**Aula 02**

**2.1 – Agente SSH**

**🔑 O que é um agente SSH?**

* O **SSH Agent** é um **programa que roda em segundo plano** no seu sistema.
* Ele guarda suas **chaves privadas SSH** carregadas na memória.
* Assim, quando você faz um git push, o Git não precisa pedir a senha/passphrase da chave toda vez.

**🔐 Por que você precisa dele?**

Imagine o fluxo:

1. Você gera uma chave SSH com passphrase (senha opcional de segurança).
2. Sempre que o Git precisa autenticar no GitHub, ele precisa dessa chave.
3. Se não houver um agente, você teria que digitar a passphrase **toda vez** que faz git push ou git pull.

➡️ O agente SSH resolve isso: você **informa a senha uma vez**, ele guarda a chave na memória, e depois as conexões usam essa chave de forma automática e transparente.

**🛠️ Como funciona na prática**

1. Inicia o agente:
2. eval "$(ssh-agent -s)"
3. Adiciona a chave ao agente:
4. ssh-add ~/.ssh/id\_ed25519
5. A partir daí, o agente responde quando o GitHub pede autenticação.

**📌 Resumindo**

* O **SSH Agent** é como um **cofre temporário na memória** que guarda suas chaves privadas.
* Ele evita que você fique digitando a senha da chave a cada comando Git.
* Por isso, quando configuramos o GitHub com SSH, quase sempre o próximo passo é **adicionar a chave ao agente**.

**2.2 – Repositórios Git**

Um repositório Git pode ser:

* **Local** → só na sua máquina.
* **Remoto** → hospedado no GitHub/GitLab/etc.

👉 O **link do repositório remoto** é o endereço que conecta os dois.

**🔑 Tipos de link**

1. **HTTPS**
2. https://github.com/usuario/repositorio.git
   * Funciona sem SSH.
   * Pede login/senha ou token de acesso quando você faz push.
3. **SSH**
4. git@github.com:usuario/repositorio.git
   * Funciona com chave SSH.
   * Depois de configurado, **não pede senha nunca mais**.

**🛠️ Onde entra o link?**

* **Clonar** um repositório (primeira vez que traz para sua máquina):
* git clone git@github.com:usuario/repositorio.git
* **Adicionar um remoto** a um repositório já existente:
* git remote add origin git@github.com:usuario/repositorio.git
* **Trocar de HTTPS para SSH** (se já tiver configurado antes com https):
* git remote set-url origin git@github.com:usuario/repositorio.git

✅ Resumindo:  
Você **precisa copiar o link do repo** porque é assim que o Git sabe onde está o repositório remoto no GitHub.

* Se você **vai clonar** um projeto → usa o link.
* Se você **já tem um projeto local e quer subir** → adiciona esse link como origin.

**2.3 – GPG Keys**

| **Aspecto** | **SSH Key** | **GPG Key** |
| --- | --- | --- |
| **Função** | Autenticação: prova que você tem acesso ao repositório remoto. | Assinatura: prova que foi você mesmo quem fez aquele commit. |
| **Onde é usada** | Ao clonar, puxar (git pull) e enviar (git push) código para GitHub/GitLab. | Dentro de cada commit, para assinar digitalmente o histórico do código. |
| **Fluxo** | Substitui login/senha → conexão segura com servidor. | Dá autenticidade ao commit → ninguém pode forjar “commit em seu nome”. |
| **Configuração** | ssh-keygen + adicionar chave pública no GitHub/GitLab. | gpg --gen-key + configurar chave pública no GitHub/GitLab + git config --global user.signingkey. |
| **Visibilidade** | Invisível dentro do repositório, só controla acesso. | Fica marcado no commit com selo “Verified” no GitHub/GitLab. |

**🔐 Exemplos**

Usando SSH Key

git clone git@github.com:usuario/repositorio.git

# ou

git push origin main

**👉 Aqui o GitHub confere sua chave SSH para ver se você tem permissão.**

**Usando GPG Key**

git commit -S -m "Corrige bug no parser"

**👉 Esse -S assina o commit.**No GitHub vai aparecer o selo verde Verified, indicando que aquele commit foi realmente feito por você.

**📌 Resumindo**

* **SSH Key** → garante que só você consiga acessar/enviar código pro repositório.
* **GPG Key** → garante que os commits no histórico foram realmente feitos por você (integridade + autoria).