Kubernetes Overview

[Kubernetes Basics: Pods, Nodes, Containers, Deployments and Clusters](https://www.youtube.com/watch?v=B_X4l4HSgtc&t=18s)

5 minutes of 
Pods, Nodes, Containers, ... 

A developer, DevOps or SRE engineer should know at least basics of how

K8S operates.

Kubernetes 
Deployment 
Pod 
Node 
Pod 
Node 
Node 

Below are PODs

Kubernetes 
Deployment 
Pod 
Node 
Pod 
Node 
Node 

Below pic shows Nodes

Kubernetes 
Deployment 
Pod 
Node 
Pod 
Node 
Node 

Below pic shows containers

Kubernetes 
Deployment 
Pod 
Node 
pod 
Node 
Node 

Below pic shows deployment

Kubernetes 
Deployment 
Pod 
Node 
Pod 
Node 
Node 

Below pic shows clusters

Kubernetes 
Deployment 
Pod 
Node 
pod 
Node 
Node 

**Node**:

Node is the smallest unit of computing hardware in the k8s cluster.

It’s a single m/c where our application will run.

A screenshot of a computer

Description automatically generated

What is a Node?

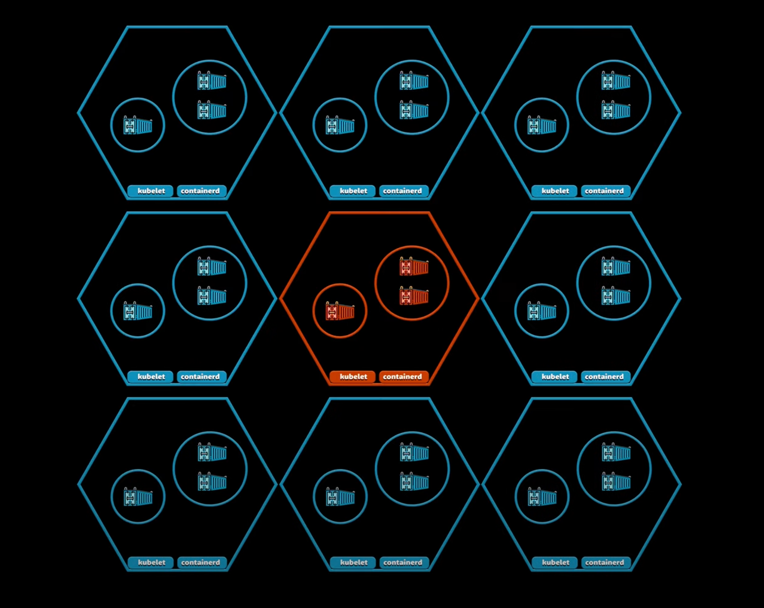
It is a single unit of computing hardware in the k8s cluster.

Node is a single machine where your application will run.

Node can be a physical server in a data center or a virtual machine in the public

cloud such as AWS or GCP.

We can even build Kubernetes from multiple raspberry-pi's.



It can be physical server in a data center or a virtual m/c in the public cloud

such as AWS or GCP.

We can even build Kubernetes from multiple raspberry-pi's.

Multiple nodes combined into a node pool.

When deploying a service, Kubernetes will inspect individual nodes for you and

select one node based on the available CPU, mem and other characteristics.

If for some reason a particular node fails, Kubernetes will make sure that your

application is rescheduled and healthy.

We can have multiple node pools/ instance groups in our cluster.

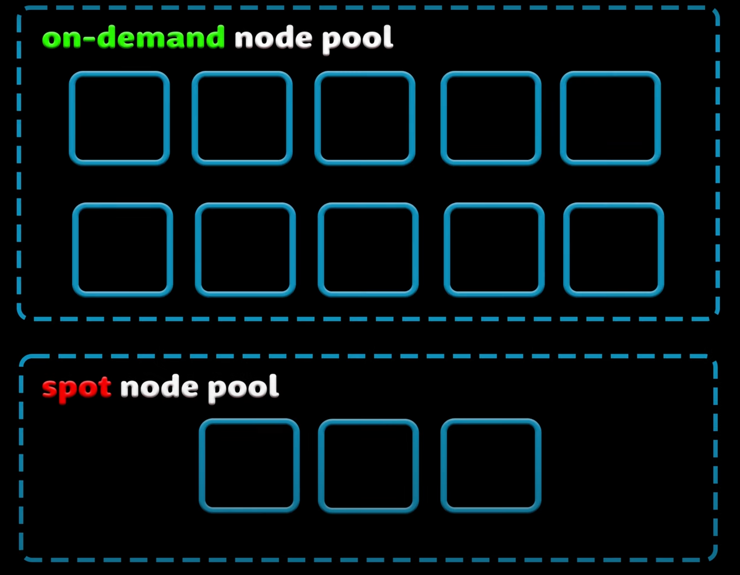
Node pools are also called as instance groups.

A black background with blue squares and white text

Description automatically generated

In the cloud it is very common to separate node pools to on-demand nodes and

spot nodes that are much cheaper but can be taken away at any moment.



Since applications running on your cluster is not guaranteed to run on a specific

node, we cannot use the local disk to save any data.

If the application saves something on the local file system then it is relocated to

another node, then the file will no longer be there.

A screenshot of a computer program

Description automatically generated

That’s why we can use a local disk as a temporary location for caching.

A black and yellow hexagon with a black background

Description automatically generated

A diagram of a diagram

Description automatically generated

To store data permanently, k8s use persistent volume.

While the CPU and memory resources of all nodes are pooled and managed by

the k8s cluster, persistent storage is not.

A computer screen with several servers

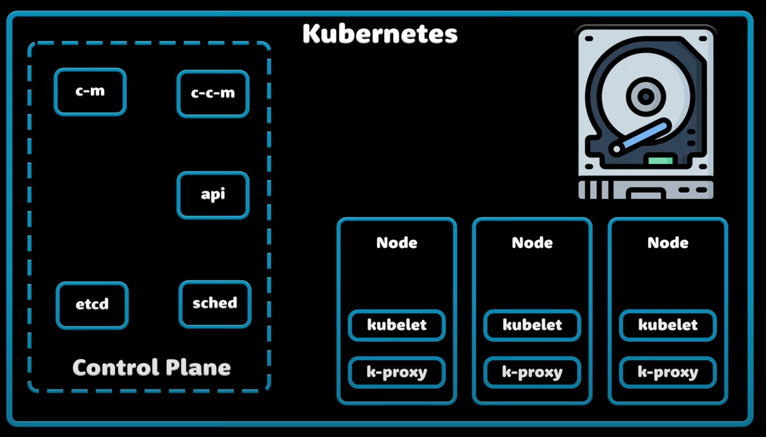
Description automatically generated

Instead, local or cloud drives can be attached to the cluster as a persistent volume.

Think of it like plugging external hard drive into the cluster.

A screen shot of a computer

Description automatically generated



Persistent volumes provide a file system that can be mounted to the cluster

without being associated with any particular node.

A diagram of a cloud server

Description automatically generated

To run an application on the K8S cluster we need to package it as Linux container.

A computer screen with a cross and a key

Description automatically generated with medium confidence

Containerization allows us to create a self-contained Linux execution environments.



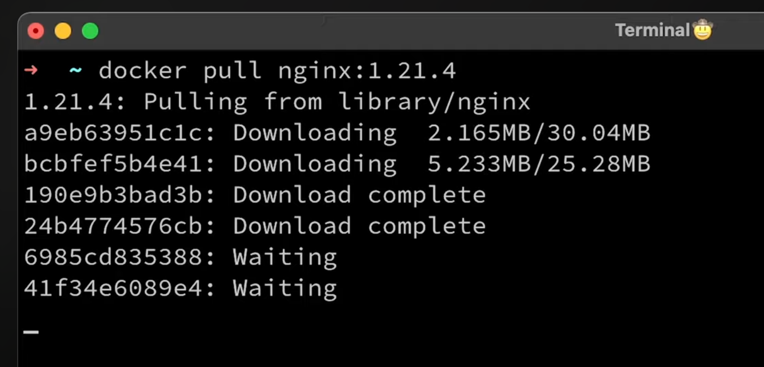
Any application and all its dependencies can be bundled up into a single image

and then can be easily distributed.



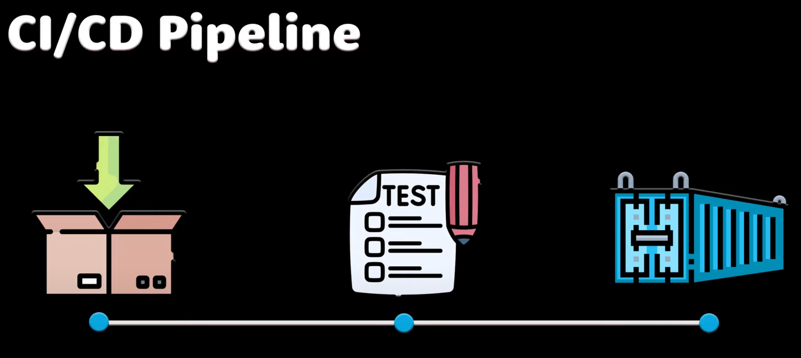
Anyone can download the image and deploy it on their infrastructure

with minimal setup required.



Usually creating a docker image is a part of CI/CD pipeline. You checkout

the code, run some unit tests and then build the image.



You can add multiple applications in 1 container but we should limit to

just 1 process per container.

Its better to have a lot of small containers than the large one.

If the container has tight focus, updates are easier to deploy, and issues

are easier to debug.

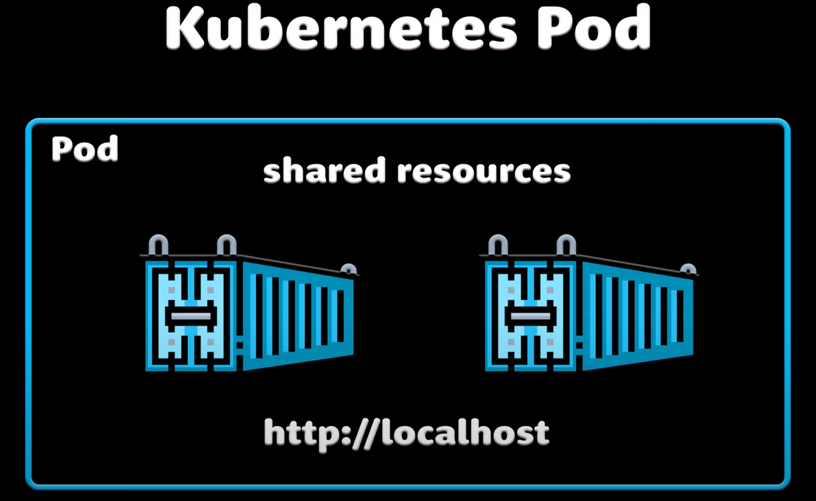
A black and white text with a blue phone and a globe

Description automatically generated with medium confidence

Kubernetes doesn’t run containers directly; instead, it wraps one or more

containers into a higher level structure called a pod.

Any containers in the same pod will share the same resources and local n/w.



Containers can easily communicate with other containers in the same pod as

though they were on the same machine while maintaining a degree of

isolation from others.

Pods are used as unit of replication in Kubernetes.

If application needs to be scaled up? Then we simply need to increase

the number of pods.

A screen shot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Kubernetes can be configured to automatically scale up and down your

application based on the load.

A diagram of a software company

Description automatically generated with medium confidence

We can use CPU, memory or even custom metrics such as number of

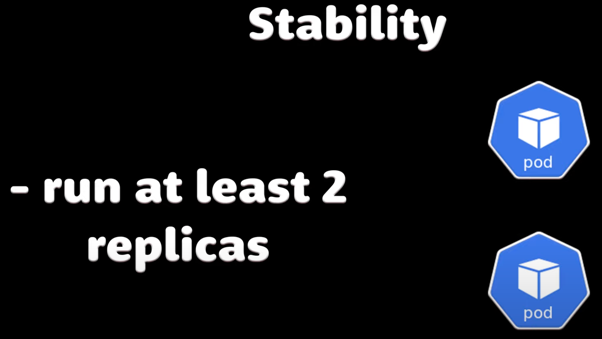
requests to the application.





Usually one would need to run multiple copies for the same application to

avoid downtimes, if something happens to single node.



As containers can have multiple processes inside a pod can have multiple

containers inside.

A screenshot of a black background

Description automatically generated

However since pods are scaled up and down as a unit, all containers in

a pod must be scale together, regardless of the individual needs. This

leads to waste of resources and expensive bill.

A black background with white text and red arrow

Description automatically generated

Pods should remain as small as possible to resolve this typically holding

only a main process and its tightly-coupled helper containers.

We typically call them side cars.

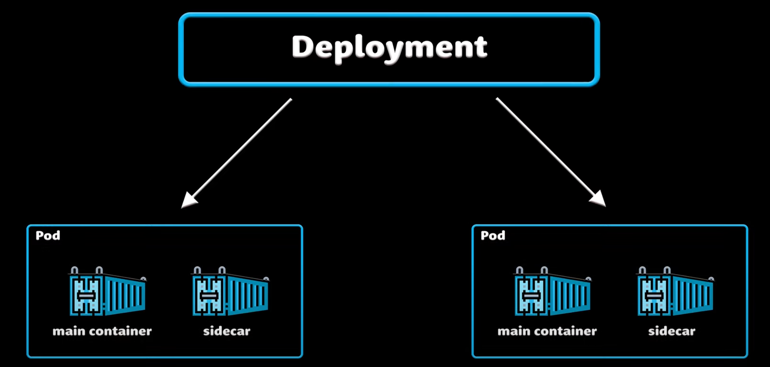
A black and blue container with white text

Description automatically generated

Pods are the basic unit of computation in K8S, but they are not typically

directly created in the cluster.

Instead K8S provides another level of abstraction such as deployment.



Deployments primary purpose is to declare how many replicas of a

pod should be running at a time.

A screenshot of a computer program

Description automatically generated

When deployment is added to the cluster, it will. Automatically spin up the

requested number of pods and then monitor it.





If a pod fails then the deployment will automatically re-create it.

Using a deployment we don’t have to deal with pods manually.

We just declare the desired state of the system and it will be managed for you automatically.

A diagram of a diagram

Description automatically generated

K8S load distribution.

K8S is an orchestration tool allowing us to run, and manage our

container based work-loads.

What is K8S?

Kubernetes is an open-source container management tool that automates

the deployment, scaling and management of containers.

**What is K8S namespace:**

In K8S namespace is a way to isolate groups of resources within a single

cluster, Kubernetes namespace provides below features:

Isolation:

Namespace allow different projects, teams, or customer to share a cluster

while keeping their resources isolated.

Resource management:

Namespace make it easier to manage resources by enforcing resource

quotas and policies.

Security

Many Kubernetes security policies are scoped to namespaces, such as

RBAC Roles and Network policies.

Collaboration

Namespace can help implement and automate secure, collaborative

workflows.

Scalability:

Namespaces can help scale infrastructure safely and efficiently.

Here are some more details about K8S namespaces:

Resource names:

Resource names within a namespace must be unique, but can overlap with

names in other namespaces.

Scoping

Namespace-based scoping only applies to namespaced objects, not

cluster-wide objects.

Communication:

Namespaces can communicate with each other, but users interacting with

one namespace cant see the content in another.

Network policies:

Network policies can be used to control access between namespaces.

Finding namespaces:

We can use *kubectl list namespaces* command to find our K8S namespaces.

In Kubernetes the most basic type of service one ca create is a ClusterIP

service type. It allows communication only between services within the

same cluster. Its important to understand that we cannot use this type

to expose your services to the outside world.

Container technology can bring speed and flexibility to enterprise IT.

But organizations need a way to automate container deployment

and manage a containers lifecycle.

Here we'll see about Kubernetes container management platform

and its main features, limitations and core components.

Kubernetes was introduced in 2014 by a team of developers at google.

Later google donated to cloud native computing foundation as open

source project.

To understand K8S we need to first understand containers.

Similar to virtual machine Containers virtualize a computers resources

and provision those resources into instances that can run software

services and applications.

But containers are smaller than virtual machines and they're ephemeral ,

meaning they might live and only exist for short durations.

This makes it impossible to create, start, organize, destroy and

monitor containers manually. So K8S and similar tools bring order

to the chaos by providing a platform to schedule and run containers,

while automating related operational tasks.

The core capabilities of K8S are container provisioning and management.

Administrators can use the platform to create and monitor containers

across cluster.

K8S can automatically restart failed containers, remove unresponsive

ones release and update and continuously check container health.

Kubernetes can also use DNS names or IP addresses to control how

containers are made available.

Its management capabilities include load balancing, which distributes

high traffic volumes across multiple container instances.

Kubernetes also handles storage, allowing admins to use varied

storage types from local storage to cloud resources for container data.

Other worthy features include rollouts and rollbacks.

K8s can set and modify the preferred state for container deployment.

This allows admins to create new container instances in a desired

state, and then migrate existing containers to the new instances while

removing the old ones.

K8S can also lend intelligence to container deployments and operations.

For example, administrators can tell K8s which nodes are available

and which resources are required for containers, and the platform

will automatically fir containers onto those nodes to optimize resource use.

Finally, organizations can use K8s to manage passwords, tokens, SSH keys

and other sensitive information's for all of its capabilities.

For all these capabilities K8s does have a few limitations:

First, K8s is not a software build tool. It handles the virtual instances

that hold software, but its not part of the CI/CD workflow.

K8s works much later in the process for deployment and operations.

Also K8s doesn’t provide middleware or application services. There

are countless services that can all be deployed in containers managed

by K8s and accessed by other applications running through K8s. But

those services are not native to K8s itself.

Kubernetes is a modular master node system.

A Node is a set of IT resources such as physical or virtual machines, on which one or more containers run.

A diagram of a computer

Description automatically generated

A Node supports Kubernetes pos, which is a group of containers and also the smallest unit of deployment in kubernetes.

The Master is a machine in the Kubernetes cluster responsible for handling control and cluster events.

The heart of the Kubernetes master system is a component called Cube Controller manger.

Cube controller manger runs controller for nodes, pod replication, service connections to pods, and services and tokens.

A scheduler called Cube scheduler decides which pods should run on a given Kubernetes node.

The scheduler makes its decisions bases on resource requirements, data locations, network traffic load, affinity and anti affinity rules, as well as hardware, software, and policy limits.

Kubernetes API is accessed through the Kubernetes API server component called cube API server.

To make a master node system work , several components must run on each Kubernetes node to manage the pods and maintain the runtime environment.

The kubelet component is an agent on each cluster node that ensure all contianers are in appropriate pod, while also checking for normal container operation and health.

A packet filtering network proxy called cube proxy also runs on each node to enforce network rules and manage communication between pods and the network.

A runtime components such as Docker, or Kubernetes container runtime interface, is also needed to operate containers and integrate them with Kubernetes.

As an open source project Kubernetes can be downloaded and used for free.

Enterprises can also choose a vendor-supported distribution of Kubernetes or use Kubernetes as a service through public cloud providers.

**Extra info about Nodes**:

A Kubernetes node is a machine that runs containerized workloads as part of Kubernetes cluster. A node can be a physical machine or virtual machine, and can be hosted on-premises or in the cloud.

A Kubernetes node is a machine that runs containers and is part of a Kubernetes cluster. Nodes can be either physical or virtual machines, depending on the cluster.

Below are few things that **K8S nodes do**:

**Run containers**: Nodes run containers, which are containerized applications and pods.

**Cluster Management**: Nodes are responsible for scheduling, detecting, and responding to events in the cluster.

**Load balancing**: Nodes are interconnected to ensure efficient load balancing.

There are two types of Nodes in K8S cluster:

**Master node**: Runs the K8S control plane, which controls entire cluster.

**Worker nodes**: Run containerized workloads.

Each **node has few components, including**:

**Kubelet**: Monitors the node, sees what it is running, and delivers instructions to the container runtime.

**Kube-proxy**: Enables networking on kubernetes nodes.

**Container runtime**: runs containers on the node.

<https://www.techtarget.com/searchitoperations/definition/Kubernetes-Node>

What are Kubernetes Node? :

A Kubernetes node is a logical collection of IT resources that runs workloads for one or more containers in a Kubernetes cluster.

Nodes contain the services necessary to run pods (Kubernetes term for groups of containers that operate together), communicate with control plane components, configure networking and run assigned workloads. Each node can host one or multiple pods.

What is secret in K8S:

* Essentially, K8S secrets are placeholders for sensitive information like credentials, tokens, and certificates, a mechanism for abstracting them.
* K8S secrets act as a separate objects which can be queried by the application Pod to provide credentials to the application for access to external resources.

A diagram of a server

Description automatically generated

K8S architecture

Kubernetes

KUBERNETES CLUSTER 
MASTER NODE 
MASTER NODE 
WORKER NODE 
WORKER NODE 
u 
. container 
Container Runtirne 
WORKER NODE 
WORKER NODE 
u let 
u -proxy 
u let 
u proxy 
u let 
Ku proxy 
: container 
Container Runtime 
container 
Container Runtirne 
container 
Container Runtime 

Pods: pods are the smaller deployable unit.

Deployment: Deployment manages pods lifecycle.

POD 
POD 
• grade-submission-portal 
Container 
Label Groupings : • grade-submission-api 
5001 
localhost 
Label Groupings 
3000 
localhost 
Container 

Services: Using K8s services we can allow our applications to find and communicate with each other.

Services allow communication between different components of our application, like webservers and databases.

45.51.17.63 
grade-submission-portal 
Container 
Grade Submission Portal 
grade-submission 
5001 
Resource Requirements 
Label Selector 
SERVICE 
00 

Namespaces: We can organize and isolate our resources within a cluster namespace.

Kubernetes Cluster 
Namespace 
Namespace 
Namespace 
Namespace 
POD 
POD 
Ca*airv 
POD 
J" Cmt*v 
POD 
POD 
POD 
POD 
POD 
Namespace 
Namespace 
Namespace 
Namespace 
POD 
POD 
POD 
POD 
îcontainer 
POD 
POD 
POD 
POD 

Deployment Objects: We can create many instances of our app using deployment objects.

Automated Deployment 
YAML Configuration File 
Desired State 
Kubernetes 
Actual State 
Desired State 

Config Maps: We can separate configuration from code, for non-sensitive data.

Configuration Management 
Deployment 
Deployment Details 
•1 
ConfigMap 
Configuration 
non-confidental 

Config Maps and secrets are used for configuration management and secretly store sensitive data.

Secrets: For sensitive data we use secrets.

Configuration Management 
Deployment 
Deployment Details 
•1 
Secret 
Configuration 
sensitive 

HPA: Horizontal pod auto scaler helps us to scale our applications automatically.

1/200 
POD 
Container 
1/200 
POD 
Container 
1/200 
POD 
Container 
HPA 
Minimum Replicas 
Maximum Replicas 
Target 
1 
10 
CPU 
Average Utilization 1% 

Ingress: We can control external access to our services using ingress controllers.

Ingress allow external access to our services. (typically HTTPS and HTTP traffic)

grade-submission-portal 
Container 
Grade Submission Portal 
Ingress Controller 
Label Grouping 
5001 
Resource Requirements 
ClusterlP 
KUBERNETES 
Ingress Rules 
Ingress Controller 
127.0.0.1 

Helm: We can simplify application deployment using helm charts.

HELM CHART 
Chart.yaml 
values.yaml 
templates/ 
- deployment 
- service 
- statefulset 
mongo DB 
HELM CHART 
Chart.yaml 
values.yaml 
templates/ 
- deployment 
- service 
- statefulset 
SQLc 
PWA. I 
HELM CHART 
Chart.yaml 
values.yaml 
templates/ 
- deployment 
- service 
- statefulset 

K8s Operators: To automate management and deployment of very complex applications.

A diagram of a computer

Description automatically generated

What Problem Does Kubernetes Solve

**What is Kubernetes?**

Kubernetes is an open-source container management tool that automates the deployment,

scaling and management of containers.

**What problem does Kubernetes solve?**

Monolithic 
Application 
Recruitment Website 
Job 
Applicants 
Job 
Vacancies 
Recruiters 
Transition to 
Docker 
Microservices 
Create containers for 
Recruitment Website 
your application 
Recruiters 
Job 
Applicants 
docker 
Job 
Vacancies 
Kubernetes 
Launch your containerised 
application in K8s 

Kubernetes addresses several challenges associated with deploying and managing

containerized application in a distributed environment.

Here are some key problems that Kubernetes solves:

**Container Orchestration:**

**Challenge:** Manual deployment and management of individual containers for

services is complex.

**Kubernetes:** Automates the deployment, scaling, and management of

microservices, ensuring seamless operation.

**Declarative Configuration:**

**Challenge:** Defining configurations for complex applications can be

error-prone.

**Kubernetes:** Uses manifests or declarative configurations, enabling you to

specify microservices desired states. Kubernetes ensures actual states match

defined, minimizing configuration errors.

**Scalability:**

**Challenge:** Scaling Microservices efficiently becomes difficult as the

recruitment platform grows.

**Kubernetes:**  Automatically scales services based on demand, Optimizing

performance across cluster nodes.

**Fault Tolerance and High Availability:**

**Challenge:** Ensuring continuous availability for all the services is challenging.

**Kubernetes:** Ensures fault tolerance by automatically rescheduling failed

containers, maintaining the desired state, and ensuring high availability.

**Service Discovery and Load Balancing:**

**Challenge:** Discovering and connecting to dynamic services is challenging.

**Kubernetes:** Automates service discovery and load balancing, facilitating

seamless communication between service interfaces.

**Rolling updates and Rollbacks:**

**Challenge:** Updating or rolling back applications without downtime is risky.

**Kubernetes:** Supports seamless updates without disrupting user activities

and provides the capability to rollback changes too.

Kubelet vs Kubectl

Kubectl is a commandline interface for working with kubernetes cluster.

Kubelet is the technology that applies, creates, updates, and destroys containers on a kubernetes node.

Kubectl is the primary means by which a developer can interact with a kubernetes cluster. For example, if you want to get the basic information about the nodes in a cluster, you'd type the command:

Kubectl get nodes

The result of the command would be similar to the following:

NAME    STATUS   ROLES    AGE   VERSION  
node1   Ready    master   89s   v1.19.1  
node2   Ready    <none>   58s   v1.19.1  
node3   Ready    <none>   58s   v1.19.1

How kubectl works:

If you wanna create a Kubernetes resource such as a service, we need to execute the command

Kubectl apply -f myservice.yaml

Here myservice.yaml is the name of ficticious manifest file that describes a Kubernetes service.