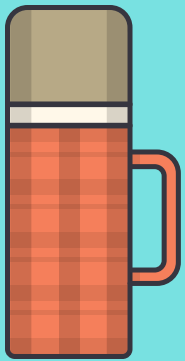


CONTROLE DE VERSÃO

TRABALHANDO COM GIT



By Yuri Pereira

PRAZER, EU SOU O YURI!

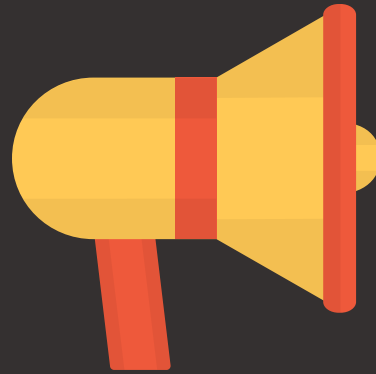


- ESTUDANTE DE ENGENHARIA DA COMPUTAÇÃO, DESDE 2015
- AMO CRIAR SOLUÇÃO PARA OS PROBLEMAS DAS PESSOAS
- CONHEÇO UM POUCO DE JAVA, .NET, SWIFT, JS E PYTHON
- UM ESTUDANTE PARA TODA A VIDA

- TELEGRAM
@yuri_souza

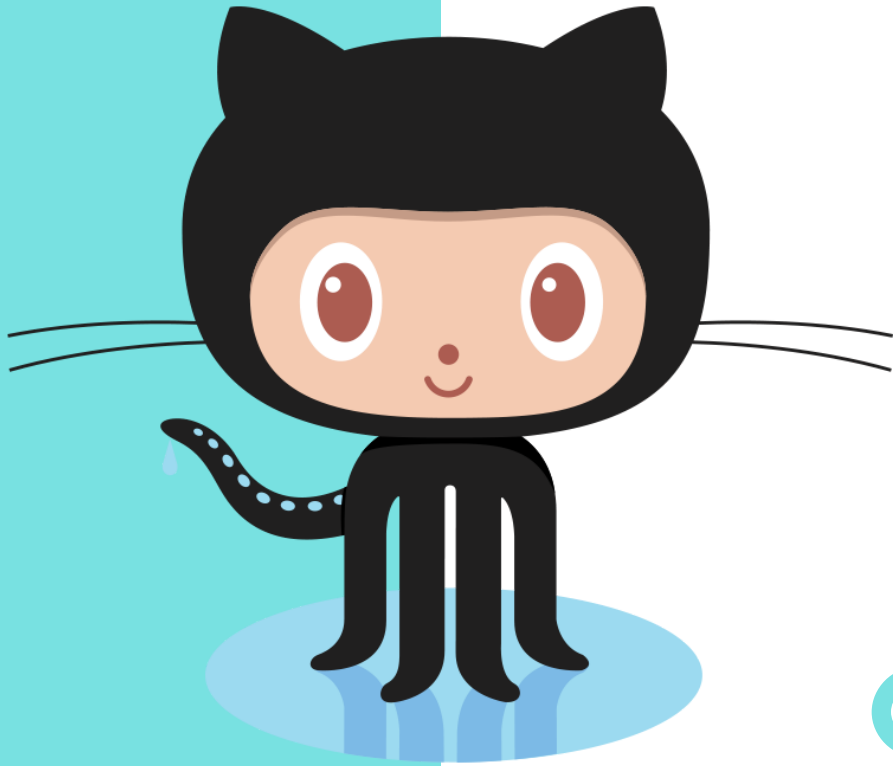
O QUE É GIT?






O GIT É UM SISTEMA DE CONTROLE DE VERSÃO QUE TEM NA SUA ESTRUTURA INTERNA UMA MÁQUINA DO TEMPO EXTREMAMENTE RÁPIDA E UM ROBÔ DE INTEGRAÇÃO BEM EFICIENTE.

FOI CRIADO EM 2005 POR LINUS TORVALDS, O MESMO CRIADOR DO LINUX, QUE ESTAVA DESCONTENTE COMO BITKEEPER.



AFINAL,
O QUE É GITHUB?

www.github.com.



VISÃO GERAL DE GIT EM 15 MINUTOS



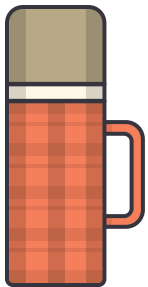
acesse:

<http://try.github.io/>

1.1 TEM 15 MINUTOS E QUER APRENDER O GIT?

O Git permite que grupos de pessoas trabalhem nos mesmos documentos (muitas vezes código) ao mesmo tempo, e sem pisar uns nos outros. É um sistema de controle de versão distribuído.

Nosso prompt de terminal abaixo está atualmente em um diretório que decidimos nomear "octobox". Para inicializar um repositório Git aqui, digite o seguinte comando:



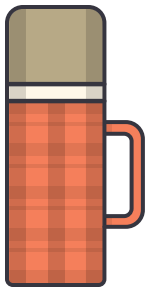
COMANDO:

GIT INIT

1.2 VERIFICANDO STATUS

Bom trabalho! Como Git acabou de nos dizer, nosso diretório "octobox" agora tem um repositório vazio em `/.git/`. O repositório é um diretório oculto onde o Git opera.

Em seguida, vamos digitar o comando `git status` para ver qual é o estado atual de nosso projeto:



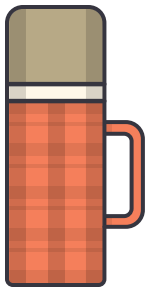
COMANDO:

GIT STATUS

1.3 ADICIONANDO E COMITANDO

Criei um arquivo chamado `octocat.txt` no repositório octobox para você (como você pode ver no navegador abaixo).

Você deve executar o comando `git status` novamente para ver como o status do repositório foi alterado:



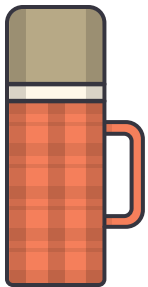
COMANDO:

GIT STATUS

1.4 ADICIONANDO MUDANÇAS

Bom, parece que o nosso repositório Git está funcionando corretamente. Observe como Git diz que `octocat.txt` é "não rastreado"? Isso significa que Git vê que `octocat.txt` é um novo arquivo.

Para dizer ao Git para iniciar o rastreamento de alterações feitas no `octocat.txt`, primeiro precisamos adicioná-lo à área de rastreo usando:

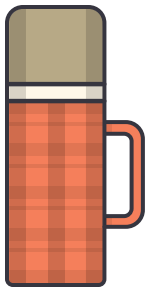


COMANDO:

GIT ADD OCTOCAT.TXT

1.5 ADICIONANDO MUDANÇAS

Bom trabalho! O Git está agora acompanhando nosso arquivo `octocat.txt`. Vamos rodar o comando `git status` novamente para ver onde estamos:



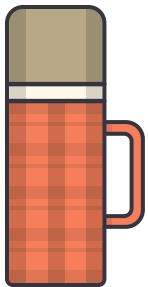
COMANDO:

GIT STATUS

1.6 COMITANDO

Observe como Git diz que as mudanças devem ser comitadas. Os arquivos listados aqui estão na área de rastreo (index), e eles ainda não estão no nosso repositório. Podemos adicionar ou remover arquivos do index antes de armazená-los no repositório.

Para armazenar nossas alterações indexadas, executamos o comando **commit** com uma mensagem descrevendo o que mudamos. Vamos fazer isso agora digitando:



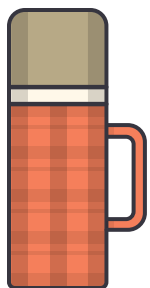
COMANDO:

GIT COMMIT -M "ADD CUTE OCTOCAT STORY"

1.7 ADICIONANDO TODAS AS MUDANÇAS

Ótimo! Você também pode usar caracteres curinga se você quiser adicionar muitos arquivos do mesmo tipo. Observe que eu adicionei um monte de arquivos .txt em seu diretório abaixo.

Coloquei alguns em um diretório chamado "octofamily" e alguns outros acabaram na raiz do nosso diretório "octobox". Felizmente, podemos adicionar todos os novos arquivos usando um curinga com git add.



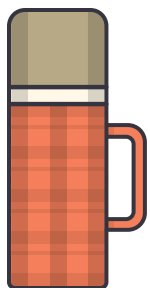
COMANDO:

GIT ADD '* .TXT'

1.8 COMITANDO TODAS AS MUDANÇAS

Ok, você adicionou todos os arquivos de texto ao index. Sinta-se livre para executar o `git status` para ver o que você está prestes a fazer commit.

Se parecer bom, vá em frente e execute:



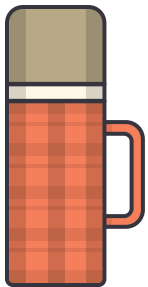
COMANDO:

`GIT COMMIT -M 'ADD ALL THE OCTOCAT TXT FILES'`

1.9 HISTÓRICO

Então fizemos alguns commits. Agora vamos navegar para ver o que mudamos.

Felizmente para nós, há git log. Pense no registro de Git como um diário que lembra todas as mudanças que comitamos até agora, na ordem em que as comitamos. Tente executá-lo agora:



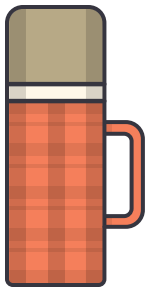
COMANDO:

GIT LOG

1.10 REPOSITÓRIO REMOTO

Bom trabalho! Criamos um novo repositório GitHub vazio para você usar com o Try Git em https://github.com/try-git/try_git.git. Para enviar o repositório local para o servidor GitHub, precisamos adicionar um repositório remoto.

Vá em frente e execute:



COMANDO:

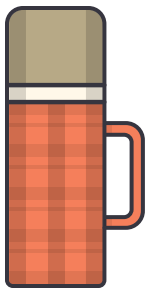
GIT REMOTE ADD ORIGIN (URL)

1.11 ENVIANDO PARA O REPOSITÓRIO REMOTO

O comando push diz ao Git onde colocar nossos commits quando estivermos prontos, e agora estamos prontos. Então, vamos enviar nossas mudanças locais para o nosso repositório de origem, também chamado de remoto (no GitHub).

O nome da nossa branch principal é **master**.

O -u diz ao Git para lembrar os parâmetros, para que da próxima vez passamos simplesmente executar git push e o Git vai saber o que fazer.



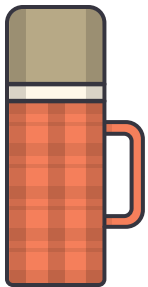
COMANDO:

GIT PUSH -U ORIGIN MASTER

1.12 BAIXANDO DO REPOSITÓRIO REMOTO

Vamos fingir que já passou algum tempo. Convidamos outras pessoas para o nosso projeto GitHub que tiraram suas alterações, fizeram seus próprios commits e push.

Podemos verificar alterações no nosso repositório GitHub e baixar para o repositório local quaisquer novas alterações executando:



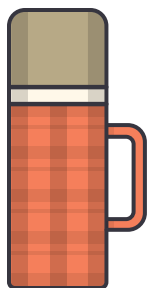
COMANDO:

GIT PULL ORIGIN MASTER

1.13 DIFERENÇAS

Uh oh, parece que houve algumas adições e mudanças na família octocat. Vamos dar uma olhada no que é diferente do nosso último commit usando o comando git diff.

Neste caso, queremos que o diff do nosso commit mais recente, que podemos referir-se a usar o ponteiro HEAD.



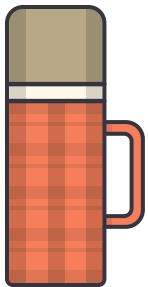
COMANDO:

GIT DIFF HEAD

1.14 DIFERENÇAS

Outro grande uso para diff é olhar para mudanças dentro de arquivos que já foram indexados. Lembre-se de que os arquivos indexados são arquivos que dissemos ao git que estão prontos para serem comitados.

Vamos usar git add para o stage `octofamily/octodog.txt`, que eu acabei de adicionar à família para você.

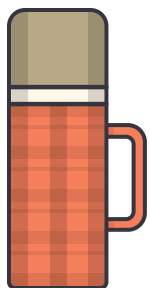


COMANDO:

GIT ADD OCTOFAMILY/OCTODOG.TXT

1.15 DIFERENÇAS

Bom, agora vá em frente e executar `git diff` com a opção `--staged` para ver as alterações que você acabou de indexar. Você deve ver que `octodog.txt` foi criado.



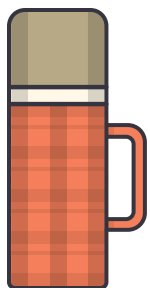
COMANDO:

`GIT DIFF --STAGED`

1.16 RESETANDO O INDEX

Então agora que o octodog é parte da família, o octocat está deprimido. Já que nós amamos o octocat mais do que o octodog, nós vamos mudar o seu olhar severo removendo o octodog.txt.

Você pode retirar do index arquivos usando o comando git reset. Vá em frente e remova octofamily/octodog.txt.



COMANDO:

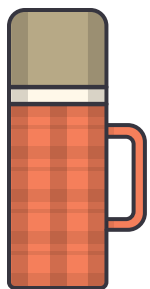
GIT RESET OCTOFAMILY/OCTODOG.TXT



1.17 DESFAZER

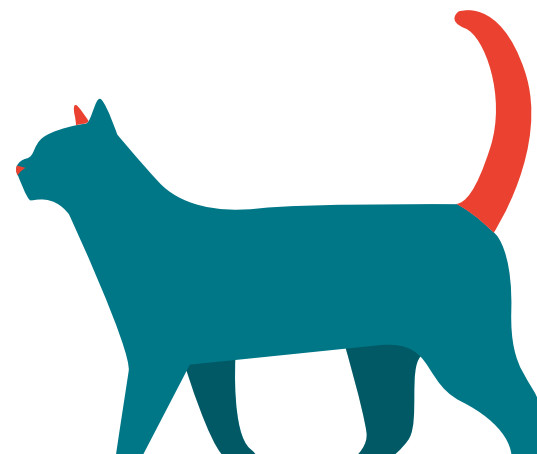
Git reset fez um ótimo trabalho parando o rastreo do octodog.txt, mas você vai notar que ele ainda está lá. Ele não está mais indexado. Seria ótimo se pudéssemos voltar para como as coisas eram antes que o octodog viesse arruinar a festa.

Os arquivos podem ser alterados de volta para a forma como estavam no último commit usando o comando: `git checkout - <target>`. Vá em frente e se livrar de todas as alterações desde o último commit para octocat.txt



COMANDO:

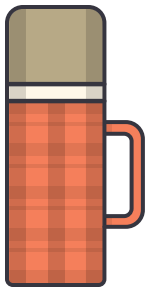
GIT CHECKOUT -- OCTOCAT.TXT



1.18 RAMIFICAÇÕES

Quando os desenvolvedores estão trabalhando em um recurso ou bug , muitas vezes criam uma cópia (branch) de seu código que podem fazer compromissos separados. Em seguida, quando eles são feitos, eles podem fundir esta branch de volta a sua branch máster, ou seja a principal.

Queremos remover todos esses octocats, então vamos criar uma branch chamado `clean_up`, onde faremos todo o trabalho:



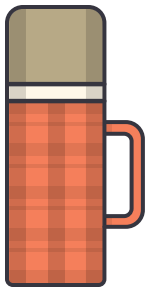
COMANDO:

GIT BRANCH CLEAN_UP

1.19 MUDANDO DE BRANCHES

Ótimo! Agora, se você digitar `git branch`, verá duas branches locais: a `master` e a nova chamada `clean_up`.

Você pode alternar entre branches usando o comando `git checkout <branch>`. Experimente agora para mudar para a branch `clean_up`:



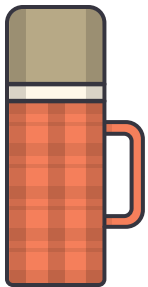
COMANDO:

GIT CHECKOUT CLEAN_UP

1.20 REMOVENDO TODAS AS COISAS

Ok, então você está na branch `clean_up`. Você pode finalmente remover todos aqueles octocats usando o comando `git rm` que não só removerá os arquivos reais do disco, mas também a remoção dos arquivos do index para nós.

Você vai querer usar um curinga novamente para obter todos os octocats em uma varredura, vá em frente e execute:



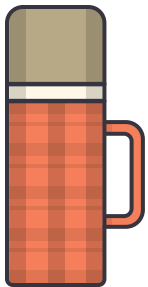
COMANDO:

`GIT RM '*.TXT'`

1.21 COMITANDO MUDANÇAS DA BRANCH

Agora que você removeu todos os gatos que você precisa para confirmar suas alterações.

Sinta-se livre para executar o `git status` para verificar as alterações que você está prestes a commitar.



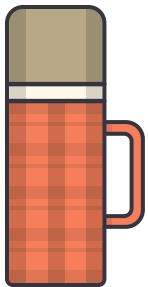
COMANDO:

`GIT COMMIT -M "REMOVE ALL THE CATS"`

1.22 VOLTANDO PARA BRANCH MASTER

Ótimo, você está quase terminando com o gato... e com o bug fix, você só precisa mudar de volta para a branch master para que você possa copiar (ou mesclar) suas alterações da branch `clean_up` de volta para a branch master.

Vá em frente e faça checkout da branch master:



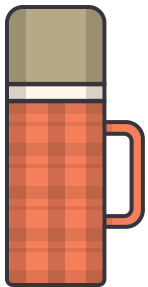
COMANDO:

GIT CHECKOUT MASTER

1.23 FAZENDO O MERGE

Certo, chegou o momento em que você tem que mesclar suas alterações da branch `clean_up` para a branch `master`. Respire fundo, não é tão assustador.

Já estamos na branch principal, então precisamos apenas dizer ao Git para mesclar a branch `clean_up` nele:



COMANDO:

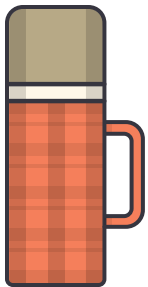
`GIT MERGE CLEAN_UP`

1.24 MANTENDO TUDO LIMPO

Parabéns! Você acabou de realizar seu primeiro bugfix bem-sucedido e mesclar as branches.

Uma vez que você terminou com a branch `clean_up` você não precisa mais dela.

Você pode usar `git branch -d <nome da branch>` para excluir uma ramificação. Vá em frente e exclua a branch `clean_up` agora:



COMANDO:

GIT BRANCH -D CLEAN_UP