**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.

Create a file first.yaml Now write the script this is for checking the errors

apiVersion: v1

kind: Pod

metadata:

    name: firstpod

    labels:

      env: prod

spec:

  containers:

    - name: test

      image: nginx

vi first.yaml

apiVersion: v1

kind: Pod

metadata:

name: firstpod

labels:

env: prod

spec:

containers:

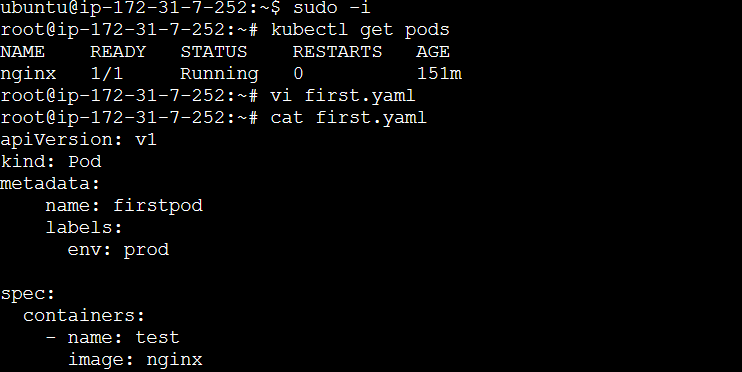
- name: test

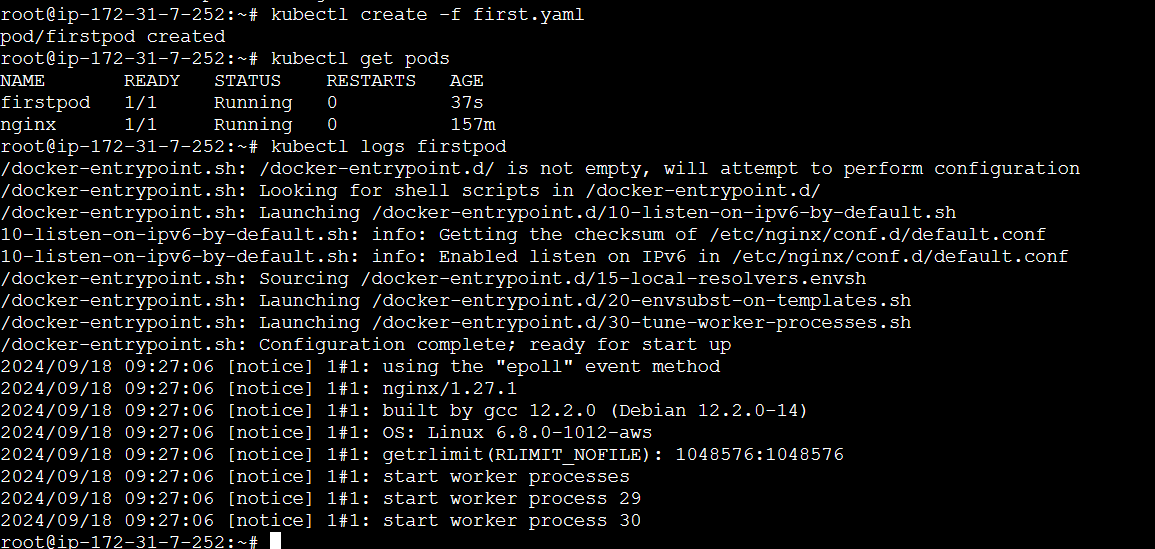
image: nginx

now, To create pod using yaml file:

kubectl create -f <file\_name>.yml --dry-run --> to check the output

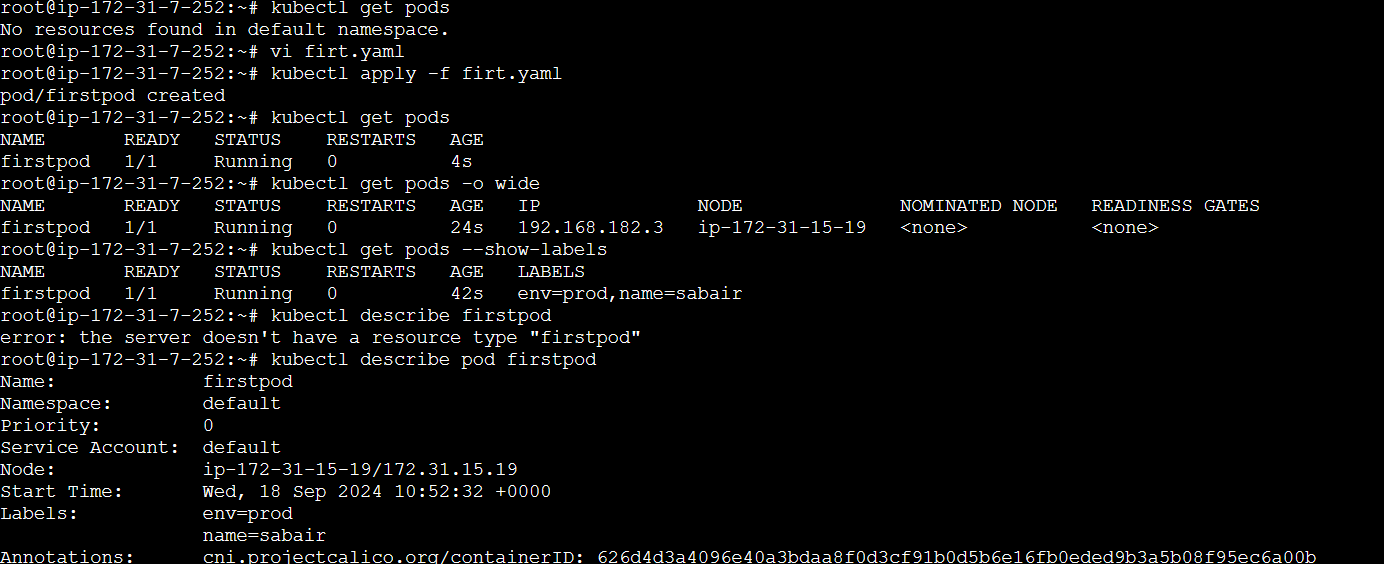
kubectl create -f <file\_name>.yml

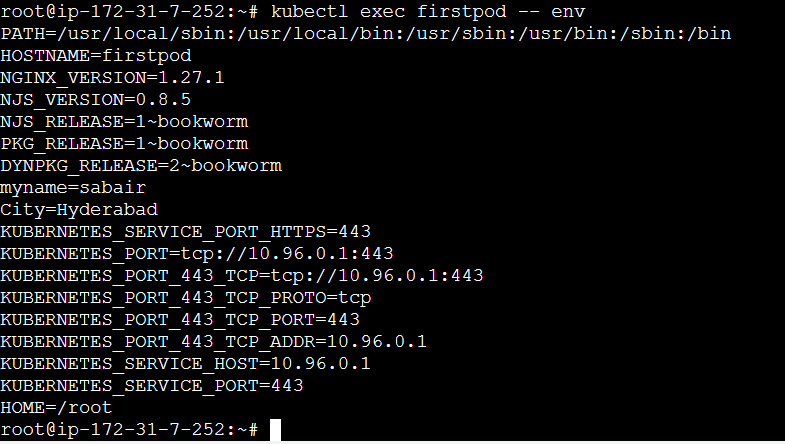




2)Set Environment Variables in a Pod

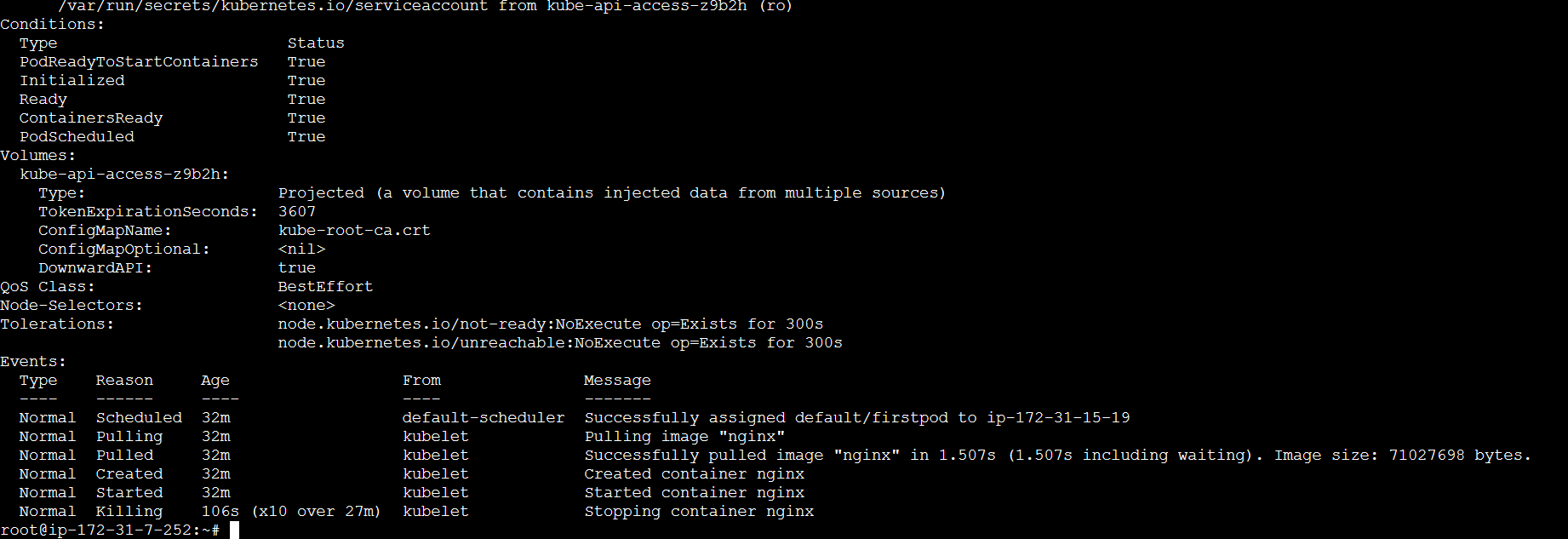
Task: Modify the YAML file to include environment variables myname: sabair and City: Hyderabad. Deploy the Pod and use kubectl exec <pod\_name> -- env to check if the environment variables are set properly.





4) 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.



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

    name: Anil

spec:

  containers:

  - name: firstcontainer

    image: nginx

    env:

      - name: myname

        value: anil

      - name: City

        value: Hyderabad

  - name: secondcontainer

    image: nginx

  initContainers:

  - name: initcontainer

    image: nginx

    env:

      - name: myname

        value: sabair

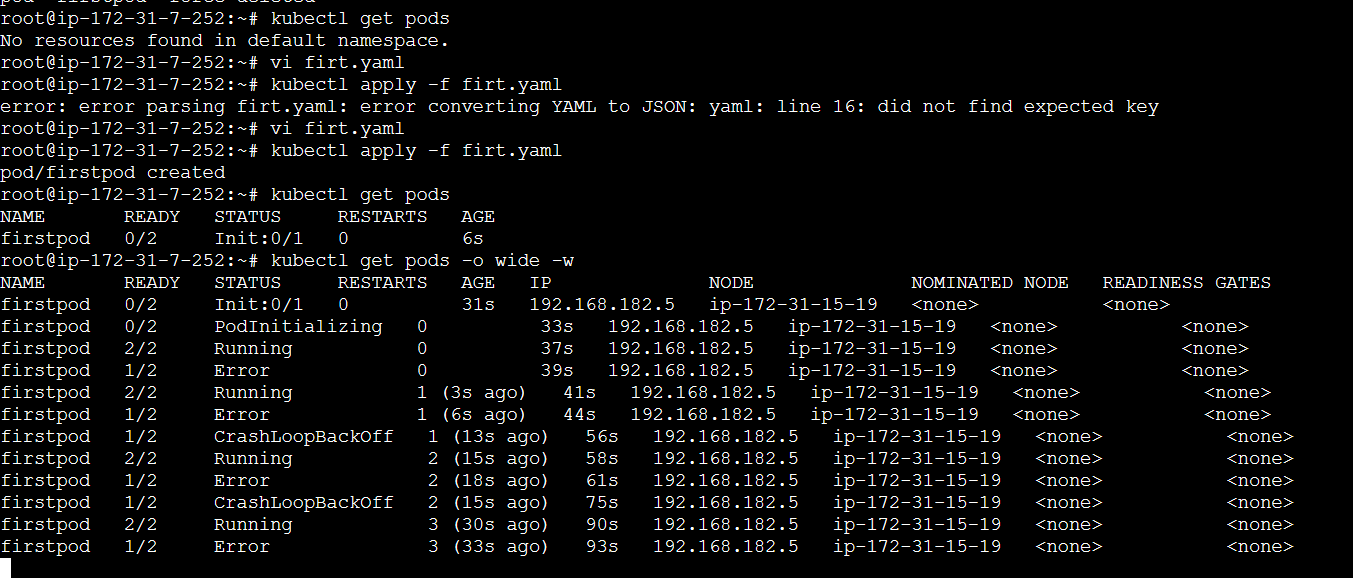
      - name: City

        value: Hyderabad

    args: ["sleep", "30"]

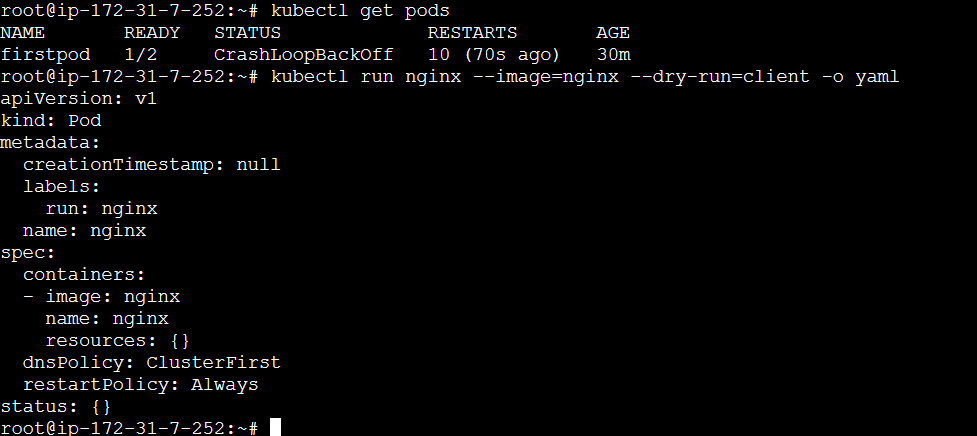
kubectl apply -f firt.yaml

kubectl get pods

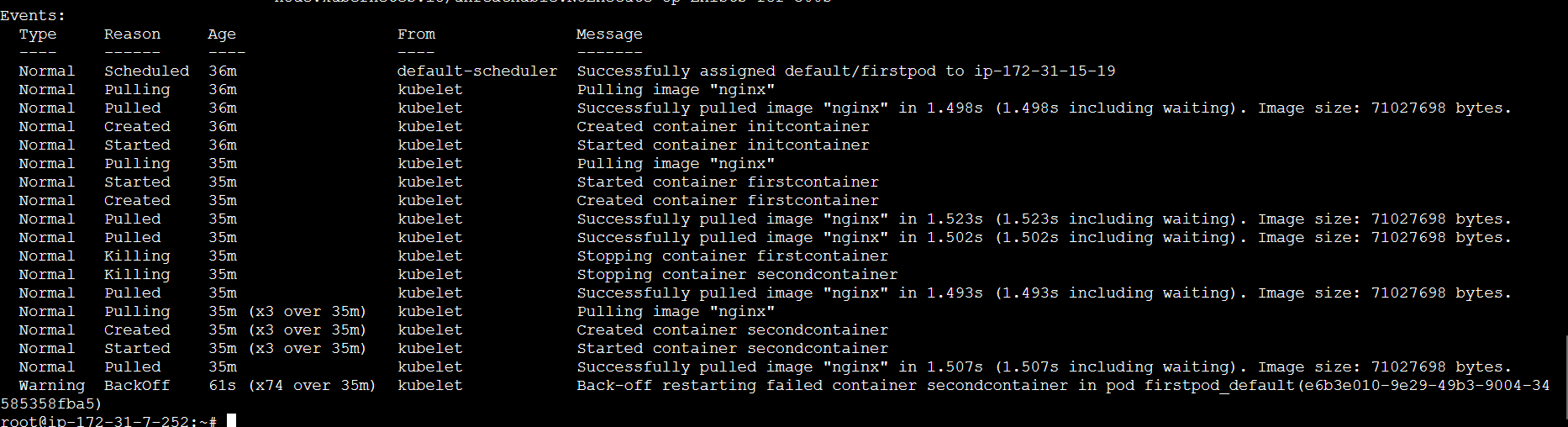


6) Run a Dry Run Command to Generate YAML

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.

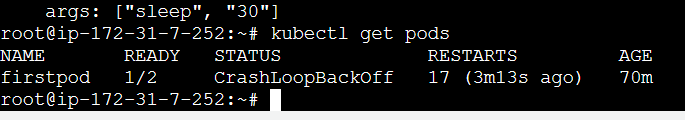


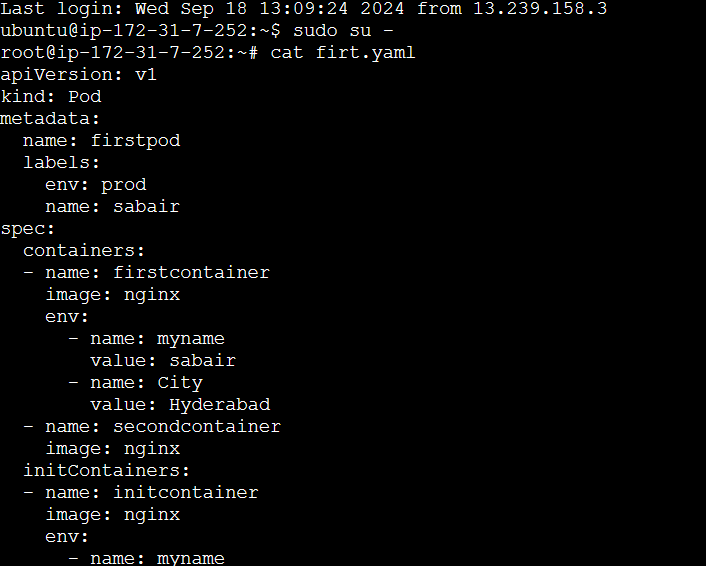
Kubectl describe pod firstpod

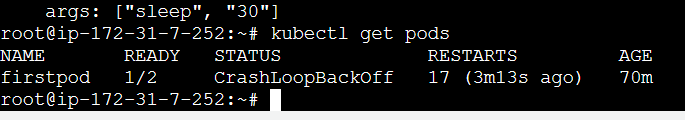


7) Use kubectl apply vs kubectl create

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 command

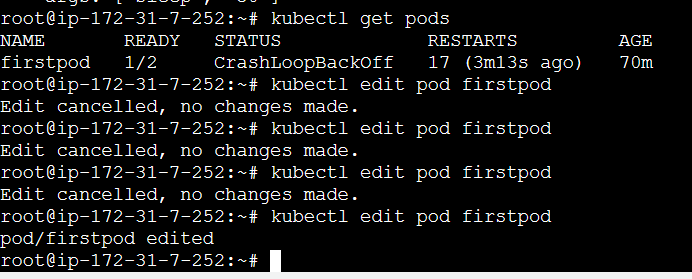






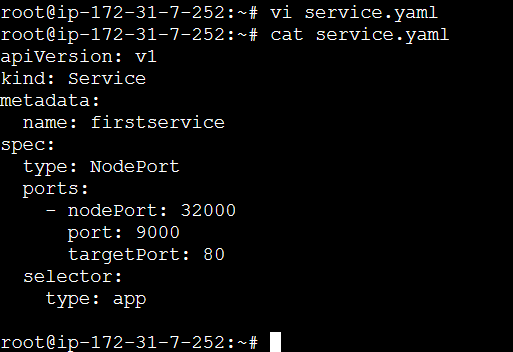
8) Edit an Existing Pod Configuration

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.



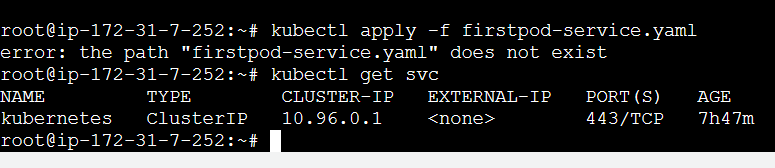
9) Expose a Pod Using a Service

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.



Ensure that your service is exposing the pod on port 80 and verify it using

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.

