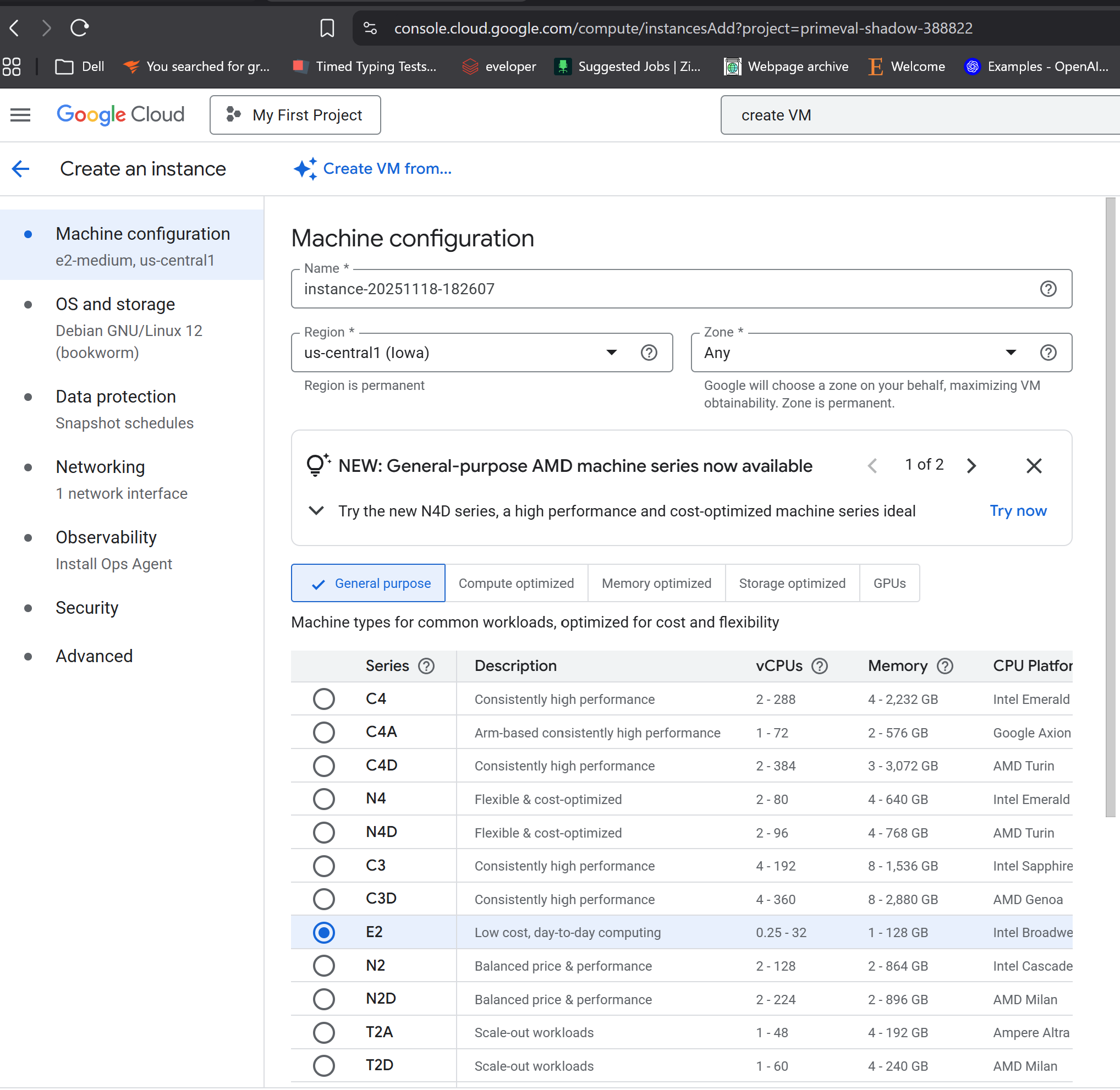
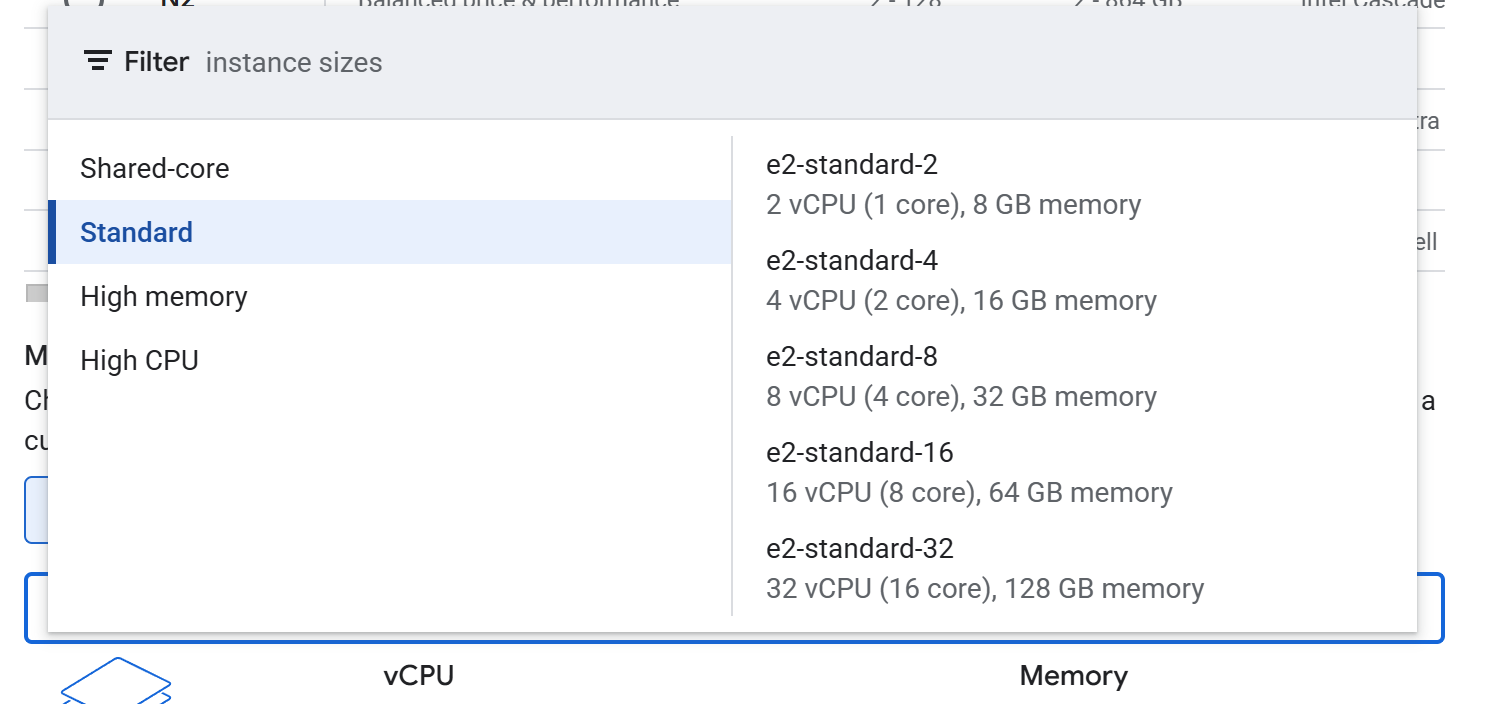


Click on create instance.



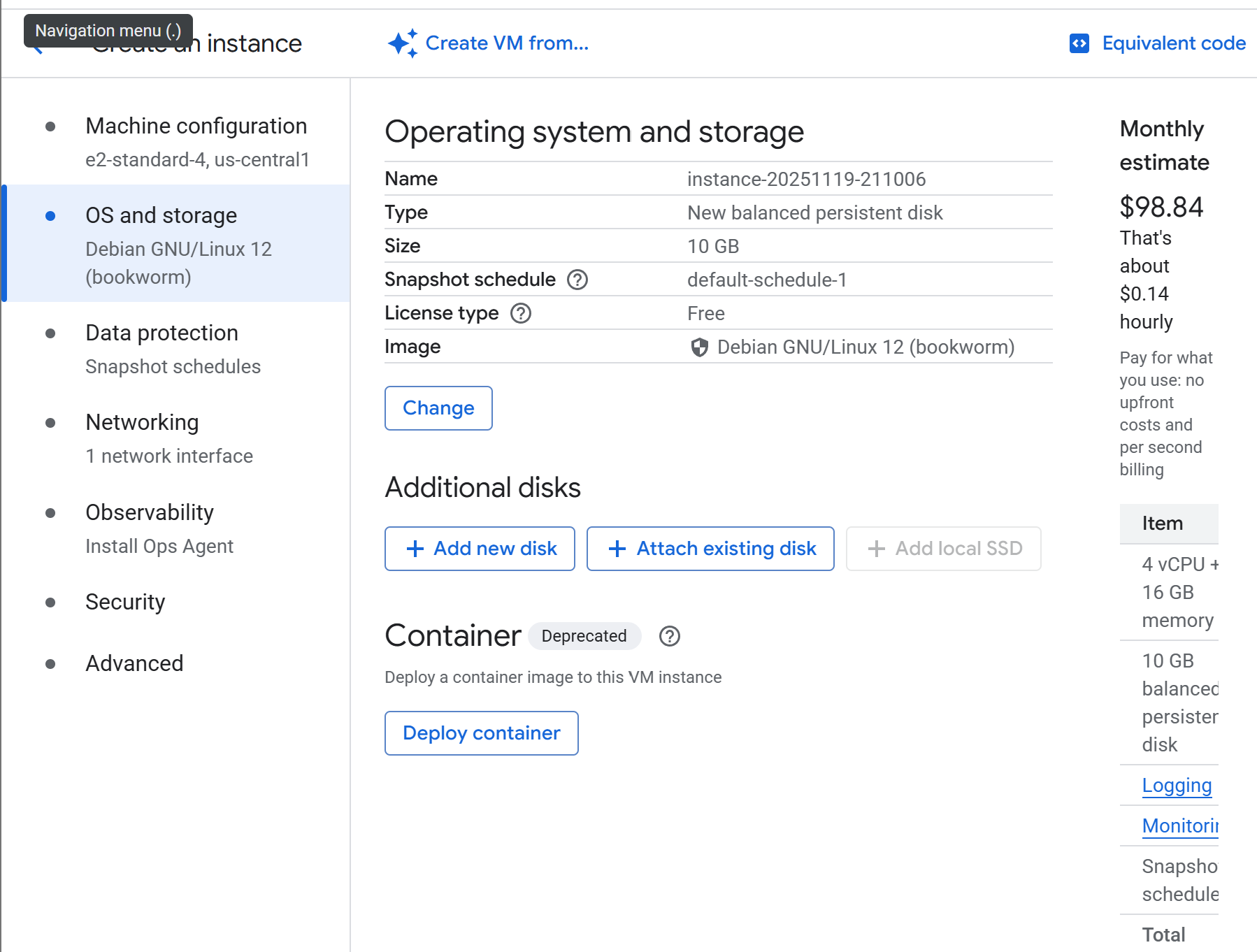
Select E2 here and not other workload

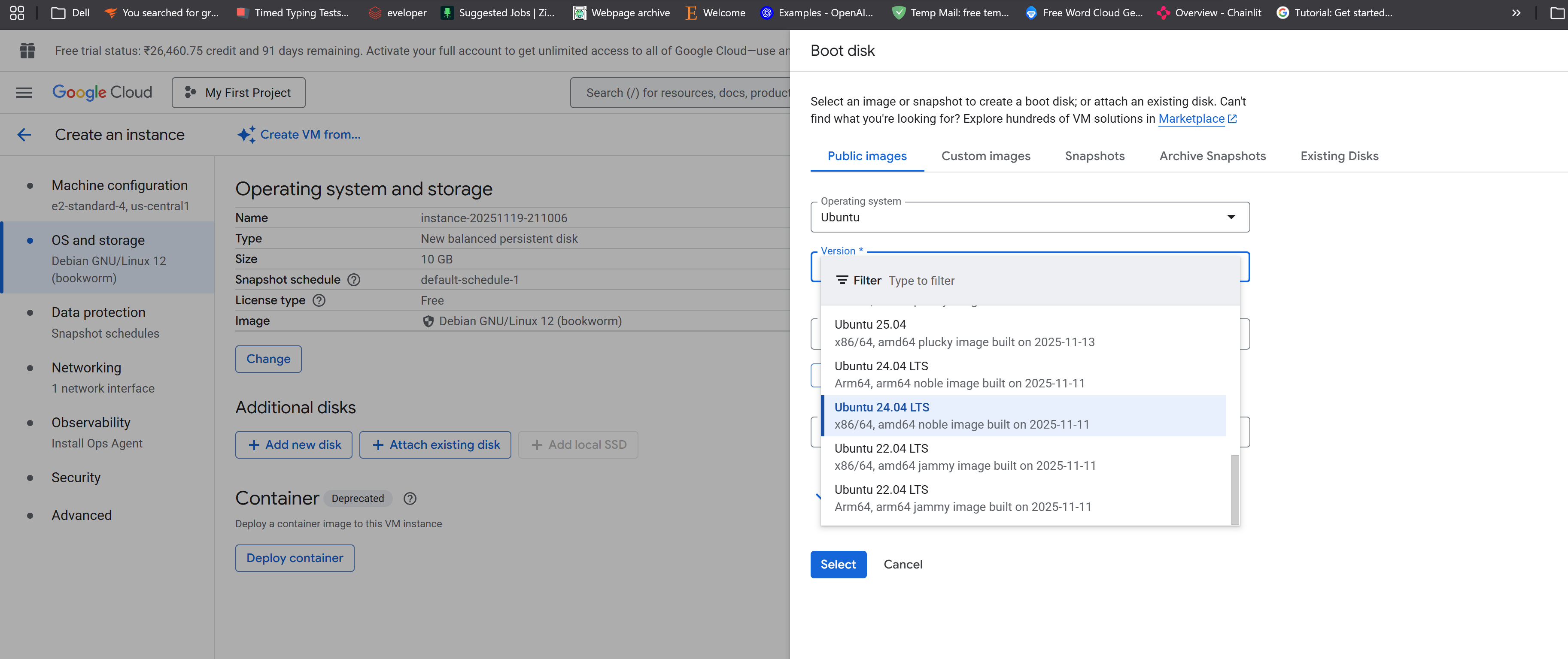
Select 16gb in machine type

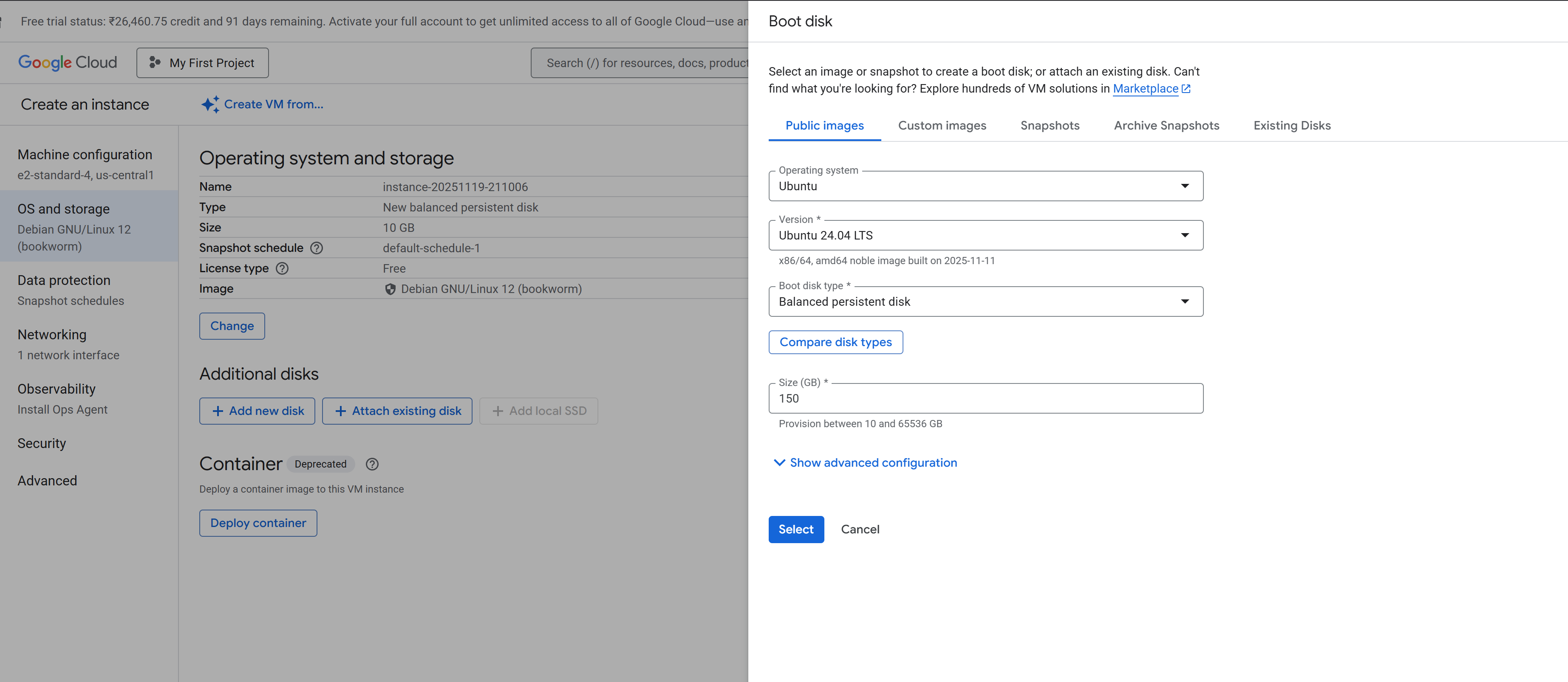


4vCPU and 16GB memory

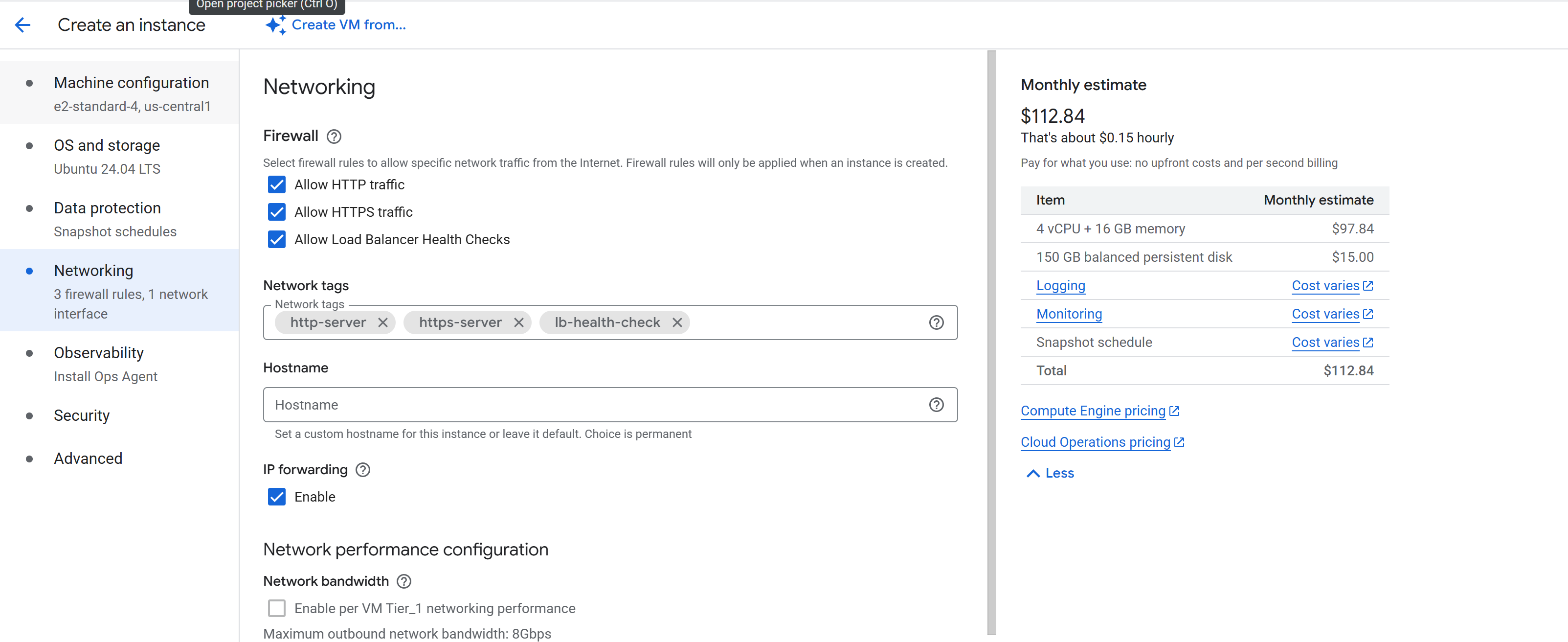
Now go to OS and Storage and Select ubuntu



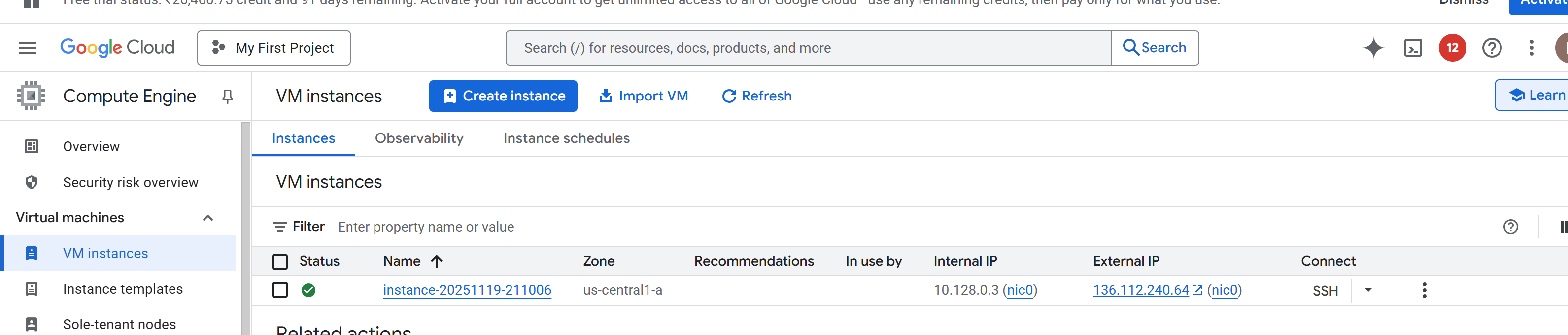




Now click on select.



Now click on create instance.



After clicking on SSH,

# Add Docker's official GPG key:

sudo apt update

sudo apt install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:

sudo tee /etc/apt/sources.list.d/docker.sources <<EOF

Types: deb

URIs: https://download.docker.com/linux/ubuntu

Suites: $(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}")

Components: stable

Signed-By: /etc/apt/keyrings/docker.asc

EOF

sudo apt update

copy paste this command on SSH.

Then copy and paste this command:

sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

then run this command to check whether docker has been installed or not?

sudo docker run hello-world

We don’t want to use sudo everytime. So, go to this page and

<https://docs.docker.com/engine/install/linux-postinstall/>

To create the **docker** group and add your user:

1. Create the **docker** group.

sudo groupadd docker

1. Add your user to the **docker** group.

sudo usermod -aG docker $USER

1. Log out and log back in so that your group membership is re-evaluated.

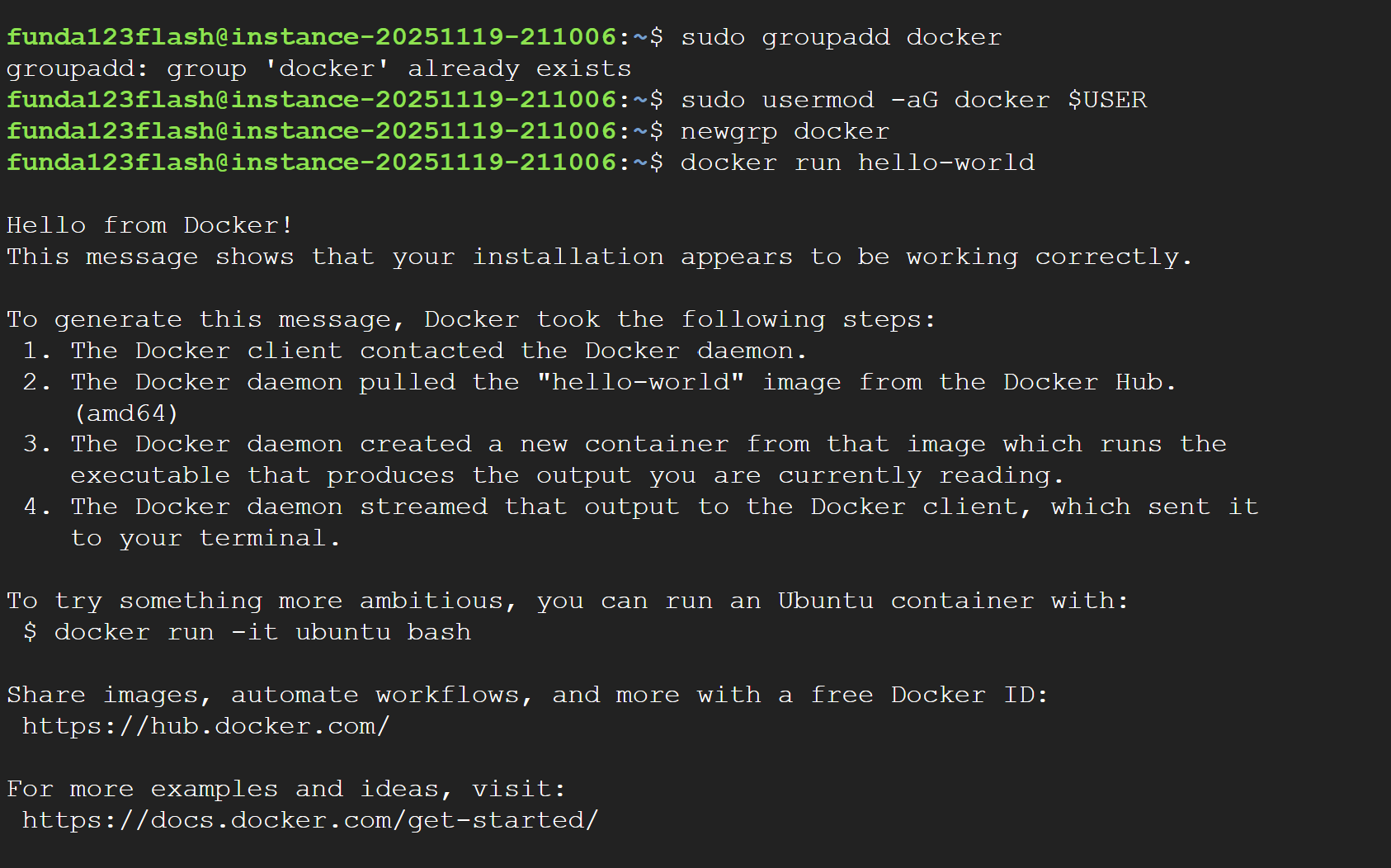
If you're running Linux in a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.

You can also run the following command to activate the changes to groups:

newgrp docker

1. Verify that you can run **docker** commands without **sudo**.

docker run hello-world



See, it is running smoothly without using sudo as expected

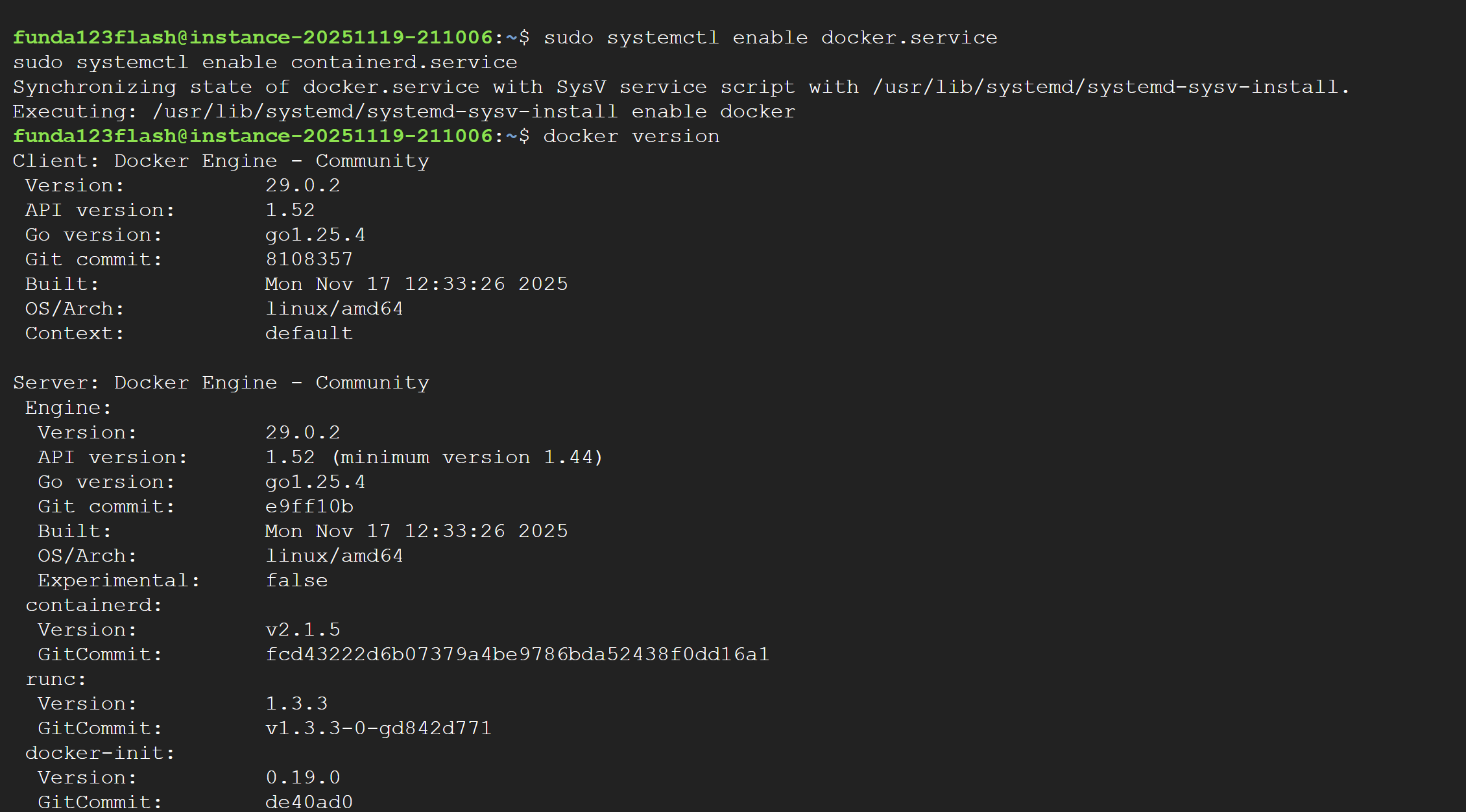
Now, run below 2 commands after above:

[Configure Docker to start on boot with systemd](https://docs.docker.com/engine/install/linux-postinstall/#configure-docker-to-start-on-boot-with-systemd)

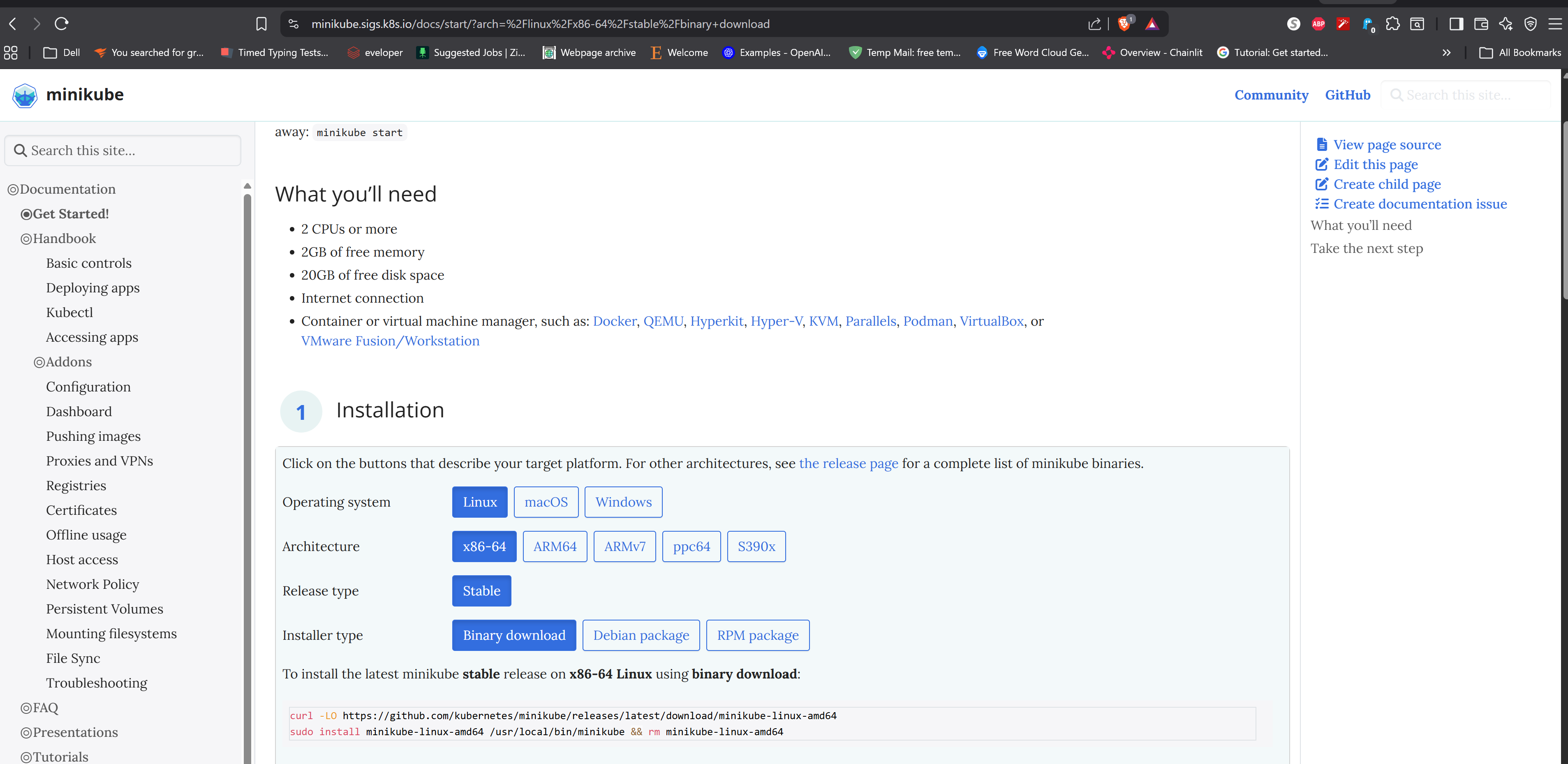
Many modern Linux distributions use [systemd](https://systemd.io/) to manage which services start when the system boots. On Debian and Ubuntu, the Docker service starts on boot by default. To automatically start Docker and containerd on boot for other Linux distributions using systemd, run the following commands:

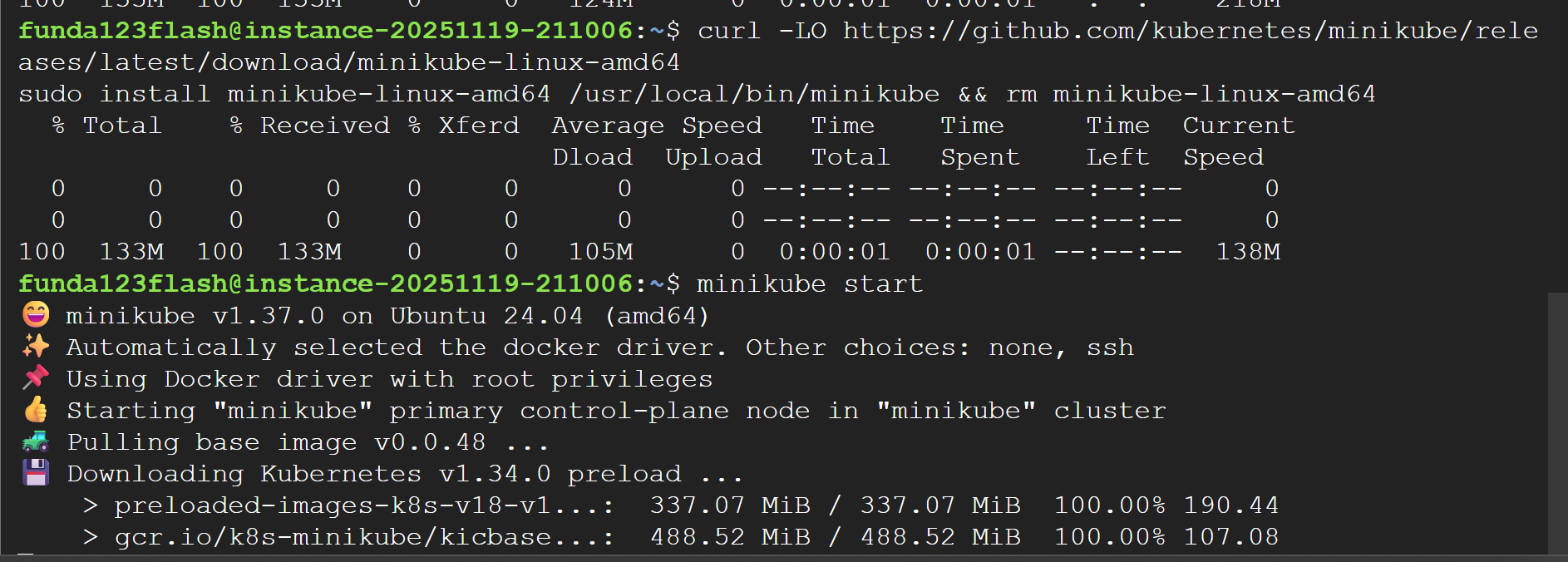
sudo systemctl enable docker.service

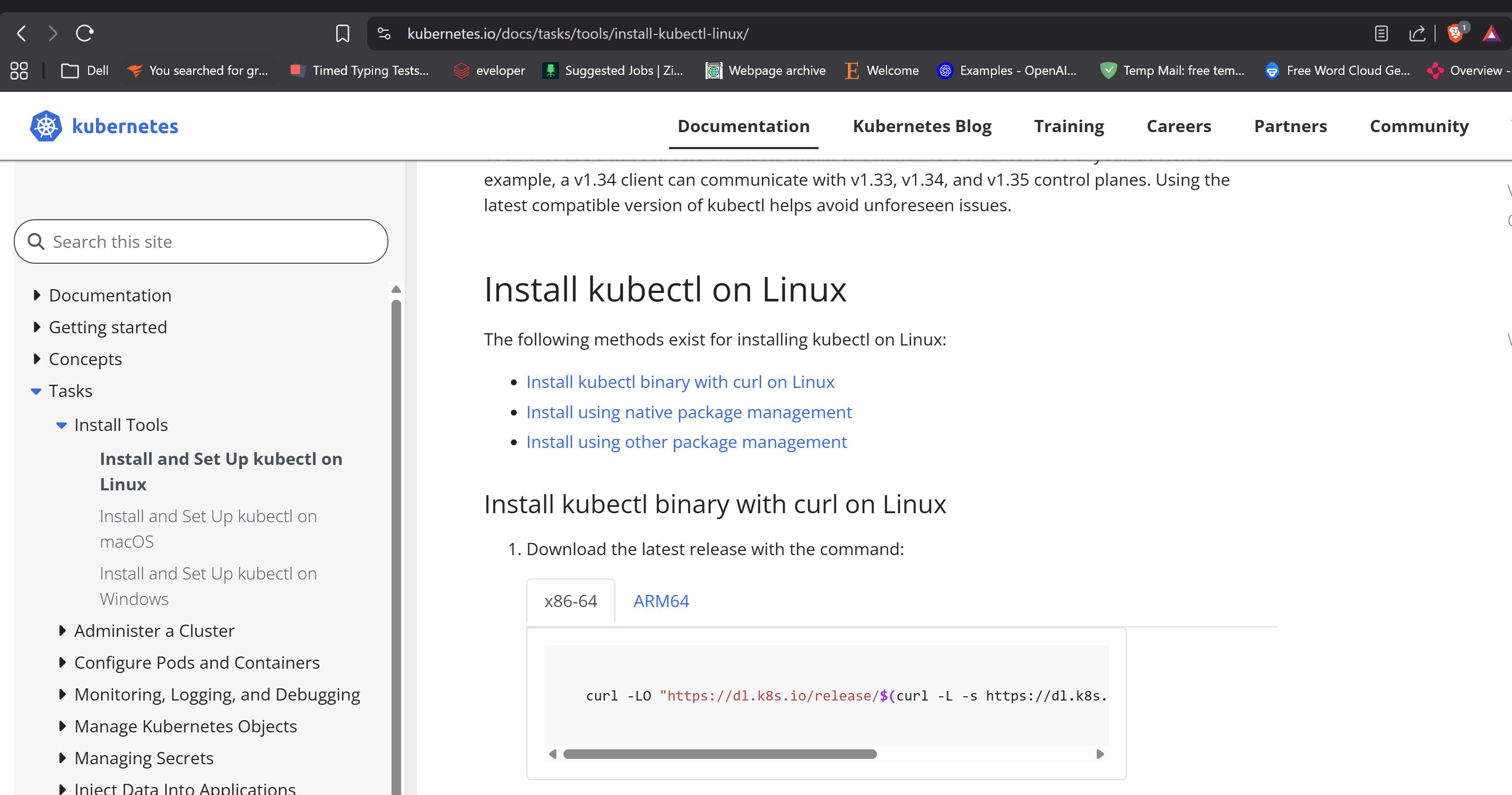
sudo systemctl enable containerd.service



As Docker is installed now, Now we will install minikube on GCP.

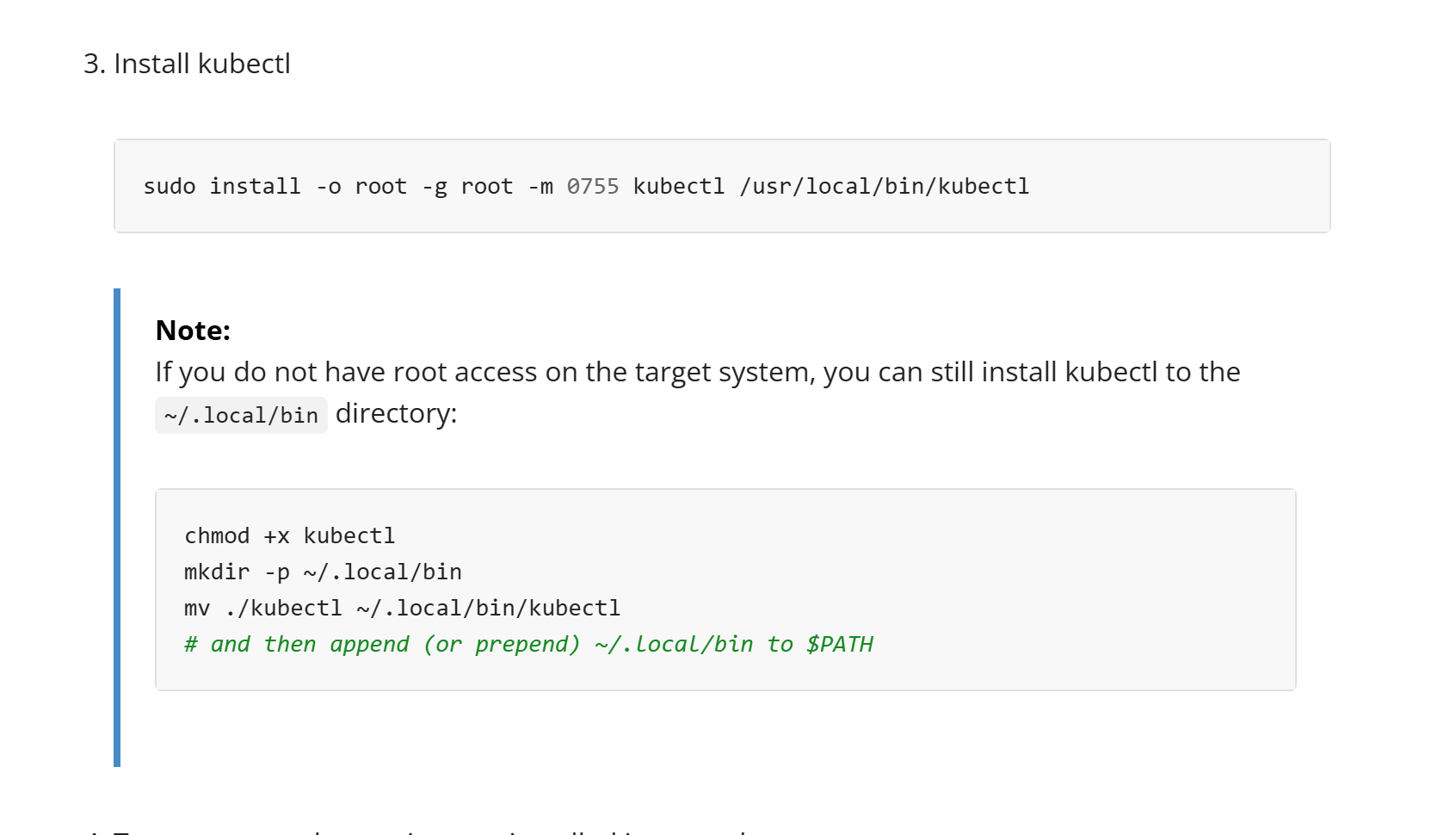




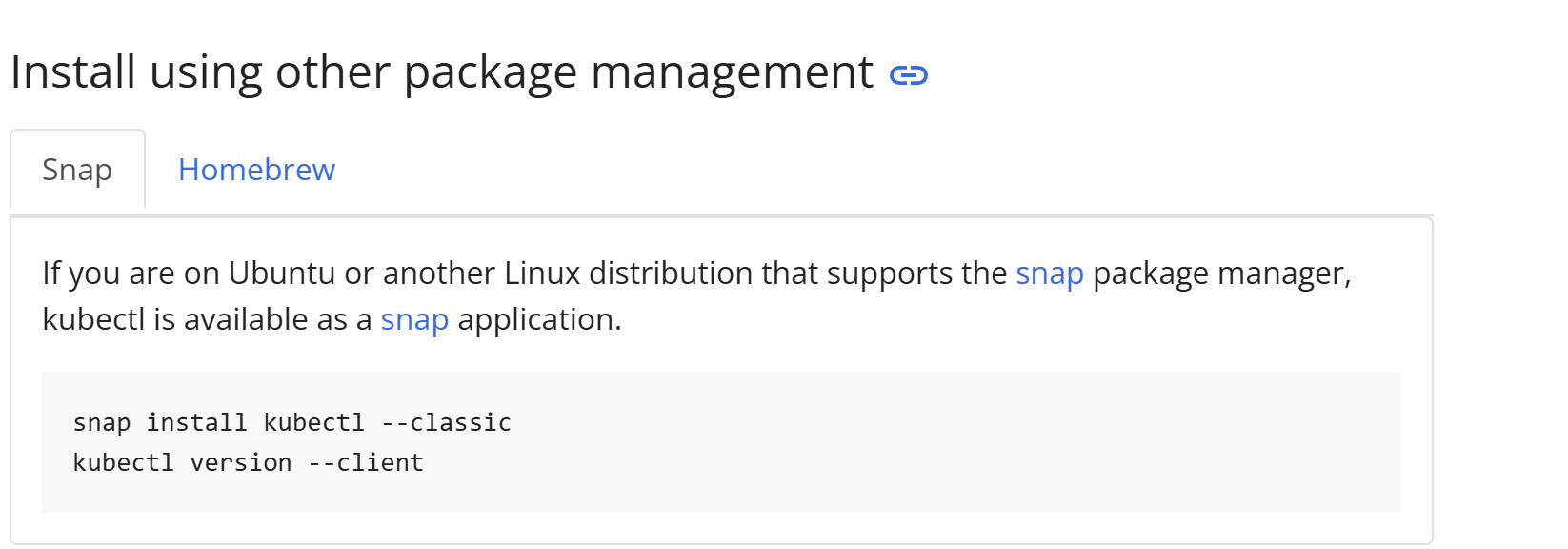


Now copy this command of kubectl and run in on SSH.

curl -LO [https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

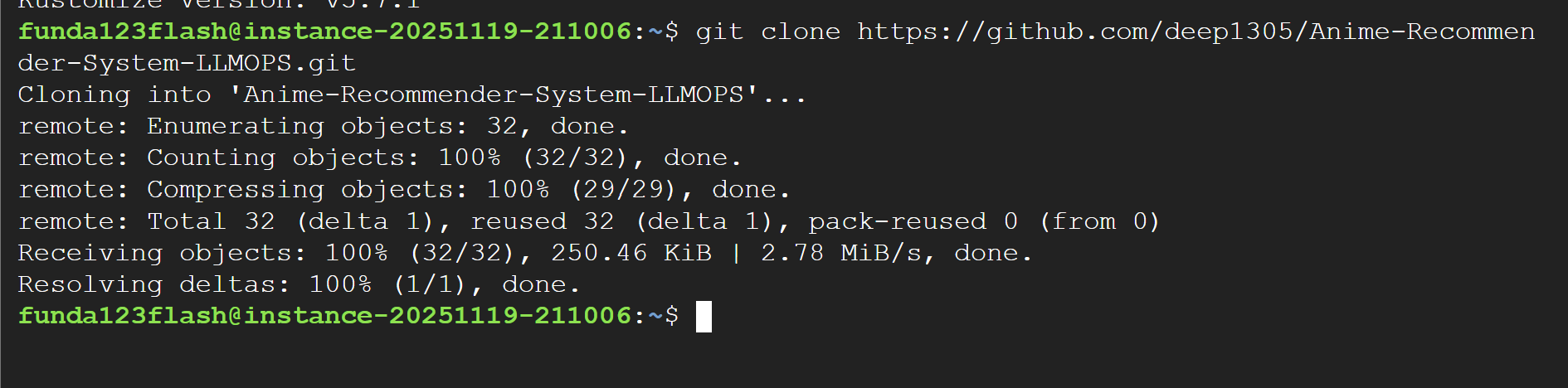


Or



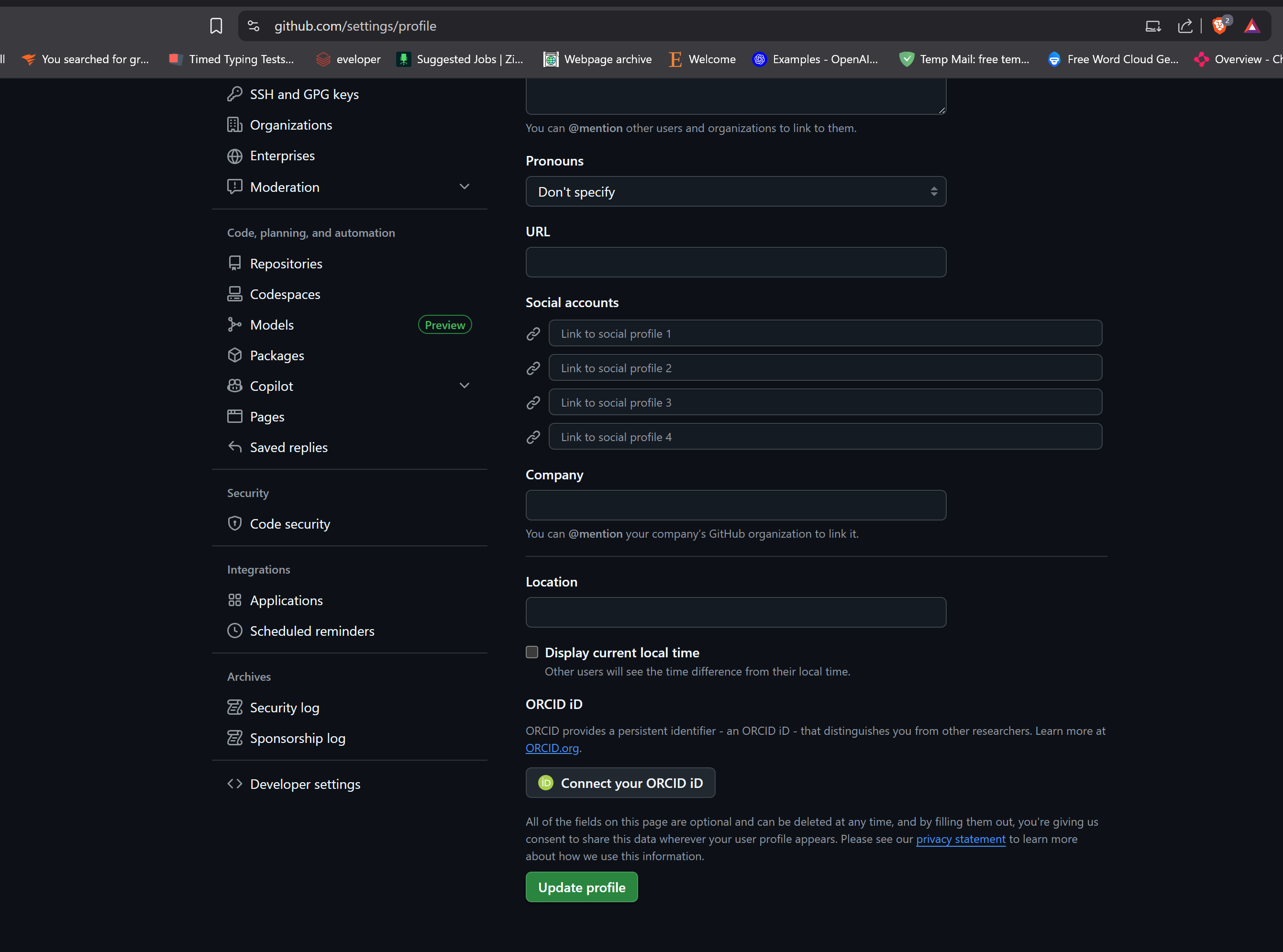
See below  

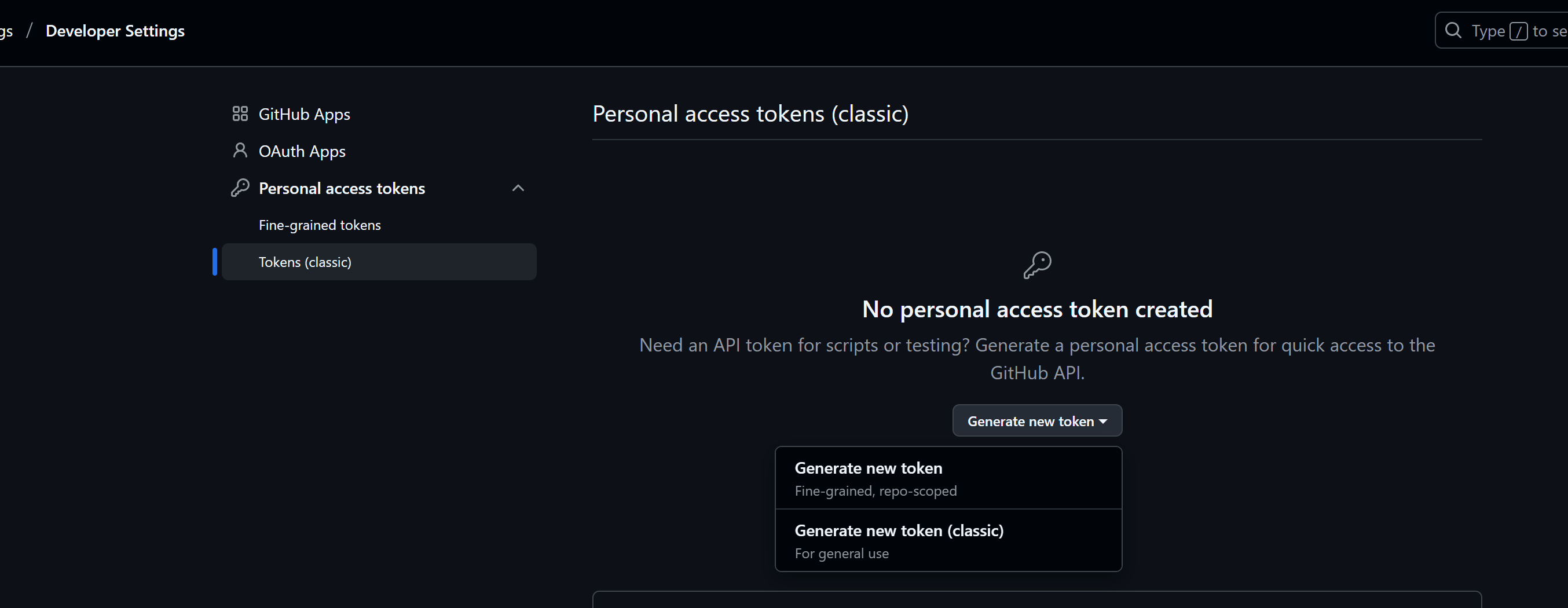

Now, we will clone the gitunb repository to our VM instance.





Click on developer settings to generate github token for VM onstance





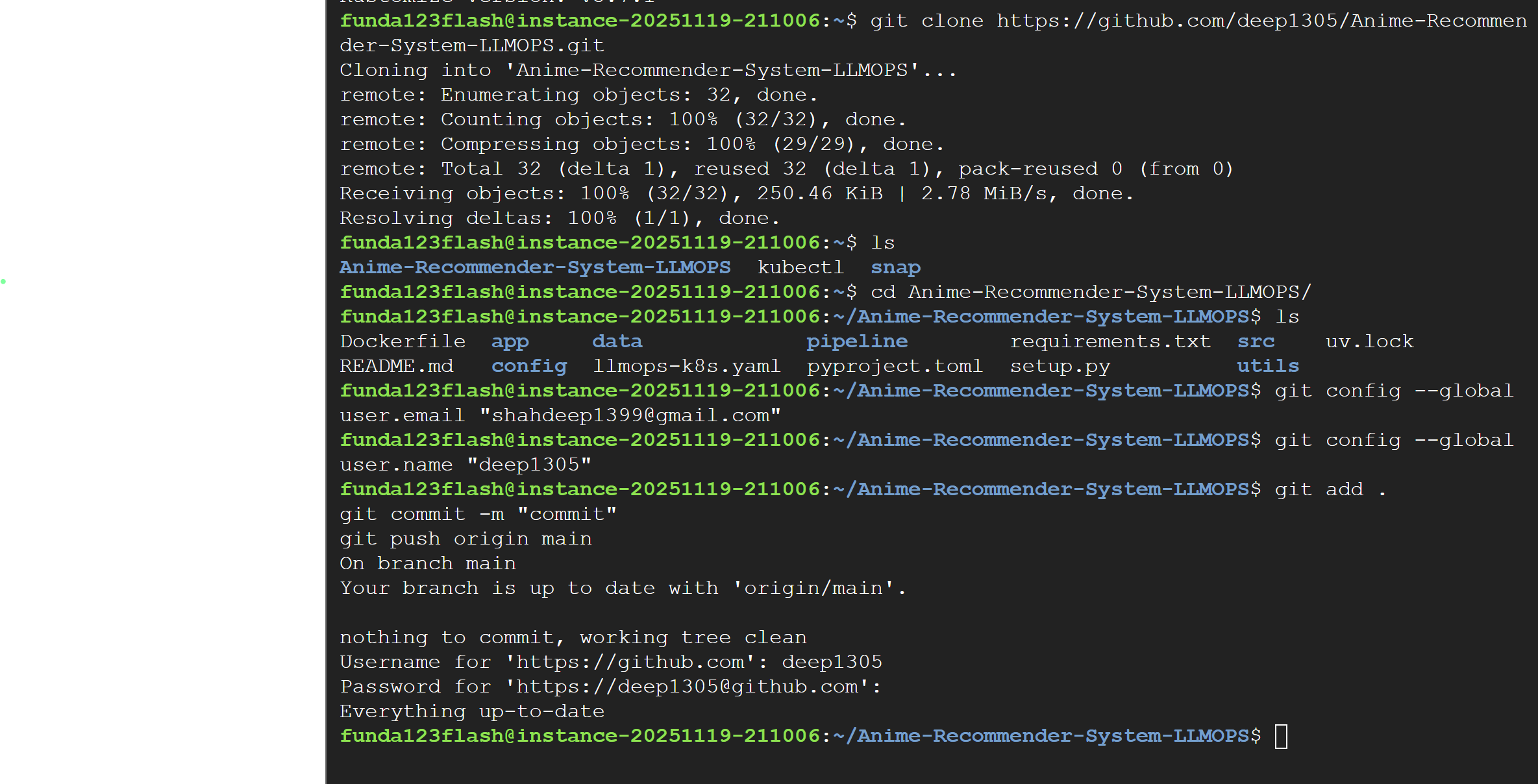
Click on classic in both

Slect this basic 5 :



ghp\_c3njH6828A7j7F9yfco9hTiwNxVa5g2e2dMO

we got this taken after clicking on generate new token



Now, we have interlinked local with Github and with VM instance