# What is Docker?

Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container.

Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container.

You can run the docker image as a docker container in any machine where docker is installed without depending on the operating system.

# Advantages :

## Portability

Docker facilitates the developers in packaging their applications with all dependencies into a single lightweight containers

Reproducibility:

Through encapsulating the applications with their dependencies within a container it ensures in software setups remaining consistent across the development, testing and production environments.

## Efficiency :

Docker through its container based architecture it optimizes the resource utilization.

Scalability : facilitated the developers in making easier of their applications handling at time of workloads increment.

# Key Components of Docker

The following are the some of the key components of Docker:

## Docker Engine:

It is a core part of docker, that handles the creation and management of containers.

## Docker Image:

It is a read-only template that is used for creating containers, containing the application code and dependencies.

## Docker Hub:

It is a cloud based repository that is used for finding and sharing the container images.

## Dockerfile:

It is a script that containing instructions to build a docker image.

## Docker Registry :

It is a storage distribution system for docker images, where you can store the images in both public and private modes.

# What is a Dockerfile?

The Dockerfile uses DSL (Domain Specific Language) and contains instructions for generating a Docker image.

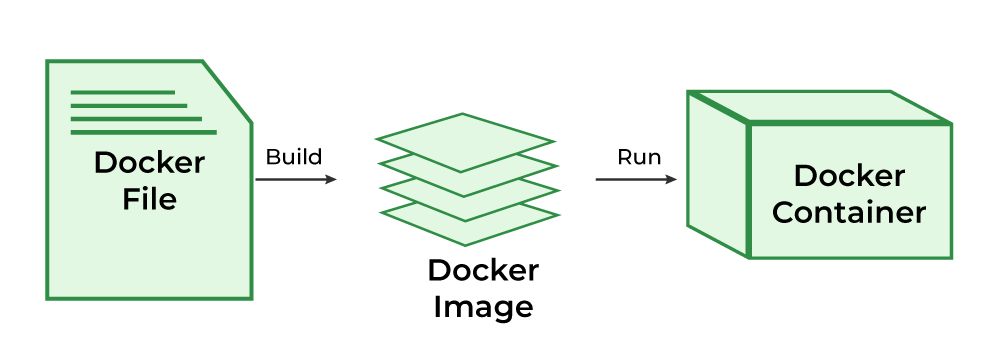
Dockerfile will define the processes to quickly produce an image.

While creating your application, you should create a Dockerfile in order since the Docker daemon runs all of the instructions from top to bottom.

# Docker Daemon

(The Docker daemon, often referred to simply as “Docker,” is a background service that manages Docker containers on a system.)

* It is a text document that contains necessary commands which on execution help assemble a Docker Image.
* Docker image is created using a Dockerfile.

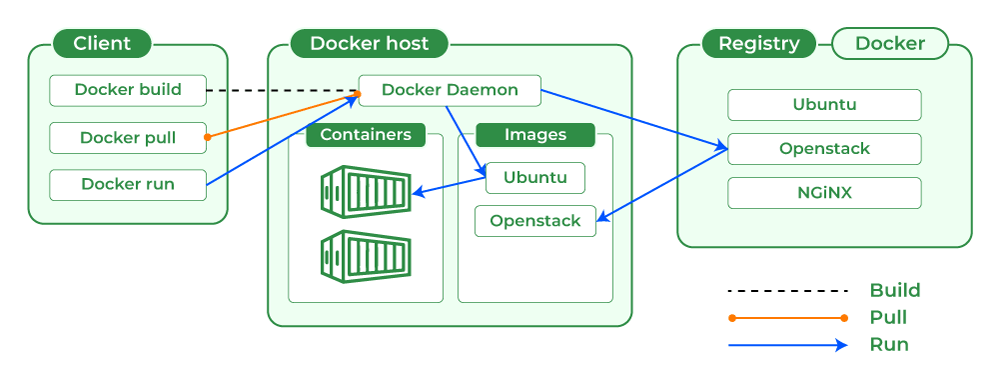


# What is Docker Architecture and How Docker Works?

Docker makes use of a client-server architecture. The Docker client talks with the docker daemon which helps in building, running, and distributing the docker containers.

The Docker client runs with the daemon on the same system or we can connect the Docker client with the Docker daemon remotely.

With the help of REST API over a UNIX socket or a network, the docker client and daemon interact with each other.



# What is Docker Image?

Docker Image is an executable package of software that includes everything needed to run an application.

This image informs how a container should instantiate, determining which software components will run and how.

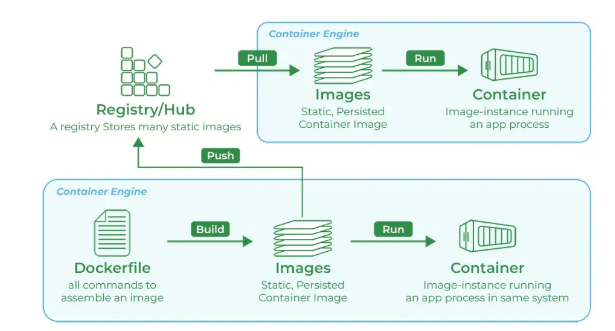
# What is Docker Container?

Docker container is a runtime instance of an image. Allows developers to package applications with all parts needed such as libraries and other dependencies.

# What is Docker Hub?

Docker Hub is a repository service and it is a cloud-based service where people push their Docker Container Images and also pull the Docker Container Images from the Docker Hub anytime or anywhere via the internet.

Generally it makes it easy to find and reuse images. It provides features such as you can push your images as private or public registry where you can store and share Docker images.



# What is Docker Compose?

* Docker Compose will execute a YAML-based multi-container application.
* The YAML file consists of all configurations needed to deploy containers Docker Compose , which is integrated with Docker Swarm , and provides directions for building and deploying containers.
* With Docker Compose, each container is constructed to run on a single host.

# Docker Swarm Mode :

[Link for Docker Swarm](https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode/)

# How to Download Docker Desktop?

Docker Desktop provides GUI to work on docker containers, docker images and docker networks.

Docker desktop provides and separate environment which contains Docker Engine, Docker CLI, Docker Compose, Kubernetes, and other tools which are needed to build, ship and run the applications in the form of containers which makes it more user friendly.

To know more how to install docker desktop refer to

[Docker Desktop Sample Image.](https://www.geeksforgeeks.org/docker-desktop-sample-image)

# Docker Commands

Through introducing the essential docker commands, docker became a powerful software in streamlining the container management process. It helps in ensuring a seamless development and deployment workflows. The following are the some of docker commands that are used commonly:

## Docker Run:

It used for launching the containers from images, with specifying the runtime options and commands.

## Docker Pull:

It fetches the container images from the container registry like Docker Hub to the local machine.

## Docker ps :

It helps in displaying the running containers along with their important information like container ID, image used and status.

## Docker Stop :

It helps in halting the running containers gracefully shutting down the processes within them.

## Docker Start:

It helps in restarting the stopped containers, resuming their operations from the previous state.

## Docker Login:

It helps to login in to the docker registry enabling the access to private repositories.

# Docker Commands

**Last Updated :**04 Jan, 2025

Docker is an open-source project that automates the deployment of applications as movable, independent containers that can run locally or in the [cloud.](https://www.geeksforgeeks.org/cloud-computing/) You can divide your applications from your infrastructure with the help of Docker, allowing for quick software delivery and it also allows you to manage your infrastructure in the same ways that you manage your applications.

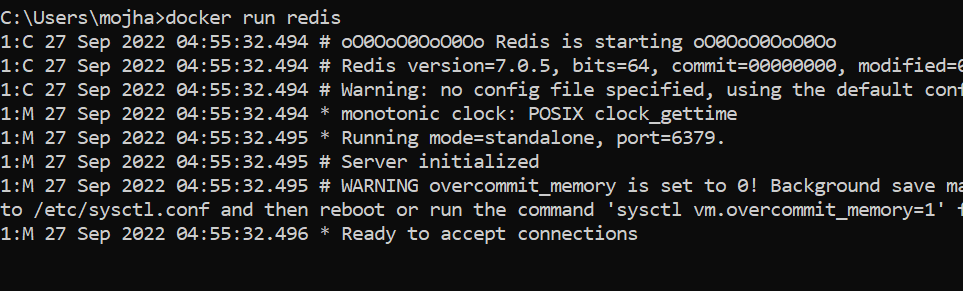
 The number of commands found in docker is very huge in number, but we will be looking at the top commands in docker. To know more about Docker commands refer to [Docker Cheat Sheet – Most Important Docker Commands.](https://www.geeksforgeeks.org/docker-cheat-sheet/)

## **Docker Commands**

### **Docker Run command**

This command is used to run a container from an image. The docker run command is a combination of the docker create and docker start commands. It creates a new container from the image specified and starts that container. if the [docker image](https://www.geeksforgeeks.org/what-is-docker-images/) is not present, then the docker run pulls that.

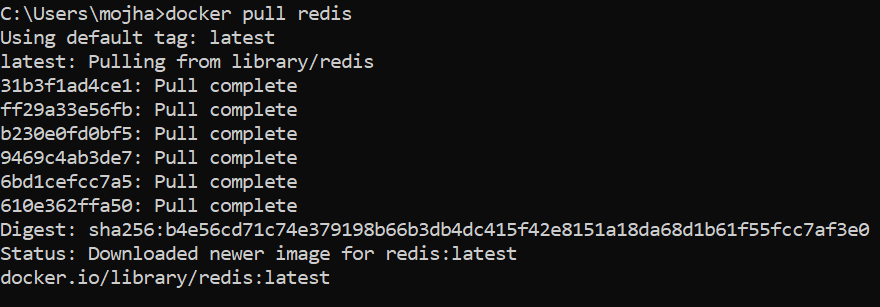
$ docker run <image\_name>  
To give name of container  
$ docker run --name <container\_name> <image\_name>



### **Docker Pull**

This command allows you to pull any image which is present in the official [registry of docker](https://www.geeksforgeeks.org/what-is-docker-registry/), [Docker hub](https://www.geeksforgeeks.org/what-is-docker-hub/). By default, it pulls the latest image, but you can also mention the version of the image.

$ docker pull <image\_name>

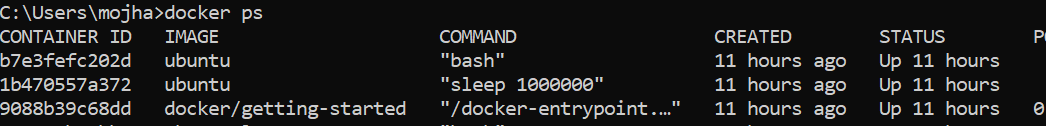


### **Docker PS**

This command (by default) shows us a list of all the running containers. We can use various flags with it.

* **-a flag:** shows us all the containers, stopped or running.
* **-l flag:**shows us the latest container.
* **-q flag**: shows only the Id of the containers.

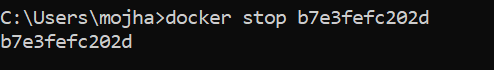
$ docker ps [options..]



### **Docker Stop**

This command allows you to stop a container if it has crashed or you want to switch to another one.

$ docker stop <container\_ID>



### **Docker Start**

Suppose you want to start the stopped container again, you can do it with the help of this command.

$ docker start <container\_ID>

### **Docker rm**

To delete a container. By default when a container is created, it gets an ID as well as an imaginary name such as confident\_boyd, heuristic\_villani, etc. You can either mention the container name or its ID.

Some important flags:

* **-f flag:**remove the container forcefully.
* **-v flag:**remove the volumes.
* **-l flag:**remove the specific link mentioned.

$ docker rm {options} <container\_name or ID>

docker remove an image

### Docker RMI

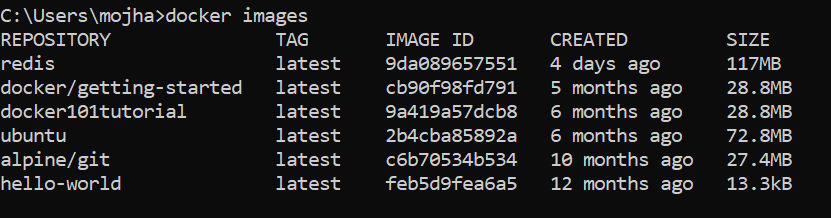
To delete the image in docker. You can delete the images which are useless from the docker local storage so you can free up the space

docker rmi <image ID/ image name>

### **Docker Images**

Lists all the pulled images which are present in our system.

$ docker images



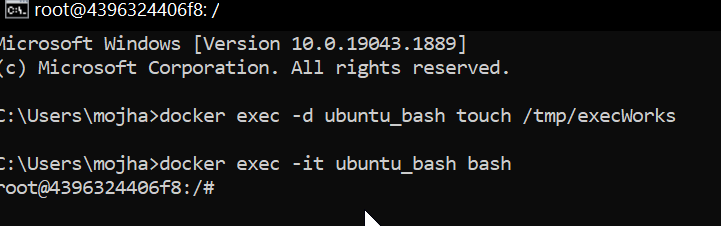
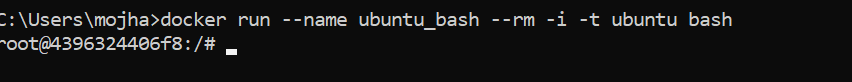
### **Docker exec**

This command allows us to run new commands in a running container. This command only works until the container is running, after the container restarts, this command does not restart.

Some important flags:

* **-d flag:**for running the commands in the background.
* **-i flag:**it will keep STDIN open even when not attached.
* **-e flag:**sets the environment variables

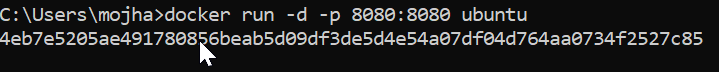
$ docker exec {options}



### **Docker Ports (Port Mapping)**

In order to access the [docker container](https://www.geeksforgeeks.org/containerization-using-docker/) from the outside world, we have to map the port on our host( Our laptop for example), to the port on the container. This is where port mapping comes into play.

$ docker run -d -p <port\_on\_host>   
<port\_on\_container> Container\_name



So these were the 9 most basic docker commands that every beginner must know. Containerization is a very vast topic but you can start from the very basic commands and by practicing them daily you can master them.

### Docker Login

The Docker login command will help you to authenticate with the Docker hub by which you can push and pull your images.

docker login

It will ask you to enter the username and password after that you will authenticate with DockerHub and you can perform the tasks.

### Docker Push

Once you build your own customized image by using Dockerfile you need to store the image in the remote registry which is DockerHub for that you need to push your image by using the following command.[To know more about How to Push a Container Image to a Docker Repository?](https://www.geeksforgeeks.org/how-to-push-a-container-image-to-a-docker-repository/)

docker push <Image name/Image ID>

### Docker Build

The docker build command is used to build the docker images with the help of [Dockerfile.](https://www.geeksforgeeks.org/docker-concept-of-dockerfile/)

docker build -t image\_name:tag .

In the place of **image\_name** use the name of the image you build with and give the **tag number** and **. “dot”**represents the current directory.

### Docker Stop

You can stop and start the docker containers where you can do the maintenance for containers. To stop and start specific containers you can use the following commands.

docker stop container\_name\_or\_id

### Stop Multiple Containers

Instead of stopping a single container. You can stop multiple containers at a time by using the following commands.

docker stop container1 container2 container3

### Docker Restart

While running the containers in Docker you may face some errors and containers fails to start. You can restart the containers to resolve the containers by using the following commands.

docker restart container\_name\_or\_id

### Docker Inspection

Docker containers will run into some errors in real time to debug the container’s errors you can use the following commands.

docker inspect container\_name\_or\_id

### Docker Commit command

After running the containers by using the current image you can make the updates to the containers by interacting with the containers from that containers you can create an image by using the following commands.

docker commit container\_name\_or\_id new\_image\_name:tag

## Docker Basic Command

Following are the some of the docker basic commands

1. **docker images:** Docker images will list all the images which are pulled or build in that docker host.
2. **docker pull:**Docker pull will the docker images from the dockerhub.
3. **docker run:**Docker run will run the docker image as an container.
4. **docker ps:**Docker run will list all the containers which are running in the docker host.
5. **docker stop:**Docker stop will stop the docker container which are already running.
6. **docker rm:**Docker rm command will remove the containers which are in the stop condition.

## Docker Commands List

Following are the docker commands which listed form build and Docker image to running it an Docker container and attaching the docker volumes to it.

### Docker Image Command

1. **docker build command:** It will build Docker images by using the **Dockerfile.**
2. **docker pull command:** Docker pull command will pull the **Docker image** whcih is avalible in the **dockerhub.**
3. **docker images command:**It will list all the images which are pulled and build in the docker host.
4. **docker inspect command:**It will helps to debug the docker image if any errors occurred while building an image or pulling the image.
5. **docker push command:**Docker command will push the docker image into the Dockerhub.
6. **docker save command:**It will save the docker image in the form of dockerfile.
7. **docker rmi command:**It will remove the docker image.

### Docker Container Command

1. **docker attach command:** Connecting to an Existing Container.
2. **docker ps command:**To list the running containers.
3. **docker container inspect infinite Command:**To Inspect the Docker containers.
4. **docker exec command:**To execute the commands in the running containers.
5. **docker cp command:** To copy the file from docker host to the docker containers,

# Docker Engine

The software that hosts the containers is named Docker Engine. Docker Engine is a client-server based application. The docker engine has 3 main components:

## Server:

It is responsible for creating and managing Docker images, containers, networks, and volumes on the Docker. It is referred to as a daemon process.

## REST API :

It specifies how the applications can interact with the Server and instructs it what to do.

## Client:

The Client is a docker command-line interface (CLI), that allows us to interact with Docker using the docker commands.

# Why to use Docker?

Docker can be used to pack the application and its dependencies which makes it lightweight and easy to ship the code faster with more reliability. Docker make its very simple to run the application in the production environment docker container can be platform independent if the docker engine is installed in the machine.

## Resource Efficiency :

Docker helps in maximizing the resource utilization by running the multiple containers on a single host. It helps in reducing the infrastructure costs and improves the efficiency.

## Version Control:

It simples the versioning for the applications and their dependencies ensuring the consistency and making easier of collaboration across the teams.

## Microservices Agility:

It enables the adoption of microservices architecture, promoting the scalability, flexibility and fault isolation agile application development.

# Difference Between Docker Containers and Virtual Machines

The following are the differences between docker containers and Virtual Machines:

|  |  |  |
| --- | --- | --- |
|  | Docker Containers | Virtual Machines |
| 1 | Docker Containers contain binaries, libraries, and configuration files along with the application itself. | Virtual Machines (VMs) run on Hypervisors, which allow multiple Virtual Machines to run on a single machine along with its own operating system. |
|  |  |  |
| 2 | They don’t contain a guest OS for each container and rely on the underlying OS kernel, which makes the containers lightweight. | Each VM has its own copy of an operating system along with the application and necessary binaries, which makes it significantly larger and it requires more resources. |
|  |  |  |
| 3. | Containers share resources with other containers in the same host OS and provide OS-level process isolation. | They provide Hardware-level process isolation and are slow to boot. |
|  |  |  |
|  |  |  |

# Sample Example: Containerizing Application Using Docker

Steps : create two source file

1. HelloWorld.java , Also compile this java program
2. Create Dockerfile file as below snip.

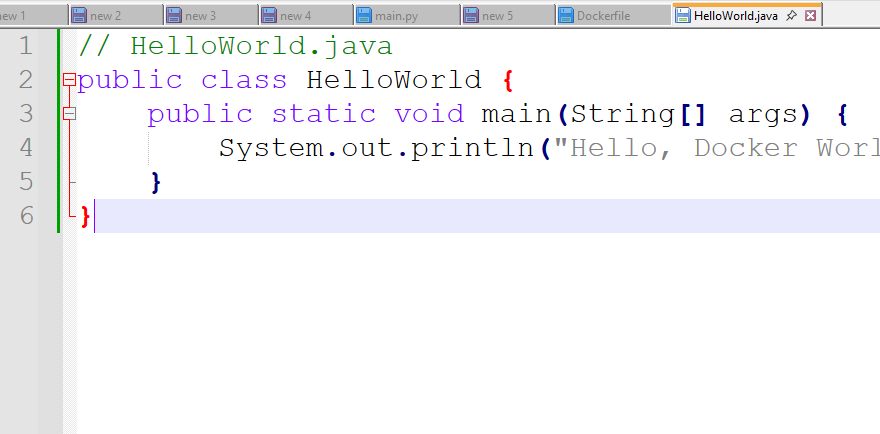
Note please : keep both the files in one folder.

## 1. Writing the Java Application

Let's start with a basic Java program.

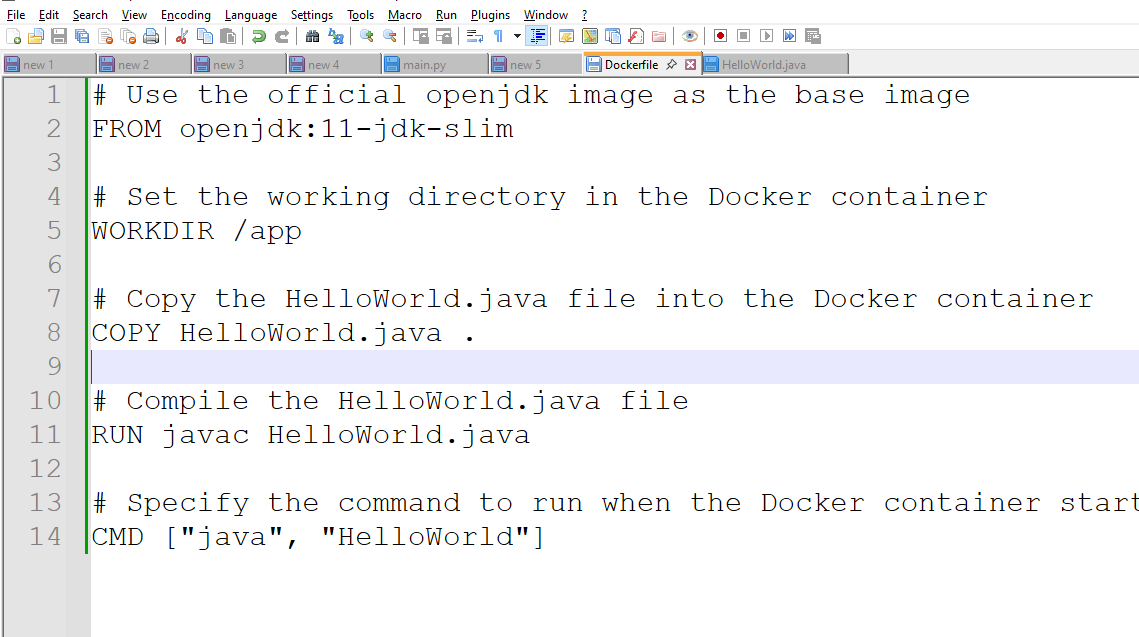
## 2. Compiling the Java Application

You can check whether your program compiles or not by using the following command:



## 3. Creating a Dockerfile

The *Dockerfile* is a script used by Docker to create a Docker image. Here's a basic *Dockerfile* to run our Java application:

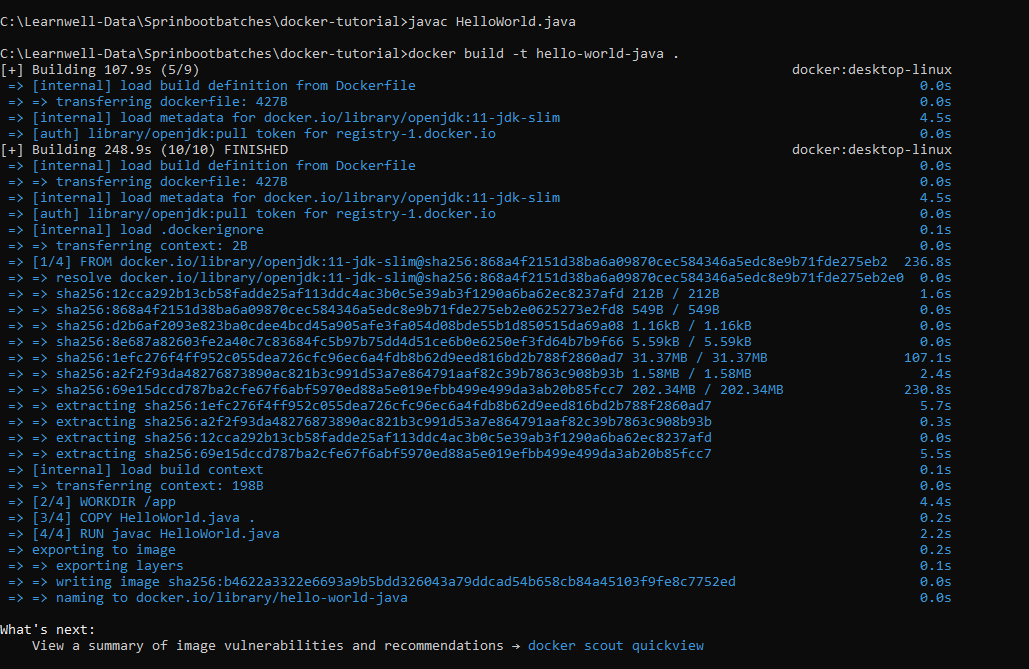


Please run this command on project source root directory

## 4. Building the Docker Image

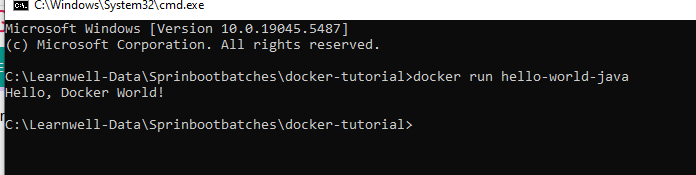
Navigate to the directory containing your Java program and *Dockerfile*. Then, build the Docker image using the following command:





## 5. Running the Docker Container

Once the image is built, you can run your Java application inside a Docker container:



References :  
https://www.javaguides.net/2023/08/dockerize-hello-world-java-program.html#:~:text=Building%20the%20Docker%20Image,in%20the%20current%20directory%20(.)