KUBERNETES

Basic Definitions

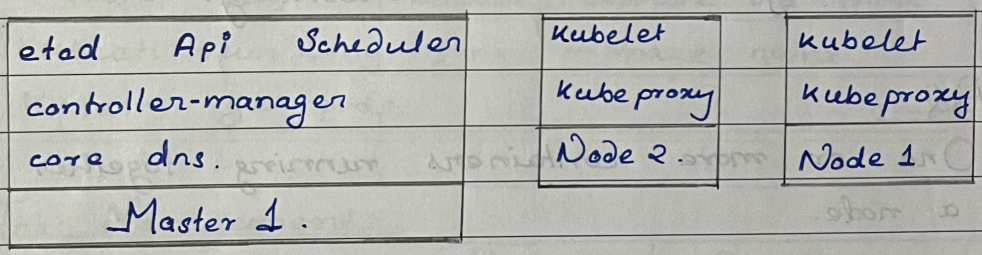
Kubectl : CLI to configure Kubernetes and manage apps.

Node : Single Server in Kubernetes Cluster.

kubelet : Kubernetes agent running on nodes.

Control Plane : Set of containers that manage the cluster. It includes API Servers, Scheduler, Controller manager etcd etc. The master.

Control Plane



There are always odd number of masters.

Inside Master

etcd : Distributed key value storage system.

API : Way to talk to clusters.

Scheduler: how and where your containers are placed in objects called pods.

Controller Manager: Looks at the state of the whole cluster (uses the API to do it). It takes the orders being given by you and determines the difference between what you are asking it to do and what is actually going on.

Core DNS: Used to control Dns.

Inside Node:

Kubelet: Agent running

Kube Proxy: to control networking.

Pod

One or more containers running together on a node.

Controller

For creating / updating pods and other objects. There are many types of controllers including Deployment, Replicaset, Stateful Set Daemon Set, Job, Cron Job etc.

classmate

Service

Network endpoint to connect to a pod.

Namespace

Filter a group of objects in a cluster.

Secrets

Store or manage information such as passw

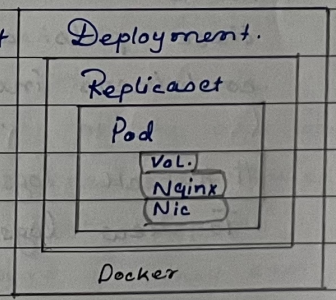
Config Maps

Api used to stores non confidential data in key value pairs.

```
# kubectl run          (Only for pod)
# kubectl create (Create resources via CLI/YAML).
# kubectl apply  (Create / Update via YAML).
# kubectl run my-nginx --image nginx
# kubectl get pods.
```

The Management.

Deployment manages Replicaset
Replicaset manages the pods.



classmate

`# kubectl run nginx --image nginx`

Creates replicaset deployment & pods on k8s version < 1.18. On >1.18 it only creates a pod.

`# kubectl create deployment nginx --image nginx`

This creates deployments.

`# kubectl scale deployment my-apache --replicas 2.`

This scale replica sets.

How does scaling work?

When we type the scale command we are actually updating the deployment spec. In k8s everything has a spec. Deployment controller changes the replica set to 2. Replica Set controller changes the pods to 2. Control Plane decides which nodes get assigned to these pods. and then kubelet creates containers inside pod.

`# kubectl logs deployment my-apache`

To view logs for a deployment.

`# kubectl logs -l run=my-apache`

To view logs for a group of pods.

`# kubectl describe pod my-apache`

To describe details of a particular pod.

`# kubectl delete pod my-apache`

If you delete a lower level abstraction ie. pod; the higher level abstraction i.e. Deployment or ReplicaSet will recreate it for you.

`# kubectl delete deployment my-apache.`

This would delete the deployment and its lower level abstractions as well.

`# kubectl get pods -w`

To watch pods.

Services

These are used to expose pods via DNS name.

Cluster IP: It is only reachable from within a cluster. This is the default service type. Pods reach svc on the app's port.

Node Port: High port assigned to service is allocated on each node. Anyone can connect to node port if they can

reach the node.
Cluster IP and NodePort are always available on Kubernetes.

Loadbalancer

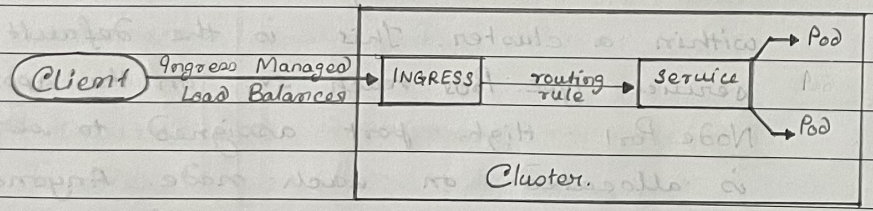
Only for traffic coming in from an external source. It can be thought of as an automation. It will automatically create Cluster IP, Node Port and then tell load balancer to send data/request to Node Port. It is only available when your infra provider gives you a load balancer eg. AWS ELB.

External

To talk outside a cluster. It adds CNAME DNS records to the core dns only. It isn't used to reach pods. It is used by pods to reach something outside k8s.

Ingress

Ingress exposes HTTP & HTTPS routes from outside the cluster to services within the cluster.




```
# kubectl expose deployment my-apache --port 8888
```

Creating a cluster IP service.

```
# kubectl get svc
```

Gets all services.

```
# kubectl expose deployment my-apache --port 8888
  --name my-apache-np --type NodePort.
```

Creating Node Port service.
Ports are shown in reverse order from docker. Container Port : Hostport → 8888 : 32334
High Port Range 30000 - 32767.

Services are Additive

Each one shall create the one below it.

Load Balancer
↓
Node Port
↓
Cluster IP.

Core DNS

It is the default DNS service. This is service discovery dns based. The following works only for services in the same namespace :- curl hostname. This is called hostname based discovery. There is also an FQDN.

spec:
All action goes in here.

More on Building Yaml files.

As is is very difficult to remember the formats in which we should write the Yaml file in order to create resources like service etc., we do 'explain' in order to see the Yaml details.

```
#kubectl explain <kind> --recursive
```

eg. services

```
#kubectl explain services.spec
#kubectl explain services.spec.type
```

The commands above will help creating the Yaml.

```
#kubectl apply -f file.yaml --dry-run.
#kubectl apply -f file.yml --server-dry-run
```

The above commands will do a dry run

```
#kubectl diff -f file.yml
```

The above command will show the difference as '+' & '-' for additions & deletions respectively.

Label

Goes under metadata; These are key value pairs which can be used as filters.

Selectors

These are put under services. They map key & value pairs.

Labels are identifying attributes of objects. They are used to organize and select subsets of objects. Selector uses label to identify objects.

FQDN: hostname.namespace.svc.cluster.local.

Pre Created Namespaces.

default
docker
kube-node-lease
kube-public auto created, visible to all users
kube-system. namespace for objects created by kubernetes

Generator

Helper templates that commands use while running.
You can see them by running
--dry-run -o yaml.

```
# kubectl create job test --image nginx --dry-run -o yaml.
```

Kubernetes Imparative

kubectl run, create, update, scale, edit are some of the kubernetes imperative commands.
These're not easy to automate.

Kubernetes Declarative

We don't care about how things are done.
We just tell kubernetes what we want.

classmate

This is much easier to automate.
This is the way to Gitops.

Kubernetes Declarative Objects.

apply -f file.yml or directory

```
# kubectl apply -f myfile.yml
# kubectl apply -f myfolder/
# kubectl apply -f URL
```

Kubernetes Configuration Yaml.

Each file has one or more manifest.
and each manifest describes one api object
eg. deployment or job or secret.
Manifest has four parts.
- apiVersion - kind - metadata - spec.

Building Yaml files.

kind:
We can get list of resources the cluster supports. Do the following

```
# kubectl api-resources.
```

api-versions:
Check this by running

```
# kubectl api-versions.
```

metadata:
Only a name is required.

classmate