K8s-02

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 pod.yaml and add the following template # vi pod.yaml

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

kubectl apply -f pod.yaml
kubectl get pods
kubectl logs firstpod

```
root@master:~# kubectl apply -f pod.yaml
pod/firstpod unchanged
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          42s
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
2024/12/31 09:35:14 [notice] 1#1: using the "epoll" event method
2024/12/31 09:35:14 [notice] 1#1: nginx/1.27.3
2024/12/31 09:35:14 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/12/31 09:35:14 [notice] 1#1: OS: Linux 6.8.0-1018-aws
2024/12/31 09:35:14 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/12/31 09:35:14 [notice] 1#1: start worker processes
2024/12/31 09:35:14 [notice] 1#1: start worker process 29
2024/12/31 09:35:14 [notice] 1#1: start worker process 30
```

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.

→ EDIT A FILE pod.yaml which we created in "1)" and add the following in that template # vi pod.yaml

```
    env:
    - name: myname
      value: sabair
    - name: City
      value: Hyderabad
```

kubectl delete firstpod

kubectl apply -f pod.yaml
kubectl get pods
kubectl exec firstpod – env

```
root@master:~# kubectl delete pod firstpod
pod "firstpod" deleted
root@master:~# kubectl apply -f pod.yaml
pod/firstpod created
root@master:~# kubectl exec firstpod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=firstpod
NGINX_VERSION=1.27.3
NJS_VERSION=0.8.7
NJS_RELEASE=1~bookworm
PKG_RELEASE=1~bookworm
DYNPKG_RELEASE=1~bookworm
City=Hyderabad
myname=sabair
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
HOME=/root
```

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.

→ EDIT A FILE pod.yaml which we created in “2)” and add the following in that template # vi pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: secondpod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    command: ["sleep"]
    args: ["50"]
    env:
    - name: myname
      value: sabair
    - name: City
      value: Hyderabad
    ports:
    - containerPort: 80
```

kubectl apply -f pod.yaml
kubectl get pods
kubectl describe secondpod

```
Containers:
  nginx:
    Container ID:  containerd://a9fb7a8949a669c9d09ac9b663a431bae549649a2e9836b62543c
04d8e4d8a40
    Image:         nginx:latest
    Image ID:      docker.io/library/nginx@sha256:42e917aaa1b5bb40dd0f6f7f4f857490ac7
747d7ef73b391c774a41a8b994f15
    Port:          80/TCP
    Host Port:     0/TCP
    Command:
      sleep
    Args:
      50
    State:          Running
      Started:      Tue, 31 Dec 2024 09:50:19 +0000
    Ready:          True
    Restart Count:  0
```

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 FILE twocontainerspod.yaml and add the following template

```
apiVersion: v1
kind: Pod
metadata:
  name: twocontainerspod
spec:
  containers:
  - name: nginx-container-1
    image: nginx:latest
    ports:
    - containerPort: 80
  - name: httpd-container-2
    image: httpd
    ports:
    - containerPort: 80
```

kubectl apply -f twocontainerspod.yaml

kubectlget pods

```
root@master:~# kubectl apply -f twocontainerspod.yaml
Warning: spec.containers[1].ports[0]: duplicate port definition with spec.containers[0].ports[0]
pod/twocontainerspod created
root@master:~# kubectl get pods
NAME               READY   STATUS    RESTARTS       AGE
secondpod          1/1     Running   5 (104s ago)   7m21s
twocontainerspod   1/2     Error     2 (22s ago)    35s
```

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 A FILE initpod.yaml AND ADD THE BELOW TEMPLATE

```
apiVersion: v1
kind: Pod
metadata:
  name: initpod
spec:
  initContainers:
  - name: init-sleep
    image: busybox
    command: ["sh", "-c", "sleep 30"]
  containers:
  - name: nginx-container
    image: nginx:latest
```

kubectl apply -f initpod.yaml

kubectl get pods

kubectl describe pod initpod

```
Init Containers:
  init-sleep:
    Container ID:  containerd://7929349f575ef1d47cf474a09c83cef190d398e8646d5498e8d7517a92630610
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:2919d0172f7524b2d8df9e50066a682669e6d170ac0f6a49676d54358fe970b5
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 30
    State:          Terminated
      Reason:       Completed
```

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 run nginx --image=nginx --dry-run=client -o yaml > nginx-pod.yaml

vi nginx-pod.yaml

→ADD THE ENVIRONMENT

```
env:
   - name: myname
     value: farsaan
   - name: City
     value: Hyderabad
```

kubectl apply -f naginx-pod.yaml

kubectl get pods

```
root@master:~# kubectl run nginx --image=nginx --dry-run=client -o yaml > nginx-pod.yaml
root@master:~# ls
1  custom-resources.yaml  initpod.yaml  nginx-pod.yaml  pod.yaml  snap  twocontainerspod.yaml
root@master:~# vi nginx-pod.yaml
root@master:~# kubectl apply -f nginx-pod.yaml
pod/nginx created
root@master:~# kubectl get pod -o wide
NAME              READY   STATUS             RESTARTS        AGE    IP              NODE        NOMINATED NODE   READINESS GATES
initpod           1/1     Running            0               8m2s   10.10.37.198    worker-02   <none>           <none>
nginx             1/1     Running            0               10s    10.10.37.199    worker-02   <none>           <none>
secondpod         0/1     CrashLoopBackOff   7 (2m17s ago)   19m    10.10.37.197    worker-02   <none>           <none>
twocontainerspod  1/2     CrashLoopBackOff   7 (101s ago)    13m    10.10.171.3     worker-01   <none>           <none>
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.

→ CREATE A YAML FILE mypod.yaml AND ADD THE FOLLOWING TEMPLATE

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

kubectl create -f mypod.yaml

kubectl get pods

```
root@master:~# kubectl create -f mypod.yaml
pod/mypod created
root@master:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
mypod   1/1     Running   0          3s
```

→ MODIFY THE FILE AND ADD nginx:1.21.6 IN THE IMAGE SECTION

→REDEPLOY THE POD

kubectl apply -f mypod.yaml

kubectl get pods -o wide

```
root@master:~# vi mypod.yaml
root@master:~# kubectl apply -f mypod.yaml
Warning: resource pods/mypod is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl
d on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
pod/mypod configured
root@master:~# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS      AGE     IP              NODE        NOMINATED NODE   READINESS GATES
mypod   1/1     Running   1 (13s ago)   3m38s   10.10.37.200    worker-02   <none>           <none>
```

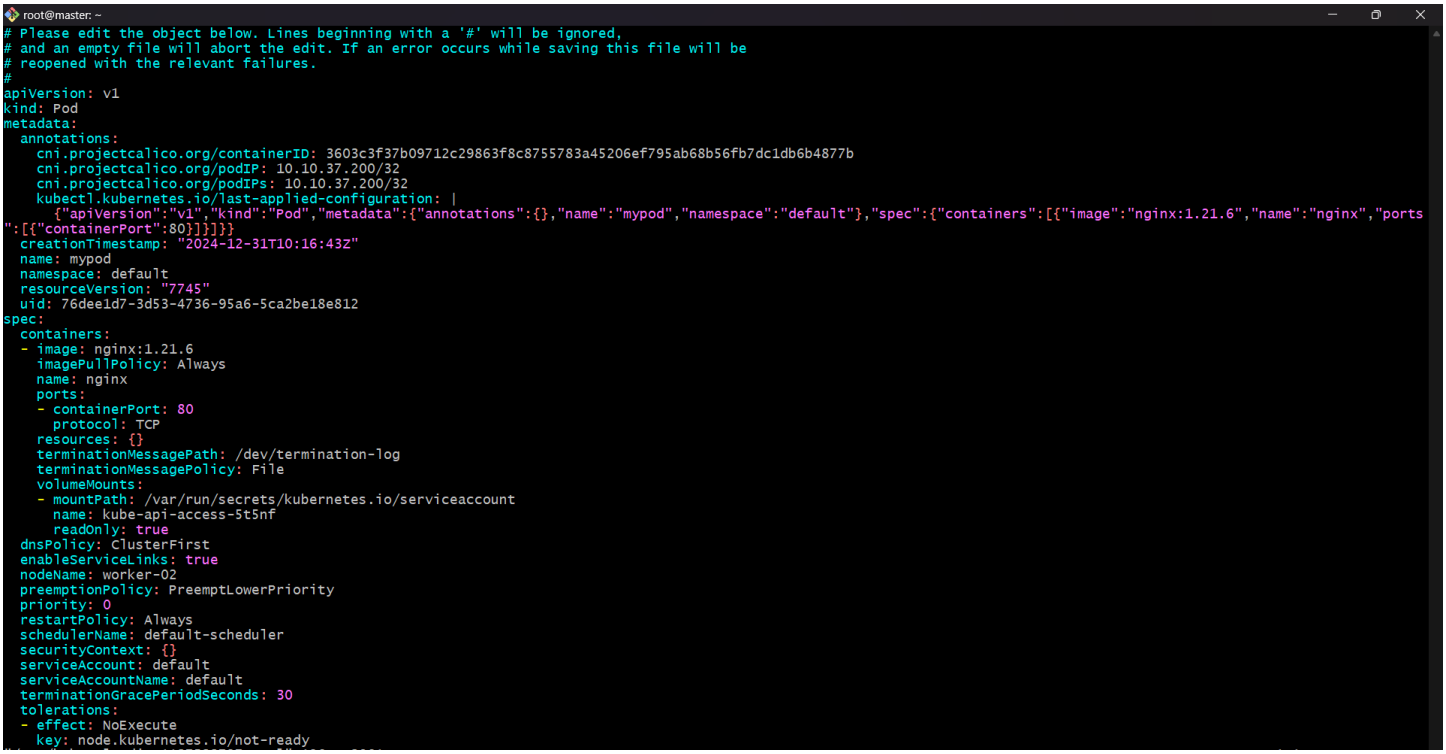
8) Edit an Existing Pod Configuration

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 mypod

→THIS WILL OPEN THE YAML EDITE YOUR CHANGES AND SAVE

→ KUBERNETES WILL APPLY THE UPDATES TO THE RUNNING POD.



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 YAML FILE service.yaml AND ADD THE FOLLOWING TEMPLATE

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

kubectl apply -f service.yaml

kubectl get svc

```
root@master:~# vi service.yaml
root@master:~# kubectl apply -f service.yaml
service/firstpod-service created
root@master:~# kubectl get svc
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
firstpod-service    ClusterIP   10.100.104.136   <none>        80/TCP    9s
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP   59m
```

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.

→CREATE A YAML FILE resourcepod.yaml AND ADD THE FOLLOWING TEMPLATE

```
APIVERSION: V1
KIND: POD
METADATA:
 NAME: RESOURCEPOD
SPEC:
 CONTAINERS:
 - NAME: NGINX-CONTAINER
  IMAGE: NGINX:LATEST
  PORTS:
  - CONTAINERPORT: 80
  RESOURCES:
   REQUESTS:
    MEMORY: "64MI"
    CPU: "250M"
   LIMITS:
    MEMORY: "128MI"
    CPU: "500M"
```

kubectl apply -f resourcepod.yaml
kubectl get pods
kubectl describe pod resourcepod