

INTRODUÇÃO AO GIT

1 Tema

[HELP](#) / [toggle view](#)

Introdução ao Git e Github

2 Objetivo

Dar suporte aos alunos que se inscreverem no curso "Como Usar Git e Github" do MOOC Udacity, disponível <https://br.udacity.com/course/how-to-use-git-and-github--ud775/>

3 Conteúdo

Exercícios que contemplam os seguintes tópicos

3.1 Gerencia de mudanças em um ou mais arquivos com diff

Contents:

- [3.1 Gerencia de mudanças em um ou mais arquivos com diff](#)

4 Competencias

Após o curso do Udacity e dos exercícios o aluno terá adquirido a competência de trabalhar em equipe de desenvolvimento de software bem como compartilhar seu código no github.

5 Metodologia

Este material não tem muita teoria e explicações porque isso tem no curso da Udacity. O que tem aqui são exercícios baseados no curso bem como a resposta dos exercícios em forma de tutoriais e vídeo aulas.

6 Duração

Ainda não tenho muito claro, estou ainda fazendo esses exercícios

7 Avaliação

As avaliações serão realizadas em sala de aulas já que entregas de trabalhos poderão ser requisitadas via repositórios por alguns professores.

8 Recursos necessários

Instalação do git

9 por que usar git?

Você deveria usar o git porque o mercado usa e existem muitos bons motivos pra isso. Se eu fosse você eu

iria querer saber quais motivos são esses. Ou seja, porque profissionais do mercado investem tempo aprendendo e estudando pra usar o git? Mesmo porque não é tão simplzinho assim aprender git, então você tem que ter claro o porquê vale a pena o esforço de aprender.

Respondendo a pergunta, sendo bastante simplório, mas bem claro e objetivo, imagine a seguinte situação: Seu grupo de TCC, de três alunos, por exemplo, está editando um texto. Pode ser o texto do TCC mesmo. Então um de vocês digita uma parte, e outro outra e assim vai... é comum ocorrer esse tipo de divisão esquisita de trabalho como por exemplo: um escreve o capítulo 1 o outro o capítulo 2 e depois junta num arquivo só. Acho que isso nunca fica bom, mas no início não dá tanto problema. O problema começa quando precisa fazer ajustes no texto e aí cada um pega uma cópia e faz ajustes e várias partes do texto. E agora pra juntar? Como você faria? Aproveita a oportunidade e pensa se ao invés de um arquivo só vocês estivessem trabalhando com uma cópia de vários arquivos de código e aí cada um faz várias alterações em partes diversas de vários arquivos. Junta tudo isso num código final como?

10 1 Vendo as diferenças entre dois arquivos

10.1 Exercício 1.1 -> diff pra ver diferença entre arquivos

(a) Crie uma pasta com o nome projeto-git-intro e um (b) arquivo com nome de nomeie "arqv1.txt" e (c) escreva nele o seguinte: A B C D

(d) Faça uma cópia desse arquivo e (e) renomeie como arqv2.txt (f) Aí você deleta o A (g) Acrescenta no final do arquivo a Letra E.

(h)Faça um print do diff e (i)interprete as mudanças com base da saída do diff.

O que significa a indicação de mudança de linha com sinal de menos? O que significa a indicação de mudança de linha com sinal de mais?

10.2 Resolução do exercício 1.1

Bom, vamos lá então... resolver o exercício 1.

Vou criar a pasta e o arquivo solicitado. Bom eu tenho no meu hd externo uma pasta que eu chamo de env-dev que significa pra mim "environment develop" ou seja, uma pasta onde eu crio meus projetos. Você pode escolher criar a pasta em qualquer diretório que vc queira, mas pelo amor de Deus... não coloca caracteres especiais no nome nem use espaços... funciona, mas vai se acostumar a não usar isso...

```
mkdir -p /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/
touch arqv1.txt
ls -la
```

```
total 12
drwxrwxr-x. 2 wagner wagner 4096 Jan  2 22:25 .
drwxrwxrwx. 8 wagner wagner 4096 Jan  2 22:25 ..
-rw-rw-r--. 1 wagner wagner   8 Jan  2 22:28 arqv1.txt
```

Criamos o arquivo, agora vamos escrever nele conforme solicitado no exercício.

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/
echo "A" > arqv1.txt
echo "B" >> arqv1.txt
echo "C" >> arqv1.txt
echo "D" >> arqv1.txt
```

```
cat arqvil.txt
```

A
B
C
D

Bom pessoal, fizemos as tarefas (a) que era a criação da pasta, (b) que é criação do arquivo e (c) escrevemos no arquivo criado o que foi solicitado. Foi utilizado aqui a linha de comando pra fazer tudo isso mas vc pode usar qualquer editor de texto. Pode fazer do jeito mais fácil pra vc. O importante é entender a parte do git, ok?

Bom, vamos continuar fazendo o que o exercício pede... Vamos fazer uma cópia do arquivo renomeando para arqv2.txt, conforme solicitado em (d) e (e).

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/  
sed -i -e "ld" arqvi2.txt  
echo "E" >> arqvi2.txt  
cat arqvi2.txt
```

B
C
D
E

A letra (f) pedia pra deletar a letra A. Usei o comando "sed -e 's/A//g' arqvi2.txt" pra fazer isso. Você não precisa fazer assim se não quiser ou se achar difícil. Fiz assim só pra enriquecer nossa experiência e também porque fica mais fácil pra escrever a resolução do exercício. Já o comando "echo "E" > arqvi2.txt" escreve a letra "E" no final do arquivo, conforme pedia a letra (g).

O último comando "cat" mostra faz um print do conteúdo do arquivo arqvi2.txt pra mostra que a gente fez o que foi solicitado.

Agora as letras (h) pede pra gente fazer um diff e a letra (i) pede pra gente interpretar a saída do diff.

Considerando que as mudanças que a gente fez no arquivo "arqvi2.txt" foi bem, mas bem simples mesmo a gente até tem uma idéia do que o diff vai mostrar. Vai mostrar que a gente deletou a letra A e que a gente acrescentou a letra E no final.

Pô assim é fácil. Mas entenda que a idéia não é descobrir qual foi a mudança que a gente fez, mas sim ver como o diff mostra pra gente essa mudança. entendeu?

Bom então vamos lá...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/  
diff -u arqvil.txt arqvi2.txt
```

```
0 [wagner@localhost projeto-git-intro]$ diff -u arqvil.txt arqvi2.txt  
1 --- arqvil.txt 2017-01-02 22:29:30.078332174 -0200  
2 +++ arqvi2.txt 2017-01-03 01:20:40.826569072 -0200  
3 @@ -1,4 +1,4 @@  
4 -A  
5 B  
6 C  
7 D
```

```
8 +E
9 [wagner@localhost projeto-git-intro]$
```

Agora vamos interpretar essa saída (i) linha por linha...

Linha	Interpretação
0	Comando
1	Indica o primeiro arquivo do o diff tá comparando mais o timestamp
2	Indica o segundo arquivo que o diff tá comparando mais o timestamp
3	"-" significa arqvi1.txt. 1,4 significa que as mudanças começam na linha 1 e envolve 4 linhas
3	"+" significa arqvi2.txt. 1,5 significa que as mudanças começam na linha 1 e envolve 5 linhas
	O sinal "-" indica o arquivo original e o "+" o novo.
4	"-" significa que foi deletado o caractere "A" que é o conteúdo da linha deletada
5	B não foi modificado, ou seja, tinha no arqvi1.txt e continua tendo no arqvi2.txt
6	C idem
7	D idem
8	"+" Indica que essa linha foi acrescentada em arqvi2.txt e o conteúdo da linha é "E"
9	é o prompt esperando um novo comando

10.3 Exercício 1.2 -> diff entre dois arquivos do jogo asteróids

Conforme solicitado no curso sobre git e github do Udacity, fazer o mesmo, ou seja, usar a ferramenta diff -u com os dois arquivos do game asteroids e interpretar as diferenças.

Por que fazer o mesmo exercício duas vezes? Não é o mesmo exercício exatamente não.. A diferença desse exercício pro exercício anterior é que no exercício anterior o foco era fazer você ver COMO O COMANDO diff -u mostrava a diferença, mesmo porque você já sabia de cor quais eram as diferenças entre os dois arquivos e nesse o foco é ver quais são as mudanças mesmo porque não dá pra saber quais são as diferenças só no visual, ou seja, precisa mesmo de uma ferramenta pra isso.

Então pra ficar e pra compreender melhor o objetivo desse exercício 2, por favor não tenha pressa não... vai com calma...

Reponda pra mim... quer dizer pra você mesmo... Como você faria se você tivesse com saber quais as diferenças entre esses dois arquivos apenas comparando os dois? Então, depois de tentar responder, você usa o diff e vai perceber que é uma ferramenta muito legal e útil.

Os dois arquivos de que estamos falando são esses... https://storage.googleapis.com/supplemental_media/udacityu/2960778928/game_old.js https://storage.googleapis.com/supplemental_media/udacityu/2960778928/game_new.js

10.4 Resolução do exercício 1.2

Bom vamos pra resolução do exercício 1.2? Já temos uma pasta "projeto-git-intro" Vamos colocar esses dois arquivos nesta pasta mesmo, beleza?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/
wget https://storage.googleapis.com/supplemental_media/udacityu/2960778928/game_old.js
wget https://storage.googleapis.com/supplemental_media/udacityu/2960778928/game_new.js
ls -la
```

```
total 144
drwxrwxr-x. 2 wagner wagner 4096 Jan  3 01:53 .
drwxrwxrwx. 8 wagner wagner 4096 Jan  2 22:25 ..
-rw-rw-r--. 1 wagner wagner   8 Jan  2 22:29 arqvil.txt
-rw-rw-r--. 1 wagner wagner   8 Jan  3 01:20 arqvi2.txt
-rw-rw-r--. 1 wagner wagner 30688 Set 17  2014 game_new.js
-rw-rw-r--. 1 wagner wagner 30688 Set 17  2014 game_new.js.1
-rw-rw-r--. 1 wagner wagner 30683 Set 17  2014 game_old.js
-rw-rw-r--. 1 wagner wagner 30683 Set 17  2014 game_old.js.1
```

A gente percebe pelo ls que o wget fez o download dos arquivos game_{new}.js e game_{old}.js. Os outros com final 1 é porque rodei duas vezes o bloco de comandos acima e o wget fez o download de novo.

Então, conforme a gente tava conversando... Abra aí esses dois arquivos lado a lado e tente ver quais são as diferenças entre eles...

Eu espero... vai lá... aproveita a oportunidade...

Percebeu que não é tão fácil e é com certeza, muito inseguro. Lembre que poderia ser um arquivo de código do seu tcc e poderia não ser só um simples exercício, poderia ser uma situação real.

Bom, agora vamos fazer o diff -u...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/
diff -u game_old.js game_new.js
```

```
[wagner@localhost projeto-git-intro]$ diff -u game_old.js game_new.js
--- game_old.js 2014-09-17 12:37:22.000000000 -0300
+++ game_new.js 2014-09-17 12:37:44.000000000 -0300
@@ -4,9 +4,9 @@
 //

KEY_CODES = {
+ 13: 'enter',
  32: 'space',
  37: 'left',
- 38: 'up',
  39: 'right',
  40: 'down',
  70: 'f',
@@ -392,7 +392,7 @@
     this.vel.rot = 0;
 }

-  if (KEY_STATUS.up) {
+  if (KEY_STATUS.space) {
+    var rad = ((this.rot-90) * Math.PI)/180;
     this.acc.x = 0.5 * Math.cos(rad);
     this.acc.y = 0.5 * Math.sin(rad);
@@ -406,7 +406,7 @@
     if (this.delayBeforeBullet > 0) {
       this.delayBeforeBullet -= delta;
     }
-  if (KEY_STATUS.space) {
+  if (KEY_STATUS.enter) {
     if (this.delayBeforeBullet <= 0) {
       this.delayBeforeBullet = 10;
       for (var i = 0; i < this.bullets.length; i++) {
@@ -919,7 +919,7 @@
     waiting: function () {
       Text.renderText(ipad ? 'Touch Sreen to Start' : 'Press Space to Start', 36, Game.canvas
       if (KEY_STATUS.space || window.gameStart) {
-        KEY_STATUS.space = false; // hack so we don't shoot right away
```

```
+      KEY_STATUS.space = false; // hack so we don't move right away
      window.gameStart = false;
      this.state = 'start';
    }
[wagner@localhost projeto-git-intro]$
```

Humm... Vamos dar uma analisada então nas modificações...

trecho	Interpretação
+ 13: 'enter',	Foi adicionado o código 13 relacionando-o com a tecla enter
- 38: 'up',	Foi excluída linha que relacionava o código 38 com a tecla up
@@ -4,9 +4,9 @@	Essas modificações são bem no início do arquivo, linha 1 a 4.
	Veja se não é isso mesmo na figura 1.
@@ -392,7 +392,7 @@	O arquivo vai mostrar modificações que começam na linha 292 tanto do arquivo
	de um arquivo como do outro. São modificações que estão lá no meio dos arqs
- if (KEYSTATUS.up) {	Dá pra perceber que o programador apagou essa linha que usava a tela "up"
+ if (KEYSTATUS.space) {	e acrescentou essa, substituindo portanto, pela tecla space
	No exercício do curso, era pra achar esse "typo" "spacr"
	Type significa em inglês um erro de digitação já que como você pode ver nas linhas
	que definem o KEYCODES que era pra se escrito "space"
@@ -406,7 +406,7 @@	Mais modificações...
- if (KEYSTATUS.space) {	O programador fez a mesma mudança da tecla up e space
+ if (KEYSTATUS.enter) {	Só que agora ele escrever space corretamente.
@@ -919,7 +919,7 @@	Mais modificações
- KEY_STATUS.space = false; // hack so we don't shoot right away	Dá pra perceber pelos comentários do programador que a tecla space era relacionada
+ KEY_STATUS.space = false; // hack so we don't move right away	a ação de atirar (shoot) agora está relacionado a movimentar (move)

Agora responde pra mim, você acha que apenas visualizando o arquivo lado a lado a gente conseguiria ter essa visão clara e segura das modificações realizadas no arquivo? Então, pegar habilidade com diff vai te recompensar mais tarde, pode ter certeza. Só uma nota adicional. O comando diff pode ser utilizado sem o parâmetro "u". A saída fica um pouco diferente e pra não ficar muito longo o nosso material de apoio a gente usou o diff só com a opção "u". Essa opção não é sem motivo. No futuro quando você for contribuir com algum código de outro programador é comum que suas modificações sejam analisadas no formato unificado, ou seja, com a opção "u" mesmo. O formato unificado é um formato padrão e por isso é bastante utilizado, dê uma olhada no link https://en.wikipedia.org/wiki/Diff_utility Se eu achar um em

português substituo depois.

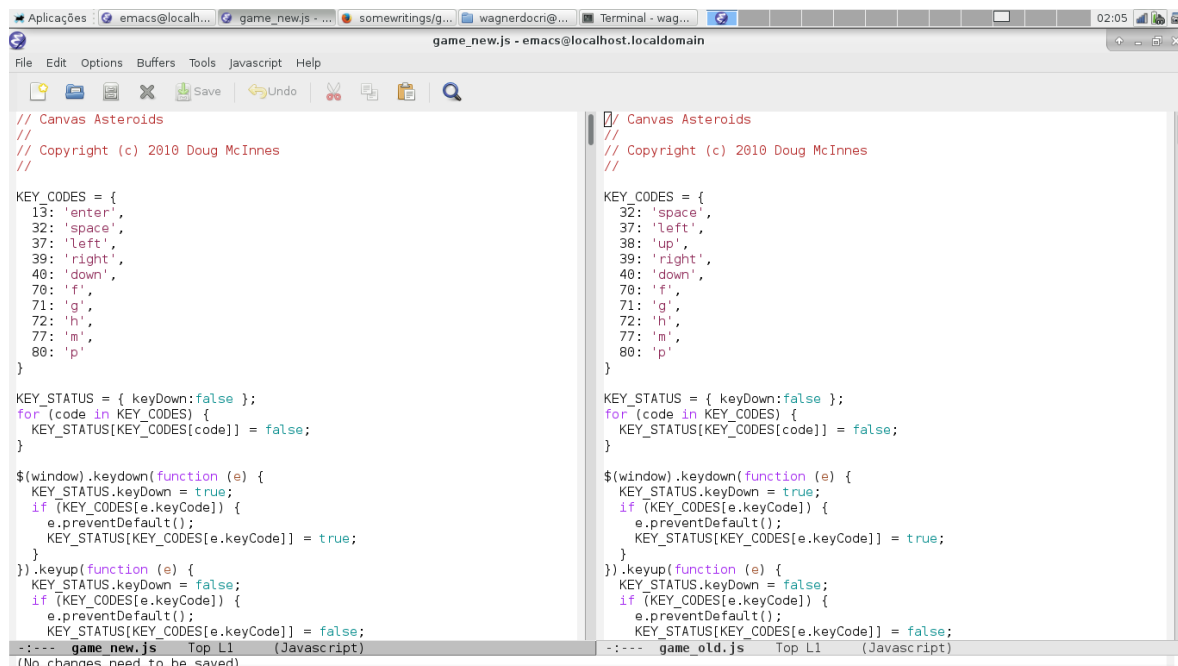


Figure 1: Arquivos game_{new}.js e game_{old}.js lado a lado

Contents:

- [10.1 Exercício 1.1 -> diff pra ver diferença entre arquivos](#)
- [10.2 Resolução do exercício 1.1](#)
- [10.3 Exercício 1.2 -> diff entre dois arquivos do jogo asteróids](#)
- [10.4 Resolução do exercício 1.2](#)

11 2 Vendo o histórico dos arquivos

11.1 Exercício 2.1 -> Não simplesmente ver uma diferença, é legal ver o histórico

É legal, é importante ver a diferença entre duas versões diferentes de arquivo. É útil e importante. Mas seria muito mais interessante ainda ver um histórico inteiro de mudanças neste arquivo. Aí sim, é muito mais interessante ainda...

Você precisa de dois conceitos pra entender isso. Claro que você assistiu aos vídeos e viu que foi falado de CVS - Concurrent Version System e de Commits, em particular commits manuais. (Não vou repetir aqui não... vê os vídeos lá, ou pergunte para o seu professor) Dica. Procure o vídeo sobre Concept Map.

Só pra contextualizar você precisar entender um conceito muito importante. Se você está interessado em fazer um diff de uma versão 1, e da 3 você precisa de uma ferramenta de software que consiga gerenciar essas mudanças pra você. Ou seja, uma ferramenta que saiba o que mudou da versão 1 pra versão 2 e 3 pra que seja possível você fazer um diff entre elas. Por isso você precisa instalar o git e inicializar o git para a pasta onde o seu arquivo se encontra pra que o git guarde o histórico das mudanças no código pra você.

Portanto, Neste exercício você deve fazer as seguintes tarefas:

1. Instale o git na sua máquina Dê os seguinte comandos Rode os os seguintes comandos: git config --global color.ui auto git config global user.name "seu nome" git config global user.email

"seu.email@domimio.com"

1. **Criando pasta para o projeto** Crie uma pasta pra sua interface gráfica html, pode chamar ela de projweb_{gui}

1. **Comando git init** Inicialize o git com git init

1. **Criando um arquivo html com tags de estrutura** Crie seu arquivo e crie a estrutura de um html nele (tags html, head e body)

1. **Usando commando git add** Adicione esse arquivo para ser controlado pelo git

1. **Usando o comando Commit** Commite com msg ("initial commit com html estrutura básica)

1. **Editando o arquivo html** Abra o arquivo novamente coloque um H1 com titulo e um paragrafo com a descrição do site

1. **Usando o comando Commit** Commite com a msg (Titulo e Descrição do site) Use o git diff e o git log pra acompanhar as modificações que você fez.

Agora use os comandos git log e git diff e interprete a saída. 11b) Explique porque diff opera sobre commits. //experimente também Aproveite pra usar essas variações do git log git log --oneline -2 (linhas)

1. **Refletindo: Tralhando off line** Interprete a saída do git log Lembra daquele menina na floresta dizendo que estava offline. Perceba que você não precisou de internet pra fazer isso certo? Ou seja, tudo isso que você está fazendo é offline.

1. **Reflexão entre commit manual e automático** Lembra que as meninas do curso explicaram que o wikipedia, o dropbox o google drive também fazem controles de versão de um documento? Só que os commits, ou seja, o registro das mudanças ão é manual, é automática, sendo registrado sempre que você salva por exemplo. Já com o git, o commit é manual? Quais são as vantagens e desvantagens de commits manuais? Ah. faça uma reflexão também sobre qual seria o melhor momento para commitar alterações dizendo o porque de sua conclusão sobre o assunto.

1. **Trabalhando com mais confiança** Já aconteceu com você de você fazer uma alteração em algum arquivo ou projeto e depois não saber muito bem onde você errou e decidir dar um control z pra voltar ao que era antes e começar de novo? Já aconteceu comigo de querer voltar pra uma versão de ontem, por exemplo, já que o control z não estava disponível. Como essa o git facilita a nossa vida diante de situações como essa?

1. **Trabalhando com multiplos arquivos** Claro que um css deve sempre que possível ser escrito num arquivo a parte. Faça isso com seu estilo e link ele no seu html.

12 a) De um git status antes de commitar 12 b) Commite 12 c) De um git status de novo

11.2 Resolução do Exercício 2.1 -> Não simplesmente ver uma diferença, é legal ver o histórico

Pra resolução desse exercício você vai precisar instalar o git. Tem um monte de tutorial na internet, essa parte é tranquila. No meu sistema o git já está instalado, vamos conferir? To rando o comando abaixo na minha máquina...

```
git --version
```

Eis o resultado..


```
git version 2.4.11
```

Agora, vou me identificar para o git. Me identificar? O git vai gerenciar o meu código, é ele quem vai viabilizar um histórico de modificações que eu faço no meu código... Nada mais justo do que ele saber quem é que tá fazendo as modificações, voce não acha? Afinal se alguém fazer alguma besteira, é bom saber quem foi pra poder orientar a pessoa. Então vamos lá, vou me identificar com os seguintes comandos.

```
git config --global color.ui auto
git config global user.name "wagnermaques"
git config global user.email "wagnerdocri@gmail.com"
```

Items 2. Criando uma pasta para o projeto...

```
mkdir -p /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/en
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
echo $(pwd)
```

Pessoal, e tranquilo os comandinhos né? Usei mkdir com opção "-p" pra criar a pasta "projExercicio2" no caminho especificado. Depois deu um change dir "cd" pra entrar na pasta que eu criei E dei um echo \$(pwd), que pode ser um pouco estranho pra alguns de vocês, mas o que isso faz é simplesmente um echo da saída pwd que é o caminho do diretório que eu estava, o projExercicio2. O mais importante é criar a pasta, você pode fazer do seu jeito, e se ficar com dúvida sobre esses comandinho pode perguntar pro professor.

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-de
```

Item3 Esse item pede pra gente inicialiar o git com o comando git init. Sem muita teoria, porque isso vcs podem pegar lá no curso da udacity, mas dando uma pincelada, você precisa entender o seguinte:

O git controla tudo que a gente faz dentro da pasta do projeto que é a projExercicio2. Como eu sei que o git não tá controlando nada ainda? Eu vou saber isso se eu listar o conteúdo da pasta e não tiver um diretório oculto ".git" dentro da pasta. Se tiver o diretório o git tá inicializado e controlando, se não tiver, o git não tá controlando nada. A gente sabe que no momento o git não tá controlando nada porque a gente acabou de criar a pasta, então se eu listar o conteúdo da pasta do projeto não vai ter esse diretório oculto ".git". Vamos conferir isso então?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
ls -la
```

Pelo resultado do comando você pode constatar que não tem a pasta .git

```
total 8
drwxrwxr-x. 2 wagner wagner 4096 Jan  3 16:03 .
drwxrwxrwx. 9 wagner wagner 4096 Jan  3 16:03 ..
```

Mas nós estamos consados de ver só diff entre dois arquivos a gente quer ver o histórico de todas as modificações e tudo que o git pode fazer por nós, então vamos inicializar o git para a pasta do nosso projeto...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
```

```
git init
```

Após rodar o comando você pode ver que a mensagem é bem clara. Se tiver dificuldades, cola no google translate pra você ver...

```
Initialized empty Git repository in /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467b
```

Bom é claro que agora a gente vai dar um ls -la de novo pra ver se a pastinha .git tá lá...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
ls -la
```

Tá, não tá?

```
total 12
drwxrwxr-x. 3 wagner wagner 4096 Jan  3 16:15 .
drwxrwxrwx. 9 wagner wagner 4096 Jan  3 16:03 ..
drwxrwxr-x. 7 wagner wagner 4096 Jan  3 16:15 .git
```

Pois é... agora a gente pode usar o git, já que é ele quem viabiliza pra gente o histórico das modificações que a gente vai fazer...

Item4 Vamos criar um arquivo html só coms tags de estrutura, conforme solicita o exercício...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
touch index.html
echo "<!DOCTYPE html>" > index.html
echo "<html>" >> index.html
echo "<head>" >> index.html
echo "<title>Page Title</title>" >> index.html
echo "</head>" >> index.html
echo "<body>" >> index.html
echo "</body>" >> index.html
echo "</html>" >> index.html
cat index.html
```

Eis o conteúdo do nosso index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
</body>
</html>
```

item5 O item 5 fala pra gente usar o git add pra adiciona-lo ao controle do git.

Vamos fazer isso sim, mas vamos primeiro dar uma olhadinha como a gente sabe que o nosso novo arquivo index.html não tá adicinado. A gente precisa saber, pra que gente poder sabe o que estamos fazendo... Como eu sei que o index.não está dicionado? com o git status Faz sentido não é? vou dar uma olhada no estatus desse arquivo... Vamos faze então...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
```

```
git status
```

wow... pera lá... vamos ler isso.. "on branch master" (deixa isso pra depois...) "Initial commit" (Também essa questão do commit, daqui a pouco a gente fala sobre isso)

Untracked files: (use "git add <file>..." to include in what will be committed)

Ah.. untracked files... agora o git tá começando a falar o que eu quero saber...

Na sequencia o git lista os arquivos que estão untracked, ou seja, ainda não gerenciados pelo git.

O git é educado, ele não mecher em nada seu sem você falar pra ele fazer isso.

Por isso ele tá dizendo que o nosso arquivo index.html é untracked.

Mas é claro que a gente quer que o git cuide das modificações nesse arquivo pra gente então vamos dizer pra ele fazer isso.. e é agora que entra o comando git add.

```
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    index.html

nothing added to commit but untracked files present (use "git add" to track)
```

Bom já interpretamos a saída do git status, agora vamos adicionar o nosso arquivo index.html para o git gerenciar ele pra gente... Vou aproveitar e dar um git status de novo na sequencia pra vc poder comparar as duas saídas ok?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git add index.html
git status
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   index.html
```

Opa.. O git tá nos dizendo que o nosso arquivo index.html está com status "new file". Já não é mais untracked. Beleza.

Ah.. antes de ir para o proximo passo da resolução do exercício, responde pra mim...

Esse arquivo sofreu alguma modificação? Não né? Parece que não mais isso é importante porque o estado desse arquivo, você pode considerar também como não modificado. O git sabe que depois que agente adicionou o arquivo pra ser gerenciado não ocorreu mudança nenhuma. Vamos modifica-lo da aqui a pouco.

Item 6 **Usando o comando Commit** Eu sei que você aprendeu o que é commit nos vídeos do curso. Mas vamos refrescar um pouco a nossa memória... O commit é um conceito importantíssimo pra gente quando

usamos o git. A primeira coisa que você precisa saber é que commit não é a mesma coisa que salvar, embora depois que o commit serve justamente pra isso, salvar nossas modificações no projeto. A diferença é que salvar é uma ação que você vai fazendo sempre que você lembra não é... tipo um documento de texto ou planilha. Você vai salvando, quanto mais melhor... tem até o recurso de salvar automático não tem? O commit não é assim, você tem que commitar quando você fizer um conjunto de mudanças que faça algum sentido. Ou seja, não seria legal você comitar uma modurança do tipo "mudei a cor da página principal do site" e depois "mudei a logo", "mudei a fonte"... Faz mais sentido você commitar "mudei o tema do site". Mudar o tema envolve um conjunto de modificações que juntas fazem um sentido. Guarda isso com carinho, você vai agradecer seu professores depois por isso...

Que que acontece se a gente der um commit agora? Vamos ver?

Pra efeitos deste exercício não vou mudar fazer 30 modificações pra depois fazer um commit. Vou commitar pouca coisa pra facilitar com o a elaboração do material, mas guarde esse conceito no seu coração. U+2665 2665

É recomendado uma assistida de novo ao vídeo que fala do versionamento do google drive, do dropbox e compara com o versionamento manual do git. Lembra desse vídeo?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git commit -m "primeiro commit pra ver o que acontece"
```

Na saída do nosso commit o git nos indica algumas coisas... master -> calma mais tarde a gente fala sobre isso. (root-commit -> idem c8a1ee6 -> é um id pro commit. Isso é importante. Os commits são pequenas versões que você vai commitando.. pode ser que você queira voltar seu código pra um commit anterior e pra isso você vai precisar do id do commit que você quiser voltar. Mas não precisa anotar não, fica no banco de dados do git. 1 file changed, 8 insertions(+) -> um arquivo modificado com 8 inserções. Lembra do sinalzinho de mais pra indicar acréscimos lá do diff -u? create mode 100644 index.html -> não me lembro agora...

```
[master (root-commit) c8a1ee6] primeiro commit pra ver o que acontece
1 file changed, 8 insertions(+)
create mode 100644 index.html
```

Já o comando de status agora vai mostrar o seguinte...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git status
```

Nada pra ser commitado. "working directory clean" significa que o diretório de trabalho (nossa pasta) está limpa. Não tem nada pra commitar.

```
On branch master
nothing to commit, working directory clean
```

Item 7 **Editando o arquivo html** Abra o arquivo novamente coloque um H1 com título e um parágrafo com a descrição do site

Pra não ficar muito complicado vou fazer isso com um editor de texto mesmo, conforme demonstra a figura abaixo...

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
  <h1>Um Titulo</h1>
  <p>Este arquivo index.html faz parte da resolucao do exercicio2
  sobre git</p>
</body>
</html>
```

Figure 2: Arquivo index.html modificado com tag H1 e p

Item 8 **Usando o comando Commit** Commite com a msg (Titulo e Descrição do site)

Já fizemos as modificações que o item 7 pediu, agora o item quer que a gente commite.

Vamos fazer isso, mas vamos dar uma examinada no git status?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git status
```

Você percebe que o status do nosso index.html agora é modified? O git percebeu que a gente fez mudanças no nosso arquivo.

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Bom vamos então fazer o que o item 8 nos pede, vamos commitar. Vou colocar opção -a mas não vou entrar em detalhes agora. Pra efeitos do nosso tutorial pode usar -a à vontade...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git commit -a -m "Adicionei tag H1 e Tag p com descricao do arquivo"
```

Essa é a saída do nosso commit.

```
[master f26f3e8] Adicionei tag H1 e Tag p com descricao do arquivo
1 file changed, 3 insertions(+)
```

E verificando o status de novo...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
```

```
git status
```

Após o commit, não tem nenhuma modificação pra commitar.

```
On branch master
nothing to commit, working directory clean
```

Este item 8 do exercício fala pra gente usar o git log e o git diff pra acompanhar as modificações que foram feitas no nosso projeto. Vamos fazer isso então...

Primeiro vamos conhecer o git log.

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git log
```

O Resultado do nosso git log é justamente o log das modificações realizadas mostrando exatamente em quais commits elas ocorreram, bem como quem fez a alteração.

```
commit f26f3e8467f762bf6fafb8db8cbf0cad403b0f2d
Author: wagnermarques <wagnerdocri@gmail.com>
Date: Tue Jan 3 17:28:40 2017 -0200

    Adicionei tag H1 e Tag p com descricao do arquivo

commit c8a1ee69166d500bc66f5b4f8c2eece8e5514c77
Author: wagnermarques <wagnerdocri@gmail.com>
Date: Tue Jan 3 17:00:57 2017 -0200

    primeiro commit pra ver o que acontece
```

Agora vamos ver a saída do git diff.

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git diff
```

Saída nenhuma. Mas se a gente identificar os commits que queremos ver as modificações...

Vamos ver então, vou passar pro git diff os dois commits...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git diff c8a1ee69166d500bc66f5b4f8c2eece8e5514c77 f26f3e8467f762bf6fafb8db8cbf0cad403b0f2d
```

Agora sim, o resultado que a gente tava querendo... Por que o git diff não funcionou? Porque o git diff sem parametro nenhum mostra as alterações que você fez e que você não comitou ainda. Como a gente tinha comitado, não tinha nada pra mostra. Mas identificando os commit é interessante porque os commits, como a gente tinha falado anteriormente, tem significados e pelo diff a gente vê esse significado baseado nas alterações realizadas.

```
diff --git a/index.html b/index.html
index 37d904d..79c4cae 100644
--- a/index.html
+++ b/index.html
@@ -4,5 +4,8 @@
<title>Page Title</title>
</head>
```

```
<body>
+ <h1>Um Titulo</h1>
+ <p>Este arquivo index.html faz parte da resolucao do exercicio2
+ sobre git</p>
</body>
</html>
```

Não vou interpretar com detalhes porque é a mesma coisa que a gente fez no exercício anterior, ok? Mas qualquer dúvida pode perguntar ao seu professor.

Aproveitando pra usar a variação de git log sugerida...

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git log --oneline -2
```

É legal porque quando você tiver bastantes commits talvez você queira vê-los assim, um em cada linha.

```
f26f3e8 Adicionei tag H1 e Tag p com descricao do arquivo
c8a1ee6 primeiro commit pra ver o que acontece
```

Item9 Refletindo sobre Tralhar off line Interprete a saída do git log Lembra daquele menina na floresta dizendo que estava offline. Perceba que você não precisou de internet pra fazer isso certo? Ou seja, tudo isso que você está fazendo é offline.

Bom o item pede pra refletir. Realmente não foi necessário internet pra nada. Todo o trabalho foi feito offline

Na verdade o git tem muitos recursos a mais pra trabalhar offline, ou seja, não precisa de github pra usar git, nem de qualquer outra site parecido como bitbunch etc..

Você instala e usa. E dá até pra compartilhar código também porque existem comando pra vc exportar seu projeto pra poder passar pra um amigo e depois de os dois trabalharem dá pra fazer um merge de tudo tranquilamente, sem precisar de internet.

Os sites tipo o github tem suas vantagens porque tem um propósito de rede social de código. Pra isso precisa de internet mesmo..

item10 Reflexão entre commit manual e automático Lembra que as meninas do curso explicaram que o wikipedia, o dropbox o google drive também fazem controles de versão de um documento? Só que os commits, ou seja, o registro das mudanças não é manual, é automática, sendo registrado sempre que você salva por exemplo. Já com o git, o commit é manual? Quais são as vantagens e desvantagens de commits manuais? Ah. faça uma reflexão também sobre qual seria o melhor momento para commitar alterações dizendo o porque de sua conclusão sobre o assunto.

Bom aqui você precisa lembrar dos vídeos. Mas o básico é que com os commits manuais você dá significados às suas mudanças enquanto que commits automáticos (que é o versionamento de arquivos do google drive, dropbox, wikipedia e outros) não dá pra relacionar significados específicos para as versões.

Item11

1. **Trabalhando com mais confiança** Já aconteceu com você de você fazer uma alteração em algum arquivo ou projeto e depois não saber muito bem onde você errou e decidir dar um control z pra voltar ao que era antes e começar de novo? Já aconteceu comigo de querer voltar pra uma versão de ontem, por exemplo, já que o control z não estava disponível. Como essa o git facilita a nossa vida diante de situações como essa?

Com git você trabalha com mais confiança com certeza porque você pode perceber que fez algo de

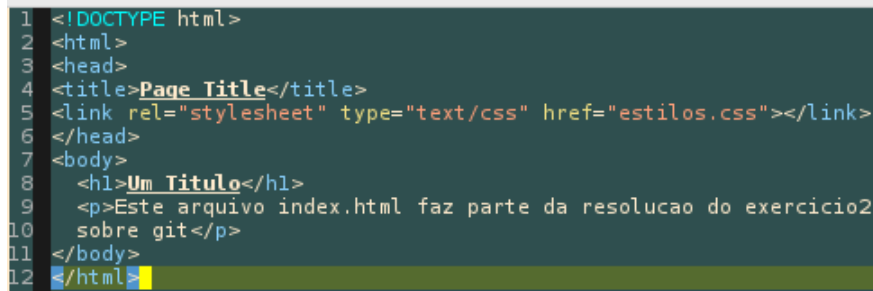
muito errado e não está conseguindo solucionar e pode ser que você queira voltar pra uma versão (commit) anterior e começar de novo. Outra grande segurança é que você pode usar o git diff pra ver quais mudanças você fez e que não deu certo. Sabe quando o código não tá rodando e você não lembra qual modificação você fez pós o momento que ele rodava? usa o diff que fica facinho e com certeza não vai passar nenhuma modificação fora de seus olhos...

Item 12 **Trabalhando com múltiplos arquivos** Claro que um css deve sempre que possível ser escrito num arquivo a parte. Faça isso com seu estilo e linke ele no seu html. 12 a) De um git status antes de commitar 12 b) Commite 12 c) De um git status de novo

Bom vamos pra solução... Vou criar os arquivos com comandos tá?

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
touch estilos.css
echo "body { background-color : yellow }" > estilos.css
ls -l
git status
```

Vou linka-lo no index.html via editor de texto, conforme mostra a figura abaixo.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Page Title</title>
5 <link rel="stylesheet" type="text/css" href="estilos.css"></link>
6 </head>
7 <body>
8 <h1>Um Titulo</h1>
9 <p>Este arquivo index.html faz parte da resolucao do exercicio2
10 sobre git</p>
11 </body>
12 </html>
```

Figure 3: Linkando arquivo css no html

Após a criação do arquivo E após linka o css no index.html Temos a seguinte saída... Perceba que o arquivo estilos.css foi criado e que está untracked Considerando que agente teve que alterar o index.html com a tag link, o status do index.html agora é "modified"

```
total 8
-rw-rw-r--. 1 wagner wagner 35 Jan  3 18:15 estilos.css
-rw-rw-r--. 1 wagner wagner 256 Jan  3 18:14 index.html
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        estilos.css

no changes added to commit (use "git add" and/or "git commit -a")
```

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
```



```
git add estilos.css
git commit -a -m "adicionado estilo ao index.html"
```

```
[master 7ffb730] adicionado estilo ao index.html
2 files changed, 2 insertions(+)
create mode 100644 estilos.css
```

```
cd /run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
git status
```

```
On branch master
nothing to commit, working directory clean
```

Beleza? Se tem um assunto que vai a pena estudar e que vai te recompensar no futuro.. esse assunto é o git... Onde você for trabalhar certamente usa... Seus professores estão à disposição, não perca a oportunidade! n

11.3 Exercício 3 -> Vendo o histórico, só que agora não do seu projeto.

1. **Usando git clone** Faça o clone do seguinte repositório <https://github.com/udacity/asteroids.git>

Contents:

- [11.1 Exercício 2.1 -> Não simplesmente ver uma diferença, é legal ver o histórico](#)
- [11.2 Resolução do Exercício 2.1 -> Não simplesmente ver uma diferença, é legal ver o histórico](#)
- [11.3 Exercício 3 -> Vendo o histórico, só que agora não do seu projeto.](#)

12 3 Dicas do dia a dia (from: <https://github.com/git-tips/tips>)

12.1 git diff revisitado

Trabalhei ontem no projeto e não me lembro muito bem quais modificações fiz ontem. git diff

12.2 Quais arquivos eu já tinha até um certo commit específico?

```
git diff-tree --no-commit-id --name-only -r <ommit-ish>
```

12.3 Quem modificou esses arquivos?

12.4 Quais modificações foram feitas num arquivo específico?

12.5 Tem arquivos em conflito?

```
git diff --name-only --diff-filter=U
```

Contents:

- [12.1 git diff revisitado](#)
- [12.2 Quais arquivos eu já tinha até um certo commit específico?](#)
- [12.3 Quem modificou esses arquivos?](#)
- [12.4 Quais modificações foram feitas num arquivo específico?](#)
- [12.5 Tem arquivos em conflito?](#)

13 4 Branches

13.1 Listando branches

git branch (lista branches locais) git branch -r (lista branches remotos)

13.2 Merging

Mergin é a mesclagem de um outro brach no seu branch de trabalho corrente. Você tá trazendo mudanças de um outro contexto (outro brach) pro seu contexto de trabalho que é o seu branch corrente combinando os arquivos que você está usando atualmente no seu branch de trabalho.¹

13.2.1 Resolvendo Conflitos

Ocorre qua se em um contexto que dois programadores mudaram um mesmo arquivo onde um modificou uma linha e o outro alterou essa mesma linha. Quem está certo? O git pode decidir, então ele marca o arquivo como estando em conflito. Pra continuar o desenvolvimento do projeto precisa resolver esse conflito.

O primeiro passo é entender o que aconteceu. Pra verificar de um git status

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev'
echo $(pwd)
git status
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-de
On branch master
Your branch and 'remotes/origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
(fix conflicts and run "git commit")

Changes to be committed:

  modified:   autocomplete_config.el
  modified:   cedet_config.el
  new file:   config-enviroment.el
  modified:   config_package_system.el
  new file:   eclim_config.el
  new file:   eclim_starter.sh
  modified:   find_files.el
  new file:   fzl_customization_functions.el
  new file:   init.bk.el
  new file:   magit_installation.el
  new file:   test.php

Unmerged paths:
(use "git add <file>..." to mark resolution)

  both added:   dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifact
  both modified: emacsinitfile.log
  both modified: init.el
  both modified: org_mode_config.el
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/sou
On branch master
Your branch and 'remotes/origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
```

```
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
(fix conflicts and run "git commit")

Changes to be committed:

    modified:   autocomplete_config.el
    modified:   cedet_config.el
    new file:   config-enviroment.el
    modified:   config_package_system.el
    new file:   eclim_config.el
    new file:   eclim_starter.sh
    modified:   find_files.el
    new file:   fzl_customization_functions.el
    new file:   init.bk.el
    new file:   magit_installation.el
    new file:   test.php

Unmerged paths:
(use "git add <file>..." to mark resolution)

    both added:   dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifacts/org
    both modified: emacsinitfile.log
    both modified: init.el
    both modified: org_mode_config.el
```

Temos uma situação de conflito que envolve quatro arquivos, conforme acima.

Vamos ver o que aconteceu com o arquivo init.el

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
echo $(pwd)
git diff --name-only --diff-filter=U
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifacts/org_mode_SETUPFILE_Expo
emacsinitfile.log
init.el
org_mode_config.el
```

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
echo $(pwd)
git diff init.el
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/
diff --cc init.el
index c887a6f,d7ab670..0000000
--- a/init.el
+++ b/init.el
@@@ -1,16 -1,26 +1,35 @@@
  (setq debug-on-error t)
  +
  (require 'cl)

++<<<<<<< HEAD
++=====
+ ;(setq url-proxy-services '(("no_proxy" . "work\\.com")
+ ;                          ("http_proxy" . "wagner:nicolas1*@192.168.0.2:3128")
+ ;                          ("ftp_proxy" . "wagner:nicolas1*@192.168.0.2:3128")
+ ;                          ("all_proxy" . "wagner:nicolas1*@192.168.0.2:3128"))
+ ;
+ ;from https://www.reddit.com/r/emacs/comments/3inht4/emacs_25_on_windows_behind_a
```

```

+ ;(setq url-proxy-services
+ ;      '(("no_proxy" . "^\\(localhost\\|192.168.0.*\\)")
+ ;        ("http" . "192.168.0.2:3128")
+ ;        ("https" . "192.168.0.2:3128")))
+ ;(setq url-http-proxy-basic-auth-storage
+ ;      (list (list "192.168.0.2:3128"
+ ;                (cons "Input your LDAP UID !"
+ ;                      (base64-encode-string "wagner:nicolas1*")))))
+
++>>>>>> 28e5f69afa0c7bb80e00b0beeec6afbb6ce8b7b0

- (setq **HOME** (concat (concat "/home/" (getenv "USER")) "/"))
+;;(setq url-proxy-services '(("no_proxy" . "work\\.com")
+;;      ("http_proxy" . "wagner:nicolas1*@192.168.0.2:3128")
+;;      ("ftp_proxy" . "wagner:nicolas1*@192.168.0.2:3128")
+;;      ("all_proxy" . "wagner:nicolas1*@192.168.0.2:3128"))

+ (setq **HOME** (concat (concat "/home/" (getenv "USER")) "/"))
+

  (defun fzl_print_global_variables()
    "print defined variables and its values"
    @@@ -47,14 -56,11 +66,21 @@@

+
+;;DEFINE THIS ENVIRONMENTAL VARIABLE
++<<<<<<< HEAD
+;;THIS JUST REPRESENTS A DIRECTORY WHERE SHOULD BE YOU DEVTOOLS WHERE ARE YOURS I
+ (if (not (and (getenv "FZL_HOME") (getenv "FZL_HOME_SERVER")))
+
+   ;;in case FZL_HOME was not defined...
+   (message "PLEASE EXPORT FZL_HOME AND FZL_HOME_SERVER ENVIRONMENT VARIABLES")
+
+   ++=====
+   ;;THIS JUST REPRESENTS A DIRECTORY WHERE SHOULD BE YOU DEVTOOLS IN ITS INTEGRATED
+   (if (not (and (getenv "FZL_HOME") (getenv "FZL_HOME_SERVER")))
+
+     ;;in case FZL_HOME was not defined...
+     (message "PLEASE EXPORT FZL_HOME AND FZL_HOME_SERVER ENVIRONMENT VARIABLES")
+
+     ++>>>>>> 28e5f69afa0c7bb80e00b0beeec6afbb6ce8b7b0

+     ;;in case FZL_HOME was defined...
+     (progn
+       @@@ -68,14 -74,13 +94,23 @@@
+       (progn
+         (setq **FZL_HOME** (file-name-as-directory **FZL_HOME_provided**))
+         (setq **FZL_HOME_SERVER** (file-name-as-directory **FZL_HOME_SERVER_pro
++<<<<<<< HEAD
+   (error "Sorry the $FZL_HOME and $FZL_HOME_SERVER must be accessible"))
++=====
+   (error "Sorry the $FZL_HOME and $FZL_HOME_SERVER must be accecible"))
++>>>>>> 28e5f69afa0c7bb80e00b0beeec6afbb6ce8b7b0

      (setq **DEV_TOOLS_BASEDIR** (concat **FZL_HOME** "integrated/"))
      (message **DEV_TOOLS_BASEDIR**)
++<<<<<<< HEAD
+
+   ;;CHANGE DEV TOOLS ACCORDINGLY
+   (setq **M2_HOME** (concat **DEV_TOOLS_BASEDIR** "build/apache-maven-3.3.3
++=====
+   ;;CHANGE DEV TOOLS ACCORDINGLY
+   (setq **M2_HOME** (concat **DEV_TOOLS_BASEDIR** "build/apache-maven-3.3.9
++>>>>>> 28e5f69afa0c7bb80e00b0beeec6afbb6ce8b7b0
+   (setq **NEXUS_HOME** (concat **DEV_TOOLS_BASEDIR** "build/nexus-3.0.1-01/"))

+   (setq **JAVA_HOME** (concat **DEV_TOOLS_BASEDIR** "jdk/jdk1.8.0_65/"))

```

Bom pra entender qual é o conflito a gente deu um git diff init.el. O git indica pra gente qual é a região de conflito através como sendo tudo que esta dentro da região marcada com

```
++<<<<<<< HEAD LINHAS DO ARQUIVO DENTRO DA REGIAO DE CONFLITO ++>>>>>>>
28e5f69afa0c7bb80e00b0beec6afb6ce8b7b0
```

Então agora é só decidir como deve ficar o arquivo final editando a região de conflito e deixando como deve ficar o resultado final e depois que fizer isso em todas as regiões de conflito, a gente tem dizer pro git que já nos decidimos e resolvemos os conflitos usando o comando git add <nome-do-arquivo>.

Vamos lá fazer isso então? Bom um dos trechos em conflito está abaixo. Vou editar essa região de conflito como exemplo pra gente acompanhar e vou editar as outras depois também pra solucionar todos os conflitos. Por último a gente dá um git add init.el

Eis a região de conflito que vamos resolver juntos...

```
<<<<<<< HEAD
;;THIS JUST REPRESENTS A DIRECTORY WHERE SHOULD BE YOU DEVTOOLS WHERE ARE YOURS INTEGRATED DIR
(if (not (and (getenv "FZL_HOME") (getenv "FZL_HOME_SERVER"))))

;;in case FZL_HOME was not defined...
(message "PLEASE EXPORT FZL_HOME AND FZL_HOME_SERVER ENVIRONMENT VARIABLES")

=====
;;THIS JUST REPRESENTS A DIRECTORY WHERE SHOULD BE YOU DEVTOOLS IN ITS INTEGRATED DIR
(if (not (and (getenv "FZL_HOME") (getenv "FZL_HOME_SERVER"))))
;;in case FZL_HOME was not defined...
(message "PLEASE EXPORT FZL_HOME AND FZL_HOME_SERVER ENVIRONMENT VARIABLES")
>>>>>>> 28e5f69afa0c7bb80e00b0beec6afb6ce8b7b0
```

```
;;THIS JUST REPRESENTS A DIRECTORY WHERE SHOULD BE YOU DEVTOOLS WHERE ARE YOURS INTEGRATED DIR
(if (not (and (getenv "FZL_HOME") (getenv "FZL_HOME_SERVER"))))

;;in case FZL_HOME was not defined...
(message "PLEASE EXPORT FZL_HOME AND FZL_HOME_SERVER ENVIRONMENT VARIABLES")
```

Não sei se vc percebeu, mas era uma questão só de estilo de codificação, neste caso. ou seja, tinha uma linha onde foi inserido um espaço o que modificou as linhas abaixo dela e gerou, portanto o conflito. Esse foi um exemplo bastante simplório porque é só pra ilustrar uma situação de como resolver conflitos com git, mas pode ser que o conflito envolvesse lógica do software e aí precisaria então uma atenção especial na solução do problema. Na prática, entretanto, para efeitos do git apenas, a solução do conflito envolve só editar a região de conflito e depois dar um git add nomeDoArquivo. Só isso.

Editei as regiões de conflito do arquivo init.el, mas ainda não dei um git add init.el

Antes ainda de dar um git add init.el, vamos averiguar se tem conflito ainda?

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
echo $(pwd)
git diff --name-only --diff-filter=U
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-de
dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifacts/org_mode_SETUPFILE_Expo
emacsinitfile.log
init.el
org_mode_config.el
```

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev'
echo $(pwd)
git status
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
On branch master
Your branch and 'remotes/origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
(fix conflicts and run "git commit")
```

Changes to be committed:

```
modified:   autocomplete_config.el
modified:   cedet_config.el
new file:   config-environment.el
modified:   config_package_system.el
new file:   eclim_config.el
new file:   eclim_starter.sh
modified:   find_files.el
new file:   fzl_customization_functions.el
new file:   init.bk.el
new file:   magit_installation.el
new file:   test.php
```

Unmerged paths:

(use "git add <file>..." to mark resolution)

```
both added:   dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifact
both modified: emacsinitfile.log
both modified: init.el
both modified: org_mode_config.el
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev/sou
On branch master
Your branch and 'remotes/origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
(fix conflicts and run "git commit")
```

Changes to be committed:

```
modified:   autocomplete_config.el
modified:   cedet_config.el
new file:   config-environment.el
modified:   config_package_system.el
new file:   eclim_config.el
new file:   eclim_starter.sh
modified:   find_files.el
new file:   fzl_customization_functions.el
new file:   init.bk.el
new file:   magit_installation.el
new file:   test.php
```

Unmerged paths:

(use "git add <file>..." to mark resolution)

```
both added:   dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifacts/org
both modified: emacsinitfile.log
both modified: init.el
both modified: org_mode_config.el
```

Percebe-se que, apesar de eu ter editado as regiões de conflito o git ainda não foi avisado que eu já fiz isso

e que não tem mais conflito. Vamos avisá-lo então...

```
git add init.el
```

Agora que o git já está avisado da resolução do conflito no arquivo init.el, vamos dar um git status de novo?

```
cd '/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev'
echo $(pwd)
git status
```

```
/run/media/wagner/96fea5f1-d297-4f63-a035-abf6511467be/wagnerdocri@gmail.com2/envs/env-dev
On branch master
Your branch and 'remotes/origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
(fix conflicts and run "git commit")

Changes to be committed:
  modified:   autocomplete_config.el
  modified:   cedet_config.el
  new file:   config-environment.el
  modified:   config_package_system.el
  new file:   eclim_config.el
  new file:   eclim_starter.sh
  modified:   find_files.el
  new file:   fzl_customization_functions.el
  new file:   init.bk.el
  modified:   init.el
  new file:   magit_installation.el
  new file:   test.php

Unmerged paths:
(use "git add <file>..." to mark resolution)

  both added:    dir_for_org_mode_tutorials_artifacts/org-mode-tutorials-artifact
  both modified: emacsinitfile.log
  both modified: org_mode_config.el
```

Dá pra perceber que o arquivo init.el não está sendo mais listado no unmerged paths, o que mostra que agora o git tá resolvido com o conflito relacionado a este arquivo. Então é só repetir para os demais.

Depois de resolver todos os conflitos é possível fazer um git push

```
[wagner@emulti2 emacsinitfile]$ git push
warning: push.default is unset; its implicit value has changed in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the traditional behavior, use:

git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
```

```
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

error: unable to read askpass response from '/usr/libexec/openssh/gnome-ssh-askpass'
Username for 'https://github.com': wagnermarques
error: unable to read askpass response from '/usr/libexec/openssh/gnome-ssh-askpass'
Password for 'https://wagnermarques@github.com':
Counting objects: 19, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (19/19), done.
Writing objects: 100% (19/19), 2.30 KiB | 0 bytes/s, done.
Total 19 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), completed with 9 local objects.
To https://github.com/wagnermarques/emacsinitfile.git
 28e5f69..111e2e5 master -> master
[wagner@emulti2 emacsinitfile]$
```

Contents:

- [13.2.1 Resolvendo Conflitos](#)

Contents:

- [13.1 Listando branches](#)
- [13.2 Merging](#)

Footnotes:

Author: Wagner & Jeferson

Created: 2017-01-03 Ter 18:36

[Validate](#)