GIT – Tutorial

Criando um repositório de código

Inicie um projeto criando um repositório de código no diretório atual:

git init

Configure o nome do usuário para o repositório atual:

git config --local user.name SeuNome

Configure o email do usuário para o repositório atual:

git config --local user.email SeuEmail

Veja o status:

git status

Crie um arquivo README e adicione todos os arquivos ao repositório:

git add .

Coloque uma mensagem de commit:

git commit

Ou

git commit -m "mensagem"

Veja o histórico do repositório:

git log --oneline

Edite alguma coisa e adicione as mudanças ao repositório:

git add .

git commit

Crie uma nova vertente do código:

git branch nomeDaVertente

Mude para esta vertente:

git checkout nomeDaVertente

Edite o arquivo README de alguma forma e adicione as mudanças ao repositório:

git add .

git commit

Compare a vertente atual com a vertente master (ela é a vertente que você estava anteriormente)

cat README

git checkout master

cat README

Viu? O mesmo arquivo contém diferentes dados dependendo da vertente!

Mescle as vertentes!

git checkout master

git merge nomeDaVertente

Colaborando com o projeto

Primeiro, em repositórios compartilhados, por padrão nós sempre deixamos a vertente master “livre”:

Liberando a vertente master

git checkout nomeDaVertente

Clonando o repositório:

coloque seu pendrive;

entre dentro dele pela linha de comando;

git clone /caminho/para/o/repositório/que/você/quer/clonar

cd nomeDaPastaCriadaPeloClone

Indo para o vertente master

Lembrando que é sempre na vertente master que a versão estável se encontra, ou seja, é a partir dela que você deve fazer suas modificações.

git checkout master

Criando e entrando numa vertente para fazermos nossa modificações

SEMPRE crie uma vertente para fazer modificações, modificar diretamente a vertente master pode dar porcaria!

git branch minhaVertenteDeModificações

git checkout minhaVertenteDeModificações

Agora faça as modificações que quiser sem medo!

Pronto? Agora mande suas modificações para o projeto!

Conecte seu pendrive de volta na máquina e entre nele pela linha de comando;

Entre na pasta que surgiu quando você clonou o repositório;

git push --set-upstream origin minhaVertenteDeModificações

Recebendo colaborações

Depois que te enviaram as vertentes faça o seguinte:

Vá para a vertente master

git checkout master

Veja as vertentes que existem no repositório:

git branch

Veja o que foi modificado em uma vertente

git diff master minhaVertenteDeModificações

OU

git difftool master minhaVertenteDeModificações

Aceite as modificações!

git merge minhaVertenteDeModificações

Libere a vertente master

git checkout nomeDeAlgumaVertenteDisponível

GIT – Folha de comandos

git init

Cria um repositório de código no diretório atual

git config --local user.name SeuNome

Configura o nome do usuário para o repositório atual

git config --local user.email SeuEmail

Configura o e-mail do usuário para o repositório atual

git status

Mostra o status atual do repositório

git add caminhoDoArquivo

Adiciona um arquivo ao repositório

git rm caminhoDoArquivo

Remove um arquivo ao repositório

git commit

Concretiza as mudanças (edição, criação ou deleção) e pede uma descrição do que foi feito. Obs: Um editor de texto se abrirá, não se esqueça de salvar a descrição, do contrário, a mudança será ignorada.

git log --oneline

Visualiza todos os commits dados

git branch nomeDaRamificação

Cria uma nova ramificação do código, assim, se pode alterar coisa sem medo de ferrar o código original, mas não sem antes dar o próximo comando.

git checkout nomeDaRamificação

Vai para a ramificação criada, agora sim você pode alterar o código! ATENÇÃO: Se você estiver trabalhando em um time, jamais dê commits na vertente principal do código (a master), sempre crie uma ramificação e lá ponha suas alterações.

git checkout master

Volta para a vertente principal do código.

git merge nomeDaRamificação

Pega todas as modificações feitas em “nomeDaRamificação” e transfere para a vertente principal do código, isso pode gerar alguns conflitos que devem ser resolvidos manualmente.

git reset --hard algumIdDeUmCommit

Volta o projeto para a versão especificada pelo id do commit (cuidado!!!)

git revert --no-edit idDoÚltimoComit idDoCommitASerExcluido

Define um intervalo de commits que devem ser “desfeitos”.