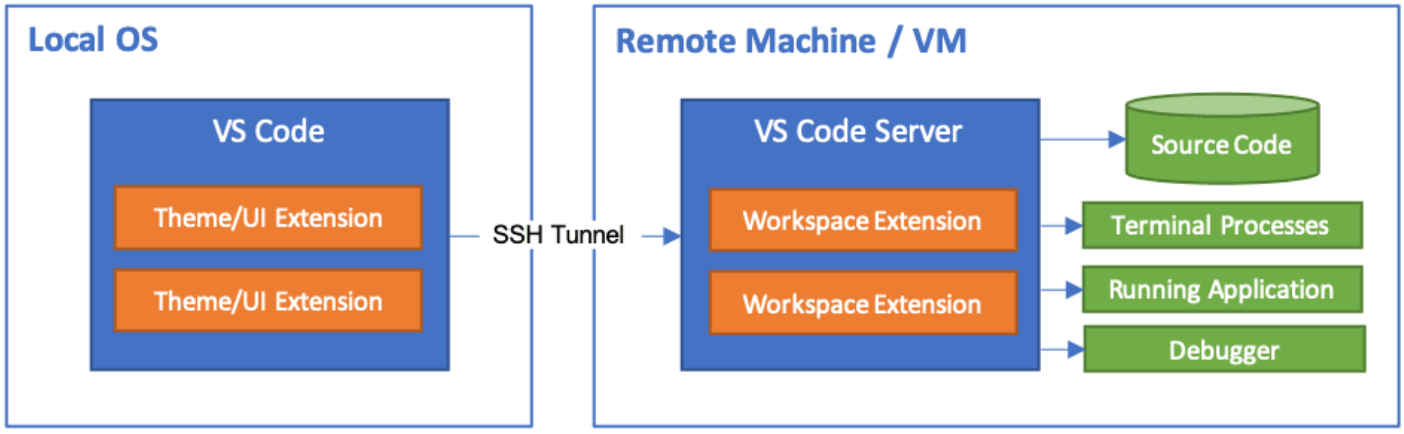
Developing on Remote Machines using SSH and Visual Studio Code

As an extension author, what do I need to do?

Remote Development using SSH

The **Visual Studio Code Remote - SSH** extension allows you to open a remote folder on any remote machine, virtual machine, or container with a running SSH server and take full advantage of VS Code's feature set. Once connected to a server, you can interact with files and folders anywhere on the remote filesystem.

No source code needs to be on your local machine to gain these benefits since the extension runs commands and other extensions directly on the remote machine.



This lets VS Code provide a **local-quality development experience** - including full IntelliSense (completions), code navigation, and debugging - **regardless of where your code is hosted**.

Getting started

Note: After reviewing this topic, you can get started with the introductory SSH tutorial.

System requirements

Local: A supported OpenSSH compatible SSH client must also be installed.

Remote SSH host: A running SSH server on:

x86_64 Debian 8+, Ubuntu 16.04+, CentOS / RHEL 7+.

ARMv7l (AArch32) Raspberry Pi OS (previously called Raspbian) Stretch/9+ (32-bit).

ARMv8l (AArch64) Ubuntu 18.04+ (64-bit).

Windows 10 / Server 2016/2019 (1803+) using the official OpenSSH Server.

macOS 10.14+ (Mojave) SSH hosts with Remote Login enabled.

1 GB RAM is required for remote hosts, but at least 2 GB RAM and a 2-core CPU is recommended.

Other `glibc` based Linux distributions for x86_64, ARMv7l (AArch32), and ARMv8l (AArch64) should work if they have the needed prerequisites. See the Remote Development with Linux article for information prerequisites and tips for getting community supported distributions up and running.

While ARMv7l (AArch32) and ARMv8l (AArch64) support is available, some extensions installed on these devices may not work due to the use of x86 native code in the extension.

Installation

To get started, you need to:

Install an OpenSSH compatible SSH client if one is not already present.

Install Visual Studio Code or Visual Studio Code Insiders.

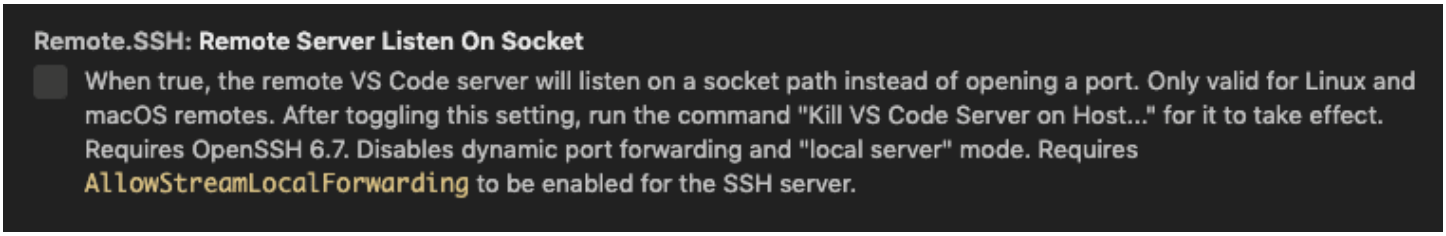
Install the Remote-SSH extension. If you plan to work with other remote extensions in VS Code, you may choose to install the Remote Development extension pack.

SSH host setup

If you do not have an SSH host set up, follow the directions for Linux, Windows 10 / Server (1803+), or macOS SSH host or create a VM on Azure.

Optional: If your Linux or macOS SSH host will be accessed by multiple users at the same time, consider enabling **Remote.SSH: Remote Server Listen On Socket** in VS Code User settings for improved security.

In the Settings editor:



See the Tips and Tricks article for details.

Optional: While password-based authentication is supported, we recommend setting up **key based authentication** for your host. See the Tips and Tricks article for details.

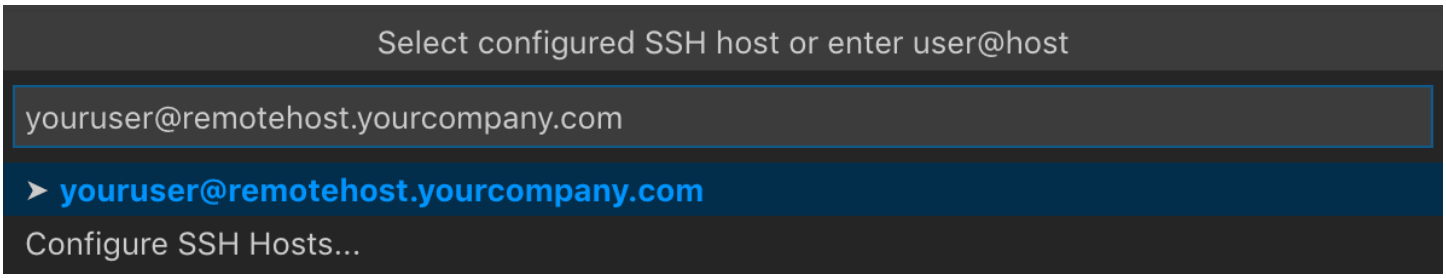
Connect to a remote host

To connect to a remote host for the first time, follow these steps:

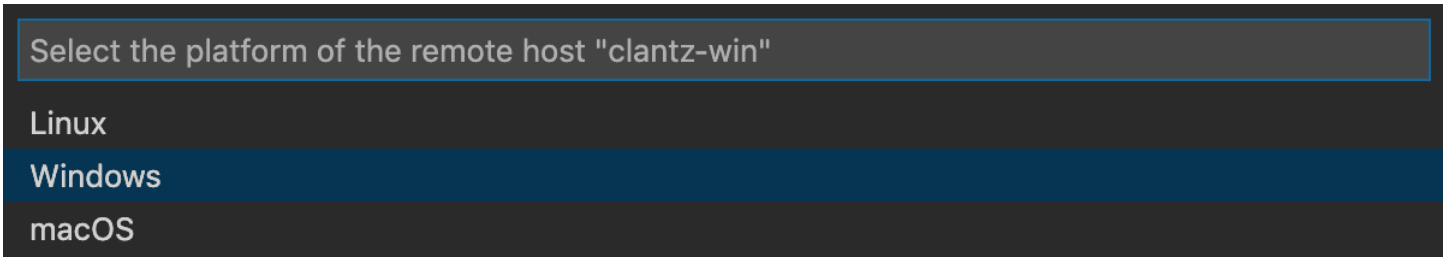
Verify you can connect to the SSH host by running the following command from a terminal / PowerShell window replacing `user@hostname` as appropriate.

```
ssh user@hostname
# Or for Windows when using a domain / AAD account
ssh user@domain@hostname
```

In VS Code, select **Remote-SSH: Connect to Host...** from the Command Palette (F1, ⇧⌘P) and use the same `user@hostname` as in step 1.



If VS Code cannot automatically detect the type of server you are connecting to, you will be asked to select the type manually.

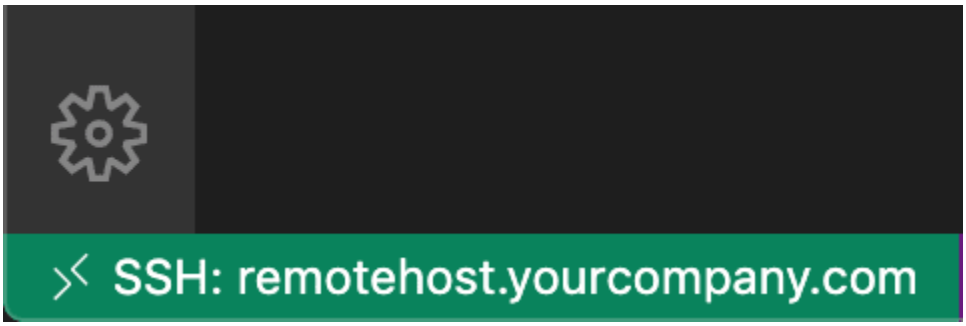


Once you select a platform, it will be stored in VS Code settings under the `remote.SSH.remotePlatform` property so you can change it at any time.

After a moment, VS Code will connect to the SSH server and set itself up. VS Code will keep you up-to-date using a progress notification and you can see a detailed log in the `Remote – SSH` output channel.

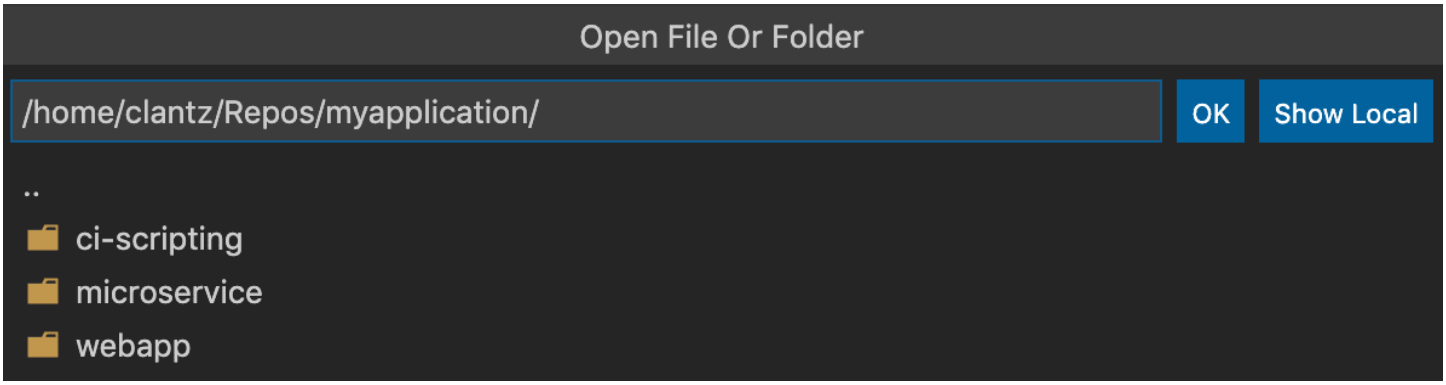
> **Tip:** Connection hanging or failing? See troubleshooting tips for information on resolving common problems.
>
> If you see errors about SSH file permissions, see the section on Fixing SSH file permission errors.

After you are connected, you'll be in an empty window. You can always refer to the Status bar to see which host you are connected to.



Clicking on the Status bar item will provide a list of remote commands while you are connected.

You can then open any folder or workspace on the remote machine using **File > Open...** or **File > Open Workspace...** just as you would locally!



From here, install any extensions you want to use when connected to the host and start editing!

> **Note:** On ARMv7l / ARMv8l `glibc` SSH hosts, some extensions may not work due to x86 compiled native code inside the extension.

Open a folder on a remote SSH host in a container

If you are using a Linux or macOS SSH host, you can use the Remote - SSH and Dev Containers extensions together to open a folder on your remote host inside of a container. You do not even need to have a Docker client installed locally.

To do so:

Follow the installation steps for the Dev Containers extension on your remote host.

Optional: Set up SSH key based authentication to the server so you do not need to enter your password multiple times.

Follow the quick start for the Remote - SSH extension to connect to a host and open a folder there.

Use the **Dev Containers: Reopen in Container** command from the Command Palette (F1,

⇧⌘P).

The rest of the Dev Containers quick start applies as-is. You can learn more about the Dev Containers extension in its documentation. You can also see the Develop on a remote Docker host article for other options if this model does not meet your needs.

Disconnect from a remote host

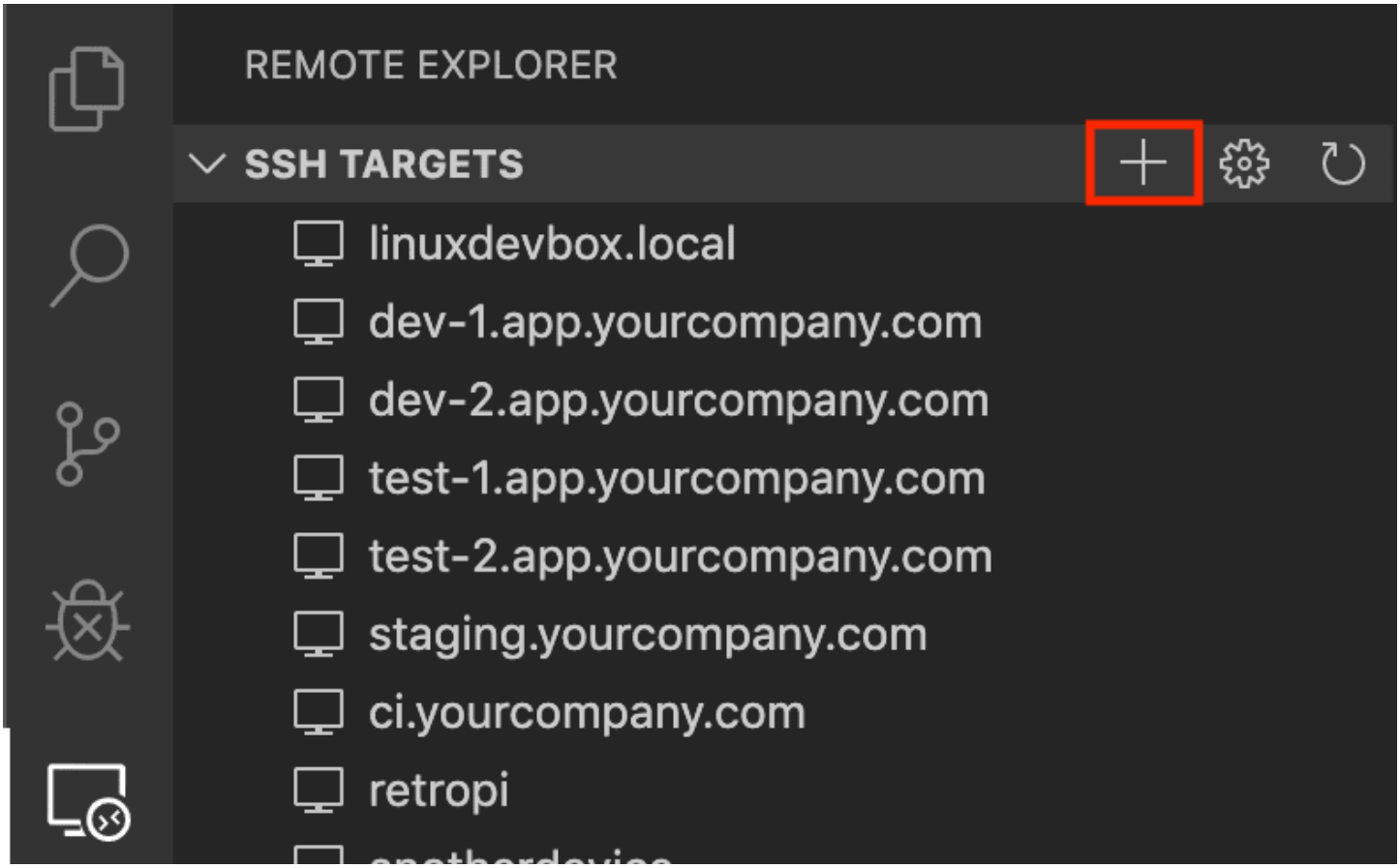
To close the connection when you finish editing files on the remote host, choose **File > Close Remote Connection** to disconnect from the host. The default configuration does not include a keyboard shortcut for this command. You can also simply exit VS Code to close the remote connection.

Remember hosts and advanced settings

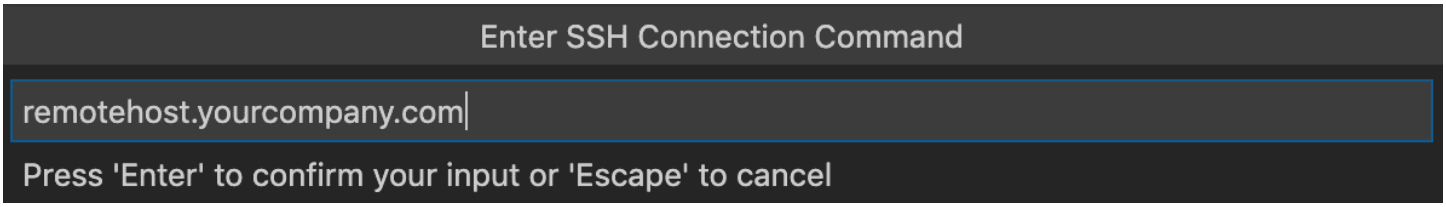
If you have a set of hosts you use frequently or you need to connect to a host using some additional options, you can add them to a local file that follows the SSH config file format.

To make setup easy, the extension can guide you through adding a host without having to hand edit this file.

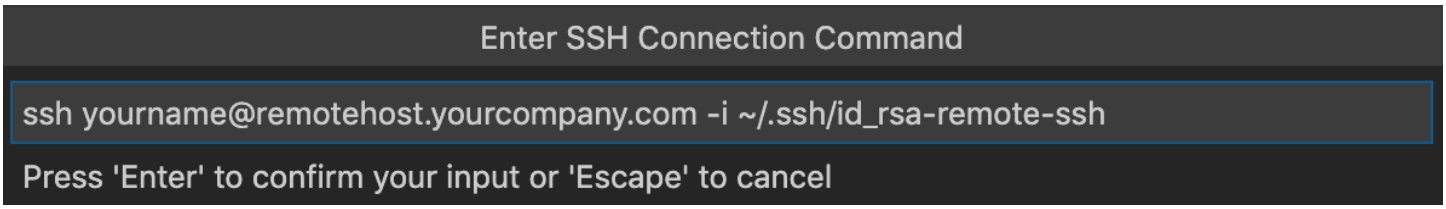
Start by selecting **Remote-SSH: Add New SSH Host...** from the Command Palette (F1, ⇧⌘P) or clicking on the **Add New** icon in the SSH **Remote Explorer** in the Activity Bar.



You'll then be asked to enter the SSH connection information. You can either enter a host name:



Or the full `ssh` command you would use to connect to the host from the command line:



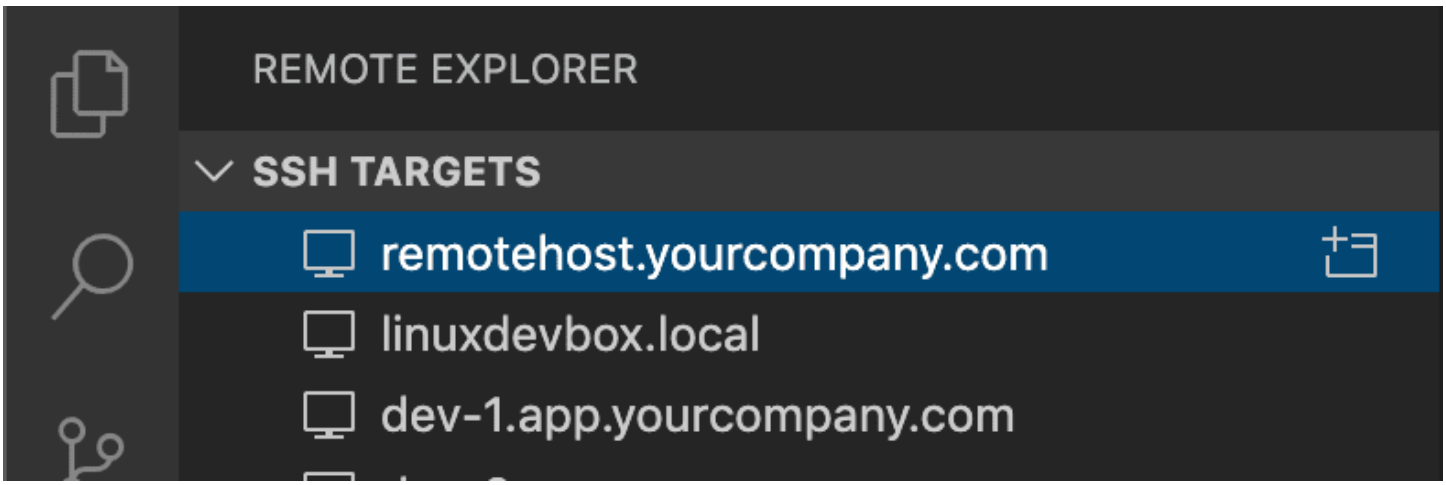
Finally, you'll be asked to pick a config file to use. You can also set the `"remote.SSH.configFile"` property in your User `settings.json` file if you want to use a different config file than those listed. The extension takes care of the rest!

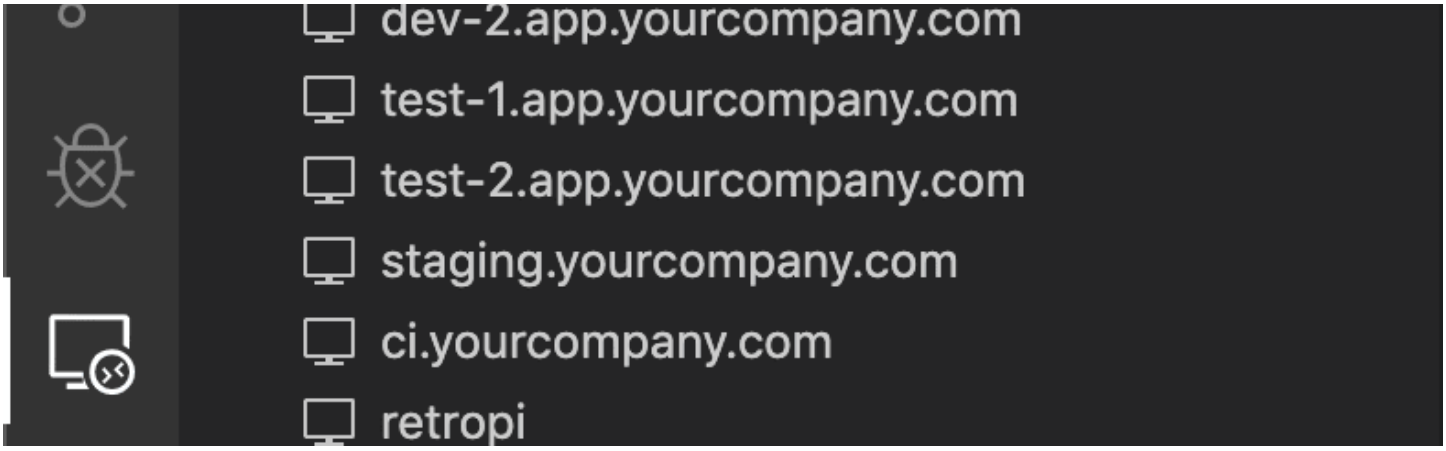
For example, entering `ssh -i ~/.ssh/id_rsa-remote-ssh yourname@remotehost.yourcompany.com` in the input box would generate this entry:

```
Host remotehost.yourcompany.com
    User yourname
    HostName another-host-fqdn-or-ip-goes-here
    IdentityFile ~/.ssh/id_rsa-remote-ssh
```

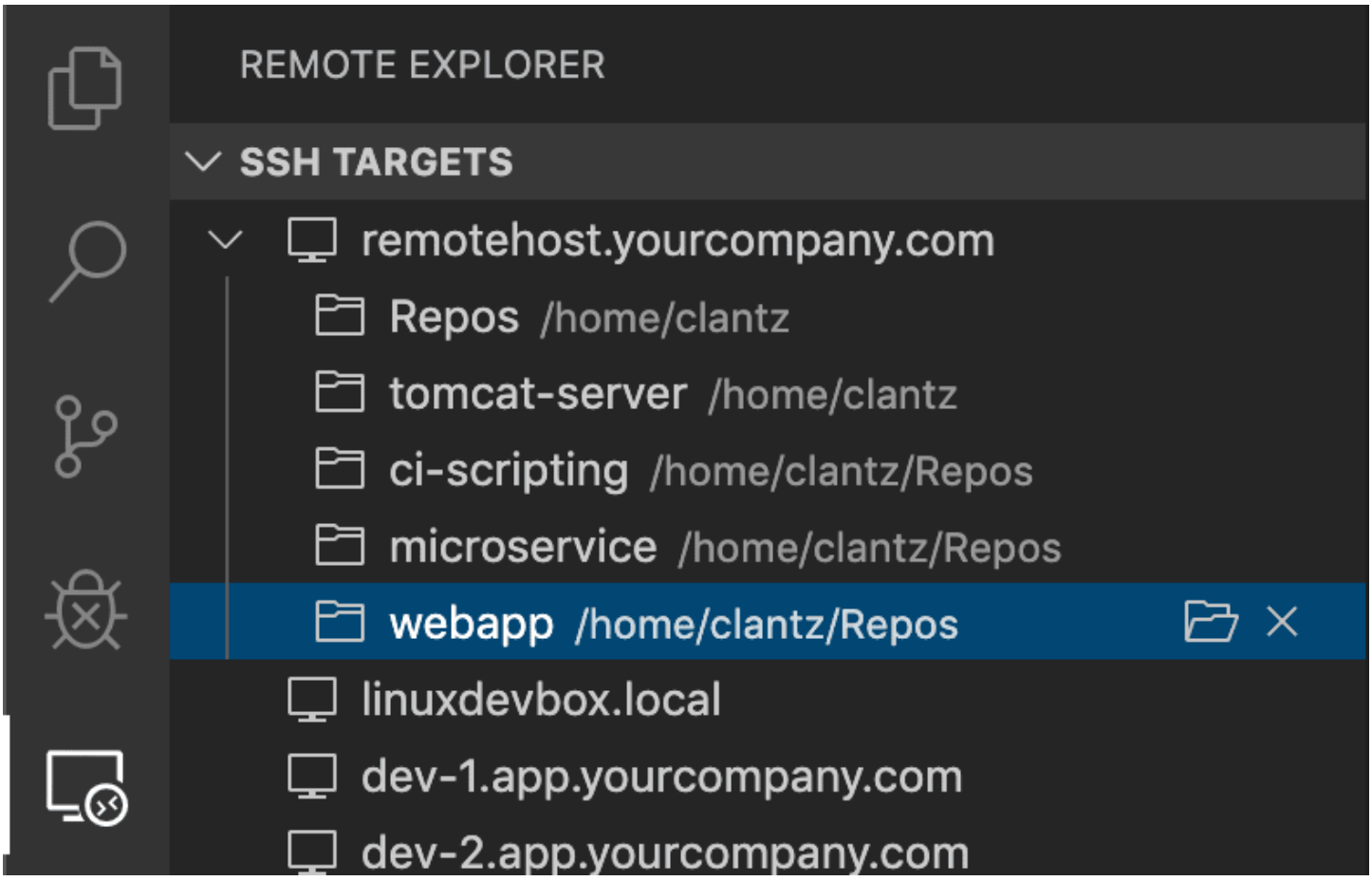
See Tips and Tricks for details on generating the key shown here. You can manually edit this file with anything the SSH config file format supports, so this is just one example.

From this point forward, the host will appear in the list of hosts when you select **Remote-SSH: Connect to Host...** from the Command Palette (F1, ⇧⌘P) or in the **SSH Targets** section of the **Remote Explorer**.





The **Remote Explorer** allows you to both open a new empty window on the remote host or directly open a folder you previously opened. Expand the host and click on the **Open Folder** icon next to the folder you want to open on the host.



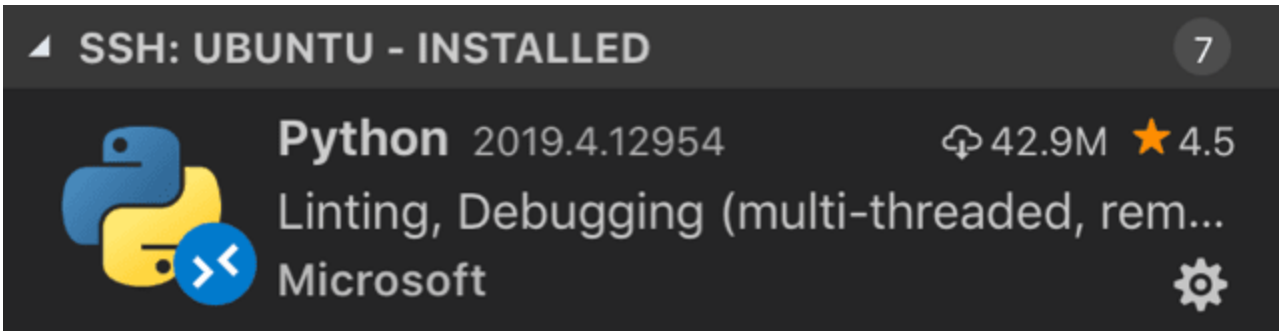
Managing extensions

VS Code runs extensions in one of two places: locally on the UI / client side, or remotely on the SSH host. While extensions that affect the VS Code UI, like themes and snippets, are installed locally, most extensions will reside on the SSH host. This ensures you have smooth experience and allows you to install any needed extensions for a given workspace on an SSH host from your local machine. This

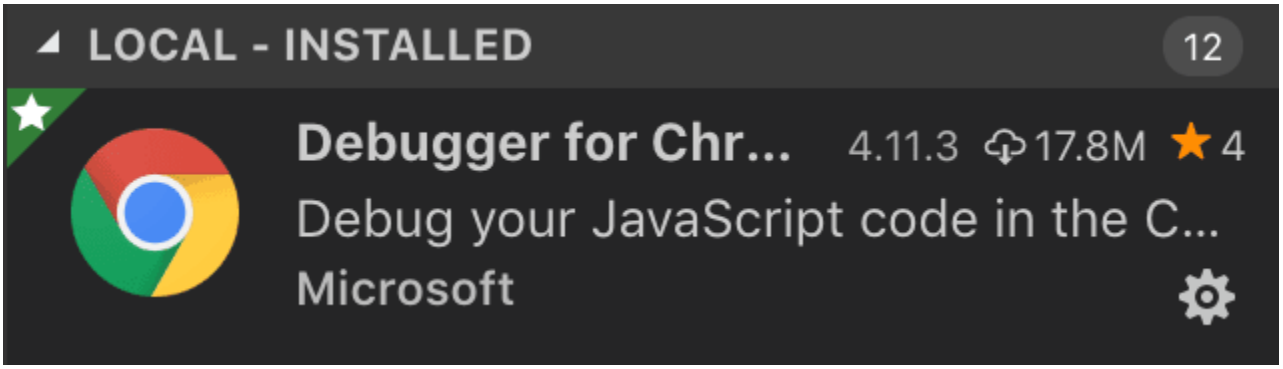
way, you can pick up exactly where you left off, from a different machine complete with your extensions.

If you install an extension from the Extensions view, it will automatically be installed in the correct location. Once installed, you can tell where an extension is installed based on the category grouping.

There will be a category for your remote SSH host:

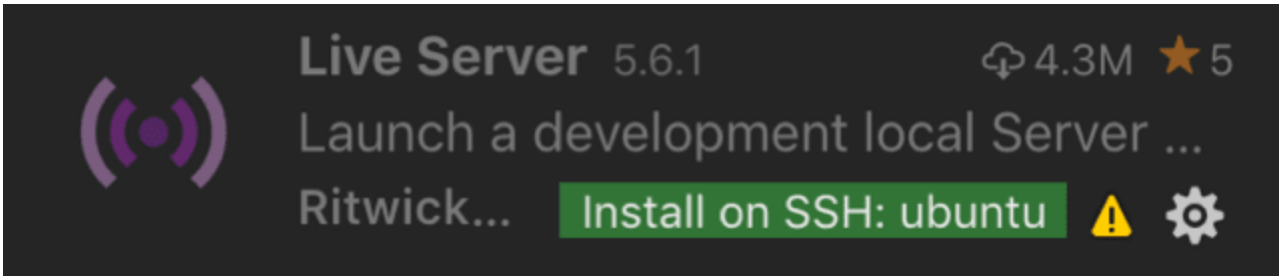


And also a **Local - Installed** category:



> **Note:** If you are an extension author and find that your extension is not working properly or installs in the wrong place, see Supporting Remote Development for details.

Local extensions that actually need to run remotely will appear dimmed and disabled in the **Local - Installed** category. Select **Install** to install an extension on your remote host.



You can also install all locally installed extensions on the SSH host by going to the Extensions view and selecting **Install Local Extensions in SSH: {Hostname}** using the cloud button at the right of the **Local - Installed** title bar. This will display a dropdown where you can select which locally installed extensions to install on your SSH host.

"Always installed" extensions

If there are extensions that you would like to always have installed on any SSH host, you can specify which ones using the `remote.SSH.defaultExtensions` property in `settings.json`. For example, if you wanted to install the GitLens and Resource Monitor extensions, specify their extension IDs as follows:

```
"remote.SSH.defaultExtensions": [
    "eamodio.gitlens",
    "mutantdino.resourcemonitor"
]
```

Advanced: Forcing an extension to run locally / remotely

Extensions are typically designed and tested to either run locally or remotely, not both. However, if an extension supports it, you can force it to run in a particular location in your `settings.json` file.

For example, the setting below will force the Docker extension to run locally and Remote - SSH: Editing Configuration Files extension to run remotely instead of their defaults:

```
"remote.extensionKind": {
    "ms-azuretools.vscode-docker": [ "ui" ],
    "ms-vscode-remote.remote-ssh-edit": [ "workspace" ]
}
```

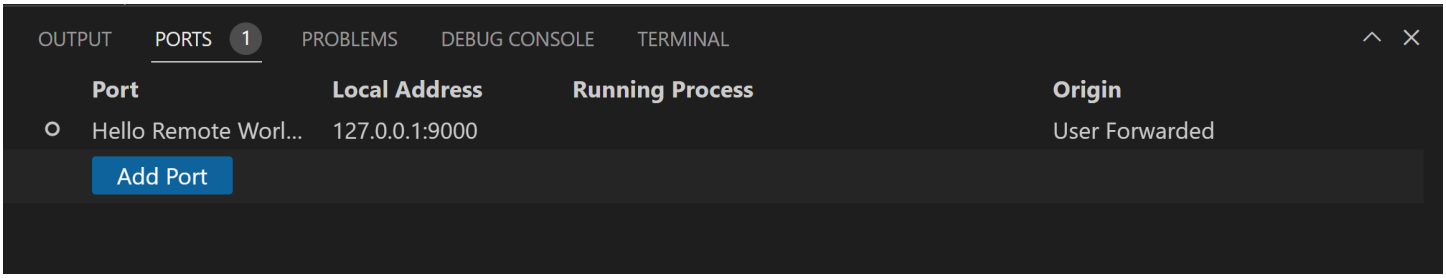
A value of `"ui"` instead of `"workspace"` will force the extension to run on the local UI/client side instead. Typically, this should only be used for testing unless otherwise noted in the extension's documentation since it **can break extensions**. See the article on Supporting Remote Development for details.

Forwarding a port / creating SSH tunnel

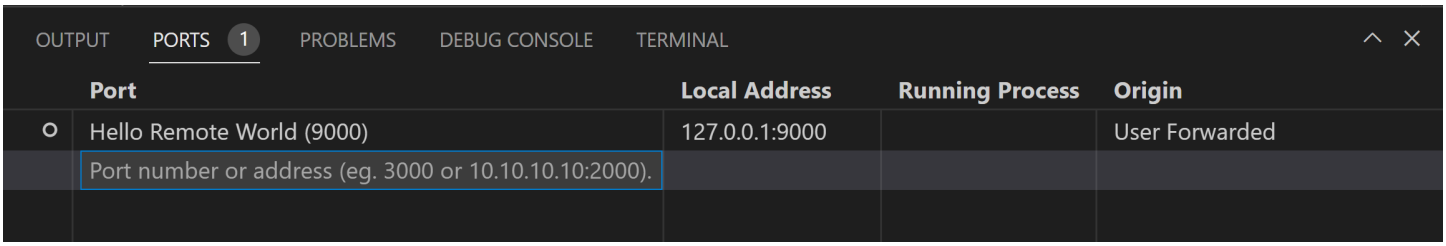
Sometimes when developing, you may need to access a port on a remote machine that is not publicly exposed. There are two ways to do this using an SSH tunnel that "forwards" the desired remote port to your local machine.

Temporarily forwarding a port

Once you are connected to a host, if you want to **temporarily forward** a new port for the duration of the session, select **Forward a Port** from the Command Palette (F1, ⇧⌘P) or select the **Add Port** button in the **Ports view**. You can see the Ports view in the bottom panel, or by running the command **Ports: Focus on Ports View**.



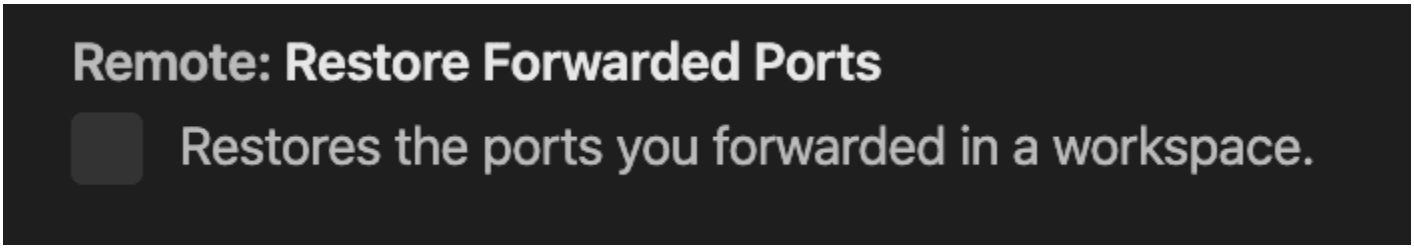
You'll be asked to enter the port you would like to forward and you can give it a name.



A notification will tell you the localhost port you should use to access the remote port. For example, if you forwarded an HTTP server listening on port 3000, the notification may tell you that it was mapped to port 4123 on localhost since 3000 was already in use. You can then connect to this remote HTTP server using `http://localhost:4123`.

This same information is available in the **Forwarded Ports** section of the Remote Explorer if you need to access it later.

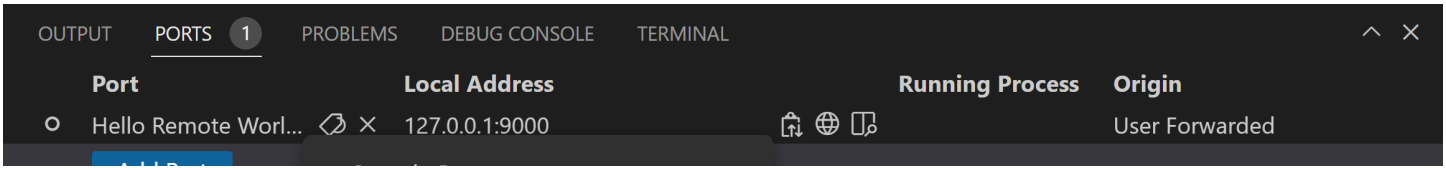
If you would like VS Code to remember any ports you have forwarded, check **Remote: Restore Forwarded Ports** in the Settings editor (⌘,) or set `"remote.restoreForwardedPorts": true` in `settings.json`.

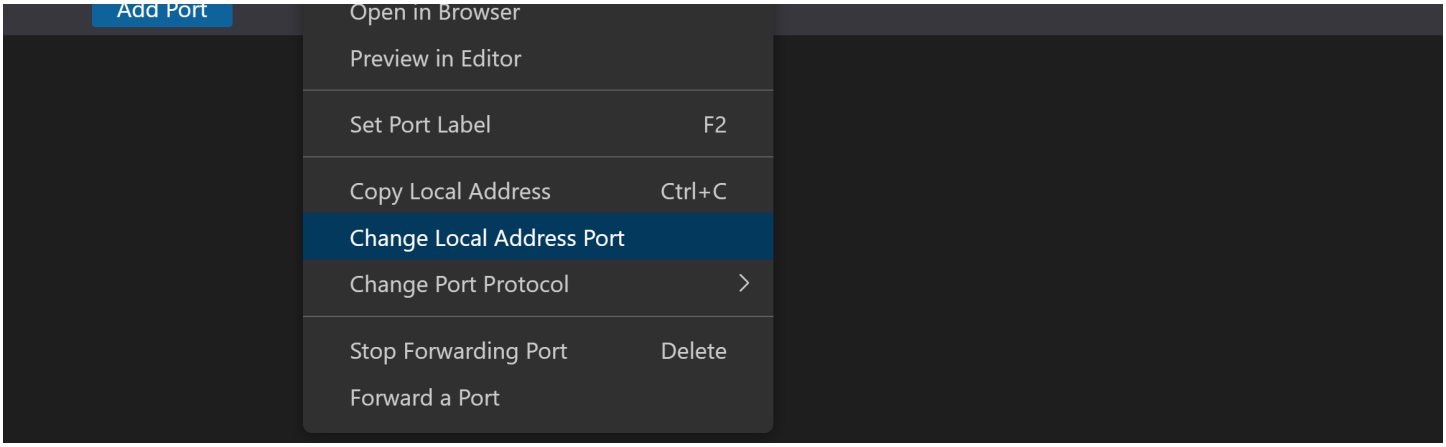


Change local port on tunnel

If you would like the local port of the tunnel to be different than the remote server's, you can change this via the **Forwarded Ports** panel.

Right-click the tunnel you want to modify, and select **Change Local Address Port** in the context menu.





[Always forwarding a port]

If you have ports that you **always want to forward**, you can use the `LocalForward` directive in the same SSH config file you use to [remember hosts and advanced settings].

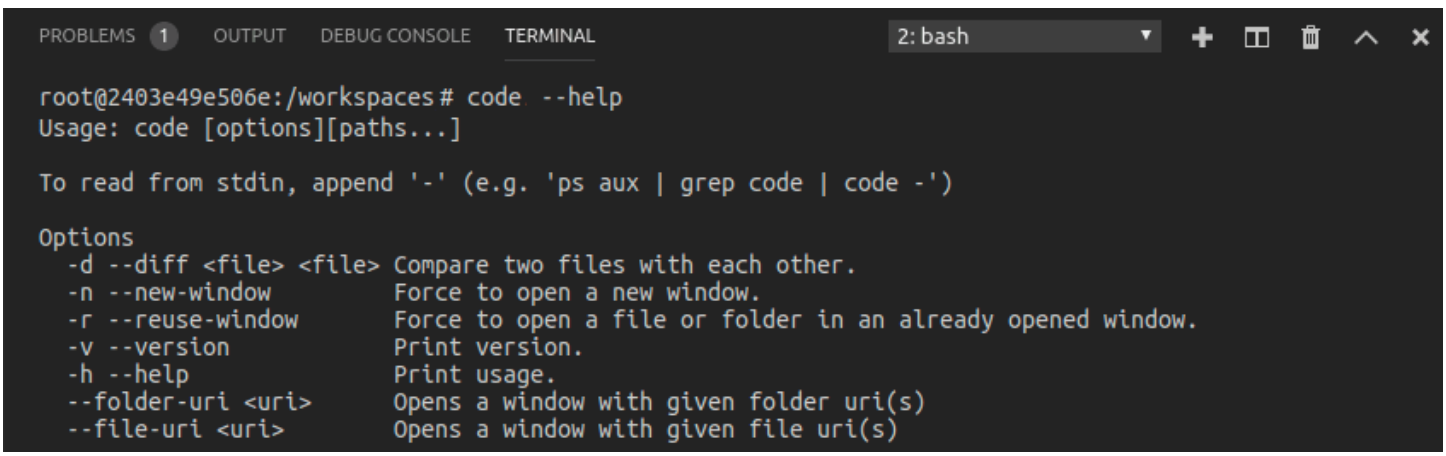
For example, if you wanted to forward ports 3000 and 27017, you could update the file as follows:

```
Host remote-linux-machine
    User myuser
    HostName remote-linux-machine.mydomain
    LocalForward 127.0.0.1:3000 127.0.0.1:3000
    LocalForward 127.0.0.1:27017 127.0.0.1:27017
```

Opening a terminal on a remote host

Opening a terminal on the remote host from VS Code is simple. Once connected, **any terminal window** you open in VS Code (**Terminal > New Terminal**) will automatically run on the remote host rather than locally.

You can also use the `code` command line from this same terminal window to perform a number of operations such as opening a new file or folder on the remote host. Type `code --help` to see all the options available from the command line.



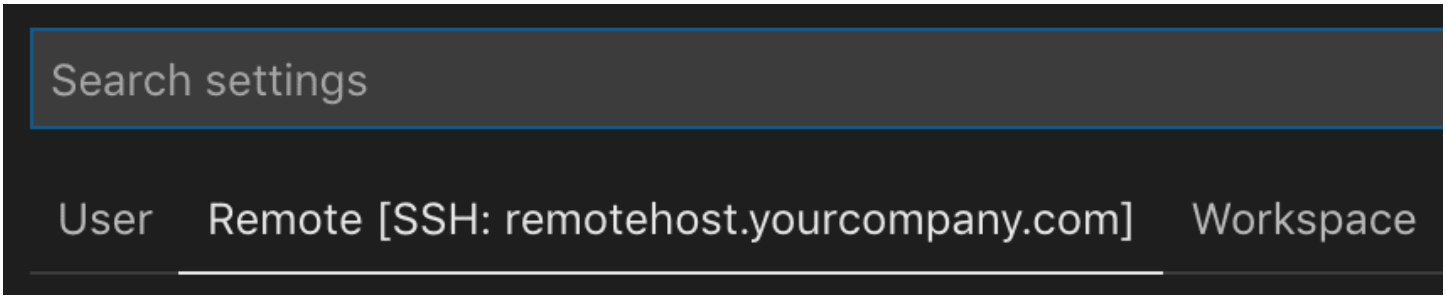
Debugging on the SSH host

Once you are connected to a remote host, you can use VS Code's debugger in the same way you would when running the application locally. For example, if you select a launch configuration in `launch.json` and start debugging (F5), the application will start on remote host and attach the debugger to it.

See the debugging documentation for details on configuring VS Code's debugging features in `.vscode/launch.json`.

SSH host-specific settings

VS Code's local User settings are also reused when you are connected to an SSH host. While this keeps your user experience consistent, you may want to vary some of these settings between your local machine and each host. Fortunately, once you have connected to a host, you can also set host-specific settings by running the **Preferences: Open Remote Settings** command from the Command Palette (F1, ⇧⌘P) or by selecting on the **Remote** tab in the Settings editor. These will override any User settings you have in place whenever you connect to the host. And Workspace settings will override Remote and User settings.



The Remote - SSH extension does not provide direct support for sync'ing source code or using local tools with content on a remote host. However, there are two ways to do this using common tools that will work with most Linux hosts. Specifically, you can:

Mount the remote filesystem using SSHFS.

Sync files to/from the remote host to your local machine using `rsync`.

SSHFS is the most convenient option and does not require any file sync'ing. However, performance will be significantly slower than working through VS Code, so it is best used for single file edits and uploading/downloading content. If you need to use an application that bulk reads/write to many files at once (like a local source control tool), rsync is a better choice.

Known limitations