**Inicializando repositório**

* Crie um a pasta qualquer contendo um arquivo ou não
* Abra o **git bash**
* Navegue até a pasta criada
* Digite **git init** e aperte ENTER
* Agora você ja iniciliaziou um repositório vazio oque for manipulado dentro dessa pasta o git conseguira absorver e mostrar todas informações

**Códigos:**

**git init**

**git config: - -global init.defaultBranch main –** Seta a branch principal que vem padrão máster para escolhida pelo usuário por boa prática o ideal é a **main**

**Criar pasta ou arquivo vazio:**

**touch** (nome da pasta ou arquivo) – Cria pasta vazio caso queira adicionar mais pastas é só dar espaço e adionar a nova pasta ou arquivo

**Criar pasta com gitkeep**

**touch** (nome da pasta)/.gitkeep – por convenção é bom criar essa pasta junto com esse arquivo

**Ver histórico:**

**git log** – Averigua o histórico básico

**git reflog** – Averigua todo o histórico feito até o momento

**Adiconar ou remover pasta ou arquivo dentro do gitignore**

**echo** (Nome da pasta) **>** .gitignore – cria um arquivo ignore no repositório e adiciona a pasta dentro dele essa pasta será ignorada pelo git

**echo > .gitignore –** Retira a pasta ou arquivo de dentro do gitignore

**echo “nome” > nome -**  Cria um arquivo

**Mudar mensagem do commit feito:**

**git commit - -amend -m “nova mensagem”**

**git commit - -amend –** entrará na tela do editor se usar o vin digite I para poder editar a mensagem do commit – para sair **ESC : w q**

**Mudar de master para main:**

**git branch -m master main**

**git push -u origin main**

**Entrar no diretório Git:**

**cd .git –** Entra no diretório git não mostrará nada pois quando colocamos .(alguma coisa) é referente a pasta oculta, caso queira ver o que tem dentro digite depois do final do endereço **.git** ele irá entrar na pasta e mostrar os arquivos dela

Acessar configurações do repositório:

Dentro do diretório **.git** digite:

**cat config**

**Averiguar último commit das branches:**

**Git branch -v – Mostra o último commit das branches que estiverem na árvore**

**Configurar seu nome e e-mail na pasta do repositório:**

**git config - - local user.name “nome”**

**git config - - local user.email “email”**

**Inicializando repositório Remoto**

**Mesmo processo acima só usar o:**

**git init - -bare**

“copie o endereço onde esse repositório remoto foi criado volte para pasta do repositório principal e siga o passo sincronizando os dados remoto na máquina”

**Caso inicializou sem o bare logo em seguida:**

**git config core.bare true**

**Puxar dados do repositório remoto para máquina pela primeira vez**

**git clone (endereço, URL, nome pasta etc) nome da pasta desejada**

**git clone** (endereço, URL, nome pasta etc) **- -branch –** (Clona uma branch especifica do repositório) **- -single-branch**

**Sincronizando os dados remoto**

**na máquina**

**Na pasta main ou máster do repositório local:**

**git remote –** mostra lista remota

**git remote add “nome de preferencia” (endereço na rede, URL ou nome pasta, “pode ser end. do github”)**

**git remote –v –** Mostra o endereço a pasta do repositório remoto (fetch=busca dados, push=envia dados)

**git remote rename** {nome atual} {novo nome}

**git clone** (endereço, URL, nome pasta etc) nome da pasta – traz um repositório vázio

**git push** {nome remoto} {nome pasta local} – envia o dados da pasta de trabalho para o repositório remoto (se colocar git push –u – após isso sempre que for enviar será só digitar git push ele enviará sempre para o endereço que digitou)

OBS: Após clonar repositório remoto para sua pasta ao executar o git remote ele mostrará o nome origin você pode deixar assim ou se quiser mudar de nome para manter uma paridade com sua prefenrência faça:

**git remote** rename origin {nome de preferência} – para o repositório remoto do github por boa prática manter o origin não renomear

OBS: Agora é só puxar os dados para máquina com, também é possível empurrar as atualizações enviadas com o push:

**git pull** {nome do remoto} {branch}

**git pull –** Caso aconteça uma alteração no mesmo arquivo. Fazendo esse comando puro ele ira puxar o que itcer do repositório remoto

Caso este problema esteja ocorrendo **- fatal: refusing to merge unrelated histories** Use:

**git pull origin main --allow-unrelated-histories -**

**Trabalhando com branch(ramificações)**

OBS: Com as branchs conseguimos trabalhar em ramificações diferentes sem interferir na branch principal sendo assim fazemos modificações em paralelo e depois de tudo certo enviamos para a principal

**git branch** – mostra as branchs e em qual estamos

**git branch {nome}** – cria uma nova branch

**Excluir branch:**

**git branch -d (nome da branch)**

OBS: Navegar nas branchs:

**git checkout** {nome branch} – muda pra branch

**git checkout –b** {nome a acrescentar} – cria branch e já muda para a branch criada

**git merge** {nome do local que vou puxar} – puxa a att de um local para onde você está atualmente

OBS: caso apareça uma tela para atualizar algum texto e você não queira atualizar digite :x

**git rebase** {nome do local que vou puxar} – Atualizar a branch desejada além de levar a atualização da branch que está trabalhando

OBS: Quando for enviar alguma atualização e caso algum colega de trabalho tenha feita alguma alteração o git irá mostrar uma mensagem que você precisa trabalhar com a versão mais atual antes de fazer o push então tem que fazer o pull primeiro depois sim fazer o push

**Remover add commit, arquivos**

**git checkout** - - {arquivo} – Desfaz a o trabalho feito antes de dar o **add**

**git reset HEAD** {arquivo} – Após ser marcada “add” execute esse comando e depois o comando acima

**git reset - -soft** {HASH commit pego no log} – Volta ao estado do commit inicial deixando o arquivos antes commitados na área de preparação.

**git reset - -mixed** {HASH commit pego no log} – O mixed é padrão do reset então pode-se deixar o reset sozinho caso queira. O arquivo commitado volta para o status de untracked sendo necessário caso queira se adicionado novamente para depois ser commitado.

**git reset - -hard** {HASH commit pego no log} – Volta ao estado de onde você escolheu voltar com HASH apagando todos os commits posteriores

**git revert** {hash do commit} – Após feito o commit execute esse comando, ele retira toda a atualização feita nesse commit direto

**git restore** (nome do arquivo) – Desfaz todas as alterações feitas em um arquivo antes de adicionar

**Remover Versionamento:**

**Excluindo .git – recursivamente e a força**

rm –rf .git

**Guardando para depois**

**git stash** – arquiva o trabalho com um hash

**git stash list** – mostra o que foi salvo

OBS: Puxar para o trabalho novamente 2 opções:

**git stash apply** {número} – puxa o trabalho mais mantém salvo em stash para apagar faça: **git stash drop**

OBS: Para puxar e apagar a ultima alteração feita use:

**git stash pop** – faz o merge das últimas atts e já tirá da stash da última att

**Viajando no tempo**

OBS: **git log - -oneline** para o hash – Use para viajar no tempo

**git checkout** {hash desejada} – posso trabalhar nesse ponto que voltei e caso não queira utilizar essas mudanças na branch principal posso somente voltar para branch principal que tudo estará como estava

OBS: Se quiser utilizar as mudanças feitas na branch principal é só criar uma nova branch com **git branch {nome} ou git checkout –b {nome}**

**Baixar arquivo da branch remota sem mesclar com a branch local:**

**git fetch (nome branch remota) (nome branch local) –** Baixa o arquivo ainda sem enviar para usa branch local sendo necessário utilizar o comando merge origin/a branch para baixar o arquivo para usa pasta local sem mesclar com sua branch local

**Vendo as alterações**

**git diff** (nome da main atual) (origin/nome da branch que baixou os arquivos) – mostra na branch remota o conteúdo dentro da mesma

**git diff** {1º hash} {2º hash} – Vizualizar da 1º hash até a 2º hash inserida

**git diff** – Vizualizar o que ainda não foi adicionado “add”

**Tags e Releases**

**git tag** –a {nome que desejar}

**git tag** – vizualizar as tag – release

**git push** {local ou github} {tag} – Pode enviar para o repositório local ou do github