kubernetes

Presented by:

MA067

TANK RAJNIKUMAR

MA075

VALAKI JAYMIN

AGENDA 1

- Architecture? And it's types.
- What is Monolith?
- Pros and Cons of Monolithic Architecture
- Problem with NETFLIX
- From Monolithic to Multi server
- Container?
- Docker vs Kubernetes

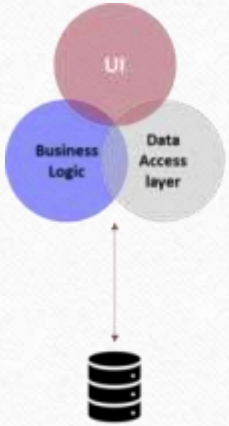
ARCHITECHURE & IT's TYPES

- **The architecture reflects how the system application is used and how it interacts with other systems and the outside world.**

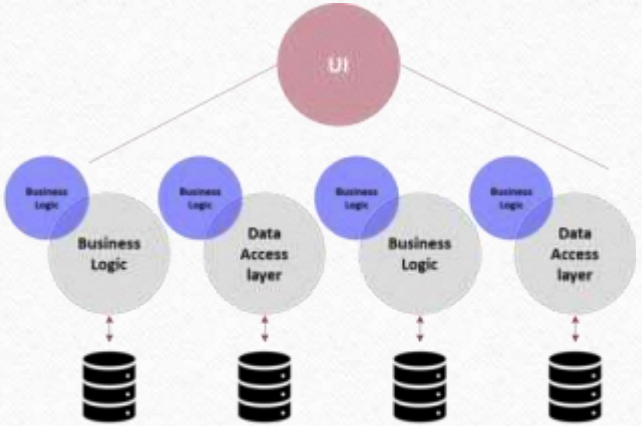
- **TYPES OF ARCHITECHURE**

  1)MONOLYTH ARCHITECHURE

  2)MULTI SERVICE ARCHITECHURE



MONOLITHIC ARCHITECTURE

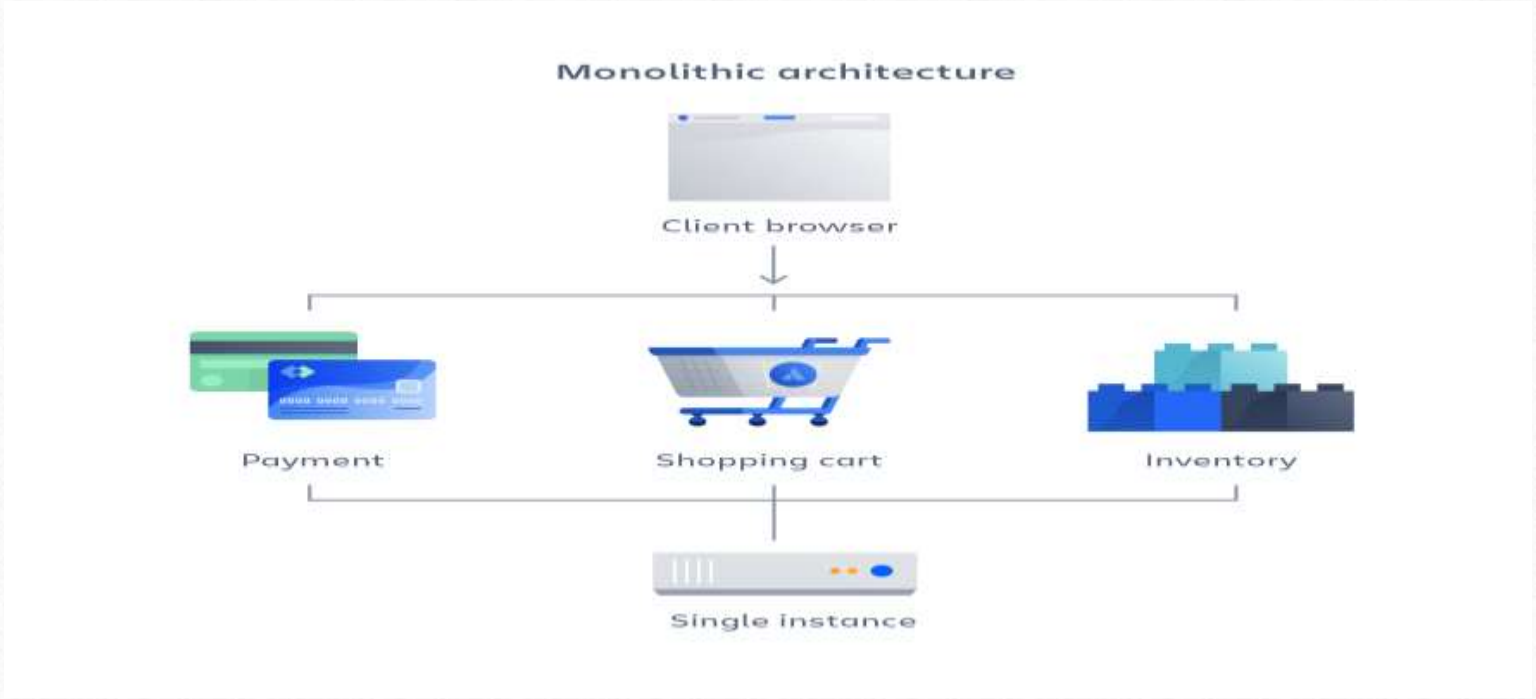


MICROSERVICE ARCHITECTURE

Monolith

- Traditional model of a software program
- built as a unified unit
- self-contained and independent from other applications
- singular, large computing network
- To make a change of application requires updating the entire stack
- building and deploying an updated version of the service-side interface

Example of Monolithic Architecture



Advantages of a Monolithic Architecture

- **Easy deployment**
- **Easy to develop**
- **Simplified testing**
- **Easy debugging**

Limitations of a Monolithic Architecture

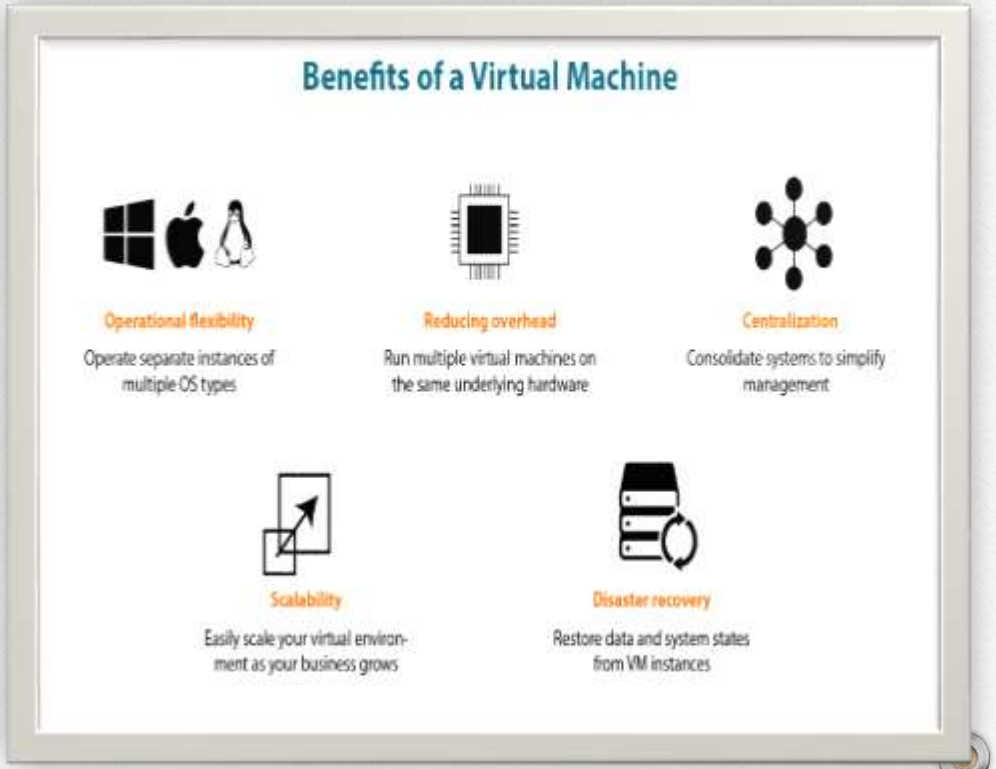
- **Slower development speed**
- **Scalability & Reliability**
- **Lack of flexibility**
- **“One application on one server”**

VIRTUAL MACHINE

- IN 1997 VM VARE COMPANY DEVELPED A CONCEPT OF VIRTUAL MACHINE

PROS & CONS OF VIRTUAL MACHINE

- WE CAN RUN MULTIPLE APPLICATION ON VIRTUAL MACHINE SO IT CAN EASILY SOLVE PROBLEM OF **“ONE SERVER MANY SERVICES”**



CONS OF VIRTUAL MACHINE

- IT NEED SPECIFIC CPU AND RAM ALLOCATION FOR DIFFERENT APPLICATION SO IT CAN MAKE SYSTEM SLOWER.
- IT CAN NOT SOLVE LATEST PROBLEM OF MODULE SHARING FROM ONE PC TO OTHER (FRIEND'S PC).

CONTAINERS – MICRO SERVISE SYSTEM

- Containers are an excellent example of micro services architecture as they allow businesses to focus on developing services without worrying about dependencies

Problem with NETFLIX

- In 2009 Netflix faced growing pains
- infrastructure couldn't keep up with the demand
- private data centers to a public cloud
- 2015 JAX Special Jury award
- Today, Netflix has more than a thousand micro-services

AGENDA

- Answer of Previous question?
- What is containerization?
- Virtualization vs Containerization
- Feature of containerization
- Different Vendors
- Docker vs Kubernetes
- Architecture of Kubernetes

DUAL BOOT VS VIRTUAL MACHINE

❖ CONTAINERIZATION

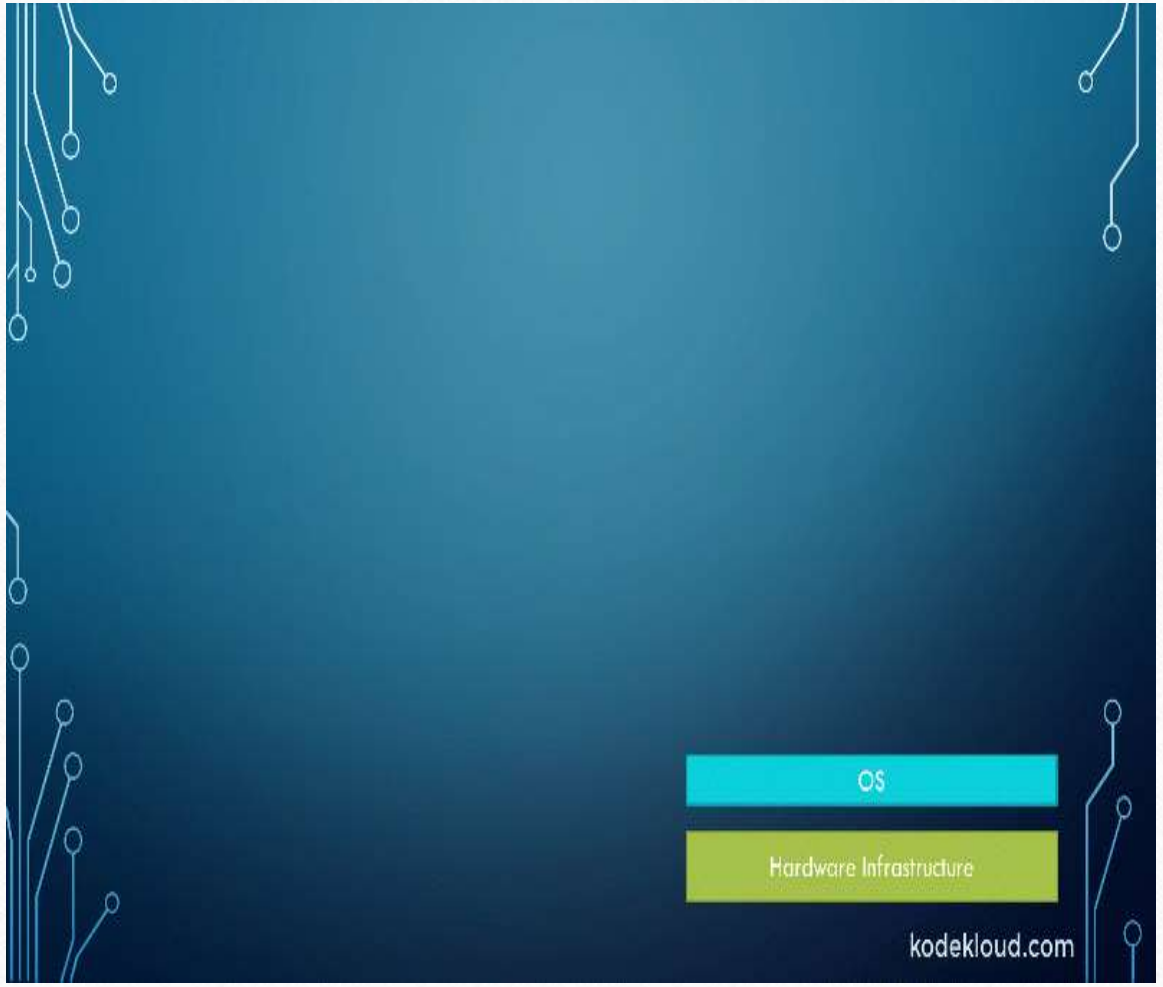
What are the containers ?

- Containers are a form of operating system virtualization. A single container might be used to run anything from a small micro service or software process to a larger application.
- Inside a container there are

1. Necessary executables binary
2. code libraries and
3. configuration files.

Virtualization vs Containerization

Each VM runs its own Operating system	All Container Share same OS of the host system
Boot up time required it is in limit	Container Initiate in second .
It is Not version controlled.	Docker containers work just like GIT , allowing you to a commit changes to your Docker images version control them.



Feature Of container

- **FLEXIBLE**

No matter the complexity, the dependencies, the languages, every application can be containerized.

- **LIGHTWEIGHT**

By sharing the same kernel, they don't consume a lot of system resources, so you save money, which is always a good thing.

- **Portable**

You can build locally, then run everywhere where Container is installed.

- **SCALABLE**

Indeed you can easily increase/decrease/automatically distribute containers replicas.

You can also use an tool, like Docker Swarm, or Kubernetes.

DIFFERENT VENDORS

- **10 Tech Companies Embracing Container Technology**

1) **DOCKER**
2) **KUBERNETS**
3) **AMAZON WEB SERVICES**
4) **DIGITAL OCEAN**
5) **FUZE and many more**

AGENDA

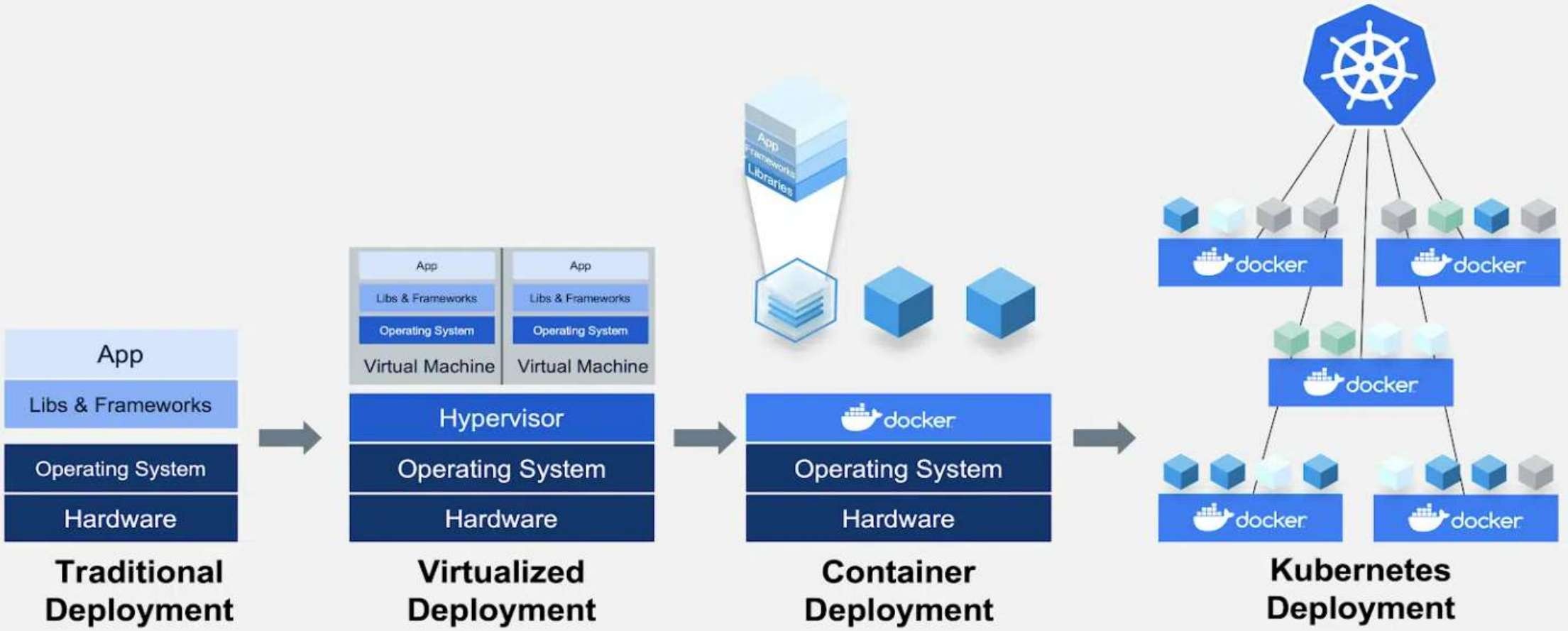
- Continue.... Docker vs Kubernetes
- What is Docker?
- Architecture of Docker
- History of Kubernetes
- What is Kubernetes?
- Architecture of Kubernetes

DOCKER VS KUBERNETS

VS

kubernetes

Kubernetes & Docker work together to build & run containerized applications

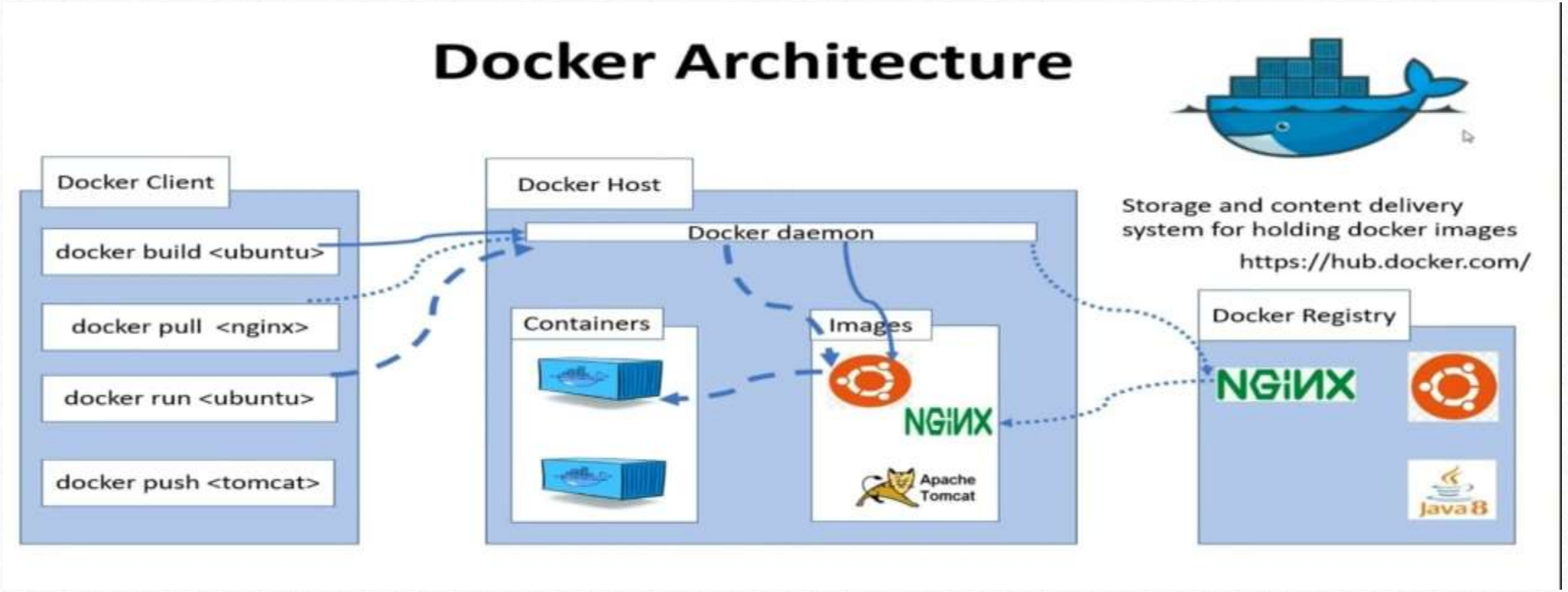


FEATURES	DOCKER	KUBERNETES
CONFIGURATION	Installation is simple; but once setup, the clusters is not very strong.	Installation is complicated; but once setup, the clusters is very strong
GUI	There is no GUI	GUI is the Kubernetes Dashboard
SCALABILITY	Highly scalable (almost 5x than kubernetes)	Scalable
DATA VOLUMES	Can share storage volumes with any other containers	Can share storage volumes only with other containers in same pod
Logging & Monitoring	3rd party tools like ELK should be used for logging and monitoring	In – built tools for logging and monitoring

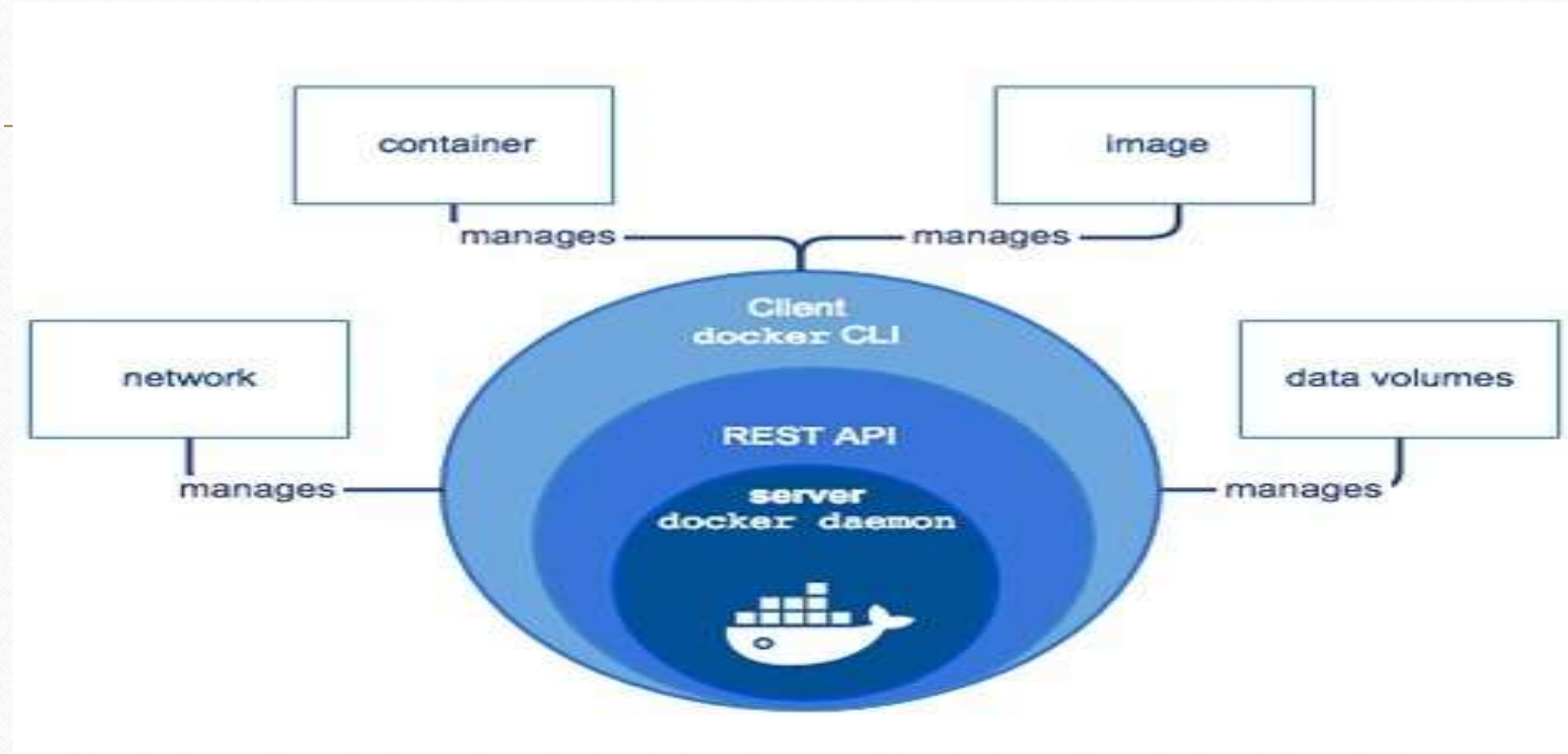
What is Docker?

- A container platform
- Allows you to build, test and deploy applications quickly.
- A developer defines all the applications and it's dependencies in a Dockerfile.
- then used to build Docker images that defines a Docker container.

Architecture of Docker



Architecture of Docker



History of Kubernetes

- AWS Offer Cloud
- GOOGLE ‘s orchestration
  - Borg
  - Omega
- Google – made K8s Open Source in 2014
- Donated to CNCF

AGENDA

- What's the problem that Docker or kubernetes solve ?
- What is Docker and it's components?
  - Docker File
  - Docker image
  - Docker Container
- Docker Command
- Docker demo

CITY GO

The Ultimate Service Finder

› This PC › DATA (D:) › DOCKER DEMO

Name	Date modified	Type
admin panel	17-02-2023 19:51	File folder
City-Go	17-02-2023 19:46	File folder
Database	17-02-2023 19:51	File folder

SHARE CITY GO SITE

MR SHARMA

FRIEND 1

FRIEND 2

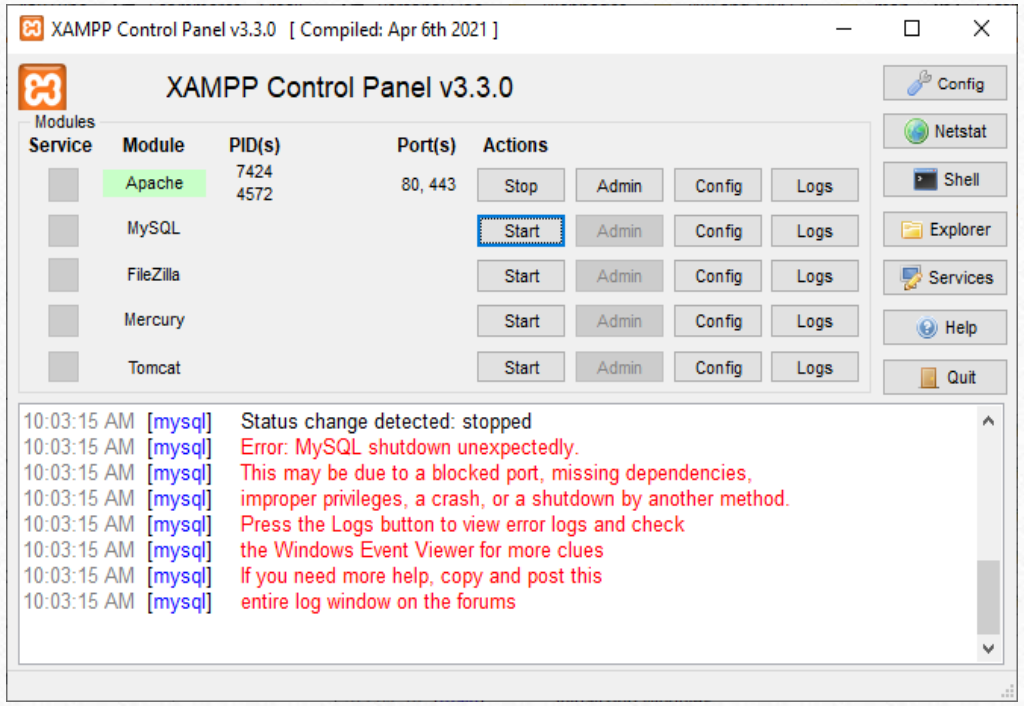
HOW MANY WAYS TO SHARE SITE

1)USE OF PENDRIVE

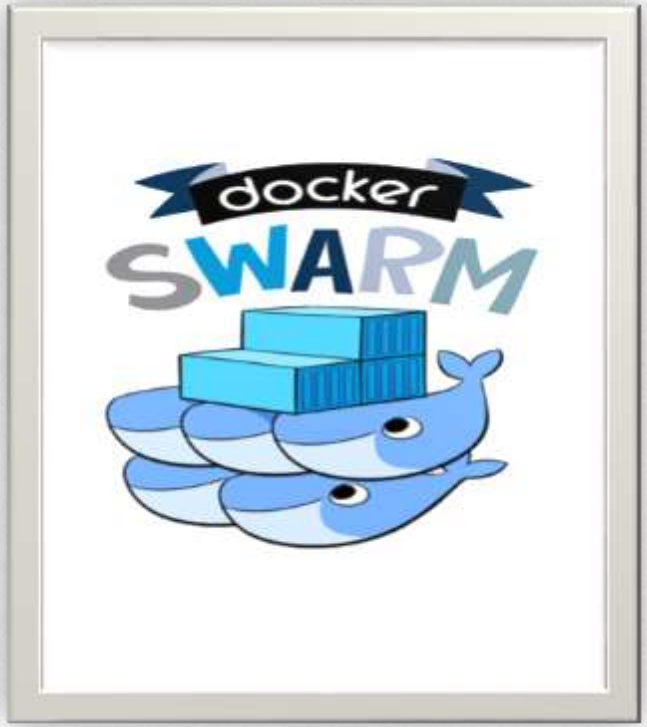
2)USE GITHUB LIKE TOOLS

3)USE CONTAINERIZATION

What Are The Issue That Friends Face ?

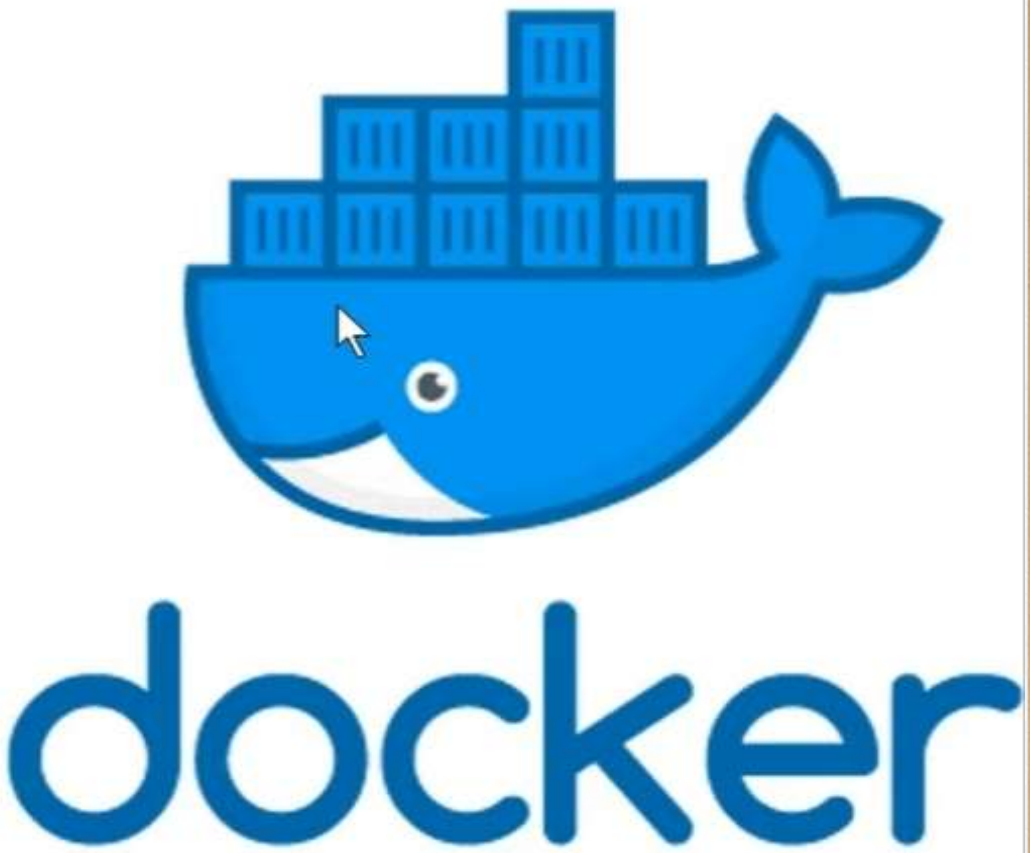


We can solve this issue by Containerization



Prerequisites

- Linux
- Software Development



Docker Demo

Let's start with demo and

some important keywords and

commands to understand Docker properly.

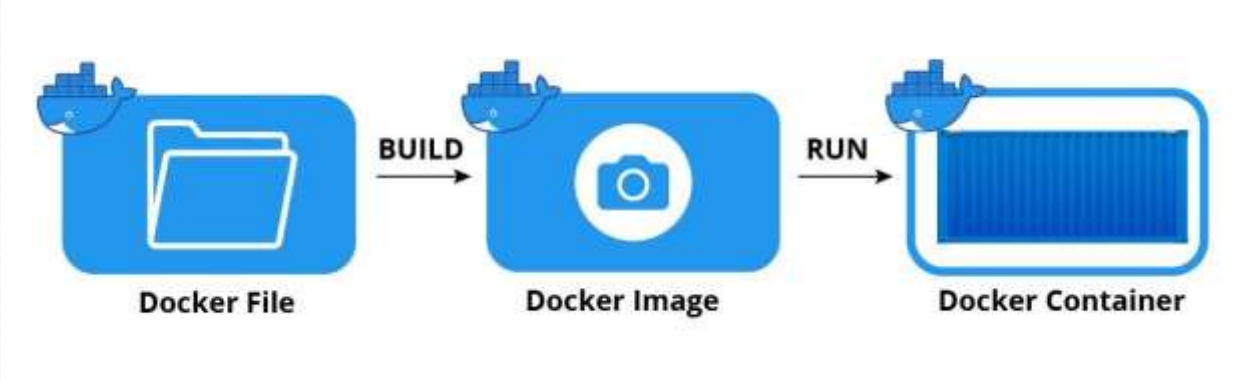
What is Docker ?

- **Docker is a platform for developers and sysadmins to build, run, and share applications with containers.**

- **The use of containers to deploy applications is called containerization.**

Important keywords related to Docker:

- Dockerfile: A text file that contains instructions for building a Docker image.
- Docker image: A lightweight executable package that includes everything needed to run a piece of software, including code, libraries & dependencies.
- Container: An instance of a Docker image that can be run, started, stopped, moved, and deleted.



What is Dockerfile?

Dockerfile

- A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image.

- Using docker build users can create an automated build that executes several command-line instructions in succession.

https://github.com/docker-library/mysql

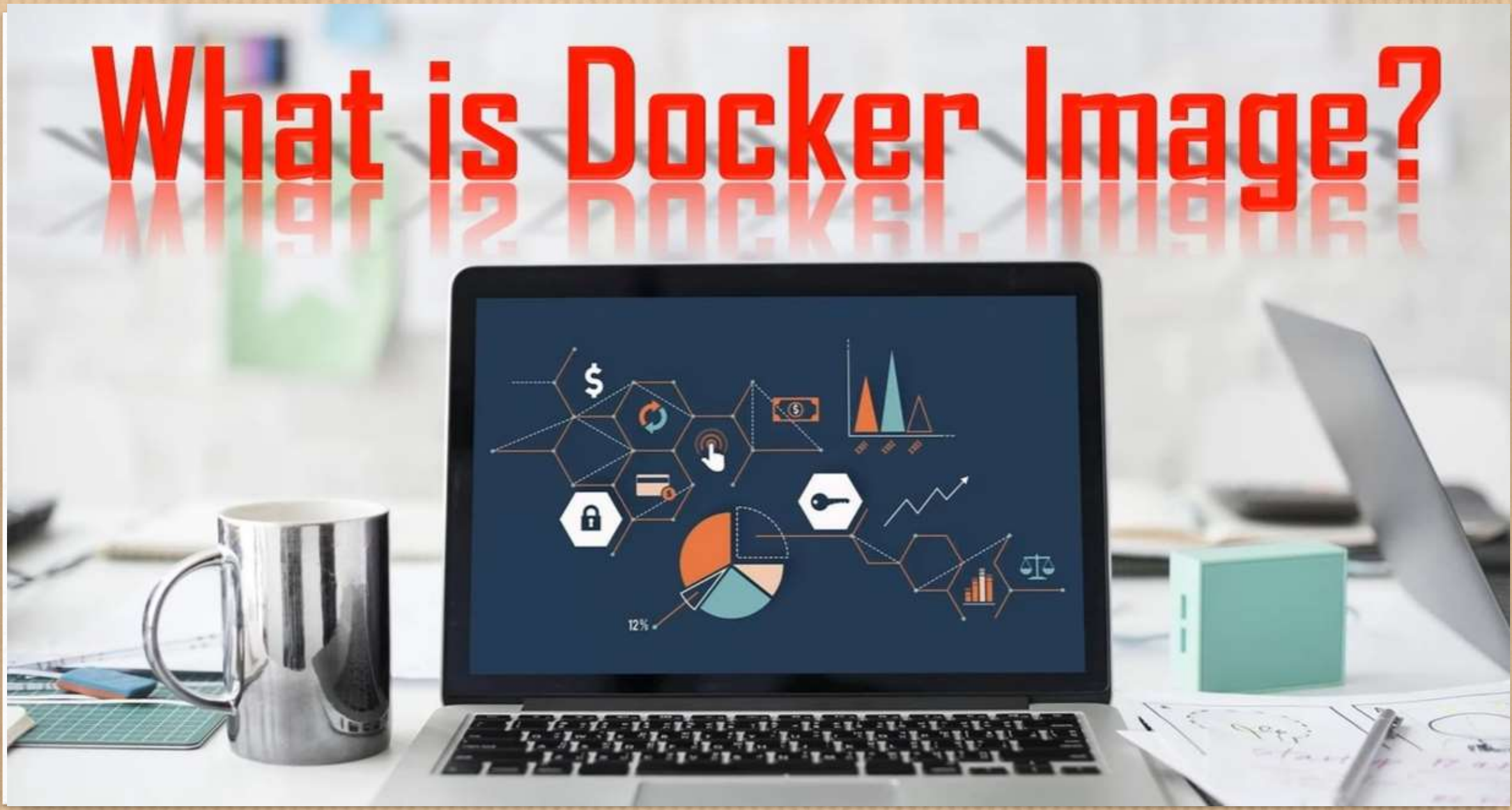
```
#
# https://github.com/dockerfile/mysql
#

# Pull base image.
FROM dockerfile/ubuntu

# Install MySQL.
RUN \
  apt-get update && \
  DEBIAN_FRONTEND=noninteractive apt-get install -y mysql-server && \
  rm -rf /var/lib/apt/lists/* && \
  sed -i 's/^\(bind-address\s.*\)/# \1/' /etc/mysql/my.cnf && \
  sed -i 's/^\(log_error\s.*\)/# \1/' /etc/mysql/my.cnf && \
  echo "mysqld_safe &" > /tmp/config && \
  echo "mysqladmin --silent --wait=30 ping || exit 1" >> /tmp/config && \
  echo "mysql -e 'GRANT ALL PRIVILEGES ON *.* TO \"root\"@\"%\" WITH GRANT OPTION;'" >> /tmp/config && \
  bash /tmp/config && \
  rm -f /tmp/config

# Define mountable directories.
VOLUME ["/etc/mysql", "/var/lib/mysql"]

# Define working directory.
WORKDIR /data

# Define default command.
CMD ["mysqld_safe"]

# Expose ports.
EXPOSE 3306
```

Dockerfile

- Using docker build users can create an automated build that executes several command-line instructions in succession.

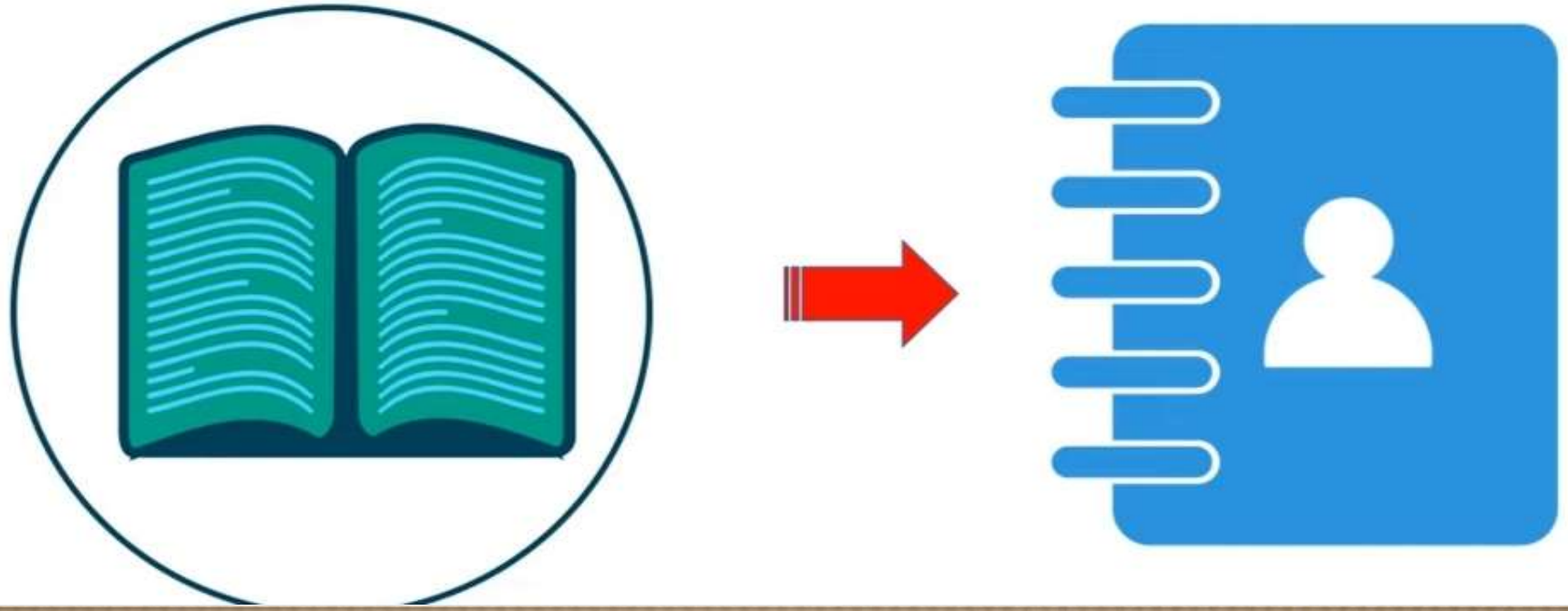
```
$ sudo docker build –t  docker_image_name .
    $ sudo docker run docker_image_name
```

What is Docker Image?



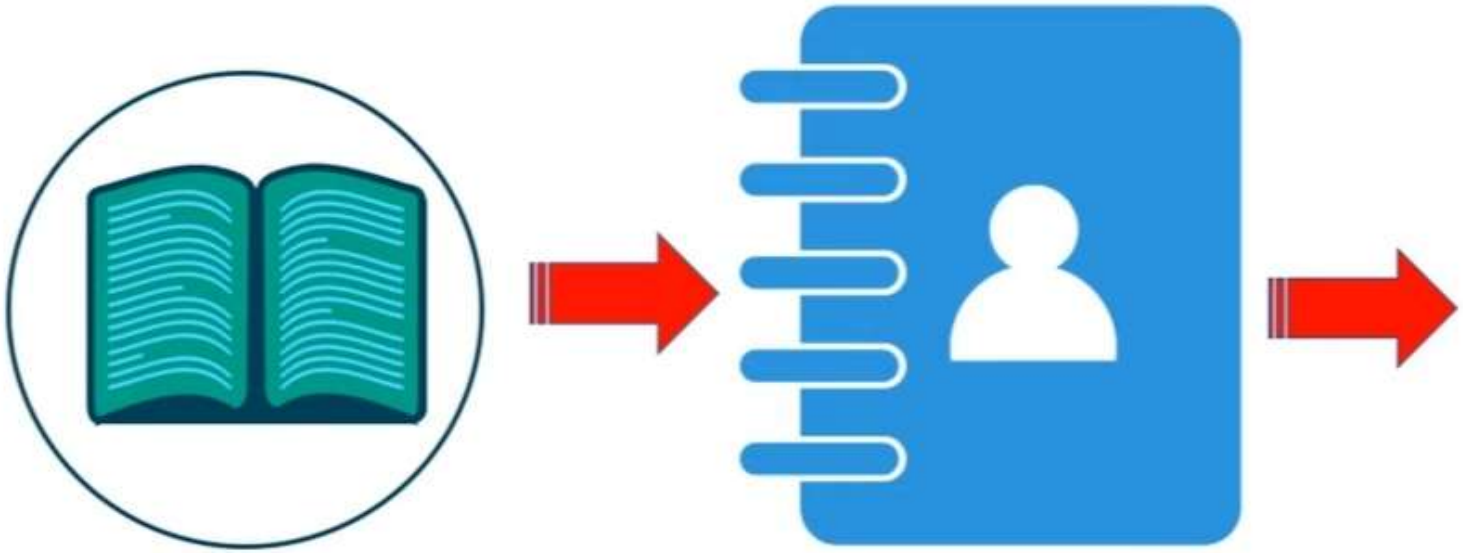
Docker Image

- A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform.

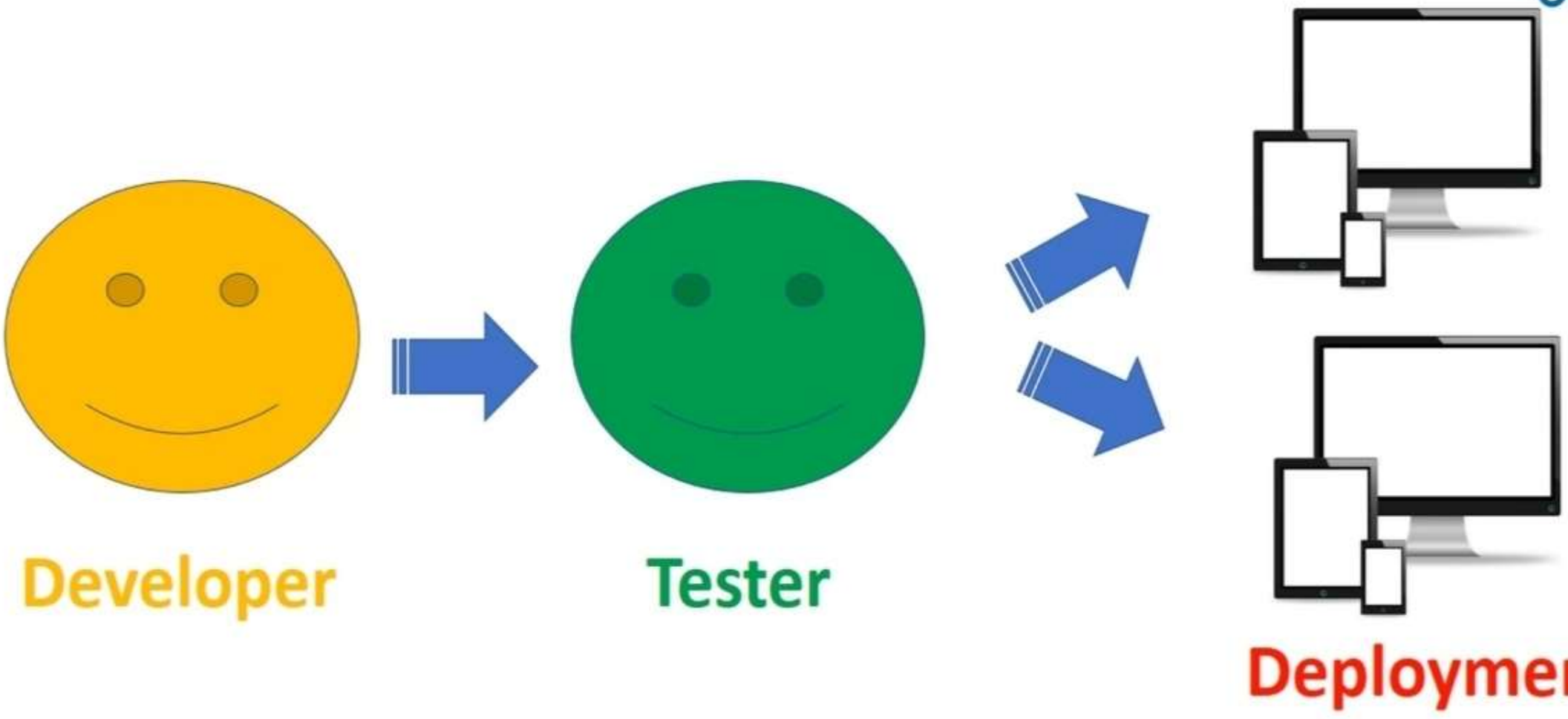


Docker Container

- A Docker container is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.



Who build Dockerfile, Image, Container?



DOCKER COMMANDS

Run

Run a container from the Alpine version 3.9 image, name the running container "web" and expose port 5000 externally, mapped to port 80 inside the container.

```
docker container run --name web -p 5000:80 alpine:3.9
```

List the running containers (add `--all` to include stopped containers)

```
docker container ls
```

Delete all running and stopped containers

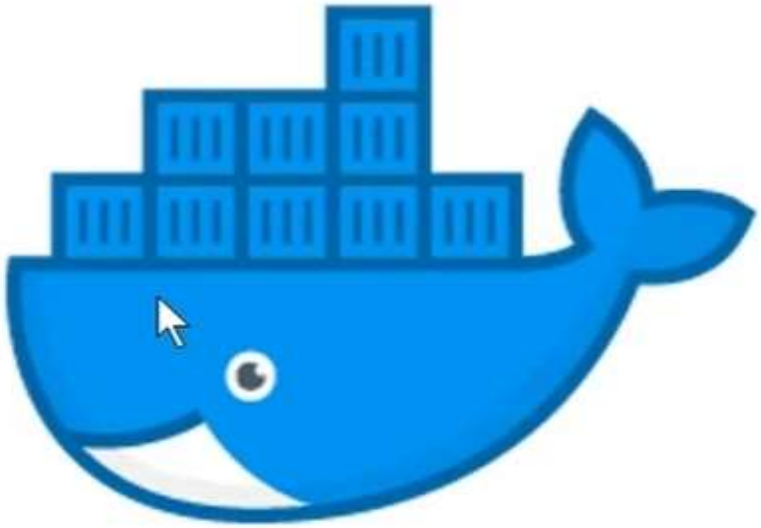
```
docker container rm -f $(docker ps -aq)
```


Share
Pull an image from a registry
docker pull myimage:1.0
Retag a local image with a new image name and tag
docker tag myimage:1.0 myrepo/myimage:2.0
Push an image to a registry
docker push myrepo/myimage:2.0

Installation of Docker

```
$ sudo apt install docker.io
$ sudo snap install docker
```

```
$ docker --version
```



docker

Docker Management

All commands below are called as options to the base **`docker`** command. **`Run docker <command> --help`** for more information on a particular command.

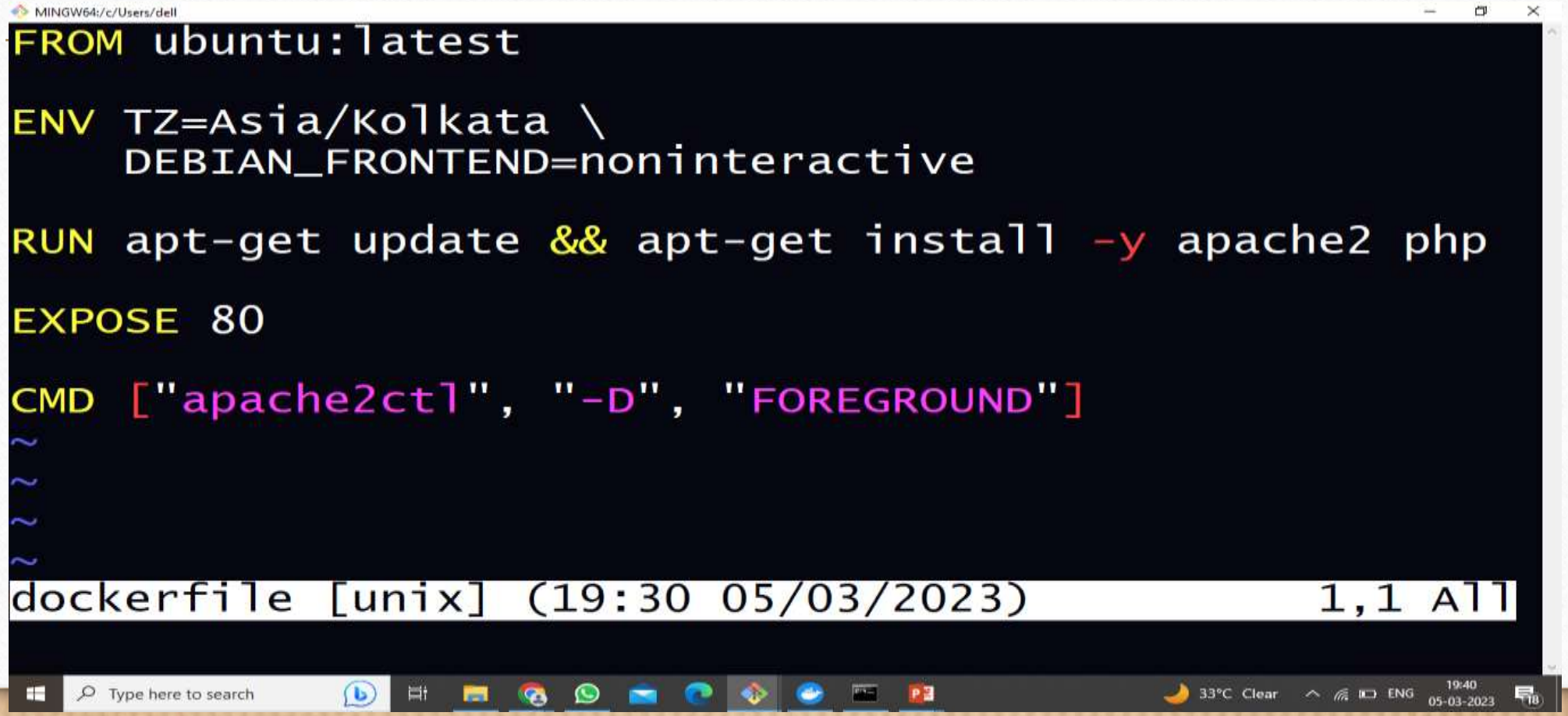
`app*`	*Docker Application*
`assemble*`	*Framework-aware builds (Docker Enterprise)*
`builder`	*Manage builds*
`cluster`	*Manage Docker clusters (Docker Enterprise)*
`config`	*Manage Docker configs*
`context`	*Manage contexts*
`engine`	*Manage the docker Engine*
`image`	*Manage images*
`network`	*Manage networks*
`node`	*Manage Swarm nodes*
`plugin`	*Manage plugins*
`registry*`	*Manage Docker registries*
`secret`	*Manage Docker secrets*

Live Demo of Docker Commands

- How the Docker-file Actually looks?
  - Custom image build command
    - Image list command
    - How to remove Image
      - Create a container

Docker Image Command

How the Docker-file Actually looks?



Custom image build command

Image list command

How to remove Image?

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
demo                      1.0       394022f1829e   2 hours ago     253MB
docker/getting-started    latest    3e4394f6b72f   2 months ago    47MB

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker image rm demo:1.0
Untagged: demo:1.0
Deleted: sha256:394022f1829e58e587f816d6e681be6a4a5ac66c2b5de8017cf209540ec94f6a

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker image ls
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
docker/getting-started    latest    3e4394f6b72f   2 months ago    47MB
```

Docker Container Command

Create a container

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker create citygo
64e7b1ec9d5eb55ec49600d4e3e556aeae475c08c13ff2a30752b6f934420499


dell@RAJNIKUMAR MINGW64 ~/docker
$ docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED       STATUS       PORTS                  NAMES
eaefe078084f   docker/getting-started   "/docker-entrypoint.…"   3 hours ago   Up 3 hours   0.0.0.0:80->80/tcp     cool_nightingale

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED         STATUS       PORTS                  NAMES
64e7b1ec9d5e   citygo                   "apache2ctl -D FOREG…"   3 minutes ago   Created                             reverent_yalow
eaefe078084f   docker/getting-started   "/docker-entrypoint.…"   3 hours ago     Up 3 hours   0.0.0.0:80->80/tcp     cool_nightingale
```

Run a command in new container

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker run citygo

dell@RAJNIKUMAR MINGW64 ~/docker
```

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS                      PORTS              NAMES
1d5384cb3936   citygo                 "/bin/bash"              45 seconds ago   Exited (0) 44 seconds ago                      competent_chatterjee
eaefe078084f   docker/getting-started "/docker-entrypoint.…"   4 hours ago      Up 4 hours                  0.0.0.0:80->80/tcp 
```

Rename an existing container

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker rename  competent_chatterjee citygo_container

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS                     PORTS                NAMES
1d5384cb3936   citygo                 "/bin/bash"              8 minutes ago   Exited (0) 8 minutes ago                        citygo_container
eaefe078084f   docker/getting-started "/docker-entrypoint.…"  4 hours ago     Up 4 hours                 0.0.0.0:80->80/tcp   cool_nightingale
```

Remove all stopped Containers

```
$

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] n
Total reclaimed space: 0B

dell@RAJNIKUMAR MINGW64 ~/docker
$
```

Create a new image from a container's files change

```
docker container commit [OPTIONS]
CONTAINER [REPOSITORY[:TAG]]
```

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker container commit citygo_container citygo:1.0
sha256:af42520f560d87fa03bcfac539ca25837edff4b1cd3d995fda24c7f58d6d9675

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED             SIZE
citygo                    1.0       af42520f560d   12 seconds ago      253MB
citygo                    latest    fb89f1693cbe   57 minutes ago      253MB
docker/getting-started    latest    3e4394f6b72f   2 months ago        47MB
```

Remove Container

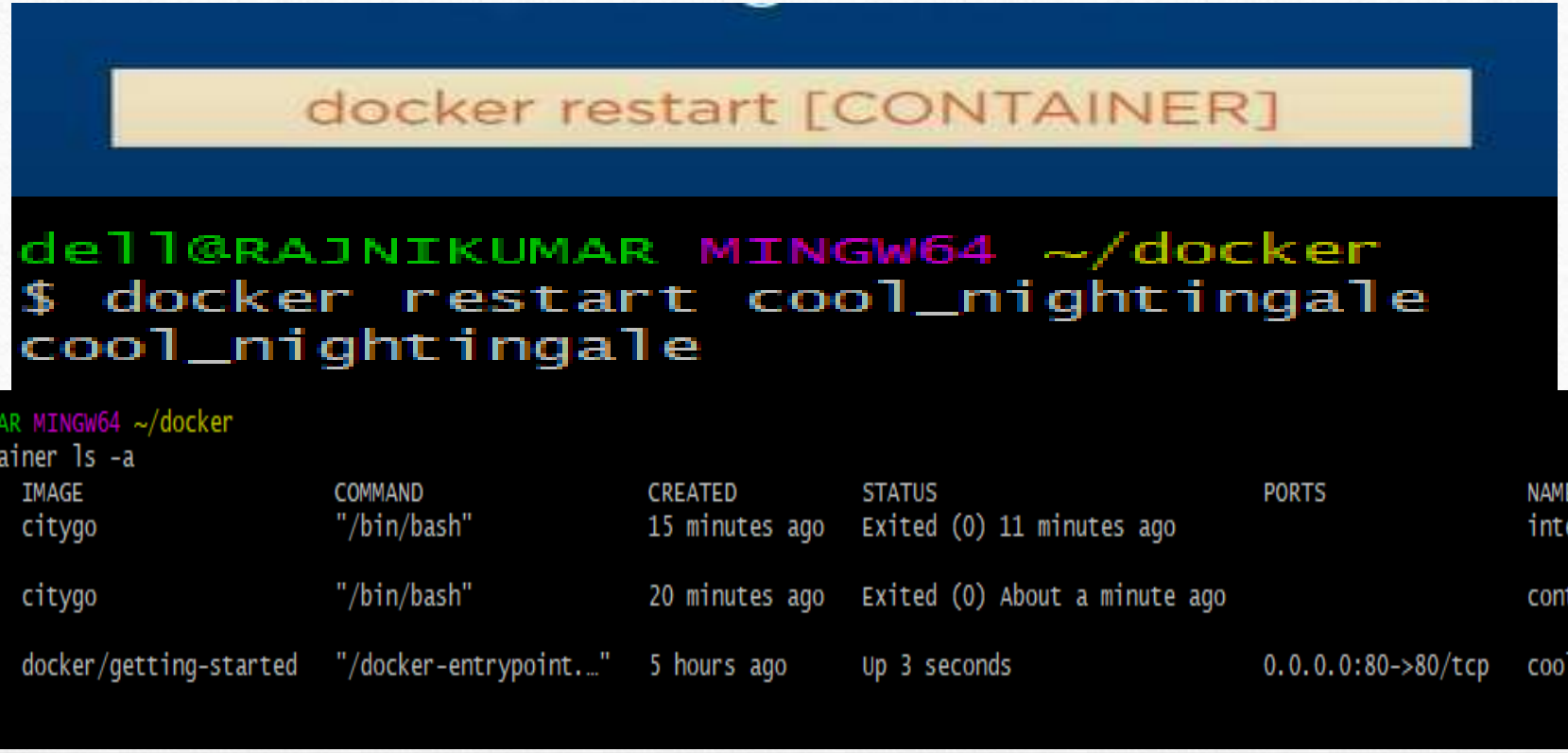
```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker container rm cool_nightingale
Error response from daemon: You cannot remove a running contai
ner eaefe078084f5fc211b24b31cd5327f428b955bda4593f2c9c3b6f0b05
4816ee. Stop the container before attempting removal or force
remove

dell@RAJNIKUMAR MINGW64 ~/docker
```

Container stop command

```
docker stop [CONTAINER_NAME]
```

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker stop cool_nightingale
cool_nightingale

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS                     PORTS     NAMES
b02bbdb7e1f9   citygo                 "/bin/bash"              10 minutes ago  Exited (0) 6 minutes ago             intelligent_napier
c20cd0f10682   citygo                 "/bin/bash"              15 minutes ago  Exited (0) 15 minutes ago            container_citygo
eaefe078084f   docker/getting-started  "/docker-entrypoint.…"  5 hours ago     Exited (0) 5 seconds ago             cool_nightingale
```

Restart container



Kill a running container

docker Kill [CONTAINER]

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker kill container_citygo
Error response from daemon: Cannot kill container: container_citygo: Container c20cd0f1068299bbdeccbf72315bda15f66a20408ad54296059e9da07cd7b6eb is not running

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker kill cool_nightingale
cool_nightingale
```

Attach local standard input and output

docker attach [CONTAINER]

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker attach cool_nightingale

[12]+  Stopped                 docker attach cool_nightingale
     0 [sig] sh 2789! sigpacket::process: Suppressing signal 18 to win32 process (pid 8220)
```

Block Container

docker wait [CONTAINER]

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker wait cool_nightingale



[13]+  Stopped                 docker wait cool_nightingale
      0 [sig] sh 2803! sigpacket::process: Suppressing signal 18 to win32 process (pid 15128)
```

Docker Share Command

Pull an image from a registry

```
docker pull [OPTIONS] NAME[:TAG]
```

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
b4ddc423e046: Pulling fs layer
b338d8e4ffd1: Pulling fs layer
b2b1b06949ab: Pulling fs layer
Digest: sha256:d8dc78532e9eb3759344bf89e6e7236a34132ab79150607eb08cc746989aa047
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
citygo                    1.0       af42520f560d   2 hours ago      253MB
citygo                    latest    fb89f1693cbe   3 hours ago      253MB
mysql                     latest    4f06b49211c0   9 days ago       530MB
docker/getting-started    latest    3e4394f6b72f   2 months ago     47MB
```

Need to login before push

```
C:\Users\dell>docker login
Login with your Docker ID to push and pull images from Docker Hub.
Username: rajnikumartank
Password:
Login Succeeded
```

Push an image to a registry


```
C:\Users\dell>docker push citygo:latest
The push refers to repository [docker.io/library/citygo]
34163163b2c7: Preparing
d2a5ca75bd91: Preparing
9296f9ee3426: Preparing
202fe64c3ce3: Preparing
denied: requested access to the resource is denied
```

Push an image to a registry

- Re-tag image with user name

```
dell@RAJNIKUMAR MINGW64 ~/docker
$ docker tag citygo:1.0 rajnikumartank/citygo:1.0

dell@RAJNIKUMAR MINGW64 ~/docker
$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED         SIZE
rajnikumartank/citygo     1.0       af42520f560d   2 hours ago     253MB
citygo                    1.0       af42520f560d   2 hours ago     253MB
citygo                    latest    fb89f1693cbe   3 hours ago     253MB
mysql                     latest    4f06b49211c0   9 days ago      530MB
docker/getting-started    latest    3e4394f6b72f   2 months ago    47MB
```

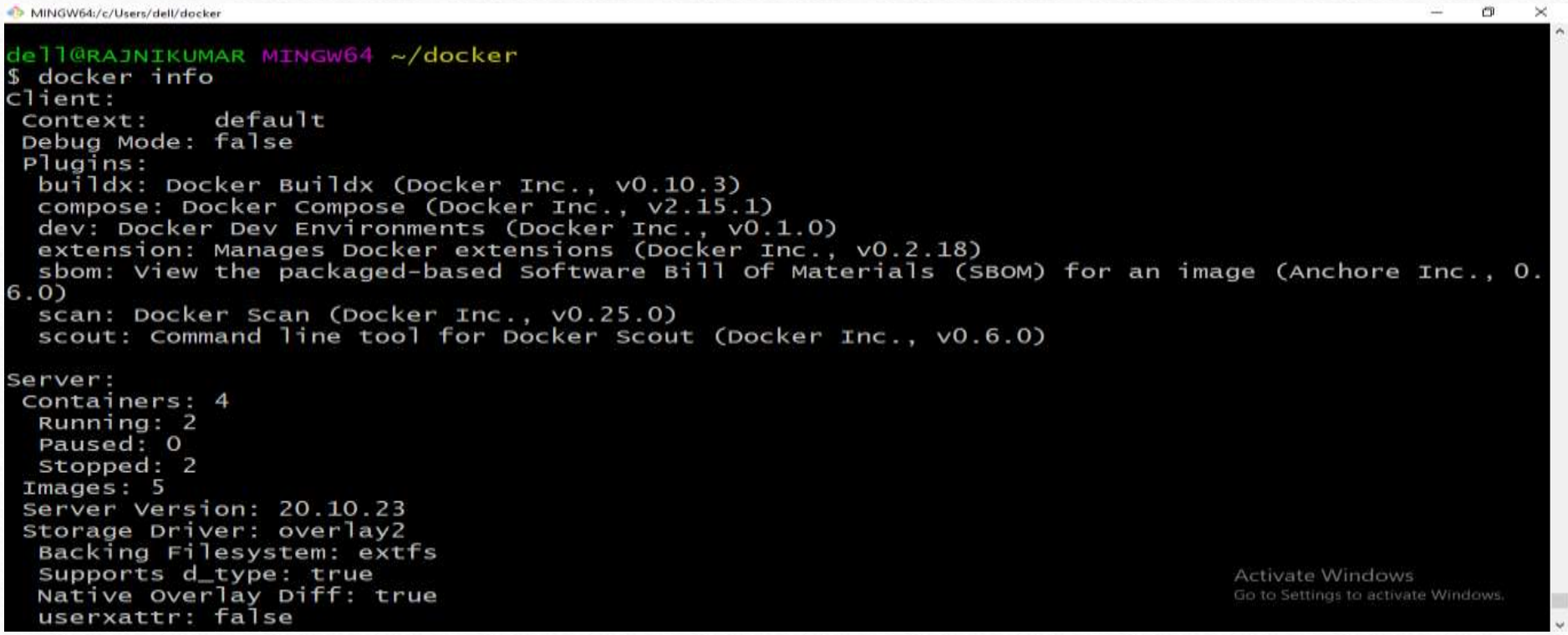
Push an image to a registry

```
C:\Users\dell>docker push rajnikumartank/citygo:1.0
The push refers to repository [docker.io/rajnikumartank/citygo]
34163163b2c7: Pushed
d2a5ca75bd91: Pushed
9296f9ee3426: Pushed
202fe64c3ce3: Pushed
1.0: digest: sha256:235c4408f1032692fae2c6393b32d75f046e9b0062d5724da022ecd8e0822d54 size: 1156
```

Run a Command in a running container


```
C:\Users\dell>
C:\Users\dell>docker exec -it suspicious_borg bash
root@b676688a8044:/usr/local/apache2# images
bash: images: command not found
root@b676688a8044:/usr/local/apache2# ps
bash: ps: command not found
root@b676688a8044:/usr/local/apache2# echo "hello"
hello
root@b676688a8044:/usr/local/apache2#
```

Get information of Docker tool



Display History of Image

AGENDA

- What is Kubernetes?
- Architecture of Kubernetes
- Benefit of K8s

What is Kubernetes?

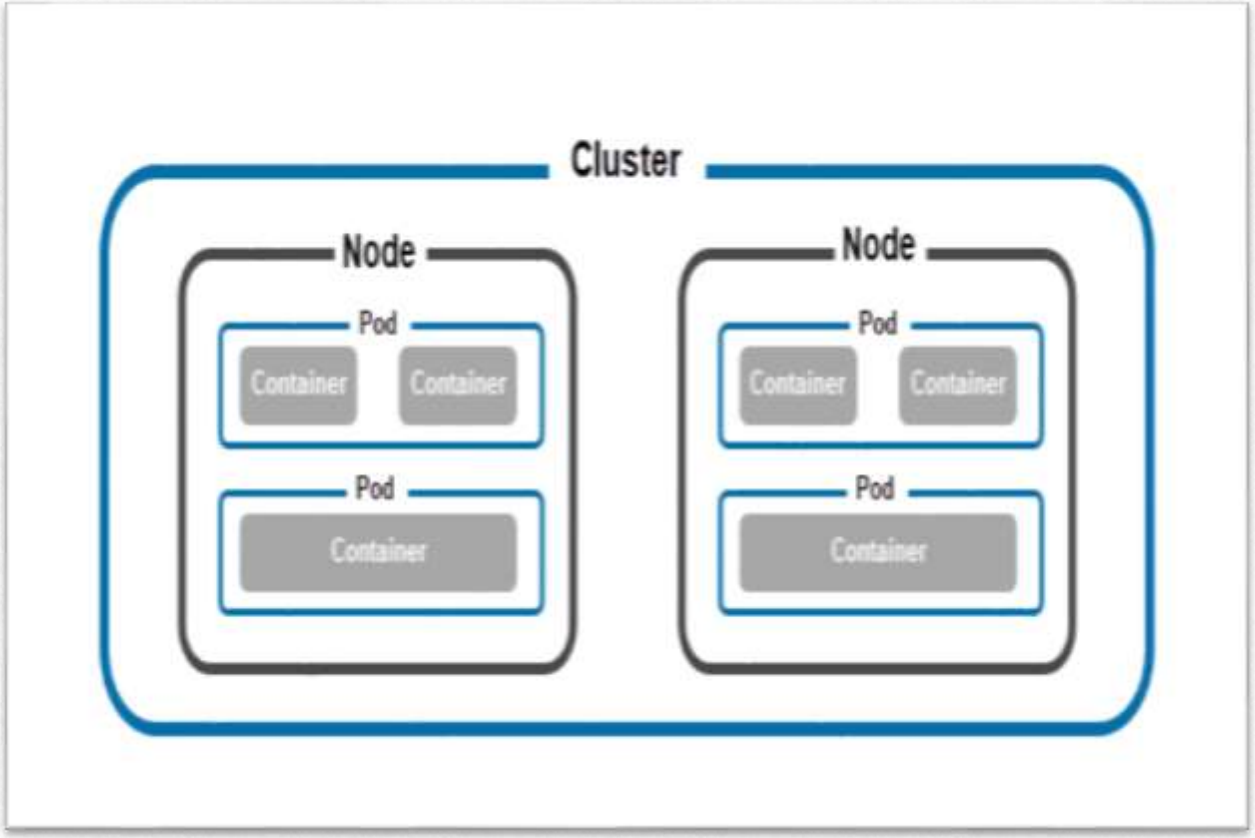
- An open-source and more than container orchestration system
- For automating software deployment, scaling, and management
- Originally developed by Google
- helps us to manage applications
- made up of hundreds or maybe thousands of containers
- helps us to manage them in different environments

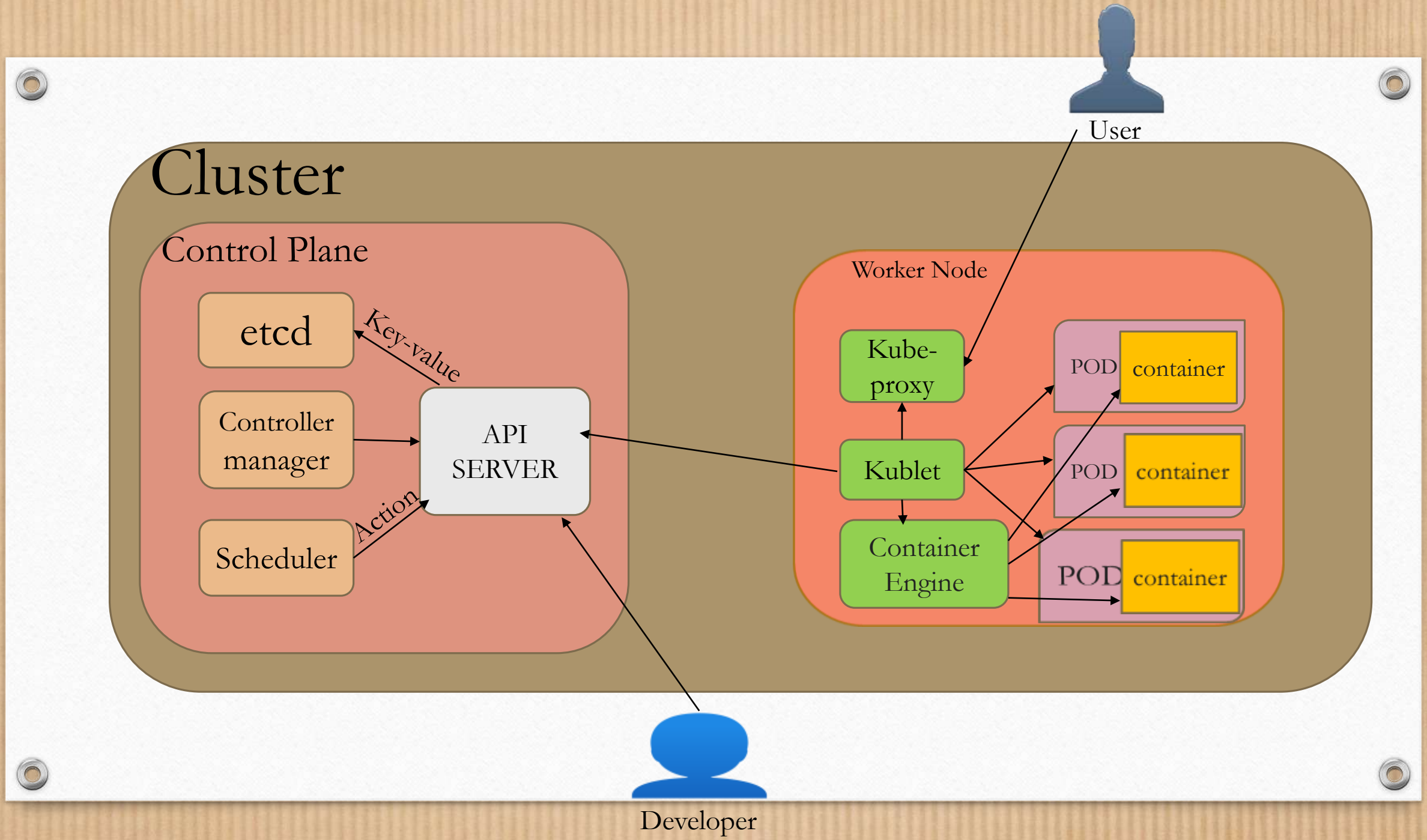
Architecture of Kubernetes

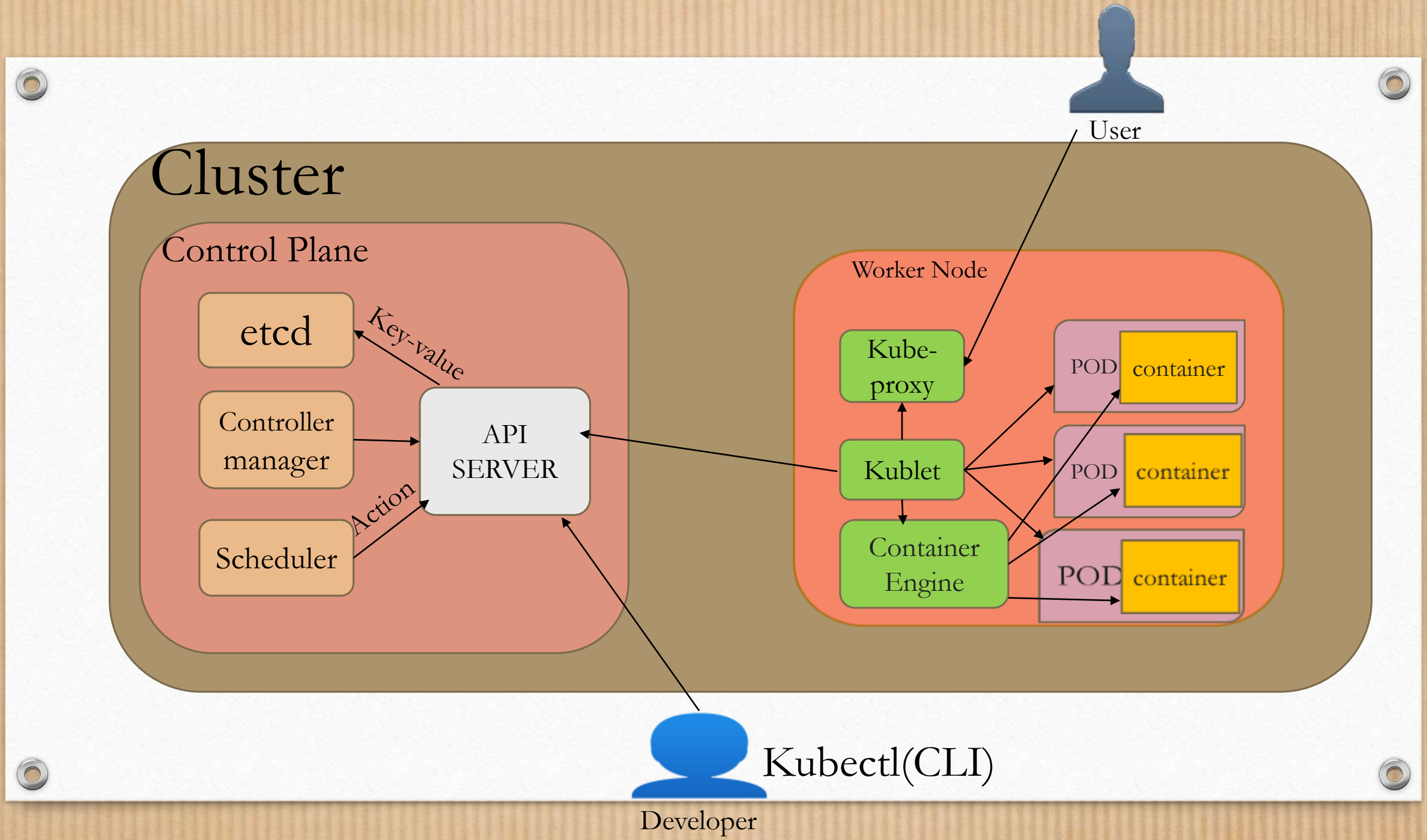
KUBERNETS ARCHITECHURE

The entire Kubernetes architecture revolves around the concept of the cluster. A cluster is a set of nodes, physical or virtual machines that perform various functions. Each cluster consists of:

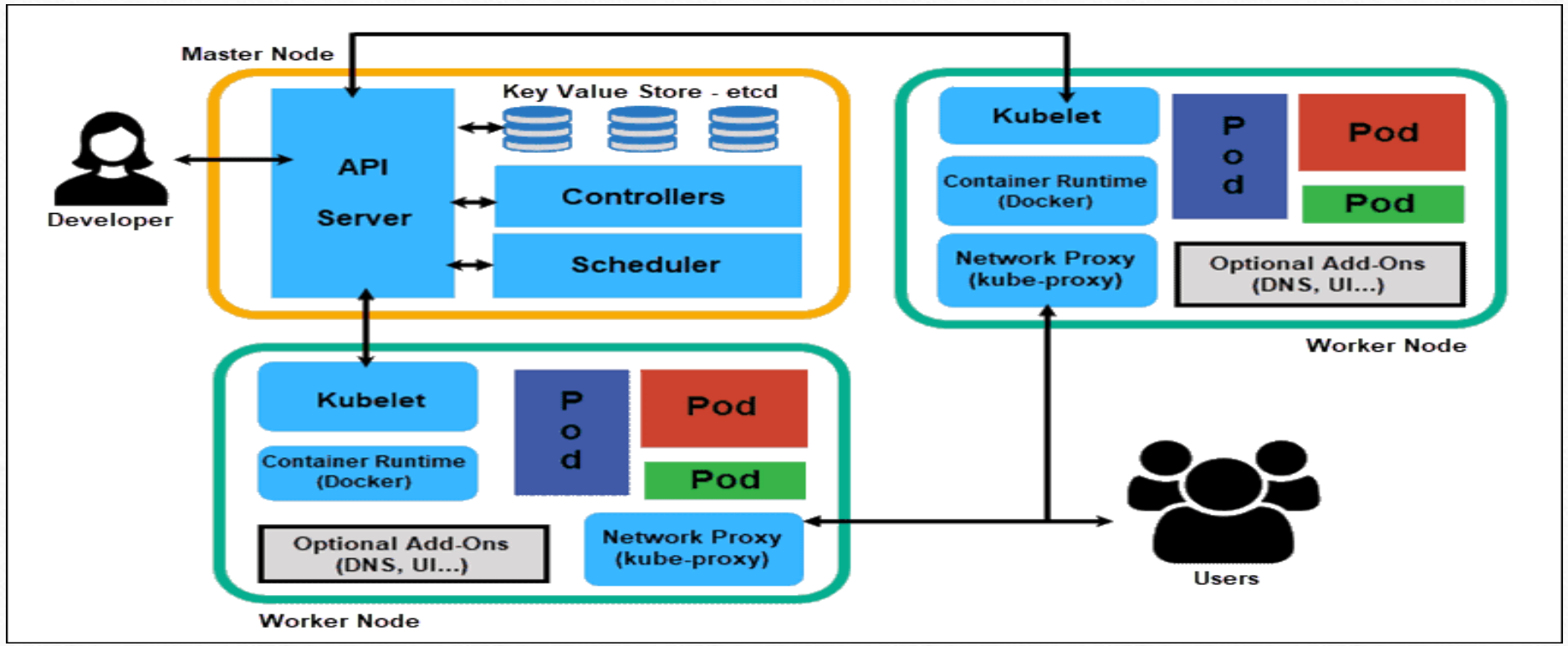
1) **The control plane**
2) **Worker nodes**




User
Cluster
Control Plane
etcd
Key-value
Controller manager
API SERVER
Scheduler
Action
Worker Node
Kube-proxy
Kublet
Container Engine
POD
container
POD
container
POD
container
Developer


User
Cluster
Control Plane
etcd
Key-value
Controller manager
API SERVER
Action
Scheduler
Worker Node
Kube-proxy
Kublet
Container Engine
POD
container
POD
container
POD
container
Kubectl(CLI)
Developer

KUBERNETS ARCHITECHURE



IMPORTANT TERMS

- The control plane has four primary components used to control communications, manage nodes and keep track of the state of a Kubernetes cluster.

1. **Kube-apiserver**. As its name suggests, the kube-apiserver exposes the Kubernetes API.
2. **etcd**. A key-value store where all data relating to the Kubernetes cluster is stored.
3. **Kube-scheduler**. Watches for new Kubernetes Pods with no assigned nodes and assigns them to a node for execution based on resources, policies, and 'affinity' specifications.
4. **Kube-controller-manager**. All controller functions of the control plane are compiled into a single binary: kube-controller-manager.

Kubernetes Benefits

- **Automated Scheduling**
- **Self Healing Capabilities**
- **Automated rollouts & rollback**
- **Horizontal Scaling & Load Balancing**

01

Automated Scheduling

Kubernetes provides advanced scheduler to launch container on cluster nodes

02

Self Healing Capabilities

Rescheduling, replacing and restarting the containers which are died.

03

Automated rollouts and rollback

Kubernetes supports rollouts and rollbacks for the desired state of the containerized application

04

Horizontal Scaling and Load Balancing

Kubernetes can scale up and scale down the application as per the requirements

AGENDA

- What we study previous ? Minikube & it's architecture
- Steps for Installation
- Commands

What will you need ?

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.

All you need is Docker (or similarly compatible) container or a Virtual Machine environment, and Kubernetes is a single command away: `minikube start`

What you'll need

- 2 CPUs or more
- 2GB of free memory
- 20GB of free disk space
- Internet connection
- Container or virtual machine manager, such as: Docker, QEMU, Hyperkit, Hyper-V, KVM,

We already see how to install docker .

https://docs.docker.com/desktop/install/windows-install/

Install Docker Desktop on Windows

Welcome to Docker Desktop for Windows. This page contains information about Docker Desktop for Windows system requirements, download URL, instructions to install and update Docker Desktop for Windows.

Docker Desktop for Windows

For checksums, see Release notes

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

Command for docker install

Steps for Installing Docker:

1. Open the terminal on Ubuntu.

2. Remove any Docker files that are running in the system, using the following command:

```
$ sudo apt-get remove docker docker-engine docker.io
```

After entering the above command, you will need to enter the password of the root and press enter.

3. Check if the system is up-to-date using the following command:

```
$ sudo apt-get update
```

4. Install Docker using the following command:

```
$ sudo apt install docker.io
```

Now let's install Minikube

- **Website direct link : https://minikube.sigs.k8s.io/docs/start/**

Second way from command line

- **Download and run using PowerShell, use this command:**
- **Step 1 : Install from github**

➢ New-Item –Path 'c:\' -Name 'minikube' -ItemType Directory -Force

➢ Invoke-WebRequest –OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing

- **Step 2: setup path**

➢ Add the minikube.exe binary to your PATH.

➢ Make sure to run PowerShell as Administrator.

To install the latest minikube **stable** release on **x86-64 Windows** using **.exe download**:

1. Download and run the installer for the latest release.
   Or if using `PowerShell`, use this command:

```
New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/dc
```

2. Add the `minikube.exe` binary to your `PATH`.
   Make sure to run PowerShell as Administrator.

```
$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ `
  [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machir
}
```

After download Minikube

- **Use Command :-**

1) Minikube start

2) Minikube status

3) kubectl get all

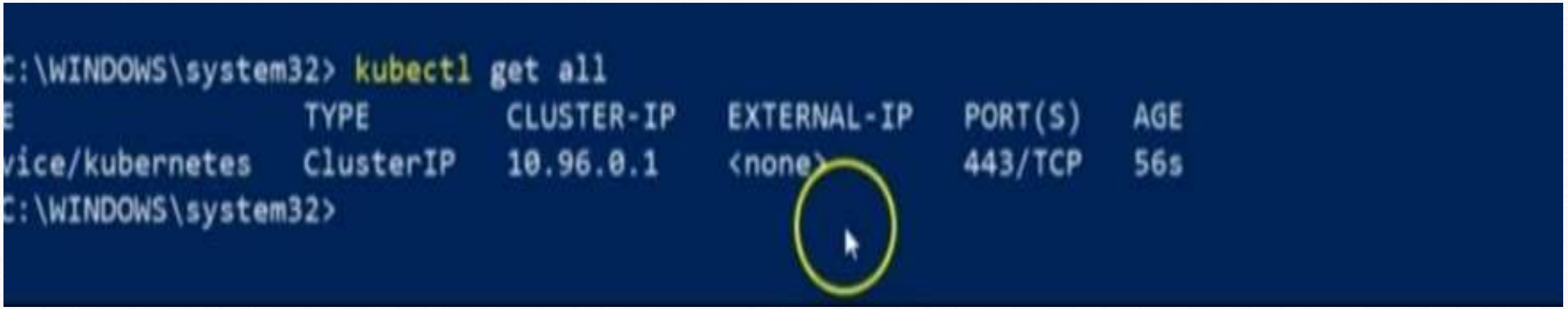
1)Minikube start : start the Minikube

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> minikube start --vm-driver=hyperv
* minikube v1.6.2 on Microsoft Windows 10 Enterprise 10.0.18363 Build 18363
* Selecting 'hyperv' driver from user configuration (alternates: [])
* Downloading VM boot image ...
    > minikube-v1.6.0.iso.sha256: 65 B / 65 B [--------------] 100.00% ? p/s 0s
    > minikube-v1.6.0.iso: 150.93 MiB / 150.93 MiB [-] 100.00% 4.28 MiB p/s 35s
* Creating hyperv VM (CPUs=2, Memory=2000MB, Disk=20000MB) ...
* Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...
* Downloading kubeadm v1.17.0
* Downloading kubelet v1.17.0
* Pulling images ...
* Launching Kubernetes ...
* Waiting for cluster to come online ...
* Done! kubectl is now configured to use "minikube"
PS C:\WINDOWS\system32>
```

2)Minikube status : check the status

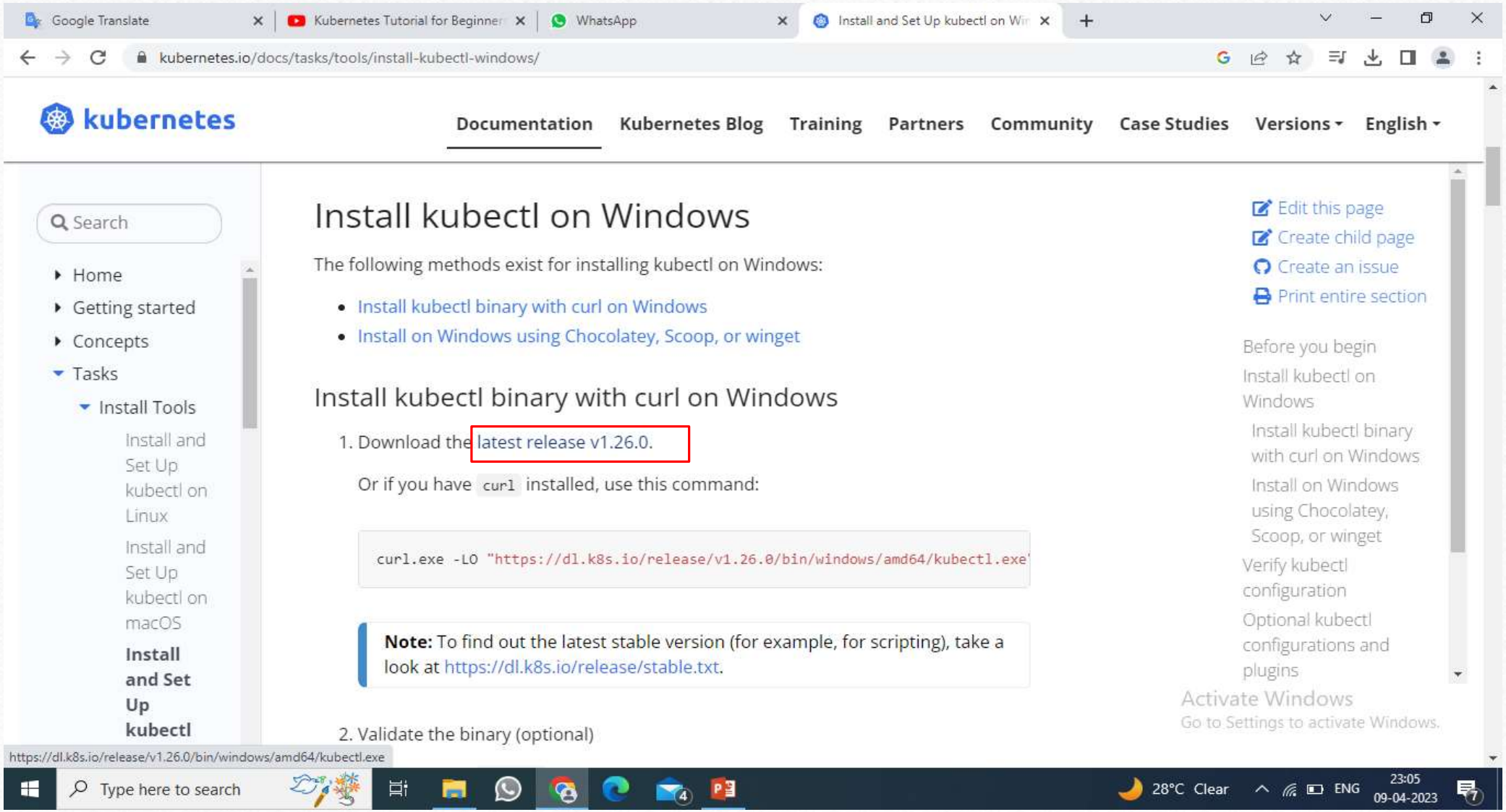
```
:\Program Files\Docker\Docker\resources\bin\kubectl.exe is version 1.25.2, which may have inc
 Want kubectl v1.23.3? Try 'minikube kubectl -- get pods -A'
one! kubectl is now configured to use "minikube" cluster and "default" namespace by default
C:\WINDOWS\system32> minikube status
ikube
e: Control Plane
t: Running
elet: Running
server: Running
econfig: Configured
```

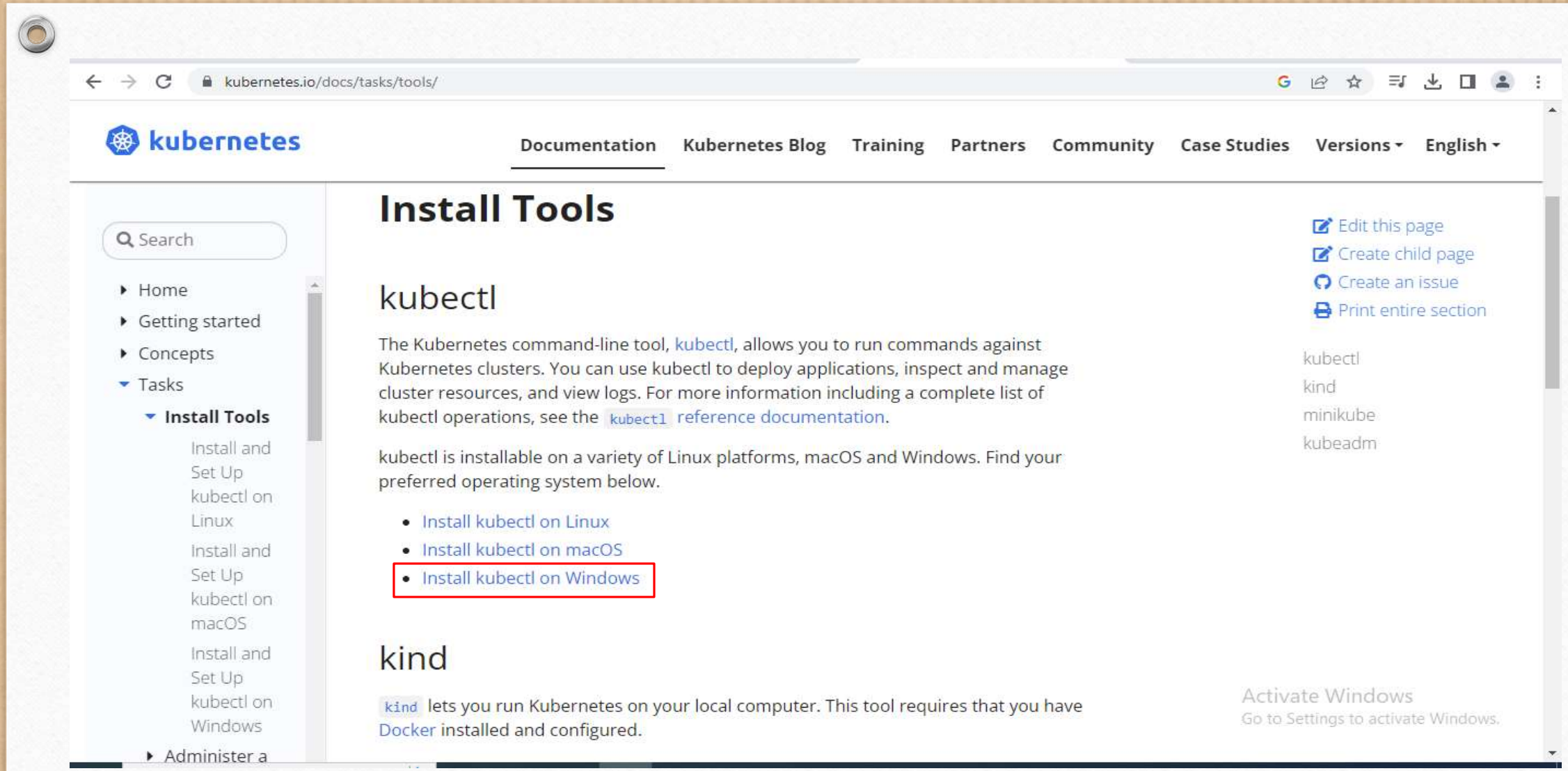
3)Kubectl get all : It display all the services running in our cluster.



1)Let's download 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.**


Google Translate
Kubernetes Tutorial for Beginner
WhatsApp
Install and Set Up kubectl on Win
kubernetes.io/docs/tasks/tools/install-kubectl-windows/
kubernetes
Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Versions
English
Search
Home
Getting started
Concepts
Tasks
Install Tools
Install and Set Up kubectl on Linux
Install and Set Up kubectl on macOS
Install and Set Up kubectl
Install kubectl on Windows
The following methods exist for installing kubectl on Windows:
Install kubectl binary with curl on Windows
Install on Windows using Chocolatey, Scoop, or winget
Install kubectl binary with curl on Windows
1. Download the latest release v1.26.0.
Or if you have curl installed, use this command:
curl.exe -LO "https://dl.k8s.io/release/v1.26.0/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.
2. Validate the binary (optional)
Edit this page
Create child page
Create an issue
Print entire section
Before you begin
Install kubectl on Windows
Install kubectl binary with curl on Windows
Install on Windows using Chocolatey, Scoop, or winget
Verify kubectl configuration
Optional kubectl configurations and plugins
Activate Windows
Go to Settings to activate Windows.
https://dl.k8s.io/release/v1.26.0/bin/windows/amd64/kubectl.exe
Type here to search
28°C Clear
ENG
23:05
09-04-2023


kubernetes.io/docs/tasks/tools/
kubernetes
Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Versions
English
Search
Home
Getting started
Concepts
Tasks
Install Tools
Install and Set Up kubectl on Linux
Install and Set Up kubectl on macOS
Install and Set Up kubectl on Windows
Administer a
Install Tools
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. For more information including a complete list of kubectl operations, see the kubectl reference documentation.
kubectl is installable on a variety of Linux platforms, macOS and Windows. Find your preferred operating system below.
Install kubectl on Linux
Install kubectl on macOS
Install kubectl on Windows
kind
kind lets you run Kubernetes on your local computer. This tool requires that you have Docker installed and configured.
Edit this page
Create child page
Create an issue
Print entire section
kubectl
kind
minikube
kubeadm
Activate Windows
Go to Settings to activate Windows.

Kubernetes commands

- **Cluster Management**
- **Deployments**
- **Daemonsets**

1) Cluster Management

Display endpoint information about the master and services in the cluster

```
kubectl cluster-info
```

Display the Kubernetes version running on the client and server

```
kubectl version
```

1) Cluster Management

Get the configuration of the cluster

```
kubectl config view
```

List the API resources that are available

```
kubectl api-resources
```

1) Cluster Management

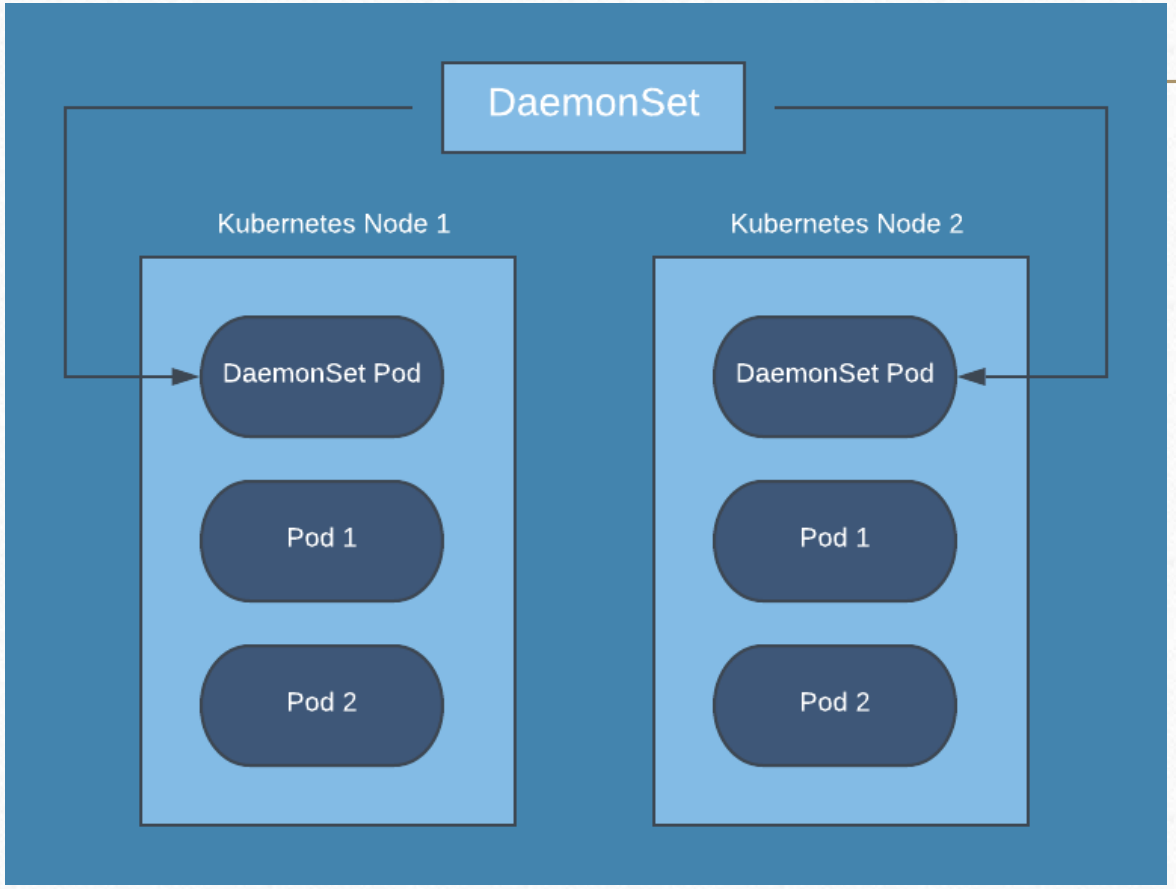
List the API versions that are available

```
kubectl api-versions
```

List everything

```
kubectl get all --all-namespaces
```

2) Daemonsets



- **A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them.**
- **As nodes are removed from the cluster, those Pods are garbage collected.**
- **Deleting a DaemonSet will clean up the Pods it created.**

2) Daemonsets

Shortcode = ds

List one or more daemonsets

```
kubectl get daemonset
```

Edit and update the definition of one or more daemonset

```
kubectl edit daemonset <daemonset_name>
```

2) Daemonsets

Delete a daemonset

```
kubectl delete daemonset <daemonset_name>
```

Create a new daemonset

```
kubectl create daemonset <daemonset_name>
```

2) Daemonsets

Manage the rollout of a daemonset

```
kubectl rollout daemonset
```

Display the detailed state of daemonsets within a namespace

```
kubectl describe ds <daemonset_name> -n <namespace_name>
```

3)Deployment : A Kubernetes Deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application.

Deployments

Shortcode = deploy

List one or more deployments

```
kubectl get deployment
```

Display the detailed state of one or more deployments

```
kubectl describe deployment <deployment_name>
```

3) Deployments

Edit and update the definition of one or more deployment on the server

```
kubectl edit deployment <deployment_name>
```

Create one a new deployment

```
kubectl create deployment <deployment_name>
```

3) Deployments

Delete deployments

```
kubectl delete deployment <deployment_name>
```

See the rollout status of a deployment

```
kubectl rollout status deployment <deployment_name>
```

AGENDA

- Deployment Minikube and kubectl
- Deploy web page.

STEPS

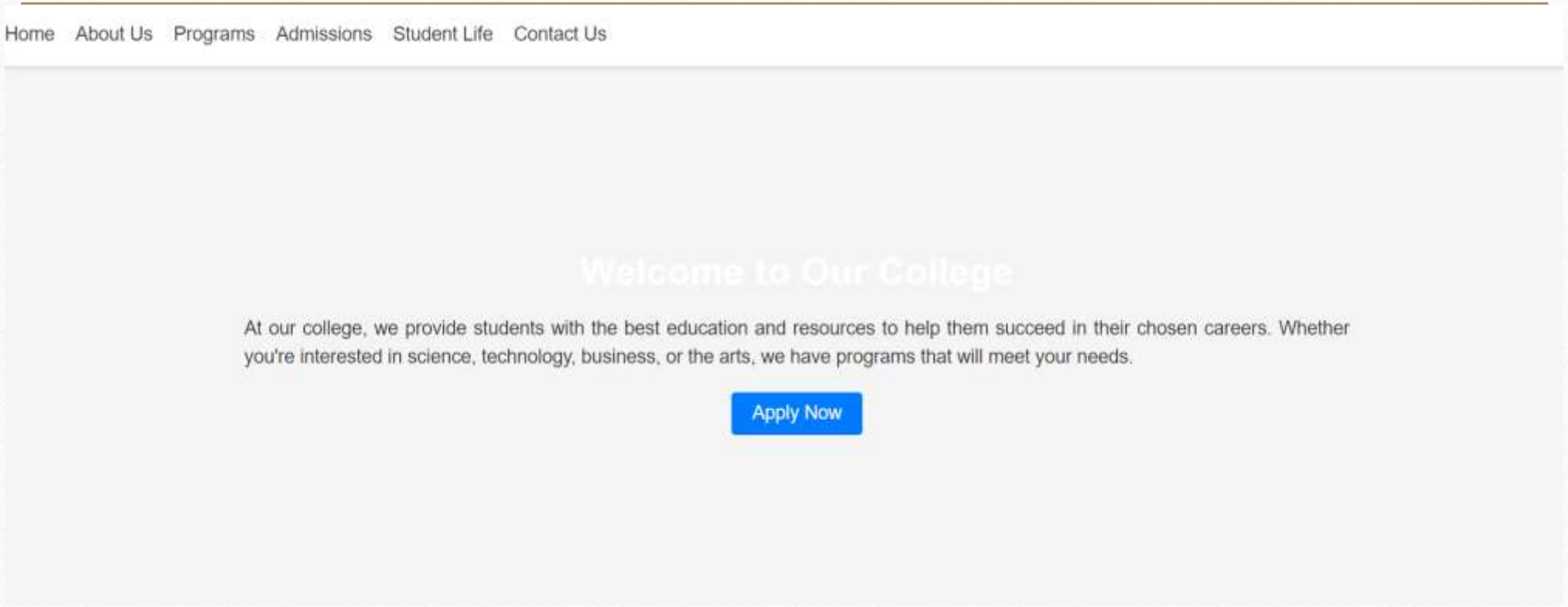
1. Install Minikube , kubectl and docker
2. Start Minikube using command
3. **Create docker image and pods**
4. **Deploy on docker site and on Minikube dashboard(local host)**

```
amja@DESKTOP-6LACVB5 MINGW64 /c/minikube
$ minikube version
minikube version: v1.30.1
commit: 08896fd1dc362c097c925146c4a0d0dac715ace0

amja@DESKTOP-6LACVB5 MINGW64 /c/minikube
$
```

```
amja@DESKTOP-6LACVB5 MINGW64 /c/minikube
$ kubectl version
WARNING: This version information is deprecated and will be replaced with the output from kubectl version --short.  Use --output=yaml|
on to get the full version.
Client Version: version.Info{Major:"1", Minor:"27", GitVersion:"v1.27.0", GitCommit:"1b4df30b3cdfeaba6024e81e559a6cd09a089d65", GitTre
tate:"clean", BuildDate:"2023-04-11T17:10:18Z", GoVersion:"go1.20.3", Compiler:"gc", Platform:"windows/amd64"}
Kustomize Version: v5.0.1
Server Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.3", GitCommit:"9e644106593f3f4aa98f8a84b23db5fa378900bd", GitTre
tate:"clean", BuildDate:"2023-03-15T13:33:12Z", GoVersion:"go1.19.7", Compiler:"gc", Platform:"linux/amd64"}

amja@DESKTOP-6LACVB5 MINGW64 /c/minikube
```

➢ What we want to deploy ?

We can deploy html web site to the Minikube dashboard and docker.



Start cluster

- Minikube start

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube
$ minikube version
minikube version: v1.30.1
commit: 08896fd1dc362c097c925146c4a0d0dac715ace0

iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube
$ minikube start
* minikube v1.30.1 on Microsoft Windows 11 Home Single Language 10.0.22621.1555
Build 22621.1555
* Automatically selected the docker driver. Other choices: virtualbox, ssh
* Using Docker Desktop driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.26.3 preload ...
    > gcr.io/k8s-minikube/kicbase...:  373.53 MiB / 373.53 MiB  100.00% 2.38 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
* Done! kubectl is now configured to use "minikube" cluster and "default" namesp
ace by default
```

File we want for a deployment

1. Source code files (index file , read me file) – **our project file**
2. Python (webapp.py) file to connect to the server.
3. Docker image file
4. Jason file or Yaml file

Webapp.py file for a connect to the server

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ cat webapp.py
import http.server
import socketserver

PORT = 8000

Handler = http.server.SimpleHTTPRequestHandler
httpd = socketserver.TCPServer(("", PORT), Handler)

print(f"serving at port {PORT}")
httpd.serve_forever()
```

Check webapp.py file is working or not

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ python webapp.py
serving at port 8000
```

STEPS

1. Install Minikube , kubectl and docker ☑
2. Start Minikube using command ☑
3. **Create docker image and pods**
4. **Deploy on docker site and on Minikube dashboard(local host)**

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ cat > dockerfile
FROM ubuntu:18.04
RUN apt-get update \
&& apt install -y --no-install-recommends curl && apt-get install -y --no-install-recommends python

WORKDIR  /
COPY . .
ENTRYPOINT ["python", "webapp.py"]
```

- Use command docker build to create a docker file

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker build -t collegesite:v1 .
#2 [internal] load .dockerignore
#2 sha256:86bc6f53b5f234bfec2a885fb72ea514ff4fb569c740ccdbd0962a50d3e20a8a
#2 transferring context: 2B 0.0s done
#2 DONE 0.1s

#1 [internal] load build definition from Dockerfile
#1 sha256:3bb80f0702847ff4c9474a7e1959893dbd491665b3fd2716ee5b81e2674f16f1
#1 transferring dockerfile: 297B 0.0s done
#1 DONE 0.1s

#3 [internal] load metadata for docker.io/library/ubuntu:18.04
#3 sha256:ae46bbb1b755529d0da663ca0256a22acd7c9fe21844946c149800baa67c4e4b
#3 ...

#4 [auth] library/ubuntu:pull token for registry-1.docker.io
#4 sha256:c1b11a9c90ac6bd91763178d224f0709f517e9b2cbe9ea36caf01e707fdb1659
#4 DONE 0.0s

#3 [internal] load metadata for docker.io/library/ubuntu:18.04
#3 sha256:ae46bbb1b755529d0da663ca0256a22acd7c9fe21844946c149800baa67c4e4b
#3 DONE 4.1s

#7 [internal] load build context
#7 sha256:041b08cab6f440b2ba81e9d2392db216194737e42b7ae1c3d6e258aff2cba1d5
#7 transferring context: 6.42kB done
#7 DONE 0.0s

#5 [1/4] FROM docker.io/library/ubuntu:18.04@sha256:8aa9c2798215f99544d1ce7439ea9c3a6dfd82de607da1cec3a8a2fae005931b
#5 sha256:bc8b512a8dde37b76679042ab31e03fc63f28df17c84012f59f6e0d5b3c42cd6
#5 resolve docker.io/library/ubuntu:18.04@sha256:8aa9c2798215f99544d1ce7439ea9c3a6dfd82de607da1cec3a8a2fae005931b 0.0s done
#5 sha256:8aa9c2798215f99544d1ce7439ea9c3a6dfd82de607da1cec3a8a2fae005931b 1.33kB / 1.33kB done
```

➢ Create pod Using Json/Yaml file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
   name: http-server
   labels:
      app: http-server
spec:
   replicas: 3
   selector:
      matchLabels:
         app: http-server
   template:
      metadata:
         labels:
            app: http-server
      spec:
         containers:
         - name: http-server
           image: collegesite:v1
           imagePullPolicy: IfNotPresent
           ports:
           - containerPort: 8000
```

Check our pod is created or not using command

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ minikube kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
http-server-759ddbd497-bbxt7        0/1     ErrImagePull       0          2m11s
http-server-759ddbd497-n7vmt        0/1     ImagePullBackOff   0          2m11s
http-server-759ddbd497-pnqrb        0/1     ImagePullBackOff   0          2m11s
nginx                               1/1     Running            0          103m
```

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker image ls
REPOSITORY                          TAG       IMAGE ID       CREATED          SIZE
collegesite                         v1        3a75ec8e30ce   17 minutes ago   144MB
gcr.io/k8s-minikube/kicbase         v0.0.39   67a4b1138d2d   12 days ago      1.05GB
```

STEPS

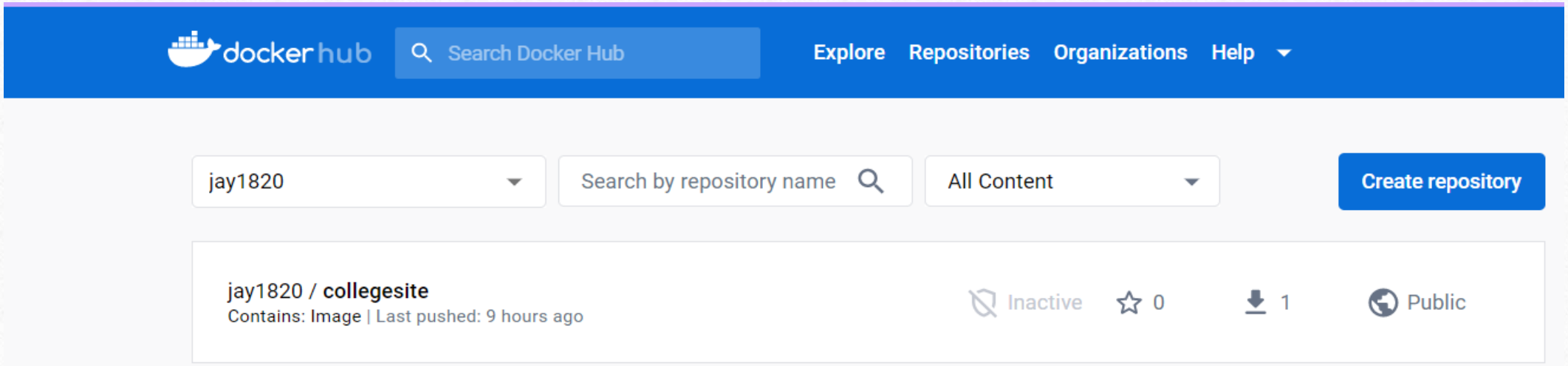
1. Install Minikube , kubectl and docker ☑
2. Start Minikube using command ☑
3. **Create docker image and pods ☑**
4. **Deploy on docker site**
    - **Login into docker**
    - **Push docker image**

➢ Login into docker site

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker login --username jay1820
Error: Cannot perform an interactive login from a non TTY device

iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ winpty docker login --username jay1820
Password:
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
```

➢ Push image into docker site

- **Push our image on https://hub.docker.com/repository/docker/jay1820/collegesite/general**
- **Create docker repository for a project**



Push docker image on docker site

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker tag collegesite:v1 jay1820/collegesite:v1

iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker push jay1820/collegesite:v1
The push refers to repository [docker.io/jay1820/collegesite]
cc161dc2a69c: Preparing
15053d3c2f57: Preparing
b7e0fa7bfe7f: Preparing
cc161dc2a69c: Pushed
b7e0fa7bfe7f: Pushed
15053d3c2f57: Pushed
v1: digest: sha256:560bd76a84f5ba84333d88c1161829485790c2b3e25706aade274f6d0fbe28a4 size: 949
```

Push docker image on docker site

jay1820 / **collegesite**

Description

college website

Last pushed: 9 hours ago

Docker commands

Public View

To push a new tag to this repository,

```
docker push jay1820/collegesite:tagname
```

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
v1		Image	9 hours ago	9 hours ago

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. Read more about automated builds.

Pull docker image from other account pc

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ docker pull jay1820/collegesite:v1
v1: Pulling from jay1820/collegesite
Digest: sha256:560bd76a84f5ba84333d88c116182948579Oc2b3e25706aade274f6d0fbe28a4
Status: Image is up to date for jay1820/collegesite:v1
docker.io/jay1820/collegesite:v1
```

STEPS

1. Install Minikube , kubectl and docker ☑
2. Start Minikube using command ☑
3. **Create docker image and pods ☑**
4. **Deploy on docker site ☑**
    - **Login into docker ☑**
    - **Push docker image ☑**
5. **Deploy on Minikube dashboard localhost**

Deployment on localhost Minikube

STEPS:-

1)Minikube kubectl get pods

2)kubectl – gets deploy http-server –o Yaml

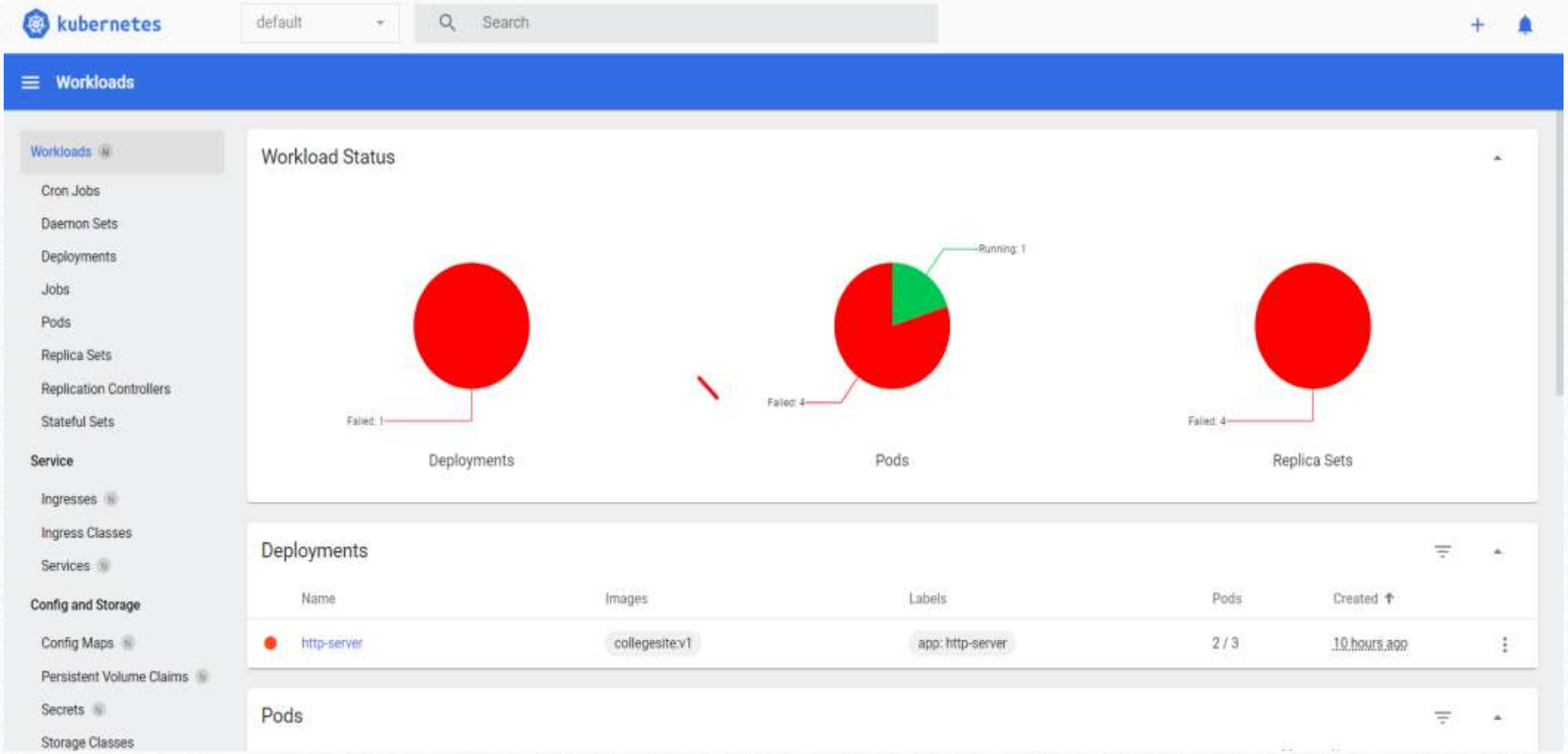
3)Minikube dashboard

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ minikube kubectl get pods
NAME                           READY   STATUS             RESTARTS        AGE
http-server-558b8666bf-b2tbk   0/1     CrashLoopBackOff   33 (93s ago)    9h
http-server-759ddbd497-bbxt7   0/1     ImagePullBackOff   0               9h
http-server-759ddbd497-n7vmt   0/1     ImagePullBackOff   0               9h
http-server-88fd498f-mr9c8     0/1     CrashLoopBackOff   32 (57s ago)    9h
nginx                          1/1     Running            1 (9h ago)      11h
```

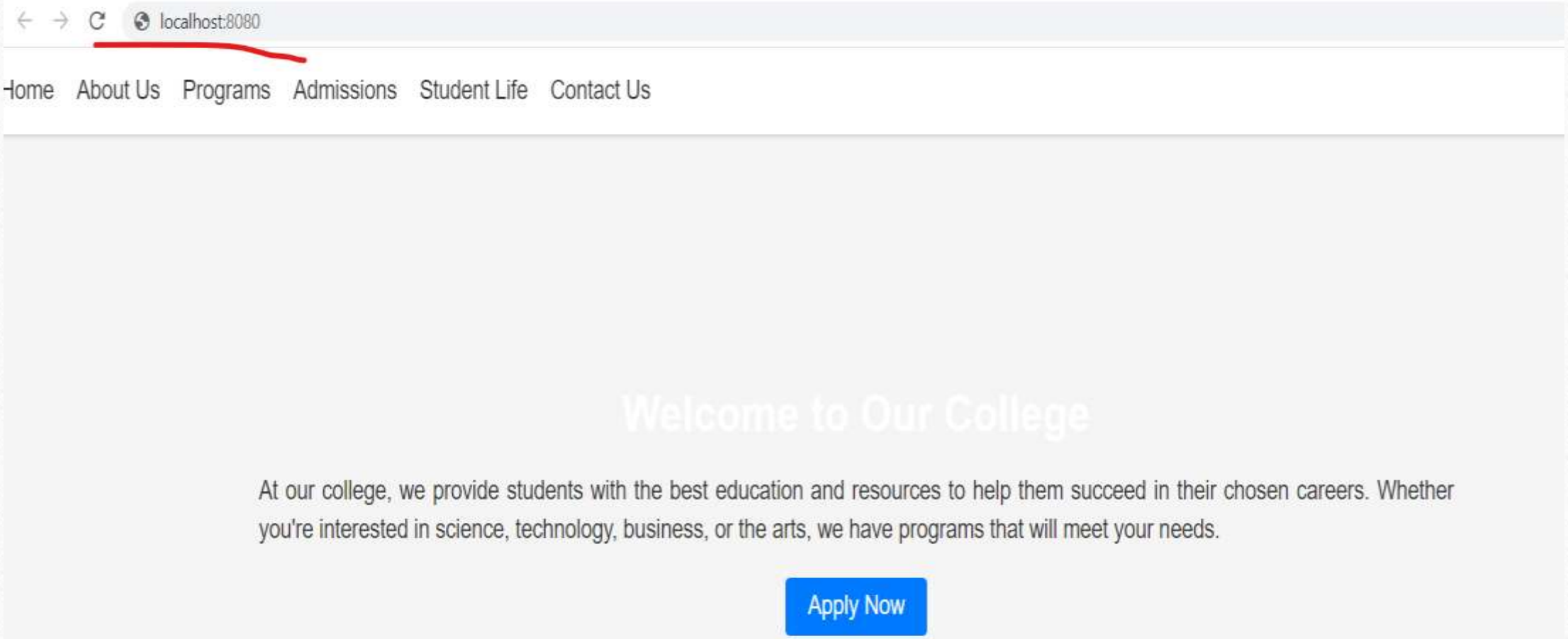
2)kubectl – gets deploy http-server –o yaml

```
iamja@DESKTOP-6LACVB5 MINGW64 /c/minikube/deployment
$ kubectl -- gets deploy http-server -o yaml
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create          Create a resource from a file or from stdin
  expose          Take a replication controller, service, deployment or pod and expose it as a new Kube
rnetes service
  run             Run a particular image on the cluster
  set             Set specific features on objects

Basic Commands (Intermediate):
  explain         Get documentation for a resource
  get             Display one or many resources
  edit            Edit a resource on the server
  delete          Delete resources by file names, stdin, resources and names, or by resources and label
 selector

Deploy Commands:
  rollout         Manage the rollout of a resource
  scale           Set a new size for a deployment, replica set, or replication controller
  autoscale       Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate     Modify certificate resources.
  cluster-info    Display cluster information
  top             Display resource (CPU/memory) usage
  cordon          Mark node as unschedulable
  uncordon        Mark node as schedulable
  drain           Drain node in preparation for maintenance
  taint           Update the taints on one or more nodes
```

3) Use command – Minikube dashboard



4)Finally run on localhost



Thank You!