Guide to install Docker

Installation

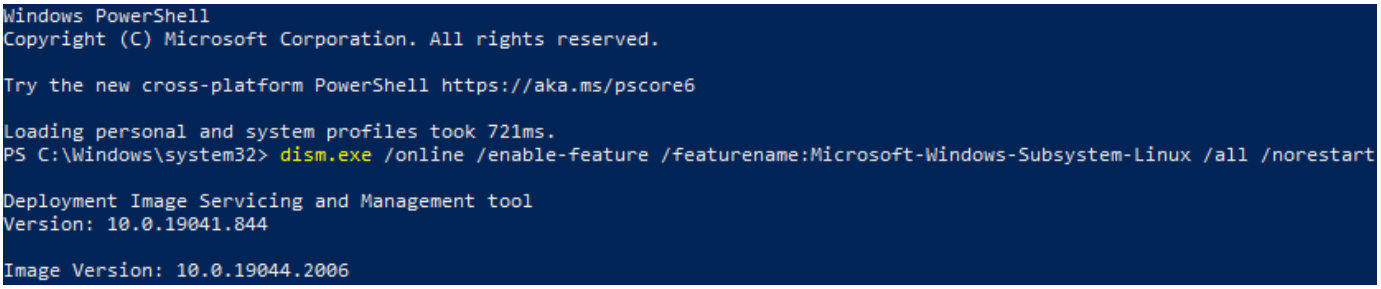
Windows

We highly recommend running Docker from WSL (Windows Subsystem for Linux) to avoid performance issues. Follow the links below for additional information.

Enable WSL

In some cases WSL is not enabled to enable it in PowerShell run this command.

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```




```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

```
PS C:\Windows\system32> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Deployment Image Servicing and Management tool
Version: 10.0.19041.844

Image Version: 10.0.19044.2006

Enabling feature(s)
[==========================100.0%==========================]
The operation completed successfully.
PS C:\Windows\system32>
```

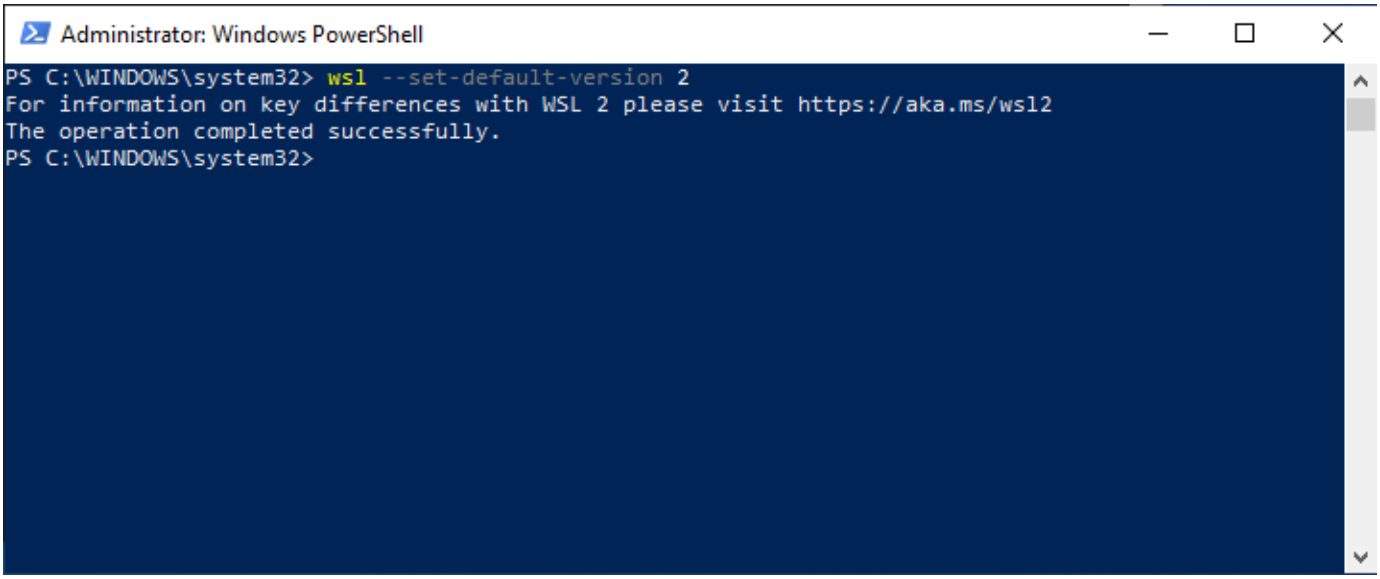
```
Enable-WindowsOptionalFeature -Online -FeatureName
VirtualMachinePlatform -NoRestart
```

```
PS C:\Windows\system32> Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform -NoRestart
WARNING: Restart is suppressed because NoRestart is specified.


Path          :
Online        : True
RestartNeeded : True
```

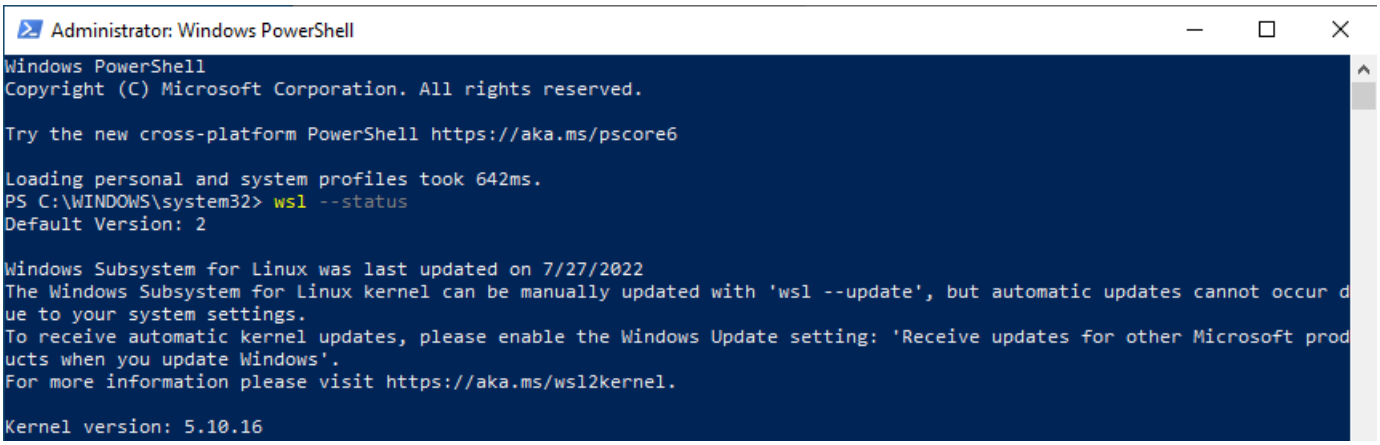
Now restart your entire laptop.

Install WSL

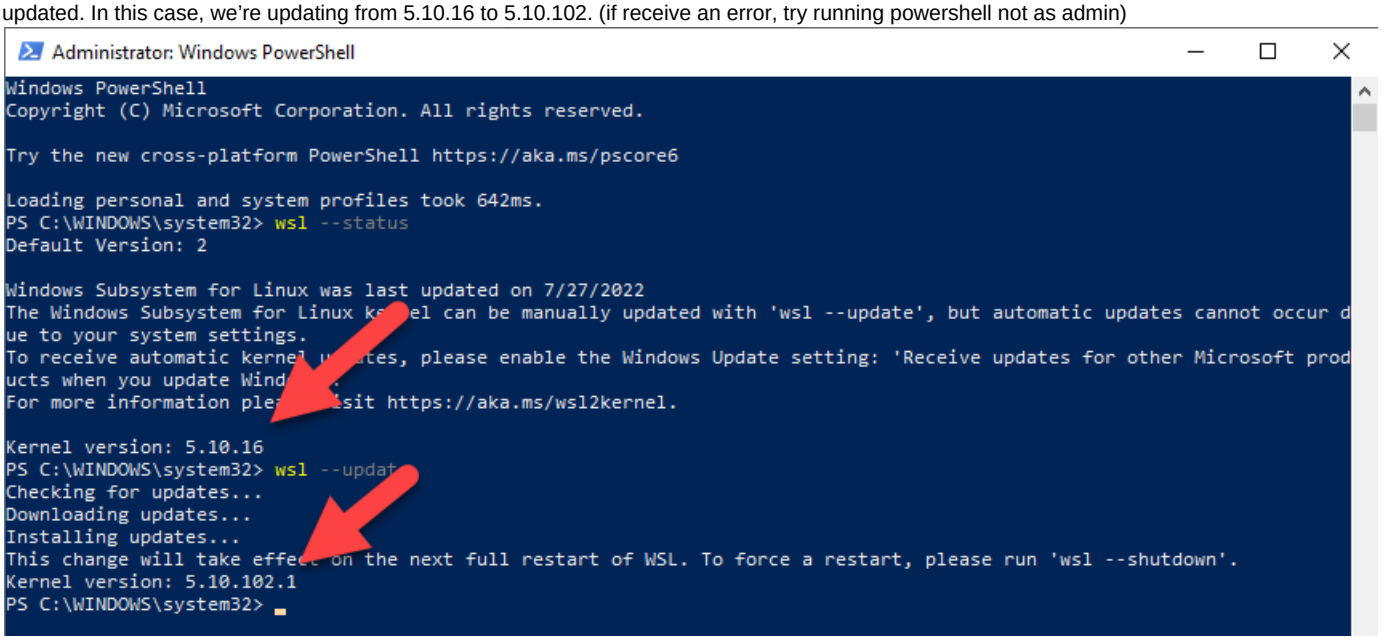
Some Windows versions have WSL 1 instead of WSL 2. Run the following command in PowerShell to switch to WSL 2: `wsl --set-default-version 2`.



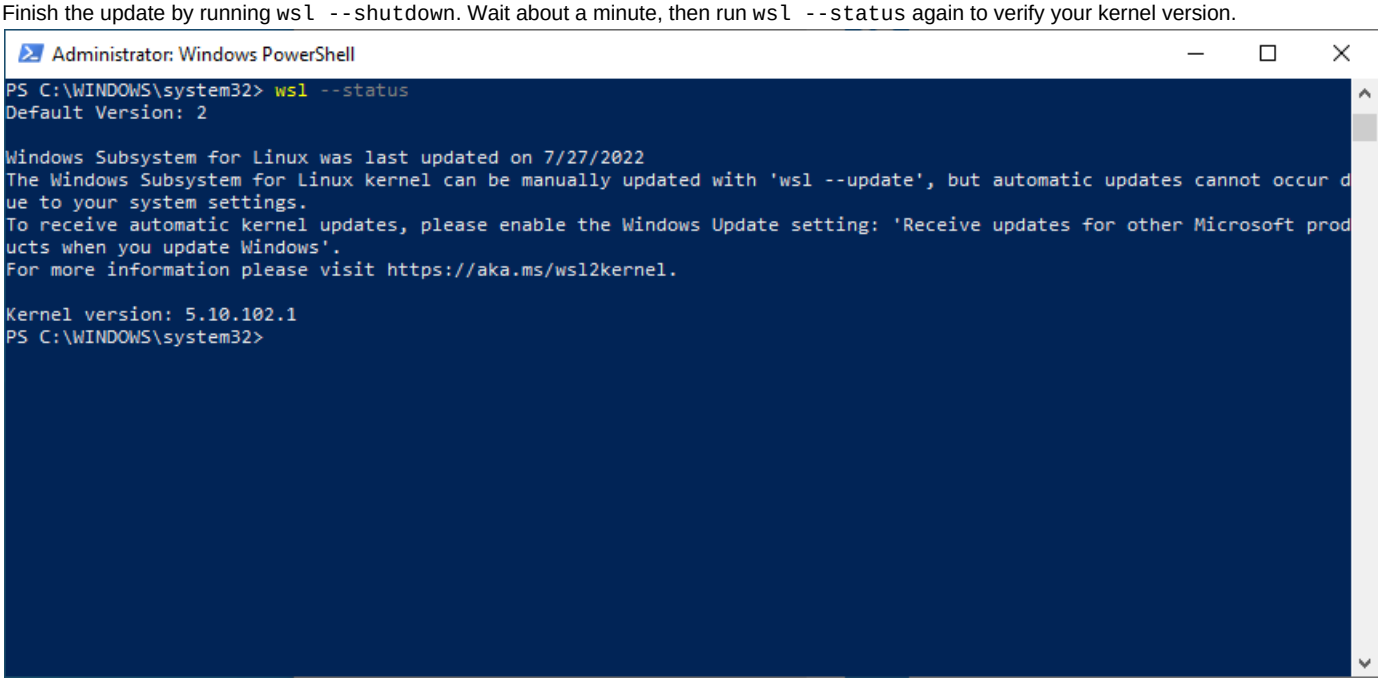
Confirm your version by running `wsl --status`. You should see “Default Version: 2.”



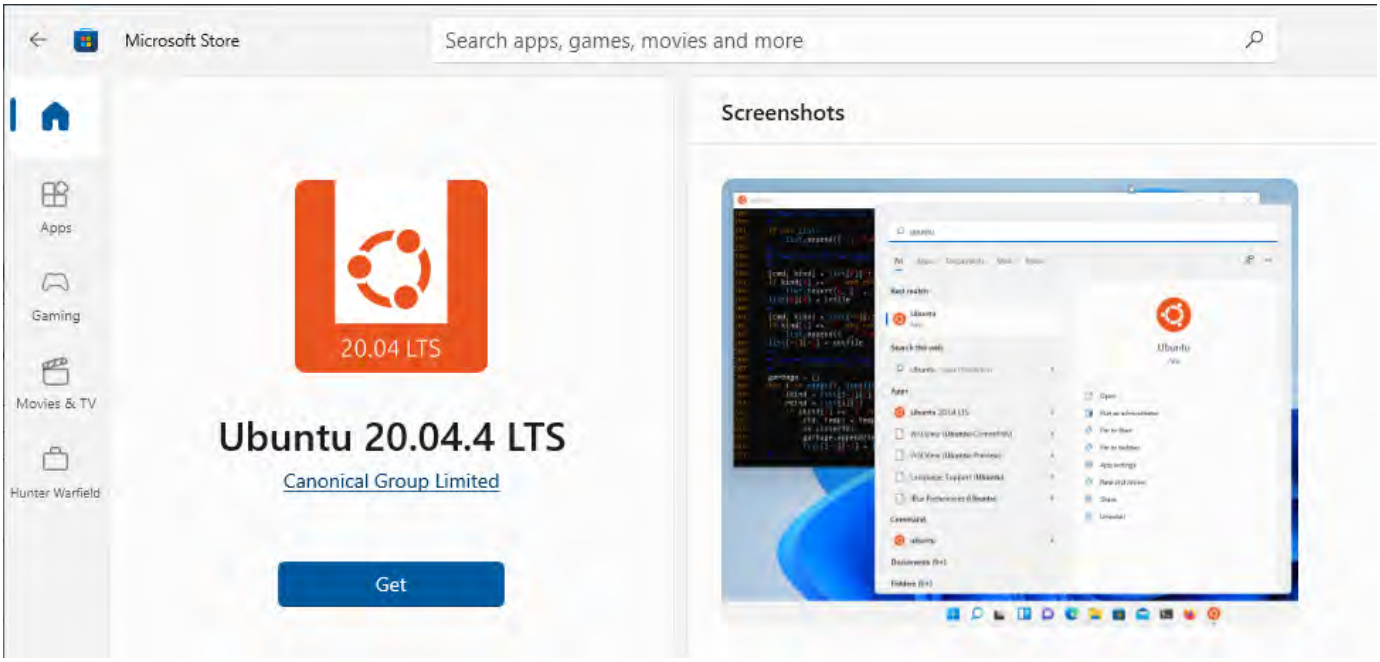
Update your kernel version by running `wsl --update`. This may take a few minutes. Once complete, you should see the new version to be updated. In this case, we're updating from 5.10.16 to 5.10.102. (if receive an error, try running powershell not as admin)



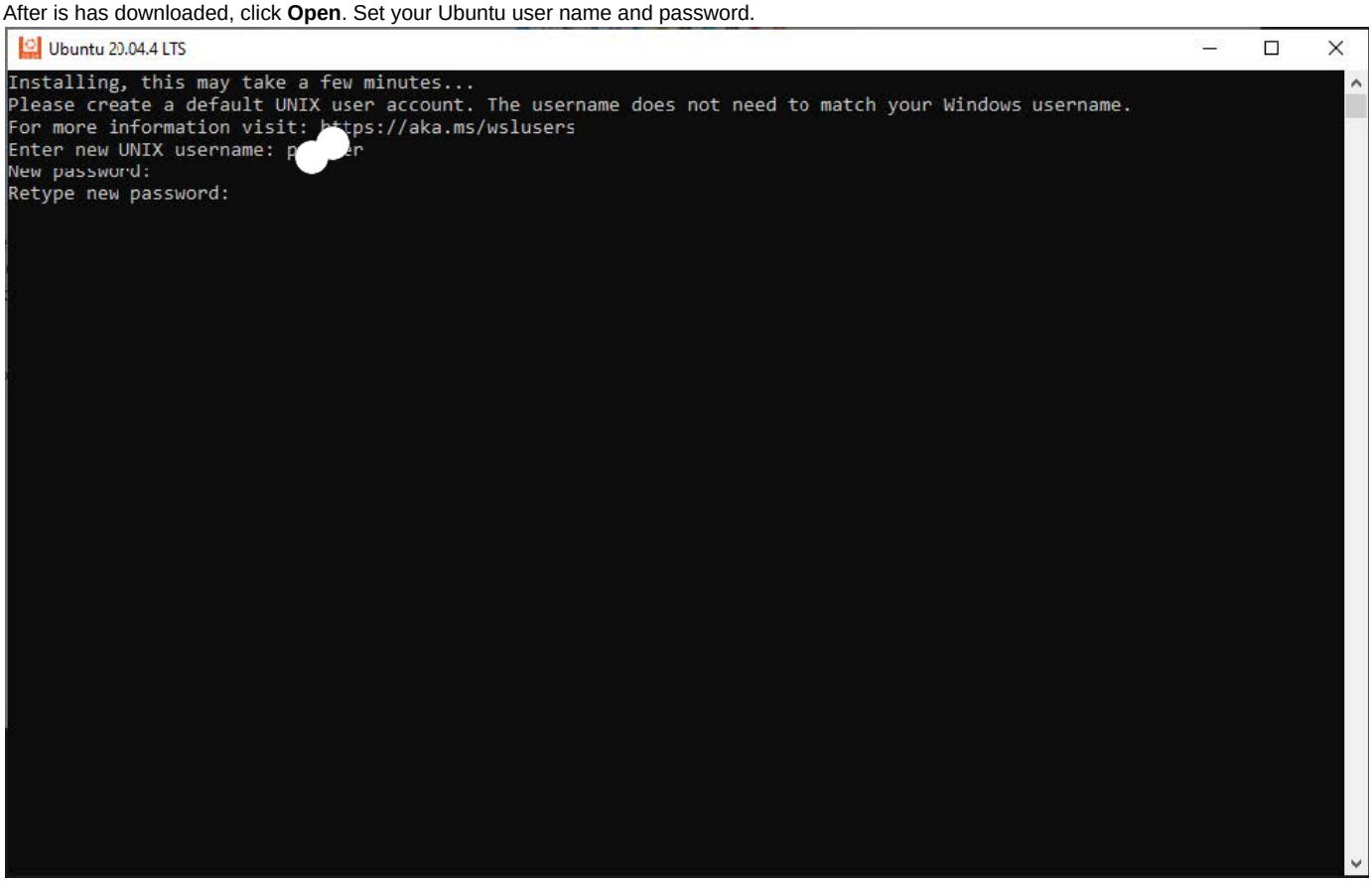
Finish the update by running `wsl --shutdown`. Wait about a minute, then run `wsl --status` again to verify your kernel version.



Download the latest Ubuntu LTS version from the Windows store.



After is has downloaded, click **Open**. Set your Ubuntu user name and password.



Run `sudo apt-get update` to apply any pending updates.

Install Docker and Docker Desktop on your Windows machine

Follow this guide: Install Docker Desktop on Windows.

Here's a walk-through of Step 5 (adding your user to the `docker-users` group), since it differs a bit for those of us using Resident Interface laptops.

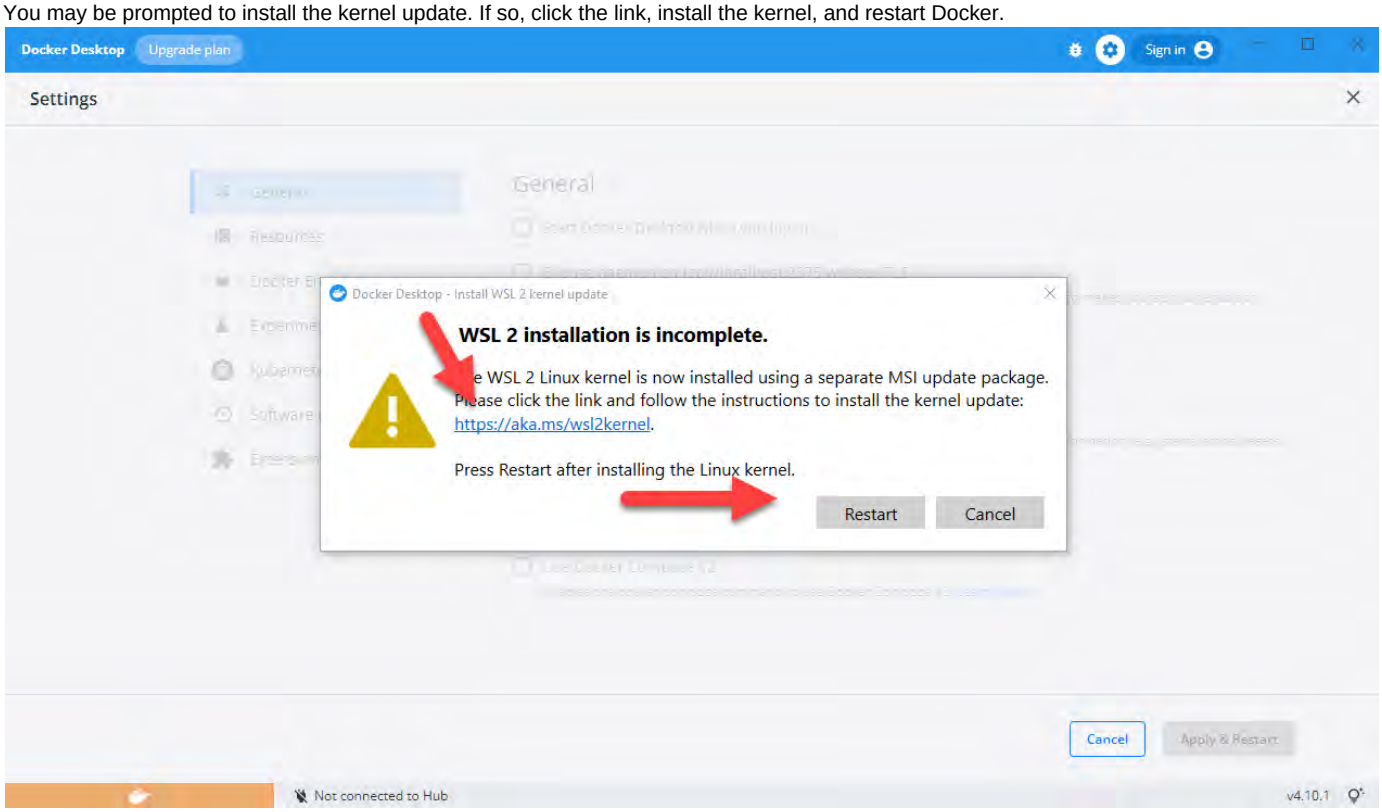
Click **OK** again, then close Computer Management. Log out of Windows and back in for the permissions to take effect

Activate the WSL Integration

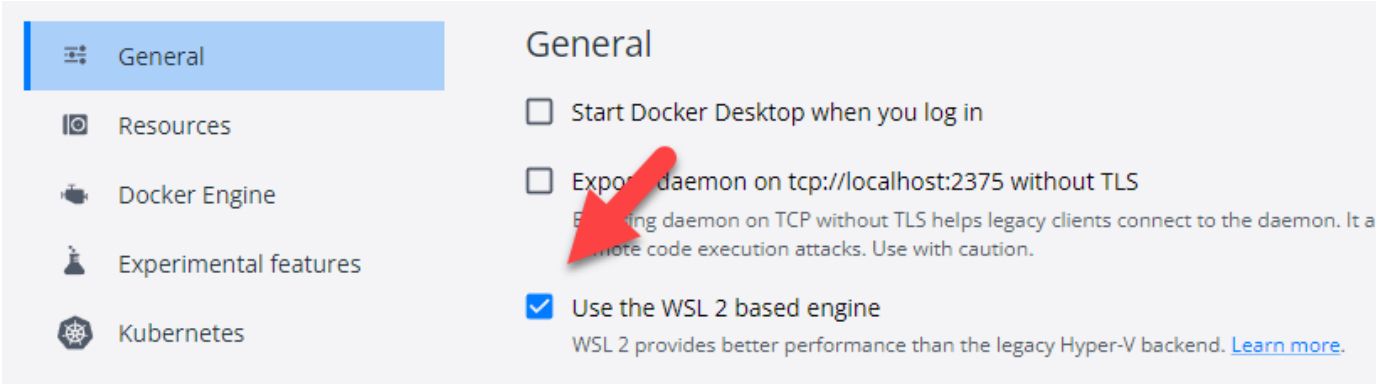
Start Docker Desktop from the Windows Start menu.

From the Docker menu, select **Settings > General**.

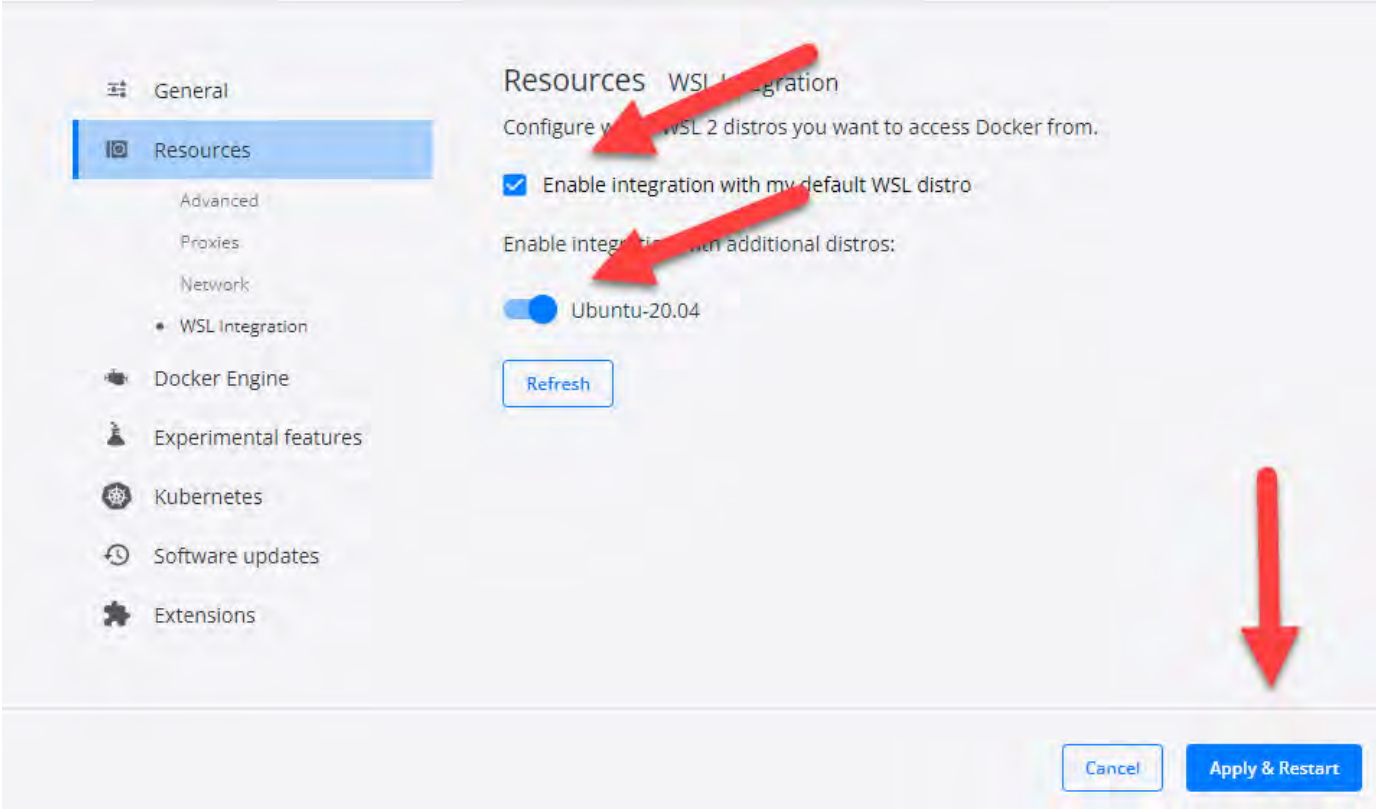
You may be prompted to install the kernel update. If so, click the link, install the kernel, and restart Docker.



Click **Settings > General** and verify “Use the WSL 2 based engine” is checked



Next, click **Resources > WSL Integration** and verify “Enable integration with my default WSL distro” is checked. Select the Linux distribution you installed, then click **Apply & Restart**.

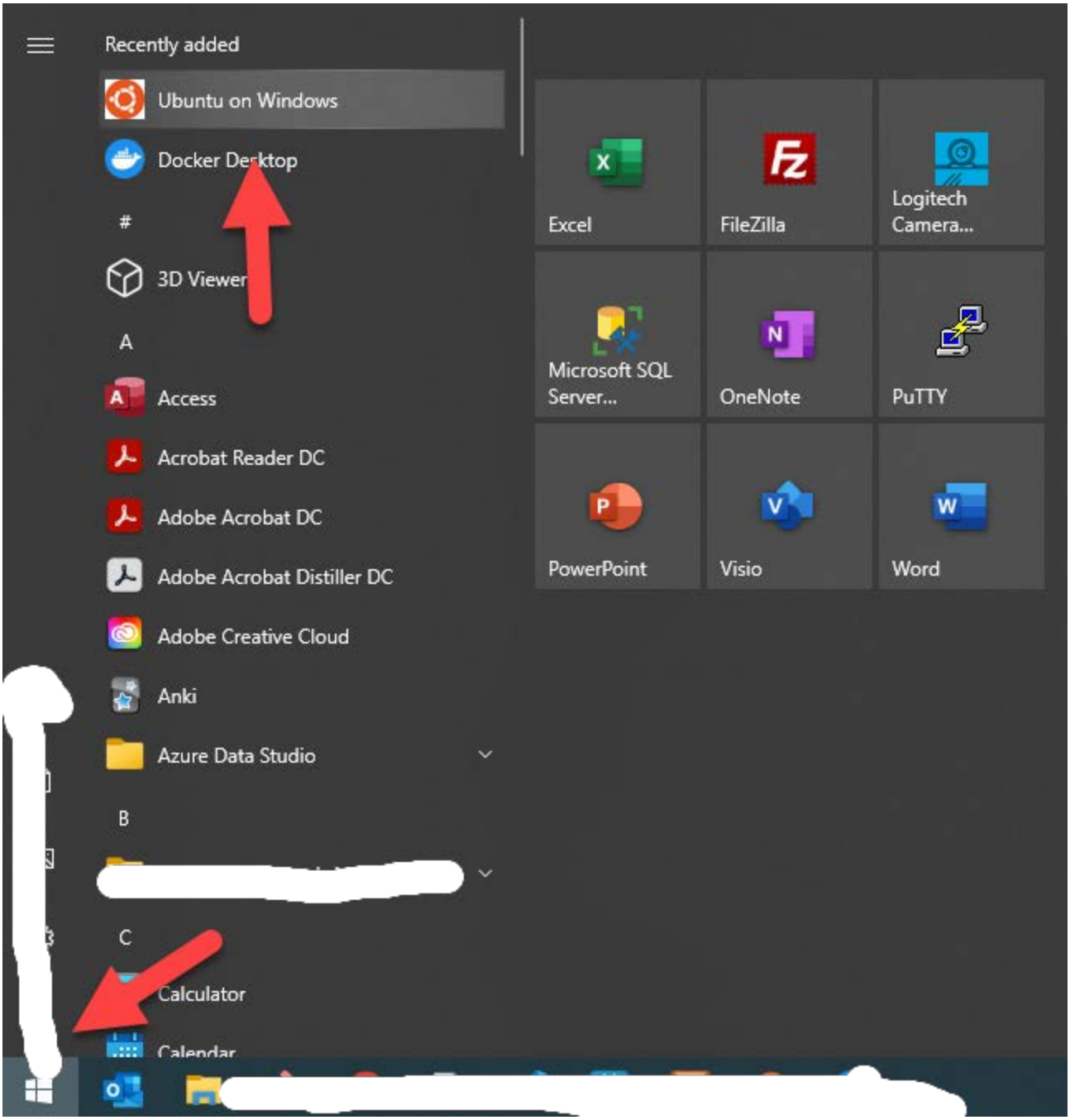


Configuration

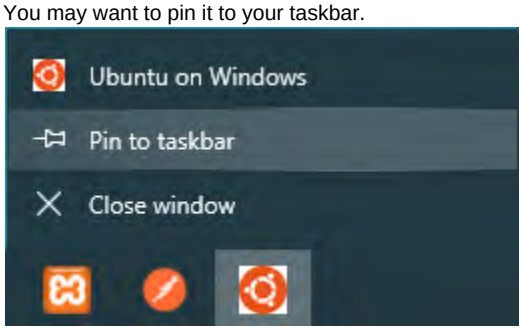
Windows

For performance reasons, you’ll want to store your local copies of repositories in Linux, rather than Windows.

To open your WSL instance, click **Start**, then **Ubuntu on Windows**.



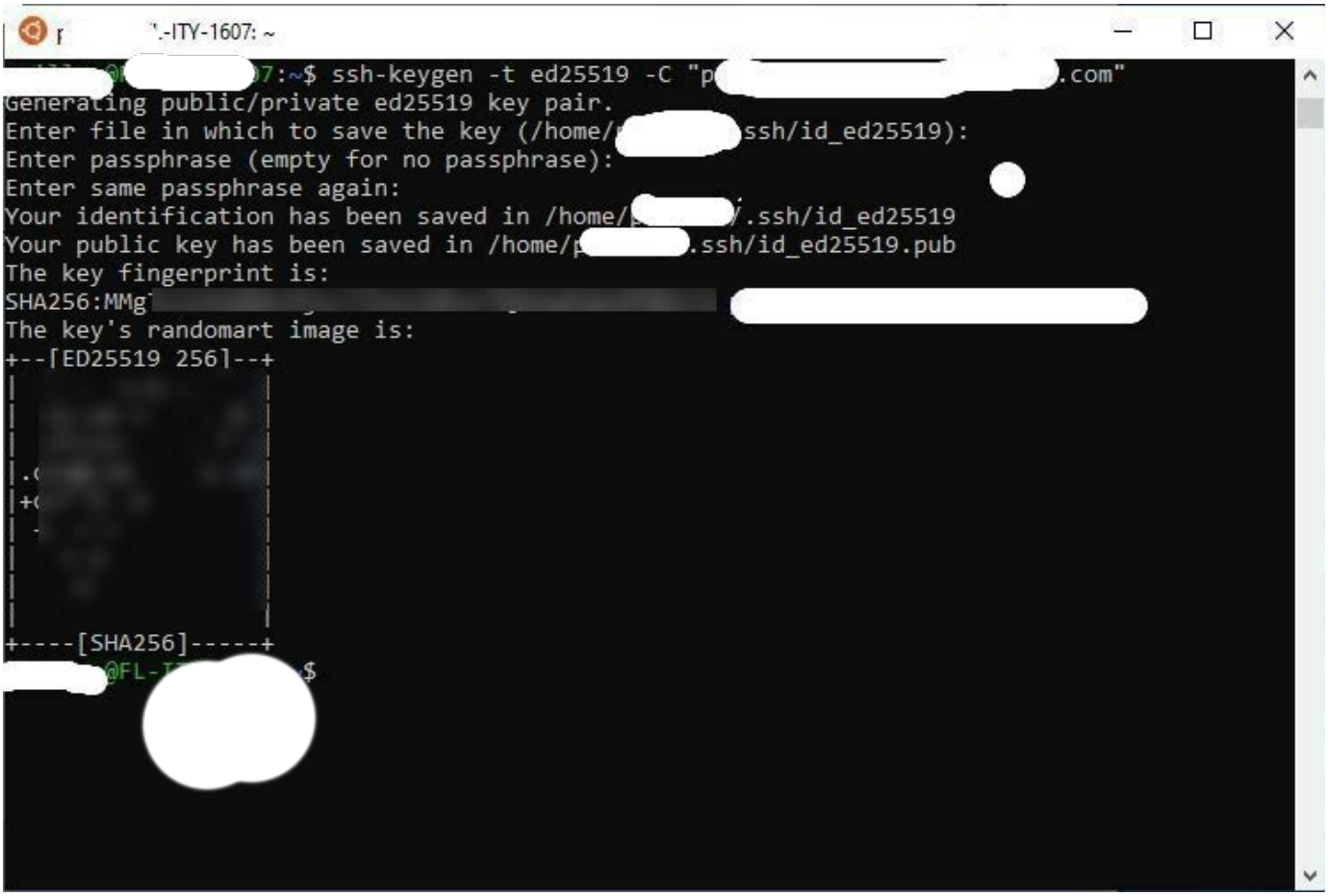
You may want to pin it to your taskbar.



Generate a New SSH Key

Source: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Run `ssh-keygen -t ed25519 -C "youremail@company.com"` to generate a new key, substituting your email address. Accept or alter the options as you see fit, although we ask that you use a passphrase. When you're done, the output should look something like this:



Add Your Key to the SSH Agent

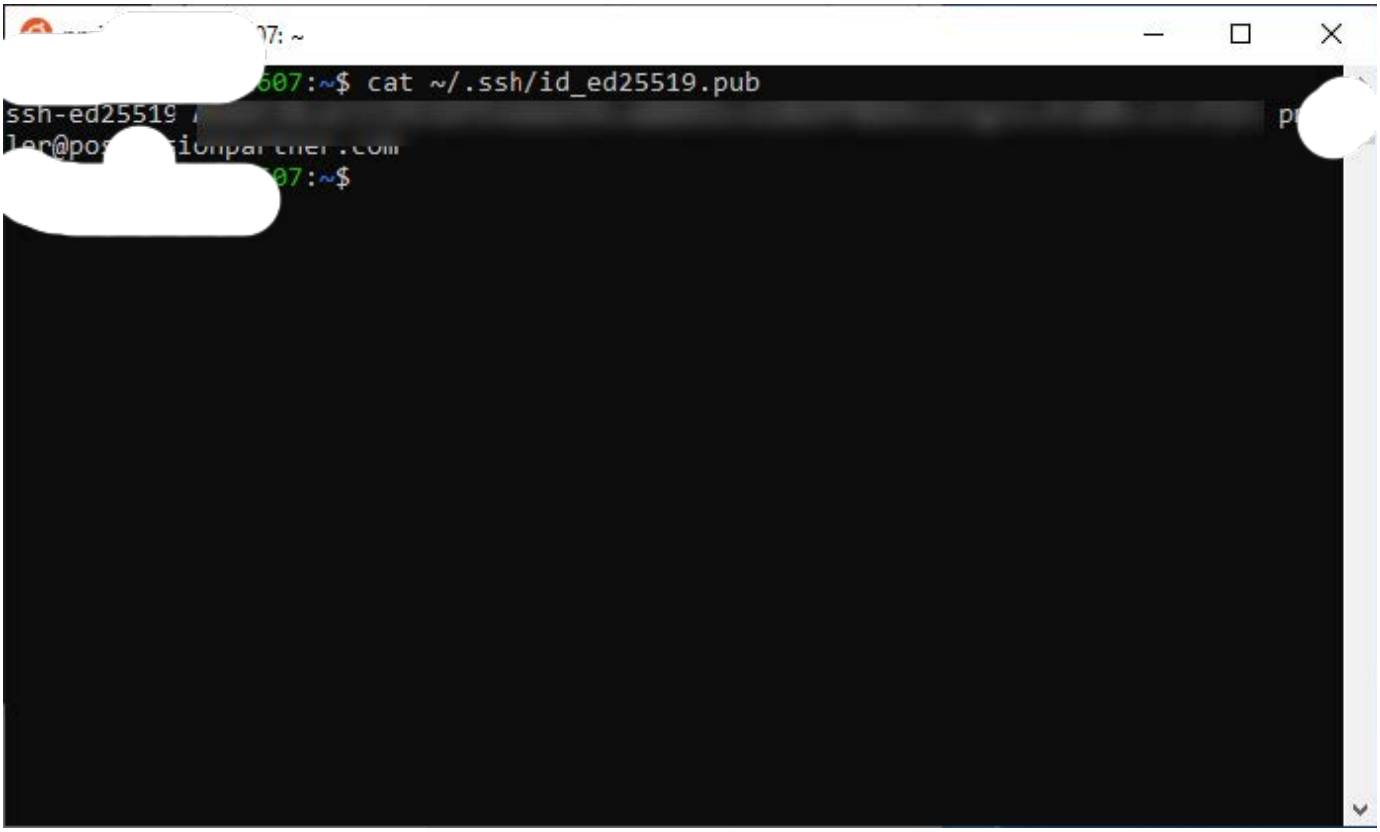
Start the SSH agent by running `eval "$(ssh-agent -s)"`.

Add your SSH private key to the ssh-agent by running `ssh-add ~/.ssh/id_ed25519`. If you created your key with a different name, or if you are adding an existing key that has a different name, replace id_ed25519 in the command with the name of your private key file.

Add Your Key to GitHub

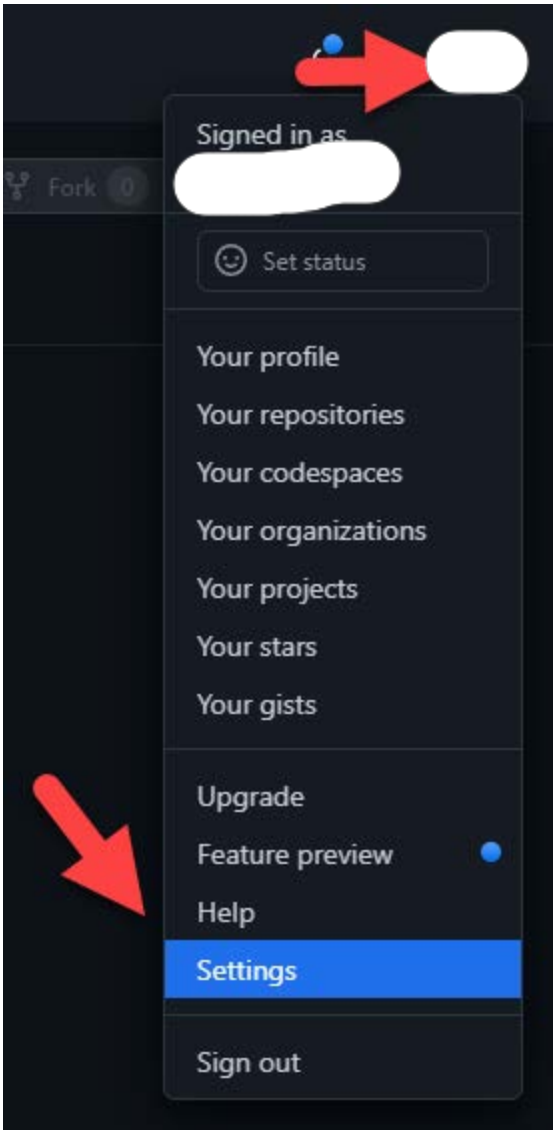
Source: https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

Run `cat ~/.ssh/id_ed25519.pub` to display your key. Copy the contents to the clipboard.

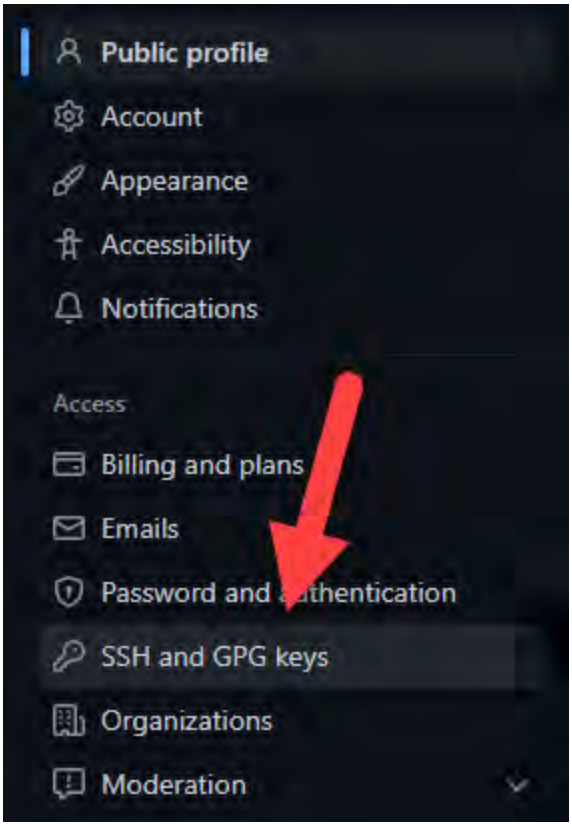


Log into your GitHub account.

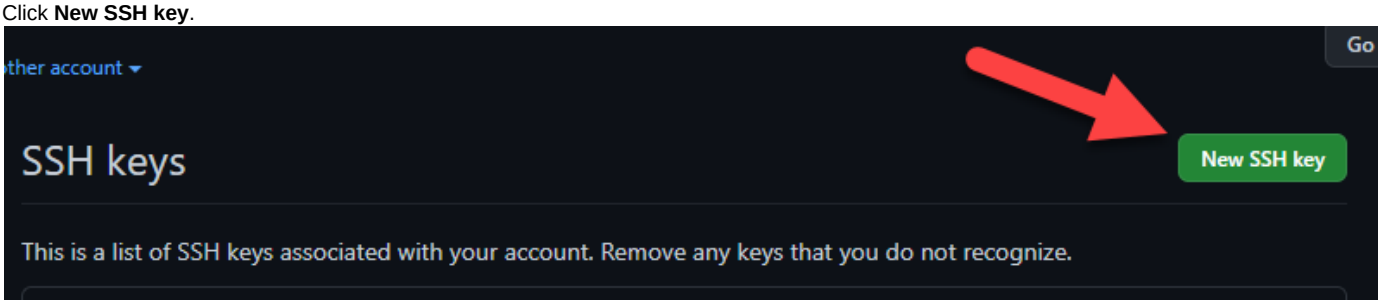
Click your icon, then click **Settings**.



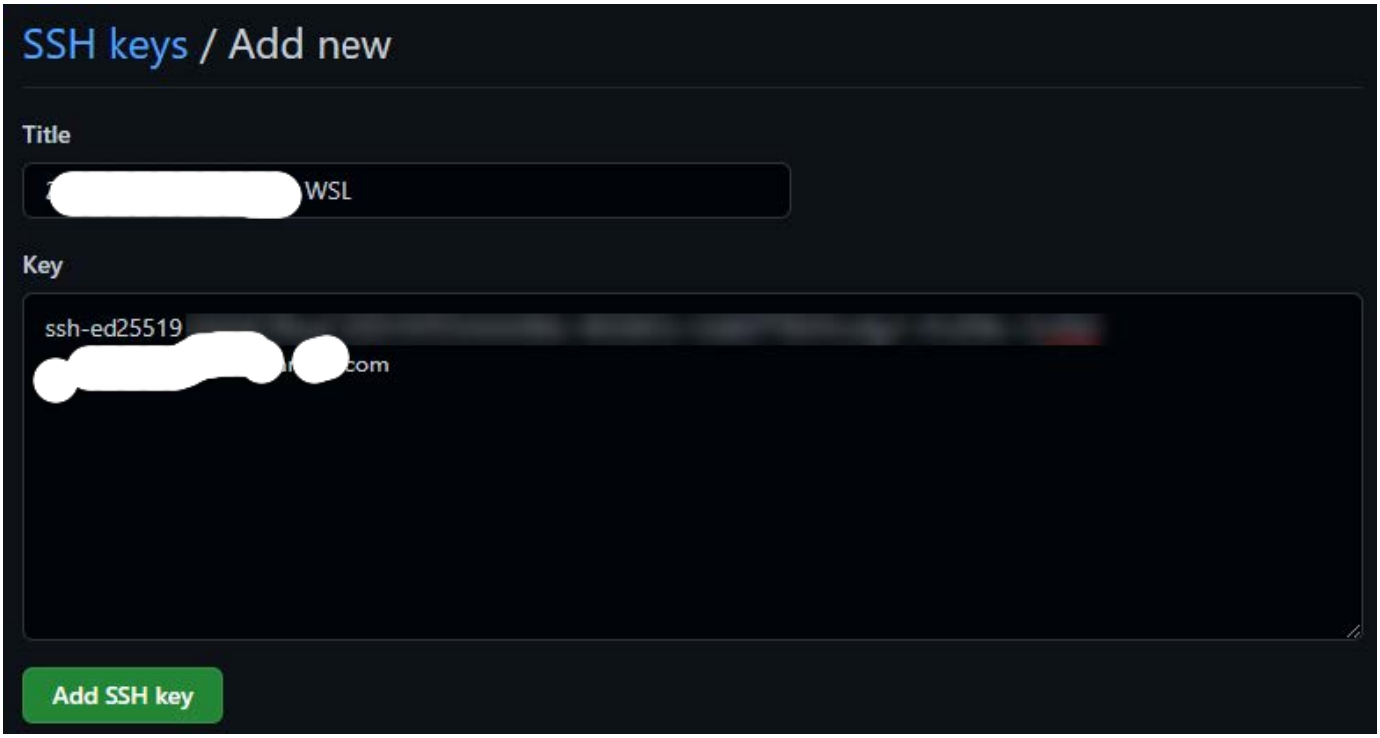
In the "Access" section of the sidebar, click **SSH and GPG keys**.



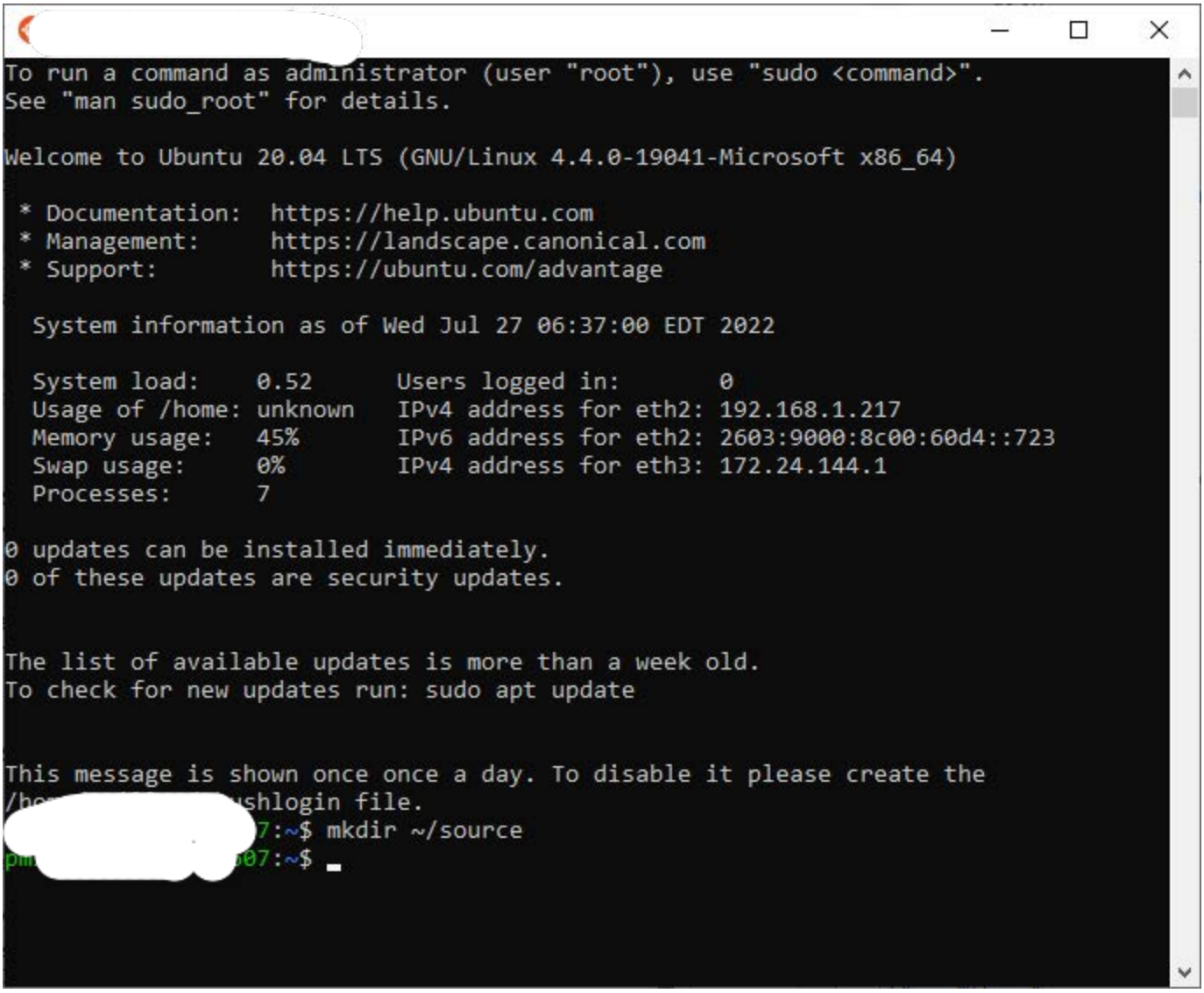
Click **New SSH key**.



In the "Title" field, add a descriptive label for the new key. Paste your key into the "Key" field, then click **Add SSH key**.

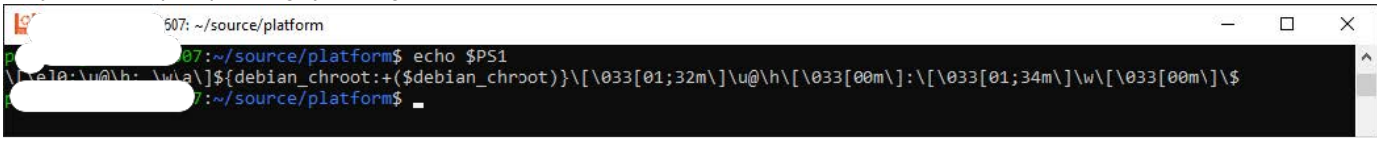


Create a directory to store your source code. In this case, we're creating a directory named `source` in our user's home (~) directory by running `mkdir ~/source`:



Display Your Branch Name in the Terminal

Get your current prompt setting by running `echo $PS1`.

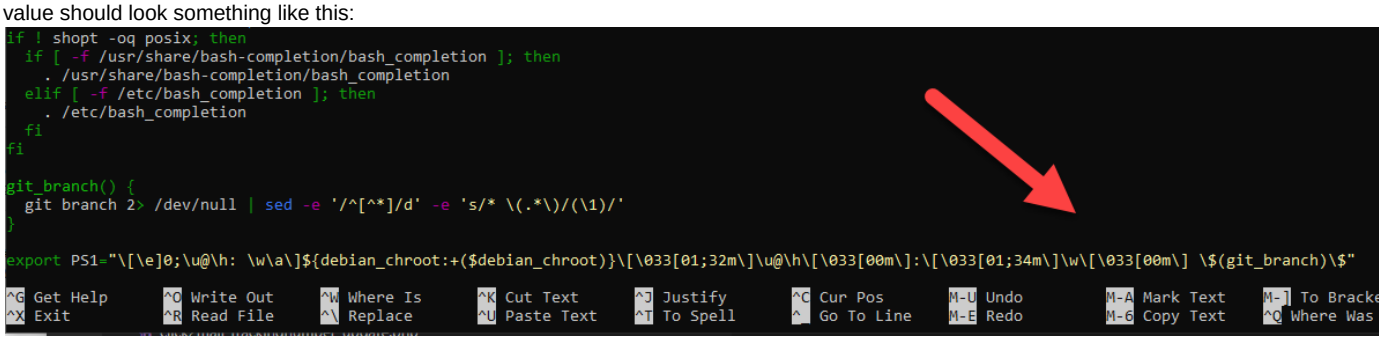


Copy the value to your clipboard.

Execute `nano ~/.bashrc` to open your bash configuration file. Add the following function at the bottom of the file:

```
git_branch() {
  git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/(\1)/'
}
```

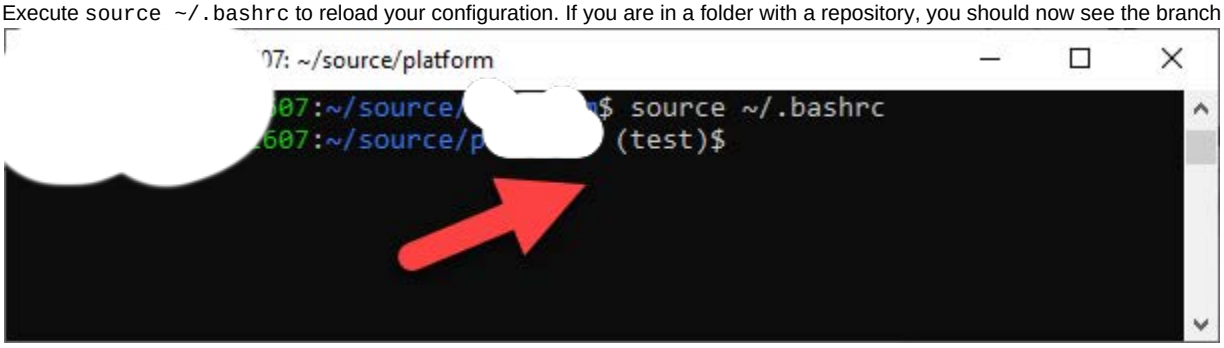
Next, create a line that begins with `export PS1="`. Paste the value of $PS1 you copied earlier. Add `(git_branch)\$"` to the end. Your final value should look something like this:



You'll see that I added a space between `[\00m\]` and `\$` to make the display a little easier to read.

Save your changes and close the file.

Execute `source ~/.bashrc` to reload your configuration. If you are in a folder with a repository, you should now see the branch name.



(**Optional)

Step 1 [PassphareAsking] How to stop git asking you for passphrase

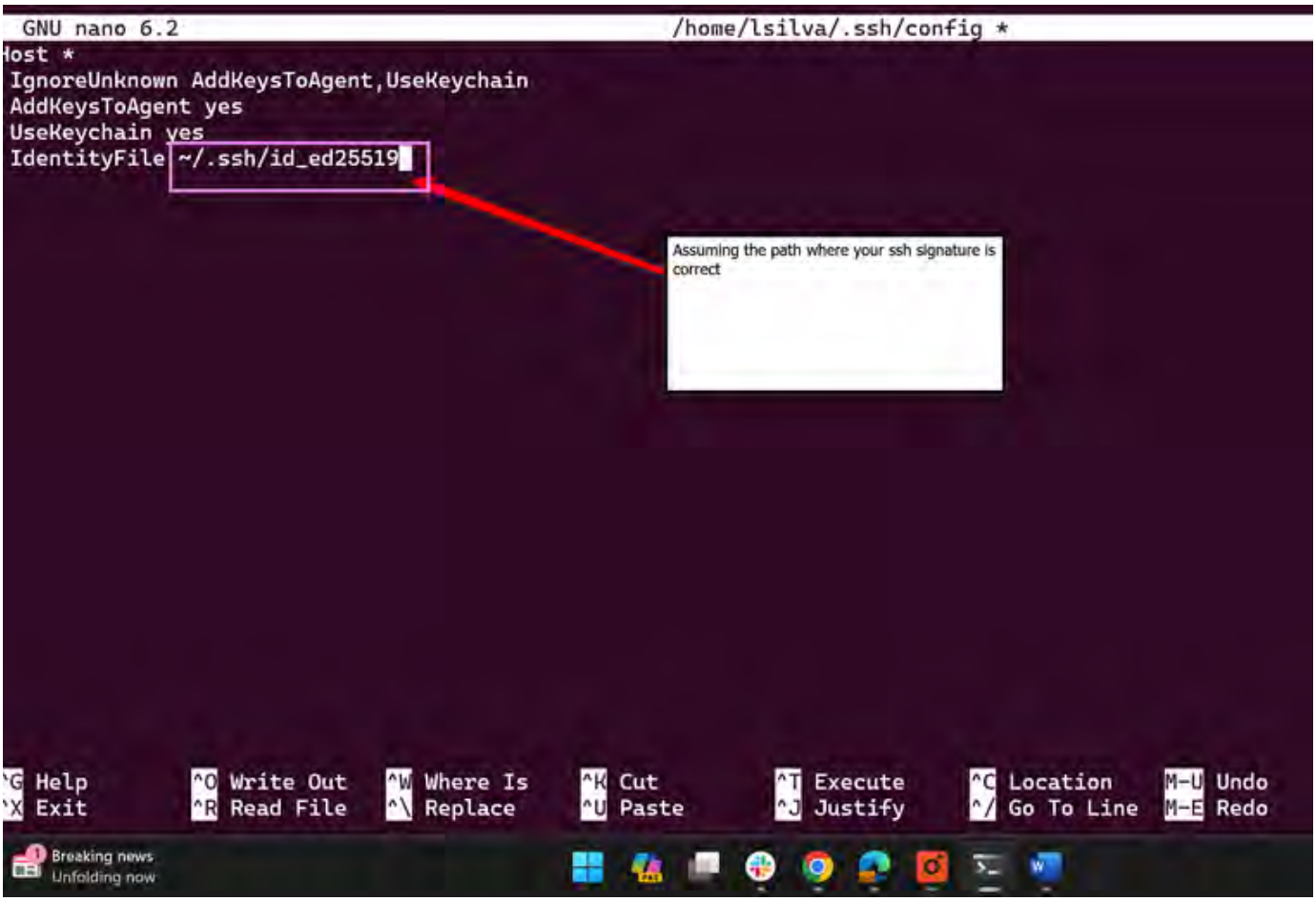
1. Star ubuntu shell
2. Edit your `nano ~/.bashrc` file
3. Add the following lines to the file at the end

```
SSH_ENV=$HOME/.ssh/environment

# start the ssh-agent
function start_agent {
    echo "Initializing new SSH agent..."
    # spawn ssh-agent
    /usr/bin/ssh-agent | sed 's/^echo/#echo/' > ${SSH_ENV}
    echo succeeded
    chmod 600 ${SSH_ENV}
    . ${SSH_ENV} > /dev/null
    /usr/bin/ssh-add
}

if [ -f "${SSH_ENV}" ]; then
     . ${SSH_ENV} > /dev/null
     ps -ef | grep ${SSH_AGENT_PID} | grep ssh-agent$ > /dev/null || {
        start_agent;
    }
else
    start_agent;
fi
```

Step 2 [PassphareAsking]

Sometimes is a little annoying to keep linux asking you for passphrase, in the scenario you like to entery your credentials once you may open the file called `nano ~/.ssh/config` and then add the following lines

```
Host *
  IgnoreUnknown AddKeysToAgent,UseKeychain
  AddKeysToAgent yes
  UseKeychain yes
  IdentityFile ~/.ssh/id_ed25519
```



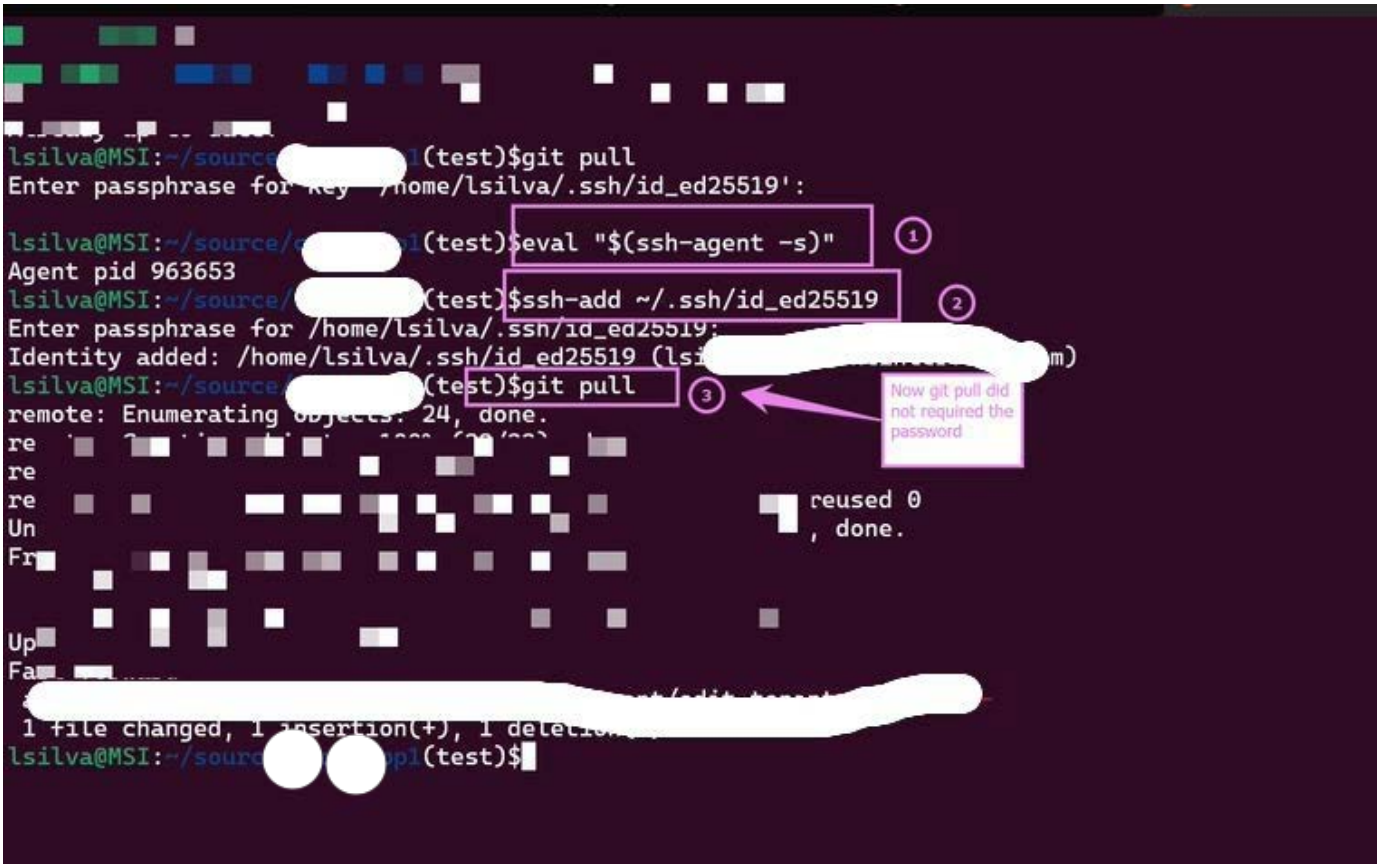
Once you save the file, run one more time the command to tell git our signature.

```
eval "$(ssh-agent -s)"
```

```
ssh-add ~/.ssh/id_ed25519
```

Check if the key is added (parameter is a lowercase L):

```
$ ssh-add -l
```



then **restart** your terminal and **passphrase** will be asked just once!. Enjoy it!