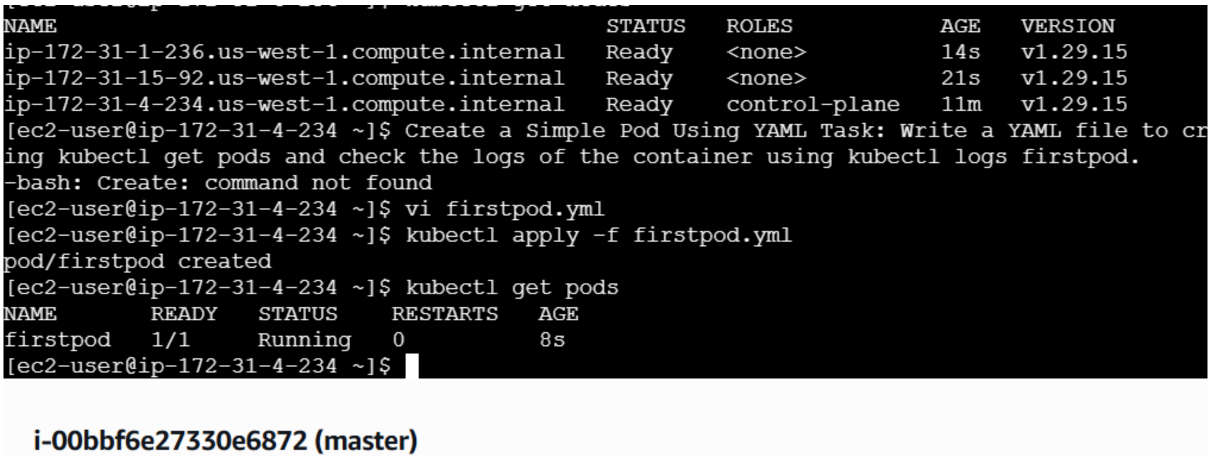
1 Create a Simple Pod Using YAML Task: Write a YAML file to create a Pod named firstpod with an nginx container. Verify the Pod creation using kubectl get pods and check the logs of the container using kubectl logs firstpod

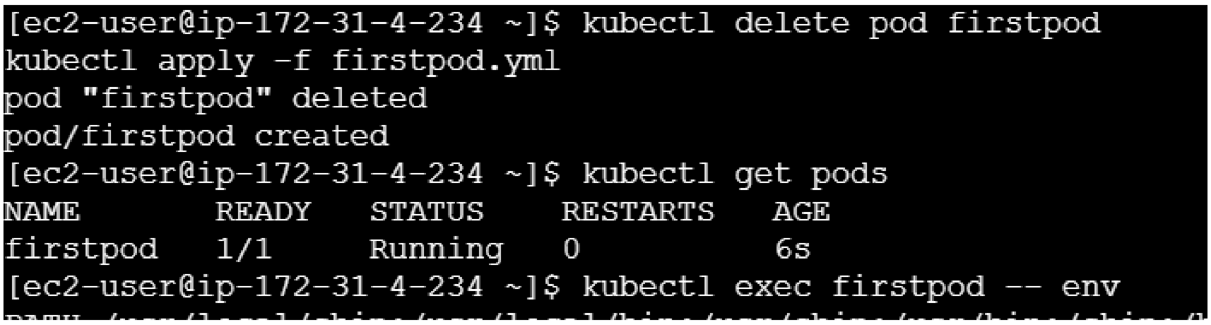
- Create a firstpod.yml file
- Add yml file
- apiVersion: v1
- kind: Pod
- metadata:
-   name: firstpod
- spec:
-   containers:
-   - name: nginx
-     image: nginx
-     ports:
-     - containerPort: 80
- Now create a pod
- kubectl apply -f firstpod.yml
- check pods
- kubectl get pods



```
NAME                                         STATUS   ROLES           AGE   VERSION
ip-172-31-1-236.us-west-1.compute.internal   Ready    <none>          14s   v1.29.15
ip-172-31-15-92.us-west-1.compute.internal   Ready    <none>          21s   v1.29.15
ip-172-31-4-234.us-west-1.compute.internal   Ready    control-plane   11m   v1.29.15
[ec2-user@ip-172-31-4-234 ~]$ Create a Simple Pod Using YAML Task: Write a YAML file to cr
ing kubectl get pods and check the logs of the container using kubectl logs firstpod.
-bash: Create: command not found
[ec2-user@ip-172-31-4-234 ~]$ vi firstpod.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f firstpod.yml
pod/firstpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          8s
[ec2-user@ip-172-31-4-234 ~]$
```

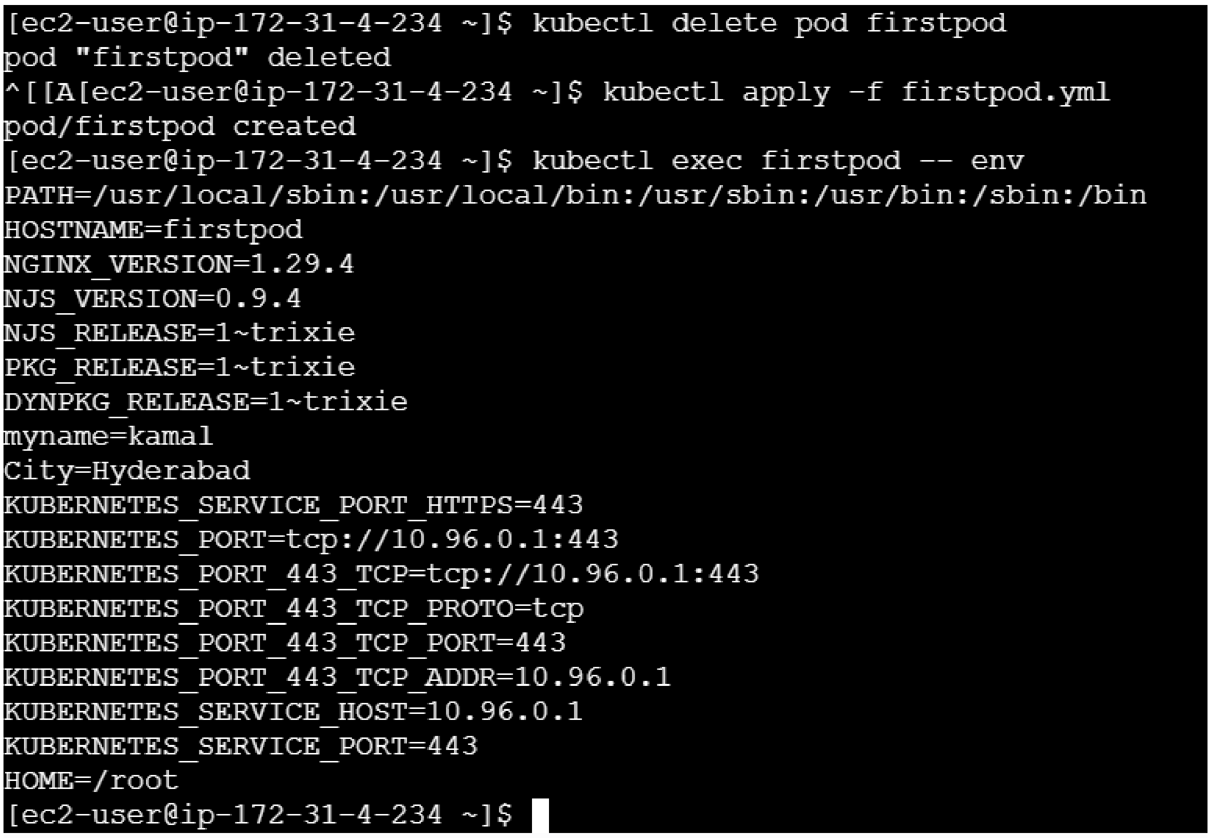
- i-00bbf6e27330e6872 (master)

1. 2 . Set Environment Variables in a Pod Task: Modify the YAML file to include environment variables myname: sabair and City: Hyderabad. Deploy the Pod and use kubectl exec <pod_name> -- env to check if the environment variables are set properly.
    - Edit the firstpod.yml file
    - Delete old and
    - Create new pod trhe yml.file



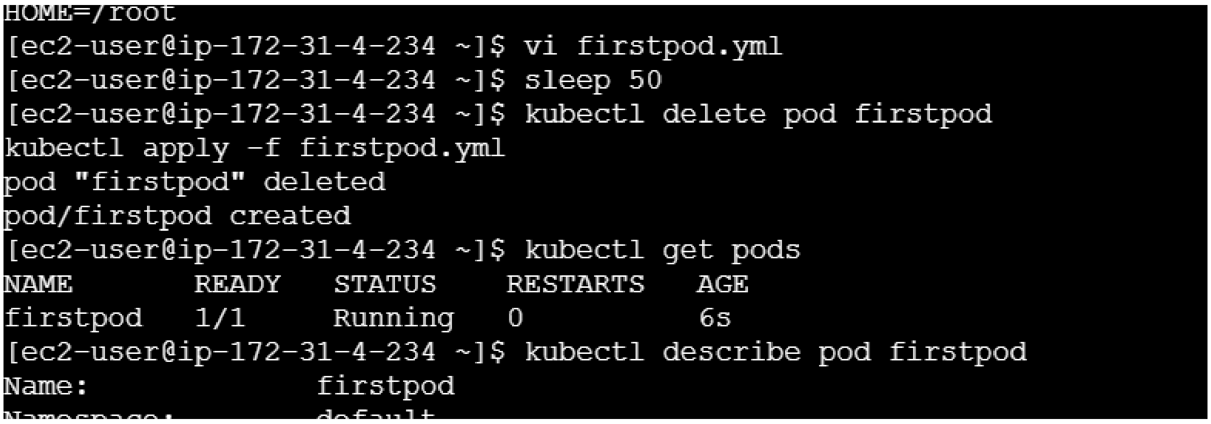
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl delete pod firstpod
kubectl apply -f firstpod.yml
pod "firstpod" deleted
pod/firstpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          6s
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec firstpod -- env
```

- Check environments

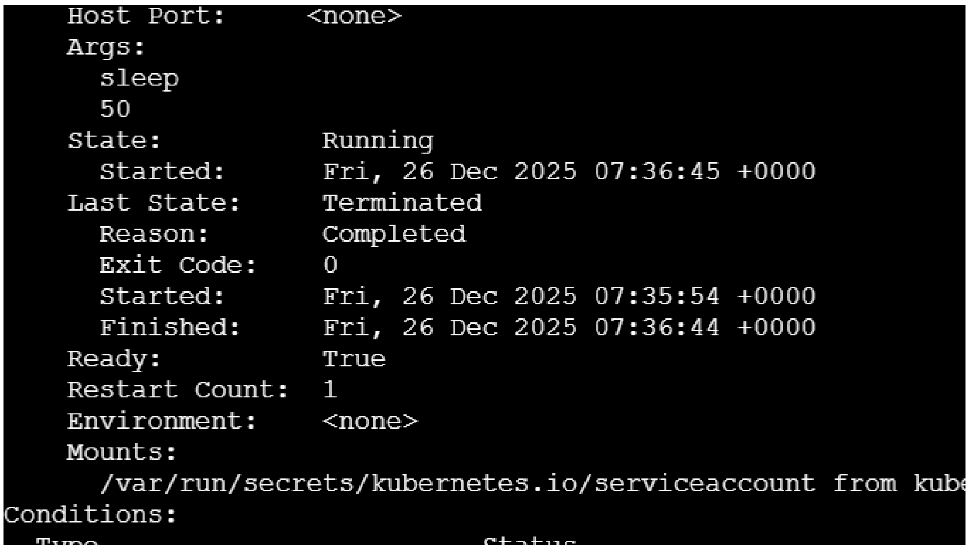


```
[ec2-user@ip-172-31-4-234 ~]$ kubectl delete pod firstpod
pod "firstpod" deleted
^[[A[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f firstpod.yml
pod/firstpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec firstpod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=firstpod
NGINX_VERSION=1.29.4
NJS_VERSION=0.9.4
NJS_RELEASE=1~trixie
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
myname=kamal
City=Hyderabad
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
HOME=/root
[ec2-user@ip-172-31-4-234 ~]$
```

-
- 3. Deploy a Pod with Commands (Args) in YAML Task: Modify the YAML file to add args that instruct the container to sleep for 50 seconds. Deploy the Pod and use kubectl describe pod to verify the args are correctly passed to the container.
- Vi firstpod.yml
- Write yml file
- apiVersion: v1
- kind: Pod
- metadata:
-   name: firstpod
- spec:
-   containers:
-   - name: nginx

- image: nginx
- args:
- - sleep
- - "50"
- 

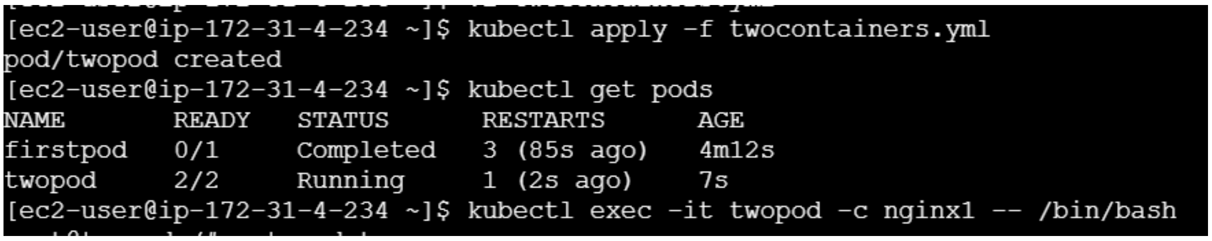
```
HOME=/root
[ec2-user@ip-172-31-4-234 ~]$ vi firstpod.yml
[ec2-user@ip-172-31-4-234 ~]$ sleep 50
[ec2-user@ip-172-31-4-234 ~]$ kubectl delete pod firstpod
kubectl apply -f firstpod.yml
pod "firstpod" deleted
pod/firstpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
firstpod    1/1     Running   0          6s
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod firstpod
Name:             firstpod
Namespace:        default
```

- Now,
- Create pod
- kubectl apply -f firstpod.yml
- now
- verify arugs by
- kubectl describe pod firstpod
- check for args section
- 

```
    Host Port:     <none>
    Args:
      sleep
      50
    State:          Running
      Started:      Fri, 26 Dec 2025 07:36:45 +0000
    Last State:     Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Fri, 26 Dec 2025 07:35:54 +0000
      Finished:     Fri, 26 Dec 2025 07:36:44 +0000
    Ready:          True
    Restart Count:  1
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube
Conditions:
  Type                        Status
```

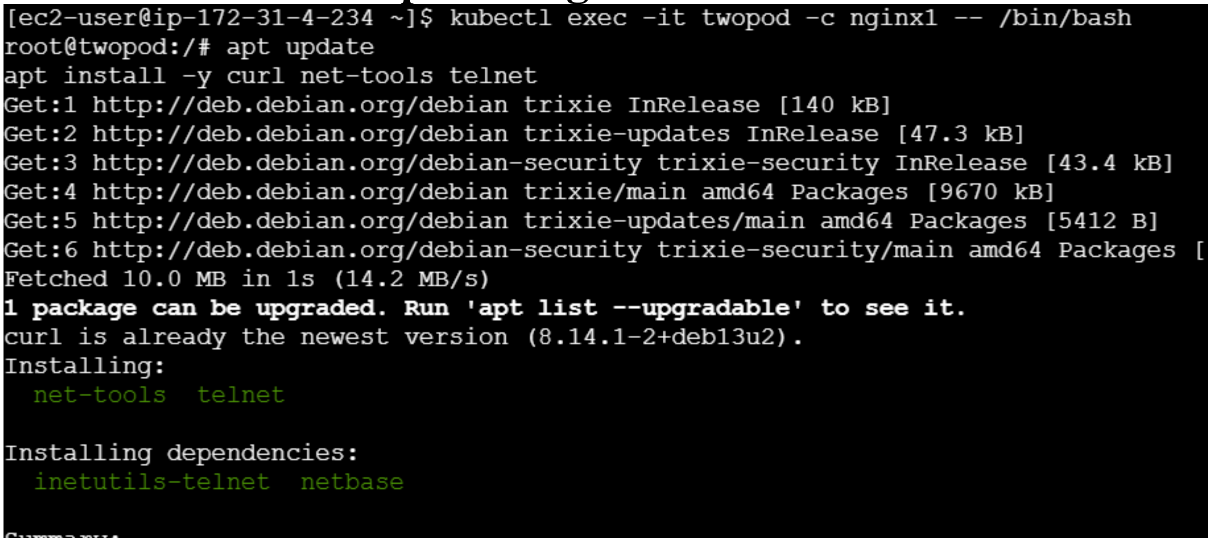
4 . Create a Pod with Two Containers Task: Create a YAML file to define a Pod with two nginx containers inside. Use kubectl exec to access both containers and verify that both containers can communicate through the same network (e.g., using telnet between them).

- vi twocontainers.yml
- write a yml file
- apiVersion: v1
- kind: Pod
- metadata:
-   name: twopod
- spec:
-   containers:
-   - name: nginx1
-     image: nginx
-     ports:
-     - containerPort: 80
-
-   - name: nginx2
-     image: nginx
-     ports:
-     - containerPort: 81
- **Deploy the Pod**
- **Kubectrl apply -f twocontainer.yml**
- Kubectl get pods



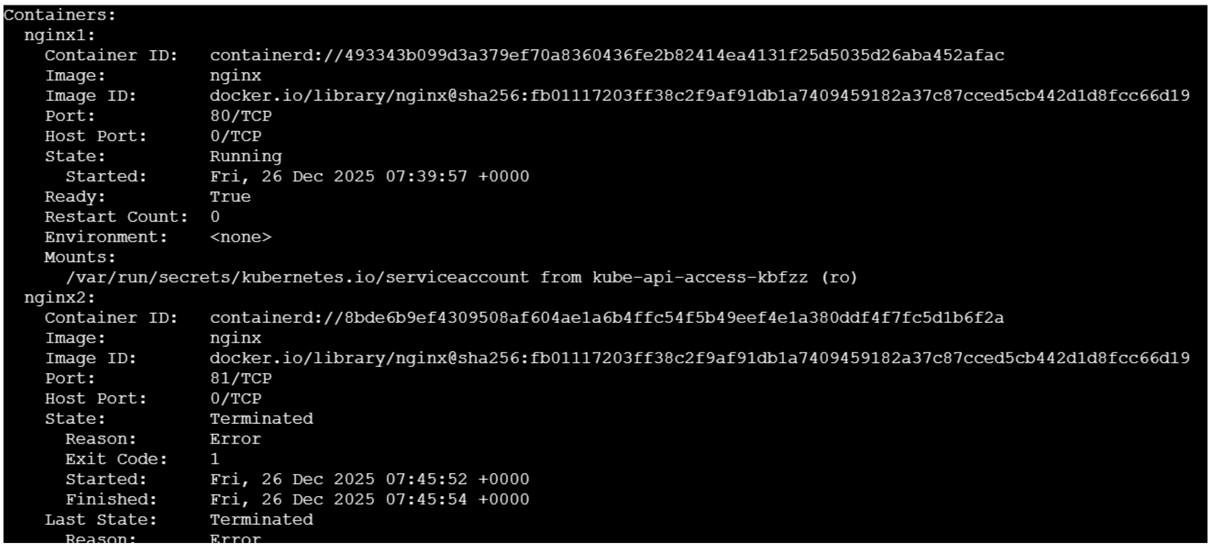
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f twocontainers.yml
pod/twopod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME        READY   STATUS      RESTARTS      AGE
firstpod    0/1     Completed   3 (85s ago)   4m12s
twopod      2/2     Running     1 (2s ago)    7s
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it twopod -c nginx1 -- /bin/bash
```

- **Two containers in one pod**
- Access one container
- **kubectl exec -it twopod -c nginx1 -- /bin/bash**



```
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it twopod -c nginx1 -- /bin/bash
root@twopod:/# apt update
apt install -y curl net-tools telnet
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4 kB]
Get:4 http://deb.debian.org/debian trixie/main amd64 Packages [9670 kB]
Get:5 http://deb.debian.org/debian trixie-updates/main amd64 Packages [5412 B]
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Packages [
Fetched 10.0 MB in 1s (14.2 MB/s)
1 package can be upgraded. Run 'apt list --upgradable' to see it.
curl is already the newest version (8.14.1-2+deb13u2).
Installing:
  net-tools  telnet

Installing dependencies:
  inetutils-telnet  netbase
```

- **Two containers**

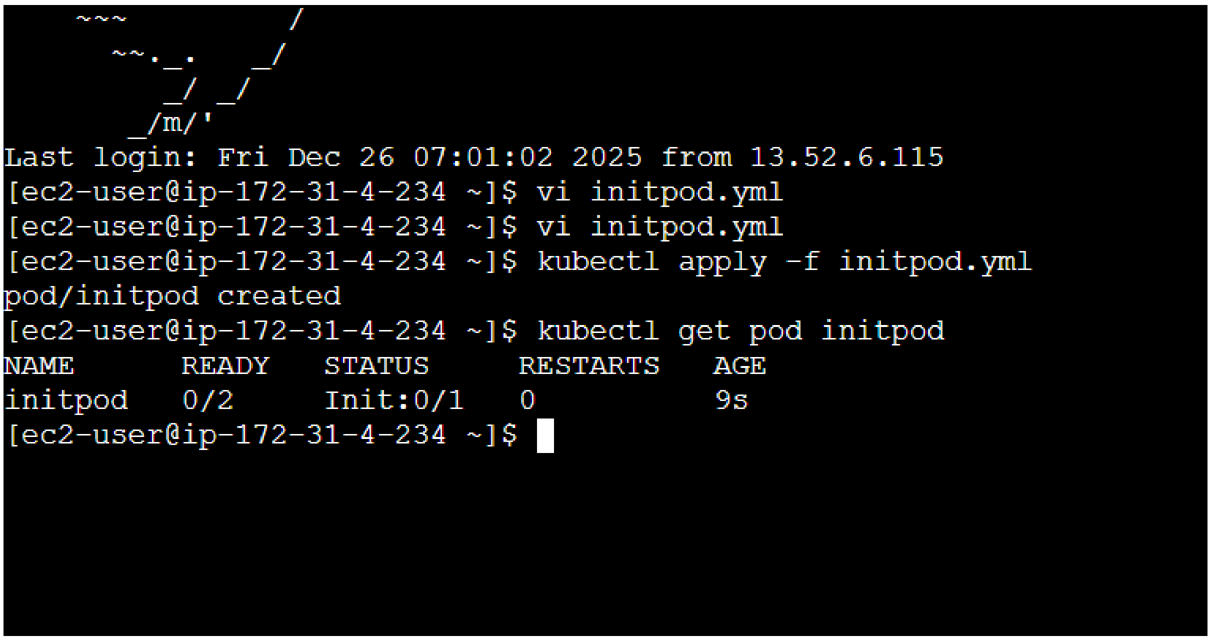


```
Containers:
  nginx1:
    Container ID:   containerd://493343b099d3a379ef70a8360436fe2b82414ea4131f25d5035d26aba452afac
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 26 Dec 2025 07:39:57 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-kbfzz (ro)
  nginx2:
    Container ID:   containerd://8bde6b9ef4309508af604ae1a6b4ffc54f5b49eef4e1a380ddf4f7fc5d1b6f2a
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
    Port:           81/TCP
    Host Port:      0/TCP
    State:          Terminated
      Reason:       Error
      Exit Code:    1
      Started:      Fri, 26 Dec 2025 07:45:52 +0000
      Finished:     Fri, 26 Dec 2025 07:45:54 +0000
    Last State:     Terminated
      Reason:       Error
```

1. **5 . Set Up an Init Container in a Pod Task: Modify the YAML to include an init container that sleeps for**

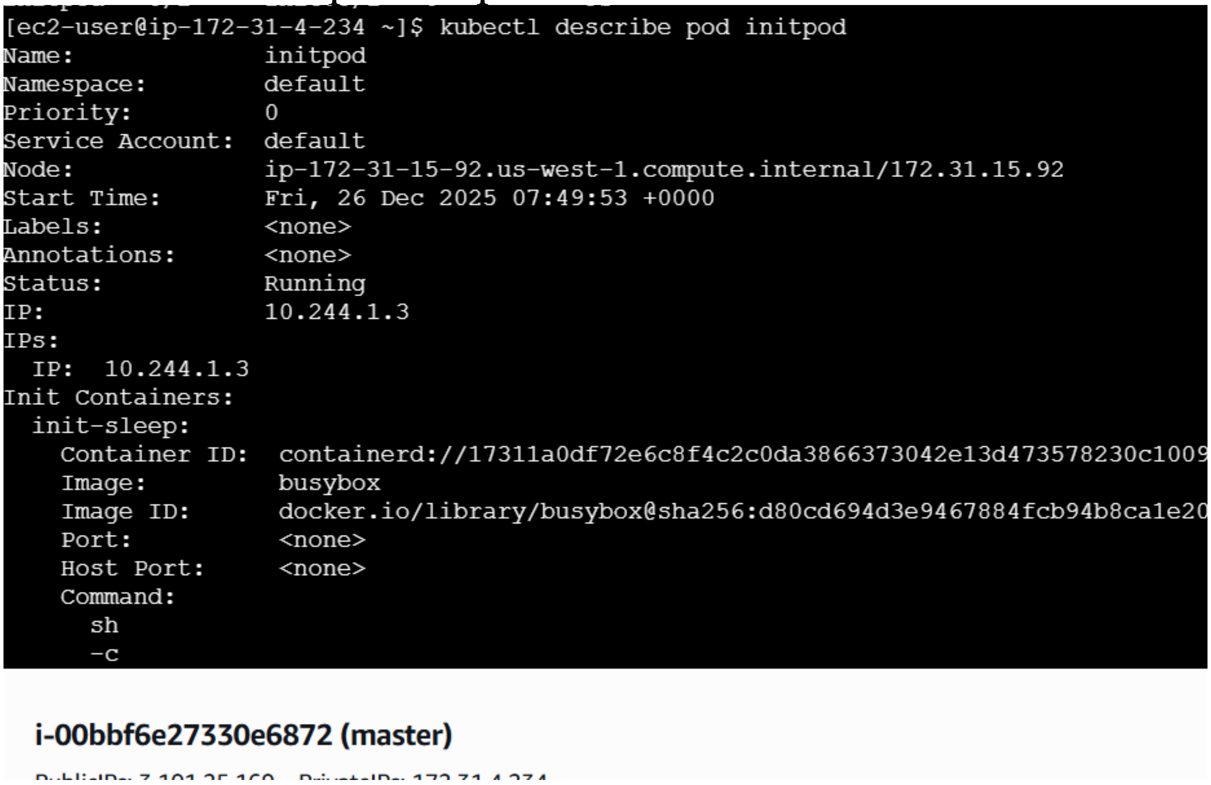
30 seconds before the main containers start. Verify the init container's execution using kubectl describe pod and check the logs to confirm its completion.

- Vi initpod.yml
- apiVersion: v1
- kind: Pod
- metadata:
-   name: initpod
- spec:
-   initContainers:
-   - name: init-sleep
-     image: busybox
-     command: ["sh", "-c", "echo Init container started; sleep 30; echo Init container completed"]
-
-   containers:
-   - name: nginx1
-     image: nginx
-     command: ["sh", "-c", "sleep 3600"]
-
-   - name: nginx2
-     image: nginx
-     command: ["sh", "-c", "sleep 3600"]
- Now
- kubectl apply -f initpod.yml

- 

```
    ~~~          /
      ~~._.    _/
         _/ _/
       _/m/'
Last login: Fri Dec 26 07:01:02 2025 from 13.52.6.115
[ec2-user@ip-172-31-4-234 ~]$ vi initpod.yml
[ec2-user@ip-172-31-4-234 ~]$ vi initpod.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f initpod.yml
pod/initpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod initpod
NAME      READY   STATUS     RESTARTS   AGE
initpod   0/2     Init:0/1   0          9s
[ec2-user@ip-172-31-4-234 ~]$
```

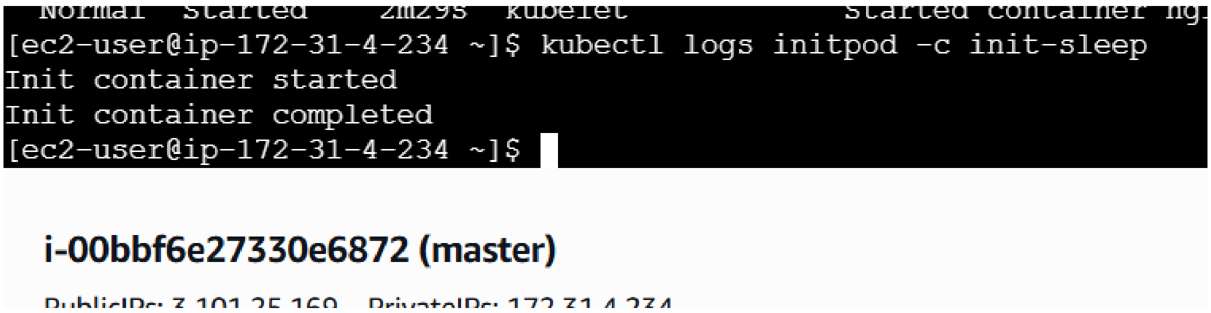
- Verify init container
- kubectl describe pod initpod



```
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod initpod
Name:             initpod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-15-92.us-west-1.compute.internal/172.31.15.92
Start Time:       Fri, 26 Dec 2025 07:49:53 +0000
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.1.3
IPs:
  IP:  10.244.1.3
Init Containers:
  init-sleep:
    Container ID:  containerd://17311a0df72e6c8f4c2c0da3866373042e13d473578230c1009
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:d80cd694d3e9467884fcb94b8ca1e20
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
```

i-00bbf6e27330e6872 (master)

- Init logs



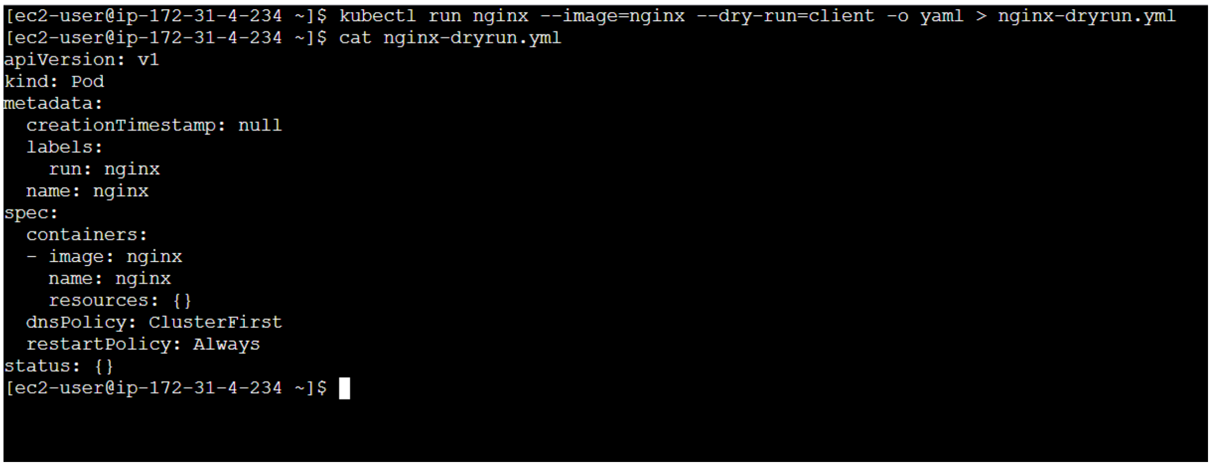
```
  Normal   Started     2m29s   kubelet            Started container ng
[ec2-user@ip-172-31-4-234 ~]$ kubectl logs initpod -c init-sleep
Init container started
Init container completed
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

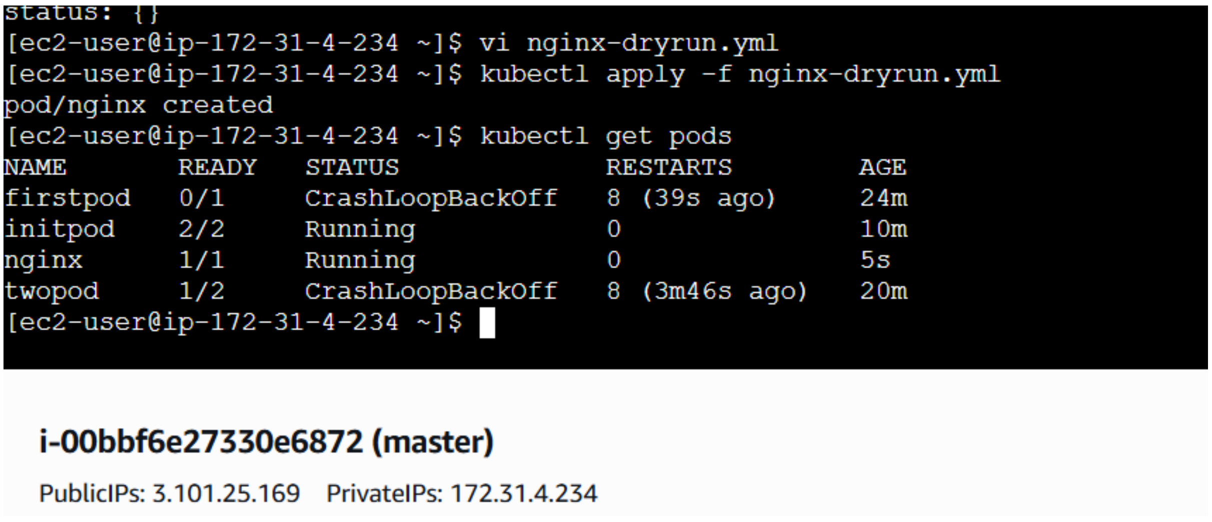
-
-
- kubectl apply -f initpod.yml

6 . Run a Dry Run Command to Generate YAML Task: Use the kubectl run nginx --image=nginx --dry-run=client -o yaml command to generate a Pod YAML definition. Modify the generated YAML to suit specific requirements (e.g., labels or environment variables) and deploy it.

- kubectl run nginx --image=nginx --dry-run=client -o yaml > nginx-dryrun.yml
- pod not created but
- yaml file created
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl run nginx --image=nginx --dry-run=client -o yaml > nginx-dryrun.yml
[ec2-user@ip-172-31-4-234 ~]$ cat nginx-dryrun.yml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
[ec2-user@ip-172-31-4-234 ~]$
```

- Create pods
- Kubectl apply -f nginx-dryrun.yml

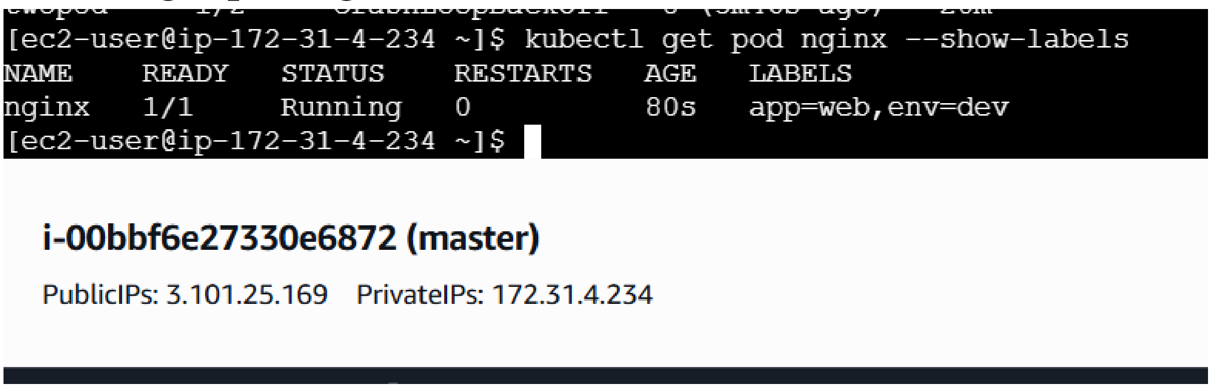


```
status: {}
[ec2-user@ip-172-31-4-234 ~]$ vi nginx-dryrun.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f nginx-dryrun.yml
pod/nginx created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME        READY   STATUS             RESTARTS        AGE
firstpod    0/1     CrashLoopBackOff   8 (39s ago)     24m
initpod     2/2     Running            0               10m
nginx       1/1     Running            0               5s
twopod      1/2     CrashLoopBackOff   8 (3m46s ago)   20m
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

-
- Kubectl get pods
- Nginx running
- Verify labels
- kubectl get pod nginx --show-labels

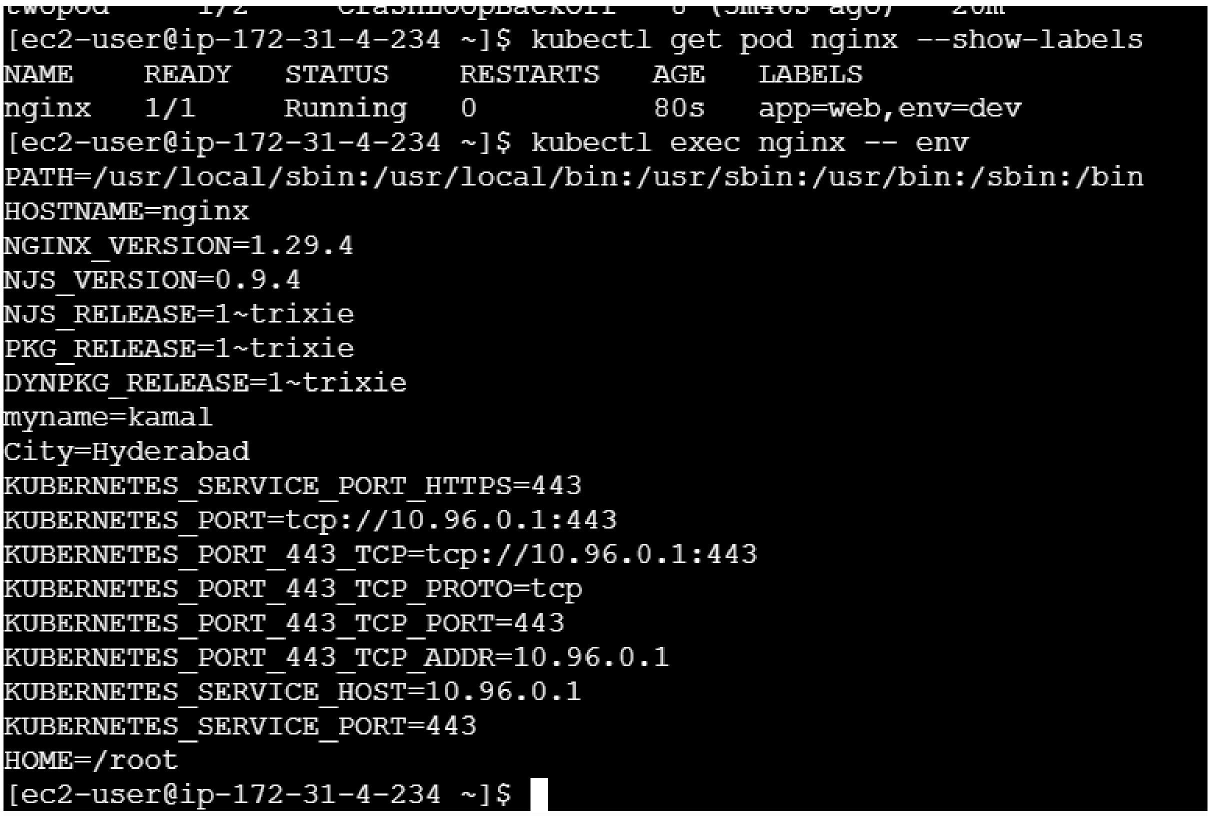


```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod nginx --show-labels
NAME    READY   STATUS    RESTARTS   AGE   LABELS
nginx   1/1     Running   0          80s   app=web,env=dev
[ec2-user@ip-172-31-4-234 ~]$
```

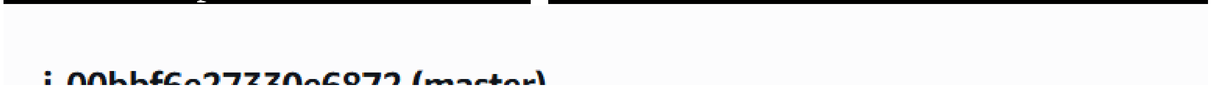
i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

-
- Verify environmetnals variables
- kubectl exec nginx – env



```
twopod     1/2     CrashLoopBackOff    8 (3m40s ago)    20m
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod nginx --show-labels
NAME     READY   STATUS    RESTARTS   AGE   LABELS
nginx    1/1     Running   0          80s   app=web,env=dev
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec nginx -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx
NGINX_VERSION=1.29.4
NJS_VERSION=0.9.4
NJS_RELEASE=1~trixie
PKG_RELEASE=1~trixie
DYNPKG_RELEASE=1~trixie
myname=kamal
City=Hyderabad
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
HOME=/root
[ec2-user@ip-172-31-4-234 ~]$
```

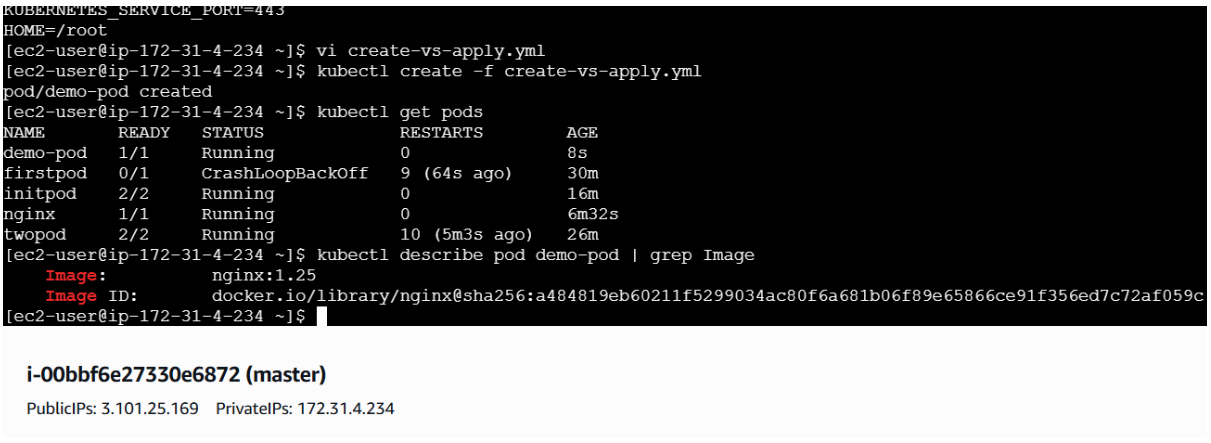
- 

  i-00bbf6e27330e6873 (master)

1. 7 . Use kubectl apply vs kubectl create Task: Create a YAML file to define a Pod. First, deploy it using kubectl create -f <file_name>.yml and then modify the YAML (e.g., change the image version). Use kubectl apply to redeploy and verify the difference between both commands.

   - Vi create vs apply
   - Add yml file
   - apiVersion: v1
   - kind: Pod
   - metadata:
   -   name: demo-pod

- spec:
-   containers:
-   - name: nginx
-     image: nginx:1.25
- Create a pod
- Kubectl apply -f createvs apply.yml
- Now
- Kubectl get pods

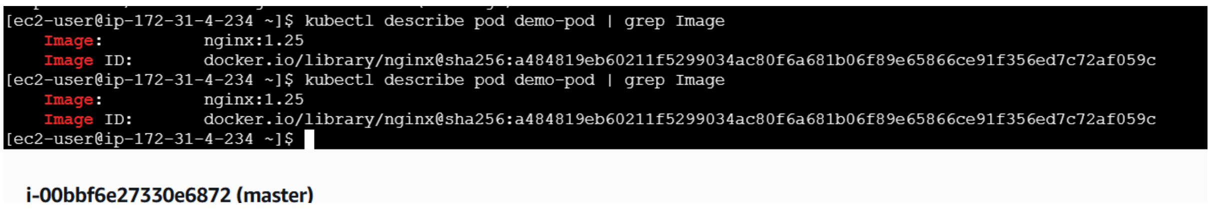


```
KUBERNETES_SERVICE_PORT=443
HOME=/root
[ec2-user@ip-172-31-4-234 ~]$ vi create-vs-apply.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl create -f create-vs-apply.yml
pod/demo-pod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME        READY   STATUS             RESTARTS       AGE
demo-pod    1/1     Running            0              8s
firstpod    0/1     CrashLoopBackOff   9 (64s ago)    30m
initpod     2/2     Running            0              16m
nginx       1/1     Running            0              6m32s
twopod      2/2     Running            10 (5m3s ago)  26m
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod demo-pod | grep Image
    Image:         nginx:1.25
    Image ID:      docker.io/library/nginx@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

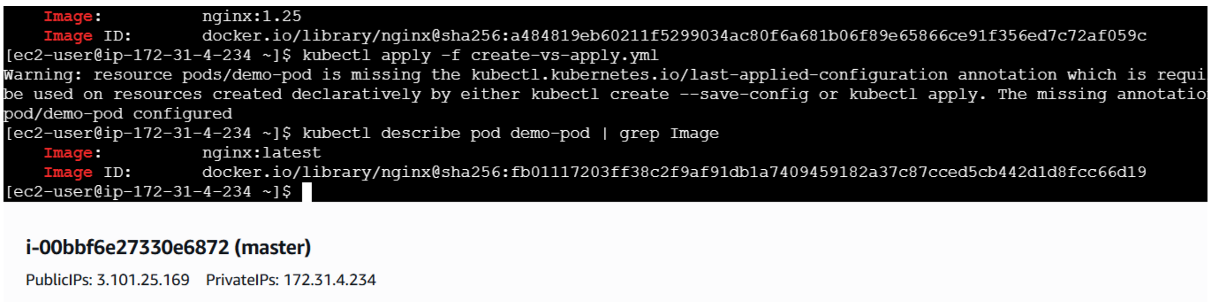
PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

- Check images
- kubectl describe pod demo-pod | grep Image



```
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod demo-pod | grep Image
    Image:         nginx:1.25
    Image ID:      docker.io/library/nginx@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod demo-pod | grep Image
    Image:         nginx:1.25
    Image ID:      docker.io/library/nginx@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c
[ec2-user@ip-172-31-4-234 ~]$
```

- i-00bbf6e27330e6872 (master)
- Now change image name to latest
- And redoploy
- kubectl apply -f create-vs-apply.yml
- and check for the image



```
    Image:          nginx:1.25
    Image ID:       docker.io/library/nginx@sha256:a484819eb60211f5299034ac80f6a681b06f89e65866ce91f356ed7c72af059c
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f create-vs-apply.yml
Warning: resource pods/demo-pod is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is requi
be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotatio
pod/demo-pod configured
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe pod demo-pod | grep Image
    Image:          nginx:latest
    Image ID:       docker.io/library/nginx@sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
[ec2-user@ip-172-31-4-234 ~]$
```

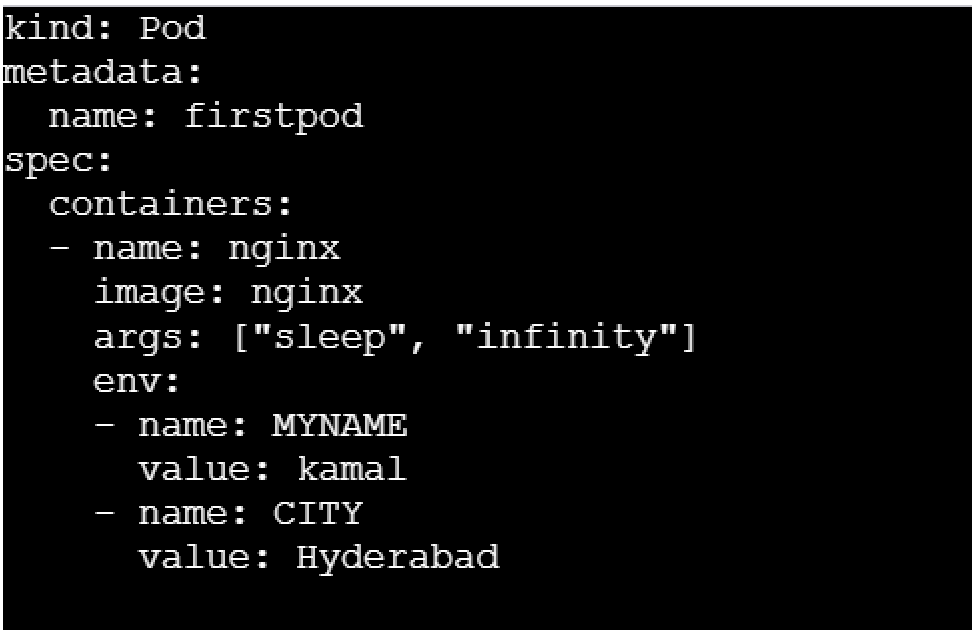
i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

•

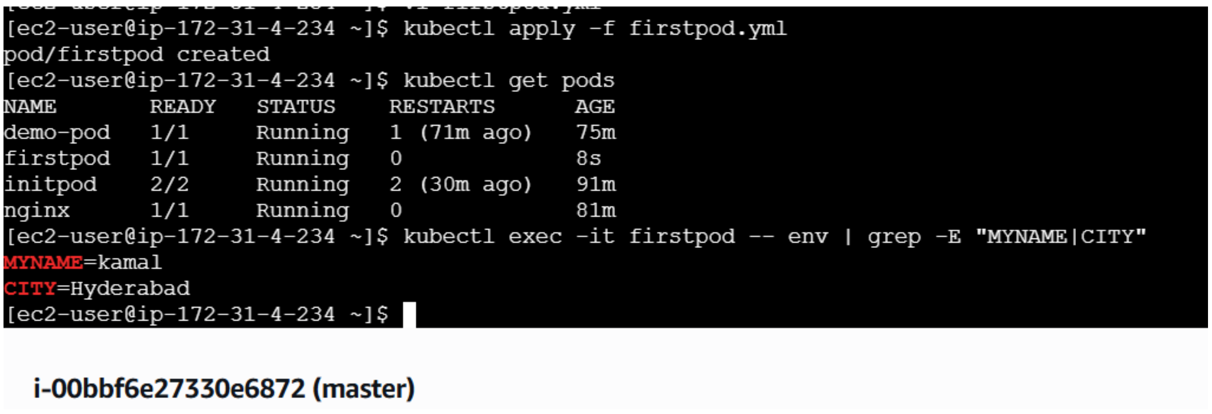
1. 8 . Edit an Existing Pod Configuration Task: Use kubectl edit pod <pod_name> to modify the running Pod's environment variables or image. After making the changes, verify if they took effect by checking the container logs or environment variables using kubectl exec.

- Write a yaml file
- apiVersion: v1
- kind: Pod
- metadata:
-   name: firstpod
- spec:
-   containers:
-   - name: nginx
-     image: nginx
-     args: ["sleep", "infinity"]
-     env:
-     - name: MYNAME
-       value: kamal
-     - name: CITY
-       value: Hyderabad
- Now
- Create a pod



```
kind: Pod
metadata:
  name: firstpod
spec:
  containers:
  - name: nginx
    image: nginx
    args: ["sleep", "infinity"]
    env:
    - name: MYNAME
      value: kamal
    - name: CITY
      value: Hyderabad
```

- kubectl apply -f firstpod.yml
- check for ppds
- kubectl get pods
- verify environmental variables
- kubectl exec -it firstpod -- env | grep -E "MYNAME|CITY"



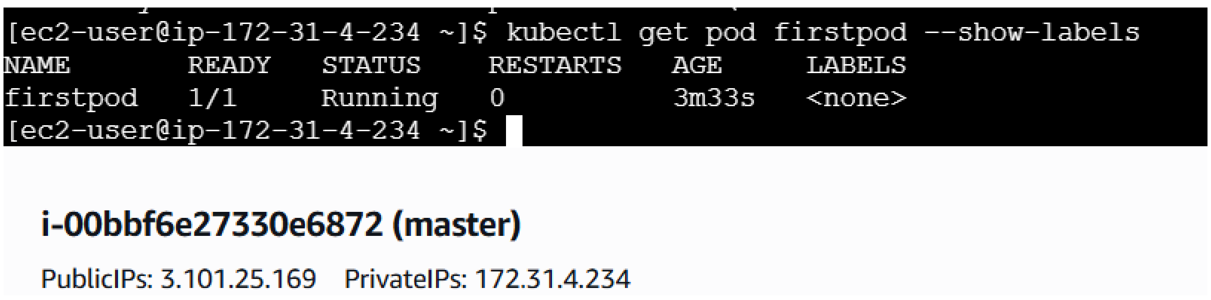
```
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f firstpod.yml
pod/firstpod created
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pods
NAME        READY   STATUS    RESTARTS      AGE
demo-pod    1/1     Running   1 (71m ago)   75m
firstpod    1/1     Running   0             8s
initpod     2/2     Running   2 (30m ago)   91m
nginx       1/1     Running   0             81m
[ec2-user@ip-172-31-4-234 ~]$ kubectl exec -it firstpod -- env | grep -E "MYNAME|CITY"
MYNAME=kamal
CITY=Hyderabad
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

1. 9 . Expose a Pod Using a Service Task: Create a YAML file to expose your firstpod using a Service (ClusterIP).

Ensure that your service is exposing the Pod on port 80 and verify it using kubectl get svc.

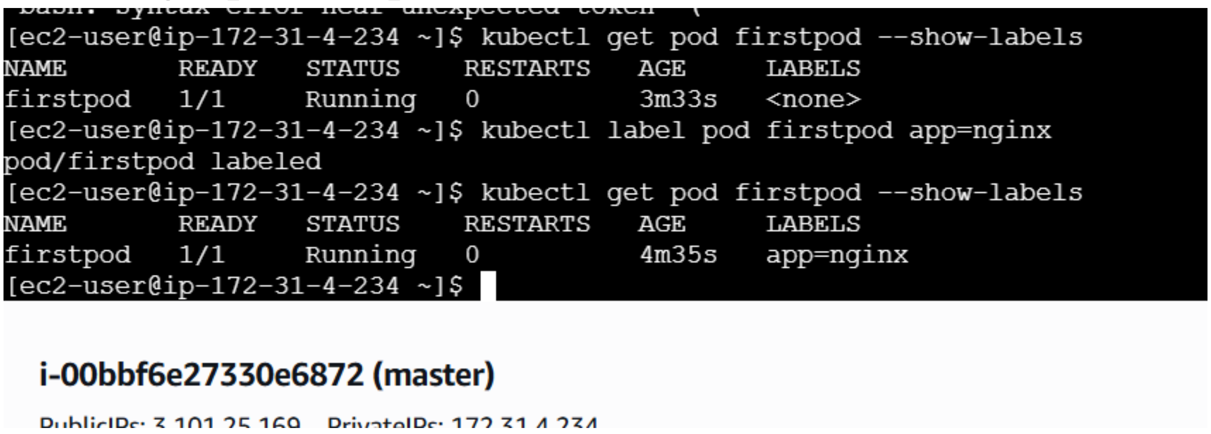
- kubectl get pod firstpod --show-labels
- check labels



```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod firstpod --show-labels
NAME        READY   STATUS    RESTARTS   AGE     LABELS
firstpod    1/1     Running   0          3m33s   <none>
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

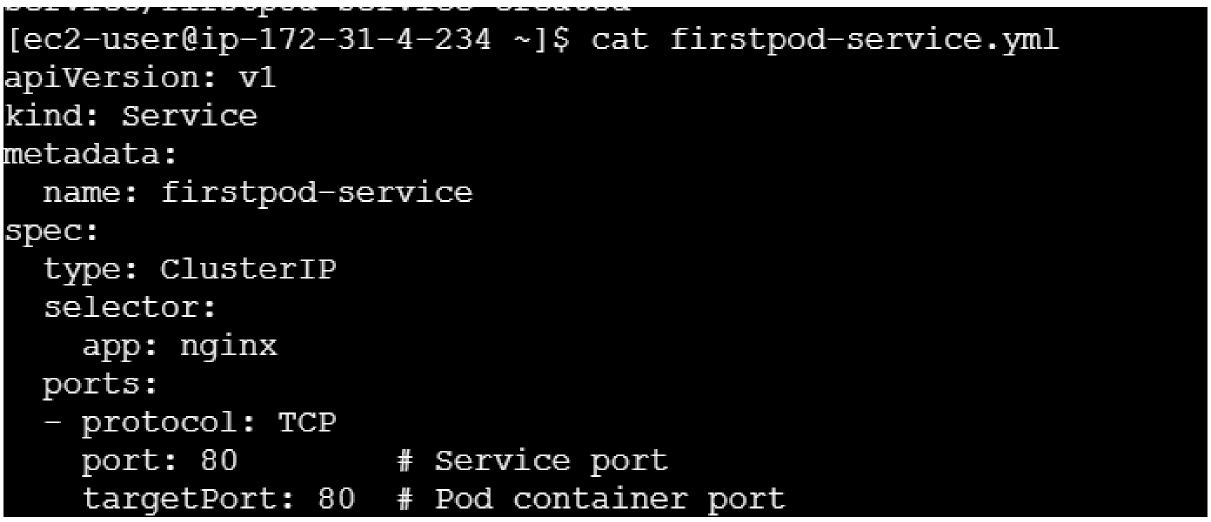
- no lables present
- so add labels
- kubectl label pod firstpod app=nginx
- now
- verify labels
- kubectl get pod firstpod --show-labels
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod firstpod --show-labels
NAME        READY   STATUS    RESTARTS   AGE     LABELS
firstpod    1/1     Running   0          3m33s   <none>
[ec2-user@ip-172-31-4-234 ~]$ kubectl label pod firstpod app=nginx
pod/firstpod labeled
[ec2-user@ip-172-31-4-234 ~]$ kubectl get pod firstpod --show-labels
NAME        READY   STATUS    RESTARTS   AGE     LABELS
firstpod    1/1     Running   0          4m35s   app=nginx
[ec2-user@ip-172-31-4-234 ~]$
```

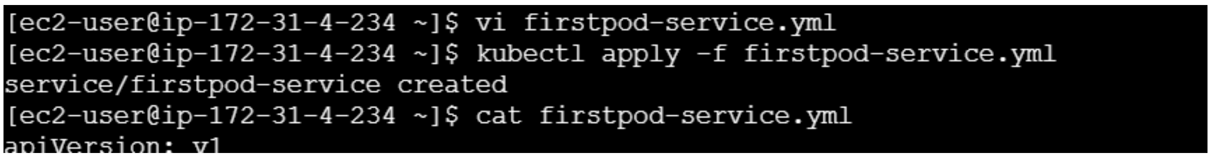
i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

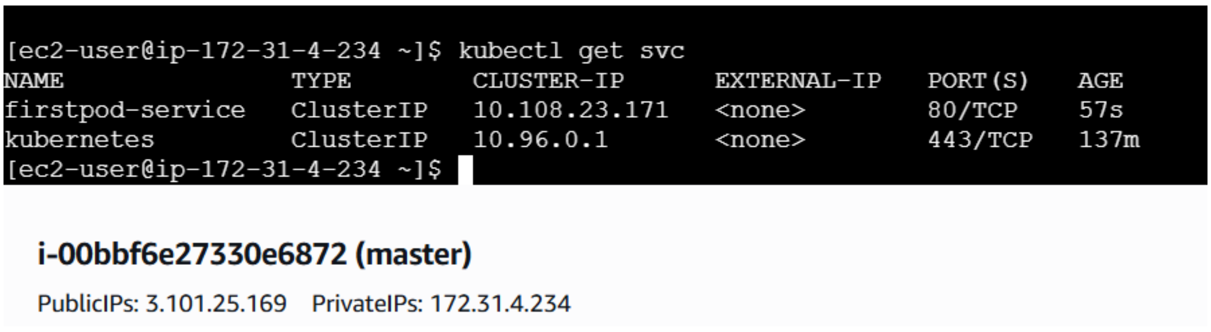
- Now create a servie file for first pod.yml
- apiVersion: v1
- kind: Service
- metadata:
-   name: firstpod-service

- spec:
-   type: ClusterIP
-   selector:
-     app: nginx
-   ports:
-   - protocol: TCP
-     port: 80   # Service port
-     targetPort: 80 # Pod container port
- 

```
[ec2-user@ip-172-31-4-234 ~]$ cat firstpod-service.yml
apiVersion: v1
kind: Service
metadata:
  name: firstpod-service
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80        # Service port
    targetPort: 80  # Pod container port
```

- Now create a pod
- kubectl apply -f firstpod-service.yml
- 

```
[ec2-user@ip-172-31-4-234 ~]$ vi firstpod-service.yml
[ec2-user@ip-172-31-4-234 ~]$ kubectl apply -f firstpod-service.yml
service/firstpod-service created
[ec2-user@ip-172-31-4-234 ~]$ cat firstpod-service.yml
apiVersion: v1
```

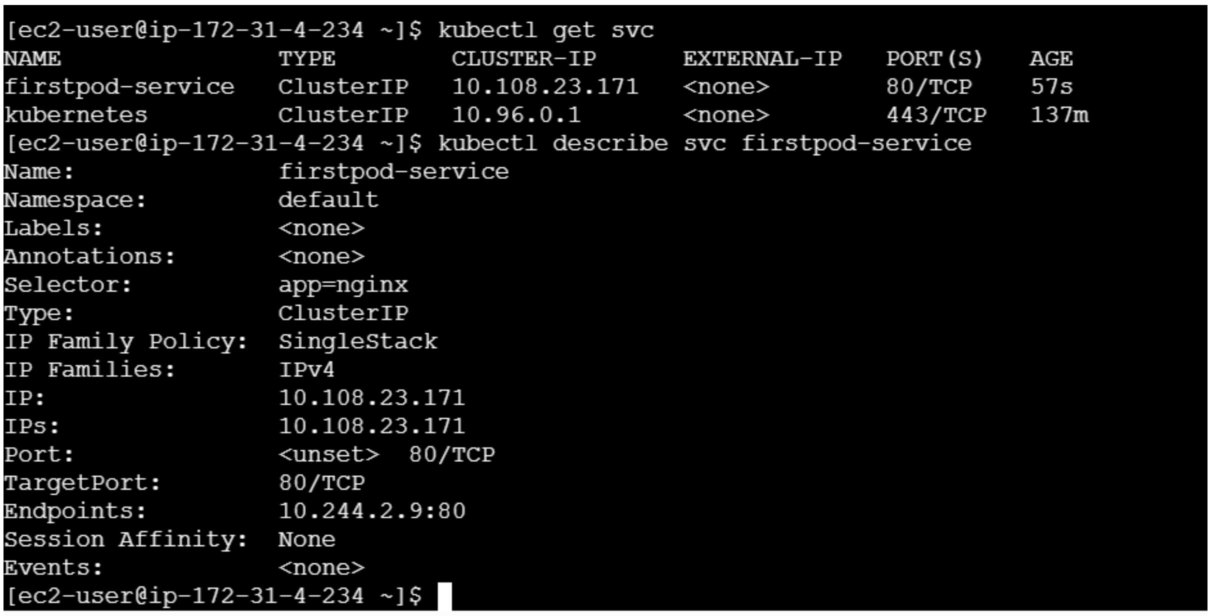
- Get service cfreation
- kubectl get svc
- 

```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get svc
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
firstpod-service   ClusterIP   10.108.23.171   <none>        80/TCP    57s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP   137m
[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

- Pod mapping
- kubectl describe svc firstpod-service

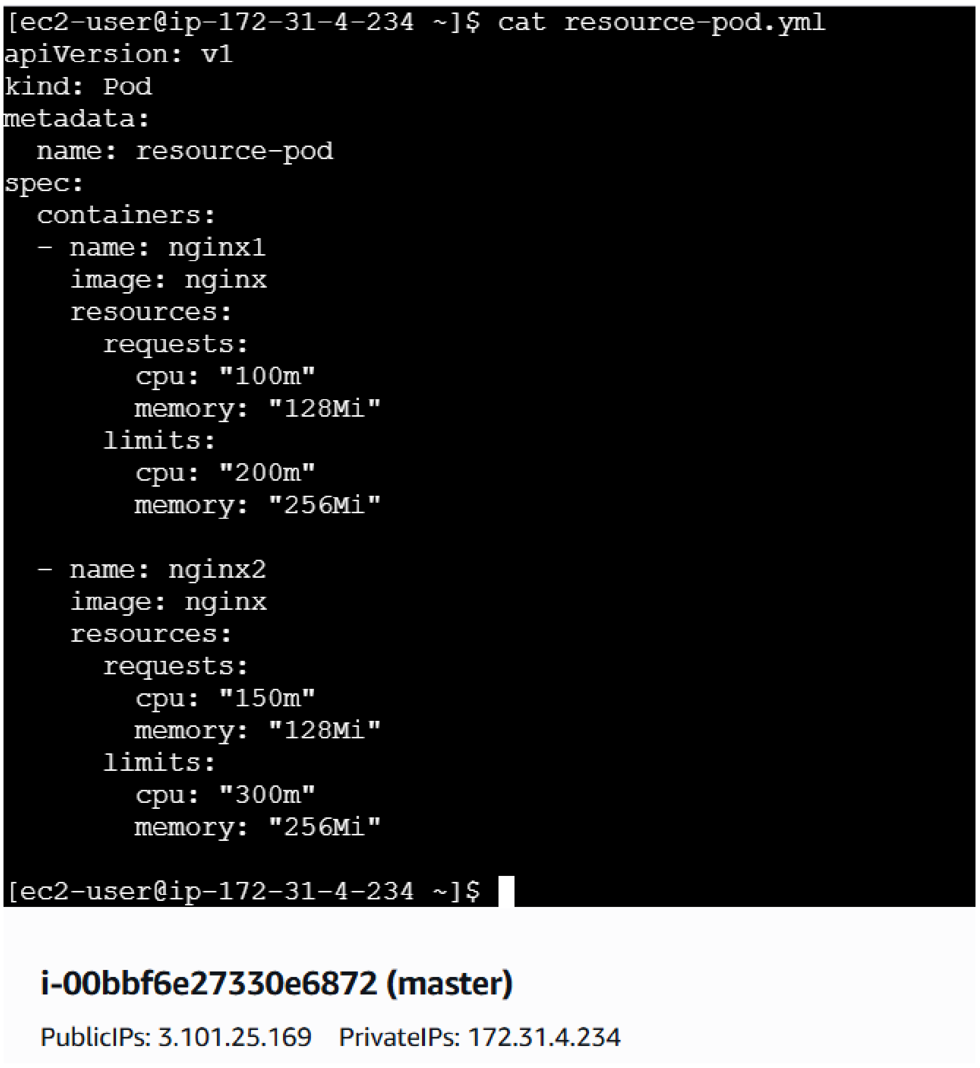


```
[ec2-user@ip-172-31-4-234 ~]$ kubectl get svc
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
firstpod-service   ClusterIP   10.108.23.171   <none>        80/TCP    57s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP   137m
[ec2-user@ip-172-31-4-234 ~]$ kubectl describe svc firstpod-service
Name:                firstpod-service
Namespace:           default
Labels:              <none>
Annotations:         <none>
Selector:            app=nginx
Type:                ClusterIP
IP Family Policy:    SingleStack
IP Families:         IPv4
IP:                  10.108.23.171
IPs:                 10.108.23.171
Port:                <unset>  80/TCP
TargetPort:          80/TCP
Endpoints:           10.244.2.9:80
Session Affinity:    None
Events:              <none>
[ec2-user@ip-172-31-4-234 ~]$
```

- **i-00bbf6e27330e6872 (master)**
- Check endpoints.

1. 10 . Pod with Resource Limits and Requests Task: Add resource requests and limits to the containers in your YAML file. Specify CPU and memory requests/limits for both containers and deploy the Pod. Use kubectl describe pod to verify if the resource configurations are correctly applied.

- **Write a resource yaml file**

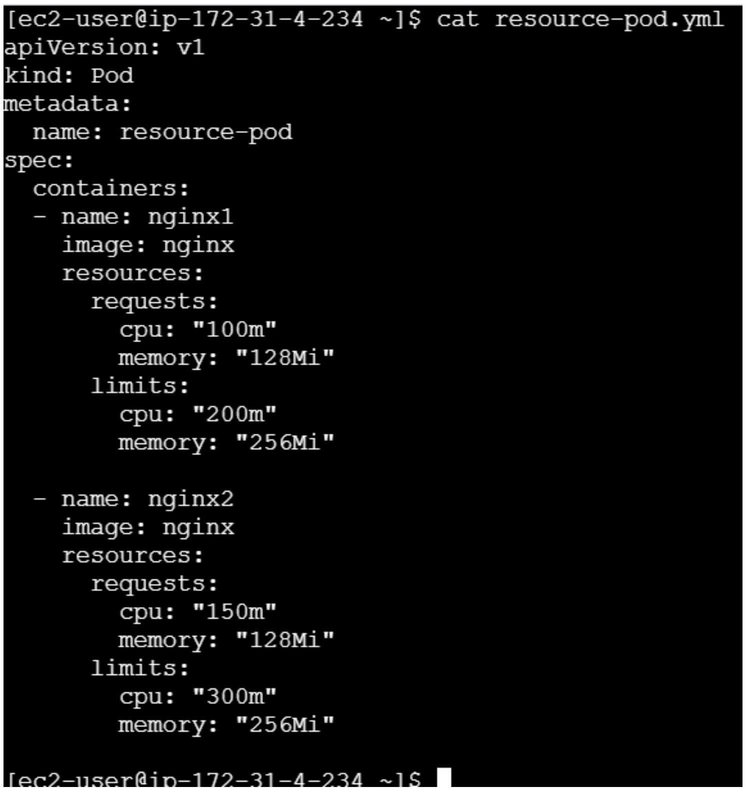


```
[ec2-user@ip-172-31-4-234 ~]$ cat resource-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: resource-pod
spec:
  containers:
  - name: nginx1
    image: nginx
    resources:
      requests:
        cpu: "100m"
        memory: "128Mi"
      limits:
        cpu: "200m"
        memory: "256Mi"

  - name: nginx2
    image: nginx
    resources:
      requests:
        cpu: "150m"
        memory: "128Mi"
      limits:
        cpu: "300m"
        memory: "256Mi"

[ec2-user@ip-172-31-4-234 ~]$
```

i-00bbf6e27330e6872 (master)

PublicIPs: 3.101.25.169   PrivateIPs: 172.31.4.234

- Deploy the pod
- kubectl apply -f resource-pod.yml
- now,
- kubectl get pods and check
- now
- now describe the pod by
- kubectl describe pod resource-pod

- 

```
[ec2-user@ip-172-31-4-234 ~]$ cat resource-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: resource-pod
spec:
  containers:
  - name: nginx1
    image: nginx
    resources:
      requests:
        cpu: "100m"
        memory: "128Mi"
      limits:
        cpu: "200m"
        memory: "256Mi"

  - name: nginx2
    image: nginx
    resources:
      requests:
        cpu: "150m"
        memory: "128Mi"
      limits:
        cpu: "300m"
        memory: "256Mi"

[ec2-user@ip-172-31-4-234 ~]$
```