### 

[Pre-requisites](#_vbfryz7x6cmg)

[Installing Docker](#_jia3mb9i2efv)

[What is Docker?](#_i6mbowxfm0ja)

[What are some Docker commands?](#_erbb97rn298h)

[How do I start using Docker?](#_9zc9cwbolsx9)

[Samples:](#_db3hamubt2or)

[Exploring docker run](#_65fql230rrlg)

[Exploring Dockerfile](#_olrybbibr02c)

[Exploring Docker Compose](#_hhizggmpp6v9)

### 

### Pre-requisites

We will need you to install 2 softwares:

1. Code Editor - We’d recommend Visual Studio Code - <https://code.visualstudio.com/download>
2. Docker - Follow the steps below

### Installing Docker

Browse the URL - <https://www.docker.com/get-started/> - and download the Docker Desktop on your local machine.

Sign up for Docker Hub account as well to browse through the different Docker images available by other users/companies.

### What is Docker?

Docker is a platform that enables developers to package, distribute, and run applications in a containerized environment. Containers are lightweight and isolated environments that package an application along with its dependencies and configuration, ensuring that the application runs consistently regardless of the environment it is deployed in.

Docker provides a way to create, manage, and run containers using a set of tools and APIs. With Docker, developers can build an application in one environment and deploy it in any other environment, without worrying about dependencies or compatibility issues.

Users can write their application in any programming language, and on top of that, they need to provide a set of instructions on how to compile their code, binaries and libraries into one lightweight container as well as how to start the container using that image.

Docker uses a client-server architecture, where the Docker client communicates with the Docker daemon to build, run, and manage containers. The Docker daemon runs on the host machine and manages the container lifecycle, including creating, starting, stopping, and deleting containers.

Docker has become very popular among developers and DevOps teams, as it simplifies the deployment and management of applications across different environments and infrastructure. It is widely used in modern application development and deployment workflows, such as microservices, continuous integration, and continuous deployment.

Look at this link to see what kind of online resources you can build with Docker:

<https://hub.docker.com/>

### What are some Docker commands?

Some popular commands that you should be familiar with include:

docker run - This command creates and starts a new container from a Docker image.

docker ps - This command lists all running containers.

docker images - This command lists all the images that are stored locally on your system.

docker stop - This command stops a running container.

docker rm - This command removes one or more containers.

docker rmi - This command removes one or more images.

docker build - This command builds a Docker image from a Dockerfile.

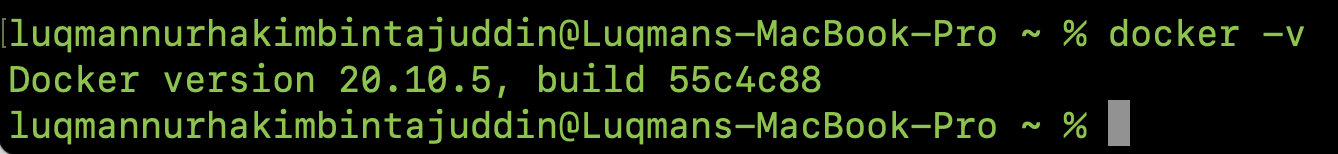
docker exec - This command runs a command inside a running container.

### How do I start using Docker?

<https://www.docker.com/blog/getting-started-with-docker-desktop/>

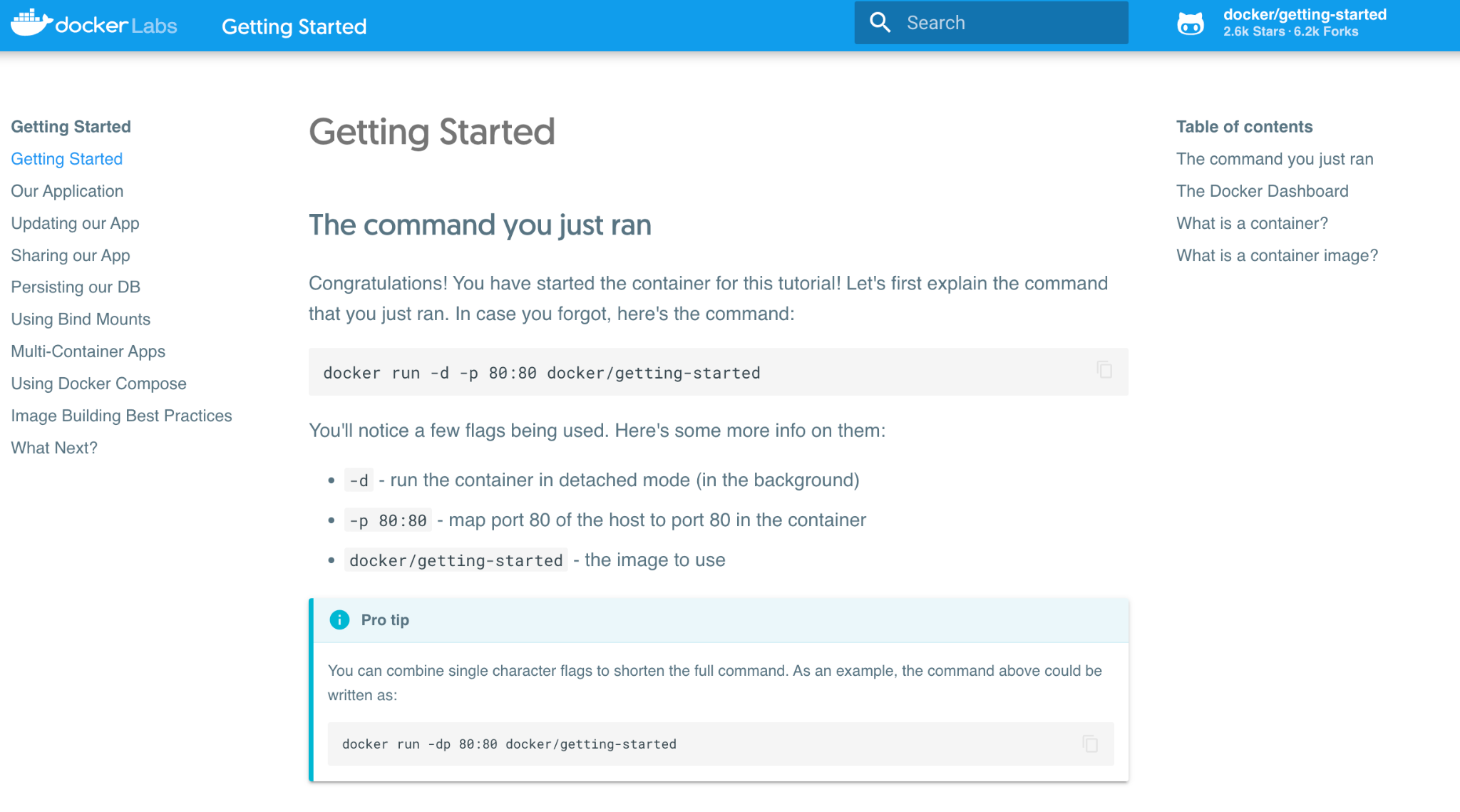
### Samples:

#### Exploring docker run

1. Check for Docker Desktop installed on your local machine.
2. Check also that your Docker CLI is working well by running docker -v 
3. Run the following command in your CLI to get started with a sample docker container:

docker run -d -p 80:80 docker/getting-started

* 1. docker run - Creates and starts a new container from the Docker image.
  2. -d - Runs the container in detached mode, which means that the container runs in the background and does not output any logs to the console.
  3. -p 80:80 - Maps port 80 on the host machine to port 80 in the container. This enables the container to receive HTTP traffic on port 80.
  4. docker/getting-started - Specifies the Docker image to use for the container. In this case, the image is docker/getting-started.

1. To check if you container is running, run: docker ps
   1. This should show the container that you just ran as the output.
2. On your Chrome/ Edge browser, go to <http://localhost>
3. If successful, you should see this:
4. 

#### Exploring Dockerfile

Next, let’s start exploring Dockerfile. Dockerfile is a **script that defines how to build a Docker image for your application**. It contains a set of instructions that specify the environment, dependencies, and configuration of the application or service that the Docker image will contain.

A Dockerfile starts with a FROM instruction, which specifies the base image to use. The base image provides the underlying environment and dependencies that the application or service will run on.

After the FROM instruction, the Dockerfile can include a series of other instructions, such as RUN, COPY, and CMD, which are used to configure the environment, install dependencies, copy files into the image, and specify the command to run when the container is started.

The Dockerfile is used by the docker build command to create a Docker image. When the docker build command is run, it reads the Dockerfile and executes each instruction in order, building up the image layer by layer.

More information can be found below:

| FROM | The FROM instruction initializes a new build stage and sets the Base Image for subsequent instructions. |
| --- | --- |
| WORKDIR | The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it. |
| COPY | The COPY instruction copies files from a local source location to a destination. |
| RUN | The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. |
| EXPOSE | The EXPOSE instruction exposes a particular port inside a Docker container |
| CMD | There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect. The main purpose of a CMD is to provide defaults for an executing container. |

Let’s begin!

1. Clone a docker examples repo by running: git clone https://github.com/docker/awesome-compose.git. Here, you will see different examples of Docker for future use, but let’s run cd awesome-compose/flask
2. Look at the Dockerfile and see what is needed to run this application. Take some time to understand what this is doing.
3. When you’re ready, let’s build a container image. Inside the flask directory, run: docker build -t myapp app/.
4. The above command will show you the different steps needed to build the docker image. Once successfully built, run docker image to see the image created earlier.
5. Next, to run the docker image, run docker run -p 8000:8000 myapp
6. You’ll notice that you are not able to run any other commands without the -d (detach) flag.
7. Let’s go to the browser and enter the url <http://localhost:8000>. You should see a Hello World! Alternatively, run the command curl http://localhost:8000
8. Run CTRL + C on your terminal. Now let’s run the same docker run command with a -d flag now: docker run -dp 8000:8000 myapp
9. You’ll notice that this allows your docker container to run in the background and you’ll get a long string output, which is the container image ID. You can still access the endpoint even when the container is running in the background.
10. Let’s try executing into the docker container via shell. To do this, run the command docker exec -it <container ID> sh. This would allow you to see the files within the container. You can run linux commands such as pwd and ls -lrt
11. To exit, simply run exit
12. To stop the container, run the command docker stop <image ID>