



eBOOK Preview

Getting Started with the SolarWinds Orion API

Chapter 2: Modern Problems Require Modern PowerShell for the Modern Orion Platform

Note: This is a special preview from a larger eBook.

For years now, the SolarWinds® Orion® API (Application Programming Interface) has been accessible to run via PowerShell on Windows machines. Originally through the Snap-In (SwisSnapIn) and later through the Module (SwisPowerShell). Although not technically deprecated, the Windows PowerShell 5.1 engine is being phased out for replacement with PowerShell 7 (formerly PowerShell Core).

Our approach to working with the functionality needs to change with the times.

The (Extremely) Quick Intro to PowerShell 7

The most succinct way to describe how PowerShell 7 is different than earlier Windows PowerShell versions is in the name: it lacks “Windows.” PowerShell 7 is a cross-platform management framework that runs on Windows, macOS, Linux, x86, x64, ARM, and ARM-64 processors. This allows the same scripts to run anywhere and everywhere. It’s a bold move, and a move in the right direction, but some things have been left behind.

WHERE IS POWERSHELL ISE?

To put it bluntly, the Integrated Scripting Environment (ISE) is no longer maintained. Like Windows PowerShell 5.1, the ISE is only capable of running on Windows machines. And while all modern versions of Windows can support it, its lack of cross-platform functionality was counterproductive given the multi-operating system support for PowerShell 7. For true multi-platform support, something new needed to be used.

ENTER VISUAL STUDIO CODE

The de facto replacement for the PowerShell ISE is Visual Studio Code (VS Code), and it’s for so much more than just PowerShell. VS Code supports dozens and dozens of programming languages, has thousands of extensions to make working easier and more efficient, and has a modern aesthetic to boot. Don’t be fooled by the name—Visual Studio Code is different from Visual Studio.

VS Code is supported on most major platforms x86, x64, ARM, and ARM-64 as well as most operating systems (Windows, Linux, macOS). This makes it an excellent tool if you’re just getting started with scripting as the experience will be near identical for all your collaborators regardless of the OS or platform.

WHY AREN'T POWERSHELL 7 AND VS CODE BUNDLED TOGETHER?

There is a myriad of reasons, but PowerShell is a command line interpreter (CLI) and needs no Graphical User Interface (GUI) while VS Code is a visual editor and needs a GUI. If you have a Linux machine that doesn't have Gnome, KDE, or another X Windows system installed, you can still run PowerShell. Just install it, and it's available to use from the terminal.

THE KEY TO USING A TOOL IS FLEXIBILITY

Microsoft knew when they started moving PowerShell (and any associated editor) to an open-source solution, consumers would want to use the tools in the best way possible for their own situations. This versatility was part of the original mandate for designing VS Code. If you want to write a script in Perl, you needed to be able to do it. If you wanted to work in Python, you needed that capability. If C# was your forte, then so be it, the tool needed to support that. If all you touch is HTML and CSS, then the solution needs to work for you. Nothing differentiates Visual Studio Code from the traditional Visual Studio products more than its ease of extensibility and configuration for the casual scripter/programmer/web dev.

Part of embracing this flexibility is using the tools that work best for you. This widely depends on if you have a preferred Integrated Development Environment/Integrated Scripting Environment (IDE/ISE) or prefer to work in other ways. VS Code is not the only choice available to you. Many successful scripters work in nothing but a plain text editor like nano, notepad, emacs, vim, or Notepad++. If you prefer Sublime, Eclipse, the full Visual Studio product, or something else, please feel free to use that. Each choice comes with their own benefits and deterrents.

For the sake of continuity, this book uses Visual Studio Code (referred to as "VS Code") for all screenshots. All code samples provided in this book will work assuming the following:

- You have installed PowerShell version 7 or higher.
- You have installed the SolarWinds Orion PowerShell module version 3 or higher.

If you elect to work with another editor (or multiple editors), please work in whatever way makes you most comfortable.

CO-HOSTING WITH ORION PLATFORM PRODUCTS

There is no rule that says you cannot install PowerShell 7 and VS Code on the server running your Orion Platform products, but my go-to has always been to run them adjacent to those machines. It doesn't even need to be a server; it just needs to be a computer where you have some type of GUI. My preference has always been to have a UTIL server on the same subnet as my Orion servers. This way, I can easily do work on the UTIL server (including rebooting or restarting local services after updates), and I won't disturb my monitoring solution from doing its job.

The following steps show how to set up PowerShell 7 and VS Code on a separate machine than that running the SolarWinds Orion Platform products. The steps are outlined in detail on the Microsoft website [[PowerShell](#), [VS Code](#)] to install both PowerShell 7 and VS Code. Please refer to those documents if you have problems.

Again, if you prefer to not use Visual Studio Code, you can install the SolarWinds Orion PowerShell module after completing your PowerShell 7 installation. Just follow the instructions under **Installing the SolarWinds Orion PowerShell Module** from a regular PowerShell session.

Step-by-Step Installation Instructions

In each of the following sections, we'll cover how to install PowerShell 7 and Visual Studio Code on Windows, macOS, and various flavors of Linux. This should cover nearly all situations, but when in doubt, refer to the official Microsoft Documentation ([PowerShell](#)/[VS Code](#) links here).

Once both PowerShell and Visual Studio Code are installed, the final steps (**Initial Launch of VS Code, Additional Recommended Extensions, Installing the SolarWinds Orion PowerShell Module, and Validating Our Work**) are identical.

REQUIRED PERMISSIONS

On Windows computers, you'll need to be a member with sufficient rights to install software. This is typically either the local Administrators group or Power Users (if running on a very old version of Windows).

On macOS, you'll need to be a member of the Administrators group on the computer. During the installs you may be re-prompted for your password to authorize the changes.

On Linux, you'll need to be in the `sudoers` group. Membership and maintenance of this group is different depending on your Linux distribution. Please see the documentation on your specific distribution for more details on `sudo`.

Setup Steps for Windows 11

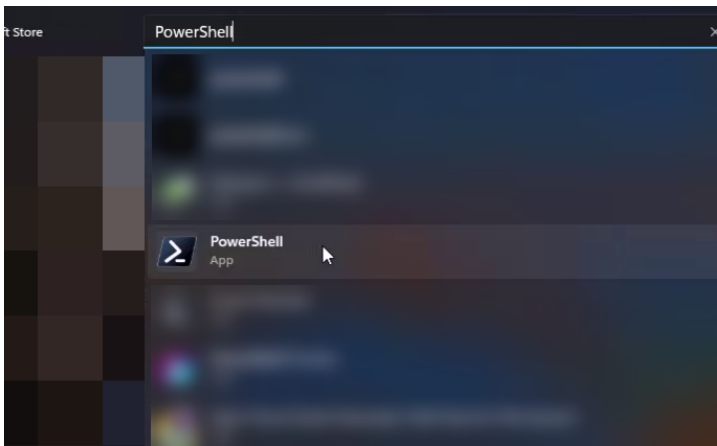
For Windows 11, both the most recent versions of PowerShell and Visual Studio Code are available in the Microsoft Store.

BASE SYSTEM DESCRIPTION

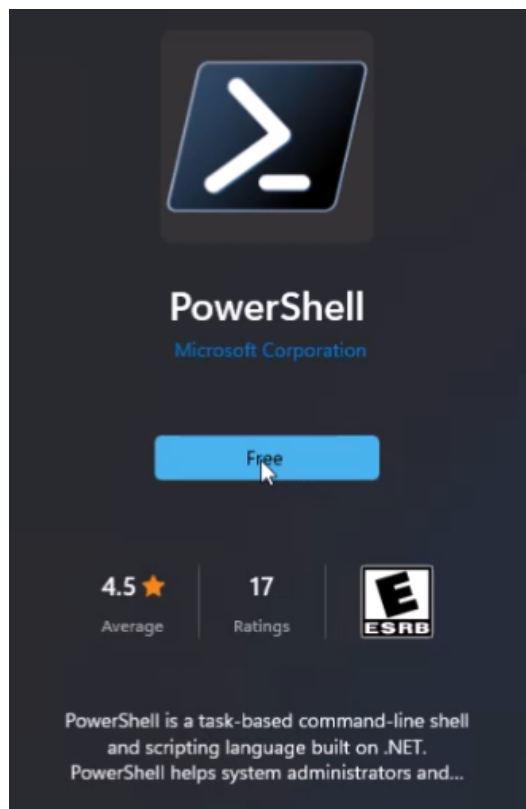
We'll be installing PowerShell and Visual Studio Code on a simplified workstation running only the out of the box applications. No additional software was installed, but Windows Updates were completed prior to installation.

INSTALL POWERSHELL 7

Open the Microsoft Store and search for "PowerShell."



Click on "Free" button to install PowerShell.



After downloading, click "Open" and type `$PSVersionTable` followed by Enter. Your returned version should be at least 7.1.

```
C:\Program Files\WindowsApps\Microsoft.PowerShell_7.1.4.0_x64_8wekyb3d8bbwe\pwsh.exe
PowerShell 7.1.4
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Windows\System32> $PSVersionTable

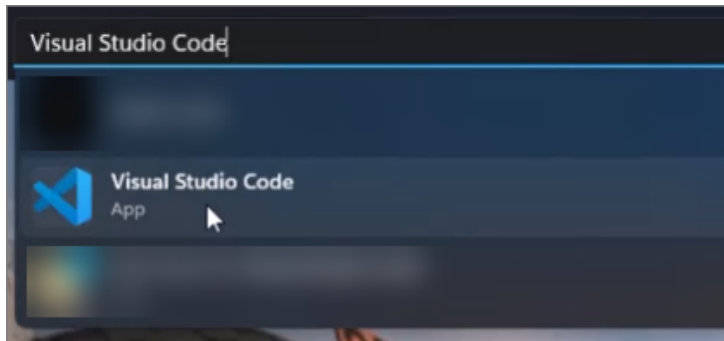
Name                           Value
----                           -
PSVersion                      7.1.4
PSEdition                     Core
GitCommitId                   7.1.4
OS                             Microsoft Windows 10.0.22000
Platform                      Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
PSRemotingProtocolVersion      2.3
SerializationVersion          1.1.0.1
WSManStackVersion              3.0

PS C:\Windows\System32> .
```

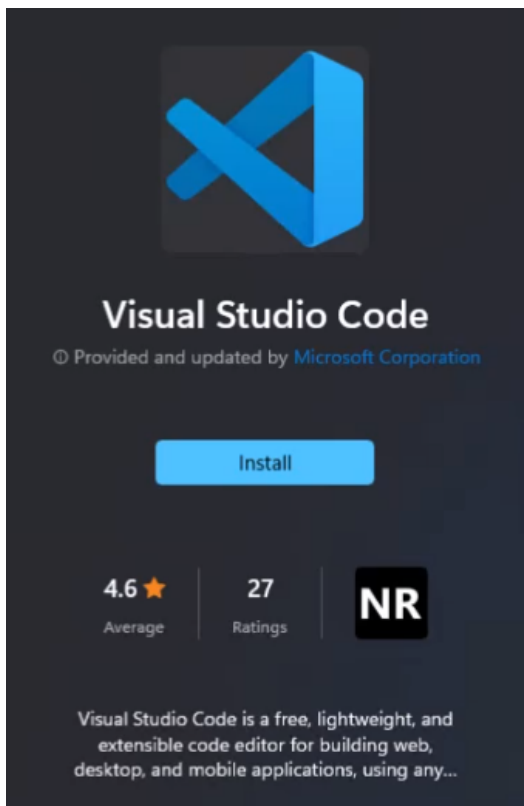
If the version is good, you are free to close this window and move on.

INSTALL VISUAL STUDIO CODE

As with PowerShell 7.x, you can install VS Code from the Microsoft Store. Begin by searching for "Visual Studio Code" and selecting it from the list.



Click "Install" to begin the installation.



Once you confirm that Visual Studio Code is installed on your system, you can continue to the Initial Launch of VS Code later in this document.

Setup Steps for Windows Server

The installation is nearly identical for all versions of Windows (server and workstation) with the exception that 32-bit machines should run the x86 version of PowerShell and VS Code and 64-bit machines should use the x64 versions.


BASE SYSTEM DESCRIPTION

In my example, my Windows machine is running Windows Server 2019 (Standard) with the Desktop Experience (non-Core) install. It's installed from an original ISO and the only thing I have done is disabled the Server Manager from auto starting. It's pretty much a naked server.

I first ran through all the Windows Updates to make sure I had the latest patches and then it was time to install PowerShell 7.

INSTALL POWERSHELL 7

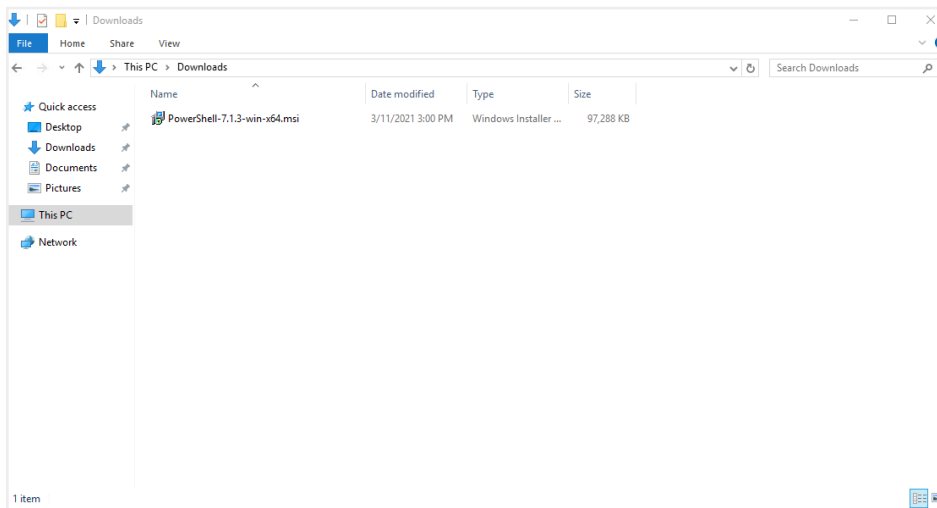
The installer for Windows is an MSI file which can be downloaded from the [PowerShell GitHub releases](#). Just locate the latest stable version, find the MSI file for x64 and download it. (If you are running a 32-bit version of Windows, you choose the x86 version.)



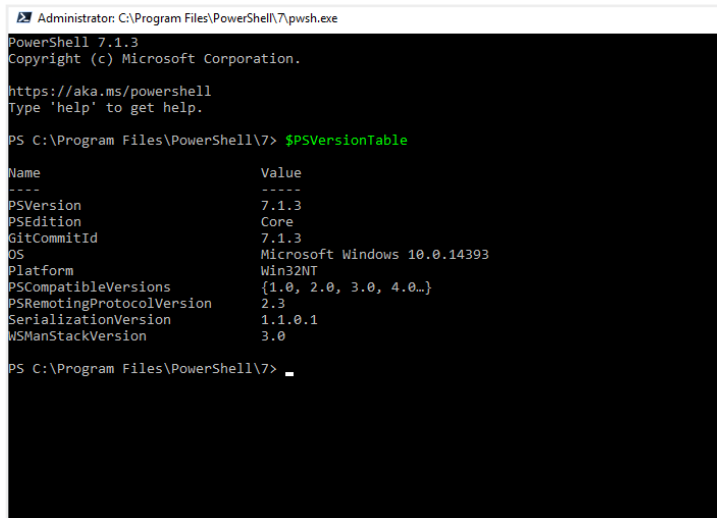
A screenshot of the PowerShell 7.1.3 releases page on GitHub. The table lists various download options for different architectures and formats.

File Name	Size
PowerShell-7.1.3-win-arm64.zip	61.2 MB
PowerShell-7.1.3-win-fxdependent.zip	21.4 MB
PowerShell-7.1.3-win-fxdependentWinDesktop.zip	20.8 MB
PowerShell-7.1.3-win-x64.msi	95 MB
PowerShell-7.1.3-win-x64.zip	96.5 MB
PowerShell-7.1.3-win-x86.msi	85.8 MB
PowerShell-7.1.3-win-x86.zip	87.3 MB

At the time of the writing, the most recent version was 7.1.3.



Once you have the MSI file, double-click on it to install. When you get to the end of the installation wizard, check the box to launch PowerShell 7 and type `$PsVersionTable` and hit Enter. You should see that you have a matching version installed.



```
Administrator: C:\Program Files\PowerShell\7\pwsh.exe
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Program Files\PowerShell\7> $PsVersionTable

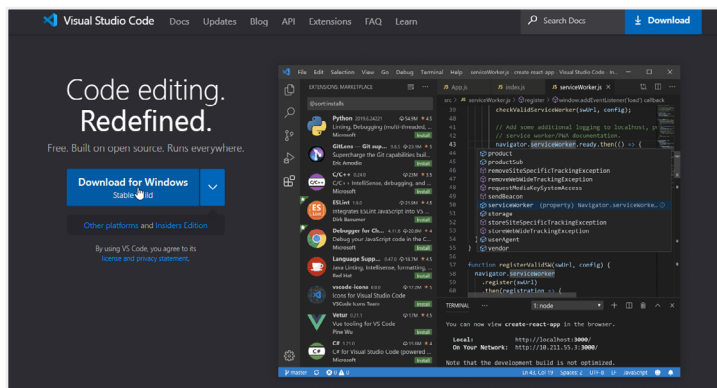
Name                           Value
----                           -
PSVersion                      7.1.3
PSEdition                      Core
GitCommitId                   7.1.3
OS                             Microsoft Windows 10.0.14393
Platform                      Win32NT
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
PSRemotingProtocolVersion      2.3
SerializationVersion          1.1.0.1
WSManStackVersion              3.0

PS C:\Program Files\PowerShell\7>
```

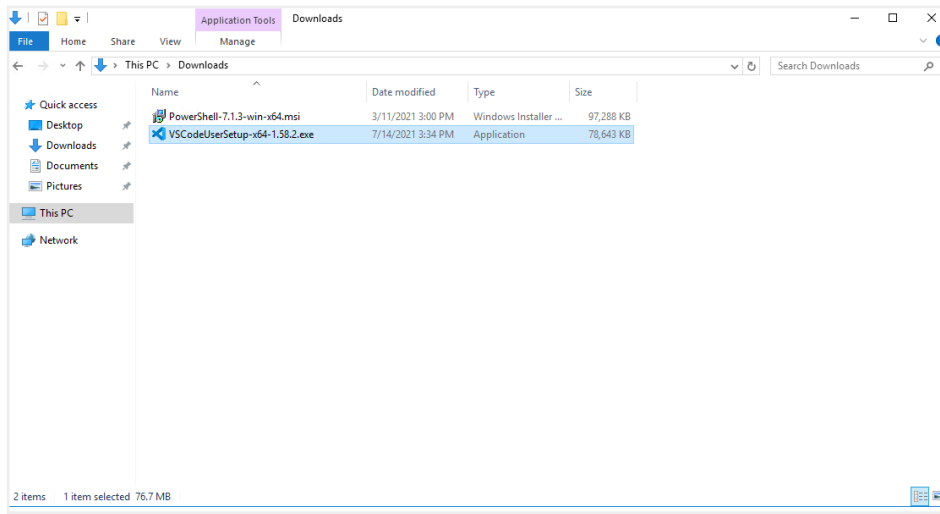
We're ready to move on and install VS Code.

INSTALL VISUAL STUDIO CODE

Like the PowerShell installation, go to the [Visual Studio Code web site](#) and download the most recent stable build for Windows.



The version I received at the time of writing is 1.58.2.



After download, run it on your machine, following the wizard and finish the install.

Once you confirm that Visual Studio Code is installed on your system, you can continue to the **Initial Launch of VS Code** later in this document.

Setup Steps for macOS

High Sierra (v. 10.13) or later is required to install PowerShell and Visual Studio Code on a macOS machine.

BASE SYSTEM DESCRIPTION

The macOS I'm configuring is a MacBook Air running Mojave (10.14.3).

INSTALL POWERSHELL 7

To install PowerShell 7, I followed the instructions on the [Microsoft page](#).

Install Homebrew

Open the Terminal App and type:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

You will most likely be prompted for your password to authorize this install, so enter it when asked.

```
bertoi@ ~ — bash — 102x44
Last login: Sun Aug 15 18:43:48 on console
bertoi@ bertoi:~$ bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for 'sudo' access (which may request your password).
Password:
==> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
==> The following existing directories will be made group writable:
/usr/local/include
/usr/local/lib
/usr/local/lib/pkgconfig
==> The following existing directories will have their owner set to bertoi:
/usr/local/include
/usr/local/lib
/usr/local/lib/pkgconfig
==> The following existing directories will have their group set to admin:
/usr/local/include
/usr/local/lib
/usr/local/lib/pkgconfig
==> The following new directories will be created:
/usr/local/bin
/usr/local/etc
/usr/local/sbin
/usr/local/share
/usr/local/var
/usr/local/opt
/usr/local/share/zsh
/usr/local/share/zsh/site-functions
/usr/local/var/homebrew
/usr/local/var/homebrew/linked
/usr/local/Cellar
/usr/local/Caskroom
/usr/local/Frameworks
==> The Xcode Command Line Tools will be installed.

Press RETURN to continue or any other key to abort

```

After Homebrew is installed, run

```
brew install --cask powershell
```

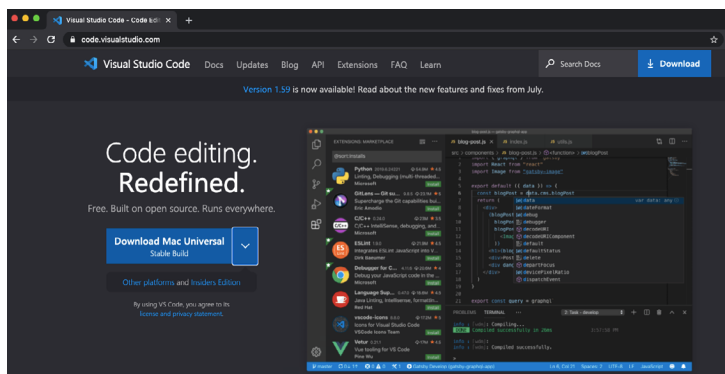
```
bertoi@ ~ — bash — 102x44
bertoi@ bertoi:~$ brew install --cask powershell
--> Downloading https://github.com/PowerShell/PowerShell/releases/download/v7.1.4/powershell-7.1.4-osx-x64.pkg
--> Downloading from https://github.com/PowerShell/PowerShell/releases/download/v7.1.4/powershell-7.1.4-osx-x64.pkg
--> Installing Cask powershell
--> Running installer for powershell; your password may be necessary.
Package installers may write to any location; options such as "--repair" are ignored.
Password:
Installer: Package name is PowerShell - 7.1.4
Installer: Installing at base path /
Installer: The install was successful.
powershell was successfully installed!
bertoi@ bertoi:~$

```

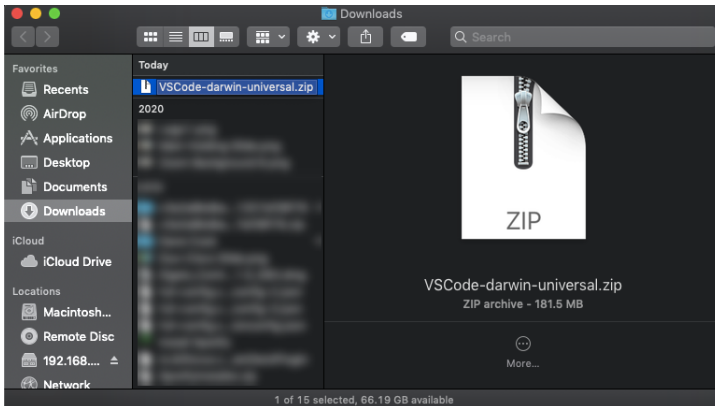
Again, enter your password for authorization.

INSTALL VISUAL STUDIO CODE

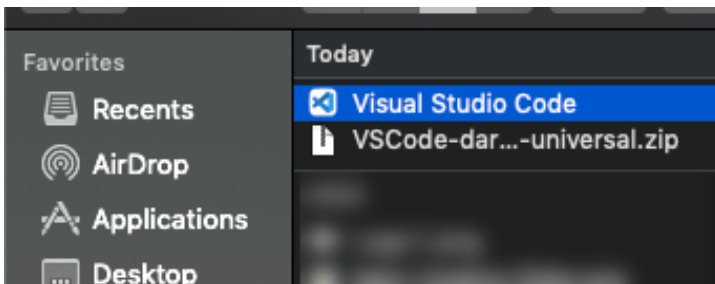
Navigate to <https://code.visualstudio.com> and download the latest stable release for macOS.



Once downloaded, open the ZIP file.



Either leave the “Visual Studio Code” application where it is or drag it to Applications (recommended).



Once you confirm that Visual Studio Code is installed on your system, you can continue to the **Initial Launch of VS Code** later in this document.

Setup Steps for DEB-based Linux

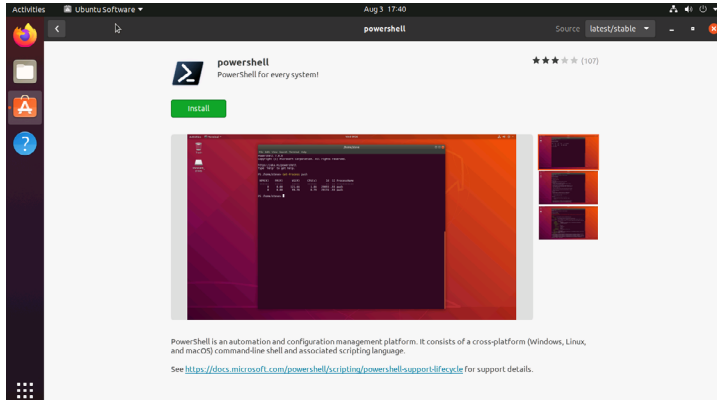
Setting up PowerShell and Visual Studio Code on any DEB-based Linux distribution is nearly identical. This procedure should cover Ubuntu, Debian, SuSe, Mint, and others if they have a GUI installed. For our example here we'll be using Ubuntu. If you encounter problems, please reference the official Microsoft documentation for your specific distribution.

BASE SYSTEM DESCRIPTION

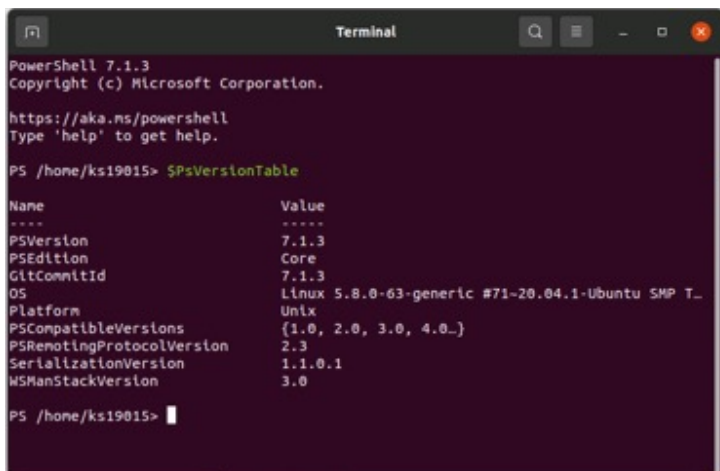
My Ubuntu desktop machine is running 20.04 with the minimal install, but with a GUI. The only thing I did after setting up the machine was to download all applicable updates.

INSTALL POWERSHELL 7

Unlike macOS, PowerShell 7 for Ubuntu is directly available in the Ubuntu Software program. Just search for PowerShell, click it, and then click install.



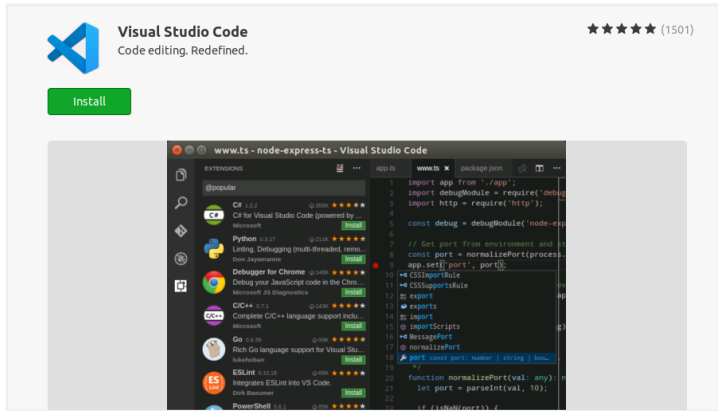
When it's complete, click your launcher select PowerShell. Type `$PsVersionTable` and hit Enter. You should see that you are running the most recent version.



You can close PowerShell and move on to installing Visual Studio Code.

INSTALL VISUAL STUDIO CODE

Like PowerShell, VS Code is available in the Ubuntu Software program. Search for “Visual Studio Code” and install it.



Once you confirm that Visual Studio Code is installed on your system, you can continue to the **Initial Launch of VS Code** later in this document.

Setup Steps for RPM-based Linux

Any RPM-based Linux should follow similar steps as outlined below. These include Red Hat, Rocky, AlmaLinux, Oracle, and others. If your distribution does not support using the Snap Store to install applications, please see the Microsoft documentation for installing PowerShell and VS Code.

BASE SYSTEM DESCRIPTION

In this example, the Rocky Linux machine is running Green Obsidian (8.4) with a GUI and no other special packages. For this example, we'll use Snap (snapcraft.io) to install the necessary packages.

INSTALL SNAP AND THE SNAP STORE

Open the Terminal program and run the following commands one at a time.

If prompted for your password for authorization, enter it.

Install the Extra Packages for Enterprise Linux (EPEL) repository

```
sudo dnf -y install epel-release && sudo dnf -y upgrade
```

```
ks19015@rocky-gui:~$ sudo dnf -y install epel-release && sudo dnf -y upgrade
[sudo] password for ks19015:
Last metadata expiration check: 1:28:55 ago on Wed 04 Aug 2021 09:26:52 AM CDT.
Dependencies resolved.
=====
Package      Architecture Version      Repository    Size
=====
Installing:
epel-release  noarch      8-10.el8     extras        22 k
=====
Transaction Summary
=====
Install 1 Package

Total download size: 22 k
Installed size: 32 k
Downloading Packages:
epel-release-8-10.el8.noarch.rpm          179 kB/s | 22 kB    00:00
-----
Total                                      89 kB/s | 22 kB    00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
```

Install the snap package manager

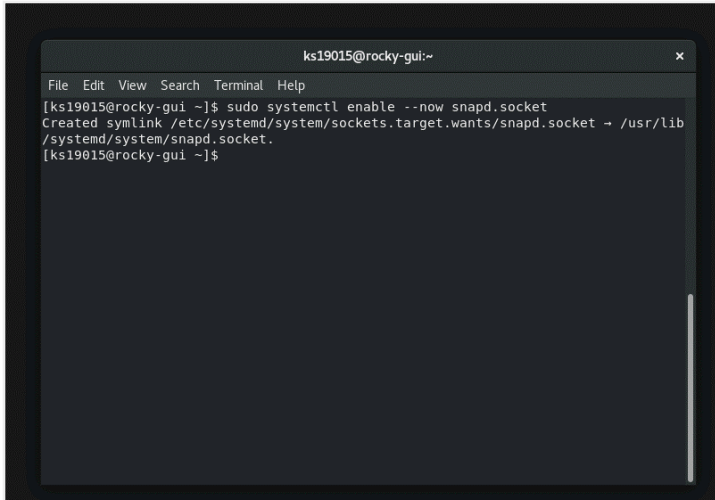
```
sudo dnf -y install snapd
```

```
ks19015@rocky-gui:~$ sudo dnf -y install snapd
Last metadata expiration check: 0:00:40 ago on Wed 04 Aug 2021 10:55:55 AM CDT.
Dependencies resolved.
=====
Package      Architecture Version      Repository    Size
=====
Installing:
snapd        x86_64      2.51-1.el8   epel          17 M
Installing dependencies:
snap-confine x86_64      2.51-1.el8   epel          3.0 M
snapd-selinux noarch      2.51-1.el8   epel          424 k
=====
Transaction Summary
=====
Install 3 Packages

Total download size: 20 M
Installed size: 62 M
Downloading Packages:
(1/3): snapd-selinux-2.51-1.el8.noarch.rpm 1.1 MB/s | 424 kB    00:00
(2/3): snap-confine-2.51-1.el8.x86_64.rpm 4.6 MB/s | 3.0 MB    00:00
(3/3): snapd-2.51-1.el8.x86_64.rpm        9.2 MB/s | 17 MB    00:01
-----
Total                                      9.0 MB/s | 20 MB    00:02
```

Enable the communications needed for snap

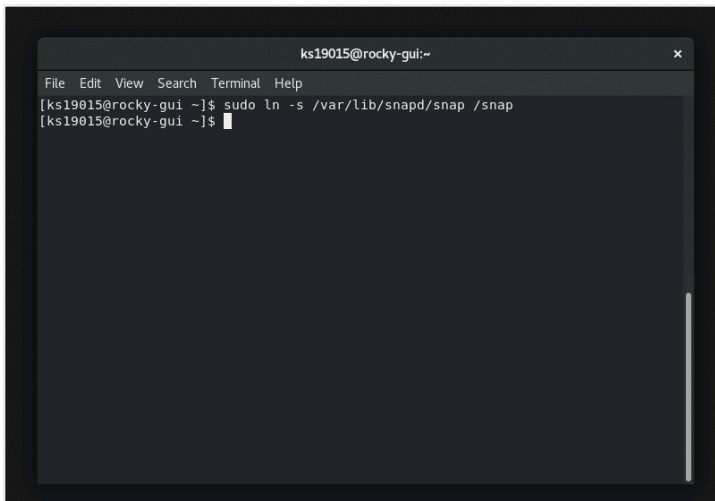
```
sudo systemctl enable --now snapd.socket
```

A terminal window titled 'ks19015@rocky-gui:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'sudo systemctl enable --now snapd.socket' being executed. The output is: 'Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket → /usr/lib/systemd/system/snapd.socket.' followed by a new prompt.

```
ks19015@rocky-gui:~  
File Edit View Search Terminal Help  
[ks19015@rocky-gui ~]$ sudo systemctl enable --now snapd.socket  
Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket → /usr/lib  
/systemd/system/snapd.socket.  
[ks19015@rocky-gui ~]$
```

Build a symbolic link

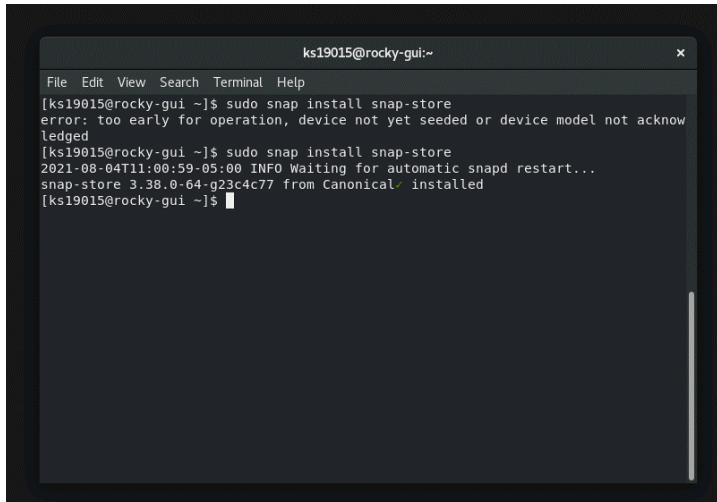
```
sudo ln -s /var/lib/snapd/snap /snap
```

A terminal window titled 'ks19015@rocky-gui:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'sudo ln -s /var/lib/snapd/snap /snap' being executed. The output is a blank line, followed by a new prompt.

```
ks19015@rocky-gui:~  
File Edit View Search Terminal Help  
[ks19015@rocky-gui ~]$ sudo ln -s /var/lib/snapd/snap /snap  
[ks19015@rocky-gui ~]$
```


Install the Snap Store app

```
sudo snap install snap-store
```



```
ks19015@rocky-gui:~  
File Edit View Search Terminal Help  
[ks19015@rocky-gui ~]$ sudo snap install snap-store  
error: too early for operation, device not yet seeded or device model not acknowledged  
[ks19015@rocky-gui ~]$ sudo snap install snap-store  
2021-08-04T11:00:59-05:00 INFO Waiting for automatic snapd restart...  
snap-store 3.38.0-64-g23c4c77 from Canonical installed  
[ks19015@rocky-gui ~]$
```



If you receive the message

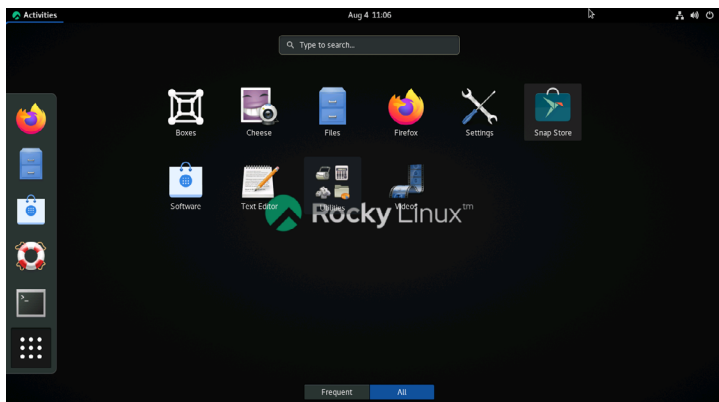
error: too early for
operating, device model not
acknowledged

then just wait a few minutes
and try to install the Snap Store
again.

Once complete, the Snap engine and Snap Store have been installed

**Important: You'll need to log out/in or reboot
for the Snap Store to appear in your launcher**

From here, the steps are the same starting with the **Install PowerShell 7** step for DEB-based Linux above, except you'll do the installations from within the Snap Store app and not the Software app.

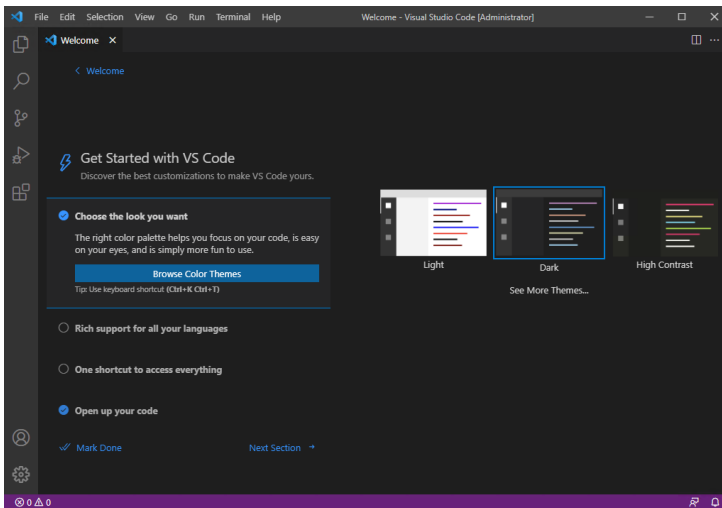


Once you confirm that Visual Studio Code is installed on your system, you can continue to the **Initial Launch of VS Code below**.

Initial Launch of VS Code

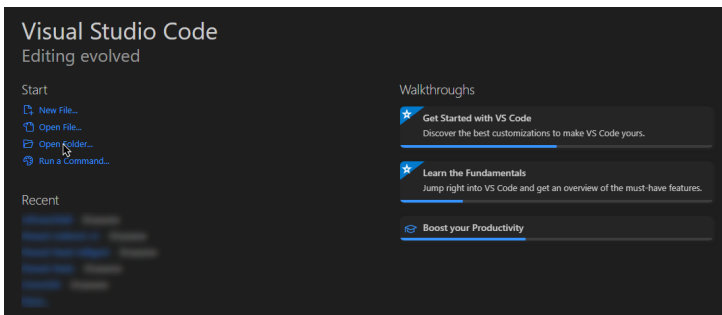
From this point forward the steps are identical regardless of your operating system. Your operating system may ask for permissions and there will be some visual difference on fonts or path formatting ('/' vs. '\'), but for the most part the steps are the same.

Open VS Code and choose a theme.

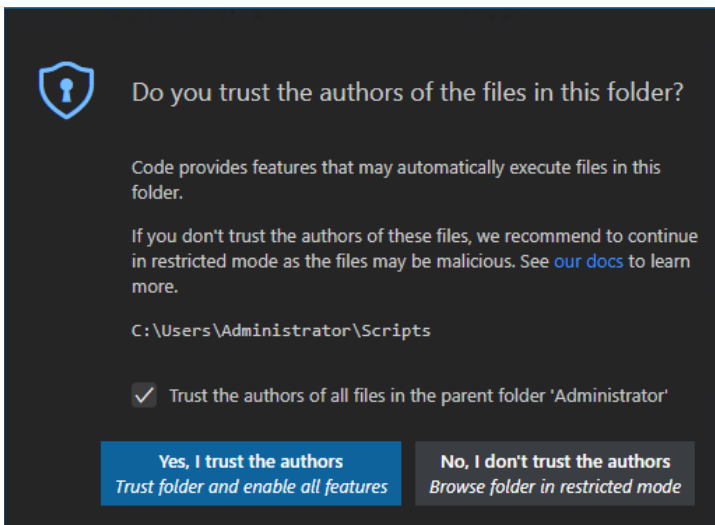


At this point, you can continue to explore the interface, or click Mark Done.

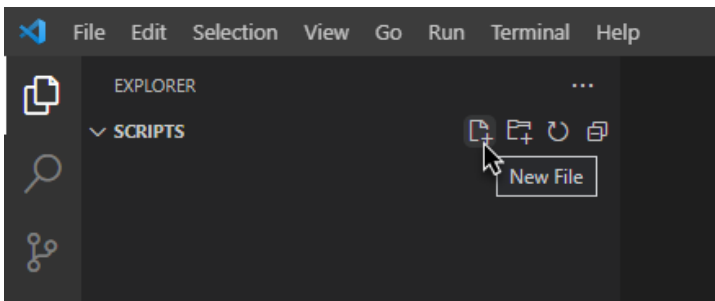
Click Open Folder and create a new folder in your user profile called "Scripts." Select that new "Scripts" folder to open.



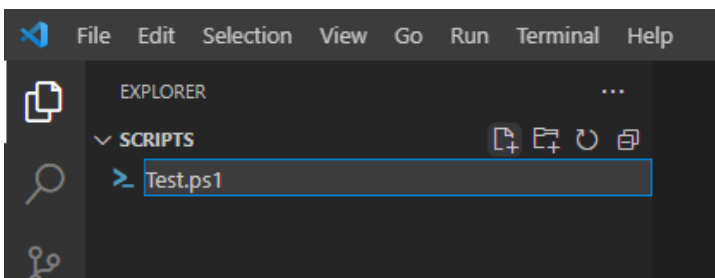
You'll be prompted to trust the authors in the parent folder as well. Since this is in your own user profile, you can select "Trust the authors."



Creating a Test File

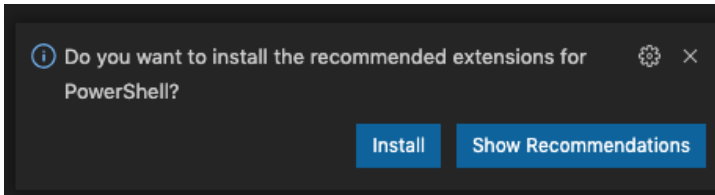


In the left-hand Explorer pane, click the new file icon to create a new file in this folder.



Name that file "Test.ps1" (notice that the icon changes to match the file type).

Since this is the first “PS1” file detected for this install, VS Code prompts if you’d like to install the recommended extensions. You can either show the recommendations and install them as you wish or just take the defaults. This autodetection of file extensions works for most common file types and VS Code will suggest the required extensions for execution.

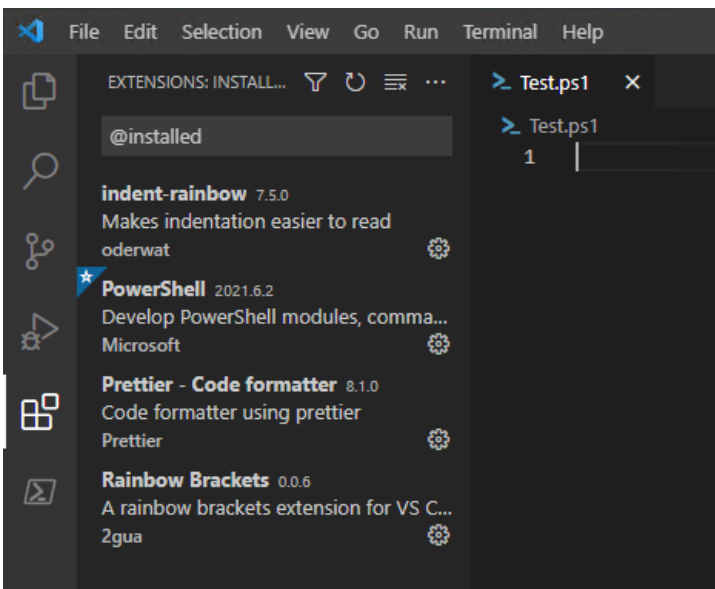


The recommendation for handling PowerShell files is just a single extension and should be installed.

Additional Recommended Extensions

On the left-hand bar, click on the ‘blocks’ to access the extensions. There are several that I like to use in addition to the required one for PowerShell, and you can either take the recommendations or leave them.

- **Rainbow Brackets** – colorizes brackets and parenthesis, making it easier to see your pairs
- **Indent-rainbow** – colorizes your tabs, making it easier to visualize script levels
- **Prettier - Code formatter** – formats your scripts the same way all the time



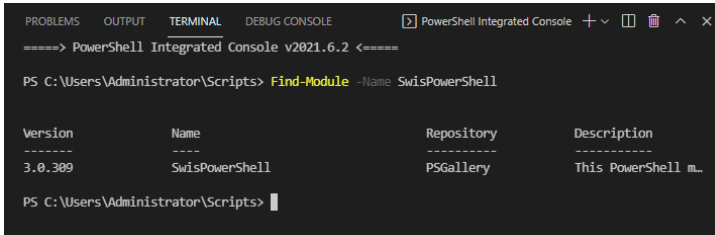
There are several thousand other extensions available. They range from those required to compile/interpret code, those for working with repositories, and those which help with the code layout/formatting itself. Choose those that work best for you and your coding style

Installing the SolarWinds Orion PowerShell Module

In the lower half of the screen, click on the "Terminal" tab and enter

```
Find-Module -Name SwisPowerShell
```

and then hit Enter.



```

===== PowerShell Integrated Console v2021.6.2 <=====
PS C:\Users\Administrator\Scripts> Find-Module -Name SwisPowerShell

Version      Name      Repository      Description
-----
3.0.309      SwisPowerShell      PSGallery      This PowerShell m...

PS C:\Users\Administrator\Scripts>

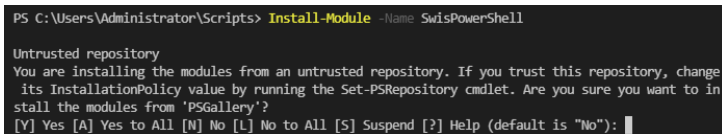
```

This is the module we'll need to install. Make sure it reports version 3.0.0 or later.

To install the module, in the same space, type

```
Install-Module -Name SwisPowerShell
```

followed by Enter.



```

PS C:\Users\Administrator\Scripts> Install-Module -Name SwisPowerShell

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from 'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "No"):

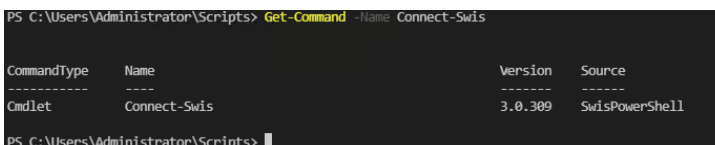
```

You'll be prompted to authorize the PSGallery as a trusted repository if you have not previously installed from it. Type Y and hit Enter to confirm.

Lastly, we'll need to confirm that the module's functions are available, so type

```
Get-Command -Name Connect-Swis
```

and hit Enter.



```

PS C:\Users\Administrator\Scripts> Get-Command -Name Connect-Swis

CommandType      Name      Version      Source
-----
Cmdlet            Connect-Swis      3.0.309      SwisPowerShell

PS C:\Users\Administrator\Scripts>

```

If anything is returned (as above), then that command is available from the module which is listed under Source.

Validating Our Work

We're good to write our Test.ps1 script. Feel free to copy the below into your PS1 file or download the [Test.ps1](#). After you have it in your editor, be sure to change the third line to use the IP or Name of your Orion Server.

```
# this is the name of your SolarWinds Orion server (or IP Address)
# replace with your own server name/IP
$SwisHost = "kmsorion01v.kmsigma.local"
# Do we have a connection already to the SolarWinds Information
Service?
# If not, prompt for credentials and build one
if ( -not $SwisConnection ) {
    $SwisCreds = Get-Credential -Message "Enter Orion credentials to
connect to '$SwisHost'"
    $SwisConnection = Connect-Swis -Hostname $SwisHost -Credential
$SwisCreds
}
# Simple SolarWinds Query Language (SWQL) Query
$SwqlQuery = "SELECT Caption, IPAddress FROM Orion.Nodes"
# Run the query against the SolarWinds Orion server
Get-SwisData -SwisConnection $SwisConnection -Query $SwqlQuery
```



In this script we use basic Orion authentication using a username/password combination. There are other authentication mechanisms available. Review the options directly in PowerShell console using

```
Get-Help -Name Connect-Swis
```

or review the [Connect-Swis](#) documentation.

From the Menu Bar, click Run / Start Debugging, press F5, or click the “play” button in the top-right to execute the script.

In the Terminal tab you'll be prompted for your Orion username and password.

If you get no errors and the data is returned, then you have successfully completed all the steps and are ready to build your own scripts to interact with the SolarWinds Orion API via PowerShell.

```
Enter Orion credentials to connect to 'kmsorion01v.kmsigma.local'
User: admin
Password for user admin: *****

Caption                                IPAddress
-----
KMSORION01v                          192.168.21.65
KMS-NAS                              192.168.4.20
pi-hole01                            192.168.4.7
kmssql01v                            192.168.21.63
Office AP                            192.168.0.123
unifi.kmsigma.local                  192.168.0.1
pi-hole02                            192.168.4.8
folding                              192.168.21.67
Work Laptop                          192.168.199.209
Living Room AP                       192.168.0.72
Kitchen AP                           192.168.0.151
```

Next Steps

Congratulations! You've successfully setup PowerShell 7, Visual Studio Code, imported the SwisPowerShell module and are pulling data from your SolarWinds Orion infrastructure.

In the next chapter we'll discuss pulling more detailed data from your infrastructure including navigating the SWQL Studio tool, leveraging navigation properties to connect data sets, and formatting your output.

If you need further assistance, feel free to peruse the many samples available on the [Orion SDK GitHub repository](#) as well as ask questions in the [Orion SDK](#) forum on THWACK. People are always willing to help, so don't be shy when asking.

ABOUT SOLARWINDS

SolarWinds (NYSE:SWI) is a leading provider of simple, powerful, and secure IT management software. Our solutions give organizations worldwide—regardless of type, size, or complexity—the power to accelerate business transformation in today's hybrid IT environments. We continuously engage with technology professionals—IT service and operations professionals, DevOps and SecOps professionals, and database administrators (DBAs)—to understand the challenges they face in maintaining high-performing and highly available IT infrastructures, applications, and environments. The insights we gain from them, in places like our **THWACK®** community, allow us to address customers' needs now and in the future. Our focus on the user and commitment to excellence in end-to-end hybrid IT management have established SolarWinds as a worldwide leader in solutions for observability, IT service management, application performance, and database management. Learn more today at www.solarwinds.com.



*For additional information, please contact SolarWinds at 866.530.8100 or email sales@solarwinds.com.
To locate an international reseller near you, visit http://www.solarwinds.com/partners/reseller_locator.aspx*

© 2022 SolarWinds Worldwide, LLC. All rights reserved. | 2202-EN

The SolarWinds, SolarWinds & Design, Orion, and THWACK trademarks are the exclusive property of SolarWinds Worldwide, LLC or its affiliates, are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other SolarWinds trademarks, service marks, and logos may be common law marks or are registered or pending registration. All other trademarks mentioned herein are used for identification purposes only and are trademarks of (and may be registered trademarks) of their respective companies.

This document may not be reproduced by any means nor modified, decompiled, disassembled, published or distributed, in whole or in part, or translated to any electronic medium or other means without the prior written consent of SolarWinds. All right, title, and interest in and to the software, services, and documentation are and shall remain the exclusive property of SolarWinds, its affiliates, and/or its respective licensors.

SOLARWINDS DISCLAIMS ALL WARRANTIES, CONDITIONS, OR OTHER TERMS, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, ON THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION NONINFRINGEMENT, ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION CONTAINED HEREIN. IN NO EVENT SHALL SOLARWINDS, ITS SUPPLIERS, NOR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, WHETHER ARISING IN TORT, CONTRACT OR ANY OTHER LEGAL THEORY, EVEN IF SOLARWINDS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.