TÉCNICAS DE INTEGRACIÓN DE CÓDIGO

Roles / Ramas Protegidas / Force Push / Flujos de Trabajo

Roles de Git

En función del manejador de repositorios que utilizas (GitHub, GitLab u otro), se tienen distintos roles con distintos permisos.

Roles de Gitlab:

https://docs.gitlab.com/ee/user/permissions.html

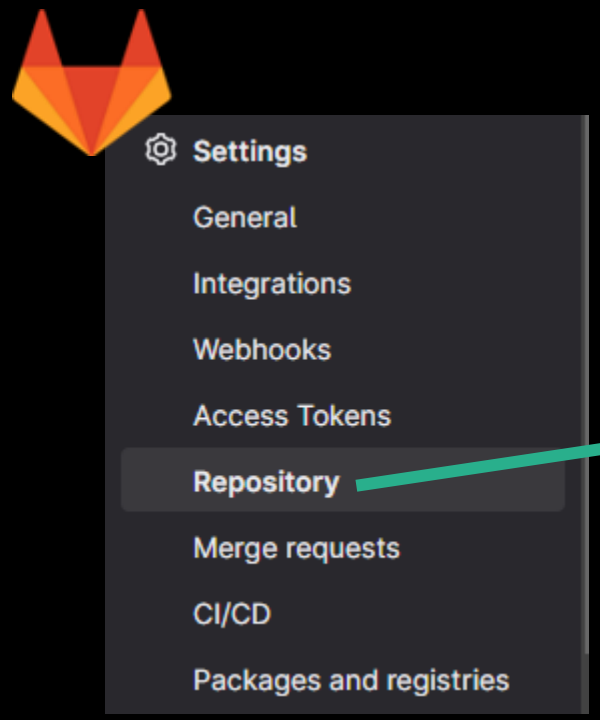
Roles de GitHub:

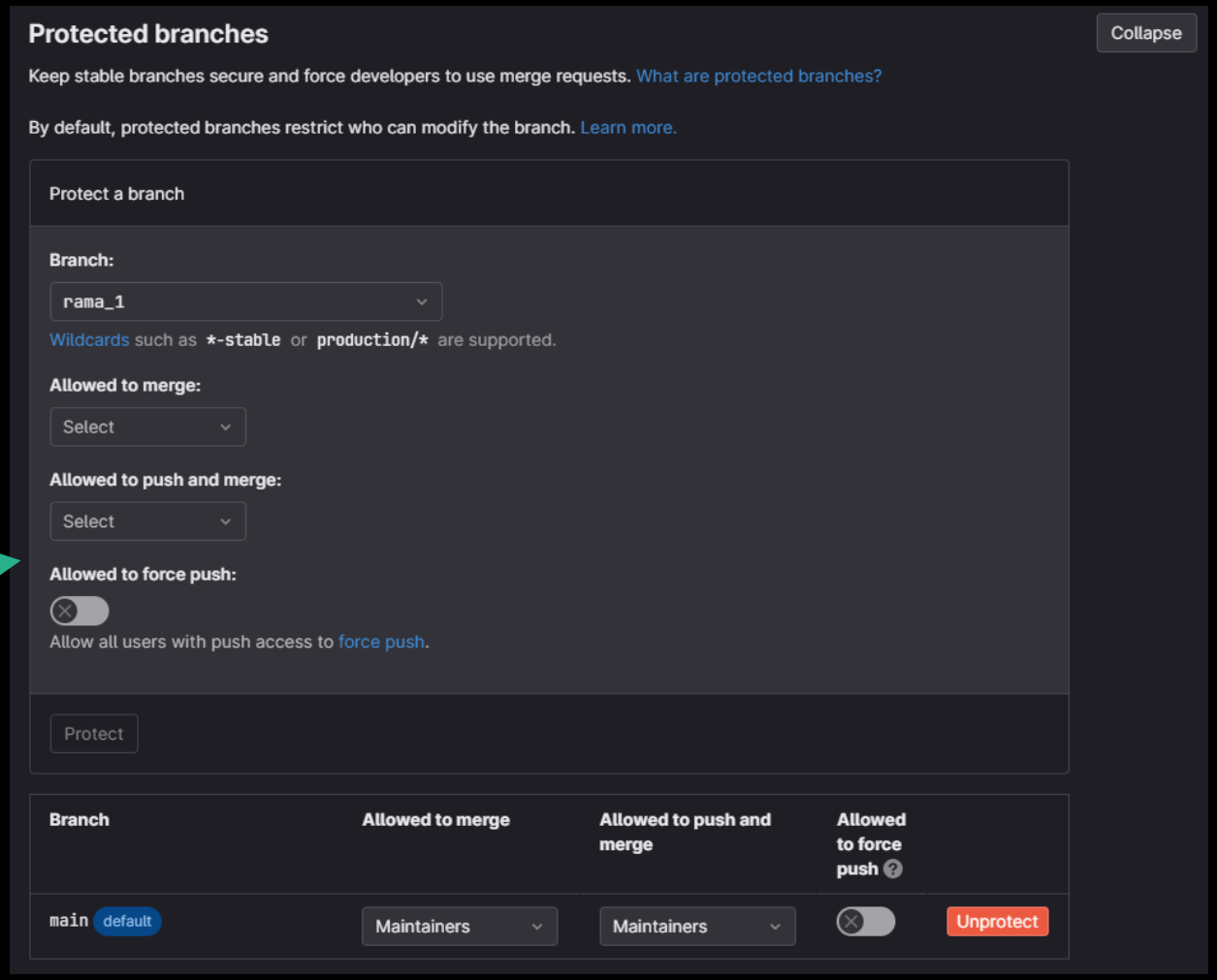
https://docs.github.com/en/organizations/managing-user-access-to-your-organizations-repositories/repository-roles-for-an-organization

Ramas protegidas

Una rama protegida es una rama en la que sólo ciertos roles o usuarios pueden trabajar.

Así puedes proteger tus ramas:



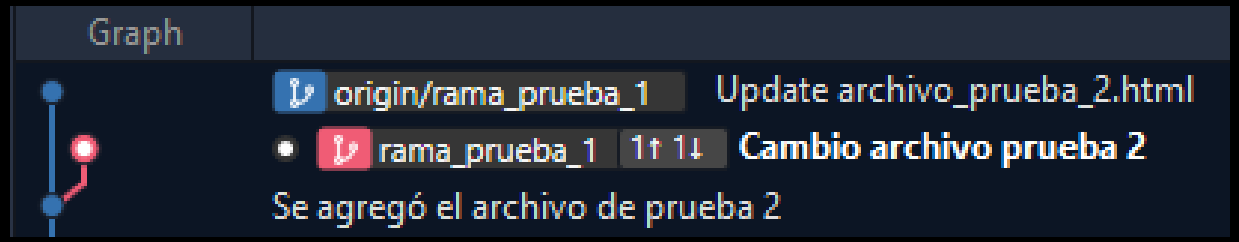


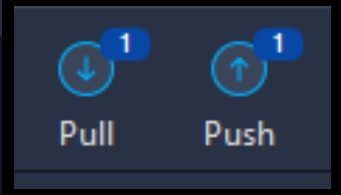
Consideraciones de ramas protegidas:

- Por defecto, la rama principal de tu repositorio (main) está protegida.
  - Cuando creas un repositorio, tu rol es el de "Owner".
  - En las ramas protegidas sólo ciertos usuarios pueden hacer "Force Push".

Force Push

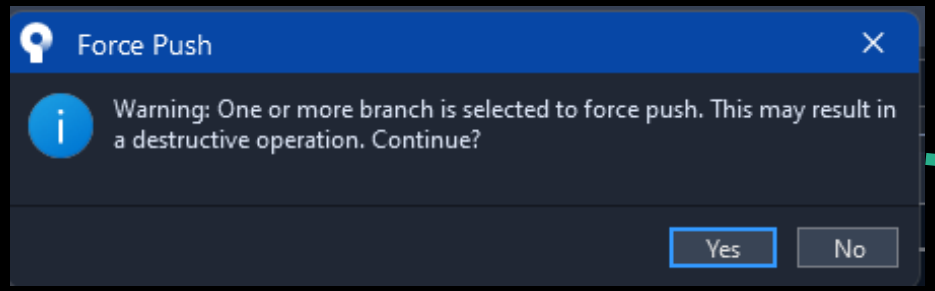
- Un "Force Push" es un push que se hace independientemente de si se tienen commits pendientes por descargar.
- Un Force Push es una **acción destructiva**, o sea, eliminará código del repositorio remoto.
- Un Force Push puede ser útil si se subió código por error y se quiere sobreescribir con el código correcto.
- Una vez eliminado el código del remoto con el Force Push, este **NO SE PUEDE RECUPERAR**.



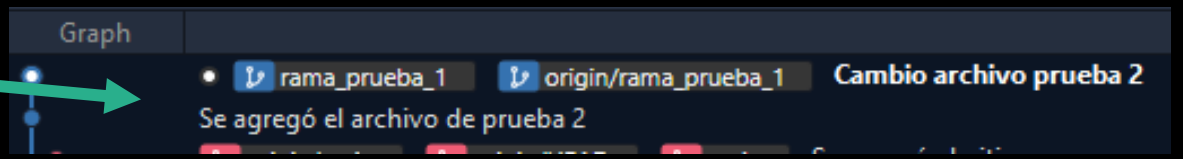


Se tiene un commit en el local "Cambio archivo prueba 2", y uno en el remoto "Update archivo_prueba_2.html". Pulsamos "Push".

Habilitamos la opción de "Force Push" (si no lo marcamos, nos rechazará el Push).



Confirmamos, y el commit del remoto ya no estará:



Ejercicio express:

Con los equipos que ya tienes, toma cualquier proyecto que tengas donde haya ramas adicionales a "main", EXCEPTO EL DE LA PRÁCTICA.

Métanse a proteger esa rama desde su manejador de repositorio.

Algún usuario debe hacer un Force Push. Planteen una situación con sus commits en las que sea necesario que hagan el force push.

Antes de hacer el Force Push, tomen captura de pantalla de cómo se ve. Subirán esto como parte de una tarea al final.

Recordando... Flujos de Trabajo en Git

Un flujo de trabajo es una estructura recomendada para usar Git.

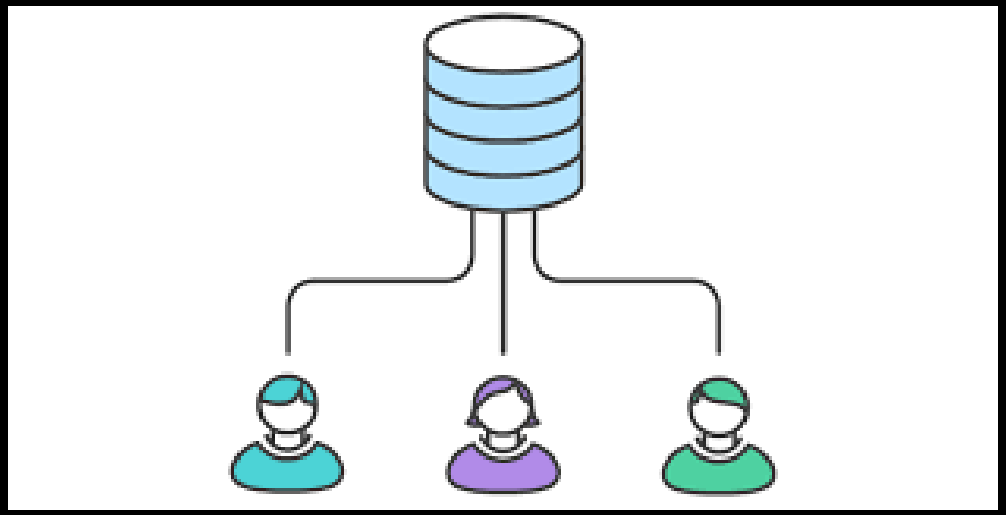
Un flujo de trabajo que se adapte a tu equipo debe ser:

- De escala correcta (¿funciona para un equipo de 2, 3, 10, 100 personas?)
- Permite modificaciones al código de forma fácil y rápida.
- Debe ser fácil de adaptarse a él.

Flujo de trabajo centralizado

Los desarrolladores clonan el "repositorio central" en su computadora de forma local. Cada quien tiene una copia del proyecto.

Todos trabajan sobre la rama "main".



¿Ventajas? ¿Desventajas?

Flujos de trabajo:

Flujo de trabajo centralizado ✅

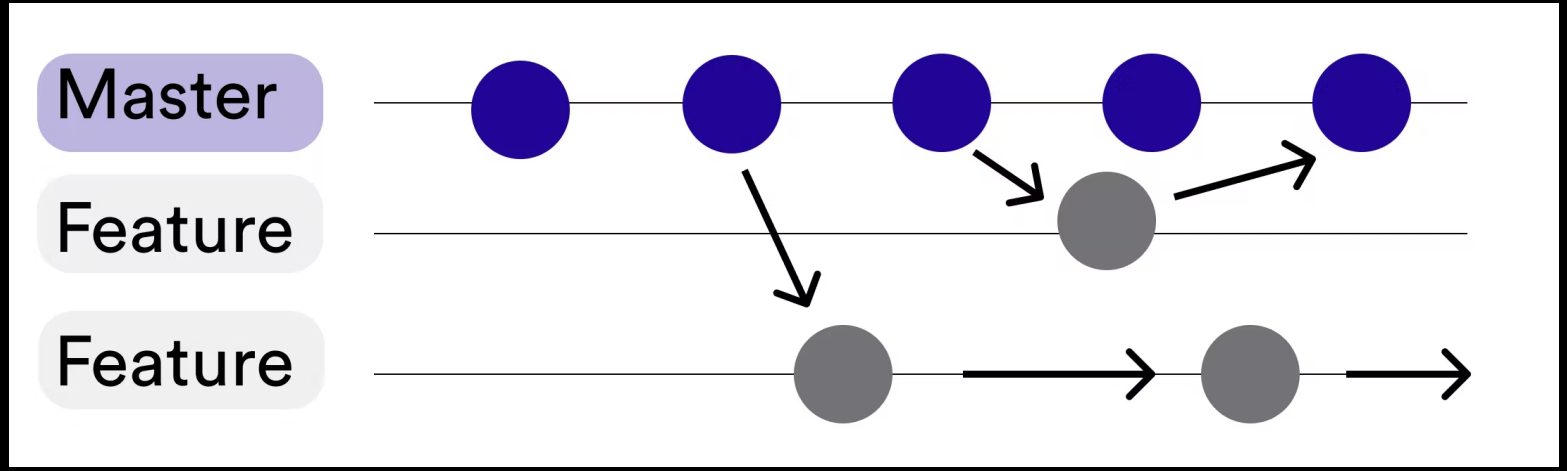
Feature branching 🕒

Gitflow Workflow 🕒

Forking Workflow 🕒

Feature Branching

Se basa en que cada función nueva en el software se haga en una nueva rama, "encapsulando" esa función en ella.



Pros / Contras del Feature Branching

Pros	Contras
Permite trabajar en nuevas funciones sin afectar el código original.	Tiene "pasos adicionales" necesarios al momento de crear ramas, lo que lo puede hacer ineficiente.
Se pueden separar más claramente las funciones nuevas del código.	Puedes llegar a tener muchas ramas en tu repositorio, lo que puede hacer que se vuelva confuso trabajar con él.
Se puede evaluar si una función se pasará a producción al verla de forma aislada en una rama.	Hacer ramas puede hacer que sea difícil hacer reverse a algún punto en específico una vez que se hace merge de alguna rama.
Te da la posibilidad de hacer merge request (lo veremos en un momento).	El proceso de validación puede ser más tardío por los merge request, lo que puede hacer que sea más lento lanzar cambios a producción.

Consideraciones del Feature Branching

- Es recomendable que las ramas no sean tan grandes. Es decir, que su tiempo de vida sea el menor posible. ¿Por qué?
- Si tus feature branches viven mucho tiempo por cualquier motivo, es recomendable hacer merge de la rama principal a tu feature branch cada cierto tiempo.
- Las ramas hechas en tu feature-branching deben tener nombres descriptivos del feature en cuestión.

Merge Request

En algunas partes, lo encontrarás también como "Pull Request".

Un "Merge Request" es, como su nombre lo indica, una solicitud para hacer merge.

Puede que un usuario sin permisos trate de hacer un merge a una rama protegida, por lo que debe hacer un merge request para que alguien que sí tiene permisos lo autorice.

Al hacer el request, se puede especificar a quién se le asignará el request (recibirá un correo informándolo).

Crear Merge Request

New merge request

From `rama_prueba_1` into `main` Change branches

Title (required)

Solicitud Merge a Producción

Mark as draft

Drafts cannot be merged until marked ready.

Add description templates to help your contributors to communicate effectively!

Description

Write Preview

Quisiera hacer merge de mi función para poder hacer que el código funcione chido

Supports Markdown. For quick actions, type `/`.

Assignee

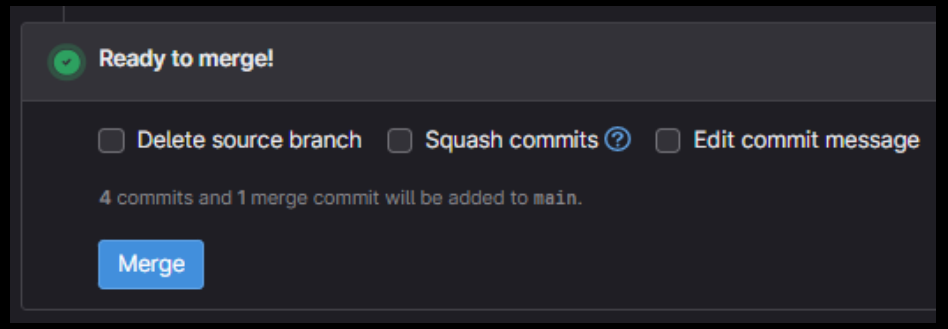
Ulises Tejeda Chávez

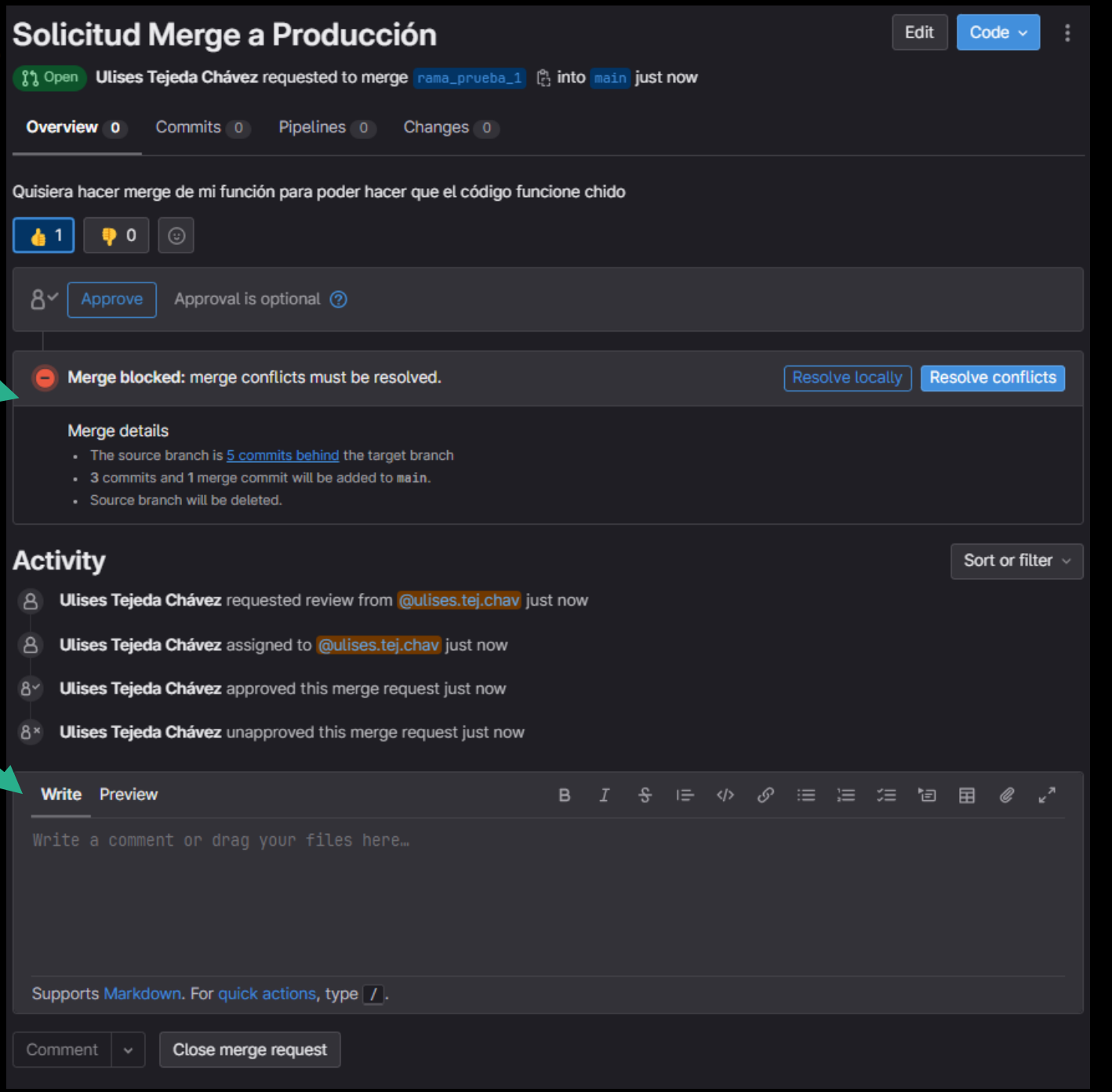
Reviewer

Ulises Tejeda Chávez

Resolver Merge Request

- Podemos resolver conflictos ahí mismo (si los hay).
- Podemos comentar en el merge request.
- Podemos hacer el Merge una vez que no tengamos conflictos.





Otro ejercicio

- Primero, arma un contexto en tu repositorio en el que debas de hacer merge de una rama a otra (en cualquier rama, haz cualquier cambio, haz commit y déjalo listo para hacer un merge request con, por ejemplo, main).
- En equipos, de nuevo, crea un merge request de la rama que sea. Quien lo crea, debe asignárselo a alguien que no sea él.
- La otra persona del equipo, a la que se le asignó el merge request, debe resolverlo con la ayuda de sus compañeros.
- Toma captura de cómo creas el merge request y de cómo lo resuelves.

Flujos de trabajo:

Flujo de trabajo centralizado ✅

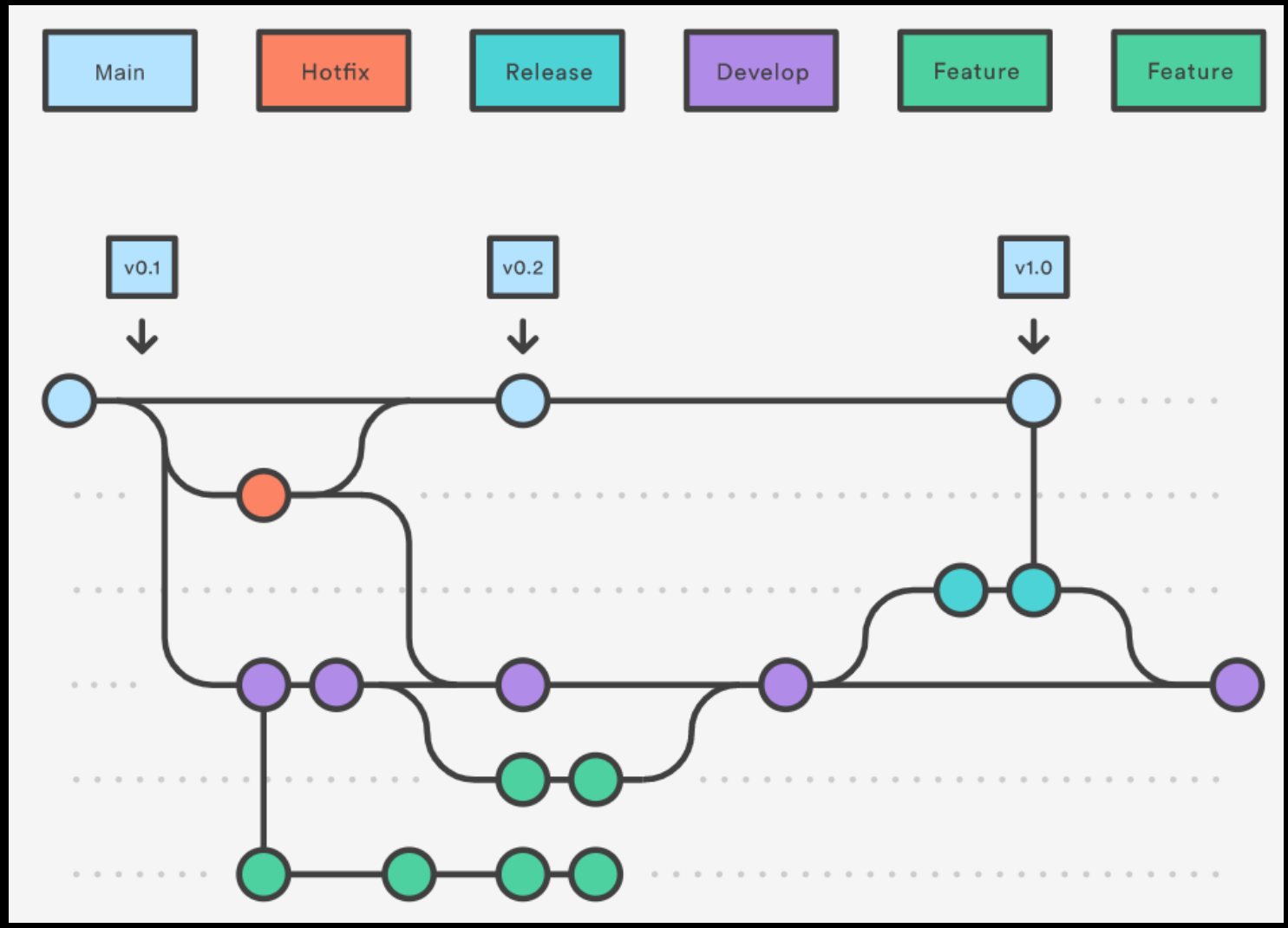
Feature branching ✅

Gitflow Workflow 🕛

Forking Workflow 🕛

Gitflow Workflow

En este flujo, se tiene una rama "main", una rama "develop", ramas "release", ramas de función y ramas de corrección.



Explicación de las ramas:

- **Main**: La rama que ya conoces, donde se tiene el código principal.
- **Develop**: Es la rama que se utiliza para integrar en ella los cambios de las feature branches.
- **Feature**: Igual que con el Feature Workflow, aquí se hacen los nuevos desarrollos. Sin embargo, al terminarse, se mandan a Develop y/o a Main (en Main, depende del contexto o de cómo se quiera trabajar).
- **Release**: Cuando Develop llega a cierto punto, se bifurca en una rama de Release. Esta rama ya no se modifica a menos que haya cambios de arreglos de errores. Tener estas ramas permite que algunos miembros del equipo trabajen en una versión publicada específica mientras otros pueden seguir trabajando en nuevas funciones.
- Hotfix: Ramas que contienen arreglos.

Flujos de trabajo:

Flujo de trabajo centralizado ✅

Feature branching ✅

Gitflow Workflow ✅

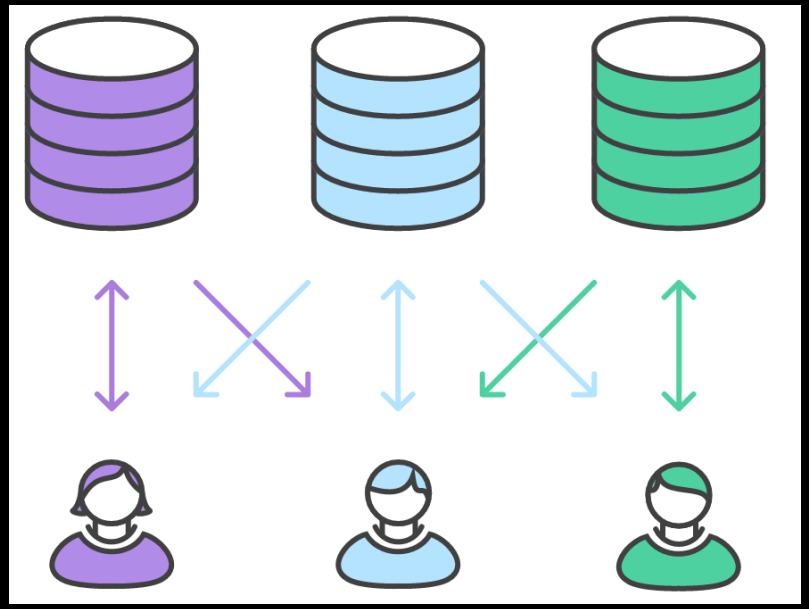
Forking Workflow 🕛

Forking Workflow

Para este repositorio, se tienen, al menos, dos repositorios remotos, no sólo uno.

Usado usualmente en proyectos Open Source.

Permite que cada desarrollador tenga su repositorio remoto, y los cambios que hacen ahí los integra el administrador del repositorio original como lo prefiera.



Flujos de trabajo:

Flujo de trabajo centralizado ✅

Feature branching ✅

Gitflow Workflow ✅

Forking Workflow ✅

Consideraciones:

- Las ramas deben de "vivir" lo menos posible. Cuando más tiempo esté separada de la rama de producción, más grande el riesgo de problemas
- Es recomendable tener un flujo de trabajo que te permita tener facilidad de revertir los cambios.
- Recuerda que tu flujo de trabajo debe hacer que tu equipo pueda producir cosas más rápido. No te obligues a usar uno que no te vaya a servir.
- Los flujos de trabajo son adaptables a como lo necesites, no son leyes a seguir.
- Hay más flujos de trabajo, estos solo son algunos de ellos.

Y última cosa...

Sube las capturas de lo hecho en clase como tarea
Siguiente clase hay examen