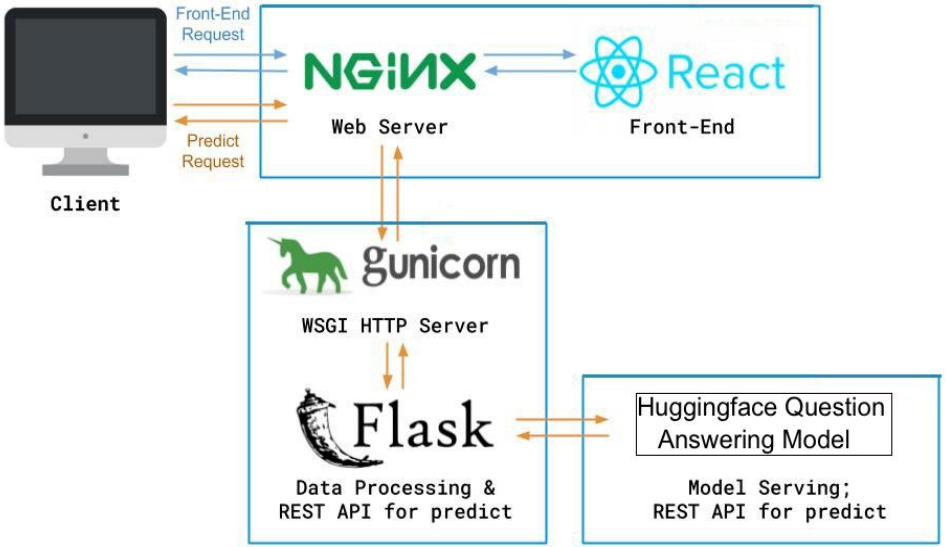
Introduction to docker

How to run your projects on any machine/OS with just a few commands.

Present state of the project

- Flask API working on your local machine (Windows / MacOS / Linux).
- Different versions of libraries working on your local machine.



What does deployment entail?

1. Set the whole project up on a cloud server.
2. Set the system configuration where you project will run.
3. Copying all the files from your machine to the cloud server.
4. Installing all the dependencies of the project.
5. Running the application.

What is the issue with this solution?

What if you had to deploy this app on a cloud server?

What steps should you take to set the whole project up on a different machine?

Agenda

- What is Docker?
- Virtual Machines vs Containers
- Architecture of Docker
- Installing Docker
- Dockerizing our Flask API - Development Workflow

Have you come across a scenario where your teammate could not run your code despite having the exact same codebase?

What could be the reasons?

- One or more files are missing
- Package version mismatch
- Different OSs and configuration settings / environment variables

What is Docker?

A platform for building, running and shipping applications consistently without fail.

Package

- Python 3.8
- Flask 1.4.3
- Numpy 1.12.1

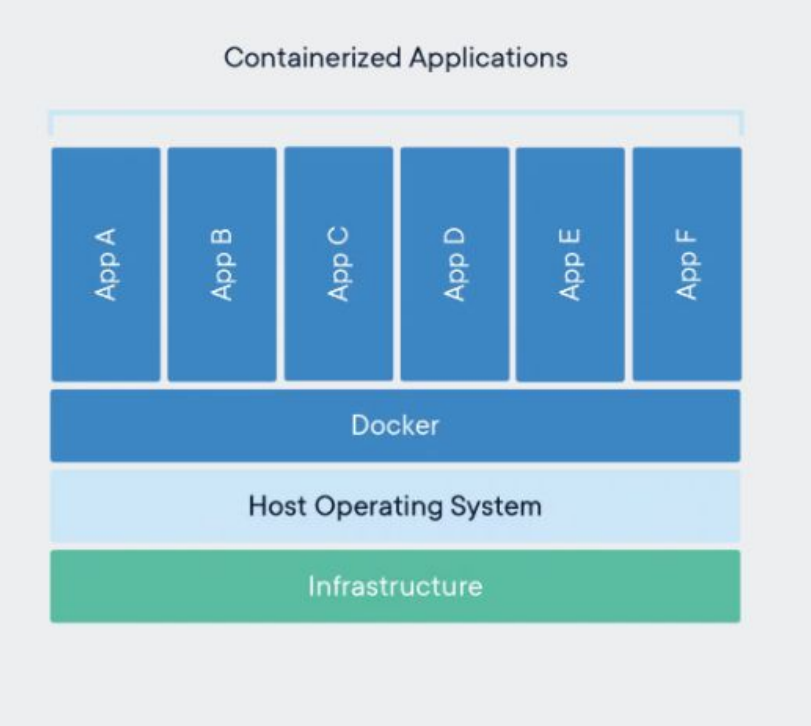
With docker, we can easily containerize our applications and run them anywhere on any machine that runs dockers.

If it runs on your machine, it's definitely going to run on your teammate's machine.

What are those containers?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Run multiple applications inside different containers, each having their own isolated environment!



Virtual Machines

An abstraction of a machine (physical hardware).

Each VM is a full copy of an operating system, it takes time to load.

You can run multiple VMs on a machine but they take up tens of GBs.

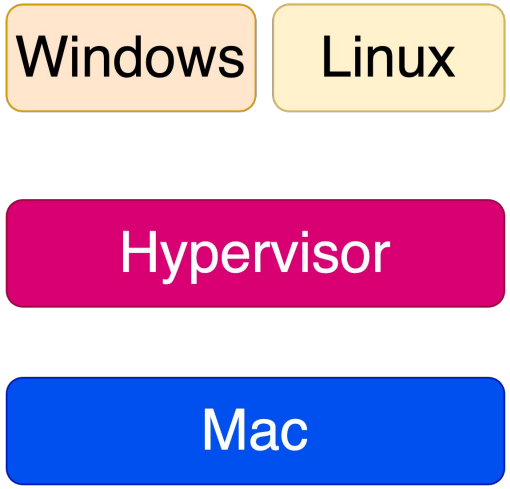
Containers

An abstraction at the app layer that packages code and dependencies together.

They take up less space than VMs.

Can handle more apps and require fewer VMs and OSs.

Virtual Machines - Enabling you to run applications in an isolated environment.



Hypervisor is a cross-platform virtual machine manager.

1. VMware virtual machine
2. VirtualBox

Problems with Virtual Machines

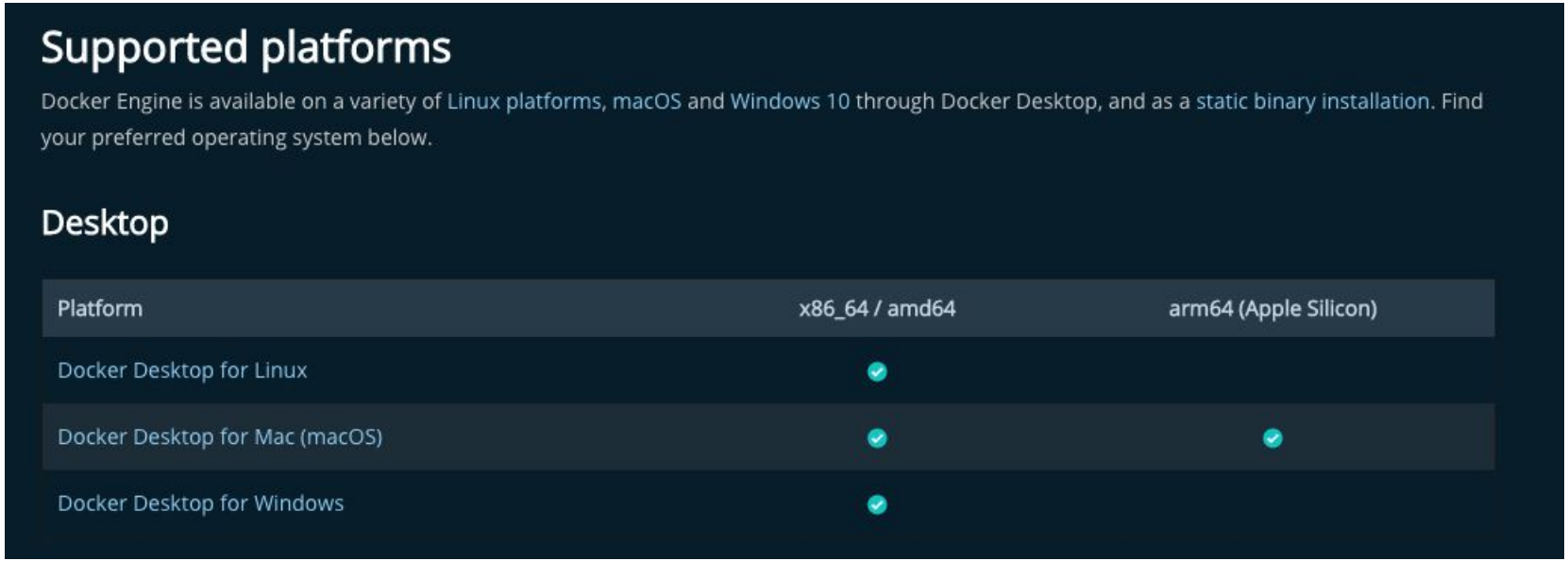
- Each VM needs a full-blown OS
- Slow to start
- Resource intensive - CPU, disk space, etc. of the host machine needs to be divided and there can only be so many VMs that you can run in parallel.

How are containers different?

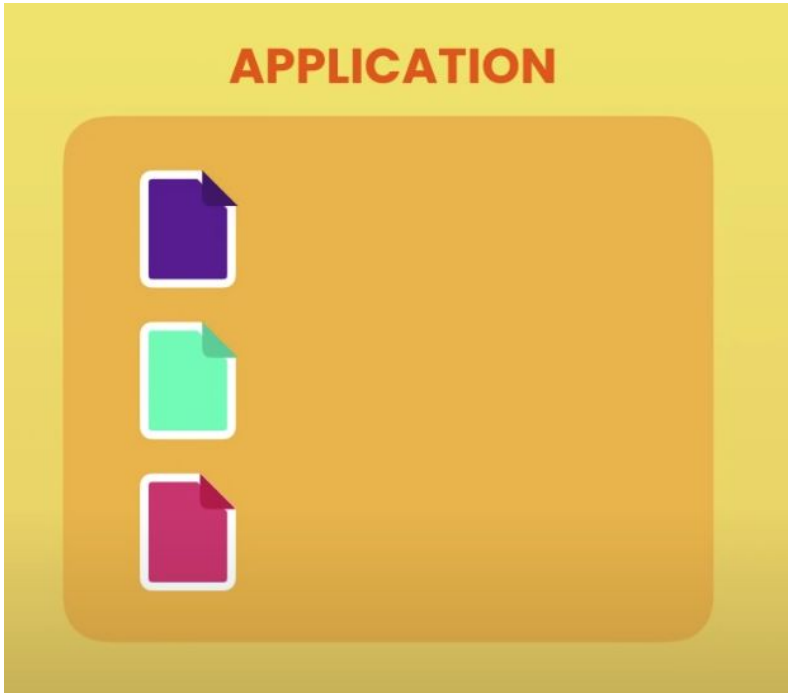
- Allow running multiple apps in isolation.
- Are lightweight
- Use OS of the host machine.
- Starts quickly
- Need less hardware resources

Installing Docker

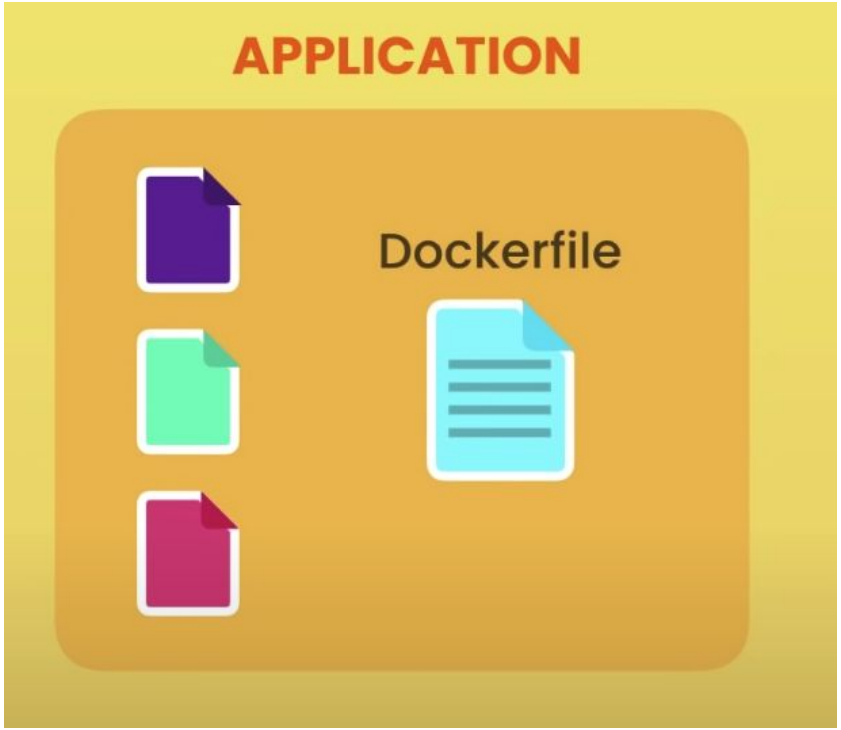
https://docs.docker.com/engine/install/



Development workflow - You have your application

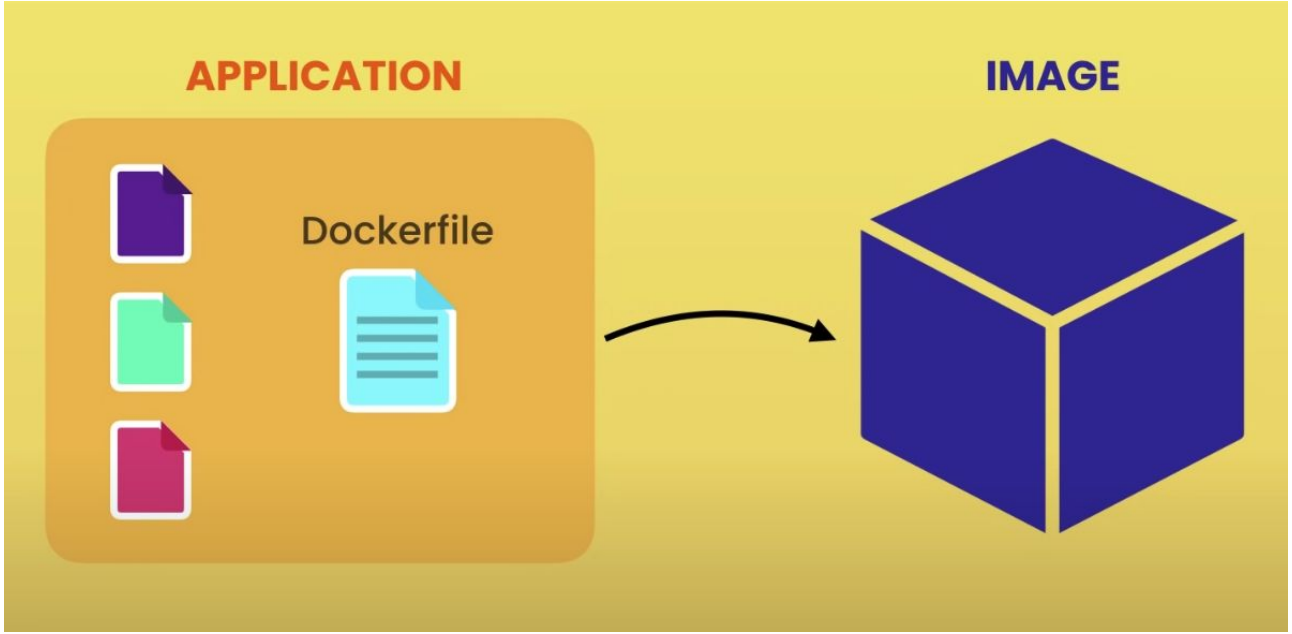


You need to dockerize/containerize your application by adding a Dockerfile



A dockerfile is a plain text files that contains instructions that docker uses to package an application into an image.

Development workflow - You have your application



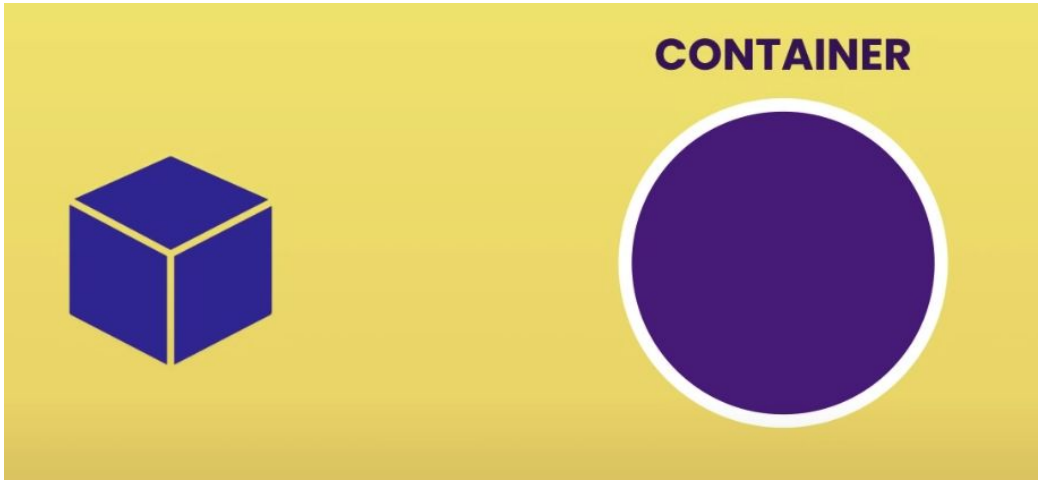
What is an image?

An image contains everything that an application needs to run:

- A cut down OS
- A runtime environment like Python
- Application files
- Third party libraries
- Environment variables

After packaging our app into an image,

Docker creates a container using this image where we can run the application.

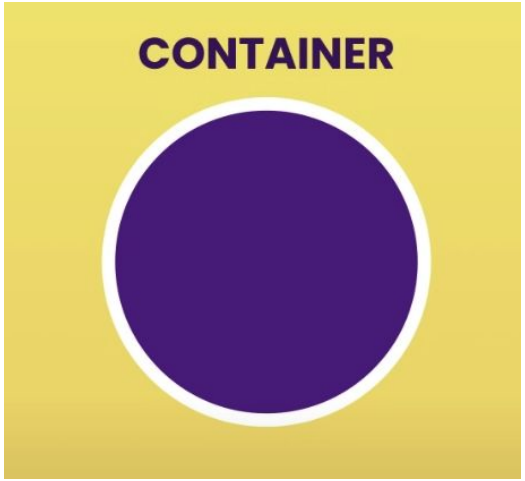


But what is this Container?

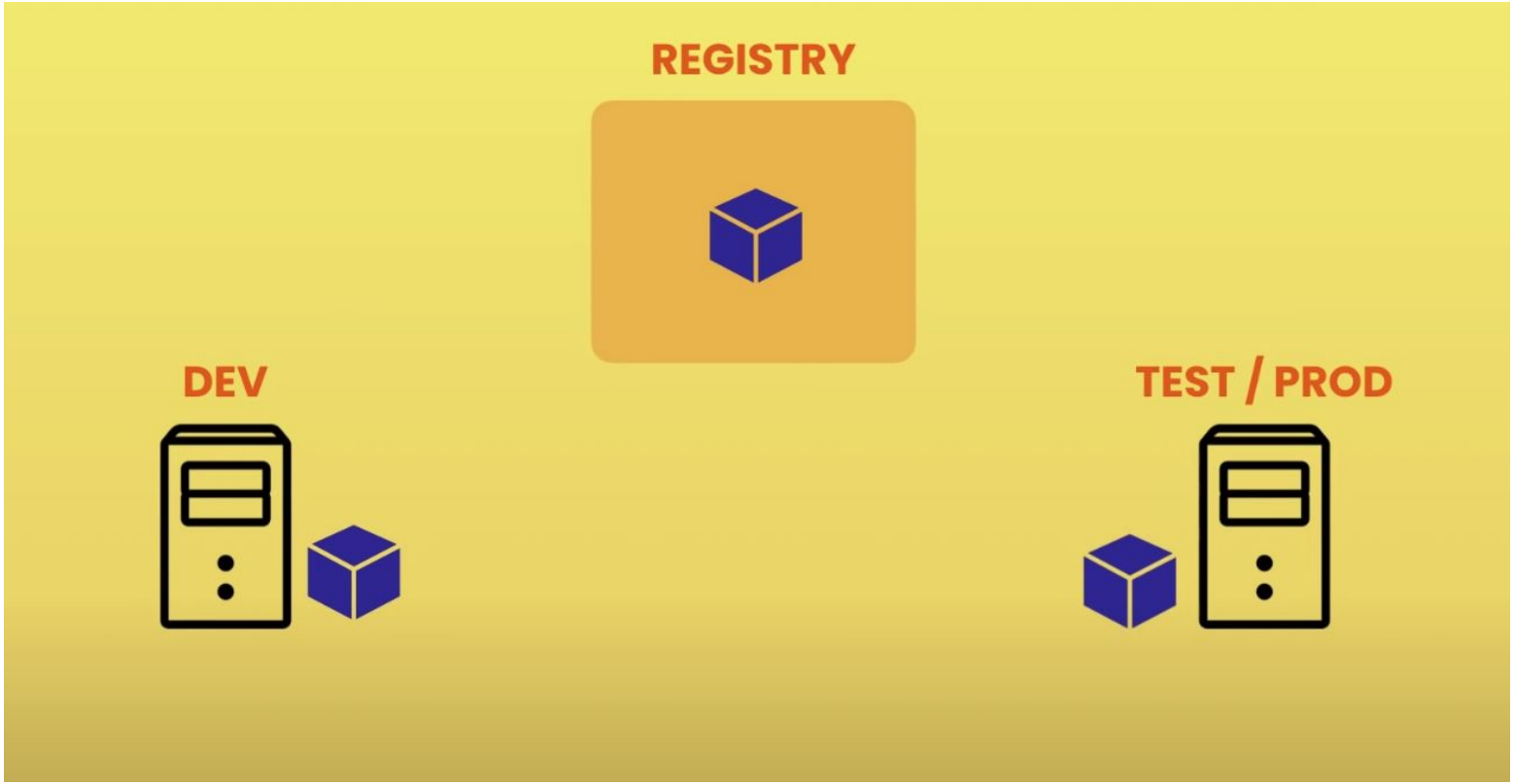
A container is a special kind of process as it has its own file system which is provided by the image.

Our application gets loaded inside a container or a process.

And instead of running our flask app directly using flask/streamlit command, we tell docker to run our app inside a container in an isolated environment.



DockerHub



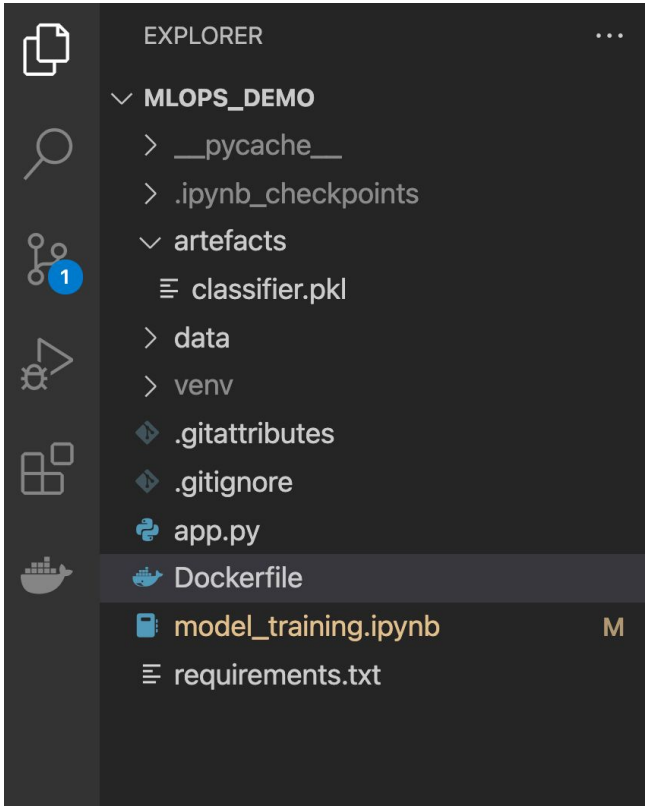
What would be the steps if you had to deploy your flask/Streamlit app?

- Start with an OS
- Install Python, Flask and other dependencies.
- Copy app files
- Run the flask app

If you had to do this all yourself repetitively for all the small changes, you'd have to keep a record of what changed in each version and share detailed release documents.

Let Docker package your application

1. Add a Dockerfile to the root directory of your project.



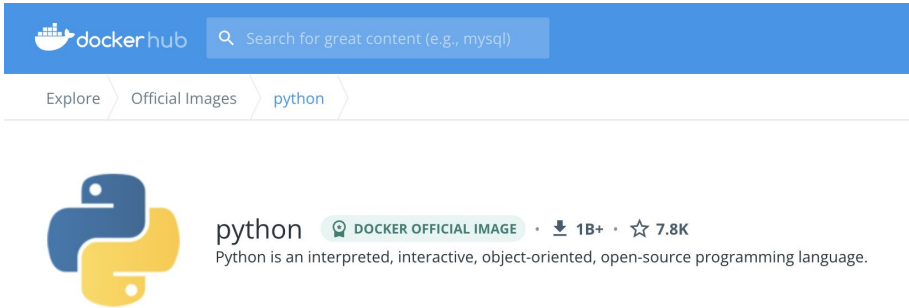
2. OS - Use a base image to start with.

What is a base image?

Dockerhub has a lot of these pre-built images that we can build on top of.

This base image has a bunch of files already and then we can add our files to it, just like inheritance in programming.

For example, we can build start from linux base image and install python on top of it or we can start directly from Python image which is built on top of linux.



How do I know which image to use?

There are so many images of Python built for different python versions and linux distributions.

Each image has a tag that tells you which Linux distribution it is created on top of.

I am choosing Python3.8-slim-buster.

```
FROM python:3.8-slim-buster
```

3. Set a working directory where all the dependencies are going to be installed.

```
1  FROM python:3.8-slim-buster
2
3  WORKDIR /flask-docker
4
```

This is the file system inside the image. You can give any suitable name to the working directory and that would become the main project directory where all the following commands are going to be run.

Further Instructions

1. Command to upgrade pip in the container.
2. Copy requirements.txt from your machine’s directory into the image.
3. Install all the requirements by running the `pip install requirements.txt`
4. COPY all the other files from the current directory into the image.
5. Command to run the flask app.

Terminal: Tell docker to package up our app

docker build -t flask-docker .

- Docker build - builds the image
- -t - we need to provide tag to our image to identify. In our case, flask-docker is the tag.
- We need to specify where the Dockerfile is, in our case in the current directory, so period(.)

Running the app inside the container

docker run -p 8000:5000 -d flask_loan_app

You use the -d flag to run the new container in “detached” mode (in the background).

You also use the -p flag to create a mapping between the host’s port 5000 to the container’s port 8000.

Without the port mapping, you wouldn’t be able to access the application.