installing WSL 2

We are aware of an issue affecting users where they are unable to boot after installing WSL 2. While we fully diagnose those issue, users have reported that [changing the buffer size](#) or [installing the right drivers](#) can help address this. Please view this [Github issue](#) to see the latest updates on this issue.

### WSL 2 errors when ICS is disabled

Internet Connection Sharing (ICS) is a required component of WSL 2. The ICS service is used by the Host Network Service (HNS) to create the underlying virtual network which WSL 2 relies on for NAT, DNS, DHCP, and host connection sharing.

Disabling the ICS service (SharedAccess) or disabling ICS through group policy will prevent the WSL HNS network from being created. This will result in failures when creating a new WSL version 2 image, and the following error when trying to convert a version 1 image to version 2.

```
There are no more endpoints available from the endpoint mapper.
```

Systems that require WSL 2 should leave the ICS service (SharedAccess) in it's default start state, Manual

(Trigger Start), and any policy that disables ICS should be overwritten or removed. We do not recommend disabling ICS, however it can be disabled [using these instructions](#)

# Release Notes for Windows Subsystem for Linux

4/27/2021 • 63 minutes to read • [Edit Online](#)

## Build 21364

For general Windows information on build 21364 visit the [Windows blog](#).

- GUI apps are now available! For more information see [this blog post](#).
- Resolve error when accessing files via \\wsl.localhost\.
- Fix potential deadlock in LxssManager service.

## Build 21354

For general Windows information on build 21354 visit the [Windows blog](#).

- Switch the \wsl prefix to \wsl.localhost to avoid issues when there is a machine on the network named "wsl". \\wsl\$ will continue to work.
- Enable Linux quick access icon for wow processes.
- Update issue where version 2 was always being passed via wslapi RegisterDistribution.
- Change the fmask of the /usr/lib/wsl/lib directory to 222 so files are marked as executable [GH 3847]
- Fix wsl service crash if Virtual Machine Platform is not enabled.

## Build 21286

For general Windows information on build 21286 visit the [Windows blog](#).

- Introduce wsl.exe --cd command to set current working directory of a command.
- Improve mapping of NTSTATUS to Linux error codes. [GH 6063]
- Improve wsl.exe --mount error reporting.
- Added an option to /etc/wsl.conf to enable start up commands:

```
[boot]
command=<string>
```

## Build 20226

For general Windows information on build 20226 visit the [Windows blog](#).

- Fix crash in LxssManager service. [GH 5902]

## Build 20211

For general Windows information on build 20211 visit the [Windows blog](#).

- Introduce `wsl.exe --mount` for mounting physical or virtual disks. For more information see [Access Linux filesystems in Windows and WSL 2](#).
- Fix crash in LxssManager service when checking if the VM is idle. [GH 5768]
- Support for compressed VHD files. [GH 4103]
- Ensure that Linux user mode libs installed to c:\windows\system32\lxss\lib are preserved across OS upgrade. [GH 5848]
- Added the ability to list available distributions that can be installed with `wsl --install --list-distributions`.
- WSL instances are now terminated when the user logs off.

## Build 20190

For general Windows information on build 20190 visit the [Windows blog](#).

- Fix bug preventing WSL1 instances from launching. [GH 5633]
- Fix hang when redirecting Windows process output. [GH 5648]
- Add %userprofile%\wslconfig option to control the VM idle timeout (wsl2.vmlIdleTimeout= <time\_in\_ms>).
- Support launching app execution aliases from WSL.
- Added support for installing the WSL2 kernel and distributions to wsl.exe --install.

## Build 20175

For general Windows information on build 20175 visit the [Windows blog](#).

- Adjust default memory assignment of WSL2 VM to be 50% of host memory or 8GB, whichever is less [GH 4166].
- Change \\wsl\$ prefix to \\wsl to support URI parsing. The old \\wsl\$ path is still supported.
- Enable nested virtualization for WSL2 by default on amd64. You can disable this via %userprofile%\wslconfig ([wsl2] nestedVirtualization=false).
- Make wsl.exe --update demand start Microsoft Update.
- Support renaming over a read-only file in DrvFs.
- Ensure error messages are always printed in the correct codepage.

## Build 20150

For general Windows information on build 20150 visit the [Windows blog](#).

- WSL2 GPU compute see [Windows blog](#) for more information.
- Introduce wsl.exe --install command line option to easily set up WSL.
- Introduce wsl.exe --update command line option to manage updates to the WSL2 kernel.
- Set WSL2 as the default.
- Increase WSL2 vm graceful shutdown timeout.
- Fix virtio-9p race condition when mapping device memory.
- Don't run an elevated 9p server if UAC is disabled.

## Build 19640

For general Windows information on build 19640 visit the [Windows blog](#).

- [WSL2] Stability improvements for virtio-9p (drvfs).

## Build 19555

For general Windows information on build 19555 visit the [Windows blog](#).

- [WSL2] Use a memory cgroup to limit the amount of memory used by install and conversion operations [GH 4669]
- Make wsl.exe present when the Windows Subsystem for Linux optional component is not enabled to improve feature discoverability.
- Change wsl.exe to print help text if the WSL optional component is not installed
- Fix race condition when creating instances
- Create wslclient.dll that contains all command line functionality
- Prevent crash during LxssManagerUser service stop
- Fix wslapi.dll fast fail when distroName parameter is NULL

## Build 19041

For general Windows information on build 19041 visit the [Windows blog](#).

- [WSL2] Clear the signal mask before launching the processes
- [WSL2] Update Linux kernel to 4.19.84
- Handle creation of /etc/resolv.conf symlink when the symlink is non-relative

## Build 19028

For general Windows information on build 19028 visit the [Windows blog](#).

- [WSL2] Update Linux kernel to 4.19.81
- [WSL2] Change the default permission of /dev/net/tun to 0666 [GH 4629]
- [WSL2] Tweak default amount of memory assigned to Linux VM to be 80% of host memory
- [WSL2] fix interop server to handle requests with a timeout so bad callers cannot hang the server

## Build 19018

For general Windows information on build 19018 visit the [Windows blog](#).

- [WSL2] Use cache=mmap as the default for 9p mounts to fix dotnet apps
- [WSL2] Fixes for localhost relay [GH 4340]
- [WSL2] Introduce a cross-distro shared tmpfs mount for sharing state between distros
- Fix restoring persistent network drive for \\wsl\$

## Build 19013

For general Windows information on build 19013 visit the [Windows blog](#).

- [WSL2] Improve memory performance of WSL utility VM. Memory that is no longer in use will be freed back to the host.
- [WSL2] Update kernel version to 4.19.79. (add CONFIG\_HIGH\_RES\_TIMERS, CONFIG\_TASK\_XACCT, CONFIG\_TASK\_IO\_ACCOUNTING, CONFIG\_SCHED\_HRTICK, and CONFIG\_BRIDGE\_VLAN\_FILTERING).
- [WSL2] Fix input relay to handle cases where stdin is a pipe handle that is not closed [GH 4424]
- Make the check for \\wsl\$ case-insensitive.

```
[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).
```

## Build 19002

For general Windows information on build 19002 visit the [Windows blog](#).

- [WSL] Fix issue with handling of some Unicode characters:  
<https://github.com/microsoft/terminal/issues/2770>
- [WSL] Fix rare cases where distros could be unregistered if launched immediately after a build-to-build upgrade.
- [WSL] Fix minor issue with wsl.exe --shutdown where instance idle timers were not cancelled.

## Build 18995

For general Windows information on build 18995 visit the [Windows blog](#).

- [WSL2] Fix an issue where DrvFs mounts stopped working after an operation was interrupted (e.g. ctrl-c) [GH 4377]
- [WSL2] Fix handling of very large hvsocket messages [GH 4105]

- [WSL2] Fix issue with interop when stdin is a file [GH 4475]
- [WSL2] Fix service crash when unexpected network state is encountered [GH 4474]
- [WSL2] Query the distro name from the interop server if the current process does not have the environment variable
- [WSL2] Fix issue with interop whe stdin is a file
- [WSL2] Update Linux kernel version to 4.19.72
- [WSL2] Add ability to specify additional kernel command line parameters via .wslconfig

```
[wsl2]
kernelCommandLine = <string> # Additional kernel command line arguments
```

## Build 18990

For general Windows information on build 18990 visit the [Windows blog](#).

- Improve the performance for directory listings in \\wsl\$
- [WSL2] Inject additional boot entropy [GH 4416]
- [WSL2] Fix for Windows interop when using su / sudo [GH 4465]

## Build 18980

For general Windows information on build 18980 visit the [Windows blog](#).

- Fix reading symlinks that deny FILE\_READ\_DATA. This includes all the symlinks Windows creates for backwards compatibility such as "C:\Document and Settings" and a bunch of symlinks in the user profile directory
- Make unexpected filesystem state non-fatal [GH 4334, 4305]
- [WSL2] Add support for arm64 if your CPU / firmware supports virtualization
- [WSL2] Allow unprivileged users to view kernel log
- [WSL2] Fix output relay when stdout / stderr sockets have been closed [GH 4375]
- [WSL2] Support battery and AC adapter passthrough
- [WSL2] Update Linux kernel to 4.19.67
- Add the ability to set default username in /etc/wsl.conf:

```
[user]
default=<string>
```

## Build 18975

For general Windows information on build 18975 visit the [Windows blog](#).

- [WSL2] Fixed a number of localhost reliability issues [GH 4340]

## Build 18970

For general Windows information on build 18970 visit the [Windows blog](#).

- [WSL2] Sync time with host time when system resumes from sleep state [GH 4245]
- [WSL2] Create NT symlinks on the Windows volumes when possible.
- [WSL2] Create distros in UTS, IPC, PID, and Mount namespaces.
- [WSL2] Fix localhost port relay when server binds to localhost directly [GH 4353]
- [WSL2] Fix interop when output is redirected [GH 4337]
- [WSL2] Support translating absolute NT symlinks.
- [WSL2] Update kernel to 4.19.59

- [WSL2] Properly set subnet mask for eth0.
- [WSL2] Change logic to break out of console worker loop when exit event is signaled.
- [WSL2] Eject distribution vhd when the distro is not running.
- [WSL2] Fix config parsing library to correctly handle empty values.
- [WSL2] Support Docker Desktop by creating cross distro mounts. A distro can opt-in to this behavior by adding the following line to the `/etc/wsl.conf` file:

```
[automount]
crossDistro = true
```

## Build 18945

For general Windows information on build 18945 visit the [Windows blog](#).

### WSL

- [WSL2] Allow listening tcp sockets in WSL2 to be accessible from the host by using localhost:port
- [WSL2] Fixes for install / conversion failures and additional diagnostics to track down future issues [GH 4105]
- [WSL2] Improve diagnosability of WSL2 network issues
- [WSL2] Update kernel version to 4.19.55
- [WSL2] Update kernel with config options required for docker [GH 4165]
- [WSL2] Increase the number of CPUs assigned to the lightweight utility VM to be the same as the host (was previously capped at 8 by CONFIG\_NR\_CPUS in the kernel config) [GH 4137]
- [WSL2] Create a swap file for the WSL2 lightweight VM
- [WSL2] Allow user mounts to be visible via `\\wsl$\distro` (for example sshfs) [GH 4172]
- [WSL2] Improve 9p filesystem performance
- [WSL2] Ensure vhd ACL does not grow unbounded [GH 4126]
- [WSL2] Update kernel config to support squashfs and xt\_contrack [GH 4107, 4123]
- [WSL2] Fix for `interop.enabled` `/etc/wsl.conf` option [GH 4140]
- [WSL2] Return ENOTSUP if the file system does not support EAs
- [WSL2] Fix CopyFile hang with `\\wsl$`
- Switch default umask to 0022 and add `filesystem.umask` setting to `/etc/wsl.conf`
- Fix `wslpath` to properly resolve symlinks, this was regressed in 19h1 [GH 4078]
- Introduce `%UserProfile%\wslconfig` file for tweaking WSL2 settings

```
[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
```

## Build 18917

For general Windows information on build 18917 visit the [Windows blog](#).

### WSL

- WSL 2 is now available! Please see [blog](#) for more details.
- Fix a regression where launching Windows processes via symlinks did not work correctly [GH 3999]

- Add wsl.exe --list --verbose, wsl.exe --list --quiet, and wsl.exe --import --version options to wsl.exe
- Add wsl.exe --shutdown option
- Plan 9: Allow opening a directory for write to succeed

## Build 18890

For general Windows information on build 18890 visit the [Windows blog](#).

### WSL

- Non-blocking socket leak [GH 2913]
- EOF input to terminal can block subsequent reads [GH 3421]
- Update resolv.conf header to refer to wsl.conf [discussed in GH 3928]
- Deadlock in epoll delete code [GH 3922]
- Handle spaces in arguments to --import and --export [GH 3932]
- Extending mmap'd files does not work properly [GH 3939]
- Fix issue with ARM64 \\wsl\$ access not working properly
- Add better default icon for wsl.exe

## Build 18342

For general Windows information on build 18342 visit the [Windows blog](#).

### WSL

- We've added the ability for users to access Linux files in a WSL distro from Windows. These files can be accessed through the command line, and also Windows apps, like file explorer, VSCode, etc. can interact with these files. Access your files by navigating to \\wsl\$<distro\_name>, or see a list of running distributions by navigating to \\wsl\$
- Add additional CPU info tags and fix Cpus\_allowed[\_list] values [GH 2234]
- Support exec from non-leader thread [GH 3800]
- Treat configuration update failures as non-fatal [GH 3785]
- Update binfmt to properly handle offsets [GH 3768]
- Enable mapping network drives for Plan 9 [GH 3854]
- Support Windows -> Linux and Linux -> Windows path translation for bind mounts
- Create read-only sections for mappings on files opened read-only

## Build 18334

For general Windows information on build 18334 visit the [Windows blog](#).

### WSL

- Redesign the way that Windows time zone is mapped to a Linux time zone [GH 3747]
- Fix memory leaks and add new string translation functions [GH 3746]
- SIGCONT on a threadgroup with no threads is a no-op [GH 3741]
- Correctly display socket and epoll file descriptors in /proc/self/fd

## Build 18305

For general Windows information on build 18305 visit the [Windows blog](#).

### WSL

- pthreads lose access to files when the primary thread exits [GH 3589]
- TIOCSCTTY should ignore the "force" parameter unless it is required [GH 3652]
- wsl.exe command line improvements and addition of import / export functionality.



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.

## Build 18277

For general Windows information on build 18277 visit the [Windows blog](#).

### WSL

- Fix "no such interface supported" error introduced in build 18272 [GH 3645]
- Ignore the MNT\_FORCE flag for umount syscall [GH 3605]
- Switch WSL interop to use the official CreatePseudoConsole API
- Maintain no timeout value when FUTEX\_WAIT restarts

## Build 18272

For general Windows information on build 18272 visit the [Windows blog](#).

## WSL

- **WARNING:** There is an issue in this build that makes WSL inoperable. When trying to launch your distribution you will see a "No such interface supported" error. The issue has been fixed and will be in next week's Insider Fast build. If you've installed this build you can roll back to the previous Windows build using "Go back to the previous version of Windows 10" in Settings->Update & Security->Recovery.

## Build 18267

For general Windows information on build 18267 visit the [Windows blog](#).

## WSL

- Fix issue where zombie process may not be reaped and remain indefinitely.
- WslRegisterDistribution hangs if error message exceeds max length [GH 3592]
- Allow fsync to succeed for read-only files on DrvFs [GH 3556]
- Ensure that /bin and /sbin directories exist before creating symlinks inside [GH 3584]
- Added an instance termination timeout mechanism for WSL instances. The timeout is currently set to 15 seconds, meaning the instance will terminate 15 seconds after the last WSL process exits. To terminate a distribution immediately, use:

```
wslconfig.exe /terminate <DistributionName>
```

## Build 17763 (1809)

For general Windows information on build 17763 visit the [Windows blog](#).

## WSL

- Setpriority syscall permission check too strict for changing same thread priority [GH 1838]
- Ensure that unbiased interrupt time is used for boot time to avoid returning negative values for clock\_gettime(CLOCK\_BOOTTIME) [GH 3434]
- Handle symlinks in the WSL binfmt interpreter [GH 3424]
- Better handling of threadgroup leader file descriptor cleanup.
- Switch WSL to use KeQueryInterruptTimePrecise instead of KeQueryPerformanceCounter to avoid overflow [GH 3252]
- Ptrace attach may cause bad return value from system calls [GH 1731]
- Fix several AF\_UNIX related issues [GH 3371]
- Fix issue that could cause WSL interop to fail if the current working directory is less than 5 characters long [GH 3379]
- Avoid one second delay failing loopback connections to non-existent ports [GH 3286]
- Add /proc/sys/fs/file-max stub file [GH 2893]
- More accurate IPV6 scope information.
- PR\_SET\_PTRACER support [GH 3053]
- Pipe filesystem inadvertently clearing edge-triggered epoll event [GH 3276]
- Win32 executable launched via NTFS symlink doesn't respect symlink name [GH 2909]
- Improved zombie support [GH 1353]
- Add wsl.conf entries for controlling Windows interop behavior [GH 1493]

```
[interop]

enabled=false # enable launch of Windows binaries; default is true

appendWindowsPath=false # append Windows path to $PATH variable; default is true
```

- Fix for getsockname not always returning UNIX socket family type [GH 1774]
- Add support for TIOCSTI [GH 1863]
- Non-blocking sockets in the process of connecting should return EAGAIN for write attempts [GH 2846]
- Support interop on mounted VHDs [GH 3246, 3291]
- Fix permission checking issue on root folder [GH 3304]
- Limited support for TTY keyboard ioctls KDGKBTYPE, KDGKBMODE and KDSKBMODE.
- Windows UI apps should execute even when launched in the background.
- Add wsl -u or --user option [GH 1203]
- Fix WSL launch issues when fast startup is enabled [GH 2576]
- Unix sockets need to retain disconnected peer credentials [GH 3183]
- Non-blocking Unix sockets failing indefinitely with EAGAIN [GH 3191]
- case=off is the new default drvfs mount type [GH 2937, 3212, 3328]
  - See [blog](#) for more information.
- Add wslconfig /terminate to stop running distributions.
- Fix issue with the WSL shell context menu entries that do not correctly handle paths with spaces.
- Expose per-directory case sensitivity as an extended attribute
- ARM64: Emulate cache maintenance operations. Resolve [dotnet issue](#).
- DrvFs: only unescape characters in the private range that correspond to an escaped character.
- Fix off-by-one error in ELF parser interpreter length validation [GH 3154]
- WSL absolute timers with a time in the past do not fire [GH 3091]
- Ensure newly created reparse points are listed as such in the parent directory.
- Atomically create case sensitive directories in DrvFs.
- Fixed an additional issue where multithreaded operations could return ENOENT even though the file exists. [GH 2712]
- Fixed WSL launch failure when UMCI is enabled. [GH 3020]
- Add explorer context menu to launch WSL [GH 437, 603, 1836]. To use, hold shift and right-click when in an explorer window.
- Fix Unix socket non-blocking behavior [GH 2822, 3100]
- Fix hanging NETLINK command as reported in GH 2026.
- Add support for mount propagation flags [GH 2911].
- Fix issue with truncate not causing inotify events [GH 2978].
- Add --exec option for wsl.exe to invoke a single binary without a shell.
- Add --distribution option for wsl.exe to select a specific distro.
- Limited support for dmesg. Applications can now log to dmesg. WSL driver logs limited information to dmesg. In future, this can be extended to carry other information/diagnostics from the driver.
  - Note: dmesg is currently supported through the `/dev/kmsg` device interface. `syslog` syscall interface is not yet supported. And, so, some of the `dmesg` command line options such as `-s`, `-c` don't work.
- Change default gid and mode of serial devices to match native [GH 3042]
- DrvFs now supports extended attributes.
  - Note: DrvFs has some limitations on the name of extended attributes. Some characters (like '/', ':' and '\*') are not allowed, and extended attribute names are not case sensitive on DrvFs

## Build 18252 (Skip Ahead)

For general Windows information on build 18252 visit the [Windows Blog](#).

### WSL

- Move init and bsdtar binaries out of lxssmanager.dll and into a separate tools folder
- Fix race around closing file descriptor when using CLONE\_FILES
- Handle optional fields in /proc/pid/mountinfo when translating DrvFs paths
- Allow DrvFs mknod to succeed without metadata support for S\_IFREG

- Readonly files created on DrvFs should have the readonly attribute set [GH 3411]
- Add /sbin/mount.drvfs helper to handle DrvFs mounting
- Use POSIX rename in DrvFs.
- Allow path translation on volumes without a volume GUID.

## Build 17738 (Fast)

For general Windows information on build 17738 visit the [Windows Blog](#).

### WSL

- Setpriority syscall permission check too strict for changing same thread priority [GH 1838]
- Ensure that unbiased interrupt time is used for boot time to avoid returning negative values for clock\_gettime(CLOCK\_BOOTTIME) [GH 3434]
- Handle symlinks in the WSL binfmt interpreter [GH 3424]
- Better handling of threadgroup leader file descriptor cleanup.

## Build 17728 (Fast)

For general Windows information on build 17728 visit the [Windows Blog](#).

### WSL

- Switch WSL to use KeQueryInterruptTimePrecise instead of KeQueryPerformanceCounter to avoid overflow [GH 3252]
- Ptrace attach may cause bad return value from system calls [GH 1731]
- Fix a number of AF\_UNIX related issues [GH 3371]
- Fix issue that could cause WSL interop to fail if the current working directory is less than 5 characters long [GH 3379]

## Build 18204 (Skip Ahead)

For general Windows information on build 18204 visit the [Windows Blog](#).

### WSL

- Pipe filesystem inadvertently clearing edge-triggered epoll event [GH 3276]
- Win32 executable launched via NTFS symlink doesn't respect symlink name [GH 2909]

## Build 17723 (Fast)

For general Windows information on build 17723 visit the [Windows Blog](#).

### WSL

- Avoid one second delay failing loopback connections to non-existent ports [GH 3286]
- Add /proc/sys/fs/file-max stub file [GH 2893]
- More accurate IPV6 scope information.
- PR\_SET\_PTRACER support [GH 3053]
- Pipe filesystem inadvertently clearing edge-triggered epoll event [GH 3276]
- Win32 executable launched via NTFS symlink doesn't respect symlink name [GH 2909]

## Build 17713

For general Windows information on build 17713 visit the [Windows Blog](#).

### WSL

- Improved zombie support [GH 1353]
- Add wsl.conf entries for controlling Windows interop behavior [GH 1493]

```
[interop]
```

```
enabled=false # enable launch of Windows binaries; default is true
```

```
appendWindowsPath=false # append Windows path to $PATH variable; default is true
```

- Fix for getsockname not always returning UNIX socket family type [GH 1774]
- Add support for TIOCSTI [GH 1863]
- Non-blocking sockets in the process of connecting should return EAGAIN for write attempts [GH 2846]
- Support interop on mounted VHDs [GH 3246, 3291]
- Fix permission checking issue on root folder [GH 3304]
- Limited support for TTY keyboard ioctls KDGKBTTYPE, KDGKBMODE and KDSKBMODE.
- Windows UI apps should execute even when launched in the background.

## Build 17704

For general Windows information on build 17704 visit the [Windows Blog](#).

### WSL

- Add wsl -u or --user option [GH 1203]
- Fix WSL launch issues when fast startup is enabled [GH 2576]
- Unix sockets need to retain disconnected peer credentials [GH 3183]
- Non-blocking Unix sockets failing indefinitely with EAGAIN [GH 3191]
- case=off is the new default drvfs mount type [GH 2937, 3212, 3328]
  - See [blog](#) for more information.
- Add wslconfig /terminate to stop running distributions.

## Build 17692

For general Windows information on build 17692 visit the [Windows Blog](#).

### WSL

- Fix issue with the WSL shell context menu entries that do not correctly handle paths with spaces.
- Expose per-directory case sensitivity as an extended attribute
- ARM64: Emulate cache maintenance operations. Resolve [dotnet issue](#).
- DrvFs: only unescape characters in the private range that correspond to an escaped character.

## Build 17686

For general Windows information on build 17686 visit the [Windows Blog](#).

### WSL

- Fix off-by-one error in ELF parser interpreter length validation [GH 3154]
- WSL absolute timers with a time in the past do not fire [GH 3091]
- Ensure newly created reparse points are listed as such in the parent directory.
- Atomically create case sensitive directories in DrvFs.

## Build 17677

For general Windows information on build 17677 visit the [Windows Blog](#).

### WSL

- Fixed an additional issue where multithreaded operations could return ENOENT even though the file exists. [GH 2712]
- Fixed WSL launch failure when UMCI is enabled. [GH 3020]

# Build 17666

For general Windows information on build 17666 visit the [Windows Blog](#).

## WSL

**WARNING: There is an issue preventing WSL from running on some AMD chipsets [GH 3134]. A fix is ready and making its way to the Insider Build branch.**

- Add explorer context menu to launch WSL [GH 437, 603, 1836]. To use hold shift and right-click when in an explorer window.
- Fix unix socket non-blocking behavior [GH 2822, 3100]
- Fix hanging NETLINK command as reported in GH 2026.
- Add support for mount propagation flags [GH 2911].
- Fix issue with truncate not causing inotify events [GH 2978].
- Add --exec option for wsl.exe to invoke a single binary without a shell.
- Add --distribution option for wsl.exe to select a specific distro.

# Build 17655 (Skip Ahead)

For general Windows information on build 17655 visit the [Windows Blog](#).

## WSL

- Limited support for dmesg. Applications can now log to dmesg. WSL driver logs limited information to dmesg. In future, this can be extended to carry other information/diagnostics from the driver.
  - Note: dmesg is currently supported through the `/dev/kmsg` device interface. `syslog` syscall interface is not yet supported. And, so, some of the `dmesg` command line options such as `-s`, `-c` don't work.
- Fixed an issue where multithreaded operations could return ENOENT even though the file exists. [GH 2712]

# Build 17639 (Skip Ahead)

For general Windows information on build 17639 visit the [Windows Blog](#).

## WSL

- Change default gid and mode of serial devices to match native [GH 3042]
- DrvFs now supports extended attributes.
  - Note: DrvFs has some limitations on the name of extended attributes. In particular, some characters (like '/', ':' and '\*') are not allowed, and extended attribute names are not case sensitive on DrvFs

# Build 17133 (Fast)

For general Windows information on build 17133 visit the [Windows Blog](#).

## WSL

- Fix for hang in WSL. [GH 3039, 3034]

# Build 17128 (Fast)

For general Windows information on build 17128 visit the [Windows Blog](#).

## WSL

- None

# Build 17627 (Skip Ahead)

For general Windows information on build 17627 visit the [Windows Blog](#).

## WSL

- Add support for the futex pi-aware operations. [GH 1006]

- Note that priorities are not currently a supported WSL feature so there are limitations, but standard usage should be unblocked.
- Windows firewall support for WSL processes. [GH 1852]
  - For example, to allow the WSL python process to listen on any port, use the elevated Windows cmd:

```
netsh.exe advfirewall firewall add rule name=wsl_python dir=in action=allow program="C:\users\  
<username>\appdata\local\packages\canonicalgroup\limited.ubuntuonwindows_79rhkp1fndgsc\localstate\rootfs\usr\bin\python;  
enable=yes
```

- For additional details on how to add firewall rules, see [link](#)
- Respect user's default shell when using wsl.exe. [GH 2372]
- Report all network interfaces as ethernet. [GH 2996]
- Better handling of corrupt /etc/passwd file. [GH 3001]

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17618 (Skip Ahead)

For general Windows information on build 17618 visit the [Windows Blog](#).

#### WSL

- Introduce pseudoconsole functionality for NT interop [GH 988, 1366, 1433, 1542, 2370, 2406].
- The legacy install mechanism (lxdm.exe) has been deprecated. The supported mechanism for installing distributions is through the Microsoft Store.

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17110

For general Windows information on build 17110 visit the [Windows Blog](#).

#### WSL

- Allow /init to be terminated from Windows [GH 2928].
- DrvFs now uses per-directory case sensitivity by default (equivalent to the "case=dir" mount option).
  - Using "case=force" (the old behavior) requires setting a registry key. Run the following command to enable "case=force" if you need to use it: reg add HKLM\SYSTEM\CurrentControlSet\Services\lxs /v DrvFsAllowForceCaseSensitivity /t REG\_DWORD /d 1
  - If you have existing directories created with WSL in older version of Windows which need to be case sensitive, use fsutil.exe to mark them as case sensitive: fsutil.exe file setcasesensitiveinfo enable
- NULL terminate strings returned from the uname syscall.

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17107

For general Windows information on build 17107 visit the [Windows Blog](#).

#### WSL

- Support TCSETSF and TCSETSW on master pty endpoints [GH 2552].
- Starting simultaneous interop processes can result in EINVAL [GH 2813].
- Fix PTRACE\_ATTACH to show proper tracing status in /proc/pid/status.
- Fix race where short-lived processes cloned with both the CLEAR\_TID and SET\_TID flags could exit without clearing the TID address.
- Display a message when upgrading the Linux file system directories when moving from a pre-17093 build. For more details on the 17093 file system changes, see the release notes for [17093](#).

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17101

For general Windows information on build 17101 visit the [Windows Blog](#).

#### WSL

- Support for signalfd. [GH 129]
- Support file-names containing illegal NTFS characters by encoding them as private Unicode characters. [GH 1514]
- Auto mount will fallback to read-only when write is not supported. [GH 2603]
- Allow pasting of Unicode surrogate pairs (like emoji characters). [GH 2765]
- Pseudo-files in /proc and /sys should return read and write ready from select, poll, epoll, et al. [GH 2838]
- Fix issue that could cause service to go into infinite loop when the registry has been tampered with or is corrupt.
- Fix netlink messages to work with newer (upstream 4.14) version of iproute2.

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17093

For general Windows information on build 17093 visit the [Windows Blog](#).

#### Important:

When starting WSL for the first time after upgrading to this build, it needs to perform some work upgrading the Linux file system directories. This may take up to several minutes, so WSL may appear to start slowly. This should only happen once for each distribution you have installed from the store.

- Improved case sensitivity support in DrvFs.
  - DrvFs now supports per-directory case sensitivity. This is a new flag that can be set on directories to indicate all operations in those directories should be treated as case sensitive, which allows even Windows applications to correctly open files that differ only by case.
  - DrvFs has new mount options controlling case sensitivity on a per-directory basis
    - case=force: all directories are treated as case sensitive (except for the drive root). New directories created with WSL are marked as case sensitive. This is the legacy behavior except for marking new directories case sensitive.
    - case=dir: only directories with the per-directory case sensitivity flag are treated as case sensitive; other directories are case insensitive. New directories created with WSL are marked as case sensitive.
    - case=off: only directories with the per-directory case sensitivity flag are treated as case



sensitive; other directories are case insensitive. New directories created with WSL are marked as case insensitive.

- Note: directories created by WSL in previous releases do not have this flag set, so will not be treated as case sensitive if you use the "case=dir" option. A way to set this flag on existing directories is coming soon.
- Example of mounting with these options (for existing drives, you must first unmount before you can mount with different options): `sudo mount -t drvfs C: /mnt/c -o case=dir`
- For now, case=force is still the default option. This will be changed to case=dir in the future.
- You can now use forward slashes in Windows paths when mounting DrvFs, e.g.: `sudo mount -t drvfs //server/share /mnt/share`
- WSL now processes the `/etc/fstab` file during instance start [GH 2636].
  - This is done prior to automatically mounting DrvFs drives; any drives that were already mounted by `fstab` will not be remounted automatically, allowing you to change the mount point for specific drives.
  - This behavior can be turned off using `wsl.conf`.
- The `mount`, `mountinfo` and `mountstats` files in `/proc` properly escape special characters like backslashes and spaces [GH 2799]
- Special files created with DrvFs such as WSL symbolic links, or fifos and sockets when metadata is enabled, can now be copied and moved from Windows.

#### **WSL is more configurable with `wsl.conf`**

We added a method for you to automatically configure certain functionality in WSL that will be applied every time you launch the subsystem. This includes automount options and network configuration. Learn more about it in our blog post at: <https://aka.ms/wslconf>

#### **AF\_UNIX allows socket connections between Linux processes on WSL and Windows native processes**

WSL and Windows applications can now communicate with each other over Unix sockets. Imagine you want to run a service in Windows and make it available to both Windows and WSL apps. Now, that's possible with Unix sockets. Read more in our blog post at <https://aka.ms/afunixinterop>

#### **WSL**

- Support `mmap()` with `MAP_NORESERVE` [GH 121, 2784]
- Support `CLONE_PTRACE` and `CLONE_UNTRACED` [GH 121, 2781]
- Handle non-SIGCHLD termination signal in clone [GH 121, 2781]
- Stub `/proc/sys/fs/inotify/max_user_instances` and `/proc/sys/fs/inotify/max_user_watches` [GH 1705]
- Error loading ELF binaries that contain load headers with non-zero offsets [GH 1884]
- Zero out trailing page bytes when loading images.
- Reduce cases where `execve` silently terminates process

#### **Console**

- No fixes.

#### **LTP Results:**

Testing in progress.

## **Build 17083**

For general Windows information on build 17083 visit the [Windows Blog](#).

#### **WSL**

- Fixed bugcheck related to `epoll` [GH 2798, 2801, 2857]
- Fixed hangs when turning off ASLR [GH 1185, 2870]
- Ensure `mmap` operations appear atomic [GH 2732]

#### **Console**

- No fixes.

### LTP Results:

Testing in progress.

## Build 17074

For general Windows information on build 17074 visit the [Windows Blog](#).

### WSL

- Fixed storage format of DrvFs metadata [GH 2777]  
**Important:** DrvFs metadata created before this build will show up incorrectly or not at all. To fix affected files, use `chmod` and `chown` to re-apply the metadata.
- Fixed issue with multiple signals and restartable syscalls.

### Console

- No fixes.

### LTP Results:

Testing in progress.

## Build 17063

For general Windows information on build 17063 visit the [Windows Blog](#).

### WSL

- DrvFs supports additional Linux metadata. This allows setting the owner and mode of files using `chmod`/`chown`, and also the creation of special files such as fifos, unix sockets and device files. This is disabled by default for now since it's still experimental. **Note:** We fixed a bug in the metadata format used by DrvFs. While metadata works on this build for experimentation, future builds will not correctly read metadata created by this build. You might need to manually update owner for modified files and devices with a custom device ID will have to be recreated.

To enable, mount DrvFs with the metadata option (to enable it on an existing mount, you must first unmount it):

```
mount -t drvfs C: /mnt/c -o metadata
```

Linux permissions are added as additional metadata to the file; they do not affect the Windows permissions. Remember, editing a file using a Windows editor may remove the metadata. In this case, the file will revert to its default permissions.

- Added mount options to DrvFs to control files without metadata.
  - `uid`: the user ID used for the owner of all files.
  - `gid`: the group ID used for the owner of all files.
  - `umask`: an octal mask of permissions to exclude for all files and directories.
  - `fmask`: an octal mask of permissions to exclude for all regular files.
  - `dmask`: an octal mask of permissions to exclude for all directories.

For example:

```
mount -t drvfs C: /mnt/c -o uid=1000,gid=1000,umask=22,fmask=111
```

Combine with the metadata option to specify default permissions for files without metadata.

- Introduced a new environment variable, `WSLENV`, to configure how environment variables flow between WSL and Win32.

For example:

```
WSLENV=GOPATH/1:USERPROFILE/pu:DISPLAY
```

`WSLENV` is a colon-delimited list of environment variables that can be included when launching WSL processes from Win32 or Win32 processes from WSL. Each variable can be suffixed with a slash followed by flags to specify how it is translated.

- `p`: The value is a path that should be translated between WSL paths and Win32 paths.
- `l`: The value is a list of paths. In WSL, it is a colon-delimited list. In Win32, it is a semicolon-delimited list.
- `u`: The value should only be included when invoking WSL from Win32
- `w`: The value should only be included when invoking Win32 from WSL

You can set `WSLENV` in `.bashrc` or in the custom Windows environment for your user.

- `drvfs` mounts correctly preserves timestamps from `tar`, `cp -p` (GH 1939)
- `drvfs` symlinks report the correct size (GH 2641)
- `read/write` works for very large IO sizes (GH 2653)
- `waitpid` works with process group IDs (GH 2534)
- significantly improved `mmap` performance for large reserve regions; improves `ghc` performance (GH 1671)
- `personality` supports `READ_IMPLIES_EXEC`; fixes `maxima` and `clisp` (GH 1185)
- `mprotect` supports `PROT_GROWSDOWN`; fixes `clisp` (GH 1128)
- `page fault` fixes in overcommit mode; fixes `sbcl` (GH 1128)
- `clone` supports more flags combinations
- Support `select/epoll` of `epoll` files (previously a no-op).
- Notify `ptrace` of unimplemented syscalls.
- Ignore interfaces that are not up when generating `resolv.conf` nameservers [GH 2694]
- Enumerate network interfaces with no physical address. [GH 2685]
- Additional bug fixes and improvements.

### Linux tools available to developers on Windows

- Windows Command line Toolchain includes `bsdtar` (`tar`) and `curl`. Read [this blog](#) to learn more about the addition of these two new tools and see how they're shaping the developer experience on Windows.
- `AF_UNIX` is available in the Windows Insider SDK (17061+). Read [this blog](#) to learn more about `AF_UNIX` and how developers on Windows can use it.

### Console

- No fixes.

### LTP Results:

Testing in progress.

## Build 17046

For general Windows information on build 17046 visit the [Windows Blog](#).

### Fixed

#### WSL

- Allow processes to run without an active terminal. [GH 709, 1007, 1511, 2252, 2391, et al.]

- Better support of CLONE\_VFORK and CLONE\_VM. [GH 1878, 2615]
- Skip TDI filter drivers for WSL networking operations. [GH 1554]
- DrvFs creates NT symlinks when certain conditions are met. [GH 353, 1475, 2602]
  - The link target must be relative, must not cross any mount points or symlinks, and must exist.
  - The user must have SE\_CREATE\_SYMBOLIC\_LINK\_PRIVILEGE (this normally requires you to launch wsl.exe elevated), unless Developer Mode is turned on.
  - In all other situations, DrvFs still creates WSL symlinks.
- Allow running elevated and non-elevated WSL instances simultaneously.
- Support /proc/sys/kernel/yama/ptrace\_scope
- Add wslpath to do WSL<->Windows path conversions. [GH 522, 1243, 1834, 2327, et al.]

```
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'
```

#### Console

- No fixes.

#### LTP Results:

Testing in progress.

## Build 17040

For general Windows information on build 17040 visit the [Windows Blog](#).

#### Fixed

##### WSL

- No fixes since 17035.

#### Console

- No fixes since 17035.

#### LTP Results:

Testing in progress.

## Build 17035

For general Windows information on build 17035 visit the [Windows Blog](#).

#### Fixed

##### WSL

- Accessing files on DrvFs could occasionally fail with EINVAL. [GH 2448]

#### Console

- Some color loss when inserting/deleting lines in VT mode.

#### LTP Results:

Testing in progress.

## Build 17025

For general Windows information on build 17025 visit the [Windows Blog](#).

#### Fixed

##### WSL

- Start initial processes in a new foreground process group [GH 1653, 2510].
- SIGHUP delivery fixes [GH 2496].
- Generate default virtual bridge name if none provided [GH 2497].
- Implement /proc/sys/kernel/random/boot\_id [GH 2518].
- More interop stdout/stderr pipe fixes.
- Stub syncfs system call.

#### Console

- Fix input VT translation for third party consoles [GH 111]

#### LTP Results:

Testing in progress.

## Build 17017

For general Windows information on build 17017 visit the [Windows Blog](#).

#### Fixed

##### WSL

- Ignore empty ELF program headers [GH 330].
- Allow LxssManager to create WSL instances for non-interactive users (ssh and scheduled task support) [GH 777, 1602].
- Support WSL->Win32->WSL ("inception") scenarios [GH 1228].
- Limited support for termination of console apps invoked via interop [GH 1614].
- Support mount options for devpts [GH 1948].
- Ptrace blocking child startup [GH 2333].
- EPOLLET missing some events [GH 2462].
- Return more data for PTRACE\_GETSIGINFO.
- Getdents with lseek gives incorrect results.
- Fix some Win32 interop app hangs, waiting for input on a pipe that has no more data.
- O\_ASYNC support for tty/pty files.
- Additional improvements and bug fixes

#### Console

- No Console related changes in this release.

#### LTP Results:

Testing in progress.

## Fall Creators Update

## Build 16288

For general Windows information on build 16288 visit the [Windows Blog](#).

#### Fixed

##### WSL

- Correctly initialize and report uid, gid, and mode for socket file descriptors [GH 2490]
- Additional improvements and bug fixes

#### Console

- No Console related changes in this release.

#### LTP Results:

No change since 16273

# Build 16278

For general Windows information on build 162738 visit the [Windows Blog](#).

## Fixed

### WSL

- Explicitly unmap mapped views of file backed sections when tearing down LX MM state [GH 2415]
- Additional improvements and bug fixes

### Console

- No Console related changes in this release.

## LTP Results:

No change since 16273

# Build 16275

For general Windows information on build 162735 visit the [Windows Blog](#).

## Fixed

### WSL

- No WSL related changes in this release.

### Console

- No Console related changes in this release.

## LTP Results:

No change since 16273

# Build 16273

For general Windows information on build 16273 visit the [Windows Blog](#).

## Fixed

### WSL

- Fixed an issue where DrvFs sometimes reported the wrong file type for directories [GH 2392]
- Allow creation of NETLINK\_KOBJECT\_UEVENT sockets to unblock programs that use uevent [GH 1121, 2293, 2242, 2295, 2235, 648, 637]
- Add support for non-blocking connect [GH 903, 1391, 1584, 1585, 1829, 2290, 2314]
- Implement CLONE\_FS clone system call flag [GH 2242]
- Fix issues around not handling tabs or quotes correctly in NT interop [GH 1625, 2164]
- Resolve access denied error when trying to re-launch WSL instances [GH 651, 2095]
- Implement futex FUTEX\_REQUEUE and FUTEX\_CMP\_REQUEUE operations [GH 2242]
- Fix permissions for various SysFs files [GH 2214]
- Fix Haskell stack hang during setup [GH 2290]
- Implement binfmt\_misc 'C' 'O' and 'P' flags [GH 2103]
- Add /proc/sys/kernel/shmmax/shmmni & /threads-max [GH 1753]
- Add partial support for ioprio\_set system call [GH 498]
- Stub SO\_REUSEPORT & adding support for SO\_PASSCRED for netlink sockets [GH 69]
- Return different error codes from RegisterDistribution if a distribution is currently being installed or uninstalled.
- Allow unregistration of partially installed WSL distributions via wslconfig.exe
- Fix python socket test hang from udp::msg\_peek
- Additional improvements and bug fixes

### Console

- No Console related changes in this release.

## LTP Results:

Total Tests: 1904

Total Skipped Tests: 209

Total Failures: 229

[LTP Test Run Logs](#)

# Build 16257

For general Windows information on build 16257 visit the [Windows Blog](#).

## Fixed

### WSL

- Implement prlimit64 system call
- Add support for ulimit -n (setrlimit RLIMIT\_NOFILE) [GH 1688]
- Stub MSG\_MORE for TCP sockets [GH 2351]
- Fix invalid AT\_EXECFN auxiliary vector behavior [GH 2133]
- Fix copy/paste behavior for console/tty, and add better full buffer handling [GH 2204, 2131]
- Set AT\_SECURE in auxiliary vector for set-user-ID and set-group-ID programs [GH 2031]
- Psuedo-terminal master endpoint not handling TIOCPGRP [GH 1063]
- Fix lseek does to rewind directories in Lxfs [GH 2310]
- /dev/ptmx locks up after heavy usage [GH 1882]
- Additional improvements and bug fixes

### Console

- Fix for horizontal Lines/Underscores Everywhere [GH 2168]
- Fix for process order changed making NPM harder to close [GH 2170]
- Added our new color scheme: <https://blogs.msdn.microsoft.com/commandline/2017/08/02/updating-the-windows-console-colors/>

## LTP Results:

No change since 16251

## Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

`prlimit64`

## Known Issues

[GitHub Issue 2392: Windows Folders not recognized by WSL ...](#)

In build 16257, WSL has issues when enumerating Windows files/folders via `/mnt/c/...`. This issue has been fixed and should be released in Insiders build during week commencing 8/14/2017.

# Build 16251

For general Windows information on build 16251 visit the [Windows Blog](#).

## Fixed

### WSL

- Remove beta tag from WSL optional component, see [blog post](#) for details.
- Correctly initialize saved-set uid and gid for set-user-ID and set-group-ID binaries on exec [GH 962, 1415, 2072]
- Added support for ptrace PTRACE\_O\_TRACEEXIT [GH 555]
- Added support for ptrace PTRACE\_GETFPREGS and PTRACE\_GETREGSET with NT\_FPREGSET [GH 555]
- Fixed ptrace to stop on ignored signals

- Additional improvements and bug fixes

#### Console

- No Console related changes in this release.

#### LTP Results:

Number of Passing Tests: 768

Number of Failing Tests: 244

Number of Skipped Tests: 96

[LTP Test Run Logs](#)

## Build 16241

For general Windows information on build 16241 visit the [Windows Blog](#).

#### Fixed

##### WSL

- No WSL related changes in this release.

##### Console

- Fix for outputting the wrong character for the crossing-lines DEC, originally reported [here](#)
- Fix for no output text being displayed in codepage 65001 (utf8)
- Do not transfer changes made to one color's RGB values to other parts of the palette on selection change. This will make the console properties sheet a lot easier to use.
- Ctrl+S doesn't appear to work correctly
- Un-Bold/-Dim completely absent from ANSI escape codes [GH 2174]
- Console doesn't correctly support Vim color themes [GH 1706]
- Cannot paste particular characters [GH 2149]
- Reflow resize interacts strangely with resizing a bash window when stuff is on the edit/command line [GH ConEmu 1123]
- Ctrl-L leaves the screen dirty [GH 1978]
- Console rendering bug when displaying VT on HDPI [GH 1907]
- Japanese characters look strange with Unicode Character U+30FB [GH 2146]
- Additional improvements and bug fixes

## Build 16237

For general Windows information on build 16237 visit the [Windows Blog](#).

#### Fixed

- Use default attributes for files without EAs in lxf (root, root, 0000)
- Added support for distributions that use extended attributes
- Fix padding for entries returned by getdents and getdents64
- Fix permissions check for the shmctl SHM\_STAT system call [GH 2068]
- Fixed incorrect initial epoll state for ttys [GH 2231]
- Fix DrvFs readdir not returning all entries [GH 2077]
- Fix LxFs readdir when files are unlinked [GH 2077]
- Allow unlinked drvfs files to be reopened through procs
- Added global registry key override for disabling WSL features (interop / drive mounting)
- Fix incorrect block count in "stat" for DrvFs (and LxFs) [GH 1894]
- Additional improvements and bug fixes



## Build 16232

For general Windows information on build 16232 visit the [Windows Blog](#).

### Fixed

- No WSL related changes in this release.

## Build 16226

For general Windows information on build 16226 visit the [Windows Blog](#).

### Fixed

- xattr related syscalls support (getxattr, setxattr, listxattr, removexattr).
- security.capability xattr support.
- Improved compatibility with certain file systems and filters, including non-MS SMB servers. [GH #1952]
- Improved support for OneDrive placeholders, GVFS placeholders, and Compact OS compressed files.
- Additional improvements and bug fixes

## Build 16215

For general Windows information on build 16215 visit the [Windows Blog](#).

### Fixed

- WSL no longer requires developer mode.
- Support directory junctions in drvfs.
- Handle uninstalling of WSL distribution appx packages.
- Update procfs to show private and shared mappings.
- Add ability for wslconfig.exe to clean up distributions that are partially installed or uninstalled.
- Added support for IP\_MTU\_DISCOVER for TCP sockets. [GH 1639, 2115, 2205]
- Infer protocol family for routes to AF\_INET.
- Serial device improvements [GH 1929].

## Build 16199

For general Windows information on build 16199 visit the [Windows Blog](#).

### Fixed

- No WSL related changes in these releases.

## Build 16193

For general Windows information on build 16193 visit the [Windows Blog](#).

### Fixed

- Race condition between sending SIGCONT and a threadgroup terminating [GH 1973]
- change tty and pty devices to report FILE\_DEVICE\_NAMED\_PIPE instead of FILE\_DEVICE\_CONSOLE [GH 1840]
- SSH fix for IP\_OPTIONS

- Moved DrvFs mounting to init daemon [GH 1862, 1968, 1767, 1933]
- Added support in DrvFs for following NT symlinks.

## Build 16184

For general Windows information on build 16184 visit the [Windows Blog](#).

### Fixed

- Removed apt package maintenance task (lxcrun.exe /update)
- Fixed output not showing up in from Windows processes in node.js [GH 1840]
- Relax alignment requirements in lxc core [GH 1794]
- Fixed handling of the AT\_EMPTY\_PATH flag in a number of system calls.
- Fixed issue where deleting DrvFs files with open handles will cause the file to exhibit undefined behavior [GH 544,966,1357,1535,1615]
- /etc/hosts will now inherit entries from the Windows hosts file (%windir%\system32\drivers\etc\hosts) [GH 1495]

## Build 16179

For general Windows information on build 16179 visit the [Windows Blog](#).

### Fixed

- No WSL changes this week.

## Build 16176

For general Windows information on build 16176 visit the [Windows Blog](#).

### Fixed

- [Enabled serial support](#)
- Added IP socket option IP\_OPTIONS [GH 1116]
- Implemented pwritev function (while uploading file to nginx/PHP-FPM) [GH 1506]
- Added IP socket options IP\_MULTICAST\_IF & IPV6\_MULTICAST\_IF [GH 990]
- Support for socket option IP\_MULTICAST\_LOOP & IPV6\_MULTICAST\_LOOP [GH 1678]
- Added IP(V6)\_MTU socket option for apps node, traceroute, dig, nslookup, host
- Added IP socket option IPV6\_UNICAST\_HOPS
- [Filesystem Improvements](#)
  - Allow mounting of UNC paths
  - Enable CDFS support in drvfs
  - Correctly handle permissions for network file systems in drvfs
  - Add support for remote drives to drvfs
  - Enable FAT support in drvfs
- Additional fixes and Improvements

### LTP Results

No changes since 15042

## Build 16170

For general Windows information on build 16170 visit the [Windows Blog](#).

We released a new [blog post](#) discussing our efforts to test WSL.

#### Fixed

- Support socket option IP\_ADD\_MEMBERSHIP & IPV6\_ADD\_MEMBERSHIP [GH 1678]
- Add support for PTRACE\_OLDSETOPTIONS. [GH 1692]
- Additional fixes and improvements

#### LTP Results

No changes since 15042

## Build 15046 to Windows 10 Creators Update

There are no more WSL fixes or features planned for inclusion in the Creators Update to Windows 10. Release notes for WSL will resume in the coming weeks for additions targeting the next major Windows Update. For general Windows information on build 15046 and future Insider releases visit the [Windows Blog](#).

## Build 15042

For general Windows information on build 15042 visit the [Windows Blog](#).

#### Fixed

- Fix for a deadlock when removing a path ending in ".."
- Fixed an issue where FIONBIO not returning 0 on success [GH 1683]
- Fixed issue with zero-length reads of inet datagram sockets
- Fix possible deadlock due to race condition in drvfs inode lookup [GH 1675]
- Extended support for unix socket ancillary data; SCM\_CREDENTIALS and SCM\_RIGHTS [GH 514, 613, 1326]
- Additional fixes and improvements

#### LTP Results:

Number of Passing Test: 737

Number of non-Passing (failing, skipped, etc...): 255

## Build 15031

For general Windows information on build 15031 visit the [Windows Blog](#).

#### Fixed

- Fixed a bug where time(2) would sporadically misbehave.
- Fixed an issue where \*SIGPROCMASK syscalls could corrupt signal mask.
- Now return full command line length in WSL process creation notification. [GH 1632]
- WSL now reports thread exit through ptrace for GDB hangs. [GH 1196]
- Fixed bug where ptys would hang after heavy tmux IO. [GH 1358]
- Fixed timeout validation in many system calls (futexp, semtimedop, ppoll, sigtimedwait, itimer, timer\_create)
- Added eventfd EFD\_SEMAPHORE support [GH 452]
- Additional fixes and improvements

#### LTP Results:

Number of Passing Test: 737

Number of non-Passing (failing, skipped, etc...): 255

## Build 15025

For general Windows information on build 15025 visit the [Windows Blog](#).

### Fixed

- Fix for bug that broke grep 2.27 [GH 1578]
- Implemented EFD\_SEMAPHORE flag for eventfd2 syscall [GH 452]
- Implemented /proc/[pid]/net/ipv6\_route [GH 1608]
- Signal driven IO support for unix stream sockets [GH 393, 68]
- Support F\_GETPIPE\_SZ and F\_SETPIPE\_SZ [GH 1012]
- Implement recvmmsg() syscall [GH 1531]
- Fixed bug where epoll\_wait() wasn't waiting [GH 1609]
- Implement /proc/version\_signature
- Tee syscall now returns failure if both file descriptors refer to the same pipe
- Implemented correct behavior for SO\_PEERCREDS for Unix sockets
- Fixed tkill syscall invalid parameter handling
- Changes to increase the performance of drvfs
- Minor fix for Ruby IO blocking
- Fixed recvmmsg() returning EINVAL for the MSG\_DONTWAIT flag for inet sockets [GH 1296]
- Additional fixes and improvements

### LTP Results:

Number of Passing Test: 732

Number of non-Passing (failing, skipped, etc...): 255

[LTP Test Run Logs](#)

## Build 15019

For general Windows information on build 15019 visit the [Windows Blog](#).

### Fixed

- Fixed bug that incorrectly reported CPU usage in procfs for tools like htop (GH 823, 945, 971)
- When calling open() with O\_TRUNC on an existing file inotify now generates IN\_MODIFY before IN\_OPEN
- Fixes to unix socket getsockopt SO\_ERROR to enable postgres [GH 61, 1354]
- Implement /proc/sys/net/core/somaxconn for the GO language
- Apt-get package update background task now runs hidden from view
- Clear scope for ipv6 localhost (Spring-Framework(Java) failure).
- Additional fixes and improvements

### LTP Results:

Number of Passing Test: 714

Number of non-Passing (failing, skipped, etc...): 249

[LTP Test Run Logs](#)

## Build 15014

For general Windows information on build 15014 visit the [Windows Blog](#).

## Fixed

- Ctrl+C now works as intended
- htop and ps auxw now show correct resource utilization (GH #516)
- Basic translation of NT exceptions to signals. (GH #513)
- fallocation now fails with ENOSPC when running out of space instead of EINVAL (GH #1571)
- Added /proc/sys/kernel/sem.
- Implemented semop and semtimedop system calls
- Fixed nslookup errors with IP\_RECVTOS & IPV6\_RECVTCLASS socket option (GH 69)
- Support for socket options IP\_RECVTTL and IPV6\_RECVHOPLIMIT
- Additional fixes and improvements

## LTP Results:

Number of Passing Test: 709

Number of non-Passing (failing, skipped, etc...): 255

[LTP Test Run Logs](#)

## Syscall Summary

Total Syscalls: 384

Total Implemented: 235

Total Stubbed: 22

Total Unimplemented: 127

[Detailed Breakdown](#)

# Build 15007

For general Windows information on build 15007 visit the [Windows Blog](#).

## Known Issue

- There is a known bug where the console does not recognize some Ctrl + input. This includes the ctrl-c command which will act as a normal 'c' keypress.
  - Workaround: Map an alternate key to Ctrl+C. For example, to map Ctrl+K to Ctrl+C do: `stty intr ^k`. This mapping is per terminal and will have to be done *every* time bash is launched. Users can explore the option to include this in their `.bashrc`

## Fixed

- Corrected the issue where running WSL would consume 100% of a CPU core
- Socket option IP\_PKTINFO, IPV6\_RECVPKTINFO now supported. (GH #851, 987)
- Truncate network interface physical address to 16 bytes in lxc (GH #1452, 1414, 1343, 468, 308)
- Additional fixes and improvements

## LTP Results:

Number of Passing Test: 709

Number of non-Passing (failing, skipped, etc...): 255

[LTP Test Run Logs](#)

# Build 15002

For general Windows information on build 15002 visit the [Windows Blog](#).

## Known Issue

Two known issues:

- There is a known bug where the console does not recognize some Ctrl + input. This includes the ctrl-c

command which will act as a normal 'c' keypress.

- Workaround: Map an alternate key to Ctrl+C. For example, to map Ctrl+K to Ctrl+C do: `stty intr ^k`. This mapping is per terminal and will have to be done *every* time bash is launched. Users can explore the option to include this in their `.bashrc`
- While WSL is running a system thread will consume 100% of a CPU core. The root cause has been addressed and fixed internally.

### Fixed

- All bash sessions must now be created at the same permission level. Attempting to start a session at a different level will be blocked. This means admin and non-admin consoles cannot run at the same time. (GH #626)
- Implemented the following NETLINK\_ROUTE messages (requires Windows admin)
  - RTM\_NEWADDR (supports `ip addr add`)
  - RTM\_NEWROUTE (supports `ip route add`)
  - RTM\_DELADDR (supports `ip addr del`)
  - RTM\_DELROUTE (supports `ip route del`)
- Scheduled task checking for packages to update will no longer run on a metered connection (GH #1371)
- Fixed error where piping gets stuck i.e. `bash -c "ls -alR /" | bash -c "cat"` (GH #1214)
- Implemented TCP\_KEEPCNT socket option (GH #843)
- Implemented IP\_MTU\_DISCOVER INET socket option (GH #720, 717, 170, 69)
- Removed legacy functionality to run NT binaries from init with NT path lookup. (GH #1325)
- Fix mode of `/dev/kmsg` to allow group / other read access (0644) (GH #1321)
- Implemented `/proc/sys/kernel/random/uuid` (GH #1092)
- Corrected error where process start time was showing as year 2432 (GH #974)
- Switched default TERM environment variable to xterm-256color (GH #1446)
- Modified the way that process commit is calculated during process fork. (GH #1286)
- Implemented `/proc/sys/vm/overcommit_memory`. (GH #1286)
- Implemented `/proc/net/route` file (GH #69)
- Fixed error where shortcut name was incorrectly localized (GH #696)
- Fixed elf parsing logic that is incorrectly validating the program headers must be less than (or equal to) PATH\_MAX. (GH #1048)
- Implemented statfs callback for procfs, sysfs, cgroupfs, and binfmtfs (GH #1378)
- Fixed AptPackageIndexUpdate windows that won't close (GH #1184, also discussed in GH #1193)
- Added ASLR personality ADDR\_NO\_RANDOMIZE support. (GH #1148, 1128)
- Improved PTRACE\_GETSIGINFO, SIGSEGV, for proper gdb stack traces during AV (GH #875)
- Elf parsing no longer fails for patchelf binaries. (GH #471)
- VPN DNS propagated to `/etc/resolv.conf` (GH #416, 1350)
- Improvements to TCP close for more reliable data transfer. (GH #610, 616, 1025, 1335)
- Now return correct error code when too many files are opened (EMFILE). (GH #1126, 2090)
- Windows Audit log now reports the image name in process create audit.
- Now gracefully fail when launching bash.exe from within a bash window
- Added error message when interop is unable to access a working directory under LxFs (i.e. notepad.exe .bashrc)
- Fixed issue where Windows path was truncated in WSL
- Additional fixes and improvements

### LTP Results:

Number of Passing Test: 690

Number of non-Passing (failing, skipped, etc.): 274

[LTP Test Run Logs](#)

## Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

shmctl

shmget

shmdt

shmat

## Build 14986

For general Windows information on build 14986 visit the [Windows Blog](#).

### Fixed

- Fixed bugchecks with Netlink and Pty IOCTLS
- Kernel version now reports 4.4.0-43 for consistency with Xenial
- Bash.exe now launches when input directed to 'nul:' (GH #1259)
- Thread IDs now reported correctly in procfs (GH #967)
- IN\_UNMOUNT | IN\_Q\_OVERFLOW | IN\_IGNORED | IN\_ISDIR flags now supported in inotify\_add\_watch() (GH #1280)
- Implement timer\_create and related system calls. This enables GHC support (GH #307)
- Fixed issue where ping returned a time of 0.000ms (GH #1296)
- Return correct error code when too many files are opened.
- Fixed issue in WSL where Netlink request for network interface data would fail with EINVAL if the interface's hardware address is 32-bytes (such as the Teredo interface)
  - Note that the Linux "ip" utility contains a bug where it will crash if WSL reports a 32-byte hardware address. This is a bug in "ip", not WSL. The "ip" utility hard-codes the length of the string buffer used to print the hardware address, and that buffer is too small to print a 32-byte hardware address.
- Additional fixes and improvements

### LTP Results:

Number of Passing Test: 669

Number of non-Passing (failing, skipped, etc...): 258

[LTP Test Run Logs](#)

## Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

timer\_create

timer\_delete

timer\_gettime

timer\_settime

## Build 14971

For general Windows information on build 14971 visit the [Windows Blog](#).

### Fixed

- Due to circumstances beyond our control there are no updates in this build for the Windows Subsystem for Linux. Regularly scheduled updates will resume on the next release.

### LTP Results:

Unchanged from 14965

Number of Passing Test: 664

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14965

For general Windows information on build 14965 visit the [Windows Blog](#).

### Fixed

- Support for Netlink sockets NETLINK\_ROUTE protocol's RTM\_GETLINK and RTM\_GETADDR (GH #468)
  - Enables ifconfig and ip commands for network enumeration
  - More information can be found in our [WSL Networking blog post](#).
- /sbin is now in the user's path by default
- NT user path now appended to the WSL path by default (i.e. you can now type notepad.exe without adding System32 to the Linux path)
- Added support for /proc/sys/kernel/cap\_last\_cap
- NT Binaries can now be launched from WSL when the current working directory contains non-ansi characters (GH #1254)
- Allow shutdown on disconnected unix stream socket.
- Added support for PR\_GET\_PDEATHSIG.
- Added support for CLONE\_PARENT
- Fixed error where piping gets stuck i.e. `bash -c "ls -alR /" | bash -c "cat"` (GH #1214)
- Handle requests to connect to the current terminal.
- Mark /proc//oom\_score\_adj as writable.
- Add /sys/fs/cgroup folder.
- sched\_setaffinity should return number of affinity bits mask
- Fix ELF validation logic which incorrectly assumes interpreter paths must be less than 64 characters long. (GH #743)
- Open file descriptors can keep console window open (GH #1187)
- Fixed error where rename() failed with trailing slash on target name (GH #1008)
- Implement /proc/net/dev file
- Fixed 0.000ms pings due to timer resolution.
- Implemented /proc/self/environ (GH #730)
- Additional bugfixes and improvements

### LTP Results:

Number of Passing Test: 664

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14959



For general Windows information on build 14959 visit the [Windows Blog](#).

#### Fixed

- Improved Pico Process notification for Windows. Additional information found on the [WSL Blog](#).
- Improved stability with Windows interoperability
- Fixed error 0x80070057 when launching bash.exe when Enterprise Data Protection (EDP) is enabled
- Additional bugfixes and improvements

#### LTP Results:

Number of Passing Test: 665

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14955

For general Windows information on build 14955 visit the [Windows Blog](#).

#### Fixed

- Due to circumstances beyond our control there are no updates in this build for the Windows Subsystem for Linux. Regularly scheduled updates will resume on the next release.

#### LTP Results:

Number of Passing Test: 665

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14951

For general Windows information on build 14951 visit the [Windows Blog](#).

#### New Feature: Windows / Ubuntu Interoperability

Windows binaries can now be invoked directly from the WSL command line. This gives users the ability to interact with their Windows environment and system in a way that has not been possible. As a quick example, it is now possible for users to run the following commands:

```
$ 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
```

More information can be found at:

- [WSL Team Blog for Interop](#)
- [MSDN Interop Documentation](#)

#### Fixed

- Ubuntu 16.04 (Xenial) is now installed for all new WSL instances. Users with existing 14.04 (Trusty) instances will not be automatically upgraded.
- Locale set during install is now displayed
- Terminal improvements including bug where redirecting a WSL process to a file does not always work
- Console lifetime should be tied to bash.exe lifetime
- Console window size should use visible size, not buffer size
- Additional bugfixes and improvements

**LTP Results:**

Number of Passing Test: 665

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14946

For general Windows information on build 14946 visit the [Windows Blog](#).

**Fixed**

- Fixed an issue that prevented creating WSL user accounts for users with NT usernames that contain spaces or quotes.
- Change VolFs and DrvFs to return 0 for directory's link count in stat
- Support IPV6\_MULTICAST\_HOPS socket option.
- Limit to a single console I/O loop per tty. Example: the following command is possible:
  - `bash -c "echo data" | bash -c "ssh user@example.com 'cat > foo.txt'"`
- replace spaces with tabs in /proc/cpuinfo (GH #1115)
- DrvFs now appears in mountinfo with a name that matches mounted Windows volume
- /home and /root now appear in mountinfo with correct names
- Additional bugfixes and improvements

**LTP Results:**

Number of Passing Test: 665

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

## Build 14942

For general Windows information on build 14942 visit the [Windows Blog](#).

**Fixed**

- A number of bugchecks addressed, including the "ATTEMPTED EXECUTE OF NOEXECUTE MEMORY" networking crash which was blocking SSH
- inotify support for notifications generated from Windows applications on DrvFs is now in
- Implement TCP\_KEEPIIDLE and TCP\_KEEPINTVL for mongod. (GH #695)
- Implement the pivot\_root system call
- Implement socket option for SO\_DONTROUTE
- Additional bugfixes and improvements

**LTP Results:**

Number of Passing Test: 665

Number of non-Passing (failing, skipped, etc...): 263

[LTP Test Run Logs](#)

**Syscall Support**

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

pivot_root
------------

# Build 14936

For general Windows information on build 14936 visit the [Windows Blog](#).

Note: WSL will install Ubuntu version 16.04 (Xenial) instead of Ubuntu 14.04 (Trusty) in an upcoming release. This change will apply to Insiders installing new instances (lxdm.exe /install or first run of bash.exe). Existing instances with Trusty will not be upgraded automatically. Users can upgrade their Trusty image to Xenial using the do-release-upgrade command.

## Known Issue

WSL is experiencing an issue with some socket implementations. The bugcheck manifests itself as a crash with the error "ATTEMPTED EXECUTE OF NOEXECUTE MEMORY". The most common manifestation of this issue is a crash when using ssh. The root cause is fixed on internal builds and will be pushed to Insiders at the earliest opportunity.

## Fixed

- Implemented the chroot system call
- Improvements in inotify ~~including support for notifications generated from Windows applications on DrvFs~~
  - Correction: Inotify support for changes originating from Windows applications not available at this time.
- Socket binding to IPV6:: now supports IPV6\_V6ONLY (GH #68, #157, #393, #460, #674, #740, #982, #996)
- WNOWAIT behavior for waitid syscall implemented (GH #638)
- Support for IP socket options IP\_HDRINCL and IP\_TTL
- Zero-length read() should return immediately (GH #975)
- Correctly handle filenames and filename prefixes that don't include a NULL terminator in a .tar file.
- epoll support for /dev/null
- Fix /dev/alarm time source
- Bash -c now able to redirect to a file
- Additional bugfixes and improvements

## LTP Results:

Number of Passing Test: 664

Number of non-Passing (failing, skipped, etc...): 264

[LTP Test Run Logs](#)

## Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

chroot

# Build 14931

For general Windows information on build 14931 visit the [Windows Blog](#).

## Fixed

- Due to circumstances beyond our control there are no updates in this build for the Windows Subsystem for Linux. Regularly scheduled updates will resume in the next release.

# Build 14926

For general Windows information on build 14926 visit the [Windows Blog](#).

## Fixed

- Ping now works in consoles which do not have administrator privileges
- Ping6 now supported, also without administrator privileges
- Inotify support for files modified through WSL. (GH #216)
  - Flags supported:
    - inotify\_init1: LX\_O\_CLOEXEC, LX\_O\_NONBLOCK
    - inotify\_add\_watch events: 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 attributes: LX\_IN\_DONT\_FOLLOW, LX\_IN\_EXCL\_UNLINK, LX\_IN\_MASK\_ADD, LX\_IN\_ONESHOT, LX\_IN\_ONLYDIR
    - read output: LX\_IN\_ISDIR, LX\_IN\_IGNORED
  - Known issue: Modifying files from Windows applications does not generate any events
- Unix socket now supports SCM\_CREDENTIALS

#### LTP Results:

Number of Passing Test: 651

Number of non-Passing (failing, skipped, etc...): 258

[LTP Test Run Logs](#)

## Build 14915

For general Windows information on build 14915 visit the [Windows Blog](#).

#### Fixed

- Socketpair for unix datagram sockets (GH #262)
- Unix socket support for SO\_REUSEADDR
- UNIX socket support for SO\_BROADCAST (GH #568)
- Unix socket support for SOCK\_SEQPACKET (GH #758, #546)
- Adding support for unix datagram socket send, recv and shutdown
- Fix bugcheck due to invalid mmap parameter validation for non-fixed addresses. (GH #847)
- Support for suspend / resume of terminal states
- Support for TIOCPKT ioctl to unblock the Screen utility (GH #774)
  - Known issue: Function keys not operational
- Corrected a race in TimerFd that could cause a freed member 'ReaderReady' to be accessed by LxpTimerFdWorkerRoutine (GH #814)
- Enable restartable system call support for futex, poll, and clock\_nanosleep
- Added bind mount support
- unshare for mount namespace support
  - Known issue: When creating a new mount namespace with `unshare(CLONE_NEWNS)` the current working directory will continue to point to the old namespace
- Additional improvements and bug fixes

## Build 14905

For general Windows information on build 14905 visit the [Windows Blog](#).

#### Fixed

- Restartable system calls are now supported (GH #349, GH #520)
- Symlinks to directories ending in / now operational (GH #650)
- Implemented RNDGETENTCNT ioctl for /dev/random

- Implemented the /proc/[pid]/mounts, /proc/[pid]/mountinfo and /proc/[pid]/mountstats files
- Additional bugfixes and improvements

## Build 14901

First Insider build for the post Windows 10 Anniversary Update release.

For general Windows information on build 14901 visit the [Windows Blog](#).

### Fixed

- Fixed trailing slash issue
  - Commands such as `$ mv a/c/ a/b/` now work
- Installing now prompts if Ubuntu locale should be set to Windows locale
- Procfs support for ns folder
- Added mount and unmount for tmpfs, procfs and sysfs file systems
- Fix mknod[at] 32-bit ABI signature
- Unix sockets moved to dispatch model
- INET socket recv buffer size set using the setsockopt should be honored
- Implement MSG\_CMSG\_CLOEXEC unix socket receive message flag
- Linux process stdin/stdout pipe redirection (GH #2)
  - Allows for piping of bash -c commands in CMD. Example: `> dir | bash -c "grep foo"`
- Bash can now be installed on systems with multiple pagefiles (GH #538, #358)
- Default INET Socket buffer size should match that of default Ubuntu setup
- Align xattr syscalls to listxattr
- Only return interfaces with a valid IPv4 address from SIOCGIFCONF
- Fix signal default action when injected by ptrace
- implement /proc/sys/vm/min\_free\_kbytes
- Use machine context register values when restoring context in sigreturn
  - This resolves the issue where java and javac were hanging for some users
- Implement /proc/sys/kernel/hostname

### Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

`waitid`

`epoll_pwait`

## Build 14388 to Windows 10 Anniversary Update

For general Windows information on build 14388 visit the [Windows Blog](#).

### Fixed

- Fixes to prepare for the Windows 10 Anniversary Update on 8/2
  - More information about WSL in the Anniversary Update can be found on our [blog](#)

## Build 14376

For general Windows information on build 14376 visit the [Windows Blog](#).

### Fixed

- Removed some instances where apt-get hangs (GH #493)
- Fixed an issue where empty mounts were not handled correctly
- Fixed an issue where ramdisks were not mounted correctly
- Change unix socket accept to support flags (partial GH #451)
- Fixed common network related bluescreen
- Fixed bluescreen when accessing /proc/[pid]/task (GH #523)
- Fixed high CPU utilization for some pty scenarios (GH #488, #504)
- Additional bugfixes and improvements

## Build 14371

For general Windows information on build 14371 visit the [Windows Blog](#).

### Fixed

- Corrected timing race with SIGCHLD and wait() when using ptrace
- Corrected some behavior when paths have a trailing / (GH #432)
- Fixed issue with rename/unlink failing due to open handles to children
- Additional bugfixes and improvements

## Build 14366

For general Windows information on build 14366 visit the [Windows Blog](#).

### Fixed

- Fix in file creation through symlinks
- Added listxattr for Python (GH 385)
- Additional bugfixes and improvements

### Syscall Support

- Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

listxattr

## Build 14361

For general Windows information on build 14361 visit the [Windows Blog](#).

### Fixed

- DrvFs is now case sensitive when running in Bash on Ubuntu on Windows.
  - Users may case.txt and CASE.TXT on their /mnt/c drives
  - Case sensitivity is only supported within Bash on Ubuntu on Windows. When outside of Bash NTFS will report the files correctly, but unexpected behavior may occur interacting with the files from Windows.
  - The root of each volume (i.e. /mnt/c) is not case sensitive
  - More information on handling these files in Windows can be found [here](#).
- Greatly enhanced pty / tty support. Applications like TMUX now supported (GH #40)
- Fixed install issue where user accounts not always created
- Optimized command line arg structure allowing for extremely long argument list. (GH #153)
- Now able to delete and chmod read\_only files from DrvFs
- Fixed some instances where the terminal hangs on disconnect (GH #43)

- chmod and chown now work on tty devices
- Allow connection to 0.0.0.0 and :: as localhost (GH #388)
- Sendmsg/recvmmsg now handle an IO vector length of > 1 (partial GH #376)
- Users can now opt-out of auto-generated hosts file (GH #398)
- Automatically match Linux locale to the NT locale during install (GH #11)
- Added the /proc/sys/vm/swappiness file (GH #306)
- strace now exits correctly
- Allow pipes to be reopened through /proc/self/fd (GH #222)
- Hide directories under %LOCALAPPDATA%\lxss from DrvFs (GH #270)
- Better handling of bash.exe ~. Commands like "bash ~ -c ls" now supported (GH #467)
- Sockets now notify epoll read available during shutdown (GH #271)
- lxr/uninstall does a better job of deleting the files and folders
- Corrected ps -f (GH #246)
- Improved support for x11 apps such as xEmacs (GH #481)
- Updated initial thread stack size to match default Ubuntu setting and reporting the size correctly to the get\_rlimit syscall (GH #172, #258)
- Improved reporting of pico process image names (e.g., for auditing)
- Implemented /proc/mountinfo for df command
- Fixed symlink error code for child name . and ..
- Additional improvements bugfixes and improvements

### Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

GETTIMER
MKNODAT
RENAMEAT
SENDFILE
SENDFILE64
SYNC_FILE_RANGE

## Build 14352

For general Windows information on build 14352 visit the [Windows Blog](#).

### Fixed

- Fixed issue where large files were not downloaded / created correctly. This should unblock npm and other scenarios (GH #3, GH #313)
- Removed some instances where sockets hang
- Corrected some ptrace errors
- Fixed issue with WSL allowing filenames longer than 255 characters
- Improved support for non-English characters
- Add current Windows timezone data and set as default
- Unique device id's for each mount point (jre fix – GH #49)
- Correct issue with paths containing "." and ".."
- Added Fifo support (GH #71)
- Updated format of resolv.conf to match native Ubuntu format
- Some procfs cleanup
- Enabled ping for Administrator consoles (GH #18)

### Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

FALLOCATE

EXECVE

LGETXATTR

FGETXATTR

## Build 14342

For general Windows information on build 14342 the [Windows Blog](#).

Information on VolFs and DriveFs can be found on the [WSL Blog](#).

### Fixed

- Fixed install issue when the Windows user had Unicode characters in the username
- The apt-get update udev workaround in the FAQ is now provided by default on first run
- Enabled symlinks in DriveFs (/mnt/) directories
- Symlinks now work between DriveFs and VolFs
- Addressed top level path parsing issue: ls ./ will now work as expected
- npm install on DriveFs and the -g options are now working
- Fixed issue preventing PHP server from launching
- Updated default environment values, such as \$PATH to closer match native Ubuntu
- Added a weekly maintenance task in Windows to update the apt package cache
- Fixed issue with ELF header validation, WSL now supports all Melkor options
- Zsh shell is functional
- Precompiled Go binaries are now supported
- Prompting on Bash.exe first run is now localized correctly
- /proc/meminfo now returns correct information
- Sockets now supported in VFS
- /dev now mounted as tempfs
- Fifo now supported
- Multi-core systems now showing correctly in /proc/cpuinfo
- Additional improvements and error messages downloading during first run
- Syscall improvements and bugfixes. Supported syscall list below.
- Additional bugfixes and improvements

### Known Issues

- Not resolving '..' correctly on DriveFs in some cases

### Syscall Support

Below are a list of new or enhanced syscalls that have some implementation in WSL. The syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

FCHOWNAT

GETEUID

GETGID

GETRESUID

GETXATTR

PTRACE

SETGID

SETGROUPS

SETHOSTNAME

SETXATTR



# Build 14332

For general Windows information on build 14332 visit the [Windows Blog](#).

## Fixed

- Better resolv.conf generation including prioritizing DNS entries
- Issue with moving files and directories between /mnt and non-/mnt drives
- Tar files can now be created with symlinks
- Added default /run/lock directory on instance creation
- Update /dev/null to return proper stat info
- Additional errors when downloading during first run
- Syscall improvements and bugfixes. Supported syscall list below.
- Additional improvements bugfixes and improvements

## Syscall Support

Below is the new syscall that has some implementation in WSL. The syscall on this list is supported in at least one scenario, but may not have all parameters supported at this time.

READLINKAT

# Build 14328

For general Windows information on build 14332 visit the [Windows Blog](#).

## New Features

- Now support Linux users. Installing Bash on Ubuntu on Windows will prompt for creation of a Linux user. For more information, visit <https://aka.ms/wslusers>
- Hostname is now set to the Windows computer name, no more @localhost
- For more information on build 14328, visit: <https://aka.ms/wip14328>

## Fixed

- Symlink improvements for non /mnt/ files
  - npm install now works
  - jdk / jre now installable using instructions found [here](#).
  - known issue: symlinks do not work for Windows mounts. Functionality will be available in a later build
- top and htop now display
- Additional error messages for some install failures
- Syscall improvements and bugfixes. Supported syscall list below.
- Additional improvements bugfixes and improvements

## Syscall Support

Below is a list of syscalls that have some implementation in WSL. Syscalls on this list are supported in at least one scenario, but may not have all parameters supported at this time.

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  
SENDMSG  
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

# Release Notes for Windows Subsystem for Linux kernel

4/22/2021 • 2 minutes to read • [Edit Online](#)

We've added support for WSL 2 distributions, [which use a full Linux kernel](#). This Linux kernel is open source, with its source code available at the [WSL2-Linux-Kernel](#) repository. This Linux kernel is delivered to your machine via Microsoft Update, and follows a separate release schedule to the Windows Subsystem for Linux which is delivered as part of the Windows image.

## 5.10.16.3

*Release Date:* Prerelease 2021/04/16

[Official Github release link](#)

- Fixes [GH 5324](#)
- Adds support for LUKS encrypted disks using `wsl --mount`

## 5.4.91

*Release Date:* Prerelease 2021/02/22

[Official Github release link](#)

## 5.4.72

*Release Date:* 2021/01/21

[Official Github release link](#)

- Fix config for 5.4.72

## 5.4.51-microsoft-standard

*Release Date:* Prerelease - 2020/10/22

[Official Github release link.](#)

- Stable release of 5.4.51

## 4.19.128-microsoft-standard

*Release Date:* 2020/09/15

[Official Github release link.](#)

- This is a stable release of 4.19.128
- Fix dxgkrnl driver IOCTL memory corruption

## 4.19.121-microsoft-standard

*Release Date:* Prerelease

[Official Github release link.](#)

- Drivers: hv: vmbus: hook up dxgkrnl
- Added support for GPU Compute

## 4.19.104-microsoft-standard

*Release Date:* 2020/06/09

[Official Github release link.](#)

- Update WSL config for 4.19.104

## 4.19.84-microsoft-standard

*Release Date:* 2019/12/11

[Official Github release link.](#)

- This is the 4.19.84 stable release