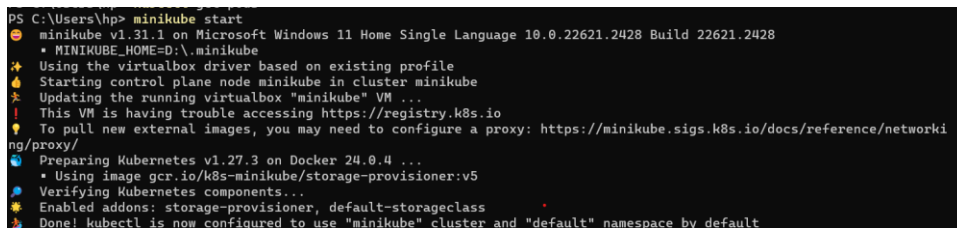
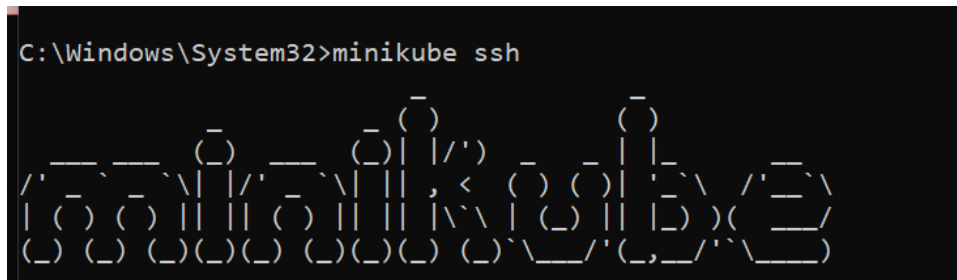


Lab Exercise 7– Creating Pods in Kubernetes

Below is a lab exercise that will help you understand and practice creating pods in Kubernetes:

Task 1: Start Kubernetes in Docker-Desktop

- Start Kubernetes service in Docker-Desktop



Task 2: Creating a Simple Pod

- Create a simple YAML manifest file named pod.yaml to define a basic Pod in Kubernetes. An example of the file content is as follows:

```
apiVersion: v1

kind: Pod

metadata:

  name: my-pod

spec:

  containers:

  - name: my-container

    image: nginx
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

hp@t4r MINGW64 ~/Desktop/ACO-LAB-2021-25-SUBMISSION/R2142211343/Exp7 (main)
$ touch pod.yaml
```

- Apply the Pod configuration using the following command:

```
kubectl apply -f pod.yaml
```

```
PS C:\Users\hp\Desktop\ACO-LAB-2021-25-SUBMISSION\R2142211343\Exp7> kubectl apply -f .\pod.yaml
pod/my-pod created
```

Check the status of the Pod using the following command:

```
kubectl get pods
```

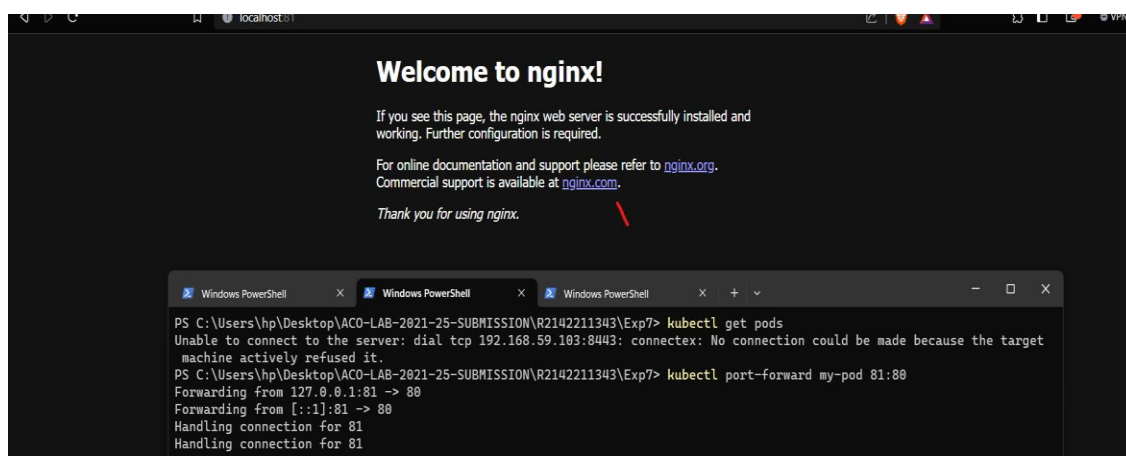
```
PS C:\Users\hp\Desktop\ACO-LAB-2021-25-SUBMISSION\R2142211343\Exp7> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
my-pod    0/1     Pending   0           0s
mydb      0/1     CrashLoopBackOff   22 (4m21s ago)    84d
```

Task 3: Accessing the Pod

Access the Pod by using port forwarding to the container. Run the following command:

```
kubectl port-forward my-pod 81:80
```

Access the Nginx server running in the Pod by opening a web browser and navigating to <http://localhost:81>.



Task 4: Exploring Pod Details

Retrieve detailed information about the Pod using the following command:

```
kubectl describe pod my-pod
```

```
PS C:\Users\hnp> kubectl describe pod my-pod
Name:          my-pod
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.59.103
Start Time:    Fri, 20 Oct 2023 21:59:20 +0530
Labels:        <none>
Annotations:   <none>
Status:        Running
IP:            10.244.0.19
IPs:
  IP: 10.244.0.19
Containers:
  my-container:
    Container ID:  docker://ccb7cc4f2a129644059ef56125daf48366523089beabd64c6bc213c0cfca09a4
    Image:         nginx
    Image ID:      docker-pullable://nginx@sha256:b4af4f8b6470febf45dc10f564551af682a802eda1743055a7dfc8332dffa595
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Fri, 20 Oct 2023 22:09:28 +0530
    Last State:    Terminated
      Reason:      Completed
      Exit Code:   0
      Started:     Fri, 20 Oct 2023 22:05:44 +0530
      Finished:    Fri, 20 Oct 2023 22:09:12 +0530
    Ready:         True
    Restart Count: 3
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-rxzkh (ro)
Conditions:
  Type              Status
  Initialized        True
  Ready              True
  ContainersReady    True
  PodScheduled       True
Volumes:
```

Check the logs of the Pod to understand its behavior using the following command:

```
kubectl logs my-pod
```

```
PS C:\Users\hnp> kubectl logs my-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: 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
2023/10/20 16:46:52 [notice] 1#1: using the "epoll" event method
2023/10/20 16:46:52 [notice] 1#1: nginx/1.25.2
2023/10/20 16:46:52 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/10/20 16:46:52 [notice] 1#1: OS: Linux 5.10.57
2023/10/20 16:46:52 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/10/20 16:46:52 [notice] 1#1: start worker processes
2023/10/20 16:46:52 [notice] 1#1: start worker process 28
2023/10/20 16:46:52 [notice] 1#1: start worker process 29
```

Task 5: Deleting the Pod

Delete the Pod using the following command:

```
kubectl delete pod my-pod
```

```
PS C:\Users\hnp> kubectl delete pod my-pod
pod "my-pod" deleted
PS C:\Users\hnp>
```

Verify that the Pod has been deleted by running the `kubectl get pods` command.

Task 6: Advanced Pod Configuration

- Experiment with advanced Pod configuration options such as environment variables, volume mounts, resource limits, and labels.
- Update the Pod manifest file and apply the changes to the Kubernetes cluster.

```
1  apiVersion: v1
2
3  kind: Pod
4
5  metadata:
6    |
7    name: my-second-pod
8
9  spec:
10   containers:
11     - name: my-container2
12       image: httpd
13       ports:
14         - containerPort: 30001
15         |
16         protocol: TCP
```

Task 7: Cleanup

Delete any remaining Pods, services, and deployments created during the exercise using the appropriate kubectl delete commands.

Task 8: Documentation and Best Practices

Document your findings and the best practices for creating and managing Pods in Kubernetes.

Through this exercise, you'll gain a better understanding of how to create, manage, and interact with Pods in Kubernetes. Adjust the exercise based on your specific use case and requirements.

We will try to assign a port in a new yaml file and try to find out how a port can be assigned through yaml file

```
1  apiVersion: v1
2
3  kind: Pod
4
5  metadata:
6    |
7    name: my-second-pod
8
9  spec:
10   containers:
11     - name: my-container2
12       image: httpd
13       ports:
14         - containerPort: 30001
15         |
16         protocol: TCP
```

After creating a new pod with image of apache web server. We will try to find out the port we assigned using `kubectl describe`

```
PS C:\Users\hp\Desktop\ACO-LAB-2021-25-SUBMISSION\R2142211343\Exp7> kubectl apply -f .\pod2.yml
pod/my-second-pod created
```

```

PS C:\Users\hp\Desktop\ACU-LAB-2021-25-SUBMISSION\R2142211343\Exp/> kubectl describe pod my-second-pod
Name:          my-second-pod
Namespace:     default
Priority:      0
Service Account: default
Node:         minikube/192.168.59.103
Start Time:    Fri, 20 Oct 2023 22:28:58 +0530
Labels:        <none>
Annotations:   <none>
Status:        Running
IP:           10.244.0.27
IPs:
  IP: 10.244.0.27
Containers:
  my-container2:
    Container ID:  docker://088d4b75d461aa7e8b0e4ccd6fc47214b281e2d6e01f6d5ecadf454eec826cfd
    Image:         httpd
    Image ID:      docker-pullable://httpd@sha256:ed6db4a8c394d075c9c59a3dbd61a3818cd302d9948057f1e19046e5bffe027
    Port:         30001/TCP
    Host Port:    0/TCP
    State:        Running
      Started:    Fri, 20 Oct 2023 22:29:20 +0530
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-wcv6m (ro)

```

Here we can see that the port number that we assigned in the Yaml file was 30001 and after using kubectl describe we can see that when the pod is being used its listening in nport number 30001