GitFlow en GitLab

Agenda

Qué es Git
Qué es GitFlow

- Dónde y cuándo aplicarlo?
- Ventajas e inconvenientes
- "Reglas del juego"
- Sintiendo el "flow"

Cómo DevOps se adapta a GitFlow
CREAR REPOSITORIO
• Crear repositorio en Gitlab
PRÁCTICA

- Requisitos
- Tabla de cambios
- Cambio #1 - feature/login-con-facebook
- Cambio #2 - feature/exportar-reporte-drive
- Cambio #3 - hotfix/login-linkedin
- Cambio #4 - release/v1.2.0

Agenda

Qué es Git
Qué es GitFlow

- Dónde y cuándo aplicarlo?
- Ventajas e inconvenientes
- "Reglas del juego"
- Sintiendo el "flow"

Cómo DevOps se adapta a GitFlow
CREAR REPOSITORIO
• Crear repositorio en Gitlab
PRÁCTICA

- Requisitos
- Tabla de cambios
- Cambio #1 - feature/login-con-facebook
- Cambio #2 - feature/exportar-reporte-drive
- Cambio #3 - hotfix/login-linkedin
- Cambio #4 - release/v1.2.0

Que es git

Software de control de versiones

Permite mantener un historico de las distintas versiones de un desarrollo

Sistema de control de versiones distribuido, open source y gratuito

Creado por Linus Torvalds, para su uso en el desarrollo del kernel de Linux

Rápido, sólido, estable y fácilmente conectable

Desde cero, o a partir de Subversion y CVS

Instalacion

+ Disponible en Linux, Mac OS X, Windows y Solaris

Debian/Ubuntu

```
$ apt-get install git
```

Fedora

```
$ yum install git
```

Gentoo

```
$ emerge --ask --verbose dev-vcs/git
```

Qué es GitFlow

- Basado en GIT- sistema de control de versiones más utilizado en desarrollo.
- Se basa en un conjunto de prácticas y flujos que aportan control y evitan conflictos en el proceso de desarrollos complejos.
- Todo gira entorno al control y gestión de ramas intermedias temporales, fijas y sus migraciones y fusiones.
- Integración continua que aporta un marco de trabajo definido entre todos los equipos que participan del desarrollo de un mismo producto (repo).
- Plataforma de plugins que nos ayuda en la CLI para crear y gestionar el ecosistema

Dónde y cuándo aplicarlo?

- **Se recomienda aplicarlo en**
  - Entornos semi complejos y complejos de desarrollo
  - Plataforma basada en microservicios con repos independientes
  - Proyectos OpenSource
  - Proyectos single repo muy contributivos
  - Equipos de + 2 personas
  - Proyectos que utilicen Agile framework
  - Proyectos que quieran utilizar mecanismos de test intermedios y unitarios con GitOps

- **No se recomienda en:**
  - Proyectos monolíticos
  - Proyectos con 2 o menos desarrolladores
  - Proyectos que no desarrollan funcionalidades de forma independiente
  - Proyectos con personal que desconozca el método y no tenga tiempo MVP

Ventajas

1. Se adapta al formato de “Sprint” basado en metodologías ágiles entregando de forma continua con CI/CD ya que el entregable es una RELEASE.
2. Evita conflictos entre ramas
3. Protege el estado y fiabilidad de la rama master
4. Mejora la trazabilidad del código
5. Facilita la cooperación colectiva creando ramas por feature
6. El concepto de hotfix agiliza resoluciones de problemas en entornos reales

Desventajas

1. Es primordial que el equipo de desarrollo y devops conozcan el método y apliquen los flujos y reglas.
2. Agrega complejidad el proceso y pasos intermedios
3. Todo gira alrededor del concepto release
4. Es recomendable crear entornos intermedios temporales
5. Complica el entorno de integración
6. Debemos tener personas que en momentos de conflictos tengan la capacidad de coordinar conflictos y/o dependencias y conozcan los plugins de git flow.
7. Es recomendable utilizar herramientas complejas y de testing pre integración que garanticen y den visibilidad del estado de las migraciones entre ramas.
8. Es preciso usar herramientas de los repos como PR para mejorar la parte contributiva

Reglas del juego

Deben existir dos ramas principales:

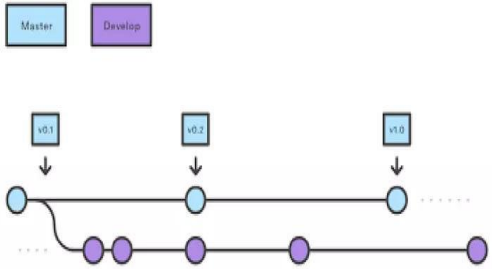
- **master** → eje de la verdad
- **develop** → eje del desarrollo

GitFlow INIT --> prefijos para las ramas auxiliares:

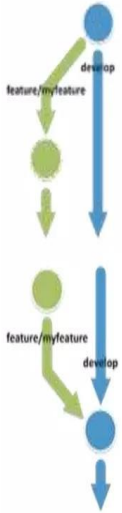
- **feature/**
- **release/**
- **hotfix/**
- **bugfix/**

KEEP CALM AND GO WITH THE FLOW

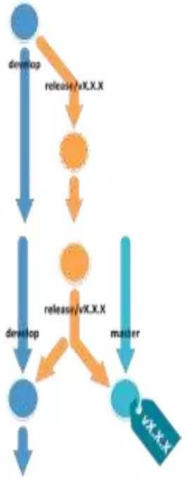
Los flujos....



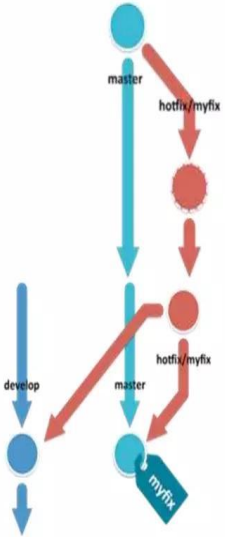
Nueva feature



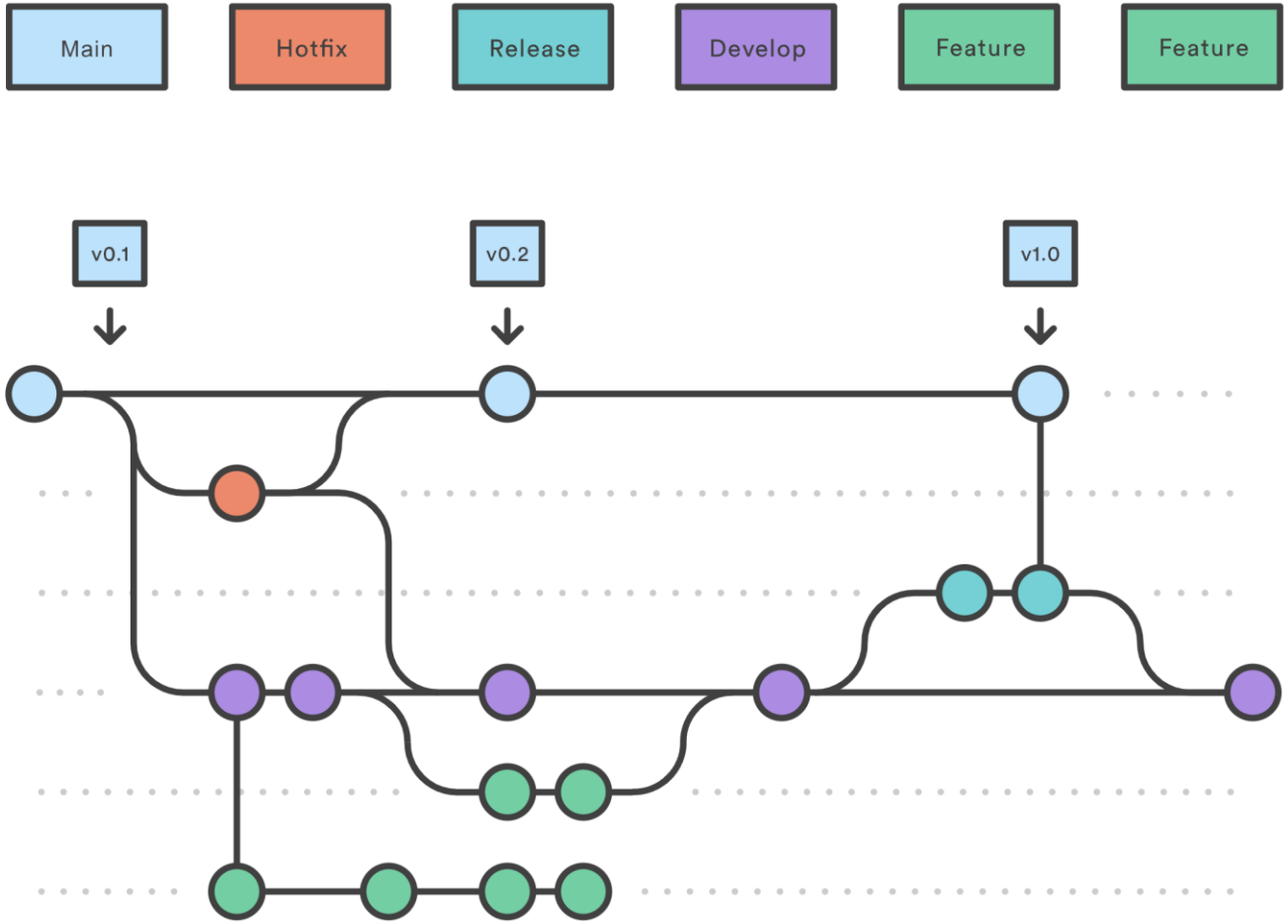
Nueva Release



Nuevo Hotfix



Sintiendo el flow

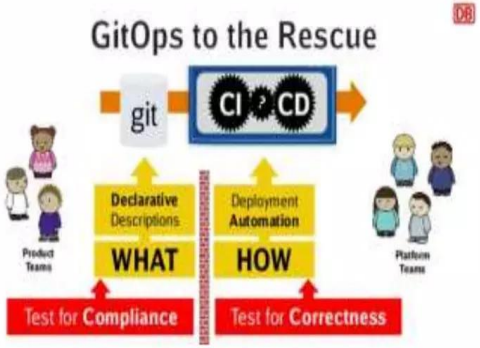


Cómo DevOps Se adapta a GitFlow?

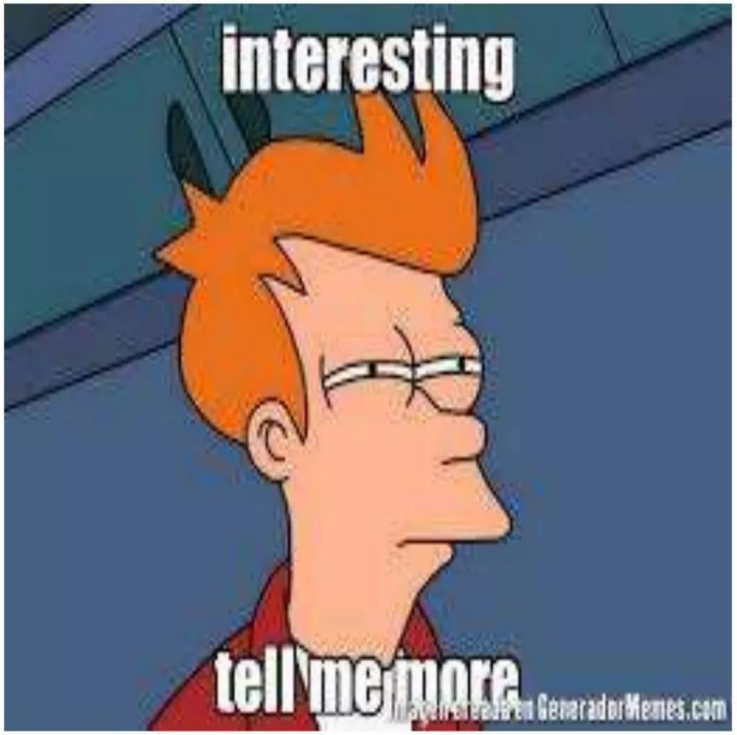
Conjunto de herramientas y prácticas basadas en automatismos centralizadas en el eje del desarrollo contínuo:

GIT OPS! TO THE RESCUE!

- **Single point of truth = GIT**
- **Procesos automáticos disparados por eventos de Git (merge, finish feature, PR...)**
- **Gestión de entornos automáticos y estáticos**
- **Automatización flujos con aprobación por QA**
- **Definición de artefactos y entregables inmutables**
- **Adaptar pipelines aisladas para trabajar de forma coordinada con los procesos CI/CD**



Cómo DevOps Se adapta a GitFlow?



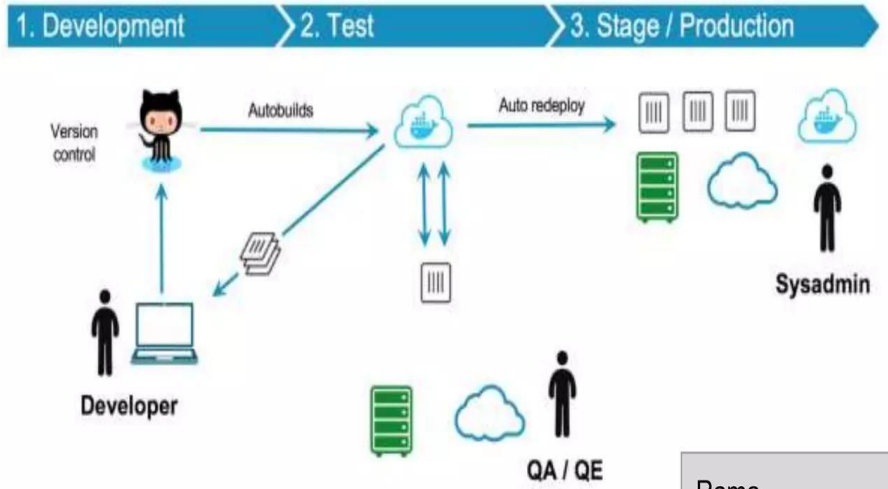
Cómo GitOps Se adapta a GitFlow?

- Para todo DevOps es primordial entender a la perfección el ciclo de desarrollo de su empresa para mejorarlo y automatizarlo.
- Con la llegada de los contenedores docker, los flujos de desarrollo han evolucionando permitiendo testear de forma segura el comportamiento de un producto sin que existan diferencias entre entornos.
- No existe un modelo de GitOps único y en cada caso existe una adaptación. GIT es el eje de unión entre desarrollo y operaciones.
- Es primordial poder testear fases de calidad y versionar la infraestructura con entregables concretos que permita portar un artefacto desde un desarrollador hasta producción.

Desafíos de GITOPS?

- Identificar entornos (pre/prod/stag/devel...)
- Identificar hooks y eventos para automatizar
- Definir herramientas complejas con entornos dinámicos
- Los flujos han de seguir siempre el mismo patrón
- Orientar entornos dinámicos en base a PR
- Es preciso ir de la mano de QA para determinar aprobaciones manuales para el paso a producción
- Definir métodos de deployment (blue/green/canary)
- Definir rollbacks y procedimientos
- Determinar procesos de DB migrations
- ...

Cómo DevOps Se adapta a GitFlow?



Rama	Entorno
Feature - PR	Entorno volátil
Develop	Integración
Release	Staging
Master	Prod

LABS TIME!

Practica

Tabla de cambios

ID	DESCRIPCIÓN CORTA	RAMA DE TRABAJO	RAMA ORIGEN	RAMA DESTINO
Cambio #1	Implementar inicio de sesión con Facebook	**feature**/login-con-facebook	develop	develop
Cambio #2	Exportar reporte de usuarios a Google Drive	**feature**/exportar-reporte-drive	develop	develop
Cambio #3	Error al iniciar sesión con Linkedin (v1.1.0)	**hotfix**/login-linkedin	master	master y develop
Cambio #4	liberar versión v1.2.0	**release**/v1.2.0	develop	master y develop

Cambio #1 - feature/login-con-facebook

- git checkout -b develop | git push -u origin develop
- git checkout -b feature/login-con-facebook
- touch login-con-facebook.txt
- git add login-con-facebook.txt
- git commit -m "Se implemento el inicio de sesion con Facebook"
- git push -u origin feature/login-con-facebook

MERGE REQUEST

- Menú > Merge Requests > New merge request

Cambio #2 - feature/exportar-reporte-drive

- git checkout develop | git pull origin develop
- git checkout -b feature/exportar-reporte-drive
- touch exportar-reporte-drive.txt
- git add exportar-reporte-drive.txt
- git commit -m "Soporte para exportar reportes de usuarios a Google Drive"
- git push -u origin feature/exportar-reporte-drive

MERGE REQUEST

- Menú > Merge Requests > New merge request

Cambio #3 - hotfix/login-linkedin

- git checkout master | git pull origin master | revisar git graph
- git checkout -b hotfix/login-linkedin
- touch login-linkedin.txt
- git add login-linkedin.txt
- git commit -m "Se soluciono el error al iniciar sesion con Linkedin"
- git push -u origin hotfix/login-linkedin

MERGE REQUEST

- Menú > Merge Requests > New merge request
- Merge request a master
- git pull origin master
- git tag -a v1.1.0 -m "version 1.1.0"
- git push -u origin v1.1.0
- Merge request a develop
- git pull origin develop

Cambio #4 - release/v1.2.0

Aprobar merge request en Gitlab

- git checkout develop | git pull origin develop | revisar git graph
- git checkout -b release/v1.2.0
- touch ajustes-release-v1-2-0.txt
- git add ajustes-release-v1-2-0.txt
- git commit -m "ultimo ajuste"
- git push -u origin release/v1.2.0

MERGE REQUEST

- Menú > Merge Requests > New merge request
- Merge request a master
- git pull origin master
- git tag -a v1.2.0 -m "version 1.2.0"
- git push -u origin v1.2.0
- Merge request a develop
- git pull origin develop

Q&A

GRACIAS!!