Minikube on AWS EC2: Quick Setup & Reusable AMI Guide

By Aman Pathak

Objective

The goal is to set up a lightweight, fully functional Kubernetes cluster using Minikube on AWS EC2 and then create an Amazon Machine Image (AMI) out of it. This way, the next time you want to spin up Minikube for learning, testing, or development, it's just one EC2 launch away. No repetitive installations, no pain.

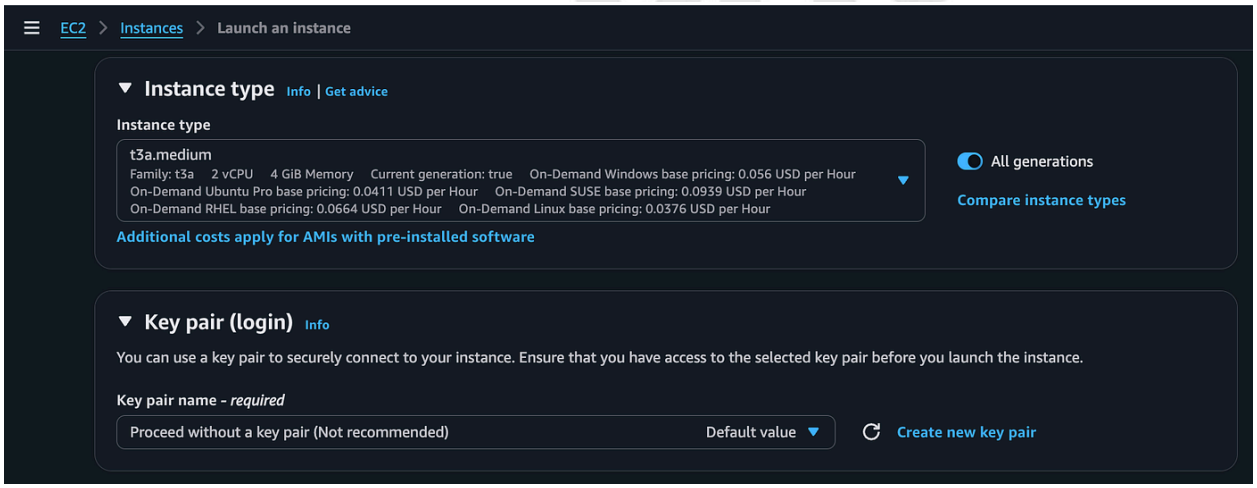
Pre-requisites

Here's what you'll need before getting started:

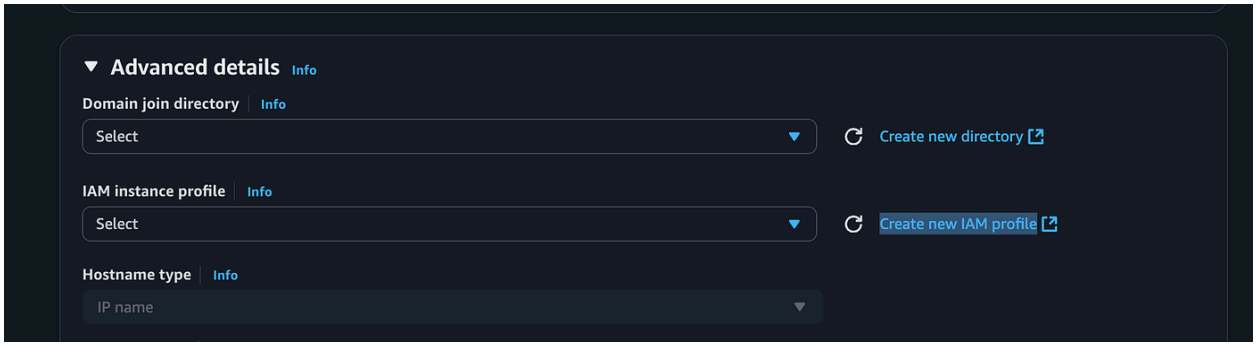
- An AWS account with permissions to create EC2, IAM roles, and AMIs
- Basic knowledge of Linux and EC2 instance management
- Familiarity with using AWS Systems Manager (for passwordless login)
- A public subnet in your default VPC to allow internet access
- Ubuntu 24.04 as the base AMI (used for Minikube setup)
- Patience during package installations (and maybe a coffee ☕)

HandsOn

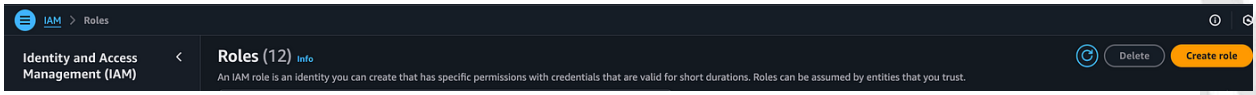
Create an EC2 Server with the following specifications.
Instance type- t3a.medium (Cheap as compare to t3.medium and required atleast 2vCPU for K8s workload)
Key Pair- No Need to provide the Key pair as we use the IAM Instance Profile to log in to the EC2
AMI- Ubuntu 24.04
VPC and other networking- Use the default VPC and default security group, but the subnet should be public for now.



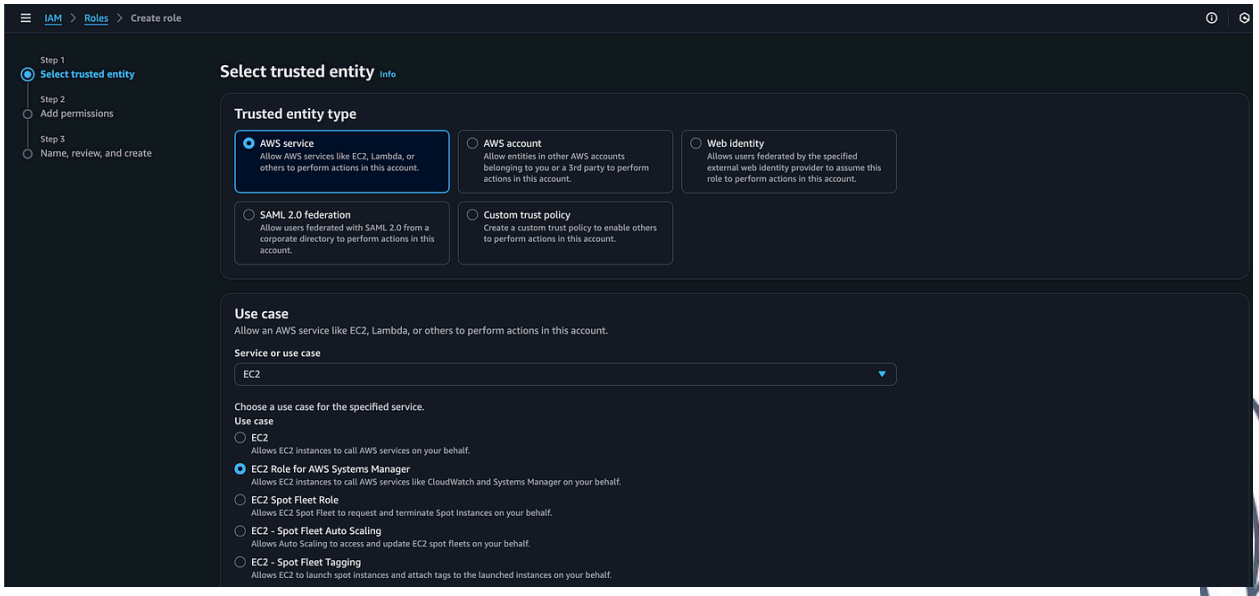
To log in via Systems Manager instead of SSH, we will have to create an IAM Instance Profile. To do it, click on Create new IAM profile



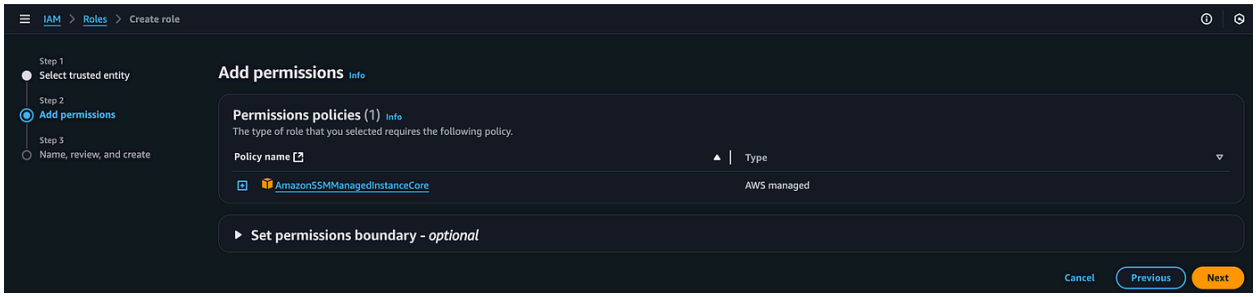
Click on Create role



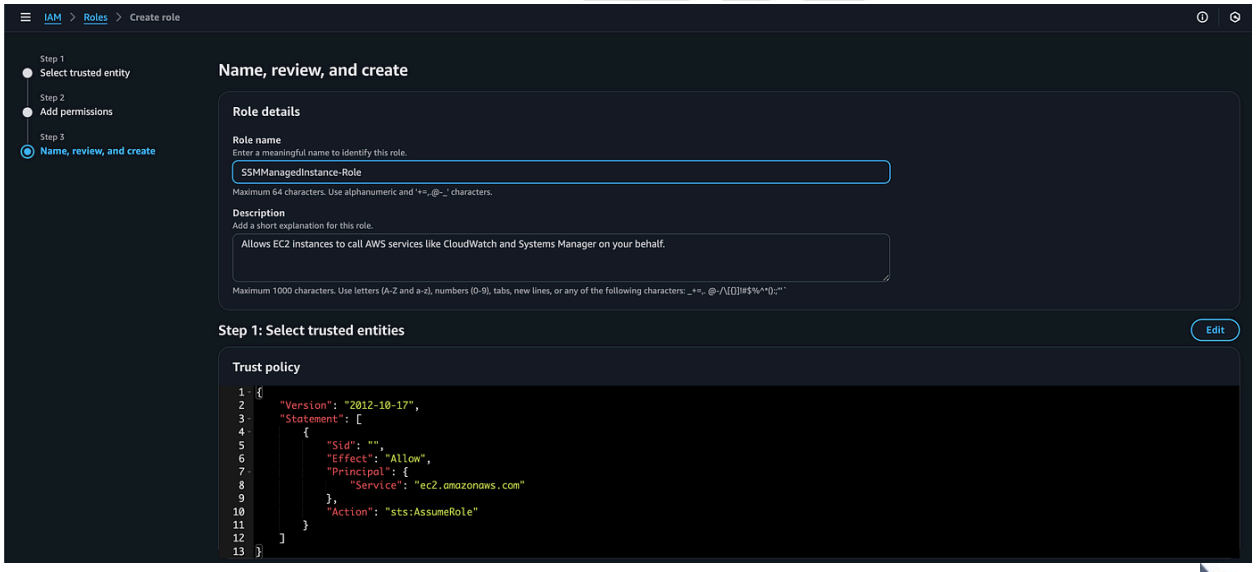
Under Use case, you have to select EC2, and then, the sub Use case for EC2 would be EC2 Role for AWS Systems Manager to access EC2 servers via Systems Manager



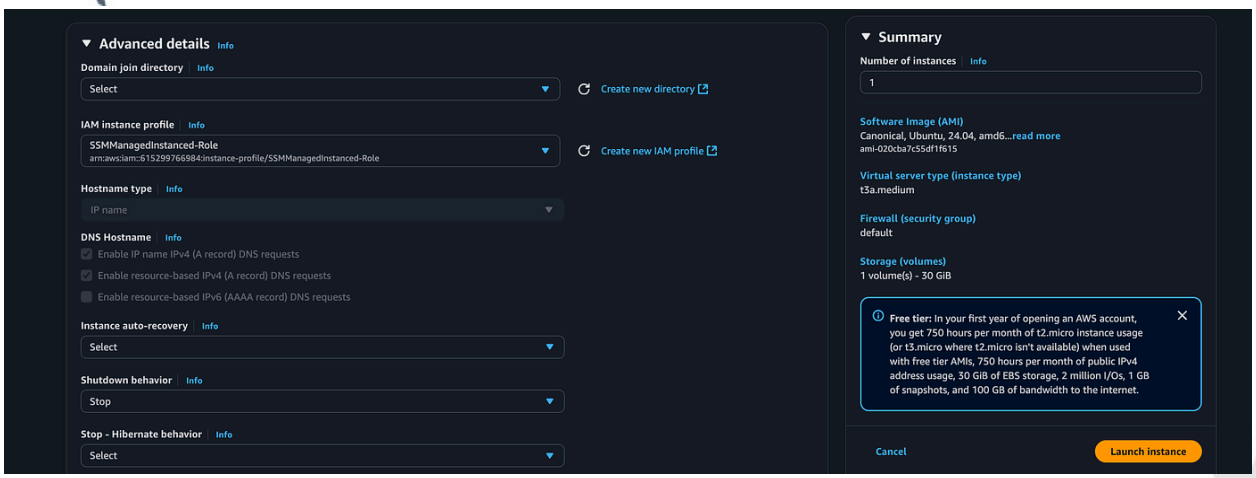
It will add the IAM policy automatically as you selected the EC2 Role for AWS Systems Manager. Click on Next.



Provide a meaningful name to your IAM Role and click on create role.

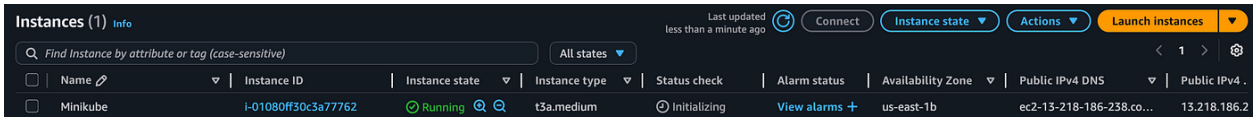


Go back to the EC2 creation console and click on the drop-down button to select the IAM Instance Profile

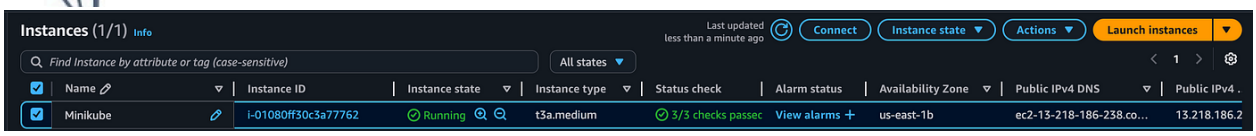


Once you select the created IAM Instance Profile, click on Launch instance.

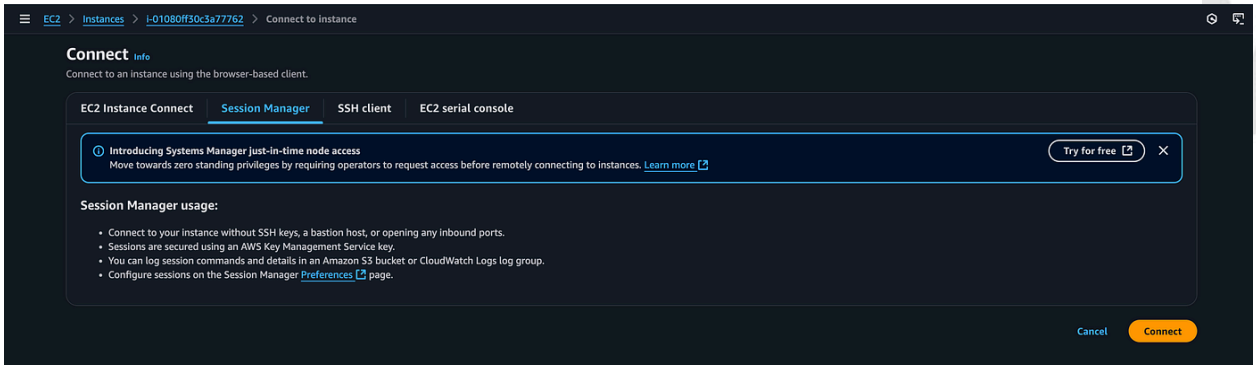
You will be able to see your server getting into the initialisation state. Wait for the 3/3 health check.



As soon as you get 3/3 checks passed status, select the created EC2 and click on Connect.



Go to the Session Manager tab and click on Connect to enter the server



Log in as the ubuntu user. To do that, run the command below

```
$ sudo su ubuntu
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

cubuntu@ip-172-31-4-185:/var/snap/amazon-ssm-agent/11320$ cd
ubuntu@ip-172-31-4-185:~$
```

Installation of Minikube

Update the packages

```
sudo apt-get update
```

```
ubuntu@ip-172-31-4-185:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1144 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [242 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1086 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [276 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [376 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [26.0 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [1224 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [256 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [21.7 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse Translation-en [4788 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 c-n-f Metadata [592 B]
Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.2 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [8676 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7068 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [27.1 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [16.5 kB]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [16.4 kB]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
```

Install required packages using the command below

```
sudo apt install curl apt-transport-https
```

```
ubuntu@ip-172-31-4-185:~$ sudo apt install curl apt-transport-https
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 23 not upgraded.
Need to get 3970 B of archives.
After this operation, 36.9 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 apt-transport-https all 2.8.3 [3970 B]
Fetched 3970 B in 0s (179 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 70681 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.8.3_all.deb ...
Unpacking apt-transport-https (2.8.3) ...
Setting up apt-transport-https (2.8.3) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-4-185:~$
```

Download the minikube binary using the command below

```
curl -O https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
ubuntu@ip-172-31-4-185:~$ curl -O https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  126M  100  126M    0     0   188M      0 --:--:-- --:--:-- --:--:--  187M
ubuntu@ip-172-31-4-185:~$
```

Move the binar to the usr/local/bin/ directory

```
sudo cp minikube-linux-amd64 /usr/local/bin/minikube
```

```
ubuntu@ip-172-31-4-185:~$ sudo cp minikube-linux-amd64 /usr/local/bin/minikube
ubuntu@ip-172-31-4-185:~$ ls /usr/local/bin/ -l
total 129656
-rw-r--r-- 1 root root 132766301 Jun 22 07:13 minikube
ubuntu@ip-172-31-4-185:~$
```

Now, we have to permit the binary for execution. Use the command below

```
sudo chmod 755 /usr/local/bin/minikube
```

```
ubuntu@ip-172-31-4-185:~$ sudo chmod 755 /usr/local/bin/minikube
ubuntu@ip-172-31-4-185:~$
```

To validate whether minikube is installed or not. Run the command below

```
minikube version
```

```
ubuntu@ip-172-31-4-185:~$ minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1-dirty
```

To interact with Kubernetes, we have to install kubectl. Use the commands below

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
ubuntu@ip-172-31-4-185:~$    curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1591      0 --:--:-- --:--:-- --:--:--  1604
100 57.3M  100 57.3M    0     0   140M      0 --:--:-- --:--:-- --:--:--  140M
```

To validate the binary, you have to install the checksum file using the command below

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
```

```
ubuntu@ip-172-31-4-185:~$    curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1258      0 --:--:-- --:--:-- --:--:--  1266
100    64  100    64    0     0    354      0 --:--:-- --:--:-- --:--:-- 12800
```

Now, validate the binary against the checksum file

```
ubuntu@ip-172-31-4-185:~$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
```

Validate whether the kubectl utility has been installed or not using the command below

```
kubectl version --client
```

```
ubuntu@ip-172-31-4-185:~$ kubectl version --client
Client Version: v1.33.2
Kustomize Version: v5.6.0
```

Now, we have to start minikube

```
ubuntu@ip-172-31-4-185:~$ minikube start
😄  minikube v1.36.0 on Ubuntu 24.04
👎  Unable to pick a default driver. Here is what was considered, in preference order:
💡  Alternatively you could install one of these drivers:
    ▪ docker: Not installed: exec: "docker": executable file not found in $PATH
    ▪ kvm2: Not installed: exec: "virsh": executable file not found in $PATH
    ▪ qemu2: Not installed: exec: "qemu-system-x86_64": executable file not found in $PATH
    ▪ podman: Not installed: exec: "podman": executable file not found in $PATH
    ▪ virtualbox: Not installed: unable to find VBoxManage in $PATH

❌  Exiting due to DRV_NOT_DETECTED: No possible driver was detected. Try specifying --driver, or see https://minikube.sigs.k8s.io/docs/start/
```

As you can see, Minikube did not start; the reason is driver unavailability. In our demonstration, we have to provide a driver to Minikube to run the workloads on it. For that, we will use Docker. Hence, we have to install Docker

Install Docker using the command below.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get 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:
echo \
 "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
 $(. /etc/os-release && echo
"${UBUNTU_CODENAME:-$VERSION_CODENAME}")
stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
ubuntu@ip-172-31-4-185:~$ # Add Docker's official GPG key:
sudo apt-get update
sudo apt-get 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:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
0 upgraded, 0 newly installed, 0 to remove and 23 not upgraded.
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [25.8 kB]
Fetched 74.7 kB in 1s (84.9 kB/s)
Reading package lists... Done
```

Install other Docker-associated utilities using the command below.

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

```
ubuntu@ip-172-31-4-185:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  cgroupfs-mount | cgroup-lite docker-model-plugin
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 23 not upgraded.
Need to get 121 MB of archives.
After this operation, 445 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Validate whether Docker has been installed or not using the command below.

sudo docker run hello-world

```
ubuntu@ip-172-31-4-185:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:940c619fbd418f9b2b1b63e25d8861f9cc1b46e3fc8b018ccfe8b78f19b8cc4f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Error

Now, if you try to list containers or try to run any Docker command, it won't work because the ubuntu user does not have permission to connect to the Docker daemon socket, which is very common.

```
ubuntu@ip-172-31-4-185:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.50/containers/js
on": dial unix /var/run/docker.sock: connect: permission denied
```

To fix the error, run the command below

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
docker run hello-world
```

```
ubuntu@ip-172-31-4-185:~$ sudo groupadd docker
groupadd: group 'docker' already exists
ubuntu@ip-172-31-4-185:~$ sudo usermod -aG docker $USER
```

```
ubuntu@ip-172-31-4-185:~$ newgrp docker
ubuntu@ip-172-31-4-185:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

ubuntu@ip-172-31-4-185:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@ip-172-31-4-185:~$
```

As the Docker issue has been fixed. Start Minikube using the command below.

```
minikube start
```

```
ubuntu@ip-172-31-4-185:~$ minikube start
😄  minikube v1.36.0 on Ubuntu 24.04
✨  Automatically selected the docker driver. Other choices: none, ssh
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.47 ...
💾  Downloading Kubernetes v1.33.1 preload ...
    > preloaded-images-k8s-v18-v1...:  347.04 MiB / 347.04 MiB  100.00% 162.08
    > gcr.io/k8s-minikube/kicbase...:  498.36 MiB / 502.26 MiB  99.22% 118.69 M
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

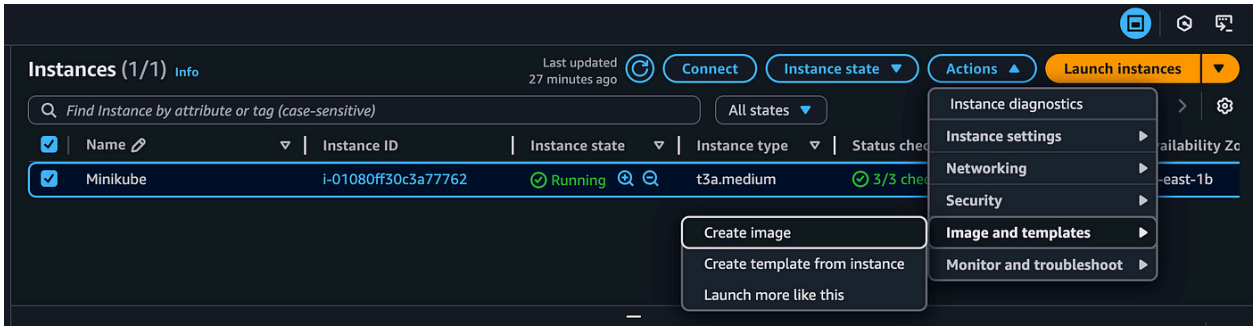
Also, you can validate whether your Kubernetes cluster has been set up perfectly through minikube or not, use the command below.

```
kubectl get nodes
kubectl get pods -A
```

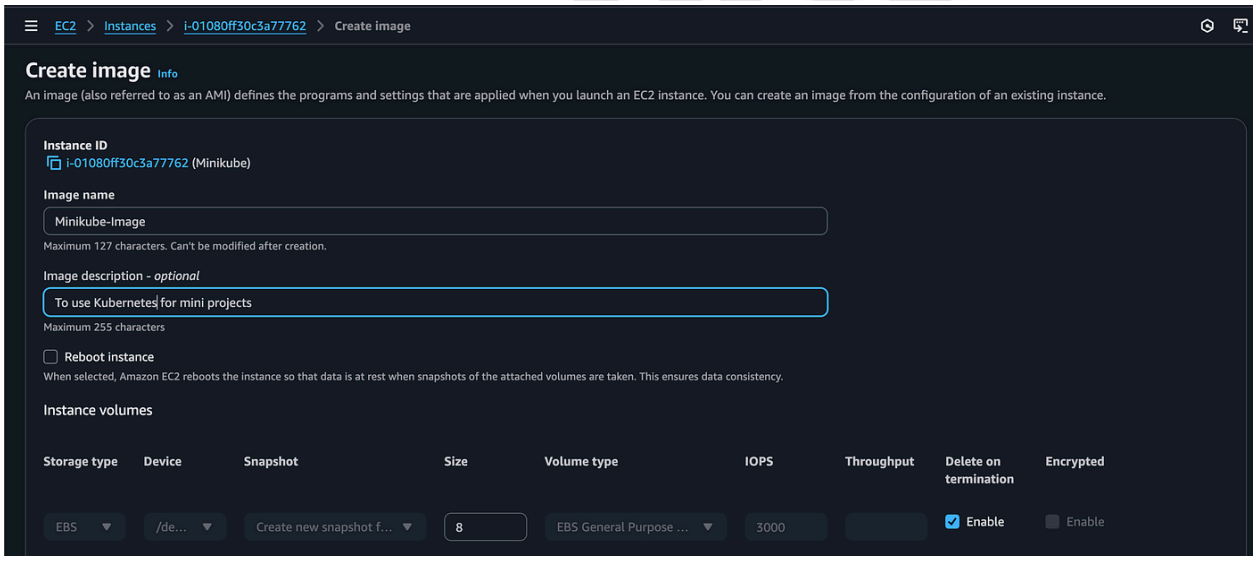
```
ubuntu@ip-172-31-4-185:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   54s   v1.33.1
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-674b8bbfcf-jv4d9           1/1     Running   0             78s
kube-system   etcd-minikube                      1/1     Running   0             84s
kube-system   kube-apiserver-minikube            1/1     Running   0             82s
kube-system   kube-controller-manager-minikube   1/1     Running   0             84s
kube-system   kube-proxy-c7tdk                   1/1     Running   0             78s
kube-system   kube-scheduler-minikube            1/1     Running   0             82s
kube-system   storage-provisioner                1/1     Running   1 (46s ago)   78s
```

Now, I aim to create this demonstration to configure a Kubernetes cluster. As we can do it through multiple ways using Kind or Minikube, or any cloud-based Cluster. But the simplest way I found out from my experience is to configure Minikube once and create an AMI on AWS. So, whenever I need Minikube, I will just need to launch an EC2 from the same Image.

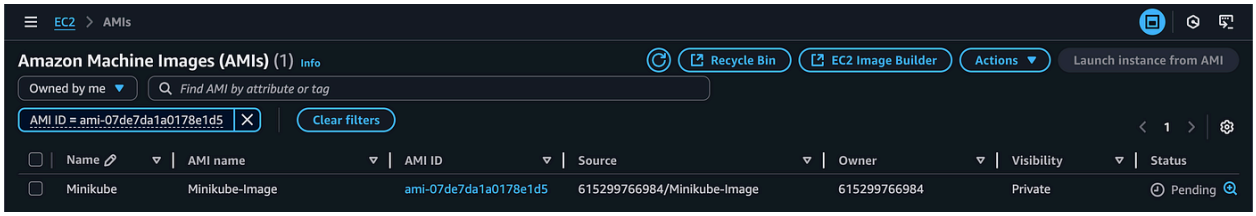
Hence, we have to create an AMI(Amazon Machine Image) of the configured Minikube.
Select the EC2 Server, click on Actions, then select Image and templates and click on Create image



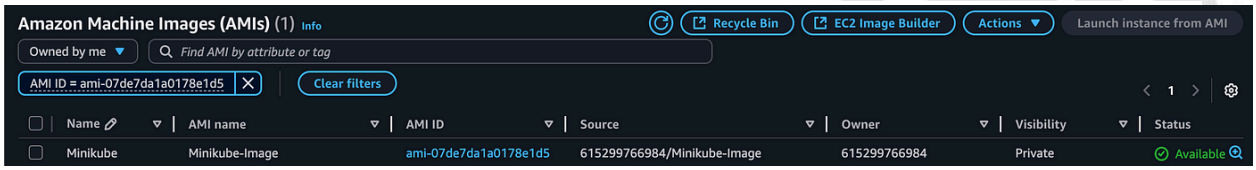
Provide a meaningful name to your AMI and description, and click on Create AMI.



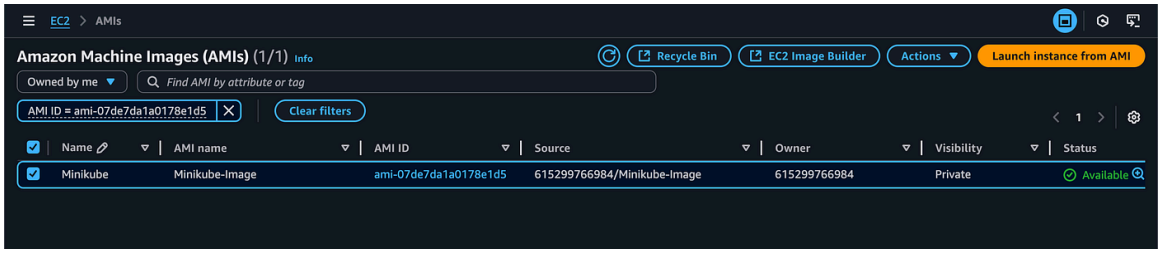
As soon as you click on create AMI, you will see the Pending Status as creation of AMI takes a little bit of time(approx 4–5 minutes, depending on the workload)



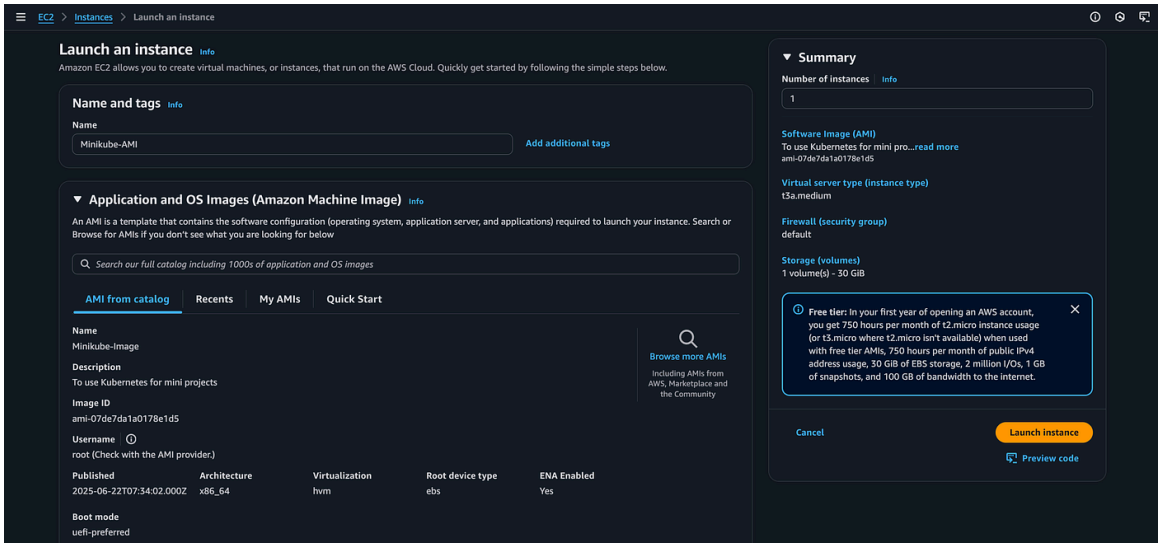
After waiting for 4–5 minutes, the AMI is in an available state

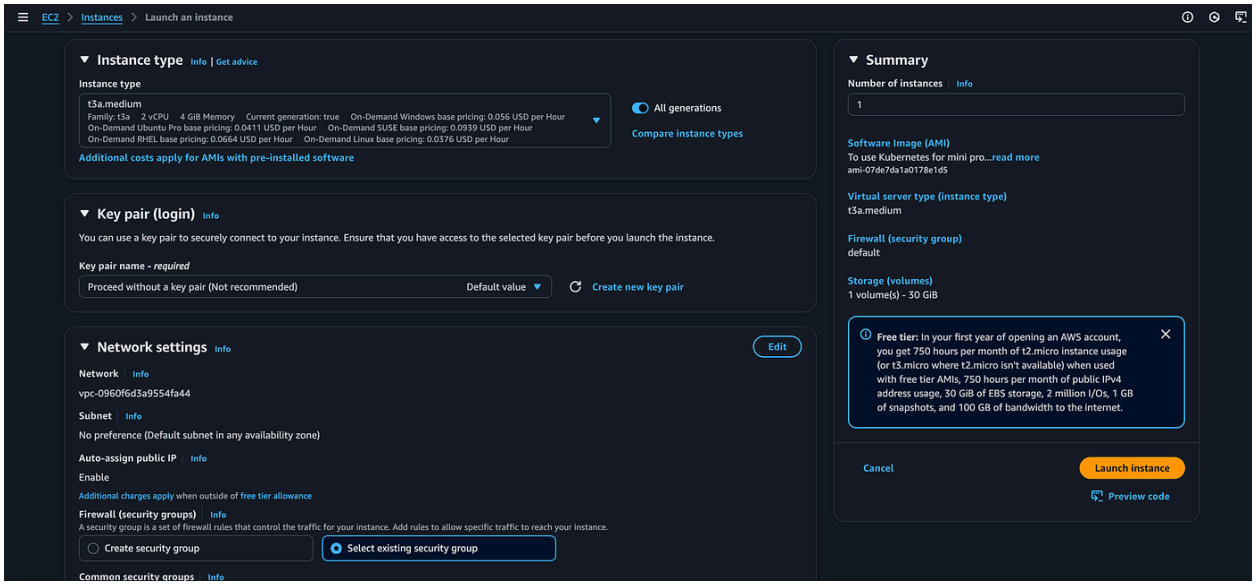


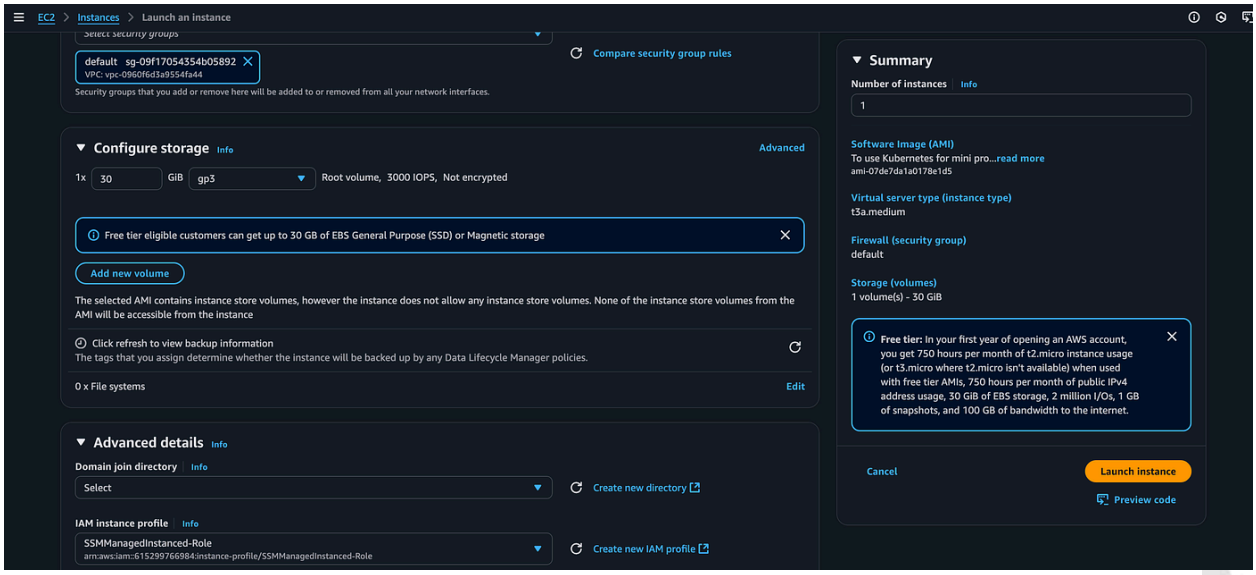
Launch an Instance from AMI to validate whether it's working or not



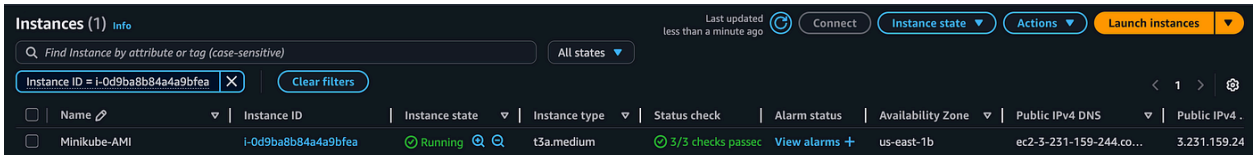
Specifications would be the same as it was for the earlier EC2



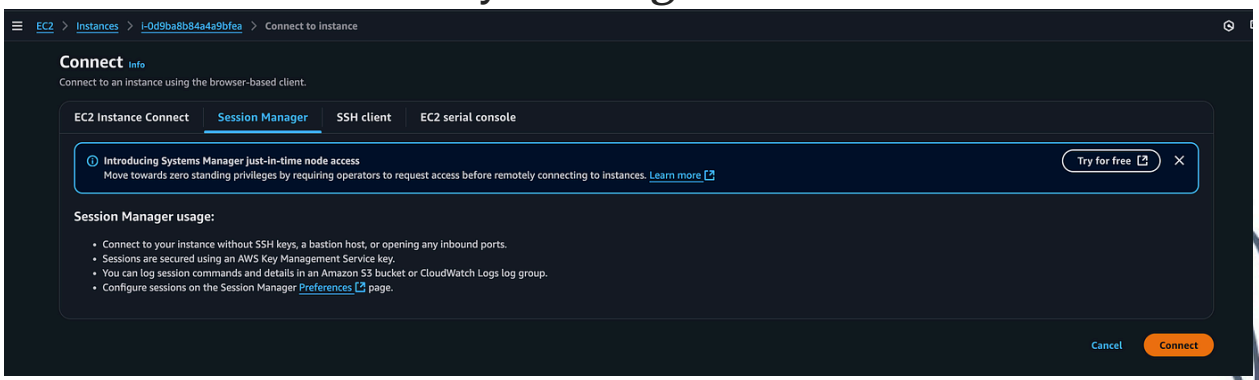




Now, the EC2 instance has been created, and the status check is 3/3.



Connect with the EC2 by clicking on Connect.



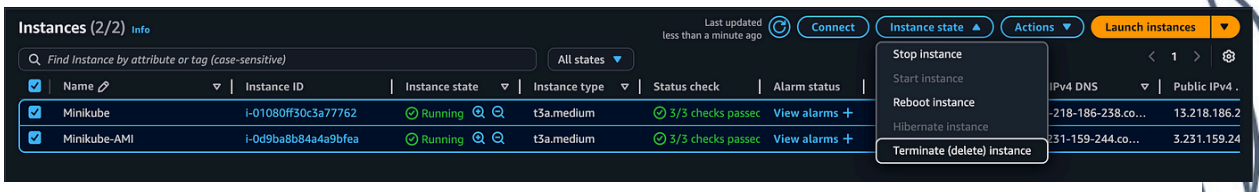
As the new machine has been created, all the processes will be stopped automatically. Hence, our minikube has also stopped.

So, start the minikube server using the command below

```
minikube start
```

```
ubuntu@ip-172-31-7-130:~$ minikube start
😄  minikube v1.36.0 on Ubuntu 24.04
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.47 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.33.1 on Docker 28.1.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-7-130:~$
ubuntu@ip-172-31-7-130:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   21m   v1.33.1
ubuntu@ip-172-31-7-130:~$
ubuntu@ip-172-31-7-130:~$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS      AGE
kube-system   coredns-674b8bbfcf-jv4d9           0/1     Running   1 (83s ago)   21m
kube-system   etcd-minikube                      1/1     Running   1 (83s ago)   21m
kube-system   kube-apiserver-minikube            1/1     Running   1 (83s ago)   21m
kube-system   kube-controller-manager-minikube   1/1     Running   1 (83s ago)   21m
kube-system   kube-proxy-c7tdk                   1/1     Running   1 (83s ago)   21m
kube-system   kube-scheduler-minikube            1/1     Running   1 (83s ago)   21m
kube-system   storage-provisioner                1/1     Running   2 (83s ago)   21m
ubuntu@ip-172-31-7-130:~$
```

To save the cost, terminate both EC2 Servers



Conclusion

That’s it! You now have a custom AMI with Minikube pre-installed — ready to go whenever you need it. This setup is especially helpful for folks working in DevOps or cloud learning environments who want a repeatable, lightweight Kubernetes cluster without provisioning heavy cloud-native options every time. Plus, you learned a bit about IAM, Docker, and EC2 along the way.

If you’re exploring Kubernetes and want a reliable playground that spins up in minutes, this Minikube + EC2 AMI trick is gold. Give it a try and feel free to tweak it to your needs!

Follow for More