**Kubernetes:**

Kubernetes, or K8s for short, is **an open-source container-orchestration tool designed by Google**. It's used for bundling and managing clusters of containerized applications — a process known as 'orchestration' in the computing world.

Why kubernetes not docker

* In docker, docker is a single host node container run-time, It runs on a single host and manages containers on that host
* In docker if the container exit, we need to find the error and fix it, then again run the docker container command.
* K8s is a distributed system that operates across multiple nodes. It consists of master node that manages the cluster and worker nodes that run the containers.
* In k8s, if the cluster in the node fails, it will be accessible in the other nodes.
* In k8s, we can expect applications to be available 24/7, and developers expect to deploy new versions of those applications several times a day.

**YAML – Basics**

* **Key value pair, Array/lists, Dictionary/Map**

A screenshot of a computer

Description automatically generated

**List of Dictionaries:**

A screen shot of a computer

Description automatically generated

* **Dictionaries = Unordered (it can be in unordered form)**
* **List = Ordered (It should in ordered form)**

A screenshot of a computer

Description automatically generated

**Simple Yaml File**

A screenshot of a computer

Description automatically generated

In the .yaml file for the Kubernetes object you want to create, you'll need to set values for the following fields:

* **apiVersion** - Which version of the Kubernetes API you're using to create this object.
* **kind** - What kind of object you want to create.
* **metadata** - Data that helps uniquely identify the object, including a name string, UID, and optional namespace.
* **spec** - What state you desire for the object.

## Types Of k8s Pod

### 1. Single-Container

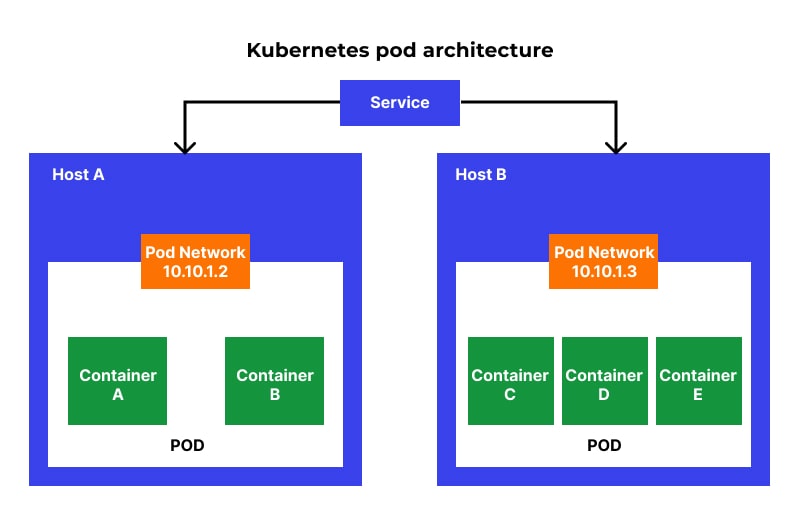
This is the simplest pod type as it features only one container per pod. It is often used for workloads deployments. Its generation is easy and requires only a kubectl command. Here is an example:

|  |
| --- |
| apiVersion: v1 |
| kind: Pod |
| metadata: |
| name: Tomcat |
| spec: |
| containers: |
| - name: Tomcat |
| image: tomcat: 8.0 |
| ports: |
| containerPort: 7500 |
| imagePullPolicy: Always |

‍

### 2. Multi-Container

|  |
| --- |
| apiVersion: v1 |
| kind: Pod |
| metadata: |
| name: Tomcat |
| spec: |
| containers: |
| - name: Tomcat |
| image: tomcat: 8.0 |
| ports: |
| containerPort: 7500 |
| imagePullPolicy: Always |
| -name: Database |
| Image: mongoDB |
| Ports: |
| containerPort: 7501 |
| imagePullPolicy: Always |



Kubernetes pod architecture

‍Pod Lifecycle

* **Pending phase**

In this phase, a Pod is fully created and is introduced to the cluster, but is not fully processed as one or many of its corresponding containers are not under process. Also, the time that Pod spends on an active node and image download is referred to as a pending phase.

* **Running phase**

A Pod is said to be in the running phase when it’s linked to a node, and all the containers are active.

* **Succeeded phase**

When all the Pod containers are successfully terminated, it’s known as a succeeded phase.

* **Failed phase**

This phase arrives upon the successful termination of containers of a pod, where one/multiple containers are terminated during a node failure.

* **Unknown**

When there is no clarity on where and what a Pod is doing, it is said to be present in an unknown phase.

‍

**COMMANDS:**

1. Create pod with nginx image/any other images.

* **kubectl run nginx --image=nginx**

1. Check the no.of pods

* **kubectl get pods**

1. check no.of nodes

* **kubectl get nodes**

1. check version

* **kubectl version**

1. Check OS

* **kubectl get nodes -o wide**

1. **create and edit the yaml file**

* **Kubectl run redis –image=redis123 –dry-run=client -o yaml**
* **Kubectl edit yamlfilename/ kubectl apply /kubectl edit pods imagename**

1. **Check replication controller**

* **Kubetcl get replicationcontroller**

1. **Replica set--- it has selector field in the yaml file to manage the pods.**

A Replica Set **ensures that a specified number of pod replicas are running at any given time.**

* New pod with same label not allowed
* Replicaset communicate through labels
* **Kubectl replace -f yamlfilename**
* **Kubectl scale –replicas=6 -f yamlfilename**
* **Kubectl get nameofreplicaset**
* **Kubectl delete replicaset nameOfTheReplicaSet (deletes all underlying pods)**
* **Kubectl create -f filename(to create the file)**

1. **Rollingupdate and Recreate**

* **Kubectl create -f filename**
* **Kubectl get name**
* **Kubectl apply -f filename**
* **Kubectl set image filename imagename**
* **Kubectl rollout status filename**
* **Kubectl rollout history filename**
* **Kubectl rollout undo deployment/my-deployment**
* **Kubectl create -f filename –record**

1. **Services**

* **Kubectl create -f servicefilename**
* **Kubectl get services**
* **Curl** [**http://ip:port**](http://ip:port)
* **kubectl apply -f /root/service-definition-1.yaml**

1. **Deployment Controller - (used for deployments and rollback) – It is like a object, if we update the application, deployment controller is required to act as a object, if we update another version and it has issues, it will help to rollback**

* Managing a set of pods in the form of Replica Sets & Hash-based labels.
* Rolling out new versions of application through new Replica Sets  
  – Rolling back to old versions of application through old Replica Sets  
  – Pause & Resume Rollout/Rollback functions  
  – Scale-Up/Down functions.

1. **Template – If the pod is terminated, we need the template to create a new pod**
2. **Cluster-IP – It gives you service inside your cluster that other apps inside your cluster can access. There is no external access.**
3. **NodePort – Able to access the application from the browser.**
4. **API Server – able to communicate with k8s through api-server (Master-node)**
5. **Kube-Controller-manager- monitors and manages the state of the applications (Master-node)**

* **It is responsible for noticing and responding when the nodes, containers and endpoints goes down. They make decision to bring up new containers in such cases.**
* **Node Controller --- responsible for noticing and responding when the nodes go down.**
* **Replication Controller ----- Responsible for maintaining the correct number of pods for every replication controller in the object.**
* **Endpoints Controller ------ Populates the Endpoints object (i.e.Json services and pods)**
* **Service account & token controller --- Creates default account and API access for new namespaces.**

1. **Etcd - stores the deployment data (Master-node)**

* **Stores all the masters and work node information’s.**

1. **Kubelet – worker nodes able to communicate (worker nodes)**

* **Kubelet is the agent that runs on every node in the cluster.**
* **Responsible for making sure that containers are running in a pod on nodes.**

1. **Kube-scheduler – It is a master node connects with worker node(kubelet)---- (Master-node)**

* **Scheduler is responsible for distributing containers across multiple nodes.**
* **It watches for newly created pods with no assigned nodes, select a node for them to run on**

1. **Kubelet proxy – user can access the application through this (user/external user)---( worker node)**

* **It runs on each node in a cluster.**
* **It maintains network rules on nodes.**

1. **Services – abstract way to expose an application running on set of pods.**
2. **Container-container---communicate through localhost+port**
3. **Pod-pod---communicate through network bridge**
4. **Node-node---communicate through network bridge or go to routing table nd check**

**WHY WE NEED INGRESS IN K8S**

1. **Ingress is a kubernetes resources that let us configure an HTTP load balancer for applications running on k8s, with advanced capabilities at HTTP Layer**

**Ingress features:**

1. **Context based routing.**
2. **Host based routing.**
3. **SSL/TLS termination.**

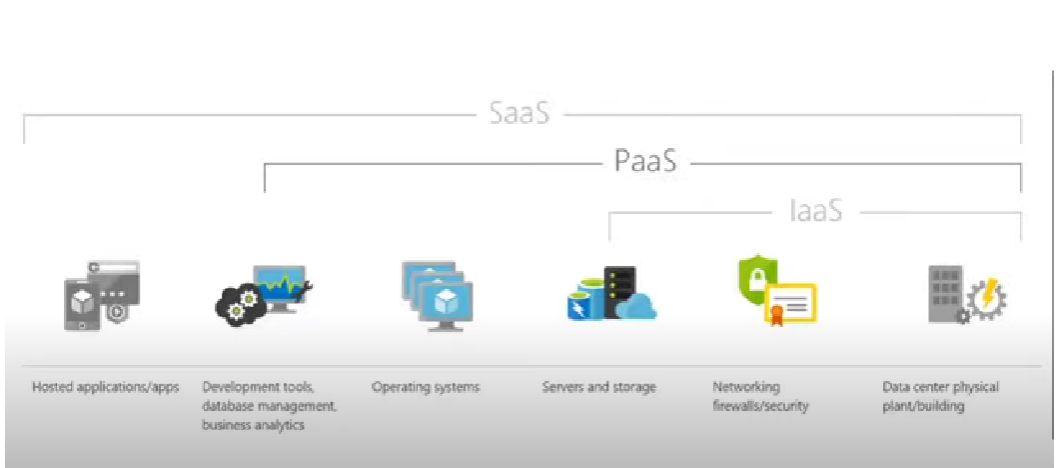
**Ingress allows access to your k8s services from outside the cluster.**

**LOAD BALANCER**

**Reduces the traffics of the server.**

**Types: round-robin, least connection, source IP hash, random, least traffic.**

Cloud-Computing – is the use of hardware and software to deliver the services over the network( internet)



Docker – to avoid OS issues

