KPLABS Course

Certified Kubernetes Administrator 2020

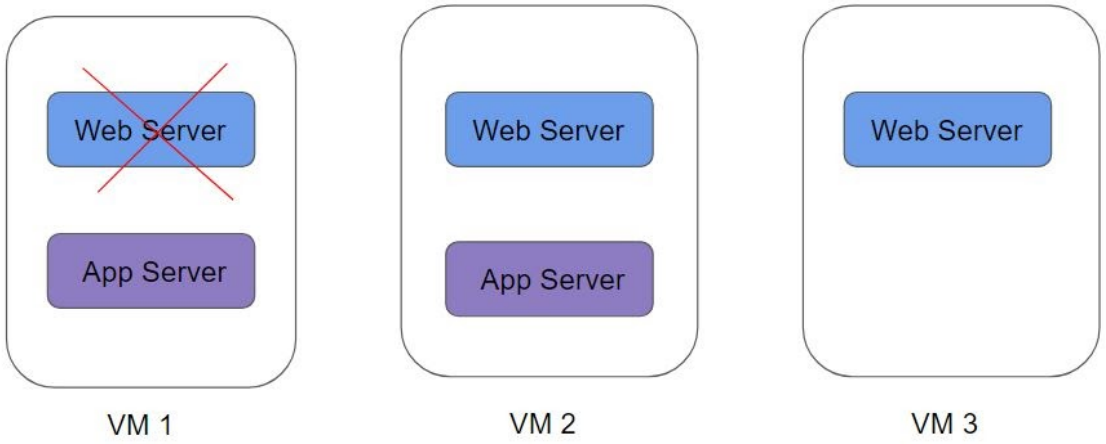
Core Concepts

ISSUED BY
Zeal Vora

REPRESENTATIVE
instructors@kplabs.in

Module 1: Overview of Container Orchestration

Container orchestration is all about managing the life cycles of containers, especially in large, dynamic environments.



Container Orchestration can be used to perform a lot of tasks, some of them includes:

- Provisioning and deployment of containers
- Scaling up or removing containers to spread application load evenly
- Movement of containers from one host to another if there is a shortage of resources
- Load balancing of service discovery between containers
- Health monitoring of containers and hosts

There are many container orchestration solutions which are available, some of the popular ones include:

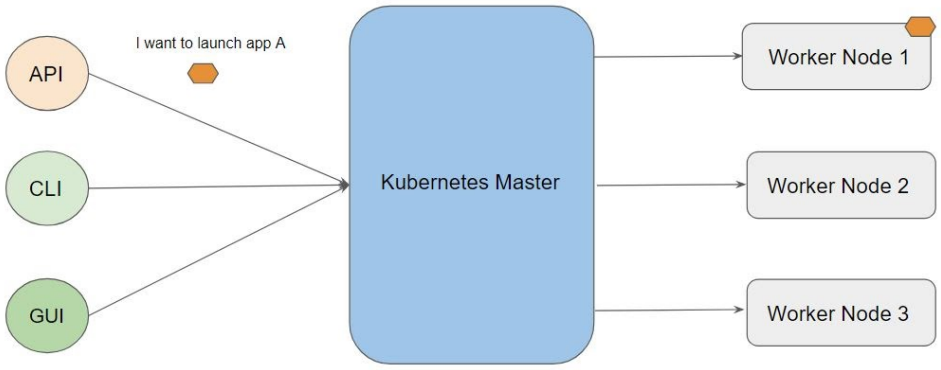
- Docker Swarm
- Kubernetes
- Apache Mesos
- Elastic Container Service (AWS ECS)

There are also various container orchestration platforms available like EKS.

Module 2: Introduction to Kubernetes

Kubernetes (K8s) is an open-source container orchestration engine developed by Google.

It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation.



Module 3: Installation Options for Kubernetes

There are multiple ways to get started with a fully functional Kubernetes environment

1. Use the Managed Kubernetes Service
2. Use Minikube
3. Install & Configure Kubernetes Manually (Hard Way)

3.1 Managed Kubernetes Service

Various providers like AWS, IBM, GCP, and others provide managed Kubernetes clusters.

Most organizations prefer to make use of this approach.

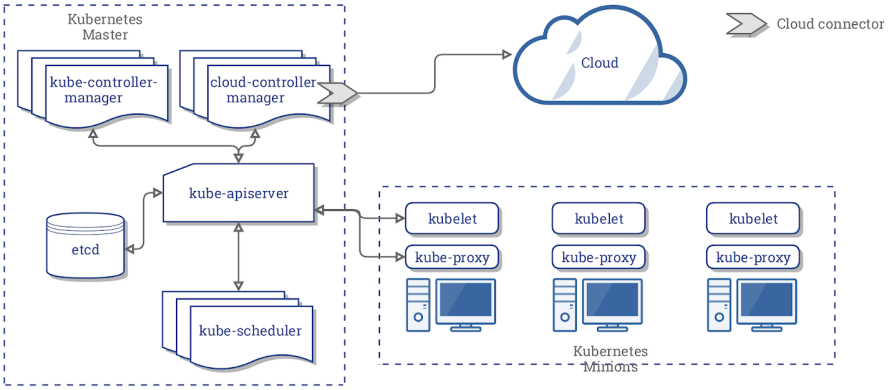
3.2 Minikube

Minikube is a tool that makes it easy to run Kubernetes locally.

Minikube runs a single-node Kubernetes cluster inside a Virtual Machine (VM) on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

3.3 Kubernetes the Hard Way

In this approach, you install and configure components of Kubernetes individually.

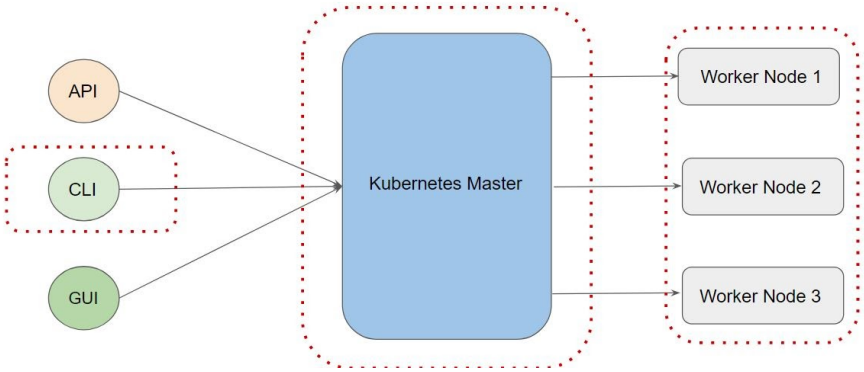


3.4 Installation Configuration:

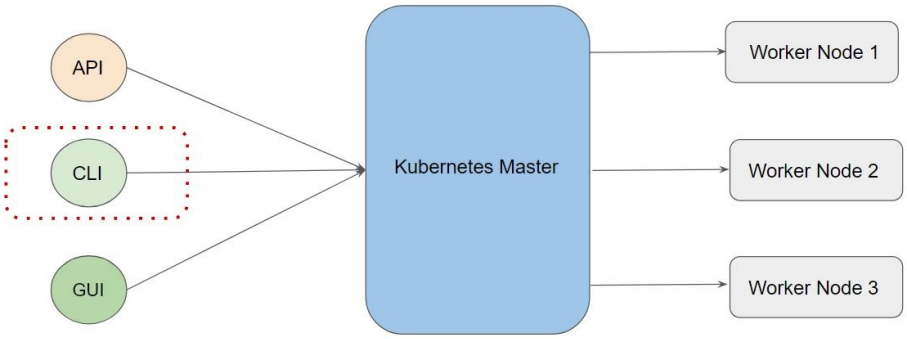
Things to configure while working with Kubernetes.

Sr No	Things to Install	Description
1	kubectl	CLI for running user commands against cluster.
2	Kubernetes Master	Kubernetes Cluster by itself.
3	Worker Node Agents	Kubernetes Node Agent

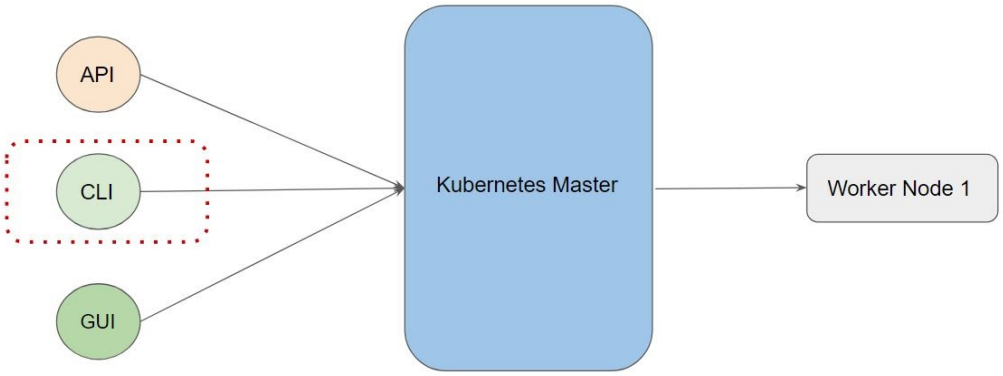
The following components highlighted in red are the ones that need to be configured while designing Kubernetes cluster in a hard way.



The following components highlighted in red are the ones that need to be configured while using managed Kubernetes cluster.

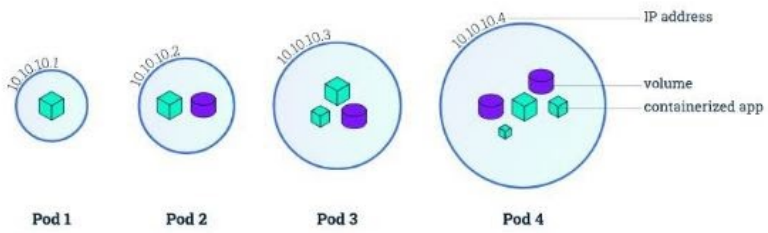


The following components highlighted in red are the ones that need to be configured while using minkube based installation method.

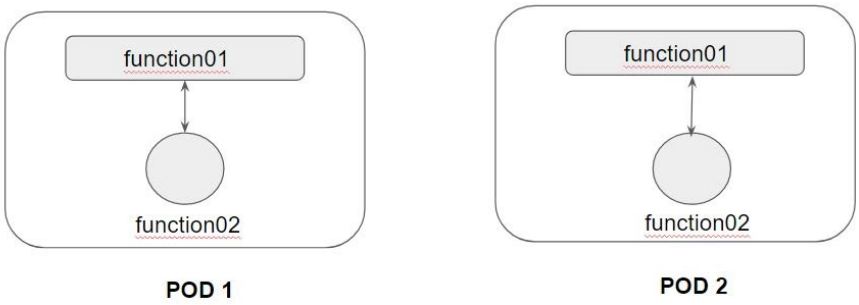


Module 4: PODS

A Pod in Kubernetes represents a group of one or more application containers and some shared resources for those containers.



Containers within a Pod share an IP address and port space and can find each other via the localhost.



A Pod always runs on a Node.

A Node is a worker machine in Kubernetes.

Each Node is managed by the Master.

A Node can have multiple pods.

Module 5: Kubernetes Object

Kubernetes Objects is basically a record of intent that you pass on to the Kubernetes cluster.

Once you create the object, the Kubernetes system will constantly work to ensure that object exists.

There are various ways in which we can configure a Kubernetes Object.

- The first approach is through the kubectl commands.

- The second approach is through a configuration file written in YAML.

pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mywebserver
spec:
  containers:
  - name: mywebserver
    image: nginx
```

YAML is a human-readable data-serialization language.

It designed to be human friendly and works perfectly with other programming languages.

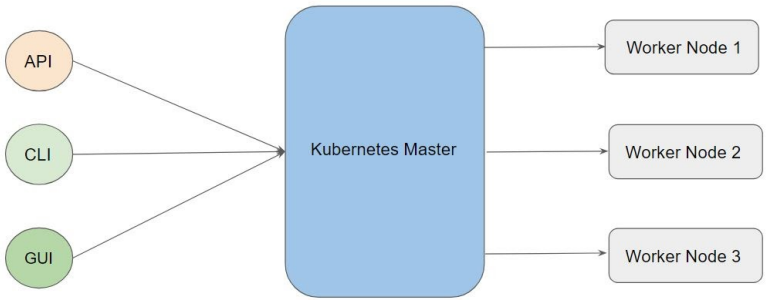
XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

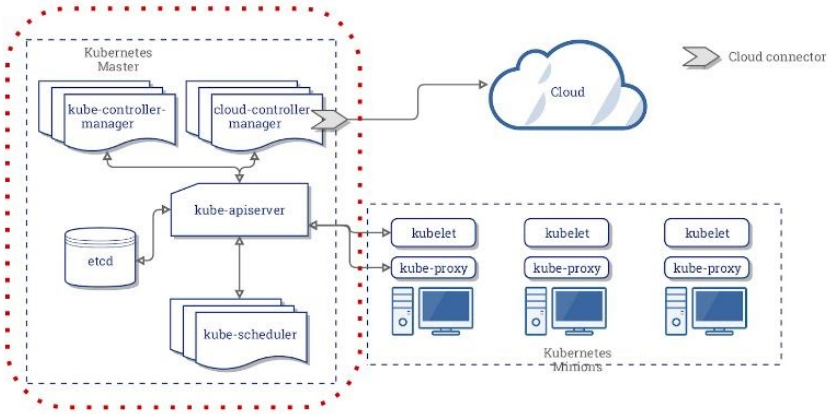
Module 6: Kubernetes Architecture

Before we start full-fledged deep dive into Kubernetes, understanding the architecture is beneficial.

ETCD, API Server, Scheduler, Controller Manager, Cloud Controller Manager.



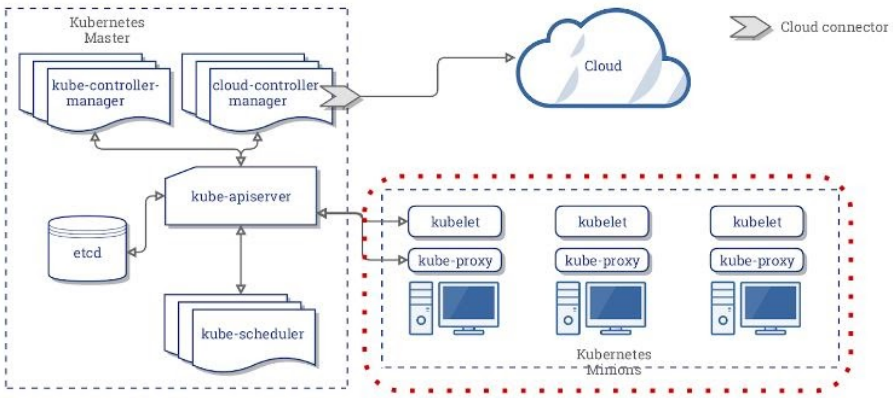
Kubernetes Master consists of five major components that are highlighted in red.



The following table describes the five components of Kubernetes Master.

Component Name	Description
kube-apiserver	Component on the master that exposes the Kubernetes API.
etcd	Key value store used as Kubernetes' backing store for all cluster data.
kube-scheduler	Component on the master that watches newly created pods that have no node assigned, and selects a node for them to run on.
kube-controller-manager	Responsible for controlling various aspects, including: Node Controllers: Responsible when node goes down. Replication controllers, endpoint controllers, Service account & Token controllers.
cloud-controller-manager	Runs controllers that interact with the underlying cloud providers.

Kubernetes Worker Node consists of two major components:



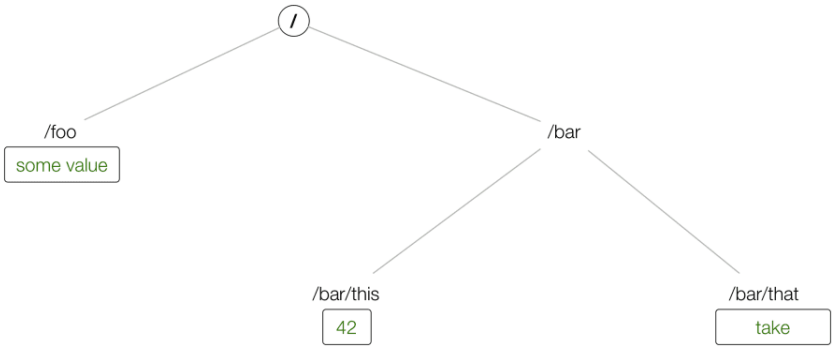
Node components run on every worker node of the Kubernetes Cluster.

Component Name	Description
kubelet	An agent that runs on each node in the cluster. It makes sure that containers are running in a pod.
kube-proxy	Acts as a network proxy which maintains network rules on the host and performing connection forwarding.
Container Runtime	Software which is responsible for running containers. Supported Runtimes: Docker, containerd, rktlet and others.

Module 7: K8s Component - ETCD

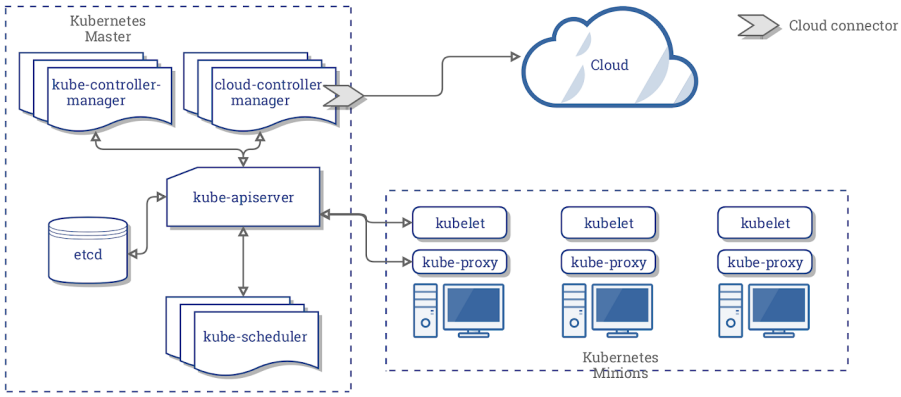
In a Linux environment, all the configurations are stored in the /etc directory.

etcd is inspired from that and there is an addition of d which is for distributed.



etcd is a distributed reliable key-value store.

etcd reliably stores the configuration data of the Kubernetes cluster, representing the state of the cluster (what nodes exist in the cluster, what pods should be running, which nodes they are running on, and a whole lot more) at any given point of time.



7.2 Important Pointers for ETCD

A Kubernetes cluster stores all its data in etcd.

Anything that you read while running kubectl get pods is stored in etcd

Any node crashing or process dying causes values in etcd to be changed.

Whenever you create something with kubectl create / kubectl run will create an entry in the etcd.

Module 8: K8s Component - kube-api server

API server acts as a gateway to the Kubernetes Cluster.

When you interact with your Kubernetes cluster using the kubectl command-line interface, you are actually communicating with the master API Server component.

The API Server is the only Kubernetes component that connects to etcd; all the other components must go through the API Server to work with the cluster state.

The API Server is also responsible for the authentication and authorization mechanism. All API clients should be authenticated in order to interact with the API Server.

8.1 Flow Diagram:

Kubectl writes to the API server (kubectl run mywebserver --image=nginx)

2. API server will authenticate and authorize. Upon validation, it will write it to etcd.

3. Upon write to etcd, API Server will invoke the scheduler.

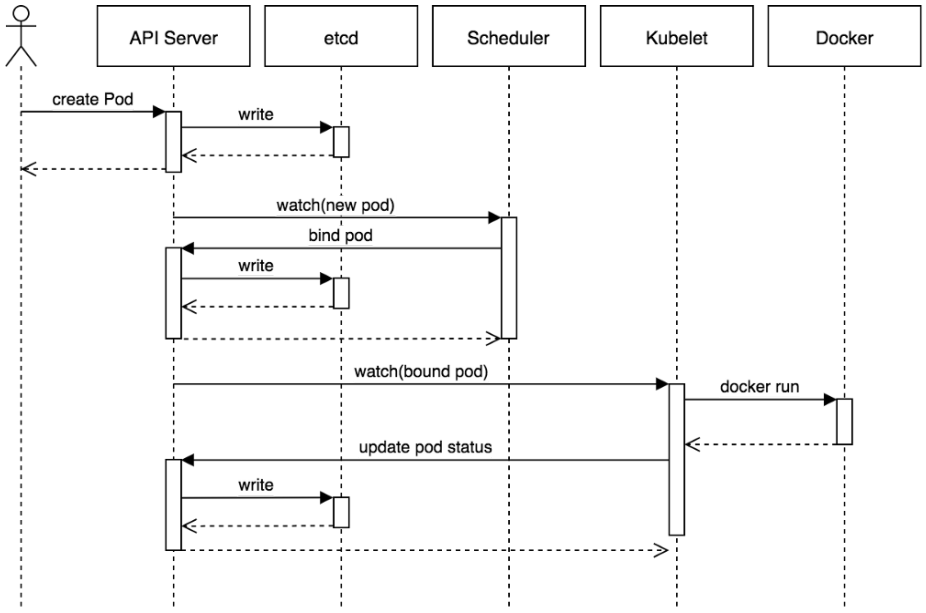
4. Scheduler decides which node the pod should run and return data to API server. API will in-turn write it back to etcd.

5. API Server will invoke the kubelet in the node decided by the scheduler.

6. Kubelet communicates to the docker daemon via Docker socket to create the container.

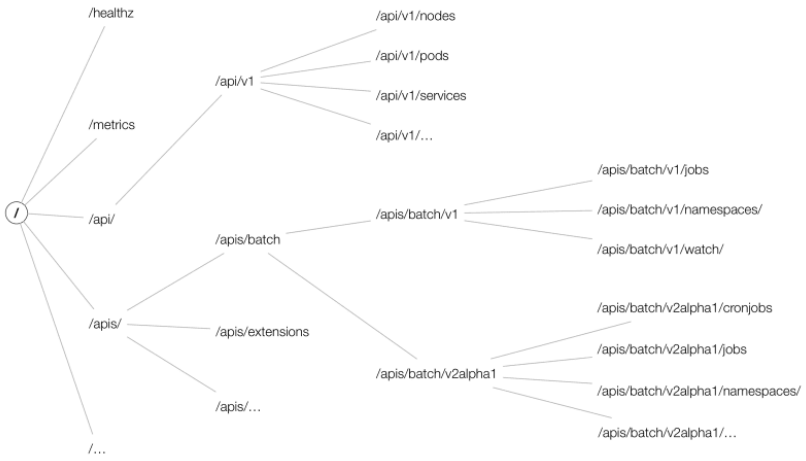
7. Kubelet will update the status of the POD back to the API server.

8. API Server will write the status details back to etcd.



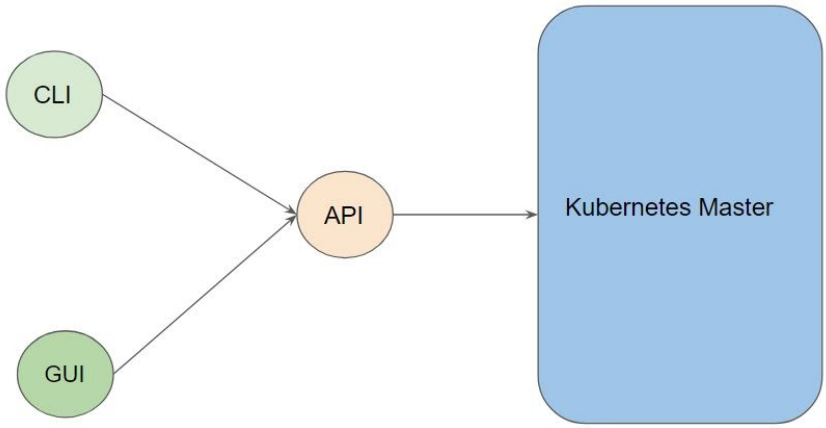
Module 9: Kubernetes API Primitives

Depending on the operation you intend to do, there are various APIs that are available.



Levels	Description
Alpha Level	• Version name contains alpha. • Example: v1alpha • May be buggy. • Enabling the feature may expose bugs. • Disabled by default.
Beta Level	• The version names contain beta (e.g. v2beta3). • Code is well tested. Enabling the feature is considered safe. Enabled by default. Recommended for only non-business-critical uses because of potential for incompatible changes in subsequent releases.
Stable Level	• The version name is vX where X is an integer. Example: v1 Recommended to use in production environments.

Following is a high-level architecture of a data flow in Kubernetes



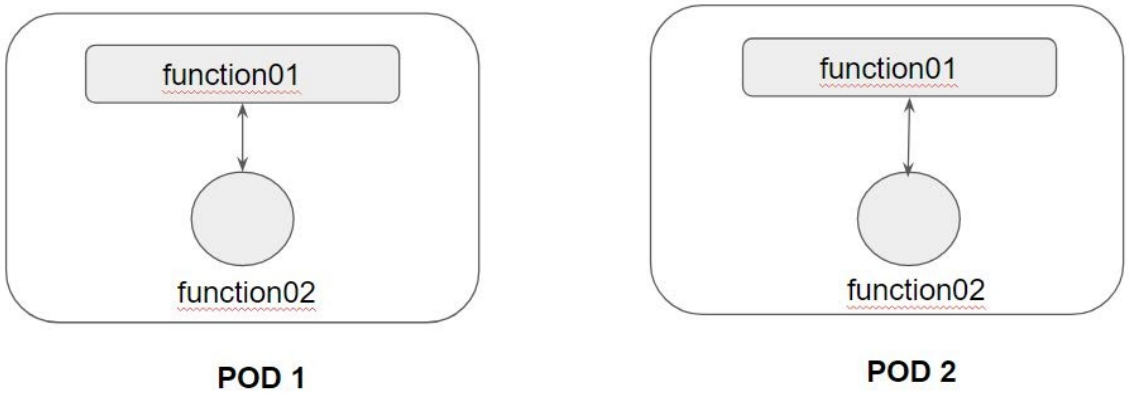
Module 10: Creating First POD Configuration in YAML

Key	Description
apiVersion	Version of API
kind	Kind of object you want to create.
metadata name	Name of object that uniquely identifies it.
spec	Desired state of the object.

Module 11: Multi-Container Pods

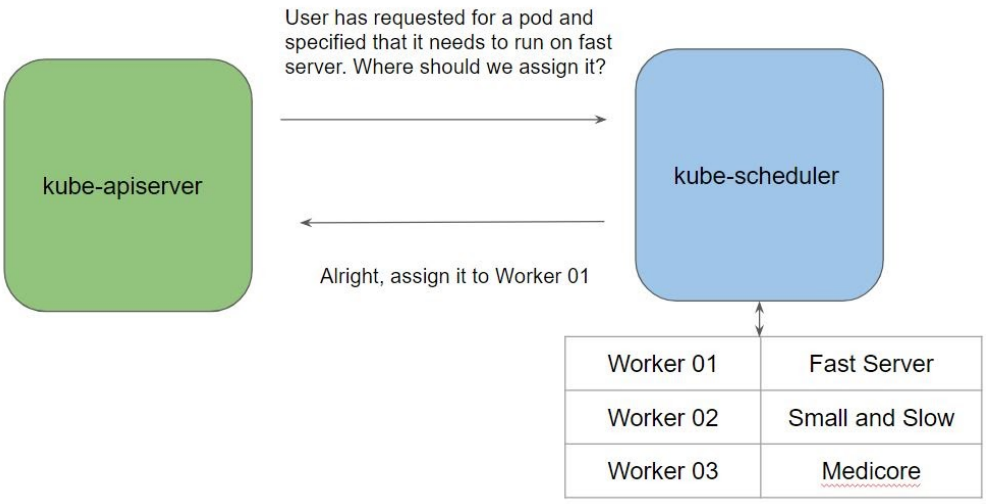
A single POD can have multiple containers as part of it. Till now we have been working based on a single container per pod.

Containers within a Pod share an IP address and port space and can find each other via localhost.



Module 12: K8s Component - kube-scheduler

Kube-scheduler watches for newly created pods that have no node assigned, and selects a node for them to run on.



There are several factors which are taken into consideration before a pod is scheduled to a node.

Some of these includes:

- Resource Requirements
- Hardware/Software policy constraints
- Affinity & Anti-Affinity
- Data Locality

Module 13: Revising Dockerfile - CMD vs ENTRYPOINT

The best use for ENTRYPOINT is to set the image's main command

ENTRYPOINT doesn't allow you to override the command.

It is important to understand distinction between CMD and ENTRYPOINT.

Module 14: Command and Arguments

During the video of ENTRYPOINT, we discussed the difference between CMD and ENTRYPOINT instruction in Dockerfile.

We can also refer them as Image Command and Image Entrypoint.

In Kubernetes, we can override the default entrypoint and CMD with command and args field.

Docker Field Name	Kubernetes Field name	Description
ENTRYPOINT	command	Command that will be run by container.
CMD	args	Argument passed to the container.

14.1 Use-Case 1: Image Entrypoint and CMD

When there is an entrypoint and CMD set for a Docker image and if we are not manually overriding it at k8s manifest level, then final decision is to use the default image specifications.

Image Entrypoint	Image Command	Container Command	Container Argument	Final
sleep	3600	<not set>	<not set>	sleep 3600

14.2 Use-Case 2: Setting Container Command

When there is an entrypoint and CMD set for a Docker image and if we are not manually overriding it at k8s manifest level, then final decision is to use the default image specifications.

Image Entrypoint	Image Command	Container Command	Container Argument	Final
sleep	3600	ping -c5 google.com	<not set>	ping -c5 google.com

14.3 Use-Case 3: Setting Container Arguments

When container argument is set in k8s manifest, the image command gets overridden.

Image Entrypoint	Image Command	Container Command	Container Argument	Final
sleep	3600	<not set>	5000	sleep 5000

14.4 Use-Case 4: Setting Container Command & Arguments

When container command and arguments are specified in k8s manifest, they will override the image command and entrypoint.

Image Entrypoint	Image Command	Container Command	Container Argument	Final
sleep	3600	ping	yahoo.com	ping yahoo.com

Module 15: CLI Documentation of K8s Resources

Till now we have been going through documentation from browser to understand about fields.

There is better way through which we can achieve similar functionality via CLI

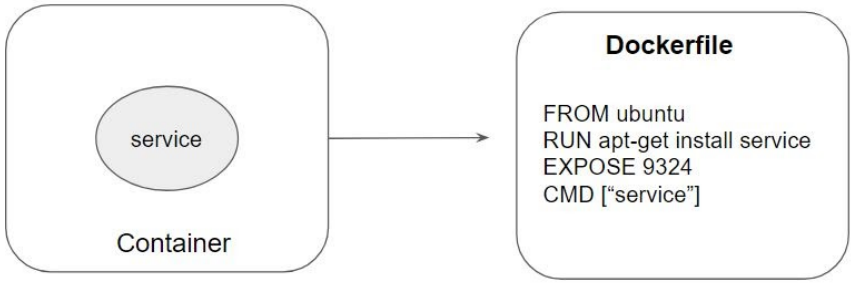
This is through kubectl explain command.

Module 16: Revising DockerFile - EXPOSE Instruction

The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.

The EXPOSE instruction does not actually publish the port.

It functions as a type of documentation between the person who builds the image and the person who runs the container, about which ports are intended to be published.



Join Our Discord Community

We invite you to join our Discord community, where you can interact with our support team for any course-based technical queries and connect with other students who are doing the same course.

Joining URL:

http://kplabs.in/chat