Tutoriales

Enter Your Email For Git News

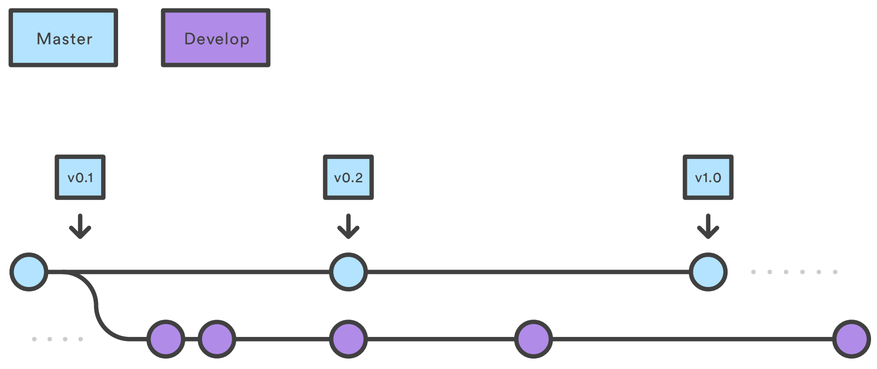
# Flujo de trabajo de Gitflow

Gitflow Workflow is a Git workflow design that was first published and made popular by Vincent Driessen at nvie. The Gitflow Workflow defines a strict branching model designed around the project release. This provides a robust framework for managing larger projects.

¿Listo para aprender Git? Prueba este tutorial interactivo.

Empieza ahora

## How it works



Aprende Git

Principiante

Empezando

## Desarrollar y dominar las ramas

Tutoriales

Enter Your Email For Git News

proyecto. La master sucursal almacena el historial de versiones oficial, y la develop sucursal sirve como una rama de integración para las características. También es conveniente etiquetar todas las confirmaciones en la master sucursal con un número de versión.

El primer paso es complementar el valor predeterminado master con una develop rama. Una forma simple de hacer esto es que un desarrollador cree una develop rama vacía localmente y la envíe al servidor:

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

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

**¿Listo para aprender Git?**

**Prueba este tutorial interactivo.**

Empieza ahora

Al usar la biblioteca de extensión git-flow, la ejecución git flow init de un repositorio existente creará la develop rama:

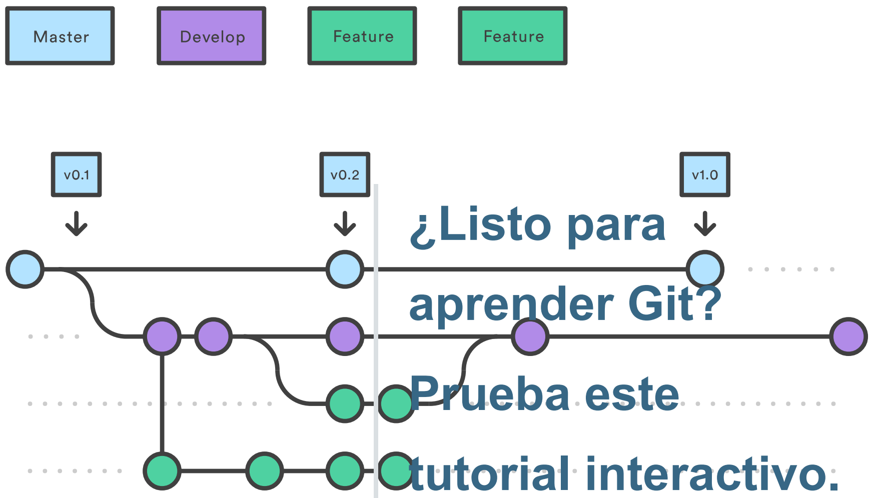
```
$ git flow init
Initialized empty Git repository in ~/project/.g
No branches exist yet. Base branches must be cre
Branch name for production releases: [master]
Branch name for "next release" development: [dev

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
```

Tutoriales

Enter Your Email For Git News

# Ramas de funciones

Cada nueva característica debe residir en su propia sucursal, que puede enviarse al depósito central para copia de seguridad / colaboración. Pero, en lugar de ramificarse `master`, las `feature`ramas usan `develop` como su rama principal. Cuando se completa una característica, se fusiona de nuevo en desarrollo . Las características nunca deben interactuar directamente con `master`.



¿Listo para aprender Git?

Prueba este tutorial interactivo.

Empieza ahora

Tenga en cuenta que las `feature`ramas combinadas con la `develop`sucursal s tos, el flujo de trabajo de la sucursal de funciones. Pero, el flujo de trabajo de Gitflow no se detiene allí.

`Feature`las ramas generalmente se crean a la última `develop`rama.

## Crear una rama de características

Without the git-flow extensions:

When using the git-flow extension:

```
git flow feature start feature_branch
```

Continue your work and use Git like you normally would.

### Finishing a feature branch

When you're done with the development work on the feature, the next step is to merge the `feature_branch` into `develop`.

Without the git-flow extensions:

```
git checkout develop
git merge feature_bran
```

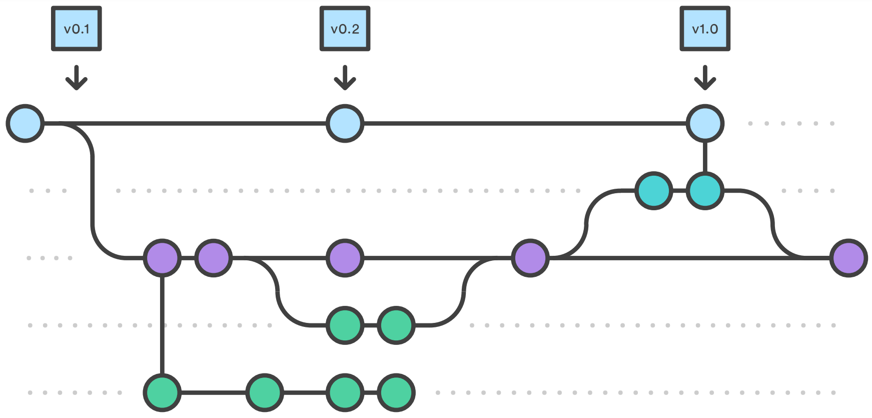
Using the git-flow extensions:

```
git flow feature finish
```

## Release Branches

Tutoriales

Enter Your Email For Git News



Once `develop` has acquired enough features for a release (or a predetermined release date is approaching), you fork a `release` branch off of `develop`. Creating this branch starts the next release cycle, so no new features can be added after this point—only bug fixes, documentation generation, and other release-oriented tasks should go in this branch. Once it's ready to ship, the `release` branch gets merged into `master` and tagged with a version number. In addition, it should be merged back into `develop`, which may have progressed since the release was initiated.

¿Listo para aprender Git? Prueba este tutorial interactivo.

Empieza ahora

Using a dedicated branch to prepare releases makes it possible for one team to polish the current release while another team contin[illegible]tures for the next release. It also cr[illegible] phases of development (e.g., it's easy to say, “This week we're preparing for version 4.0,” and to actually see it in the structure of the repository).

Making `release` branches is another straightforward branching operation. Like `feature` branches, `release` branches are based on the `develop` branch. A new `release` branch can be created using the following methods.

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

When using the git-flow extensions:

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

Once the release is ready to ship, it will get merged it into `master` and `develop`, then the `release` branch will be deleted. It's important to merge back into `develop` because critical updates may have been added to the `release` branch and they need to be accessible to new features. If your organization stresses code review, this would be an ideal place for a pull request.

To finish a `release` branch, use the following methods:

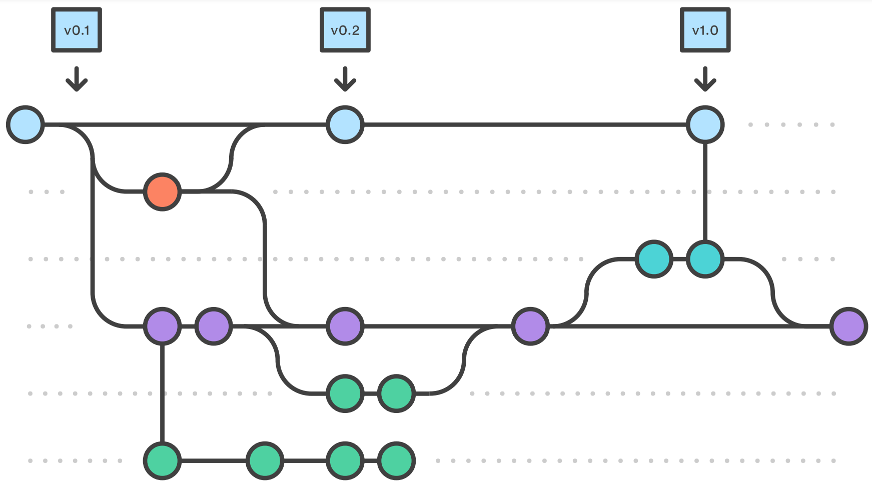
Without the git-flow extensions:

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

Or with the git-flow extens

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

## Hotfix Branches



Maintenance or “hotfix” branches are used to quickly patch production releases. Hotfix branches are a lot like release branches and feature branches except they're based on master instead of develop. This is the only branch that should fork directly off of master. As soon as the fix is complete, it should be merged into both master and develop (or the current release branch), and master should be tagged with an updated version number.

Having a dedicated line of development for bug fixes lets your team address issues without interrupting the rest of the workflow or waiting for the next release cycle. You can think of ma s as ad hoc release branches that work directly with master. A hotfix branch can be created using the following methods:

Without the git-flow extensions:

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

When using the git-flow extensions:

Similar to finishing a `release` branch, a `hotfix` branch gets merged into both `master` and `develop`.

```
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
```

## Example

A complete example demonstrating a Feature Branch Flow is as follows. Assuming we have a repo setup with a `master` branch.

```
git checkout -b develop
git checkout -b releas
git checkout develop
git checkout -b feature_branch
# a bunch of work is done on the features
git checkout release/0.1.0
git merge feature_brar
# assuming that the re                    lone with
git checkout develop
git merge release/0.1.0
git checkout master
git merge release/0.1.0
git branch -D release/0.1.0
```

In addition to the `feature` and `release` flow, a `hotfix` example is as follows:

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

Tutoriales

Enter Your Email For Git News

## Summary

Here we discussed the Gitflow Workflow. Gitflow is one of many styles of Git workflows you and your team can utilize.

Some key takeaways to know about Gitflow are:

- The workflow is great for a release-based software workflow.
- Gitflow offers a dedicated channel for hotfixes to production.

¿Listo para aprender Git? Prueba este tutorial interactivo.

Empieza ahora

The overall flow of Gitflow is:

1. A `develop` branch is created from `master`
2. A `release` branch is created from `develop`
3. `Feature` branches are p
4. Cuando a `feature` se completa se fusiona en la `develop` rama
5. Cuando el `release` se realiza rama en la que se combina en `develop` y `master`
6. Si `master` se detecta un problema en, `hotfix` se crea una rama desde `master`
7. Una vez que `hotfix` se completa, se fusiona con ambos `develop` y `master`

Tutoriales

Enter Your Email For Git News

trabajo".

## ¿Listo para aprender Git?

## Prueba este tutorial interactivo.

Empieza ahora

Siguiente:

Flujo de trabajo bifurcante

COMI
SIGUIENT

¿Listo para aprender Git?

Prueba este tutorial interactivo.

Empieza ahora

Energizado por

Tutoriales

Enter Your Email For Git News

futuros?

Enter Your Email For Git News

## ¿Listo para aprender Git?

## Prueba este tutorial interactivo.

Empieza ahora