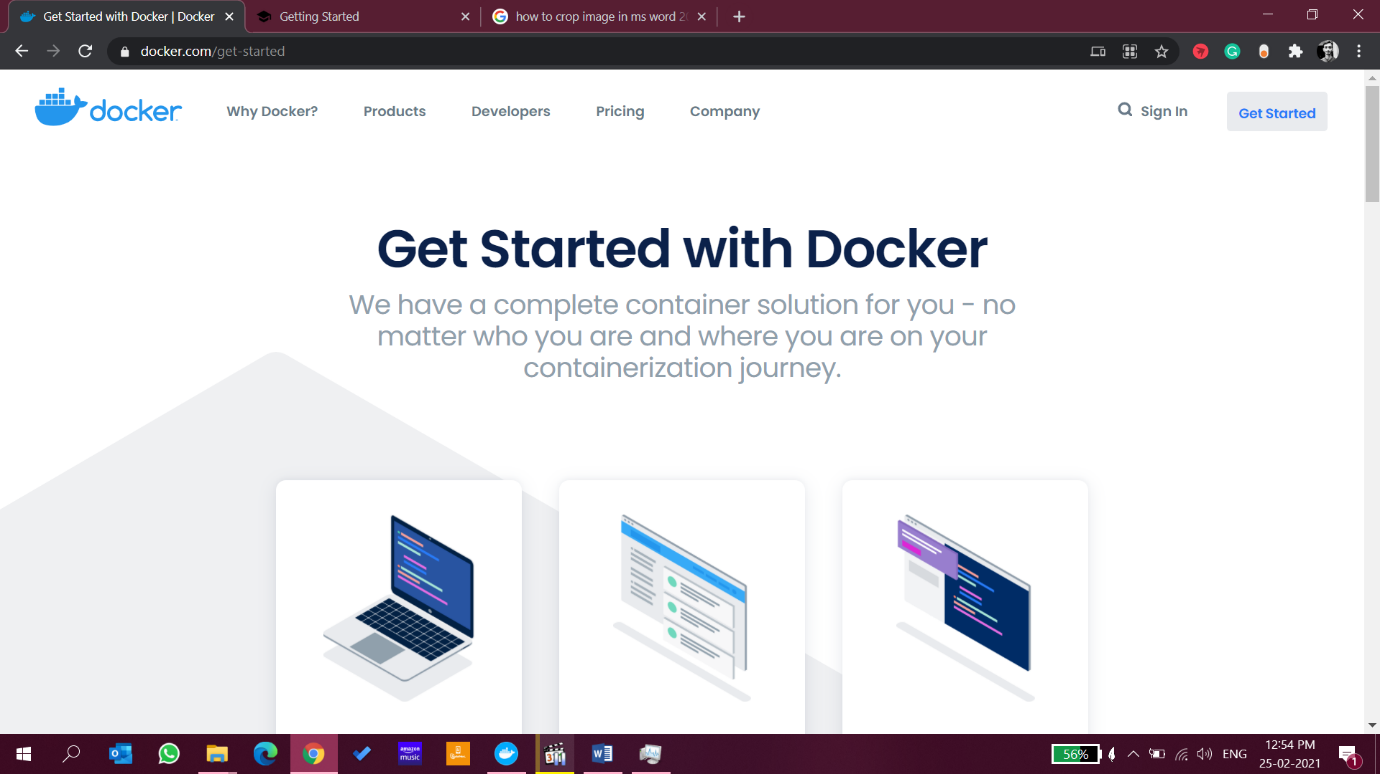
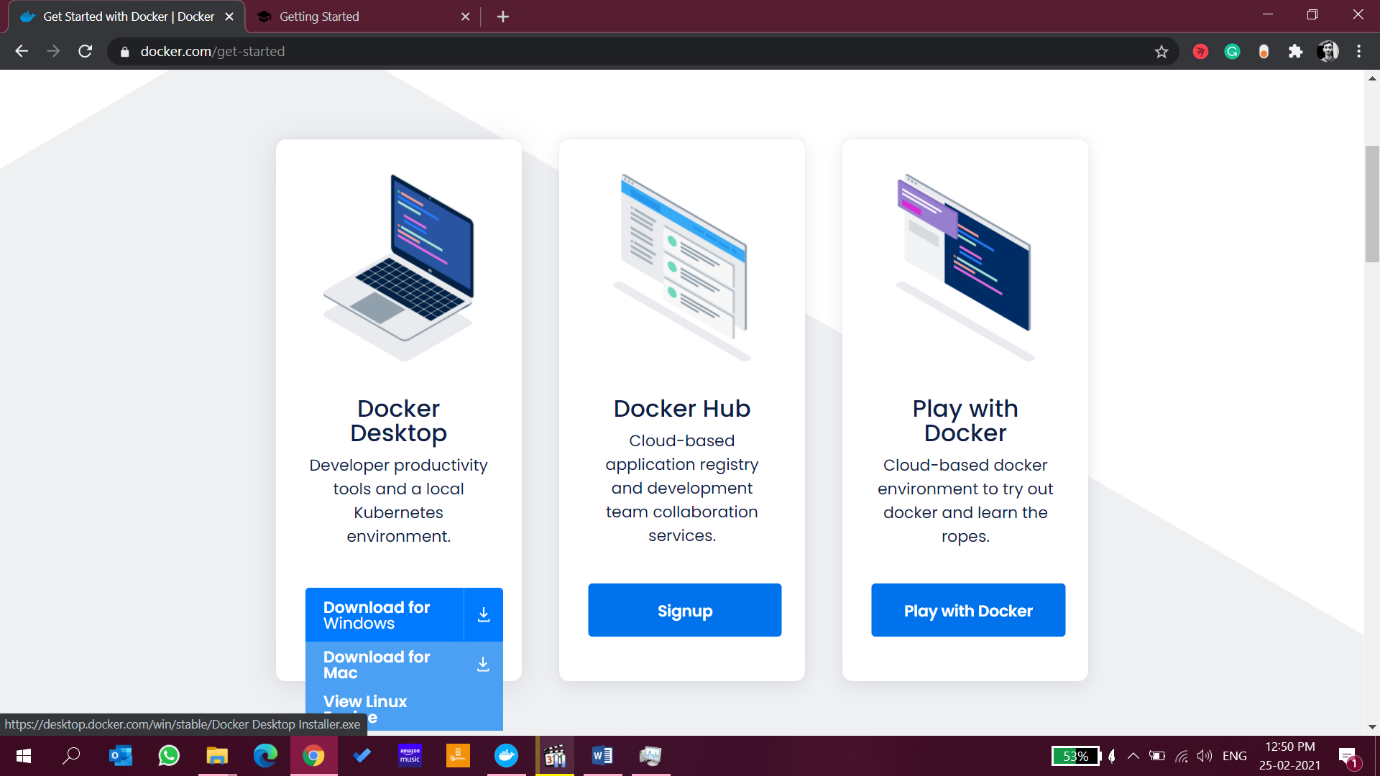
**DOCKER**

**How to install Docker on windows?**

First of all go to <https://www.docker.com/get-started>.

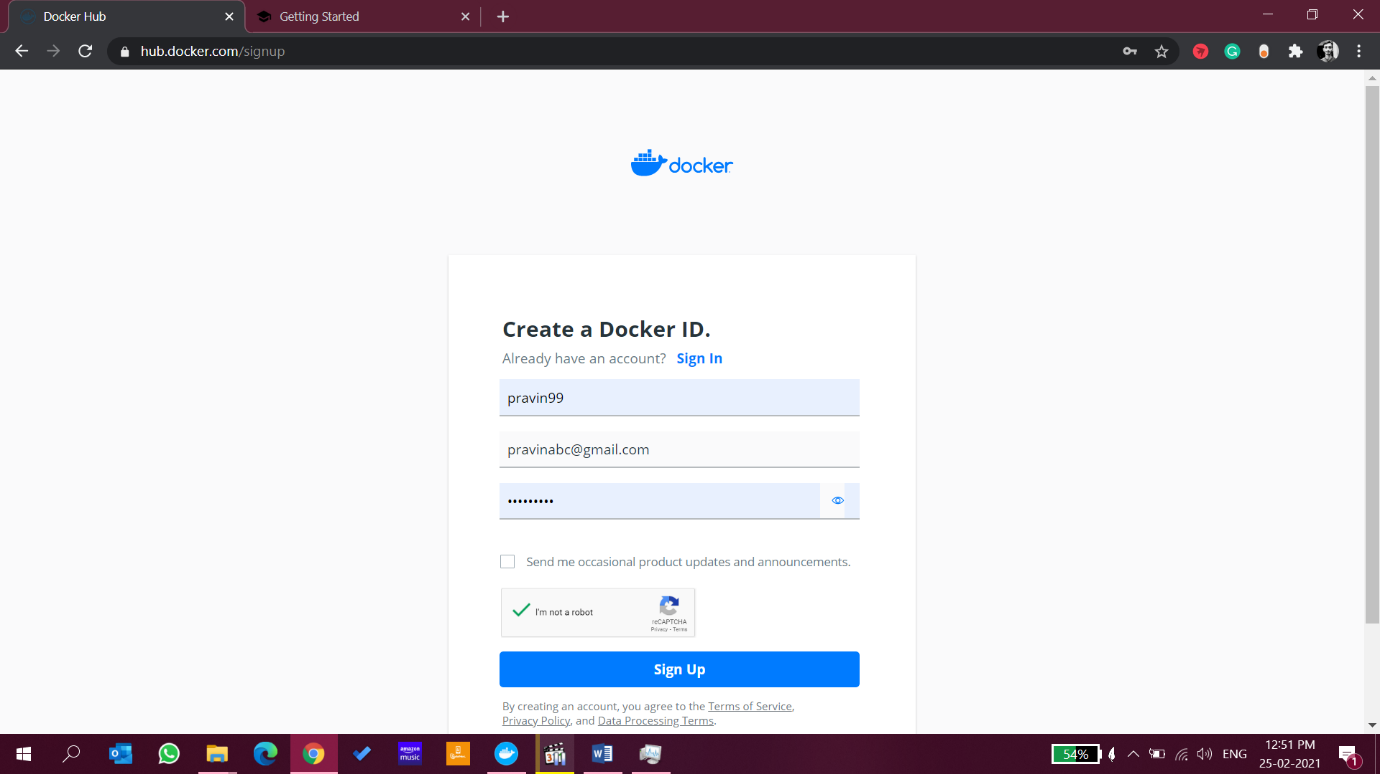


Scroll down a little bit you will find a section as Docker Desktop, Click on Download for windows



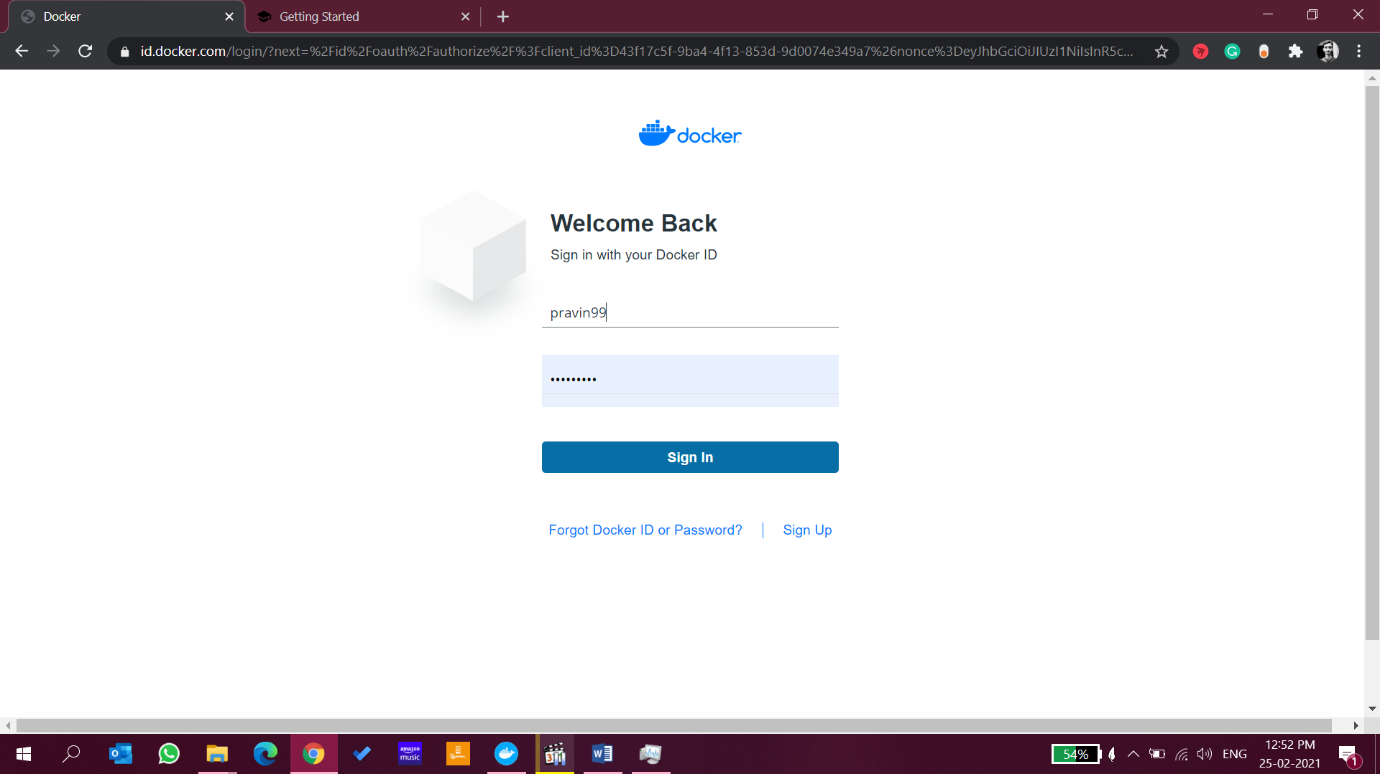
You will find a download prompt for downloading a file regarding the same.

Now, you have to signup to use Docker account. For that you have to go to <https://hub.docker.com/signup>

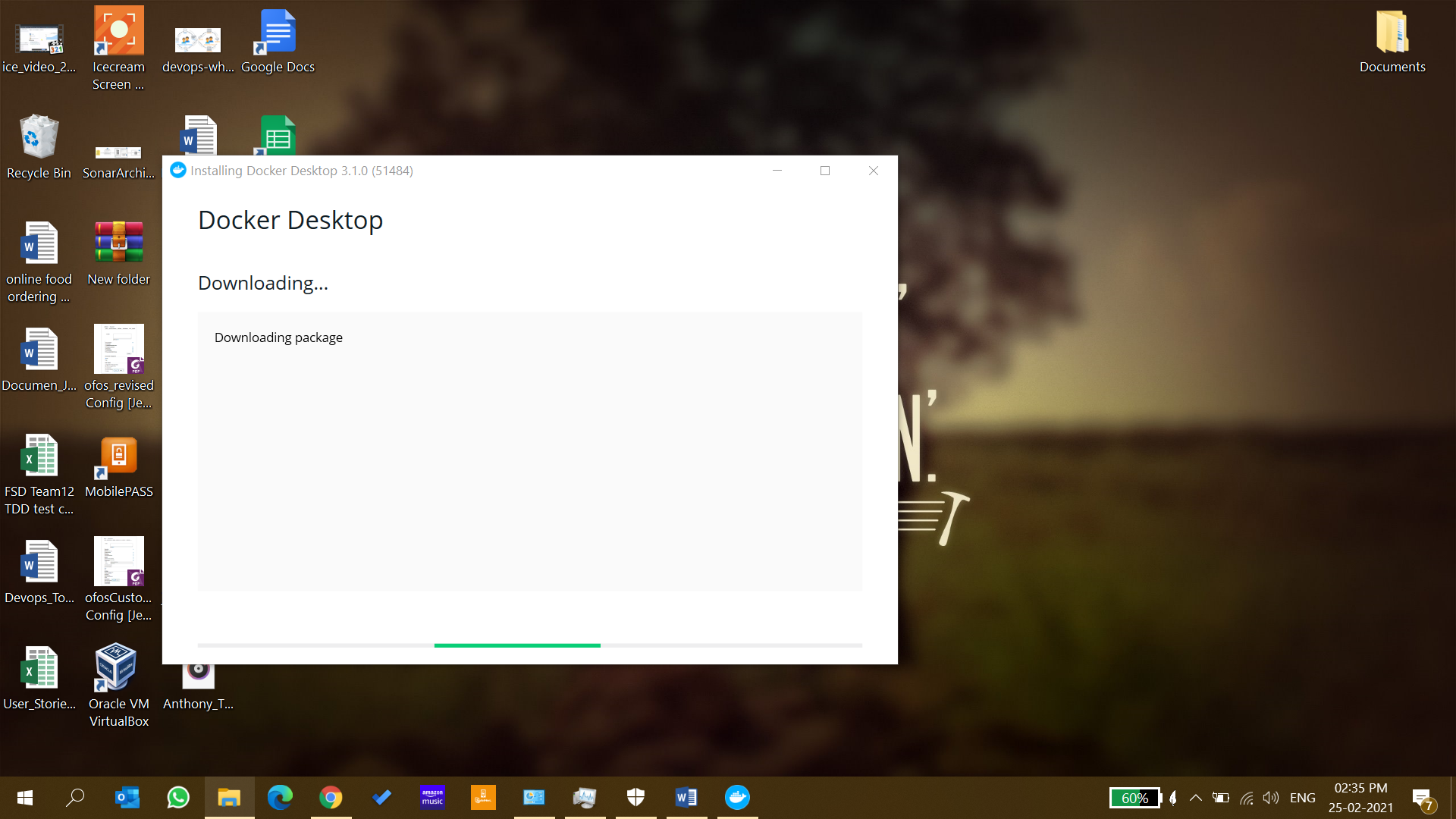


Keep in mind for DockerID you have to enter the Id which you want as your username and same will be visible to others.

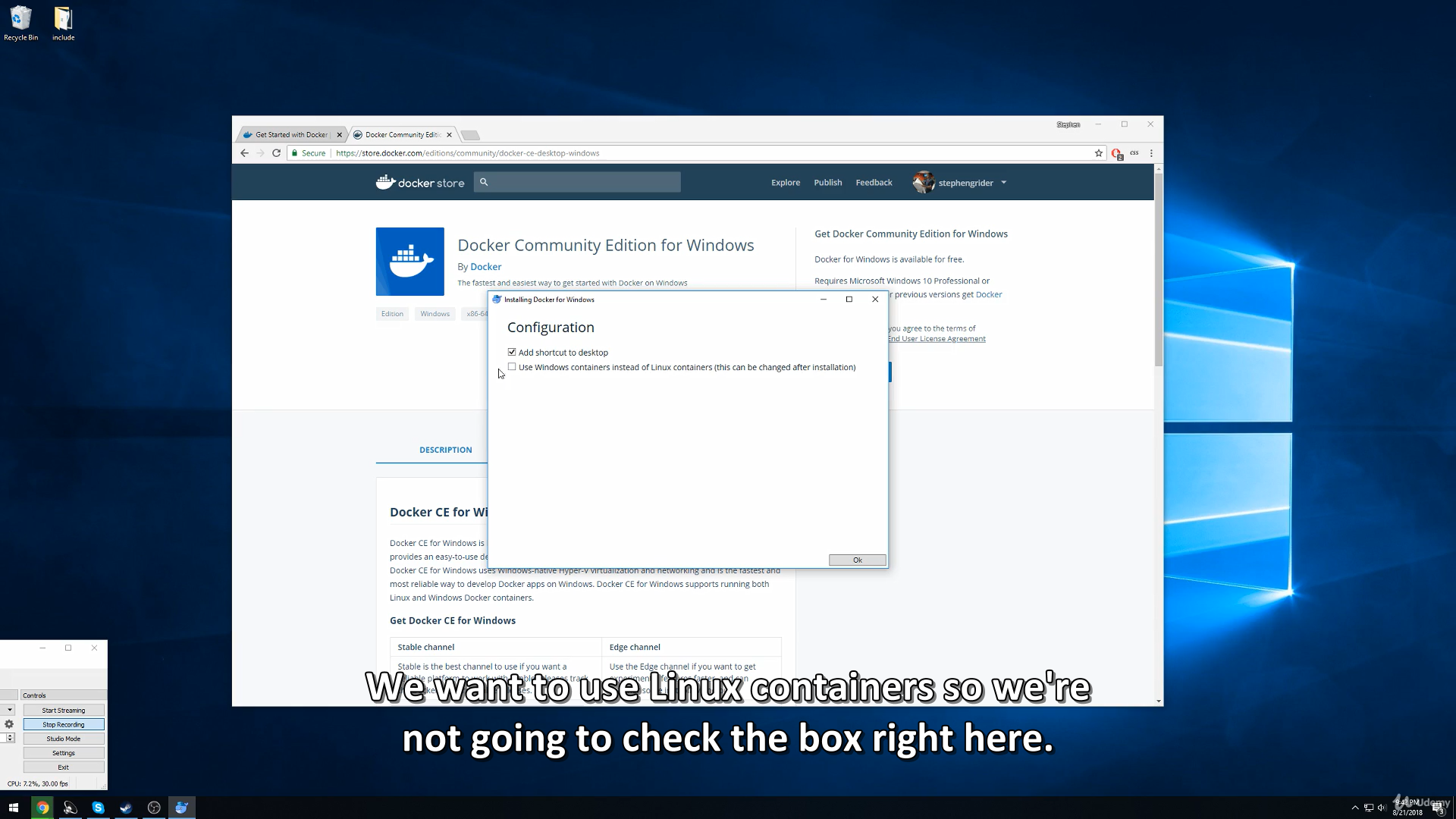
After successful signup you can login in your account.



Now click on Docker Setup installer,



Wait for some time, installer is downloading some packages and important installation files.



Click on OK and proceed for installation

Keep in mind, Virtualization should be enabled on the system before installing Docker.

After installation is completed click on restart now, it will apply some updates to the system.

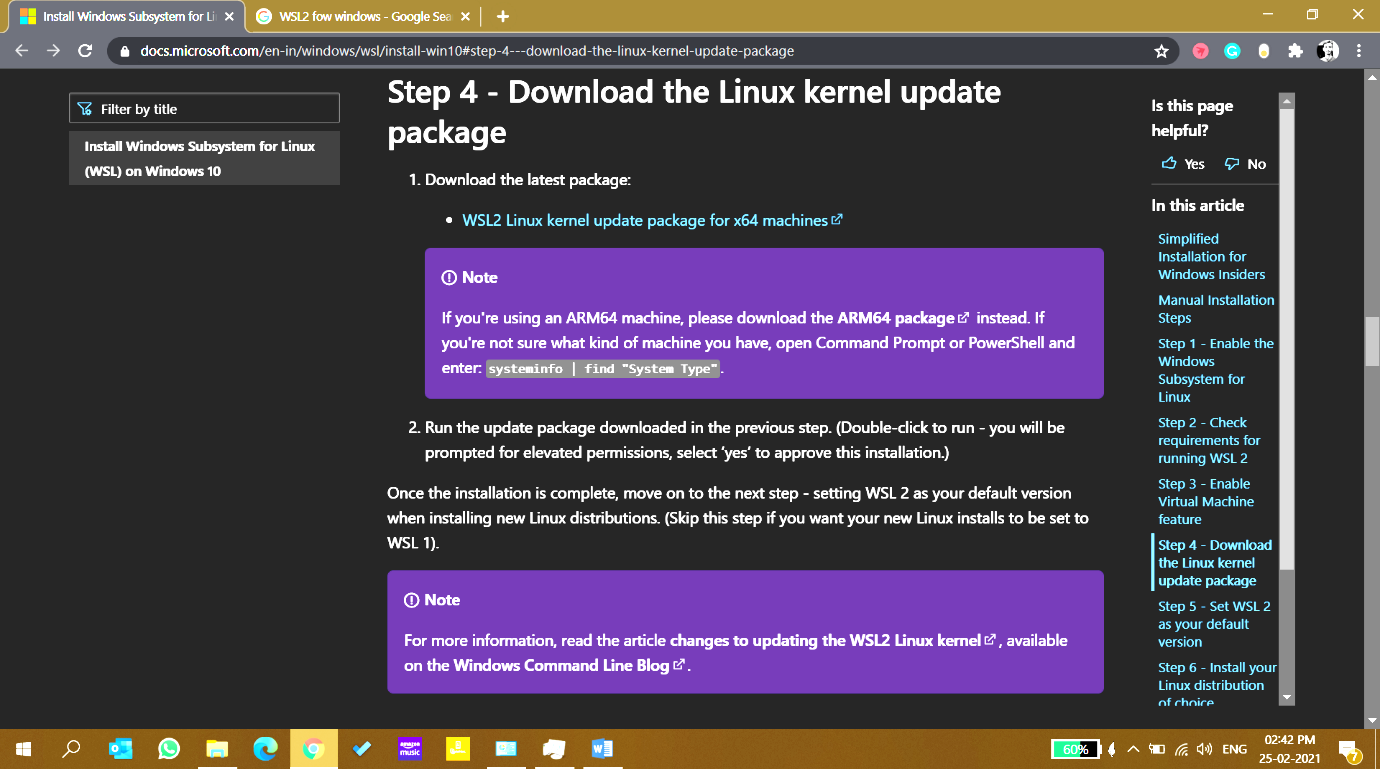
**Windows Subsystem for Linux**

Windows Subsystem for Linux is a compatibility layer for running Linux binary executables natively on Windows 10 and Windows Server 2019. In May 2019, WSL 2 was announced, introducing important changes such as a real Linux kernel, through a subset of Hyper-V features.

After the pc is restarted, click on “Docker Desktop” icon on desktop.

It will prompt a message to download the **Linux kernel update package.**

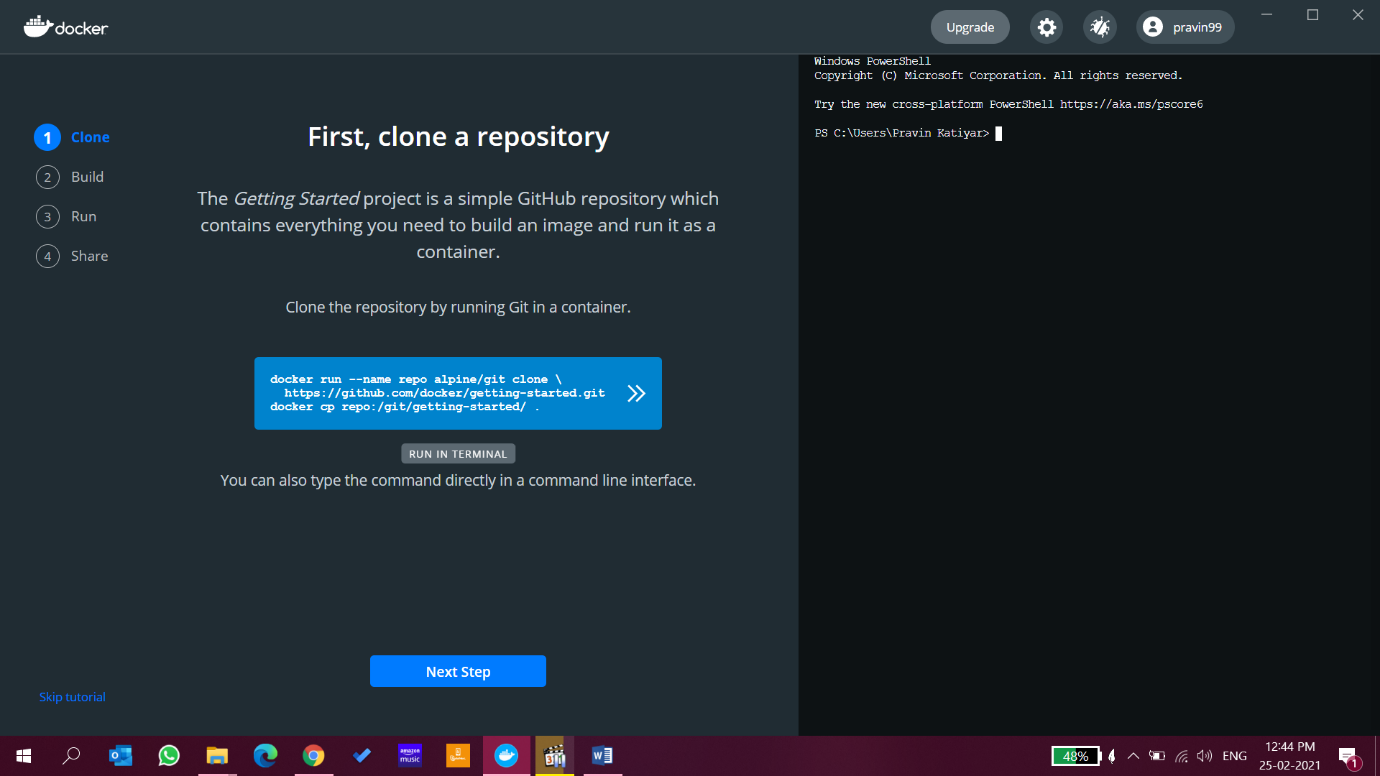
Now, go to <https://docs.microsoft.com/en-in/windows/wsl/install-win10#step-4---download-the-linux-kernel-update-package>

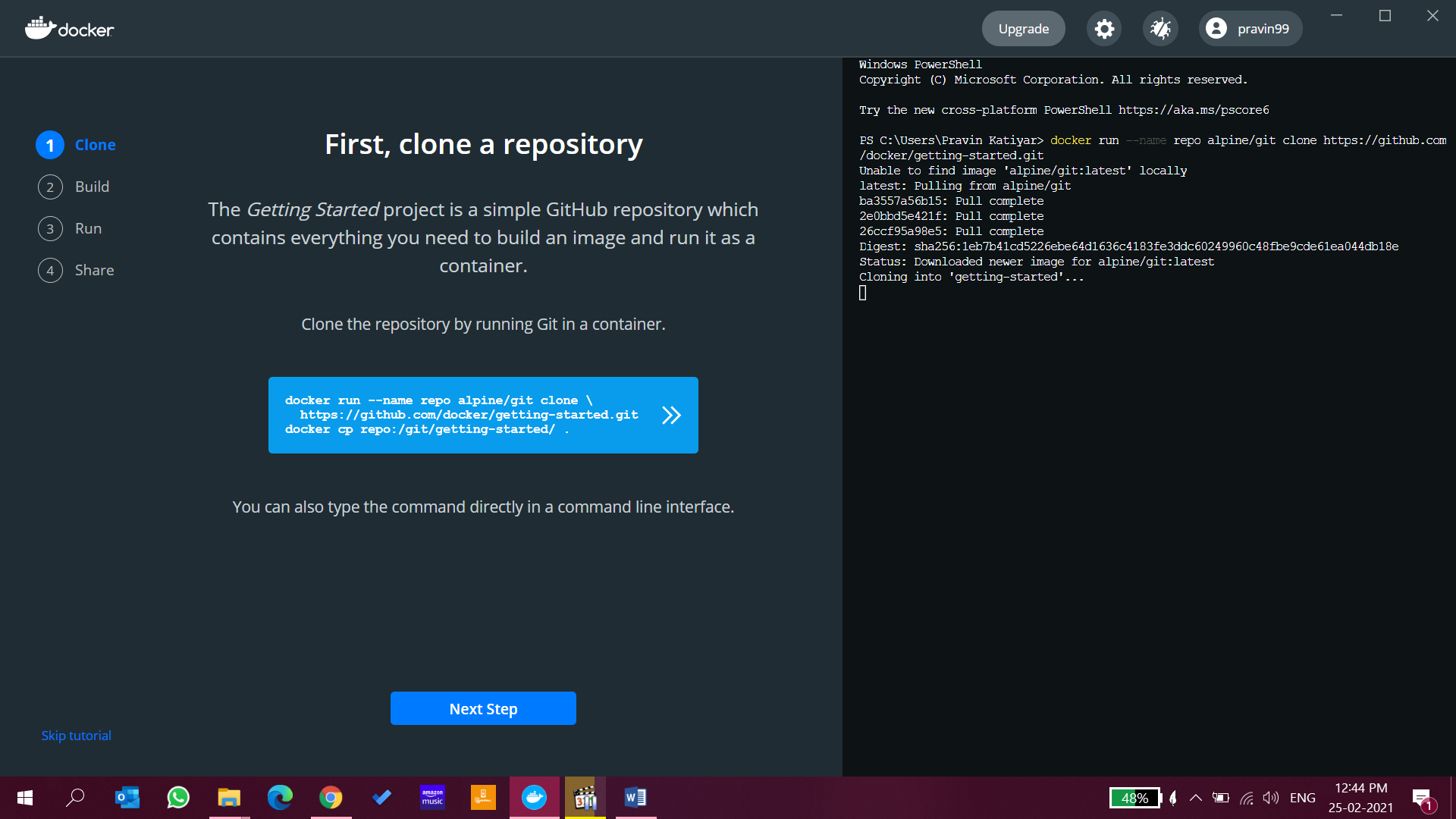


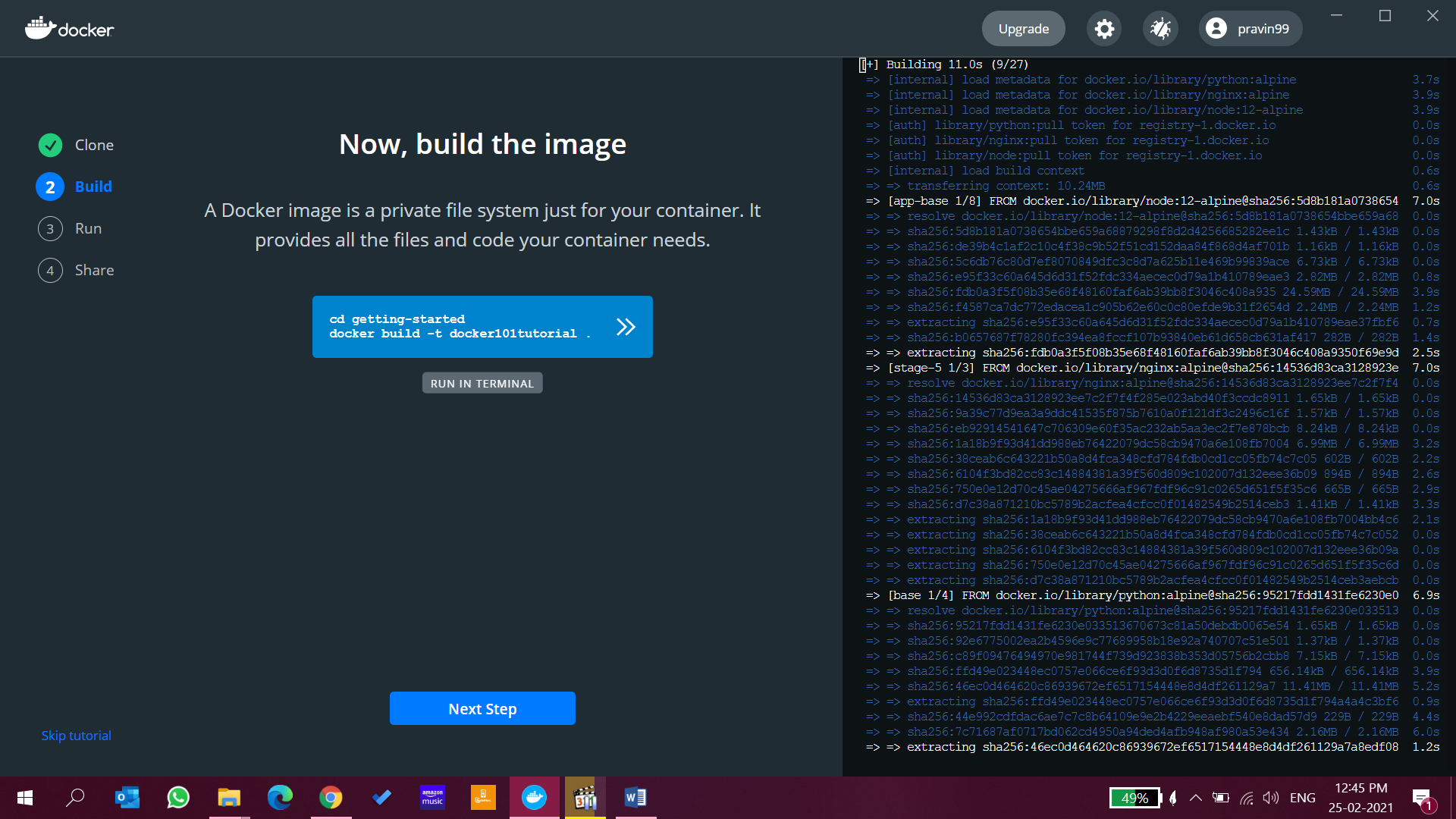
Click on [WSL2 Linux kernel update package for x64 machines](https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi) to download.

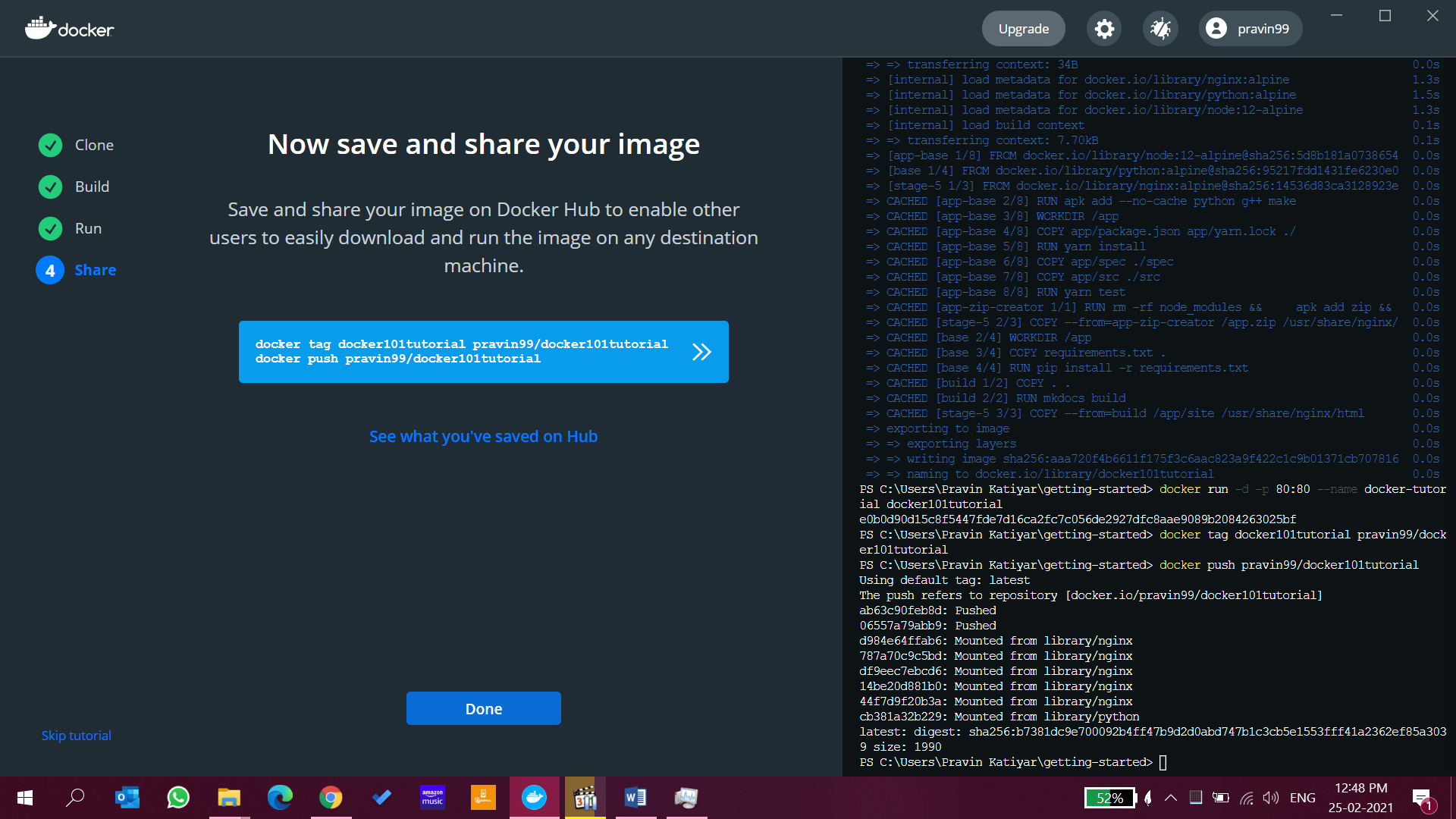
Download and install the file.

Now, click on “Docker Desktop” icon on desktop and follow instructions.









Congratulation, Installation is completed successfully

***DOCKER:***

**Docker editions. Which Do I use?**

Enterprise (Paid) and community edition (Free)

Community Editions=>

Edge Release (Not tested, Every Month)

Stable release (Well tested, Quarterly)

Docker for Windows (Set of Tool Box)

Docker natively does not support windows or mac

Docker support Linux natively

To install docker in windows we need a set of tools to install. These tools create linux virtual machine on your windows pc

And run docker on these virtual machine.

***Image vs Containers:***

***Why Container required?***

When we are trying to deploy an application, the application needs certain dependencies, environment and other binary to execute successfully.

But if you want to ship that application to customer or to other developer then we have to setup certain dependencies and environment again on another machine to run application properly.

With the help of docker we create an executable image of docker

**Image** is basically an executable format. An image is package of (dependencies. Program source code, dependencies and binaries).

And container is required to run that Image.We have wrapped up everything in an Image and container is carrying that image.

**Image is the application we want to run**

**Container is a running instance of an image**

Docker containers are the **lightweight** alternatives of the virtual machine.

You can have containers running of the same image.

**Docker Architecture**

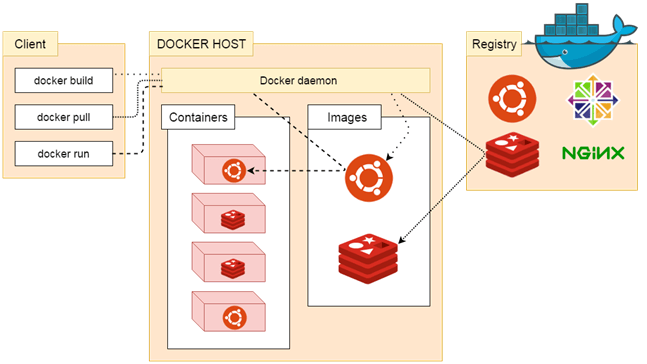
Before learning the Docker architecture, first, you should know about the Docker Daemon.

What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage. s

### Docker architecture

Docker follows Client-Server architecture, which includes the three main components that are **Docker Client**, **Docker Host**, and **Docker Registry**.



### 1. Docker Client

Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

#### **Note: Docker Client has an ability to communicate with more than one docker daemon.**

Docker Client uses Command Line Interface (CLI) to run the following commands -

docker build

docker pull

docker run

### 2. Docker Host

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

### 3. Docker Registry

Docker Registry manages and stores the Docker images.

There are two types of registries in the Docker -

**Pubic Registry -** Public Registry is also called as **Docker hub**.

**Private Registry -** It is used to share images within the enterprise.

## **Docker Objects**

There are the following Docker Objects -

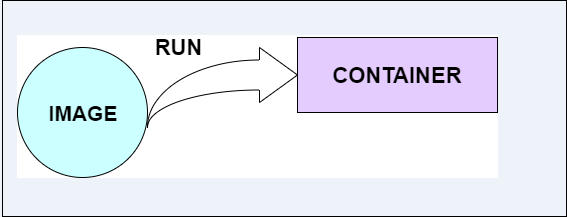
### Docker Images

Docker images are the **read-only binary templates** used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

### Docker Containers

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

In other words, we can say that the image is a template, and the container is a copy of that template.



### Docker Networking

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

* **Bridge -** Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
* **Host -** It is used when we don't need for network isolation between the container and the host.
* **None -** It disables all the networking.
* **Overlay -** Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
* **Macvlan -** Macvlan is used when we want to assign MAC addresses to the containers.

### Docker Storage

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

* **Data Volume -** Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
* **Directory Mounts -** It is one of the best options for docker storage. It mounts a host's directory into a container.
* **Storage Plugins -** It provides an ability to connect to external storage platforms.

# **Docker Dockerfile**

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Dockerfile.

The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

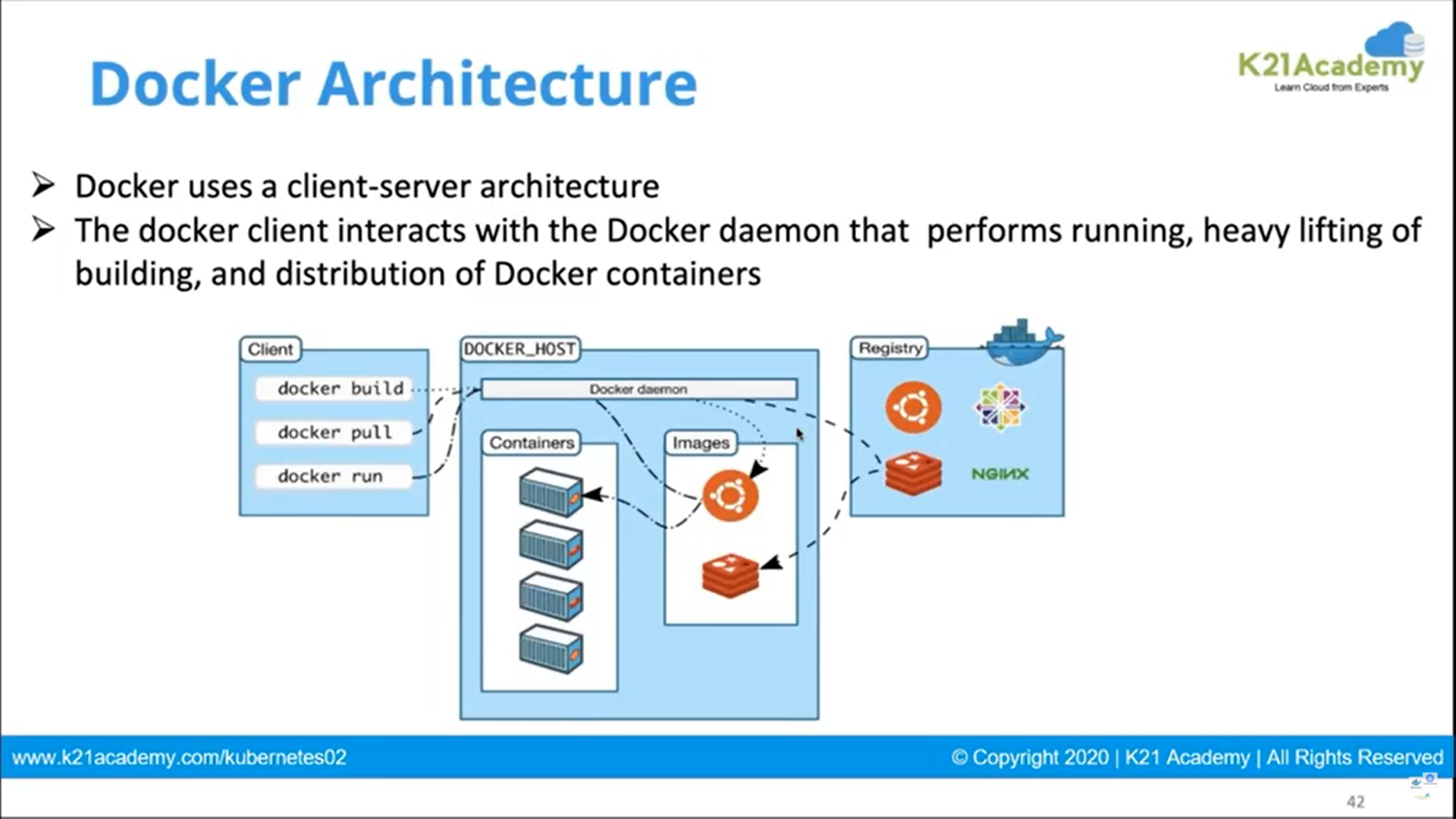
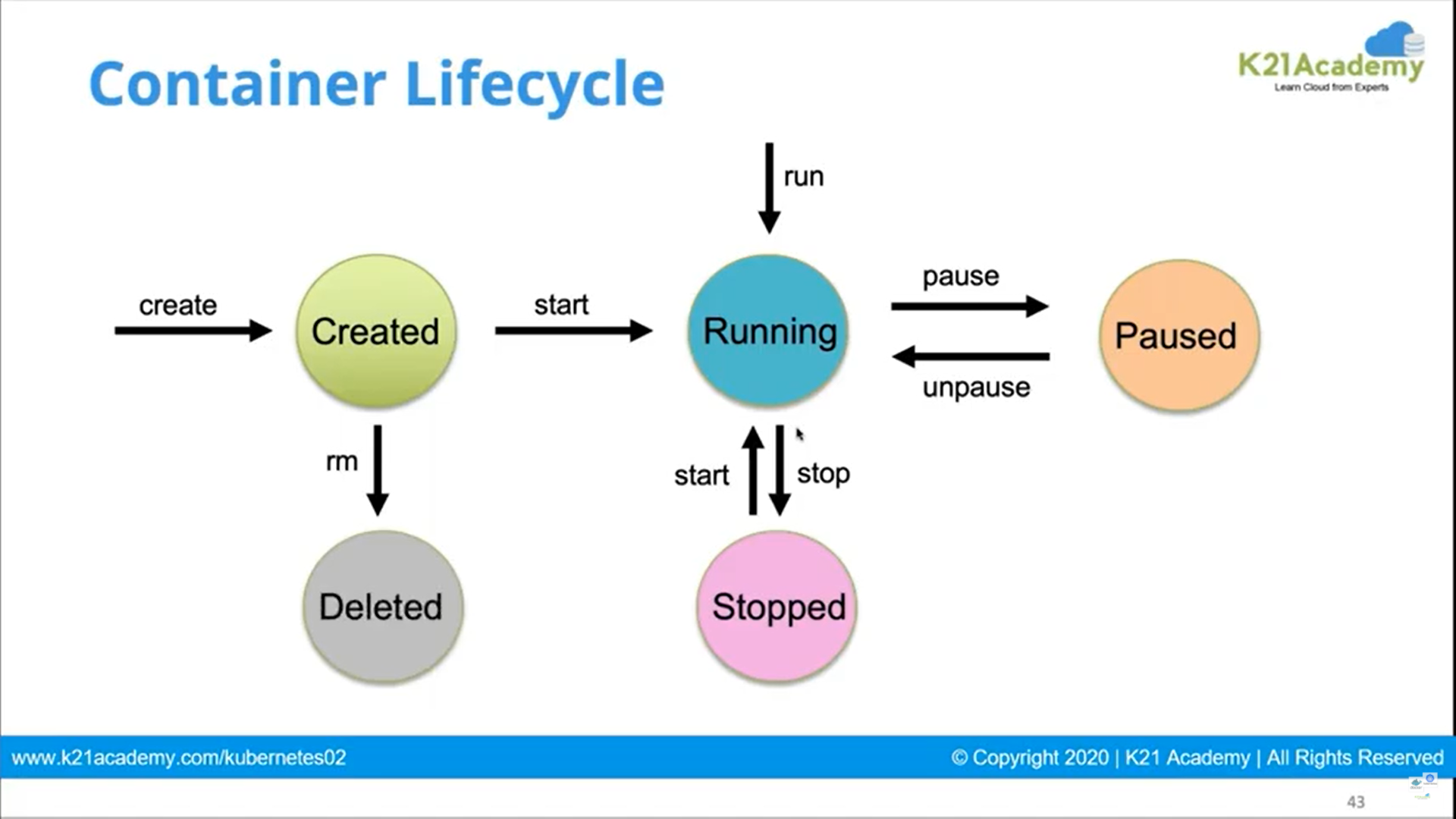
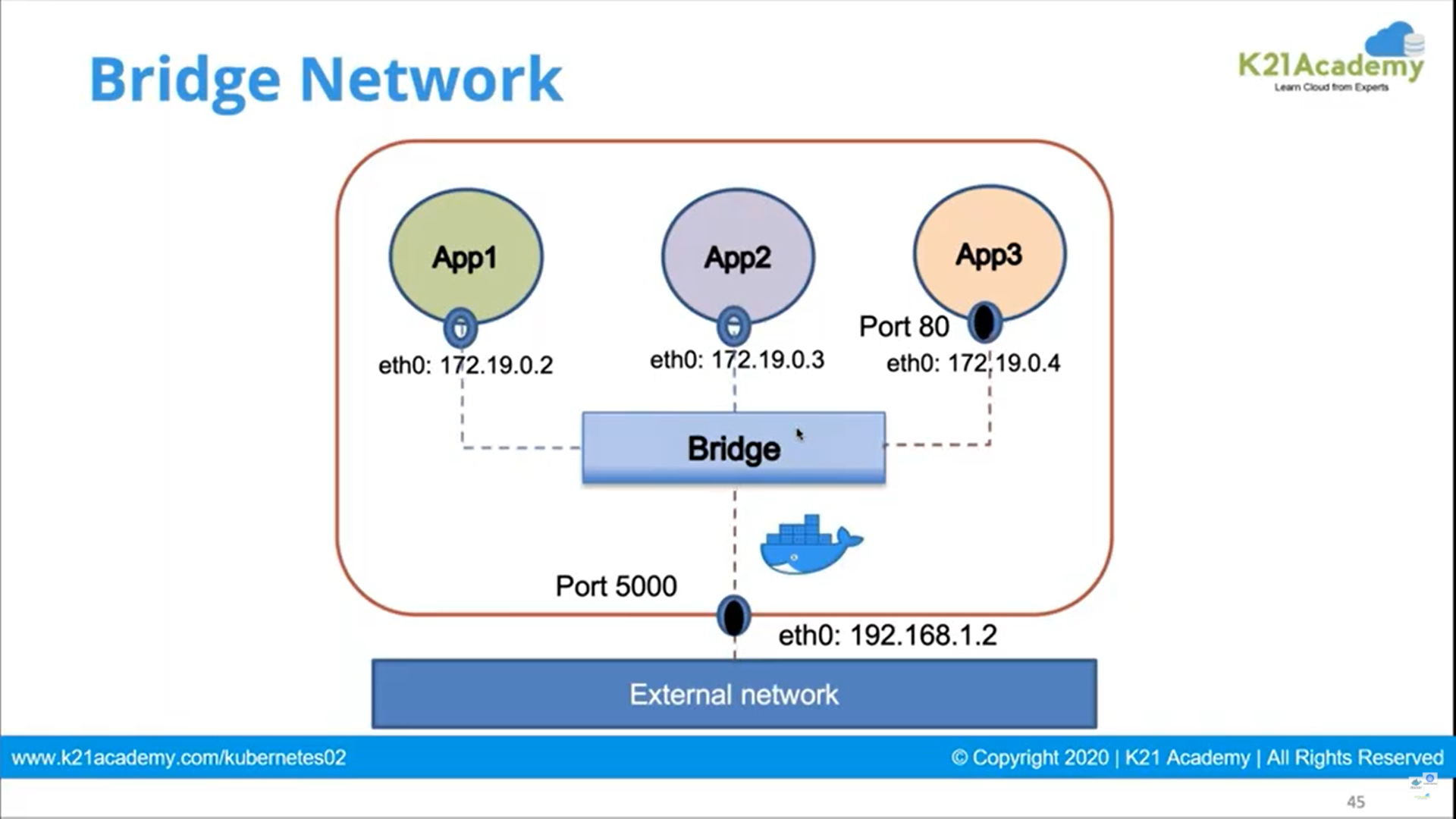
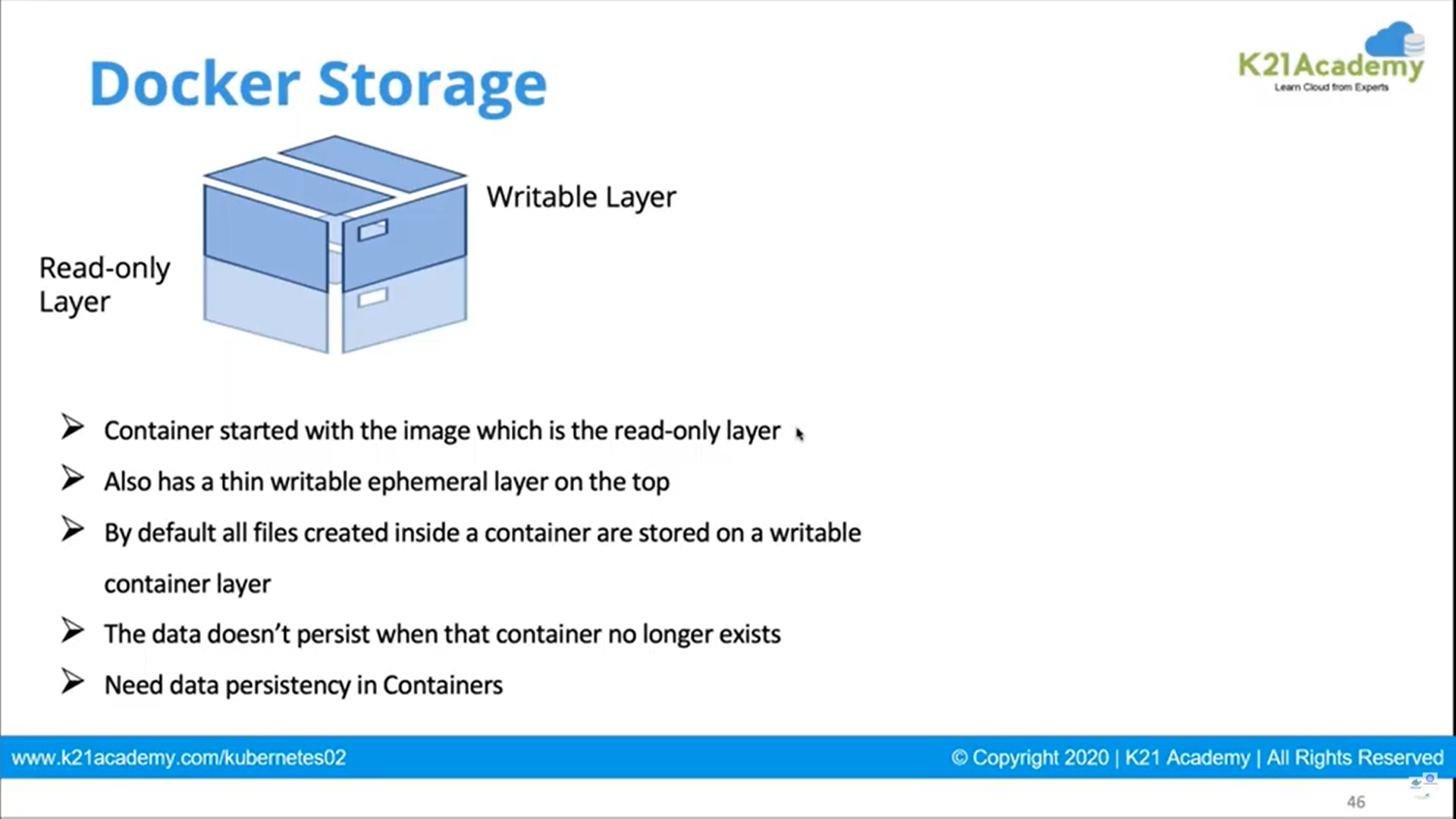
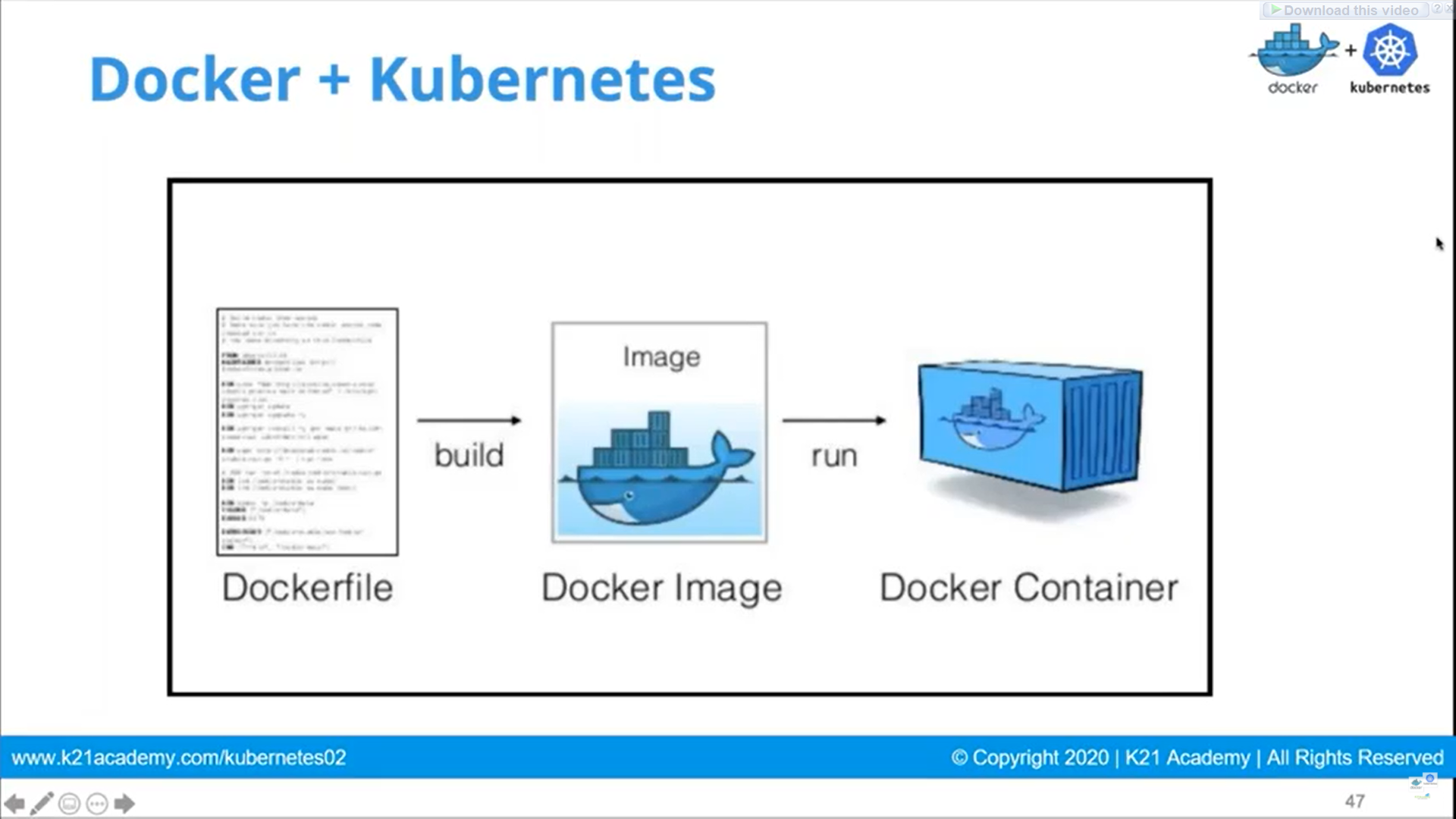
1. $ docker build -f /path/to/a/Dockerfile .

We will use Open source Nginx Web-Server

Containers vs. Virtual Machine

|  |  |
| --- | --- |
| **Containers** | **Virtual Machine** |
| Integration in a container is faster and cheap. | Integration in vm is slow and costly. |
| No wastage of memory. | Wastage of memory. |
| It uses the same kernel, but different distribution. | It uses multiple independent operating systems. |

There are the following advantages of Docker -

* It uses less memory.
* It provides lightweight virtualization.
* It does not require full operating system to run applications.
* It uses application dependencies to reduce the risk.
* Docker allows you to use a remote repository to share your container with others.
* It provides continuous deployment and testing environment
* 
* 
* 
* 
* 

# **Docker Features**

Although Docker provides lots of features, we are listing some major features which are given below.

* Easy and Faster Configuration
* Increase productivity
* Application Isolation
* Swarm
* Routing Mesh
* Services
* Security Management

### Easy and Faster Configuration

This is a key feature of docker that helps us to configure the system easily and faster.

We can deploy our code in less time and effort. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.

### Increase productivity

By easing technical configuration and rapid deployment of application. No doubt it has increase productivity. Docker not only helps to execute the application in isolated environment but also it has reduced the resources.

### Application Isolation

It provides containers that are used to run applications in isolation environment. Each container is independent to another and allows us to execute any kind of application.

### Swarm

It is a clustering and scheduling tool for Docker containers. Swarm uses the Docker API as its front end, which helps us to use various tools to control it. It also helps us to control a cluster of Docker hosts as a single virtual host. It's a self-organizing group of engines that is used to enable pluggable backends.

### Routing Mesh

It routes the incoming requests for published ports on available nodes to an active container. This feature enables the connection even if there is no task is running on the node.

### Services

Services is a list of tasks that lets us specify the state of the container inside a cluster. Each task represents one instance of a container that should be running and Swarm schedules them across nodes.

### Security Management

It allows us to save secrets into the swarm itself and then choose to give services access to certain secrets.

It includes some important commands to the engine like secret inspect, secret create etc.

***DOCKER HUB***

We will explore about Docker central Repository also known as docker hub

<https://hub.docker.com>

On docker hub we can push our images privately or publicly or we can pull or download any publicly available image

Docker run hello-world => to run hello-world container

Docker-info => to get information of the docker

Now we will try to start Nginx web server in docker

docker run --detach --publish 80:80 --name webserver nginx

80 is the hosting machine port number

docker ps=> to list all container (ps=>process status)

docker rm <container id>=> to remove container by container id

docker stop <container id> => to stop container by container name

docker inspect => To get full information of the container.

docker start <container id> =>to start container by container name

docker image ls => to view list of all images

docker container ls=>To view all containers

docker ps –a=>to get all process status

docker images =>To get All images present in docker

docker stop $(docker ps -aq) =>to stop all the docker containers at once

docker rm $(docker ps -aq)=> To remove all the docker containers at once

docker rm $(docker ps -a -q) –f =>This docker command will remove all containers, even if they are running.

docker pull openjdk:8 => pull Java 8 from docker hub to local machine

docker pull mysql => pull mysql from docker hub to local machine

***Docker Compose***

Docker compose

: tool for defining & running multi-container docker applications

: use yaml files to configure application services (docker-compose.yml)

: can start all services with a single command

: docker compose up

: can stop all services with a single command

: docker compose down

: can scale up selected services when required

Step 1 : install docker compose

(already installed on windows and mac with docker) docker-compose –v

Step 2 : Create docker compose file at any location on your system

docker-compose.yml

Step 3 : Check the validity of file by command

docker-compose config

Step 4 : Run docker-compose.yml file by command

docker-compose up -d

Steps 5 : Bring down application by command

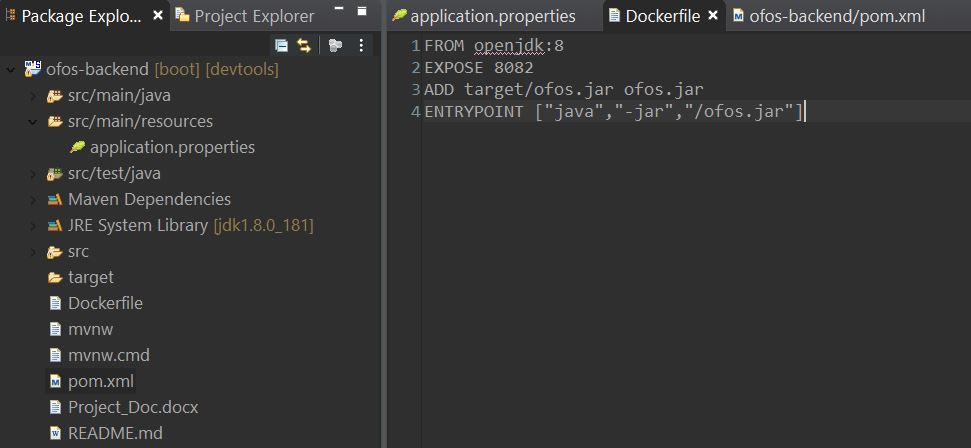
docker-compose down

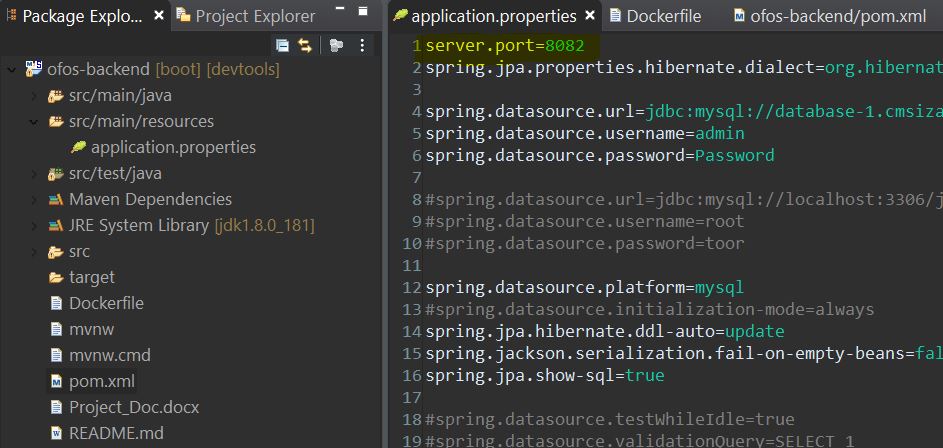
How to scale services

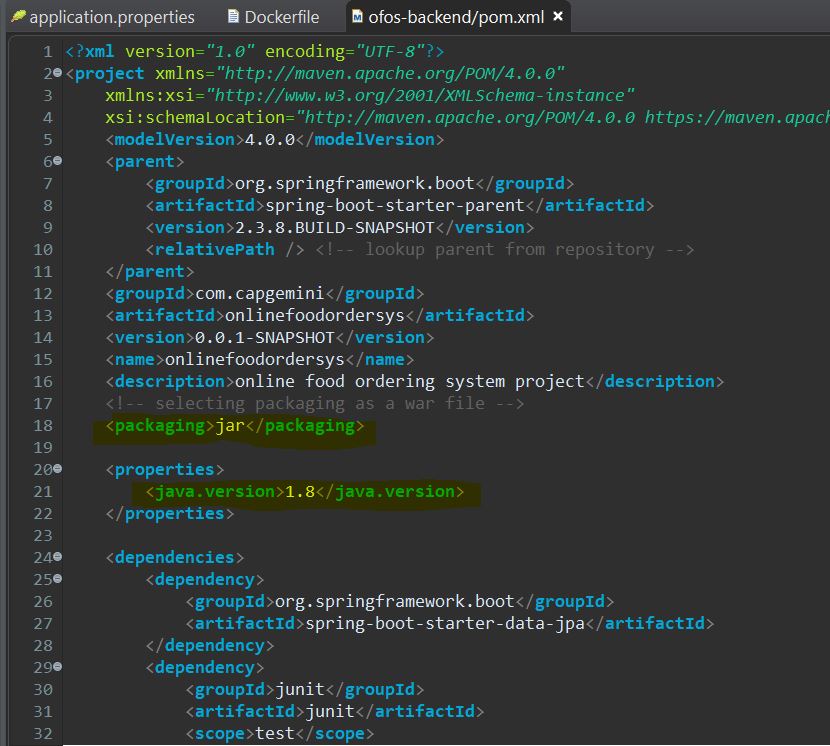
—scale docker-compose up -d --scale database=4

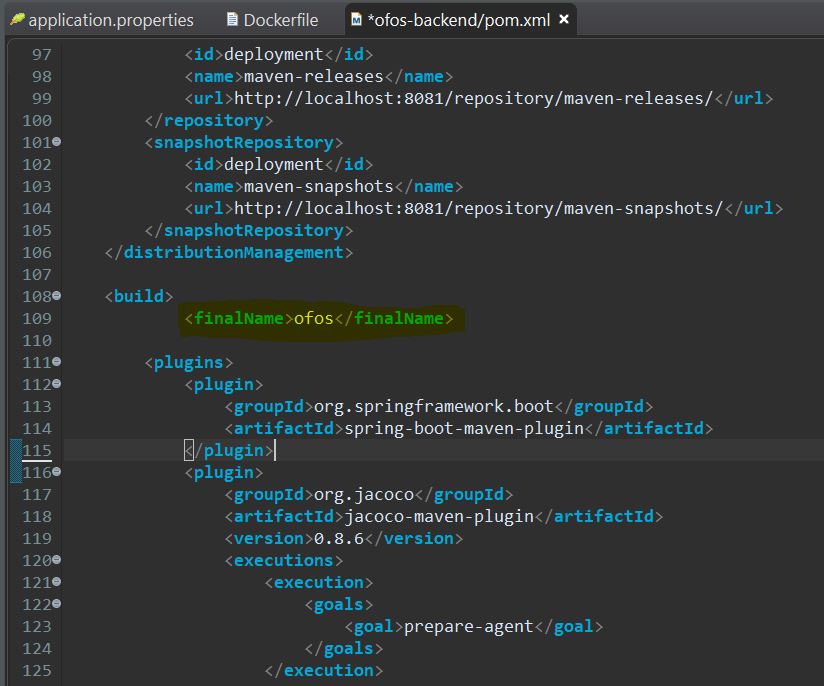
**Creating Java Docker Images**

Docker runs instructions of Dockerfile in top to bottom order. The first instruction must be **FROM** in order to specify the Base Image.









docker build –t ofos.jar:1.0 .

docker run –p 8082:8082 ofos.jar

***push image to docker Hub***

1: login into docker account

pravin99 is a repository name

W:\JavaWorkspace\JavaKubernetes>docker login

Authenticating with existing credentials...

Login Succeeded

2: Tag container image

W:\JavaWorkspace\JavaKubernetes>docker tag javakubernetes.jar:1.0 pravin99/javakubernetesapp

3: Push the image to hub

W:\JavaWorkspace\JavaKubernetes>docker images

W:\JavaWorkspace\JavaKubernetes>docker push pravin99/javakubernetesapp

Using default tag: latest

The push refers to repository [docker.io/pravin99/javakubernetes.jar]

81524c36447a: Pushed

d8460eb1eb77: Mounted from library/openjdk

30b0cc543ee7: Mounted from library/openjdk

26863b4714ee: Mounted from library/openjdk

5d5962699bd5: Mounted from library/openjdk

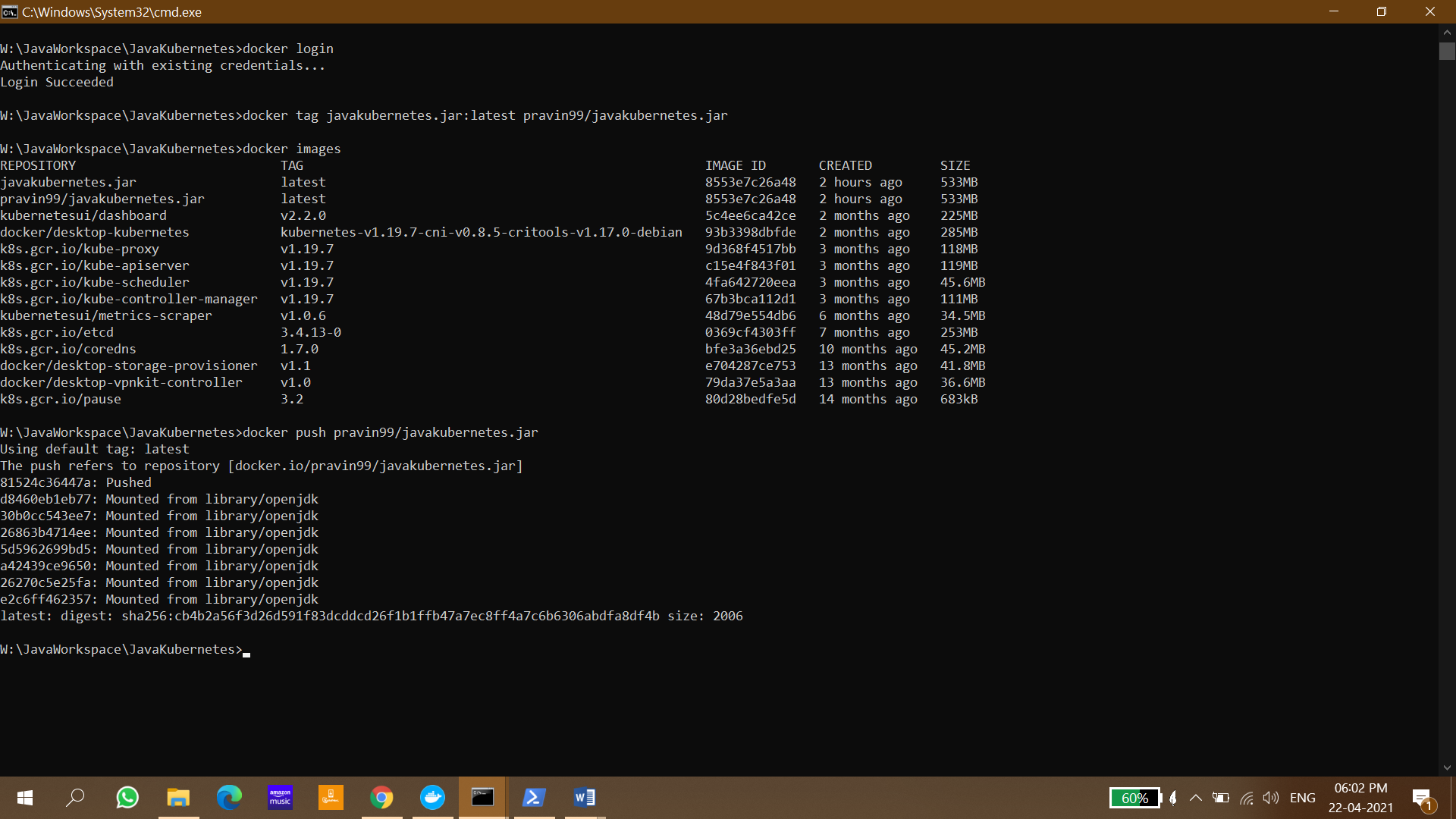
a42439ce9650: Mounted from library/openjdk

26270c5e25fa: Mounted from library/openjdk

e2c6ff462357: Mounted from library/openjdk

latest: digest: sha256:cb4b2a56f3d26d591f83dcddcd26f1b1ffb47a7ec8ff4a7c6b6306abdfa8df4b size: 2006

Image uploaded successfully



***DOCKER VS DOCKER-COMPOSE***

The docker cli is used when managing individual containers on a docker engine. It is the client command line to access the docker daemon api.

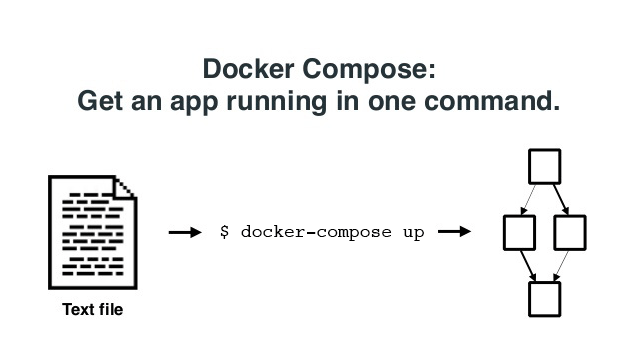
The docker-compose cli can be used to manage a multi-container application. It also moves many of the options you would enter on the docker run cli into the docker-compose.yml file for easier reuse. It works as a front end "script" on top of the same docker api used by docker, so you can do everything docker-compose does with docker commands and a lot of shell scripting.

**docker** manages single containers

**docker-compose** manages multiple container applications

Usage of *docker-compose* requires 3 steps:

1. Define the app environment with a **Dockerfile**
2. Define the app services in **docker-compose.yml**
3. Run **docker-compose up** to start and run app

[](https://blog.docker.com/2015/05/online-meetup-recap-docker-compose-1-2-0/)