RED UNIVERSITARIA DE COLABORACIÓN
EN INGENIERÍA DE SOFTWARE Y BASES DE DATOS

El uso y herramientas de Git y GitLab

ING. MIGUEL ANGEL MARTINEZ

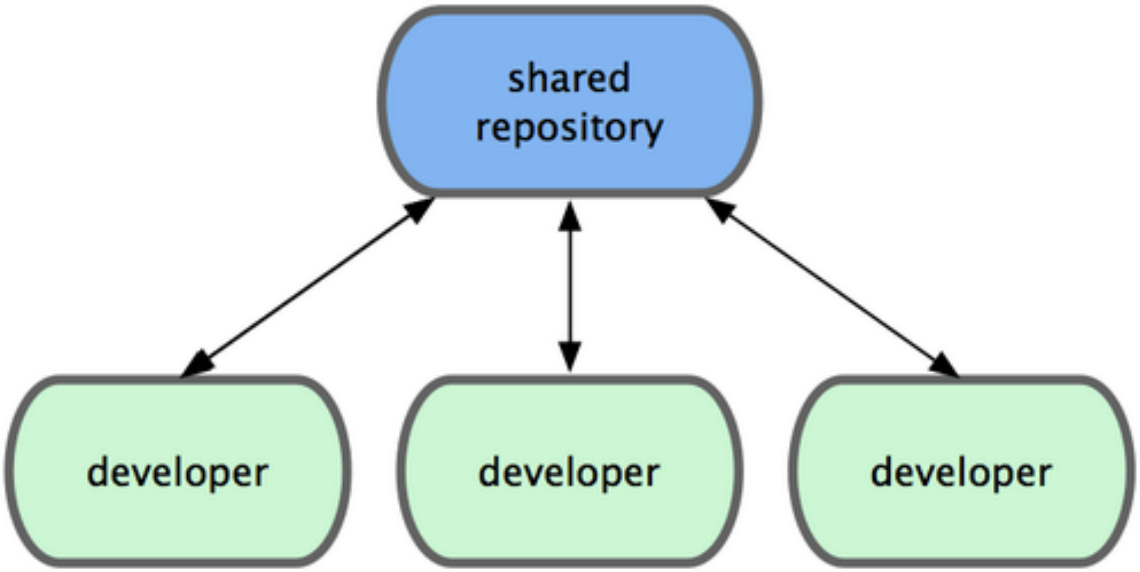
ING. OSCAR ALVAREZ ING. OMAR BAHENA ING. MA. DEL ROCIO CARRILLO

PRIMER MÓDULO

GIT

¿Qué es Git?

- Sistema de control de versiones distribuido.
- Código abierto y gratuito.
- Diseñado para proyectos grandes y pequeños.



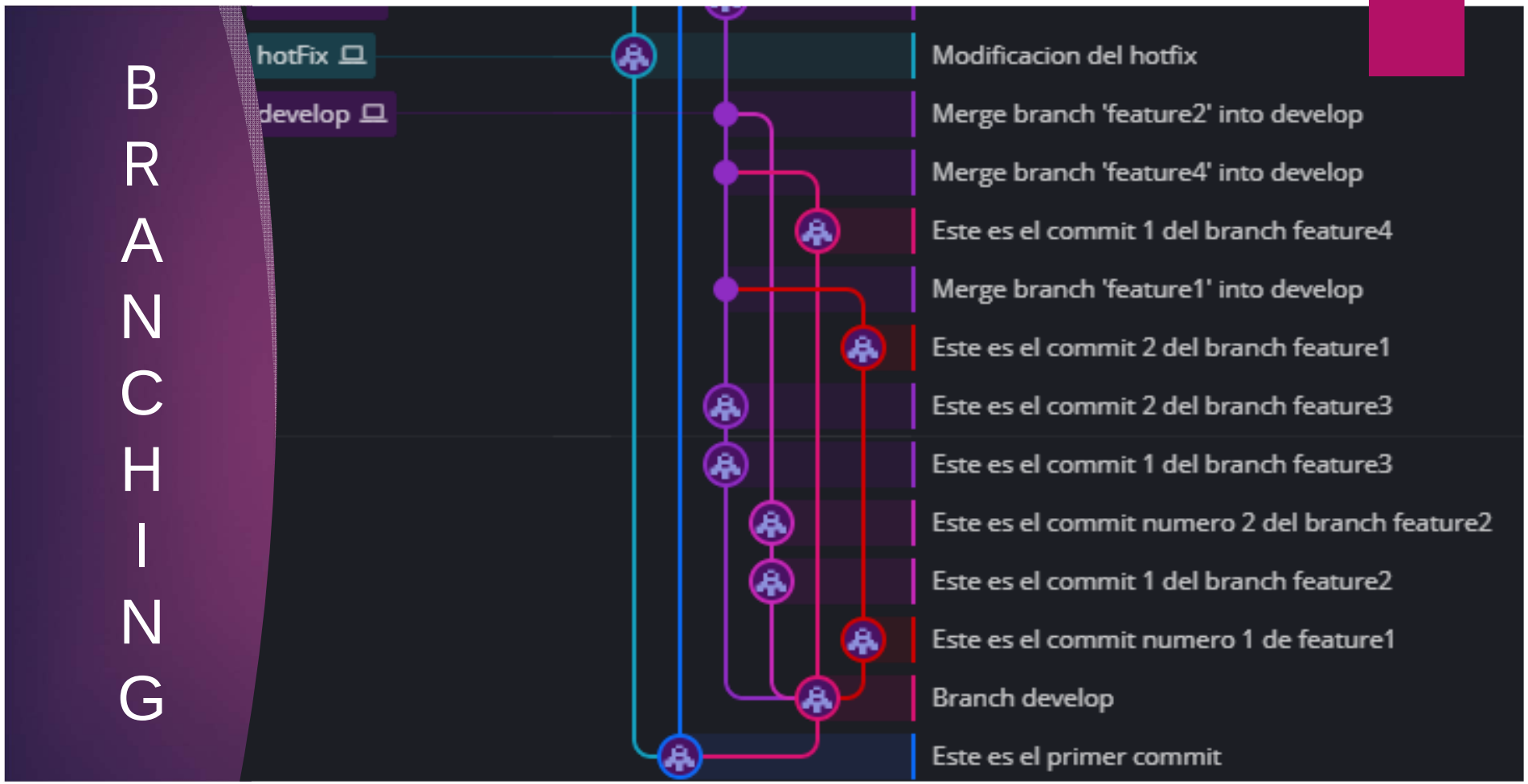
Historia

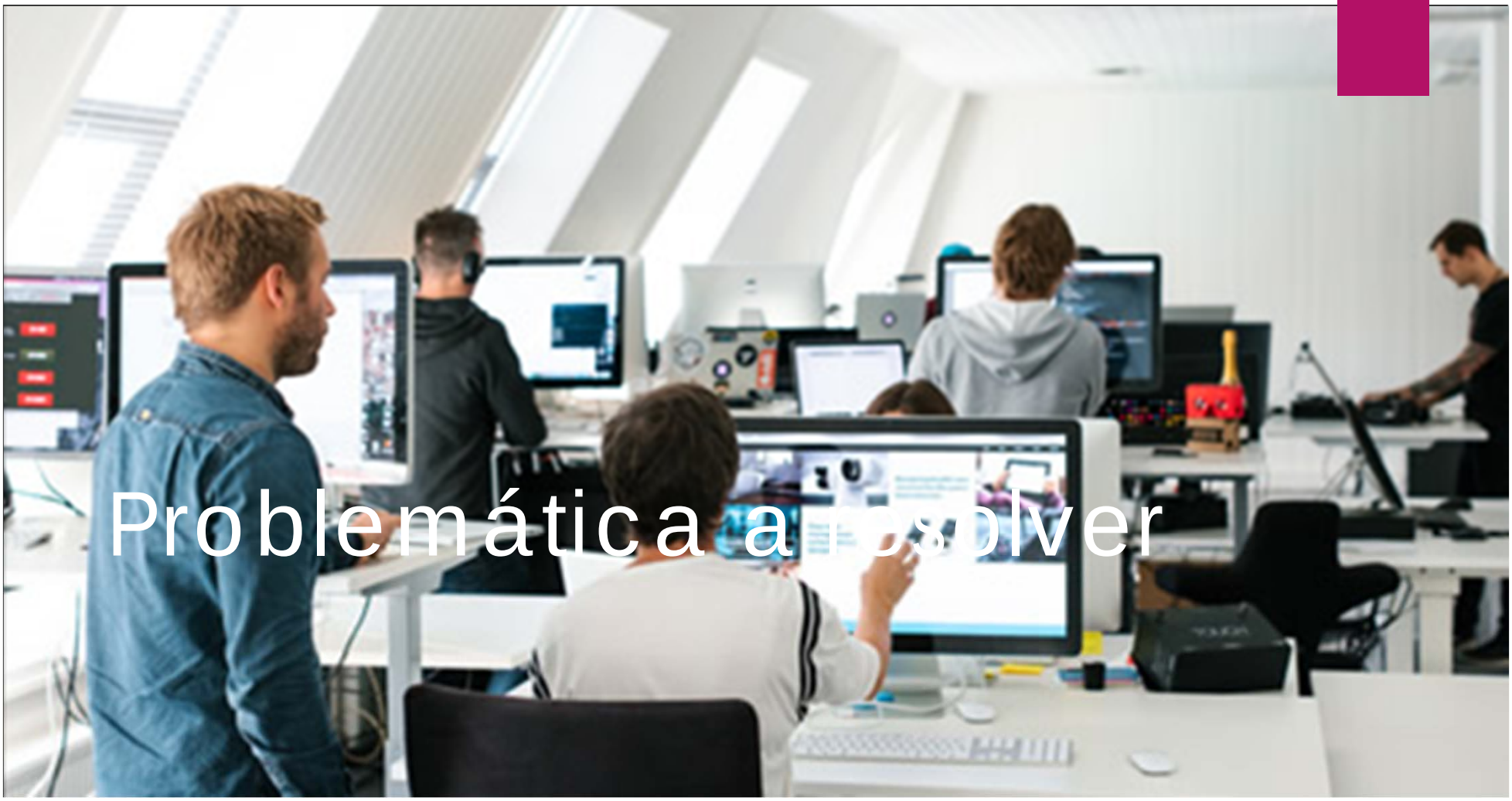

1991 - Cambios en el software se pasan en forma de parches y archivos
2002 - DVCS propietario BitKeeper
2005 - Desarrollo de su propia herramienta
2009 – Mantenimiento de Git

Características

- Velocidad
- Diseño sencillo
- Completamente distribuido
- Se adapta para todo tipo de proyectos.
- Desarrollo no lineal (branching).

BRANCHING

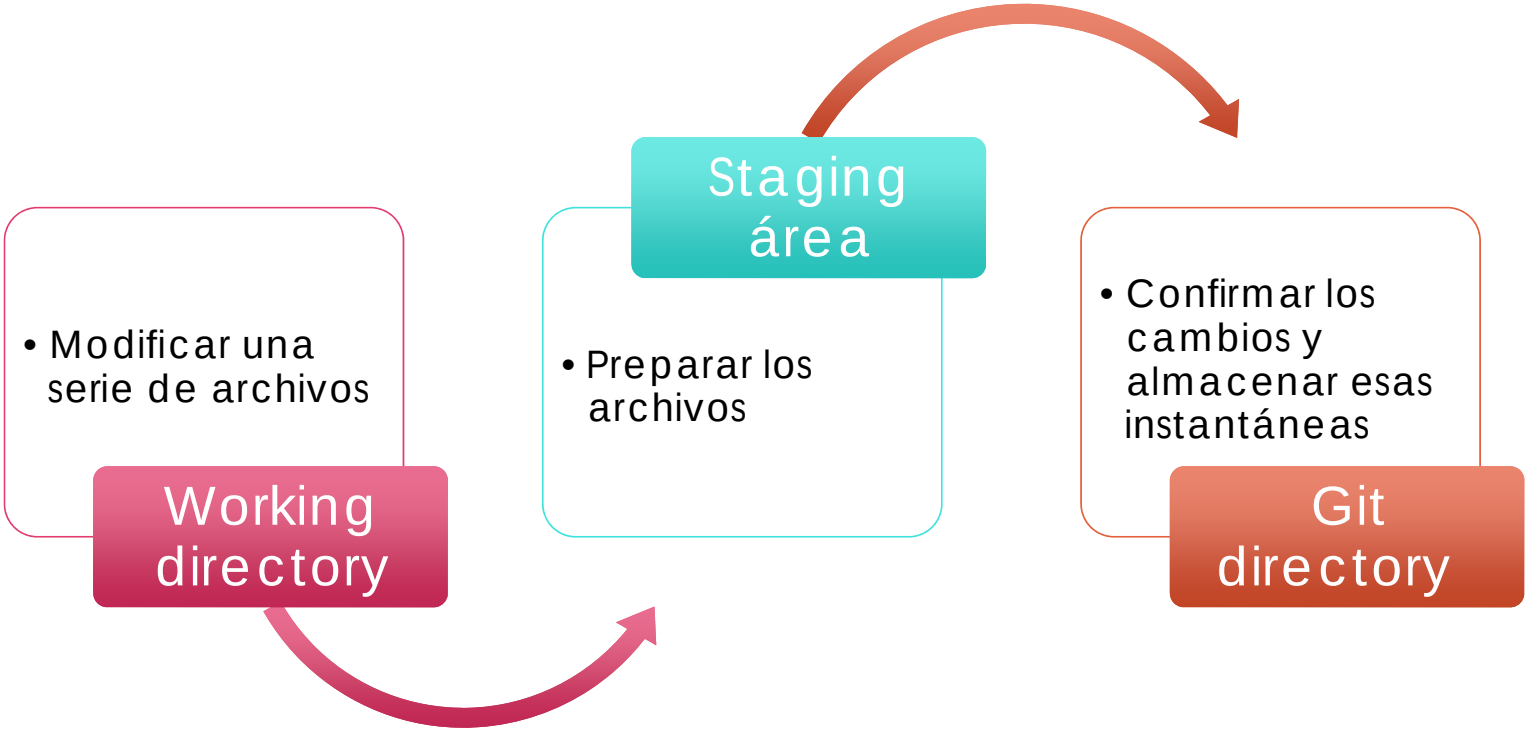

hotFix
develop
Modificacion del hotfix
Merge branch 'feature2' into develop
Merge branch 'feature4' into develop
Este es el commit 1 del branch feature4
Merge branch 'feature1' into develop
Este es el commit 2 del branch feature1
Este es el commit 2 del branch feature3
Este es el commit 1 del branch feature3
Este es el commit numero 2 del branch feature2
Este es el commit 1 del branch feature2
Este es el commit numero 1 de feature1
Branch develop
Este es el primer commit


Problemática a resolver

Inicio – Configuración de Git

```
git config --global user.name "John Doe"
git config --global user.email correoEjemplo@example.com
git config --list
git config --global -l
```

Flujo de trabajo

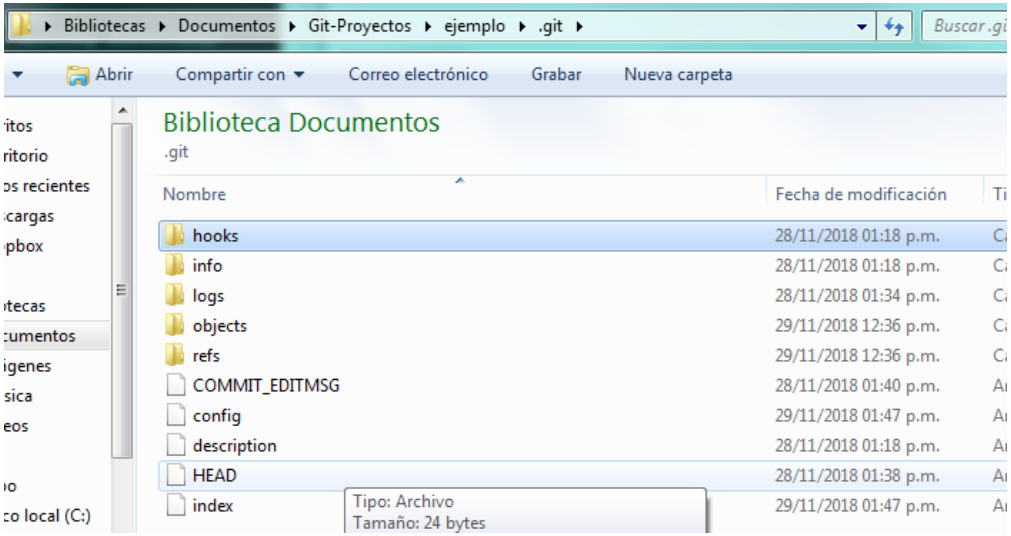


Flujo de trabajo

init

Iniciando un nuevo repositorio en blanco o reinicializando alguno existente

```
git init
```



add

Con el siguiente comando vamos a ingresar los archivos modificados en el directorio de trabajo al área de preparación

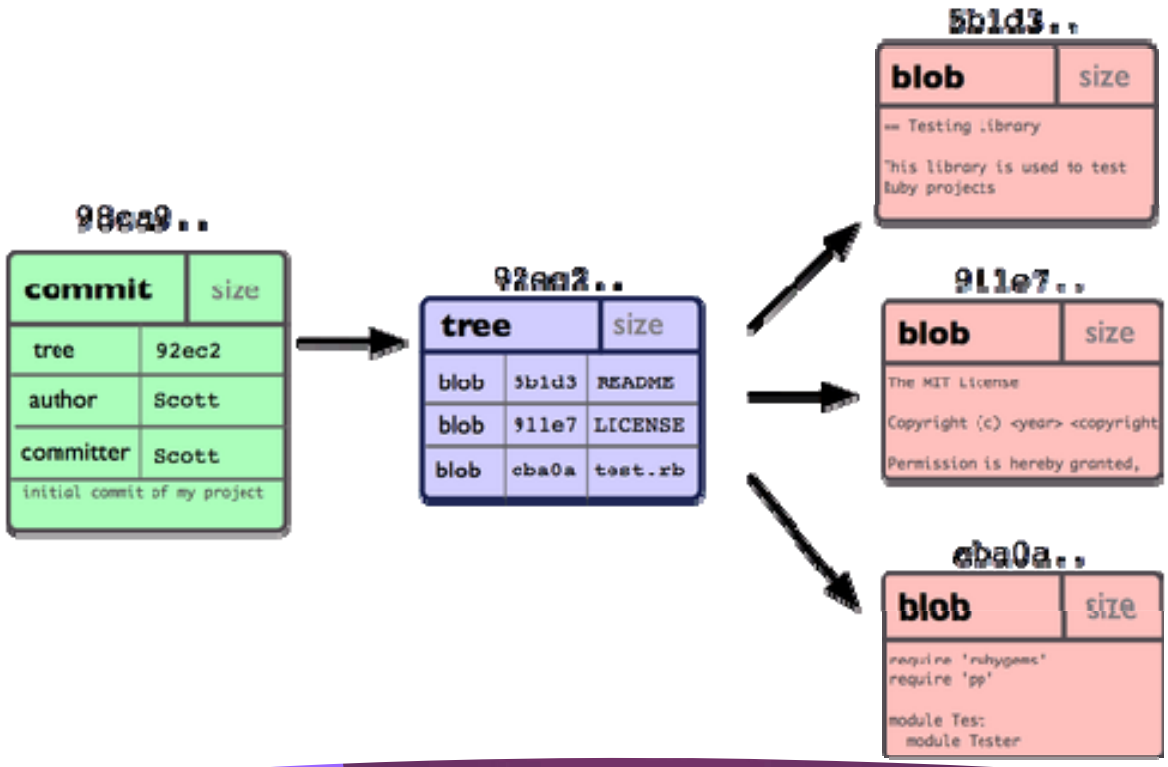
```
git add .
```

git commit

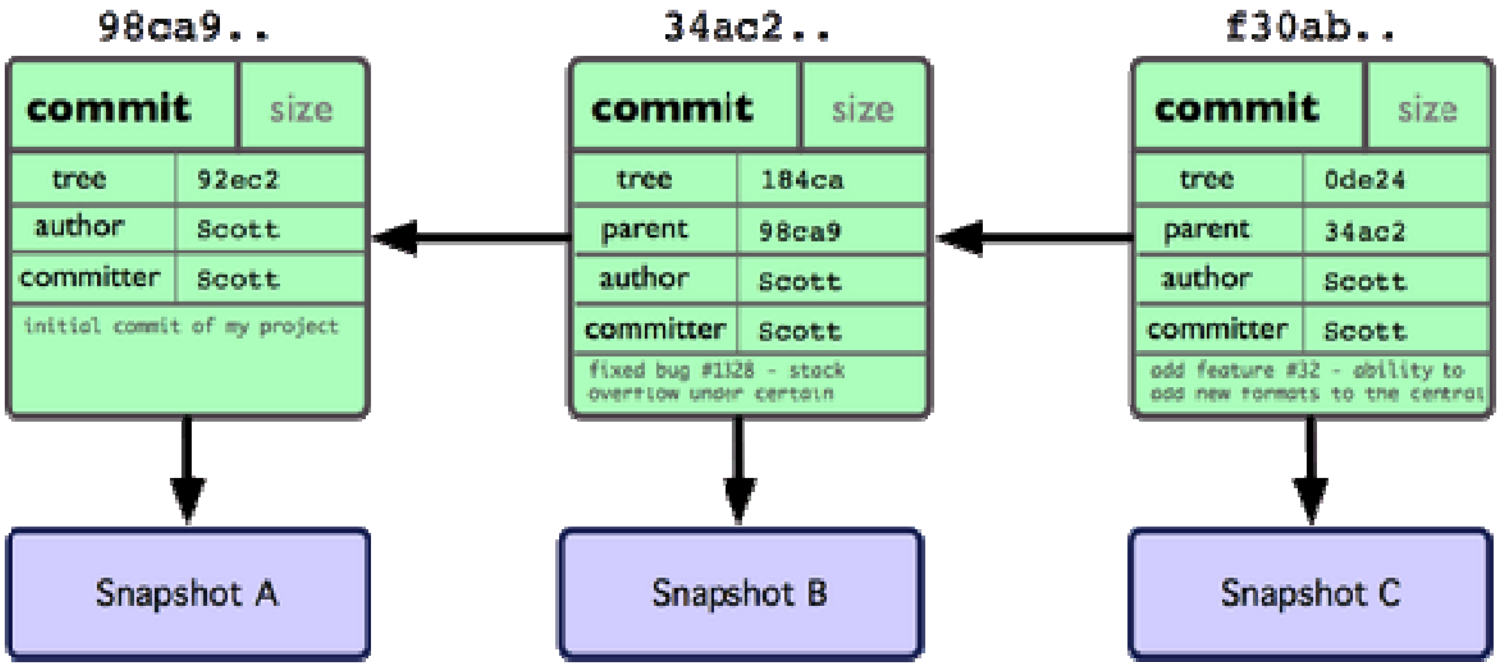
- Git almacena una instantánea de tu trabajo preparado.

Instantánea

- Contiene:
  - Metadatos
  - Autor
  - Mensaje explicativo
  - Apuntadores a confirmaciones que sean padres directos o múltiples (merge – dos o más ramas)



Fuente: https://git-scm.com/figures/18333fig0301-tn.png



Fuente: https://git-scm.com/figures/18333fig0302-tn.png

Revertir commit

Ubicar el nombre de la rama o ubicar el SHA1 del commit:

git log --oneline

Una vez ubicado agregamos el comando:

git revert OPCIONES

OPCIONES = HEAD / Branch_name / SHA1 commit

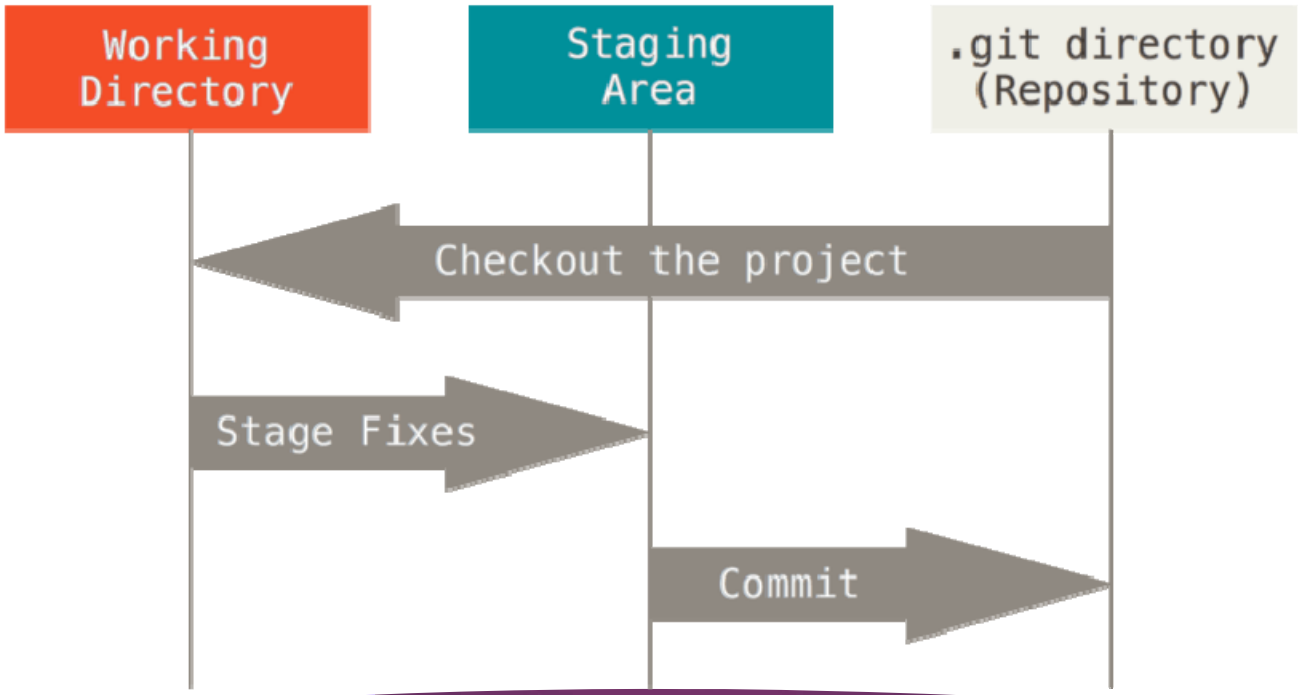
merge

En este paso hay dos ramas a unir, ponemos un ejemplo, se desea integrar la rama **hotFix** a la rama **master**, por lo que primero nos ubicamos en la rama master:

git checkout master

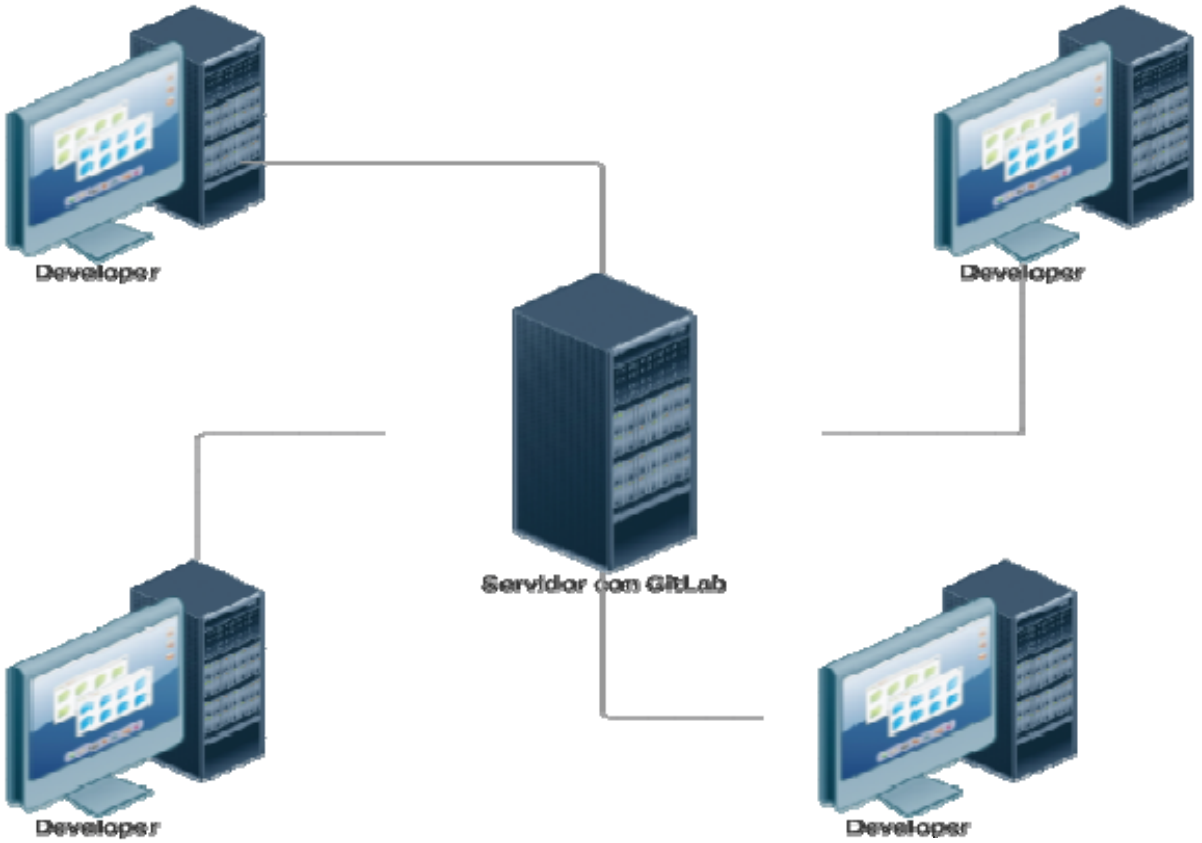
Después ejecutamos el comando

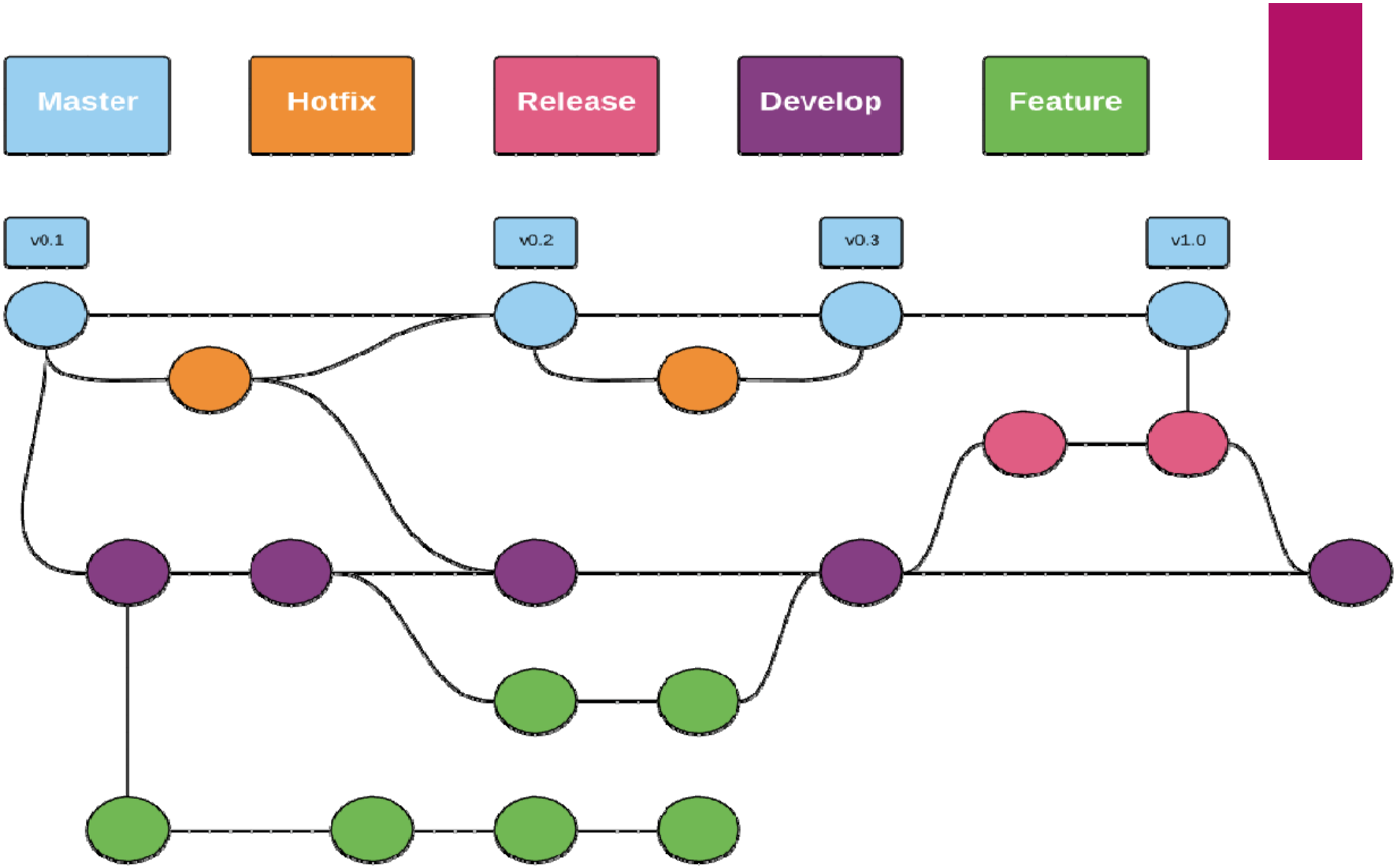
git merge hotFix



Directorio de trabajo, área de almacenamiento, y el directorio Git.

Fuente: https://git-scm.com/book/en/v2/images/areas.png


Developer
Developer
Servidor com GitLab
Developer
Developer


Master
Hotfix
Release
Develop
Feature
v0.1
v0.2
v0.3
v1.0

Revertir commit

git revert --no-commit HEAD

git revert --no-commit HEAD~1

git revert --continue

Revertir merge

tag

Creación de etiquetas

git tag nombre_del_tag

Listado de etiquetas:

git tag -l 'v1.4.2.*'

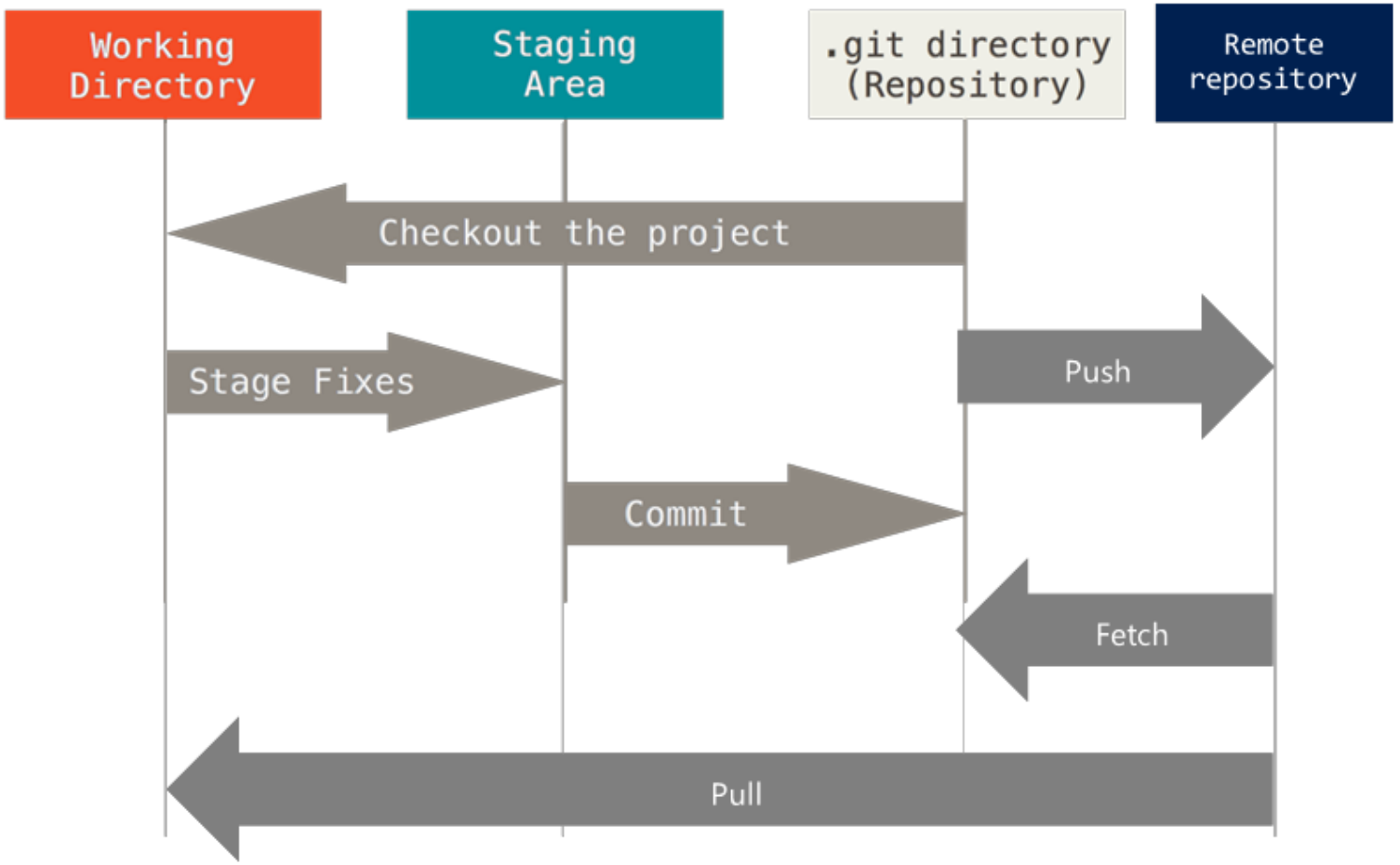
Creación de una etiqueta con alguna anotación:

git tag -a v1.0.0 -m “versión para utilizar”

Termina el primer modulo

SEGUNDO MÓDULO

GITLAB


Working Directory
Staging Area
.git directory (Repository)
Remote repository
Checkout the project
Stage Fixes
Push
Commit
Fetch
Pull

clone

git clone URL_REPOSITORIO

push

git push --set-upstream NAME_REPO master

git push NAME_REPO BRANCH_NAME

git push REMOTE --all

git push REMOTE --tags

pull

git checkout branch_name

git pull URL_REPO o NOMBRE_REMOTE nombre_branch

Ej:. git pull http://mirepo.com develop

Termina el segundo modulo

Buenas prácticas

Cada desarrollador o equipo de desarrollo puede hacer uso de Git de la forma que le parezca mas conveniente. Sin embargo una buena práctica es la siguiente:

Se deben utilizar 4 tipos de ramas: Master, Development, Features, y Hotfix.

Master:

Es la rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.

Development:

Es una rama sacada de master. Es la rama de integración, todas las nuevas funcionalidades se deben integrar en esta rama. Luego que se realice la integración y se corrijan los errores (en caso de haber alguno), es decir que la rama se encuentre estable, se puede hacer un merge de development sobre la rama master.

Features:

Cada nueva funcionalidad se debe realizar en una rama nueva, específica para esa funcionalidad. Estas se deben sacar de development. Una vez que la funcionalidad esté desarrollada, se hace un merge de la rama sobre development, donde se integrará con las demás funcionalidades.

Hotfix:

Son bugs que surgen en producción, por lo que se deben arreglar y publicar de forma urgente. Es por ello, que son ramas sacadas de master. Una vez corregido el error, se debe hacer un merge de la rama sobre master. Al final, para que no quede desactualizada, se debe realizar el merge de master sobre development.

Referencias

Scott Chacon, Jason Long. 2018. Git Una breve historia de Git. Site is open source and maintained by members of the Git community: *https://git-scm.com*. https://git-scm.com/book/es/v1/Empezando-Una-breve-historia-de-Git.

Scott Chacon, Jason Long. 2018. Git Una breve historia de Git. Site is open source and maintained by members of the Git community: *https://git-scm.com*. https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git.

Apellido, A. A. (Fecha). Título de la página. Lugar de publicación: *Nombre de la página web*. dirección de donde se extrajo el documento (URL).

Apellido, A. A. (Fecha). Título de la página. Lugar de publicación: *Nombre de la página web*. dirección de donde se extrajo el documento (URL).

Caso de dudas:
miguelangel@dgp.unam.mx