Aprendendo Git. Para isso, execute os passos listados abaixo.

1. No terminal (ou **Git Bash**, no Windows) navegue até a pasta recém criada (utilize o comando cd para navegar entre pastas);
2. Execute o comando git init para inicializar o git

3) Execute o comando git add index.html para marcar o arquivo para ser salvo (*commitado*);

4) Execute git status e confira que o arquivo agora mudou de estado e está pronto para ser salvo (*commitado*);

5) Após adicionar, execute o comando git commit -m "Criando arquivo index.html com lista de cursos", ou git commit-am onde este comando adiciona o comentário para todos os arquivo e pastas que vc adiciou. Sinta-se à vontade para alterar a mensagem de *commit*, se desejar;

6) Altere o arquivo **index.html**. Adicione o acento em "Integração Continua", por exemplo;

7) Adicione o arquivo para ser salvo, com git add .;

8) Execute o comando git commit -m "Acento adicionado no curso de Integração Contínua". Sinta-se à vontade para alterar a mensagem de *commit*, se desejar. O commit é o comentário que foi feito no projeto.

8) Execute o comando git log e analise a sua saída. Execute também git log --oneline, git log -p e outras alternativas que quiser testar;

9) Crie um arquivo vazio com o nome que quiser, por exemplo, ide-config;

9) Crie o arquivo **.gitignore** e adicione uma linha com o nome do arquivo recém-criado (ide-config, no exemplo acima);

10) Execute git status e verifique que o arquivo **ide-config** não está na lista para ser adicionado;

11) Adicione (com git add .gitignore) e realize o commit (com git commit -m "Adicionando .gitignore") o arquivo **.gitignore**.

Diversas formas de como posso usar o log

<https://devhints.io/git-log>

Para criar uma pasta o comando é “mkdir”

Repositório remoto - Para criarmos um repositório remoto – Usamos git remote add [nome-Desejado] + local.

Git remote -m – serve para consultar visualizar o acessos remotos adicionados

Push - Adicionar nossa atualização do projeto, usamos git push master.

Pull – Para puchar a última atualização no git.

Branch

Trabalhado com banch – Ela serve para realizarmos trabalhos compartilhados, onde um determinado usuário deseja trabalhar em uma determinada parte criamos o bransh. Para navegamos entre branch que criamos usamos o comando git checkout [Nome – Exemplos: master] e para adicionarmos as atualizações na branch principal e unirmos os projetos entramos na principal e digitamos - git merge [Nome da branch da qual estávamos trabalhando, exemplo – titulo] ou git rebase [Nome da branch da qual estávamos trabalhando].

Passo à passo em como podemos trabalhar com branch

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Execute o comando git branch e veja que apenas a *branch* master existe no seu repositório;

2) Execute o comando git branch titulo e logo após execute o comando git branch. Veja que uma nova *branch* foi criada;

3) Agora, para começar a trabalhar nesta *branch*, digite git checkout titulo;

4) Execute novamente git branch e confira que agora você está na *branch* chamado titulo;

5) Altere o título da página **index.html** para "Cursos de DevOps da Alura";

6) Adicione as alterações com git add index.html;

7) Faça o *commit*, com git commit -m "Alterando título da página";

8) Execute o comando git log e confira -= o novo *commit*;

9) Altere o título da página para "Lista de cursos de DevOps da Alura";

10) Repita os passos 6 e 7, para adicionar um novo *commit*, alterando a mensagem;

11) Repita o passo 8 para conferir o novo *commit*;

12) Execute o comando git checkout master para voltar à linha de desenvolvimento master;

13) Execute git log para conferir que os últimos dois *commits* não estão lá. Confira se o conteúdo do seu arquivo também voltou ao seu estado original;

14) Na pasta criada para representar o trabalho de outra pessoa na aula anterior:

* Execute git checkout -b lista para criar uma nova *branch*, chamada lista e passar a trabalhar nela;
* Adicione o curso de "Kubernetes" na lista;
* Repita os passos 6 e 7 para adicionar um novo *commit*, alterando a mensagem;
* Execute o comando git checkout master para voltar à linha de desenvolvimento master;

15) Volte para a pasta que representa o seu próprio trabalho;

16) Altere o nome do curso de Docker para "Docker: Criando containers sem dor de cabeça";

17) Repita os passos 6 e 7 para adicionar um novo *commit*, alterando a mensagem;

18) Execute o comando git merge titulo para trazer o trabalho feito na *branch* titulo para a *branch* master;

19) Execute o comando git log --graph para ver as linhas de desenvolvimento (*branches*);

20) Execute git checkout titulo para trabalhar na *branch* chamada titulo;

21) Altere o título para ter a palavra "Cursos" com letra maiúscula;

22) Repita os passos 6 e 7 para adicionar um novo *commit*, alterando a mensagem;

23) Execute o comando git checkout master para voltar à linha de desenvolvimento master;

24) Execute o comando git rebase titulo;

25) Execute o comando git log e confira que o *commit* foi adicionado antes do *commit* realizado diretamente na *branch* master;

26) Execute o comando git push local master para enviar suas alterações para o repositório remoto que criamos na última aula;

27) Na pasta criada para representar o trabalho de outra pessoa na aula anterior:

* Execute o comando git pull local master para baixar as alterações que você já realizou;
* Execute o comando git checkout lista para continuar trabalhando na lista de cursos;
* Altere o nome do curso de Docker para "Curso de Docker: Criando containers sem dor de cabeça";
* Repita os passos 6 e 7 para adicionar um novo *commit*, alterando a mensagem;
* Execute o comando git checkout master para voltar à linha de desenvolvimento master;
* Tente juntar seu trabalho com git merge lista;
* Veja que há conflitos. Corrija-os, deixando apenas a linha com o nome correto do curso;
* Execute o comando git add index.html para informar que os conflitos neste arquivo foram corrigidos;
* Execute o comando git commit para que o Git finalize o *merge*;
* Execute o comando git push local master para enviar as suas alterações;

28) Volte para a pasta que representa o seu próprio trabalho;

29) Altere o nome do curso de Vagrant para "Vagrant: Gerenciando máquinas virtuais";

30) Repita os passos 6 e 7 para adicionar um novo *commit*, alterando a mensagem;

31) Tente executar o comando git push local master. Veja a falha;

32) Execute o comando git pull local master para trazer as alterações da outra pessoa;

33) Agora sim, execute o comando git push local master para enviar as alterações.