Part 1

[slide 4]

Check the list of docker images:

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

Get our first docker image (**syntax** `docker pull IMAGE`):

```
$ docker pull python:alpine
```

Check the list of docker images:

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
python              alpine              8eb1c554687d        3 weeks ago         90.4MB
```

[slide 5]

Run the our first container:

```
$ docker run python:alpine
```

Nothing happened, since there's no pre-configured command to run.

Let's indicate a command then (**syntax** `docker run IMAGE CMD`):

```
$ docker run python:alpine python --version
Python 3.6.5
```

Hello world!

Python:

```
$ python -c 'print("hello world!")'
hello world!
```

Python in Docker:

```
$ docker run python:alpine python -c 'print("hello world!")'
hello world!
```

Alternative:

```
$ echo "hello world!"
hello world!
$ docker run python:alpine echo "hello world!"
hello world!
```

Listing containers running

Let's check the list of containers running:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
```

Well, nothing as expected.

Let's run a container for some time then.

```
$ man sleep
$ docker run python:alpine sleep 10
```

In the next 10 seconds, run in another terminal:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED                  STATUS              PORTS
6e6be0c8f31c        python:alpine       "sleep 10"          Less than a second ago   Up 1 second
```

[slide 5]

Let's create simple container

Our first `Dockerfile`:

```
FROM python:alpine

CMD python -c 'print("hello world!")'
```

Let's now build the docker image with a repository and tag, as in `python:alpine` (**syntax** `docker build -f FILE -t IMAGE DIR`):

```
$ docker build -f Dockerfile -t vitorenesduarte/tutorial:hello .
```

And check that the new image is in the list of images:

```
$ docker images
REPOSITORY                 TAG                 IMAGE ID            CREATED              SIZE
vitorenesduarte/tutorial   hello               49aa76850e83        About a minute ago   90.4MB
python                     alpine              8eb1c554687d        3 weeks ago          90.4MB
```

Let's run our app:

```
$ docker run vitorenesduarte/tutorial:hello
hello world!
```

But this is not how we write apps, right?

Let's then create a file named `app.py` with:

```
print("hello world!")
```

Verify it is okay:

```
$ python app.py
hello world!
```

And modify `Dockerfile` to:

```
FROM python:alpine

COPY app.py /

CMD python app.py
```

Let's build the image again:

```
$ docker build -t vitorenesduarte/tutorial:hello .
```

Notice we didn't indicate which file to use. Docker tries to find a file named `Dockerfile` in the directory passed as argument.

Verify `app.py` was indeed copied to the docker image:

```
$ docker run vitorenesduarte/tutorial:hello ls | grep app
app.py
```

Let's run it again:

```
docker run vitorenesduarte/tutorial:hello
hello world!
```

[slide 7]

[slide 8]

Create file `docker-compose.yml` with:

```
version: "3"
services:
  app:
    build: .
```

Now build and run with:

```
$ docker-compose up --build
```

Change `docker-compose.yml` to:

```
version: "3"
services:
  app:
    build: .
  sleeper:
    build: .
    command: sleep 10
```

And again, in another terminal:

```
$ docker ps
```