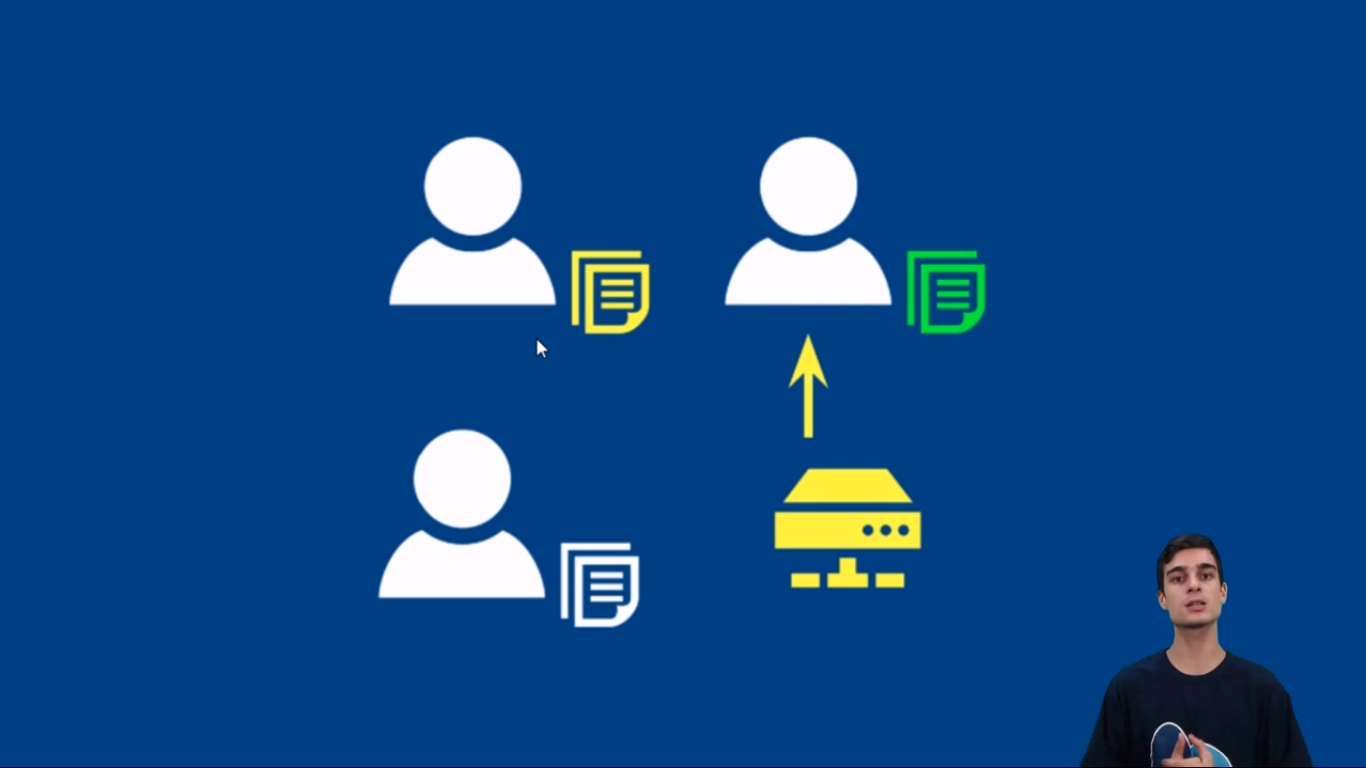
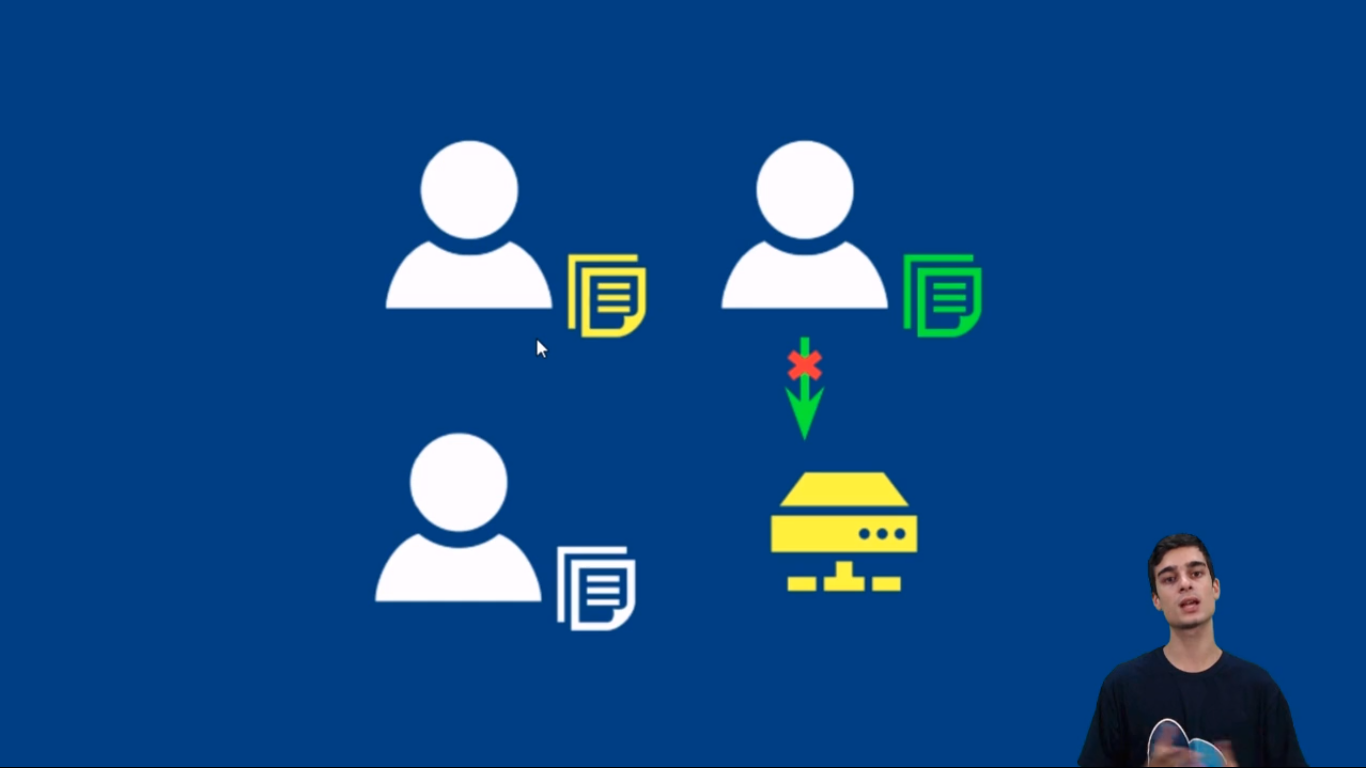
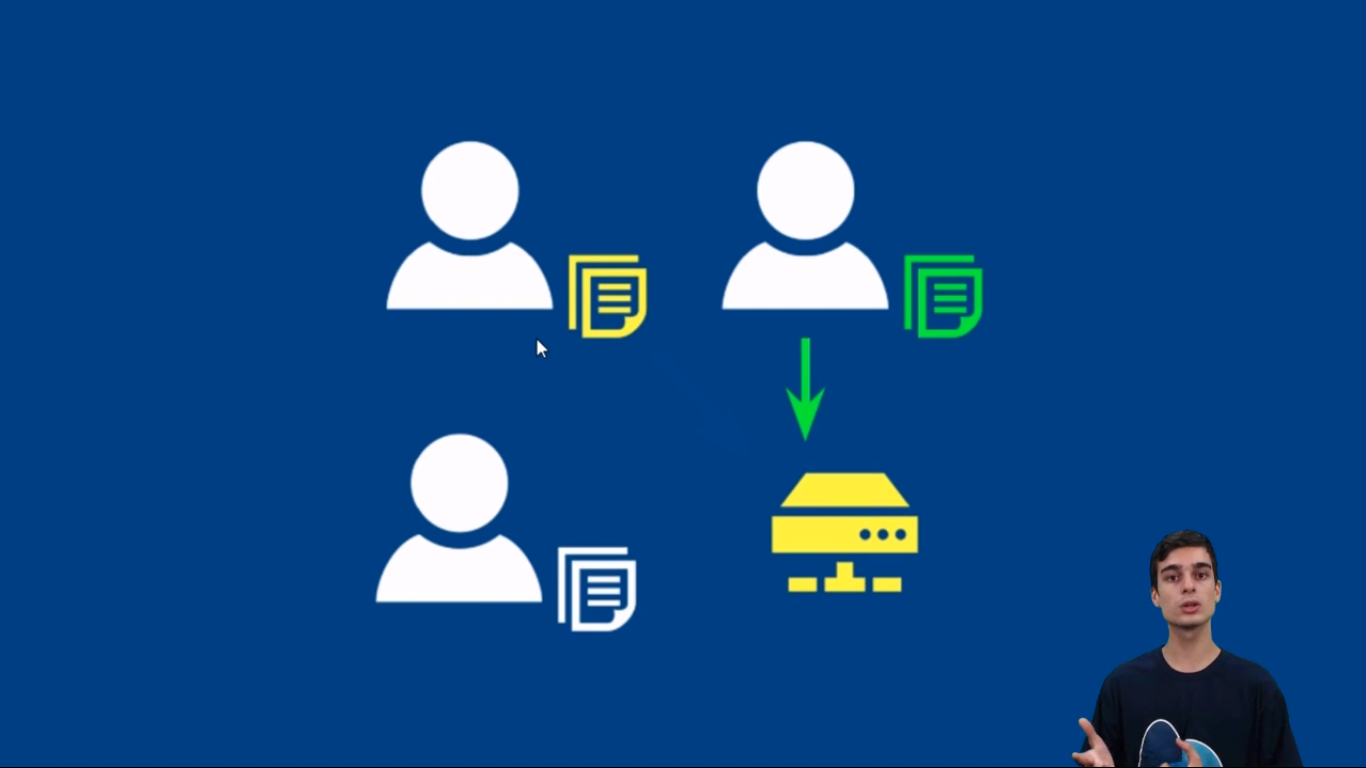
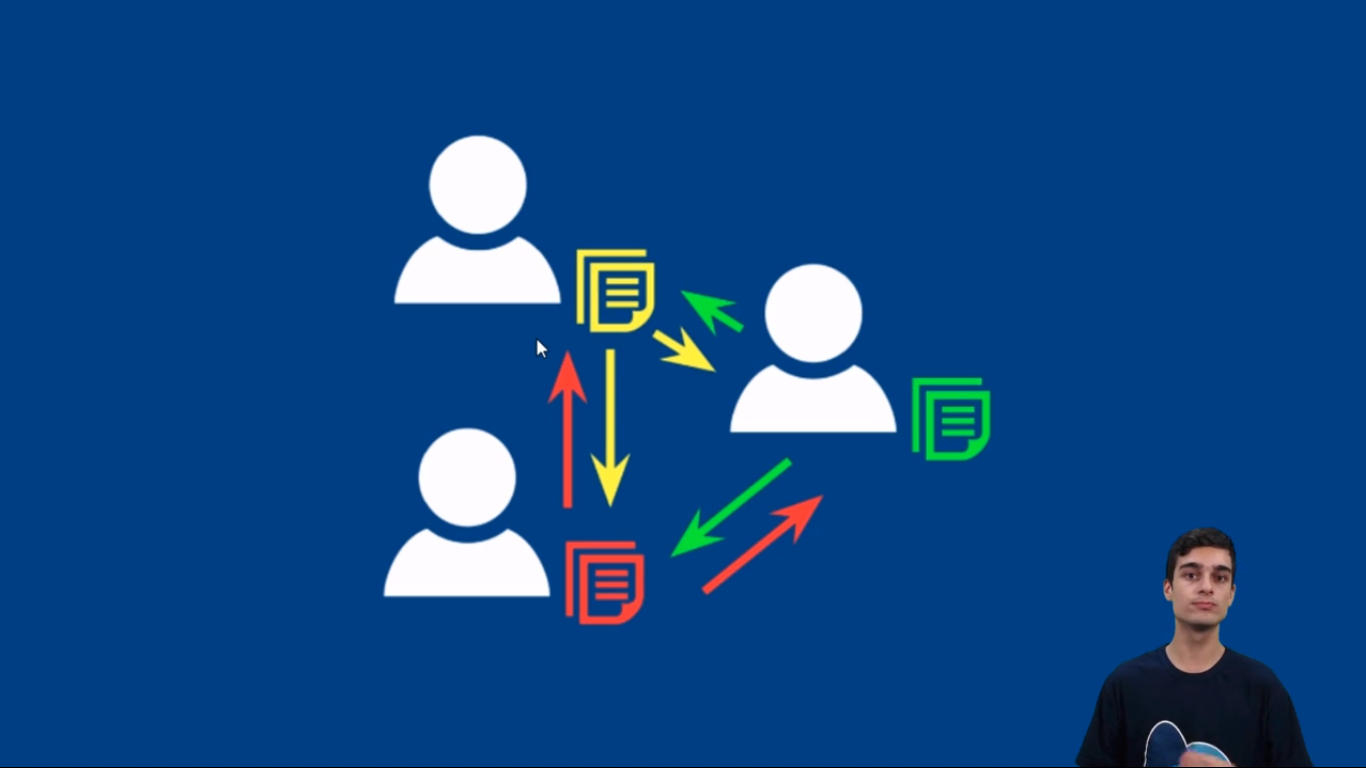
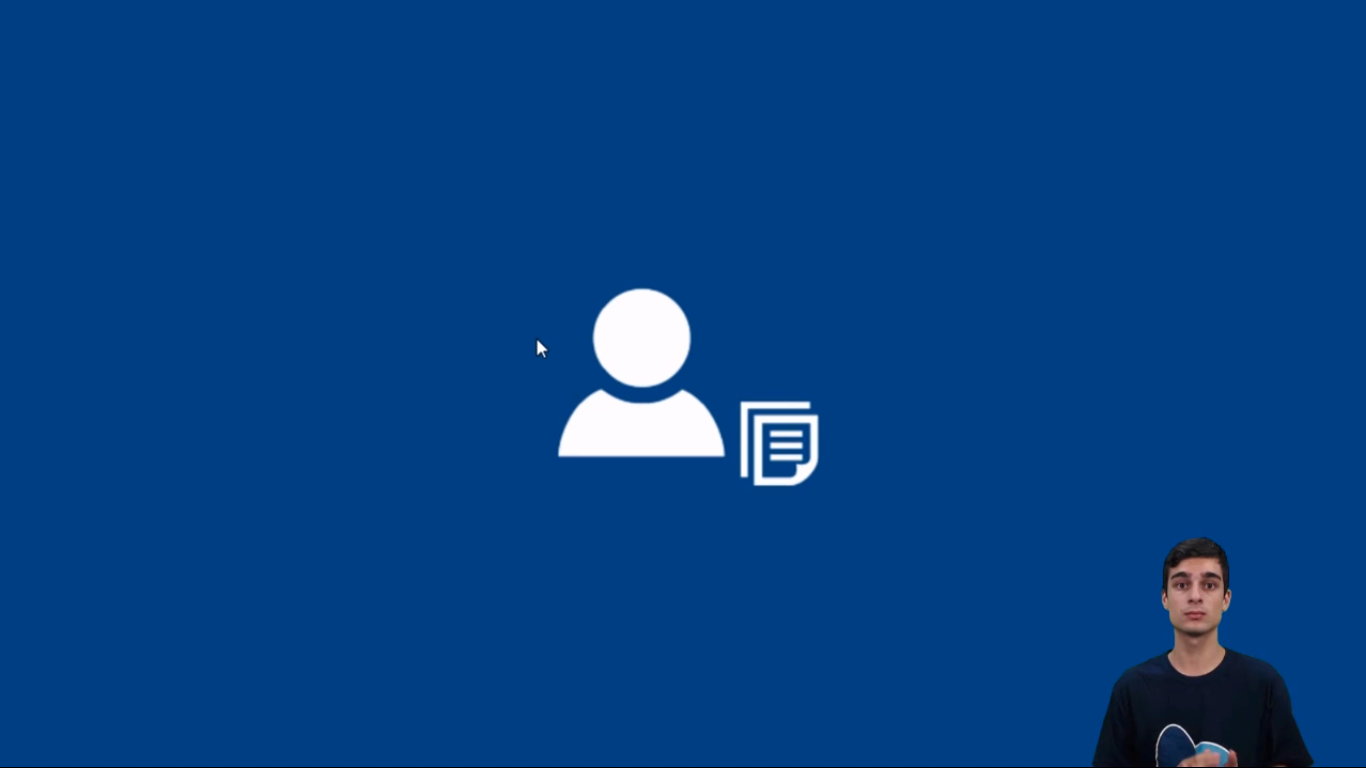
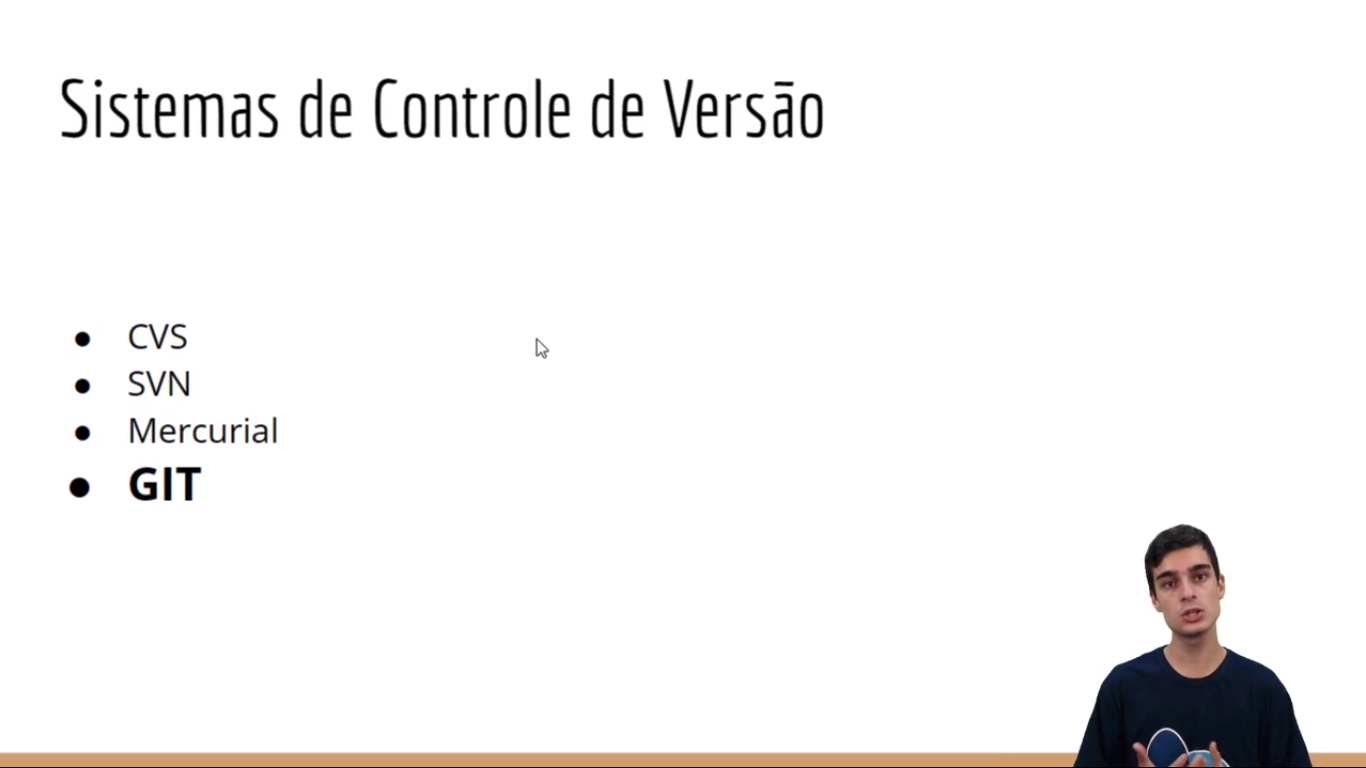
Semana 06

Git e Github - Controle e compartilhe seu código





Baixe e instale o git em sua máquina

git-scm.com

ao utilizar o git em sua máquina você poderá usar tanto o terminal do Windows (cmd) quando o terminal do git(git bash), nos próximos exemplos iremos utilizar o git bash

ls - lista os arquivos

git --version - verifica a versão instalada em sua máquina

iremos usar o repositório como exemplo:

/Documents/git-e-github/vinicius

Dentro desse repositório teremos o arquivo index.html e digitaremos

git init – inicia um novo repositório



Ao final do caminho será mostrado o ramo que se encontra esse repositório, nesse caso master

Para informar ao git quem sera o usuário:

git config --local user.name "Seu nome aqui"

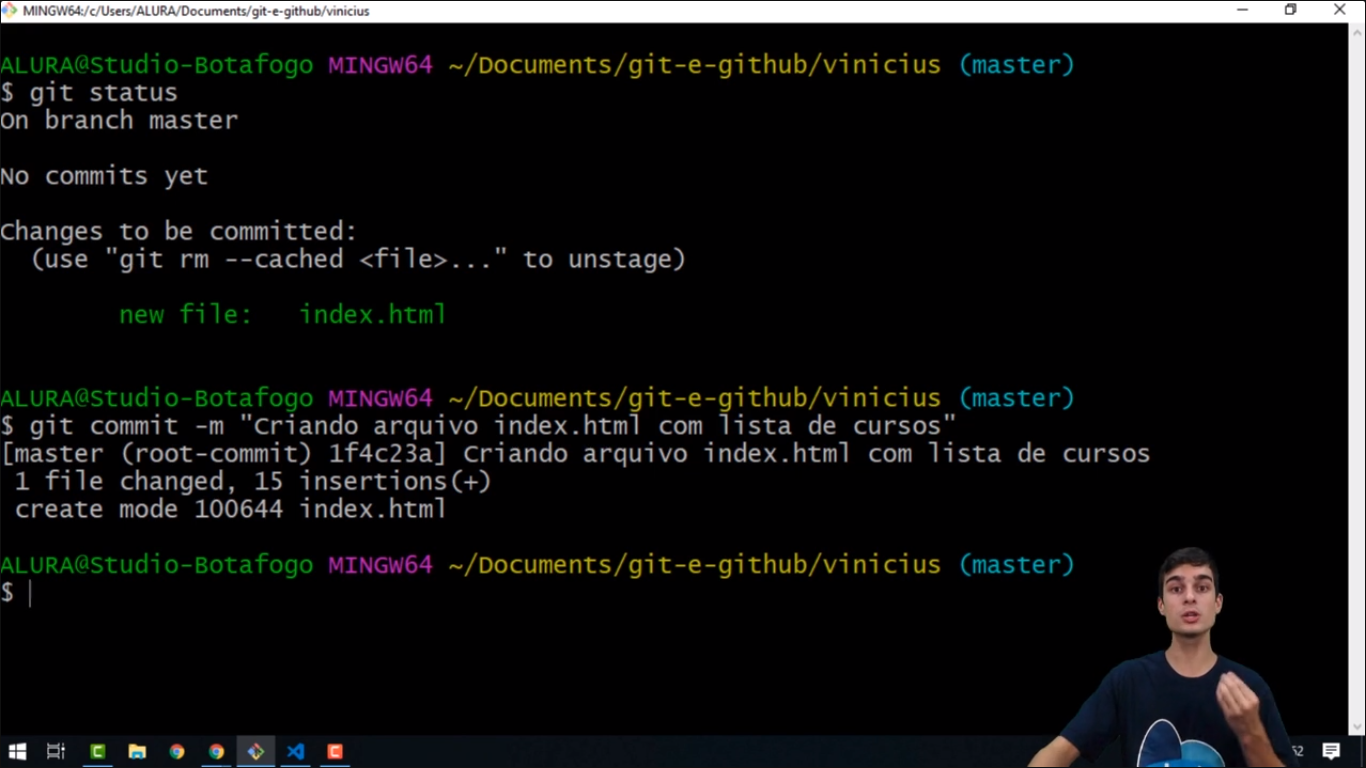
git config --local user.email "seu@email.aqui"

git add index.js – para monitora qualquer mudança nesse arquivo

git add . – para monitora todos os arquivo nesse caminho

git rm --cacher <arquivo> - para cancelar o monitoramento de um arquivo

git commit -m “primeiro commit” – enviar para área de comitado



git log – lista os commits feitos um históricos

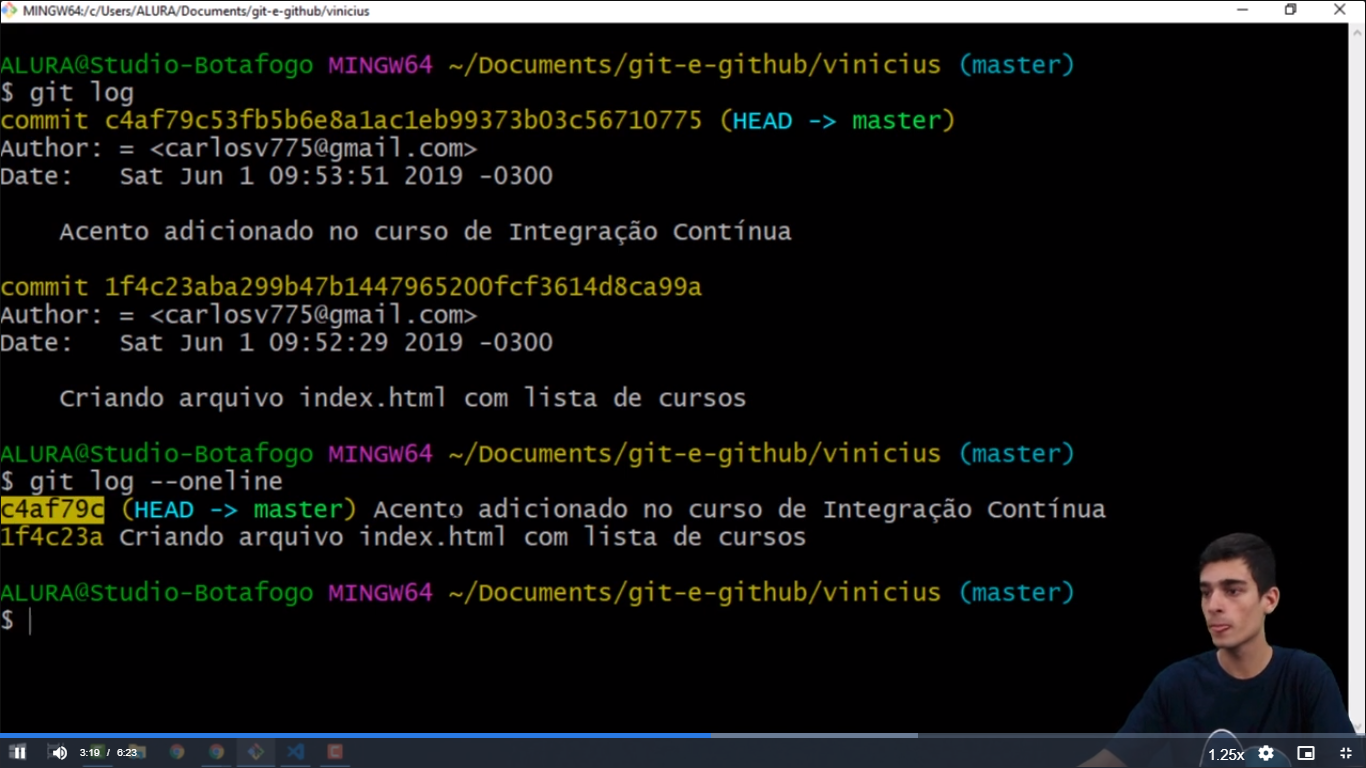
git config --local – para configurar somente para esse projeto

git config --global – uma configuração para todos os projetos

git config --local user.name “rodrigo de souza” – configura a configuração desse projeto como nome do usuário sendo rodrigo de Souza

git config user.name – para verificar qual o nome está no projeto

git log --oneline – mostra o log resumido em uma linha mostrando apenas o hash o branch que está e sua mensagem



git log -p – mostra as informações de cada log mais detalhada

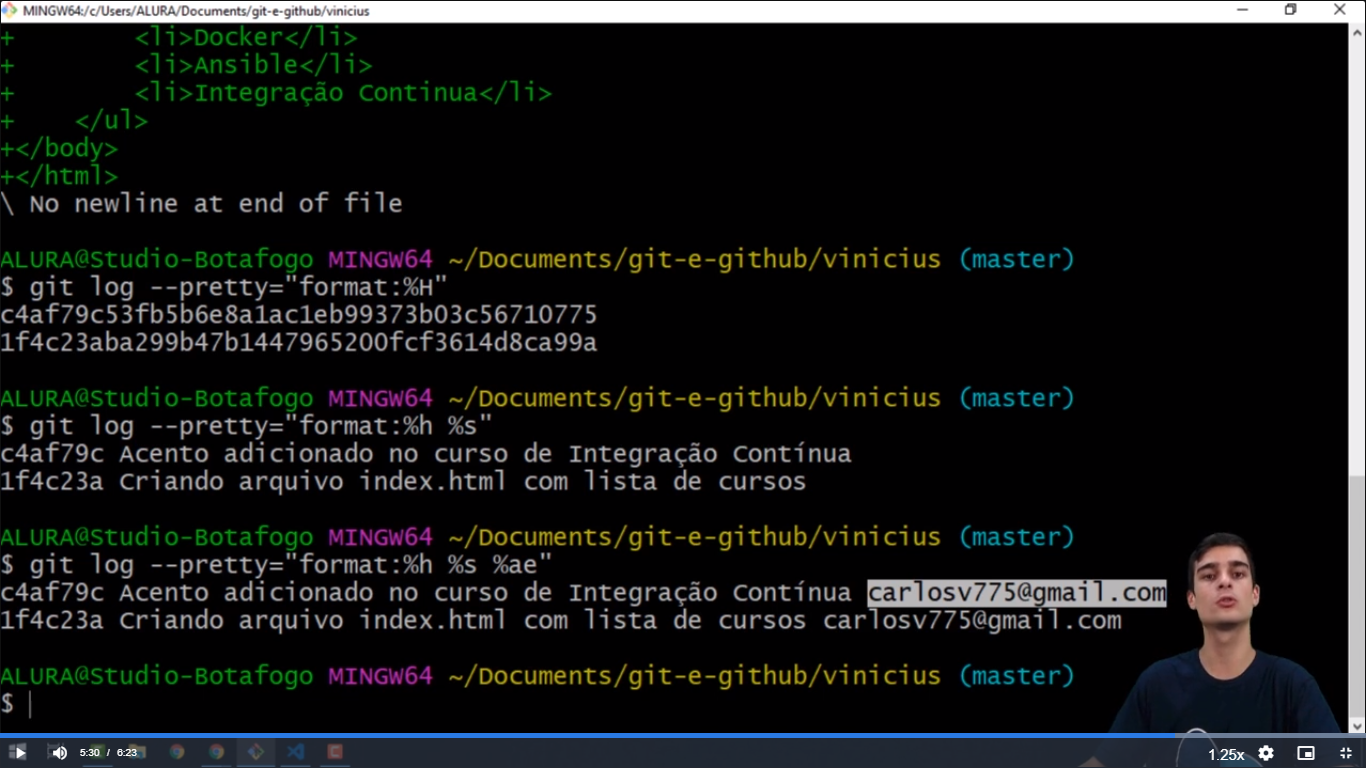
link para algumas opções que podem ser usada para mostrar o log

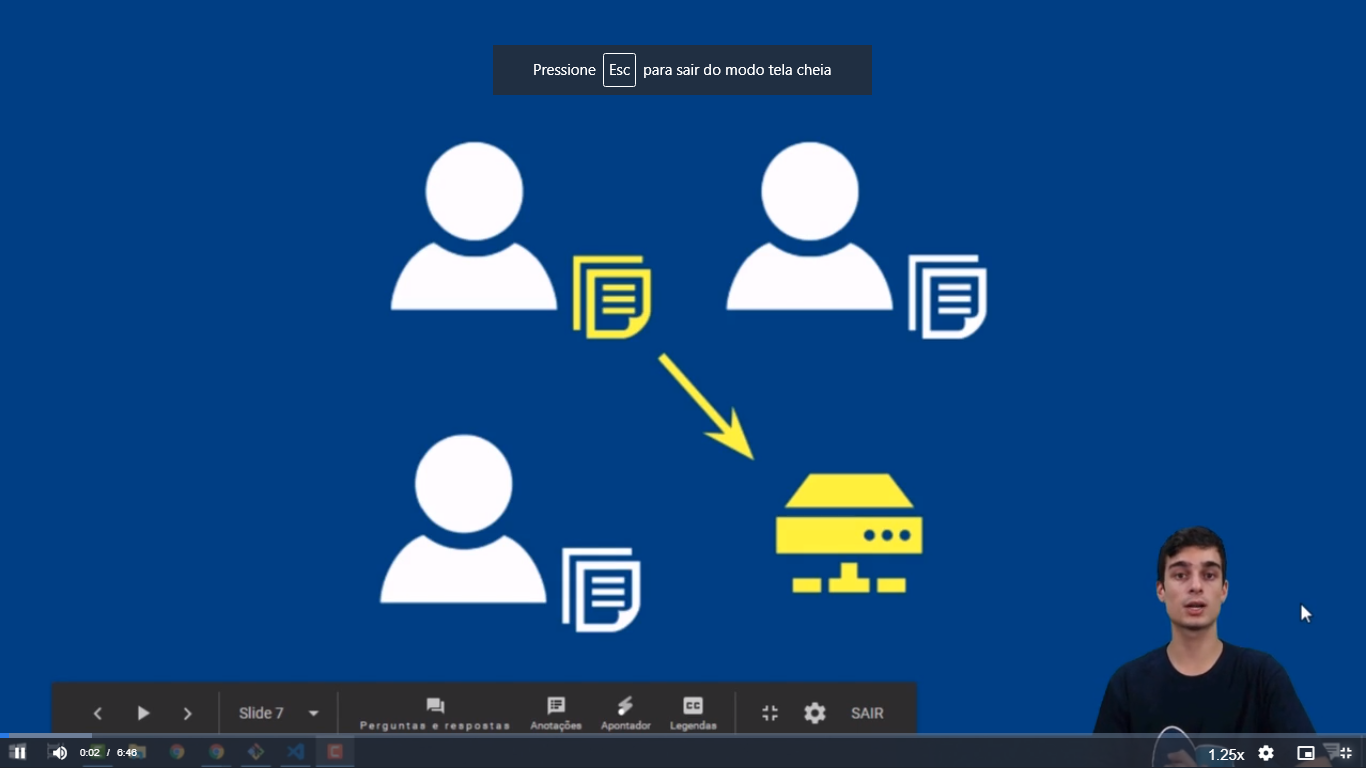
<https://devhints.io/git-log>.

git log --pretty=”format:%H” – para formata o log mostra só o hash



git log --pretty=”format: %h %s %ae” – mostrar no log a hash mensagem e o autor no log





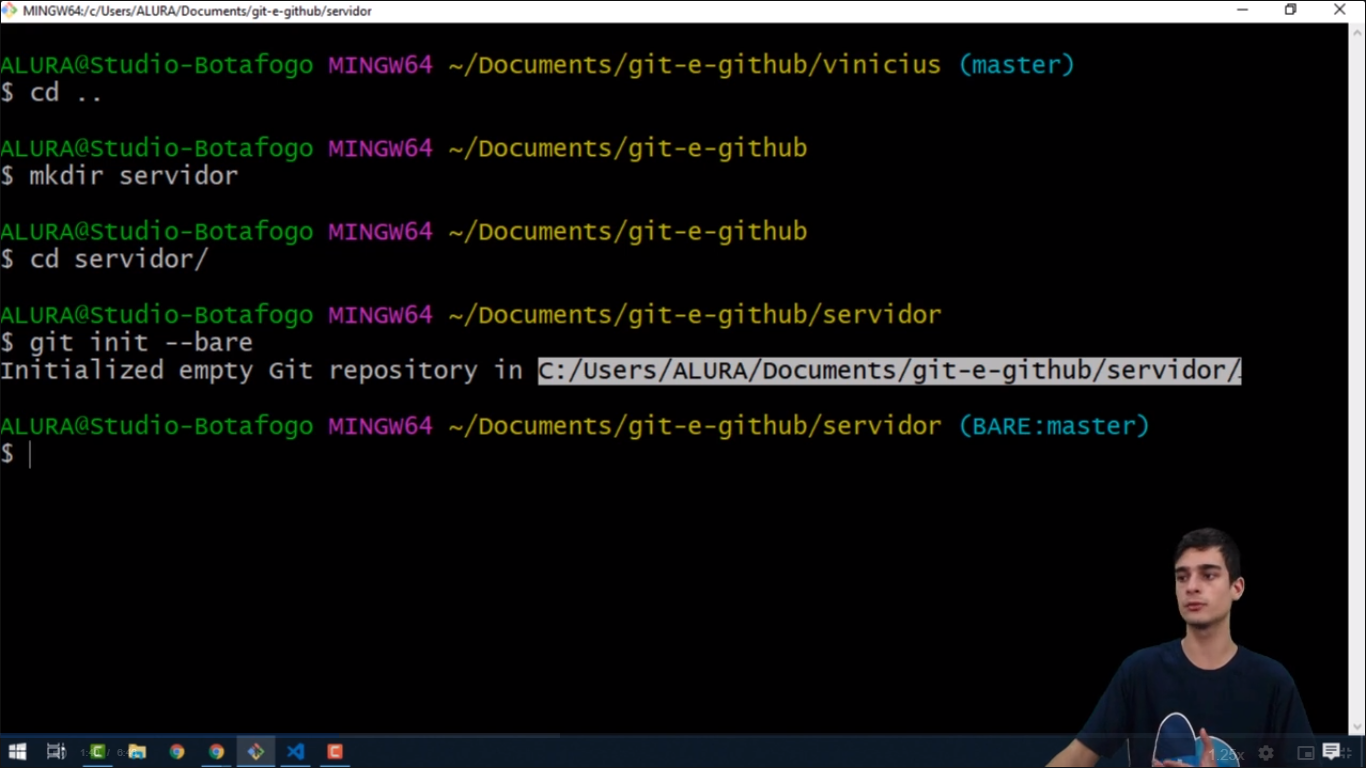
mk dir servidor – criado no caminho /Documents/git-e-github

irá criar uma pasta servidor nesse caminho

no /Documents/git-e-github/servidor foi feito um init

git init --bare – esse tipo de init significa que esse repositório só vai servir para armazenar informações

git config core.bare true – caso tenha esquecido o --bare



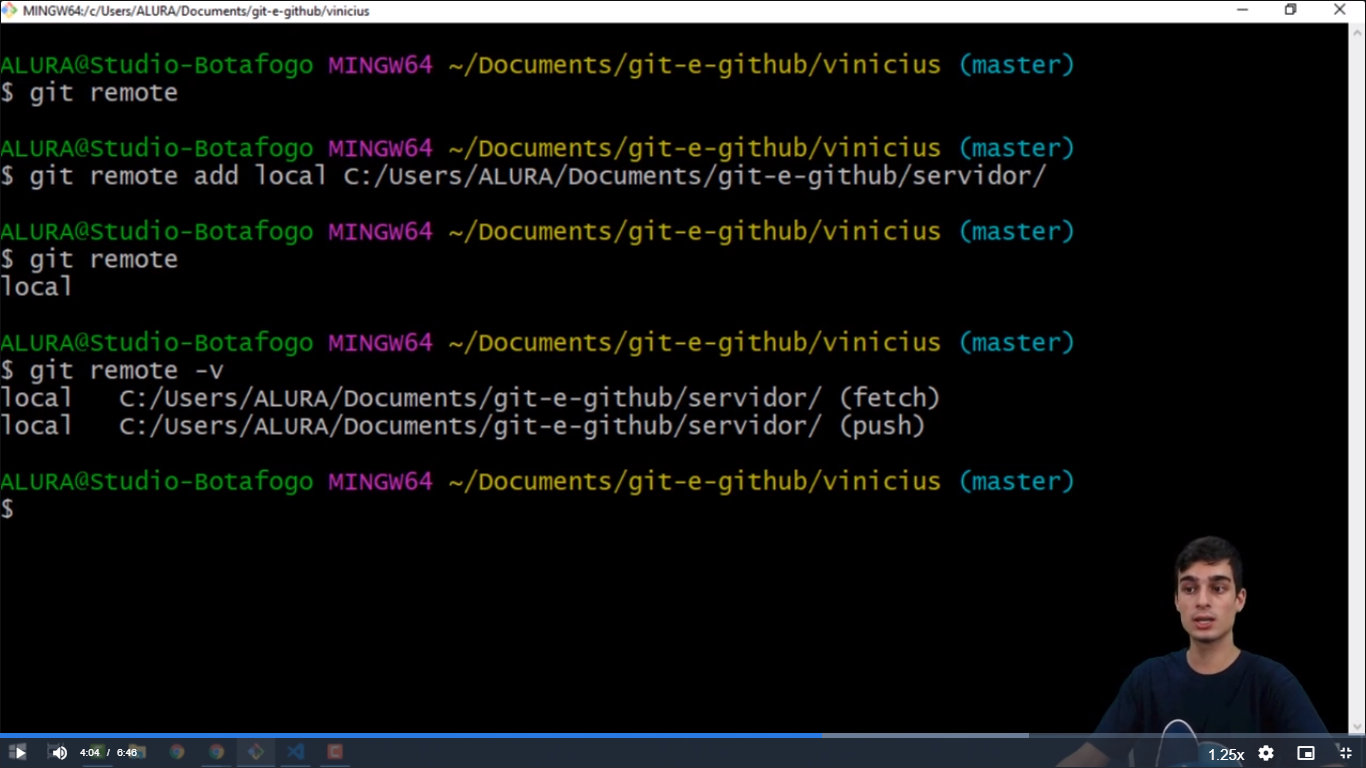
Voltando para o caminho do Vinicius

git remote – lista todos os repositório remotos que o usuário local conhece

git remote add <nome\_repositorio\_remoto> <caminho\_para\_esse\_repositorio> - cria um novo repositório remoto

git remote add local c:/users/alura/documents/git-e-github/servidor/ - foi criado um repositório local chamado local para o caminho indicado

git remote -v – mostra os endereços dos repositório remoto



Fetch – buscar dados nesse caminho

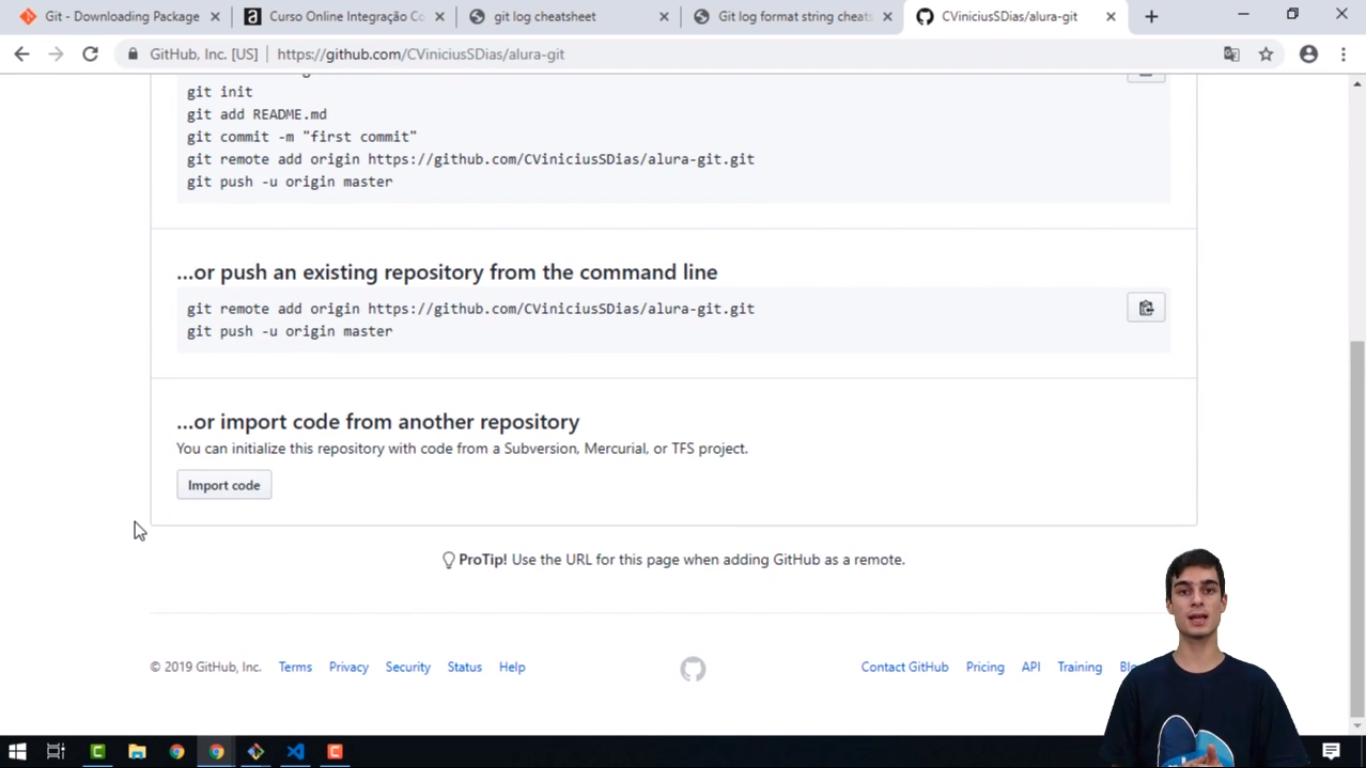
Push – enviar dados para esse caminho

mkdir ana – criado uma pasta para ana

documents/git-e-github/ana

git clone <caminho\_repositorio> - para copiar e clonar um repositório

git clone c:/users/alura/documents/git-e-github/servidor/ projeto - ira clonar os dados que estão nesse repositório com o nome projeto caso esse último parâmetro não existisse, esse projeto iria se chamar servidor



Modelo como criar um repositório local no github no modelo https, primeiro modelo caso não tenha um iniciado o git init e o segundo caso já tenha feito o git init

git push -u origin master – Quando se coloca o -u ao fazer um push ele indica que toda a vez que você for fazer um push em master ele será enviado para o origin

git pull origin master – para puxar as informações de um repositório

git remote rename origin local – alterar o nome do repositório de origin para local

git branch – mostra as ramificações do seu projeto

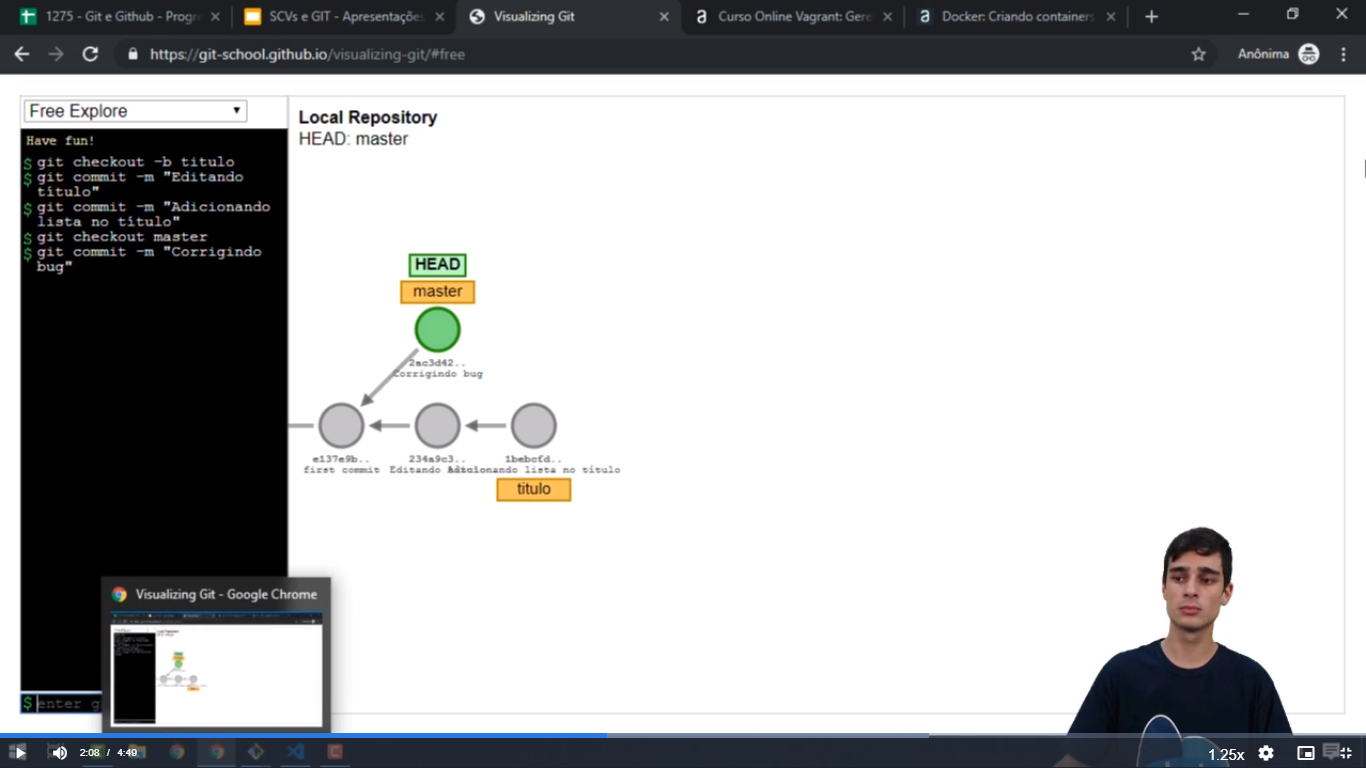
git branch titulo – cria um novo branch, uma nova ramificação

git checkout titulo – muda para um novo branch

ferramenta para visualizar o comportamento de seu git

https://git-school.github.io/visualizing-git/

git checkout -b lista – para já criar seu novo branch e mudar a ramificação para ele

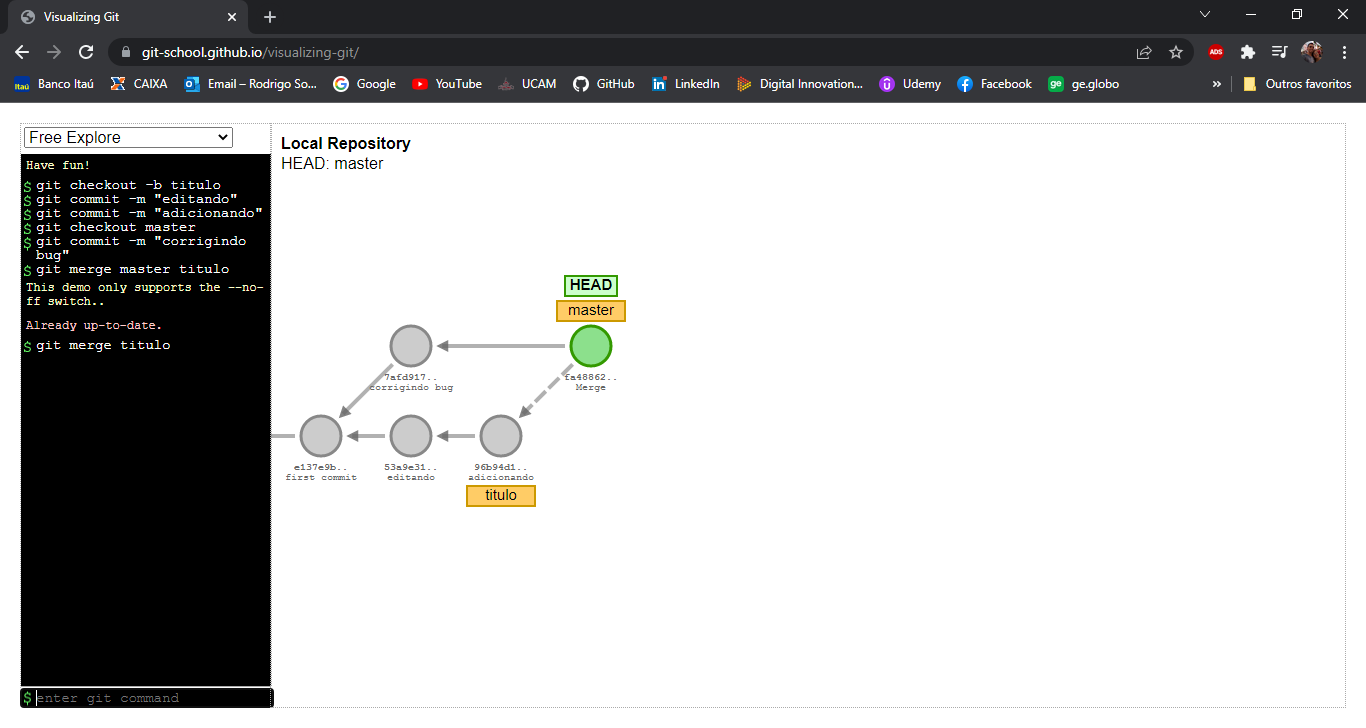


Agora iremos aprender a juntar essas duas ramificações

git merge titulo – irá juntar o branch atual(nesse caso a master) com o titulo

feito isso ele dará uma mensagem, para confirmar basta:

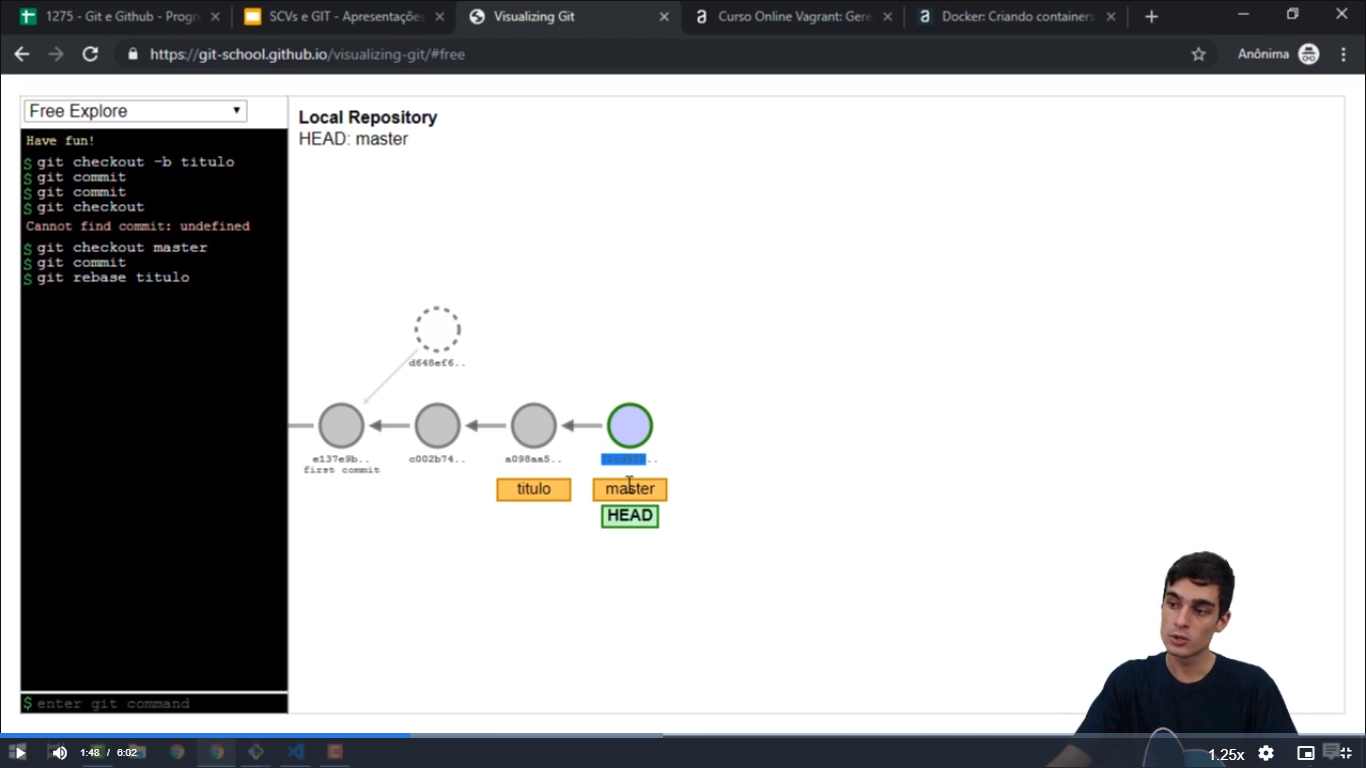
aperta “:” e “X “e da Enter



Ele criar um novo commit contendo todo o conteúdo da branch de titulo

Assim você estará unindo duas branch, duas linhas de desenvolvimento

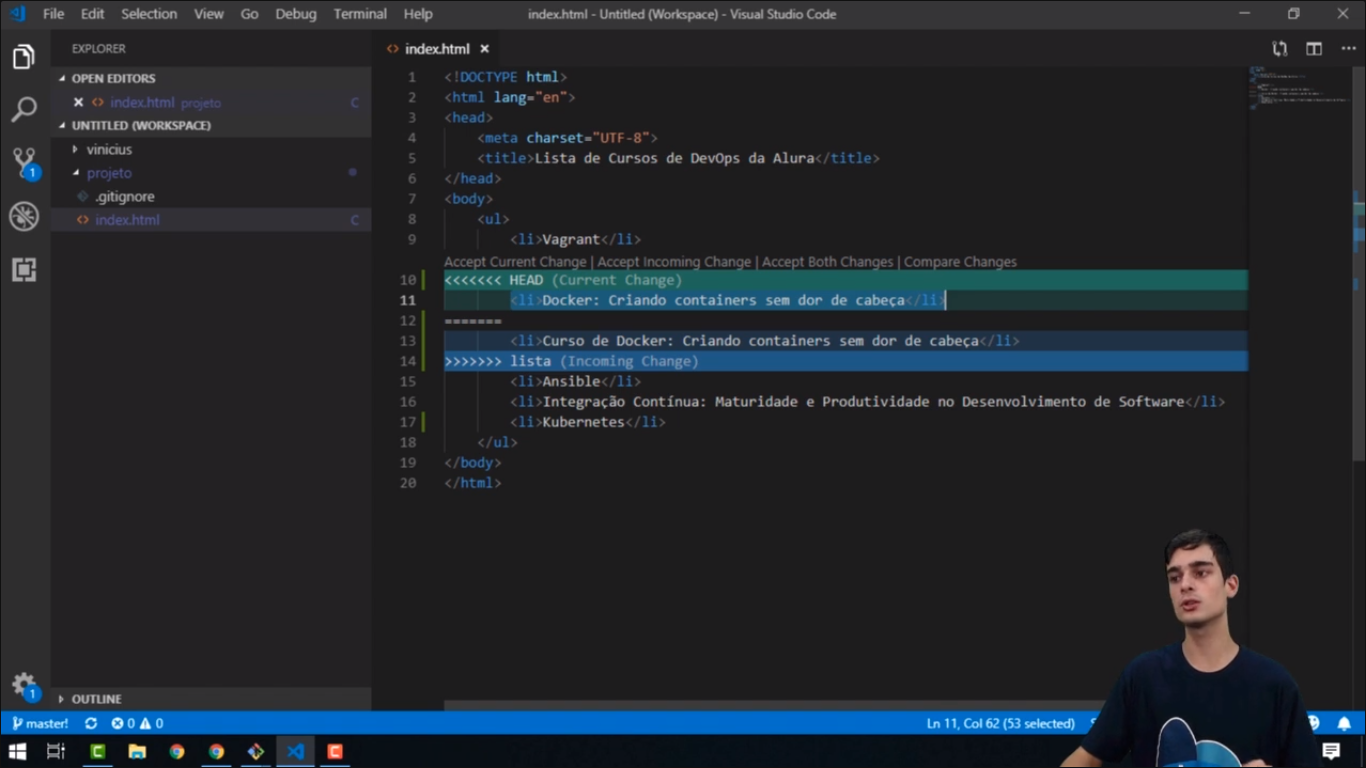
git rebase titulo – o git ira pegar os commit da branch titulo a atualizar na branch master (isso estando com o HEAD em master), assim ela terá todos os commit que tinham na branch titulo



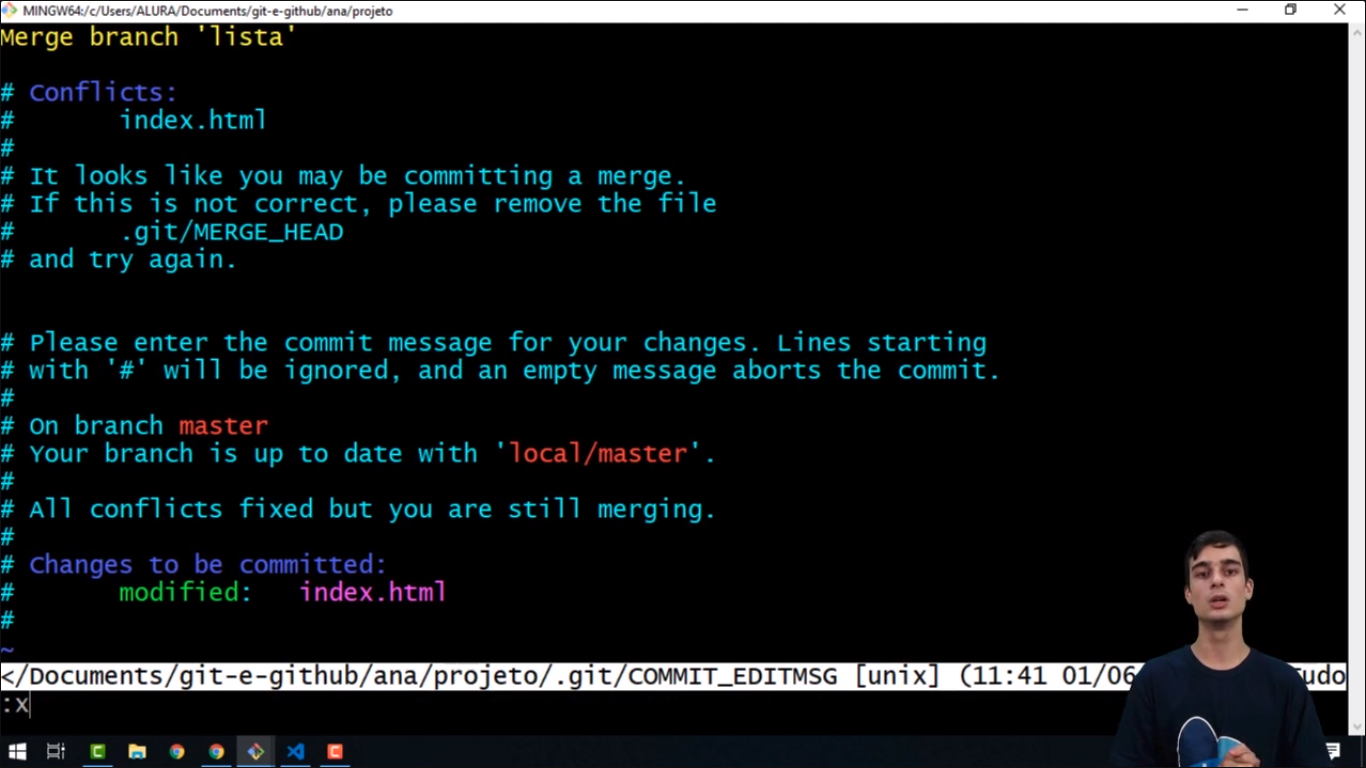
git log --graph – mostra o gráfico de ramificações e sua linha de commit

O merge junta os trabalhos e gera um merge commit. O rebase aplica os *commits* de outra *branch* na *branch* atual.

Caso o corra um conflito de informação(quando uma pessoa já tenha alterado um código que você está alterando e salvando) ao fazer o rebase, o seu editor de código ira mostrar as duas opções que foram alteradas, basta você escolher qual ficou melhor e apagar a que não gostou, feito isso basta da um git add ., feito isso basta fazer o git commit, para informar que o erro já pode ser comitado git retornará a mensagem para sair dela aperte :X



Exemplo de como e mostrada no editor de código, onde ah um conflito de qual escolher, bastando apenas apagar a que menos gostar



Exemplo de como e mostrada a tela com a mensagem

Sequência para arrumar o rebase ou merge

git merge lista

abre o editor e arrumar os conflitos

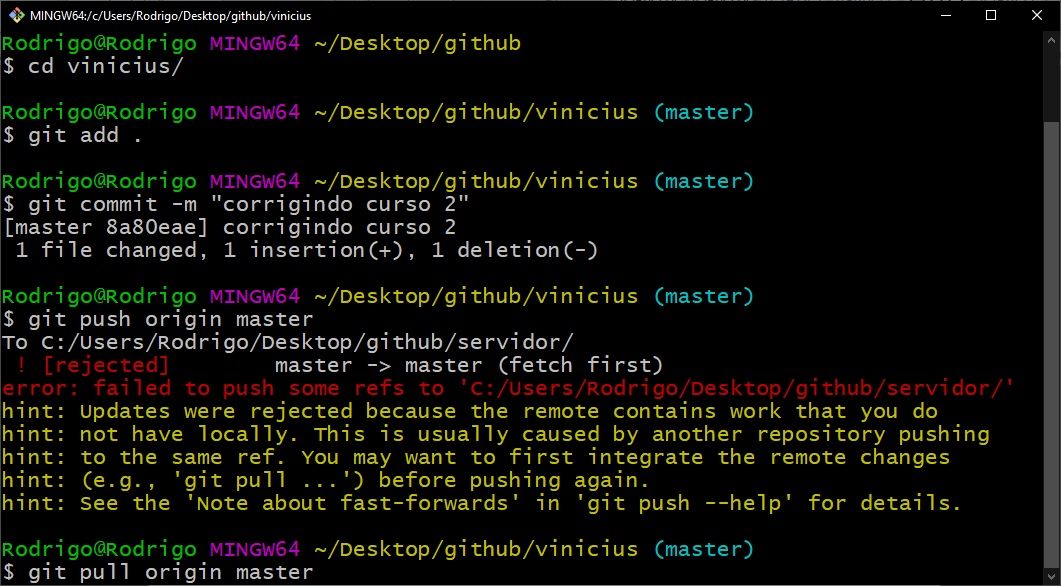
git add <nome\_arquivo>

git commit

:x

Git push origin master

Ao ir para outra branch e necessário trazer do servidor a versão mais nova do projeto, fizemos o rebase com a Ana ao usar a conta do Vinicius temos que atualizar nosso projeto, isso tudo antes de enviar qualquer informação



Por isso e muito importante quando se trabalha com equipe que fazem rebase, ao iniciar um projeto rode o código de pull para deixar seu código sempre atualizado

git checkout -- <arquivo> - para descartar uma alteração feita no código

git checkout -- index.html – descarta a alteração feita no index, caso tenha feito uma alteração no código, a mesma será desfeita

git restore --staged index.html – para voltar seu git na posição anterior ao git add, as alterações continuarão lá mais não estão marcada para commit(usada para depois que tenha feito o git add .)

git revert def3cd1c2ffcbbf1dd42b9e2e86c6a – desfaz o commit que foi informado pelo hash, vai criar um novo commit desfazendo o commit que ele tinha feito

1 - git checkout

2 - git reset

3 - git revert

Alternativa correta! Com o git checkout nós desfazemos uma alteração que ainda não foi adicionada ao index ou stage, ou seja, antes do git add. Depois de adicionar com git add, para desfazer uma alteração, precisamos tirá-la deste estado, com git reset. Agora, se já realizamos o commit, o comando git revert pode nos salvar.

git stash - Salva todas as alterações para um local temporário, onde possa recuperar depois onde não precise fazer um commit

git stash list – lista de tudo que está salvo, cada stash fica salvo com um numero que representa sua posição

git stash apply 0 – método para pegar o que foi salvo na posição passada, dessa maneira a stash não e apagada da área temporária

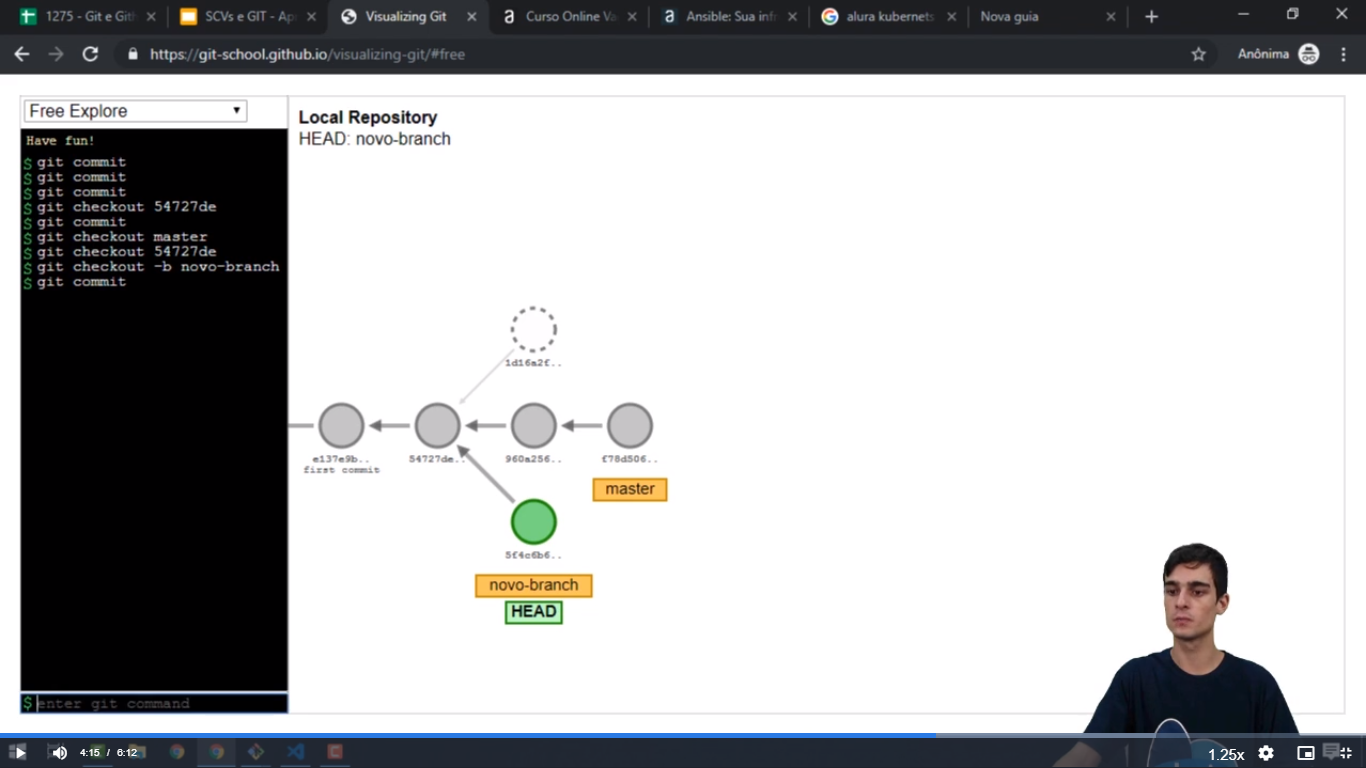
git stash drop – para remover o stash da área temporária

git stash pop – dessa maneira ele pega o último valor adicionada na área temporária e já exclua ela de lá também

git checkout 4eb5afc – dessa maneira seu código ira se descolar do caminho que estava seguindo e pegar uma outra “estrada” que terá uma branch como o nome da hash escolhida

tudo o que for feito a partir desse momento não ficara salva a não ser que você crie um novo branch

git checkout -b novo-branch – agora suas alterações se manterão no mesmo caminho



Para voltar como era só escolher a branch principal do projeto:

git checkout master

git diff ea59987..u55621t – para mostrar a diferença de um commit até outro commit, mostra os códigos que foram alterados

git diff – quando não adicionado, não passado nenhuma hash, ele mostra o que está diferente, mostra o que foi modificado

git log -n 2 – mostra os últimos 2 commits

git tag -a <nome\_tag> -m <mensagem\_tag> - para gerar uma tag um marco na sua aplicação, um ponto que não pode ser modificado, um ponto fixo

qualquer alteração será adicionada na próxima versão

git tag -a v0.1.0 -m “lançando a primeira versão da aplicação”

git tag – Mostra todas as tags disponíveis

git push origin v0.1.0 – enviando para o github, que pode ser visualizado em releases(versão pronta)