DOCKER Steps to run Java Program

Docker has its own terminology, which I'll be using throughout this introduction. Take a minute to familiarize yourself with these terms:

- Docker engine: The process running on your production machine that sits between your running Docker application and the underlying operating system and hardware.
- Dockerfile: A text file that contains the instructions (or commands) used to build a Docker image.
- Docker image: The result of building a Dockerfile and executing the Dockerfile's commands. It is constructed from a root operating system, installed applications, and commands executed in such a way that it can run your application. A Docker image serves as the basis for Docker containers and is the static template from which they are created.
- Docker container: A runtime instance of a Docker image. Whereas the Docker image is like a template (built from a Dockerfile that contains the correct operating system, applications, and commands used to build the image), the container is an actual running instance of that image.
- Docker host: A physical or virtual machine that runs the Docker engine to host your Docker container's DockerHub.
- DockerHub: The repository that hosts Docker images. Think of a DockerHub being to Docker images what GitHub is to Git repositories: a central location for hosting and serving up Docker images.

1)Create a directory

```
$mkdir Dockertest
```

2)Create a Java File

Now create a Java file. Save this file as **Hello.java** file.

// Hello.java

```
class Hello{
public static void main(String[] args){
System.out.println("This is my Docker");
}
}
```

3) **Create a Dockerfile**

After creating a Java file, we need to create a Dockerfile which contains instructions for the Docker. Dockerfile does not contain any file extension. So, save it simple with **Dockerfile** name.

// Dockerfile

```
FROM java:8
COPY . /vars/www/java
WORKDIR /vars/www/java
RUN javac Hello.java
CMD ["java", "Hello"]
```

Build the application.

```
$ docker build -t Dockertest .
```

1. **Run Docker Image**

   After creating image successfully. Now we can run docker by using run command. The following command is used to run java-app.

```
$ docker run Dockertest
```

```
$git clone https://github.com/spring-guides/gs-spring-boot-docker.git
```

```
$ docker run -p 8080:8080 -t gregturn/gs-spring-boot-docker
```

Docker container Stop:

```
docker container stop [OPTIONS] CONTAINER [CONTAINER...]
```

Purging All Unused or Dangling Images, Containers, Volumes, and Networks

Docker provides a single command that will clean up any resources — images, containers, volumes, and networks — that are dangling (not associated with a container):

docker system prune

To additionally remove any stopped containers and all unused images (not just dangling images), add the -a flag to the command:

docker system prune -a

Remove Containers:

```
docker rm ID_or_Name ID_or_Name
```

Stop and remove all containers

You can review the containers on your system with docker ps. Adding the -a flag will show all containers. When you're sure you want to delete them, you can add the -q flag to supply the IDs to the docker stop and docker rm commands:

List:

docker ps -a

Remove:

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)