Clase 23

git + GitHub

¿Qué es?

Es un sistema que ayuda a organizar el código, el historial y su evolución, funciona como una máquina del tiempo que permite navegar a diferentes versiones del proyecto y si queremos agregar una funcionalidad nueva nos permite crear una rama (branch) para dejar intacta la versión estable y crear un ambiente de trabajo en el cual podemos trabajar en nueva funcionalidad sin afectar el original.

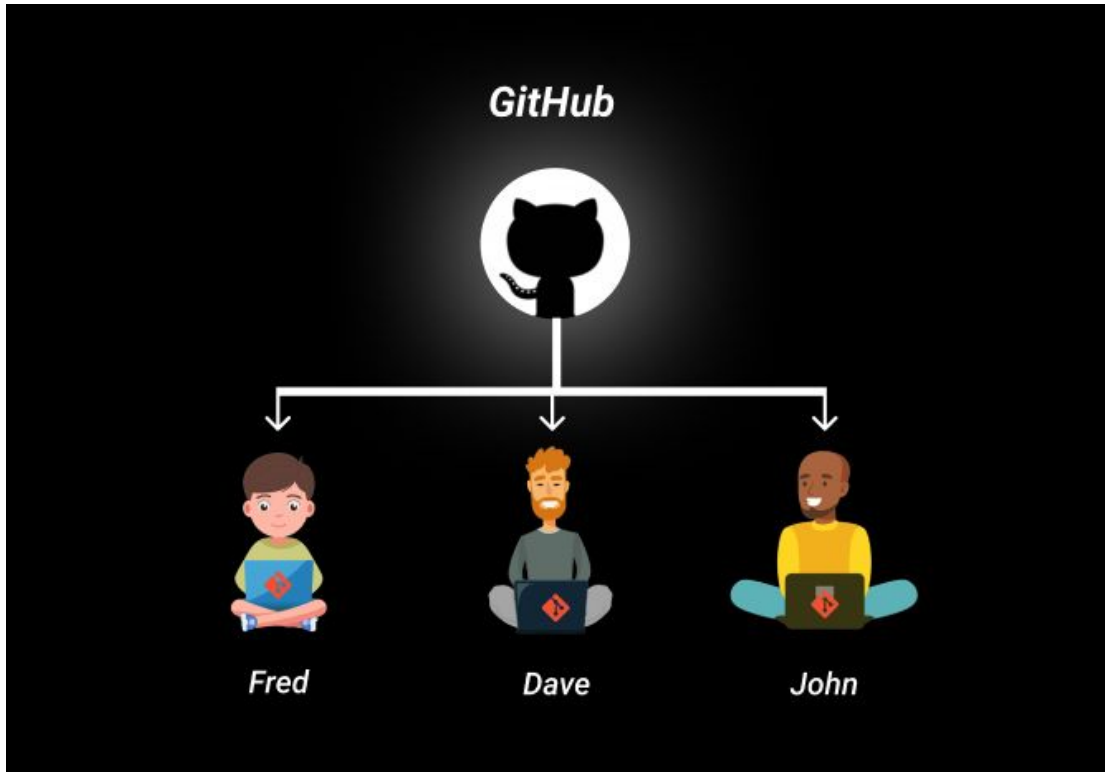
Tutorial

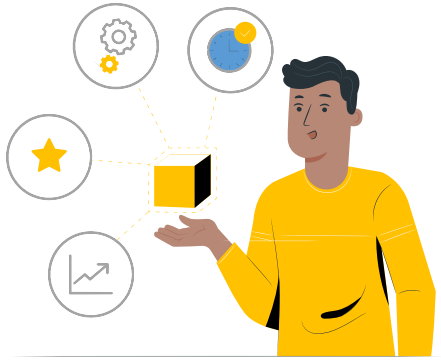
Videos del profesor Alejandro Zapata de Codo a Codo:
https://www.youtube.com/watch?v=ptXiQwE535s&list=PLoCpUTIZIYORkDzYwdunkVf-KIqGjyoot

Video de Udemy de Noelia Silva Docente de Codo a Codo:

https://www.udemy.com/course/introduccion-git-github/learn/lecture/18156802?components=purchase%2Ccacheable_buy_button%2Cbuy_button%2Crecommendation#overview

GIT!





- Control de versiones distribuido.
- Manejar distintas versiones del proyecto.
- Guardar el historial o se guardan todas las versiones de todos los archivos del proyecto.
- Trabajar simultáneamente sobre un proyecto

TUTORIAL: https://try.github.io/levels/1/challenges/1

¿Cómo funciona?

Git almacena instantáneas de un mini sistema de archivos, cada vez que confirmamos un cambio lo que git hace es tomar una “foto” al aspecto del proyecto en ese momento y crea una referencia a esa instantánea, si un archivo no cambió git no almacena el nuevo archivo sino que crea un enlace a la imagen anterior idéntica que ya tiene almacenada.

Instalación

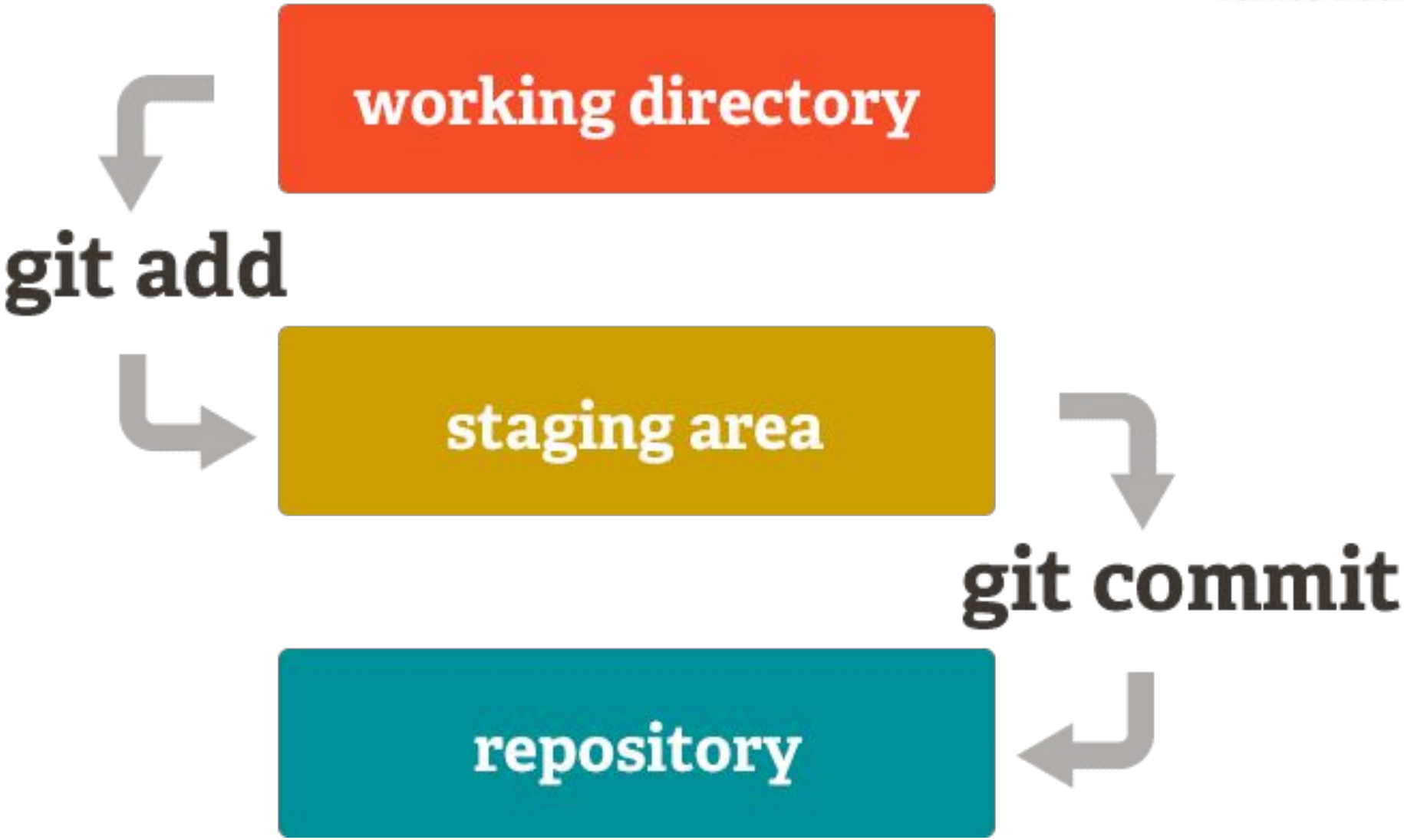
- ✔ Descargarlo **acá**

Una vez descargado, emplearemos la interfaz de línea de comando de tu sistema operativo para interactuar con GIT:

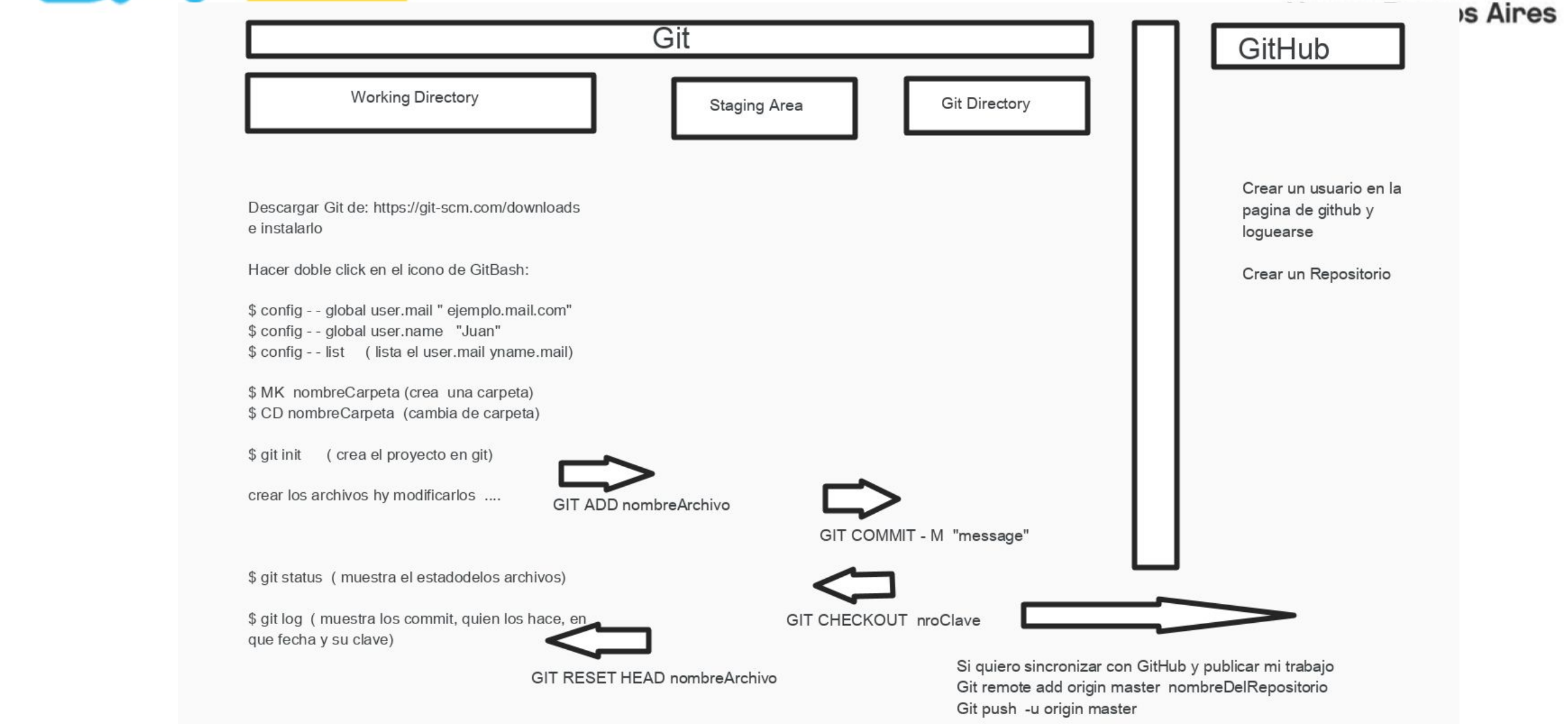
- ✔ En **Windows**: abrir la aplicación Git Bash que se instaló junto con GIT.
- ✔ En **Mac**: abrir la terminal mediante el finder.
- ✔ En **Linux**: abrir la consola bash.

Para verificar si está instalado, podemos ejecutar el comando **git --version**. Si obtenemos respuesta nos indicará la versión de GIT que tenemos instalada. En caso de que no poder, ver las instrucciones **acá**.

Estados

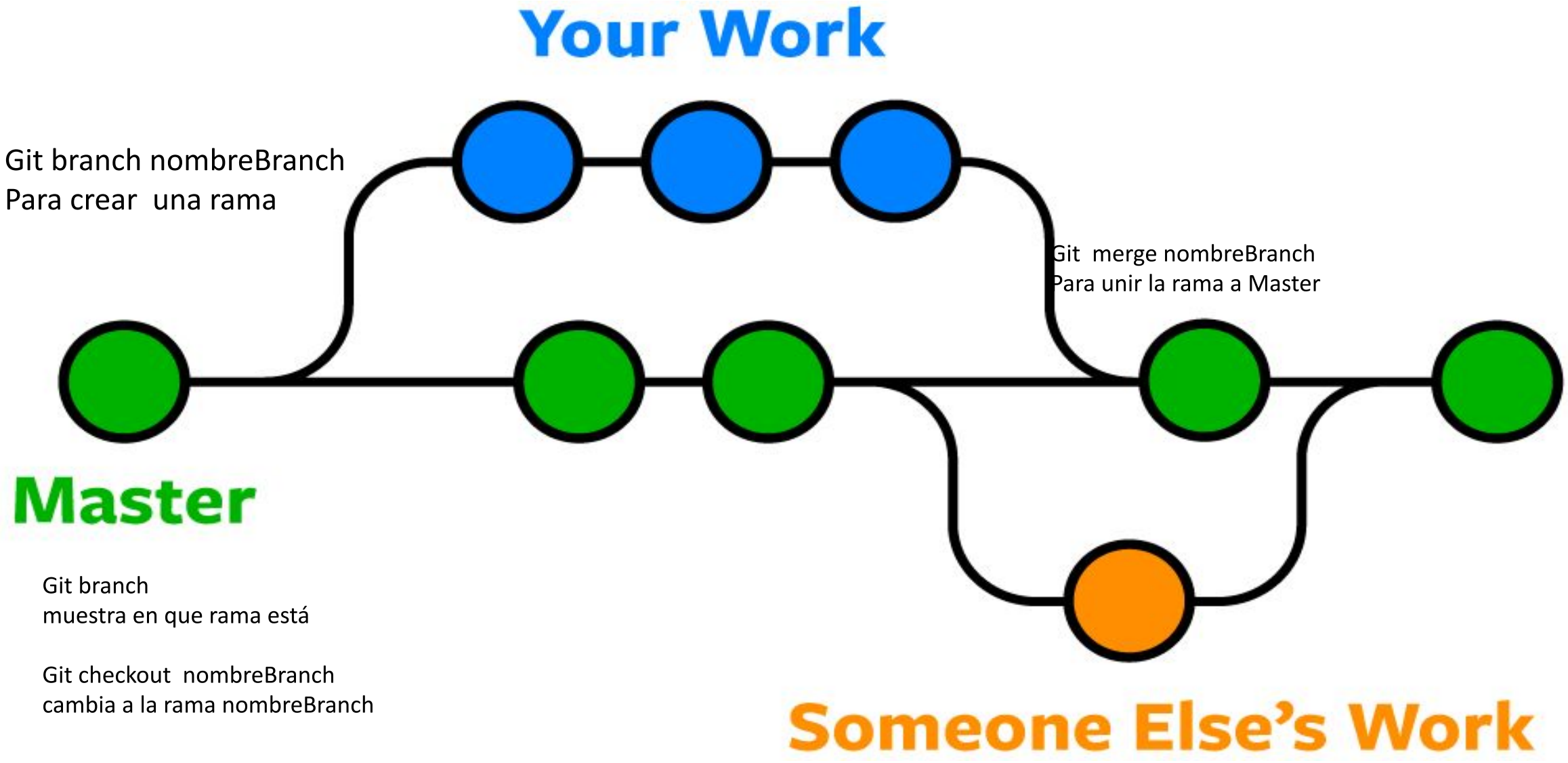

working directory
git add
staging area
git commit
repository

Estados


Git
Working Directory
Staging Area
Git Directory
GitHub
Descargar Git de: https://git-scm.com/downloads
e instalarlo
Hacer doble click en el icono de GitBash:
$ config - - global user.mail " ejemplo.mail.com"
$ config - - global user.name "Juan"
$ config - - list (lista el user.mail yname.mail)
$ MK nombreCarpeta (crea una carpeta)
$ CD nombreCarpeta (cambia de carpeta)
$ git init (crea el proyecto en git)
crear los archivos hy modificarlos ....
GIT ADD nombreArchivo
GIT COMMIT - M "message"
$ git status (muestra el estadodelos archivos)
$ git log (muestra los commit, quien los hace, en
que fecha y su clave)
GIT RESET HEAD nombreArchivo
GIT CHECKOUT nroClave
Crear un usuario en la
pagina de github y
loguearse
Crear un Repositorio
Si quiero sincronizar con GitHub y publicar mi trabajo
Git remote add origin master nombreDelRepositorio
Git push -u origin master

Branch (ramas)

Vamos Buenos Aires



Git branch nombreBranch
Para crear una rama

Git merge nombreBranch
Para unir la rama a Master

Git branch
muestra en que rama está

Git checkout nombreBranch
cambia a la rama nombreBranch

Empecemos

Una vez realizada la configuración básica, hay que utilizar la línea de comando para ubicarnos en la carpeta que queremos hacer control de versión.
Ejemplo: cd /Users/aylen/Desktop/proyecto y una vez parados en esa carpeta, inicializamos git mediante el comando **git init** que también va a crear un archivo oculto llamado .git que se va a encargar de llevar el control de todas las modificaciones que se hagan en esa carpeta.

Empecemos

Con el comando **git status** se van a mostrar en rojo todos los archivos que git sabe que existen en la carpeta pero que todavía no están registrados. Para que esos cambios queden registrados tenemos que añadirlos con el comando **git add .** el punto indica que queremos añadir todos los archivos y si ahora hacemos **git status** todos los archivos van a aparecer en verde.

Terminología

Repositorio: Es la carpeta principal donde se encuentran almacenados los archivos que componen el proyecto. El directorio contiene metadatos gestionados por Git, de manera que el proyecto es configurado como un repositorio local.

Terminología

Commit: Un commit es el estado de un proyecto en un determinado momento de la historia del mismo, imaginemos esto como punto por punto cada uno de los cambios que van pasando. Depende de nosotros determinar cuántos y cuales archivos incluirá cada commit.

Terminología

Rama (branch): Una rama es una línea alterna del tiempo, en la historia de nuestro repositorio. Funciona para crear features, arreglar bugs, experimentar, sin afectar la versión estable o principal del proyecto. La rama principal por defecto es **master**.

Terminología

Pull Request: En proyectos con un equipo de trabajo, cada persona puede trabajar en una rama distinta pero llegado el momento puede pasar que dicha rama se tenga que unir a la rama principal, para eso se crea un pull request donde comunicas el código que incluye tu cambio y usualmente revisan tu código, se agregan comentarios y por último lo aprueban para darle merge. En el contexto de GIT, merge significa unir dos trabajos, en este caso tu branch con master.

GitLab vs GitHub

Investigar las diferencias :https://www.redeszone.net/2019/01/10/github-vs-gitlab-diferencias/