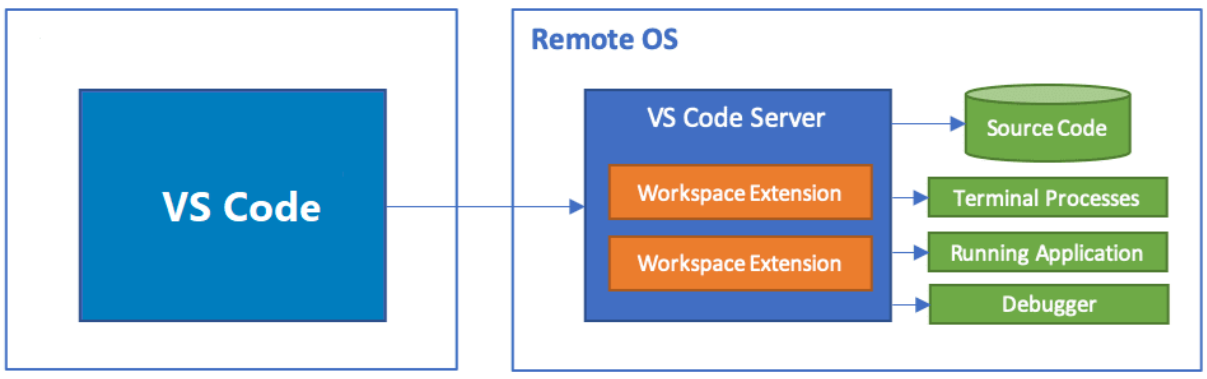
The Visual Studio Code Remote - Tunnels extension lets you connect to a remote machine, like a desktop PC or virtual machine (VM), via a secure tunnel. You can connect to that machine from a VS Code client anywhere, without the requirement of SSH.

Tunneling securely transmits data from one network to another via Microsoft dev tunnels.

This can eliminate the need for source code to be on your VS Code client machine since the extension runs commands and other extensions directly on the remote machine.



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

Getting Started

You have two paths to work with tunnels:

- Run the `tunnel` command of the `code` command-line interface (CLI).
- Enable tunneling through the VS Code Desktop UI.

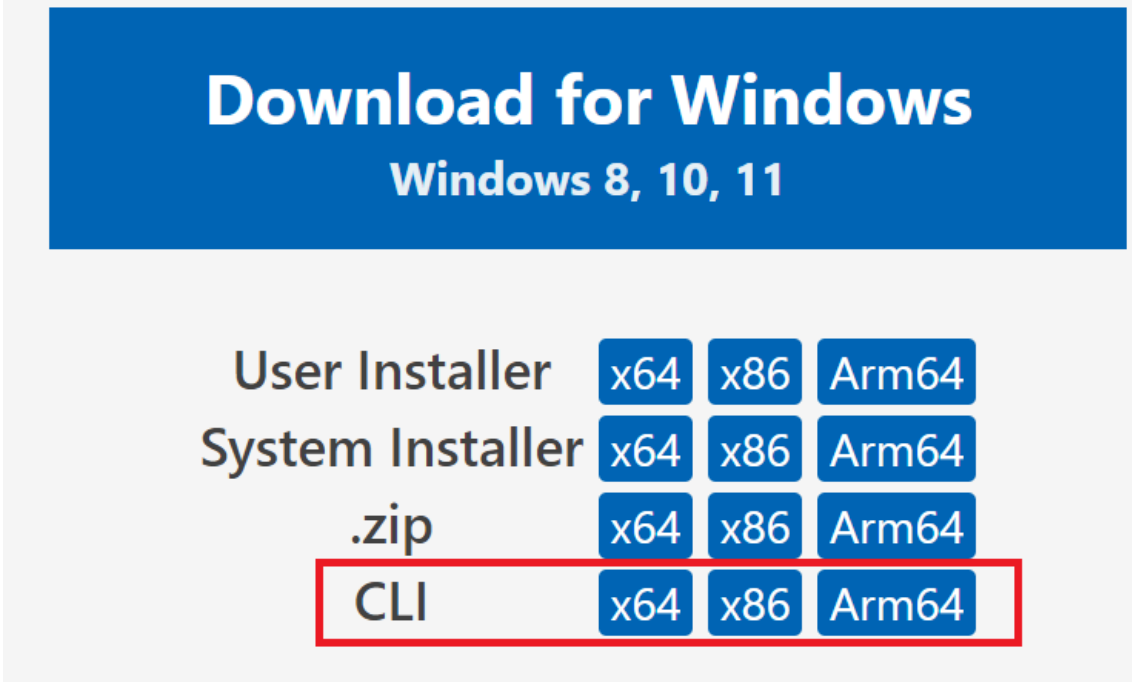
Both of these paths result in the same tunneling functionality – you can use whichever tooling works best for you. The CLI is a great option if you can't install the full VS Code Desktop on your remote machine. Using the VS Code Desktop UI is convenient if you're already doing some work in VS Code and would then like to enable tunneling for your current machine.

Subscribe
Ask questions
Follow @code
Request features
Report issues
Watch videos

required.

Alternative downloads

Alternatively, you can grab the CLI through a standalone install on our download page, which is separate from a VS Code Desktop installation:



You can also install and unpack the CLI through the terminal of your remote machine. This may be especially helpful if your remote doesn't have a UI:

```
curl -Lk 'https://code.visualstudio.com/sha/download?build=stable&os=cli-alpine-x64' --output vscode_cli.tar.gz

tar -xf vscode_cli.tar.gz
```

Copy

> **Note:** If you're using the standalone or terminal install, the commands in the following section will start with `./code` rather than `code`.

2. Create a secure tunnel with the `tunnel` command:

   ```
   code tunnel
   ```

   Copy

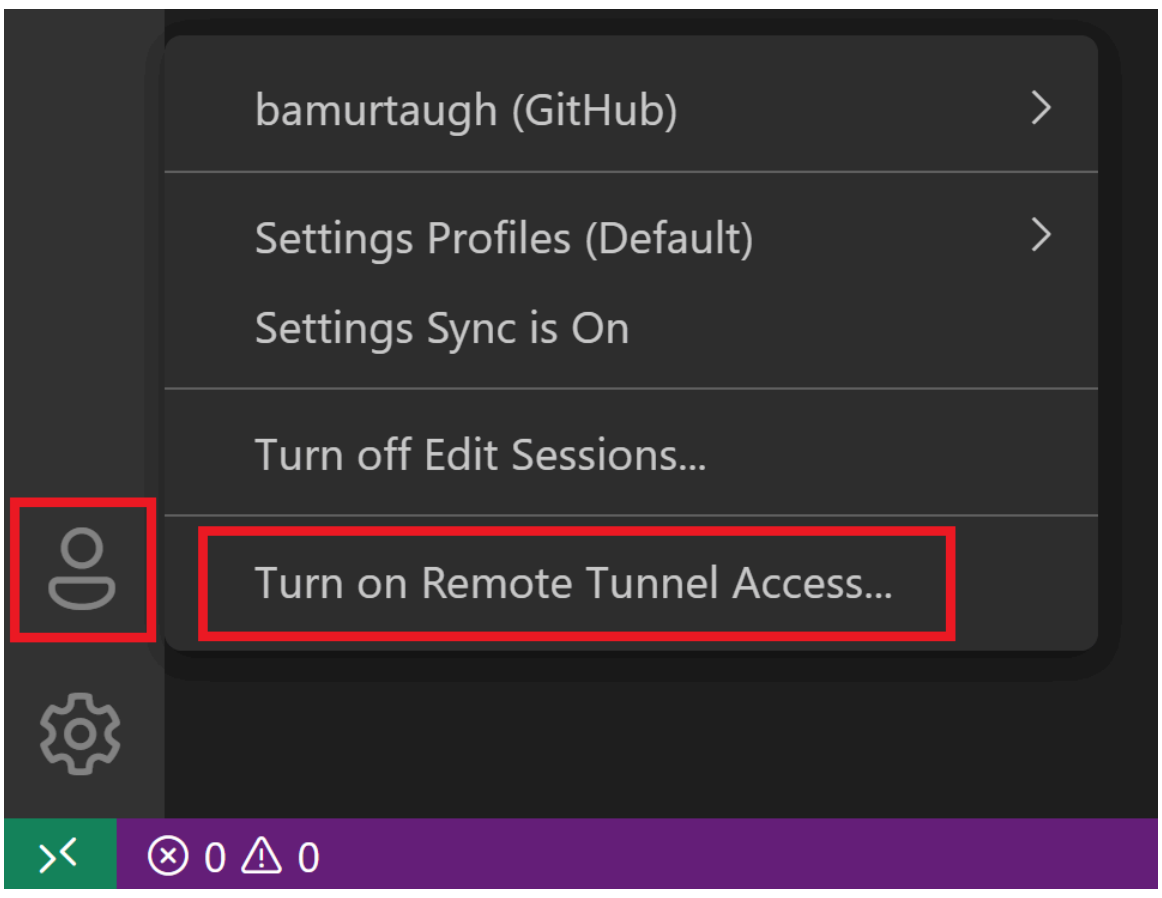
   This command downloads and starts the VS Code Server on this machine and then creates a tunnel to it.

   > **Note:** You will be prompted to accept the server license terms when you first start a tunnel on a machine. You can also pass `--accept-server-license-terms` on the command line to avoid the prompt.

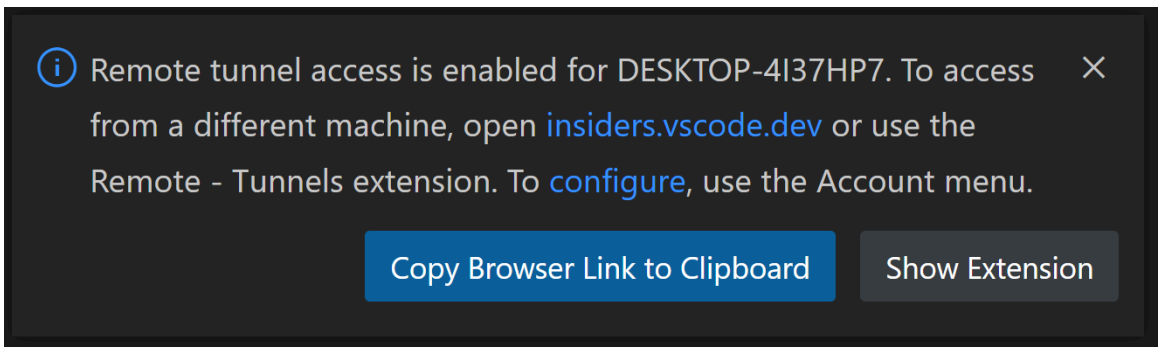
3. This CLI will output a vscode.dev URL tied to this remote machine, such as `https://vscode.dev/tunnel/<machine_name>/<folder_name>`. You can open this URL on a client of your choosing.

4. When opening a vscode.dev URL for the first time on this client, you'll be prompted to log into your GitHub account at a `https://github.com/login/oauth/authorize...` URL. This authenticates you to

(`F1`) in VS Code and run the command **Remote Tunnels: Turn on Remote Tunnel Access...**.



3. You'll be prompted to log into GitHub. Once you're logged in, a tunnel will start up on your current machine, and you'll be able to connect to this machine remotely.



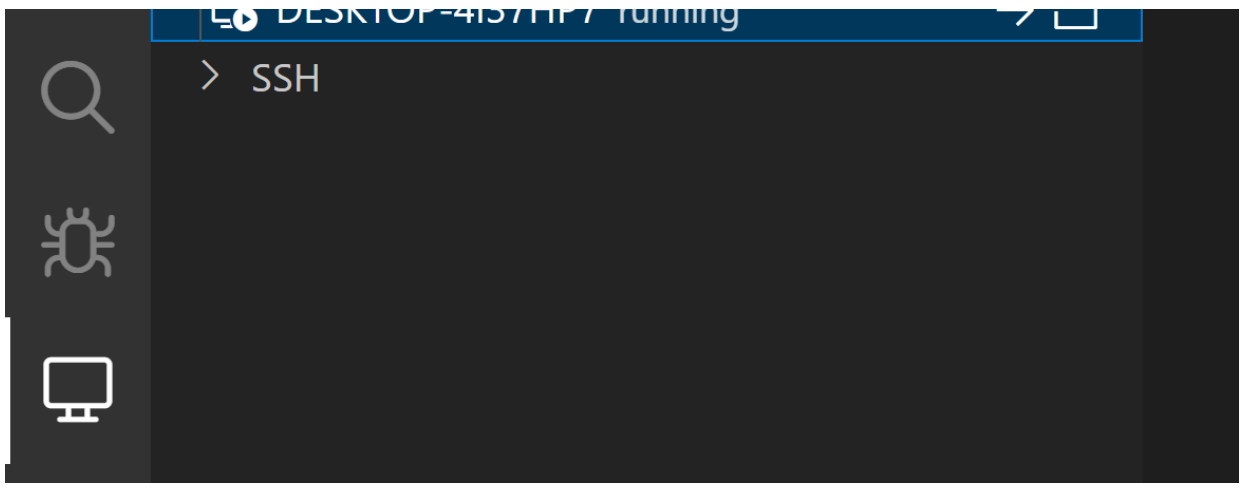
4. In a client of your choice, you may open the vscode.dev link from the notification above and start coding!

> **Note:** The remote machine will only be reachable through a tunnel while VS Code remains running there. Once you exit VS Code it will no longer be possible to tunnel to it until you start VS Code there again or run the `code tunnel` CLI command.

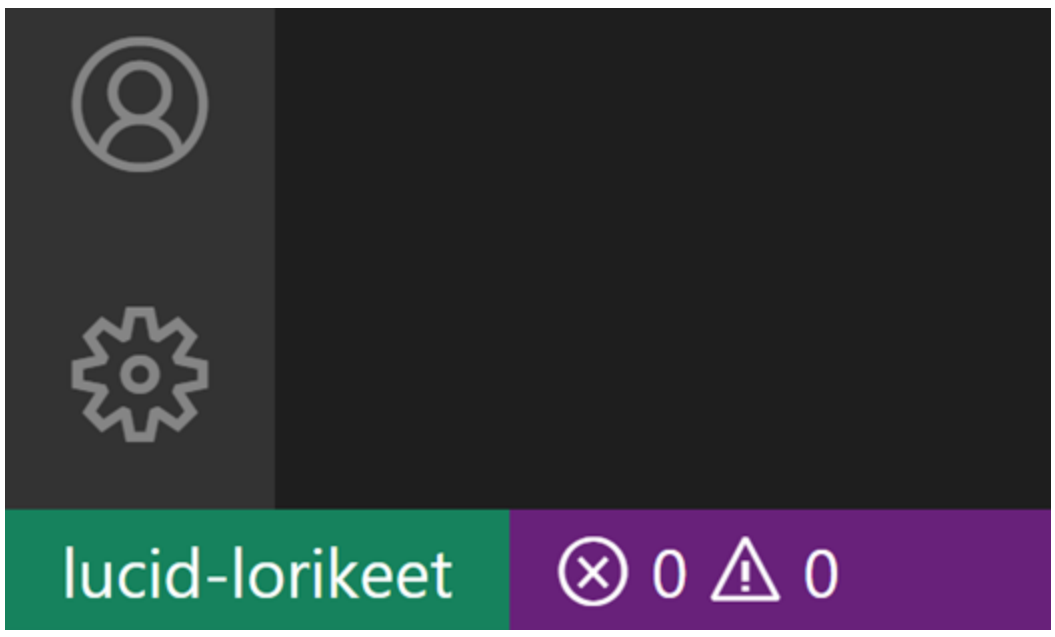
Remote Tunnels extension

The vscode.dev instances you open through the `code` CLI or VS Code UI come with the Remote - Tunnels extension preinstalled.

If you're already working in VS Code (desktop or web) and would like to connect to a remote tunnel, you can install and use the Remote - Tunnels extension directly. Once you install the extension, open the Command Palette (`F1`) and run the command **Remote Tunnels: Connect to Tunnel**. You'll be able to connect to any remote machines with an active tunnel.



Like the other Remote Development extensions, the name of your remote machine will be listed in the lower left green remote indicator. Clicking on this indicator is another way to explore Remote Tunnels commands, along with options to close your remote connection or install VS Code Desktop.



Open a folder on a Remote Tunnels host in a container

You can use the Remote - Tunnels 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:

1. Follow the installation steps for installing Docker on your remote host and VS Code and the Dev Containers extension locally.
2. Follow the Getting Started instructions for the Remote - Tunnels extension to set up a tunnel, connect to it and open a folder there.
3. Use the **Dev Containers: Reopen in Container** command from the Command Palette (`F1`, `Ctrl+Shift+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.

Common questions

machine, and file system.

We've released this VS Code Server backend component as a service you can run yourself (which you may read more about in its documentation), rather than it only being solely installed and managed by the Remote Development extensions.

Accessing the VS Code Server involves a few components:

- The VS Code Server: Backend server that makes VS Code remote experiences possible.
- Remote - Tunnels extension: Extension that facilitates the connection to the remote machine, where you have an instance of the server running.

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

The VS Code extension API abstracts away local/remote details so most extensions will work without modification. However, given extensions can use any node module or runtime they want, there are situations where adjustments may need to be made. We recommend you test your extension to be sure that no updates are required. See Supporting Remote Development for details.

Can multiple users or clients access the same remote instance simultaneously?

No, an instance of the server is designed to be accessed by one user or client at a time.

How do I remove a tunnel or machine?

If you'd like to stop a tunnel you're running via the CLI, you may use `Ctrl + C` to end the active tunnel. If you've enabled tunneling through the VS Code UI, you can run the command **Remote Tunnels: Turn off Remote Tunnel Access...** in VS Code.

You can remove a machine's association with tunneling by running `code tunnel unregister` on that machine. You can also open any VS Code client, select the Remote Explorer view, right-click on the machine you'd like to remove, and select **unregister**.

How are tunnels secured?

Both hosting and connecting to a tunnel requires authentication with the same Github or Microsoft account on each end. In both cases, VS Code will make outbound connections to a service hosted in Azure; no firewall changes are generally necessary, and VS Code doesn't set up any network listeners.

Once you connect from a remote VS Code instance, an SSH connection is created over the tunnel in order to provide end-to-end encryption. The current preferred cipher for this encryption is AES 256 in CTR mode, and the code that implements this is open source.

You can learn more about the security of the underlying dev tunnels service in its documentation.

Are there usage limits for the tunneling service?

Can I configure policies across my organization?

If you're part of an organization that wants to control access to port forwarding, you can do so by allowing or denying access to the domain `global.rel.tunnels.api.visualstudio.com`.

For users running Windows devices, you can also configure and then deploy group policy settings for dev tunnels. You can learn more in the dev tunnels documentation.

How can I ensure I keep my tunnel running?

You have a few options:

- Use the `service` command to run as a service. You can run `code tunnel service install` and `code tunnel service uninstall` to install and remove them.
- Use the `no-sleep` option, `code tunnel --no-sleep`, to prevent your remote machine from going to sleep.

As mentioned in the `code` CLI doc, you can explore all the possible CLI commands and options through `code tunnel --help`.

Can I use other Remote Development Extensions or a dev container while I'm tunneling?

Yes! Currently, you can connect to WSL and dev containers over Remote - Tunnels.

10/29/2024