Using docker compose file we can run more than one container. Those container running independently or interact with each other to share the data. In docker compose all container must be running on same machine or node.

Container management tool or orchestration tool. Orchestration tool is responsible to manage life of more than one container. Health of the container, scale up or down, backup of any container down or destroy etc.

Orchestration tool list

1. Docker swarm
2. Kubernetes
3. Open Shift
4. EKS Amazon Elastic Kubernetes service
5. Azure containers app

Kubernetes : Kubernetes also known as K8s. Kubernetes is also one of the type of Orchestration tool which is responsible to manage more than one container life. Kubernetes is an open source container management tool provided by google. Using Kubernetes we can manage more than one container running on same machine or node or different machine or node.

Features of Kubernetes

1. We can deploy more than one application using container in Kubernetes environment.
2. Scale up / down
3. Load balance
4. Batch execution
5. Monitor health of containers.
6. Rollback
7. Deploy application

Node or machine : it is a single host machine or device that device can be local machine or vm machine.

Cluster : it is a collection of host or machine or server node or machine that help us to share resources.

1. We can use docker desktop : which provide single node cluster environment.
2. Minikube : it is an another open source gui base single node cluster provide for Kubernetes
3. kubeadm : it provide multi cluster environment for Kubernetes.
4. Kind : it is also one of the type single node cluster.

To interact with Kubernetes cluster we need to download one of the command line interface ie kube control ie kubectl. It is command line tool which help to interact with Kubernetes cluster using this tool we can deploy our application.

>kubectl cluster-info single cluster

This command provide us cluster information

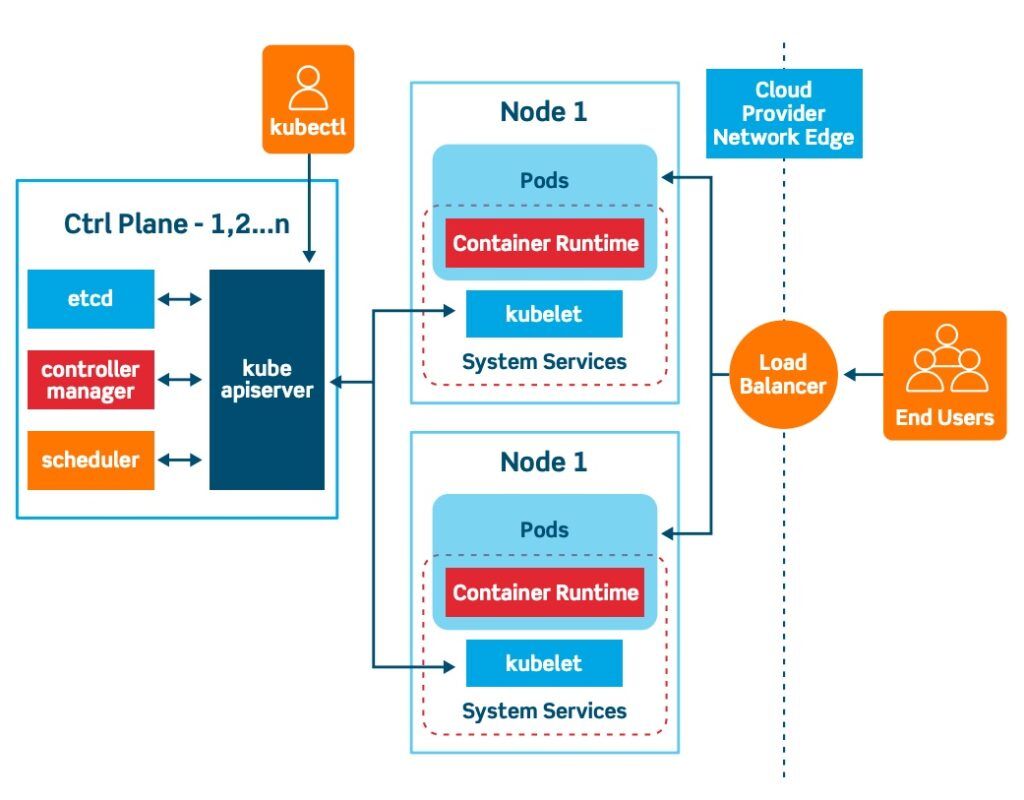
->kubectl get node

It provide us our machine details. This machine running cluster information ie control plane ie master node or main node.

We are only master node as well as worker node.

In real time using kubeadm we can run master node

Kubectl init to join more than one node to deploy the docker image.



In Kubernetes we can deploy or interact container directly. We need to use pods. Pods wrap one or more than one container and each container responsible to run application ie mysql or spring boot.

using pods we can run more than one container. We can specify optionally how much CPU and memory for each container required.

To find the pods information we need to run the command as

kubectl get pods

this command provide us pods details.

We can create the pods using ,yml or using commands. Ie imperative or declarative manner.

kubectl create depoyment hello-pods --image=akashkale/my-spring:1.111

kubectl get deployment

kubectl get pods

deployment provide extra layer to created or provide configuration to run more than one container using pods.

To delete deployment using command

Kubectl delete deployment hello-pods

Kubectl delete pods podsname

We can create more than one replica ie pods for same using kubectl commands.

kubectl create deployment hello-more-deployment --image=akashkale/my-spring:1.111 --replicas=10

Pods are part of Kubernetes cluster environment. We can’t access those pods outside cluster environment. We can access those pods within a cluster environment with private ip address.

To allow to access these application outside cluster environment we need to expose our pods as a service.

This we can expose using command or using yml file.

Using command we expose our service

Type of service

LoadBalancer

Cluster-Ipd

NodePort

kubectl expose deployment deployment\_name --type=LoadBalancer --port=8080

kubectl expose deployment my-deployment --type=LoadBalancer --port=8080

kubectl get service

then open the application with external-ip ie localhost

<http://localhost:8080>

1. we need to create spring boot application with simple rest api.

please test this application running using eclipse IDE.

<http://localhost:8080>

1. create the jar file for this project.
2. Then create the Dockerfile and create the image.

FROM openjdk:17

COPY ./target/filename.jar .

CMD [“java”,”-jar”,”filename.jar”]

1. Then create the image for that Dockerfile

docker build -t my-spring-boot . -f Dockerfile

1. Create the tag for your image

docker tag my-spring-boot dockerhubaccountid/my-spring-boot:k8s

1. Push this image in Docker hub account

docker push dockerhubaccountid/my-spring-boot:k8s

1. Ready to deploy image in Kubernetes cluster environment.
2. Enable Kubernetes features in Docker desktop ie setting and select Kubernetes checkbox and apply.
3. Open the command prompt check cluster information

kubectl cluster-info

kubectl get deployment

kubectl get pods

kubectl get service

if you want delete old deployment and service.

kubectl delete deployment deployment

kubectl delete service serviename

1. kubectl create deployment spring-boot-k8s-deployment --image=dockerhubaccountid/imageName:tag

like

kubectl create deployment my-deployment --image=akashkale/my-spring:1.111

1. check your deployment details or pods details

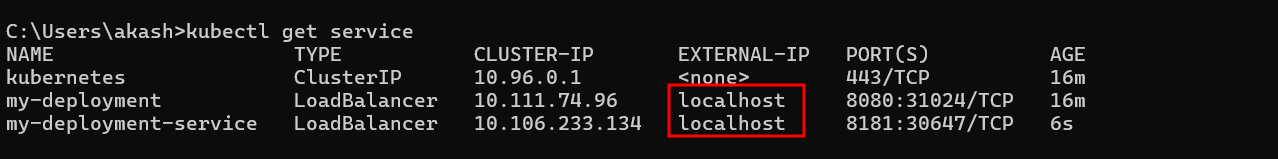
kubectl get deployment

kubectl get pods

1. kubectl expose deployment deploymentname --name=deployment-service-name --type=LoadBalancer –port=8080

kubectl expose deployment spring-boot-k8s-deployment --name= spring-boot-k8s-service --type=LoadBalancer --port=8181

using kubectl get service



Then test the application

<http://localhost:8080>