Introduction to Docker and Ansible

Alfonso Pérez, Germán Moltó, Marica Antonacci

Universitat Politècnica de Valènica , INFN-Bari

INDIGO-DataCloud is co-founded by the Horizon 2020Framework Programme

Index

- **Docker**
- Ansible

INDIGO - DataCloud

Objectives

INDIGO - DataCloud

- Learn the importance of Docker as a container application encapsulation tool
- Understand the differences, advantages and disadvantages between Virtual Machines (VMs) and Containers
- Understand the principal concepts related with Docker

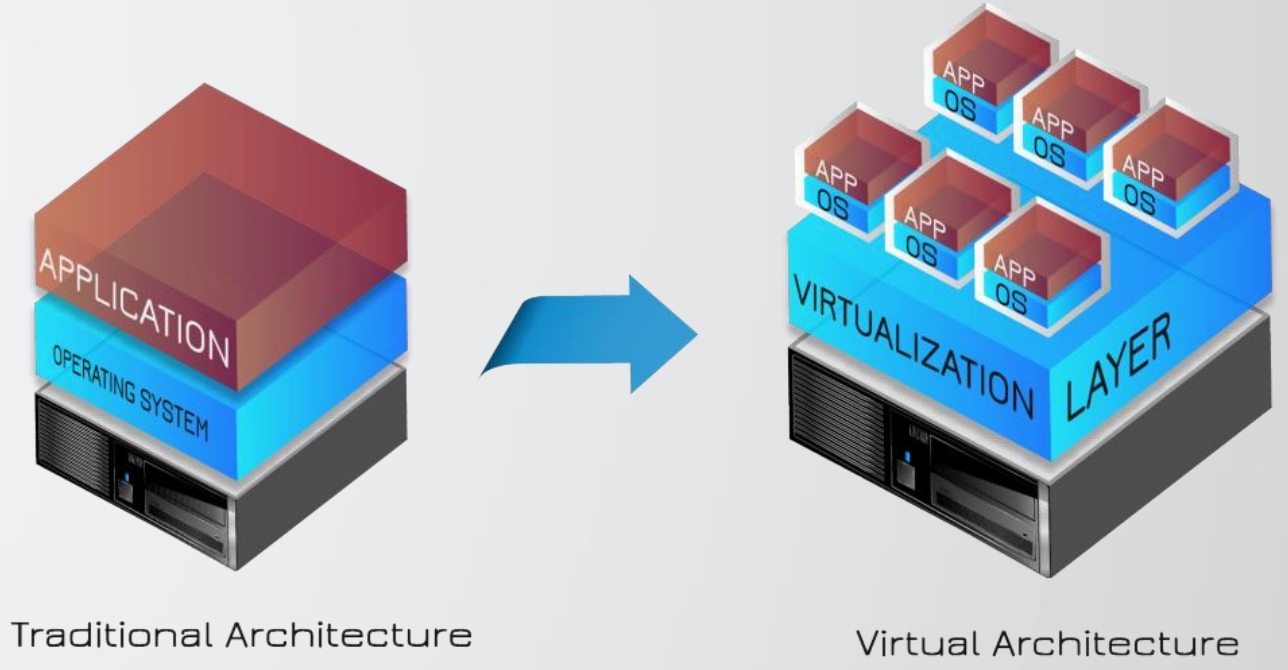
Some context: Distributed applications

- Distributed applications need:
  - Computation
  - Data
  - Network
- Hundreds of tools, programming languages technologies and platforms
- Different mechanisms to deliver software
  - RPM and DEB packages, JAR libraries, Homebrew, NPM, etc.
  - Incompatibilities

Old School	Trendy
On-premises	Cloud computing
Monolithic	Microservices
XML + WSDL	JSON + REST
Long Development Cycles	Continuous Delivery

About virtualization

- Virtualization has introduced several advantages in the last years with the popularization of hypervisors such as KVM, XEN or VMWare.
- Application encapsulation including all its dependencies.
- Cloud computing.



Traditional Architecture

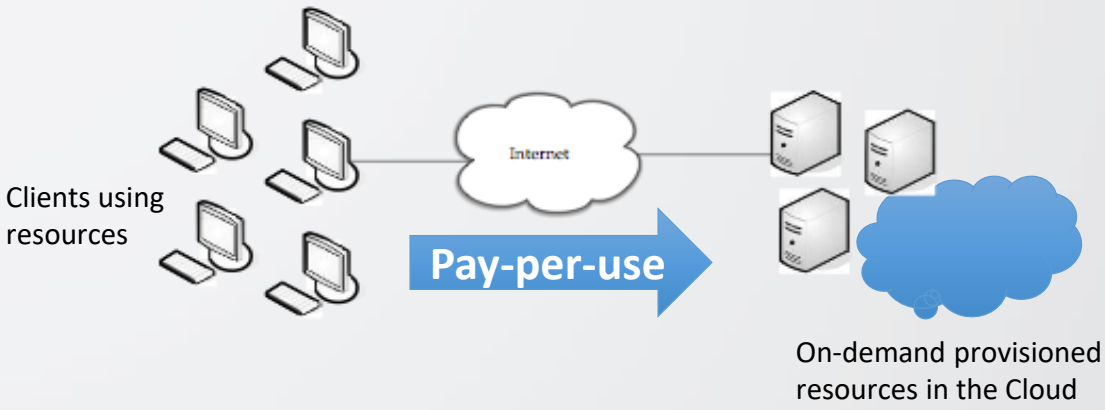
Virtual Architecture

Image: http://exelos.com/solutions/virtualization/

What is Cloud Computing?

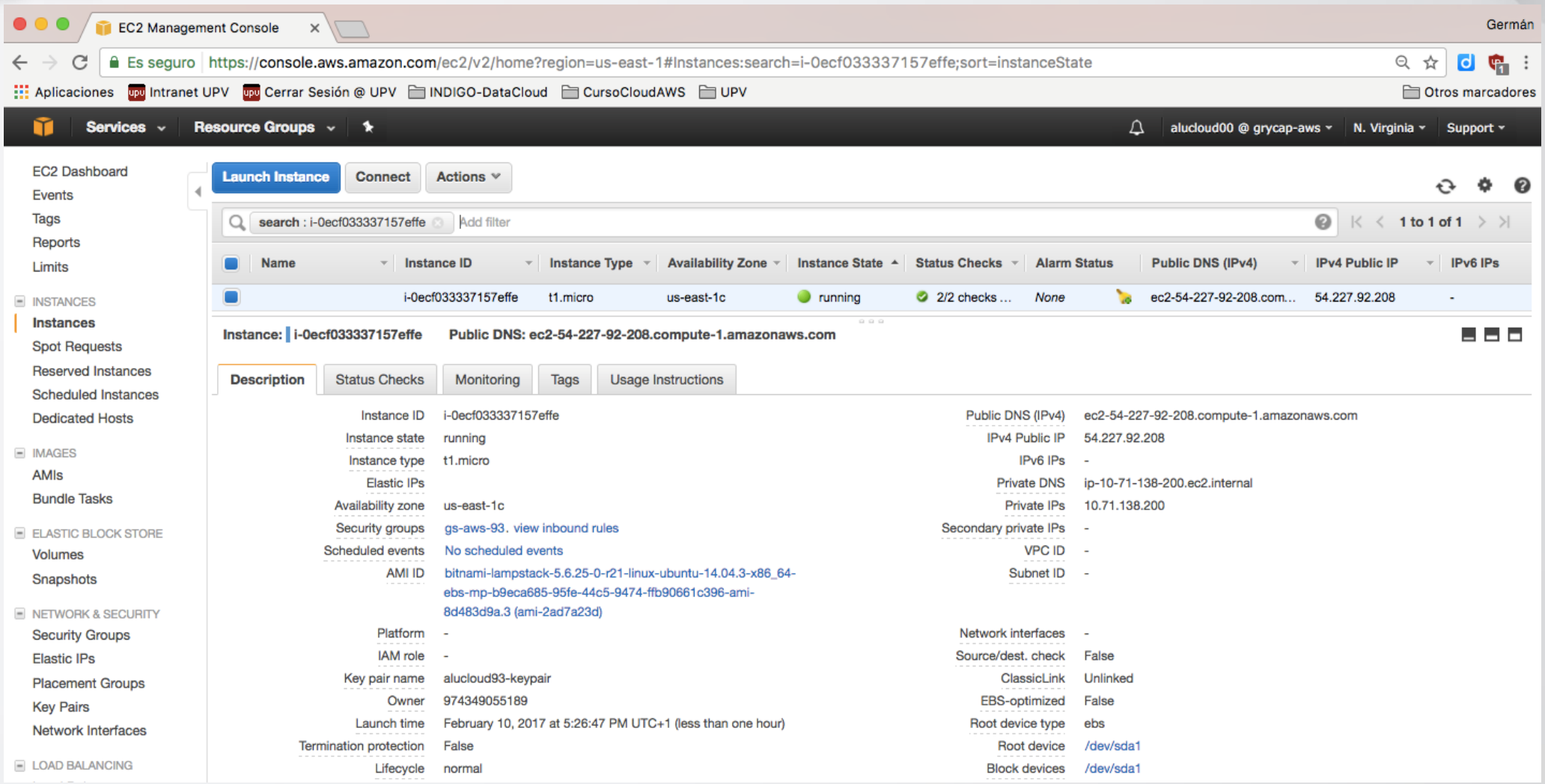
INDIGO - DataCloud

- Cloud computing is a paradigm that allows to offer services (computation, storage, etc.) through internet
  - Computing / storage / networking services offered by a big provider to the clients
  - Harnessing economies of scale to offer lowers costs to the final users
  - Pay-per-use, without initial investment



Virtual Machines

INDIGO - DataCloud

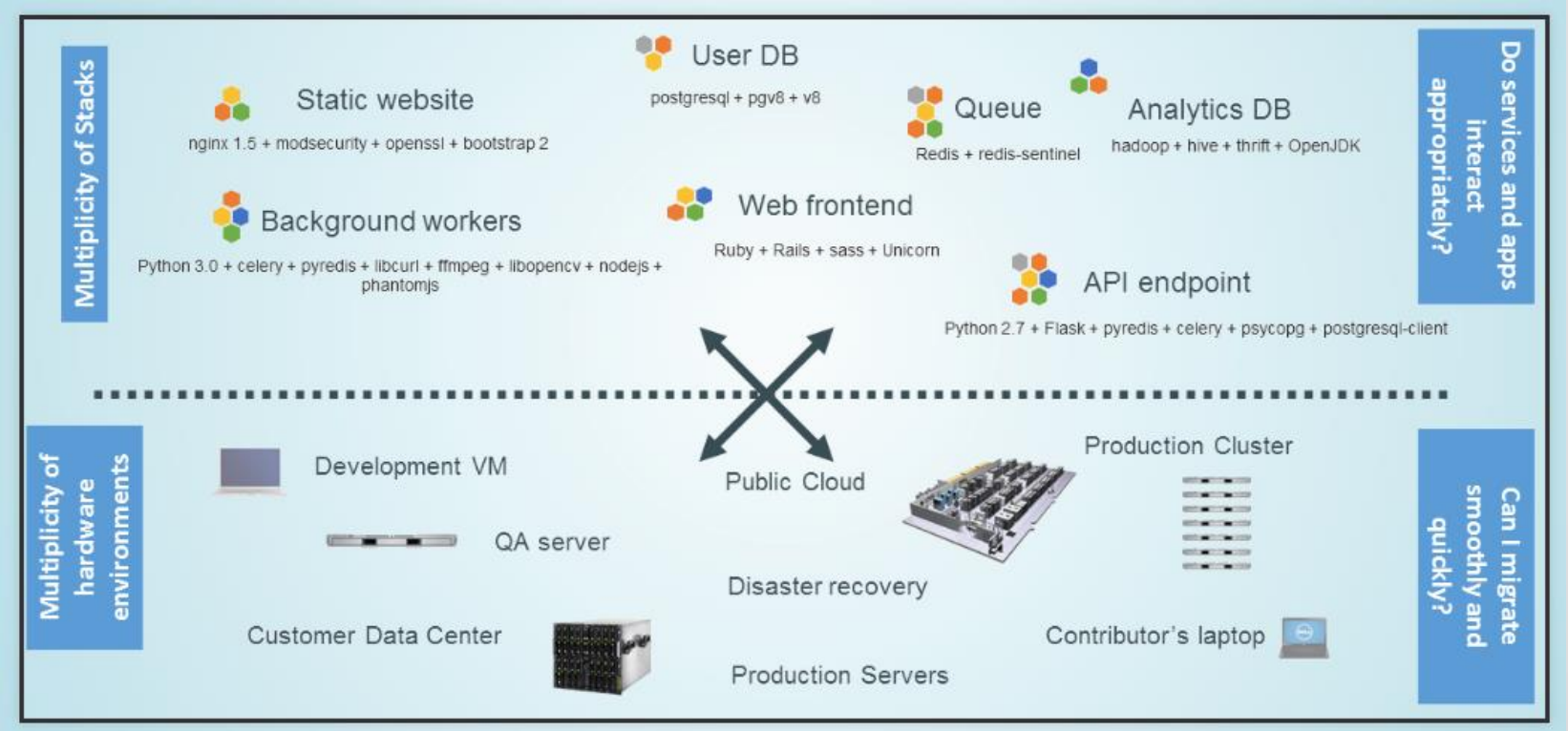


About virtual machines

- Host completely encapsulated (OS + Apps)
- Require several minutes to load
  - http://ieeexplore.ieee.org/document/6253534/
- VMs images could take several GBs of space
  - A change inside the application requires the creation of a new VM
- Images are not portable between hypervisors
  - Raw vs qcow2, ide dispositives, qemu-img tool

Problems

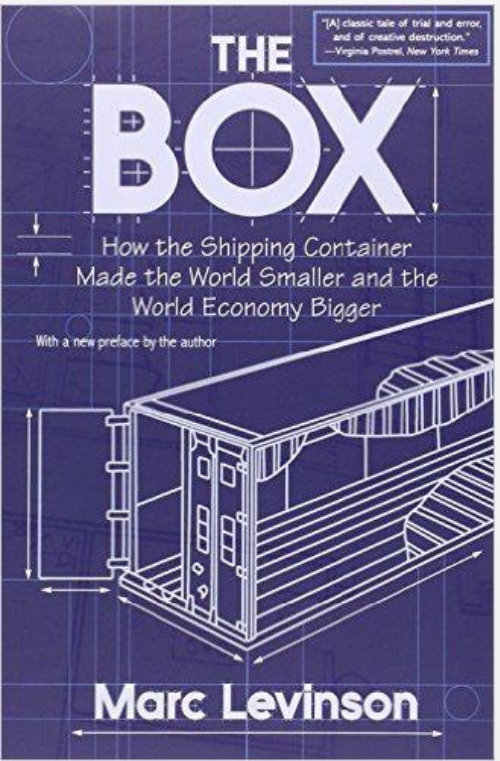
- Develop distributed applications usually require different OSs, programming languages, execution environments, libraries, etc. and can be deployed under different platforms.



Real world example

http://disney.github.io/docker-training

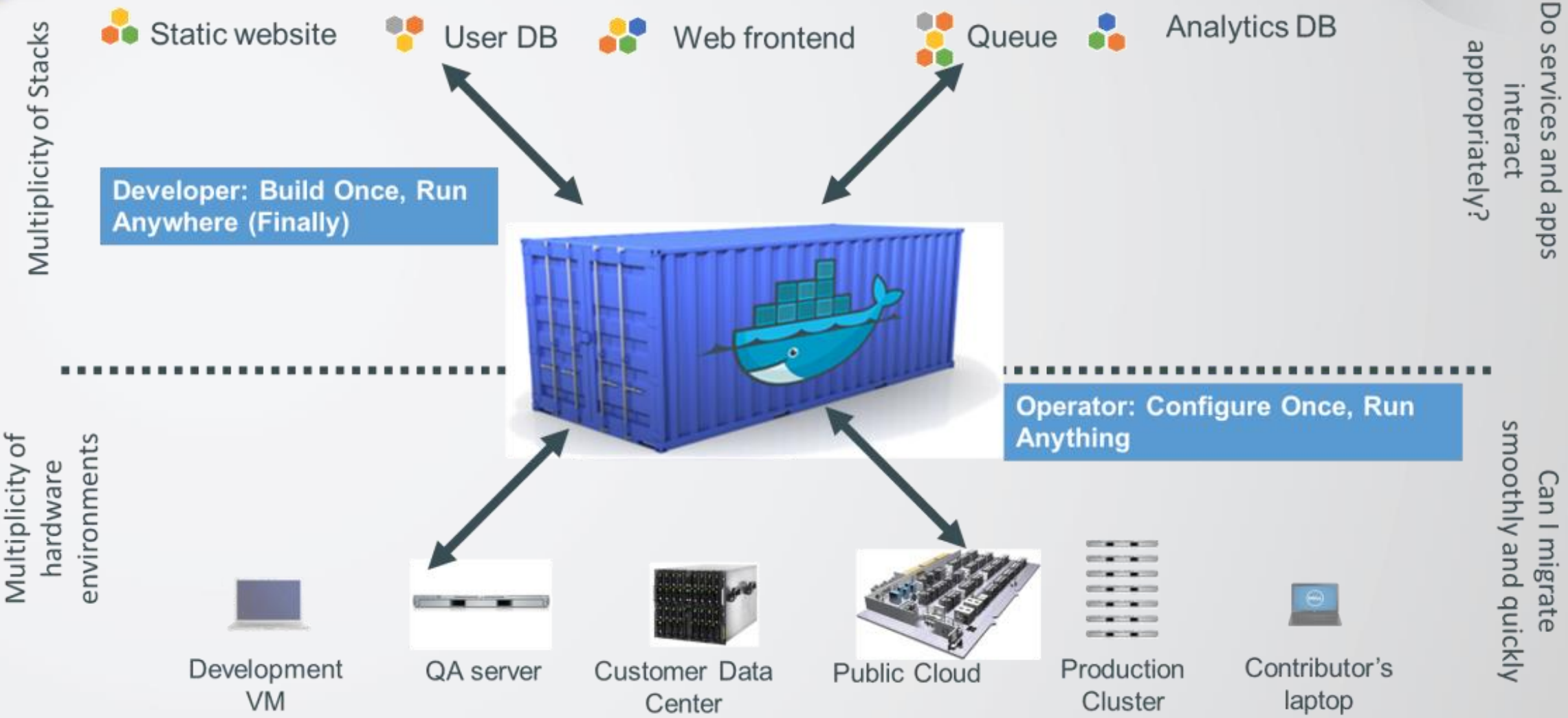
Real world solution



http://disney.github.io/docker-training

http://www.amazon.com/The-Box-Shipping-Container-Smaller/dp/0691136408

Docker containers



What is docker?

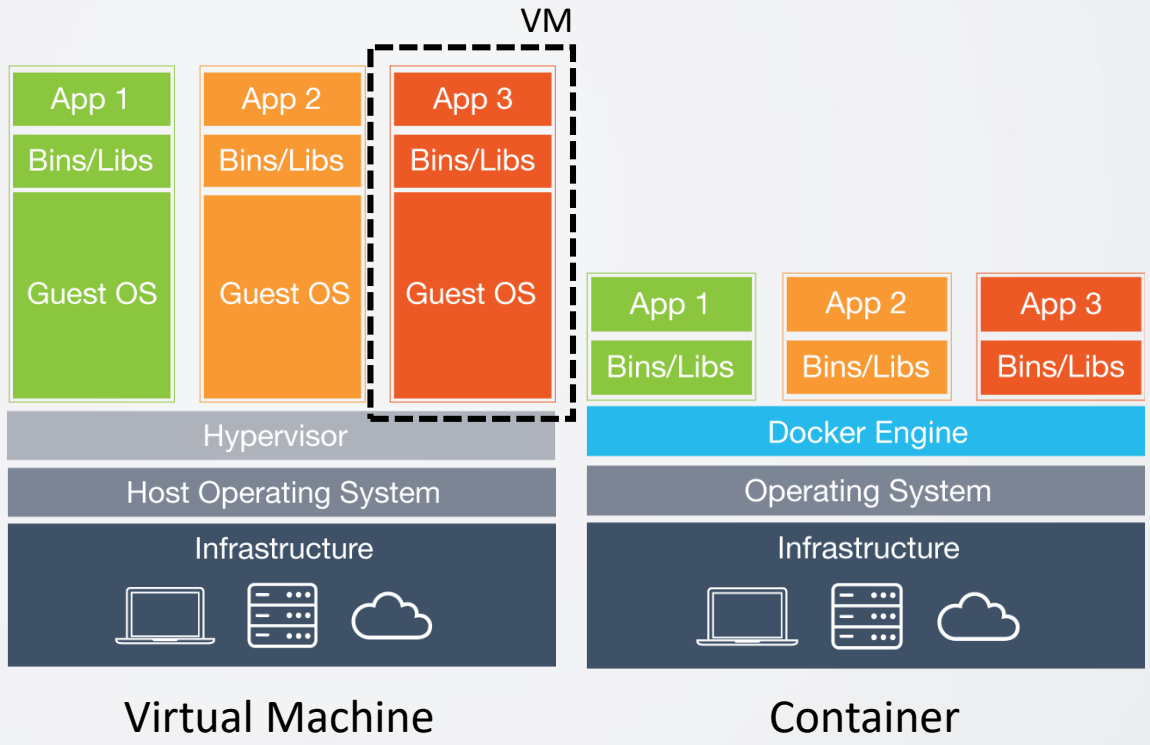
INDIGO - DataCloud

- Docker is a platform aimed to developers and system administrators that permits to build, send and execute distributed applications.

- Allows to encapsulate and application with all its dependencies to be posteriorly executed in different platforms
  - Main objective: *Fast, consistent delivery of applications*

- Allows to quickly deploy application execution environments and in a repeatable manner

Virtual Machines VS Containers



Virtual Machine

Container

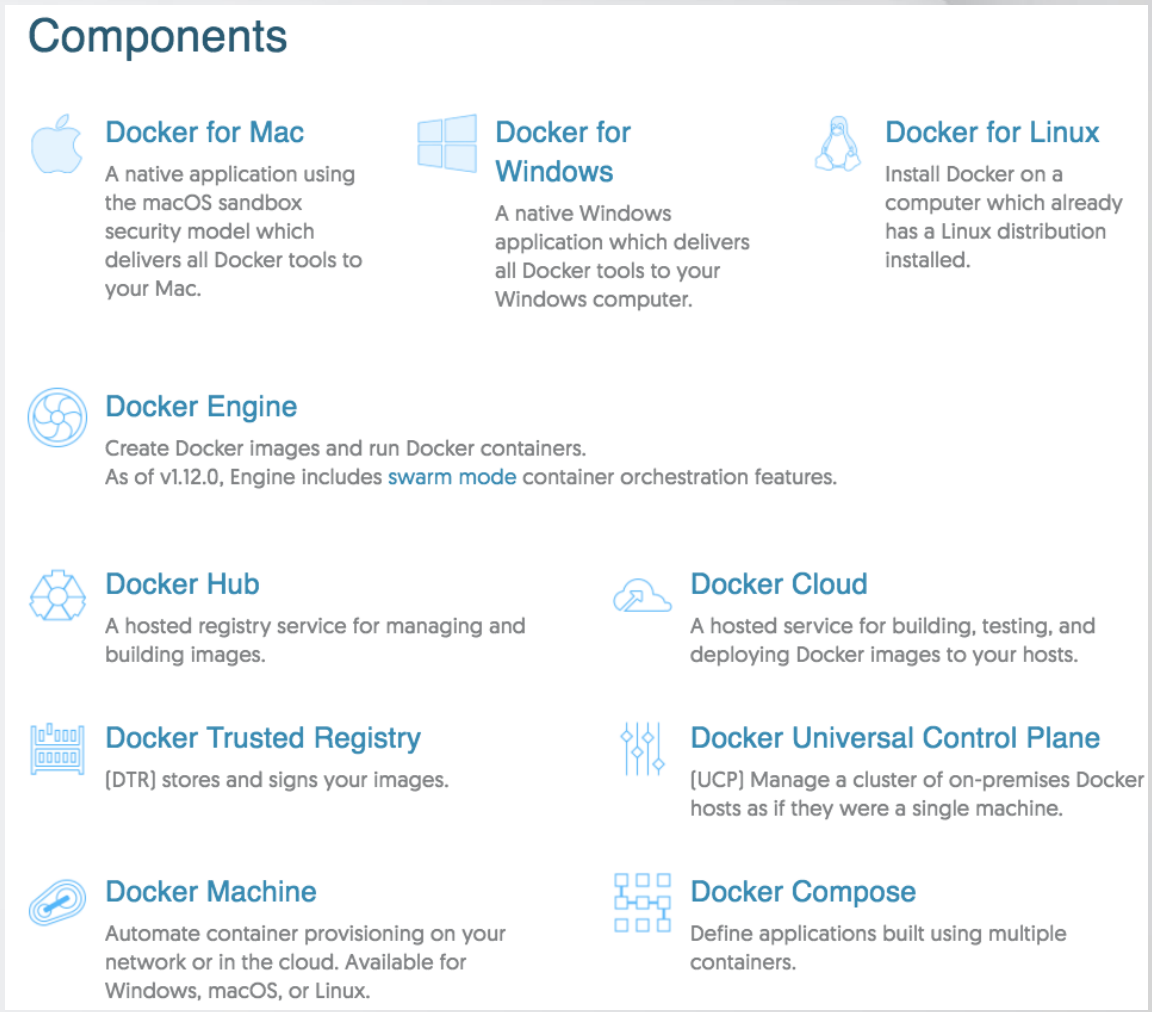
- Containers (Pros)
  - Less image weight
  - Instant execution
  - No virtualization overload
  - All application dependencies encapsulated
  - Write Once Run Anywhere*
- Containers (Cons)
  - Impossible to execute Windows on Linux
  - Security isolation
    - Host kernel sharing

*x86 with Linux 3.2+ or 2.6.32+ for Fedora, CentOS, etc.

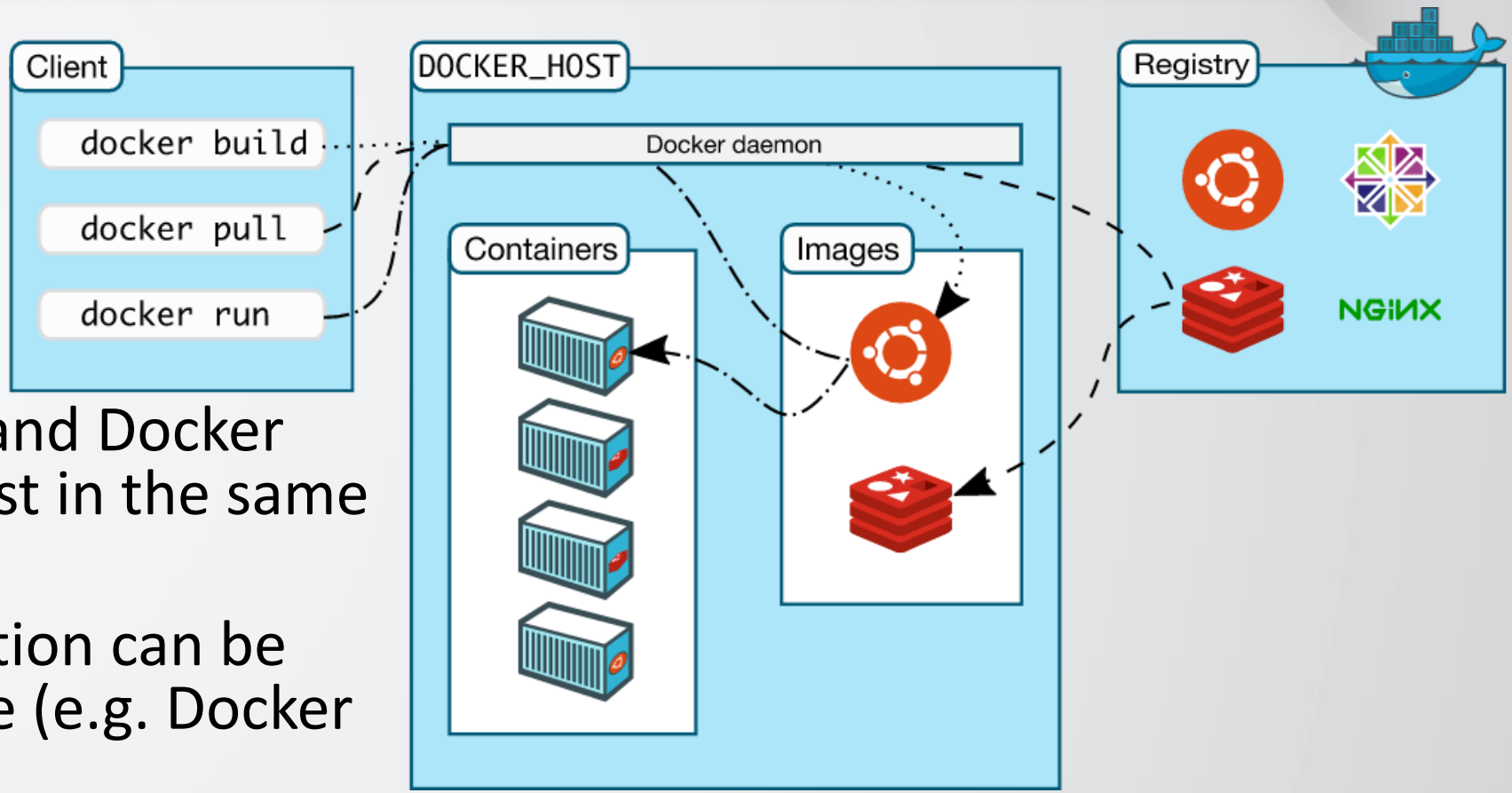
Docker components

- Docker is comprised by a variety of tools around the Docker engine

- OCI (Open Container Initiative)
  - https://www.opencontainers.org/



Docker Engine architecture



- Docker Client and Docker Host can coexist in the same machine
- Image registration can be local or remote (e.g. Docker Hub)

Docker Engine technology

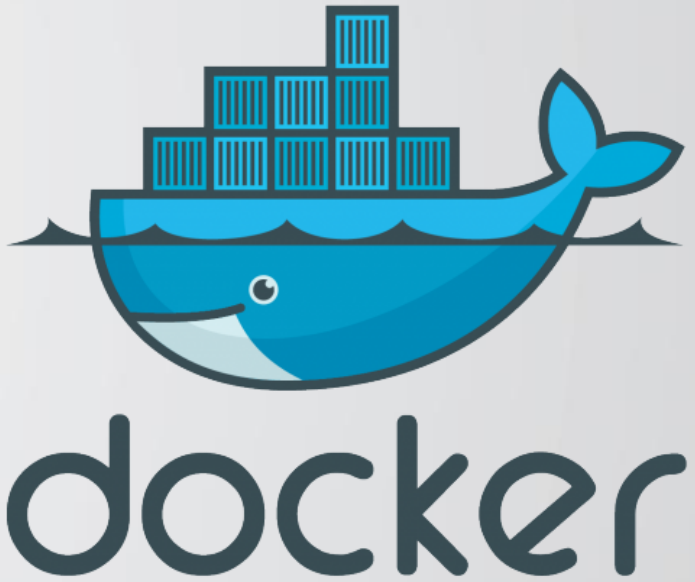
- Namespaces
  - Isolation layer
  - Different namespaces are created for an running container (pid, net, ipc, mnt, uts)
  - Prevents a running process in the container from seeing other process running in the host
  - Blocks the access to other devices of the host
- Cgroups
  - Limits the resource usage of the container
- UnionFS
  - File system based in layers that allows to storage the container’s modifications in different layers

Docker Engine installation

INDIGO - DataCloud

```
curl –fsSL https://get.docker.com/ | sh
```

- There are other ways to install it:
  - https://docs.docker.com/engine/installation/



Play with Docker

- http://play-with-docker.com
- Allows to create instances with Docker installed for four hours

Docker Playground

play-with-docker.com/p/fcaa87c2-9f5f-472d-859c-19593ae9e8db#fcaa87c2_node1

03:54:44

CLOSE SESSION

Instances

+ ADD NEW INSTANCE

fcaa87c2_node1

IP

10.0.43.3

Memory

2.49% (102.1MiB / 3.996GiB)

CPU

1.21%

DELETE

```
[node1] (local) root@10.0.43.3 ~
$ echo "bye" > hello.txt && docker run --rm -v `pwd`/hello.txt:/tmp/hello.txt alpine cat /tmp/hello.txt
bye
[node1] (local) root@10.0.43.3 ~
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
 NAMES
[node1] (local) root@10.0.43.3 ~
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine              latest              4a415e366388        11 days ago         3.99 MB
[node1] (local) root@10.0.43.3 ~
$
```

Basic Docker Engine concepts

- **Image**
  - Contains the SO distribution (e.g. Ubuntu 16.04) and a specific configuration of packages / applications / data determined by the image creator
- **Docker Hub**
  - Image repository and catalogue
  - Accessible using a CLI, a web interface and a REST API
- **Container**
  - Instance of an specific image running as an isolated process in an specific machine (Docker Host)
- **Docker Host**
  - Machine with Docker Engine installed in charge of executing the containers
- **Docker client**
  - Machine from where the Docker container deployment is requested (can be the same as the Docker Host).
  - Client tool used to interact with the Docker engine.

Docker Engine Workflow

INDIGO - DataCloud

- The users use ***Docker Client*** to deploy containers in a ***Docker Host*** from images previously stored in ***Docker Hub***. These images can be modified and stored again both in ***Docker Hub*** or a ***Docker Private Registry***

  - Usually there are multiple containers running at the same time in the same Docker Host

  - Containers share the kernel host

  - Different applications can be connected to the same port in different containers.

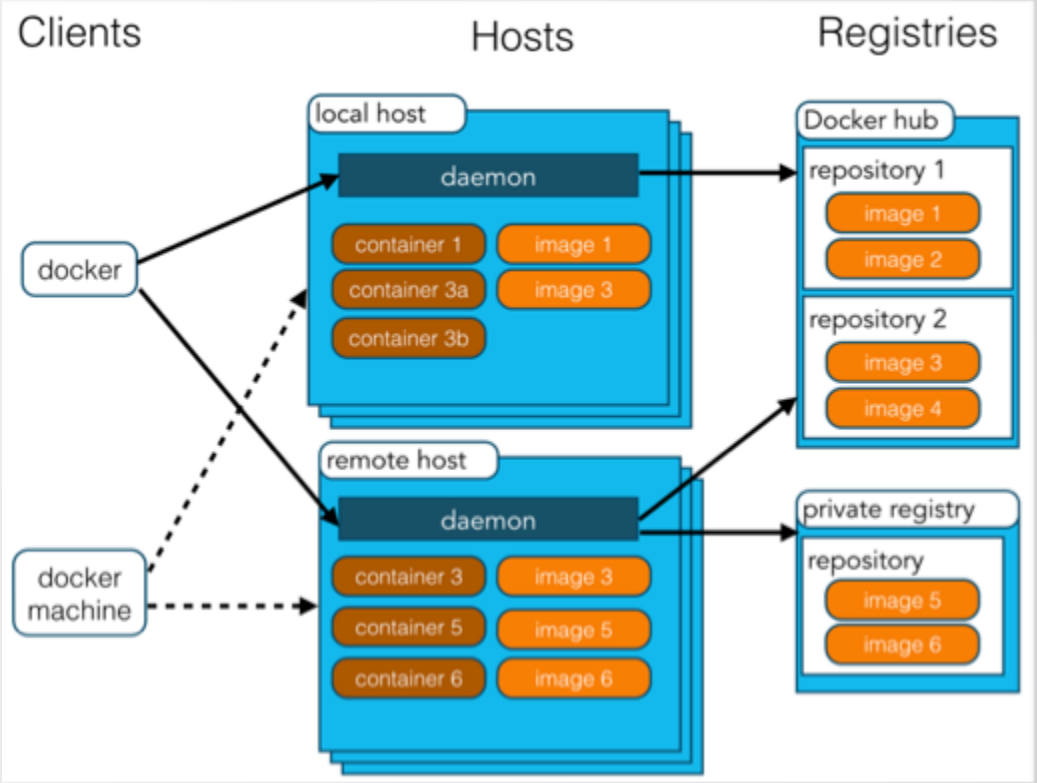
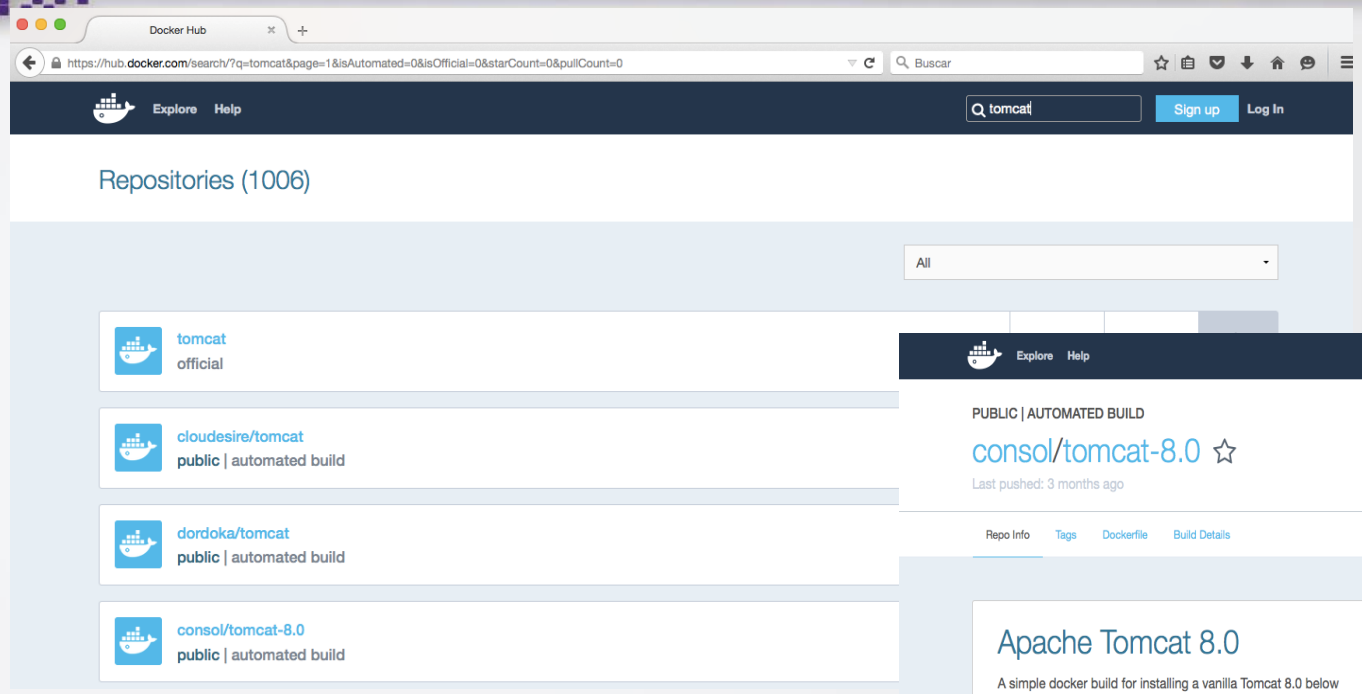


Image from: https://dzone.com/refcardz/getting-started-with-docker-1

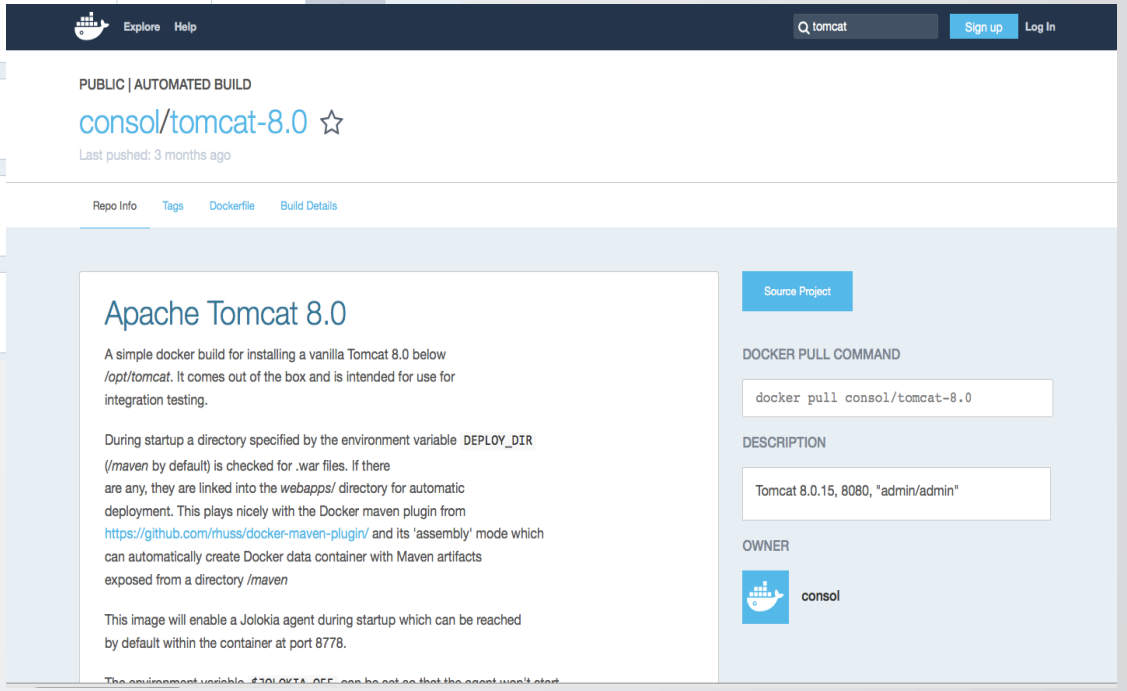
What can you do with Docker Engine?

- Manage the container lifecycle
  - Start, , stop, kill, restart, etc.
- Manage the container images
  - push, pull, tag, rmi, etc.
- Inspect / access the container
  - logs, attach
- …
- Where can we found a Docker images catalogue?

Docker Hub



https://hub.docker.com



- Repositories with images of Docker containers
- *Automated Builds* from GitHub

Demo

INDIGO - DataCloud

a.k.a A DEMO IS IS WORTH A THOUSAND WORDS

Docker 101: Containers (1)

INDIGO - DataCloud

```
gmolto@felis-2 ~ docker run alpine echo hello world
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
0a8490d0dfd3: Pull complete
Digest: sha256:dfbd4a3a8ebca874ebd2474f044a0b33600d4523d03b0df76e5c5986cb02d7e8
Status: Downloaded newer image for alpine:latest
hello world
```

- Docker downloads automatically the image alpine:latest from Docker Hub
- The image is stored in the local registry of the Docker Host
- Lastly, the container is launched and the command is executed inside it showing the output on the screen

Docker 101: Images (1)

- Docker Images contain the OS + Apps
- The images can be tagged and stored in different Docker Registries
  - https://docs.docker.com/registry/deploying/

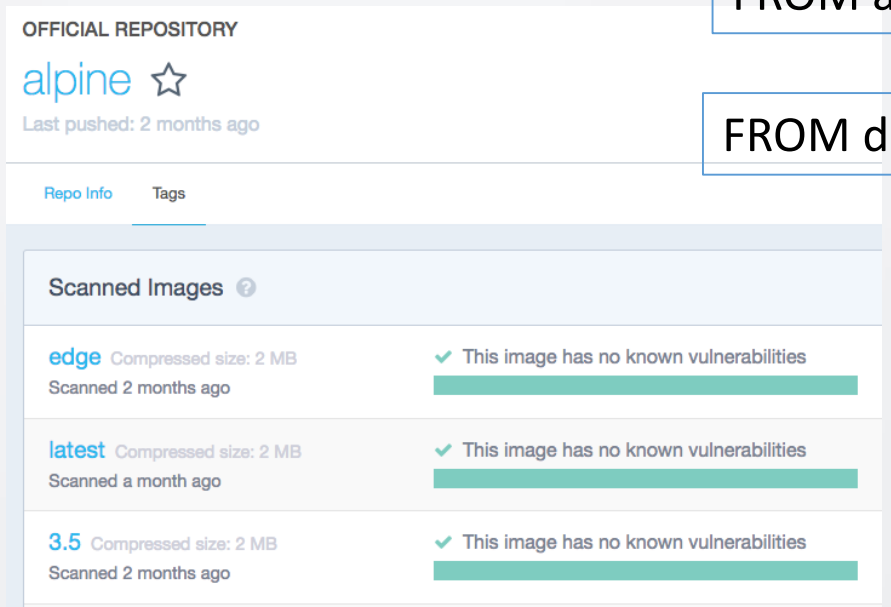
```
gmolto@felis-2 ~ docker images
REPOSITORY                      TAG                 IMAGE ID            CREATED             SIZE
redis                           latest              74d8f543ac97        9 days ago          184 MB
jjmerelo/docker-daleksay        latest              5bf18c53ecd5        3 weeks ago         72.1 MB
starefossen/node-imagemagick    latest              fd39b463447c        3 weeks ago         675 MB
busybox                         latest              7968321274dc        3 weeks ago         1.11 MB
alpine                          latest              88e169ea8f46        6 weeks ago         3.98 MB
examplevotingapp_result         latest              4b1b9a9aa48e        8 weeks ago         227 MB
examplevotingapp_worker         latest              a9bb84ce3459        8 weeks ago         574 MB
examplevotingapp_vote           latest              607747fc0e0c        8 weeks ago         84.3 MB
postgres                        9.4                 452864725827        8 weeks ago         265 MB
grycap/odisea                   latest              5e795229a921        2 months ago        560 MB
redis                           alpine              9947c5a33865        2 months ago        21 MB
python                          2.7-alpine          9c8c07c0f9b7        2 months ago        72.2 MB
microsoft/dotnet                1.0.0-preview2-sdk  6704971aa9c1        3 months ago        537 MB
jpetazzo/trainingwheels         latest              db38019622f1        8 months ago        686 MB
node                            5.11.0-slim         cb888ea932ad        9 months ago        207 MB
gmolto@felis-2 ~
```

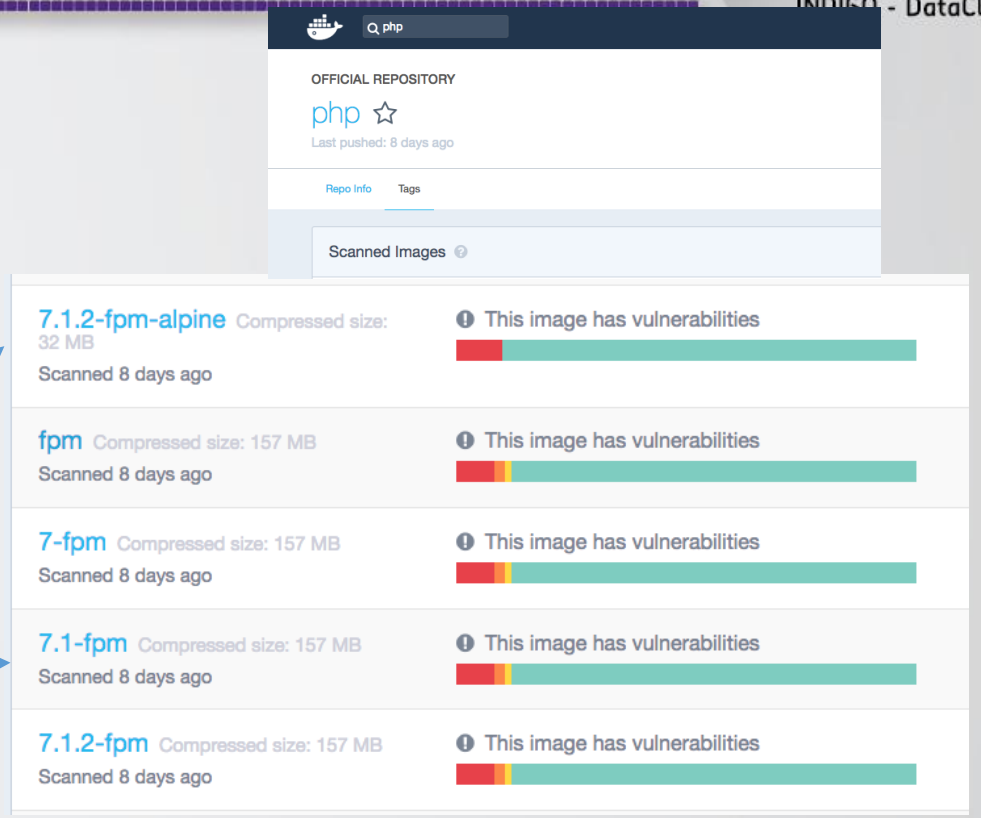
Docker 101: Images (2)

- Regarding images, size **DOES** matter
  - Using a base OS as Alpine can reduce the image to a quarter of its size (compared to other OSs such as Ubuntu, CentOS, etc.)

FROM alpine:3.4

FROM debian:jessie



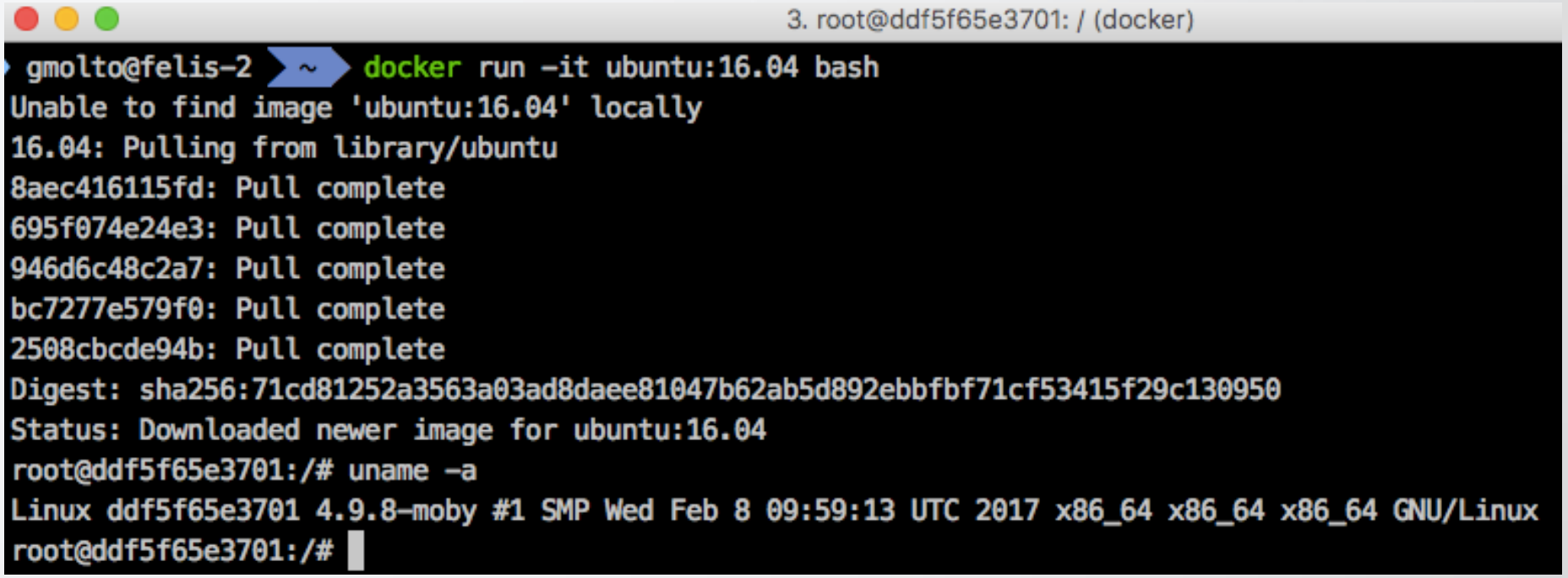


https://hub.docker.com/r/_/alpine/
https://alpinelinux.org/

Docker 101: Containers (2)

INDIGO - DataCloud

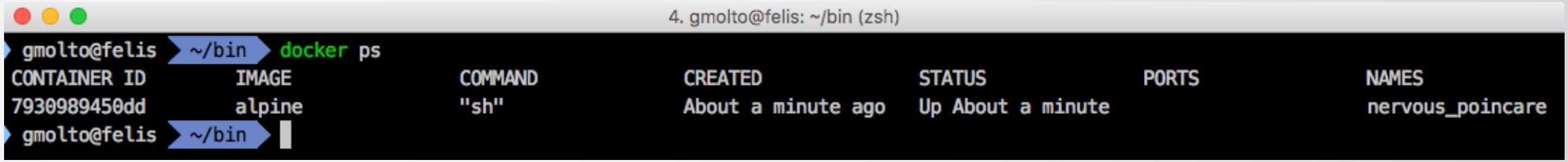
- Interactive session with a Docker container
  - docker run -it ubuntu:16.04 /bin/bash
  - The container can be used as any other machine: install applications, internet access, etc



Docker 101: Containers (3)

INDIGO - DataCloud

- Containers receive a name and an ID



- This can be used to obtain information about the container and manage its lifecycle

- A container must be stopped (*stop*) before being removed (*rm*)

Docker 101: Docker Help

docker --help

…

Commands:

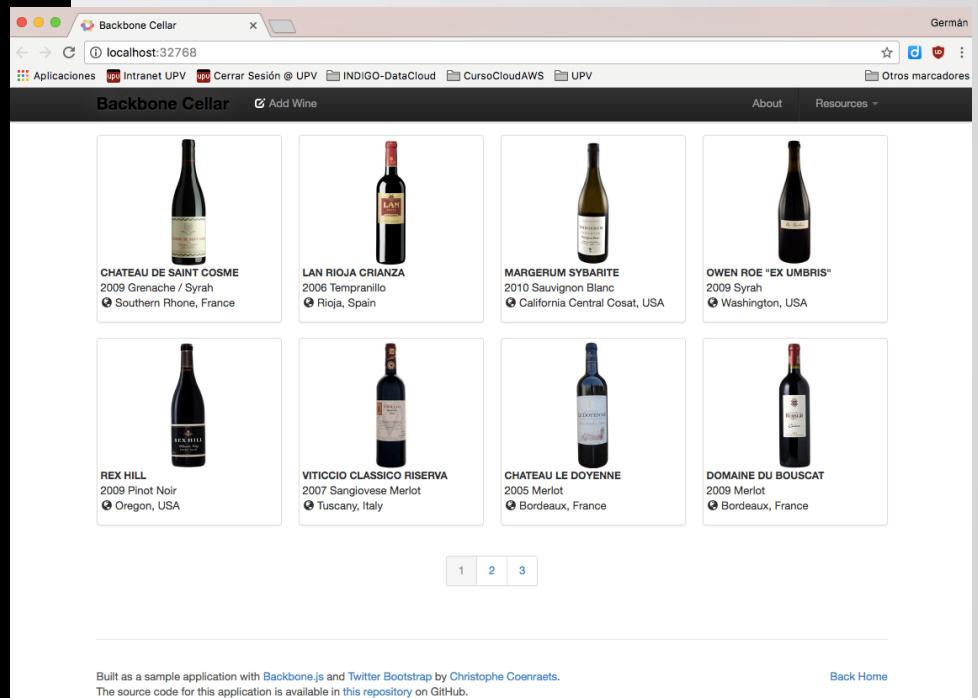
attach Attach to a running container
build Build an image from a Dockerfile
commit Create a new image from a container's changes
cp Copy files/folders between a container and the local filesystem
create Create a new container
deploy Deploy a new stack or update an existing stack
diff Inspect changes on a container's filesystem
events Get real time events from the server
exec Run a command in a running container
export Export a container's filesystem as a tar archive
history Show the history of an image
images List images
import Import the contents from a tarball to create a filesystem image
info Display system-wide information
inspect Return low-level information on Docker objects
kill Kill one or more running containers
load Load an image from a tar archive or STDIN
login Log in to a Docker registry
logout Log out from a Docker registry
logs Fetch the logs of a container
pause Pause all processes within one or more containers
port List port mappings or a specific mapping for the container
ps List containers
pull Pull an image or a repository from a registry
push Push an image or a repository to a registry
rename Rename a container
restart Restart one or more containers
rm Remove one or more containers
rmi Remove one or more images
run Run a command in a new container
save Save one or more images to a tar archive (streamed to STDOUT by default)
search Search the Docker Hub for images
start Start one or more stopped containers
stats Display a live stream of container(s) resource usage statistics
stop Stop one or more running containers
tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top Display the running processes of a container
unpause Unpause all processes within one or more containers
update Update configuration of one or more containers
version Show the Docker version information
wait Block until one or more containers stop, then print their exit codes

Docker 101: Web Application

INDIGO - DataCloud

- A port of an application running inside a container can be mapped to an specific port of the Docker Host.
  - **docker run --rm -d -P --name backbone-cellar-mem cursocloudaws/backbone-cellar-mem**

```
gmolto@felis-2 ~ docker run -d -P --name backbone-cellar-mem cursocloudaws/backbone-cellar-mem
Unable to find image 'cursocloudaws/backbone-cellar-mem:latest' locally
latest: Pulling from cursocloudaws/backbone-cellar-mem
386a066cd84a: Pull complete
269e95c6053a: Pull complete
6243d5c57a34: Pull complete
872f6d38a33b: Pull complete
e5ea5361568c: Pull complete
f81f18e77719: Pull complete
f9dbc878ca0c: Pull complete
195935e4100b: Pull complete
c047d6392f67: Pull complete
6d5afcbf41ee: Pull complete
bbe672c318f3: Pull complete
c015a3b2e201: Pull complete
6eb6d78a72af: Pull complete
8aa3ab12db25: Pull complete
Digest: sha256:afda0cb6b915f557803f4debf79f26afa7511916ec5d89099db48d2e56f054cc
Status: Downloaded newer image for cursocloudaws/backbone-cellar-mem:latest
055970a83ccc920100f84c1977eff04062d2b638c5d0c402605a430e90f14013
gmolto@felis-2 ~ docker port backbone-cellar-mem 80
0.0.0.0:32768
```



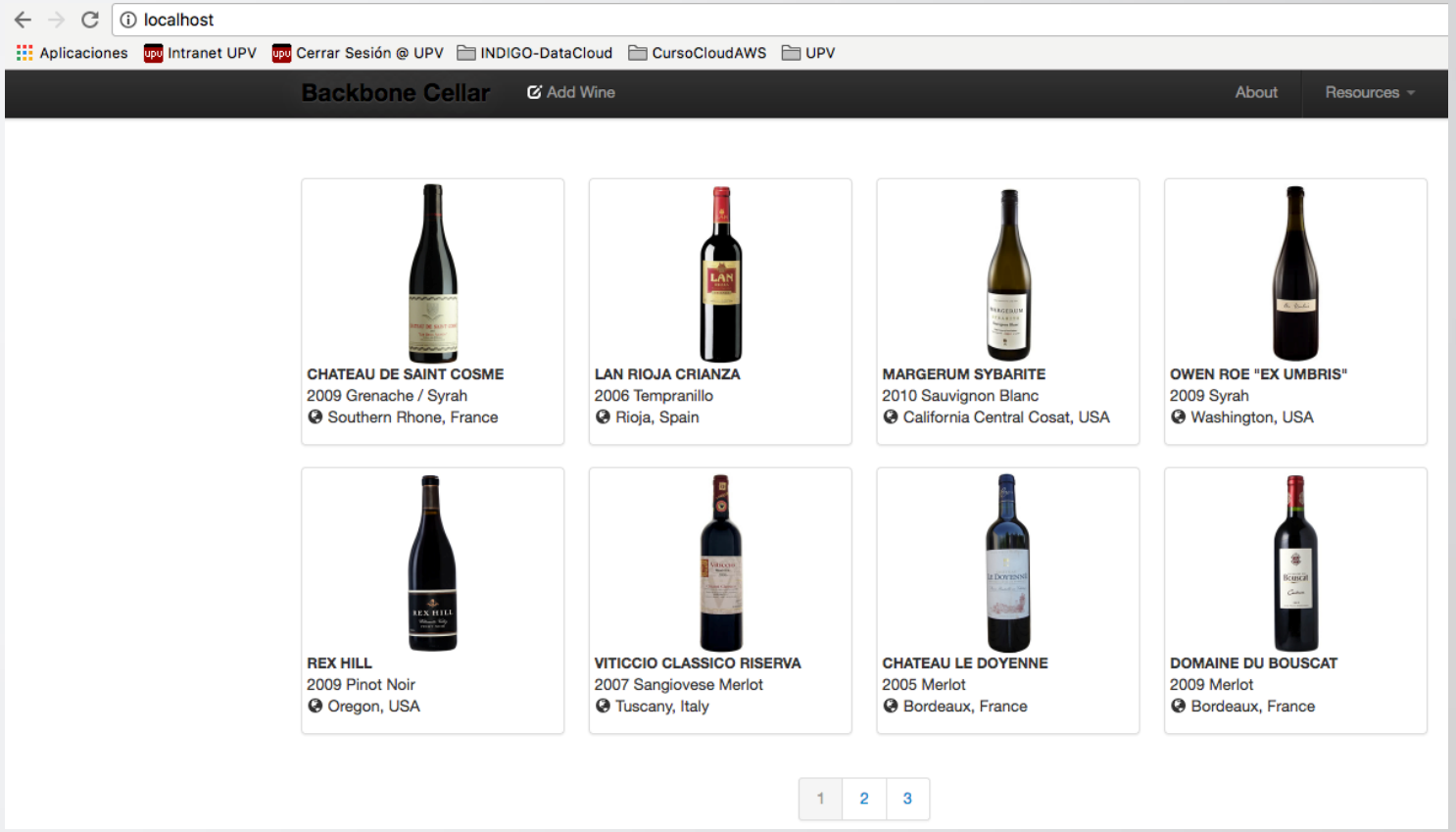
Docker 101: Mounting volumes

- **docker run --rm -d -p 80:80 -v `pwd`/cellar-mem:/var/www/html php:5.6-apache**

- A folder in the Docker Host can be mounted inside the container

- Usefull for having different test environments for a local application

- Any local changes are reflected in the application



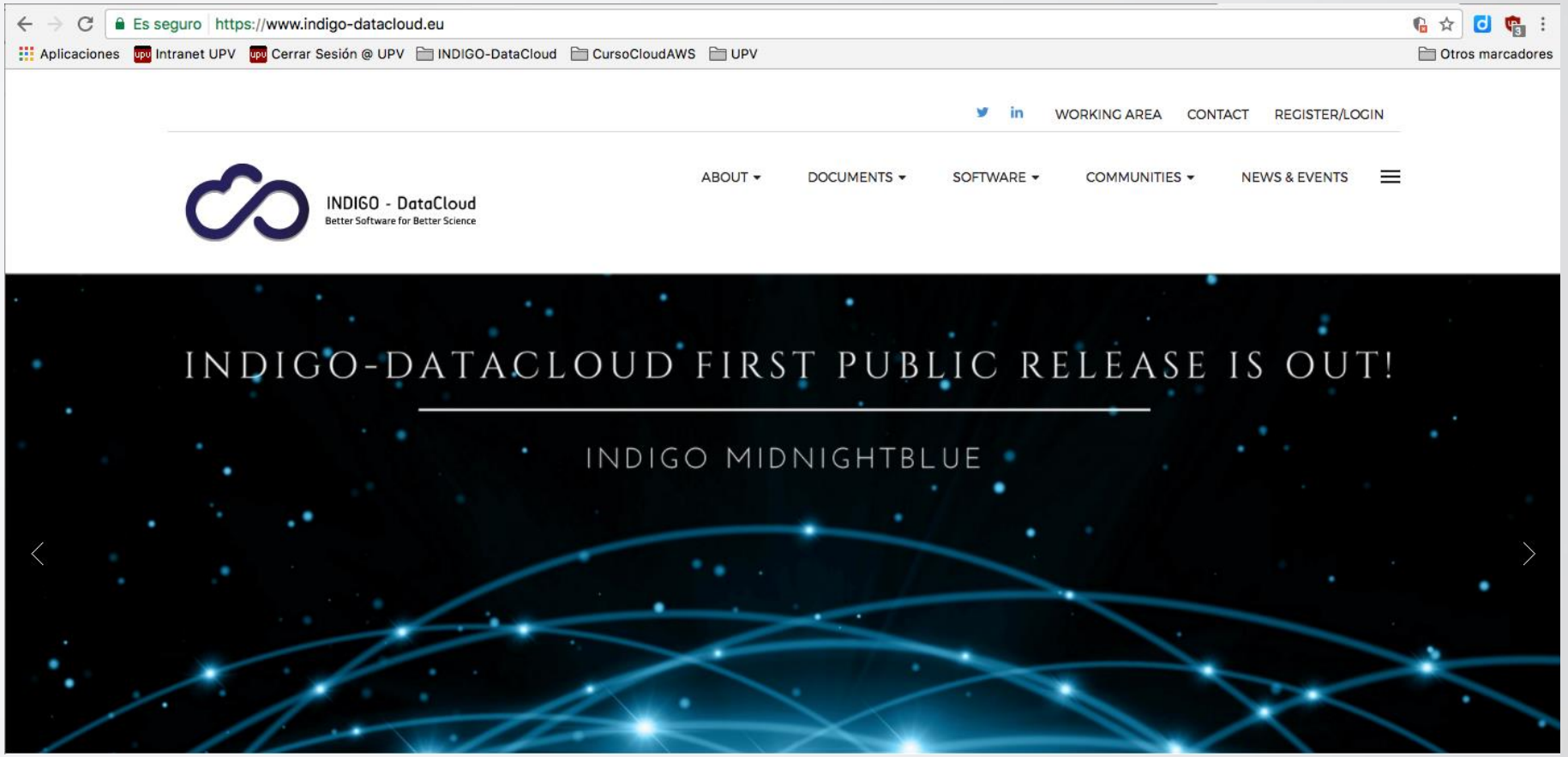
Docker 101: Privileges

- A container have certain restricted privileges (e.g. mounting volumes)
- Privileged containers:
  - docker run -it --privileged ubuntu:16.04 /bin/bash
  - **WARNING !** : this is equivalent to execute a process with root privileges in the Docker Host
- Fine grain control:
  - https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities
  - --cap-add
  - --cap-drop

Docker 101: Application delivery (I)

- Docker eases application delivery
  - Encapsulates applications as Docker images (includes all the application dependencies)
  - Images are stored in Docker Hub
  - Docker Engine used as a runtime environment
  - Build it Once, Run it Everywhere
  - Useful for legacy applications that only works with specific library versions
  - Avoids the installation of incompatible libraries in a production system to satisfy the requirements of a new application
- Prevents “It works on my computer”

Docker 101: Real examples



https://www.indigo-datacloud.eu/

Docker 101: Application delivery (II)

- Example: distribution of the CLI application Orchent
  - Building of the Docker image stored in Docker Hub that encapsulates the executable and its dependencies.
  - Define the following in the client:
    - alias orchent='docker run --rm -e ORCHENT_TOKEN=$ORCHENT_TOKEN -v $PWD:/data marica/orchent:latest’
    - We are passing variables between the host and the container
  - Clients use the tool as expected
    - orchen depls

Docker 101: Application delivery (II)

- Launching “orchent”:
  - If the image doesn’t exist in the local repository it is downloaded from Docker Hub
  - An ephimeral container starts running (finish running when the process finishes):
    - The current directory is mounted in the ‘*/data*’ directory of the container
    - An environment variable is passed from the host to the container
- The application is always executed inside a container in a transparent manner
  - With the exception of the file paths

One-Minute Quiz

- What's the ouput of the following command?
  - echo "bye" > hello.txt && docker run --rm -v `pwd`/ hello.txt:/tmp/ hello.txt alpine cat /tmp/ hello.txt
- What Docker functionalities is it using?
- What is the image used?
- Where is the image stored?

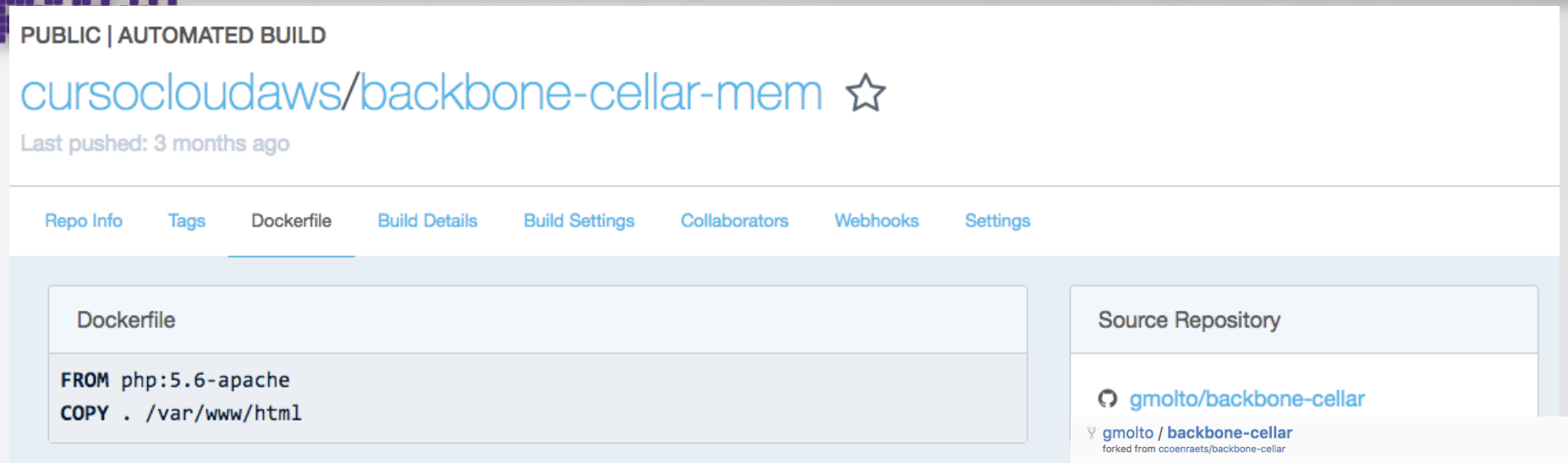
- Would the container continue running after the command execution?

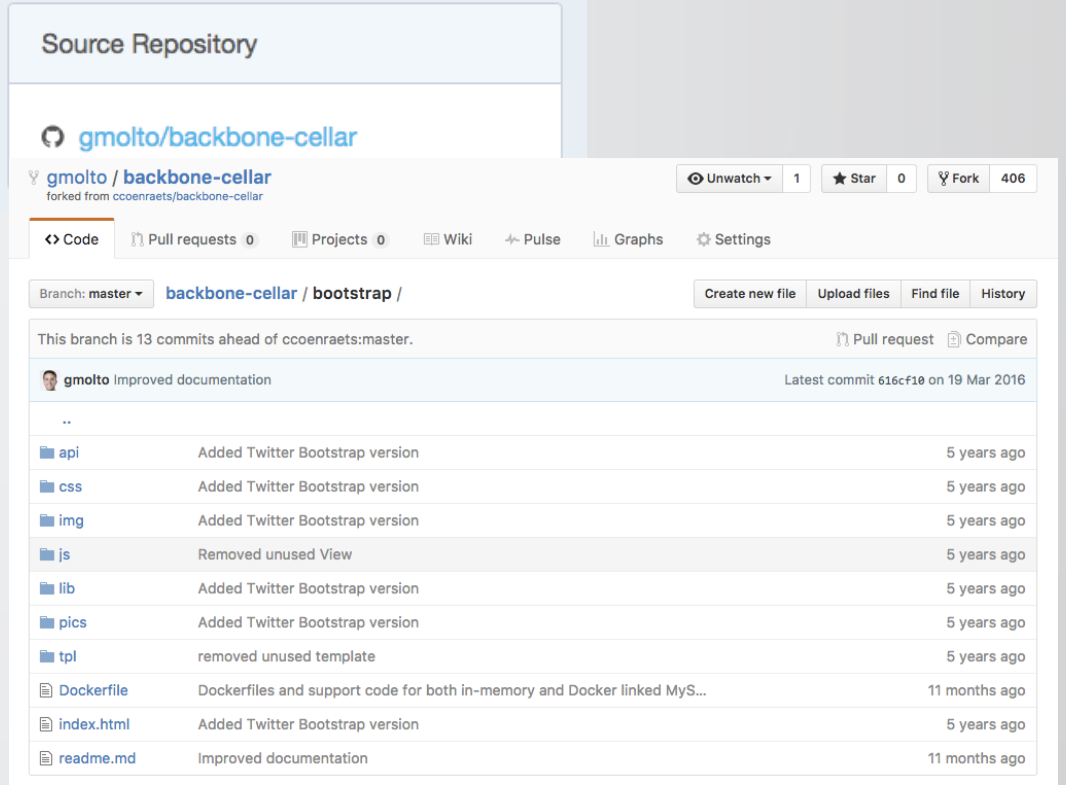
Docker 101: Building images

INDIGO - DataCloud

- Option 1:
  - Modify a running container, exit the container and save the content as a new image that will be stored in the registry
- Option 2:
  - Build the Docker image from a Dockerfile. The Dockerfile contains a recipe with the installation commands and a specific OS

Docker 101: Dockerfile (I)





- An existing image is taken as base to start defining a Dockerfile
- The rest of the Dockerfile describes the installation process of the application
- To build this example:
  - docker build -t cellar-mem .

Docker 101: Dockerfile (II)

INDIGO - DataCloud

- Dockerfile to build the Infrastructure Manager image

```
FROM ubuntu:16.04
MAINTAINER Miguel Caballer <micafer1@upv.es>
LABEL version="1.5.2"
LABEL description="Container image to run the IM service. (http://www.grycap.upv.es/im)"
EXPOSE 8899 8800
RUN apt-get update && apt-get install -y gcc python-dbg python-dev python-pip libmysqld-dev python-pysqlite2 openssh-client sshpass libssl-dev libffi-dev python-requests
RUN pip install setuptools --upgrade -I
RUN pip install CherryPy==8.9.1
RUN pip install pyOpenSSL --upgrade -I
RUN pip install MySQL-python msrest msrestazure azure-common azure-mgmt-storage azure-mgmt-compute azure-mgmt-network azure-mgmt-resource
RUN pip install IM
COPY ansible.cfg /etc/ansible/ansible.cfg
CMD im_service.py
```

https://github.com/grycap/im/blob/master/docker/Dockerfile

Docker 101: Dockerfile (III)

INDIGO - DataCloud

- Dockerfile to build the Kepler image

```
FROM indigodatacloud/ubuntu-sshd:14.04
MAINTAINER Mario David <mariojmdavid@gmail.com>
LABEL description="Container image to run Kepler WF engine"

RUN ansible-galaxy install indigo-dc.kepler && \
    ansible-playbook /etc/ansible/roles/indigo-dc.kepler/tests/kepler.yml

EXPOSE 22 5900
USER indigo
CMD /etc/init.d/vmcontext start &&  /bin/rm --force /tmp/.X0-lock /tmp/.X11-unix/X0 && \
    /usr/bin/vncserver -fg :0
```

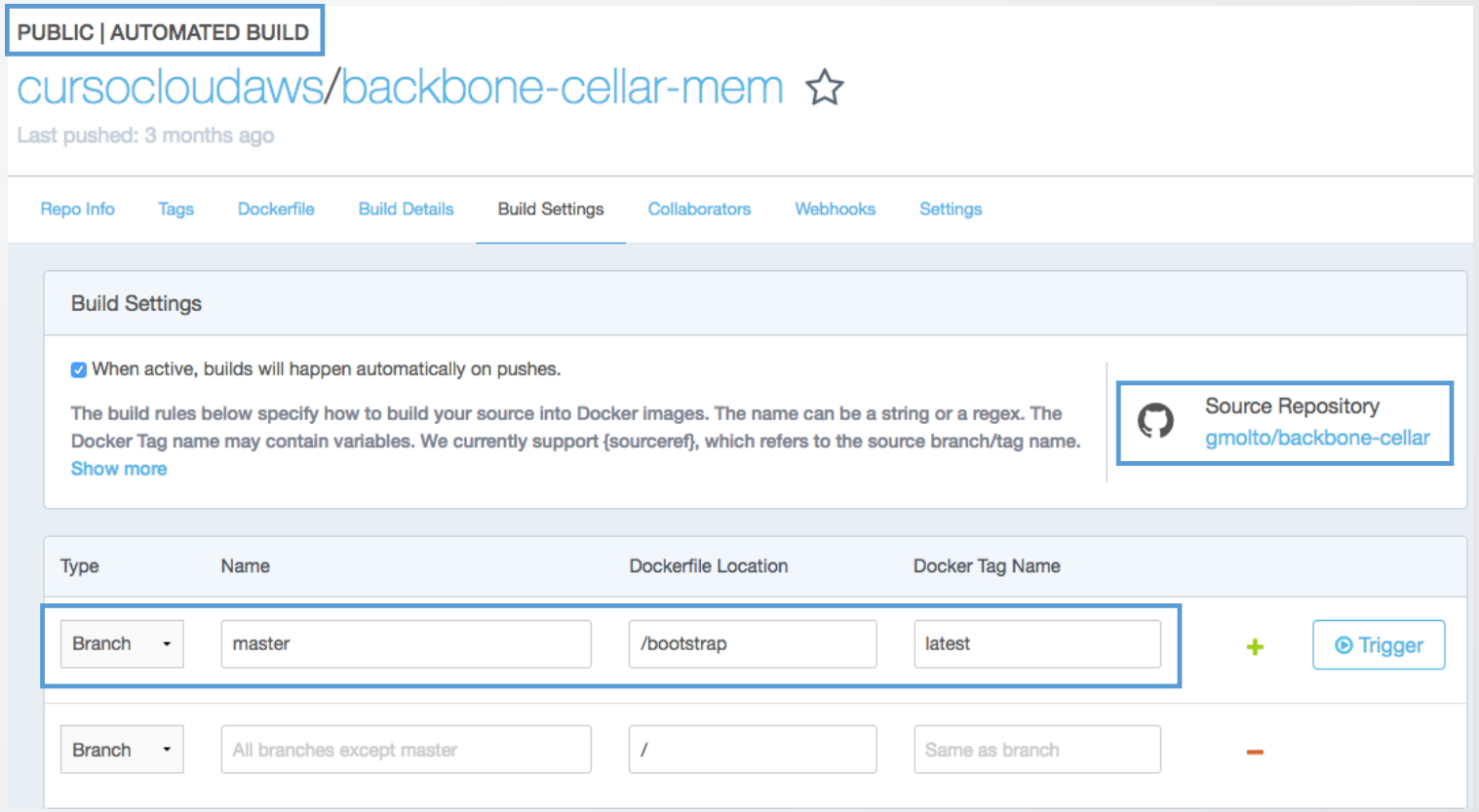
Ansible roles in Ansible Galaxy. Unified installation mechanism (*cross-platform. cross-OS)*

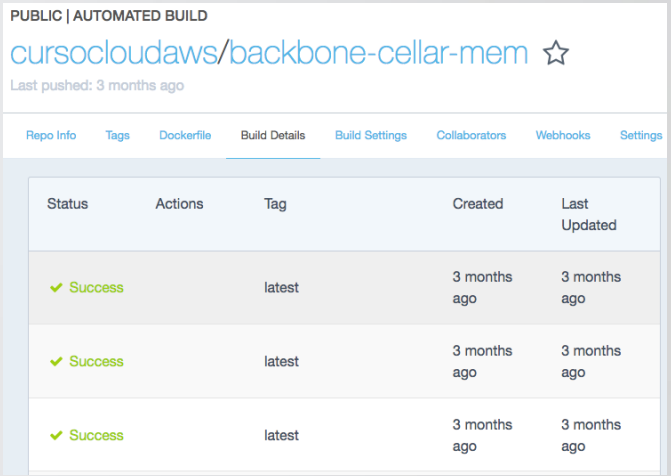
https://kepler-project.org/
https://github.com/indigo-dc/ansible-role-kepler/blob/master/docker-kepler/Dockerfile

Docker 101: Automated Build

INDIGO - DataCloud

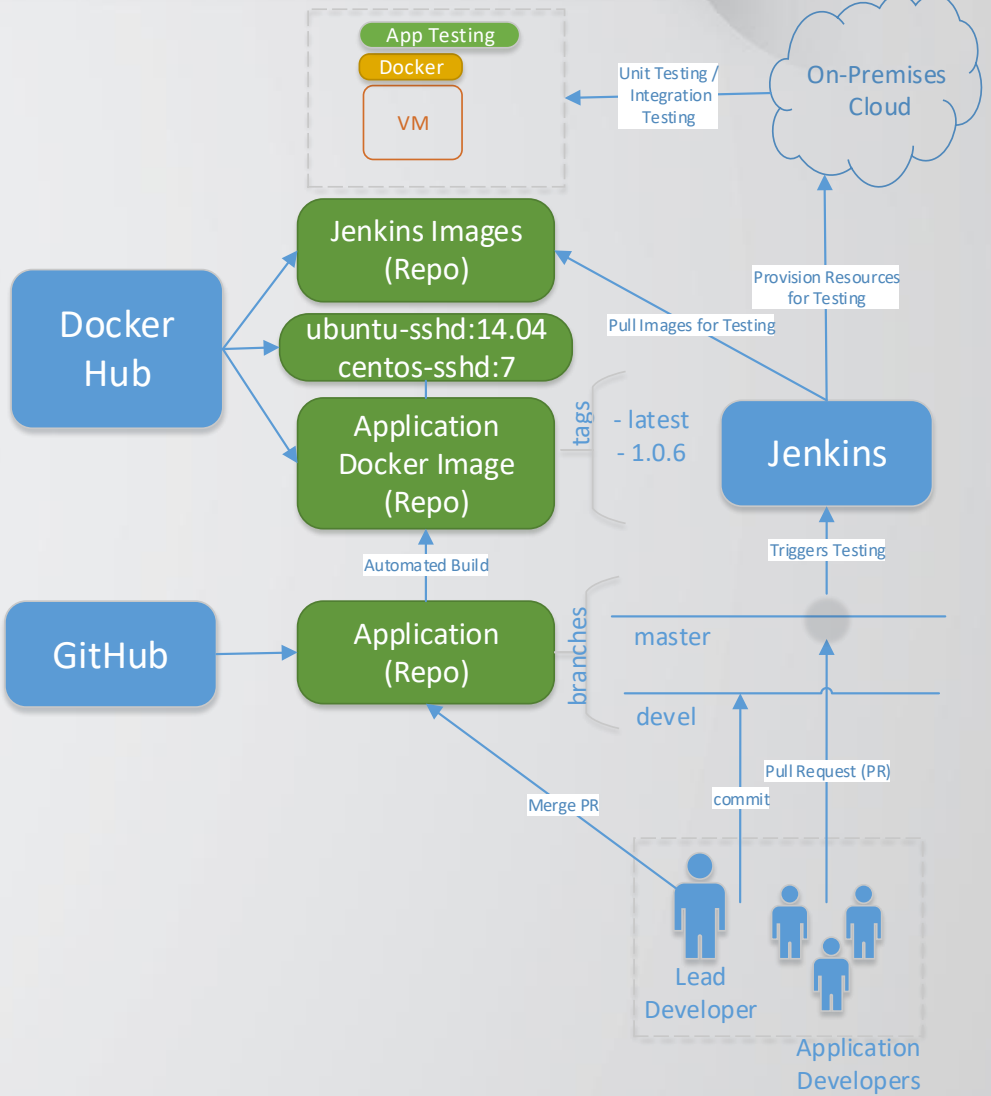
- Automated build of the Docker image in DockerHub when a change occurs in the application's code repository





Use case: Docker based CI (I)

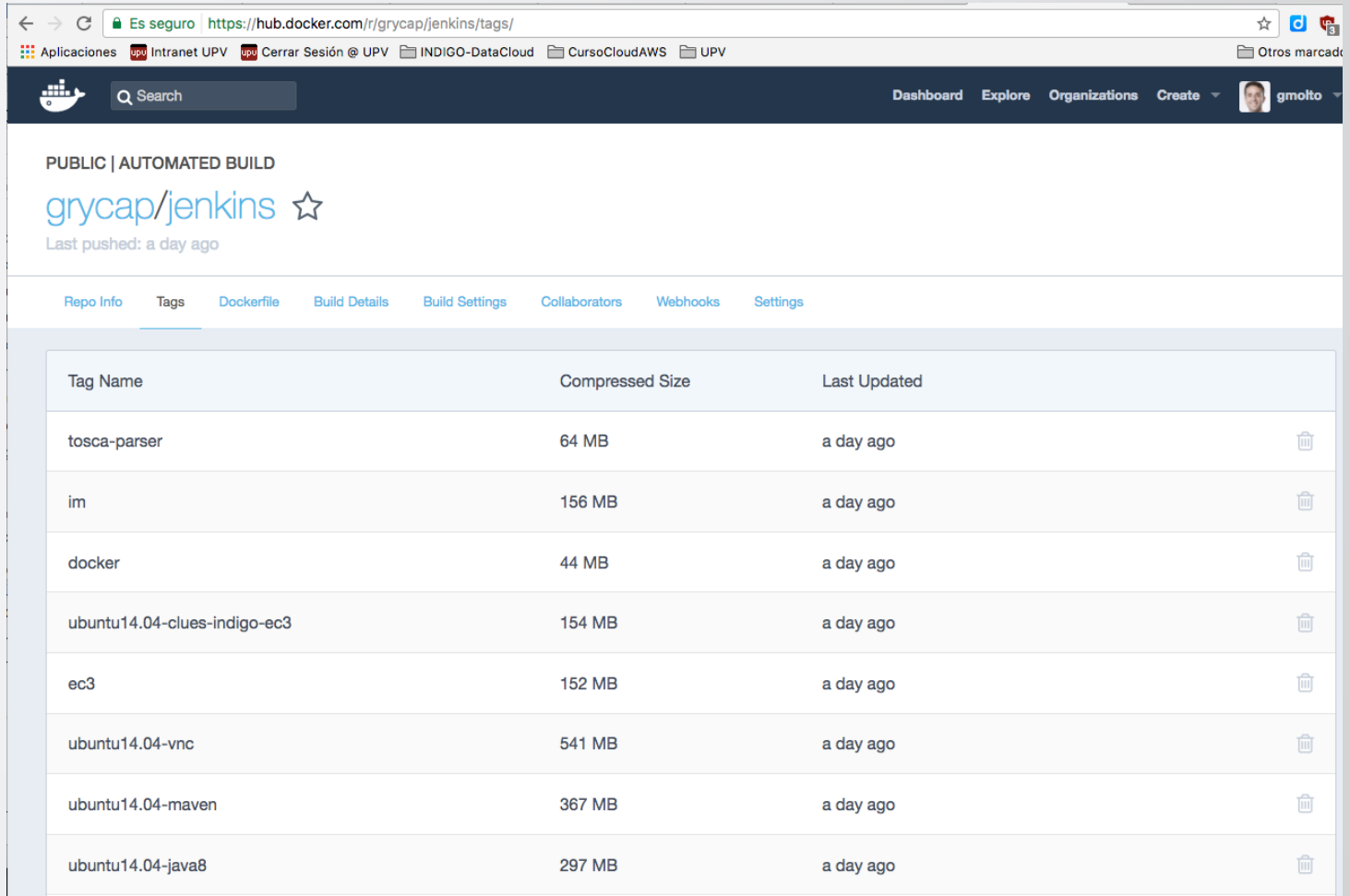
- Developers working on the devel branch of a GitHub repo.
- A PR on the master branch triggers the CI in Jenkins/Travis.
- Docker images in DockerHub are used to execute the Jenkins jobs in the right execution env.
- Merging the PR into the master branch triggers an Automated Build to create a new Docker image in Docker Hub.



Use case: Docker based CI (II)

- Different tags for different configurations
- Allows to deploy a testing environment in seconds
- Integrated with Jenkins

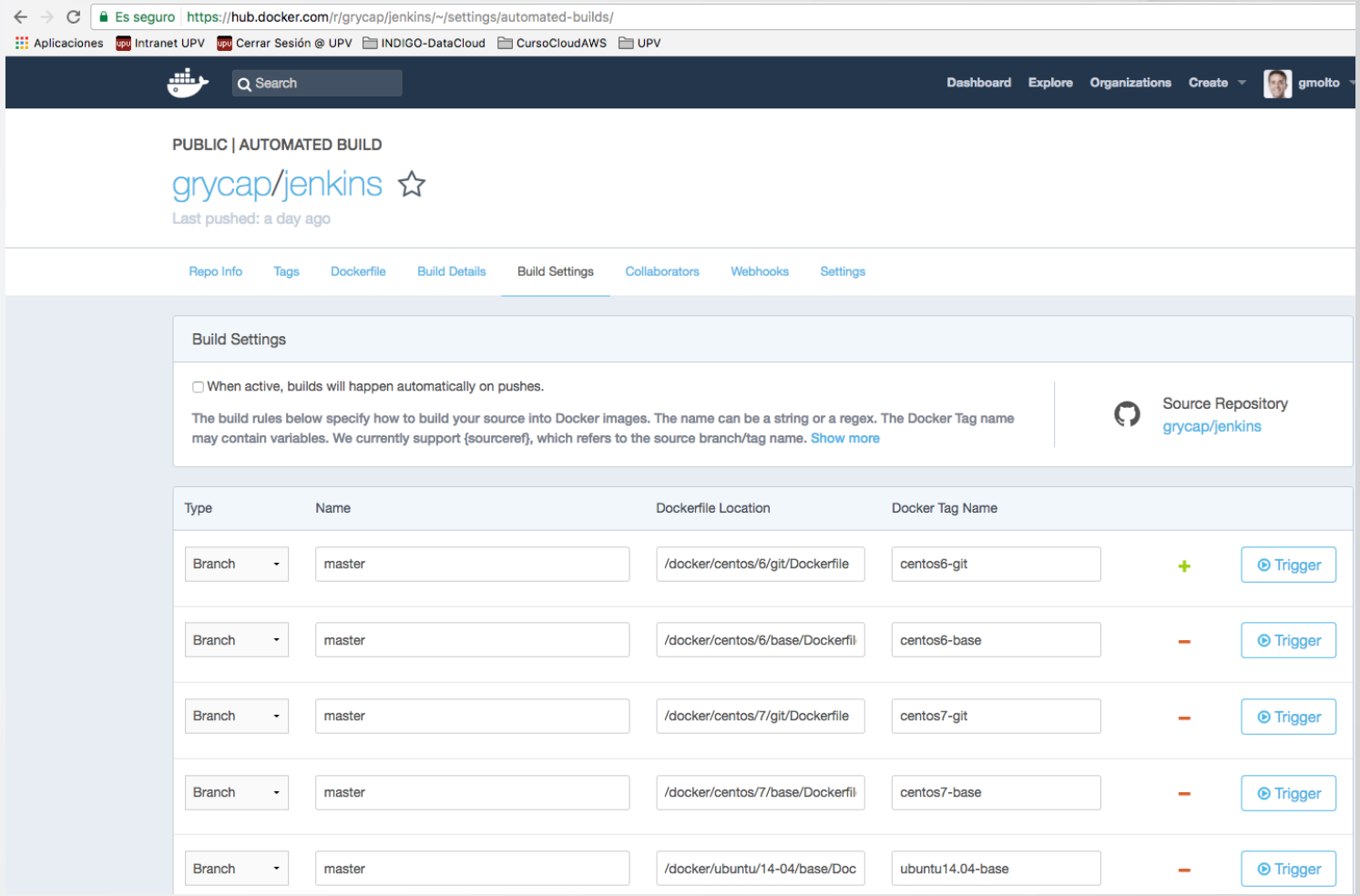
https://hub.docker.com/r/grycap/jenkins/tags/



Use case: Docker based CI (III)

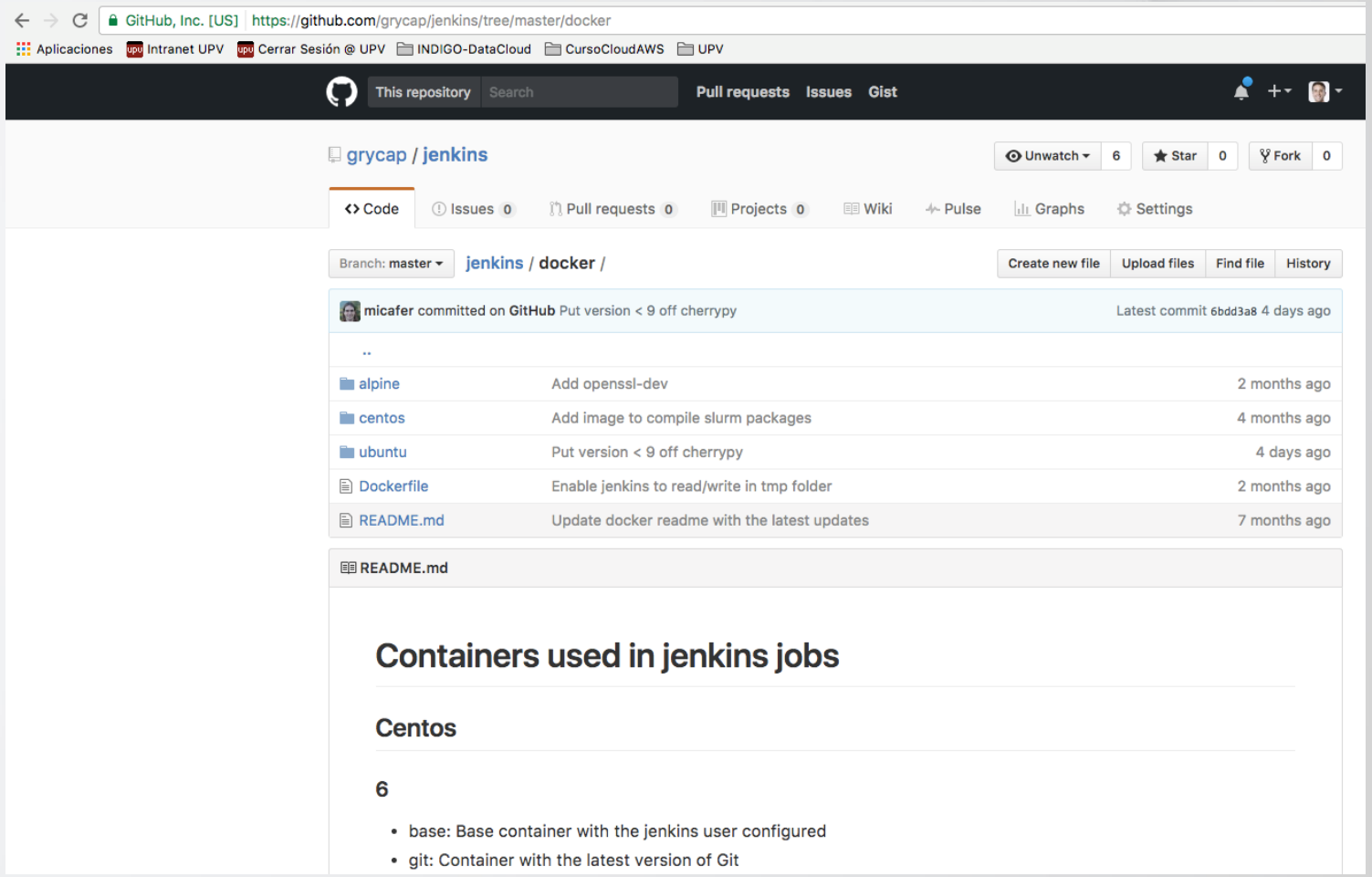
INDIGO - DataCloud

- Multiple Docker images from multiple Dockerfiles where the executing environment is specified



Use case: Docker based CI (IV)

- A commit in the master branch starts the automated building process in DockerHub



Conclusions

- Docker is a platform for the creation and execution of containers as well as the management and storage of container images
- Docker can be used to ease the development and execution of applications in multiple environments
- Docker appears as an efective solution in environments where traditionally there has been virtualization of GNU/Linux over GNU/Linux. In such environments Docker doesn’t present innecesary overcost

Index

- Docker
- **Ansible**

Automated Deployment with Ansible

- Ansible is a configuration management and provisioning tool, similar to Chef, Puppet or Salt.
- Agentless
  - uses SSH by default to make the connection to the target node
  - can be run locally
- Configuration in YAML
- Very easy to learn
  - http://docs.ansible.com/ansible/index.html

Building blocks: Playbook

- Defines sequences of tasks (Plays) to be executed on a group of hosts
  - Describes policies machines under management shall enforce
  - Contains variables, tasks, handlers, files, templates and roles
  - Expressed in YAML
- Idempotent

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

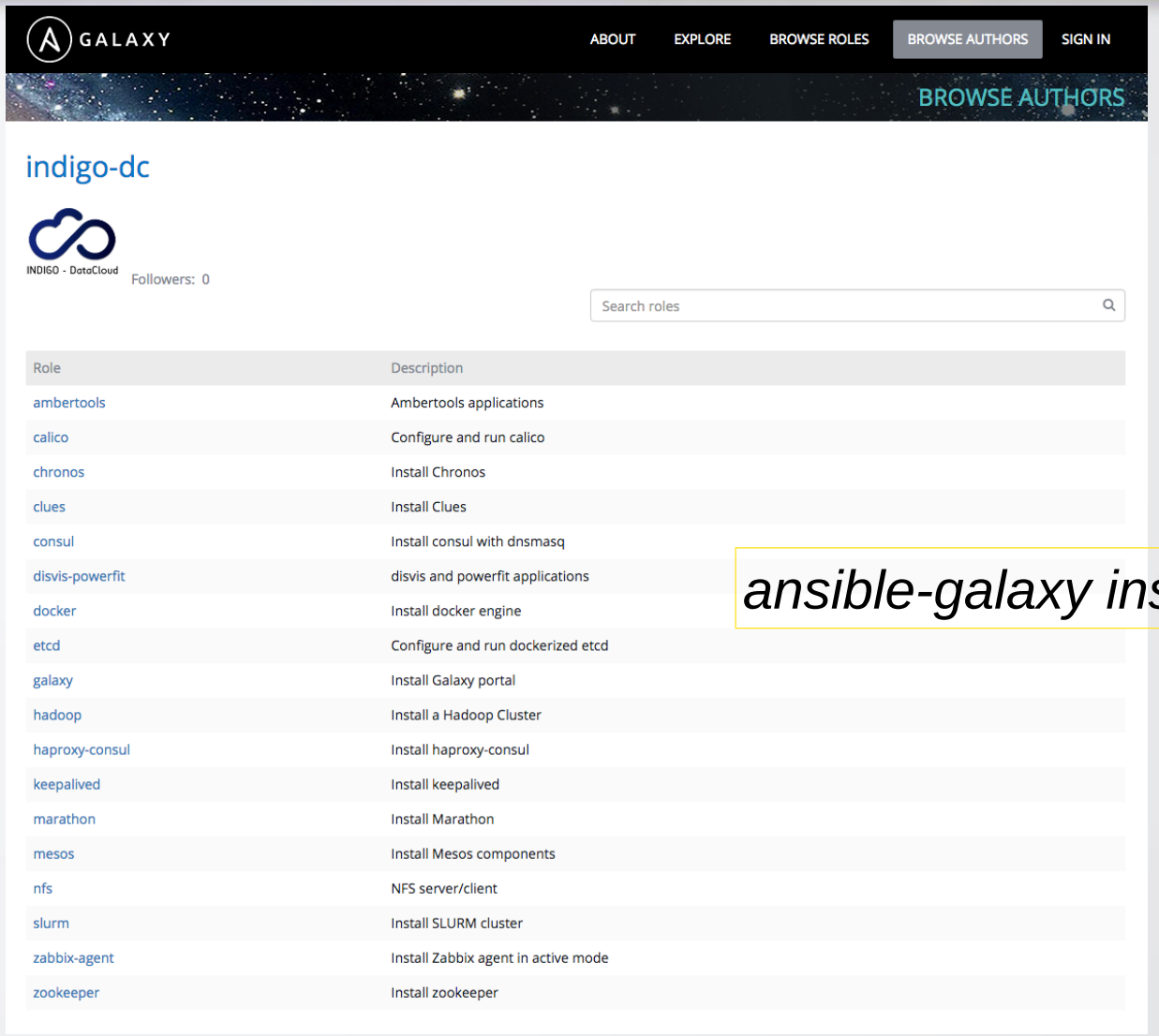
Building blocks: Roles

INDIGO - DataCloud

- The best way to organize your playbooks.
  - Structure content into related vars, tasks, files, handlers, etc.
  - File structure for automated inclusion of role specific content
  - Roles can be shared and pulled from Ansible Galaxy, GitHub, etc.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  roles:
     - webserver
```

Ansible Galaxy: indigo-dc roles



Role	Description
ambertools	Ambertools applications
calico	Configure and run calico
chronos	Install Chronos
clues	Install Clues
consul	Install consul with dnsmasq
disvis-powerfit	disvis and powerfit applications
docker	Install docker engine
etcd	Configure and run dockerized etcd
galaxy	Install Galaxy portal
hadoop	Install a Hadoop Cluster
haproxy-consul	Install haproxy-consul
keepalived	Install keepalived
marathon	Install Marathon
mesos	Install Mesos components
nfs	NFS server/client
slurm	Install SLURM cluster
zabbix-agent	Install Zabbix agent in active mode
zookeeper	Install zookeeper

ansible-galaxy install indigo-dc.<role>