GitFlow - DevOps + Jenkins X Demo

Agenda

- Qué es GitFlow
  - Dónde y cuándo aplicarlo?
  - Ventajas e inconvenientes
  - “Reglas del juego”
  - Sintiendo el “flow”
- Cómo DevOps se adapta a GitFlow
- Jenkins X - LABS
- Q&A

geko

Hello World - `$whoami`

Tech Lead, Senior DevOps, Speaker, Technical Coach, AWS certified and Internet Infrastructure Specialist

- CEO at Geko Cloud
- Vueling
- Privalia
- Wallapop
- Emagister
- ...

https://www.linkedin.com/in/joseluissanchez/
https://geko.cloud

geko

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

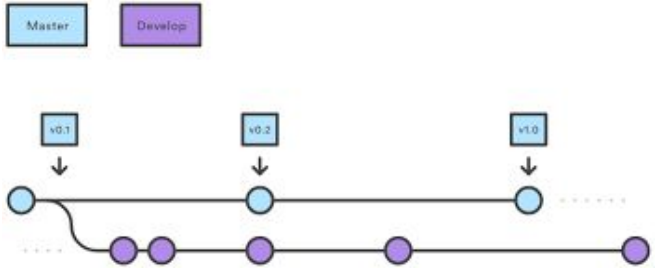
GitFlow INIT --> prefijos para las ramas auxiliares:

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

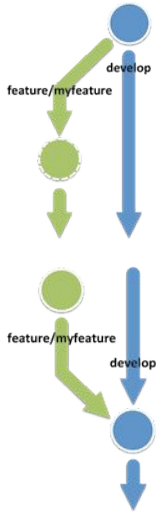
KEEP CALM AND GO WITH THE FLOW

keep-calm.net

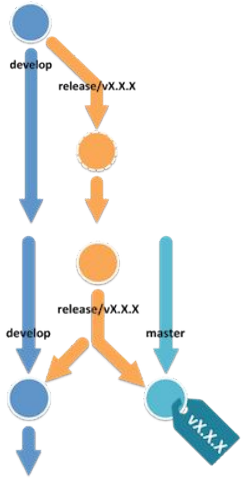
Los flujos....



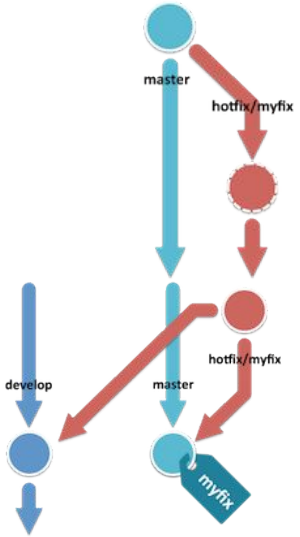
Nueva feature



Nueva Release



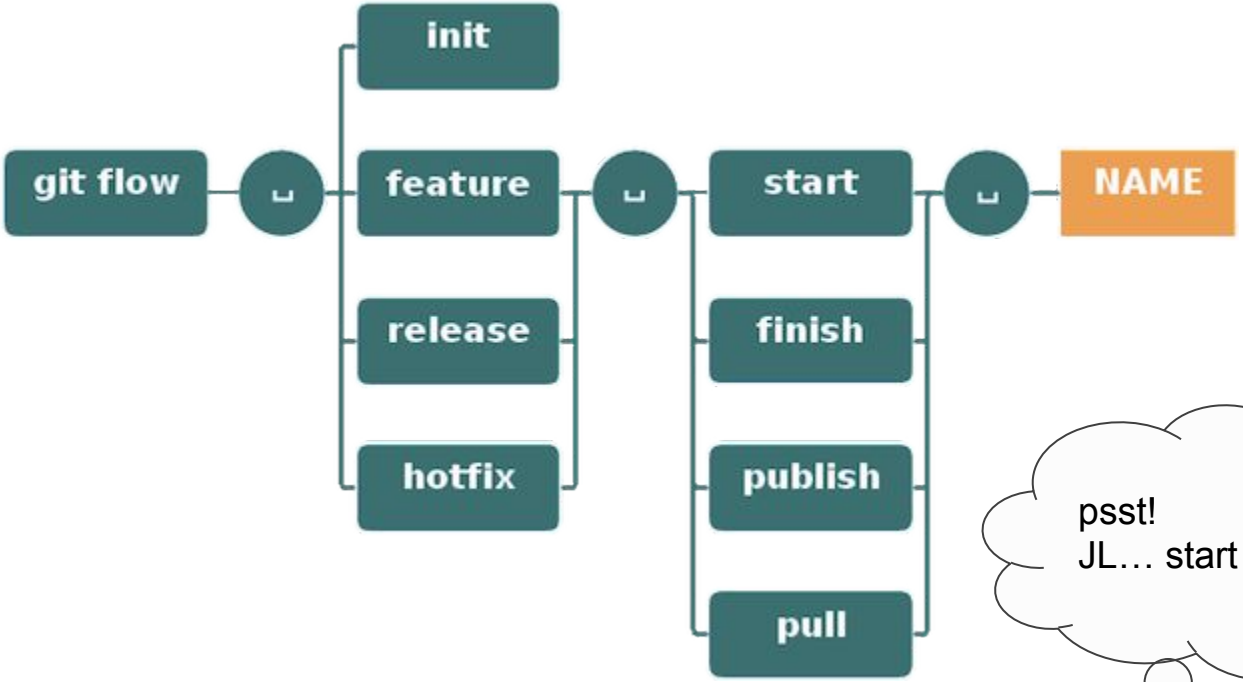
Nuevo Hotfix



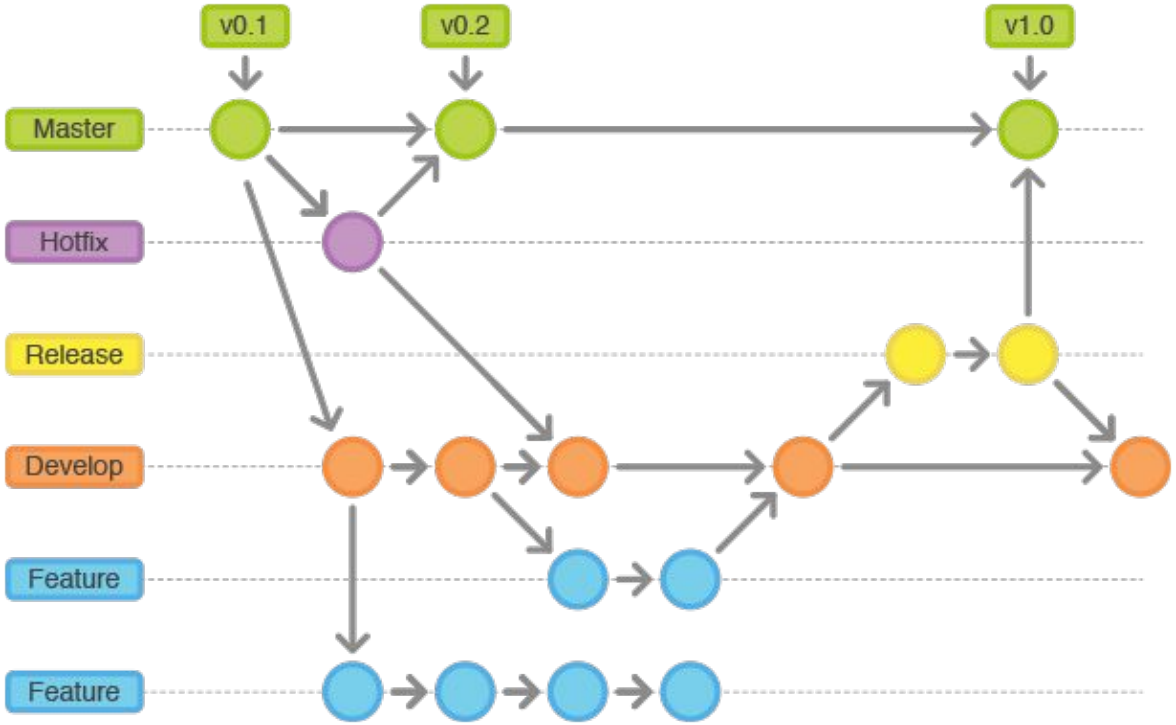
Git Flow CLI

```
$ brew install git-flow-avh
```

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



Sintiendo el flow

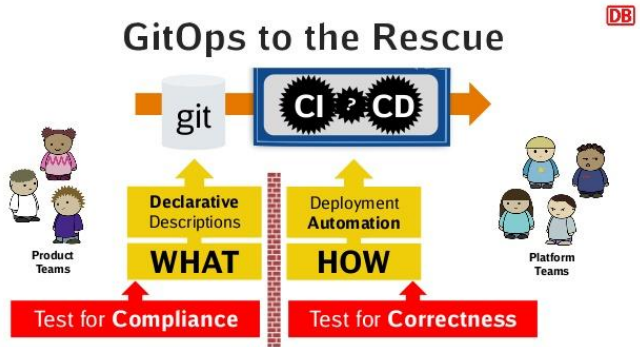


Cómo DevOps Se adapta a GitFlow?

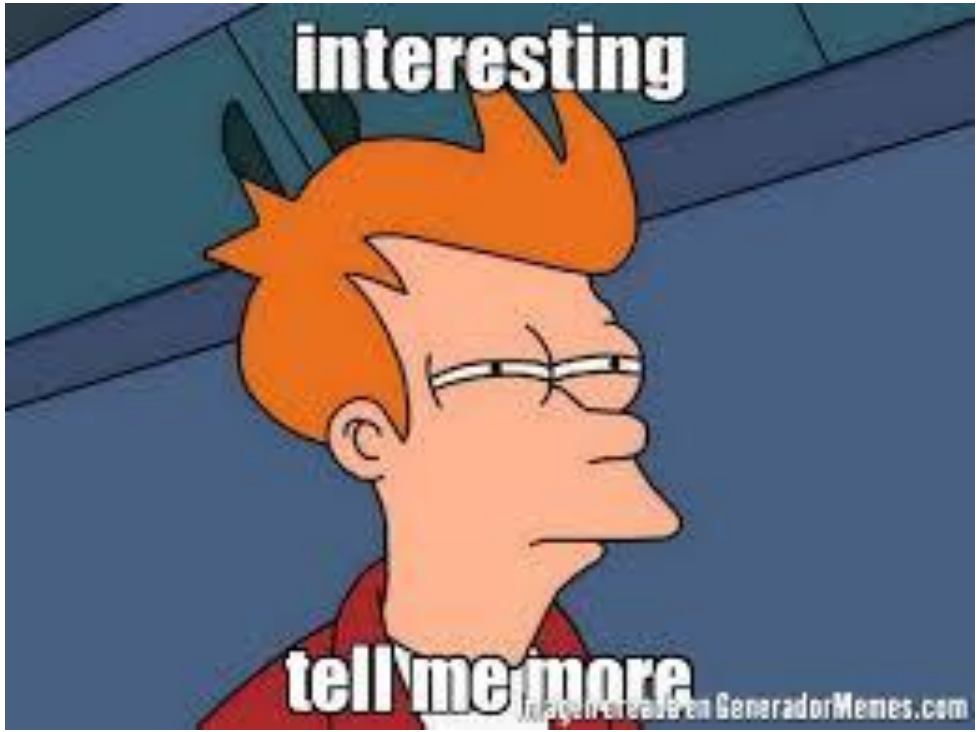
Conjunto de herramientas y prácticas basadas en automatismos centralizadas en el eje del desarrollo contínuo:

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?



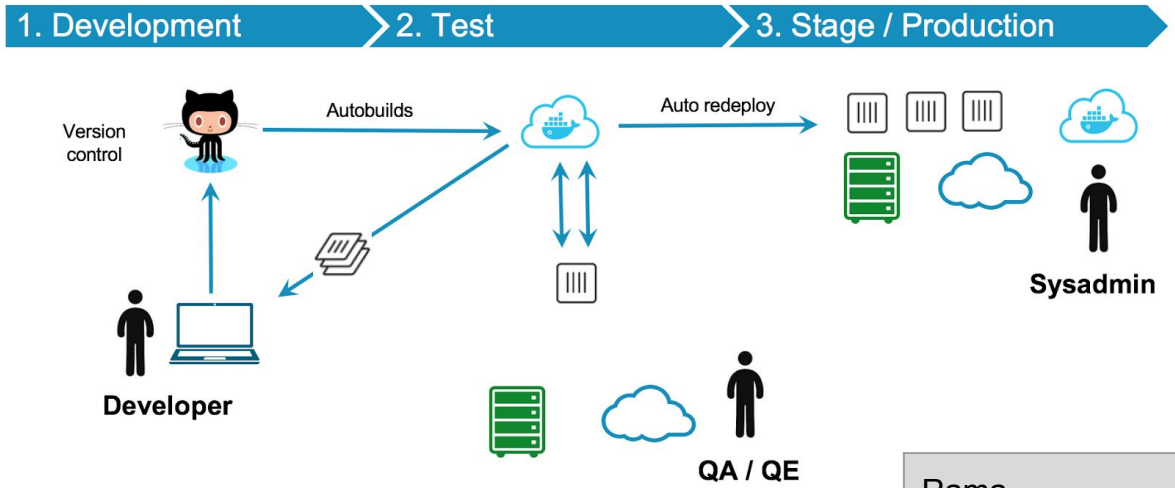
Cómo GitOps Se adapta a GitFlow?

- 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

Demo Time!

CLOUD NATIVE COMPUTING FOUNDATION

CD FOUNDATION
A Neutral Home for the Next Generation of Continuous Delivery Collaboration
cd.foundation

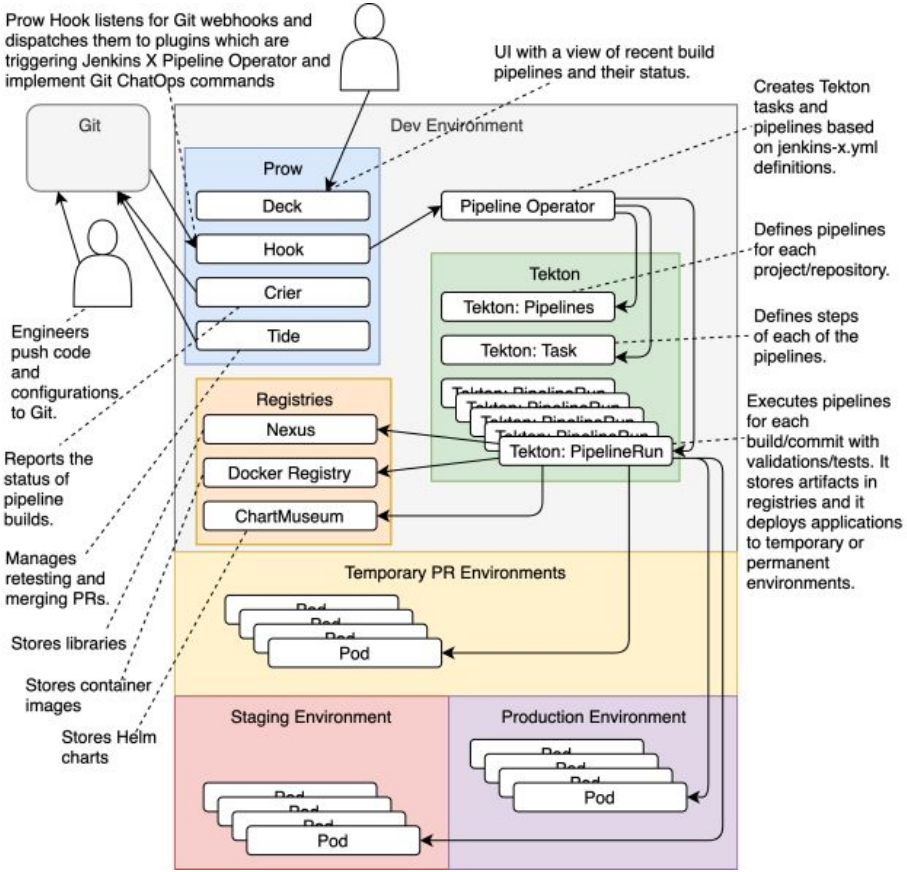
Jenkins X

- Como dice su creador: *‘open source opinionated way to do continuous delivery with Kubernetes, natively’*.

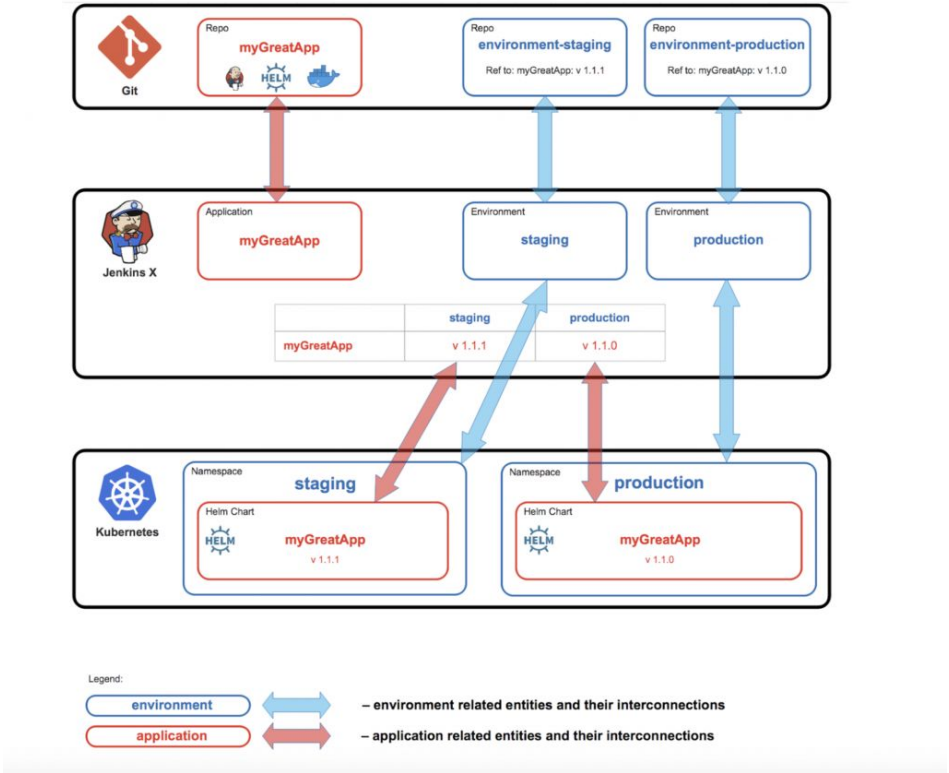
- Jenkins X permite desplegar las aplicaciones en un ecosistema Kubernetes. Usa las mejores prácticas de DevOps y mejora la calidad y velocidad de CI/CD.

- Ejecuta un set de herramientas para cumplir un objetivos: simplificar los procesos CI/CD basándonos en GIT: Helm, Tekton, Prow, Draft, monocular, ChartMuseum, Nexus, etc.

Jenkins X flow diagram



Jenkins X flow diagram



LABS TIME!

Q&A

Geko Cloud

Feel the Geko way

https://geko.cloud
info@geko.cloud