

# Configuración del Entorno

---

Git + Java + Maven en Linux

Clase 1 · Primeros pasos

GLUD — Grupo GNU/Linux Universidad Distrital

Control de Versiones y Desarrollo Colaborativo

# Mapa de ruta

1    ¿Por qué Git?

2    Instalación de Git

3    Java y Maven

4    Arquitectura de Git

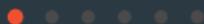
5    Práctica Guiada

6    Resumen

# 1

¿Por qué Git?

---



## Sin Git

- Archivos duplicados sin control

# El valor de Git en tu flujo de trabajo

¿Por qué Git?

## Sin Git

- Archivos duplicados sin control
- Colaboración por USB o email

## Con Git

- Historial completo y navegable

# El valor de Git en tu flujo de trabajo

¿Por qué Git?

## Sin Git

- Archivos duplicados sin control
- Colaboración por USB o email
- Miedo a experimentar

## Con Git

- Historial completo y navegable
- Colaboración simultánea

Hoy configuraremos todo lo necesario para trabajar profesionalmente

# El valor de Git en tu flujo de trabajo

¿Por qué Git?

## Sin Git

- Archivos duplicados sin control
- Colaboración por USB o email
- Miedo a experimentar
- No hay registro de cambios

## Con Git

- Historial completo y navegable
- Colaboración simultánea
- Ramas para experimentar

Hoy configuraremos todo lo necesario para trabajar profesionalmente

# El valor de Git en tu flujo de trabajo

¿Por qué Git?

## Sin Git

- Archivos duplicados sin control
- Colaboración por USB o email
- Miedo a experimentar
- No hay registro de cambios

## Con Git

- Historial completo y navegable
- Colaboración simultánea
- Ramas para experimentar
- Trazabilidad total

Hoy configuraremos todo lo necesario para trabajar profesionalmente

“

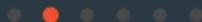
*The best time to start using version control  
was yesterday. The second best time is now.*

— Desarrollador Anónimo

# 2

## Instalación de Git

---



### Debian/Ubuntu

```
sudo apt update  
sudo apt install git -y
```

### Fedora/RHEL

```
sudo dnf install git -y
```

# Instalación en Linux

## Instalación de Git

### Debian/Ubuntu

```
sudo apt update  
sudo apt install git -y
```

### Arch Linux

```
sudo pacman -Sy git
```

### Fedora/RHEL

```
sudo dnf install git -y
```

### Verificación

Después de instalar:

```
git --version
```

## 1 Configura tu identidad (obligatorio)

```
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"
```

## 1 Configura tu identidad (obligatorio)

```
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"
```

## 2 Configura la rama por defecto

```
git config --global init.defaultBranch main
```

## 1 Configura tu identidad (obligatorio)

```
git config --global user.name "Tu Nombre"  
git config --global user.email "tu.email@ejemplo.com"
```

## 2 Configura la rama por defecto

```
git config --global init.defaultBranch main
```

## 3 Configura tu editor preferido

```
git config --global core.editor nano  
# Alternativas: vim, emacs, code --wait
```

Ver toda la configuración

```
git config --list --global
```

# Verificar Configuración

Instalación de Git

Ver toda la configuración

```
git config --list --global
```

Salida esperada

```
user.name=Tu Nombre  
user.email=tu.email@ejemplo.com  
init.defaultbranch=main  
core.editor=nano
```

# 3

## Java y Maven

---



Debian/Ubuntu

```
sudo apt update  
sudo apt install openjdk-21-jdk maven -y
```

## Debian/Ubuntu

```
sudo apt update  
sudo apt install openjdk-21-jdk maven -y
```

## Fedora/RHEL

```
sudo dnf install java-21-openjdk-devel maven -y
```

## Debian/Ubuntu

```
sudo apt update  
sudo apt install openjdk-21-jdk maven -y
```

## Fedora/RHEL

```
sudo dnf install java-21-openjdk-devel maven -y
```

## Verificación

```
java -version  
mvn -v
```

## Debian/Ubuntu

```
sudo apt update  
sudo apt install openjdk-21-jdk maven -y
```

## Fedora/RHEL

```
sudo dnf install java-21-openjdk-devel maven -y
```

## Verificación

```
java -version  
mvn -v
```

## ¿Qué son?

Las **variables de entorno** son valores que el sistema operativo utiliza para configurar el comportamiento de programas y scripts.

## ¿Qué son?

Las **variables de entorno** son valores que el sistema operativo utiliza para configurar el comportamiento de programas y scripts.

### JAVA\_HOME

Apunta a la instalación de Java

```
/usr/lib/jvm/  
java-21-openjdk
```

## ¿Qué son?

Las **variables de entorno** son valores que el sistema operativo utiliza para configurar el comportamiento de programas y scripts.

### JAVA\_HOME

Apunta a la instalación de Java

```
/usr/lib/jvm/  
java-21-openjdk
```

### PATH

Directorios donde buscar ejecutables

```
$JAVA_HOME/bin:  
/usr/local/bin:  
/usr/bin
```

- 1 Edita tu archivo de configuración de shell

```
nano ~/.bashrc      # o ~/.zshrc
```

- 1 Edita tu archivo de configuración de shell

```
nano ~/.bashrc      # o ~/.zshrc
```

- 2 Agrega al final del archivo

```
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk  
export PATH="$JAVA_HOME/bin:$PATH"
```

- 1 Edita tu archivo de configuración de shell

```
nano ~/.bashrc      # o ~/.zshrc
```

- 2 Agrega al final del archivo

```
export JAVA_HOME=/usr/lib/jvm/java-21-openjdk  
export PATH="$JAVA_HOME/bin:$PATH"
```

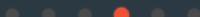
- 3 Recarga la configuración

```
source ~/.bashrc
```

# 4

## Arquitectura de Git

---



# Las Cuatro Áreas de Git

Arquitectura de Git



1 Working

Modificas archivos

2 Staging

Seleccionas cambios

3 Local Repo

Guardas commits

4 Remote

Sincronizas

# Las Cuatro Áreas de Git

Arquitectura de Git



1 Working

Modificas archivos

2 Staging

Seleccionas cambios

3 Local Repo

Guardas commits

4 Remote

Sincronizas

# Las Cuatro Áreas de Git

Arquitectura de Git



# Las Cuatro Áreas de Git

Arquitectura de Git



1 Working

Modificas archivos

2 Staging

Seleccionas cambios

3 Local Repo

Guardas commits

4 Remote

Sincronizas

### Definición

El **Working Directory** es tu carpeta de trabajo donde modificas archivos libremente.

## Definición

El **Working Directory** es tu carpeta de trabajo donde modificas archivos libremente.

## Características:

- Archivos sin rastrear
- Archivos modificados
- Puedes editar libremente

## Definición

El **Working Directory** es tu carpeta de trabajo donde modificas archivos libremente.

## Características:

- Archivos sin rastrear
- Archivos modificados
- Puedes editar libremente

Ver estado

git status

Muestra archivos modificados y sin rastrear

## Definición

El **Staging Area** es donde preparas exactamente qué cambios quieres guardar en el próximo commit.

## Definición

El **Staging Area** es donde preparas exactamente qué cambios quieres guardar en el próximo commit.

## Agregar archivos

```
git add archivo.txt  
git add src/  
git add .
```

## Definición

El **Staging Area** es donde preparas exactamente qué cambios quieres guardar en el próximo commit.

### Agregar archivos

```
git add archivo.txt  
git add src/  
git add .
```

### Agregar selectivamente

```
git add -p archivo.txt
```

Te permite elegir qué partes agregar

## Definición

El **Staging Area** es donde preparas exactamente qué cambios quieres guardar en el próximo commit.

### Agregar archivos

```
git add archivo.txt  
git add src/  
git add .
```

### Agregar selectivamente

```
git add -p archivo.txt
```

Te permite elegir qué partes agregar

El staging permite commits **atómicos** y bien organizados

## 1 Ver cambios en Working Directory

```
git diff
```

- 1 Ver cambios en Working Directory

```
git diff
```

- 2 Ver cambios en Staging

```
git diff --staged
```

- 1 Ver cambios en Working Directory

```
git diff
```

- 2 Ver cambios en Staging

```
git diff --staged
```

- 3 Ver cambios entre commits

```
git diff HEAD~1 HEAD
```

- 1 Ver cambios en Working Directory

```
git diff
```

- 2 Ver cambios en Staging

```
git diff --staged
```

- 3 Ver cambios entre commits

```
git diff HEAD~1 HEAD
```

git diff es tu mejor amigo para revisar cambios

### Definición

El **Local Repository** almacena el historial completo de commits en tu máquina.

### Definición

El **Local Repository** almacena el historial completo de commits en tu máquina.

### Crear commit

```
git commit -m "mensaje"
```

### Ver historial

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

### Definición

El **Local Repository** almacena el historial completo de commits en tu máquina.

#### Crear commit

```
git commit -m "mensaje"
```

#### Ver un commit

```
git show <hash>
```

#### Ver historial

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

#### Tip

Escribe mensajes descriptivos y claros

### Definición

Es una copia de tu repositorio alojada en un servidor (GitHub, GitLab, etc.).

## Definición

Es una copia de tu repositorio alojada en un servidor (GitHub, GitLab, etc.).

## Conectar con remoto

```
git remote add origin https://github.com/usuario/repo.git
```

## Definición

Es una copia de tu repositorio alojada en un servidor (GitHub, GitLab, etc.).

## Conectar con remoto

```
git remote add origin https://github.com/usuario/repo.git
```

## Enviar cambios

```
git push -u origin main
```

## Definición

Es una copia de tu repositorio alojada en un servidor (GitHub, GitLab, etc.).

## Conectar con remoto

```
git remote add origin https://github.com/usuario/repo.git
```

## Enviar cambios

```
git push -u origin main
```

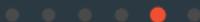
## Clonar repositorio

```
git clone https://github.com/usuario/repo.git
```

# 5

## Práctica Guiada

---



## 1 Crear directorio y entrar

```
mkdir mi-primer-repo && cd mi-primer-repo
```

- 1 Crear directorio y entrar

```
mkdir mi-primer-repo && cd mi-primer-repo
```

- 2 Inicializar Git

```
git init
```

## 1 Crear directorio y entrar

```
mkdir mi-primer-repo && cd mi-primer-repo
```

## 2 Inicializar Git

```
git init
```

### Resultado

Se crea una carpeta oculta .git/ que contiene toda la información del repositorio.

3

Crear archivo README

```
echo "# Mi Primer Repositorio" > README.md
```

## 3 Crear archivo README

```
echo "# Mi Primer Repositorio" > README.md
```

## 4 Agregar al staging

```
git add README.md
```

## 3 Crear archivo README

```
echo "# Mi Primer Repositorio" > README.md
```

## 4 Agregar al staging

```
git add README.md
```

## 5 Crear commit

```
git commit -m "feat: agregar README inicial"
```

## ¿Qué es .gitignore?

Archivo que especifica qué archivos NO deben ser rastreados por Git.

## ¿Qué es .gitignore?

Archivo que especifica qué archivos NO deben ser rastreados por Git.

### Crear .gitignore para Java/Maven

```
target/  
*.class  
.idea/  
.vscode/  
*.log
```

Siempre crea .gitignore al inicio del proyecto

## Secuencia típica

```
# 1. Hacer cambios en archivos  
vim archivo.java  
  
# 2. Ver qué cambió  
git status  
git diff  
  
# 3. Agregar al staging  
git add archivo.java  
  
# 4. Crear commit  
git commit -m "feat: implementar nueva funcionalidad"  
  
# 5. Enviar al remoto  
git push
```

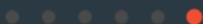
## Estructura del ejercicio

```
mkdir proyecto-java && cd proyecto-java
git init
mkdir -p src/main/java/com/ ejemplo
touch README.md .gitignore
git add README.md
git commit -m "docs: agregar README"
git add .gitignore
git commit -m "chore: configurar gitignore"
# ... continúa con más commits
```

# 6

## Resumen

---



## Instalación:

- Git en Linux
- JDK 17 + Maven
- Variables de entorno
- Configuración global

## Conceptos:

- Working Directory
- Staging Area
- Local Repository
- Remote Repository

## Comandos:

- `git init`
- `git add / git add -p`
- `git commit`
- `git diff / --staged`
- `git push / pull / fetch`
- `git remote add`
- `git clone`

## Documentación:

- Pro Git (libro oficial)
- <https://git-scm.com/docs>
- <https://training.github.com>

## Práctica:

- <https://learngitbranching.js.org>
- <https://gitexercises.fracz.com>
- GitHub Student Developer Pack

La práctica hace al maestro

# Próxima Clase

Estados del archivo (Untracked, Staged, Committed). Conventional Commits.  
Estructura de proyecto del stack elegido. Configuración de .gitignore apropiado  
([gitignore.io](http://gitignore.io)).

# ¡Gracias!

Nos vemos en la próxima clase