Git & GitHub

Em 2005 Linus Torvalds estava desenvolvendo o sistema Linux que foi feito de uma forma totalmente colaborativa com pessoas distribuídas pelo mundo inteiro. Portanto havia a necessidade de um sistema de versionamento de códigos que permitisse que mais de uma pessoa trabalhasse editando o mesmo arquivo ou até mesmo editando a mesma linha. Mas naquela época o sistema existente não era capas de suportar essa tarefa. Desconte com o esse sistema (CVS) Linus decidiu criar seu próprio sistema de versionamento de códigos o Git, que ajuda a criar e monitorar diferentes versões do código.

Observação

Software não é um trabalho de um homem só e sim um trabalho para ser feito de forma colaborativa.

**Git** = sistema de versionamento de código distribuído.

**GitHub** = nuvem onde os códigos são armazenados

* Empresa da Microsoft.
* Funciona como um tipo de rede social.

O Git e o GitHub são duas tecnologias diferentes que se complementam. As duas tecnologias são as que estão em alta no mercado atualmente, mas existem outras tecnologias com o mesmo intuito.

**Benefícios de aprender Git & GitHub**

1. Controle de versão
2. Armazenamento em nuvem
3. Trabalho em equipe
4. Melhorar seu código
5. Reconhecimento

Objetivos da aula

1. Mudar de pastas
2. Listar pastas
3. Criar pastas/arquivos
4. Deletar pastas/arquivos

GUI X CLI

CLI (Comand Line Interface) → interação por linha de comando

|  |  |
| --- | --- |
| Windows | Unix |
| Cd | Cd |
| Dir | Is |
| Mkdir | Mkdir |
| Del/rmdir | Rm-rf |

GUI (Grafic user interface) → Interação por interface gráfica

Comandos de navegação

* Dir/ls → utilizado para listar pastas/diretórios; serve para podermos nos situar dentro do sistema operacional e enteder em qual local estamos.
* Cd → utilizado para entrar em uma pasta específica
* Cd .. → sair da pasta
* Cls/clear → utilizado para limpa o terminal; limpa a tela.
* Mkdir → utilizado para criar uma pasta/diretório.
* Echo → printa de volta no terminal uma frase ou texto escrito
* > → utilizado para redirecionar o fluxo; vai criar um arquivo
* Del (windows) → deleta arquivos da pasta
* Rmdir/rm → deleta a pasta/diretório.

Observação

Para apagar uma pasta de forma simplificada devemos escrever:

“rmdir *nome da pasta* /s /q “ para Windows

“rm – rf *nome da pasta*” para Linux

Exemplo:

Echo hello > hello.txt

Atalhos:

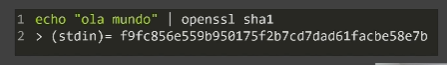
* Linus
  + CTRL + L - Clear
* Windows
  + TAB - Auto completar
  + ↑ - navegar entre o histórico de comando executado no terminal

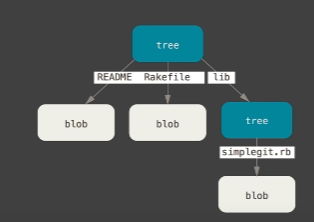
Todos os comandos possuem variâncias, possuem flags (complementos passados para o comando), que acrescentam, modificam ou formatam a forma que esses comandos são devolvidos.

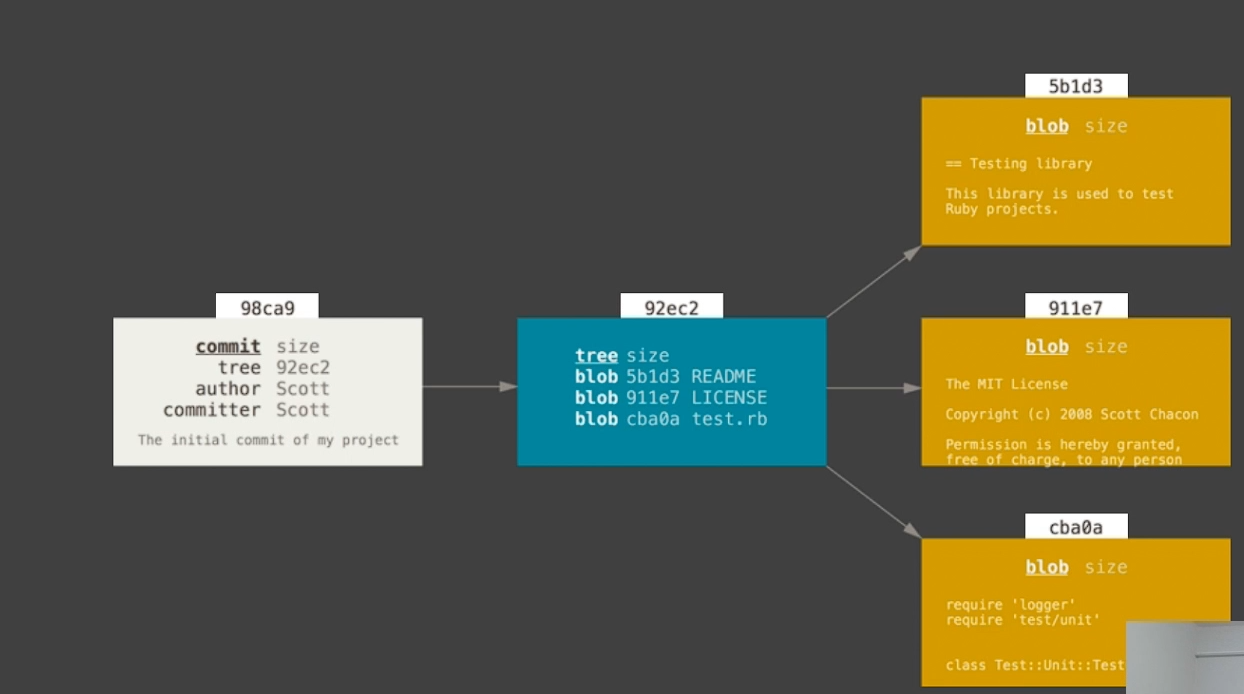
SHA1

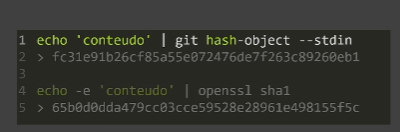
É um algoritmo de encriptação que pega o arquivo e embaralha ele de uma forma específica, ou seja, ele cria uma forma curta de representar um arquivo. A sigla SHA significa Secure Hash Algorithm (Algoritmo de hash Seguro), é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA). A encripitação gera um conjunto de characteres identificador de 40 digítos único.

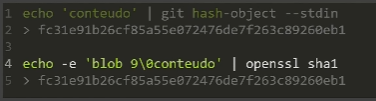
Exemplo:



Objetos internos do HIT



* Blobs – é onde ficam guardados os objeto/diretório; é o bloco básico de composição.
  + Não guarda o nome do arquivo
  + Possuem uma encripitação (SHA1)
  + Exemplo:



* Trees – armazenam e aponta para os blobs; resposável por montar a estrutura de onde está os arquivos.
  + Pode apontar para blobls ou trees
  + Possuem uma encripitação (SHA1)
* Commits – é aquele que vai dar sentido para a alteração que está sendo feita, que vai juuntar tudo;
  + Tem um carimbo de tempo (data e hora de quando foi criado)
  + Possuem uma encripitação (SHA1)
  + O SHA1 da commit é o hash de toda essa informação



Observação

Se você alterar um dado dentro de uma blob ele vai gerar um SHA1, ao alterar os dados da blob o SHA1 da tree também será alterado o que consequentemente também irá alterar no SHA1

Sistema distribuído e seguro

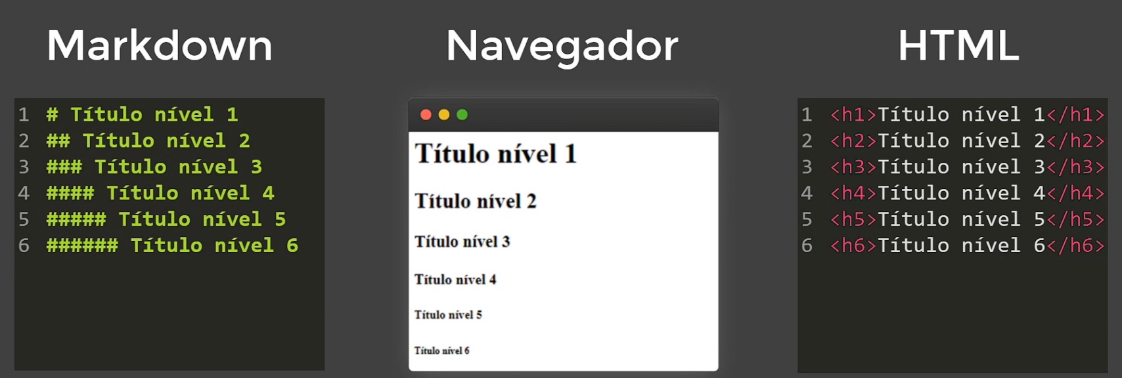
O GIT é um sitema distribuido por ser um servidor que além de guardar em sua propria nuvem os dados ele também gera um arquivo seguro para o seu ursuário.

Objetivos da aula

1. Iniciar o GIT
2. Iniciar o versionamento de código
3. Criação de commit

Comandos

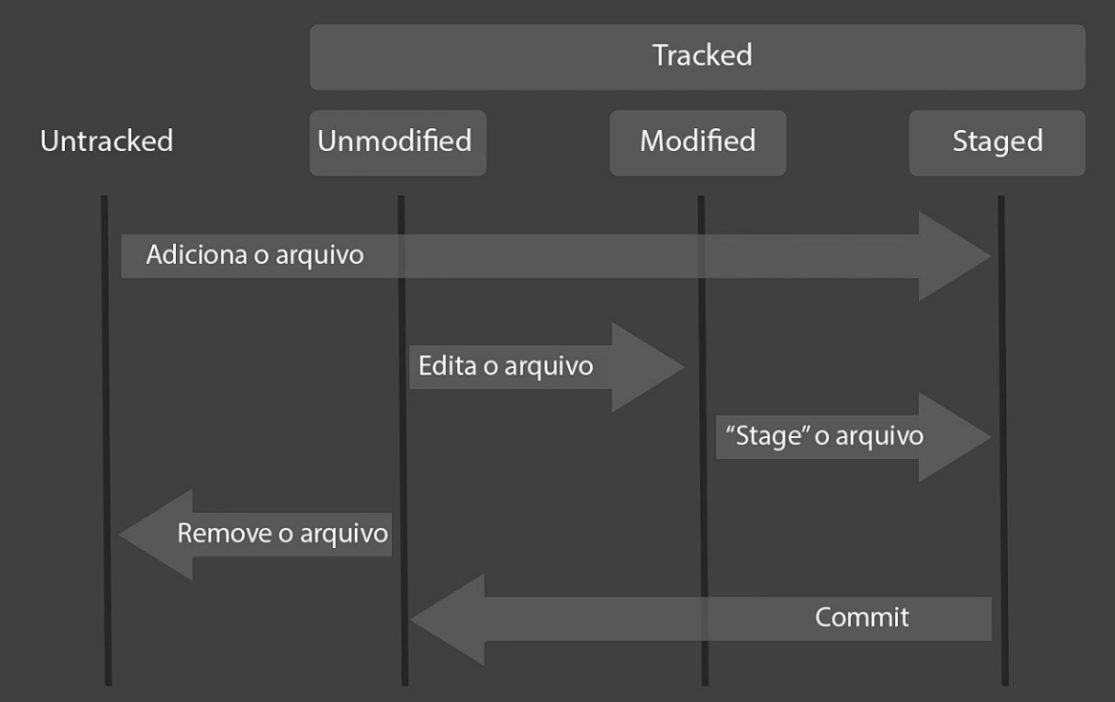
* git init → inciar o repositório do GIT; além de criar a pasta .git ele inicia o repositório do GIT;
* git add → mover arquivos e dar início ao versionamento
* git commit → criação do primeiro commit
* Criando um repositório
* Ls -a → mostrar a pasta oculta, gerecial, do GIT

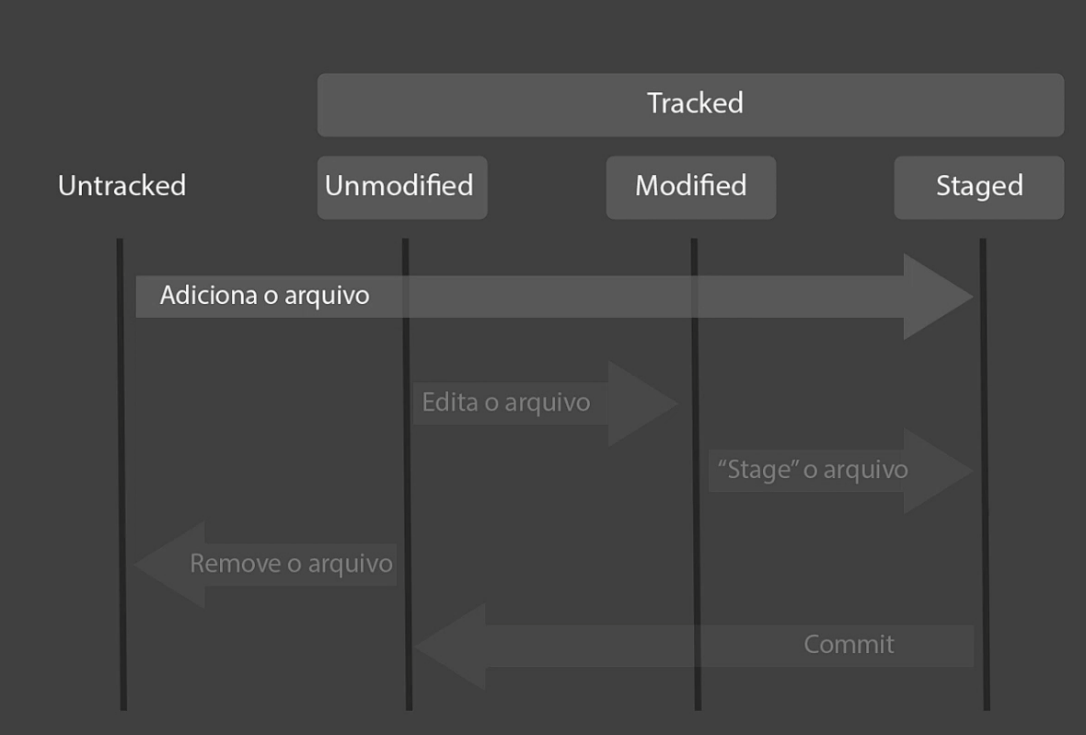
Adicionando um arquivo

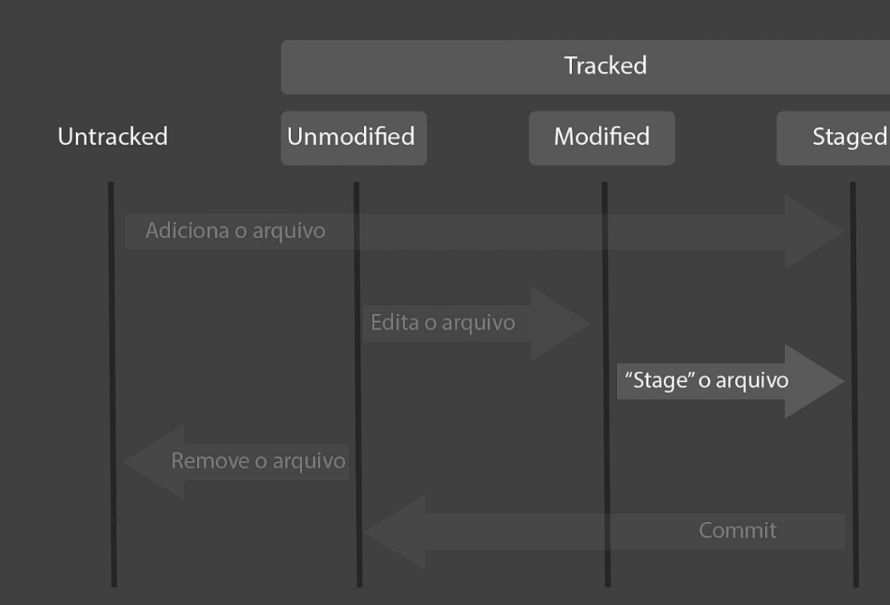
* Markdow – forma mais “humana” de se escrever um arquivo HTML
* HTML – é o esqueleto, a estrutura básica, de qualquer página na web

Git add or git add *nome do arquivo* or git add.- colocando o arquivo no staged

Git commit -m “*texto*” → Código para a mensagem que vai ser atrelada ao doc;

Ciclo de vida dos arquivos

* Tracked – arquivos rastreados pelo GIT; arquivos que temos ciencia de sua existência
  + Unmodified → arquivo que ainda não foi modificado
  + Modified → arquivo que já sofreu modificação
  + Staged → local onde fica os arquivos que estão “se preparando” para poder fazer parte de outro tipo de agrupamento (commit)
* Untracked → arquivos de o GIT ainda não tem ciência

Ao fazer o comando “git add \*” em um arquivo modified ele automaticamente muda para Staged.

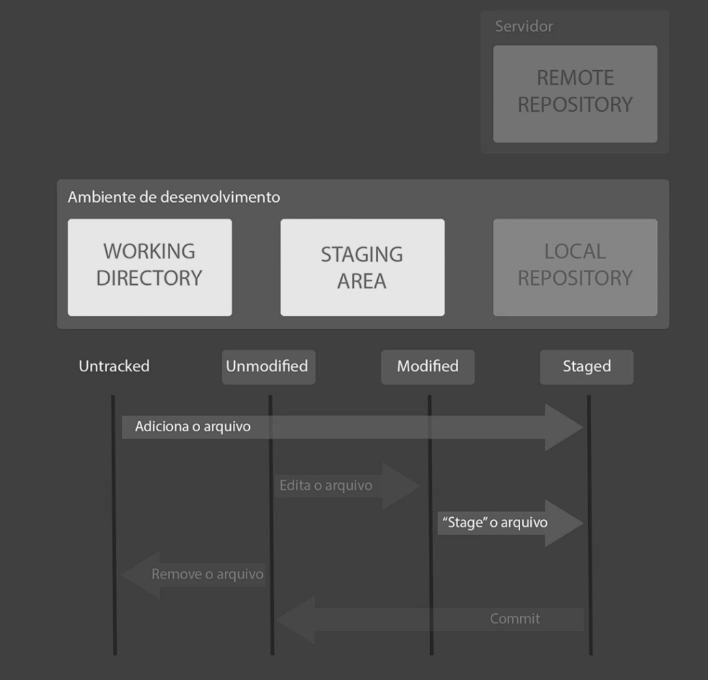
O que são repositórios

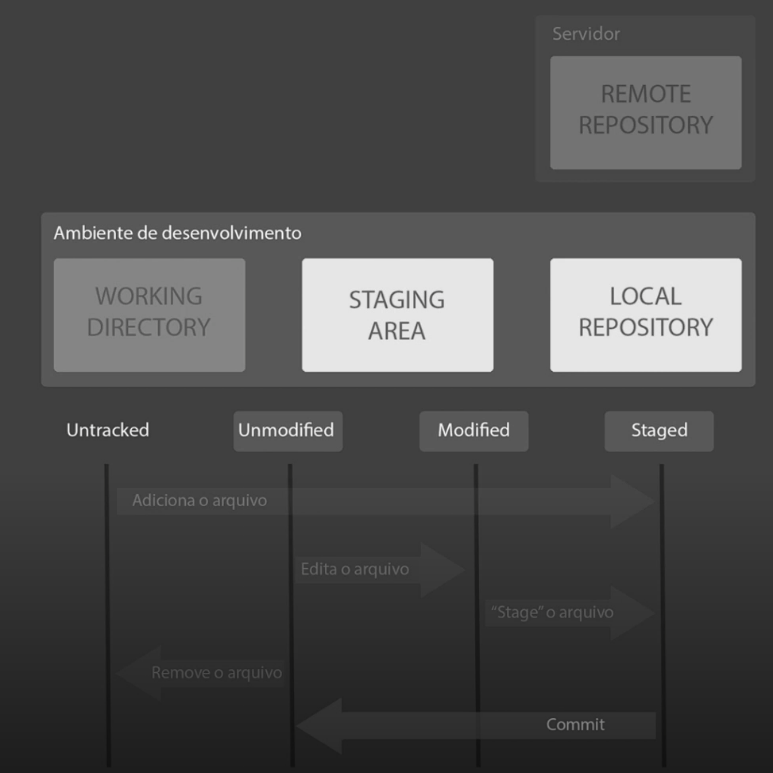
As alterações feitas no código na máquina nao repercurte imediatamente na versão que está no repositório remoto. Só se executar um conjunto de códigos específicos essa alteração vai ser “lançada” no repositório remoto.

Os arquivos vão estar sempre alterando em a área de trabalho e a área de staging.

Quando fazemos um commit ele passa a fazer parte do reposiório local e do repositório local pode ser “lançado” no repositório remoto.

Remote Repository (Repositorio Remoto) – GitHub

Quando adicionamos o arquivo Untracked e damos um “git add\*” ele é movido para área de staging.

Quando temos um arquivo Modified e damos um “git add\*” ele também é movido para área de staging.

Quando fazemos o commit movemos os arquivos para área de staging e ele volta a ser Unmodified

Todo arquivo que vai ser enviado para o respositório remoto tem que estar com commit se não ele não pode ser “lançado” no repositório remoto.

Git status – vai a monitorar o status do arquivo (Untracked, modified,....)

* Resetar name: git config --global --unset user.name
* Resetar e-mail: git config --global --unset user.email
* Inset name: git config --global user.name
* Inser email: git config --global user.email
* Listar as configuração do git: git config –list
* Enviar um arquivo do repositório local para o remoto: git remote add origin *colocar o link do github*
* Verificar as pedências no repositório: Git status
* Listar repositórios remotos: git remote -v
* Enviar o arquivo do repósitorio remoto: git push *origin master*
* Juntar o arquivo do repositório remoto com o local: git pull origin master
* Clonar um arquivo ou pasta do git hub para o repositório local: git clone *url github*