

IPAP

Instituto Provincial de la Administración Pública

IPAP

SUBSECRETARÍA DE
GESTIÓN Y EMPLEO PÚBLICO

SECRETARÍA
GENERAL



GOBIERNO DE LA PROVINCIA DE
BUENOS AIRES

PROGRAMA DE FORMACIÓN GENERAL

CURSO VIRTUAL

GIT y herramientas para implementar el control de versiones en aplicaciones para desarrolladores/as

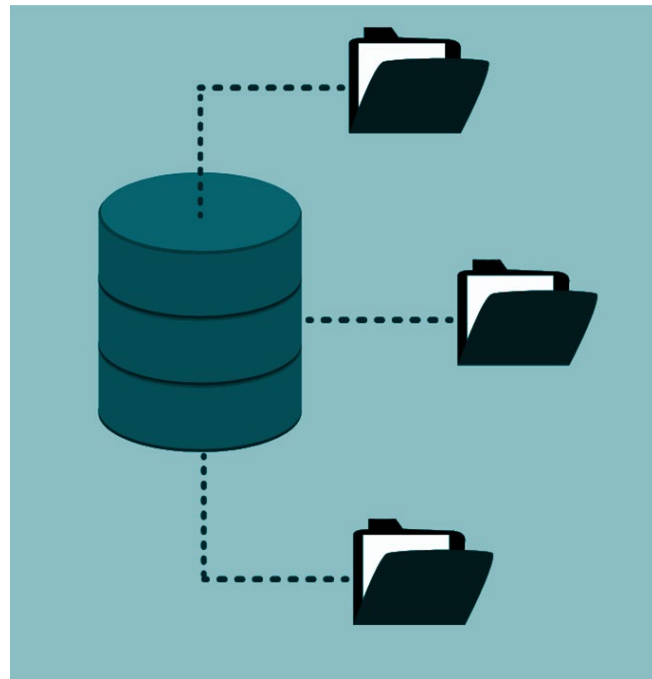
Docente: Agustín Parmisano Sabbione

Clase 2. Repositorios y cambios

1. Concepto de repositorio.
2. Crear, inicializar, asociar o clonar un repositorio.
3. Creación de repositorios desde Github y Gitlab.
4. Comandos básicos para cambios: commit, push, pull.
5. Buenas prácticas para trabajo en equipo: flujo de trabajo, escritura de títulos y descripciones.

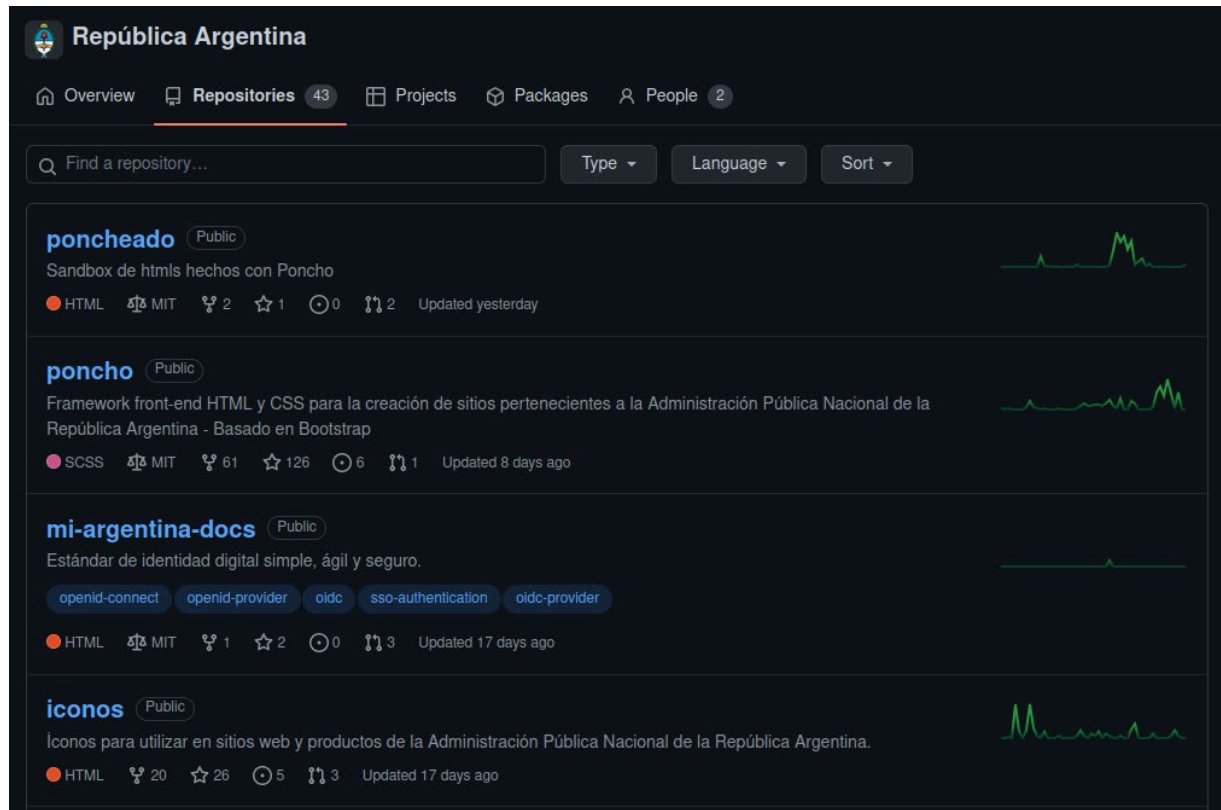
Concepto de repositorio

Un **repositorio** es un **espacio centralizado** donde se **almacena, organiza, mantiene y difunde información digital**, habitualmente **archivos**, que pueden contener **trabajos científicos, conjuntos de datos o software**. Los **repositorios** tienen sus inicios en los **años 90**, en el **área de la física y las matemáticas**, donde los académicos aprovecharon la **red** para **compartir** sus investigaciones con otros colegas. Este proceso era valioso porque **aceleraba el ciclo** científico de **publicación y revision** de resultados.



Concepto de repositorio

Ejemplo del **gestor de repositorios** en **GitHub** de la República Argentina donde se puede visualizar que al día de la fecha hay **creados 43 repositorios**. Para cada uno figura su **nombre**, **descripción**, **lenguajes de programación utilizados** mayormente, **licencias**, **forks** (copias) y **actividad reciente**.



Administración de repositorios

Crear un repositorio: en general la forma más sencilla y común de **crear** un **repositorio** es hacerlo mediante la **interfaz gráfica del cliente web de nuestro gestor de repositorios (Github, Gitlab o Bitbucket)** que veremos más adelante.

Inicializar un repositorio: también es posible **inicializar** un **repositorio** desde cualquier lugar del **sistema de archivos de nuestro Sistema Operativo** con el comando ***git init*** . Este comando **inicializa** un **repositorio git** con los **directorios** y **archivos** en el **directorio de trabajo actual** como participantes del repositorio.

Asociar o clonar un repositorio: otra alternativa es **clonar** un **repositorio propio** o de **terceros** mediante el comando ***git clone <nombre_repositorio>*** . Al clonar un repositorio **nos descargamos todos los archivos de todas las versiones en nuestro SO.**


Creación de repositorios desde Github


<> Start writing code

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

AgustinParmisano /

☐  **Public**
Anyone on the internet can see this repository

☒  **Private**
You choose who can see and commit to this repository

Create a new repository

1

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *  AgustinParmisano /

Repository name *

Great repository names are short and memorable. Need inspiration? How about [vigilant-guide?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

 You are creating a public repository in your personal account.

Create repository

Creación de repositorios desde Gitlab



Projects

Filter by name

Name

Create new project



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.



Create from template

Create a project pre-populated with the necessary files to get you started quickly.



Import project

Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.



Run CI/CD for external repository

Connect your external repository to GitLab CI/CD.



Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

New project > Create blank project

Project name

My awesome project

Project URL

https://gitlab.com/ Pick a group or namespace

Project slug

my-awesome-project

Want to organize several dependent projects under the same namespace? [Create a group](#).

Project deployment target (optional)

Select the deployment target

Visibility Level

☒ Private

Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Public

The project can be accessed without any authentication.

Project Configuration

☒ Initialize repository with a README

Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)

Analyze your source code for known security vulnerabilities. [Learn more](#).

Create project

Cancel

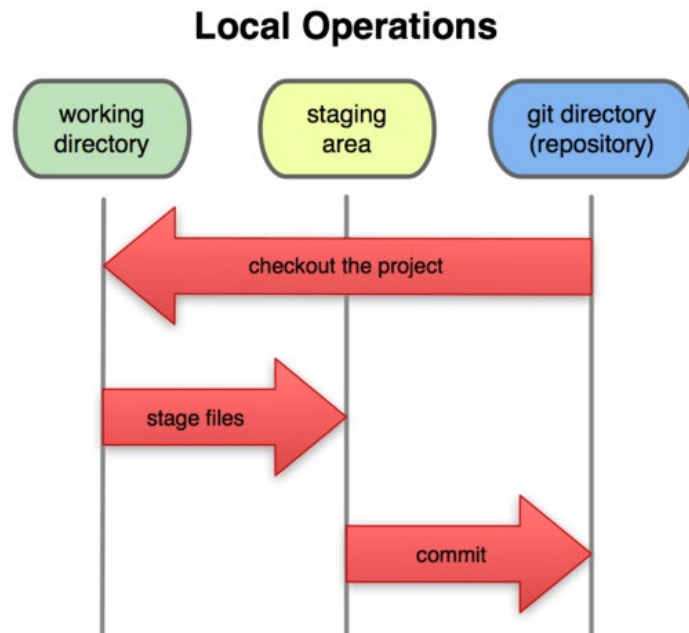
Gestión de cambios: áreas

En **git** existen dos áreas: el área local y el área remota.

En la **copia local de Git**, los **archivos** pueden estar en distintos **estados**:

- **En el repo local:** committed (**cambios agregados** para **nueva versión**).
- **Modificados y checked out pero no comiteados:** working copy (archivos **cambiados** pero **no agregados** para nueva versión).
- **En el medio de estos dos**, en la **staging area**: los **archivos** staged están **listos** para ser **comiteados**.

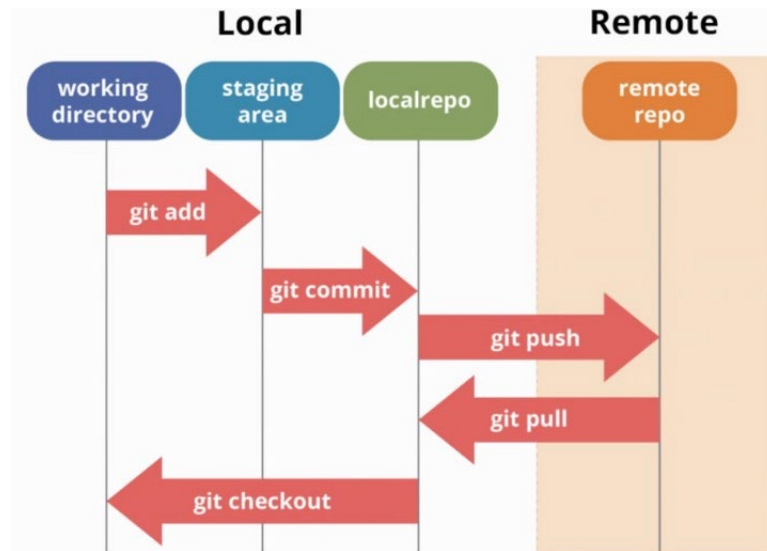
Un commit guarda una instantánea de todos los archivos en estado staged.



Gestión de cambios: áreas

Una vez **creada** una **nueva versión local** con un **commit** es posible continuar trabajando nuevas versiones localmente y/o **persistir** todas las **versiones** creadas al **repositorio remoto** con un **push**.

A su vez si el **repositorio remoto** tiene nuevas **versiones** que **no tenemos localmente** deberemos versionar nuestro trabajo (**commit**) para **traernos las versiones remotas, compararlas** y en caso de **incompatibilidades** (conflicts), **mezclarlas** (**merge**) resolviendo los conflictos manualmente.



***Nota:** todas las configuraciones, versiones y estado de los archivos locales se guardan en un archivo `.git` oculto en nuestro repositorio*

Gestión de cambios: comandos

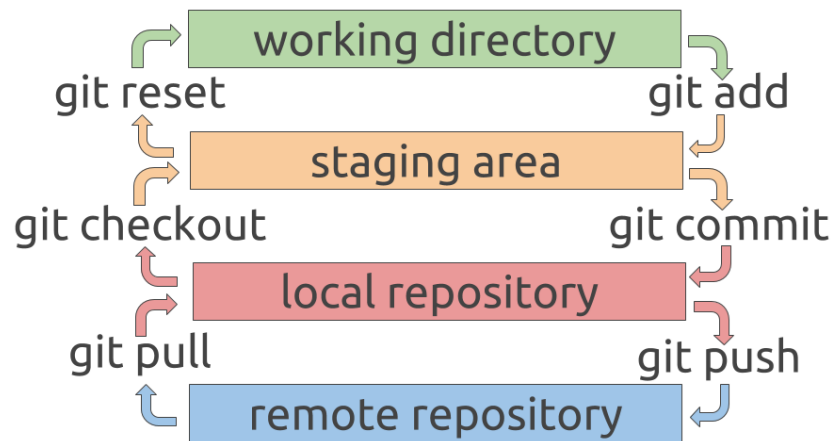
add: comando para **agregar** uno o más **archivos/directorios** del **directorio de trabajo (WD)** a la **staging área (SA)**.

status: comando para **visualizar** el **estado** de los **archivos** en el **WD** y la **SA**.

commit: comando para **crear** una **nueva versión** en el **repositorio local (LR)**. Es preciso **agregar** un **comentario** que **ayude** a **comprender** la **versión**.

push: comando para **subir** el **LR** con **todas** las **versiones locales** al **repositorio remoto (RR)**.

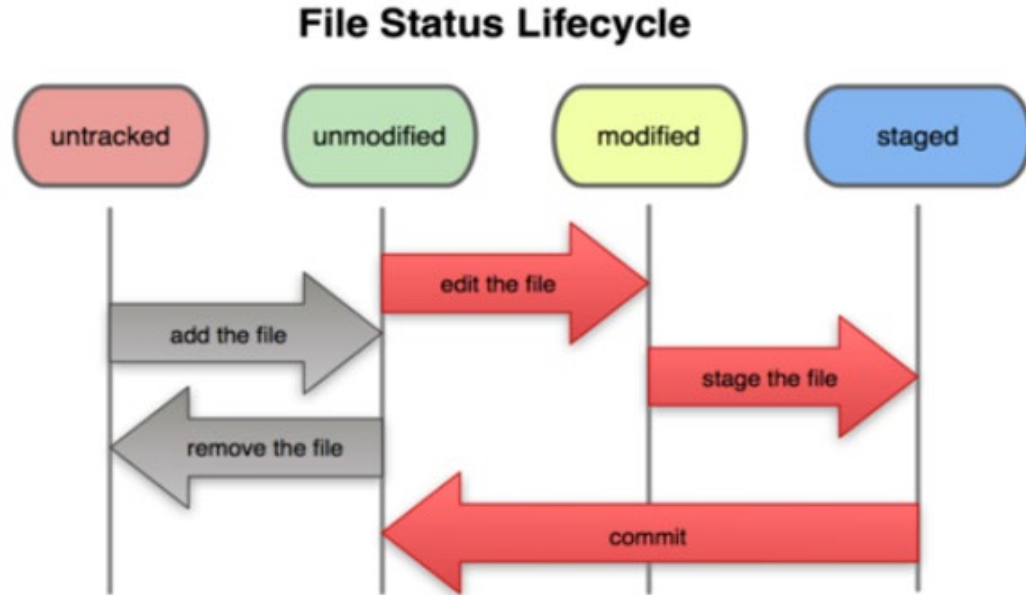
pull: comando para **descargar** **todas** las **versiones** del **RR** al **LR**.



Buenas prácticas para trabajo en equipo

Flujo de trabajo básico/normal:

- Modificar archivos en el directorio de trabajo
- Stage archivos (add), agregando instantánea de éstos en el área de staging.
- Commit, que guarda todos los archivos en el área de staging de forma permanente en el directorio Git (local).



Buenas prácticas para trabajo en equipo

Flujo de trabajo: Por lo general **antes** de **empezar** a **trabajar** en nuestro **repositorio local** es bueno **verificar** si tenemos archivos fuera del área de **staging** (sin agregar), **agregarlos** y **commitarlos** para luego hacer un **pull** en caso de que **existan versiones nuevas remotas**. A su vez es **buena práctica** **guardar**, **commitear** y **pushear** nuestro trabajo **antes** de **terminar la jornada**.

Escritura de títulos, descripciones y comentarios: es también **buena práctica** ser **descriptivo** con los **títulos** y los **comentarios** de nuestros **commits/versiones** y de nuestro **código**.

Escribir un buen README.md: el archivo **README.md** es el que **describe** nuestro **repositorio** y como se **utiliza**, es **muy buena práctica** tener un **README descriptivo y actualizado**.

In case of fire



1. git commit
2. git push
3. exit building

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAHAHAHAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Buenas prácticas para trabajo en equipo

Laboratorio 1:

1. Crear un repositorio en GitHub o GitLab con un README.md con la descripción del repositorio.
2. Clonar nuestro nuevo repositorio en nuestro SO local.
3. Modificar el README.md agregando este enunciado. Se recomienda utilizar dillinger.io para escribir en lenguaje de marcado [Markdown](#) correctamente.
4. Utilizar los comandos vistos como status, add, commit (crear un buen comentario descriptivo).
5. Subir el archivo al repositorio remoto.

Pregunta/Debate

Para comentar en el foro

¿Qué tan importante es escribir comentarios en los commits? ¿Por qué?

¿Por qué crees que es importante tener copias en repositorios remotos?

¿Cuál crees que es la ventaja de tener todas las versiones en una copia local?

ipap.gba.gob.ar

IPAP

SUBSECRETARÍA DE
GESTIÓN Y EMPLEO PÚBLICO

SECRETARÍA
GENERAL



GOBIERNO DE LA PROVINCIA DE
BUENOS AIRES