GIT

¿QUÉ ES?

Geometric **I**nvariant **T**heory, nos centraremos en curvas polarizadas

Curvas de grado d

$\forall \quad d > 2(2g-2)$ considere el cubesquema abierto y cerrado $Ch^{-1}(Chow_d^{ss}) \subset Hilb_d$ usaremos la notación abreviada

$$H_d = Ch^{-1}(Chow_d^{ss}) \subset Hilb_d$$

H_d es el principal componente de Chow-semistable locus, similar a

$$H_d = Hilb_d^{sso} = \{[X \subset P^r] \in Hilb_d^{ss} : X \quad es \quad conexo\}$$

Un breve comienzo con Git y GitHub

COMUNIDAD DE INNOVACIÓN ESCOM

¿Qué es una versión de control y cuál es su objetivo?

1. Es un sistema de administración de múltiples versiones de un proyecto.
2. Su objetivo es realizar un seguimiento de los cambios realizados en nuestros archivos

Git

TRABAJANDO LOCALMENTE

git-scm.com

Git como VCS

1. Sistema de control de versiones distribuido de código abierto.
2. Almacena códigos, configuración y clasifica la versión de los sistemas
3. Trabaja como servidor y como cliente.
4. Revisa los cambios hechos por otros

Instalar Git

Para ver si tenemos instalado git ponemos el comando

```
git --version
```

Linux

Debian/Ubuntu sudo apt install git	Fedora yum/dnf install git
Gentoo emerge --ask --verbose dev-vcs/git	Arch Linux pacman –S git
OpenSUSE zypper install git	Mageia urpmi git
Nix/NixOS nix-env -I git	FreeBSD pkg install git
Solaris 9/10/11 pkgutil -i git	Solaris 11 Express pkg install developer/versioning/git
OpenBSD pkg_add git	Alpine apk add git
Slitaz tazpkg get-install git	

Windows

Descargar la versión más reciente en

gitforwindows.org

iOS

Ver si tiene instalado git con

```
git --version
```

Descargar en ese mismo cuadro de diálogo

Primeros pasos con Git

IDENTIDAD, REPOSITORIOS, WORKING DIRECTORY, STAGING AREA Y ALGUNOS COMANDOS

Identidad

- Nombre

```
git config --global user.name "my_name"
```

- Correo electrónico

```
git config --global user.email "my_email"
```

El comando --global indica que se debe de usar esa configuración para todos los futuros repositorios de Git

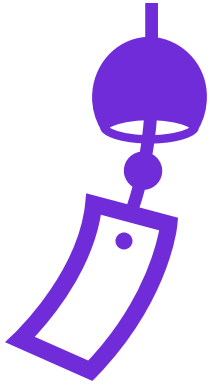
Actividad 1

- Revisar Git en la computadora
- Configurar nuestra identidad

Repositorios

LUGAR DONDE SE ALMACENAN LOS PROYECTOS Y LOS CAMBIOS REALIZADOS

Crear un repositorio



Crear el repositorio desde cero

Clonar un repositorio existente

Comando **git init**

CREA UN NUEVO REPOSITORIO EN LA CARPETA DESEADA

Actividad 2

- Hacer nuestro primer repositorio
- Ver el contenido

.git

- Lugar donde se guardan las copias de los archivos que tenemos
- Cada copia se hace con un *commit*
- El archivo HEAD apunta al *branch* ó a la subversión del proyecto en el cual andamos trabajando
- *MASTER* es el *branch* principal

Working Directory

- Lugar donde podemos interactuar directamente con nuestros archivos
- Área vacía afuera del directorio ".git"
- Precursor al Staging Area

Comando **git status**

- Nos permite ver los cambios que se han hecho en el Working Directory
- Muestra los archivos que serán enviados al Staging Area

Staging Area

- Área donde los archivos van antes de hacer alguna copia en el repositorio
- Sólo los archivos en el Staging Area serán copiados

Comando
git add name_of_file

- Agrega un archivo del Working Area al Staging Area

1. Archivos selectos

   git add f1 f2

1. Todos los archivos

   git add .

- Verificamos los archivos en el Staging Area con git status

Comando **git commit**

Indica hacer una copia de los archivos del Staging Area

git commit –m "message"

Comando **git log**

Crea una lista de todos los comit que se han hecho

git log --oneline

Nombre, autor, fecha, mensaje

Identificar un commit

- Se identifican por los 40 caracteres que son obtenidos cuando se hace un hash al commit, estos se obtienen a partir de un SHA1

Comando

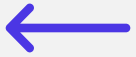
git show commit_ID

MOSTRAR MÁS A DETALLE EL CONTENIDO DEL *COMMIT* QUE HEMOS HECHO

Flujo de trabajo de Git

Directorio .git

Working Area



Staging Area

Repositorio

Comando
git commit –a –m "message"

Pone un archivo directo del Working Area al Repositorio

NOTA: Se puede usar solo para modificaciones de archivos, es decir, con archivos que ya estén en el repositorio

Actividad 3

- Working Directory
- Git status
- Git commit
- Modificación de archivos
- Git log

Información

CAMBIOS, BORRAR Y RENOMBRAR LOS ARCHIVOS

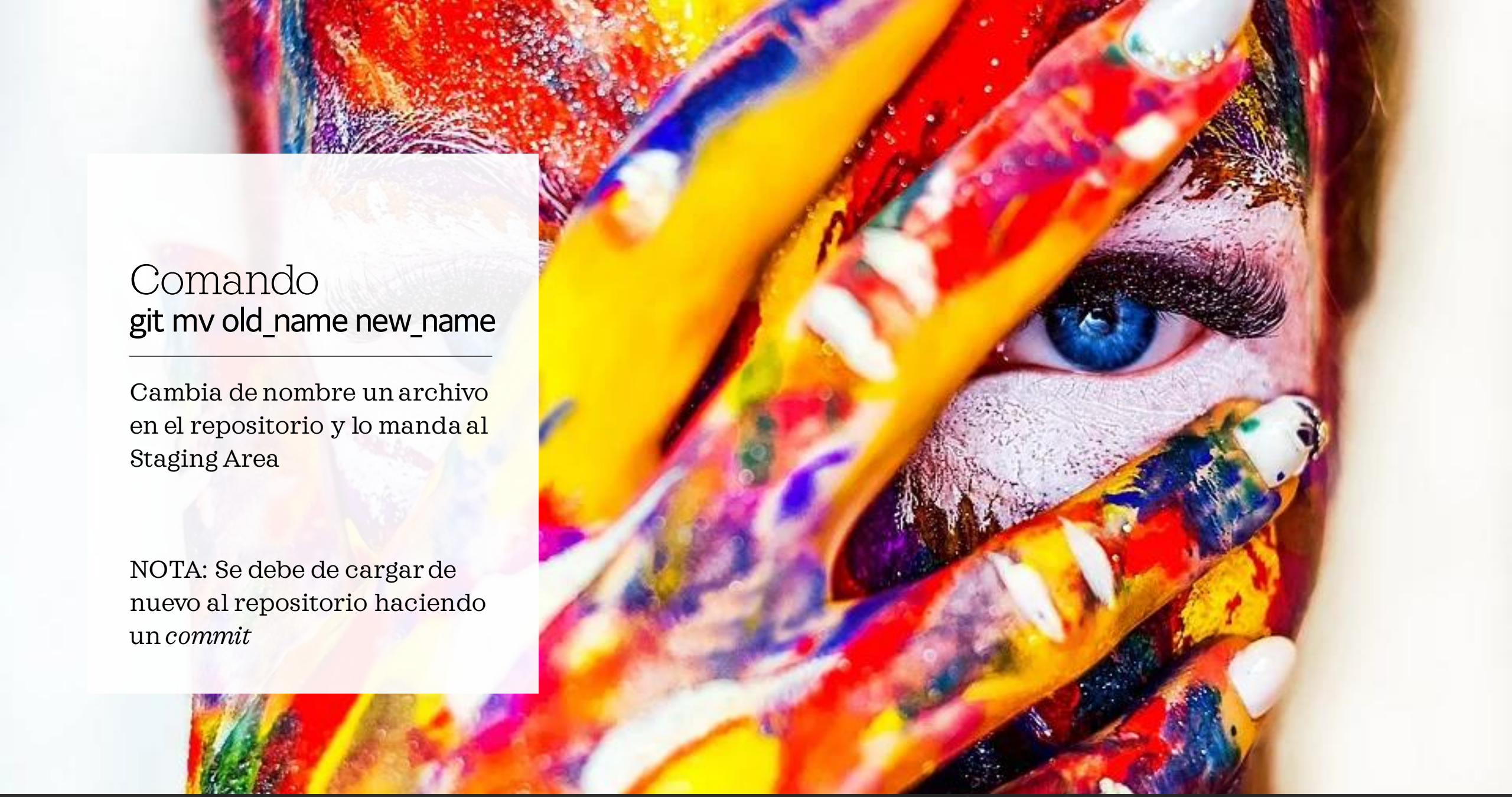
Comando **git diff**

- Muestra las modificaciones que hemos hecho a los archivos que están en el Working Area

  git diff

- Muestra las modificaciones que hemos hecho a los archivos que están en el Staging Area

  git diff --staged



Comando
git mv old_name new_name

Cambia de nombre un archivo en el repositorio y lo manda al Staging Area

NOTA: Se debe de cargar de nuevo al repositorio haciendo un *commit*

Comando
git revert commitID

Regresa un commit a su versión anterior más reciente

Actividad 4

- Modificaciones
- Renombrar archivos
- Reestablecer versiones

GitHub

SERVICIO DE HOSTING PARA REPOSITORIOS EN LA WEB

Actividad 5

- Crear una cuenta de GitHub

Clave SSH (Secure SHell)

- Proporciona acceso remoto a un servidor con toda la información cifrada.
- La usaremos para evitar escribir nuestras credenciales continuamente

Comandos a usar: Crear SSH

- Generar SSH:

```
ssh-keygen -t ed25519 -C "email@example.com"
```

- Instalar xclip (linux)

```
sudo apt-get install xclip -y
```

Actividad 6

- Crear una clave SSH

Agregar SSH a GitHub

macOS	pbcopy < ~/.ssh/id_ed25519.pub
GNU/Linux	xclip -sel clip < ~/.ssh/id_ed25519.pub
Windows	cat ~/.ssh/id_ed25519.pub | clip

Actividad 7

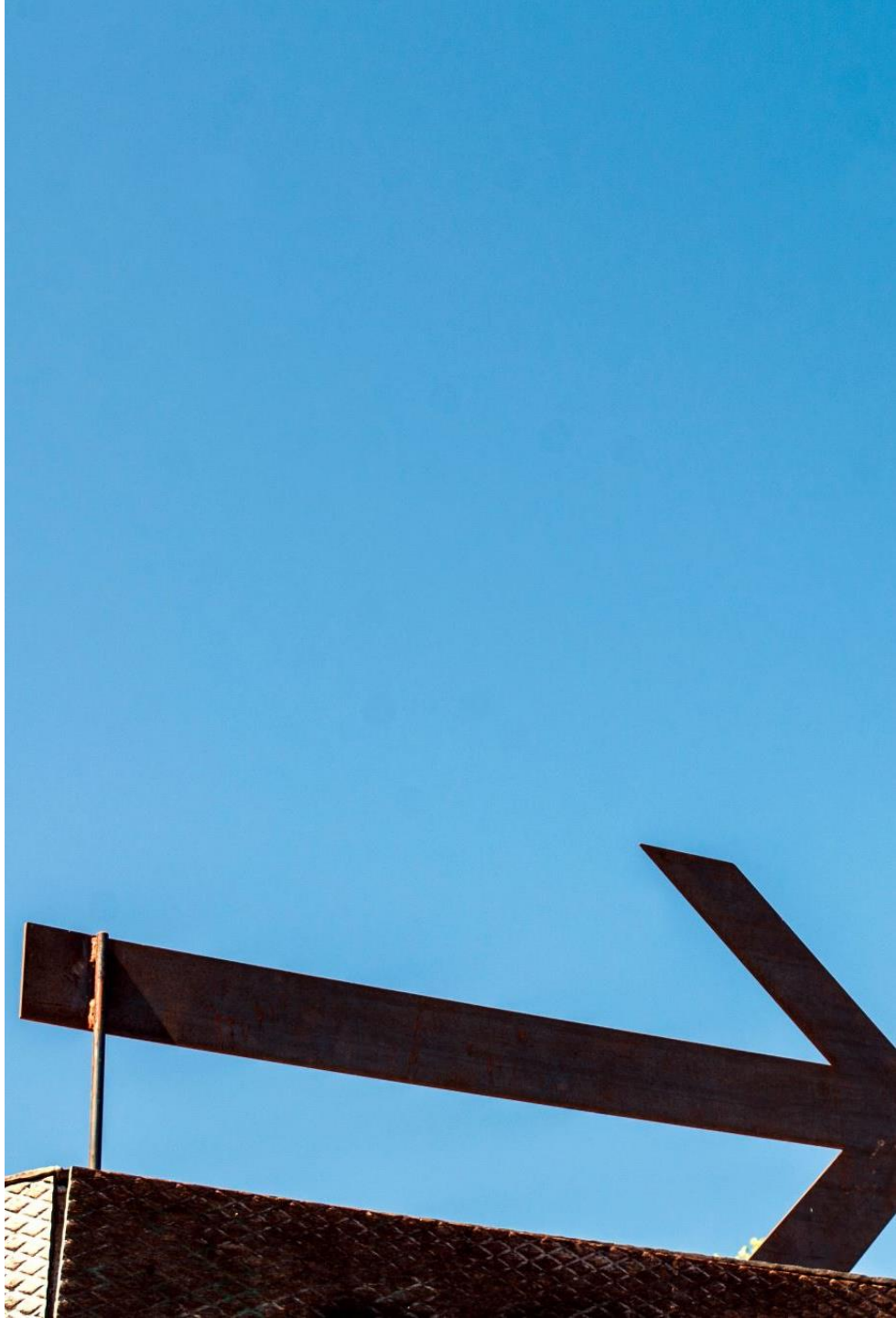
- Agregar la clave a GitHub

Repositorios en GitHub (Repos)

LUGAR DONDE SE ENCUENTRAN ALMACENADOS LOS ARCHIVOS

Remoto

Que no está en un local, cosas en GitHub



Tipos de repos en GitHub

- Privados: Nadie puede tener acceso a ellos

- Público: Cualquiera lo puede ver (pero no hacer cambios) y proponer cambios a el

Información extra de repos

Tienen un *link* único

Se localizan en un servidor de GitHub

Sólo el dueño del repo lo puede editar, pero podemos sugerir cambios

Se pueden seguir los repos

Actividad 8

- Crear un repo en GitHub

Comando git remote

SE USA PARA ENLISTAR, AGREGAR O ELIMINNAR REPOS REMOTOS

Comando
git remote add *name_of_remote_repo link_of_remote_repo*

1. Nombrar el repositorio remoto para distinguirlo, por convención es llamado *origin*
2. Enlazamos repo local y remoto con

git remote add origin *link_of_remote_repo*

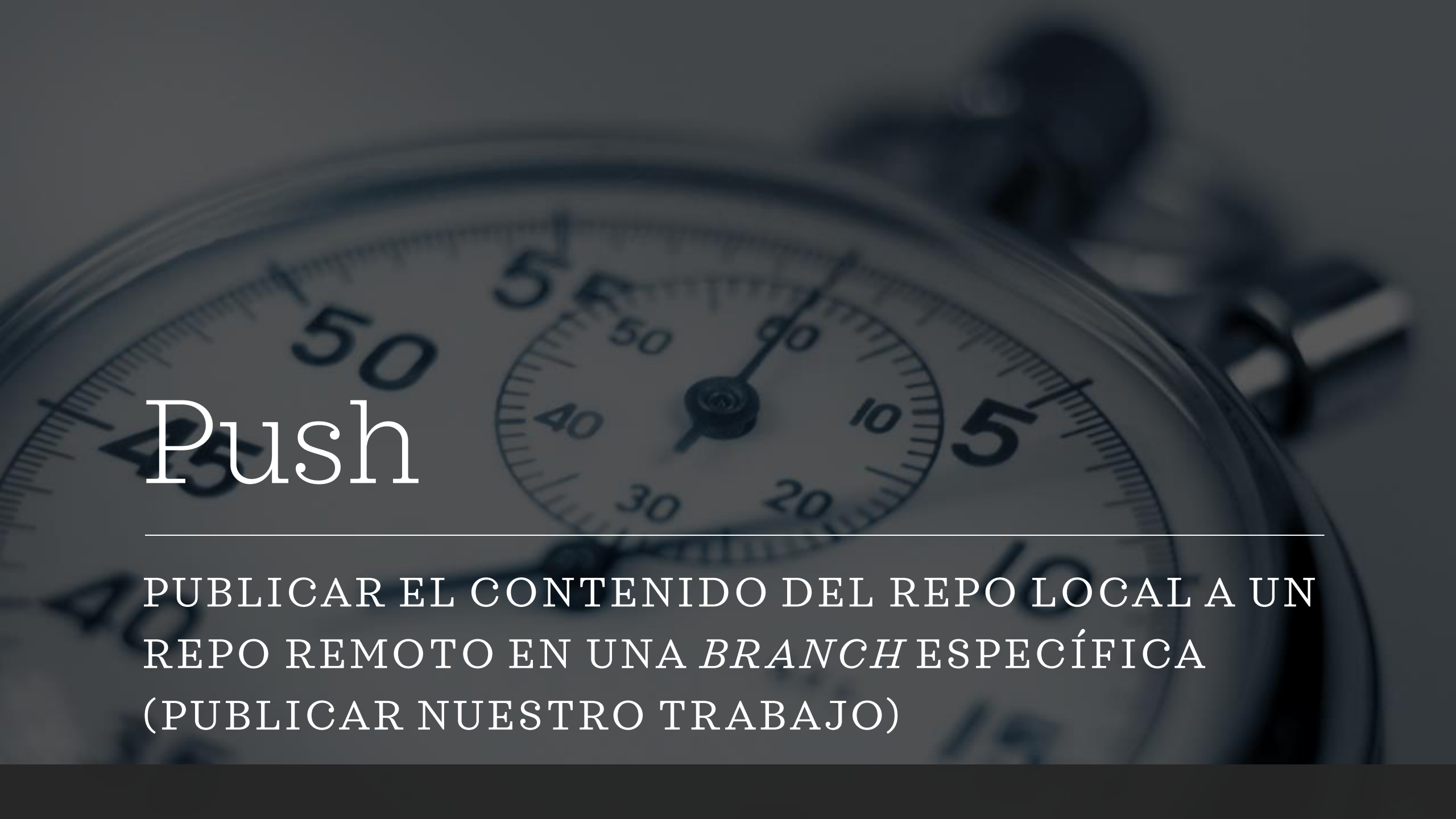
Actividad 9

- Enlazar repos

Fetch & Push

Fetch: URL que usa HTTP para acceso de sólo lectura

Push: URL que usa HTTPS ó SSH para el control de acceso



Push

PUBLICAR EL CONTENIDO DEL REPO LOCAL A UN REPO REMOTO EN UNA *BRANCH* ESPECÍFICA (PUBLICAR NUESTRO TRABAJO)

Comando

git push *remote_name branch_name*

Hace un *push* al repo y a la *branch* que se quiere

Actividad 10

- Hacer un push a nuestro repo

Issues

TAREAS Y COMENTARIOS EN GITHUB

Actividad 11

- Hacer una issue
- Clasificarla
- Asignarla

Resolver las issues

COMMITS Y COMANDOS DIRECTOS

Pasos

DECIDIR QUÉ ISSUE VAMOS A RESOLVER

PROGRAMAR LA SOLUCIÓN

SUBIR SOLUCIÓN

Forma del commit

1. Título
2. Mensaje
3. Referencia al issue
   #number_of_issue

NOTA: Las partes anteriores deben de ir separados por un salto de línea

Actividad 12

- Resolver una issue
- Hacer push

Branches

COPIA DEL PROYECTO DONDE SE PUEDE TRABAJAR SIN AFECTAR EL PRINCIPAL

Branch MASTER

- Primera branch que crea Git al momento de iniciar un repositorio

- Lugar donde se suben los cambios definitivos, aprovados y revisados

Usos de las Branch

Se crean para trabajar con las issues

Forma de trabajo temporal

Copiar un proyecto completo

NOTA: Los cambios en las branch sólo afectan a la branch en cuestión, no afectan otras partes del proyecto

Referencia HEAD

Referencia a la branch en la que estamos actualmente

Comando
git branch name

Crea una branch con nombre *name*

NOTA: Las branch aceptan valores alfanuméricos ó guiones. No aceptan espacios entre nombres

Comando **git branch**

Enlista las branches que tenemos en ese repositorio

Pone un aterisco en el HEAD (branch donde estamos actualmente)

Comando
git checkout name

Cambia de la branch en la que estamos actualmente a la branch con nombre *name*

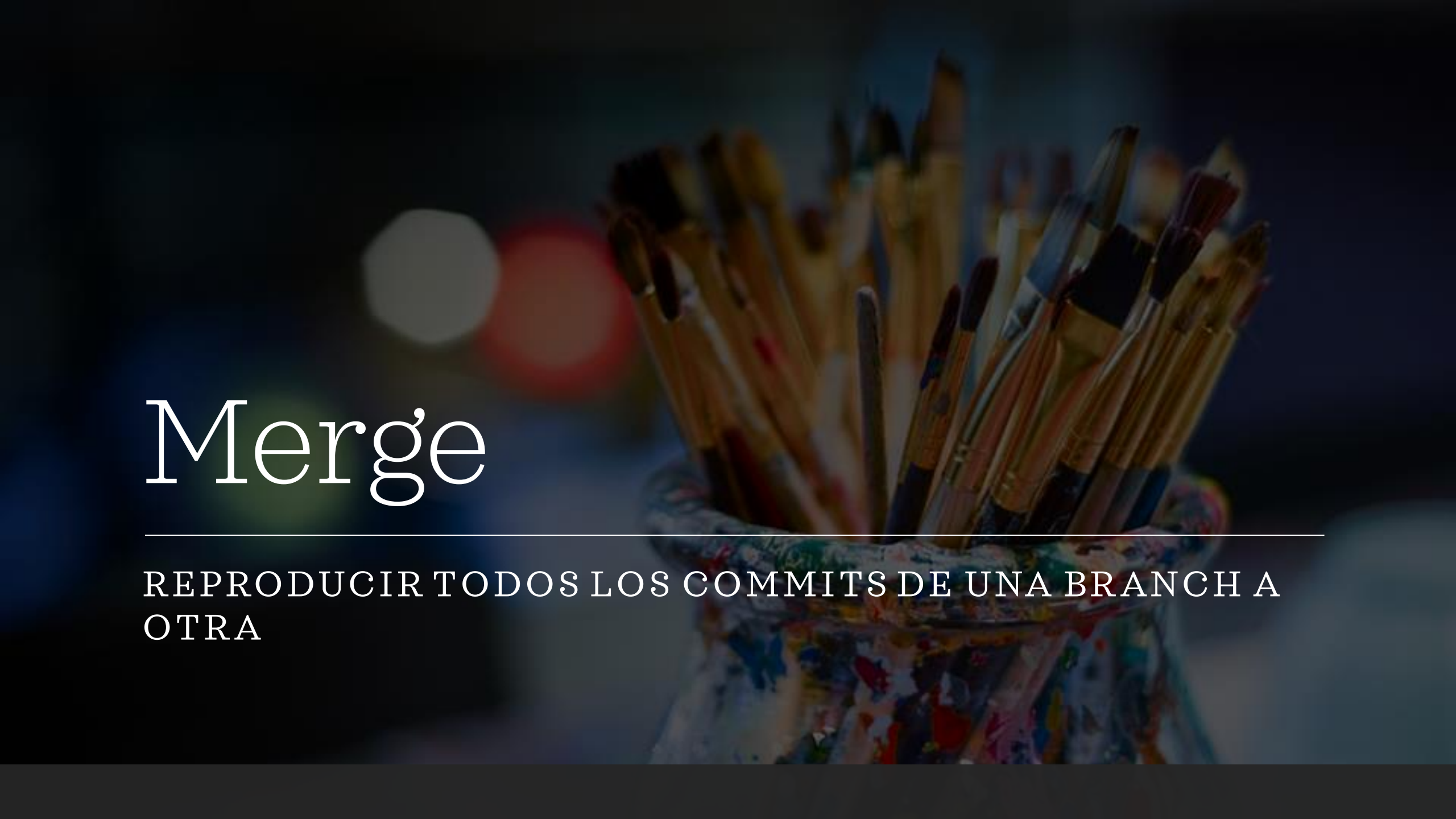
NOTA: No podemos cambiarnos de branch si no le hemos hecho *commit* a nuestros cambios en la branch actual

Comando
git branch –d name

Borra la branch con el nombre *name* desde cualquier branch en la que estemos

Actividad 13

- Crear una branch
- Ver referencias
- Interactuar con branches



Merge

REPRODUCIR TODOS LOS COMMITS DE UNA BRANCH A OTRA

Comando
git merge name

Hace un merge entre dos branches

- Reproduce los cambios de la branch *name* a la branch donde estemos ubicados

Eg.

Si estamos en la branch *X* y hacemos git merge *P*, esto copiaralos archivos de *P* a *X*

Pushing branches a GitHub

El comando para hacer un push a GitHub es

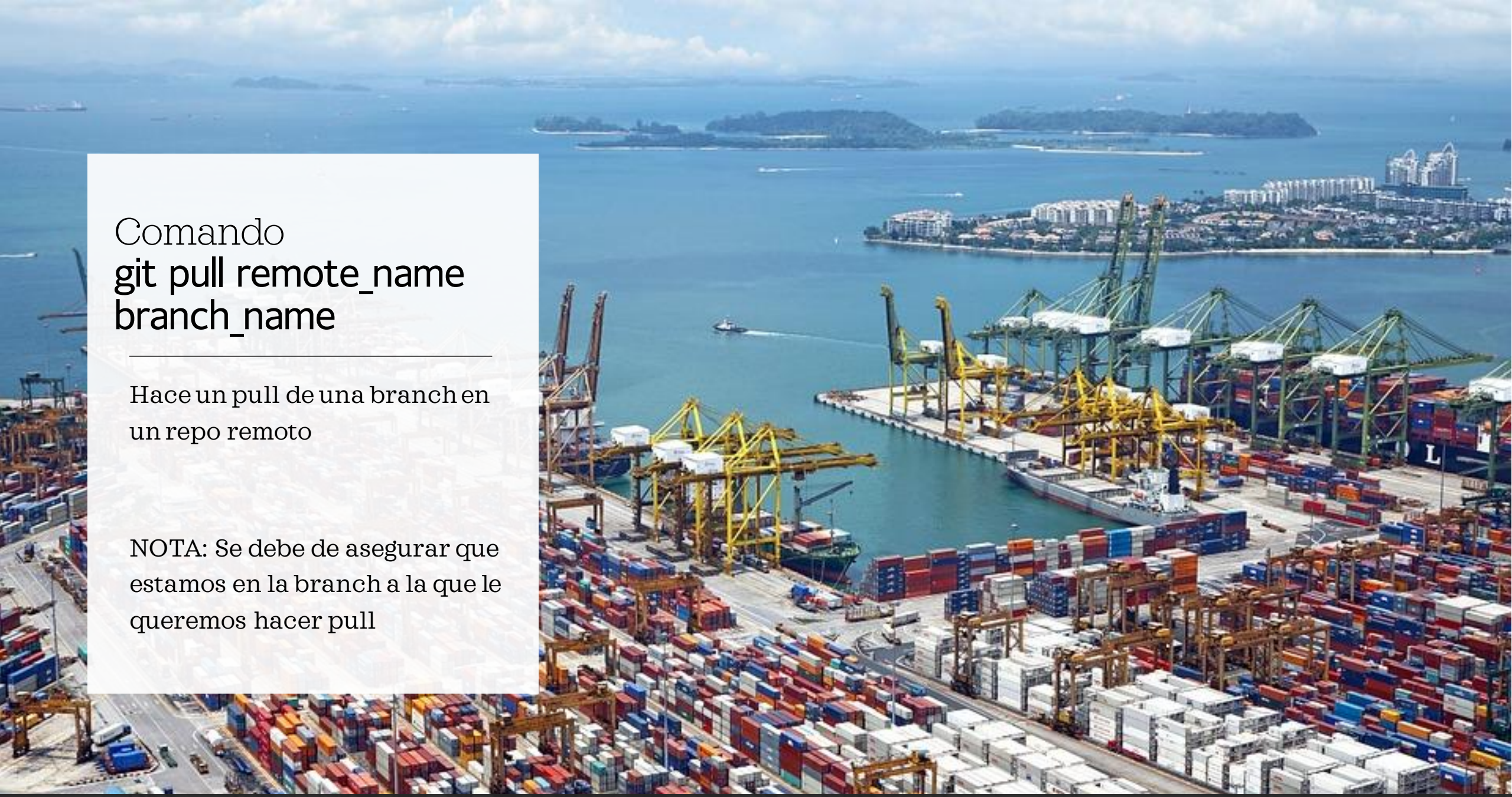
git push *remote_name branch_name*

Actividad 14

- Hacer merge entre branches
- Subir los cambios a GitHub

Pull

TOMA UNA BRANCH REMOTA Y COPIA LOS COMMITS EN UN REPO LOCAL



Comando

git pull remote_name branch_name

Hace un pull de una branch en un repo remoto

NOTA: Se debe de asegurar que estamos en la branch a la que le queremos hacer pull

Actividad 15

- Hacer pull