

Distributing Docker Images

INTRODUCTION TO DOCKER



Tim Sangster

Software Engineer @ DataCamp

Private Docker registries

- Unlike Docker official images there is no quality guarantee
- Name starts with the url of the private registry

```
dockerhub.myprivateregistry.com/classify_spam
```

```
docker pull dockerhub.myprivateregistry.com/classify_spam:v1
```

```
Using tag: v1
latest: Pulling from dockerhub.myprivateregistry.com
ed02c6ade914: Pull complete
Digest: sha256:b6b83d3c331794420340093eb706b6f152d9c1fa51b262d9bf34594887c2c7ac
Status: Downloaded newer image for dockerhub.myprivateregistry.com/classify_spam:v1
dockerhub.myprivateregistry.com/classify_spam:v1
```

Pushing to a registry

```
docker image push <image name>
```

Pushing to a specific registry --> name of the image needs to start with the registry url

```
docker tag classify_spam:v1 dockerhub.myprivateregistry.com/classify_spam:v1
```

```
docker image push dockerhub.myprivateregistry.com/classify_spam:v1
```

Authenticating against a registry

- Docker official images --> No authentication needed
- Private Docker repository --> Owner can choose

```
docker login dockerhub.myprivateregistry.com
```

```
user@pc ~ % docker login dockerhub.myprivateregistry.com
Username: student
Password:
Login succeeded
```

Docker images as files

Sending a Docker image to one or a few people? Send it as a file!

Save an image

```
docker save -o image.tar classify_spam:v1
```

Load an image

```
docker load -i image.tar
```

Summary of new commands

Usage	Command
Pull image from private registry	<code>docker pull <private-registry-url>/<image-name></code>
Name an image	<code>docker tag <old-name> <new-name></code>
Push an image	<code>docker image push <image-name></code>
Login to private registry	<code>docker login <private-registry-url></code>
Save image to file	<code>docker save -o <file-name> <image-name></code>
Load image from file	<code>docker load -i <file-name></code>

Let's practice!

INTRODUCTION TO DOCKER

Creating your own Docker images

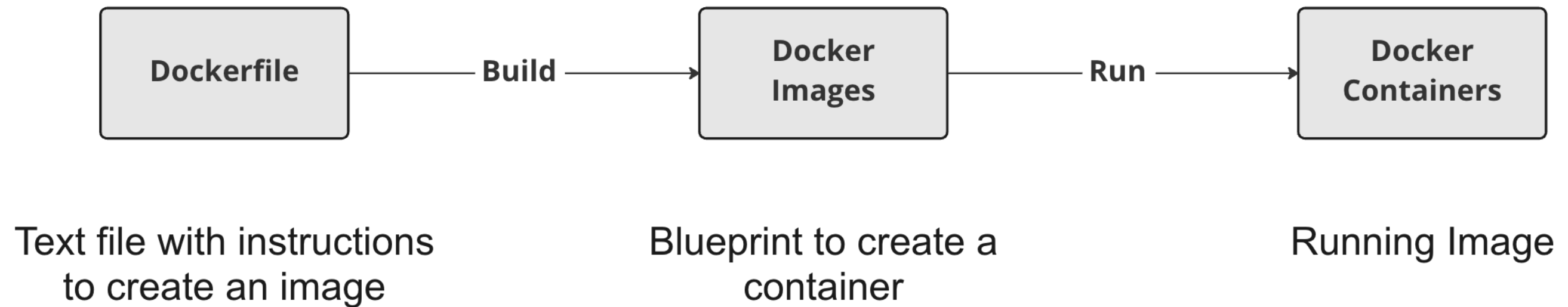
INTRODUCTION TO DOCKER



Tim Sangster

Software Engineer @ DataCamp

Creating images with Dockerfiles



Starting a Dockerfile

A Dockerfile always start from another image, specified using the FROM instruction.

```
FROM postgres
FROM ubuntu
FROM hello-world
FROM my-custom-data-pipeline
```

```
FROM postgres:15.0
FROM ubuntu:22.04
FROM hello-world:latest
FROM my-custom-data-pipeline:v1
```

Building a Dockerfile

Building a Dockerfile creates an image.

```
docker build /location/to/Dockerfile
docker build .
```

```
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 54B
...
=> CACHED [1/1] FROM docker.io/library/ubuntu
=> exporting to image
=> => exporting layers
=> => writing image sha256:a67f41b1d127160a7647b6709b3789b1e954710d96df39ccaa21..
```

Naming our image

In practice we almost always give our images a name using the `-t` flag:

```
docker build -t first_image .
```

```
...  
=> => writing image sha256:a67f41b1d127160a7647b6709b3789b1e954710d96df39ccaa21..  
=> => naming to docker.io/library/first_image
```

```
docker build -t first_image:v0 .
```

```
=> => writing image sha256:a67f41b1d127160a7647b6709b3789b1e954710d96df39ccaa21..  
=> => naming to docker.io/library/first_image:v0
```

Customizing images

```
RUN <valid-shell-command>
```

```
FROM ubuntu
```

```
RUN apt-get update
```

```
RUN apt-get install -y python3
```

Use the -y flag to avoid any prompts:

```
...  
After this operation, 22.8 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

Building a non-trivial Dockerfile

When building an image Docker actually runs commands after RUN

Docker running `RUN apt-get update` takes the same amount of time as us running it!

```
root@host:/# apt-get update
Get:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease [270 kB]
...
Get:17 http://ports.ubuntu.com/ubuntu-ports jammy-security/restricted arm64 Pack..
Fetched 23.0 MB in 2s (12.3 MB/s)
Reading package lists... Done
```

Summary

Usage	Dockerfile Instruction
Start a Dockerfile from an image	FROM <image-name>
Add a shell command to image	RUN <valid-shell-command>
Make sure no user input is needed for the shell-command.	RUN apt-get install -y python3

Usage	Shell Command
Build image from Dockerfile	docker build /location/to/Dockerfile
Build image in current working directory	docker build .
Choose a name when building an image	docker build -t first_image .

Let's practice!

INTRODUCTION TO DOCKER

Managing files in your image

INTRODUCTION TO DOCKER



Tim Sangster

Software Engineer @ DataCamp

COPYing files into an image

The COPY instruction copies files from our local machine into the image we're building:

```
COPY <src-path-on-host> <dest-path-on-image>  
COPY /projects/pipeline_v3/pipeline.py /app/pipeline.py
```

```
docker build -t pipeline:v3 .  
...  
[4/4] COPY ./projects/pipeline_v3/pipeline.py /app/pipeline.py
```

If the destination path does not have a filename, the original filename is used:

```
COPY /projects/pipeline_v3/pipeline.py /app/
```

COPYing folders

Not specifying a filename in the src-path will copy all the file contents.

```
COPY <src-folder> <dest-folder>  
COPY /projects/pipeline_v3/ /app/
```

`COPY /projects/pipeline_v3/ /app/` will copy everything under `pipeline_v3/` :

```
/projects/  
  pipeline_v3/  
    pipeline.py  
    requirements.txt  
    tests/  
      test_pipeline.py
```

Copy files from a parent directory

```
/init.py
/projects/
  Dockerfile
  pipeline_v3/
    pipeline.py
```

If our current working directory is in the `projects/` folder.

We can't copy `init.py` into an image.

```
docker build -t pipeline:v3 .
=> ERROR [4/4] COPY ../init.py /      0.0s
failed to compute cache key: "../init.py" not found: not found
```

Downloading files

Instead of copying files from a local directory, files are often downloaded in the image build:

- Download a file

```
RUN curl <file-url> -o <destination>
```

- Unzip the file

```
RUN unzip <dest-folder>/<filename>.zip
```

- Remove the original zip file

```
RUN rm <copy_directory>/<filename>.zip
```

Downloading files efficiently

- Each instruction that downloads files adds to the total size of the image.
- Even if the files are later deleted.
- The solution is to download, unpack and remove files in a single instruction.

```
RUN curl <file_download_url> -o <destination_directory>/<filename>.zip \  
&& unzip <destination_directory>/<filename>.zip -d <unzipped-directory> \  
&& rm <destination_directory>/<filename>.zip
```

Summary

Usage	Dockerfile Instruction
Copy files from host to the image	<code>COPY <src-path-on-host> <dest-path-on-image></code>
Copy a folder from host to the image	<code>COPY <src-folder> <dest-folder></code>
We can't copy from a parent directory where we build a Dockerfile	<code>COPY ../<file-in-parent-directory> /</code>

Keep images small by downloading, unzipping, and cleaning up in a single RUN instruction:

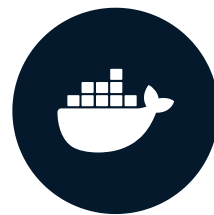
```
RUN curl <file_download_url> -o <destination_directory> \  
&& unzip <destination_directory>/<filename>.zip -d <unzipped-directory> \  
&& rm <destination_directory>/<filename>.zip
```

Let's practice!

INTRODUCTION TO DOCKER

Choosing a start command for your Docker image

INTRODUCTION TO DOCKER



Tim Sangster

Software Engineer @ DataCamp

What is a start command?

The hello-world image prints text and then stops.

```
docker run hello-world
```

```
Hello from Docker!
```

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon created a new container from the hello-world image which runs executable that produces the output you are currently reading.
3. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

What is a start command?

An image with python could start python on startup.

```
docker run python3-sandbox
```

```
Python 3.10.6 (main, Nov  2 2022, 18:53:38) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>  
...
```

```
....  
>>> exit()  
repl@host:/#
```

Running a shell command at startup

```
CMD <shell-command>
```

The CMD instruction:

- Runs when the image is started.
- Does not increase the size of the image .
- Does not add any time to the build.
- If multiple exist, only the last will have an effect.

Typical usage

Starting an application to run a workflow or that accepts outside connections.

```
CMD python3 my_pipeline.py
```

```
CMD postgres
```

Starting a script that, in turn, starts multiple applications

```
CMD start.sh
```

```
CMD python3 start_pipeline.py
```

When will it stop?

- hello-world image -> After printing text
- A database image -> When the database exits

A more general image needs a more general start command.

- An Ubuntu image -> When the shell is closed

Overriding the default start command

Starting an image

```
docker run <image>
```

Starting an image with a custom start command

```
docker run <image> <shell-command>
```

Starting an image interactively with a custom start command

```
docker run -it <image> <shell-command>
```

```
docker run -it ubuntu bash
```

Summary

Usage	Dockerfile Instruction
Add a shell command run when a container is started from the image.	CMD <shell-command>

Usage	Shell Command
Override the CMD set in the image	docker run <image> <shell-command>
Override the CMD set in the image and run interactively	docker run -it <image> <shell-command>

Let's practice!

INTRODUCTION TO DOCKER