

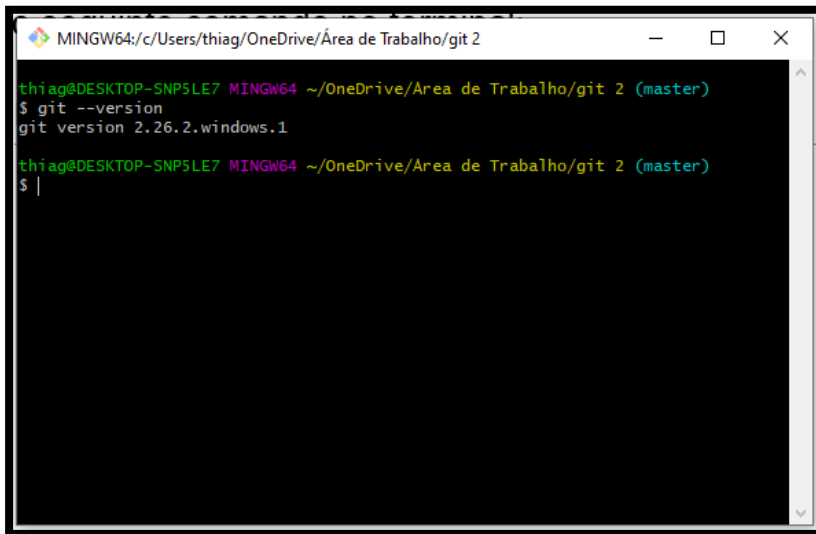
## Práticas e Ferramentas de Desenvolvimento de Software

### Controle de versão II: Prática de controle de versão (git base e GitHub)

# git

Para saber se o git já está instalado no sistema e ao mesmo tempo a versão do git executamos o seguinte comando no terminal:

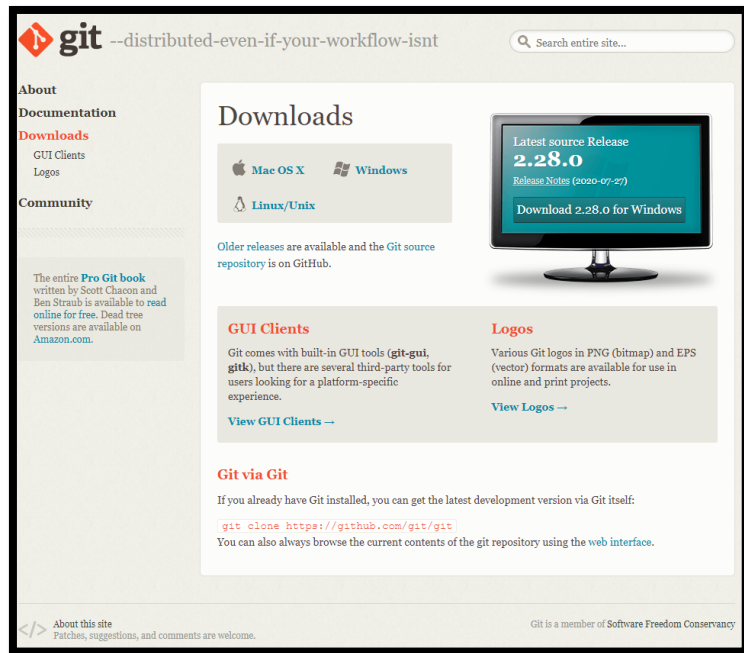
## git --version



```
MINGW64; c:/Users/thiag/OneDrive/Área de Trabalho/git 2
thiag@DESKTOP-SNP5LE7 MINGW64 ~/OneDrive/Área de Trabalho/git 2 (master)
$ git --version
git version 2.26.2.windows.1
thiag@DESKTOP-SNP5LE7 MINGW64 ~/OneDrive/Área de Trabalho/git 2 (master)
$ |
```

# git

Caso o comando anterior não seja reconhecido, será necessário instalar o Git no sistema. o download do Git pode ser feito no seguinte website: <https://git-scm.com/downloads>. Após acessar a página, baixe o Git de acordo com o sistema operacional a ser utilizado.

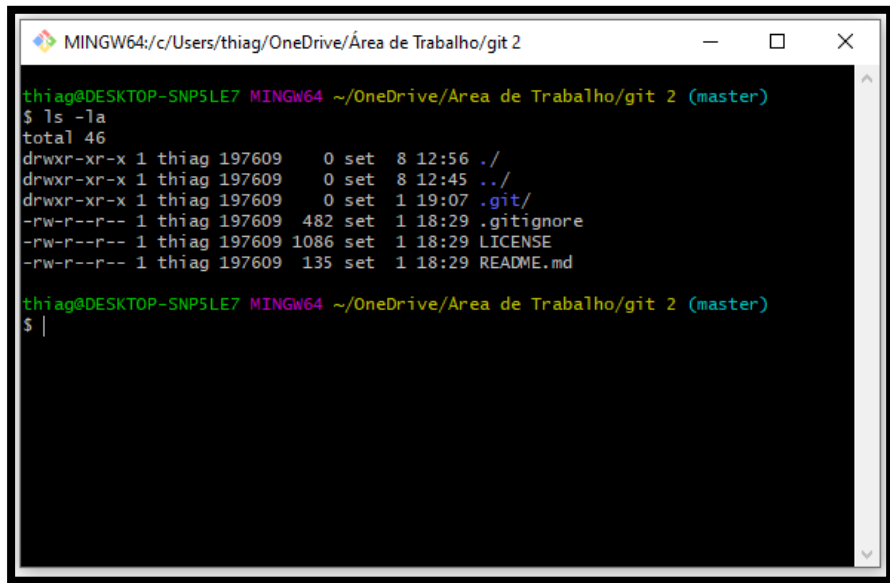


# git

Depois da instalação do Git, podemos navegar até o diretório que queremos fazer controle de versão (geralmente o diretório já possui código fonte ainda não versionado) e criar um novo repositório do Git. Para isso, utilizando o terminal (de preferência o git bash no Windows) e dentro do diretório do novo repositório digitamos o seguinte comando:

## git init

Após a execução do comando anterior será criado um diretório oculto chamado `.git` que armazenará uma estrutura de arquivos e diretórios necessários para o Git conseguir manusear o repositório.



```
MINGW64: c:/Users/thiag/OneDrive/Área de Trabalho/git 2
thiag@DESKTOP-SNP5LE7 MINGW64 ~/OneDrive/Área de Trabalho/git 2 (master)
$ ls -la
total 46
drwxr-xr-x 1 thiag 197609  0 set  8 12:56 ./
drwxr-xr-x 1 thiag 197609  0 set  8 12:45 ../
drwxr-xr-x 1 thiag 197609  0 set  1 19:07 .git/
-rw-r--r-- 1 thiag 197609 482 set  1 18:29 .gitignore
-rw-r--r-- 1 thiag 197609 1086 set  1 18:29 LICENSE
-rw-r--r-- 1 thiag 197609 135 set  1 18:29 README.md

thiag@DESKTOP-SNP5LE7 MINGW64 ~/OneDrive/Área de Trabalho/git 2 (master)
$ |
```

# git

Para relacionar o repositório atual com um repositório remoto hospedado em um servidor git (i.e Github) é necessário primeiro criar o repositório remoto e depois copiar o link para esse repositório. Por exemplo, para o repositório que criamos durante as aulas o link para o repositório remoto seria:

<https://github.com/tadsifsp/TADS-PFDA1.git>

De posse do link do repositório,  
digitamos o seguinte comando  
no terminal:

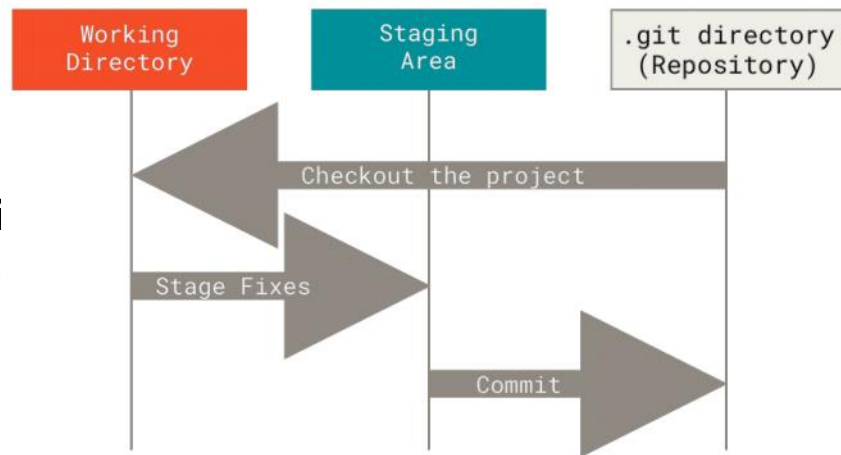
**git remote add origin <https://github.com/tadsifsp/TADS-PFDA1.git>**

# Git - PULL

A partir desse momento nosso repositório local criado pelo comando **git init** está relacionado com o repositório remoto hospedado no **GitHub**. Como o nosso repositório remoto já tem alguns arquivos criados (README, LICENSE, .gitignore), o próximo passo é atualizar o repositório local com o conteúdo disponível no repositório remoto. Para isso executamos o seguinte comando no terminal:

**git pull origin master**

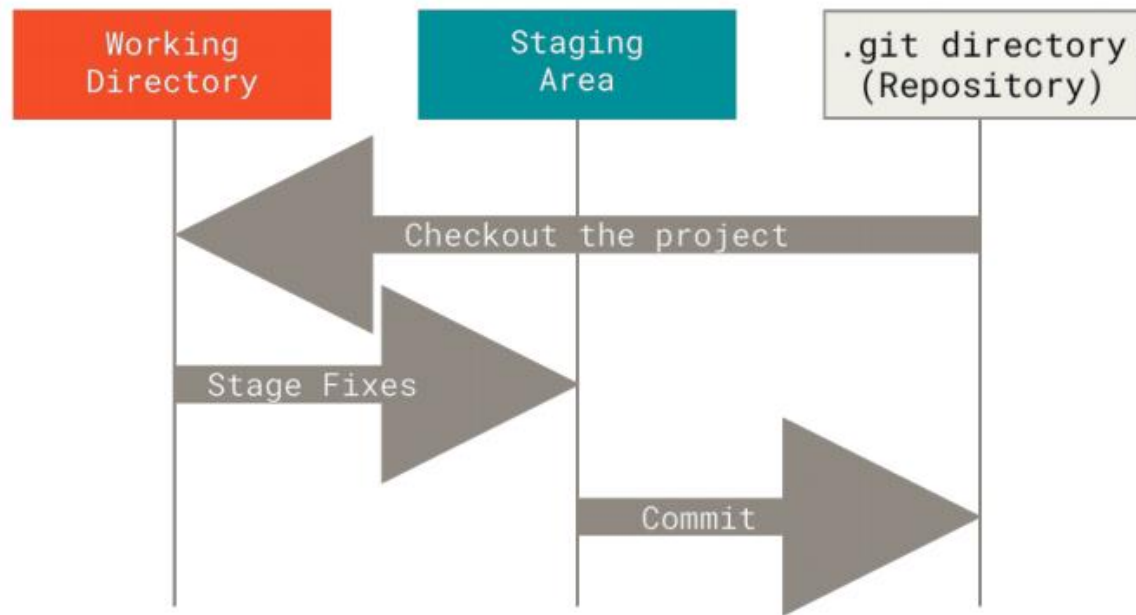
Após a execução desse comando, todo o conteúdo do repositório remoto será baixado e estará disponível no diretório atual (**working directory**) junto com os demais arquivos que já estavam.



# GIT – relembrando os três estados

O Git tem três estados principais em que os arquivos de um repositório podem residir: **modified** (modificado), **staged** (preparado) e **committed** (confirmado).

- **Modified** (modificado) significa que você alterou o arquivo, mas ainda não o enviou ao banco de dados.
- **Staged** (Preparado) significa que você marcou um arquivo modificado em sua versão atual para ir para o próximo **commit**.
- **Committed** (Confirmado) significa que os dados estão armazenados com segurança em seu banco de dados local.



# Git - Status

Após a execução desses procedimentos, precisamos saber quais arquivos presentes no diretório atual ainda não estão no repositório remoto (isto é, estão no estado modificado). Para saber quais arquivos foram criados, modificados ou excluídos, executamos o seguinte comando:

## **git status**

Após a execução desse comando no terminal, uma lista de arquivos ainda não versionados (que não estão no repositório remoto) vai surgir com cada item na cor vermelha.

Devemos agora decidir quais arquivos vamos querer enviar para nossa área de preparo (staging área).



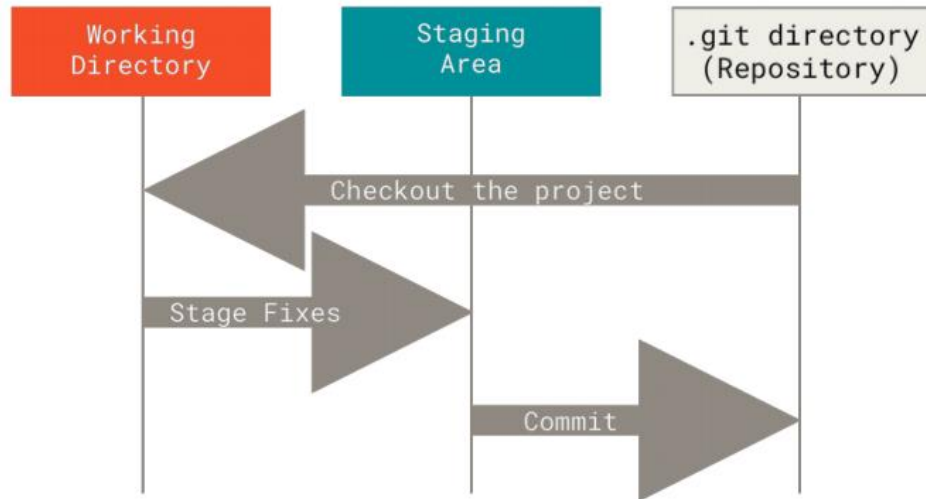
# Git - ADD

A partir da lista de arquivos ainda não versionados, podemos adicionar um a um com o seguinte comando:

**git add <<nome do arquivo / diretório>>**

Onde <<nome do arquivo / diretório>> é o nome do arquivo ou diretório que será posteriormente versionado. O uso do comando **git add <<nome do arquivo / diretório>>** deve ser repetido até todos os arquivos que se deseja versionar aparecerem em verde ao executar o comando **git status**.

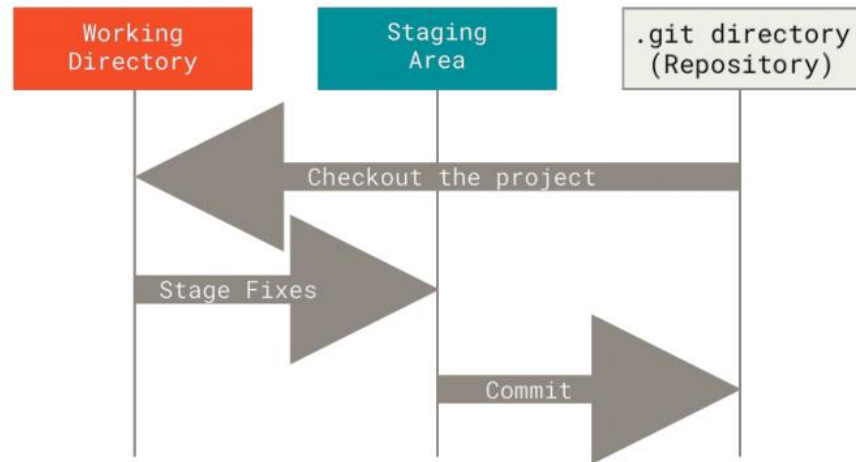
**git add .** (Adiciona todos os arquivos modificados, removidos ou adicionados)



# Git - COMMIT

Depois que os arquivos e diretórios foram adicionados na área de preparado (staging área) com o comando **git add**, é necessário criar um **commit** com as alterações realizadas. Um **commit** é como um fotografia atual do todo o repositório com as alterações que foram introduzidas e adicionadas pelo comando **git add**. Para realizar um **commit** executamos o seguinte comando:

**git commit -m <<mensagem sobre o commit>>**



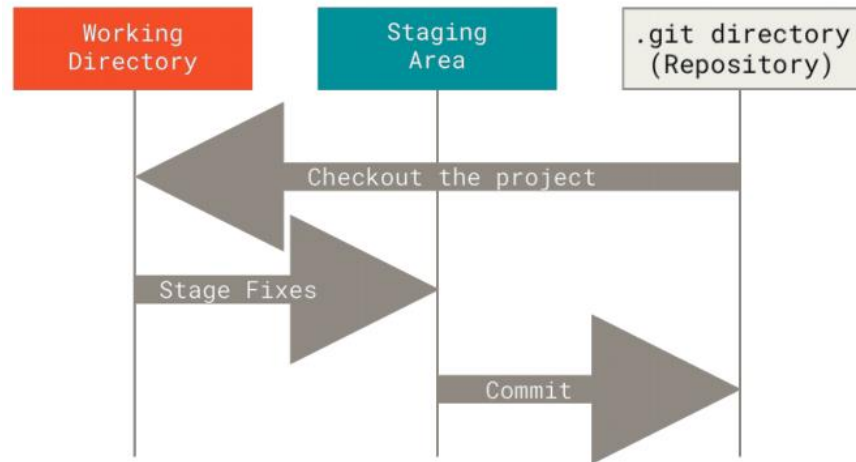
# Git - PUSH

Depois de feito a confirmação das alterações com o comando **git commit**, podemos fazer o upload das atualizações para o repositório remoto. Como é o primeiro **push** que estamos fazendo a partir do repositório local, precisamos executar o seguinte comando:

**git push --set-upstream origin master**

Depois de executado o comando anterior, será solicitado as credenciais do usuário do GitHub, preencha com as suas credenciais.

Em seguida, todas as alterações realizadas e registrados no **commit**, serão enviadas para o repositório remoto, nesse momento uma nova versão do código será criada.



# GIT – fluxo de trabalho

O fluxo de trabalho básico do Git funciona da seguinte forma:

- Você modifica arquivos na sua árvore de trabalho;
- Você seleciona as mudanças que deseja que façam parte de seu próximo **commit** e encaminha elas para a área de preparo.
- Você faz um **commit** (confirmação), que encaminhará os arquivos conforme eles estão na área de preparo e armazena eles permanentemente em seu diretório Git (repositório).

