# Algoritmos y Estructuras de Datos I

Introducción a Git

8 de mayo de 2020

- Sistema de control de versiones distribuido[1], orientado a repositorios[2] y con énfasis en la eficiencia.
  1. Se tiene un servidor que permite el intercambio de los repositorios entre los usuarios.
  2. Cada usuario tiene una copia local[3] del repositorio completo.

[1] podemos trabajar en un mismo proyecto desde distintas computadoras
[2] un repositorio tiene el historial de cambios y versiones de un proyecto
[3] cada quien tiene que hacer su propia copia del repositorio

# Git: Comandos[7]

- Copiamos un repositorio existente en un servidor a nuestro directorio de trabajo:
  `git clone (url)`[4]
- Creamos un archivo y lo agregamos a la *staging area*[5]:
  `git add (archivo)`
- Hacemos commit de los cambios al repositorio local, es decir confirmamos que los cambios que agregamos con *add* hasta el momento son los que voy a hacer efectivos:
  `git commit -m (mensaje)` [6]
- Enviamos todos los cambios confirmados hasta el último *commit* al servidor:
  `git push`
- Aplicamos los cambios del servidor en nuestra *copia local*:
  `git pull`

[4] La direccion la pueden obtener de la pagina del repositorio en el boton "clone"
[5] Los archivos que efectivamente queremos modificar en el repositorio
[6] Mensaje es un resumen de los cambios escrito entre comillas
[7] Estos comandos se deben utilizar en la terminal ubicandose en el directorio del repositorio usando el comando cd

# Git: Mas Comandos

- Consultas sobre el estado actual del repositorio:
  `git status`[8]
  `git diff`
- Borrar un archivo controlado por Git:
  `git rm (archivo)`
- Mover o renombrar un archivo controlado por Git:
  `git mv (archivo) (nuevo)`

[8] Podemos ver que archivos se modificaron, agregaron o eliminaron desde el último commit/pull y los archivos que faltan agregar al repositorio en el directorio local

# Otros conceptos

- **Tag:** Nombre asignado a una versión particular, habitualmente para *releases* de versiones a usuarios.
- **Branch:** Línea paralela de desarrollo, para corregir un bug, trabajar en una nueva versión o experimentar con el código.[9]

[9] Por ejemplo, si queremos probar alguna modificación en el código sin copiar todo, podemos hacerlo en otro branch

# Consejos

- Hacer commits pequeños y puntuales, con la mayor frecuencia posible.[10]
- Mantener actualizada la copia local del repositorio, para estar sincronizados con el resto del equipo.[11]
- Commitear los archivos fuente, nunca los archivos derivados![12]
- Manejar inmediatamente los conflictos[13].

[10] PERO no hacer commits con código que no compila

[11] Hacer git pull antes de ponerse a trabajar en los archivos de un repositorio es una buena práctica.

[12] ej: .tex y no .pdf, .cpp y no su ejecutable asociado

[13] Un conflicto se genera cuando dos personas modifican lo mismo y git no sabe que cambio aplicar, debemos decidir "manualmente" que cambio queremos conservar y hacer el *push* correspondiente

# Links útiles

- **Repos hosts**
  - GitHub: `https://github.com`[14]
  - GitLab Exactas: `https://git.exactas.uba.ar`[15]

- **Bibliografía**
  - **Git - la guía sencilla**:
    `http://rogerdudler.github.io/git-guide/index.es.html`
  - **Pro Git book**:
    `https://git-scm.com/book/en/v2`
  - **Try git**:
    `https://try.github.io`
  - **Ejercicios interactivos de branches (EN ESPAÑOL)**
    `https://learngitbranching.js.org/?locale=es_AR`

[14] página de repositorios mas importante actualmente
[15] página de repositorios de la facultad, deben tener cuenta @dc.uba.ar para acceder