***Ciclo de vida dos Status de arquivos no GIt***

***Como funciona o Git***

O git e um gerenciador de versão do código o git possui recursos avançados de monitoramento e acompanhar o arquivo.

Após ativar em um projeto o GIT começa a monitorar as mudança ocorridas neste projeto e nos mostra através de um poderoso sistema de estágios se e possível encontrar o arquivos.

***Comando básicos***

Para renomear um arquivo ou repositório usamos o comando ***mv*** e usado para renomear caso não faça e vai fazer isso manualmente vai ter que adicionar novamente como mostrado abaixo.

***Git mv nome\_projeto\_original nome\_novo***

Como criar uma nova pasta usamos o comando **mkdir** como mostrado abaixo:

***Mkdir nome\_pasta***

Para deletar para um arquivo e repositórios usamos o comando ***rm***  e o comando para remover um arquivo mas caso queira deletar o arquivo usando o outro jeito sem usar o git vai ter que usar o comando para adicionar como mostrado abaixo.

***Rm nome\_arquivos***

Para ver as modificação usamos o comando  ***diff –staged*** era usado para comparar os arquivos como mostrado abaixo:

***Git diff –staged***

Ele vai mostrar a alteração que teve nos arquivos.

Para sair das configuração anterioe usamos o comando ***git config –unset credentiam.helper***

Para voltar par a configuração usamos o comando ***git config cresentian .helper store.*** Depois usar o comando ***git remote set-url origin***.

***Como iniciar o projeto***

Primeiro e criar um ***repositório*** primeiro e criar uma pasta e depois usar o comando ***git init*** e um comando para falar que o git vai começar a observar oque acontece no codigo.

O comando ***git status*** ele e usado para verificar o estado atual do repositório depois ele vai mostrar oque foi modificado e ainda não foram adicionado oque não estar sendo rastreado pelo ***git.***

***Como usar o .gitignore***

Esse arquivo e usado para ignora outro arquivos ou ate diretórios para fazer para ignora os arquivo ou repositório primeiro criamos um arquivo com o nome de ***.gitignore*** e dentro dele e colocado os arquivo que vai ser ignorado.

Para ignora um ou mais arquivos.

Usamos o nome do arquivo para ignorar como mostrado abaixo.

***Nome.java***

E pode colocar o repositórios primeiro colocar o nome do repositório.

***Configurando o git***

Para configurar o git colocando o nome e o gmail e a primeira coisa que será feito quando cria um repositório para colocar o nome e o gmail usamos o comando abaixo:

Para configurar o nome do usuário usamos o comando ***user.name*** e para configurar o gmail usamos o comando ***user.gmail*** para configurar usamos o comando ***git config*** como mostrado abaixo:

***Git config user.name “jhefferson”***

***Para adicionar um novo arquivo***

Para adiciona um novo arquivo usamos o comando ***git add .*** dai o nome do arquivo mas para adicionar todos os arquivos usamos o comando ***git add .*** . como mostrado abaixo.

***Git add .***

***Commit***

Quando usamos o coando ***commit*** ele vai submeter um arquivo usando o comando ***git commit –m*** que e usado para deixa o arquivo pronto para mandar para o repositório como mostrado abaixo:

***Git commit –m “primeiro commit “***

Era usado para deixa pronta para mandar para mandar para o repositório.

Mas quando fazemos um ***commit*** e feito uma foto do projeto como esta o projeto na hora do ***commit .***

Quando fizemos um commit agente cria um novo nó que e possível acessar usando o comando ***checkout***

***Consultando o histórico***

Quando queremos ver os histórico dos últimos ***commit*** usamos o comando ***log***  como mostrado abaixo:

***Git log***

Ele vai mostrar todos os commit feitos mas caso ele esta menor que o espaço ele vai mostrar os commit ate preencher dai ele vai aparecer dois pontos dai e possível afazer uma pesquisa usando o ( / ) vai colocar uma pesquisa das mensagem só as palavras.

E para sair da pesquisa usamos a letra ( q ) para sair.

Caso queira pegar os em um local especifico usamos os ( - ) dai o numero dai ele vai pegar os últimos feito como mostrado abaixo:

***Git log -5***

Ele vai pegar os últimos 5 commit feitos e pode mudar para qualquer numero ele vai pegar o numero que quer.

E para ver o ultimo usamos -1.

E para pegar um resumo usamos o comando ***–oneline*** ele vai pegar todos o commit em uma linha só como mostrado abaixo:

***Git log –oneline***

e pode colocar mais de um comando juntos como mostrado abaixo:

***git log –oneline -3***

E possível pegar por data usando o comando ***–bifore*** como mostrado abaixo:

***Git log –bifore “2020-05-20”***

Ele vai pegar a data anterior a data especificada.

Mas caso queira pegar após usando o comando ***–after*** como mostrado abaixo:

***Git log –after “2025-02-20***

Para var para ver os últimos usamos o comado ***–since*** ele via pegar após um dia específicos.

E possível pesquisar por autor usando o coando ***–author*** como mostrado abaixo:

***Git log –author=”autor”***

Cause queira ver mais e possível usar o ***git log help*** e para sair usamos o q.

Voltando no tempo

Para voltar para um commit antigo usamos o comando ***checkout*** como mostrado abaixo.

***Git checkout 614316***

Ele vai para ultimo no que e o commit respectivo ao nó.

E para voltar ele vai ter que usar o comando ***git checkout master*** .

Caso queira voltar para o ultimo comit feito também da usando o comando.

Alterando um erro

Quando e feito um commit errado podemos usar o comando ***commit –amend –m “mensagem novo”*** como mostrado abaixo:

***commit –amend –m “novo commit”***

***tirando arquivo de estagio***

quando adicionamos um arquivo sem querer que seja acionado usamos o comando ***restore –stagem*** dai colocar o nome do arquivo que e usado para não adiciona o arquivo como mostrado abaixo.

***Git restore –stagem nome\_arquivo***

Caso presise voltar para antes de adicionar os arquivos podemos usar o comando ***reset HEAD –hard*** ele vai voltar para o ultimo commit feito como mostrado abaixo.

***Git reset HEAD –hard***

Mas para descartar o ultimo commit usamos o ( ^ ) como mostrado abaixo:

***Git reset HEAD^ –hard***

***Criando novos ramos do projeto***

Um projeto e igual uma arvorem temos o projeto principal isso se chama de ***branch master*** e possível criar um novo ramo ele e usado para quando quer adicionar uma nova funcionalidade sem atingir o ramo principal e bom para adicionar novas alteração sem destruir o ramo principal.

Para criar um novo ramo usamos o comado ***git branch nome\_ramo*** como no exemplo abaixo;

***Git branch fução***

E para ver todos as ***branch*** usamos o comando ***git branch.***

e para mudar para outra ***branch*** usamos o comando ***checkout nome\_branche*** como mostrado no exemplo abaixo.

***Git checkout***

E para adicionar os arquivos usamos o comando ***git add*** e dai pode usar o commit para adiciona a alteração.

E para deletar um branch usamos o comando ***git branch –d nome\_ramo*** dai ele vai dar uma mensagem para que esta com certeza para deletar essa nova branch dai pode usar o comando ***git branch –D nome\_ramo.***

***Fundindo os ramos***

Quando usamos o comando ***marge*** e usado para quando queremos unir um brange e quando queremos unir os branch e todos os arquivos que tem em uma branch mas não tem na brench master ele vai unir os arquivos arquivos gerament e feito depois de um commit .

Mas quando tem mais de uma pessoa mexendo pode dar problema porque tem duas pessoas fazendo marge.

***Git marger nome\_rmo***

Depois de fazer o marge não e mais preciso usar o comando ***git branch –D nome\_ramo*** e pode usar o comando ***git branch –d nome\_ramo*** para deletar o ramo.

Quando usamos o comando ***rbase*** ele desmonta tira os antigos commits temporariamente e encacha o ramo que foi colocado e depois recoloca o commit antigos.

***Clone repositório e push de um repositório***

O comando ***git clone*** ele e usado para clonar um repositório e para mandar para o repositório remoto para mandar para um repositório agente usa o comando ***git push*** caso tem um repositório mas tenta atualizar um repositório mas não e um repositório ele vai dar um erro mas mais na frente vou falar mais sobre.

Mas quando usamos o comando ***git fetch*** mas ele abaixa o arquivo mas sem usar um merge mas quando e usado esse comando ele geralmente e usado o comando ***git rbase*** mas pode usar o comando ***git pull*** ele vai usar o comando git rbase e o comando fetch ao mesmo tempo.

***Usando o comando base repository***

Quando usamos o comando ***git init –bere*** ele vai ser usado para centralizar os repositórios e para submeter usamos agora e bom usar o comando ***git push*** ele vai submeter para o arquivo que e o controller agora para atualizar posemos usar o comando ***git pull*** ele vai atualizar o clone.

***Mexendo com tag***

Quando criamos uma teg agente faz isso quando esta em uma versão que já esta em uma versão já pode ser usado mas para criar uma teg usamos o comando ***git tag nome\_dag*** e para enviar para uma versão para o github agente usa o comando ***git push origin nome\_tag*** mas quando fizemos isso as pessoas não ira ver e para atualizar podemos usar o comando ***git pull*** ele vai atualizar.

***Reverter os commit***

Quando usamos o comando revert ele vai refazer o commir e vai e refaz o commit usando o comando ***git resert <commit\_id>*** como no exemplo abaixo:

***Git resert <431643>***

***Ajuda o projeto***

Para ajudar com o projeto pode ser usado para ajudar em um projeto mas só pode ajudar com o projeto podemos usar o ***pull requeste*** só se o profeto estivesse ***Fork*** de um projeto.