

## Q1) Pull any image from the docker hub, create its container, and execute it showing the output.

**Docker Hub:** Docker Hub is a hosted repository service provided by Docker for finding and sharing container images with your team.

->Key features include: Private Repositories, Push and pull container images.

->It automatically build container images from GitHub and Bitbucket and push them to Docker Hub.

->**docker pull <img name>**

--Just want to download the img not run the img in the container then use pull command to pull that img.

--after that if we want to run or execute that img then we can use “docker run resin “ command.

->**docker create <img name>**

--To create a container

->**docker start -a <container id>**

--runs the container and shows output. This command tells docker to run . After executing this command we can see output of the image we executed in the container.

```
Command Prompt
C:\Users\krish>docker ps -all
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
C:\Users\krish>docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:6e8b6f026e0b9c419ea0fd02d3905dd0952ad1feea67543f525c73a0a790fefb
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
C:\Users\krish>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
java-app      latest   514e757bab13   29 hours ago   526MB
<none>        <none>    2952781dce84   30 hours ago   526MB
ubuntu        latest   58db3edaf2be   3 weeks ago    77.8MB
resin/docs    latest   592de848a9b7   4 months ago   1.1GB
hello-world   latest   feb5d9fea6a5   17 months ago  13.3kB
C:\Users\krish>docker create hello-world
938406eb19911afd88e085bdca8c7373aa8237301a4f5786d9e12ac373102afd
C:\Users\krish>docker start -a 938406eb19911afd88e085bdca8c7373aa8237301a4f5786d9e12ac373102afd

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the   executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

C:\Users\krish>
```

(Or)

->docker run <img name>

--it checks whether our system has hello-world image or not if its present then it runs that ,or else it downloads from docker hub and then runs it..

\*\*\* docker run does the work of docker pull, docker create and docker start.

```
C:\Users\krish>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:6e8b6f026e0b9c419ea0fd02d3905dd0952ad1fee67543f525c73a0a790fefb
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

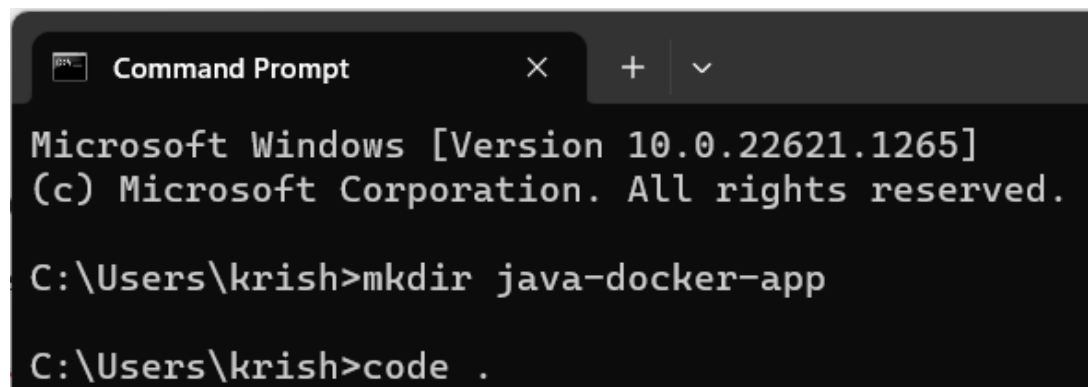
C:\Users\krish>
```

**Q2) Create the basic java application, generate its image with necessary files, and execute it with docker.**

**Steps to run a java application in docker:**

**1.Create a directory (in cmd)**

->mkdir java-docker-app



```
Microsoft Windows [Version 10.0.22621.1265]
(c) Microsoft Corporation. All rights reserved.

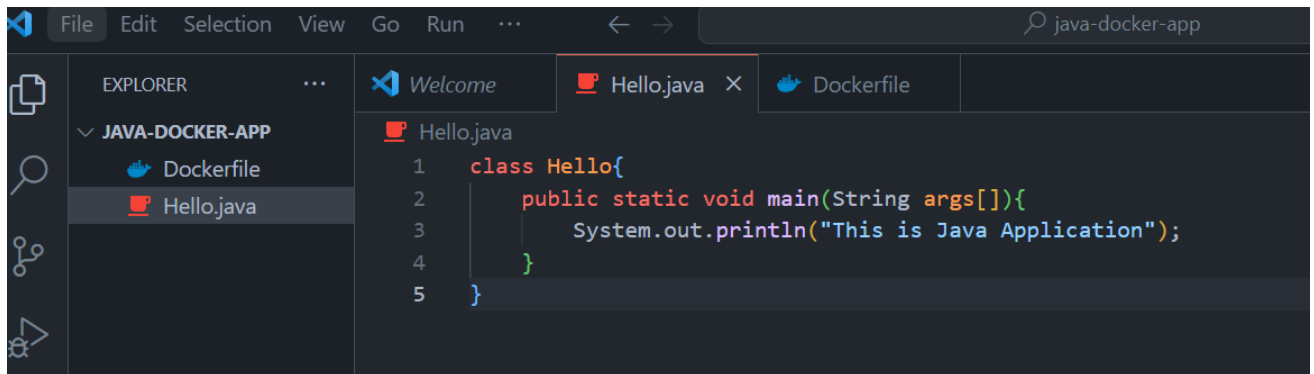
C:\Users\krish>mkdir java-docker-app

C:\Users\krish>code .
```

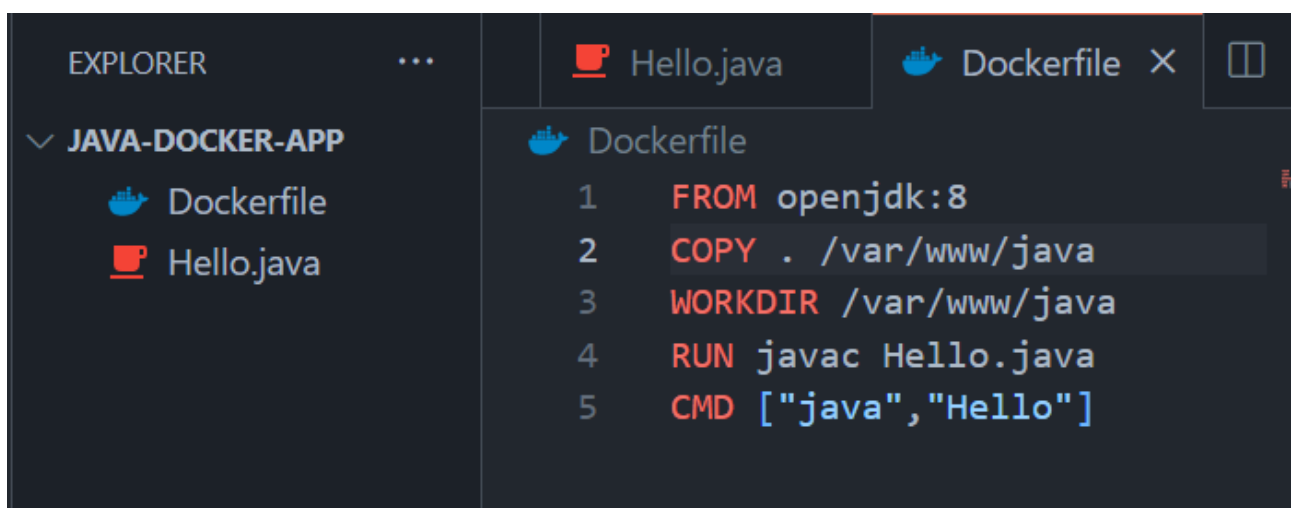
Open vscode to write our java application inside the created directory. ->code .

Open two files 1).java file 2)docker file

## 2.Create a file Hello.java



## 3.Create another file Dockerfile



## 4. Come back to cmd and give command and build docker image.

->cd java-docker-app

```
C:\Users\krish>cd java-docker-app
```

->docker build -t java-app .

```
Command Prompt
C:\Users\krish\java-docker-app>docker build -t java-app .
[+] Building 67.9s (10/10) FINISHED
=> [internal] load build definition 0.0s
=> => transferring dockerfile: 141B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for dock 7.8s
=> [auth] library/openjdk:pull token 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 304B 0.0s
=> [1/4] FROM docker.io/library/ope 58.4s
=> => resolve docker.io/library/open 0.0s
=> => sha256:001c 55.00MB / 55.00MB 34.0s
=> => sha256:d9d4b9 5.16MB / 5.16MB 11.3s
=> => sha256:2068 10.88MB / 10.88MB 34.8s
=> => sha256:3af2ac9 1.79kB / 1.79kB 0.0s
=> => sha256:b273004 7.81kB / 7.81kB 0.0s
=> => sha256:86e863c 1.04kB / 1.04kB 0.0s
=> => sha256:9dae 54.58MB / 54.58MB 41.0s
=> => sha256:d85151 5.42MB / 5.42MB 41.4s
=> => extracting sha256:001c52e26ad5 1.7s
=> => sha256:52a8c426d3 210B / 210B 36.3s
=> => extracting sha256:d9d4b9b6e964 0.2s
=> => extracting sha256:2068746827ec 0.2s
=> => sha256:87 105.92MB / 105.92MB 56.3s
=> => extracting sha256:9daef329d350 1.9s
=> => extracting sha256:d85151f15b66 0.2s
=> => extracting sha256:52a8c426d30b 0.0s
=> => extracting sha256:8754a66e0050 1.7s
=> [2/4] COPY . /var/www/java 0.5s
=> [3/4] WORKDIR /var/www/java 0.0s
=> [4/4] RUN javac Hello.java 1.0s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:2952781dc 0.0s
=> => naming to docker.io/library/ja 0.0s

Use 'docker scan' to run Snyk tests against images to find vul
```

## 5.Run the docker image.

->docker run java-app

```
=> => extracting sha256:8754a66e0050 1.7s
=> [2/4] COPY . /var/www/java 0.5s
=> [3/4] WORKDIR /var/www/java 0.0s
=> [4/4] RUN javac Hello.java 1.0s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:2952781dc 0.0s
=> => naming to docker.io/library/ja 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\krish\java-docker-app>docker run java-app
This is Java Application

C:\Users\krish\java-docker-app>|
```

\_\_\*\*\*\_\_

