Name: Om Sanjay Bhavsar Roll

No.: 331069

PRN: 22120084

Batch: A3

Assignment No 8

Aim: Setup Single Node Kubernetes Cluster with Minikube and Deploy an web app on Kubernetes cluster.

Theory:

- **What is Kubernetes?**

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetesservices,support,andtoolsarewidelyavailable.

The name Kubernetes originates from Greek, meaning helmsman or pilot. Google open- sourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale withbest-of-breedideasand practices from the community

- **Why you need Kubernetes and what it can do**

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another containerneedstostart.Wouldn'tit be easier if this behavior was handled by a system?

That's how Kubernetes comes to the rescue! Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. For example, Kubernetes can easily manage a canary deployment for your system.

Kubernetes provides you with:

- **Service discovery and load balancing**
- **Storage orchestration**
- **Automated rollouts and rollbacks**
- **Automatic bin packing**
- **Self-healing**
- **Secret and configuration management**

- **Kubernetes Components**

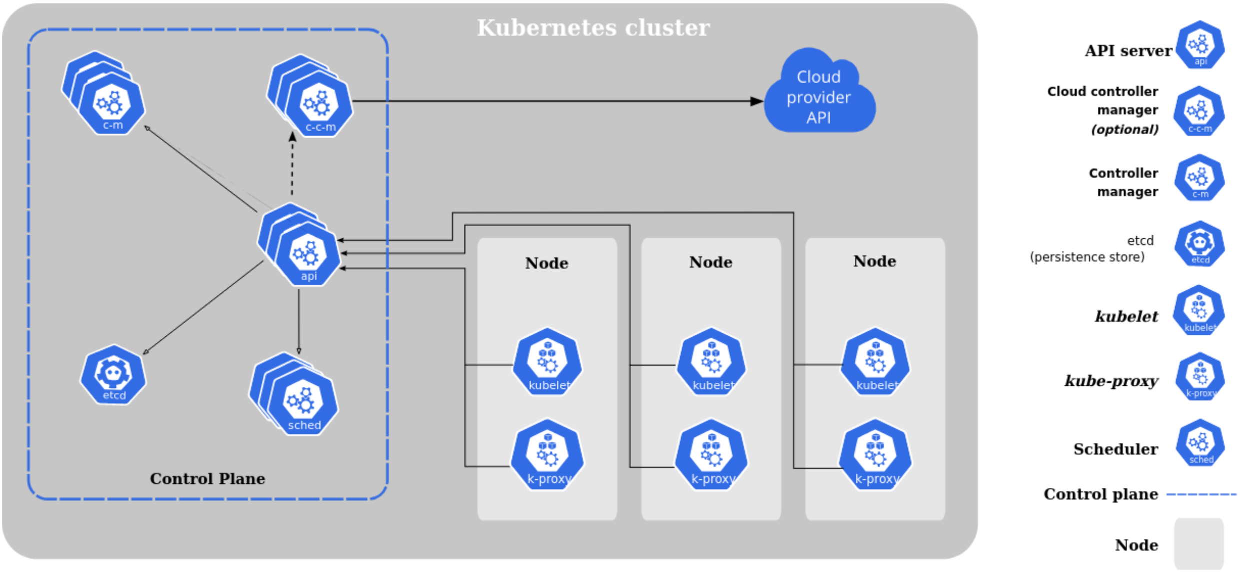
When you deploy Kubernetes, you get a cluster.

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.

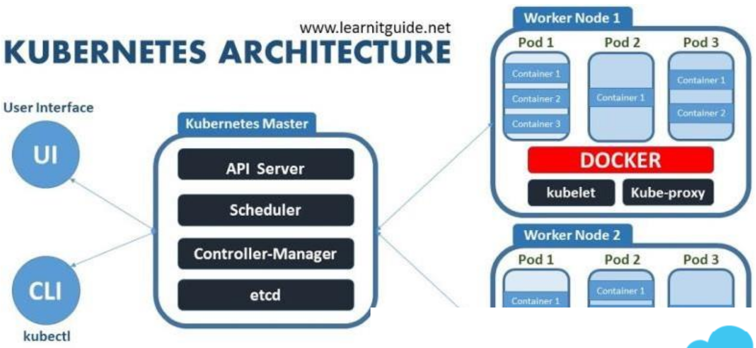
The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

This document outlines the various components you need to have a complete and working Kubernetes cluster.

Here's the diagram of a Kubernetes cluster with all the components tied together.

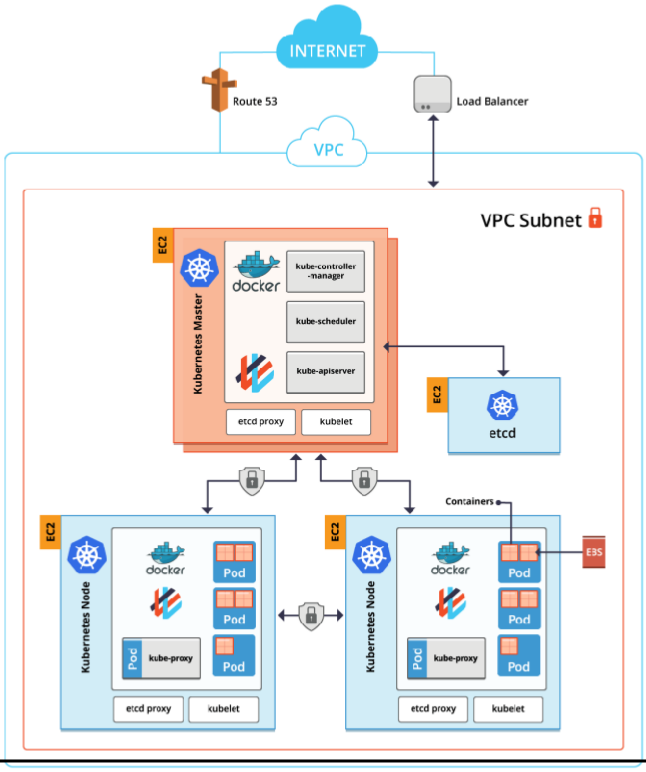


➢ **Kubernetes Architecture**



Kubernetes

Architecture on AWS:



➢ Control Plane Components

The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas f ield is unsatisfied).

Control plane components can be run on any machine in the cluster. However, for simplicity, set up scripts typically start all control plane components on the same machine, and do not run user containers on this machine. See Building High-Availability Clusters for an example multi- master-VM setup.

1. kube-apiserver

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver. kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

2. etcd

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data.

If your Kubernetes cluster uses etcd as its backing store, make sure you have a back up plan for those data.

You can f ind in-depth information about etcd in the official documentation.

3. kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

4. kube-controller-manager

Control Plane component that runs controller processes.

Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

These controllers include:

- Node controller: Responsible for noticing and responding when nodes go down.
- Replication controller: Responsible for maintaining the correct number of pods for every replication controller object in the system.
- Endpoints controller: Populates the Endpoints object (that is, joins Services & Pods).
- Service Account & Token controllers: Create default accounts and API access tokens for new namespaces.

5. cloud-controller-manager

A Kubernetes control plane component that embeds cloud-specific control logic. The cloud controller manager lets you link your cluster into your cloud provider's API, and separates out

the components that interact with that cloud platform from components that just interact with your cluster.
The cloud-controller-manager only runs controllers that are specific to your cloud provider. If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.

As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that you run as a single process. You can scale horizontally (run more than one copy) to improve performance or to help tolerate failures.

The following controllers can have cloud provider dependencies:

- Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
- Route controller: For setting up routes in the underlying cloud infrastructure
- Service controller: For creating, updating and deleting cloud provider load balancers

➢ Node Components

Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

1. kubelet

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

2. kube-proxy

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

3. Container runtime

The container runtime is the software that is responsible for running containers.

Kubernetes supports several container runtimes: Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

➢ Pods

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage/network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host.

➢ Using Pods

Usually you don't need to create Pods directly, even singleton Pods. Instead, create them using workload resources such as Deployment or Job. If your Pods need to track state, consider the StatefulSet resource.

Pods in a Kubernetes cluster are used in two main ways:

- **Pods that run a single container**. The "one-container-per-Pod" model is the most common Kubernetes use case; in this case, you can think of a Pod as a wrapper around a single container; Kubernetes manages Pods rather than managing the containers directly.
- **Pods that run multiple containers that need to work together** . A Pod can encapsulate an application composed of multiple co-located containers that are tightly coupled and need to share resources. These co-located containers form a single cohesive unit of service—for example, one container serving data stored in a shared volume to the public, while a separate *sidecar* container refreshes or updates those files. The Pod wraps these containers, storage resources, and an ephemeral network identity together as a single unit.

➢ Service

An abstract way to expose an application running on a set of Pods as a network service.

With Kubernetes you don't need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.

➢ Ingress

FEATURE STATE: Kubernetes v1.19 [stable]

An API object that manages external access to the services in a cluster, typically HTTP. Ingress may provide load balancing, SSL termination and name-based virtual hosting.

➢ Terminology

For clarity, this guide defines the following terms:

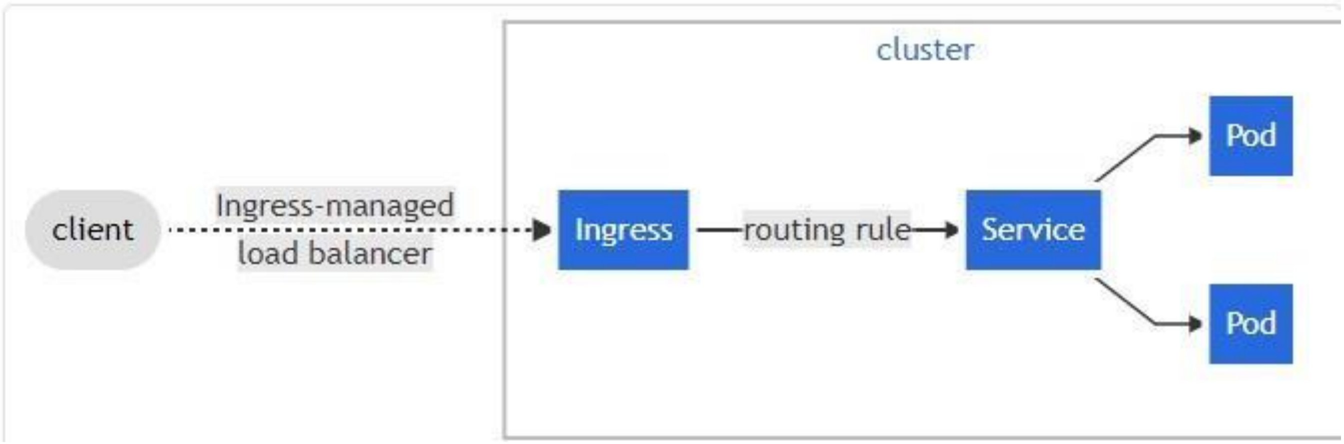
- Node: A worker machine in Kubernetes, part of a cluster.
- Cluster: A set of Nodes that run containerized applications managed by Kubernetes. For this example, and in most common Kubernetes deployments, nodes in the cluster are not part of the public internet.
- Edge router: A router that enforces the firewall policy for your cluster. This could be a gateway managed by a cloud provider or a physical piece of hardware.
- Cluster network: A set of links, logical or physical, that facilitate communication within a cluster according to the Kubernetes networking model.

- Service: A Kubernetes Service that identifies a set of Pods using label selectors. Unless mentioned otherwise, Servicesare assumedto have virtual IPs only routable within the cluster network.

➢ What is Ingress?

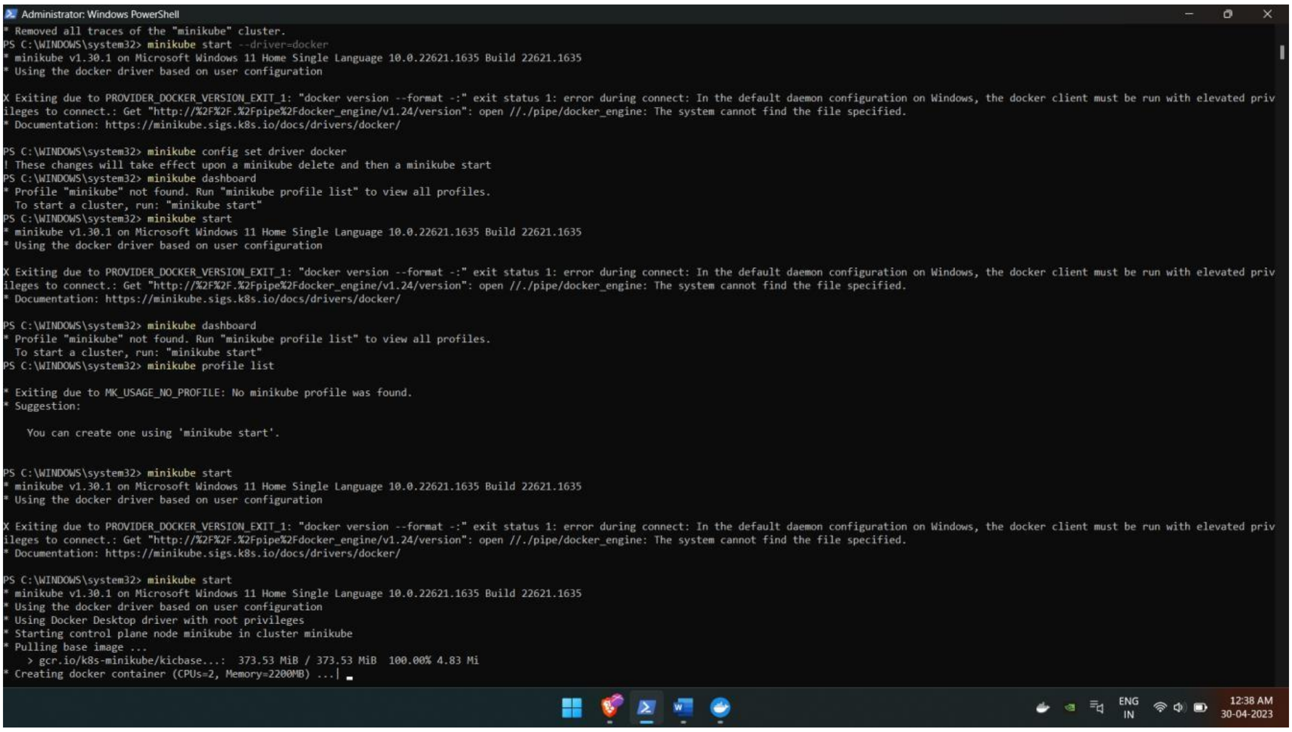
Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.

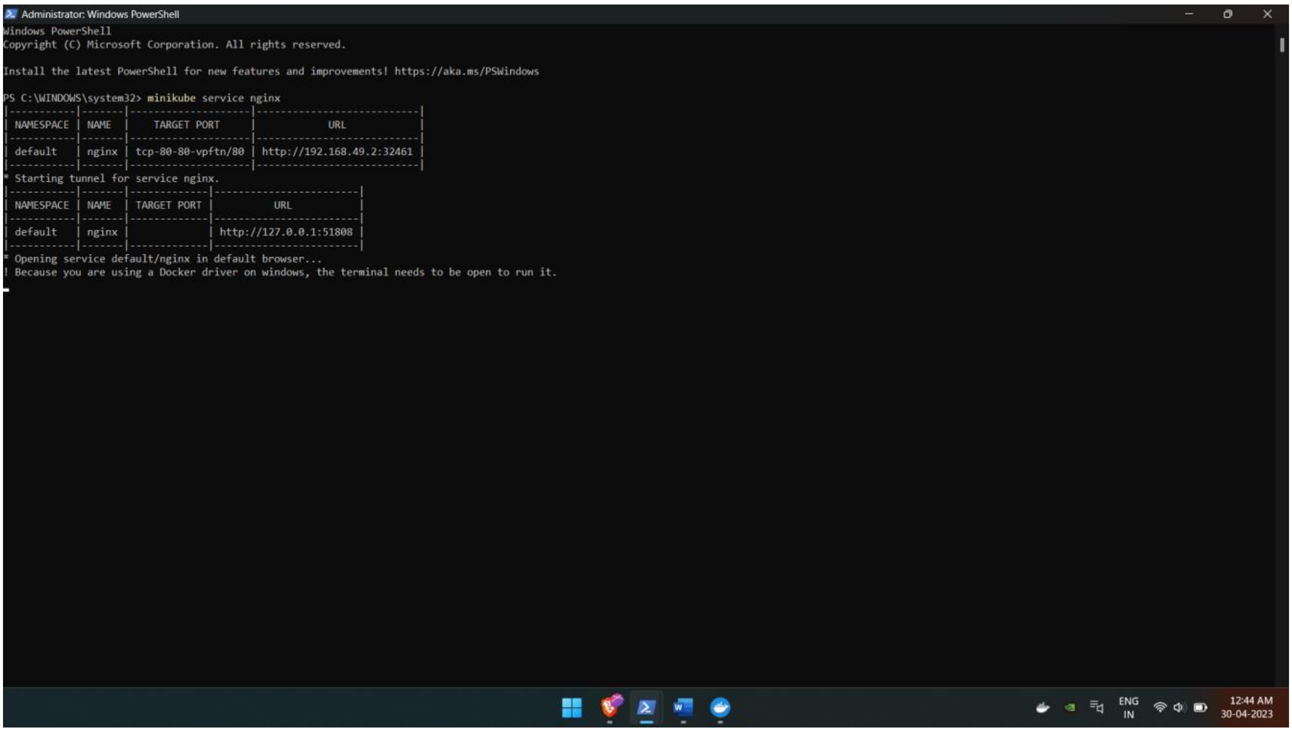
Here is a simple example where an Ingress sends all its traffic to one Service:

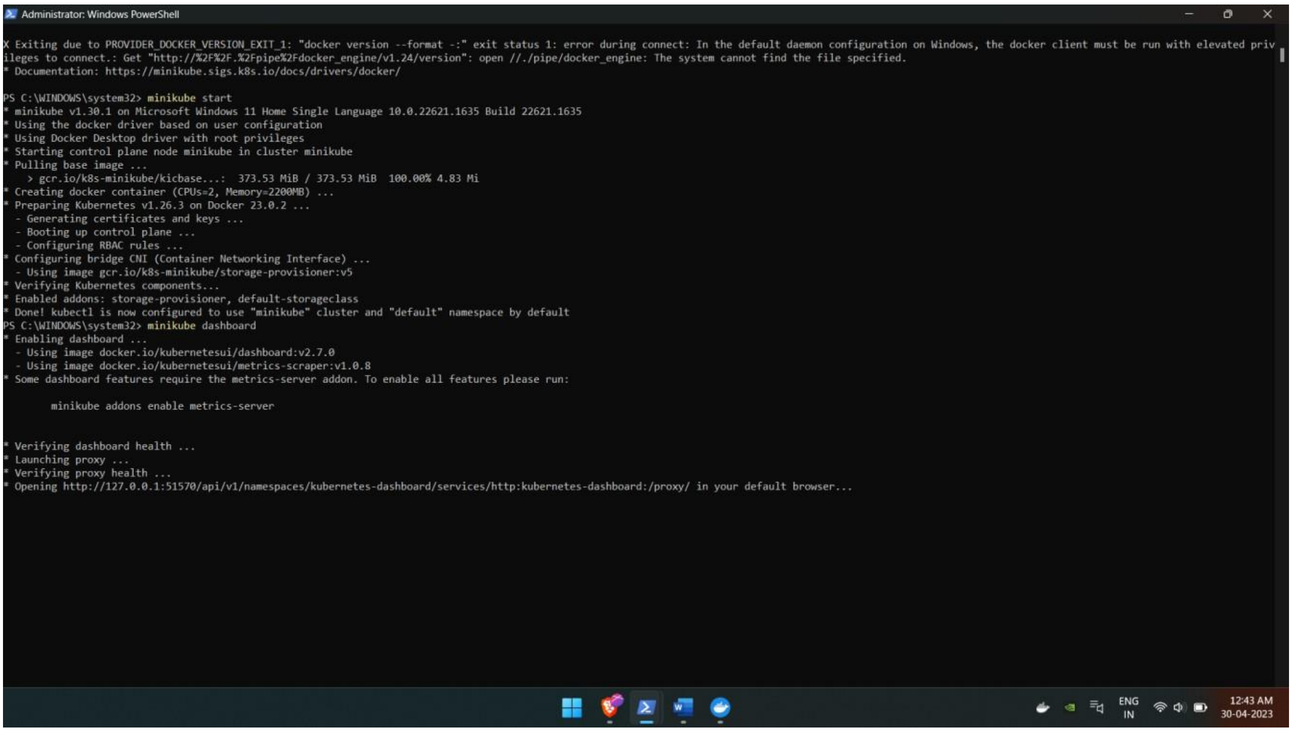


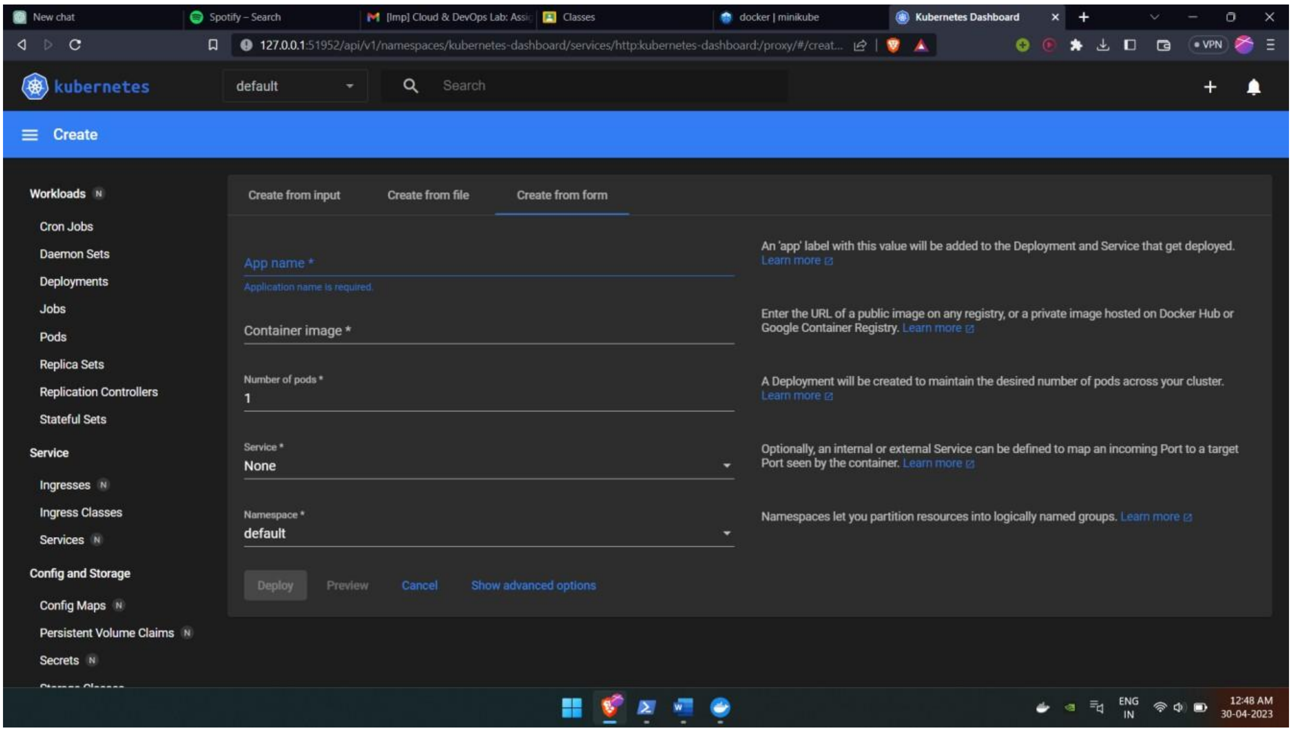
An Ingress may be configured to give Services externally-reachable URLs, load balance traffic, terminate SSL / TLS, and offer name-based virtual hosting. An Ingress controller is responsible for fulfilling the Ingress, usually with a load balancer, though it may also configure your edge router or additional frontends to help handle the traffic.

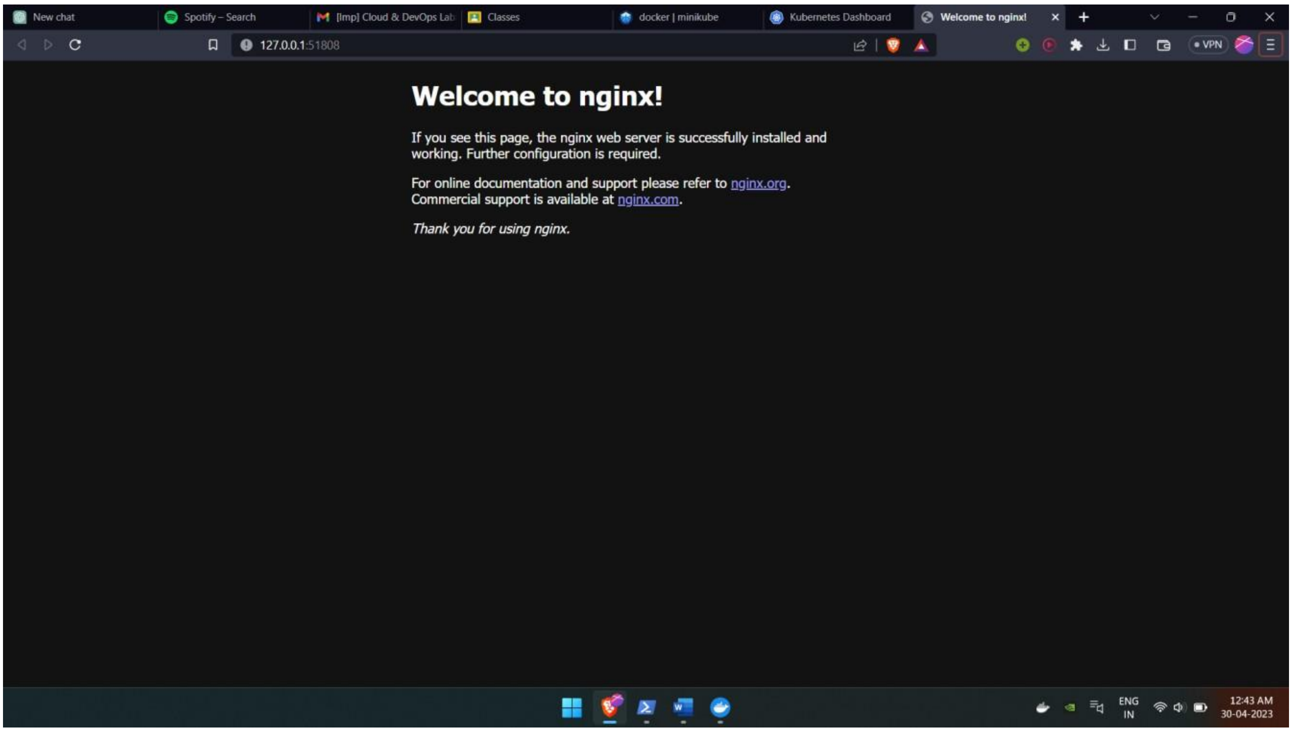
An Ingress does not expose arbitrary ports or protocols. Exposing services other than HTTP and HTTPS to the internet typically uses a service of type Service.Type=NodePort or Service.Type=LoadBalancer.

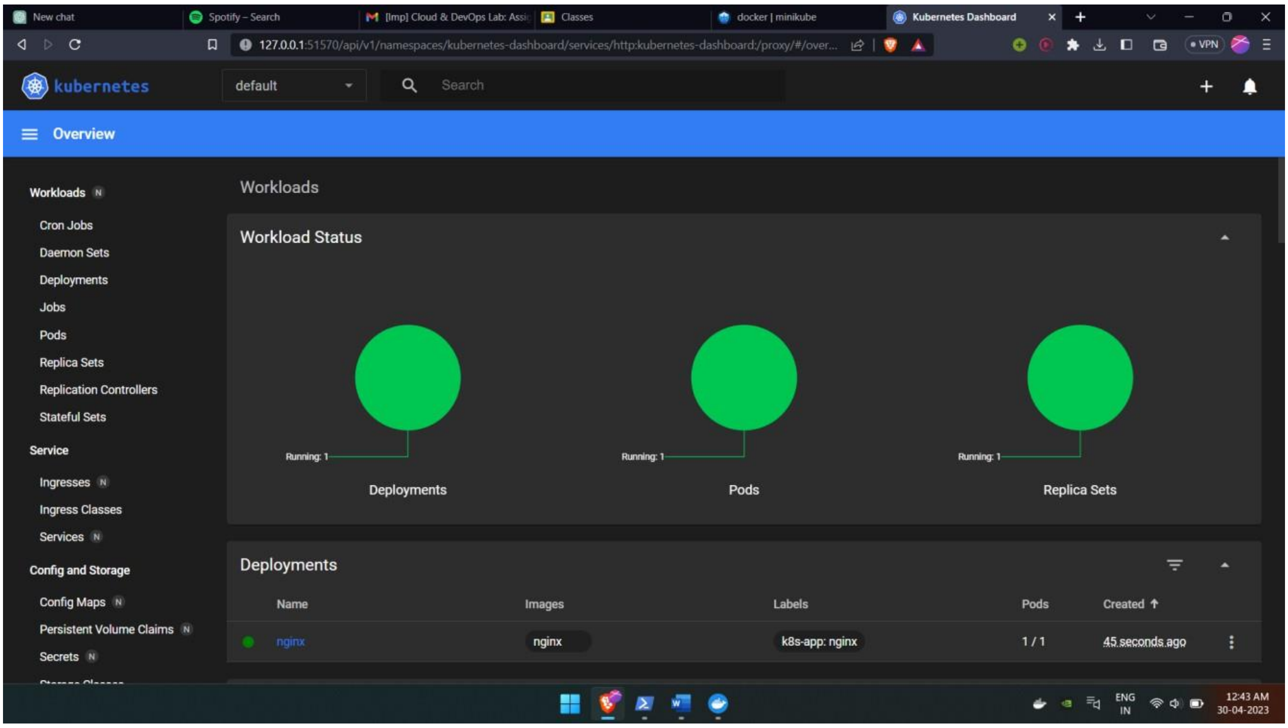


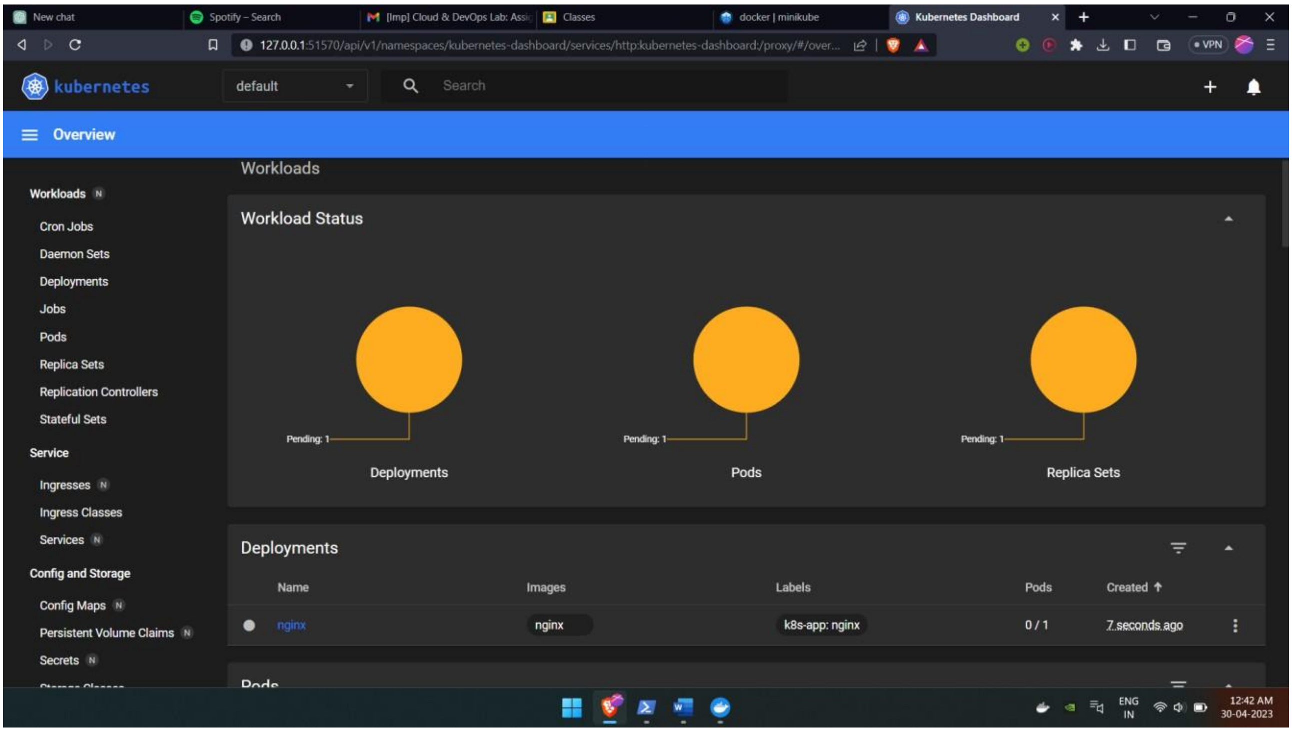

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\WINDOWS\system32> minikube service nginx
| NAMESPACE | NAME | TARGET PORT | URL |
| default | nginx | tcp-80-80-vpftn/80 | http://192.168.49.2:32461 |
* Starting tunnel for service nginx.
| NAMESPACE | NAME | TARGET PORT | URL |
| default | nginx | | http://127.0.0.1:51808 |
* Opening service default/nginx in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
ENG
IN
12:44 AM
30-04-2023

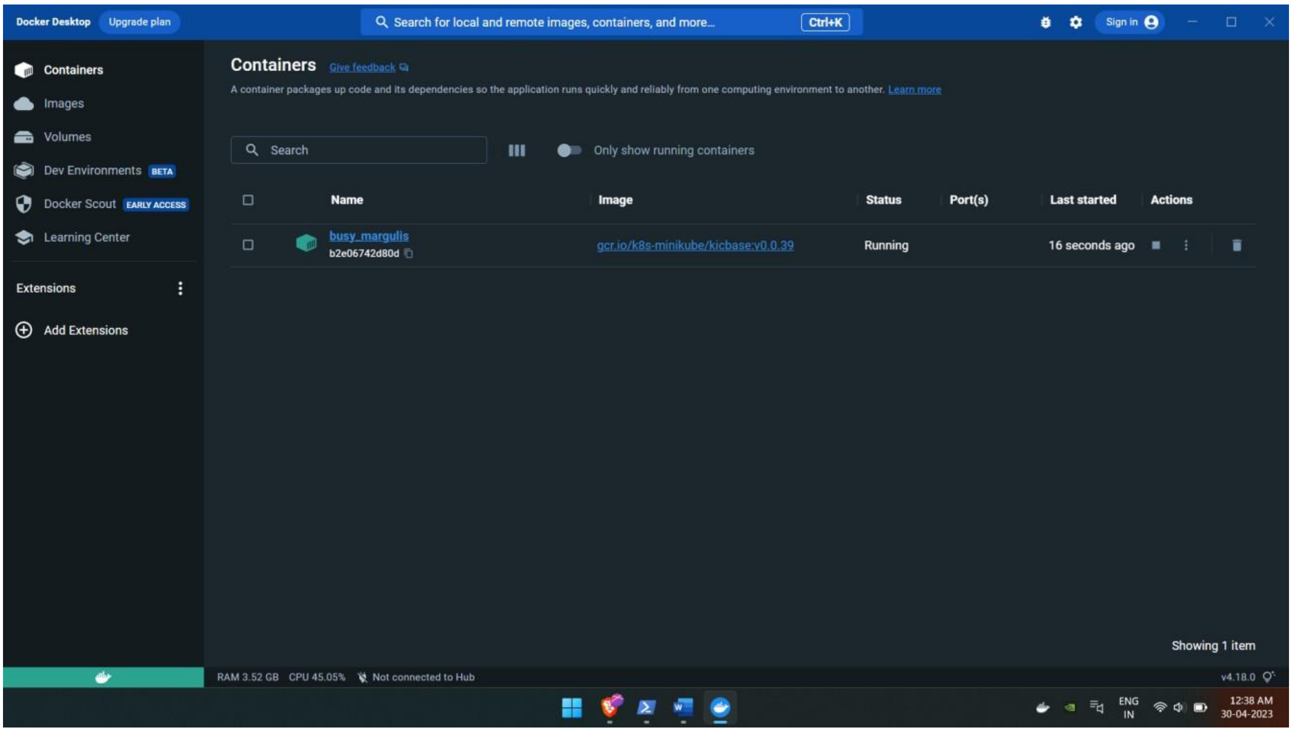

Administrator: Windows PowerShell
X Exiting due to PROVIDER_DOCKER_VERSION_EXIT_1: "docker version --format -:" exit status 1: error during connect: In the default daemon configuration on Windows, the docker client must be run with elevated priv
ileges to connect.: Get "http://%2F%2F.%2Fpipe%2Fdocker_engine/v1.24/version": open //./pipe/docker_engine: The system cannot find the file specified.
* Documentation: https://minikube.sigs.k8s.io/docs/drivers/docker/
PS C:\WINDOWS\system32> minikube start
* minikube v1.30.1 on Microsoft Windows 11 Home Single Language 10.0.22621.1635 Build 22621.1635
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
> gcr.io/k8s-minikube/kicbase...: 373.53 MiB / 373.53 MiB 100.00% 4.83 Mi
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
- Generating certificates and keys ...
- Booting up control plane ...
- Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
- Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Verifying Kubernetes components...
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS C:\WINDOWS\system32> 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 features please run:
minikube addons enable metrics-server
* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:51570/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser...
ENG
IN
12:43 AM
30-04-2023


New chat
Spotify – Search
[Imp] Cloud & DevOps Lab: Assi
Classes
docker | minikube
Kubernetes Dashboard
127.0.0.1:51952/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/creat...
kubernetes
default
Search
Create
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Create from input
Create from file
Create from form
App name *
Application name is required.
An 'app' label with this value will be added to the Deployment and Service that get deployed. Learn more
Container image *
Enter the URL of a public image on any registry, or a private image hosted on Docker Hub or Google Container Registry. Learn more
Number of pods *
1
A Deployment will be created to maintain the desired number of pods across your cluster. Learn more
Service *
None
Optionally, an internal or external Service can be defined to map an incoming Port to a target Port seen by the container. Learn more
Namespace *
default
Namespaces let you partition resources into logically named groups. Learn more
Deploy
Preview
Cancel
Show advanced options
ENG
IN
12:48 AM
30-04-2023


New chat
Spotify – Search
[Imp] Cloud & DevOps Lab
Classes
docker | minikube
Kubernetes Dashboard
Welcome to nginx!
127.0.0.1:51808
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.
ENG
IN
12:43 AM
30-04-2023


kubernetes
default
Search
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Workload Status
Running: 1
Deployments
Running: 1
Pods
Running: 1
Replica Sets
Deployments
Name
Images
Labels
Pods
Created
nginx
nginx
k8s-app: nginx
1 / 1
45 seconds ago
12:43 AM
30-04-2023


kubernetes
default
Search
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Secrets
Workload Status
Pending: 1
Deployments
Pending: 1
Pods
Pending: 1
Replica Sets
Deployments
Name
Images
Labels
Pods
Created
nginx
nginx
k8s-app: nginx
0 / 1
7 seconds ago
12:42 AM
30-04-2023



- **References for further study:**
  1) https://kubernetes.io/docs/home/
  2) you need to learn Kubernetes RIGHT NOW!! - Networkchuck
  3) techworld with Nana
  4) Kubernetes Concepts Explained in 9 minutes! - KodeKloud
  5) Kubernetes Tutorial for Beginners | Kubernetes Tutorial | Intellipaat
  6) Install Kubernetes | Setup Kubernetes Step by Step | Kubernetes Training | Intellipaat