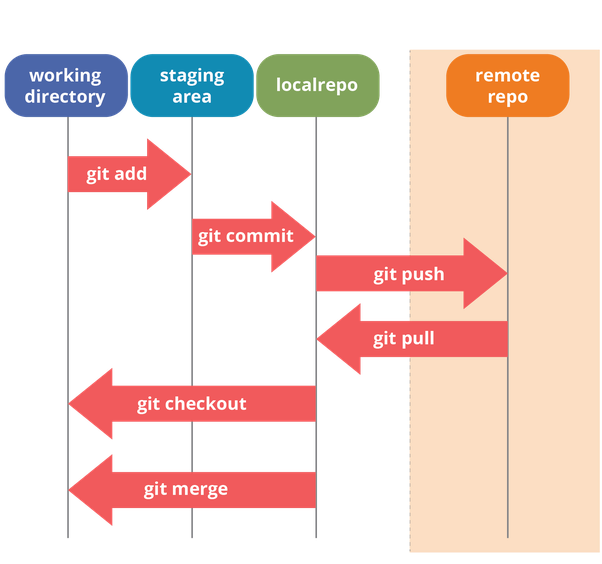
TP2\_GIT



Abri una cuenta el gitlab o github.

Genera un proyecto tp\_Python.

Instalarlo en tu pc git.

Haz una carpeta tp\_Python en tu disco local o servidor local .

Coloca solo este archivo alli (Ahora, antes de modificarlo).

Inicializa git.

coloca tus datos.

Agrega este archivo.

Haz un commit con tu nombre como comentario.

Haz un push a tu servidor. Con el archivo antes de modificarlo.

Modifica este archivo con lo que se te pide debajo.

* ¿Que es git?
* ¿Que es github?
* ¿Que es gitlab?
* Diferencias
* ¿Como se usa y cual es su propósito general?
* ¿Quienes pueden usar sus datos?
* ¿Que proyectos colocarias en github y cuales en gitlab?¿Porque?
* ¿Cuales son sus principales comandos?¿ como se usan?

Guarda este archivo con los datos que agregaste.

Actualiza este archivo.

Haz un commit con tu nombre como comentario.

Haz un push a tu servidor. Con el archivo ya modificarlo.

Renombra este archivo.

Haz un pull y merge donde bajaras el archivo con el nombre viejo.

Elijan un proyecto en github o gitlab entre los alumnos.

Suban todos su aporte a ese proyecto y bajen el de los demas, asi pueden tener en local el trabajo de todos.

Editen un archivo con las mejores datos de todos los tp.

Subanlo y hagan un pull request.

Borren sus aportes personales en lo local y en sus proyectos y actualícelo con el archivo generado y mejorado por todo el grupo.

* ¿Que utilidad encontras git?
* ¿Que ventajas y desventajas tiene usar git como sistema de seguridad?
* ¿Que ventajas y desventajas tiene usar git como sistema de trabajo en grupo?

Nota: Ten en cuenta el espacio en disco

Cualquier consulta comuniquense a

[cursos.agt@gmail.com](mailto:cursos.agt@gmail.com) o <https://github.com/CursosAGT>

1. Git es un sistema de control de versiones, un sistema que en si funciona para mantener un control y revisión de un proyecto es sus distintas etapas de desarrollo. De esta forma, en caso de que algo no llegue a funcionar, puedes usar una versión anterior para ver en que esta fallando. Y también puedes ramificar un proyecto y al final unirlo
2. Github es una… dicho en criollo, una plataforma estilo Facebook para desarrolladores, donde uno puede postear su proyecto y asi obtener colaboraciones o también para buscar algún proyecto que facilite la creación de uno que tengas en mente.
3. Este, es prácticamente un GitHub, pero OpenSource.
   * GitLab tiene un software que permite montar un servidor propio, donde puedes

evitar que se filtren tus códigos como lo haría estando en el servidor comunitario.

* + GitHub tiene una comunidad mucho más amplia.
  + GitLab es de código abierto.
  + GitHub es de Microsoft.

1. GitHub y GitLab son herramientas basadas en Git. Generalmente usas Git a través de la línea de comandos, o a través de programas de interfaz gráfica, o del propio editor de código.

Además del hosting remoto para repositorios, estas plataformas ofrecen una interfaz web para controlar el repositorio y muchas otras herramientas. Ofrece la posibilidad de examinar el código en cualquiera de sus versiones, realizar acciones relacionadas con el sistema de repositorios como mergear el código de versiones de proyecto o gestionar las "pull request", gestionar problemática de tu software diversa, automatizar procesos como el despliegue o la ejecución de pruebas del software, etc. Toda esta operativa la realizas, o configuras, por medio de una web.

Por tanto, para usar GitLab o Github simplemente necesitas las mismas herramientas que ya utilizas en tu día a día, el terminal o un programa de interfaz gráfica para gestionar tu repositorio, así como el navegador web para acceder a el ecosistema de herramientas disponible en el sitio de GitLab. Por supuesto, todas estas herramientas las puedes usar desde cualquier ordenador conectado a Internet, independientemente de su sistema operativo.

1. Toda persona que este agregada por el administrador del proyecto si es privada, si es publica toda persona tiene acceso a el proyecto pero no su edición.
   * **Github** lo usaría si es para crear un código abierto, donde los usuarios más del estilo tecnológico sean quienes opinen. Además, si quiero que tenga una contribución más rápida o en más abundancia, ya que es la plataforma con más usuarios hasta el día de la fecha.
   * Por otro lado, **GitLab** los usaría en caso de que mi código necesite cierto conocimiento más amplio o sienta que la comunidad sea más propensa a poseer conocimientos científicos. También en caso de que si en cierto punto, no sería muy conveniente que mi código lo conozca todo el mundo, usaría la herramienta de crear mi propio servidor cerrado de GitLab para mí y mi empresa.
2. * **git help**

**Muestra una lista con los comandos más utilizados en GIT.**

* + **git init**

**Podemos ejecutar ese comando para crear localmente un repositorio con GIT y así utilizar todo el funcionamiento que GIT ofrece. Basta con estar ubicados dentro de la carpeta donde tenemos nuestro proyecto y ejecutar el comando. Cuando agreguemos archivos y un commit, se va a crear el branch master por defecto.**

* + **git add + path**

**Agrega al repositorio los archivos que indiquemos.**

* + **git add -A**

**Agrega al repositorio TODOS los archivos y carpetas que estén en nuestro proyecto, los cuales GIT no está siguiendo.**

* + **git commit -m "mensaje" + archivos**

**Hace commit a los archivos que indiquemos, de esta manera quedan guardados nuestras modificaciones.**

* + **git commit -am "mensaje"**

**Hace commit de los archivos que han sido modificados y GIT los está siguiendo.**

* + **git checkout -b NombreDeBranch**

**Crea un nuevo branch y automaticamente GIT se cambia al branch creado, clonando el branch desde donde ejecutamos el comando.**

* + **git branch**

**Nos muestra una lista de los branches que existen en nuestro repositorio.**

* + **git checkout NombreDeBranch**

**Sirve para moverse entre branches, en este caso vamos al branch que indicamos en el comando.**

* + **git merge NombreDeBranch**

**Hace un merge entre dos branches, en este caso la dirección del merge sería entre el branch que indiquemos en el comando, y el branch donde estémos ubicados.**

* + **git status**

**Nos indica el estado del repositorio, por ejemplo cuales están modificados, cuales no están siendo seguidos por GIT, entre otras características.**

* + **git clone URL/name.git NombreProyecto**

**Clona un proyecto de git en la carpeta NombreProyecto.**

* + **git push origin NombreDeBranch**

**Luego de que hicimos un git commit, si estamos trabajando remotamente, este comando va a subir los archivos al repositorio remoto, específicamente al branch que indiquemos.**

* + **git pull origin NombreDeBranch**

**Hace una actualización en nuestro branch local, desde un branch remoto que indicamos en el comando.**

* **git remote add orirgin** *URL*

Agrega una dirección a donde vas a mandar los archivos

* **git push –origin master**

Envia los archivos a la plataforma.

* **git checkout master**

**git merge origin/master --allow-unrelated-histories**

No se la verdad, pero me sirvieron para arreglar el error de “fatal: refusing to merge unrelated histories”

* **pdw**

Con este se en que sitio estoy parado