Colaborando con Git

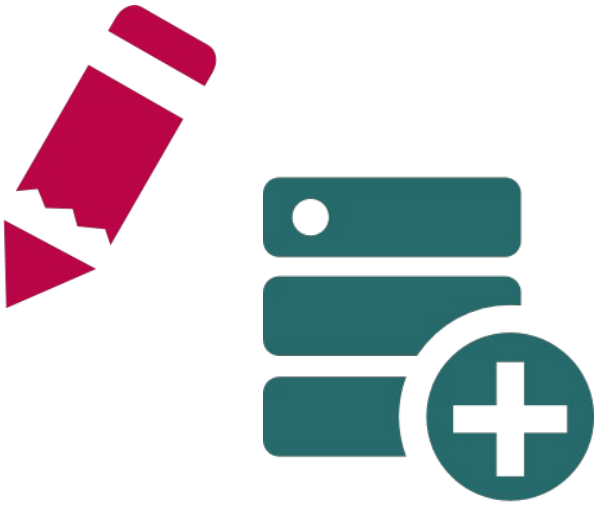
ISC. Max Jonathan Rodríguez Beltrán

@Jaxmetalmax | max@sml.mx

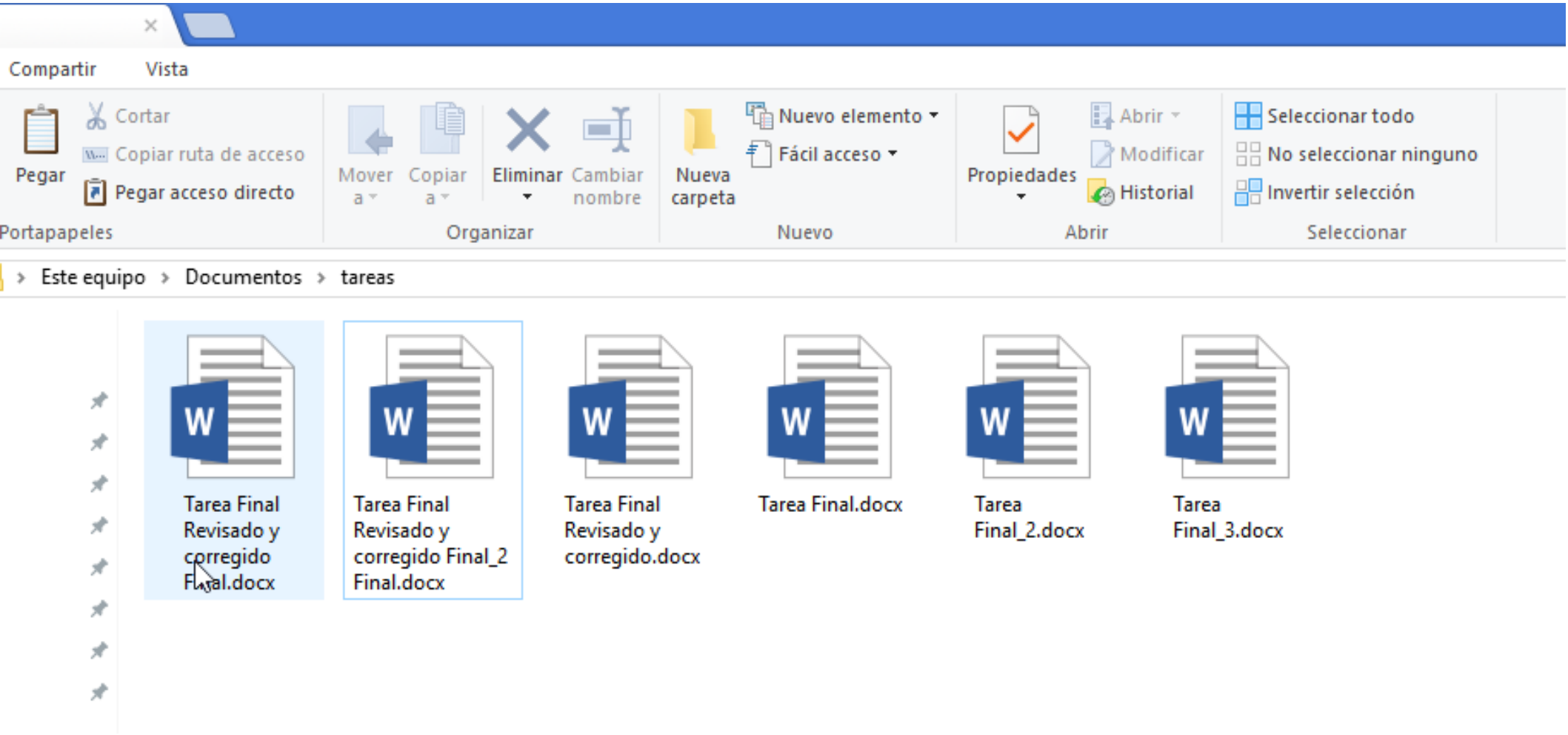
¿Sistema de control de versiones?

Flujo habitual de trabajo

- Creamos un archivo.
- Lo editamos.
- Lo guardamos.
- Lo volvemos a editar.
- Y lo volvemos a guardar 'n' numero de veces.



El típico caso de “Tarea Final v2.doc”



Entonces...

Es un sistema que **registra los cambios** de uno o varios archivos **a través del tiempo**, permitiéndote recuperar **versiones especificas** de estos mas adelante.

Marty, you gotta come back with me!

¿Sistemas de control de versiones?

https://www.finditez.com/images/vcs_logos.png

Tecnologías y prácticas

Seguir y controlar cambios

¿Que podemos usar para organizar esto?

got git?

Git

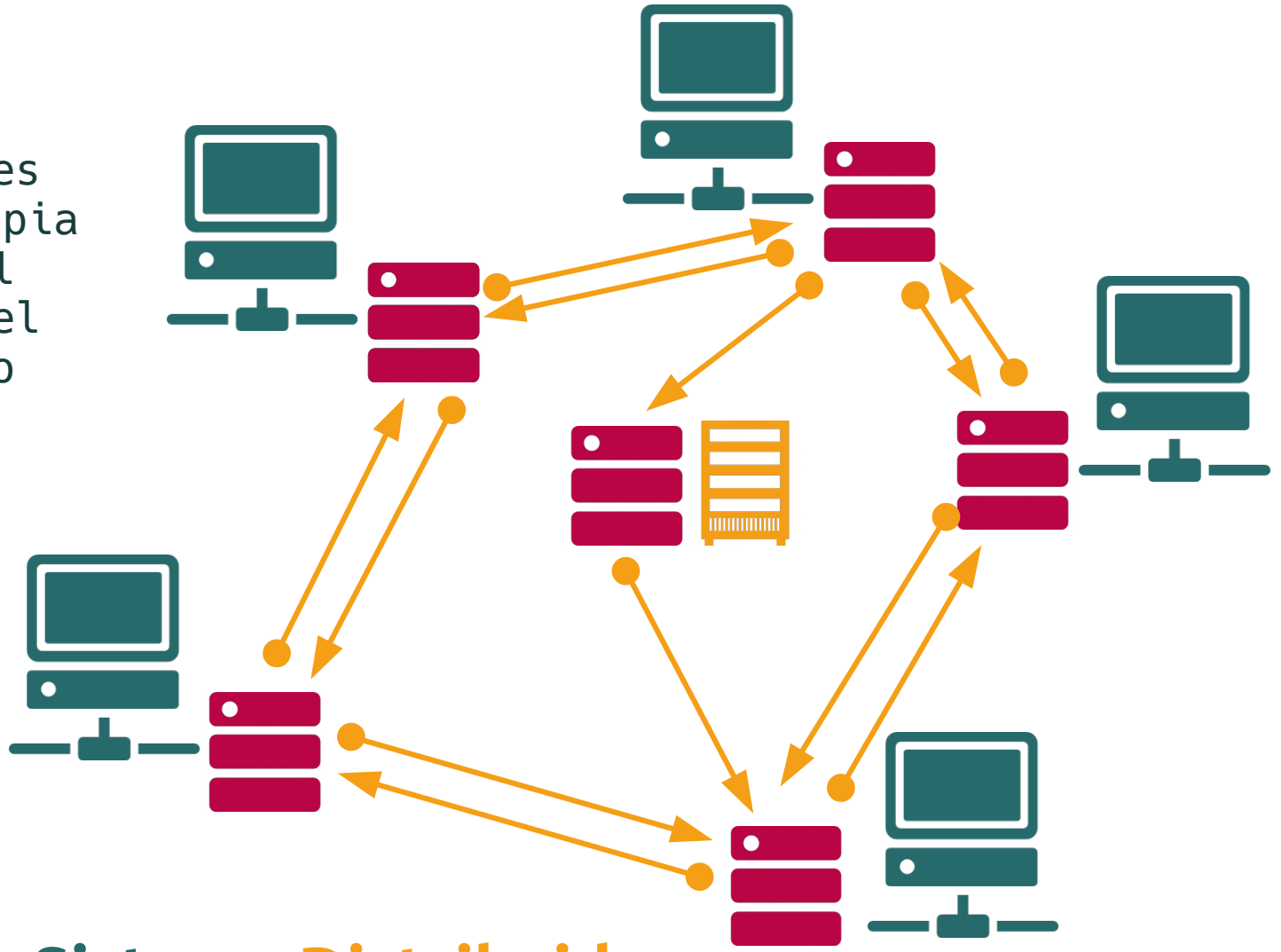
Diseñado por Linus Torvalds (Creador del kernel Linux)

Surge en el 2005 a partir de la necesidad de reemplazar el sistema de control de versiones que se utilizaba en ese entonces (Perforce) para el proyecto del kernel de Linux.

Actualmente es unos de los sistemas de control de versiones mas utilizados.

Características de Git

Todos los colaboradores tienen una copia con todo el historial del repositorio



Sistema Distribuido

Git es seguro

Git utiliza hashes SHA1 para garantizar que todos tienen el mismo archivo.

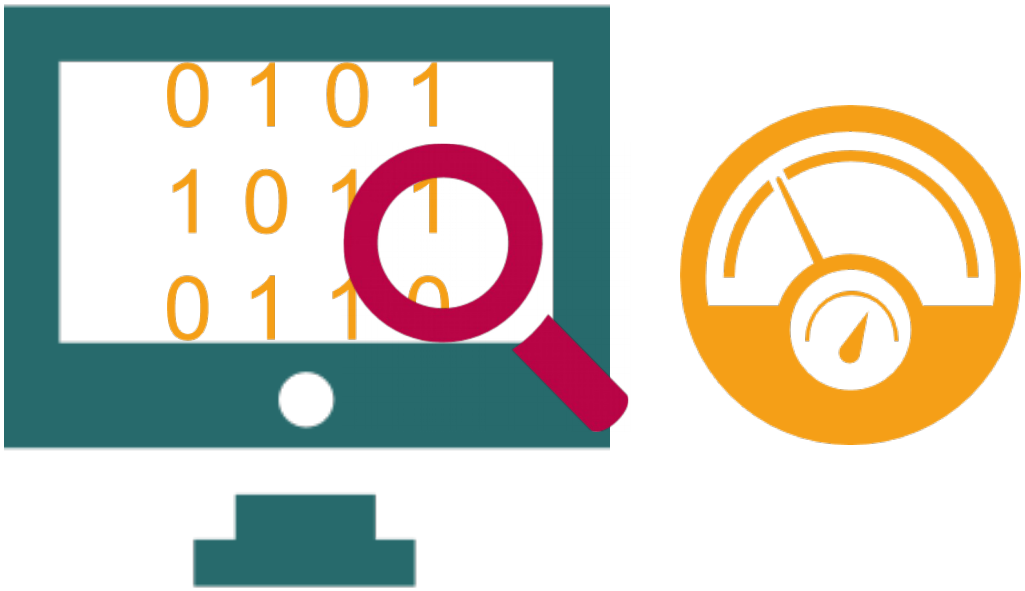
Ademas al utilizar el protocolo SSH asegura que los datos viajan seguros por la red.

Sin dependencia

Git al ser un sistema de control de versiones distribuido nos permite trabajar sin una conexión directa al servidor.

¡Git es rápido!

La mayoría de las operaciones en git son locales, esto disminuye bastante el tiempo de respuesta por ejemplo al buscar en el historial.



Cada repositorio de un usuario es un **clon completo**, suponiendo que nuestro servidor central falla, no perdemos el historial de versiones.

Si uno cae, otro sale al quite.

Si algo pasa con nuestro equipo, siempre y cuando se hayan **subido cambios** al repositorio, nuestro código estará **seguro**

Si decidimos designar un servidor central tenemos opciones como:

github
SOCIAL CODING

GitLab

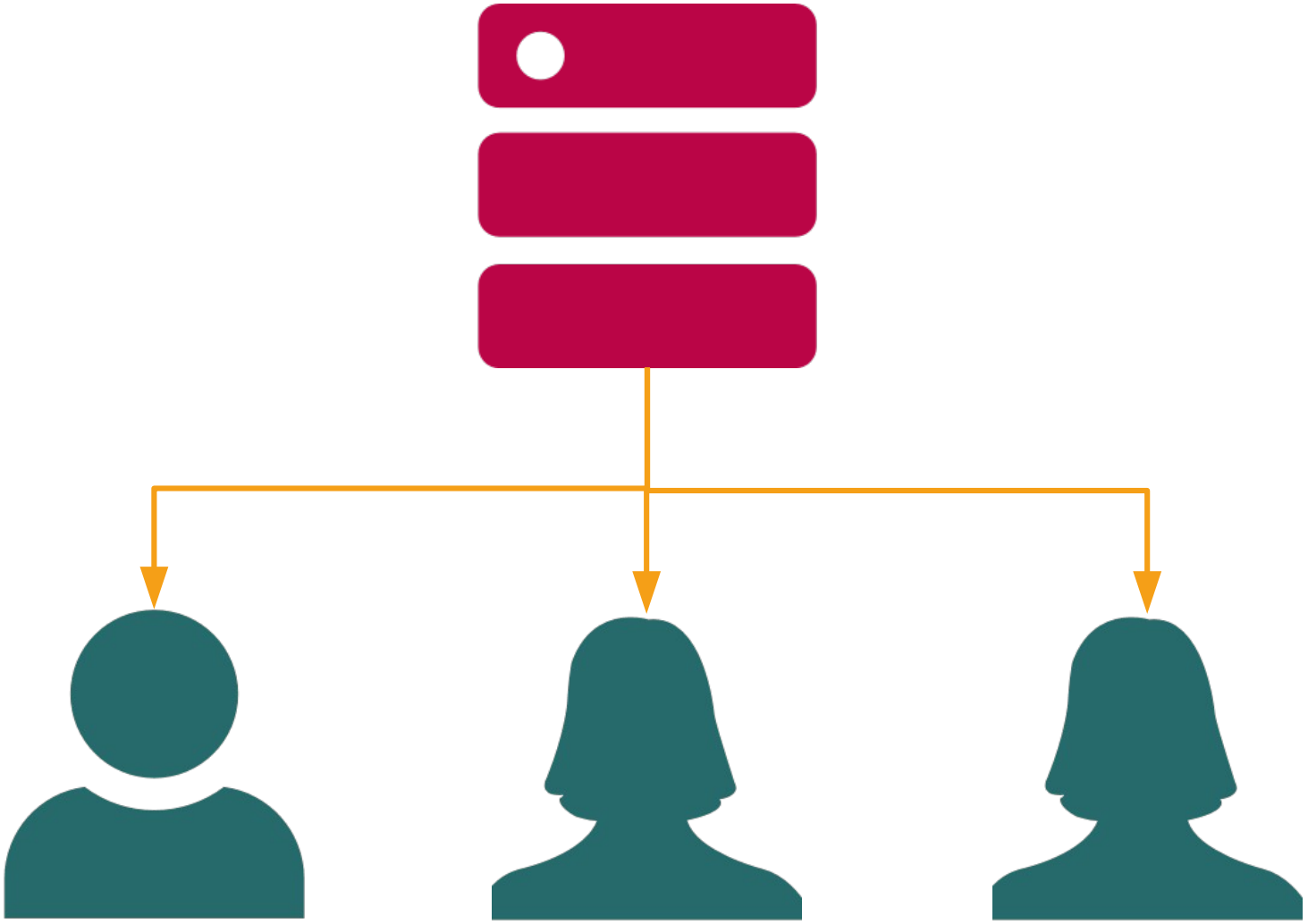
Bitbucket

Flujos de trabajo con Git

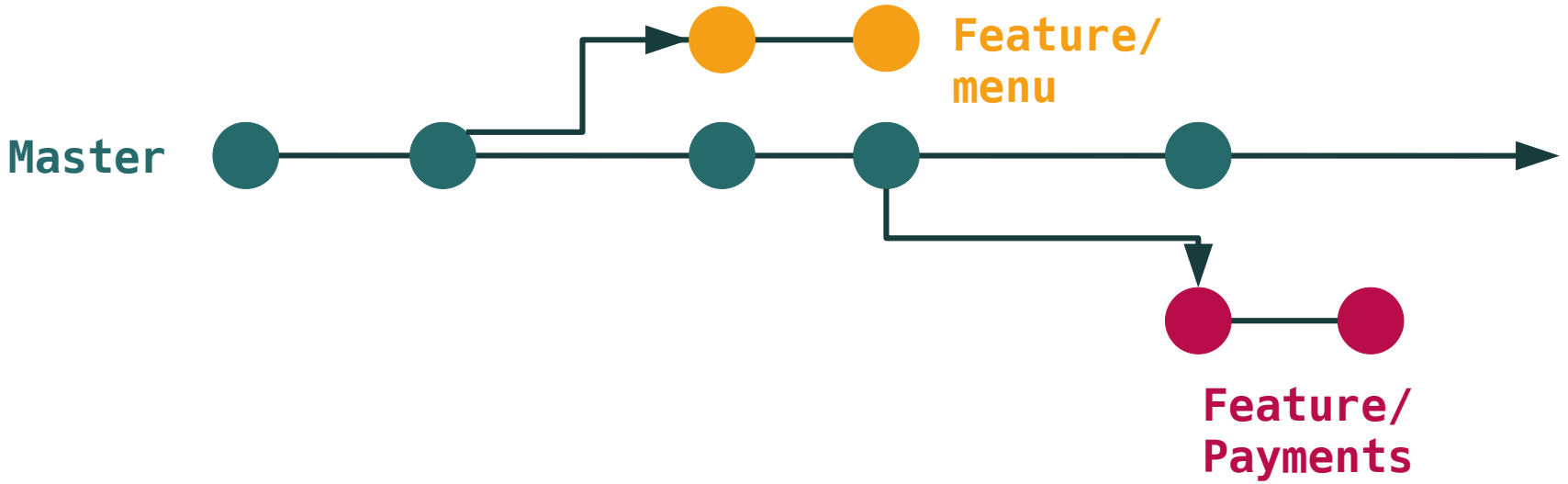
Git es tan flexible que puedes adaptarlo a tu forma de trabajo

Existen algunos flujos de trabajo que podemos tomar como ejemplo:

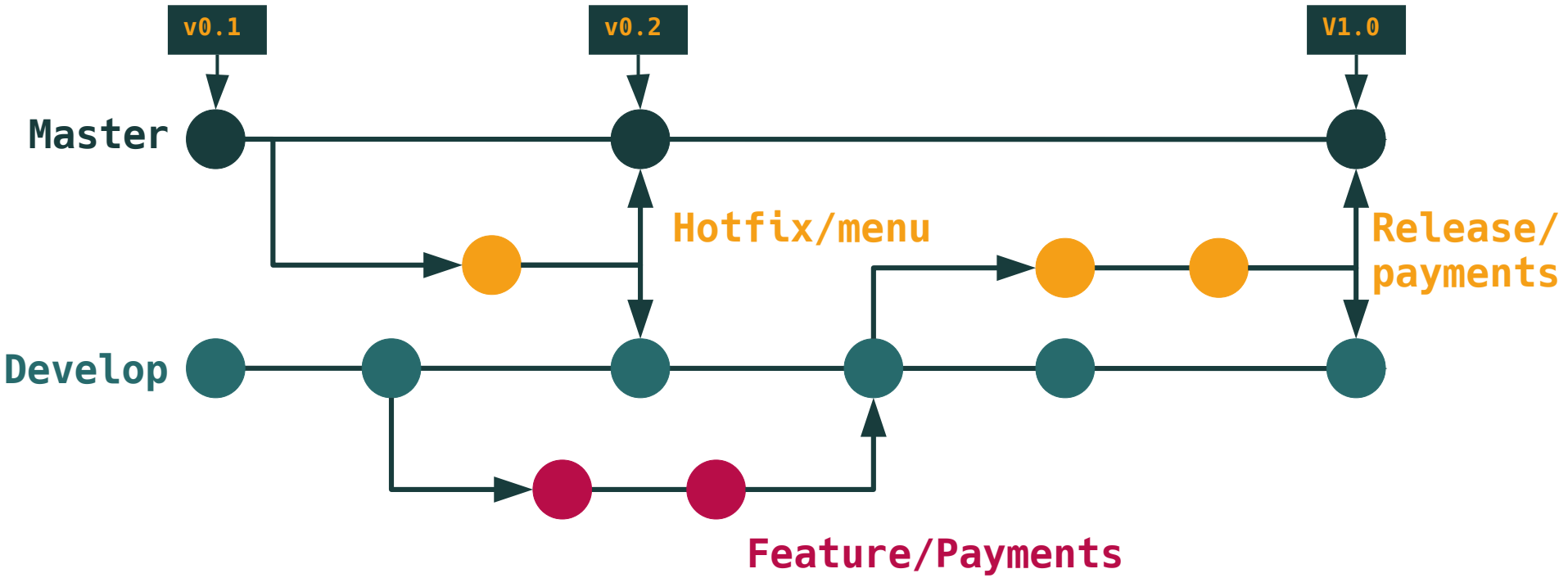
Workflow centralizado



Workflow basado en ramas (Feature Branch)



Flujo de trabajo Git (GitFlow)



Git init & Git add

Iniciar repositorio

Iniciar en directorio

```
$ git init
```

Añadir archivos y directorios que no queremos versionar

```
.gitignore
```

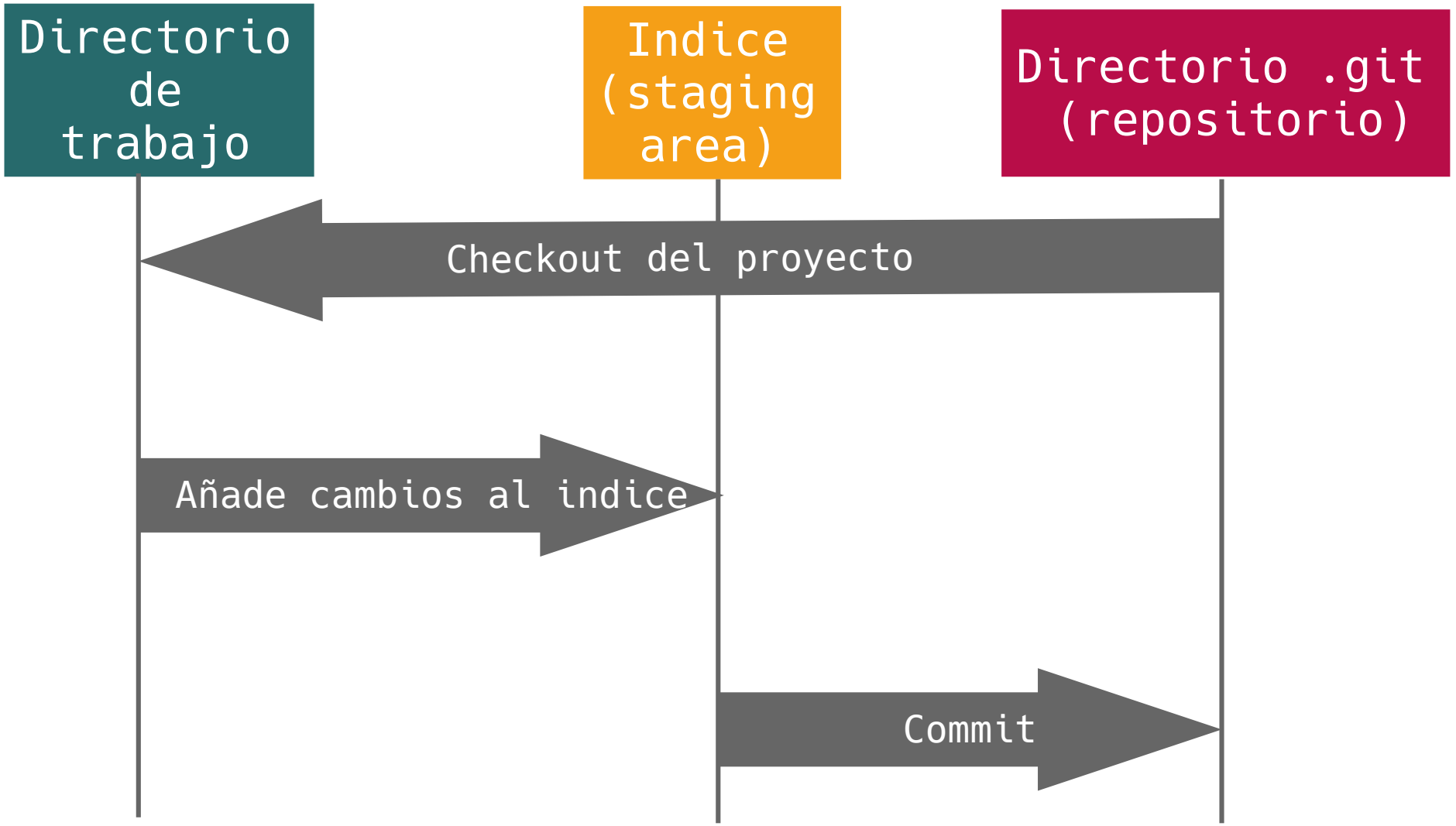
Es importante tener en cuenta este archivo para no versionar archivos innecesarios o con información sensible

Añadir archivos

Añadir archivos

```
$ git add .
        ó
$ git add --all
```

En este punto los archivos se encuentran en el índice o Staging area, listos para guardarse (crear una versión)


Directorio de trabajo
Indice (staging area)
Directorio .git (repositorio)
Checkout del proyecto
Añade cambios al indice
Commit

Git status & Git Commit

Revisamos el estado del repositorio

Antes de guardar revisamos el estado del repositorio

```
$ git status
```

Ésto nos mostrará el estado del repositorio y sus archivos

Revisar cambios en el indice
Podemos ver los cambios que estamos por guardar con:

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

Guardar cambios

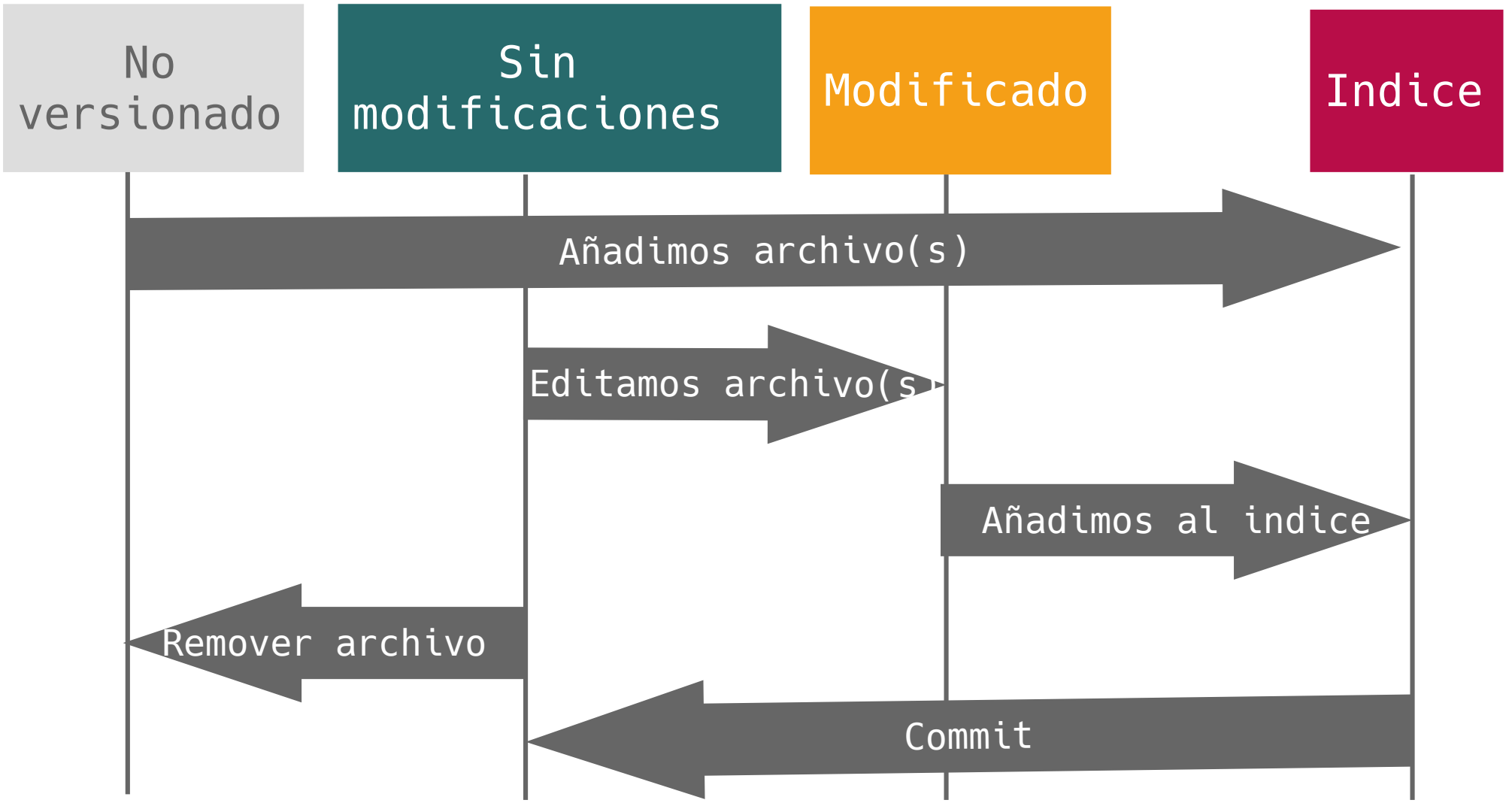
Hacemos commit de los archivos

```
$ git commit -m
[mensaje de la versión]
```

Y tenemos la versión de atajo *(estos dos parámetros nos permiten añadir los archivos al indice y hacer commit en un solo comando)*:

```
$ git commit -a -m
[mensaje de la versión]
```

NOTA: el parámetro "-a" solo funciona para los archivos que previamente han sido registrados en un commit previo


No versionado
Sin modificaciones
Modificado
Indice
Añadimos archivo(s)
Editamos archivo(s)
Añadimos al indice
Remover archivo
Commit

Git log & Git remote

Revisemos el historial de commits

Una vez que tengamos algunos commits podemos ver el historial con:

```
$ git log
```

Ésto nos mostrará un listado de los commits, comenzando por el mas reciente hasta el mas antiguo

Podemos dar diferentes formatos
Por ejemplo, el historial resumido en una sola linea y en formato de gráfica

```
$ git log --graph --oneline
```

Definiendo un remoto

Vamos a definir un servidor remoto donde estará nuestro repositorio

```
$ git remote -v
```

```
$ git remote add [nombre] [url]
```

Por default el remoto principal se llama “origin” pero podemos tener mas de un remoto y nombrarlos a nuestro gusto

Git push/pull & Git clone

Realizar push de nuestro repositorio

Una vez que tengamos algunos commits y nuestro remoto definido vamos subir los commits:

```
$ git push [remoto] [rama]
```

Ésto nos mostrará un listado de los commits, comenzando por el mas reciente hasta el mas antiguo

Podemos dar diferentes formatos
Por ejemplo, el historial resumido en una sola linea y en formato de gráfica

```
$ git log --graph --oneline
```

Clonando un repositorio remoto

Si tenemos un repositorio remoto y queremos descargarlo en nuestro equipo para trabajar necesitamos clonarlo:

```
$ git clone [url]
```

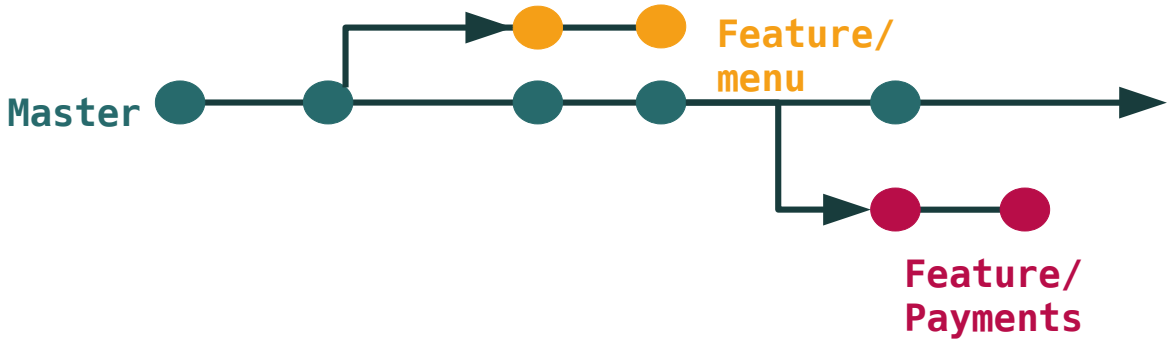
Si queremos descargar cambios en nuestro repositorio vamos a realizar un pull:

```
$ git pull [remoto] [rama]
```

Git branch & Git checkout

Una de las principales ventajas de git son sus branches

Git utiliza un modelo de arbol para manejar las versiones (no es incremental como otros SCV), entonces una rama es un apuntador a un punto en especifico del arbol o rama padre.



Listar las ramas existentes

```
$ git branch
```

Crea una rama nueva y nos cambiamos a esa rama

```
$ git checkout -b [rama]
```

Crea una rama

```
$ git branch [nombreRama]
```

Git merge

Continuando con las ramas, una vez que terminamos de trabajar en una, la fusionaremos a la rama “padre”

Nos movemos a nuestra rama padre a donde fusionaremos nuestra rama hija:

```
$ git checkout [rama]
```

Y fusionamos la rama:

```
$ git merge [rama]
```

Y podemos borrar nuestra rama hija

```
$ git branch -d [rama]
```

Gracias por su atención

Twitter & email
@Jaxmetalmax
maxjrb@openitsinaloa.com

Comunidades en Facebook:
https://www.openitsinaloa.com
https://www.facebook.com/groups/openitsinaloa/
https://www.facebook.com/TomatoValley/?fref=ts