q11) Create a kubernetes cluster with docker as a hypervisor. List master processes running inside minikube Find how many nodes are there in the cluster and role(s) of each node.

1. minikube start --driver=docker *//start the cluster with docker as hyperviser*
2. minikube status *//check status of minikube*
3. kubectl get pods --namespace=kube-system -l component=kube-apiserver *//list of master processes*
4. kubectl get nodes -o wide **or** kubectl get nodes

q12) Create a deployment with name = “staging” and image as “nginx” Get name of pods running in a kubernetes cluster Get kubernetes services and cluster IP. Demonstrate pods with replicaset value of 2

1. To create a deployment named "staging" with the "nginx" image, you can use the following command: **kubectl create deployment staging --image=nginx**
2. To get the names of all the pods running in the cluster, you can use the following command: **kubectl get pods**
3. To get a list of all the services running in the cluster and their cluster IPs, you can use the following command: **kubectl get services -o wide**
4. To demonstrate pods with a replicaset value of 2, you can scale up the deployment with the following command: : **kubectl scale deployment staging --replicas=2**

Viva:

1. A pod is the smallest execution unit in Kubernetes. A pod encapsulates one or more applications. Pods are ephemeral by nature, if a pod (or the node it executes on) fails, Kubernetes can automatically create a new replica of that pod to continue operations.
2. Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache.

q13) Create a kubernetes cluster with hyperv as hypervisor. Get the status of the kubernetes service that controls. pod in a kubernetes cluster.. Access dashboard in minikube. Stop kubernetes cluster

1. minikube start --driver=hyperv
2. minikube status
3. systemctl status kubelet.service
4. minikube dashboard
5. minikube stop

q14) Steps necessary to get jenkins docker image and how to start jenkins . Demonstrate use of a Jenkins project to build a simple Go lang source code with source code to build in Git Hub repo

1. Pull the Jenkins Docker image from Docker Hub using the following command: **docker pull jenkins/jenkins**
2. docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins
3. Open the Jenkins UI in your web browser and log in with your credentials.
4. Click on "New Item" and provide a name for the project. Select "Freestyle project" and click "OK".
5. In the project configuration page, scroll down to the "Source Code Management" section and select "Git".
6. In the "Repository URL" field, enter the URL of the Git repository where the Go lang source code is hosted.
7. In the "Build" section, click on "Add build step" and select "Execute shell".
8. go build main.go

q15) Create a kubernetes cluster with virtualbox as hypervisor. List master processes running inside minikube Find how many nodes are there in the cluster and the roles of each node. Delete kubernetes cluster

Once you have the VM installed, you can launch Minikube to create the Kubernetes cluster. To do this, open the command prompt and execute the “minikube start” command. This will create the cluster and set up the environment for you.

1. kubectl get pods
2. kubectl get nodes
3. minikube delete

q16) Create a kubernetes cluster with virtualbox as hypervisor. List master processes running inside minikube Find how many nodes are there in the cluster and the roles of each node. 4. Delete kubernetes cluster

Once you have the VM installed, you can launch Minikube to create the Kubernetes cluster. To do this, open the command prompt and execute the “minikube start” command. This will create the cluster and set up the environment for you.

1. kubectl get pods
2. kubectl get nodes
3. minikube delete

q17) Create a Docker image which contains Python.Create a tag and push the image to Docker Hub registry

1. docker build -f my.Dockerfile .
2. docker build -t my-python-image .
3. docker tag my-python-image <your-dockerhub-username>/my-python-image:latest

4. docker login
5. docker push <your-dockerhub-username>/my-python-image:latest

q18) Deploy an application using minikube- create deployment.yaml, 3 pods, each containing one Docker container .

1. minikube start
2. Create deployment.yaml file with necessary image name and replicas:3
3. kubectl apply -f deployment.yaml
4. kubectl get deployment
5. kubectl get pods

q19) Create a docker image which contains a C language compiler installed in it. Provide the following outputs: a) Contents the Dockerfile to build the image (b) Output of command required to build the docker image. © Ensure the image is tagged as "cdev" (d) Show listing of the image in your docker environment (e) Push the image into your docker hub account (f) Make the image public in your docker hub repository

a) Contents of the Dockerfile to build the image:
```
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y \
   build-essential \
   gcc-8 \
   g++-8
CMD ["/bin/bash"]
```

b) Output of command required to build the docker image:
```
docker build -t cdev .
```

c) Ensure the image is tagged as "cdev":
```
docker tag cdev cdev:latest
```

d) Show listing of the image in your docker environment:
```
docker images
```

e) Push the image into your docker hub account:

```
docker push <username>/cdev
```

f) Make the image public in your docker hub repository:
```
docker login
docker push <username>/cdev:latest
```

q20) Refer to this link: GitHub - d1egoaz/minikube-kafka-cluster: Configures a minikube kafka cluster with 1 zookeeper node and 3 kafka brokers Follow the instructions to deploy Apache Kafka cluster with Zookeeper on Kubernetes using minikube.

q21) (a) Install minikube on your system, if it not already installed (Search Internet to locate minikube installer specific to your desktop/laptop Operating System) (b) Start minikube instance (c) List all Pods running in your cluster (d) Access minikube Dashboard (e) Deploy Kafka cluster

1. Install Minikube
   curl -Lo minikube
   https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
   && chmod +x minikube
2. Start Minikube
   ./minikube start
3. kubectl get pods --all-namespaces
4. minikube dashboard
5. kubectl create -f kafka-deployment.yaml

q22) Demonstrate all steps required to run nginx web-server in a docker and connect to it.
Document output of following steps: 1. Obtaining version 1.23.3 of nginx from Official Nginx Image repository in docker hub 2. Show the listing of the above docker image on your local system 3. Run the nginx document with the following Parameters: name: alpha-nginx, host-port:80, container-port: 80, use detach mode4. Login to the container and check status of nginx process inside the container 5. Stop the Container and Remove the docker image from your local system

Open command prompt
1. docker pull nginx:1.23.3
2. docker images
3. ipconfig (copy the ip address)
4. docker run -d --add-host = alpha-local:<ip-address> -p 80:80 nginx

5. docker container ls
6. docker container stop <container-id> (copy the id from step 5)
7. docker images

q23) Demonstrate a python based web server and run as a container. Document output of the following steps -or- Create a python based web server and run as container -or- Create a docker with RESTAPI

1. Download the python code from:
   https://github.com/sandeepacademe/devops/blob/main/py-webserver/py-websever.py
   Note : this requires python and it's dependencies flask, jsonify, requests to be installed
2. Create docker file called my-docker-file
   ```
   FROM python:3
   ADD py-webserver.py /
   RUN pip install flask
   RUN pip install requests
   RUN pip install jsonify
   CMD ["python", "./dewdrop.py"]
   ```
3. docker build -f .\my-docker-file -t myserver:1
4. docker images
5. docker run --add-host=alpha-local:<ipaddress> -p 80:8080 myserver