**Kubernetes Architecture**



**Kubernetes:**

Kubernetes is a container management technology to manage containerized applications in different kind of environments such as physical, virtual, and cloud infrastructure

Kubernetes comes with a capability of automating deployment, scaling of application, and operations of application containers across clusters. It is capable of creating container centric infrastructure.

**Features of Kubernetes**

* Continues development, integration and deployment
* Containerized infrastructure
* Application-centric management
* Auto-scalable infrastructure
* Environment consistency across development testing and production
* Loosely coupled infrastructure, where each component can act as a separate unit

**Worker Node**

**Node**: A node is simple physical or virtual machine

**Docker Runtime:** To run containers

**Kubelet**: It ensures everything should be running as part of configuration. The **kubelet** process then assumes responsibility for maintaining the state of work and the node server. It manages network rules, port forwarding, etc.

**Kube-proxy**: To handle networking services

**Master Node**

**Kubernetes Control plan:** All the PODS are managed by control plan through API**.**

**API Server:** which handles data validation and configuration for all of the APIs?

**Etcd:** This stores all the data needed by Kubernetes in key value pair.

**Scheduler**: this is the mechanism responsible for allocating pods to available nodes. The scheduler is responsible for workload utilization and allocating pod to new node.

**Control Manager:** Core Kubernetes logic and it’s a life cycle manager, The key controllers are replication controller, endpoint controller, namespace controller, and service account controller. The controller manager runs different kind of controllers to handle nodes, endpoints, etc .

**Cloud Control Manager:** This let Kubernetes to hook with the cloud engine.

**Docker Vs: Kubernetes:**

Docker is about packaging containerized applications on a single node and Kubernetes is meant to run them across a cluster.

**Steps to build & deploy to Kubernetes using docker and helm:**

**Docker container :** Docker is a software platform that simplifies the process of building, running, managing and distributing applications. It does this by virtualizing the operating system of the computer on which it is installed and running.

Docker image contains the Linux, Windows, mac’s minimal kernel software. The open source java is also part of the container. The application code uses java and Java uses the dependency packages to make the image file.

**Build Java application using maven build:**

C:\Users\bvn\_p\Downloads\demo\mvn build

The demo-0.0.1-SNAPSHOT.jar generated under C:\Users\bvn\_p\Downloads\demo\package

**Docker Commands:**

>>cd C:\Users\bvn\_p\Downloads\demo

**run the docker build by executing the Dockerfile commands**

>>docker build -t springio/gs-spring-boot-docker .

**Then run the docker commands**

>>docker container run -p 8081:8080 -d -it --name democontainer2 -t springio/gs-spring-boot-docker

**To see the docker containers running**

>>docker ps

http://localhost:8081/hello

**To stop the docker containers**

>>docker stop <NAME of the container>

**Kubernetes Commands**

>>kubectl create -f create-demo-pod.yml

kubectl create -f deployment-kubectl

**To check the POD status**

>>kubectl get pods

**To see the docker error**

>>kubectl describe pod demo-pod1

**To delete a pod from kubernetes-dashboard**

>>kubectl delete pod demo-pod

>>kubectl logs demo-pod1

**Sample “Dockerfile” for Linux image**

FROM openjdk:8-jdk-alpine

COPY target/demo-0.0.1-SNAPSHOT.jar demo-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["java","-jar","demo-0.0.1-SNAPSHOT.jar"]

**deployment-kubectl**

****

**Helm:**

Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.

Helm package contains…

Chart.yaml

values.yaml

Charts **folder**

Template **folder**

services.yaml -

deployement.yaml -

ingress.yaml - Contains input port

**Using EKS**