Contents

Set up your development environment on Windows

6/30/2021 • 3 minutes to read • Edit Online

Windows invites you to code as you are. Use whatever coding language or framework you prefer - whether developing with tools on Windows or with Linux tools on the Windows Subsystem for Linux, this guide will help you get set up and install what you need to start coding, debugging, and accessing services to put your work into production.

Development paths

Get started with JavaScript

Get started with JavaScript by setting up your development environment on Windows or Windows Subsystem for Linux and install Node.js, React, Vue, Express, Gatsby, Next.js, or Nuxt.js.

Get started with Python

Install Python and get your development environment setup on Windows or Windows Subsystem for Linux.

Get started with Android

Install Android Studio, or choose a cross-platform solution like Xamarin, React, or Cordova, and get your development environment setup on Windows.

Get started with Windows Desktop

Get started building desktop apps for Windows using the Windows App SDK, UWP, Win32, WPF, Windows Forms, or updating and deploying existing desktop apps with MSIX and XAML Islands.

Get started with C++ and C

Get started with C++, C, and assembly to develop apps, services, and tools.

Get started with C#

Get started building apps using C# and .NET Core.

Get started with Docker Desktop for Windows

Create remote development containers with support from Visual Studio, VS Code, .NET, Windows Subsystem for Linux, or a variety of Azure services.

Get started with PowerShell
Get started with cross-platform task automation and configuration management using PowerShell, a command-line shell and scripting language.

Get started with Rust
Get started programming with Rust—including how to set up Rust for Windows by consuming the *windows* crate.

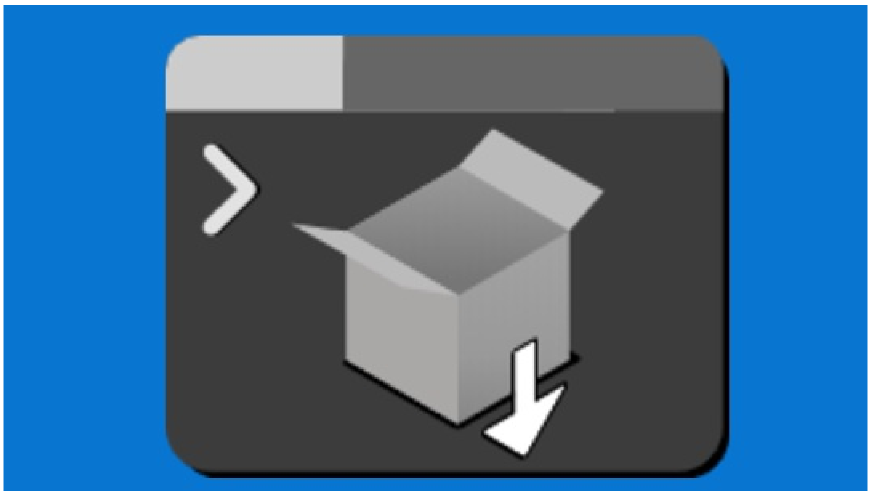
Tools and platforms

Windows Subsystem for Linux
Use your favorite Linux distribution fully integrated with Windows (no more need for dual-boot).
Install WSL

Windows Terminal

Customize your terminal environment to work with multiple command line shells.

Install Terminal



Windows Package Manager

Use the winget.exe client, a comprehensive package manager, with your command line to install applications on Windows.

Install Windows Package Manager (public preview)

Microsoft PowerToys

Tune and streamline your Windows experience for greater productivity with this set of power user utilities.

Install PowerToys

VS Code

A lightweight source code editor with built-in support for JavaScript, TypeScript, Node.js, a rich ecosystem of extensions (C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity).

Install VS Code

Visual Studio

An integrated development environment that you can use to edit, debug, build code, and publish apps, including compilers, intellisense code completion, and many more features.

Install Visual Studio

Azure

A complete cloud platform to host your existing apps and streamline new development. Azure services integrate everything you need to develop, test, deploy, and manage your apps.

Set up an Azure account

.NET
An open source development platform with tools and libraries for building any type of app, including web, mobile, desktop, gaming, IoT, cloud, and microservices.
Install .NET

Run Windows and Linux

Windows Subsystem for Linux (WSL) allows developers to run a Linux operating system right alongside Windows. Both share the same hard drive (and can access each other's files), the clipboard supports copy-and-paste between the two naturally, there's no need for dual-booting. WSL enables you to use BASH and will provide the kind of environment most familiar to Mac users.

- Learn more in the WSL docs or via WSL videos on Channel 9.

You can also use Windows Terminal to open all of your favorite command line tools in the same window with multiple tabs, or in multiple panes, whether that's PowerShell, Windows Command Prompt, Ubuntu, Debian, Azure CLI, Oh-my-Zsh, Git Bash, or all of the above.

Learn more in the Windows Terminal docs or via Windows Terminal videos on Channel 9.

Transitioning between Mac and Windows

Check out our guide to transitioning between between a Mac and Windows (or Windows Subsystem for Linux) development environment. It can help you map the difference between:

- Keyboard shortcuts
- Trackpad shortcuts
- Terminal and shell tools
- Apps and utilities

Additional resources

- Tips for improving your workflow
- Stories from developers who have switched from Mac to Windows
- Popular tutorials, courses, and code samples
- Microsoft's Game Stack documentation

Tips for improving performance and development workflows

6/30/2021 • 3 minutes to read • Edit Online

We've gathered a few tips that we hope will help to make your workflow more efficient and enjoyable. Do you have additional tips to share? File a pull request, using the "Edit" button above, or an issue, using the "Feedback" button below and we may add it to the list.

NOTE

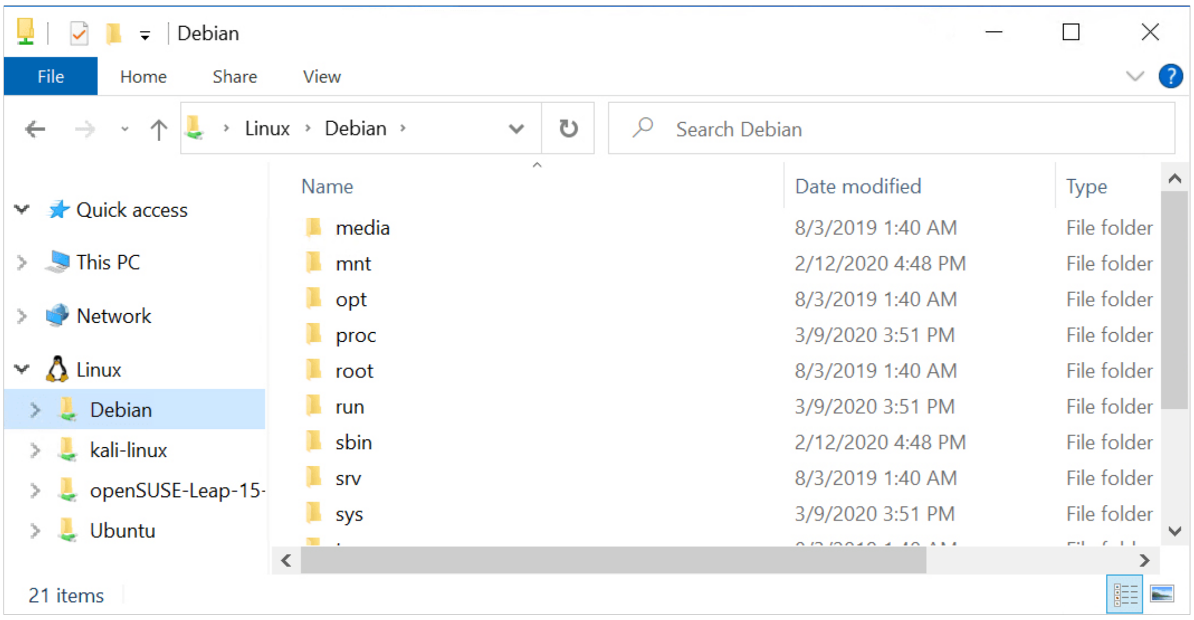
If you are experiencing any performance issues related to developing on Windows 10, such as:

- Dev tools (e.g. compilers, linkers, etc.) running slower on Windows than expected.
- Runtime platforms (e.g. node, .NET, Python) running slower on Windows than other platforms.
- Your apps experiencing file IO/networking/process-creation related perf issues.

Please let us know by filing an issue in the Windows Developer (WinDev) Issues repo!

Use shortcuts to open a project in VS Code or Windows File Explorer

You can launch VS Code from your command line into the project that you have open by using the command: `code .` or open your project directory from the command line with Windows File Explorer using `explorer.exe .` from Windows or your WSL distribution. You may need to add the VS Code executable to your PATH environment variable if this doesn't work by default. Learn more about Launching from the Command Line.



Name	Date modified	Type
media	8/3/2019 1:40 AM	File folder
mnt	2/12/2020 4:48 PM	File folder
opt	8/3/2019 1:40 AM	File folder
proc	3/9/2020 3:51 PM	File folder
root	8/3/2019 1:40 AM	File folder
run	3/9/2020 3:51 PM	File folder
sbin	2/12/2020 4:48 PM	File folder
srv	8/3/2019 1:40 AM	File folder
sys	3/9/2020 3:51 PM	File folder

Use the Credential Manager to your streamline authentication process

If you're using Git for version control and collaboration, you can streamline your authentication process by setting up Git Credential Manager to store your tokens in the Windows Credential Manager. We also recommend adding a .gitignore file to your project.

Use WSL for testing your production pipeline before deploying to the cloud

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.

WSL targets a developer audience with the intent to be used as part of an inner development loop. Let's say that Sam is creating a CI/CD pipeline (Continuous Integration & Continuous Delivery) and wants to test it first on a local machine (laptop) before deploying it to the cloud. Sam can enable WSL (& WSL 2 to improve speed and performance), and then use a genuine Linux Ubuntu instance locally (on the laptop) with whatever Bash commands and tools they prefer. Once the development pipeline is verified locally, Sam can then push that CI/CD pipeline up to the cloud (ie Azure) by making it into a Docker container and pushing the container to a cloud instance where it runs on a production-ready Ubuntu VM.

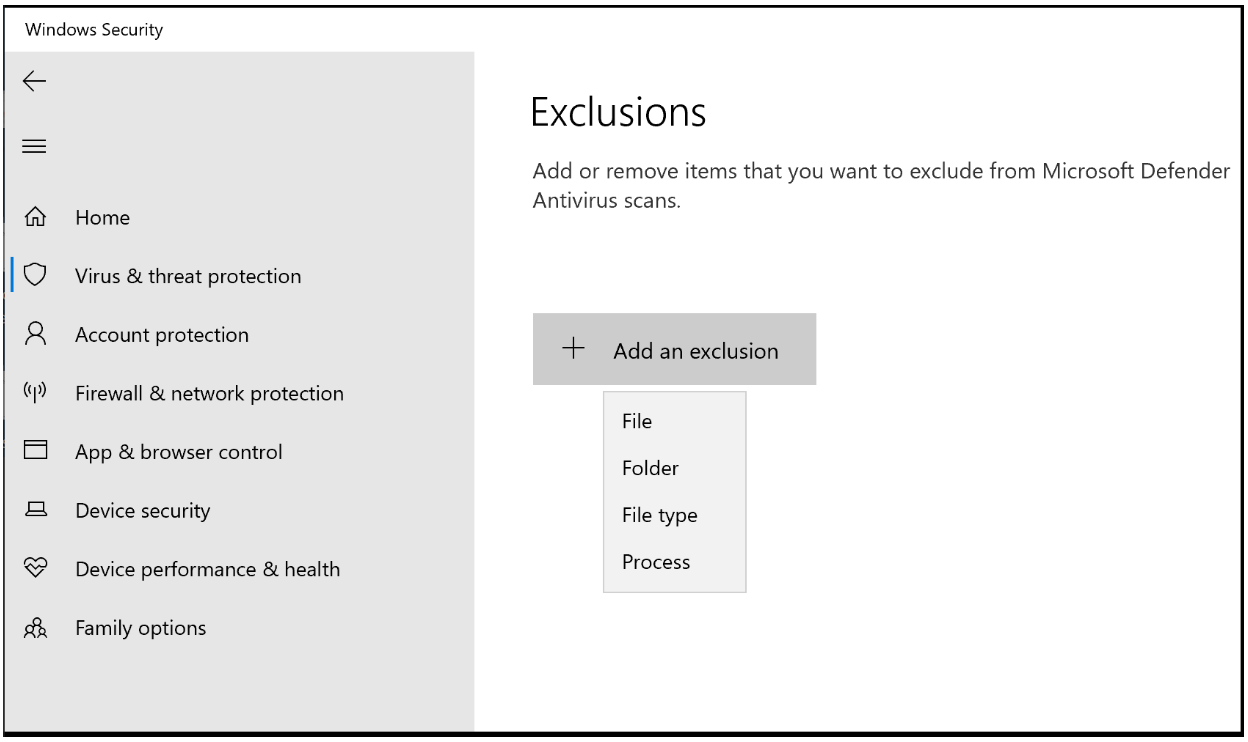
For more ways to use WSL, check out this Tabs vs Spaces episode on WSL 2.

Improve performance speed for WSL by not crossing over file systems

If you're working with both Windows and Windows Subsystem for Linux, you have two file systems installed: NTFS (Windows) and WSL (your Linux distro). For fast performance, ensure that your project files are stored in the same system as the tools you're using. Learn more about choosing the correct file system for faster performance.

Improve build speeds by adding Windows Defender exclusions

You can improve your build speed by updating your Windows Defender settings to add exclusions for project folders or file types that you trust enough to avoid scanning for security threats. Learn more about how to Update Windows Defender settings to improve performance.


Windows Security
Home
Virus & threat protection
Account protection
Firewall & network protection
App & browser control
Device security
Device performance & health
Family options
Exclusions
Add or remove items that you want to exclude from Microsoft Defender Antivirus scans.
Add an exclusion
File
Folder
File type
Process

Launch all your command lines in Windows Terminal at once

- You can launch multiple command lines, like PowerShell, Ubuntu, and Azure CLI, all into a single window with multiple panes using Windows Terminal Command Line Arguments. After installing Windows Terminal, WSL/Ubuntu, and Azure CLI, enter this command in PowerShell to open a new multi-pane window with all three:

  ```
  wt -p "Command Prompt" `; split-pane -p "Windows PowerShell" `; split-pane -H wsl.exe
  ```

Share your tips

Do you have tips for to help other developers using Windows improve their workflow? Please submit a pull request adding your tip to the page or file an issue if you'd like use to add a tip on a particular topic.

Do you have performance-related issues that you would like us to address? File it in the new WinDev Issues repo.

Thank you developers. We're listening and trying to improve your experience!

Stories from developers who have switched from Mac to Windows

3/6/2021 • 2 minutes to read • Edit Online

We thought it may be helpful to hear from other developers about their experiences switching between a Mac and Windows development environment. Most found the process reasonably simple, appreciated that they could still use their favorite Linux and open source tools, while also having integrated access to Windows productivity tools, like Microsoft Office, Outlook, and Teams. Here are a few articles and blog entries that we found:

- Ken Wang, "Think Different — Software Developer Switching from Mac to Windows"
- Owen Williams, "The state of switching to Windows from Mac in 2019"
- August Lilleaas, "Why I ditched macOS, Linux, and chose Windows for development work"
- Brent Rose, "What Happened When I Switched From Mac to Windows"
- Jack Franklin, "Using Windows 10 and WSL for frontend web development"
- Aaron Schlesinger, "Coming from a Mac to Windows & WSL 2"
- David Heinemeier Hansson, "Back to windows after twenty years"
- Ray Elenteny, "Why I returned to Windows"

Popular tutorials, courses, and code samples

5/10/2021 • 2 minutes to read • Edit Online

We've listed a few tutorials, course, and code samples below to help you get started on some common development tasks and scenarios.

Create a database

- Create a MongoDB app with React and Azure Cosmos DB
- Deploy a Python (Django) web app with PostgreSQL in Azure App Service

Build an Android app

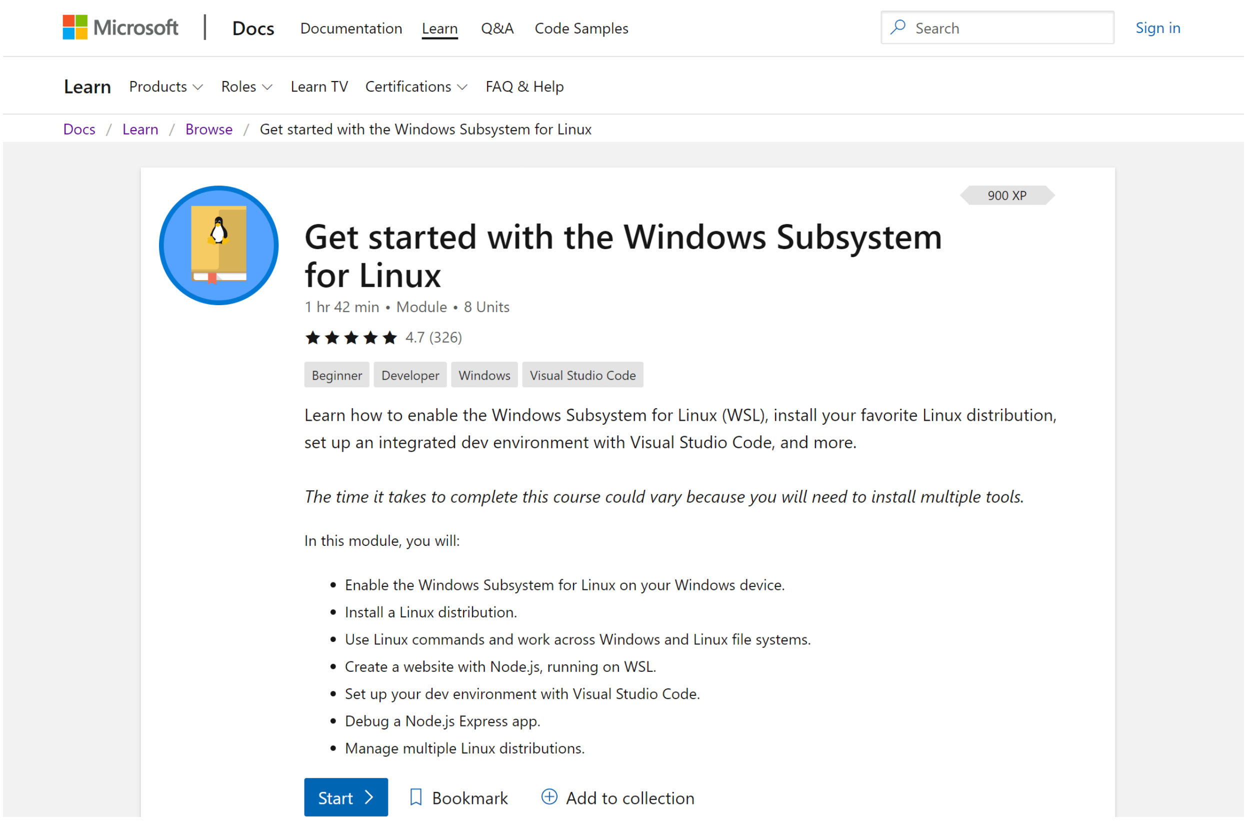
- Build an Android dual-screen app with drag and drop capabilities

Create a cross-platform app

- Build a to-do list cross-platform app with Xamarin.Forms
- Build a Xamarin.Android app that utilizes Google Play Services to demo the Google Maps API

Get started with WSL

- An introductory to Windows Subsystem for Linux


Microsoft | Docs Documentation Learn Q&A Code Samples
Search
Sign in
Learn Products Roles Learn TV Certifications FAQ & Help
Docs / Learn / Browse / Get started with the Windows Subsystem for Linux
900 XP
Beginner Developer Windows Visual Studio Code
Start
Bookmark
Add to collection

Get started with the Windows Subsystem for Linux

1 hr 42 min • Module • 8 Units

★★★★★ 4.7 (326)

Learn how to enable the Windows Subsystem for Linux (WSL), install your favorite Linux distribution, set up an integrated dev environment with Visual Studio Code, and more.

The time it takes to complete this course could vary because you will need to install multiple tools.

In this module, you will:

- Enable the Windows Subsystem for Linux on your Windows device.
- Install a Linux distribution.
- Use Linux commands and work across Windows and Linux file systems.
- Create a website with Node.js, running on WSL.
- Set up your dev environment with Visual Studio Code.
- Debug a Node.js Express app.
- Manage multiple Linux distributions.

Build a web app or call an API

- Build your first ASP.Net Core web app with Blazor

- Call an ASP.NET Core Web API from a WPF application using Azure AD V2

Build a console app

- Build a Java app with Microsoft Graph

Create a microservice

- Create and deploy a cloud-native ASP.NET Core microservice

Additional resources

- Explore free online courses on Microsoft Learn
- Explore online courses from Pluralsight

Guide for changing your dev environment from Mac to Windows

5/10/2021 • 4 minutes to read • Edit Online

The following tips and control equivalents should help you in your transition between a Mac and Windows (or WSL/Linux) development environment.

For app development, the nearest equivalent to Xcode would be Visual Studio. There is also a version of Visual Studio for Mac, if you ever feel the need to go back. For cross-platform source code editing (and a huge number of plug-ins) Visual Studio Code is the most popular choice.

Keyboard shortcuts

OPERATION	MAC	WINDOWS
Copy	Command+C	Ctrl+C
Cut	Command+X	Ctrl+X
Paste	Command+V	Ctrl+V
Undo	Command+Z	Ctrl+Z
Save	Command+S	Ctrl+S
Open	Command+O	Ctrl+O
Lock computer	Command+Control+Q	WindowsKey+L
Show desktop	Command+F3	WindowsKey+D
Open file browser	Command+N	WindowsKey+E
Minimize windows	Command+M	WindowsKey+M
Search	Command+Space	WindowsKey
Close active window	Command+W	Control+W
Switch current task	Command+Tab	Alt+Tab
Maximize a window to full screen	Control+Command+F	WindowsKey+Up
Save screen (Screenshot)	Command+Shift+3	WindowsKey+Shift+S
Save window	Command+Shift+4	WindowsKey+Shift+S
View item information or properties	Command+I	Alt+Enter

OPERATION	MAC	WINDOWS
Select all items	Command+A	Ctrl+A
Select more than one item in a list (noncontiguous)	Command, then click each item	Control, then click each item
Type special characters	Option+ character key	Alt+ character key

Trackpad shortcuts

Note: Some of these shortcuts require a "Precision Trackpad", such as the trackpad on Surface devices and some other third party laptops.

OPERATION	MAC	WINDOWS
Scroll	Two finger vertical swipe	Two finger vertical swipe
Zoom	Two finger pinch in and out	Two finger pinch in and out
Swipe back and forward between views	Two finger sideways swipe	Two finger sideways swipe
Switch virtual workspaces	Four fingers sideways swipe	Four fingers sideways swipe
Display currently open apps	Four fingers upward swipe	Three fingers upward swipe
Switch between apps	N/A	Slow three finger sideways swipe
Go to desktop	Spread out four fingers	Three finger swipe downwards
Open Cortana / Action center	Two finger slide from right	Three finger tap
Open extra information	Three finger tap	N/A
Show launchpad / start an app	Pinch with four fingers	Tap with four fingers

Note: Trackpad options are configurable on both platforms.

Command-line shells and terminals

Windows supports several command-line shells and terminals which sometimes work a little differently to the Mac's BASH shell and terminal emulator apps like Terminal and iTerm.

Windows shells

Windows has two primary command-line shells:

1. **PowerShell** - PowerShell is a cross-platform task automation and configuration management framework, consisting of a command-line shell and scripting language built on .NET. Using PowerShell, administrators, developers, and power-users can rapidly control and automate tasks that manage complex processes and various aspects of the environment and operating system upon which it is run. PowerShell is fully open-source, and because it is cross-platform, also available for Mac and Linux.

   Mac and Linux BASH shell users: PowerShell also supports many command-aliases that you are already familiar with. For example:

- List the contents of the current directory, using: `ls`
- Move files with: `mv`
- Move to a new directory with: `cd <path>`

Some commands and arguments are different in PowerShell vs. BASH. Learn more by entering: `get-help` in PowerShell or checkout the compatibility aliases in the docs.

To run PowerShell as an Administrator, enter "PowerShell" in your Windows start menu, then select "Run as Administrator."

2. **Windows Command Line (Cmd)**: Windows still ships the traditional Command Prompt (and Console – see below), providing compatibility with current and legacy MS-DOS-compatible commands and batch files. Cmd is useful when running existing/older batch files or command-line operations, but in general, users are recommended to learn and use PowerShell since Cmd is now in maintenance, and will not be receiving any improvements or new features in the future.

Linux shells

Windows Subsystem for Linux (WSL) can now be installed to support running a Linux shell within Windows. This means that you can run **bash**, with whichever specific Linux distribution you choose, integrated right inside Windows. Using WSL will provide the kind of environment most familiar to Mac users. For example, you will **ls** to list the files in a current directory, not **dir** as you would with the traditional Windows Cmd Shell. To learn about installing and using WSL, see the Windows Subsystem for Linux Installation Guide for Windows 10. Linux distributions that can be installed on Windows with WSL include:

1. Ubuntu 20.04 LTS
2. Kali Linux
3. Debian GNU/Linux
4. openSUSE Leap 15.1
5. SUSE Linux Enterprise Server 15 SP1

Just to name a few. Find more in the WSL install docs and install them directly from the Microsoft Store.

Windows Terminals

In addition to many 3rd party offerings, Microsoft provides two "terminals" – GUI applications that provide access to command-line shells and applications.

1. **Windows Terminal**: Windows Terminal is a new, modern, highly configurable command-line terminal application that provides very high performance, low-latency command-line user experience, multiple tabs, split window panes, custom themes and styles, multiple "profiles" for different shells or command-line apps, and considerable opportunities for you to configure and personalize many aspects of your command-line user experience.

   You can use Windows Terminal to open tabs connected to PowerShell, WSL shells (like Ubuntu or Debian), the traditional Windows Command Prompt, or any other command-line app (e.g. SSH, Azure CLI, Git Bash).

2. **Console**: On Mac and Linux, users usually start their preferred terminal application which then creates and connects to the user's default shell (e.g. BASH).

   However, due to a quirk of history, Windows users traditionally start their shell, and Windows automatically starts and connects a GUI Console app.

   While one can still launch shells directly and use the legacy Windows Console, it's highly recommended that users instead install and use Windows Terminal to experience the best, fastest, most productive command-line experience.

Apps and utilities

APP	MAC	WINDOWS
Settings and Preferences	System Preferences	Settings
Task manager	Activity Monitor	Task Manager
Disk formatting	Disk Utility	Disk Management
Text editing	TextEdit	Notepad
Event viewing	Console	Event Viewer
Find files/apps	Command+Space	Windows key

Guide for changing your dev environment from Mac to Windows

5/10/2021 • 4 minutes to read • Edit Online

The following tips and control equivalents should help you in your transition between a Mac and Windows (or WSL/Linux) development environment.

For app development, the nearest equivalent to Xcode would be Visual Studio. There is also a version of Visual Studio for Mac, if you ever feel the need to go back. For cross-platform source code editing (and a huge number of plug-ins) Visual Studio Code is the most popular choice.

Keyboard shortcuts

OPERATION	MAC	WINDOWS
Copy	Command+C	Ctrl+C
Cut	Command+X	Ctrl+X
Paste	Command+V	Ctrl+V
Undo	Command+Z	Ctrl+Z
Save	Command+S	Ctrl+S
Open	Command+O	Ctrl+O
Lock computer	Command+Control+Q	WindowsKey+L
Show desktop	Command+F3	WindowsKey+D
Open file browser	Command+N	WindowsKey+E
Minimize windows	Command+M	WindowsKey+M
Search	Command+Space	WindowsKey
Close active window	Command+W	Control+W
Switch current task	Command+Tab	Alt+Tab
Maximize a window to full screen	Control+Command+F	WindowsKey+Up
Save screen (Screenshot)	Command+Shift+3	WindowsKey+Shift+S
Save window	Command+Shift+4	WindowsKey+Shift+S
View item information or properties	Command+I	Alt+Enter

OPERATION	MAC	WINDOWS
Select all items	Command+A	Ctrl+A
Select more than one item in a list (noncontiguous)	Command, then click each item	Control, then click each item
Type special characters	Option+ character key	Alt+ character key

Trackpad shortcuts

Note: Some of these shortcuts require a "Precision Trackpad", such as the trackpad on Surface devices and some other third party laptops.

OPERATION	MAC	WINDOWS
Scroll	Two finger vertical swipe	Two finger vertical swipe
Zoom	Two finger pinch in and out	Two finger pinch in and out
Swipe back and forward between views	Two finger sideways swipe	Two finger sideways swipe
Switch virtual workspaces	Four fingers sideways swipe	Four fingers sideways swipe
Display currently open apps	Four fingers upward swipe	Three fingers upward swipe
Switch between apps	N/A	Slow three finger sideways swipe
Go to desktop	Spread out four fingers	Three finger swipe downwards
Open Cortana / Action center	Two finger slide from right	Three finger tap
Open extra information	Three finger tap	N/A
Show launchpad / start an app	Pinch with four fingers	Tap with four fingers

Note: Trackpad options are configurable on both platforms.

Command-line shells and terminals

Windows supports several command-line shells and terminals which sometimes work a little differently to the Mac's BASH shell and terminal emulator apps like Terminal and iTerm.

Windows shells

Windows has two primary command-line shells:

1. PowerShell - PowerShell is a cross-platform task automation and configuration management framework, consisting of a command-line shell and scripting language built on .NET. Using PowerShell, administrators, developers, and power-users can rapidly control and automate tasks that manage complex processes and various aspects of the environment and operating system upon which it is run. PowerShell is fully open-source, and because it is cross-platform, also available for Mac and Linux.

   Mac and Linux BASH shell users: PowerShell also supports many command-aliases that you are already familiar with. For example:

- List the contents of the current directory, using: `ls`
- Move files with: `mv`
- Move to a new directory with: `cd <path>`

Some commands and arguments are different in PowerShell vs. BASH. Learn more by entering: `get-help` in PowerShell or checkout the compatibility aliases in the docs.

To run PowerShell as an Administrator, enter "PowerShell" in your Windows start menu, then select "Run as Administrator."

2. **Windows Command Line (Cmd)**: Windows still ships the traditional Command Prompt (and Console – see below), providing compatibility with current and legacy MS-DOS-compatible commands and batch files. Cmd is useful when running existing/older batch files or command-line operations, but in general, users are recommended to learn and use PowerShell since Cmd is now in maintenance, and will not be receiving any improvements or new features in the future.

Linux shells

Windows Subsystem for Linux (WSL) can now be installed to support running a Linux shell within Windows. This means that you can run **bash**, with whichever specific Linux distribution you choose, integrated right inside Windows. Using WSL will provide the kind of environment most familiar to Mac users. For example, you will **ls** to list the files in a current directory, not **dir** as you would with the traditional Windows Cmd Shell. To learn about installing and using WSL, see the Windows Subsystem for Linux Installation Guide for Windows 10. Linux distributions that can be installed on Windows with WSL include:

1. Ubuntu 20.04 LTS
2. Kali Linux
3. Debian GNU/Linux
4. openSUSE Leap 15.1
5. SUSE Linux Enterprise Server 15 SP1

Just to name a few. Find more in the WSL install docs and install them directly from the Microsoft Store.

Windows Terminals

In addition to many 3rd party offerings, Microsoft provides two "terminals" – GUI applications that provide access to command-line shells and applications.

1. **Windows Terminal**: Windows Terminal is a new, modern, highly configurable command-line terminal application that provides very high performance, low-latency command-line user experience, multiple tabs, split window panes, custom themes and styles, multiple "profiles" for different shells or command-line apps, and considerable opportunities for you to configure and personalize many aspects of your command-line user experience.

   You can use Windows Terminal to open tabs connected to PowerShell, WSL shells (like Ubuntu or Debian), the traditional Windows Command Prompt, or any other command-line app (e.g. SSH, Azure CLI, Git Bash).

2. **Console**: On Mac and Linux, users usually start their preferred terminal application which then creates and connects to the user's default shell (e.g. BASH).

   However, due to a quirk of history, Windows users traditionally start their shell, and Windows automatically starts and connects a GUI Console app.

   While one can still launch shells directly and use the legacy Windows Console, it's highly recommended that users instead install and use Windows Terminal to experience the best, fastest, most productive command-line experience.

Apps and utilities

APP	MAC	WINDOWS
Settings and Preferences	System Preferences	Settings
Task manager	Activity Monitor	Task Manager
Disk formatting	Disk Utility	Disk Management
Text editing	TextEdit	Notepad
Event viewing	Console	Event Viewer
Find files/apps	Command+Space	Windows key

Install JavaScript frameworks on Windows

6/4/2021 • 2 minutes to read • Edit Online

This guide will help you get started using JavaScript frameworks on Windows, including Node.js, React.js, Vue.js, Next.js, Nuxt.js, or Gatsby.

Choose a JavaScript framework to install and set up your dev environment

Node.js overview

Learn about what you can do with Node.js and how to set up a Node.js development environment.

- Install on Windows
- Install on WSL
- Try a beginner-level tutorial

React overview

Learn about what you can do with React and how to set up a React development environment.

- Install on Windows for building web apps
- Install on WSL for building web apps
- Install on Windows for building desktop apps
- Install on Windows for building Android mobile apps
- Try a beginner-level tutorial

Vue.js

Vue.js overview

Learn about what you can do with Vue.js and how to set up a Vue.js development environment.

- Install on Windows
- Install on WSL
- Try a beginner-level tutorial

NEXT.JS

Install Next.js on WSL

Next.js is a framework for creating server-rendered JavaScript apps based on React.js, Node.js, Webpack and Babel.js. Learn how to install it on the Windows Subsystem for Linux.

NUXT JS

Install Nuxt.js on WSL

Nuxt.js is a framework for creating server-rendered JavaScript apps based on Vue.js, Node.js, Webpack and Babel.js. Learn how to install it on the Windows Subsystem for Linux.

Gatsby

Install Gatsby on WSL

Gatsby is a static site generator framework based on React.js. Learn how to install it on the Windows Subsystem for Linux.

What is NodeJS?

5/13/2021 • 3 minutes to read • Edit Online

Node.js is an open-source, cross-platform, server-side JavaScript runtime environment built on Chrome's V8 JavaScript engine originally authored by Ryan Dahl and released in 2009.

Does Node.js work on Windows?

Yes. Windows 10 supports two different environments for developing apps with Node.js:

- Install a Node.js development environment on Windows
- Install a Node.js development environment on Windows Subsystem for Linux

For help determining which environment to use, check out Should I install on Windows or Windows Subsystem for Linux?

What can you do with NodeJS?

Node.js is primarily used for building fast and scalable web applications. It uses an event-driven, non-blocking I/O model, making it lightweight and efficient. It's a great framework for data-intensive real-time applications that run across distributed devices. Here are a few examples of what you might create with Node.js.

- **Single-page apps (SPAs)**: These are web apps that work inside a browser and don't need to reload a page every time you use it to get new data. Some example SPAs include social networking apps, email or map apps, online text or drawing tools, etc.
- **Real-time apps (RTAs)**: These are web apps that enable users to receive information as soon as it's published by an author, rather than requiring that the user (or software) check a source periodically for updates. Some example RTAs include instant messaging apps or chat rooms, online multiplayer games that can be played in the browser, online collaboration docs, community storage, video conference apps, etc.
- **Data streaming apps**: These are apps (or services) that send data/content as it arrives (or is created) while keeping the connection open to continue downloading further data, content, or components as needed. Some examples include video- and audio-streaming apps.
- **REST APIs**: These are interfaces that provide data for someone else's web app to interact with. For example, a Calendar API service could provide dates and times for a concert venue that could be used by someone else's local events website.
- **Server-side rendered apps (SSRs)**: These web apps can run on both the client (in your browser / the front-end) and the server (the back-end) allowing pages that are dynamic to display (generate HTML for) whatever content is known and quickly grab content that is not known as it's available. These are often referred to as "isomorphic" or "universal" applications. SSRs utilize SPA methods in that they don't need to reload every time you use it. SSRs, however, offer a few benefits that may or may not be important to you, like making content on your site appear in Google search results and providing a preview image when links to your app are shared on social media like Twitter or Facebook. The potential drawback being that they require a Node.js server constantly running. In terms of examples, a social networking app that supports events that users will want to appear in search results and social media may benefit from SSR, while an email app may be fine as an SPA. You can also run server-rendered no-SPA apps, which my be something like a WordPress blog. As you can see, things can get complicated, you just need to decide what's important.
- **Command line tools**: These allow you to automate repetitive tasks and then distribute your tool across the vast Node.js ecosystem. An example of a command line tool is cURL, which stand for client URL and is used to download content from an internet URL. cURL is often used to install things like Node.js or, in our case, a

Node.js version manager.

- **Hardware programming**: While not quite as popular as web apps, Node.js is growing in popularity for IoT uses, such as collecting data from sensors, beacons, transmitters, motors, or anything that generates large amounts of data. Node.js can enable data collection, analyzing that data, communicating back and forth between a device and server, and taking action based on the analysis. NPM contains more than 80 packages for Arduino controllers, raspberry pi, Intel IoT Edison, various sensors, and Bluetooth devices.

Next steps

- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install NodeJS on Windows
- Install NodeJS on WSL
- Microsoft Learn: Build JavaScript applications with Node.js

Install Node.js on Windows Subsystem for Linux (WSL2)

6/30/2021 • 9 minutes to read • Edit Online

If you are using Node.js professionally, find performance speed and system call compatibility important, want to run Docker containers that leverage Linux workspaces and avoid having to maintain both Linux and Windows build scripts, or just prefer using a Bash command line, then you want to install Node.js on the Windows Subsystem for Linux (more specifically, WSL 2).

Using Windows Subsystem for Linux (WSL), enables you to install your preferred Linux distribution (Ubuntu is our default) so that you can have consistency between your development environment (where you write code) and production environment (the server where your code is deployed).

NOTE

If you are new to developing with Node.js and want to get up and running quickly so that you can learn, install Node.js on Windows. This recommendation also applies if you plan to use a Windows Server production environment.

Install WSL 2

WSL 2 is the most recent version available on Windows 10 and we recommend it for professional Node.js development workflows. To enable and install WSL 2, follow the steps in the WSL install documentation. These steps will include choosing a Linux distribution (for example, Ubuntu).

Once you have installed WSL 2 and a Linux distribution, open the Linux distribution (it can be found in your Windows start menu) and check the version and codename using the command: `lsb_release -dc`.

We recommend updating your Linux distribution regularly, including immediately after you install, to ensure you have the most recent packages. Windows doesn't automatically handle this update. To update your distribution, use the command: `sudo apt update && sudo apt upgrade`.

Install Windows Terminal (optional)

Windows Terminal is an improved command line shell that allows you to run multiple tabs so that you can quickly switch between Linux command lines, Windows Command Prompt, PowerShell, Azure CLI, or whatever you prefer to use. You can also create custom key bindings (shortcut keys for opening or closing tabs, copy+paste, etc.), use the search feature, customize your terminal with themes (color schemes, font styles and sizes, background image/blur/transparency), and more. Learn more in the Windows Terminal docs.

Install Windows Terminal using the Microsoft Store: By installing via the store, updates are handled automatically.

Install nvm, node.js, and npm

Besides choosing whether to install on Windows or WSL, there are additional choices to make when installing Node.js. We recommend using a version manager as versions change very quickly. You will likely need to switch between multiple versions of Node.js based on the needs of different projects you're working on. Node Version Manager, more commonly called nvm, is the most popular way to install multiple versions of Node.js. We will walk through the steps to install nvm and then use it to install Node.js and Node Package Manager (npm). There are alternative version managers to consider as well covered in the next section.

IMPORTANT

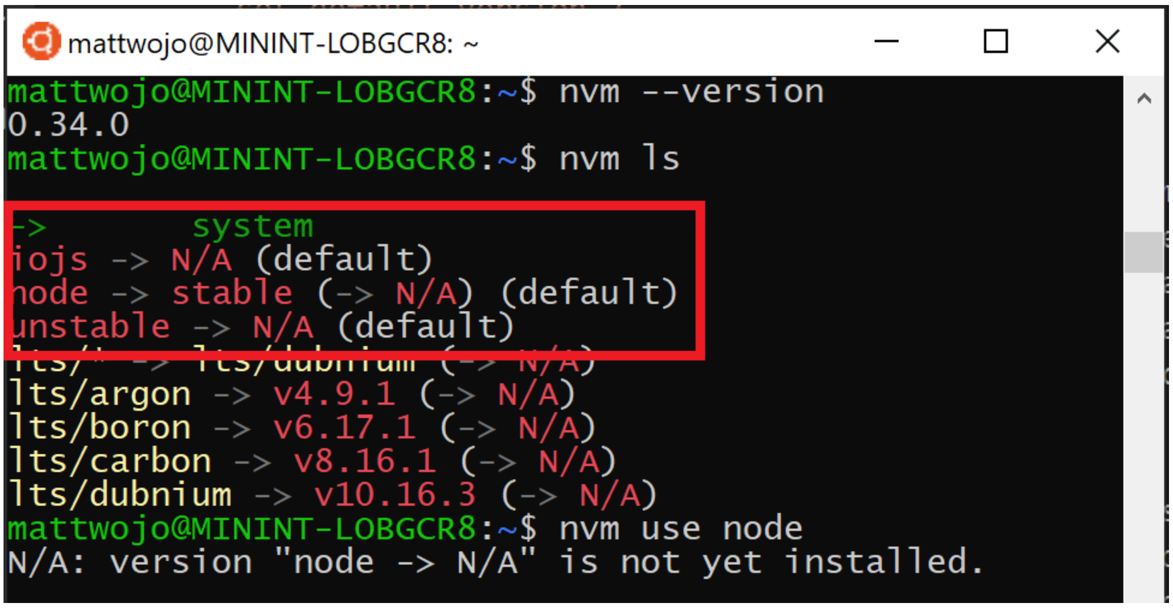
It is always recommended to remove any existing installations of Node.js or npm from your operating system before installing a version manager as the different types of installation can lead to strange and confusing conflicts. For example, the version of Node that can be installed with Ubuntu's `apt-get` command is currently outdated. For help with removing previous installations, see How to remove nodejs from ubuntu.)

1. Open your Ubuntu 18.04 command line.
2. Install cURL (a tool used for downloading content from the internet in the command-line) with: `sudo apt-get install curl`
3. Install nvm, with: `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash`

   NOTE

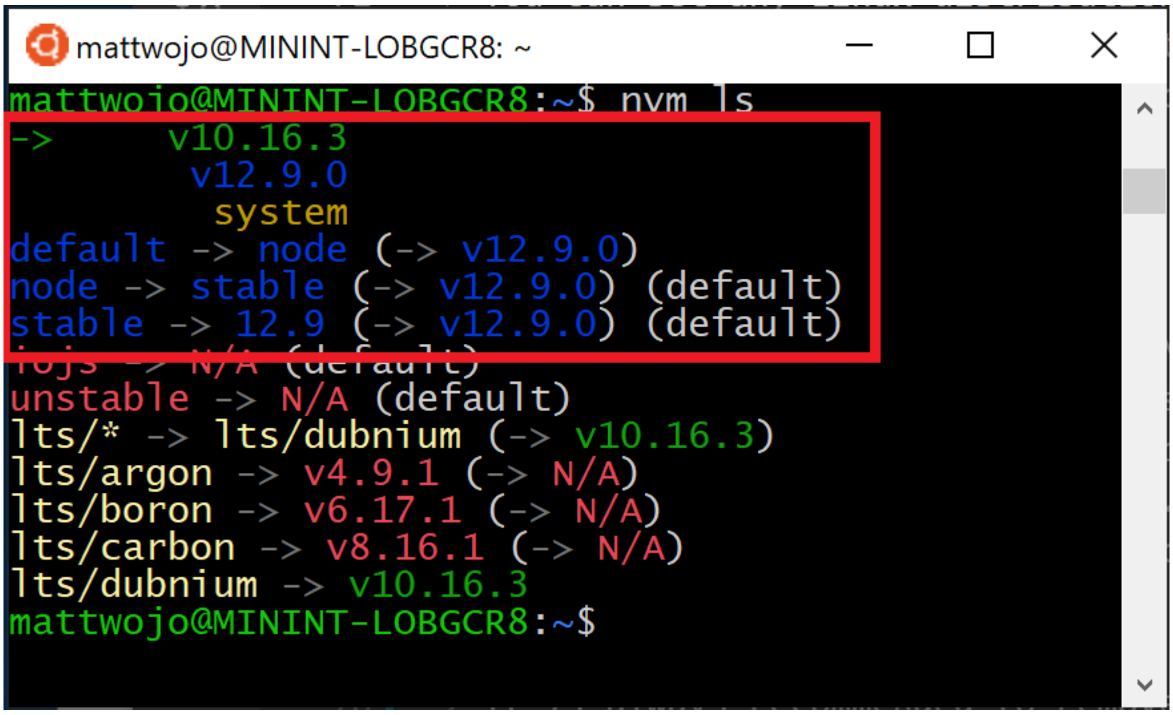
   At the time of publication, NVM v0.35.3 was the most recent version available. You can check the GitHub project page for the latest release of NVM, and adjust the above command to include the newest version. Installing the newer version of NVM using cURL will replace the older one, leaving the version of Node you've used NVM to install intact. For example: `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.36.0/install.sh | bash`

4. To verify installation, enter: `command -v nvm` ...this should return 'nvm', if you receive 'command not found' or no response at all, close your current terminal, reopen it, and try again. Learn more in the nvm github repo.
5. List which versions of Node are currently installed (should be none at this point): `nvm ls`



```
mattwojo@MININT-LOBGCR8:~$ nvm --version
0.34.0
mattwojo@MININT-LOBGCR8:~$ nvm ls

->       system
iojs -> N/A (default)
node -> stable (-> N/A) (default)
unstable -> N/A (default)
lts/* -> lts/dubnium (-> N/A)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.16.1 (-> N/A)
lts/dubnium -> v10.16.3 (-> N/A)
mattwojo@MININT-LOBGCR8:~$ nvm use node
N/A: version "node -> N/A" is not yet installed.
```

6. Install the current release of Node.js (for testing the newest feature improvements, but more likely to have issues): `nvm install node`
7. Install the latest stable LTS release of Node.js (recommended): `nvm install --lts`
8. List what versions of Node are installed: `nvm ls` ...now you should see the two versions that you just installed listed.



```
mattwojo@MININT-LOBGCR8: ~
mattwojo@MININT-LOBGCR8:~$ nvm ls
->      v10.16.3
         v12.9.0
          system
default -> node (-> v12.9.0)
node -> stable (-> v12.9.0) (default)
stable -> 12.9 (-> v12.9.0) (default)
iojs -> N/A (default)
unstable -> N/A (default)
lts/* -> lts/dubnium (-> v10.16.3)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.16.1 (-> N/A)
lts/dubnium -> v10.16.3
mattwojo@MININT-LOBGCR8:~$
```

9. Verify that Node.js is installed and the currently default version with: `node --version`. Then verify that you have npm as well, with: `npm --version` (You can also use `which node` or `which npm` to see the path used for the default versions).

10. To change the version of Node.js you would like to use for a project, create a new project directory `mkdir NodeTest`, and enter the directory `cd NodeTest`, then enter `nvm use node` to switch to the Current version, or `nvm use --lts` to switch to the LTS version. You can also use the specific number for any additional versions you've installed, like `nvm use v8.2.1`. (To list all of the versions of Node.js available, use the command: `nvm ls-remote`).

If you are using NVM to install Node.js and NPM, you should not need to use the SUDO command to install new packages.

Alternative version managers

While nvm is currently the most popular version manager for node, there are a few alternatives to consider:

- n is a long-standing `nvm` alternative that accomplishes the same thing with slightly different commands and is installed via `npm` rather than a bash script.
- fnm is a newer version manager, claiming to be much faster than `nvm`. (It also uses Azure Pipelines.)
- Volta is a new version manager from the LinkedIn team that claims improved speed and cross-platform support.
- asdf-vm is a single CLI for multiple languages, like ike gvm, nvm, rbenv & pyenv (and more) all in one.
- nvs (Node Version Switcher) is a cross-platform `nvm` alternative with the ability to integrate with VS Code.

Install Visual Studio Code

We recommend using Visual Studio Code with the Remote-development extension pack for Node.js projects. This splits VS Code into a “client-server” architecture, with the client (the VS Code user interface) running on your Windows operating system and the server (your code, Git, plugins, etc) running "remotely" on your WSL Linux distribution.

NOTE

This “remote” scenario is a bit different than you may be accustomed to. WSL supports an actual Linux distribution where your project code is running, separately from your Windows operating system, but still on your local machine. The Remote-WSL extension connects with your Linux subsystem as if it were a remote server, though it’s not running in the cloud… it’s still running on your local machine in the WSL environment that you enabled to run alongside Windows.

- Linux-based Intellisense and linting is supported.
- Your project will automatically build in Linux.
- You can use all your extensions running on Linux (ES Lint, NPM Intellisense, ES6 snippets, etc.).

Other code editors, like IntelliJ, Sublime Text, Brackets, etc. will also work with a WSL 2 Node.js development environment, but may not have the same sort of remote features that VS Code offers. These code editors may run into trouble accessing the WSL shared network location (\wsl$\Ubuntu\home) and will try to build your Linux files using Windows tools, which likely not what you want. The Remote-WSL Extension in VS Code handles this compatibility for you, with other IDEs you may need to set up an X server. Support for running GUI apps in WSL (like a code editor IDE) is coming soon.

Terminal-based text editors (vim, emacs, nano) are also helpful for making quick changes from right inside your console. The article, Emacs, Nano, or Vim: Choose your Terminal-Based Text Editor Wisely does a nice job explaining some differences and a bit about how to use each.

To install VS Code and the Remote-WSL Extension:

1. Download and install VS Code for Windows. VS Code is also available for Linux, but Windows Subsystem for Linux does not support GUI apps, so we need to install it on Windows. Not to worry, you'll still be able to integrate with your Linux command line and tools using the Remote - WSL Extension.

2. Install the Remote - WSL Extension on VS Code. This allows you to use WSL as your integrated development environment and will handle compatibility and pathing for you. Learn more.

IMPORTANT

If you already have VS Code installed, you need to ensure that you have the 1.35 May release or later in order to install the Remote - WSL Extension. 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-server/extensions.

Helpful VS Code Extensions

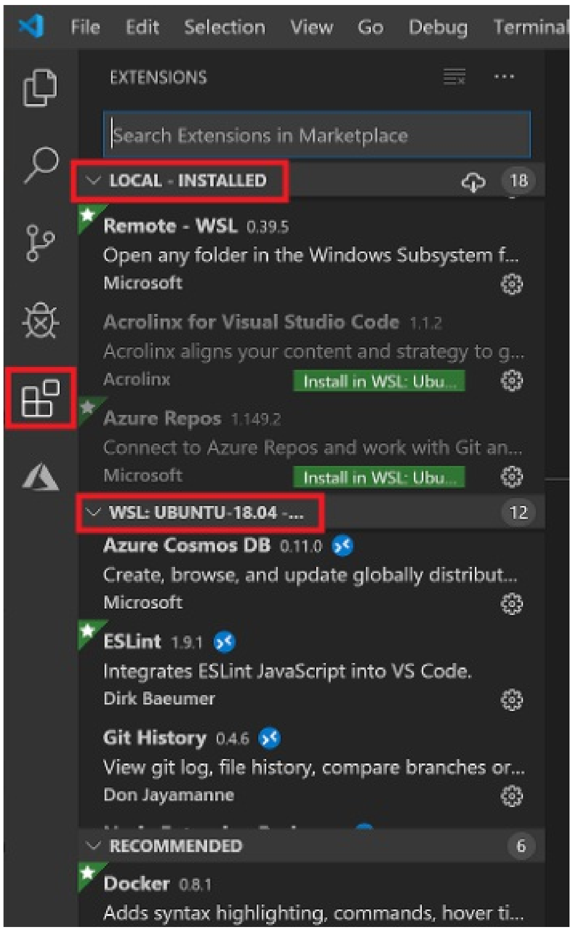
While VS Code comes with many features for Node.js development out of the box, there are some helpful extensions to consider installing available in the Node.js Extension Pack. Install them all or pick and choose which seem the most useful to you.

To install the Node.js extension pack:

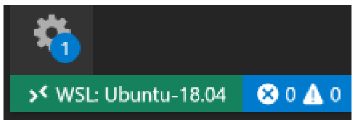
1. Open the **Extensions** window (Ctrl+Shift+X) in VS Code.

   The Extensions window is now divided into three sections (because you installed the Remote-WSL extension).

   - "Local - Installed": The extensions installed for use with your Windows operating system.
   - "WSL:Ubuntu-18.04-Installed": The extensions installed for use with your Ubuntu operating system (WSL).
   - "Recommended": Extensions recommended by VS Code based on the file types in your current project.


File
Edit
Selection
View
Go
Debug
Terminal
EXTENSIONS
Search Extensions in Marketplace
LOCAL - INSTALLED
18
Remote - WSL 0.39.5
Open any folder in the Windows Subsystem f...
Microsoft
Acrolinx for Visual Studio Code 1.1.2
Acrolinx aligns your content and strategy to g...
Acrolinx
Install in WSL: Ubu...
Azure Repos 1.149.2
Connect to Azure Repos and work with Git an...
Microsoft
Install in WSL: Ubu...
WSL: UBUNTU-18.04 -...
12
Azure Cosmos DB 0.11.0
Create, browse, and update globally distribut...
Microsoft
ESLint 1.9.1
Integrates ESLint JavaScript into VS Code.
Dirk Baeumer
Git History 0.4.6
View git log, file history, compare branches or...
Don Jayamanne
RECOMMENDED
6
Docker 0.8.1
Adds syntax highlighting, commands, hover ti...

2. In the search box at the top of the Extensions window, enter: **Node Extension Pack** (or the name of whatever extension you are looking for). The extension will be installed for either your Local or WSL instances of VS Code depending on where you have the current project opened. You can tell by selecting the remote link in the bottom-left corner of your VS Code window (in green). It will either give you the option to open or close a remote connection. Install your Node.js extensions in the "WSL:Ubuntu-18.04" environment.



A few additional extensions you may want to consider include:

- Debugger for Chrome: Once you finish developing on the server side with Node.js, 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.
- 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.

Set up Git (optional)

To set up Git for a Node.js project on WSL, see the article Get started using Git on Windows Subsystem for Linux in the WSL documentation.

Install NodeJS on Windows

4/21/2021 • 6 minutes to read • Edit Online

If you are new to developing with Node.js and want to get up and running quickly so that you can learn, follow the steps below to install Node.js directly on Windows.

NOTE

If you are using Node.js professionally, find performance speed and system call compatibility important, want to run Docker containers that leverage Linux workspaces and avoid having to maintain both Linux and Windows build scripts, or just prefer using a Bash command line, then install Node.js on Windows Subsystem for Linux (more specifically, WSL 2).

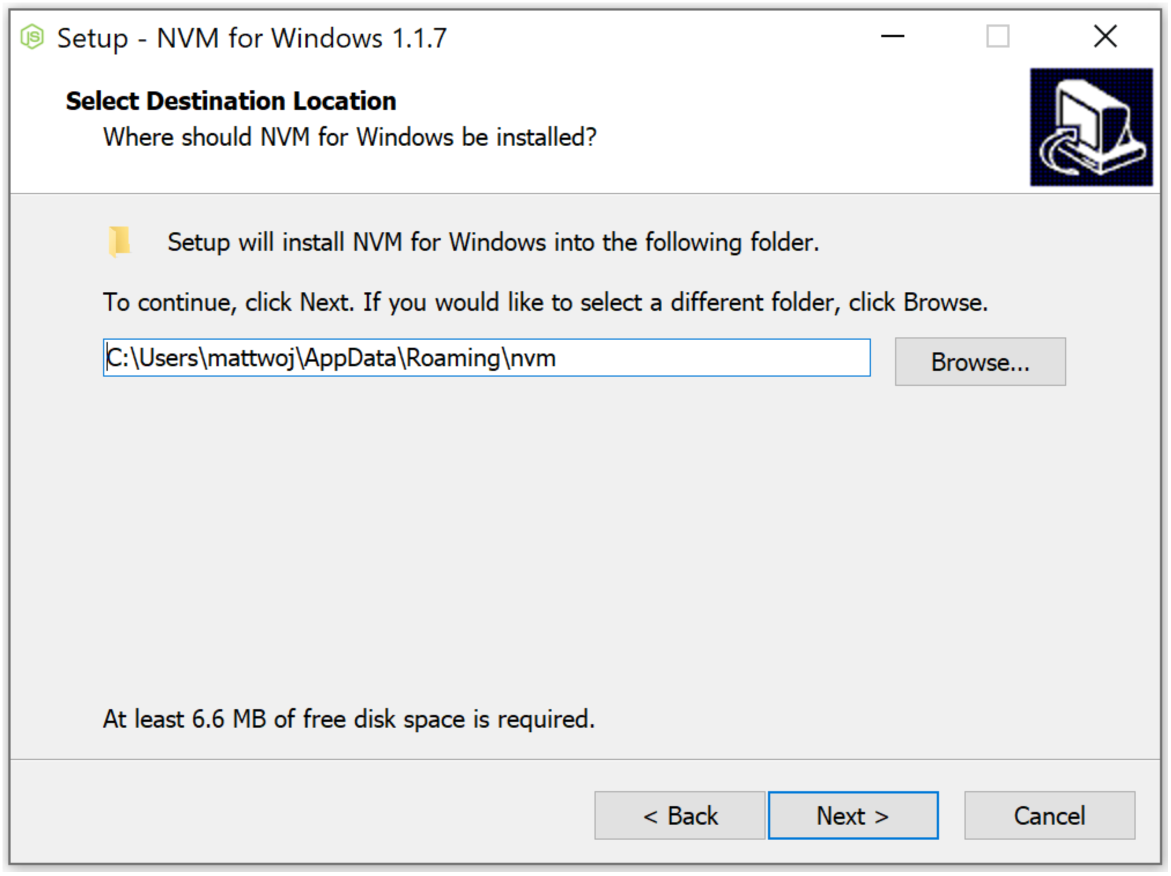
Install nvm-windows, node.js, and npm

Besides choosing whether to install on Windows or WSL, there are additional choices to make when installing Node.js. We recommend using a version manager as versions change very quickly. You will likely need to switch between multiple Node.js versions based on the needs of different projects you're working on. Node Version Manager, more commonly called nvm, is the most popular way to install multiple versions of Node.js, but is only available for Mac/Linux and not supported on Windows. Instead, we will walk through the steps to install nvm-windows and then use it to install Node.js and Node Package Manager (npm). There are alternative version managers to consider as well covered in the next section.

IMPORTANT

It is always recommended to remove any existing installations of Node.js or npm from your operating system before installing a version manager as the different types of installation can lead to strange and confusing conflicts. This includes deleting any existing nodejs installation directories (e.g., "C:\Program Files\nodejs") that might remain. NVM's generated symlink will not overwrite an existing (even empty) installation directory. For help with removing previous installations, see How to completely remove node.js from Windows.)

1. Open the windows-nvm repository in your internet browser and select the **Download Now** link.
2. Download the **nvm-setup.zip** file for the most recent release.
3. Once downloaded, open the zip file, then open the **nvm-setup.exe** file.
4. The Setup-NVM-for-Windows installation wizard will walk you through the setup steps, including choosing the directory where both nvm-windows and Node.js will be installed.


Setup - NVM for Windows 1.1.7

Select Destination Location

Where should NVM for Windows be installed?

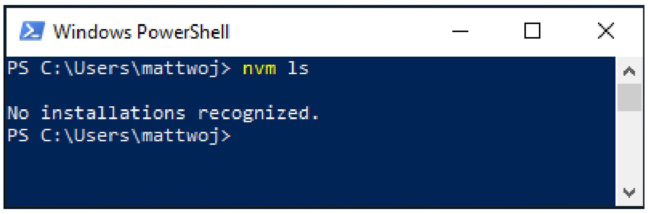
Setup will install NVM for Windows into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Users\mattwoj\AppData\Roaming\nvm

At least 6.6 MB of free disk space is required.

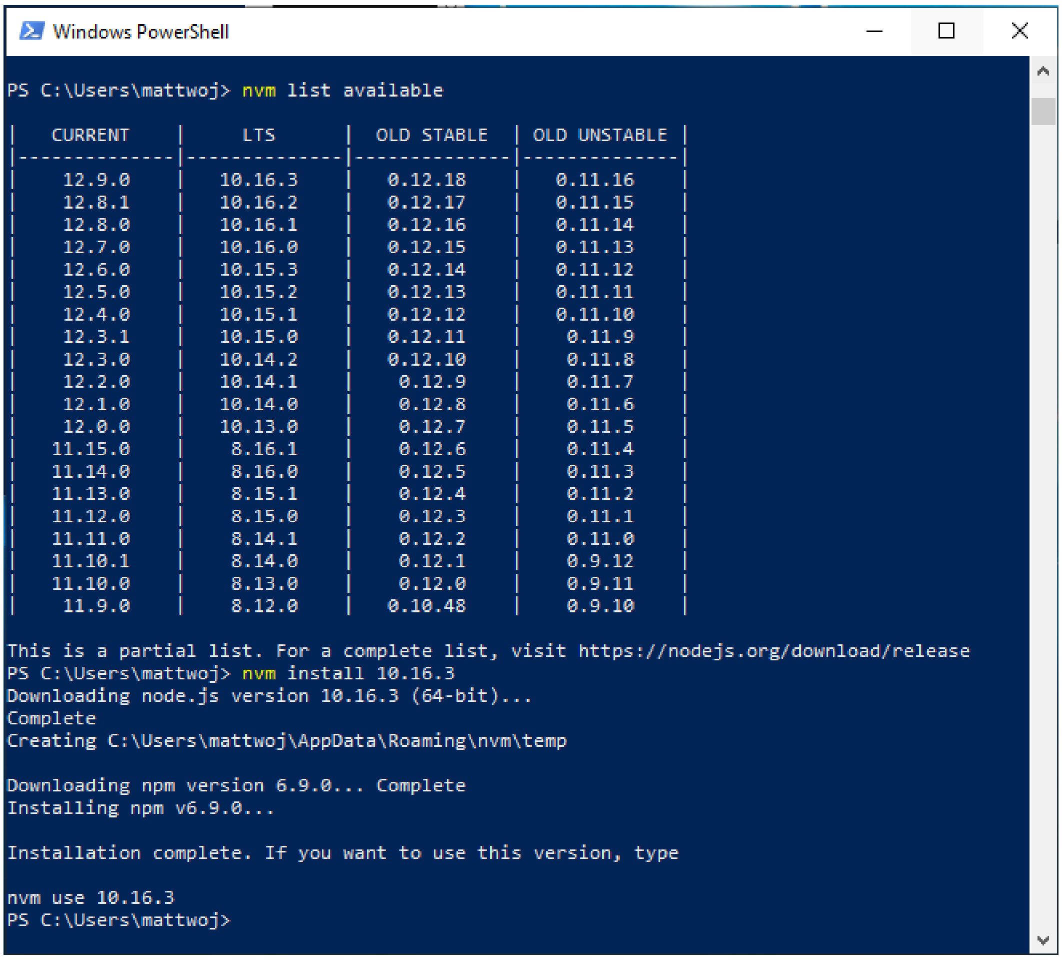
5. Once the installation is complete. Open PowerShell and try using windows-nvm to list which versions of Node are currently installed (should be none at this point): `nvm ls`


Windows PowerShell

```
PS C:\Users\mattwoj> nvm ls

No installations recognized.
PS C:\Users\mattwoj>
```

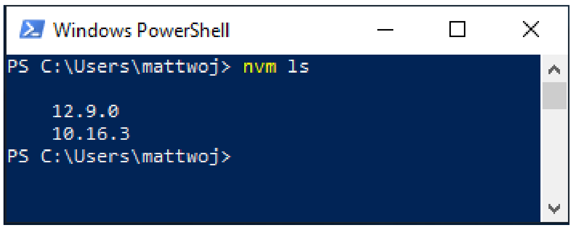
6. Install the current release of Node.js (for testing the newest feature improvements, but more likely to have issues than the LTS version): `nvm install latest`

7. Install the latest stable LTS release of Node.js (recommended) by first looking up what the current LTS version number is with: `nvm list available`, then installing the LTS version number with: `nvm install <version>` (replacing `<version>` with the number, ie: `nvm install 12.14.0`).



```
PS C:\Users\mattwoj> nvm list available

|   CURRENT    |     LTS      |  OLD STABLE  | OLD UNSTABLE |
|--------------|--------------|--------------|--------------|
|    12.9.0    |   10.16.3    |   0.12.18    |   0.11.16    |
|    12.8.1    |   10.16.2    |   0.12.17    |   0.11.15    |
|    12.8.0    |   10.16.1    |   0.12.16    |   0.11.14    |
|    12.7.0    |   10.16.0    |   0.12.15    |   0.11.13    |
|    12.6.0    |   10.15.3    |   0.12.14    |   0.11.12    |
|    12.5.0    |   10.15.2    |   0.12.13    |   0.11.11    |
|    12.4.0    |   10.15.1    |   0.12.12    |   0.11.10    |
|    12.3.1    |   10.15.0    |   0.12.11    |    0.11.9    |
|    12.3.0    |   10.14.2    |   0.12.10    |    0.11.8    |
|    12.2.0    |   10.14.1    |    0.12.9    |    0.11.7    |
|    12.1.0    |   10.14.0    |    0.12.8    |    0.11.6    |
|    12.0.0    |   10.13.0    |    0.12.7    |    0.11.5    |
|   11.15.0    |    8.16.1    |    0.12.6    |    0.11.4    |
|   11.14.0    |    8.16.0    |    0.12.5    |    0.11.3    |
|   11.13.0    |    8.15.1    |    0.12.4    |    0.11.2    |
|   11.12.0    |    8.15.0    |    0.12.3    |    0.11.1    |
|   11.11.0    |    8.14.1    |    0.12.2    |    0.11.0    |
|   11.10.1    |    8.14.0    |    0.12.1    |    0.9.12    |
|   11.10.0    |    8.13.0    |    0.12.0    |    0.9.11    |
|    11.9.0    |    8.12.0    |   0.10.48    |    0.9.10    |

This is a partial list. For a complete list, visit https://nodejs.org/download/release
PS C:\Users\mattwoj> nvm install 10.16.3
Downloading node.js version 10.16.3 (64-bit)...
Complete
Creating C:\Users\mattwoj\AppData\Roaming\nvm\temp

Downloading npm version 6.9.0... Complete
Installing npm v6.9.0...

Installation complete. If you want to use this version, type

nvm use 10.16.3
PS C:\Users\mattwoj>
```

8. List what versions of Node are installed: `nvm ls` ...now you should see the two versions that you just installed listed.



```
PS C:\Users\mattwoj> nvm ls

    12.9.0
    10.16.3
PS C:\Users\mattwoj>
```

9. After installing the Node.js version numbers you need, select the version that you would like to use by entering: `nvm use <version>` (replacing `<version>` with the number, ie: `nvm use 12.9.0`).

10. To change the version of Node.js you would like to use for a project, create a new project directory `mkdir NodeTest`, and enter the directory `cd NodeTest`, then enter `nvm use <version>` replacing `<version>` with the version number you'd like to use (ie v10.16.3`).

11. Verify which version of npm is installed with: `npm --version`, this version number will automatically change to whichever npm version is associated with your current version of Node.js.

Alternative version managers

While windows-nvm is currently the most popular version manager for node, there are alternatives to consider:

- nvs (Node Version Switcher) is a cross-platform `nvm` alternative with the ability to integrate with VS Code.

- Volta is a new version manager from the LinkedIn team that claims improved speed and cross-platform support.

To install Volta as your version manager (rather than windows-nvm), go to the **Windows Installation** section of their Getting Started guide, then download and run their Windows installer, following the setup instructions.

IMPORTANT

You must ensure that Developer Mode is enabled on your Windows machine before installing Volta.

To learn more about using Volta to install multiple versions of Node.js on Windows, see the Volta Docs.

Install Visual Studio Code

We recommend you install Visual Studio Code, as well as the Node.js Extension Pack, for developing with Node.js on Windows. Install them all or pick and choose which seem the most useful to you.

To install the Node.js extension pack:

1. Open the **Extensions** window (Ctrl+Shift+X) in VS Code.
2. In the search box at the top of the Extensions window, enter: "Node Extension Pack" (or the name of whatever extension you are looking for).
3. Select **Install**. Once installed, your extension will appear in the "Enabled" folder of your **Extensions** window. You can disable, uninstall, or configure settings by selecting the gear icon next to the description of your new extension.

A few additional extensions you may want to consider include:

- Debugger for Chrome: Once you finish developing on the server side with Node.js, 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.
- 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.

Alternative code editors

If you prefer to use a code editor or IDE other than Visual Studio Code, the following are also good options for your Node.js development environment:

- IntelliJ IDEA
- Sublime Text
- Atom
- Brackets
- Notepad++

Install Git

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. You first need to install Git to power the Source Control panel.

1. Download and install Git for Windows from the git-scm website.

2. An Install Wizard is included that will ask you a series of questions about settings for your Git installation. We recommend using all of the default settings, unless you have a specific reason for changing something.

3. If you've never worked with Git before, GitHub Guides can help you get started.

4. We recommend adding a .gitignore file to your Node projects. Here is GitHub's default gitignore template for Node.js.

Use Windows Subsystem for Linux for production

Using Node.js directly on Windows is great for learning and experimenting with what you can do. Once you are ready to build production-ready web apps, which are typically deployed to a Linux-based server, we recommend using Windows Subsystem for Linux version 2 (WSL 2) for developing Node.js web apps. Many Node.js packages and frameworks are created with a *nix environment in mind and most Node.js apps are deployed on Linux, so developing on WSL ensures consistency between your development and production environments. To set up a WSL dev environment, see Set up your Node.js development environment with WSL 2.

NOTE

If you are in the (somewhat rare) situation of needing to host a Node.js app on a Windows server, the most common scenario seems to be using a reverse proxy. There are two ways to do this: 1) using iisnode or directly. We do not maintain these resources and recommend using Linux servers to host your Node.js apps.

Tutorial: Node.js for Beginners

4/21/2021 • 6 minutes to read • Edit Online

If you're brand new to using Node.js, this guide will help you to get started with some basics.

- Try using Node.js in Visual Studio Code
- Create your first Node.js web app using Express
- Try using a Node.js module

Prerequisites

- Installing on Node.js on Windows or on Windows Subsystem for Linux

If you are a beginner, trying Node.js for the first time, we recommend installing directly on Windows. For more information, see Should I install Node.js on Windows or Windows Subsystem for Linux

Try NodeJS with Visual Studio Code

If you have not yet installed Visual Studio Code, return to the prerequisite section above and follow the installation steps linked for Windows or WSL.

1. Open your command line and create a new directory: `mkdir HelloNode`, then enter the directory: `cd HelloNode`
2. Create a JavaScript file named "app.js" with a variable named "msg" inside: `echo var msg > app.js`
3. Open the directory and your app.js file in VS Code using the command: `code .`
4. Add a simple string variable ("Hello World"), then send the contents of the string to your console by entering this in your "app.js" file:

   ```
   var msg = 'Hello World';
   console.log(msg);
   ```

5. To run your "app.js" file with Node.js. Open your terminal right inside VS Code by selecting **View** > **Terminal** (or select Ctrl+`, using the backtick character). If you need to change the default terminal, select the dropdown menu and choose **Select Default Shell**.
6. In the terminal, enter: `node app.js`. You should see the output: "Hello World".

NOTE
Notice that when you type `console` in your 'app.js' file, VS Code displays supported options related to the `console` object for you to choose from using IntelliSense. Try experimenting with Intellisense using other JavaScript objects.

Create your first NodeJS web app using Express

Express is a minimal, flexible, and streamlined Node.js framework that makes it easier to develop a web app that can handle multiple types of requests, like GET, PUT, POST, and DELETE. Express comes with an application generator that will automatically create a file architecture for your app.

To create a project with Express.js:

1. Open your command line (Command Prompt, Powershell, or whatever you prefer).
2. Create a new project folder: `mkdir ExpressProjects` and enter that directory: `cd ExpressProjects`
3. Use Express to create a HelloWorld project template: `npx express-generator HelloWorld --view=pug`

   > **NOTE**
   >
   > We are using the `npx` command here to execute the Express.js Node package without actually installing it (or by temporarily installing it depending on how you want to think of it). If you try to use the `express` command or check the version of Express installed using: `express --version`, you will receive a response that Express cannot be found. If you want to globally install Express to use over and over again, use: `npm install -g express-generator`. You can view a list of the packages that have been installed by npm using `npm list`. They'll be listed by depth (the number of nested directories deep). Packages that you installed will be at depth 0. That package's dependencies will be at depth 1, further dependencies at depth 2, and so on. To learn more, see Difference between npx and npm? on StackOverflow.

4. Examine the files and folders that Express included by opening the project in VS Code, with: `code .`

   The files that Express generates will create a web app that uses an architecture that can appear a little overwhelming at first. You'll see in your VS Code **Explorer** window (Ctrl+Shift+E to view) that the following files and folders have been generated:

   - `bin`. Contains the executable file that starts your app. It fires up a server (on port 3000 if no alternative is supplied) and sets up basic error handling.
   - `public`. Contains all the publicly accessed files, including JavaScript files, CSS stylesheets, font files, images, and any other assets that people need when they connect to your website.
   - `routes`. Contains all the route handlers for the application. Two files, `index.js` and `users.js`, are automatically generated in this folder to serve as examples of how to separate out your application's route configuration.
   - `views`. Contains the files used by your template engine. Express is configured to look here for a matching view when the render method is called. The default template engine is Jade, but Jade has been deprecated in favor of Pug, so we used the `--view` flag to change the view (template) engine. You can see the `--view` flag options, and others, by using `express --help`.
   - `app.js`. The starting point of your app. It loads everything and begins serving user requests. It's basically the glue that holds all the parts together.
   - `package.json`. Contains the project description, scripts manager, and app manifest. Its main purpose is to track your app's dependencies and their respective versions.

5. You now need to install the dependencies that Express uses in order to build and run your HelloWorld Express app (the packages used for tasks like running the server, as defined in the `package.json` file). Inside VS Code, open your terminal by selecting **View** > **Terminal** (or select Ctrl+`, using the backtick character), be sure that you're still in the 'HelloWorld' project directory. Install the Express package dependencies with:

   ```
   npm install
   ```

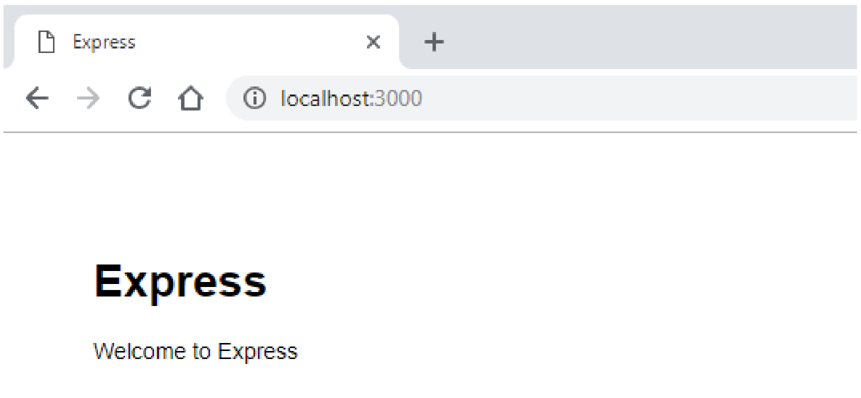
6. At this point you have the framework set up for a multiple-page web app that has access to a large variety of APIs and HTTP utility methods and middleware, making it easier to create a robust API. Start the Express app on a virtual server by entering:

```
npx cross-env DEBUG=HelloWorld:* npm start
```

TIP

The `DEBUG=myapp:*` part of the command above means you are telling Node.js that you want to turn on logging for debugging purposes. Remember to replace 'myapp' with your app name. You can find your app name in the `package.json` file under the "name" property. Using `npx cross-env` sets the `DEBUG` environment variable in any terminal, but you can also set it with your terminal specific way. The `npm start` command is telling npm to run the scripts in your `package.json` file.

7. You can now view the running app by opening a web browser and going to: **localhost:3000**



8. Now that your HelloWorld Express app is running locally in your browser, try making a change by opening the 'views' folder in your project directory and selecting the 'index.pug' file. Once open, change `h1= title` to `h1= "Hello World!"` and selecting **Save** (Ctrl+S). View your change by refreshing the **localhost:3000** URL on your web browser.

9. To stop running your Express app, in your terminal, enter: **Ctrl+C**

Try using a Node.js module

Node.js has tools to help you develop server-side web apps, some built in and many more available via npm. These modules can help with many tasks:

TOOL	USED FOR
gm, sharp	Image manipulation, including editing, resizing, compression, and so on, directly in your JavaScript code
PDFKit	PDF generation
validator.js	String validation
imagemin, UglifyJS2	Minification
spritesmith	Sprite sheet generation
winston	Logging

TOOL	USED FOR
commander.js	Creating command-line applications

Let's use the built-in OS module to get some information about your computer's operating system:

1. In your command line, open the Node.js CLI. You'll see the `>` prompt letting you know you're using Node.js after entering: `node`
2. To identify the operating system you are currently using (which should return a response letting you know that you're on Windows), enter: `os.platform()`
3. To check your CPU's architecture, enter: `os.arch()`
4. To view the the CPUs available on your system, enter: `os.cpus()`
5. Leave the Node.js CLI by entering `.exit` or by selecting Ctrl+C twice.

   TIP

   You can use the Node.js OS module to do things like check the platform and return a platform-specific variable: Win32/.bat for Windows development, darwin/.sh for Mac/unix, Linux, SunOS, and so on (for example, `var isWin = process.platform === "win32";`).

React overview

5/13/2021 • 7 minutes to read • Edit Online

What is React JS?

React is an open-source JavaScript library for building front end user interfaces. Unlike other JavaScript libraries that provide a full application framework, React is focused solely on creating application views through encapsulated units called **components** that maintain state and generate UI elements. You can place an individual component on a web page or nest hierarchies of components to create a complex UI.

React components are typically written in JavaScript and JSX (JavaScript XML) which is a JavaScript extension that looks likes a lot like HTML, but has some syntax features that make it easier to do common tasks like registering event handlers for UI elements. A React component implements the **render** method, which returns the JSX representing the component's UI. In a web app, the JSX code returned by the component is translated into browser-compliant HTML rendered in the browser.

Does React work on Windows?

Yes. Windows 10 supports two different environments for developing React apps:

- Install a React development environment on Windows
- Install a React development environment on Windows Subsystem for Linux

For help determining which environment to use, check out Should I install on Windows or Windows Subsystem for Linux?

What can you do with React?

Windows 10 supports a wide range of scenarios for React developers, including:

- **Basic web apps**: If you are new to React and primarily interested in learning about building a basic web app with React, we recommend that you install create-react-app directly on Windows. If you're planning to create a web app that will be deployed for production, you may want to consider installing create-react-app on Windows Subsystem for Linux (WSL), for better performance speed, system call compatibility, and alignment between your local development environment and deployment environment (which is often a Linux server).

- **Single-Page Apps (SPAs)**: These are websites that interact with the user by dynamically rewriting the current web page with new data from a server, rather than the browser default of loading entire new pages. If you want to build a static content-oriented SPA website, we recommend installing Gatsby on WSL. If you want to build a server-rendered SPA website with a Node.js backend, we recommend installing Next.js on WSL. (Though Next.js now also offers static file serving).

- **Native desktop apps**: React Native for Windows + macOS enables you to build native desktop applications with JavaScript that run across various types of desktop PCs, laptops, tablets, Xbox, and Mixed Reality devices. It supports both the Windows SDK and macOS SDK.

- **Native mobile apps**: React Native is a cross-platform way to create Android and iOS apps with JavaScript that render to native platform UI code. There are two main ways to install React Native -- the Expo CLI and the React Native CLI. There's a good comparison of the two on StackOverflow. Expo has a client app for iOS and Android mobile devices for running and testing your apps. For instructions on developing an Android app using Windows, React Native, and the Expo CLI see React Native for Android development on Windows.

Installation options

There are several different ways to install React along with an integrated toolchain that best works for your use-case scenario. Here are a few of the most popular.

- Install create-react-app directly on Windows
- Install create-react-app on Windows Subsystem for Linux (WSL)
- Install the Next.js framework on WSL
- Install the Gatsby framework on WSL
- Install React Native for Windows desktop development
- Install React Native for Android development on Windows
- Install React Native for mobile development across platforms)
- Install React in the browser with no toolchain: Since React is a JavaScript library that is, in its most basic form, just a collection of text files, you can create React apps without installing any tools or libraries on your computer. You may only want to add a few "sprinkles of interactivity" to a web page and not need build tooling. You can add a React component by just adding a plain `<script>` tag on an HTML page. Follow the "Add React in One Minute" steps in the React docs.

React tools

While writing a simple React component in a plain text editor is a good introduction to React, code generated this way is bulky, difficult to maintain and deploy, and slow. There are some common tasks production apps will need to perform. These tasks are handled by other JavaScript frameworks that are taken by the app as a dependency. These tasks include:

- **Compilation** - JSX is the language commonly used for React apps, but browsers can't process this syntax directly. Instead, the code needs to be compiled into standard JavaScript syntax and customized for different browsers. Babel is an example of a compiler that supports JSX.
- **Bundling** - Since performance is key for modern web apps, it's important that an app's JavaScript includes only the needed code for the app and combined into as few files as possible. A bundler, such as webpack performs this task for you.
- **Package management** - Package managers make it easy to include the functionality of third-party packages in your app, including updating them and managing dependencies. Commonly used package managers include Yarn and npm.

Together, the suite of frameworks that help you create, build, and deploy your app are called a toolchain. An easy toolchain to get started with is create-react-app, which generates a simple one-page app for you. The only setup required to use **create-react-app** is Node.js.

- For Windows development, follow the instructions to install Node.js on WSL or install Node.js on Windows. For help deciding which to use, check out the article: Should I install on Windows or Windows Subsystem for Linux?.

React Native component directory

The components that can be used in a React Native app include the following:

- Core components - Components that are developed and supported as part of the React Native framework.
- Community components - Components that are developed and maintained by the community.
- Native components - Components that you author yourself, using platform-native code, and register to be accessible from React Native.

The React Native Directory provides a list of component libraries that can be filtered by target platform.

Fullstack React toolchain options

React is a library, not a framework, so may require additional tools to create a more complex app. In addition to using React, you may want to consider using:

- Package manager: Two popular package managers for React are npm (which is included with NodeJS) and yarn. Both support a broad library of well-maintained packages that can be installed.
- React Router: a collection of navigational components that can help you with things like bookmarkable URLs for your web app or a composable way to navigate in React Native. React is really only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of a routing library like React Router.
- Redux: A predictable state container that helps with data-flow architecture. It is likely not something you need until you get into more advanced React development. To quote Dan Abramov, one of the creators of Redux: "Don't use Redux until you have problems with vanilla React."
- Webpack: A build tool that lets you compile JavaScript modules, also known as module bundler. When webpack processes your application, it internally builds a dependency graph which maps every module your project needs and generates one or more bundles.
- Uglify: A JavaScript parser, minifier, compressor and beautifier toolkit.
- Babel: A JavaScript compiler mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments. It can also be helpful to use babel-preset-env so that you don't need to micromanage syntax transforms or browser polyfills and can define what internet browsers to support.
- ESLint: A tool for identifying and reporting on patterns found in your JavaScript code that helps you make your code more consistent and avoid bugs.
- Enzyme: A JavaScript testing utility for React that makes it easier to test your React Components' output.
- Jest: A testing framework built into the create-react-app package to help with writing idiomatic JavaScript tests.
- Mocha: A testing framework that runs on Node.js and in the browser to help with asynchronous testing, reporting, and mapping uncaught exceptions to the correct test cases.

React courses and tutorials

Here are a few recommended places to learn React and build sample apps:

- Microsoft Learn: The React Learning Path contains online course modules to help you get started with the basics.
- Build a single-page app (SPA) that runs in the browser (like this sample web app that retrieves calendar info for a user with the Microsoft Graph API)
- Build a server-rendered app with Next.js or a static-site-generated app with Gatsby
- Create a user interface (UI) for a native app that runs on Windows, Android, and iOS devices (checkout these native Windows app samples or this sample native app that retrieves calendar info for a user with the Microsoft Graph API)
- Develop an app for Surface Duo dual-screen device
- Create a web app or native app using Fluent UI React components
- Build a React app with TypeScript

Additional resources

- The official React docs offer all of the latest, up-to-date information on React
- Microsoft Edge Add-ons for React Developer Tools: Add two tabs to your Microsoft Edge dev tools to help with your React development: Components and Profiler.

Install React on Windows Subsystem for Linux

5/13/2021 • 3 minutes to read • Edit Online

This guide will walk through installing React on a Linux distribution (ie. Ubuntu) running on the Windows Subsystem for Linux (WSL) using the create-react-app toolchain.

We recommend following these instructions if you are creating a single-page app (SPA) that you would like to use Bash commands or tools with and/or plan to deploy to a Linux server or use Docker containers. If you are brand new to React and just interested in learning, you may want to consider installing with create-react-app directly on Window.

For more general information about React, deciding between React (web apps), React Native (mobile apps), and React Native for Windows (desktop apps), see the React overview.

Prerequisites

- Install the latest version of Windows 10 (Version 1903+, Build 18362+)
- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: These instructions use Node Version Manager (nvm) for installation, you will need a recent version of NodeJS to run create-react-app, as well as a recent version of Node Package Manager (npm). For exact version requirements, see the Create React App website.

IMPORTANT

Installing a Linux distribution with WSL will create a directory for storing files: `\\wsl\Ubuntu-20.04` (substitute Ubuntu-20.04 with whatever Linux distribution you're using). To open this directory in Windows File Explorer, open your WSL command line, select your home directory using `cd ~`, then enter the command `explorer.exe .` Be careful not to install NodeJS or store files that you will be working with on the mounted C drive (`/mnt/c/Users/yourname$`). Doing so will significantly slow down your install and build times.

Install React

To install the full React toolchain on WSL, we recommend using create-react-app:

1. Open a WSL command line (ie. Ubuntu).
2. Create a new project folder: `mkdir ReactProjects` and enter that directory: `cd ReactProjects`.
3. Install React using npx:

```
npx create-react-app my-app
```

NOTE

npx is the package runner used by npm to execute packages in place of a global install. It basically creates a temporary install of React so that with each new project you are using the most recent version of React (not whatever version was current when you performed the global install above). Using the NPX package runner to execute a package can also help reduce the pollution of installing lots of packages on your machine.

4. This will first ask for your permission to temporarily install create-react-app and it's associated packages. Once completed, change directories into your new app ("my-app" or whatever you've chosen to call it): `cd my-app`.

5. Start your new React app:

   ```
   npm start
   ```

   This command will start up the Node.js server and launch a new browser window displaying your app. You can use **Ctrl + c** to stop running the React app in your command line.

   > **NOTE**
   >
   > Create React App includes a frontend build pipeline using Babel and webpack, but doesn't handle backend logic or databases. If you are seeking to build a server-rendered website with React that uses a Node.js backend, we recommend installing Next.js, rather than this create-react-app installation, which is intended more for single-page apps. You also may want to consider installing Gatsby if you want to build a static content-oriented website.

6. When you're ready to deploy your web app to production, running `npm run build` will create a build of your app in the "build" folder. You can learn more in the Create React App User Guide.

Add React to an existing web app

Since React is a JavaScript library that is, in its most basic form, just a collection of text files, you can create React apps without installing any tools or libraries on your computer. You may only want to add a few "sprinkles of interactivity" to a web page and not need build tooling. You can add a React component by just adding a plain `<script>` tag on an HTML page. Follow the "Add React in One Minute" steps in the React docs.

Additional resources

- React docs
- Create React App docs
- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install Next.js
- Install Gatsby
- Install React Native for Windows
- Install NodeJS on Windows
- Install NodeJS on WSL
- Try the tutorial: Using React in Visual Studio Code
- Try the Microsoft Learn online course: React

Install React directly on Windows

5/13/2021 • 2 minutes to read • Edit Online

This guide will walk through installing React directly on Windows using the create-react-app toolchain.

We recommend following these instructions if you are new to React and just interested in learning. If you are creating a single-page app (SPA) that you would like to use Bash commands or tools with and/or plan to deploy to a Linux server, we recommend that you install with create-react-app on Windows Subsystem for Linux (WSL).

For more general information about React, deciding between React (web apps), React Native (mobile apps), and React Native for Windows (desktop apps), see the React overview.

Prerequisites

- Install the latest version of Windows 10 (Version 1903+, Build 18362+)
- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: These instructions use Node Version Manager (nvm) for installation, you will need a recent version of NodeJS to run create-react-app, as well as a recent version of Node Package Manager (npm). For exact version requirements, see the Create React App website.

Create your React app

To install the full React toolchain on WSL, we recommend using create-react-app:

1. Open a terminal(Windows Command Prompt or PowerShell).
2. Create a new project folder: `mkdir ReactProjects` and enter that directory: `cd ReactProjects`.
3. Install React using create-react-app, a tool that installs all of the dependencies to build and run a full React.js application:

   ```
   npx create-react-app my-app
   ```

   NOTE
   npx is the package runner used by npm to execute packages in place of a global install. It basically creates a temporary install of React so that with each new project you are using the most recent version of React (not whatever version was current when you performed the global install above). Using the NPX package runner to execute a package can also help reduce the pollution of installing lots of packages on your machine.

4. This will first ask for your permission to temporarily install create-react-app and it's associated packages. Once completed, change directories into your new app ("my-app" or whatever you've chosen to call it): `cd my-app`.
5. Start your new React app:

   ```
   npm start
   ```

   This command will start up the Node.js server and launch a new browser window displaying your app.

You can use **Ctrl + c** to stop running the React app in your command line.

> **NOTE**
>
> Create React App includes a frontend build pipeline using Babel and webpack, but doesn't handle backend logic or databases. If you are seeking to build a server-rendered website with React that uses a Node.js backend, we recommend installing Next.js, rather than this create-react-app installation, which is intended more for single-page apps. You also may want to consider installing Gatsby if you want to build a static content-oriented website.

6. When you're ready to deploy your web app to production, running `npm run build` will create a build of your app in the "build" folder. You can learn more in the Create React App User Guide.

Additional resources

- React docs
- Create React App docs
- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install Next.js
- Install Gatsby
- Install React Native for Windows
- Install NodeJS on Windows
- Install NodeJS on WSL
- Try the tutorial: Using React in Visual Studio Code
- Try the Microsoft Learn online course: React

Get started build a desktop app with React Native for Windows

5/13/2021 • 2 minutes to read • Edit Online

React Native for Windows allows you to create a Universal Windows Platform (UWP) app using React.

Overview of React Native

React Native is an open-source mobile application framework created by Facebook. It is used to develop applications for Android, iOS, Web and UWP (Windows) providing native UI controls and full access to the native platform. Working with React Native requires an understanding of JavaScript fundamentals.

For more general information about React, see the React overview page.

Prerequisites

The setup requirements for using React Native for Windows can be found on the System Requirements page. Ensure Developer Mode is turned ON in Windows Settings App.

Install React Native for Windows

You can create a Windows desktop app using React Native for Windows by following these steps.

1. Open a command line window (terminal) and navigate to the directory where you want to create your Windows desktop app project.
2. You can use this command with the Node Package Executor (NPX) to create a React Native project without the need to install locally or globally install additional tools. The command will generate a React Native app in the directory specified by `<projectName>`.

   ```
   npx react-native init <projectName> --template react-native@^0.63.2
   ```

3. Switch to the project directory and run the following command to install the React Native for Windows packages:

   ```
   cd projectName
   npx react-native-windows-init --overwrite
   ```

4. To run the app, first launch your web browser (ie. Microsoft Edge), then execute the following command:

   ```
   npx react-native run-windows
   ```

Debug your React Native desktop app using Visual Studio

- Install Visual Studio 2019 with the following options checked:
  - Workloads: Universal Windows Platform development & Desktop development with C++.
  - Individual Components: Development activities & Node.js development support.

- Open the solution file in the application folder in Visual Studio (e.g., AwesomeProject/windows/AwesomeProject.sln if you used AwesomeProject as).
- Select the Debug configuration and the x64 platform from the combo box controls to the left of the Run button and underneath the Team and Tools menu item.
- Run `yarn start` from your project directory, and wait for the React Native packager to report success.
- Select **Run** to the right of the platform combo box control in VS, or select the Debug->Start without Debugging menu item. You now see your new app and Chrome should have loaded http://localhost:8081/debugger-ui/ in a new tab.
- Select the F12 key or Ctrl+Shift+I to open Developer Tools in your web browser.

Debug your React Native desktop app using Visual Studio Code

- Install Visual Studio Code
- Open your applications folder in VS Code.
- Install the React Native Tools plugin for VS Code.
- Create a new file in the applications root directory, .vscode/launch.json and paste the following configuration:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Debug Windows",
            "cwd": "${workspaceFolder}",
            "type": "reactnative",
            "request": "launch",
            "platform": "windows"
        }
    ]
}
```

- Press F5 or navigate to the debug menu (alternatively press Ctrl+Shift+D) and in the Debug dropdown select "Debug Windows" and press the green arrow to run the application.

Additional resources

- React Native for Windows docs
- React Native docs
- React docs
- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install NodeJS on Windows
- Try the Microsoft Learn online course: React

Get started developing for Android using React Native

4/21/2021 • 3 minutes to read • Edit Online

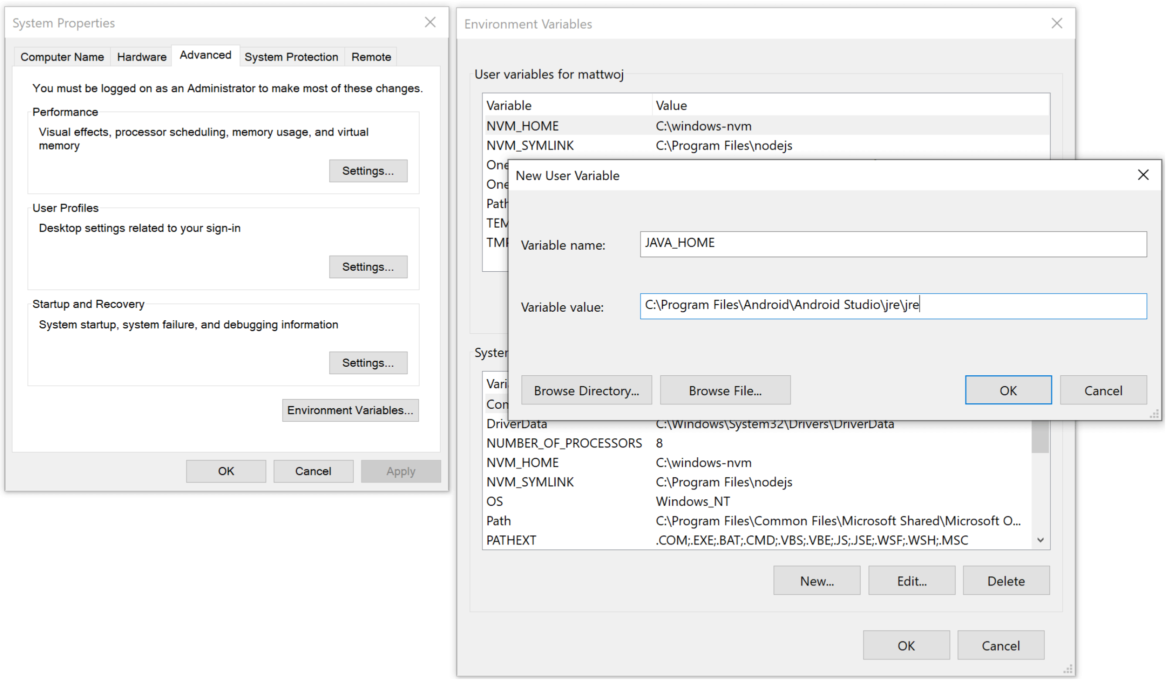
This guide will help you to get started using React Native on Windows to create a cross-platform app that will work on Android devices.

Overview

React Native is an open-source mobile application framework created by Facebook. It is used to develop applications for Android, iOS, Web and UWP (Windows) providing native UI controls and full access to the native platform. Working with React Native requires an understanding of JavaScript fundamentals.

Get started with React Native by installing required tools

1. Install Visual Studio Code (or your code editor of choice).
2. Install Android Studio for Windows. Android Studio installs the latest Android SDK by default. React Native requires Android 6.0 (Marshmallow) SDK or higher. We recommend using the latest SDK.
3. Create environment variable paths for the Java SDK and Android SDK:
   - In the Windows search menu, enter: "Edit the system environment variables", this will open the **System Properties** window.
   - Choose **Environment Variables...** and then choose **New...** under **User variables**.
   - Enter the Variable name and value (path). The default paths for the Java and Android SDKs are as follows. If you've chosen a specific location to install the Java and Android SDKs, be sure to update the variable paths accordingly.
     - JAVA_HOME: C:\Program Files\Android\Android Studio\jre\jre
     - ANDROID_HOME: C:\Users\username\AppData\Local\Android\Sdk



User variables for mattwoj

Variable	Value
NVM_HOME	C:\windows-nvm
NVM_SYMLINK	C:\Program Files\nodejs

New User Variable

Variable name: JAVA_HOME

Variable value: C:\Program Files\Android\Android Studio\jre\jre

System variables

Variable	Value
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
NVM_HOME	C:\windows-nvm
NVM_SYMLINK	C:\Program Files\nodejs
OS	Windows_NT
Path	C:\Program Files\Common Files\Microsoft Shared\Microsoft O...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

4. Install NodeJS for Windows You may want to consider using Node Version Manager (nvm) for Windows

if you will be working with multiple projects and version of NodeJS. We recommend installing the latest LTS version for new projects.

NOTE

You may also want to consider installing and using the Windows Terminal for working with your preferred command-line interface (CLI), as well as, Git for version control. The Java JDK comes packaged with Android Studio v2.2+, but if you need to update your JDK separately from Android Studio, use the Windows x64 Installer.

Create a new project with React Native

1. Use npx, the package runner tool that is installed with **npm** to create a new React Native project. from the Windows Command Prompt, PowerShell, Windows Terminal, or the integrated terminal in VS Code (View > Integrated Terminal).

   ```
   npx react-native init MyReactNativeApp
   ```

2. Open your new "MyReactNativeApp" directory:

   ```
   cd MyReactNativeApp
   ```

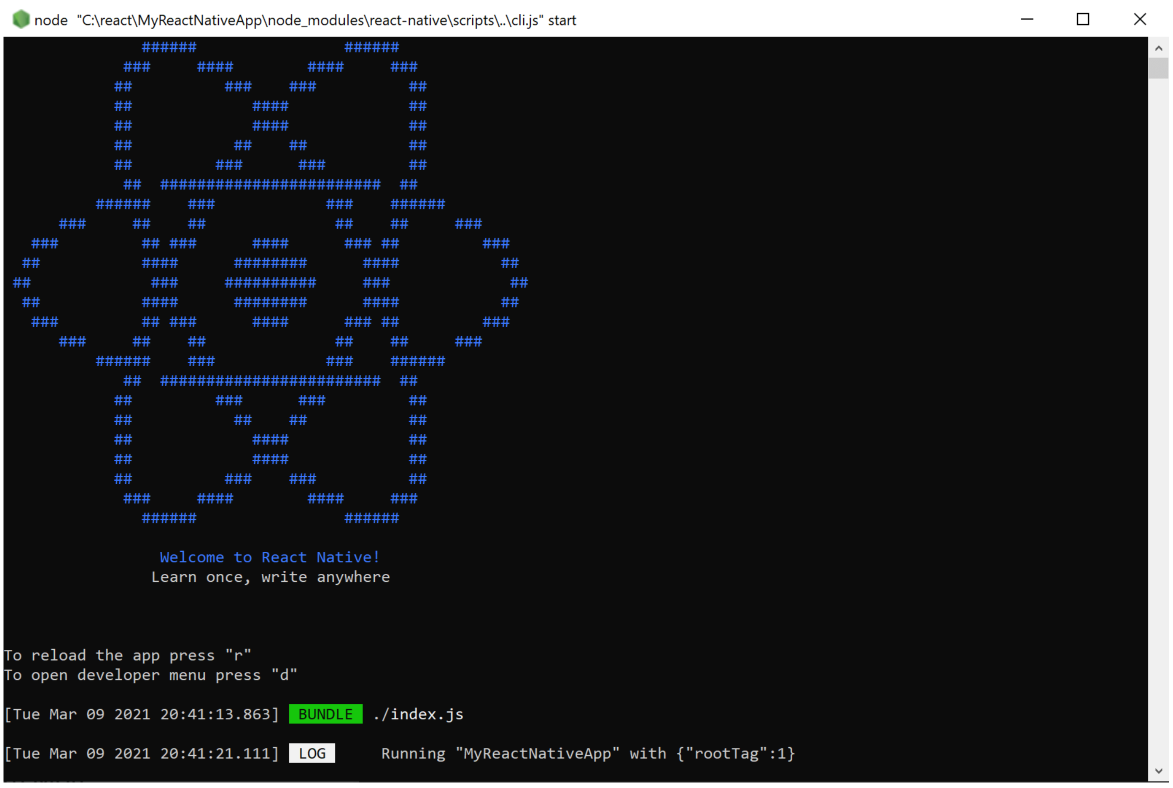
3. If you want to run your project on a hardware Android device, connect the device to your computer with a USB cable.

4. If you want to run your project on an Android emulator, you shouldn't need to take any action as Android Studio installs with a default emulator installed. If you want to run your app on the emulator for a particular device. Click on the **AVD Manager** button in the toolbar.

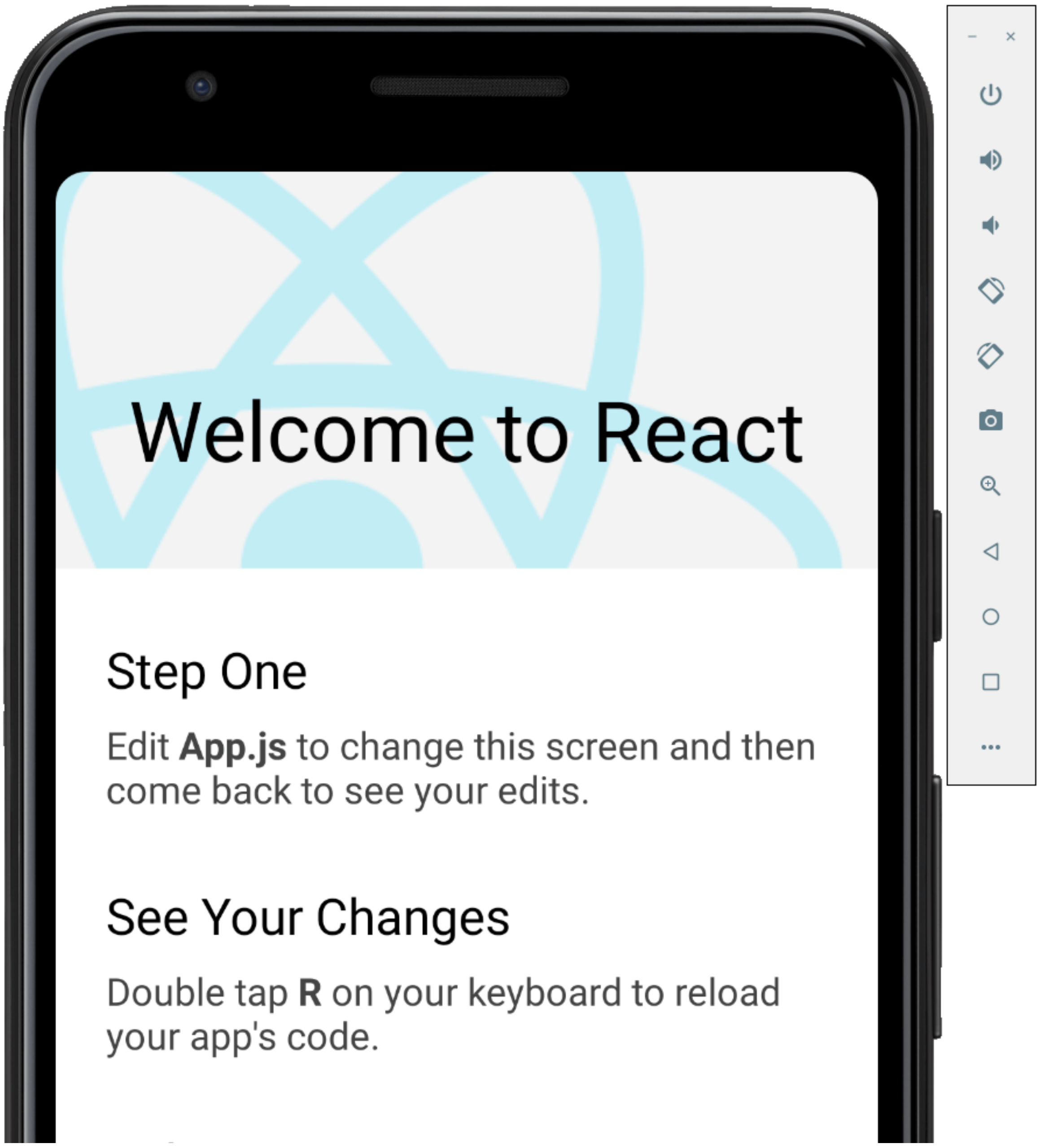
   .

5. To run your project, enter the following command. This will open a new console window displaying Node Metro Bundler.

   ```
   npx react-native run-android
   ```



```
node  "C:\react\MyReactNativeApp\node_modules\react-native\scripts\..\cli.js" start
               ######                ######
             ###     ####        ####     ###
            ##          ###    ###          ##
            ##             ####             ##
            ##             ####             ##
            ##           ##    ##           ##
            ##         ###      ###         ##
             ##  ########################  ##
          ######    ###            ###    ######
      ###     ##    ##              ##    ##     ###
   ###         ## ###      ####      ### ##         ###
  ##           ####      ########      ####           ##
 ##             ###     ##########     ###             ##
  ##           ####      ########      ####           ##
   ###         ## ###      ####      ### ##         ###
      ###     ##    ##              ##    ##     ###
          ######    ###            ###    ######
             ##  ########################  ##
            ##         ###      ###         ##
            ##           ##    ##           ##
            ##             ####             ##
            ##             ####             ##
            ##          ###    ###          ##
             ###     ####        ####     ###
               ######                ######

                 Welcome to React Native!
                Learn once, write anywhere


To reload the app press "r"
To open developer menu press "d"

[Tue Mar 09 2021 20:41:13.863]  BUNDLE  ./index.js

[Tue Mar 09 2021 20:41:21.111]  LOG      Running "MyReactNativeApp" with {"rootTag":1}
```



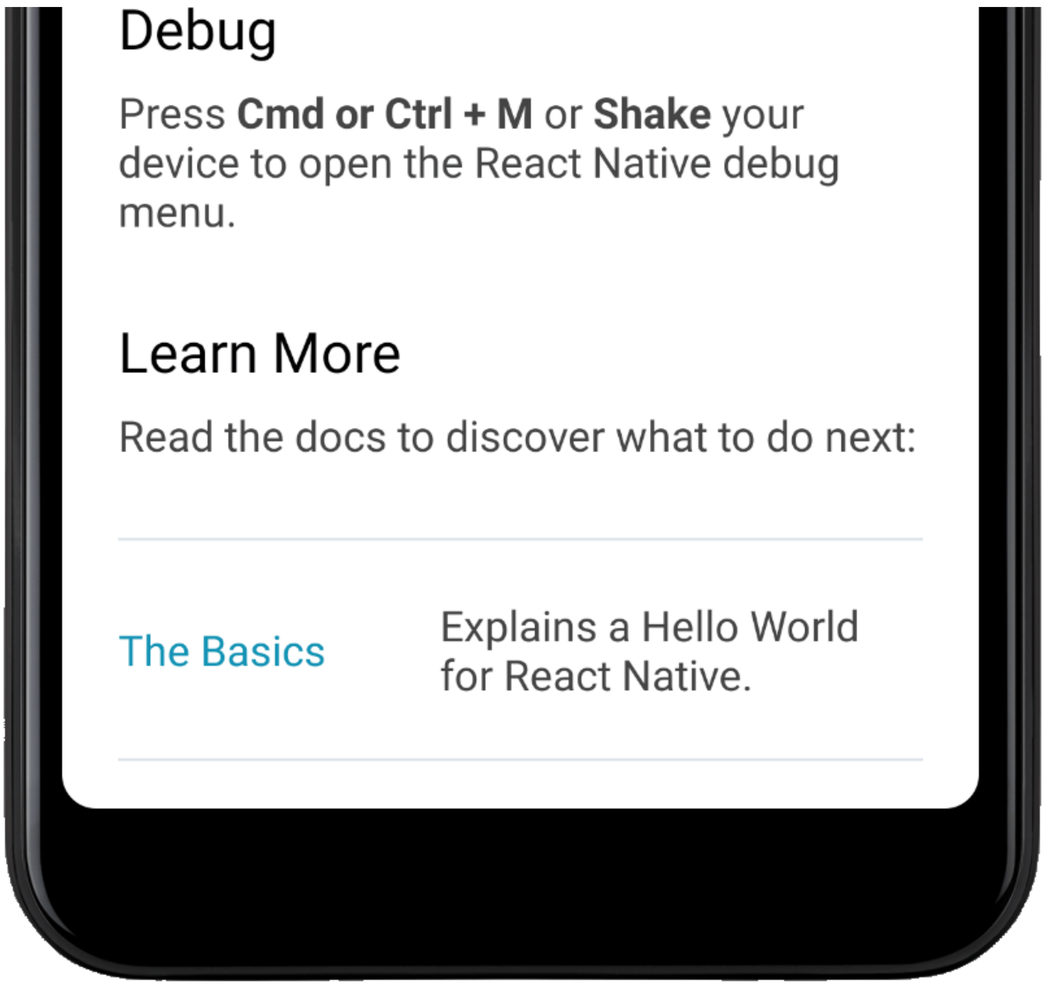
Welcome to React

Step One

Edit **App.js** to change this screen and then come back to see your edits.

See Your Changes

Double tap **R** on your keyboard to reload your app's code.



Debug

Press **Cmd or Ctrl + M** or **Shake** your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:

The Basics | Explains a Hello World for React Native.

> **NOTE**
>
> If you are using a new install of Android Studio and haven't yet done any other Android development, you may get errors at the command line when you run the app about accepting licenses for the Android SDK. Such as "Warning: License for package Android SDK Platform 29 not accepted." To resolve this, you can click the **SDK Manager** button in Android Studio . Or, you can list and accept the licenses with the following command, making sure to use the path to the SDK location on your machine.

```
C:\Users\[User Name]\AppData\Local\Android\Sdk\tools\bin\sdkmanager --licenses
```

6. To modify the app, open the `MyReactNativeApp` project directory in the IDE of your choice. We recommend VS Code or Android Studio.

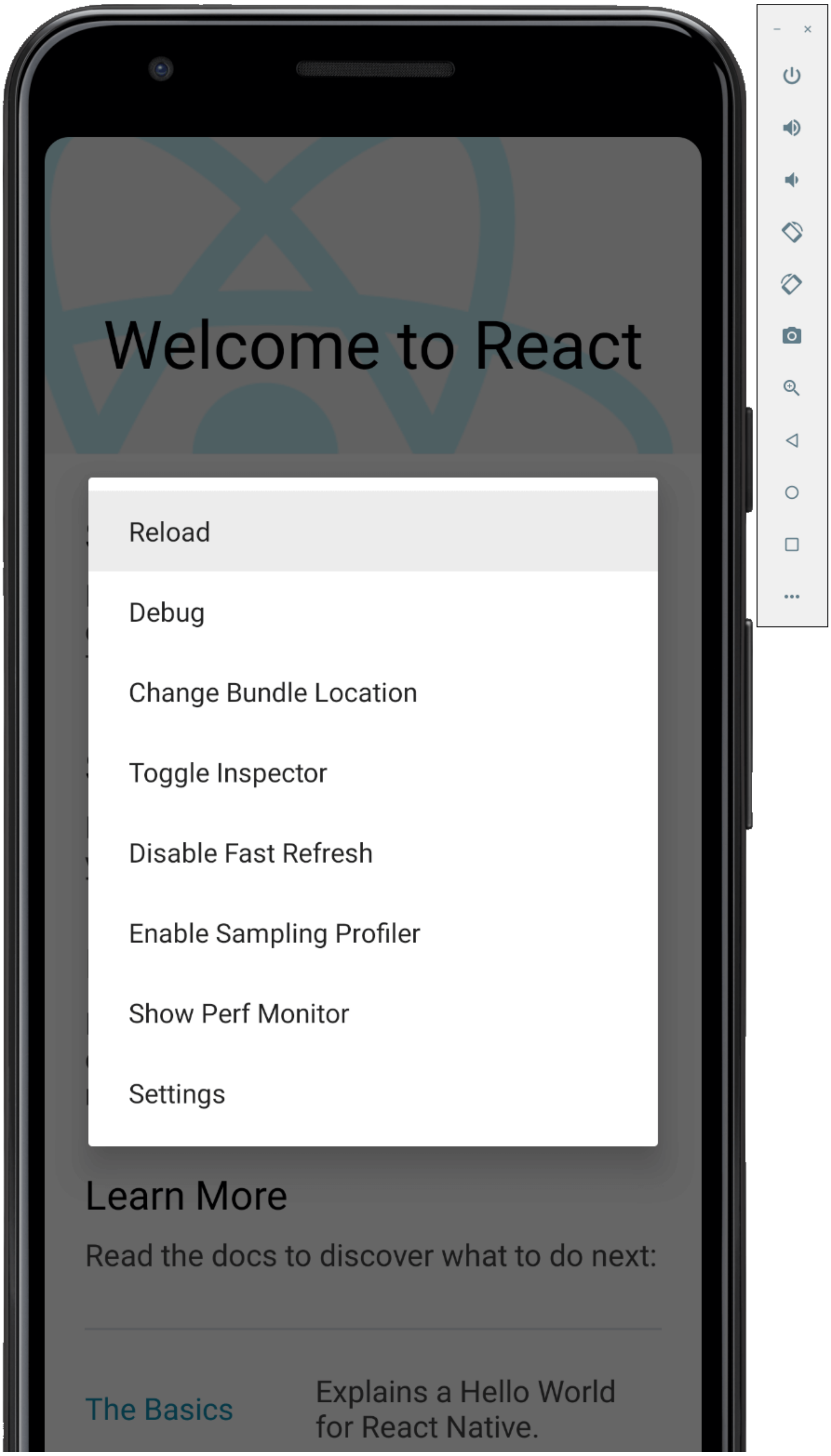
7. The project template created by `react-native init` uses a main page named `App.js`. This page is pre-populated with a lot of useful links to information about React Native development. Add some text to the first **Text** element, like the "HELLO WORLD!" string shown below.

```
<Text style={styles.sectionDescription}>
  Edit <Text style={styles.highlight}>App.js</Text> to change this
  screen and then come back to see your edits. HELLO WORLD!
</Text>
```

8. Reload the app to show the changes you made. There are several ways to do this.

   - In the Metro Bundler console window, type "r".

- In the Android device emulator, double tap "r" on your keyboard.
- On a hardware android device, shake the device to bring up the React Native debug menu and select `Reload'.



Additional resources

- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance
- Enable Virtualization support to improve Emulator performance

Get started with Next.js on Windows

5/24/2021 • 3 minutes to read • Edit Online

A guide to help you install the Next.js web framework and get up and running on Windows 10.

Next.js is a framework for creating server-rendered JavaScript apps based on React.js, Node.js, Webpack and Babel.js. It is basically a project boilerplate for React, crafted with attention to best practices, that allows you to create "universal" web apps in a simple, consistent way, with hardly any configuration. These "universal" server-rendered web apps are also sometimes called “isomorphic”, meaning that code is shared between the client and server.

To learn more about React and other JavaScript frameworks based on React, see the React overview page.

Prerequisites

This guide assumes that you've already completed the steps to set up your Node.js development environment, including:

- Install the latest version of Windows 10 (Version 1903+, Build 18362+)
- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: This includes a version manager, package manager, Visual Studio Code, and the Remote Development extension.

We recommend using the Windows Subsystem for Linux when working with NodeJS apps for better performance speed, system call compatibility, and for parody when running Linux servers or Docker containers.

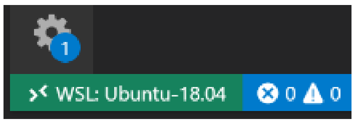
IMPORTANT

Installing a Linux distribution with WSL will create a directory for storing files: `\\wsl\Ubuntu-20.04` (substitute Ubuntu-20.04 with whatever Linux distribution you're using). To open this directory in Windows File Explorer, open your WSL command line, select your home directory using `cd ~`, then enter the command `explorer.exe .` Be careful not to install NodeJS or store files that you will be working with on the mounted C drive (`/mnt/c/Users/yourname$`). Doing so will significantly slow down your install and build times.

Install Next.js

To install Next.js, which includes installing next, react, and react-dom:

1. Open a WSL command line (ie. Ubuntu).
2. Create a new project folder: `mkdir NextProjects` and enter that directory: `cd NextProjects`.
3. Install Next.js and create a project (replacing 'my-next-app' with whatever you'd like to call your app): `npx create-next-app my-next-app`.
4. Once the package has been installed, change directories into your new app folder, `cd my-next-app`, then use `code .` to open your Next.js project in VS Code. This will allow you to look at the Next.js framework that has been created for your app. Notice that VS Code opened your app in a WSL-Remote environment (as indicated in the green tab on the bottom-left of your VS Code window). This means that while you are using VS Code for editing on the Windows OS, it is still running your app on the Linux OS.

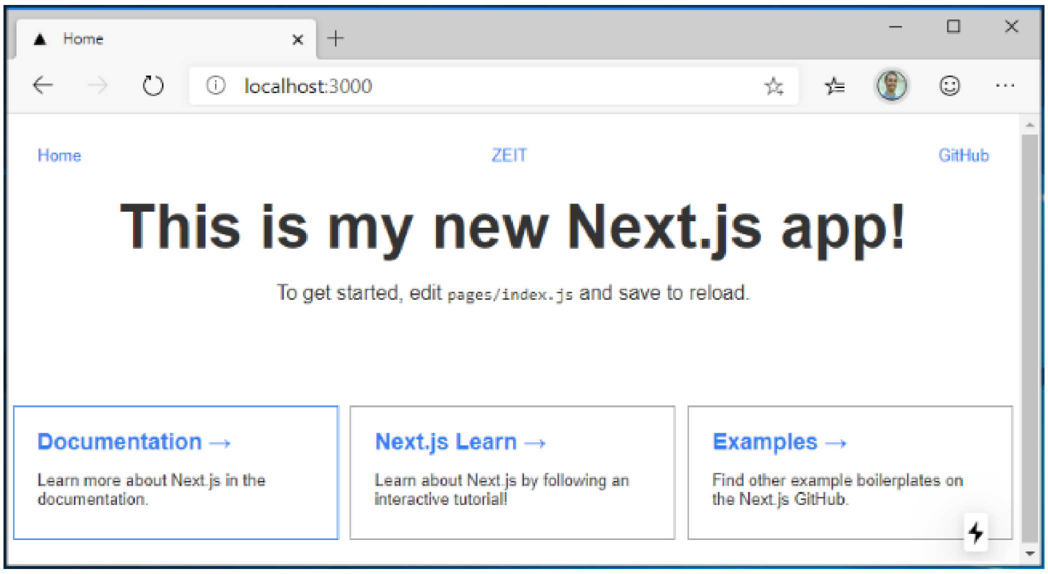


5. There are 3 commands you need to know once Next.js is installed:

   - `npm run dev` for running a development instance with hot-reloading, file watching and task re-running.
   - `npm run build` for compiling your project.
   - `npm start` for starting your app in production mode.

   Open the WSL terminal integrated in VS Code (**View** > **Terminal**). Make sure that the terminal path is pointed to your project directory (ie. `~/NextProjects/my-next-app$`). Then try running a development instance of your new Next.js app using: `npm run dev`

6. The local development server will start and once your project pages are done building, your terminal will display "compiled successfully - ready on http://localhost:3000". Select this localhost link to open your new Next.js app in a web browser.



This is my new Next.js app!

To get started, edit `pages/index.js` and save to reload.

Documentation →
Learn more about Next.js in the documentation.

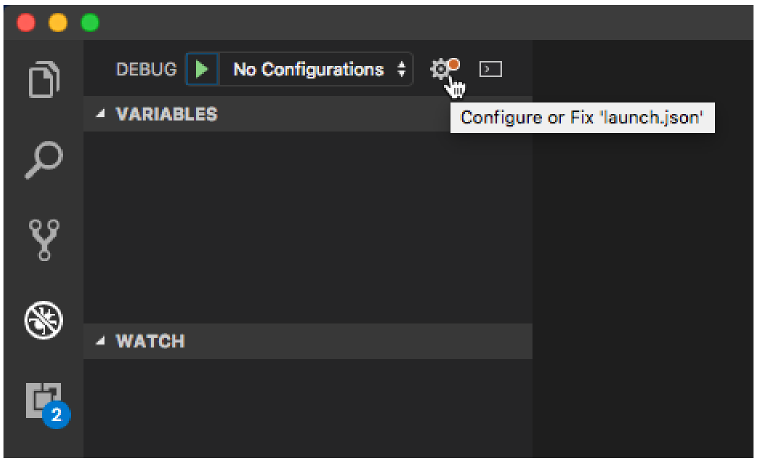
Next.js Learn →
Learn about Next.js by following an interactive tutorial!

Examples →
Find other example boilerplates on the Next.js GitHub.

7. Open the `pages/index.js` file in your VS Code editor. Find the page title `<h1 className='title'>Welcome to Next.js!</h1>` and change it to `<h1 className='title'>This is my new Next.js app!</h1>`. With your web browser still open to localhost:3000, save your change and notice the hot-reloading feature automatically compile and update your change in the browser.

8. Let's see how Next.js handles errors. Remove the `</h1>` closing tag so that your title code now looks like this: `<h1 className='title'>This is my new Next.js app!`. Save this change and notice that a "Failed to compile" error will display in your browser, and in your terminal, letting your know that a closing tag for `<h1>` is expected. Replace the `</h1>` closing tag, save, and the page will reload.

You can use VS Code's debugger with your Next.js app by selecting the F5 key, or by going to **View** > **Debug** (Ctrl+Shift+D) and **View** > **Debug Console** (Ctrl+Shift+Y) in the menu bar. If you select the gear icon in the Debug window, a launch configuration (`launch.json`) file will be created for you to save debugging setup details. To learn more, see VS Code Debugging.



To learn more about Next.js, see the Next.js docs.

Get started with Gatsby.js on Windows

4/21/2021 • 4 minutes to read • Edit Online

A guide to help you install the Gatsby.js web framework and get up and running on Windows 10.

Gatsby.js is a static site generator framework based on React.js, as opposed to being server-rendered like Next.js. A static site generator generates static HTML on build time. It doesn't require a server. Next.js generates HTML on runtime (each time a new request comes in), requiring a server to run. Gatsby also dictates how to handle data in your app (with GraphQL), whereas Next.js leaves that decision up to you.

To learn more about React and other JavaScript frameworks based on React, see the React overview page.

Prerequisites

This guide assumes that you've already completed the steps to set up your Node.js development environment, including:

- Install the latest version of Windows 10 (Version 1903+, Build 18362+)
- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: This includes a version manager, package manager, Visual Studio Code, and the Remote Development extension.

We recommend using the Windows Subsystem for Linux when working with NodeJS apps for better performance speed, system call compatibility, and for parody when running Linux servers or Docker containers.

IMPORTANT

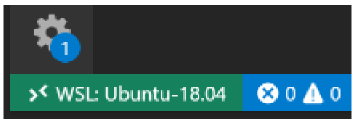
Installing a Linux distribution with WSL will create a directory for storing files: `\\wsl\Ubuntu-20.04` (substitute Ubuntu-20.04 with whatever Linux distribution you're using). To open this directory in Windows File Explorer, open your WSL command line, select your home directory using `cd ~`, then enter the command `explorer.exe .` Be careful not to install NodeJS or store files that you will be working with on the mounted C drive (`/mnt/c/Users/yourname$`). Doing so will significantly slow down your install and build times.

Install Gatsby.js

To create a Gatsby.js project:

1. Open your WSL terminal (ie. Ubuntu 18.04).
2. Create a new project folder: `mkdir GatsbyProjects` and enter that directory: `cd GatsbyProjects`
3. Use npm to install the Gatsby CLI: `npm install -g gatsby-cli`. Once installed, check the version with `gatsby --version`.
4. Create your Gatsby.js project: `gatsby new my-gatsby-app`
5. Once the package has been installed, change directories into your new app folder, `cd my-gatsby-app`, then use `code .` to open your Gatsby project in VS Code. This will allow you to look at the Gatsby.js framework that has been created for your app using VS Code's File Explorer. Notice that VS Code opened your app in a WSL-Remote environment (as indicated in the green tab on the bottom-left of your VS Code window). This means that while you are using VS Code for editing on the Windows OS, it is still

running your app on the Linux OS.



6. There are 3 commands you need to know once Gatsby is installed:

   - `gatsby develop` for running a development instance with hot-reloading.
   - `gatsby build` for creating a production build.
   - `gatsby serve` for starting your app in production mode.

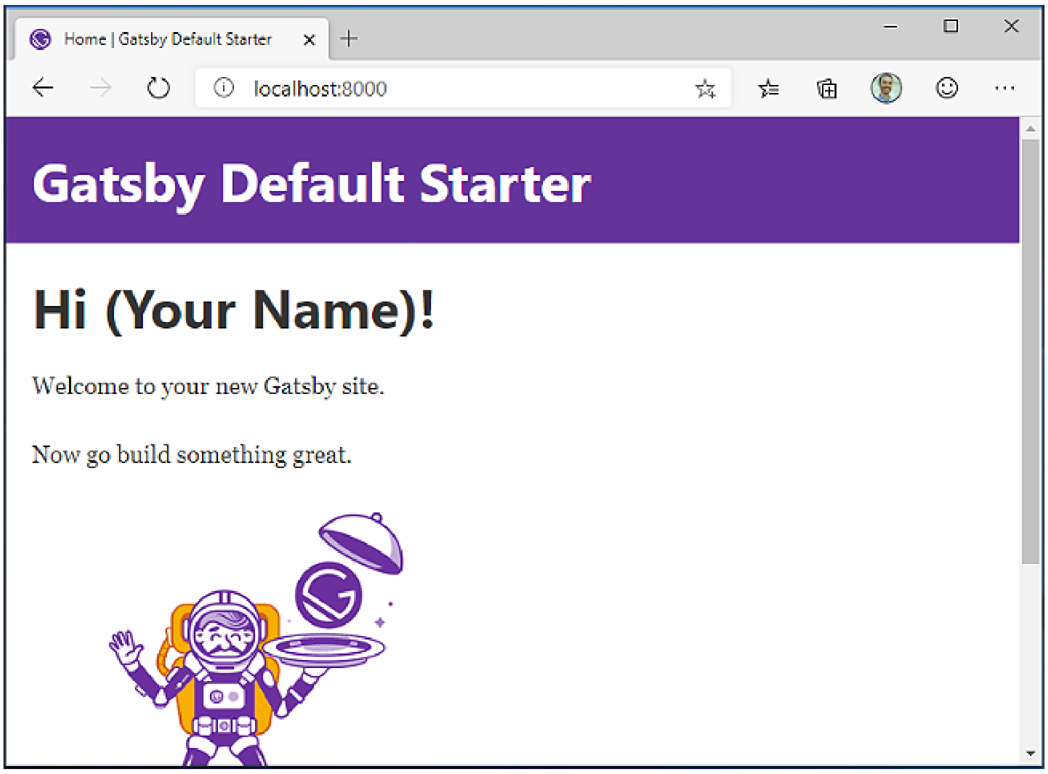
   Open the WSL terminal integrated in VS Code (**View > Terminal**). Make sure that the terminal path is pointed to your project directory (ie. `~/GatsbyProjects/my-gatsby-app$`). Then try running a development instance of your new app using: `gatsby develop`

7. Once your new Gatsby project finishes compiling, your terminal will display that "You can now view gatsby-starter-default in the browser. http://localhost:8000/." Select this localhost link to view your new project built in a web browser.

NOTE

You'll notice that your terminal output also let's you know that you can "View GraphiQL, an in-browser IDE, to explore your site's data and schema: http://localhost:8000/___graphql." GraphQL consolidates your APIs into a self-documenting IDE (GraphiQL) built into Gatsby. In addition to exploring your site's data and schema, you can perform GraphQL operations such as queries, mutations, and subscriptions. For more info, see Introducing GraphiQL.

8. Open the `src/pages/index.js` file in your VS Code editor. Find the page title `<h1 >Hi people</h1>` and change it to `<h1 >Hi (Your Name)!</h1>`. With your web browser still open to localhost:8000, save your change and notice the hot-reloading feature automatically compile and update your change in the browser.



Gatsby Default Starter

Hi (Your Name)!

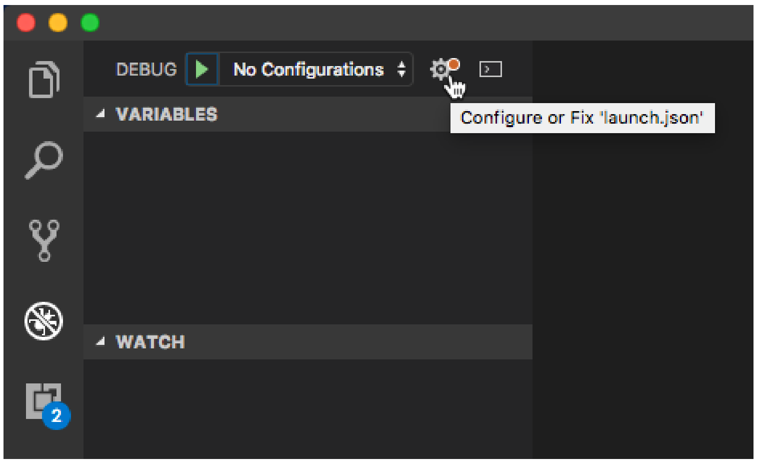
Welcome to your new Gatsby site.

Now go build something great.

9. Let's see how Next.js handles errors. Remove the `</h1>` closing tag so that your title code now looks like this: `<h1>Hi (Your Name)!`. Save this change and notice that a "Failed to compile" error will display in your

browser, and in your terminal, letting your know that a closing tag for `<h1>` is expected. Replace the `</h1>` closing tag, save, and the page will reload.

You can use VS Code's debugger with your Next.js app by selecting the F5 key, or by going to **View > Debug** (Ctrl+Shift+D) and **View > Debug Console** (Ctrl+Shift+Y) in the menu bar. If you select the gear icon in the Debug window, a launch configuration (`launch.json`) file will be created for you to save debugging setup details. To learn more, see VS Code Debugging.



To learn more about Gatsby, see the Gatsby.js docs.

Tutorial: React on Windows for beginners

5/13/2021 • 6 minutes to read • Edit Online

If you're brand new to using React, this guide will help you to get started with some basics.

- A few basic terms and concepts
- Try using React in Visual Studio Code
- Try using React with an API

Prerequisites

- Install React *(Should I install on Windows or Windows Subsystem for Linux)*
  - Install React on Windows
  - Install React on Windows Subsystem for Linux
- Install VS Code. We recommend installing VS Code on Windows, regardless of whether you plan to use React on Windows or WSL.

A few basic terms and concepts

React is a JavaScript library for building user interfaces.

- It is open-source, meaning that you can contribute to it by filing issues or pull requests. *(Just like these docs!)*
- It is declarative, meaning that you write the code that you want and React takes that declared code and performs all of the JavaScript/DOM steps to get the desired result.
- It is component-based, meaning that applications are created using prefabricated and reusable independent code modules that manage their own state and can be glued together using the React framework, making it possible to pass data through your app while keeping state out of the DOM.
- The React motto is "Learn once, write anywhere." The intention is for code reuse and not making assumptions about how you will use React UI with other technologies, but to make components reusable without the need to rewrite existing code.
- JSX is a syntax extension for JavaScript written to be used with React that looks like HTML, but is actually a JavaScript file that needs to be compiled, or translated into regular JavaScript.
- Virtual DOM: DOM stands for Document Object Model and represents the UI of your app. Every time the state of your app's UI changes, the DOM gets updated to represent the change. When a DOM is frequently updating, performance becomes slow. A Virtual DOM is only a visual representation of the DOM, so when the state of the app changes, the virtual DOM is updated rather than the real DOM, reducing the performance cost. It's a *representation* of a DOM object, like a lightweight copy.
- *Views*: are what the user sees rendered in the browser. In React, view is related to the concept of rendering elements that you want a user to see on their screen.
- State: refers to the data stored by different views. The state will typically rely on who the user is and what the user is doing. For example, signing into a website may show your user profile (view) with your name (state). The state data will change based on the user, but the view will remain the same.

Try using React in Visual Studio Code

There are many ways to create an application with React (see the React Overview for examples). This tutorial will walk through how to use create-react-app to fast-forward the set up for a functioning React app so that you can see it running and focus on experimenting with the code, not yet concerning yourself with the build tools.

1. Use create-react-app on Windows or WSL (see the prerequisites above) to create a new project: `npx create-react-app hello-world`

2. Change directories so that you're inside the folder for your new app: `cd hello-world` and start your app: `npm start`

   Your new React Hello World app will compile and open your default web browser to show that it's running on localhost:3000.

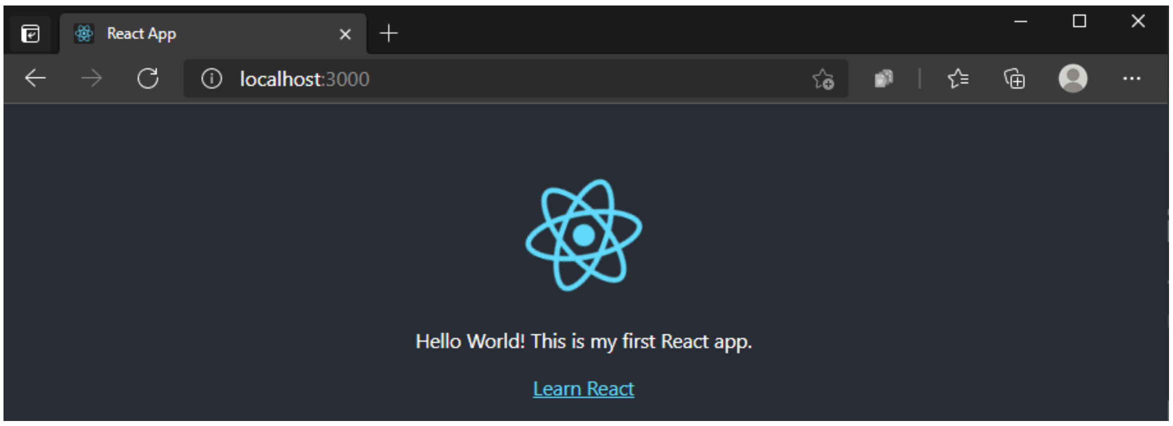
3. Stop running your React app (Ctrl+c) and open it's code files in VS Code by entering: `code .`

4. Find the src/App.js file and find the header section that reads:

   ```
   <p>Edit <code>src/App.js</code> and save to reload.</p>
   ```

   Change it to read:

   ```
   <p>Hello World! This is my first React app.</p>
   ```

   Hello World! This is my first React app.

   Learn React

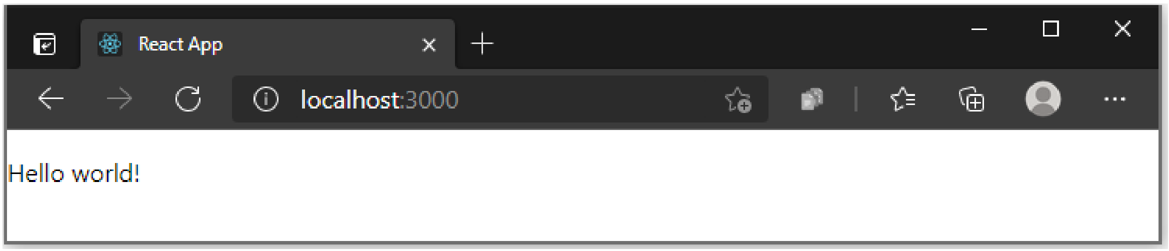
Try using React with an API

Using the same Hello World! app that you built with React and updated with Visual Studio Code, let's try adding an API call to display some data.

1. First, let's remove everything from that app.js file and make it into a class component. We will first import *component* from React and use it to create the class component. (There are two types of components: class and function). We will also add some custom JSX code in a `return()` statement. You can reload the page to see the result.

   Your app.js file should now look like this:

```
import React, { Component } from 'react';

class App extends Component {
  render() {
    return (
      <p>Hello world! This is my first React app.</p>
    );
  }
}
export default App;
```



Hello world!

2. Next, let's set a local state where we can save data from an API. A state object is where we can store data to be used in the view. The view is rendered to the page inside of `render()`.

   To add a local state, we need to first add a constructor. When implementing the constructor for a React.Component subclass, you should call `super(props)` before any other statement. Otherwise, `this.props` will be undefined in the constructor, which can lead to bugs. Props are what pass data down into components.

   We also need to initialize the local state and assign an object to `this.state`. We will use "posts" as an empty array that we can fill with post data from an API.

   Your app.js file should now look like this:

```
import React, { Component } from 'react';

class App extends Component {
constructor(props) {
    super(props);
    this.state = {
      posts: []
    }
  }
  render() {
    return (
      <p>Hello world!</p>
    );
  }
}
export default App;
```

3. To call an API with data for us to use in our React app, we will use the .fetch JavaScript method. The API we will call is JSONPlaceholder, a free API for testing and prototyping that serves up fake placeholder data in JSON format. The `componentDidMount` method is used to mount the `fetch` to our React component. The data from the API is saved in our state (using the setState request).

```
import React, { Component } from 'react';

class App extends Component {
constructor(props) {
    super(props);
    this.state = {
      posts: []
    }
  }
  componentDidMount() {
    const url = "https://jsonplaceholder.typicode.com/albums/1/photos";
    fetch(url)
    .then(response => response.json())
    .then(json => this.setState({ posts: json }))
  }
  render() {
    return (
      <p>Hello world!</p>
    );
  }
}
export default App;
```

4. Let's take a look at what sort of data the API has saved in our `posts` state. Below is some of the contents of the fake JSON API file. We can see the format the data is listed in, using the categories: "albumId", "id", "title", "url", and "thumbnailUrl".

```
[
  {
    "albumId": 1,
    "id": 1,
    "title": "accusamus beatae ad facilis cum similique qui sunt",
    "url": "https://via.placeholder.com/600/92c952",
    "thumbnailUrl": "https://via.placeholder.com/150/92c952"
  },
  {
    "albumId": 1,
    "id": 2,
    "title": "reprehenderit est deserunt velit ipsam",
    "url": "https://via.placeholder.com/600/771796",
    "thumbnailUrl": "https://via.placeholder.com/150/771796"
  }
]
```

5. We will need to add some page styling to display our API data. Let's just use Bootstrap to handle the styling for us. We can copy + paste the Bootstrap CDN stylesheet reference inside the `./public/index.html` file of our React app.

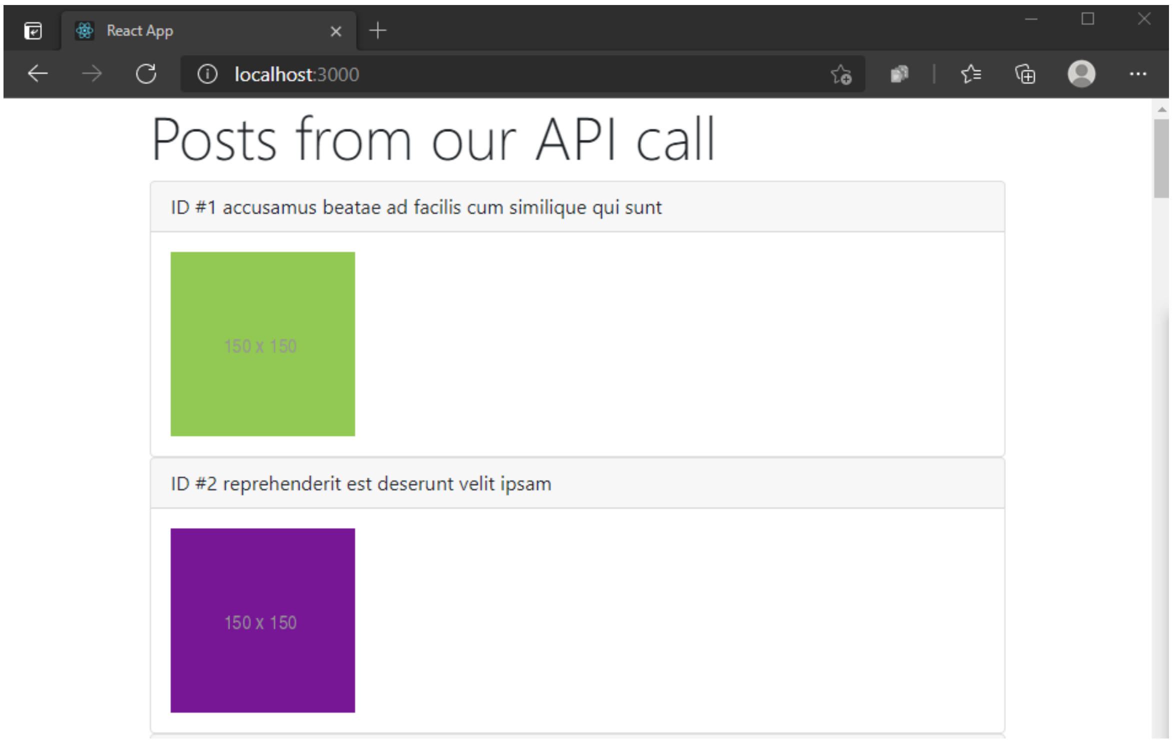
```
    <!-- Bootstrap -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6"
crossorigin="anonymous">

    <title>React App</title>
  </head>
  <body>
```

6. To display the API data, referencing our Bootstrap classes for styling, we will now need to add a bit of JSX code inside the rendered `return()` statement. We will add a container, a header ("Posts from our API call"), and a card for each piece of data from our API. We will use the `map()` method to display our data

from the `posts` object that we stored it in as keys. Each card will display a header with "ID #" and then the post.id key value + post.title key value from our JSON data. Followed by the body displaying the image based on the thumbnailURL key value.

```
render() {
  const { posts } = this.state;
  return (
    <div className="container">
      <div class="jumbotron">
        <h1 class="display-4">Posts from our API call</h1>
      </div>
      {posts.map((post) => (
        <div className="card" key={post.id}>
          <div className="card-header">
            ID #{post.id} {post.title}
          </div>
          <div className="card-body">
            <img src={post.thumbnailUrl}></img>
          </div>
        </div>
      ))}
    </div>
  );
}
```

7. Run your React app again: `npm start` and take a look in your local web browser on `localhost:3000` to see your API data being displayed.



Posts from our API call

ID #1 accusamus beatae ad facilis cum similique qui sunt

ID #2 reprehenderit est deserunt velit ipsam

Additional resources

- The official React docs offer all of the latest, up-to-date information on React
- Microsoft Edge Add-ons for React Developer Tools: Add two tabs to your Microsoft Edge dev tools to help with your React development: Components and Profiler.
- Microsoft Learn: The React Learning Path contains online course modules to help you get started with the basics.

What is Vue.js?

5/13/2021 • 2 minutes to read • Edit Online

Vue is an open-source, front end JavaScript framework for building user interfaces and single-page applications on the web. Created by Evan You, released in 2014 and maintained by Evan and his core team, Vue focuses on declarative rendering and component composition offering a core library for the view layer only.

If you want to build a server-rendered Vue web app with advanced features such as routing, state management and build tooling, take a look at Nuxt.js.

What makes Vue unique?

Vue uses a model-view-viewmodel architecture. Evan You previously worked on AngularJS at Google and extracted parts of Angular to offer a more lightweight framework. Vue is in may ways similar to React, Angular, Ember, Knockout, etc. See the Vue documentation for a more in-depth comparison to these other JavaScript frameworks.

What can you do with Vue?

- Build a single-page app (SPA)
- Use just a component of Vue to add a simple to-do list to your app or find more complex examples
- Build a server-rendered website with a Node.js backend, with help from Nuxt.js

Vue tools

Vue.js is focused only on the view layer, so may require additional tools to create a more complex app. You may want to consider using:

- Package manager: Two popular package managers for Vue are npm (which is included with NodeJS) and yarn. Both support a broad library of well-maintained packages that can be installed.
- Vue CLI: a standard toolkit for rapid Vue.js development with out-of-the-box support for Babel, PostCSS, TypeScript, ESLint, etc.
- Nuxt.js: A framework to make server-side rendered Vue.js apps possible. Server-side rendering can improve SEO and make user interfaces more responsive.
- Vue extension pack for VS Code: Adds syntax highlighting, code formatting, and code snippets to your .vue files.
- Vuetify: A Vue UI library offering Material Design Framework components.
- Vuesion: A Vue boilerplate for production-ready Progressive Web Apps (PWAs).
- Storybook: A development and testing environment for Vue user interface components.
- Vue Router: Supports mapping application URLs to Vue components.
- Vue Design System: An open source tool for building Design Systems with Vue.js.
- VueX: State management system for Vue apps.

Additional resources

- Vue docs
- Vue.js overview
- Should I install on Windows or Windows Subsystem for Linux (WSL)?

- Install Vue.js on WSL
- Install Vue.js on Windows
- Install Nuxt.js
- Microsoft Learn online course: Take your first steps with Vue.js
- Try a Vue tutorial with VS Code

Install Vue.js on Windows Subsystem for Linux

5/13/2021 • 2 minutes to read • Edit Online

A guide to help you set up a Vue.js development environment on Windows 10 by installing the Vue.js web framework on Windows Subsystem for Linux (WSL). Learn more on the Vue.js overview page.

Vue can be installed directly on Windows or on WSL. We generally recommend installing on WSL if you are planning to interact with a NodeJS backend, want parody with a Linux production server, or plan to follow along with a tutorial that utilizes Bash commands. For more info, see Should I install on Windows or Windows Subsystem for Linux?.

Prerequisites

- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: This includes a version manager, package manager, Visual Studio Code, and the Remote Development extension. The Node Package Manager (npm) is used to install Vue.js.

IMPORTANT

Installing a Linux distribution with WSL will create a directory for storing files: `\\wsl\Ubuntu-20.04` (substitute Ubuntu-20.04 with whatever Linux distribution you're using). To open this directory in Windows File Explorer, open your WSL command line, select your home directory using `cd ~`, then enter the command `explorer.exe .` Be careful not to install or store files that you will be working with on the mounted C drive (`/mnt/c/Users/yourname$`). Doing so will significantly slow down your install and build times.

Install Vue.js

To install Vue.js on WSL:

1. Open a WSL command line (ie. Ubuntu).
2. Create a new project folder: `mkdir VueProjects` and enter that directory: `cd VueProjects`.
3. Install Vue.js using Node Package Manager (npm):

```
npm install vue
```

Check the version number you have installed by using the command: `vue --version`.

NOTE

To install Vue.js using a CDN, rather than NPM, see the Vue.js install docs.

Install Vue CLI

Vue CLI is a toolkit for working with Vue in your terminal / command line. It enables you to quickly scaffold a new project (vue create), prototype new ideas (vue serve), or manage projects using a graphical user interface (vue ui). Vue CLI is a globally installed npm package that handles some of the build complexities (like using

Babel or Webpack) for you. *If you are not building a new single-page app, you may not need or want Vue CLI.*

To install Vue CLI, use npm. You must use the `-g` flag to globally install in order to upgrade (`vue upgrade --next`):

```
npm install -g @vue/cli
```

To learn more about additional plugins that can be added (such as linting or Apollo for integrating GraphQL), visit Vue CLI plugins in the Vue CLI docs.

Additional resources

- Vue docs
- Vue.js overview
- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install Vue.js on Windows
- Install Nuxt.js
- Microsoft Learn online course: Take your first steps with Vue.js
- Try a Vue tutorial with VS Code

Install Vue.js directly on Windows

7/16/2021 • 2 minutes to read • Edit Online

A guide to help you set up a Vue.js development environment on Windows 10. Learn more on the Vue.js overview page.

Vue can be installed directly on Windows or on the Windows Subsystem for Linux (WSL). We generally recommend that you install Vue on WSL if you are planning to interact with a NodeJS backend, want parity with a Linux production server, or plan to follow along with a tutorial that utilizes Bash commands. For more info, see Should I install on Windows or Windows Subsystem for Linux?.

Prerequisites

- Install Node.js on Windows: This includes a version manager, package manager, and Visual Studio Code. The Node Package Manager (npm) is used to install Vue.js.

Install Vue.js

To install Vue.js:

1. Open a command line (ie. Windows Command Prompt or PowerShell).
2. Create a new project folder: `mkdir VueProjects` and enter that directory: `cd VueProjects`.
3. Install Vue.js using Node Package Manager (npm):

```
npm install vue
```

Check the version number you have installed by using the command: `vue --version`.

NOTE

To install Vue.js using a CDN, rather than NPM, see the Vue.js install docs. See the Vue docs for an Explanation of different Vue builds.

Install Vue CLI

Vue CLI is a toolkit for working with Vue in your terminal / command line. It enables you to quickly scaffold a new project (vue create), prototype new ideas (vue serve), or manage projects using a graphical user interface (vue ui). Vue CLI is a globally installed npm package that handles some of the build complexities (like using Babel or Webpack) for you. *If you are not building a new single-page app, you may not need or want Vue CLI.*

To install Vue CLI, use npm. You must use the `-g` flag to globally install in order to upgrade (`vue upgrade --next`):

```
npm install -g @vue/cli
```

To learn more about additional plugins that can be added (such as linting or Apollo for integrating GraphQL), visit Vue CLI plugins in the Vue CLI docs.

Additional resources

- Vue docs
- Vue.js overview
- Should I install on Windows or Windows Subsystem for Linux (WSL)?
- Install Vue.js on WSL
- Install Nuxt.js
- Microsoft Learn online course: Take your first steps with Vue.js
- Try a Vue tutorial with VS Code

Get started with Nuxt.js on Windows

4/21/2021 • 4 minutes to read • Edit Online

A guide to help you install the Nuxt.js web framework and get up and running on Windows 10.

Nuxt.js is a framework for creating server-rendered JavaScript apps based on Vue.js, Node.js, Webpack and Babel.js. It was inspired by Next.js. It is basically a project boilerplate for Vue. Just like Next.js, it is crafted with attention to best practices and allows you to create "universal" web apps in a simple, consistent way, with hardly any configuration. These "universal" server-rendered web apps are also sometimes called “isomorphic”, meaning that code is shared between the client and server.

To learn more about Vue, see the Vue overview page.

Prerequisites

This guide assumes that you've already completed the steps to set up your Node.js development environment, including:

- Install Windows Subsystem for Linux (WSL), including a Linux distribution (like Ubuntu) and make sure it is running in WSL 2 mode. You can check this by opening PowerShell and entering: `wsl -l -v`
- Install Node.js on WSL 2: This includes a version manager, package manager, Visual Studio Code, and the Remote Development extension.

We recommend using the Windows Subsystem for Linux when working with NodeJS apps for better performance speed, system call compatibility, and for parody when running Linux servers or Docker containers.

IMPORTANT

Installing a Linux distribution with WSL will create a directory for storing files: `\\wsl\Ubuntu-20.04` (substitute Ubuntu-20.04 with whatever Linux distribution you're using). To open this directory in Windows File Explorer, open your WSL command line, select your home directory using `cd ~`, then enter the command `explorer.exe .` Be careful not to install NodeJS or store files that you will be working with on the mounted C drive (`/mnt/c/Users/yourname$`). Doing so will significantly slow down your install and build times.

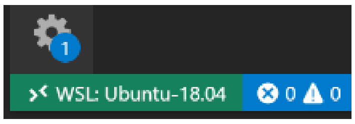
Install Nuxt.js

To install Nuxt.js, you will need to answer a series of questions about what sort of integrated server-side framework, UI framework, testing framework, mode, modules, and linter you would like to install:

1. Open a WSL command line (ie. Ubuntu).
2. Create a new project folder: `mkdir NuxtProjects` and enter that directory: `cd NuxtProjects`.
3. Install Nuxt.js and create a project (replacing 'my-nuxt-app' with whatever you'd like to call your app): `npm create nuxt-app my-nuxt-app`
4. The Nuxt.js installer will now ask you the following questions:
   - Project Name: my-nuxtjs-app
   - Project description: Description of my Nuxt.js app.
   - Author name: I use my GitHub alias.
   - Choose the package manager: Yarn or **Npm** - we use NPM for our examples.

- Choose UI framework: None, Ant Design Vue, Bootstrap Vue, etc. Let's choose **Vuetify** for this example, but the Vue Community created a nice summary comparing these UI frameworks to help you choose the best fit for your project.
- Choose custom server frameworks: None, AdonisJs, Express, Fastify, etc. Let's choose **None** for this example, but you can find a 2019-2020 server framework comparison on the Dev.to site.
- Choose Nuxt.js modules (use spacebar to select modules or just enter if you don't want any): Axios (for simplifying HTTP requests) or PWA support (for adding a service-worker, manifest.json file, etc). Let's not add a module for this example.
- Choose linting tools: **ESLint**, Prettier, Lint staged files. Let's choose **ESLint** (a tool for analyzing your code and warning you of potential errors).
- Choose a test framework: **None**, Jest, AVA. Let's choose **None** as we won't cover testing in this quickstart.
- Choose rendering mode: **Universal (SSR)** or Single Page App (SPA). Let's choose **Universal (SSR)** for our example, but the Nuxt.js docs point out some of the differences -- SSR requiring a Node.js server running to server-render your app and SPA for static hosting.
- Choose development tools: **jsconfig.json** (recommended for VS Code so Intellisense code completion works)

5. Once your project is created, `cd my-nuxtjs-app` to enter your Nuxt.js project directory, then enter `code .` to open the project in the VS Code WSL-Remote environment.

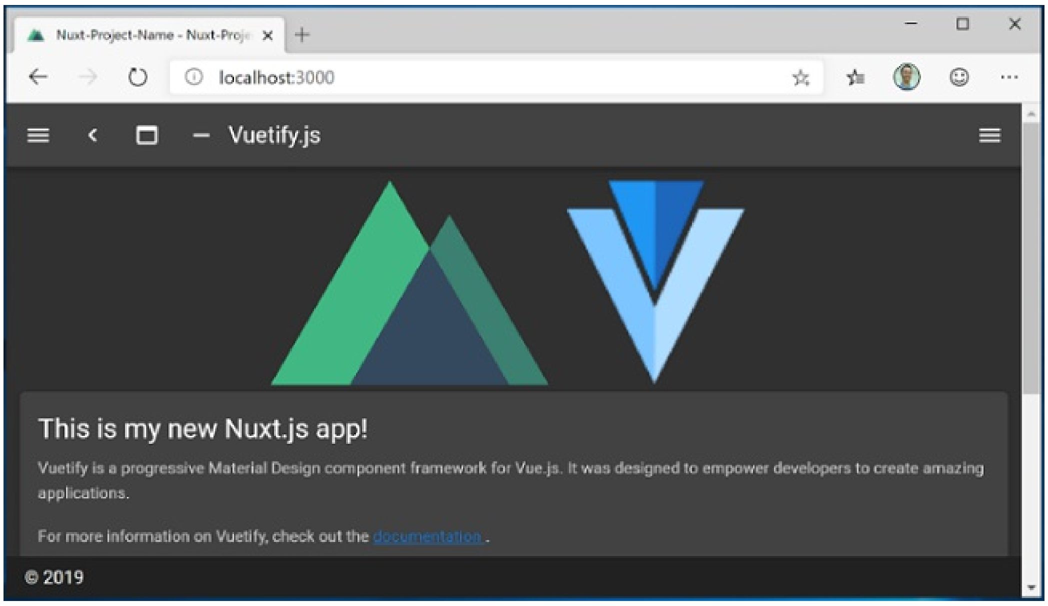


6. There are 3 commands you need to know once Nuxt.js is installed:

   - `npm run dev` for running a development instance with hot-reloading, file watching and task re-running.
   - `npm run build` for compiling your project.
   - `npm start` for starting your app in production mode.

   Open the WSL terminal integrated in VS Code (**View > Terminal**). Make sure that the terminal path is pointed to your project directory (ie. `~/NuxtProjects/my-nuxt-app$`). Then try running a development instance of your new Nuxt.js app using: `npm run dev`

7. The local development server will start (displaying some kind of cool progress bars for the client and server compiles). Once your project is done building, your terminal will display "Compiled successfully" along with how much time it took to compile. Point your web browser to http://localhost:3000 to open your new Nuxt.js app.



This is my new Nuxt.js app!

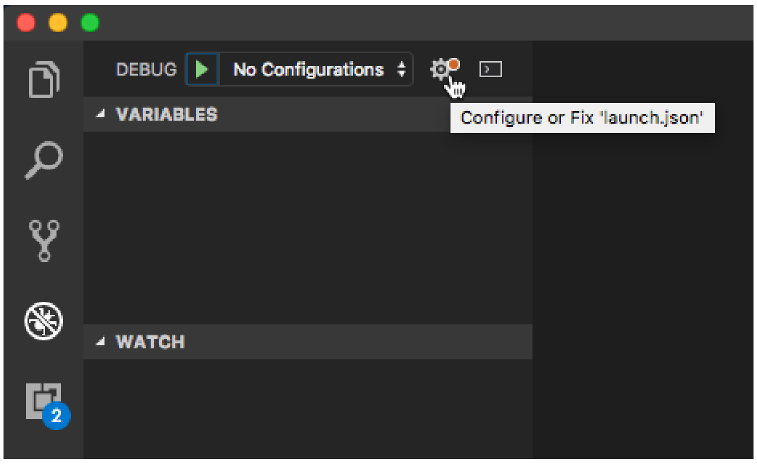
Vuetify is a progressive Material Design component framework for Vue.js. It was designed to empower developers to create amazing applications.

For more information on Vuetify, check out the documentation .

8. Open the `pages/index.vue` file in your VS Code editor. Find the page title `<v-card-title class="headline">Welcome to the Vuetify + Nuxt.js template</v-card-title>` and change it to `<v-card-title class="headline">This is my new Nuxt.js app!</v-card-title>`. With your web browser still open to localhost:3000, save your change and notice the hot-reloading feature automatically compile and update your change in the browser.

9. Let's see how Nuxt.js handles errors. Remove the `</v-card-title>` closing tag so that your title code now looks like this: `<v-card-title class="headline">This is my new Nuxt.js app!`. Save this change and notice that a compiling error will display in your browser, and in your terminal, letting your know that a closing tag for `<v-card-title>` is missing, along with the line numbers where the error can be found in your code. Replace the `</v-card-title>` closing tag, save, and the page will reload.

You can use VS Code's debugger with your Nuxt.js app by selecting the F5 key, or by going to **View > Debug** (Ctrl+Shift+D) and **View > Debug Console** (Ctrl+Shift+Y) in the menu bar. If you select the gear icon in the Debug window, a launch configuration (`launch.json`) file will be created for you to save debugging setup details. To learn more, see VS Code Debugging.



To learn more about Nuxt.js, see the Nuxt.js guide.

Tutorial: Vue.js for Beginners

5/13/2021 • 2 minutes to read • Edit Online

If you're brand new to using Vue.js, this guide will help you to get started with some basics.

- Try the Vue.js HelloWorld code sandbox
- Try using Node.js in Visual Studio Code

Prerequisites

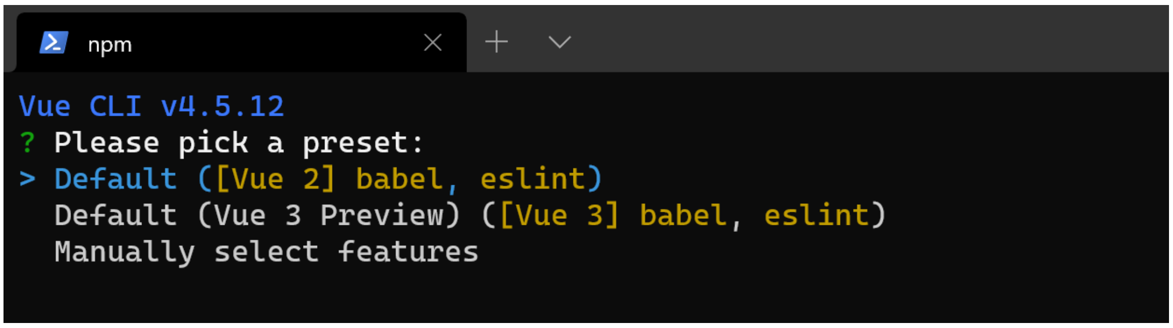
- You must first install Vue.js on Windows or on Windows Subsystem for Linux. Not sure which to use, generally we recommend beginners install on Windows to learn, but professionals install on WSL, see Should I install Node.js on Windows or Windows Subsystem for Linux.

Try NodeJS with Visual Studio Code

If you don't already have it, install VS Code. We recommend installing VS Code on Windows, regardless of whether you plan to use Vue on Windows or WSL.

1. Open your command line and create a new directory: `mkdir HelloVue`, then enter the directory: `cd HelloVue`
2. Install the Vue CLI: `npm install -g @vue/cli`
3. Create your Vue app: `vue create hello-vue-app`

   You'll need to choose whether to use Vue 2 or Vue 3 Preview, or manually select the features you want.

   ```
   Vue CLI v4.5.12
   ? Please pick a preset:
   > Default ([Vue 2] babel, eslint)
     Default (Vue 3 Preview) ([Vue 3] babel, eslint)
     Manually select features
   ```

4. Open the directory of your new hello-vue-app: `cd hello-vue-app`
5. Try running you new Vue app in your web browser: `npm run serve`

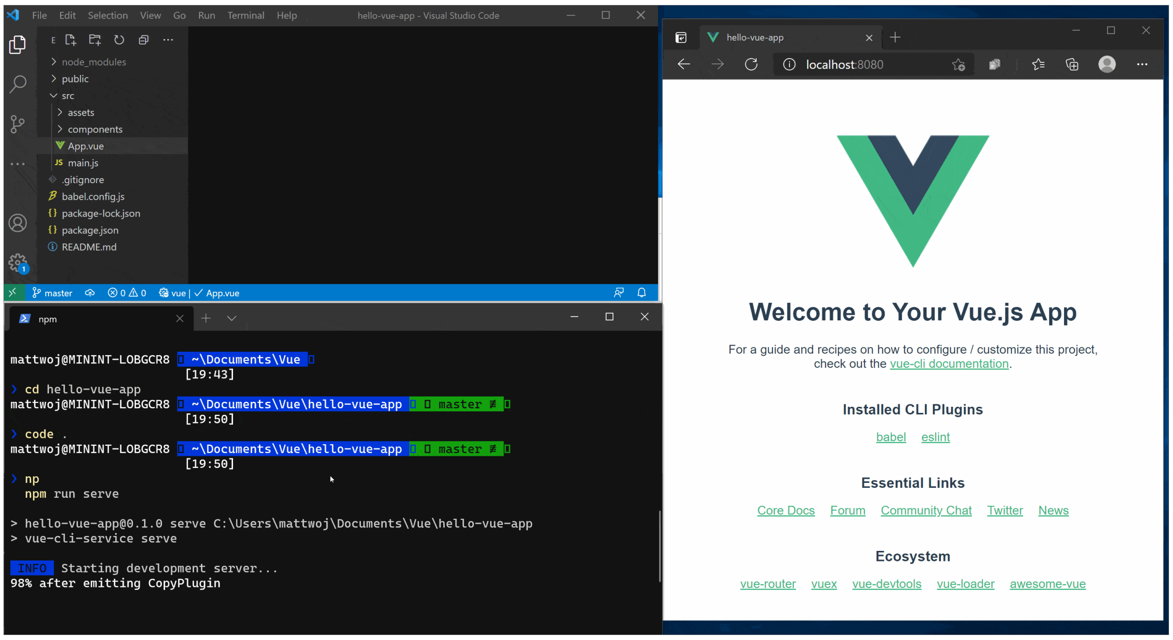
   You should see "Welcome to your Vue.js App" on http://localhost:8080 in your browser. You can press `Ctrl+C` to stop the vue-cli-service server.

   NOTE

   If using WSL (with Ubuntu or your favorite Linux distribution) for this tutorial, you'll need to make sure that you have the Remote - WSL Extension installed for the best experience running and editing your code with VS remote server.

6. Try updating the welcome message by opening your Vue app's source code in VS Code, enter: `code .`
7. VS Code will launch and display your Vue application in the File Explorer. Run your app in the terminal

again with `npm run serve` and have your web browser open to the localhost so that you can see the Vue page welcome page displayed. Find the `App.vue` file in VS Code. Try changing "Welcome to your Vue.js App" to "Welcome to the Jungle!". You will see your Vue app "hot reload" as soon as you save your change.



```
mattwoj@MININT-LOBGCR8  ~\Documents\Vue 
                          [19:43]
❯ cd hello-vue-app
mattwoj@MININT-LOBGCR8  ~\Documents\Vue\hello-vue-app   master ≢ 
                          [19:50]
❯ code .
mattwoj@MININT-LOBGCR8  ~\Documents\Vue\hello-vue-app   master ≢ 
                          [19:50]
❯ np
  npm run serve

> hello-vue-app@0.1.0 serve C:\Users\mattwoj\Documents\Vue\hello-vue-app
> vue-cli-service serve

 INFO  Starting development server...
98% after emitting CopyPlugin
```

Welcome to Your Vue.js App

For a guide and recipes on how to configure / customize this project, check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

Essential Links

Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

Additional resources

- Using Vue in Visual Studio Code: Find more about using Vue with VS Code, including the Vetur extension that provides Vue syntax highlighting, IntelliSense, debugging support, and more.
- Vue.js docs
- Vue comparison with other frameworks like React or Angular
- Vue.js overview
- - Microsoft Learn online course: Take your first steps with Vue.js

Get started using Python on Windows for beginners

5/5/2021 • 10 minutes to read • Edit Online

The following is a step-by-step guide for beginners interested in learning Python using Windows 10.

Set up your development environment

For beginners who are new to Python, we recommend you install Python from the Microsoft Store. Installing via the Microsoft Store uses the basic Python3 interpreter, but handles set up of your PATH settings for the current user (avoiding the need for admin access), in addition to providing automatic updates. This is especially helpful if you are in an educational environment or a part of an organization that restricts permissions or administrative access on your machine.

If you are using Python on Windows for **web development**, we recommend a different set up for your development environment. Rather than installing directly on Windows, we recommend installing and using Python via the Windows Subsystem for Linux. For help, see: Get started using Python for web development on Windows. If you're interested in automating common tasks on your operating system, see our guide: Get started using Python on Windows for scripting and automation. For some advanced scenarios (like needing to access/modify Python's installed files, make copies of binaries, or use Python DLLs directly), you may want to consider downloading a specific Python release directly from python.org or consider installing an alternative, such as Anaconda, Jython, PyPy, WinPython, IronPython, etc. We only recommend this if you are a more advanced Python programmer with a specific reason for choosing an alternative implementation.

Install Python

To install Python using the Microsoft Store:

1. Go to your **Start** menu (lower left Windows icon), type "Microsoft Store", select the link to open the store.
2. Once the store is open, select **Search** from the upper-right menu and enter "Python". Select which version of Python you would like to use from the results under Apps. We recommend using the most recent unless you have a reason not to (such as aligning with the version used on a pre-existing project that you plan to work on). Once you've determined which version you would like to install, select **Get**.
3. Once Python has completed the downloading and installation process, open Windows PowerShell using the **Start** menu (lower left Windows icon). Once PowerShell is open, enter `Python --version` to confirm that Python3 has installed on your machine.
4. The Microsoft Store installation of Python includes **pip**, the standard package manager. Pip allows you to install and manage additional packages that are not part of the Python standard library. To confirm that you also have pip available to install and manage packages, enter `pip --version`.

Install Visual Studio Code

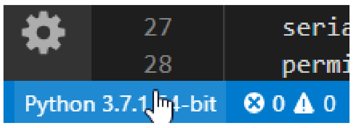
By using VS Code as your text editor / integrated development environment (IDE), you can take advantage of IntelliSense (a code completion aid), Linting (helps avoid making errors in your code), Debug support (helps you find errors in your code after you run it), Code snippets (templates for small reusable code blocks), and Unit testing (testing your code's interface with different types of input).

VS Code also contains a built-in terminal that enables you to open a Python command line with Windows Command prompt, PowerShell, or whatever you prefer, establishing a seamless workflow between your code editor and command line.

1. To install VS Code, download VS Code for Windows: https://code.visualstudio.com.

2. Once VS Code has been installed, you must also install the Python extension. To install the Python extension, you can select the VS Code Marketplace link or open VS Code and search for **Python** in the extensions menu (Ctrl+Shift+X).

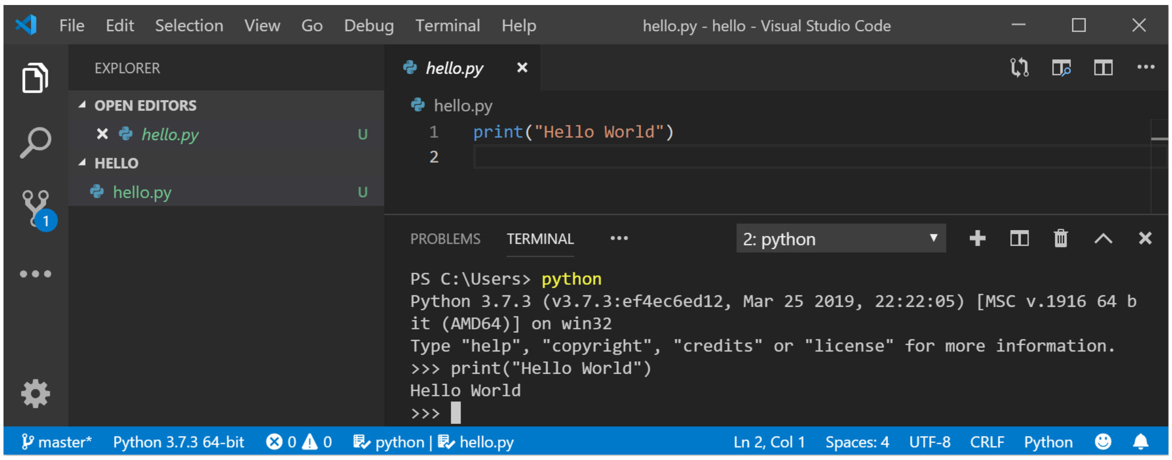
3. Python is an interpreted language, and in order to run Python code, you must tell VS Code which interpreter to use. We recommend using the most recent version of Python unless you have a specific reason for choosing something different. Once you've installed the Python extension, select a Python 3 interpreter by opening the **Command Palette** (Ctrl+Shift+P), start typing the command **Python: Select Interpreter** to search, then select the command. You can also use the **Select Python Environment** option on the bottom Status Bar if available (it may already show a selected interpreter). The command presents a list of available interpreters that VS Code can find automatically, including virtual environments. If you don't see the desired interpreter, see Configuring Python environments.



4. To open the terminal in VS Code, select **View** > **Terminal**, or alternatively use the shortcut **Ctrl+`** (using the backtick character). The default terminal is PowerShell.

5. Inside your VS Code terminal, open Python by simply entering the command: `python`

6. Try the Python interpreter out by entering: `print("Hello World")`. Python will return your statement "Hello World".


File Edit Selection View Go Debug Terminal Help
hello.py - hello - Visual Studio Code
EXPLORER
OPEN EDITORS
hello.py U
HELLO
hello.py U
PROBLEMS TERMINAL
2: python
master* Python 3.7.3 64-bit 0 0 python | hello.py Ln 2, Col 1 Spaces: 4 UTF-8 CRLF Python

```python
print("Hello World")
```

```
PS C:\Users> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

Install Git (optional)

If you plan to collaborate with others on your Python code, 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. You first need to install Git to power the Source Control panel.

1. Download and install Git for Windows from the git-scm website.

2. An Install Wizard is included that will ask you a series of questions about settings for your Git installation. We recommend using all of the default settings, unless you have a specific reason for changing something.

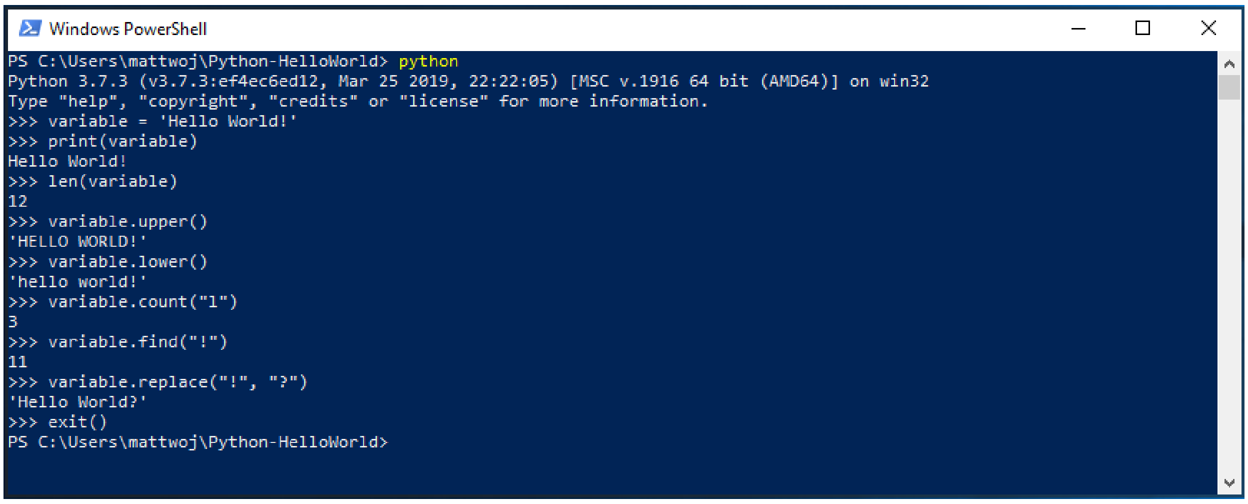
3. If you've never worked with Git before, GitHub Guides can help you get started.

Hello World tutorial for some Python basics

Python, according to its creator Guido van Rossum, is a “high-level programming language, and its core design philosophy is all about code readability and a syntax which allows programmers to express concepts in a few lines of code.”

Python is an interpreted language. In contrast to compiled languages, in which the code you write needs to be translated into machine code in order to be run by your computer's processor, Python code is passed straight to an interpreter and run directly. You just type in your code and run it. Let's try it!

1. With your PowerShell command line open, enter `python` to run the Python 3 interpreter. (Some instructions prefer to use the command `py` or `python3`, these should also work). You will know that you're successful because a >>> prompt with three greater-than symbols will display.
2. There are several built-in methods that allow you to make modifications to strings in Python. Create a variable, with: `variable = 'Hello World!'`. Press Enter for a new line.
3. Print your variable with: `print(variable)`. This will display the text "Hello World!".
4. Find out the length, how many characters are used, of your string variable with: `len(variable)`. This will display that there are 12 characters used. (Note that the blank space it counted as a character in the total length.)
5. Convert your string variable to upper-case letters: `variable.upper()`. Now convert your string variable to lower-case letters: `variable.lower()`.
6. Count how many times the letter "l" is used in your string variable: `variable.count("l")`.
7. Search for a specific character in your string variable, let's find the exclamation point, with: `variable.find("!")`. This will display that the exclamation point is found in the 11th position character of the string.
8. Replace the exclamation point with a question mark: `variable.replace("!", "?")`.
9. To exit Python, you can enter `exit()`, `quit()`, or select Ctrl-Z.



```
PS C:\Users\mattwoj\Python-HelloWorld> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> variable = 'Hello World!'
>>> print(variable)
Hello World!
>>> len(variable)
12
>>> variable.upper()
'HELLO WORLD!'
>>> variable.lower()
'hello world!'
>>> variable.count("l")
3
>>> variable.find("!")
11
>>> variable.replace("!", "?")
'Hello World?'
>>> exit()
PS C:\Users\mattwoj\Python-HelloWorld>
```

Hope you had fun using some of Python's built-in string modification methods. Now try creating a Python program file and running it with VS Code.

Hello World tutorial for using Python with VS Code

The VS Code team has put together a great Getting Started with Python tutorial walking through how to create a Hello World program with Python, run the program file, configure and run the debugger, and install packages like *matplotlib* and *numpy* to create a graphical plot inside a virtual environment.

1. Open PowerShell and create an empty folder called "hello", navigate into this folder, and open it in VS

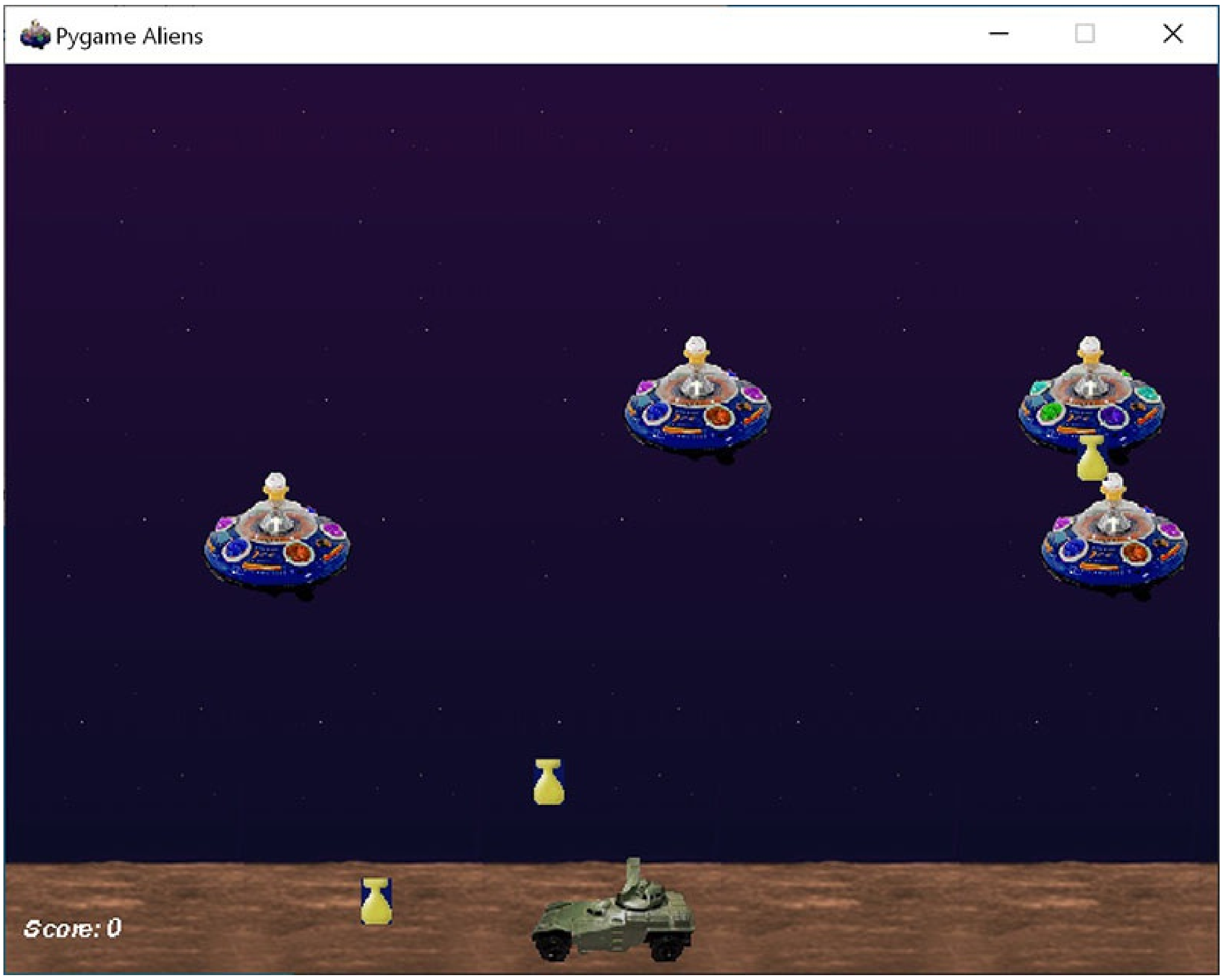
Code:

```
mkdir hello
cd hello
code .
```

2. Once VS Code opens, displaying your new *hello* folder in the left-side **Explorer** window, open a command line window in the bottom panel of VS Code by pressing **Ctrl+`** (using the backtick character) or selecting **View** > **Terminal**. By starting VS Code in a folder, that folder becomes your "workspace". VS Code stores settings that are specific to that workspace in .vscode/settings.json, which are separate from user settings that are stored globally.

3. Continue the tutorial in the VS Code docs: Create a Python Hello World source code file.

Create a simple game with Pygame



Pygame is a popular Python package for writing games - encouraging students to learn programming while creating something fun. Pygame displays graphics in a new window, and so it will not work under the command-line-only approach of WSL. However, if you installed Python via the Microsoft Store as detailed in this tutorial, it will work fine.

1. Once you have Python installed, install pygame from the command line (or the terminal from within VS Code) by typing `python -m pip install -U pygame --user`.

2. Test the installation by running a sample game : `python -m pygame.examples.aliens`

3. All being well, the game will open a window. Close the window when you are done playing.

Here's how to start writing your own game.

1. Open PowerShell (or Windows Command Prompt) and create an empty folder called "bounce". Navigate to this folder and create a file named "bounce.py". Open the folder in VS Code:

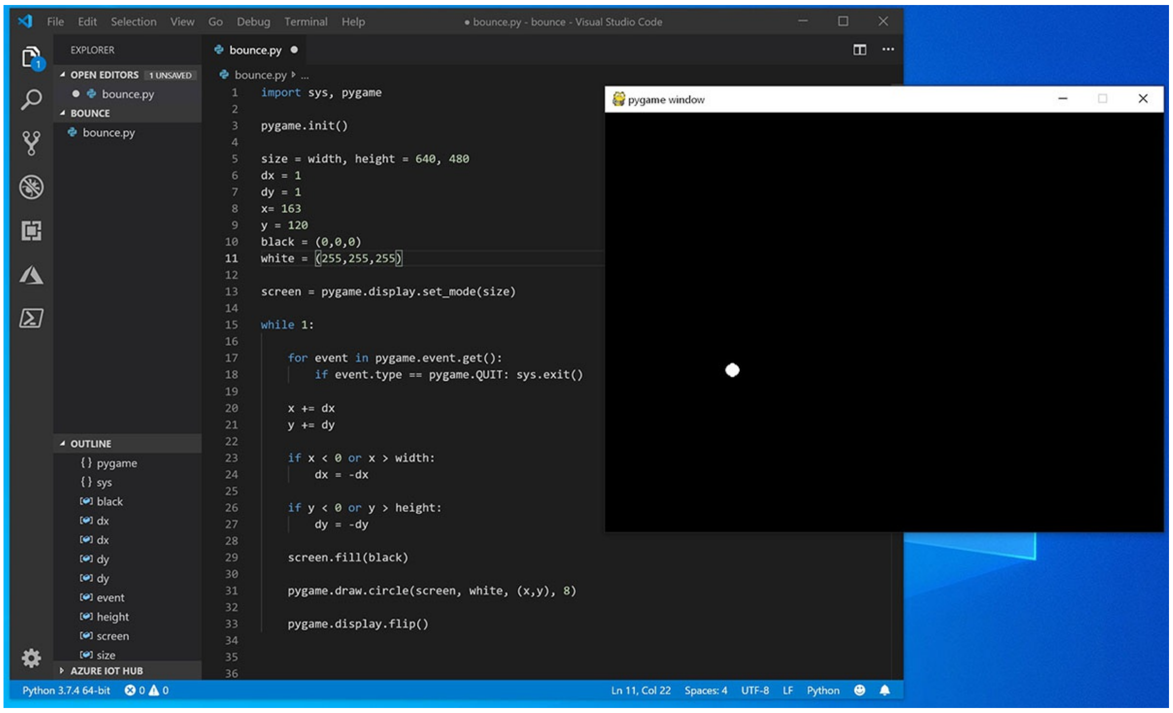
```
mkdir bounce
cd bounce
new-item bounce.py
code .
```

2. Using VS Code, enter the following Python code (or copy and paste it):

```
import sys, pygame

pygame.init()

size = width, height = 640, 480
dx = 1
dy = 1
x= 163
y = 120
black = (0,0,0)
white = (255,255,255)

screen = pygame.display.set_mode(size)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    x += dx
    y += dy

    if x < 0 or x > width:
        dx = -dx

    if y < 0 or y > height:
        dy = -dy

    screen.fill(black)

    pygame.draw.circle(screen, white, (x,y), 8)

    pygame.display.flip()
```

3. Save it as: `bounce.py`.

4. From the PowerShell terminal, run it by entering: `python bounce.py`.



```python
import sys, pygame

pygame.init()

size = width, height = 640, 480
dx = 1
dy = 1
x= 163
y = 120
black = (0,0,0)
white = (255,255,255)

screen = pygame.display.set_mode(size)

while 1:

    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()

    x += dx
    y += dy

    if x < 0 or x > width:
        dx = -dx

    if y < 0 or y > height:
        dy = -dy

    screen.fill(black)

    pygame.draw.circle(screen, white, (x,y), 8)

    pygame.display.flip()
```

Try adjusting some of the numbers to see what effect they have on your bouncing ball.

Read more about writing games with pygame at pygame.org.

Resources for continued learning

We recommend the following resources to support you in continuing to learn about Python development on Windows.

Online courses for learning Python

- Introduction to Python on Microsoft Learn: Try the interactive Microsoft Learn platform and earn experience points for completing this module covering the basics on how to write basic Python code, declare variables, and work with console input and output. The interactive sandbox environment makes this a great place to start for folks who don't have their Python development environment set up yet.
- Python on Pluralsight: 8 Courses, 29 Hours: The Python learning path on Pluralsight offers online courses covering a variety of topics related to Python, including a tool to measure your skill and find your gaps.
- LearnPython.org Tutorials: Get started on learning Python without needing to install or set anything up with these free interactive Python tutorials from the folks at DataCamp.
- The Python.org Tutorials: Introduces the reader informally to the basic concepts and features of the Python language and system.
- Learning Python on Lynda.com: A basic introduction to Python.

Working with Python in VS Code

- Editing Python in VS Code: Learn more about how to take advantage of VS Code's autocomplete and IntelliSense support for Python, including how to customize their behavior... or just turn them off.
- Linting Python: Linting is the process of running a program that will analyse code for potential errors. Learn about the different forms of linting support VS Code provides for Python and how to set it up.
- Debugging Python: Debugging is the process of identifying and removing errors from a computer program. This article covers how to initialize and configure debugging for Python with VS Code, how to set and validate breakpoints, attach a local script, perform debugging for different app types or on a remote computer, and some basic troubleshooting.

- Unit testing Python: Covers some background explaining what unit testing means, an example walkthrough, enabling a test framework, creating and running your tests, debugging tests, and test configuration settings.

Get started using Python for web development on Windows

3/5/2021 • 17 minutes to read • Edit Online

The following is a step-by-step guide to get you started using Python for web development on Windows, using the Windows Subsystem for Linux (WSL).

Set up your development environment

We recommend installing Python on WSL when building web applications. Many of the tutorials and instructions for Python web development are written for Linux users and use Linux-based packaging and installation tools. Most web apps are also deployed on Linux, so this will ensure you have consistency between your development and production environments.

If you are using Python for something other than web development, we recommend you install Python directly on Windows 10 using the Microsoft Store. WSL does not support GUI desktops or applications (like PyGame, Gnome, KDE, etc). Install and use Python directly on Windows for these cases. If you're new to Python, see our guide: Get started using Python on Windows for beginners. If you're interested in automating common tasks on your operating system, see our guide: Get started using Python on Windows for scripting and automation. For some advanced scenarios, you may want to consider downloading a specific Python release directly from python.org or consider installing an alternative, such as Anaconda, Jython, PyPy, WinPython, IronPython, etc. We only recommend this if you are a more advanced Python programmer with a specific reason for choosing an alternative implementation.

Install Windows Subsystem for Linux

WSL lets you run a GNU/Linux command line environment integrated directly with Windows and your favorite tools, like Visual Studio Code, Outlook, etc.

To enable and install WSL (or WSL 2 depending on your needs), follow the steps in the WSL install documentation. These steps will include choosing a Linux distribution (for example, Ubuntu).

Once you have installed WSL and a Linux distribution, open the Linux distribution (it can be found in your Windows start menu) and check the version and codename using the command: `lsb_release -dc`.

We recommend updating your Linux distribution regularly, including immediately after you install, to ensure you have the most recent packages. Windows doesn't automatically handle this update. To update your distribution, use the command: `sudo apt update && sudo apt upgrade`.

TIP

Consider installing the new Windows Terminal from the Microsoft Store to enable 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 set up custom themes (color schemes, font styles and sizes, background image/blur/transparency). Learn more.

Set up Visual Studio Code

Take advantage of IntelliSense, Linting, Debug support, Code snippets, and Unit testing by using VS Code. VS Code integrates nicely with the Windows Subsystem for Linux, providing a built-in terminal to establish a seamless workflow between your code editor and your command line, in addition to supporting Git for version control with common Git commands (add, commit, push, pull) built right into the UI.

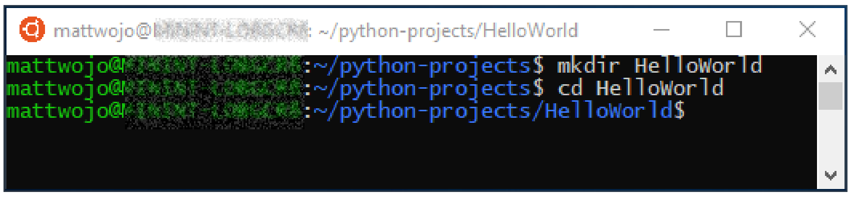
1. Download and install VS Code for Windows. VS Code is also available for Linux, but Windows Subsystem for Linux does not support GUI apps, so we need to install it on Windows. Not to worry, you'll still be able to integrate with your Linux command line and tools using the Remote - WSL Extension.
2. Install the Remote - WSL Extension on VS Code. This allows you to use WSL as your integrated development environment and will handle compatibility and pathing for you. Learn more.

> **IMPORTANT**
>
> If you already have VS Code installed, you need to ensure that you have the 1.35 May release or later in order to install the Remote - WSL Extension. 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-server/extensions.

Create a new project

Let's create a new project directory on our Linux (Ubuntu) file system that we will then work on with Linux apps and tools using VS Code.

1. Close VS Code and open Ubuntu 18.04 (your WSL command line) by going to your **Start** menu (lower left Windows icon) and typing: "Ubuntu 18.04".
2. In your Ubuntu command line, navigate to where you want to put your project, and create a directory for it: `mkdir HelloWorld`.



```
mattwojo@[redacted]: ~/python-projects/HelloWorld
mattwojo@[redacted]:~/python-projects$ mkdir HelloWorld
mattwojo@[redacted]:~/python-projects$ cd HelloWorld
mattwojo@[redacted]:~/python-projects/HelloWorld$
```

> **TIP**
>
> An important thing to remember when using Windows Subsystem for Linux (WSL) is that **you are now working between two different file systems**: 1) your Windows file system, and 2) your Linux file system (WSL), which is Ubuntu for our example. You will need to pay attention to where you install packages and store files. You can install one version of a tool or package in the Windows file system and a completely different version in the Linux file system. Updating the tool in the Windows file system will have no effect on the tool in the Linux file system, and vice-versa. WSL mounts the fixed drives on your computer under the `/mnt/<drive>` folder in your Linux distribution. For example, your Windows C: drive is mounted under `/mnt/c/`. You can access your Windows files from the Ubuntu terminal and use Linux apps and tools on those files and vice-versa. We recommend working in the Linux file system for Python web development given that much of the web tooling is originally written for Linux and deployed in a Linux production environment. It also avoids mixing file system semantics (like Windows being case-insensitive regarding file names). That said, WSL now supports jumping between the Linux and Windows files systems, so you can host your files on either one. Learn more.

Install Python, pip, and venv

Ubuntu 18.04 LTS comes with Python 3.6 already installed, but it does not come with some of the modules that you may expect to get with other Python installations. We will still need to install **pip**, the standard package manager for Python, and **venv**, the standard module used to create and manage lightweight virtual environments.

1. Confirm that Python3 is already installed by opening your Ubuntu terminal and entering: `python3 --version`. This should return your Python version number. If you need to update your version of Python, first update your Ubuntu version by entering: `sudo apt update && sudo apt upgrade`, then update Python using `sudo apt upgrade python3`.

2. Install **pip** by entering: `sudo apt install python3-pip`. Pip allows you to install and manage additional packages that are not part of the Python standard library.

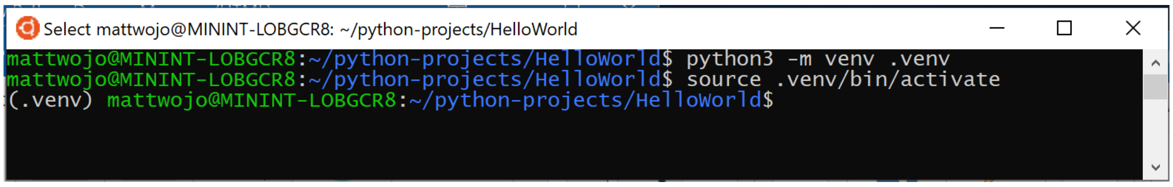
3. Install **venv** by entering: `sudo apt install python3-venv`.

Create a virtual environment

Using virtual environments is a recommended best practice for Python development projects. By creating a virtual environment, you can isolate your project tools and avoid versioning conflicts with tools for your other projects. For example, you may be maintaining an an older web project that requires the Django 1.2 web framework, but then an exciting new project comes along using Django 2.2. If you update Django globally, outside of a virtual environment, you could run into some versioning issues later on. In addition to preventing accidental versioning conflicts, virtual environments let you install and manage packages without administrative privileges.

1. Open your terminal and, inside your *HelloWorld* project folder, use the following command to create a virtual environment named **.venv**: `python3 -m venv .venv`.

2. To activate the virtual environment, enter: `source .venv/bin/activate`. If it worked, you should see **(.venv)** before the command prompt. You now have a self-contained environment ready for writing code and installing packages. When you're finished with your virtual environment, enter the following command to deactivate it: `deactivate`.


Select mattwojo@MININT-LOBGCR8: ~/python-projects/HelloWorld

```
mattwojo@MININT-LOBGCR8:~/python-projects/HelloWorld$ python3 -m venv .venv
mattwojo@MININT-LOBGCR8:~/python-projects/HelloWorld$ source .venv/bin/activate
(.venv) mattwojo@MININT-LOBGCR8:~/python-projects/HelloWorld$
```

TIP

We recommend creating the virtual environment inside the directory in which you plan to have your project. Since each project should have its own separate directory, each will have its own virtual environment, so there is not a need for unique naming. Our suggestion is to use the name **.venv** to follow the Python convention. Some tools (like pipenv) also default to this name if you install into your project directory. You don't want to use **.env** as that conflicts with environment variable definition files. We generally do not recommend non-dot-leading names, as you don't need `ls` constantly reminding you that the directory exists. We also recommend adding **.venv** to your .gitignore file. (Here is GitHub's default gitignore template for Python for reference.) For more information about working with virtual environments in VS Code, see Using Python environments in VS Code.

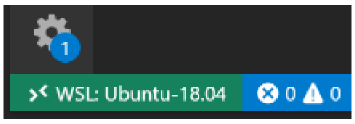
Open a WSL - Remote window

VS Code uses the Remote - WSL Extension (installed previously) to treat your Linux subsystem as a remote server. This allows you to use WSL as your integrated development environment. Learn more.

1. Open your project folder in VS Code from your Ubuntu terminal by entering: `code .` (the "." tells VS Code to open the current folder).

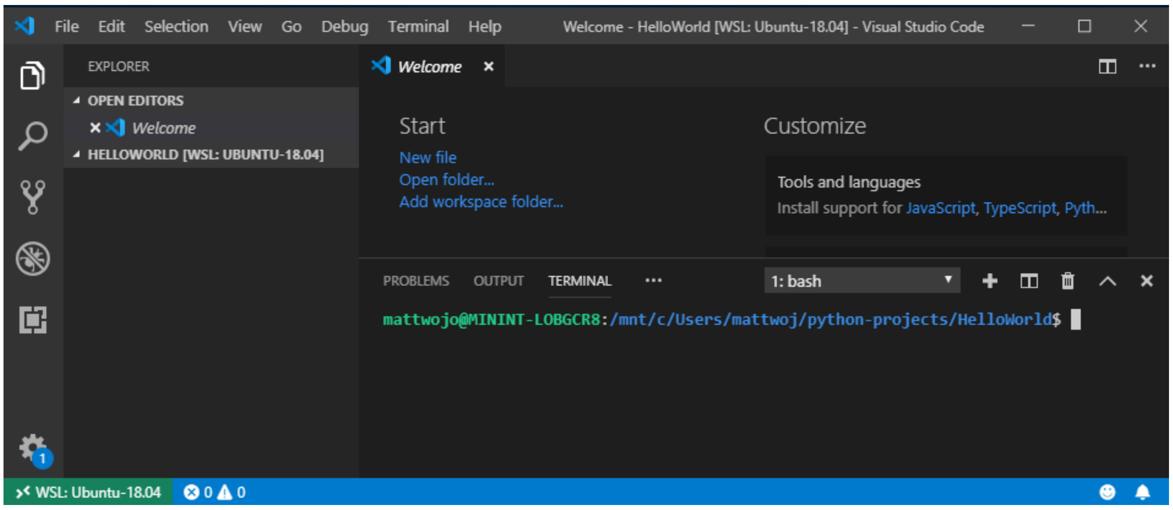
2. A Security Alert will pop-up from Windows Defender, select "Allow access". Once VS Code opens, you should see the Remote Connection Host indicator, in the bottom-left corner, letting you know that you are

editing on **WSL: Ubuntu-18.04**.



3. Close your Ubuntu terminal. Moving forward we will use the WSL terminal integrated into VS Code.

4. Open the WSL terminal in VS Code by pressing **Ctrl+`** (using the backtick character) or selecting **View** > **Terminal**. This will open a bash (WSL) command-line opened to the project folder path that you created in your Ubuntu terminal.


File Edit Selection View Go Debug Terminal Help
Welcome - HelloWorld [WSL: Ubuntu-18.04] - Visual Studio Code
EXPLORER
OPEN EDITORS
Welcome
HELLOWORLD [WSL: UBUNTU-18.04]
Welcome
Start
New file
Open folder...
Add workspace folder...
Customize
Tools and languages
Install support for JavaScript, TypeScript, Pyth...
PROBLEMS
OUTPUT
TERMINAL
1: bash
WSL: Ubuntu-18.04

```
mattwojo@MININT-LOBGCR8:/mnt/c/Users/mattwoj/python-projects/HelloWorld$
```

Install the Microsoft Python extension

You will need to install any VS Code extensions for your Remote - WSL. Extensions already installed locally on VS Code will not automatically be available. Learn more.

1. Open the VS Code Extensions window by entering **Ctrl+Shift+X** (or use the menu to navigate to **View** > **Extensions**).

2. In the top **Search Extensions in Marketplace** box, enter: **Python**.

3. Find the **Python (ms-python.python) by Microsoft** extension and select the green **Install** button.

4. Once the extension is finished installing, you will need to select the blue **Reload Required** button. This will reload VS Code and display a **WSL: UBUNTU-18.04 - Installed** section in your VS Code Extensions window showing that you've installed the Python extension.

Run a simple Python program

Python is an interpreted language and supports different types of interpretors (Python2, Anaconda, PyPy, etc). VS Code should default to the interpreter associated with your project. If you have a reason to change it, select the interpreter currently displayed in blue bar on the bottom of your VS Code window or open the **Command Palette** (Ctrl+Shift+P) and enter the command **Python: Select Interpreter**. This will display a list of the Python interpreters that you currently have installed. Learn more about configuring Python environments.

Let's create and run a simple Python program as a test and ensure that we have the correct Python interpreter selected.

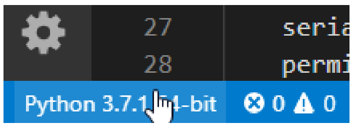
1. Open the VS Code File Explorer window by entering **Ctrl+Shift+E** (or use the menu to navigate to **View** > **Explorer**).

2. If it's not already open, open your integrated WSL terminal by entering **Ctrl+Shift+`** and ensure that

your **HelloWorld** python project folder is selected.

3. Create a python file by entering: `touch test.py`. You should see the file you just created appear in your Explorer window under the .venv and .vscode folders already in your project directory.

4. Select the **test.py** file that you just created in your Explorer window to open it in VS Code. Because the .py in our file name tells VS Code that this is a Python file, the Python extension you loaded previously will automatically choose and load a Python interpreter that you will see displayed on the bottom of your VS Code window.



5. Paste this Python code into your test.py file and then save the file (Ctrl+S):

```
print("Hello World")
```

6. To run the Python "Hello World" program that we just created, select the **test.py** file in the VS Code Explorer window, then right-click the file to display a menu of options. Select **Run Python File in Terminal**. Alternatively, in your integrated WSL terminal window, enter: `python test.py` to run your "Hello World" program. The Python interpreter will print "Hello World" in your terminal window.

Congratulations. You're all set up to create and run Python programs! Now let's try creating a Hello World app with two of the most popular Python web frameworks: Flask and Django.

Hello World tutorial for Flask

Flask is a web application framework for Python. In this brief tutorial, you'll create a small "Hello World" Flask app using VS Code and WSL.

1. Open Ubuntu 18.04 (your WSL command line) by going to your **Start** menu (lower left Windows icon) and typing: "Ubuntu 18.04".

2. Create a directory for your project: `mkdir HelloWorld-Flask`, then `cd HelloWorld-Flask` to enter the directory.

3. Create a virtual environment to install your project tools: `python3 -m venv .venv`

4. Open your **HelloWorld-Flask** project in VS Code by entering the command: `code .`

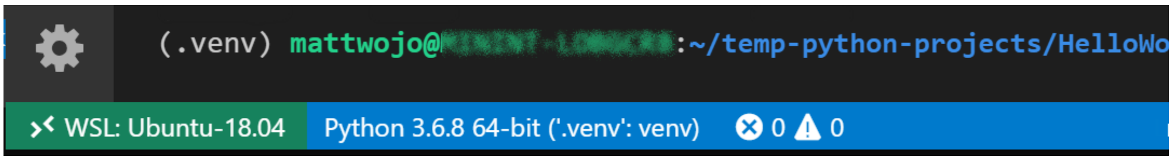
5. Inside VS Code, open your integrated WSL terminal (aka Bash) by entering **Ctrl+Shift+`** (your **HelloWorld-Flask** project folder should already be selected). *Close your Ubuntu command line as we will be working in the WSL terminal integrated with VS Code moving forward.*

6. Activate the virtual environment that you created in step #3 using your Bash terminal in VS Code: `source .venv/bin/activate`. If it worked, you should see (.venv) before the command prompt.

7. Install Flask in the virtual environment by entering: `python3 -m pip install flask`. Verify that it's installed by entering: `python3 -m flask --version`.

8. Create a new file for your Python code: `touch app.py`

9. Open your **app.py** file in VS Code's File Explorer (`Ctrl+Shift+E`, then select your app.py file). This will activate the Python Extension to choose an interpreter. It should default to **Python 3.6.8 64-bit ('.venv': venv)**. Notice that it also detected your virtual environment.



(.venv) mattwojo@[redacted]:~/temp-python-projects/HelloWo

WSL: Ubuntu-18.04 Python 3.6.8 64-bit ('.venv': venv) ⊗ 0 ⚠ 0

10. In **app.py**, add code to import Flask and create an instance of the Flask object:

```
from flask import Flask
app = Flask(__name__)
```

11. Also in **app.py**, add a function that returns content, in this case a simple string. Use Flask's **app.route** decorator to map the URL route "/" to that function:

```
@app.route("/")
def home():
    return "Hello World! I'm using Flask."
```

TIP

You can use multiple decorators on the same function, one per line, depending on how many different routes you want to map to the same function.

12. Save the **app.py** file (**Ctrl+S**).

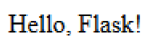
13. In the terminal, run the app by entering the following command:

```
python3 -m flask run
```

This runs the Flask development server. The development server looks for **app.py** by default. When you run Flask, you should see output similar to the following:

```
(env) user@USER:/mnt/c/Projects/HelloWorld$ python3 -m flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

14. Open your default web browser to the rendered page, **Ctrl+Click** the http://127.0.0.1:5000/ URL in the terminal. You should see the following message in your browser:



Hello, Flask!

15. Observe that when you visit a URL like "/", a message appears in the debug terminal showing the HTTP request:

```
127.0.0.1 - - [19/Jun/2019 13:36:56] "GET / HTTP/1.1" 200 -
```

16. Stop the app by using **Ctrl+C** in the terminal.

TIP

If you want to use a different filename than **app.py**, such as **program.py**, define an environment variable named **FLASK_APP** and set its value to your chosen file. Flask's development server then uses the value of **FLASK_APP** instead of the default file **app.py**. For more information, see Flask's Command Line Interface documentation.

Congratulations, you've created a Flask web application using Visual Studio Code and Windows Subsystem for Linux! For a more in-depth tutorial using VS Code and Flask, see Flask Tutorial in Visual Studio Code.

Hello World tutorial for Django

Django is a web application framework for Python. In this brief tutorial, you'll create a small "Hello World" Django app using VS Code and WSL.

1. Open Ubuntu 18.04 (your WSL command line) by going to your **Start** menu (lower left Windows icon) and typing: "Ubuntu 18.04".
2. Create a directory for your project: `mkdir HelloWorld-Django`, then `cd HelloWorld-Django` to enter the directory.
3. Create a virtual environment to install your project tools: `python3 -m venv .venv`
4. Open your **HelloWorld-DJango** project in VS Code by entering the command: `code .`
5. Inside VS Code, open your integrated WSL terminal (aka Bash) by entering **Ctrl+Shift+`** (your **HelloWorld-Django** project folder should already be selected). *Close your Ubuntu command line as we will be working in the WSL terminal integrated with VS Code moving forward.*
6. Activate the virtual environment that you created in step #3 using your Bash terminal in VS Code: `source .venv/bin/activate`. If it worked, you should see (.venv) before the command prompt.
7. Install Django in the virtual environment with the command: `python3 -m pip install django`. Verify that it's installed by entering: `python3 -m django --version`.
8. Next, run the following command to create the Django project:

   ```
   django-admin startproject web_project .
   ```

   The `startproject` command assumes (by use of `.` at the end) that the current folder is your project folder, and creates the following within it:

   - `manage.py`: The Django command-line administrative utility for the project. You run administrative commands for the project using `python manage.py <command> [options]`.
   - A subfolder named `web_project`, which contains the following files:
     - `__init__.py`: an empty file that tells Python that this folder is a Python package.
     - `wsgi.py`: an entry point for WSGI-compatible web servers to serve your project. You typically leave this file as-is as it provides the hooks for production web servers.
     - `settings.py`: contains settings for Django project, which you modify in the course of developing a web app.
     - `urls.py`: contains a table of contents for the Django project, which you also modify in the course of development.
9. To verify the Django project, start Django's development server using the command `python3 manage.py runserver`. The server runs on the default port 8000, and you should see output like

the following output in the terminal window:

```
Performing system checks...

System check identified no issues (0 silenced).

June 20, 2019 - 22:57:59
Django version 2.2.2, using settings 'web_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

When you run the server the first time, it creates a default SQLite database in the file `db.sqlite3`, which is intended for development purposes, but can be used in production for low-volume web apps. Also, Django's built-in web server is intended *only* for local development purposes. When you deploy to a web host, however, Django uses the host's web server instead. The `wsgi.py` module in the Django project takes care of hooking into the production servers.

If you want to use a different port than the default 8000, specify the port number on the command line, such as `python3 manage.py runserver 5000`.

10. `Ctrl+click` the `http://127.0.0.1:8000/` URL in the terminal output window to open your default browser to that address. If Django is installed correctly and the project is valid, you'll see a default page. The VS Code terminal output window also shows the server log.

11. When you're done, close the browser window and stop the server in VS Code using `Ctrl+C` as indicated in the terminal output window.

12. Now, to create a Django app, run the administrative utility's `startapp` command in your project folder (where `manage.py` resides):

```
python3 manage.py startapp hello
```

The command creates a folder called `hello` that contains a number of code files and one subfolder. Of these, you frequently work with `views.py` (that contains the functions that define pages in your web app) and `models.py` (that contains classes defining your data objects). The `migrations` folder is used by Django's administrative utility to manage database versions as discussed later in this tutorial. There are also the files `apps.py` (app configuration), `admin.py` (for creating an administrative interface), and `tests.py` (for tests), which are not covered here.

13. Modify `hello/views.py` to match the following code, which creates a single view for the app's home page:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello, Django!")
```

14. Create a file, `hello/urls.py`, with the contents below. The `urls.py` file is where you specify patterns to route different URLs to their appropriate views. The code below contains one route to map root URL of the app (`""`) to the `views.home` function that you just added to `hello/views.py`:

```
from django.urls import path
from hello import views

urlpatterns = [
    path("", views.home, name="home"),
]
```

15. The `web_project` folder also contains a `urls.py` file, which is where URL routing is actually handled. Open `web_project/urls.py` and modify it to match the following code (you can retain the instructive comments if you like). This code pulls in the app's `hello/urls.py` using `django.urls.include`, which keeps the app's routes contained within the app. This separation is helpful when a project contains multiple apps.

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path("", include("hello.urls")),
]
```

16. Save all modified files.

17. In the VS Code Terminal, run the development server with `python3 manage.py runserver` and open a browser to `http://127.0.0.1:8000/` to see a page that renders "Hello, Django".

Congratulations, you've created a Django web application using VS Code and Windows Subsystem for Linux! For a more in-depth tutorial using VS Code and Django, see Django Tutorial in Visual Studio Code.

Additional resources

- Python Tutorial with VS Code: An intro tutorial to VS Code as a Python environment, primarily how to edit, run, and debug code.
- Git support in VS Code: Learn how to use Git version control basics in VS Code.
- Learn about updates coming soon with WSL 2!: This new version changes how Linux distributions interact with Windows, increasing file system performance and adding full system call compatibility.
- Working with multiple Linux distributions on Windows: Learn how to manage multiple different Linux distributions on your Windows machine.

Get started using Python on Windows for scripting and automation

5/25/2021 • 9 minutes to read • Edit Online

The following is a step-by-step guide for setting up your developer environment and getting you started using Python for scripting and automating file system operations on Windows.

NOTE

This article will cover setting up your environment to use some of the helpful libraries in Python that can automate tasks across platforms, like searching your file system, accessing the internet, parsing file types, etc., from a Windows-centered approach. For Windows-specific operations, check out ctypes, a C-compatible foreign function library for Python, winreg, functions exposing the Windows registry API to Python, and Python/WinRT, enabling access Windows Runtime APIs from Python.

Set up your development environment

When using Python to write scripts that perform file system operations, we recommend you install Python from the Microsoft Store. Installing via the Microsoft Store uses the basic Python3 interpreter, but handles set up of your PATH settings for the current user (avoiding the need for admin access), in addition to providing automatic updates.

If you are using Python for **web development** on Windows, we recommend a different setup using the Windows Subsystem for Linux. Find a walkthrough in our guide: Get started using Python for web development on Windows. If you're brand new to Python, try our guide: Get started using Python on Windows for beginners. For some advanced scenarios (like needing to access/modify Python's installed files, make copies of binaries, or use Python DLLs directly), you may want to consider downloading a specific Python release directly from python.org or consider installing an alternative, such as Anaconda, Jython, PyPy, WinPython, IronPython, etc. We only recommend this if you are a more advanced Python programmer with a specific reason for choosing an alternative implementation.

Install Python

To install Python using the Microsoft Store:

1. Go to your **Start** menu (lower left Windows icon), type "Microsoft Store", select the link to open the store.
2. Once the store is open, select **Search** from the upper-right menu and enter "Python". Select which version of Python you would like to use from the results under Apps. We recommend using the most recent unless you have a reason not to (such as aligning with the version used on a pre-existing project that you plan to work on). Once you've determined which version you would like to install, select **Get**.
3. Once Python has completed the downloading and installation process, open Windows PowerShell using the **Start** menu (lower left Windows icon). Once PowerShell is open, enter `Python --version` to confirm that Python3 has been installed on your machine.
4. The Microsoft Store installation of Python includes **pip**, the standard package manager. Pip allows you to install and manage additional packages that are not part of the Python standard library. To confirm that you also have pip available to install and manage packages, enter `pip --version`.

Install Visual Studio Code

By using VS Code as your text editor / integrated development environment (IDE), you can take advantage of IntelliSense (a code completion aid), Linting (helps avoid making errors in your code), Debug support (helps you find errors in your code after you run it), Code snippets (templates for small reusable code blocks), and Unit testing (testing your code's interface with different types of input).

Download VS Code for Windows and follow the installation instructions: https://code.visualstudio.com.

Install the Microsoft Python extension

You will need to install the Microsoft Python extension in order to take advantage of the VS Code support features. Learn more.

1. Open the VS Code Extensions window by entering **Ctrl+Shift+X** (or use the menu to navigate to **View** > **Extensions**).
2. In the top **Search Extensions in Marketplace** box, enter: **Python**.
3. Find the **Python (ms-python.python) by Microsoft** extension and select the green **Install** button.

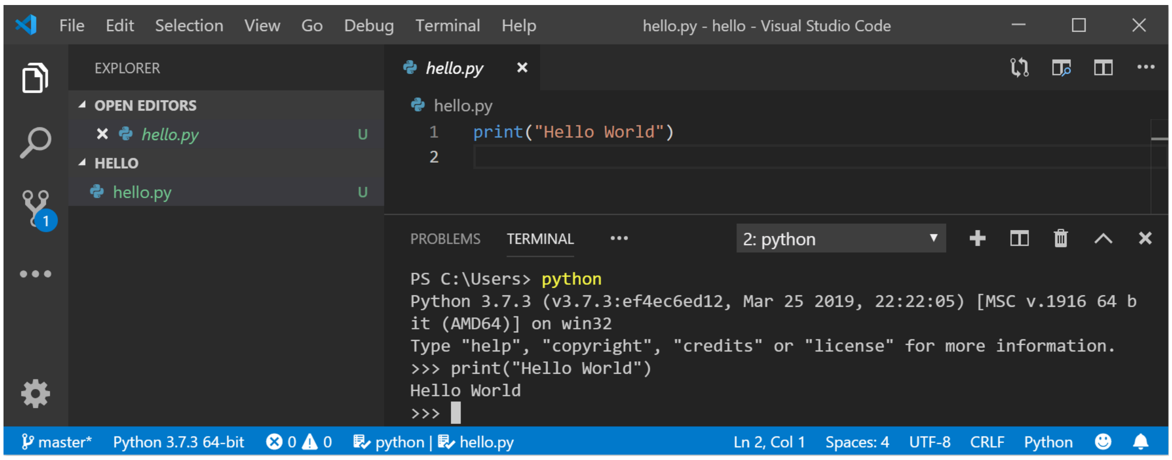
Open the integrated PowerShell terminal in VS Code

VS Code contains a built-in terminal that enables you to open a Python command line with PowerShell, establishing a seamless workflow between your code editor and command line.

1. Open the terminal in VS Code, select **View** > **Terminal**, or alternatively use the shortcut **Ctrl+`** (using the backtick character).

> **NOTE**
>
> The default terminal should be PowerShell, but if you need to change it, use **Ctrl+Shift+P** to enter the command pallette. Enter **Terminal: Select Default Shell** and a list of terminal options will display containing PowerShell, Command Prompt, WSL, etc. Select the one you'd like to use and enter **Ctrl+Shift+`** (using the backtick) to create a new terminal.

2. Inside your VS Code terminal, open Python by entering: `python`
3. Try the Python interpreter out by entering: `print("Hello World")`. Python will return your statement "Hello World".



```
print("Hello World")
```

```
PS C:\Users> python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

4. To exit Python, you can enter `exit()`, `quit()`, or select Ctrl-Z.

Install Git (optional)

If you plan to collaborate with others on your Python code, 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. You first need to install Git to power the Source Control panel.

1. Download and install Git for Windows from the git-scm website.
2. An Install Wizard is included that will ask you a series of questions about settings for your Git installation. We recommend using all of the default settings, unless you have a specific reason for changing something.
3. If you've never worked with Git before, GitHub Guides can help you get started.

Example script to display the structure of your file system directory

Common system administration tasks can take a huge amount of time, but with a Python script, you can automate these tasks so that they take no time at all. For example, Python can read the contents of your computer's file system and perform operations like printing an outline of your files and directories, moving folders from one directory to another, or renaming hundreds of files. Normally, tasks like these could take up a ton of time if you were to perform them manually. Use a Python script instead!

Let's begin with a simple script that walks a directory tree and displays the directory structure.

1. Open PowerShell using the **Start** menu (lower left Windows icon).
2. Create a directory for your project: `mkdir python-scripts`, then open that directory: `cd python-scripts`.
3. Create a few directories to use with our example script:

   ```
   mkdir food, food\fruits, food\fruits\apples, food\fruits\oranges, food\vegetables
   ```

4. Create a few files within those directories to use with our script:

   ```
   new-item food\fruits\banana.txt, food\fruits\strawberry.txt, food\fruits\blueberry.txt,
   food\fruits\apples\honeycrisp.txt, food\fruits\oranges\mandarin.txt, food\vegetables\carrot.txt
   ```

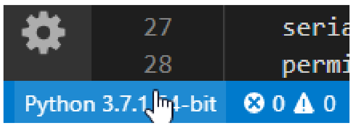
5. Create a new python file in your python-scripts directory:

   ```
   mkdir src
   new-item src\list-directory-contents.py
   ```

6. Open your project in VS Code by entering: `code .`
7. Open the VS Code File Explorer window by entering **Ctrl+Shift+E** (or use the menu to navigate to **View** > **Explorer**) and select the list-directory-contents.py file that you just created. The Microsoft Python extension will automatically load a Python interpreter. You can see which interpreter was loaded on the bottom of your VS Code window.

NOTE

Python is an interpreted language, meaning that it acts as a virtual machine, emulating a physical computer. There are different types of Python interpreters that you can use: Python 2, Python 3, Anaconda, PyPy, etc. In order to run Python code and get Python IntelliSense, you must tell VS Code which interpreter to use. We recommend sticking with the interpreter that VS Code chooses by default (Python 3 in our case) unless you have a specific reason for choosing something different. To change the Python interpreter, select the interpreter currently displayed in blue bar on the bottom of your VS Code window or open the **Command Palette** (Ctrl+Shift+P) and enter the command **Python: Select Interpreter**. This will display a list of the Python interpreters that you currently have installed. Learn more about configuring Python environments.



8. Paste the following code into your list-directory-contents.py file and then select **save**:

```
import os

root = os.path.join('..', 'food')
for directory, subdir_list, file_list in os.walk(root):
    print('Directory:', directory)
    for name in subdir_list:
        print('Subdirectory:', name)
    for name in file_list:
        print('File:', name)
    print()
```

9. Open the VS Code integrated terminal (**Ctrl+`**, using the backtick character) and enter the src directory where you just saved your Python script:

```
cd src
```

10. Run the script in PowerShell with:

```
python3 .\list-directory-contents.py
```

You should see output that looks like this:

```
Directory: ..\food
Subdirectory: fruits
Subdirectory: vegetables

Directory: ..\food\fruits
Subdirectory: apples
Subdirectory: oranges
File: banana.txt
File: blueberry.txt
File: strawberry.txt

Directory: ..\food\fruits\apples
File: honeycrisp.txt

Directory: ..\food\fruits\oranges
File: mandarin.txt

Directory: ..\food\vegetables
File: carrot.txt
```

11. Use Python to print that file system directory output to it's own text file by entering this command directly in your PowerShell terminal: `python3 list-directory-contents.py > food-directory.txt`

Congratulations! You've just written an automated systems administration script that reads the directory and files you created and uses Python to display, and then print, the directory structure to it's own text file.

NOTE

If you're unable to install Python 3 from the Microsoft Store, see this issue for an example of how to handle the pathing for this sample script.

Example script to modify all files in a directory

This example uses the files and directories you just created, renaming each of the files by adding the file's last modified date to the beginning of the filename.

1. Inside the **src** folder in your **python-scripts** directory, create a new Python file for your script:

```
new-item update-filenames.py
```

2. Open the update-filenames.py file, paste the following code into the file, and save it:

   NOTE

   os.getmtime returns a timestamp in ticks, which is not easily readable. It must be converted to a standard datetime string first.

```
import datetime
import os

root = os.path.join('..', 'food')
for directory, subdir_list, file_list in os.walk(root):
    for name in file_list:
        source_name = os.path.join(directory, name)
        timestamp = os.path.getmtime(source_name)
        modified_date = str(datetime.datetime.fromtimestamp(timestamp)).replace(':', '.')
        target_name = os.path.join(directory, f'{modified_date}_{name}')

        print(f'Renaming: {source_name} to: {target_name}')

        os.rename(source_name, target_name)
```

3. Test your update-filenames.py script by running it: `python3 update-filenames.py` and then running your list-directory-contents.py script again: `python3 list-directory-contents.py`

4. You should see output that looks like this:

```
Renaming: ..\food\fruits\banana.txt to: ..\food\fruits\2019-07-18 12.24.46.385185_banana.txt
Renaming: ..\food\fruits\blueberry.txt to: ..\food\fruits\2019-07-18 12.24.46.391170_blueberry.txt
Renaming: ..\food\fruits\strawberry.txt to: ..\food\fruits\2019-07-18 12.24.46.389174_strawberry.txt
Renaming: ..\food\fruits\apples\honeycrisp.txt to: ..\food\fruits\apples\2019-07-18
12.24.46.395160_honeycrisp.txt
Renaming: ..\food\fruits\oranges\mandarin.txt to: ..\food\fruits\oranges\2019-07-18
12.24.46.398151_mandarin.txt
Renaming: ..\food\vegetables\carrot.txt to: ..\food\vegetables\2019-07-18 12.24.46.402496_carrot.txt

PS C:\src\python-scripting\src> python3 .\list-directory-contents.py
..\food\
Directory: ..\food
Subdirectory: fruits
Subdirectory: vegetables

Directory: ..\food\fruits
Subdirectory: apples
Subdirectory: oranges
File: 2019-07-18 12.24.46.385185_banana.txt
File: 2019-07-18 12.24.46.389174_strawberry.txt
File: 2019-07-18 12.24.46.391170_blueberry.txt

Directory: ..\food\fruits\apples
File: 2019-07-18 12.24.46.395160_honeycrisp.txt

Directory: ..\food\fruits\oranges
File: 2019-07-18 12.24.46.398151_mandarin.txt

Directory: ..\food\vegetables
File: 2019-07-18 12.24.46.402496_carrot.txt
```

5. Use Python to print the new file system directory names with the last-modified timestamp prepended to it's own text file by entering this command directly in your PowerShell terminal: `python3 list-directory-contents.py > food-directory-last-modified.txt`

Hope you learned a few fun things about using Python scripts for automating basic systems administration tasks. There is, of course, a ton more to know, but we hope this got you started on the right foot. We've shared a few additional resources to continue learning below.

Additional resources

- Python Docs: File and Directory Access: Python documentation about working with file systems and using modules for reading the properties of files, manipulating paths in a portable way, and creating temporary files.
- Learn Python: String_Formatting tutorial: More about using the "%" operator for string formatting.
- 10 Python File System Methods You Should Know: Medium article about manipulating files and folders With `os` and `shutil`.
- The Hitchhikers Guide to Python: Systems Administration: An "opinionated guide" that offers overviews and best practices on topics related to Python. This section covers System Admin tools and frameworks. This guide is hosted on GitHub so you can file issues and make contributions.

Overview of Android development on Windows

5/13/2021 • 4 minutes to read • Edit Online

There are multiple paths for developing an Android device app using the Windows operating system. These paths fall into three main types: **Native Android development**, **Cross-platform development**, and **Android game development**. This overview will help you decide which development path to follow for developing an Android app and then provide next steps to help you get started using Windows to develop with:

- Native Android
- Xamarin.Android
- Xamarin.Forms
- React Native
- Cordova, Ionic, or PhoneGap
- C/C++ for game development

In addition, this guide will provide tips on using Windows to:

- Test on an Android device or emulator
- Update Windows Defender settings to improve performance
- Develop dual-screen apps for Android and get the Surface Duo device SDK

Native Android

Native Android development on Windows means that your app is targeting only Android (not iOS or Windows devices). You can use Android Studio or Visual Studio to develop within the ecosystem designed specifically for the Android operating system. Performance will be optimized for Android devices, the user-interface look and feel will be consistent with other native apps on the device, and any features or capabilities of the user's device will be straight-forward to access and utilize. Developing your app in a native format will help it to just 'feel right' because it follows all of the interaction patterns and user experience standards established specifically for Android devices.

Cross-platform

Cross-platform frameworks provide a single codebase that can (mostly) be shared between Android, iOS, and Windows devices. Using a cross-platform framework can help your app to maintain the same look, feel, and experience across device platforms, as well as benefiting from the automatic rollout of updates and fixes. Instead of needing to understand a variety of device-specific code languages, the app is developed in a shared codebase, typically in one language.

While cross-platform frameworks aim to look and feel as close to native apps as possible, they will never be as seamlessly integrated as a natively developed app and may suffer from reduced speed and degraded performance. Additionally, the tools used to build cross-platform apps may not have all of the features offered by each different device platform, potentially requiring workarounds.

A codebase is typically made up of **UI code**, for creating the user interface like pages, buttons controls, labels, lists, etc., and **logic code**, for calling web services, accessing a database, invoking hardware capabilities and managing state. On average, 90% of this can be reused, though there is typically some need to customize code for each device platform. This generalization largely depends on the type of app you're building, but provides a bit of context that hopefully will help with your decision-making.

Choosing a cross-platform framework

Xamarin Native (Xamarin.Android)

- UI code: XML with Android Designer, and Material Theme
- Logic code: C# or F#
- Still able to tap into some native Android elements, but good for reuse of the code base for other platforms (iOS, Windows).
- Only logic code is shared across platforms, not UI code.
- Great for more complex apps with a device-specific user interface.

Xamarin Forms (Xamarin.Forms)

- UI code: XAML and .NET (with Visual Studio)
- Logic code: C#
- Shares around 60–90% of the logic and UI code across Android, iOS, and Windows device apps.
- Uses common user controls like Button, Label, Entry, ListView, StackLayout, Calendar, TabbedPage, etc. Create a Button and Xamarin Forms will figure out how to call the native button for each platform using the Binding Library to call Java or Swift code from C#.
- Great for simple apps, like internal or Line Of Business (LOB) apps, prototypes or MVPs. Any app that can look somewhat standard or generic, utilizing a simple user interface.

React Native

- UI code: JavaScript
- Logic code: JavaScript
- The goal of React Native isn't to write the code once and run it on any platform, rather to learn-once (the React way) and write-anywhere.
- The community has added tools such as Expo and Create React Native App to help those wanting to build apps without using Xcode or Android Studio.
- Similar to Xamarin (C#), React Native (JavaScript) calls native UI elements (without the need for writing Java/Kotlin or Swift).

Progressive Web Apps (PWAs)

- UI code: HTML, CSS, JavaScript
- Logic code: JavaScript
- PWAs are web apps built with standard patterns to allow them to take advantage of both web and native app features. They can be built without a framework, but a couple of popular frameworks to consider are Ionic and PhoneGap.
- PWAs can be installed on a device (Android, iOS, or Windows) and can work offline thanks to the incorporation of a service-worker.
- PWAs can be distributed and installed without an app store using only a web URL. The Microsoft Store and Google Play Store allow PWAs to be listed, the Apple Store currently does not, though they can still be installed on any iOS device running 12.2 or later.
- To learn more, check out this introduction to PWAs on MDN.

Game development

Game development for Android is often unique from developing a standard Android app since games typically use custom rendering logic, often written in OpenGL or Vulkan. For this reason, and because of the many C libraries available that support game development, it's common for developers to use C/C++ with Visual Studio, along with the Android Native Development Kit (NDK), to create games for Android. Get started with C/C++ for game development.

Another common path for developing games for Android is to use a game engine. There are many free and open-source engines available, such as Unity with Visual Studio, Unreal Engine, MonoGame with Xamarin, UrhoSharp with Xamarin, SkiaSharp with Xamarin.Forms CocoonJS, App Game Kit, Fusion, Corona SDK, Cocos 2d, and more.

Next steps

- Get started with native Android development on Windows
- Get started developing for Android using Xamarin.Android
- Get started developing for Android using Xamarin.Forms
- Get started developing for Android using React Native
- Get started developing a PWA for Android
- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance
- Enable Virtualization support to improve emulator performance

Get started with native Android development on Windows

7/14/2021 • 8 minutes to read • Edit Online

This guide will get you started using Windows to create native Android applications. If you would prefer a cross-platform solution, see Overview of Android development on Windows for a brief summary of some options.

The most straight-forward way to create a native Android app is using Android Studio with either Java or Kotlin, though it is also possible to use C or C++ for Android development if you have a specific purpose. The Android Studio SDK tools compile your code, data, and resource files into an archive Android package, .apk file. One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.

Install Android Studio

Android Studio is the official integrated development environment for Google's Android operating system. Download the latest version of Android Studio for Windows.

- If you downloaded an .exe file (recommended), double-click to launch it.
- If you downloaded a .zip file, unpack the ZIP, copy the android-studio folder into your Program Files folder, and then open the android-studio > bin folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).

Follow the setup wizard in Android Studio and install any SDK packages that it recommends. As new tools and other APIs become available, Android Studio will notify you with a pop-up, or check for updates by selecting **Help** > **Check for Update**.

Create a new project

Select **File** > **New** > **New Project**.

In the **Choose your project** window, you will be able to choose between these templates:

- **Basic Activity**: Creates a simple app with an app bar, a floating action button and two layout files: one for the activity and one to separate out text content.

- **Empty Activity**: Creates an empty activity and a single layout file with sample text content.

- **Bottom Navigation Activity**: Creates a standard bottom navigation bar for an activity. For more information on this, see the Bottom Navigation Component section of the Material Design guidelines by Google.

- Templates are commonly used to add activities to new and existing app modules. For example, to create a login screen for your app's users, add an activity with the Login Activity template. To learn more about selecting an activity and how to add code from a template, see Android Developer guide by Google.

NOTE

The Android operating system is based on the idea of **components** and uses the terms **activity** and **intent** to define interactions. An **activity** represents a single, focused task that a user can do. An **activity** provides a window for building the user interface using classes based on the **View** class. There is a lifecycle for **activities** in the Android operating system, defined by six callbacks: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, and `onDestroy()`. The activity components interact with one another using **intent** objects. Intent either defines the activity to start or describes the type of action to perform (and the system selects the appropriate activity for you, which can even be from a different application). Learn more about Activities, the Activity Lifecycle, and Intents in the Android Developer guide by Google.

Java or Kotlin

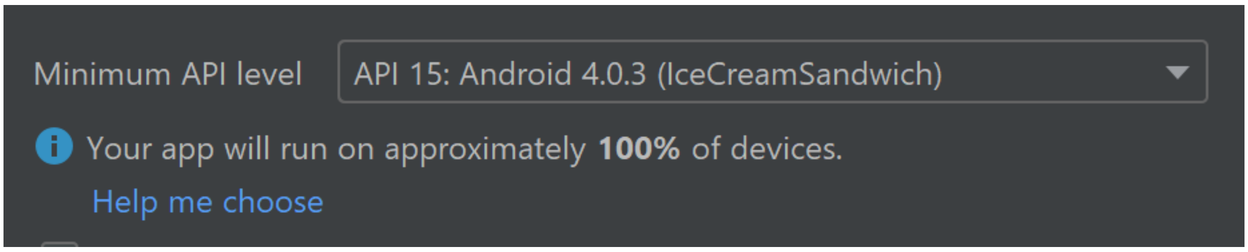
Java became a language in 1991, developed by what was then Sun Microsystems, but which is now owned by Oracle. It has become one of the most popular and powerful programming languages with one of the largest support communities in the world. Java is class-based and object-oriented, designed to have as few implementation dependencies as possible. The syntax is similar to C and C++, but it has fewer low-level facilities than either of them.

Kotlin was first announced as a new open-source language by JetBrains in 2011 and has been included as an alternative to Java in Android Studio since 2017. In May 2019, Google announced Kotlin as it's preferred language for Android app developers, so despite being a newer language, it also has a strong support community and has been identified as one of the fastest growing programming languages. Kotlin is cross-platform, statically typed, and designed to interoperate fully with Java.

Java is more widely used for a broader range of applications and offers some features that Kotlin does not, such as checked exceptions, primitive types that are not classes, static members, non-private fields, wildcard-types, and ternary-operators. Kotlin is specifically designed for and recommended by Android. It also offers some features that Java does not, such as null references controlled by the type system, no raw types, invariant arrays, proper function types (as opposed to Java's SAM-conversions), use-site variance without wildcards, smart casts, and more. Find a more in-depth look at the comparison to Java in the Kotlin documentation.

Minimum API Level

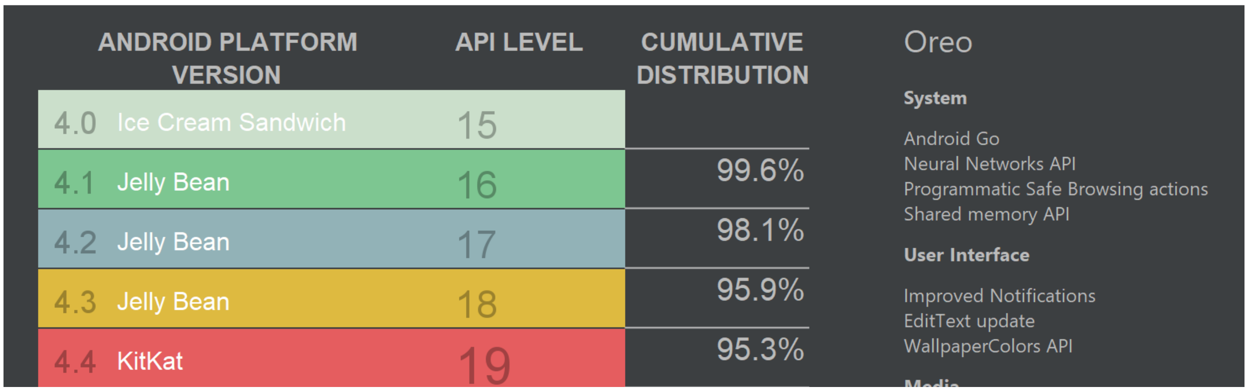
You will need to decide the minimum API level for your application. This determines which version of Android your application will support. Lower API levels are older and therefore generally support more devices, but higher API levels are newer and therefor provide more features.


Minimum API level: API 15: Android 4.0.3 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Select the **Help me choose** link to open a comparison chart showing the device support distribution and key features associated with the platform version release.

Android Platform/API Version Distribution



ANDROID PLATFORM VERSION		API LEVEL	CUMULATIVE DISTRIBUTION
4.0	Ice Cream Sandwich	15	
4.1	Jelly Bean	16	99.6%
4.2	Jelly Bean	17	98.1%
4.3	Jelly Bean	18	95.9%
4.4	KitKat	19	95.3%

Oreo

System

Android Go
Neural Networks API
Programmatic Safe Browsing actions
Shared memory API

User Interface

Improved Notifications
EditText update
WallpaperColors API

Instant app support and Androidx artifacts

You may notice a checkbox to **Support instant apps** and another to **Use androidx artifacts** in your project creation options. The *instant apps support* is not checked and the *androidx* is checked as the recommended default.

Google Play **Instant apps** provide a way for people to try an app or game without installing it first. These instant apps can be surfaced across the Play Store, Google Search, social networks, and anywhere you share a link. By checking the **Support instant apps** box, you are asking Android Studio to include the Google Play Instant Development SDK with your project. Learn more about Google Play Instant apps in the Android developer guide.

AndroidX artifacts represents the new version of the Android support library and provides backwards-compatibility across Android releases. AndroidX provides a consistent namespace starting with the string androidx for all available packages.

NOTE

AndroidX is now the default library. To uncheck this box and use the previous support library requires removing the lastest Android Q SDK. See Uncheck use Androidx artifacts on StackOverflow for instructions, but first note that the former Support Library packages have been mapped into corresponding androidx.* packages. For a full mapping of all the old classes and build artifacts to the new ones, see Migrating to AndroidX.

Project files

The Android Studio **Project** window, contains the following files (be sure that the Android view is selected from the drop-down menu):

app > java > com.example.myfirstapp > MainActivity

The main activity and entry point for your app. When you build and run your app, the system launches an instance of this Activity and loads its layout.

app > res > layout > activity_main.xml

The XML file defining the layout for the activity's user interface (UI). It contains a TextView element with the text "Hello World"

app > manifests > AndroidManifest.xml

The manifest file describing the fundamental characteristics of the app and each of its components.

Gradle Scripts > build.gradle

There are two files with this name: "Project: My First App", for the entire project, and "Module: app", for each app module. A new project will initially only have one module. Use the module's build.file to control how the Gradle plugin builds your app. Learn more about how to configure your build in the Android developer guide.

Use C or C++ for Android game development

The Android operating system is designed to support applications written in Java or Kotlin, benefiting from tooling embedded in the system's architecture. Many system features, like Android UI and Intent handling, are only exposed through Java interfaces. There are a few instances where you may want to **use C or C++ code via the Android Native Development Kit (NDK)** despite some of the associated challenges. Game development is an example, since games typically use custom rendering logic written in OpenGL or Vulkan and benefit from a wealth of C libraries focused on game development. Using C or C++ *might* also help you squeeze extra performance out of a device to achieve low latency or run computationally intensive applications, such as physics simulations. The NDK **is not appropriate for most novice Android programmers** however. Unless you have a specific purpose for using the NDK, we recommend sticking with Java, Kotlin, or one of the cross-platform frameworks.

To create a new project with C/C++ support:

- In the **Choose your project** section of the Android Studio wizard, select the *Native C++** project type. Select **Next**, complete the remaining fields, then select **Next** again.
- In the **Customize C++ Support** section of the wizard, you can customize your project with the **C++ Standard** field. Use the drop-down list to select which standardization of C++ you want to use. Selecting **Toolchain Default** uses the default CMake setting. Select **Finish**.
- Once Android Studio creates your new project, you can find a **cpp** folder in the **Project** pane that contains the native source files, headers, build scripts for CMake or ndk-build, and prebuilt libraries that are a part of your project. You can also find a sample C++ source file, `native-lib.cpp`, in the `src/main/cpp/` folder which provides a simple `stringFromJNI()` function returning the string "Hello from C++". Additionally, you should see a CMake build script, `CMakeLists.txt`, in your module's root directory required for building your native library.

To learn more, about adding C and C++ code to your project, see the Android developer guide. To find Android NDK samples with C++ integration, see the Android NDK samples repo on GitHub. To compile and run a C++ game on Android, use the Google Play Game services API.

Design guidelines

Device users expect applications to look and behave a certain way... whether swiping or tapping or using voice-controls, users will hold specific expectations for what your application should look like and how to use it. These expectations should remain consistent in order to reduce confusion and frustration. Android offers a guide to these platform and device expectations that combines the Google Material Design foundation for visual and navigational patterns, along with quality guidelines for compatibility, performance, and security.

Learn more in the Android design documentation.

Fluent Design System for Android

Microsoft also offers design guidance with the goal of providing a seamless experience across the entire portfolio of Microsoft's mobile apps.

Fluent Design System for Android design and build custom apps that are natively Android while still uniquely Fluent.

- Sketch toolkit
- Figma toolkit
- Android font
- Android User Interface Guidelines

- Guidelines for Android app icons

Additional resources

- Android Application Fundamentals
- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance
- Enable Virtualization support to improve emulator performance

Get started developing for Android using Xamarin.Android

3/6/2021 • 4 minutes to read • Edit Online

This guide will help you to get started using Xamarin.Android on Windows to create a cross-platform app that will work on Android devices.

In this article, you will create a simple Android app using Xamarin.Android and Visual Studio 2019.

Requirements

To use this tutorial, you'll need the following:

- Windows 10
- Visual Studio 2019: Community, Professional, or Enterprise (see note)
- The "Mobile development with .NET" workload for Visual Studio 2019

> **NOTE**
>
> This guide will work with Visual Studio 2017 or 2019. If you are using Visual Studio 2017, some instructions may be incorrect due to UI differences between the two versions of Visual Studio.

You will also to have an Android phone or configured emulator in which to run your app. See Configuring an Android emulator.

Create a new Xamarin.Android project

Start Visual Studio. Select File > New > Project to create a new project.

In the new project dialog, select the **Android App (Xamarin)** template and click **Next**.

Name the project **TimeChangerAndroid** and click **Create**.

In the New Cross Platform App dialog, select **Blank App**. In the **Minimum Android Version**, select **Android 5.0 (Lollipop)**. Click **OK**.

Xamarin will create a new solution with a single project named **TimeChangerAndroid**.

Create a UI with XAML

In the **Resources\layout** directory of your project, open **activity_main.xml**. The XML in this file defines the first screen a user will see when opening TimeChanger.

TimeChanger's UI is simple. It displays the current time and has buttons to adjust the time in increments of one hour. It uses a vertical `LinearLayout` to align the time above the buttons and a horizontal `LinearLayout` to arrange the buttons side-by-side. The content is centered in the screen by setting **android:gravity** attribute to **center** in the vertical `LinearLayout`.

Replace the contents of **activity_main.xml** with the following code.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="At runtime, I will display current time"
        android:id="@+id/timeDisplay"
    />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Up"
            android:id="@+id/upButton"/>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Down"
            android:id="@+id/downButton"/>
    </LinearLayout>
</LinearLayout>
```

At this point you can run **TimeChangerAndroid** and see the UI you've created. In the next section, you will add functionality to your UI displaying the current time and enabling the buttons to perform an action.

Add logic code with C#

Open **MainActivity.cs**. This file contains the code-behind logic that will add functionality to the UI.

Set the current time

First, get a reference to the `TextView` that will display the time. Use **FindViewById** to search all UI elements for the one with the correct **android:id** (which was set to `"@+id/timeDisplay"` in the xml from the previous step). This is the `TextView` that will display the current time.

```
var timeDisplay = FindViewById<TextView>(Resource.Id.timeDisplay);
```

UI controls must be updated on the UI thread. Changes made from another thread may not properly update the control as it displays on the screen. Because there is no guarantee this code will always be running on the UI thread, use the **RunOnUiThread** method to make sure any updates display correctly. Here is the complete `UpdateTimeLabel` method.

```
private void UpdateTimeLabel(object state = null)
{
    RunOnUiThread(() =>
    {
        TimeDisplay.Text = DateTime.Now.ToLongTimeString();
    });
}
```

Update the current time once every second

At this point, the current time will be accurate for, at most, one second after TimeChangerAndroid is launched. The label must be periodically updated to keep the time accurate. A **Timer** object will periodically call a callback method that updates the label with the current time.

```
var clockRefresh = new Timer(dueTime: 0, period: 1000, callback: UpdateTimeLabel, state: null);
```

Add HourOffset

The up and down buttons adjust the time in increments of one hour. Add an **HourOffset** property to track the current adjustment.

```
public int HourOffset { get; private set; }
```

Now update the UpdateTimeLabel method to be aware of the HourOffset property.

```
TimeDisplay.Text = DateTime.Now.AddHours(HourOffset).ToLongTimeString();
```

Create the button Click event handlers

All the up and down buttons need to do is increment or decrement the HourOffset property and call UpdateTimeLabel.

```
public void UpButton_Click(object sender, System.EventArgs e)
{
    HourOffset++;
    UpdateTimeLabel();
}
```

Wire up the up and down buttons to their corresponding event handlers

To associate the buttons with their corresponding event handlers, first use FindViewById to find the buttons by their ids. Once you have a reference to the button object, you can add an event handler to its `Click` event.

```
Button upButton = FindViewById<Button>(Resource.Id.upButton);
upButton.Click += UpButton_Click;
```

Completed MainActivity.cs file

When you're finished, MainActivity.cs should look like this:

```
using Android.App;
using Android.OS;
using Android.Support.V7.App;
using Android.Runtime;
using Android.Widget;
using System;
using System.Threading;

namespace TimeChangerAndroid
{
    [Activity(Label = "@string/app_name", Theme = "@style/AppTheme", MainLauncher = true)]
    public class MainActivity : AppCompatActivity
    {
        public TextView TimeDisplay { get; private set; }
        public int HourOffset { get; private set; }

        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);

            // Set the view from the "main" layout resource
            SetContentView(Resource.Layout.activity_main);

            var clockRefresh = new Timer(dueTime: 0, period: 1000, callback: UpdateTimeLabel, state: null);

            Button upButton = FindViewById<Button>(Resource.Id.upButton);
            upButton.Click += OnUpButton_Click;

            Button downButton = FindViewById<Button>(Resource.Id.downButton);
            downButton.Click += OnDownButton_Click;

            TimeDisplay = FindViewById<TextView>(Resource.Id.timeDisplay);
        }

        private void UpdateTimeLabel(object state = null)
        {
            // Timer callbacks run on a background thread, but UI updates must run on the UI thread.
            RunOnUiThread(() =>
            {
                TimeDisplay.Text = DateTime.Now.AddHours(HourOffset).ToLongTimeString();
            });
        }

        public void OnUpButton_Click(object sender, System.EventArgs e)
        {
            HourOffset++;
            UpdateTimeLabel();
        }

        public void OnDownButton_Click(object sender, System.EventArgs e)
        {
            HourOffset--;
            UpdateTimeLabel();
        }
    }
}
```

Run your app

To run the app, press **F5** or click Debug > Start Debugging. Depending on how your debugger is configured, your app will launch on a device or in an emulator.

Related links

- Test on an Android device or emulator.
- Create an Android sample app using Xamarin.Forms

Get started developing for Android using Xamarin.Forms

3/6/2021 • 4 minutes to read • Edit Online

This guide will help you to get started using Xamarin.Forms on Windows to create a cross-platform app that will work on Android devices.

In this article, you will create a simple Android app using Xamarin.Forms and Visual Studio 2019.

Requirements

To use this tutorial, you'll need the following:

- Windows 10
- Visual Studio 2019: Community, Professional, or Enterprise (see note)
- The "Mobile development with .NET" workload for Visual Studio 2019

NOTE

This guide will work with Visual Studio 2017 or 2019. If you are using Visual Studio 2017, some instructions may be incorrect due to UI differences between the two versions of Visual Studio.

You will also to have an Android phone or configured emulator in which to run your app. See Test on an Android device or emulator.

Create a new Xamarin.Forms project

Start Visual Studio. Click File > New > Project to create a new project.

In the new project dialog, select the **Mobile App (Xamarin.Forms)** template and click **Next**.

Name the project **TimeChangerForms** and click **Create**.

In the New Cross Platform App dialog, select **Blank**. In the Platform section, check **Android** and un-check all other boxes. Click **OK**.

Xamarin will create a new solution with two projects: **TimeChangerForms** and **TimeChangerForms.Android.**

Create a UI with XAML

Expand the **TimeChangerForms** project and open **MainPage.xaml**. The XAML in this file defines the first screen a user will see when opening TimeChanger.

TimeChanger's UI is simple. It displays the current time, and has buttons to adjust the time in increments of one hour. It uses a vertical StackLayout to align the time above the buttons, and a horizontal StackLayout to arrange the buttons side-by-side. The content is centered in the screen by setting the vertical StackLayout's **HorizontalOptions** and **VerticalOptions** to **"CenterAndExpand"**.

Replace the contents of MainPage.xaml with the following code.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="TimeChangerForms.MainPage">

    <StackLayout HorizontalOptions="CenterAndExpand"
                 VerticalOptions="CenterAndExpand">
        <Label x:Name="time"
               HorizontalOptions="CenterAndExpand"
               VerticalOptions="CenterAndExpand"
               Text="At runtime, this Label will display the current time.">
        </Label>
        <StackLayout Orientation="Horizontal">
            <Button HorizontalOptions="End"
                    VerticalOptions="End"
                    Text="Up"
                    Clicked="OnUpButton_Clicked"/>
            <Button HorizontalOptions="Start"
                    VerticalOptions="End"
                    Text="Down"
                    Clicked="OnDownButton_Clicked"/>
        </StackLayout>
    </StackLayout>
</ContentPage>
```

At this point, the UI is complete. TimeChangerForms, however, will not build because the methods **UpButton_Clicked** and **DownButton_Clicked** are referenced in the XAML but not defined anywhere. Even if the app did run, the current time would not be displayed. In the next section, you will fix these errors and add functionality to your UI.

Add logic code with C#

In the Solution Explorer, right click MainPage.xaml and click **View Code**. This file contains the code behind that will add functionality to the UI.

Set the current time

Code in this file can reference controls declared in the XAML using the value of the control's **x:Name** attribute. In this case the label that displays the current time is called `time`.

UI controls must be updated on the main thread. Changes made from another thread may not properly update the control as it displays on the screen. Because there is no guarantee this code will always be running on the main thread, use the **BeginInvokeOnMainThread** method to make sure any updates display correctly. Here is the complete UpdateTimeLabel method.

```
private void UpdateTimeLabel(object state = null)
{
    Device.BeginInvokeOnMainThread(() =>
        {
            time.Text = DateTime.Now.ToLongTimeString();
        }
    );
}
```

Update the current time once every second

At this point, the current time will be accurate for, at most, one second after TimeChangerForms is launched. The label must be periodically updated to keep the time accurate. A **Timer** object will periodically call a callback method that updates the label with the current time.

```
var clockRefresh = new Timer(dueTime: 0, period: 1000, callback: UpdateTimeLabel, state: null);
```

Add HourOffset

The up and down buttons adjust the time in increments of one hour. Add an **HourOffset** property to track the current adjustment.

```
public int HourOffset { get; private set; }
```

Now update the UpdateTimeLabel method to be aware of the HourOffset property.

```
currentTime.Text = DateTime.Now.AddHours(HourOffset).ToLongTimeString();
```

Add button Click event handlers

All the up and down buttons need to do is increment or decrement the HourOffset property and call UpdateTimeLabel.

```
private void UpButton_Clicked(object sender, EventArgs e)
{
    HourOffset++;
    UpdateTimeLabel();
}
```

When you're finished, MainPage.xaml.cs should look like this:

```
using System;
using System.ComponentModel;
using System.Threading;
using Xamarin.Forms;

namespace TimeChangerForms
{
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        public int HourOffset { get; private set; }

        public MainPage()
        {
            InitializeComponent();
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            var clockRefresh = new Timer(dueTime: 0, period: 1000, callback: UpdateTimeLabel, state: null);
        }

        private void UpdateTimeLabel(object state = null)
        {
            Device.BeginInvokeOnMainThread(() =>
                {
                    time.Text = DateTime.Now.AddHours(HourOffset).ToLongTimeString();
                }
            );
        }

        private void OnUpButton_Clicked(object sender, EventArgs e)
        {
            HourOffset++;
            UpdateTimeLabel();
        }

        private void OnDownButton_Clicked(object sender, EventArgs e)
        {
            HourOffset--;
            UpdateTimeLabel();
        }
    }
}
```

Run the app

To run the app, press **F5** or click Debug > Start Debugging. Depending on how your debugger is configured, your app will launch on a device or in an emulator.

Related links

- Test on an Android device or emulator.
- Create an Android sample app using Xamarin.Android

Get started developing for Android using React Native

4/21/2021 • 3 minutes to read • Edit Online

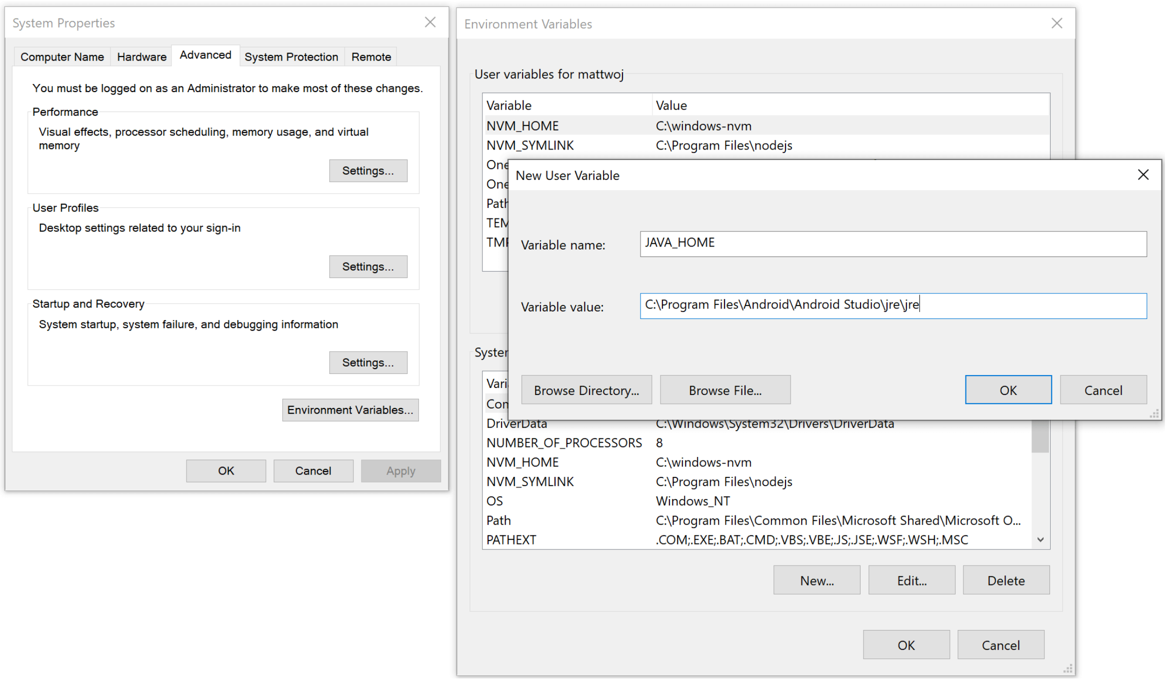
This guide will help you to get started using React Native on Windows to create a cross-platform app that will work on Android devices.

Overview

React Native is an open-source mobile application framework created by Facebook. It is used to develop applications for Android, iOS, Web and UWP (Windows) providing native UI controls and full access to the native platform. Working with React Native requires an understanding of JavaScript fundamentals.

Get started with React Native by installing required tools

1. Install Visual Studio Code (or your code editor of choice).
2. Install Android Studio for Windows. Android Studio installs the latest Android SDK by default. React Native requires Android 6.0 (Marshmallow) SDK or higher. We recommend using the latest SDK.
3. Create environment variable paths for the Java SDK and Android SDK:
   - In the Windows search menu, enter: "Edit the system environment variables", this will open the **System Properties** window.
   - Choose **Environment Variables...** and then choose **New...** under **User variables**.
   - Enter the Variable name and value (path). The default paths for the Java and Android SDKs are as follows. If you've chosen a specific location to install the Java and Android SDKs, be sure to update the variable paths accordingly.
     - JAVA_HOME: C:\Program Files\Android\Android Studio\jre\jre
     - ANDROID_HOME: C:\Users\username\AppData\Local\Android\Sdk



You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your sign-in

System startup, system failure, and debugging information

User variables for mattwoj

Variable	Value
NVM_HOME	C:\windows-nvm
NVM_SYMLINK	C:\Program Files\nodejs
One	
One	
Path	
TEM	
TMI	

Variable name: JAVA_HOME

Variable value: C:\Program Files\Android\Android Studio\jre\jre

Syster

Vari	
Con	
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
NVM_HOME	C:\windows-nvm
NVM_SYMLINK	C:\Program Files\nodejs
OS	Windows_NT
Path	C:\Program Files\Common Files\Microsoft Shared\Microsoft O...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

4. Install NodeJS for Windows You may want to consider using Node Version Manager (nvm) for Windows

if you will be working with multiple projects and version of NodeJS. We recommend installing the latest LTS version for new projects.

> **NOTE**
>
> You may also want to consider installing and using the Windows Terminal for working with your preferred command-line interface (CLI), as well as, Git for version control. The Java JDK comes packaged with Android Studio v2.2+, but if you need to update your JDK separately from Android Studio, use the Windows x64 Installer.

Create a new project with React Native

1. Use npx, the package runner tool that is installed with **npm** to create a new React Native project. from the Windows Command Prompt, PowerShell, Windows Terminal, or the integrated terminal in VS Code (View > Integrated Terminal).

   ```
   npx react-native init MyReactNativeApp
   ```

2. Open your new "MyReactNativeApp" directory:

   ```
   cd MyReactNativeApp
   ```

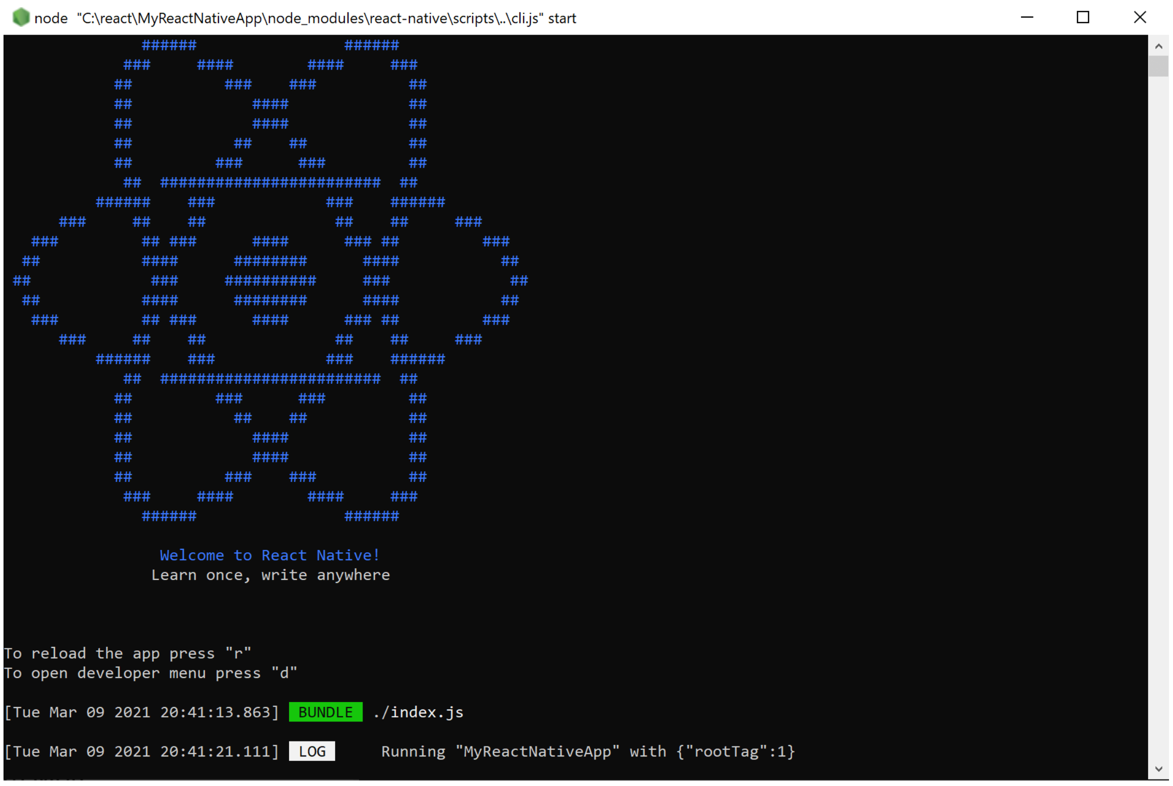
3. If you want to run your project on a hardware Android device, connect the device to your computer with a USB cable.

4. If you want to run your project on an Android emulator, you shouldn't need to take any action as Android Studio installs with a default emulator installed. If you want to run your app on the emulator for a particular device. Click on the **AVD Manager** button in the toolbar.

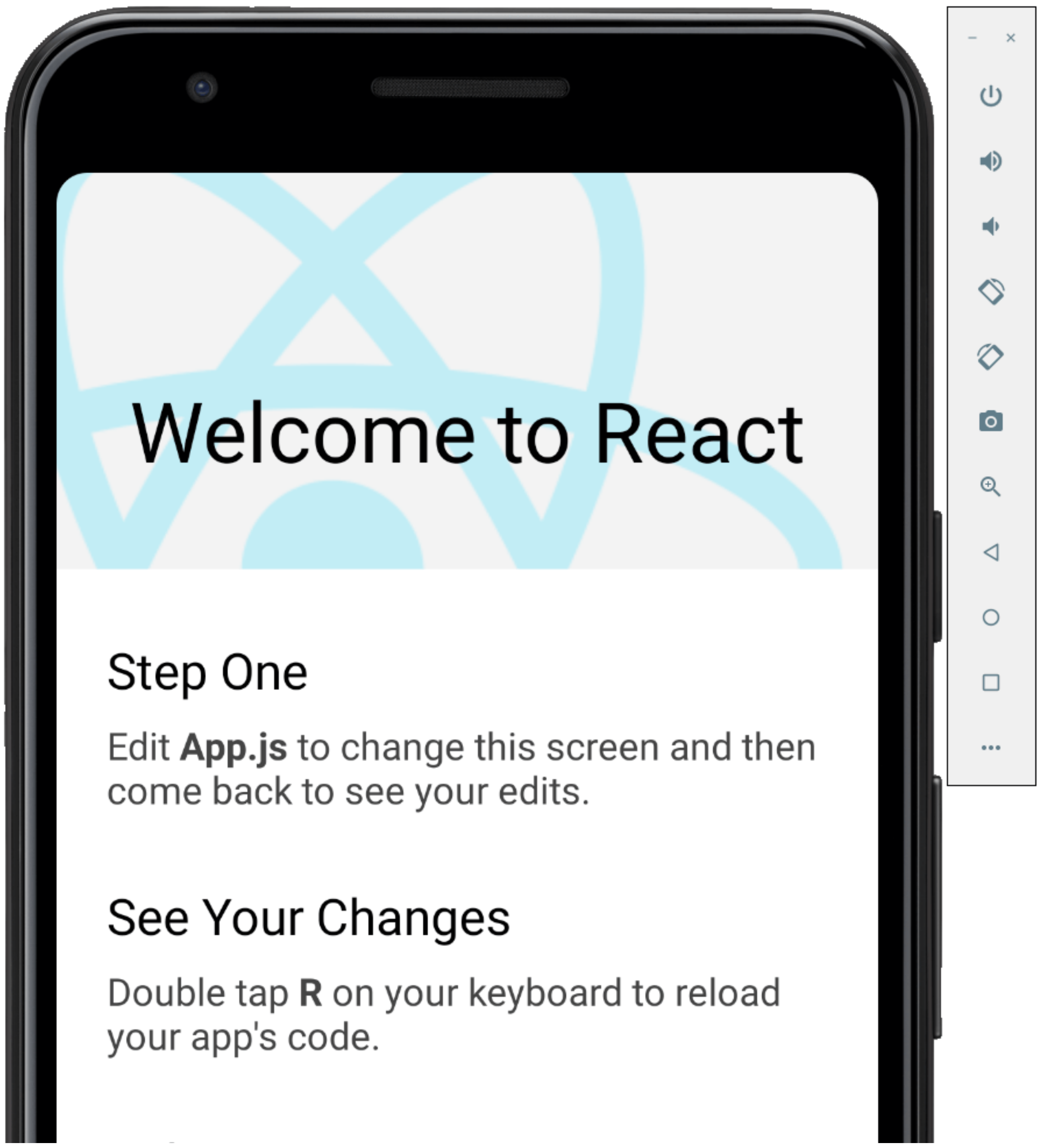
   .

5. To run your project, enter the following command. This will open a new console window displaying Node Metro Bundler.

   ```
   npx react-native run-android
   ```


node "C:\react\MyReactNativeApp\node_modules\react-native\scripts\..\cli.js" start

```
               ######                ######
             ###     ####        ####     ###
            ##          ###    ###          ##
            ##             ####             ##
            ##             ####             ##
            ##           ##    ##           ##
            ##         ###      ###         ##
             ##  ########################  ##
          ######    ###            ###    ######
      ###     ##    ##              ##    ##     ###
   ###         ## ###      ####      ### ##         ###
  ##           ####      ########      ####           ##
 ##             ###     ##########     ###             ##
  ##           ####      ########      ####           ##
   ###         ## ###      ####      ### ##         ###
      ###     ##    ##              ##    ##     ###
          ######    ###            ###    ######
             ##  ########################  ##
            ##         ###      ###         ##
            ##           ##    ##           ##
            ##             ####             ##
            ##             ####             ##
            ##          ###    ###          ##
             ###     ####        ####     ###
               ######                ######

                 Welcome to React Native!
                Learn once, write anywhere


To reload the app press "r"
To open developer menu press "d"

[Tue Mar 09 2021 20:41:13.863]  BUNDLE  ./index.js

[Tue Mar 09 2021 20:41:21.111]  LOG      Running "MyReactNativeApp" with {"rootTag":1}
```



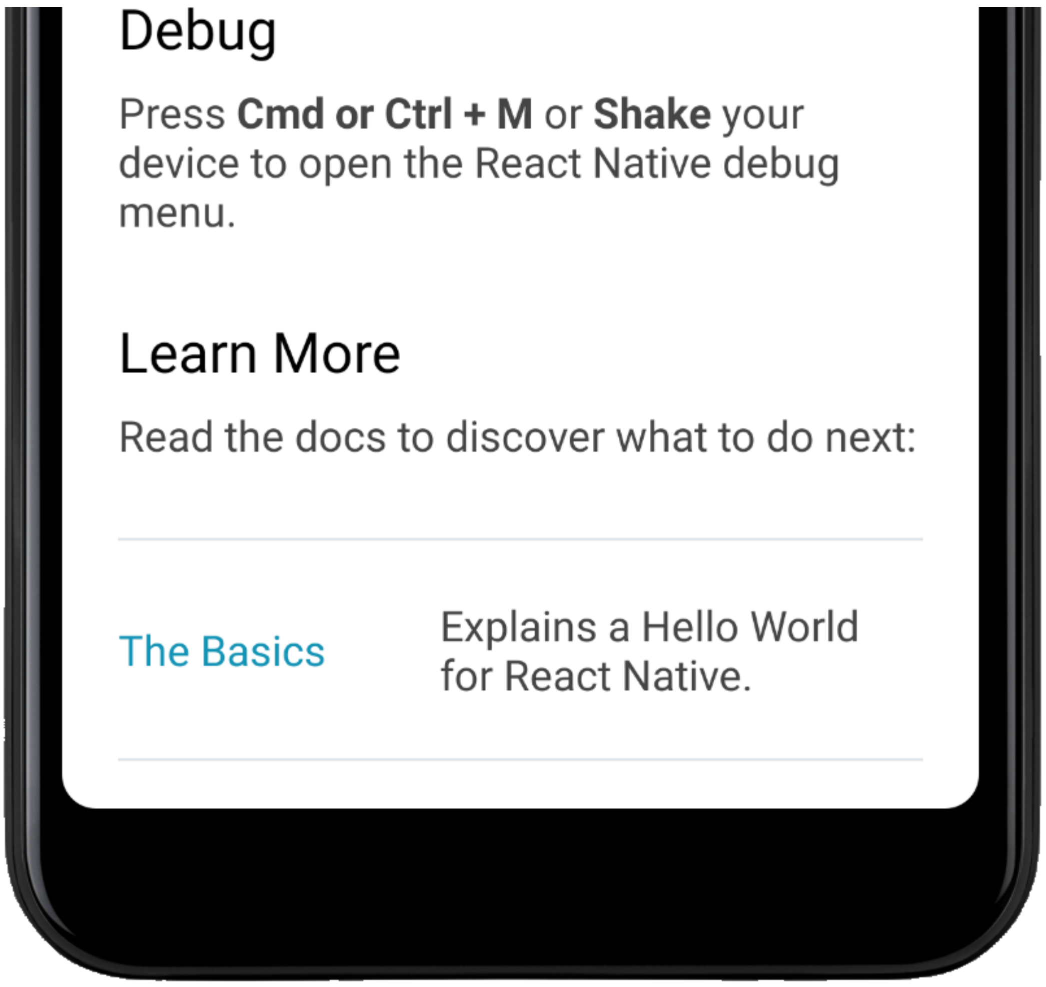
Welcome to React

Step One

Edit **App.js** to change this screen and then come back to see your edits.

See Your Changes

Double tap **R** on your keyboard to reload your app's code.


Debug
Press **Cmd or Ctrl + M** or **Shake** your device to open the React Native debug menu.
Learn More
Read the docs to discover what to do next:
The Basics
Explains a Hello World for React Native.

NOTE

If you are using a new install of Android Studio and haven't yet done any other Android development, you may get errors at the command line when you run the app about accepting licenses for the Android SDK. Such as "Warning: License for package Android SDK Platform 29 not accepted." To resolve this, you can click the **SDK Manager** button in Android Studio . Or, you can list and accept the licenses with the following command, making sure to use the path to the SDK location on your machine.

```
C:\Users\[User Name]\AppData\Local\Android\Sdk\tools\bin\sdkmanager --licenses
```

6. To modify the app, open the `MyReactNativeApp` project directory in the IDE of your choice. We recommend VS Code or Android Studio.

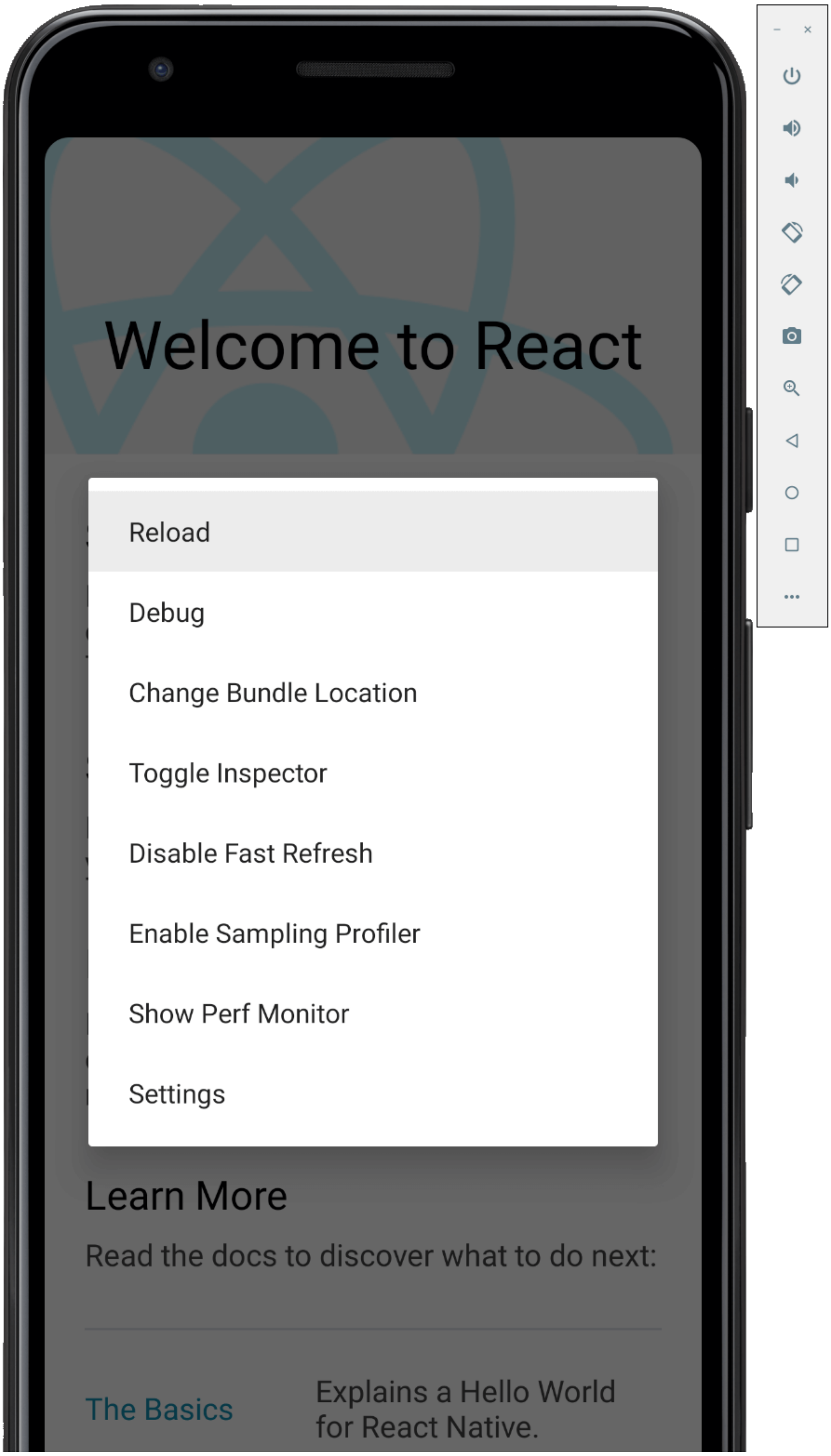
7. The project template created by `react-native init` uses a main page named `App.js`. This page is pre-populated with a lot of useful links to information about React Native development. Add some text to the first **Text** element, like the "HELLO WORLD!" string shown below.

```
<Text style={styles.sectionDescription}>
  Edit <Text style={styles.highlight}>App.js</Text> to change this
  screen and then come back to see your edits. HELLO WORLD!
</Text>
```

8. Reload the app to show the changes you made. There are several ways to do this.

- In the Metro Bundler console window, type "r".

- In the Android device emulator, double tap "r" on your keyboard.
- On a hardware android device, shake the device to bring up the React Native debug menu and select `Reload'.



Additional resources

- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance
- Enable Virtualization support to improve Emulator performance

Get started developing a PWA or Hybrid web app for Android

7/14/2021 • 5 minutes to read • Edit Online

This guide will help you to get started creating a hybrid web app or Progressive Web App (PWA) on Windows using a single HTML/CSS/JavaScript codebase that can be used on the web and across device platforms (Android, iOS, Windows).

By using the right frameworks and components, web-based applications can work on an Android device in a way that looks to users very similar to a native app.

Features of a PWA or Hybrid web app

There are two main types of web apps that can be installed on Android devices. The main difference being whether your application code is embedded in an app package (hybrid) or hosted on a web server (pwa).

- **Hybrid web apps**: Code (HTML, JS, CSS) is packaged in an APK and can be distributed via the Google Play Store. The viewing engine is isolated from the users' internet browser, no session or cache sharing.
- **Progressive Web Apps (PWAs)**: Code (HTML, JS, CSS) lives on the web and doesn't need to be packaged as an APK. Resources are downloaded and updated as needed using a Service Worker. The Chrome browser will render and display your app, but will look native and not include the normal browser address bar, etc. You can share storage, cache, and sessions with the browser. This is basically like installing a shortcut to the Chrome browser in a special mode. PWAs can also be listed in the Google Play Store using Trusted Web Activity.

PWAs and hybrid web apps are very similar to a native Android app in that they:

- Can be installed via the App Store (Google Play Store and/or Microsoft Store)
- Have access to native device features like camera, GPS, Bluetooth, notifications, and list of contacts
- Work Offline (no internet connection)

PWAs also have a few unique features:

- Can be installed on the Android home screen directly from the web (without an App Store)
- Can additionally be installed via the Google Play Store using a Trusted Web Activity
- Can be discovered via web search or shared via a URL link
- Rely on a Service Worker to avoid the need to package native code

You don't need a framework to create a Hybrid app or PWA, but there are a few popular frameworks that will be covered in this guide, including PhoneGap (with Cordova) and Ionic (with Cordova or Capacitor using Angular or React).

Apache Cordova

Apache Cordova is an open-source framework that can simplify the communication between your JavaScript code living in a native WebView and the native Android platform by using plugins. These plugins expose JavaScript endpoints that can be called from your code and used to call native Android device APIs. Some example Cordova plugins include access to device services like battery status, file access, vibration / ring tones, etc. These features are not typically available to web apps or browsers.

There are two popular distributions of Cordova:

- PhoneGap: Support has been discontinued by Adobe.
- Ionic

Ionic

Ionic is a framework that adjusts the user interface (UI) of your app to match the design language of each platform (Android, iOS, Windows). Ionic enables you to use either Angular or React.

> **NOTE**
>
> There is a new version of Ionic that uses an alternative to Cordova, called Capacitor. This alternative uses containers to make your app more web-friendly.

Get started with Ionic by installing required tools

To get started building a PWA or hybrid web app with Ionic, you should first install the following tools:

- Node.js for interacting with the Ionic ecosystem. Download NodeJS for Windows or follow the NodeJS installation guide using Windows Subsystem for Linux (WSL). You may want to consider using Node Version Manager (nvm) if you will be working with multiple projects and version of NodeJS.
- VS Code for writing your code. Download VS Code for Windows. You may also want to install the WSL Remote Extension if you prefer to build your app with a Linux command line.
- Windows Terminal for working with your preferred command-line interface (CLI). Install Windows Terminal from Microsoft Store.
- Git for version control. Download Git.

Create a new project with Ionic Cordova and Angular

Install Ionic and Cordova by entering the following in your command line:

```
npm install -g @ionic/cli cordova
```

Create an Ionic Angular app using the "Tabs" app template by entering the command:

```
ionic start photo-gallery tabs
```

Change into the app folder:

```
cd photo-gallery
```

Run the app in your web browser:

```
ionic serve
```

For more information, see the Ionic Cordova Angular docs. Visit the Making your Angular app a PWA section of the Ionic docs to learn how to move your app from being a hybrid to a PWA.

Create a new project with Ionic Capacitor and Angular

Install Ionic and Cordova-Res by entering the following in your command line:

```
npm install -g @ionic/cli native-run cordova-res
```

Create an Ionic Angular app using the "Tabs" app template and adding Capacitor by entering the command:

```
ionic start photo-gallery tabs --type=angular --capacitor
```

Change into the app folder:

```
cd photo-gallery
```

Add components to make the app a PWA:

```
npm install @ionic/pwa-elements
```

Import @ionic/pwa-elements by add the following to your `src/main.ts` file:

```
import { defineCustomElements } from '@ionic/pwa-elements/loader';

// Call the element loader after the platform has been bootstrapped
defineCustomElements(window);
```

Run the app in your web browser:

```
ionic serve
```

For more information, see the Ionic Capacitor Angular docs. Visit the Making your Angular app a PWA section of the Ionic docs to learn how to move your app from being a hybrid to a PWA.

Create a new project with Ionic and React

Install the Ionic CLI by entering the following in your command line:

```
npm install -g @ionic/cli
```

Create a new project with React by entering the command:

```
ionic start myApp blank --type=react
```

Change into the app folder:

```
cd myApp
```

Run the app in your web browser:

```
ionic serve
```

For more information, see the Ionic React docs. Visit the Making your React app a PWA section of the Ionic docs to learn how to move your app from being a hybrid to a PWA.

Test your Ionic app on a device or emulator

To test your Ionic app on an Android device, plug-in your device (make sure it is first enabled for development), then in your command line enter:

```
ionic cordova run android
```

To test your Ionic app on an Android device emulator, you must:

1. Install the required components -- Java Development Kit (JDK), Gradle, and the Android SDK.
2. Create an Android Virtual Device (AVD): See the [Android developer guide]] (https://developer.android.com/studio/run/managing-avds.html).
3. Enter the command for Ionic to build and deploy your app to the emulator: `ionic cordova emulate [<platform>] [options]`. In this case, the command should be:

```
ionic cordova emulate android --list
```

See the Cordova Emulator in the Ionic docs for more info.

Additional resources

- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance
- Enable Virtualization support to improve emulator performance

Update Windows Defender settings to improve performance

7/14/2021 • 2 minutes to read • Edit Online

This guide covers how to set up exclusions in your Windows Defender security settings in order to improve your build times and the overall performance speed of your Windows machine.

Windows Defender Overview

In Windows 10, version 1703 and later, the Windows Defender Antivirus app is part of Windows Security. Windows Defender aims to keep your PC safe with built-in, real-time protection against viruses, ransomware, spyware, and other security threats.

However, Windows Defender's real-time protection will also dramatically slow file system access and build speed when developing Android apps.

During the Android build process, many files are created on your computer. With antivirus real-time scanning enabled, the build process will halt each time a new file is created while the antivirus scans that file.

Fortunately, Windows Defender has the capability to exclude files, project directories, or file types that you know to be secure from it's antivirus scanning process.

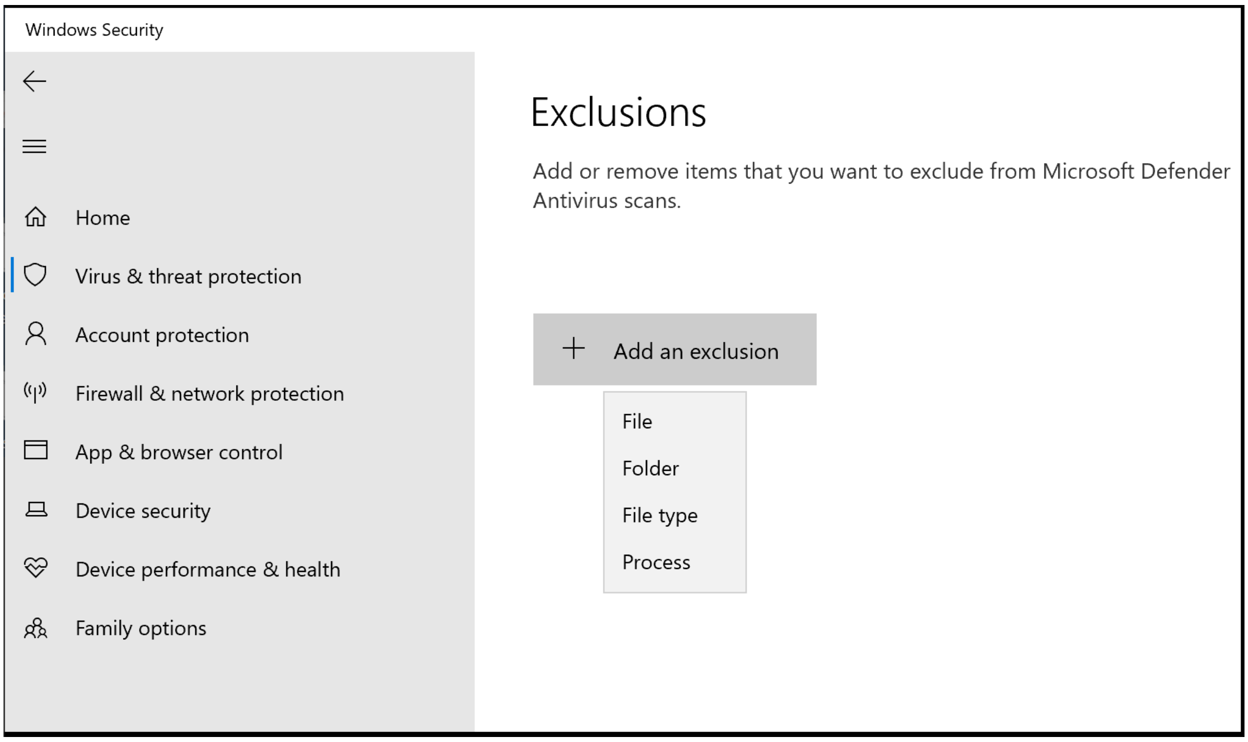
WARNING

To ensure that your computer is safe from malicious software, you should not completely disable real-time scanning or your Windows Defender antivirus software.

Add exclusions to Windows Defender

To improve your Android build speed, add exclusions in the Windows Defender Security Center by:

1. Select the Windows menu **Start** button
2. Enter **Windows Security**
3. Select **Virus and threat protection**
4. Select **Manage settings** under **Virus & threat protection settings**
5. Scroll to the **Exclusions** heading and select **Add or remove exclusions**
6. Select **+ Add an exclusion**. You will then need to choose whether the exclusion you wish to add is a **File**, **Folder**, **File type**, or **Process**.


Windows Security

Home

Virus & threat protection

Account protection

Firewall & network protection

App & browser control

Device security

Device performance & health

Family options

Exclusions

Add or remove items that you want to exclude from Microsoft Defender Antivirus scans.

Recommended exclusions

The following list shows the default location of each Android Studio directory recommended to add as an exclusion from Windows Defender real-time scanning:

- Gradle cache: `%USERPROFILE%\.gradle`
- Android Studio projects: `%USERPROFILE%\AndroidStudioProjects`
- Android SDK: `%USERPROFILE%\AppData\Local\Android\SDK`
- Android Studio system files: `%USERPROFILE%\.AndroidStudio<version>\system`

These directory locations may not apply to your project if you have not used the default locations set by Android Studio or if you have downloaded a project from GitHub (for example). Consider adding an exclusion to the directory of your current Android development project, wherever that may be located.

Additional exclusions you may want to consider include:

- Visual Studio dev environment process: `devenv.exe`
- Visual Studio build process: `msbuild.exe`
- JetBrains directory: `%LOCALAPPDATA%\JetBrains\<Transient directory (folder)>`

For more information on adding antivirus scanning exclusions, including how to customize directory locations for Group Policy controlled environments, see the Antivirus Impact section of the Android Studio documentation.

NOTE

Daniel Knoodle has set up a GitHub repo with recommended scripts to add Windows Defender exclusions for Visual Studio 2017.

Additional resources

- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance

- Enable Virtualization support to improve emulator performance

Test on an Android device or emulator

7/14/2021 • 4 minutes to read • Edit Online

There are several ways to test and debug your Android application using a real device or emulator on your Windows machine. We have outlined a few recommendations in this guide.

Run on a real Android device

To run your app on a real Android device, you will first need to enable your Android device for development. Developer options on Android have been hidden by default since version 4.2 and enabling them can vary based on the Android version.

Enable your device for development

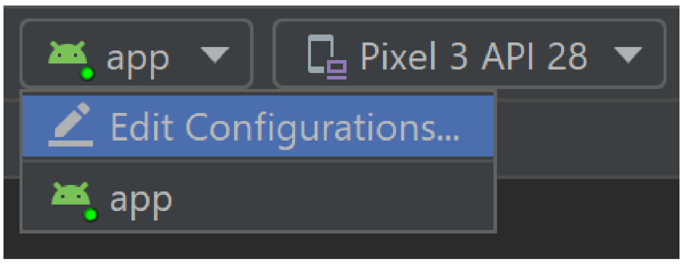
For a device running a recent version of Android 9.0+:

1. Connect your device to your Windows development machine with a USB cable. You may receive a notification to install a USB driver.
2. Open the **Settings** screen on your Android device.
3. Select **About phone**.
4. Scroll to the bottom and tap **Build number** seven times, until **You are now a developer!** is visible.
5. Return to the previous screen, select **System**.
6. Select **Advanced**, scroll to the bottom, and tap **Developer options**.
7. In the **Developer options** window, scroll down to find and enable **USB debugging**.

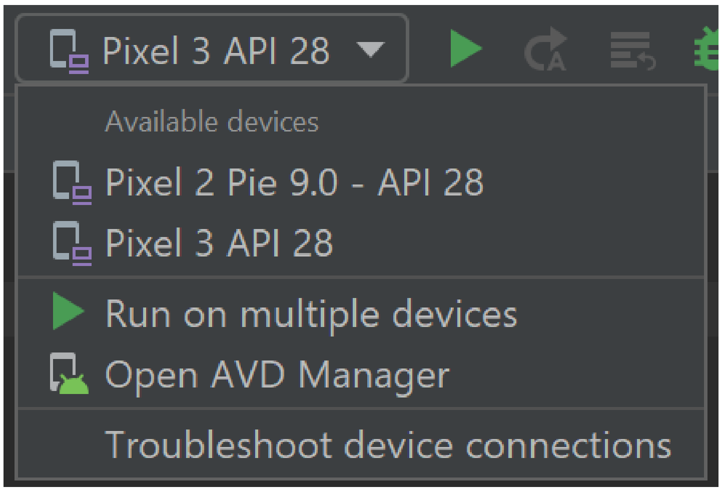
For a device running an older version of Android, see Set Up Device for Development.

Run your app on the device

1. In the Android Studio toolbar, select your app from the **run configurations** drop-down menu.



2. From the **target device** drop-down menu, select the device that you want to run your app on.



3. Select Run ▷. This will launch the app on your connected device.

Run your app on a virtual Android device using an emulator

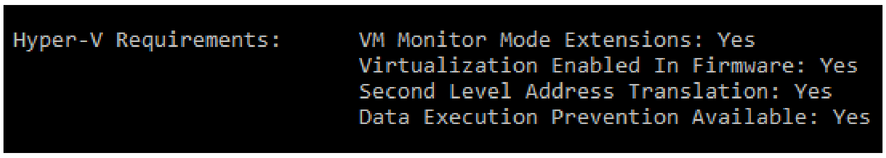
The first thing to know about running an Android emulator on your Windows machine is that regardless of your IDE (Android Studio, Visual Studio, etc), emulator performance is vastly improved by enabling virtualization support.

Enable virtualization support

Before creating a virtual device with the Android emulator, it is recommended that you enable virtualization by turning on the Hyper-V and Windows Hypervisor Platform (WHPX) features. This will allow your computer's processor to significantly improve the execution speed of the emulator.

> To run Hyper-V and Windows Hypervisor Platform, your computer must:
>
> - Have 4GB of memory available
> - Have a 64-bit Intel processor or AMD Ryzen CPU with Second Level Address Translation (SLAT)
> - Be running Windows 10 build 1803+ (Check your build #)
> - Have updated graphics drivers (Device Manager > Display adapters > Update driver)
>
> If your machine doesn't fit this criteria, you may be able to run Intel HAXM or AMD Hypervisor. For more info, see the article: Hardware acceleration for emulator performance or the Android Studio Emulator documentation.

1. Verify that your computer hardware and software is compatible with Hyper-V by opening a command prompt and entering the command: `systeminfo`



```
Hyper-V Requirements:      VM Monitor Mode Extensions: Yes
                           Virtualization Enabled In Firmware: Yes
                           Second Level Address Translation: Yes
                           Data Execution Prevention Available: Yes
```

2. In the Windows search box (lower left), enter "windows features". Select **Turn Windows features on or off** from the search results.

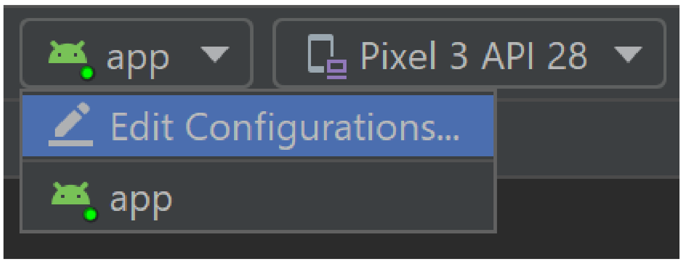
3. Once the **Windows Features** list appears, scroll to find **Hyper-V** (includes both Management Tools and Platform) and **Windows Hypervisor Platform**, ensure that the box is checked to enable both, then select **OK**.

4. Restart your computer when prompted.

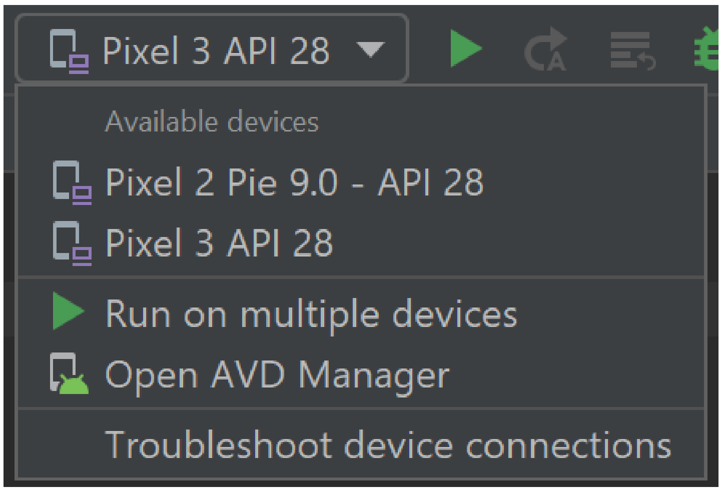
Emulator for native development with Android Studio

When building and testing a native Android app, we recommend using Android Studio. Once your app is ready for testing, you can build and run your app by:

1. In the Android Studio toolbar, select your app from the **run configurations** drop-down menu.



2. From the **target device** drop-down menu, select the device that you want to run your app on.



3. Select Run ▷. This will launch the Android Emulator.

> **TIP**
>
> Once your app is installed on the emulator device, you can use `Apply Changes` to deploy certain code and resource changes without building a new APK. See the Android developer guide for more information.

Emulator for cross-platform development with Visual Studio

There are many Android emulator options available for Windows PCs. We recommend using the Google Android emulator, as it offers access to the latest Android OS images and Google Play services.

Install Android emulator with Visual Studio

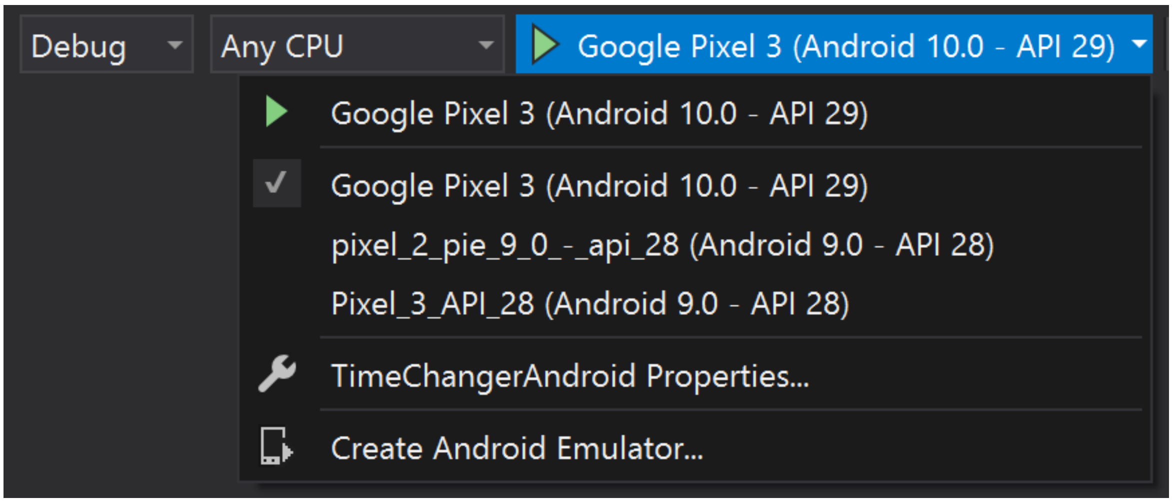
1. If you don't already have it installed, download Visual Studio 2019. Use the Visual Studio Installer to Modify your workloads and ensure that you have the **Mobile development with .NET workload**.

2. Create a new project. Once you've set up the Android Emulator, you can use the Android Device Manager to create, duplicate, customize, and launch a variety of Android virtual devices. Launch the Android Device Manager from the Tools menu with: **Tools** > **Android** > **Android Device Manager**.

3. Once the Android Device Manager opens, select **+ New** to create a new device.

4. You will need to give the device a name, choose the base device type from a drop-down menu, choose a processor, and OS version, along with several other variables for the virtual device. For more information, Android Device Manager Main Screen.

5. In the Visual Studio toolbar, choose between **Debug** (attaches to the application process running inside the emulator after your app starts) or **Release** mode (disables the debugger). Then choose a virtual device from the device drop-down menu and select the **Play** button ▷ to run your application in the emulator.


Debug
Any CPU
Google Pixel 3 (Android 10.0 - API 29)
Google Pixel 3 (Android 10.0 - API 29)
Google Pixel 3 (Android 10.0 - API 29)
pixel_2_pie_9_0_-_api_28 (Android 9.0 - API 28)
Pixel_3_API_28 (Android 9.0 - API 28)
TimeChangerAndroid Properties...
Create Android Emulator...

Additional resources

- Develop Dual-screen apps for Android and get the Surface Duo device SDK
- Add Windows Defender exclusions to improve performance

Overview of Docker remote development on Windows

3/5/2021 • 4 minutes to read • Edit Online

Using containers for remote development and deploying applications with the Docker platform is a very popular solution with many benefits. Learn more about the variety of support offered by Microsoft tools and services, including Windows Subsystem for Linux (WSL), Visual Studio, Visual Studio Code, .NET, and a broad variety of Azure services.

Docker on Windows 10

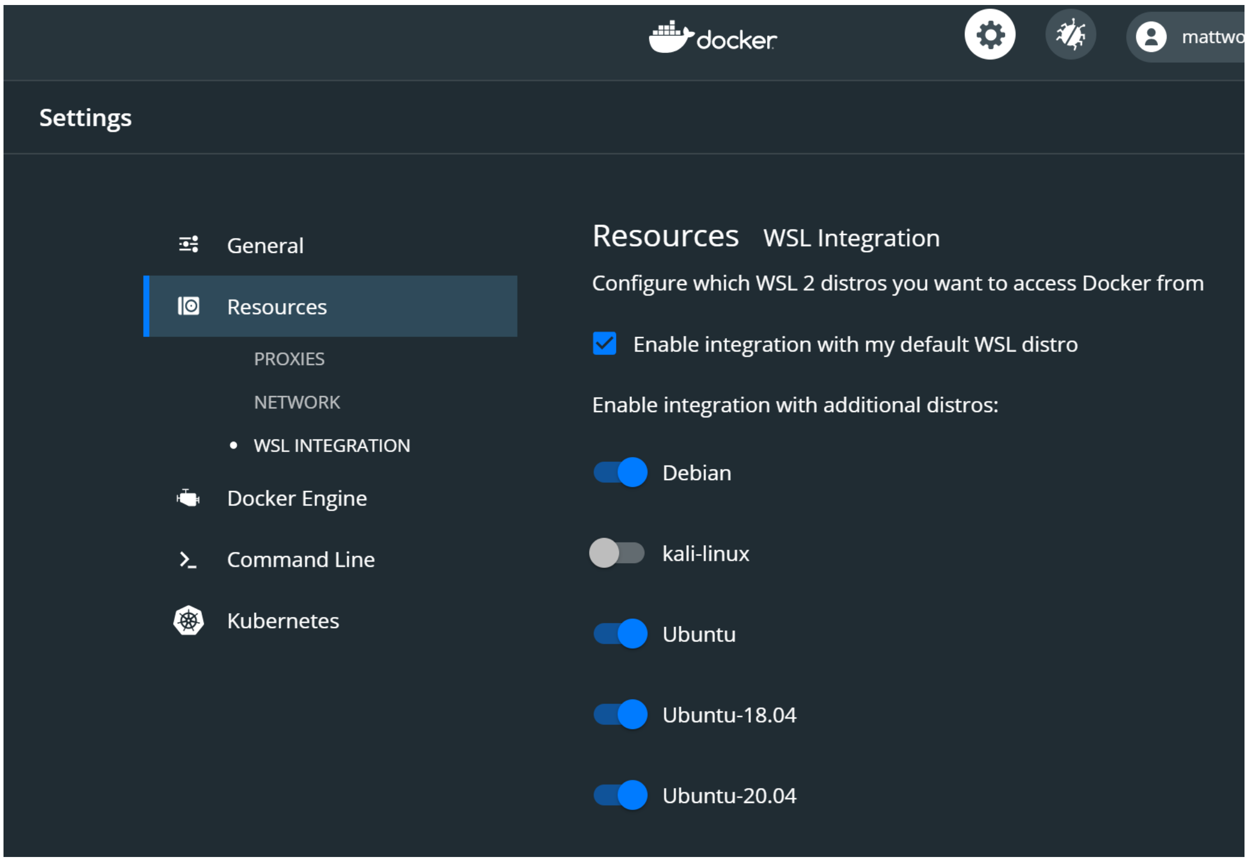
Install Docker Desktop for Windows
Find installation steps, system requirements, what's included in the installer, how to uninstall, differences between stable and edge versions, and how to switch between Windows and Linux containers.

Get started with Docker

Docker orientation and setup docs with step-by-step instructions on how to get started, including a video walk-through.

MS Learn course: Introduction to Docker containers

Microsoft Learn offers a free intro course on Docker containers, in addition to a variety of courses on get started with Docker and connecting with Azure services.



Resources WSL Integration

Configure which WSL 2 distros you want to access Docker from

[✓] Enable integration with my default WSL distro

Enable integration with additional distros:

- [✓] Debian
- [ ] kali-linux
- [✓] Ubuntu
- [✓] Ubuntu-18.04
- [✓] Ubuntu-20.04

Get started with Docker remote containers on WSL 2

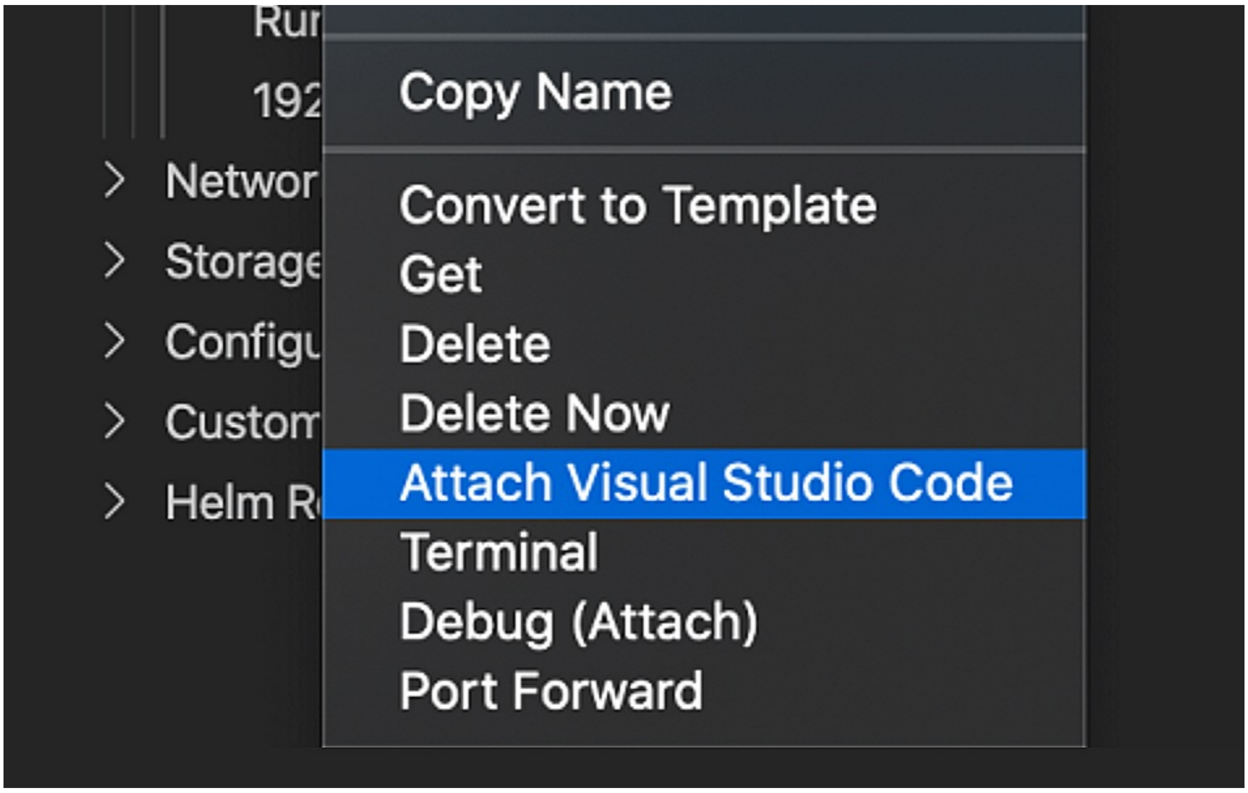
Learn how to set up Docker Desktop for Windows to use with a Linux command line (Ubuntu, Debian, SUSE, etc) using WSL 2 (Windows Subsystem for Linux, version 2).

VS Code and Docker

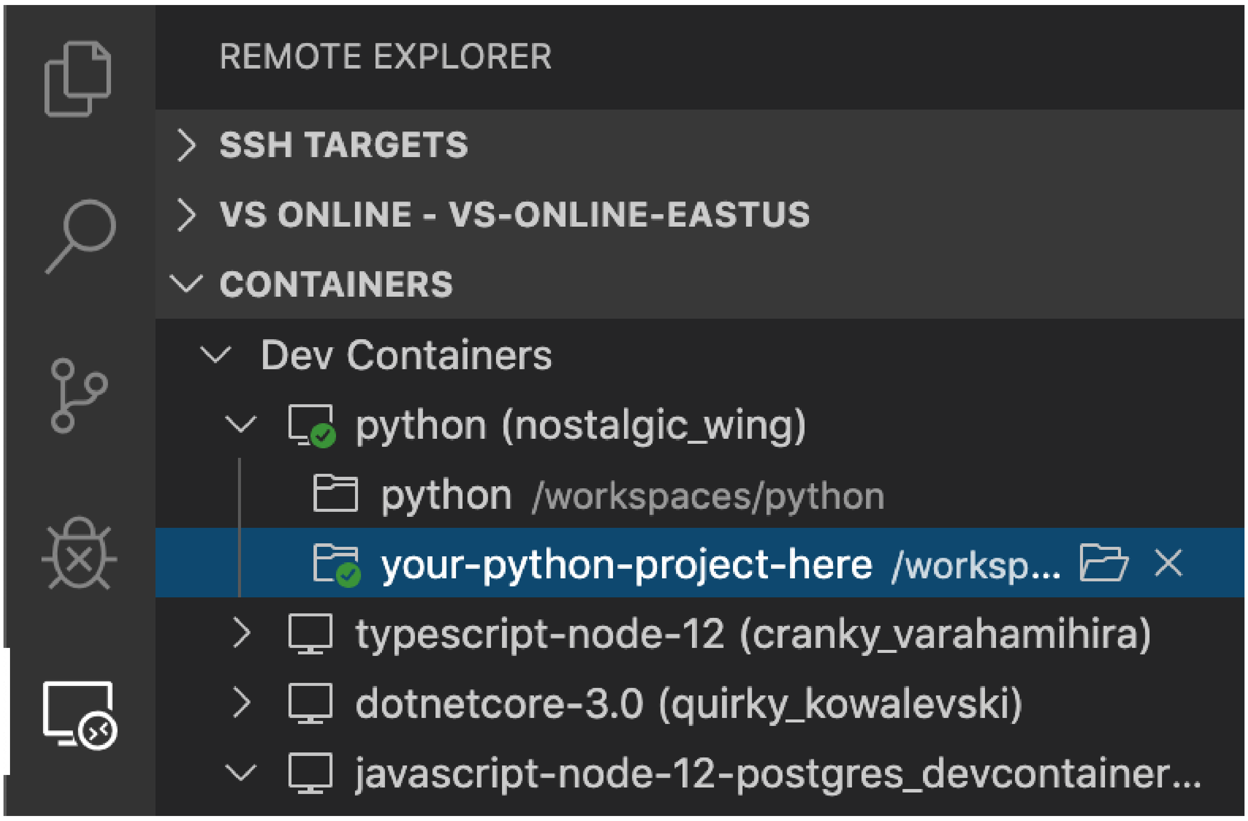
```
#-------------------------------------------------------------
# Copyright (c) Microsoft Corporation. All rights rese
# Licensed under the MIT License. See https://go.micros
#-------------------------------------------------------------

FROM rust:1

# Copy endpoint specific user setting overrides into co
COPY settings.vscode.json /root/.vscode-remote/data/Mac

RUN rustup update
RUN rustup component add rls rust-analysis rust-src
```

Create a Docker container with VS Code

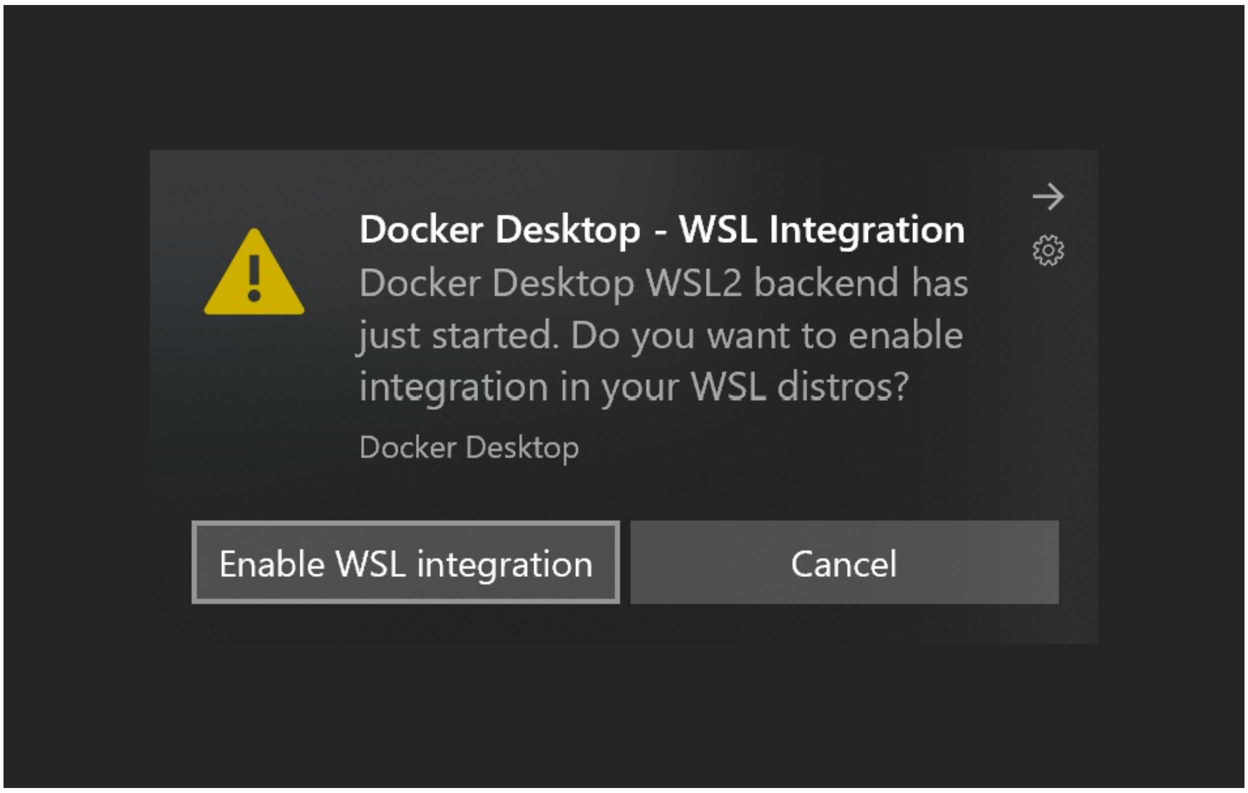
Set up a full-featured dev environment inside of a container with the Remote - Containers extension and find tutorials to set up a NodeJS container, a Python container, or a ASP.NET Core container.



Attach VS Code to a Docker container
Learn how to attach Visual Studio Code to a Docker container that is already running or to a container in a Kubernetes cluster.


REMOTE EXPLORER
SSH TARGETS
VS ONLINE - VS-ONLINE-EASTUS
CONTAINERS
Dev Containers
python (nostalgic_wing)
python /workspaces/python
your-python-project-here /worksp...
typescript-node-12 (cranky_varahamihira)
dotnetcore-3.0 (quirky_kowalevski)
javascript-node-12-postgres_devcontainer...

Advanced Container Configuration
Learn about advanced setup scenarios for using Docker containers with Visual Studio Code or read this article on how to Inspect Containers for debugging with VS Code.


Docker Desktop - WSL Integration

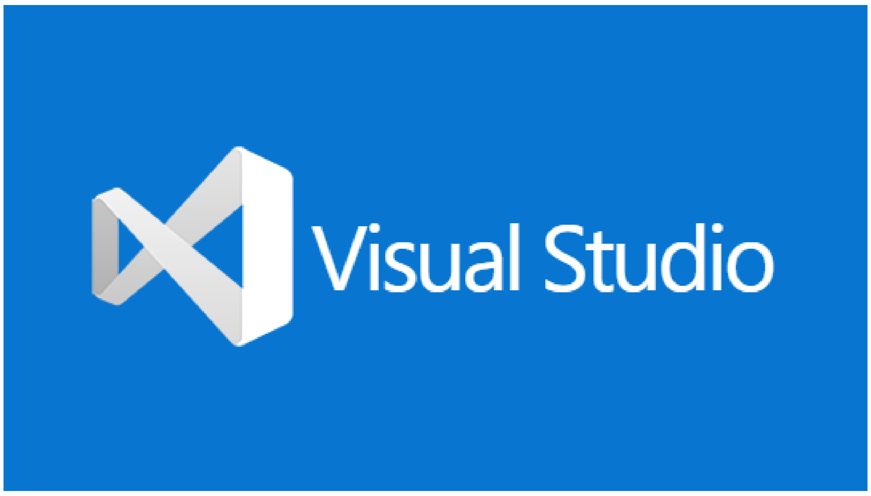
Docker Desktop WSL2 backend has just started. Do you want to enable integration in your WSL distros?

Docker Desktop

Using Remote Containers in WSL 2

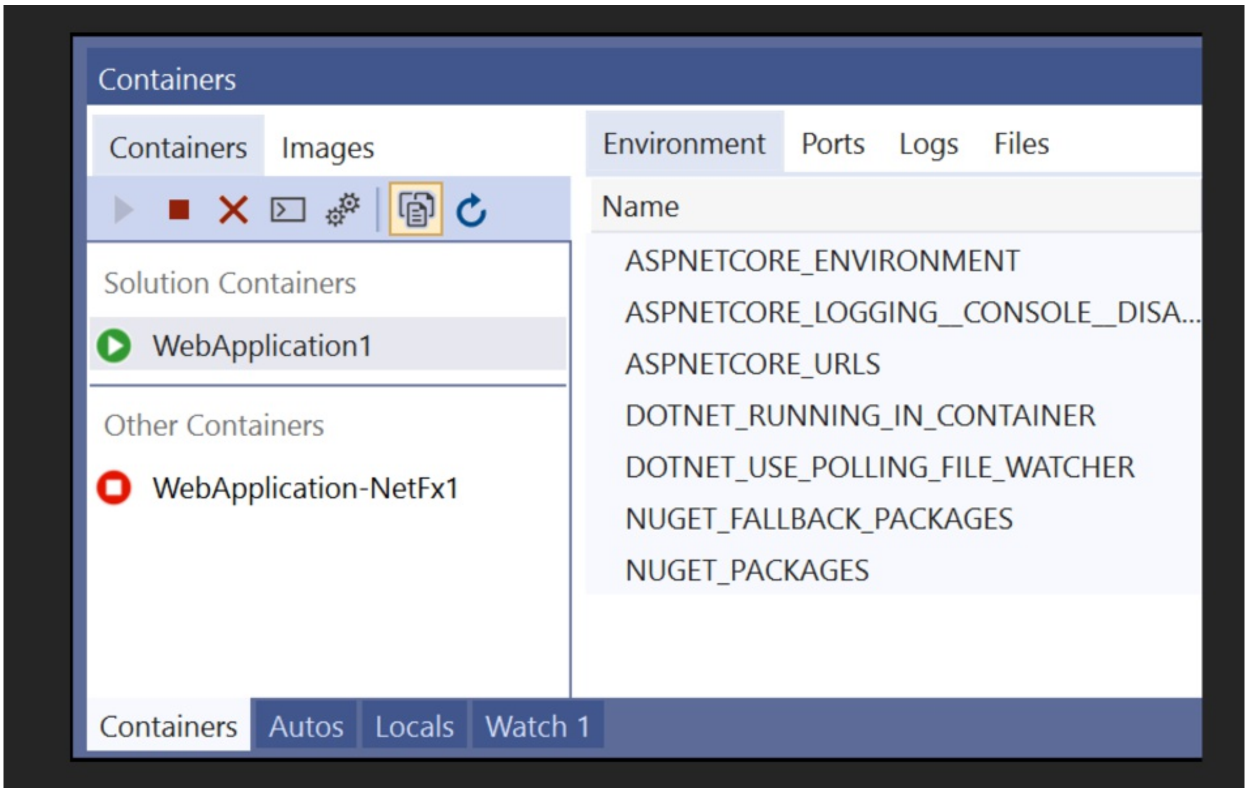
Read about using Docker containers with WSL 2 (Windows Subsystem for Linux, version 2) and how to set everything up with VS Code. You can also read about how it works.

Visual Studio and Docker



Docker support in Visual Studio

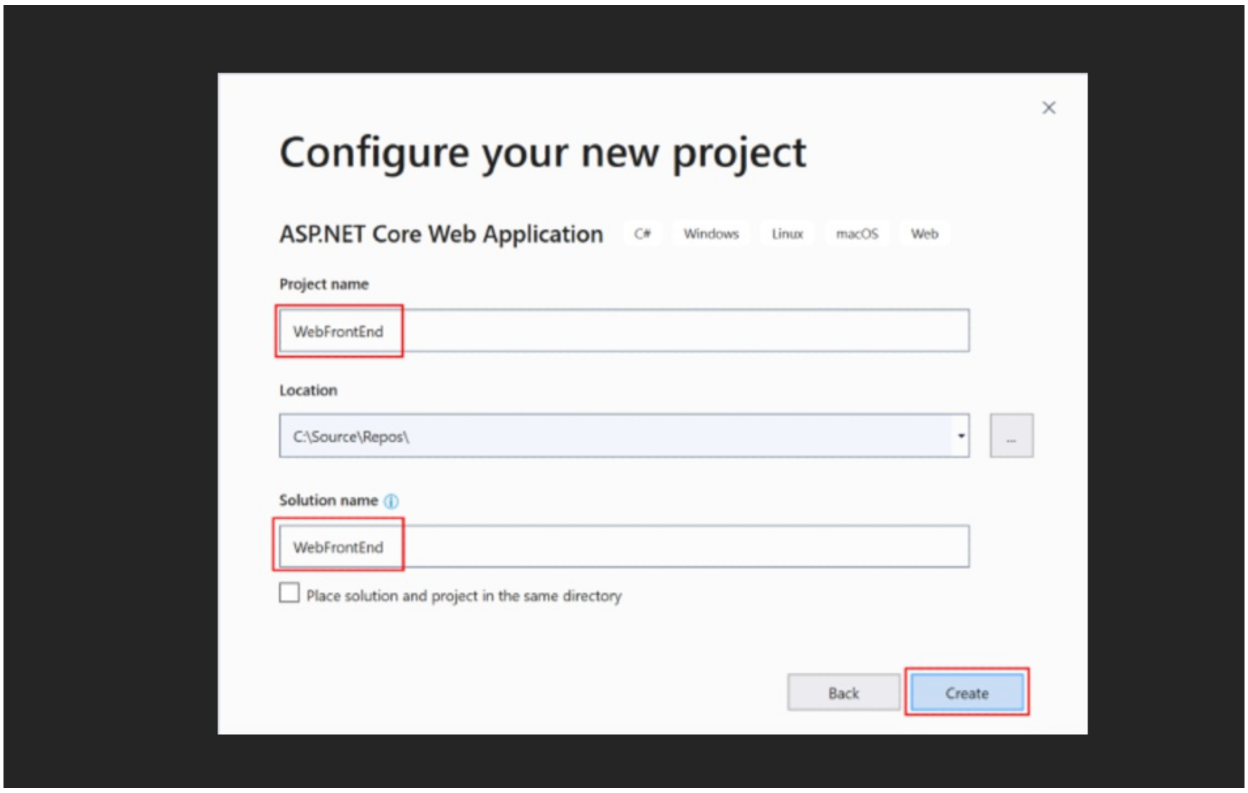
Learn about the Docker support available for ASP.NET projects, ASP.NET Core projects, and .NET Core and .NET Framework console projects in Visual Studio, in addition to support for container orchestration.



ASPNETCORE_ENVIRONMENT
ASPNETCORE_LOGGING__CONSOLE__DISA...
ASPNETCORE_URLS
DOTNET_RUNNING_IN_CONTAINER
DOTNET_USE_POLLING_FILE_WATCHER
NUGET_FALLBACK_PACKAGES
NUGET_PACKAGES

Quickstart: Docker in Visual Studio

Learn how to build, debug, and run containerized .NET, ASP.NET, and ASP.NET Core apps and publish them to Azure Container Registry (ACR), Docker Hub, Azure App Service, or your own container registry with Visual Studio.



Project name: WebFrontEnd

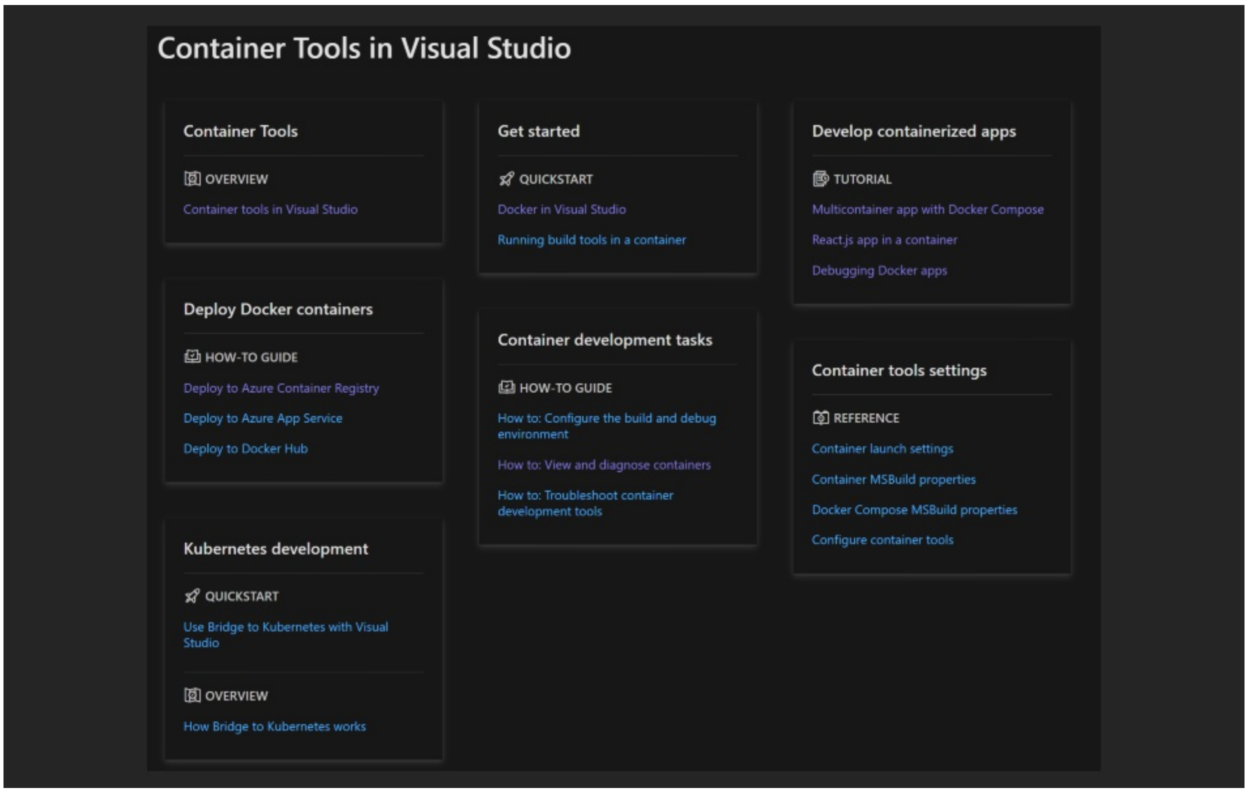
Location: C:\Source\Repos\

Solution name: WebFrontEnd

☐ Place solution and project in the same directory

Tutorial: Create a multi-container app with Docker Compose

Learn how to manage more than one container and communicate between them when using Container Tools in Visual Studio. You can also find links to tutorials like how to Use Docker with a React Single-page App.



Container Tools in Visual Studio

Container Tools
OVERVIEW
Container tools in Visual Studio

Get started
QUICKSTART
Docker in Visual Studio
Running build tools in a container

Develop containerized apps
TUTORIAL
Multicontainer app with Docker Compose
React.js app in a container
Debugging Docker apps

Deploy Docker containers
HOW-TO GUIDE
Deploy to Azure Container Registry
Deploy to Azure App Service
Deploy to Docker Hub

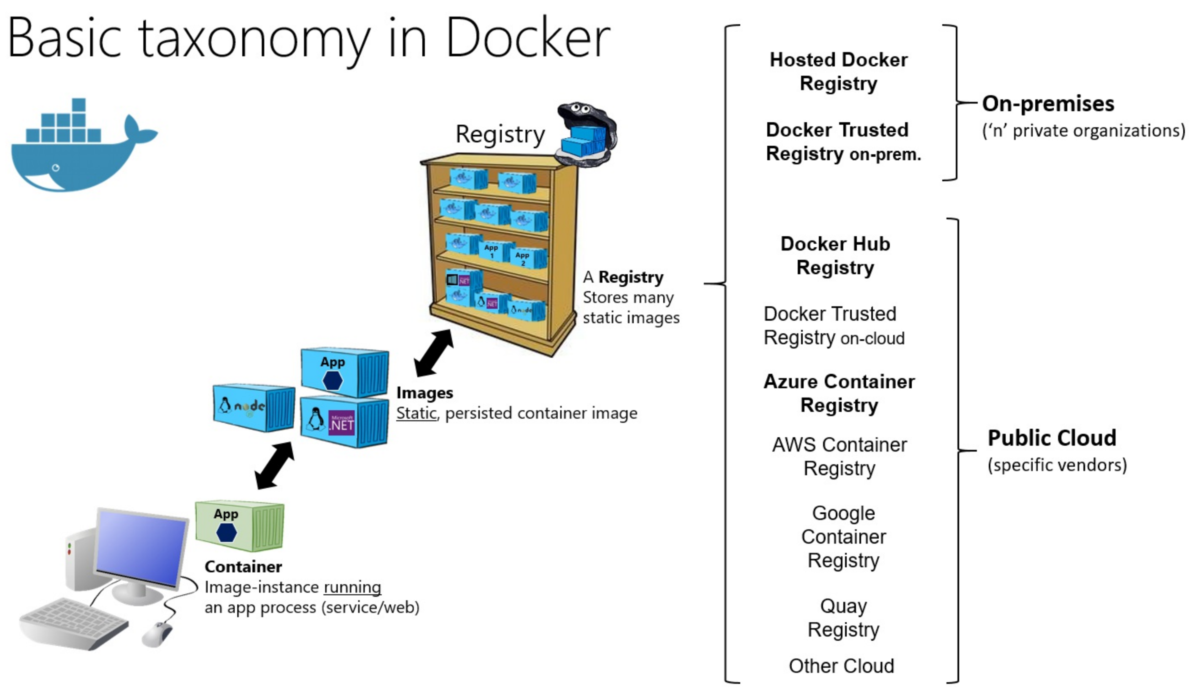
Container development tasks
HOW-TO GUIDE
How to: Configure the build and debug environment
How to: View and diagnose containers
How to: Troubleshoot container development tools

Container tools settings
REFERENCE
Container launch settings
Container MSBuild properties
Docker Compose MSBuild properties
Configure container tools

Kubernetes development
QUICKSTART
Use Bridge to Kubernetes with Visual Studio
OVERVIEW
How Bridge to Kubernetes works

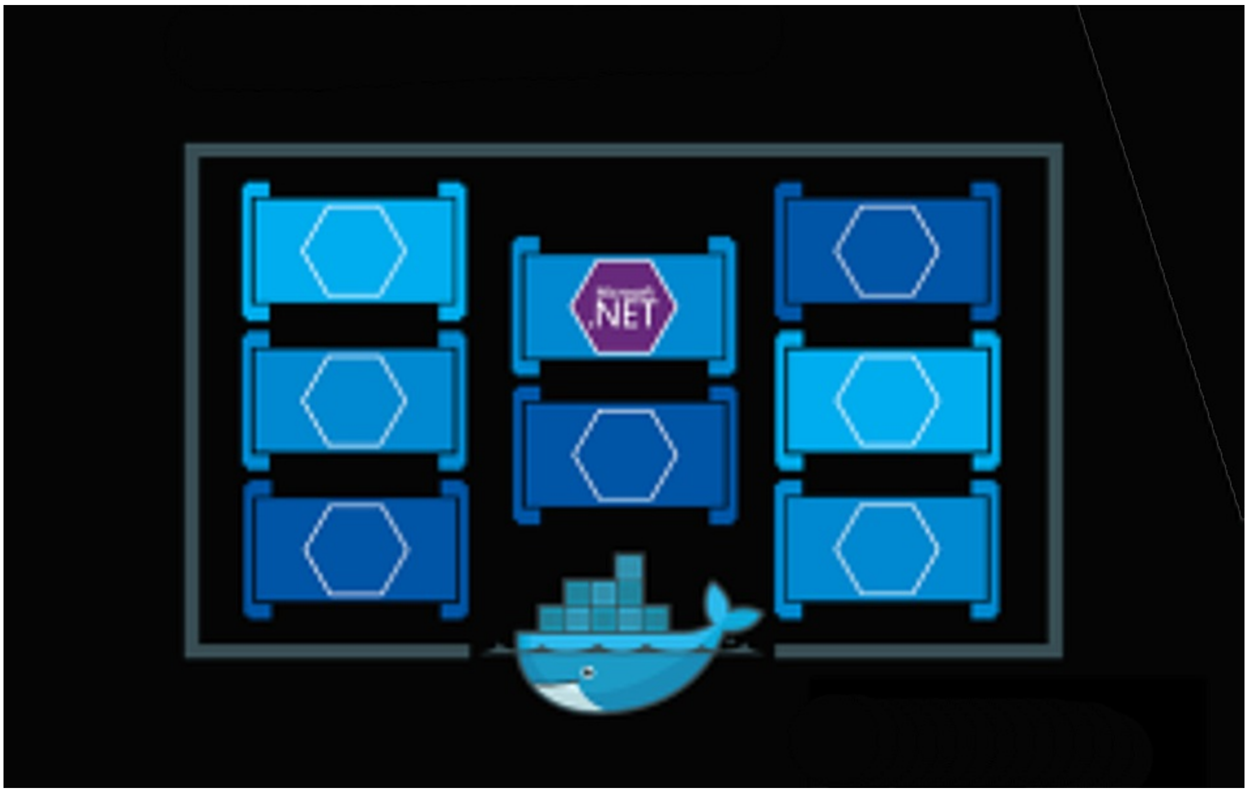
Container Tools in Visual Studio

Find topics covering how to run build tools in a container, debugging Docker apps, troubleshoot development tools, deploy Docker containers, and bridge Kubernetes with Visual Studio.



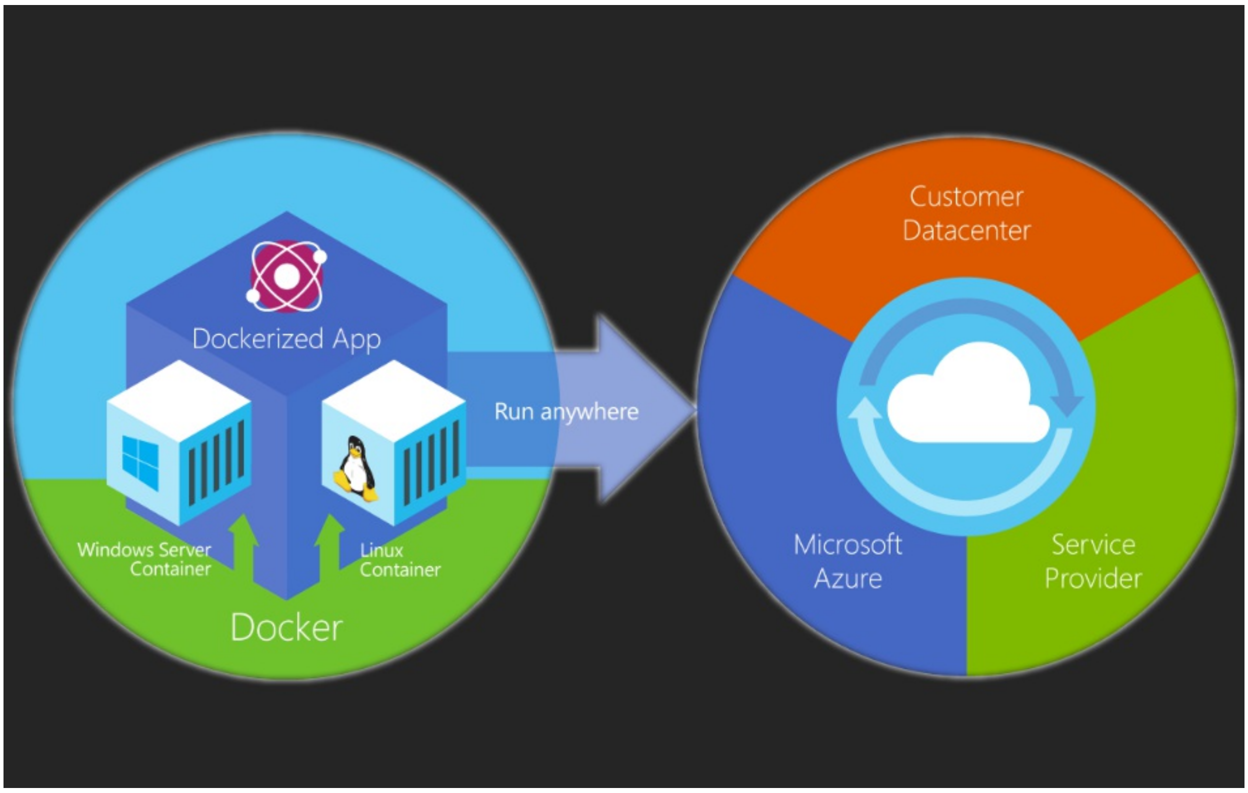
Basic taxonomy in Docker
Registry
A Registry
Stores many static images
App
node
.NET
Images
Static, persisted container image
App
Container
Image-instance running an app process (service/web)
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises
('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud
(specific vendors)

.NET Core and Docker



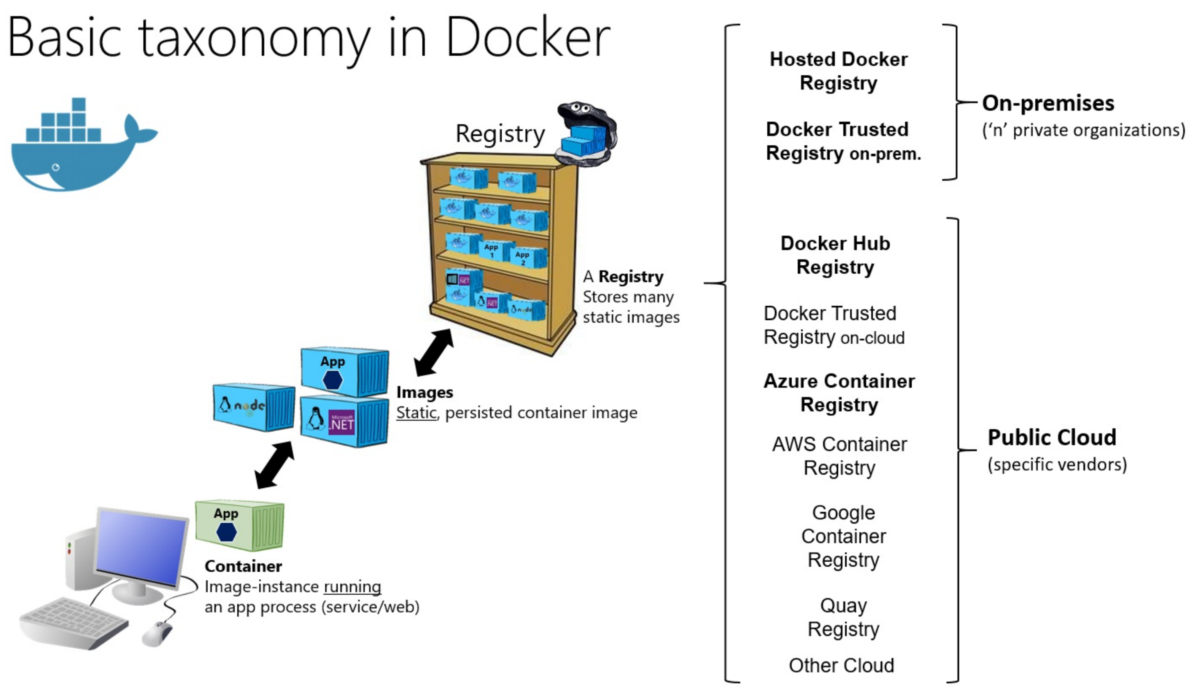
.NET Guide: Microservice apps and containers

Intro guide to microservices-based apps managed with containers.



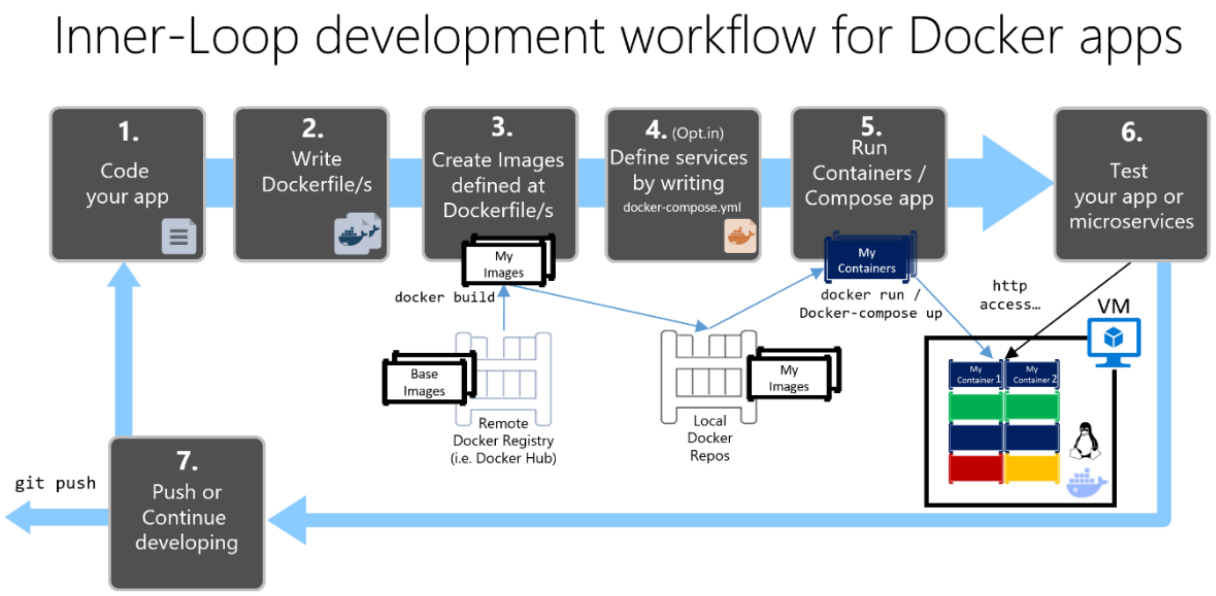
What is Docker?

Basic explanation of Docker containers, including Comparing Docker containers with Virtual machines and a basic taxonomy of Docker terms and concepts explaining the difference between containers, images, and registries.


Basic taxonomy in Docker
Registry
A Registry Stores many static images
Images Static, persisted container image
App
node
.NET
Container Image-instance running an app process (service/web)
Hosted Docker Registry
Docker Trusted Registry on-prem.
On-premises ('n' private organizations)
Docker Hub Registry
Docker Trusted Registry on-cloud
Azure Container Registry
AWS Container Registry
Google Container Registry
Quay Registry
Other Cloud
Public Cloud (specific vendors)

Tutorial: Containerize a .NET Core app

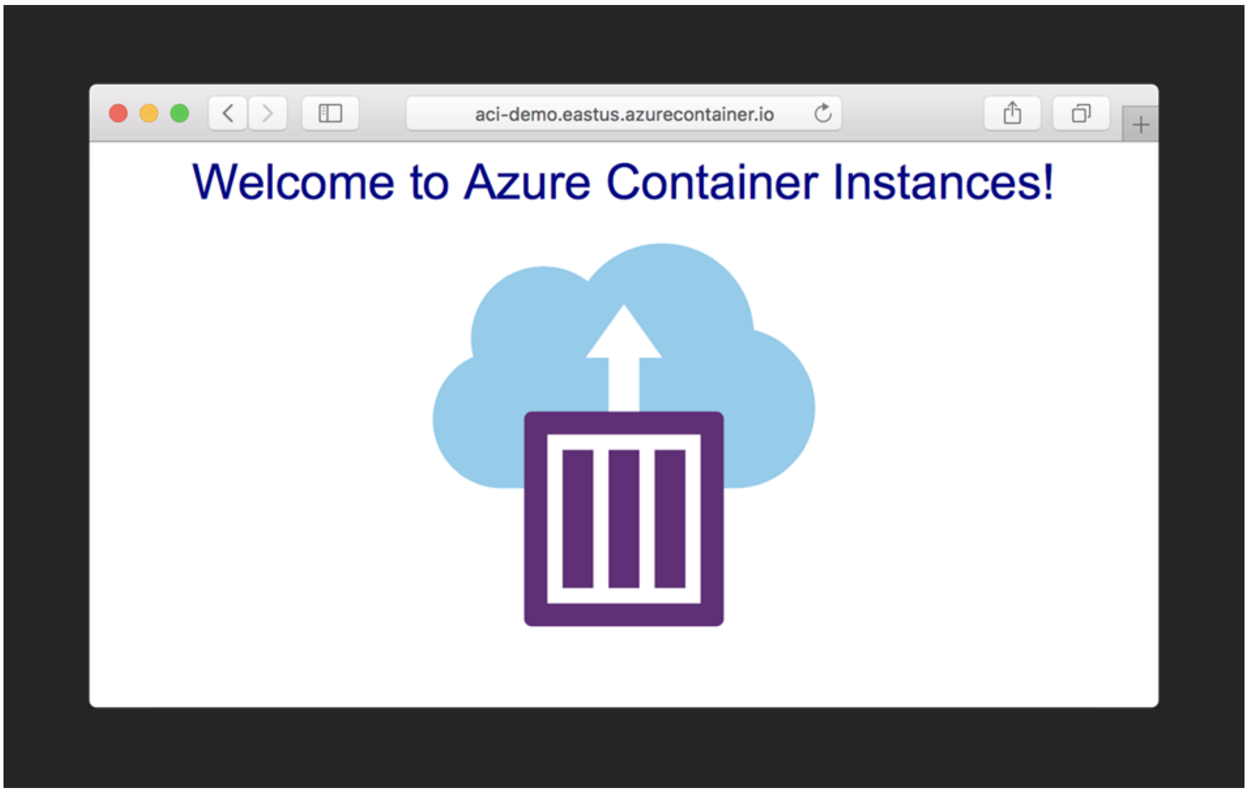
Learn how to containerize a .NET Core application with Docker, including creation of a Dockerfile, essential commands, and cleaning up resources.


Inner-Loop development workflow for Docker apps
1. Code your app
2. Write Dockerfile/s
3. Create Images defined at Dockerfile/s
4. (Opt.in) Define services by writing docker-compose.yml
5. Run Containers / Compose app
6. Test your app or microservices
7. Push or Continue developing
My Images
docker build
Base Images
Remote Docker Registry (i.e. Docker Hub)
Local Docker Repos
My Containers
docker run / Docker-compose up
http access...
VM
My Container 1
My Container 2
git push

Development workflow for Docker apps

Describes the inner-loop development workflow for Docker container-based applications.

Azure Container Services

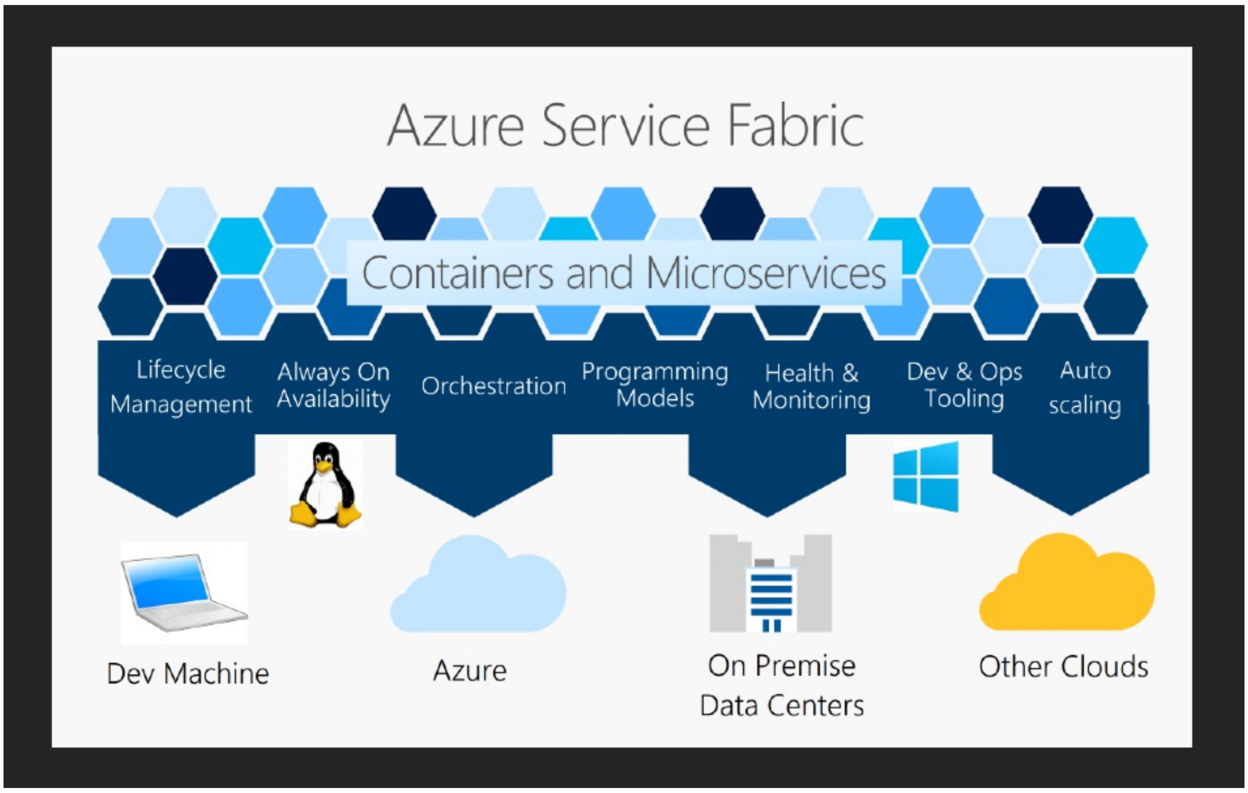

aci-demo.eastus.azurecontainer.io
Welcome to Azure Container Instances!

Azure Container Instances

Learn how to run Docker containers on-demand in a managed, serverless Azure environment, includes ways to deploy with Docker CLI, ARM, Azure Portal, create multi-container groups, share data between containers, connect to a virtual network, and more.

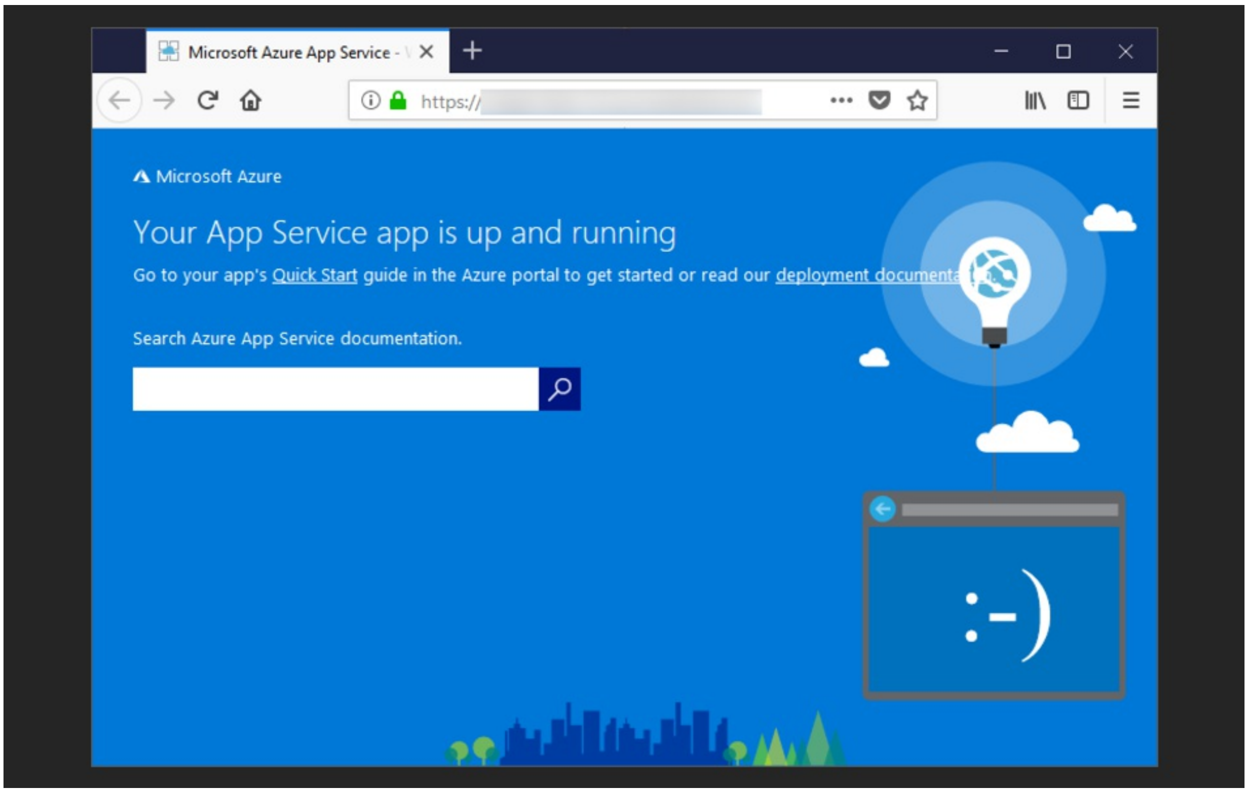
Azure Container Registry

Learn how to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Create Azure container registries for your existing container development and deployment pipelines, set up automation tasks, and learn how to manage your registries, including geo-replication and best practices.



Azure Service Fabric

Learn about Azure Service Fabric, a distributed systems platform for packaging, deploying, and managing scalable and reliable microservices and containers.



Your App Service app is up and running

Go to your app's Quick Start guide in the Azure portal to get started or read our deployment documenta

Search Azure App Service documentation.

Azure App Service

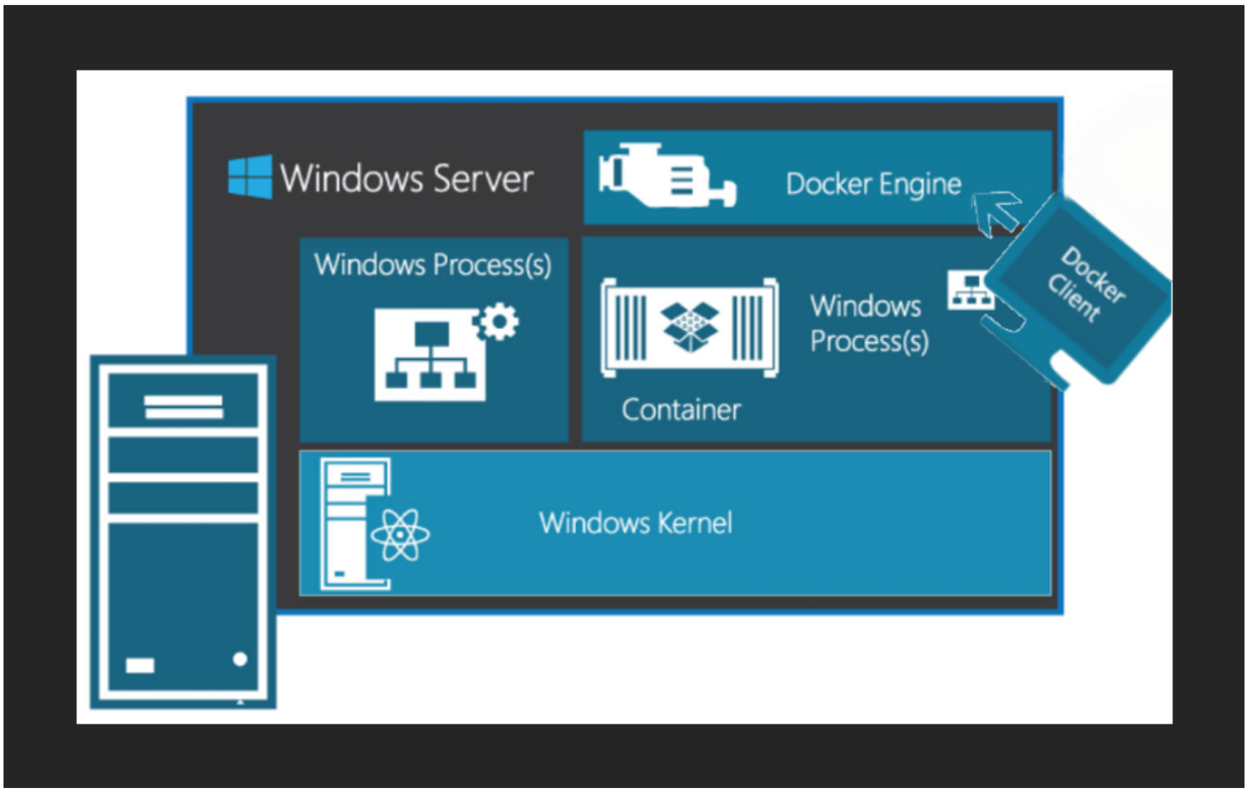
Learn how to build and host web apps, mobile back ends, and RESTful APIs in the programming language of your choice without managing infrastructure. Try the Azure App Service course on MS Learn to deploy a web app based on a Docker image and configure continuous deployment.

Learn about more Azure services that support containers.

Docker Containers Explainer Video

Kubernetes and Container Orchestration Explainer Video

Containers on Windows

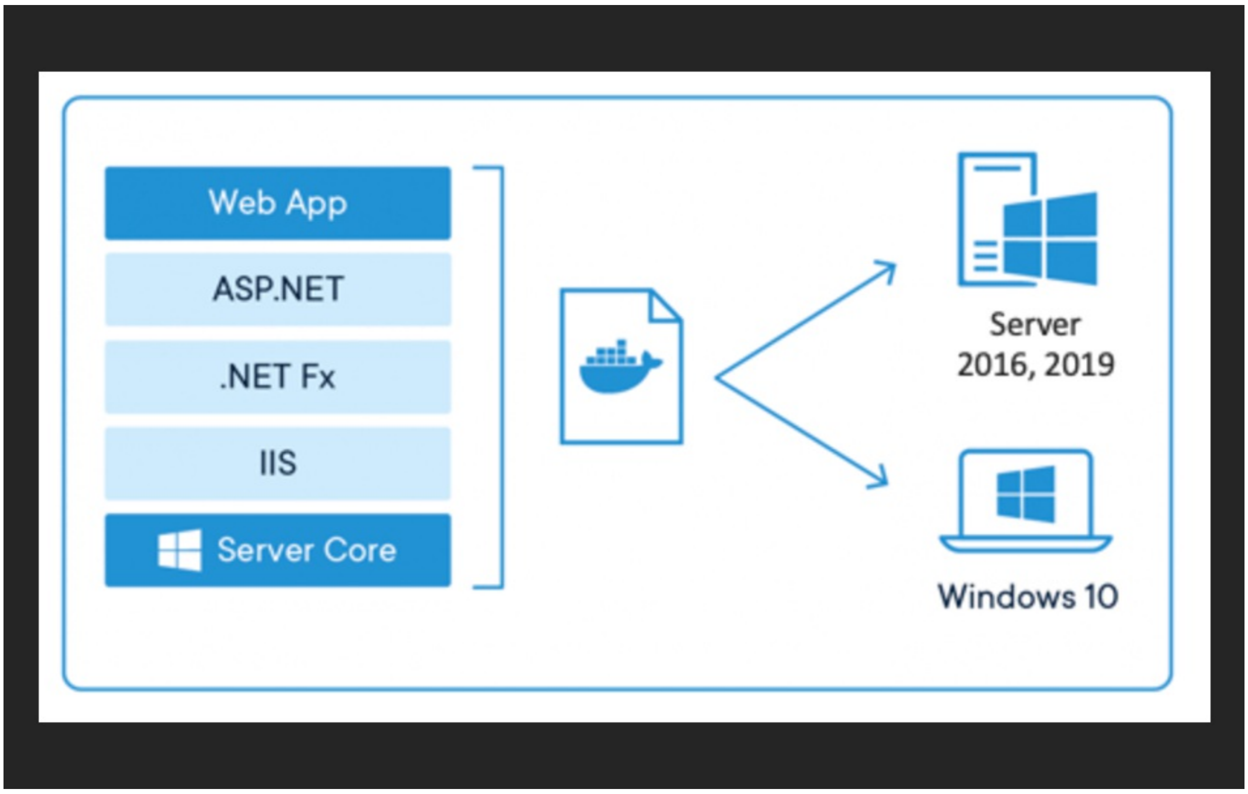


Containers on Windows docs

Package apps with their dependencies and leverage operating system-level virtualization for fast, fully isolated environments on a single system. Learn about Windows containers, including quick starts, deployment guides, and samples.

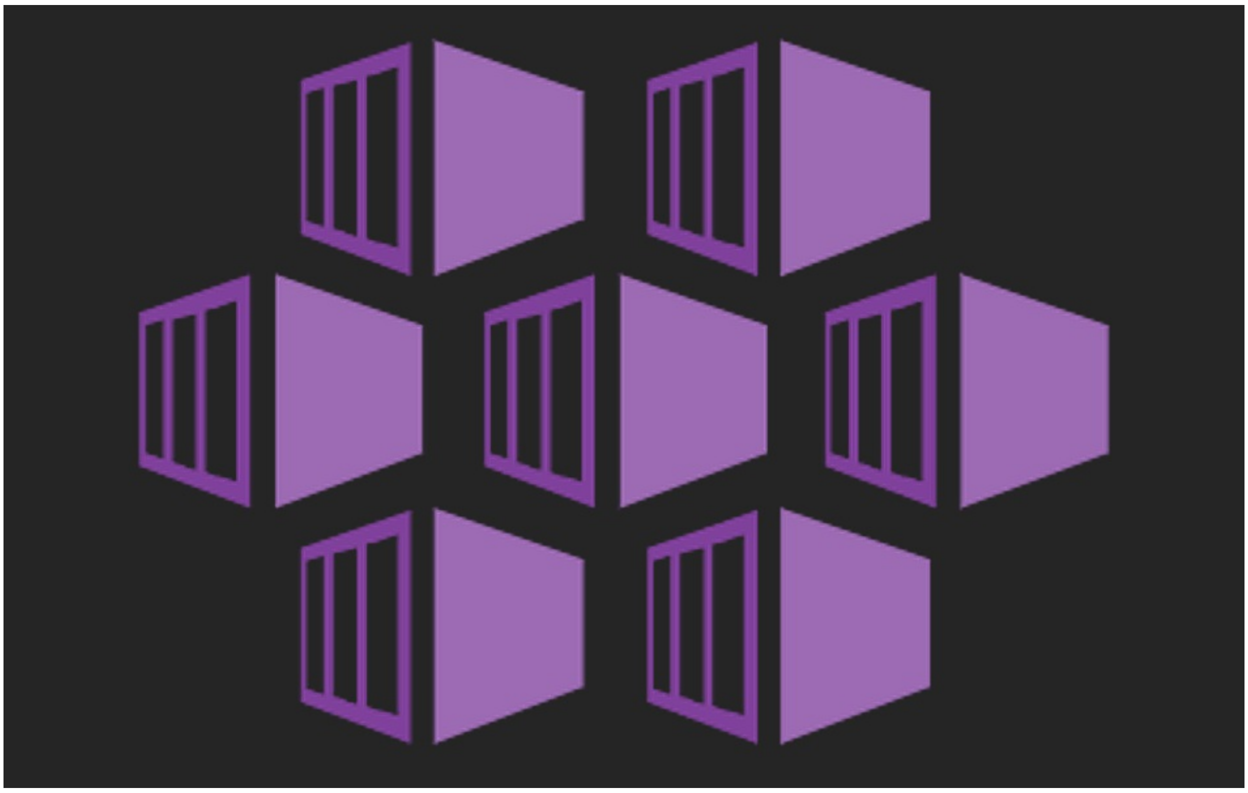
FAQs about Windows containers

Find frequently asked questions about containers. Also see this explanation in StackOverflow on "What's the difference between Docker for Windows and Docker on Windows?"



Set up your environment

Learn how to set up Windows 10 or Windows Server to create, run, and deploy containers, including prerequisites, installing Docker, and working with Windows Container Base Images.



Create a Windows Server container on an Azure Kubernetes Service (AKS)

Learn how to deploy an ASP.NET sample app in a Windows Server container to an AKS cluster using the Azure CLI.

Overview of developing on Windows with Rust

6/2/2021 • 2 minutes to read • Edit Online

It's not hard to get started with Rust. If you're a beginner who's interested in learning Rust using Windows 10, then we recommend that you follow each detail of this step-by-step guide. It shows you what to install, and how to set up your development evironment.

TIP
If you're already sold on Rust and you have your Rust environment already set up, and you just want to start calling Windows APIs, then feel free to jump forward to the Rust for Windows, and the windows crate topic.

What is Rust?

Rust is a systems programming language, so it's used for writing systems (such as operating systems). But it can also be used for applications where performance and trustworthiness are important. The Rust language syntax is comparable to that of C++, provides performance on par with modern C++, and for many experienced developers Rust hits all the right notes when it comes to compilation and runtime model, type system, and deterministic finalization.

In addition, Rust is designed around the promise of guaranteed memory safety, without the need for garbage collection.

So why did we choose Rust for the latest language projection for Windows? One factor is that Stack Overflow's annual developer survey shows Rust to be the best-loved programming language by far, year after year. While you might find that the language has a steep learning curve, once you're over the hump it's hard not to fall in love.

Furthermore, Microsoft is a founding member of the Rust Foundation. The Foundation is an independent non-profit organization, with a new approach to sustaining and growing a large, participatory, open source ecosystem.

The pieces of the Rust development toolset/ecosystem

We'll introduce some Rust tools and terms in this section. You can refer back here to refresh yourself on any of the descriptions.

- A *crate* is a Rust unit of compilation and linking. A crate can exist in source code form, and from there it can be processed into a crate in the form of either a binary executable (*binary* for short), or a binary library (*library* for short).
- A Rust project is known as a *package*. A package contains one or more crates, together with a `Cargo.toml` file that describes how to build those crates.
- `rustup` is the installer and updater for the Rust toolchain.
- *Cargo* is the name of Rust's package management tool.
- `rustc` is the compiler for Rust. Most of the time, you won't invoke `rustc` directly; you'll invoke it indirectly via Cargo.
- *crates.io* (`https://crates.io/`) is the Rust community's crate registry.

Setting up your development environment

In the next topic, we'll see how to set up your dev environment on Windows for Rust.

Related

- The Rust website
- Rust for Windows, and the windows crate
- Stack Overflow annual developer survey
- Rust Foundation
- crates.io
- Set up your dev environment on Windows for Rust

Set up your dev environment on Windows for Rust

5/4/2021 • 6 minutes to read • Edit Online

In the Overview of developing on Windows with Rust topic, we introduced Rust and talked about what it is and what some of its main moving parts are. In this topic, we'll set up our development environment.

We recommend that you do your Rust development on Windows. However, if you plan to locally compile and test on Linux, then developing with Rust on the Windows Subsystem for Linux (WSL) is also an option.

Install Visual Studio (recommended) or the Microsoft C++ Build Tools

On Windows, Rust requires certain C++ build tools.

You can either download the Microsoft C++ Build Tools, or (recommended) you might prefer just to install Microsoft Visual Studio.

IMPORTANT

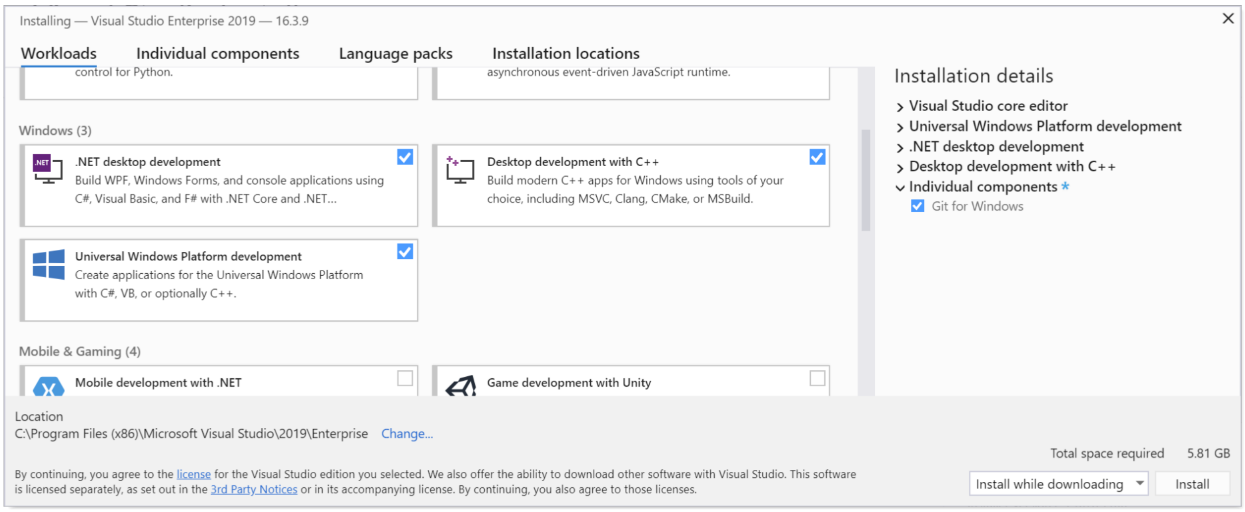
Use of the Microsoft C++ Build Tools, or Visual Studio Build Tools, requires a valid Visual Studio license (either Community, Pro, or Enterprise).

NOTE

We'll be using Visual Studio Code as our integrated development environment (IDE) for Rust, and not Visual Studio. But you can still install Visual Studio without expense. A Community edition is available—it's free for students, open-source contributors, and individuals.

While installing Visual Studio, there are several Windows workloads that we recommend you select—**.NET desktop development**, **Desktop development with C++**, and **Universal Windows Platform development**. You might not think that you'll need all three, but it's likely enough that some dependency will arise where they're required that we feel it's just simpler to select all three.

New Rust projects default to using Git. So also add the individual component **Git for Windows** to the mix (use the search box to search for it by name).


Installing — Visual Studio Enterprise 2019 — 16.3.9

Workloads | Individual components | Language packs | Installation locations

control for Python.

asynchronous event-driven JavaScript runtime.

Windows (3)

[✓] .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...

[✓] Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

[✓] Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Mobile & Gaming (4)

[ ] Mobile development with .NET

[ ] Game development with Unity

Installation details
- Visual Studio core editor
- Universal Windows Platform development
- .NET desktop development
- Desktop development with C++
- Individual components *
  - [✓] Git for Windows

Location
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise Change...

Total space required 5.81 GB

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Install while downloading | Install

Install Rust

Next, install Rust from the Rust website. The website detects that you're running Windows, and it offers you 64- and 32-bit installers of the `rustup` tool for Windows, as well as instructions on installing Rust to the Windows Subsystem for Linux (WSL).

> **TIP**
>
> Rust works very well on Windows; so there's no need for you to go the WSL route (unless you plan to locally compile and test on Linux). Since you have Windows, we recommend that you just run the `rustup` installer for 64-bit Windows. You'll then be all set to write apps *for* Windows using Rust.

When the Rust installer is finished, you'll be ready to program with Rust. You won't have a convenient IDE yet (we'll cover that in the next section—Install Visual Studio Code). And you're not yet set up to call Windows APIs. But you could launch a command prompt (`cmd.exe`), and perhaps issue the command `cargo --version`. If you see a version number printed, then that confirms that Rust installed correctly.

If you're curious about the use of the `cargo` keyword above, *Cargo* is the name of the tool in the Rust development environment that manages and builds your projects (more properly, *packages*) and their dependencies.

And if you really do want to dive in to some programming at this point (even without the convenience of an IDE), then you could read the Hello, World! chapter of the The Rust Programming Language book on the Rust website.

Install Visual Studio Code

By using Visual Studio Code (VS Code) as your text editor/integrated development environment (IDE), you can take advantage of language services such as code completion, syntax highlighting, formatting, and debugging.

VS Code also contains a built-in terminal that enables you to issue command-line arguments (to issue commands to Cargo, for example).

1. First, download and install Visual Studio Code for Windows.

2. After you've installed VS Code, install the **rust-analyzer** *extension*. You can either install the rust-analyzer extension from the Visual Studio Marketplace, or you can open VS Code, and search for **rust-analyzer** in the extensions menu (Ctrl+Shift+X).

3. For debugging support, install the **CodeLLDB** extension. You can either install the CodeLLDB extension from the Visual Studio Marketplace, or you can open VS Code, and search for **CodeLLDB** in the extensions menu (Ctrl+Shift+X).

   > **NOTE**
   >
   > An alternative to the **CodeLLDB** extension for debugging support is the Microsoft **C/C++** extenson. The **C/C++** extension doesn't integrate as well with the IDE as **CodeLLDB** does. But the **C/C++** extension provides superior debugging information. So you might want to have that standing by in case you need it.
   >
   > You can either install the C/C++ extension from the Visual Studio Marketplace, or you can open VS Code, and search for **C/C++** in the extensions menu (Ctrl+Shift+X).

4. If you want to open the terminal in VS Code, select **View** > **Terminal**, or alternatively use the shortcut **Ctrl+`** (using the backtick character). The default terminal is PowerShell.

Hello, world! tutorial (Rust with VS Code)

Let's take Rust for a spin with a simple "Hello, world!" app.

1. First, launch a command prompt (`cmd.exe`), and `cd` to a folder where you want to keep your Rust projects.

2. Then ask Cargo to create a new Rust project for you with the following command.

   ```
   cargo new first_rust_project
   ```

   The argument you pass to the `cargo new` command is the name of the project that you want Cargo to create. Here, the project name is *first_rust_project*. The recommendation is that you name your Rust projects using snake case (where words are lower-case, with each space replaced by an underscore).

   Cargo creates a project for you with the name that you supply. And in fact Cargo's new projects contain the source code for a very simple app that outputs a *Hello, world!* message, as we'll see. In addition to creating the *first_rust_project* project, Cargo has created a folder named *first_rust_project*, and has put the project's source code files in there.

3. So now `cd` into that folder, and then launch VS Code from within the context of that folder.

   ```
   cd first_rust_project
   code .
   ```

4. In VS Code's Explorer, open the `src` > `main.rs` file, which is the Rust source code file that contains your app's entry point (a function named **main**). Here's what it looks like.

   ```
   // main.rs
   fn main() {
     println!("Hello, world!");
   }
   ```

   NOTE

   When you open the first `.rs` file in VS Code, you'll get a notification saying that some Rust components aren't installed, and asking whether you want to install them. Click **Yes**, and VS Code will install the Rust language server.

   You can tell from glancing at the code in `main.rs` that **main** is a function definition, and that it prints the string "Hello, world!". For more details about the syntax, see Anatomy of a Rust Program on the Rust website.

5. Now let's try running the app under the debugger. Put a breakpoint on line 2, and click **Run** > **Start Debugging** (or press **F5**). There are also **Debug** and **Run** commands embedded inside the text editor.

   NOTE

   When you run an app under the debugger for the first time, you'll see a dialog box saying "Cannot start debugging because no launch configuration has been provided". Click **OK** to see a second dialog box saying "Cargo.toml has been detected in this workspace. Would you like to generate launch configurations for its targets?". Click **Yes**. Then close the launch.json file and begin debugging again.

6. As you can see, the debugger breaks at line 2. Press **F5** to continue, and the app runs to completion. In the **Terminal** pane, you'll see the expected output "Hello, world!".

Rust for Windows

Not only can you use Rust *on* Windows, you can also write apps *for* Windows using Rust. Via the *windows* crate, you can call any Windows API past, present, and future. There are more details about that, and code examples, in the Rust for Windows, and the windows crate topic.

Related

- Rust for Windows, and the windows crate
- Windows Subsystem for Linux (WSL)
- Microsoft C++ Build Tools
- Microsoft Visual Studio
- Visual Studio Code for Windows
- rust-analyzer extension
- CodeLLDB extension
- C/C++ extension

Rust for Windows, and the *windows* crate

7/12/2021 • 2 minutes to read • Edit Online

Introducing Rust for Windows

In the Overview of developing on Windows with Rust topic, we demonstrated a simple app that outputs a *Hello, world!* message. But not only can you use Rust *on* Windows, you can also write apps *for* Windows using Rust.

Rust for Windows is the latest language projection for Windows. It's currently in preview form, and you can see it develop from version to version in its change log.

Rust for Windows lets you use any Windows API (past, present, and future) directly and seamlessly via the *windows* crate (*crate* is Rust's term for a binary or a library, and/or the source code that builds into one).

Whether it's timeless functions such as CreateEventW and WaitForSingleObject, powerful graphics engines such as Direct3D, traditional windowing functions such as CreateWindowExW and DispatchMessageW, or more recent user interface (UI) frameworks such as Composition and Xaml, the *windows* crate has you covered.

The win32metadata project aims to provide metadata for Win32 APIs. This metadata describes the API surface—strongly-typed API signatures, parameters, and types. This enables the entire Windows API to be projected in an automated and complete way for consumption by Rust (as well as languages such as C# and C++). Also see Making Win32 APIs more accessible to more languages.

As a Rust developer, you'll use Cargo (Rust's package management tool)—along with `https://crates.io` (the Rust community's crate registry)—to manage the dependencies in your projects. The good news is that you can reference the *windows* crate from your Rust apps, and then immediately beginning calling Windows APIs. You can also find Rust documentation for the *windows* crate over on `https://docs.rs`.

Similar to C++/WinRT, Rust for Windows is an open source language projection developed on GitHub. Use the Rust for Windows repo if you have questions about Rust for Windows, or if you wish to report issues with it.

The Rust for Windows repo also has some simple examples that you can follow. And there's an excellent sample app in the form of Robert Mikhayelyan's Minesweeper.

Contribute to Rust for Windows

Rust for Windows welcomes your contributions!

- Identify and fix bugs in the source code

Rust documentation for the Windows API

Rust for Windows benefits from the polished toolchain that Rust developers enjoy. But if having the entire Windows API at your fingertips seems a little daunting, there's also Rust documentation for the Windows API.

This resource essentially documents how the Windows APIs and types are projected into idiomatic Rust. Use it to browse or search for the APIs you need to know about, and that you need to know how to call.

Writing an app with Rust for Windows

The next topic is the RSS reader tutorial, where we'll walk through writing a simple app with Rust for Windows.

Related

- Overview of developing on Windows with Rust
- RSS reader tutorial
- The *windows* crate
- Documentation for the *windows* crate
- Win32 metadata
- Making Win32 APIs more accessible to more languages
- Rust documentation for the Windows API
- Rust for Windows
- Minesweeper sample app

RSS reader tutorial (Rust for Windows with VS Code)

7/6/2021 • 6 minutes to read • Edit Online

The previous topic introduced Rust for Windows, and the windows crate.

Now let's try out Rust for Windows by writing a simple app that downloads the titles of blog posts from a Really Simple Syndication (RSS) feed.

1. Launch a command prompt (`cmd.exe`), and `cd` to a folder where you want to keep your Rust projects.
2. Then, via Cargo, create a new Rust project named *rss_reader*, and `cd` into the project's newly-created folder.

   ```
   cargo new rss_reader
   cd rss_reader
   ```

3. Now—again via Cargo—we're going to create a new sub-project named *bindings*. As you can see in the command below, this new project is a library, and it's going to serve as the medium through which we bind to the Windows APIs that we want to call. At build time, the *bindings* library sub-project will build into a *crate* (which is Rust's term for a binary or a library). We'll be consuming that crate from within the *rss_reader* project, as we'll see.

   ```
   cargo new --lib bindings
   ```

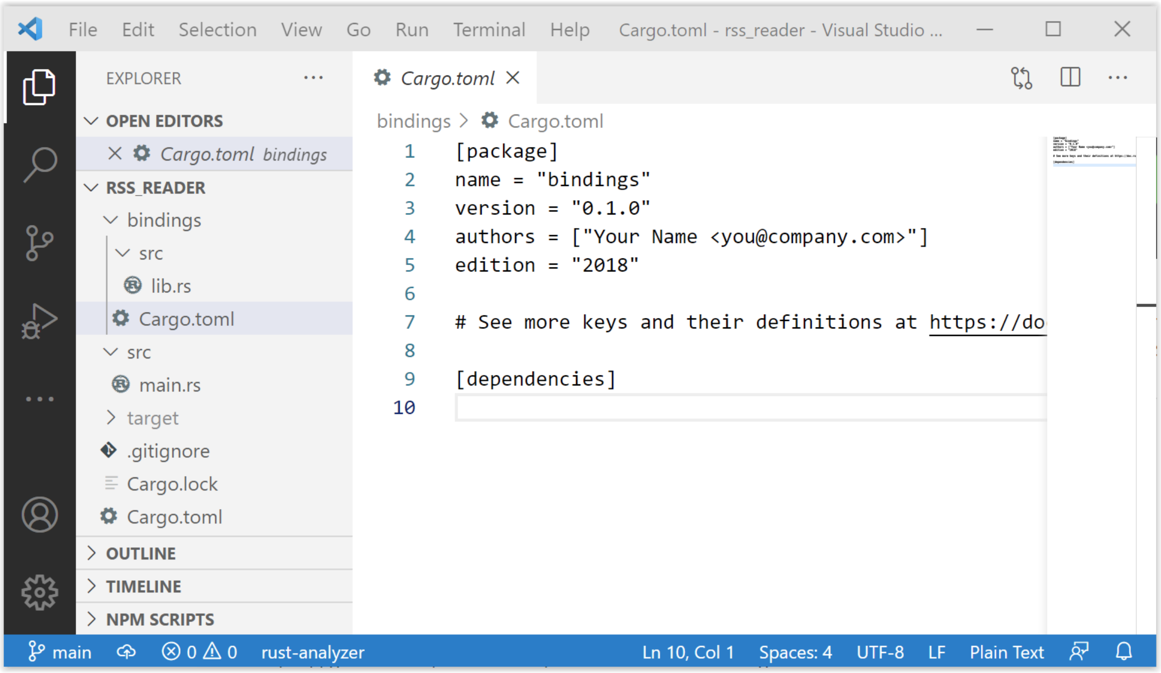
   Making *bindings* a nested crate means that when we build *rss_reader*, *bindings* will be able to cache the results of any bindings we import.

4. Then open the *rss_reader* project in VS Code.

   ```
   code .
   ```

5. Let's work on the *bindings* library first.

   In VS Code's Explorer, open the `bindings` > `Cargo.toml` file.



```
[package]
name = "bindings"
version = "0.1.0"
authors = ["Your Name <you@company.com>"]
edition = "2018"

# See more keys and their definitions at https://do

[dependencies]

```

A `Cargo.toml` file is a text file that describes a Rust project, including any dependencies it has.

Right now, the `dependencies` section is empty. So now edit that section (and also add a `[build-dependencies]` section) so that it looks like this.

```
# bindings\Cargo.toml
...

[dependencies]
windows="0.9.1"

[build-dependencies]
windows="0.9.1"
```

We've just added a dependency on the *windows* crate, both for the *bindings* library and for its build script. That allows Cargo to download, build, and cache Windows support as a package. Set the version number to whatever the latest version is—you'll be able to see that on the web page for the windows crate.

6. Now we can add the build script itself; this is where we'll generate the bindings that we'll ultimately rely on. In VS Code, right-click the *bindings* folder, and click **New File**. Type in the name *build.rs*, and press **Enter**. Edit `build.rs` to look like this.

```
// bindings\build.rs
fn main() {
    windows::build!(
        Windows::Foundation::Collections::IVector,
        Windows::Foundation::{IAsyncOperationWithProgress, Uri},

        Windows::Web::Syndication::{
            ISyndicationText, RetrievalProgress, SyndicationClient, SyndicationFeed, SyndicationItem,
        },
    );
}
```

The `windows::build!` macro takes care of resolving any dependencies in the form of `.winmd` files, and generating bindings for selected types directly from metadata. We *could* have asked for an entire namespace (with `Windows::Web::Syndication::*`). But here we're asking for bindings to be generated only for the particular types we specify (such as SyndicationClient). In this way, you can import as little or as much as you need, and avoid waiting for code to be generated and compiled for things that you'll never need.

As well as the types that we'll be using explicitly, we also specify all of their dependencies. For example, we'll be using a method of **SyndicationClient** that expects a parameter of type Uri. So in the *build* macro we also include the definition for **Windows::Foundation::Uri** so that we can call that method. Other types are part of the **windows** crate itself. For example, **windows::Result** (which we'll see in use soon) is defined by the *windows* crate, so it's always available. Note the lower-case **windows** in **windows::Result**, as compared to the Pascal-cased namespace and type names for Windows types.

7. Open the `bindings` > `src` > `lib.rs` source code file. To include the bindings generated in the previous step, replace the default code that you'll find in `lib.rs` with the following.

```
// bindings\src\lib.rs
windows::include_bindings!();
```

The `windows::include_bindings!` macro includes the source code that was generated in the previous step by the build script. Now, any time you need access to additional APIs, just list them in the build script (`build.rs`).

8. Let's now implement the main *rss_reader* project. First, open the `Cargo.toml` file at the root of the project, and add the following dependency on the inner *bindings* crate, along with a dependency on the *windows* crate.

```
# Cargo.toml
...

[dependencies]
bindings = { path = "bindings" }
windows = "0.9.1"
```

9. Finally, open the *rss_reader* project's `src` > `main.rs` source code file. In there is the simple code that outputs a *Hello, world!* message. Add this code to the beginning of `main.rs` .

```
// src\main.rs
use bindings::{
    Windows::Foundation::Uri,
    Windows::Web::Syndication::SyndicationClient,
};

fn main() {
    println!("Hello, world!");
}
```

The `use` declaration shortens the path to the types that we'll be using. There's the **Uri** type that we mentioned earlier.

10. To create a new Uri, add this code into the **main** function.

```
// src\main.rs
...

fn main() -> windows::Result<()> {
    let uri = Uri::CreateUri("https://blogs.windows.com/feed")?;

    Ok(())
}
```

Notice that we're using **windows::Result** as the return type of the **main** function. This will make things easier, as it's common to deal with errors from operating system (OS) APIs. **windows::Result** helps us with error propagation, and concise error handling.

You can see the question-mark operator at the end of the line of code that creates a **Uri**. To save on typing, we do that to make use of Rust's error-propagation and short-circuiting logic. That means we don't have to do a bunch of manual error handling for this simple example. For more info about this feature of Rust, see The ? operator for easier error handling.

11. To download this RSS feed, we'll create a new **SyndicationClient** object.

```
// src\main.rs
...

fn main() -> windows::Result<()> {
    let uri = Uri::CreateUri("https://blogs.windows.com/feed")?;
    let client = SyndicationClient::new()?;

    Ok(())
}
```

The **new** function is Rust's equivalent of the default constructor.

12. Now we can use the **SyndicationClient** object to retrieve the feed.

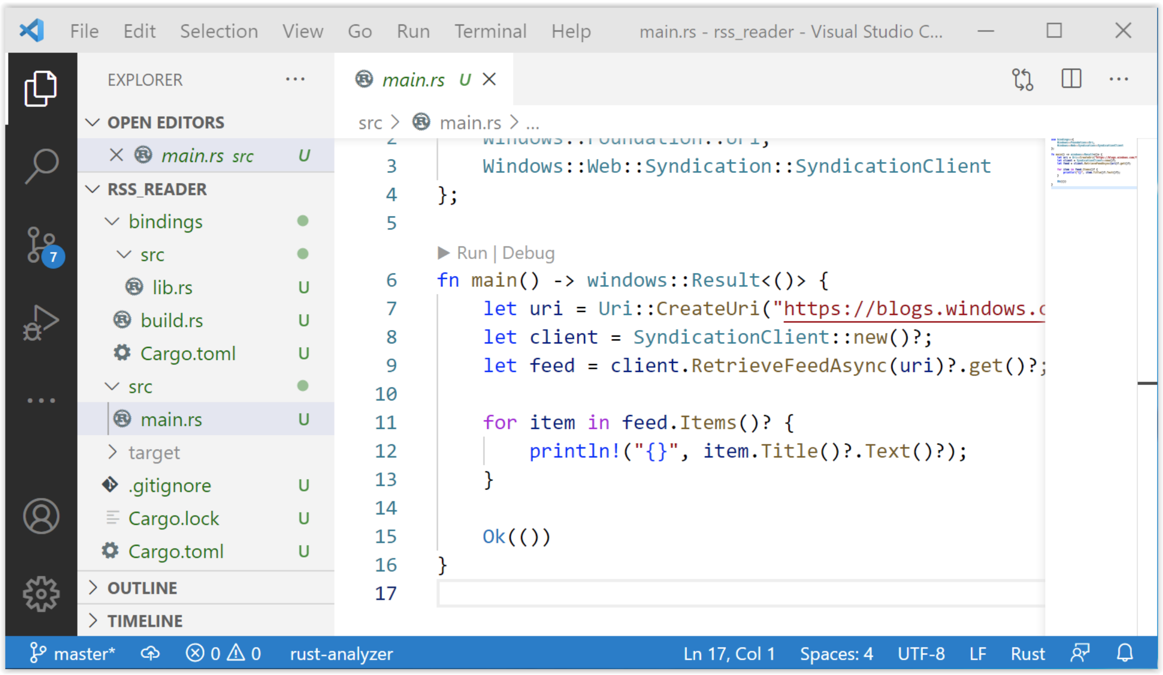
```
// src\main.rs
...

fn main() -> windows::Result<()> {
    let uri = Uri::CreateUri("https://blogs.windows.com/feed")?;
    let client = SyndicationClient::new()?;
    let feed = client.RetrieveFeedAsync(uri)?.get()?;

    Ok(())
}
```

Because **RetrieveFeedAsync** is an asynchronous API, we can use the blocking **get** function (as shown above). Alternatively, we could use the `await` operator within an `async` function (to cooperatively wait for the results), much as you would do in C# or C++.

13. Now we can simply iterate over the resulting items, and let's print out just the titles.

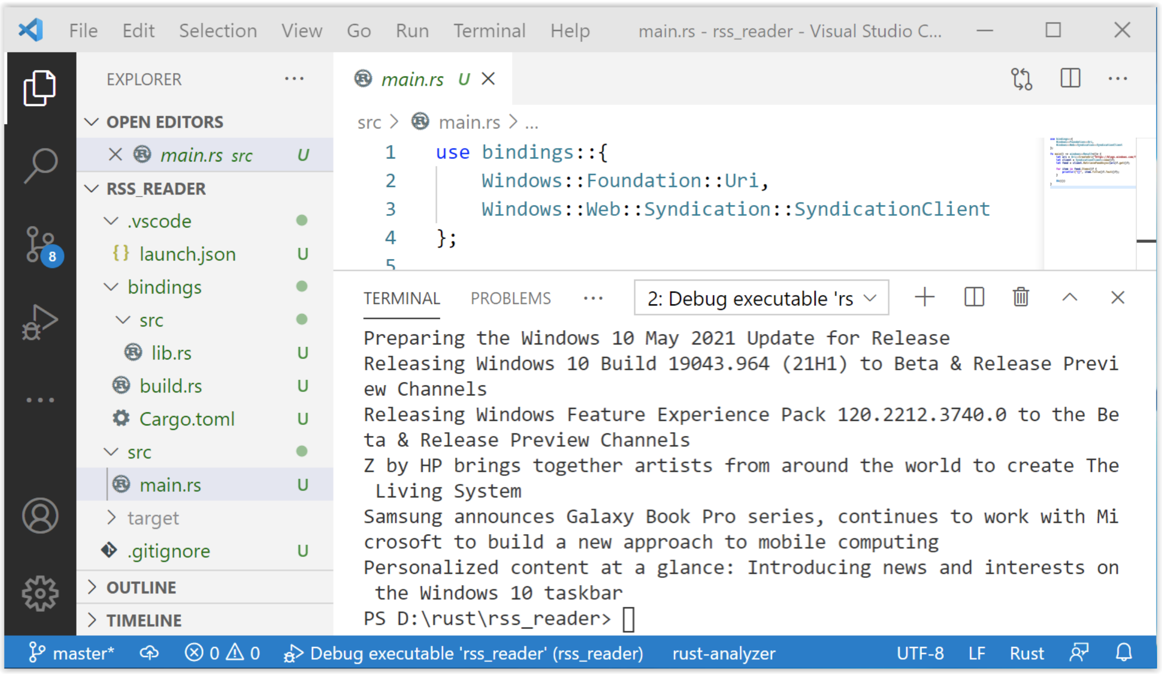
```
// src\main.rs
...

fn main() -> windows::Result<()> {
    let uri = Uri::CreateUri("https://blogs.windows.com/feed")?;
    let client = SyndicationClient::new()?;
    let feed = client.RetrieveFeedAsync(uri)?.get()?;

    for item in feed.Items()? {
        println!("{}", item.Title()?.Text()?);
    }

    Ok(())
}
```

14. Now let's confirm that we can build and run by clicking **Run** > **Run Without Debugging** (or pressing **Ctrl+F5**). There are also **Debug** and **Run** commands embedded inside the text editor. Alternatively, you can submit the command `cargo run` from the command prompt (`cd` into the `rss_reader` folder first), which will build and then run.



```
    Windows::Web::Syndication::SyndicationClient
};

Run | Debug
fn main() -> windows::Result<()> {
    let uri = Uri::CreateUri("https://blogs.windows.c
    let client = SyndicationClient::new()?;
    let feed = client.RetrieveFeedAsync(uri)?.get()?;

    for item in feed.Items()? {
        println!("{}", item.Title()?.Text()?);
    }

    Ok(())
}
```

Down in the **Terminal** pane, you can see that Cargo successfully downloads and compiles the **windows** crate, caching the results, and using them to make subsequent builds complete in less time. It then builds the sample, and runs it, displaying a list of blog post titles.



```
use bindings::{
    Windows::Foundation::Uri,
    Windows::Web::Syndication::SyndicationClient
};
```

```
Preparing the Windows 10 May 2021 Update for Release
Releasing Windows 10 Build 19043.964 (21H1) to Beta & Release Previ
ew Channels
Releasing Windows Feature Experience Pack 120.2212.3740.0 to the Be
ta & Release Preview Channels
Z by HP brings together artists from around the world to create The
 Living System
Samsung announces Galaxy Book Pro series, continues to work with Mi
crosoft to build a new approach to mobile computing
Personalized content at a glance: Introducing news and interests on
 the Windows 10 taskbar
PS D:\rust\rss_reader>
```

That's as simple as it is to program Rust for Windows. Under the hood, however, a lot of love goes into building the tooling so that Rust can both parse `.winmd` files based on ECMA-335 (Common Language Infrastructure, or CLI) at compile time, and also faithfully honor the COM-based application binary interface (ABI) at run-time with both safety and efficiency in mind.

Showing a message box

We did say that Rust for Windows lets you call any Windows API (past, present, and future). So in this section we'll add code to show a Windows message box to the user.

1. Open the `bindings` > `build.rs` source code file, and add to the generated bindings the function shown below.

```
// bindings\build.rs
fn main() {
    windows::build!(
        ...
        Windows::Win32::UI::WindowsAndMessaging::MessageBoxA,
    );
}
```

2. Next, open the project's `src` > `main.rs` source code file, and update the `use` declaration with the new namespace, or module. And finally add code to call the MessageBoxA function (also see MessageBoxA in the Rust documentation for the Windows API, which includes a link to MESSAGEBOX_STYLE).

```
// src\main.rs
use bindings::{
    Windows::Foundation::Uri,
    Windows::Web::Syndication::SyndicationClient,
    Windows::Win32::UI::WindowsAndMessaging::*,
};

fn main() {
    ...

    unsafe {
        MessageBoxA(None, "Text", "Caption", MB_OK);
    }

    Ok(())
}
```

As you can see, we mark these older Win32 APIs as `unsafe` (see Unsafe blocks).

This time when you build and run, Rust displays a Windows message box after listing the blog post titles.

Related

- Rust for Windows, and the windows crate
- ECMA-335
- The ? operator for easier error handling
- Unsafe blocks

Windows Package Manager

5/26/2021 • 2 minutes to read • Edit Online

Windows Package Manager is a comprehensive package manager solution that consists of a command line tool and set of services for installing applications on Windows 10.

Windows Package Manager for developers

Developers use the **winget** command line tool to discover, install, upgrade, remove and configure a curated set of applications. After it is installed, developers can access **winget** via the Windows Terminal, PowerShell, or the Command Prompt.

For more information, see Use the winget tool to install and manage applications.

Windows Package Manager for ISVs

Independent Software Vendors (ISVs) can use Windows Package Manager as a distribution channel for software packages containing their tools and applications. To submit software packages (containing .msix, .msi, or .exe installers) to Windows Package Manager, we provide the open source **Microsoft Community Package Manifest Repository** on GitHub where ISVs can upload package manifests to have their software packages considered for inclusion with Windows Package Manager. Manifests are automatically validated and may also be reviewed manually.

For more information, see Submit packages to Windows Package Manager.

Understanding package managers

A package manager is a system or set of tools used to automate installing, upgrading, configuring and using software. Most package managers are designed for discovering and installing developer tools.

Ideally, developers use a package manager to specify the prerequisites for the tools they need to develop solutions for a given project. The package manager then follows the declarative instructions to install and configure the tools. The package manager reduces the time spent getting an environment ready, and it helps ensure the same versions of packages are installed on their machine.

Third party package managers can leverage the Microsoft Community Package Manifest Repository to increase the size of their software catalog.

Related topics

- Use the winget tool to install and manage software packages
- Submit packages to Windows Package Manager

Use the winget tool to install and manage applications

7/7/2021 • 3 minutes to read • Edit Online

The **winget** command line tool enables developers to discover, install, upgrade, remove and configure applications on Windows 10 computers. This tool is the client interface to the Windows Package Manager service.

The **winget** tool is currently a preview, so not all planned functionality is available at this time.

Install winget

There are several ways to install the **winget** tool:

- The **winget** tool is included in the flight or preview version of Windows App Installer. You must install the preview version of **App Installer** to use **winget**. To gain early access, submit your request to the Windows Package Manager Insiders Program. Participating in the flight ring will guarantee you see the latest preview updates.
- Participate in the Windows Insider flight ring.
- Install the Windows Desktop App Installer package located on the Releases page for the winget repository.

NOTE

The **winget** tool requires Windows 10, version 1809 (10.0.17763), or a later version of Windows 10.

Administrator considerations

Installer behavior can be different depending on whether you are running **winget** with administrator privileges.

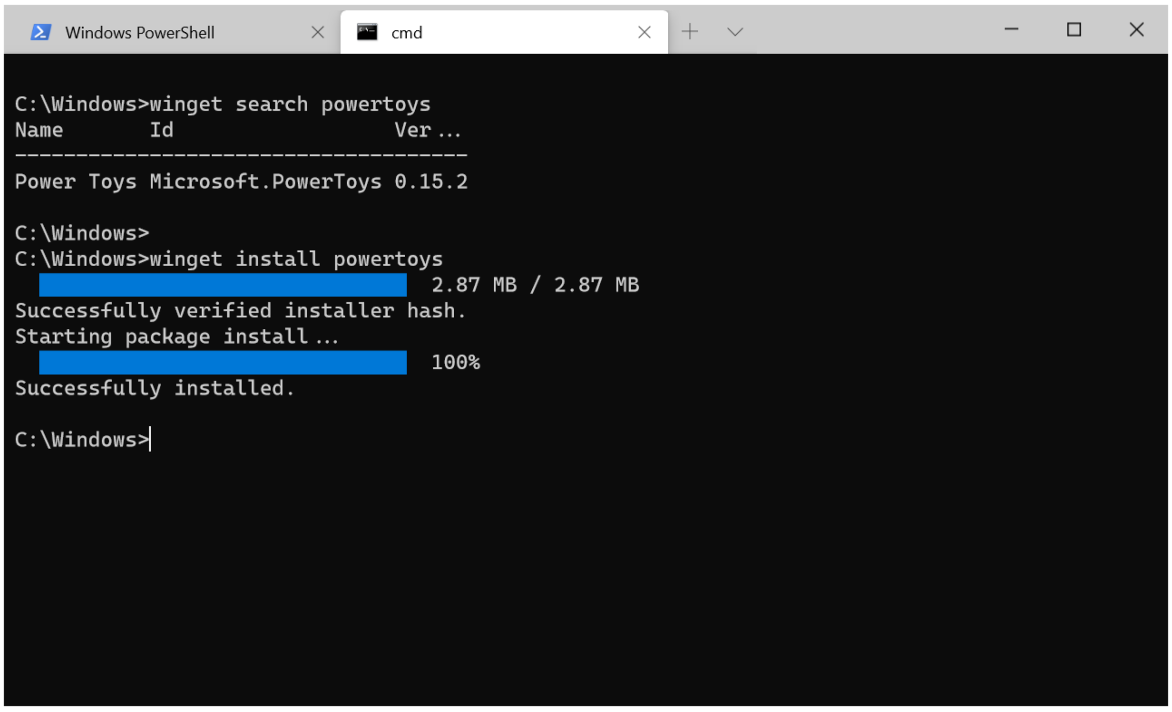
- When running **winget** without administrator privileges, some applications may require elevation to install. When the installer runs, Windows will prompt you to elevate. If you choose not to elevate, the application will fail to install.
- When running **winget** in an Administrator Command Prompt, you will not see elevation prompts if the application requires it. Always use caution when running your command prompt as an administrator, and only install applications you trust.

Use winget

After **App Installer** is installed, you can run **winget** by typing 'winget' from a Command Prompt.

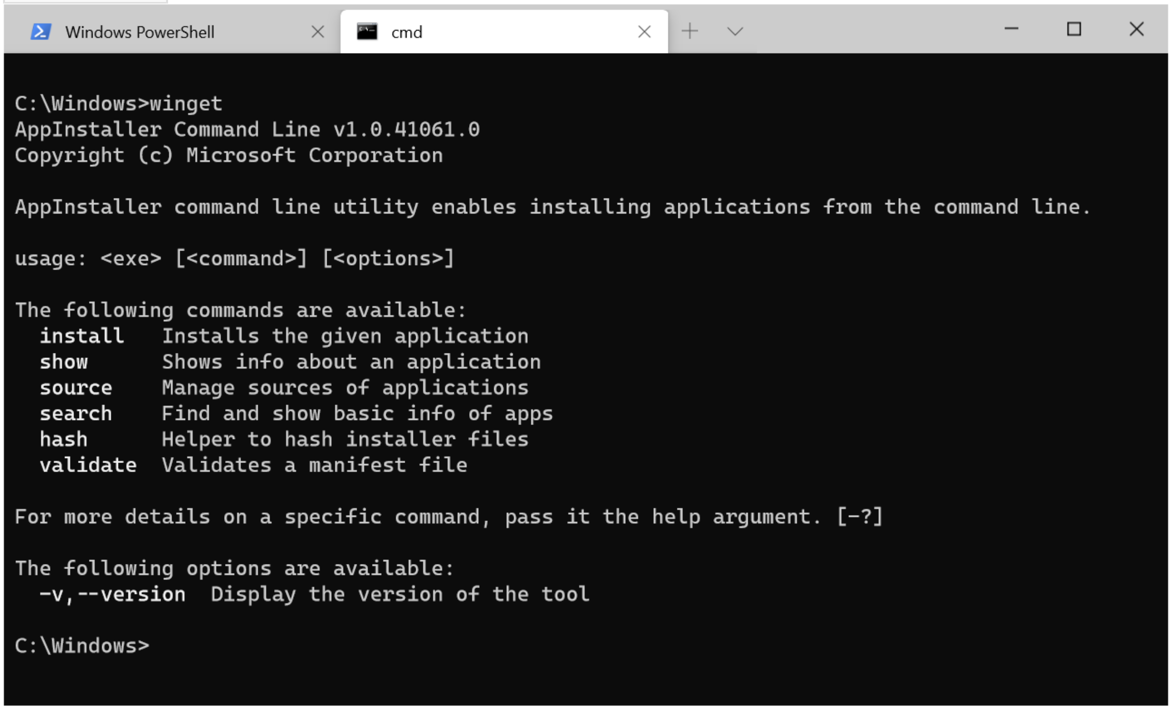
One of the most common usage scenarios is to search for and install a favorite tool.

1. To search for a tool, type `winget search <appname>`.
2. After you have confirmed that the tool you want is available, you can install the tool by typing `winget install <appname>`. The **winget** tool will launch the installer and install the application on your PC.



```
C:\Windows>winget search powertoys
Name       Id                   Ver ...
-------------------------------------
Power Toys Microsoft.PowerToys 0.15.2

C:\Windows>
C:\Windows>winget install powertoys
                                  2.87 MB / 2.87 MB
Successfully verified installer hash.
Starting package install ...
                                  100%
Successfully installed.

C:\Windows>
```

3. In addition to install and search, **winget** provides a number of other commands that enable you to show details on applications, change sources, and validate packages. To get a complete list of commands, type: `winget --help`.



```
C:\Windows>winget
AppInstaller Command Line v1.0.41061.0
Copyright (c) Microsoft Corporation

AppInstaller command line utility enables installing applications from the command line.

usage: <exe> [<command>] [<options>]

The following commands are available:
  install   Installs the given application
  show      Shows info about an application
  source    Manage sources of applications
  search    Find and show basic info of apps
  hash      Helper to hash installer files
  validate  Validates a manifest file

For more details on a specific command, pass it the help argument. [-?]

The following options are available:
  -v,--version  Display the version of the tool

C:\Windows>
```

Commands

The current preview of the **winget** tool supports the following commands.

COMMAND	DESCRIPTION
export	Exports a list of the installed packages.
features	Shows the status of experimental features.
hash	Generates the SHA256 hash for the installer.
import	Installs all the packages in a file.

COMMAND	DESCRIPTION
install	Installs the specified application.
list	Display installed packages.
search	Searches for an application.
settings	Open settings.
show	Displays details for the specified application.
source	Adds, removes, and updates the Windows Package Manager repositories accessed by the **winget** tool.
validate	Validates a manifest file for submission to the Windows Package Manager repository.
uninstall	Uninstalls the given package.
upgrade	Upgrades the given package.

Options

The current preview of the **winget** tool supports the following options.

OPTION	DESCRIPTION
-v, --version	Returns the current version of winget.
--info	Provides you with all detailed information on winget, including the links to the license, privacy statement, and configured group policies.
-?, --help	Shows additional help for winget.

Supported installer formats

The current preview of the **winget** tool supports the following types of installers:

- EXE
- MSIX
- MSI

Scripting winget

You can author batch scripts and PowerShell scripts to install multiple applications.

```
@echo off
Echo Install Powertoys and Terminal
REM Powertoys
winget install Microsoft.Powertoys
if %ERRORLEVEL% EQU 0 Echo Powertoys installed successfully.
REM Terminal
winget install Microsoft.WindowsTerminal
if %ERRORLEVEL% EQU 0 Echo Terminal installed successfully.   %ERRORLEVEL%
```

NOTE

When scripted, **winget** will launch the applications in the specified order. When an installer returns success or failure, **winget** will launch the next installer. If an installer launches another process, it is possible that it will return to **winget** prematurely. This will cause **winget** to install the next installer before the previous installer has completed.

Missing tools

If the community repository does not include your tool or application, please submit a package to our repository. By adding your favorite tool, it will be available to you and everyone else.

Customize winget settings

You can configure the **winget** command line experience by modifying the **settings.json** file. For more information, see https://aka.ms/winget-settings. Note that the settings are still in an experimental state and not yet finalized for the preview version of the tool.

Open source details

The **winget** tool is open source software available on GitHub in the repo https://github.com/microsoft/winget-cli/. The source for building the client is located in the src folder.

The source for **winget** is contained in a Visual Studio 2019 C++ solution. To build the solution correctly, install the latest Visual Studio with the C++ workload.

We encourage you to contribute to the **winget** source on GitHub. You must first agree to and sign the Microsoft CLA.

export command (winget)

5/26/2021 • 2 minutes to read • Edit Online

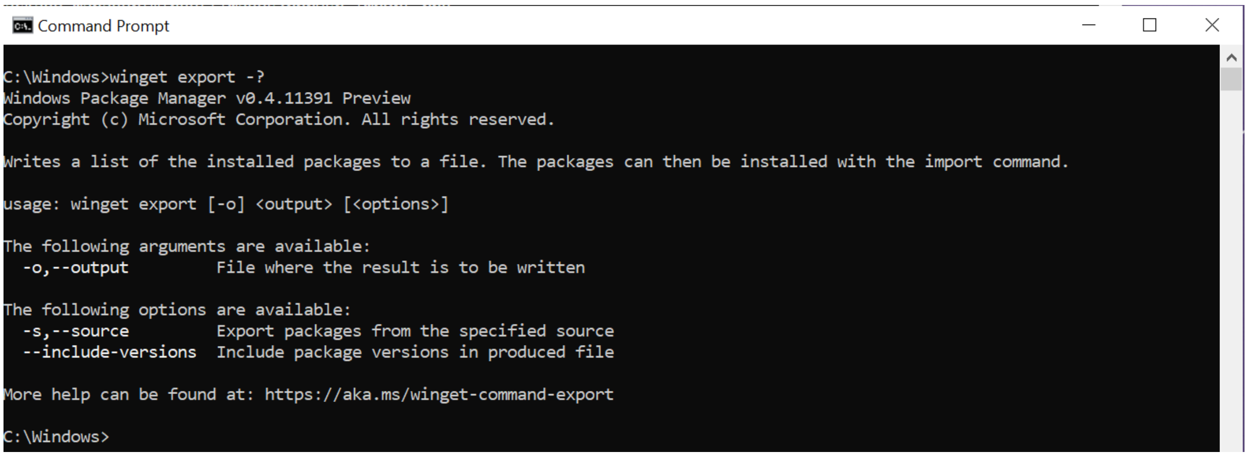
The **export** command of the winget tool exports a JSON file of apps to a specified file. The **export** command uses JSON as the format. The JSON schema used by **winget** can be found here.

The **export** combined with the **import** command allows you to batch install applications on your PC.

The **export** command is often used to create a file that you can share with other developers, or for use when restoring your build environment.

Usage

```
winget export [-o] <output> [<options>]
```



```
C:\Windows>winget export -?
Windows Package Manager v0.4.11391 Preview
Copyright (c) Microsoft Corporation. All rights reserved.

Writes a list of the installed packages to a file. The packages can then be installed with the import command.

usage: winget export [-o] <output> [<options>]

The following arguments are available:
  -o,--output          File where the result is to be written

The following options are available:
  -s,--source          Export packages from the specified source
  --include-versions   Include package versions in produced file

More help can be found at: https://aka.ms/winget-command-export

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-o,--output	Path to the JSON file to be created

Options

The options allow you to customize the export experience to meet your needs.

OPTION	DESCRIPTION
-s, --source	[optional] Specifies a source to export files from. Use this option when you only want files from a specific source.
--include-versions	[optional] Includes the version of the app currently installed. Use this option if you want a specific version. By default, unless specified, **import** will use latest.

JSON schema

The driving force behind the **export** command is the JSON file. You can find the schema for the JSON file here.

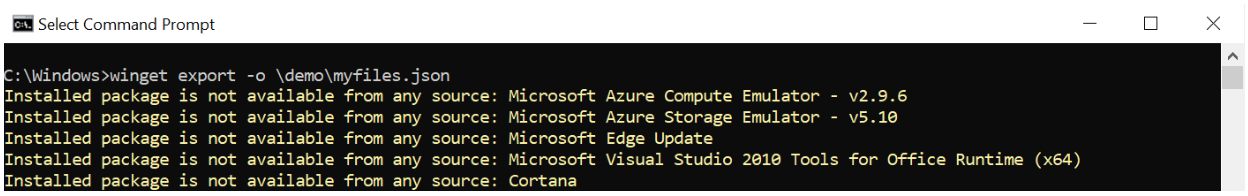
The JSON file includes the following hierarchy.

ENTRY	DESCRIPTION
Sources	The sources application manifests come from.
Packages	The collection of packages to install.
Id	The Windows Package Manager package identifier used to specify the package.
Version	[optional] The specific version of the package to install.

Exporting files

When the Windows Package Manager exports the JSON file, it attempts to export all the applications installed on the PC. If the **winget export** command is not able to match an application to an application from an available **source**, the export command will show a warning.

> **NOTE**
>
> Matching an application depends on metadata in the manifest from a configured source, and metadata in **Add / Remove Programs** in Windows based on the package installer.



```
C:\Windows>winget export -o \demo\myfiles.json
Installed package is not available from any source: Microsoft Azure Compute Emulator - v2.9.6
Installed package is not available from any source: Microsoft Azure Storage Emulator - v5.10
Installed package is not available from any source: Microsoft Edge Update
Installed package is not available from any source: Microsoft Visual Studio 2010 Tools for Office Runtime (x64)
Installed package is not available from any source: Cortana
```

After the export is complete, you can edit the resulting JSON file in your favorite editor. You can remove apps you do not wish to import in the future.

features command (winget)

5/26/2021 • 2 minutes to read • Edit Online

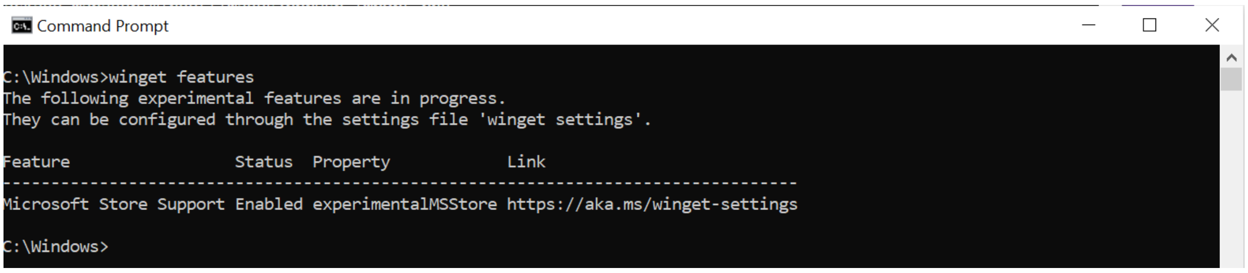
The **features** command of the winget tool displays a list of the experimental features available with your version of the Windows Package Manager.

Each feature can be turned on individually by enabling the features through **settings**.

You can find the latest up to date information features on the experimental features web page.

Usage

```
winget features
```



Command Prompt

```
C:\Windows>winget features
The following experimental features are in progress.
They can be configured through the settings file 'winget settings'.

Feature                Status  Property            Link
---------------------------------------------------------------------------------
Microsoft Store Support Enabled experimentalMSStore https://aka.ms/winget-settings

C:\Windows>
```

Notice above that the status of each feature is listed. If the feature is **disabled** you will not be able to use it. If the feature is **enabled** you will notice that the command will be available to you through **winget**.

To enabled any disabled features, go to **settings** and enable the feature.

NOTE

Features may be managed by group policy. You can use the **winget --info** command to view any policies in effect on your system.

hash command (winget)

5/26/2021 • 2 minutes to read • Edit Online

The **hash** command of the winget tool generates the SHA256 hash for an installer. This command is used if you need to create a manifest file for submitting software to the **Microsoft Community Package Manifest Repository** on GitHub. In addition, the **hash** command also supports generating a SHA256 certificate hash for MSIX files.

Usage

```
winget hash [-f] \<file> [\<options>]
```

The **hash** sub command can only run on a local file. To use the **hash** sub command, download your installer to a known location. Then pass in the file path as an argument to the **hash** sub command.

Arguments

The following arguments are available:

ARGUMENT	DESCRIPTION
-f,--file	The path to the file to be hashed.
-m,--msix	Specifies that the hash command will also create the SHA 256 SignatureSha256 for use with MSIX installers.
-?, --help	Gets additional help on this command.

Related topics

- Use the winget tool to install and manage applications
- Submit packages to Windows Package Manager

help command (winget)

5/26/2021 • 2 minutes to read • Edit Online

The **help** command of the winget tool displays help for all the supported commands and sub commands. In addition, you can pass the **--help** argument to any other command to get details about all additional command options.

Usage

- Display help for all commands: `winget --help`
- View options for a command: `winget <command> --help`

Related topics

- Use the winget tool to install and manage applications

import command (winget)

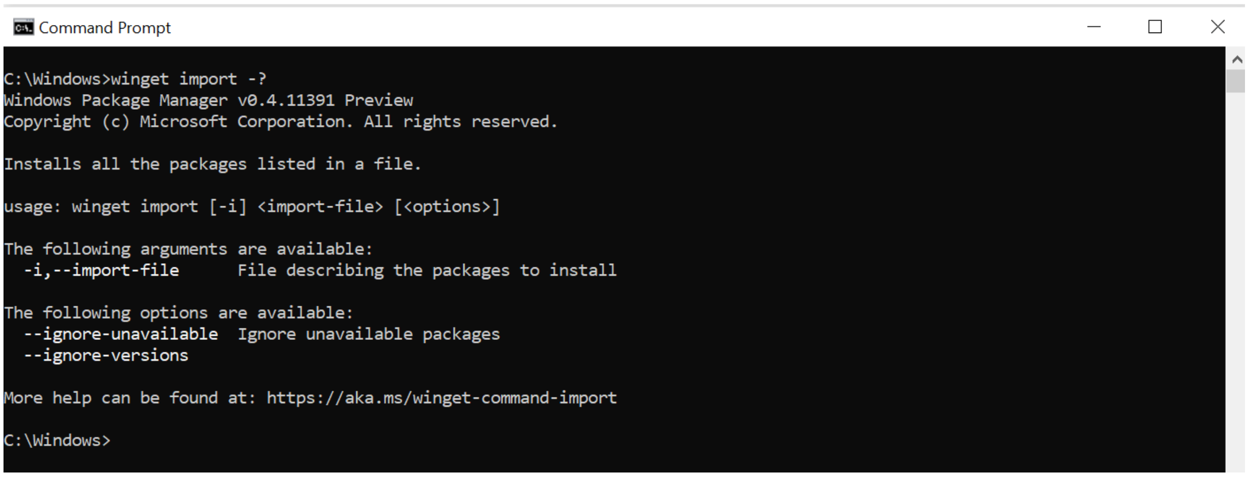
5/26/2021 • 2 minutes to read • Edit Online

The **import** command of the winget tool imports a JSON file of apps to install. The **import** command combined with the **export** command allows you to batch install applications on your PC.

The **import** command is often used to share your developer environment or build up your PC image with your favorite apps.

Usage

```
winget import [-i] <import-file> [<options>]
```



Command Prompt

```
C:\Windows>winget import -?
Windows Package Manager v0.4.11391 Preview
Copyright (c) Microsoft Corporation. All rights reserved.

Installs all the packages listed in a file.

usage: winget import [-i] <import-file> [<options>]

The following arguments are available:
  -i,--import-file      File describing the packages to install

The following options are available:
  --ignore-unavailable  Ignore unavailable packages
  --ignore-versions

More help can be found at: https://aka.ms/winget-command-import

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-i,--import-file	JSON file describing the packages to install.

Options

The options allow you to customize the import experience to meet your needs.

OPTION	DESCRIPTION
--ignore-unavailable	Suppresses errors if the app requested is unavailable.
--ignore-versions	Ignores versions specified in the JSON file and installs the latest available version.

JSON Schema

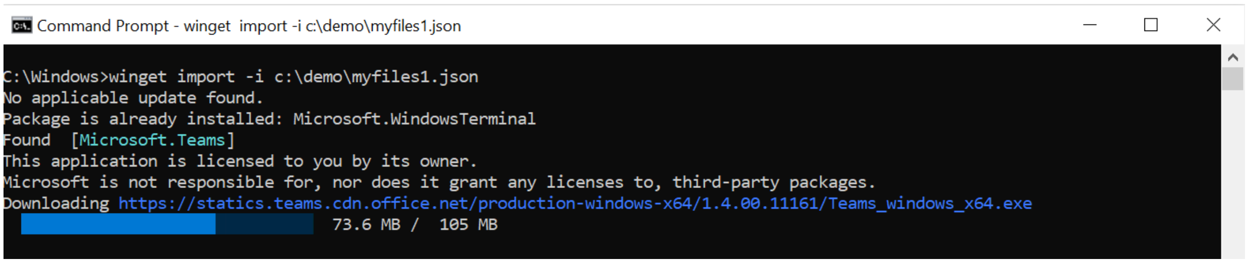
The driving force behind the **import** command is the JSON file. You can find the schema for the JSON file here.

The JSON file includes the following hierarchy.

ENTRY	DESCRIPTION
Sources	The sources application manifests come from.
Packages	The collection of packages to install.
Id	The Windows Package Manager package identifier used to specify the package.
Version	[optional] The specific version of the package to install.

Importing files

When the Windows Package Manager imports the JSON file, it attempts to install the specified applications in a serial fashion. If the application is not available or the application is already installed, it will notify the user of that case.



```
Command Prompt - winget  import -i c:\demo\myfiles1.json

C:\Windows>winget import -i c:\demo\myfiles1.json
No applicable update found.
Package is already installed: Microsoft.WindowsTerminal
Found  [Microsoft.Teams]
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://statics.teams.cdn.office.net/production-windows-x64/1.4.00.11161/Teams_windows_x64.exe
                                  73.6 MB /  105 MB
```

In the previous example, the Microsoft.WindowsTerminal was already installed. Therefore the import command skipped passed the installation.

install command (winget)

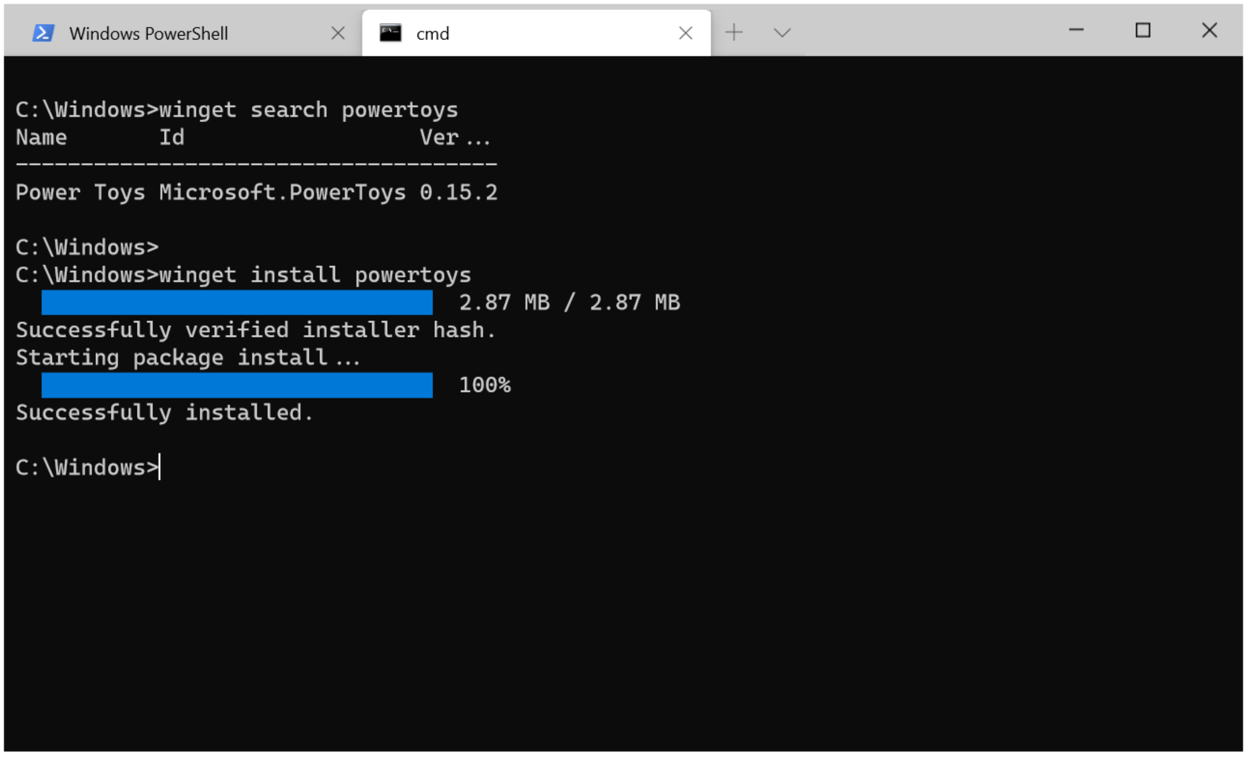
7/6/2021 • 2 minutes to read • Edit Online

The **install** command of the winget tool installs the specified application. Use the **search** command to identify the application you want to install.

The **install** command requires that you specify the exact string to install. If there is any ambiguity, you will be prompted to further filter the **install** command to an exact application.

Usage

```
winget install [[-q] \<query>] [\<options>]
```



```
C:\Windows>winget search powertoys
Name       Id                   Ver ...
-------------------------------------
Power Toys Microsoft.PowerToys 0.15.2

C:\Windows>
C:\Windows>winget install powertoys
                              2.87 MB / 2.87 MB
Successfully verified installer hash.
Starting package install...
                              100%
Successfully installed.

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--**query**	The query used to search for an app.
-?, --**help**	Get additional help on this command.

Options

The options allow you to customize the install experience to meet your needs.

OPTION	DESCRIPTION
-m, --manifest	Must be followed by the path to the manifest (YAML) file. You can use the manifest to run the install experience from a local YAML file.
--id	Limits the install to the ID of the application.
--name	Limits the search to the name of the application.
--moniker	Limits the search to the moniker listed for the application.
-v, --version	Enables you to specify an exact version to install. If not specified, latest will install the highest versioned application.
-s, --source	Restricts the search to the source name provided. Must be followed by the source name.
--scope	Allows you to specify if the installer should target user or machine scope.
-e, --exact	Uses the exact string in the query, including checking for case-sensitivity. It will not use the default behavior of a substring.
-i, --interactive	Runs the installer in interactive mode. The default experience shows installer progress.
-h, --silent	Runs the installer in silent mode. This suppresses all UI. The default experience shows installer progress.
--locale	Specifies which locale to use (BCP47 format).
-o, --log	Directs the logging to a log file. You must provide a path to a file that you have the write rights to.
--override	A string that will be passed directly to the installer.
-l, --location	Location to install to (if supported).
--force	Overrides the installer hash check. Not recommended.

Example queries

The following example installs a specific version of an application.

```
winget install powertoys --version 0.15.2
```

The following example installs an application from its ID.

```
winget install --id Microsoft.PowerToys
```

The following example installs an application by version and ID.

```
winget install --id Microsoft.PowerToys --version 0.15.2
```

Multiple selections

If the query provided to **winget** does not result in a single application, then **winget** will display the results of the search. This will provide you with the additional data necessary to refine the search for a correct install.

The best way to limit the selection to one file is to use the **id** of the application combined with the **exact** query option. For example:

```
winget install --id Git.Git -e
```

If multiple sources are configured, it is possible to have duplicate entries. Specifying a source is required to further disambiguate.

```
winget install --id Git.Git -e -source winget
```

Local install

The **manifest** option enables you to install an application by passing in a YAML file directly to the client. If the manifest is a multi-file manifest, the directory containing the files must be used. The **manifest** option has the following usage.

Usage: `winget install --manifest \<path>`

OPTION	DESCRIPTION
-m, --manifest	The path to the manifest of the application to install.

Log files

The log files for winget unless redirected, will be located in the following folder: **%temp%\AICLI\*.log**

Related topics

- Use the winget tool to install and manage applications

list command (winget)

6/30/2021 • 2 minutes to read • Edit Online

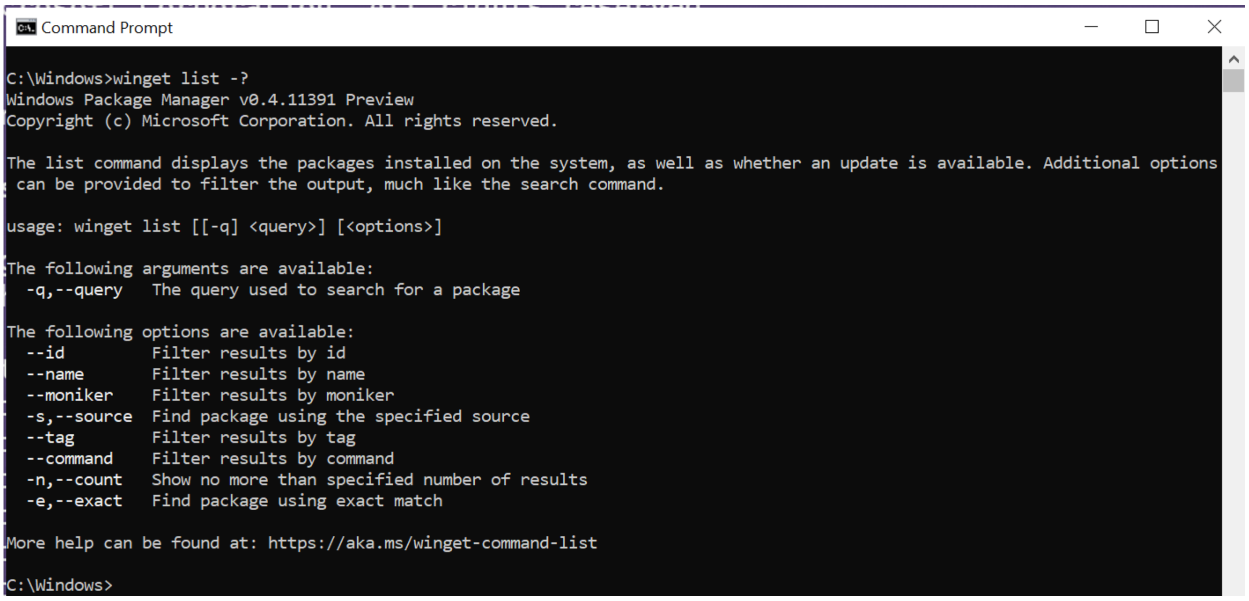
The **list** command of the winget tool displays a list of the applications currently installed on your computer. The list command will show apps that were installed through the Windows Package Manager as well as apps that were installed by other means.

The **list** command will also display if an update is available for an app, and you can use the **upgrade** command to update the app.

The **list** command also supports filters which can be used to limit your list query.

Usage

```
winget list [[-q] \<query>] [\<options>]
```



```
C:\Windows>winget list -?
Windows Package Manager v0.4.11391 Preview
Copyright (c) Microsoft Corporation. All rights reserved.

The list command displays the packages installed on the system, as well as whether an update is available. Additional options
 can be provided to filter the output, much like the search command.

usage: winget list [[-q] <query>] [<options>]

The following arguments are available:
  -q,--query    The query used to search for a package

The following options are available:
  --id          Filter results by id
  --name        Filter results by name
  --moniker     Filter results by moniker
  -s,--source   Find package using the specified source
  --tag         Filter results by tag
  --command     Filter results by command
  -n,--count    Show no more than specified number of results
  -e,--exact    Find package using exact match

More help can be found at: https://aka.ms/winget-command-list

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--**query**	The query used to search for an app.
-?, --**help**	Get additional help on this command.

Options

The options allow you to customize the list experience to meet your needs.

OPTION	DESCRIPTION
--**id**	Limits the list to the ID of the application.

OPTION	DESCRIPTION
--name	Limits the list to the name of the application.
--moniker	Limits the list to the moniker listed for the application.
-s, --source	Restricts the list to the source name provided. Must be followed by the source name.
--tag	Filters results by tags.
--command	Filters results by command specified by the application.
-n, --count	Limits the number of apps displayed in one query.
-l, --location	Location to list to (if supported).
-e, --exact	Uses the exact string in the list query, including checking for case-sensitivity. It will not use the default behavior of a substring.

Example queries

The following example lists a specific version of an application.

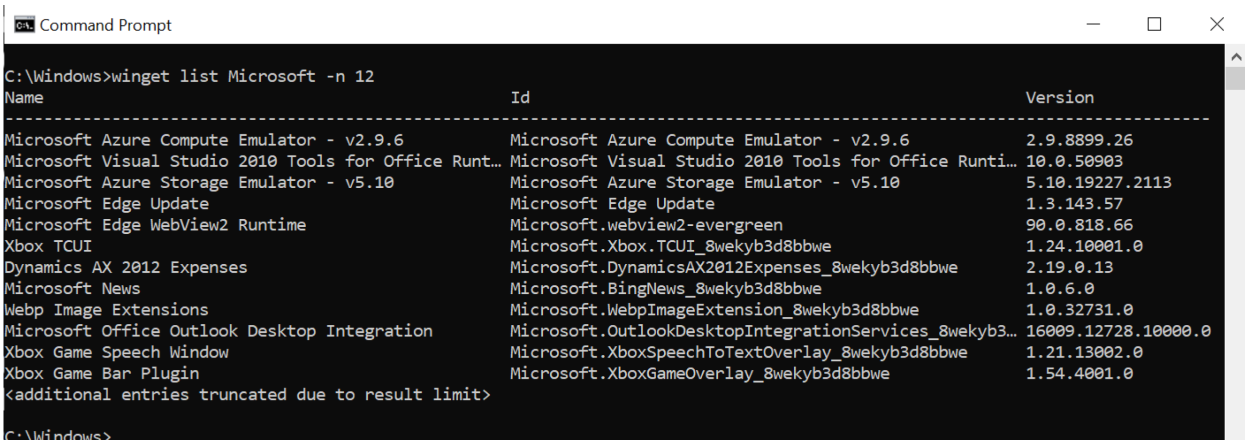
```
winget list --name powertoys
```

The following example lists all application by ID from a specific source.

```
winget list --id Microsoft.PowerToys --source winget
```

The following example limits the output of list to twelve apps.

```
winget list -n 12
```



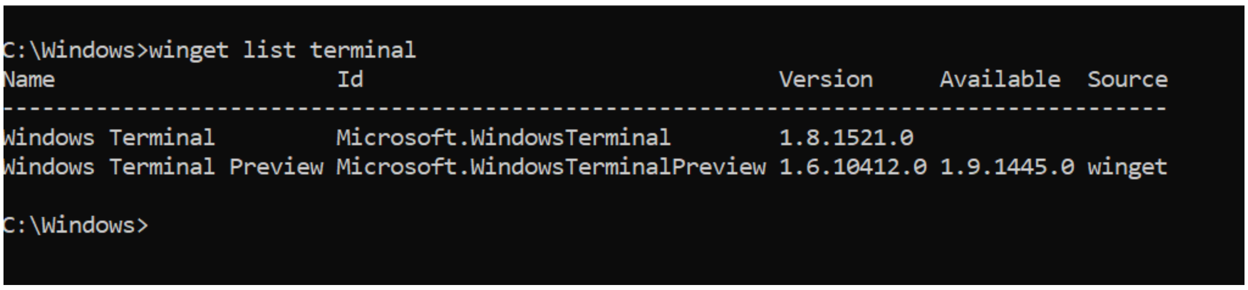
```
C:\Windows>winget list Microsoft -n 12
Name                                                Id                                                   Version
----------------------------------------------------------------------------------------------------------------------------
Microsoft Azure Compute Emulator - v2.9.6           Microsoft Azure Compute Emulator - v2.9.6            2.9.8899.26
Microsoft Visual Studio 2010 Tools for Office Runt… Microsoft Visual Studio 2010 Tools for Office Runti… 10.0.50903
Microsoft Azure Storage Emulator - v5.10            Microsoft Azure Storage Emulator - v5.10             5.10.19227.2113
Microsoft Edge Update                               Microsoft Edge Update                                1.3.143.57
Microsoft Edge WebView2 Runtime                     Microsoft.webview2-evergreen                         90.0.818.66
Xbox TCUI                                           Microsoft.Xbox.TCUI_8wekyb3d8bbwe                    1.24.10001.0
Dynamics AX 2012 Expenses                           Microsoft.DynamicsAX2012Expenses_8wekyb3d8bbwe       2.19.0.13
Microsoft News                                      Microsoft.BingNews_8wekyb3d8bbwe                     1.0.6.0
Webp Image Extensions                               Microsoft.WebpImageExtension_8wekyb3d8bbwe           1.0.32731.0
Microsoft Office Outlook Desktop Integration        Microsoft.OutlookDesktopIntegrationServices_8wekyb3… 16009.12728.10000.0
Xbox Game Speech Window                             Microsoft.XboxSpeechToTextOverlay_8wekyb3d8bbwe      1.21.13002.0
Xbox Game Bar Plugin                                Microsoft.XboxGameOverlay_8wekyb3d8bbwe              1.54.4001.0
<additional entries truncated due to result limit>

C:\Windows>
```

List with update

As stated above, the **list** command allows you to see what apps you have installed that have updates available.

In the image below, you will notice the preview version of Terminal has an update available.

Command Prompt



```
C:\Windows>winget list terminal
Name                    Id                                Version     Available   Source
-----------------------------------------------------------------------------------------
Windows Terminal        Microsoft.WindowsTerminal         1.8.1521.0
Windows Terminal Preview Microsoft.WindowsTerminalPreview 1.6.10412.0 1.9.1445.0 winget

C:\Windows>
```

The `list` command will show not only the update version available, but the source that the update is available from.

If there are no updates available, `list` will only show you the currently installed version and the update column will not be displayed.

- Use the winget tool to list and manage applications

search command (winget)

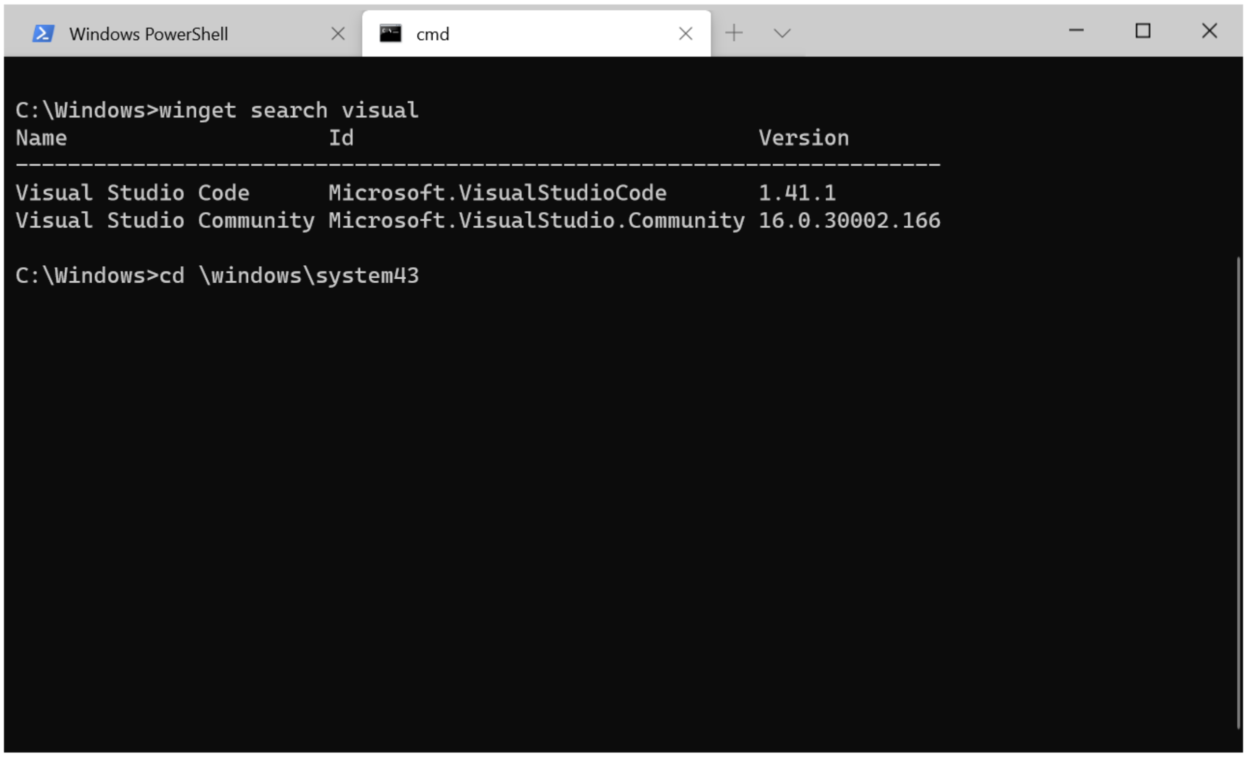
5/26/2021 • 2 minutes to read • Edit Online

The **search** command of the winget tool queries the sources for available applications that can be installed.

The **search** command can show all applications available, or it can be filtered down to a specific application. The **search** command is used typically to identify the string to use to install a specific application.

Usage

```
winget search [[-q] \<query>] [\<options>]
```



```
C:\Windows>winget search visual
Name                     Id                               Version
-----------------------------------------------------------------------
Visual Studio Code       Microsoft.VisualStudioCode       1.41.1
Visual Studio Community  Microsoft.VisualStudio.Community 16.0.30002.166

C:\Windows>cd \windows\system43
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--query	The query used to search for an app.
-?, --help	Gets additional help on this command.

Show all

If the search command includes no filters or options, it will display all available applications in the default source. You can also search for all applications in another source if you pass in just the **source** option.

Search strings

Search strings can be filtered with the following options.

OPTION	DESCRIPTION
--id	Limits the search to the ID of the application. The ID includes the publisher and the application name.
--name	Limits the search to the name of the application.
--moniker	Limits the search to the moniker specified.
--tag	Limits the search to the tags listed for the application.
--command	Limits the search to the commands listed for the application.

The string will be treated as a substring. The search by default is also case insensitive. For example, `winget search micro` could return the following:

- Microsoft
- microscope
- MyMicro

Search options

The search commands supports a number of options or filters to help limit the results.

OPTION	DESCRIPTION
-e, --exact	Uses the exact string in the query, including checking for case-sensitivity. It will not use the default behavior of a substring.
-n, --count	Restricts the output of the display to the specified count.
-s, --source	Restricts the search to the specified source name.

Related topics

- Use the winget tool to install and manage applications

settings command (winget)

5/26/2021 • 2 minutes to read • Edit Online

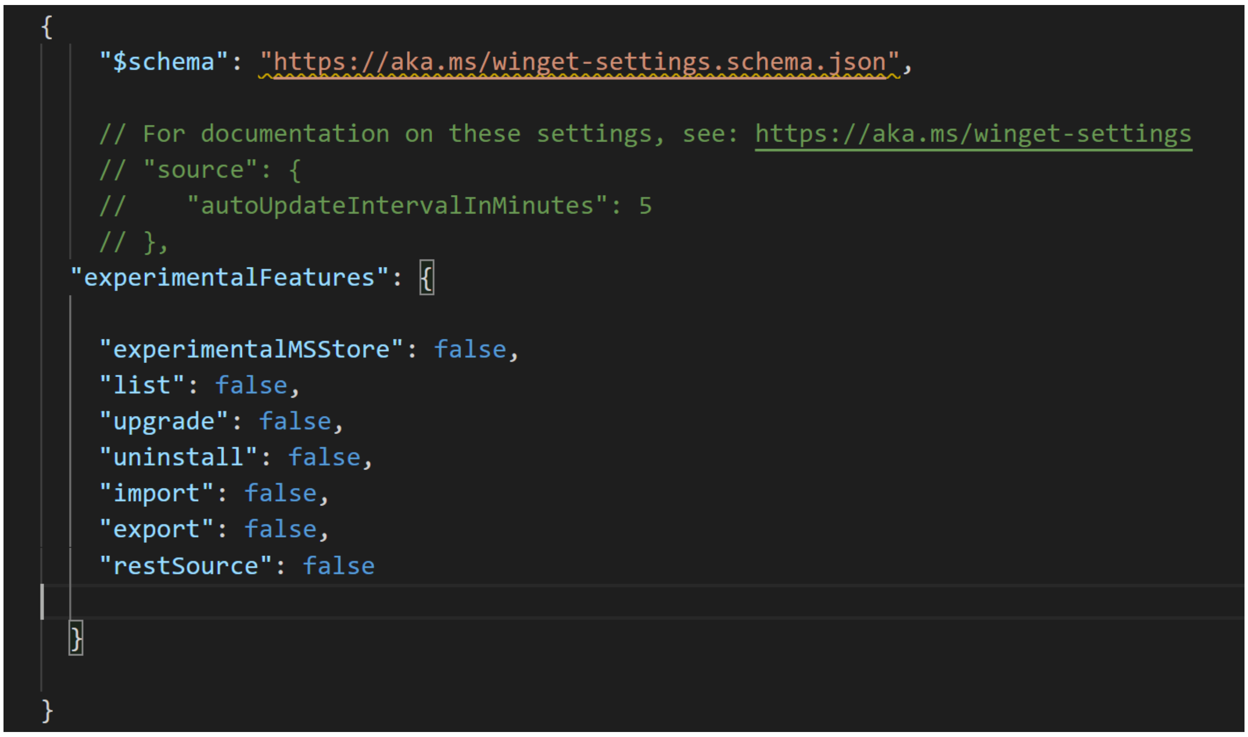
The **settings** command of the winget tool allows you to customize your Windows Package Manager client experience. You can change defaults and try out experimental features that are enabled in your client.

The **settings** command will launch your default text editor. Windows by default will launch Notepad as an option. We recommend using a tool like Visual Studio code.

NOTE
You can easily install Visual Studio Code by typing `winget install Microsoft.VisualStudioCode`

Usage

Launch your default JSON editing tool: `winget settings`



```
{
    "$schema": "https://aka.ms/winget-settings.schema.json",

    // For documentation on these settings, see: https://aka.ms/winget-settings
    // "source": {
    //    "autoUpdateIntervalInMinutes": 5
    // },
  "experimentalFeatures": {

    "experimentalMSStore": false,
    "list": false,
    "upgrade": false,
    "uninstall": false,
    "import": false,
    "export": false,
    "restSource": false

  }

}
```

When you launch the settings for the first time, there will be no settings specified. At the top of the JSON file we provide a link where you can discover the latest experimental features and settings.

We have also defined a schema for the settings file. This allows you to use TAB to discover settings and syntax if your JSON editor supports JSON schemas.

Updating settings

The following settings are available for the 1.0 release of the Windows Package Manager.

source

The `source` settings involve configuration to the WinGet source.

```
"source": {
    "autoUpdateIntervalInMinutes": 3
},
```

autoUpdateIntervalInMinutes

A positive integer represents the update interval in minutes. The check for updates only happens when a source is used. A zero will disable the check for updates to a source. Any other values are invalid.

- Disable: 0
- Default: 5

To manually update the source use `winget source update`.

visual

The `visual` settings involve visual elements that are displayed by WinGet

```
"visual": {
    "progressBar": "accent"
},
```

progressBar

Color of the progress bar that WinGet displays when not specified by arguments.

- accent (default)
- retro
- rainbow

installBehavior

The `installBehavior` settings affect the default behavior of installing and upgrading (where applicable) packages.

preferences and requirements

Some of the settings are duplicated under `preferences` and `requirements`.

- The `preferences` setting affects how the various available options are sorted when choosing the one to act on. For example, the default scope of package installs is for the current user, but if that is not an option then a machine level installer will be chosen.
- The `requirements` setting filters the options, potentially resulting in an empty list and a failure to install. In the previous example, a user scope requirement would result in no applicable installers and an error.

Any arguments passed on the command line will effectively override the matching `requirement` setting for the duration of that command.

scope

The `scope` behavior affects the choice between installing a package for the current user or for the entire machine. The matching parameter is `--scope`, and uses the same values (`user` or `machine`).

```
"installBehavior": {
    "preferences": {
        "scope": "user"
    }
},
```

locale

The `locale` behavior affects the choice of installer based on installer locale. The matching parameter is

`--locale`, and uses bcp47 language tag.

```
"installBehavior": {
    "preferences": {
        "locale": [ "en-US", "fr-FR" ]
    }
},
```

telemetry

The `telemetry` settings control whether winget writes ETW events that may be sent to Microsoft on a default installation of Windows.

See details on telemetry, and our primary privacy statement.

disable

```
"telemetry": {
    "disable": true
},
```

If set to true, the `telemetry.disable` setting will prevent any event from being written by the program.

network

The `network` settings influence how winget uses the network to retrieve packages and metadata.

downloader

The `downloader` setting controls which code is used when downloading packages. The default is `default`, which may be any of the options based on our determination.

`wininet` uses the WinINet APIs, while `do` uses the Delivery Optimization service.

```
"network": {
    "downloader": "do"
}
```

Enabling experimental features

To discover which experimental features are available, go to https://aka.ms/winget-settings where you can see the experimental features available to you.

show command (winget)

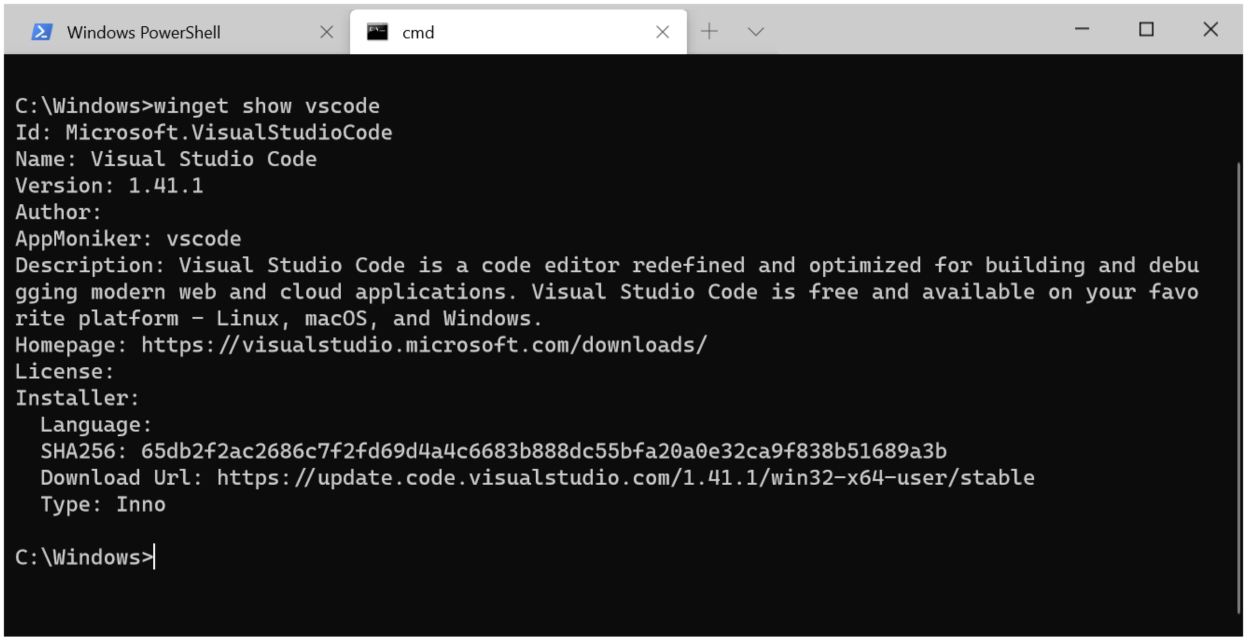
5/26/2021 • 2 minutes to read • Edit Online

The **show** command of the winget tool displays details for the specified application, including details on the source of the application as well as the metadata associated with the application.

The **show** command only shows metadata that was submitted with the application. If the submitted application excludes some metadata, then the data will not be displayed.

Usage

```
winget show [[-q] \<query>] [\<options>]
```



```
C:\Windows>winget show vscode
Id: Microsoft.VisualStudioCode
Name: Visual Studio Code
Version: 1.41.1
Author:
AppMoniker: vscode
Description: Visual Studio Code is a code editor redefined and optimized for building and debu
gging modern web and cloud applications. Visual Studio Code is free and available on your favo
rite platform - Linux, macOS, and Windows.
Homepage: https://visualstudio.microsoft.com/downloads/
License:
Installer:
  Language:
  SHA256: 65db2f2ac2686c7f2fd69d4a4c6683b888dc55bfa20a0e32ca9f838b51689a3b
  Download Url: https://update.code.visualstudio.com/1.41.1/win32-x64-user/stable
  Type: Inno

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--query	The query used to search for an application.
-?, --help	Gets additional help on this command.

Options

The following options are available.

OPTION	DESCRIPTION
-m,--manifest	The path to the manifest of the application to install.
--id	Filter results by ID.

OPTION	DESCRIPTION
--name	Filter results by name.
--moniker	Filter results by application moniker.
-v,--version	Use the specified version. The default is the latest version.
-s,--source	Find the application using the specified source.
-e,--exact	Find the application using exact match.
--versions	Show available versions of the application.

Multiple selections

If the query provided to **winget** does not result in a single application, then **winget** will display the results of the search. This will provide you with the additional data necessary to refine the search.

Results of show

If a single application is detected, the following data will be displayed.

Metadata

VALUE	DESCRIPTION
Id	Id of the application.
Name	Name of the application.
Publisher	Publisher of the application.
Version	Version of the application.
Author	Author of the application.
AppMoniker	AppMoniker of the application.
Description	Description of the application.
License	License of the application.
LicenseUrl	The URL to the license file of the application.
Homepage	Homepage of the application.
Tags	The tags provided to assist in searching.
Command	The commands supported by the application.
Channel	The details on whether the application is preview or release.

VALUE	DESCRIPTION
Minimum OS Version	The minimum OS version supported by the application.

Installer details

VALUE	DESCRIPTION
Arch	The architecture of the installer.
Language	The language of the installer.
Installer Type	The type of installer.
Download Url	The Url of the installer.
Hash	The Sha-256 of the installer.
Scope	Displays whether the installer is per machine or per user.

Related topics

- Use the winget tool to install and manage applications

source command (winget)

5/26/2021 • 2 minutes to read • Edit Online

NOTE

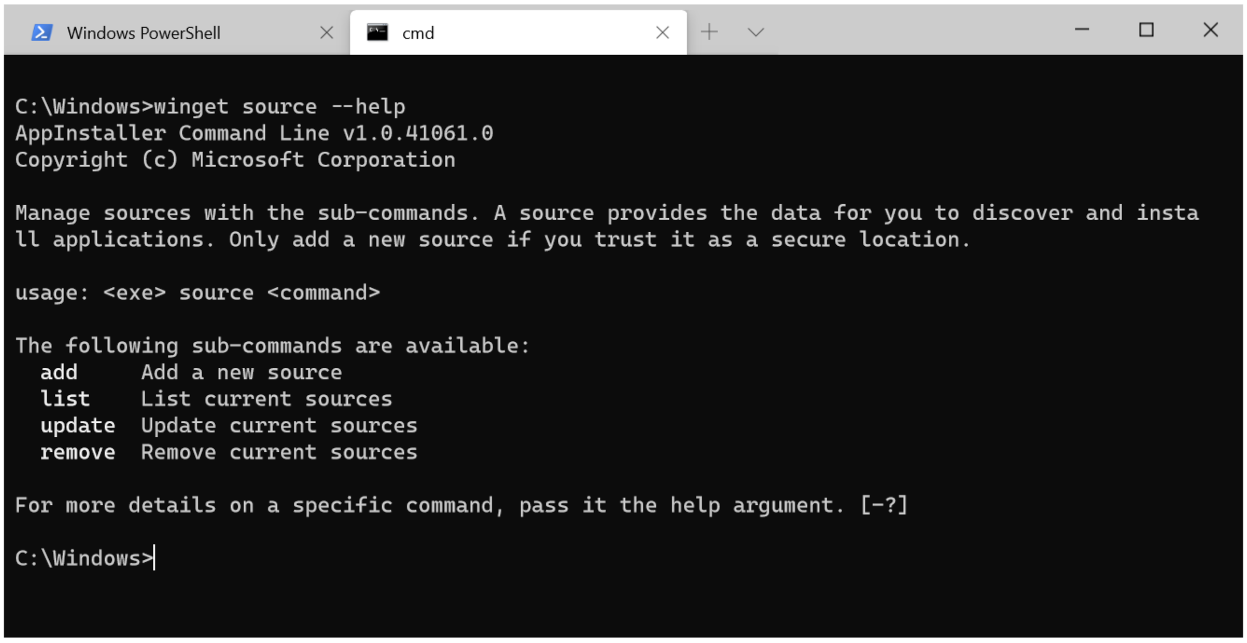
The **source** command is currently for internal use only. Additional sources are not supported at this time.

The **source** command of the winget tool manages the repositories accessed by Windows Package Manager. With the **source** command you can **add**, **remove**, **list**, and **update** the repositories.

A source provides the data for you to discover and install applications. Only add a new source if you trust it as a secure location.

Usage

```
winget source \<sub command> \<options>
```



```
C:\Windows>winget source --help
AppInstaller Command Line v1.0.41061.0
Copyright (c) Microsoft Corporation

Manage sources with the sub-commands. A source provides the data for you to discover and insta
ll applications. Only add a new source if you trust it as a secure location.

usage: <exe> source <command>

The following sub-commands are available:
  add      Add a new source
  list     List current sources
  update   Update current sources
  remove   Remove current sources

For more details on a specific command, pass it the help argument. [-?]

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-?, --**help**	Gets additional help on this command.

Sub commands

Source supports the following sub commands for manipulating the sources.

SUB COMMAND	DESCRIPTION
add	Adds a new source.

SUB COMMAND	DESCRIPTION
list	Enumerates the list of enabled sources.
update	Updates a source.
remove	Removes a source.
reset	Resets **winget** back to the initial configuration.

Options

The **source** command supports the following options.

OPTION	DESCRIPTION
-**n**,--**name**	The name to identify the source by.
-**a**,--**arg**	The URL or UNC of the source.
-**t**,--**type**	The type of source.
-?, --**help**	Gets additional help on this command.

add

The **add** sub command adds a new source. This sub command requires the --**name** option and the **name** argument.

Usage: `winget source add [-n, --name] \<name> [-a] \<url> [[-t] \<type>]`

Example: `winget source add --name Contoso https://www.contoso.com/cache`

The **add** sub command also supports the optional **type** parameter. The **type** parameter communicates to the client what type of repository it is connecting to. The following types are supported.

TYPE	DESCRIPTION
Microsoft.PreIndexed.Package	The type of source <default>.

list

the **list** sub command enumerates the currently enabled sources. This sub-command also provides details on a specific source.

Usage: `winget source list [-n, --name] \<name>`

list all

The **list** sub-command by itself will reveal the complete list of supported sources. For example:

```
> C:\winget source list
> Name   Arg
> -----------------------------------------
> winget https://winget.azureedge.net/cache
```

list source details

In order to get complete details on the source, pass in the name used to identify the source. For example:

```
> C:\winget source list --name contoso
> Name   : contoso
> Type   : Microsoft.PreIndexed.Package
> Arg    : https://pkgmgr-int.azureedge.net/cache
> Data   : AppInstallerSQLiteIndex-int_g4ype1skzj3jy
> Updated: 2020-4-14 17:45:32.000
```

Name displays the name to identify the source by. **Type** displays the type of repo. **Arg** displays the URL or path used by the source. **Data** displays the optional package name used if appropriate. **Updated** displays the last date and time the source was updated.

update

The **update** sub command forces an update to an individual source or for all.

usage: `winget source update [-n, --name] \<name>`

update all

The **update** sub command by itself will request and update to each repo. For example: `C:\winget update`

update source

The **update** sub command combined with the **--name** option can direct and update to an individual source. For example: `C:\winget source update --name contoso`

remove

The **remove** sub command removes a source. This sub command requires the **--name** option and **name argument** in order to identify the source.

Usage: `winget source remove [-n, --name] \<name>`

For example: `winget source remove --name Contoso`

reset

The **reset** sub-command resets the client back to its original configuration. The **reset** sub-command removes all sources and sets the source to the default. This sub command should only be used in rare cases.

Usage: `winget source reset`

For example: `winget source reset`

Default repository

Windows Package Manager specifies a default repository. You can identify the repository by using the **list** command. For example: `winget source list`

Related topics

- Use the winget tool to install and manage applications

uninstall command (winget)

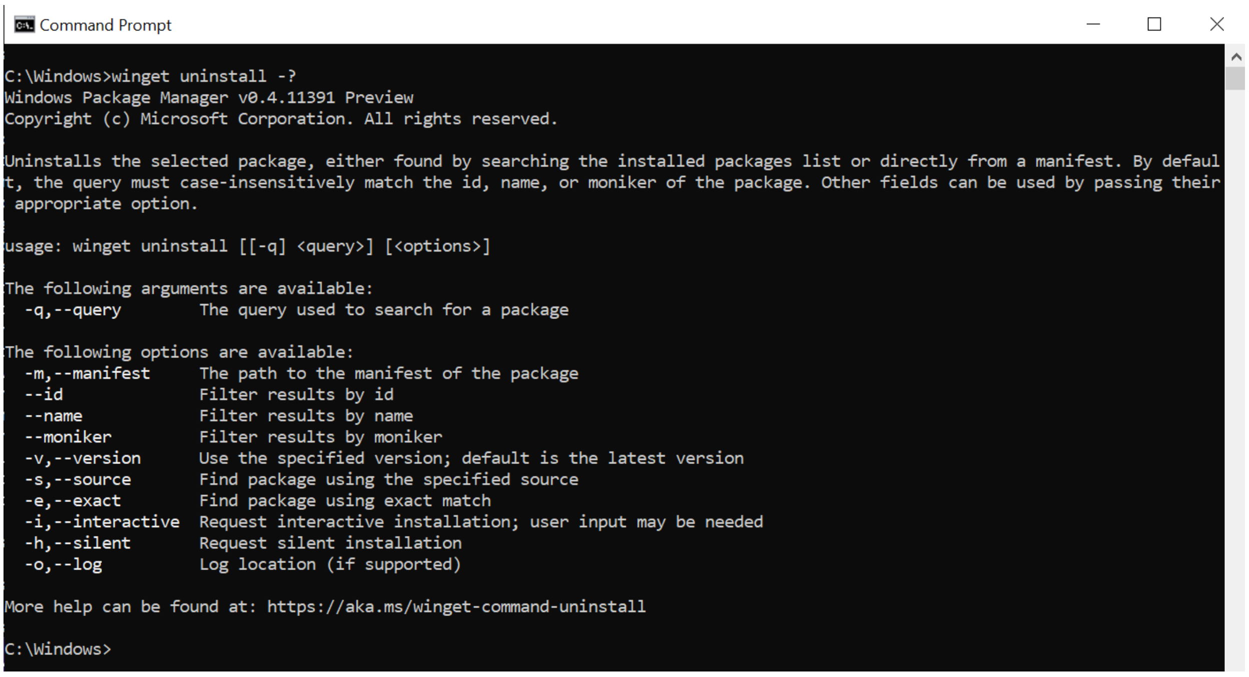
5/26/2021 • 2 minutes to read • Edit Online

The **uninstall** command of the winget tool uninstalls the specified application.

The **uninstall** command requires that you specify the exact string to uninstall. If there is any ambiguity, you will be prompted to further filter the **uninstall** command to an exact application.

Usage

```
winget uninstall [[-q] \<query>] [\<options>]
```



```
C:\Windows>winget uninstall -?
Windows Package Manager v0.4.11391 Preview
Copyright (c) Microsoft Corporation. All rights reserved.

Uninstalls the selected package, either found by searching the installed packages list or directly from a manifest. By defaul
t, the query must case-insensitively match the id, name, or moniker of the package. Other fields can be used by passing their
 appropriate option.

usage: winget uninstall [[-q] <query>] [<options>]

The following arguments are available:
  -q,--query        The query used to search for a package

The following options are available:
  -m,--manifest     The path to the manifest of the package
  --id              Filter results by id
  --name            Filter results by name
  --moniker         Filter results by moniker
  -v,--version      Use the specified version; default is the latest version
  -s,--source       Find package using the specified source
  -e,--exact        Find package using exact match
  -i,--interactive  Request interactive installation; user input may be needed
  -h,--silent       Request silent installation
  -o,--log          Log location (if supported)

More help can be found at: https://aka.ms/winget-command-uninstall

C:\Windows>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--query	The query used to search for an app.
-?, --help	Get additional help on this command.

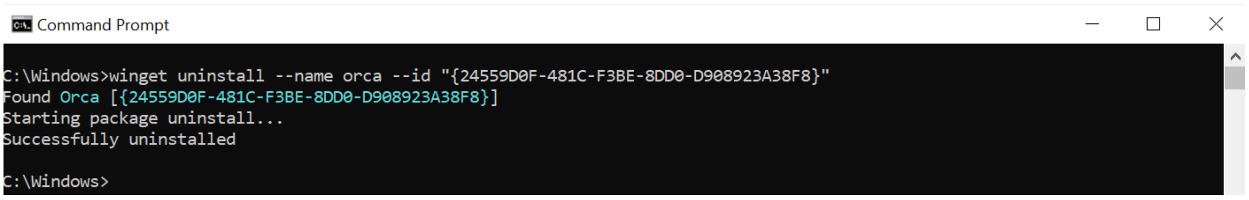
Options

The options allow you to customize the uninstall experience to meet your needs.

OPTION	DESCRIPTION
-m, --manifest	Must be followed by the path to the manifest (YAML) file. You can use the manifest to run the uninstall experience from a local YAML file.

OPTION	DESCRIPTION
--id	Limits the uninstall to the ID of the application.
--name	Limits the search to the name of the application.
--moniker	Limits the search to the moniker listed for the application.
-v, --version	Enables you to specify an exact version to uninstall. If not specified, latest will uninstall the highest versioned application.
-s, --source	Restricts the search to the source name provided. Must be followed by the source name.
-e, --exact	Uses the exact string in the query, including checking for case-sensitivity. It will not use the default behavior of a substring.
-i, --interactive	Runs the uninstaller in interactive mode. The default experience shows uninstaller progress.
-h, --silent	Runs the uninstaller in silent mode. This suppresses all UI. The default experience shows uninstaller progress.
-o, --log	Directs the logging to a log file. You must provide a path to a file that you have the write rights to.

After you have successfully identified the application intended to uninstall, winget will execute the uninstall command. In the example below, the **name** 'orca' and the **id** was passed in.



```
C:\Windows>winget uninstall --name orca --id "{24559D0F-481C-F3BE-8DD0-D908923A38F8}"
Found Orca [{24559D0F-481C-F3BE-8DD0-D908923A38F8}]
Starting package uninstall...
Successfully uninstalled

C:\Windows>
```

Example queries

The following example uninstalls a specific version of an application.

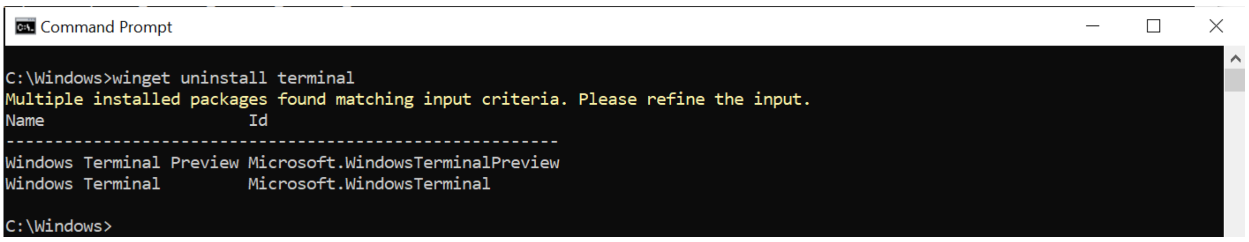
```
winget uninstall --name powertoys --version 0.15.2
```

The following example uninstalls an application using its ID.

```
winget uninstall --id "{24559D0F-481C-F3BE-8DD0-D908923A38F8}"
```

Multiple selections

If the query provided to **winget** does not result in a single application to uninstall, then **winget** will display multiple results. You can then use additional filters to refine the search for a correct application.



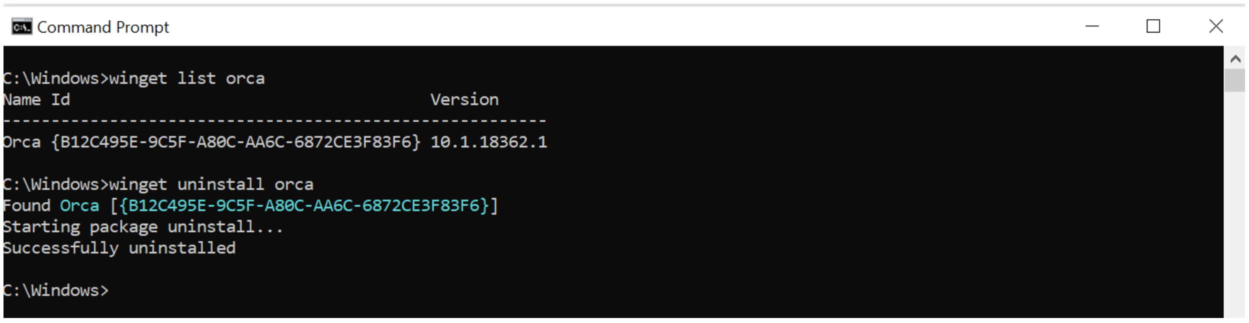
Command Prompt

```
C:\Windows>winget uninstall terminal
Multiple installed packages found matching input criteria. Please refine the input.
Name                     Id
--------------------------------------------------------
Windows Terminal Preview Microsoft.WindowsTerminalPreview
Windows Terminal         Microsoft.WindowsTerminal

C:\Windows>
```

Uninstalling apps not installed with Windows Package Manager

As mentioned in list, the **winget list** command will display more than just apps installed with the **winget**. Therefore you can use these commands to quickly and easily remove apps from your PC.

In this example, **list** was used to find the application, and then the **id** was passed in as part of uninstall.



Command Prompt

```
C:\Windows>winget list orca
Name Id                                     Version
-------------------------------------------------------
Orca {B12C495E-9C5F-A80C-AA6C-6872CE3F83F6} 10.1.18362.1

C:\Windows>winget uninstall orca
Found Orca [{B12C495E-9C5F-A80C-AA6C-6872CE3F83F6}]
Starting package uninstall...
Successfully uninstalled

C:\Windows>
```

upgrade command (winget)

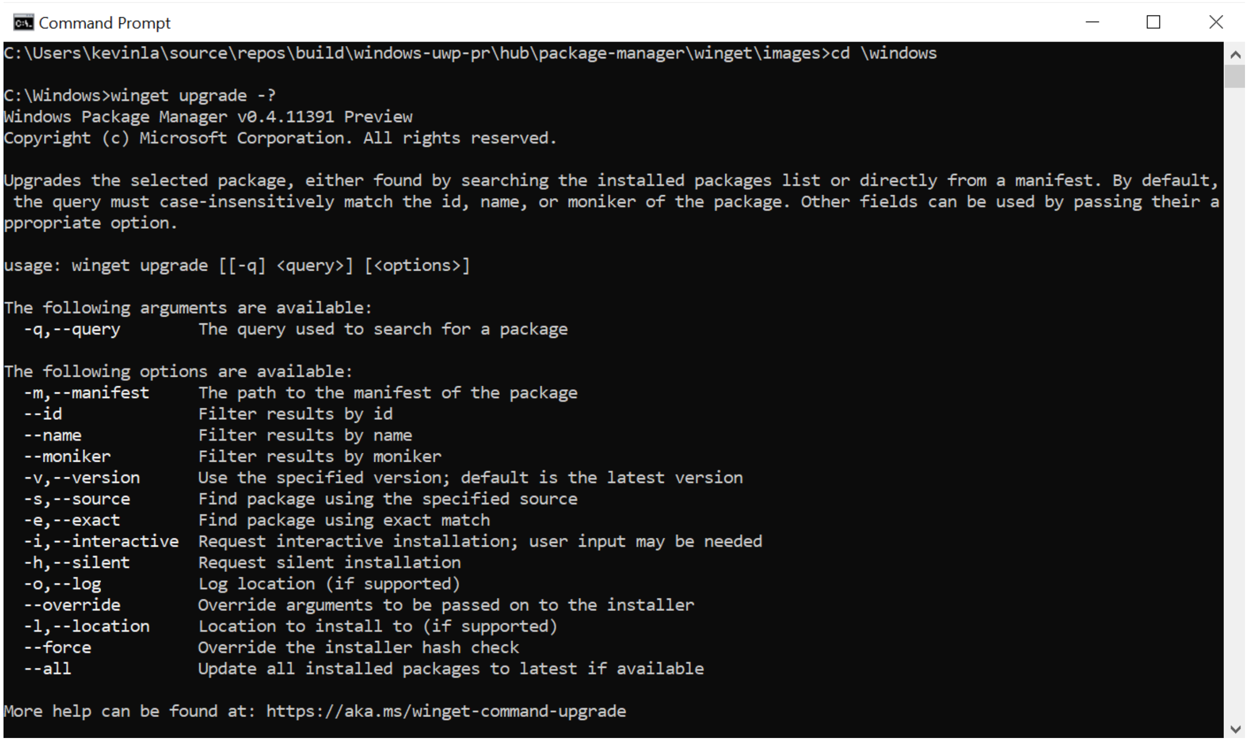
6/3/2021 • 2 minutes to read • Edit Online

The **upgrade** command of the winget tool upgrades the specified application. Optionally, you may use the **list** command to identify the application you want to upgrade.

The **upgrade** command requires that you specify the exact string to upgrade. If there is any ambiguity, you will be prompted to further filter the **upgrade** command to an exact application.

Usage

```
winget upgrade [[-q] \<query>] [\<options>]
```



```
C:\Users\kevinla\source\repos\build\windows-uwp-pr\hub\package-manager\winget\images>cd \windows

C:\Windows>winget upgrade -?
Windows Package Manager v0.4.11391 Preview
Copyright (c) Microsoft Corporation. All rights reserved.

Upgrades the selected package, either found by searching the installed packages list or directly from a manifest. By default,
 the query must case-insensitively match the id, name, or moniker of the package. Other fields can be used by passing their a
ppropriate option.

usage: winget upgrade [[-q] <query>] [<options>]

The following arguments are available:
  -q,--query        The query used to search for a package

The following options are available:
  -m,--manifest     The path to the manifest of the package
  --id              Filter results by id
  --name            Filter results by name
  --moniker         Filter results by moniker
  -v,--version      Use the specified version; default is the latest version
  -s,--source       Find package using the specified source
  -e,--exact        Find package using exact match
  -i,--interactive  Request interactive installation; user input may be needed
  -h,--silent       Request silent installation
  -o,--log          Log location (if supported)
  --override        Override arguments to be passed on to the installer
  -l,--location     Location to install to (if supported)
  --force           Override the installer hash check
  --all             Update all installed packages to latest if available

More help can be found at: https://aka.ms/winget-command-upgrade
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
-q,--query	The query used to search for an app.
-?, --help	Get additional help on this command.

Options

The options allow you to customize the upgrade experience to meet your needs.

OPTION	DESCRIPTION

OPTION	DESCRIPTION
-m, --manifest	Must be followed by the path to the manifest (YAML) file. You can use the manifest to run the upgrade experience from a local YAML file.
--id	Limits the upgrade to the ID of the application.
--name	Limits the search to the name of the application.
--moniker	Limits the search to the moniker listed for the application.
-v, --version	Enables you to specify an exact version to upgrade. If not specified, latest will upgrade the highest versioned application.
-s, --source	Restricts the search to the source name provided. Must be followed by the source name.
-e, --exact	Uses the exact string in the query, including checking for case-sensitivity. It will not use the default behavior of a substring.
-i, --interactive	Runs the installer in interactive mode. The default experience shows installer progress.
-h, --silent	Runs the installer in silent mode. This suppresses all UI. The default experience shows installer progress.
-o, --log	Directs the logging to a log file. You must provide a path to a file that you have the write rights to.
--override	A string that will be passed directly to the installer.
-l, --location	Location to upgrade to (if supported).
--force	When a hash mismatch is discovered will ignore the error and attempt to install the package.
--all	Updates all available packages to the latest application.

Example queries

The following example upgrades a specific version of an application.

```
winget upgrade powertoys --version 0.15.2
```

The following example upgrades an application from its ID.

```
winget upgrade --id Microsoft.PowerToys
```

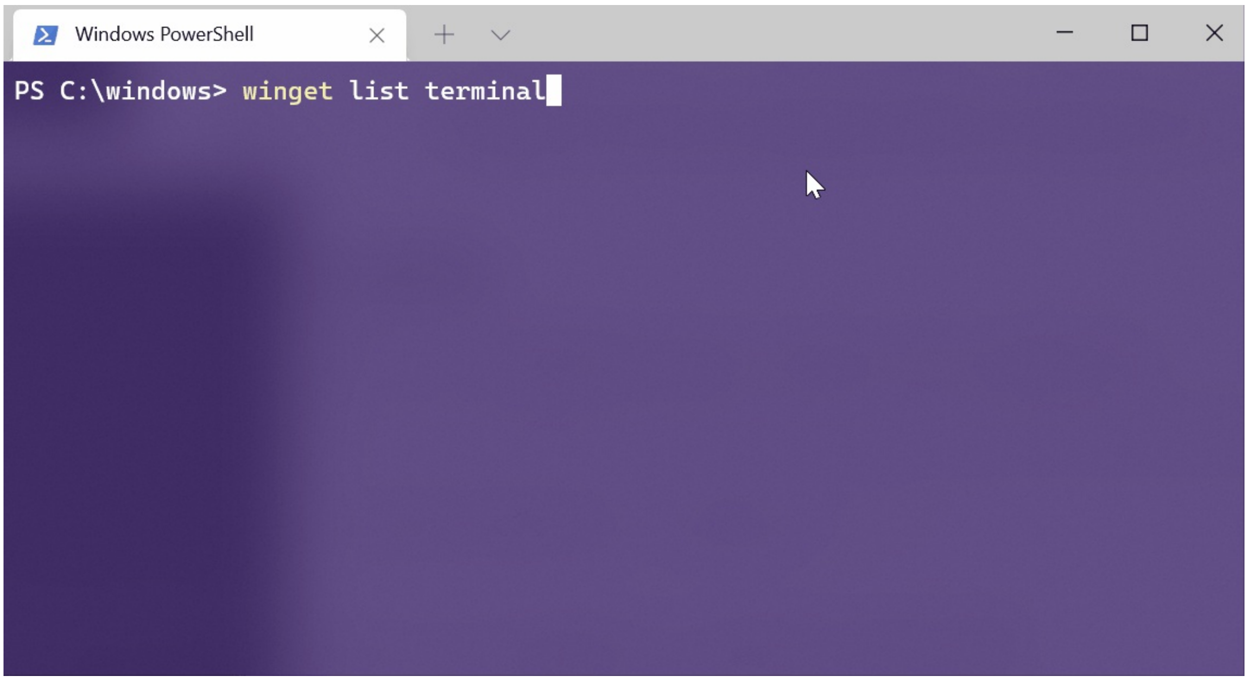
The following example shows upgrading all apps

```
winget upgrade --all
```

Using list and upgrade

It is common to use the list command to identify apps in need of an update, and then to use **upgrade** to install the latest.

In the example below you will see list identifies that an update is available for Microsoft.WindowsTerminalPreview, and then the user uses **upgrade** to update the application.



```
PS C:\windows> winget list terminal
```

upgrade --all

upgrade --all will identify all the applications with upgrades available. When you run **winget upgrade --all** the Windows Package Manager will look for all applications that have updates available and attempt to install the.

> **NOTE**
>
> Some applications do not provide a version. They are always latest. Because the Windows Package Manager cannot identify if there is a newer version of the app, an upgrade will not be possible.

validate command (winget)

5/26/2021 • 2 minutes to read • Edit Online

The **validate** command of the winget tool validates a manifest for submitting software to the **Microsoft Community Package Manifest Repository** on GitHub. The manifest must be a YAML file that follows the specification.

Usage

```
winget validate [--manifest] \<path>
```

Arguments

The following arguments are available.

ARGUMENT	DESCRIPTION
--manifest	the path to the manifest to be validated.
-?, --help	get additional help on this command

Related topics

- Use the winget tool to install and manage applications
- Submit packages to Windows Package Manager

Submit packages to Windows Package Manager

5/26/2021 • 2 minutes to read • Edit Online

This section provides guidance about submitting packages to Windows Package Manager.

Independent Software Vendor (ISV) or Publisher

If you are an ISV or Publisher, you can use Windows Package Manager as a distribution channel for software packages containing your applications. Windows Package Manager currently supports installers in the following formats: MSIX, MSI, and EXE.

To submit software packages to Windows Package Manager, follow these steps:

1. Create a package manifest that provides information about your application. Manifests are YAML files that follow the Windows Package Manager schema.
2. Submit your manifest to the Windows Package Manager repository. This is an open source repository on GitHub that contains a collection of manifests that the **winget** tool can access.

Community member

If you are a GitHub community member, you may also submit packages to Windows Package Manager following the steps above.

Optionally, you may also request help to have a package added to the community repository. To do so, create a new Package Request/Submission issue.

Related topics

- Use the winget tool
- Create your package manifest
- Submit your manifest to the repository

Create your package manifest

7/6/2021 • 6 minutes to read • Edit Online

If you want to submit a software package to the Windows Package Manager repository, start by creating a package manifest. The manifest is a YAML file that describes the application to be installed.

This article describes the contents of a package manifest for Windows Package Manager.

YAML basics

The YAML format was chosen for package manifests because of its relative ease of human readability and consistency with other Microsoft development tools. If you are not familiar with YAML syntax, you can learn the basics at Learn YAML in Y Minutes.

NOTE

Manifests for Windows Package Manager currently do not support all YAML features. Unsupported YAML features include anchors, complex keys, and sets.

Conventions

These conventions are used in this article:

- To the left of `:` is a literal keyword used in manifest definitions.
- To the right of `:` is a data type. The data type can be a primitive type like **string** or a reference to a rich structure defined elsewhere in this article.
- The notation `[` *datatype* `]` indicates an array of the mentioned data type. For example, `[ string ]` is an array of strings.
- The notation `{` *datatype* `:` *datatype* `}` indicates a mapping of one data type to another. For example, `{ string: string }` is a mapping of strings to strings.

Manifest contents

A package manifest must include a set of required items, and can also include further optional items that can help improve the customer experience of installing your software. This section provides brief summaries of the required manifest schema and complete manifest schemas, and examples of each.

Each field in the manifest file must be Pascal-cased and cannot be duplicated.

For a complete list and descriptions of items in a manifest, see the manifest specification in the https://github.com/microsoft/winget-cli repository.

Minimal required schema

As specified in the singleton JSON schema, only certain fields are required. The minimal supported YAML file would look like the example below. The singleton format is only valid for packages containing a single installer and a single locale. If more than one installer or locale is provided, the multiple YAML file format and schema must be used.

The partitioning scheme was added to help with GitHub's UX. Folders with thousands of children do not render well in the browser.

- Minimal required schema
- Example

```
PackageIdentifier:  # Publisher.package format.
PackageVersion:     # Version numbering format.
PackageLocale:      # BCP 47 format (e.g. en-US)
Publisher:          # The name of the publisher.
PackageName:        # The name of the application.
License:            # The license of the application.
ShortDescription:   # The description of the application.
Installers:
 - Architecture:    # Enumeration of supported architectures.
   InstallerType:   # Enumeration of supported installer types (exe, msi, msix, inno, wix, nullsoft, appx).
   InstallerUrl:    # Path to download installation file.
   InstallerSha256: # SHA256 calculated from installer.
ManifestType:       # The manifest file type
ManifestVersion: 1.0.0
```

Multiple manifest files

To provide the best user experience, manifests should contain as much meta-data as possible. In order to separate concerns for validating installers and providing localized metadata, manifests should be split into multiple files. The minimum number of YAML files for this kind of manifest is three. Additional locales should also be provided.

- A version file.
- The default locale file.
- An installer file.
- Additional locale files.

The example below shows many optional metadata fields and multiple locales. Note the default locale has more requirements than additional locales. In the show command, any required fields that aren't provided for additional locales will display fields from the default locale.

- Version file example
- Default locale file example
- Additional locale file example
- Installer file example

Path: manifests / m / Microsoft / WindowsTerminal / 1.6.10571.0 / Microsoft.WindowsTerminal.yaml

```
PackageIdentifier: "Microsoft.WindowsTerminal"
PackageVersion: "1.6.10571.0"
DefaultLocale: "en-US"
ManifestType: "version"
ManifestVersion: "1.0.0"
```

NOTE

If your installer is an .exe and it was built using Nullsoft or Inno, you may specify those values instead. When Nullsoft or Inno are specified, the client will automatically set the silent and silent with progress install behaviors for the installer.

Installer switches

You can often figure out what silent `Switches` are available for an installer by passing in a `-?` to the installer from the command line. Here are some common silent `Switches` that can be used for different installer types.

INSTALLER	COMMAND	DOCUMENTATION
MSI	`/q`	MSI Command-Line Options
InstallShield	`/s`	InstallShield Command-Line Parameters
Inno Setup	`/SILENT or /VERYSILENT`	Inno Setup documentation
Nullsoft	`/S`	Nullsoft Silent Installers/Uninstallers

Tips and best practices

- The package identifier must be unique. You cannot have multiple submissions with the same package identifier. Only one pull request per package version is allowed.
- Avoid creating multiple publisher folders. For example, do not create "Contoso Ltd." if there is already a "Contoso" folder.
- All tools must support a silent install. If you have an executable that does not support a silent install, then we cannot provide that tool at this time.
- Provide as many fields as possible. The more meta-data you provide the better the user experience will be. In some cases, the fields may not yet be supported by the Windows Package Manager client (winget.exe). For example, the `AppMoniker` field is optional. However, if you include this field, customers will see results associated with the `AppMoniker` value when performing the search command (for example, **vscode** for **Visual Studio Code**). If there is only one app with the specified `AppMoniker` value, customers can install your application by specifying the moniker rather than the fully qualified package identifier.
- The length of strings in this specification should be limited to 100 characters before a line break.
- The `PackageName` should match the entry made in **Add / Remove Programs** to help the correlation with manifests to support **export**, and **upgrade**.
- The `Publisher` should match the entry made in **Add / Remove Programs** to help the correlation with manifests to support **export**, and **upgrade**.
- Package installers in MSI format use product codes to uniquely identify applications. The product code for a given version of a package should be included in the manifest to help ensure the best **upgrade** experience.
- Limit the length of strings in your manifest to 100 characters before a line break.
- When more than one installer type exists for the specified version of the package, an instance of `InstallerType` can be placed under each of the `Installers`.

Submit your manifest to the repository

5/26/2021 • 3 minutes to read • Edit Online

After you create a package manifest that describes your application, you're ready to submit your manifest to the Windows Package Manager repository. This a public-facing repository that contains a collection of manifests that the **winget** tool can access. To submit your manifest, you'll upload it to the open source https://github.com/microsoft/winget-pkgs repository on GitHub.

After you submit a pull request to add a new manifest to the GitHub repository, an automated process will validate your manifest file and check to make sure the package complies with the Windows Package Manager polices and is not known to be malicious. If this validation is successful, your package will be added to the public-facing Windows Package Manager repository so it can be discovered by the **winget** client tool. Note the distinction between the manifests in the open source GitHub repository and the public-facing Windows Package Manager repository.

IMPORTANT

Microsoft reserves the right to refuse a submission for any reason.

Third-party repositories

There are currently no known third party repositories. Microsoft is working with multiple partners to develop protocols or an API to enable third party repositories.

Manifest validation

When you submit a manifest to the https://github.com/microsoft/winget-pkgs repository on GitHub, your manifest will be automatically validated and evaluated for the safety of the Windows ecosystem. Manifests may also be reviewed manually.

For more information about the validation process, see Windows Package Manager validation process.

How to submit your manifest

To submit a manifest to the repository, follow these steps.

Step 1: Validate your manifest

The **winget** tool provides the validate command to confirm that you have created your manifest correctly. To validate your manifest, use this command.

```
winget validate \<manifest-file>
```

If your validation fails, use the errors to locate the line number and make a correction. After your manifest is validated, you can submit it to the repository.

Step 2: Clone the repository

Next, create a fork of the repository and clone it.

1. Go to https://github.com/microsoft/winget-pkgs in your browser and click **Fork**.

2. From a command line environment such as the Windows Command Prompt or PowerShell, use the following command to clone your fork.

```
git clone \<your-fork-name>
```

3. If you are making multiple submissions, make a branch instead of a fork. We currently allow only one manifest file per submission.

```
git checkout -b \<branch-name>
```

Step 3: Add your manifest to the local repository

You must add your manifest file to the repository in the following folder structure:

manifests / letter / publisher / application / version / Yaml file

- The **manifests** folder is the root folder for all manifests in the repository.
- The **letter** folder is the first letter of the publisher name.
- The **publisher** folder is the name of the company that publishes the software. For example, **Microsoft**.
- The **application** folder is the name of the application or tool. For example, **VSCode**.
- The **version** folder is the version of the application or tool. For example, **1.0.0**.
- **Yaml File** is the file name of the manifest. The file name must be set to the name and publisher of the application. For example, **Contoso.ContosoApp.yaml**.

The `PackageIdentifier` value in the manifest must match the publisher and application names in the manifest folder path, and the `PackageVersion` value in the manifest must match the version in the file name. For more information, see Create your package manifest.

Step 4: Submit your manifest to the remote repository

You're now ready to push your new manifest to the remote repository.

1. Use the `add` command to prepare for submission.

```
git add manifests\C\Contoso\ContosoApp\1.0.0\Contoso.ContosoApp.yaml
```

2. Use the `commit` command to commit the change and provide information on the submission.

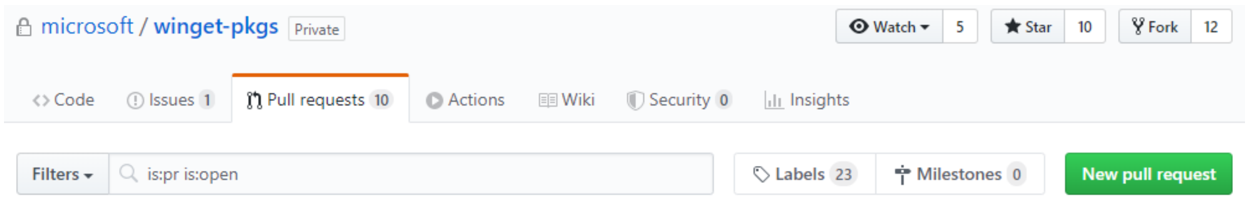
```
git commit -m "Submitting ContosoApp version 1.0.0.yaml"
```

3. Use the `push` command to push the changes to the remote repository.

```
git push
```

Step 5: Create a pull request

After you push your changes, return to https://github.com/microsoft/winget-pkgs and create a pull request to merge your fork or branch to the main branch.



Submission process

When you create a pull request, this will start an automated process that validates the manifests and verifies your pull request. During this process we will run tests against the installer and installed binaries to validate the submission.

We add labels to your pull request so you can track its progress. For more information, see Validation process.

Once complete, the submission will be automatically merged and the application will get added to the Windows Package Manager catalog.

If there is an error during the process, you will be notified and our labels and bot will assist you in fixing your submission. For the list of common errors, see Validation process.

Windows Package Manager validation process

5/26/2021 • 7 minutes to read • Edit Online

When you create a pull request to submit your manifest to the Windows Package Manager repository, this will start an automation process that validates the manifest and processes your pull request. GitHub labels are used to share progress and allow you to communicate with us.

Submission expectations

All application submissions to the Windows Package Manager repository should be well-behaved and adhere to the Windows Package Manager repository policies. Here are some expectations for submissions:

- The manifest complies with the schema requirements.
- All URLs in the manifest lead to safe websites.
- The installer and application are virus free. The package may be identified as malware by mistake. If you believe it is a false positive you can submit the installer to the Microsoft Defender team for analysis.
- The application installs and uninstalls correctly for both administrators and non-administrators.
- The installer supports non-interactive modes.
- All manifest entries are accurate and not misleading.
- The installer comes directly from the publisher's website.

For a complete list of the policies, see Windows Package Manager policies.

Pull request labels

During validation, we apply a series of labels to pull requests to communicate progress. Some labels will direct you to take action, while others will be directed to the Windows Package Manager engineering team.

Status labels

The following table describes the **status labels** you might encounter.

LABEL	DETAILS
Azure-Pipeline-Passed	The manifest has completed the test pass. It is waiting for approval. If no issues are encountered during the test pass it will automatically be approved. If a test fails, it may be flagged for manual review.
Blocking-Issue	This label indicates that the pull request cannot be approved because there is a blocking issue. You can often tell what the blocking issue is by the included error label.
Needs: Attention	This label indicates that the pull request needs to be investigated by the Windows Package Manager development team. This is either due to a test failure that needs manual review, or a comment added to the pull request by the community.

LABEL	DETAILS
Needs: author feedback	Indicates there is a failure with the submission. We will reassign the pull request back to you. If you do not address the issue within 10 days, the bot will close the pull request. **Needs: author feedback** labels are typically added when there was a failure with the pull request that should be updated, or if the person reviewing the pull request has a question.
Validation-Completed	Indicates that the test pass has been completed successfully and your pull request will be merged.

Error labels

The following table describes the **error labels** you might encounter. Not all of the error cases will be assigned to you immediately. Some may trigger manual validation.

LABEL	DETAILS
Binary-Validation-Error	The application included in this pull request failed to pass the **Installers Scan** test. This test is designed to ensure that the application installs on all environments without warnings. For more details on this error, see Binary validation errors.
Error-Analysis-Timeout	The **Binary-Validation-Test** test timed out. The pull request will get assigned to a Windows Package Manager engineer to investigate.
Error-Hash-Mismatch	The submitted manifest could not be processed because the **InstallerSha256** hash provided for the **InstallerURL** did not match. Update the **InstallerSha256** in the pull request and try again.
Error-Installer-Availability	The validation service was unable to download the installer. This may be related to Azure IP ranges being blocked, or the installer URL may be incorrect. Check that the **InstallerURL** is correct and try again. If you feel this has failed in error, add a comment and the pull request will get assigned to a Windows Package Manager engineer to investigate.
Manifest-Path-Error	The manifest files must be put into a specific folder structure. This label indicates a problem with the path of your submission. For example, the folder structure does not have the required format. Update your manifest and path resubmit your pull request.
Manifest-Validation-Error	The submitted manifest contains a syntax error. Address the syntax issue with the manifest and re-submit. For details on the manifest format and schema, see required format.
PullRequest-Error	The pull request is invalid because not all files submitted are under manifest folder or there is more than one package or version in the pull request. Update your pull request to address the issue and try again.

LABEL	DETAILS
URL-Validation-Error	The **URLs Validation Test** could not locate the URL and responded with a HTTP error status code (403 or 404), or the URL reputation test failed. You can identify which URL is in question by looking at the pull request check details. To address this issue, update the URLs in question to resolve the HTTP error status code. If the issue is not due to an HTTP error status code, you can submit the URL for review to avoid the reputation failure.
Validation-Defender-Error	During dynamic testing, Microsoft Defender reported a problem. To reproduce this problem, install your application, then run a Microsoft Defender full scan. If you can reproduce the problem, fix the binary or submit it for analysis for false positive assistance. If you are unable to reproduce the problem, add a comment to get the Windows Package Manager engineers to investigate.
Validation-Domain	The test has determined the domain if the **InstallerURL** does not match the domain expected. The Windows Package Manager policies requires that the InstallerUrl comes directly from the ISV's release location. If you believe this is a false detection, add a comment to the pull request to get the Windows Package Manager engineers to investigate.
Validation-Error	Validation of the Windows Package Manager failed during manual approval. Look at the accompanying comment for next steps.
Validation-Executable-Error	During installation testing, the test was unable to locate the primary application. Make sure the application installs correctly on all platforms. If your application does not install an application, but should still be included in the repository, add a comment to the pull request to get the Windows Package Manager engineers to investigate.
Validation-Hash-Verification-Failed	During installation testing, the application fails to install because the **InstallerSha256** no longer matches the **InstallerURL** hash. This can occur if the application is behind a vanity URL and the installer was updated without updating the **InstallerSha256**. To address this issue, update the **InstallerSha256** associated with the **InstallerURL** and submit again.
Validation-HTTP-Error	The URL used for the installer does not use the HTTPS protocol. Update the **InstallerURL** to use HTTPS and resubmit the **Pull Request.**
Validation-Indirect-URL	The URL is not coming directly from the ISVs server. Testing has determined a redirector has been used. This is not allowed because the Windows Package Manager policies require that the InstallerUrl comes directly from the ISV's release location. Remove the redirection and resubmit.
Validation-Installation-Error	During manual validation of this package, there was a general error. Look at the accompanying comment for next steps.

LABEL	DETAILS
Validation-Merge-Conflict	This package could not be validated due to a merge conflict. Please address the merge conflict and resubmit your pull request.
Validation-MSIX-Dependency	The MSIX package has a dependency on package that could not be resolved. Update the package to include the missing components or add the dependency to the manifest file and resubmit the pull request.
Validation-Unapproved-URL	The test has determined the domain if the **InstallerURL** does not match the domain expected. The Windows Package Manager policies requires that the InstallerUrl comes directly from the ISV's release location.
Validation-Unattended-Failed	During installation, the test timed out. This most likely is due to the application not installing silently. It could also be due to some other error being encountered and stopping the test. Verify that you can install your manifest without user input. If you need assistance, add a comment to the pull request and the Windows Package Manager engineers will investigate.
Validation-Uninstall-Error	During uninstall testing, the application did not clean up completely following uninstall. Look at the accompanying comment for more details.
Validation-VCRuntime-Dependency	The package has a dependency on the C++ runtime that could not be resolved. Update the package to include the missing components or add the dependency to the manifest file and resubmit the pull request.

Content policy labels

The following table lists **content policy labels**. If one of these labels is added, something in the manifest metadata triggered additional manual content review to ensure that the metadata is following the Windows Package Manager policies.

LABEL	DETAILS
Policy-Test-2.1	See General Content Requirements.
Policy-Test-2.2	See Content Including Names, Logos, Original and Third Party
Policy-Test-2.3	See Risk of Harm.
Policy-Test-2.4	See Defamatory, Libelous, Slanderous and Threatening.
Policy-Test-2.5	See Offensive Content.
Policy-Test-2.6	See Alcohol, Tobacco, Weapons and Drugs.
Policy-Test-2.7	See Adult Content.
Policy-Test-2.8	See Illegal Activity.

LABEL	DETAILS
Policy-Test-2.9	See Excessive Profanity and Inappropriate Content.
Policy-Test-2.10	See Country/Region Specific Requirements.
Policy-Test-2.11	See Age Ratings.
Policy-Test-2.12	See User Generated Content.

Internal labels

The following table lists internal error labels. When internal errors are encountered, your pull request will be assigned to the Windows Package Manager engineers to investigate.

LABEL	DETAILS
Internal-Error-Domain	An error occurred during the domain validation of the URL.
Internal-Error-Dynamic-Scan	An error occurred during the validation of the installed binaries.
Internal-Error-Keyword-Policy	An error occurred during the validation of the manifest.
Internal-Error-Manifest	An error occurred during the validation of the manifest.
Internal-Error-NoArchitectures	An error occurred because the test could not determine the architecture if the application.
Internal-Error-NoSupportedArchitectures	An error occurred because the current architecture is not supported.
Internal-Error-PR	An error occurred during the processing of the pull request.
Internal-Error-Static-Scan	An error occurred during static analysis of the installers.
Internal-Error-URL	An error occurred during reputation validation of the installers.
Internal-Error	A generic failure or unknown error was encountered during the test pass.

Troubleshooting submissions to Windows Package Manager

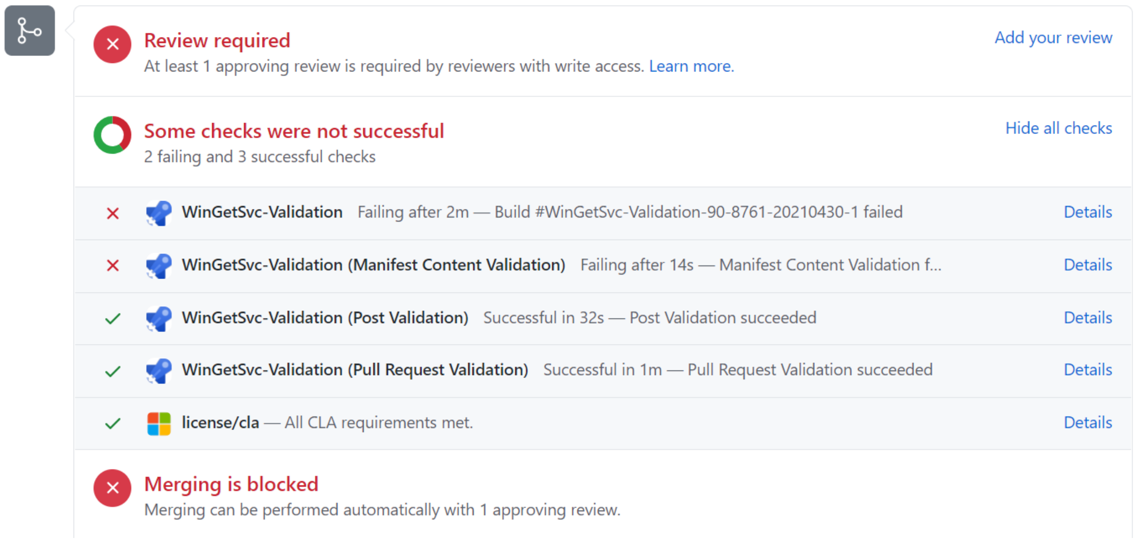
5/26/2021 • 2 minutes to read • Edit Online

When Windows Package Manager is processing the manifest files in the pipeline, it displays labels. If your pull request fails then then you may need to investigate to understand the failure better.

This article walks you through how you can get more information on your pull request failure.

Walkthrough of investigating a failure

1. When a pull request fails, it indicates the failure at the bottom of the web page. It will indicate a failure with the string **Some checks were not successful**. Click the **Details** link.



Review required
At least 1 approving review is required by reviewers with write access. Learn more.
Add your review

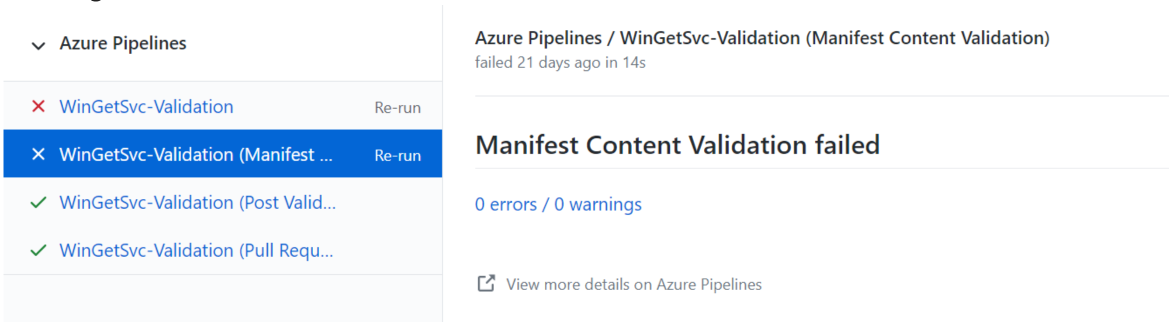
Some checks were not successful
2 failing and 3 successful checks
Hide all checks

✗	WinGetSvc-Validation Failing after 2m — Build #WinGetSvc-Validation-90-8761-20210430-1 failed	Details
✗	WinGetSvc-Validation (Manifest Content Validation) Failing after 14s — Manifest Content Validation f...	Details
✓	WinGetSvc-Validation (Post Validation) Successful in 32s — Post Validation succeeded	Details
✓	WinGetSvc-Validation (Pull Request Validation) Successful in 1m — Pull Request Validation succeeded	Details
✓	license/cla — All CLA requirements met.	Details

Merging is blocked
Merging can be performed automatically with 1 approving review.

.

2. After you click **Details**, you will go to an Azure Pipelines page. Click the link with the string **0 errors / 0 warnings**.



Azure Pipelines

✗	WinGetSvc-Validation	Re-run
✗	WinGetSvc-Validation (Manifest ...	Re-run
✓	WinGetSvc-Validation (Post Valid...	
✓	WinGetSvc-Validation (Pull Requ...	

Azure Pipelines / WinGetSvc-Validation (Manifest Content Validation)
failed 21 days ago in 14s

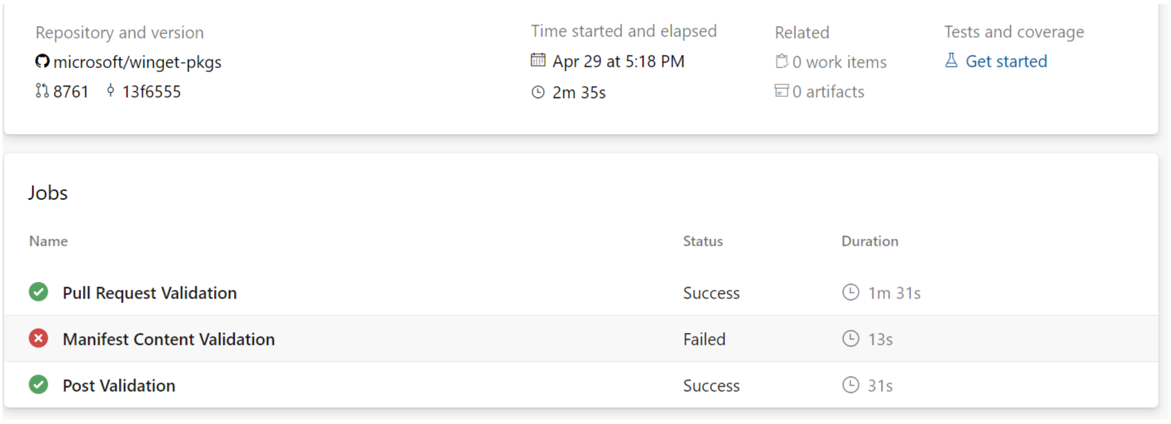
Manifest Content Validation failed

0 errors / 0 warnings

View more details on Azure Pipelines

.

3. The next page lists the job that failed. In the following screenshot, the failed job is **Manifest Content Validation**. Click the failed job.


Repository and version
microsoft/winget-pkgs
8761 13f6555

Time started and elapsed
Apr 29 at 5:18 PM
2m 35s

Related
0 work items
0 artifacts

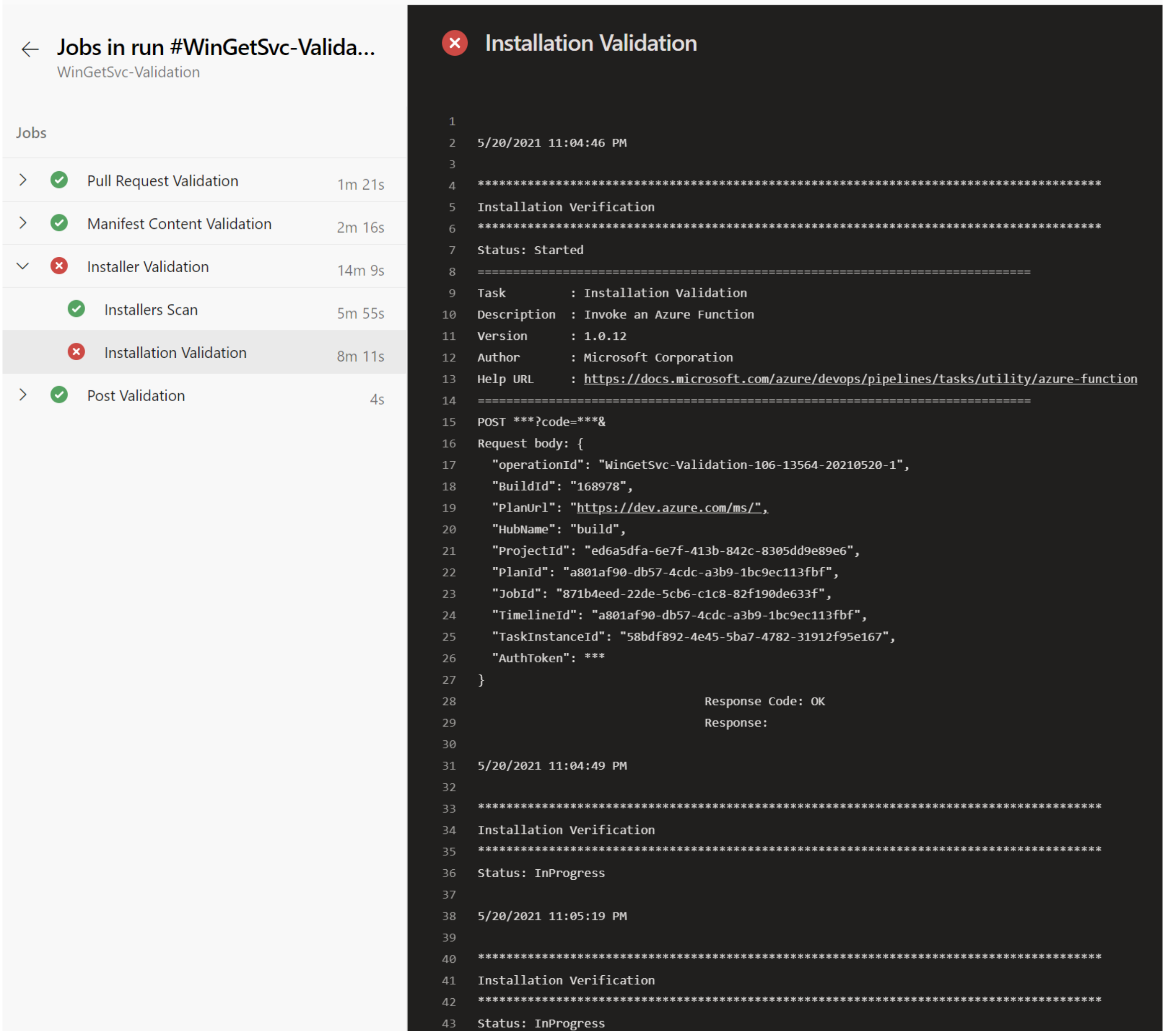
Tests and coverage
Get started

Jobs

Name	Status	Duration
✓ Pull Request Validation	Success	1m 31s
✗ Manifest Content Validation	Failed	13s
✓ Post Validation	Success	31s

.

4. The next page displays the output for the failed job. You can use this information to debug the issue. In the following example, the failure was during **Installation Validation** task. The output should help you identify the change that needs to be made to fix the manifest.


Jobs in run #WinGetSvc-Valida...
WinGetSvc-Validation

Jobs

✓ Pull Request Validation	1m 21s
✓ Manifest Content Validation	2m 16s
✗ Installer Validation	14m 9s
✓ Installers Scan	5m 55s
✗ Installation Validation	8m 11s
✓ Post Validation	4s

Installation Validation

```

5/20/2021 11:04:46 PM

****************************************************************************************
Installation Verification
****************************************************************************************
Status: Started
==============================================================================
Task         : Installation Validation
Description  : Invoke an Azure Function
Version      : 1.0.12
Author       : Microsoft Corporation
Help URL     : https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/azure-function
==============================================================================
POST ***?code=***&
Request body: {
  "operationId": "WinGetSvc-Validation-106-13564-20210520-1",
  "BuildId": "168978",
  "PlanUrl": "https://dev.azure.com/ms/",
  "HubName": "build",
  "ProjectId": "ed6a5dfa-6e7f-413b-842c-8305dd9e89e6",
  "PlanId": "a801af90-db57-4cdc-a3b9-1bc9ec113fbf",
  "JobId": "871b4eed-22de-5cb6-c1c8-82f190de633f",
  "TimelineId": "a801af90-db57-4cdc-a3b9-1bc9ec113fbf",
  "TaskInstanceId": "58bdf892-4e45-5ba7-4782-31912f95e167",
  "AuthToken": ***
}
                                Response Code: OK
                                Response:

5/20/2021 11:04:49 PM

****************************************************************************************
Installation Verification
****************************************************************************************
Status: InProgress

5/20/2021 11:05:19 PM

****************************************************************************************
Installation Verification
****************************************************************************************
Status: InProgress
```

.

Binary validation errors

5/26/2021 • 2 minutes to read • Edit Online

If your pull request fails to pass the `Installers Scan` test and is given the `Binary-Validation-Error` label, this indicates that your application failed to install on all environments. This article provides more background and guidance about this error.

Understanding the Installers Scan test

Windows Package Manager goes to great lengths to create an excellent user experience when installing applications. In order to do this, we must ensure that all applications install on PCs without errors regardless of environment.

To that end, a key test we use for the Windows Package Manager is to ensure that all installers will install without warnings on a variety of popular antivirus configurations. While Windows provides Microsoft Defender as a built-in antivirus program, many enterprise customers and users employ a wide range of antivirus software.

Therefore, each submission to the Windows Package Manager will be run through several antivirus programs. These programs all have different virus detection algorithms for identifying Potentially unwanted application (PUA) and malware.

If an application fails validation, Microsoft will first attempt to verify that the flagged software is not a false positive with the antivirus vendors. In many cases, after notification and validation, the antivirus vendor will update their algorithm and the application will pass.

What to do if you see the Binary-Validation-Error label

In some cases, the code anomaly detected is not able to be determined to be a false positive by the antivirus vendors. In this case the application cannot be added to the Windows Package Manager repository, and the pull request will be rejected with a `Binary-Validation-Error` label.

If the `Binary-Validation-Error` label is applied to your pull request, update your software to remove the code detected as PUA.

What if I cannot remove that code?

Occasionally, genuine tools used for debugging and low-level activities will appear as PUA to the antivirus vendors. This is because the code necessary to do the debugging will have a similar signature to unwanted software. Even though this is a legitimate use of that coding practice, unfortunately we are unable to allow those applications into the Windows Package Manager repository.

Windows Package Manager repository policies

7/6/2021 • 13 minutes to read • Edit Online

Document version: 1.0

Document date: May 22, 2021

Effective date: May 22, 2021

Thank you for your interest in providing a Product to the Windows Package Manager repository.

- **Product** refers to content in whatever form including, but not limited to, apps, games, titles, and any additional content sold or offered from within a **Product**.
- **Submission** refers to a pull request of manifest files and includes but is not limited to the Product and metadata about the Product.

A few principles to get you started:

- Offer unique and distinct value within your Submission. Provide a compelling reason to download the Product from the Windows Package Manager repository.
- Don't mislead our customers about what your Submission can do, who is offering it, etc.
- Don't attempt to cheat customers, the system or the ecosystem. There is no place in the repository for any kind of fraud, be it ratings and review manipulation, credit card fraud or other fraudulent activity.

Adhering to these policies should help you make choices that enhance your Submission's appeal and audience.

Your Submissions are crucial to the experience of hundreds of millions of customers. We can't wait to see what you create and want to help deliver your Submissions to the world.

If you have feedback on the policies or Windows Package Manager, let us know by commenting in our GitHub issues forum

Table of Contents

Product Policies

1.1 Distinct Function & Value; Accurate Representation

The Product and its associated metadata, including but not limited to the app title, description, screenshots, trailers, content rating and Product category, must accurately and clearly reflect the source, functionality, and features of the Product.

1.1.1

All aspects of the Product should accurately describe the functions, features and any important limitations of the Product.

1.1.2

Tags may not exceed 16 unique tags and should be relevant to the Product.

1.1.3

The Product must have distinct and informative metadata and must provide a valuable and quality user experience.

1.1.4

The InstallerUrl must be the ISV's release location for the Product. Products from download websites will not be allowed.

1.2 Security

The Product must not jeopardize or compromise user security, or the security or functionality of the device, system or related systems.

1.2.1

The Product must not attempt to change or extend its described functionality through any form of dynamic inclusion of code that is in violation of Windows Package Manager Policies. The Product should not, for example, download a remote script and subsequently execute that script in a manner that is not consistent with the described functionality.

1.2.2

The Product must not contain or enable malware as defined by the Microsoft criteria for Unwanted and Malicious Software.

1.2.3

The Product may contain fully integrated middleware (such as third-party cross-platform engines and third-party analytics services).

The Product may depend on non-integrated software (such as another Product, module, or service) to deliver its primary functionality.

1.3 Product is Testable

The Product must be testable. If it is not possible to test your submitted Product for any reason your Product may fail this requirement.

1.4 Usability

The Product should meet usability standards, including, but not limited to, those listed in the subsections below.

1.4.1

The Product should support the devices and platforms on which they are downloaded, including compatibility with the software, hardware and screen resolution requirements specified by the Product. If the Product is downloaded on a device with which it is not compatible, it should detect that at launch and display a message to the customer detailing the requirements.

1.4.2

The Product should continue to run and remain responsive to user input. Products should shut down gracefully and not close unexpectedly. The Product should handle exceptions raised by any of the managed or native system APIs and remain responsive to user input after the exception is handled.

1.4.3

The Product should start up promptly and must stay responsive to user input.

1.5 Personal Information

The following requirements apply to Products that access Personal Information. Personal Information includes all information or data that identifies or could be used to identify a person, or that is associated with such information or data.

1.5.1

If the Product accesses, collects or transmits Personal Information, or if otherwise required by law, it should maintain a privacy policy. The submission, should include the PrivacyUrl which links to the privacy policy of the Product.

1.5.2

If the Product publishes the Personal Information of customers of the Product to an outside service or third party, the Product should only do so after obtaining opt-in consent from those customers. Opt-in consent means the customer gives their express permission in the Product user interface for the requested activity, after the Product has:

- Described to the customer how the information will be accessed, used or shared, indicating the types of parties to whom it is disclosed, and
- Provided the customer a mechanism in the Product user interface through which they can later rescind this permission and opt-out.

1.5.3

If the Product publishes a person's Personal Information to an outside service or third party through the Product or its metadata, but the person whose information is being shared is not a customer of the Product, the Product must obtain express written consent to publish that Personal Information, and must permit the person whose information is shared to withdraw that consent at any time. If the Product provides a customer with access to another person's Personal Information, this requirement would also apply.

1.5.4

If the Product collects, stores or transmits Personal Information, it must do so securely, by using modern cryptography methods.

1.5.5

The Product must not collect, store or transmit highly sensitive personal information, such as health or financial data, unless the information is related to the Product's functionality. The Product must also obtain express user consent before collecting, storing or transmitting such information. The Product's privacy policy must clearly tell the user when and why it is collecting Personal Information and how it will be used.

1.5.6

If the Product supports Microsoft identity authentication it must do so only by using Microsoft-approved methods.

1.5.7

Products that receive device location must provide settings that allow the user to enable and disable the Product's access to and use of location from the Location Service API.

1.6 Capabilities

If the Product declares the use of capabilities, then the capabilities the Product declares must legitimately relate to the functions of the Product. The Product must not circumvent operating system checks for capability usage.

1.7 Localization

If the Product you should provide localized all languages that it supports. The experience provided by a product must be reasonably similar in all languages that it supports.

1.8 Financial Transactions

If your product includes in-product purchase, subscriptions, virtual currency, billing functionality or captures financial information, the following requirements apply:

1.8.1

In-product offerings sold in your product cannot be converted to any legally valid currency (for example, USD, Euro, etc.) or any physical goods or services.

1.8.2

The Product must use a secure purchase API for purchases of physical goods or services, and a secure purchase API for payments made in connection with real world gambling or charitable contributions. If the Product is used to facilitate or collect charitable contributions or to conduct a promotional sweepstakes or contest, it must do so in compliance with applicable law. The Product must also state clearly that Microsoft is not the fundraiser or sponsor of the promotion.

The Product must use a secure purchase API to receive voluntary donations from users.

The following requirements apply to your use of a secure purchase API:

- At the time of the transaction or when the Product collects any payment or financial information from the customer, the Product must identify the commerce transaction provider, authenticate the user, and obtain user confirmation for the transaction.
- The product can offer the user the ability to save this authentication, but the user must have the ability to either require an authentication on every transaction or to turn off in-product transactions.
- If the product collects credit card information or uses a third-party payment processor that collects credit card information, the payment processing must meet the current PCI Data Security Standard (PCI DSS).

1.8.3

The product and its associated metadata must provide information about the types of in-product purchases offered and the range of prices. The Product not mislead customers and must be clear about the nature of the in-product promotions and offerings including the scope and terms of any trial experiences. If the Product restricts access to user-created content during or after a trial, it must notify users in advance. In addition, the Product must make it clear to users that they are initiating a purchase option in the Product.

If your game offers "loot boxes" or other mechanisms that provide randomized virtual items, then you must disclose the odds of receiving each item to customers prior to purchase. These disclosures may appear: in-product, such as in an in-app store, on the Microsoft Store Product Description Page (PDP), and/or on a developer or publisher website, with a link from the Store Product Description Page (PDP) and/or in-app.

10.8.4

All pricing, including sales or discounting, for your digital products or services shall comply with all applicable laws, regulations and regulatory guidelines, including without limitation, the Federal Trade Commission Guides Against Deceptive Pricing.

1.9 Notifications

If the Product supports notifications, then the Product must respect system settings for notifications and remain functional when they are disabled. This includes the presentation of ads and notifications to the customer, which must also be consistent with the customer's preferences, whether the notifications are provided by the Microsoft Push Notification Service (MPNS), Windows Push Notification Service (WNS) or any other service. If the customer disables notifications, either on an Product-specific or system-wide basis, the Product must remain functional.

1.10 Advertising Conduct and Content

For all advertising related activities, the following requirements apply:

1.10.1

- The primary purpose of the Product should not be to get users to click ads.
- The Product may not do anything that interferes with or diminishes the visibility, value, or quality of any ads it displays.
- The Product must respect advertising ID settings that the user has selected.
- All advertising must be truthful, non-misleading and comply with all applicable laws, regulations, and regulatory guidelines.

Content Policies

The following policies apply to content and metadata (including publisher name, Product name, Product icon, Product description, Product screenshots, Product trailers and trailer thumbnails, and any other Product metadata) offered for distribution in the Windows Package Manager repository. Content means the Product name, publisher name, Product icon, Product description, the images, sounds, videos and text contained in the Product, the tiles, notifications, error messages or ads exposed through the Product, and anything that's delivered from a server or that the Product connects to. Because Product and the Windows Package Manager repository are used around the world, these requirements will be interpreted and applied in the context of regional and cultural norms.

2.1 General Content Requirements

Metadata and other content you submit to accompany your submission may contain only content that would merit a rating of PEGI 12, ESRB EVERYONE 10+, or lower.

2.2 Content Including Names, Logos, Original and Third Party

All content in the Product and associated metadata must be either originally created by the application provider, appropriately licensed from the third-party rights holder, used as permitted by the rights holder, or used as otherwise permitted by law.

2.3 Risk of Harm

2.3.1

The Product must not contain any content that facilitates or glamorizes the following real world activities: (a) extreme or gratuitous violence; (b) human rights violations; (c) the creation of illegal weapons; or (d) the use of weapons against a person, animal, or real or personal property.

2.3.2

The Product must not: (a) pose a safety risk to, nor result in discomfort, injury or any other harm to end users or to any other person or animal; or (b) pose a risk of or result in damage to real or personal property. You are solely responsible for all Product safety testing, certificate acquisition, and implementation of any appropriate feature safeguards. You will not disable any platform safety or comfort features, and you must include all legally required and industry-standard warnings, notices, and disclaimers in the Product.

2.4 Defamatory, Libelous, Slanderous and Threatening

The Product must not contain any content that is defamatory, libelous, slanderous, or threatening.

2.5 Offensive Content

The Product and associated metadata must not contain potentially sensitive or offensive content. Content may be considered sensitive or offensive in certain countries/regions because of local laws or cultural norms. In addition, the Product and associated metadata must not contain content that advocates discrimination, hatred, or violence based on considerations of race, ethnicity, national origin, language, gender, age, disability, religion, sexual orientation, status as a veteran, or membership in any other social group.

2.6 Alcohol, Tobacco, Weapons and Drugs

The Product must not contain any content that facilitates or glamorizes excessive or irresponsible use of alcohol or tobacco Products, drugs, or weapons.

2.7 Adult Content

The Product must not contain or display content that a reasonable person would consider pornographic or sexually explicit.

2.8 Illegal Activity

The Product must not contain content or functionality that encourages, facilitates or glamorizes illegal activity in the real world.

2.9 Excessive Profanity and Inappropriate Content

- The Product must not contain excessive or gratuitous profanity.
- The Product must not contain or display content that a reasonable person would consider to be obscene.

2.10 Country/Region Specific Requirements

Content that is offensive in any country/region to which the Product is targeted is not allowed. Content may be considered offensive in certain countries/regions because of local laws or cultural norms. Examples of potentially offensive content in certain countries/regions include the following:

China

- Prohibited sexual content
- Disputed territory or region references
- Providing or enabling access to content or services that are illegal under applicable local law

2.11 Age Ratings

The Product should have a age rating that would merit a rating of PEGI 12, ESRB EVERYONE 10+, or lower.

2.11.1

If the Product provides content (such as user-generated, retail or other web-based content) that might be appropriate for a higher age rating than its assigned rating, you must enable users to opt in to receiving such content by using a content filter or by signing in with a pre-existing account.

2.12 User Generated Content

User Generated Content (UGC) is content that users contribute to an app or Product and which can be viewed or accessed by other users in an online state. If the Product contains UGC, the Product should:

- Publish and make available to users a Product terms of service and/or content guidelines for User Generated Content either in Product or on the Product website.
- Provide a means for users to report inappropriate content within the Product to the developer for review and removal/disablement if in violation of content guidelines and/or implement a method for proactive detection of inappropriate or harmful UGC.
- Remove or disable UGC when requested by Microsoft.

See also

- Change history for Windows Package Manager policies
- Windows Package Manager Code of Conduct
- Windows Package Manager Contributing requirements

Change history for Windows Package Manager policies

5/26/2021 • 2 minutes to read • Edit Online

DATE	DOCUMENT VERSION	CHANGE DESCRIPTION
5/25/2021	1.0	Initial publishing of Windows Package Manager policies.

Microsoft PowerToys: Utilities to customize Windows 10

6/30/2021 • 3 minutes to read • Edit Online

Microsoft PowerToys is a set of utilities for power users to tune and streamline their Windows 10 experience for greater productivity.

Install PowerToys

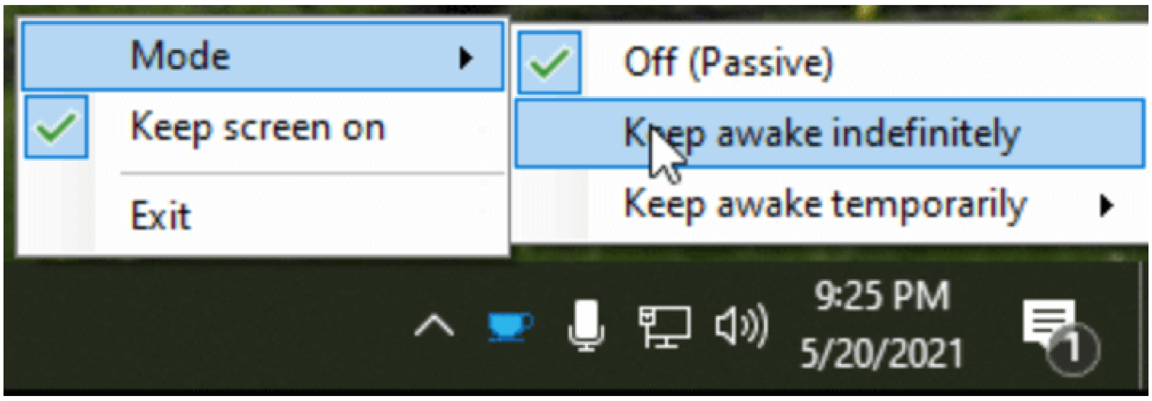
Processor support

- `x64`: Supported
- `x86`: In development (see issue #602)
- `ARM`: In development (see issue #490)

Current PowerToy utilities

The currently available utilities include:

Awake

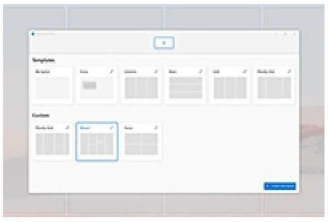


Awake is designed to keep a computer awake without having to manage its power & sleep settings. This behavior can be helpful when running time-consuming tasks, ensuring that the computer does not go to sleep or turn off its screens.

Color Picker

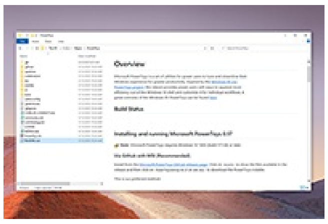
ColorPicker is a system-wide color picking utility activated with `Win+Shift+C`. Pick colors from any currently running application, the picker automatically copies the color into your clipboard in a configurable format. Color Picker also contains an editor that shows a history of previously picked colors, allows you to fine-tune the selected color and to copy different string representations. This code is based on Martin Chrzan's Color Picker.

Fancy Zones



FancyZones is a window manager that makes it easy to create complex window layouts and quickly position windows into those layouts.

File Explorer add-ons



File Explorer add-ons enable preview pane rendering in File Explorer to display SVG icons (.svg) and Markdown (.md) file previews. To enable the preview pane, select the "View" tab in File Explorer, then select "Preview Pane".

Image Resizer

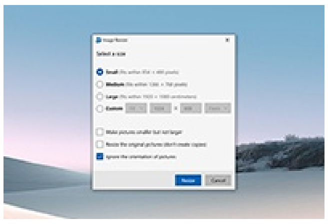
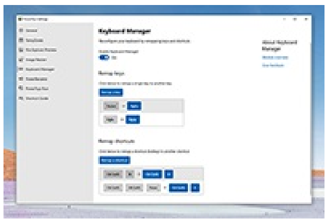


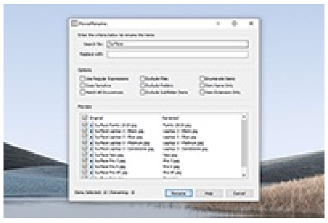
Image Resizer is a Windows Shell Extension for quickly resizing images. With a simple right click from File Explorer, resize one or many images instantly. This code is based on Brice Lambson's Image Resizer.

Keyboard Manager



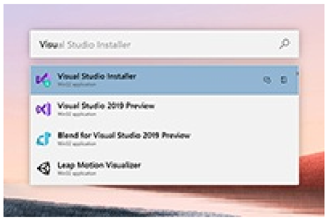
Keyboard Manager allows you to customize the keyboard to be more productive by remapping keys and creating your own keyboard shortcuts. This PowerToy requires Windows 10 1903 (build 18362) or later.

PowerRename



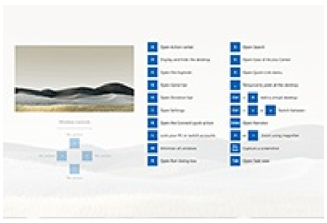
PowerRename enables you to perform bulk renaming, searching and replacing file names. It includes advanced features, such as using regular expressions, targeting specific file types, previewing expected results, and the ability to undo changes. This code is based on Chris Davis's SmartRename.

PowerToys Run



PowerToys Run can help you search and launch your app instantly - just enter the shortcut `Alt+Space` and start typing. It is open source and modular for additional plugins. Window Walker is now included as well. This PowerToy requires Windows 10 1903 (build 18362) or later.

Shortcut Guide



Windows key shortcut guide appears when a user holds the Windows key down for more than one second and shows the available shortcuts for the current state of the desktop.

PowerToys video walk-through

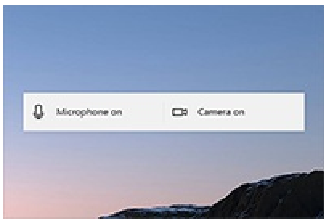
In this video, Clint Rutkas (PM for PowerToys) walks through how to install and use the various utilities available, in addition to sharing some tips, info on how to contribute, and more.

Future PowerToy utilities

Experimental PowerToys

Install the pre-release experimental version of PowerToys to try the latest experimental utilities, including:

Video Conference Mute (Experimental)



Video Conference Mute is a quick way to globally "mute" both your microphone and camera using ⊞ `Win+N` while on a conference call, regardless of the application that currently has focus. This is only included in the pre-release/experimental version of PowerToys and requires Windows 10 1903 (build 18362) or later.

Known issues

Search known issues or file a new issue in the Issues tab of the PowerToys repository on GitHub.

Contribute to PowerToys (Open Source)

PowerToys welcomes your contributions! The PowerToys development team is excited to partner with the power user community to build tools that help users get the most out of Windows. There are a variety of ways to contribute:

- Write a tech spec
- Submit a design concept or recommendation
- Contribute to documentation
- Identify and fix bugs in the source code
- Code new features and PowerToy utilities

Before starting work on a feature that you would like to contribute, **read the Contributor's Guide**. The PowerToys team will be happy to work with you to figure out the best approach, provide guidance and mentorship throughout feature development, and help avoid any wasted or duplicate effort.

PowerToys release notes

PowerToys release notes are listed on the install page of the GitHub repo. For reference, you can also find the Release checklist on the PowerToys wiki.

PowerToys history

Inspired by the Windows 95 era PowerToys project, this reboot provides power users with ways to squeeze more efficiency out of the Windows 10 shell and customize it for individual workflows. A great overview of the Windows 95 PowerToys can be found here.

PowerToys roadmap

PowerToys is a rapid-incubation, open source team aimed at providing power users with ways to squeeze more efficiency out of the Windows 10 shell and customize it for individual workflows. Work priorities will consistently be examined, reassessed, and adjusted with the aim of improving our users productivity.

- New specs for possible PowerToys
- Backlog priority list
- Version 1.0 Strategy spec, February 2020

Install PowerToys

6/30/2021 • 2 minutes to read • Edit Online

WARNING

PowerToys v0.37 and beyond will require Windows 10 v1903 or greater. The v1 settings, which supports older Windows versions, will be removed in v0.37.

We recommend installing PowerToys using the Windows executable button linked below, but alternative install methods are also listed if you prefer using a package manager.

Install with Windows executable file

Install PowerToys

To install PowerToys using a Windows executable file:

1. Visit the Microsoft PowerToys GitHub releases page.
2. Browse the list of stable and experimental versions of PowerToys that are available.
3. Select the **Assets** drop-down menu to display the files for the release.
4. Select the `PowerToysSetup-0.##.#-x64.exe` file to download the PowerToys executable installer.
5. Once downloaded, open the executable file and follow the installation prompts.

Requirements

- Windows 10 1803 (build 17134) or later.
- .NET Core 3.1 Desktop Runtime. The PowerToys installer will handle this requirement.
- x64 architecture currently supported. ARM and x86 support to become available at a later date.

To ensure that your machine meets these requirements, check your Windows 10 version and build number by selecting the ⊞ **Win** *(Windows key)* + **R**, then type **winver**, select **OK**. (Or enter the `ver` command in Windows Command Prompt). You can update to the latest Windows version in the **Settings** menu.

Alternative Install Methods

- Windows Package Manager *(Preview)*
- Community-driven install tools *(Not officially supported)*

Install with Windows Package Manager (Preview)

To install PowerToys using the Windows Package Manager (WinGet) preview:

1. Download PowerToys from Windows Package Manager.
2. Run the following command from the command line / PowerShell:

```
WinGet install powertoys
```

Community-driven install tools

These community-driven alternative install methods are not officially supported and the PowerToys team does not update or manage these packages.

Install with Chocolatey

To install PowerToys using Chocolatey, run the following command from your command line / PowerShell:

```
choco install powertoys
```

To upgrade PowerToys, run:

```
choco upgrade powertoys
```

If you have issues when installing/upgrading, visit the PowerToys package on Chocolatey.org and follow the Chocolatey triage process.

Install with Scoop

To install PowerToys using Scoop, run the following command from the command line / PowerShell:

```
scoop bucket add extras
scoop install powertoys
```

To update PowerToys, run the following command from the command line / PowerShell:

```
scoop update powertoys
```

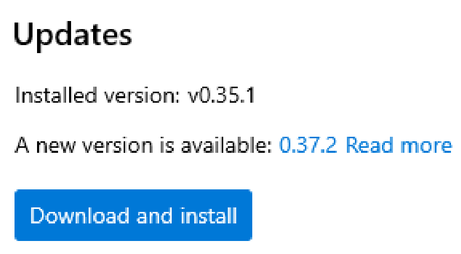
If you have issues when installing/updating, file an issue in the Scoop repo on GitHub.

Post Install

After successfully installing PowerToys, an overview window will display with introductory guidance on each of the available utilities.

Updates

PowerToys uses an auto-updater that checks for new versions when the app is running. If enabled, a toast notification will appear when an update is available. Updates can also be checked for manually from the PowerToys Settings menu under the General tab.


Updates

Installed version: v0.35.1

A new version is available: 0.37.2 Read more

PowerToys running with administrator elevated permissions

6/1/2021 • 2 minutes to read • Edit Online

If you're running any application as an administrator (also referred to as elevated permissions), PowerToys may not work correctly when the elevated applications are in focus or trying to interact with a PowerToys feature like FancyZones. This can be addressed by also running PowerToys as an administrator.

Options

There are two options for PowerToys to support applications running as administrator (with elevated permissions):

- **[Recommended]**: PowerToys will display a prompt when an elevated process is detected. Open **PowerToys Settings**. Inside the **General** tab, select "Restart as administrator".
- Enable "Always run as administrator" in the **PowerToys Settings**.

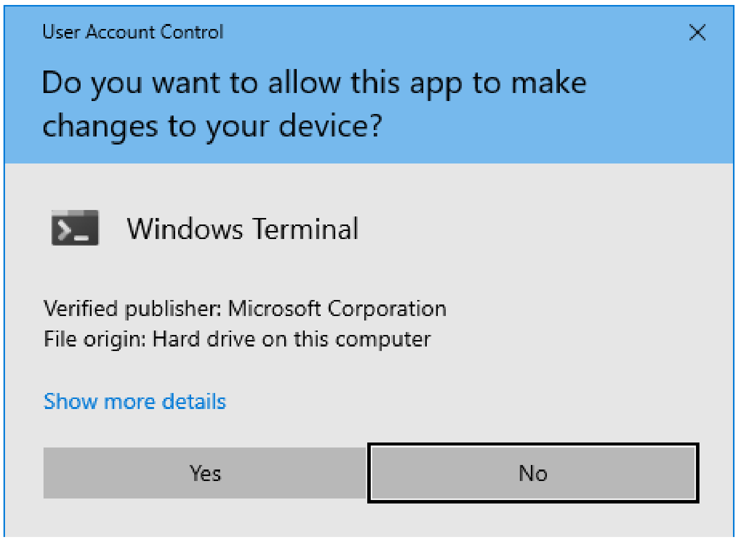
Run as administrator elevated processes explained

Windows applications run in **User mode** by default. To run an application in **Administrative mode** or as an *elevated process* means that app will run with additional access to the operating system.

The simplest way to run an app or program in administrative mode is to right-click the program and select **Run as administrator**. If the current user is not an administrator, Windows will ask for the administrator username and password.

Most apps do not need to run with elevated permission. A common scenario, however, for requiring administrator permission would be to run certain PowerShell commands or edit the registry.

If you see this prompt (User Access Control prompt), the application is requesting administrator level elevated permission:



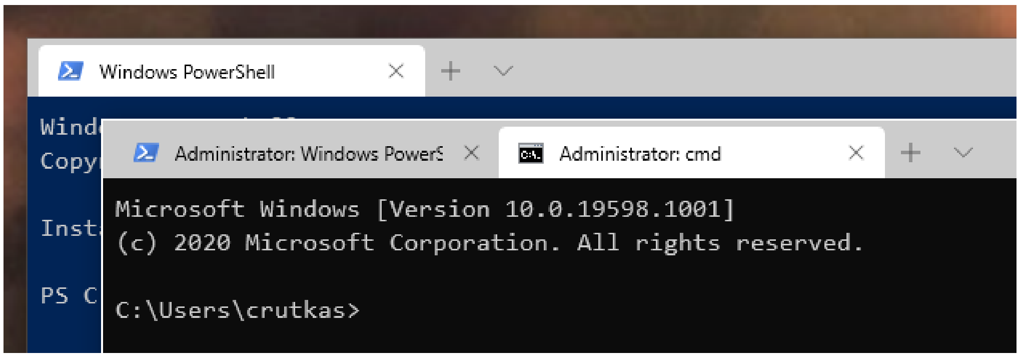
User Account Control

Do you want to allow this app to make changes to your device?

Windows Terminal

Verified publisher: Microsoft Corporation
File origin: Hard drive on this computer

In the case of an elevated command line, typically the title "Administrator" will be appended to the title bar.



```
Microsoft Windows [Version 10.0.19598.1001]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\crutkas>
```

Support for admin mode with PowerToys

PowerToys only needs elevated administrator permission when interacting with other applications that are running in administrator mode. If those applications are in focus, PowerToys may not function unless it is elevated as well.

These are the two scenarios we will not work in:

- Intercepting certain types of keyboard strokes
- Resizing / Moving windows

Affected PowerToys utilities

Admin mode permissions may be required in the following scenarios:

- FancyZones
  - Snapping an elevated window (e.g. command prompt) into a Fancy Zone
  - Moving the elevated window to a different zone
- Shortcut guide
  - Display shortcut
- Keyboard remapper
  - Key to key remapping
  - Global level shortcuts remapping
  - App-targeted shortcuts remapping
- PowerToys Run
  - Display shortcut

Awake utility

6/30/2021 • 2 minutes to read • Edit Online

Awake is a utility tool for Windows designed to keep a computer awake without having to manage its power & sleep settings. This behavior can be helpful when running time-consuming tasks, ensuring that the computer does not go to sleep or turn off its screens.

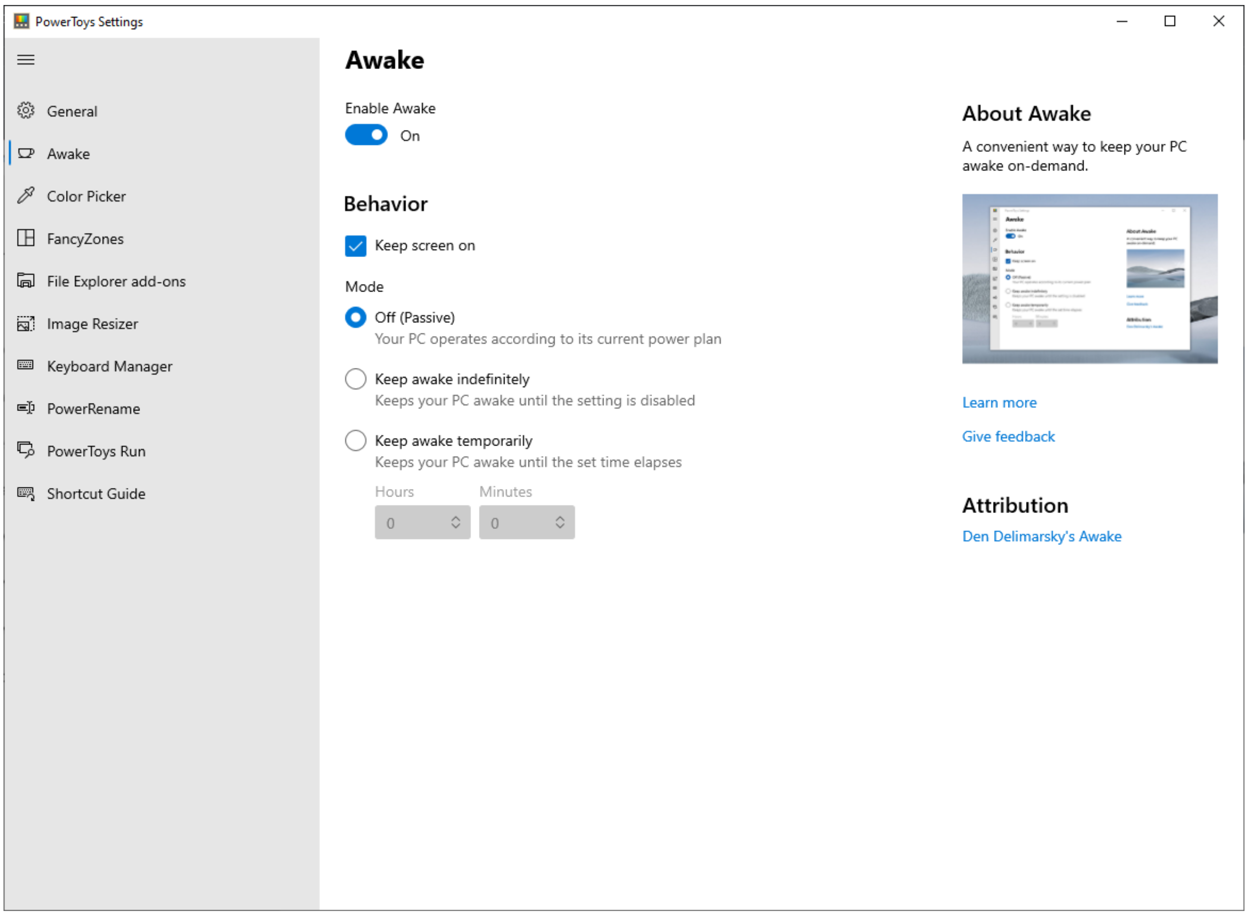
Get started

Awake can be used directly from PowerToys settings or as a standalone executable. When the tool is running from PowerToys, it can be managed from PowerToys settings or the system tray.

> **NOTE**
>
> Awake does not modify any of the Windows power plan settings, and does not depend on a custom power plan configuration. Instead, it spawns background threads that tell Windows that they require a specific state of the machine.

PowerToys settings

In the PowerToys settings view, start Awake by using the **Enable Awake** toggle. Once enabled, the application will manage the awakeness state of the computer.



Awake

Enable Awake: On

Behavior

[✓] Keep screen on

Mode

[✓ Off (Passive)] Your PC operates according to its current power plan

Keep awake indefinitely — Keeps your PC awake until the setting is disabled

Keep awake temporarily — Keeps your PC awake until the set time elapses

Hours: 0 Minutes: 0

About Awake

A convenient way to keep your PC awake on-demand.

Attribution

Den Delimarsky's Awake

The following Awake states can be selected:

- **Off (Passive)** - The computer awakeness state is unaffected. The application is waiting for user input.
- **Keep awake indefinitely** - The computer stays awake indefinitely, until the user explicitly puts the machine

to sleep or exits/disables the application.

- **Keep awake temporarily** - Keep machine awake for a pre-defined limited time. Once the time elapses, computer resumes its previous awakeness state.

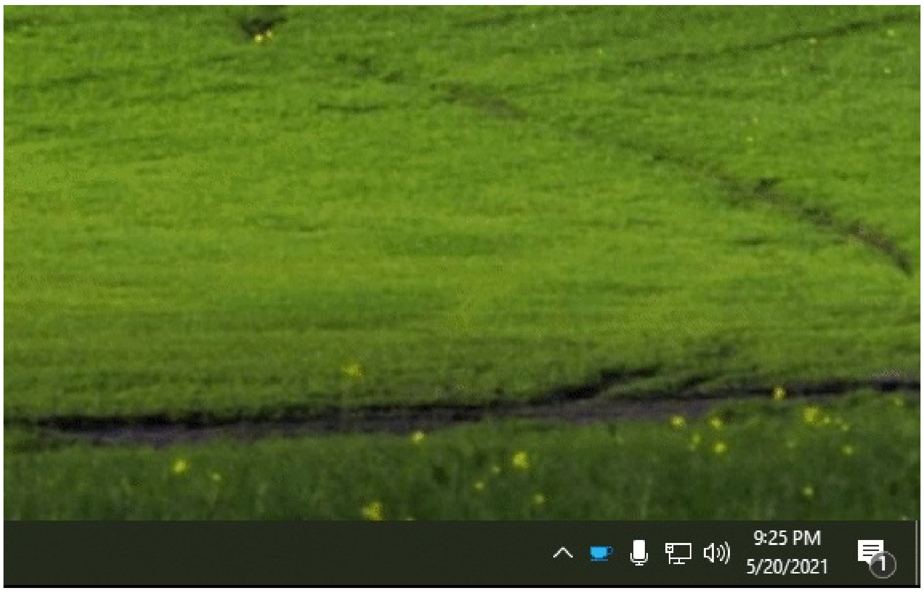
> **NOTE**
>
> Changing the hours or minutes while the computer is kept awake temporarily will reset the timer.

Keep screen on

While Awake can keep the computer awake indefinitely or temporarily, in its default state the displays connected to the machine will turn off, even though the computer won't go to sleep. If you need the displays to be available, use the **Keep screen on** switch, which will ensure that all monitors remain on.

System tray

To manage the execution of the tool from the system tray, right-click on the Awake icon.



Command Line Interface (CLI)

Awake can also be executed as a standalone application, directly from the PowerToys folder. The following command line arguments can be used when running `PowerToys.Awake.exe` from the terminal:

ARGUMENT	DESCRIPTION
`--use-pt-config`	Use the PowerToys configuration file to manage the settings. This assumes that there is a `settings.json` file for Awake, generated by PowerToys, that contains all required runtime information. This includes the Behavior Mode (indefinite or timed), whether screens should be kept on, and what the values for hours and minutes are for a temporary keep-awake. When this argument is used, all other arguments are ignored. Awake will look for changes in the `settings.json` file to update its state.

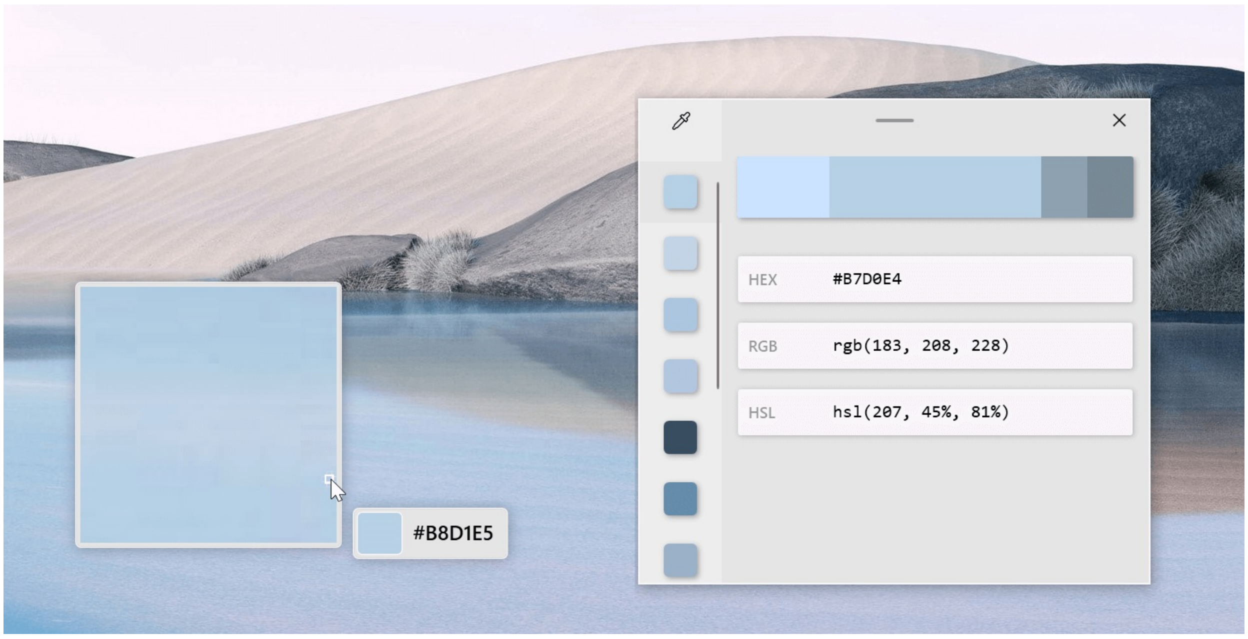
ARGUMENT	DESCRIPTION
`--display-on`	Determines whether the screens should be kept on or off while the machine is kept awake. Expected values are `true` or `false`.
`--time-limit`	Duration, in seconds, during which Awake keeps the computer awake. Can be used in combination with `--display-on`.
`--pid`	Attaches the execution of Awake to a Process ID (PID). When the process with a given PID terminates, Awake terminates as well.

In absence of command-line arguments, Awake will keep the computer awake indefinitely.

Color Picker utility

6/1/2021 • 2 minutes to read • Edit Online

A system-wide color picking utility for Windows 10 that enables you to pick colors from any currently running application and automatically copies it in a configurable format to your clipboard.



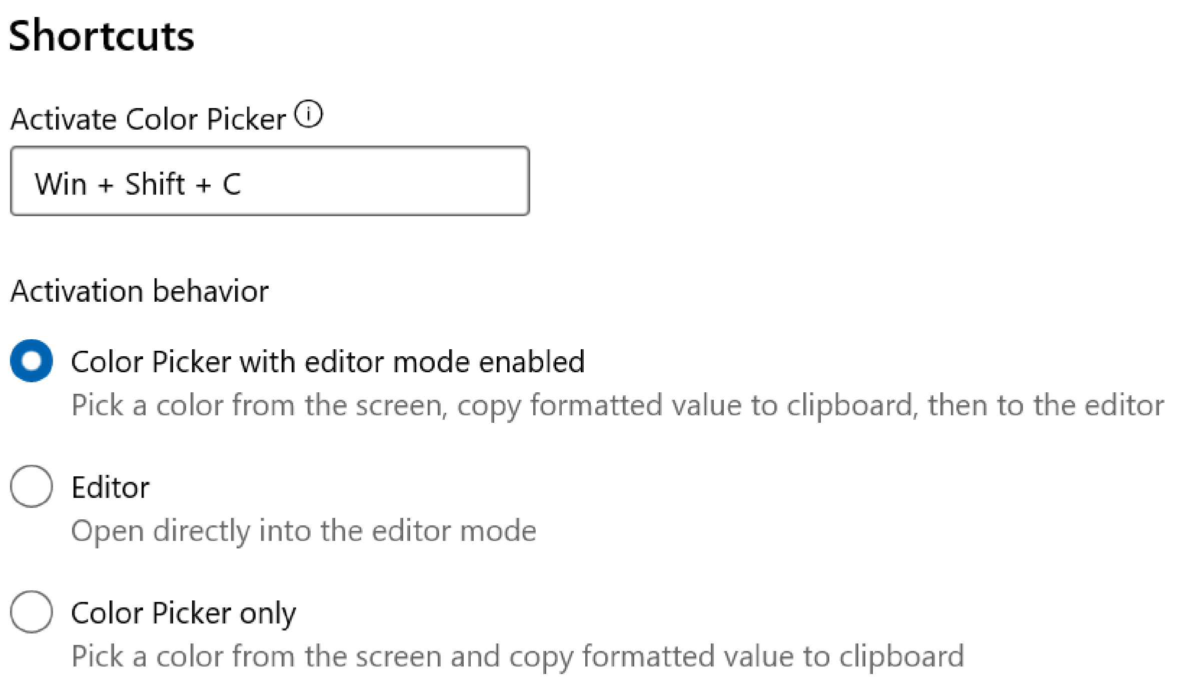
Getting started

Enable

To start using Color Picker, you need to first make sure it is enabled in the PowerToys settings (Color Picker section).

Activate

Once enabled, you can choose one of the following three behaviors to be executed when launching Color Picker with the activation shortcut `Win+Shift+C` (note that this shortcut can be changed in the settings dialog):



Shortcuts

Activate Color Picker ⓘ

Win + Shift + C

Activation behavior

[✓ Color Picker with editor mode enabled]
Pick a color from the screen, copy formatted value to clipboard, then to the editor

○ Editor
Open directly into the editor mode

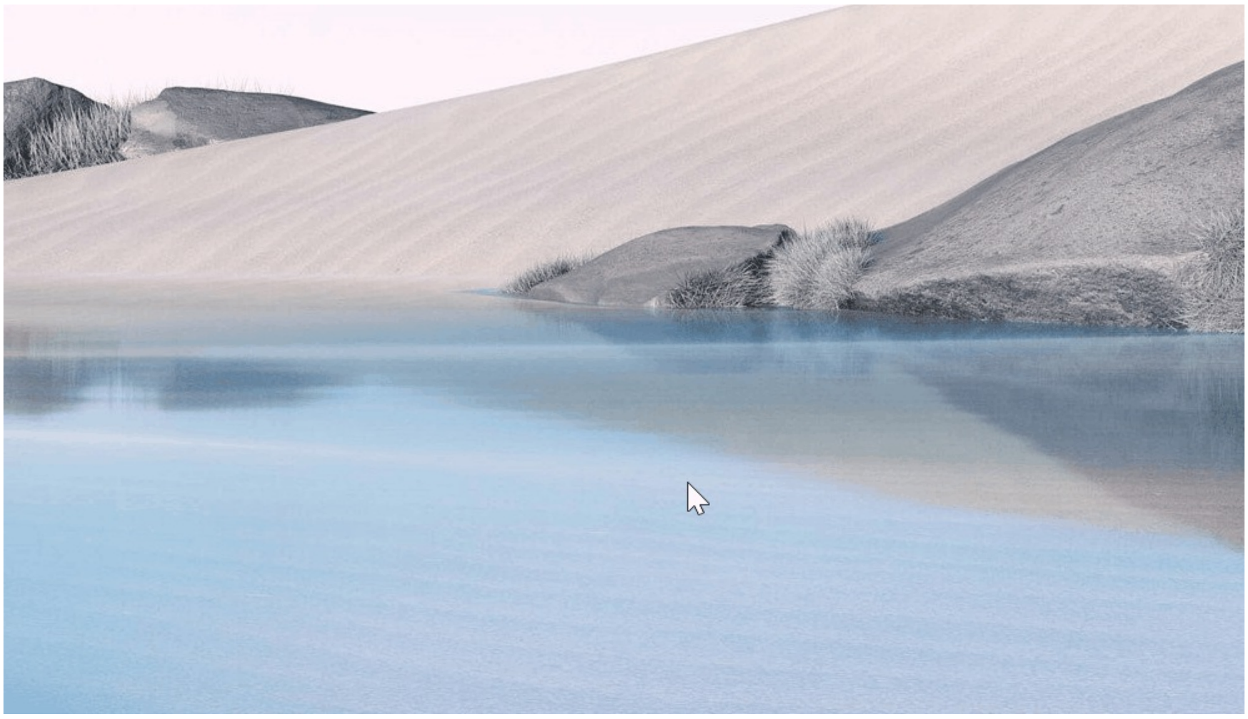
○ Color Picker only
Pick a color from the screen and copy formatted value to clipboard

- **Color Picker with editor mode enabled** - Opens Color Picker, after selecting a color, the editor is opened and the selected color is copied into the clipboard (in the default format - configurable in the settings dialog).
- **Editor** - Opens Editor directly, from here you can choose a color from the history, fine tune a selected color, or capture a new color with by opening the color picker.
- **Color Picker only** - Opens Color Picker only and the selected color will be copied into the clipboard.

Select color

After the Color Picker is activated, hover your mouse cursor over the color you would like to copy and left-click the mouse button to select a color. If you want to see the area around your cursor in more detail, scroll up to zoom in.

The copied color will be stored in your clipboard in the format that is configured in the settings (HEX by default).

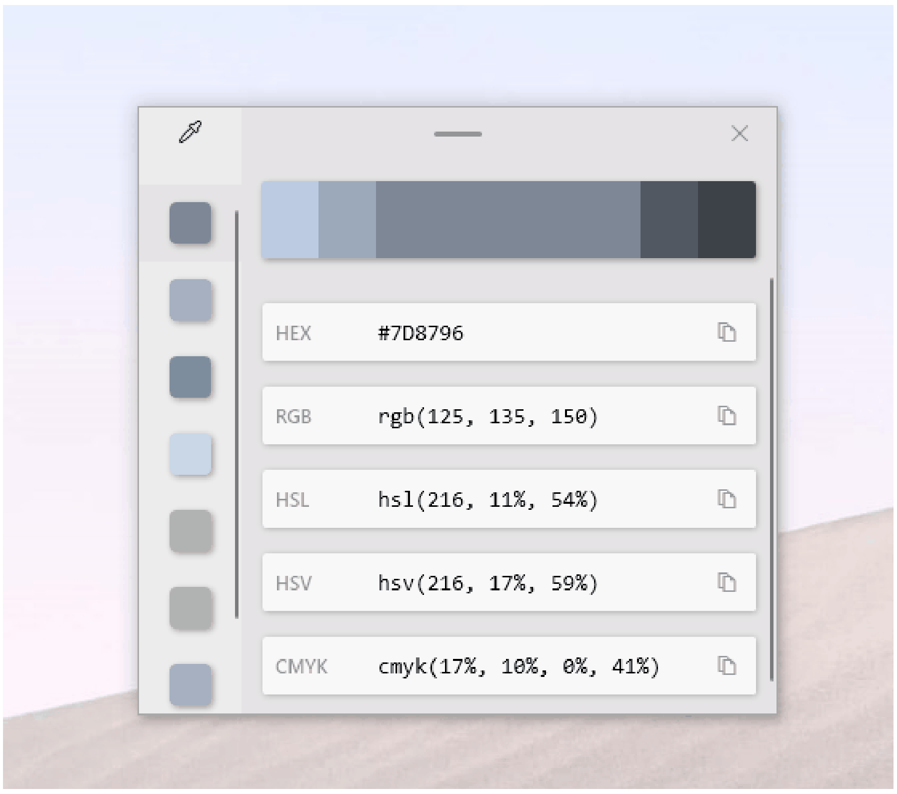


Editor usage

The editor lets you see the history of picked colors (up to 20) and copy their representation in any predefined string format. You can configure what color formats are visible in the editor, along with the order that they appear. This configuration can be found in PowerToys settings.

The editor also allows you to fine tune any picked color or get a new similar color. Editor previews different shades of currently selected color - 2 lighter and 2 darker ones.

Clicking on any of those alternative color shades will add the selection to the history of picked colors (appears on the top of the colors history list). Color in the middle represents your currently selected color from the colors history. By clicking on it, the fine tuning configuration control will appear, that will let you change HUE or RGB values of the current color. Pressing OK will add newly configured color into the colors history.



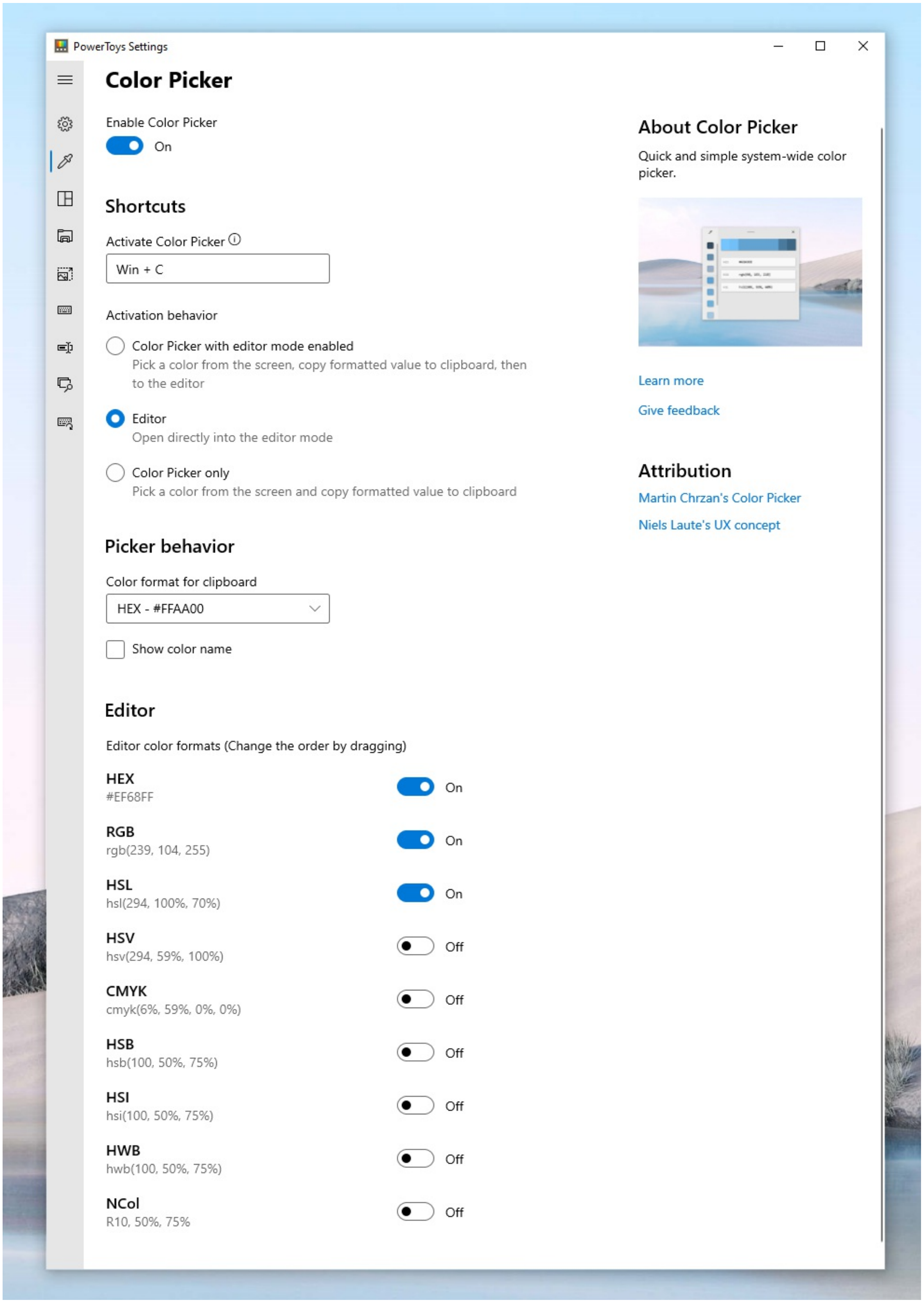
HEX	#7D8796
RGB	rgb(125, 135, 150)
HSL	hsl(216, 11%, 54%)
HSV	hsv(216, 17%, 59%)
CMYK	cmyk(17%, 10%, 0%, 41%)

To remove any color from the colors history, right click a desired color and select *Remove*.

Settings

Color picker will let you change following settings:

- Activation shortcut
- Behavior of activation shortcut
- Format of a copied color (HEX, RGB, etc.)
- Order and appearance of color formats in the editor



PowerToys Settings

Color Picker

Enable Color Picker

On

Shortcuts

Activate Color Picker

Win + C

Activation behavior

Color Picker with editor mode enabled
Pick a color from the screen, copy formatted value to clipboard, then to the editor

[✓ Editor]
Open directly into the editor mode

Color Picker only
Pick a color from the screen and copy formatted value to clipboard

Picker behavior

Color format for clipboard

HEX - #FFAA00

Show color name

Editor

Editor color formats (Change the order by dragging)

HEX	#EF68FF	On
RGB	rgb(239, 104, 255)	On
HSL	hsl(294, 100%, 70%)	On
HSV	hsv(294, 59%, 100%)	Off
CMYK	cmyk(6%, 59%, 0%, 0%)	Off
HSB	hsb(100, 50%, 75%)	Off
HSI	hsi(100, 50%, 75%)	Off
HWB	hwb(100, 50%, 75%)	Off
NCol	R10, 50%, 75%	Off

About Color Picker

Quick and simple system-wide color picker.

Learn more

Give feedback

Attribution

Martin Chrzan's Color Picker

Niels Laute's UX concept

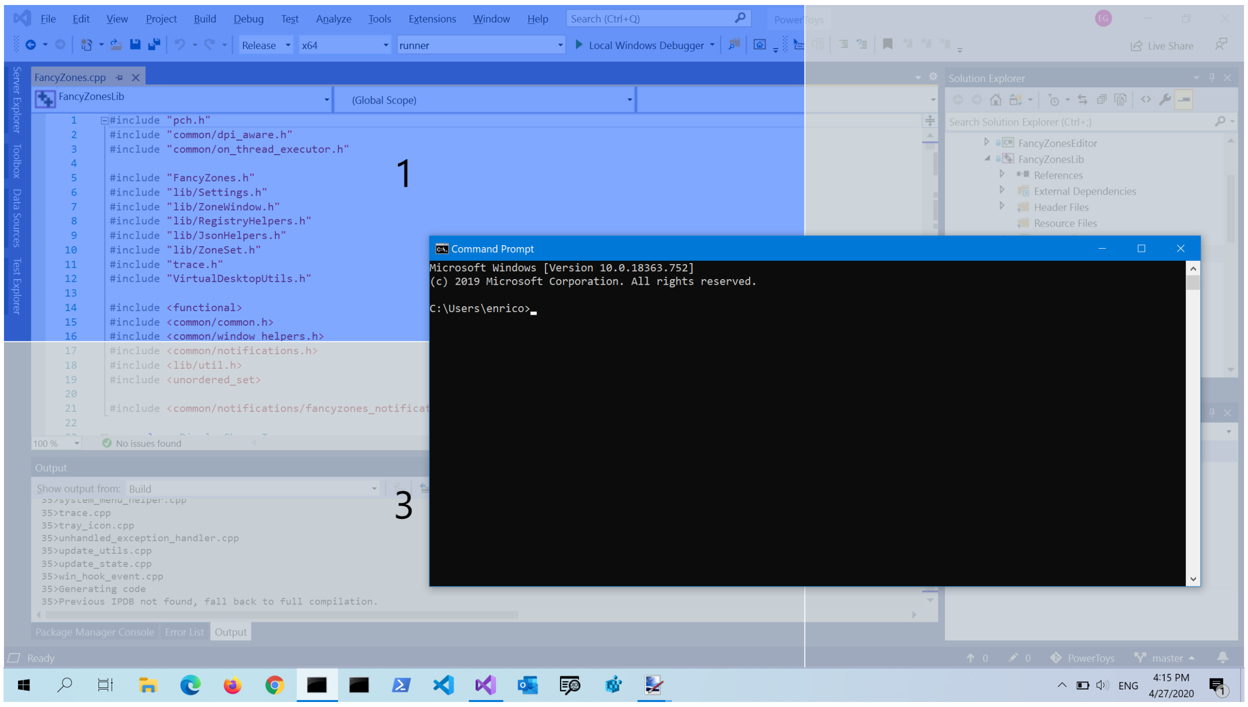
Limitations

- Color picker can't be displayed on top of the start menu or action center (you can still pick a color).
- If the currently focused application was started with an administrator elevation (Run as administrator), the Color Picker activation shortcut will not work, unless PowerToys was also started with an administrator elevation.

FancyZones utility

6/30/2021 • 7 minutes to read • Edit Online

FancyZones is a window manager utility for arranging and snapping windows into efficient layouts to improve the speed of your workflow and restore layouts quickly. FancyZones allows the user to define a set of window locations for a desktop that are drag targets for windows. When the user drags a window into a zone, the window is resized and repositioned to fill that zone.



```
#include "pch.h"
#include "common/dpi_aware.h"
#include "common/on_thread_executor.h"

#include "FancyZones.h"
#include "lib/Settings.h"
#include "lib/ZoneWindow.h"
#include "lib/RegistryHelpers.h"
#include "lib/JsonHelpers.h"
#include "lib/ZoneSet.h"
#include "trace.h"
#include "VirtualDesktopUtils.h"

#include <functional>
#include <common/common.h>
#include <common/window_helpers.h>
#include <common/notifications.h>
#include <lib/util.h>
#include <unordered_set>

#include <common/notifications/fancyzones_notificat
```

```
35>trace.cpp
35>tray_icon.cpp
35>unhandled_exception_handler.cpp
35>update_utils.cpp
35>update_state.cpp
35>win_hook_event.cpp
35>Generating code
35>Previous IPDB not found, fall back to full compilation.
```

```
Microsoft Windows [Version 10.0.18363.752]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\enrico>_
```

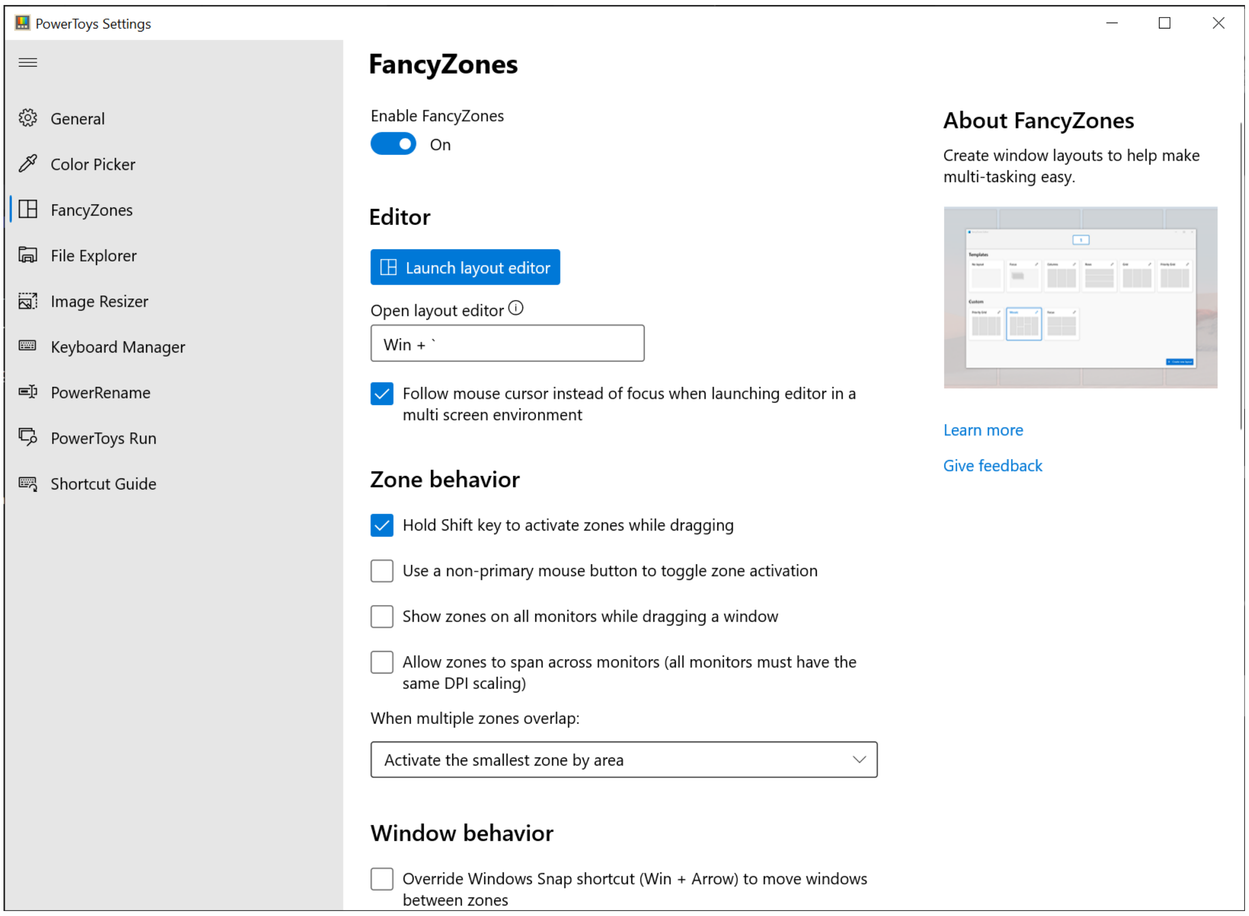
Getting started

Enable

To get started using FancyZones, you need to enable the utility in PowerToys settings and then invoke the FancyZones editor UI.

Launch zones editor

Launch the zones editor using the button in the PowerToys Settings menu or by pressing `Win+`` (note that this shortcut can be changed in the settings dialog).



PowerToys Settings

General
Color Picker
FancyZones
File Explorer
Image Resizer
Keyboard Manager
PowerRename
PowerToys Run
Shortcut Guide

FancyZones

Enable FancyZones
On

Editor

Launch layout editor

Open layout editor
Win + `

☑ Follow mouse cursor instead of focus when launching editor in a multi screen environment

Zone behavior

☑ Hold Shift key to activate zones while dragging
☐ Use a non-primary mouse button to toggle zone activation
☐ Show zones on all monitors while dragging a window
☐ Allow zones to span across monitors (all monitors must have the same DPI scaling)

When multiple zones overlap:
Activate the smallest zone by area

Window behavior

☐ Override Windows Snap shortcut (Win + Arrow) to move windows between zones

About FancyZones

Create window layouts to help make multi-tasking easy.

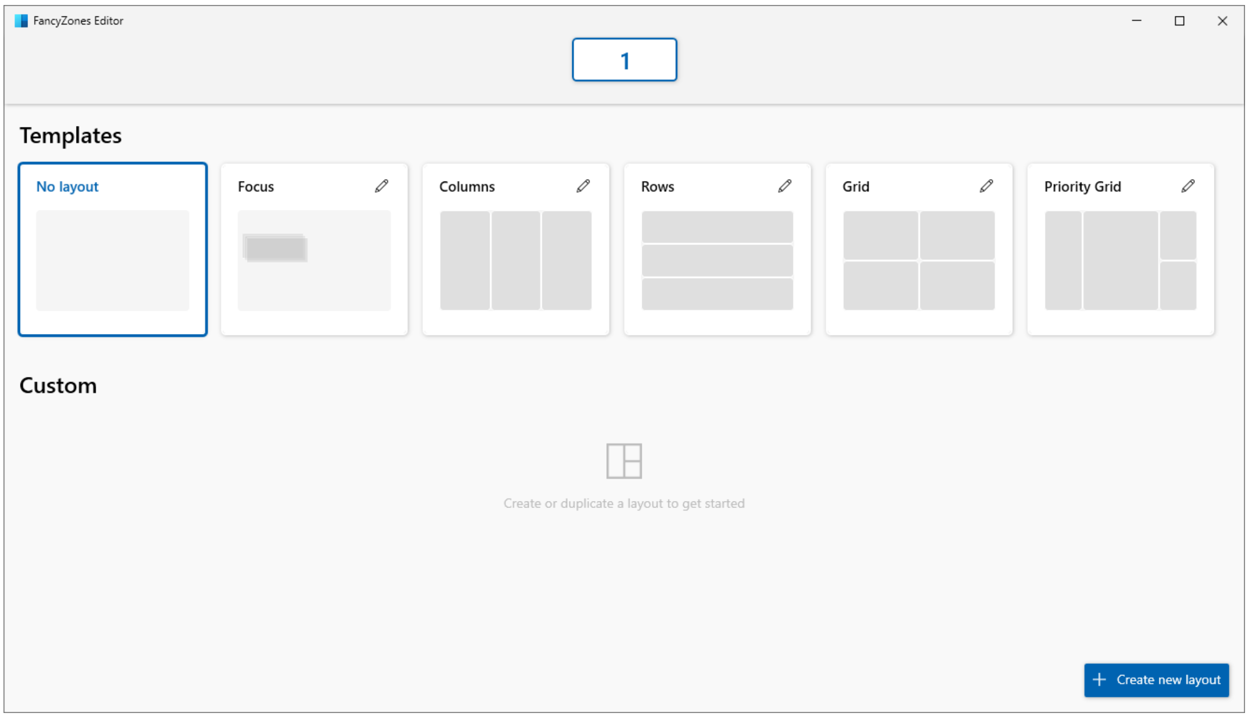
Learn more
Give feedback

Elevated permission admin apps

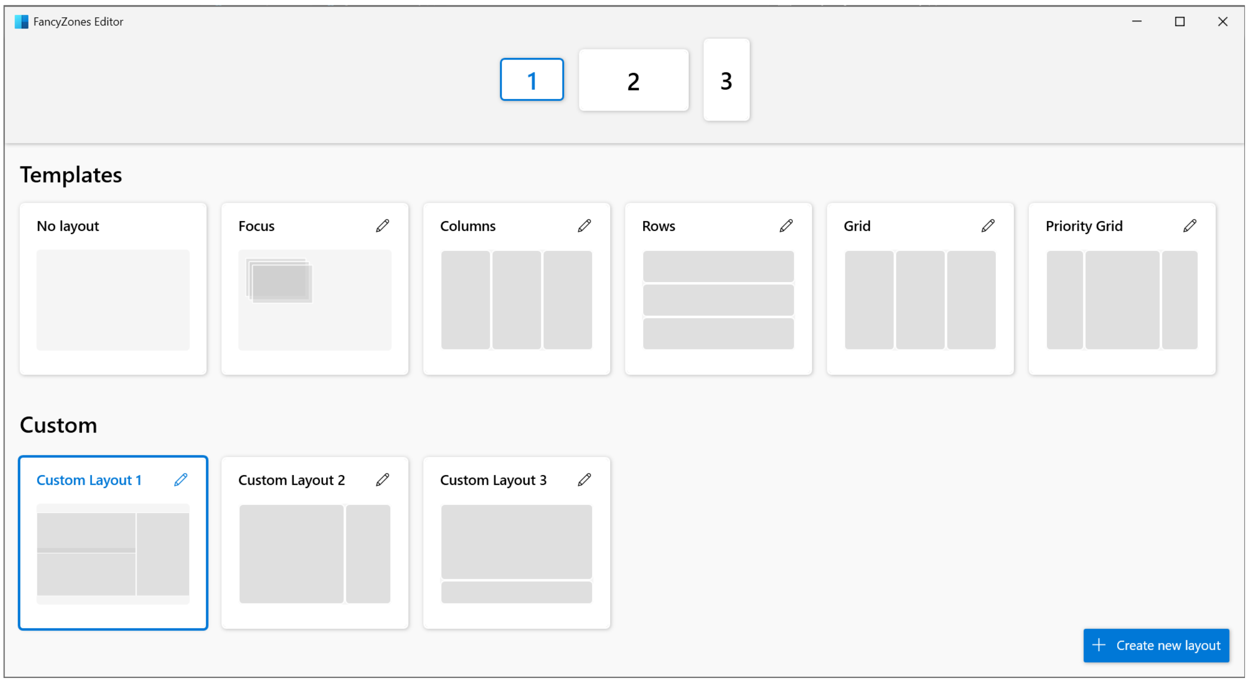
If you have applications that are elevated, run in administrator mode, read PowerToys and running as administrator for more information.

Choose your layout (Layout Editor)

When first launched, the zones editor presents a list of layouts that can be adjusted by how many windows are on the monitor. Choosing a layout shows a preview of that layout on the monitor. The selected layout is applied automatically. Double-clicking the layout will apply it and automatically dismiss the editor.



If multiple displays are in use, the editor will detect the available monitors and display them for the user to choose between. The chosen monitor will then be the target of the selected layout.

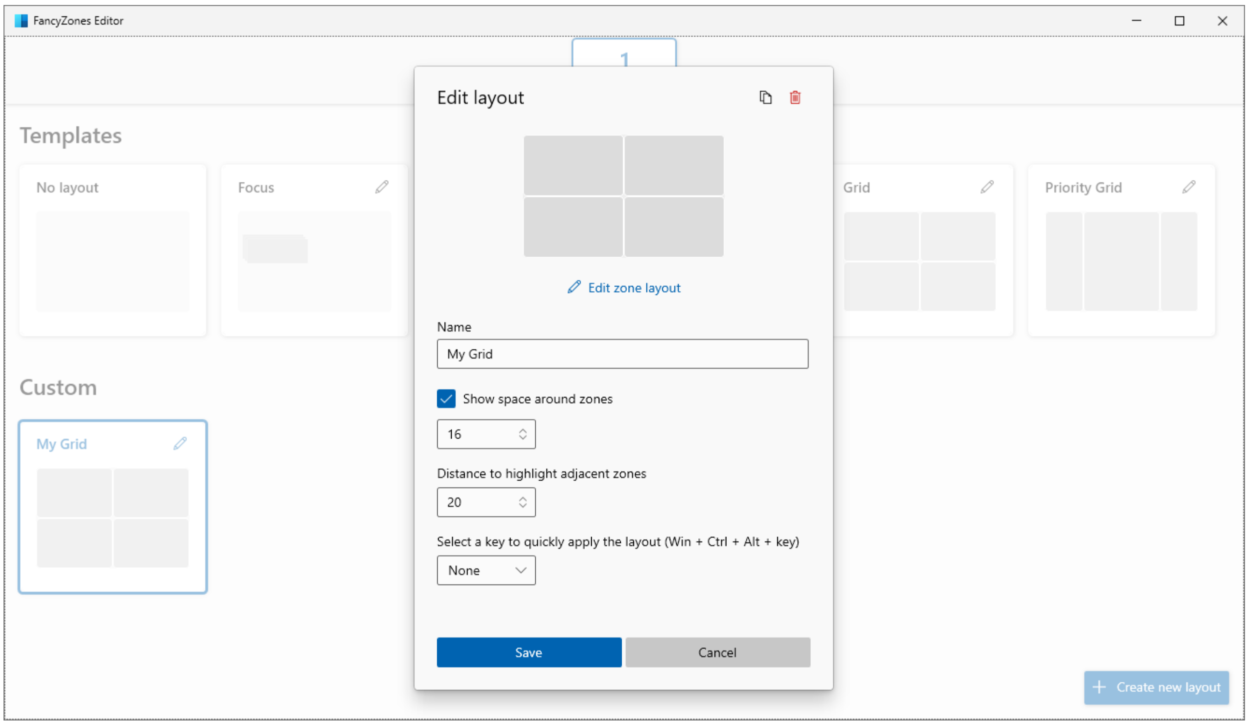


Space around zones

The **Show space around zones** toggle enables you to determine what sort of border or margin will surround each FancyZone window. The **Space around zones** field enables you to set a custom value for how wide the border will be.

The **Distance to highlight adjacent zones** enables you to set a custom value for the amount of space between FancyZone windows until they snap together, or before both are highlighted enabling them to merge together.

With the Zones Editor open, check and uncheck the **Show space around zones** box after changing the values to see the new value applied.



Edit layout

Name: My Grid

☑ Show space around zones

16

Distance to highlight adjacent zones

20

Select a key to quickly apply the layout (Win + Ctrl + Alt + key)

None

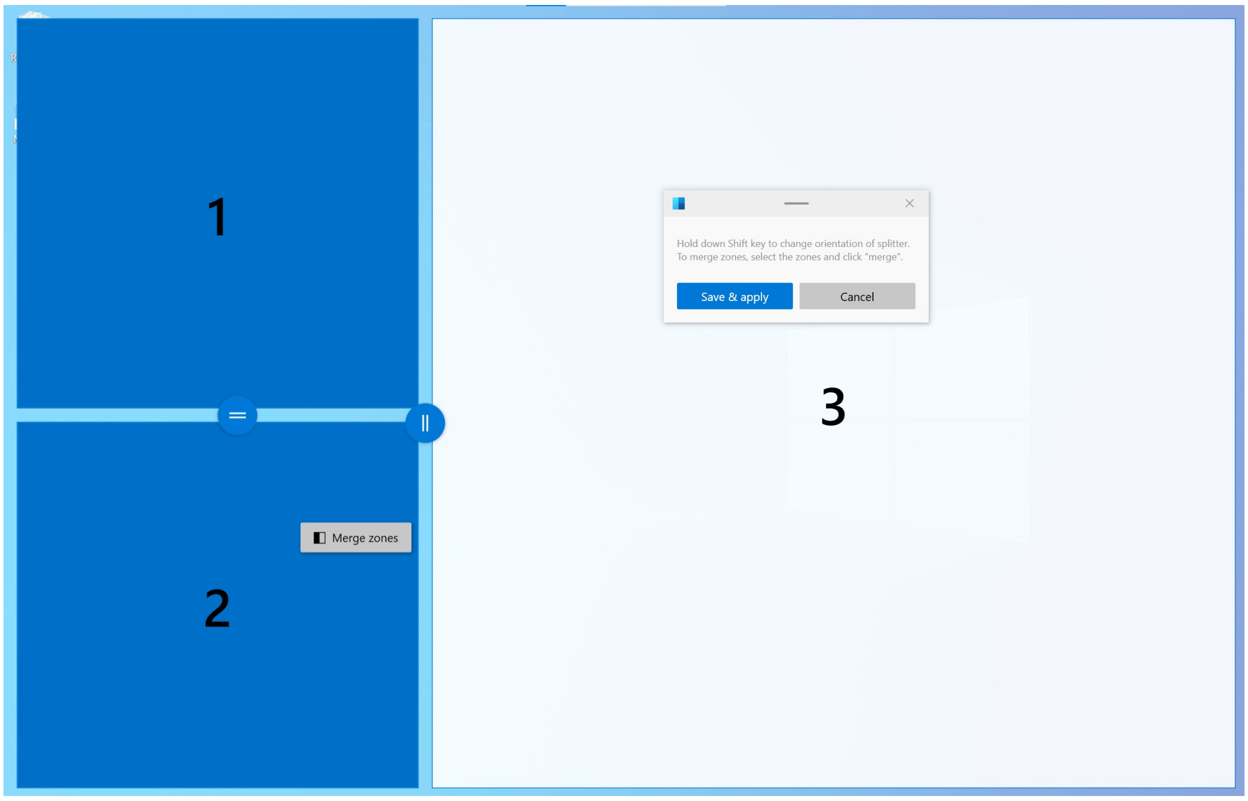
Creating a custom layout

The zones editor also supports creating and saving custom layouts. Select the **+ Create new layout** button at the bottom-right.

There are two ways to create custom zone layouts: **Grid** layout and **Canvas** layout. These can also be thought of as subtractive and additive models.

The subtractive **Grid** model starts with a three column grid and allows zones to be created by splitting and merging zones, resizing the gutter between zones as desired.

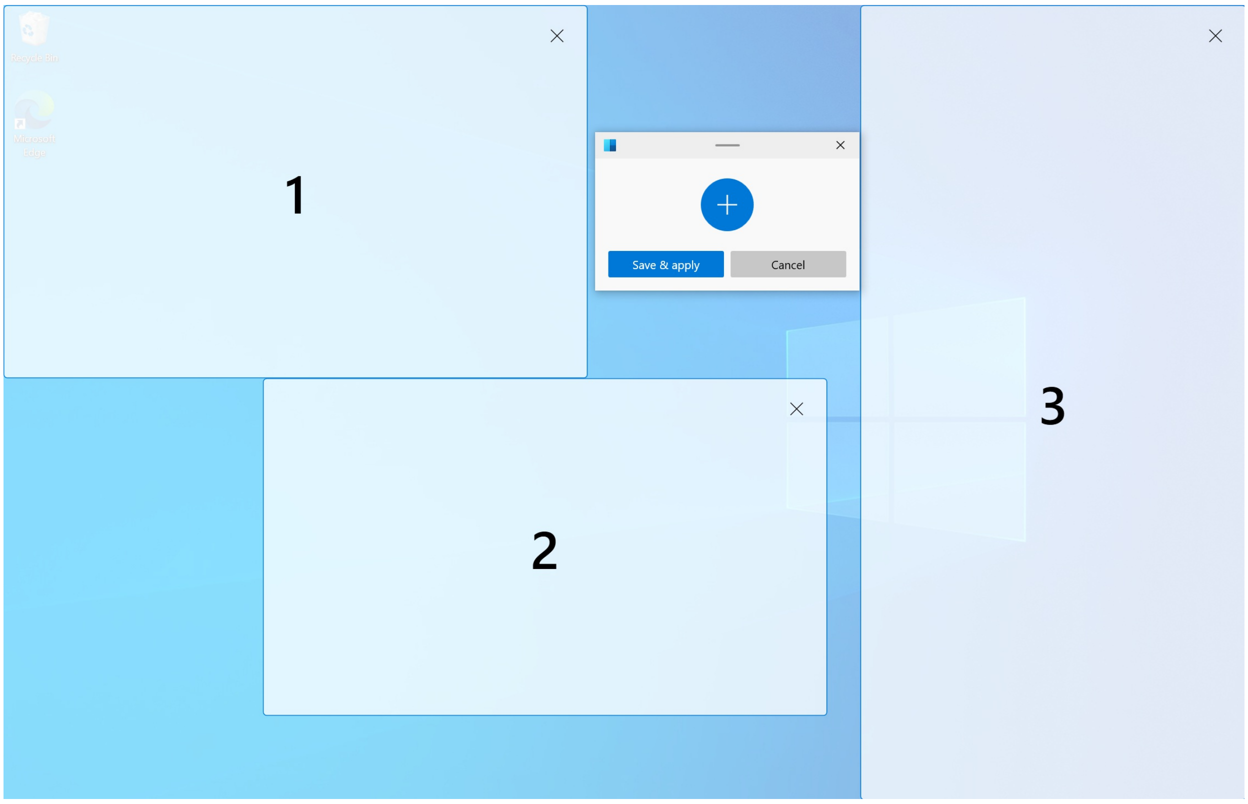
To merge two zones, select and hold the left mouse button and drag the mouse until a second zone is selected, then release the button and a popup menu will show up.



Hold down Shift key to change orientation of splitter.
To merge zones, select the zones and click "merge".

The additive **Canvas** model starts with a blank layout and supports adding zones that can be dragged and resized similar to windows.

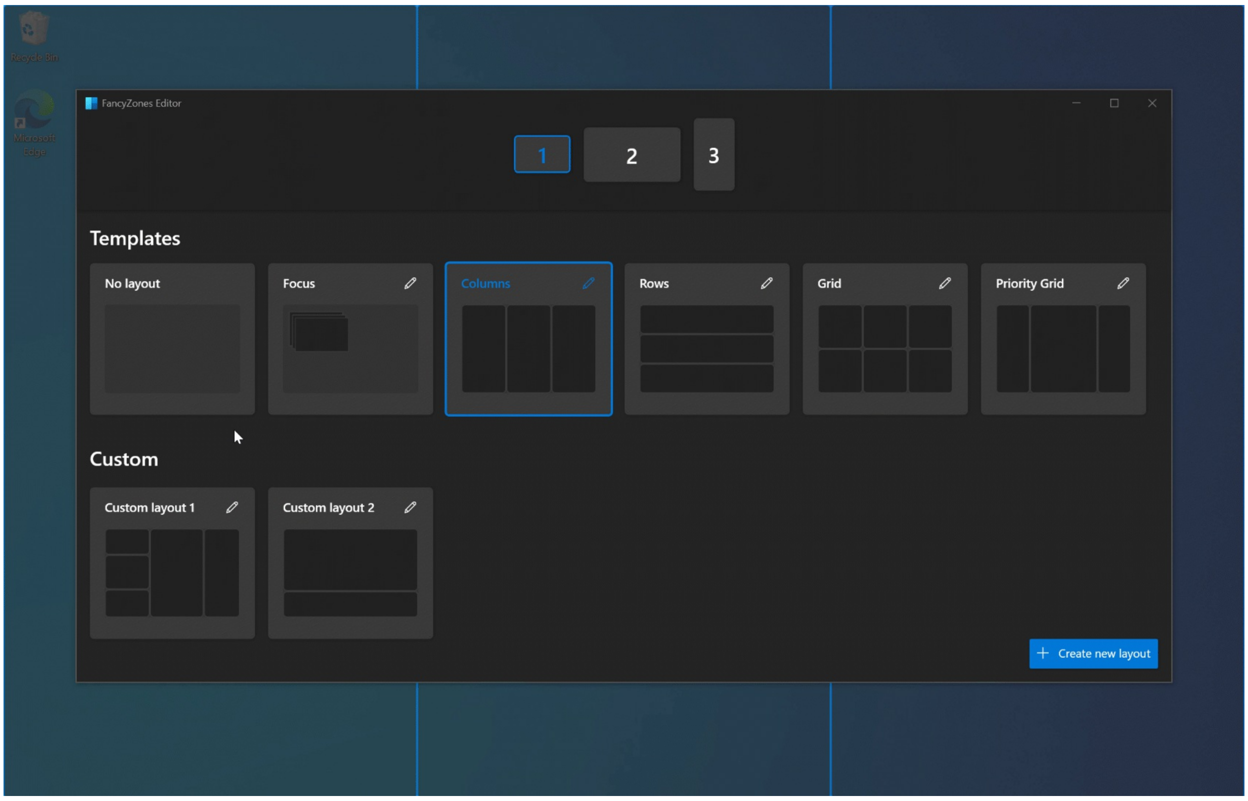
Canvas layout also has keyboard support for zone editing. Use the `Arrows` keys to move a zone by 10 pixels, or `Ctrl + Arrows` to move a zone by 1 pixel. Use the `Shift + Arrows` keys to resize a zone by 10 pixels (5 per edge), or `Ctrl + Shift + Arrows` to resize a zone by 2 pixels (1 per edge). To switch between the editor and dialog, press the `Ctrl + Tab` keys.



Quickly changing between layouts

With a custom layout, this layout can be configured to a user-defined hotkey to quickly apply it to the desired desktop. The hotkey can be set by opening the custom layout's edit menu. Once set, the custom layout can be applied by pressing the `Win ⊞ + Ctrl + Alt + [hotkey]` binding. The layout can also be applied by pressing the hotkey when dragging a window.

In the demo below, we start with a default template applied to the screen and 2 custom layouts that we assign hotkeys for. We then use the `Win ⊞ + Ctrl + Alt + [hotkey]` binding to apply the first custom layout and bind a window to it. Finally, we apply the second custom layout while dragging a window and bind the window to it.

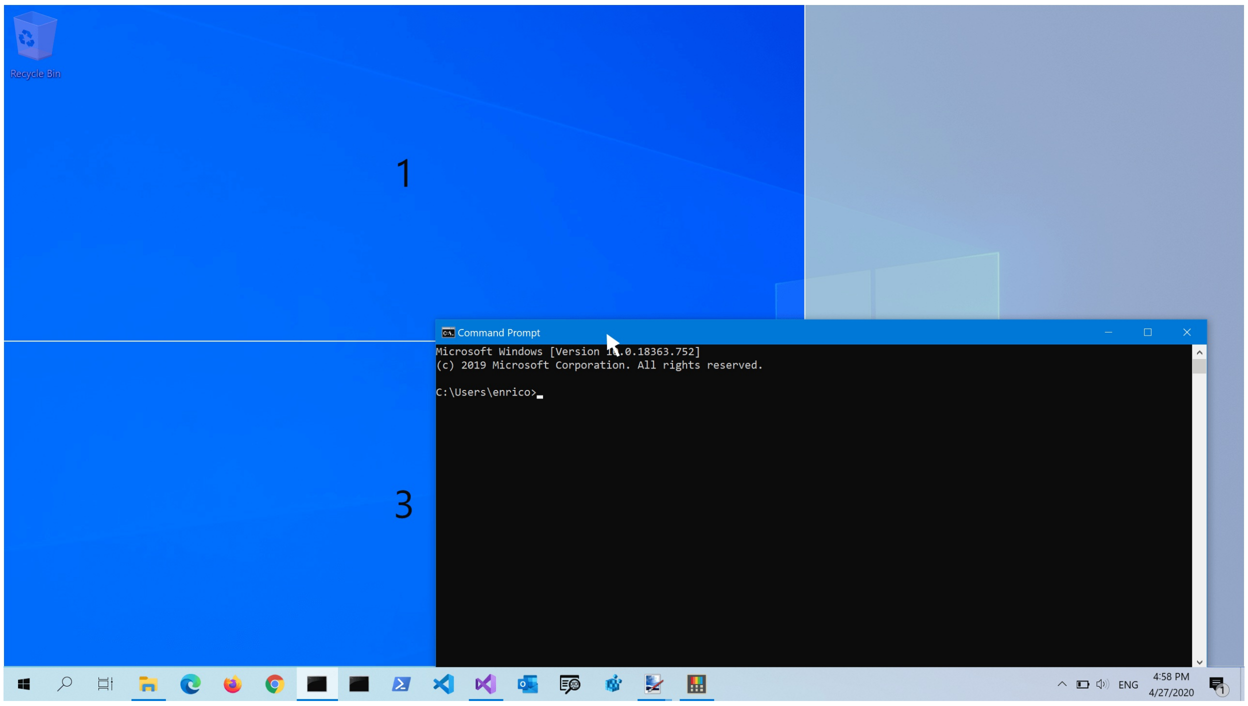


Snapping a window to two or more zones

If two zones are adjacent, a window can be snapped to the sum of their area (rounded to the minimum rectangle that contains both). When the mouse cursor is near the common edge of two zones, both zones are activated simultaneously, allowing you to drop the window into both zones.

It's also possible to snap to any number of zones: first drag the window until one zone is activated, then press and hold the `Control` key while dragging the window to select multiple zones.

To snap a window to multiple zone using only the keyboard, first turn on the two options `Override Windows Snap hotkeys (Win+Arrow) to move between zones` and `Move windows based on their position`. After snapping a window to one zone, use `Win` + `Control` + `Alt` + arrows to expand the window to multiple zones.



```
Microsoft Windows [Version 10.0.18363.752]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\enrico>_
```

Shortcut Keys

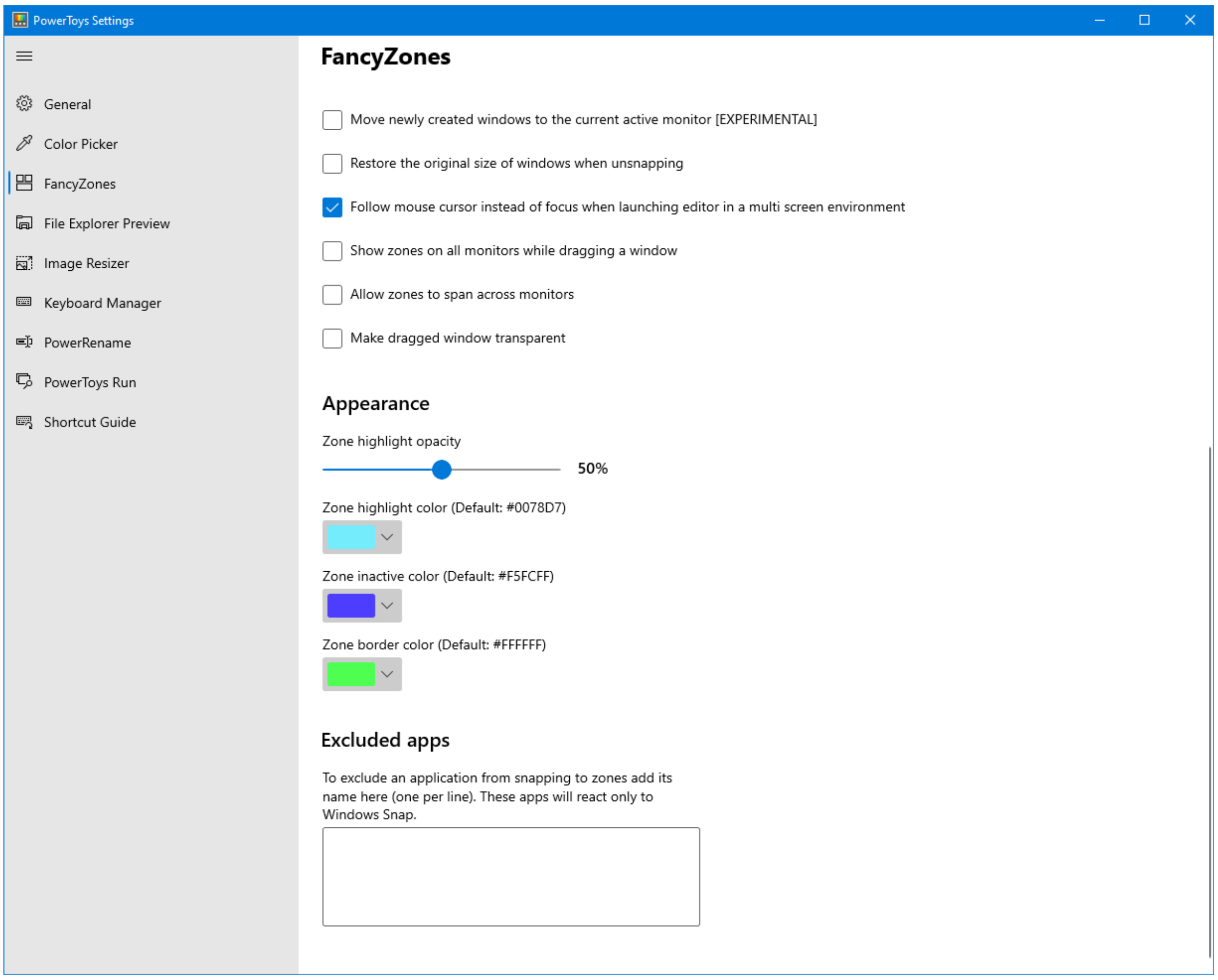
SHORTCUT	ACTION
Win ⊞ + `	The Windows key + backtick (⊞ + `) launches the editor (this shortcut is editable in the settings dialog)
Win ⊞ + Left/Right Arrow	Move focused window between zones (only if `Override Windows Snap hotkeys` setting is turned on, in that case only the `Windows ⊞ key + Left Arrow` and `Windows key ⊞ + Right Arrow` are overridden, while the `Win ⊞ + Up Arrow` and `Win ⊞ + Down Arrow` keep working as usual)

FancyZones doesn't override the Windows 10 `Win ⊞ + Shift + Arrow` to quickly move a window to an adjacent monitor.

Settings

SETTING	DESCRIPTION
Configure the zone editor hotkey	To change the default hotkey, click on the textbox (it's not necessary to select or delete the text) and then press on the keyboard the desired key combination
Hold Shift key to activate zones while dragging	Toggles between auto-snap mode with the shift key disabling snapping during a drag and manual snap mode where pressing the shift key during a drag enables snapping
Use a non-primary mouse button to toggle zone activation	When this option is on, clicking a non-primary mouse button toggles the zones activation
Override Windows Snap hotkeys (Win ⊞ +Arrow) to move between zones	When this option is on and FancyZones is running, it overrides two Windows Snap keys: `Win ⊞ + Left Arrow` and `Win ⊞ + Right Arrow`
Move windows based on their position	Allows to use Win ⊞ + Up/Down/Right/Left arrows to snap a window based on its position relatively to the zone layout
Move windows between zones across all monitors	When this option is off, snapping with Win ⊞ + Arrow cycles the window through the zones on the current monitor, when is on, it cycles the window through all the zones on all monitors
Keep windows in their zones when the screen resolution changes	After a screen resolution change, if this setting is enabled, FancyZones will resize and reposition windows into the zones they were previously in
During zone layout changes, windows assigned to a zone will match new size/positions	When this option is on, FancyZones will resize and position windows into the new zone layout by maintaining the previous zone number location of each window
Move newly created windows to the last known zone	Automatically move a newly opened window into the last zone location that application was in
Move newly created windows to the current active monitor [EXPERIMENTAL]	When this option is on, and "Move newly created windows to the last known zone" is off or the application doesn't have a last known zone, it keeps the application on the current active monitor
Restore the original size of windows when unsnapping	When this option is on, unsnapping a window will restore its size as before it was snapped
Follow mouse cursor instead of focus when launching editor in a multi-monitor environment	When this option is on, the editor hotkey will launch the editor on the monitor where the mouse cursor is, when this option is off, the editor hotkey will launch the editor on monitor where the current active window is
Show zones on all monitors while dragging a window	By default FancyZones shows only the zones available on the current monitor, this feature may have performance impact when turned on
Allow zones to span across monitors (all monitors must have the same DPI scaling)	This option allows to treat all connected monitors as one large screen. To work correctly it requires all monitors to have the same DPI scaling factor

SETTING	DESCRIPTION
Make dragged window transparent	When the zones are activated, the dragged window is made transparent to improve the zones visibility
Zone highlight color (Default #008CFF)	The color that a zone becomes when it is the active drop target during a window drag
Zone Inactive color (Default #F5FCFF)	The color that zones become when they are not an active drop during a window drag
Zone border color (Default #FFFFFF)	The color of the border of active and inactive zones
Zone opacity (%) (Default 50%)	The percentage of opacity of active and inactive zones
Exclude applications from snapping to zones	Add the applications name, or part of the name, one per line (e.g., adding `Notepad` will match both `Notepad.exe` and `Notepad++.exe`, to match only `Notepad.exe` add the `.exe` extension)



FancyZones

- [ ] Move newly created windows to the current active monitor [EXPERIMENTAL]
- [ ] Restore the original size of windows when unsnapping
- [x] Follow mouse cursor instead of focus when launching editor in a multi screen environment
- [ ] Show zones on all monitors while dragging a window
- [ ] Allow zones to span across monitors
- [ ] Make dragged window transparent

Appearance

Zone highlight opacity 50%

Zone highlight color (Default: #0078D7)

Zone inactive color (Default: #F5FCFF)

Zone border color (Default: #FFFFFF)

Excluded apps

To exclude an application from snapping to zones add its name here (one per line). These apps will react only to Windows Snap.

File Explorer add-ons utility

6/1/2021 • 2 minutes to read • Edit Online

File Explorer add-ons currently include:

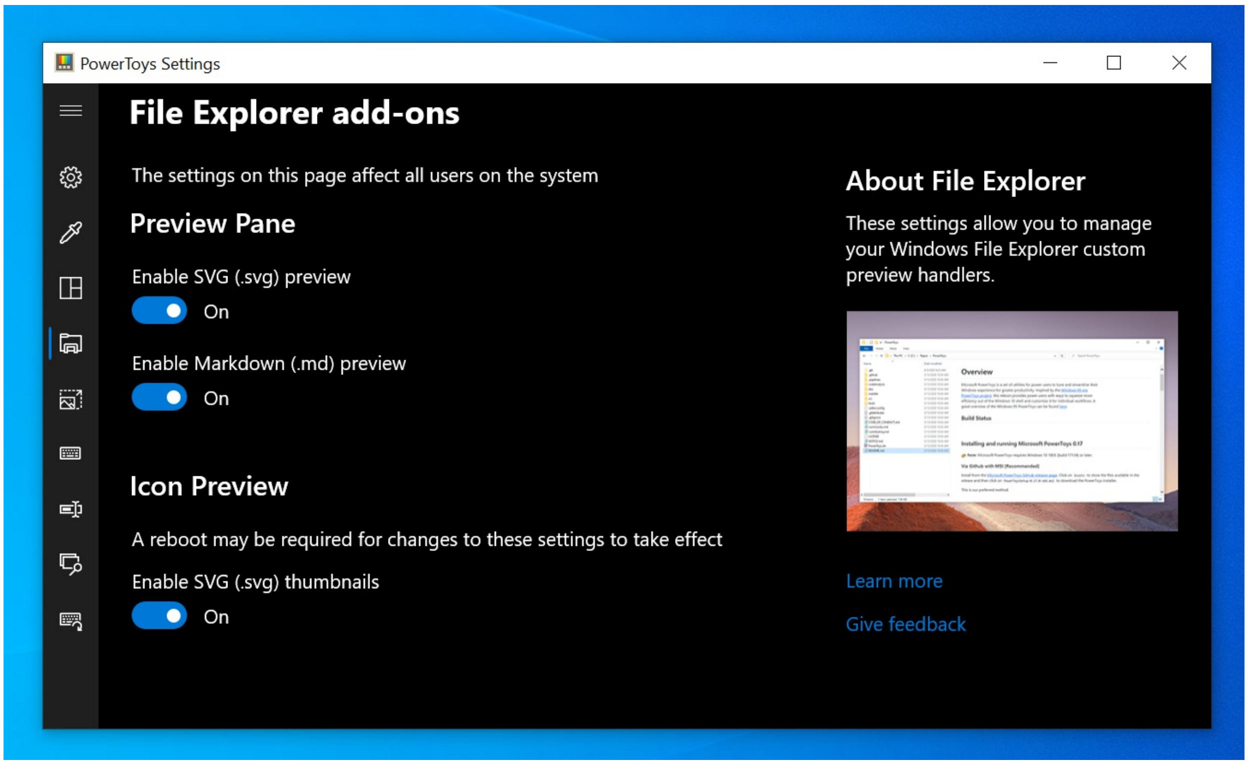
- Preview Pane rendering of SVG icons (.svg)
- Preview Pane rendering of Markdown files (.md)

Preview Pane

Preview Pane is an existing feature in the Windows File Explorer which shows a lightweight, rich, read-only preview of the file's contents in the view's reading pane. PowerToys adds two extensions, Markdown and SVG.

Enabling Preview Pane

To enable, first ensure that "Enable SVG preview", "Enable SVG thumbnails", and "Enable Markdown preview" are all set to On in the PowerToys Settings.



PowerToys Settings

File Explorer add-ons

The settings on this page affect all users on the system

Preview Pane

Enable SVG (.svg) preview: On

Enable Markdown (.md) preview: On

Icon Preview

A reboot may be required for changes to these settings to take effect

Enable SVG (.svg) thumbnails: On

About File Explorer

These settings allow you to manage your Windows File Explorer custom preview handlers.

Next, open Windows File Explorer, select the **View** tab in the File Explorer ribbon, then select `Preview Pane`.

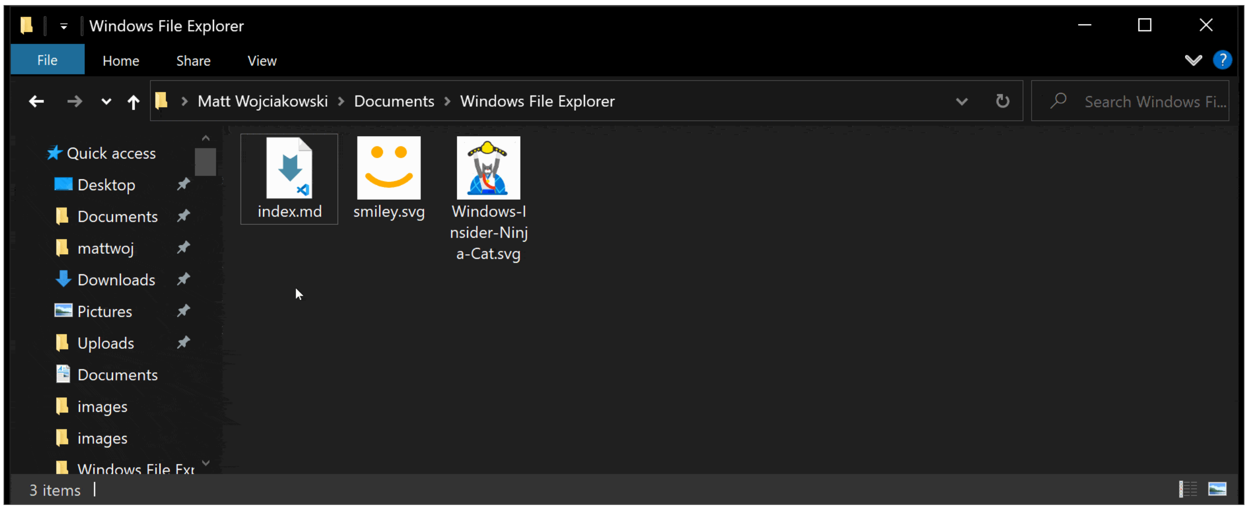
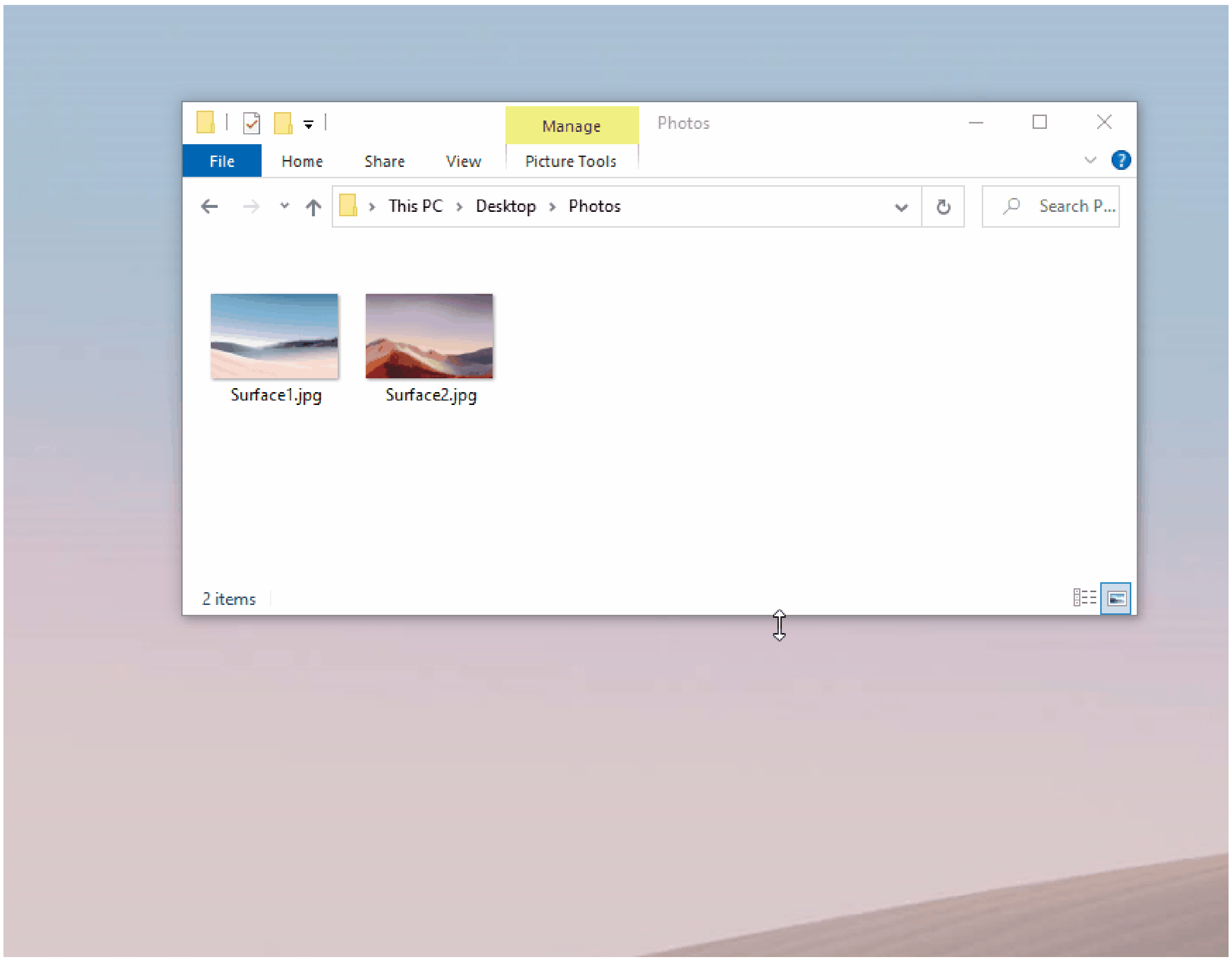

Windows File Explorer
File
Home
Share
View
Matt Wojciakowski > Documents > Windows File Explorer
Search Windows Fi...
Quick access
Desktop
Documents
mattwoj
Downloads
Pictures
Uploads
Documents
images
images
Windows File Exp
index.md
smiley.svg
Windows-Insider-Ninja-Cat.svg
3 items

Image Resizer utility

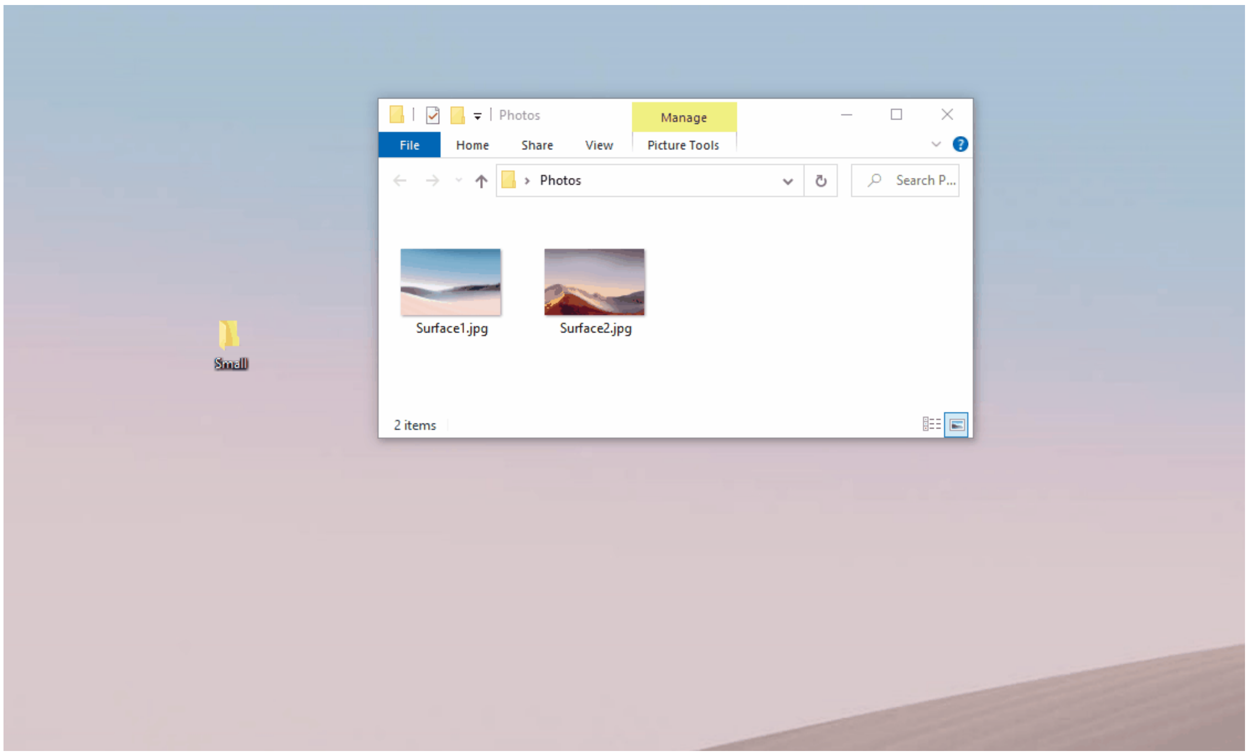
6/1/2021 • 2 minutes to read • Edit Online

Image Resizer is a Windows shell extension for bulk image-resizing. After installing PowerToys, right-click on one or more selected image files in File Explorer, and then select **Resize pictures** from the menu.



Drag and Drop

Image Resizer also allows you to resize images by dragging and dropping your selected files **with the right mouse button**. This allows you to quickly save your resized pictures in another folder.



Settings

Inside the PowerToys Image Resizer tab, you can configure the following settings.

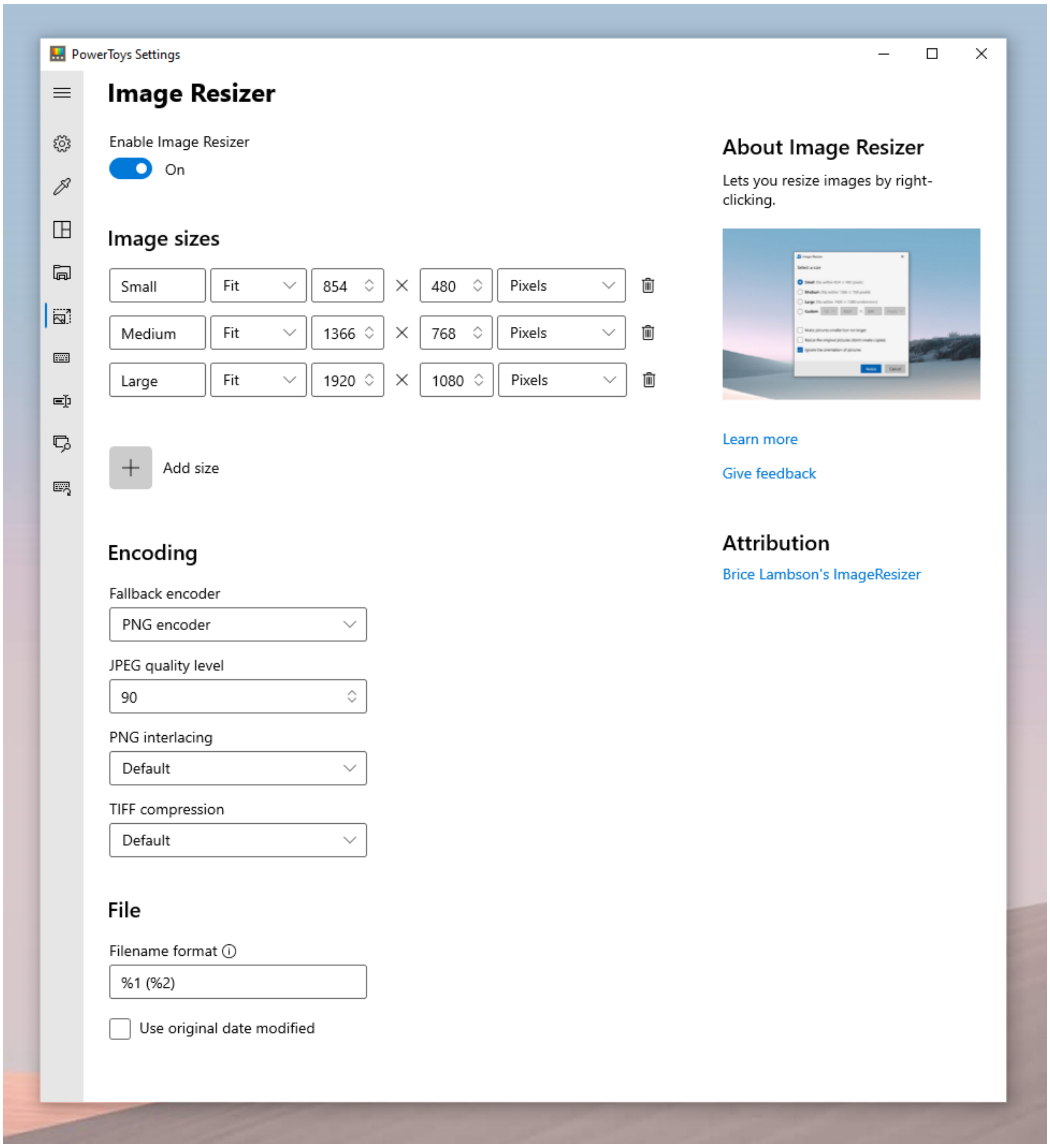

PowerToys Settings

Image Resizer

Enable Image Resizer
On

Image sizes

Small	Fit	854	×	480	Pixels
Medium	Fit	1366	×	768	Pixels
Large	Fit	1920	×	1080	Pixels

Add size

Encoding

Fallback encoder: PNG encoder

JPEG quality level: 90

PNG interlacing: Default

TIFF compression: Default

File

Filename format: %1 (%2)

☐ Use original date modified

About Image Resizer

Lets you resize images by right-clicking.

Learn more

Give feedback

Attribution

Brice Lambson's ImageResizer

Sizes

Add new preset sizes. Each size can be configured as Fill, Fit or Stretch. The dimension to be used for resizing can also be configured as Centimeters, Inches, Percent and Pixels.

Fill vs Fit vs Stretch

- **Fill**: Fills the entire specified size with the image. Scales the image proportionally. Crops the image as needed.
- **Fit**: Fits the entire image into the specified size. Scales the image proportionally. Does not crop the image.
- **Stretch**: Fills the entire specified size with the image. Stretches the image disproportionally as needed. Does not crop the image

The width and height of the specified size may be swapped to match the orientation (portrait/landscape) of the current image. To always use the width and height as specified, un-check: **Ignore the orientation of pictures**.

Fallback encoding

The fallback encoder is used when the file cannot be saved in it's original format. For example, the Windows Metafile (.wmf) image format has a decoder to read the image, but no encoder to write a new image. In this case, the image cannot be saved in it's original format. Image Resizer enables you to specify what format the fallback encoder will use: PNG, JPEG, TIFF, BMP, GIF, or WMPhoto settings. *This is not a file type conversion tool, but only works as a fallback for unsupported file formats.*

File

The file name of the resized image can be modified with the following parameters:

- `%1` : Original filename
- `%2` : Size name (as configured in the PowerToys Image Resizer settings)
- `%3` : Selected width
- `%4` : Selected height
- `%5` : Actual height
- `%6` : Actual width

For example, setting the filename format to: `%1 (%2)` on the file `example.png` and selecting the `Small` file size setting, would result in the file name `example (Small).png` .

Setting the format to `%1_%4` on the file `example.jpg` and selecting the size setting `Medium 1366 x 768px` would result in the file name: `example_768.jpg` .

You can also choose to retain the original *last modified* date on the resized image.

Auto width/height

You can leave the height or width empty. This will honor the specified dimension and "lock" the other dimension to a value proportional to the original image aspect ratio.

Sub-directories

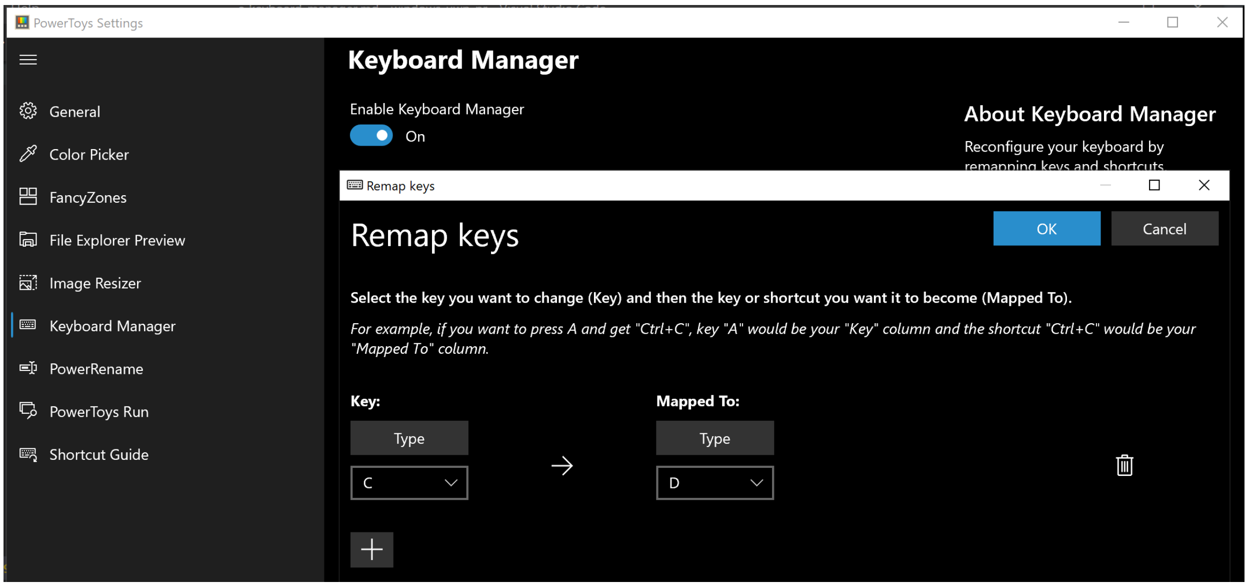
You can specify a directory in the filename format to group resized images into sub-directories. For example, a value of `%2\%1` would save the resized image to `Small\Sample.jpg`

Keyboard Manager utility

6/1/2021 • 7 minutes to read • Edit Online

The PowerToys Keyboard Manager enables you to redefine keys on your keyboard.

For example, you can exchange the letter A for the letter D on your keyboard. When you select the A key, a D will display.


PowerToys Settings

General
Color Picker
FancyZones
File Explorer Preview
Image Resizer
Keyboard Manager
PowerRename
PowerToys Run
Shortcut Guide

Keyboard Manager

Enable Keyboard Manager
On

About Keyboard Manager
Reconfigure your keyboard by remapping keys and shortcuts

Remap keys

Remap keys

OK
Cancel

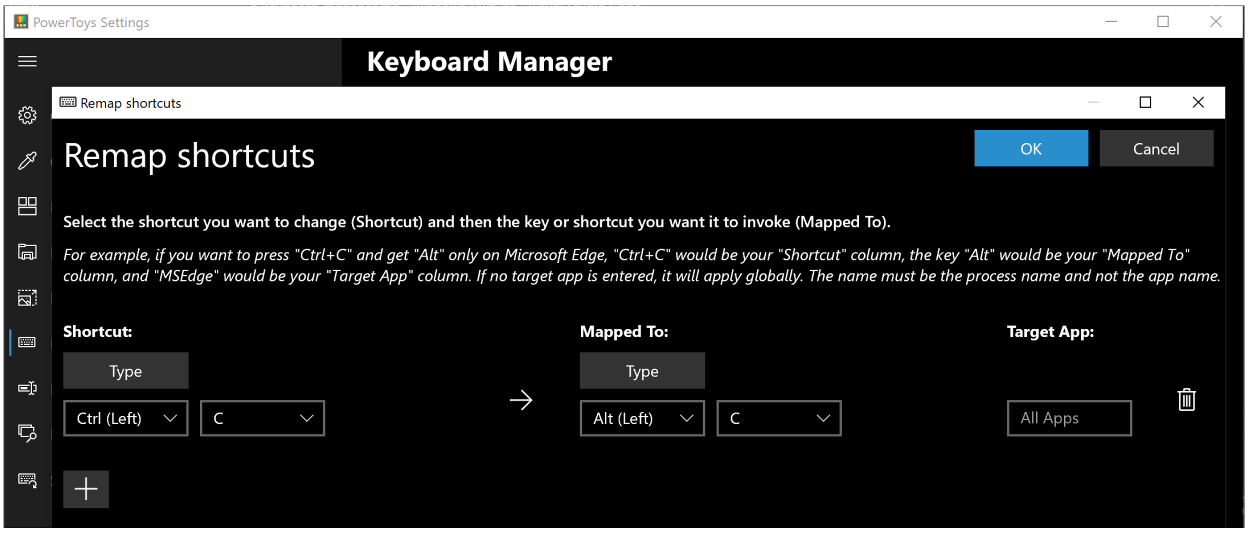
Select the key you want to change (Key) and then the key or shortcut you want it to become (Mapped To).

For example, if you want to press A and get "Ctrl+C", key "A" would be your "Key" column and the shortcut "Ctrl+C" would be your "Mapped To" column.

Key: Type — C → Mapped To: Type — D

You can also exchange shortcut key combinations. For example, the shortcut key, `Ctrl+C`, will copy text in Microsoft Word. With PowerToys Keyboard Manager utility, you can exchange that shortcut for ⊞ `Win+C`). Now, ⊞ `Win+C`) will copy text. If you do not specify a targeted application in PowerToys Keyboard Manager, the shortcut exchange will be applied globally across Windows.

PowerToys Keyboard Manager must be enabled (with PowerToys running in the background) for remapped keys and shortcuts to be applied. If PowerToys is not running, key remapping will no longer be applied.


PowerToys Settings

Keyboard Manager

Remap shortcuts

Remap shortcuts

OK
Cancel

Select the shortcut you want to change (Shortcut) and then the key or shortcut you want it to invoke (Mapped To).

For example, if you want to press "Ctrl+C" and get "Alt" only on Microsoft Edge, "Ctrl+C" would be your "Shortcut" column, the key "Alt" would be your "Mapped To" column, and "MSEdge" would be your "Target App" column. If no target app is entered, it will apply globally. The name must be the process name and not the app name.

Shortcut: Type — Ctrl (Left), C → Mapped To: Type — Alt (Left), C — Target App: All Apps

NOTE

There are some shortcut keys that are reserved for the operating system and cannot be replaced. Keys that cannot be remapped include:

- `⊞ Win` + `L` and `Ctrl` + `Alt` + `Del` cannot be remapped as they are reserved by the Windows OS.
- The `Fn` (function) key cannot be remapped (in most cases). The `F1` - `F12` (and `F13` - `F24`) keys can be mapped.
- `Pause` will only send a single keydown event. So mapping it against the backspace key, for instance, and pressing + holding will only delete a single character.

Settings

To create mappings with Keyboard Manager, you will need to open the PowerToys Settings (search for the PowerToys app in your Windows Start menu, selecting it will open the PowerToys Settings window). Inside PowerToys Settings, on the Keyboard Manager tab, you will see the options to:

- Launch the Remap Keyboard settings window by selecting `Remap a Key`
- Launch the Remap Shortcuts settings window by selecting the `Remap a shortcut`

Remap Keys

To remap a key, changing it to new value, launch the Remap Keyboard settings window with the `Remap a Key` button. When first launched, no predefined mappings will be displayed. You must select the + button to add a new remap.

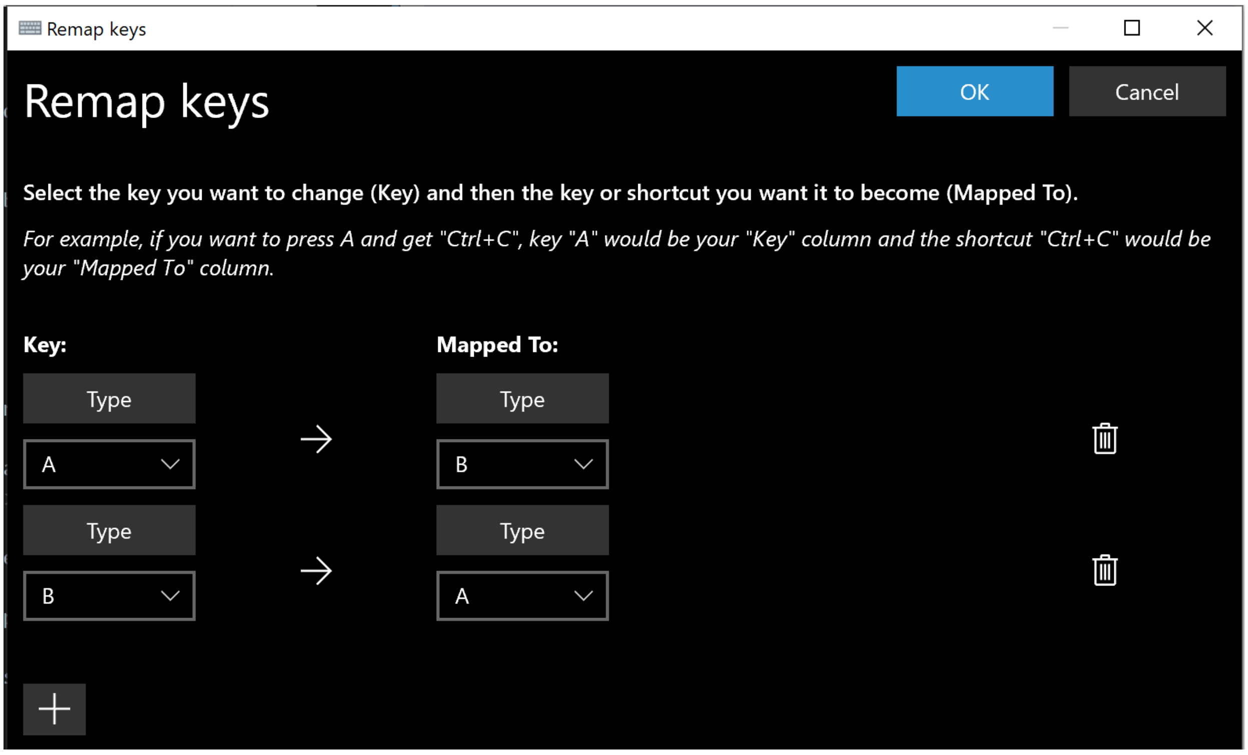
Once a new remap row appears, select the key whose output you want to ***change*** in the “Key” column. Select the new key value to assign in the “Mapped To” column.

For example, if you want to press A and have B appear:

- Key: "A"
- Mapped To: "B"

To swap key positions between the "A" and "B" keys, add another remapping with:

- Key: "B"
- Mapped To: "A"


Remap keys

Remap keys

Select the key you want to change (Key) and then the key or shortcut you want it to become (Mapped To).

For example, if you want to press A and get "Ctrl+C", key "A" would be your "Key" column and the shortcut "Ctrl+C" would be your "Mapped To" column.

Key:		Mapped To:
A	→	B
B	→	A

Key to Shortcut

To remap a key to a shortcut (combination of keys), enter the shortcut key combination in the "Mapped To" column.

For example, if you want to select the "C" key and have it result in "Ctrl + V":

- Key: "C"
- Mapped To: "Ctrl + V"

NOTE

Key remapping will be maintained even if the remapped key is used in another shortcut. For example, entering the shortcut "Alt + C" would result as "Alt + Ctrl + V", since the C key has been remapped to "Ctrl + V".

Remap Shortcuts

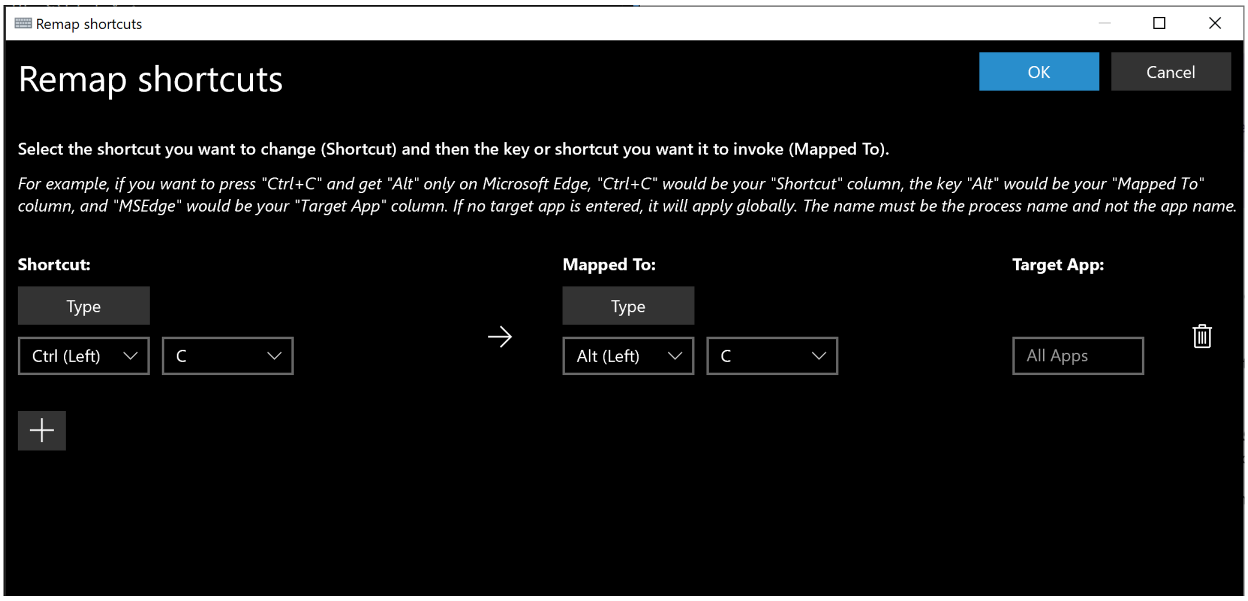
To remap a shortcut key combination, like "Ctrl + v", select `Remap a shortcut` to launch the Remap Shortcuts settings window.

When first launched, no predefined mappings will be displayed. You must select the **+** button to add a new remap.

Once a new remap row appears, select the key whose output you want to ***change*** in the “Shortcut” column. Select the new shortcut value to assign in the “Mapped To” column.

For example, the shortcut `Ctrl+C` copies your selected text. To remap that shortcuts to use the `Alt` key, rather than the `Ctrl` key:

- Shortcut: "Ctrl" + "C"
- Mapped To: "Alt" + "C"


Remap shortcuts

Select the shortcut you want to change (Shortcut) and then the key or shortcut you want it to invoke (Mapped To).

For example, if you want to press "Ctrl+C" and get "Alt" only on Microsoft Edge, "Ctrl+C" would be your "Shortcut" column, the key "Alt" would be your "Mapped To" column, and "MSEdge" would be your "Target App" column. If no target app is entered, it will apply globally. The name must be the process name and not the app name.

There are a few rules to follow when remapping shortcuts (these rules only apply on the "Shortcut" column):

- Shortcuts must begin with a modifier key: `Ctrl`, `Shift`, `Alt`, or ⊞ `Win`
- Shortcuts must end with an action key (all non-modifier keys): A, B, C, 1, 2, 3, etc.
- Shortcuts cannot be longer than 3 keys

Remap a shortcut to a single key

It is possible to remap a shortcut (key combination) to a single key press by selecting the **Remap a Key** button in PowerToys Settings.

For example, to replace the shortcut key ⊞ `Win` + `<` (left arrow) with a single key press, `Alt`:

- Key: "Alt"
- Mapped To: "⊞ Win" (Windows key) + < (left arrow)

NOTE

Shortcut remapping will be maintained even if the remapped key is used in another shortcut. For example, entering the shortcut "Alt" + "Shift", after remapping the "Alt" key as above, would result in "⊞ Win" + < (left arrow) + "Shift". The order of keypress matters in this scenario as the action is executed during keydown, not keyup. Pressing the `Alt` key will first execute ⊞ `Win` + `Left Arrow`. Pressing the `Shift` key first will execute `Shift` + ⊞ `Win` + `Left Arrow`.

App-specific shortcuts

Keyboard Manager enables you to remap shortcuts for only specific apps (rather than globally across Windows).

For example, in the Outlook email app the shortcut "Ctrl + E" is set by default to search for an email. If you prefer instead to set "Ctrl + F" to search your email (rather than forward an email as set by default), you can remap the shortcut with "Outlook" set as your "Target app."

Keyboard Manager uses the process-names (not application names) to target apps. For example, Microsoft Edge is set as "msedge" (process name), not "Microsoft Edge" (application name). To find an application's process name, open PowerShell and enter the command `get-process` or open Command Prompt and enter the command `tasklist`. This will result in a list of process names for all applications you currently have open. Below is a list of a few popular application process names.

APPLICATION	PROCESS NAME
Microsoft Edge	msedge.exe
OneNote	onenote.exe
Outlook	outlook.exe
Teams	Teams.exe
Adobe Photoshop	Photoshop.exe
File Explorer	explorer.exe
Spotify Music	spotify.exe
Google Chrome	chrome.exe
Excel	excel.exe
Word	winword.exe
Powerpoint	powerpnt.exe

Keys that cannot be remapped

There are certain shortcut keys that are not allowed for remapping. These include:

- `Ctrl`+`Alt`+ `Del` (interupt command)
- ⊞ `Win`+`L` (locking your computer)
- The function key, `Fn`, cannot be remapped (in most cases) but the `F1`-`F12` can be mapped.

How to select a key

To select a key or shortcut to remap, you can:

- Use the `Type Key` button.
- Use the drop-down menu.

Once you select the `Type Key / Shortcut` button, a dialogue will pop up in which you can enter the key or shortcut using your keyboard. Once you're satisfied with the output, hold `Enter` to continue. If you'd like to leave the dialogue, hold the `Esc` button.

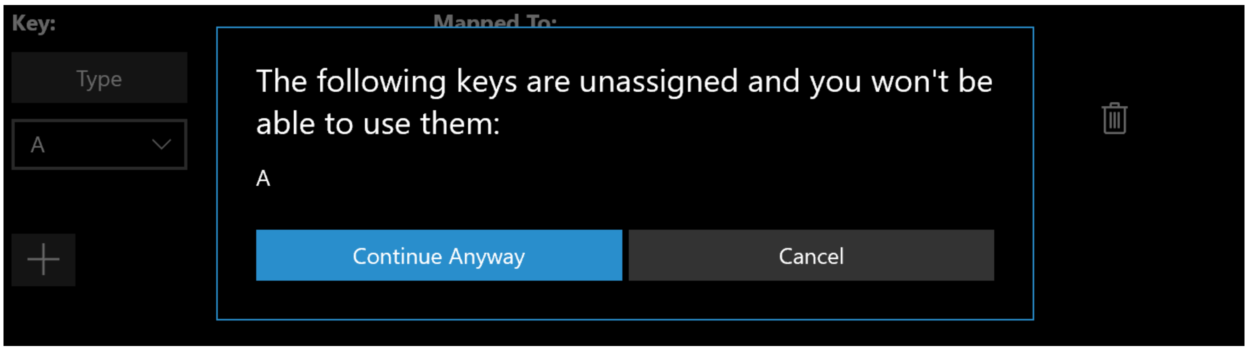
Using the drop-down menu, you can search with the key name and additional drop-down values will appear as you progress. However, you can not use the type-key feature while the drop-down menu is open.

Orphaning Keys

Orphaning a key means that you mapped it to another key and no longer have anything mapped to it.

For example, if the key is remapped from A `->` B, then a key no longer exists on your keyboard that results in A.

To fix this, use + to create another remapped key that is mapped to result in A. To ensure this does not happen by accident, a warning will display for any orphaned keys.



The following keys are unassigned and you won't be able to use them:

A

Frequently asked questions

I remapped the wrong keys, how can I stop it quickly?

For key remapping to work, PowerToys must be running in the background and Keyboard Manager must be enabled. To stop remapped keys, close PowerToys or disable Keyboard Manger in the PowerToys settings.

Can I use Keyboard Manager at my log-in screen?

No, Keyboard Manager is only available when PowerToys is running and doesn't work on any password screen, including Run As Admin.

Do I have to turn off my computer for the remapping to take effect?

No, remapping should occur immediately upon pressing **Apply**.

Where are the Mac/Linux profiles?

Currently Mac and Linux profiles are not included.

Will this work on video games?

It depends on how the game accesses your keys. Certain keyboard APIs do not work with Keyboard Manager.

Will remapping work if I change my input language?

Yes it will. Right now if you remap `A` to `B` on English (US) keyboard and then change the language setting to French, typing `A` on the French keyboard (`Q` on the English US physical keyboard) would result in `B`, this is consistent with how Windows handles multilingual input.

Troubleshooting

If you have tried to remap a key or shortcut and are having trouble, it could be one of the following issues:

- **Run As Admin:** Remapping will not work on an app / window if that window is running in administrator (elevated) mode and PowerToys is not running as administrator. Try running PowerToys as an administrator.
- **Not Intercepting Keys:** Keyboard Manger intercepts keyboard hooks to remap your keys. Some apps that also do this and can interfere with Keyboard Manager. To fix this, go to the settings and disable then re-enable Keyboard Manager.

Known Issues

- Caps light indicator not toggling correctly
- Remaps not working for FancyZones and Shortcut Guide
- Remapping keys like Win, Ctrl, Alt or Shift may break gestures and some special buttons

See the list of open keyboard manager issues.

PowerRename utility

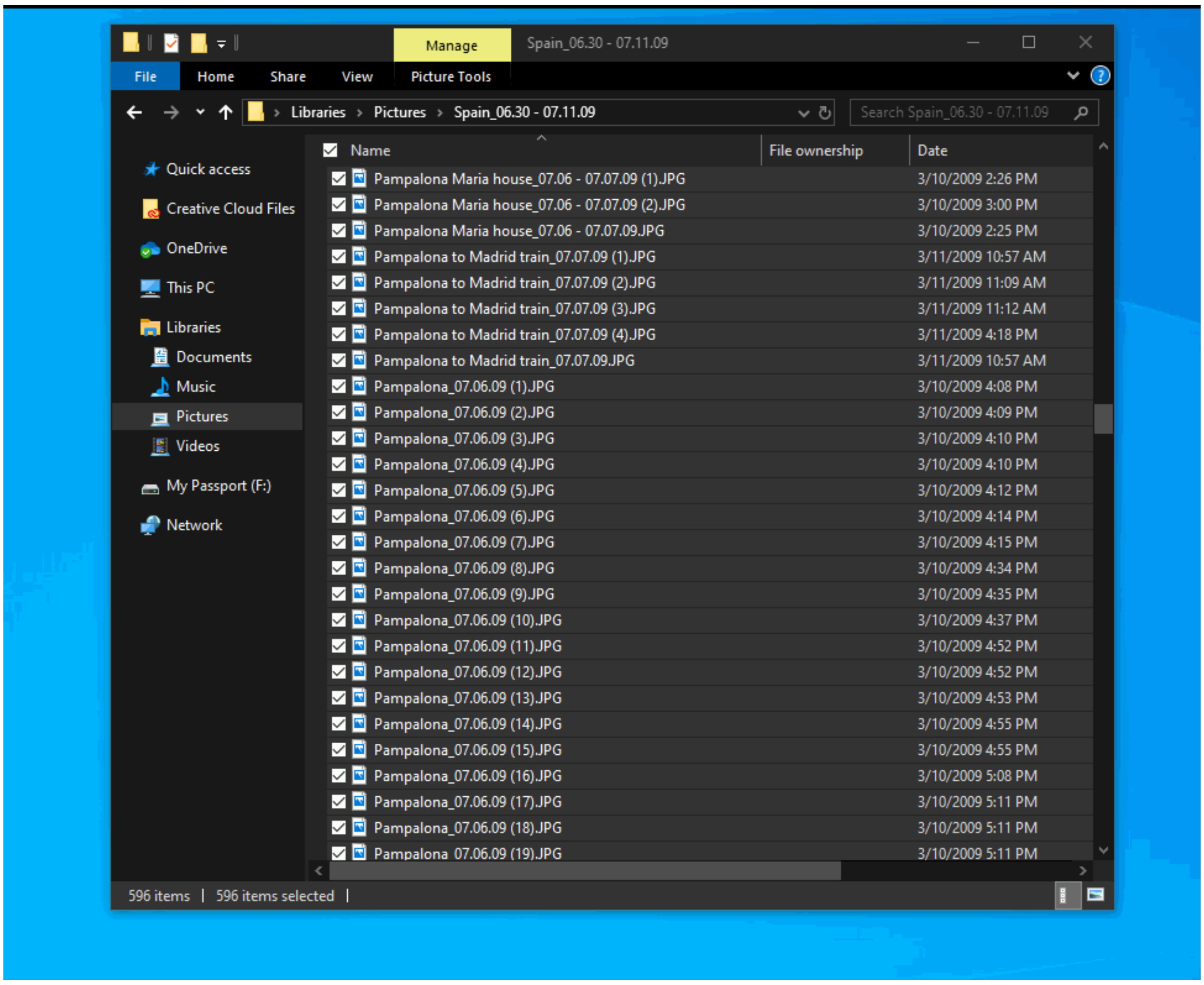
6/1/2021 • 6 minutes to read • Edit Online

PowerRename is a bulk renaming tool that enables you to:

- Modify the file names of a large number of files *(without renaming all of the files the same name)*.
- Perform a search and replace on a targeted section of file names.
- Perform a regular expression rename on multiple files.
- Check expected rename results in a preview window before finalizing a bulk rename.
- Undo a rename operation after it is completed.

Demo

In this demo, all instances of the file name "Pampalona" are replaced with "Pamplona". Since all of the files are uniquely named, this would have taken a long time to complete manually one-by-one. PowerRename enables a single bulk rename. Notice that the "Undo Rename" (Ctrl+Z) command enables the ability to undo the change.

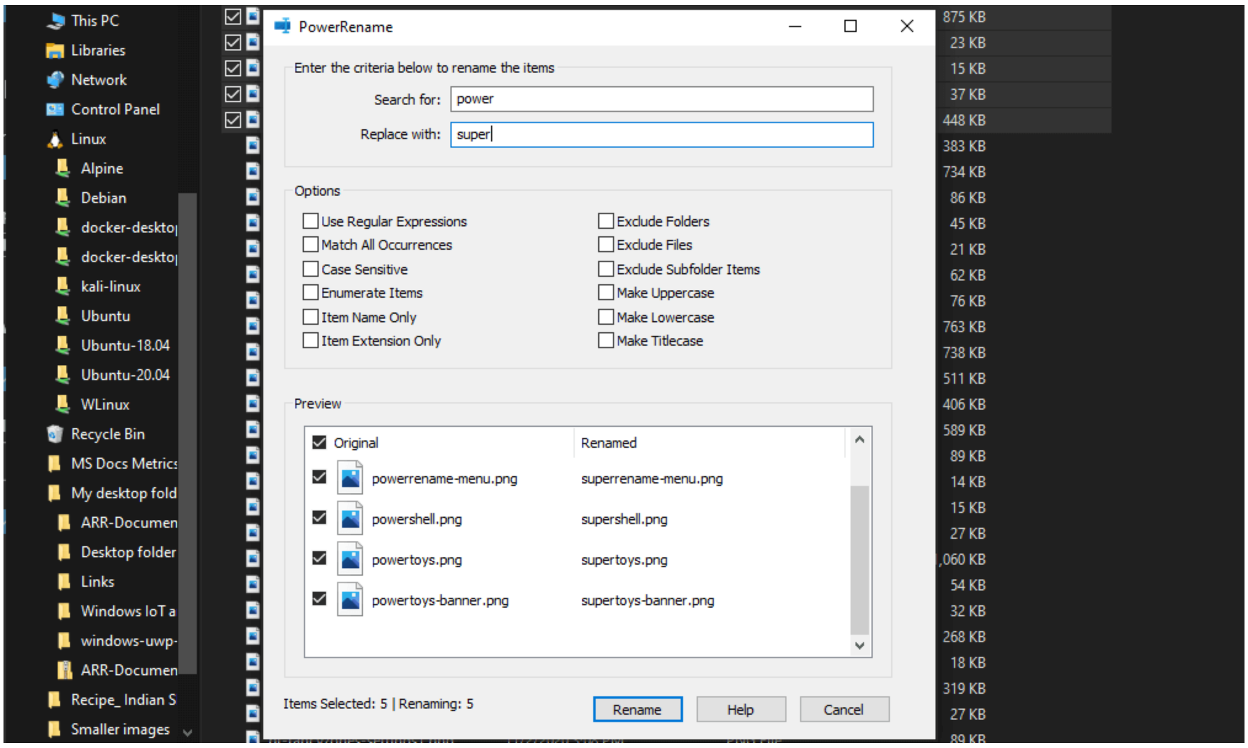


Name	File ownership	Date
Pampalona Maria house_07.06 - 07.07.09 (1).JPG		3/10/2009 2:26 PM
Pampalona Maria house_07.06 - 07.07.09 (2).JPG		3/10/2009 3:00 PM
Pampalona Maria house_07.06 - 07.07.09.JPG		3/10/2009 2:25 PM
Pampalona to Madrid train_07.07.09 (1).JPG		3/11/2009 10:57 AM
Pampalona to Madrid train_07.07.09 (2).JPG		3/11/2009 11:09 AM
Pampalona to Madrid train_07.07.09 (3).JPG		3/11/2009 11:12 AM
Pampalona to Madrid train_07.07.09 (4).JPG		3/11/2009 4:18 PM
Pampalona to Madrid train_07.07.09.JPG		3/11/2009 10:57 AM
Pampalona_07.06.09 (1).JPG		3/10/2009 4:08 PM
Pampalona_07.06.09 (2).JPG		3/10/2009 4:09 PM
Pampalona_07.06.09 (3).JPG		3/10/2009 4:10 PM
Pampalona_07.06.09 (4).JPG		3/10/2009 4:10 PM
Pampalona_07.06.09 (5).JPG		3/10/2009 4:12 PM
Pampalona_07.06.09 (6).JPG		3/10/2009 4:14 PM
Pampalona_07.06.09 (7).JPG		3/10/2009 4:15 PM
Pampalona_07.06.09 (8).JPG		3/10/2009 4:34 PM
Pampalona_07.06.09 (9).JPG		3/10/2009 4:35 PM
Pampalona_07.06.09 (10).JPG		3/10/2009 4:37 PM
Pampalona_07.06.09 (11).JPG		3/10/2009 4:52 PM
Pampalona_07.06.09 (12).JPG		3/10/2009 4:52 PM
Pampalona_07.06.09 (13).JPG		3/10/2009 4:53 PM
Pampalona_07.06.09 (14).JPG		3/10/2009 4:55 PM
Pampalona_07.06.09 (15).JPG		3/10/2009 4:55 PM
Pampalona_07.06.09 (16).JPG		3/10/2009 5:08 PM
Pampalona_07.06.09 (17).JPG		3/10/2009 5:11 PM
Pampalona_07.06.09 (18).JPG		3/10/2009 5:11 PM
Pampalona_07.06.09 (19).JPG		3/10/2009 5:11 PM

596 items | 596 items selected |

PowerRename menu

After selecting some files in Windows File Explorer, right-clicking and selecting *PowerRename* (which will appear only when enabled in PowerToys), the PowerRename menu will appear. The number of items (files) you've selected will be displayed, along with search and replace values, a list of options, and a preview window displaying results of the search and replace values you've entered.


PowerRename

Enter the criteria below to rename the items

Search for: power

Replace with: super

Options

- [ ] Use Regular Expressions
- [ ] Match All Occurrences
- [ ] Case Sensitive
- [ ] Enumerate Items
- [ ] Item Name Only
- [ ] Item Extension Only
- [ ] Exclude Folders
- [ ] Exclude Files
- [ ] Exclude Subfolder Items
- [ ] Make Uppercase
- [ ] Make Lowercase
- [ ] Make Titlecase

Preview

Original	Renamed
powerrename-menu.png	superrename-menu.png
powershell.png	supershell.png
powertoys.png	supertoys.png
powertoys-banner.png	supertoys-banner.png

Items Selected: 5 | Renaming: 5

Search for

Enter text or a regular expression to find the files in your selection that contain the criteria matching your entry. You will see the matching items in the *Preview* window.

Replace with

Enter text to replace the *Search for* value entered previously that match you're selected files. You can view the original file name and renamed file in the *Preview* window.

Options - Use Regular Expressions

If checked, the Search value will be interpreted as a regular expression (regex). The Replace value can also contain regex variables (see examples below). If not checked, the Search value will be interpreted as plain text to be replaced with the text in the Replace field.

For more information regarding the `Use Boost library` option in the settings menu for extended regex functionalities, see the regular expressions section.

Options - Case Sensitive

If checked, the text specified in the Search field will only match text in the items if the text is the same case. Case matching will be insensitive (not recognizing a difference between upper and lowercase letters) by default.

Options - Match All Occurrences

If checked, all matches of text in the Search field will be replaced with the Replace text. Otherwise, only the first instance of the Search for text in the file name will be replaced (left to right).

For example, given the file name: `powertoys-powerrename.txt`:

- Search for: `power`
- Rename with: `super`

The value of the renamed file would result in:

- Match All Occurrences (unchecked): `supertoys-powerrename.txt`
- Match All Occurrences (checked): `supertoys-superrename.txt`

Options - Exclude Files

Files will not be included in the operation. Only folders will be included.

Options - Exclude Folders

Folders will not be included in the operation. Only files will be included.

Options - Exclude Subfolder Items

Items within folders will not be included in the operation. By default, all subfolder items are included.

Options - Enumerate Items

Appends a numeric suffix to file names that were modified in the operation. For example: `foo.jpg` -> `foo (1).jpg`

Options - Item Name Only

Only the file name portion (not the file extension) is modified by the operation. For example: `txt.txt` -> `NewName.txt`

Options - Item Extension Only

Only the file extension portion (not the file name) is modified by the operation. For example: `txt.txt` -> `txt.NewExtension`

Replace using file creation date and time

The creation date and time attributes of a file can be used in the *Replace with* text by entering a variable pattern according to the table below.

VARIABLE PATTERN	EXPLANATION
`$YYYY`	Year represented by a full four or five digits, depending on the calendar used.
`$YY`	Year represented only by the last two digits. A leading zero is added for single-digit years.
`$Y`	Year represented only by the last digit.
`$MMMM`	Name of the month
`$MMM`	Abbreviated name of the month
`$MM`	Month as digits with leading zeros for single-digit months.
`$M`	Month as digits without leading zeros for single-digit months.
`$DDDD`	Name of the day of the week
`$DDD`	Abbreviated name of the day of the week
`$DD`	Day of the month as digits with leading zeros for single-digit days.
`$D`	Day of the month as digits without leading zeros for single-digit days.

VARIABLE PATTERN	EXPLANATION
`$hh`	Hours with leading zeros for single-digit hours
`$h`	Hours without leading zeros for single-digit hours
`$mm`	Minutes with leading zeros for single-digit minutes.
`$m`	Minutes without leading zeros for single-digit minutes.
`$ss`	Seconds with leading zeros for single-digit seconds.
`$s`	Seconds without leading zeros for single-digit seconds.
`$fff`	Milliseconds represented by full three digits.
`$ff`	Milliseconds represented only by the first two digits.
`$f`	Milliseconds represented only by the first digit.

For example, given the file names:

- `powertoys.png`, created on 11/02/2020
- `powertoys-menu.png`, created on 11/03/2020

Enter the criteria to rename the items:

- Search for: `powertoys`
- Rename with: `$MMM-$DD-$YY-powertoys`

The value of the renamed file would result in:

- `Nov-02-20-powertoys.png`
- `Nov-03-20-powertoys-menu.png`

Regular Expressions

For most use cases, a simple search and replace is sufficient. There may be occasions, however, in which complicated renaming tasks come along that require more control. Regular Expressions can help.

Regular Expressions define a search pattern for text. They can be used to search, edit and manipulate text. The pattern defined by the regular expression may match once, several times, or not at all for a given string. PowerRename uses the ECMAScript grammar, which is common amongst modern programming languages.

To enable regular expressions, check the "Use Regular Expressions" checkbox.

Note: You will likely want to check "Match All Occurrences" while using regular expressions.

To use the Boost library instead of the standard library, check the `Use Boost library` option in the PowerToys settings. It enables extended features, like lookbehind, which are not supported by the standard library.

Examples of regular expressions

Simple matching examples

SEARCH FOR	DESCRIPTION
`^`	Match the beginning of the filename
`$`	Match the end of the filename
`.*`	Match all the text in the name
`^foo`	Match text that begins with "foo"
`bar$`	Match text that ends with "bar"
`^foo.*bar$`	Match text that begins with "foo" and ends with "bar"
`.+?(?=bar)`	Match everything up to "bar"
`foo[\s\S]*bar`	Match everything between "foo" and "bar"

Matching and variable examples

When using the variables, the "Match All Occurrences" option must be enabled.

SEARCH FOR	REPLACE WITH	DESCRIPTION
`(.*).png`	`foo_$1.png`	Prepends "foo_" to the existing file name
`(.*).png`	`$1_foo.png`	Appends "_foo" to the existing file name
`(.*)`	`$1.txt`	Appends ".txt" extension to existing file name
`(^\w+\.$)\|(^\w+$)`	`$2.txt`	Appends ".txt" extension to existing file name only if it does not have an extension
`(\d\d)-(\d\d)-(\d\d\d\d)`	`$3-$2-$1`	Move numbers in the filename: "29-03-2020" becomes "2020-03-29"

Additional resources for learning regular expressions

There are great examples/cheat sheets available online to help you

Regex tutorial — A quick cheatsheet by examples

ECMAScript Regular Expressions Tutorial

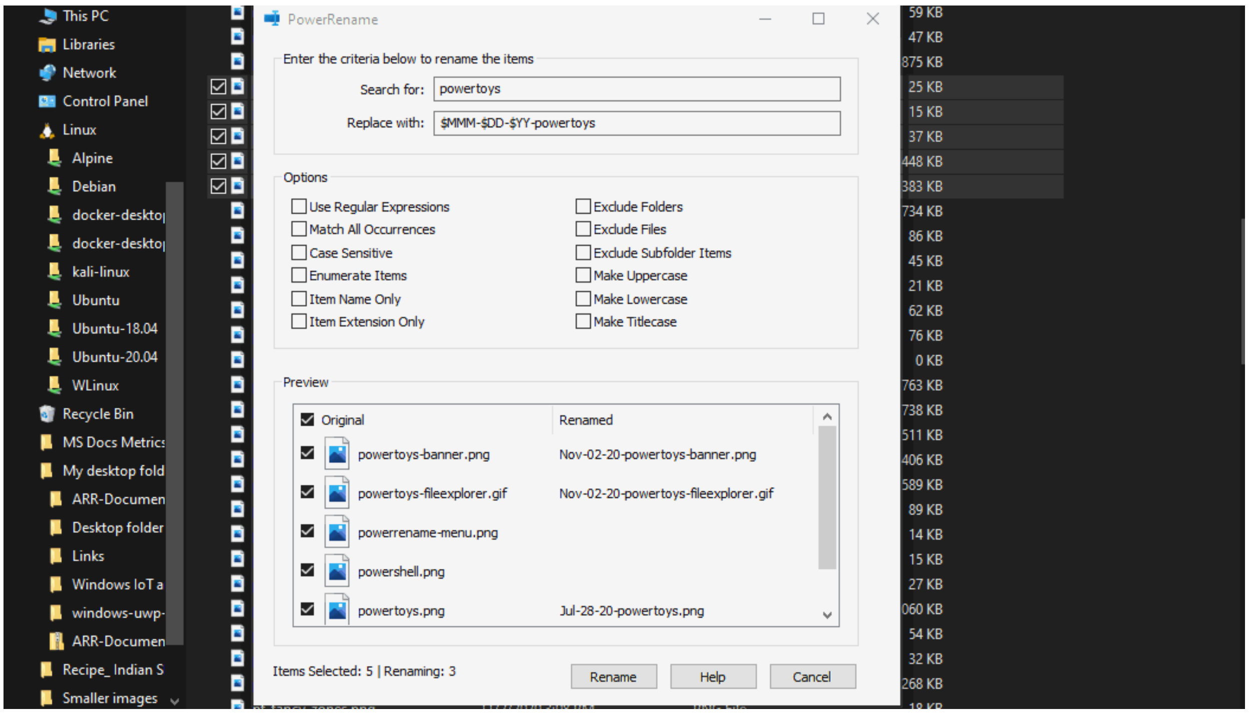
File List Filters

Filters can be used in PowerRename to narrow the results of the rename. Use the *Preview* window to check expected results. Select the column headers to switch between filters.

- **Original**, the first column in the *Preview* window cycles between:
  - Checked: The file is selected be renamed.
  - Unchecked: The file is not selected to be renamed (even though it fits the value entered in the search

criteria).

- **Renamed**, the second column in the *Preview* windows can be toggled.
  - The default preview will show all selected files, with only files matching the *Search for* criteria displaying the updated rename value.
  - Selecting the *Renamed* header will toggle the preview to only display files that will be renamed. Other selected files from your original selection will not be visible.



Enter the criteria below to rename the items

Search for: powertoys

Replace with: $MMM-$DD-$YY-powertoys

Options: Use Regular Expressions, Match All Occurrences, Case Sensitive, Enumerate Items, Item Name Only, Item Extension Only, Exclude Folders, Exclude Files, Exclude Subfolder Items, Make Uppercase, Make Lowercase, Make Titlecase

Preview

Original	Renamed
powertoys-banner.png	Nov-02-20-powertoys-banner.png
powertoys-fileexplorer.gif	Nov-02-20-powertoys-fileexplorer.gif
powerrename-menu.png	
powershell.png	
powertoys.png	Jul-28-20-powertoys.png

Items Selected: 5 | Renaming: 3

PowerToys Run utility

6/30/2021 • 4 minutes to read • Edit Online

PowerToys Run is a quick launcher for power users that contains some additional features without sacrificing performance. It is open source and modular for additional plugins.

To use PowerToys Run, select `Alt+Space` and start typing!

If that shortcut isn't what you like, don't worry, it is fully configurable in the settings.

Requirements

- Windows 10 version 1903 or higher
- After installing, PowerToys must be enabled and running in the background for this utility to work

Features

PowerToys Run features include:

- Search for applications, folders, or files
- Search for running processes (previously known as WindowWalker)
- Clickable buttons with keyboard shortcuts (such as *Open as administrator* or *Open containing folder*)
- Invoke Shell Plugin using `>` (for example, `> Shell:startup` will open the Windows startup folder)
- Do a simple calculation using calculator

Settings

The following Run options are available in the PowerToys settings menu.

SETTINGS	ACTION
Open PowerToys Run	Define the keyboard shortcut to open/hide PowerToys Run
Ignore shortcuts in Fullscreen mode	When in full-screen (F11), Run won't be engaged with the shortcut
Maximum number of results	Maximum number of results shown without scrolling
Clear the previous query on launch	When launched, previous searches will not be highlighted
Disable drive detection warning	The warning, if all of your drives aren't indexed, will no longer be visible.

Keyboard shortcuts

SHORTCUTS	ACTION
Alt+Space	Open or hide PowerToys Run
Esc	Hide PowerToys Run
Ctrl+Shift+Enter	(Only applicable to applications) Open the selected application as administrator
Ctrl+Shift+E	(Only applicable to applications and files) Open containing folder in File Explorer
Ctrl+C	(Only applicable to folders and files) Copy path location
Tab	Navigate through the search result and context menu buttons

Action keys

These default activation phrases will force PowerToys run into only targeted plugins.

ACTION KEY	ACTION
`=`	Calculator only. Example `=2+2`.
`?`	File searching only. Example `?road` to find `roadmap.txt`.
`.`	Installed programs only. Example `.code` to get Visual Studio Code. See Program parameters for options on adding parameters to a program's startup.
`//`	URLs only. Example `//` to launch your default browser, or `//docs.microsoft.com` to have your default browser go to https://docs.microsoft.com.

ACTION KEY	ACTION
`<`	Running processes only. Example `<outlook` to find all processes that contain outlook.
`>`	Shell command only. Example `>ping localhost` to do a ping query.
`:`	Registry keys only. Example `:hkcu` to search for the HKEY_CURRENT_USER registry key.
`!`	Windows services only. Example `!alg` to search for the Application Layer Gateway service to be started or stopped.
`{`	Visual Studio Code previously opened workspaces, remote machines (SSH or Codespaces) and containers. Example `{powertoys` to search for workspaces that contain 'powertoys' in their paths. This plugin is off by default.
`%%`	Unit converter only. Example `%% 10 ft in m` to calculate the number of meters in 10 feet.
`$`	Windows settings only. Example `$ Add/Remove Programs` to launch the Windows settings menu for managing installed programs. To list all settings of an area category, type `:` after the category name. Ex: `$ Device:` to view all available Device settings.

System commands

PowerToys Run enables a set of system level actions that can be executed.

ACTION KEY	ACTION
`Shutdown`	Shuts down the computer
`Restart`	Restarts the computer
`Sign Out`	Signs current user out
`Lock`	Locks the computer
`Sleep`	Sleeps the computer
`Hibernate`	Hibernates the computer
`Empty Recycle Bin`	Empties the recycle bin

Plugin manager

The PowerToys Run settings menu includes a plugin manager that allows you to enable/disable the various plugins currently available. By selecting and expanding the sections, you can customize the activation phrases used by each plugin. In addition, you can select whether a plugin appears in global results, as well as set additional plugin options where available.

Program parameters

The PowerToys Run program plugin allows for program arguments to be added when launching an application. The program arguments must follow the expected format as defined by the program's command line interface.

For example, when launching Visual Studio Code, you can specify the folder to be opened with:

```
Visual Studio Code -- C:\myFolder
```

Visual Studio Code also supports a set of command line parameters, which can be utilized with their corresponding arguments in PowerToys Run to, for instance, view the difference between files:

```
Visual Studio Code -d C:\foo.txt C:\bar.txt
```

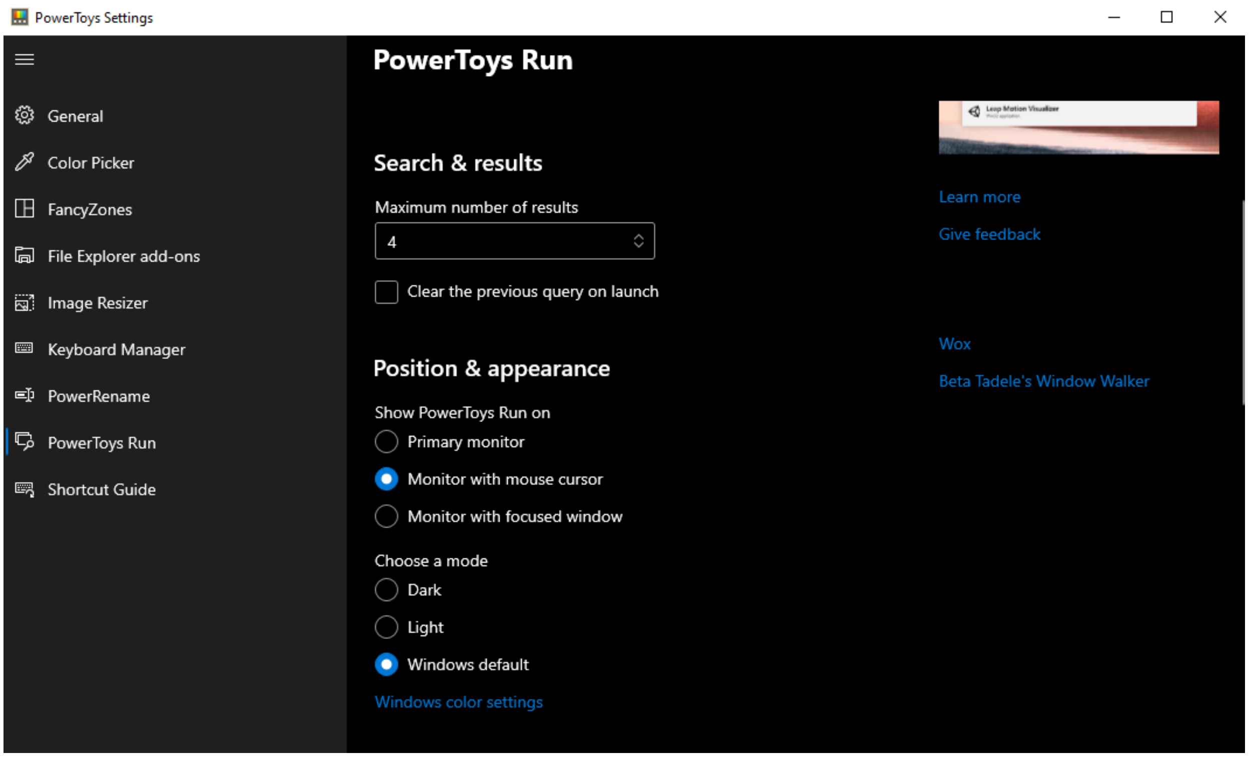
If the program plugin's option "Include in global result" is not selected, be sure to include the activation phrase, `.` by default, to invoke the plugin's behavior:

```
.Visual Studio Code -- C:\myFolder
```

Monitor Positioning

If multiple monitors are in use, PowerToys Run can be launched on the desired monitor by configuring the appropriate launch behavior in the Settings menu. Options include opening on:

- Primary monitor
- Monitor with mouse cursor
- Monitor with focused window



PowerToys Run

Search & results

Maximum number of results: 4

☐ Clear the previous query on launch

Position & appearance

Show PowerToys Run on

Primary monitor [✓ Monitor with mouse cursor] Monitor with focused window

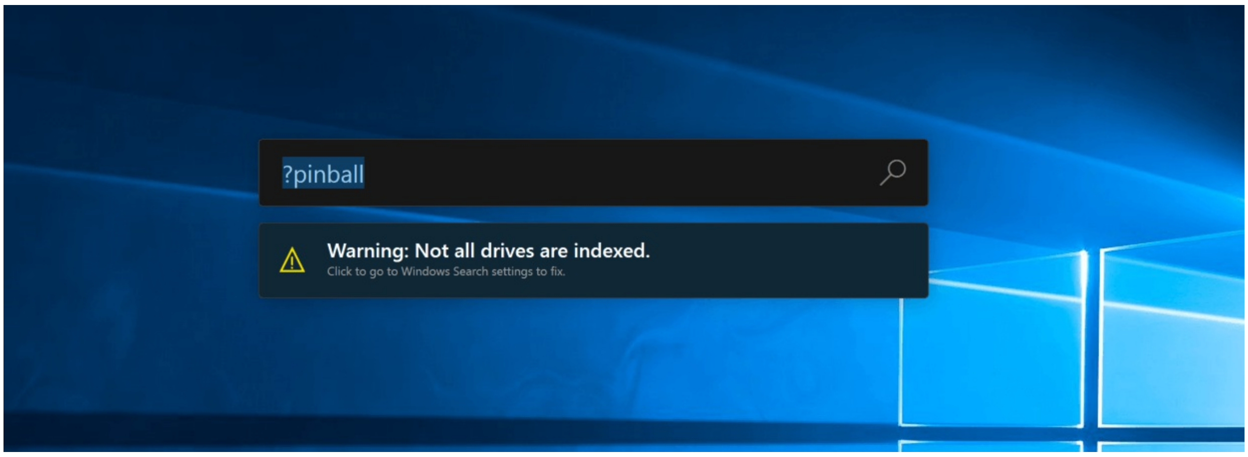
Choose a mode

Dark Light [✓ Windows default]

Windows color settings

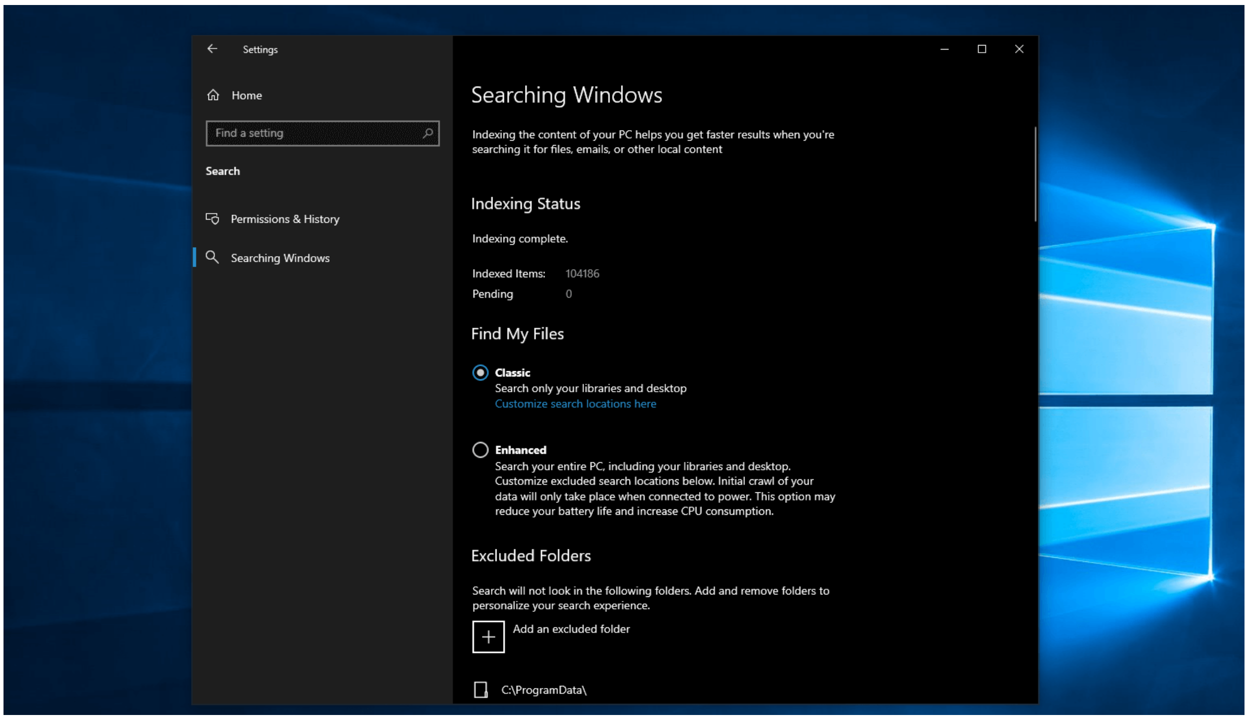
Windows Search settings

If the Windows Search plugin is not set to cover all drives, you will receive the following warning:


?pinball

Warning: Not all drives are indexed.
Click to go to Windows Search settings to fix.

You can turn off the warning in the PowerToys Run plugin manager options for Windows Search, or select the warning to expand which drives are being indexed. After selecting the warning, the Windows 10 settings "Searching Windows" options menu will open.



Searching Windows

Indexing the content of your PC helps you get faster results when you're searching it for files, emails, or other local content

Indexing Status

Indexing complete.

Indexed Items: 104186
Pending 0

Find My Files

[✓ Classic]
Search only your libraries and desktop
Customize search locations here

Enhanced
Search your entire PC, including your libraries and desktop. Customize excluded search locations below. Initial crawl of your data will only take place when connected to power. This option may reduce your battery life and increase CPU consumption.

Excluded Folders

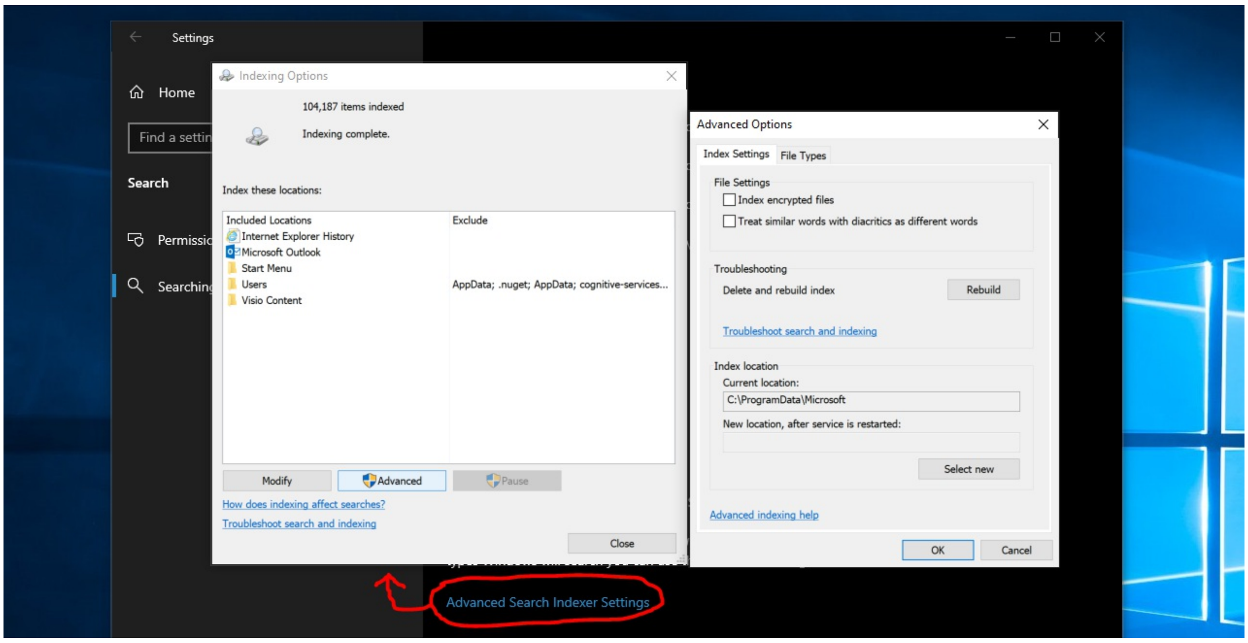
Search will not look in the following folders. Add and remove folders to personalize your search experience.

Add an excluded folder

C:\ProgramData\

In this "Searching Windows" menu, you can:

- Select "Enhanced" mode to enable indexing across all of the drives on your Windows 10 machine.
- Specify folder paths to exclude.
- Select the "Advanced Search Indexer Settings" (near the bottom of the menu options) to set advanced index settings, add or remove search locations, index encrypted files, etc.



Indexing Options

104,187 items indexed

Indexing complete.

Index these locations:

Included Locations	Exclude
Internet Explorer History	
Microsoft Outlook	
Start Menu	
Users	AppData; .nuget; AppData; cognitive-services...
Visio Content	

Modify | Advanced | Pause

How does indexing affect searches?

Troubleshoot search and indexing

Close

Advanced Options

Index Settings | File Types

File Settings

☐ Index encrypted files

☐ Treat similar words with diacritics as different words

Troubleshooting

Delete and rebuild index — Rebuild

Troubleshoot search and indexing

Index location

Current location:

C:\ProgramData\Microsoft

New location, after service is restarted:

Select new

Advanced indexing help

OK | Cancel

Known issues

For a list of all known issues and suggestions, see the PowerToys product repo issues on GitHub.

Attribution

- Wox
- Beta Tadele's Window Walker

Windows key shortcut guide

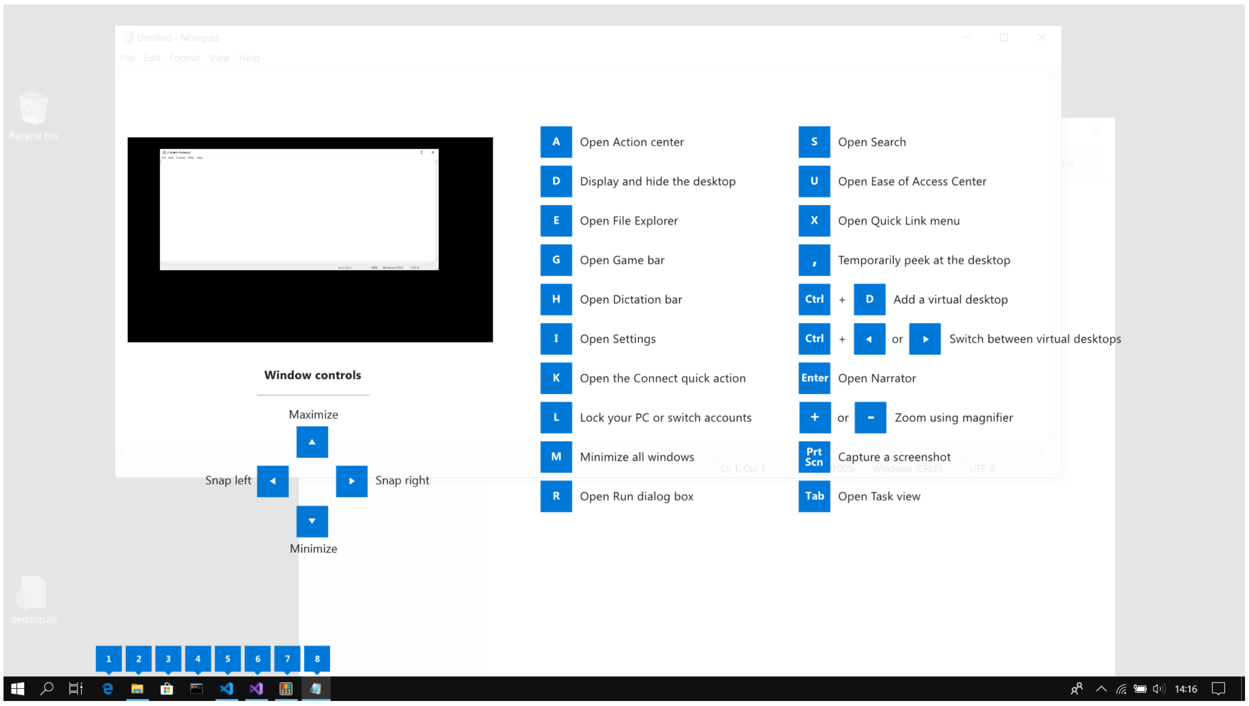
6/30/2021 • 2 minutes to read • Edit Online

This guide uses PowerToys to display common keyboard shortcuts that use the Windows ⊞ key.

Usage

Open the shortcut guide with the shortcut key combination: `⊞ Win + ?` *(this may require using the Shift key)*. An overlay will appear showing keyboard shortcuts that use the Windows ⊞ key, including:

- common Windows shortcuts,
- shortcuts for changing the position of the active window,
- taskbar shortcuts.



A	Open Action center
D	Display and hide the desktop
E	Open File Explorer
G	Open Game bar
H	Open Dictation bar
I	Open Settings
K	Open the Connect quick action
L	Lock your PC or switch accounts
M	Minimize all windows
R	Open Run dialog box
S	Open Search
U	Open Ease of Access Center
X	Open Quick Link menu
,	Temporarily peek at the desktop
Ctrl + D	Add a virtual desktop
Ctrl + ◂ or ▸	Switch between virtual desktops
Enter	Open Narrator
+ or -	Zoom using magnifier
Prt Scn	Capture a screenshot
Tab	Open Task view

Keyboard shortcuts using the Windows key (`⊞ Win`) can be used while the guide is displayed. The result of those shortcuts (active window moved, arrow shortcut behavior changes, etc) will be displayed in the guide.

Pressing the shortcut key combination again will dismiss the overlay.

Tapping the Windows key (`⊞ Win`) will display the Windows Start menu.

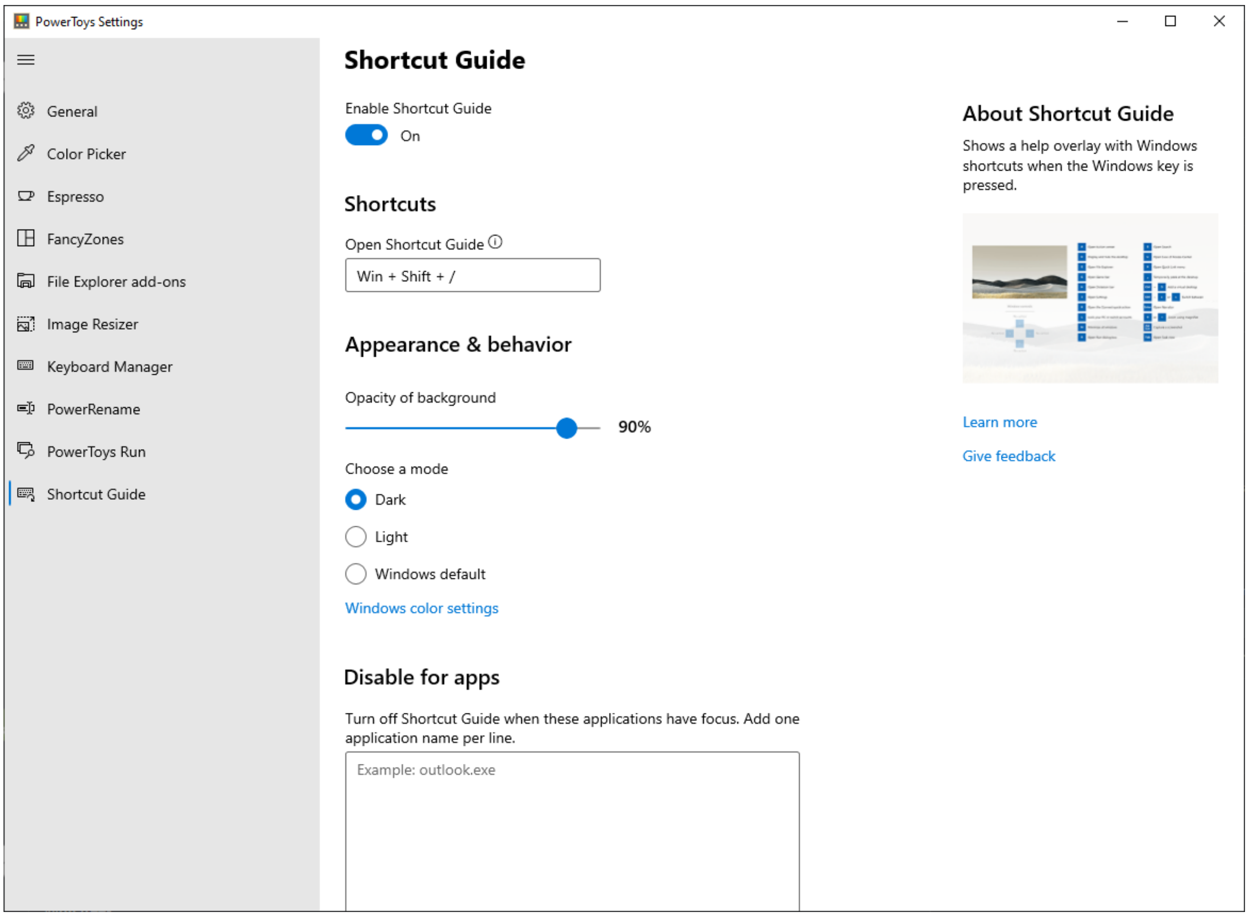
> **IMPORTANT**
>
> The PowerToys app must be running and Shortcut Guide must be enabled in the PowerToys settings for this feature to be used.

Settings

These configurations can be edited from the PowerToys Settings:

- **Open Shortcut Guide**: The shortcut used to launch the shortcut guide.
- **Opacity of background**: This slider bar controls the opacity of the Shortcut Guide overlay. *(The degree to*

which you can see through it).


PowerToys Settings

General
Color Picker
Espresso
FancyZones
File Explorer add-ons
Image Resizer
Keyboard Manager
PowerRename
PowerToys Run
Shortcut Guide

Shortcut Guide

Enable Shortcut Guide
On

Shortcuts

Open Shortcut Guide
Win + Shift + /

Appearance & behavior

Opacity of background
90%

Choose a mode
[✓ Dark]
Light
Windows default
Windows color settings

Disable for apps

Turn off Shortcut Guide when these applications have focus. Add one application name per line.
Example: outlook.exe

About Shortcut Guide

Shows a help overlay with Windows shortcuts when the Windows key is pressed.

Learn more
Give feedback

Video Conference Mute (Preview)

6/1/2021 • 2 minutes to read • Edit Online

IMPORTANT

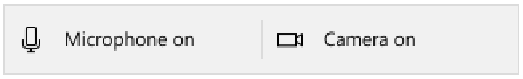
This is a preview feature and only included in the pre-release version of PowerToys. Running this pre-release requires Windows 10 version 1903 (build 18362) or later.

Quickly mute your microphone (audio) and turn off your camera (video) while on a conference call with a single keystroke, regardless of what application has focus on your computer.

Usage

The default settings to use Video Conference Mute are:

- ⊞ `Win+N` to toggle both Audio and Video at the same time
- ⊞ `Win+Shift+A` to toggle microphone
- ⊞ `Win+Shift+O` to toggle video



When using the microphone and/or camera toggle shortcut keys, you will see a small toolbar window display letting you know whether the your Microphone and Camera are set to on, off, or not in use. You can set the position of this toolbar in the Video Conference Mute tab of PowerToys settings.

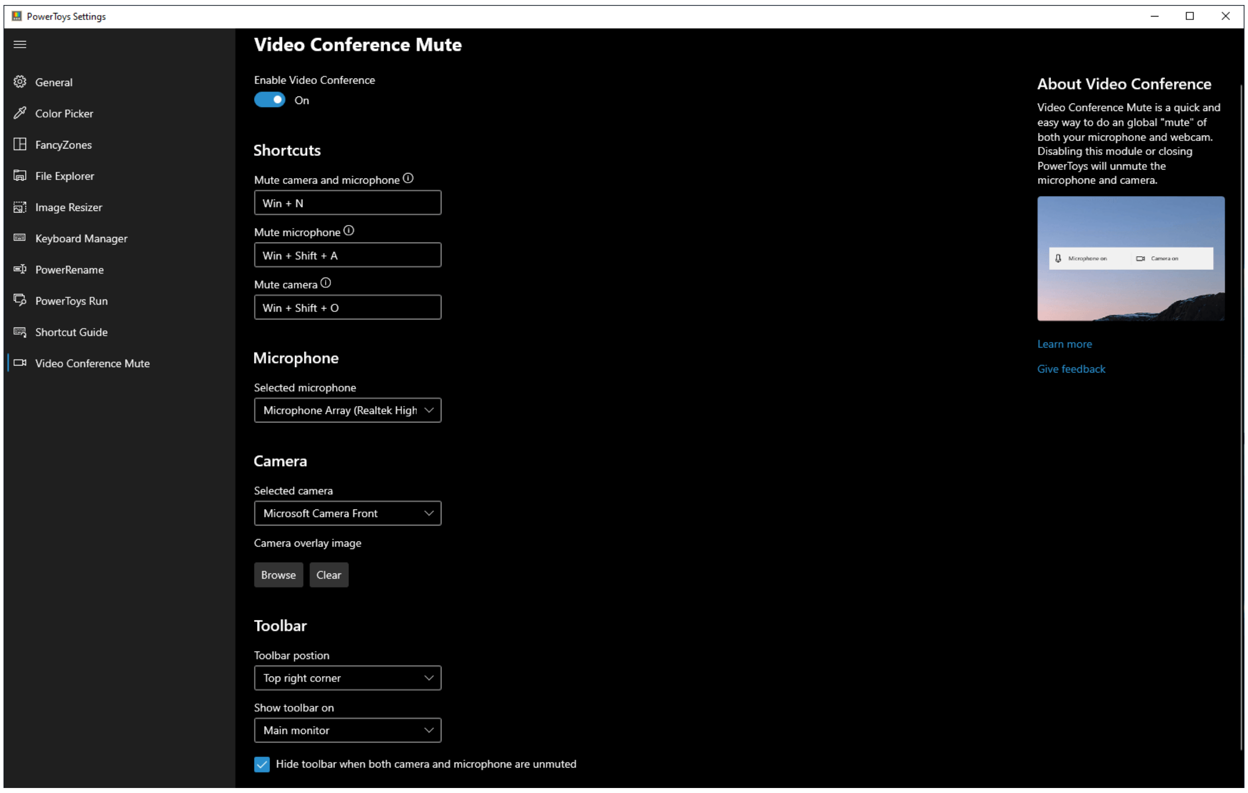
NOTE

Remember that you must have the pre-release/experimental version of PowerToys installed and running, with the Video Conference Mute feature enabled in PowerToys settings in order to use this utility.

Settings

The Video Conference Mute tab in PowerToys settings provides the following options:

- **Shortcuts**: Change the shortcut key used to mute your microphone, camera, or both combined.
- **Selected microphone**: Select the microphone on your machine that this utility will target.
- **Selected camera**: Select the camera on your machine that this utility will target.
- **Camera overlay image**: Select an image to that will be used as a placeholder when your camera is turned off. (By default, a black screen will appear when your camera is turned off with this utility).
- **Toolbar**: Determine the position where the *Microphone On, Camera On* toolbar displays when toggled (top right corner by default).
- **Show toolbar on**: Select whether you prefer the toolbar to be displayed on the main monitor only (default) or on all monitors.
- **Hide toolbar when both camera and microphone are unmuted**: A checkbox is available to toggle this option.


PowerToys Settings

General
Color Picker
FancyZones
File Explorer
Image Resizer
Keyboard Manager
PowerRename
PowerToys Run
Shortcut Guide
Video Conference Mute

Video Conference Mute

Enable Video Conference: On

Shortcuts

Mute camera and microphone: Win + N

Mute microphone: Win + Shift + A

Mute camera: Win + Shift + O

Microphone

Selected microphone: Microphone Array (Realtek High

Camera

Selected camera: Microsoft Camera Front

Camera overlay image: Browse | Clear

Toolbar

Toolbar postion: Top right corner

Show toolbar on: Main monitor

☑ Hide toolbar when both camera and microphone are unmuted

About Video Conference

Video Conference Mute is a quick and easy way to do an global "mute" of both your microphone and webcam. Disabling this module or closing PowerToys will unmute the microphone and camera.

Learn more

Give feedback

How does this work under the hood

Application interact with audio and video in different ways.

If a camera stops working, the application using it tends not to recover until the API does a full reset. To toggle the global privacy camera on and off while using the camera in an application, typically it will crash and not recover.

So, how does PowerToys handle this so you can keep streaming?

- **Audio**: PowerToys uses the global microphone mute API in Windows. Apps should recover when this is toggled on and off.
- **Video**: PowerToys has a virtual driver for the camera. The video is routed through the driver and back to the application. Selecting the Video Conference Mute shortcut key stops video from streaming, but the application still thinks it is receiving video, the video is just replaced with black or the image placeholder you've saved in the settings.

Debug the camera driver

To debug the camera driver, look in this folder on your machine:

```
C:\Windows\ServiceProfiles\LocalService\AppData\Local\Temp\PowerToysVideoConference.log
```

You could also create an empty `PowerToysVideoConferenceVerbose.flag` in the same directory to enable verbose logging mode in the driver.

Known issues

To view all of the known issues currently open on the Video Conference Mute utility, see PowerToys tracking issue #6246 on GitHub. The PowerToys development team and contributor community is actively working toward resolving these issues and plans to keep the utility in pre-release until essential issues are resolved.