

FLUJO DE ADMINISTRACION DE GITHUB

Gerencia de soluciones



Instructivo personal administrativo



CONTENIDO



Hola soy Anna, aquí aprenderás los pasos básicos que debe realizar un administrador:

1. Modulo #1: Información General
2. Modulo #2: GitHub Web
3. Modulo #3: Repositorios y ramas
4. Modulo #4: Configuración y seguridad

CONTENIDO(Modulos)

Modulo #1: Información General.

1. Conceptos generales.
2. Roles.
3. Prefijos.
4. Solicitud de rama.
5. Solicitud de repositorio.
6. Arquitectura de ramas.
7. Estrategia de versionamiento

Modulo #3: Configuración y seguridad.

1. Seguridad de ramas
2. Flujo de validaciones
3. Casos de flujo administrativo
4. PullRequest
5. Review

Modulo #2: GitHub Web.

1. GitHub Web.
2. Overview.
3. Repositories

Modulo #3: Repositorios y ramas.

1. Creación de repositorios.
2. Datos generales de desarrollo
3. Finalización de repositorios.
4. Nomenclatura de ramas.
5. Creación de ramas

Modulo #1: Información General.





CONCEPTOS GENERALES

Antes de iniciar, debemos conocer un poco sobre los comandos, términos o acciones más comunes en las tareas que se desempeñen, tales como:



- **RAMA MAIN:** es la **rama principal** de nuestro repositorio (ambiente producción).
- **RAMA QA:** es la **rama principal** utilizada para la certificación de los cambios (ambiente QA).
- **RAMA DEVELOP:** es la **rama principal** utilizada para el desarrollo de nuevas funcionalidad y/o características del producto (ambiente desarrollo).
- **RAMAS _QA y _DEVELOP:** son las **ramas secundarias, creadas a partir de sus ramas principales** (develop y qa), utilizadas para procesar los cambios desarrollados (_develop) y que, posteriormente, serán certificados (_qa).



- **REPOSITORIO GITHUB:** comprende un espacio de almacenamiento virtual para la gestión de las versiones de código de un proyecto.
- **CLONAR UN REPOSITORIO:** comprende la creación de una copia en nuestro ambiente local (equipo personal) del repositorio remoto.
- **GIT COMMIT:** comando más utilizado para guardar cualquier cambio efectuado en las ramas de esta herramienta.
- **PULL REQUEST:** petición que necesita aprobación para integrar los cambios de código de una rama a otra.

ROLES



Dentro de este flujo, si bien existen acciones y/o comandos, también existen quienes ejecutan estas acciones, las cuales son personas con los siguientes roles:

- **DESARROLLADOR:** será quien podrá trabajar sobre el código del proyecto.
- **REVIEWER:** será la persona encargada de las revisiones y aprobaciones de código nuevo.
- **ASIGNEES:** será el administrador quien realizará el paso de un ambiente a otro mediante el **PULL REQUEST**.

PREFIJOS



Para la identificación del cambio que se va realizar, utilizaremos los siguientes prefijos:

- **feature**: utilizado cuando se esté desarrollando una **nueva** característica o funcionalidad.
- **refactor**: utilizado cuando se esté realizando una **mejora** de una característica o funcionalidad existente.
- **fix**: utilizado para identificar los cambios que requieren corrección de errores. (**ambiente desarrollo**).
- **hotfix**: utilizado para identificar los cambios que requieren corrección de errores. (**ambiente producción**).
- **integration**: utilizado para identificar las ramas donde se estarán comparando cambios de 2 o más desarrolladores para la resolución de conflictos.

SOLICITUD DE RAMA



Para tener acceso a un repositorio y la creación de una **nueva rama**, es necesario realizar una solicitud al equipo administrador, considerando lo siguiente:

1. Completar el **Formato de Solicitud de Rama**. El formato se encuentra en el siguiente enlace:
2. Enviar el **Formato de Solicitud de Rama (vía correo)** a los administradores de GitHub.
3. El equipo administrador realizará lo siguiente:
 - a) Revisión de la solicitud enviada.
 - b) Si la solicitud está correcta, se procede con la creación de la rama solicitada.
4. Se enviará la **Resolución de la Solicitud (vía correo)**.

SOLICITUD DE REPOSITORIO



Para la creación de un **nuevo repositorio**, es necesario realizar una solicitud al equipo administrador, considerando lo siguiente:

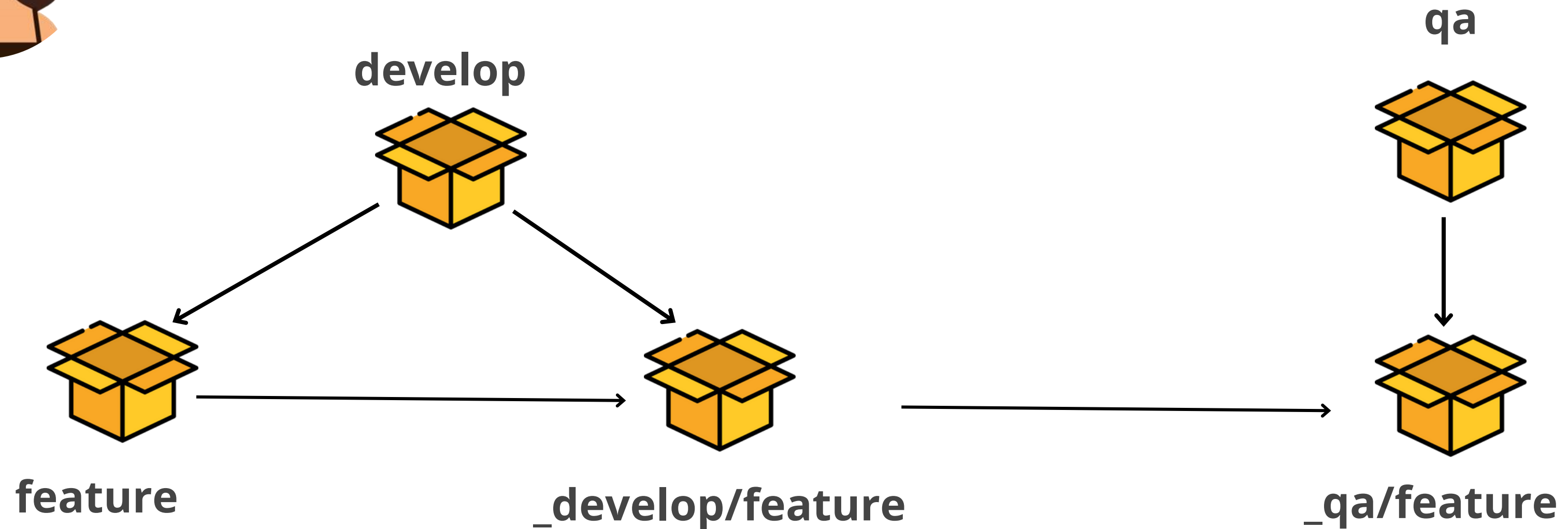
1. Completar el **Formato de Solicitud de Repositorio**. El formato se encuentra en el siguiente enlace:
2. Enviar el **Formato de Solicitud de Repositorio (vía correo)** a los administradores de GitHub.
3. El equipo administrador realizará lo siguiente:
 - a) Revisión de la solicitud enviada.
 - b) Si la solicitud está correcta, se procede con la creación del repositorio solicitado.
4. Se enviará la **Resolución de la Solicitud (vía correo)**.

IMPORTANTE: esta solicitud solo debe ser realizada por parte de **líder de equipo / líder de proyecto**.

ARQUITECTURA DE RAMAS



El flujo administrativo contará con reglas y requerimientos por cada uno de los ambientes necesarios.



ESTRATEGIA DE VERSIONAMIENTO

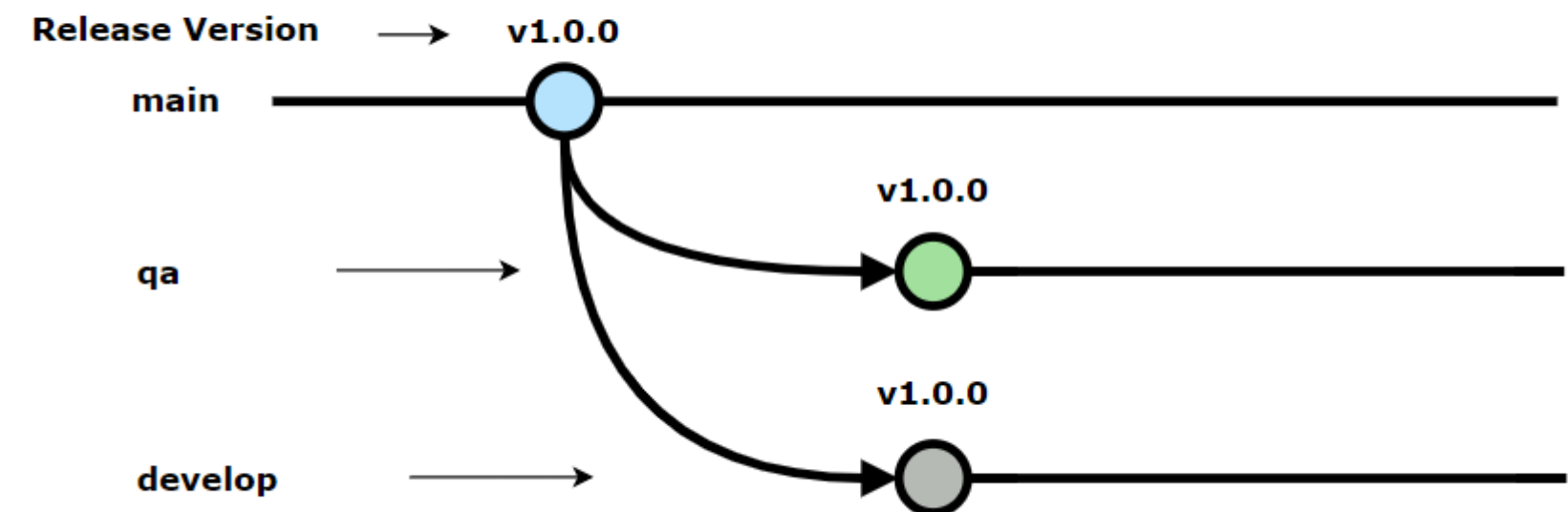


Como parte de la estrategia, nuestro repositorio inicialmente contará con **3 ramas principales:**

- **main**: es la rama que apunta a nuestro ambiente de producción.
- **qa**: es la rama que apunta a nuestro ambiente de certificación. Esta rama será creada a partir de la rama **main**.
- **develop**: es la rama que apunta a nuestro ambiente de desarrollo. Al igual que la rama de **qa**, será creada a partir de la rama **main**.

IMPORTANTE: en las **ramas principales**, NO está permitido integrar o realizar cambios directamente; esto sin excepciones.

Creación de ramas principales





Algunas de las restricciones de las ramas principales son:

1. Estas **nunca deben ser afectadas directamente** (commits o merge directos), ya que de esta forma se mantienen con la versión más estable (según ambiente producción).
2. Solo se afectan por medio de una **integración de cambios** (Pull Request).
3. La integración de cambios que se realicen a estas ramas serán gestionados por el personal encargado de la administración del repositorio.



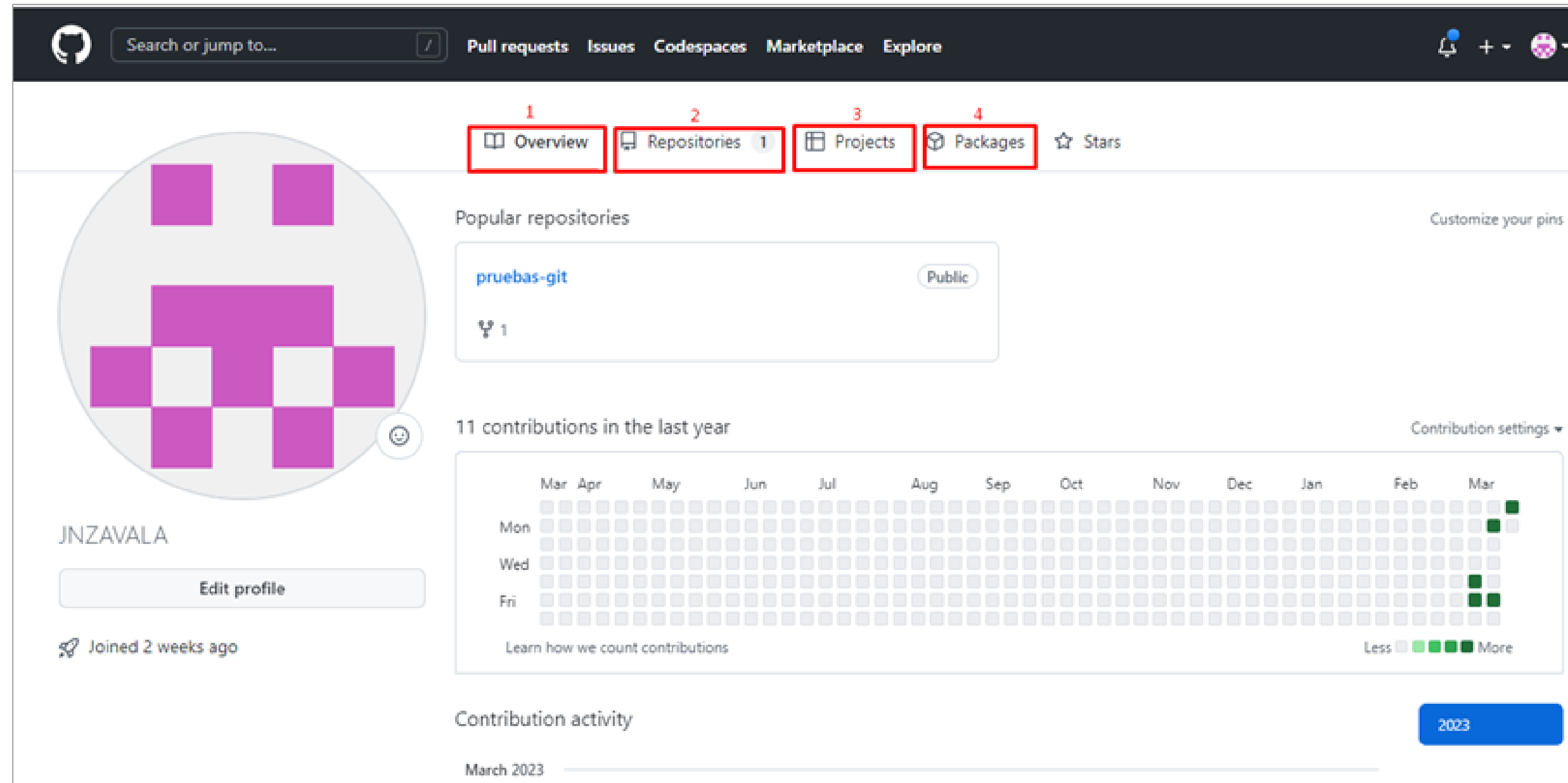
Para crear una funcionalidad, realizar una mejora o algún ajuste, se tomará en cuenta lo siguiente:

1. Mediante la **Solicitud de Gestión de Repositorio**, se debe indicar la **información de los usuarios** que requieren el acceso, así como el **detalle de los cambios que se van realizar**.
Dependiendo del tipo de cambio, se utilizará algún prefijo que sea acorde al cambio a realizar (**feature, refactor, fix, hotfix, integration**).
2. Una vez validada la gestión y el detalle del cambio por parte del personal administrativo, se procederá con la creación de la rama o las ramas donde se estarán desarrollando las funcionalidades.

Modulo #2: GitHub Web.



GitHub(Web)



Search or jump to... Pull requests Issues Codespaces Marketplace Explore

1 Overview 2 Repositories 1 3 Projects 4 Packages ☆ Stars

Popular repositories

pruebas-git Public

11 contributions in the last year

Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar

Mon Wed Fri

Learn how we count contributions

Less More

Contribution activity

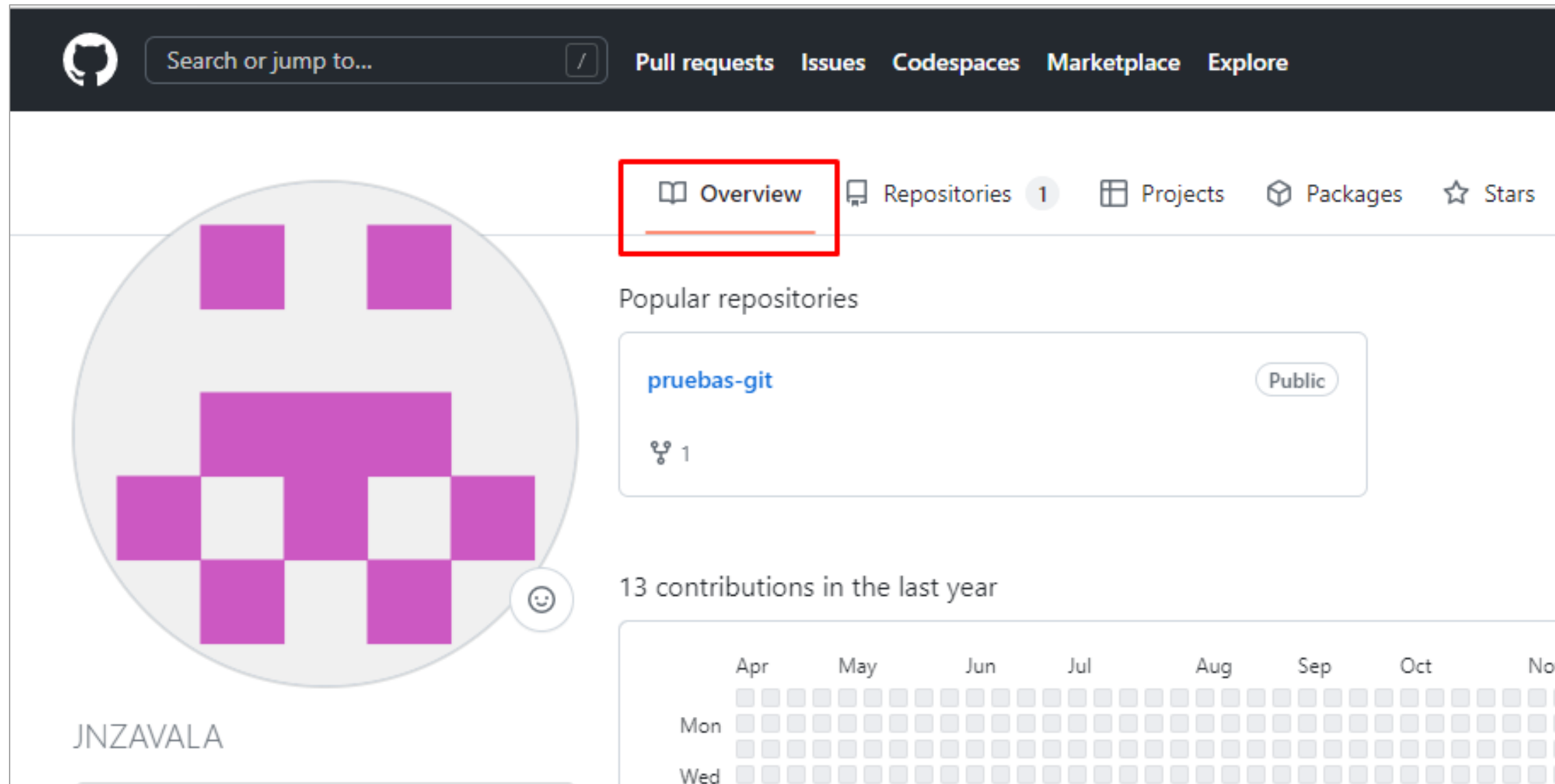
2023

March 2023



Comenzaremos con la pantalla principal de nuestro github, donde podremos visualizar una vista rapida de los trabajos y el trafico de datos que estos poseen.
Para ello se han marcado las características principales.

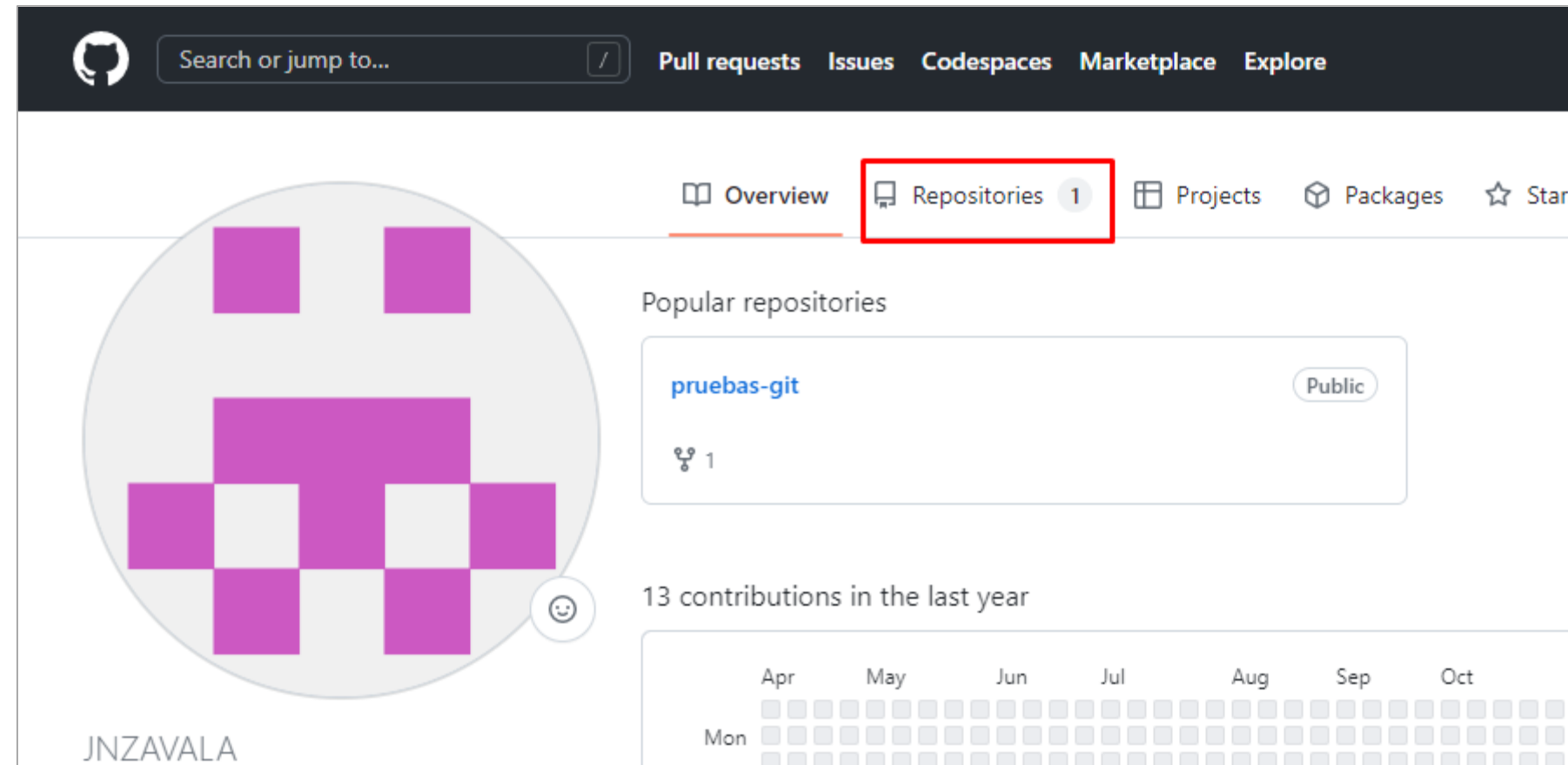
Overview



Para comenzar, nuestro dashboard contará con 2 pestañas muy importantes, las cuales serán las mas importantes y las que mas utilizarán.

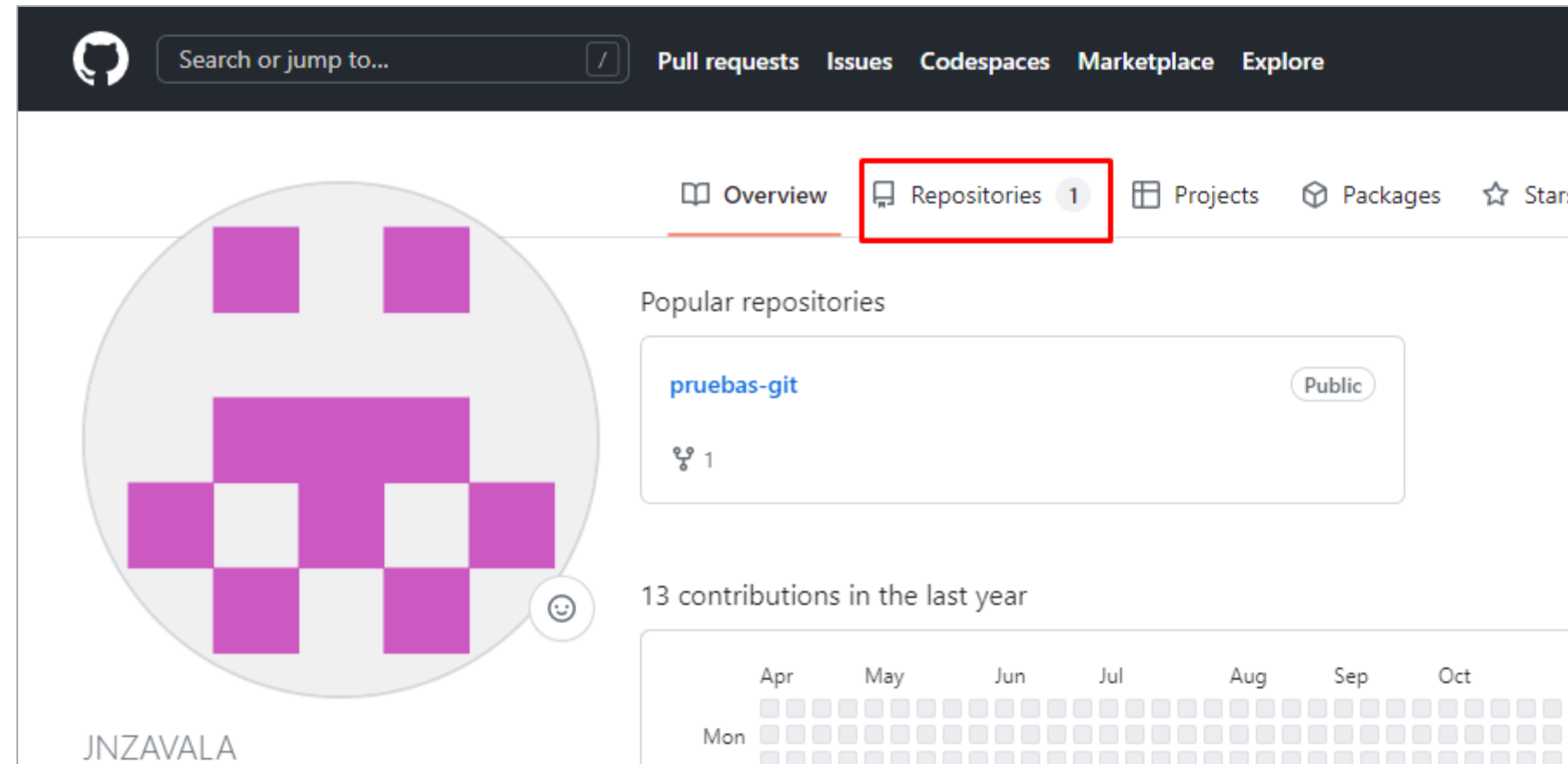
en donde Overview será nuestra vista principal y rapida, mostrando el trafico de actividad durante los dias y los repositorios mas visitados

Repositories



Luego tendremos Repositories(Repositorios), el cual será nuestro apartado que listará los repositorios creados y a nuestra disposición, de los cuales podemos ser partícipes o dueños (**Owners** en Github)

CREACION DE REPOSITORIO

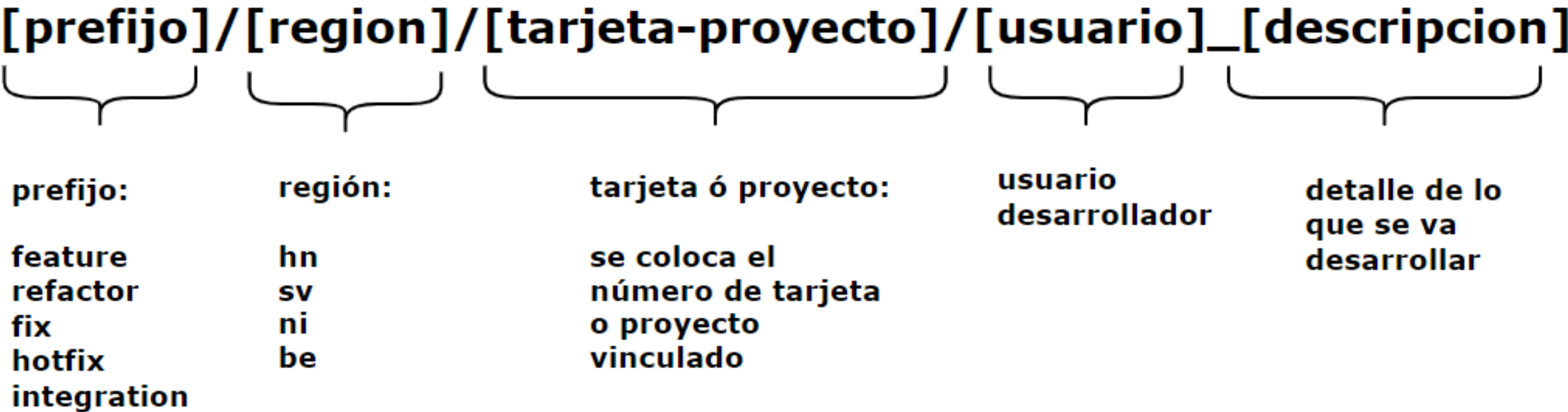


Para comenzar con la creación navegaremos hasta
“Repositories”

NOMENCLATURA DE RAMAS



La estructura de una rama creada por el administrador para un desarrollador, estará definida de la siguiente manera:



EJEMPLOS:

fix/hn/1000123/BA_dmeza_dev_correccion_pantalla_usuarios

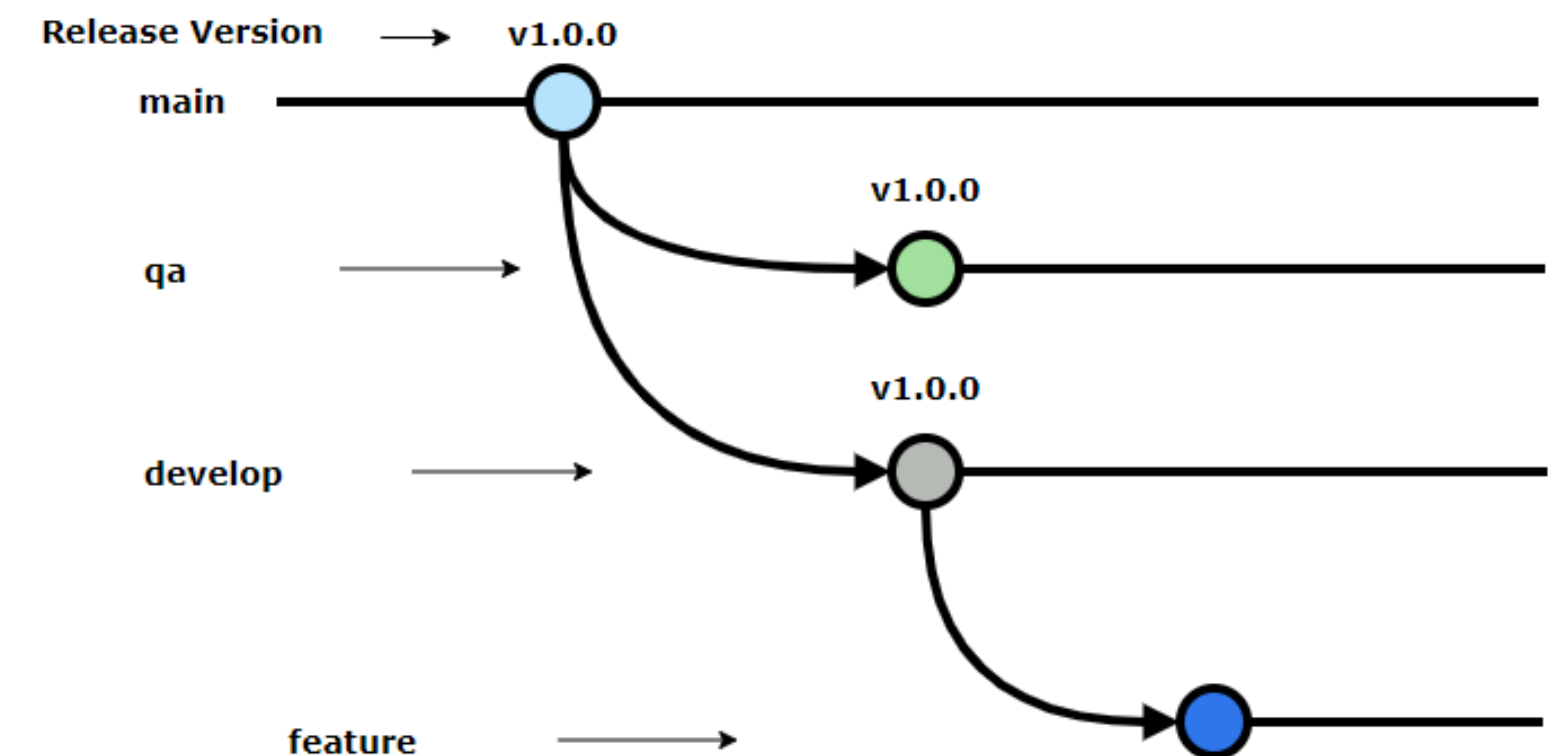
feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios



Una vez creada la rama, la estructura de nuestro proyecto a nivel de GitHub se verá de la siguiente manera:

- Para nuestro ejemplo, la rama del desarrollador (**feature**), se crea a partir de la **rama principal de desarrollo (develop)**.
- Los cambios del desarrollador, se irán integrando (**commits**) a la rama que se creó específicamente para el cambio solicitado.
- Esta rama (**feature**), sólo puede ser utilizada por el desarrollador que la solicitó.

Creación de rama para el desarrollador



PROYECTOS



Consideraciones al momento de integrar los cambios a los distintos ambientes:

- Para la integración al ambiente de desarrollo, se creará, por parte del administrador, una rama especial para ese cambio. La rama a utilizar tendrá la siguiente estructura:

develop/[prefijo]/[region]/[tarjeta-proyecto]/[usuario][descripcion]

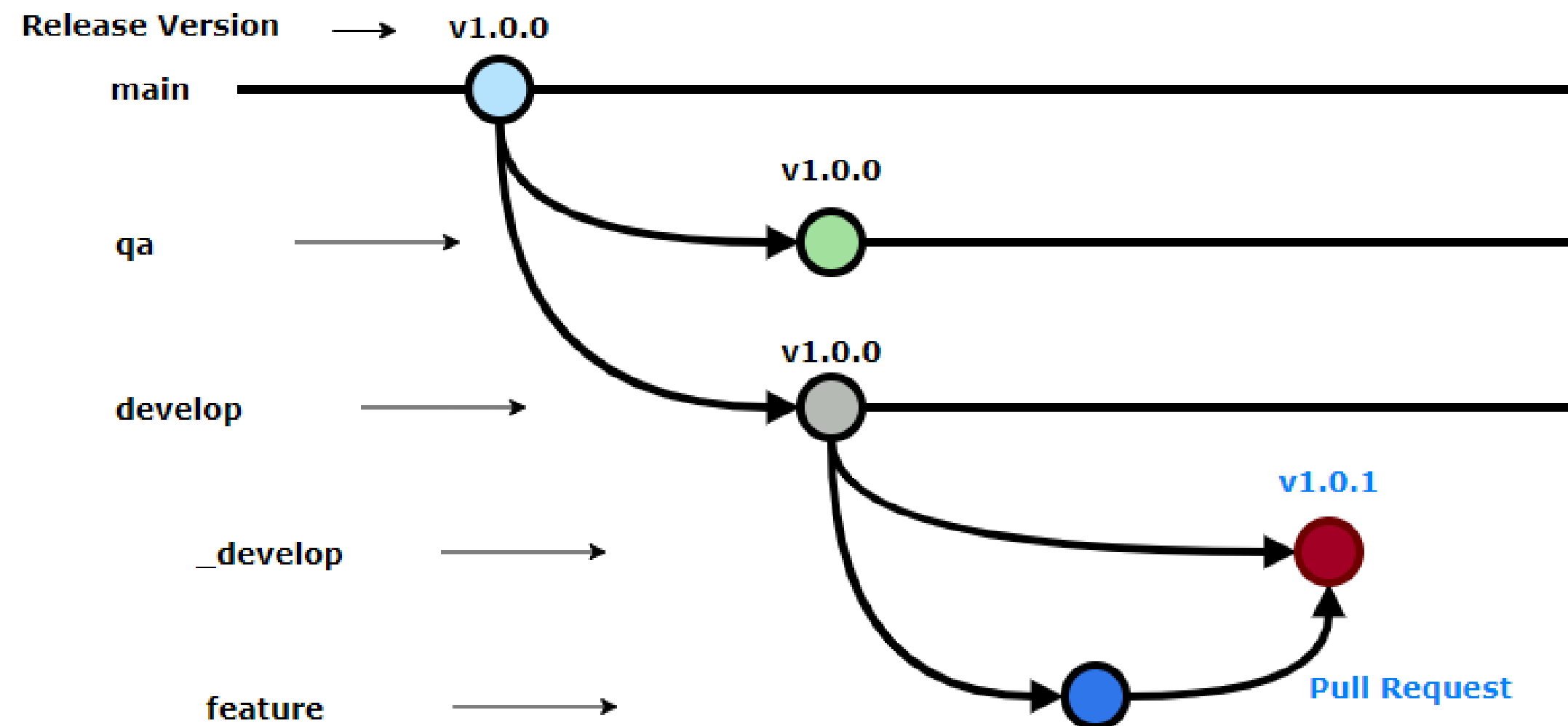
Ejemplo: _develop/feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios

- Si al momento de la **integración de los cambios** al ambiente de desarrollo (**_develop**), existiera algún conflicto por alguna nueva versión, el desarrollador deberá descargar esos cambios a su rama de desarrollo (**merge develop -> feature**) y validar los conflictos existentes. Luego de solventarlos, se solicitará la integración nuevamente.



Cuando se finalicen los cambios por parte del desarrollador, se integrará a su respectiva rama **_develop**, así como se ve en el siguiente ejemplo:

Integración de rama desarrollor a **_develop**



- Finalizando la integración en el ambiente de desarrollo (**_develop**); si no existen conflictos, se procederá con la integración al ambiente de certificación (**_qa**).
- Al igual que la integración al ambiente de desarrollo, se creará por parte del administrador, una rama especial para su certificación. La rama a utilizar tendrá la siguiente estructura:

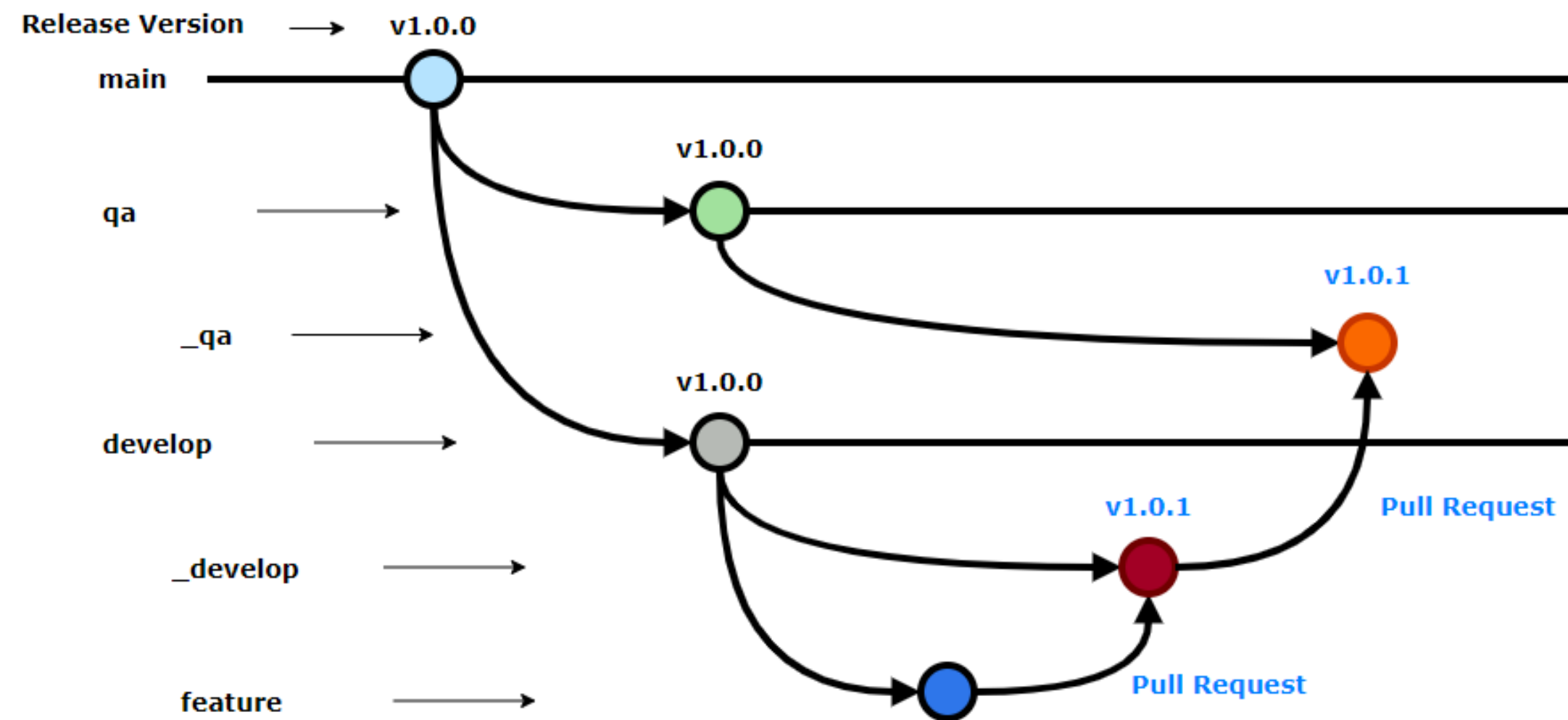
qa/[prefijo]/[region]/[tarjeta-proyecto]/[usuario][descripcion]

Ejemplo: _qa/feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios

1. Si al momento de la **integración de los cambios** al ambiente de certificación (**_qa**), existiera algún conflicto por alguna nueva versión, el desarrollador deberá descargar esos cambios a su rama de desarrollo (**merge develop -> feature**) y validar los conflictos existentes. Luego de solventarlos, se solicitará la integración nuevamente.

- Este es un ejemplo de cómo será la integración del ambiente de desarrollo (**_develop**) al ambiente de certificación (**_qa**).

Integración de **_develop** a **_qa**





En esta etapa, nuestro cambio se encuentra en proceso de certificación. Algunas de las consideraciones en esta etapa son:

- Si se presentan inconvenientes o errores en la etapa de certificación, para poder solventar los errores, el desarrollador deberá utilizar la rama asignada:

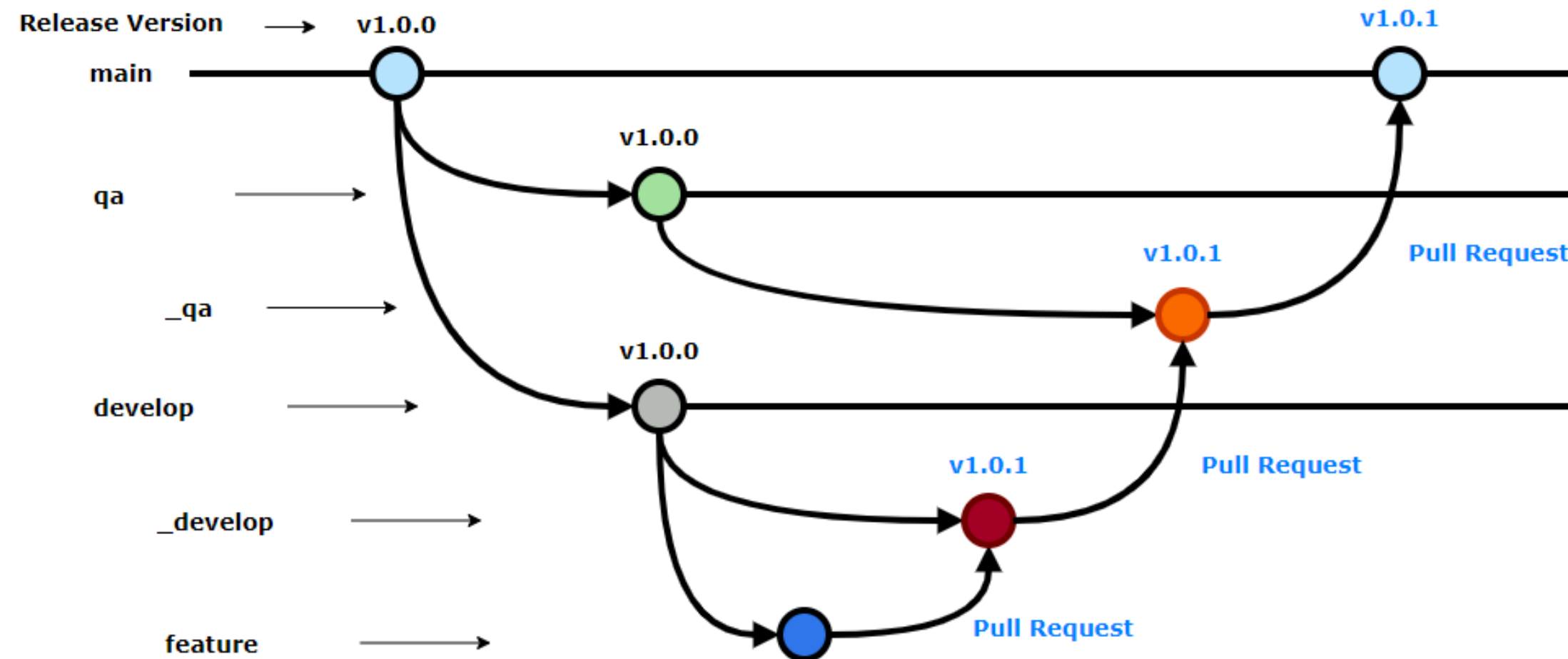
Ejemplo: `feature/hn/proyecto_abc/BA_prodriguez_dev_creacion_pantalla_usuarios`

- Una vez finalizadas las correcciones, se solicitará nuevamente la integración al ambiente de desarrollo (**_develop**) y, posteriormente, al ambiente de certificación (**_qa**).

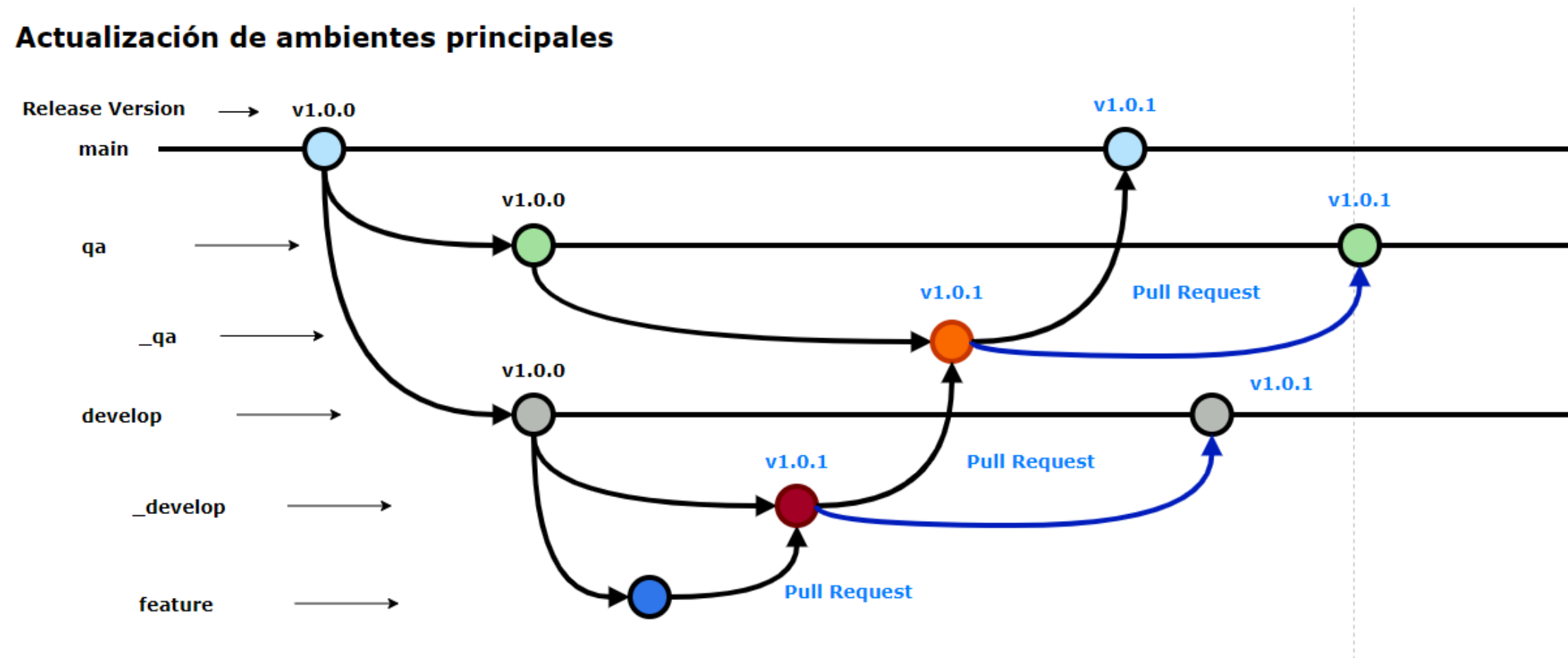
IMPORTANTE: las ramas utilizadas para el desarrollo de la funcionalidad, **NO se eliminarán** sino hasta que el cambio sea **implementado en ambiente de producción y el periodo de garantía haya finalizado**.

- Una vez certificado el cambio, se realizará la integración al ambiente de producción **(main)**.

Integración de _qa a main (producción)

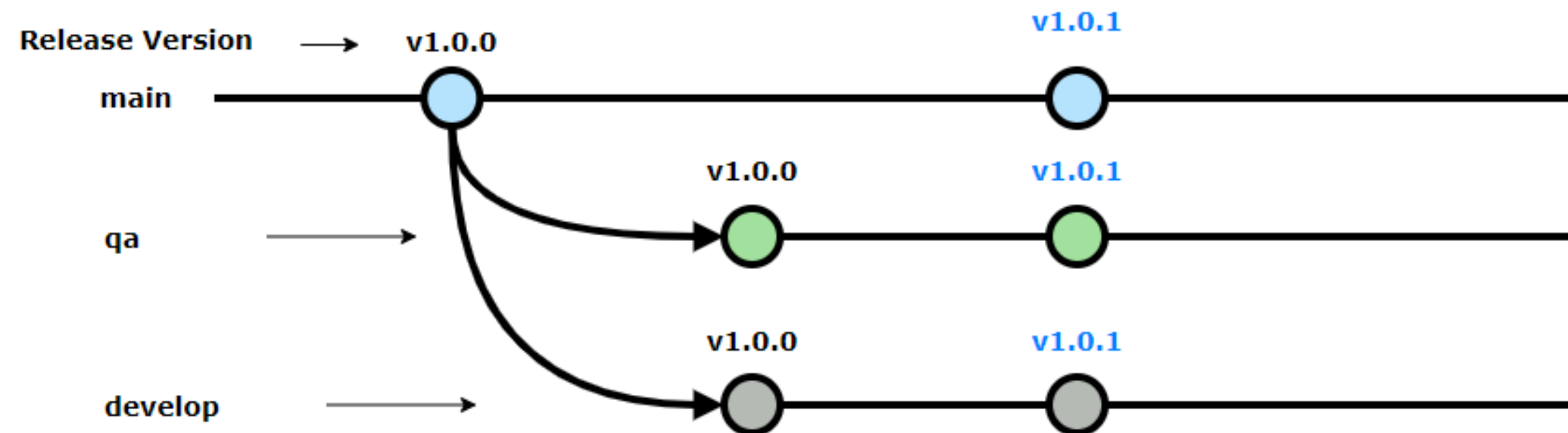


- Una vez el cambio **se integre al ambiente de producción y finalice el tiempo de garantía**, se deberán actualizar los ambientes principales de desarrollo (**merge _develop -> develop**) y de certificación (**merge _qa -> qa**), con la nueva versión estable.



- Cuando los ambientes principales de desarrollo (**develop**) y el de certificación (**qa**) estén actualizados, **el administrador eliminará las ramas** que se crearon para el desarrollo (**feature, _develop y _qa**).

Eliminación de ramas de desarrollador



INCIDENTES / PROBLEMAS



Para el manejo de **incidentes y problemas**, se tomará en consideración la misma estrategia mencionada para los **proyectos**, con las siguientes diferencias:

1. Para el caso de los desarrollos que se consideran **mejoras o ajustes a aplicaciones existentes (Incidentes / Problemas)**, el proceso parte de una **etapa de priorización**, considerando lo siguiente:
 - a) Si se tienen 2 o más desarrollos en paralelo, donde se estén realizando cambios en los mismos archivos, procesos o servicios, se debe entrar en una etapa de **priorización de cambios**. Se debe valorar qué cambio es más importante que sea certificado y posteriormente puesto en producción. Se opta por este proceso; ya que, de lo contrario, podría provocar que se envíen cambios a certificación y/o producción que no están finalizados.

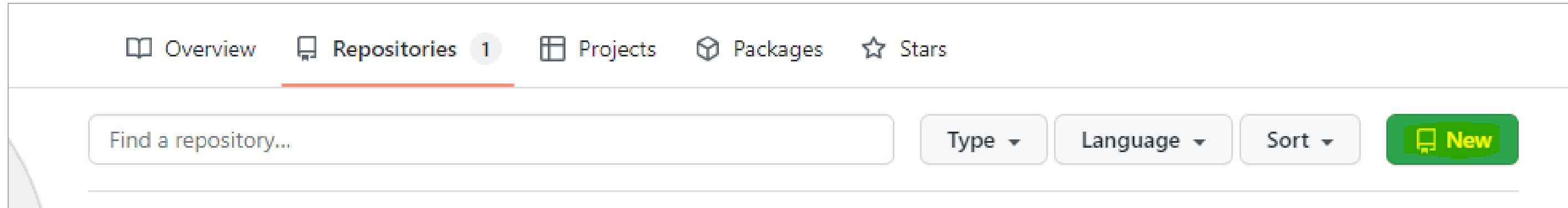


- b) Para los proyectos, **no será necesario** el envío de formato de solicitud de gestión de ramas cuando se requiera realizar la **integración de cambios de la rama del desarrollador a la rama _develop**. Para el caso de **Incidentes / Problemas**, el envío de esta solicitud es **obligatoria**.
- c) Para los **Incidentes / Problemas**, los accesos a los repositorios serán **temporales**. Una vez se finalice con la mejora o ajuste, se removerá al usuario del repositorio.

Modulo #3: Repositorio y ramas

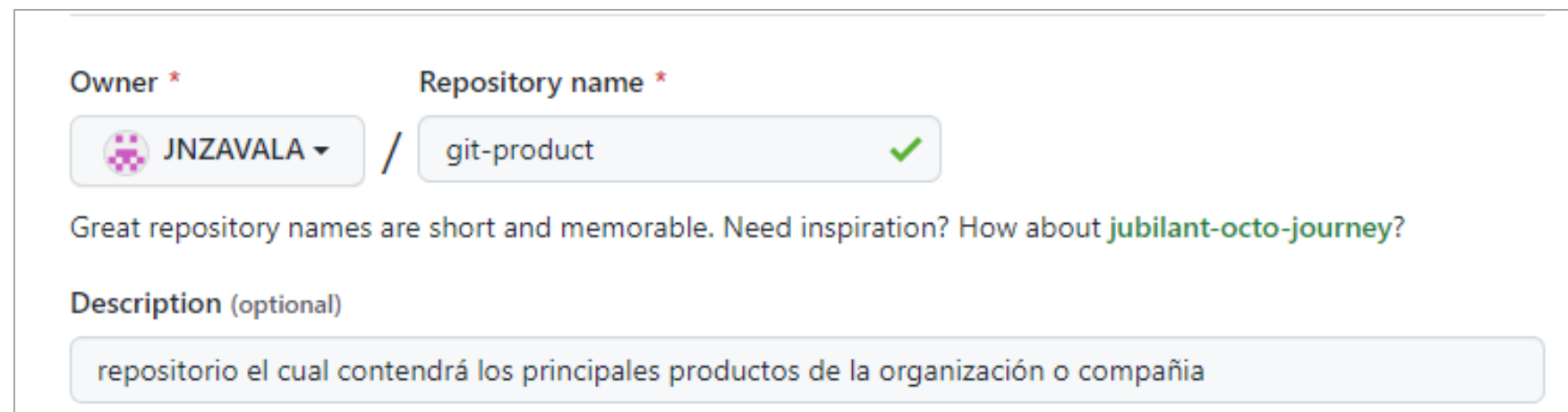


DATOS GENERALES DE REPOSITORIO



The screenshot shows the top navigation bar of the GitHub interface. The 'Repositories' tab is selected and underlined. Below the navigation bar, there is a search bar labeled 'Find a repository...', followed by three filter buttons: 'Type', 'Language', and 'Sort'. To the right of these buttons is a green 'New' button with a repository icon.

En la pestaña Repositories tendremos el botón New el cual nos permitirá iniciar con la creación




The screenshot shows the 'New repository' form. It has two main sections: 'Owner' and 'Repository name'. The 'Owner' section has a dropdown menu with 'JNZAVALA' selected. The 'Repository name' section has a text input field with 'git-product' and a green checkmark. Below these sections, there is a hint: 'Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-journey?](#)'. At the bottom, there is a 'Description (optional)' section with a text input field containing 'repositorio el cual contendrá los principales productos de la organización o compañía'.

llenaremos los campos generales para su creación, siempre y cuando se mantenga el estandar para los nombres largos siempre separados por guiones (-)





FINALIZACION DE REPOSITORIO

This will set  `main` as the default branch. Change the default name in your [settings](#).

Grant your Marketplace apps access to this repository

You are subscribed to 1 Marketplace app

☒  **Microsoft Teams for GitHub**
Connect your code without leaving Microsoft Teams

 You are creating a public repository in your personal account.

[Create repository](#)

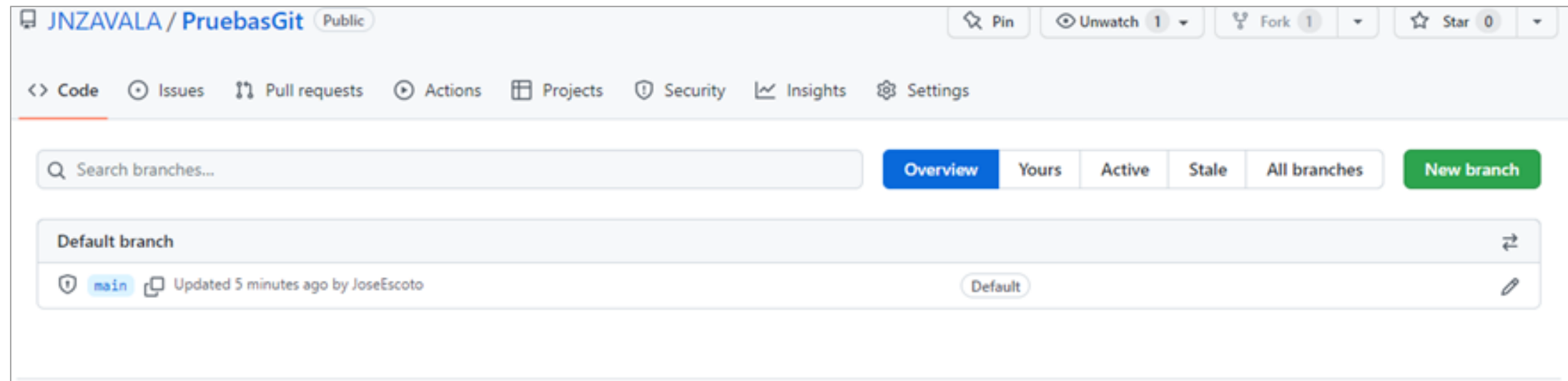


Si terminaste con los campos generales, ¡todo está listo!



Dato extra: si utilizas **Microsoft Teams** para tu trabajo, este puede ser integrado a **github**.

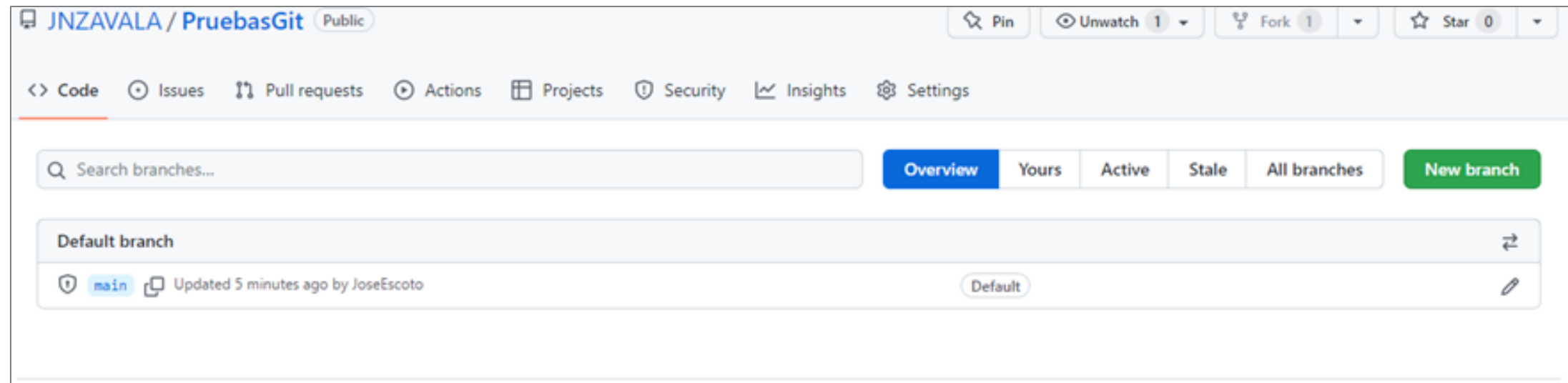
RAMA MAIN



Una vez creado el repositorio, este nos creará una rama main por defecto.

Ahora comenzaremos con la creación de las ramas necesarias para la gestión.

CREACION DE RAMAS



Haciendo uso del boton New branch, realizaremos la creacion de las ramas nuevas y replicaremos la estructura esperada.

RAMAS DESARROLLO

Create a branch

×

Branch name

develop

Branch source

main

Beta

Share feedback

Create branch

Create a branch

×

Branch name

`_develop/feature/hn/10001001/actualizacion_commits`

Branch source

develop

Beta

Share feedback

Create branch

Haciendo uso del boton New branch, realizaremos la creacion de las ramas nuevas y replicaremos la estructura esperada.

RAMAS DE QA

Create a branch

Branch name

qa

Branch source

main

Beta Share feedback

Create branch

Create a branch

Branch name

_qa/feature/hn/10001001/creación_flujo_usuarios

Branch source

qa

Beta Share feedback

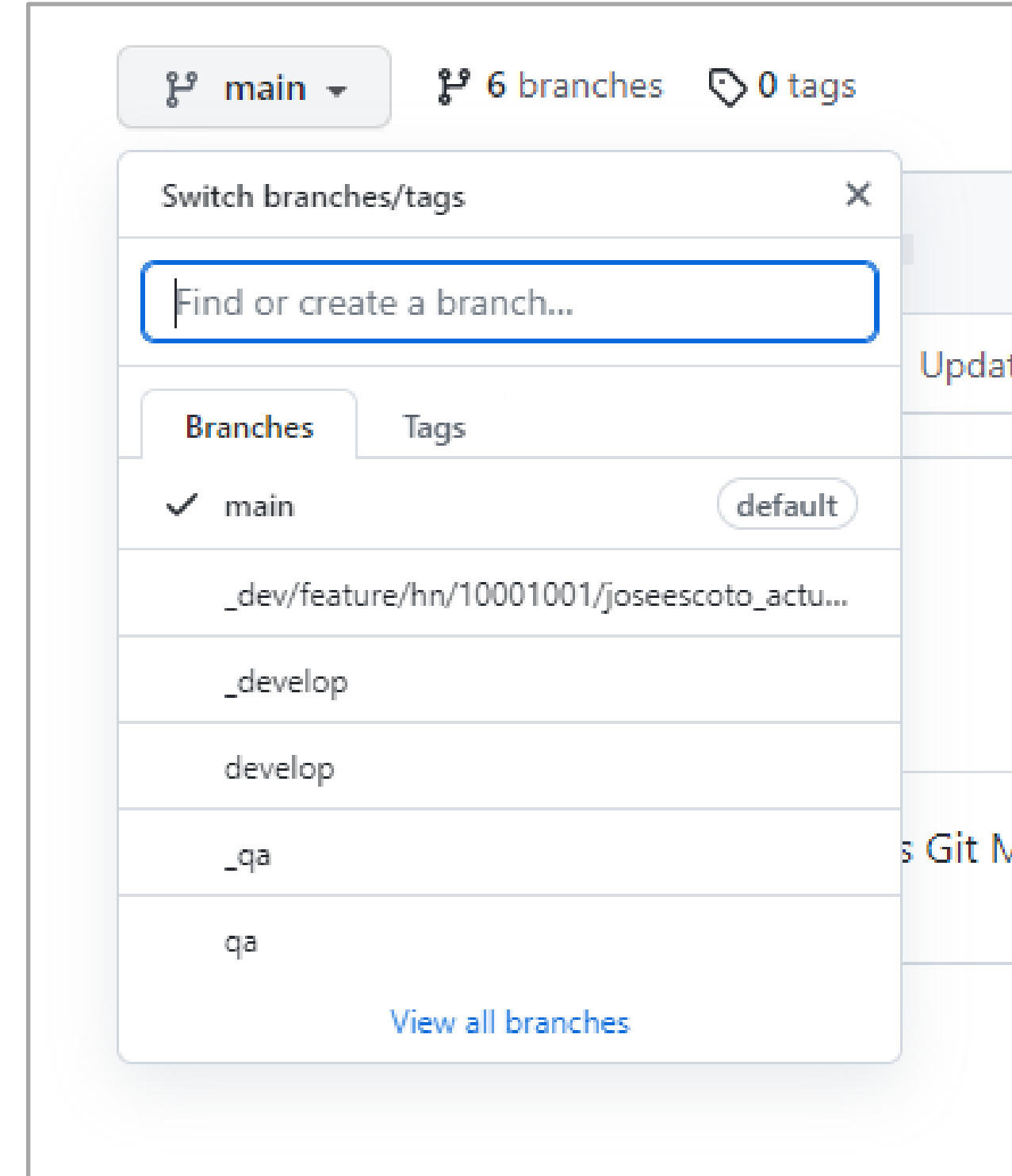
Create branch



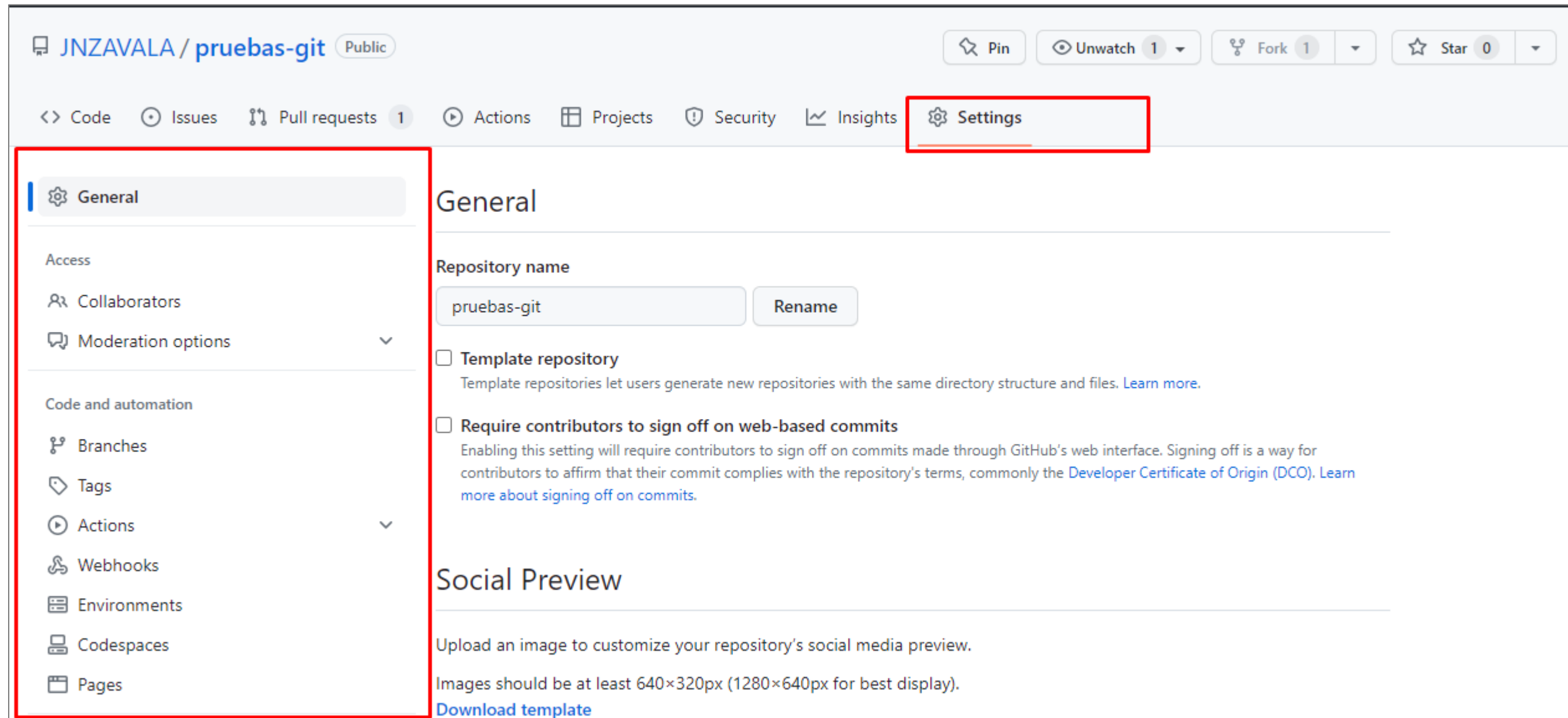
La segunda rama de qa tendrá la peculiaridad de que existirá conforme a la función creada y el desarrollador que la trabaje.

LISTADO DE RAMAS

Una vez creadas todas las ramas estas se verán reflejadas de la siguiente manera



SEGURIDAD EN RAMAS



Muy bien! Tenemos nuestro repositorio y nuestras ramas creadas, ahora pasaremos a un punto muy importante, la seguridad.

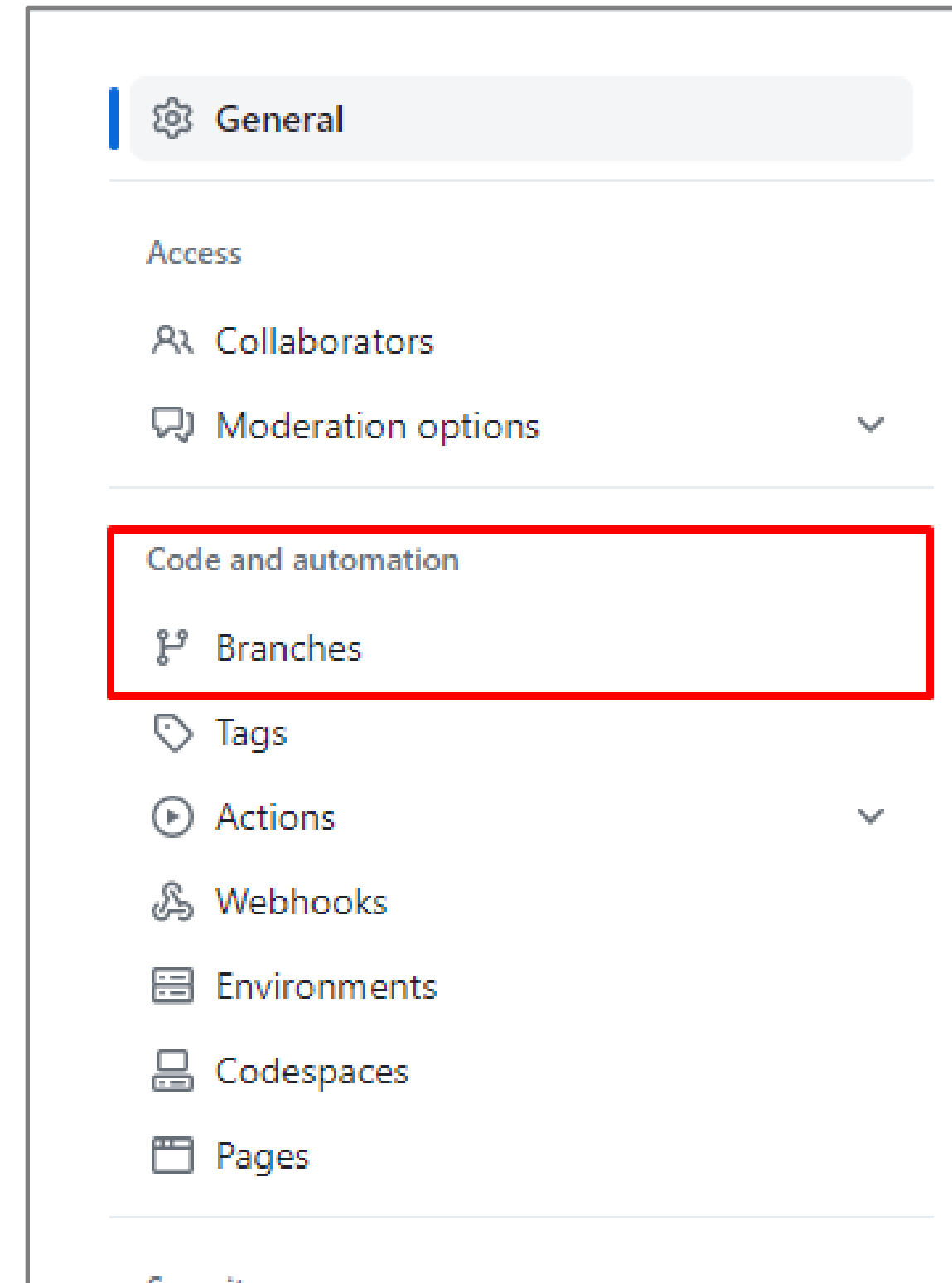
Esto será posible gracias al apartado Settings de nuestro github

NAVEGACION SETTINGS



Dentro de la configuración en la barra de navegación poseemos el apartado Branches o "ramas" en español

Aquí será donde las reglas y la capa de seguridad para las ramas será creada



REGLAS DE RAMAS



Si navegamos hasta aquí, por defecto nos aparecerá nuestra rama main

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main✎ ↺

Para la creación de una nueva regla en el apartado de "branch protection rules", daremos click en "Add rules"



Branch protection rules

Add rule

SELECCION DE REGLAS



Para la creación de las ramas y que esta sea aplicada donde pertenece, este deberá tener un patron en su nombre

Branch protection rule

Branch name pattern *

develop

Applies to 1 branch

develop

Protect matching branches

☒ **Require a pull request before merging**

When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ **Require approvals**

When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.

Required number of approvals before merging: 1 ▼

☐ **Dismiss stale pull request approvals when new commits are pushed**

New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

REGLAS PRINCIPALES

☒ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request to a branch that matches this rule.

☒ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and can be merged.

Required number of approvals before merging: 1 ▼

- ☒ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6

☐ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request to a branch that matches this rule.

☐ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and can be merged.

Required number of approvals before merging: 1 ▼

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6

☐ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request to a branch that matches this rule.

☐ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and can be merged.

Required number of approvals before merging: 1 ▼

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6



Seleccionaremos las siguientes reglas:
"Require a pull request before merging"
"Require approvals" y "do not allow bypassing the above settings"

REGLA #1 PULL REQUEST

☒ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request to a branch that matches this rule.

☒ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and can be merged.

Required number of approvals before merging: 1 ▼

- ☒ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6



"Require a pull request before merging" (para indicarle que se requiere un pullrequest, para que los cambios no entren sin un proceso respectivo de revisión)

REGLA #2 APPROVALS

☒ **Require a pull request before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request to a branch that matches this rule.

☒ **Require approvals**
When enabled, pull requests targeting a matching branch require a number of approvals and can be merged.

Required number of approvals before merging: 1 ▼

- ☒ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6

☐ Require a pull request before merging



"Require approvals" (para denotar que la revision requiere una cantidad de aprobaciones necesarias)

REGLA #2 Bypassing rule

☐ Lock branch
Branch is read-only. Users cannot push to the branch.

☒ Do not allow bypassing the above settings
The above settings will apply to administrators and custom roles with the "bypass branch protections" permission.

☐ Restrict who can push to matching branches
Specify people, teams, or apps allowed to push to matching branches. Required status checks will still prevent these people, teams, and apps from merging if the checks fail.



"Do not allow bypassing the above setting" (Regla aplicada para que los administradores no puedan omitir ninguna de las configuraciones anteriores)

LISTADO DE REGLAS





Cada una de estas reglas se establecerá para las ramas creadas previamente

Default branch

The default branch is considered the “base” branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

main

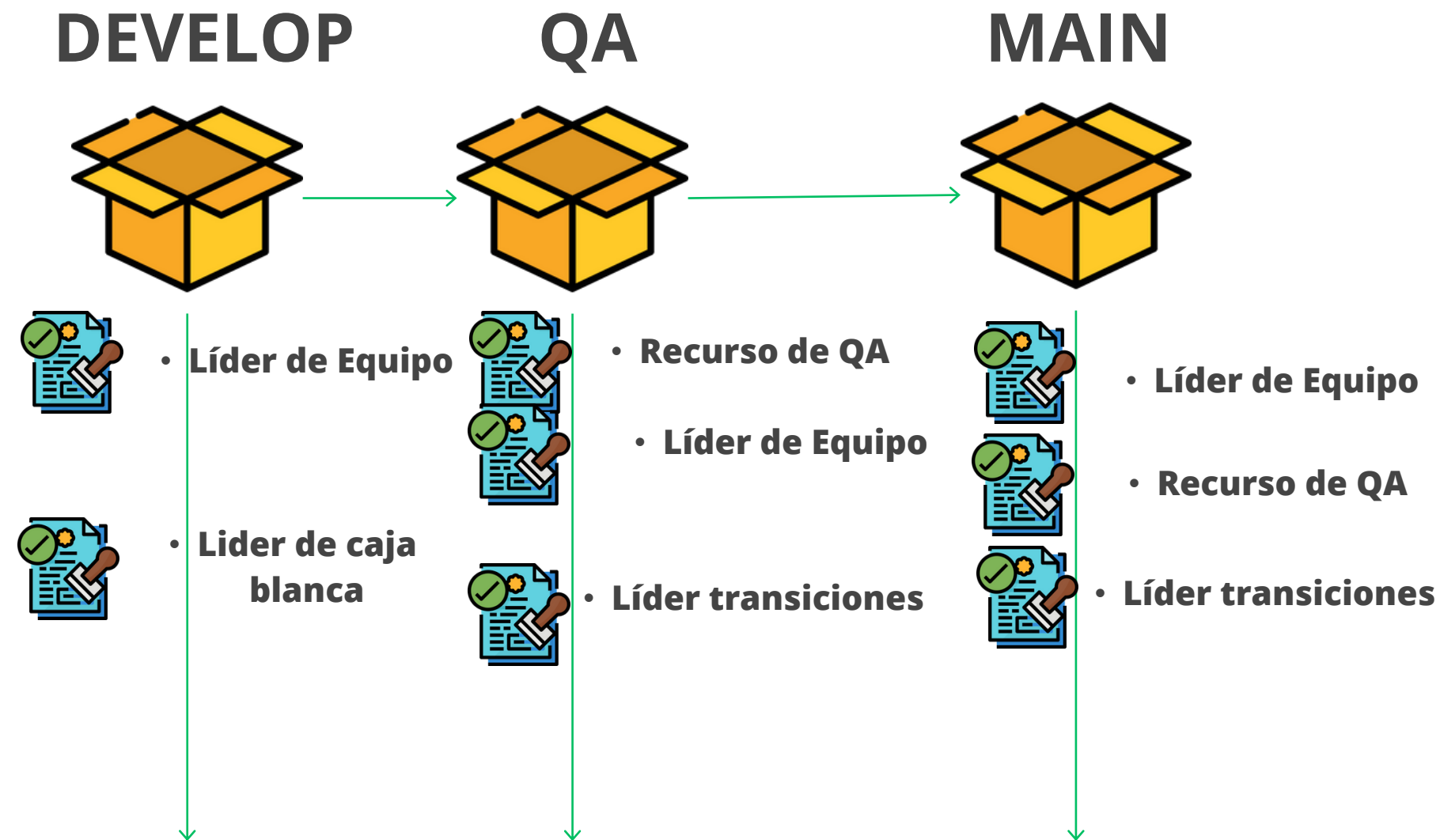


Branch protection rules

Add rule

main	Currently applies to 1 branch	Edit	Delete
develop	Currently applies to 1 branch	Edit	Delete
qa	Currently applies to 1 branch	Edit	Delete
_qa	Currently applies to 1 branch	Edit	Delete
_develop/feature/hn*	Currently applies to 0 branches	Edit	Delete

FLUJO DE VALIDACIONES



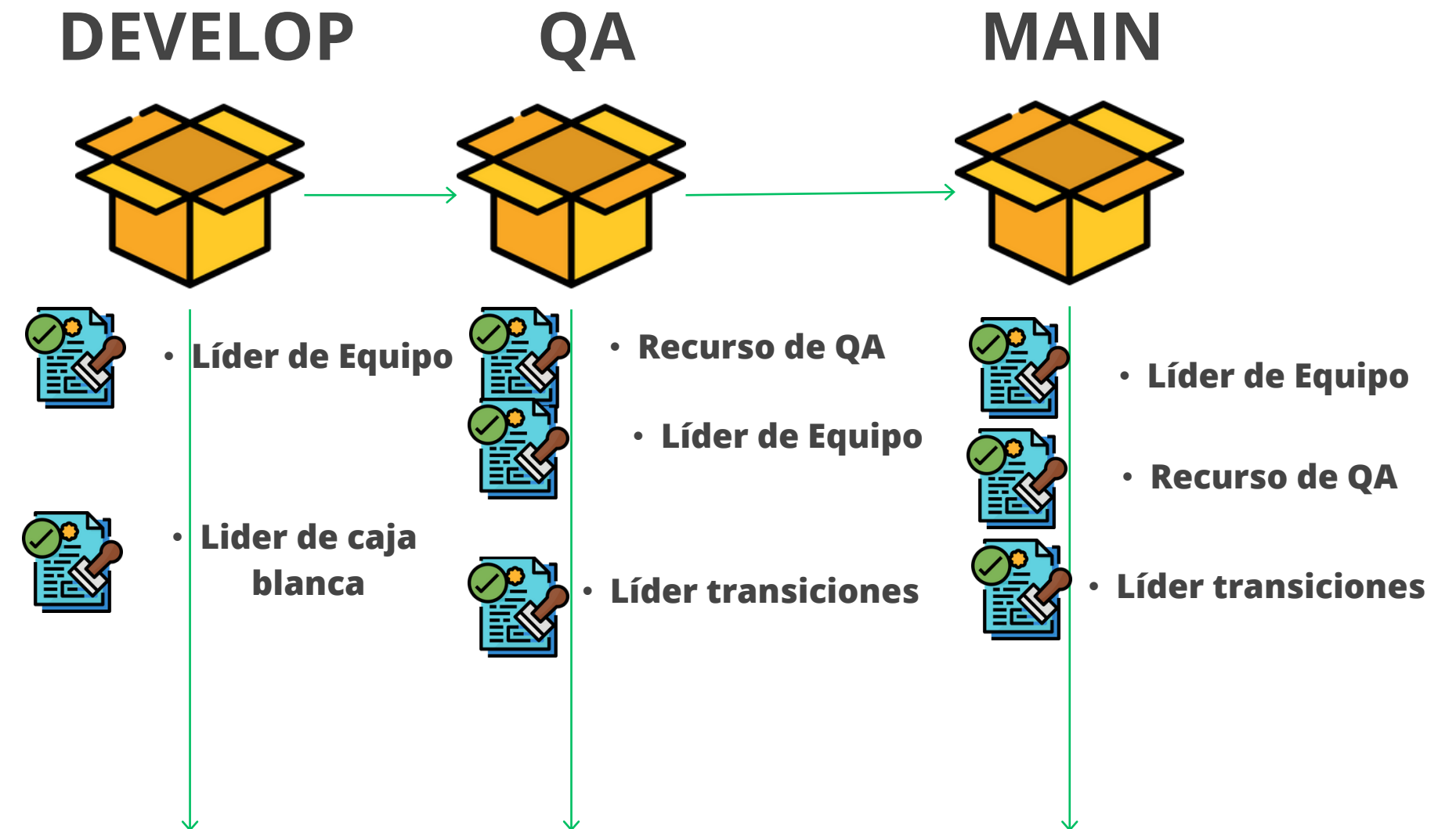
Aquí se puede apreciar la estructura de validaciones que se requieren para el flujo administrativo y el paso de un ambiente a otro.

CASO#1



¿Que sucede si en alguno de estos puntos, no se obtiene una de las aprobaciones requeridas o algo falla?

Eso quiere decir que el flujo de vida de los cambios y/o funcionalidades comienza de nuevo a raíz de una petición de cambio.

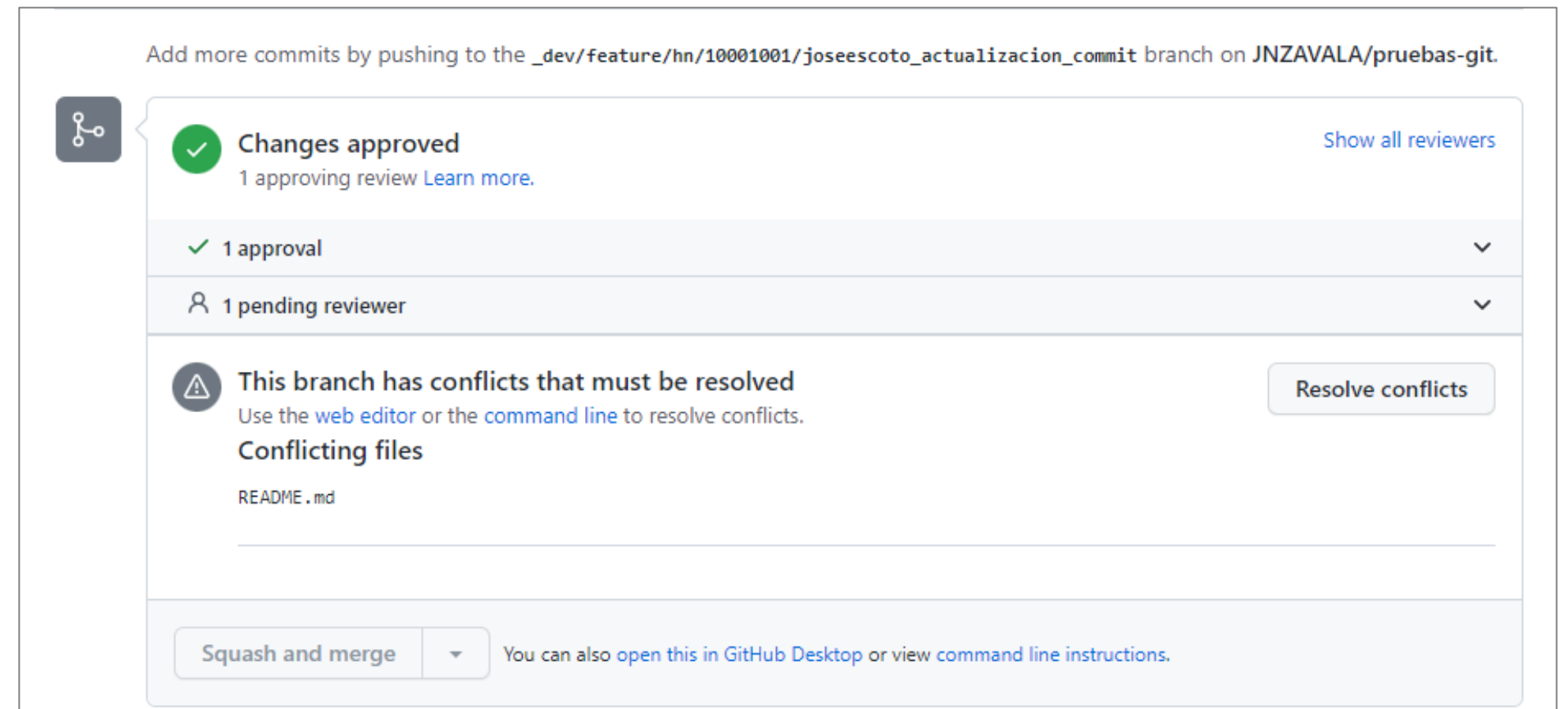


CASO#2



¿Qué hacer en caso de obtener conflictos?

Notificar a la persona que trabajó en los cambios y reiniciar el proceso de pullrequest luego de la solución de conflictos



Como **Administrador** no resolver conflictos



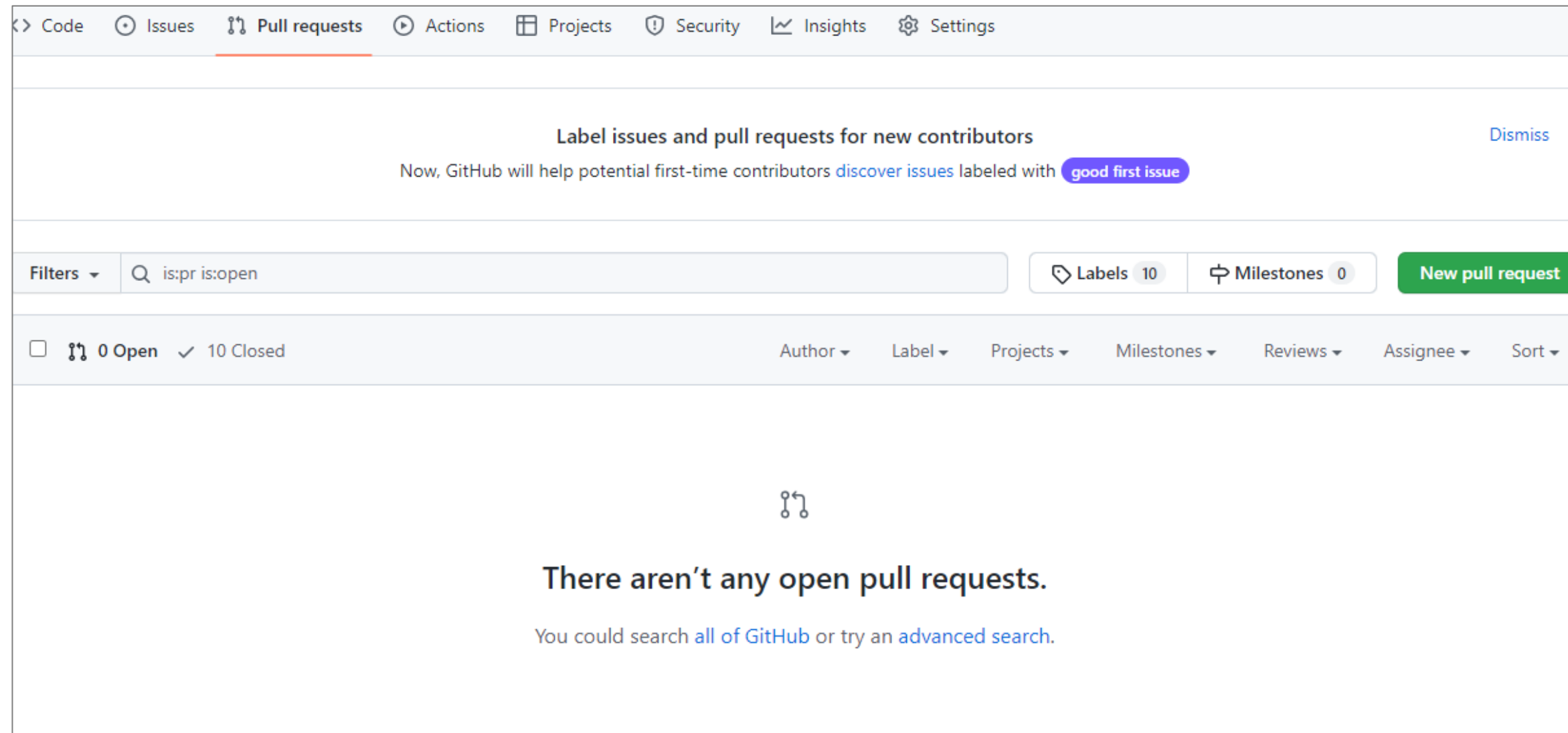
REQUERIMIENTOS PARA UN PULLREQUEST



Para que un pullrequest exista, deberán de cumplir una serie de puntos:

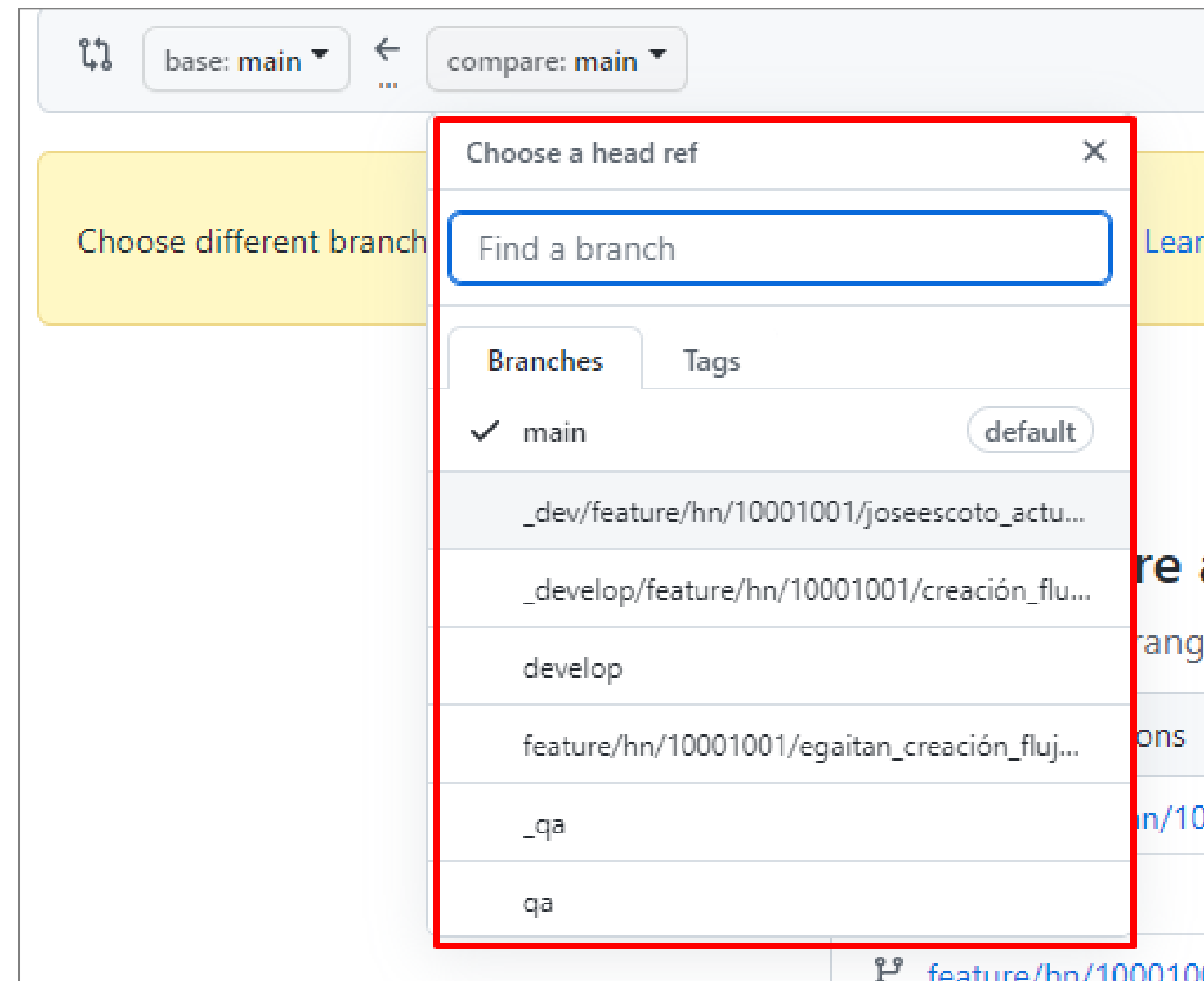
- Su ultimo Commit deberá de llevar el prefijo **END**(para identificar que este será el ultimo cambio realizado).
- convocar a los administradores y lideres que se ha realizado el último commit para que este sea revisado y pueda ser parte del proceso de despliegue y puesta en producción.
- Crear una nueva rama correspondiente a la funcionalidad desarrollada y que está sea replicada en cada uno de los ambientes establecidos.

PULLREQUEST



Repositorio? Listo! **Ramas?** Creadas! **Reglas?** Enlazadas.
perfecto, ahora será parte de la creación de los
pullrequest o Merge request

RAMAS DE PULLREQUEST

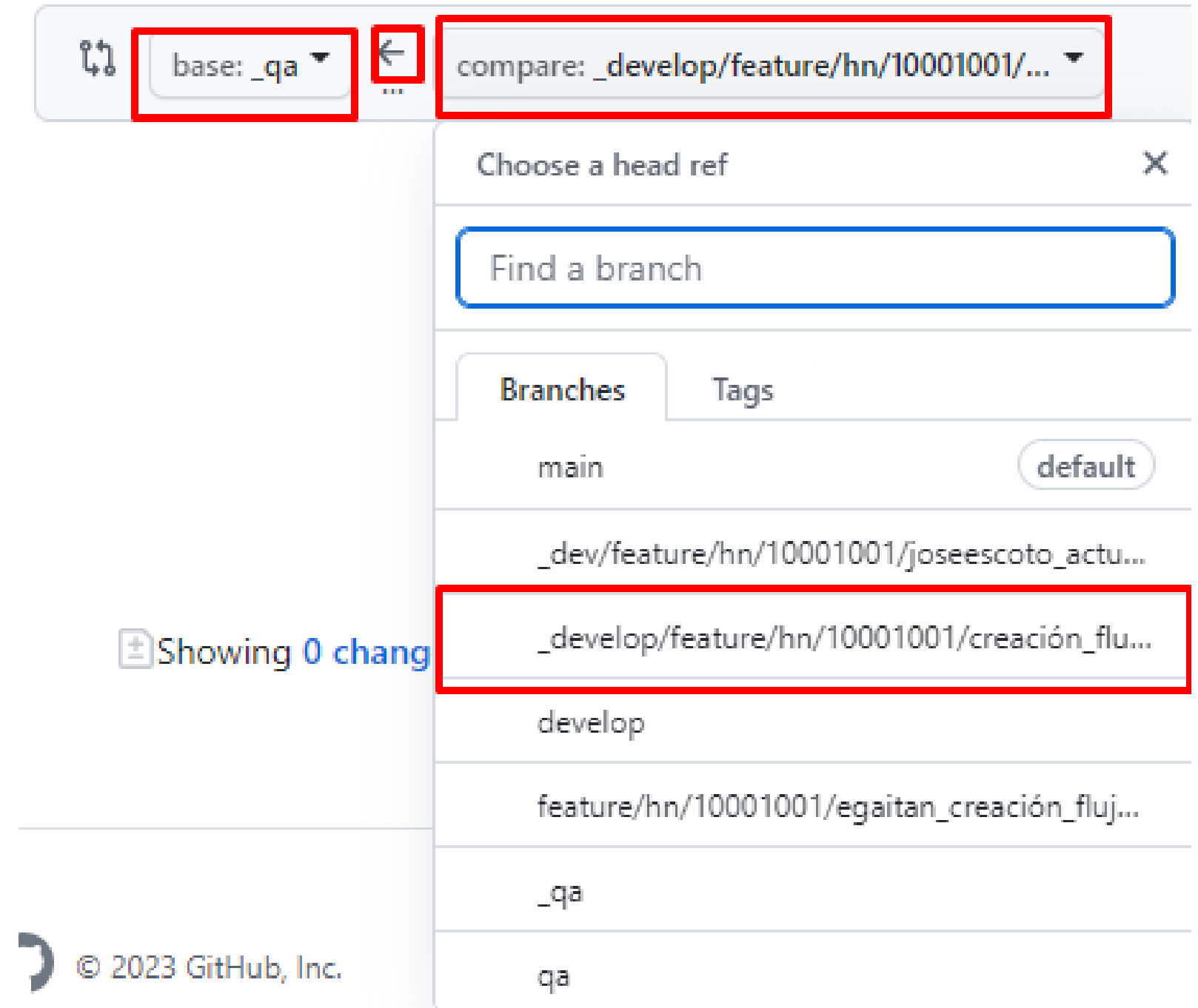


Para crearlo necesitaremos la rama de cambios y la rama donde se depositarán los cambios

CREACION DE PULLREQUEST



Es decir, el apartado Compare donde se seleccionará nuestra rama de funcionalidad, la cual sus cambios serán enviados al apartado base o nuestro ambiente indicado, como continuación nos muestra la pantalla



CREACION DE PULLREQUEST

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base: `_qa` ← compare: `_dev/feature/hn/10001001/jose...` ✗ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

5 commits 2 files changed 3 contributors

Commits on Mar 17, 2023

Update README.md JoseEscoto committed 2 days ago	Verified	<code>f36a877</code>	<>
Update README.md JoseEscoto committed 2 days ago		<code>76c2ad0</code>	<>
actualizacion de archivo readme y commitemplate JNZAVALA committed 2 days ago		<code>7a88eba</code>	<>
new push JNZAVALA committed 2 days ago		<code>012f14f</code>	<>
new commit JNZAVALA committed 2 days ago		<code>cdd0c03</code>	<>

Showing 2 changed files with 19 additions and 1 deletion.

Split Unified



Sí seguimos el flujo administrativo pasaremos de la rama del desarrollador a su rama de `_qa` designada unicamente a esta funcionalidad

↺

base: _qa ▾

←

compare: _dev/feature/hn/10001001/jose... ▾

✖ Can't automatically merge. Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

5 commits

2 files changed

3 contributors





una vez seleccionado el destino de los cambios pasaremos a crear el pull request mediante el boton **"create pullrequest"**


CREACION DE PULLREQUEST


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: `_qa`

compare: `_dev/feature/hn/10001001/jose...`

 **Can't automatically merge.** Don't worry, you can still create the pull request.



Dev/feature/hn/10001001/joseescoto/MensajeError

Write

Preview

H

B

I

≡

<>

🔗

≡

≡

🔍

@

🗨

↩

📌


Funcionalidad lista para ser evaluada en su proceso de certificación

Attach files by dragging & dropping, selecting or pasting them.

Create pull request


▼

Reviewers

 JoseEscoto

At least 2 approving reviews are required to merge this pull request.

Assignees

 JNZAVALA

Labels

bug

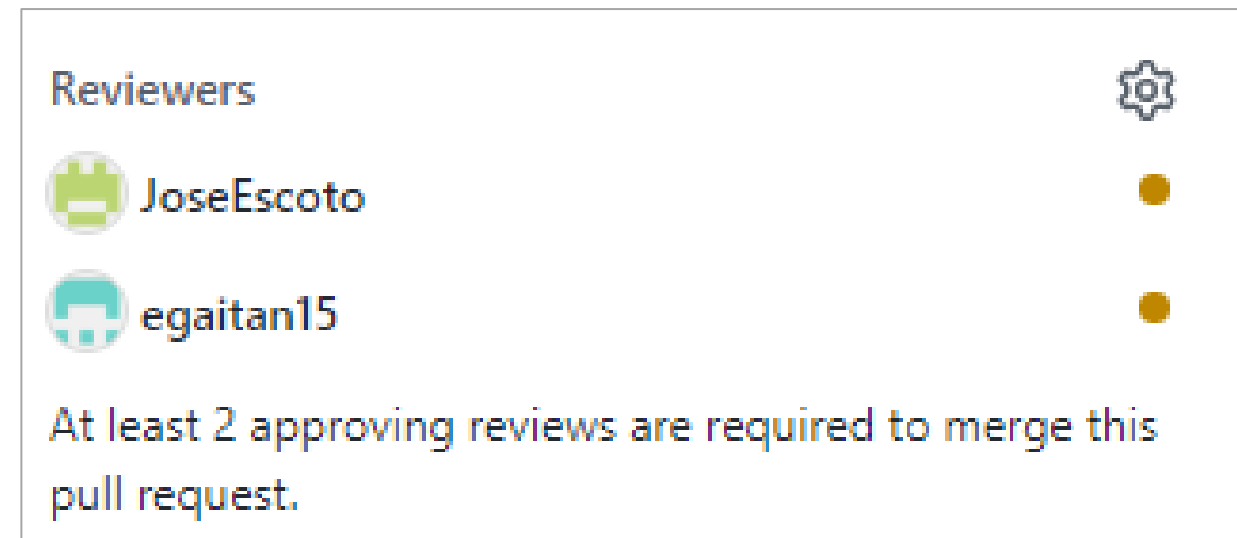
Projects

None yet

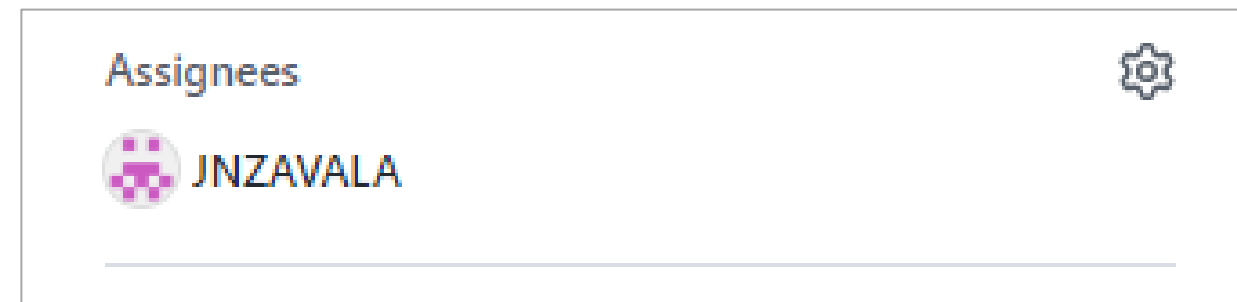
Milestone

Cada **pullrequest** supondrá un nombre y comentario de lo que se realiza y lo que se espera.

ROLES DE PULLREQUEST



Siguiendo el ejemplo podremos establecer que el usuario JoseEscoto es el lider del proyecto y egaitan15 es el recurso del nuevo equipo de pruebas de caja blanca





Por último tenemos el assignee quien es el administrador de transiciones establecido para el paso entre ambientes.






Reviewers & Assignees

Reviewers

 JoseEscoto

 egaitan15

At least 2 approving reviews are required to merge this pull request.

Assignees

 JNZAVALA





Es importante saber quien lo revisará y realizará el paso entre ambientes, en este caso los reviewers serán designados según el ambiente y el Assignee siempre será tu administrador mas cercano

REVIEW DE PULLREQUEST

Dev/feature/hn/10001001/joseescoto/MensajeError #11 Edit <> Code

Open JNZAVALA wants to merge 5 commits into `_qa` from `_dev/feature/hn/10001001/joseescoto_actualizacion_commit`

Conversation 0 Commits 5 Checks 0 Files changed 2 +19 -1

JNZAVALA commented 1 minute ago Owner ...

Funcionalidad lista para ser evaluada en su proceso de certificación

JoseEscoto and others added 5 commits 2 days ago

- Update README.md Verified f36a877
- Update README.md 76c2ad0
- actualizacion de archivo readme y committemplate 7a88eba
- new push 012f14f
- new commit cdd0c03

JNZAVALA added the **bug** label 1 minute ago

JNZAVALA requested review from **egaitan15** and **JoseEscoto** 1 minute ago

Reviewers

- egaitan15**
- JoseEscoto**

At least 2 approving reviews are required to merge this pull request.

Still in progress? [Convert to draft](#)

Assignees

- JNZAVALA**

Labels

- bug**

Projects

None yet

Milestone



Listo, al crear el pull request se apreciará esta pantalla donde se realizará la respectiva revision y paso de los cambios al siguiente ambiente



Muchas gracias por haber
prestado algo de tu tiempo.

¡Ahora la práctica hace al
maestro!

Continúa aprendiendo más
sobre GitHub.