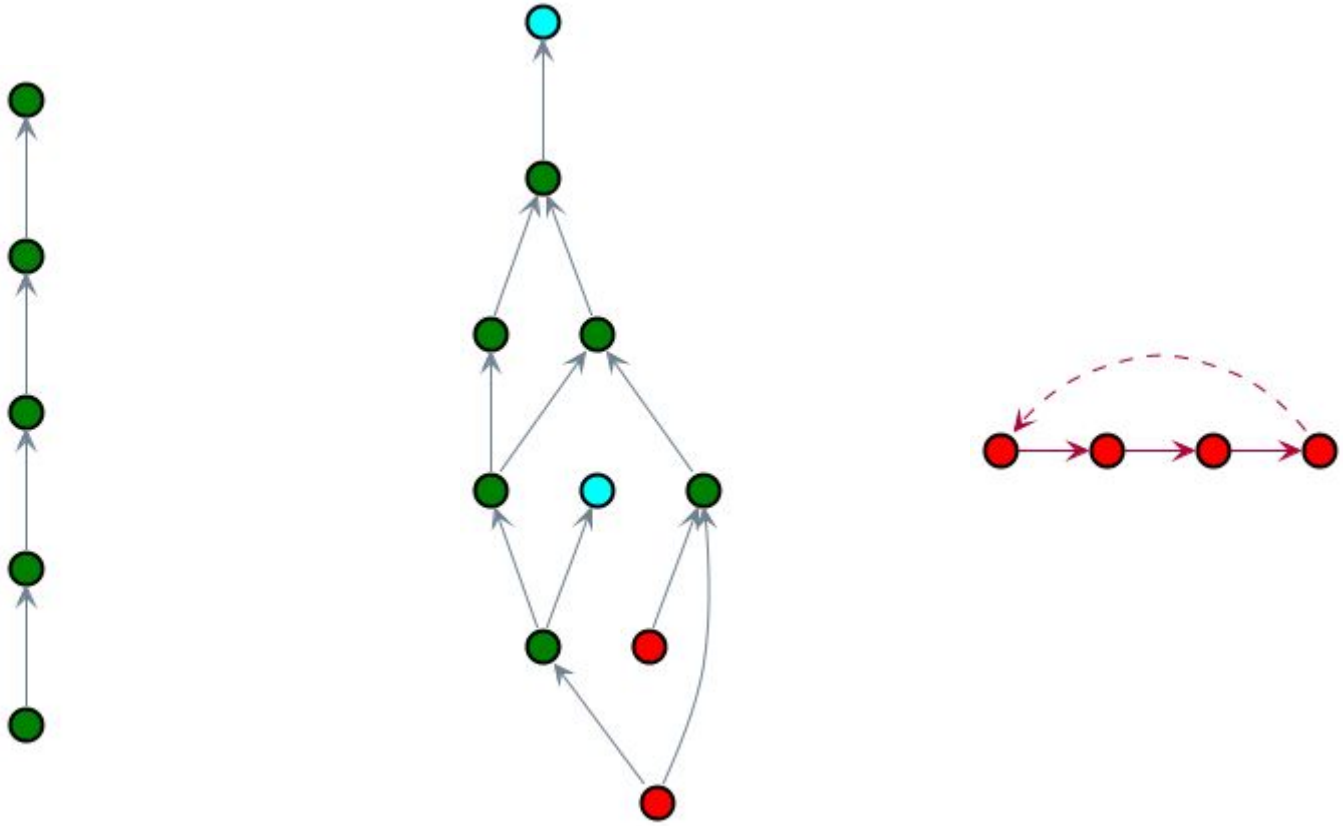
Git

Commit

- Algún estado del sistema
- Queremos compartirlo con el resto del equipo

Historia

- Unico requisito: Sin ciclos
  - ⇒ Reflejamos nuestras intenciones en la historia



Comunicacion

1) **Codigo**
2) Fecha / orden
3) **Mensajes de commit**
4) **Forma de los branches**

Comunicacion

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

https://xkcd.com/1296/

Commits Atomicos

Un commit que aplica **un** cambio **completo**.

- Una sola razón para todo lo que cambió
- Todo lo que cambia por la misma razón cambia junto

Mensajes de Commit

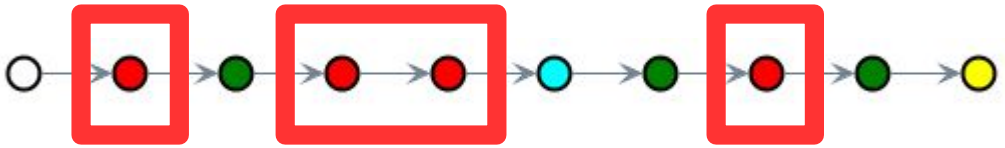
```
git-class: Agrega diapos de mensajes de
commit
                (linea en blanco)
Este texto explica el contexto del
cambio, incluyendo cosas como:
- Efectos que no resultan obvios del
  codigo
- Problema que resuelve este commit
- Metadata que leen otros sistemas
  (issues resueltos, firmas, etc)
```

Historia Lineal

- Proyectos
  - Simples
  - Poca superposition

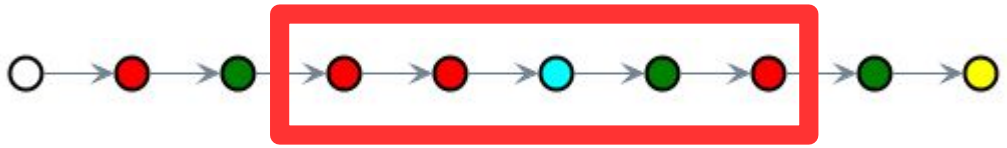
Historia Lineal

- Desventajas
  - Cuesta separar líneas de trabajo



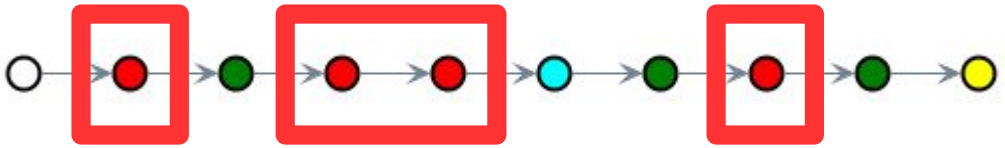
Historia Lineal

- Desventajas
  - Cuesta separar líneas de trabajo
  - Los errores impactan a otros



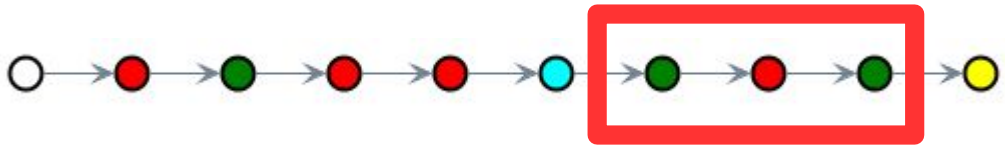
Historia Lineal

- Desventajas
  - Cuesta separar líneas de trabajo



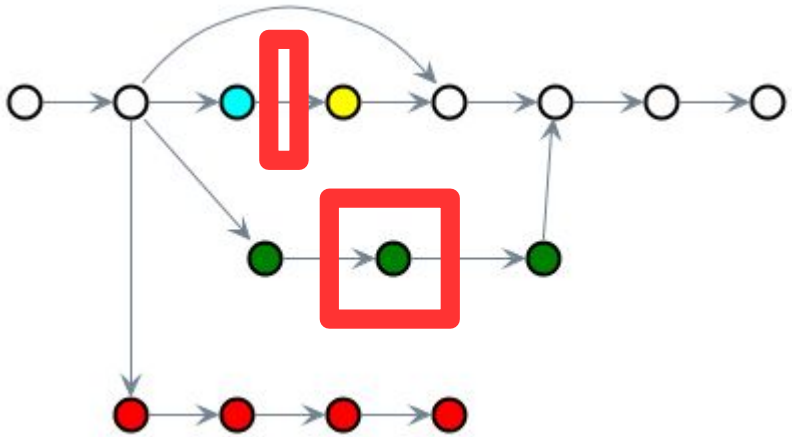
Historia Lineal

- Desventajas
  - Cuesta separar líneas de trabajo
  - Los errores impactan a otros
  - Estados intermedios inesperados



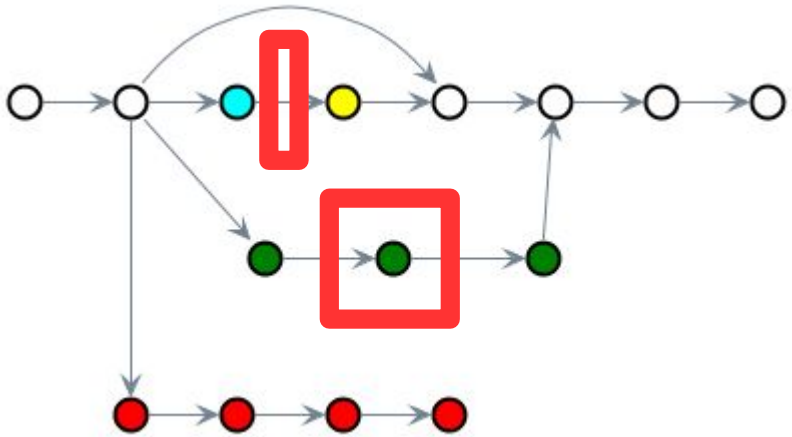
Feature Branches

- Branches dedicados a hacer un feature



Feature Branches

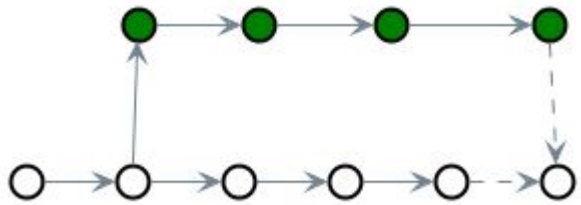
- Branches dedicados a hacer un feature
  - Menos cambios para hacer
  - Menos gente trabajando en cada branch
  - El resto del equipo ve todos los cambios juntos



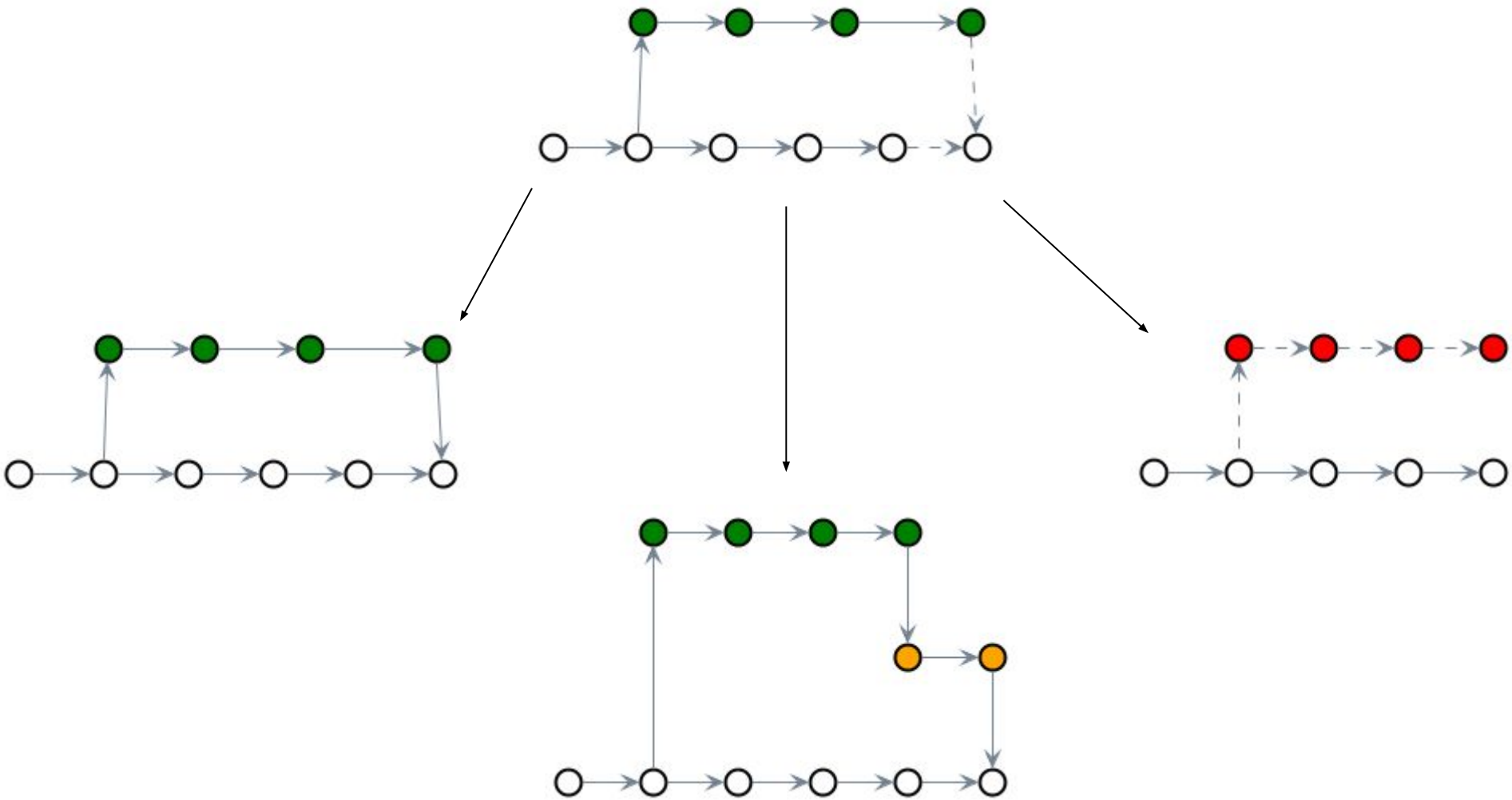
Pull Requests

Tambien llamadas **Merge Requests**

- Antes del merge, hay una oportunidad de verificar:
  - Calidad del código (Dev)
  - Calidad de la aplicación (QA)
  - Si el cambio conviene
  - Si el cambio tuvo éxito

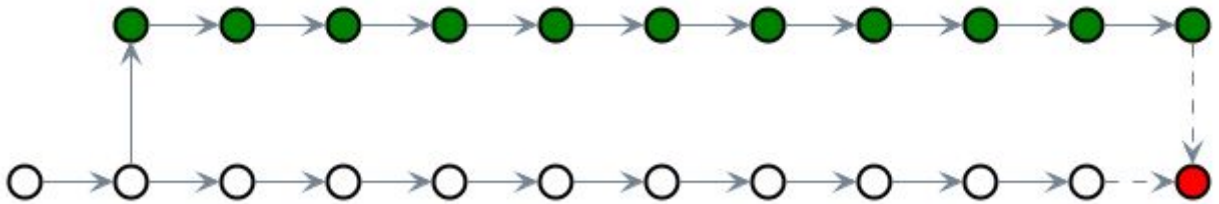


Pull Requests



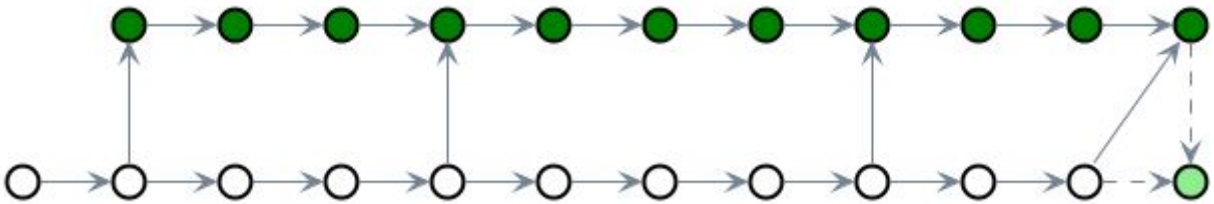
Branches a Largo Plazo

- Si mantenemos un branch separando durante demasiado tiempo:
  - Bugs encontrados y arreglados (por separado) de ambos lados
  - Cambios diferentes a un mismo lugar
  - **El merge final**



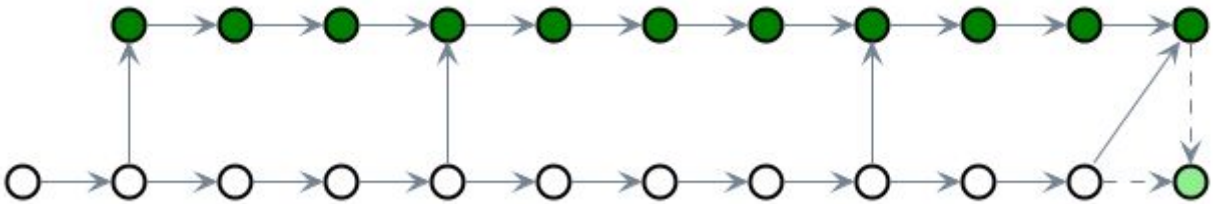
Branches a Largo Plazo

- Haciendo merges intermedios:
  - Menos cambios en cada merge
  - Esos cambios son más recientes
  - Evitamos duplicar trabajo



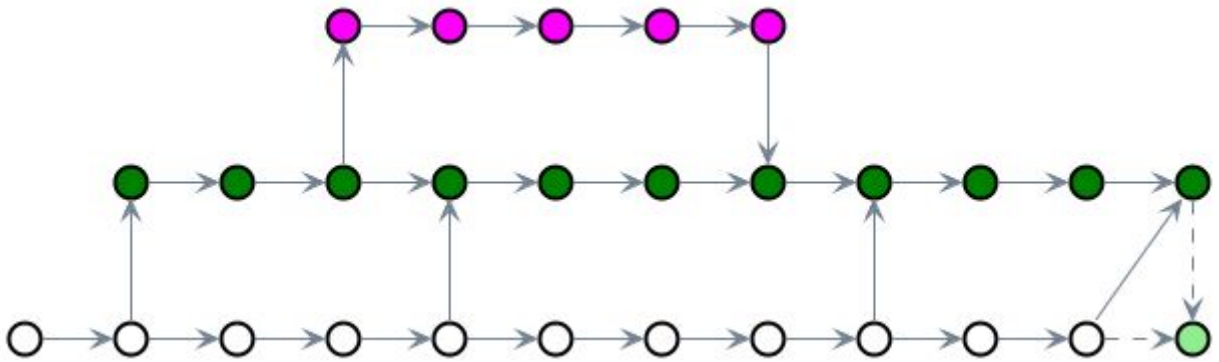
Branches a Largo Plazo

- Dentro del branch:
  - Cuesta separar líneas de trabajo
  - Los errores impactan a otros
  - Estados intermedios inesperados
- Esto recien lo vimos!



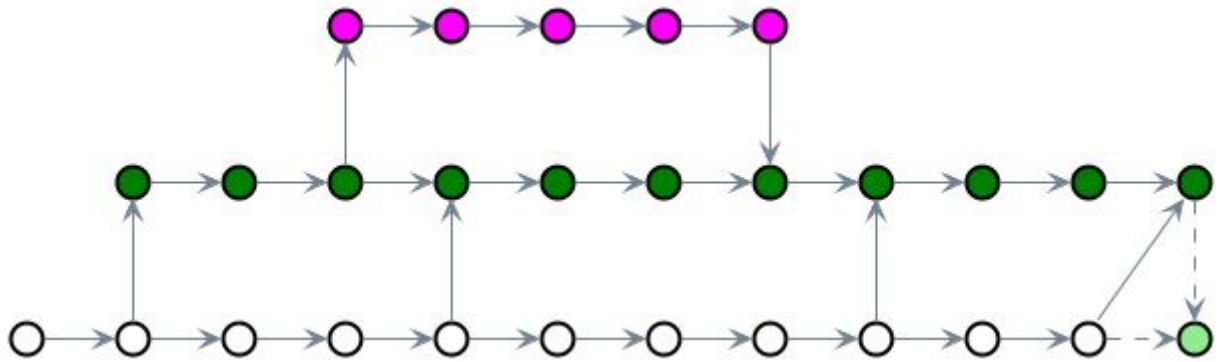
Branches a Largo Plazo

- Un branch a largo plazo puede tener feature branches internos!



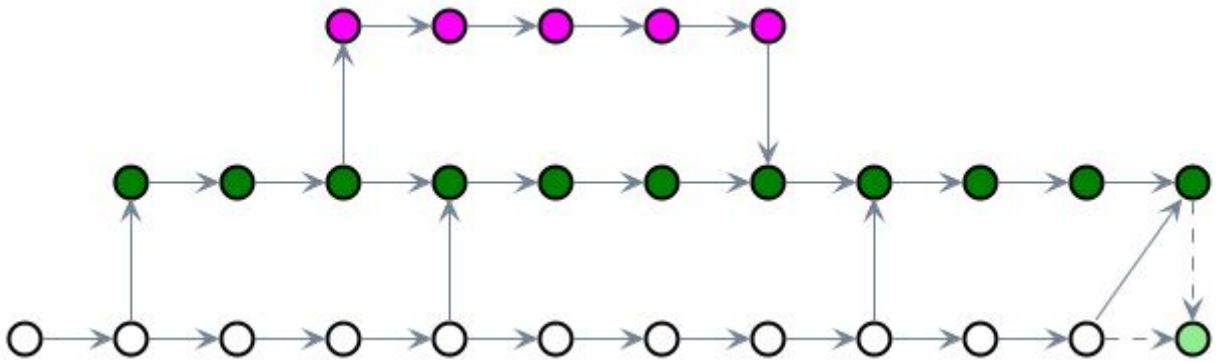
Branches a Largo Plazo

- Un branch a largo plazo puede tener feature branches internos!
- La diferencia es que propósito le da el equipo a cada branch

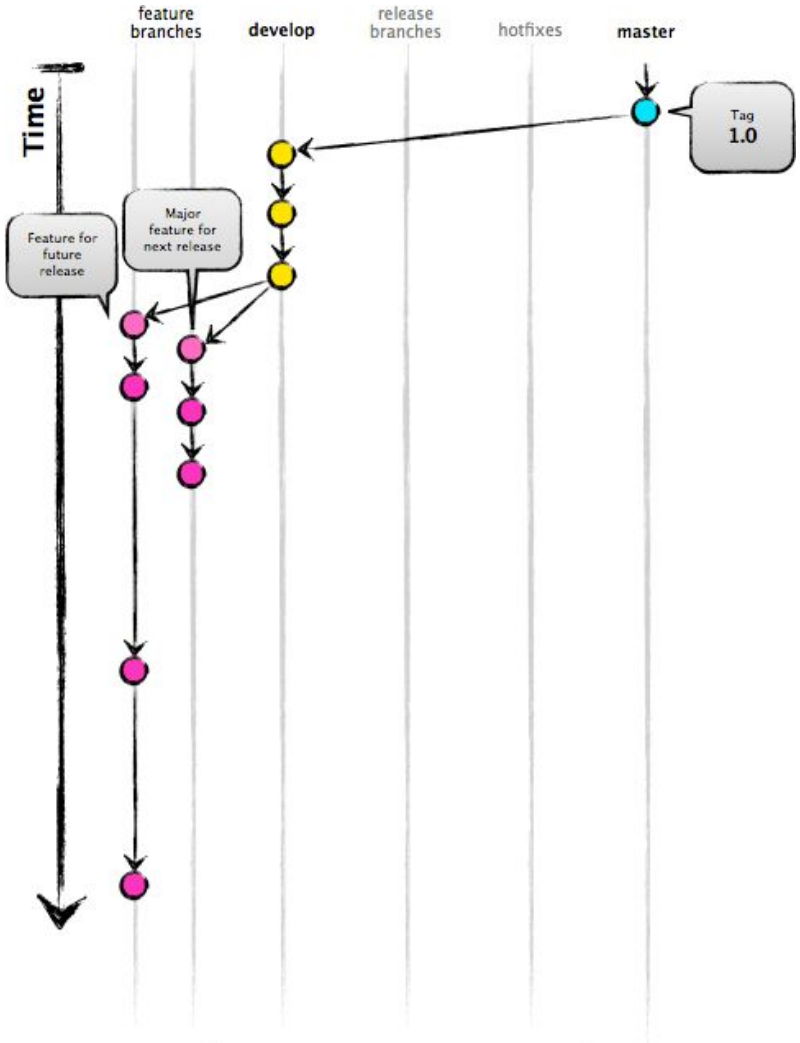


Branches a Largo Plazo

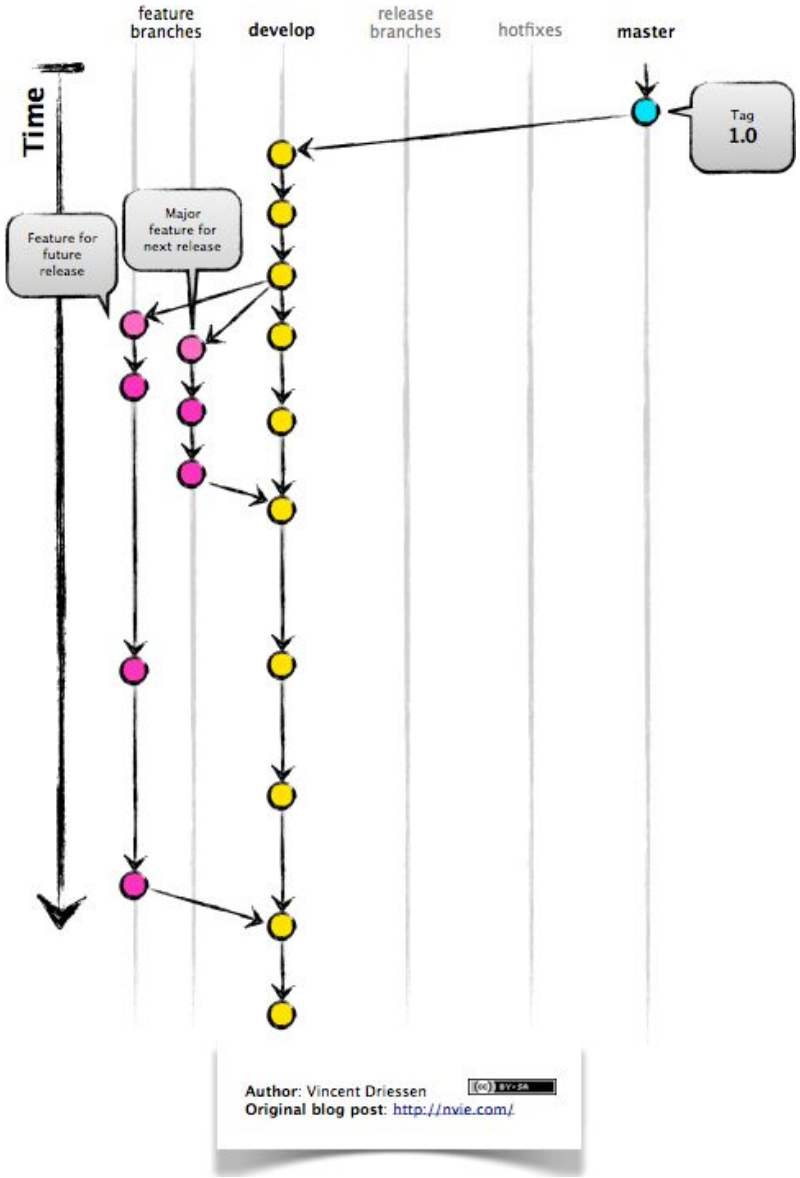
- La diferencia es que propósito le da el equipo a cada branch. Usualmente:
  - master: Cada commit es una entrega al cliente
  - develop: Cada commit es un build interno



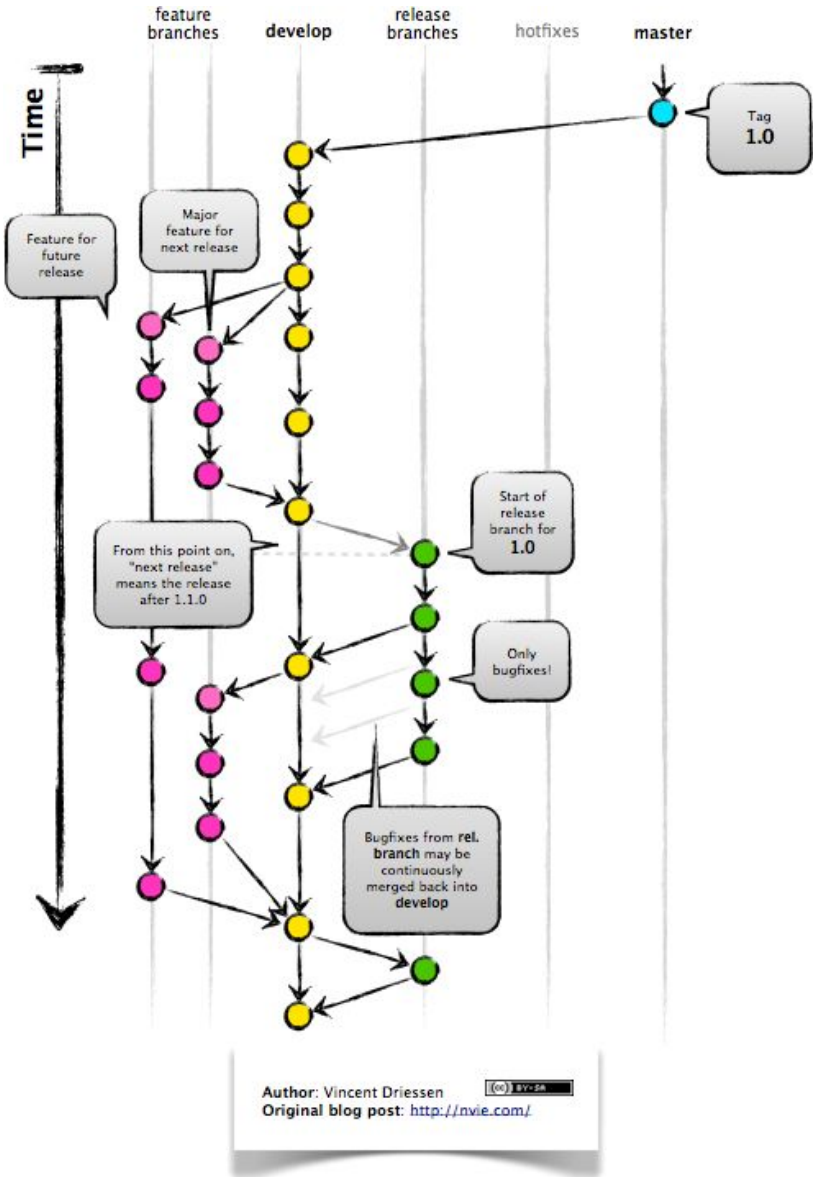
Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Author: Vincent Driessen
Original blog post: http://nvie.com/

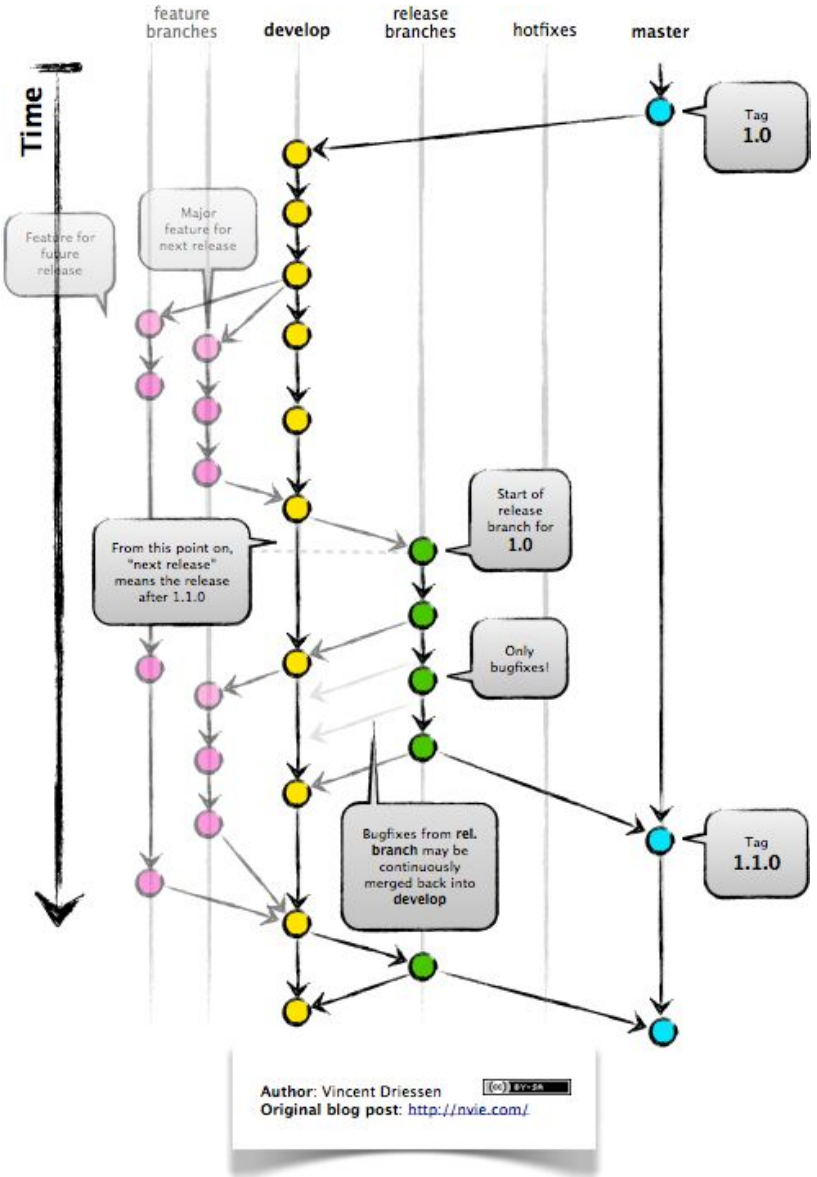
Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Author: Vincent Driessen
Original blog post: http://nvie.com/

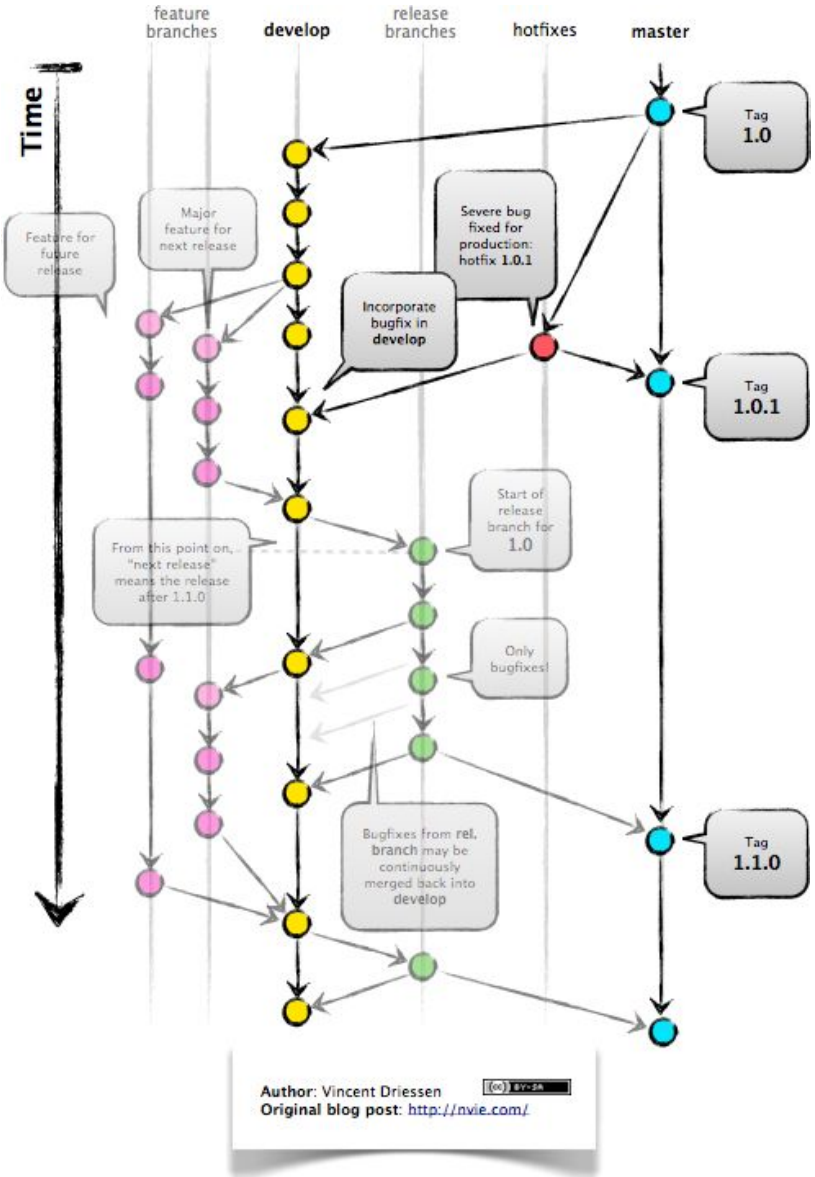
Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Start of release branch for 1.0
From this point on, "next release" means the release after 1.1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Author: Vincent Driessen
Original blog post: http://nvie.com/

Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Tag 1.0
Feature for future release
Major feature for next release
Start of release branch for 1.0
From this point on, "next release" means the release after 1.1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.1.0
Author: Vincent Driessen
Original blog post: http://nvie.com/

Git Flow


feature branches
develop
release branches
hotfixes
master
Time
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 1.0.1
Incorporate bugfix in develop
Tag 1.0
Tag 1.0.1
Start of release branch for 1.0
From this point on, "next release" means the release after 1.1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.1.0
Author: Vincent Driessen
Original blog post: http://nvie.com/

Recursos

- Try Git
- How to Write a Git Commit Message
- Managing your work on GitHub
- Git Flow (Post Original)