



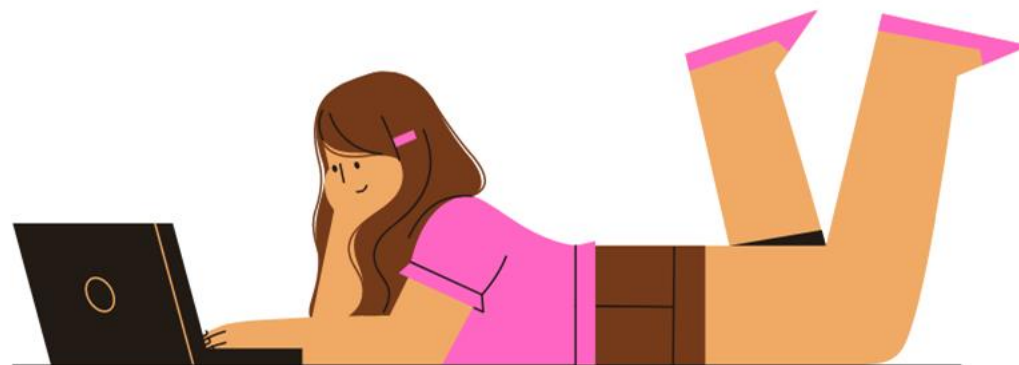
Powered by  serasa



# Sistemas de controle de versões

É um sistema que grava e gerencia alterações em um arquivo ou conjunto de arquivos.

Ele permite que você reverta para estado anterior, compare mudanças feitas, veja quem e quando algum arquivo foi modificado, e muito mais.



# Hospedagem de repositórios Git

Aplicações que possibilitam a criação de um servidor para a hospedagem de repositórios Git.



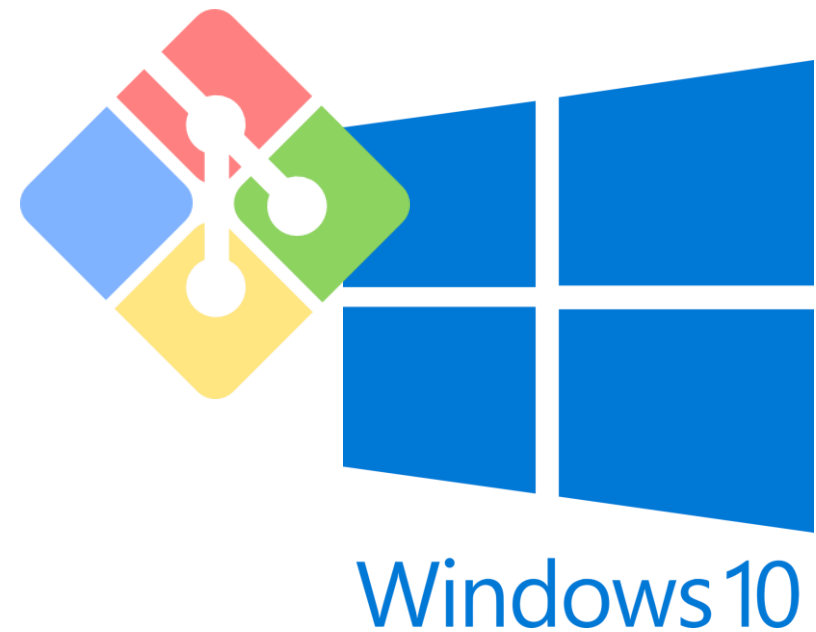


# Instalando e configurando o Git

# Instalação no Windows

Acesse <https://git-scm.com/downloads> e baixe a última versão disponível.

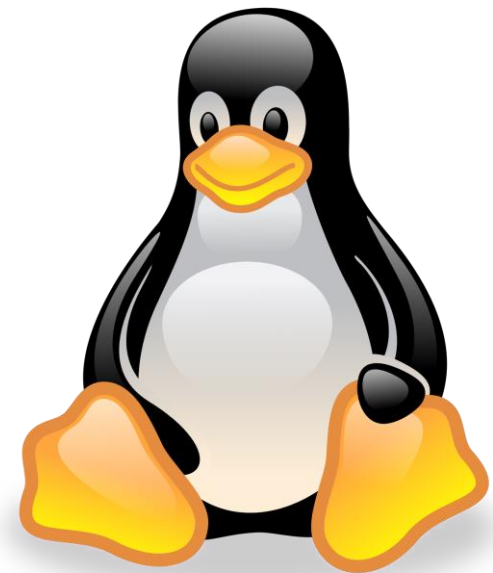
Manteremos a instalação padrão, clicando em "Next" em todas as etapas do instalador.



# Instalação no Linux

Em algumas distribuições não é necessária, pois ele já vem instalado.

Caso não seja o caso de sua distribuição, confira as instruções em <https://git-scm.com/download/linux>



# Instalação no MacOs

Confira as instruções em <https://git-scm.com/download/mac>



Mac<sup>TM</sup>OS





# Verificando se o Git foi instalado corretamente

```
git --version
```



# Configurando nosso usuário

```
git config --global user.name <seu nome>
```

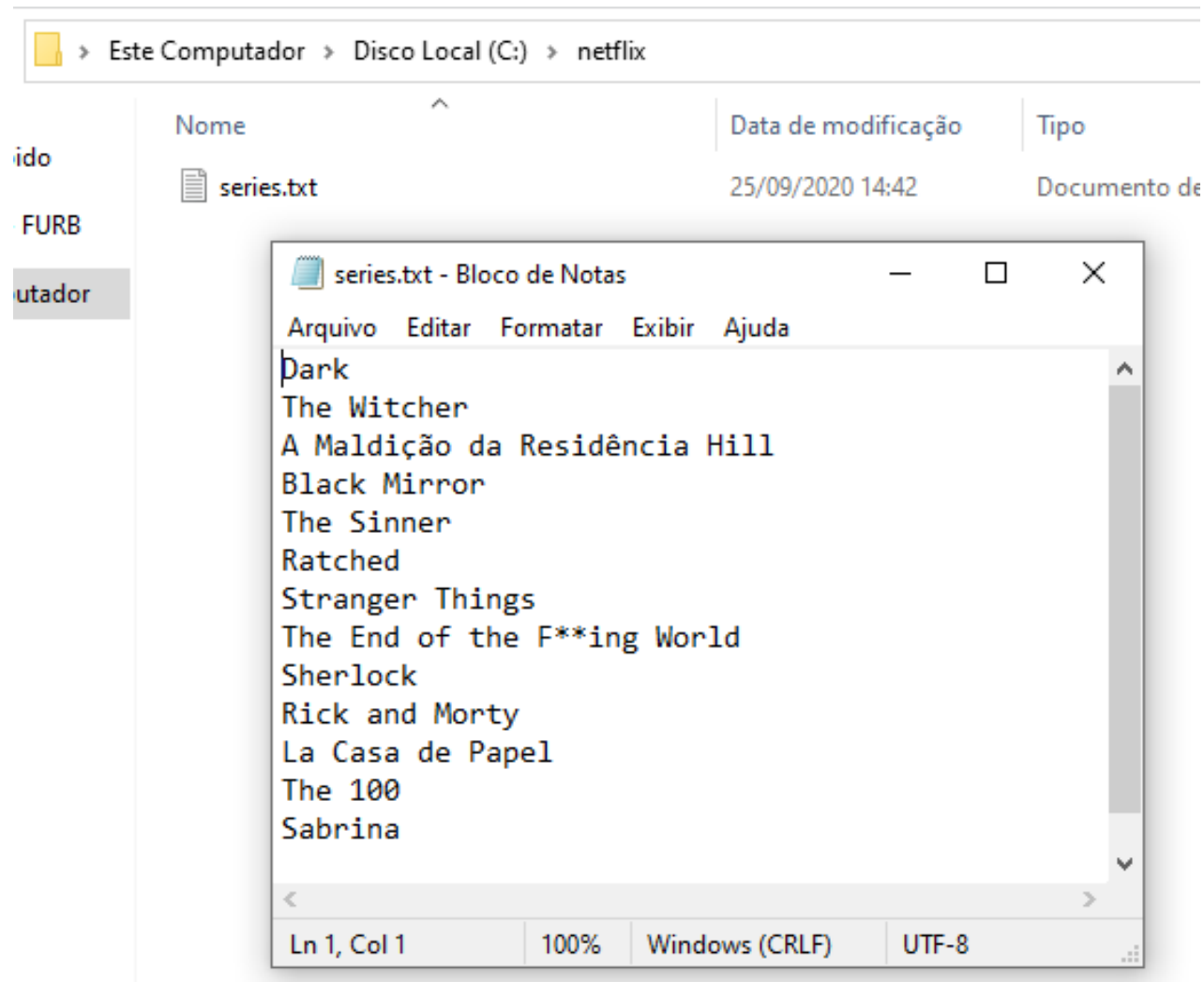
```
git config --global user.email <seu e-mail>
```



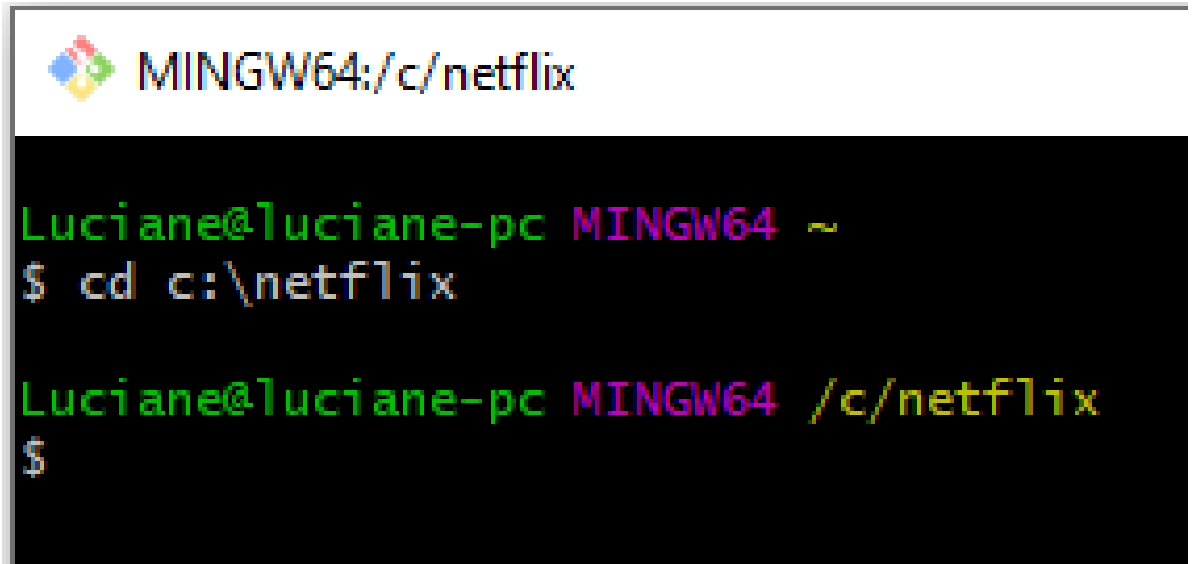


# Tour prático

# Criando o projeto



# Acessando o diretório do projeto



```
MINGW64:/c/netflix

Luciane@luciane-pc MINGW64 ~
$ cd c:\netflix

Luciane@luciane-pc MINGW64 /c/netflix
$
```



# Criando o nosso repositório

```
git init
```



# Rastreando o arquivo

Vamos ver a situação do arquivo no nosso repositório Git

```
git status
```

Agora vamos rastrear o arquivo

```
git add <nome do arquivo>
```

```
git add .
```



# Gravando o arquivo no repositório

Vamos realizar um commit

```
git commit -m "escreva aqui um comentário"
```

.





# Alterando o arquivo

Vamos adicionar uma nova série no arquivo,

Para monitorarmos a modificação, devemos executar o comando abaixo novamente

```
git add <nome do arquivo>
```

E agora vamos realizar um commit dessa alteração

```
git commit -m "escreva aqui um comentário"
```



# Verificando alterações realizadas

Para verificar o histórico das alterações gravadas no repositório, podemos executar o comando

```
git log
```

Veja como montar utilizando filtros: <http://devhints.io/git-log>



# Ignorando arquivos

Para ignorar determinados arquivos, precisamos criar um arquivo chamado `.gitignore` no diretório principal do nosso projeto, e adicionar o nome ou extensões do que queremos ignorar.

Há um projeto no GitHub com exemplos de arquivos `.gitignore` para diversas linguagens de programação e tecnologias: <https://github.com/github/gitignore>

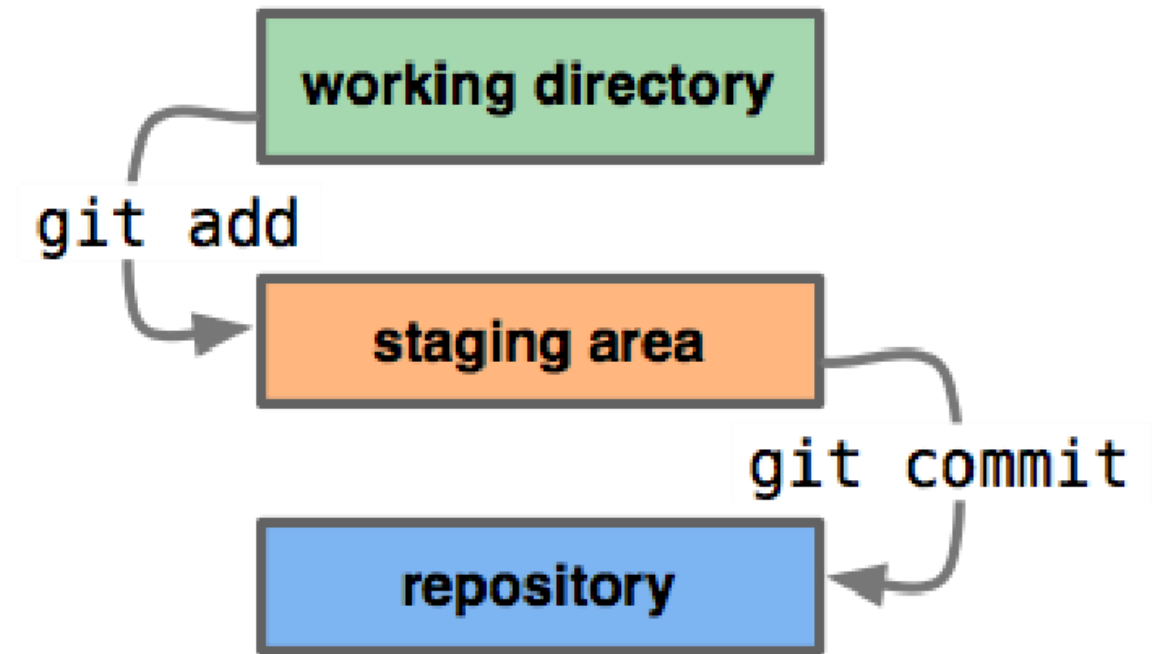


# Fluxo de trabalho

**Working Directory** - contém os arquivos vigentes.

**Stage ou Index** - funciona como uma área temporária.

**Repository ou Head** - aponta para o último commit que você fez.





Compartilhando o  
trabalho

# Hospedando o projeto remotamente

Todos os commits estão registrados no nosso computador. Para que outras pessoas da equipe tenham acesso ao projeto, temos que compartilhá-lo. Para isso, iremos enviar nosso projeto para uma aplicação web chamada GitHub.



# Criando sua conta no GitHub

Acesse <https://github.com/> e crie sua conta.



# Dica

GitHub Student Developer Pack oferece acesso gratuito a ferramentas pagas para estudantes.

Acesse <https://education.github.com/pack/offers> e verifique os benefícios.



# GitHub Student Developer Pack

DEV





# Criando um repositório no GitHub

Crie um novo repositório no GitHub para armazenar seu projeto

.



# Apontando seu projeto para o GitHub

Devemos agora apontar o repositório da nossa máquina para o repositório do GitHub

```
git remote add origin <endereço repositório do  
GitHub>
```

.



# Enviando as alterações para o GitHub

Com o repositório remoto configurado, podemos enviar nossas mudanças para o GitHub.

```
git push origin master
```

Forneça seu usuário e senha do GitHub quando solicitado.



# Obtendo projeto do GitHub

Com o projeto no GitHub, qualquer um pode acessar o código e ver o histórico.

Para baixar, crie um novo diretório e execute

```
git clone <endereço do repositório do GitHub>
```



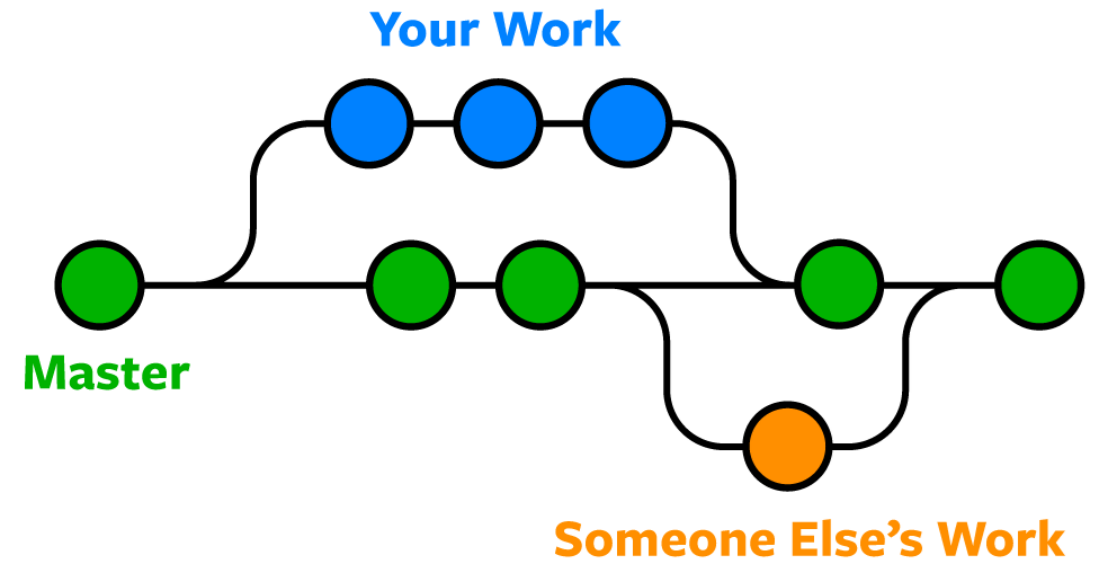


Trabalhando em  
equipe

# A branch master

O Git permite trabalho em paralelo através de branches. Uma branch é uma linha paralela de desenvolvimento em que podemos comitar novas versões do código sem afetar outras branches.

Git possui por padrão uma branch principal chamada master.



# Criando uma Branch

Para listar as branches do nosso repositório

```
git branch
```

Para criar uma nova branch

```
git branch <nome da branch>
```

.

.



# Trocando de branch

Já criamos nossa branch mas ainda estamos na master. Para trocarmos devemos executar:

```
$ git checkout <nome da branch>
```

.





# Deletando uma branch

Não é possível remover uma branch enquanto estivermos nela. Por isso, devemos ir para outra branch.

Para deletar uma branch, devemos executar

```
git branch -d <nome da branch>
```

Para deletar uma branch onde foi realizado um commit, devemos executar:

```
git branch -D <nome da branch>
```



# Mesclando alterações

Vamos enviar as alterações que fizemos na nossa branch para a branch master

```
git merge <nome da branch> -m "Merge com a  
branch criada"
```





# Resolvendo conflito

# Resolvendo conflito

