1. Descarga de Git

<https://git-scm.com/download/win>

1. Creamos nuestra cuenta en git
2. Asociamos nuestra cuenta y correo electrónico

 $git config --global user.name "Miguel Nieva"

 $git config --global user.email "m@platzi.com"

1. Comprobamos que se asocien correctamente

$git config --list

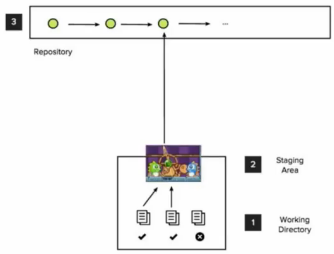
user.name=Miguel Nieva

user.email=m@platzi.com

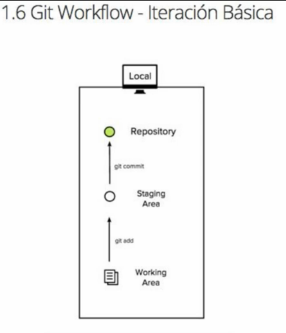
1. GIT El sistema de control de versiones

$ git config --list

Arquitectura de Arbol



Iteración Basica



**Ejercicio**

<https://github.com/platzi/platzigit>

* $ ls: listar contenido de la carpeta
* $ cd: navegar entre carpetas
* $ git init: iniciar repositorio, cada vez que cambiemos de proyecto hay que iniciar
* $ git status: en donde me encuentro, muestra cambios
* $ git add: Agrega archivos al staging área
  + $ git add –A: Agrega todos los archivos
* $ git commit –m “mensaje del commit”: manda todos los archivos al repositorio –m “mensaje”
* $ git commit –am “mensaje del commit”: agiliza el proceso pasando por add y el del mensaje
* git commit -m "Nombre del commit" --amend: Sustituye el ultimo commit
* $ git log: log de cambios.
* $ git --oneline: muestra solo el mensaje del commit y el commit, en una sola línea
* $ git log --decorate: Te muestra donde se encuentra el head point en el log
* $ git log --stat: Explica el numero de lineas que se cambiaron brevemente
* $ git log -p: Explica el numero de lineas que se cambiaron y te muestra que se cambio en el contenido
* $ git shortlog: Indica que commits a realizado un usuario, mostrando el usuario y el titulo de sus commits.
* $git log pretty=format:"%cn hizo un commit %h el dia %cd" : Muestar mensajes perzonalizados de los commits
* $ git log --graph --oneline --decorate: muestra en línea de tiempo
* $ git log --after="2016-1-2"
* $ git log --before="today"
* $ git log --grep="FOOTER": Busca los commits que cumplan tal cual esta escrito entre las comillas.
* $ git log -- index.html: Busca los commits en un archivo en especifico.
* $ git log -S"Por contenido": Buscar los commits con el contenido dentro del archivo, dentro del código.
* $ git config –global alias.nombredealias “comando”: le asigna un alias a un comando personalizado.

git config --global alias.superlog "log --graph --abbrev-commit --decorate --date=relative --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white) - %an%C(reset)%C(bold yellow)%d%C(reset)' --all"

* $ git reset --hard iddelultimocommit: borra a partir de un commit especificado, modifica desde el working área.
* $ git reset --mixed iddelcommit: Se iguala el staging área a un commit en específico, sin tocar el working área. listo para agregarlos en.
* $ git reset --soft iddelcommit: cambios solo en el repositorio. staging = working, ya listo para hacer un commit
* $ git checkout: explorar lo que hay en cada commit
* $ git checkout -b “nombredelarama”: Crea una rama
* $ touch nombredelarchvo.ext: Crea un archivo}
* $ git merge nombredelarama: fuciona una rama (ubicar el head sobre la rama que va a absorber)
  + Presionar o para que deje escribir
  + escribir el nombre de la fusión
  + escribir “esc” “:x”
  + Si existe un conflicto se tiene que editar manualmente
* git branch -d nombredelarama: borrar la rama, siempre y cuando haya sido fusionada

716f6e5 123

fad9078 2

27feb65 1

a3a1646 edicion de pagina de ingles a espa<C3><B1>ol

894bd25 Titulo y Descripcion agregados

668510a Creacion inicial del proyecto

* rm -rf .git (carpeta .git , si borramos la carpeta se **reinicia** todo el repositorio)

## Clonar un repositorio

* $git clone <https://github.com/github/gemoji.git>

SSH: Secure Shell que genera dos llaves

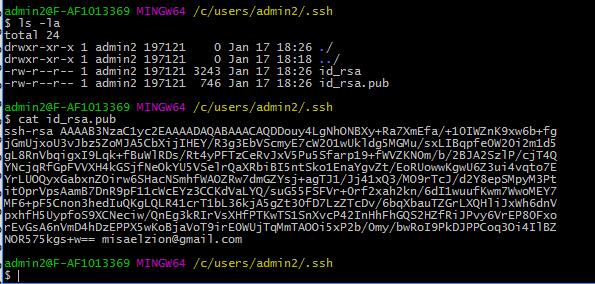
Llave publica:

Llave privada:

* Crear un repositorio público en la página github
* Crear llave ssh para el repositorio

$ ssh-keygen -t rsa -**b** 4096 -C "***misaelzion@gmail.com***"

y aquí se genera la clave en c/users/nombredeusuario/.ssh/llave.pub



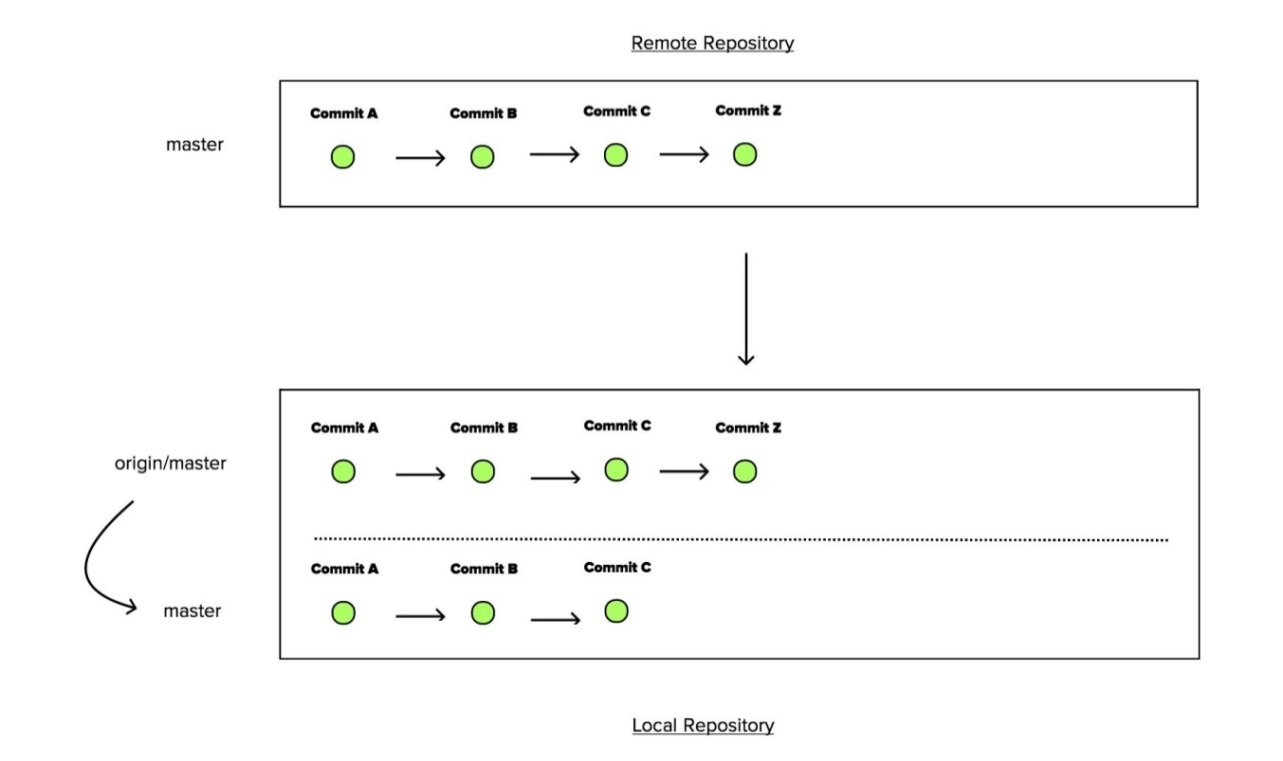
* Esa llave la copiamos en github: personal settings 🡪SSH keys
* Iteracion básica y subir a github
  + $git remote add origin direccion ssh
  + $ git push origin master

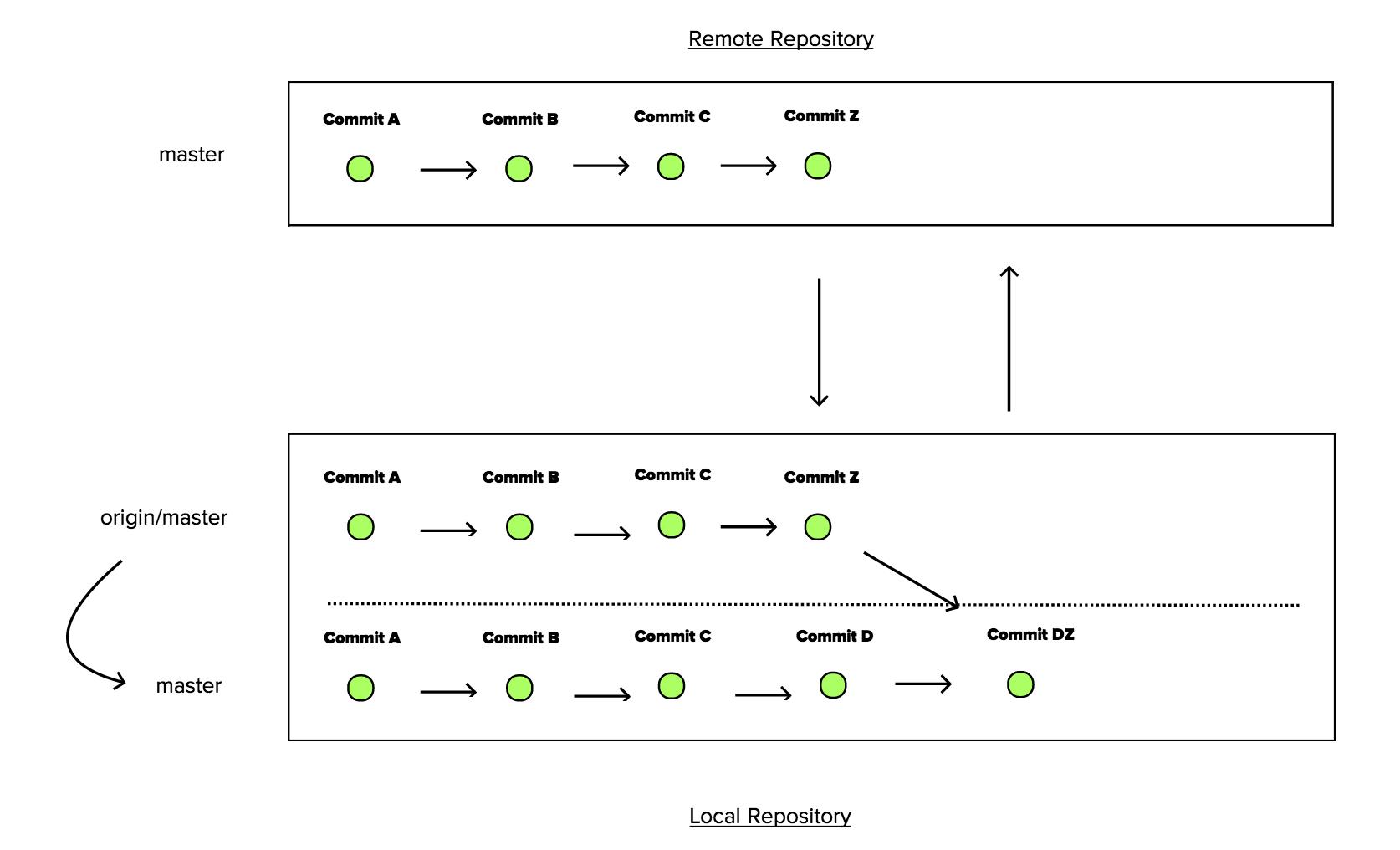
Hacer una conexión remota, subir archivos.

* Crear repositorio
* Iteracion basica
* $ git remote add origin direccionssh (solo la primera vez)
* $ git remote -v Verifica cual es el origen y el destino
* $ git push origin master

## b) Proyectos creados por tu organización ó equipos.

* Son propietarios del proyecto.
* Todos suben cambios, sin pedir permisos.
* Siempre verificar cambios nuevos antes de subir propios.



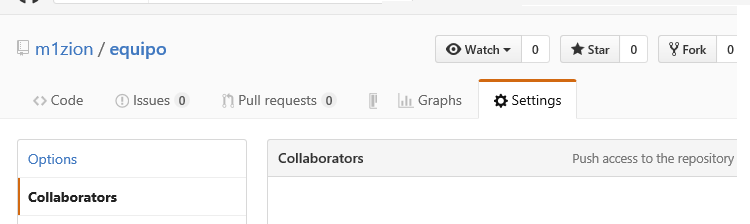


Se descargan dos repositorios, con el fin de poder trabajar offline, en el que trabajamos es en el master.

1. Origin/Master 🡪 Refleja la rama de github, estos cambios se descargan con **git fetch**
2. Después se fusionan con master (DZ) git merge origin/master
3. Haces un push origin/master de DZ

Ejercicio 3

* Crear nuevo repositorio en github
* Utilizar una cuenta alterna para simular trabajo en equipo
* Agregar colaboradores



Descargamos el proyecto de github en git, primero en origin/master y luego en master y después desarrollamos

* **$ git remote add origin [SSH]**
* **$ git fetch origin** (bajarlo a origin/master)
* **$ git merge origin/master**: Es hasta esta fusión cuando se bajan los archivos a mis carpetas (master)
* \*\* Desarrollamos ciclo basico\*\*
* **$ git fetch origin** : Descarga los cambios en origin/master
* **$ git merge origin/master**: fusiona de origin/master a master o+esc+: por si hubo cambios en ambos
* **$ git push origin master**: sube los cambios a github
* **$git pull origin master** baja los cambios al repositorio

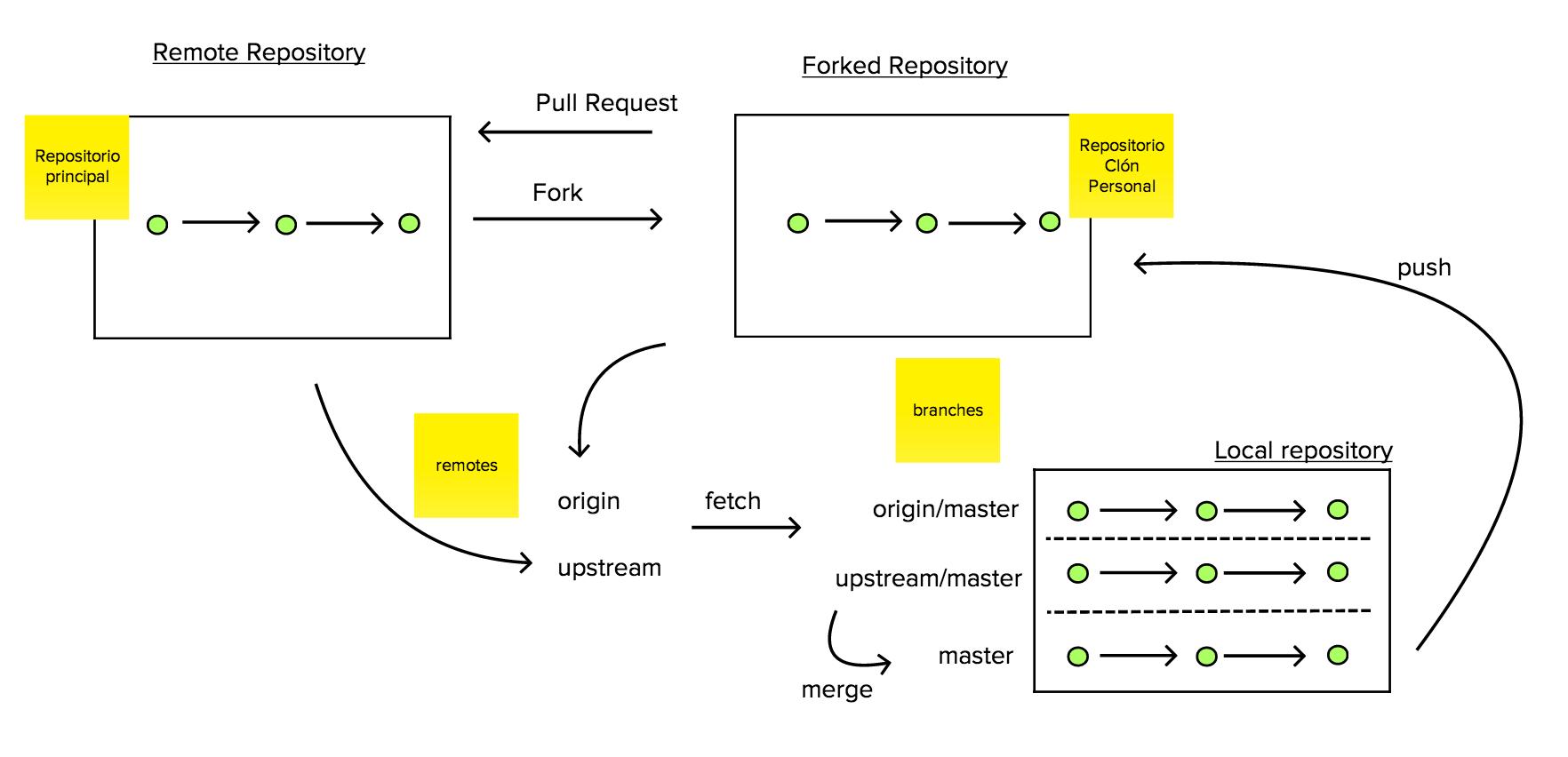
***En caso de hacer un git merge y que este tenga conflictos, hacemos el git fetch, git merge origin/master, git push origin master***

***O shift +z***

*Ponemos una o, el mensaje, le damos esc y :x para salvar*

### c) Proyectos de terceros. (Repositorios “forked”)

Existirán 2 repositorios:

1. El repositorio original.
2. El repositorio forked (La réplica del original, en tu cuenta de GitHub).

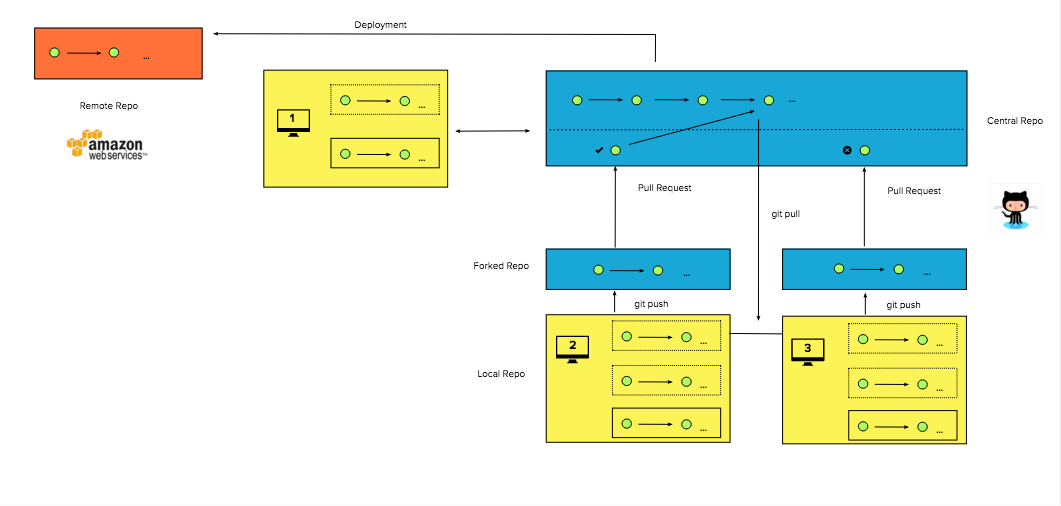
Crear ó entrar a la carpeta del proyecto

* $ git remote add origin [HTTPS ó SSH del proyecto forked]
* $ git remote add upstream [HTTPS ó SSH del proyecto principal]
* $ git fetch upstream
* $ git merge origin/upstream
* $ git fetch origin
* $ git merge origin/master
* Hacer cambios en local
* $ git fetch upstream
* $ git merge origin/upstream
* $ git push origin master

Ejercicio Final

1. Hacer fork al repositorio que deseamos.
   1. buscar el repositorio y clic en el botón fork
2. conectar los remotos origin y upstream
   1. $git remote add origin ssh (de mi repositorio copia)
3. conectar con el repositorio original
   1. $git remote add upstream ssh (del repositorio principal)
4. Bajar los cambios de mi repositorio local a master
   1. $git fetch origin
   2. $git merge origin/master
5. Bajar los cambios (sincronizar) del remote repositori a origin, pos si hicieron cambios mientras trabajaba
   1. $git fetch upstream
   2. $git merge upstream/master
6. Hacer cambios e iteración básica
7. Push a fork repository
   1. $git push origin/master
8. Hacer el pull request en github

### Project Management con Github

Crear organizacion

Agregar colaboradores

Crear equipos

Travis CI (test ) travis-ci.org página para hacer tests

### Issues y Milestones

**Milestones**: Son referencias de cuando inicia y termina un proyecto/proceso, contiene las actividades de cada proceso.

* Dentro de Issues y Repositorio

**Issues**: Actividades que hay que hacer, se pueden agregar etiquetas y responsables (le llega un correo al responsable) , el responsable tiene que dar “close and comment” para terminar el issue.