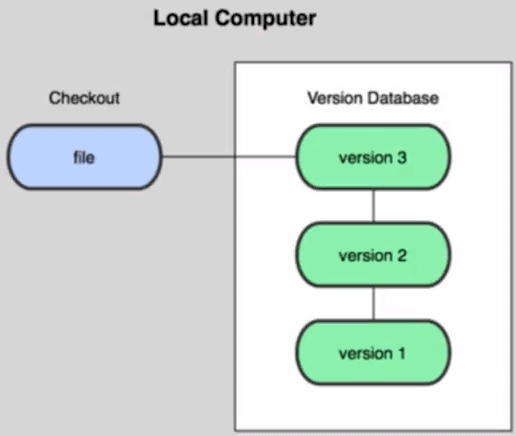
**SISTEMA DE CONTROL DE VERSIONES**

Un sistema que registra los cambios de un archivo al largo del tiempo.

Tipos

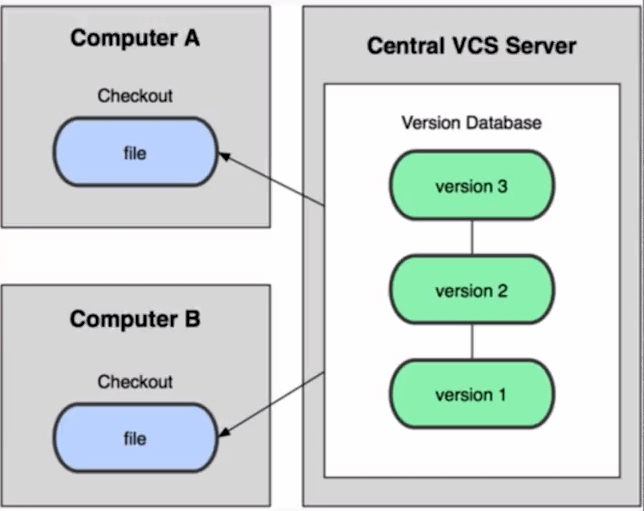
* Local computer (ejm: ctrl + z)

los cambios se encuentran en la misma computadora.



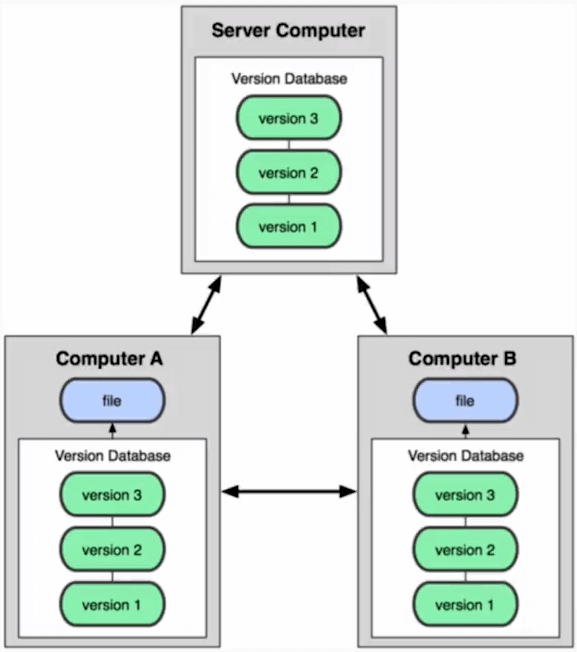
* Centralizada

No depende del computador ya que los cambios son almacenados en un súper servidor. (de forma jerárquica)



* Distribuida

Los cambios de un archivo se almacenan tanto en un súper servidor como también en el computador, en caso que se pierda la inf. Se puede recuperar ya que está almacenado en diferentes partes.



**GIT**

Git es un sistema de control de versiones, es el conjunto de herramientas que tenemos disponibles para poder gestionar de una forma eficiente las distintas versiones de nuestros proyectos.

Los 3 estados de GIT:



* WORKING DIRECTORY: Área donde trabajas tus archivos.
* STAGING AREA: Área de preparación de los archivos para ser registrados en el repositorio.
* GIT DIRECTORY(repositorio): Área donde se almacena los archivos trabajados.

**CONFIGURACION DE GIT**

* Configurar el email para GIT

**git config --global user.email saldivar.rednaxela@gmail.com**

* Nombre

**git config –global user.name “Alexander Saldivar”**

**git config –global user.name Alexander**

**COMANDOS DE GIT**

* Cd
* Ls: listar
* clear (cls): limpiar consola
* mkdir: crear carpeta
* rm: eliminar archivo
* rm –rf: eliminar carpeta
* touch: crear archive
* vim : editar archive
* :wq = salir y guarder
* :q! = salir sin guardar cambios

**CREAR UN REPOSITORIO**

**git init nombre\_del\_repositorio** crear repositorio

**git init**  (segunda opción)

**AGREGANDO, QUITANDO Y VIENDO EL STATUS**

**git status** Los archivos marcados en rojo están en **WORKING DIRECTORY** (Muestra a los archivos nuevos o modificados)

**git add <file>** Los archivos fueron enviados al **STAGING AREA** listos para realizar un commit.

**git rm –cached <file>** Para retirar el archivo de **STAGING AREA** a **WORKING DIRECTORY**

**git add –A** Agregar todos los archivos a **STAGING AREA**

**git add .** Agregar todos los archivos creados y modificados, pero no lo eliminados

**git add –u** Agrega todos los archivos modificados y eliminados, pero no los creados

**git add –n <file>** comprobar la existencia del archivo

**COMMIT (confirmar cambios)**

**git commit –m “mensaje”** Agrega todos los archivos de **STARGING AREA** a **GIT DIRECTORY.**

**git commit –am “mensaje”** cumple lo mismo de (git add –A + git commit –m “msj”)

**git commit --amend** concatenar los nuevos cambios con los cambios anteriores

(agrupandolos los cambios en un solo commit)

**git log** mostrar el historial de commit’s

**ETIQUETANDO CONFIRMACIONES (GIT TAG)**

**git tag agregar etiquetas**

**-a** para la anotación

**-m**  para el mensaje

**-l** listar las etiquetas

**-f** para renombrar

**-d** para borrar

Ejemplos:

**git tag –a 0.5 –m “mensaje”**

**git tag 0.3 <codigo del commit>**

**git tag –d <numero (0.3)>**

**git tag –f –a 0.1 –m “mensaje”**

**GIT LOG (listar los commit)**

**git log**

**git log –oneline**

**git log –oneline –graph**

**git log -<numero de orden de commit>**

**GIT DIFF (Comparar los cambios realizados)**

**Git diff <version 1> <version 2>**

**GIT RESET (eliminar commit)**

Este commando elimina los anteriores commit del commit seleccionado, los archivos dentro de estos commits los manda a **STARGING AREA.**

**git reset –soft <cod. de commit>**

Este commando elimina los anteriores commits del commit seleccionado, los archivos dentro de estos commits los manda a **WORKING DIRECTORY.**

**git reset –mixed <cod. de commit>**

Este commando elimina todos los archivos que esten e el STARGING AREA

**git reset –hard**

Este commando el peligroso, elimina los anteriores commit del commit seleccionado y tambien los archivos dentro del area de **STARGING AREA**, los archivos dentro de estos commits tambien son eliminado pero menos los archivos que estan en el area **WORKING DIRECTORY.**

**git reset –hard <cod. de commit>**

**NOTA:** Estos archivos pueden ser recuperados con el mismo comando pero con el ultimo <cod de commit>

**git reset –hard <ultimo cod commit>**

**GIT BRANCH (ramas)**

**git branch <nombre>** crear una rama

**git brach –l**  listar ramas

**git branch –d <nombre>** eliminar una rama

**git brach –D <nombre>** forzar eliminacion de la rama

**git branch –m <nombreAnterior> <nombreNuevo>** renombrar

**MOVIMIENTO ENTRE RAMAS Y VERSIONES (git checkout)**

**git checkout <nombreBranch>** accede a la rama

**git checkout <cod. commit>** nos permite ver el estado de esa version o commit

**git checkout –b <NuevoBranch>** crear una rama y accede al mismo tiempo

**MEXCLAR RAMAS Y RESOLVER CONFLICTOS (git merge)**

**git merge <nombreBranch>**

**GUARDAR CAMBIOS TEMPORALMENTE (git stash)**

**git stash** guardar temporalmente un cambio (archivos modificados o q esten en **STARGING AREA**).

**git stash list** listar los cambios temporales

**git stash drop stash@{numero}** eliminar lista seleccionada

**git stash apply stash@{numero}** aplicar el cambio temporal

**git stash apply** aplicar el ultimo cambio temporal

**ESCOGIENDO COMMIT SELECTIVAMENTE (cherry pick)**

**git cherry-pick <cod branch>** para extraer un commit de un branch a otro branch

**GITHUB**

Github es un servicio de alojamiento que ofrece a los desarrolladores repositorio de software usando el sistema de control de versiones de GIT (almacena los cambios de los proyectos).

Aspectos de Github:

* GitHub permite que alojemos proyectos en repositorios de forma gratuita, pero a la vista de todos.
* Permite que puedas compartir tus proyectos de una forma mucho más fácil.
* Es la opción perfecta para poder trabajar en equipo en un mismo proyecto.

**LLAVE SSH**

Las llaves ssh sirven para autentifica tu usuario y contraseñas cada vez que subas una modificación a tu repositorio.

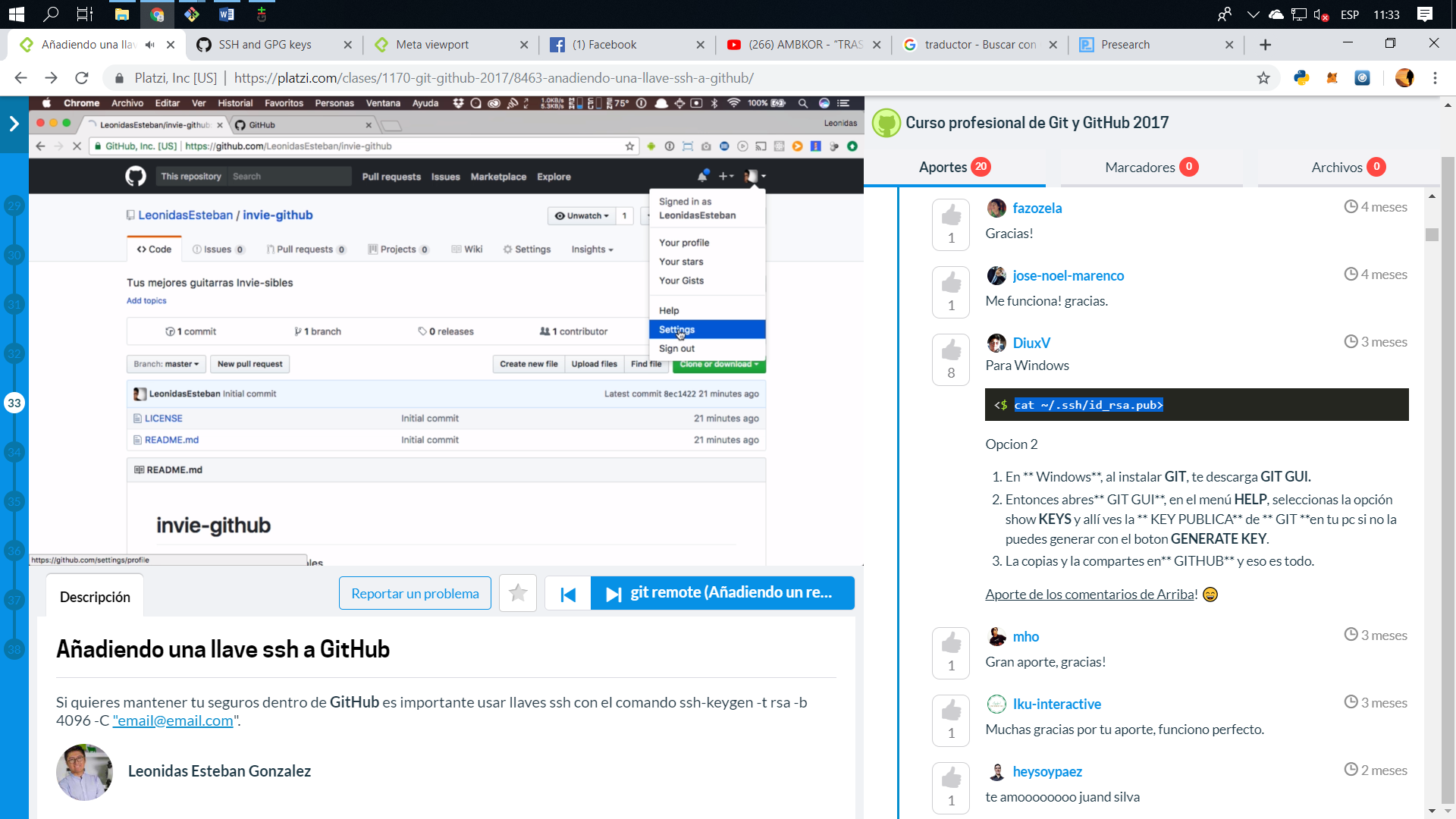
Para generar la llave ssh

**ssh-keygen -t rsa -b 4096 -C "saldivar.rednaxela@gmail.com"**

copiar la llave

**cat ~/.ssh/id\_rsa.pub**

en caso que no copie



Luego copiar la llave en github.

**AÑADIENDO UN REPOSITORIO REMOTO A UNO LOCAL (git remote)**

git remote add [origin] [url git] conectar un repositorio con nuestro local

git remote –v listar los repositorios conectados

git remote remove eliminar alguna conexión con algún repositorio.

**TRAYENDO CAMBIOS DESDE EL REPOSITORIO REMOTO (git pull / git fetch)**

**git fetch origin master** Traerá los archivos del repositorio remoto (del repositorio origin se traerá la rama maste

Para traer los cambios del repositorios remoto vamos a usar el comando git fetch. Este comando recibe dos parámetros 1. desde donde y hacia donde van.

**git merge origin/master** Vamos a mezclar la rama del repositorio remoto con la rama master del local

**git merge origin/master –allow-unrealated-histories**

**git pull origin master** Es la combinación de un fetch con un merge (realiza los 2 pasos)

**ENVIANDO CAMBIOS AL REPOSITORIO REMOTO (git push)**

**git push origin master subir los archivos de master**

**git push origin master –tags subir los tags agregados en el GIT**

**git push origin <nombreBranch> subir el branch nombrado al GITHUB**

**GITIGNORE**

Ignorar archivos no deseados, no subir archivos

**touch .gitignore** crear un archivo <.gitignore> y dentro de este archivo poner los nombres de los archivos que queremos ingnorar.