Lista de comandos básicos do Git

Fazer login no git:

> git config --global user.name "Your Name"

> git config --global user.email "seu_email@gmail.com

> git config -- list

O terminal deverá estar em um diretório que escolhemos. Para inicializar um repositório Git nesse diretório, digite o seguinte comando:

> git init

Em seguida, digite o comando git status para ver qual é o estado atual do nosso projeto:

> git status

Você pode criar arquivos nesse repositório. Para dizer ao Git que comece a monitorar as alterações feitas nos arquivos criados, primeiro precisamos adicioná-lo à área de teste (Staging Area) usando git add.

> git status

> git add <nome_do_arquivo>.formato Ex: "octocat.txt"

> git status

Os arquivos listados aqui estão na área de teste, e eles ainda não estão em nosso repositório. Poderíamos adicionar ou remover arquivos na área de teste antes de armazená-los no repositório.

Para armazenar nossas mudanças em etapas, executamos o comando commit com uma mensagem descrevendo o que mudamos.

> git commit -m "comentario sobre o repositório"

Você também pode usar “o curinga *” se desejar adicionar muitos arquivos do mesmo tipo. Para isso digite:

> git add '* .txt'

> status git

> git git commit -m 'Adicionar todos os arquivos txt'

Agora que já sabemos como fazer os commits vamos ver quais alterações fizemos. Felizmente há o git log. Pense no log da Git como um jornal que lembra todas as mudanças que commitamos até agora, na ordem em que as commitamos. Tente executá-lo:

> git log

Use esse comando se quiser adicionar um repositório remoto

> git remote add <nome_do_repositorio> URL do repositorio

O comando push diz a Git onde colocar nossos commits quando estivermos prontos. Então, vamos fazer o push de nossas mudanças locais em nosso repo de origem (no GitHub).

O nome do nosso controle remoto é nome_do_repositorio e o nome do branch padrão é master. O -u diz a Git para lembrar os parâmetros, então a próxima vez que executarmos o git push o Git saberá o que fazer.

> git push -u origin master

Caso tenhamos convidado outras pessoas para o nosso projeto GitHub que fizeram suas mudanças, fizeram seus próprios commits podemos verificar mudanças no nosso repositório GitHub e commitar qualquer nova alteração executando:

> git pull original master

Podemos dar uma olhada no que há de diferente do nosso último commit usando o comando git diff.

> git diff HEAD

Você pode remover arquivos da area de teste com o comando git reset.

> git reset <nome_do_repositorio>/<nome_do_arquivo>.txt

Ao usar o git reset você vai perceber que o arquivo continua no seu diretório mas ele não está mais na sua area de teste, ou seja ele não será inserido no seu repositório do Github.

Os arquivos podem ser alterados de volta para como eles foram no último commit usando o comando: git checkout - <nome_do_arquivo>. Para se livrar de todas as mudanças desde o último commit digite:

> git checkout - <nome_do_arquivo>.txt

Quando os desenvolvedores estão trabalhando em um recurso ou bug, eles geralmente criam uma cópia (aka. Branch) de seu código em que eles podem fazer commits separados. Então, quando terminarem, podem juntar este branch de volta ao seu branch principal, o branch master.

> git branch <nome_do_branch_copia>

Agora, se você digitar git branch você verá dois branchs locais: um branch principal chamado master e seu novo branch chamado nome_do_branch_copia.

Você pode alternar branchs usando o comando git checkout <branch>. Experimente agora mudar para o ramo nome_do_branch_copia:

> git checkout nome_do_branch_copia

Agora você está no branch nome_do_branch_copia. Você pode finalmente remover todos os arquivos dele usando o comando git rm que não só removerá os arquivos reais do disco, mas também irá organizar a remoção dos arquivos para nós.

Você pode usar um “curinga” novamente para todos os arquivos em um formato de uma só vez:

> git rm '* .txt'

> status git

> git commit -m "Removendo tudo"

Você agora precisa voltar para o branch master para que você possa copiar (ou mesclar) suas mudanças do branch nome_do_branch_copia de volta para o branch principal. Volte ao branch master com:

> git checkout master

Chegou o momento em que você deve mesclar as mudanças do branch nome_do_branch_copia para o branch master. Nós já estamos no ramo principal, então precisamos apenas dizer ao Git que combine o branch cnome_do_branch_copia a ele:

> git merge nome_do_branch_copia

Você acabou de realizar seu primeiro bugfix e merge. Tudo o que resta para fazer é limpar repositório copia. Como você terminou com o branch nome_do_branch_copia, você não precisa mais dele.

Você pode usar git branch -d <nome_do_branch> para excluir um branch.

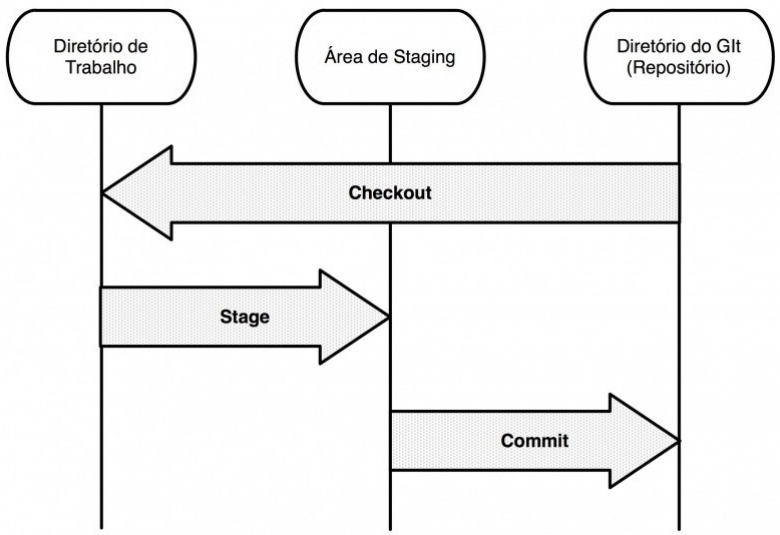
> git branch -d nome_do_branch_copia

O último passo. Tudo o que resta para você fazer agora é empurrar tudo que você fez no seu repositório remoto, e pronto! Para isso use o comando:

> git push

Como clonar repositório pelo terminal (precisa determinar o seu diretório no computador antes)

> git clone "https://"



git init	Inicializa um repositório GIT no diretório corrente
git clone <endereço>	Clona todo conteúdo de um repositório remoto para o diretório corrente e inicializa o repositório local
git add <arquivo>	Adiciona arquivos/diretórios para área de staging. Este comando também é utilizado para rastrear novos arquivos/diretórios
git commit [–a] -m “<mensagem>”	Submete as alterações ao repositório local. Deve-se informar uma mensagem que justifique o commit. O argumento -a informa o que o GIT deverá submeter todas as alterações de todos os arquivos e não somente aqueles que se encontram no status Staged
git rm <arquivo>	Remove um arquivo/diretório do diretório atual e informa que GIT deverá remover este arquivo no próximo commit
git mv <orig> <dest>	Move um arquivo de origem para um determinado destino. O comando pode ser utilizado também para renomear arquivos/diretórios
git reset	Reverte todas as modificações realizadas nos arquivos e ainda não submetidas
git branch <branch>	Cria um novo branch de nome <branch>
git checkout [-b] <branch>	Troca o branch de trabalho atual para o branch de nome <branch>. Caso o argumento -b seja informado o GIT cria um branch novo.
git remote add <endereço>	Adiciona um link de um repositório remoto ao repositório local. Um repositório GIT pode conter vários repositórios remotos
git push <remote> <branch>	Submete todas as alterações armazenadas no repositório local ao repositório remoto de nome <remote> em um branch específico. O branch é automaticamente criado caso não exista no repositório remoto
git fetch	Copia todos os dados do repositório remoto de nome <remote> para o repositório local. As alterações não são incorporadas ao repositório local. Este comando é semelhante ao git clone, exceto que é executado em um repositório existente
git pull <remote> <branch>	Copia todas as mudanças do repositório remoto de nome <remote> e reintegra ao repositório local. Caso o argumento <branch> não seja informado, o comando é executado sobre o branch de trabalho atual