How To Set Up a Full Development Environment on the iPad

Author: Kayvan Sylvan *Updated: 8/25/2022*

This guide walks through the step by step process of setting up a full remote development environment using Microsoft's free VSCode code-server.

This setup can even be used on an iPad with an attached keyboard and mouse or trackpad combination.

Note that code-server isn't safe to expose over the public internet. Therefore, we will use the Tailscale free solo plan to set up a VPN fabric to securely connect your iPad with the Linux server running code-server.

We will use the Caddy web server to serve up the code-server instance as well as give file browser access to the home directory (to be able to access, for example, the "cargo doc" generated local documentation pages created by the Rust tool chain). Caddy integrates with Tailscale to automatically fetch a LetsEncrypt SSL certificate, and this makes the full functionality of VSCode available in the web application.

Set up a Cloud Server

This could be using any VPS provider. I use Vultr and it's a fast and good service for the value. You can use my referral code to create an account.

Set up a Debian server in your locality. In my case, the server I set up is:

- Cloud Compute (cheap, shared vCPU)
- CPU: intel regular performance (powered by previous generation Intel CPUs and regular SSD, but fast enough)
- Location: Silicon Valley
- Base OS: Debian 11
- Server Size (small, 25GB SSD)
- No automatic backups (this is just for a Dev server, and with the correct setup, you commit often to your GitHub/GitLab repo, so no reason for backups).
- Enable ipv6.
- Add your ssh keys.
- Set the hostname in the dashboard.

This setup comes to $5 per month. Press submit and wait till the server comes up.

Add Local User and Setup sudoers

Login to the cloud server via ssh and set up a local user and add that user to sudoers.

```
LOCAL_USER=kayvan
# useradd $LOCAL_USER -s /bin/bash
cat > /etc/sudoers.d/$LOCAL_USER << EOF
# User rules for $LOCAL_USER
$LOCAL_USER ALL=(ALL) NOPASSWD:ALL
EOF
```

Now add your local ssh keys into ~/.ssh/authorized_keys and test that you have access:

```
% ssh kayvan@XXX.XX.XX.XXX
Linux rust-dev 5.10.0-16-amd64 #1 SMP Debian 5.10.127-2 (2022-07-23)
x86_64

The programs included with the Debian GNU/Linux system are free
software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Aug 25 03:08:18 2022 from 163.114.132.6
kayvan@rust-dev:~$
```

Now we can set up the software for the remote development environment.

Install Tailscale on Debian

The version of Debian is 11.0 (bullseye), so follow the instructions here.

```
curl -fsSL https://pkgs.tailscale.com/stable/debian/bullseye.noarmor.gpg
| sudo tee /usr/share/keyrings/tailscale-archive-keyring.gpg >/dev/null
curl -fsSL
https://pkgs.tailscale.com/stable/debian/bullseye.tailscale-keyring.list
| sudo tee /etc/apt/sources.list.d/tailscale.list
sudo apt-get update
sudo apt-get install tailscale
sudo tailscale up
```

Authenticate using your browser and then verify that you can access the server on your Tailscale VPN mesh by doing ssh to the IP address reported by the following command:

```
tailscale ip -4
```

Install code-server

```
curl -fsSL https://code-server.dev/install.sh | sh
sudo systemctl enable --now code-server@$USER
```

Set up code-server to run on the tailscale interface and restart the server:

```
cat > ~/.config/code-server/config.yaml << EOF
bind-addr: $(tailscale ip -4):8080
auth: none
cert: false
EOF
sudo systemctl restart code-server@$USER
```

Make sure to open the 8080, http, and https ports via ufw:

```
sudo ufw allow 8080
sudo ufw allow http
sudo ufw allow https
sudo service ufw restart
```

Now, on the tailscale network, you should be able to connect to port 8080 on that IP address to verify that the setup so far is working. However, at this point, we are not using the HTTPS port.

Enable HTTPS via the Tailscale Dashboard

Visit the DNS settings of your Tailscale dashboard here https://login.tailscale.com/admin/dns

Enable MagicDNS and HTTPS Certificates here. This will make it possible for us to set up Caddy to automatically provision and use a LetEncrypt SSL certificate.

When MagicDNS and HTTPS are enabled, go through the menus to generate the “domain alias” and note it down for further use later. In my setup, this is the random domain alias:

```
TS_DOMAIN_ALIAS=feist-goblin.ts.net
```

Install Caddy

Run the following commands to install Caddy on debian:

```
sudo apt install -y debian-keyring debian-archive-keyring
apt-transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo
gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt'
| sudo tee /etc/apt/sources.list.d/caddy-stable.list
```

```
sudo apt update
sudo apt install caddy
```

Enable Caddy to Work with Tailscale HTTPS certificates

```
sudo tee -a /etc/default/tailscaled  << EOF

#
https://tailscale.com/kb/1190/caddy-certificates/#provide-non-root-users
-with-access-to-fetch-certificate
TS_PERMIT_CERT_UID=caddy
EOF

sudo service tailscaled restart
sudo service caddy restart
```

Now, we fix the Caddyfile setup to look like this:

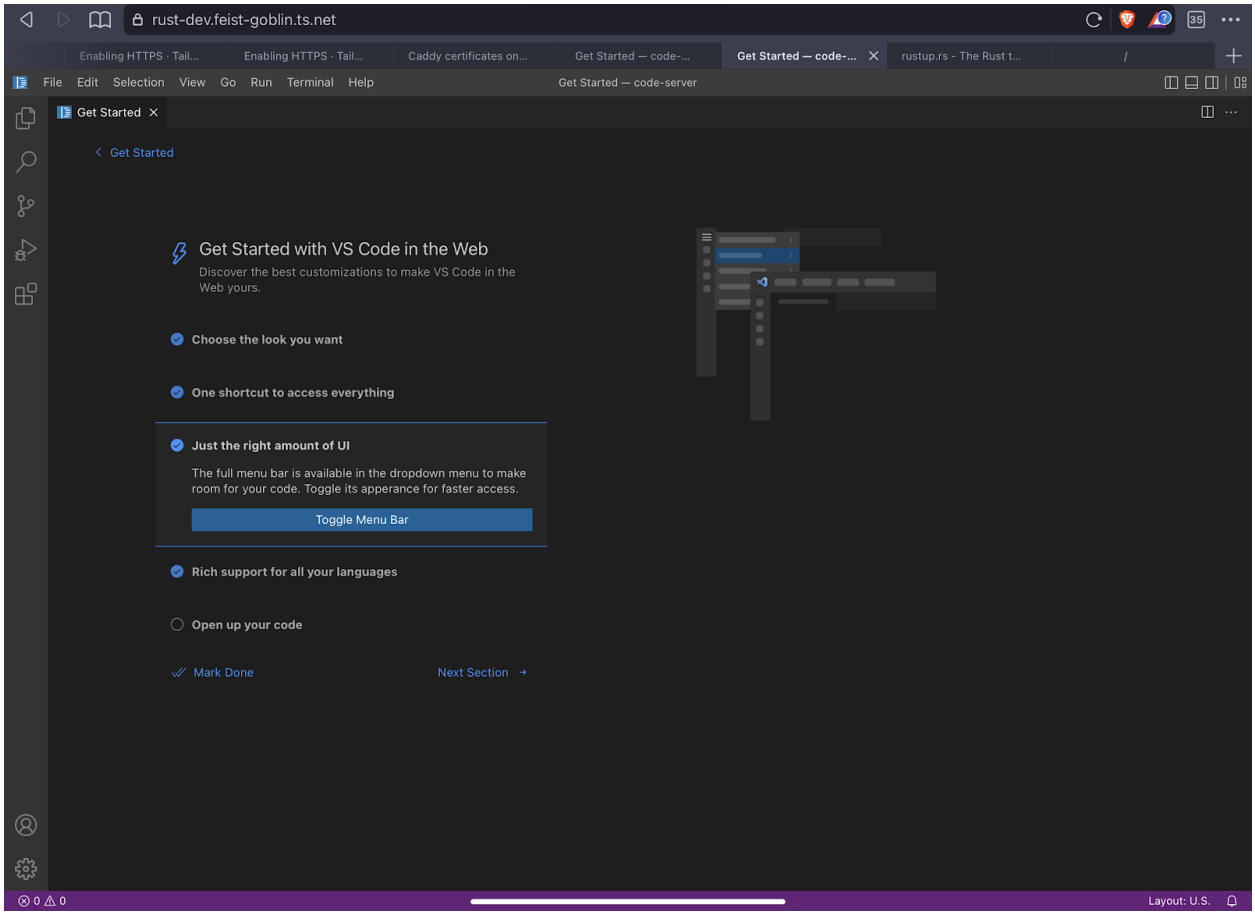
```
TS_DOMAIN_ALIAS=feist-goblin.ts.net
sed 's/^/# /' /etc/caddy/Caddyfile > /tmp/caddy
cat >> /tmp/caddy << EOF
${HOSTNAME}.${TS_DOMAIN_ALIAS} {
      handle * {
            reverse_proxy ${HOSTNAME}.${TS_DOMAIN_ALIAS}:8080
      }
      handle_path /code* {
            root * $HOME
            file_server browse
      }
}
EOF
sudo cp /tmp/caddy /etc/cadd/Caddyfile
sudo service caddy restart
```

At this point, visiting https://$HOSTNAME.$TS_DOMAIN_ALIAS will launch the code-server front end with full functionality. The SSL certificate is automatically generated and works seamlessly.

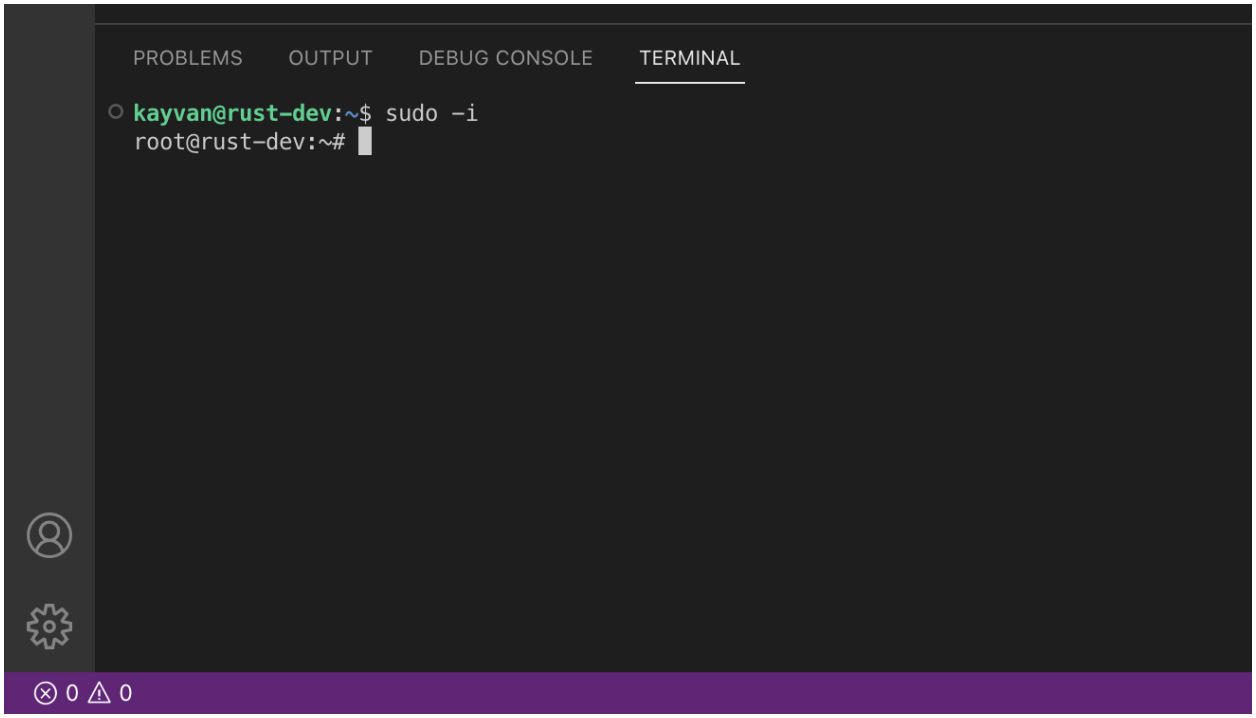
And, visiting the same URL with the “/code/“ subdirectory will show the file browser for the home directory, making it possible to navigate to any generated local files.

Final Steps

The resulting VSCode web application looks like this; a full featured IDE suitable for coding in the programming language of your choice.



In addition, using the Terminal in VSCode, we have full access to the machine:



Here, we can install, for example, “rustup” for the Rust tool chain, or any other programming language, along with the required VSCode extensions.

Rust Toolchain

Having set up the code-server, start a Terminal and install “rustup” following the instructions here: https://rustup.rs/

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

When rustup is installed, set up the bash completions for rustup and for cargo.

```
mkdir -p ~/.local/share/bash-completion/completions
rustup completions bash >>
~/.local/share/bash-completion/completions/rustup
rustup completions bash cargo >>
~/.local/share/bash-completion/completions/cargo
```

Also, install git:

```
sudo apt install git
```

VSCode Extensions for Rust

The following set of extensions are recommended for Rust development:

- rust-analyzer
- tabnine - AI auto-complete
- Better TOML - For Cargo.toml support
- Search crates.io - crates suggestions
- Better Comments
- Crates
- CodeLLB - for debugging
- markdownlint

References

- Setting up Tailscale on Linux
- Coding on iPad using VSCode, Caddy, and code-server
- Caddy certificates on Tailscale
- Install — Caddy on Debian/Ubuntu
- Rustup Installer