**Kubernetes Task-2**

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**.

Step1: Create a YML file

Vi nginx-pod.yml

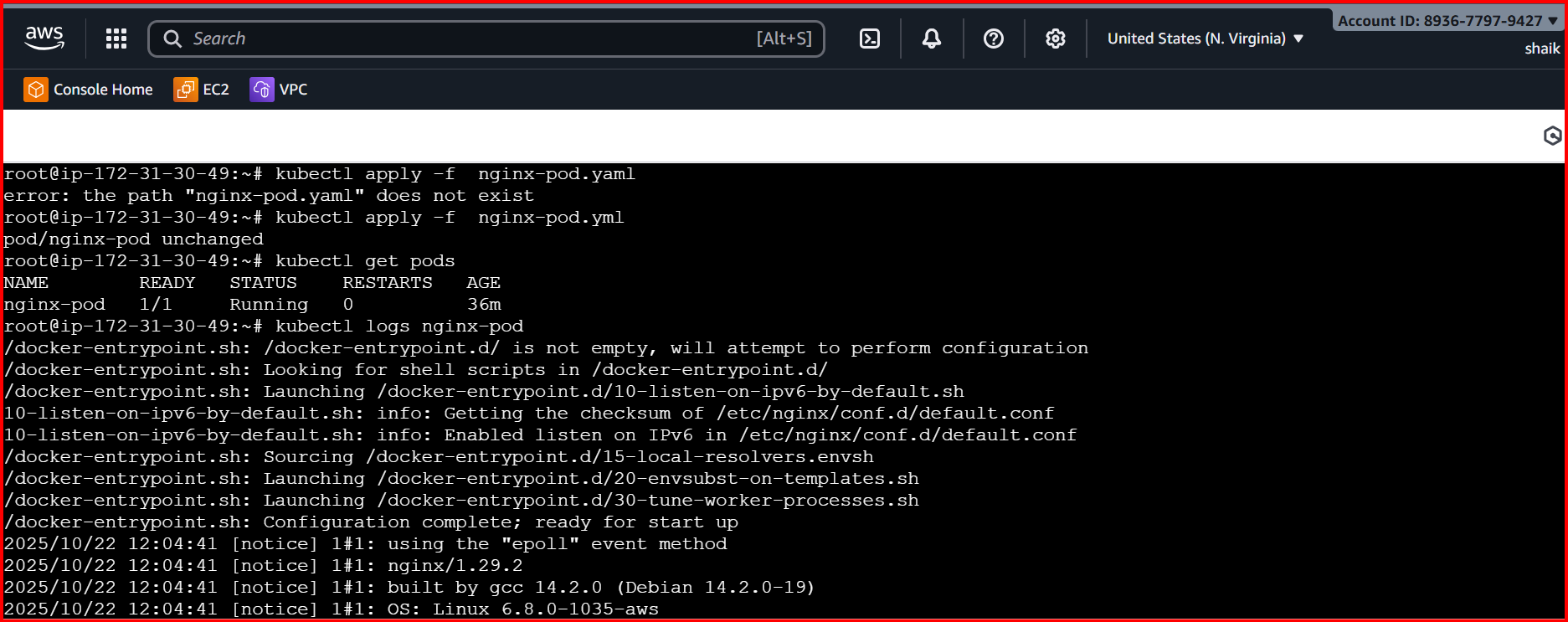


**Execute below commands to create and check logs of pod.**

kubectl apply -f nginx-pod.yml

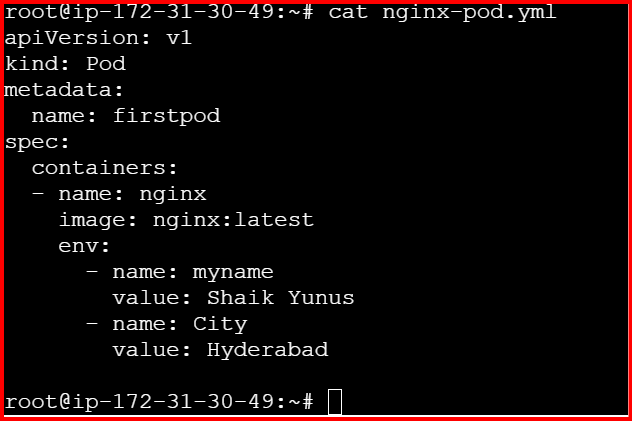
kubectl get pods

kubectl logs nginx-pod



**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.**

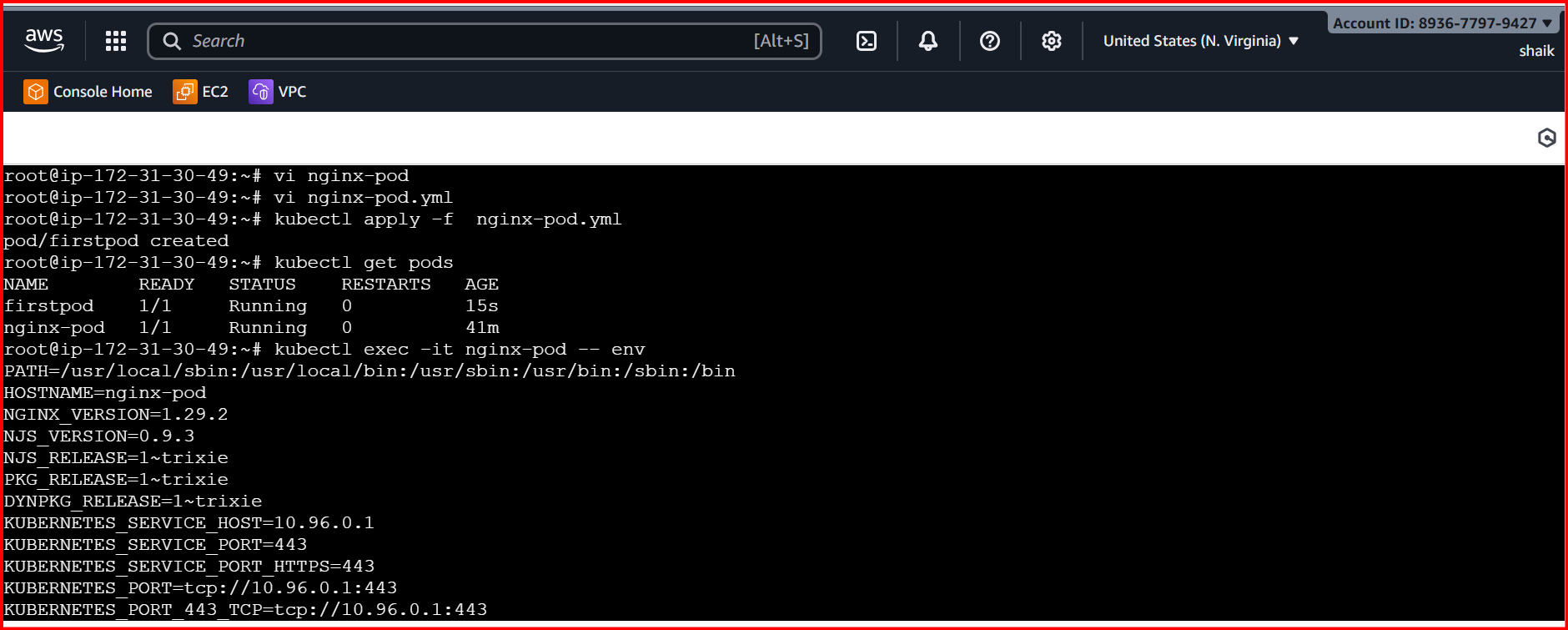
**vi nginx-pod**



**kubectl apply -f nginx-pod.yml**

**kubectl apply -f nginx-pod.yml**

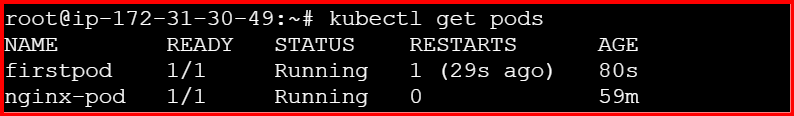
**kubectl exec -it nginx-pod -- env**



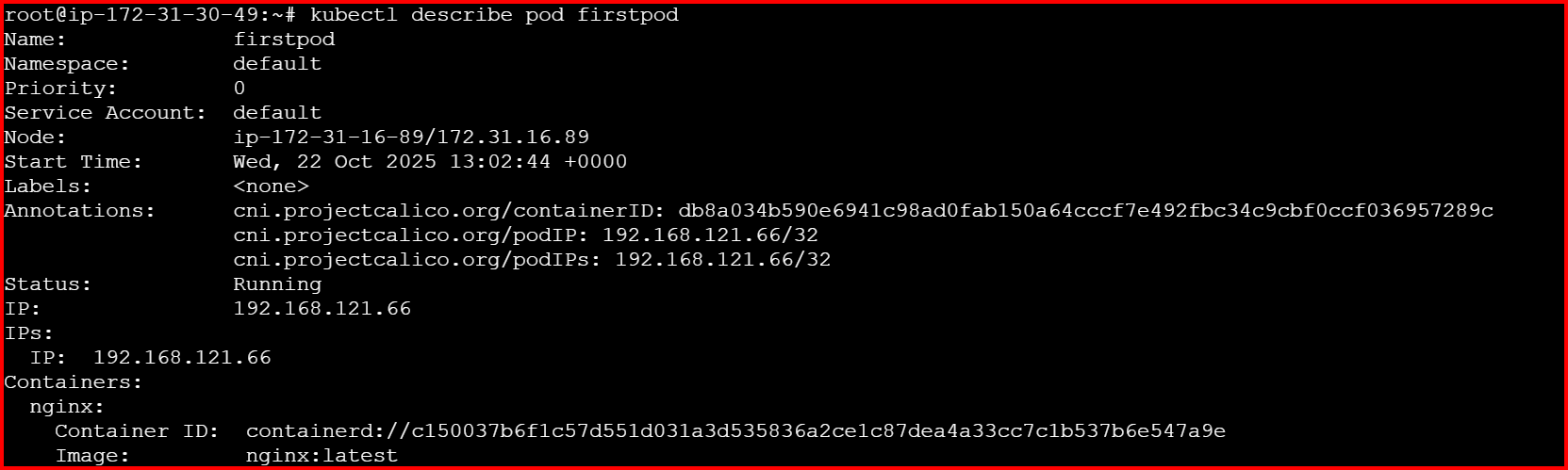
**3.Deploy a Pod with Commands (Args) in YAML Task: Modify the YAML file to add args that instruct the container to sleep for 50 seconds. Deploy the Pod and use kubectl describe pod to verify the args are correctly passed to the container.**

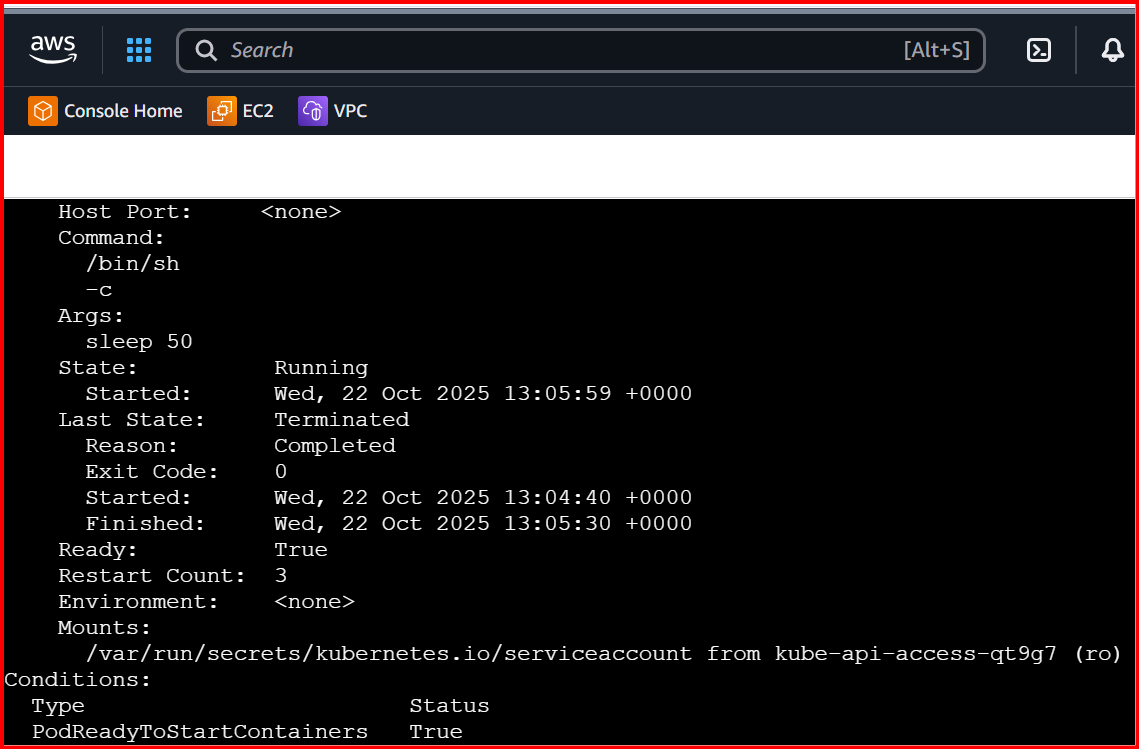
Deploy a Pod with Commands (Args)

**kubectl get pods**



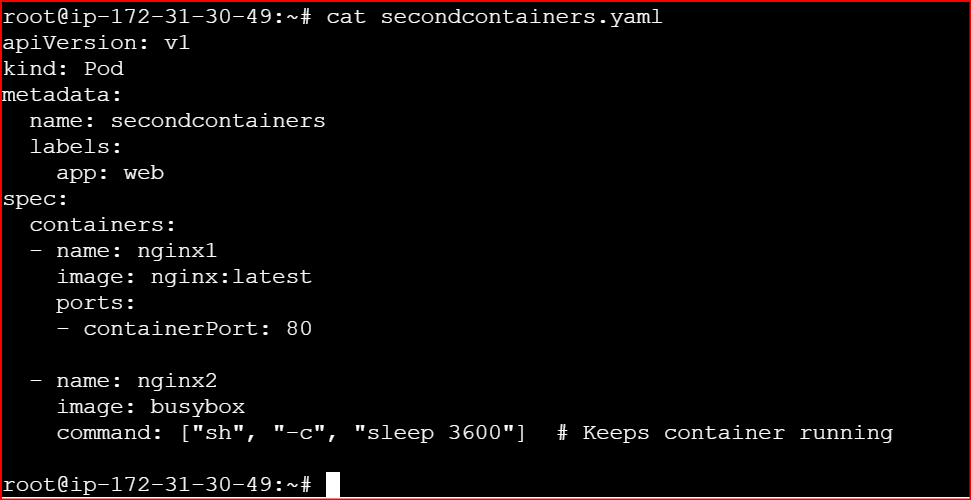
**Kubectl describe pod firstpod**





**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).**

**Create a pod secondcontainers.yaml**

****

apiVersion: v1

kind: Pod

metadata:

name: twocontainers

labels:

app: web

spec:

containers:

- name: nginx1

image: nginx:latest

ports:

- containerPort: 80

- name: nginx2

image: busybox

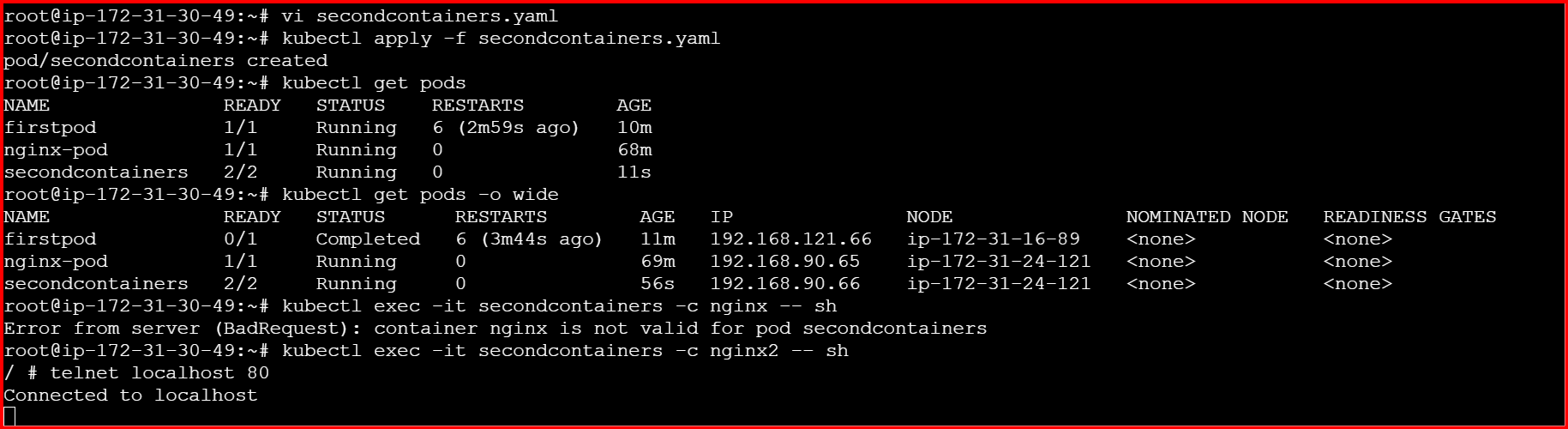
command: ["sh", "-c", "sleep 3600"] # Keeps container running

**kubectl apply -f secondcontainers.yaml**

**kubectl get pods**

**kubectl get pods -o wide**

**kubectl exec -it secondcontainers -c nginx2 -- sh**

****

**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.**

Create pod **vi pod-with-init.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-with-init

labels:

app: init-demo

spec:

# Step 1: Init container runs first and must complete before main container starts

initContainers:

- name: init-sleeper

image: busybox

command: ["sh", "-c", "echo 'Init container started...'; sleep 30; echo 'Init container completed.'"]

# Step 2: Main container starts only after init container completes

containers:

- name: nginx

image: nginx:latest

ports:

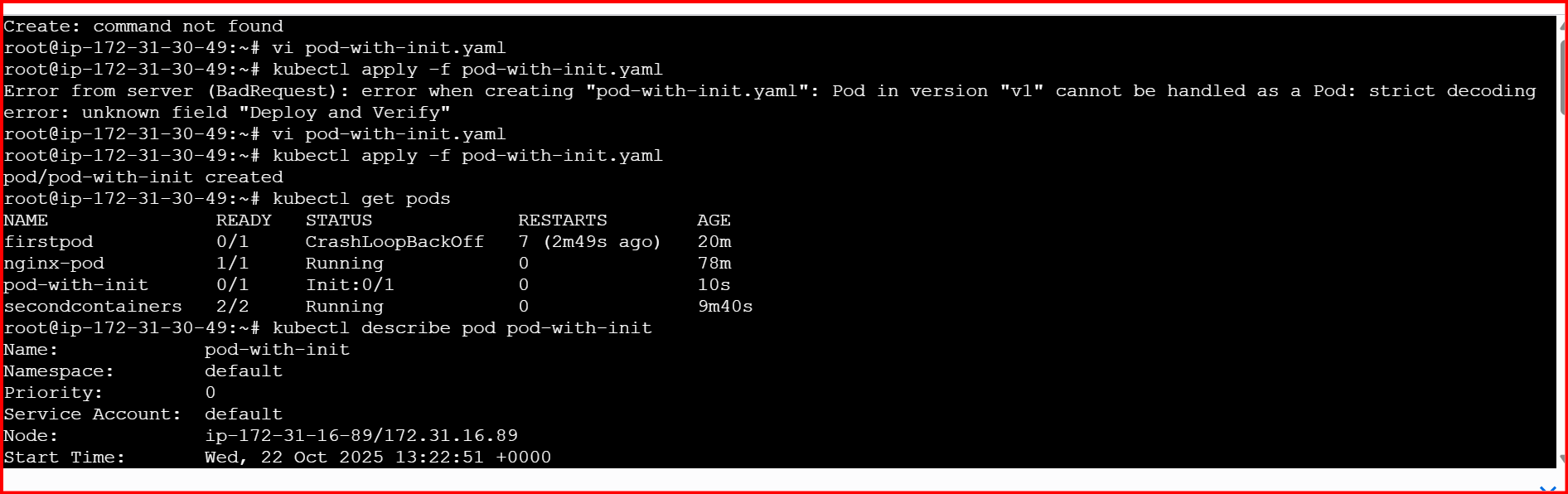
- containerPort: 80

**Deploy and Verify:**

**kubectl apply -f pod-with-init.yaml**

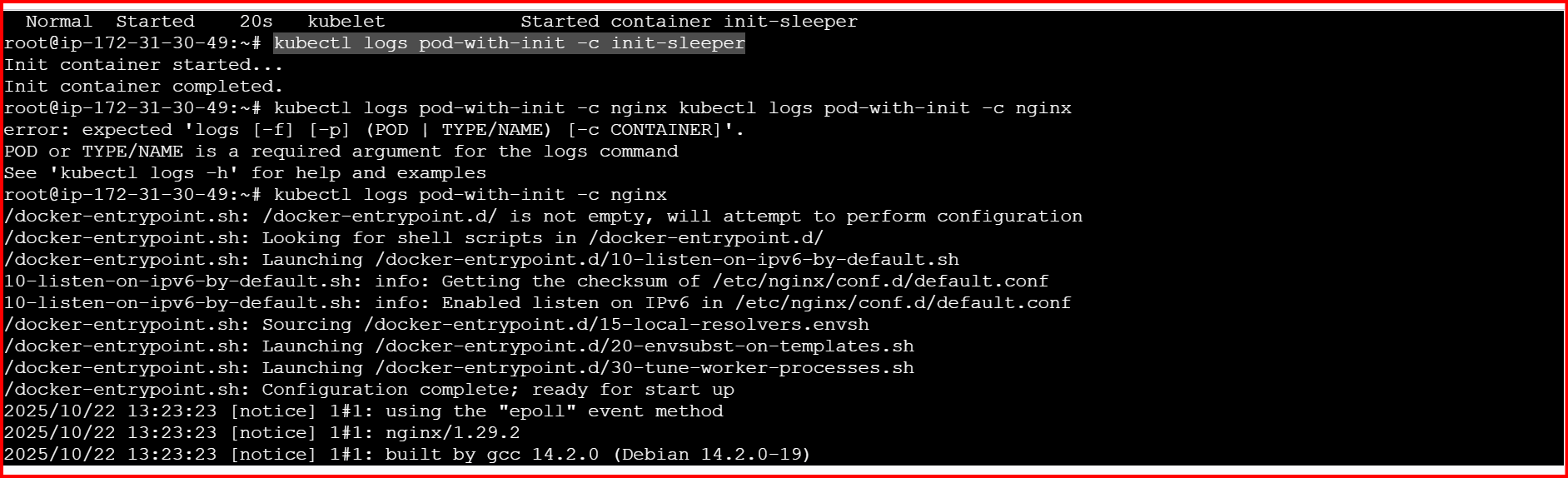
**kubectl get pods**

**kubectl describe pod pod-with-init**

****

**kubectl logs pod-with-init -c init-sleeper**

**kubectl logs pod-with-init -c nginx**

****

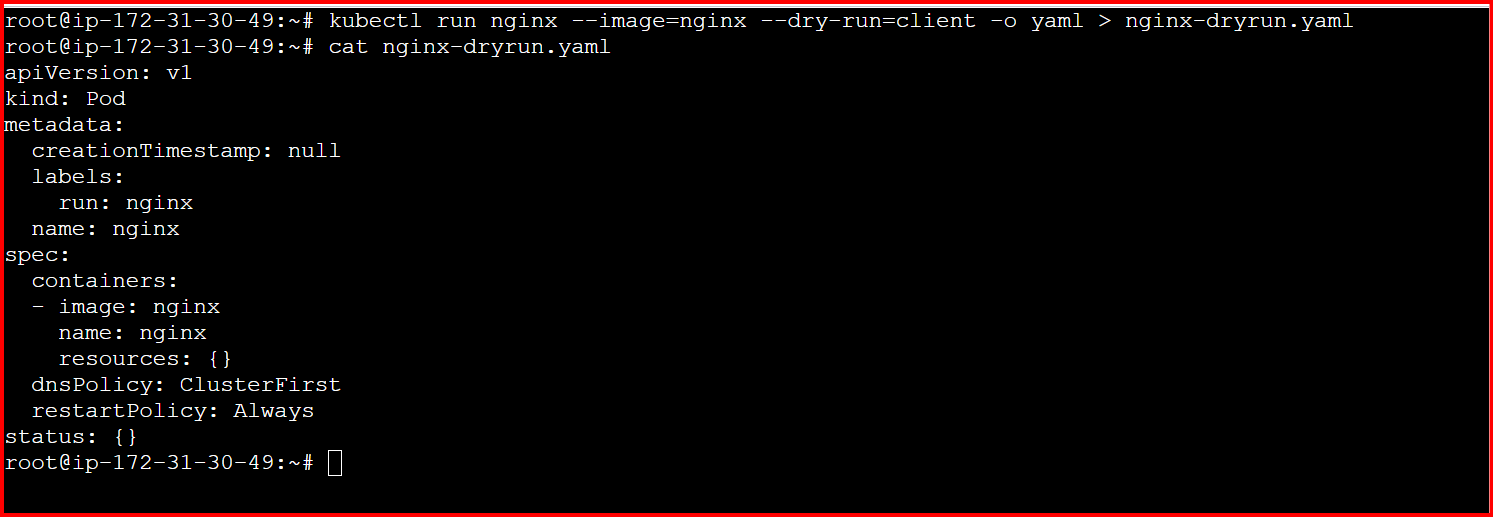
**6.Run a Dry Run Command to Generate YAML Task: Use the kubectl run nginx --image=nginx --dry-run=client -o yaml command to generate a Pod YAML definition. Modify the generated YAML to suit specific requirements (e.g., labels or environment variables) and deploy it.**

**Generate a Pod YAML using Dry Run:**

**kubectl run nginx --image=nginx --dry-run=client -o yaml > nginx-dryrun.yaml**

This creates a file nginx-dryrun.yaml **without actually deploying** the Pod.

If you open the file, it looks like this

****

**Now we can add Custom labels, Environment variables & Port definition**

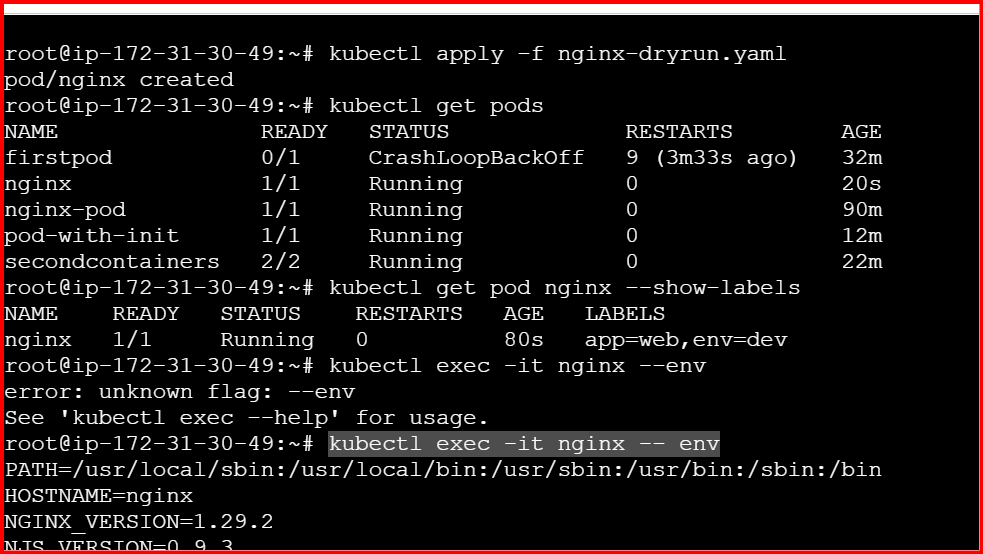


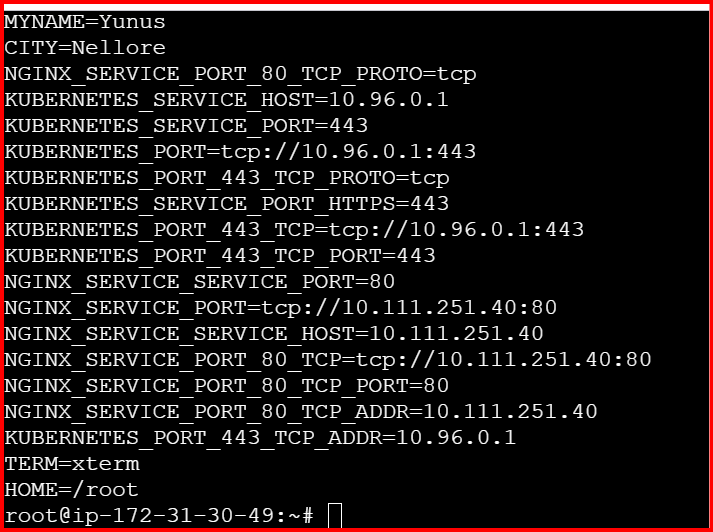
**kubectl apply -f nginx-dryrun.yaml**

**kubectl get pods**

**kubectl get pod nginx --show-labels**

**kubectl exec -it nginx – env**





**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.**

**Create a yaml file**

Vi apply-vs-create.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

labels:

app: web

spec:

containers:

- name: nginx

image: nginx:1.25

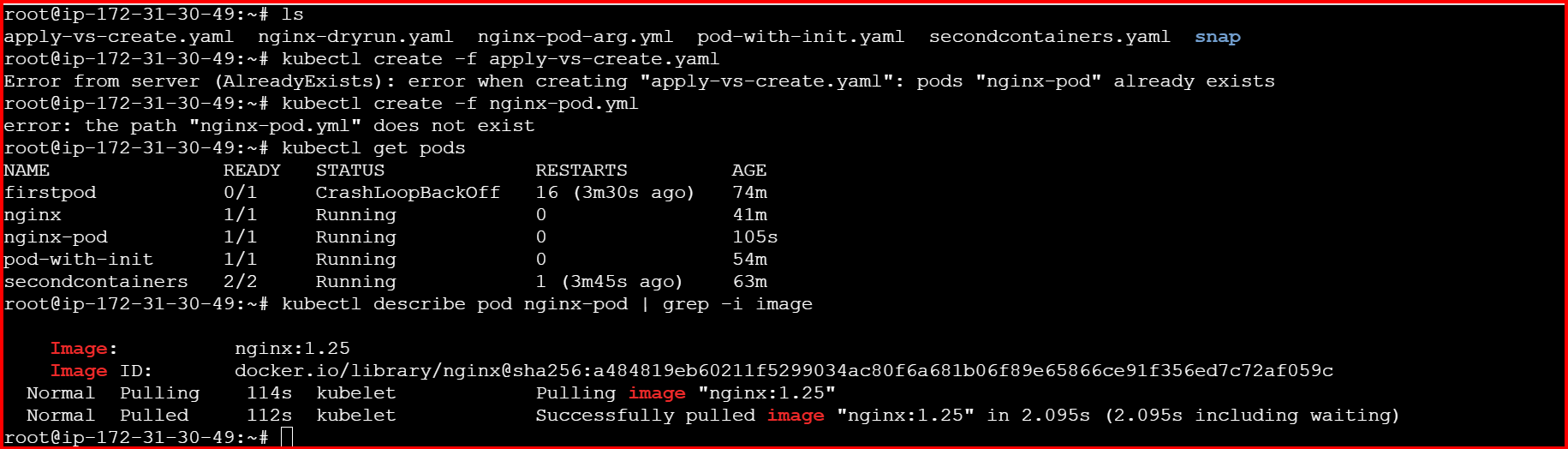
ports:

- containerPort: 80

**kubectl create -f apply-vs-create.yaml**

**kubectl get pods**

**kubectl describe pod nginx-pod | grep -i image**

****

**Modify the YAML:**

**change the image version** from **nginx:1.25 → nginx:1.27**

Apply the Changes Using **kubectl apply**

kubectl apply -f apply-vs-create.yaml

**Redeploy using kubectl apply**

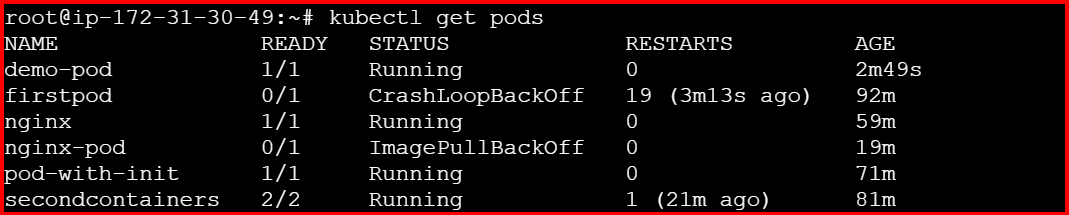


**8.Edit an Existing Pod Configuration Task: Use kubectl edit pod <pod\_name> to modify the running Pod's environment variables or image. After making the changes, verify if they took effect by checking the container logs or environment variables using kubectl exec.**

Create a Deployment:

kubectl create deployment nginx-deploy --image=nginx:1.21

kubectl get pods



Edit the Deployment:

kubectl edit deployment nginx-deploy

Find the container spec, and add an environment variable under env:

spec:

containers:

- name: demo-container

image: nginx:alpine

env:

- name: MY\_ENV

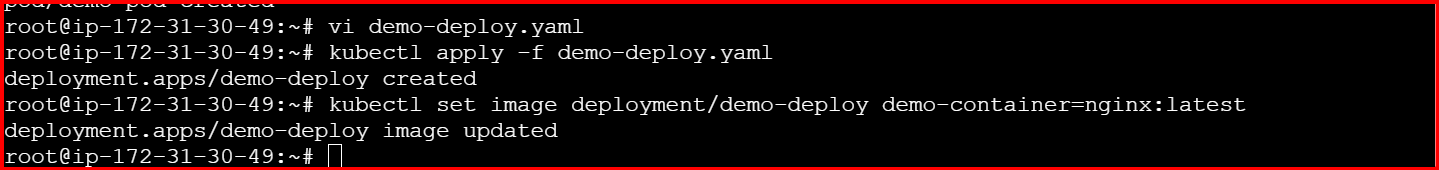
value: "HelloWorld"

command: ["/bin/sh", "-c"]

args: ["echo $MY\_ENV && sleep 3600"]

ports:

- containerPort: 80



**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.**

Create a Pod (example: first.yaml)

apiVersion: v1

kind: Pod

metadata:

name: firstpod

labels:

app: firstpod

spec:

containers:

- name: nginx

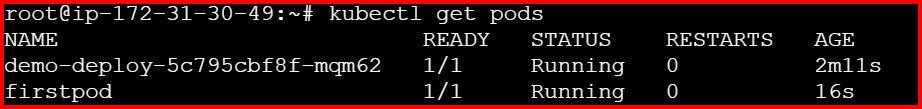
image: nginx:1.21

ports:

- containerPort: 80

Create the pod:

kubectl apply -f first.yaml



**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.**

**Vi resource-pod.yaml**

apiVersion: v1

kind: Pod

metadata:

name: resource-pod

spec:

containers:

- name: nginx-container

image: nginx:1.21

resources:

requests:

memory: "128Mi"

cpu: "250m"

limits:

memory: "256Mi"

cpu: "500m"

ports:

- containerPort: 80

- name: busybox-container

image: busybox

command: ["sh", "-c", "sleep 3600"]

resources:

requests:

memory: "64Mi"

cpu: "100m"

limits:

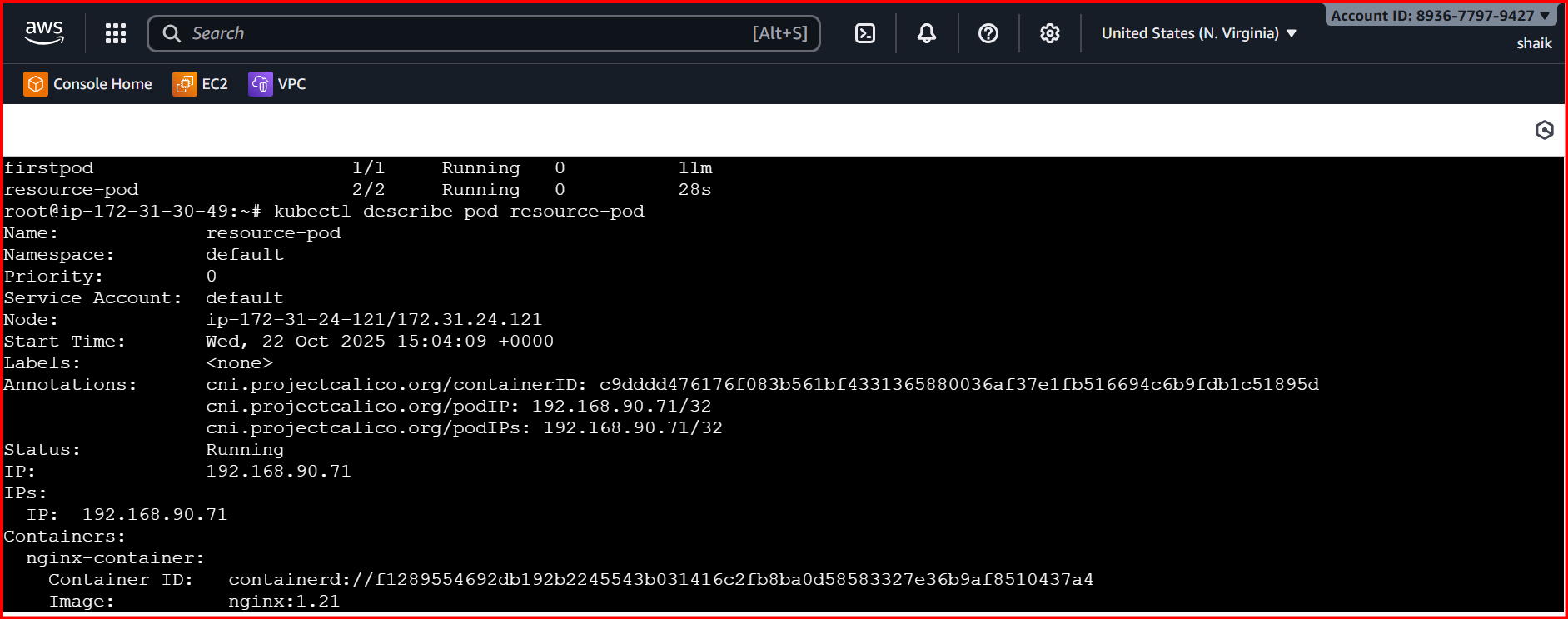
memory: "128Mi"

cpu: "200m"

**kubectl apply -f resource-pod.yaml**

****

**kubectl describe pod resource-pod**

****