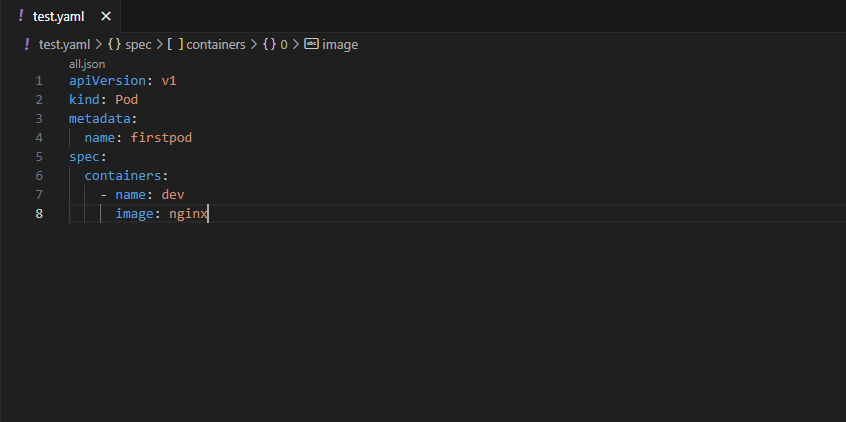
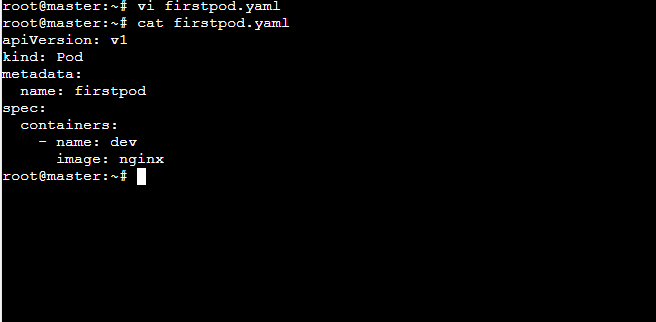
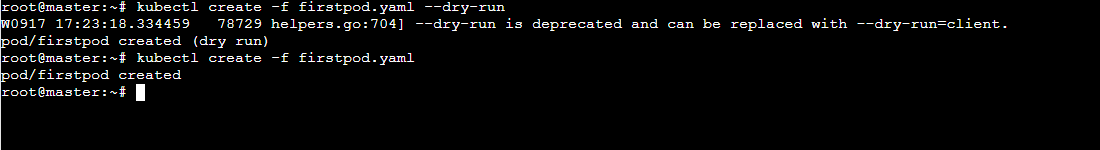
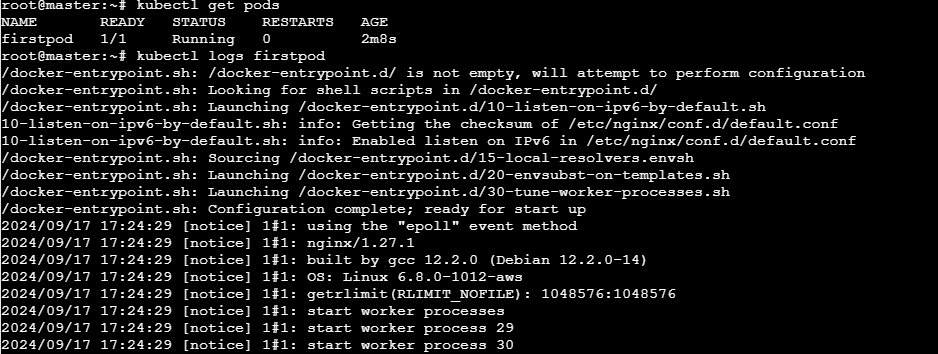
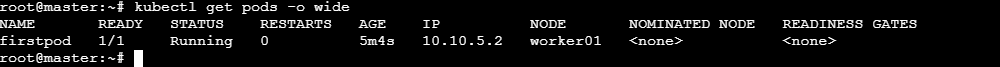
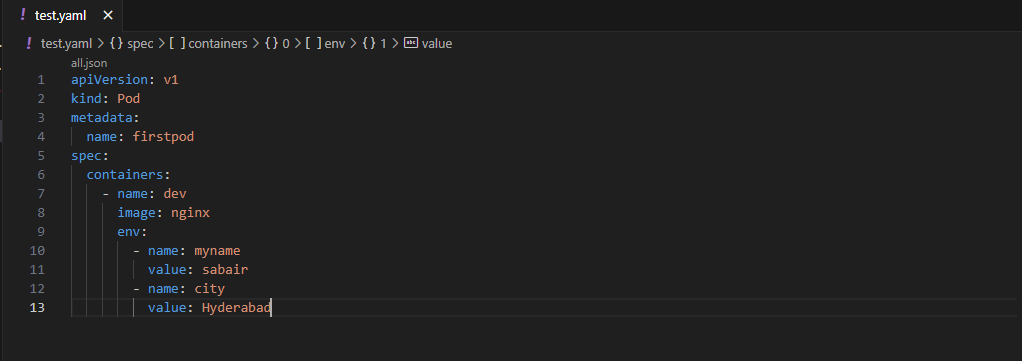
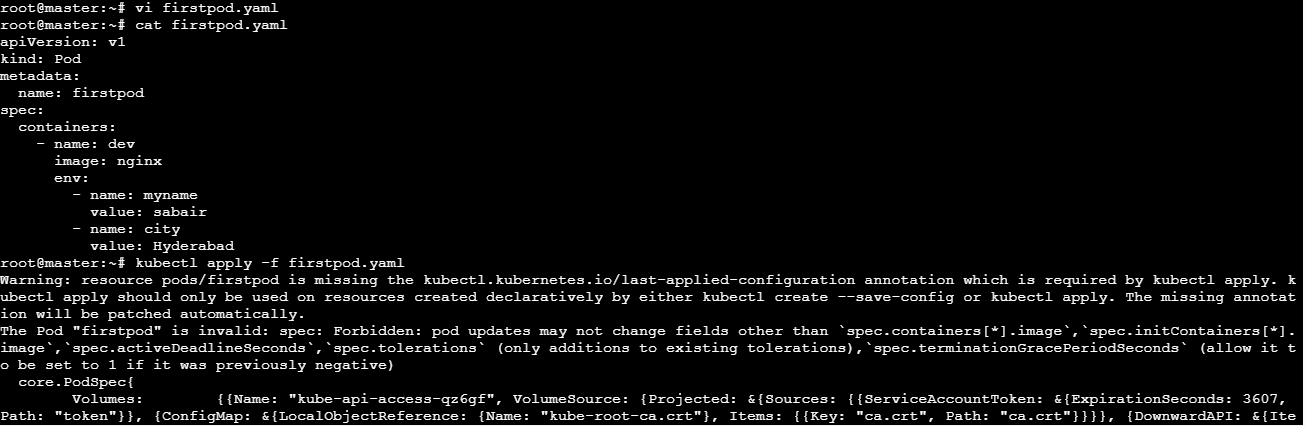
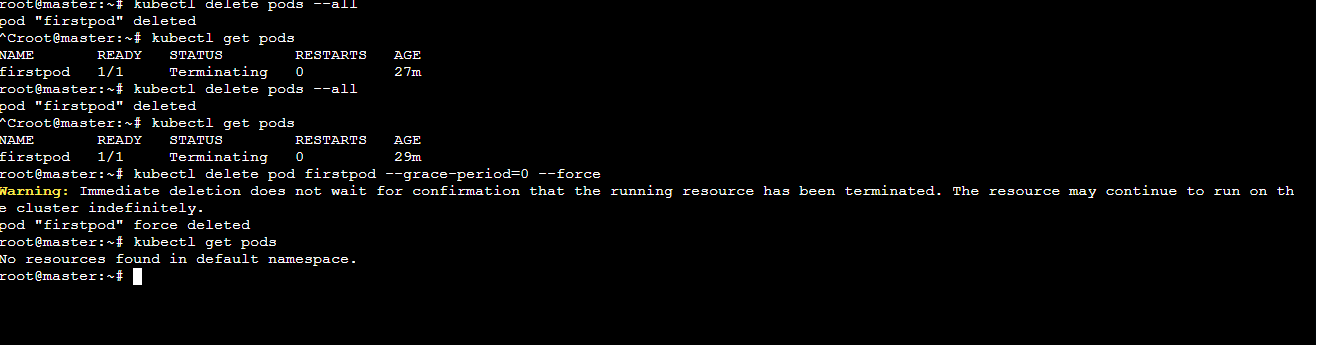
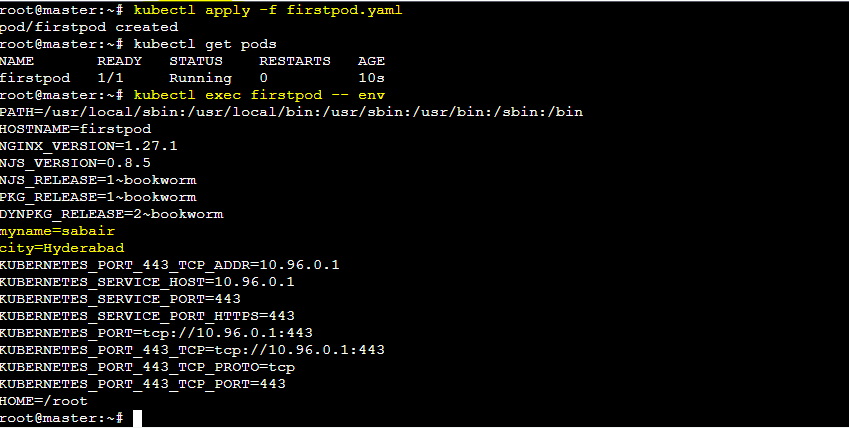
1) Create a Simple Pod Using YAML

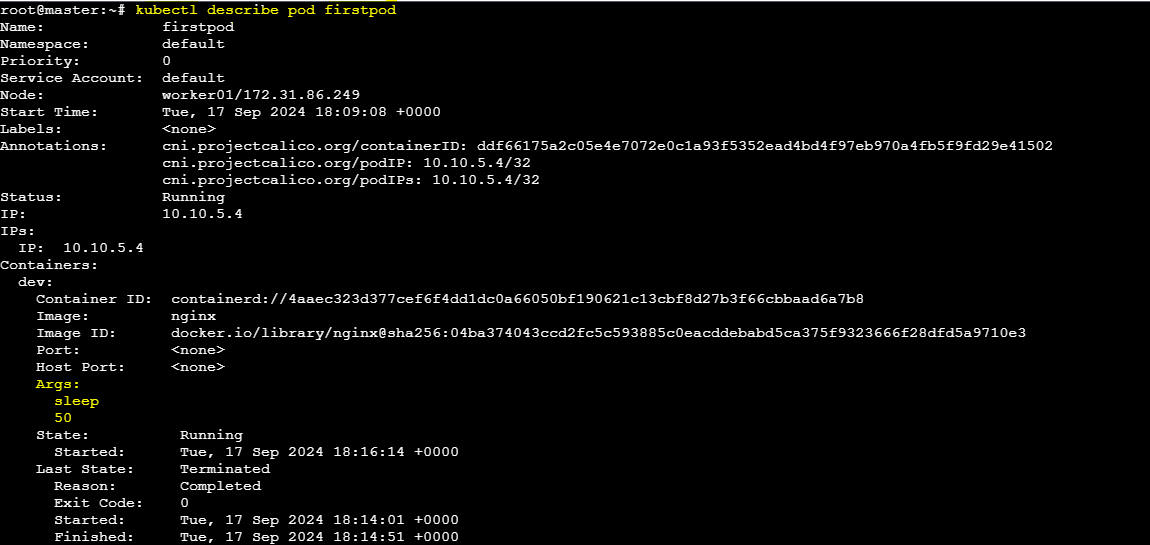
Task: Write a YAML file to create a Pod named firstpod with an nginx container. Verify the Pod creation using kubectl get pods and check the logs of the container using kubectl logs firstpod.  
  
Tried to write the code in v s code:  
  
then created a YAML file in master server to run this code:  
  
  
  
  
  
checked the code is correct or not with the help of dry run and then executed the command create to create the pod with required specifications:  
  
  
Verified the Pod creation using kubectl get pods and also checked the logs of the container using kubectl logs firstpod:  


Detailed information regarding our pods:  
  
Deleted the pod with: $kubectl delete pod firstpod --grace-period=0 --force

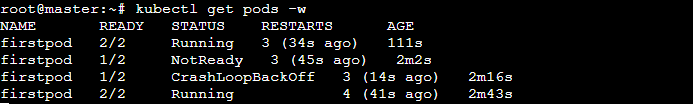
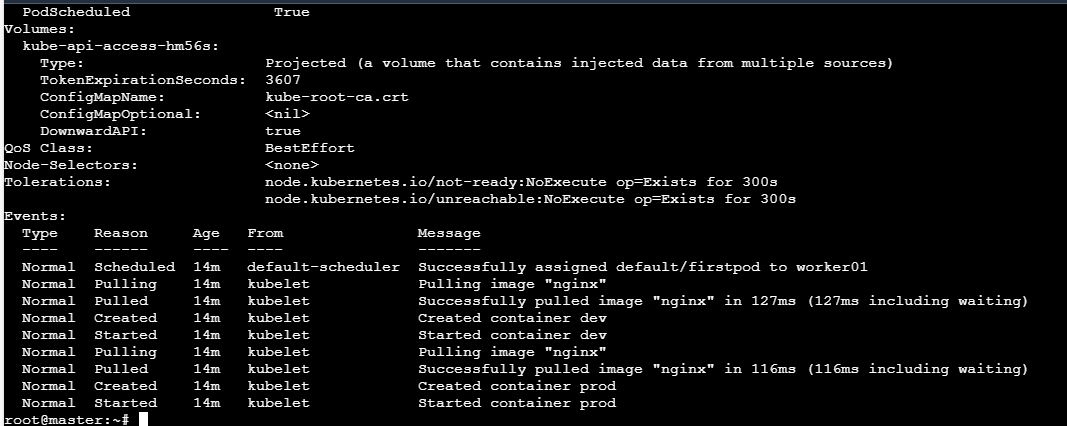
2) Set Environment Variables in a Pod

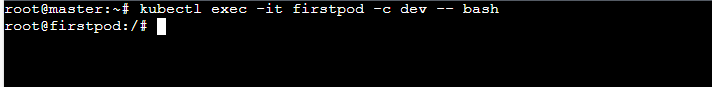
Task: Modify the YAML file to include environment variables myname: sabair and City: Hyderabad. Deploy the Pod and use kubectl exec <pod\_name> -- env to check if the environment variables are set properly.   
  
  
  
I have tried to edit the Yaml file added the env variables in it and tried to use apply command but got an error:  
  
So, I have deleted the pod and executed the yaml file again with variables:  
  
while deleting the pod was not deleting so I have used forced command to delete forcefully and it has been deleted :  
  
  
  
  
Now creating new pod with the environment variables:  
  
  
Deleted the pod with: $kubectl delete pod firstpod --grace-period=0 --force

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.  
  
  
Created the pod:  
  
  
used kubectl describe pod to verify the args are correctly passed to the container:  
  
Deleted the pod with: $kubectl delete pod firstpod --grace-period=0 --force

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).  
  
  
  
  
created the pod with two containers:  
  
Monitored the containers:  
With the help of describe command checked the container are created or not: $kubectl describe pod firstpod  


Used the exec command and login to both the containers: $ kubectl exec -it firstpod -c dev -- bash  
  
to connect the another container used this command $netstat –nltp but it was not installed in the container:  
root@firstpod:/# cat /etc/resolv.conf

search default.svc.cluster.local svc.cluster.local cluster.local ec2.internal

nameserver 10.96.0.10

options ndots:5

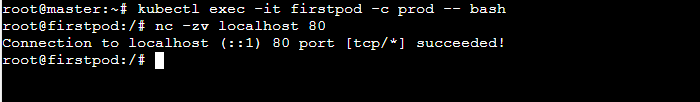
### root@firstpod:/# echo "nameserver 8.8.8.8" > /etc/resolv.conf root@firstpod:/# apt-get update && apt-get install -y net-tools $netstat -nltp nc -zv localhost 80 Steps to Install Netcat in the prod Container

1. **Check the Base Image:** The prod container might be based on a minimal image that doesn't include nc. You can check which image it uses by describing the Pod:

kubectl describe pod firstpod

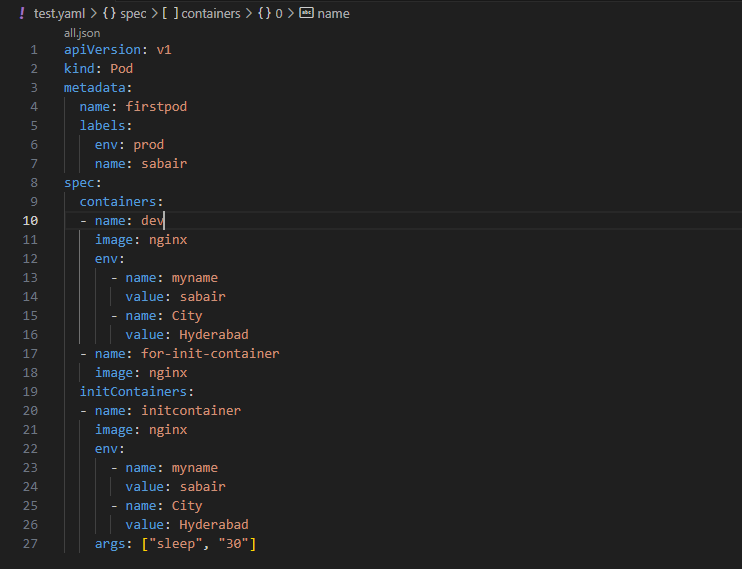
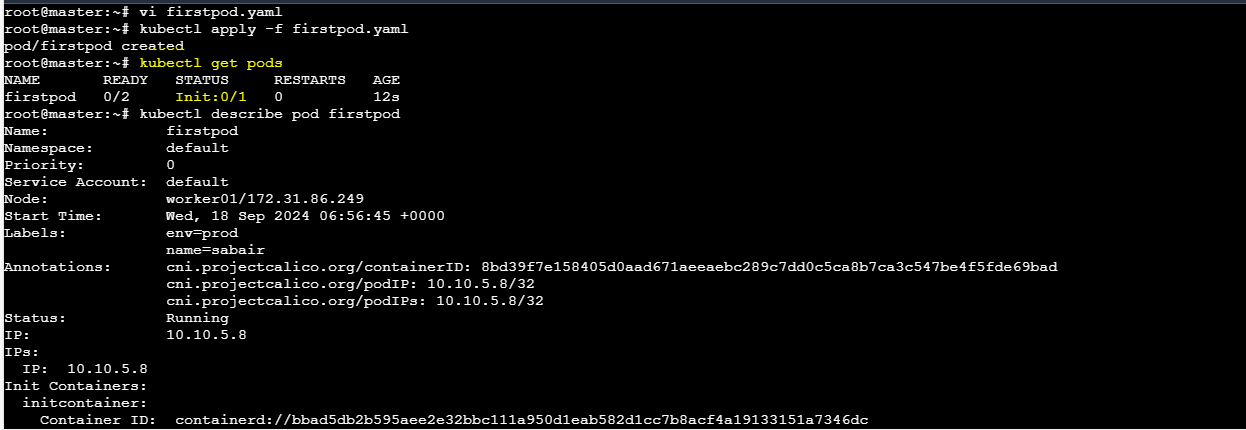
1. **Install Netcat:** If you can install packages in the prod container, you can try installing netcat. The method depends on the base image:
   * **For Debian/Ubuntu-based images:**

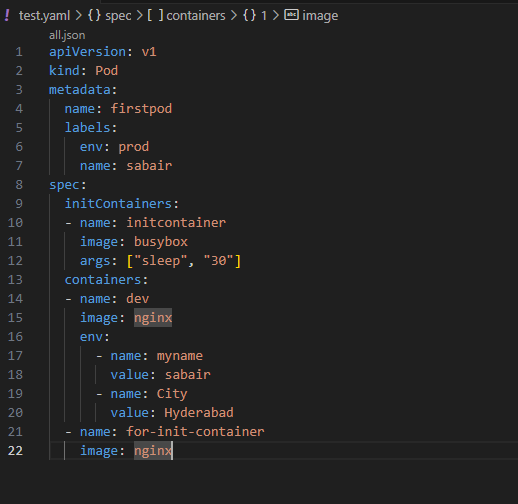
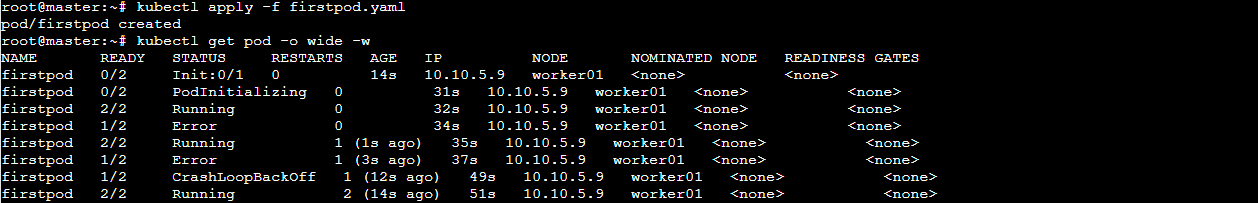
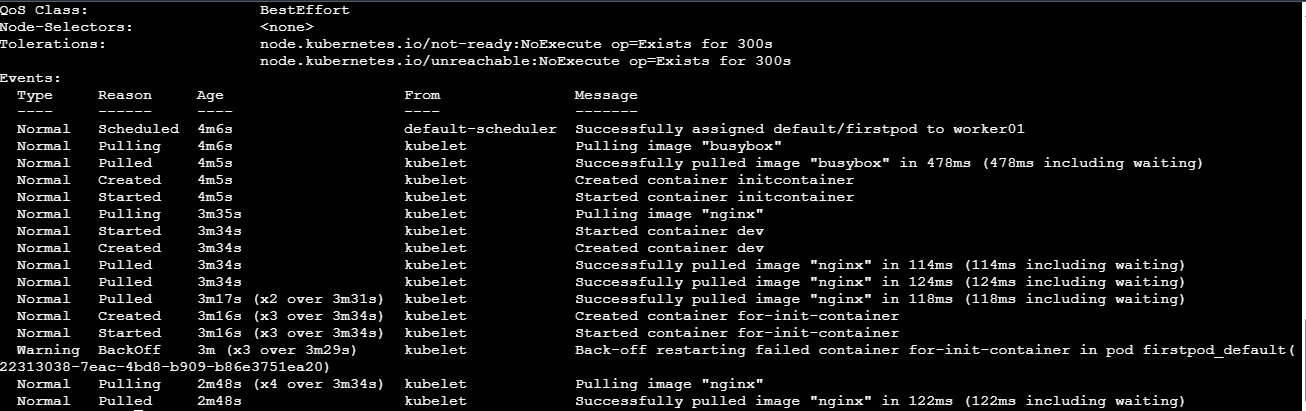
apt-get update && apt-get install -y netcat-openbsd

**nc -zv localhost 80**Deleted the pod with: $kubectl delete pod firstpod --grace-period=0 --force

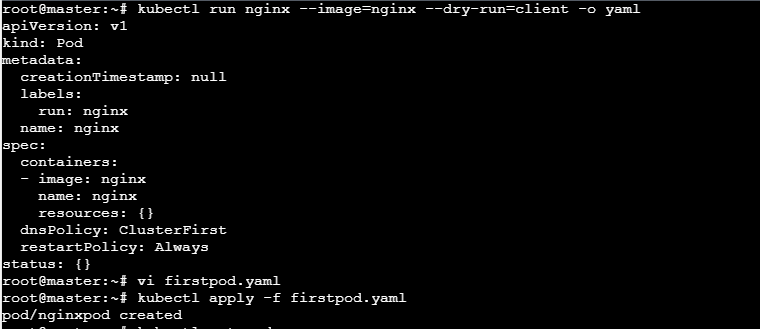
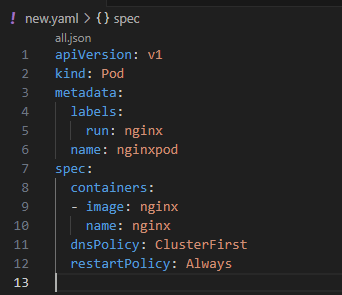
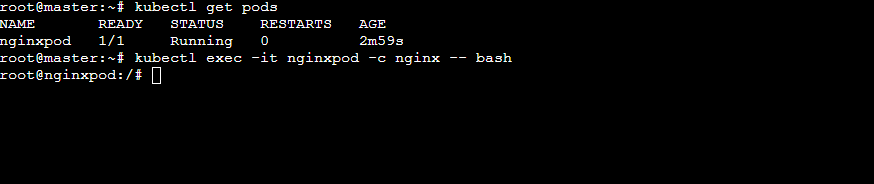
5) Set Up an Init Container in a Pod

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.

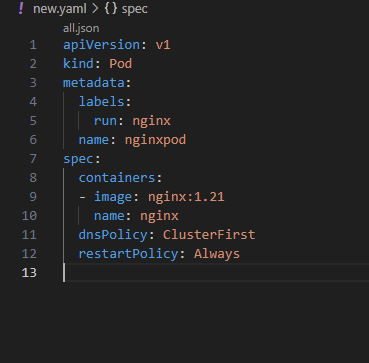
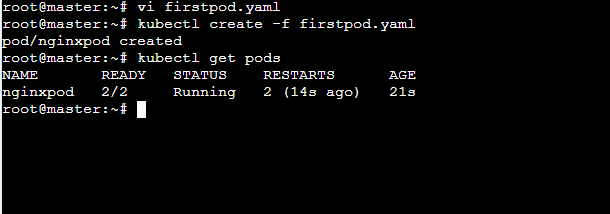
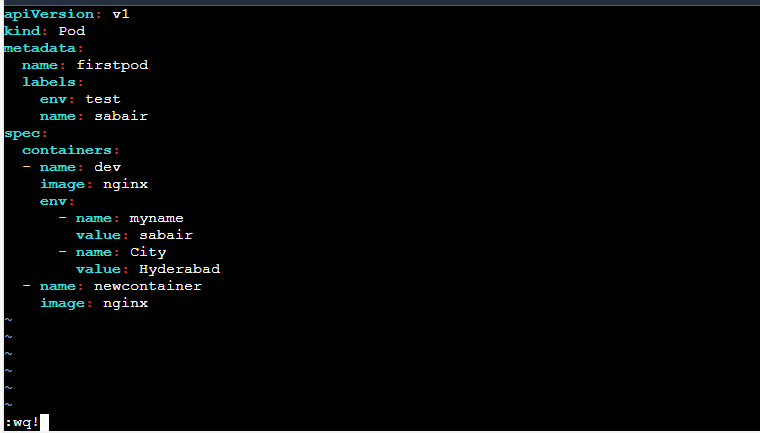
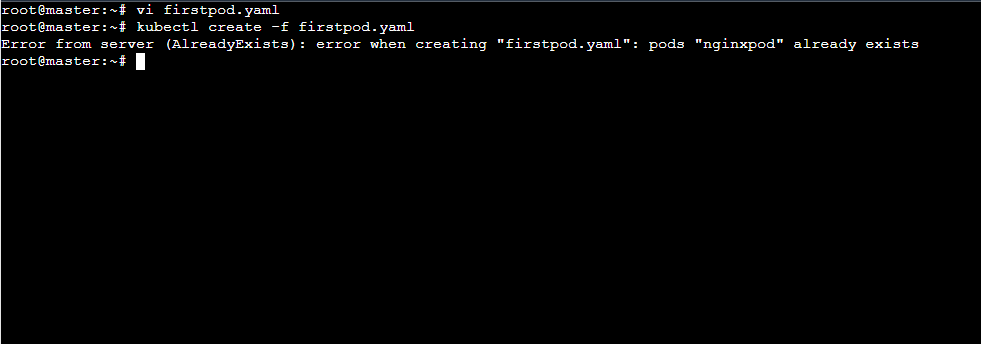
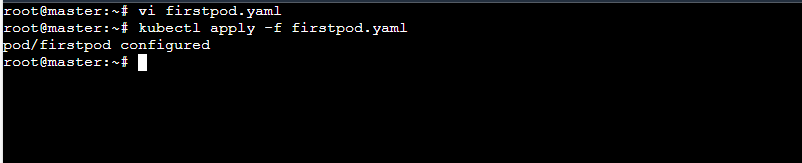
Created the container within the pod:  


As the second container was getting failed due some loopcrash issue I have change the script to a loght weight image of busy box and now executing the script:  
  
Done with the execution and checked with the monitoring command:  
  
also with the describe command:  
  
Deleted the pod with: $kubectl delete pod firstpod --grace-period=0 --force  
  
  
  
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.  
Ran this command: $kubectl run nginx --image=nginx --dry-run=client -o yaml   
adjusted according to my requirements:

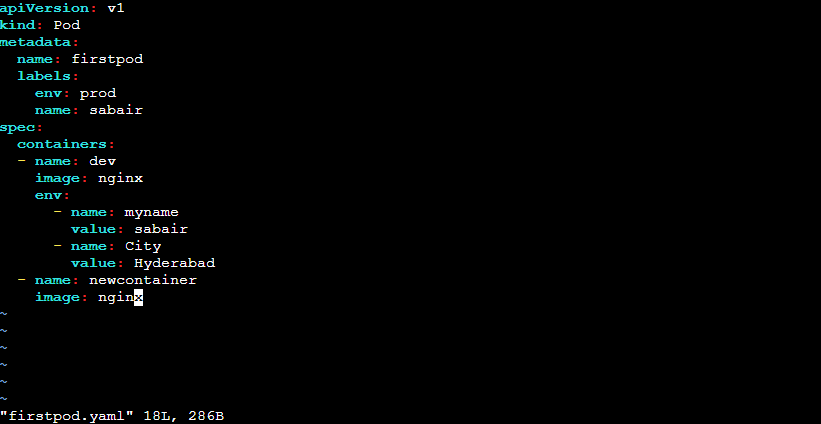
  
  
Successfully created the new pod:  
  
deleted the pod: $kubectl delete pod nginxpod --grace-period=0 --force

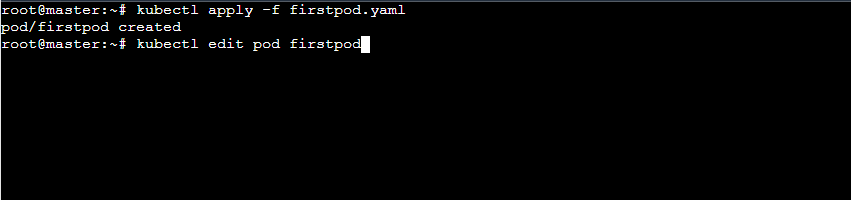
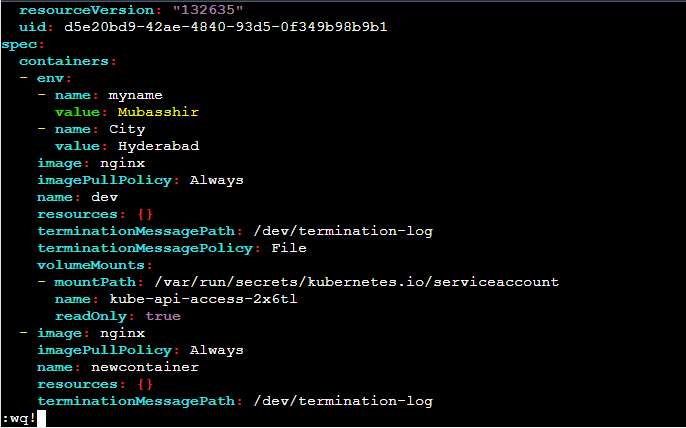
7) Use kubectl apply vs kubectl create

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.  
  
  
Creating container with create command:  
  
change the pod name in the file:  
  
Now I have tried the create command again and it was giving output like this:  
  
Then tried with $ kubectl apply -f firstpod.yaml :  
  


8) Edit an Existing Pod Configuration

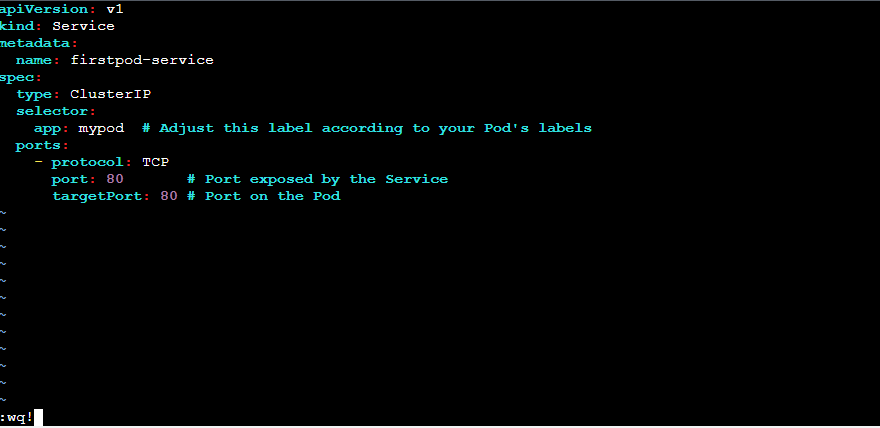
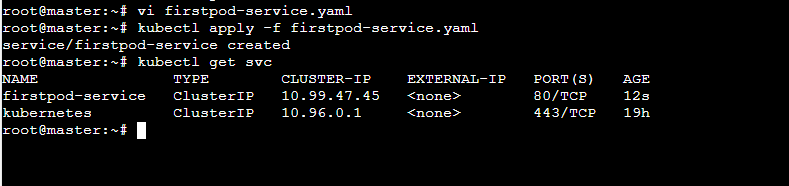
Task: Use kubectl edit pod <pod\_name> to modify the running Pod's environment variables or image. After making the changes, verify if they took effect by checking the container logs or environment variables using kubectl exec.

  
With the help of edit command changing the variable:

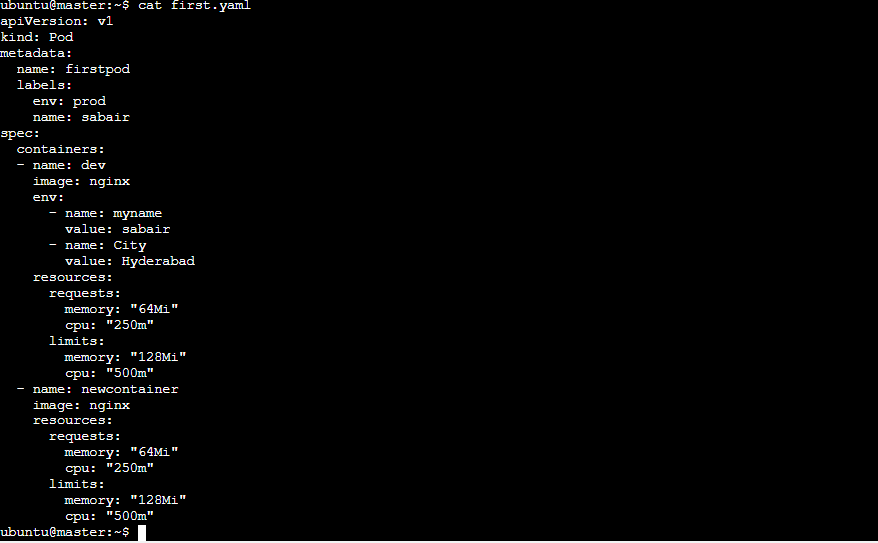
  
edited:  


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.

  
Apply the YAML to Create the Service: Run the following command to create the Service:  
$ kubectl apply -f firstpod-service.yaml  
Verify the Service: After creating the Service, you can verify it by running:  
$ 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.  
  
Checked the specifications:  
$kubectl describe pod firstpod  
