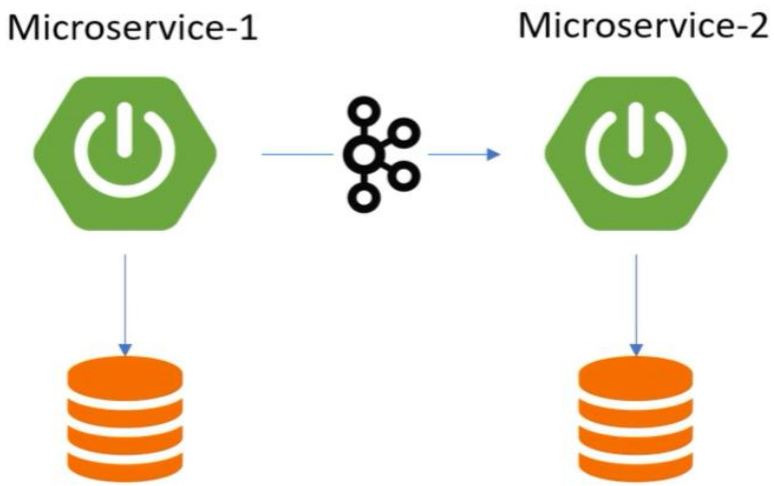
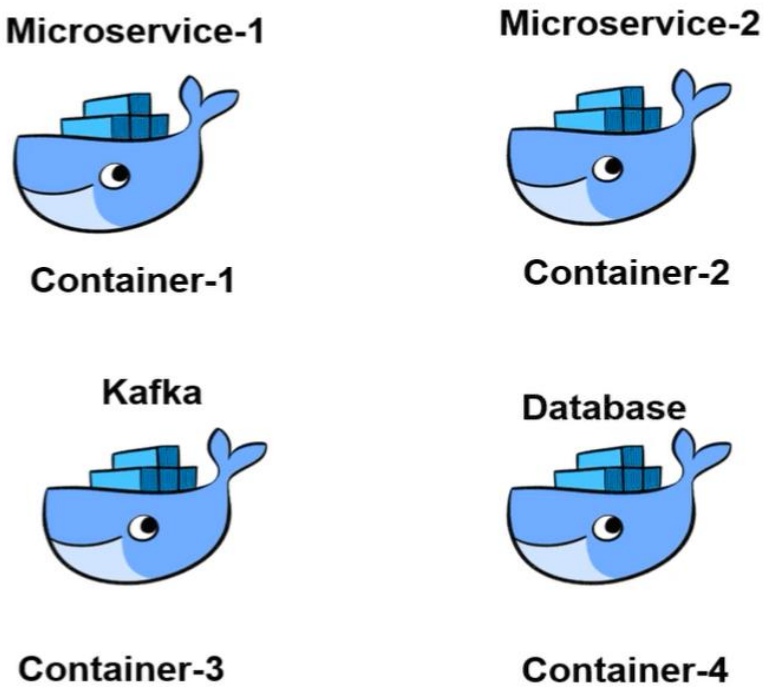
kubernetes

container orchestration platform

INTRODUCTION TO KUBERNETES





Dockerize microservice -1

Dockerize microservice - 2

Pull Kafka image Pull Database image

If it is just 4 containers, we can manage them, but if it is 4000?

Why Kubernetes?

- Picture a popular online store during a sale.
- Traffic spikes dramatically.
- If you're using Docker alone, you'd need to manually spin up new containers.
- Also you have to manage 100s of containers

- But with Kubernetes, it can automatically scale your application to handle the traffic, and once the rush is over, it can scale back down.

What is Kubernetes?

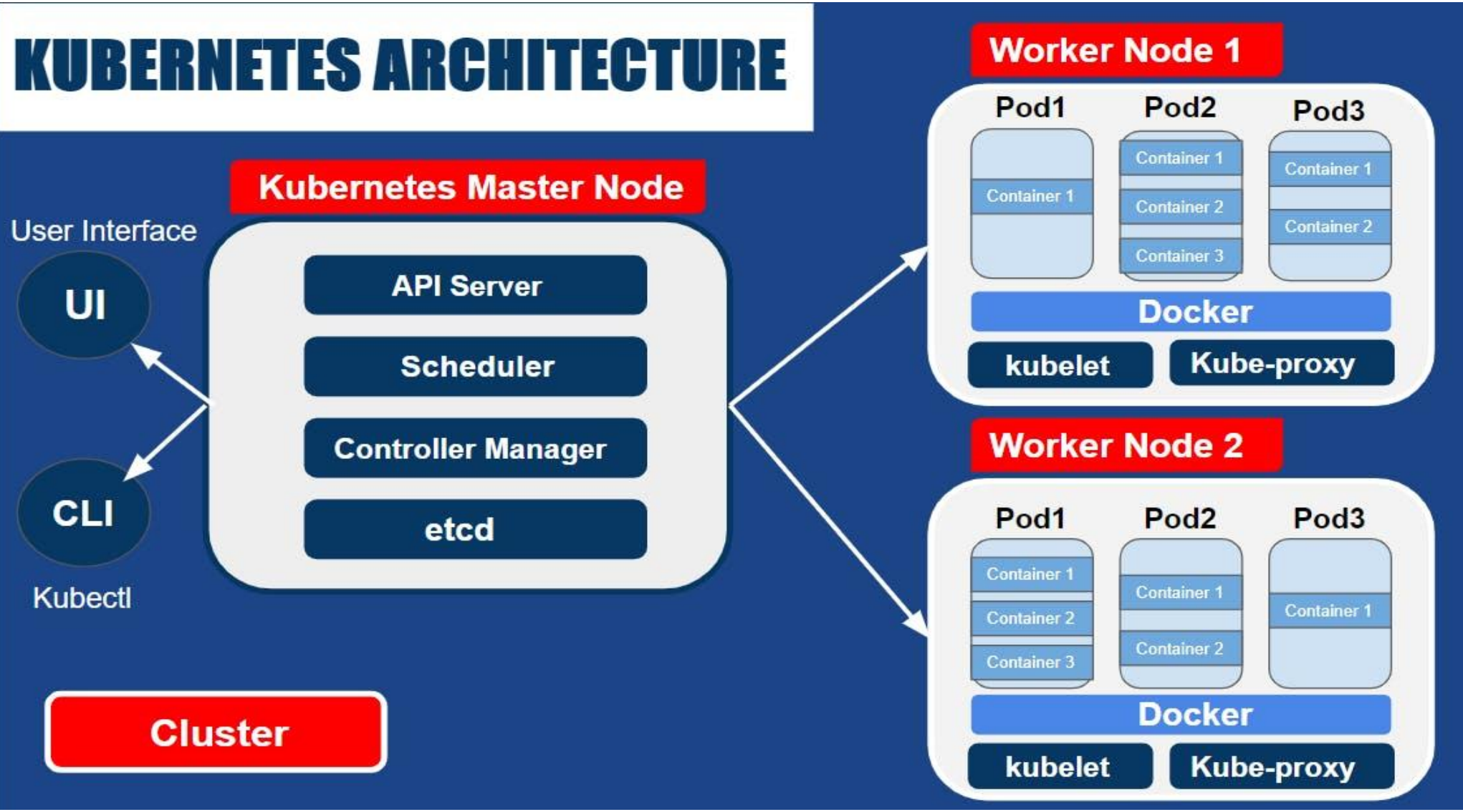
- Kubernetes (often abbreviated as K8s) is an open-source container orchestration platform that automates deploying, scaling, and managing containerized applications.
- Originally developed by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF).

Key Features

- **Self-healing**: Automatically replaces and reschedules failed containers and can kill unresponsive ones.
- **Horizontal** Scaling: Easily scale applications up or down.
- **Load Balancing**: Distributes traffic among the Pods.
- **Service Discovery**: Automatically assigns IP addresses and a single DNS name for a set of Pods.
- **Secrets and Configuration Management**: Manages sensitive information like passwords and API keys, as well as application configuration.
- **Rolling Updates and Rollbacks**: Facilitates seamless updates to applications without downtime and allows easy rollbacks if something goes wrong.

Kubernetes

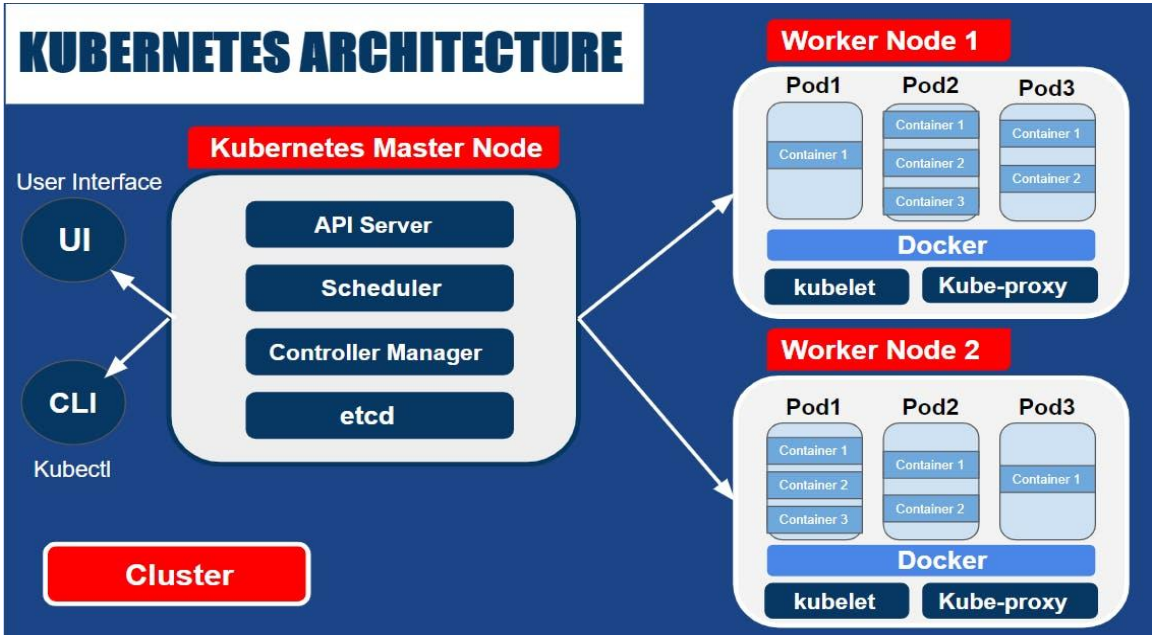
ARCHITECTURE



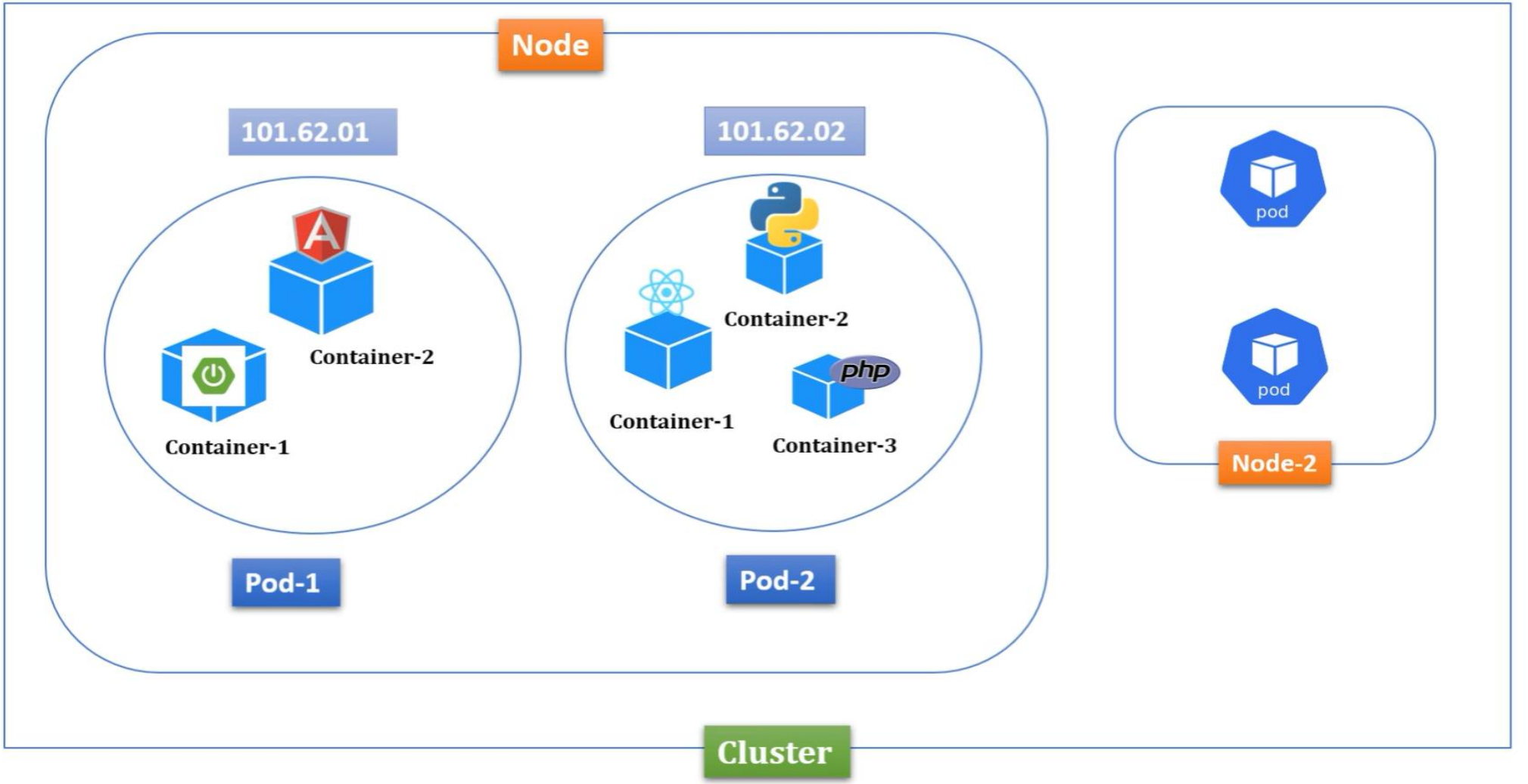
Here Docker is a container runtime

Key Components

- Cluster
- Master Node
- Node (Worker Node)
- Pod
- Deployment
- ReplicaSet
- Service
- kubelet
- API Server
- Scheduler
- Controller Manager
- etcd



Kubernetes Cluster



- **Cluster**
  - **What it is:** The collection of machines (nodes) managed by Kubernetes.
  - **Why it's needed:** It’s your working environment — you deploy everything into a cluster.

- **Master Node**
  - **What it is:** A special node running the **control plane components** like API Server, Scheduler, Controller Manager, and etcd.
  - **Why it's needed:** It manages and controls the cluster — makes all global decisions and maintains the desired state of the cluster.

- **Node(Worker Node)**
  - **What it is:** A worker machine (VM or physical) in the cluster.
  - **Why it's needed:** It runs your actual application pods (via containers).

- **Pod**
  - **What it is:** The **basic unit** of deployment in Kubernetes. It wraps one or more containers.
  - **Why it's needed:** All your apps run **inside pods**.

- **Deployment**
  - **What it is:** A higher-level object that manages pods.
  - **Why it's needed:** Automatically handles:
    - Creating pods
    - Updating them
    - Ensuring the desired number of pods are always running

- **ReplicaSet**
  - **What it is:** A controller that ensures a specific number of pod replicas are running at all times.
  - **Why it's needed:** Maintains the desired number of pods — if a pod dies, the ReplicaSet replaces it.
  - **Note:** It’s **usually created and managed automatically by a Deployment**

- **Service**
  - **What it is:** A network abstraction that exposes your pods.
  - **Why it's needed:** Without it, you can't access your app (even within the cluster).

- **kubelet *(runs on each node)***
  - **What it is:** Node agent that runs and manages pods.
  - **Why it's needed:** Talks to the API server and tells the node to start/stop pods.

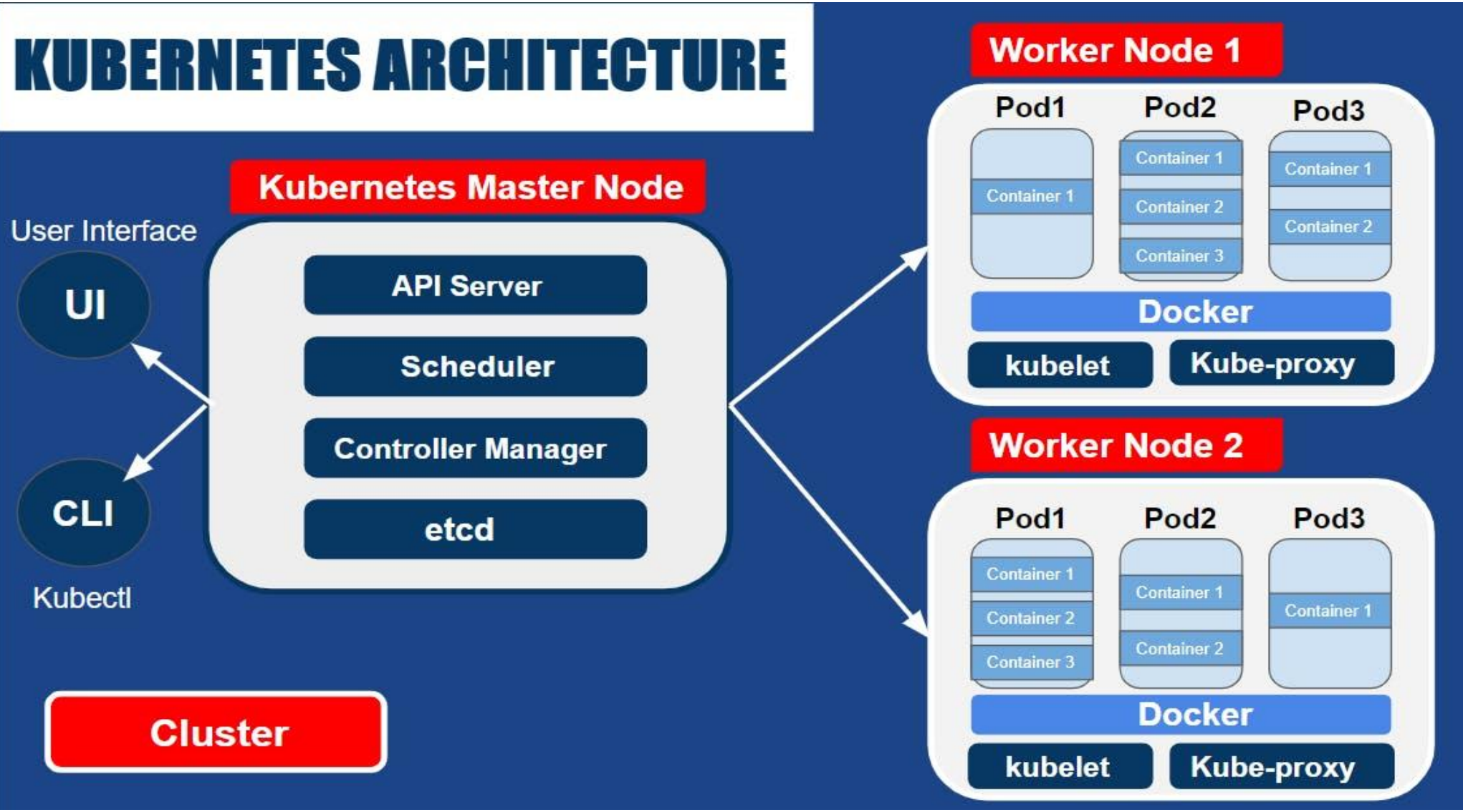
- **API Server**
  - **What it is:** The main entry point into Kubernetes.
  - **Why it's needed:** Every command (kubectl, dashboards, or internal control loops) talks to the API server.

- **Scheduler**
  - **What it is:** Assigns pods to available nodes.
  - **Why it's needed:** Decides **where** each pod should run.

- **Controller Manager**
  - **What it is:** Ensures the system maintains the desired state.
  - **Why it's needed:** Watches Deployments and ensures the correct number of pods are running.

- **etcd**
  - **What it is:** The cluster's **key-value database**.
  - **Why it's needed:** Stores the desired and current state of the cluster (Deployments, Services, etc.).

- **kube-proxy**
  - **What it is:** A network component that runs on each node and maintains network rules.
  - **Why it's needed:**
    - Enables **network communication** to and from pods.
    - Forwards traffic to the correct pod based on Kubernetes **Service** rules.
    - Implements **load balancing** for services across pod replicas.
    - Works with IPTables, IPVS, or eBPF depending on the setup.



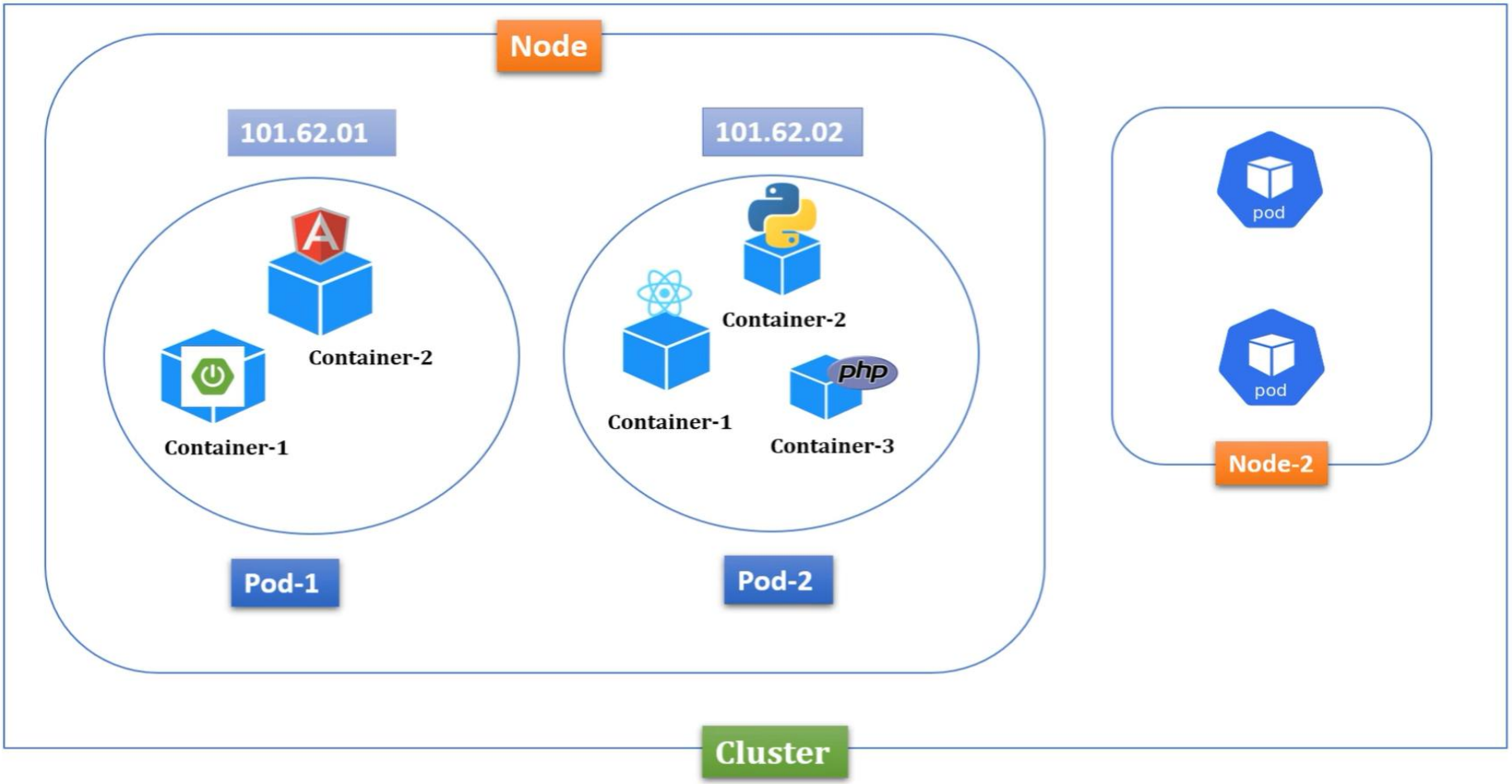
Here Docker is a container runtime

Kubernetes

SETUP

To run an application in the kubernetes, we need this architecture.

From where can we get the infrastructure

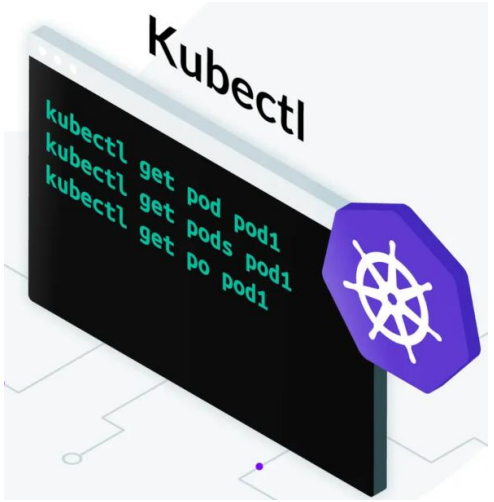


Minikube

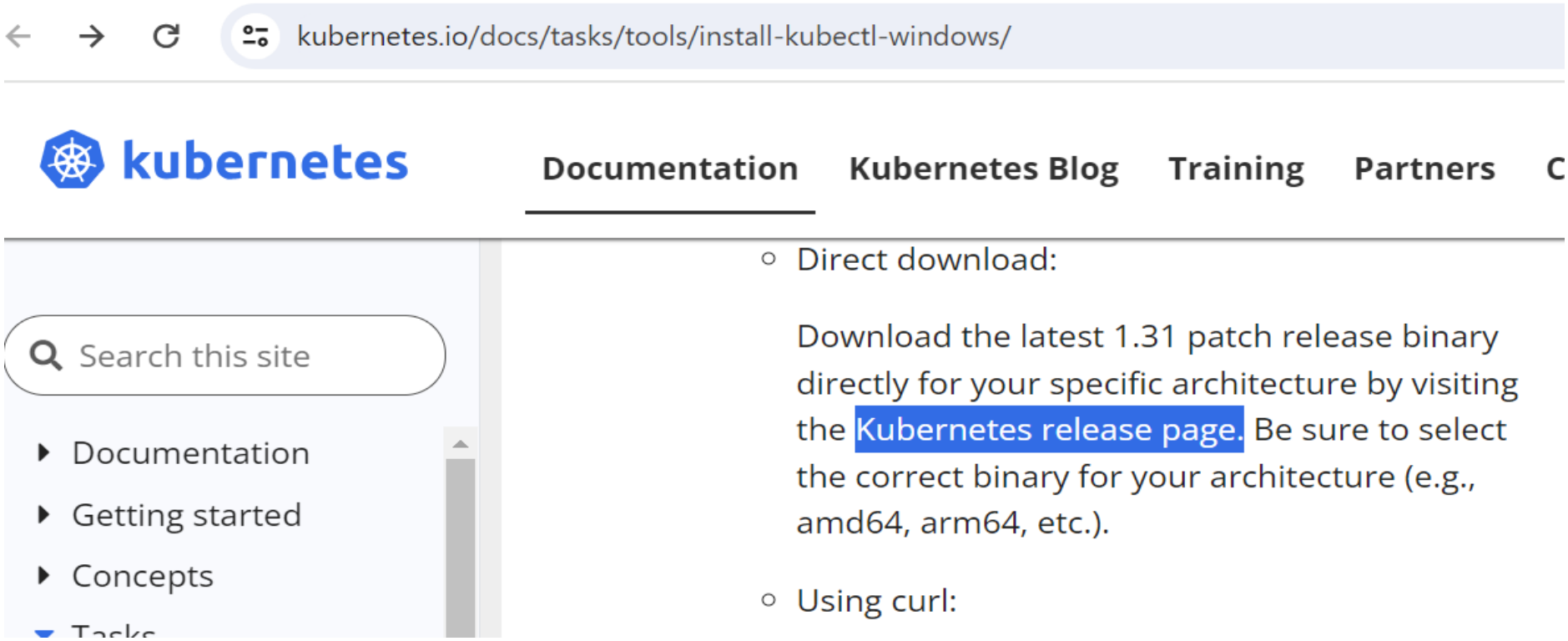
- **Minikube** is a tool that simplifies the process of running a Kubernetes cluster locally on your machine.

- It is particularly useful for developers who want to test Kubernetes applications without needing a full-fledged cluster in a cloud environment.

- **kubectl** is the command-line tool for interacting with Kubernetes clusters.



- Lets Download kubectl
- https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/



Download Kubectl

kubernetes.io/releases/download/#binaries

ernetes Documentation Kubernetes Blog Training Partners Community Cas

s site

es

s Release

ases

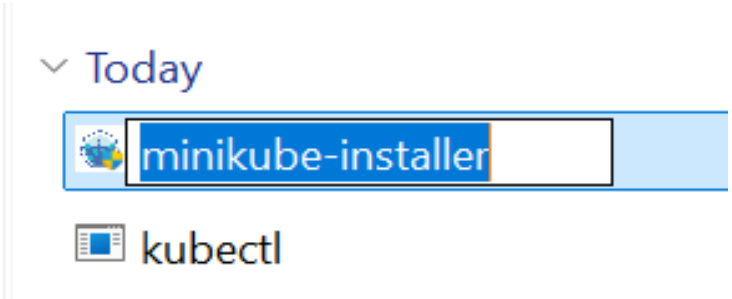
anagers

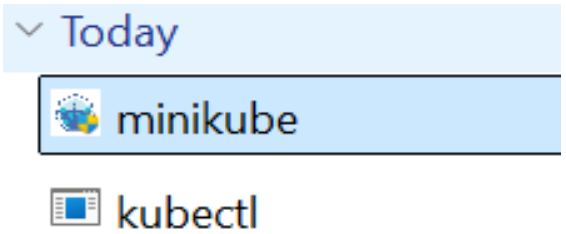
otes

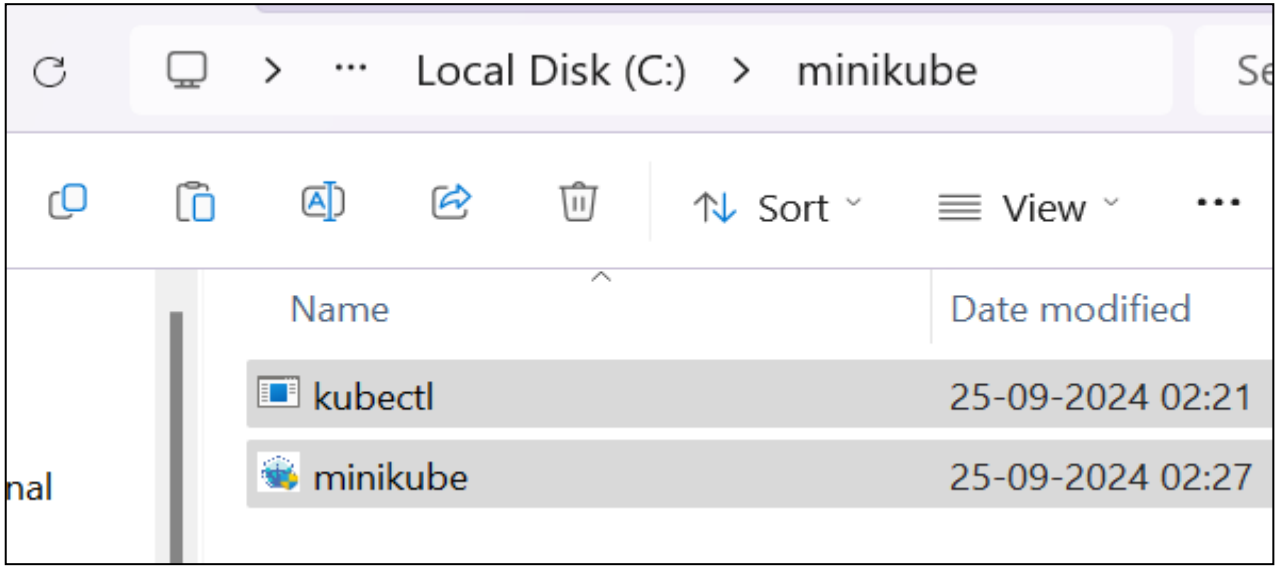
				dl.k8s.io/v1.31 (checksum | si
v1.31.1	windows	amd64	kubectl-convert.exe	dl.k8s.io/v1. convert.exe (ch
v1.31.1	windows	amd64	kubectl.exe	dl.k8s.io/v1.31 (checksum | si
v1.31.1	windows	amd64	kubelet.exe	dl.k8s.io/v1.31

Download minikube

Rename, organize & set path







Update path in environment variables

Install the minikube.exe

Confirm minikube installation

```
C:\WINDOWS\system32

Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ritit>minikube version
minikube version: v1.34.0
commit: 210b148df93a80eb872ecbeb7e35281b3c582c61

C:\Users\ritit>
```

To start the minikube

- Start the Docker desktop

```
C:\Users\ritit>minikube start --driver=docker
* minikube v1.34.0 on Microsoft Windows 11 Home 10.0.22631.4169 Build 22631.4169
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.45 ...
* Downloading Kubernetes v1.31.0 preload ...
    > gcr.io/k8s-minikube/kicbase...:  487.90 MiB / 487.90 MiB  100.00% 8.12 Mi
    > preloaded-images-k8s-v18-v1...:  326.69 MiB / 326.69 MiB  100.00% 3.47 Mi
* Creating docker container (CPUs=2, Memory=4000MB) ...
! Failing to connect to https://registry.k8s.io/ from inside the minikube container
* To pull new external images, you may need to configure a proxy: https://minikube.sigs
/proxy/
* Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by

C:\Users\ritit>
```

- Use the command to check the status
  - minikube status
- As per the documentation minikube will create a single node cluster for us to try application in local machine.
- To confirm that use the following commands
  - kubectl cluster-info
  - Kubectl get node

```
C:\Users\ritit>kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:58805
CoreDNS is running at https://127.0.0.1:58805/api/v1/namespaces

To further debug and diagnose cluster problems, use 'kubectl cl

C:\Users\ritit>kubectl get node
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   5m38s   v1.31.0
```

Kubernetes + springboot

DEPLOYMENT

Allow kebernetes to read docker Repo

Run cmd : minikube docker-env

Copy the last line and run it

```
C:\Users\ritit>minikube docker-env
SET DOCKER_TLS_VERIFY=1
SET DOCKER_HOST=tcp://127.0.0.1:60822
SET DOCKER_CERT_PATH=C:\Users\ritit\.minikube\certs
SET MINIKUBE_ACTIVE_DOCKERD=minikube
REM To point your shell to minikube's docker-daemon, run:
REM @FOR /f "tokens=*" %i IN ('minikube -p minikube docker-env --shell cmd') DO @%i

C:\Users\ritit>@FOR /f "tokens=*" %i IN ('minikube -p minikube docker-env --shell cmd') DO @%i

C:\Users\ritit>
```

Now kubernetes can talk to docker local repository

Try the cmd: docker images

It will list all the images of kube

The highlighted images are the part of Masternode

```
C:\Users\ritit>docker images
REPOSITORY
registry.k8s.io/kube-controller-manager
registry.k8s.io/kube-scheduler
registry.k8s.io/kube-apiserver
registry.k8s.io/kube-proxy
registry.k8s.io/etcd
registry.k8s.io/pause
registry.k8s.io/coredns/coredns
gcr.io/k8s-minikube/storage-provisioner

C:\Users\ritit>
```

Create a Spring Boot App

Generate jar : Run As -> Maven Install

Include a Dockerfile in the project root

Package Explorer

- kube-demo [boot]
  - src/main/java
    - com.rit
    - com.rit.controller
      - HomeController.java
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-17
  - Maven Dependencies
  - target/generated-sources/anr
  - target/generated-test-sources
  - src
  - target
  - Dockerfile
  - HELP.md
  - mvnw

HomeController.java

```java
@RestController
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "Welcome to kubernetes deployed App !"
    }
}
```

Dockerfile

```dockerfile
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "app.jar" ]
```

Build a docker image

In terminal move to the project directory

cmd: docker build –t <image-name> <path-to-Dockerfile>

ex: docker build –t kube-demo .

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>docker build -t kube-demo:1.0 .
[+] Building 27.4s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 158B
 => [internal] load metadata for docker.io/library/openjdk:17-jdk-slim
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load .dockerignore
```

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>docker images
REPOSITORY                                TAG        IMAGE ID       CREATED
kube-demo                                 1.0        26200abbae4e   2 minutes ago
registry.k8s.io/kube-scheduler            v1.31.0    1766f54c897f   6 weeks ago
registry.k8s.io/kube-controller-manager   v1.31.0    045733566833   6 weeks ago
registry.k8s.io/kube-apiserver            v1.31.0    604f5db92eaa   6 weeks ago
registry.k8s.io/kube-proxy                v1.31.0    ad83b2ca7b09   6 weeks ago
```

We got the image and now we want to run it inside the pod

For that we need to create a deployment object

Check Docker Images

- It should list kubernetes images as well as the newly created application images
- If the application is not listed then load it using minikube
  - minikube image load <image:tag>

```
E:\Programs\SpringMicroservices\SBootKube>minikube image load sbkubeimg:1.0

E:\Programs\SpringMicroservices\SBootKube>docker images
REPOSITORY                                  TAG        IMAGE ID       CREATED
sbkubeimg                                   1.0        c28a864db38f   16 minutes a
registry.k8s.io/kube-controller-manager     v1.31.0    045733566833   11 months ag
registry.k8s.io/kube-scheduler              v1.31.0    1766f54c897f   11 months ag
registry.k8s.io/kube-apiserver              v1.31.0    604f5db92eaa   11 months ag
registry.k8s.io/kube-proxy                  v1.31.0    ad83b2ca7b09   11 months ag
```

Deployment Object

deployments are kubernetes object,
that are used for managing the pods
we can describe deployment object utils using
command prompt or yaml files

Using Command prompt

kubectl create deployment <deployment-name>
--image =<image-name:tag> --port=<port-no>

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl create
deployment kube-demo-k8s --image=kube-demo:1.0 --port=8080
deployment.apps/kube-demo-k8s created

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>
```

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get deployment
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
kube-demo-k8s   1/1     1            1           71s
```

Get Deployment Objects and details

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
kube-demo-k8s   1/1     1            1           32m
```

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl describe deployment kube-demo-k8s
Name:                   kube-demo-k8s
Namespace:              default
CreationTimestamp:      Wed, 25 Sep 2024 12:52:22 +0530
Labels:                 app=kube-demo-k8s
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=kube-demo-k8s
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kube-demo-k8s
  Containers:
   kube-demo:
    Image:         kube-demo:1.0
    Port:          8080/TCP
    Host Port:     0/TCP
```

Now the application is deployed in kebernetes

To check the status of pods

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
kube-demo-k8s-66f4fd6fc-fhpsw     1/1     Running   0          14m
```

- It means
  - Kubernetes is able to pull the spring boot image from docker
  - It created a pod and started a container and executed the image
- To check the spring boot application running inside the pod
  - Cmd: kubectl logs <pod-name>

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl logs kube-demo-k8s-66f4fd6fc-fhpsw

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.4)

2024-09-25T07:22:23.883Z  INFO 1 --- [kube-demo] [           main] com.rit.KubeDemoApplication
         : Starting KubeDemoApplication v0.0.1-SNAPSHOT using Java 17.0.2 with PID 1 (/app/app.j
started by root in /app)
```

To access the url outside the cluster

We need to expose the deployment

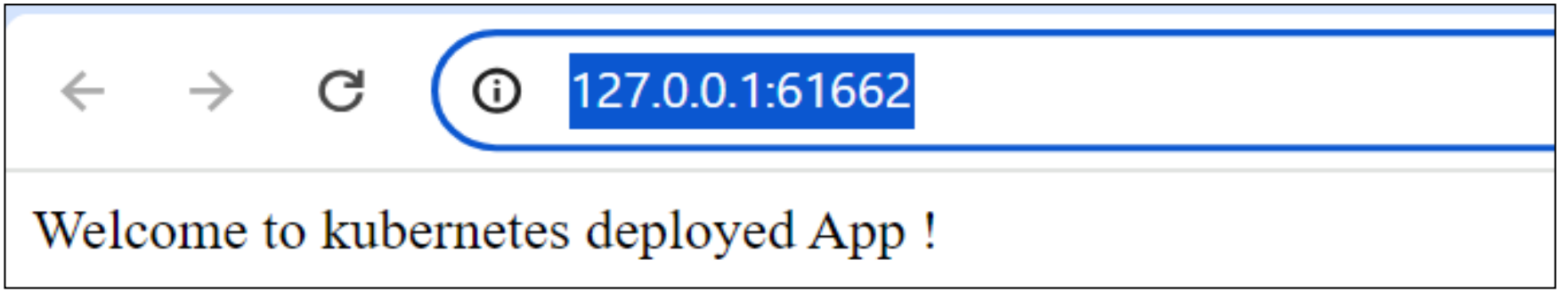
- Get the deployments
  - kubectl get deployments
- Expose the deployment to service
  - kubectl expose deployment <deployment-name> --type=NodePort
  - kubectl expose deployment kube-demo-k8s --type=NodePort

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get service
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kube-demo-k8s   NodePort    10.100.207.154   <none>        8080:31937/TCP   4m47s
kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP          10h
```

All the traffic comes to this service and it load balances the request with pods.

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>minikube service kube-demo-k8s --url
http://127.0.0.1:61662
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

Access the url in the browser



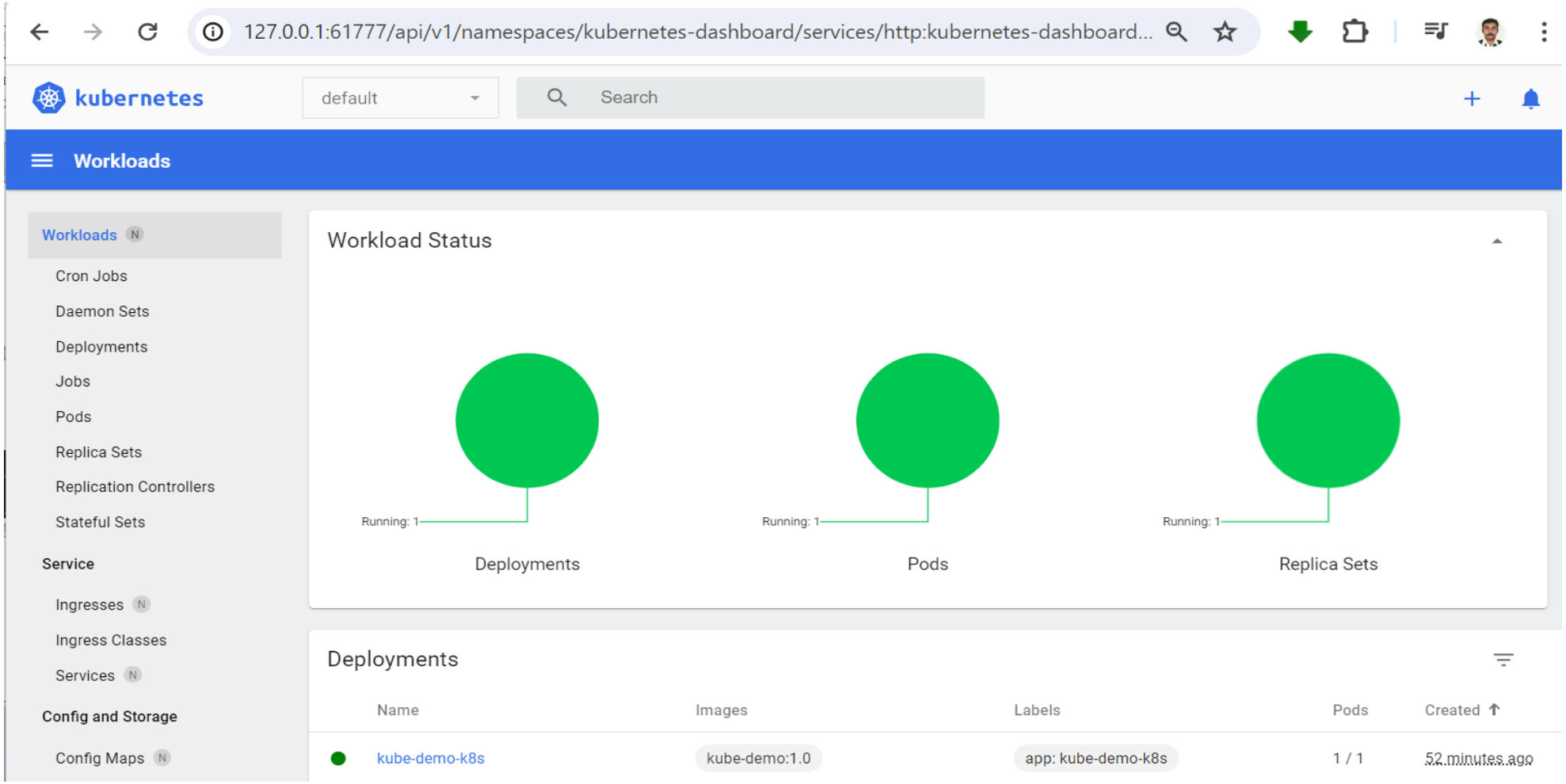
To check the health status

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>minikube dashboard
* Enabling dashboard ...
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all feat

        minikube addons enable metrics-server

* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:61777/api/v1/namespaces/kubernetes-dashboard/servic
```

Minikube Dashboard



Self Healing

If we delete a pod, it will get recreate

Pods

	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
●	kube-demo-k8s-66f4fd6fc-rh496	kube-demo:1.0	app: kube-demo-k8s pod-template-hash: 66f4fd6fc	minikube	Running	0	-	-	10 seconds ago

Lets cleanup all the objects

- Delete the service
  - kubectl delete service kube-demo-k8s
- Delete the deployment
  - kubectl delete deployment kube-demo-k8s
- To verify the pods
  - kubectl get pods
- To verify the service
  - kubectl get svc
- To verify the deployments
  - kubectl get deployments
- To stop the minikube
  - minikube stop
- To verify the nodes
  - kubectl get nodes
- To delete the local kubernetes cluster and minikube
  - minikube delete

Lets cleanup all the objects

- Delete the service
  - kubectl delete service kube-demo-k8s
- Delete the deployment
  - kubectl delete deployment kube-demo-k8s
- To verify the pods
  - kubectl get pods
- To verify the service
  - kubectl get svc
- To verify the deployments
  - kubectl get deployments
- To stop the minikube
  - minikube stop
- To verify the nodes
  - kubectl get nodes
- To delete the local kubernetes cluster and minikube
  - minikube delete

Lets cleanup all the objects

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl delete service kube-demo-k8s
service "kube-demo-k8s" deleted

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl delete deployment kube-demo-k8s
deployment.apps "kube-demo-k8s" deleted

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get pods
No resources found in default namespace.

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get svc
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   11h

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get deployments
No resources found in default namespace.

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>minikube stop
* Stopping node "minikube"  ...
* Powering off "minikube" via SSH ...
* 1 node stopped.

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>kubectl get nodes
E0925 13:56:37.098301   12608 memcache.go:265] couldn't get current server API group list: Ge
```

```
E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>minikube delete
* Deleting "minikube" in docker ...
* Deleting container "minikube" ...
* Removing C:\Users\ritit\.minikube\machines\minikube ...
* Removed all traces of the "minikube" cluster.

E:\Course PPT\DevOps\Kubernetes\workspace\kube-demo>
```