LUMBRERA

Integración de Herramientas Devops: Docker, Gitlab, Jenkins

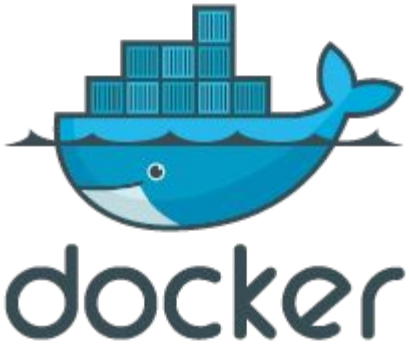
Anthony Mogrovejo
Consultor IT

Introducción

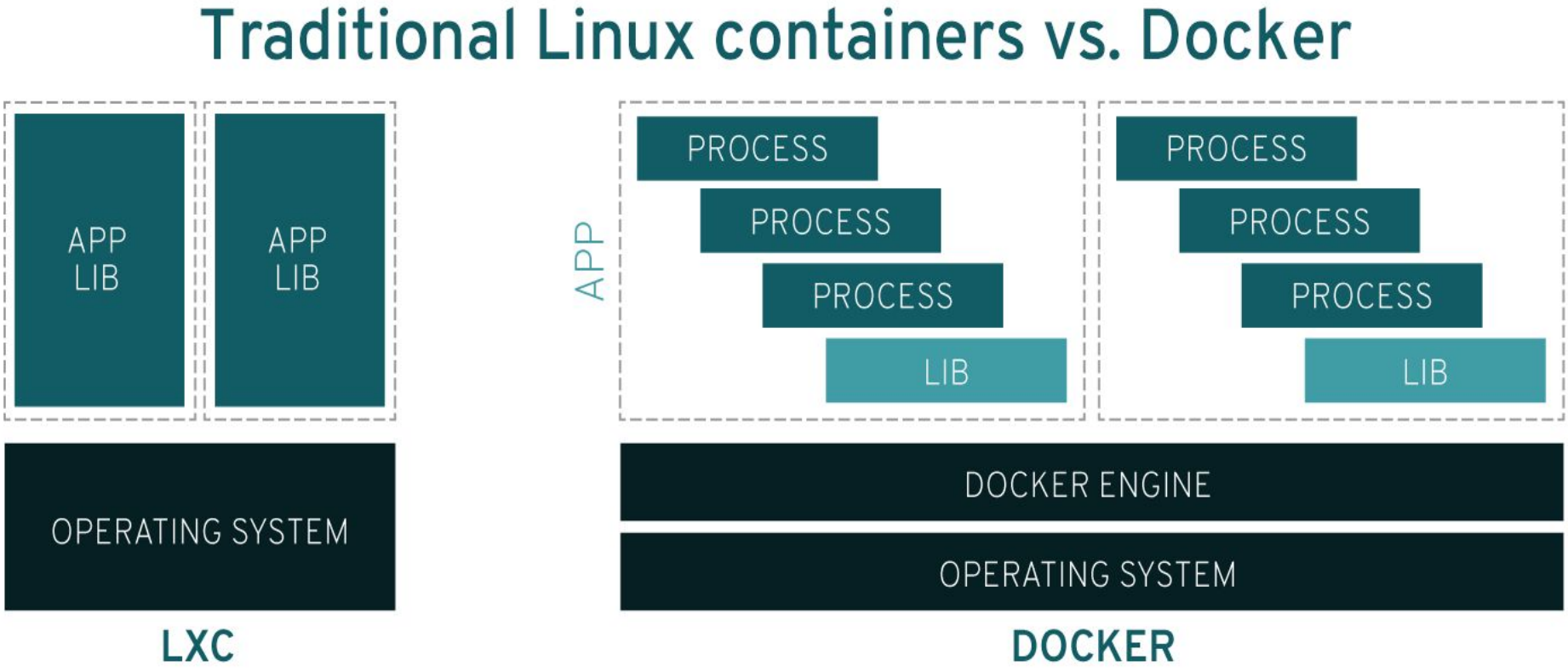
- Docker
- Git/GitLab
- Jenkins
- Despliegue

¿Quién soy yo?

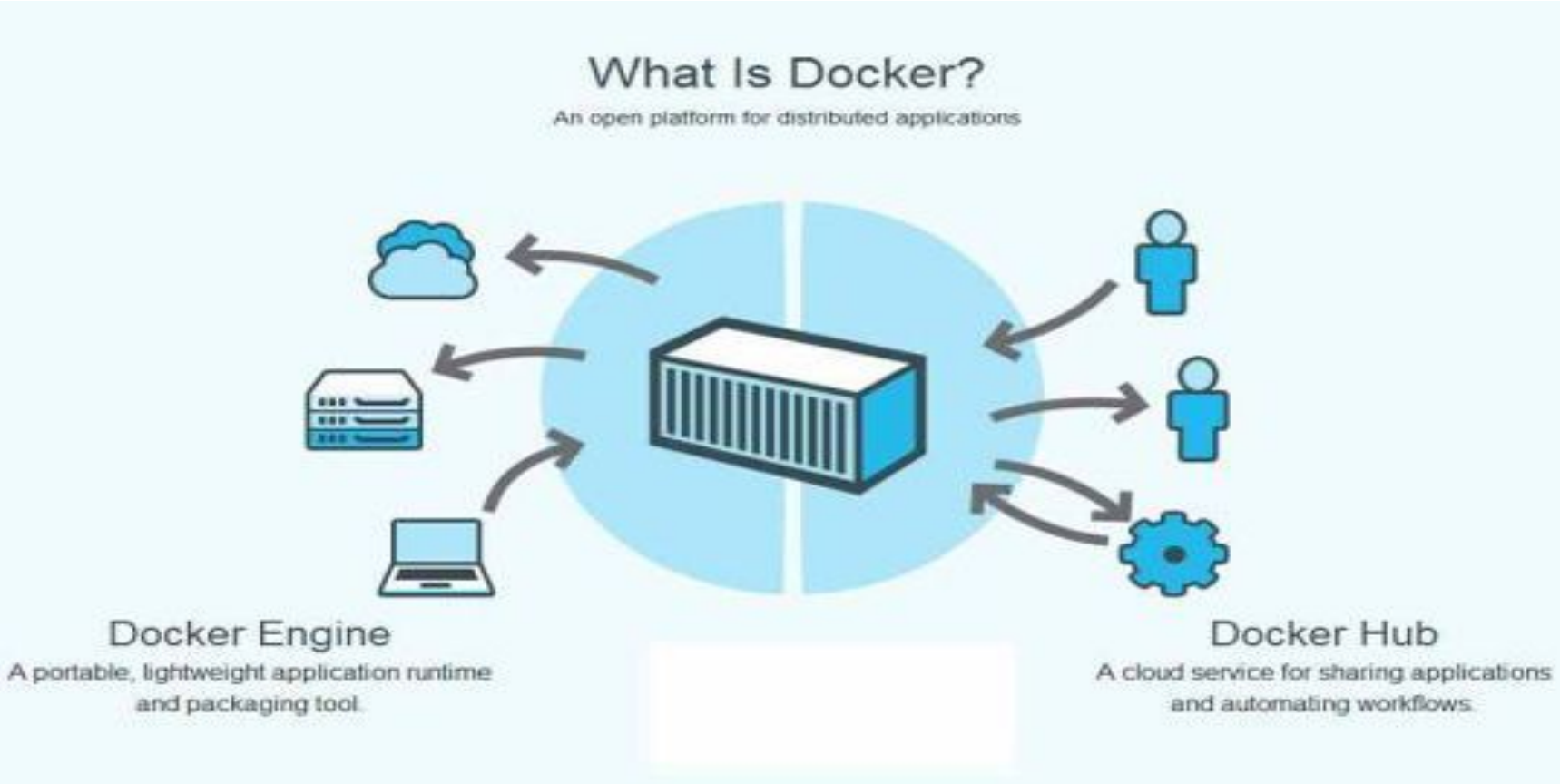
- Solution Architect - BBVA Banco Continental
- Especialista Infraestructura
- Geek Unix/Linux fan boy !!!
- RHCE / RHCSA v6
- Security Fan Boy
- Daddy
- Thai Boxing - Box - BJJ
- Embassador Fedora
- @kdetony


docker

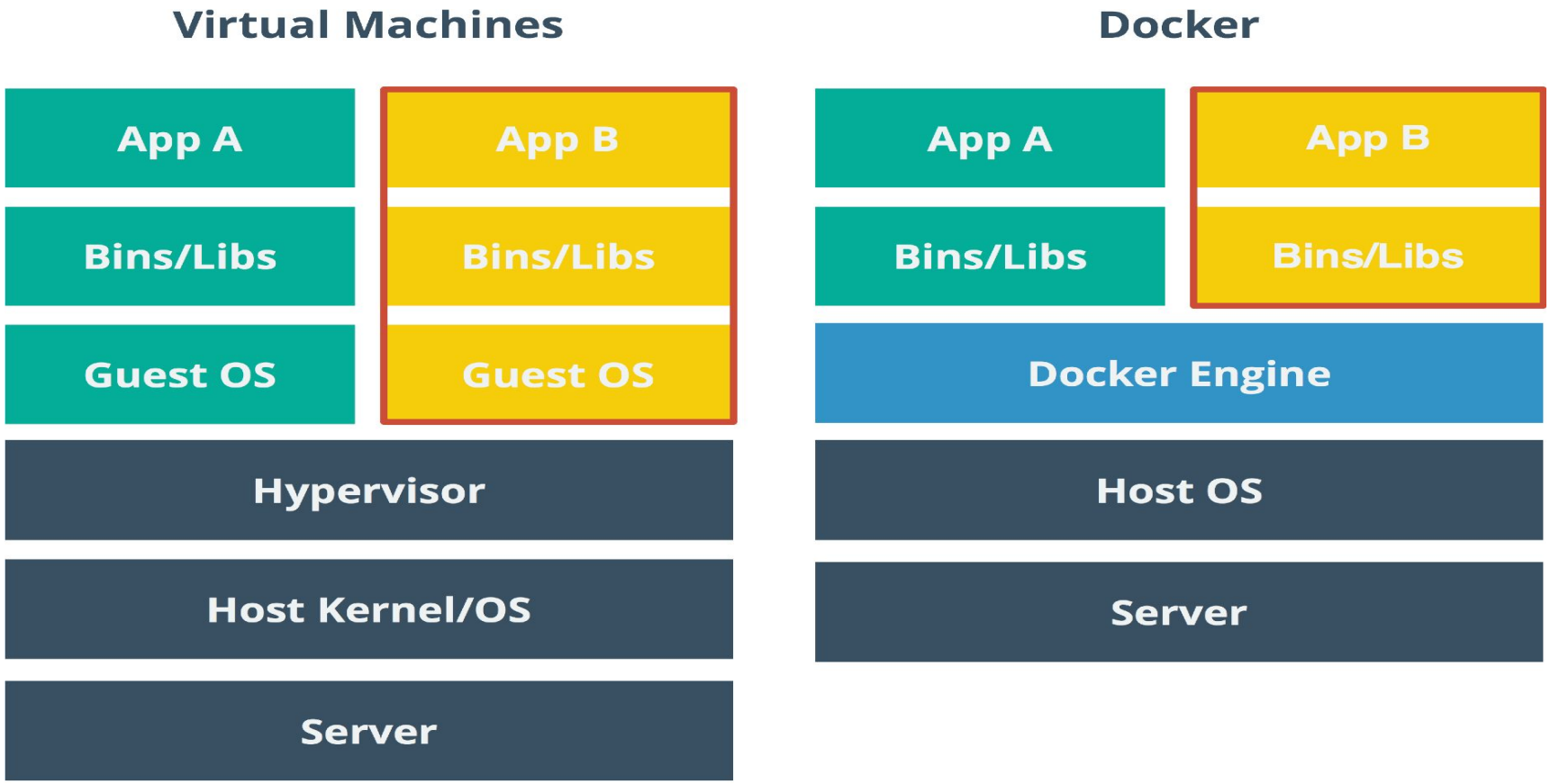
Docker


Traditional Linux containers vs. Docker
APP
LIB
APP
LIB
OPERATING SYSTEM
LXC
APP
PROCESS
PROCESS
PROCESS
LIB
PROCESS
PROCESS
PROCESS
LIB
DOCKER ENGINE
OPERATING SYSTEM
DOCKER

Docker


What Is Docker?
An open platform for distributed applications
Docker Engine
A portable, lightweight application runtime and packaging tool.
Docker Hub
A cloud service for sharing applications and automating workflows.

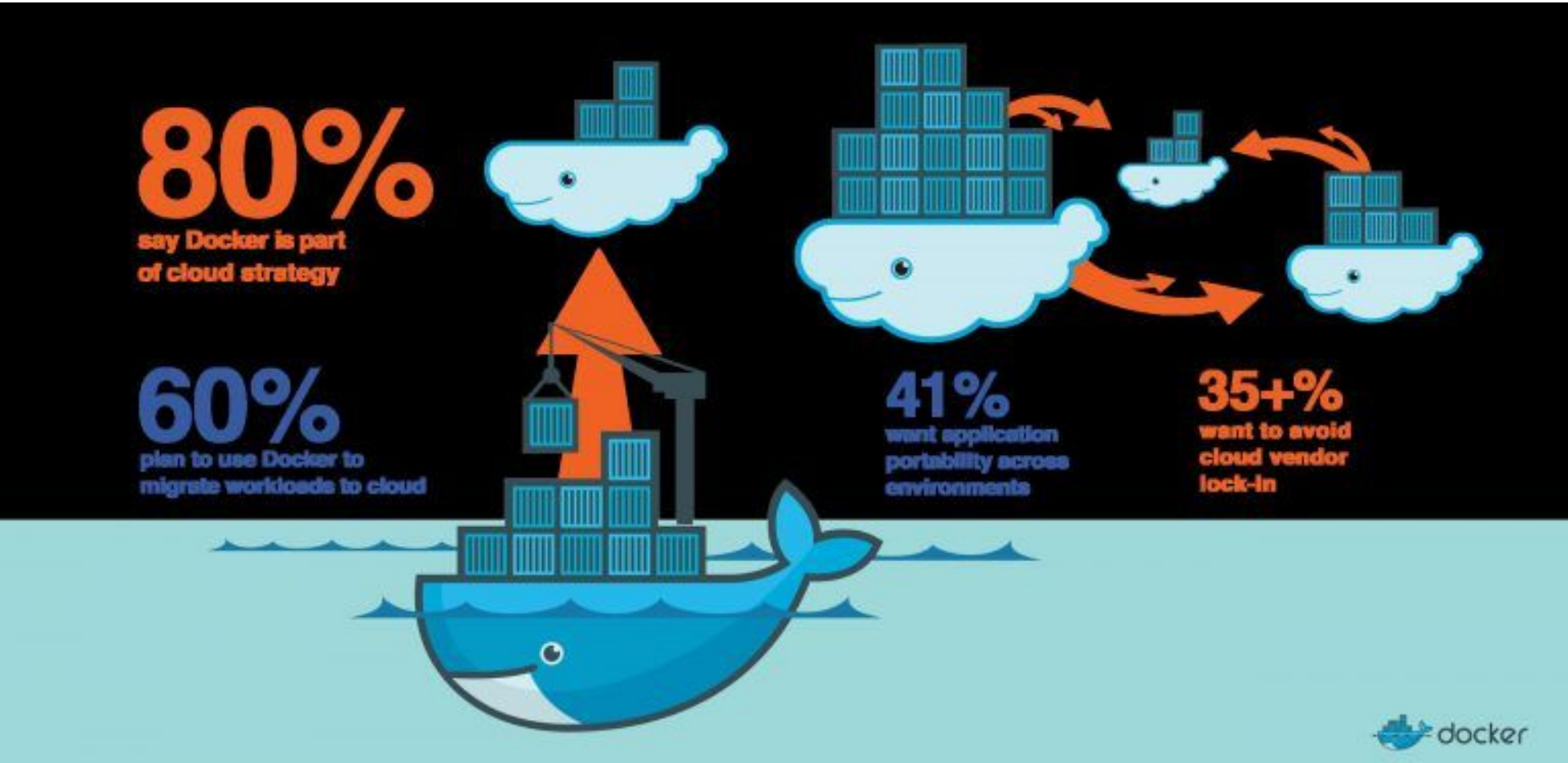
Docker


Virtual Machines
App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host Kernel/OS
Server
Docker
App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server

Docker

- Herramienta para construir, entregar y ejecutar aplicaciones.
- Empaqueta aplicaciones de forma general o particular.

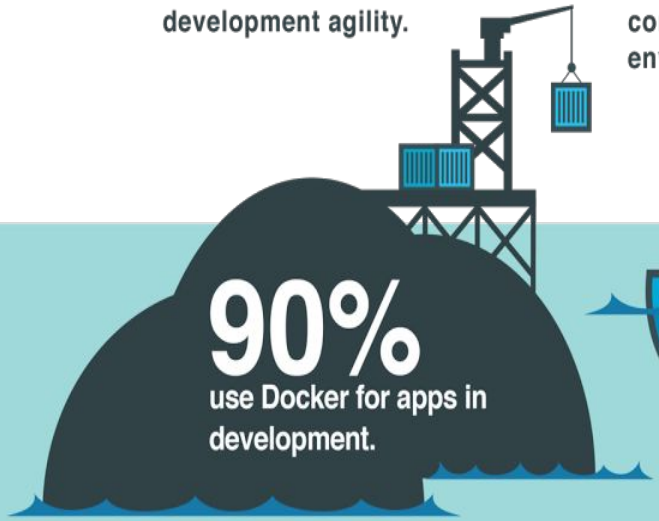
Docker


80%
say Docker is part of cloud strategy
60%
plan to use Docker to migrate workloads to cloud
41%
want application portability across environments
35+%
want to avoid cloud vendor lock-in
docker

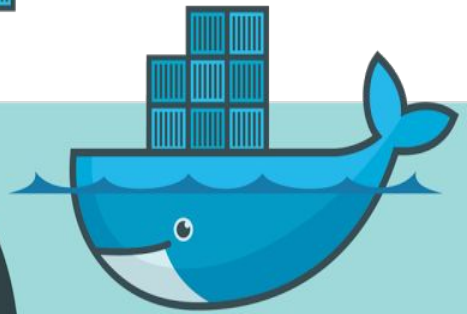
65%
use Docker to deliver development agility.

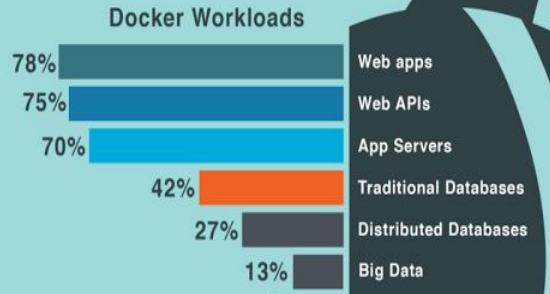
48%
use Docker to control app environments.

41%
use Docker to achieve app portability.



90%
use Docker for apps in development.





58%
use Docker for apps in production.

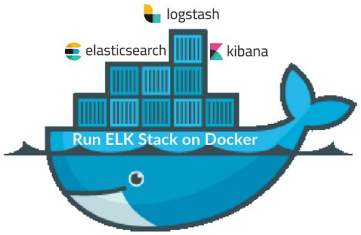
90%
plan dev environments around Docker.

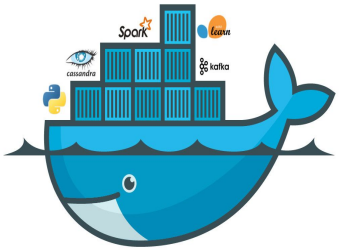
80%
plan DevOps around Docker.

docker

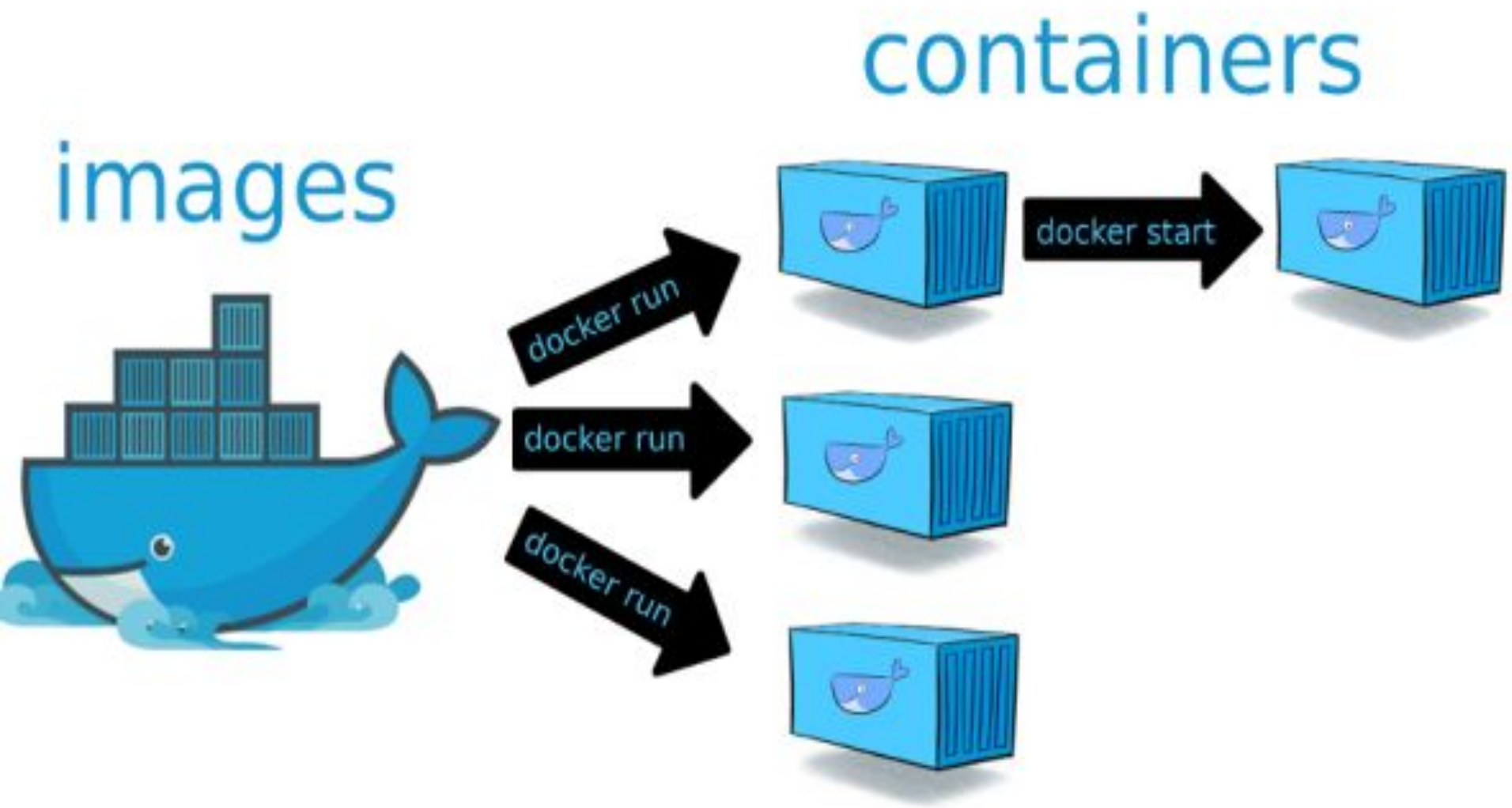
Docker

Imágenes





Docker


containers
images
docker run
docker run
docker run
docker start

Web Application

Layer 9 : CMD *Start Pintail.ai website*
Layer 8 : RUN *Commands to set up Pintail.ai Website*
Layer 7 : ADD *Pintail.ai binaries*
Layer 6 : FROM *Node.js - Apline Linux*

Application Framework

Layer 5 : RUN *Commands to set up Node.js*
Layer 4 : ADD *Node.js binaries*
Layer 3 : FROM *Alpine Linux*

Operating System

Layer 2 : RUN *Commands to set up OS*
Layer 1 : ADD *Operating System binaries*
Layer 0 : FROM *Scratch*

```
FROM debian:jessie

RUN apt-get update && \
    apt-get install -y python python-dev python-pip python-virtualenv wget unzip && \
    rm -rf /var/lib/apt/lists/*

WORKDIR /tmp

RUN wget https://github.com/pjcoole/ascii-telnet-server/archive/master.zip && \

    unzip master.zip && \
    mkdir /app && \
    dd if=/dev/zero of=output.dat bs=1024 count=102400 && \
    cp -r ./ascii-telnet-server-master/ascii-telnet-server /app && \
    rm -rf /tmp/*

WORKDIR /app/ascii-telnet-server

RUN chmod +x ascii-telnet-server.py

EXPOSE 23

CMD ./ascii-telnet-server.py -f sw1.txt --standalone

```

```yaml
version: '2'
 services:
   db:
     image: mysql:5.7
     volumes:
       - "./.data/db:/var/lib/mysql"
     ports:
       - "3306:3306"
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: 1234
       MYSQL_DATABASE: foro
       MYSQL_USER: pedro
       MYSQL_PASSWORD: 1234

   mybb:
     depends_on:
       - db
     image: fbmac/mybb:latest
     links:
       - db
     ports:
       - "8000:80"
     restart: always
```

Docker

PASO 1 : Construir la Imagen

$docker build -t “NAME” .

Docker

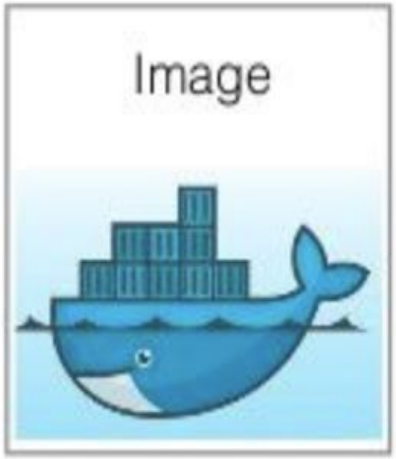
PASO 2 : Construir el contenedor

```
php:
  build: php
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - ./php/www:/var/www/html
  links:
    - db
```

$docker-compose up -d

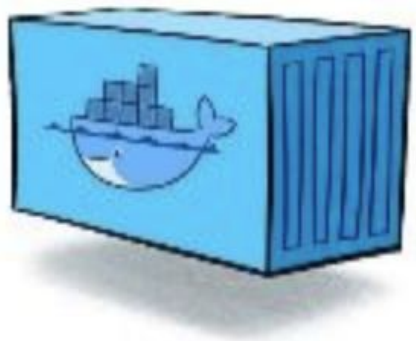
Dockerfile

build



Docker Image

run



Docker Container

> COMANDOS DE DOCKER_

DOCKER PS

DOCKER RUN

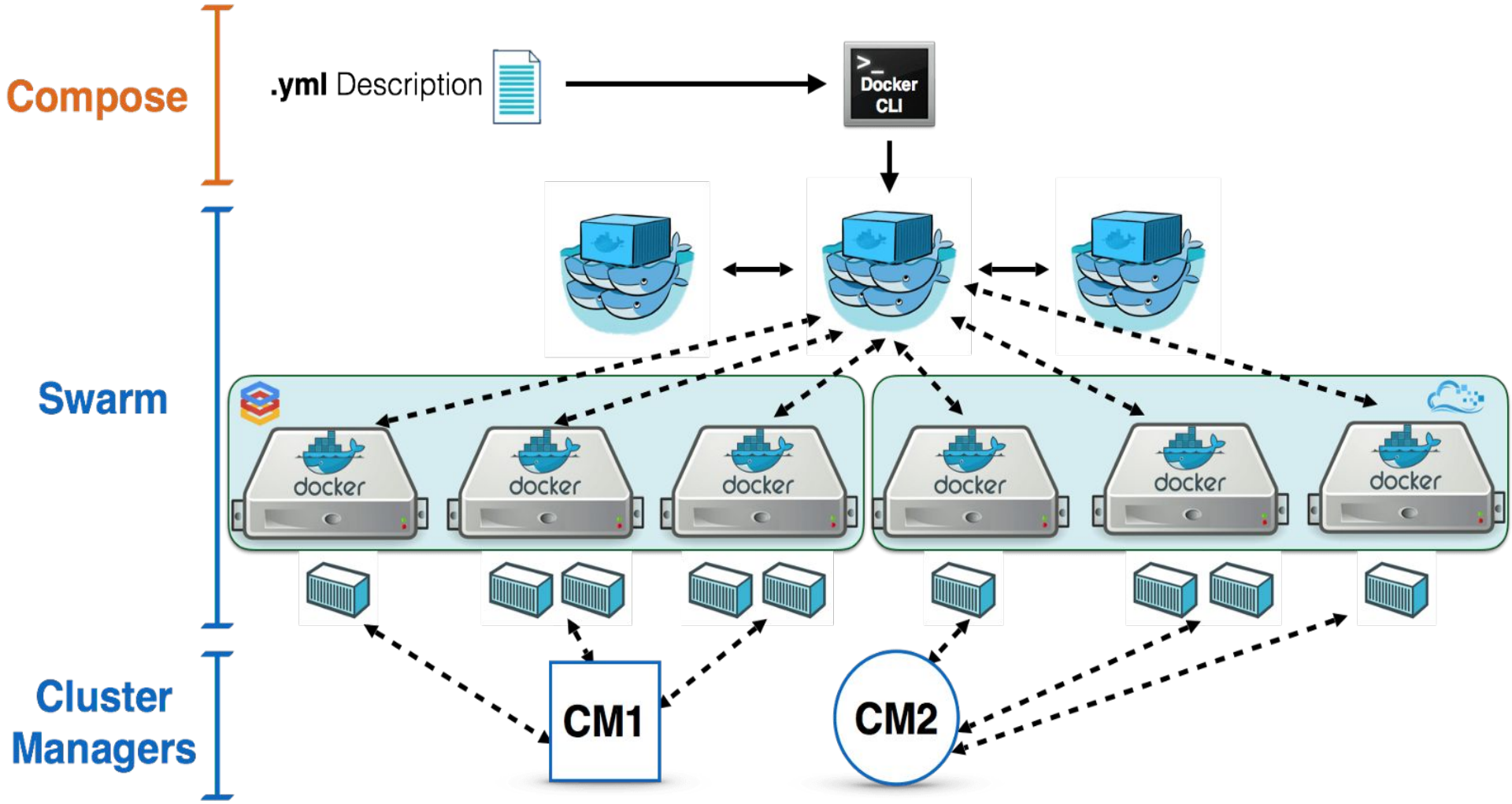
DOCKER COMMIT

DOCKER IMAGE

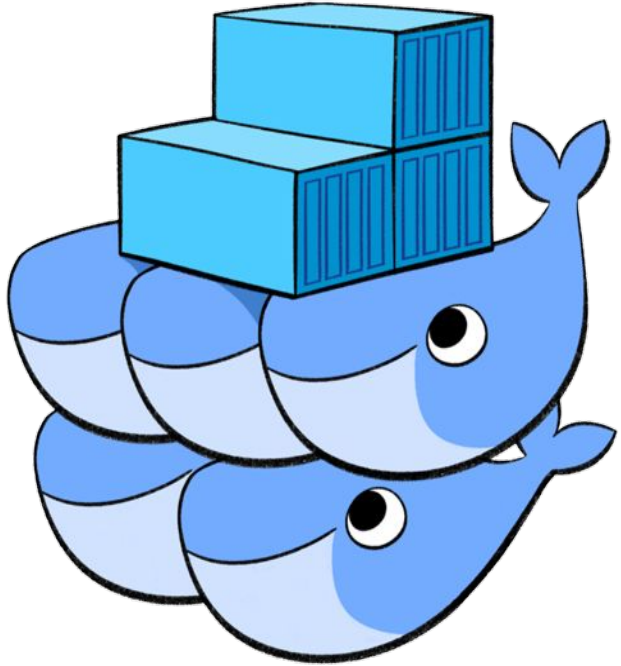
Docker

Comandos básicos - Images

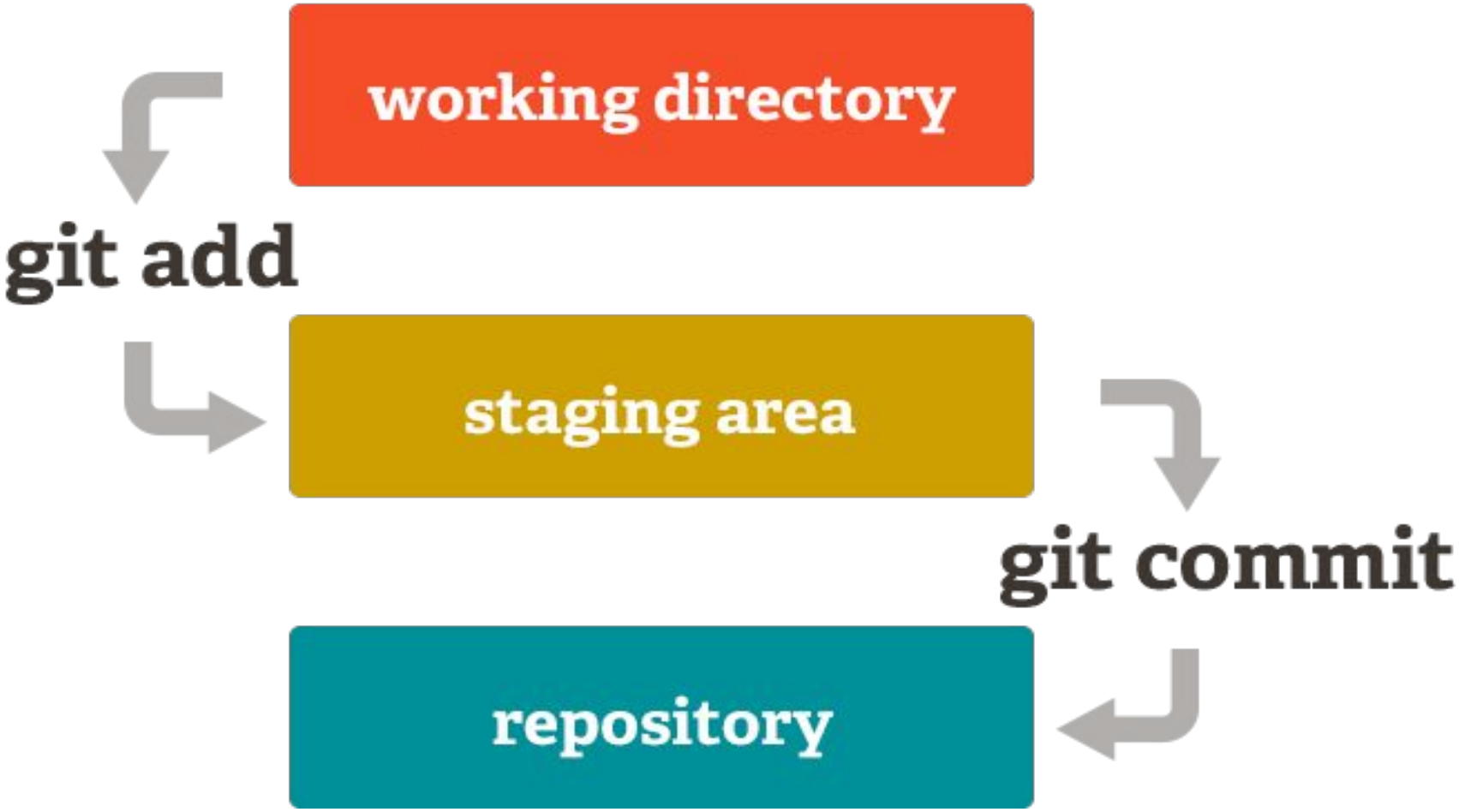
- **Listar**: *$ docker images*
- **Buscar**: *$docker search alpine*
- **Descargar**: *$ docker pull alpine:latest*
- **Eliminar**: *$ docker rmi alpine:tag*
- **Contruir**: *$ docker build -t myimage:latest .*
- **Subir**: *$ docker push myimage:latest*


Compose
.yml Description
>_
Docker
CLI
Swarm
docker
docker
docker
docker
docker
docker
Cluster
Managers
CM1
CM2

VS




git


working directory
git add
staging area
git commit
repository

VERSIONADO

Estandar *X.YY.ZZ*

- X -> Mayor versión
- YY -> Minor versión
- ZZ -> Patch

TIPOS DE RAMAS

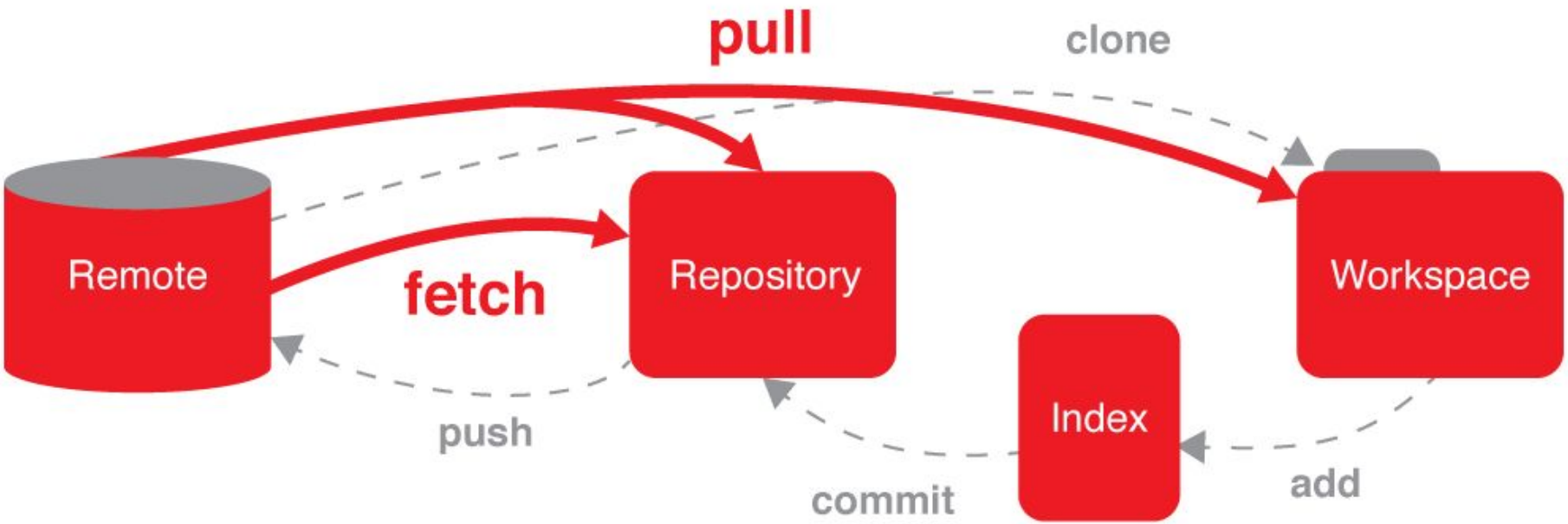
- Infinitas: *develop*, *master*
- Temporales: *features*, *releases*, *hotfix*

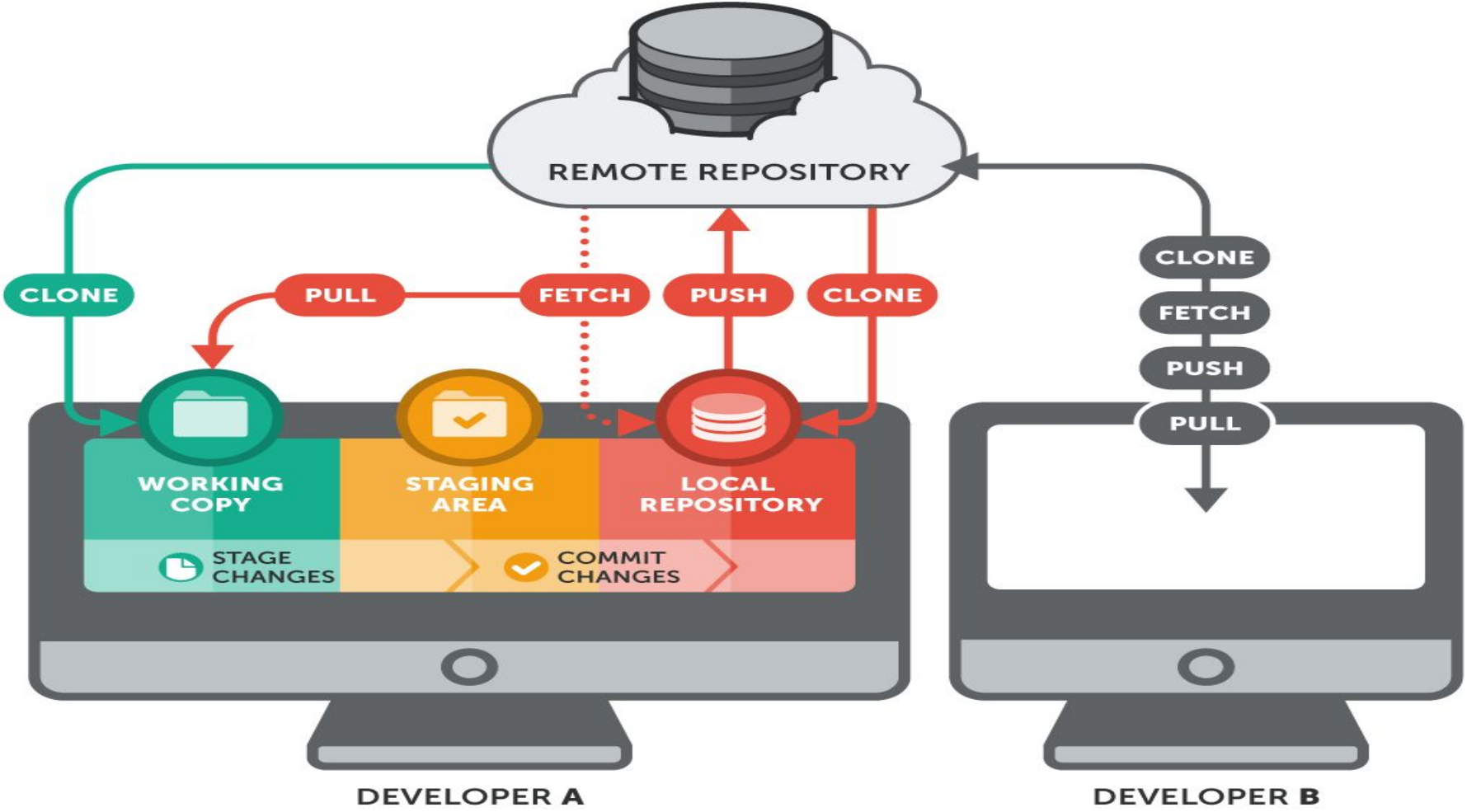
DEVELOP

- Rama de desarrollo
- Cualquier funcionalidad nueva parte de esta rama y se mergea de nuevo aquí
- Jenkins configurado para ejecutarse contra ella

MASTER

- Rama de producción
- Nunca se commitea en ella
- Jenkins configurado para ejecutarse contra ella, aunque nunca debería fallar ;-)


pull
clone
Remote
fetch
Repository
Workspace
push
Index
commit
add


REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
CLONE
FETCH
PUSH
PULL
DEVELOPER B

GIT

Comandos

Iniciar repositorio
git init

Ver cambios
git status

Agregar cambios
git add .

Salvar cambios
git commit -m 'Mi primer commit xD'

Ver historial
git log

Enviar cambios
git push -u origin master

Traer cambios
git pull origin master

Clonar repositorio
git clone git@github.com:user/repositorio.git

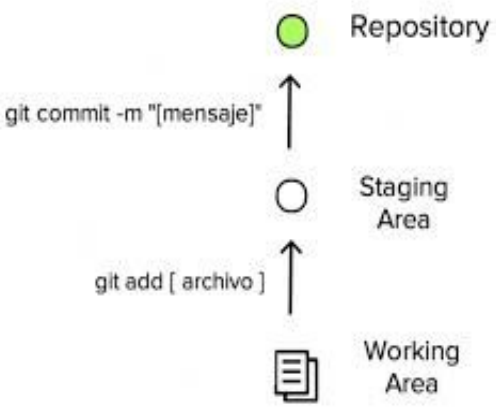
GIT

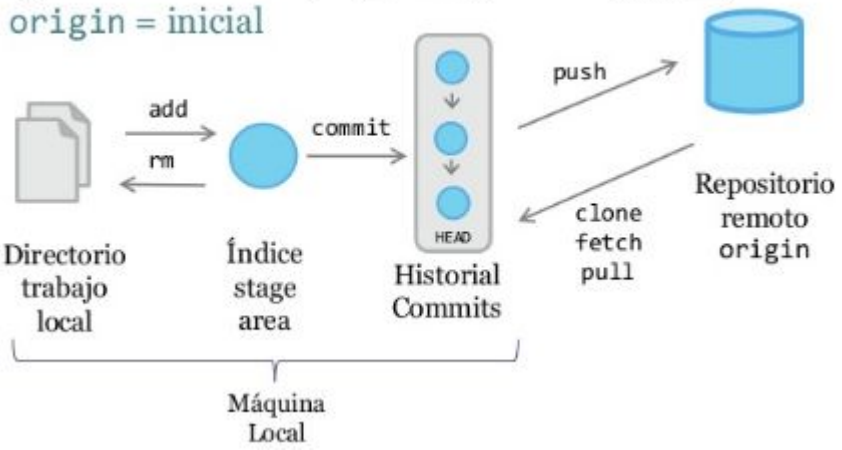
Comandos

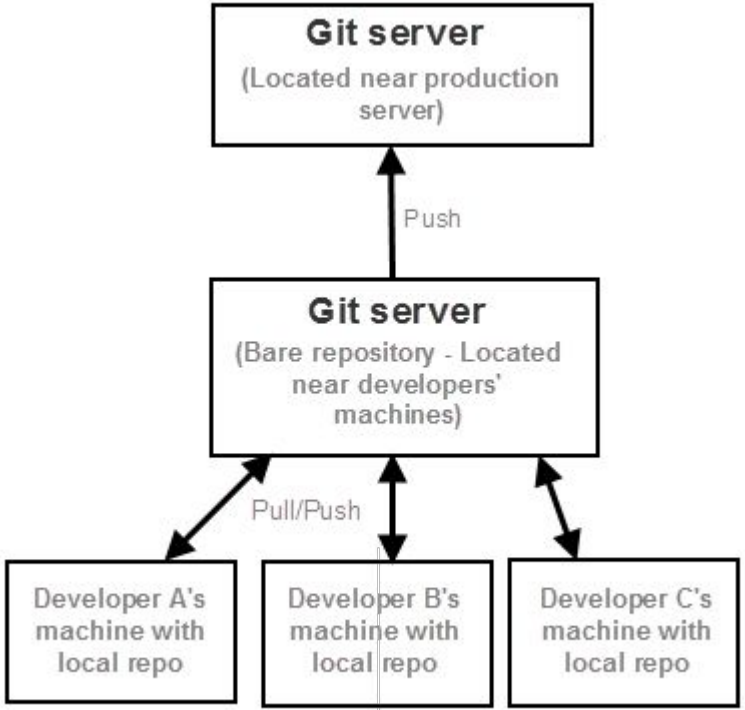
- *git add* - agrega al index (staging)
- *git commit* - guarda el index actual con el autor, mensaje, fecha, etc.
- *git status* - muestra los archivos con diferencias en working dir, staging, y los untracked
- *git diff* - lo que cambió pero no está staged
- *git diff --cached* - lo que va para el próximo commit
- *git log*

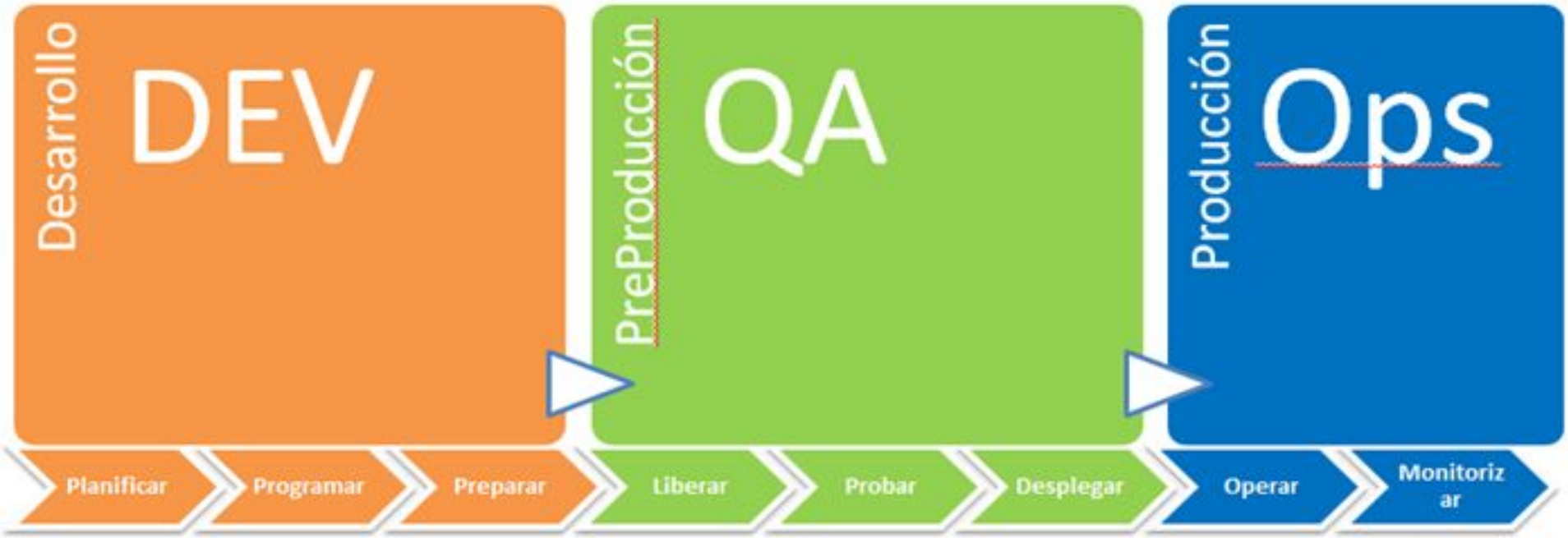
- $git config
- $git init
- $git clone <path>
- $git add <file_name>
- $git commit
- $git status
- $git remote
- $git checkout <branch_name>
- $git branch
- $git push
- $git pull
- $git merge <branch_name>
- $git diff
- $git reset
- $git revert
- $git tag
- $git log

GIT - Arquitectura









Agile

Integración continua(CI)

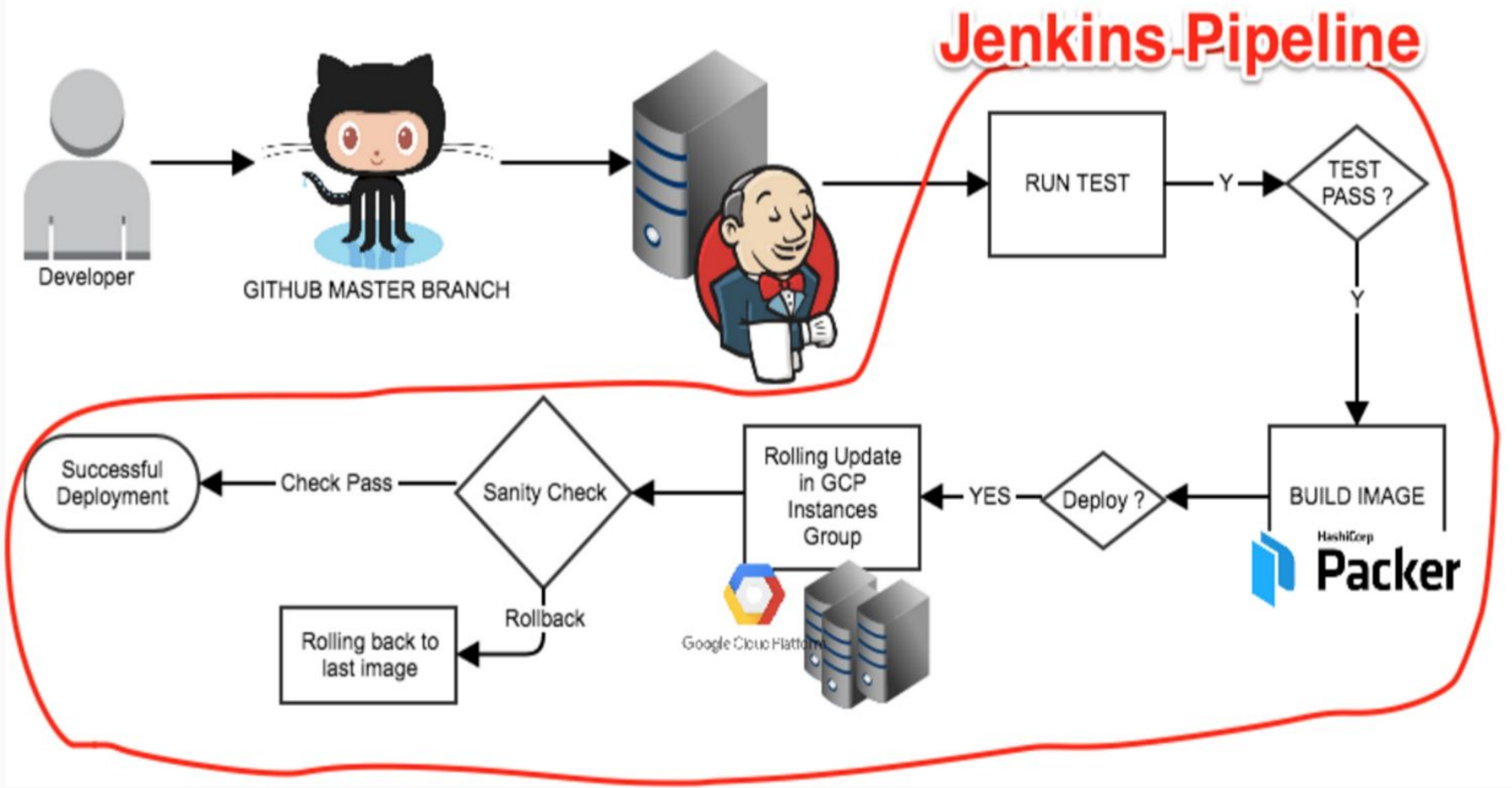
Entrega continua(CD)

Despliegue continuo(Cd)

INTRODUCCIÓN A JENKINS

¿Qué es Jenkins?

- Es un servidor de integración continua, gratuito, open-source y actualmente uno de los más empleados para esta función.
- Esta herramienta, proviene de otra similar llamada **Hudson**, ideada por Kohsuke Kawaguchi, que trabajaba en Sun.
- La base de Jenkins son las tareas, donde indicamos qué es lo que hay que hacer en un build. Por ejemplo, podríamos programar una tarea en la que se compruebe el repositorio de control de versiones cada cierto tiempo, y cuando un desarrollador quiera subir su código al control de versiones, este se compile y se ejecuten las pruebas.

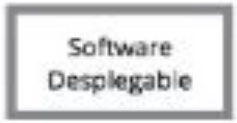

Jenkins-Pipeline
Developer
GITHUB MASTER BRANCH
RUN TEST
Y
TEST PASS ?
Y
BUILD IMAGE
HashiCorp
Packer
Deploy ?
YES
Rolling Update in GCP Instances Group
Google Cloud Platform
Sanity Check
Check Pass
Successful Deployment
Rollback
Rolling back to last image

Integración Continua

Integración Continua

Para ponerlo de la forma más sencilla, la integración continua es cuando un desarrollador puede integrar sus cambios, de manera continua (diferente a frecuente), en el repositorio de código. Y con integrar quiere decir que los cambios introducidos no dañen ni afecten el código existente ni que el mismo programador tenga que intervenir sustancialmente en el proceso.

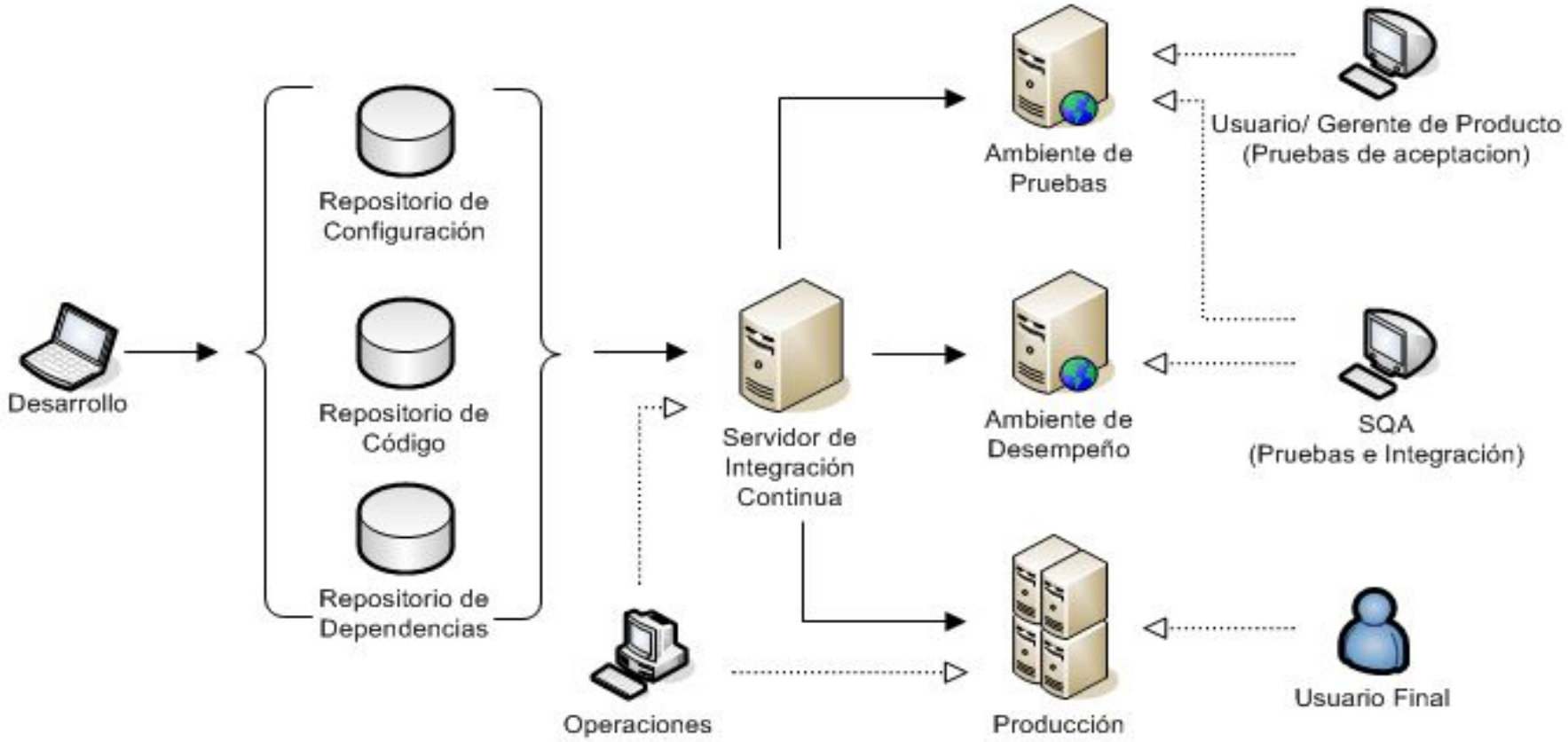
Entrega Continua



Entrega Continua

Esta vendría siendo el siguiente paso luego de la integración continua. De una forma sencilla implica que todo cambio subido al repositorio y cuyos tests sean exitosos, pasarán a un servidor donde el conjunto de todos los cambios (pueden ser 1 o más, de diferentes desarrolladores), sean compilados, probados y verificados. Al final, el servidor de pruebas indica si las pruebas fueron buenas o no en cuyo caso el mismo sistema debe notificar del resultado a una integradora.

Despliegue Continuo

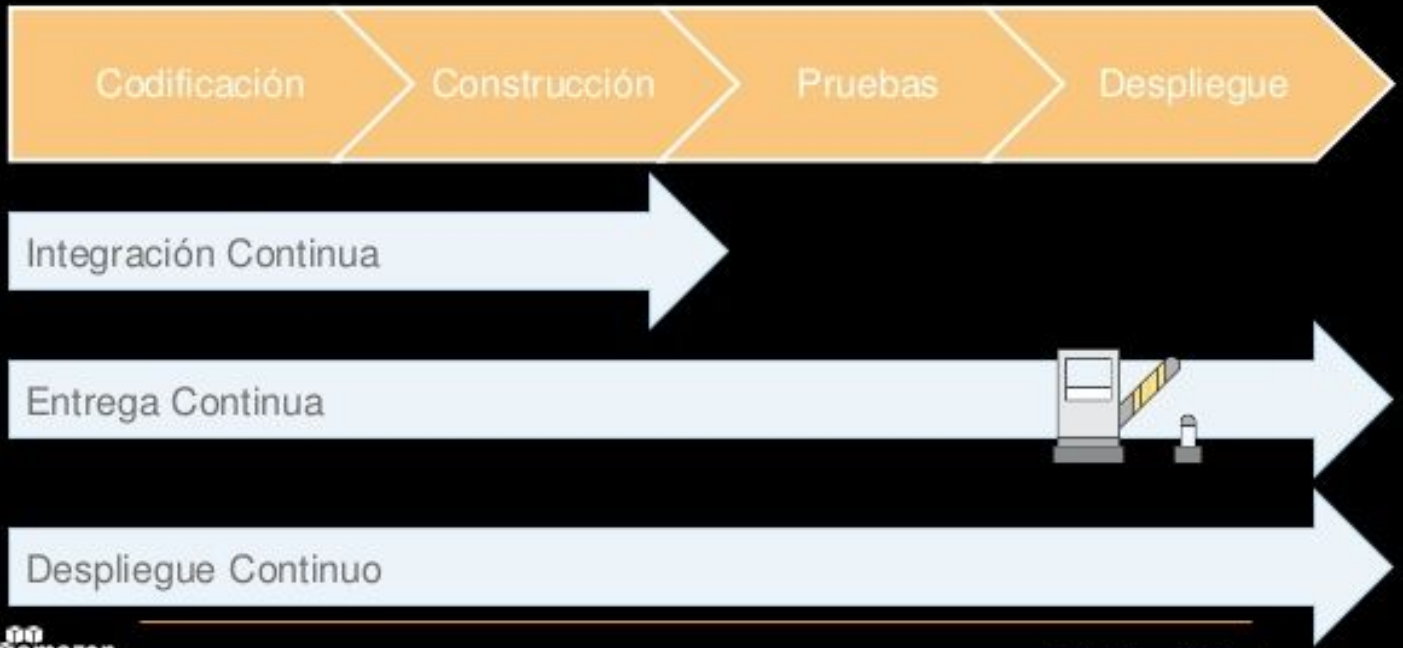


Despliegue Continuo

Con la entrega continua, todos los cambios en el código se crean, se prueban y se envían a un entorno de almacenamiento o pruebas de no producción. Pueden efectuarse varias pruebas al mismo tiempo antes de la implementación en producción. En el último paso, el desarrollador aprueba la actualización para su envío a producción cuando está listo. El proceso se diferencia de la implementación continua en que en el segundo caso el envío a producción se efectúa automáticamente, sin aprobación explícita.


Despliegue
Continuo
Despliegue
Automático
Integración Continua
Automatizaciones Básicas

Niveles del proceso de liberación de software



amazon web services

DevOps. ¿Qué aporta?

Valor

Comunicación

Enfocado en la mejora de la comunicación de los equipos de Desarrollo, Operaciones y QA.

Valor

Entrega continua

Maximiza el lanzamiento de versiones y servicios a negocio mejorando la entrega continua reduciendo así el Time To Market

Valor

Reduce el MTTR (Main Time to Repair)

Mejora el diagnostico y resolución de incidencias.