GIT FLOW

¿Qué es Git-flow?

- Gitflow es un diseño de workflow de Git que fue publicado y popularizado por Vincent Driessen
- El workflow de Gitflow define un modelo de ramas estricto, diseñado en torno a la versiones del proyecto
- Esto proporciona un marco robusto para la gestión de proyectos más grandes
- Ideal para proyectos que tienen un ciclo de release planificado
- Este workflow no agrega ningún concepto o comando nuevo más allá de lo que se requiere para el workflow de la rama de características
- En cambio, asigna funciones muy específicas a diferentes ramas y define cómo y cuándo deberían interactuar.
- Además de las ramas de “features”, se usan ramas individuales para preparar, mantener y registrar releases.
- Por supuesto, también puede aprovechar todos los beneficios del workflow de la rama Feature: pull requests, experimentos aislados y colaboración más eficiente

¿Qué es Git-flow?

- Es una idea abstracta de un workflow de Git.
- Esto significa que dicta qué tipo de ramas configurar y cómo combinarlas
- git-flow es una herramienta de línea de comando real que tiene un proceso sencillo de instalación
- Disponible en múltiples sistemas operativos: OSX, Windows, Linux
- Git-flow es un “envoltorio” alrededor de Git
- El comando `git flow init` es una extensión del comando `git init` predeterminado y no cambia nada en su repositorio que no sea crear ramas por el usuario

¿Cómo funciona Git-flow?

- En lugar de trabajar con una única rama maestra, nuestro workflow utiliza dos ramas para registrar el historial del proyecto.
  - La rama master almacena el historial de versiones oficial
  - La rama develop sirve como rama de integración de las “features” añadidas
- Es conveniente etiquetar (tags) todas las confirmaciones en la rama master con un número de versión
- El primer paso es completar la rama master por defecto con una rama de desarrollo.

```
$ git branch develop
$ git push -u origin develop
```

- Esta rama contendrá el historial completo del proyecto, mientras que la master contendrá una versión abreviada. Otros desarrolladores deberían ahora clonar el repositorio central y crear una rama de seguimiento de develop

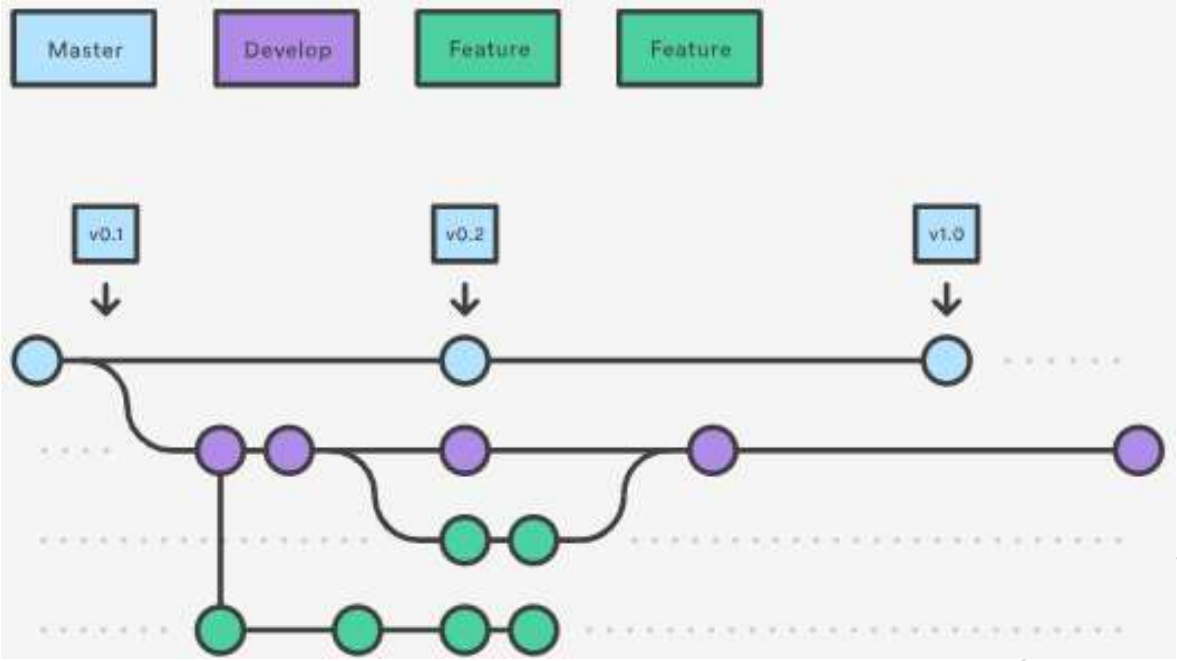
¿Cómo funciona Git-flow?

- La ejecución de `git flow init` en un repositorio existente creará la rama de develop:

```
$ git flow init
Initialized empty Git repository in ~/project/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
Feature branches? [feature/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
$ git branch
* develop
  master
```

Ramas Feature

- Cada nueva “feature” debe residir en su propia rama, que pueda ser enviada al repositorio central para que se pueda hacer backup/colaboración
- En lugar de ramificarse desde la rama master, las ramas Feature utilizan develop como su rama padre.
- Cuando se completa una feature, se mergea de nuevo con la rama develop. Las características nunca deben interactuar directamente con la rama master



Ramas Feature

- Tener en cuenta que las ramas Feature combinadas con la rama develop, son a todos los efectos, el workflow de la rama Feature
- Las ramas Feature generalmente se crean en la última rama de Develop

Ramas Feature

- Tener en cuenta que las ramas Feature combinadas con la rama develop, son a todos los efectos, el workflow de la rama Feature
- Las ramas Feature generalmente se crean en la última rama de Develop
- Para iniciar una rama Feature:

```
$ git flow feature start nombre_rama_feature
```

  - Tras esto, se puede trabajar con Git como haríamos normalmente

- Para terminar una rama Feature:

```
$ git flow feature finish nombre_rama_feature
```

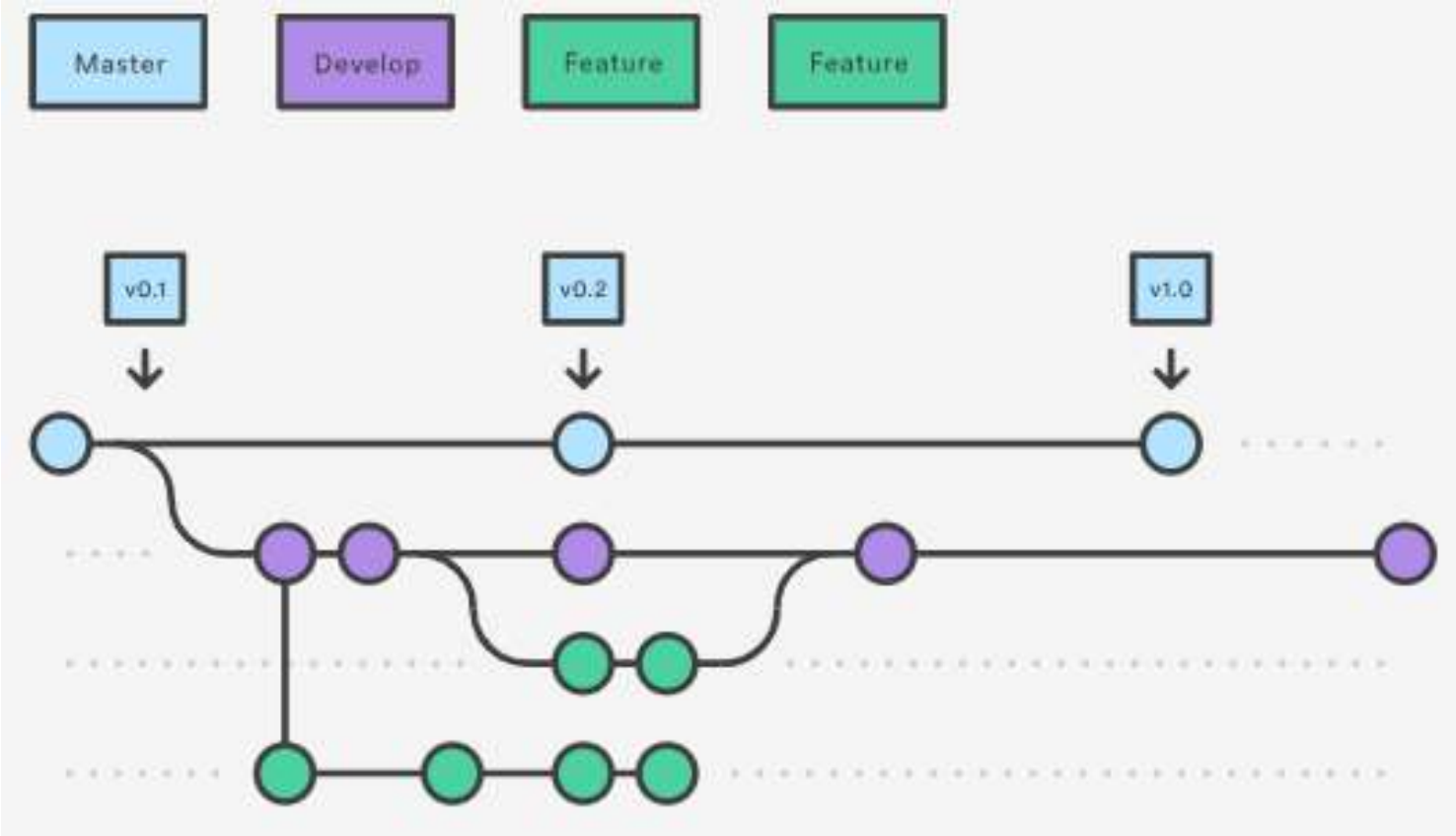
- Sin Git Flow
  - Crear rama para la feature nueva

```
$ git checkout develop
$ git checkout -b nombre_rama_feature
```

  - Terminar con la rama

```
$ git checkout develop
$ git merge nombre_rama_feature
```

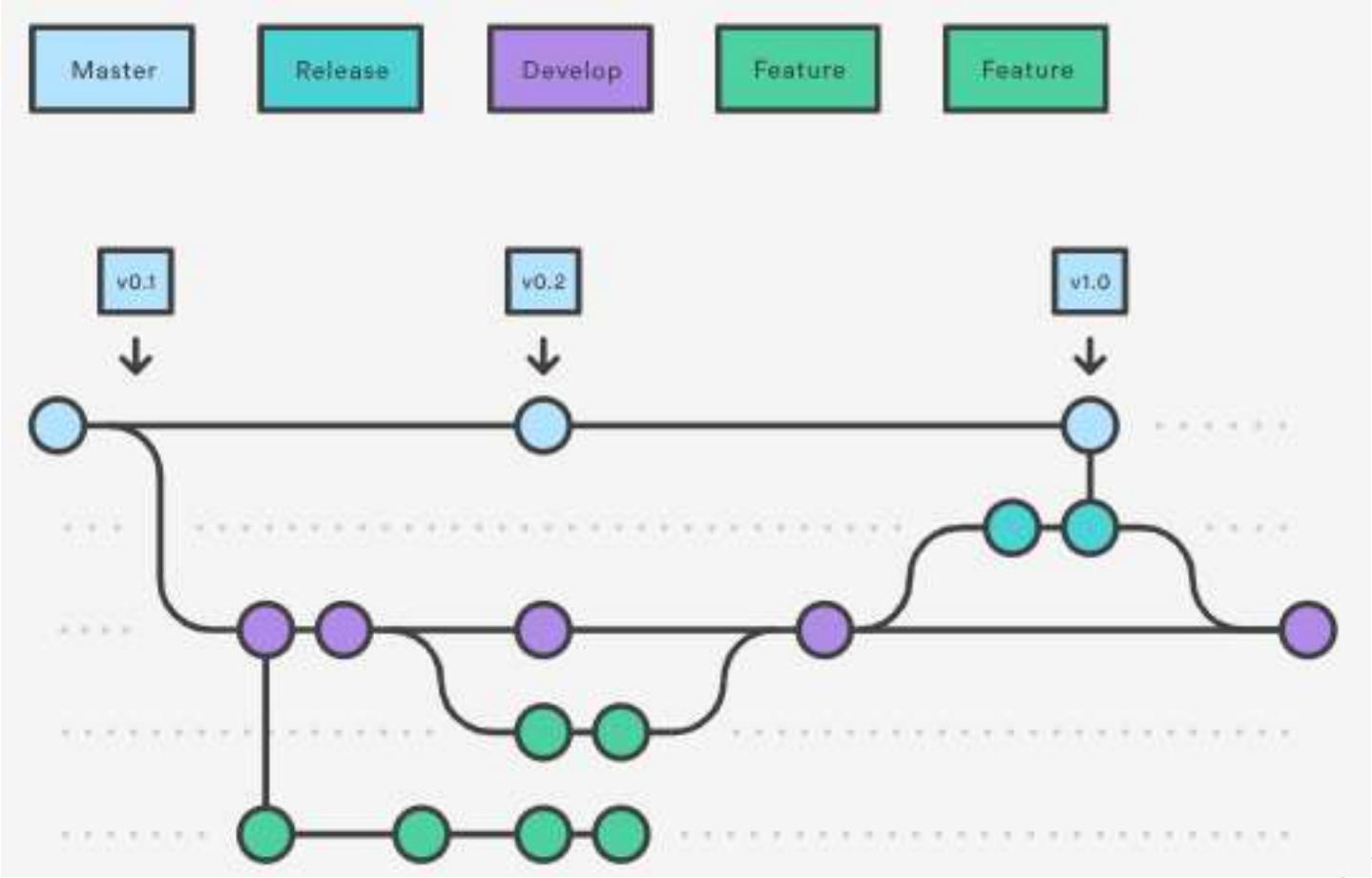
Ramas Feature



Ramas Release

- Una vez que la rama develop ha completado suficientes “features” para una reléase(se acerca la fecha de release), se bifurca una rama Release desde la rama develop
- La creación de esta rama inicia el siguiente ciclo de Release, por lo que no se pueden agregar nuevas features después de este punto
- Sólo se admite introducir en esta rama correcciones de errores, generación de documentación y otras tareas orientadas a la versión
- Una vez que está lista la rama Release, se fusiona con la master y se etiqueta con un número de versión.
- Además, debería fusionarse nuevamente con develop, que puede haber progresado desde que se inició la release

Ramas Release



Ramas Release

- El uso de una rama dedicada para preparar releases permite que un equipo pueda pulir la release actual mientras que otro equipo continúa trabajando en las features para la próxima versión
- También crea fases de desarrollo bien definidas (por ejemplo, es fácil decir: "Esta semana nos estamos preparando para la release 4.0" y verla realmente en la estructura del repositorio).
- Hacer ramas Release es otra operación directa de ramificación. Al igual que las ramas Features, las ramas Release se basan en la rama Develop
- Crear rama Release con Git Flow:

```
$ git flow release start 0.1.0
Switched to a new branch 'release/0.1.0’
```

- Crear rama Release sin Git Flow:

```
$ git checkout develop
$ git checkout -b release/0.1.0
```

Ramas Release

- Una vez que la reléase esté listo para lanzarse, se hará un merge en las ramas master y Develop, luego se eliminará la rama de Release
- Es importante mergerar nuevamente en rama Develop porque las actualizaciones críticas se pueden haber agregado a la rama Release y deben ser accesibles a las nuevas features
- Si su organización hace hincapié en la revisión del código, este sería un lugar ideal para un pull request
- Para finalizar una rama Release con Git Flow:

```
$ git checkout master
$ git checkout merge release/0.1.0
$ git flow release finish '0.1.0'
```

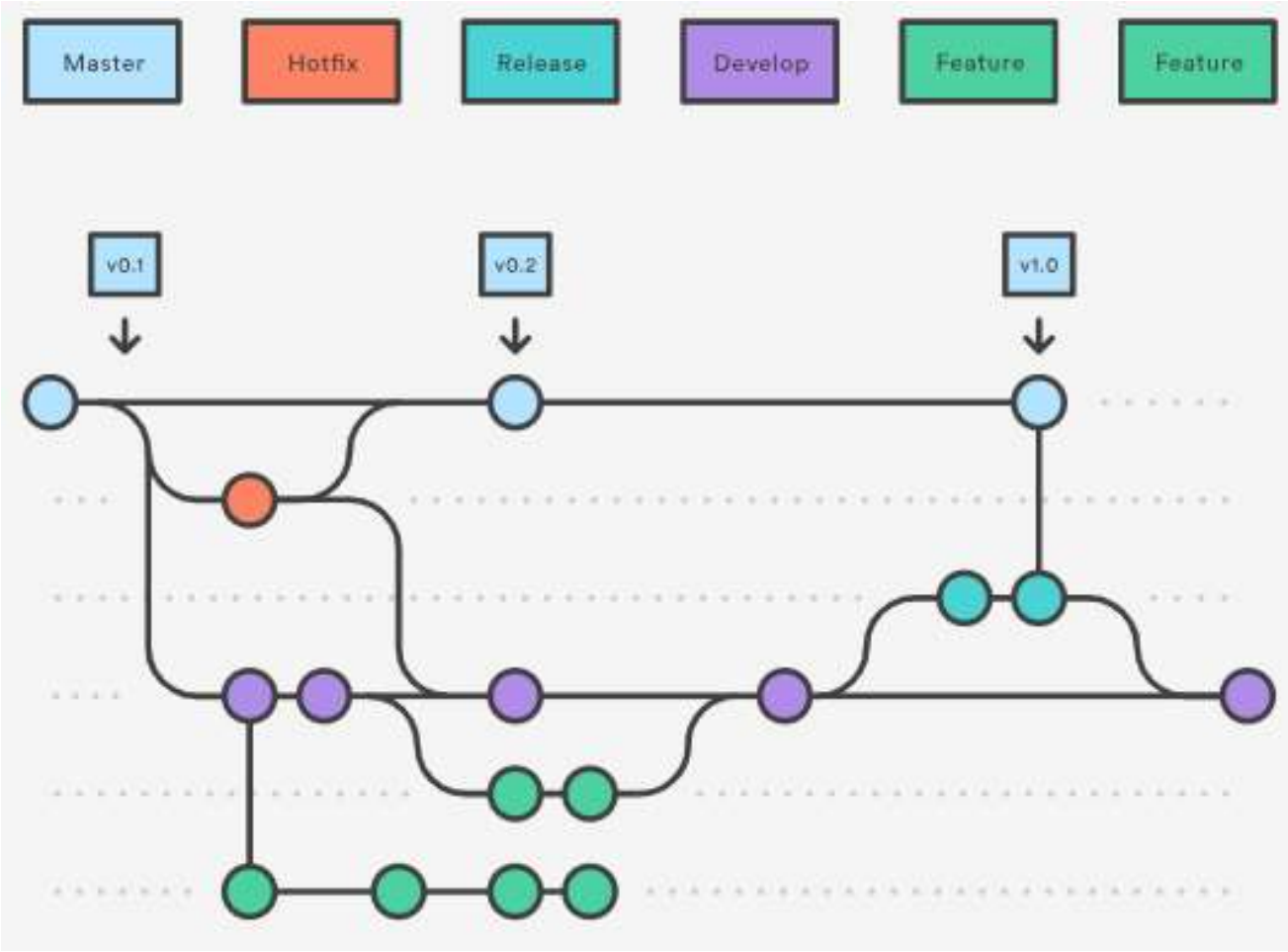
- Para finalizar una rama Release sin Git Flow:

```
$ git checkout develop
$ git merge release/0.1.0
```

Ramas Hotfix

- Las ramas de mantenimiento o "hotfix" se utilizan para parchear rápidamente releases de producción.
- Las ramas Hotfix se parecen mucho a las ramas Release y ramas Feature, excepto que se basan en la rama Master en lugar de Develop.
- Esta es la única rama que debe partir directamente de Master.
- Tan pronto como se arregle el problema, se debe mergear en rama Master Desarrollo (o la rama de reléase actual), y la rama master debe etiquetarse con un número de versión actualizado
- Tener una línea dedicada de desarrollo para corregir bugs permite al equipo solucionar problemas sin interrumpir el resto del workflow o esperar el siguiente ciclo de reléase
- Se puede pensar en ramas de mantenimiento como ramas Release ad hoc que trabajan directamente con rama Master.

Ramas Hotfix



Ramas Hotfix

- Trabajar con Hotfix con Git Flow:

```
$ git flow hotfix start hotfix_branch
...
$ git checkout master
$ git merge hotfix_branch
$ git checkout develop
$ git merge hotfix_branch
$ git branch -D hotfix_branch
$ $ git flow hotfix finish hotfix_branch
```

- Trabajar con Hotfix sin Git Flow:

```
$ git checkout master
$ git checkout -b hotfix_branch
```

Workflow con GitFLow

1. Se crea una rama Develop desde Master
2. Se crea una rama Releasedes de Develop
3. Las ramas Features se crean a partir de Develop
4. Cuando se completa una Feature, se mergea en la rama de Develop
5. Cuando se completa la rama de Release, se mergea en Develop
6. Si se detecta un problema en Master, se crea una rama Hotfix(revisión) desde la rama Master
7. Una vez que se completa el Hotfix, se mergea con Master y Develop

Ejemplo con Git Flow

- Ejemplo que demuestra workflow con rama Feature. Suponiendo que tenemos un repositorio configurado con la rama Master:
- Trabajado sobre una nueva feature:

```
git checkout master
git checkout -b develop
git checkout -b feature_branch
# Aquí va el trabajo sobre la rama feature
git checkout develop
git merge feature_branch
git checkout master
git merge develop
git branch -d feature_branch
```

Ejemplo con Git Flow

- Ejemplo que demuestra workflow con rama Feature. Suponiendo que tenemos un repositorio configurado con la rama Master:
- Trabajado con Hotfix:

```
git checkout master
git checkout -b hotfix_branch
# work is done commits are added to the hotfix_branch
git checkout develop
git merge hotfix_branch
git checkout master
git merge hotfix_branch
```

Ejercicio con Git Flow

- Probar los siguientes ejemplos referentes a Git Flow:
- https://blog.axosoft.com/gitflow/
- http://aprendegit.com/git-flow-release-branches/
- https://blog.axosoft.com/gitflow/