



Git x GitHub - Fundamentos

Programação III

Prof. Edson Mota, PhD, MSc, PMP

O que são essas tais
ferramentas **GIT**?



“

“Git é um sistema de controle de versões distribuído, rápido e escalável.”

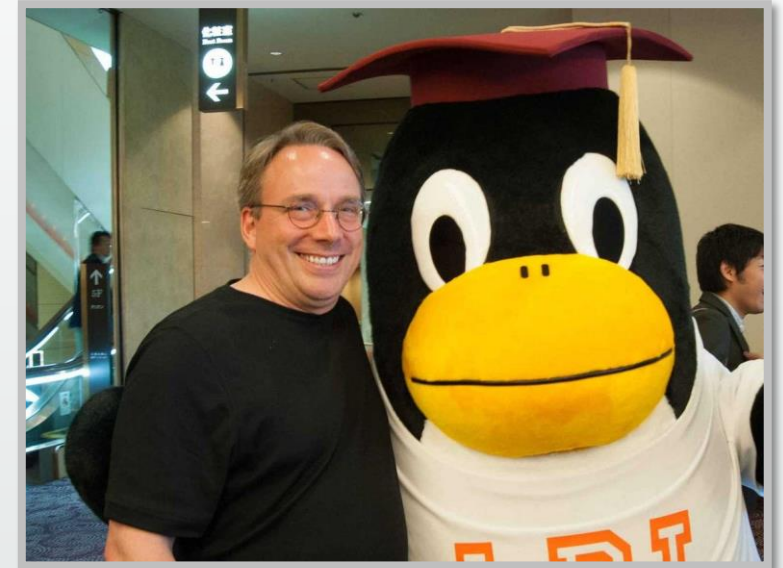
O que é Git

- Git é um “versionador”
- Permite gerenciar versões de arquivos em desenvolvimento
- Permite rastrear todas as mudanças mantendo um histórico sempre acessível
- Prevê a restauração de versões anteriores
- Permite a colaboração em desenvolvimento em um mesmo repositório, muitas vezes no mesmo arquivo
- Prevê um sistema de gerenciamento de versões eficiente, retirando do desenvolvedor preocupações relativas à sincronização e consistência dos arquivos



Git

- Desenvolvido por Linus Torvalds (Criador do Linux)
- A demanda veio da necessidade de criar um sistema de colaboração para desenvolvimento do ambiente Linux .



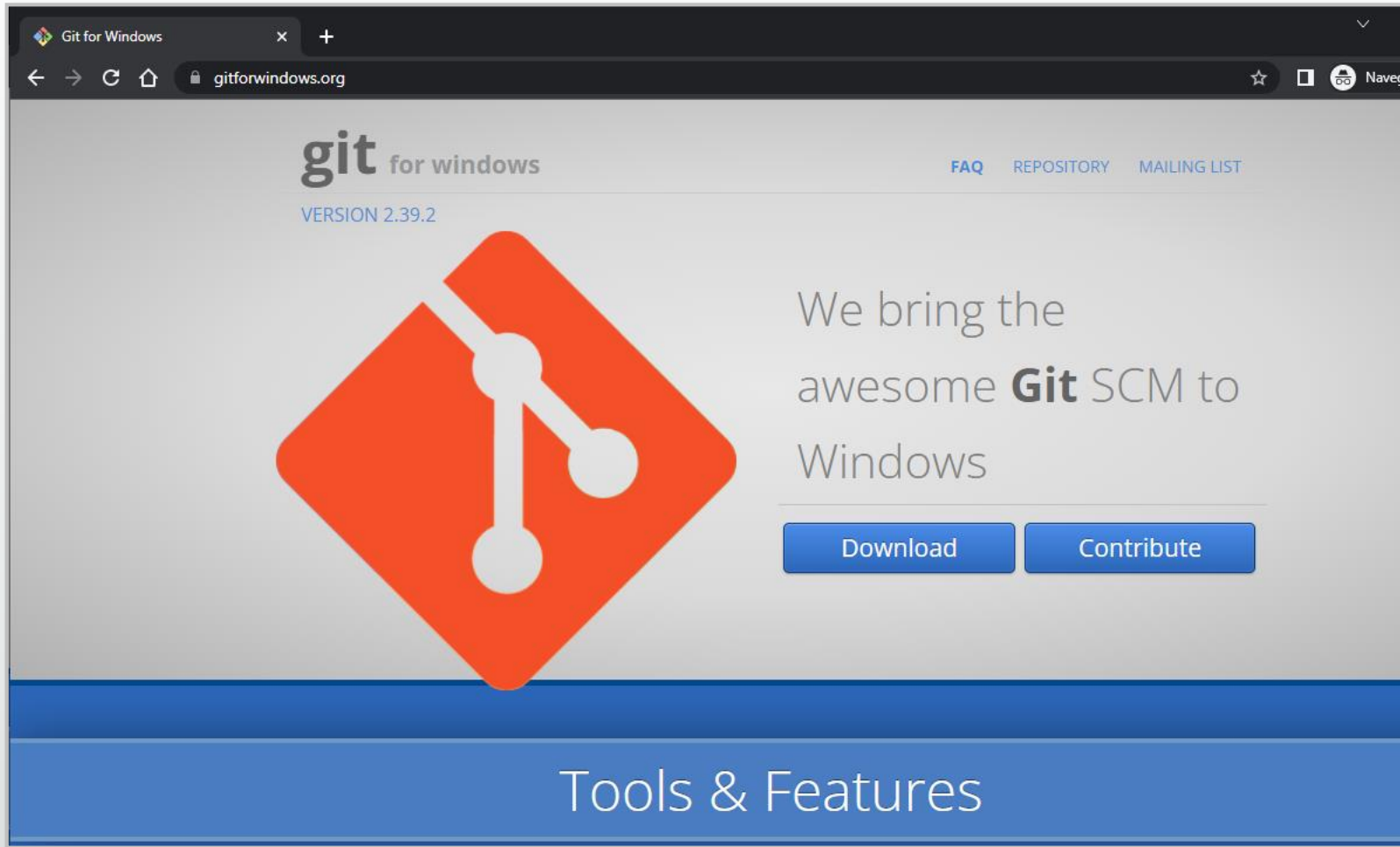
Linus Torvalds

Terminologia Git – Principais

- Os usuários do mundo Git possuem um vocabulário muito particular
 - **Repository:** Local onde ficarão todos os nossos códigos
 - **Commit:** Coleções de alterações realizadas
 - **Branch:** Uma ramificação de seu projeto, a cada branch abre-se uma nova ramificação independente
 - **Fork:** Uma bifurcação do projeto, copia-se o projeto existente para seguir em outra direção
 - **Merge:** Incorporação de alterações a partir de diferentes branches (ramificações)

**Vamos rodar alguns
comandos?**

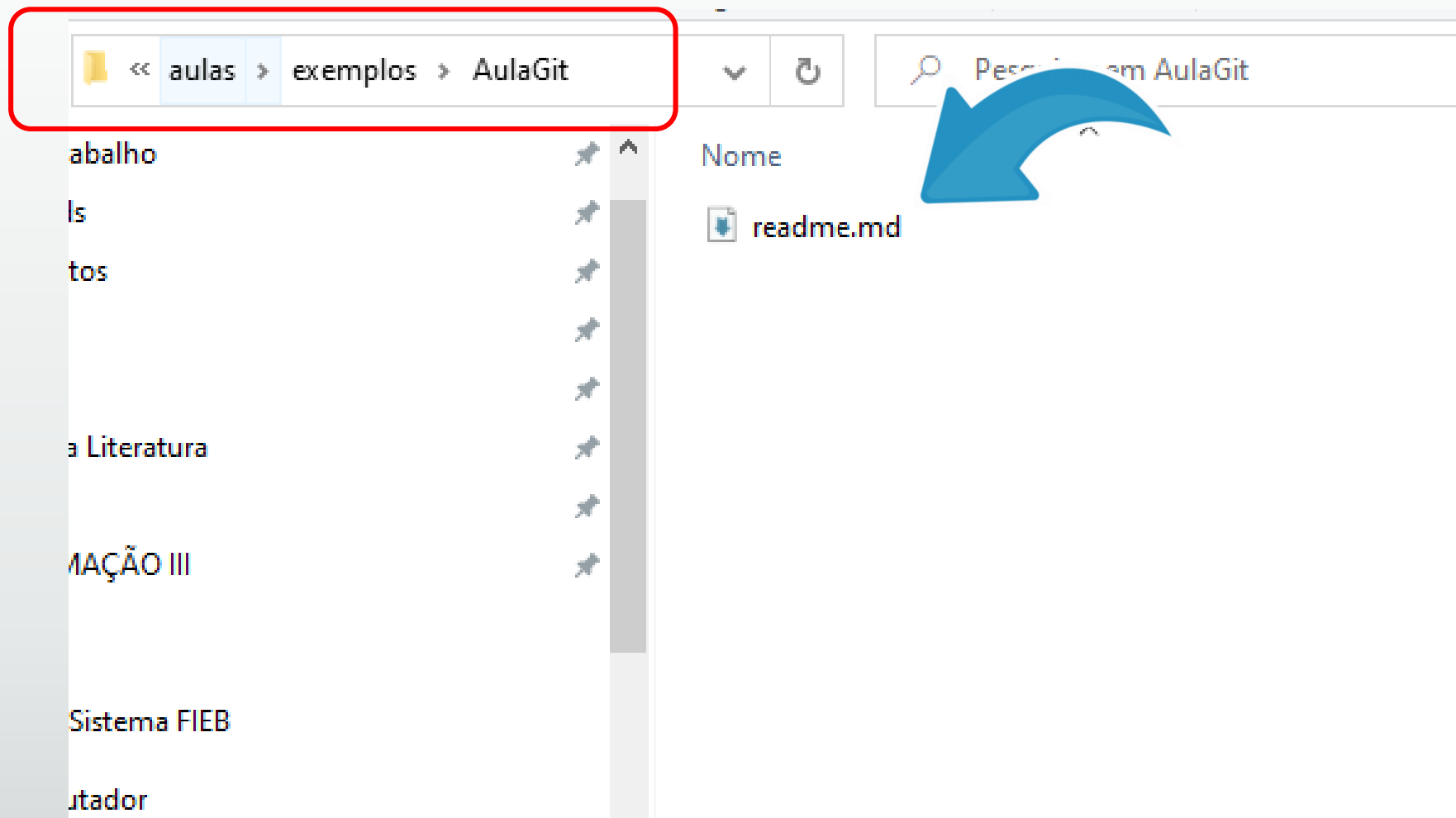
Instale o Git



Faça o download e siga os passos da instalação

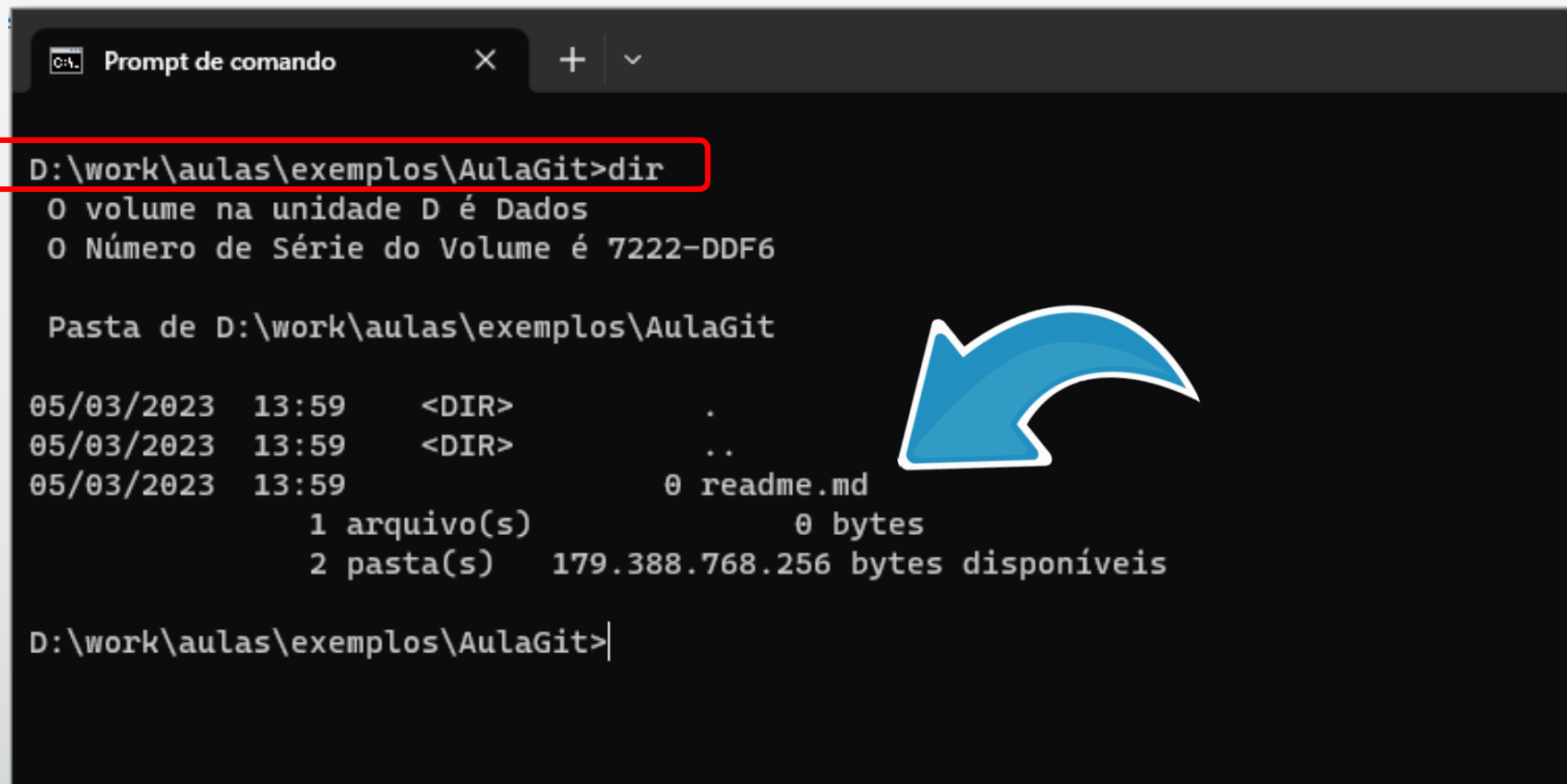
<https://gitforwindows.org/>

Criando Projetos



Prompt de Comando

- Importante se familiarizar com o **prompt de comando**



```
Prompt de comando
D:\work\aulas\exemplos\AulaGit>dir
O volume na unidade D é Dados
O Número de Série do Volume é 7222-DDF6

Pasta de D:\work\aulas\exemplos\AulaGit

05/03/2023  13:59    <DIR>          .
05/03/2023  13:59    <DIR>          ..
05/03/2023  13:59                0 readme.md
                1 arquivo(s)                0 bytes
                2 pasta(s)  179.388.768.256 bytes disponíveis

D:\work\aulas\exemplos\AulaGit>
```

O Git está instalado?

- Escreva Git no prompt e verifique o resultado
- Se for similar ao resultado ao lado, o git esta devidamente instalado

```
C:\> Prompt de comando
D:\work\aulas\exemplos\AulaGit>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        [--super-prefix=<path>] [--config-env=<name>=<envvar>]
        <command> [<args>]

These are common Git commands used in various situations:

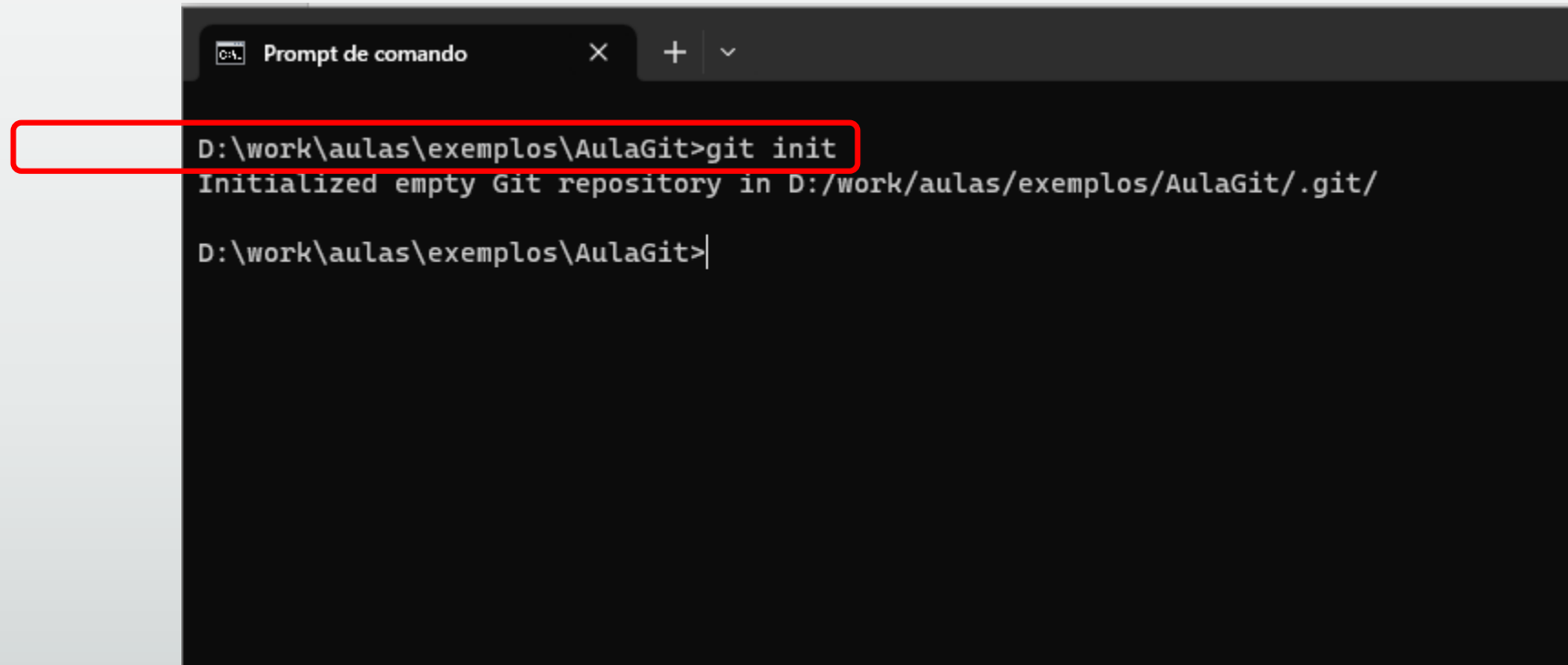

start a working area (see also: git help tutorial)
    clone      Clone a repository into a new directory
    init       Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
    add        Add file contents to the index
    mv         Move or rename a file, a directory, or a symlink
    restore    Restore working tree files
    rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
    bisect     Use binary search to find the commit that introduced a bug
    diff       Show changes between commits, commit and working tree, etc
    grep       Print lines matching a pattern
    log        Show commit logs
    show       Show various types of objects
    status     Show the working tree status
```

Git Init

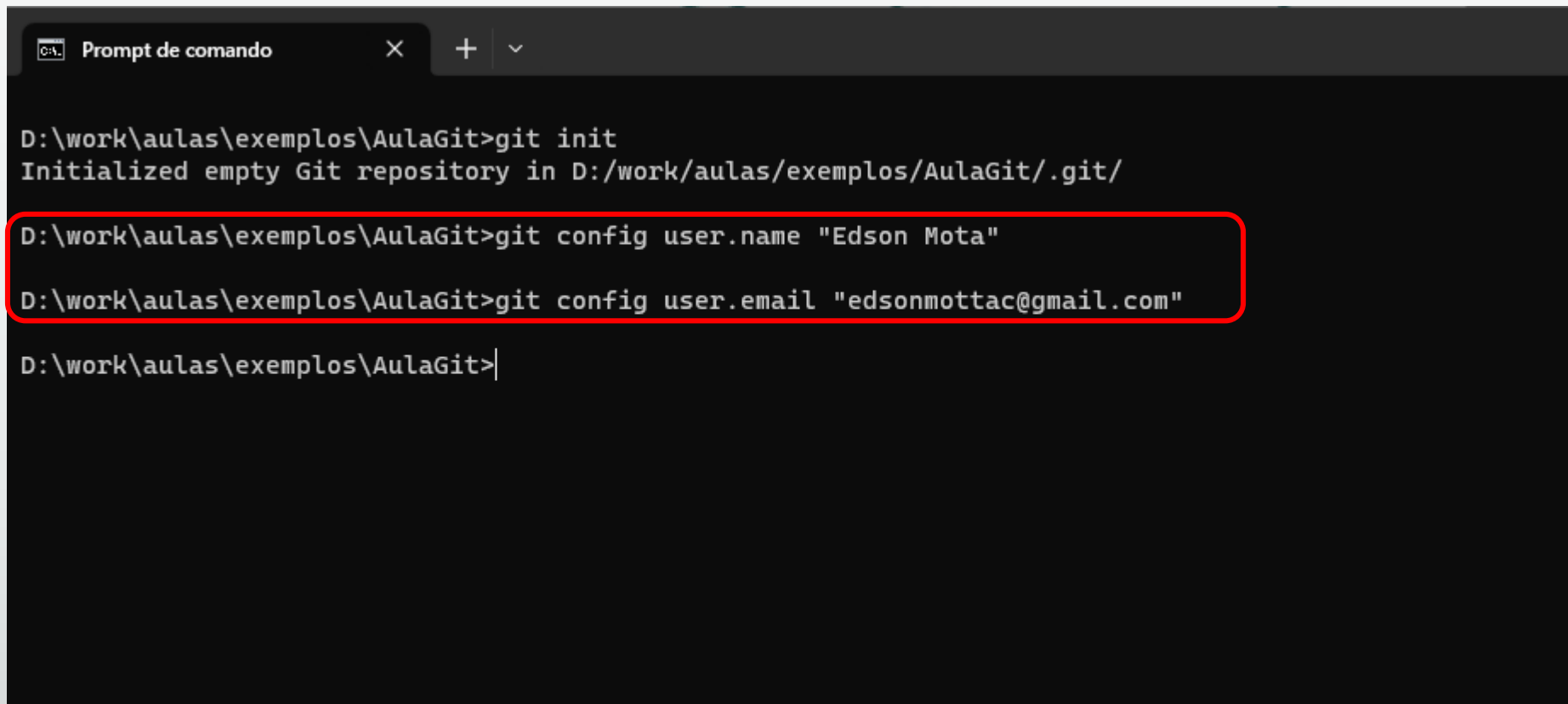
- O comando `git init` inicializa um projeto git.
 - A partir deste comando, o git irá criar uma área de *stage*, na qual as alterações do seu projeto serão gerenciadas localmente.



```
Prompt de comando
D:\work\aulas\exemplos\AulaGit>git init
Initialized empty Git repository in D:/work/aulas/exemplos/AulaGit/.git/
D:\work\aulas\exemplos\AulaGit>
```

Informações do Usuário Git

- Para prosseguirmos, precisamos fornecer algumas informações que serão inseridas nos registros de alterações do GIT, identificando assim, “quem fez o quê e quando”



```
Prompt de comando
D:\work\aulas\exemplos\AulaGit>git init
Initialized empty Git repository in D:/work/aulas/exemplos/AulaGit/.git/
D:\work\aulas\exemplos\AulaGit>git config user.name "Edson Mota"
D:\work\aulas\exemplos\AulaGit>git config user.email "edsonmottac@gmail.com"
D:\work\aulas\exemplos\AulaGit>
```

Adicionando Arquivos (git add .)

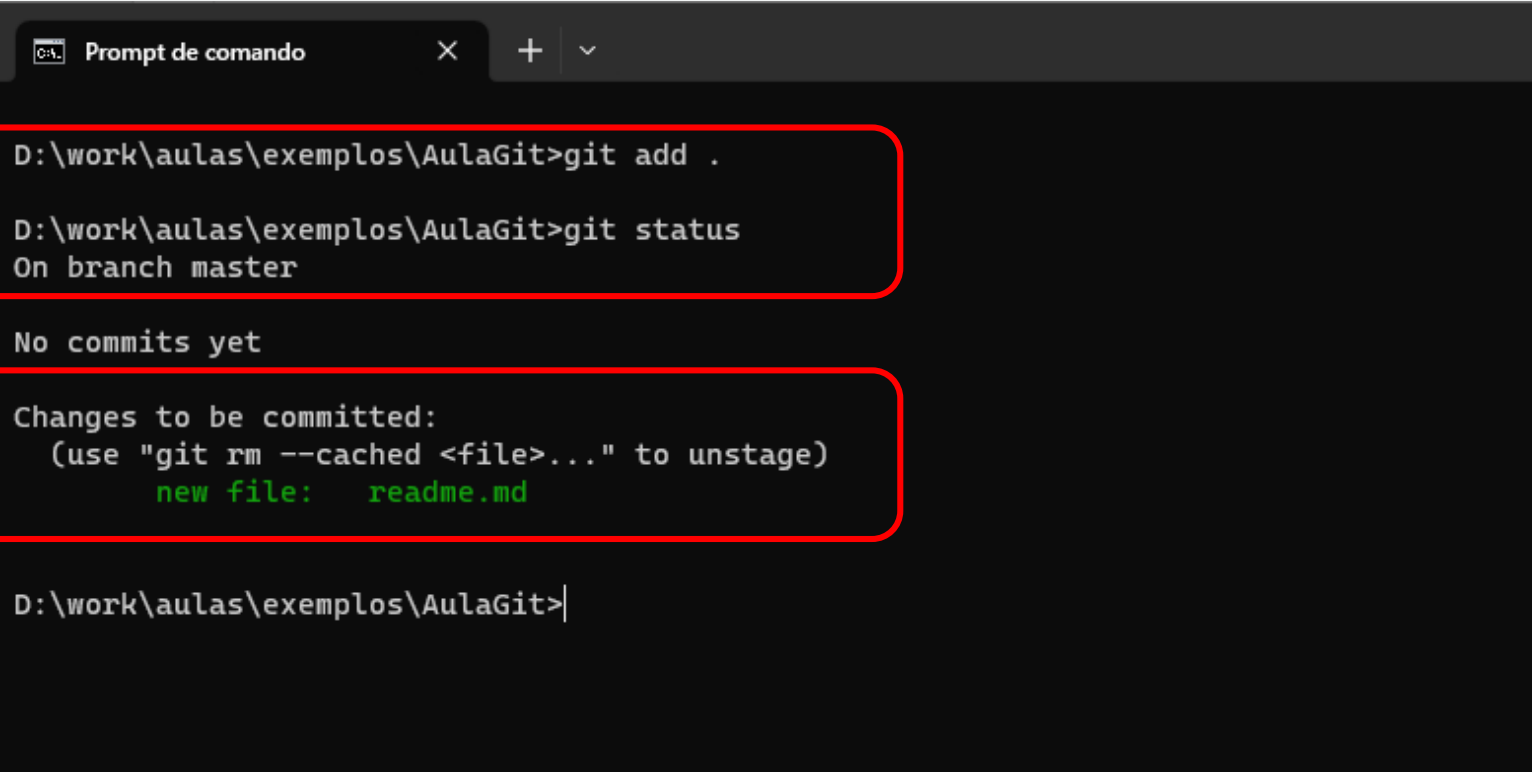
- Para adicionarmos o arquivo readme.md, e assim selecioná-lo para que entre no próximo commit, precisamos utilizar o comando “git add”
- Temos duas formas de utilizar este comando:
 - **git readme.md**
 - Envia apenas o arquivo especificado para a área de stage
 - **git add .**
 - Envia todos os arquivos, seus diretórios e subdiretórios a partir da pasta root

Arquivo Adicionado?

- Após executarmos o comando `git add .` podemos também verificar se o arquivo foi devidamente adicionado, usando o comando `git status`

Comandos

Adições realizadas



```
C:\> Prompt de comando X + v
D:\work\aulas\exemplos\AulaGit>git add .
D:\work\aulas\exemplos\AulaGit>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   readme.md

D:\work\aulas\exemplos\AulaGit>
```

Agora sim, vamos “commitar”

- Consiste nas alterações realizadas em um arquivo ou conjunto de arquivos que são registrados no histórico do repositório local.
- Uma mensagem que descreve o que foi alterado em cada evento commit.
- Um commit deve seguir a seguinte estrutura

```
git commit -m "atualiza processo de autenticação"
```


Commit realizado!

Branch onde o commit foi gravado

Comando
Commit

```
D:\work\aulas\exemplos\AulaGit>git commit -m "atualiza processo de autenticação"
[master (root-commit) c230a3a] atualiza processo de autenticação
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 readme.md

D:\work\aulas\exemplos\AulaGit>
```

Retorno da operação

Conteúdo

Outros Comandos Úteis

- **git log**

- Lista todos os commits já realizados em ordem decrescente

- **git status**

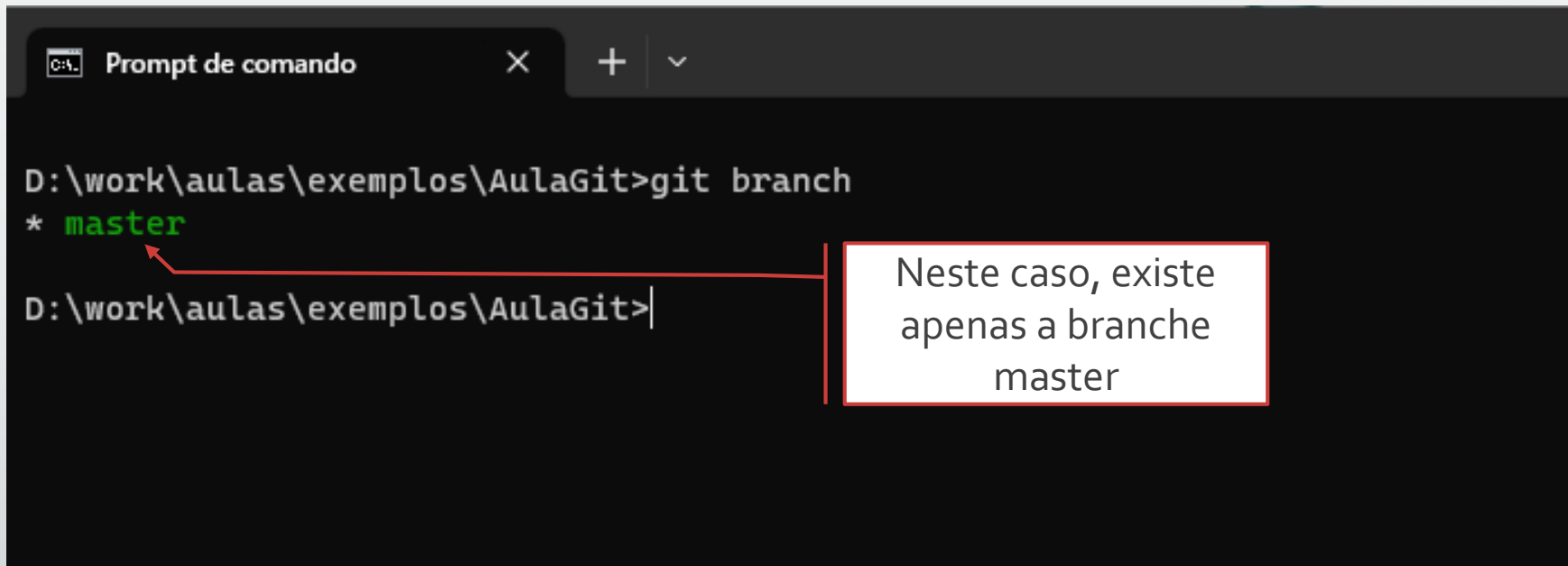
- Arquivos alterados desde o último commit

- **git show**

- Mostra as alterações realizadas no último commit

Branches

- Branche é uma cópia separada do código-fonte em um repositório git que permite que os desenvolvedores trabalhem em paralelo em diferentes partes do projeto.
- Todo novo repositório possui uma branch default chamada master
- Podemos listar todas as branches com o comando **git branch**



The screenshot shows a Windows Command Prompt window titled "Prompt de comando". The command prompt shows the directory `D:\work\aulas\exemplos\AulaGit` and the command `git branch` has been executed. The output is `* master`, where `master` is highlighted in green. A red arrow points from the word `master` to a text box on the right that says "Neste caso, existe apenas a branch master".

```
C:\> Prompt de comando
D:\work\aulas\exemplos\AulaGit>git branch
* master
D:\work\aulas\exemplos\AulaGit>
```

Neste caso, existe apenas a branch master

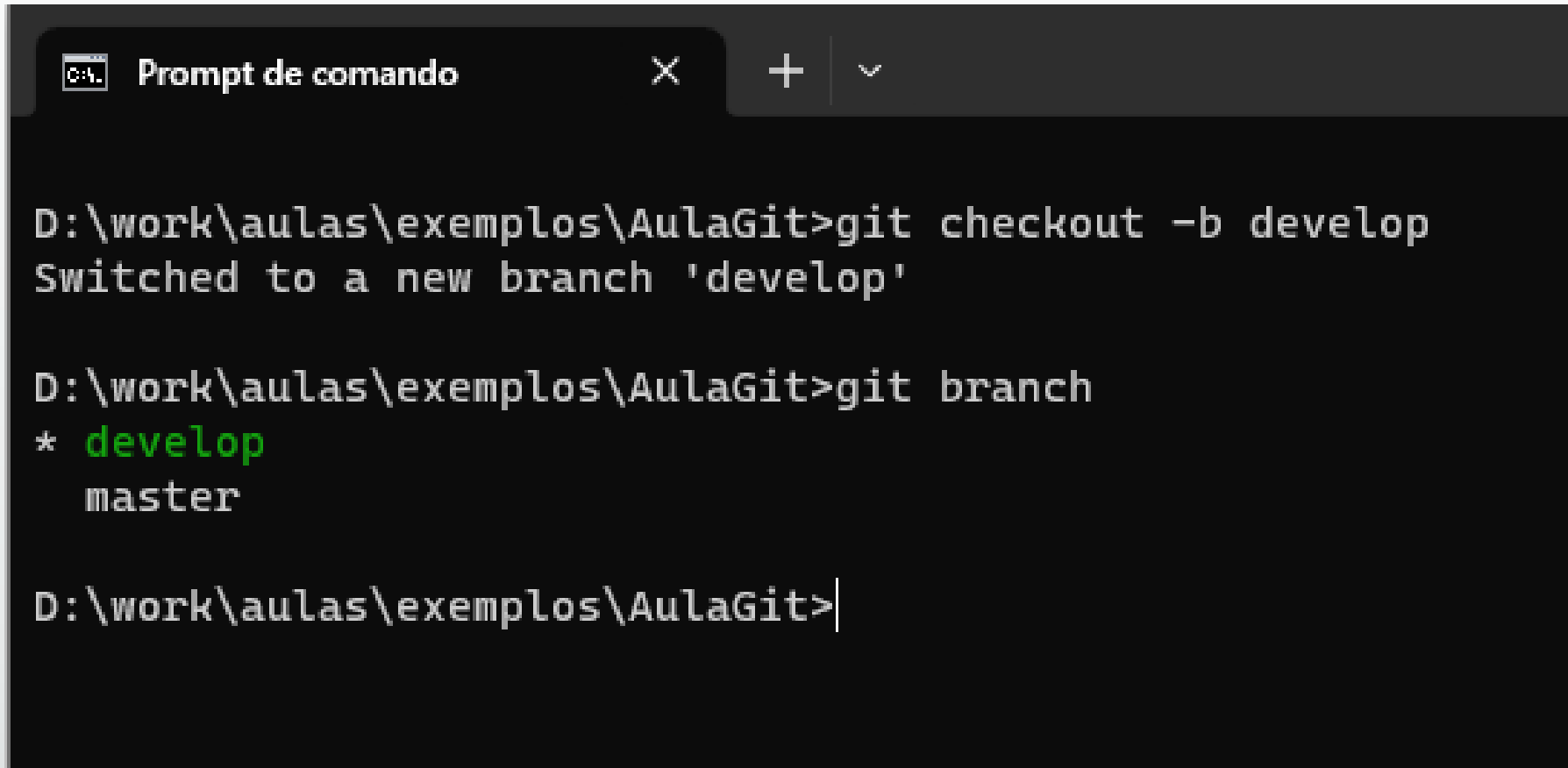
Branches

- Podemos usar o comando **git checkout** para realizar 3 operações encadeadas
 1. Criar uma nova branch
 2. Alocar o conteúdo da atual
 3. Migrar os cursos para esta nova branch criada

```
git checkout -b "develop"
```

- Esse comando fará com o que o git saia da **branch master** (atual) e **migre** para a **develop**

Nova Branch Criada



```
Prompt de comando

D:\work\aulas\exemplos\AulaGit>git checkout -b develop
Switched to a new branch 'develop'

D:\work\aulas\exemplos\AulaGit>git branch
* develop
  master

D:\work\aulas\exemplos\AulaGit>
```

Para retornar a branch anterior, basta executar o comando
`git checkout "nome da branch"`

Merge

- Supondo que fizemos uma alteração no arquivo **readme.md** na branch “*develop*” e agora queremos unificar essas alterações na branch master...
- Podemos fazer isso utilizando o comando `git merge`

Arquivos Modificados?

```
Prompt de comando

D:\work\aulas\exemplos\AulaGit>git checkout -b develop
Switched to a new branch 'develop'

D:\work\aulas\exemplos\AulaGit>git branch
* develop
  master

D:\work\aulas\exemplos\AulaGit>git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")

D:\work\aulas\exemplos\AulaGit>
```

Vamos adicionar, e em seguida realizar o commit dessa alteração utilizando os mesmos comandos:

```
git add .
```

```
git commit -m "alteração1"
```

Comando git status mostra os arquivos modificados na branch develop

Tudo Atualizado

```
Prompt de comando X + v

D:\work\aulas\exemplos\AulaGit>git add .

D:\work\aulas\exemplos\AulaGit>git commit -m "alteração 1"
[develop 9832212] alteração 1
1 file changed, 1 insertion(+)

D:\work\aulas\exemplos\AulaGit>git status
On branch develop
nothing to commit, working tree clean

D:\work\aulas\exemplos\AulaGit>
```


Vamos ao Merge

- Queremos agora replicar essas alteração na branch master, que em geral é tida como a branch principal, onde os códigos de produção normalmente ficam atualizados.
- Precisamos realizar os seguintes passos:

1. Ir para a branch que desejamos atualizar, utilizando o comando

git checkout master

2. Realizar a fusão (merge) das alterações na branch “develop” para a branch “master”

git merge develop

Executando um Merge

Checkout para a branch master

Verificando a branch atual

Operação git merge

```
Prompt de comando
D:\work\aulas\exemplos\AulaGit>git checkout master
Switched to branch 'master'
```

```
D:\work\aulas\exemplos\AulaGit>git branch
develop
* master
```

```
D:\work\aulas\exemplos\AulaGit>git merge develop
Updating c230a3a..9832212
Fast-forward
 readme.md | 1 +
 1 file changed, 1 insertion(+)
```

```
D:\work\aulas\exemplos\AulaGit>
```

Atualizando a Branch Secundária

- Uma outra forma comum de reunir alterações é o git rebase
- Esse comando é usado principalmente quando precisamos trazer as alterações realizadas na branch principal para uma branch secundária
- Em nosso caso: **master → develop**
 - Para isso devemos estar na branch secundária a ser atualizada

git checkout develop

- E executar o comando

git rebase master

Comando [git rebase]

Verifica a branch
atual

```
D:\work\aulas\exemplos\AulaGit>git branch  
develop  
* master
```

Checkout para a
branch develop

```
D:\work\aulas\exemplos\AulaGit>git checkout develop  
Switched to branch 'develop'
```

Verifica se foi
alterado

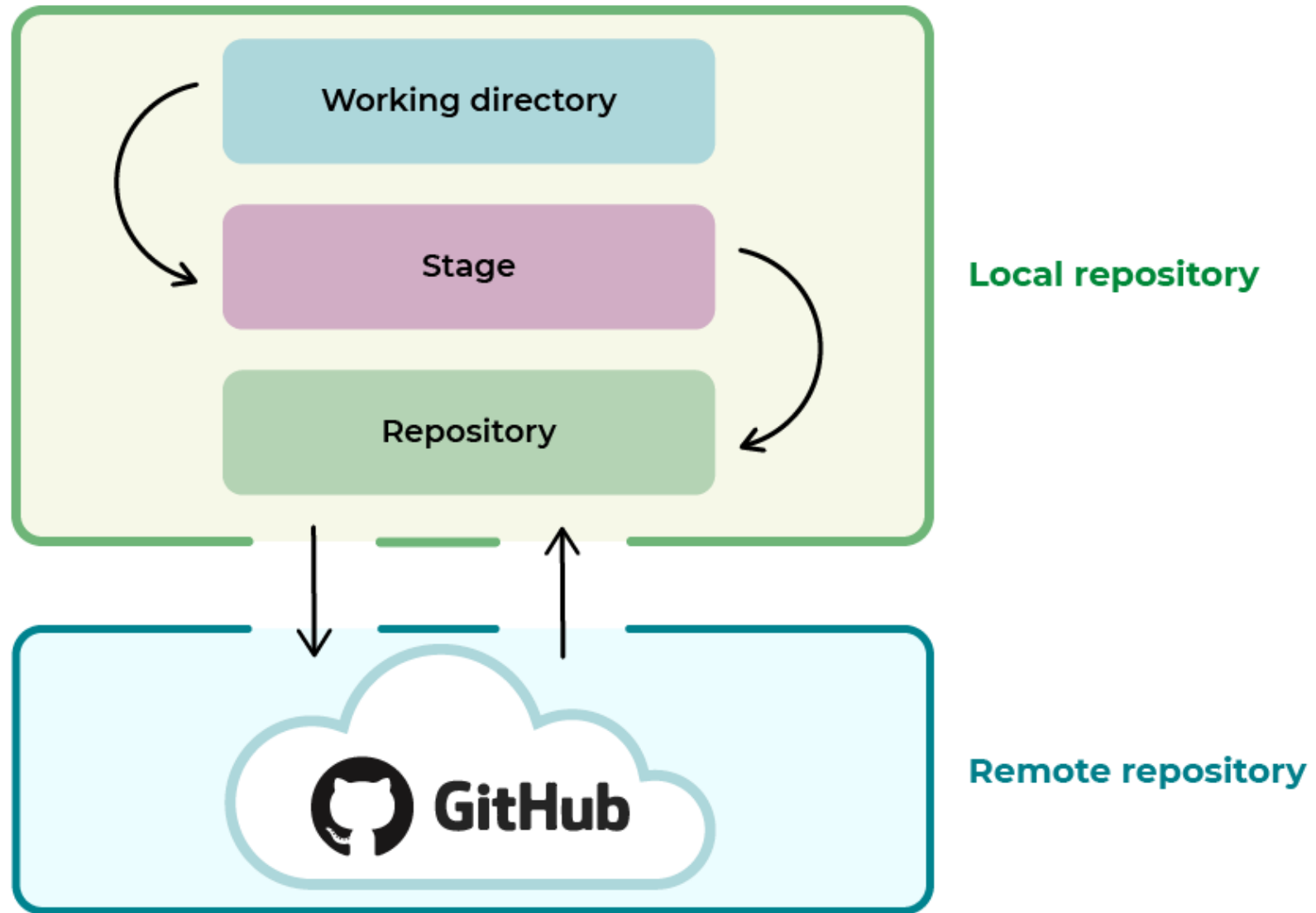
```
D:\work\aulas\exemplos\AulaGit>git branch  
* develop  
master
```

Executa o comando
rebase

```
D:\work\aulas\exemplos\AulaGit>git rebase master  
Successfully rebased and updated refs/heads/develop.
```

“

Até o momento, tudo foi feito
localmente, então, **como subir** os
códigos em um **repositório**?



Repositórios Remotos

- Atuam de forma similar aos repositórios locais
- Requerem comandos de sincronização
- Existem muitas plataformas de gerenciamento Git, algumas delas são:



Repositórios Remotos

- Nesse curso, utilizaremos o GitHub
- A primeira coisa a fazer é criar uma conta em <https://github.com/>
- Em seguida, configurar a chave SSH que permitirá a comunicação com a plataforma
 - Para isso, basta seguir os seguintes passos:
 - <https://docs.github.com/en/get-started/getting-started-with-git/setting-your-username-in-git>



Clonando um Repositório

- Consiste em criar uma cópia de um repositório remoto, trazendo o projeto para o seu ambiente de git local
- Podemos fazer isso usando o comando

```
git clone http://endererço-git-remoto
```

Projeto Clonado

```
Prompt de comando
D:\work\aulas\exemplos>git clone https://github.com/edsonmottac/aula-programacao-2023.1.git
Cloning into 'aula-programacao-2023.1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
D:\work\aulas\exemplos>
```

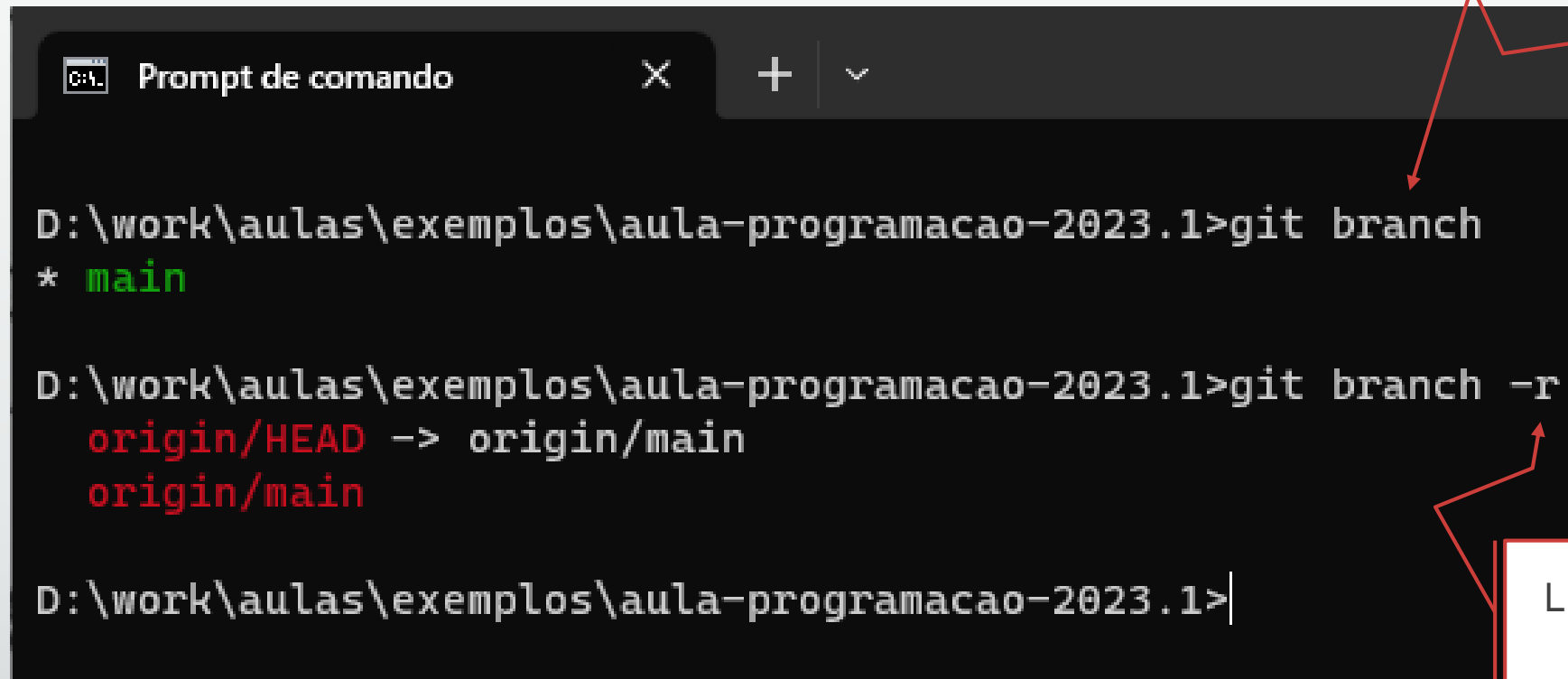
Endereço do projeto no GitHub

Nome da pasta que será criada no seu diretório local

Resultado da cópia

Branch Remota

- Obtendo dados das branches locais e remotas



The screenshot shows a Windows Command Prompt window titled "Prompt de comando". The command prompt is open at the directory `D:\work\aulas\exemplos\aula-programacao-2023.1`. The first command entered is `git branch`, which outputs `* main`. The second command entered is `git branch -r`, which outputs `origin/HEAD -> origin/main` and `origin/main`. The third command entered is `git branch -r`, which outputs nothing.

```
D:\work\aulas\exemplos\aula-programacao-2023.1>git branch
* main

D:\work\aulas\exemplos\aula-programacao-2023.1>git branch -r
origin/HEAD -> origin/main
origin/main

D:\work\aulas\exemplos\aula-programacao-2023.1>
```

Listando branches locais

Listando branches remotas

Criando a Nossa Branch

- Podemos criar uma branch local com base na branch remota de forma similar as demais branches criadas localmente.

```

Prompt de comando
D:\work\aulas\exemplos\aula-programacao-2023.1>git branch
* main

D:\work\aulas\exemplos\aula-programacao-2023.1>git branch -r
origin/HEAD -> origin/main
origin/main

D:\work\aulas\exemplos\aula-programacao-2023.1>git checkout -b ajustes origin/main
Switched to a new branch 'ajustes'
branch 'ajustes' set up to track 'origin/main'.

D:\work\aulas\exemplos\aula-programacao-2023.1>git branch
* ajustes
main

D:\work\aulas\exemplos\aula-programacao-2023.1>|
```

Verifica as branches locais

Verifica as branches remotas

Cria e copia os códigos para uma nova branch local

Verifica se a nova branch foi criada

Enviando para o Repositório

- Supondo que realizamos alterações nessa branch e agora queremos integrá-la a master local e no repositório...
- Podemos usar o comando merge para unificarmos a branch “ajustes” a branch “main” (local)
- Em seguida, o comando **push** para enviarmos a nova versão para o repositório. *Este comando envia de fato a última versão do código repositório remoto.*

```

C:\ Prompt de comando - git pus X + v
D:\work\aulas\exemplos\aula-programacao-2023.1>git merge ajustes
Updating 86eae3d..bbe6c05
Fast-forward
 README.md | 2 + -
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\work\aulas\exemplos\aula-programacao-2023.1>git push -u origin main

```

O comando push
requer permissões
de acesso

Escolha a opção de
autenticação de sua
preferência e siga as
instruções para permitir
a realização do push no
repositório remoto

Connect to GitHub

GitHub
Sign in

Browser/Device Token

Sign in with your browser

Sign in with a code

Don't have an account? [Sign Up](#)

Pronto! Tudo Atualizado!

- Depois disso, o repositório deverá ser atualizado.

```
Prompt de comando

D:\work\aulas\exemplos\aula-programacao-2023.1>git merge ajustes
Updating 86eae3d..bbe6c05
Fast-forward
 readme.md | 2 + -
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\work\aulas\exemplos\aula-programacao-2023.1>git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/edsonmottac/aula-programacao-2023.1.git
 86eae3d..bbe6c05  main -> main
branch 'main' set up to track 'origin/main'.

D:\work\aulas\exemplos\aula-programacao-2023.1>
```

Git + Github

- A infraestrutura GIT unida às plataformas de gerenciamento como github, gitlab, entre outras, proporcionam uma infinidade de recursos e possibilidades de gerenciamento de código e versionamentos.
- Aqui, discutimos alguns conceitos fundamentais.
- A utilização correta da plataforma GIT é um requisito importante para os desenvolvedores de software atuais.

Bons estudos!