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?

El valor de Git en tu flujo de trabajo

Sin Git

- Archivos duplicados sin control

El valor de Git en tu flujo de trabajo

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

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

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

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

Instalación en Linux

Debian/Ubuntu

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

Fedora/RHEL

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

Instalación en Linux

Debian/Ubuntu

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

Fedora/RHEL

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

Arch Linux

```
sudo pacman -Sy git
```

Verificación

Después de instalar:

```
git --version
```

Configuración Inicial de Git

1. Configura tu identidad (obligatorio)

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

Configuración Inicial de Git

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
```

Configuración Inicial de Git

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
```

Verificar Configuración

Ver toda la configuración

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

Verificar Configuración

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

Instalación de JDK 21 y Maven

Debian/Ubuntu

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

Instalación de JDK 21 y Maven

Debian/Ubuntu

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

Fedora/RHEL

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

Instalación de JDK 21 y Maven

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
```

Instalación de JDK 21 y Maven

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
```

Variables de Entorno

¿Qué son?

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

Variables de Entorno

¿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
```

Variables de Entorno

¿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
```

Configurar Variables de Entorno

1. Edita tu archivo de configuración de shell

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

Configurar Variables de Entorno

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"
```

Configurar Variables de Entorno

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

Working Directory

Archivos editables

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

1 Working

Modificas archivos

2 Staging

Seleccionas cambios

3 Local Repo

Guardas commits

4 Remote

Sincronizas

Las Cuatro Áreas de Git

1 Working
Modificas archivos

2 Staging
Seleccionas cambios

3 Local Repo
Guardas commits

4 Remote
Sincronizas

Working Directory

Definición

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

Working Directory

Definición

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

Características:

- Archivos sin rastrear
- Archivos modificados
- Puedes editar libremente

Working Directory

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

Staging Area (Index)

Definición

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

Staging Area (Index)

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 .
```

Staging Area (Index)

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

Staging Area (Index)

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

Comparando Cambios

1. Ver cambios en Working Directory

```
git diff
```

Comparando Cambios

1. Ver cambios en Working Directory

```
git diff
```

2. Ver cambios en Staging

```
git diff --staged
```

Comparando Cambios

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
```

Comparando Cambios

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

Local Repository

Definición

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

Local Repository

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
```

Local Repository

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

Remote Repository

Definición

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

Remote Repository

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
```

Remote Repository

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
```

Remote Repository

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

Creando tu Primer Repositorio

1. Crear directorio y entrar

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

Creando tu Primer Repositorio

1. Crear directorio y entrar

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

2. Inicializar Git

```
git init
```

Creando tu Primer Repositorio

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.

Primer Commit

3 Crear archivo README

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

Primer Commit

3 Crear archivo README

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

4 Agregar al staging

```
git add README.md
```

Primer Commit

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"
```

Gitignore: Ignorar Archivos

¿Qué es .gitignore?

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

Gitignore: Ignorar Archivos

¿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

Flujo Completo de Trabajo

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
```

Comandos de Ayuda

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

Lo que Aprendimos

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`

Recursos Útiles

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).

¡Gracias!

Nos vemos en la próxima clase