# **DevOps:** primeros pasos

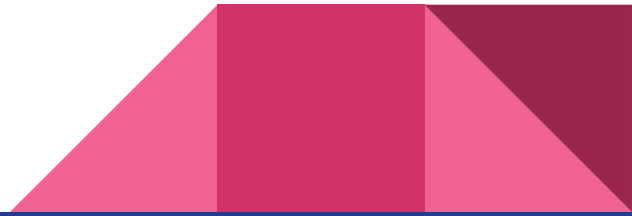
Club de Investigación Tecnológica

Allan Cascante
Martín Flores

27 de Julio. 2016

# Agenda

- La idea detrás de *DevOps*
- Beneficios
- Prácticas para lograr ser *DevOps*
- Conclusiones



# Operaciones

Administradores de sistemas

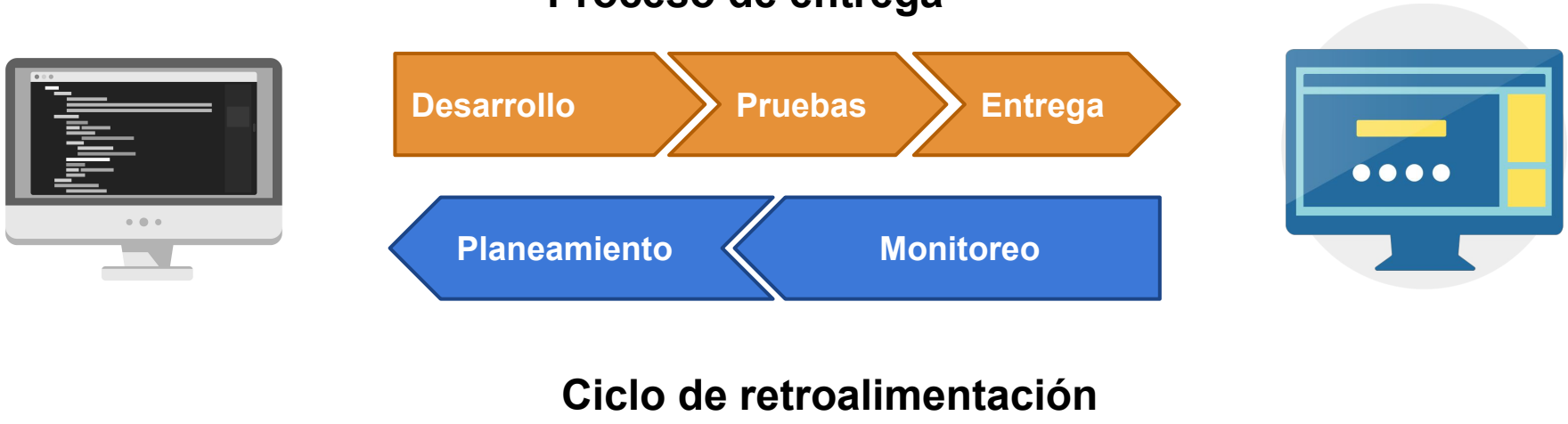
Administradores de servidores

Sysadmins

Infraestructura “Infra”

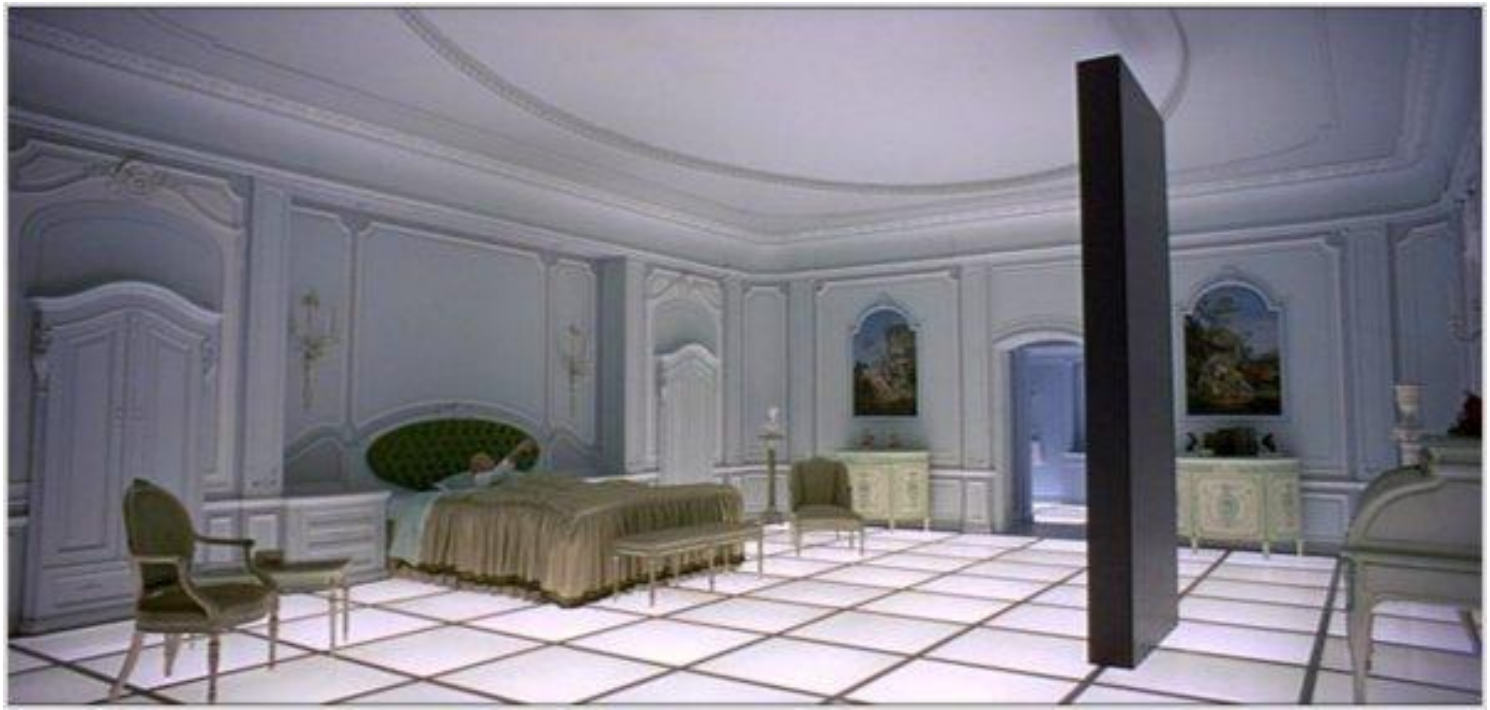
...

# Ciclo de desarrollo de software




***DevOps***: mejoras que hacen que este ciclo sea más rápido.

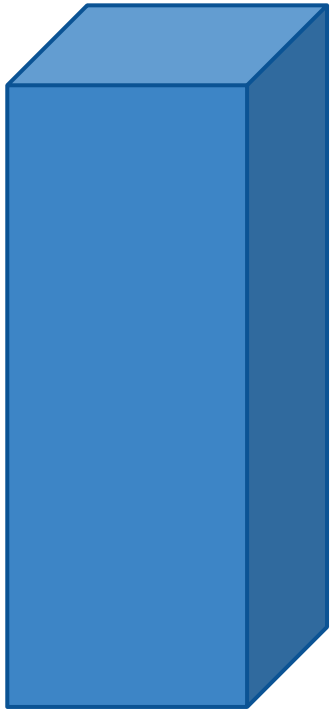
# La historia de la compañía X



2001 Odisea en el espacio.  El “*salto temporal más largo de la historia del cine*”

# Ciclo de vida de desarrollo del monolito

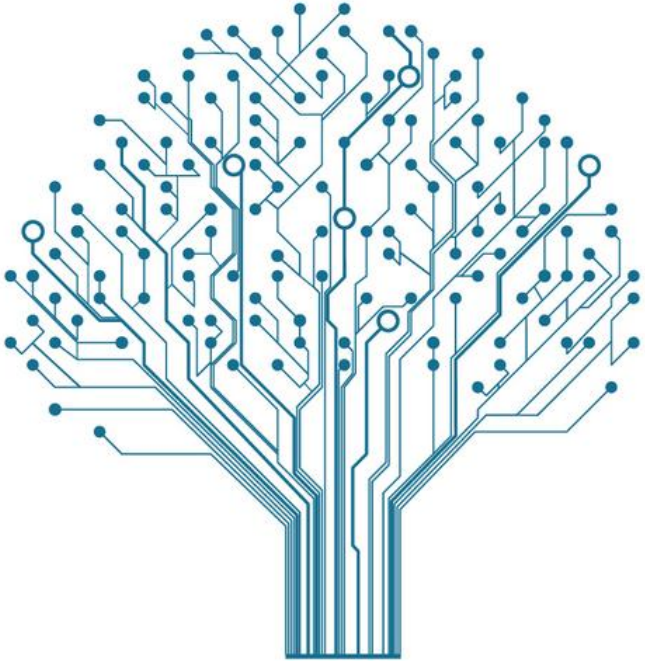
Desarrolladores



Aplicación

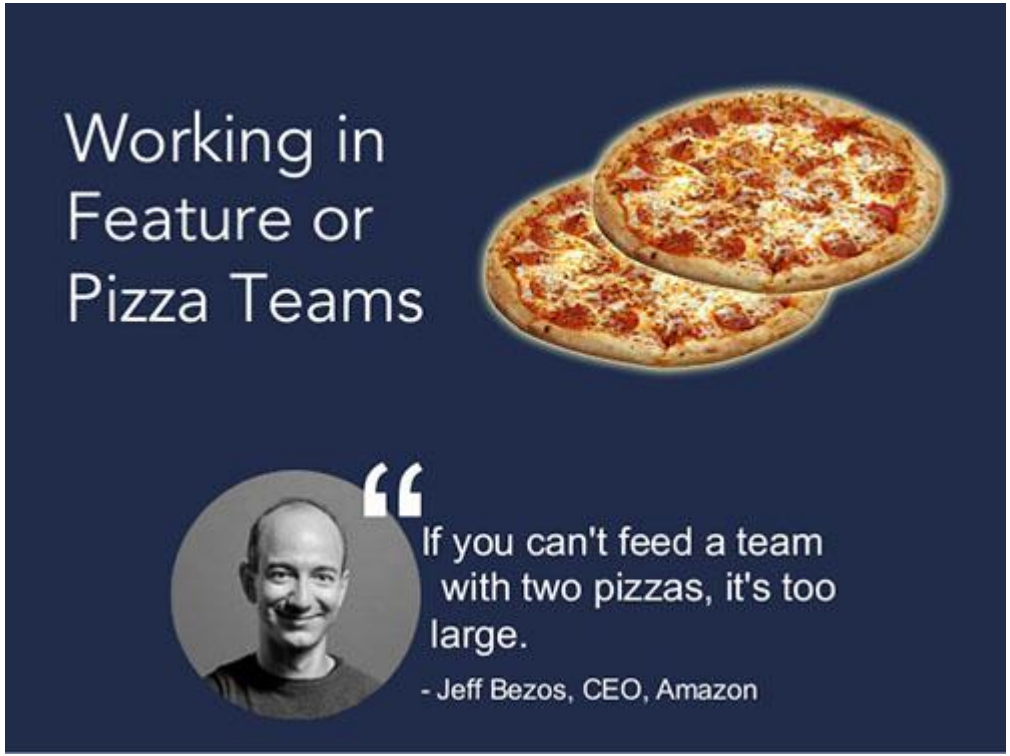
# Cambios en arquitectura

- Servicios altamente desacoplados
- Una sola responsabilidad
- Conectados a través de *APIs*
- *Microservicios*

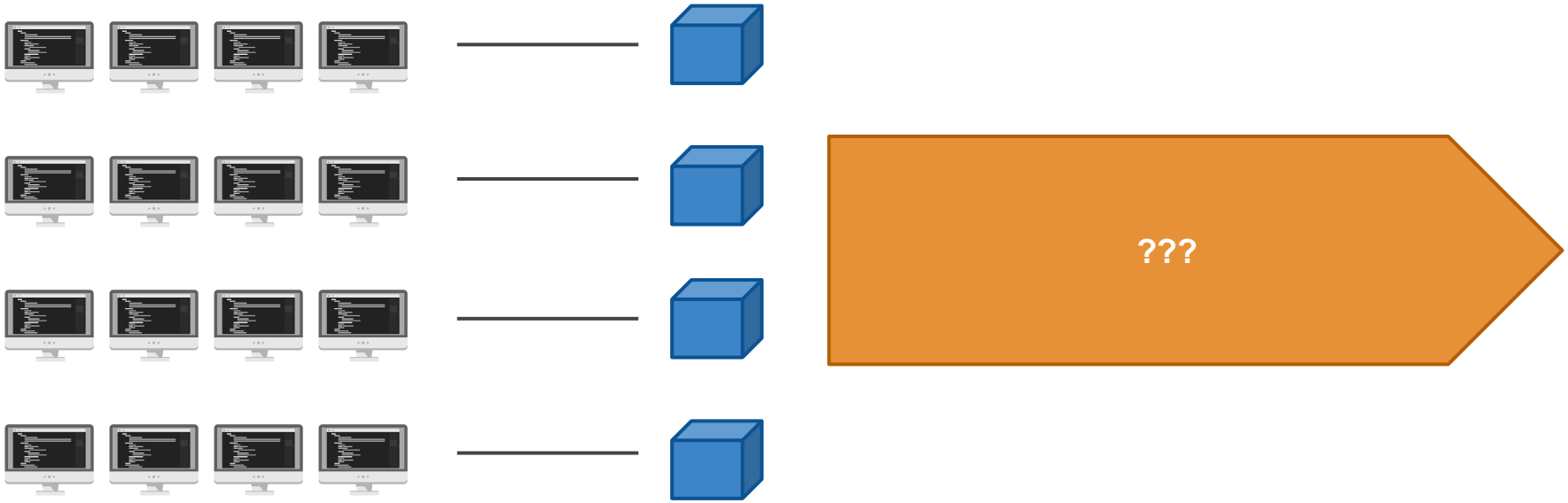


# Cambios organizacionales

- Responsabilidad total
- Autonomía total
- Comparten los mismos incentivos y objetivos
- *DevOps*



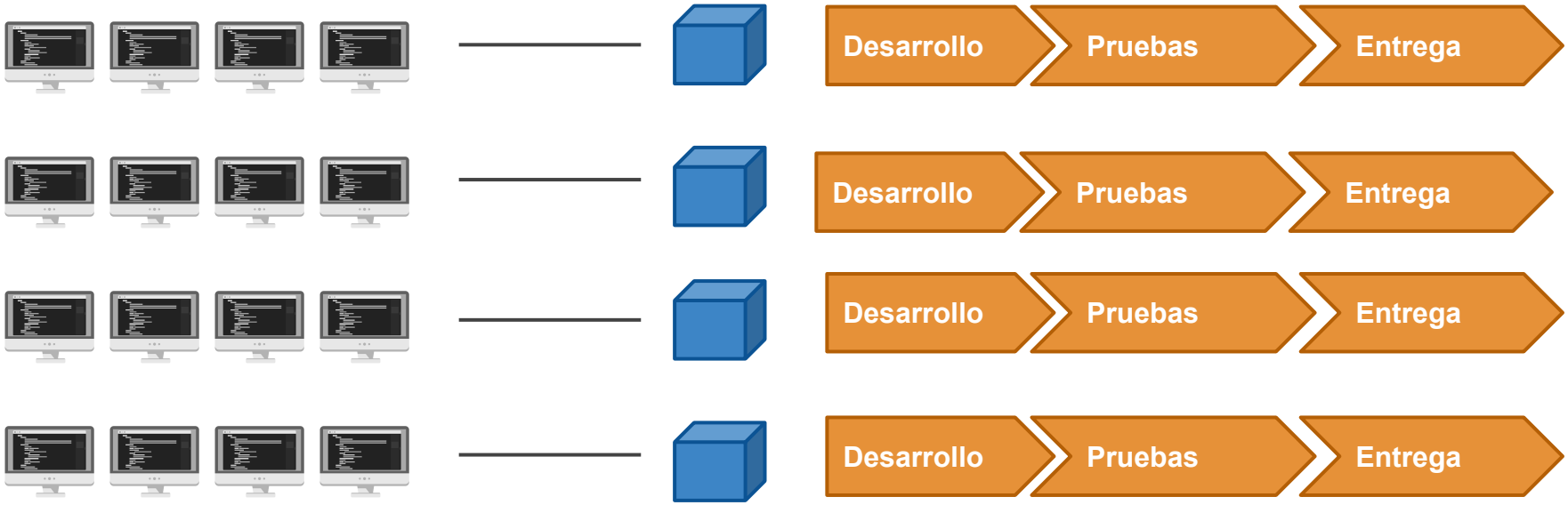
# Hubo mejora pero...



# Mejoras en el proceso de entrega

- Se crearon herramientas para acelerar el proceso
  - *ChatOps*
- Ambientes de producción independientes
- Creación de “tuberías de entrega” (*delivery pipelines*)
- **Automatización**

# Ciclo de vida de los microservicios




**Desarrolladores** **Servicios** **Proceso de entrega**

# Amazon.com (2014)

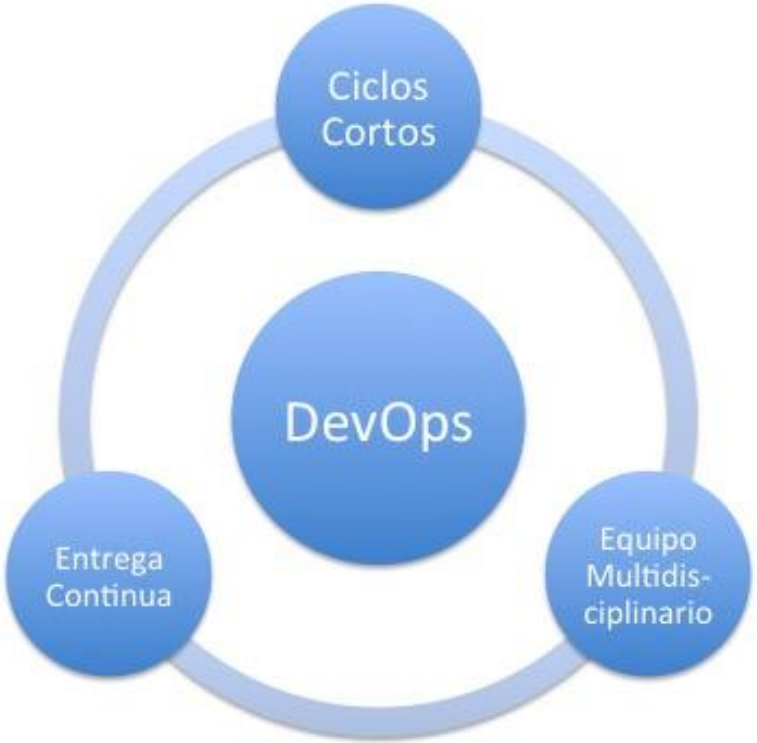
Miles de equipos
**x** Arquitectura basada en microservicios
**x** Entrega continua
**x** Múltiples ambientes

---

**= 50 millones de *deployments* por año**

# DevOps

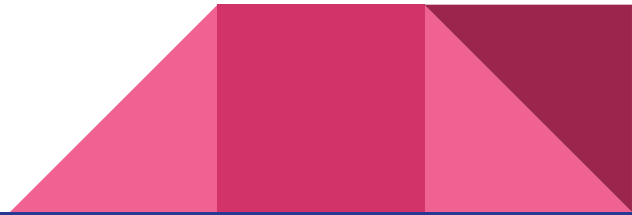

Desarrollo
Calidad
DevOps
Operaciones


Ciclos Cortos
DevOps
Entrega Continua
Equipo Multidis-ciplinario


oneServer

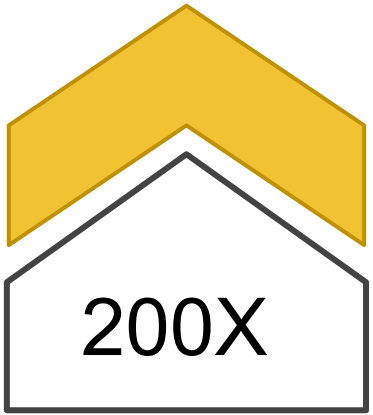
# Una aplicación moderna

1. Archivo de construcción y administración (maven, make, rake, npm ….)
2. Configuración de repositorio de código
3. Configuración de tareas de integración
4. Código fuente
5. Código para pruebas (unitarias, integración, carga)
6. Provisionamiento
7. Pruebas de aceptación



# *DevOps*: tiempos de entrega

2016 State of DevOps Report

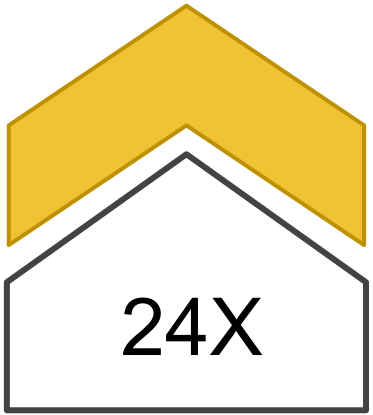



*Deployments* más frecuentes


Tiempos de espera más cortos

# *DevOps*: impacto de los cambios

2016 State of DevOps Report




Mayor rapidez en la recuperación de fallas


Tasa de errores

# *DevOps:* tiempos de entrega

2016 State of DevOps Report


Menos de tiempo utilizado arreglando de errores

Dado que este es un proceso de mejora continua, la estabilidad y calidad del producto se pone a prueba y se verifica en cada paso.

# Amazon.com

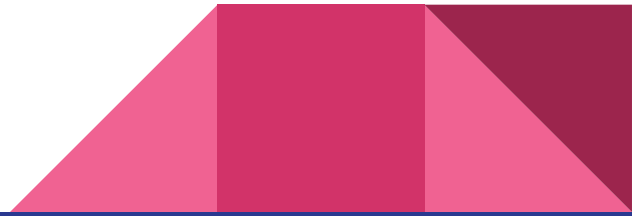
Pases a producción cada 11,6 segundos

# etsy.com

25 versiones nuevas diarias

# ¿Necesito *DevOps*?

1. Frecuencia en los pases a producción
2. Cantidad de cambios
3. Tiempo de espera
4. Porcentaje de pases a producción fallidos
5. Tiempo de recuperación
6. Cantidad de defectos reportados
7. Cambios en el uso de la aplicación
8. Disponibilidad
9. Tiempo de respuesta

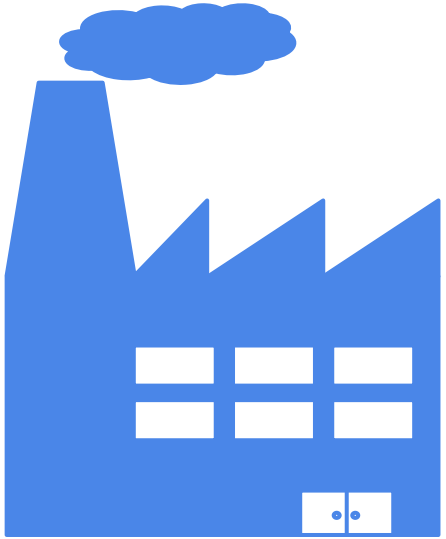


# ¿Cómo lograrlo?

Primeros pasos

# La fábrica de DevOps

1. El código como infraestructura
2. Integración continua
3. Pruebas automatizadas
4. Administración del desempeño de la aplicación
5. Entrega continua
6. Administración de la configuración



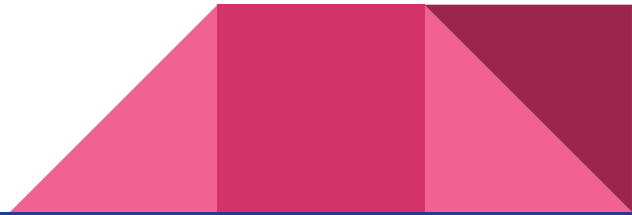
# El código como infraestructura

Infrastructure as Code (IaC)

# El código como infraestructura (IaC)

- Técnicas, procesos, herramientas para administrar y configurar la entrega de aplicaciones.
- Mayor control sobre el hardware, mantenimiento repetitivo, resultados predecibles.
- Se refina el ambiente de ejecución general de una aplicación.

- ***Anti-patrón***: scripts aislados, parches manuales, ambientes no sincronizados, pases a producción para ver qué pasa.

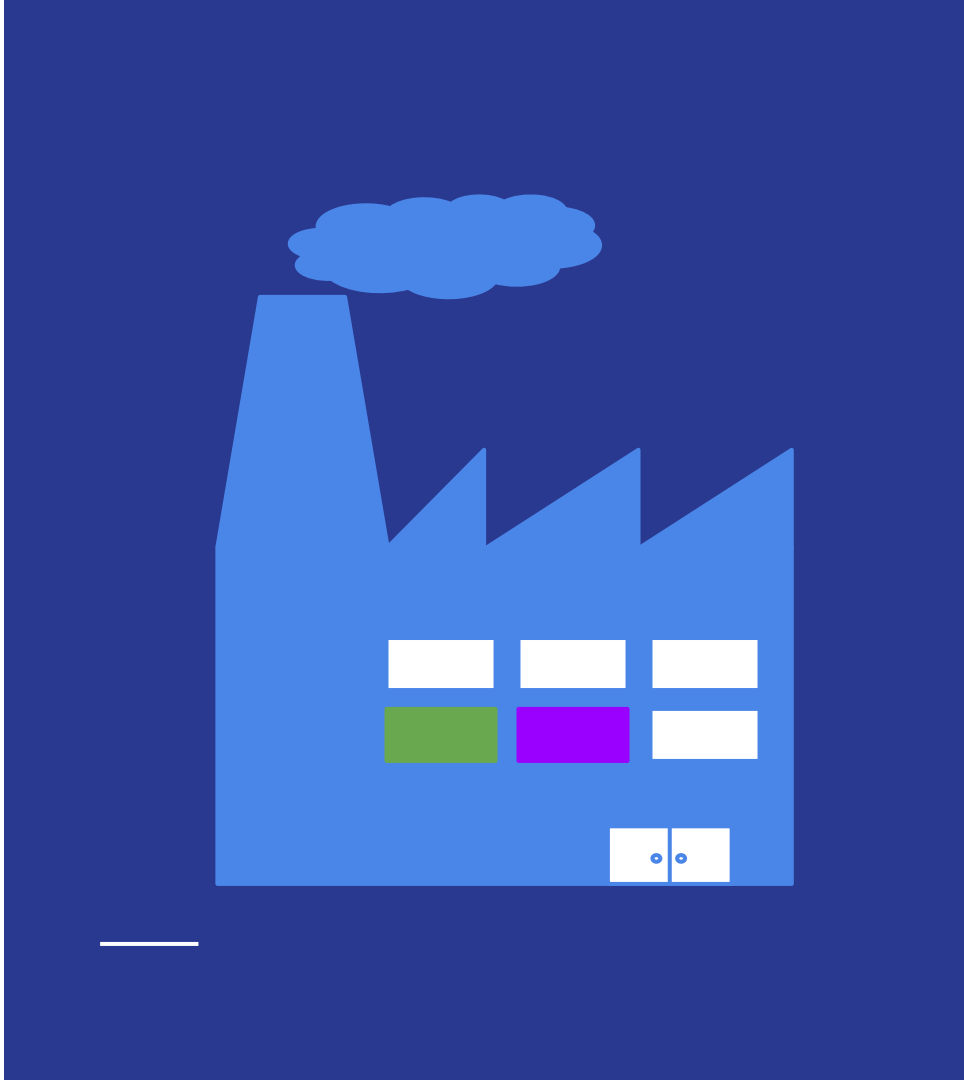


# IaC - Herramientas

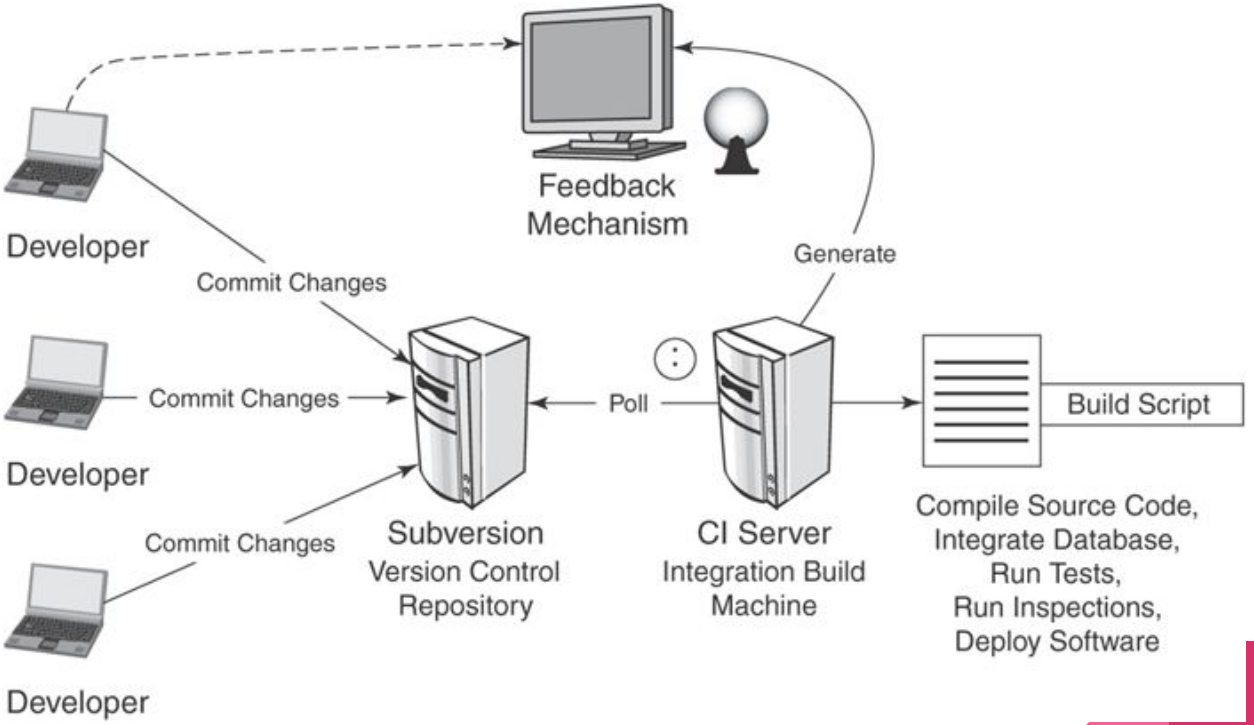
- Puppet (http:www.puppet.com)
- Chef (www.chef.io)
- Ansible (http:www.ansible.com)
- CFEngine (http://cfengine.com)
- Vagrant (http://vagrantup.com)
- Docker (http://www.docker.com)
- Herramientas provistas en servicios como Amazon AWS y Azure.

# Integración Continua

Continuous Integration (CI)



# Integracion continua




**Fuente:** *Continuous Integration: Improving Software Quality and Reducing Risk*

# CI - Beneficios

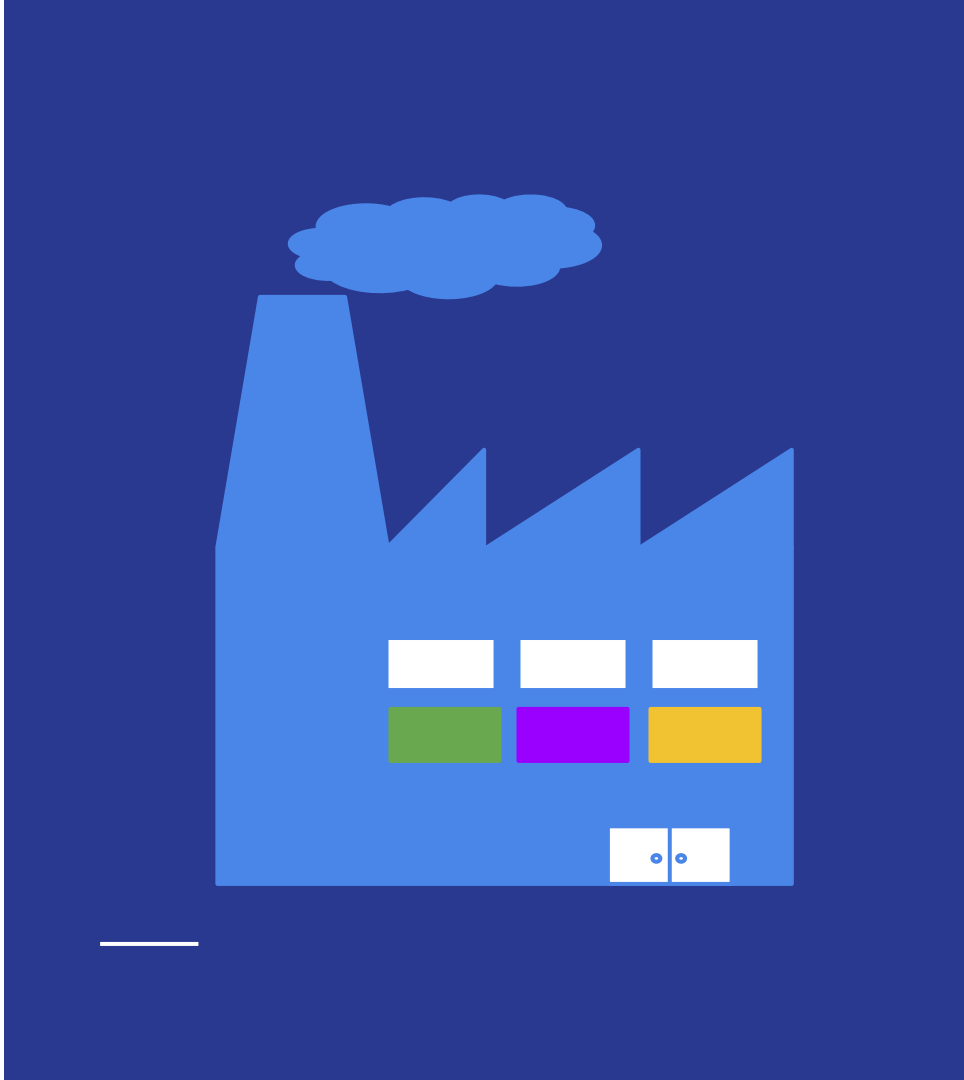
- Promover código constantemente
- Automatizar las pruebas (todo lo posible)
- Construcciones (*builds*) frecuentes.
- No promover código inestable
- Automatizar el despliegue
- Estrategia de versionamiento de artefactos de software
- Retroalimentación

# CI - Implementacion

- Sistema de control de versiones (Git, Subversion, Perforce, Mercurial)
- Herramienta de Integración Continua (Jenkins, TeamCity, CruiseControl)
- Construcciones que se disparan automáticamente
- Análisis estático de código (Sonarqube)
- Automatizar todo lo que se pueda
- Administración de artefactos de software (artifactory, nexus, npm)
- Pruebas en ambientes iguales a los de producción

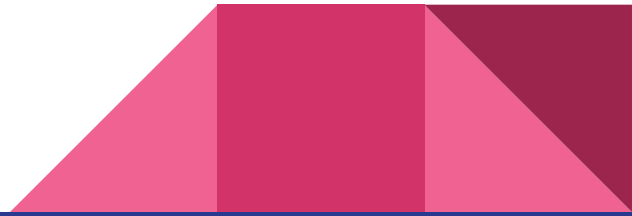
***¿Anti-patrón?***

# Pruebas automatizadas



# Pruebas automatizadas

- Realizar pruebas al software es crítico pero es una tarea que puede consumir mucho tiempo.
- Respuesta: un conjunto de pruebas automatizadas.
- Proveen visibilidad del rendimiento de la aplicación y del impacto de sus cambios.
- Equipos pueden entregar nuevas versiones de software más rápido sin comprometer la calidad.

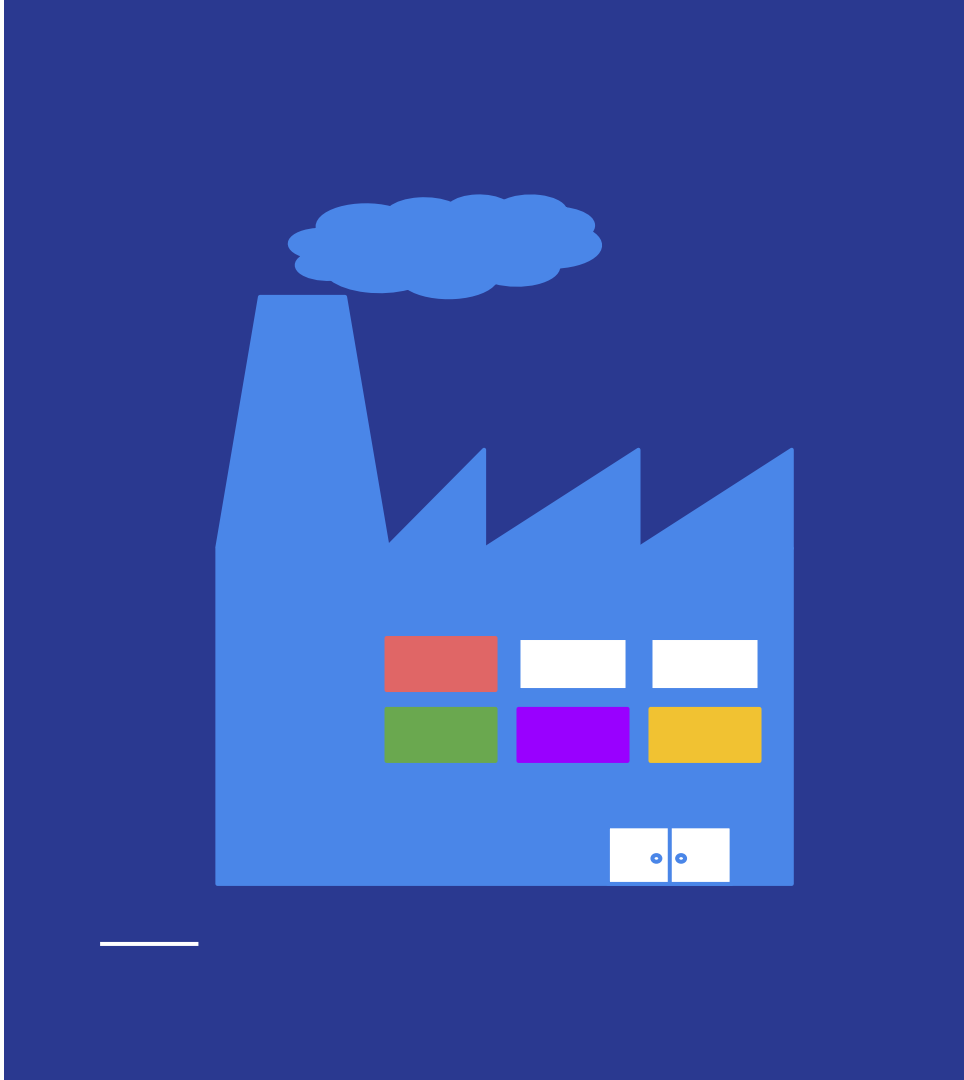


# Pruebas automatizadas

- QA promueve el uso de herramientas y frameworks que permiten probar de forma prematura el software durante el ciclo de desarrollo.

- ***Anti-patrón***: exceso de pruebas manuales, pruebas tardías, despliegues a producción para ver qué pasa.

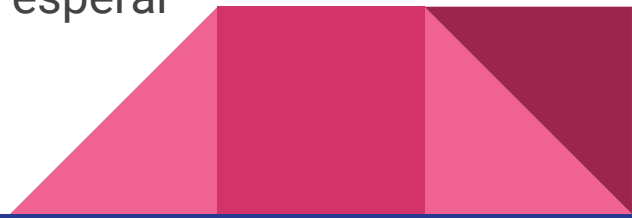
# Gestión del desempeño



# Gestión del desempeño - Monitoreo

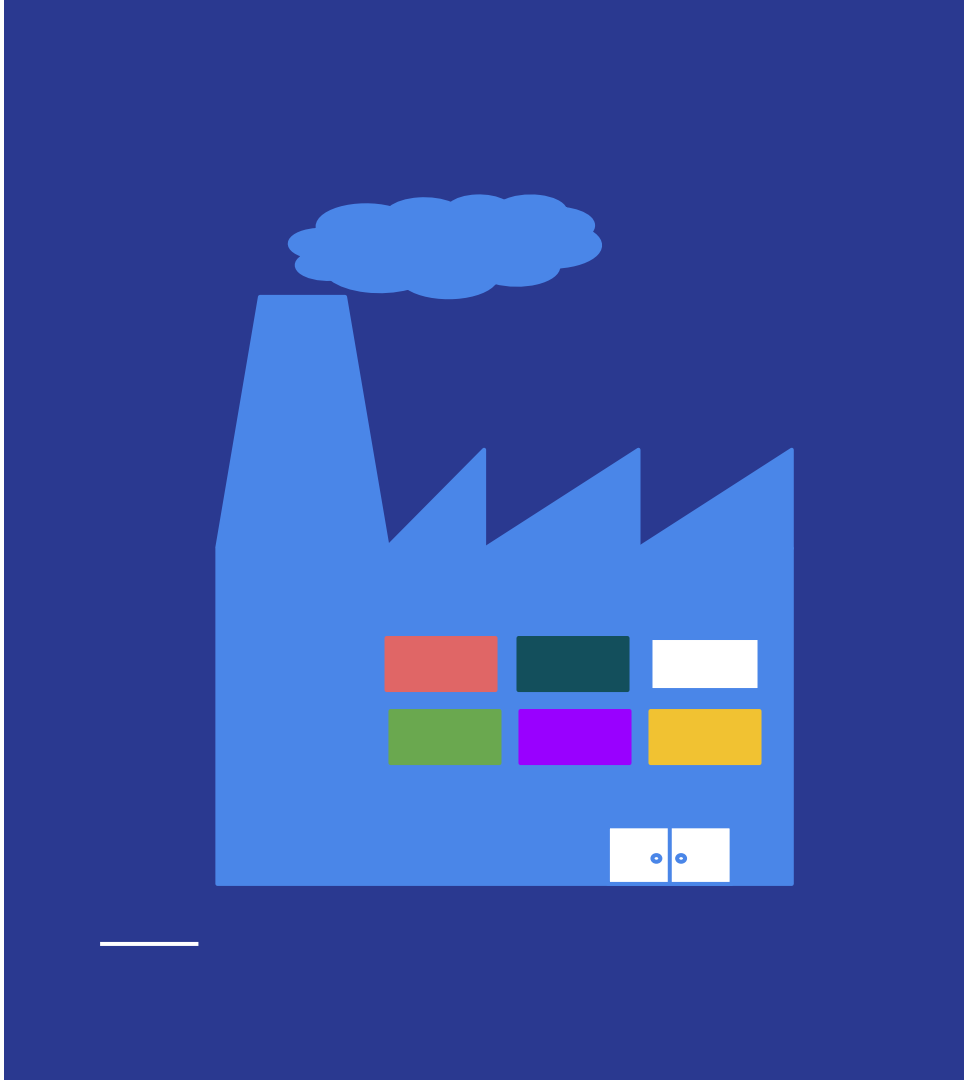
- Lo que no se mide no mejora
- Es necesario tener la mayor visibilidad del estado actual de la aplicación
- Al analizar el desempeño se pueden identificar errores en:
  - El código
  - La infraestructura
  - Alguna política del negocio

- ***Anti-patrón:*** no medir, política del “silencio positivo”, esperar por las quejas de los clientes.



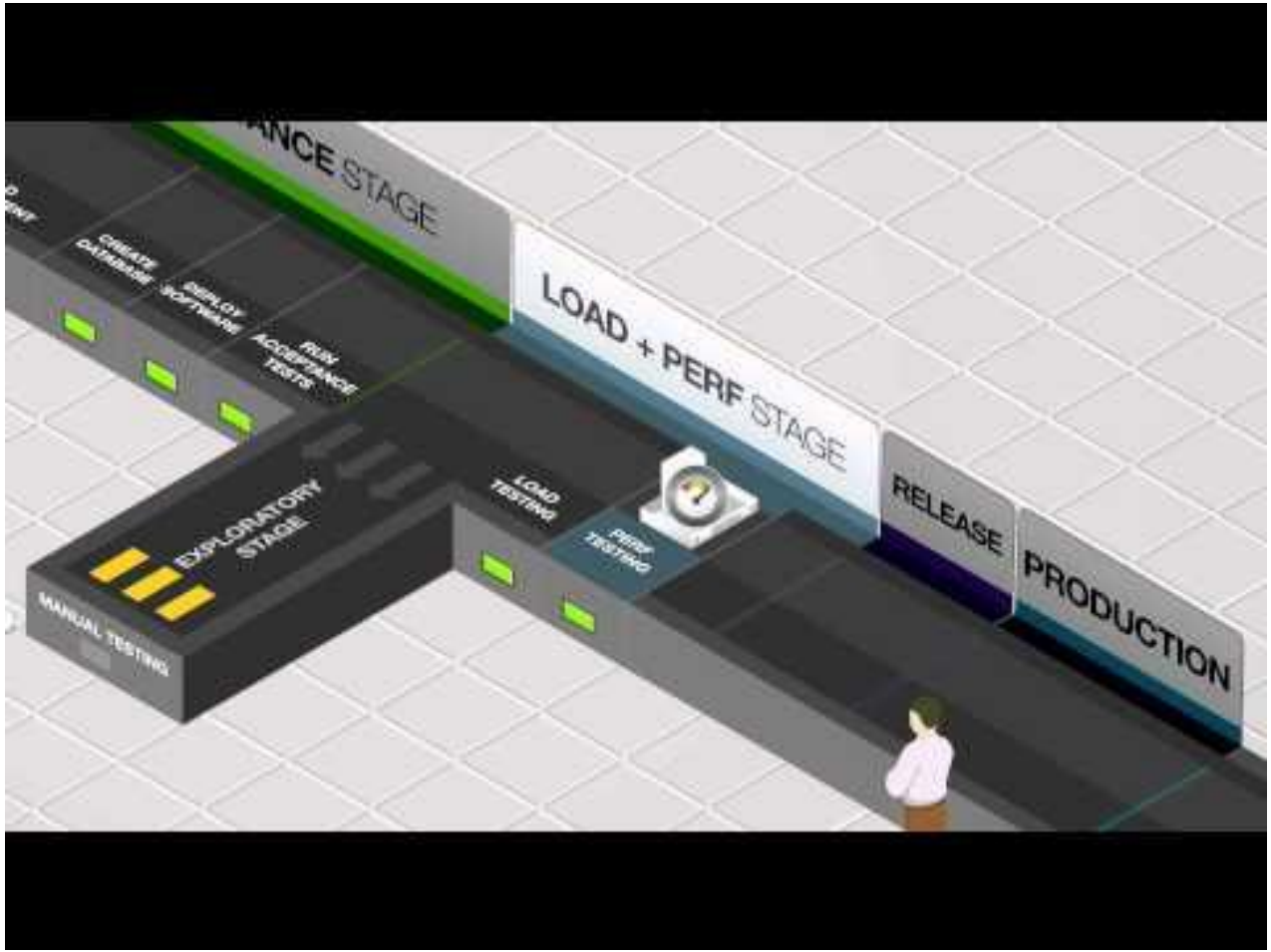
# Entrega continua

Continuous Delivery (CD)

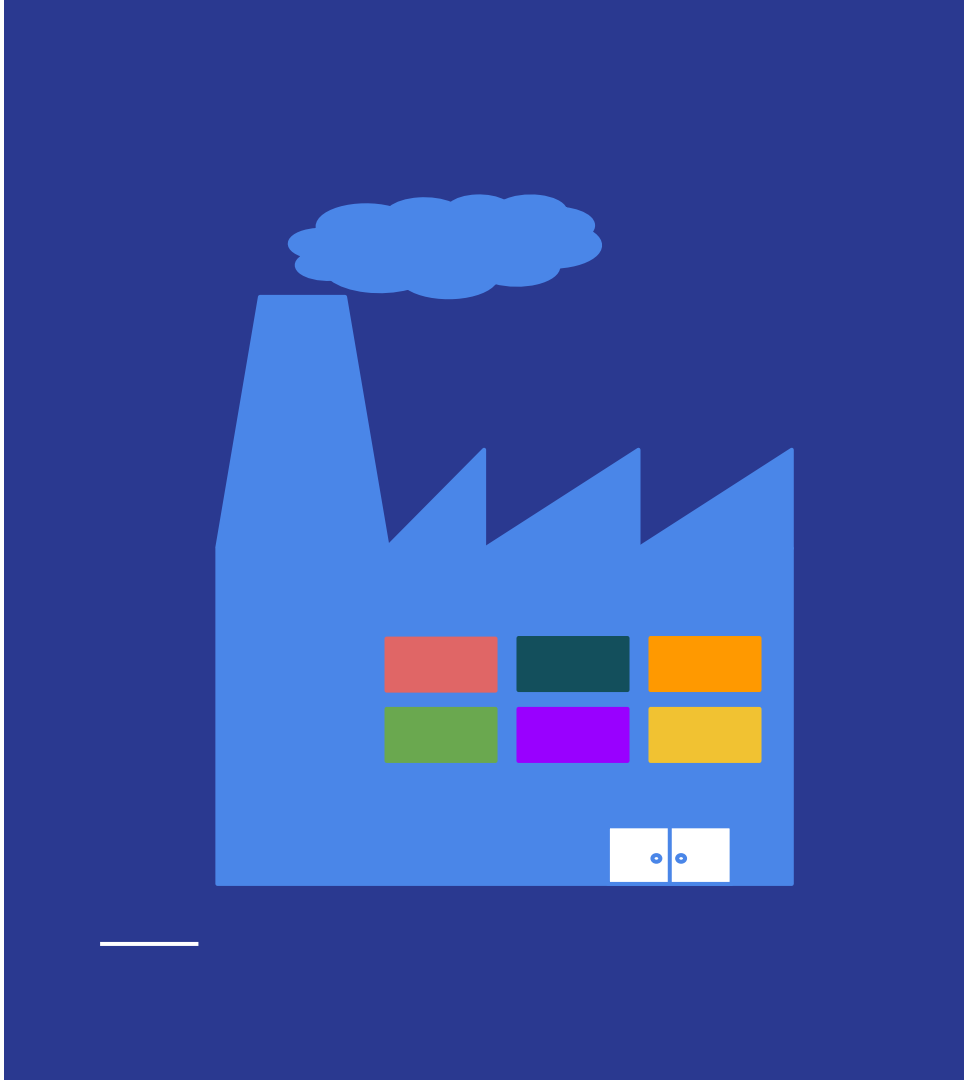


# Entrega continua

- Típicamente después de la Integración continua
- La entrega continua envía un nuevo artefacto estable a un ambiente de producción
- **Una tubería de entrega (*delivery pipeline*) automatizada**
- Implementado adecuadamente ayuda a reducir el costo, tiempo y riesgo en la entrega de nuevos cambios en las aplicaciones

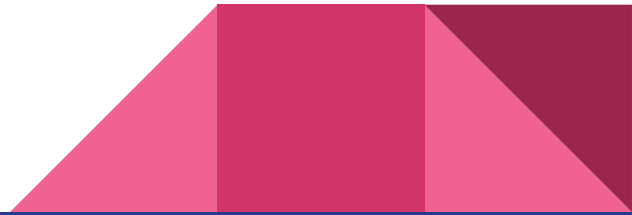

ANCE STAGE
CREATE DATABASE
DEPLOY SOFTWARE
RUN ACCEPTANCE TESTS
LOAD + PERF STAGE
LOAD TESTING
PERF TESTING
RELEASE
PRODUCTION
EXPLORATORY STAGE
MANUAL TESTING

# Administración de la configuración



# Conclusiones

- *DevOps:* intersección entre desarrollo, calidad y operaciones
- Aplicaciones más estables, ciclos de retroalimentación cortos, entrega continua
- Es el estilo de desarrollo que han adoptado las grandes compañías
- La *Fábrica de DevOps* es un buen punto de partida para lograr una implementación de este estilo
- Nada de lo anterior es posible sin un cambio cultural en la organización



# Muchas gracias!