



El futuro digital
es de todos

MinTIC



Vigilada Mineducación

CICLO IV:

Desarrollo de Aplicaciones Web

Mision
TIC2022



El futuro digital
es de todos

MinTIC

UN UNIVERSIDAD
DEL NORTE

Vigilada Mineducación

Sesión 04:

Desarrollo de Aplicaciones Web

Control de versiones con GIT

Mision
TIC2022



Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Realizar el control de versiones de una aplicación web.
2. Utilizar instrucciones avanzadas de GIT para desarrollo de aplicaciones web colaborativo.



GIT

- Git es un software especializado en el seguimiento y control de archivos el cual se enfoca principalmente en el versionamiento de proyectos de desarrollo de software.
- Fue Ideado por Linus Torvalds, el creador del sistema operativo Linux.
- Es una línea de comandos (CLI), la cual nos permite definir un repositorio y agregar archivos con el fin de hacer seguimiento de los cambios realizados sobre los mismos.
- Hay organizaciones orientadas a git como lo serian Github, Gitlab, Bitbucket, entre otros. Estos son entornos en la nube que se encargan de almacenar repositorios de git ya sea de forma pública o privada.



Imagen tomada de [git](#)

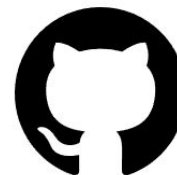


Imagen tomada de [github](#)



GIT

- En git se manejan tres estados:
 - Directorio de trabajo.
 - Stage.
 - Repositorio.
- El flujo de trabajo de git consiste en:
 - Estar pendiente a los cambios de los archivos en el directorio de trabajo.
 - Luego, un desarrollador se encarga de agregarlos al area de stage.
 - Finalmente, hacer un commit al repositorio con estos cambios.
- Cabe resaltar que hay repositorios locales y remotos (en la nube).

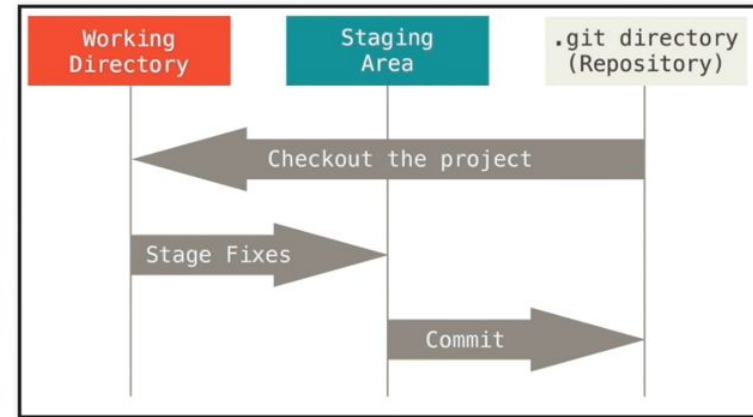


Imagen tomada de [Medium/@cassandra-cryptoassets](https://medium.com/@cassandra-cryptoassets)



GIT - Comandos Básicos

- **init:** Tiene como fin crear un repositorio de git con una branch master o main, acorde la configuración de la instalación de git.
- **status:** Se encarga de listar los cambios en el directorio de trabajo que no han sido agregados a al area de stage o que están listos para ser enviados a un repositorio mediante un commit.
- **add:** Se encarga de agregar cambios, así mismo como archivos nuevos a los que se les debe hacer seguimiento, del directorio de trabajo al area de stage.
- **rm:** Se encarga de eliminar archivos tanto del directorio de trabajo, así mismo como del area de stage.



GIT - Comandos Básicos

- **checkout:** Este comando nos permite bajo una misma rama deshacer cambios del directorio de trabajo. Por otro lado este mismo de comando nos permite cambiar de rama.
- **restore:** Este comando nos permite deshacer cambios del area de stage hacia el directorio de trabajo.
- **commit:** Este comando es el que se encarga de subir los cambios del area de stage hacia nuestro repositorio local.
- **push:** Este comando se encarga de sincronizar los cambios de nuestro repositorio local con el repositorio remoto.
- **pull:** Este comando se encarga de sincronizar nuestro repositorio local con los cambios realizados en el repositorio remoto.



GIT - Branch

- Git nos permite dividir nuestro trabajos en ramas a partir de una rama origen.
- Esto nos permite tener un flujo ordenado de trabajo para enfocarnos en entornos de desarrollo por rama.
- Eventualmente se llegan a tener ramas dedicadas a un único feature o característica a desarrollar.

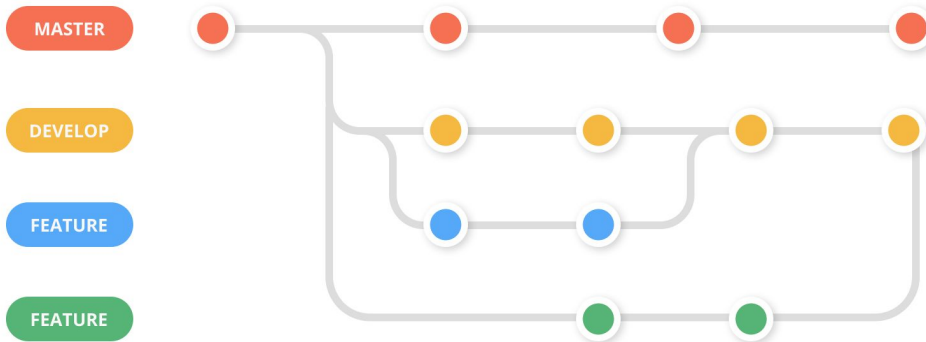


Imagen tomada de [Zepel](#)



GIT - Merge

- Git Merge consiste en combinar los cambios presentados por dos ramas diferentes.
- Supongamos que queremos combinar el feature A (el corbatín verde), y feature B (la apariencia parda del gato).
- Entonces para poder combinar ambos features realizamos un merge para así incorporar ambos cambios.

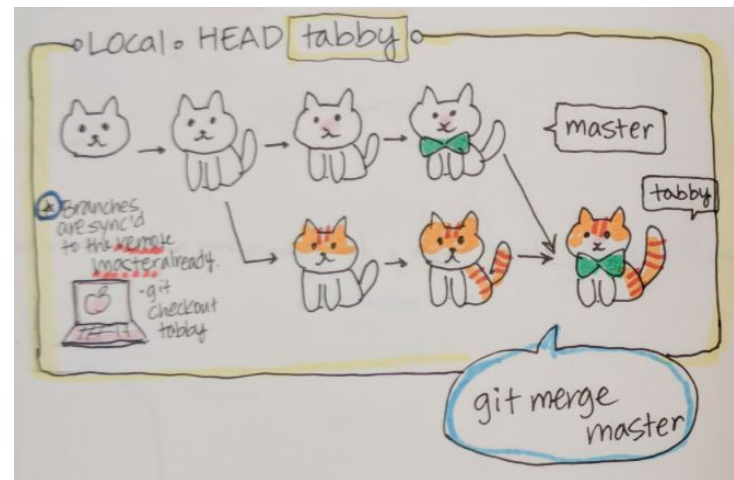


Imagen tomada de [git-purr](#)



GIT - Pull Request (PR)

- Es un proceso relacionado a git que ocurre en un proveedor de git en la nube como lo suele ser Github, Gitlab, etc.
- Consiste en introducir los cambios de una rama a otra a través de una página web.
- Las personas relacionadas en el proyecto pueden revisar los cambios a ser introducidos.
- Una vez todo el equipo esté de acuerdo, el PR finaliza y los cambios se incorporan al codebase o la rama objetivo.
- Se considera una buena práctica porque mantiene informado al equipo con respecto a cambios en el repositorio.

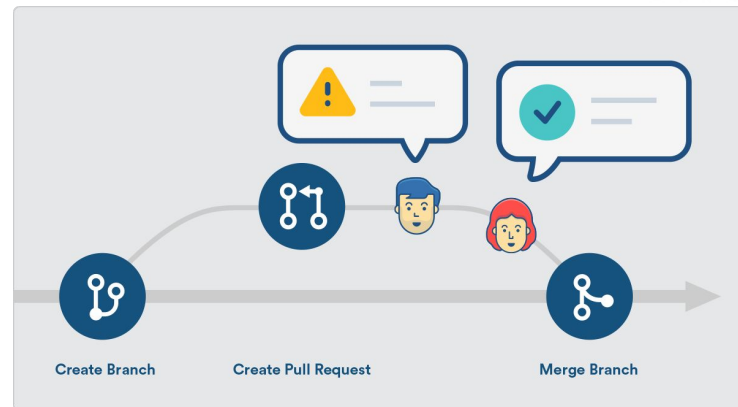


Imagen tomada de [Atlassian](#)



GIT - Comandos



Git Cheat Sheet



Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit] : [file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/>

for official GitHub training.

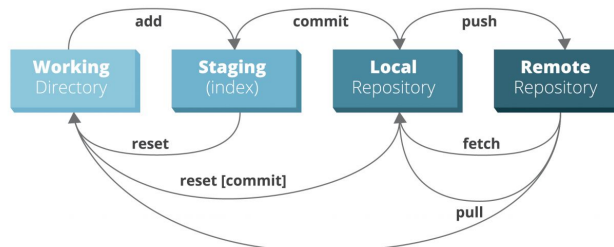


Imagen tomada de
[BlogCode/gregorgonzalez](https://blogcode.com/gregorgonzalez)



GIT - CI/CD

- Git ofrece una integración de flujo de trabajo donde se pueden ejecutar acciones después de ciertos eventos como lo sería el caso de realizar un Pull Request a una rama en específico.
- Estos eventos se conocen como Integración Continua (CI) y Desarrollo Continuo (CD).
- **CI:** Fase en la cual se realizan pruebas, o se le da formato a nuestro código para validar su calidad.
- **CD:** Fase en la cual se ejecutan scripts de descarga, instalación y despliegue a un entorno en específico.

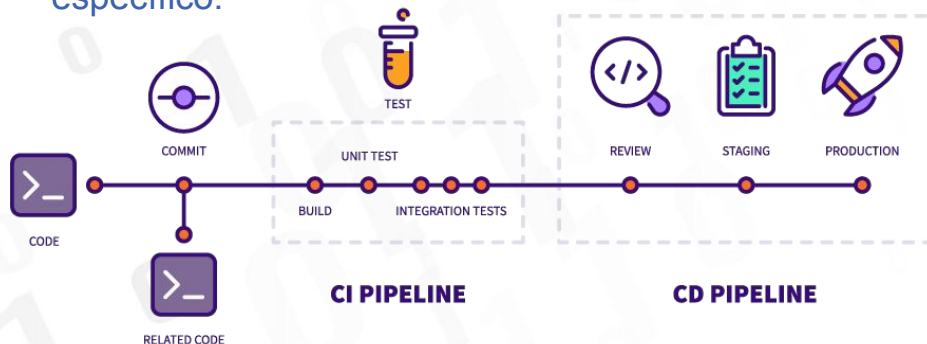


Imagen tomada de
[GitLab](#)



El futuro digital
es de todos

MinTIC

UN UNIVERSIDAD
DEL NORTE

Vigilada Mineducación

Ejercicios de práctica

Mision
TIC2022



Referencias

- <https://medium.com/chaya-thilakumara/an-introduction-to-git-for-beginners-c97e701cecf9>
- <https://medium.com/cassandra-cryptoassets/git-basics-a-step-by-step-tutorial-c3098934fa95>
- <https://girliemac.com/blog/2017/12/26/git-purr/>
- <https://about.gitlab.com/>
- <https://github.com/>
- <https://bitbucket.org/>



El futuro digital
es de todos

MinTIC

UN UNIVERSIDAD
DEL NORTE

Vigilada Mineducación

¡GRACIAS
POR SER PARTE DE
ESTA EXPERIENCIA
DE APRENDIZAJE!



Misión
TIC 2022