MINI CURSO DE GIT (1º Dia)

Links Interessantes

Curso interativo de GIT

https://www.codeschool.com/courses/try-git

Site com exemplos fáceis de vários operações no GIT

http://gitready.com/

Documentação do Próprio site do GIT

https://git-scm.com/doc

Site com várias perguntas e respostas referente ao GIT

https://stackoverflow.com/questions/tagged/git

Esse mini-curso está dividido em 5 partes:

1ª Conceitos Chaves – Essa parte explica conceitos que são importantes para a compreensão do git (coisas como branchs, commits, index, …)

2ª Trabalhando com Repo Local – Essa parte mostra como utilizar o repositório local, efetuar commits, criar branchs, restaurar uma branch ou commit para o diretório de trabalho, indexar alterações para o próximo commit, ...

3ª Trabalhando com Repo Remoto – Nessa parte são mostradas as operações para adicionar um repositório remoto (git remote add), buscar atualizações desse repositório (git fetch) e enviar suas alterações para esse repositório (git push) e buscar as alterações que outros membros da equipe enviaram para o repositório remoto (git pull). Além disso aprendemos o comando git clone que literalmente clona um repositório remoto para sua máquina local.

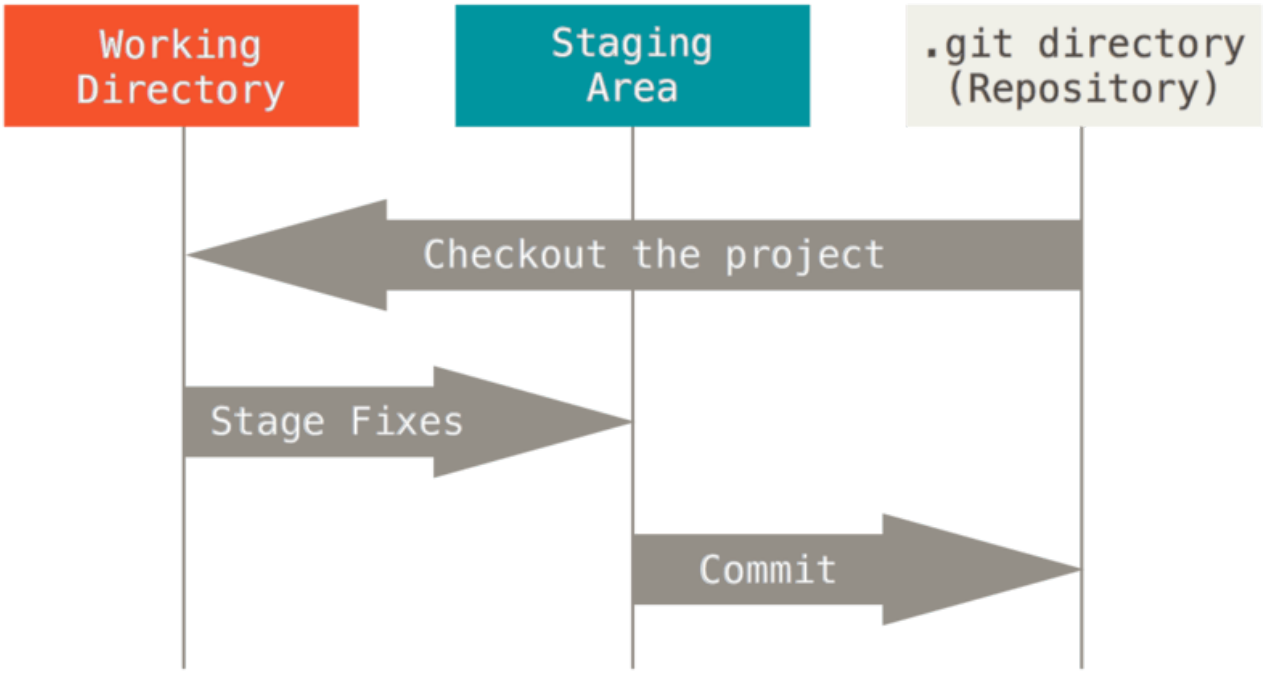
4ª Github – Nessa parte você aprenderá a diferença entre Fork e Clone, o que é um Pull Request (Merge Request), o que são issues e a Markup Language.

5ª Ferramentas Interessantes – Aqui são mostradas algumas ferramentas interessantes que facilitam o uso do GIT. (meld, gitk, tig)

1. CONCEITOS CHAVES

Estados

- **Diretório de Trabalho:** É o diretório no qual estamos trabalhando.
- **Index (Staging Area):** Inclui as alterações que farão parte do próximo commit.
- **Repositório (diretório .git):** Contém de fato os commits.



Commit - Representa um “snapshot” dos seus dados estagiados naquele momento. (salva de fato as alterações no repositório)

Branch - são “estradas” diferentes “paralelas” na qual fazemos alteração sem afetar o branch principal. Depois esses caminhos podem se “fundir” atravéz do merge.

HEAD - Aponta para o commit em cima do qual você faz alterações no seu diretório de trabalho)

Tags - São commits especiais que representam uma determinada versão do software)

2. Trabalhando com Repositório Local

1. git init

Inicia um repositório git dentro do diretório atual.

2. git help (man git)

Obtém ajuda sobre um comando no git. (Importante entender os conceitos-chave porque o manual se refere a eles frequentemente)

Ex.: $git help init

3. git status

Mostra o branch atual, se há arquivos não monitorados, quais arquivos estão indexados para o próximo commit.

git add

Se o(s) arquivo(s) informado(s) para o index (com isso eles farão parte do próximo commit)

Ex.: $git add carros.txt

git commit

Gera um “snapshot” com os arquivos que estão indexados (no staging area). Esse commit vai fazer fazer parte da “lista” de commits do branch atual.

Ex.:
$git commit -m “Mensagem do commit”

git log [--oneline | --pretty | --abbrev-commit | --graph]

Mostra o log (histórico) de commits do repositório.

Ex.:
$git log   (mostra o histórico de logs do branch atual)
$git log --oneline (mostra o histórico de commits um commit por linha)
$git log –oneline –graph (mostra gráfico dos commits da branch atual)

git diff

Mostra a diferença entre o diretório de trabalho e o index ou entre index e repositório.

Ex.:
$git diff carros.txt
$git diff (mostra a diferença entre o diretório de trabalho e o index)
$git diff –cached (mostra a diferença entre o index e o repositório)
$git diff HEAD~..HEAD~~ (mostra a diferença entre o penúltimo e o anti penúltimo commit)

git branch [-D] [branch] :[remote/remote_branch]

Comando para criar, remover e renomear branchs (tanto locais quanto remotas)

Ex.:
$git branch atualizacao (cria a branch atualização)
$git branch -D atualizacao (remove a branch atualizacao)

git checkout [file | branch | remote/remote_branch] [-b]

Traz determinado commit (ou branch) para o diretório de trabalho.

Ex.:
$git checkout correcao (carrega para o diretório de trabalho a branch correcao)
$git checkout bAce732 (carrega para o diretório de trabalho o conteúdo do commit bAce732)
$git checkout -b nova_branch (cria uma nova branch e faz o checkout para ela)

git merge

Faz a fusão entre o dois branchs (as duas “estradas paralelas” se juntam trazendo o histórico de commit do outro branch e gerando um commit de merge)

Ex.:
$ git checkout master (muda para o branch que irá conter o merge)
$ git merge correcao (faz o merge do branch correcao no branch atual)

.gitignore

Esse arquivo serve para ignorar arquivos que não devem ser versionados. (Por exemplo códigos compilados)

Ex.:

$vim .gitignore

*.o (muito usado em projetos c/c++ para ignorar código objetos com extensão .o)

https://github.com/github/gitignore
(contém exemplos de arquivos .gitignore para diferentes tipos de linguagens)

2. Trabalhando com Repositório Remoto

Fork

Fazer um fork é criar um repositório “filho” de um outro repositório. O github e gitlab está cheio de repositórios mas não podemos fazer alterações livremente nesses repositórios porque não somos os donos. Quando fazer um fork, criamos uma cópia da qual somos donos e podemos fazer qualquer alteração. Essa operação é feita pela interfaces do GitHub e GitLab.

git clone

O comando git clone serve para fazer um cópia de um repositório remoto (geralmente no GitHub ou Gitlab)

Ex.:
git clone git@github.com:jplobianco/plugin-senhas-face.git

git fetch

Busca as alterações (metadados) que existem no repositório remoto.

Ex.:
git fetch origin

git pull

Serve para buscar as alterações que foram feitas no repositório remoto.

Ex.:

git pull origin master

(Pega as alterações que foram feitas no branch master do repositório remoto origin)

git push

Serve para publicar as nossas alterações no repositório local para o repositório remoto.

Ex.:

git push origin master

(nesse exemplo estamos publicando nossas alterações para o repositório remoto origin na branch master)