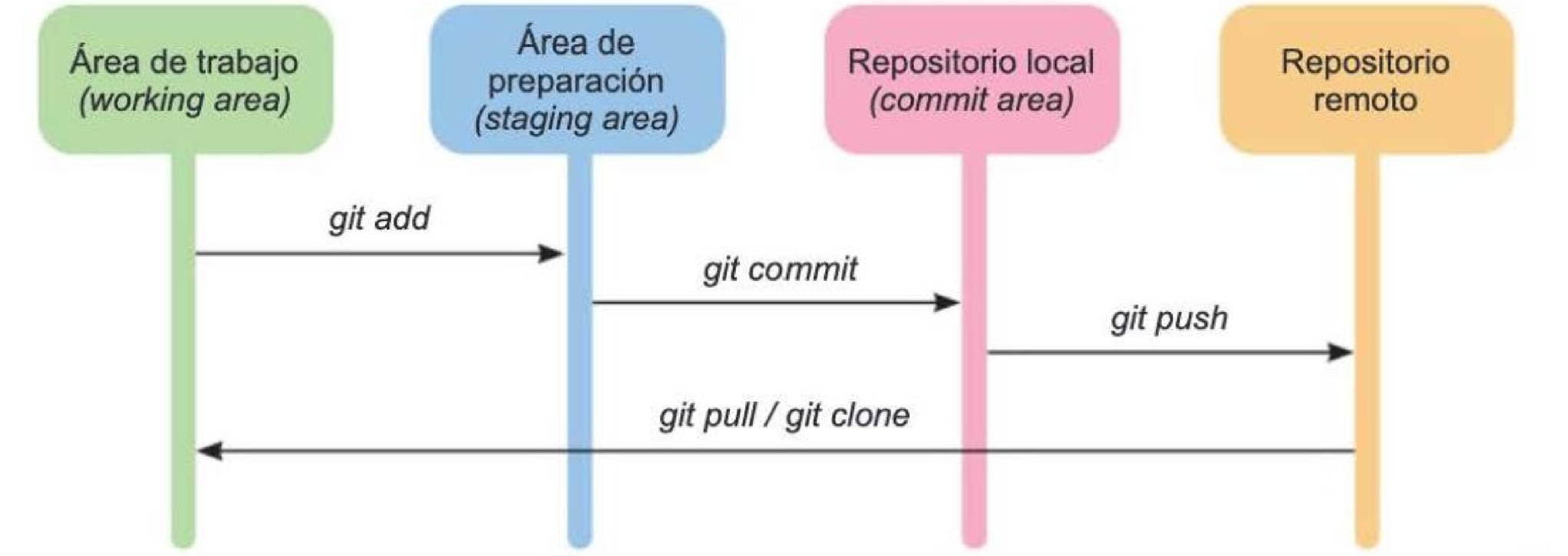
# Git

## Clase 05

staging

# staging


Área de trabajo
(working area)
Área de
preparación
(staging area)
Repositorio local
(commit area)
Repositorio
remoto
git add
git commit
git push
git pull / git clone

# staging

Casos en los que se puede encontrar un fichero

# git log

# git log

El comando git log en Git es utilizado para revisar el historial de commits que se han realizado en un repositorio.

Este comando es extremadamente útil para rastrear cambios, entender la historia del desarrollo y realizar diagnósticos

# git log

```
git log --oneline
```

# git log

```
git log --oneline --reverse
```

# git log

git log --oneline --graph

Muy útil ahora cuando empecemos con las ramas

# git show

# git show

Es un comando poderoso en Git utilizado para mostrar varios tipos de objetos en el repositorio de Git, incluyendo commits, tags, y más. Este comando es extremadamente útil para revisar cambios históricos, diferencias entre versiones y detalles específicos de la configuración del repositorio.

# git show

1. **Mostrar un Commit Específico**

bash Copy code

```
git show [commit]
```

# git show

`--stat`: Muestra un resumen estadístico del commit, incluyendo qué archivos fueron modificados y cuántas líneas fueron añadidas o eliminadas.

bash Copy code

```
git show --stat [commit]
```

# git show

`--name-only`: Lista los archivos modificados en el commit sin mostrar las diferencias.

bash Copy code

```
git show --name-only [commit]
```

# git show

`--format=oneline`: Muestra la información del commit en una sola línea.

bash Copy code

```
git show --format=oneline [commit]
```

# git show

Tabmién puede ser utilizado también para mostrar el contenido de un archivo en un punto específico en la historia del repositorio de Git, o para ver cómo era un archivo en un determinado commit

# git show

1. **Mostrar un Archivo en un Commit Específico**Para ver el estado de un archivo como estaba en un commit específico, utiliza:

bash Copy code

```
git show [commit]:[path/to/file]
```

bash Copy code

```
git show 1a2b3c4d5e:path/to/example.txt
```

# git show

3. **Ver los Cambios de un Archivo en un Commit Específico**Si quieres ver qué cambios se hicieron en un archivo específico en un commit particular, puedes usar `**`git show`**` seguido por el hash del commit y filtrar por archivo.**Ejemplo:**

bash Copy code

```
git show 1a2b3c4d5e -- path/to/example.txt
```

# Ramas

# Ramas

¿Os acordáis de la configuración por defecto que hicimos de git al principio?

Una de las cosas era la rama por defecto

# Ramas

Para qué se utilizan

- Varios desarrolladores
- Varias versiones
- Desarrollo
- Poducción
- Errores

# Ramas

Checkout

Merge

# Ramas

Comandos basicos de Git

## git ¿Que es un Branch y cómo funciona un Merge en Git?

### Ideas

**Master:**
Todo lo que esta en esta rama va a producción.

**Development:**
Las nuevas features, características y experimentos

**HotFix:**
Aqui van los errores se solucionan tan pronto como sea posible.

Si las ramas tienen algun conflicto para unirse git te avisara y te pedira que los corrijas, los conflictos pueden deberse a que se modficaron las mismas lineas del archivo en las dos ramas.

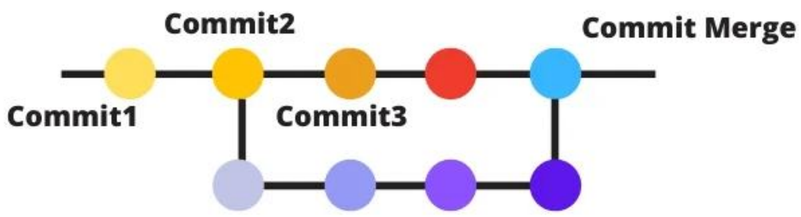

### Notas Clase

Cuando creas tu repositorio en tu carpeta de trabajo se crea por defecto una rama (Branch) principal llamada master.

Branch se puede ver como el mapa lineal de los commits que haz realizados al archivo.


Usas checkout para crear una nueva rama desde el commit que desees de tu rama master, esta rama te sirve para hacer experimentos o reparar errores de tu código principal sin afectar al mismo.


Y para unir los cambios de esta rama de prueba con master utilizas merge, de esta manera las dos ramas se unirán formando una nueva rama.




### Resumen

Branch es aquella que representan los commits como un mapa lineal de tiempo, es posible crear nuevas ramas para realizar modificaciones de nuestro código sin afectar la rama principal.

Merge es el comando que se usa para unir dos ramas, generalmente los merge se hacen desde la rama master, se debe tomar en cuenta que puede haber conflicto entre las ramas.

@YisusJoe

# Como volver en la historia

# Historia

Restaurar archivo a último commit

git checkout archivo.ext

# Historia

Navegación entre commits

git checkout commit

# Historia

Restauración de versiones anteriores de archivo

git checkout commit archivo.ext

# Volver en la historia persistentemente

# Historia persistente

Es un comando poderoso en Git que se utiliza para deshacer cambios en el estado actual del repositorio. Es fundamental para ajustar el estado del índice (staging area), manipular el historial de commits y modificar el HEAD actual.

# Historia persistente

## Deshacer Commits

git reset se usa para deshacer commits. Puede revertir un repositorio a un estado anterior, eliminando commits del historial.

# Historia persistente

## Despreparar Cambios

También se utiliza para remover cambios que han sido añadidos al índice (staging area) sin eliminar los cambios en el directorio de trabajo.

# Historia persistente

**Limpiar el Índice y el Directorio de Trabajo**

Con opciones específicas, git reset puede ser usado para eliminar todos los cambios en el índice y el directorio de trabajo.

# Historia persistente

## git reset --soft

El modo --soft revierte el repositorio a un commit específico, pero no altera el índice ni el directorio de trabajo. Esto significa que los cambios que estaban preparados para el commit antes del reset seguirán estando en el índice después del reset.

# Historia persistente

**git reset --hard**

El modo --hard es el más drástico. Revierterá el repositorio a un commit específico, y descartará todos los cambios en el índice y en el directorio de trabajo, limpiando cualquier cambio no guardado.

# ‹Despedida›

**Email**

bienvenidosaez@gmail.com

**Instagram**

@bienvenidosaez

**Youtube**

youtube.com/bienvenidosaez