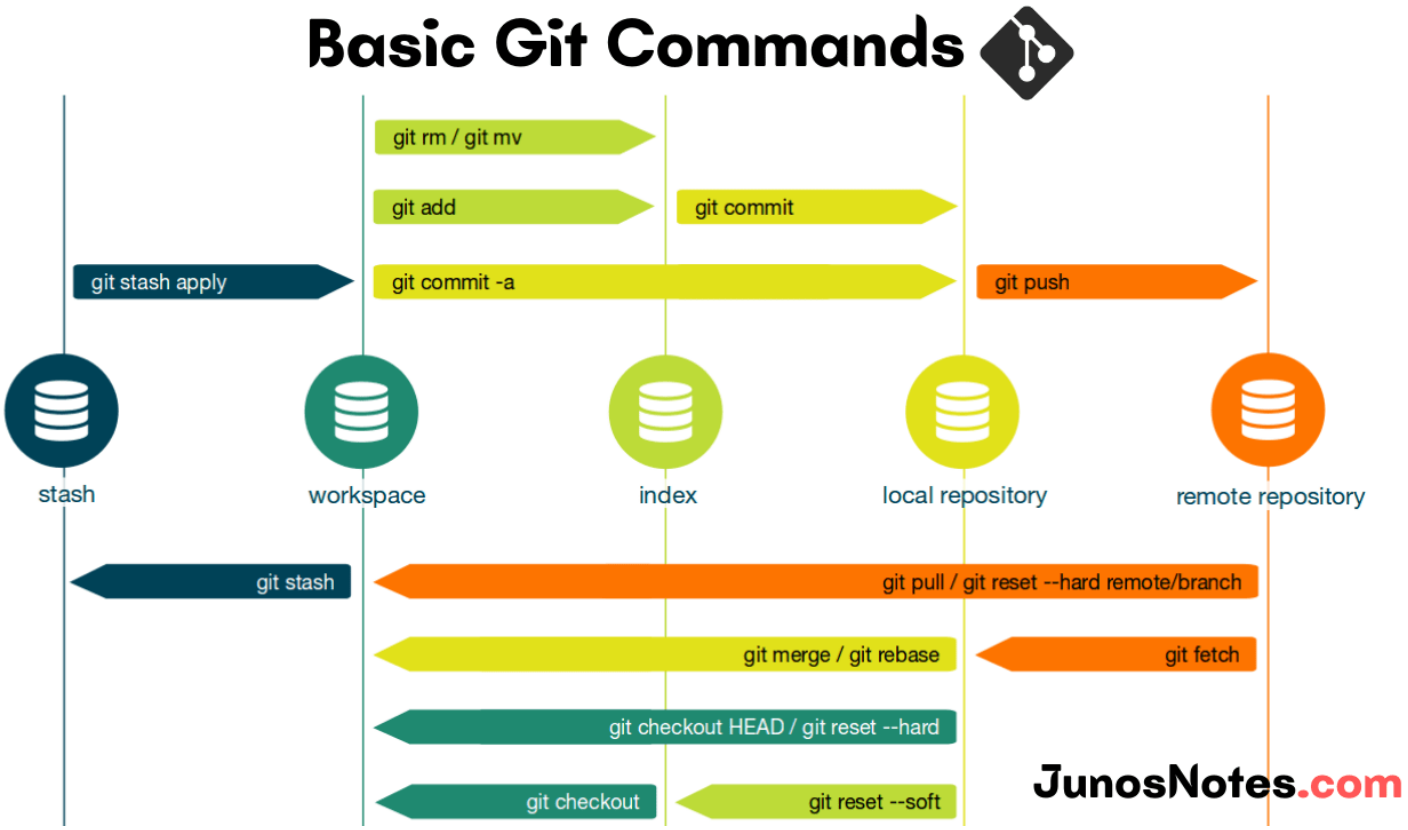


git init

Esse comando é responsável por inicializar um novo repositório. Em um processo de versionamento de código, esse é o primeiro comando básico.

git add

Este é responsável por adicionar supostas mudanças no diretório do projeto à área de staging, dando uma oportunidade de preparar o codigo antes de fazer o commit, existem duas formas de trabalha-lo com o '.' no final para enviar tudo, ou esepcificando a pasta ou arquivo a ser adiconado *'git add hello_world . php'*

git commit

Para submeter as mudanças inclusas no git add, usa-se o comando git commit, habitualmente é usado com a flag *'-m "mensagem"' ', que significa mensagem,* ou seja iremos adiconar um titulo a este commit, relacionado a alteração atrelada a ele. *'git commit - m " Adicionei o arquivo hello_world . php" '*

git status

Após realizar um commit é de práica comum realizar um git status, para verificar se existem mundaças no repositorio remoto as quais seu codigo local ainda não tem, existem duas possiveis respostas deste comando
1- 'On branch master No commits yet' - o que quer dizer que você esta em dia com o repositorio remoto, e não a mundaças a serem feitas.
2- 'On branch master Your branch is ahead of "nome da branch" ' - o que quer dizer que algum dos arquivos foi atualizado por outro usuario e o seu código local esta desatualizado

git push
Este comando é responsavel por enviar seu commit mais recente até o repositório remoto, existem algums formas de trabalhar com ele, e erros, os mais comuns são:

Git push -u origin 'nome da branch' - *a primeira forma é enviar o codigo para uma branch especifica*

Git push --set-upsteram-branch origin 'nome da branch' - aqui estamos associando esta branch uma flag -upstream que significa que esta branch esta atualizada e será a padrão, após realizar este comando, na proxima vez que for realizar o envio do codigo para o repositorio remoto pode usar apenas o *git push sem nenhuma flag ou especificação. * cuidado ao usar este comando trabalhando em branchs diferentes.*

O Erro mais comum é relacionado ao comando anterior:

```
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/CADOX-Technology/ARKEEVA.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Isso acontece porque estamos tentando enviar um código para um repositorio que esta a frente do local, para resolver este problema existem dois comando, e o mais usado é o git pull

Git pull

Git pull é a versão automatizada do git fetch. Ele baixa uma ramificação de um repositório remoto e a mescla imediatamente na ramificação atual, este comando tambe pode retornar erros, que veremos a frente com o ***git merge***

Após a realização bem sucedida deste comando, deve possiblitar a realização do git push para que ambos os códigos local e remoto fiquem iguais.

-Versionamento, atualização e erros-

Saindo dos comandos básicos

Git branch

Suas principaisfunções são a de criar uma branch, listar todas as branchs e deletar uma branch.
Uma branch pode nos auxiliar imensamente no processo de manutenção do código. Por exemplo, atualmente nosso arquivo "hello_world.php" está printando uma informação com erro de digitação. Para fazer a alteração correndo o menor número de risco, podemos criar uma nova branch apenas para isso. Assim, criamos a branch usando o seguinte código: git branch' nome da branch' (que em vias de documentação é recomendado que seja relacionado ao erro a ser consertado ou nova função a ser implementada)

git branch sem nenhuma especificação irá nos listar todas as branchs do projeto,
git branch -d 'nome da branch' é utilizado para deletar um branch

git checkout

Esse comando do Git tem como função mudar de branch ou voltar para algum estado do seu projeto. Sendo de extrema importância para qualquer repositório que utilize branchs, basta urilizar o comando git checkout 'nome da branch que eu quero utilizar'

Outra funcionalidade interessante do git checkout é sua utilização para voltar em algum commit anterior utilizando o comando base mais o código de soma de verificação do commit:
git checkout 166 ae0c4d3f420721acbb115cc33848dfcc2121a

Para localizar o código do commit iremos utilizar um novo comando

Git tag

Essa função do Git serve para criar etiquetas de estados que sejam relevantes, como por exemplo, aversão final ou alguma versão já utilizável do projeto. Assim, o Git salva versões para organizarmelhor o projeto e também para poder voltar se em uma versão posterior ocorrer algum problema.Existem três formas de criar uma tag utilizando o Git, sendo elas: com anotações, sem anotações e após o commit já feito.
Para criar uma tag com mensagem se utiliza o comando abaixo colocando o nome da tag após o -a e uma mensagem qualquer depois do -m:
git tag -a v1 .0 -m ' Minha vers ã o 1.0 '
Já para criar uma tag sem anotações usa-se apenas o comando base com nome da versão:
git tag v1 .1
E para marcar um commit depois de já tê-lo feito se usa o comando git tag com o nome da versão e a soma de verificação do commit.
git tag v1 .2 166 ae0c4d3f420721acbb115cc33848dfcc2121a

Agora como achar a verificação do commit? Cada commit feito tem uma soma de numero e letras para indentifica-lo

git log

Permite que você explore as revisões passadas de um projeto. Ele oferece muitas opções de formatação para indicar os snapshots commitados, ao rodar o comando no cmd temos um resultado parecido com este:

commit 2d952be9182e53bd1f7f0fd4e652e44f0fb3333c (HEAD -> master)
Author: lMainente <caetanomainente@outlook.com>
Date: Thu Feb 1 16:51:51 2024 -0300

manual git

commit 7c46c3b13e0879f523d8a47089f7297ea576e923 (origin/master, origin/HEAD)
Author: lMainente <caetanomainente@outlook.com>
Date: Thu Feb 1 13:03:43 2024 -0300

conexao com o banco

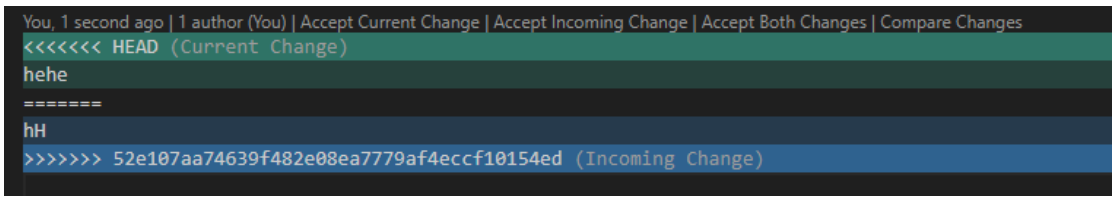
commit 5bfcbf8a116654434d9d1996a27b441fb87eda3e
Author: lMainente <caetanomainente@outlook.com>

Onde podemos identificar o codigo do commit para açoes futuras, junto com seu autor e comentario.

Git merge

Combina as alterações de diferentes branches no repositório. O comando git merge é usado para incorporar as mudanças de uma branch específica em outra branch ativa.
Exemplo:
Suponha que você esteja atualmente na branch "main" e queira mesclar as alterações da branch "feature-nova" nela.
Certifique-se de estar na branch de destino:
git checkout main
Execute o comando git merge para mesclar as alterações da branch "feature-nova":
git merge feature-nova
Isso irá incorporar as alterações da branch "feature-nova" na branch "main". Se houver conflitos, você precisará resolvê-los manualmente.

Este é um comando que pode ser realizado automaticamente com o git pull, quando nos temos um arquivo com codigos conflitantes nas versoes, usando a IDE vscode é possivel realizar este processo via source control, escolhendo entre quatro opçoes



Accept Current Change, Accept Incoming Change, Accept both changes e Compare Changes

Onde:

Accept Current Change mantem o código feito por você, que esta no seu repositório local

Accept Incoming Change substitui o seu bloco de codigo por um no qual esta vindo do repositorio remoto

Accept both changes mantem os dois códigos

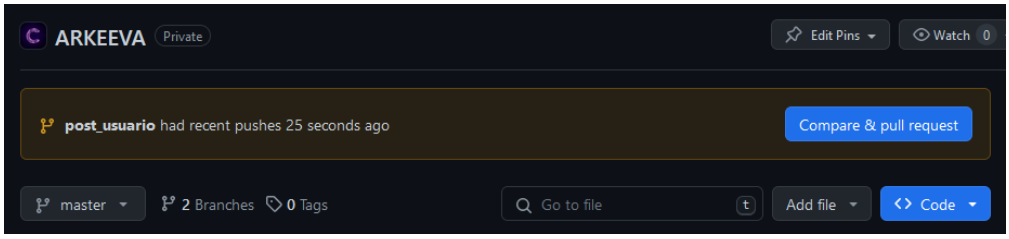
Compare Changes abre um editor para que você possa vizualizar ambos os codigos conflitantes de forma mais clara e dependendo do tamanho do bloco, resolve-los a mão

Após Realizar este processo é necessario novamente commitar as mundaças realizadas e agora sim enviar via git push para o repositorio remoto, porem tambem existe outra forma de trabalhar este envio, dependendo da alteração ou tamanho da sua equipe, o processo de git push ou merge direto na branch main pode afetar a ordem de trabalho, por isso existe:

Git pull request

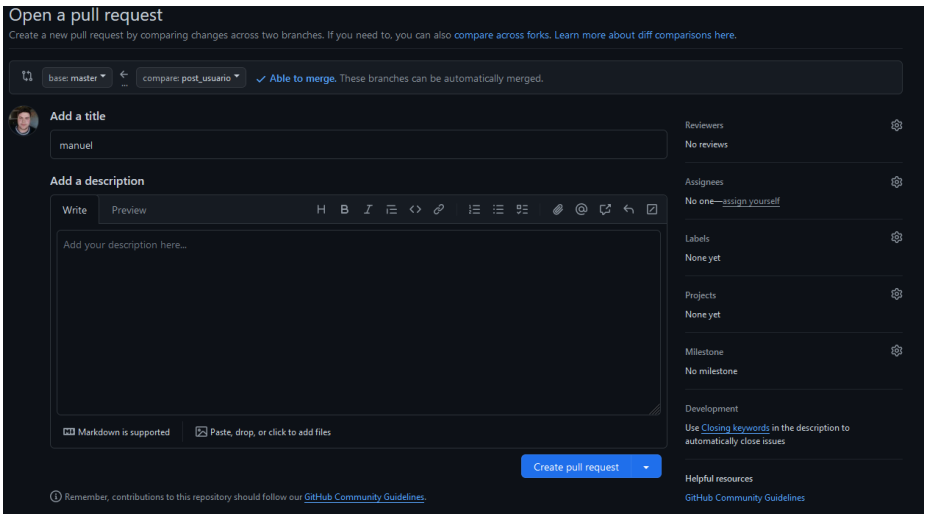
Este comando que não é bem um comando é realizado pelo github, ou seja pela interface grafica do git.

Após realizar o push do codigo na sua nova branch, você ira notar no repositorio uma notificação



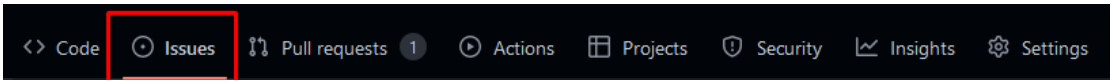
Essa notifiação se refere a que você pode criar um pull request do codigo da sua branch, para que um responsavel realize o comando git merge na branch principal.

Ao clicar voce pode colcoar um titulo assim como descrição e atrelar tags, e reponsaveis pela revisão

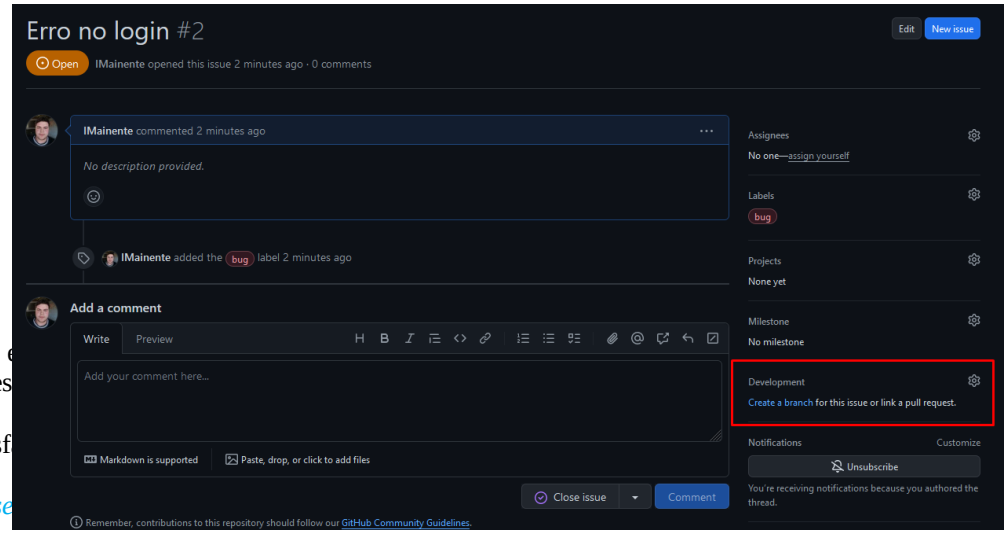


Após solciitar você vera seu pull resquest na fila para merge, podendo ser realizado pelo git hub, ou comando merge no cmd.

Agora iremos para a parte de issues do github



Uma forma de manter os bugs encontrado no projeto de forma organizada e sempre que identificado um criar um issue no github, desta forma é possivel vizualizar erros que ja aconteceram e qual o código relacionado a resolução dele, uma vez a qual voce pode criar um branch pelo github para este erro, assim caso aconteça algo parecido de novo existe um log.



Agora e
versoes

1. Desf

git rese

Este comando desfaz o último commit, mantendo as alterações no seu working directory. Você pode então fazer as correções necessárias e commitar novamente.

2. Desfazer o último commit (inclusivo dos arquivos):

*git reset --hard HEAD^*

Este comando desfaz o último commit e descarta as alterações feitas, revertendo completamente para o estado do commit anterior.

3. Alterar a mensagem do último commit:

git commit --amend -m "Nova mensagem do commit"

Este comando permite que você altere a mensagem do último commit. Ele também pode ser usado para adicionar mais alterações ao último commit se você esqueceu de incluir alguma coisa.

4. Reverter um commit existente:

git revert codigo-do-Commit (revisitar git log)

Isso cria um novo commit que desfaz as alterações introduzidas por um commit anterior, mantendo o histórico de commits intacto.

5. Rebase interativo para alterar commits:

git rebase -i HEAD~n

O rebase interativo permite reescrever o histórico de commits de uma branch. n é o número de commits que você deseja revisar. Uma vez no modo interativo, você pode escolher "edit" para cada commit que deseja alterar, fazer as alterações necessárias e, em seguida, continuar com o rebase.