Introduction to Kubernetes

Managed Cloud Services:

GKE: Google Kubernetes Engine

- Compute Engines

EKS: Elastic Kubernetes Services

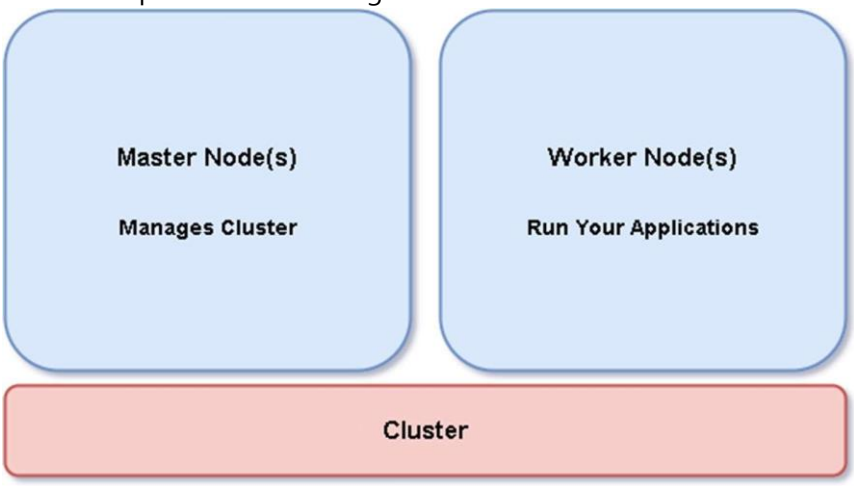
- EC2: Elastic Compute Cloud

AKS: Azure Kubernetes Services

- Virtual Machines

Kubernetes Cluster

- Combination of Nodes (worker) and Master Node
- Master Node is responsible for making worker nodes available

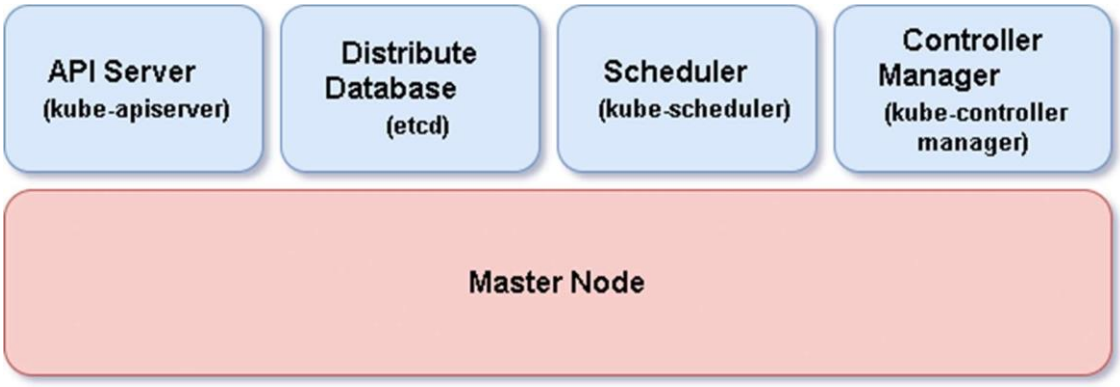


Kubernetes Architecture

- Important parts of Master Node
  - ✓ **etcd :**
    1. All the cluster, pods, replicasets, deployments information would be stored in distributed database
    2. This distributed database will always maintains 3-5 replicas of data in order to maintain the cluster details
  - ✓ **kube-apiserver:**
    1. API server is used to carry the commands from kubectl (command prompt) to kubernetes cluster
  - ✓ **kube-scheduler:**
    1. Scheduler is responsible scheduling the pods unto the nodes

2. When a new pod is getting created, on which node this pod has to be created would be done by scheduler
3. How much CPU is available? Are there any port conflicts? Scheduler considers all these facts before it schedules a pod on a node

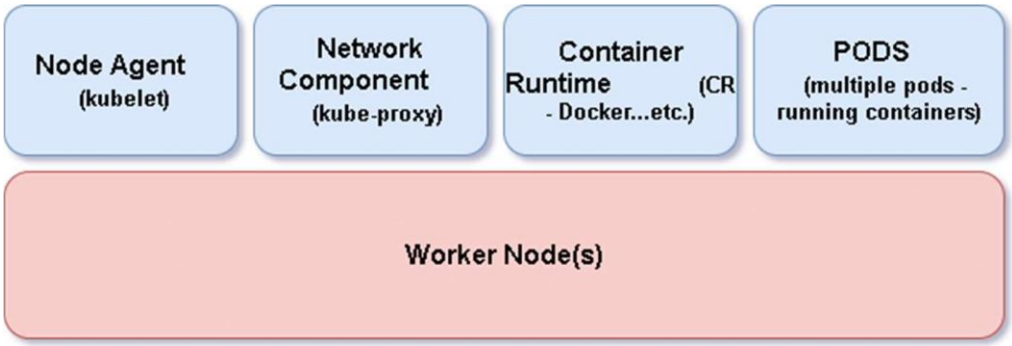
- ✓ **kube-controller manager:**
  1. kubectl manager is responsible for overall health of the cluster
  2. It make sures the actual state of kubernetes cluster matches to the desired state
  3. User applications will not be scheduled unto master node. But they would work only on worker node



Kubernetes Architecture

- ❖ Important parts of Worker Node
  - ✓ **Kubelet**
    1. The job of kubelet is to make sure what is happening on the node and communicates it back to the master node
    2. If a pod goes down, kubelet communicates to master node
  - ✓ **Kube-proxy**
    1. Creating deployments and exposing the deployments as service is the job of networking component
    2. It helps you in exposing services around your nodes and pods
  - ✓ **Container Runtime:**
    1. You can use Kubernetes with any OCI (Open Connector Interface)
    2. Most frequent one that is used as container runtime is Docker

- ❖ Commands
  - ✓ kubectl get componentstatuses



Kubernetes Architecture

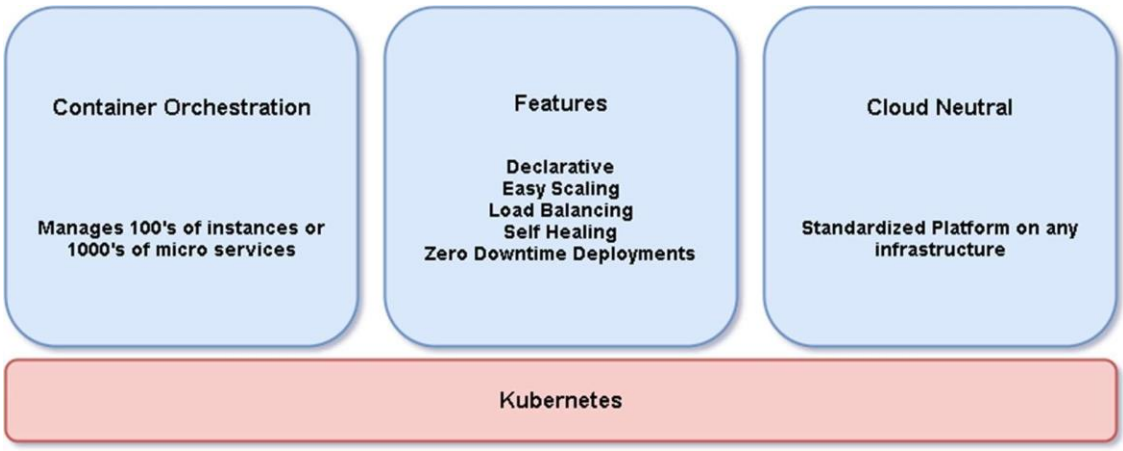
Creating Kubernetes Cluster:

Deploying Spring Boot App in Kubernetes:

- Kubernetes Controller – kubectl
- kubectl create deployment hello-world-rest-api --image=muralisocial123/hello-world-rest-api:0.0.1.RELEASE
- kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

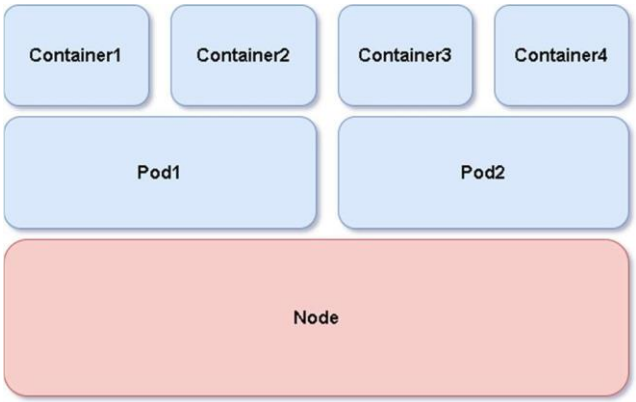
Kubernetes Concepts

- Kubernetes uses single responsibility concept.
- Each of pod, replicaset, service and deployment have very import role to play
- kubectl create deployment ===> created deployment, replicaset & pod
- kubectl expose deployment ===> created service



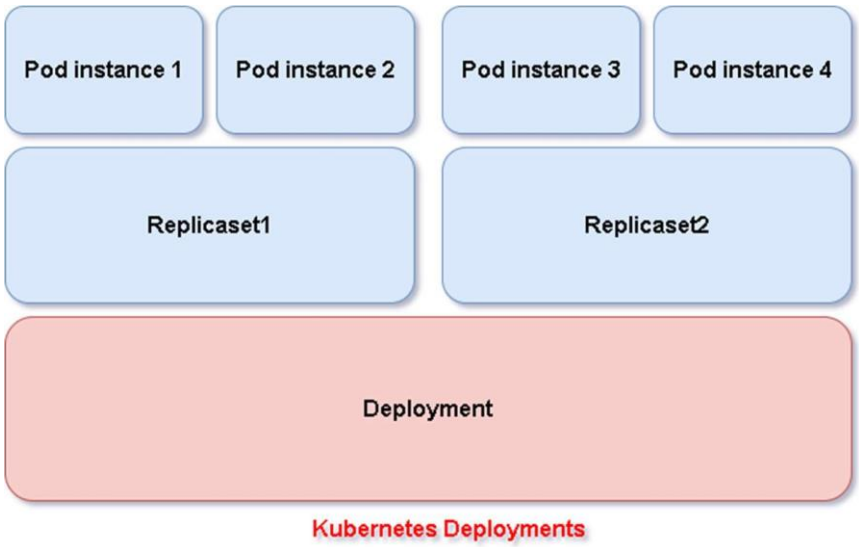
1. **Pods**
   - ❖ Pod is the smallest deployable unit in Kubernetes
   - ❖ Pod is a wrapper for set of containers
   - ❖ Pod provides a way to keep the containers together

- Pod has IP address, labels, annotations…etc.
- Commands
    1. kubectl get events
    2. kubectl get pods
    3. kubectl get replicaset
    4. kubectl get deployment
    5. kubectl get services
    6. kubectl get pods -o wide
    7. kubectl explain pods #what a pod is
    8. kubectl decribe pod pod_name #namespace, labels, annotations(meta-info), Status



2. **Replicasets [Maintains Required Number of Pods]**
    - Replicaset ensures that the specific number of pods are running at all times
    - Replicaset always keeps monitoring the pods and if there are lesser number of pods than what is needed then it creates new pod(s)
        - ✓ kubectl get pods -o wide #check the single pod created
        - ✓ kubectl delete pod_name #delete that single pod
        - ✓ kubectl get pods -o wide #replicaset will create new pod again automatically within 1 min to support the current services/containers
    - How can we convey to maintain fixed number of pods?
        - ✓ kubectl scale deployment hello-world-rest-api --replicas=3 #application has been scaled to 3 pods
        - ✓ kubectl get pods
        - ✓ kubectl get replicaset #DESIRED is increased to 3
        - ✓ kubectl get events --sort-by=.metadata.creationTimestamp #to know how scaling has happened
        - ✓ kubectl explain replicaset
    - Replicaset is always tied with a specific release version
3. **Deployment [Zero Down Time for Application Version upgrades]**

- When we would like to upgrade or downgrade our applications to specific version, without any down time Kubernetes can upgrade or downgrade your application instances
  - ✓ Deployment is need for release upgrade without downtime
  - ✓ Strategy: Rolling updates strategy for zero downtime deployment is the default one in strategies
  - ✓ It updates one pod at a time and deletes the old pod
  - ✓ Different release strategies



Kubernetes Deployments

- Create dummy deployment but still your old pod is running
  - ✓ kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY_IMAGE:TEST
  - ✓ kubectl get rs -o wide
  - ✓ kubectl get pods
  - ✓ kubectl describe pod pod_name

  - ✓ kubectl set image deployment hello-world-rest-api hello-world-rest-api=muralisocial123/hello-world-rest-api:0.0.2.RELEASE
  - ✓ kubectl get pods
  - ✓ kubectl get rs
- A

4. **Services [Always available external interface to the applications inside pod]**
   - In the Kubernetes world the pod is always a throw away unit. We can delete one pod and new pod can come up. But what about consumer side [url] changes due the pod changes? We don't want our URL get affected due to change in pod IP addresses

- Service in Kubernetes is to provide always available external interface to the applications which are running inside the pods
- Service allows the applications to receive the traffic through permanent lifetime IP address
- The service was created when we had executed "kubectl expose deployment" command
- GCP load balancer is handling the pods IPs mapping to permanent external interface [fixed IP]
- Commands:
  - ✓ kubectl get services

5. **GCP Workloads – Actions** [GUI]
   - Scaling
   - Auto Scaling
   - Expose
   - Rolling Update
6. **GCP Workloads – Edit** [GUI]
   - Yaml file
7. **Installing Gcloud**
   - https://cloud.google.com/sdk/docs/install
8. **Installing Kubectl**
   - https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/

Deploying Micro Services over Kubernetes