



KPLABS Course

Certified Kubernetes Application Developer 2022

Core Concepts

ISSUED BY

Zeal Vora

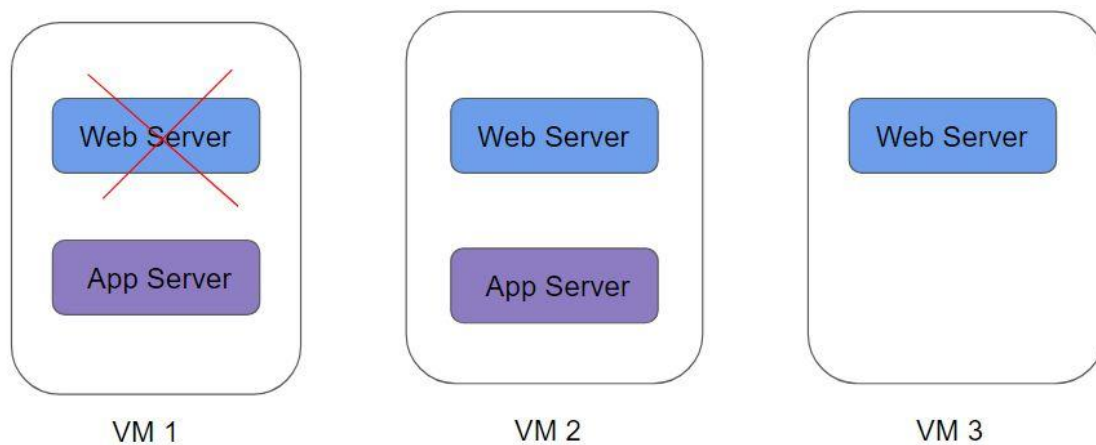
REPRESENTATIVE

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:

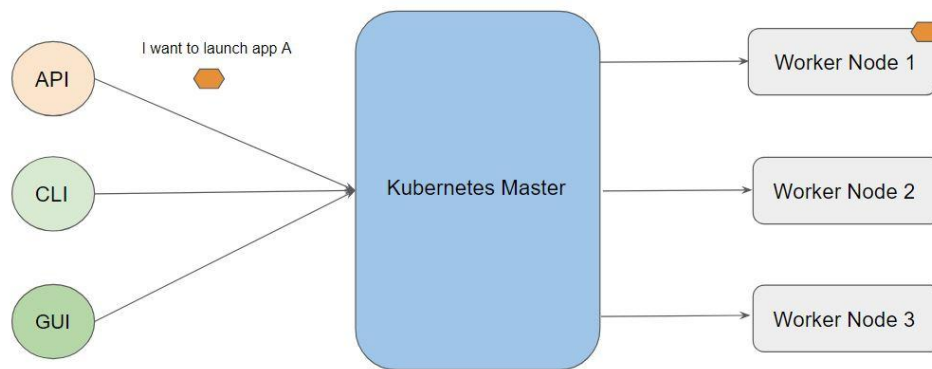
- Docker Swarm
- Kubernetes
- Apache Mesos
- Elastic Container Service (AWS ECS)

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

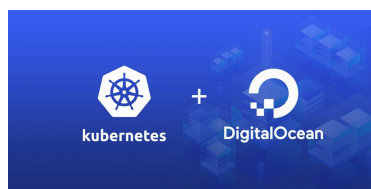
There are multiple ways to get started with a fully functional Kubernetes environment

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

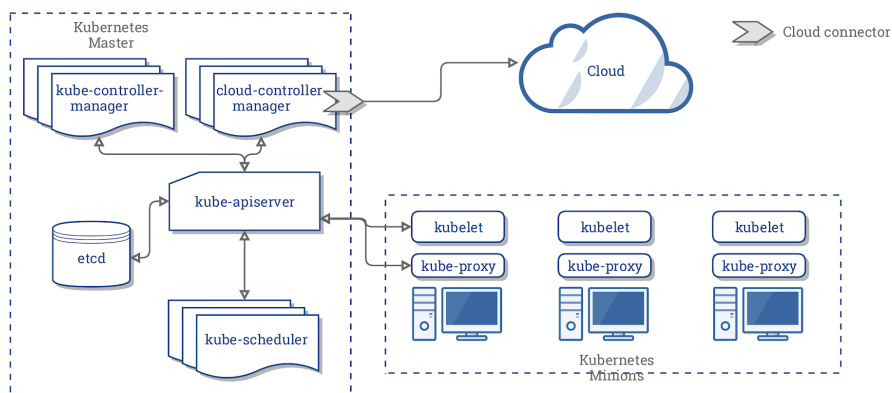
Minikube is a tool that makes it easy to run Kubernetes locally.

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.

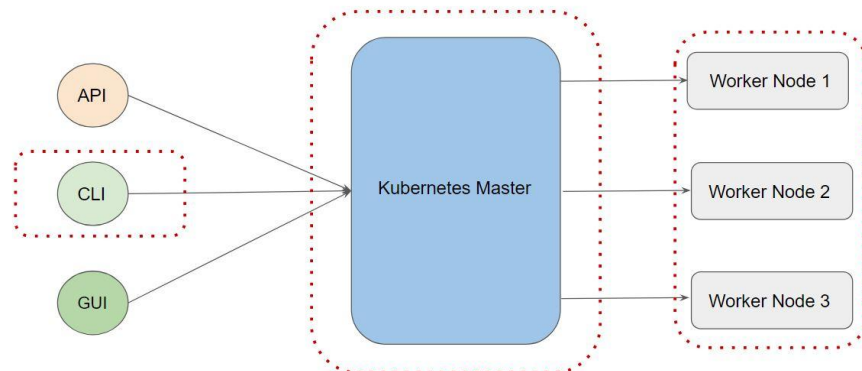


3.4 Installation Configuration:

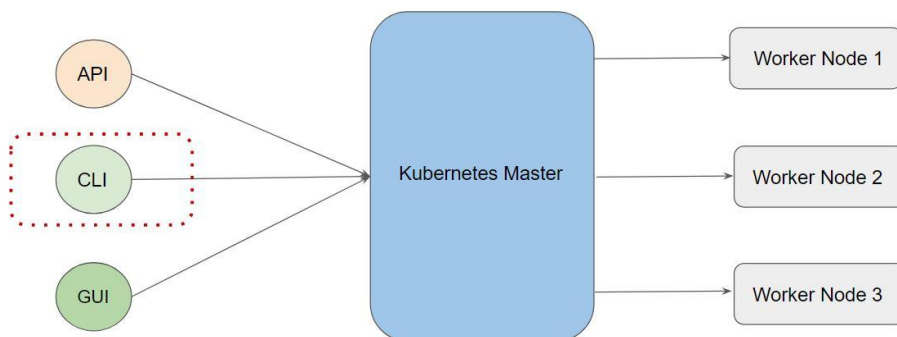
Things to configure while working with Kubernetes.

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

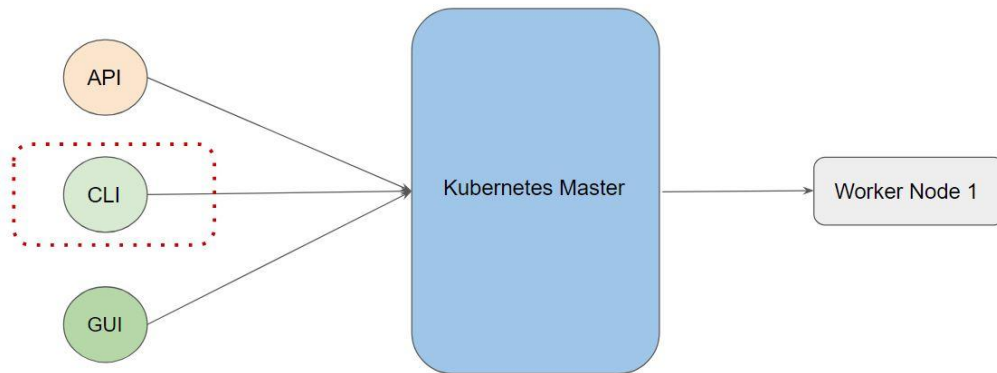
The following components highlighted in red are the ones that need to be configured while designing Kubernetes cluster in a hard way.



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



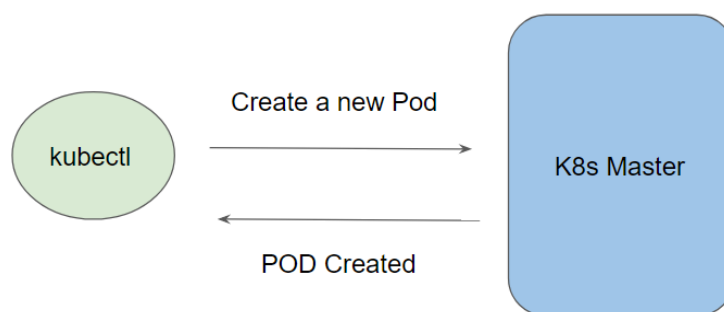
The following components highlighted in red are the ones that need to be configured while using minikube based installation method.



Module 4: Overview of kubectl

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters.

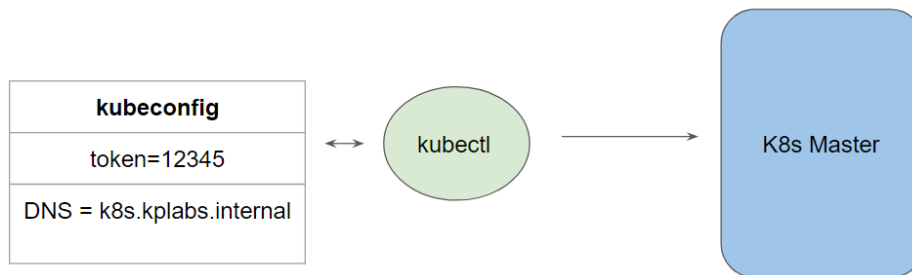
You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.



Let us understand the high-level workflow:

To connect to the Kubernetes Master, there are two important data which kubectl needs:

- DNS / IP of the Cluster
- Authentication Credentials



The overall installation of kubectl is straightforward and can be installed on a variety of Linux platforms, macOS and Windows

Install kubectl binary with curl on Windows [↗](#)

1. Download the [latest release v1.21.0](#).

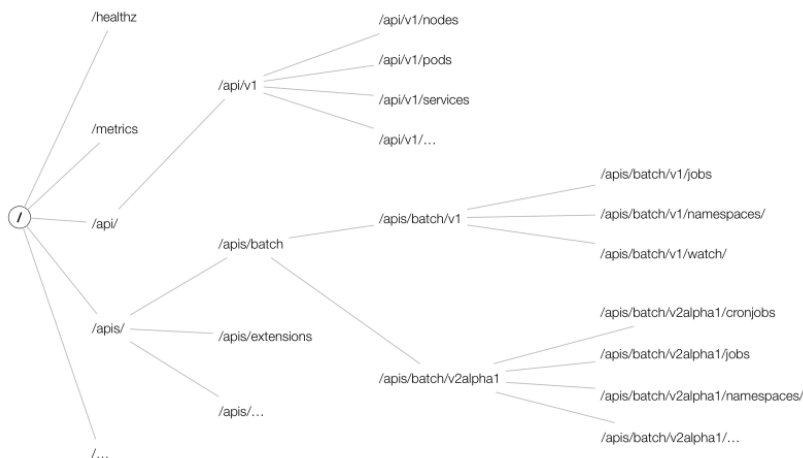
Or if you have `curl` installed, use this command:

```
curl -LO https://dl.k8s.io/release/v1.21.0/bin/windows/amd64/kubectl.exe
```

Note: To find out the latest stable version (for example, for scripting), take a look at <https://dl.k8s.io/release/stable.txt>.

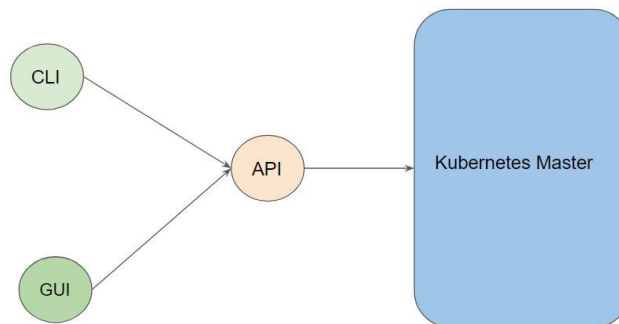
Module 5 Kubernetes API Primitives

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



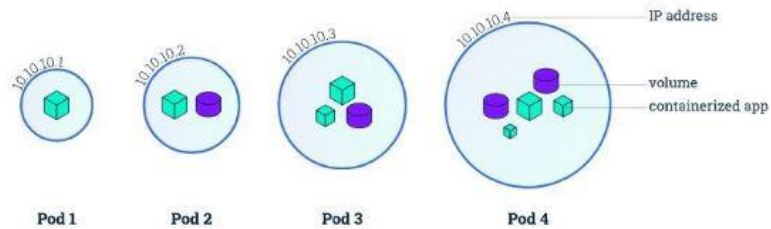
Levels	Description
Alpha Level	<ul style="list-style-type: none"> Version name contains alpha. Example: v1alpha May be buggy. Enabling the feature may expose bugs. Disabled by default.
Beta Level	<ul style="list-style-type: none"> The version names contain beta (e.g. v2beta3). Code is well tested. Enabling the feature is considered safe. Enabled by default. <p>Recommended for only non-business-critical uses because of potential for incompatible changes in subsequent releases.</p>
Stable Level	<ul style="list-style-type: none"> The version name is vX where X is an integer. <p>Example: v1 Recommended to use in production environments.</p>

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

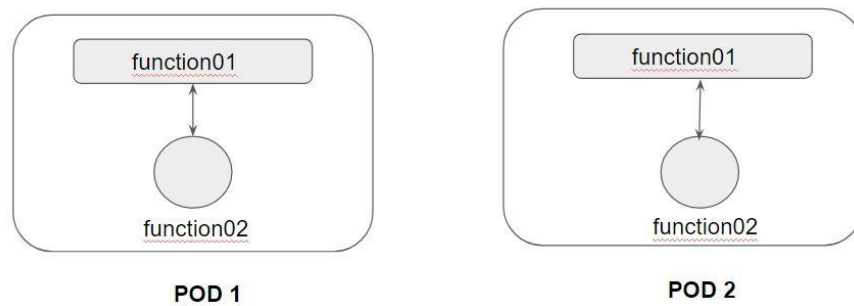


Module 6: PODS

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 7: 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
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: mywebserver
5  spec:
6    containers:
7    - name: mywebserver
8      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
<pre><Servers> <Server> <name>Server1</name> <owner>John</owner> <created>123456</created> <status>active</status> </Server> </Servers></pre>	<pre>{ Servers: [{ name: Server1, owner: John, created: 123456, status: active }] }</pre>	<pre>Servers: - name: Server1 owner: John created: 123456 status: active</pre>

Module 8: 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.



```
pod.yaml
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: mywebserver
5  spec:
6    containers:
7      - name: mywebserver
8        image: nginx
```

Module 9: 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 10: 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.

10.1 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

10.2 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

10.3 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

10.4 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 11: 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

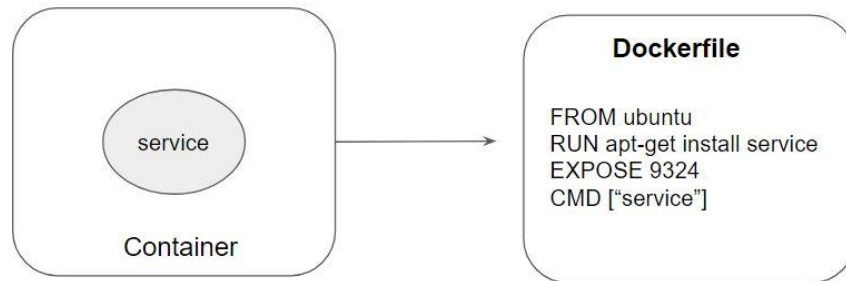
This is through the kubectl explain command.

Module 12: 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>

