Ciclo de vida de desarrollo: CI/CD

6.2 Fases de un pipeline

Fases de un pipeline

1. **Fundamentos**
2. Planificación
3. Arquitectura
4. Desarrollo
5. Testing
6. Despliegue
7. Mantenimiento

Fases de un pipeline

Fundamentos

- Flujo de trabajo (con git)
  - Git Flow
  - GitHub Flow
  - Otros

- Requisitos
  - Conocimiento de Git
  - Sistemas de Control de Versiones (como GitHub)

Fases de un pipeline

Git Flow

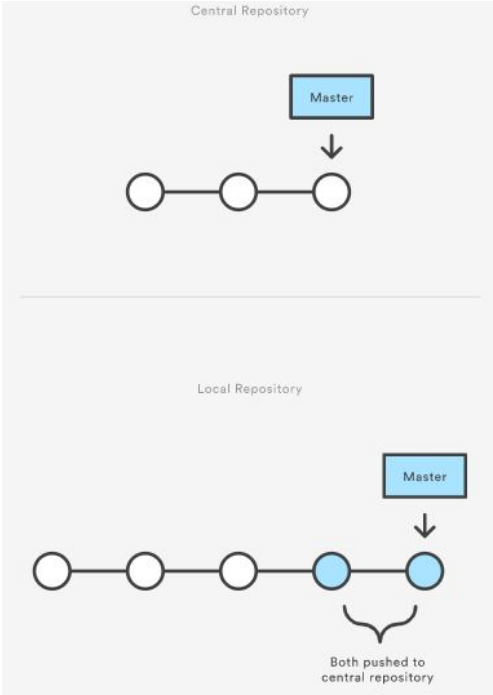
- Estándar de facto
- Diferentes aproximaciones
  - Centralizado (master)
  - Master/develop
  - Master/develop + feature/release/hotfix
  - Forking

https://datasift.github.io/gitflow/IntroducingGitFlow.html

Fases de un pipeline

Centralizado

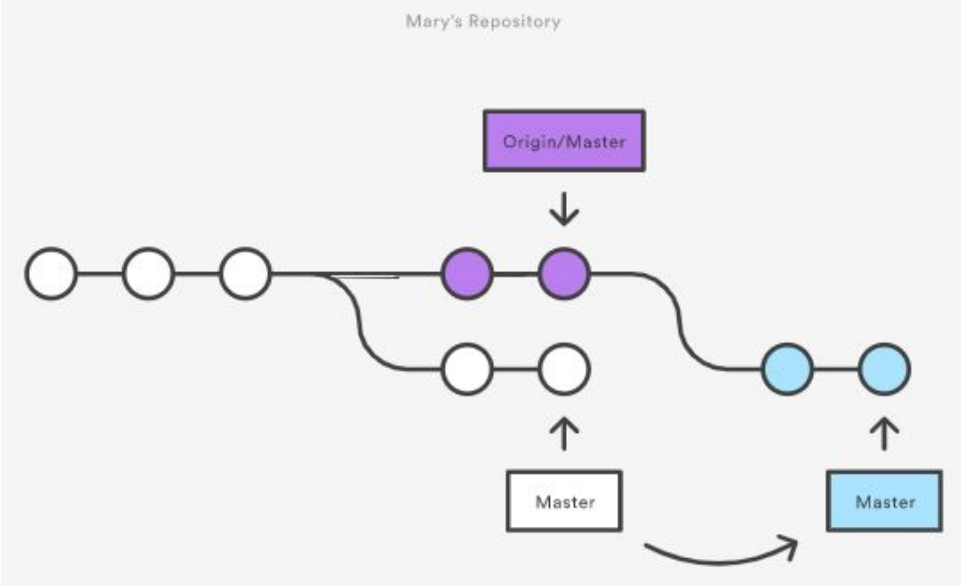
- Una única rama (master)
- Conveniente pocos desarrolladores
- Migrar de SVN a GIT



Fases de un pipeline

Centralizado

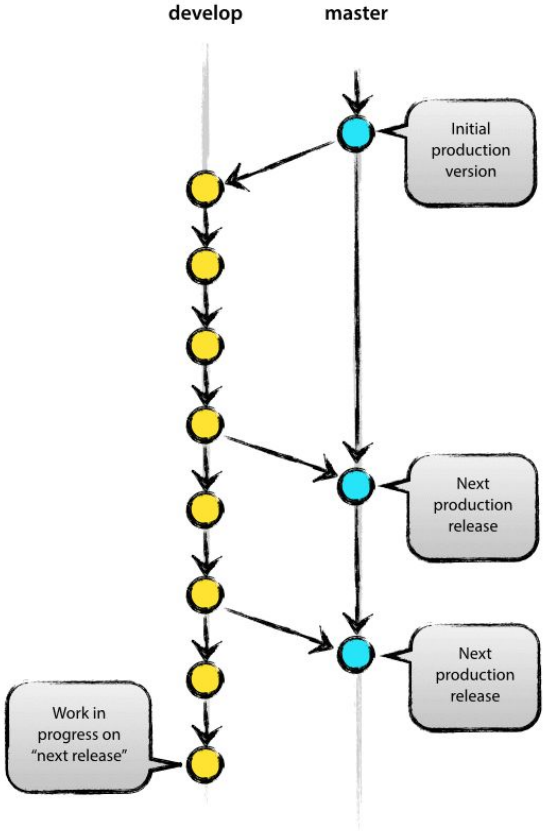
- Local *vs* remoto
- Resolución de conflictos
- Sin revisión de código



Fases de un pipeline

Master/develop

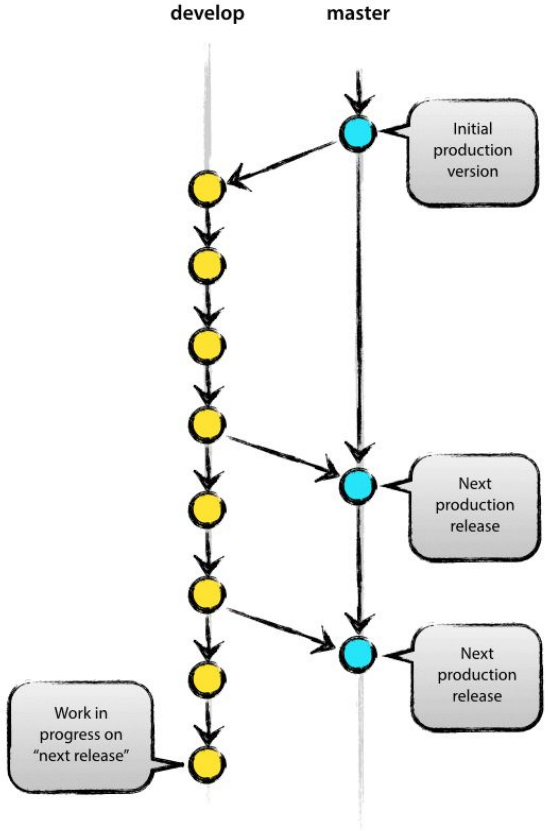
- Ramas principales
  - Ciclo de vida indefinido
  - Revisión develop → master
- Master
  - Código de producción
- Develop
  - Futuro código de producción



Fases de un pipeline

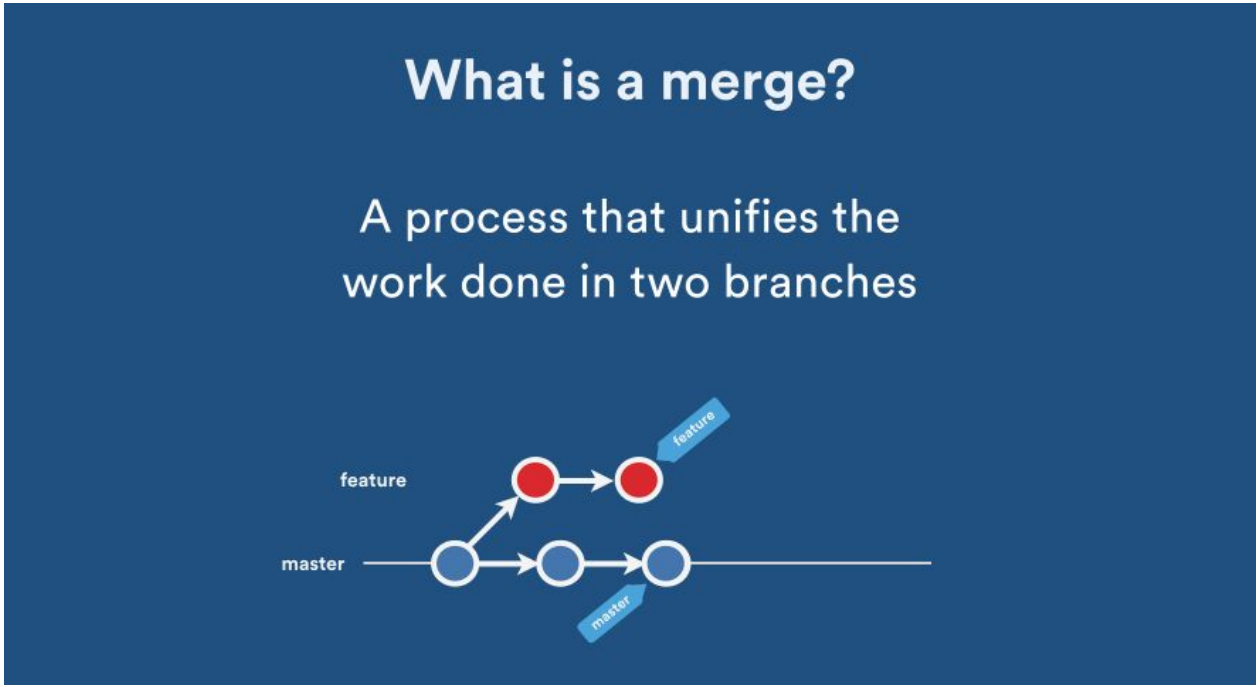
Master/develop

- Pocos desarrolladores
- Mejora la revisión de código
- Resolución de conflictos
  - *Merge (non fast-forward)*
  - *Rebase*
  - *Merge (fast-forward)*
  - *Squash*
  - *Cherry pick*



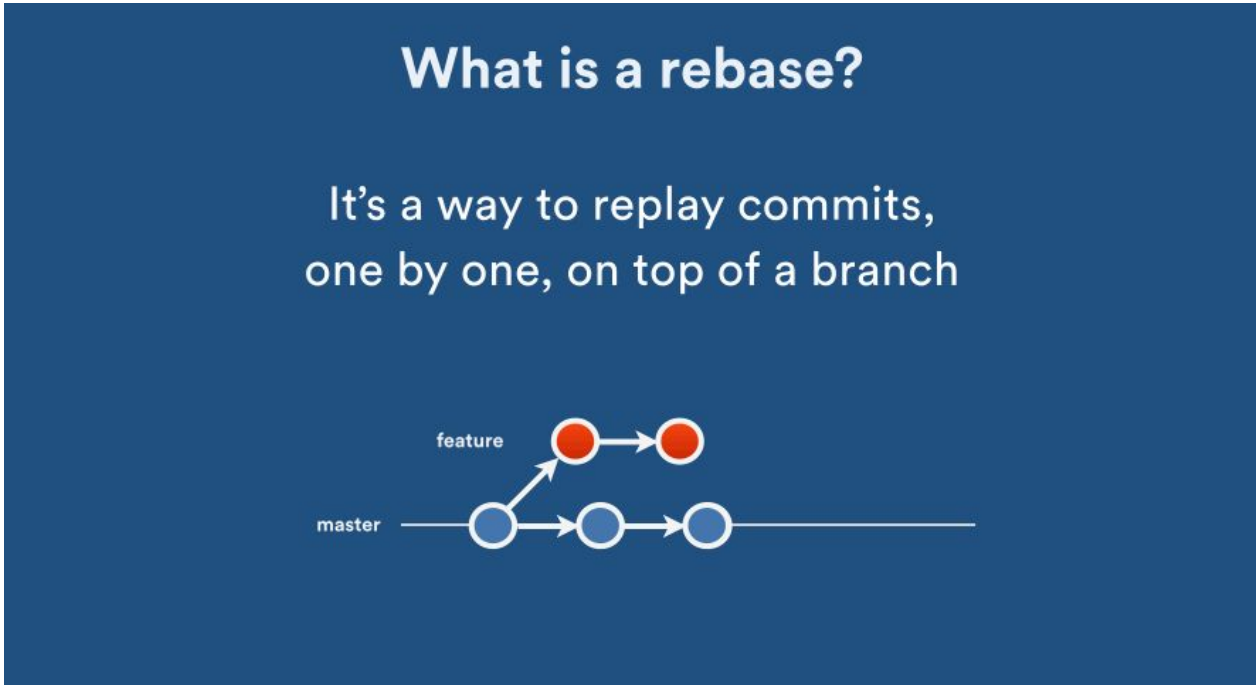
Fases de un pipeline

Merge (no ff)



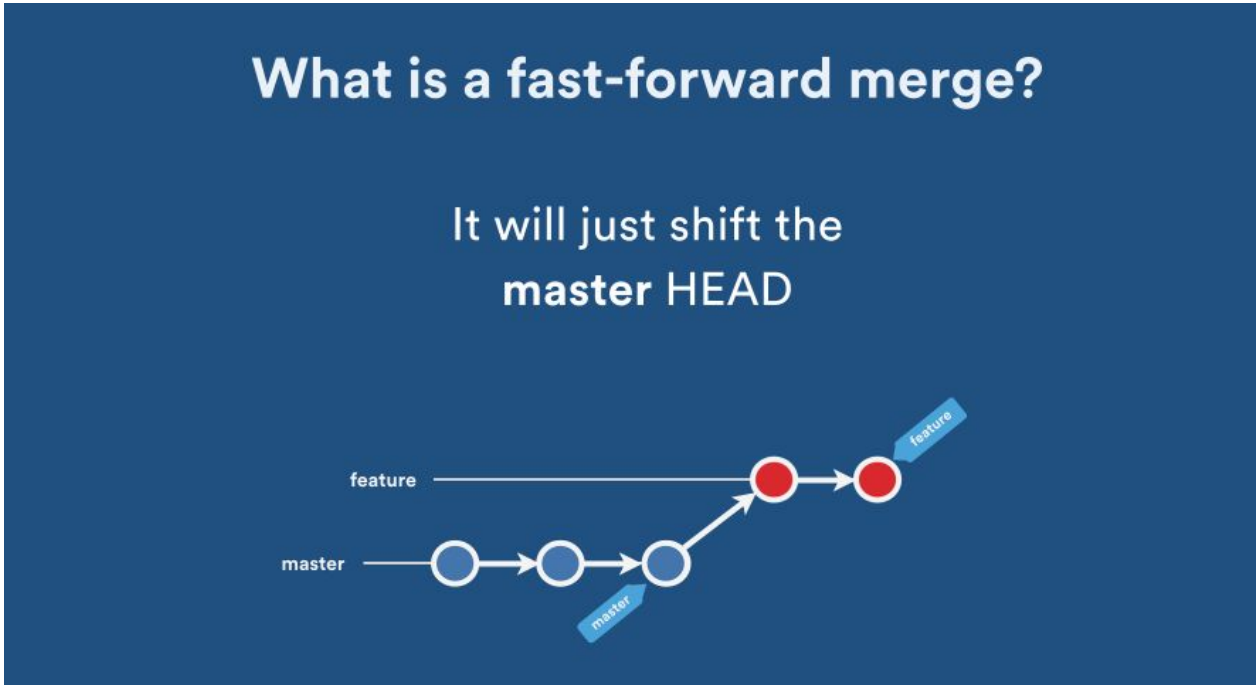
Fases de un pipeline

Rebase



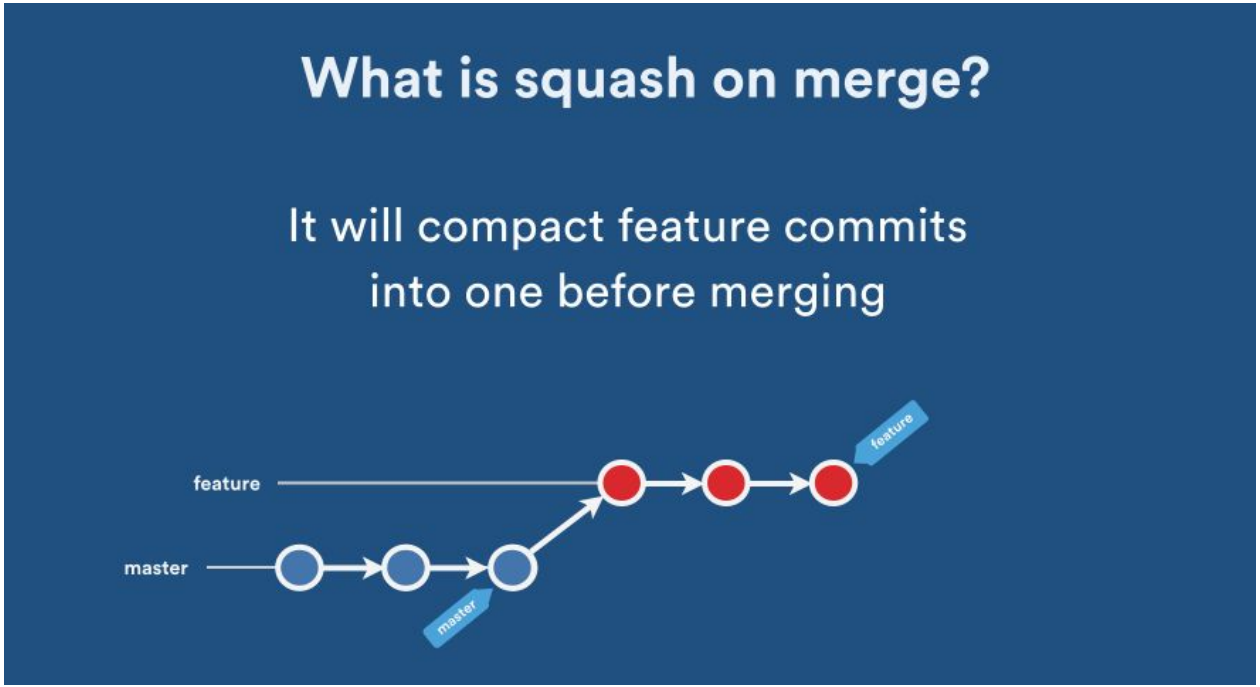
Fases de un pipeline

Merge (ff)



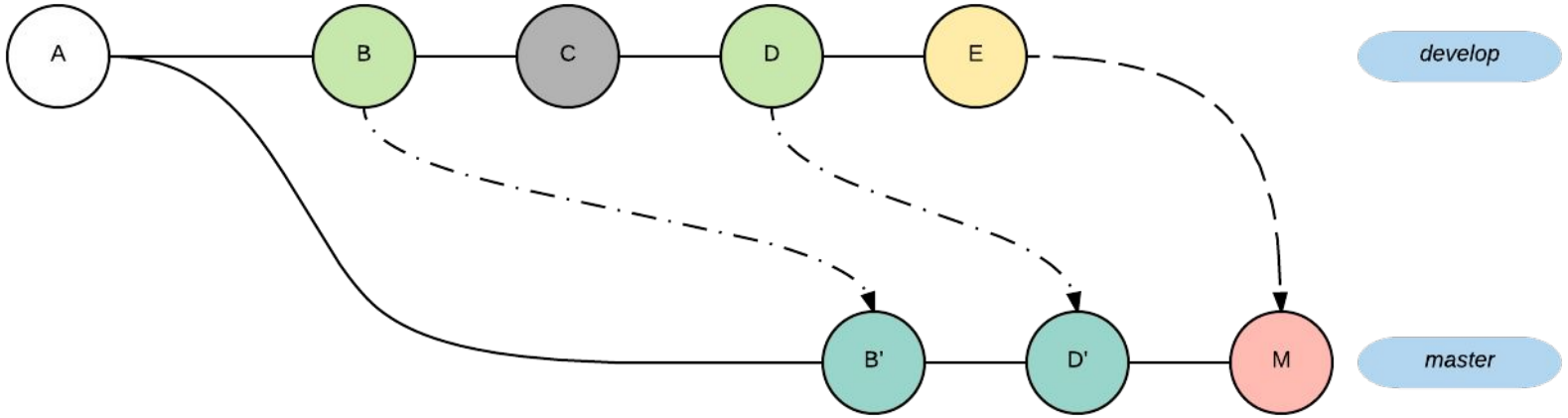
Fases de un pipeline

Squash



Fases de un pipeline

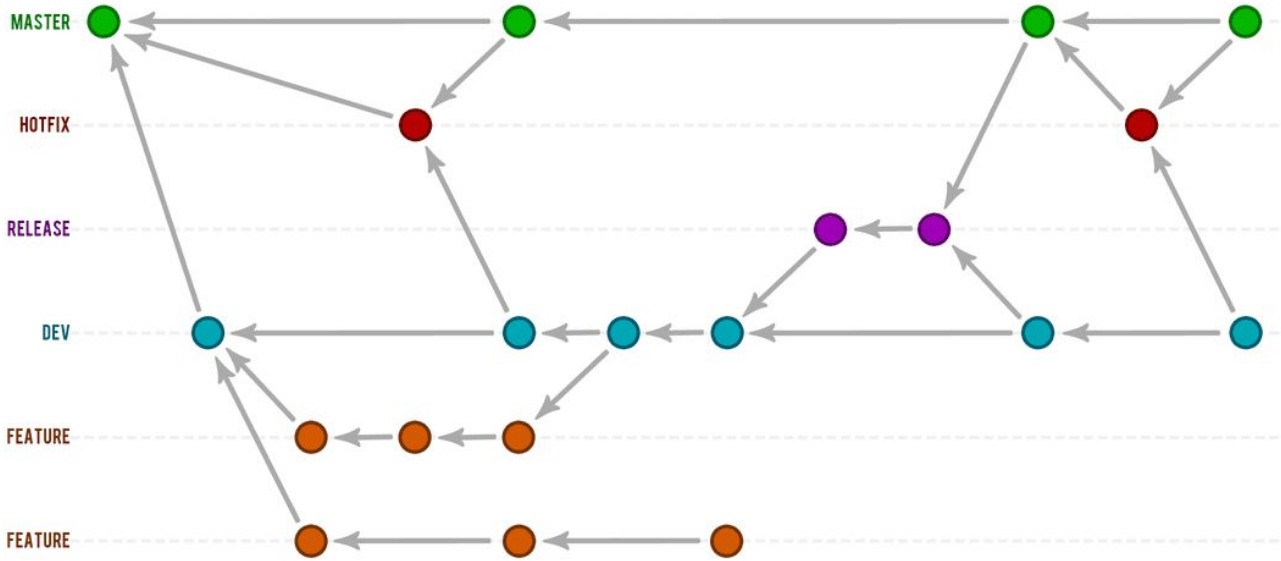
Cherry pick



Fases de un pipeline

Master/develop + feature/release/hotfix

- Feature
- Hotfix
- Release



Fases de un pipeline

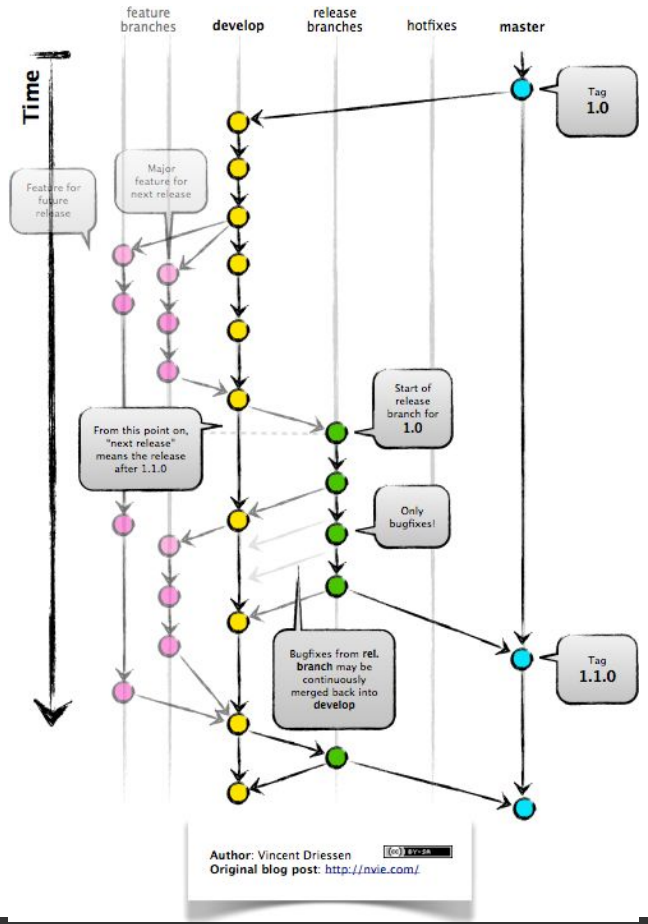
Feature

- Nuevas funcionalidades
- Uno o varios desarrolladores
- Sub-features
- Normalmente a develop

Fases de un pipeline

Release

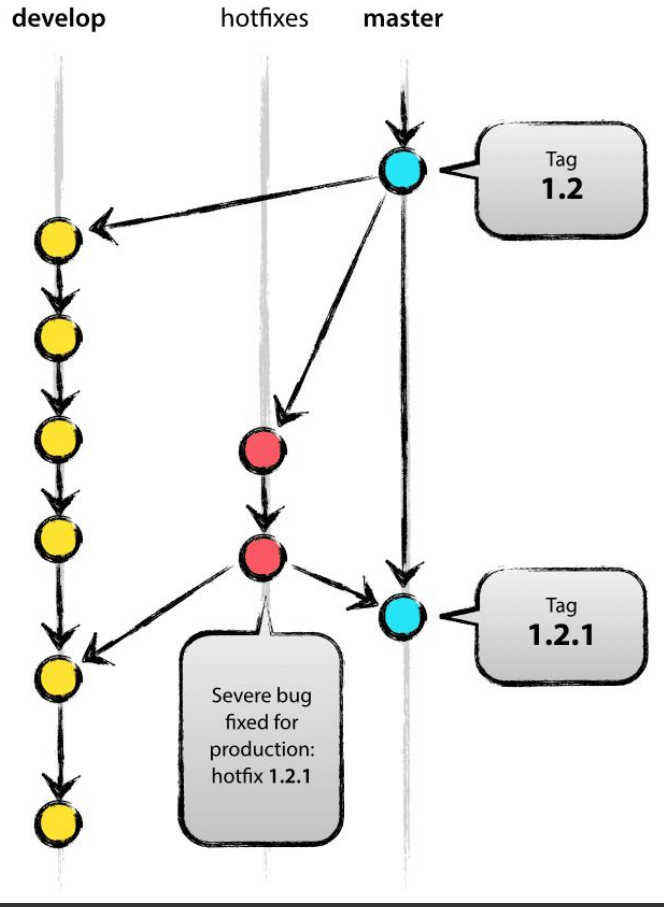
- Conjunto de funcionalidades nuevas
- Develop a master
- Genera una versión nueva
- Libera develop para nuevos cambios



Fases de un pipeline

Hotfix

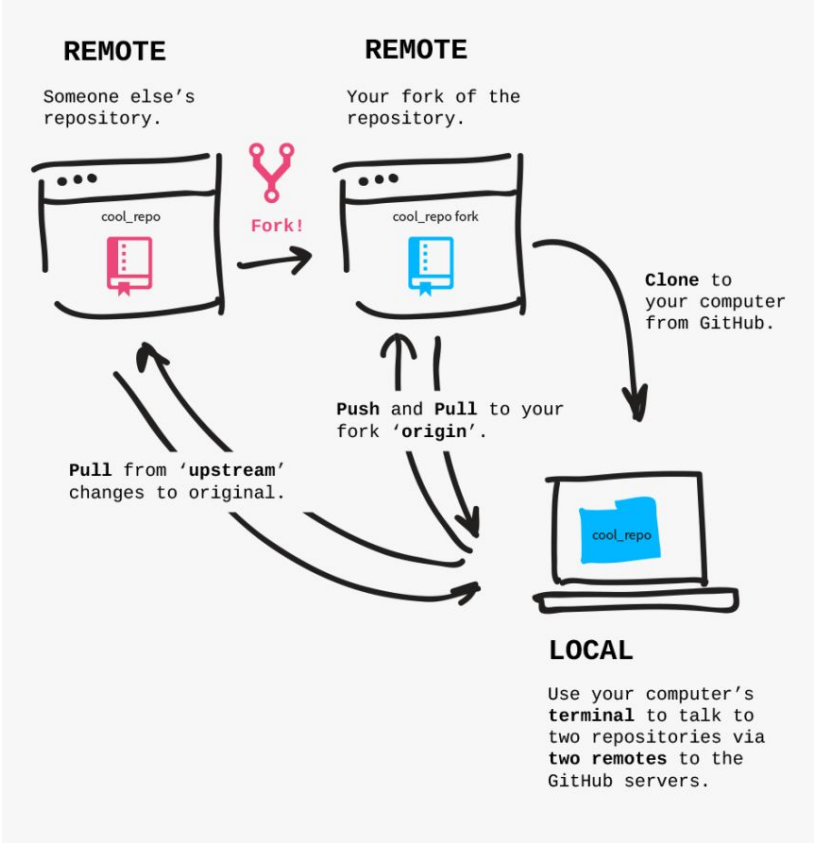
- Resuelve problemas en producción
- Se mezclan a master primero
- Generan una nueva versión
- No bloquean develop



Fases de un pipeline

Forking

- Diferentes “repositorios”
- Proyectos de la comunidad
- Cambios en paralelo
- Tienden a divergir en el tiempo

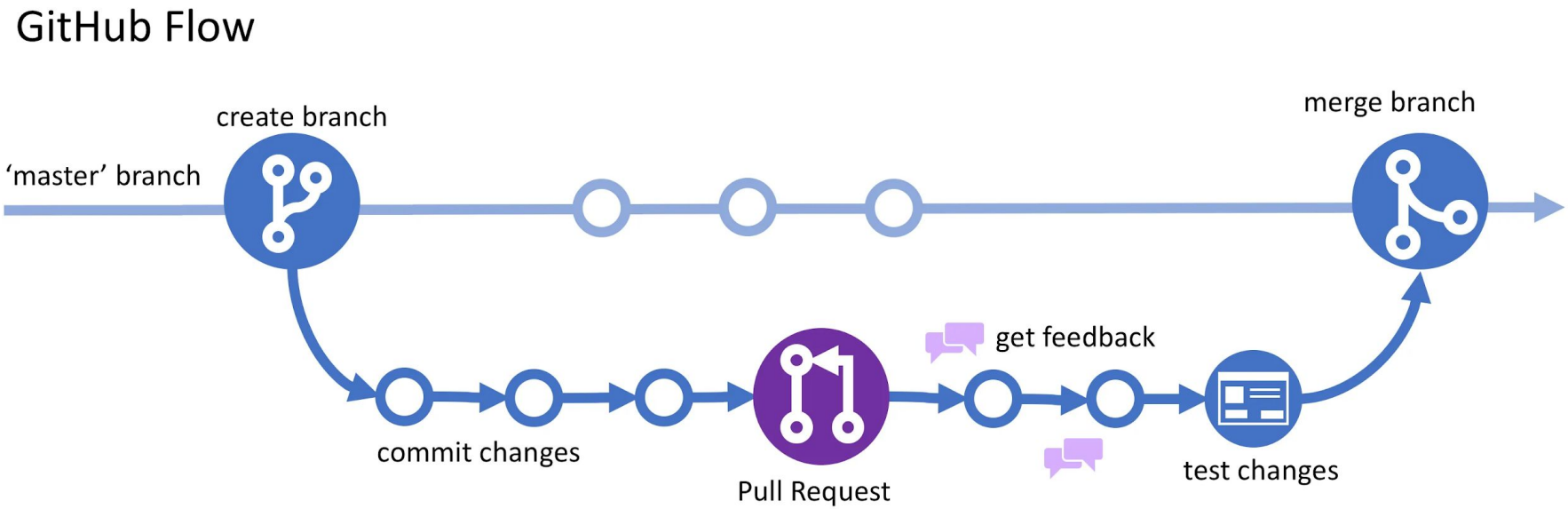


Fases de un pipeline

Git Flow

- Complejo
- master y develop redundantes
- Lucha contra los conflictos eterna
- Ralentiza la entrega

Fases de un pipeline



Fases de un pipeline

GitHub Flow

- Respuesta de Github a Git flow
- Sencillez
- Minimizar el código no entregado
- Orientado a CD

https://guides.github.com/introduction/flow/

Fases de un pipeline

GitHub Flow

- No vale para todos
- Despliegues complicados
- Uno (o dos) entornos
- Dependiente de herramientas

Fases de un pipeline

Otros

- Gitlab Flow → https://docs.gitlab.com/ee/workflow/gitlab_flow.html
- One Flow → https://www.endoflineblog.com/oneflow-a-git-branching-model-and-workflow
- Git DMZ → https://gist.github.com/djspiewak/9f2f91085607a4859a66
- Trunk-based → https://trunkbaseddevelopment.com/
- {{ingresa aqui tu modo}}

Suelen ser muy parecidos, con ligeras variaciones.
No hay un anillo único para dominarlos a todos.

Fases de un pipeline

Fundamentos

- Versionado
  - *Semantic* versioning MAJOR.MINOR.PATCH
    https://semver.org/
  - Incremental (por ejemplo, build de Jenkins)
  - *Naming* versioning (android, debian, etc)

Fases de un pipeline

Fundamentos

- Tags
  - *human friendly*
  - Versión <-> código
  - Artefacto <-> código

Fases de un pipeline

1. Fundamentos
2. **Planificación**
3. Arquitectura
4. Desarrollo
5. Testing
6. Despliegue
7. Mantenimiento

Fases de un pipeline

Planificación

- Preguntas
  - ¿Cuál es/son el/los artefacto/s del proyecto?
  - ¿Cuál es el mínimo número de pasos necesarios para conseguir ese/os artefacto/s?
  - ¿Podemos/queremos automatizar estos pasos?

Fases de un pipeline

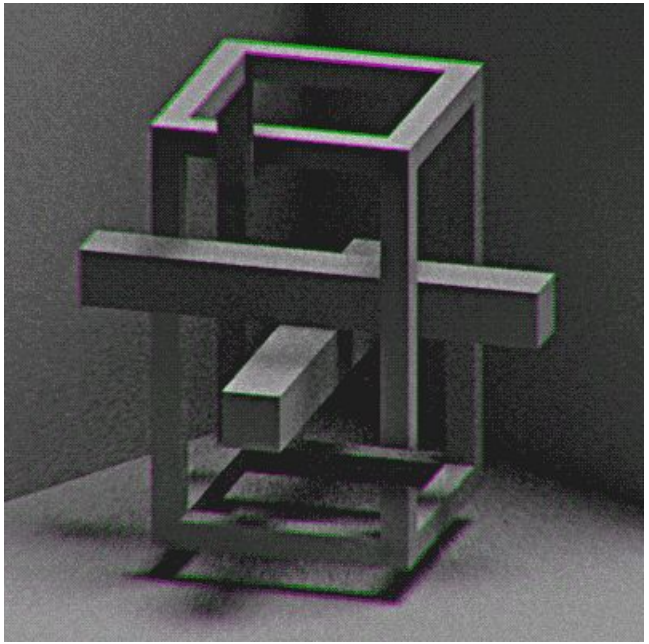
Planificación

- Reglas generales
    - Cada entregable/artefacto debe estar versionado
    - Cada entregable/artefacto se construye una única vez
    - Cada entregable/artefacto se prueba* en todos los entornos
    - El pipeline debe ser reproducible localmente
    - La lógica debe estar autocontenida en el pipeline

* Los hotfix pueden ir directos a producción

Fases de un pipeline

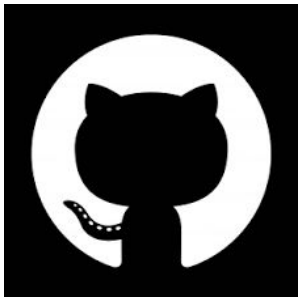
1. Fundamentos
2. Planificación
3. **Arquitectura**
4. Desarrollo
5. Testing
6. Despliegue
7. Mantenimiento



Fases de un pipeline

Arquitectura

- Servidor de CI (y CD)
- Gestor de artefactos/binarios
- Repositorio de código



Fases de un pipeline

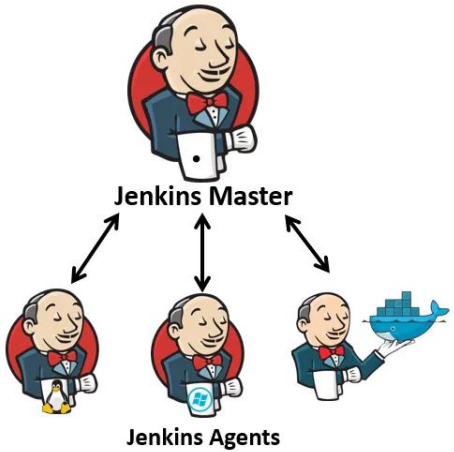
Arquitectura

- Escalabilidad
- Independencia
- Robustez
- Reusabilidad

Fases de un pipeline

Arquitectura

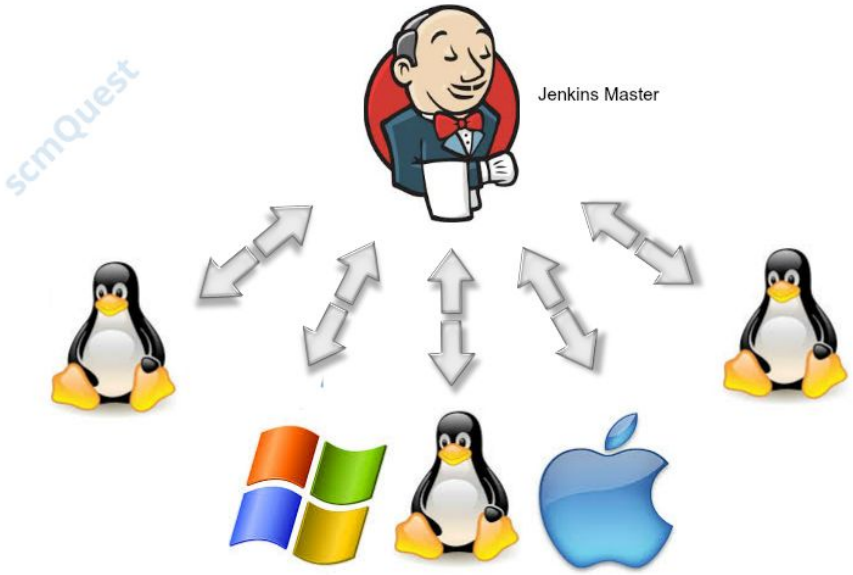
- Escalabilidad
  - Jenkins
    - Agentes
  - GitLab CI
    - Runners
  - Drone CI/GoCD/TeamCity
    - Agentes
  - Múltiples instancias
    - Por equipo, departamento, proyecto, costes...



Fases de un pipeline

Arquitectura

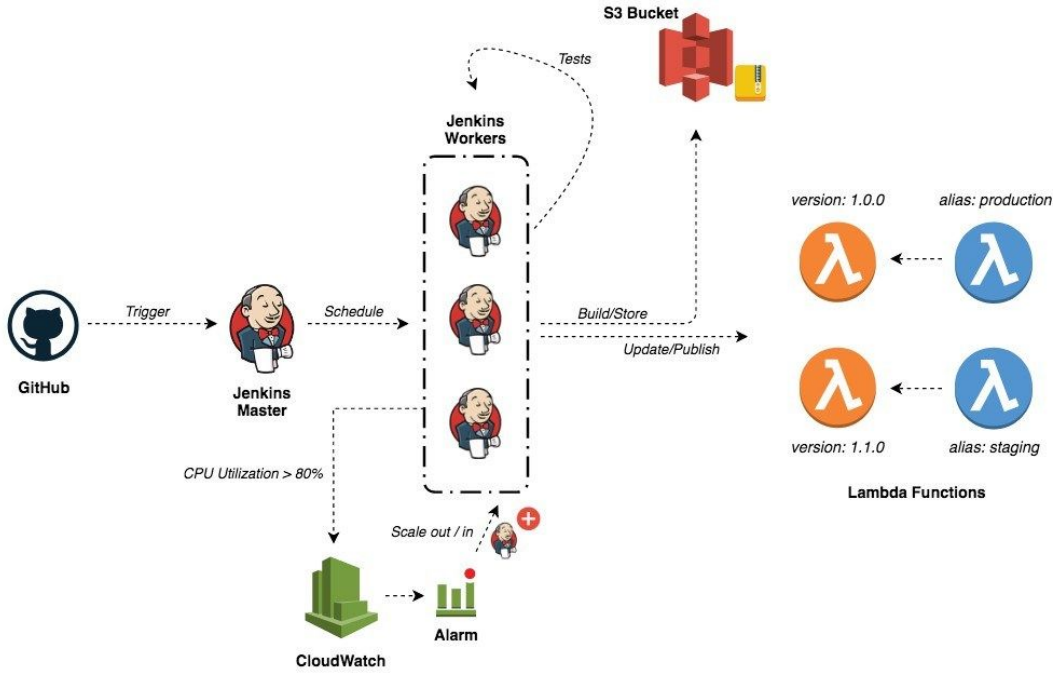
- Independencia
  - Runtime
  - Plataforma
  - Arquitectura



Fases de un pipeline

Arquitectura

- Robustez
  - Servidor CI
    - Estabilidad
    - Pipelines
  - Infraestructura IT
    - Estabilidad
    - Resiliencia

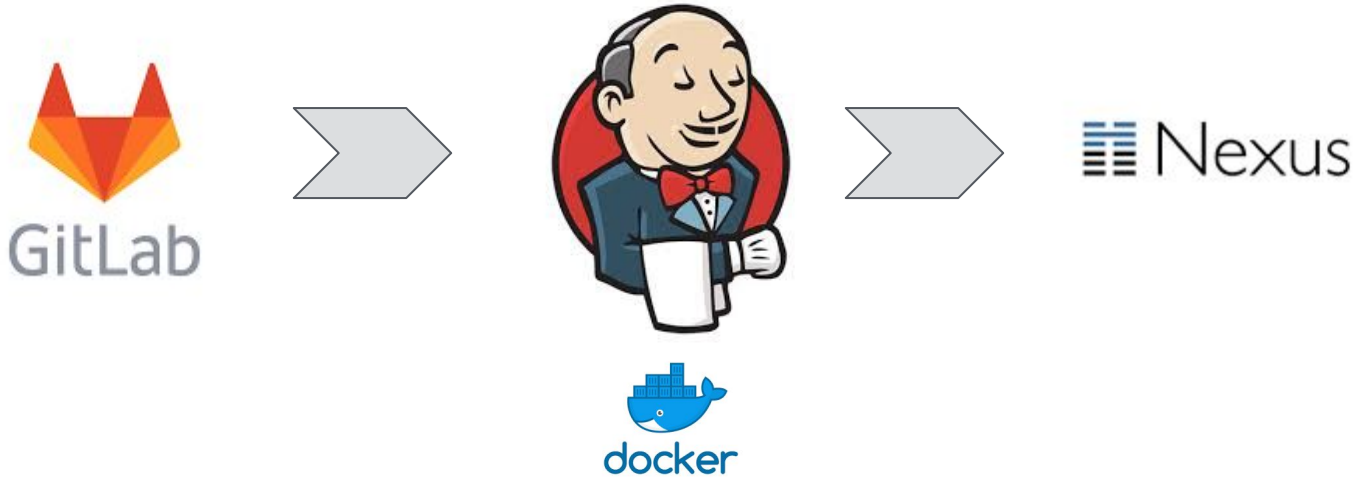


Fases de un pipeline

Arquitectura

- Reusabilidad
  - Local vs remote
  - Pipelines como código
  - Infraestructura como código
  - Librerías compartidas

Fases de un pipeline



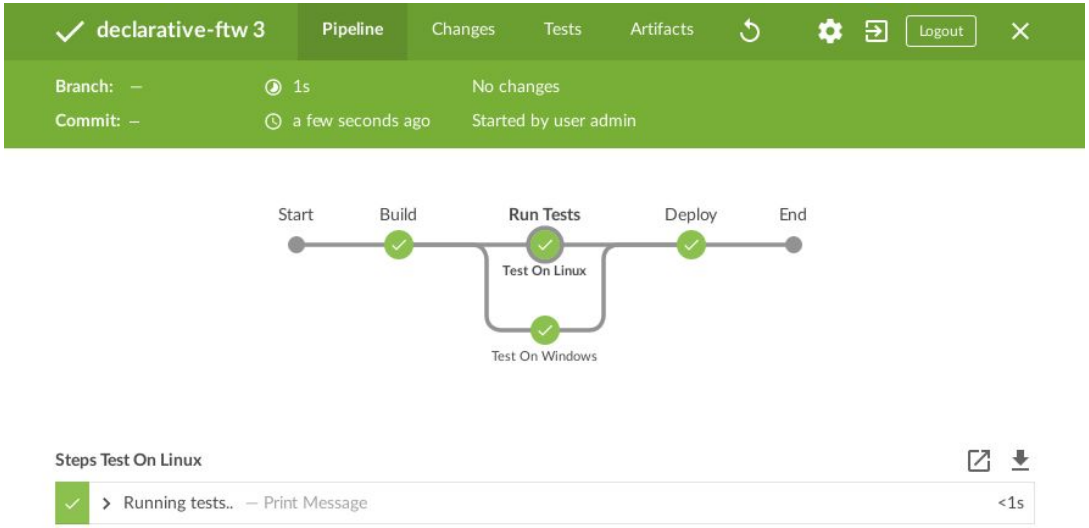
Fases de un pipeline

1. Fundamentos
2. Planificación
3. Arquitectura
4. **Desarrollo**
5. Testing
6. Despliegue
7. Mantenimiento

Fases de un pipeline

Desarrollo

- ¿Qué es un pipeline?
- ¿Cómo se define?
  - Manual
  - Código
- Jenkins
  - Script (Groovy)
  - Declarativo
- CI modernos
  - Declarativo



Fases de un pipeline

Desarrollo

- Manualmente
  - Mediante UI generamos las etapas y se definen las instrucciones de cada una de ellas.

- Como código
  - El pipeline se define completamente con código
  - Se almacena en control de versiones, normalmente junto al código

Fases de un pipeline

Desarrollo

- Sintaxis script (antes de 2016, versión 1)
  - DSL original de definir pipelines
  - DSL basado en Groovy
  - Modelo imperativo, enfocado a la lógica
  - Gestión de errores recae en el usuario
  - Permite usar construcciones de Groovy de manera nativa

https://www.jenkins.io/doc/book/pipeline/syntax/#scripted-pipeline

Fases de un pipeline

Desarrollo

```
node {
    stage('Example') {
        try {
            sh 'exit 1'
        }
        catch (exc) {
            echo 'Something failed, I should sound the klaxons!'
            throw
        }
    }
}
```

Fases de un pipeline

Desarrollo

- Sintaxis declarativa (desde 2016 versión 2)
  - Manera recomendada de definir pipelines
  - DSL más estructurado y estricto
  - Modelo declarativo, enfocado al estado deseado
  - Gestión de errores incluido
  - Curva de aprendizaje mucho menor
  - Más fácil de leer y entender

https://www.jenkins.io/doc/book/pipeline/syntax/#declarative-pipeline

Fases de un pipeline

Desarrollo

```
pipeline {
    agent none
    stages {
        stage('Example Build') {
            steps {
                echo 'Hello World'
            }
        }
        stage('Example Deploy') {
            when {
                triggeredBy "TimerTrigger"
            }
            steps {
                echo 'Deploying'
            }
        }
    }
}
```

Fases de un pipeline

Desarrollo

- Secciones
  - Obligatorias
    - Agent
    - Stages
    - Steps
  - Opcionales
    - Post

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
    post { 1
        always { 2
            echo 'I will always say Hello again!'
        }
    }
}
```

Fases de un pipeline

Desarrollo

- Directivas
  - Environment
  - Options
  - Parameters
  - Triggers
  - Cron
  - Cron
  - Stage
  - Tools
  - Input
  - When

https://www.jenkins.io/doc/book/pipeline/syntax/#declarative-directives

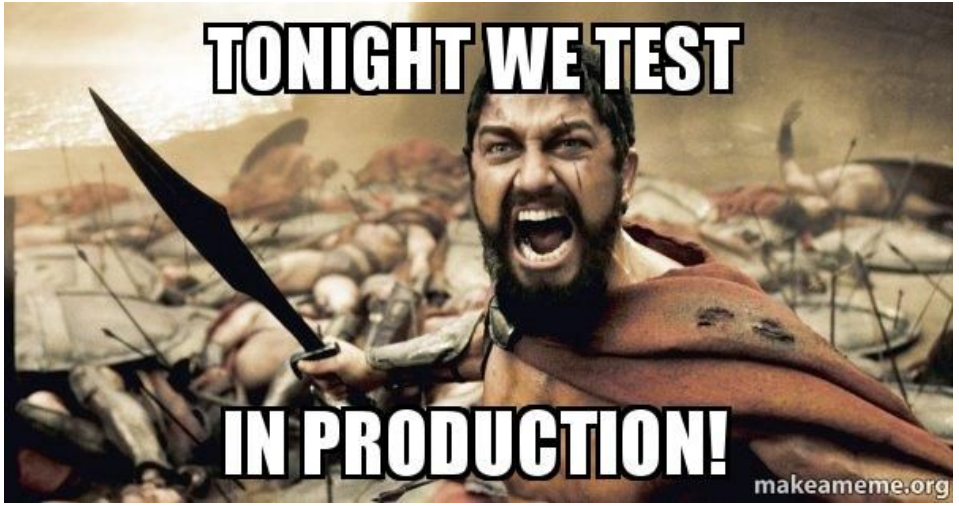
Fases de un pipeline

Desarrollo

- Secuencial
  https://www.jenkins.io/doc/book/pipeline/syntax/#sequential-stages
- Parallelo (fan out)
  https://www.jenkins.io/doc/book/pipeline/syntax/#parallel
- Matrix
  https://www.jenkins.io/doc/book/pipeline/syntax/#matrix-cell-directives

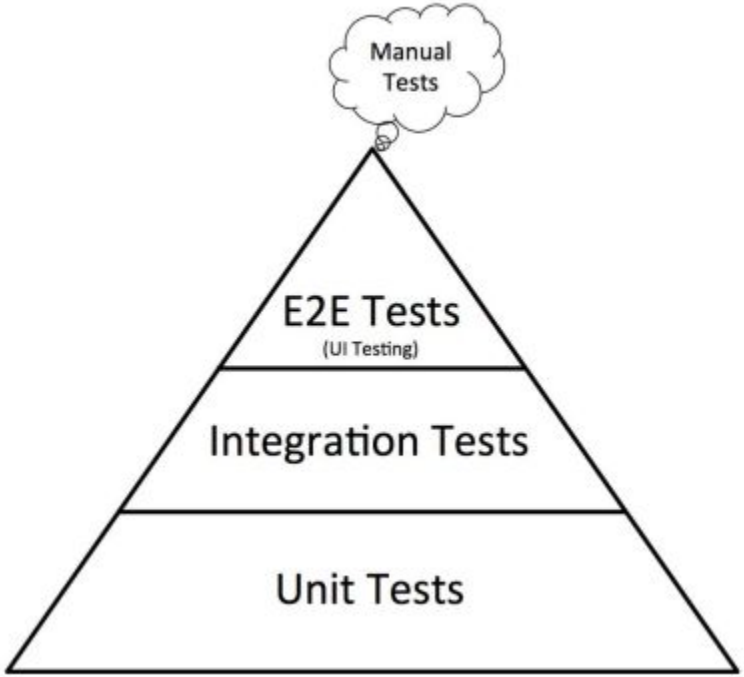
Fases de un pipeline

1. Fundamentos
2. Planificación
3. Arquitectura
4. Desarrollo
5. **Testing**
6. Despliegue
7. Mantenimiento



Fases de un pipeline

Testing



Fases de un pipeline

Testing

- Testing en CI
  - Ejecución de tests como parte del pipeline
  - Resultado binario o ternario
    - Funciona
    - No funciona
    - Funciona “a medias”

Fases de un pipeline

Testing

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
            post {
                always {
                    junit 'target/surefire-reports/*.xml'
                }
            }
        }
    }
}
```

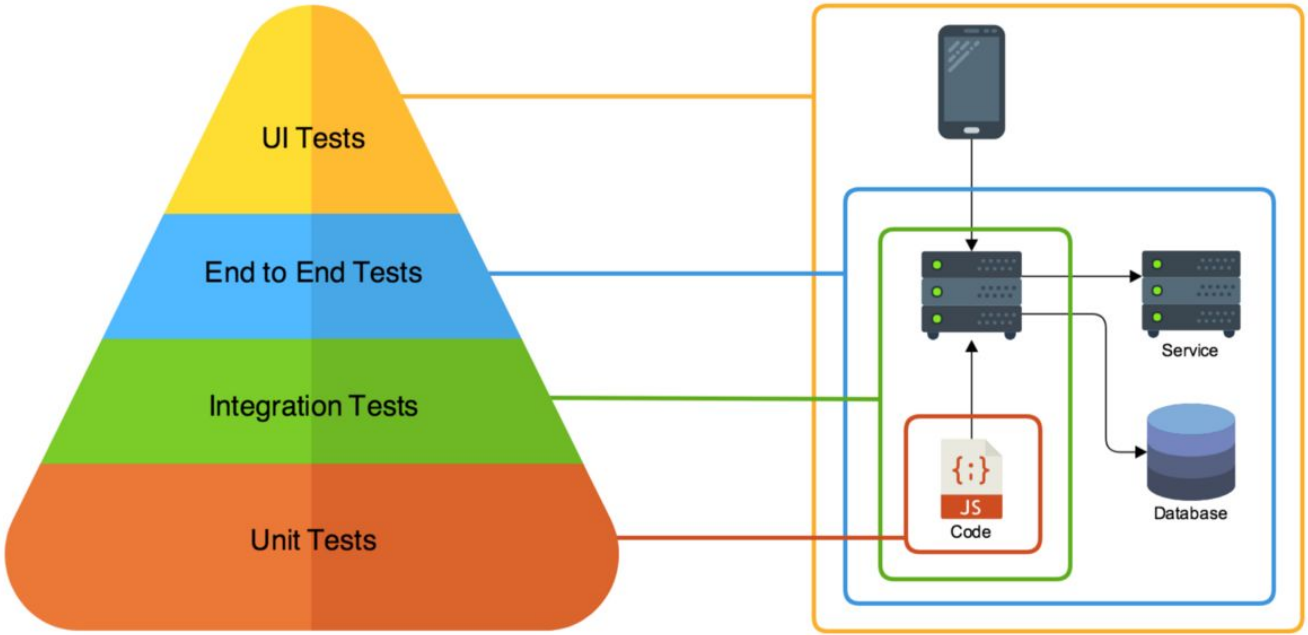
Fases de un pipeline

Testing

- Test unitarios
  - Ejecución en cada rama
- Test de integración
  - Tests en ramas de integración (master)
  - En Merge/Pull requests
- Test End2End
  - Release branch
- UI tests
  - Release branch
  - Producción

Fases de un pipeline

Testing



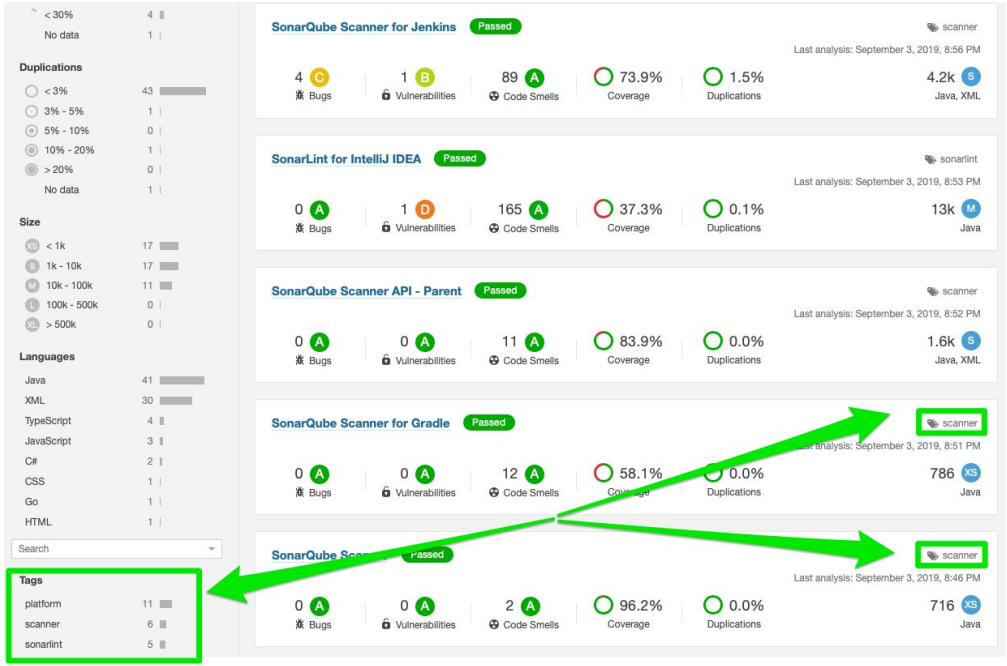
Fases de un pipeline

Testing

- Testing estático
  - Sonarqube
    - Linter
    - Coverage
    - Tests
  - Sentry
    - Exceptions

Fases de un pipeline

Testing



Fases de un pipeline

Testing

SENTRY

Fases de un pipeline

1. Fundamentos
2. Planificación
3. Arquitectura
4. Desarrollo
5. Testing
6. **Despliegue**
7. Mantenimiento

Fases de un pipeline

Despliegue

- Entregar artefacto
  - Versión
  - Entorno

- Tres etapas
  - Desplegar (deploy)
  - Validación (check)
  - Vuelta atrás (roll-back)

Fases de un pipeline

Despliegue

- Desplegar
    - Publicar artefacto (apps, paquetes, librerías…)
    - Aplicar configuración (chef, ansible, puppet, salt...)
    - Actualizar servicio (docker, tomcat, IIS, lambda functions…)
    - Reemplazar servicio (VM, swarm, kubernetes...)
    - Publicar contenido (estáticos html, doc, imágenes…)
    - Aplicar infraestructura (terraform, cloud formation, pulumi…)
    - Actualizar aplicación (django, flask, laravel, grails, rails…)

Fases de un pipeline

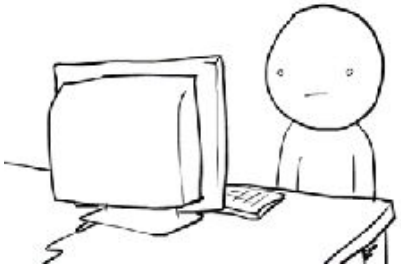
Despliegue

- Validar
  - El despliegue fue correcto?
  - Ejecución de tests de validación
  - Monitorización
  - A/B testing

Fases de un pipeline

Despliegue

- Vuelta atrás (rollback)
    a. Deshacer cambios
    b. Deshacer config
    c. Deshacer datos
    d. Validar todo ok



Fases de un pipeline

1. Fundamentos
2. Planificación
3. Arquitectura
4. Desarrollo
5. Testing
6. Despliegue
7. **Mantenimiento**

Fases de un pipeline

Mantenimiento

- Backups
  - Infraestructura (código)
  - Pipelines (código)
  - Secretos

- Actualizaciones
  - Plugins
  - Incompatibilidades