**Versionamento de código**

- A cada alteração do código geramos uma nova versão

**Sistemas de Controle de Versão:**

controlam as versões de um arquivo ao longo do tempo; registra o histórico de atualizações; gerenciam quais foram as atualizações, a data e o autor; oferece mais organização, controle e segurança

**- VCS Centralizado:** um único servidor com os autores conectados a ele (traz alguns riscos)

**- VCS Distribuído:** Clona o repositório completo (incluindo o histórico de versões) tornando possível fazer alterações mesmo sem estar conectado ao servidor ou a rede

Git

- Sistema de controle de versão distribuído

- É gratuito e open source

- Tem ramificações (branching) e fusões (merging) eficientes

- É leve e rápido

git clone 🡪 vai clonar o repositório existente para uma pasta local

git commit 🡪 grava a alteração no repositório

git pull 🡪”puxa” as alterações do repositório remoto para o local (busca e mescla)

git push 🡪 “empurra” as alterações do repositório local para o remoto

**Configurando o Git**

git config 🡪 permite definir variáveis de configuração no Git (global, system e local)

Definir um email e senha no global (vai servir para todos os repositórios que criarmos)

- git config –global user.email [*seu@email.com*](mailto:seu@email.com)

- git config –global user.name *seuNome*

Para ver se está configurado é só colocar *git config user.name* e ele retornara o nome

O nome que está configurado ficará em commits futuros

**Alterar a Branch**

- O que é Branch: (ou ramificações) são "cópias paralelas" do seu projeto, onde você pode trabalhar em novas funcionalidades, corrigir bugs ou testar coisas sem mexer no código principal

git config init.defaultBranch irá retornar o nome da Branch padrão

Para alterar git config --global init.defaultBranch e o nome será alterado para Main globalmente

GitHub

- Plataforma de hospedagem de código para controle de versão com Git e colaboração

- Comunidade ativa

**Autenticação e Token**

- Se tentar clonar um repositório privado sem o token de autenticação, vai retornar um erro

- Para **ativar a autenticação e criar o Token**: Settings – Developer Settings – Personal Acess Token - Token (classic) – Generete New Token (classic)

- O note é o nome que vai dizer para o que serve esse token e depois seleciona a data de expiração

- o escopo são as permissões que o token pode ter

- ao final gera o código de acesso ao token para ser utilizado como senha no git clone (é necessário utilizá-lo no exato momento, pois não será possível retornar a essa página)

**Salvar as credenciais na máquina**

git config --globlal credential.helper store

* Assim já colona automaticamente sem ficar pedindo a senha(credencial)

**Importante:** *o token pode ser deletado manualmente nas configs antes de expirar e também podem ser criados outros tokens. Se for para uso pessoal só um token basta*

*Evite usar um token com permissões demais se não for necessário. Se só precisa clonar/puxar, pode usar um token só com permissão de leitura.*

**Para mostrar onde estão os valores configurados:**

git config --global –show-origin credential.helper

Se entrar na pasta git config mostra o usuário, a branch e a credencial cat .gitconfig

**SSH**

- Protocolo de rede para permitir que o seu computador e o servidor remoto (github) se comuniquem de forma segura e criptografada por meio da internet

**- Como fazer:** settings – SSH Keys and GPG Keys

**Para verificar se já temos chaves:**

- No gitbash ls -al ~/.ssh (**ls** exibe os arquivos | **-al** inclui os arquivos ocultos - começam com **.** | **~** é um atalho para o diretório pessoal)

- Se retornar que não existe esse arquivo ou diretório, significa que não temos a chave SSH na máquina

- Poder retornar:

id\_rsa: sua **chave privada** (nunca compartilhe essa). **É a que NÃO tem pub**

id\_rsa.pub: sua **chave pública** (essa você pode colar no GitHub, por exemplo).

known\_hosts: lista de servidores conhecidos (para segurança).

- Para gerar uma chave é só seguir os comandos do GitHub em “Gerar nova chave SSH”

- Para adicionar a chave pública ao GitHub: cat id\_25519.pub

**Importante:** *a chave GPG ou a própria SSH (mais fácil se já tiver) também servem para assinatura de commit. Ao lado do seu commit aparece verified mostrando que o commit realmente foi seu (útil em open source e empresas)*

**Dicas:**

git pull 🡪 Para puxar as alterações do repositório remoto para o local

Se passar git clone linkSSD nome\_da\_pasta, você consegue mudar o nome do repositório na sua máquina

git remote -v mostra quais repositórios remotos você está vinculado

echo nome\_pasta/> .gitignore adiciona a pasta ao gitignore

**Desfazer alterações no repositório local**

git restore nome\_arquivo restaura o arquivo de volta, antes da atualização

git log mostra o histórico dos commits

- Mudar nome do commit: git commit --amend -m”novo\_nome”

**IMPORTANTE:**

HEAD → último commit

HEAD~1 → commit anterior

HEAD~2 → dois commits atrás

HEAD~3 → três commits atrás

git restore arquivo 🡪 Serve pra **desfazer mudanças na bancada de trabalho (working directory). R**estaura o arquivo para o que estava no stage (ou seja, desfaz só a última alteração que ainda **não** tinha sido adicionada com git add

git restore --staged 🡪 Tira o arquivo da preparação (staging), mas mantém a edição.

git reset --soft HEAD 🡪 Desfazer o último commit, mas manter os arquivos no stage. Tipo: "Ops, comitei cedo demais. Quero ajustar e commitar de novo."

git reset --mixed HEAD 🡪 Desfazer o commit e tirar os arquivos da preparação

⚠️ git reset --hard HEAD 🡪 Apaga TUDO: o último commit; a área de preparação; as edições que você fez nos arquivos

Branches

- São ramificações do seu projeto

- Começa na branch Main por padrão

- Ex: você quer testar uma funcionalidade, mas não tem certeza se vai funcionar ou vai querer, então você cria uma nova branch paralela ao seu projeto, que será tipo uma cópia. Assim, você pode mexer sem se preocupar em afetar o projeto na branch principal

- git branch 🡪 exibe as branches do projeto

**Criar uma nova branch e entrar nela:**

git checkout -b teste 🡪 criou a branch teste

**só mudar de branch:** git checkout main

git branch -v 🡪 Lista o último commit de cada branch

**Para mesclar as Branches:** git merge teste

- Assim o commit que estava na branch teste vai para a branch main

- Se não for mais utilizar a branch teste, seu conteúdo pode ser apagado

⚠️ git branch -d teste 🡪 exclui a branch teste

**Clonar branch**: git clone link\_repo --branch nome\_branch --single branch

**Importante:** *Pesquise convenções na hora de nomear as branches e commits*

**Conflitos de Merge**

- Quando duas pessoas editam a **mesma linha de um arquivo**, o Git não consegue decidir qual versão manter automaticamente. Ao tentar enviar (fazer push) para o GitHub, você recebe uma **mensagem de conflito**  
Nesse caso, é preciso fazer um git pull primeiro para **trazer as alterações do colega**, resolver os conflitos **manualmente no arquivo** (escolhendo qual código manter), e então fazer o commit normalmente.

git fetch 🡪 Serve pra atualizar seu repositório local com as alterações do repositório remoto, mas sem aplicar essas alterações no seu código atual

Depois, se quiser aplicar as alterações git merge origin/main