Day 31 Task: Launching your First Kubernetes Cluster with Nginx running

How to Install, Configure, and Deploy Nginx on a Kubernetes Cluster

What is minikube?

Minikube is a lightweight tool that enables developers to run and test Kubernetes clusters on their local machines. It creates a single-node Kubernetes cluster within a virtual machine, allowing users to experiment with Kubernetes features and develop applications that can be deployed to production Kubernetes environments. Minikube is a popular tool for testing, development, and learning Kubernetes.

Features of minikube

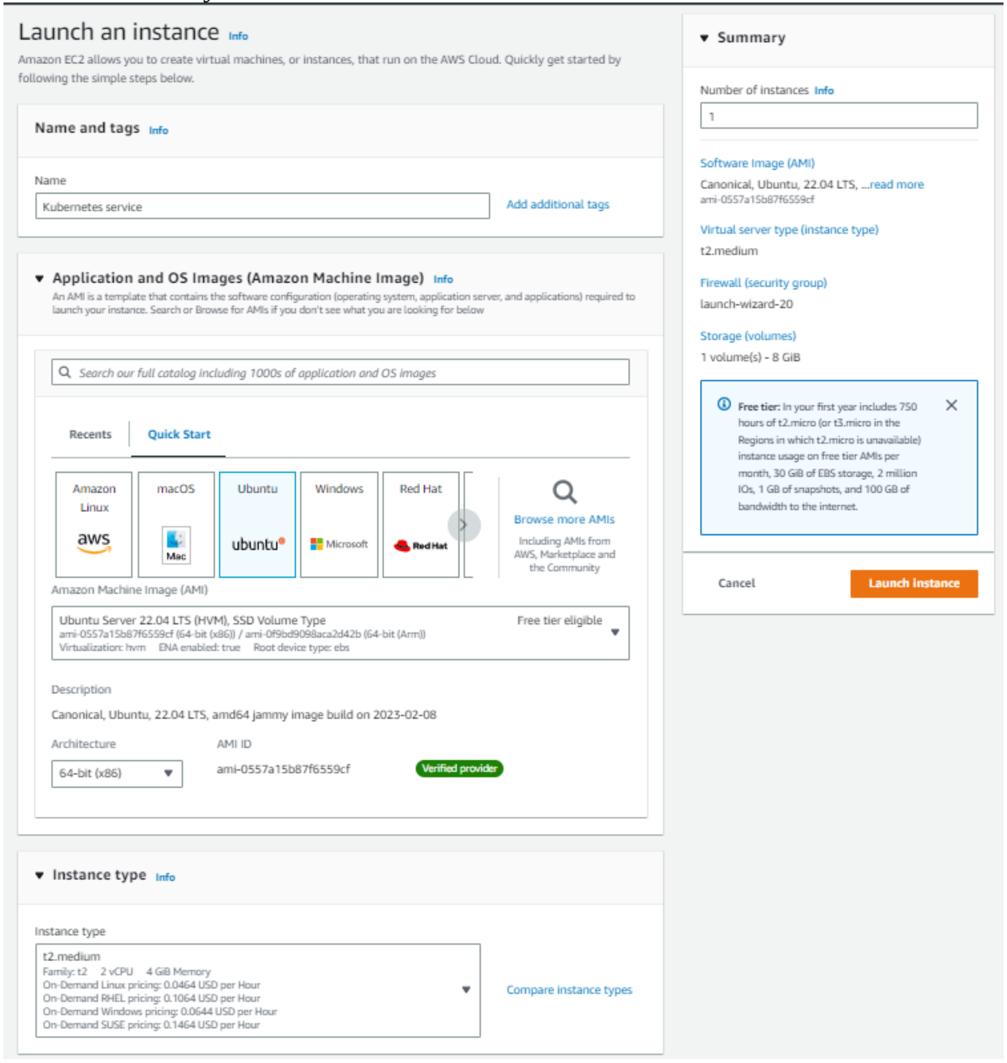
Minikube is a tool that enables you to run and develop applications on a local Kubernetes cluster. Its key features include:

1. Local Kubernetes cluster: Minikube enables you to create and run a Kubernetes cluster on your local machine, making it easy to develop and test applications without needing a full-scale Kubernetes environment.
2. Multi-node clusters: Minikube can create multi-node Kubernetes clusters on a single machine, allowing you to test more complex applications and configurations.
3. Easy setup: Minikube is easy to set up and can be run on various platforms, including Windows, Mac, and Linux.
4. Support for various Kubernetes versions: Minikube supports different Kubernetes versions, allowing you to test your applications against different versions of Kubernetes.
5. Integration with container runtimes: Minikube supports different container runtimes such as Docker and CRI-O, enabling you to choose the runtime that works best for your use case.
6. Add-ons: Minikube supports various add-ons such as Kubernetes Dashboard, Ingress controller, and many others, allowing you to extend the capabilities of your local Kubernetes cluster.
7. Virtualization support: Minikube can work with different virtualization technologies, such as VirtualBox, HyperKit, and KVM, providing flexibility to choose the best virtualization environment for your local Kubernetes cluster.

Task-01:

Install minikube on your local

1. Launch an instance Kubernetes -server with **t2.medium** instance-type, **Ubuntu OS** and Connect this to your terminal.



Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: Kubernetes service

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0557a15b87f6559cf (64-bit (x86)) / ami-0f9bd9098aca2d42b (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-02-08

Architecture: 64-bit (x86)

AMI ID: ami-0557a15b87f6559cf — Verified provider

Instance type Info

Instance type
t2.medium
Family: t2 2 vCPU 4 GiB Memory
On-Demand Linux pricing: 0.0464 USD per Hour
On-Demand RHEL pricing: 0.1064 USD per Hour
On-Demand Windows pricing: 0.0644 USD per Hour
On-Demand SUSE pricing: 0.1464 USD per Hour

Compare instance types

Summary

Number of instances Info: 1

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-0557a15b87f6559cf

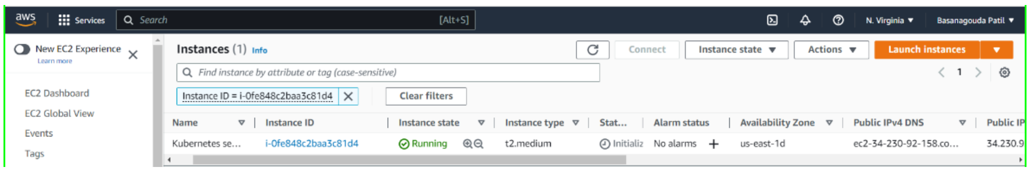
Virtual server type (instance type)
t2.medium

Firewall (security group)
launch-wizard-20

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel | Launch instance



Services | Search [Alt+S] | N. Virginia | Basanagouda Patil

New EC2 Experience — Learn more

EC2 Dashboard
EC2 Global View
Events
Tags

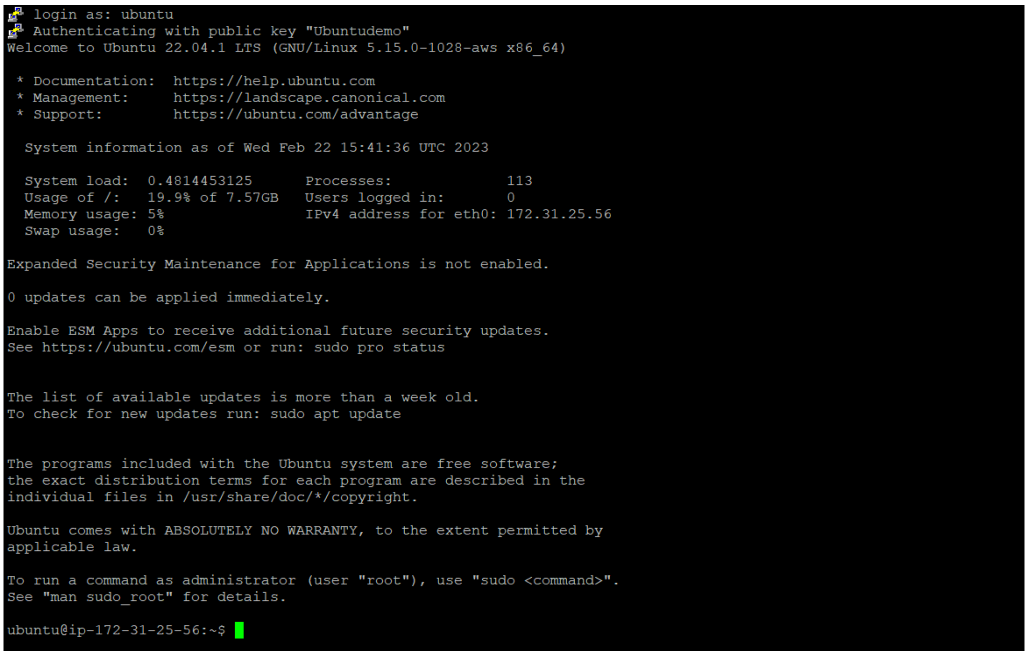
Instances (1) Info

Connect | Instance state | Actions | Launch instances

Find instance by attribute or tag (case-sensitive)

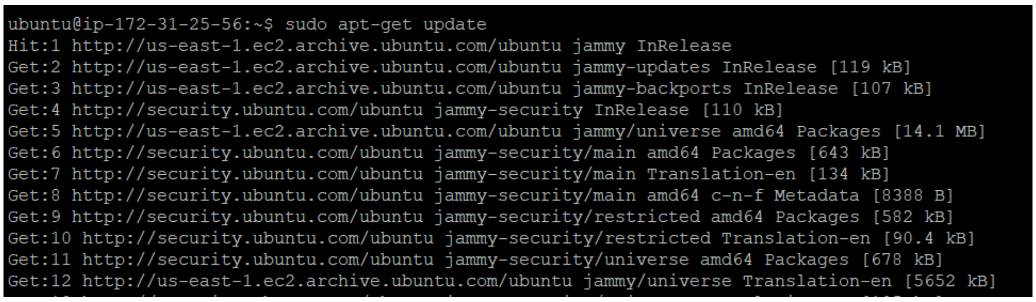
Instance ID = i-0fe848c2baa3c81d4 | Clear filters

Name	Instance ID	Instance state	Instance type	Stat...	Alarm status	Availability Zone	Public IPv4 DNS	Public IP
Kubernetes se...	i-0fe848c2baa3c81d4	Running	t2.medium	Initializ	No alarms	us-east-1d	ec2-34-230-92-158.co...	34.230.9

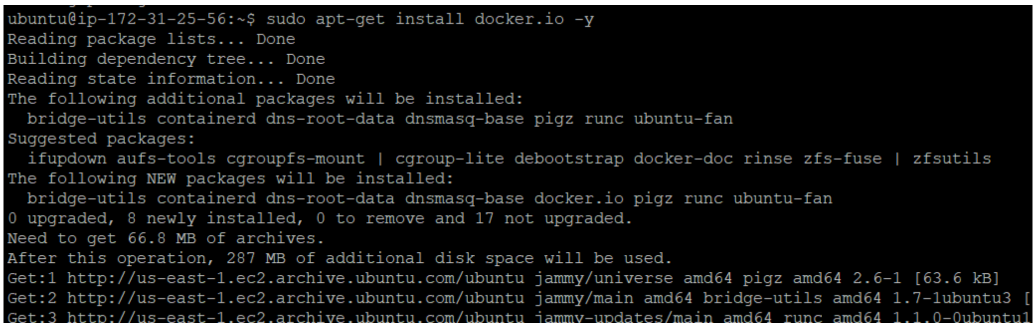


```
login as: ubuntu
Authenticating with public key "Ubuntudemo"
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Feb 22 15:41:36 UTC 2023

  System load:  0.4814453125      Processes:             113
  Usage of /:   19.9% of 7.57GB   Users logged in:       0
  Memory usage: 5%                IPv4 address for eth0: 172.31.25.56
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

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

ubuntu@ip-172-31-25-56:~$
```

2. Update the packet manager & install docker by using following commands
   sudo apt-get update
   sudo apt install docker.io -y command.

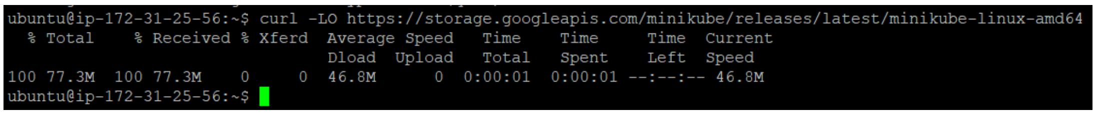


```
ubuntu@ip-172-31-25-56:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [643 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [134 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [8388 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [582 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [90.4 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [678 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
```



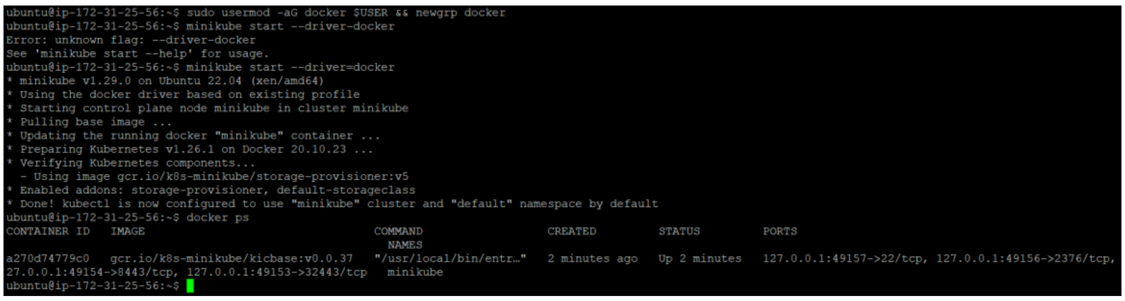
```
ubuntu@ip-172-31-25-56:~$ sudo apt-get install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 17 not upgraded.
Need to get 66.8 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.0-0ubuntu1
```

3. Install minikube by using Linux commands such as "**curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && sudo install minikube-linux-amd64 /usr/local/bin/minikube**"



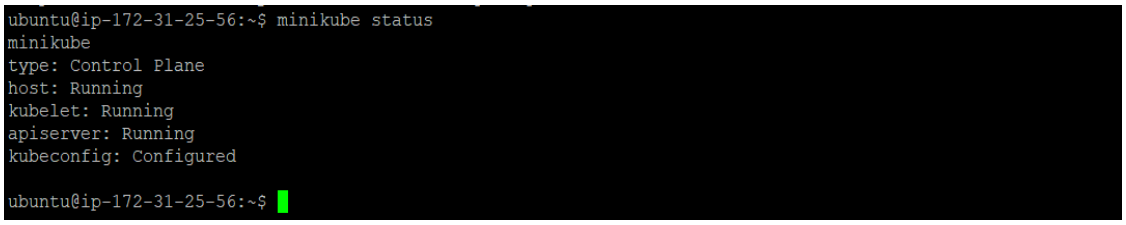
```
ubuntu@ip-172-31-25-56:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 77.3M  100 77.3M    0     0  46.8M      0  0:00:01  0:00:01 --:--:-- 46.8M
ubuntu@ip-172-31-25-56:~$
```

4. Assign permissions by using command **sudo usermod -aG docker $USER && newgrp docker** and perform "**minikube start –driver=docker**" command, then check the running status of minikube by **docker ps** command.



```
ubuntu@ip-172-31-25-56:~$ sudo usermod -aG docker $USER && newgrp docker
ubuntu@ip-172-31-25-56:~$ minikube start --driver-docker
Error: unknown flag: --driver-docker
See 'minikube start --help' for usage.
ubuntu@ip-172-31-25-56:~$ minikube start --driver=docker
* minikube v1.29.0 on Ubuntu 22.04 (xen/amd64)
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...
* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
* 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-172-31-25-56:~$ docker ps
CONTAINER ID   IMAGE                                 COMMAND                  CREATED         STATUS         PORTS
                                                        NAMES
a270d74779c0   gcr.io/k8s-minikube/kicbase:v0.0.37   "/usr/local/bin/entr…"   2 minutes ago   Up 2 minutes   127.0.0.1:49157->22/tcp, 127.0.0.1:49156->2376/tcp,
27.0.0.1:49154->8443/tcp, 127.0.0.1:49153->32443/tcp   minikube
ubuntu@ip-172-31-25-56:~$
```

5. Once minikube start finishes, run the command below to check the status of the cluster: **minikube status**



```
ubuntu@ip-172-31-25-56:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

ubuntu@ip-172-31-25-56:~$
```

If you want to try an alternative way,

To install Minikube on Ubuntu, you can follow these steps:

1. Install the dependencies: **sudo apt-get update && sudo apt-get install -y curl**
2. Download the latest version of the Minikube binary: **curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64**
3. Add the executable permission to the binary: **sudo chmod +x minikube-linux-amd64**
4. Move the binary to the /usr/local/bin directory: **sudo mv minikube-linux-amd64 /usr/local/bin/minikube**
5. Verify that Minikube is installed correctly by running the command: **minikube version**

If the installation is successful, the **minikube version** command should output the version of Minikube that was installed.

Let's understand the concept pod

In Kubernetes, a Pod is the smallest deployable unit that can be created and managed. A Pod represents a single instance of a running process in a cluster, and it can contain one or more containers that share the same network namespace, storage, and other resources.

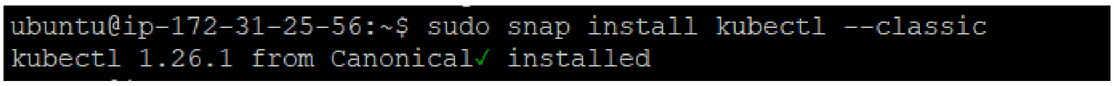
Pods are designed to be ephemeral and disposable, which means that they can be easily created, destroyed, and replaced by the Kubernetes cluster as needed. Pods are commonly used to run containerized applications and microservices, and they provide a layer of abstraction between the application and the underlying infrastructure, enabling developers to focus on writing code without worrying about the underlying infrastructure details.

A Pod can have one or more containers, which are tightly coupled and share the same resources and network namespace. Containers in a Pod are scheduled together and share the same lifecycle, meaning that they are started and stopped together. This makes it easy to manage the application as a single unit, even if it is composed of multiple containers.

Task-02:

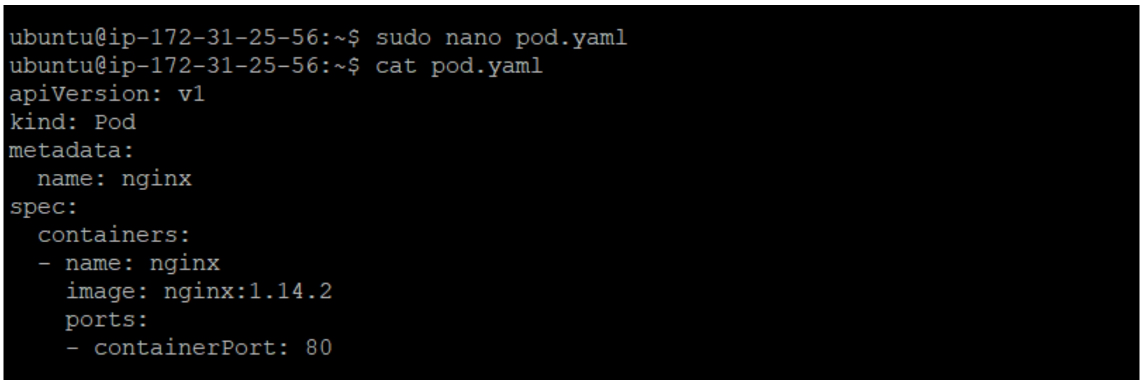
Create your first pod on Kubernetes through minikube.

1. Install kubectl by using following command **sudo snap install kubectl --classic command.**



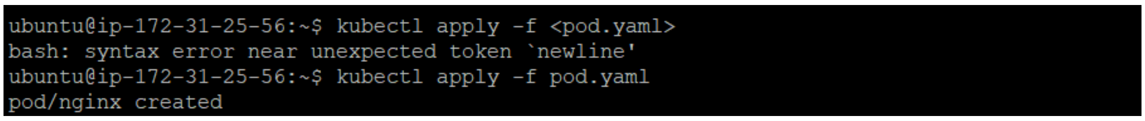
```
ubuntu@ip-172-31-25-56:~$ sudo snap install kubectl --classic
kubectl 1.26.1 from Canonical✓ installed
```

2. Create a folder, inside folder create pod.yaml file for nginx.



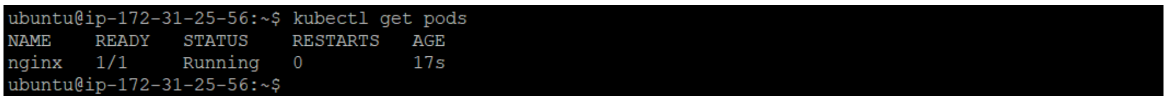
```
ubuntu@ip-172-31-25-56:~$ sudo nano pod.yaml
ubuntu@ip-172-31-25-56:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

3. To create a pod using pod.yaml file use below command: **kubectl apply -f pod.yaml**



```
ubuntu@ip-172-31-25-56:~$ kubectl apply -f <pod.yaml>
bash: syntax error near unexpected token `newline'
ubuntu@ip-172-31-25-56:~$ kubectl apply -f pod.yaml
pod/nginx created
```

4. To check list of pods: **kubectl get pods**



```
ubuntu@ip-172-31-25-56:~$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          17s
ubuntu@ip-172-31-25-56:~$
```

Happy Learning :)