# **Τι είναι το Docker;**

Το docker είναι ένα εργαλείο το οποίο μας επιτρέπει να τρέχουμε εφαρμογές σε ένα απομονωμένο περιβάλλον. Δημιουργούμε ένα περιβάλλον ώστε να μπορούμε να δουλέψουμε. Dockerfile, Docker image, Docker Hub, Docker Container.

>docker

>docker run -d 80:80 nginx (ο nginx είναι ένας webserver, -d daemon, στο background – πρέπει να κάνουμε publish ορισμένες θύρες ώστε να μπορέσουμε να χρησιμοποιήσουμε το nginx του container [-p]) [80:80 -- κάνε publish την θύρα του τοπικού μου μηχανήματος:στην θύρα του server ]

>**docker run -d -p 80:80 nginx**

>**docker ps** (βλέπουμε τα ενεργά containers)

>**docker stop <container\_id>**

# **Hub Docker;**

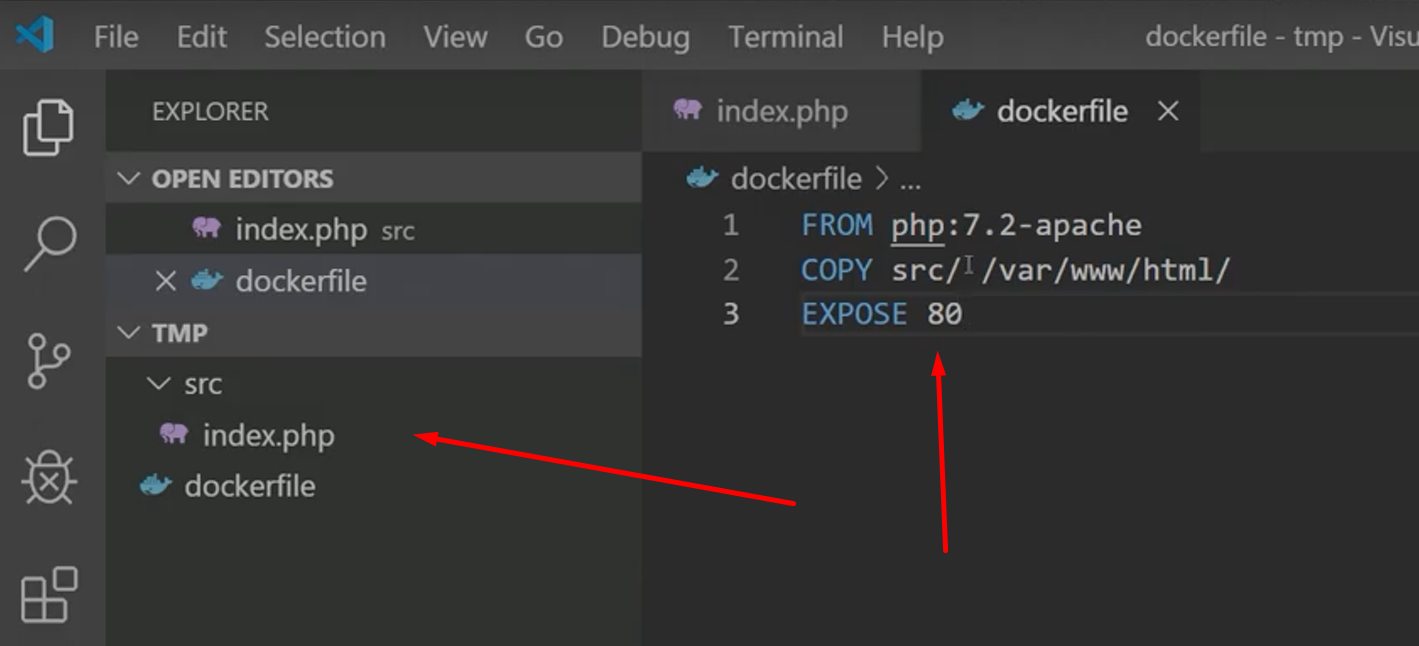
Πως μπορούμε να φτιάξουμε τα images που κατεβάζουμε, να τα παραμετροποιήσουμε χρησιμοποιώντας το docker file.

<https://hub.docker.com/search?image_filter=official&q=&type=image>

Το image που κατεβάζει το docker το βρίσκει στο repository του. Repository είναι οποιοσδήποτε server που φιλοξενεί docker images. Μπορούμε να δημιουργήσουμε και εμείς δικά μας images και να τα ανεβάσουμε στο hub.docker.com.

Δημιουργώ ένα php αρχείο 🡪 <?php echo “Hello World”;

Ψάχνουμε στο docker hub για το image που με ενδιαφέρει, search-for=php, και βλέπουμε το image που μας δινει η php.



>Για να τραβήξω το image της php και να το βάλω στο σύστημα μου και με τις παραμέτρους που θέλω γράφω: **docker build -t myphpapache .** Τρέχουμε την εντολή docker build ώστε να ψάξει και να εκτελέσει το dockerfile μας. (βάζουμε ένα tag -t για να ονομάσουμε το image που θα φτιαχτεί και μετά θα πρέπει να ορίσουμε το path του dockerfile ή αν είναι στον τρέχον φάκελο γράφουμε .)

>docker images

**Οποιαδήποτε αλλαγή και αν κάνουμε στο dockerfile θα πρέπει να κάνουμε build.**

>docker run -d -p 80:80 myphpapache (τρέχω το image που θέλω και συγκεκριμένα τώρα θα τρέξουμε το image με όνομα myphpapache)

>docker stop <container\_id>

**Τι είναι όμως ένα image;**

# **Εξερεύνηση εντολών, containers**

>docker ps (βλέπουμε τα containers που τρέχουν)

>docker ps -a (που τρέχουν ή που είναι σταματημένα)

>docker run -d -p 8000:80 nginx

>docker run -d -p 8001:80 httpd

>docker stop <container\_id>

>docker ps -a (μας δείχνει τα containers που δεν τρέχουν ή που τρέχουν)

>docker rm <container\_id>

> docker run -d --rm -p 80:80 myphpapache

Αν το σταματήσουμε τώρα, τότε θα δούμε ότι έχει διαγραφεί αυτόματα.

>docker run -d –rm –name=myapache -p 8001:80 httpd (δίνουμε δικό μας όνομα στο container)

Χωρίς dockerfile κάνουμε αυτά για να δουλέψουν τα αρχεία μας στον server

>docker run -dit –name my-apache-app -p 8080:80 -v ${PWD}:/usr/local/apache2/htdocs/ httpd:2.4

>docker run -dit –name my-nginx-app -p 8081:80 -v ${PWD}:/usr/share/nginx/html nginx

Χωρίς dockerfile οι αλλαγές περνάνε απευθείας, δεν χρειάζεται να κάνουμε build. Αυτό κερδίζουμε με τα volumes.

**Υπάρχει διαφορά μεταξύ images και containers ή είναι το ίδιο πράγμα;**

# **Συνδέοντας containers. Wordpress, mysql, phpmyadmin**

>docker pull run (συνδέεται με το docker hub και κατεβάζει το image του ubuntu)

>docker images

>docker run -ti ubuntu bash (terminal interactive)

>docker exec -ti <container\_id> bash

>docker –help

>docker run –help

Για να χρησιμοποιήσω ένα container πρέπει να τρέξω την εντολή docker start <container\_id>

>docker run -r MYVAR=testing -ti ubuntu bash (δίνω τιμές σε παραμέτρους, ακολουθεί καλύτερο παράδειγμα)

>docker run –name mysupersql -e MYSQL\_ROOT\_PASSWORD=root -d mysql:5.7

>docker run –name mywordpress -p 8080:80 –link mysupersql:mysql -d wordpress

>docker run –name myadmin -d –link mysupersql:db -p 8081:80 phpmyadmin/phpmyadmin

**Ποια είναι η διαφορά του -d (detached, run at background) και του -ti?**

# **Dockerfile, Jupyter**

Dockerfile:

FROM ubuntu

EXPOSE 8888

RUN apt-get update -y;

RUN apt-get install -y python3 python3-pip;

RUN pip3 install –upgrade pip

RUN pip3 install jupyterlab;

WORKDIR /src

CMD jupyter-lab –ip=0.0.0.0 –allow-root –NotebookApp.token=’ ’ (μονο τοπικά όμως το NotebookApp.token)

>docker build -t myjupyter .

>docker run -v ${PWD}:/src -p 8888:8888 -d myjupyter

# **Docker Compose**

Docker-compose.yml:

Version: ’3’

Services:

Mysql:

Image: mysql:5.7.20

Environment:

* “MYSQL\_USER=root”
* “MYSQL\_ROOT\_PASSWORD=root”

Ports:

* 3306:3306

Volumes:

* ./mysqldata:/var/lib/mysql

PhpMyAdmin:

Image: phpMyAdmin/phmyadmin

Depends\_on:

* mysql

ports:

- “8090:80”

Links:

- mysql:db

Wordpress:

Image: wordpress

Depends\_on:

- mysql

Ports:

* 8080:80

Environment:

WORDPRESS\_DB\_HOST: mysql

WORDPRESS\_DB\_USER: root

WORDPRESS\_DB\_PASSWORD: root

WORDPRESS\_DB\_NAME: mywordpress

Jupyter-lab:

Build:

Context: ./

Dockerfile: jupyter.dockerfile

Volumes:

* ./:/lab

Working\_dir: /lab

Ports:

* “8888:8888”

Command: jupyter-lab –ip=0.0.0.0 –allow-root –NotebookApp.token =’ ’

>docker-compose up

>docker-compose up -d

>docker-compose logs

>docker-compose stop

>docker-compose up –build -d (ξαναφτιάχνει τα containers από την αρχή)

# Docker Tutorial for Beginners [FULL COURSE in 3 Hours]

What is a container?

A way to package application with all the necessary dependencies and configuration.

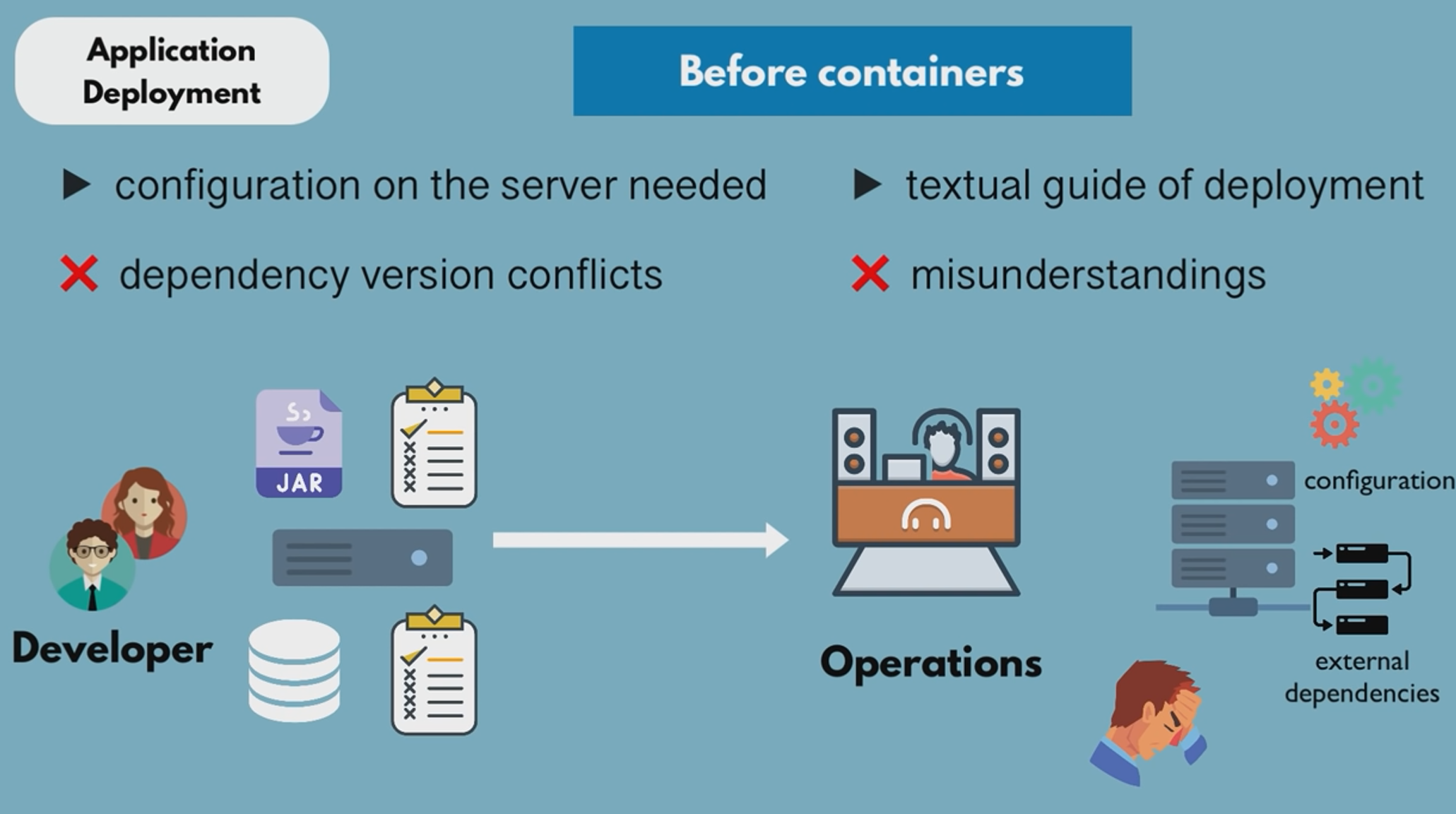
We can find the containers in the Docker Hub. 140.000 container images.

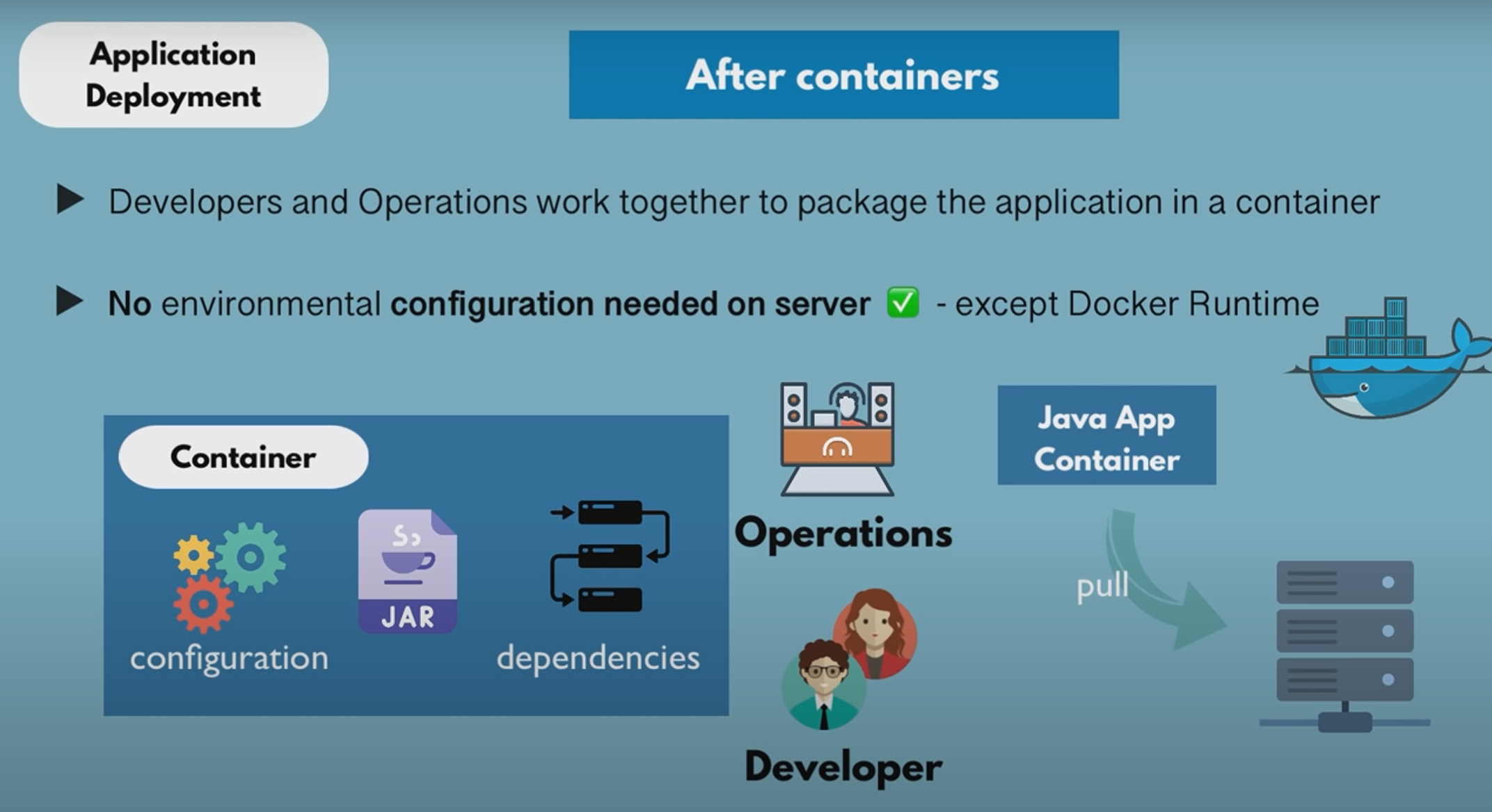
**Development process Before containers and after**

Before containers the installation process of the apps was different for each OS environment and it had many steps where something could go wrong.

With containers we don’t have to install all these programs we can just download a container which is an isolated environment, packaged with all needed configuration, one command to install the app. Run same app with 2 different versions

**Deployment process before and after containers**





Docker run -it centos bash

Docker run -d centos sleep 2000

Docker rmi ubuntu //removes images

docker run redis:4.0 //using a tag to specify the version that we want to download

docker exec -it <container\_id> sh //execution of commands inside a running container

docker logs -f --details --timestamps <container\_id> //we can see the logs that

docker ps --filter “name=nostalgic\_stallman”

docker rm $(docker ps --filter status=exited -q) //delete multiple containers in one line - the -q shows us only the container id

docker stop $(docker ps -q)

docker rmi $(docker images)

docker search ubuntu //we can search for an image through terminal just like the search of docker hub

docker search --limit=3 ubuntu

docker search --format “{{.Name}}\t{{/IsAutomated}}\t{{IsOfficial}}”

docker rename <old\_container\_name> <new\_container\_name> //rename a container

docker info //gives information of the system that I’m using

docker save redis:latest | gzip > myredis.tar.gz //compress a container image in a file

docker load < /Users/joan\_amengual/Desktop/my\_image/myredis.tar.gz //load the compressed docker image

docker stats //visualize the cpu usage that a container has etc.