

Conceitos básicos

De



GITHUB
GITHUB
GITHUB

Execute Devs

Dedicatória

Nossa missão aqui na Execute Dev é trazer para você uma maneira de pensar como podemos utilizar qualquer ferramenta para nosso dia a dia, seja para trabalho ou até mesmo de forma casual, espero que entenda como essa ferramenta pode utilizada, assim podendo usa-la no seu dia a dia agregando o máximo de valor possível. Desejo oferecer a você não apenas um aprendizado de como utilizar uma das ferramentas de maior destaque no mercado de trabalho.

Desejo que você possa analisar e tomar suas próprias decisões para utilização da ferramenta, mostrarei a você que qualquer ferramenta que esteja disponível para uso, ela pode agregar total valor para sua necessidade, seja em no trabalho ou de forma casual. Até mesmo de forma distinta com um objetivo diferente de projeto.

Git e GitHub são conceitos e ferramentas utilizadas para versionamento de arquivos, mas existe outro ponto que essa ferramenta traz consigo, o valor que pode agregar ao trabalho dos desenvolvedores. Uma maneira de centralizar seus projetos e atividades, sendo assim, podendo efetuar versionamento de arquivos contendo versões. GitHub mostra ser uma ferramenta flexível para qualquer necessidade. Não apenas como uma ferramenta de versionamento, GitHub consegue trazer consigo diversas ferramentas como montar um Kanban para tarefas que devam ser desenvolvidas, seja de projeto ou até mesmo um trabalho de faculdade, podemos efetuar hospedagem de sites dentre outras ferramentas disponíveis no GitHub.



Sumário

Dedicatória	1
GitHub & Git	3
Git	3
GitHub	4
Git & GitHub	5
Requisitos	6
GitHub e Conceitos	11
Conceito	12
Capítulo 1	13
Primeiro passo	13
Capítulo 2	16
Code	16
Issues	17
Pull requests	18
Actions	19
Projects	20
Wiki	21
Security	22
Insights	23
Settings	24
Capítulo 3	25
Code	25
Iniciando seu repositório	26
Git Bash	27
Capítulo 4	34
Criando Branch	34
Atualizando Branch	36
Capítulo 5	40
Capítulo 6	47

GitHub & Git

Para iniciarmos nosso treinamento na ferramenta GitHub primeiramente precisamos saber brevemente a diferença entre Git e GitHub, assim conhecendo sua história e principalmente porque está é uma das ferramentas com um valor tão grande no mercado de trabalho e fora do mercado de trabalho também.

Git

Como muitas coisas boas na vida, Git começou com um pouco de destruição criativa e controvérsia ardente.

O kernel Linux é um projeto de software de código aberto de escopo bastante amplo. Durante a maior parte da vida útil da manutenção do kernel do Linux (1991–2002), as alterações no software foram passadas como patches e arquivos arquivados. Em 2002, o projeto do kernel Linux começou a usar um DVCS proprietário chamado BitKeeper.

Em 2005, o relacionamento entre a comunidade que desenvolveu o kernel Linux e a empresa comercial que desenvolveu o BitKeeper foi interrompido, e o status de gratuito da ferramenta foi revogado. Isso levou a comunidade de desenvolvimento do Linux (e em particular Linus Torvalds, o criador do Linux) a desenvolver sua própria ferramenta com base em algumas das lições que aprenderam ao usar o BitKeeper. Alguns dos objetivos do novo sistema eram os seguintes:

- Rapidez
- Design simples
- Suporte forte para desenvolvimento não linear (milhares de ramificações paralelas)
- Totalmente distribuído
- Capaz de lidar com grandes projetos como o kernel Linux de forma eficiente (velocidade e tamanho dos dados)

Desde seu nascimento em 2005, o Git evoluiu e amadureceu para ser fácil de usar e ainda assim manter essas qualidades iniciais. É incrivelmente rápido, muito eficiente com grandes projetos e possui um sistema incrível de ramificação para desenvolvimento não linear (consulte Ramificação Git).



GitHub

O GitHub foi lançado no ano de 2008 pelos desenvolvedores da Logical Awesome e é um sistema Web Hosting compartilhado.

O GitHub nasceu com o principal objetivo de abrigar projetos que são versionados via Git.

O GitHub foi escrito em Ruby on Rails, o que é bem interessante pelo fato de na época a linguagem/framework possuir pouco tempo de vida e estabilidade. A medida que a popularidade foi aumentando, percebeu-se que seria necessário a implementação de funcionalidades que deixassem o GitHub com cara de rede social. Foram adicionadas listas de discussões, gráficos referentes a contribuições em projetos e até mesmo a opção de seguir algum usuário, algo muito parecido com o que o Twitter emprega. Atualmente o GitHub é tão reconhecido na comunidade que projetos de grande importância estão desfrutam dos seus serviços, como por exemplo o Linux e o Ruby on Rails.



Git & GitHub

A partir deste momento iremos dar início ao curso com foco no GitHub, nosso objetivo é de ensiná-lo há como utilizar o Git e GitHub da melhor forma possível agregando valor a ferramenta. Lembre-se que uma ferramenta bem desenvolvida consiste em dois conceitos.

- Custo
- Valor

Quando nos referimos a custo, toda ferramenta tem o seu capital que precisa ser injetado/aplicado para ser desenvolvido. Até mesmo para usufruir de uma ferramenta existente precisamos aplicar um capital.

Mas quando falamos de valor estamos referindo-se o que aquela ferramenta irá agregar no seu dia a dia, em um trabalho, projeto ou até mesmo para usá-lo de maneira casual (Guardar documentação/Servidor de game).

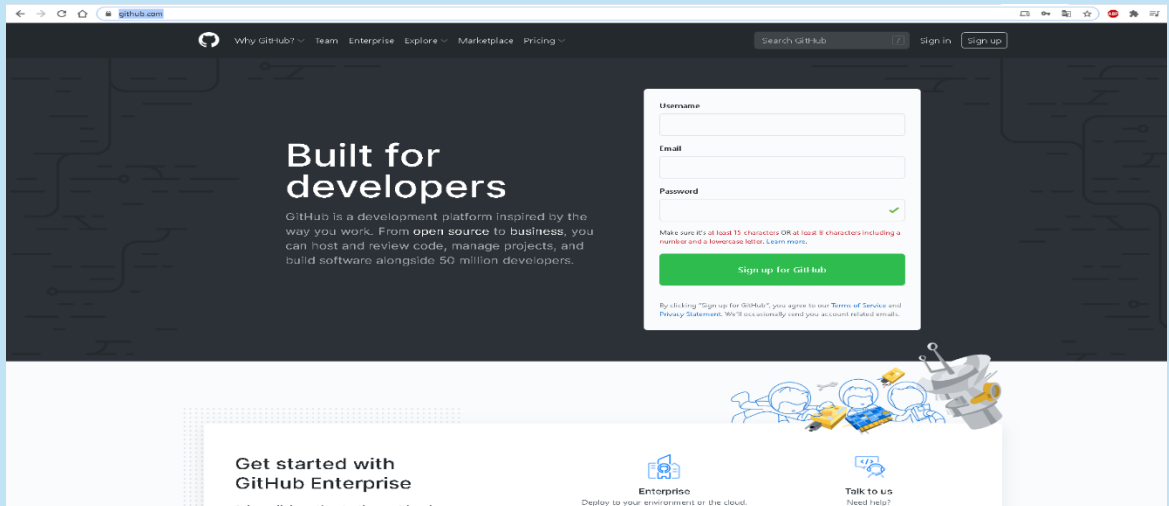
Gostaria de deixar claro que podemos usar ferramentas para nosso dia a dia de forma inimagináveis então ao concluir este curso espero que consiga aproveitar essa ferramenta e usá-la da melhor forma possível.... Conte-me depois como foi sua experiência.



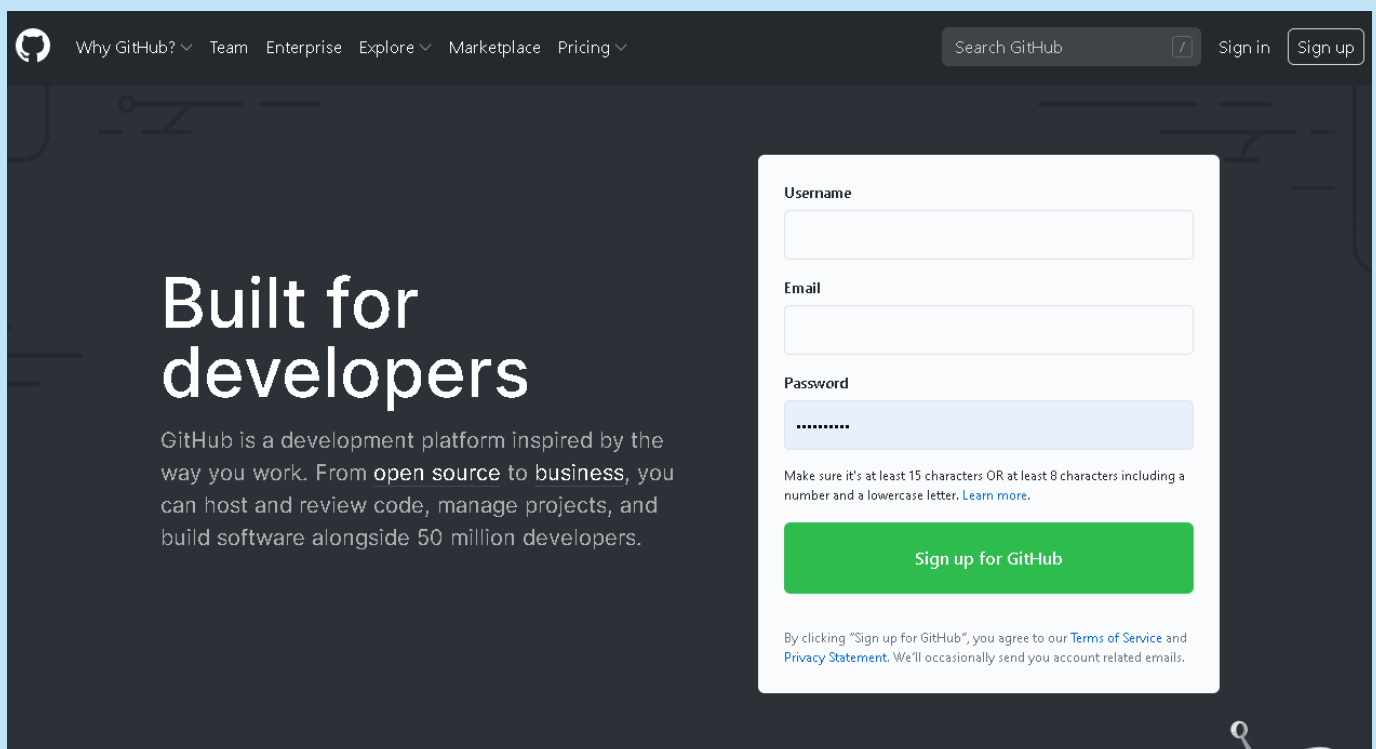
Requisitos

Para poder utilizar a ferramenta primeiramente vamos precisar criar uma conta, siga os passos a passos e crie sua conta.

1. Acesse o site: <https://github.com/>



2. Ao entrar na página irá se deparar com o site em inglês nela poderá clicar no botão **Sign Up** (Localizado ao lado direito superior na tela) ou bem no início da página poderá criar sua conta.



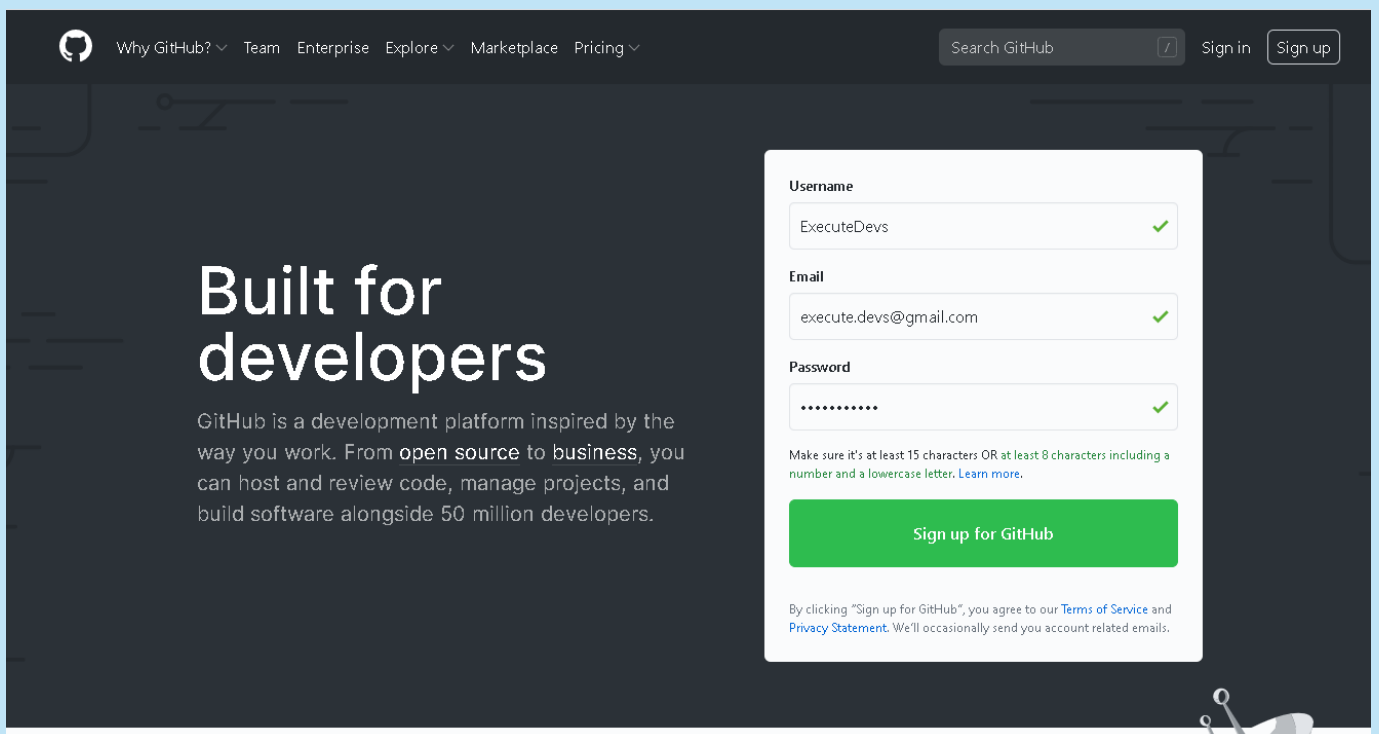
3. Preencha as informações necessárias para o cadastro do login.

Username: desta forma as pessoas podem enviar convites para você pelo seu nome cadastrado

E-mail: recomenda-se um e-mail profissional até mesmo para poder compartilhar seu projeto via e-mail para empresas que necessitem ver seu portfólio.

Password: Crie uma senha para seu cadastro sempre que for configurar ou até mesmo acessar o GitHub estará protegido.

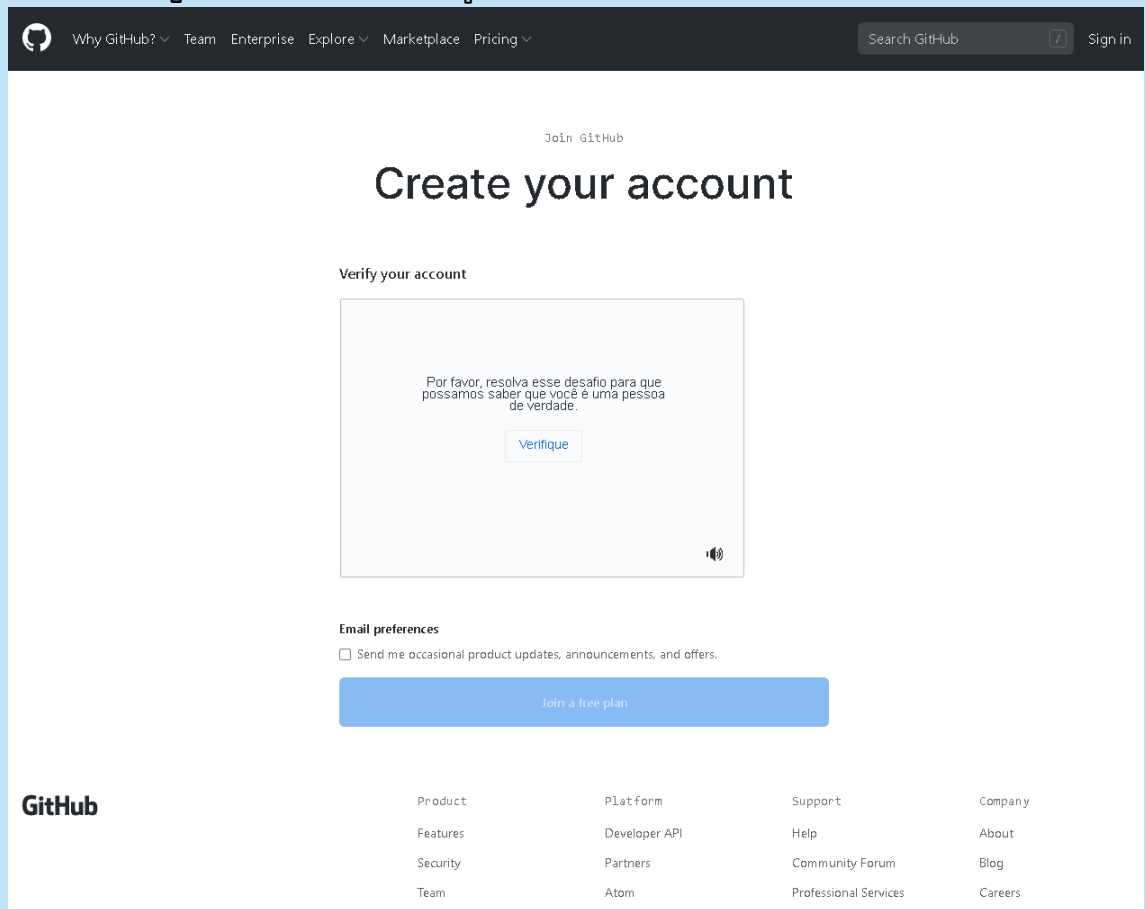
4. Confirmação de cadastro:



The screenshot shows the GitHub sign-up page. On the left, the text "Built for developers" is prominently displayed, followed by a description of GitHub as a development platform. On the right, a white registration form is overlaid on a dark background. The form contains three input fields: "Username" with the value "ExecuteDevs", "Email" with the value "execute.devs@gmail.com", and "Password" with masked characters. Each field has a green checkmark to its right, indicating it is valid. Below the password field, there is a note about password requirements: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)". A large green button labeled "Sign up for GitHub" is positioned below the form. At the bottom of the form, a small disclaimer states: "By clicking 'Sign up for GitHub', you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails." The top of the page features the GitHub navigation bar with links like "Why GitHub?", "Team", "Enterprise", "Explore", "Marketplace", and "Pricing", along with a search bar and "Sign in" / "Sign up" buttons.



5. Após clicar em **Sign up for GitHub** ele tentará validar se você não é um robô com algum método de validação:



The screenshot shows the GitHub 'Create your account' page. At the top, there's a navigation bar with links like 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. A search bar and a 'Sign in' link are on the right. The main heading is 'Create your account'. Below it, a section titled 'Verify your account' contains a challenge box. The box has the text: 'Por favor, resolva esse desafio para que possamos saber que você é uma pessoa de verdade.' and a 'Verifique' button. Below the challenge box, there's an 'Email preferences' section with a checkbox 'Send me occasional product updates, announcements, and offers.' and a 'Join a free plan' button. At the bottom, there's a footer with the GitHub logo and a grid of links categorized by Product, Platform, Support, and Company.

Join GitHub

Create your account

Verify your account

Por favor, resolva esse desafio para que possamos saber que você é uma pessoa de verdade.

Verifique

Email preferences

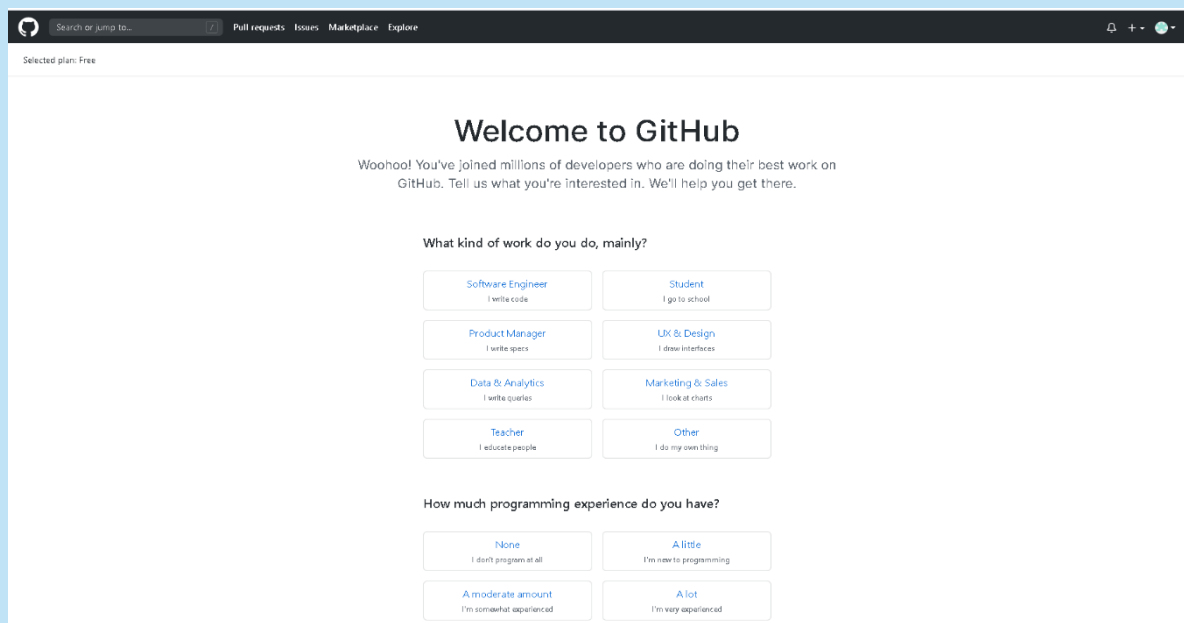
☐ Send me occasional product updates, announcements, and offers.

Join a free plan

GitHub

Product	Platform	Support	Company
Features	Developer API	Help	About
Security	Partners	Community Forum	Blog
Team	Atom	Professional Services	Careers

6. Últimos passos o GitHub irá questioná-lo de como você irá utilizar a ferramenta e para qual propósito, responda as questões e escreva o que melhor lhe atende.



The screenshot shows the GitHub 'Welcome to GitHub' onboarding page. At the top, there's a navigation bar with links like 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. A search bar and a 'Selected plan: Free' indicator are on the left. The main heading is 'Welcome to GitHub'. Below it, there's a message: 'Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.' The page is divided into two main sections: 'What kind of work do you do, mainly?' and 'How much programming experience do you have?'. Each section has several buttons with icons and text. The first section has buttons for 'Software Engineer', 'Student', 'Product Manager', 'UX & Design', 'Data & Analytics', 'Marketing & Sales', 'Teacher', and 'Other'. The second section has buttons for 'None', 'A little', 'A moderate amount', and 'A lot'.

Search or jump to... Pull requests Issues Marketplace Explore

Selected plan: Free

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

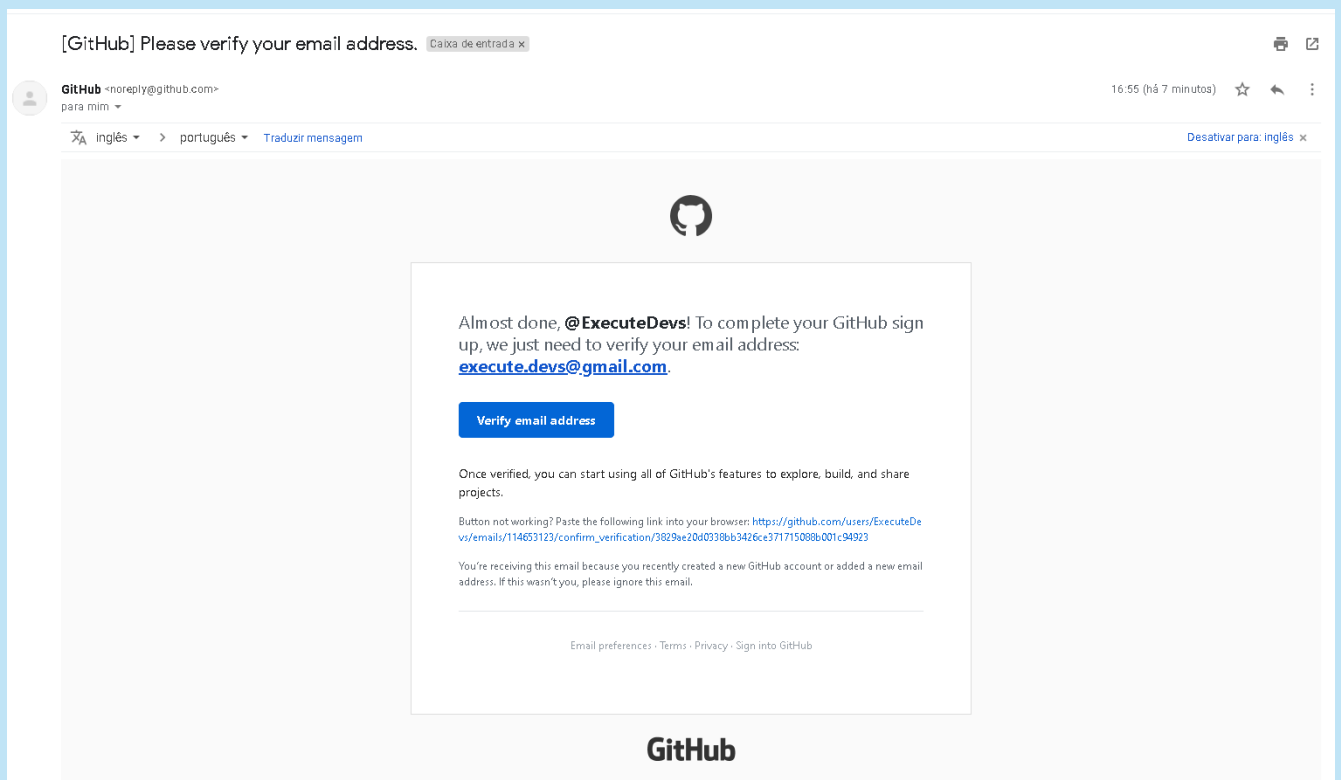
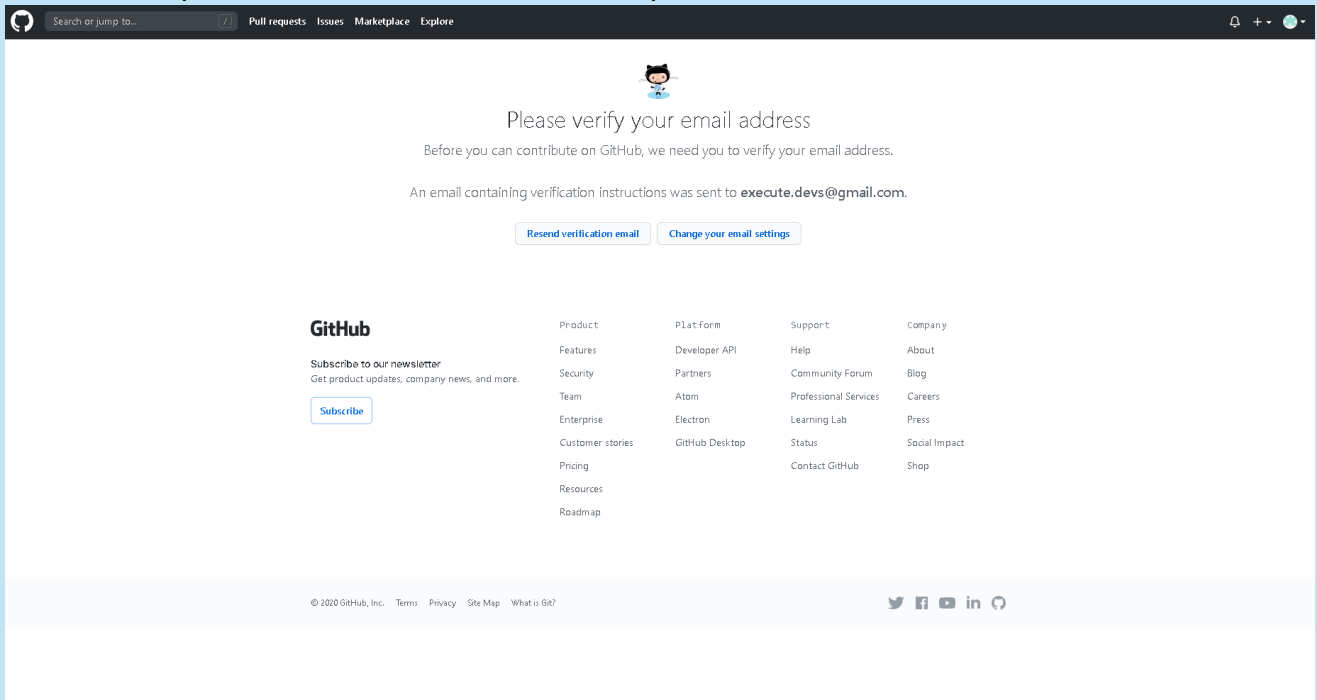
Software Engineer I write code	Student I go to school
Product Manager I write specs	UX & Design I draw interfaces
Data & Analytics I write queries	Marketing & Sales I look at charts
Teacher I educate people	Other I do my own thing

How much programming experience do you have?

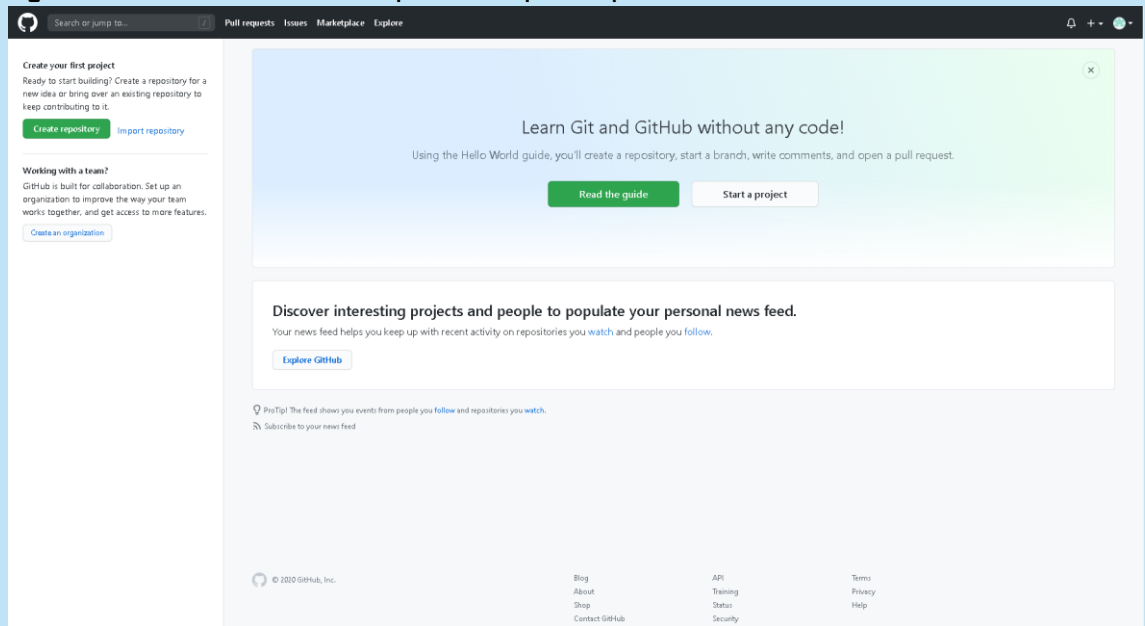
None I don't program at all	A little I'm new to programming
A moderate amount I'm somewhat experienced	A lot I'm very experienced



7. Por último o GitHub irá precisar que você confirme em seu e-mail a conta para utilizar da melhor forma, cheque o e-mail cadastrado.

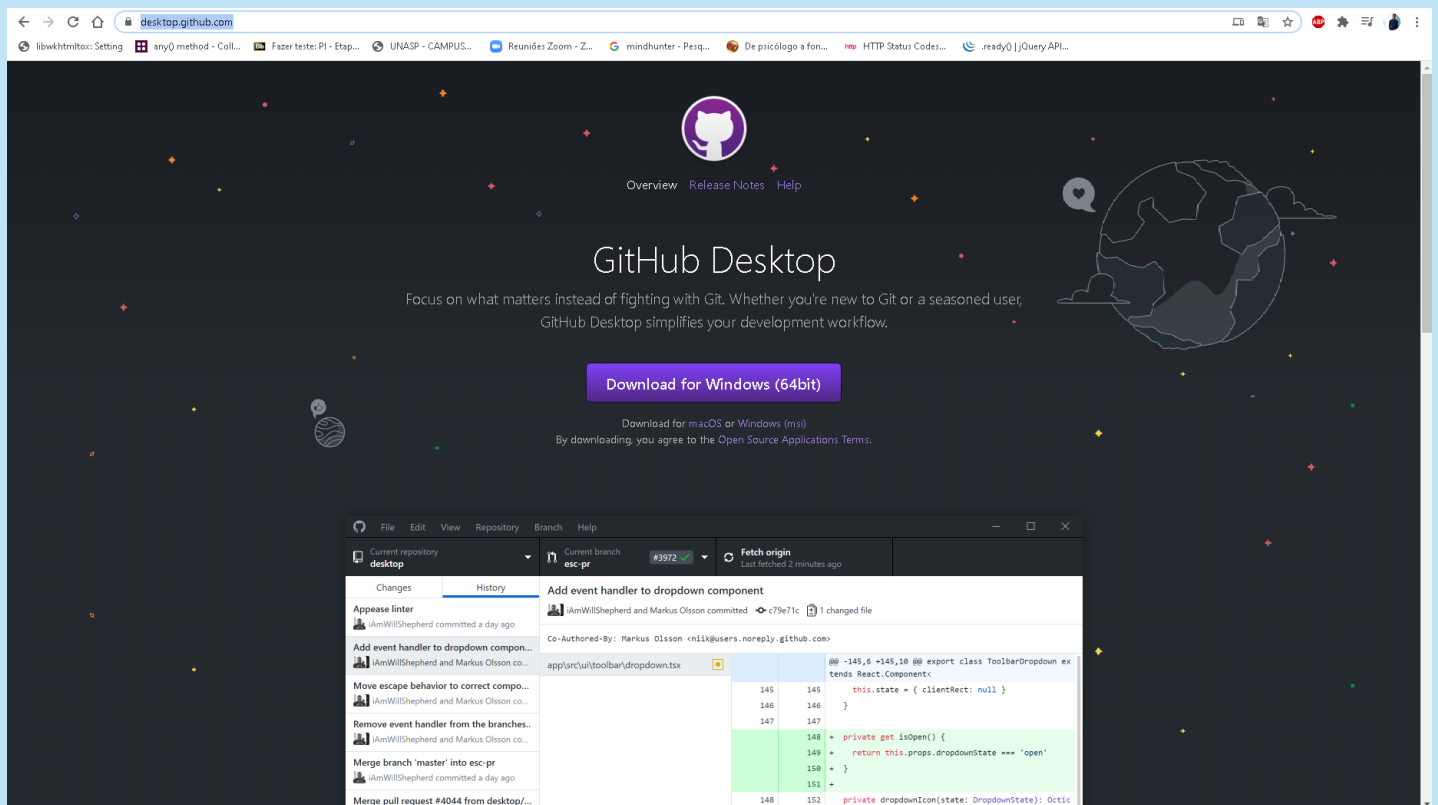


8. Agora com sua conta criada podemos partir para



9. Baixe o GitHub para utilização no DeskTop

Link: <https://desktop.github.com/>



10. Faça a instalação de forma convencional



GitHub e Conceitos

Quando utilizamos o GitHub na maioria dos casos as pessoas pensam em apenas fazer controle de versionamento de arquivo... Ou seja estamos falando de deixar projetos disponíveis e atualizados para serem manipulados em outras máquinas ou servidores.

Mas quando falamos em versionamento de arquivos não podemos apenas nos apegar que está ferramenta tem apenas este serviços para oferecer, podemos hospedar páginas HTML estáticas, contendo CSS, também JS, pode-se dizer que podemos fazer um portfólio para que você possa usar de forma gratuita, para outras pessoas acessarem e analisem o seu grau de conhecimento a respeito de Web.

Páginas GitHub

As páginas do GitHub foram projetadas para hospedar suas páginas pessoais, da organização ou do projeto de um repositório do GitHub.

Fonte

As páginas do GitHub estão desativadas no momento. Você deve primeiro adicionar conteúdo ao seu repositório antes de publicar um site de páginas do GitHub. [Saiba mais](#).

Nenhum ▾

Salve □

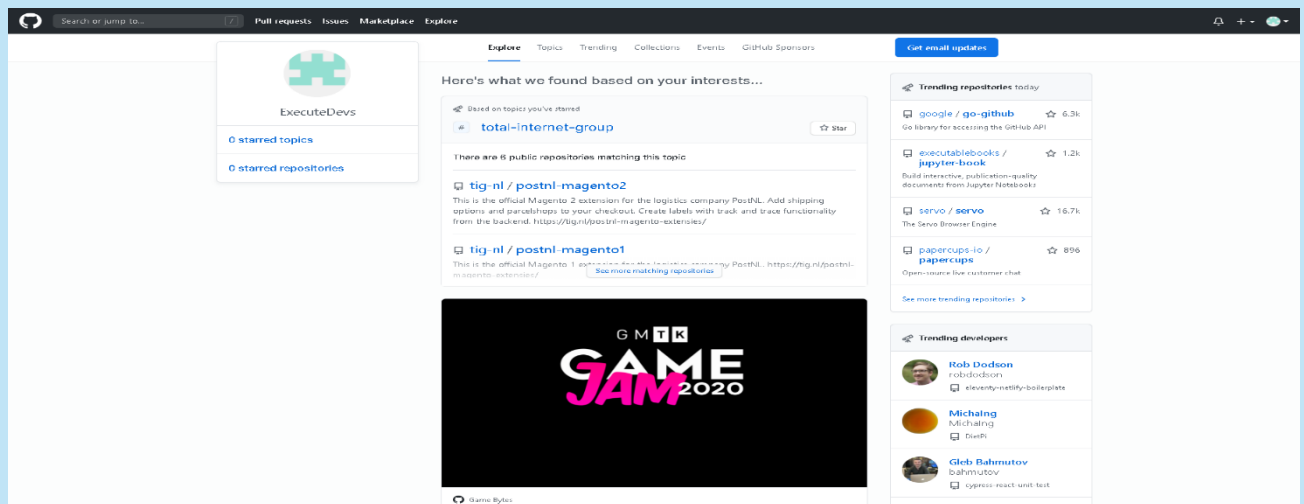
Seletor de Tema

Selecione um tema para publicar seu site com um tema Jekyll usando o gh-pagesbranch. [Saiba mais](#).

Escolha um tema

Temos também uma rede social muito poderosa podendo ser utilizada por todos, para compartilhar pensamentos, discussões, pedir ajuda ou até mesmo publicar um projeto público para que outras pessoas possam ajudar a construção do seu site/aplicativo.





Conceito

GitHub tem um conceito muito simples de arquitetura, imagine uma árvore contendo diversos galhos e nestes galhos cada um tem sua quantidade de folhas, todos galhos são ligados ao tronco da árvore, logo se este tronco de árvore for alterado (danificado) todos os galhos conectados a este tronco vão sofrer juntamente, ou até mesmo o processo inverso, se um galho muito grande for retirado, o tronco central da árvore irá sentir essa mudança. GitHub tem o mesmo conceito, imagine um projeto sendo desenvolvido ou até mesmo um servidor, você precisa de uma equipe para desenvolver, então você como líder cria um repositório, neste repositório você irá conter um ambiente chamado de **Master** (tronco central da árvore), desta forma o projeto que consiste na **Master** será seu projeto mais atualizado. Em seguida vem a etapa de desenvolver novas funções a este projeto, cada desenvolvedor irá precisar criar uma nova função, esta função não poderá ser desenvolvida na **Master**, porque desta forma todos desenvolvedores acabaram por ventura modificando o código aonde outro desenvolvedor está mexendo, ou seja, podemos acabar quebrando a aplicação, a solução para este caso é criar galhos (**Branch**) para que cada desenvolvedor tenha o seu ambiente de desenvolvimento, sem modificar algo aonde seu parceiro também esteja mexendo, assim que cada desenvolvedor terminar de efetuar suas alterações eles enviam seus códigos para a Master. Por final a Master sempre irá conter o projeto atualizado e versionado.



Capítulo 1

Criando seu primeiro Repositório

Primeiro passo

O primeiro passo a ser tomado para iniciarmos o versionamento de arquivos é criando seu primeiro repositório no site do GitHub: <https://github.com/>

Ao entrar no site pela primeira vez, para criar um repositório, você irá reparar no canto esquerdo superior o GitHub questionando se você está pronto para criar seu primeiro repositório. Basta apenas clicar no botão verde **Create Repository**.

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository

[Import repository](#)

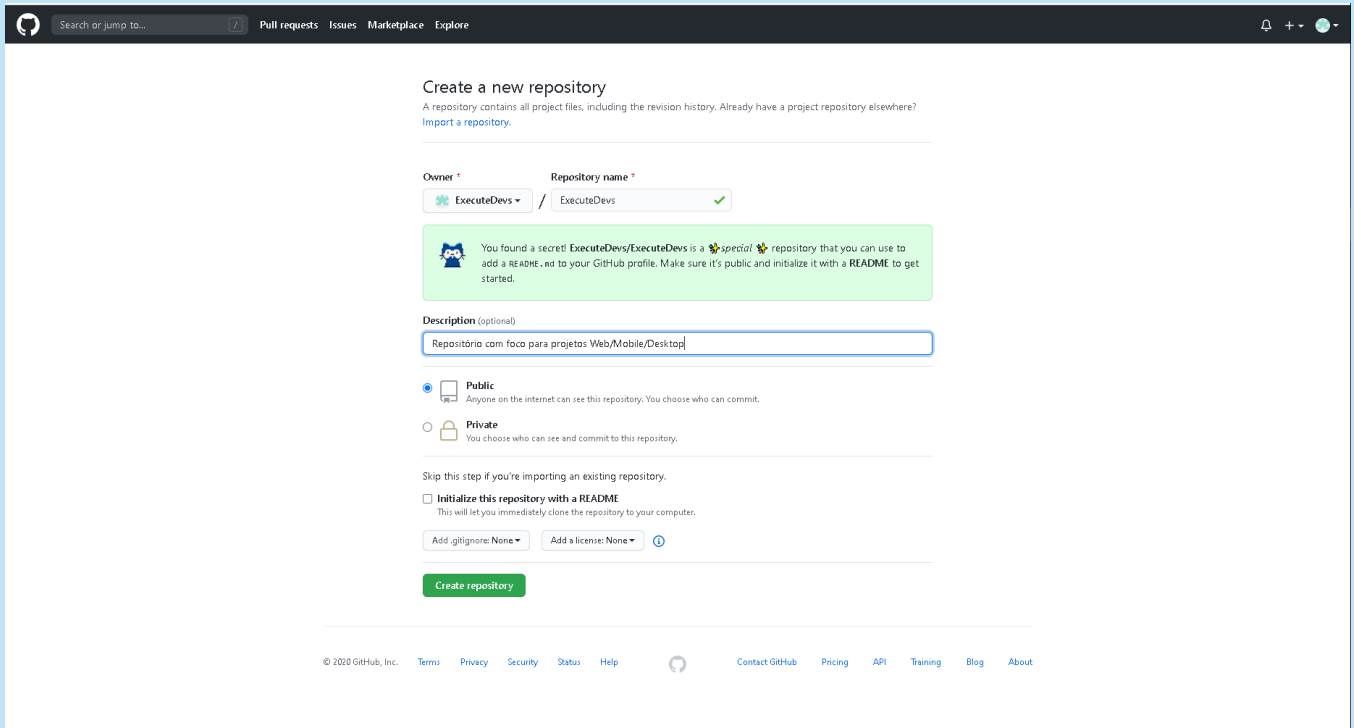
Working with a team?

GitHub is built for collaboration. Set up an organization to improve the way your team works together, and get access to more features.

[Create an organization](#)



Na etapa a seguir você irá criar e configurar seu repositório pela primeira vez, após a imagem irei descrever cada função e etapa.



Opções:

- **Owner**
Owner é o proprietário do repositório, na maioria dos casos quem está logado é o proprietário.
- **Repository name**
Em repository name você irá colocar o nome do seu repositório que achar mais adequado (ExecuteDevs ou ServidorJogo).
- **Description**
Description é uma pequena descrição para você e que outras pessoas saibam qual a finalidade deste repositório.
- **Public / Private**
Neste local você irá configurar seu repositório para ele ser público ou privado.
Caso seu objetivo seja ter este repositório para controle pessoal, projetos pessoais ... é recomendado que seja criado como privado assim apenas você ou pessoas que convidar podem manusear estes arquivos.
Mas se o seu intuito seja construir uma aplicação por exemplo uma



Calculadora para ajudar diversas pessoas, seja descobrindo como criar uma calculadora ou até mesmo para usarem o aplicativo, recomenda-se deixar este repositório público, mas lembrando que se optar em deixar este repositório público qualquer pessoa poderá manipular os arquivos que você colocou. Mas futuramente você poderá alterar essa configuração também.

- **Initialize this repository with a README**

Quando marcada essa opção seus projetos vão conter um **"TXT"** aonde você poderá editar para colocar regras e informações que achar necessário.

- **GitIgnore**

O GitIgnore é uma configuração para projetos, aonde ele irá ignorar arquivos que você não deseja subir ou não seja necessário, por exemplo vamos dizer que você criou um arquivo de texto, com uma cola sua dos comandos de GitHub, você pode colocar o nome do arquivo no Gitignore aonde ele irá sempre atualizar o repositório mas retirando este arquivo que você criou.

- **Add License**

Caso necessário um projeto e necessitar de licenças basta apenas procurar a licença que condiz com a linguagem que você está desenvolvendo seu projeto.

Por último basta apenas clicar no botão verde de **Create Repository**

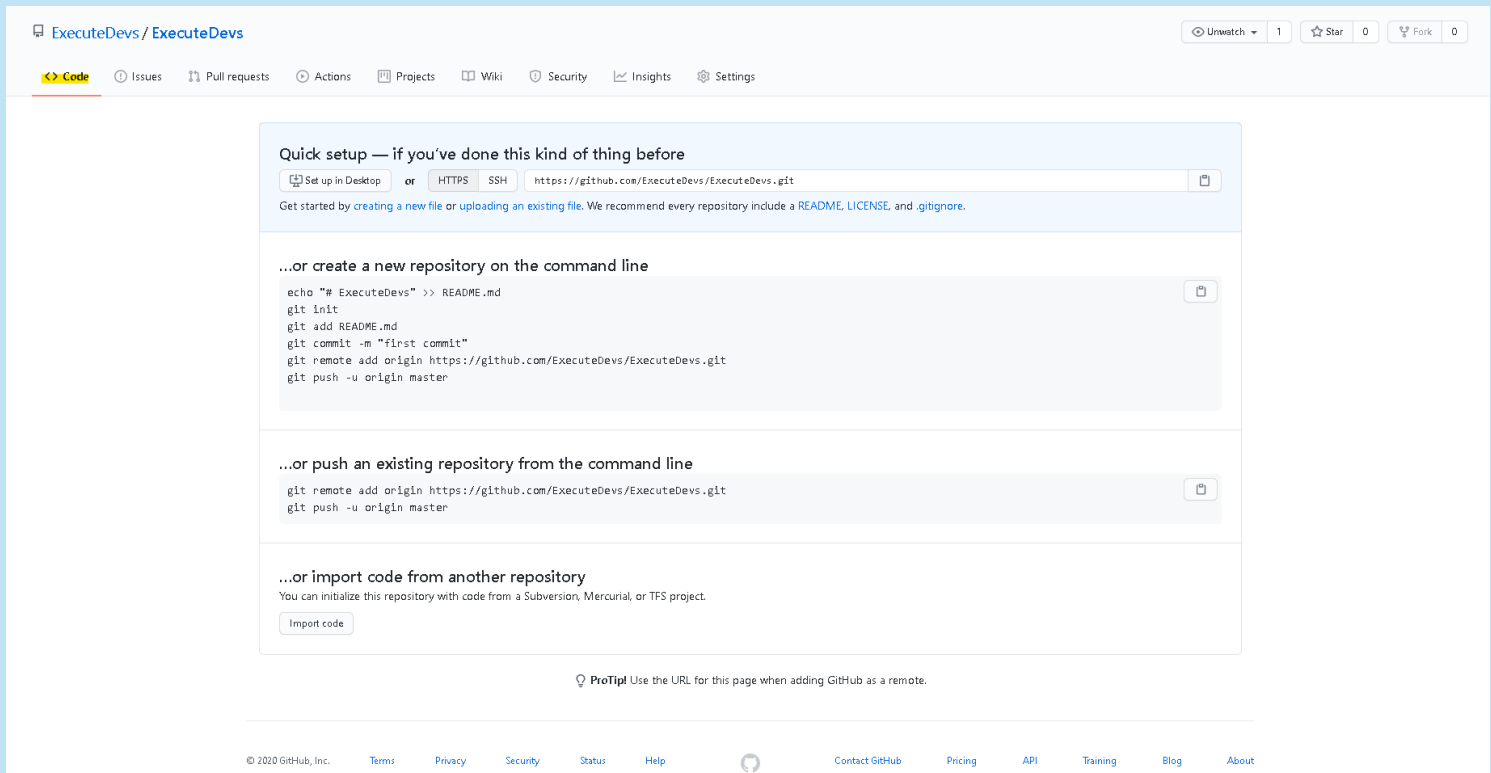


Capítulo 2

Configurando seu Repositório

Neste capítulo vamos explorar melhor a ferramenta em sua parte web para que você possa entender o que pode ser contruído, configurado ou até mesmo excluir.

Code



The screenshot shows the GitHub repository page for `ExecuteDevs/ExecuteDevs`. The **Code** tab is selected, displaying instructions for setting up the repository. The page includes a navigation bar with links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. In the top right corner, there are buttons for Unwatch (1), Star (0), and Fork (0). The main content area provides three methods for cloning the repository:

- Quick setup — if you've done this kind of thing before:** Offers options to 'Set up in Desktop', 'HTTPS', or 'SSH'. The HTTPS URL is `https://github.com/ExecuteDevs/ExecuteDevs.git`. A note suggests including a `README`, `LICENSE`, and `gitignore`.
- ...or create a new repository on the command line:** Provides a series of terminal commands:

```
echo "# ExecuteDevs" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/ExecuteDevs/ExecuteDevs.git
git push -u origin master
```
- ...or push an existing repository from the command line:** Provides terminal commands:

```
git remote add origin https://github.com/ExecuteDevs/ExecuteDevs.git
git push -u origin master
```
- ...or import code from another repository:** Includes a note about initializing from Subversion, Mercurial, or TFS, and an 'Import code' button.

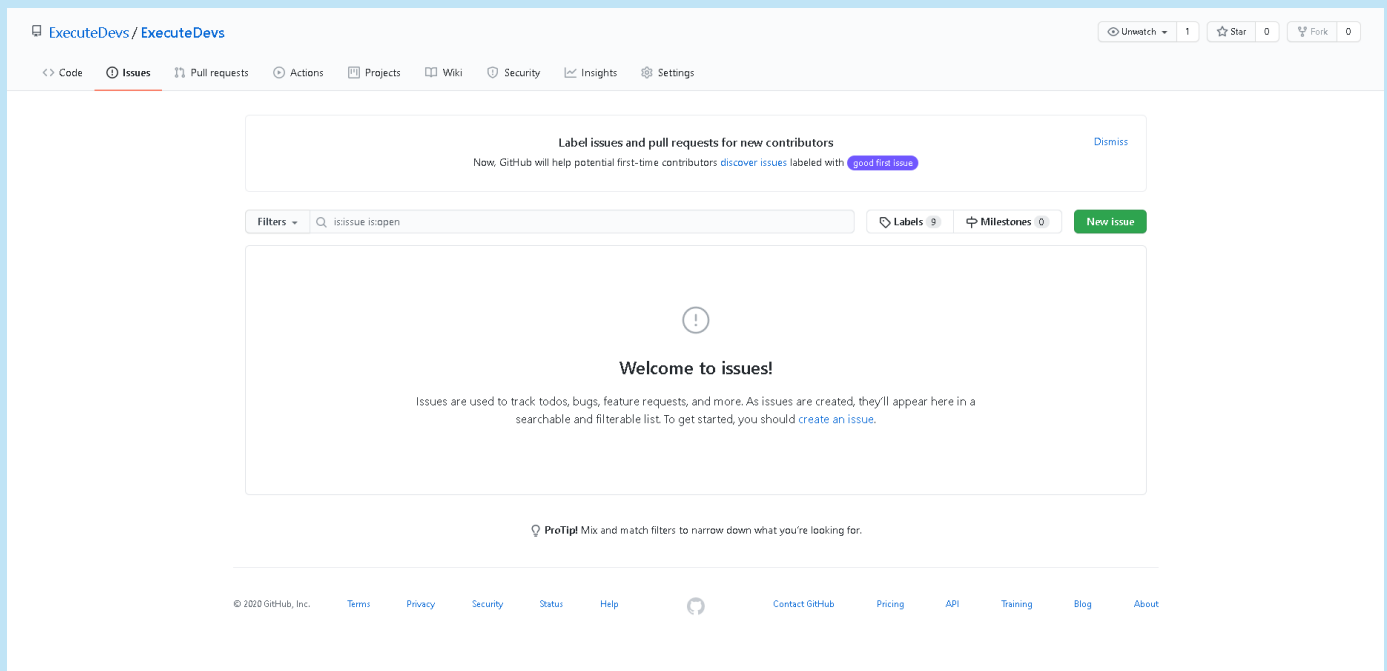
A **ProTip!** at the bottom suggests using the page URL when adding GitHub as a remote. The footer contains copyright information for 2020 GitHub, Inc., and links to Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About.

Nesta primeira aba em **Code** estará concentrado seu projeto/código/arquivos desta forma podendo monitorar e gerenciar seu repositório.

Neste primeiro passo seu repositório estará vazio, mas o GitHub explica para você como pode iniciar seu primeiro versionamento. Mas iremos abordar o versionamento no capítulo 3.



Issues



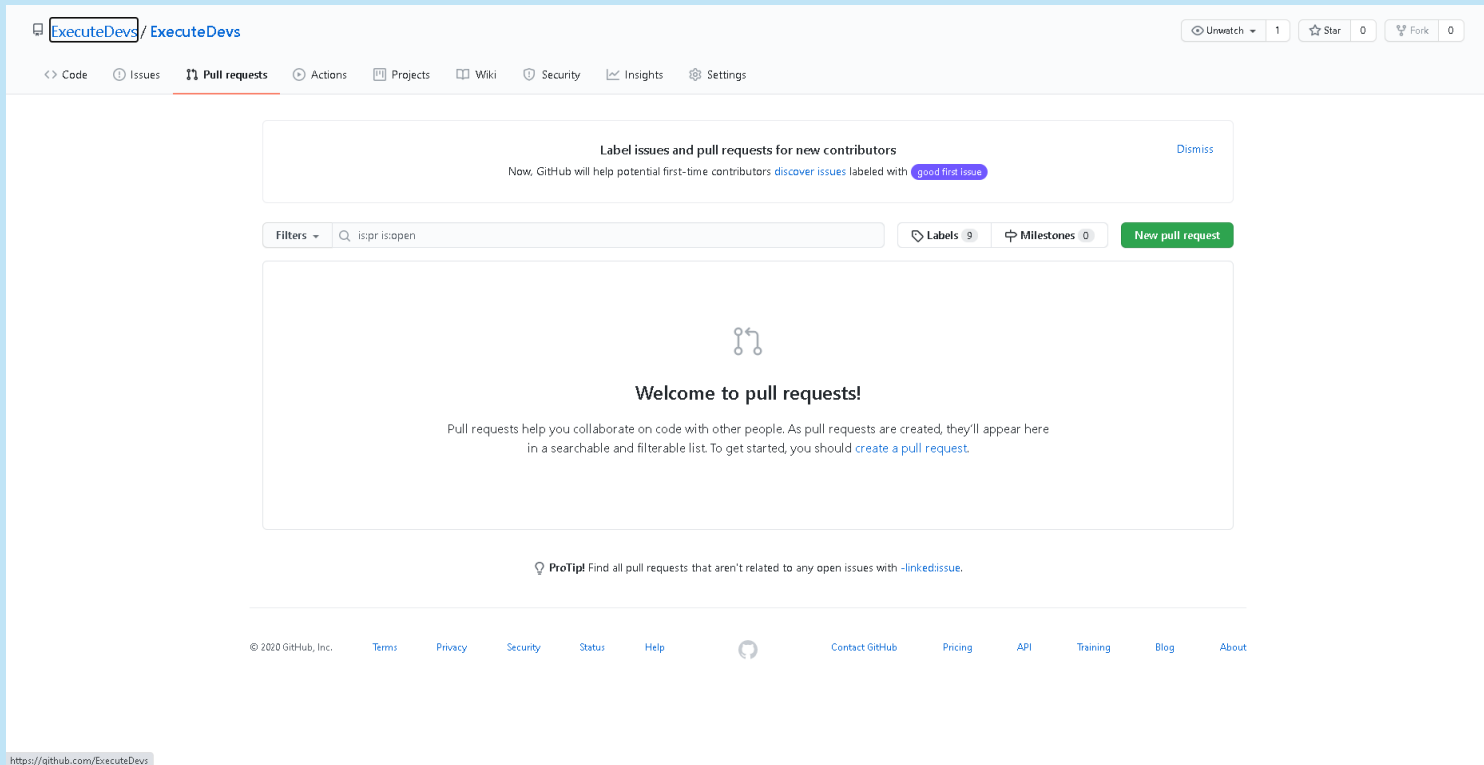
Nesta aba de **Issues** você consegue monitorar o que está ocorrendo em seu projeto, em relação a requisições, conflitos, versões etc.

Sendo assim imagine que você tem um projeto Web aonde você e diversos desenvolvedores estão trabalhando, alguém acabou subindo uma versão errada, ou até mesmo antes do tempo você consegue monitorar todo o ciclo.

Juntamente nesta aba você consegue fazer demarcações **Milestones**, onde desta forma você pode colocar ponteiros indicando um exemplo quando aquele projeto foi publicado .



Pull requests

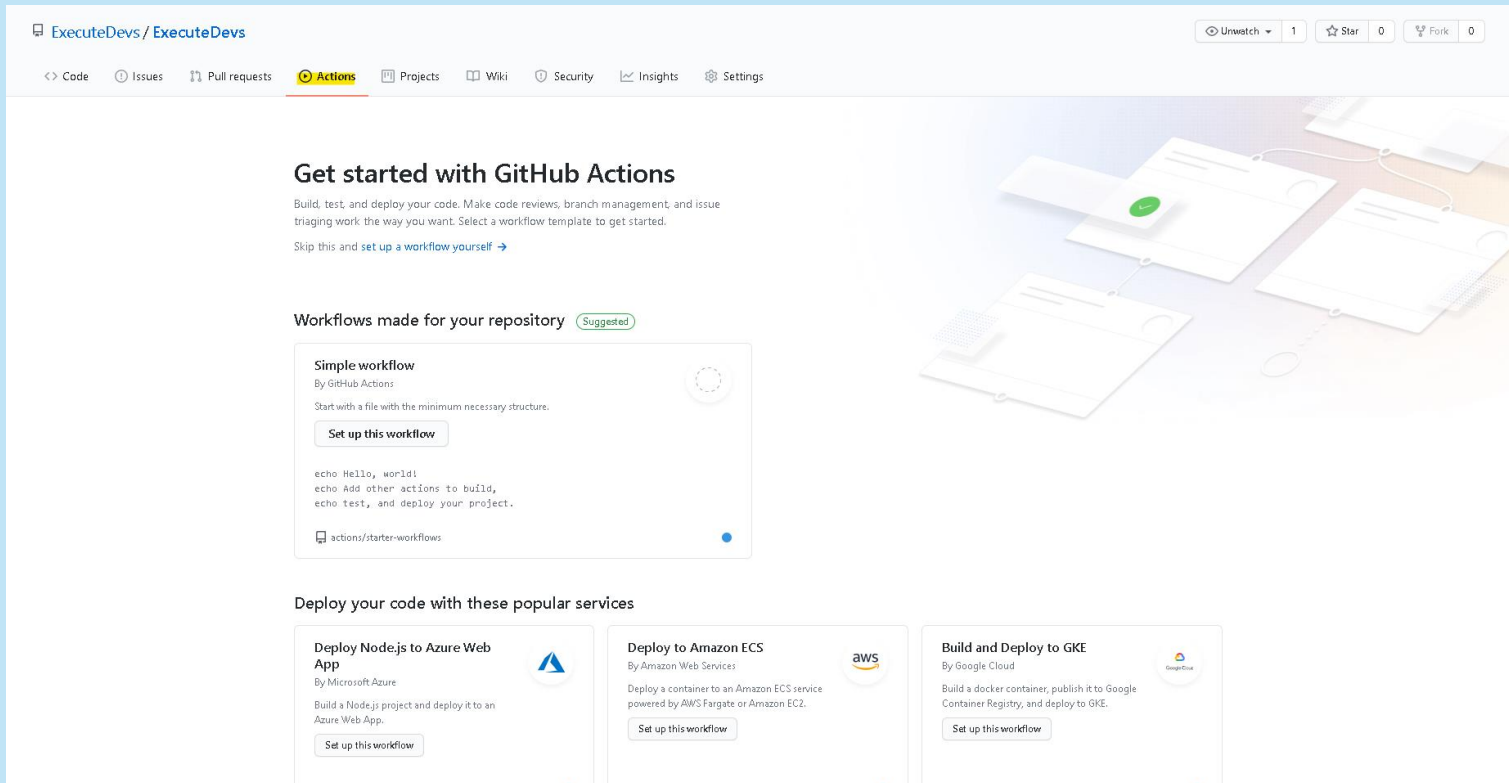


The screenshot shows the GitHub interface for the repository 'ExecuteDevs'. At the top, there's a navigation bar with links for Code, Issues, Pull requests (active), Actions, Projects, Wiki, Security, Insights, and Settings. On the right, there are buttons for Unwatch (1), Star (0), and Fork (0). Below the navigation bar, a message states: 'Label issues and pull requests for new contributors. Now, GitHub will help potential first-time contributors discover issues labeled with good first issue.' A 'Dismiss' link is on the right. Below this, there's a search bar with 'is:pr is:open' and buttons for Filters, Labels (9), Milestones (0), and a green 'New pull request' button. The main content area features a 'Welcome to pull requests!' message, explaining that pull requests help collaborate on code and will appear in a searchable and filterable list. A 'ProTip!' at the bottom suggests finding pull requests not related to open issues with the '-linked:issue' filter. The footer contains copyright information for GitHub, Inc. and various links like Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About.

Trabalhando com **Pull Request**, como mencionado acima o GitHub trabalha com um método parecido com árvore, desta forma imagine que você como o dono do repositório está trabalhando com diversos desenvolvedores, cada desenvolver irá conter uma **Branch** em sua máquina, desta forma sempre que o desenvolvedor terminar de codificar e precisar manter para a raiz da árvore o código novo e atualizado, ele vai passa pela sua autorização aonde você irá verificar o que foi modificado, quando e por quem, ciente destas alterações você pode aceitar(**Pull**) ou rejeitar esse envio(**Push**) realizado pelo desenvolvedor.



Actions



The screenshot shows the GitHub Actions interface for the repository 'ExecuteDevs/ExecuteDevs'. At the top, there's a navigation bar with links to Code, Issues, Pull requests, Actions (highlighted), Projects, Wiki, Security, Insights, and Settings. On the right, there are buttons for Unwatch, 1, Star, 0, Fork, and 0. The main content area is titled 'Get started with GitHub Actions' and includes a brief description of Actions: 'Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow template to get started.' Below this, there's a link to 'Skip this and set up a workflow yourself'. The section 'Workflows made for your repository' is marked as 'Suggested' and features a 'Simple workflow' by GitHub Actions. This workflow is described as starting with a file with the minimum necessary structure and includes a 'Set up this workflow' button. The workflow steps are: 'echo Hello, world!', 'echo Add other actions to build,', 'echo test, and deploy your project.', and a link to 'actions/starter-workflows'. Below this, the section 'Deploy your code with these popular services' offers three templates: 'Deploy Node.js to Azure Web App' by Microsoft Azure, 'Deploy to Amazon ECS' by Amazon Web Services, and 'Build and Deploy to GKE' by Google Cloud. Each template includes a brief description, a 'Set up this workflow' button, and a link to the workflow.

Trabalhando um pouco mais voltado com projetos na aba **Actions** conseguimos utilizar diversos serviços, dentre eles estão:

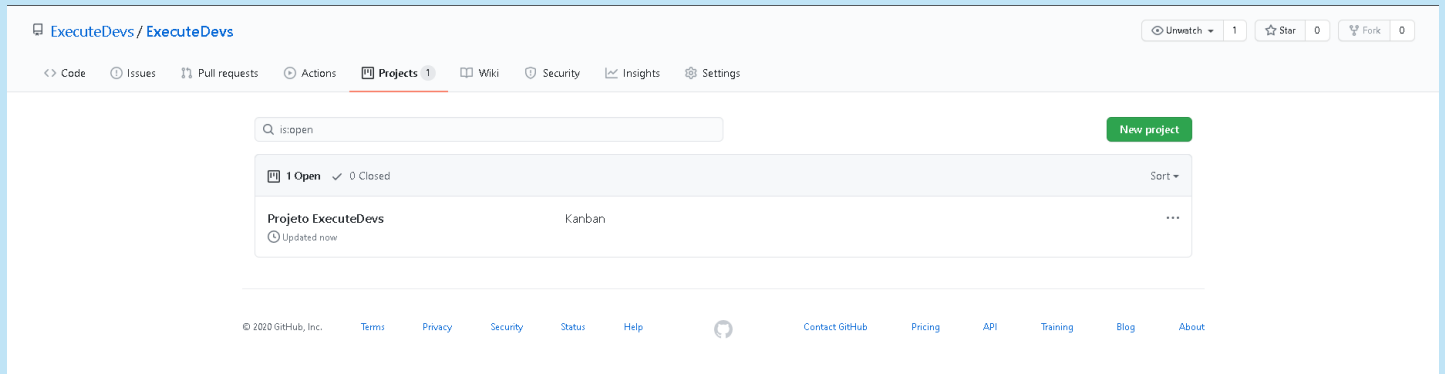
- Publicação de projeto
- Testes em aplicações e projetos
- Automações

Pense da seguinte forma, você tem uma aplicação em JSP ou .Net aonde está efetuando o versionamento dela no seu repositório, diremos que você precise efetuar a publicação deste projeto agora na **AWS**, dentro de **Actions** você consegue efetuar o deploy desta aplicação na **AWS** diretamente do seu repositório.

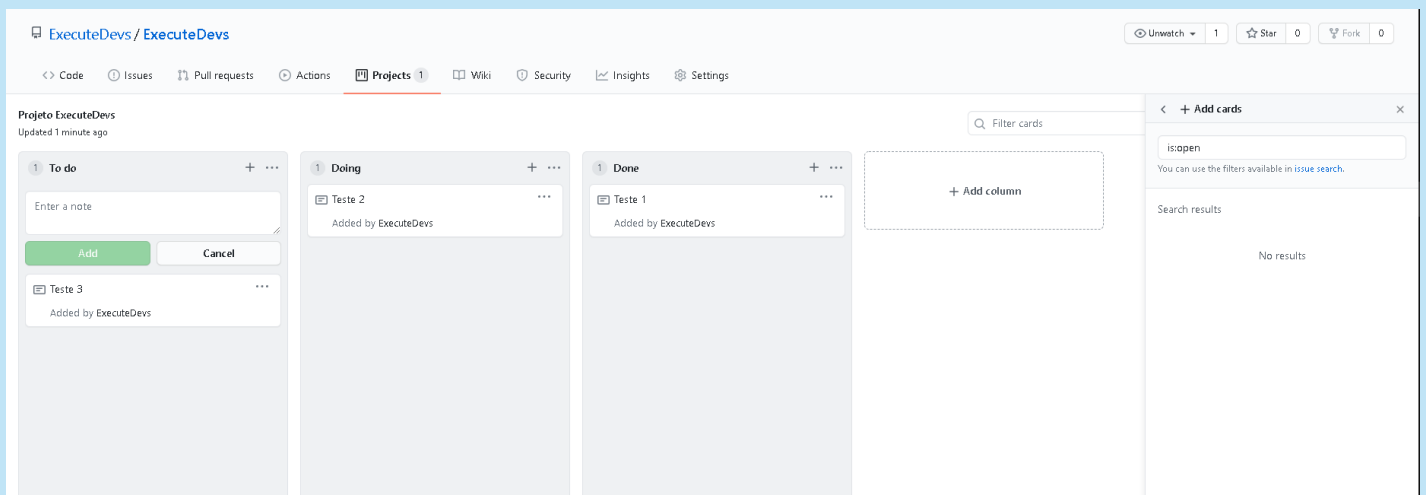
Pesando além você pode criar um ciclo de automação aonde ao colocar o projeto mais atual no seu repositório ele irá testar seu projeto e publicar automaticamente na **AWS**.



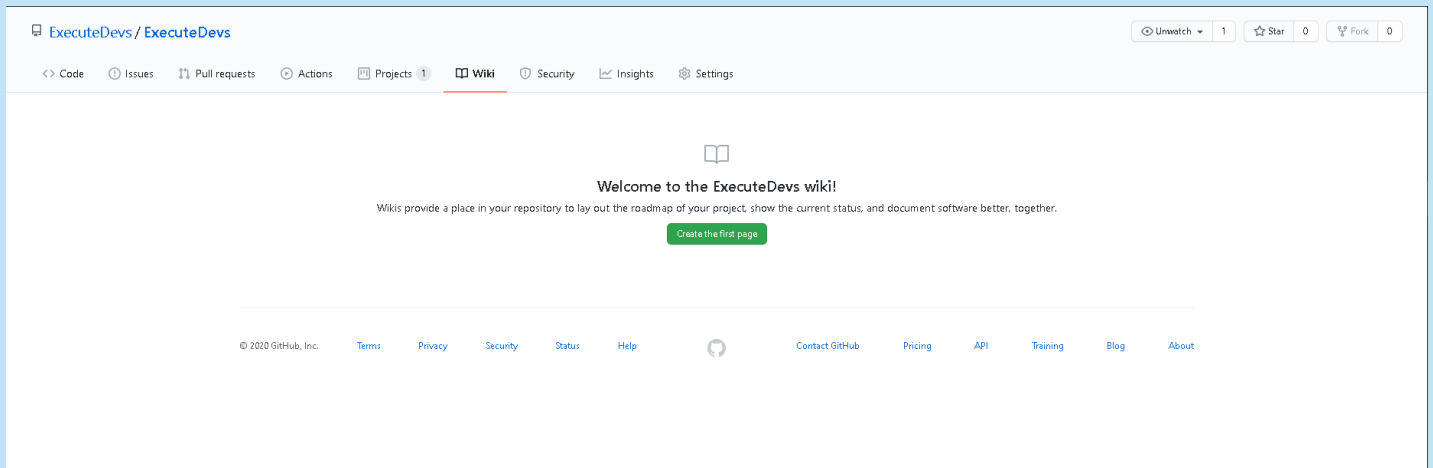
Projects



Entrando mais afundo em projetos, podemos fazer um gerenciamento de tarefas a desenvolver dentro do próprio GitHub. Podemos montar um Kanban dentro do GitHub onde podemos centralizar tudo em uma única ferramenta, sem necessidade de manter duas ou mais ferramentas para auxílio do desenvolvimento do projeto e até mesmo gerenciamento e divisão de tarefas.



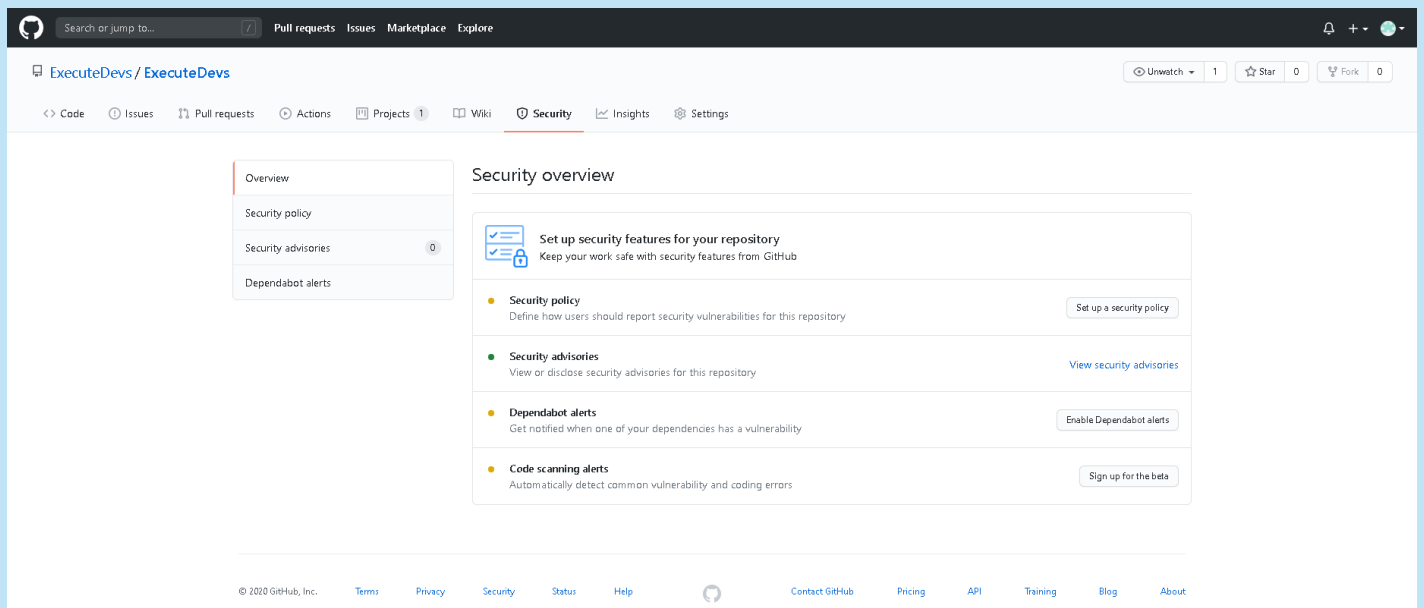
Wiki



Pensando mais adiante e explorando temos a aba Wiki, sua função pode ser definida de uma forma bem prática e direta, pense que todo projeto necessita de uma documentação sendo assim, aonde mais podemos atribuir uma documentação de forma específica e centralizada? No próprio repositório do projeto.



Security

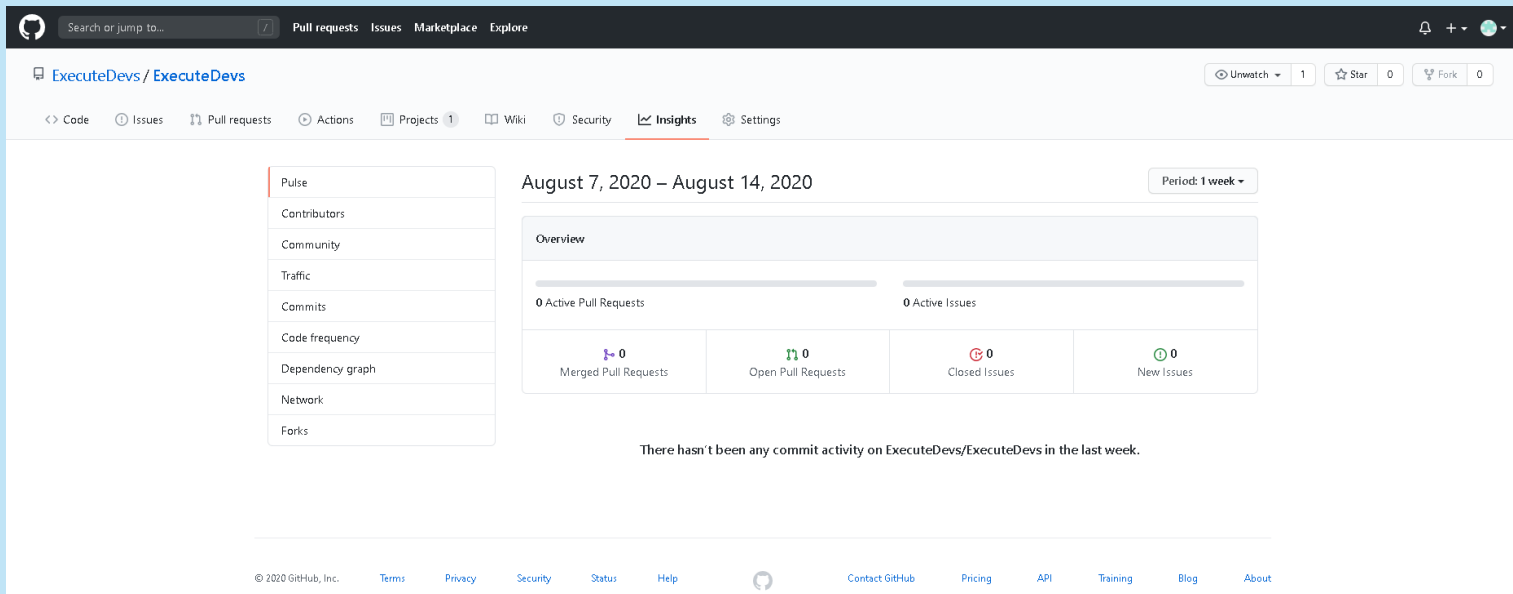


The screenshot shows the GitHub interface for the repository 'ExecuteDevs / ExecuteDevs'. The 'Security' tab is selected in the top navigation bar. On the left, a sidebar lists 'Overview' (active), 'Security policy', 'Security advisories' (0), and 'Dependabot alerts'. The main content area, titled 'Security overview', contains a section 'Set up security features for your repository' with the instruction 'Keep your work safe with security features from GitHub'. Below this, four features are listed: 'Security policy' (with a 'Set up a security policy' button), 'Security advisories' (with a 'View security advisories' link), 'Dependabot alerts' (with an 'Enable Dependabot alerts' button), and 'Code scanning alerts' (with a 'Sign up for the beta' button). The footer includes copyright information and various links like 'Terms', 'Privacy', 'Security', 'Status', 'Help', 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About'.

Um pouco mais para o lado e nos deparamos com a aba de Security, neste momento imagine que você tem um projeto público, com uma finalidade de fornecer e também receber ajuda de outros desenvolvedores para construção de um projeto, desta forma diversas pessoas podem acessar e manipular seu repositório, mas agora pense bem isso poderia ser perigoso porque alguém pode apagar seu projeto e tentar subir uma pasta vazia para seu repositório, sendo assim desta forma Security é uma aba aonde você pode organizar políticas de segurança e visualizar até mesmo falhas e visualizar se a segurança do seu projeto foi violada.



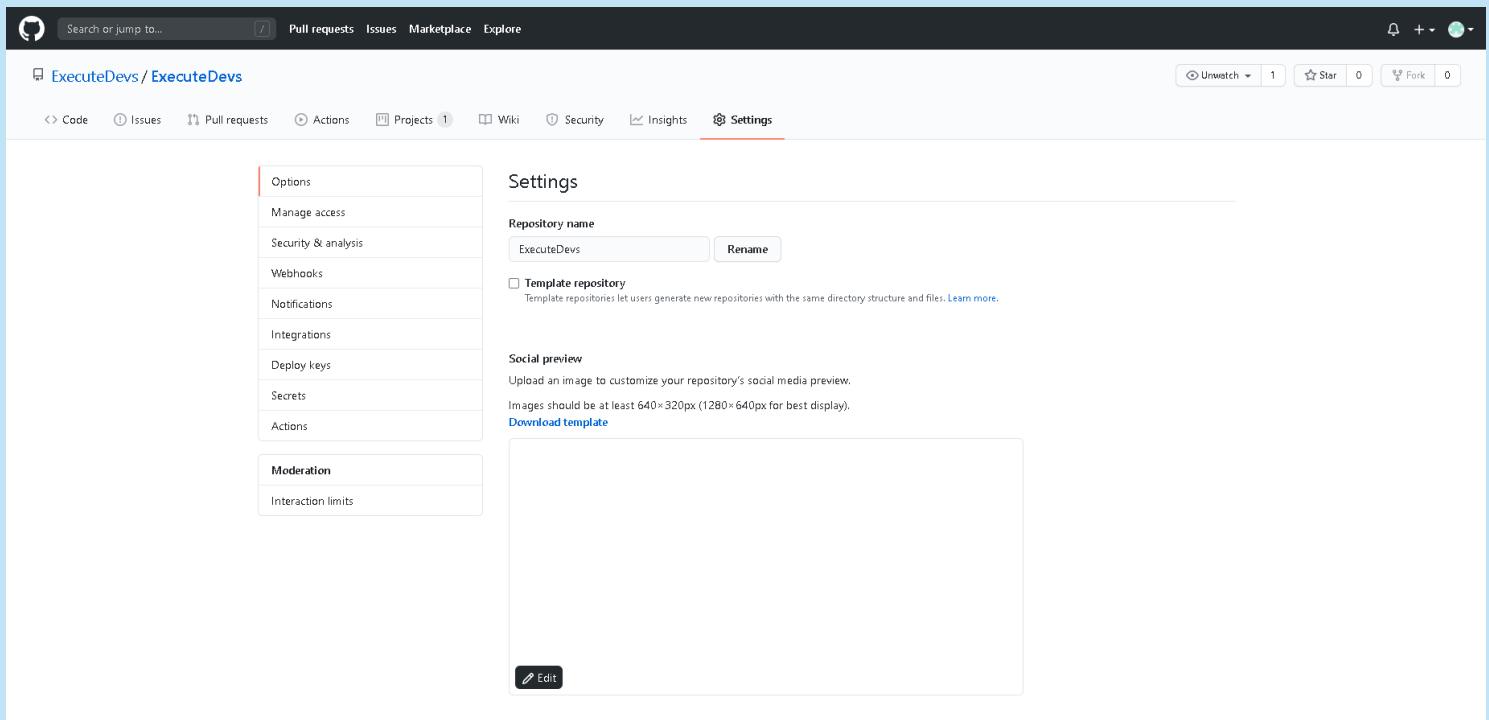
Insights



Podemos ter diversos acompanhamentos pela plataforma e uma delas é monitoramento via gráficos de como está sendo utilizado o repositório, ou seja, podemos ver os colaboradores do repositório e sua frequência, podemos monitorar qual o volume que está subindo de dados e o fluxo dentre outras opções.



Settings



The screenshot shows the GitHub interface for the repository 'ExecuteDevs'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository name 'ExecuteDevs' is displayed, along with 'Unwatch', 'Star' (0), and 'Fork' (0) buttons. The left sidebar contains a list of settings categories: Options, Manage access, Security & analysis, Webhooks, Notifications, Integrations, Deploy keys, Secrets, Actions, Moderation, and Interaction limits. The main content area is titled 'Settings' and includes sections for 'Repository name' (with a 'Rename' button), 'Template repository' (with a checkbox and a link to 'Learn more'), and 'Social preview' (with instructions on image requirements and a 'Download template' link). An 'Edit' button is located at the bottom of the social preview section.

Por último mas não menos importante encontramos na última aba Settings tudo em relação a um repositório ou seja, podemos configurar ele de todas formas, alterando de privado para público e público para privado, podemos convidar pessoas para trabalhar e ajuda em nosso repositório, alterar nomes, criar chaves de acesso dentre outras coisas que você poderá visualizar nos próximos capítulos.

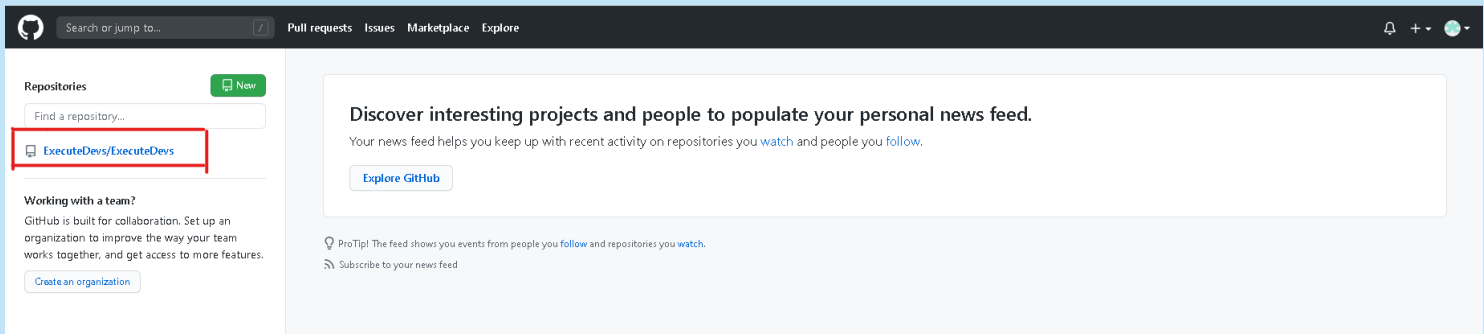


Capítulo 3

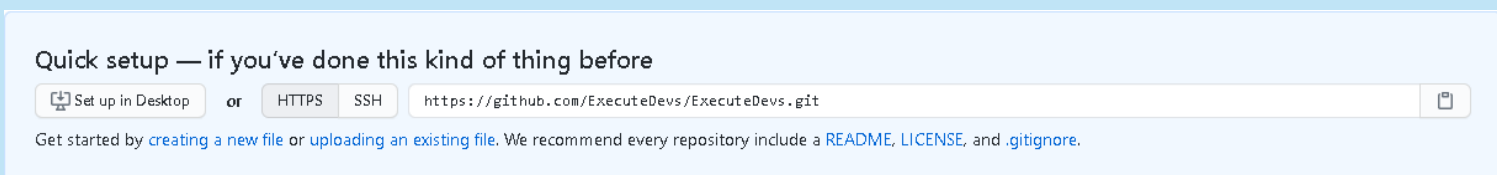
Efetuando seu primeiro versionamento

Neste capítulo vamos iniciar seu primeiro versionamento de arquivos

Code



Agora para iniciarmos a terceira parte do curso, você deverá selecionar um projeto/arquivo que deseja fazer o versionamento em sua máquina. Após decidir qual projeto deseja versionar selecione um repositório clicando no nome do projeto.



Em seguida procure na aba **Code**, a imagem acima onde ele referência o link para iniciar o repositório do seu projeto em sua máquina.

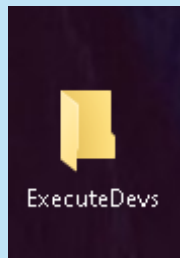
Podemos acessar o repositório de duas formas:

- SSH
Acessamos o repositório via Shel (Linha de comando)
- HTTPS
Acessamos o repositório via URL (Como se acessaríamos uma página Web)

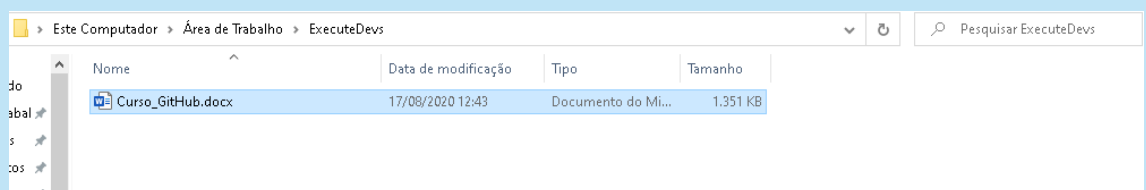
Logo após localizar essa informação copie o link HTTPS do seu repositório, você pode selecionar a URL e pressionar Ctrl + C ou clicando no ícone ao fim da linha.



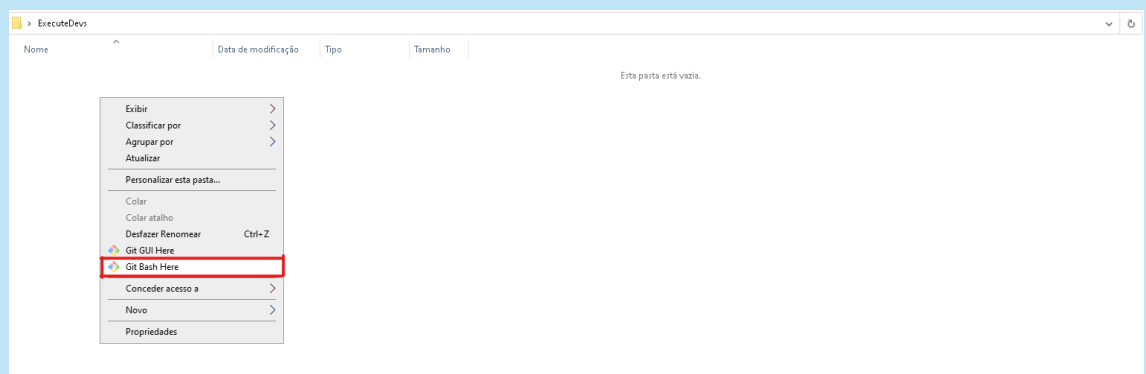
Iniciando seu repositório



Para iniciarmos seu repositório de maneira adequada, primeiro crie uma pasta na sua área de trabalho com o nome que achar adequado.



Em seguida como segunda etapa adicione o arquivo que deseja na pasta, para que ele seja disponibilizado em seu repositório, neste caso vamos demonstrar com o próprio arquivo do curso que estamos criando para disponibilizar.

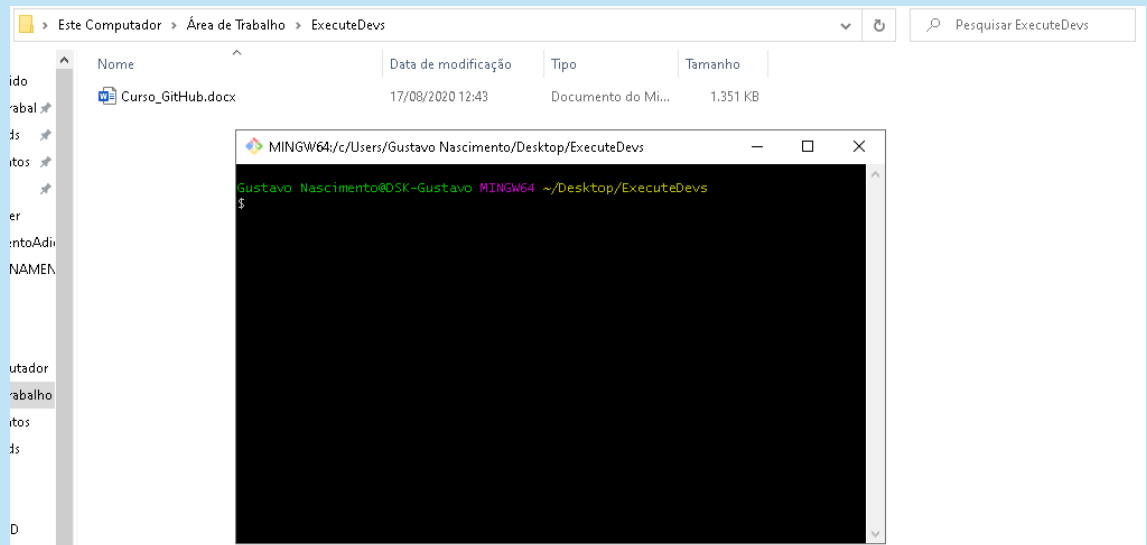


Em seguida clique com o botão direito em qualquer lugar da pasta aonde esteja vazio.

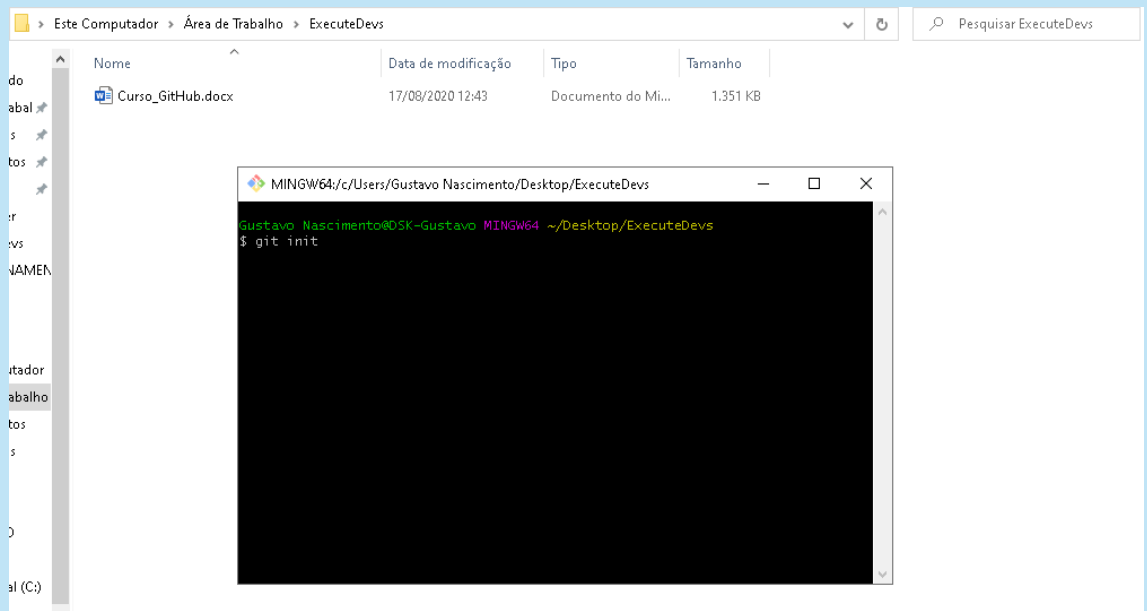
Um menu com algumas opções estará disponível para você procure pelo menu **Git Bash Here**, caso não encontre o menu certifique-se que você seguiu o passo a passo de **Requisitos**.



Git Bash





Após abrir o prompt de comando do GitHub vamos efetuar todo o versionamento via linha de comando, o GitHub tem sua parte gráfica para efetuarmos o versionamento, mas neste momento vamos lhe mostrar como utilizar via linha de comando, muitas empresas acabam optando efetuar via linha de comando o versionamento, mas não se engane, efetuar versionamento pela parte gráfica(GUI) também não contem diferença alguma.



Para iniciarmos a subida dos arquivos ao repositório devemos digitar o seguinte comando **git init** (tudo em minúsculo), este comando irá criar uma pasta oculta para você com as configurações básicas iniciais.



Nome	Data de modificação	Tipo	Tamanho
 Curso_GitHub.docx	17/08/2020 12:43	Documento do Mi...	1.351 KB
 .git	17/08/2020 15:41	Pasta de arquivos	

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs
$ git init
Initialized empty Git repository in C:/Users/Gustavo Nascimento/Desktop/ExecuteDevs/.git/

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Curso_GitHub.docx
        ~$rso_GitHub.docx

nothing added to commit but untracked files present (use "git add" to track)

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ |
```

Em seguida logo após ele ter criado as configurações padrões, vamos verificar quais arquivos estão constando como alterados no seu repositório, basta executar o comando **git status**.

Os Arquivos em vermelho são arquivos novos que ainda não existem no diretório, caso esses arquivos já existam, ele apenas trocaria de cor caso esse arquivo consta-se qualquer mudança.

Observação nem toda alteração significa que aquele arquivo já existe e sofreu alteração dentro dele, toda alteração é classificada da seguinte maneira:

Se existir qualquer coisa de diferente em relação ao index (Ponteiro/Repositório) vai ser considerado uma alteração, seja adição, remoção ou alteração de arquivo.



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs
$ git init
Initialized empty Git repository in C:/Users/Gustavo Nascimento/Desktop/ExecuteDevs/.git/
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git add .
```

Logo após ele ter adicionado à pasta do Git contendo suas configurações iniciais e verificar que existem arquivos pendentes para subir em seu repositório, será necessário adicionar outro comando **git add .** ou **git add all**, desta forma você irá indicar para o GitHub que deseja adicionar em um estado de subida todos arquivos acusados de diferentes naquela pasta, caso deseje subir apenas um arquivo específico coloque o comando com o nome do arquivo **git add Curso_GitHub.docx**. Pressione Enter, caso não mostre nenhuma mensagem de erro, significa que o arquivo foi adicionado com sucesso, mas isso significa que ele está em um estado de “pendente”, ainda será necessário indicar com um **Commit** (Comentário) o que está ocorrendo com essa mudança.



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs
$ git init
Initialized empty Git repository in C:/Users/Gustavo Nascimento/Desktop/ExecuteDevs/.git/

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Curso_GitHub.docx
        ~$rso_GitHub.docx

nothing added to commit but untracked files present (use "git add" to track)

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git commit -m "Subindo primeiro arquivo para o repositório"
```

Para adicionar um comentário a uma mudança será necessário adicionar o seguinte comando **git commit -m "Mensagem de texto que achar adequado"**. Toda mudança na pasta do seu repositório precisar ser adicionada para subida e deve conter um comentário de identificação, para que quando analisado no seu repositório contenha as informações explicando aquela alteração, seja um arquivo novo adicionado, alterado ou até mesmo excluído. Logo em seguida pressione Enter.

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs

nothing added to commit but untracked files present (use "git add" to track)

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git commit -m "Subindo primeiro arquivo para o repositório"

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Gustavo Nascimento@DSK-Gustavo.(none)')

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$
```



Mas agora vamos com calma, provavelmente você deve ter se deparado com uma tela pedindo para que você se identifique.

Isso ocorre apenas a primeira vez que você está efetuando sua primeira subida via desktop.

Vamos configurar seu usuário do GitHub Desktop com o usuário do GitHub que você efetua acesso na web, pegue seu e-mail e nome de usuário cadastro no GitHub.

Adicione a seguinte linha

git config --global user.email "emailcadastrado@email.com" pressione Enter.

git config --global user.name "UsuarioCadastrado" pressione Enter.

Após isso repita o comando de **git commit -m "Texto"** e pressione Enter.

```
MINGW64:/c:/Users/Gustavo Nascimento/Desktop/ExecuteDevs
git config --global user.email "you@example.com"
git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'Gustavo Nascimento@DSK-Gustavo.(none)')

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git config --global user.email "execute.devs@gmail.com"

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git config --global user.name "ExecuteDevs"

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git commit -m "Subindo primeiro arquivo para o repositório"
[master (root-commit) 93e6db5] Subindo primeiro arquivo para o repositório
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Curso_GitHub.docx
create mode 100644 ~$rso_GitHub.docx

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$
```

Seu resultado será semelhante a este.

O GitHub está informando que 2 arquivos foram criados no commit efetuado.

A partir deste momento você deixa seus arquivos em estado de “pronto para subida”, basta finalizarmos apontando para qual diretório desejamos fazer o upload projeto e por último efetuar o upload.




```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
$ git config --global user.name "ExecuteDevs"

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git commit -m "Subindo primeiro arquivo para o repositório"
[master (root-commit) 93e6db5] Subindo primeiro arquivo para o repositório
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Curso_GitHub.docx
create mode 100644 ~$rso_GitHub.docx

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git remote add origin https://github.com/ExecuteDevs/ExecuteDevs.git
```

Vamos definir qual repositório desejamos que seja aplicado essas alterações consistentes. Execute o seguinte comando **git remote add origin** "Link indicado na parte web no caso do ExecuteDev:"
<https://github.com/ExecuteDevs/ExecuteDevs.git>.

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
$ git push
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin master

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git push --set-upstream origin master
remote: Permission to ExecuteDevs/ExecuteDevs.git denied to Caous.
fatal: unable to access 'https://github.com/ExecuteDevs/ExecuteDevs.git/': The requested URL returned error: 403

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git push -u origin master
remote: Permission to ExecuteDevs/ExecuteDevs.git denied to Caous.
fatal: unable to access 'https://github.com/ExecuteDevs/ExecuteDevs.git/': The requested URL returned error: 403

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.10 MiB | 1.01 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ExecuteDevs/ExecuteDevs.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$
```

Agora para finalizarmos a subida com êxito digite o seguinte comando **git push -u origin master**, este comando irá empurrar para o seu repositório suas mudanças sendo assim irá persistir os arquivos que estão na máquina no repositório.



Agora volte ao seu navegar no seu repositório na aba <>**Code**, você vai se deparar com seus arquivos e o comentário(**commit**) efetuado, com a data.

ExecuteDevs / ExecuteDevs

Unwatch 1 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects 1 Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

ExecuteDevs Subindo primeiro arquivo para o repositório 93e6db5 4 hours ago 1 commits

Curso_GitHub.docx	Subindo primeiro arquivo para o repositório	4 hours ago
~\$rsa_GitHub.docx	Subindo primeiro arquivo para o repositório	4 hours ago

ExecuteDevs/ExecuteDevs is a special repository. Its README.ad will appear on your public profile! [Send feedback](#)

Add a README

About

Repositório com foco para projetos Web/Mobile/Desktop

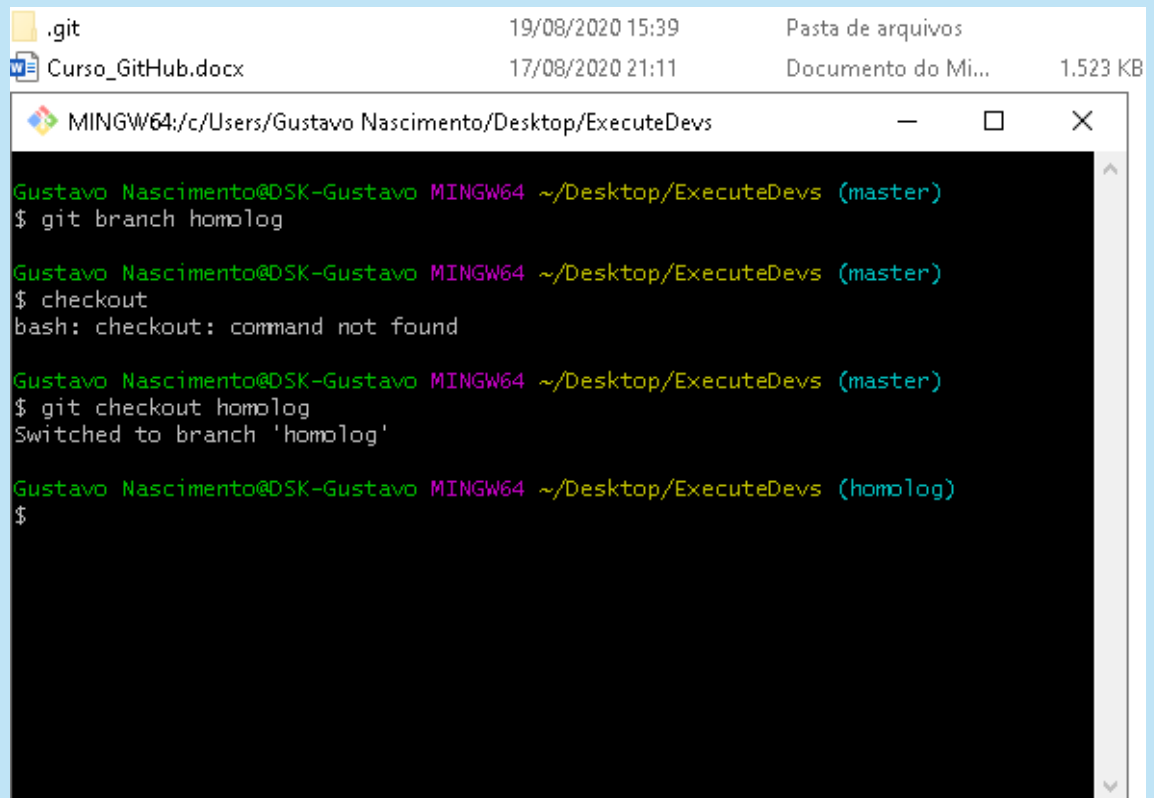


Capítulo 4

Versionamento de arquivos

No capítulo 4 vamos simular que este repositório será tratado por diversas pessoas, onde o documento que se encontra no repositório ExecuteDevs será alterado e tratado.

Criando Branch



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git branch homolog
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ checkout
bash: checkout: command not found
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (master)
$ git checkout homolog
Switched to branch 'homolog'
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$
```

A partir deste momento vamos passar a utilizar as **Branch**, como mencionado ao início do curso, imagine o GitHub como uma árvore, desta forma temos o tronco da árvore que vamos denominar de Master. Uma árvore contém diversas ramificações, onde essas ramificações podem ser alteradas e enviadas novamente para a Master, podemos criar qualquer nome que acharmos compatíveis com nossas tarefas, neste caso vamos criar duas **Branch**, uma vamos denominar de **Homolog** e outra de **Cap_4_Adicao**.

O ambiente denominado **Homolog** será um ambiente aonde vamos validar todas alterações antes de **commitar** e efetuarmos **push** para a **Master**,



normalmente encontramos esse processo em diversos projetos. Criamos uma função em um projeto, mas antes de publicarmos para todos passarem a usufruir desta função precisamos que pessoas responsáveis validem se aquele desenvolvimento está de acordo com o solicitado.

O ambiente denominado **Cap_4_Adicao**, tem como objetivo criar um ambiente para o desenvolvimento da tarefa. **Homolog**, imagine em um projeto Web, você é responsável por criar uma nova página HTML, desta forma você precisará desenvolver esta página, mas não poderá desenvolver na **Master**, porque se você publicar essa alteração e sua página, e esta página estiver com qualquer erro, você poderá derrubar o seu site, mas também não podemos fazer diretamente em **Homolog**, porque é um ambiente para testes e apenas enviamos para a branch Homolog quando o desenvolvimento foi concluído.

Então neste caso criamos uma Branch nova, onde ela irá conter todas as informações da Branch **Homolog**, para que seu projeto esteja atualizado, assim também evitando que você não desenvolva e solte esta versão diretamente na Homologação da sua aplicação Web.

Para criarmos uma nova branch e navegarmos entre elas vamos usar os seguintes comandos::

git branch homolog

git checkout homolog

Desta forma você irá identificar no fim da linha que entre parênteses está a palavra (**homolog**).

Para disponibilizarmos a branch homolog, para que outros colaboradores possam, enxergar essa branch vamos utilizar o comando de push.

git push --set-upstream origin homolog;

Desta forma vamos conseguir subir nossa branch homolog

Repita o mesmo processo para criação da branch **Cap_4_Adicao**

git branch Cap_4_Adicao;



```
git checkout Cap_4_Adicao;
```

```
git push --set-upstream origin Cap_4_Adicao;
```

```
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git branch Cap_4_Adicao
```

```
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git checkout Cap_4_Adicao
Switched to branch 'Cap_4_Adicao'
A      ~$rso_GitHub.docx
```

```
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git push --set-upstream origin Cap_4_Adicao
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Cap_4_Adicao' on GitHub by visiting:
remote:   https://github.com/ExecuteDevs/ExecuteDevs/pull/new/Cap_4_Adicao
remote:
To https://github.com/ExecuteDevs/ExecuteDevs.git
 * [new branch]      Cap_4_Adicao -> Cap_4_Adicao
Branch 'Cap_4_Adicao' set up to track remote branch 'Cap_4_Adicao' from 'origin'
.
```

Atualizando Branch

Vamos manusear as branch com o arquivo atual, a proposta será da seguinte forma.

Iremos adicionar novos conteúdos ao capítulo 4, no arquivo **Curso_GitHub**, mas faremos essas alterações apenas na branch **Cap_4_Adicao**. Após a adição dos conteúdos no capítulo 4, vamos efetuar commit e push para enviarmos essas alterações ao repositório. Em seguida vamos trocar para branch **Homolog**, com o intuito de verificar se na branch **Homolog** contém esse novo conteúdo adicionado

Mas quando conferirmos vamos ter a certeza, que as alterações ficaram apenas na branch **Cap_4_Adicao**, então como consequência vamos atualizar a branch **Homolog** e **Master**, desta forma vamos manter as branch atualizadas e com o capítulo 4 recém criado.



Para isso vamos efetuar os dois seguinte comandos:

git add .

git commit -m "Subindo alterações no capítulo 4"

git push

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git commit -m "Subindo alterações no capítulo 4"
[Cap_4_Adicao 5207bc0] Subindo alterações no capítulo 4
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ~$rso_GitHub.docx

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 52.14 KiB | 2.48 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
   9d3988d..5207bc0  Cap_4_Adicao -> Cap_4_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ |
```

Após esses três comandos executados vamos alterar para a branch de **homolog** e verificar se essas alterações estão no arquivo da branch **homolog**. Digite:

git checkout homolog.



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git commit -m "Adição de mais informações"
[Cap_4_Adicao 5db35f1] Adição de mais informações
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ~WRL0005.tmp

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 47.26 KiB | 2.36 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
   5207bc0..5db35f1  Cap_4_Adicao -> Cap_4_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git checkout homolog
Switched to branch 'homolog'
Your branch is up to date with 'origin/homolog'.
```

Ao abrir o arquivo novamente vamos nos deparar que a branch de homolog não contém nem o capítulo 4.

Arquivo Ferramentas Modo de Exibição

Curso_GitHub.docx - Salvo neste PC

Navegação

Pesquisar documento

Títulos

Páginas

Resultados

Objetivo

GitHub & Git

Git

GitHub

Git & GitHub

Criação de conta e configuração

GitHub e suas funções

Capítulo 1

Primeiro passo

Segundo passo

Capítulo 2

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

Capítulo 3

Code

Iniciando seu repositório

Git Bash

Observações do capítulo:

Observações do capítulo:

Fim do documento

MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs

```
$ git commit -m "Adição de mais informações"
[Cap_4_Adicao 5db35f1] Adição de mais informações
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ~WRL0005.tmp

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 47.26 KiB | 2.36 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
   5207bc0..5db35f1  Cap_4_Adicao -> Cap_4_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git checkout homolog
Switched to branch 'homolog'
Your branch is up to date with 'origin/homolog'.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$
```

Tela 43 de 43

Após verificarmos, está totalmente correto não conter o capítulo 4, nós criamos um ambiente para desenvolver o capítulo 4, então no ambiente de Homolog (testes), não deve conter alterações que ainda não foram concluídas. Mas agora vamos dizer que o capítulo 4 esteja completo, então devemos mandar para o ambiente de teste para que aprove as alterações. Sendo assim para atualizar o ambiente de Homolog vamos utilizar o comando **Git Merge**, este comando irá mesclara as alterações que foram efetuadas no arquivo da branch Cap_4_Adicao para sua branch **Homolog**.

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
5207bc0..5db35f1  Cap_4_Adicao -> Cap_4_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_4_Adicao)
$ git checkout homolog
Switched to branch 'homolog'
Your branch is up to date with 'origin/homolog'.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git pull
Already up to date.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git merge Cap_4_Adicao
Updating 9d3988d..5db35f1
Fast-forward
 Curso_GitHub.docx | Bin 1558780 -> 1606431 bytes
 ~$rso_GitHub.docx | Bin 0 -> 162 bytes
 ~WRL0005.tmp      | Bin 0 -> 1587415 bytes
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ~$rso_GitHub.docx
 create mode 100644 ~WRL0005.tmp

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$
```

Assim sucessivamente faremos o mesmo método para a branch Master, mas não neste momento porque lembre o ambiente Master é seu ambiente de produção, no momento temos muito ainda para produzir antes de efetuarmos merge para master.

Comando para atualizar:

git merge <Nome_Branch>



Capítulo 5

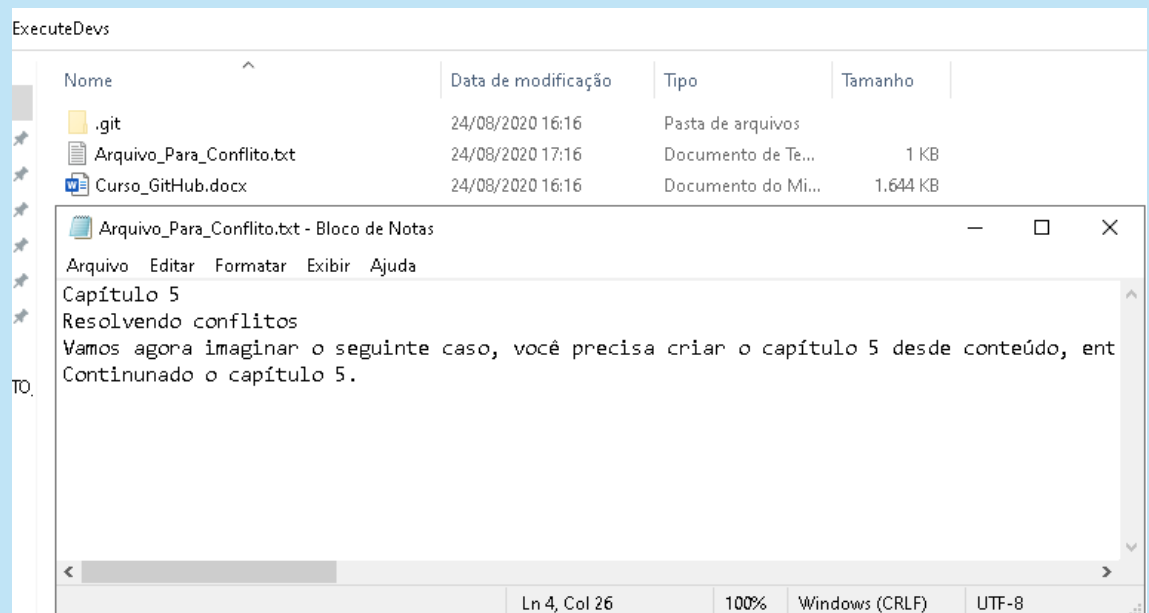
Resolvendo conflitos

Vamos agora imaginar o seguinte caso, você precisa criar o capítulo 5, então você cria uma branch chamada de **Cap_5_Adicao**, sendo assim você cria seu capítulo, mas em paralelo um colega também criou o capítulo 5 em outra branch, na hora que você foi efetuar o merge da **Cap_5_Adicao** para **Homolog** ele irá conflitar avisando que ambos lugares no arquivo foram editados.

Observação: Conflitos são comuns no dia a dia de desenvolvedores que trabalham com equipe grandes, mas é necessário tomar muito cuidado, as vezes um conflito, mas resolvido pode resultar em perda de códigos, textos etc.

Neste caso vamos criar um arquivo separado para não correremos risco de perder qualquer coisa no documento deste curso, vamos criar um arquivo chamado: **Arquivo_Para_Conflitos.txt**.

Neste arquivo vamos colocar um trecho do capítulo 5. Em seguida vamos trocar para a branch **Cap_5_Adicao** para alterarmos uma das linhas do trecho no arquivo.



Após adicionarmos o arquivo na branch Homolog vamos atualizar a branch Homolog e criar nossa branch **Cap_5_Adicao** contendo este novo arquivo.



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
        modified:   Curso_GitHub.docx

no changes added to commit (use "git add" and/or "git commit -a")

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git commit -m "Inicio capitulo 5"
[homolog 375d34c] Inicio capitulo 5
1 file changed, 0 insertions(+), 0 deletions(-)

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 32.45 KiB | 1.41 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
```

Após atualizarmos a branch **Homolog** vamos criar a branch **Cap_5_Adicao**

```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
$ git branch Cap_5_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git chekout Cap_5_Adicao
git: 'chekout' is not a git command. See 'git --help'.

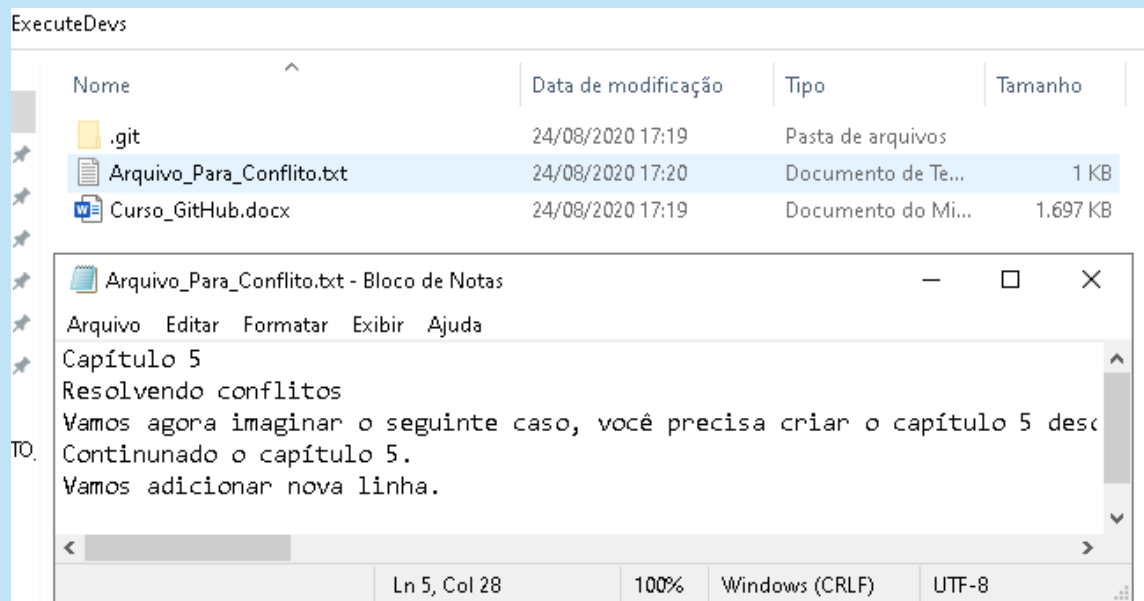
The most similar command is
    checkout

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git checkout Cap_5_Adicao
Switched to branch 'Cap_5_Adicao'

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git push --set-upstream origin Cap_5_Adicao
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'Cap_5_Adicao' on GitHub by visiting:
remote:   https://github.com/ExecuteDevs/ExecuteDevs/pull/new/Cap_5_Adicao
remote:
To https://github.com/ExecuteDevs/ExecuteDevs.git
 * [new branch]      Cap_5_Adicao -> Cap_5_Adicao
Branch 'Cap_5_Adicao' set up to track remote branch 'Cap_5_Adicao' from 'origin'.
```

Em seguida na branch **Cap_5_Adicao** vamos adicionar uma nova linha ao arquivo, reparem que a linha adicionada é de número 5.





Agora vamos efetuar **commit** e **push**, mas não vamos atualizar a branch **Homolog** utilizando o comando **merge**.

```

MINGW64:/c:/Users/Gustavo Nascimento/Desktop/ExecuteDevs
2 files changed, 2 insertions(+), 2 deletions(-)
create mode 100644 ~$rso_GitHub.docx

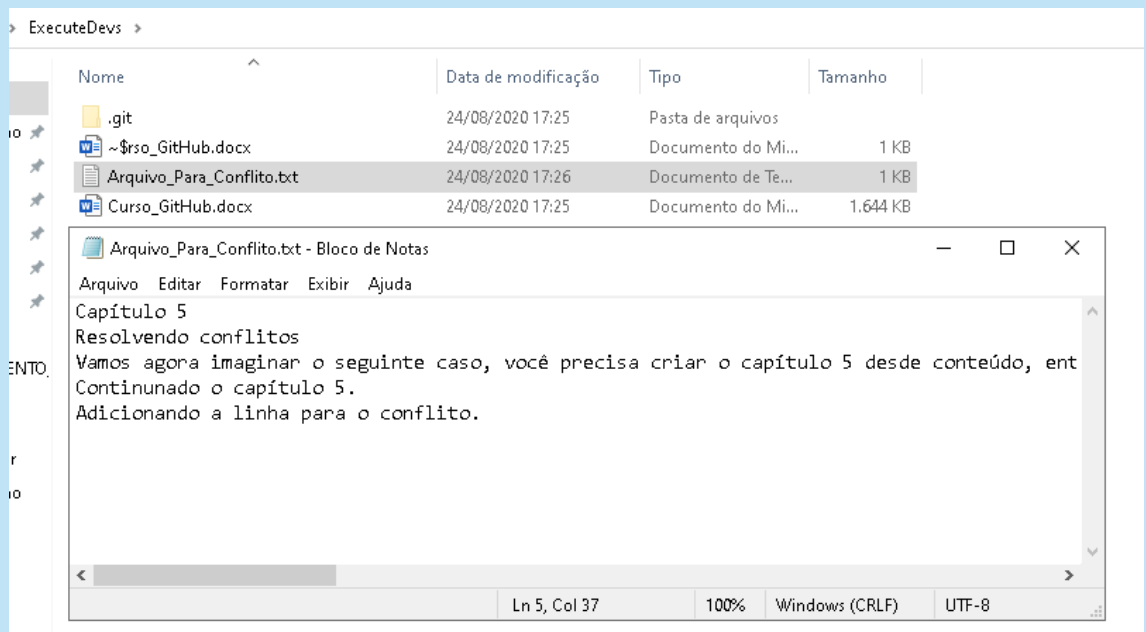
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git commit -m "Adicionando linha no arquivo de conflito"
[Cap_5_Adicao e0b1513] Adicionando linha no arquivo de conflito
3 files changed, 1 insertion(+)
delete mode 100644 ~$rso_GitHub.docx

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 88.07 KiB | 3.67 MiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ExecuteDevs/ExecuteDevs.git
  87088eb..e0b1513  Cap_5_Adicao -> Cap_5_Adicao
  
```

Ao invés de atualizarmos a branch **Homolog** com o comando **merge**, vamos trocar para a branch **Homolog**, nosso arquivo não irá conter a linha **"Vamos adicionar nova linha"**, então vamos adicionar outro texto na linha 5. Neste caso: **"Adicionando a linha para o conflito"**





Após adicionarmos a linha para conflito, vamos apenas efetuar o **commit** e **push** em seguida alterar para branch **Cap_5_Adicao**.

```
MINGW64/c:/Users/Gustavo Nascimento/Desktop/ExecuteDevs
Your branch is up to date with 'origin/homolog'.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git commit -m "Adicionando linha do conflito"
[homolog b227433] Adicionando linha do conflito
1 file changed, 1 insertion(+)

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 407 bytes | 407.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
01176d8..b227433 homolog -> homolog

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ |
```

Após a branch de **Homolog** estar atualizada vamos alterar para a branch **Cap_5_Adicao**, onde vamos efetuar um merge da **Homolog** para **Cap_5_Adicao**, normalmente efetuamos o inverso, mas desta vez vamos resolver o conflito e normalmente indica-se resolver conflitos na ramificação (branch) diferente da **Master** e **Homolog**.



```
MINGW64/c:/Users/Gustavo Nascimento/Desktop/ExecuteDevs
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 407 bytes | 407.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
   01176d8..b227433  homolog -> homolog

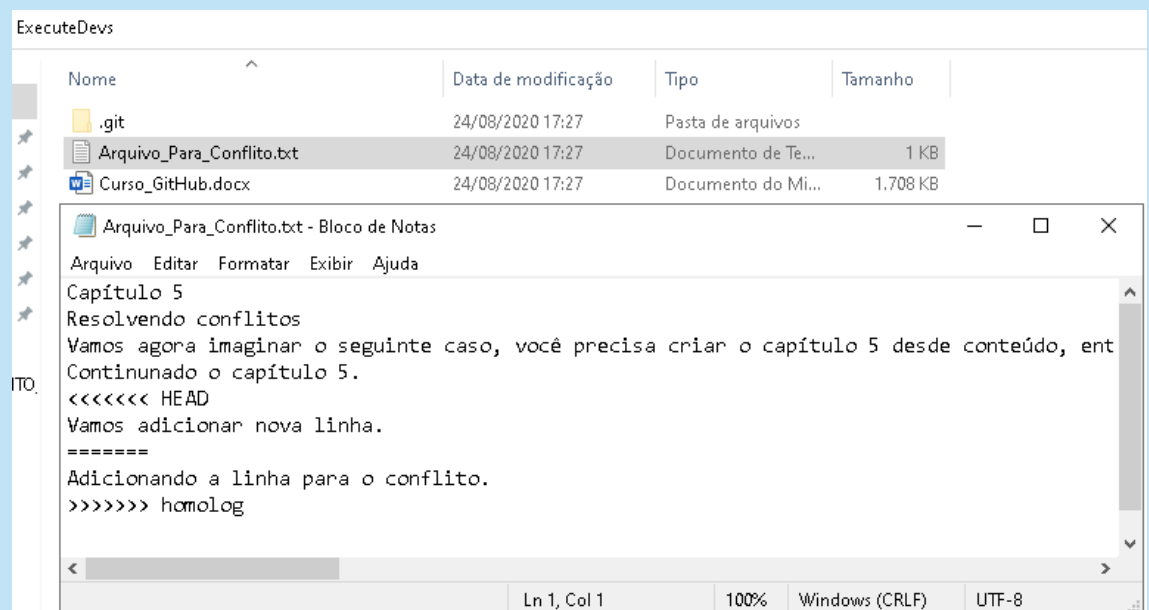
Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (homolog)
$ git checkout Cap_5_Adicao
Switched to branch 'Cap_5_Adicao'
Your branch is up to date with 'origin/Cap_5_Adicao'.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git merge homolog
Auto-merging Arquivo_Para_Conflito.txt
CONFLICT (content): Merge conflict in Arquivo_Para_Conflito.txt
Automatic merge failed; fix conflicts and then commit the result.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao|MERGING)
$ |
```

Efetuamos o merge utilizando **git merge homolog** desta forma vamos conseguir trazer as alterações existentes na Homolog para Cap_5_Adicao. Mas não vamos conseguir ter um merge com sucesso o git irá nos informar que houve uma falha para efetuar o merge, porque existe conflitos no arquivo, aonde ele encontrou um texto diferente no arquivo Arquivo_Para_Conflitos.txt na linha 5 de ambos.

Após o git avisar que houve conflito, vamos entrar no arquivo para verificarmos como esse conflito é identificado e em paralelo vamos resolver o conflito.

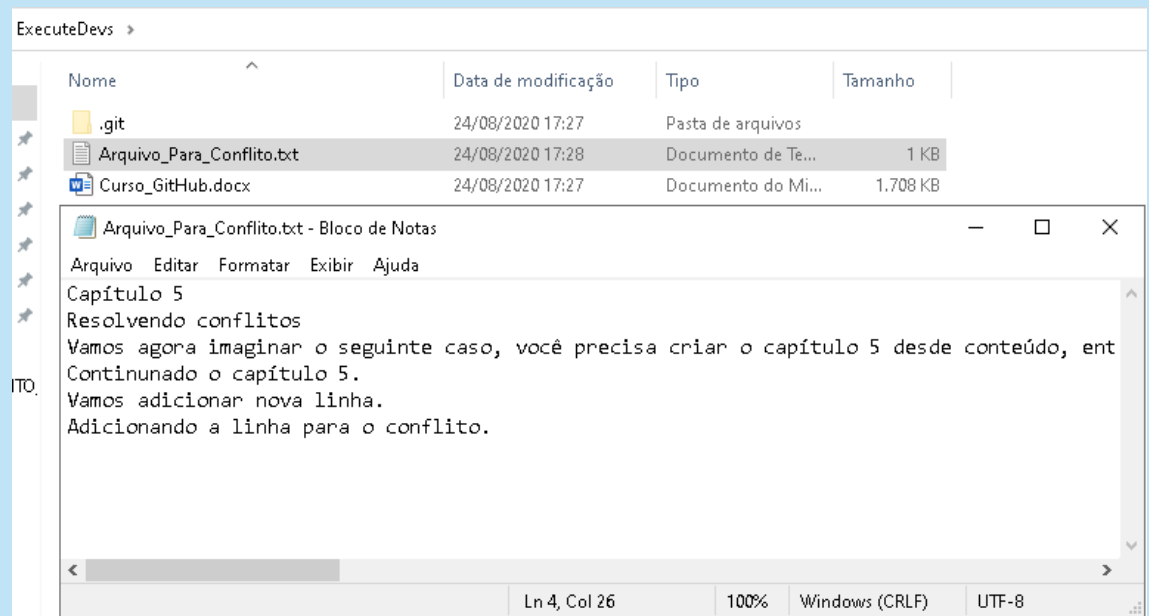


Normalmente o Git demarca a diferença de um arquivo para outro mostrando o Head (Aonde você está) e a branch que você solicitou, neste caso se analisarmos o <<<<<<< **Head** está mostrando conteúdo que existe no seu arquivo: Vamos adicionar nova linha.

E separando com ===== para mostrar abaixo a diferença:
Adicionando a linha para o conflito.

E identificando no final de onde vem essa alteração >>>> **Homolog**.

Para resolvermos o conflito vamos retirar a linha <<< **HEAD** juntamente os ===== e >>> homolog.



Então vamos solucionar os conflitos mesclando as duas linhas, vamos deixar uma na linha 5 outra na linha 6.

Após resolvermos conflitos vamos efetuar o commit e push.



```
MINGW64:/c/Users/Gustavo Nascimento/Desktop/ExecuteDevs
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao|MERGING)
$ git add .

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao|MERGING)
$ git commit -m "Resolvendo conflito"
[Cap_5_Adicao 668ec91] Resolvendo conflito

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 404 bytes | 404.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ExecuteDevs/ExecuteDevs.git
   e0b1513..668ec91  Cap_5_Adicao -> Cap_5_Adicao

Gustavo Nascimento@DSK-Gustavo MINGW64 ~/Desktop/ExecuteDevs (Cap_5_Adicao)
$
```



Capítulo 6

Comandos

No capítulo 6 vamos conhecer a maior parte dos comandos existentes para serem utilizados no GitHub, mostrando suas funcionalidades e mostrando momentos que podem ser utilizados.

- **Git init**
git init é um comando que inicia um repositório vazio, onde cria a pasta oculta .git com alguns subdiretórios.
- **Git push**
git push é um comando que atualiza o repositório remoto com as alterações locais em sua máquina.
- **Git pull**
git pull é um comando utilizado para pegar todas alterações que estão na branch aonde você se encontra.
- **Git checkout**
git checkout é um comando para atualizar seus arquivos igual está no índice, ou seja, se temos uma branch que deriva da master ao efetuar git checkout todos arquivos vão estar atualizados de acordo com a master, mas note que ficará igual se você tiver clonado recentemente
- **Git checkout <nome_branch>**
git checkout <nome_da_branch> utilizamos para navegar entre branches
- **Git add .**
git add . é um comando que utilizamos atualiza o índice utilizando o conteúdo atual encontrado na árvore de trabalho para preparar o conteúdo para o próximo commit.



- **Git status**
git status podemos verificar quais alterações encontram-se naquela branch que estamos
- **Git log**
git log utilizamos para verificar os commits efetuados dentro da branch em que se encontra.
- **Git clone**
git clone serve apenas quando temos um repositório já existente, a partir desta regra você pode obter este repositório em sua máquina para poder efetuar contribuições.
- **Git merge <nome_branch>**
git merge <nome_branch> atualiza seu branch com a branch seleciona.
- **Git config -global user.name**
git config -global user.name utilizamos para configurar seu usuário para efetuar subidas ao repositório.
- **Git revert**
git revert utilizamos para reverter commits efetuados na branch e podendo reverter específicos
- **Git rebase**
git rebase tem uma função parecida com git merge, mas existe uma lógica diferente nele, ele pega os commits efetuados em outra branch e adiciona na sua, os seus commits passam a serem os mais recentes.
- **Git fetch**
Busque as ramificações ou as tags (coletivamente, refs) de um ou mais repositórios, juntamente com os objetos necessários para completar seus históricos. As ramificações de rastreamento remoto



são atualizadas (consulte a descrição de <refspec> abaixo para conhecer maneiras de controlar esse comportamento).

- **Git tag**
Da mesma forma que a maioria dos VCSs, o Git tem a habilidade de marcar pontos específicos na história como sendo importantes. Normalmente as pessoas usam essa funcionalidade para marcar pontos onde foram feitas releases (v1.0 e assim por diante). Nessa sessão, você irá aprender como listar as tags existentes, como criar tags e quais são os diferentes tipos de tags.
- **Touch stash**
Use `git stash` quando quiser registrar o estado atual do diretório de trabalho e do índice, mas quiser voltar para um diretório de trabalho limpo. O comando salva suas modificações locais e reverte o diretório de trabalho para corresponder ao HEAD commit.
- **Touch stash pop**
comando adiciona os arquivos novamente na mudança da branch permitindo assim que você efetue o commit novamente e um push
- **Git cherry-pick**
Aplica as alterações de commit na branch atual.
- **Git blame**
comando utilizado para mostrar alterações feitas em um arquivo, por linha de comando, mostra o autor do commit.



- Git bisect

Faz buscas binárias no commits, utilizado normalmente para commits antigos aonde a busca pelo código não é fácil de se encontrar.

