

Control de versiones + Base de datos



Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra: Desarrollo de Software



Contenidos

- **Control de Versiones**
 - Introducción al Control de versiones.
 - Introducción a GIT.
 - Repositorios.
 - Comandos básicos.
 - Actividad Git.
- **Base de Datos.**
 - Necesidad de una base de datos
 - Base de datos relacionales
 - SQLite
 - Actividad SQLite.



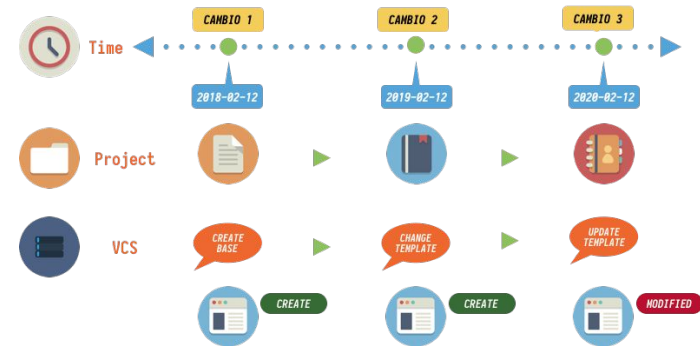
Introducción al Control de versiones (VCS):

- En la vida de cualquier **proyecto de software** llega el momento de tener que **colaborar con otros programadores**.
- **Problema:** incorporar los cambios de a poco sin romper el proyecto original.
- **Solución:** utilizar un **sistema de control de versiones**, VCS por sus siglas en inglés (Version Control System)



Introducción al Control de versiones (VCS):

- La idea de un VCS es almacenar todos los cambios que se realizaron sobre un conjunto de archivos, generando así un **historial de cambios**, donde se puede ver cada modificación realizada en los archivos, y llegado el caso volver a una versión previa.
- Hoy en día, Git es el sistema de control de versiones moderno más utilizado del mundo.
- Git es un proyecto de **código abierto** maduro y con un mantenimiento activo.



Introducción a Git: Evitando confusiones

Git es el VCS.

GitHub es un servicio muy popular que permite crear repositorios Git en sus servidores. Como también Gitlab, Gogs o Bitbucket.

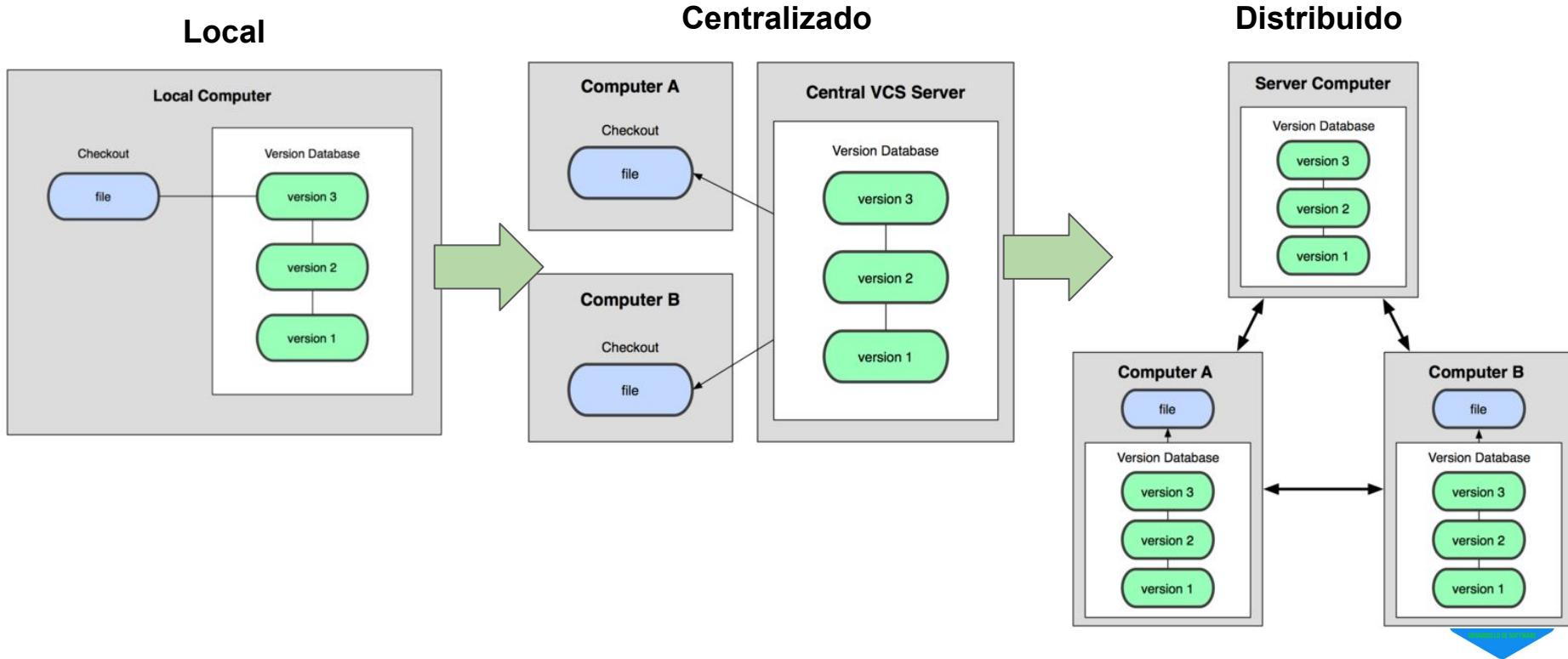


Introducción a Git: Características

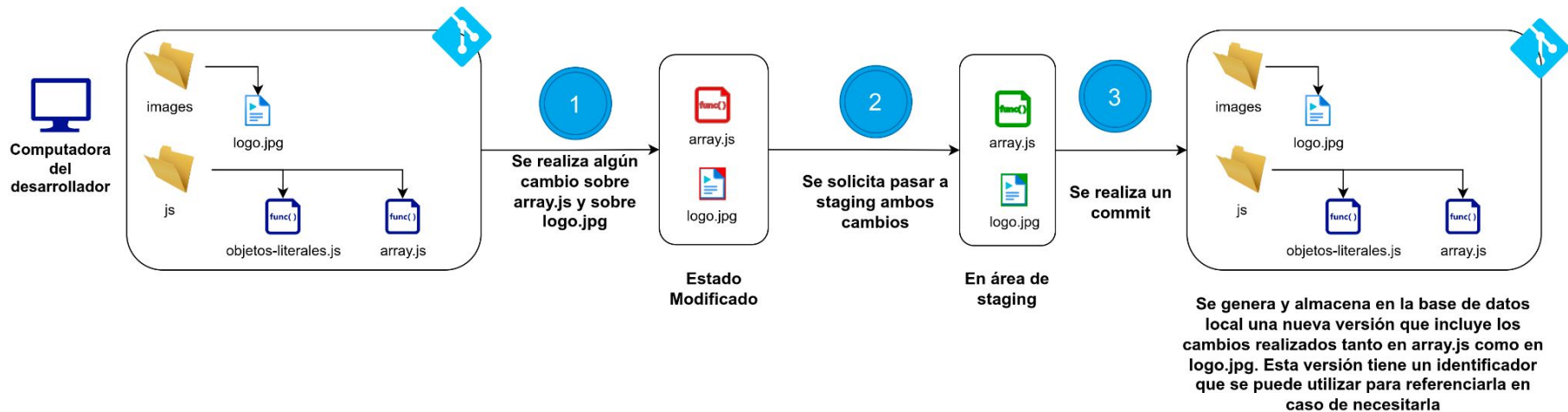
- Cada versión registrada en el tiempo es una revisión (commit).
- Cada operación se realiza en el repositorio local.
- Permite enviar cambios a repositorios remotos (como gitlab, github, etc).
- Es multiplataforma (windows, linux, macOS, etc).



Introducción a Git: Evolución



Introducción a Git: Estados



Introducción a Git: Repositorio

¿Qué es un repositorio?

- Un repositorio es un espacio que tenemos asignado para poder alojar todos los archivos de nuestro proyecto. Un repositorio es el lugar donde van a estar todos los commits que forman parte del historial del proyecto.

Introducción a Git: Repositorio local vs remoto

- GIT trabaja con un **repositorio local** que está en nuestro equipo, donde iremos agregando nuestros commits.
- También trabaja con un **repositorio remoto** en el cual podemos subir nuestros commits o del cual podemos bajarnos los commits que haya subido alguien



Local



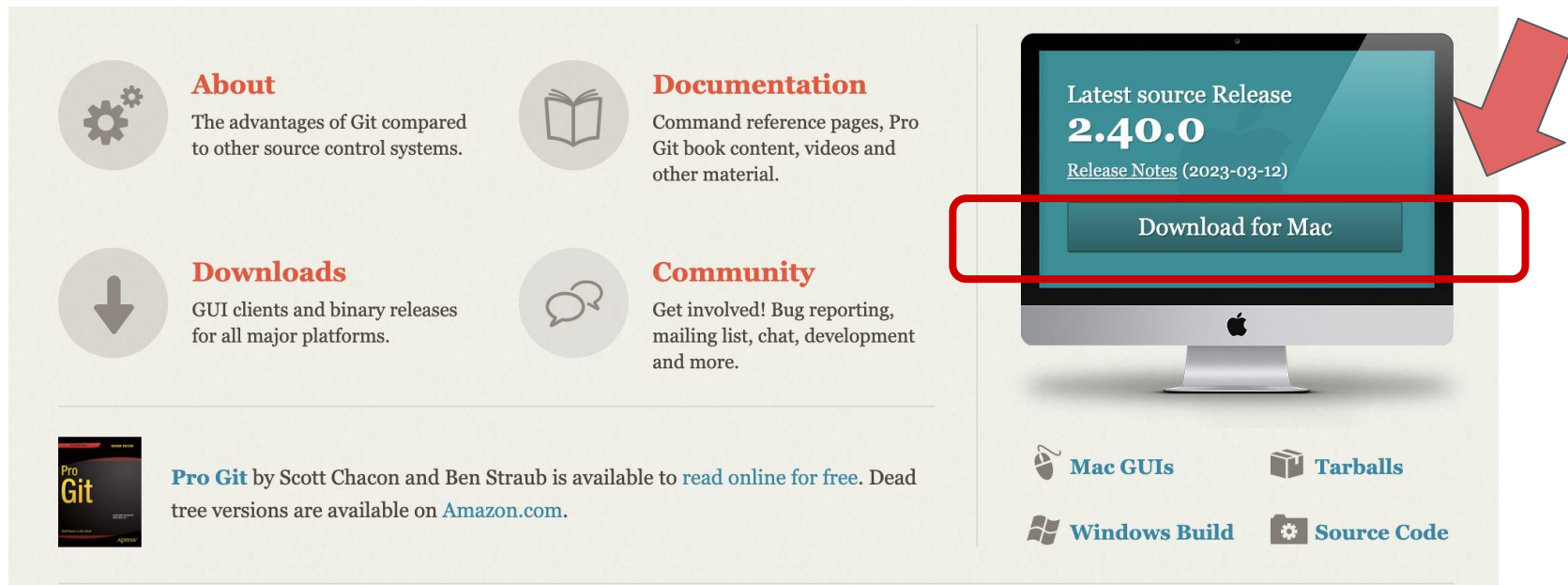
GitLab

Remoto

Comandos Git: Cuentas gitLab

- En el laboratorio se ha creado una cuenta para cada alumno para usar en el contexto de la asignatura:
 - Ejemplo: **22222@sistemas.frc.utn.edu.ar**
- Cada alumno ha recibido un correo en la bandeja del **correo institucional**.
- Para cambiar la contraseña pueden seleccionar olvidé contraseña en el login de gitLab:
 - **<https://labsys.frc.utn.edu.ar/gitlab/desarrollo-de-software1>**
- Ante algún problema en particular solo la configuración de la cuenta puedes contactar a la profesora a **nzea@frc.utn.edu.ar**

Comandos Git: Instalación Cliente Git



The image is a screenshot of the Git website's main page, annotated with red markings to highlight installation options. On the left, there are four circular icons with corresponding text: a gear for 'About', a book for 'Documentation', a downward arrow for 'Downloads', and two speech bubbles for 'Community'. The 'Downloads' section is highlighted with a red box. On the right, a computer monitor displays the 'Latest source Release 2.40.0' and a 'Download for Mac' button, which is also highlighted with a red box and a large red arrow pointing to it. Below the monitor, there are four links: 'Mac GUIs', 'Tarballs', 'Windows Build', and 'Source Code'. At the bottom left, there is a book cover for 'Pro Git' and a link to 'read online for free'.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Latest source Release
2.40.0
[Release Notes \(2023-03-12\)](#)

Download for Mac

Mac GUIs **Tarballs**
Windows Build **Source Code**

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



Comandos Git: Configuraciones globales

Luego de haber instalado algún cliente de GIT en nuestro equipo, debemos realizar mínimamente las siguientes dos configuraciones:

```
git config --global user.name "TU NOMBRE"  
git config --global user.email "TU DIRECCION DE EMAIL"
```



Para esto es necesario abrir un terminal de línea de comandos.

Comandos Git: Clonando Repositorios

Para esto se utiliza el comando git clone. Este comando tiene la siguiente sintaxis:

```
git clone {URL} {destino}
```

Donde *URL* es la url del repositorio, que puede ser de diferentes tipos dependiendo de donde esté el repositorio:

- Una dirección de un sitio web, usando protocolo https:
<https://labsys.frc.utn.edu.ar/gitlab/desarrollo-de-software1/proyectodds.git>
- Un repositorio accedido vía SSH, usando claves pública/privada para autenticación:
22222@labsys.frc.utn.edu.ar:gitlab/desarrollo-de-software1/proyectodds.git donde se indica usuario, servidor y ruta del repositorio usuario@servidor:/ruta.git
- Una carpeta en el sistema de archivos local: C:\Users\Usuario\Proyectos\vscode (Windows) o /home/usuario/proyectos/vscode (Linux). Se usa para copiar un repositorio ya creado o eliminado y pegarlo en la carpeta. Ejemplo: git clone /ruta/a/repositorio/original destino



Comandos Git: Status

```
# git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git restore <file>..." to discard changes in working directory)
```

```
    modified:   recetas.txt
```

```
Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
```

```
    ingredientes.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```



Comandos Git: Diff

Este formato se llama **unified diff**, y las líneas que tienen un signo menos adelante fueron eliminadas, y las que tienen un signo más fueron agregadas. Entonces se ve que se agregó una línea en blanco y otra que dice *Alguna receta*

```
# git diff
# git diff {archivo}

diff --git a/recetas.txt b/recetas.txt
index 55d5941..a6dfa2d 100644
--- a/recetas.txt
+++ b/recetas.txt
@@ -1,3 @@
-Recetas:
\ No newline at end of file
+Recetas:
+
+Alguna receta
\ No newline at end of file
```



Comandos Git: Creando commits

- Git Add: El comando git add se puede hacer archivo por archivo, o con el flag -u para agregar todos los archivos que han sido modificados.

```
# git add {archivo}
```

```
# git add .
```

Comandos Git: Creando commits

- **Add:** El comando git add se puede hacer archivo por archivo, o con el flag -u para agregar todos los archivos que han sido modificados.

```
# git add {archivo}  
  
# git add .
```

- **Commit:** cada commit queda en local hasta que se haga el push.

```
# git commit -a -m "[comentario]"
```

- **Push:** Subir todos los commits pendientes al Repositorio Remoto GitLab

```
# git push origin main
```



Comandos Git: Descargar commit remotos

- Descargar los commit que se encuentran en el servidor Git (en nuestro caso GitLab):

```
# git pull
```

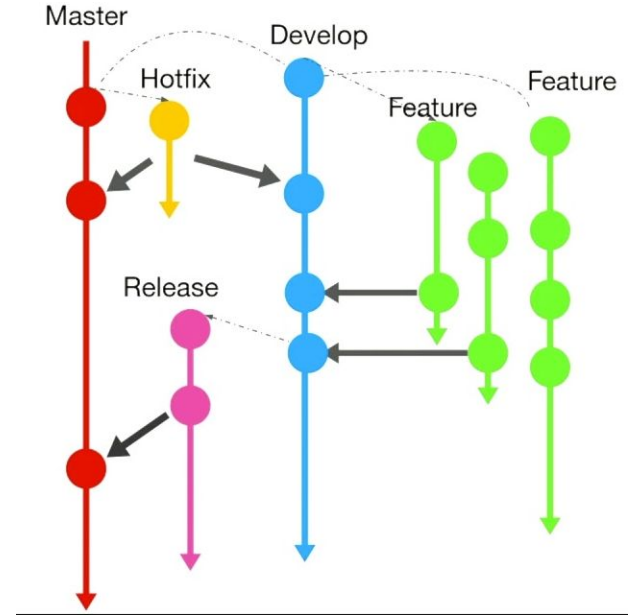
Comandos Git: Branches

1. Crear y cambiar de Branch (Rama)

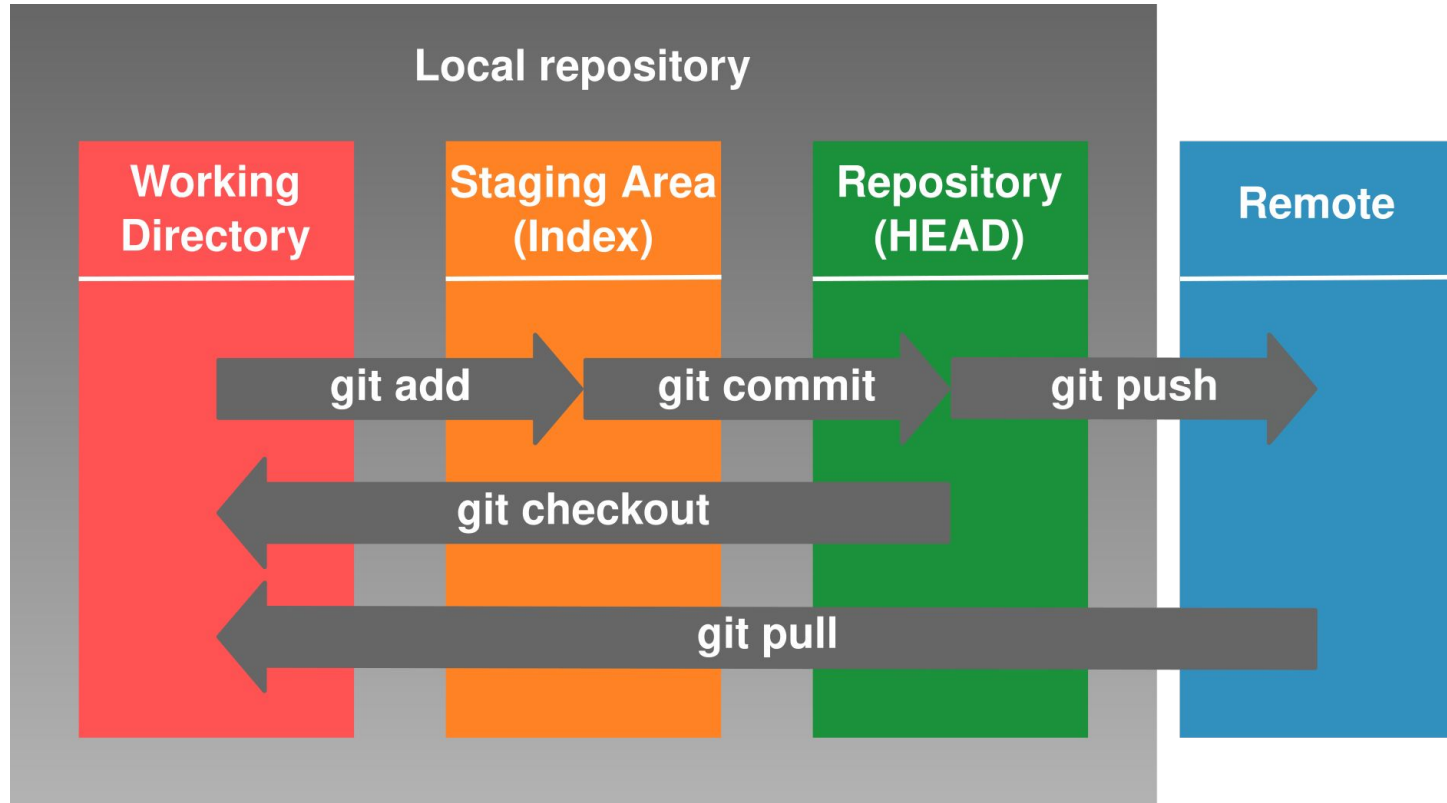
```
# git checkout -b [nombre de branch]
```

2. Cambiar de Branch (Rama)

```
# git checkout -b [nombre de branch]
```



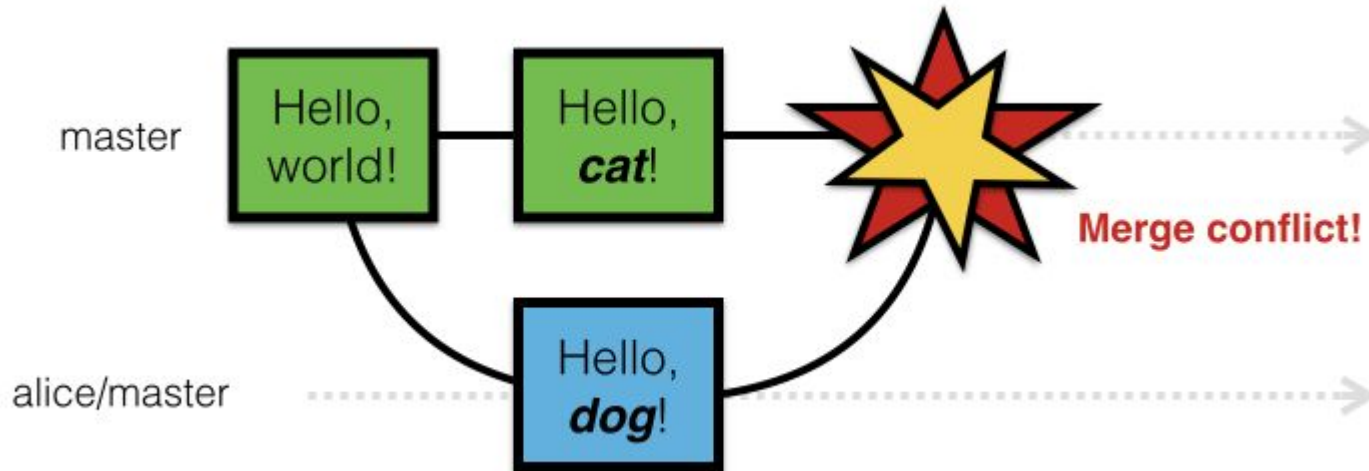
Comandos Git: Resumiendo!



Comandos Git: Merges

Comandos Git: Resolver Conflictos

- Dos usuarios que modifican un mismo archivo sobre una **misma rama**. El primero pusha los cambios y el segundo tiene un error de conflicto. Para lo cual debe decidir qué versión del archivo es la correcta.



Comandos Git: Conflictos

- Un mismo archivo es modificado en dos ramas, y ambas se intentan fusionar (merge) a una rama principal (develop o master).



Git - Comandos comunes

Ayuda memoria:

<https://docs.google.com/presentation/d/16aioHVVY3uLf-f8UZ3brU-icn1swMIVBW1Hc6OQMPfDU/edit?usp=sharing>

Para clonar un repositorio

- `git clone <url_repo>`

Para configurar nombre y mail que aparecerán en los commit

- `git config --local user.name "<nombre>"`
- `git config --local user.email "<email>"`

Para decirle a git que no pida nuestras credenciales en cada pull o push

- `git config --local credential.helper store`

Git - Comandos comunes

Para saber si hicimos cambios

- `git status`

Para actualizar nuestro repositorio local con lo que tiene el remoto:

- `git pull`
- `git pull --no-ff origin <rama>`

Para subir nuestros cambios (del repositorio local) al repositorio remoto:

- `git add .`
- `git commit -m "<mensaje>"`
- `git push -u origin <rama>`

Para cambiar/crear branch/rama

- `git checkout <rama>`
- `git checkout -b <rama>`

Para saber en qué rama estamos

- `git branch`

Para borrar una rama local

- `git branch -D <rama>`

Base de datos



Universidad Tecnológica Nacional
Facultad Regional Córdoba
Cátedra: Desarrollo de Software



Base de Datos: Necesidad de una base de datos

- En general, sea cual sea el propósito de una aplicación web, éstas necesitan manipular y almacenar datos.
- Los gestores de bases de datos (DBMS - Database Management System) nos permiten abstraernos de cómo los datos se almacenan físicamente, aseguran la integridad de los datos, permiten la concurrencia en el acceso a los datos, y nos permiten realizar operaciones con los datos de forma eficiente.
- Existen distintos tipos de bases de datos según la organización de los datos: bases de datos relacionales, jerárquicas, orientadas a objetos, NoSQL, documentales, etc.



Base de Datos: Bases de datos relacionales

- Las bases de datos relacionales (RDBMS - Relational Database Management System) son el tipo de base de datos más extendido y de propósito más general.
- Las bases de datos relacionales organizan los datos en forma de tablas y utilizan el lenguaje SQL (Structured Query Language) para la gestión y recuperación de los datos.

Base de Datos: SQLite

- En este curso vamos a utilizar **SQLite** (habitualmente pronunciado /'si: kwə 'lait/) que es un sistema de gestión de bases de datos relacional de código abierto escrito en lenguaje C.
- Este sistema tiene como ventajas la disponibilidad en cualquier sistema ya que no es cliente servidor como la mayoría de los DBMS y sin embargo tiene la mayoría de las características de un sistema de gestión de base de datos relacionales más robusto como puede ser MySQL, PostgreSQL u otros.
- Es muy utilizado en aplicaciones móviles y para actividades como testing ya que permite hasta disponer de bases de datos en memoria (sin persistencia física)



Actividad SQLite: Paso a paso SQLite

- https://labsys.frc.utn.edu.ar/gitlab/desarrollo-de-software1/semana-3/-/blob/main/clase06/pap_sqlite/README.md?plain=0

