Kubernete Application & Archictecture(components) [Assignment-12]

1. **Container Management Tool: Docker Swarm**

- Docker Swarm is Docker's native clustering and container orchestration tool.
- It allows users to manage multiple Docker containers deployed across multiple host machines.

Disadvantages:

- Manual auto-scaling: No built-in support for dynamic scaling of containers.
- Manual load balancing: Requires manual configuration for traffic distribution.

---

2. **Evolution of Kubernetes (K8s)**

- Google Borg: Internal container management system at Google.
- Kubernetes was developed by Google using Borg’s principles and donated to CNCF (Cloud Native Computing Foundation) in 2014.
- Open-source and No Vendor Lock-in: Unlike proprietary tools, Kubernetes is not tied to a single cloud provider.

---

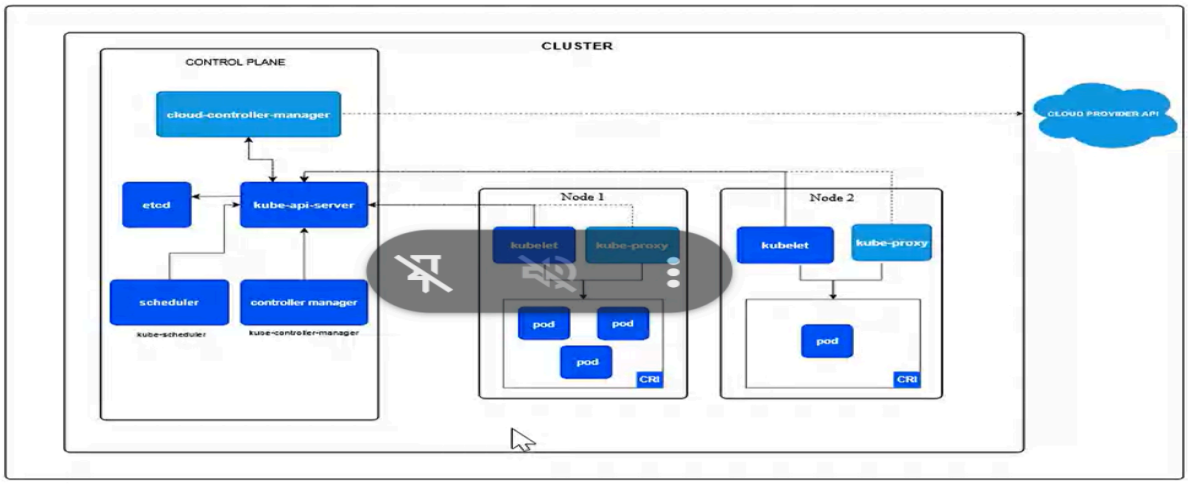
3. **Kubernetes Architecture and Cluster Components**

Control Plane Components:

1. etcd
    - Key-value store that stores all cluster data, configuration, and state.
2. kube-apiserver
    - Central management entity that exposes the Kubernetes API.
    - Handles communication between components.
3. scheduler
    - Assigns newly created Pods to nodes based on resource requirements and availability.
4. controller manager
    - Manages different controllers (e.g., replication, endpoint, namespace).
    - Ensures desired state of the cluster is maintained.
5. cloud-controller-manager (CCM)
    - Integrates Kubernetes with cloud providers like AWS, GCP, etc.
    - Manages cloud-specific services (load balancers, node information).

Node Components (Worker Nodes):

1. kubelet
    - Agent that runs on each node and communicates with the control plane.
    - Ensures containers are running in Pods as expected.
2. kube-proxy
    - Handles networking and load-balancing between Pods.
    - Manages network rules on nodes.
3. Pod
    - Smallest deployable unit in Kubernetes.
    - Can contain one or more containers sharing the same network and storage.
4. CRI (Container Runtime Interface)
    - Allows Kubernetes to use different container runtimes (e.g., containerd, CRI-O).
    - Responsible for pulling images and starting containers.



Kubernetes Cluster Workflow to Deploy 2 NGINX Containers

Step 1: Write a Deployment YAML File

Create a file named `nginx-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Step 2: Apply the Deployment

Run the following command to create the deployment:

```
kubectl apply -f nginx-deployment.yaml
```

Step 3: Kubernetes Control Plane Processes the Request

- **kube-apiserver** receives the deployment request.
- The request is stored in **etcd** (cluster state database).
- **Controller Manager** notices the desired state (2 replicas) and creates 2 Pods.
- **Scheduler** assigns the 2 Pods to available worker nodes.

Step 4: Worker Node Executes Pods

- **kubelet** on the assigned node pulls the `nginx:latest` image.
- It launches containers using the container runtime (**containerd/CRI**).
- The containers run inside Pods.

Step 5: kube-proxy Manages Networking

- **kube-proxy** sets up necessary network rules to route traffic to the Pods.
- If you expose the deployment using a **Service**, kube-proxy routes incoming traffic accordingly.

(Optional) Step 6: Expose the Deployment

To make NGINX accessible outside the cluster:

```
kubectl expose deployment nginx-deployment --port=80
--type=NodePort
```

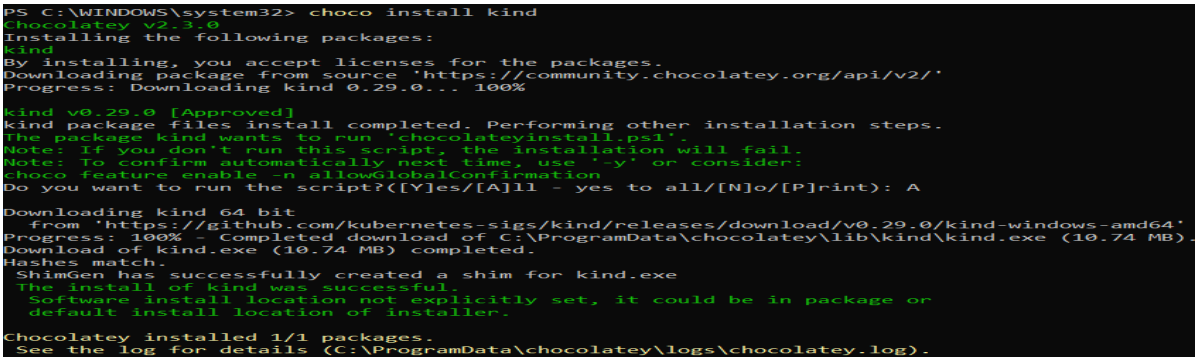
Verification

```
kubectl get pods

kubectl get deployments

kubectl get services
```

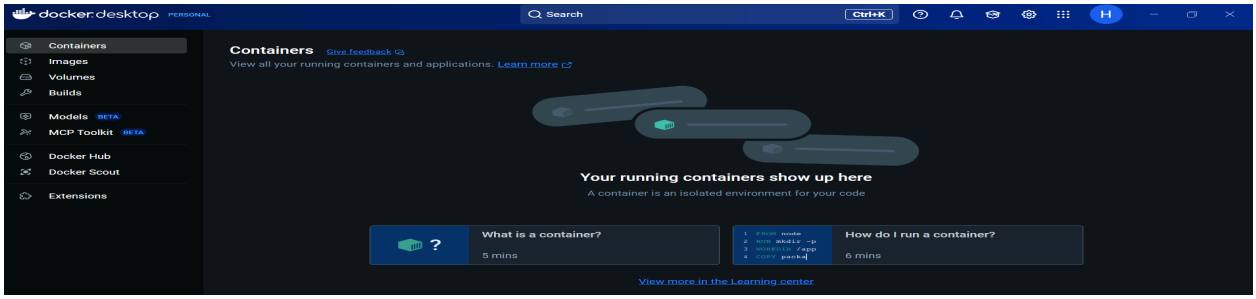
Lab Session commands: -

1. Create the Kind Cluster in Local.

    a. Install Kind (use choco to install)



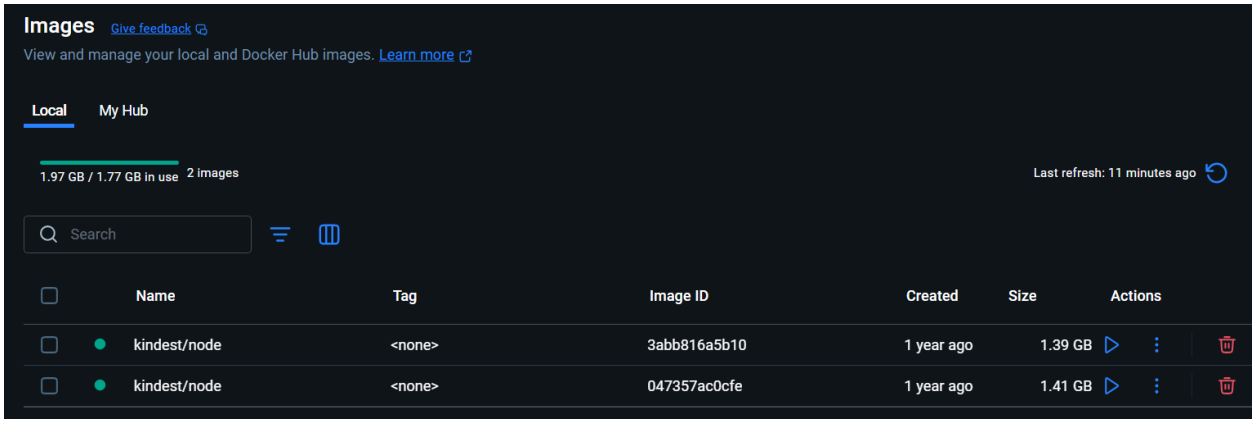
    b. Install docker desktop

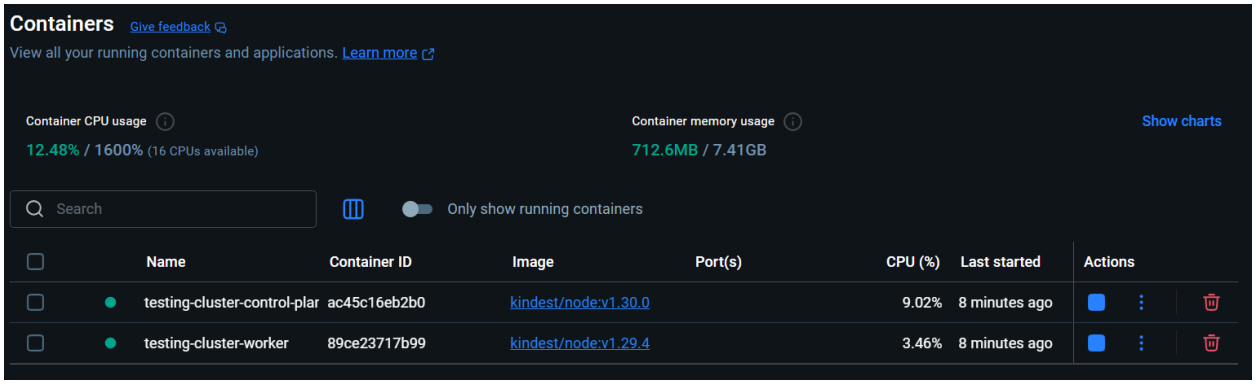


c. Use the command: - kind create cluster –name testing-cluster –config cluster.yaml

Cluster.yaml file:-

```
kind: Cluster

apiVersion: kind.x-k8s.io/v1alpha4

nodes:

  - role: control-plane

    image: kindest/node:v1.30.0@sha256:047357ac0cfea04663786a612ba1eaba9702bef25227a794b52890dd8bcd692e

  - role: worker

    image: kindest/node:v1.29.4@sha256:3abb816a5b1061fb15c6e9e60856ec40d56b7b52bcea5f5f1350bc6e2320b6f8
```





2. Or You can use online playground at killercoda.com
3. Use these commands and take screenshot of the outputs as the assignment.
    a. Kubectl get ns

```
PS C:\Users\Ayush Singh\Downloads\DevOps-Assignment> kubectl get ns
NAME                 STATUS   AGE
default              Active   12m
kube-node-lease      Active   12m
kube-public          Active   12m
kube-system          Active   12m
local-path-storage   Active   12m
```

    b. Kubectl get pods
    c. Kubectl get pods -n <namespace-name>
    d. Kubectl describe pod/<pod_name>

```
PS C:\Users\Ayush Singh\Downloads\DevOps-Assignment> kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
devops-pod   1/1     Running   0          2m33s
```

    e. Kubectl logs pod/<pod_name>

```
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl describe pod devops-pod
Name:             devops-pod
Namespace:        default
Priority:         0
Service Account:  default
Node:             testing-cluster-worker/172.18.0.2
Start Time:       Wed, 25 Jun 2025 16:20:15 +0530
Labels:           run=devops-pod
Annotations:      <none>
Status:           Running
IP:               10.244.1.2
IPs:
  IP:  10.244.1.2
Containers:
  devops-pod:
    Container ID:   containerd://ccd0f4f680f26a61d919766850e24c192955f4ad35e18393a398810ad203a8ef
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:dc53c8f25a10f9109190ed5b59bda2d707a3bde0e45857ce9e1ef
aa32ff9cbc1
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 25 Jun 2025 16:21:03 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-q4x2n (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-q4x2n:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
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
```

f. Kubectl run devops-pod –image= nginx

```
PS C:\Users\Ayush Singh\Downloads\DevOps-Assignment> kubectl run devops-pod --image=nginx
pod/devops-pod created
```

g. Kubectl get pods -n kube-system :- to display the components of the control plane pod of Kubernetes.

```
PS C:\Users\Ayush Singh\Downloads\DevOps-Assignment> kubectl get pods -n kube-system
NAME                                                 READY   STATUS    RESTARTS   AGE
coredns-7db6d8ff4d-kk6cx                             1/1     Running   0          17m
coredns-7db6d8ff4d-wwjxm                             1/1     Running   0          17m
etcd-testing-cluster-control-plane                   1/1     Running   0          17m
kindnet-q7lzq                                        1/1     Running   0          17m
kindnet-sjknx                                        1/1     Running   0          17m
kube-apiserver-testing-cluster-control-plane         1/1     Running   0          17m
kube-controller-manager-testing-cluster-control-plane   1/1     Running   0          17m
kube-proxy-6nhrw                                     1/1     Running   0          17m
kube-proxy-vslth                                     1/1     Running   0          17m
kube-scheduler-testing-cluster-control-plane         1/1     Running   0          17m
```

4. Create EKS cluster in the AWS.

    a. Command to create it:- eksctl create cluster \

    --name my-eks-cluster \

    --region us-west-2 \

    --nodegroup-name linux-nodes \

    --node-type t3.medium \

    --nodes 2 \

    --nodes-min 1 \

    --nodes-max 3 \

    --managed è check the cloudformation template and analyze it what is being created.

```
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl logs devops-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
2025/06/25 10:51:04 [notice] 1#1: using the "epoll" event method
2025/06/25 10:51:04 [notice] 1#1: nginx/1.29.0
2025/06/25 10:51:04 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14+deb12u1)
2025/06/25 10:51:04 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/06/25 10:51:04 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/06/25 10:51:04 [notice] 1#1: start worker processes
2025/06/25 10:51:04 [notice] 1#1: start worker process 32
2025/06/25 10:51:04 [notice] 1#1: start worker process 33
2025/06/25 10:51:04 [notice] 1#1: start worker process 34
2025/06/25 10:51:04 [notice] 1#1: start worker process 35
2025/06/25 10:51:04 [notice] 1#1: start worker process 36
2025/06/25 10:51:04 [notice] 1#1: start worker process 37
2025/06/25 10:51:04 [notice] 1#1: start worker process 38
2025/06/25 10:51:04 [notice] 1#1: start worker process 39
PS C:\Users\sidd2\Downloads\kubernetes_class>
```

b. Command to delete the Resource: - eksctl delete cluster –region <region-code> --name <name of the cluster.>

c. Also execute the sa

me commands in the EKS cluster also

d. Create two namespaces in EKS cluster and then create two pod of apache webserver (httpd) in that particular namespaces only,

```
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl create ns team-a
namespace/team-a created
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl create ns team-b
namespace/team-b created
```

```
Error: failed to create cluster "my-eks-cluster"
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl create ns team-a
namespace/team-a created
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl create ns team-b
namespace/team-b created
PS C:\Users\sidd2\Downloads\kubernetes_class>
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl run apache1 --image=httpd -n team-a
pod/apache1 created
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl run apache2 --image=httpd -n team-a
pod/apache2 created
PS C:\Users\sidd2\Downloads\kubernetes_class>
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl run apache1 --image=httpd -n team-b
pod/apache1 created
PS C:\Users\sidd2\Downloads\kubernetes_class> kubectl run apache2 --image=httpd -n team-b
pod/apache2 created
```