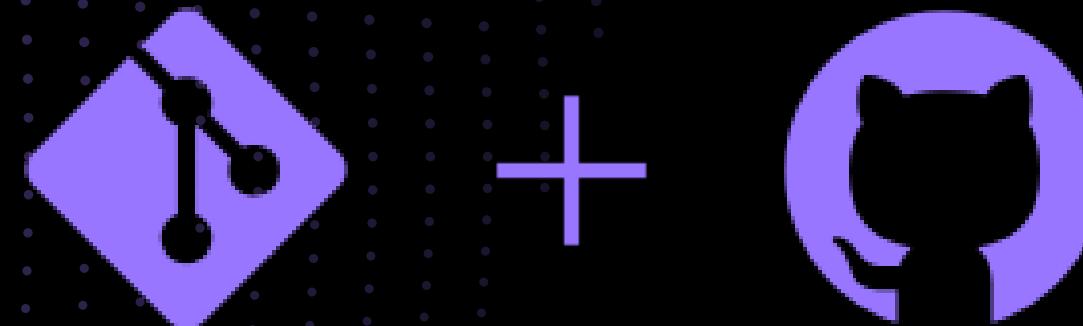


Git e GitHub

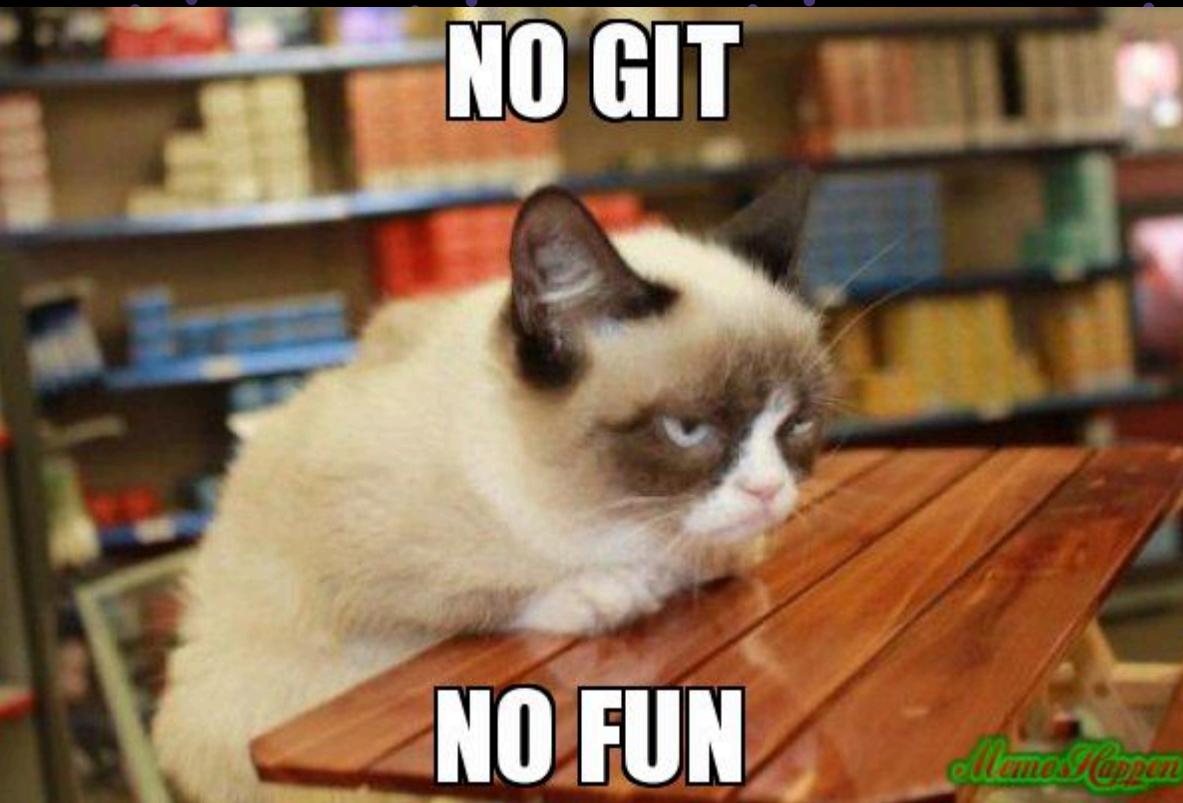


Ministrante: Kaio Christaldo do Nascimento

Introdução ao Git

O que é o Git?

- Sistema de controle de versão distribuído gratuito.
- Código aberto.
- Projetado para lidar com tudo, desde projetos pequenos a muito grandes, com velocidade e eficiência.



De onde surgiu? Como Funciona?

- Linus Torvalds em 2005
- Ele supera ferramentas SCM como Subversion, CVS, Perforce e ClearCase com recursos como ramificação local barata, áreas de teste convenientes e vários fluxos de trabalho.



Conceitos Básicos



Repositórios

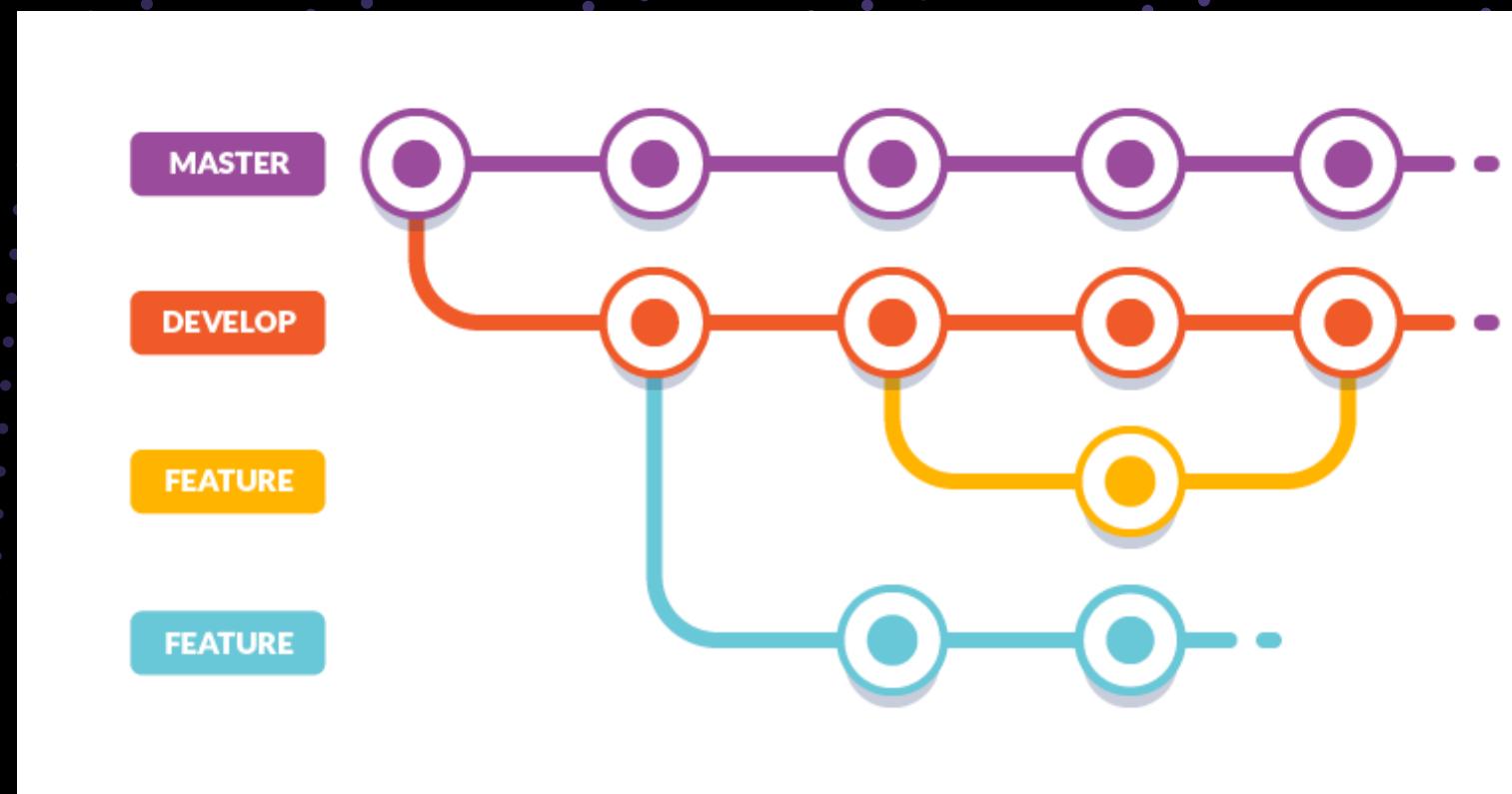
- Diretório chamado .git dentro do projeto.
- Esse repositório rastreia todas as mudanças e cria um histórico.

Commits

- Os commits marcam um ponto na história do seu projeto.
- Geralmente usados ao finalizar uma nova funcionalidade na aplicação.
- Estágios: modified, staging, committed

Branches

- Ramificações
- Times
- Versionamento



Configuração

Token

```
git config --global user.name "Meu Nome de Usuario"  
git config --global user.email "Meu Email"
```

```
git config --global credential.helper 'cache --timeout=tempos-segundos'  
git config --list
```

Criando Repositórios



Do Zero

Objetivo: Criar um novo repositório Git em um diretório vazio.

Comando: `git init`

Conteúdo Inicial: O repositório começa vazio, sem arquivos ou histórico.

Uso: Ideal para projetos novos onde você deseja iniciar do zero.

Configuração Inicial: Você pode adicionar arquivos manualmente e configurar o repositório conforme necessário.

Clone

Objetivo: Copiar um repositório Git já existente (incluindo todo o histórico) para sua máquina local.

Comando: `git clone <url-do-repositorio>`

Conteúdo Inicial: O repositório clonado contém todos os arquivos, commits e branches do repositório original.

Uso: Ideal para projetos já em andamento ou colaborativos, onde você deseja acessar o trabalho de outras pessoas.

Configuração Inicial: Após o clone, você já tem um repositório completo, pronto para ser usado.

Adicionando Arquivos

Adicionando...

Use `git add nome-do-arquivo.ext` para adicionar arquivos individuais.

Use `git add .` para adicionar todos os arquivos na pasta.

Fazendo Commits

Commitanto...

Use `git status` para verificar quais arquivos foram adicionados.
Use `git commit -m "mensagem"` para registrar as mudanças.

<https://github.com/iuricode/padroes-de-commits>

Desfazendo Commits

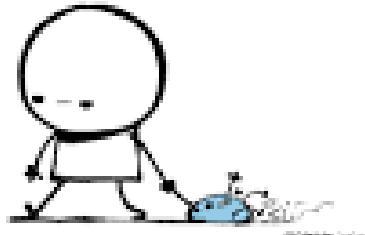
Checkout

- Navegar entre ramificações ou restaurar arquivos para um estado anterior.
- Para acessar um commit específico: `git checkout <commit-hash>`.
- Pode levar a um estado "detached HEAD" se você acessar um commit diretamente, mas não altera o histórico de commits..

Revert

- Desfazer as alterações feitas por um commit específico, criando um novo commit que reverte essas alterações.
- `git revert <commit-hash>`
- Adiciona um novo commit ao histórico, mantendo o histórico de commits intacto. É ideal para desfazer alterações em um repositório compartilhado, pois não altera o histórico anterior.

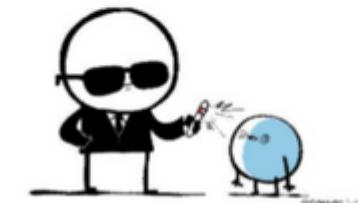
Git revert



Reset

- Reverter o repositório a um estado anterior, alterando o histórico de commits.
- Para desfazer o último commit, mantendo as alterações: `git reset --soft HEAD~1`
- Para desfazer o último commit e descartar as alterações: `git reset --hard HEAD~1`
- Para desfazer vários commits: `git reset --soft HEAD~n` (onde n é o número de commits).
- O reset pode modificar o histórico de commits e as mudanças no diretório de trabalho, podendo descartar alterações.

`Git reset --hard`



Ignorando Arquivos

Criando Branches

Branches

- Verifique branches existentes: `git branch`
- Crie uma nova branch: `git branch nome-da-nova-branch`
- Mude para a nova branch: `git checkout nome-da-nova-branch`
- Crie e mude para a nova branch: `git checkout -b nome-da-nova-branch`

Fundindo Branches

Merge

- Verifique branches existentes: `git branch`
- Mude para a branch de destino: `git checkout nome-da-branch`
- Mescle a branch de origem: `git merge nome-da-branch`
- Resolva conflitos se necessário e finalize com `git commit`.



Resolvendo Conflitos

Conflitos

- Um conflito ocorre quando duas branches modificam a mesma linha de um arquivo.
- Você deve editar o arquivo manualmente para resolver o conflito e, em seguida, adicionar e comitar as alterações.

```
Linha 1
<<<<< HEAD
Linha 2 - Alteração da branch main
=====
Linha 2 - Alteração da branch feature
>>>>> feature
Linha 3
```

Introdução ao GitHub



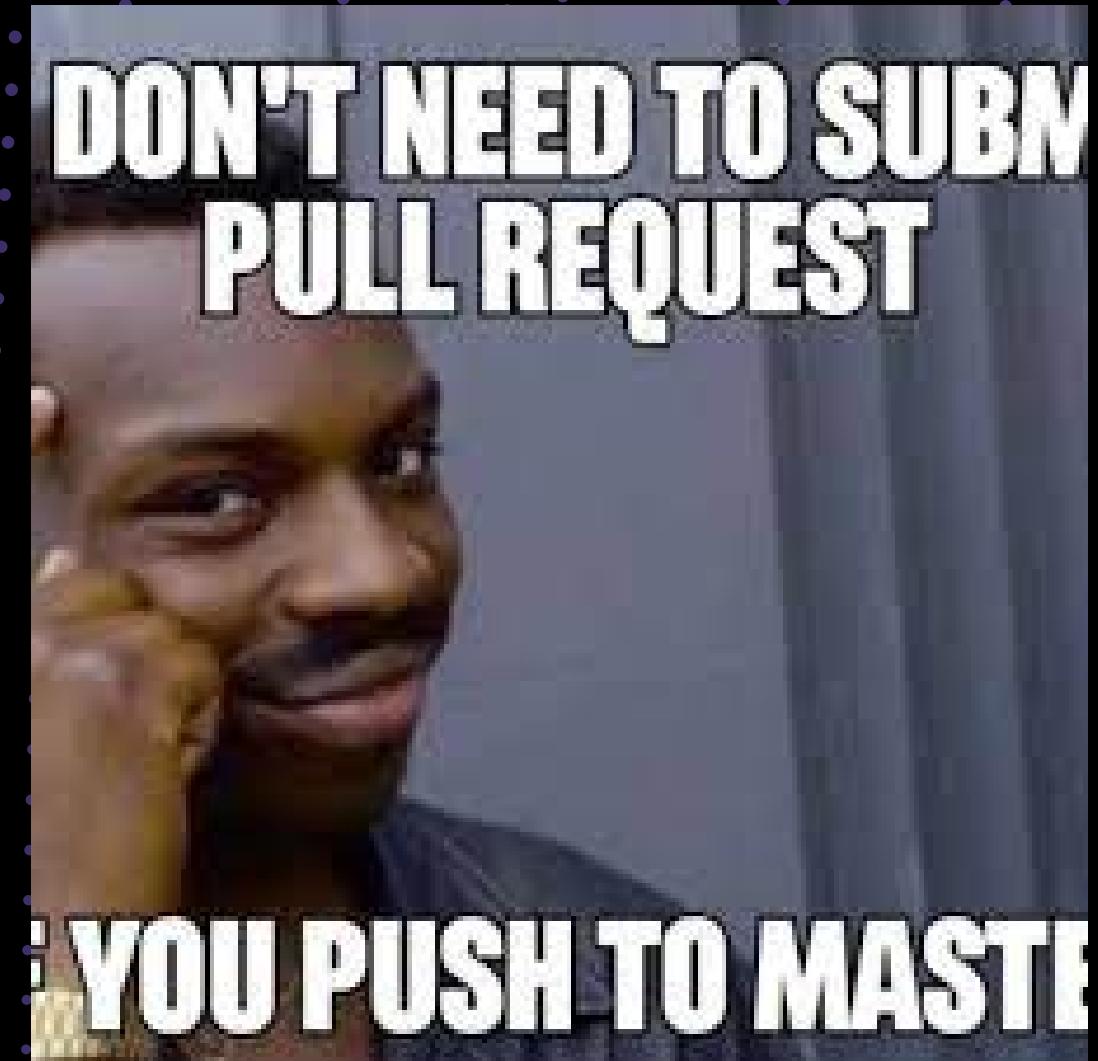
Criar uma conta

- Sem muito segredo
- Configurar token de acesso

Fazendo Pull Request

Pull Request

- Crie uma nova branch: `git checkout -b nome-da-nova-branch`
- Faça alterações e commit: `git add` e `git commit -m "mensagem"`
- Envie a branch para o remoto: `git push origin nome-da-nova-branch`
- Crie o Pull Request na plataforma de controle de versão.



Fazendo Fork de um Repositório

Práticas

- Adicionando pessoas no seu repositorio do github
- Fork no meu repositorio da sic

Dicas para aprender +

- Exercitar através de projetos, contribuições em repositórios públicos.
- Jogar joguinhos de aprendizado de git
- Buscar conteúdos de git no youtube

Coo

- Exercitar através de projetos, contribuições em repositórios públicos.
- Jogar joguinhos de aprendizado de git
- Buscar conteúdos de git no youtube