Git

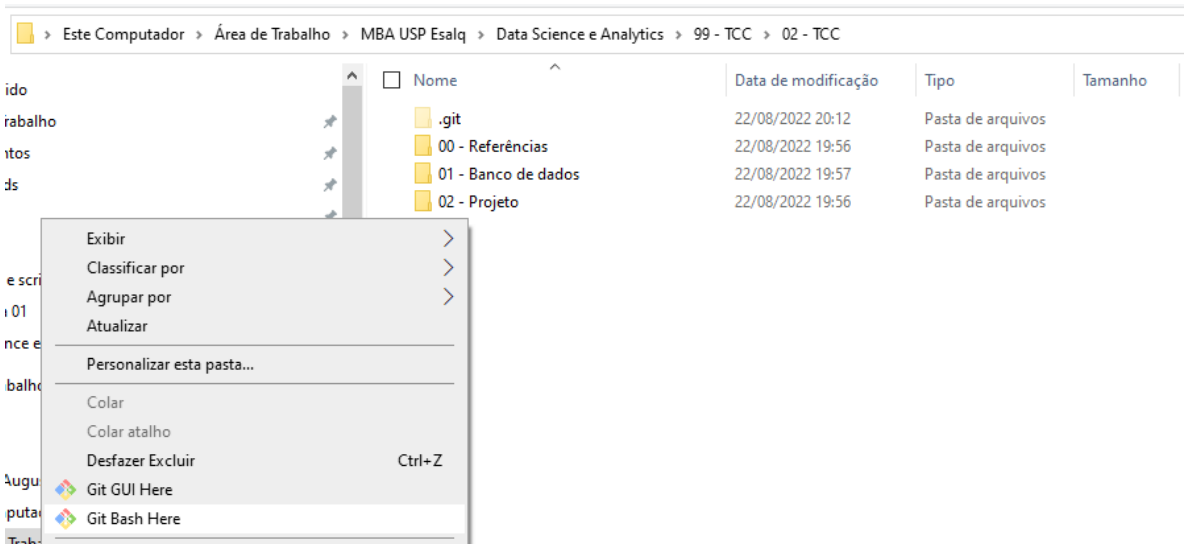
- Sistema de controle de versão mais utilizado no mundo;
- Baseado em repositórios, que contem todas as versões do código e também as cópias de cada desenvolvedor;
- Todas as operações do git são otimizadas para ter alto desempenho;
- Todos os objetos do git são protegidos com criptografia para evitar alterações indevidas e/ou maliciosas;
- O git é um projeto de código aberto.

Repositório

- Local onde o código será armazenado;
- Cada projeto tem o seu repositório;
- GitHub e BitBucket são servidores especializados para gerenciar repositórios;

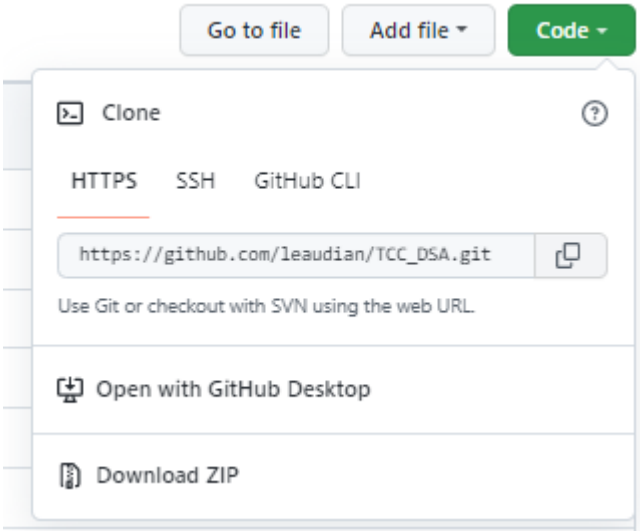
Comandos

1. Selecionar a pasta onde serão armazenados os arquivos, clicar com o botão direito do mouse e selecionar a opção em "Git Bash Here"



2. Utilizar o comando "git init" para iniciar o projeto;
3. Utilizar o comando "git add ." para adicionar todos os arquivos do endereço de rede do projeto;
4. Utilizar o comando "git commit -a -m "comentários"" para comitar a versão dos arquivos (salvar);
5. Utilizar o comando "git branch -M main" para criar o branch principal do repositório;
6. Utilizar o comando "git remote add origin endereçoGIT" para vincular o repositório à pasta desejada;
7. Utilizar o comando "git push -u origin main" para ativar o repositório;
8. Utilizar o comando "git push" para subir os arquivos (enviar o código para o servidor);
9. Utilizar o comando "git pull" para baixar atualizações dos arquivos (buscar atualizações do código do servidor);

10. Utilizar o comando "git clone endereçoGIT" para baixar o repositório inteiro (clonar o repositório do servidor);



8. Utilizar o comando "git rm arquivo.extensão" para remover o arquivo do repositório do servidor;

Obs: ainda será necessário fazer o commit e push para concretizar a remoção do arquivo do repositório.

9. Utilizar o comando "git log" para exibir o log dos commits realizados bem como os respectivos comentários, responsáveis e datas;

10. Utilizar o comando "git mv arquivo.extensão endereço_novo/arquivo.extensão" para mover o arquivo entre pastas dentro do diretório;

Obs: ainda será necessário fazer o commit e push para concretizar a remoção do arquivo do repositório.

11. Utilizar o comando "git mv nome_arquivo_velho.extensão nome_arquivo_novo.extensão" para renomear um arquivo dentro do diretório;

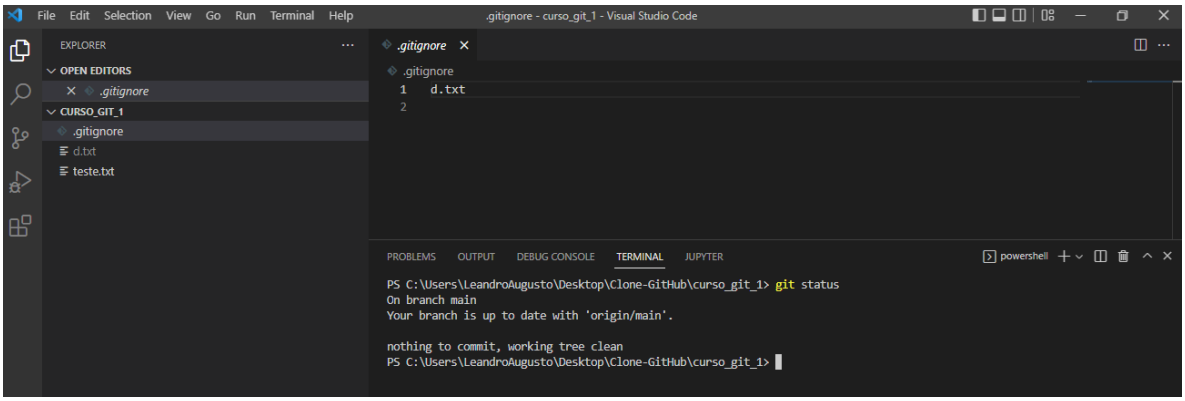
Obs: ainda será necessário fazer o commit e push para concretizar a alteração de nome do arquivo do repositório.

12. Utilizar o comando "git checkout arquivo.extensão" para cancelar a pendência "modified" de um arquivo que não seja interessante atualizar no repositório do servidor (reseta o arquivo conforme condição original no repositório do servidor);

13. Criar um arquivo ".gitignore" e dentro dele listar os arquivos a serem ignorados com sua respectiva extensão.

Obs1: Criar primeiro o arquivo do gitignore contendo a lista de arquivos a serem ignorados e posteriormente criar os arquivos ignorados.

Obs2: Será necessário fazer commit e push para concretizar o funcionamento do arquivo ignore.



14. Utilizar o comando "git reset --hard origin/main" quando tivermos itens comitados e não comitados que deverão ser desfeitos antes de serem submetidos ao servidor.

15. Utilizar o comando "git branch" para listar todos os branchs do projeto;

Obs: O branch que aparecer com "*" ao lado é o branch atual de trabalho.

16. Utilizar o comando "git branch novo_branch" para criar novo branch;

17. Utilizar o comando "git push origin nome_do_branch" para disponibilizar o branch no repositório;

18. Utilizar o comando "git branch -D nome_do_branch_a_ser_deletado" para se deletar um branch;

19. Utilizar o comando "git checkout nome_do_branch" para se alterar o branch de trabalho atual;

20. Utilizar o comando "git merge nome_do_branch" para se mesclar o branch de trabalho atual com o branch indicado no comando;

Obs: ainda será necessário fazer o commit e push para concretizar a mescla de branchs no repositório.

21. Utilizar o comando "git stash" para se salvar na memória temporária a alteração feita sem a necessidade de commit;

22. Utilizar o comando "git stash list" para exibir a relação de stashs salvas;

23. Utilizar o comando "git stash apply numero_da_stash" para aplicar a stash escolhida conforme seu número na stash list;

24. Utilizar o comando "git stash show -p numero_da_stash" para exibir detalhes da stash conforme número de stash selecionado;

25. Utilizar o comando "git stash drop numero_da_stash" para deletar a stash conforme número de stash selecionado;

26. Utilizar o comando "git stash clear" para deletar a lista completa de stash;

27. Utilizar o comando "git tag -a V1.0 -m "comentários"" para se criar uma tag;

Obs: Possui a finalidade de registro de marcos um branch, podendo ser utilizada como referência para retroceder o código até determinados pontos conforme estiver estruturado as tags.

28. Utilizar o comando "git tag" para exibir a lista de tags;

29. Utilizar o comando "git tag" para exibir a lista de tags;

30. Utilizar o comando "git show nome_da_tag" para exibir detalhes da tag selecionada;

31. Utilizar o comando "git checkout nome_da_tag" direciona o código para o momento da tag selecionada;

32. Utilizar o comando "git push origin nome_da_tag" para disponibilizar a tag no repositório;

33. Utilizar o comando "git push origin --tags" para disponibilizar todas as tags no repositório;

34. Utilizar o comando "git fetch -a" para buscar branches criados por outros usuários;

Obs: Mesmo após o comando do fetch o branch ainda não estará mapeado, para que isto aconteça será necessário realizar ao menos uma vez o git checkout para este novo branch.

35. Utilizar o comando "git show" para mostrar um log de modificações no branch;

36. Utilizar o comando "git diff master nome_do_branch" para mostrar detalhes da modificação do branch;

37. Utilizar o comando "git clean -f" para remover os arquivos untracked;

38. Utilizar o comando "git gb" (garbage collector) para otimizar o uso da memória do repositório;

39. Utilizar o comando "git fsck" (file system check) para verificar a existência de arquivos corrompidos dentro do repositório;

40. Utilizar o comando "git reflog" para mapear todas as alterações dentro do repositório (de todos os branches) executadas dentro do prazo de 30 dias;

Obs: Exibe a lista de código de cada movimentação, estes códigos (hashes) associados ao comando git reset permite retornar a alguma etapa anterior do desenvolvimento.

41. Utilizar o comando "git reset --hard codigo" permite retonar a alguma etapa anterior baseado no código (hash) obtido ao usar o comando git reflog;

42. Utilizar o comando "git archive --format zip --output nome_do_arquivo_zipado.zip nome_do_branch_a_ser_zipado" permite exportar uma cópia de um branch por um arquivo de extensão .zip;

43. Utilizar o comando "git checkout -b private_nome_do_branch" permite exportar uma cópia de um branch por um arquivo de extensão .zip;

Observações

git commit nome_do_arquivo.extensão -m "comentário" -> faz a atualização somente do arquivo indicado associando esta atualização ao comentário.

git commit -a -m "comentário" -> faz a atualização de todos os arquivos trackeados associando esta atualização ao comentário.

Quando uma alteração é feita e na sequência desfeita em um arquivo do repositório, aparecerá a pendência "modified" no git status. Para que suma esta pendência basta utilizar o comando "git checkout arquivo.extensão"