Part two

Dockerfiles.
How to create your image from scratch.

Nicola Malavasi - Marie Curie Fellow @ MPE - 16/04/2024

Creating your own image

It may happen that the image you need is not on Docker Hub or that you want to create your own image.

Docker allows you to do this via a script: a Dockerfile

Essentially a Dockerfile is a script where you start from an existing image, you modify it adding features, and you create a new image out of it to use for a container.

Images are immutable once created: if software is added to an existing image a new one needs to be created.

Inefficient to install software at container stage: once container is deleted, sofware is lost and needs to be reinstalled in new container.

First steps with containers

Actions:

- Image pull: download a pre-made image from a registry.
- Image build: create image from scratch.
- Container start: start a container from an image (i.e. a contained process in an environment set by an image).
- Container run/exec: execute a command in a container.
- Container stop: stop a container.

Dockerfile example

Users > nmalavasi > Desktop > PUNCH_useful > Presentations > PYA_tutorial_container > tutorial_material > Dockerfile_python_pip > ...

```
1  # syntax=docker/dockerfile:1
2
3  #Start from python image
4  FROM python:latest
5
6  #Use pip to install matplotlib
7  RUN pip install matplotlib
```

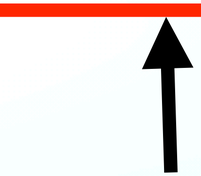
This already constitutes a basic Dockerfile: takes Python as base image and installs matplotlib via pip. The new image created will have both.

Building from Dockerfile

```
docker build -t pm_i -f ./Dockerfile_python_pip .
```

Building from Dockerfile

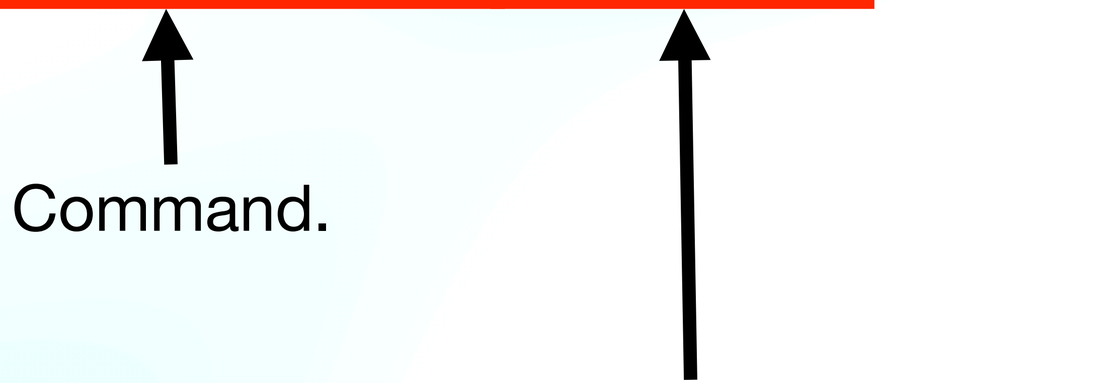
```
docker build -t pm_i -f ./Dockerfile_python_pip .
```



Command.

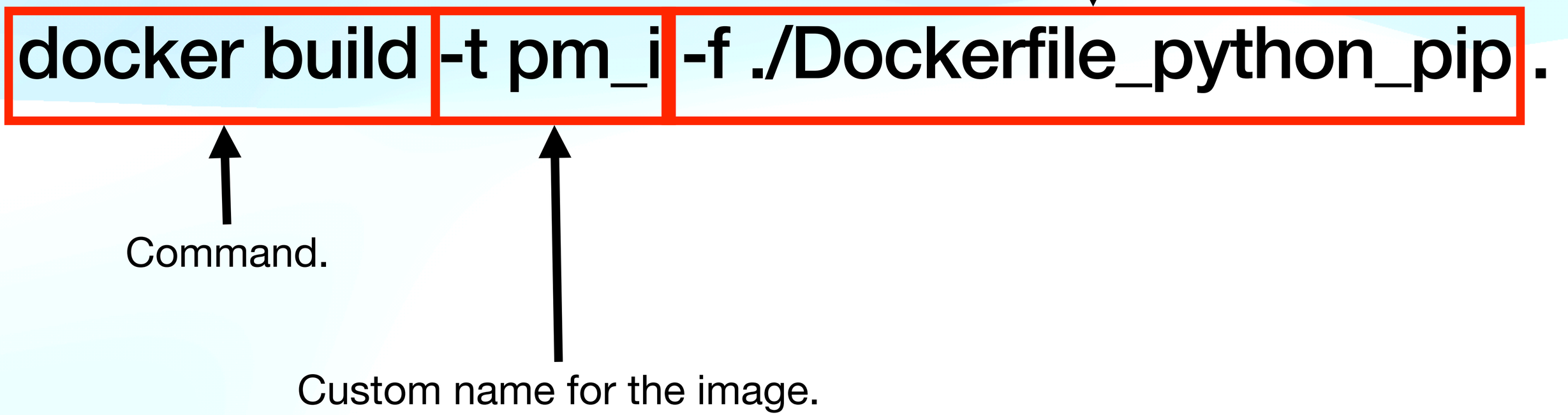
Building from Dockerfile

```
docker build -t pm_i -f ./Dockerfile_python_pip .
```

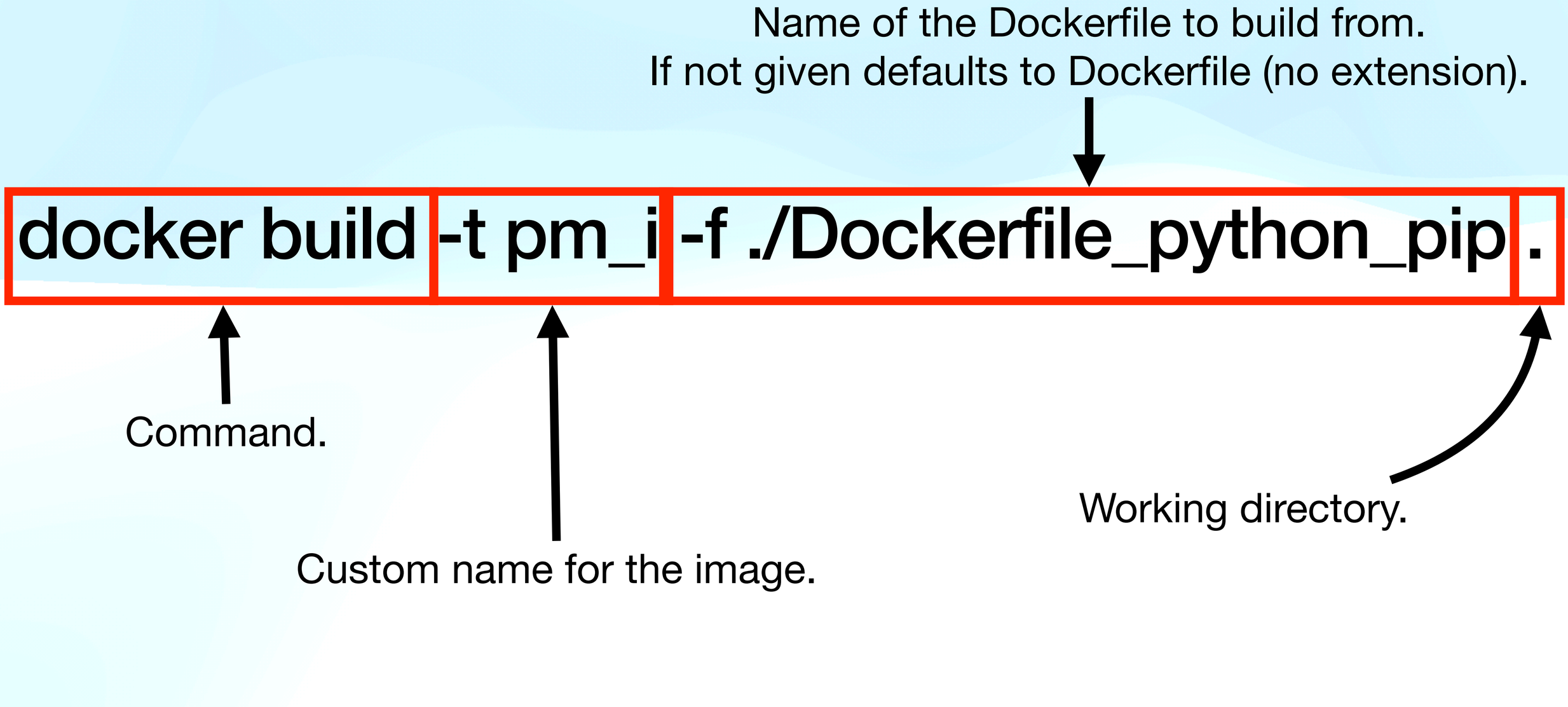


Building from Dockerfile

Name of the Dockerfile to build from.
If not given defaults to Dockerfile (no extension).



Building from Dockerfile



Building from Dockerfile

```
nmalavasi@ga-lt7982 ~ % cd Desktop/PUNCH_useful/Presentations/PYA_tutorial_container/tutorial_material
nmalavasi@ga-lt7982 tutorial_material % ls
Dockerfile_python_pip          disperse_dockerfile            example_script_simple.py
nmalavasi@ga-lt7982 tutorial_material % docker build -t pm_i -f ./Dockerfile_python_pip .
[+] Building 20.1s (10/10) FINISHED
 => [internal] load build definition from Dockerfile_python_pip                                                    0.0s
 => => transferring dockerfile: 187B                                                                               0.0s
 => [internal] load .dockerignore                                                                                  0.0s
 => => transferring context: 2B                                                                                    0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                         1.9s
 => docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56  0.5s
 => => resolve docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56  0.0s
 => => sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56 8.40kB / 8.40kB                      0.0s
 => => sha256:ab4d84ec8cad8a7b0e8b6ad1fb49536f000b86e30993c74a251bb16f3e6e3f5b 482B / 482B                          0.0s
 => => sha256:6bb827dc58ef706c3d261e31d3a2ebd9965fa640ec3bc1cef4c98d8068b55685 1.26kB / 1.26kB                     0.0s
 => => sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7 11.23MB / 11.23MB                   0.3s
 => => extracting sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7                          0.1s
 => [internal] load build definition from Dockerfile_python_pip                                                    0.0s
 => [internal] load metadata for docker.io/library/python:latest                                                   1.6s
 => [internal] load .dockerignore                                                                                  0.0s
 => [1/2] FROM docker.io/library/python:latest@sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f  8.4s
 => => resolve docker.io/library/python:latest@sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f  0.0s
 => => sha256:374850c6db1702573c7004d630027931be318b2d71cb28e890e2fcd0f0730712 23.58MB / 23.58MB                   1.1s
 => => sha256:421c44fab18bc9f4c62ca481e074d50b3a036e7c95c7607b6d036c34d67c5264 63.99MB / 63.99MB                   1.8s
 => => sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f 2.14kB / 2.14kB                     0.0s
 => => sha256:18ba2f0e30767e776ab7dd7056c8342b301fa148a6121d3e044b68c95d43a3fe 7.11kB / 7.11kB                     0.0s
 => => sha256:1e92f3a395ff98a929e797a3c392bb6d0f05531068d34b81d3cd41ed6ce82ca4 49.60MB / 49.60MB                   1.2s
 => => sha256:1b43f83fd0fbecda5f733fd91d8f14a58649831cad94144e58cdcd6229d603e1 2.01kB / 2.01kB                     0.0s
 => => sha256:b9717a38adec9939307bba3151627c24c2bbac069b221c2fcb0500a40f2736ec 202.54MB / 202.54MB                 4.4s
 => => sha256:51795e508cf7cc0fd50df0d1abf30e0e99cc3b9d8c35df081e8c642026675d79 6.47MB / 6.47MB                     1.6s
 => => extracting sha256:1e92f3a395ff98a929e797a3c392bb6d0f05531068d34b81d3cd41ed6ce82ca4                          1.6s
 => => sha256:bc54e015d093dbbb737406a0b0435c74686b53c0a66562dc5a6f30eff80672ee 22.21MB / 22.21MB                   2.2s
 => => sha256:3822d8fd74911d66b2022e5bbeb89912c8607001c209cb6df58d6cd84a7b07a0 241B / 241B                         2.0s
 => => sha256:4fc85492ad0c89cc848b998232abbd11e3a39af28fd0278ff52e94dd48b60066 2.70MB / 2.70MB                     2.3s
 => => extracting sha256:374850c6db1702573c7004d630027931be318b2d71cb28e890e2fcd0f0730712                          0.5s
 => => extracting sha256:421c44fab18bc9f4c62ca481e074d50b3a036e7c95c7607b6d036c34d67c5264                          1.3s
 => => extracting sha256:b9717a38adec9939307bba3151627c24c2bbac069b221c2fcb0500a40f2736ec                          2.6s
 => => extracting sha256:51795e508cf7cc0fd50df0d1abf30e0e99cc3b9d8c35df081e8c642026675d79                          0.1s
 => => extracting sha256:bc54e015d093dbbb737406a0b0435c74686b53c0a66562dc5a6f30eff80672ee                          0.3s
 => => extracting sha256:3822d8fd74911d66b2022e5bbeb89912c8607001c209cb6df58d6cd84a7b07a0                          0.0s
 => => extracting sha256:4fc85492ad0c89cc848b998232abbd11e3a39af28fd0278ff52e94dd48b60066                          0.1s
 => [2/2] RUN pip install matplotlib                                                                               7.0s
 => exporting to image                                                                                             0.5s
 => => exporting layers                                                                                            0.5s
 => => writing image sha256:f6e6ee04e6c63947ef423fc096da30299d3f97a6430e586f37f5ecdf714f77a1                       0.0s
 => => naming to docker.io/library/pm_i                                                                            0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
nmalavasi@ga-lt7982 tutorial_material %
```

The new image is created via “docker build” (specifying file name if needed).

Building from Dockerfile

```
nmalavasi@ga-lt7982 ~ % cd Desktop/PUNCH_useful/Presentations/PYA_tutorial_container/tutorial_material
nmalavasi@ga-lt7982 tutorial_material % ls
Dockerfile_python_pip          disperse_dockerfile            example_script_simple.py
nmalavasi@ga-lt7982 tutorial_material % docker build -t pm_i -f ./Dockerfile_python_pip .
[+] Building 20.1s (10/10) FINISHED
 => [internal] load build definition from Dockerfile_python_pip                                                0.0s
 => => transferring dockerfile: 187B                                                                           0.0s
 => [internal] load .dockerignore                                                                              0.0s
 => => transferring context: 2B                                                                                0.0s
 => resolve image config for docker.io/docker/dockerfile:1                                                     1.9s
 => docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56  0.5s
 => => resolve docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56  0.0s
 => => sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56 8.40kB / 8.40kB                  0.0s
 => => sha256:ab4d84ec8cad8a7b0e8b6ad1fb49536f000b86e30993c74a251bb16f3e6e3f5b 482B / 482B                      0.0s
 => => sha256:6bb827dc58ef706c3d261e31d3a2ebd9965fa640ec3bc1cef4c98d8068b55685 1.26kB / 1.26kB                 0.0s
 => => sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7 11.23MB / 11.23MB               0.3s
 => => extracting sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7                      0.1s
 => [internal] load build definition from Dockerfile_python_pip                                                0.0s
 => [internal] load metadata for docker.io/library/python:latest                                               1.6s
 => [internal] load .dockerignore                                                                              0.0s
 => [1/2] FROM docker.io/library/python:latest@sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f  8.4s
 => => resolve docker.io/library/python:latest@sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f  0.0s
 => => sha256:374850c6db1702573c7004d630027931be318b2d71cb28e890e2fcd0f0730712 23.58MB / 23.58MB               1.1s
 => => sha256:421c44fab18bc9f4c62ca481e074d50b3a036e7c95c7607b6d036c34d67c5264 63.99MB / 63.99MB               1.8s
 => => sha256:e0e2713ebf0f7b114b8bf9fbcaba9a69ef80e996b9bb3fa5837e42c779dcdc0f 2.14kB / 2.14kB                 0.0s
 => => sha256:18ba2f0e30767e776ab7dd7056c8342b301fa148a6121d3e044b68c95d43a3fe 7.11kB / 7.11kB                 0.0s
 => => sha256:1e92f3a395ff98a929e797a3c392bb6d0f05531068d34b81d3cd41ed6ce82ca4 49.60MB / 49.60MB               1.2s
 => => sha256:1b43f83fd0fbecda5f733fd91d8f14a58649831cad94144e58cdcd6229d603e1 2.01kB / 2.01kB                 0.0s
 => => sha256:b9717a38adec9939307bba3151627c24c2bbac069b221c2fcb0500a40f2736ec 202.54MB / 202.54MB             4.4s
 => => sha256:51795e508cf7cc0fd50df0d1abf30e0e99cc3b9d8c35df081e8c642026675d79 6.47MB / 6.47MB                 1.6s
 => => extracting sha256:1e92f3a395ff98a929e797a3c392bb6d0f05531068d34b81d3cd41ed6ce82ca4                      1.6s
 => => sha256:bc54e015d093dbbb737406a0b0435c74686b53c0a66562dc5a6f30eff80672ee 22.21MB / 22.21MB               2.2s
 => => sha256:3822d8fd74911d66b2022e5bbeb89912c8607001c209cb6df58d6cd84a7b07a0 241B / 241B                     2.0s
 => => sha256:4fc85492ad0c89cc848b998232abbd11e3a39af28fd0278ff52e94dd48b60066 2.70MB / 2.70MB                 2.3s
 => => extracting sha256:374850c6db1702573c7004d630027931be318b2d71cb28e890e2fcd0f0730712                      0.5s
 => => extracting sha256:421c44fab18bc9f4c62ca481e074d50b3a036e7c95c7607b6d036c34d67c5264                      1.3s
 => => extracting sha256:b9717a38adec9939307bba3151627c24c2bbac069b221c2fcb0500a40f2736ec                      2.6s
 => => extracting sha256:51795e508cf7cc0fd50df0d1abf30e0e99cc3b9d8c35df081e8c642026675d79                      0.1s
 => => extracting sha256:bc54e015d093dbbb737406a0b0435c74686b53c0a66562dc5a6f30eff80672ee                      0.3s
 => => extracting sha256:3822d8fd74911d66b2022e5bbeb89912c8607001c209cb6df58d6cd84a7b07a0                      0.0s
 => => extracting sha256:4fc85492ad0c89cc848b998232abbd11e3a39af28fd0278ff52e94dd48b60066                      0.1s
 => [2/2] RUN pip install matplotlib                                                                           7.0s
 => exporting to image                                                                                         0.5s
 => => exporting layers                                                                                        0.5s
 => => writing image sha256:f6e6ee04e6c63947ef423fc096da30299d3f97a6430e586f37f5ecdf714f77a1                   0.0s
 => => naming to docker.io/library/pm_i                                                                        0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
nmalavasi@ga-lt7982 tutorial_material %
```

The new image is created via “docker build” (specifying file name if needed).

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

FROM

FROM allows you to select the starting “base” image. Only one FROM statement for Dockerfile.

It will pull the image from Docker Hub and execute the next steps of the build inside that image.

```
# syntax=docker/dockerfile:1

#Start from  python image
FROM python:latest

#Start from  ubuntu image
FROM ubuntu:latest

#Start from  debian image
FROM debian:latest

#GENERALLY ONLY ONE FROM STATEMENT
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

RUN

RUN executes a command. It can be any command that is accepted within the current OS/ environment of the image.

E.g. in linux image linux commands will be executed, provided software is installed.

```
# syntax=docker/dockerfile:1

#Start from  pyton image
FROM python:latest

#Execute command
RUN pip install matplotlib

#Non-python examples
RUN apt-get install git

RUN git clone my_repository

#Can be any command
RUN wget my_url/my_tarball.tar

RUN rm /file_I_dont_need.txt
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

ENV

ENV sets the value of an environmental variable

```
# syntax=docker/dockerfile:1

#Start from  pyton image
FROM python:latest

#Set environmental variable HOME
ENV HOME=/folder/path
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

COPY

COPY copies files from a local folder to inside the image. These files will be available within the containers run from that image.

```
# syntax=docker/dockerfile:1

#Start from  python image
FROM python:latest

#Set environmental variable HOME
ENV HOME=/folder/path

#Copy file requirements.txt from outside to inside
COPY requirements.txt ${HOME}/my_dir

#Pip install requirements.txt
RUN pip install -r ${HOME}/my_dir/requirements.txt

#Remove file: we don't want this in container
RUN rm ${HOME}/my_dir/requirements.txt

#Copy source code inside: we want this in container
COPY my_file.exe ./
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

USER

By default, unless specified otherwise, all commands at image creation are run as root (WITHIN the image). USER allows to switch to a different user (if it exists). There is no going back afterwards.

```
# syntax=docker/dockerfile:1

#Start from  pyton image
FROM python:latest

# I AM ROOT - no need to sudo
RUN apt-get install my_package

#Switch to different user
USER nmalavasi

# I AM NO LONGER ROOT - This does not work anymore
RUN apt-get install my_other_package

#When I run a container in this image I will be user "nmalavasi"
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

WORKDIR

Sets the current working directory.

```
# syntax=docker/dockerfile:1

#Start from  pyton image
FROM python:latest

#By default we are in /

#Create new folder
RUN mkdir /path_to_folder

#Equivalent to cd /path_to_folder
WORKDIR /path_to_folder
```

Useful keywords

- FROM: uses existing image as base
- RUN: executes a command
- ENV: sets environmental variable
- COPY: copies local files inside image (useful e.g. for source code)
- USER: sets user information to be used when container is created from image (otherwise user is root!)
- WORKDIR: sets working directory
- CMD: sets the command that is executed when container is created from image

Dockerfile reference: https://docs.docker.com/reference/dockerfile/

CMD

If present, this command is executed when a container is created. There can be only one CMD instruction.

```
# syntax=docker/dockerfile:1

#Start from  pyton image
FROM python:latest

#Install module
RUN pip install matplotlib

#This starts cointainer with python prompt
CMD python3
```

Part three

Exercise.

Create your Dockerfile and install your code.

Nicola Malavasi - Marie Curie Fellow @ MPE - 16/04/2024

Exercise suggestion

Simple

- Pull image and start container: ubuntu, python, any other language you are comfortable with.
- Pull image and start three containers: experiment with starting and stopping them, switch from one to the other or experience running them in parallel.

Exercise suggestion

Intermediate

- Pull image, start container, execute command in container.
- Bind mount volume into container, experiment with file input and output from container to local folder.

Exercise suggestion

Advanced

- Do you have a code to install? Write a Dockerfile for it, test it by running a container.
- Do you have a sample analysis? Bind mount a volume to your newly created container, test software execution with input/output.
- Write a Dockerfile for a generic software (e.g. python? Other language?). Start from a base image, add on top to it. Set environmental variables, change the user and see how it is reflected in the container.
- Try to add a command to it.