UNIVERSIDAD DE LOS LAGOS

DIPLOMADO
LENGUAJE DE PROGRAMACIÓN C#

MÓDULO 1 - ARQUITECTURA
CLASE 2 - PROGRAMACIÓN ORIENTADA A OBJETOS

JOEL TORRES CARRASCO
CRISTHIAN AGUILERA CARRASCO
CRISTIAN VALLEJOS VEGA

DEPARTAMENTO DE CIENCIAS DE LA INGENIERÍA
INGENIERÍA CIVIL EN INFORMÁTICA

Campus Osorno
Av. Fuchslocher 1305
Teléfono +56 64 2333 000
Fax +56 64 2333 774
Osorno, Chile

Campus Puerto Montt
Camino a Chinquihue Km 6
Teléfono +56 65 2322 536
Puerto Montt, Chile

Sede Santiago
República 517
Barrio Universitario
Teléfono +56 02 2675 3057
Santiago, Chile

Sede Chiloé
Ubaldo Mansilla Barrientos 131
Teléfono 56 65 2322 409
Castro, Chile
Eleuterio Ramírez 348
Teléfono +56 65 2322 476
Ancud, Chile

UNIVERSIDAD ACREDITADA
Septiembre de 2021 - Septiembre de 2026
Gestión Institucional · Docencia de Pregrado
Investigación · Vinculación con el Medio
AVANZADA

www.ulagos.cl

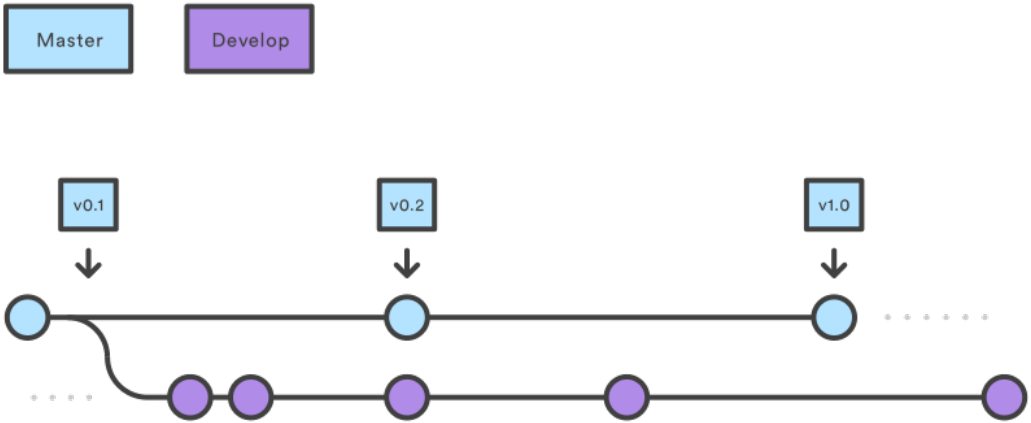
Tabla de Contenido

Sección Siguiente

¿QUÉ ES GIT?

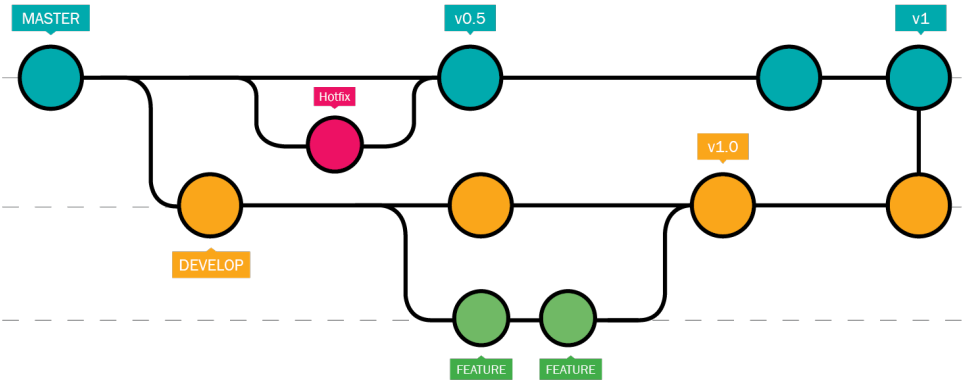
- Control de versiones distribuido
- Coordina el trabajo de multiples desarrolladores
- Identifica quien y cuando se hacen los cambios
- Permite recuperar versiones anteriores
- Permite trabajar con repositorios locales y remotos

¿QUÉ ES GIT?



- Mantiene el seguimiento al historial del código
- Captura una *instantánea* de los archivos
- El desarrollador decidirá cuando realizar la captura
- Se puede acceder en cualquier momento

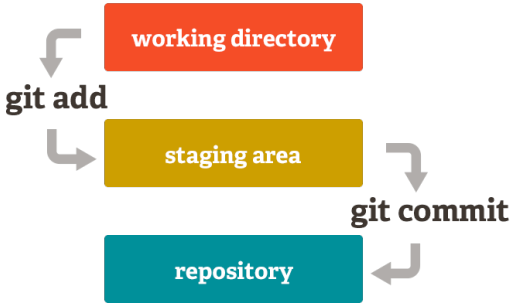
DINÁMICA DE TRABAJO DE GIT



¿Cómo funciona git?

Dinámica de git:

- Inicializar proyecto git
- Archivos al proyecto
- Gestión de Versiones



INSTALACIÓN DE GIT

Enlace de Descarga:
`https://git-scm.com`

*Versiones para sistemas operativos linux, windows y Mac OS.

CONFIGURACIÓN DE GIT

Revisar versión de git

```
git --version
```

Identificar desarrollador

```
git config --global user.name "Nombre Apellido"
git config --global user.email nombre.apellido@email.ext
```

*Este correo debe ser el usado en las cuentas de GitHub o GitLab

```
git config --list
```

Con este comando comprobamos la configuración

COMANDOS BÁSICOS DE GIT

Es importante situarse en el directorio del proyecto.

Inicializar Proyecto

De esta manera iniciamos el repositorio en git desde 0 (luego hablaremos de proyectos en marcha)

```
git init
```

con este comando se ve el estatus del repositorio en *working directory*

```
git status
```

Añadir archivos

Al añadir un archivo se transfieren desde *working directory* al *staging area*

```
git add <nombre_archivo>
```

ARCHIVO GITIGNORE

Cuando existen archivos o directorios que no se quieren agregar al proyecto, pero están dentro del directorio del proyecto, se deben incorporar en el archivo *.gitignore*

```
archivo_ignorado_1
directorio_ignorado_1
.
.
.
```

Luego es necesario incorporarlos a través de los siguientes comandos

```
git add .gitignore
git commit -m "gitignore incorporado."
```

COMANDOS BÁSICOS DE GIT

Guardando Cambios

Al guardar un archivo se transfieren desde *staging area* al *repository*

```
git commit -m "mensaje descriptivo del cambio a guardar"
```

Revisión de historial de cambios

Se revisarán todos los commit realizados por todos los usuarios a lo largo de aquel proyecto, mostrando marca de tiempo y el mensaje del cambio

```
git log
```

También podemos añadir opciones que permitan conocer todas las versiones de todas las ramas

```
git log --oneline --decorate --all --graph
```

COMANDOS BÁSICOS DE GIT

Revertir Cambios

Se puede volver a un punto específico del repositorio

```
git checkout -- <nombre_archivo.ext>
```

Volvemos a la última versión en la rama seleccionada.

Conocer diferencias entre versiones

Detalla las líneas de código entre las versiones de un archivo

```
git diff <nombre_archivo.ext>
```

COMANDOS BÁSICOS DE GIT

Seleccionar ramas

Para seleccionar la rama de desarrollo

```
git checkout <nombre_rama>
```

Seleccionamos la *rama*, por ejemplo: *master*, puede crear todas las ramas necesarias en desarrollo.

Visualizar ramas

Es necesario ver las líneas de desarrollo

```
git branch –a
```

Crear ramas

Si existen desarrollos más específicos se pueden crear líneas de desarrollo para evitar problemas de compatibilidad con el resto del proyecto

```
git branch <nombre_rama>
```

COMANDOS BÁSICOS DE GIT

Fusionando versiones entre ramas

Si la versión de una rama ya está lista, se puede añadir a la rama de la que procede.

```
git ckeckout <nombre_rama_destino>
git merge <nombre_rama_origen>
```

De esta manera, se unifican todas las versiones en las ramas involucradas. Automáticamente, git resuelve de forma recursiva añadir todas las versiones, permitiendo conocer su procedencia y sin perdida de contenido que pudiera ayudar en el futuro.

COMANDOS BÁSICOS DE GIT

Reiniciar Cambios aún en trabajo

Elimina todos los cambios realizados en *working directory*. Esto no afecta otras ramas ni áreas del repositorio

```
git reset --hard HEAD
```

O, también, eliminar un commits hasta un commit en específico

```
git reset --hard <id_commit>
```

Reiniciar Cambios Guardados

Es un comando que revierte el comando *git add*, es decir, que elimina todos los cambios en *staging area*

```
git reset --mixed HEAD
```

Es posible que solamente quiera eliminar un archivo.

```
git reset HEAD -- <nombre_archivo.ext>
```

COMANDOS BÁSICOS DE GIT

Revertir cambios sin eliminar versiones

Puede volver a una versión anterior, sin eliminar los commit efectuados posteriormente.

```
git revert <id_commit>
```

En este procedimiento, puede haber posibilidad de encontrar conflictos de versiones, por eliminación o agregación de archivos. Por lo que hay que revisar los conflictos y resolverlos. Las opciones son:

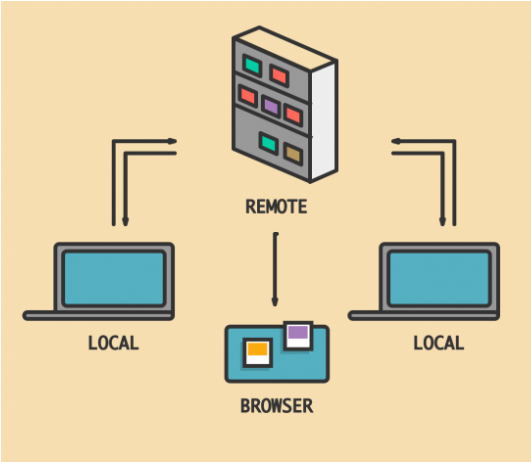
```
git revert --continue
git revert --abort
```

Para revolver conflictos, se deben añadir los archivos en conflicto de la siguiente manera:

```
git add .
git revert --continue
```

Sección Siguiente

El problema de trabajar en equipo



- Cada desarrollador tiene su propio trabajo en local
- Es difícil sincronizar versiones
- Es necesario identificar quien realizó ciertos cambios
- Es posible encontrar conflictos de versiones

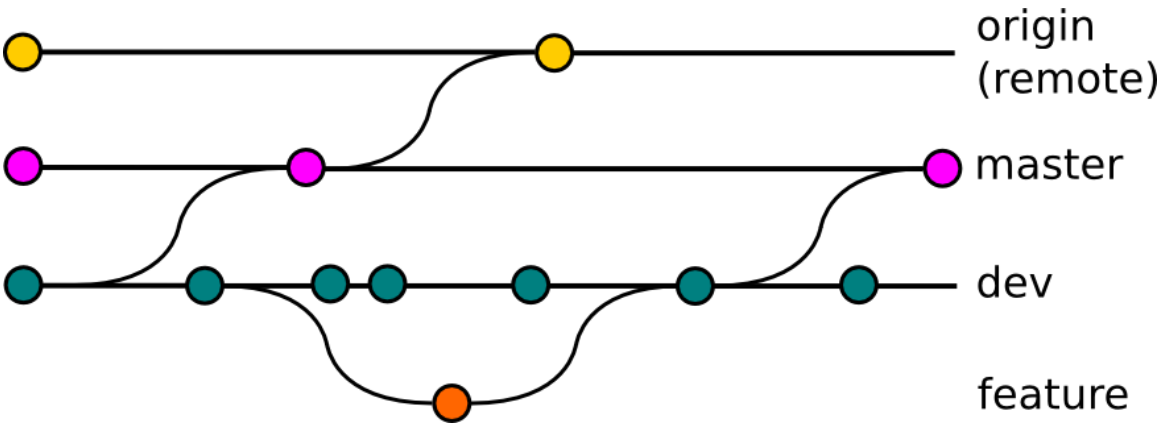
¿Qué es GitHub?

- Es una plataforma de desarrollo en la nube inspirada en la forma en que trabajan los equipos
- Es de código abierto, permite almacenar y revisar código, administrar proyectos y equipos
- Es la más popular por sobre *GitLab*, *BitBucket*, *SourceForge*, *Launchpad*, *Google Cloud Source Repositories*, *AWS CodeCommit*, *Phabricator*, *Gogs*, *Gitea*, y *Apache Allura*.

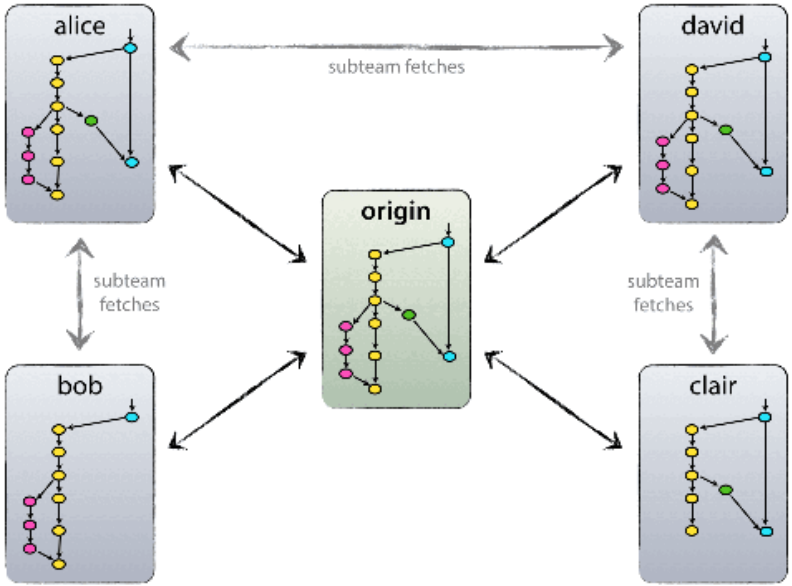
OBTENER CUENTA GITHUB

Enlace:
`https://github.com`

Dinámica trabajo Remoto

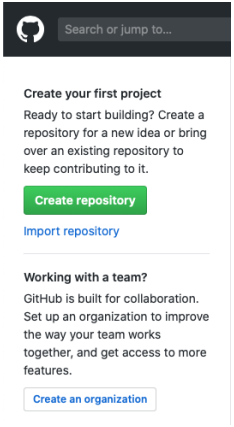


DINÁMICA TRABAJO REMOTO

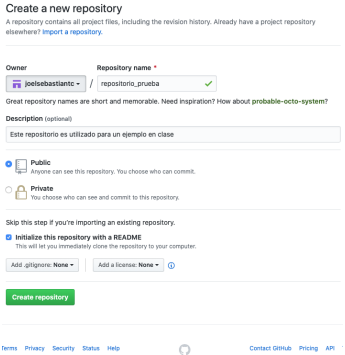


CREAR REPOSITORIO EN GITHUB

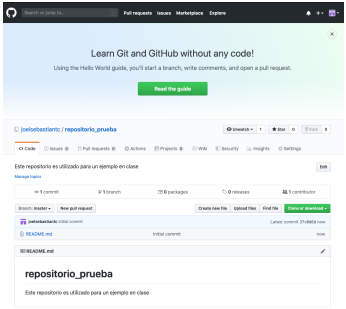
1. "Create Repository"



2. Completar campos



3. Repositorio creado



TRABAJAR COMO PROPIETARIO CON REPOSITORIO REMOTO

Conectar el repositorio remoto

```
git remote add origin
    https://github.com/<username>/<repository>.git
```

Agregamos la dirección remota donde se encuentra el repositorio git

```
git remote -v
```

Verificamos la conexión con servidor remoto

Subir proyecto local a remoto

```
// git push -u <remote> <branch>
git push -u origin master
```

Se abrirá las credenciales para GitHub, para dar permisos sobre el repositorio remoto, y luego subir el proyecto local al repositorio remoto

GUARDAR CAMBIOS EN GITHUB

Guardar en Local y enviar a Remoto

Si modificamos el proyecto, podemos agregar la nueva versión a remoto.

```
git add .
git commit -m "Guardando en Local"
// git push <remoto> <branch>
git push origin master
```

Se envía la versión guardada de forma local como versión remota.

PRACTICANDO GITHUB 1

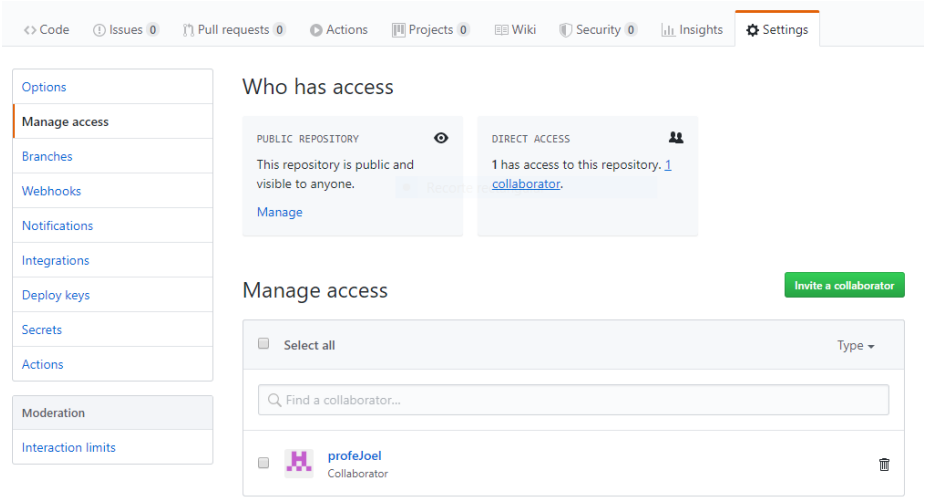
Sube tu propio proyecto:

- Crearán su cuenta personal en GitHub
- Crear un repositorio remoto
- Asociar el repositorio remoto
- Subirán los cambios a GitHub
- Modifique su proyecto local
- Guarde cambios y envíe la nueva versión a remoto.

TRABAJAR COMO COLABORADOR EN GITHUB

Añadir Colaboradores al Proyecto

Settings»Manage access»Invite a colaborator



TRABAJAR COMO COLABORADOR EN GITHUB

Clonar el repositorio remoto y trabajar en local

```
git clone https://github.com/<username>/<repository>.git
```

Se descarga el repositorio remoto a local, incluyendo todas las versiones.

Unificar las versiones de distintos desarrolladores

```
git fetch <remoto> <branch>
git merge <remoto> <branch>
```

o también está la siguiente versión equivalente.

```
git pull <remoto> <branch>
```

Esto indica que se descargará la información actualizada del proyecto y se fusionarán con las versiones locales.

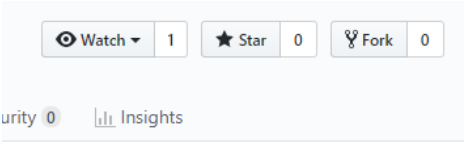
PRACTICANDO GITHUB 2

Vamos a hacer una lista de asistencia:

- Me informarán el nombre de usuario de GitHub para añadirlo como colaborador (email o Discord)
- Clonarán el proyecto *git_test_poo*
- Bajarán el proyecto a sus equipos
- Añadirán su nombre a la lista en el archivo de su Asignatura
- Guardarán la versión
- Subirán los cambios a GitHub

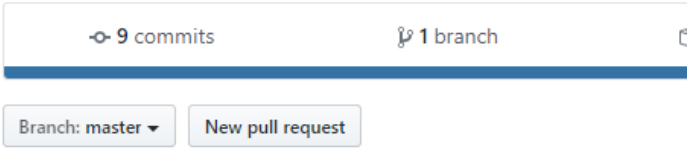
UTILIZAR CÓDIGO DE OTROS

Descargar el repositorio remoto y trabajar el proyecto aparte



Se copia todo el proyecto del repositorio remoto a tu repositorio remoto, incluyendo todas las versiones, pero se separa del proyecto original.

Solicitar integrar una versión al proyecto original



Toda la versión del nuevo branch al proyecto original queda en manos del propietario en añadirlo.

PRACTICANDO GITHUB 3

Vamos a hacer una lista de asistencia:

- Crearán un fork al proyecto *git_test_poo*
- Bajarán el proyecto a sus equipos
- Añadirán su nombre a un nuevo archivo y lo añadirán al proyecto
- Guardarán la versión
- Subirán los cambios a GitHub (su versión personal)
- Solicitarán añadir la versión con Pull Request

Pack de Estudiante Desarrollador de GitHub

Enlace:
`https://education.github.com/pack`