# NAVEGAÇÃO VIA CLI E INSTALAÇÃO

# ENTENDENDO COMO O GIT FUNCIONA

# PRIMEIROS COMANDOS COM O GIT

## INICIANDO O GIT E CRIANDO UM COMMIT

* **git init:** cria um repositório no Git dentro do diretório desejado
* **git config --global user.email “***inserir endereço de e-mail***”:** seta configurações de e-mail de forma global (e-mail do GitHub)
* **git config --global user.name “***inserir username***”:** seta configurações de username de forma global (username do GitHub)
* **git add \*:** adiciona todos os arquivos **untracked** e **modified** da pasta para o estado **staged** para posterior commit para o Git e envio (**push**) para o GitHub.
* **git commit -m “***inserir mensagem identificando o commit***”:** faz o commit para o repositório local e acrescenta uma mensagem identificadora (usando a flag **-m**)

# Ciclo de vida dos arquivos no Git

* tipos de arquivo: untracked e tracked.
* **untracked:** arquivos os quais o Git ainda não tem ciência deles.
* tracked:
* arquivos os quais o Git já tem ciência dele.
* se dividem em 3: unmodified, modified e staged.
* **unmodified:** arquivo que já sofreu **commit**.
* **modified:** arquivo já criado que sofreu alguma modificação.
* **staged:** estado para o qual os arquivos **untracked** e **modified** vão, antes de sofrerem um **commit**.
* ambientes: servidor e ambiente de desenvolvimento
* ambiente de desenvolvimento (localmente): working directory, staging area e local repository.
* **working directory:** repositório de trabalho, onde ficam os arquivos que estão sendo manipulados.
* **staging area:** para onde vão os arquivos quando estão pronto para serem enviados para o repositório local (vão para essa área após o uso do comando **add**)
* **local repository:** repositório local, onde ficam guardados os arquivos, após passarem pela **staging area** e receberem o commit (são enviados para o repositório local após o uso do comando **commit**)
* servidor (remotamente): **remote repository**
* **git status:** mostra os status dos arquivos (**untracked, modified, staged)**, além de outras informações relevantes.
* **mv** *nome do arquivo ./nome do repositório***:** move o arquivo para o repositório.

# Introdução ao GitHub

* ideal (mas não obrigatório) que username e e-mail configurados no Git sejam os mesmo utilizados na criação da conta no GitHub.
* **git config --list:** mostra a lista de todas as configurações feitas no Git.
* **git config -- global --unset** *propriedade da configuração***:** desfaz a configuração especificada no comando.
* criar um repositório remoto no GitHub como o mesmo nome do repositório local do Git.
* **git remote add origin** *link do repositório remoto do GitHub:* adiciona a “origem” remota para onde serão enviados os arquivos do nosso repositório local.
* a palavra **origin** é apenas um *alias* para que não seja necessário colocar o link toda vez que um **push** for feito.
* **git remote -v:** lista os repositórios remotos que estão cadastrados no Git.
* **git push origin master:** envia o conteúdo do repositório local para o repositório remoto.
* a palavra **master** diz respeito ao nome da **branch** onde estão sendo enviados os códigos.

# Resolvendo conflitos

* **git pull origin master:** puxa a versão que está no GitHub para a máquina local.
* para clonar um repositório, basta ir no repositório desejado, clicar no botão verde **Code**, e copiar o caminho do mesmo (**url**).
* **git clone** *link do repositório do GitHub*: clona um repositório específico do GitHub para a máquina local.