

Table of Contents

Git Pie: Aprenda sobre VCS	2
Conceitos Básicos de Versionamento	8
Tipos de Sistemas de Controle de Versão	11
História do Git	14
Conceitos Básicos do Git	17
Fluxo de Trabalho do Git	19
Comandos Essenciais do Git	21
Links e Referências	23

Git Pie: Aprenda sobre VCS



American pie

Nota do Autor

Olá pessoas, nesse texto irei falar sobre VCS (Sistema de Versionamento de Código, sigla em inglês) ou melhor, como o tema é mais conhecido - falarei sobre Git.

O que você vai aprender aqui?

⚠ "Deixa que o Stifler te explica essa parada!"

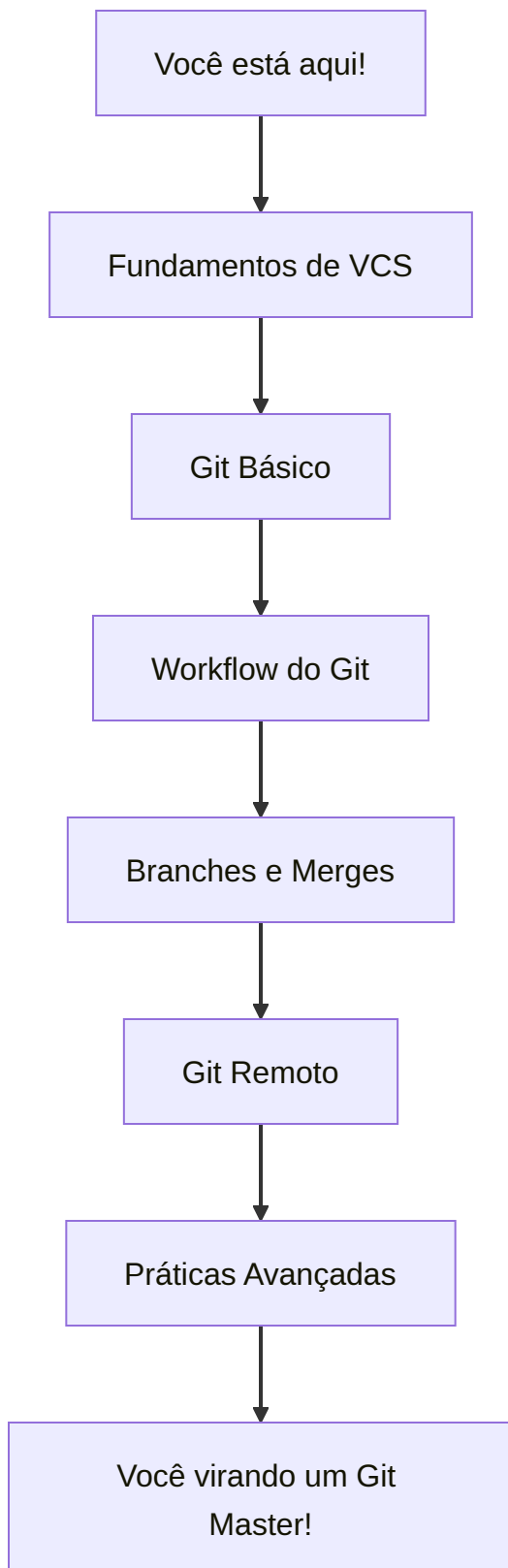
Nesse guia você vai aprender:

- Como não perder código igual perdeu aquela crush do ensino médio
- Como trabalhar em equipe sem querer matar seus colegas
- Como versionar código igual um profissional (e não usando projeto-final-v3-agora-vai-mesmo.zip)
- Como usar Git e não passar vergonha nas entrevistas de emprego

Roadmap de Aprendizado

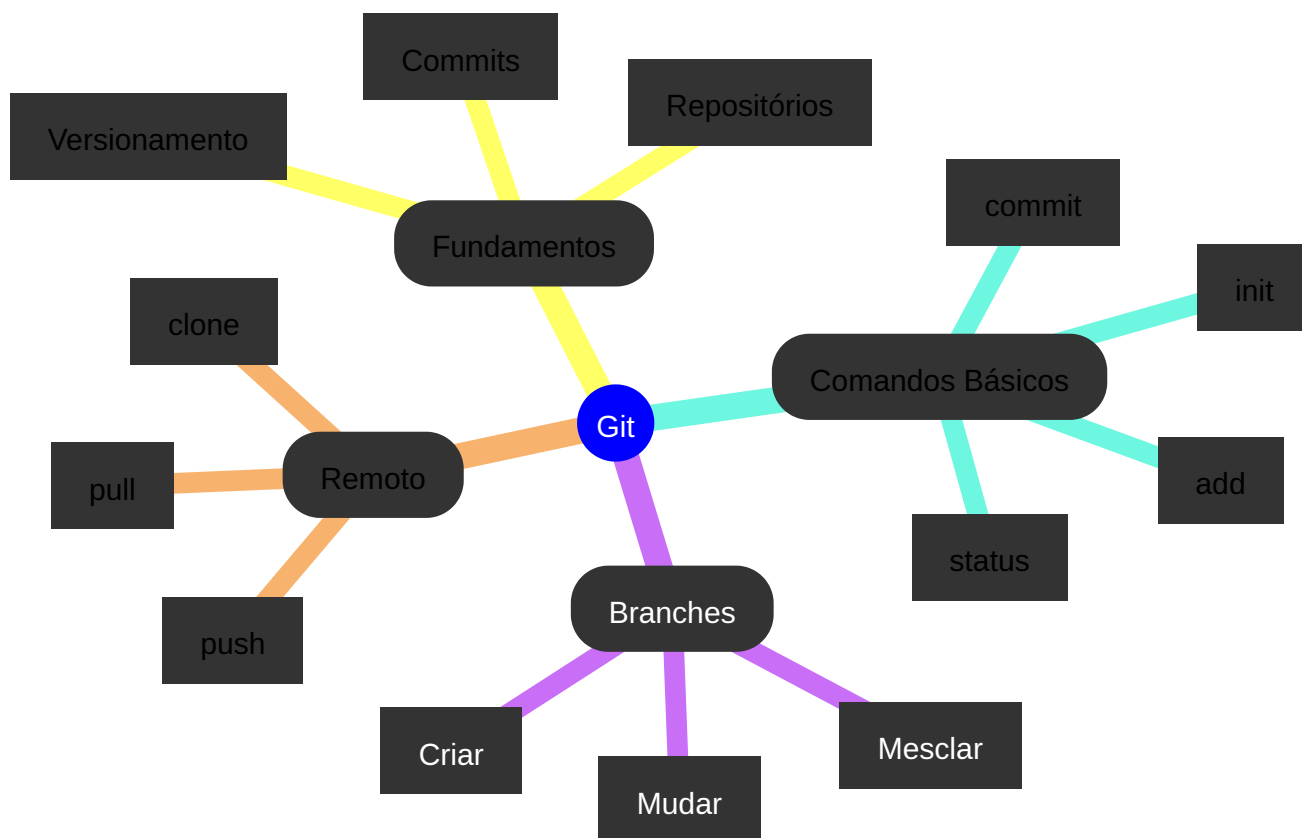


A estrada do conhecimento é longa, mas é divertida!



Mapa Mental dos Conceitos

⚠ Para você que gosta de ver o todo antes de se perder nos detalhes (tipo quando você olha o cardápio inteiro antes de pedir)



Por que você deveria aprender Git?

⚠ "Confia no pai que essa é boa!"

Imagina só:

- Você tá lá, codando tranquilo
- Fez alterações MASSAS no projeto
- Aí seu PC resolve dar aquela travada marota
- E... BOOM! 💣 Perdeu tudo!

Ou pior:

- Você e seu amigo precisam trabalhar no mesmo projeto
- Vocês ficam trocando arquivo por WhatsApp
- projeto_final.zip, projeto_final_v2.zip, projeto_final_v2_agora_vai.zip
- No final ninguém sabe qual é a versão certa 🧑

É aí que entra o Git! Ele é tipo aquele amigo que:

- Guarda todas as versões do seu código
- Deixa você voltar no tempo quando der m*rda
- Permite que você e seus amigos trabalhem juntos sem criar caos
- Te salva de passar vergonha em entrevistas de emprego

Pré-requisitos



"O que eu preciso saber antes de começar?"

- Saber usar um terminal básico (tipo `cd`, `ls`, essas coisas)
- Ter um editor de código (VSCode, Sublime, ou qualquer outro que você curta)
- Vontade de aprender (e senso de humor para aguentar minhas piadas ruins)

Como usar este guia

Este material está organizado de forma progressiva:

1. Começamos com o básico dos básicos
2. Vamos evoluindo aos poucos
3. No final você estará usando Git igual um profissional

⚠ Dica do Stifler: Não pule etapas! É tipo American Pie, você precisa ver o primeiro filme antes de entender as piadas do segundo!

Bora começar?

⚠ É hora de botar a mão na massa!

Escolha sua aventura:

- Fundamentos de Versionamento ([Conceitos Básicos de Versionamento](#)) - Para entender o básico
- História do Git ([História do Git](#)) - Para os curiosos
- Git na Prática ([Fluxo de Trabalho do Git](#)) - Para quem quer ir direto ao código

⚠ Nota: Se em algum momento você se perder, não se preocupe! É normal, todo mundo já passou por isso. Até o Stifler já perdeu código antes de aprender Git!

Conceitos Básicos de Versionamento

Versionamento de Código

Versionamento é um conceito muito simples e usado no dia a dia de forma que nem percebemos. Por exemplo: Estamos em um projeto onde temos dois desenvolvedores:

- Stifler



Stifler dude no

- Jim



Jim american pie

Esses dois desenvolvedores estão fazendo o "Milfs Go" uma especie revolucionaria e inovadora, além do tempo sendo um *app* para acharem a "milfs".



Aqui está uma *milf* para aqueles não habituados com o termo:



American pie good stuff

Controle de Versão

Versionamento é o ato de manipular versões, agora o Controle de Versão é um sistema que vai registrar as mudanças tanto num arquivo como em um projeto gigante ao longo do tempo.

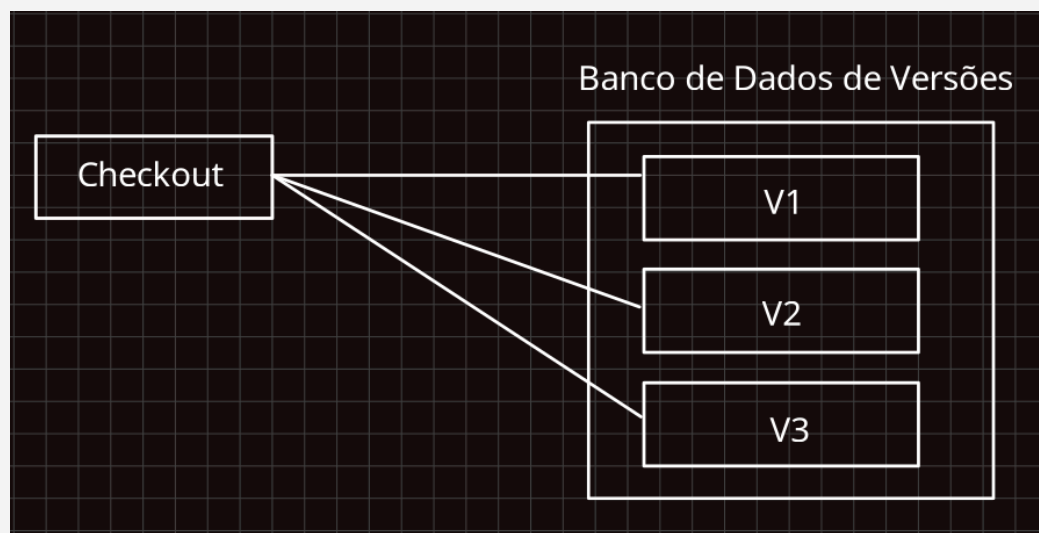
Importância

Talvez agora você levante uma questão de o porque aprender "este trem" - como diria um amigo mineiro. Logo, a resposta é simples: esse tipo de ferramenta é essencial para o desenvolvimento já que nos entrega um poder de não somente trabalhar em conjunto de forma assíncrona e sem medo de acabar perdendo o que já foi feito.

Tipos de Sistemas de Controle de Versão

Sistemas Locais

Esse tipo de sistema é mantido em uma máquina. Por exemplo, Jim vai fazer o versionamento da sua parte do *frontend*, onde ele possui um arquivo de *checkout* que vai servir para conferir/adicionar as versões e um banco de dados (poderia ser um outro arquivo) contendo as versões que ele salvou.

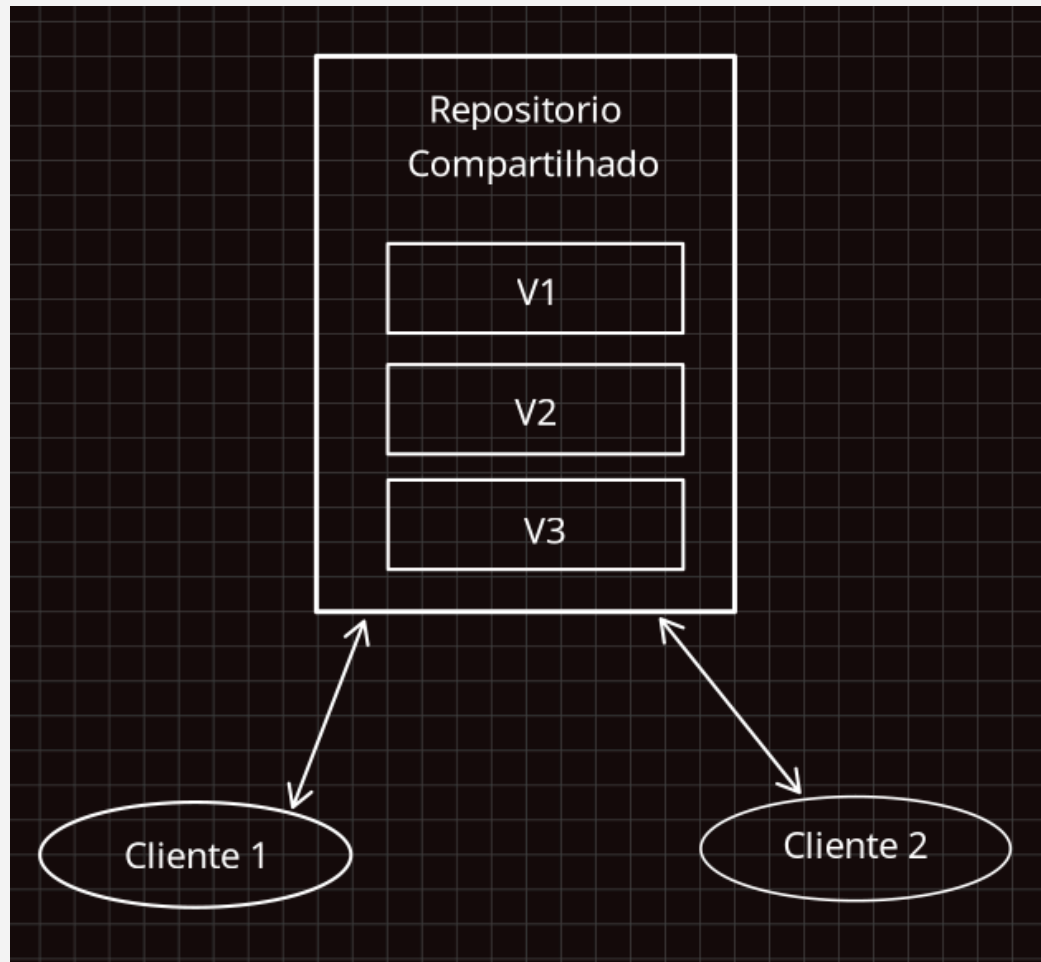


Version control system sistema local

- Diagrama de um sistema local

Sistemas Centralizados

Estes sistemas nascem com a problemática que o Sistema Local trás que é justamente um não compartilhamento simultâneo, já que como no nosso exemplo esses dois teriam problemas de versões já que estarão em computadores diferentes.

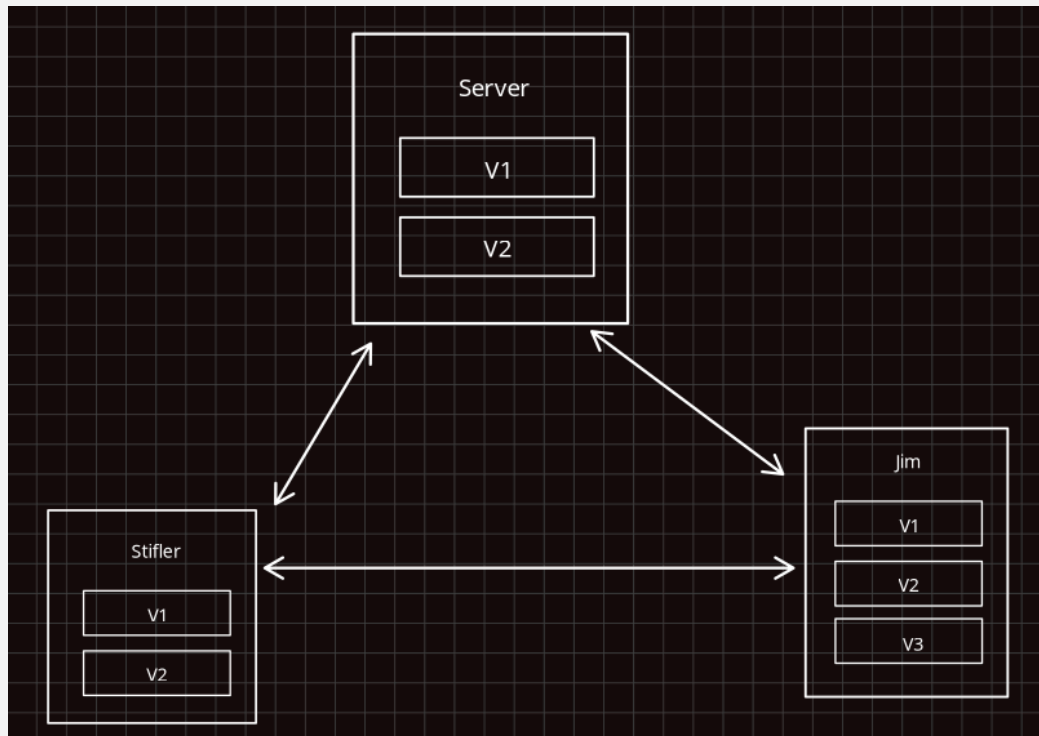


Version control system sistema compartilhado

- Diagrama de um sistema compartilhado

Sistemas Distribuídos

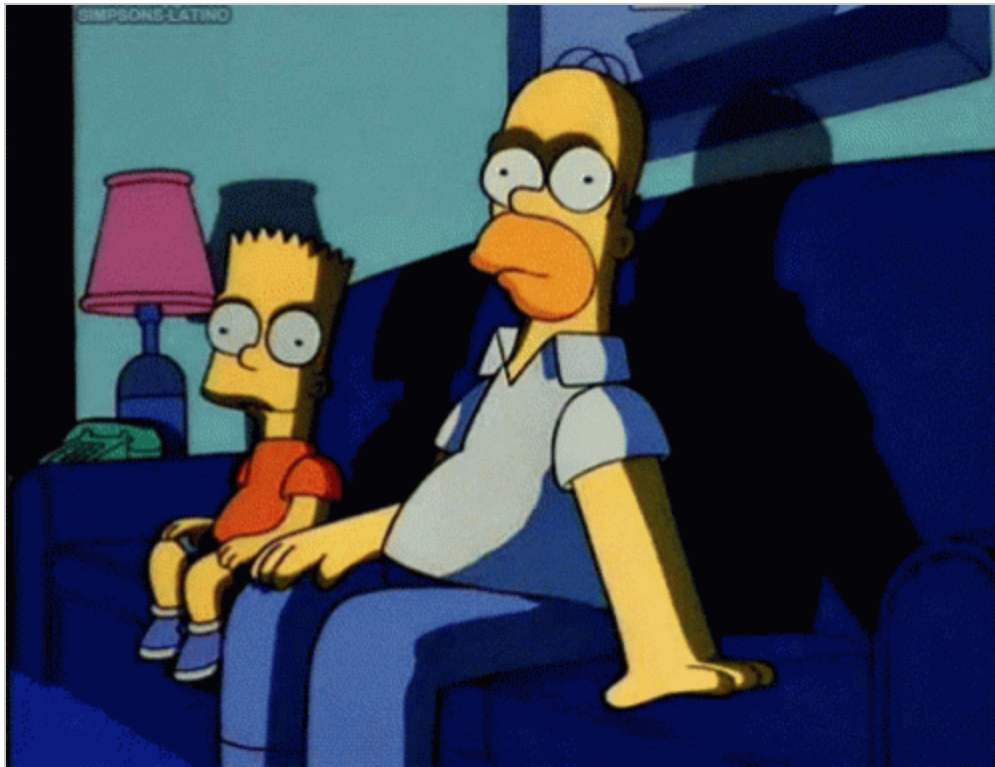
Com isso os DVCS (Sistemas de Controle de Versão Distribuídos) se tornam um protagonista, já que os clientes não somente clonam os estados atuais, mas também fazem uma cópia completa de todo o repositório localmente.



Version control system sistemasdistribuidos

- Diagrama de um sistema distribuído

História do Git



The simpsons homer

Para começar a história do Git é até bem curta e direta. A comunidade do Linux usava um VCS distribuído chamado **BitKeeper** só que ele é proprietário.

Sim, um sistema open source usando um proprietário. Claramente isso era algo que causava um estranhamento na comunidade.



Stifler kiss

Que por sua vez chegou ao ápice quando o BitKeeper se tornou pago, logo a comunidade do Linux ficou alerta já que eles teriam que fazer o versionamento do núcleo do Linux em outro sistema.

Assim então a comunidade começou a criar seu próprio VCS que fosse:

- Simples
- Veloz
- Não linear, ou seja, que aceite vários ramos (**branches**) de modificação
- Capaz de lidar com grandes projetos, afinal, Linux é gigante

E assim nasceu o Git, exatamente em 2005 e até hoje está em evolução sendo um dos VCS mais utilizados em todo o mundo de desenvolvimento de gambiarras (softwares).



Ou seja, tudo nasceu de uma revolta popular



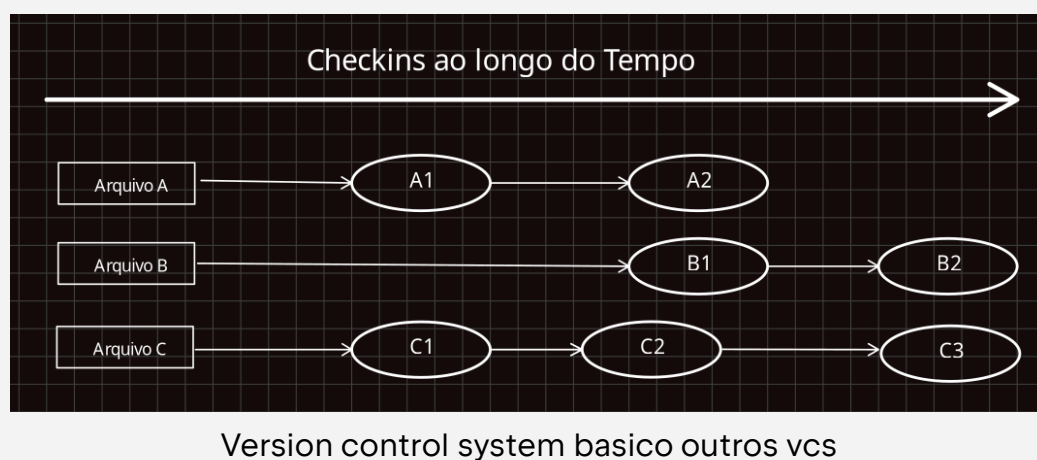
Cachorro comuna

Conceitos Básicos do Git

Como o Git Funciona

O Git funciona de forma diferente de outros VCS. Em um outro VCS ele terá os arquivos e quando houver alteração eles criam uma lista somente das alterações.

Em um outro VCS ele terá os arquivos e quando houver alteração eles criam uma lista somente das alterações:



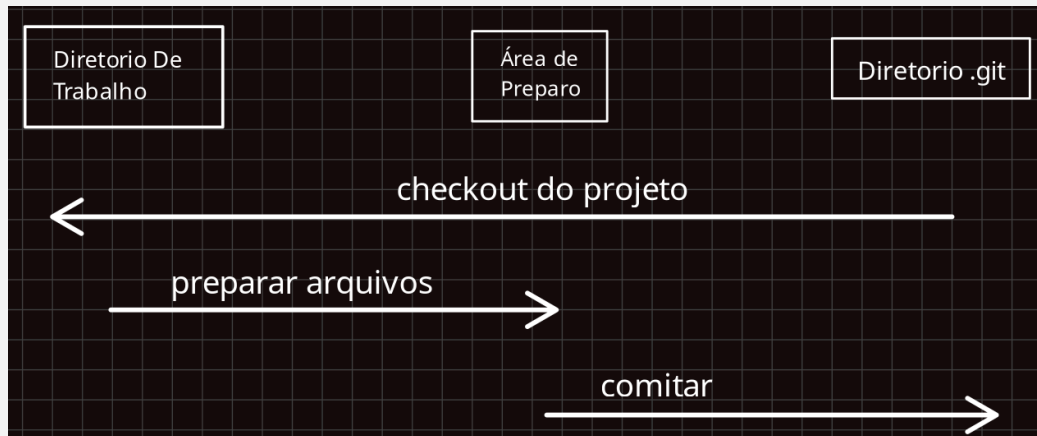
Agora com o Git ele faz diferente, já que vai tirando *snapshots* que são como fotos quando ocorre uma mudança e caso tenha algum arquivo que não foi alterado será guardado uma referencia para ele, assim pode ser recuperado.

Estrutura de Diretórios

Assim temos dois níveis principais:

- Diretório de trabalho
- Área de preparo
- Diretório `.git` que vai ser o repositório ou banco de dados local





Version control system fluxodetrabalho

Diretórios quando se trabalha com Git

Fluxo de Trabalho do Git

Iniciando um Repositório

Devemos usar o comando abaixo para iniciar o repositório para que o Git consiga ver os arquivos.

```
md MilfsGo # Cria a pasta
cd MilfsGo # acessa a pasta
git init
```

Fazendo Alterações

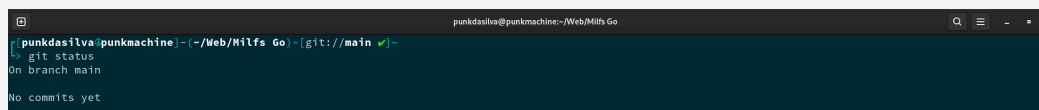
Agora vamos fazer alterações básicas como adicionar um *README* para o projeto.

⚠ README são arquivos geralmente em markdown (.md) para registrar a documentação do repositório com informações importantes como:

- Nome
- Descrição
- Como usar
- Etc

Verificando Status

```
git status
```

A screenshot of a terminal window showing the output of the 'git status' command. The prompt is 'punkdasilva@punkmachine' and the directory is '~/Web/Milfs Go'. The output shows 'On branch main' and 'No commits yet'. The terminal window has a title bar with 'punkdasilva@punkmachine - ~/Web/Milfs Go' and standard window controls.

Version control system gitstatus

Resultado da execução do comando

Comandos Essenciais do Git

Cheat Sheet (Tabela de preguiçoso)



American pie its not what it looks like

Essa tabela fornece uma visão geral dos principais comandos Git e suas funcionalidades básicas.

Comando Git	Descrição
<code>git init</code>	Inicializa um novo repositório Git
<code>git add <arquivo></code>	Adiciona um arquivo modificado à área de stage
<code>git add .</code>	Adiciona todos os arquivos modificados à área de stage
<code>git commit -m "Mensagem do commit"</code>	Cria um novo commit com a mensagem especificada
<code>git mv <arquivo-original> <arquivo-novo></code>	Renomeia ou move um arquivo no repositório

Links e Referências

- GIT-SCM.COM. Git - Documentation. Disponível em: <https://git-scm.com/doc> (<https://git-scm.com/doc>).
- YOUTUBE. YouTube. Disponível em: <https://www.youtube.com/watch?v=un8CDE8qOR8> (<https://www.youtube.com/watch?v=un8CDE8qOR8>).
- GITLAB. GitLab Documentation. Disponível em: <https://docs.gitlab.com/> (<https://docs.gitlab.com/>).
- GITHUB. Git Cheat Sheet. Disponível em: <https://education.github.com/git-cheat-sheet-education.pdf> (<https://education.github.com/git-cheat-sheet-education.pdf>).