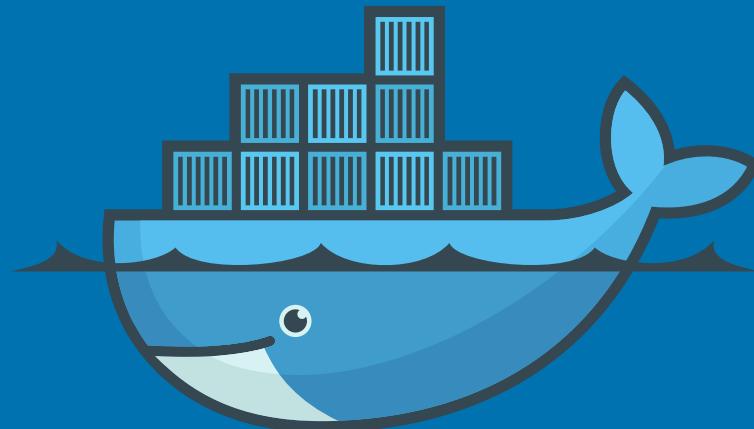


curzona



docker

Virtualización por contenedores

Ing. Sergio Méndez
IT Architect

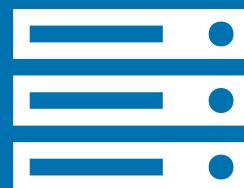
FIUSAC
Abril 2016

Indice

- 1. Qué es virtualización?
- 2. Tipos de virtualización
- 3. Hipervisor
- 4. Virtual Machine
- 5. Container
- 6. Arquitectura Monolítica
- 7. Solución Stack
- 8. Problemas de Arq. Monolítica
- 9. Microservicios
- 10. Principios de Microservicios
- 11. Devops
- 12. Ecosistema contemporáneo
- 13. Integración Continuada
- 14. Entrega Continua
- 15. Despliegue continuo
- 16. Fases del despliegue continuo
- 17. Qué es Docker?
- 18. Componentes de Docker
- 19. Comandos básicos
- 20. Ventajas de usar containers
- 21. Software Configuration Management
- 22. Herramientas de terceros
- 23. Alianzas
- 24. Tendencias
- 25. Video real
- 26. Demostración/Preguntas

Qué es virtualización?

Es el conjunto de componentes o metodología que permite la división de recursos hardware de una computadora en múltiples ambientes de ejecución, aplicando uno o más conceptos o técnicas como el particionamiento de hardware y software, time sharing, parcial o completa simulación o emulación de un arquitectura de hardware, Quality of Service y otros.



Tipos de virtualización

Full Virtualization:

Técnica de virtualización que provee una simulación completa del comportamiento de hardware en una máquina virtual.

Paravirtualization:

Técnica de virtualización que provee una simulación parcial del comportamiento de hardware en una máquina virtual.

***Operating System Virtualization:**

Virtualización basada en una simple instancia de sistema operativo.

Native o Híbrida:

Mezcla la Full y paravirtualization, combinando aceleración de E/S

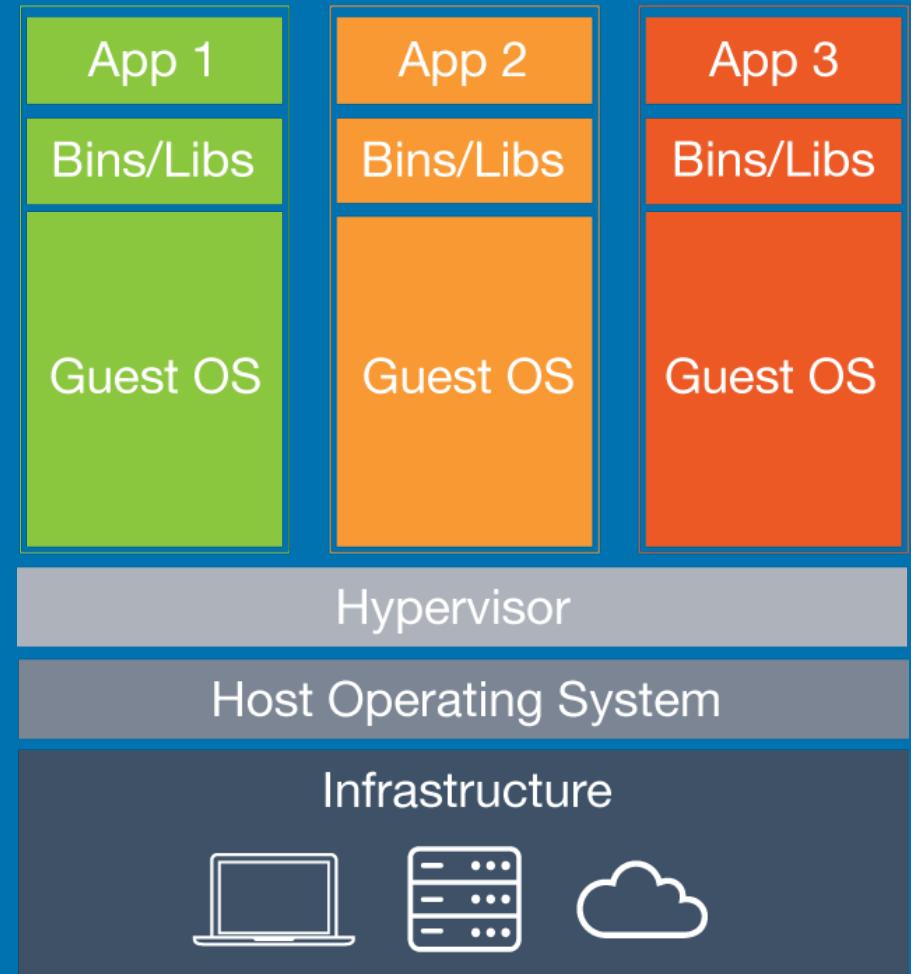
Hipervisor

Es el programa encargado de crear y administrar máquinas virtuales(Guests) en una máquina Host. El uso de hipervisores es exclusivo en técnicas de virtualización como la Full y Paravirtualization.

Existen 2 tipos los cuales varían en rendimiento y las capacidades de emulación y simulación del comportamiento de una arquitectura de hardware.

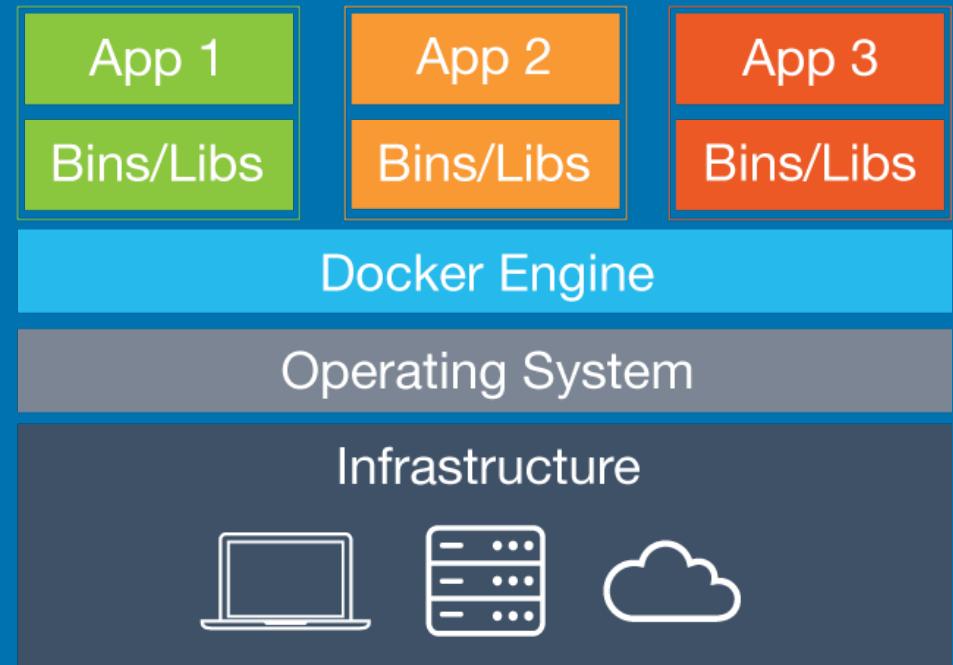
Virtual Machine

Cada máquina virtual incluye su aplicación, las librerías y binarios necesarios y el sistema operativo guest, las cuales pueden ocupar Gigas en tamaño.



Container

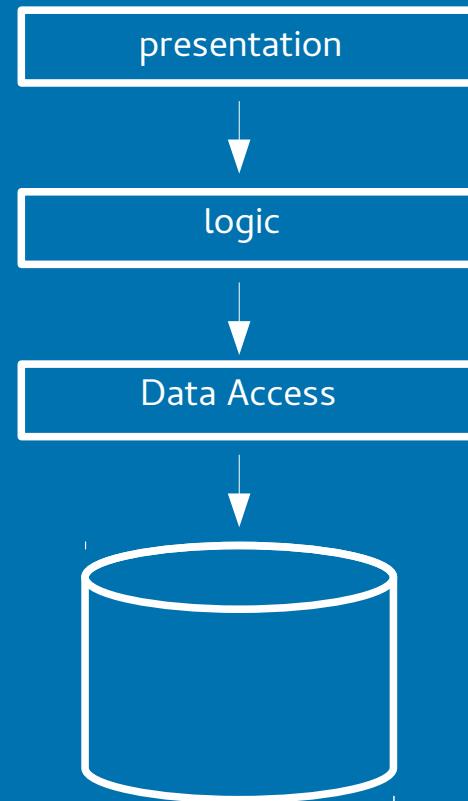
Un Container incluye la aplicación y todas sus dependencias, y comparte su kernel con otros containers. Se ejecutan como un proceso aislado en un userspace del SO. No están atados a ninguna infraestructura, simplemente corren en cualquier computadora o infraestructura en la nube.



Arquitectura Monolítica

Es una arquitectura de software que contiene una unidad de código final, despliegue y tecnologías stack o capas.

*Los sistemas por capas descomponen la arquitectura monolítica regularmente en 3 capas:
Presentación, lógica y datos.*



Solución Stack

Conjunto de aplicaciones que crean una plataforma en la cual puede ser ejecutada una solución de software.

Un stack común es:

LAMP= Linux + Apache + MySQL + PHP

Problemas de Arq. Monolítica

- Problemas de mantenimiento del sistema
- Complicaciones en la evolución del sistema y reemplazo de componentes con versión más reciente
- El desarrollo en capaz hace más difícil la escalabilidad de un sistema
- Crecimiento desordenado y limitado
- Escalabilidad eficiente pero limitada
- Los desarrolladores se vuelven especialistas en una capa
- Poca o ninguna documentación en sistemas diseñados en capaz

Microservicios

Un microservicio es un estilo de arquitectura que aprovecha el desarrollo de una aplicación simple a través de un conjunto de pequeños servicios.

Principio: Realizar una descomposición funcional para crear componentes independientes de despliegue.



Vertical Slicing

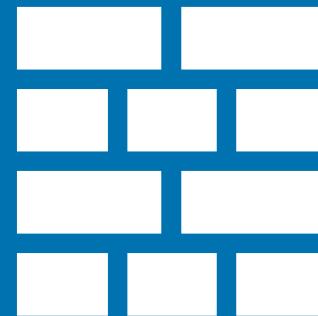
Principios de Microservicios

- Encapsulamiento
- Automatización
- Dominio del negocio
céntrico
- De-centralización
- Independencia
- Recuperación a fallos
- Observación

DevOps

Development, Operations, se refiere a una metodología de desarrollo de software que se centra en la comunicación, colaboración e integración entre desarrolladores de software y los profesionales de operaciones en las tecnologías de la información (IT).

Quiero cambios!



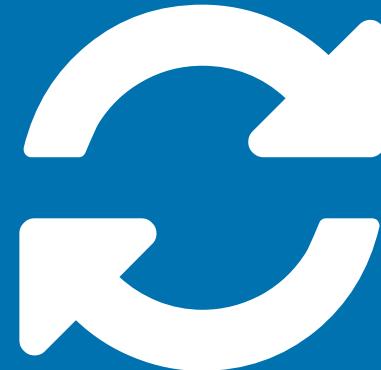
Quiero estabilidad!



Muro de la confusión

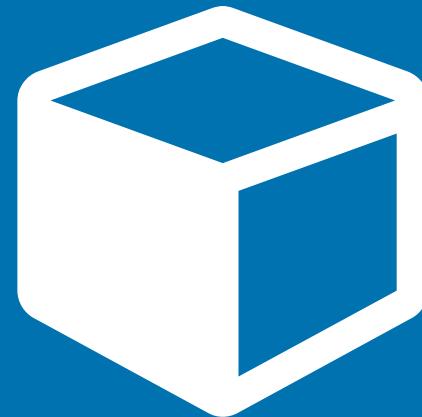
Integración Continua

Es el proceso que nos permite de manera automática, construir los paquetes de código y ejecutar los test necesarios para cumplir con los criterios de calidad establecidos, además de comprobar que el nuevo código funciona perfectamente con el resto de piezas del sistema.



Entrega continua

Es el proceso que incluye a Continuos integration y que permite de manera automática, crear una release del código que hemos creado o modificado. Lo que obtenemos es un entregable preparado para desplegarlo en producción.

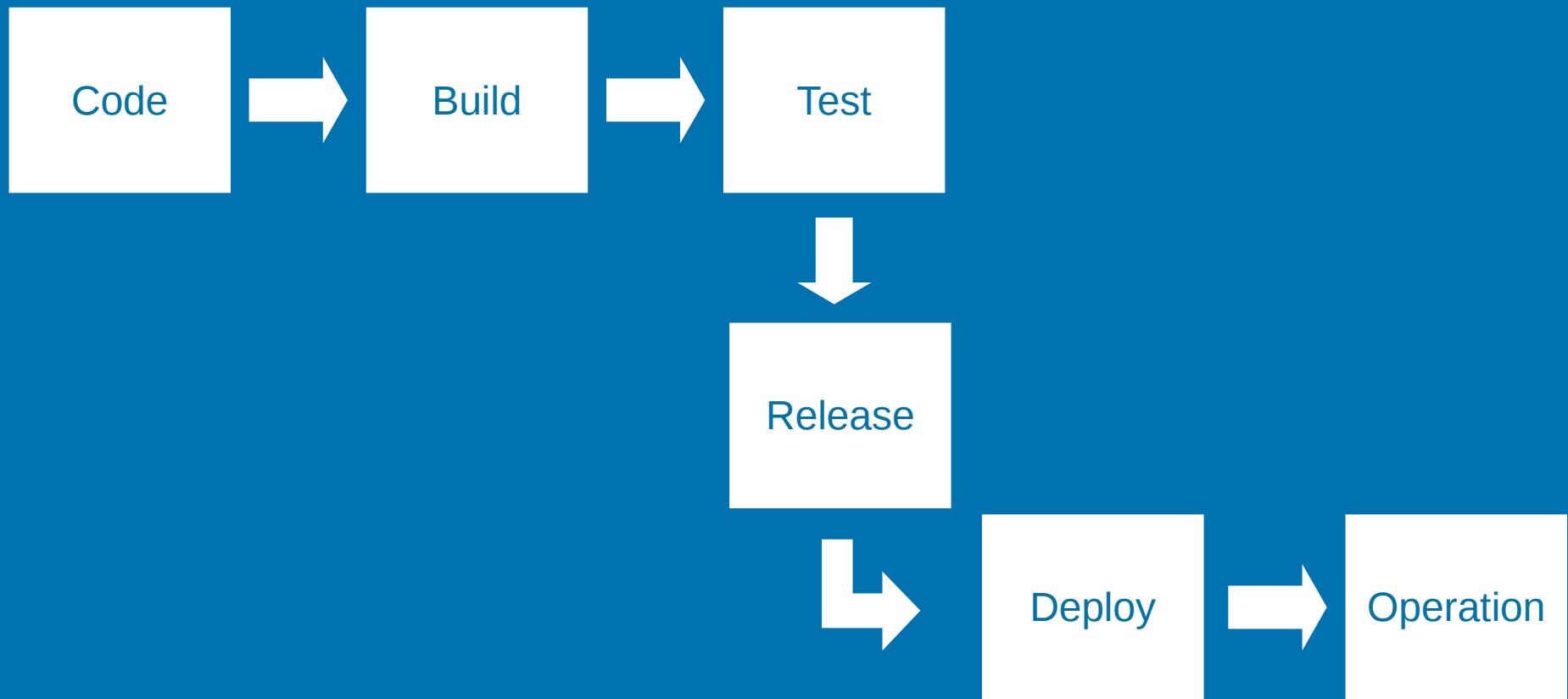


Despliegue Continuo

Es el proceso por el cual permitimos que todo el código escrito de una aplicación, se pueda subir a producción de manera automática, pasando por las distintas fases de su ciclo de vida.

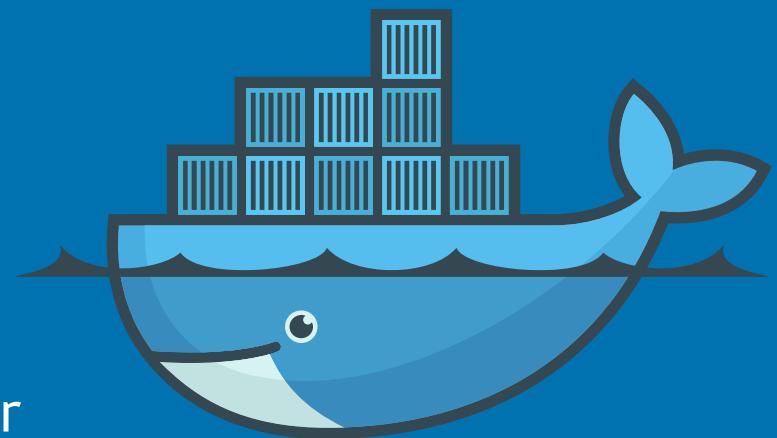


Fases del despliegue continuo

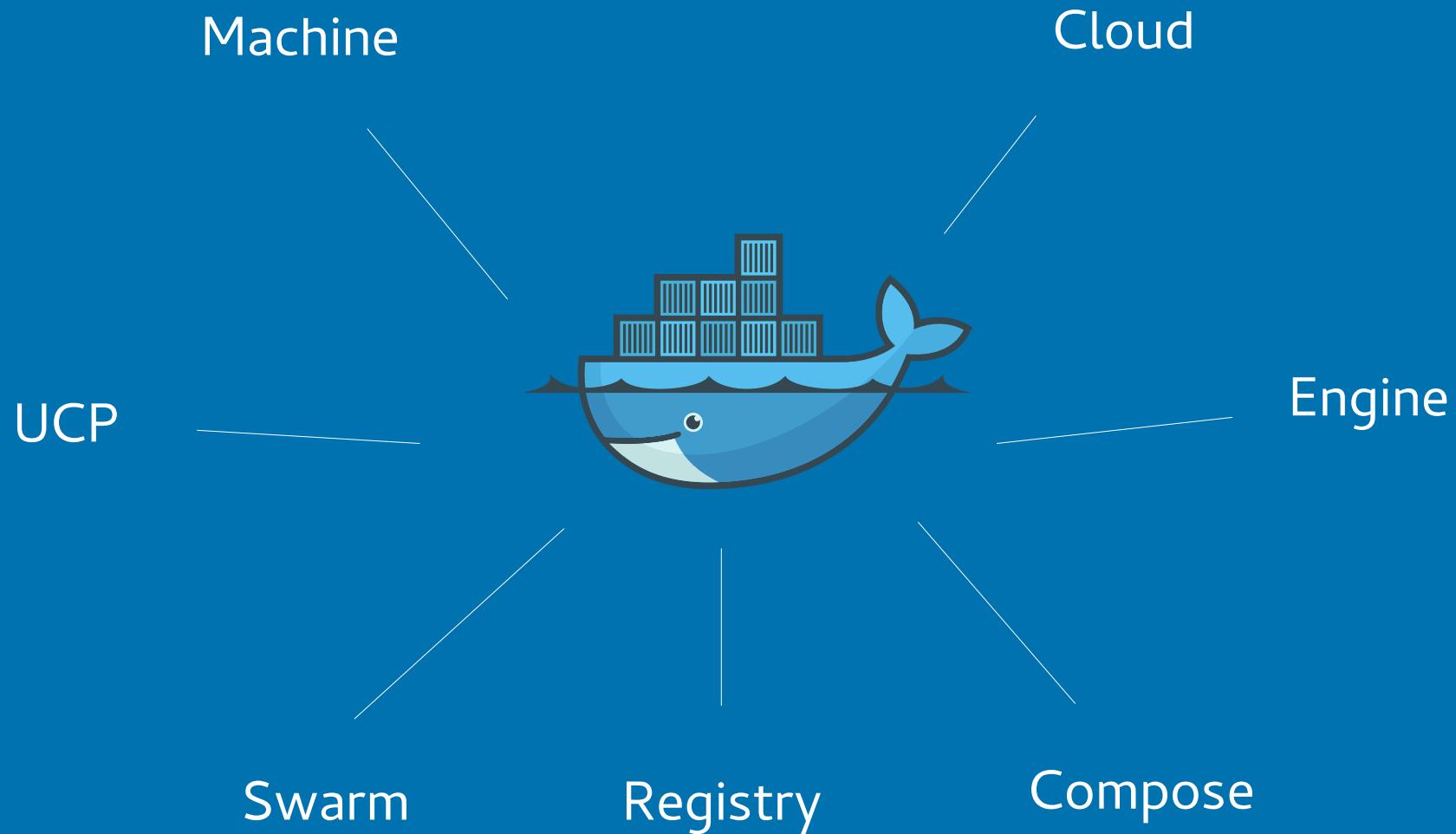


Qué es Docker?

Docker es un programa que permite envolver una pieza de software en un sistema de archivos completo que contiene todo lo que necesita para ejecutar: código, ambientes de ejecución, herramientas de sistema, librerías todo lo que se necesite instalar en un servidor, esto siempre garantiza que se ejecutara de la misma forma en cualquier ambiente en el que se ejecute.



Componentes de Docker



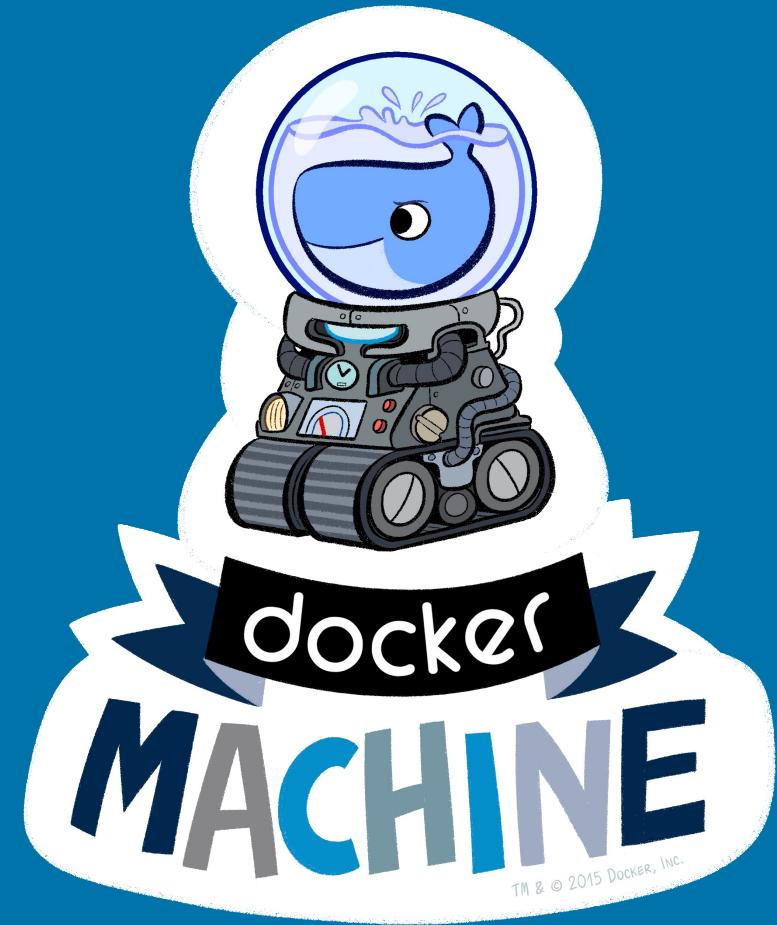
Docker Registry

Docker Registry provee una forma de almacenar imágenes de contenedores de forma publica y privada.



Docker Machine

Docker Machine automatiza el provisionamiento de contenedores a través de tu la red o de la nube.



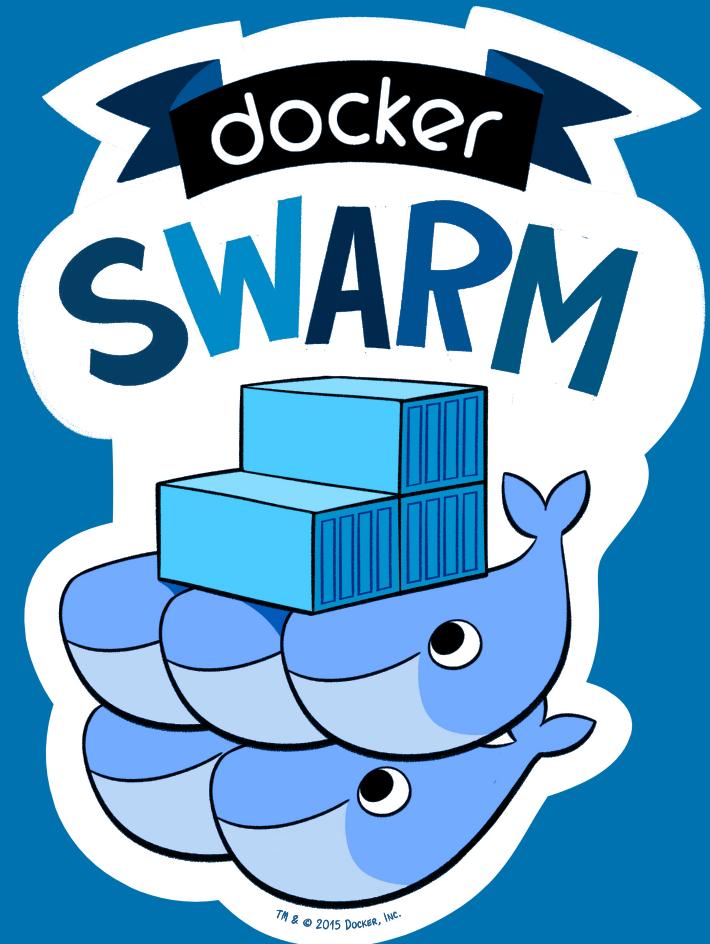
Docker Compose

Docker Compose define aplicaciones multi contenedores.



Docker Swarm

Docker Swarm permite realizar clusters
y programar contenedores.



Docker Experimental

Son las nuevas herramientas que sigue desarrollando el equipo de Docker.



Comandos básicos de docker

docker search [image]

docker pull [image]

docker ps -l

docker kill

docker inspect id_container

docker run [image] [command]

docker commit id_container [name]

docker push

docker start -a [container]

docker network ls

docker network create [net]

Ventajas de usar containers

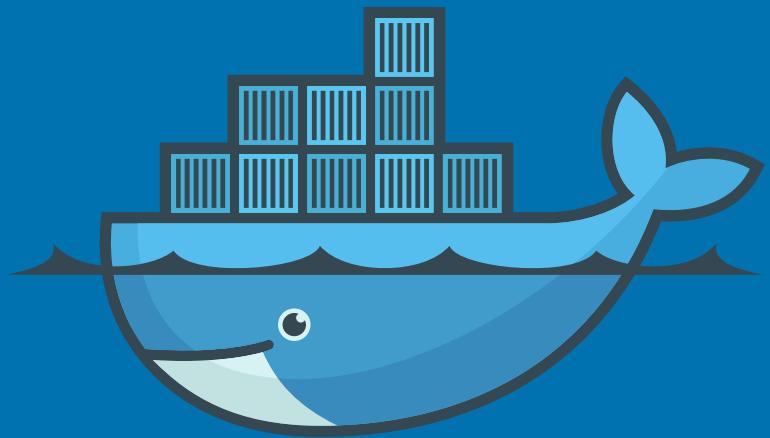
- No se necesita un hardware especial
- Las aplicaciones son ejecutadas con la misma velocidad de la máquina física
- Movilización de contenedor más rápida y efectiva
- Reducción de brecha entre Developers y Operators
- Escalabilidad de forma más sencilla

Software Configuration Management

Docker permite llevar el control de las imágenes de la misma forma que versionamos código.

Está disponibles las opciones del uso de tags, commit y nombres de imágenes.

Todo esto relacionado con el Software Configuration Management a nivel DevOps.



Herramientas de terceros

Mesosphere

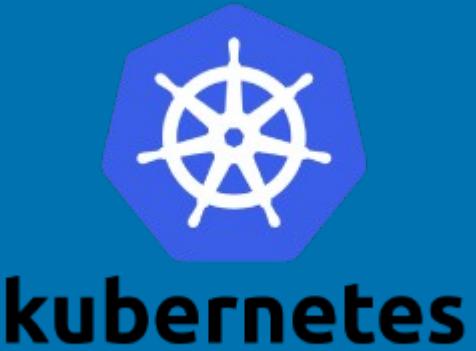
Es una plataforma open source que permite escalar datacentros en ambientes de producción y la creación de servicios tipo stateful.



MESOSPHERE

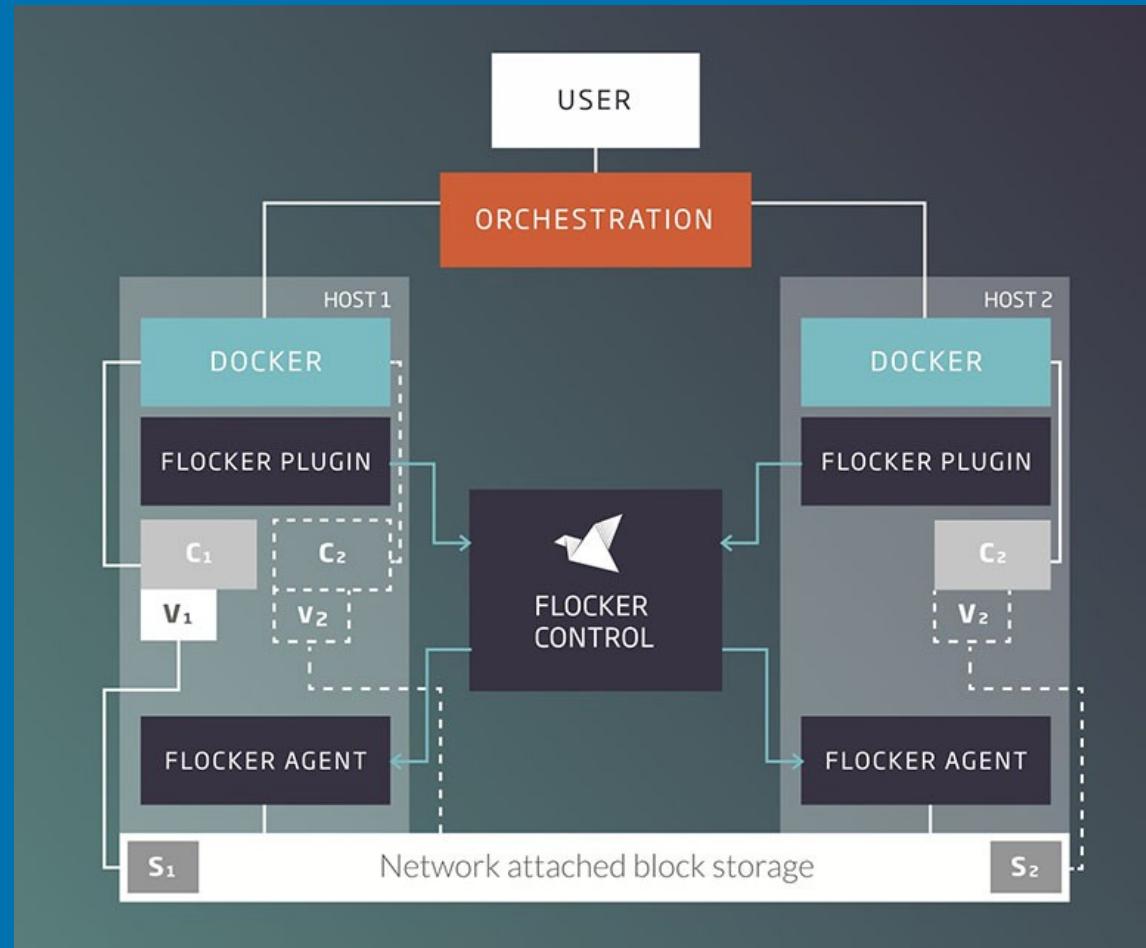
Kubernetes

Kubernetes es un sistema open source que permite la automatización de despliegue, operaciones y escalar aplicaciones en contenedores.

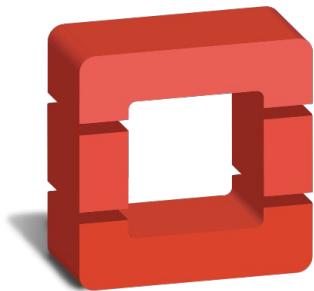


Flocker

Flocker es un administrador open source de volúmenes de datos para contenedores para aplicaciones Dockerizadas.



Alianzas

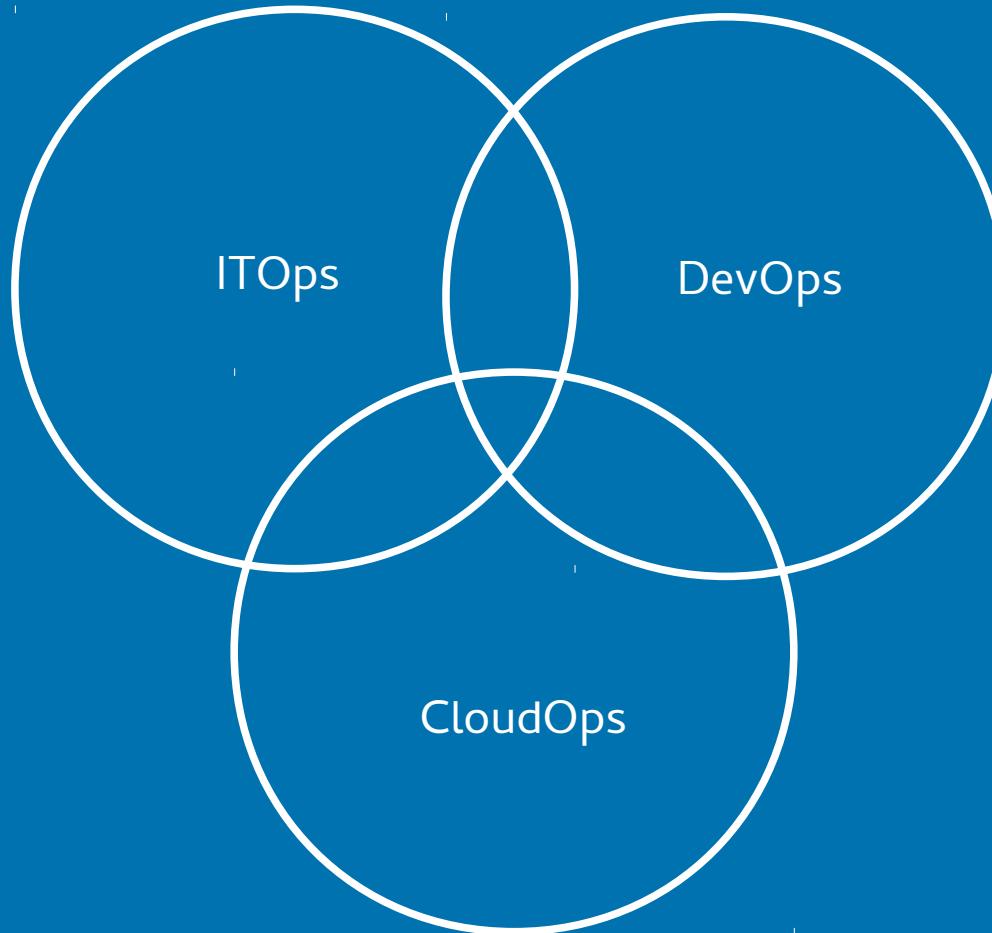


openstack[®]
CLOUD SOFTWARE



DigitalOcean

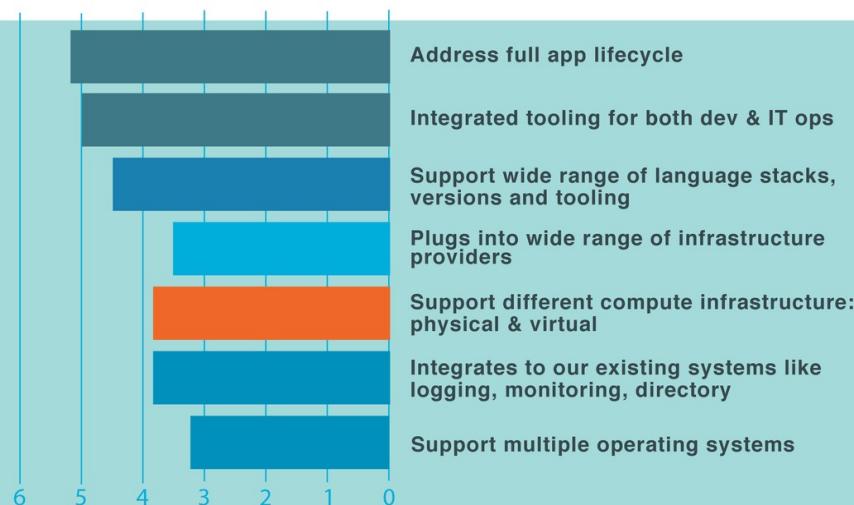
Ecosistema contemporáneo



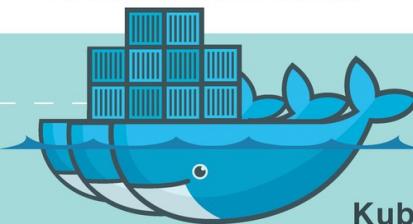
El reto es orquestar el sistema

Tendencias

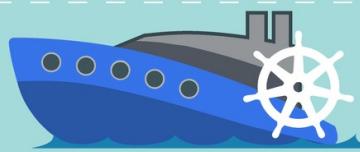
Ranked Platform Requirements



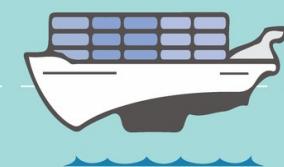
Docker Swarm 34%



Kubernetes 32%



Amazon ECS 29%



5X faster than Kubernetes
to spin up a new container*

7X faster than Kubernetes
to list all the running containers*

*Evaluating Container Platforms at Scale, medium.com/on-docker/evaluating-container-platforms-at-scale-5e7b44d93f2c



Tendencias

65%
of orgs have
challenges
maintaining
legacy apps

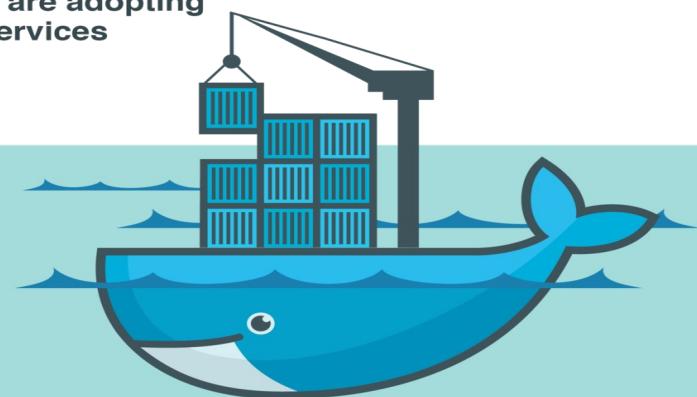


59%
of orgs have
challenges from
inertia of legacy apps
and infrastructure

39%
of orgs are modernizing
legacy apps



44%
of orgs are adopting
microservices



78%
are using, or planning to
use, Docker to build new
microservices applications.



71%
are using, or planning to
use, Docker to containerize
a legacy app.



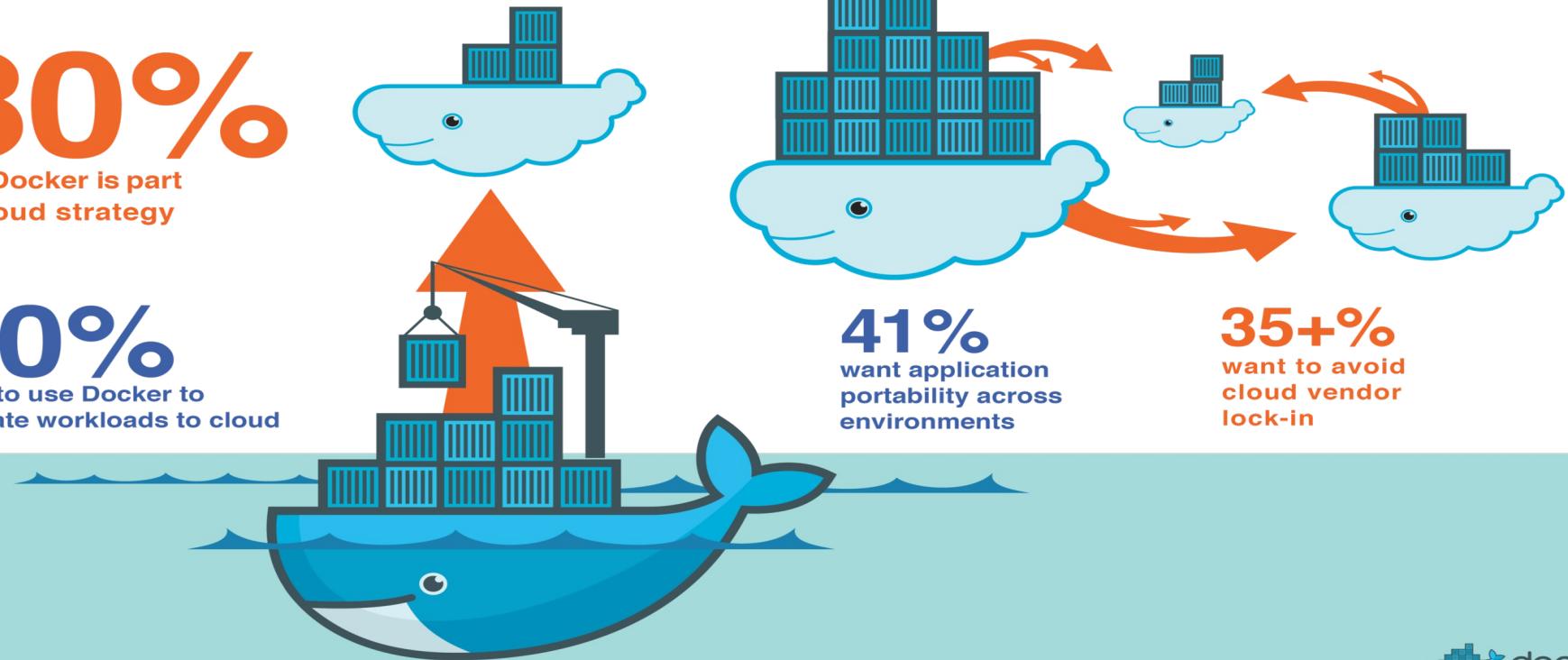
Tendencias

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



Tendencias

65%

use Docker to deliver development agility.

48%

use Docker to control app environments.

41%

use Docker to achieve app portability.

90%

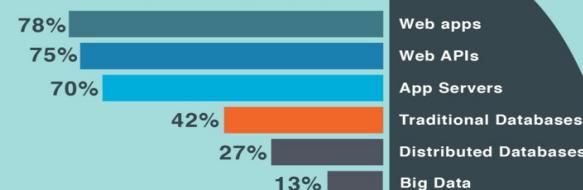
use Docker for apps in development.



58%

use Docker for apps in production.

Docker Workloads



90%

plan dev environments around Docker.



80%

plan DevOps around Docker.



Tendencias

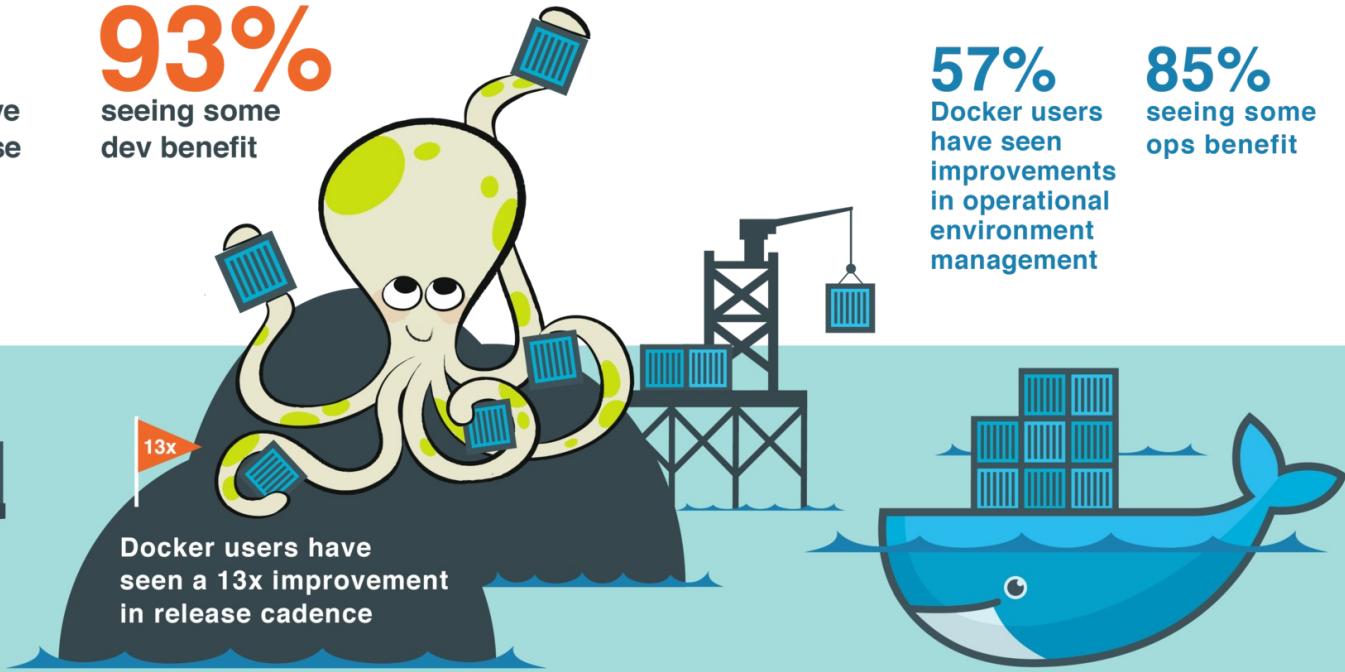
45%

of Docker users have been able to increase the frequency of software releases



93%

seeing some dev benefit



57%

Docker users have seen improvements in operational environment management

85%

seeing some ops benefit



70%

of Docker users say '*Docker has dramatically transformed... etc*

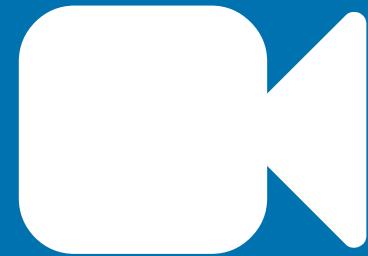


62%

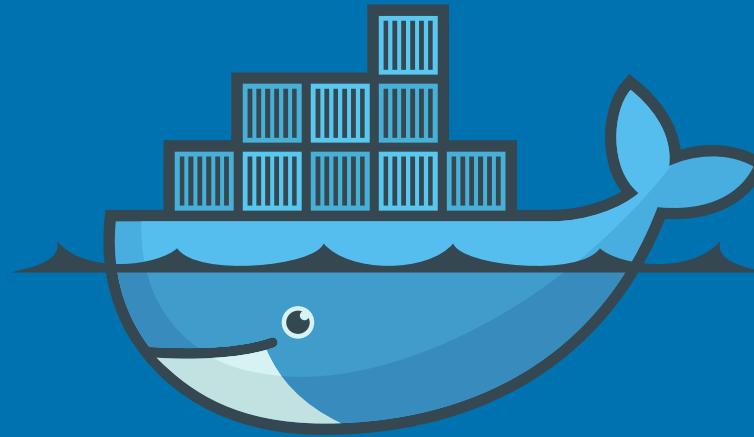
have seen improved MTTR on software issues.



Video caso real



Demostración



Preguntas



Contacto



sergioarm@curzona