Flujos Remotos

GitHub, GitLab y el Mundo Exterior

GLUD — Grupo GNU/Linux Universidad Distrital

Control de Versiones y Desarrollo Colaborativo

Mapa de ruta

1. Del Local al Remoto
2. Autenticación SSH
3. Repositorios Remotos
4. Gestión de Colaboradores
5. Configuración del Repositorio
6. Protección de Ramas

1

Del Local al Remoto

Git es Distribuido

Recordatorio

Git es un sistema de control de versiones **distribuido**. Cada clon es un repositorio completo.

Git es Distribuido

Recordatorio

Git es un sistema de control de versiones **distribuido**. Cada clon es un repositorio completo.

Hasta ahora:

- Todo en tu máquina local
- Commits, ramas, merges
- Sin conexión a internet
- Solo tú tienes acceso

Git es Distribuido

Recordatorio

Git es un sistema de control de versiones **distribuido**. Cada clon es un repositorio completo.

Hasta ahora:

- Todo en tu máquina local
- Commits, ramas, merges
- Sin conexión a internet
- Solo tú tienes acceso

Ahora:

- Conectar con servidores remotos
- Compartir código con el equipo
- Respaldo en la nube
- Colaboración real

GitHub vs GitLab vs Otros

GitHub

- El más popular
- Propiedad de Microsoft
- GitHub Actions
- Comunidad enorme

GitLab

- Open source
- Self-hosted posible
- CI/CD integrado
- Todo en uno

Otros

- Bitbucket (Atlassian)
- Gitea (self-hosted)
- Azure DevOps
- Codeberg

GitHub vs GitLab vs Otros

GitHub

- El más popular
- Propiedad de Microsoft
- GitHub Actions
- Comunidad enorme

GitLab

- Open source
- Self-hosted posible
- CI/CD integrado
- Todo en uno

Otros

- Bitbucket (Atlassian)
- Gitea (self-hosted)
- Azure DevOps
- Codeberg

Los comandos de Git son los mismos — solo cambia la interfaz web

2

Autenticación SSH

¿Por Qué SSH?

HTTPS

URL: `https://github.com/user/repo.git`

- Pide usuario/contraseña
- Necesita token de acceso
- Más fácil de configurar
- Puede ser tedioso

¿Por Qué SSH?

HTTPS

URL: `https://github.com/user/repo.git`

- Pide usuario/contraseña
- Necesita token de acceso
- Más fácil de configurar
- Puede ser tedioso

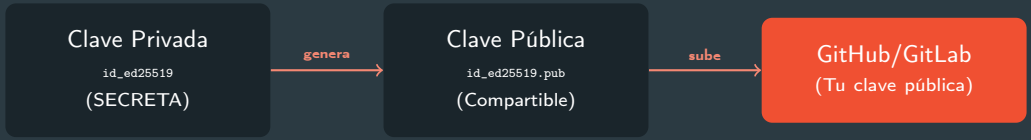
SSH

URL: `git@github.com:user/repo.git`

- Usa clave pública/privada
- Sin contraseñas repetidas
- Más seguro
- Configuración inicial

Recomendación: Usar SSH para desarrollo profesional

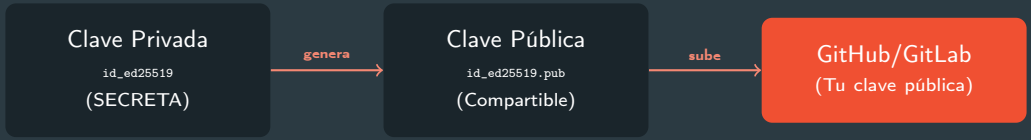
Criptografía de Clave Pública



Nunca compartas

Puedes compartir

Criptografía de Clave Pública



Analogía

La clave pública es como un candado que regalas. Solo tú tienes la llave (clave privada) para abrirlo.

Generar Par de Claves SSH

Generar clave (algoritmo moderno)

```
ssh-keygen -t ed25519 -C "tu@email.com"
```

Generar Par de Claves SSH

Generar clave (algoritmo moderno)

```
ssh-keygen -t ed25519 -C "tu@email.com"
```

Preguntas interactivas

```
Enter file in which to save the key (/home/user/.ssh/id_ed25519):  [Enter]
Enter passphrase (empty for no passphrase):  [opcional]
Enter same passphrase again:  [opcional]
```

Generar Par de Claves SSH

Generar clave (algoritmo moderno)

```
ssh-keygen -t ed25519 -C "tu@email.com"
```

Preguntas interactivas

```
Enter file in which to save the key (/home/user/.ssh/id_ed25519):  [Enter]
Enter passphrase (empty for no passphrase):  [opcional]
Enter same passphrase again:  [opcional]
```

Passphrase

La passphrase es una capa extra de seguridad. Si la usas, deberás ingresarla cada vez (o usar ssh-agent).

Ubicación de las Claves

Las claves se guardan en:

```
# Linux/macOS
~/.ssh/id_ed25519     # Clave PRIVADA
~/.ssh/id_ed25519.pub # Clave PÚBLICA
```

Ubicación de las Claves

Las claves se guardan en:

```
# Linux/macOS
~/.ssh/id_ed25519     # Clave PRIVADA
~/.ssh/id_ed25519.pub # Clave PÚBLICA
```

Ver tu clave pública

```
cat ~/.ssh/id_ed25519.pub
```

Salida: `ssh-ed25519 AAAAC3NzaC1lZDI... tu@email.com`

Agregar Clave a GitHub

1 Copiar tu clave pública

```
# Linux
cat ~/.ssh/id_ed25519.pub | xclip -selection clipboard
```

Agregar Clave a GitHub

1 Copiar tu clave pública

```
# Linux
cat ~/.ssh/id_ed25519.pub | xclip -selection clipboard
```

2 En GitHub/GitLab

Settings → SSH and GPG Keys → New SSH Key → Pegar → Add

Verificar Conexión SSH

Probar conexión con GitHub

```
ssh -T git@github.com
```

Verificar Conexión SSH

Probar conexión con GitHub

```
ssh -T git@github.com
```

Respuesta exitosa

```
Hi username!  You've successfully authenticated, but GitHub
does not provide shell access.
```

Verificar Conexión SSH

Probar conexión con GitHub

```
ssh -T git@github.com
```

Respuesta exitosa

```
Hi username!  You've successfully authenticated, but GitHub
does not provide shell access.
```

Probar conexión con GitLab

```
ssh -T git@gitlab.com
```

Si funciona, ya puedes clonar y pushear sin contraseña

3

Repositorios Remotos

Conectar Local con Remoto

Caso 1: Ya tienes un repo local

```
git remote add origin git@github.com:usuario/proyecto.git
git branch -M main
git push -u origin main
```

Conectar Local con Remoto

Caso 1: Ya tienes un repo local

```
git remote add origin git@github.com:usuario/proyecto.git
git branch -M main
git push -u origin main
```

Caso 2: Empezar desde cero

```
git clone git@github.com:usuario/proyecto.git
cd proyecto
```

El Concepto de “Origin”

Definición

`origin` es el nombre por defecto del repositorio remoto principal. Es solo un **alias**.

El Concepto de “Origin”

Definición

`origin` es el nombre por defecto del repositorio remoto principal. Es solo un **alias**.

Ver remotos configurados

```
git remote -v
```

El Concepto de “Origin”

Definición

`origin` es el nombre por defecto del repositorio remoto principal. Es solo un **alias**.

Ver remotos configurados

```
git remote -v
```

Salida típica

```
origin  git@github.com:usuario/proyecto.git (fetch)
origin  git@github.com:usuario/proyecto.git (push)
```

Origin vs Upstream

origin

Tu repositorio

- Tu fork o repo principal
- Donde haces push
- Tienes permisos de escritura

Origin vs Upstream

origin

Tu repositorio

- Tu fork o repo principal
- Donde haces push
- Tienes permisos de escritura

upstream

Repositorio original

- El proyecto original
- De donde haces fetch
- Solo lectura (normalmente)

Origin vs Upstream

origin

Tu repositorio

- Tu fork o repo principal
- Donde haces push
- Tienes permisos de escritura

upstream

Repositorio original

- El proyecto original
- De donde haces fetch
- Solo lectura (normalmente)

Agregar upstream

```
git remote add upstream git@github.com:original/proyecto.git
```

4

Gestión de Colaboradores

Roles en GitHub

Repositorio Personal:

- Owner (tú)
- Collaborators (invitados)

Solo dos niveles: dueño o colaborador con acceso completo

Roles en GitHub

Repositorio Personal:

- Owner (tú)
- Collaborators (invitados)

Solo dos niveles: dueño o colaborador con acceso completo

Organización:

- Owner
- Admin
- Maintainer
- Write
- Triage
- Read

Para proyectos grandes, usar Organizaciones permite control granular

5

Configuración del Repositorio

Settings Esenciales (GitHub)

Repository → Settings:

- **General:** Nombre, descripción, visibilidad
- **Default branch:** Cambiar rama principal
- **Features:** Wikis, Issues, Projects, Discussions
- **Danger Zone:** Archivar, transferir, eliminar

Settings Esenciales (GitHub)

Repository → Settings:

- **General:** Nombre, descripción, visibilidad
- **Default branch:** Cambiar rama principal
- **Features:** Wikis, Issues, Projects, Discussions
- **Danger Zone:** Archivar, transferir, eliminar

Cambiar rama por defecto

Settings → General → Default Branch → Cambiar de `master` a `main`

Settings Esenciales (GitLab)

Project → Settings → General:

- **Naming:** Nombre y descripción
- **Visibility:** Private, Internal, Public
- **Advanced:** Transfer, archive, delete

Settings Esenciales (GitLab)

Project → Settings → General:

- **Naming:** Nombre y descripción
- **Visibility:** Private, Internal, Public
- **Advanced:** Transfer, archive, delete

Project → Settings → Repository:

- **Default branch:** Rama principal
- **Protected branches:** Reglas de protección
- **Deploy keys:** Acceso automatizado

6

Protección de Ramas

¿Por Qué Proteger Ramas?

El Problema

Sin protección, cualquiera puede hacer push directo a `main` y potencialmente romper producción.

¿Por Qué Proteger Ramas?

El Problema

Sin protección, cualquiera puede hacer push directo a `main` y potencialmente romper producción.

Branch Protection Rules permiten:

- Prohibir push directo a main
- Requerir aprobaciones antes de merge
- Requerir que CI pase antes de merge
- Prohibir force push
- Requerir firma de commits

Configurar Branch Protection

Branch name pattern

```
main
```

Opciones recomendadas:

- ✓ Require a pull request before merging
- ✓ Require status checks to pass
- ✓ Do not allow bypassing the above settings
- ✓ Restrict who can push (opcional)

Ejemplo de Reglas por Rama

Rama	Push Directo	PR/MR	CI Requerido
main	No	Sí	Sí
develop	Maintainers	Recomendado	Sí
feature/*	Developers	No	Opcional
hotfix/*	Maintainers	Sí	Sí

Esto implementa Git Flow a nivel de permisos del repositorio

7

Webhooks y Automatización

¿Qué son los Webhooks?

Definición

Un **webhook** es una notificación HTTP que GitHub/GitLab envía a un servidor externo cuando ocurre un evento.

¿Qué son los Webhooks?

Definición

Un **webhook** es una notificación HTTP que GitHub/GitLab envía a un servidor externo cuando ocurre un evento.

Eventos comunes:

- Push a una rama
- Creación de rama/tag
- Pull Request creado
- Comentarios
- Issues
- Releases
- Merge
- Deployment

Casos de Uso de Webhooks

- **CI/CD externo:** Trigger de Jenkins, CircleCI
- **Notificaciones:** Slack, Discord, Teams
- **Deployment:** Desplegar automáticamente al push
- **Validaciones:** Verificar código, linting
- **Integraciones:** Jira, Trello, Notion

Casos de Uso de Webhooks

- **CI/CD externo:** Trigger de Jenkins, CircleCI
- **Notificaciones:** Slack, Discord, Teams
- **Deployment:** Desplegar automáticamente al push
- **Validaciones:** Verificar código, linting
- **Integraciones:** Jira, Trello, Notion

Configuración

Settings → Webhooks → Add webhook
Payload URL, Content type, Secret, Events

Deploy Keys

Definición

Una **Deploy Key** es una clave SSH que da acceso a un solo repositorio (no a toda tu cuenta).

Deploy Keys

Definición

Una **Deploy Key** es una clave SSH que da acceso a un solo repositorio (no a toda tu cuenta).

¿Cuándo usarla?

- Servidores de CI/CD
- Scripts automatizados
- Acceso de solo lectura para despliegues
- Cuando no quieres usar tu clave personal

Configuración: Settings → Deploy keys → Add deploy key

Forks: Contribuir a Proyectos Ajenos

Definición

Un **fork** es una copia de un repositorio ajeno en tu cuenta, donde puedes hacer cambios libremente.

Forks: Contribuir a Proyectos Ajenos

Definición

Un **fork** es una copia de un repositorio ajeno en tu cuenta, donde puedes hacer cambios libremente.

Flujo de contribución:

1. Fork del proyecto original
2. Clone de TU fork
3. Crear rama, hacer cambios, push
4. Abrir Pull Request hacia el original
5. Mantener sincronizado con upstream

Es la forma estándar de contribuir a proyectos open source

Sincronizar Fork con Upstream

Configurar upstream (una vez)

```
git remote add upstream git@github.com:original/proyecto.git
```

Sincronizar Fork con Upstream

Configurar upstream (una vez)

```
git remote add upstream git@github.com:original/proyecto.git
```

Sincronizar con el original

```
git fetch upstream
git switch main
git merge upstream/main
git push origin main
```

Sincronizar Fork con Upstream

Configurar upstream (una vez)

```
git remote add upstream git@github.com:original/proyecto.git
```

Sincronizar con el original

```
git fetch upstream
git switch main
git merge upstream/main
git push origin main
```

GitHub también tiene botón “Sync fork” en la interfaz web

Próxima Clase: Code Review

Clase 7

Ya sabemos conectarnos y sincronizar. Ahora aprenderemos el **arte de revisar código ajeno**.

En la Clase 7 aprenderemos:

- Pull Requests / Merge Requests
- El proceso de code review
- Comentarios constructivos
- Aprobaciones y cambios solicitados
- Rol rotativo de Maintainer

¡Preparen sus repositorios con Branch Protection!

“

Talk is cheap. Show me the code.

— Linus Torvalds

¡Gracias!

Ahora están conectados al mundo