**Kubernetes Tasks-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.

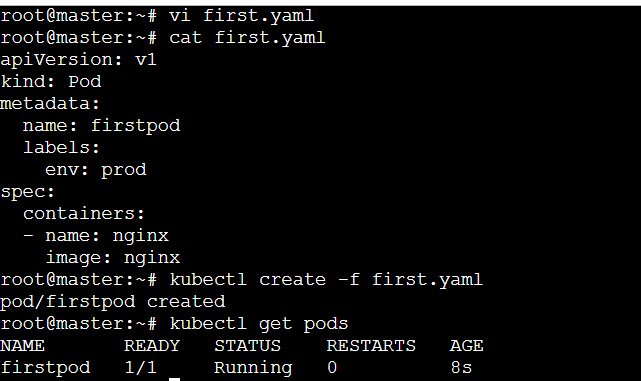
* **Create a yaml file:**

**>>>** vi first.yaml

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:    name: firstpod    labels:      env: prod  spec:    containers:    - name: nginx      image: nginx |

* **Create a pod from the first.yaml file**

**>>>**kubectl create -f first.yaml



* **Check the logs of the pod we created:**

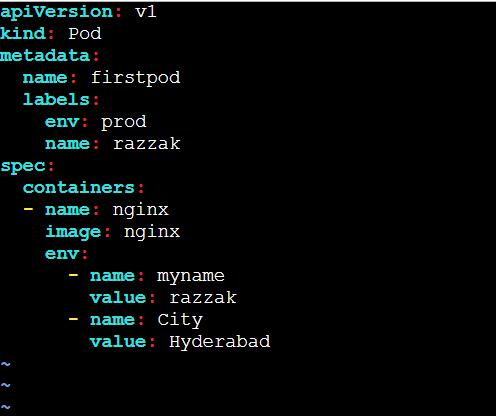
**>>>** kubectl logs firstpod



**2) Set Environment Variables in a Pod**

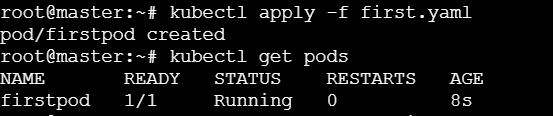
**Task**: Modify the YAML file to include environment variables myname: sabair and City: Hyderabad. Deploy the Pod and use kubectl exec <pod\_name> -- env to check if the environment variables are set properly.

**>>>** Vi first.yaml



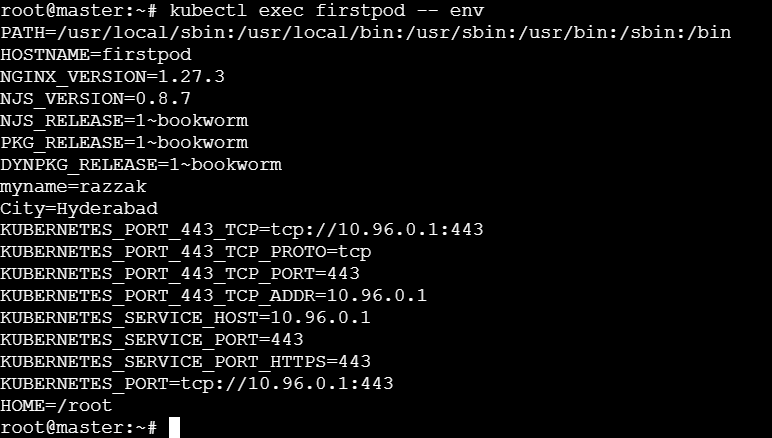
* **Create a pod from the yaml file using declarative approach:**

>>> kubectl apply -f first.yaml



* check if the environment variables are set properly using the command:

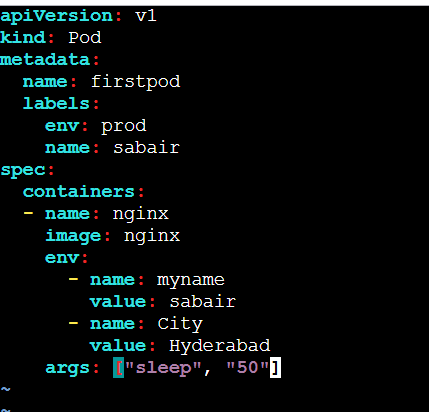
>>> kubectl exec firstpod -- env



**3) Deploy a Pod with Commands (Args) in YAML**

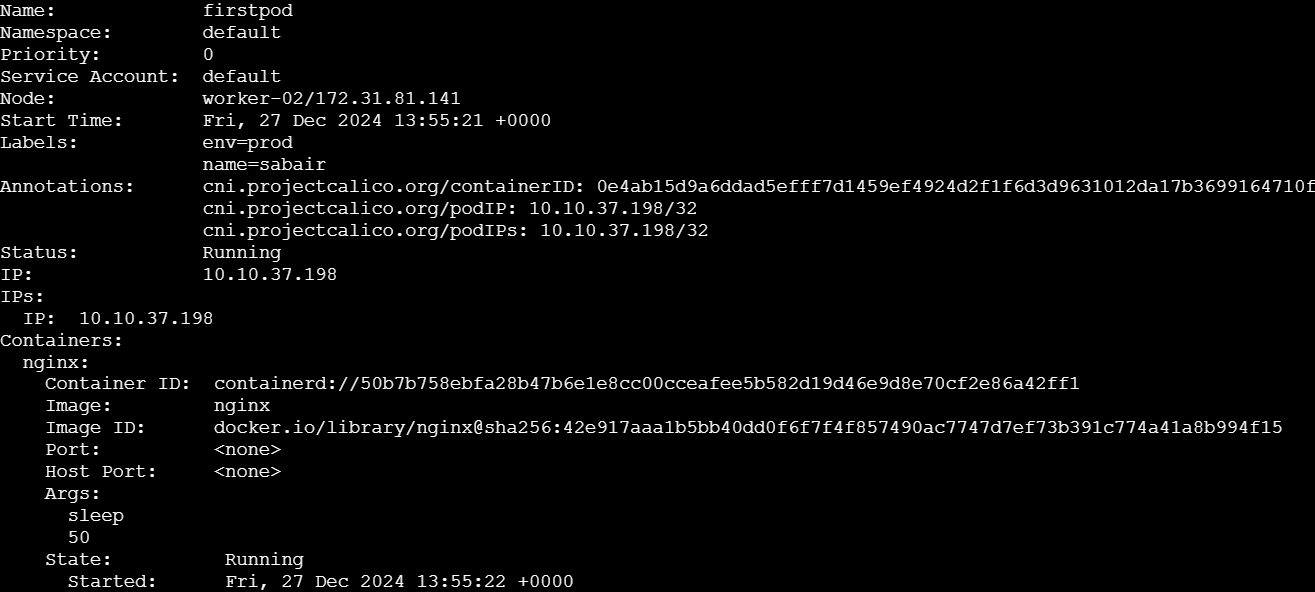
**Task:** Modify the YAML file to add args that instruct the container to sleep for 50 seconds. Deploy the Pod and use kubectl describe pod to verify the args are correctly passed to the container.

>>> vi first.yaml



* **verify the args are correctly passed to the container.**

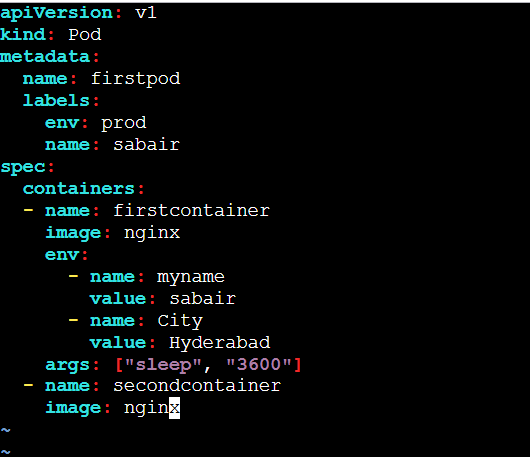
**>>>** kubectl describe pods firstpod



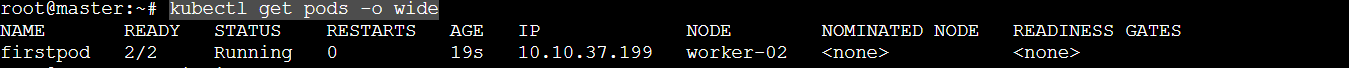
**4) Create a Pod with Two Containers**

**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).

>> vi first.yaml



>>> kubectl get pods -o wide

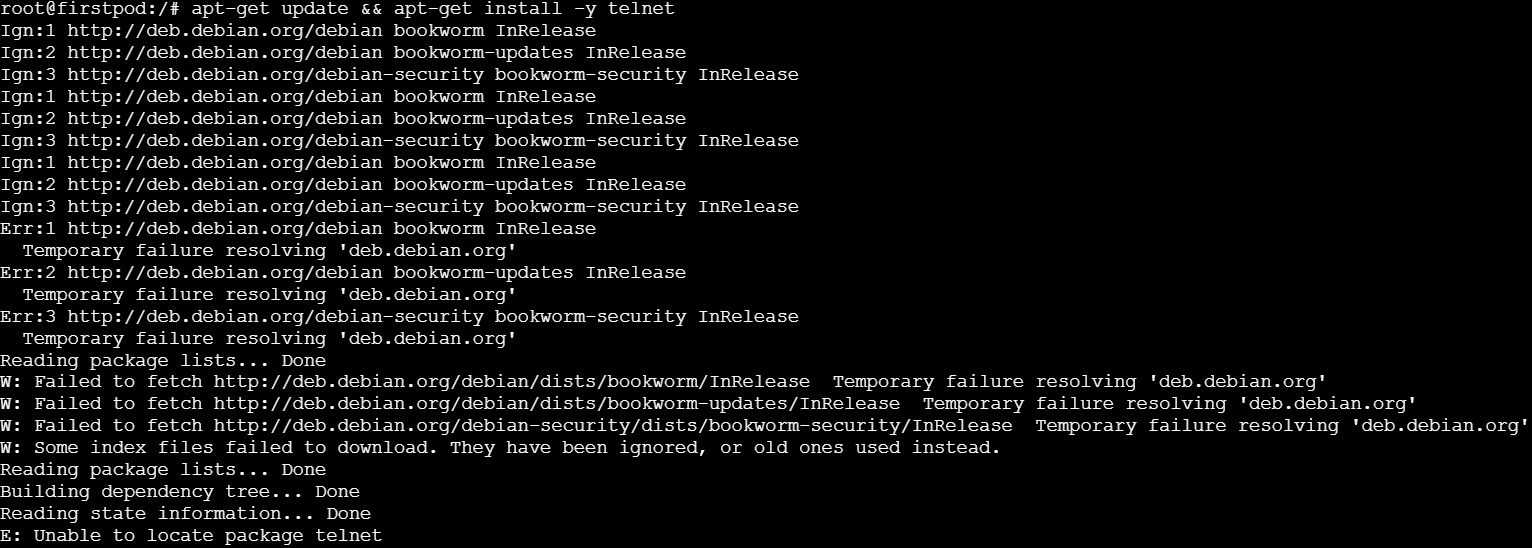


* **login to both the containers using the command.**

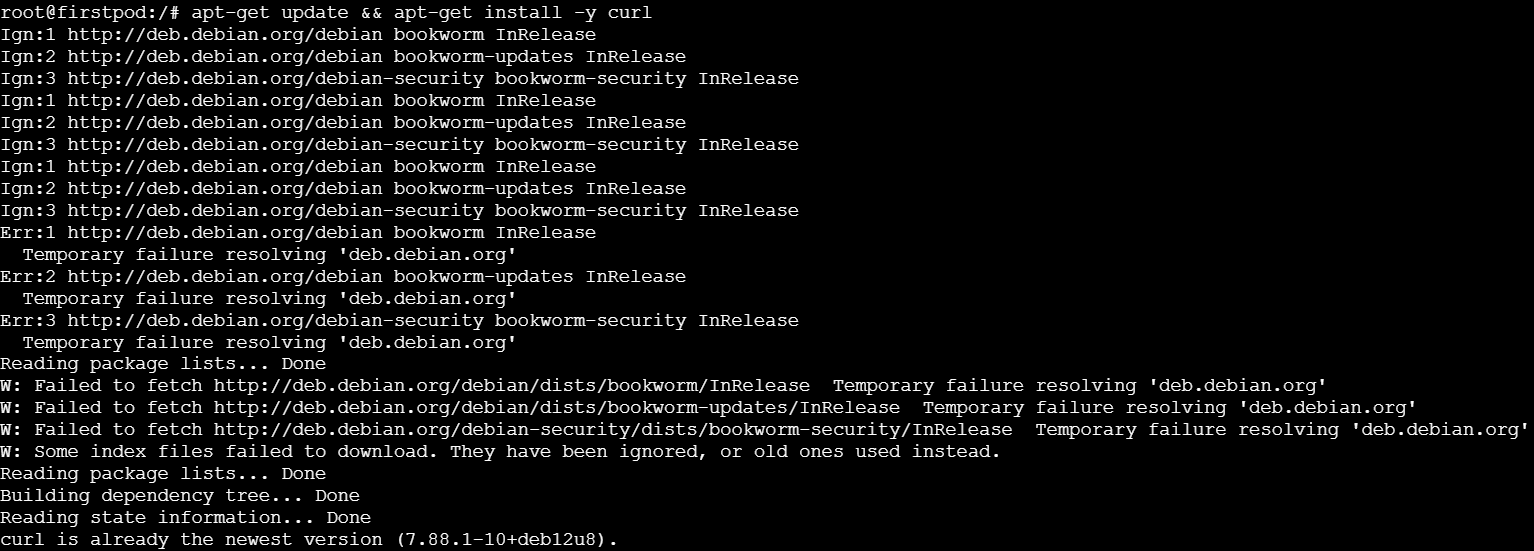
**>>>** kubectl exec firstpod -it -c firstcontainer bash

* **Install Telnet**

**>>>** apt-get update && apt-get install -y telnet

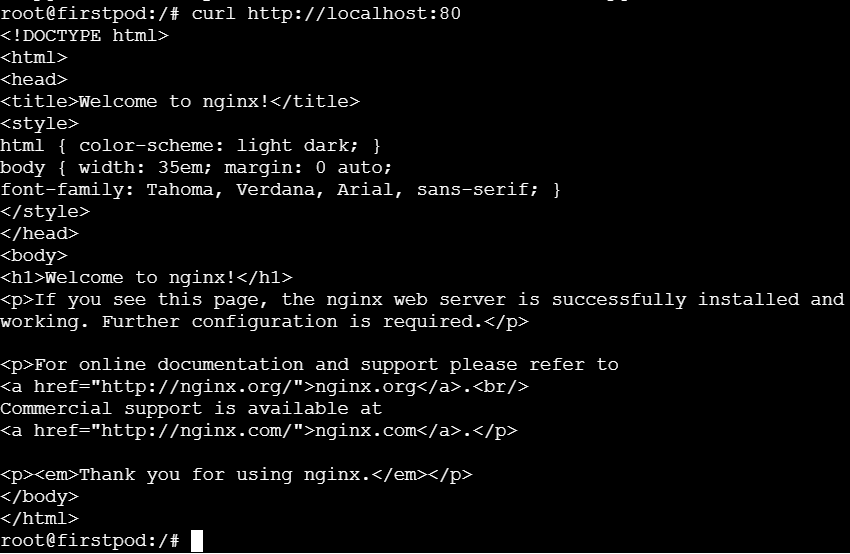
****

* **As telnet installation was throwing error. Installing curl instead.**

****

* **Checking both containers can communicate through the same network.**

**>>>** curl <http://localhost:80>



**5) Set Up an Init Container in a Pod**

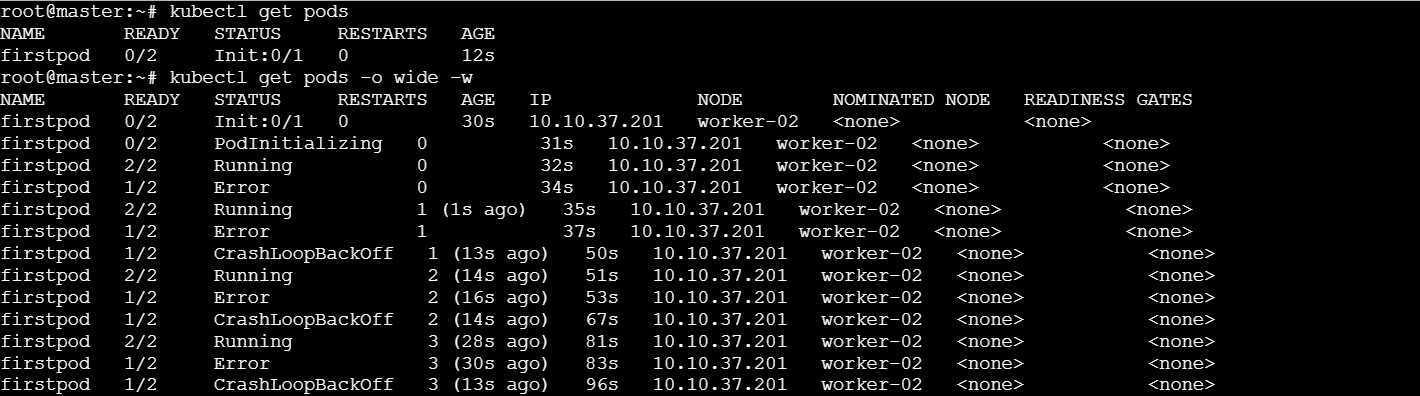
**Task**: Modify the YAML to include an init container that sleeps for 30 seconds before the main containers start. Verify the init container's execution using kubectl describe pod and check the logs to confirm its completion.

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: firstpod  labels:  env: prod  app: sabair  spec:  containers:  - name: firstcontainer  image: nginx  env:  - name: myname  value: sabair  - name: City  value: Hyderabad  - name: secondcontainer  image: nginx  initContainers:  - name: initcontainer  image: nginx  env:  - name: myname  value: sabair  - name: City  value: Hyderabad  args: ["sleep", "30"] |

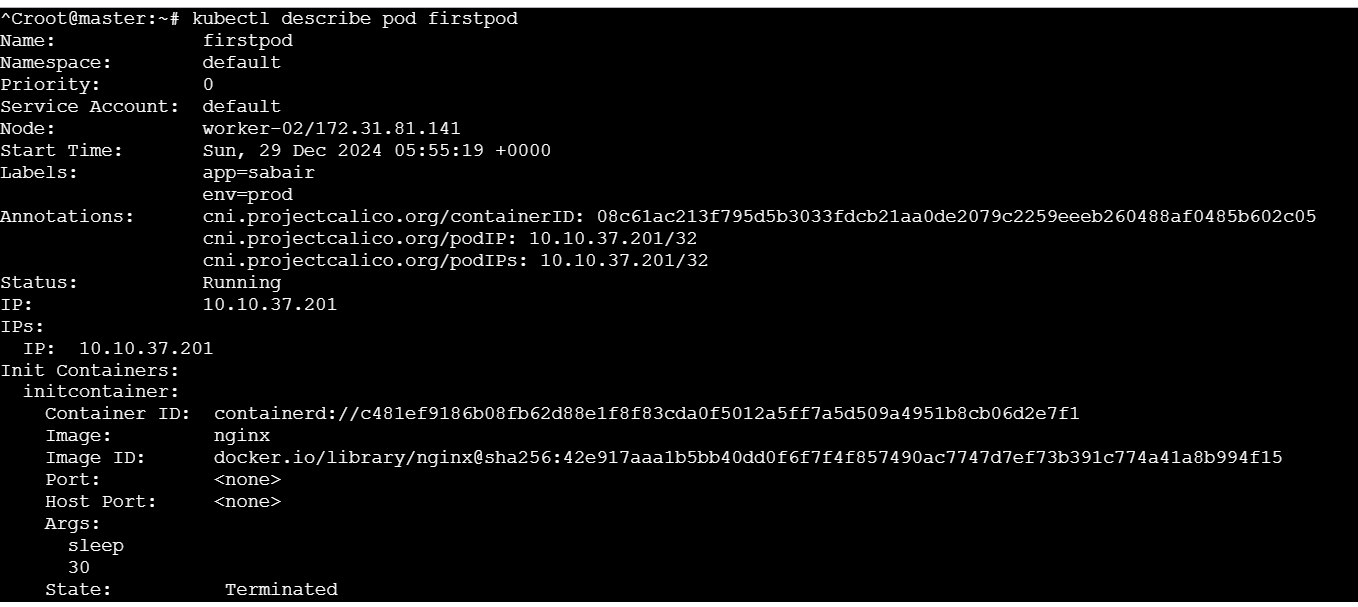
**>>>** cat first.yaml

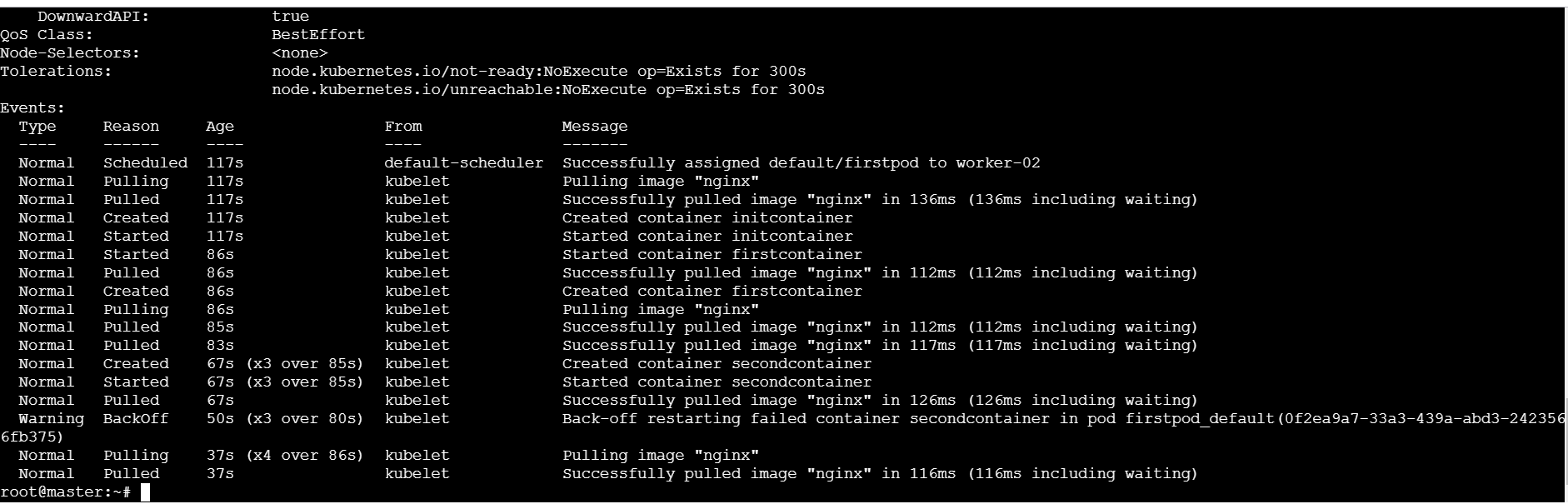


**>>>** Kubectl get pods -o wide -w

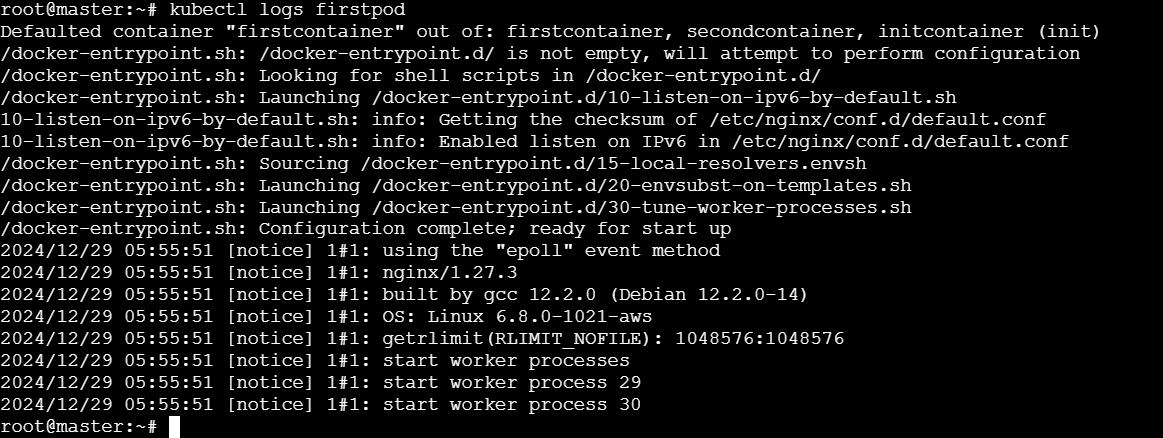


**>>>** kubectl describe pod firstpod



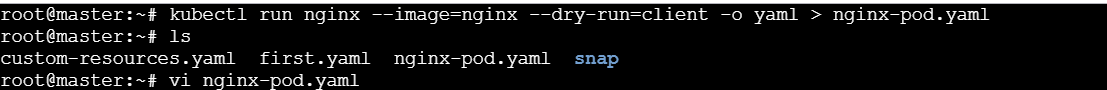


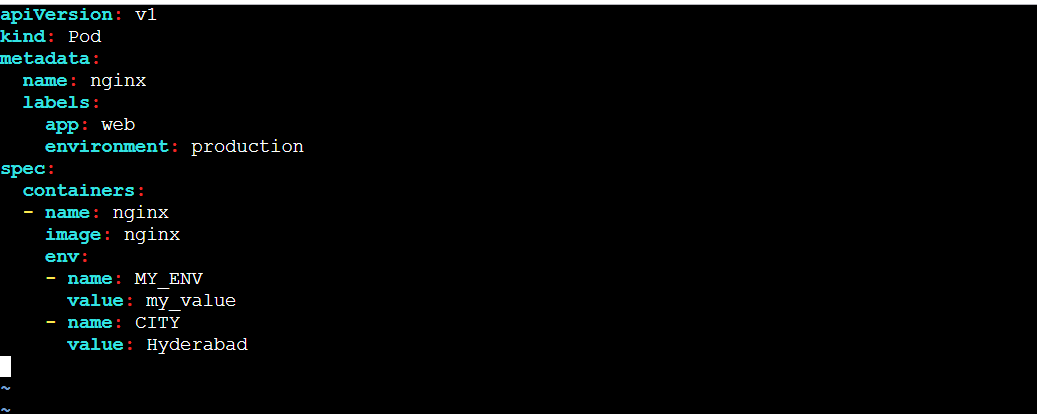
**>>>** kubectl logs firstpod

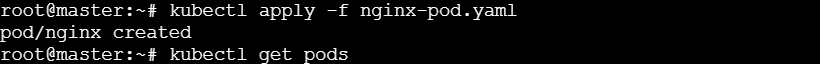


**6) Run a Dry Run Command to Generate YAML**

**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.

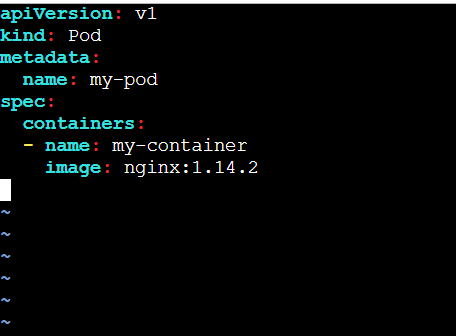


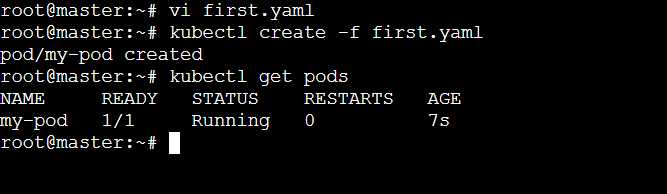


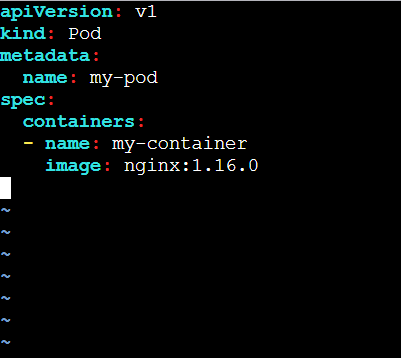


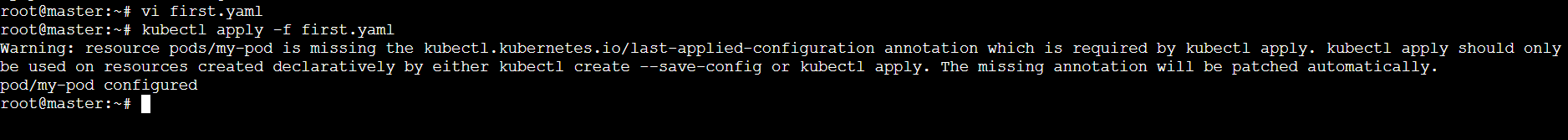
7) **Use kubectl apply vs kubectl create**

**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.





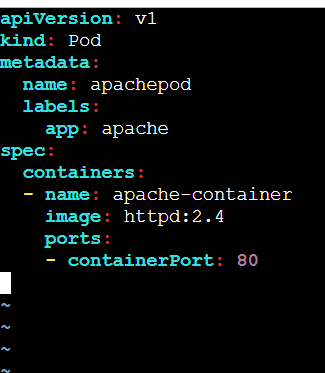




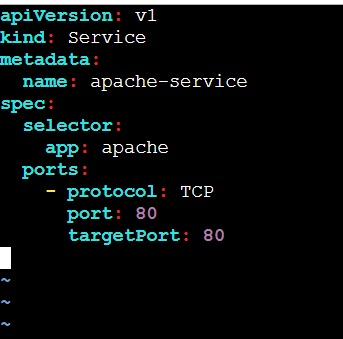
**9) Expose a Pod Using a Service**

**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.

**>>>** vi apache-pod.yaml



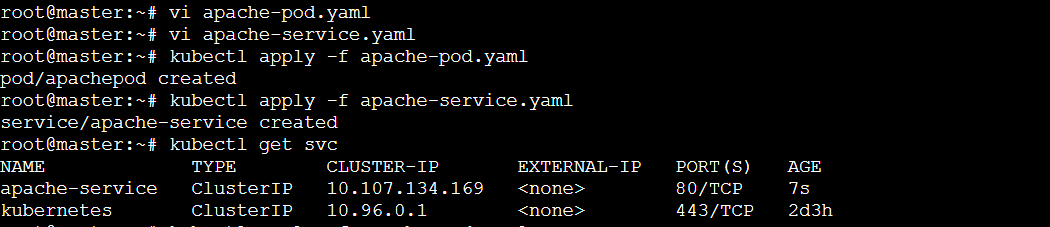
**>>>** vi apache-service.yaml



**>>>** kubectl apply -f apache-pod.yaml

**>>>** kubectl apply -f apache-service.yaml

**>>>** kubectl get svc



**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.

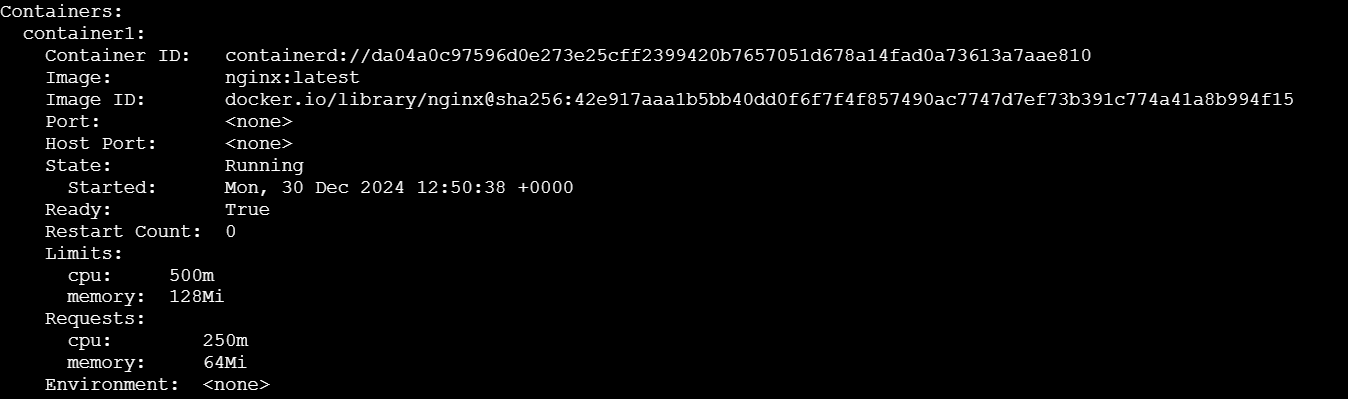
**>>> vi resource-pod.yaml**

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: resource-pod  labels:  app: resource-app  spec:  containers:  - name: container1  image: nginx:latest  resources:  requests:  memory: "64Mi"  cpu: "250m"  limits:  memory: "128Mi"  cpu: "500m"  - name: container2  image: busybox  command: ["sleep", "3600"]  resources:  requests:  memory: "32Mi"  cpu: "100m"  limits:  memory: "64Mi"  cpu: "200m" |

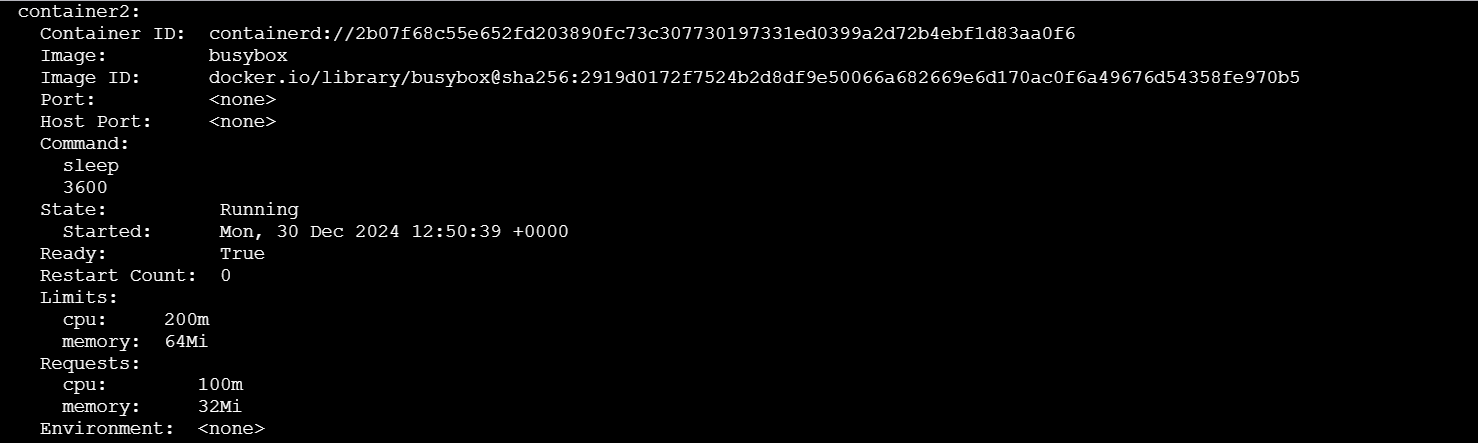
**>>> kubectl apply -f resource-pod.yaml**

**>>> kubectl describe pod resource-pod**

**Container 1:**

****

**Container 2:**

****