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.

→ YAML file to create a Pod named firstpod with an nginx container:

apiVersion: v1

kind: Pod

metadata:

  name: firstpod

spec:

  containers:

  - name: nginx-container

    image: nginx

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
spec:
  containers:
  - name: nginx-container
    image: nginx
```

→created pod with this command

kubectl create -f firstpod.yaml

```
root@master:~# kubectl create -f firstpod.yaml
pod/firstpod created
```

→ Verified the Pod creation using kubectl get pods

```
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
firstpod    0/1     Pending   0          4m34s
```

→ **checked the logs of the container using 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
2025/04/14 11:56:31 [notice] 1#1: using the "epoll" event method
2025/04/14 11:56:31 [notice] 1#1: nginx/1.27.4
2025/04/14 11:56:31 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/04/14 11:56:31 [notice] 1#1: OS: Linux 6.8.0-1026-aws
2025/04/14 11:56:31 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/14 11:56:31 [notice] 1#1: start worker processes
2025/04/14 11:56:31 [notice] 1#1: start worker process 29
2025/04/14 11:56:31 [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.

→ **the YAML file to include environment variables myname: narendar and City: Hyderabad**

```
  apiVersion: v1
kind: Pod
metadata:
 name: env-demo-pod
spec:
 containers:
 - name: env-demo-container
   image: busybox
   command: ["sh", "-c", "sleep 3600"] # Keeps the pod running
   env:
   - name: myname
     value: "narendar"
```

- name: City

  value: "Hyderabad"

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: env-demo-pod
spec:
  containers:
  - name: env-demo-container
    image: busybox
    command: ["sh", "-c", "sleep 3600"] # Keeps the pod running
    env:
    - name: myname
      value: "narendar"
    - name: City
      value: "Hyderabad"
```

→run yaml file then pod is created:

```
root@master:~# kubectl apply -f env-demo-pod.yaml
The Pod "env-demo-pod" is invalid: spec: Forbidden: pod updates may not change fields other than `spec.contai
ec.activeDeadlineSeconds`,`spec.tolerations` (only additions to existing tolerations),`spec.terminationGraceP
previously negative)
  core.PodSpec{
        Volumes:        {{Name: "kube-api-access-ws785", VolumeSource: {Projected: &{Sources: {{ServiceAccoun
}, {ConfigMap: &{LocalObjectReference: {Name: "kube-root-ca.crt"}, Items: {{Key: "ca.crt", Path: "ca.crt"}}}}
ldRef: &{APIVersion: "v1", FieldPath: "metadata.namespace"}}}}}, DefaultMode: &420}}}},
        InitContainers: nil,
        Containers: []core.Container{
                {
                        Name:  "env-demo-container",
                        Image: "busybox",
                        Command: []string{
-                               "sh",
+                               "sleep",
-                               "-c",
-                               "sleep 3600",
                        }
```

```
NAME           READY   STATUS      RESTARTS        AGE
busybox        0/1     Completed   0               70m
env-demo-pod   1/1     Running     1 (10m ago)     74m
```

→ **checked if the environment variables are set properly by exec command:**

```
root@master:~# kubectl exec -it env-demo-pod -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=env-demo-pod
myname=narendar
City=Hyderabad
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
KUBERNETES_SERVICE_PORT=443
TERM=xterm
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.

→ **YAML file to add args that instruct the container to sleep for 50 seconds:**

```
apiVersion: v1

kind: Pod

metadata:

  name: sleep-pod

spec:

  containers:

  - name: sleep-container

    image: busybox

    command: ["sleep"]

    args: ["50"]
```

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: sleep-pod
spec:
  containers:
  - name: sleep-container
    image: busybox
    command: ["sleep"]
    args: ["50"]
```

→Deploy the Pod:

```
root@master:~# vi sleep-pod.yaml
root@master:~# kubectl apply -f sleep-pod.yaml
pod/sleep-pod created
root@master:~# kubectl get pods
NAME           READY   STATUS    RESTARTS      AGE
busybox        1/1     Running   0             10m
env-demo-pod   1/1     Running   0             14m
sleep-pod      1/1     Running   1 (13s ago)   64s
```

→ verified the args are correctly passed to the container by using below command:

kubectl describe pod sleep-pod

```
root@master:~# kubectl describe pod sleep-pod
Name:             sleep-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-13-158/172.31.13.158
Start Time:       Mon, 14 Apr 2025 10:59:56 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: ffb618df4237431f4f6b68e91f826ece65cf022ba404fadd2890de5
                  cni.projectcalico.org/podIP: 192.168.201.195/32
                  cni.projectcalico.org/podIPs: 192.168.201.195/32
Status:           Running
IP:               192.168.201.195
IPs:
  IP:  192.168.201.195
Containers:
  sleep-container:
    Container ID:  containerd://934e47de605b78b7ea2e9ca86cb844aa242b837ef76f69013d3bf5458e3a852d
```

```
Status:           Running
IP:               192.168.201.195
IPs:
  IP:  192.168.201.195
Containers:
  sleep-container:
    Container ID:  containerd://934e47de605b78b7ea2e9ca86cb844aa242b837ef76f69013d3bf5458
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:37f7b378a29ceb4c551b1b5582e27747b855b
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
    Args:
      50
    State:          Running
      Started:      Mon, 14 Apr 2025 11:00:47 +0000
    Last State:     Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Mon, 14 Apr 2025 10:59:57 +0000
      Finished:     Mon, 14 Apr 2025 11:00:47 +0000
```

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).

→ YAML file to define a Pod with two nginx containers inside:

```
apiVersion: v1

kind: Pod

metadata:

  name: multi-nginx-pod

spec:

  containers:

  - name: nginx1
```

image: nginx

- name: nginx2

image: nginx

- name: busybox

image: busybox

command: ["sh", "-c", "sleep 3600"]

→then apply these two commands:

kubectl delete pod multi-nginx-pod --ignore-not-found

kubectl apply -f multi-nginx-pod.yaml

→check pods:

```
root@master:~# kubectl get pods
NAME              READY   STATUS    RESTARTS      AGE
demo-pod          1/1     Running   1 (65m ago)   82m
firstpod          1/1     Running   0             40m
multi-nginx-pod   2/3     Error     4 (49s ago)   99s
nginx             1/1     Running   0             117m
resource-pod      1/1     Running   0             25m
```

→ Test Internal Communication with Telnet:

kubectl exec -it multi-nginx-pod -c busybox – sh

--give this inside the shell:

telnet localhost 80

--we should see: connected to localhost

--give GET/

```
root@master:~# kubectl exec -it multi-nginx-pod -c busybox -- sh
/ #
/ # telnet localhost 80
Connected to localhost
GET /
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
```

→ **get an NGINX welcome HTML:**

```
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web
working. Further configuration is requ
```

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.

→ **YAML to include an init container that sleeps for 30 seconds before the main containers start:**

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  initContainers:
  - name: sleep-init
    image: busybox
    command: ['sleep', '30']
  containers:
  - name: main-container
    image: nginx
    ports:
    - containerPort: 80
```

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  initContainers:
  - name: sleep-init
    image: busybox
    command: ['sleep', '30']
  containers:
  - name: main-container
    image: nginx
    ports:
    - containerPort: 80
```

→ **Deploy the Pod:**

kubectl apply -f init.yaml

```
root@master:~# kubectl apply -f init.yaml
pod/example-pod created
root@master:~# kubectl describe pod example-pod
Name:             example-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-13-158/172.31.13.158
```

→ **Verified the init container's execution using kubectl describe pod:**

kubectl describe pod example-pod

```
root@master:~# kubectl describe pod example-pod
Name:             example-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-13-158/172.31.13.158
Start Time:       Mon, 14 Apr 2025 12:16:18 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: e9b67c05abc0592ac2689cebd37deea9fb6bc6e1036475bbf1e258ccbf87a
                  cni.projectcalico.org/podIP: 192.168.201.196/32
                  cni.projectcalico.org/podIPs: 192.168.201.196/32
Status:           Running
IP:               192.168.201.196
IPs:
  IP:  192.168.201.196
```

```
Status:           Running
IP:               192.168.201.196
IPs:
  IP:  192.168.201.196
Init Containers:
  sleep-init:
    Container ID:  containerd://3c74ccee06e3c9e27769a3010ef9fdefe7921e599a93f28990bf43b85563e46c
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11
    Port:          <none>
    Host Port:     <none>
    Command:
      sleep
      30
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Mon, 14 Apr 2025 12:16:19 +0000
      Finished:     Mon, 14 Apr 2025 12:16:49 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
```

→ **check the logs to confirm its completion:**

```
root@master:~# kubectl logs example-pod
Defaulted container "main-container" out of: main-container, sleep-init (init)
/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
2025/04/14 12:16:50 [notice] 1#1: using the "epoll" event method
2025/04/14 12:16:50 [notice] 1#1: nginx/1.27.4
2025/04/14 12:16:50 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/04/14 12:16:50 [notice] 1#1: OS: Linux 6.8.0-1026-aws
2025/04/14 12:16:50 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/14 12:16:50 [notice] 1#1: start worker processes
2025/04/14 12:16:50 [notice] 1#1: start worker process 30
2025/04/14 12:16:50 [notice] 1#1: start worker process 31
```

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.

→ Used the kubectl run nginx --image=nginx --dry-run=client -o yaml command to generate a Pod YAML definition:

kubectl run nginx --image=nginx --dry-run=client -o yaml

--generated a pod YAML definition:

```
root@master:~# kubectl run nginx --image=nginx --dry-run=client -o yaml > nginx-pod.yaml
root@master:~# kubectl get pods
No resources found in default namespace.
root@master:~# cat nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

→ Modifyied the generated YAML to suit specific requirements (e.g., labels or environment variables):

apiVersion: v1

kind: Pod

metadata:

  name: nginx

  labels:

    app: web

    tier: frontend

spec:

  containers:

  - name: nginx

    image: nginx

env:

- name: ENVIRONMENT

  value: production

ports:

- containerPort: 80

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: web
    tier: frontend
spec:
  containers:
  - name: nginx
    image: nginx
    env:
    - name: ENVIRONMENT
      value: production
    ports:
    - containerPort: 80

~
~
```

→ Deploy the Pod:

kubectl apply -f nginx-pod.yaml

```
root@master:~# kubectl apply -f nginx-pod.yaml
pod/nginx created
root@master:~# vi nginx-pod.yaml
root@master:~# kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
nginx    1/1     Running   0          4m50s
```

→ Inspect the pod in detail for labels and environment variables:

```
root@master:~# kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-4-112/172.31.4.112
Start Time:       Mon, 14 Apr 2025 12:46:13 +0000
Labels:           app=web
                  tier=frontend
Annotations:      cni.projectcalico.org/containerID: 2112b188c330fcbc1
```

```
    State:          Running
      Started:      Mon, 14 Apr 2025 12:46:13 +0000
    Ready:          True
    Restart Count:  0
    Environment:
      ENVIRONMENT:  production
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dgd2b (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
```

```
root@master:~# kubectl exec -it nginx -- printenv ENVIRONMENT
production
```

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 simple Pod definition file called my-pod.yml:

```
apiVersion: v1

kind: Pod

metadata:

  name: demo-pod

spec:

  containers:

  - name: nginx

    image: nginx:1.21

    ports:

    - containerPort: 80
```

→**Deploy Using kubectl create:**

--Run the following command to deploy the Pod:

kubectl create -f my-pod.yaml

```
root@master:~# kubectl create -f my-pod.yaml
pod/demo-pod created
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
demo-pod    1/1     Running   0          18s
nginx       1/1     Running   0          18m
```

→**Modify the YAML:**

--Now, change the image version in my-pod.yml from nginx:1.21 to nginx:1.25

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: demo-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.25
    ports:
    - containerPort: 80

~
```

→ **Redeploy Using kubectl apply:**

kubectl apply -f my-pod.yml

❗ **However, this will not update an existing pod because Pods are not designed to be updated in-place (they are immutable once created). You'll see a message like:**

```
root@master:~# kubectl apply -f my-pod.yaml
Warning: resource pods/demo-pod is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply
hould only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automa
cally.
pod/demo-pod configured
```

→ **Since Pods are immutable, you need to delete the old one first:**

--then use apply command

```
root@master:~# kubectl delete pod demo-pod
pod "demo-pod" deleted
root@master:~# ^C
root@master:~# kubectl apply -f my-pod.yaml
pod/demo-pod created
root@master:~# kubectl describe pod demo-pod
Name:             demo-pod
Namespace:        default
Priority:         0
Service Account:  default
```

→**we should see the updated image version (nginx:1.25):**

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  26s   default-scheduler  Successfully assigned default/demo-pod to ip-172-31-13-158
  Normal  Pulled     26s   kubelet            Container image "nginx:1.25" already present on machine
  Normal  Created    26s   kubelet            Created container: nginx
  Normal  Started    26s   kubelet            Started container nginx
```

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.

→ **List the Running Pods:**

```
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
demo-pod   1/1     Running   0          10m
nginx      1/1     Running   0          45m
```

→ **Edit the Running Pod Configuration:**

--modified image: from nginx:1.25 to nginx:1.21

By using command

kubectl edit pod demo-pod

```
root@master: ~
# 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:
    cni.projectcalico.org/containerID: afbeaac5ab792038e9dc98f2c0953231cc68ea26ef495a2b6f971
    cni.projectcalico.org/podIP: 192.168.201.199/32
    cni.projectcalico.org/podIPs: 192.168.201.199/32
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"demo-pod","namespa
:"nginx","ports":[{"containerPort":80}]}]}}
  creationTimestamp: "2025-04-14T13:20:51Z"
  name: demo-pod
  namespace: default
  resourceVersion: "24413"
  uid: 38de8f51-7552-4781-8227-f9e952df402f
spec:
  containers:
  - image: nginx:1.25
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
      protocol: TCP
```

```
:"nginx","ports":[{"containerPort":80}]}]}}
  creationTimestamp: "2025-04-14T13:20:51Z"
  name: demo-pod
  namespace: default
  resourceVersion: "24413"
  uid: 38de8f51-7552-4781-8227-f9e952df402f
spec:
  containers:
  - image: nginx:1.21
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
```

→**chevk the pod status:**

```
root@master:~# kubectl get pod demo-pod
NAME        READY   STATUS    RESTARTS        AGE
demo-pod    1/1     Running   1 (2m18s ago)   19m
```

→ verify the image version by using below command:

```
root@master:~# kubectl describe pod demo-pod
Name:             demo-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             ip-172-31-13-158/172.31.13.158
Start Time:       Mon, 14 Apr 2025 13:20:51 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: afbeaac5ab792038e9dc98f2c0953231cc68ea26ef495a2b6f97155803
```

--image version changed

```
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age                From               Message
  ----    ------     ----               ----               -------
  Normal  Scheduled  20m                default-scheduler  Successfully assigned default/demo-pod to ip-172-31-13-158
  Normal  Pulled     20m                kubelet            Container image "nginx:1.25" already present on machine
  Normal  Created    3m1s (x2 over 20m) kubelet            Created container: nginx
  Normal  Started    3m1s (x2 over 20m) kubelet            Started container nginx
  Normal  Killing    3m1s               kubelet            Container nginx definition changed, will be restarted
  Normal  Pulled     3m1s               kubelet            Container image "nginx:1.21" already present on machine
root@master:~#
```

→Checked th logs:

```
root@master:~# kubectl logs demo-pod
/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: 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
2025/04/14 13:37:55 [notice] 1#1: using the "epoll" event method
2025/04/14 13:37:55 [notice] 1#1: nginx/1.21.6
2025/04/14 13:37:55 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2025/04/14 13:37:55 [notice] 1#1: OS: Linux 6.8.0-1026-aws
2025/04/14 13:37:55 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/04/14 13:37:55 [notice] 1#1: start worker processes
2025/04/14 13:37:55 [notice] 1#1: start worker process 31
2025/04/14 13:37:55 [notice] 1#1: start worker process 32
```

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 the Pod firstpod.yaml:

 # firstpod.yaml file:

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

→deploy it using below command:

kubectl apply -f firstpod.yaml

```
root@master:~# kubectl apply -f firstpod.yaml
pod/firstpod created
```

```
root@master:~# kubectl get pods
NAME        READY   STATUS    RESTARTS      AGE
demo-pod    1/1     Running   1 (28m ago)   45m
firstpod    1/1     Running   0             2m57s
nginx       1/1     Running   0             79m
```

→ **Create a Service YAML to Expose the Pod:**

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

→ **Deploy the Service:**

kubectl apply -f firstpod-service.yaml

```
root@master:~# vi firstpod-service.yaml
root@master:~# kubectl apply -f firstpod-service.yaml
service/firstpod-service created
```

→**verified the service by using below command:**

kubectl get svc

```
root@master:~# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
firstpod-service    ClusterIP   10.111.36.119   <none>        80/TCP    18s
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP   4h12m
root@master:~#
```

-- we can also describe the service for more detail:

kubectl describe svc firstpod-service

```
root@master:~# kubectl describe svc firstpod-service
Name:                     firstpod-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=myapp
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.111.36.119
IPs:                      10.111.36.119
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
Endpoints:                192.168.201.200:80
Session Affinity:         None
Events:                   <none>
root@master:~#
```

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.

→ Pod YAML with Resource Requests and Limits:

 # resource-pod.yaml file

```
apiVersion: v1

kind: Pod

metadata:

  name: resource-pod

spec:

  containers:

  - name: nginx
```

```
  image: nginx
  ports:
  - containerPort: 80
  resources:
    requests:
      memory: "64Mi"
      cpu: "250m"
    limits:
      memory: "128Mi"
      cpu: "500m"
```

→ **Deploy the Pod using below command:**

kubectl apply -f resource-pod.yaml

--pod is created

```
root@master:~# kubectl apply -f resource-pod.yaml
pod/resource-pod created
```

→ **Verify the Resource Configuration using below command:**

kubectl describe pod resource-pod

--Look for the Containers section. You should see output like this:

```
Containers:
  nginx:
    Container ID:   containerd://e6cd693c862a0b7628f82a8c61aa4378f0955382bf
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:09369da6b10306312cd90866
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Mon, 14 Apr 2025 14:17:36 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  128Mi
    Requests:
      cpu:        250m
      memory:     64Mi
```