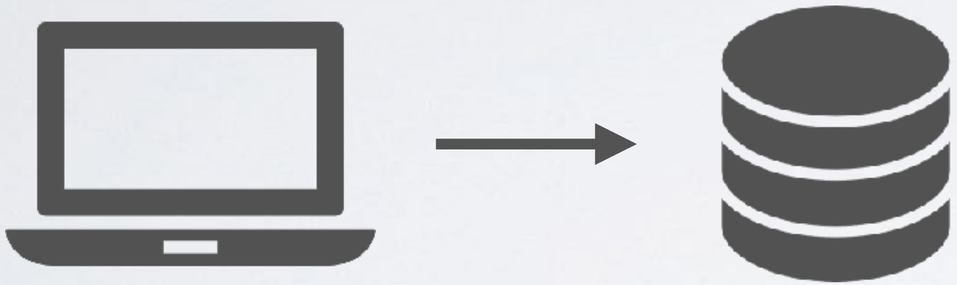
GIT

¿QUÉ ES GIT?

Git es un software de control de versiones open source, diseñado para mantener archivos de manera eficaz y confiable.

Pero…

¿QUÉ ES UN CONTROL DE VERSION?



Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

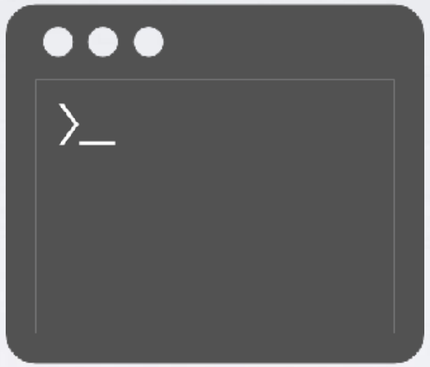
Existen varios sistemas de versionamiento. Nosotros nos centraremos en GIT, debido por sus características y amplio uso, además de la facilidad de trabajar en equipo.

¿CUÁNDO USAR GIT?

Se recomienda usar siempre

[Explicar que sirve para hacer tracking de archivos de texto y de código, pero es igual de útil con imágenes o videos]

Para usar Git en nuestro computador debemos verificar si tenemos instalado y/o configurado GIT



INSTALAR Y CONFIGURAR GIT

VERIFICAR SI TENEMOS GIT EN MAC

```
$ git  --version
git version 2.14.3
```

- Primero debemos saber si lo tenemos instalado. Para hacerlo hay que ingresar al terminal y escribir **git - -version.**

INSTALAR GIT EN MAC

Si no lo tienes instalado debes descargar la última versión de git desde el siguiente link:

Link de Descarga

INSTALAR GIT EN MAC

Sigue las instrucciones de instalación

INSTALAR GIT EN MAC

Volvamos a la terminal y volvamos a escribir **git - -version** para saber si tenemos salió todo bien

INSTALAR GIT EN LINUX

```
$ sudo apt-get update
$ sudo apt-get install git
```

- Para descargar la última versión de Git en tu computador debes usar el comando **apt-get** desde la terminal.
- Primero ingresa s**udo apt-get update.**
- Y luego, **escribe sudo apt-get install git.**

INSTALAR GIT EN LINUX

```
$ git  --version
git version 2.14.3
```

- Verifiquemos que todo salió bien usando **git - -version.**

INSTALAR GIT EN WINDOWS

Recuerdas que en el video anterior descargamos desde **git for windows** e instalamos la terminal en nuestro computador…
Pues bien, estas de suerte, ya que tienes instalada la última versión de Git en tu computador.

CONFIGURAR GIT

```
$ git config --global user.name “Tu nombre”
```

- Para configurar git debemos escribir en el terminal el siguiente comando: **git config --global user.name**, seguido de tu nombre dentro de **comillas dobles**.

CONFIGURAR GIT

```
$ git config --global user.email
“tucorreo@mail.com“
```

- Luego, escribamos el siguiente comando: **git config --global user.email**, seguido de tu correo dentro de **comillas dobles**.

CONFIGURAR GIT

```
$ git config - -list
user.name= tu nombre
user.mail= tucorreo@mail.com
```

- Revisemos si configuramos correctamente Git usando el comando **git config - -list.**

INICIAR GIT

```
 $ git init
(ruta: Users/mi-usuario/Desktop/mi-carpeta)
```

- Para iniciar Git **debes entrar a la carpeta que quieres versionar.**
- Cuando estes dentro escribe **git init**.

CONFIGURAR GIT

```
$ ls -a
.git
```

- Si revisas con **ls -a**, verás una carpeta oculta llamada **.git.**
- Esta carpeta guarda todos nuestros cambios.
- Si borramos esta carpeta perderemos todos nuestros cambios versionados.

ADVERTENCIA

```
$ ls -a
.git
```

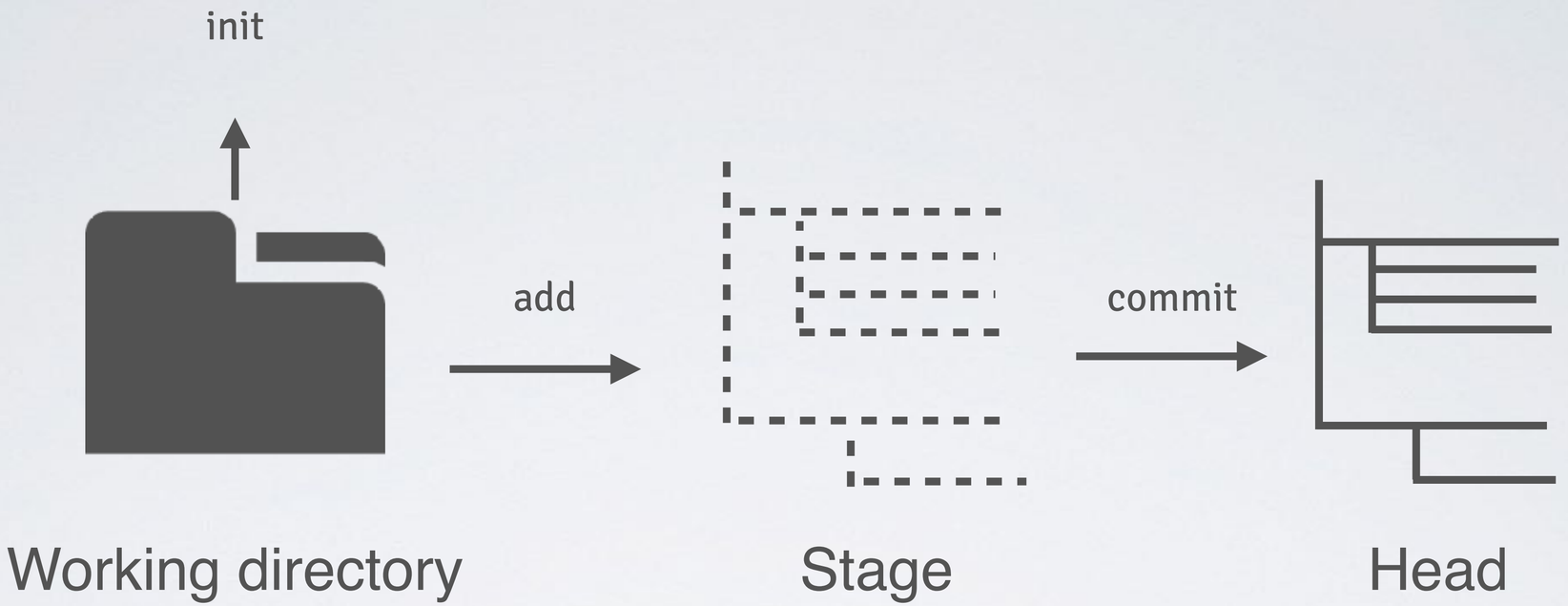
- Es muy importante que NO iniciemos GIT en la carpeta de nuestro usuario o en la carpeta raíz, porque no queremos tener todo nuestro computador bajo control de versiones
- Cada proyecto debe tener su propio control de versiones.

Antes de seguir es importante conocer algunos términos usados en Git

- **Working Directory:** *(El directorio donde estamos trabajando*), contiene todas las modificaciones no añadidas no confirmadas.

- **El índice (Stage):** contiene los cambios añadidos pero no confirmados.

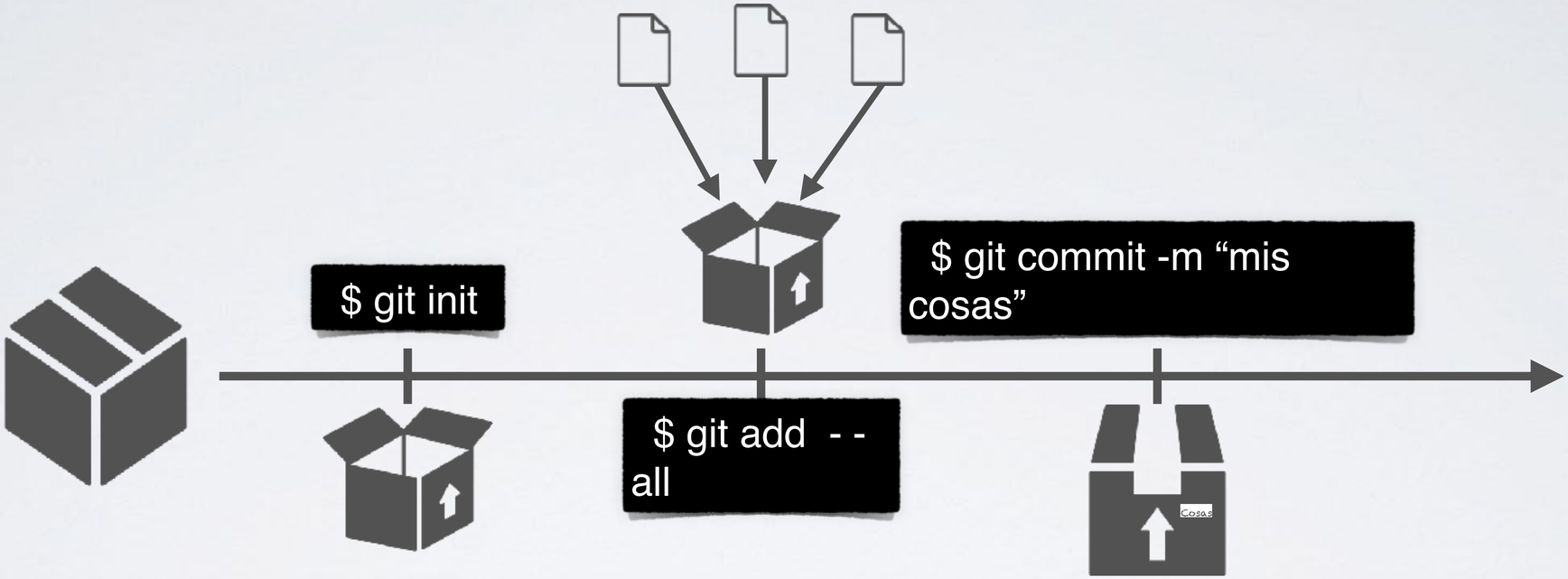
- **Head:** Referencia al último commit hecho o sea apunta a los cambios añadidos y confirmados



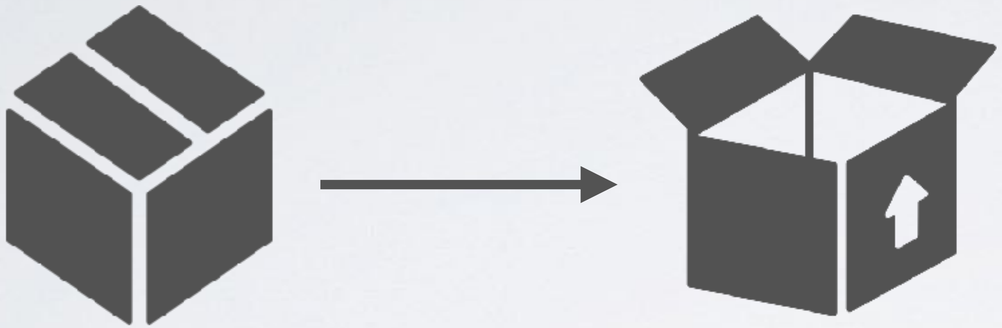
Cuando trabajamos en el flujo de trabajo de Git,
siempre pasa por estos tres puntos.

Para comprender como funciona el flujo de trabajo de Git, imaginemos que queremos guardar nuestras cosas dentro de una caja.

FLUJO DE TRABAJO DE GIT



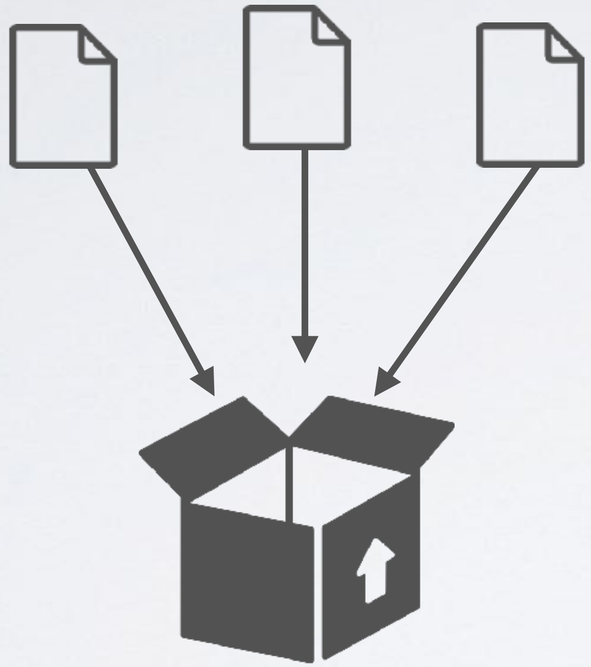
ABRIR LA CAJA



```
$ git init
```

- Para hacerlo primero debemos abrir la caja.
- Abrir la caja es igual a escribir **git init** en la terminal.

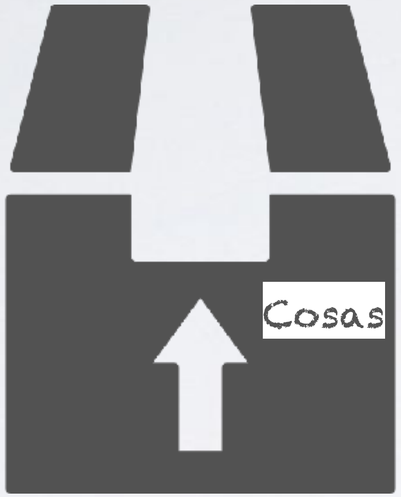
AGREGAR COSAS A LA CAJA



```
$ git add --all
```

- Ahora que esta abierta, es momento de agregar todas nuestras cosas a la caja.
  - Esta acción es igual a escribir **git add - -all o uno a uno con git add [nombre_de_archivo] .**

CERRAR Y ETIQUETAR LA CAJA

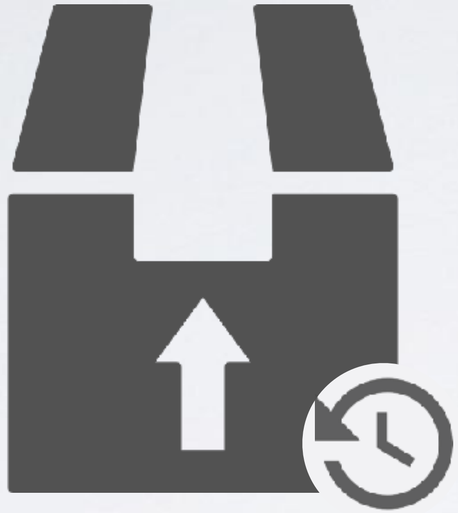



```
$ git commit -m “mis cosas”
```

- Luego de agregar todas nuestras cosas, habrá que etiquetar y enlistar las cosas de nuestra caja.
- Esta acción es igual a escribir **git commit -m “mis cosas”**.

Además, existen dos comandos que te ayudarán a revisar partes del proceso y el historial de cambios hechos.

REVISAR LOS CAMBIOS EN LA CAJA



- Cuando quieras revisar el historial de cambios ha tenido la caja usa **git log**.
- Con el podrás ver el **checksum**, la persona que hizo el cambio, la fecha y el mensaje del commit.

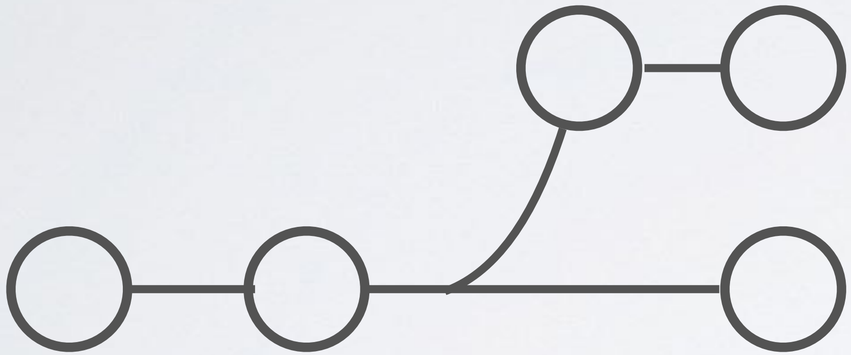
```
$ git log
commit c15d309c4eeb88d0d86f8092ab726108db0c5784
-> Checksum
Author: Lalo landa <micorreo@mail.com>
Date:   Tue apr 2 09:59:58 2017 -0300
 mis cosas
```

REVISAR LA CAJA

```
$ git status
On branch master
nothing to commit, working tree clean/mi-usuario/Desktop/mi-carpeta)
```

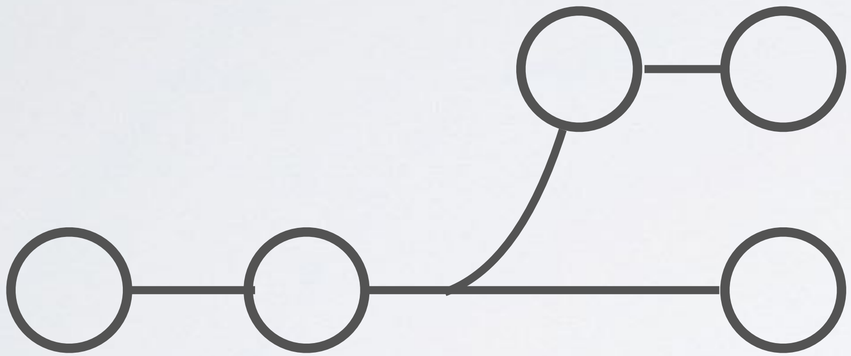
- **git status** nos ayudará a saber en que parte del flujo de trabajo nos encontramos.
- Es es similar a revisar el proceso dentro de una lista.

GIT BRANCH



- Git branch representa a una linea de desarrollo en las cuales se pueden editar y hacer cambios.
- Estás son importantes a la hora de trabajar con otras personas ya que dan un espacio de trabajo único para cada desarrollador.

¿PARA QUE SIRVEN LOS BRANCHES?



- Para coordinar el trabajo con varias personas
- Para implementar cambios complejos y si no resultan destruirlos fácilmente
- Para poder implementar hotfixes mientras estamos trabajando en partes complejas que no hemos terminado de implementar

COMANDOS PARA BRANCHES

- **git branch:** Muestra todos los branches en el proyecto.
- **git checkout [nombre rama]:** Cambia a una rema especificada.
- **git branch -b [nombre rama]:** Crea y cambia a una rama especificada.
- **git merge [nombre rama]:** Une una rama especificada al lugar donde estas.
- **git branch -d [nombre rama]:** borra una rama ya mergeada.
- **git branch -D [nombre rama]:** Borra una rama no mergeada.