

# Dev-ops: definiendo una pipeline

José A. Galindo



Objetivo principal:  
aprender a organizar  
un flujo de Dev-Ops en  
pipeline

# Continuous Integration

---

- Idea nacida de Martin Fowler; parte de lo que se conoce como XP y métodos ágiles
- 10 principios:
  1. Mantener un único repositorio de código
  2. Automatizar los *build*
  3. Hacer que los *build* sean auto-testeables
  4. Todo el mundo hace *commit* a la línea principal *una vez al día*
  5. Todos los *commits* deben lanzar un trabajo de integración en una máquina de integración
  6. Haz que los *build* sean rápidos y cortos
  7. Haz las pruebas en un clone del entorno de producción
  8. Haz que sea fácil para todo el mundo acceder al último ejecutable
  9. Todo el mundo puede ver lo que está pasando
  10. Automatizar los despliegues

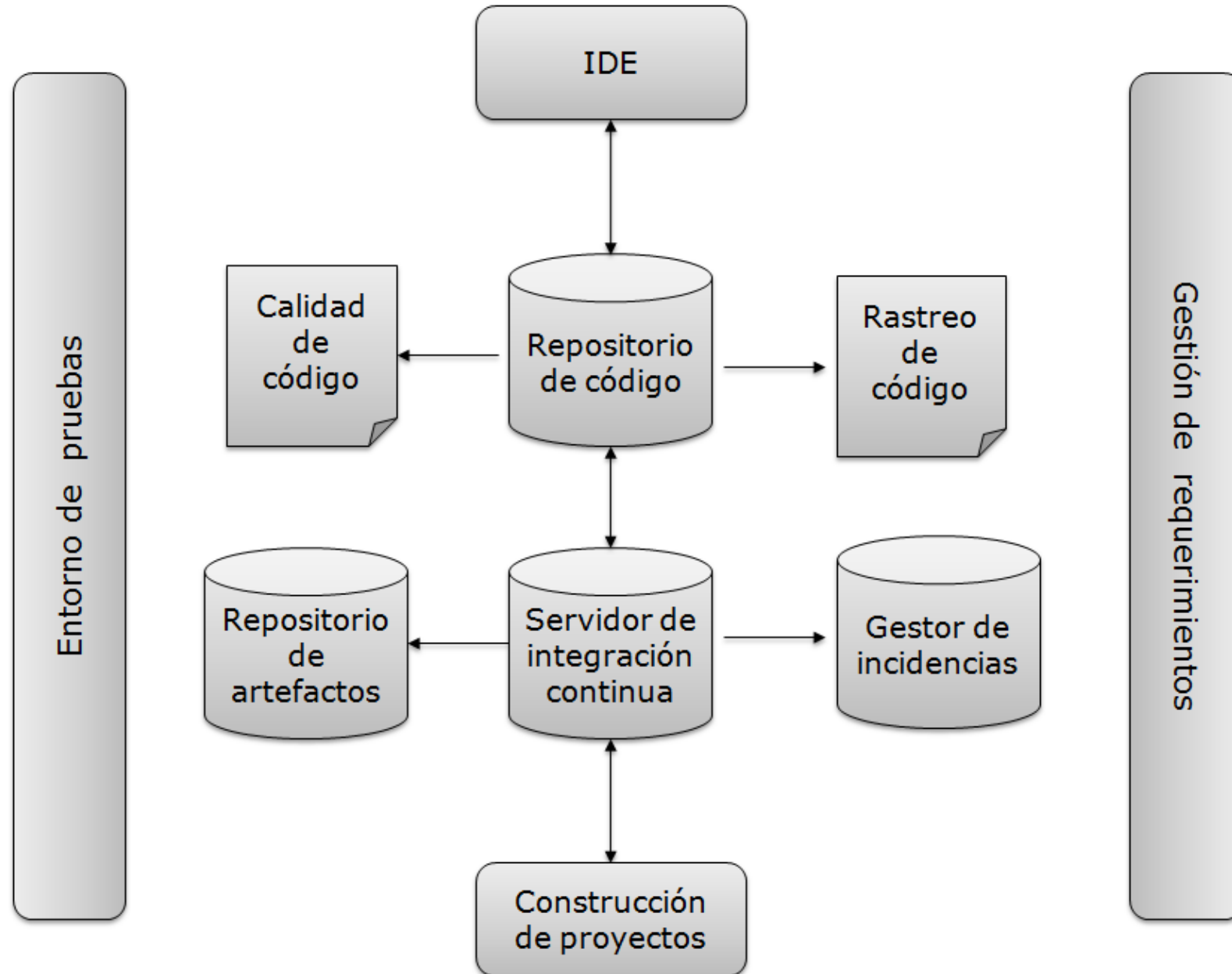


# So devops

---

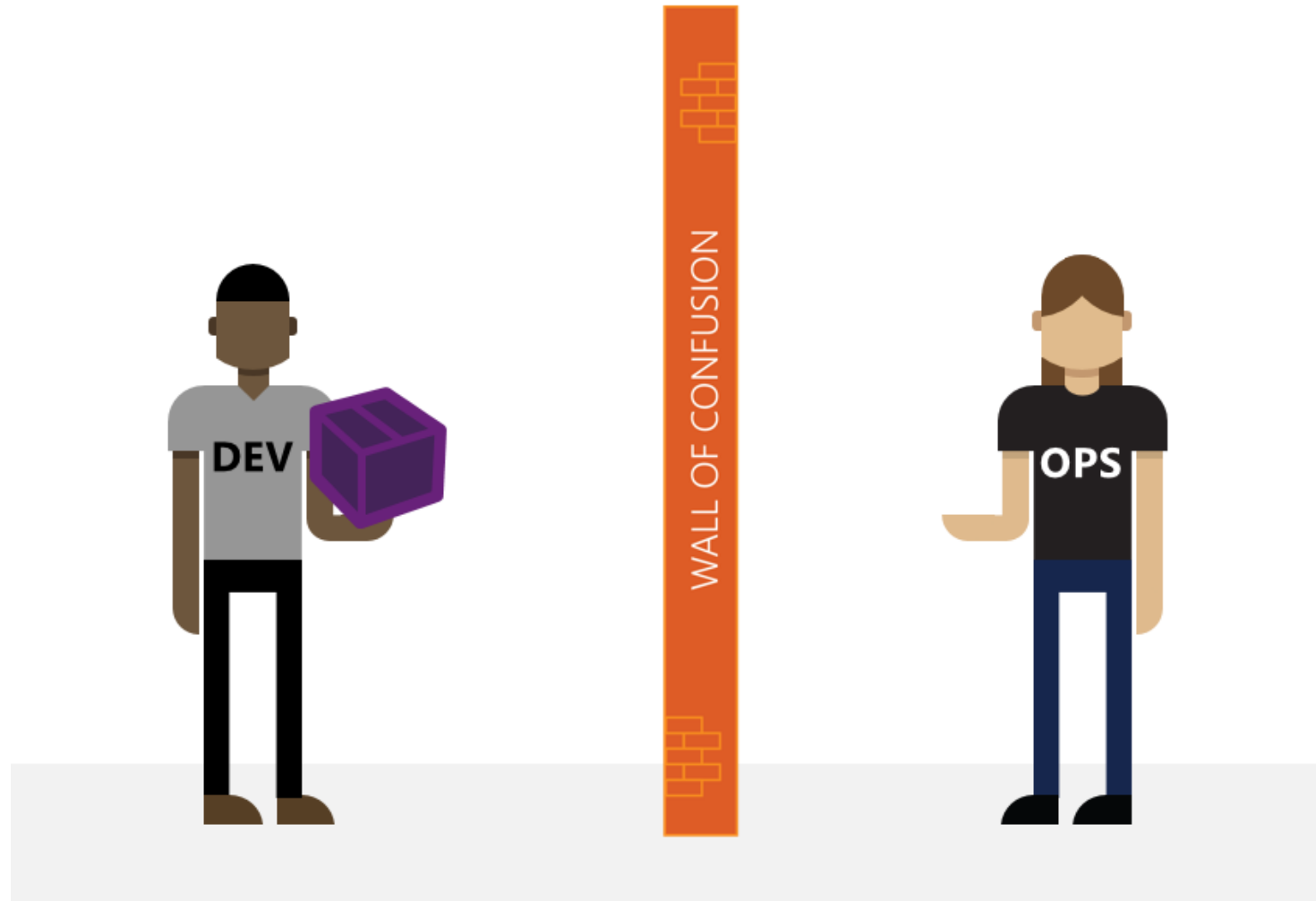


# Ecosistema de desarrollo con CI

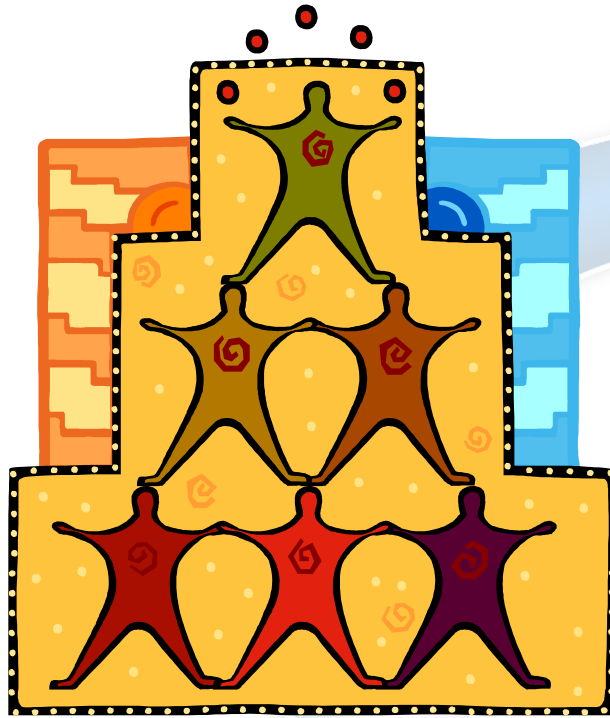


# Problema: El muro de la confusión

---



# Equipo vs tropa



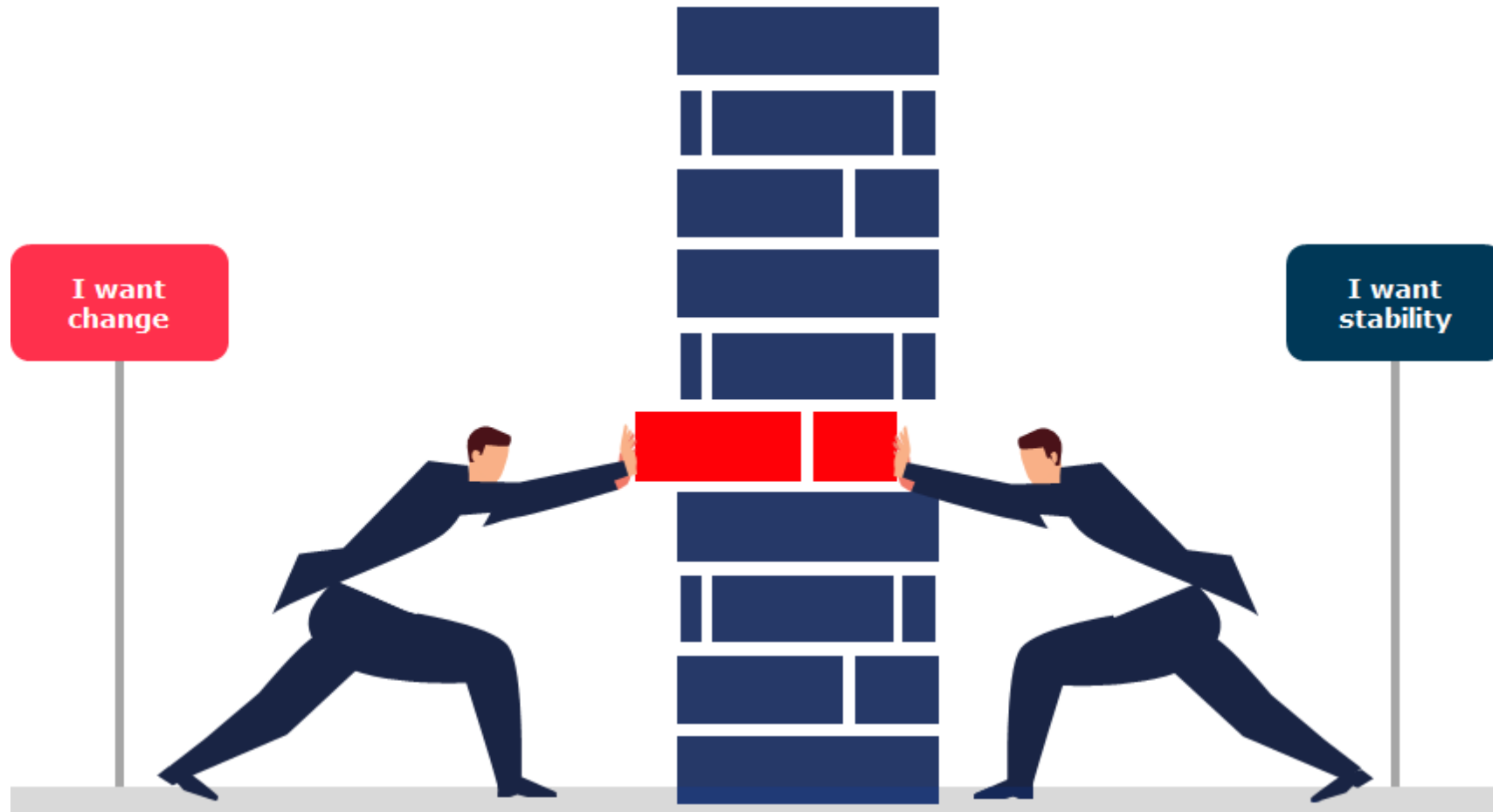
El equipo usa la inteligencia  
Dado un problema, busca  
soluciones

La tropa usa fuerza bruta  
Dado un plan de trabajo lo  
ejecuta



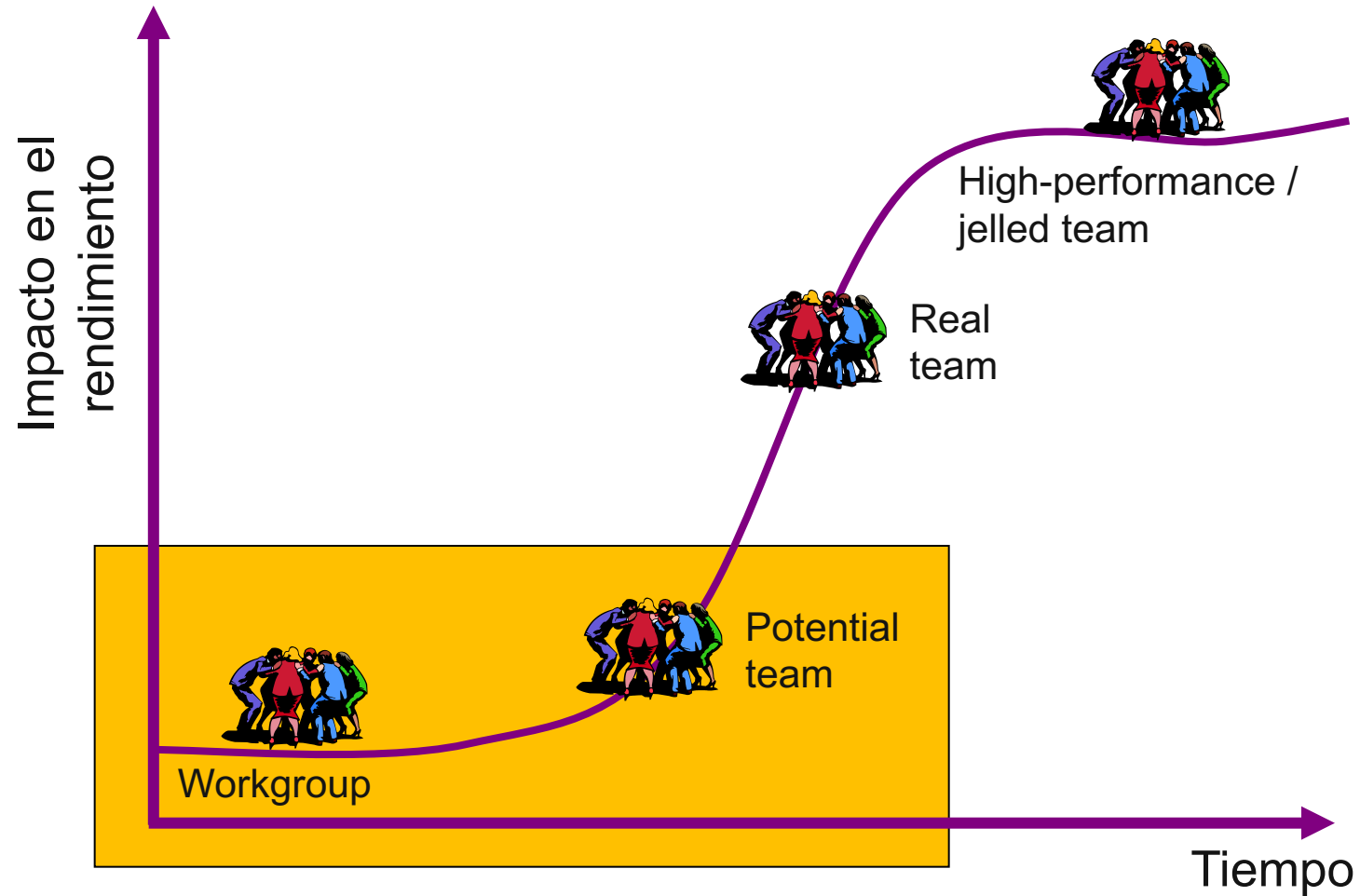
# Distintos objetivos. Distintos premios

---



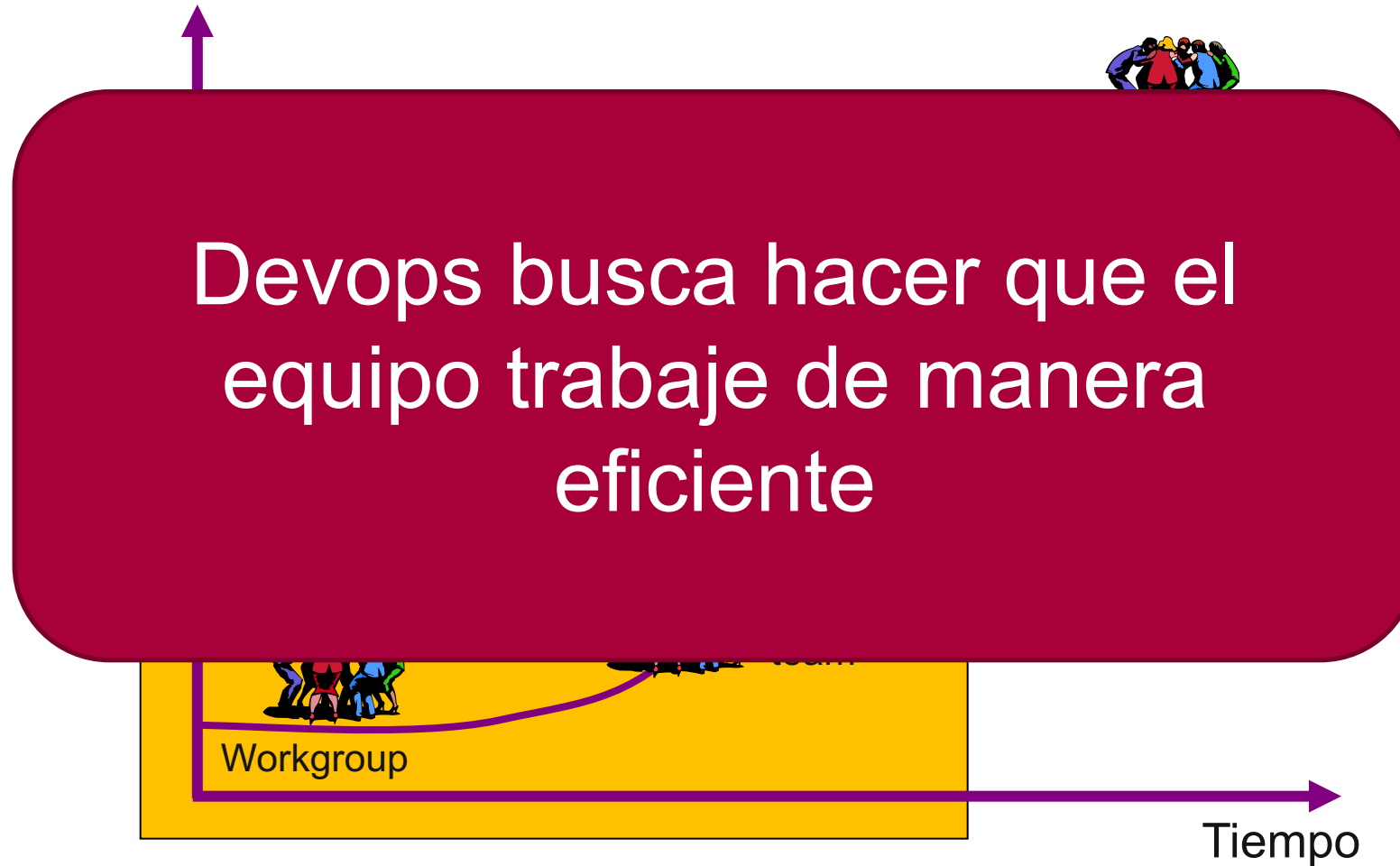


# La curva de la evolución de un grupo



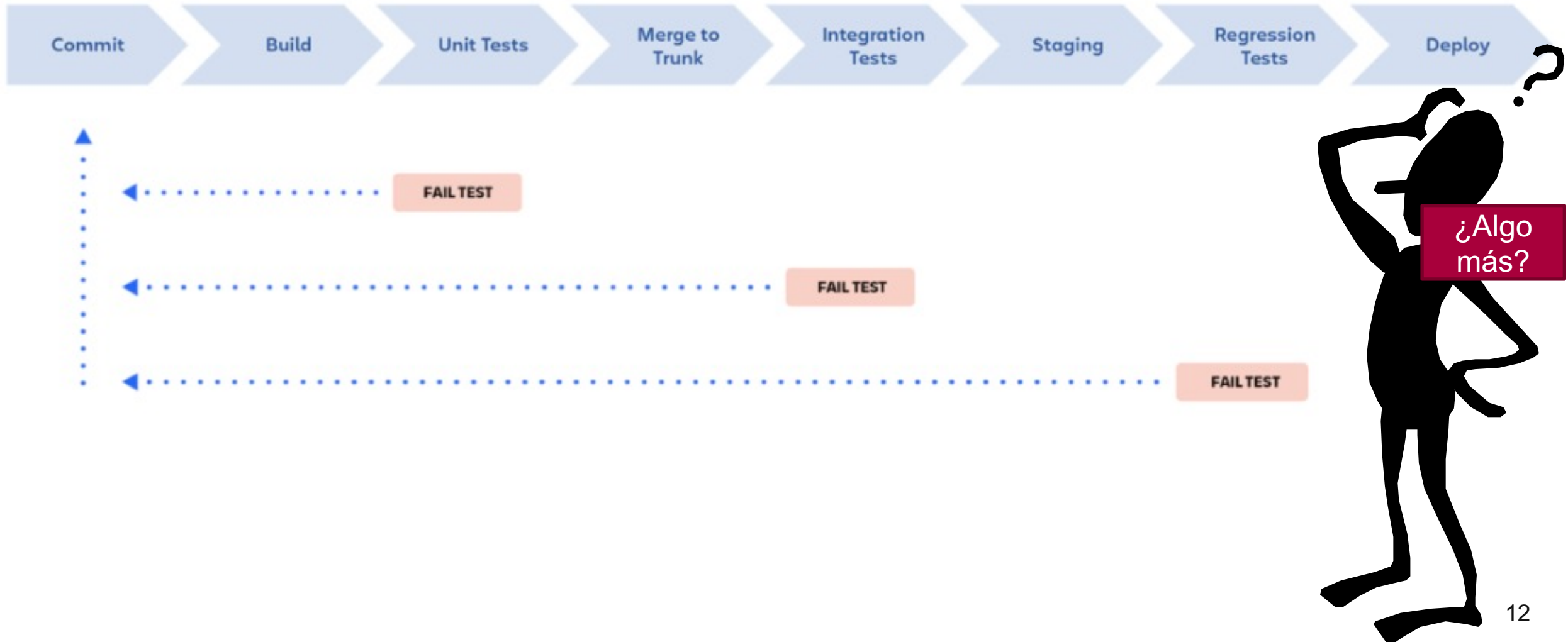
# La curva de la evolución de un grupo

---



¿Porqué no simplificar  
el modelo?

# ¿Cómo ordenar el trabajo que realizamos?



# Commits

---

- Mensajes de commits

## Commit message with description and breaking change footer

```
feat: allow provided config object to extend other configs  
  
BREAKING CHANGE: `extends` key in config file is now used for extending other config files
```

## Commit message with `!` to draw attention to breaking change

```
feat!: send an email to the customer when a product is shipped
```

## Commit message with scope and `!` to draw attention to breaking change

```
feat(api)!: send an email to the customer when a product is shipped
```

## Commit message with both `!` and BREAKING CHANGE footer

```
chore!: drop support for Node 6  
  
BREAKING CHANGE: use JavaScript features not available in Node 6.
```

# Estilo

## ■ Code Style check (Usando linters) PEP 8

```
def sumar_numeros(num1, num2):  
    """
```

Esta función toma dos números como entrada y devuelve su suma.

```
    :param num1: Primer número a sumar.
```

```
    :type num1: int
```

```
    :param num2: Segundo número a sumar.
```

```
    :type num2: int
```

```
    :return: La suma de los dos números.
```

```
    :rtype: int
```

```
    """
```

```
    resultado = num1 + num2
```

```
    return resultado
```

# Uso de dos líneas en blanco antes de una nueva función

# Espacio después de la coma en los parámetros de la función

# Longitud de línea dentro del límite de 79 caracteres



```
def sumarnumerosmal( numero1,numero2):  
    """Esta función devuelve la suma de dos números."""  
    resultado=numero1+numero2  
    return resultado
```

# Problemas/soluciones

---

- Mensajes de commits
  - Plantillas
  - Verificadores de commits en línea (conventional commits)
- Estilo unificado
  - PEP8 o similares

# Build

- Requisitos bien definidos
- Gestión de dependencias

	conda	pip
install python package	✓	✓
create virtual environment	✓, built-in	✗, requires <code>virtualenv</code> or <code>venv</code>
package format	<code>.tar.bz2</code> , <code>.conda</code>	<code>.whl</code> , <code>.tar.gz</code>
manages	binaries	wheel or source
can require compilers	✗	✓
package types	any	Python-only
dependency checks	✓	✗
package sources	Anaconda repo and Anaconda cloud	PyPI

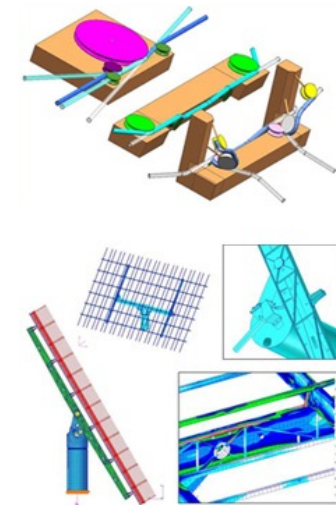
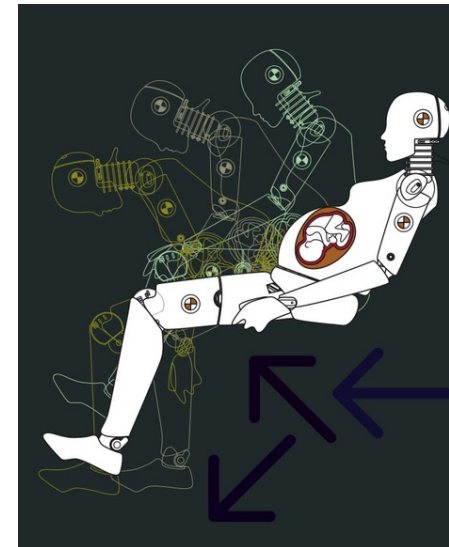
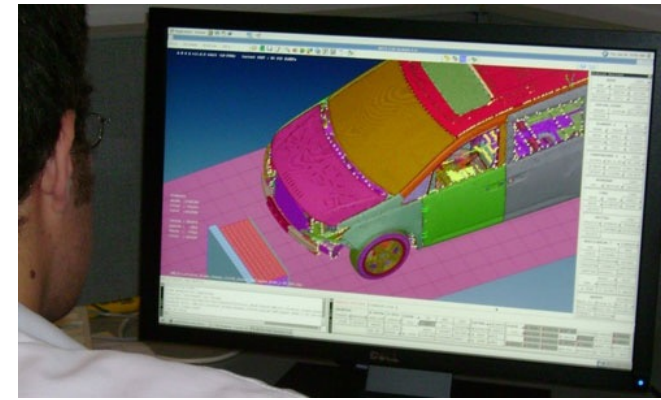


# Problemas/soluciones

---

- ¿Donde se compila el software?
  - Contenedores
  - Maquinas virtuales
  - Maquina local
  - ...
- ¿Cómo se integran los componentes del software?
  - Pip, conda, compilación, etc

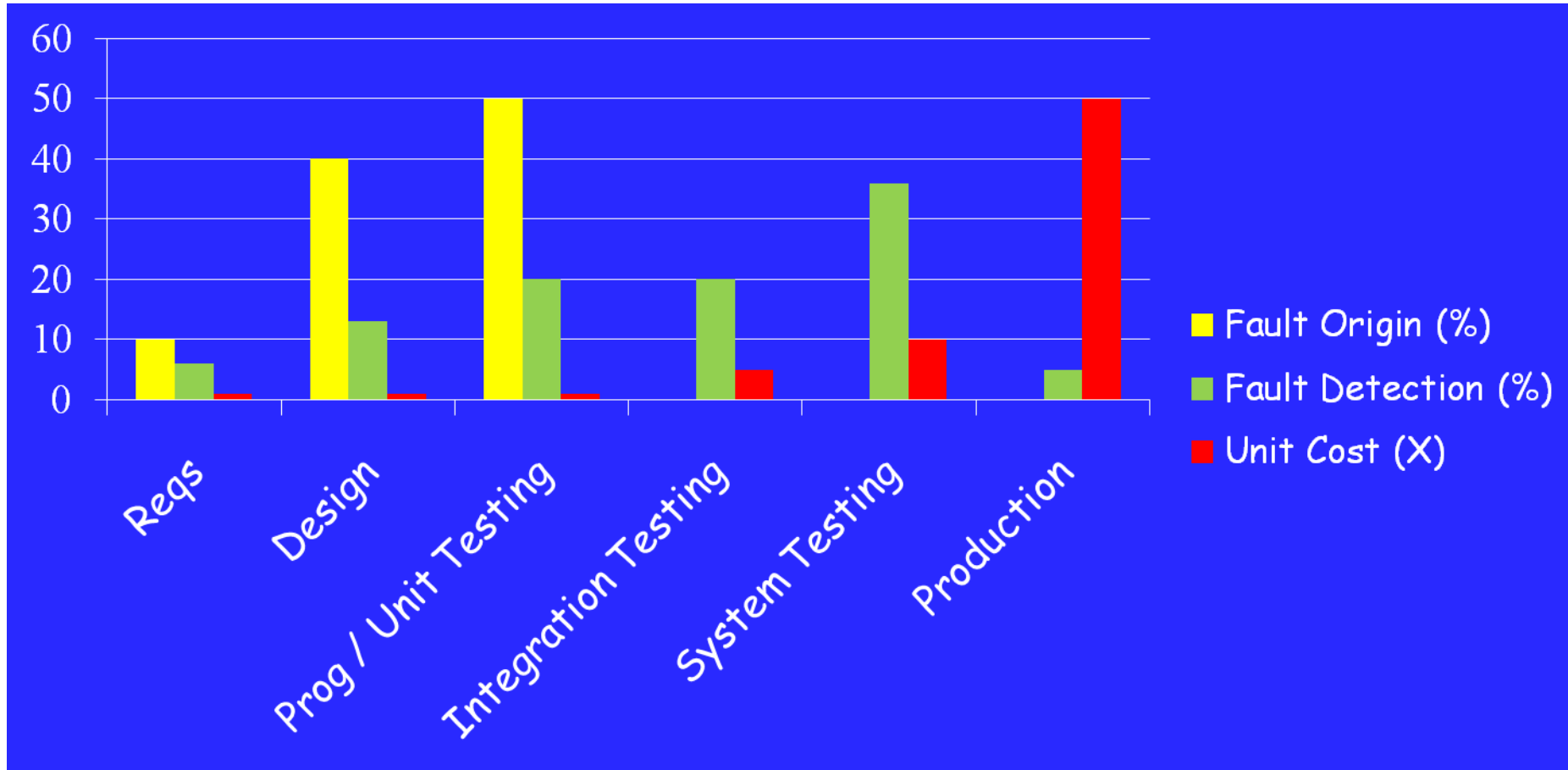
# Pruebas



**Ejecución de la prueba**

**Diseño de la prueba**

# Etapas dónde se producen errores



# Repo usage model

- ¿Qué tipo de pruebas?
- Mejora la cobertura de las pruebas
- ¿Cómo las integramos en el servidor de integración continua?




**All checks have passed**

3 successful checks

[Hide all checks](#)



 **separate-jobs / unit-tests (pull\_request)** Successful in 18s

[Details](#)



 **single-job / combined-tests (pull\_request)** Successful in 1m

[Details](#)



 **separate-jobs / integration-tests (pull\_request)** Successful in 1m

[Details](#)



**This branch has no conflicts with the base branch**

Merging can be performed automatically.

**Merge pull request**



You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Staggin'

- Basado en secuencias:
  - Importancia del cambio:
    - X.Y.Z
      - X: cambios sustanciales en funcionalidad
      - Y: cambios menores en funcionalidad
      - Z: cambios menores, no hay cambios de funcionalidad
    - Estado de la versión:
      - Alpha: primera liberación, alpha, alpha1, alpha2,..
      - Beta: fase inicial, beta, beta1, beta 2, ....
      - Release candidate: candidata a versión final, rc, rc1, rc2,..
      - Final release: liberación final
  - Fecha de liberación:
    - Ubuntu 5.10, 10.04, 17.10, etc..
    - Wine 20040505

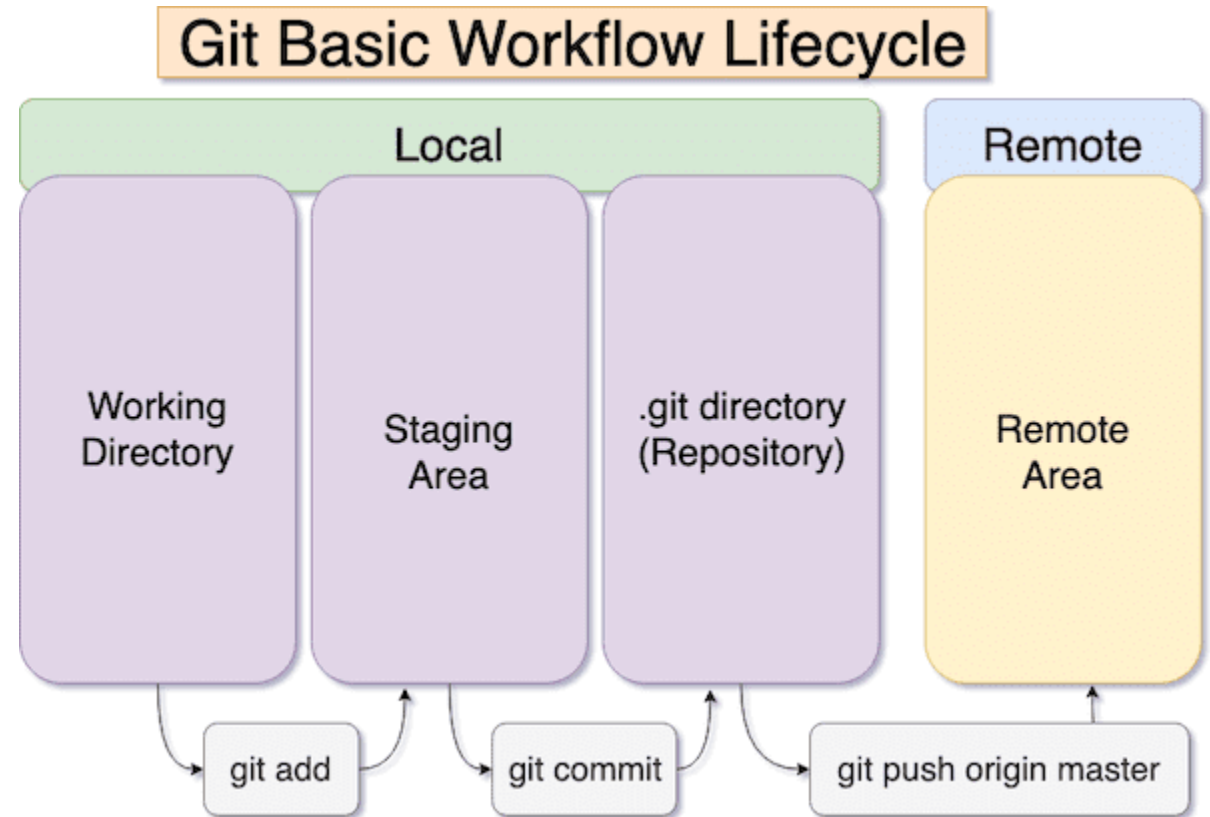
Versionado

```
$>python3 --version  
Python 3.8.5
```

# Problemas/soluciones

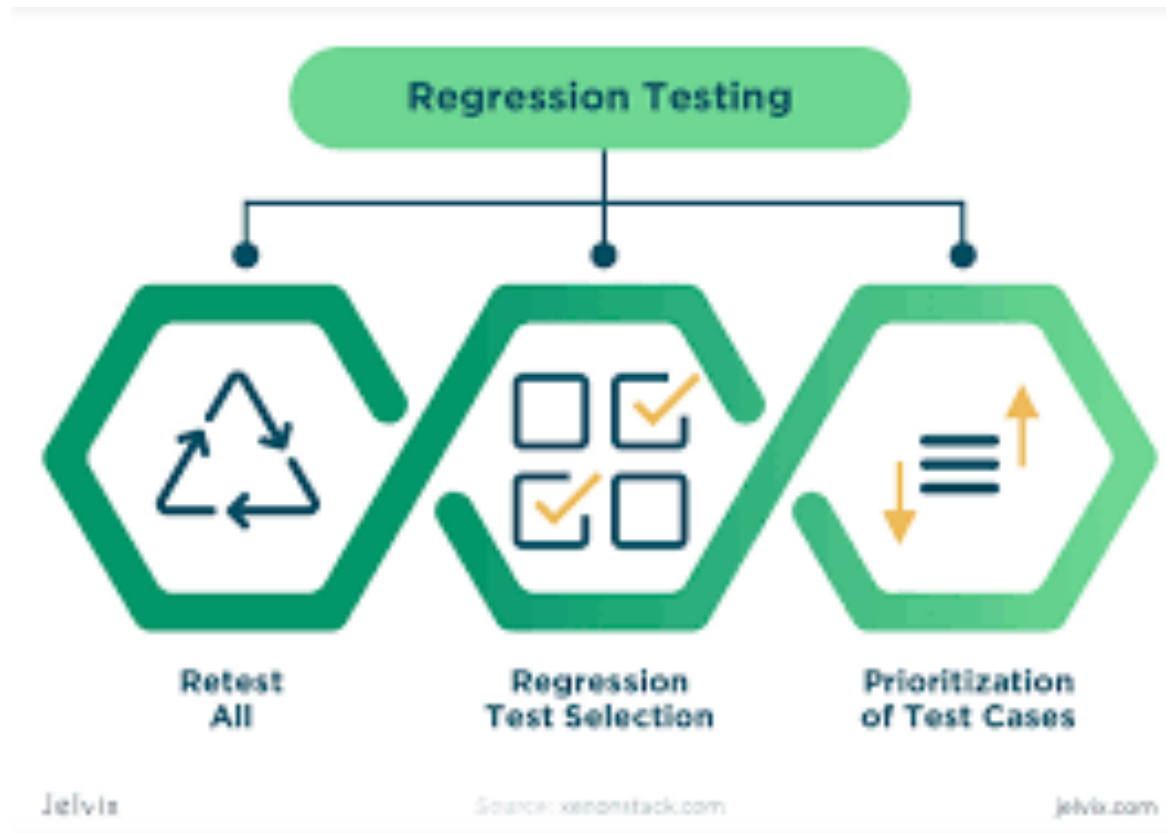
- Git tag
- Git Branch

- ¿Cuándo consideramos que añadimos algo al stage?



# Regression tests

---



# Deploy

---





# Problemas/soluciones

- Precio

**TABLE 1: Basic Cloud Server Pricing for Virtual Servers**

	VIRTUAL MACHINE INSTANCE	BANDWIDTH IN	BANDWIDTH OUT	BACKUP	SUPPORT
RackSpace Cloud 1,024 MB/40 GB	\$.06/hour or \$43.80/month	\$0.08/GB	\$0.22/GB	\$0.15/GB	Included
Amazon EC2— Small server Linux	\$325/year, plus \$0.03/hour	\$0.10/GB	\$0.17/GB	\$0.15/GB/ month	\$0.015/ instance/ hour
GoGrid 1,024 MB/ 60 GB	\$0.19/hour	Free	\$0.50/GB	\$0.15/GB	Included

SOURCE: FRANK OELHORST

- Mantenimiento
  - Precio acceso, conexión, etc.
- Acceso
  - Servidor local vs remoto