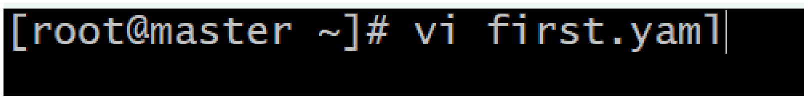
1. Create a Simple Pod Using YAML Task: Write a YAML file to create a Pod named firstpod with an nginx container. Verify the Pod creation using kubectl get pods and check the logs of the container using kubectl logs firstpod.

Login to your master server

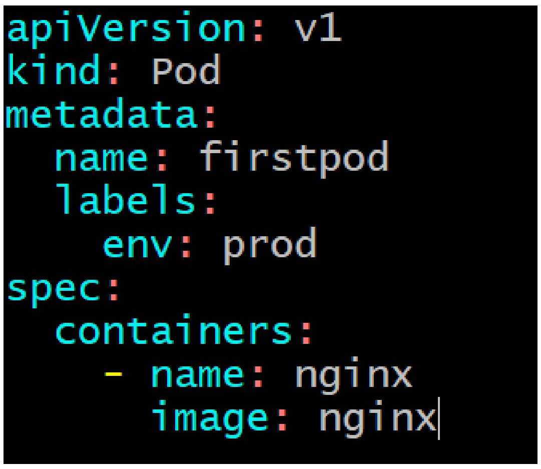
- **vi first.yaml**



```
[root@master ~]# vi first.yaml
```

Add this script in your yaml file

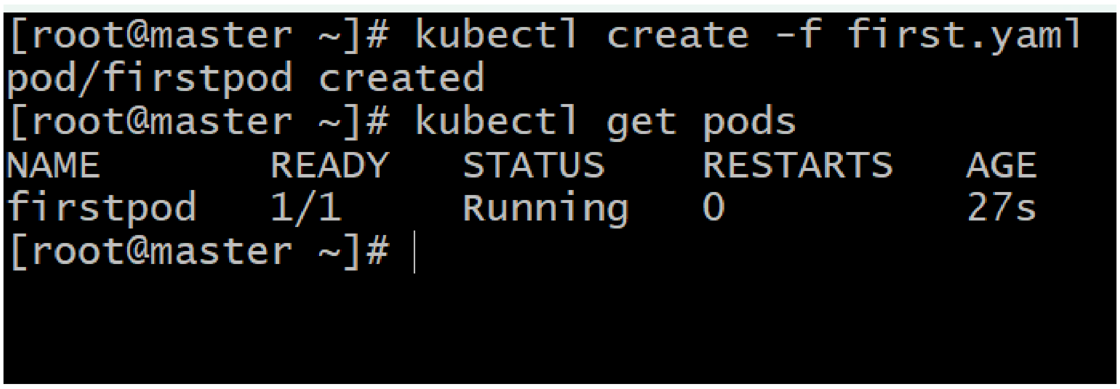
```
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
  labels:
    env: prod
spec:
  containers:
    - name: nginx
      image: nginx
```



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

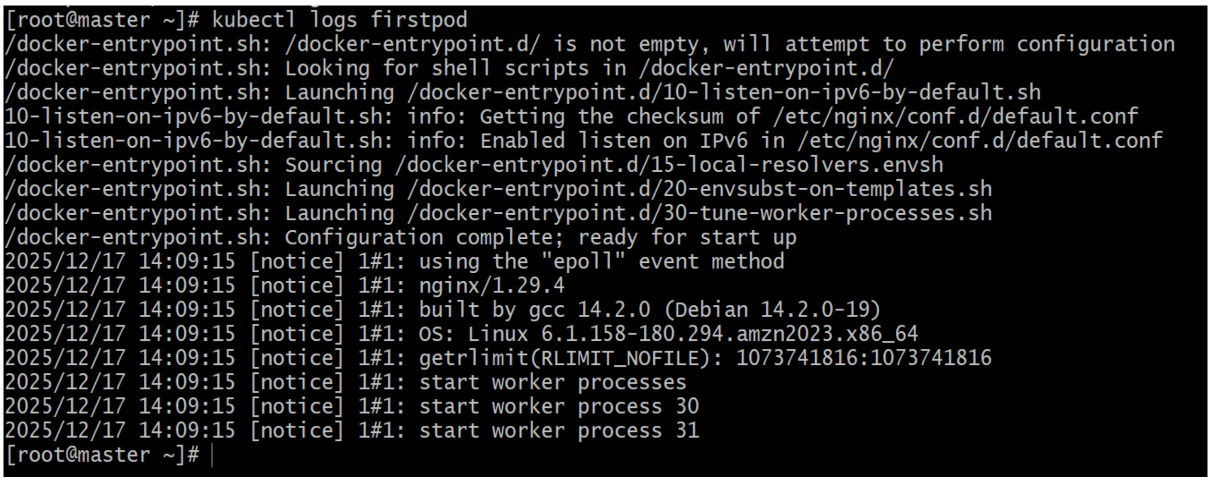
To run the yaml file execte this command.

- **kubectl create -f first.yaml**
- **kubectl get pods**



```
[root@master ~]# kubectl create -f first.yaml
pod/firstpod created
[root@master ~]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          27s
[root@master ~]#
```

- **kubectl logs firstpod**



```
[root@master ~]# kubectl logs firstpod
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/12/17 14:09:15 [notice] 1#1: using the "epoll" event method
2025/12/17 14:09:15 [notice] 1#1: nginx/1.29.4
2025/12/17 14:09:15 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/12/17 14:09:15 [notice] 1#1: OS: Linux 6.1.158-180.294.amzn2023.x86_64
2025/12/17 14:09:15 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1073741816:1073741816
2025/12/17 14:09:15 [notice] 1#1: start worker processes
2025/12/17 14:09:15 [notice] 1#1: start worker process 30
2025/12/17 14:09:15 [notice] 1#1: start worker process 31
[root@master ~]#
```

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.

Delete the pods in your machine.

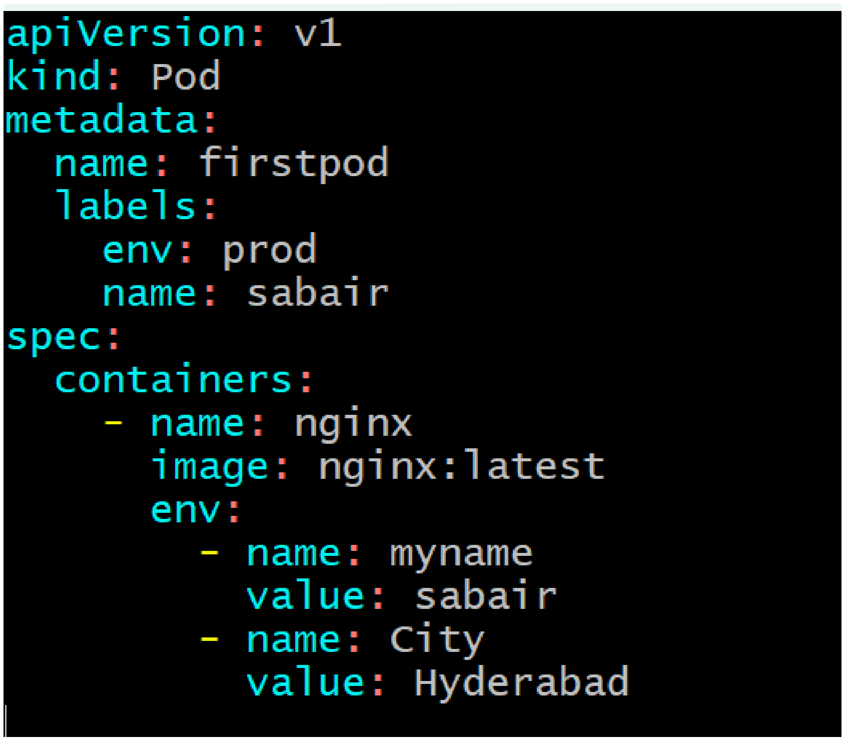
- **kubectl delete pods --all**

Edit the first.yaml file and give this script.

```
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
  labels:
    env: prod
    name: sabair
spec:
  containers:
    - name: nginx
      image: nginx:
      env:
        - name: myname
          value: sabair
```

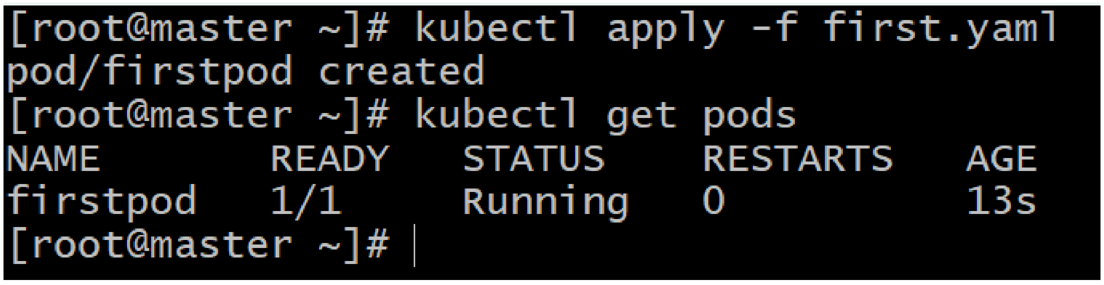
- name: City

value: Hyderabad



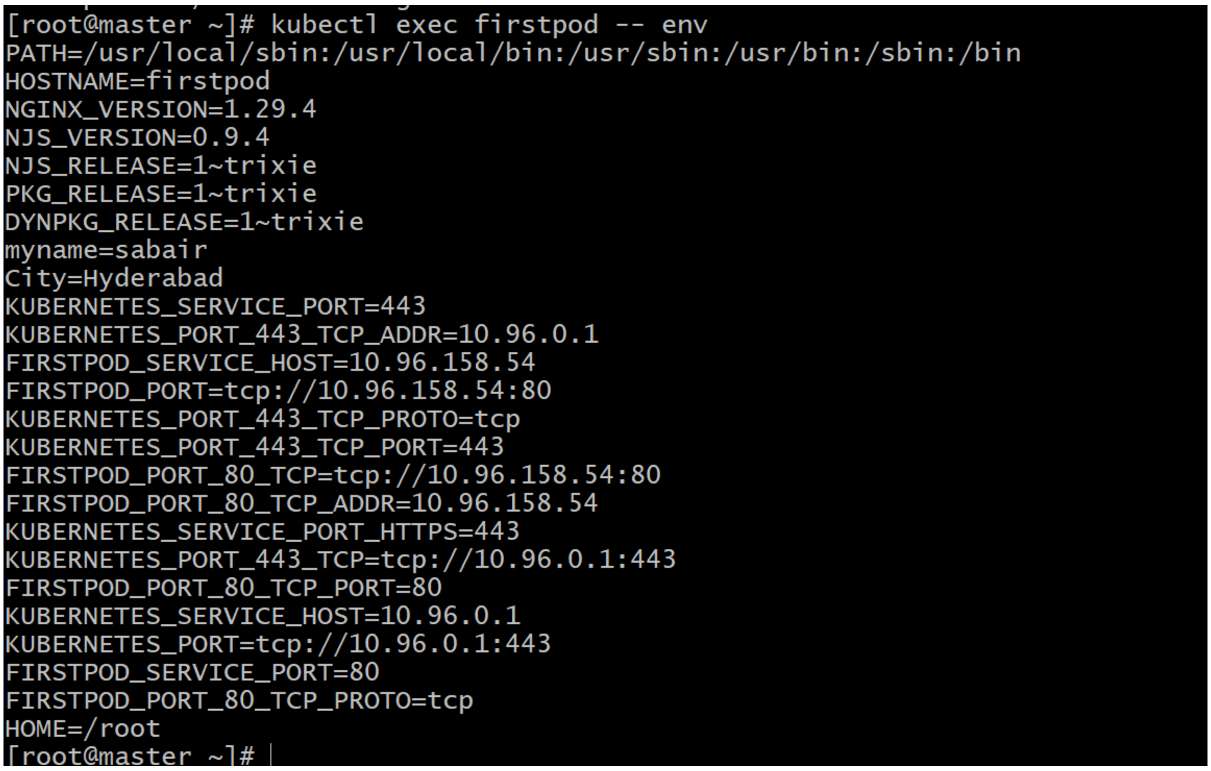
```
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
  labels:
    env: prod
    name: sabair
spec:
  containers:
    - name: nginx
      image: nginx:latest
      env:
        - name: myname
          value: sabair
        - name: City
          value: Hyderabad
```

- **kubectl apply -f first.yaml**



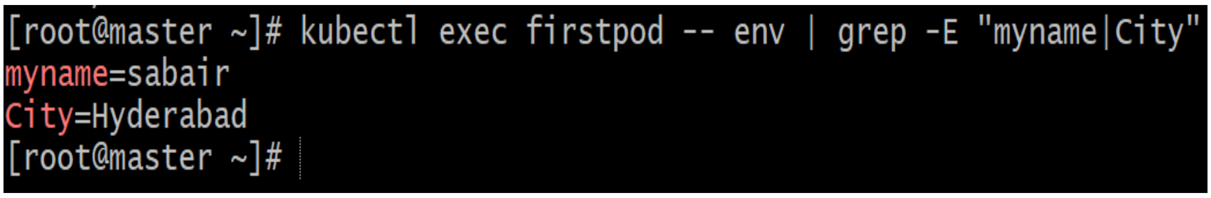
```
[root@master ~]# kubectl apply -f first.yaml
pod/firstpod created
[root@master ~]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          13s
[root@master ~]#
```

- **kubectl exec firstpod -- env**



```
[root@master ~]# 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=sabair
City=Hyderabad
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
FIRSTPOD_SERVICE_HOST=10.96.158.54
FIRSTPOD_PORT=tcp://10.96.158.54:80
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
FIRSTPOD_PORT_80_TCP=tcp://10.96.158.54:80
FIRSTPOD_PORT_80_TCP_ADDR=10.96.158.54
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
FIRSTPOD_PORT_80_TCP_PORT=80
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
FIRSTPOD_SERVICE_PORT=80
FIRSTPOD_PORT_80_TCP_PROTO=tcp
HOME=/root
[root@master ~]#
```

- **kubectl exec firstpod -- env | grep -E "myname|City"**



```
[root@master ~]# kubectl exec firstpod -- env | grep -E "myname|City"
myname=sabair
City=Hyderabad
[root@master ~]#
```

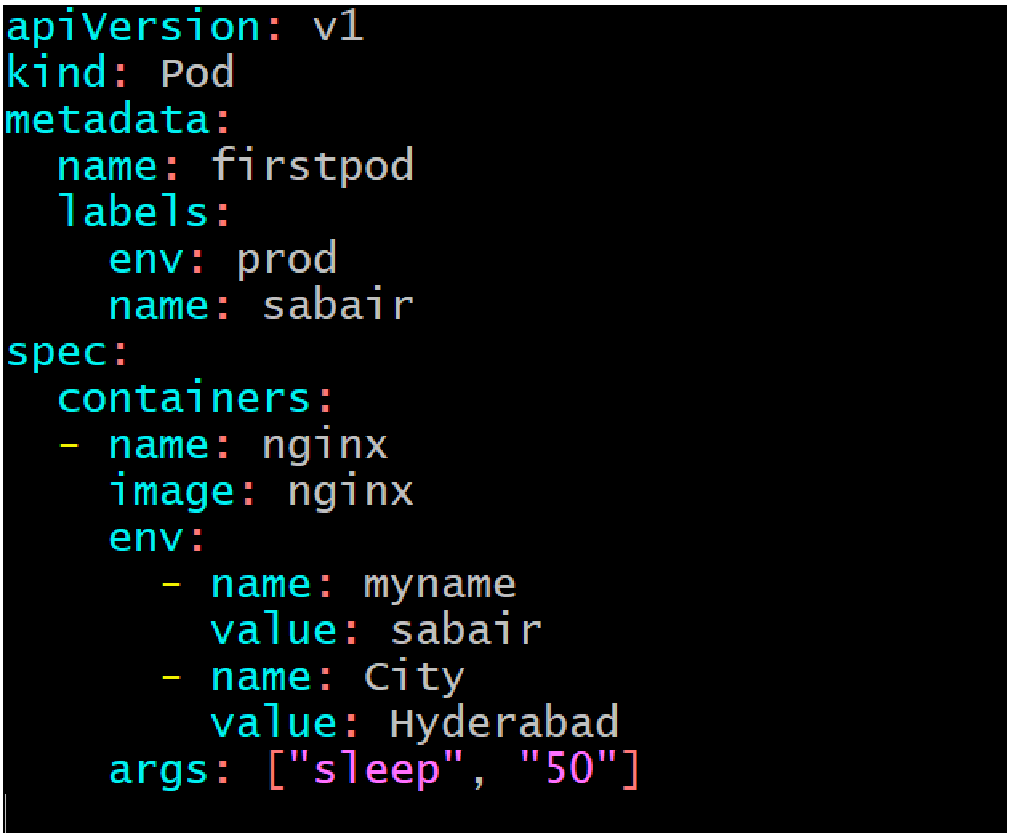
3. Deploy a Pod with Commands (Args) in YAML Task: Modify the YAML file to add args that instruct the container to sleep for 50 seconds. Deploy the Pod and use kubectl describe pod to verify the args are correctly passed to the container.

Give this script in your yaml file.

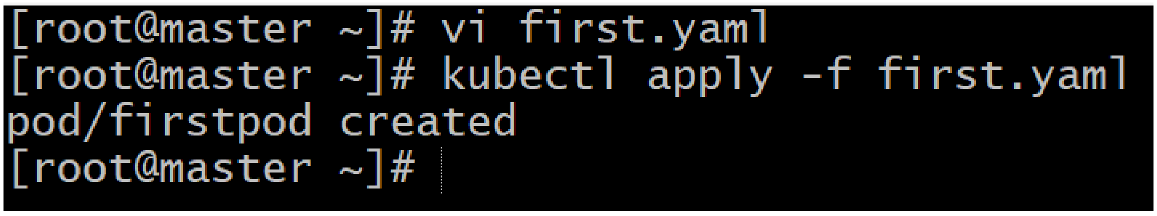
apiVersion: v1

kind: Pod

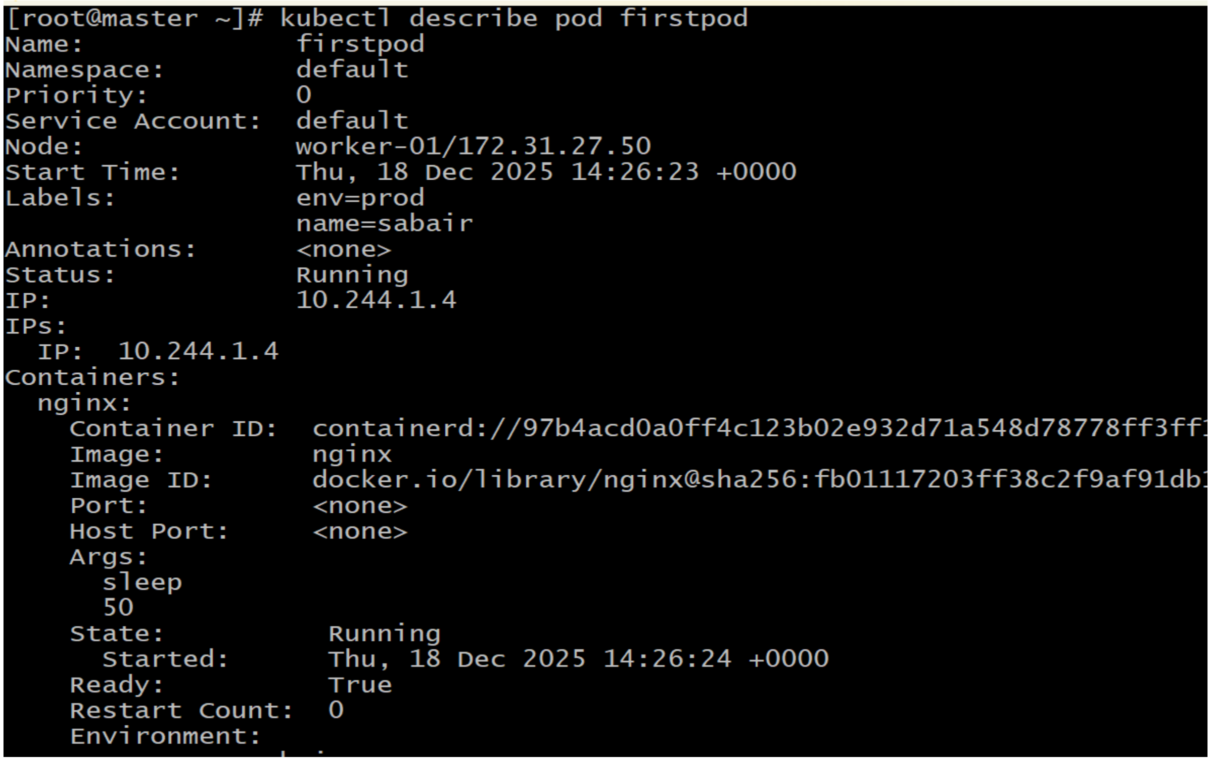
```yaml
metadata:
  name: firstpod
  labels:
    env: prod
    name: sabair
spec:
  containers:
  - name: nginx
    image: nginx
    env:
      - name: myname
        value: sabair
      - name: City
        value: Hyderabad
    args: ["sleep", "50"]
```



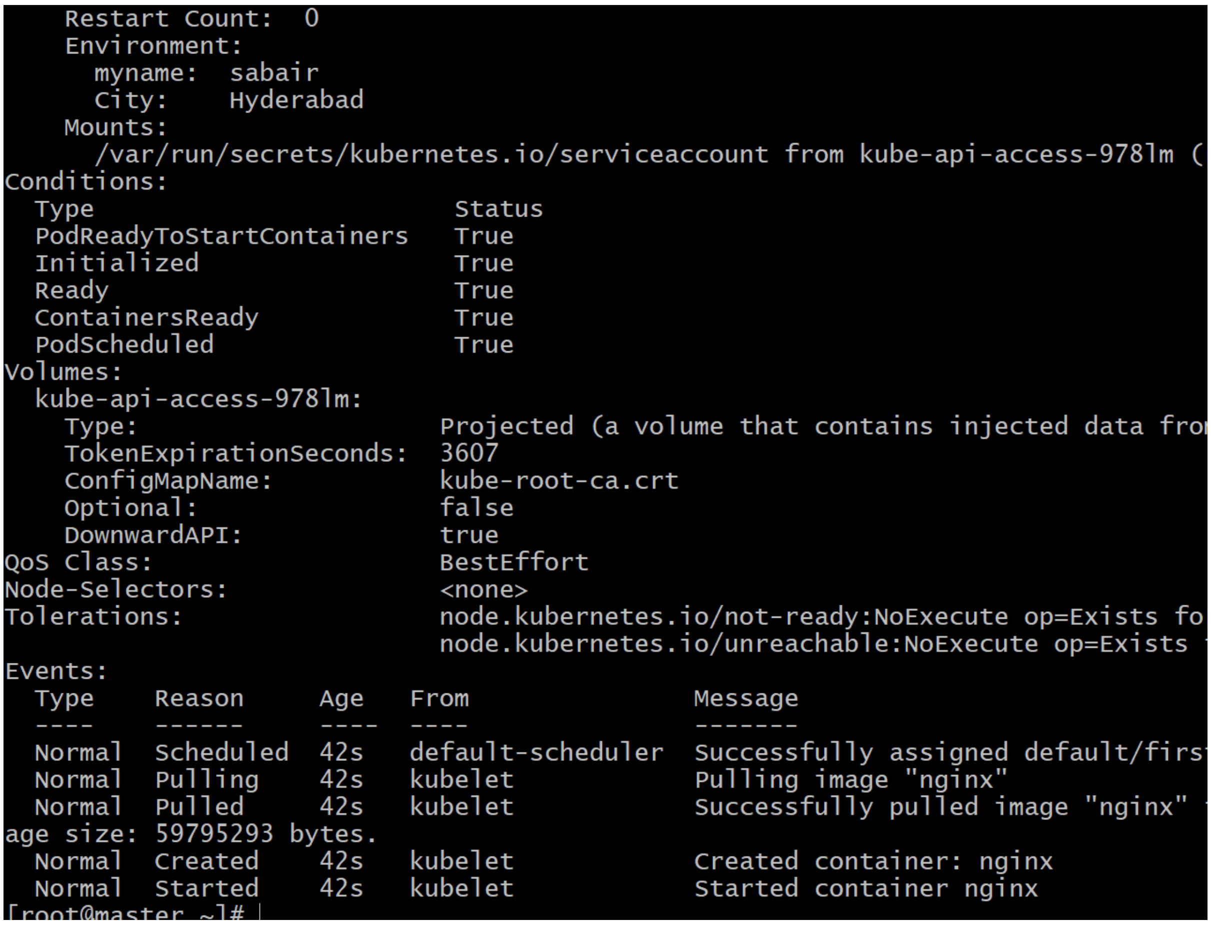
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
  labels:
    env: prod
    name: sabair
spec:
  containers:
  - name: nginx
    image: nginx
    env:
      - name: myname
        value: sabair
      - name: City
        value: Hyderabad
    args: ["sleep", "50"]
```



```
[root@master ~]# vi first.yaml
[root@master ~]# kubectl apply -f first.yaml
pod/firstpod created
[root@master ~]#
```



```
[root@master ~]# kubectl describe pod firstpod
Name:               firstpod
Namespace:          default
Priority:           0
Service Account:    default
Node:               worker-01/172.31.27.50
Start Time:         Thu, 18 Dec 2025 14:26:23 +0000
Labels:             env=prod
                    name=sabair
Annotations:        <none>
Status:             Running
IP:                 10.244.1.4
IPs:
  IP:  10.244.1.4
Containers:
  nginx:
    Container ID:  containerd://97b4acd0a0ff4c123b02e932d71a548d78778ff3ff
    Image:         nginx
    Image ID:      docker.io/library/nginx@sha256:fb01117203ff38c2f9af91db
    Port:          <none>
    Host Port:     <none>
    Args:
      sleep
      50
    State:          Running
      Started:      Thu, 18 Dec 2025 14:26:24 +0000
    Ready:          True
    Restart Count:  0
    Environment:
```



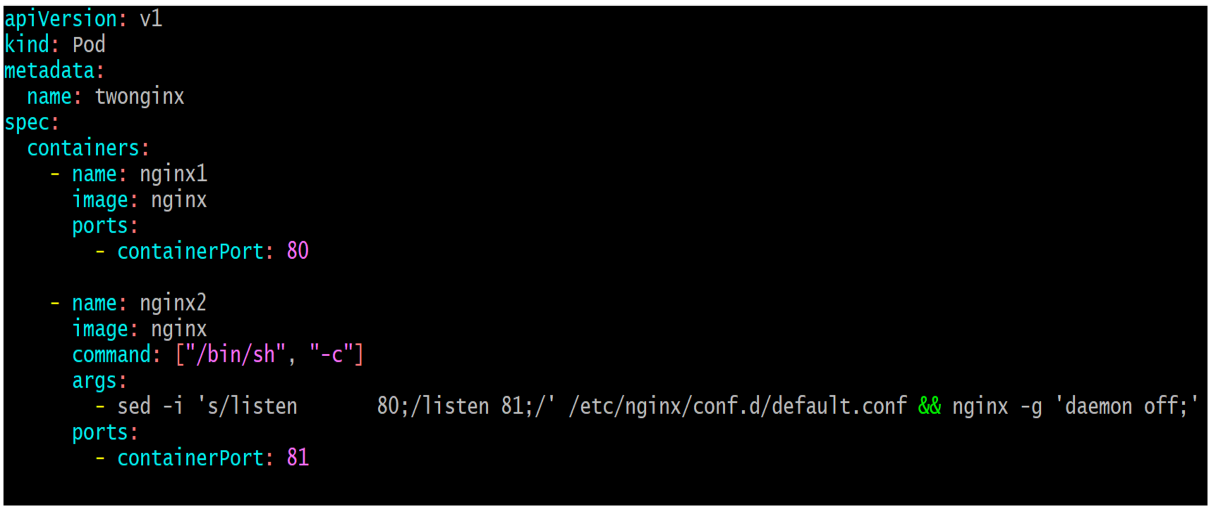
```
    Restart Count:  0
    Environment:
      myname:  sabair
      City:    Hyderabad
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-978lm (
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-978lm:
    Type:                    Projected (a volume that contains injected data fro
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    Optional:                false
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists fo
                             node.kubernetes.io/unreachable:NoExecute op=Exists
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  42s   default-scheduler  Successfully assigned default/firs
  Normal  Pulling    42s   kubelet            Pulling image "nginx"
  Normal  Pulled     42s   kubelet            Successfully pulled image "nginx"
age size: 59795293 bytes.
  Normal  Created    42s   kubelet            Created container: nginx
  Normal  Started    42s   kubelet            Started container nginx
[root@master ~]#
```

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).

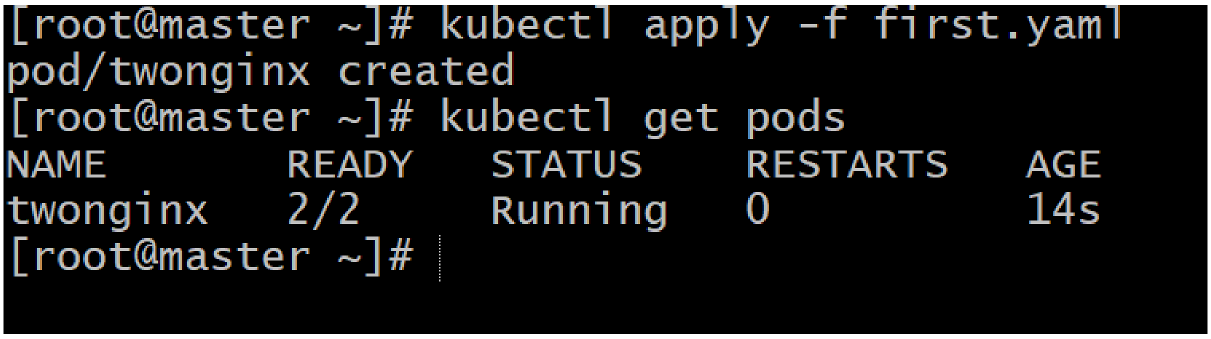
Write this script in your yaml.

```
apiVersion: v1

kind: Pod

metadata:

  name: twonginx
```

```
spec:
  containers:
    - name: nginx1
      image: nginx
      ports:
        - containerPort: 80


    - name: nginx2
      image: nginx
      command: ["/bin/sh", "-c"]
      args:
        - sed -i 's/listen       80;/listen 81;/'
/etc/nginx/conf.d/default.conf && nginx -g 'daemon off;'
      ports:
        - containerPort: 81
```

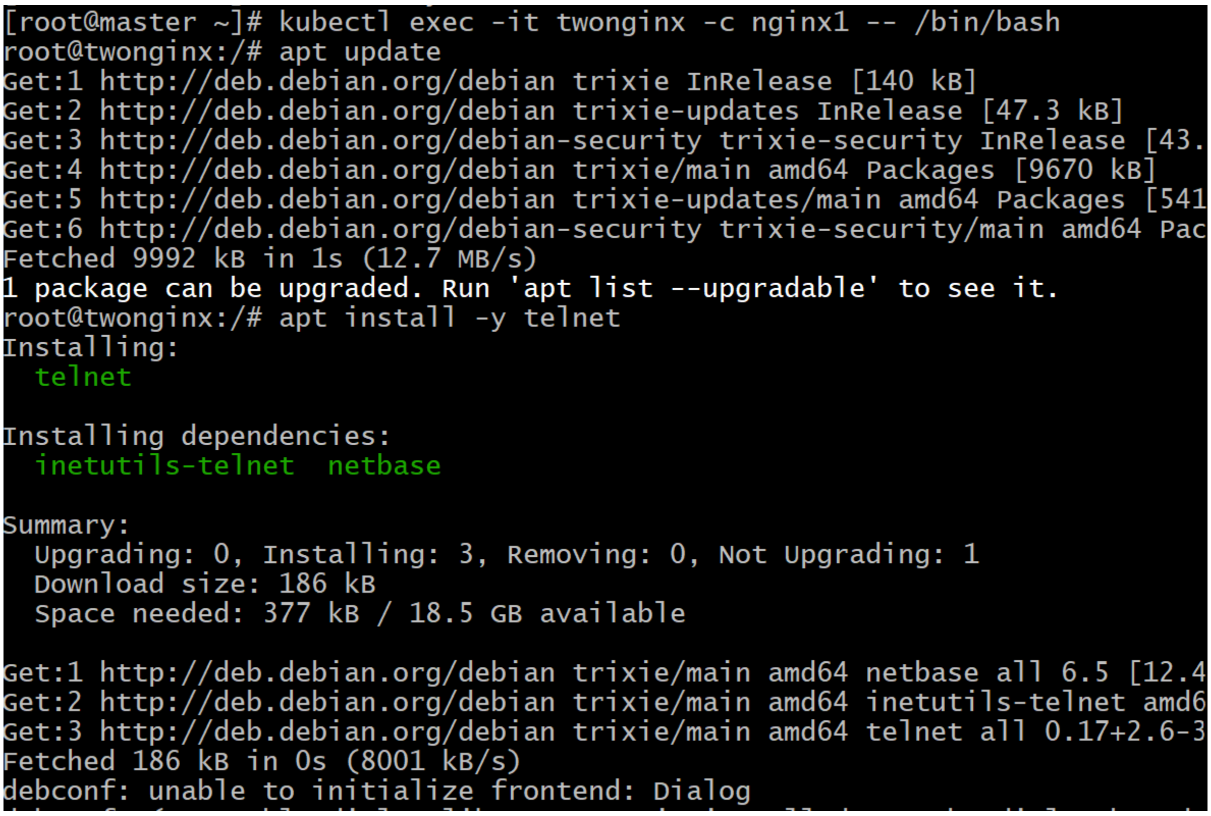


```
apiVersion: v1
kind: Pod
metadata:
  name: twonginx
spec:
  containers:
    - name: nginx1
      image: nginx
      ports:
        - containerPort: 80

    - name: nginx2
      image: nginx
      command: ["/bin/sh", "-c"]
      args:
        - sed -i 's/listen       80;/listen 81;/' /etc/nginx/conf.d/default.conf && nginx -g 'daemon off;'
      ports:
        - containerPort: 81
```

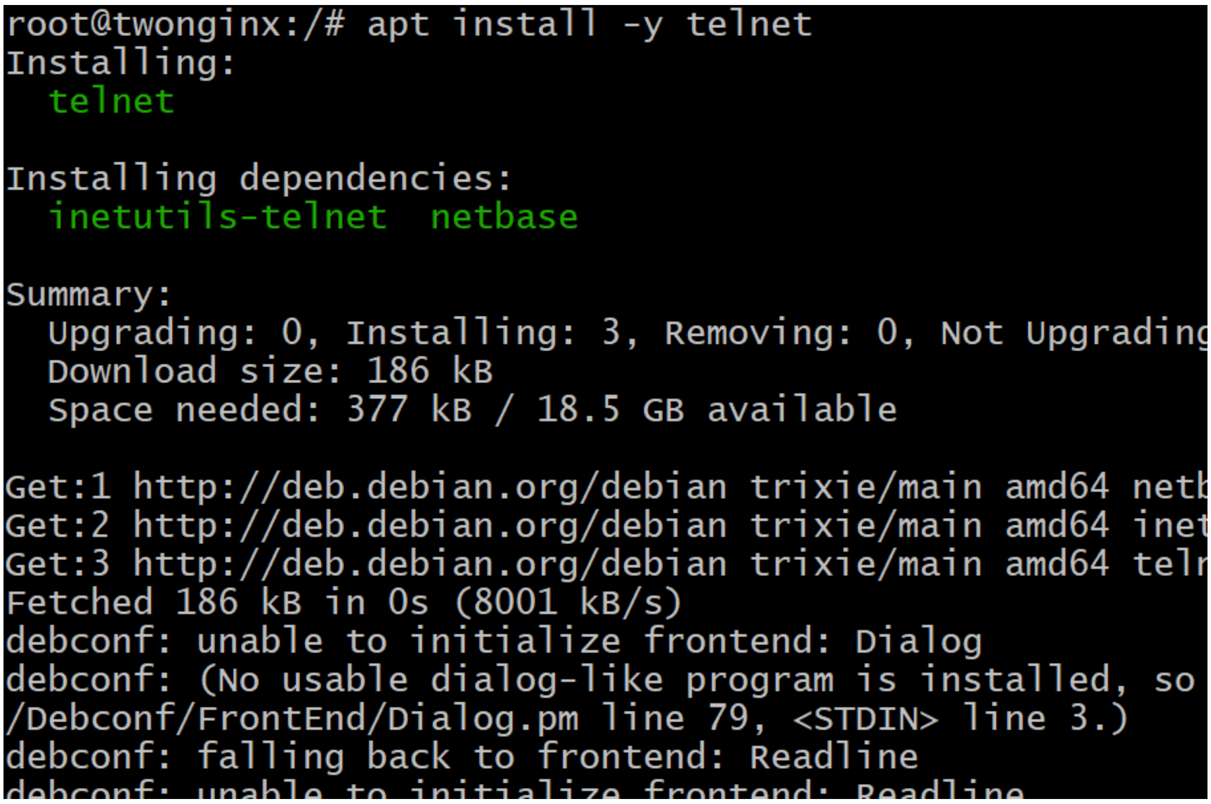


```
[root@master ~]# kubectl apply -f first.yaml
pod/twonginx created
[root@master ~]# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
twonginx    2/2     Running   0          14s
[root@master ~]#
```

- **kubectl exec -it twonginx -c nginx1 -- /bin/bash**
- **apt update**
- **apt install -y telnet**

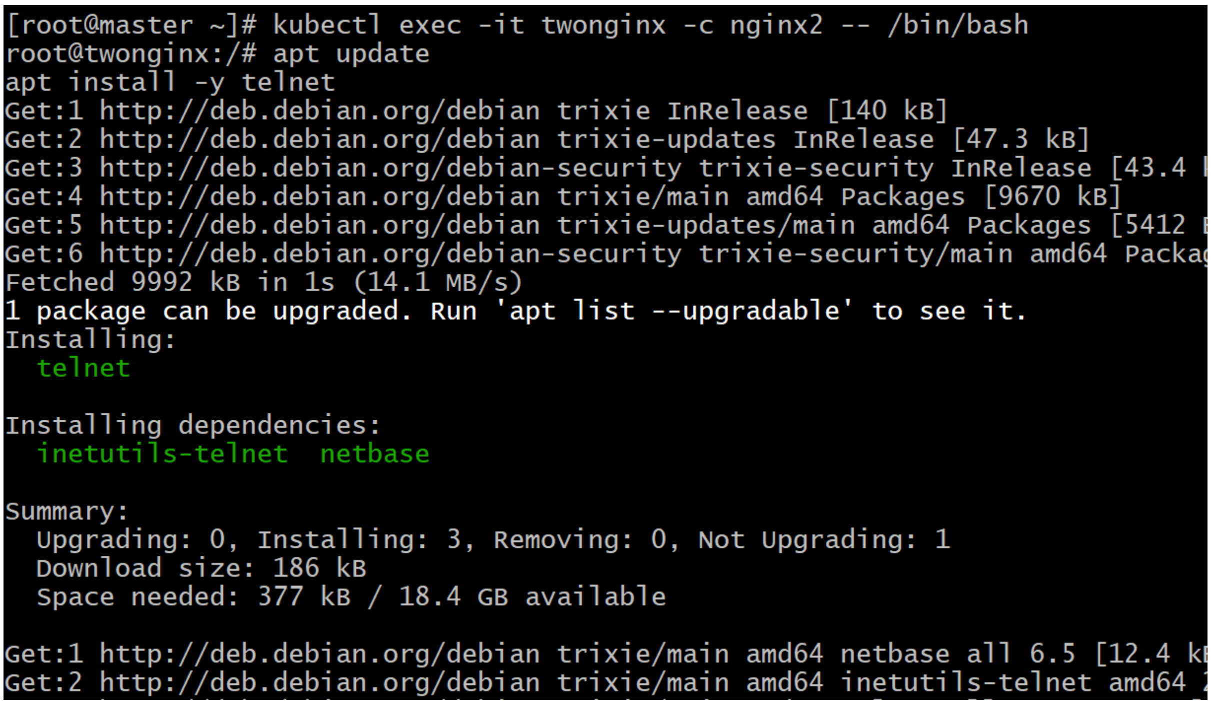


```
[root@master ~]# kubectl exec -it twonginx -c nginx1 -- /bin/bash
root@twonginx:/# apt update
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.
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 [541
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Pac
Fetched 9992 kB in 1s (12.7 MB/s)
1 package can be upgraded. Run 'apt list --upgradable' to see it.
root@twonginx:/# apt install -y telnet
Installing:
  telnet

Installing dependencies:
  inetutils-telnet  netbase

Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 1
  Download size: 186 kB
  Space needed: 377 kB / 18.5 GB available

Get:1 http://deb.debian.org/debian trixie/main amd64 netbase all 6.5 [12.4
Get:2 http://deb.debian.org/debian trixie/main amd64 inetutils-telnet amd6
Get:3 http://deb.debian.org/debian trixie/main amd64 telnet all 0.17+2.6-3
Fetched 186 kB in 0s (8001 kB/s)
debconf: unable to initialize frontend: Dialog
```



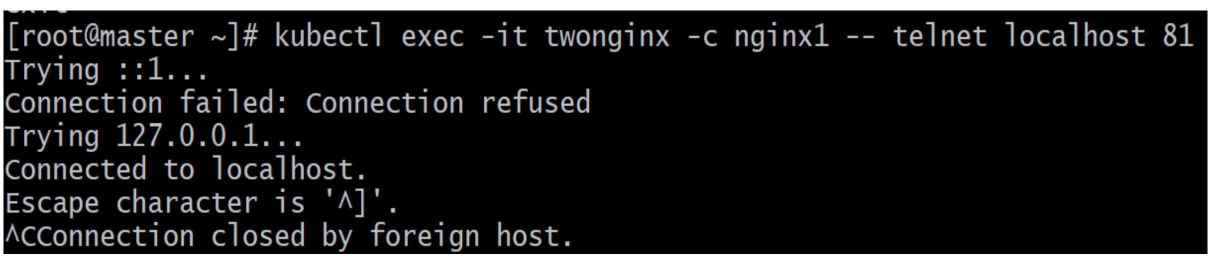
```
root@twonginx:/# apt install -y telnet
Installing:
  telnet

Installing dependencies:
  inetutils-telnet  netbase

Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading
  Download size: 186 kB
  Space needed: 377 kB / 18.5 GB available

Get:1 http://deb.debian.org/debian trixie/main amd64 netb
Get:2 http://deb.debian.org/debian trixie/main amd64 inet
Get:3 http://deb.debian.org/debian trixie/main amd64 teln
Fetched 186 kB in 0s (8001 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so
/Debconf/FrontEnd/Dialog.pm line 79, <STDIN> line 3.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
```

- **kubectl exec -it twonginx -c nginx2 -- /bin/bash**
- **apt update**
- **apt install -y telnet**

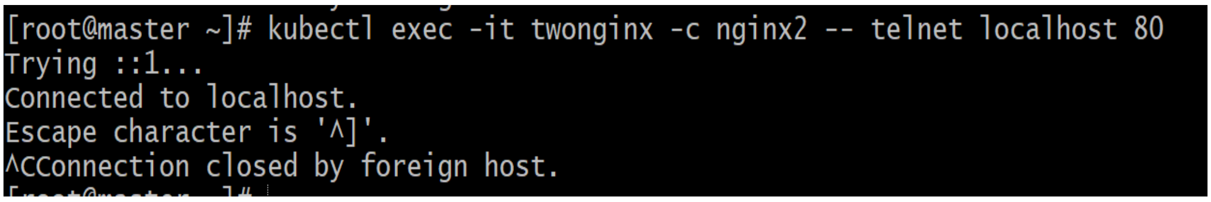


```
[root@master ~]# kubectl exec -it twonginx -c nginx2 -- /bin/bash
root@twonginx:/# apt update
apt install -y telnet
Get:1 http://deb.debian.org/debian trixie InRelease [140 kB]
Get:2 http://deb.debian.org/debian trixie-updates InRelease [47.3 kB]
Get:3 http://deb.debian.org/debian-security trixie-security InRelease [43.4
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
Get:6 http://deb.debian.org/debian-security trixie-security/main amd64 Packag
Fetched 9992 kB in 1s (14.1 MB/s)
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Installing:
  telnet

Installing dependencies:
  inetutils-telnet  netbase

Summary:
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 1
  Download size: 186 kB
  Space needed: 377 kB / 18.4 GB available

Get:1 http://deb.debian.org/debian trixie/main amd64 netbase all 6.5 [12.4 k
Get:2 http://deb.debian.org/debian trixie/main amd64 inetutils-telnet amd64
```

- **kubectl exec -it twonginx -c nginx1 -- telnet localhost 81**


```
[root@master ~]# kubectl exec -it twonginx -c nginx1 -- telnet localhost 81
Trying ::1...
Connection failed: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
^CConnection closed by foreign host.
```

- **kubectl exec -it twonginx -c nginx2 -- telnet localhost 80**

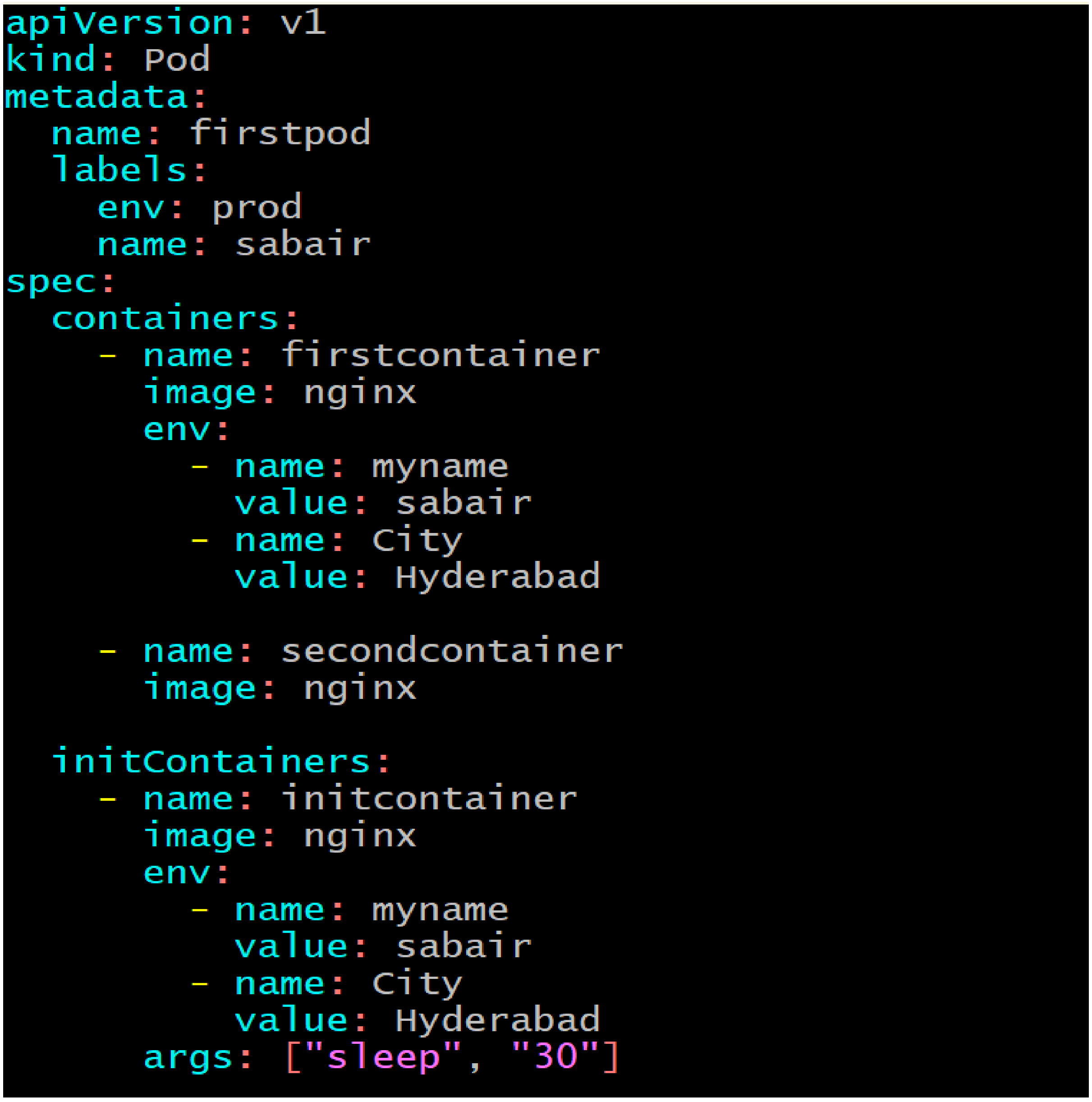

```
[root@master ~]# kubectl exec -it twonginx -c nginx2 -- telnet localhost 80
Trying ::1...
Connected to localhost.
Escape character is '^]'.
^CConnection closed by foreign host.
```

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.

Write this script in yaml file.

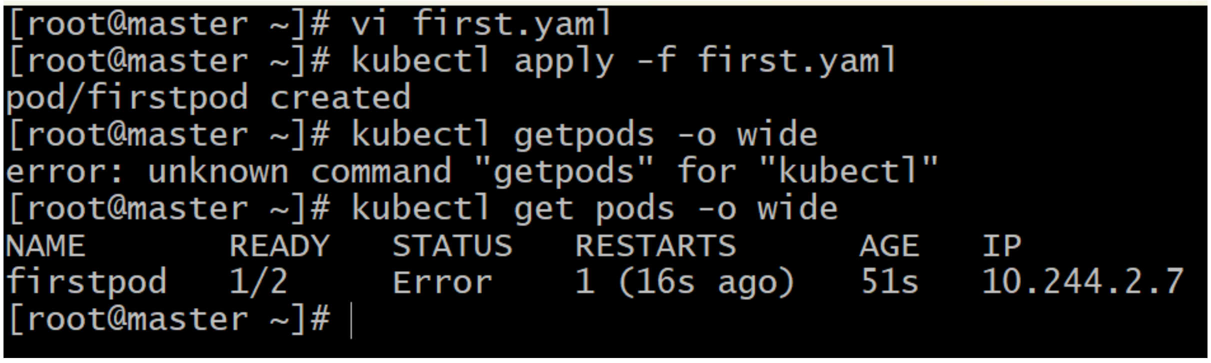
```
apiVersion: v1

kind: Pod

metadata:

  name: firstpod

  labels:

    env: prod

    name: sabair

spec:
```

```yaml
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"]
```



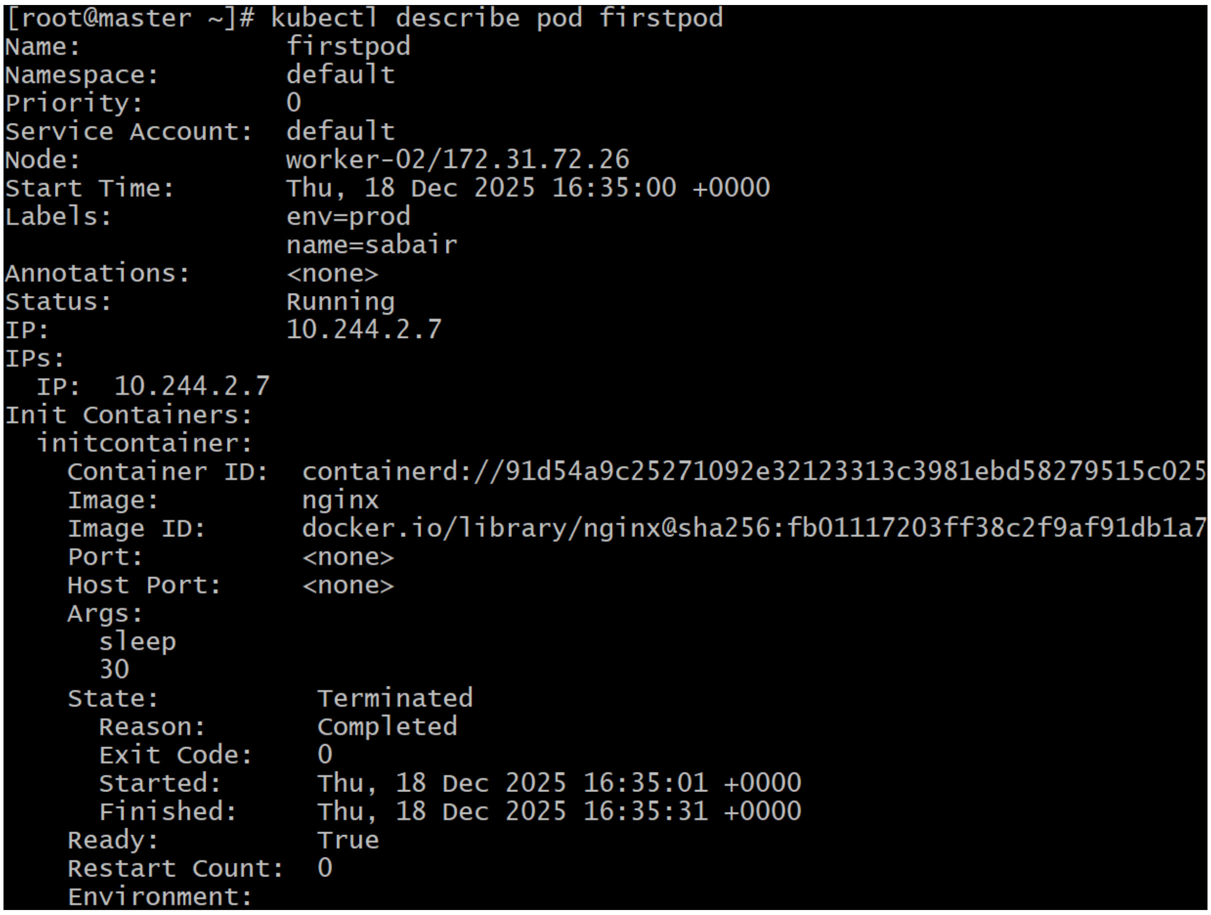
```
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
  labels:
    env: prod
    name: sabair
spec:
  containers:
    - name: firstcontainer
      image: nginx
      env:
        - name: myname
          value: sabair
        - name: City
          value: Hyderabad

    - name: secondcontainer
      image: nginx

  initContainers:
    - name: initcontainer
      image: nginx
      env:
        - name: myname
          value: sabair
        - name: City
          value: Hyderabad
      args: ["sleep", "30"]
```

- **kubectl apply -f first.yaml**
- **kubectl get pods**



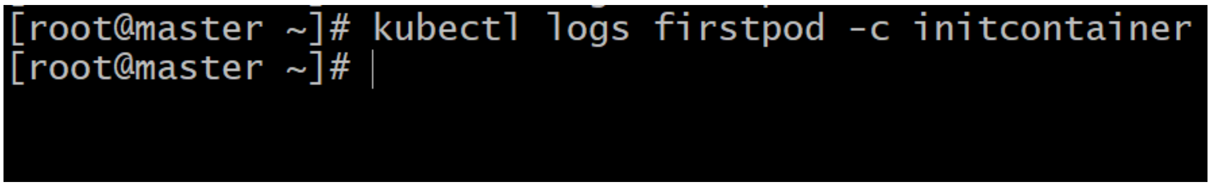
```
[root@master ~]# vi first.yaml
[root@master ~]# kubectl apply -f first.yaml
pod/firstpod created
[root@master ~]# kubectl getpods -o wide
error: unknown command "getpods" for "kubectl"
[root@master ~]# kubectl get pods -o wide
NAME         READY    STATUS    RESTARTS        AGE    IP
firstpod     1/2      Error     1 (16s ago)     51s    10.244.2.7
[root@master ~]# 
```

- **kubectl describe pod firstpod**



```
[root@master ~]# kubectl describe pod firstpod
Name:             firstpod
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker-02/172.31.72.26
Start Time:       Thu, 18 Dec 2025 16:35:00 +0000
Labels:           env=prod
                  name=sabair
Annotations:      <none>
Status:           Running
IP:               10.244.2.7
IPs:
  IP:  10.244.2.7
Init Containers:
  initcontainer:
    Container ID:  containerd://91d54a9c25271092e32123313c3981ebd58279515c025
    Image:         nginx
    Image ID:      docker.io/library/nginx@sha256:fb01117203ff38c2f9af91db1a7
    Port:          <none>
    Host Port:     <none>
    Args:
      sleep
      30
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Thu, 18 Dec 2025 16:35:01 +0000
      Finished:     Thu, 18 Dec 2025 16:35:31 +0000
    Ready:          True
    Restart Count:  0
    Environment:
```

- **kubectl logs firstpod -c initcontainer**



```
[root@master ~]# kubectl logs firstpod -c initcontainer
[root@master ~]#
```

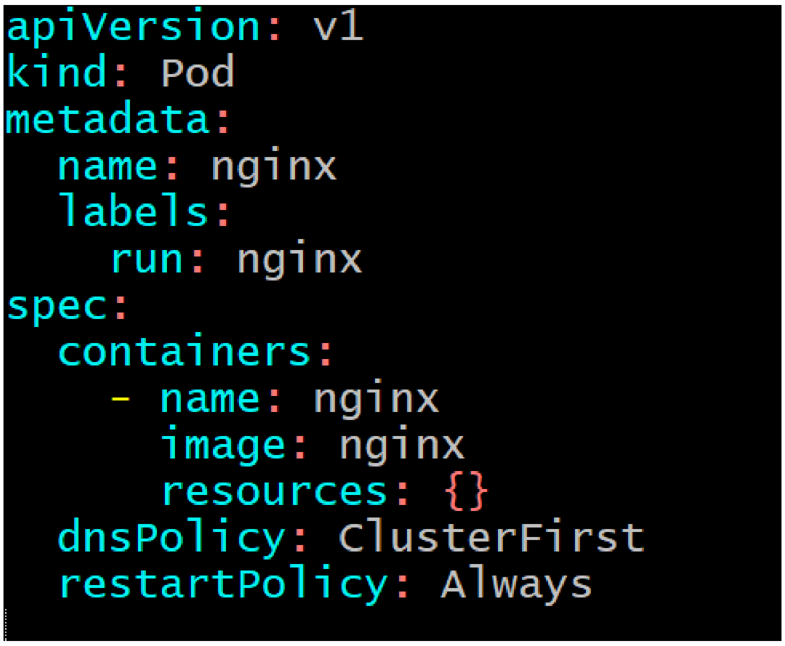
6. Run a Dry Run Command to Generate YAML Task: Use the kubectl run nginx --image=nginx --dry-run=client -o yaml command to generate a Pod YAML definition. Modify the generated YAML to suit specific requirements (e.g., labels or environment variables) and deploy it.

- **vi dryrun-pod.yml**

apiVersion: v1

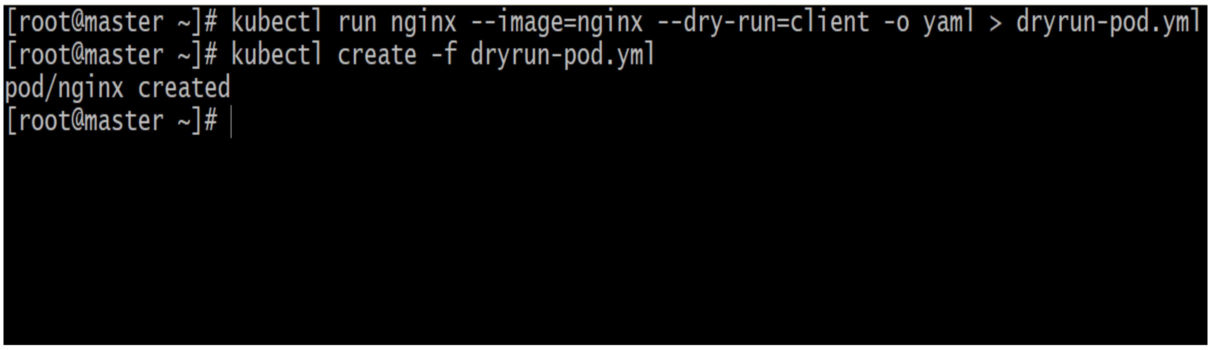
kind: Pod

```
metadata:
  name: nginx
  labels:
    run: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    run: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```

- **kubectl run nginx --image=nginx --dry-run=client -o yaml > dryrun-pod.yml**
- **kubectl create -f dryrun-pod.yml**



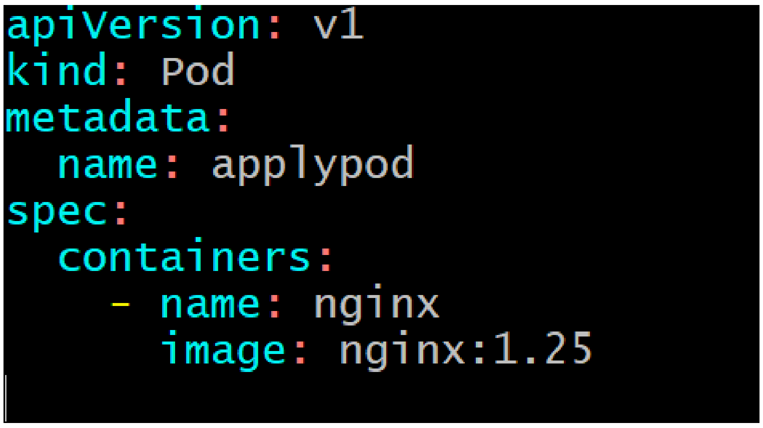
```
[root@master ~]# kubectl run nginx --image=nginx --dry-run=client -o yaml > dryrun-pod.yml
[root@master ~]# kubectl create -f dryrun-pod.yml
pod/nginx created
[root@master ~]# 
```

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 firstpod.yml**

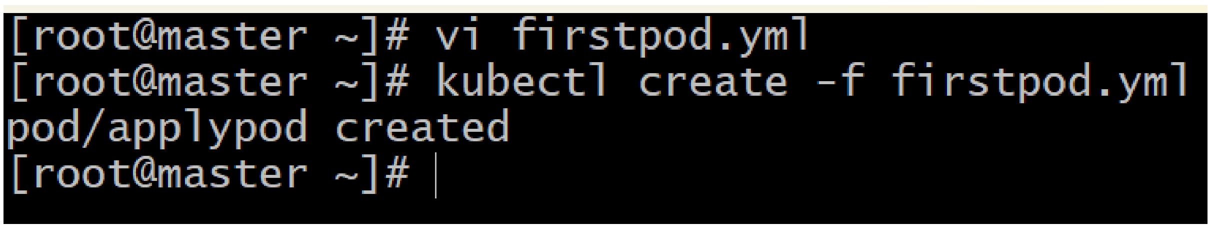
add this script in the yaml file.

```
apiVersion: v1

kind: Pod

metadata:

  name: applypod

spec:

  containers:

    - name: nginx

      image: nginx:1.25
```



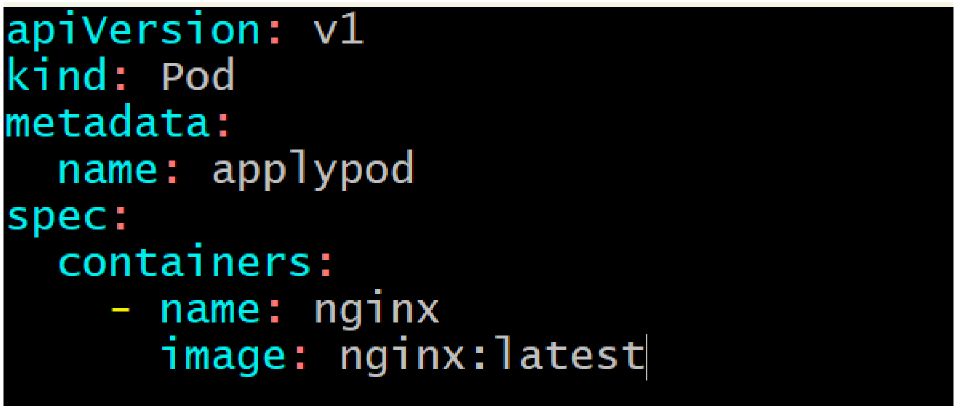
```
apiVersion: v1
kind: Pod
metadata:
  name: applypod
spec:
  containers:
    - name: nginx
      image: nginx:1.25
```

- **kubectl create -f firstpod.yml**



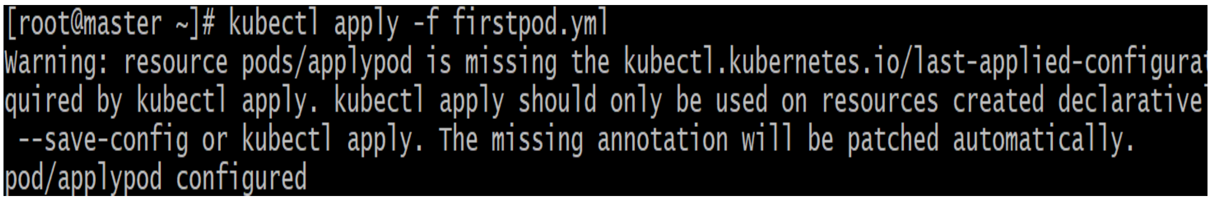
```
[root@master ~]# vi firstpod.yml
[root@master ~]# kubectl create -f firstpod.yml
pod/applypod created
[root@master ~]#
```

In that same file edit the image version as latest.



```
apiVersion: v1
kind: Pod
metadata:
  name: applypod
spec:
  containers:
    - name: nginx
      image: nginx:latest
```

- **kubectl apply -f firstpod.yml**

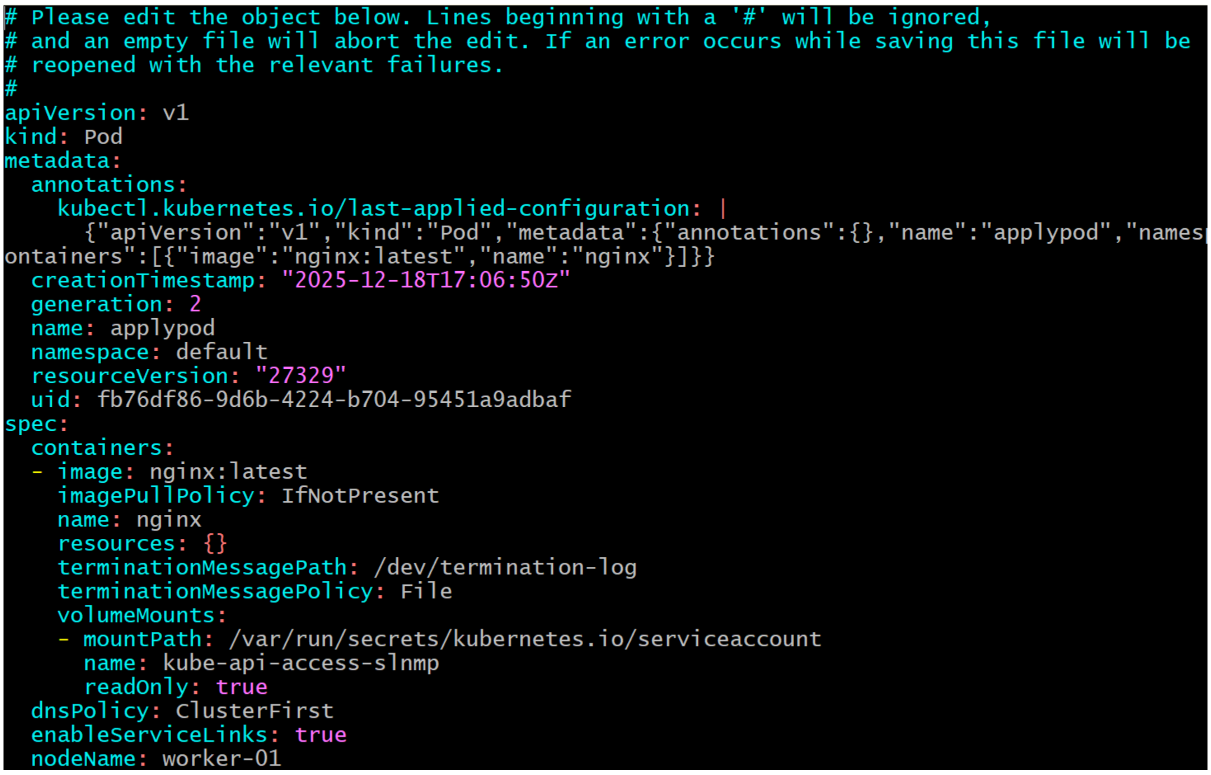


```
[root@master ~]# kubectl apply -f firstpod.yml
Warning: resource pods/applypod is missing the kubectl.kubernetes.io/last-applied-configura
quired by kubectl apply. kubectl apply should only be used on resources created declarative
 --save-config or kubectl apply. The missing annotation will be patched automatically.
pod/applypod configured
```

8. Edit an Existing Pod Configuration Task: Use kubectl edit pod <pod_name> to modify the running Pod's environment

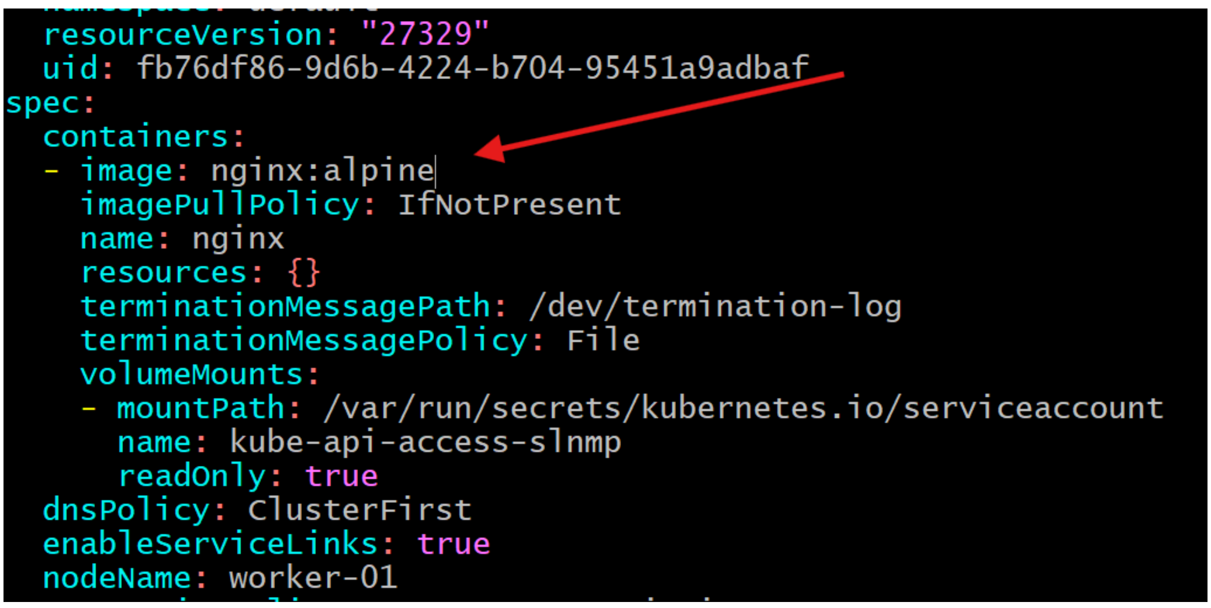
variables or image. After making the changes, verify if they took effect by checking the container logs or environment variables using kubectl exec.

- **Kubectl edit pod applypod**

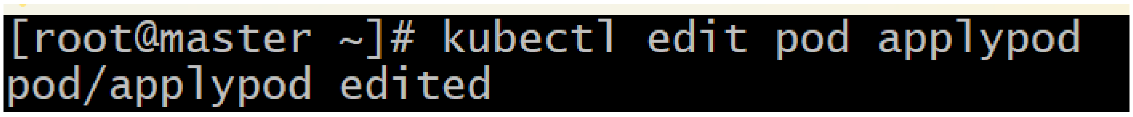


```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"applypod","names
ontainers":[{"image":"nginx:latest","name":"nginx"}]}}
  creationTimestamp: "2025-12-18T17:06:50Z"
  generation: 2
  name: applypod
  namespace: default
  resourceVersion: "27329"
  uid: fb76df86-9d6b-4224-b704-95451a9adbaf
spec:
  containers:
  - image: nginx:latest
    imagePullPolicy: IfNotPresent
    name: nginx
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-slnmp
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: worker-01
```

change the image version as alpine

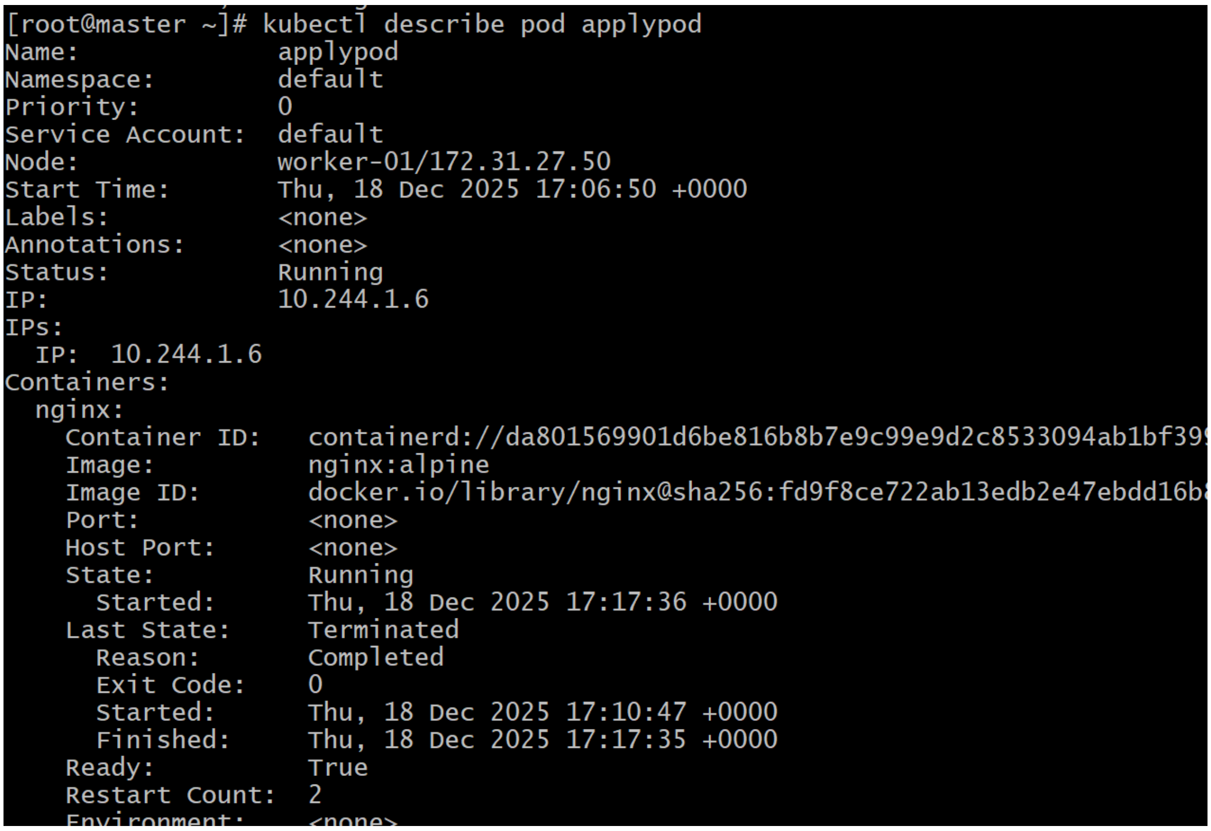


```
  resourceVersion: "27329"
  uid: fb76df86-9d6b-4224-b704-95451a9adbaf
spec:
  containers:
  - image: nginx:alpine
    imagePullPolicy: IfNotPresent
    name: nginx
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-slnmp
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: worker-01
```



```
[root@master ~]# kubectl edit pod applypod
pod/applypod edited
```

- **kubectl describe pod applypod**



```
[root@master ~]# kubectl describe pod applypod
Name:             applypod
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker-01/172.31.27.50
Start Time:       Thu, 18 Dec 2025 17:06:50 +0000
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               10.244.1.6
IPs:
  IP:  10.244.1.6
Containers:
  nginx:
    Container ID:   containerd://da801569901d6be816b8b7e9c99e9d2c8533094ab1bf39
    Image:          nginx:alpine
    Image ID:       docker.io/library/nginx@sha256:fd9f8ce722ab13edb2e47ebdd16b
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 18 Dec 2025 17:17:36 +0000
    Last State:     Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Thu, 18 Dec 2025 17:10:47 +0000
      Finished:     Thu, 18 Dec 2025 17:17:35 +0000
    Ready:          True
    Restart Count:  2
    Environment:    <none>
```

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.

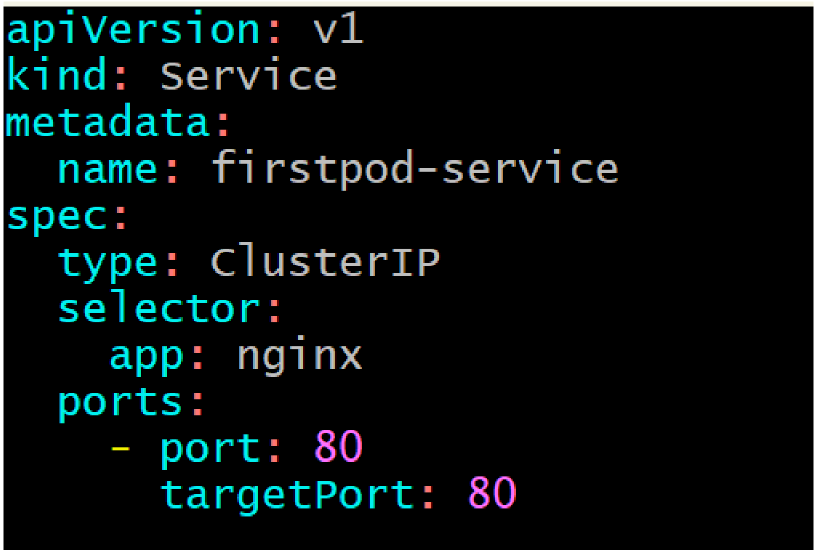
Write this script in your yml file.

apiVersion: v1

kind: Service

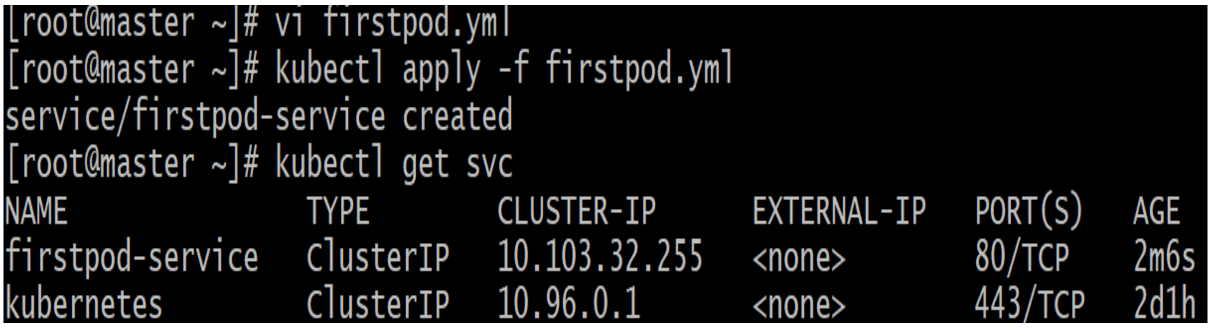
metadata:

```
  name: firstpod-service
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```



```
apiVersion: v1
kind: Service
metadata:
  name: firstpod-service
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - port: 80
      targetPort: 80
```

- **kubectl apply -f firstpod.yml**
- **kubectl get svc**



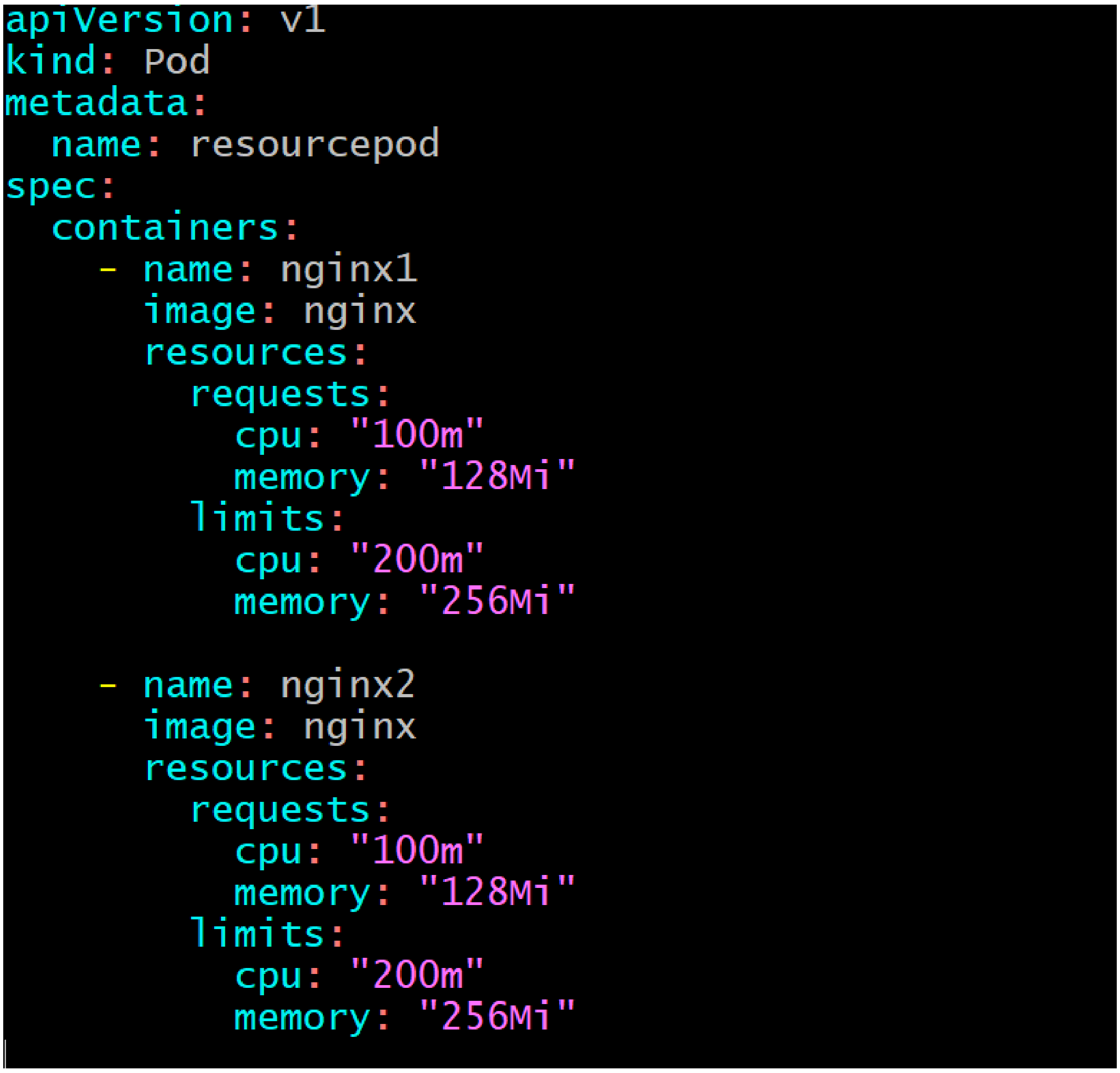
```
[root@master ~]# vi firstpod.yml
[root@master ~]# kubectl apply -f firstpod.yml
service/firstpod-service created
[root@master ~]# kubectl get svc
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
firstpod-service   ClusterIP   10.103.32.255   <none>        80/TCP    2m6s
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP   2d1h
```

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.

give this script in yaml file.

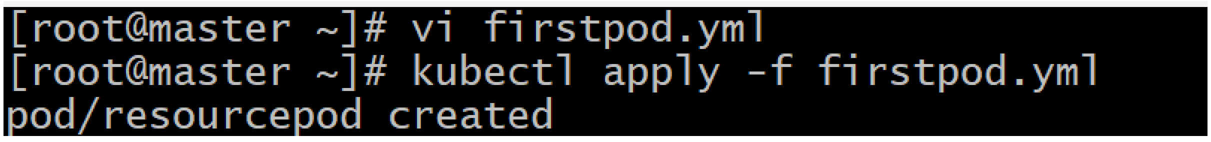
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: resourcepod
spec:
  containers:
    - name: nginx1
      image: nginx
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "200m"
          memory: "256Mi"
```

```yaml
- name: nginx2
  image: nginx
  resources:
    requests:
      cpu: "100m"
      memory: "128Mi"
    limits:
      cpu: "200m"
      memory: "256Mi"
```



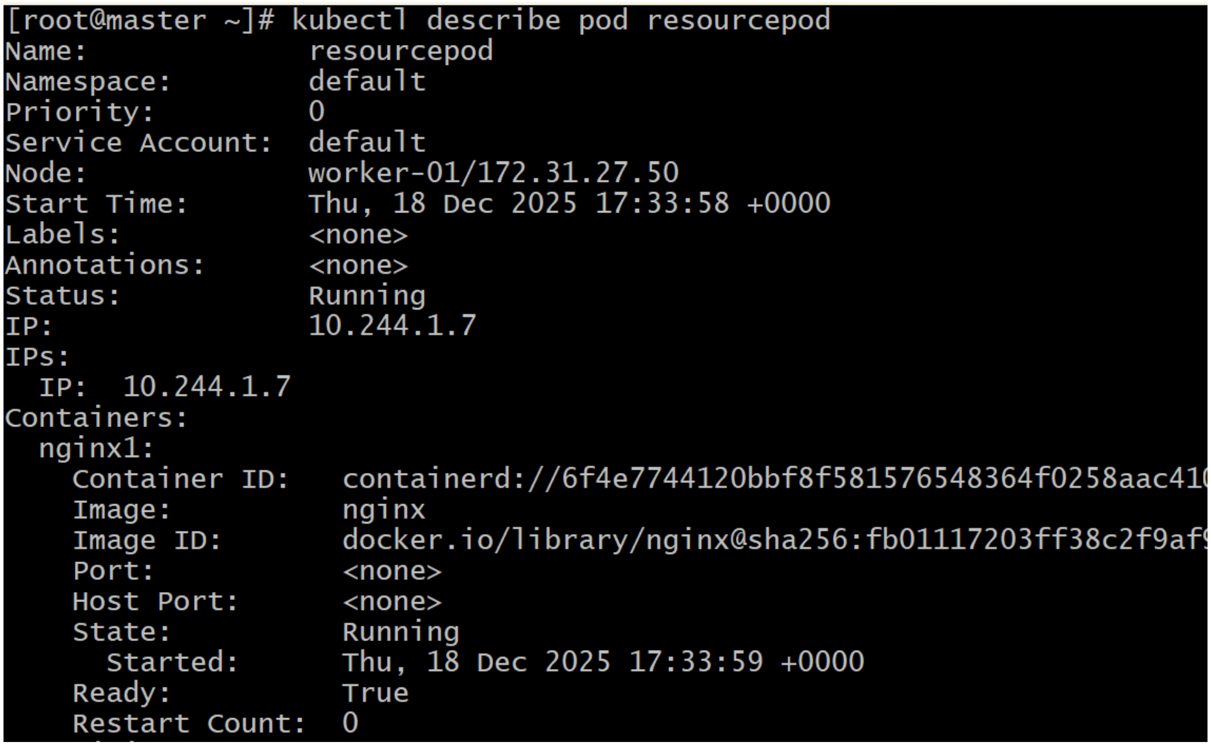
```
apiVersion: v1
kind: Pod
metadata:
  name: resourcepod
spec:
  containers:
    - name: nginx1
      image: nginx
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "200m"
          memory: "256Mi"

    - name: nginx2
      image: nginx
      resources:
        requests:
          cpu: "100m"
          memory: "128Mi"
        limits:
          cpu: "200m"
          memory: "256Mi"
```

- **kubectl apply -f firstpod.yml**



```
[root@master ~]# vi firstpod.yml
[root@master ~]# kubectl apply -f firstpod.yml
pod/resourcepod created
```

- **kubectl describe pod resourcepod**



```
[root@master ~]# kubectl describe pod resourcepod
Name:                 resourcepod
Namespace:            default
Priority:             0
Service Account:      default
Node:                 worker-01/172.31.27.50
Start Time:           Thu, 18 Dec 2025 17:33:58 +0000
Labels:               <none>
Annotations:          <none>
Status:               Running
IP:                   10.244.1.7
IPs:
  IP:  10.244.1.7
Containers:
  nginx1:
    Container ID:   containerd://6f4e7744120bbf8f581576548364f0258aac41
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:fb01117203ff38c2f9af
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 18 Dec 2025 17:33:59 +0000
    Ready:          True
    Restart Count:  0
```