* É onde o código será armazenado;
* Na maioria das vezes cada projeto tem um repositório;
* Quando criamos um repositório estamos iniciando um projeto;
* O repositório pode ir para servidores que são especializados em gerenciar repôs, como: GitHub e Bitbucket;
* Cada um dos desenvolvedores do time pode baixar o repositório e criar versões diferentes em sua máquina;
* Para criar um repositório utilizamos o comando: git init;
* Desta maneira o git vai criar os arquivos necessários para inicializá-lo;
* Que estão na pasta oculta .git;
* Após este comando o diretório atual será reconhecido pelo git como um projeto e responderá aos seus demais comandos;

Para acessar o Git através do VSCode, você precisará ir lá em cima em terminal e clicar na opção New Terminal  
Que irá liberar o terminal para você conseguir se comunicar com o git

Para criar o git em alguma pasta crie uma pasta onde você vai salvar o git e depois no terminal digite o caminho onde você vai querer dar o git init:

Exemplo:  
cd C:\Users\pafonso\Desktop\Curso\_prático\git\arquivo

Após o terminal ir até o caminho selecionado, de o git init para que o git seja criado naquela pasta escolhida,

Digitando git status vc ver o status do seu repositório, caso não tenha nenhum repositório criado aparecerá a seguinte mensagem  
Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Para criar um repositório utilizamos o comando git init:  
Texto

O conteúdo gerado por IA pode estar incorreto.  
  
Tanto no Git Bash quanto no vscode, quando damos um git init, o repositório vai ser criado e se vc der um git status, você vera que o “fatal:”, não aparecerá mais.

Provavelmente apareça isso:  
Texto

O conteúdo gerado por IA pode estar incorreto.  
  
essas pastas em vermelho são arquivos e pastas não rastreados pelo git, como você acabou de dar o git init, o git tá indicando tudo o que está no diretório, mas que ainda não foi adicionado ao controle de versão, não é um erro.

* É um serviço para gerenciar repositório, gratuito e amplamente utilizado;
* Podemos enviar nossos projetos para o GitHub e disponibilizá-lo para outros devs;
* O GitHub é gratuito tanto para projetos públicos como privados;
* Podemos facilmente enviar nossos repôs para o GitHub;
* Precisamos criar o projeto no GItHUb, inicializar o git em nossa máquina, sincronizar com o GH e enviar;
* E esta sequência que parece ser complexa é facilmente executada por poucos comandos;
* Vale lembrar que só fazemos uma vez por projeto este fluxo;

PASSA A PASSO

* Primeiro passo clicar, em new repositório e criar um repositório no GitHub, segue exemplo:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

* Readme é um arquivo que explica o que vem no repositório, de tudo que ele faz, do projeto em si.
* .gitignore é um arquivo que vc adicionar para que o git não monitore arquivos que você não queira monitorar.
* A licença diz o que as pessoas podem ou não fazer com o seu código.
* Após criar o seu repositório no GH ele vai direcionar para você para um local que te mostrará o passo a passo de como seguir para fazer na sua máquina, onde você executará no terminal do seu VSCODE ou Git Bash, segue o print:  
  Interface gráfica do usuário, Texto, Aplicativo, Email

  O conteúdo gerado por IA pode estar incorreto.
* Primeiro passo criar a pasta onde que conterá o arquivo que vc deseja subir para o repositório
* Ao seguir o passo a passo de abrir um novo terminal no VS Code, caso o terminal não esteja aberto na pasta que vc deseja, basta dar o comando cd(+caminho da pasta) que vc deseja, caso ocorra do seu perfil de usuário esteja com o espaço no nome como no exemplo abaixo, você deve colocar o caminho entre aspas, desse jeito:



* Após estar na pasta, dar um git init para criar o git e dar um git status para ver se foi criada corretamente

Texto

O conteúdo gerado por IA pode estar incorreto.

* Lembrando que para subir um repositório para o GH, o repositório não pode estar vazio, então no exemplo acima criamos um arquivo de texto, chamado teste.txt dentro da pasta 1\_primeiro\_repo, ele tá em vermelho com essa informação “untracked” pq ainda não está sendo mapeado pelo git, para resolver isso basta dar um git add (nome e tipo de arquivo) conforme printe abaixo

Texto

O conteúdo gerado por IA pode estar incorreto.

* Após dar o git add, você pode verificar se o arquivo está sendo mapeado corretamente dando um git status e ele estará na cor verde, indicando que agora está sendo mapeado corretamente.
* Próximo passo, você vai dar um git commit -m “atualização”, o git commit é o comando que salva sua alteração dentro do git, para caso futuramente vc queira reutilizar, usei o nome atualização como referência para quando vc querer colocar outro nome, inicialmente colocamos git commit -m “Hello World git”, por causa da superstição dos programadores
* Pode ocorrer de dar o seguinte erro:  
  Texto

  O conteúdo gerado por IA pode estar incorreto.
* Esse erro acima indica que você ainda não configurou sua identidade, ou seja, seu nome e e-mail, que são obrigatórios pra fazer commits.

Para resolver isso é muito simples, abra o terminal e digite os dois comandos abaixo com seus dados:

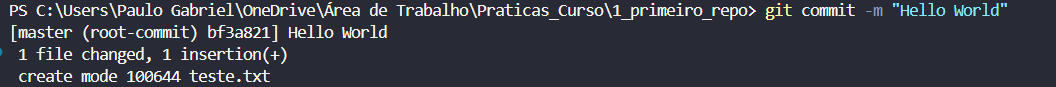
git config --global user.name "Paulo Gabriel"

git config --global user.email “[paulo@email.com](mailto:paulo@email.com)”

Segue exemplo:



Não precisa ser seu e-mail real, mas é bom usar um que você pretende usar no GitHub.

* Depois de configurar você pode dar só um git commit -m “Hello World git”, que irá aparecer da seguinte forma:  
  
* Logo em seguinda basta dar um git status para verificar se está tudo corretamente:  
  Texto

  O conteúdo gerado por IA pode estar incorreto.
* Agora precisaremos criar uma Branch master, uma Branch é uma versão parelela do seu projeto que te permite fazer, alterações, correções ou testes e não interfiram no seu código principal, e a Branch master é a padrão que o Git cria quando você inicializa um repositório, é como a trilha oficial do projeto onde geralmente ficam as versões estáveis e prontas para produção. O comando para criar a Branch é git branch -M master
* Próximo passo é o comando git remote add origin <https://github.com/PauloGAfonso/curso.git_1.git>, ele conecta seu repositório local ao repositório remoto no GitHub, o nome Origin é o apelido padrão para esse repositório remoto.
* E por fim git push-u origin master, que envia seu código (faz o push) seu código local para o GitHub, na Branch master. O -u configura essa Branch para acompanhar a origin/master, ou seja, facilita futuros git push e git pull, como saber se funcionou corretamente?
* Segue comprovação do funcionamento:  
  Interface gráfica do usuário, Texto

  O conteúdo gerado por IA pode estar incorreto.
* Pronto, você subiu um repositório para o github, compreendeu agora?
* As mudanças do projeto podem ser verificadas por: git status
* Este comando é utilizado muito frequentemente;
* Aqui serão mapeadas todas as alterações do projeto;
* Como: arquivos não monitorados e arquivos modificados;
* Podemos também dizer que é a diferença do que já está enviado ao servidor ou salvo no projeto;
* Caso vc adicione novos arquivos ao seu projeto, quando vc der um git status, ele aparecerá como um arquivo não mapeado (untracked), enquanto aqueles que já foram mapeados porém houve alguma alteração neles, a mínima que seja, ele aparecerá com um M indicando que foi modificado e precisa ser comitado, caso ocorra tudo isso ele ficará assim:  
  Interface gráfica do usuário, Texto, Aplicativo

  O conteúdo gerado por IA pode estar incorreto.

Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

* Para adicionar arquivos novos a um projeto utilizamos: git add
* Podemos adicionar um arquivo específico como também diversos de uma vez só;
* Somente adicionando arquivos eles serão monitorados pelo git;
* Ou seja, se não adicionar ele não está no controle de versão;
* É interessante utilizar este comando de tempos em tempos para não perder algo por descuido;

Para adicionar um arquivo vc, precisa usar o comando git add (nome do arquivo, sem os parênteses é claro), antes fazer isso, se vc der um git status, ele estará em vermelho indicando quais pastas vc poderá mapear, pois ainda estão untracked ou não mapeados, após o git add, parecerá que nada aconteceu e ele retornará ao prompt, mas se vc der um git status ele estará assim:  
Texto

O conteúdo gerado por IA pode estar incorreto.

O arquivo em verde indica que foi adicionado, e ele aparecerá assim na barra lateral.  
Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

O conteúdo gerado por IA pode estar incorreto.

Esse “A” indica que os arquivos foram adicionados no mapeamento do git.

Para evitar ficar adicionando um a um, o que atrasa muito o trabalho na hora de precisar agilizar o trabalho, vc pode usar o comando “git add .” que adiciona tudo de uma vez, mas tenha certeza de que vc que quando vc der esse comando, porque depois de executado, vc pode perder muito trabalho dando um “git reset”

O “git reset” é totalmente o oposto do “git add” enquanto o “git add” adicionando os arquivos para a área de staging(área que está preparada para ser comitada), o git reset remove dessa área, vc pode usar dois comandos do “git reset” caso vc queira remover apenas um arquivo faz da seguinte maneira

Para remover um arquivo específico:  
“git reset (nome do arquivo sem parênteses, por favor)”

Para remover todos os arquivos

“git reset”

Fique tranquilo que isso não apagará, nem alterará nenhum arquivo, só removerão da área de staging, área onde os arquivos ficam aguardando para serem comitados, mais ou menos como uma sala de espera antes de serem enviados ao repositório.

Para ter certeza de que os arquivos estão na área de staging, basta dar um git status:

Texto

O conteúdo gerado por IA pode estar incorreto.

Se eles estiverem em verde é porque estão lá, se ele estiver em vermelho, mesmo depois do git add, significa que houve alteração, ou seja, ele teoricamente se torna um novo arquivo com novas informações, vc precisará de novo dar um “git add” pois senão, se vc der um commit, ele salvará a informação antes da alteração que vc fez.

porém para salvar precisa ser comitado, se não a alteração não ocorre mesmo com o git add, é o mesmo que vc preparar um pacote para ser enviado, vc guarda o pacote numa caixa, mas isso não indica que será enviado ao destinatário.

* AS alterações salvas do projetos são realizadas por git commit
* Podemos commitar arquivos específicos ou vários de uma vez com a flag -a
* É uma boa prática enviar uma mensagem a cada commit, com as alterações que foram feitas;
* A mensagem pode ser adicionada com a flag -m

A flag “-m” significa “message” é usada para vc deixar a informação do que foi feito nessa alteração para subir, e a flag “-a” significa “all” serve para vc não precisar ficar comitando um arquivo por vez e já poder adicionar tudo de uma vez

LEMBRANDO QUE VOCÊ SÓ PODE COMITAR ARQUIVOS QUE ESTEJAM NA AREA DE STAGING, OU SEJA, ARQUIVOS QUE VC JÁ DEU UM GIT ADD.

Quando vc precisar subir apenas um arquivo vc faz da seguinte forma.  
Antes de comitar eles estarão assim:  
Texto

O conteúdo gerado por IA pode estar incorreto.

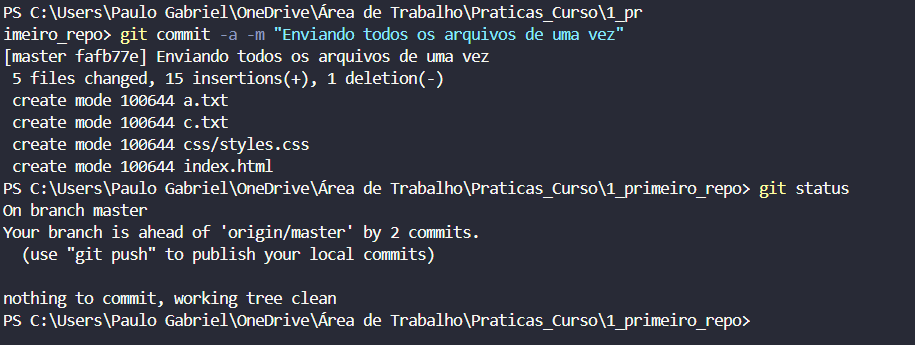
Eles em verde indicam que estão foram mapeados e estão na área de staging, esperando para serem comitados, para serem comitados vc pode utilizar duas formas, a primeira é:

Adicionando um arquivo por vez usando  
“git commit exemplo.txt -m “Escreve o que vc fez”  
Texto

O conteúdo gerado por IA pode estar incorreto.

Perceba que no exemplo acima a gente fez um commit do arquivo “b.txt e dando um git status depois, podemos ver que ele não está mais na área de staging, pois já foi comitado.

Além de adicionar um arquivo por vez podemos adicionar todos os arquivos na área de staging de uma vez usando:

“git commit exemplo.txt -a -m “Escreve o que vc alterou ou fez”  


Agora pode verificar que todos os arquivos foram comitados, e durante o comit, houve mudança de 5 arquivos, foram adicionados 15 conteúdos e algo foi deletado (Não se preocupe muito com essa parte, só fique de olho para verificar se não teve nenhuma mensagem de erro indicando que o commit falhou, para ter certeza lembre-se de dar um git status, e as opções em verde sumirão, pq não estão na área de staging mais, foram salvas prontas para ser enviadas ao repositório.

* Quando finalizamos uma funcionalidade nova, enviamos o código ao repositório remoto, que é código-fonte;
* Esta ação é feita pelo git push
* Após esta ação o código do servidor será atualizado baseando-se no código local enviado;

Primeiro passo devemos dar um “git status”, para termos certeza, que todos os arquivos não estão na área de staging ou não estão como untracked(não mapeados):  
Texto

O conteúdo gerado por IA pode estar incorreto.  
Se estiver escrito como está na imagem acima, quer dizer que tá tudo certo, todos os arquivos foram mapeados e comitados e estão prontos para ser enviados ao repositório remoto