



CURSO DE GITHUB - BÁSICO

Ministrado por:

- Matheus Vieira
- Allan Macedo

by NUAGE
IT's everywhere.

Parte 1

GIT

GIT O QUE É?

GIT é um sistema de controle de versão distribuída, rápido e escalável!

Em resumo, o GIT é um versionador de arquivos, sendo utilizado principalmente para gerenciar versões de softwares desenvolvidos por um ou mais desenvolvedores, com ele podemos implementar novas funcionalidades e tudo é registrado em um histórico, o qual podemos retroceder sempre que necessário.

GIT
GITHUB



Você sabia? O GIT foi desenvolvido pelo criador do Linux, o Linus Torvalds, pois tinha a necessidade de controlar a versão do kernel do Linux.



GIT

O QUE É?

```
intellipaatt@DESKTOP-SPC6JQB MINGW64 ~  
$ cd ProjectGit1  
  
intellipaatt@DESKTOP-SPC6JQB MINGW64 ~/ProjectGit1  
$ git init  
Initialized empty Git repository in C:/Users/intellipaatt/ProjectGit1/.git/  
  
intellipaatt@DESKTOP-SPC6JQB MINGW64 ~/ProjectGit1 (master)  
$ |
```

GIT
GITHUB



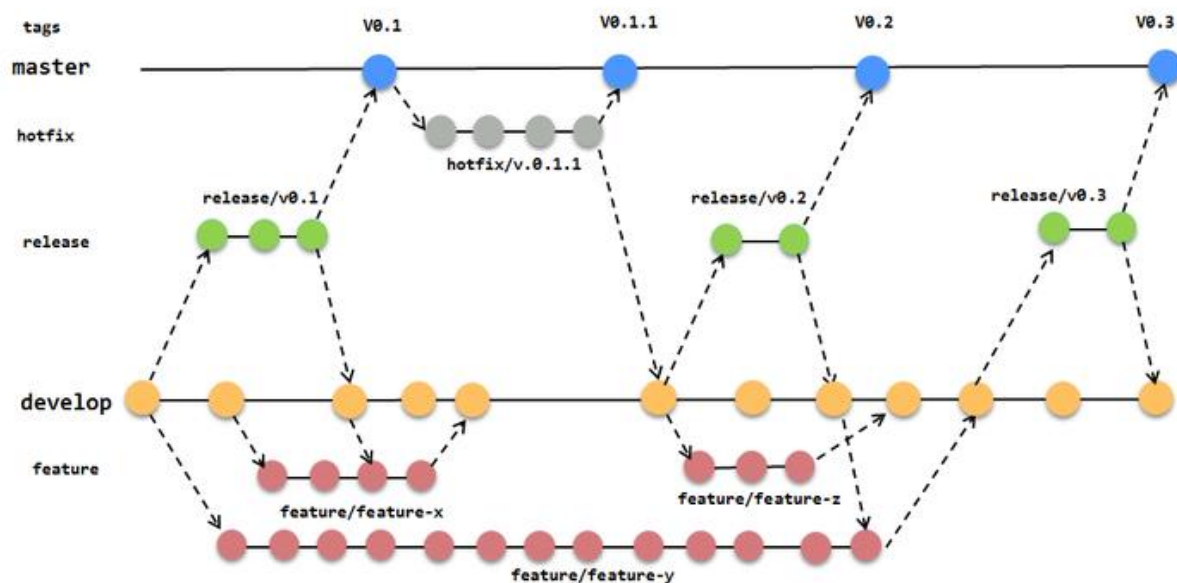
O git pode ser utilizado via terminal, podendo ser instalado em ambientes Linux e derivados, Windows e MacOs.



GIT GIT FLOW

O Git Flow é uma maneira dinâmica e ativa de organização de branches dentro do Git.

Ele se destaca por auxiliar na organização do versionamento de códigos, sendo muito utilizado por equipes de desenvolvimento de software.

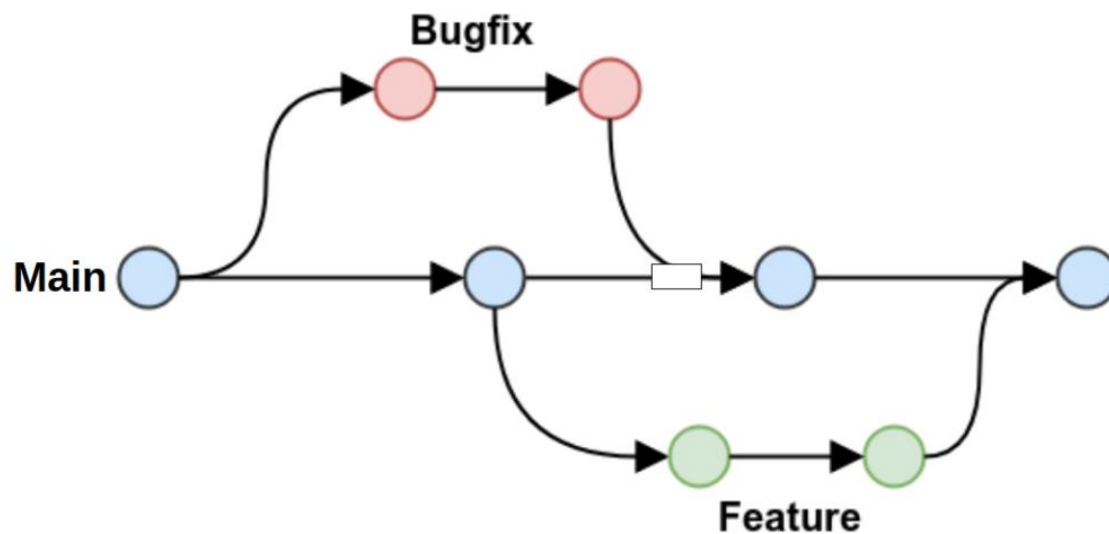


GIT
GITHUB



O Git Flow é uma maneira dinâmica e ativa de organização de branches dentro do Git.

GIT GIT FLOW



GIT
GITHUB



A branch Main é a cópia fiel do sistema em produção, um espelho de todo o sistema que está operando em produção



GIT

PRINCIPAIS TERMOS

- **Repository:** Local onde fica todos os arquivos do projeto, inclusive os históricos e versões.
- **Commit:** Coleção de alterações realizadas, é como se fosse um “checkpoint” do seu projeto sempre que necessário você pode retroceder até algum commit.
- **Branch:** É uma ramificação do seu projeto, cada Branch representa uma versão do seu projeto, e podemos seguir uma linha de desenvolvimento a partir de cada branch.
- **Fork:** Basicamente é uma bifurcação, uma cópia de um projeto existente para seguir em uma nova direção.

GIT
GITHUB



Já jogaram algum jogo que possui um checkpoint? Ou seja, um ponto de salvamento, onde você pode voltar a qualquer momento caso o personagem morra?



GIT

PRINCIPAIS TERMOS

- **Version (tag):** Cria um registro histórico do código imutável que pode ser uma nova versão ou release.
- **Merge:** É a capacidade de incorporar alterações do git, onde acontece uma junção das branches.

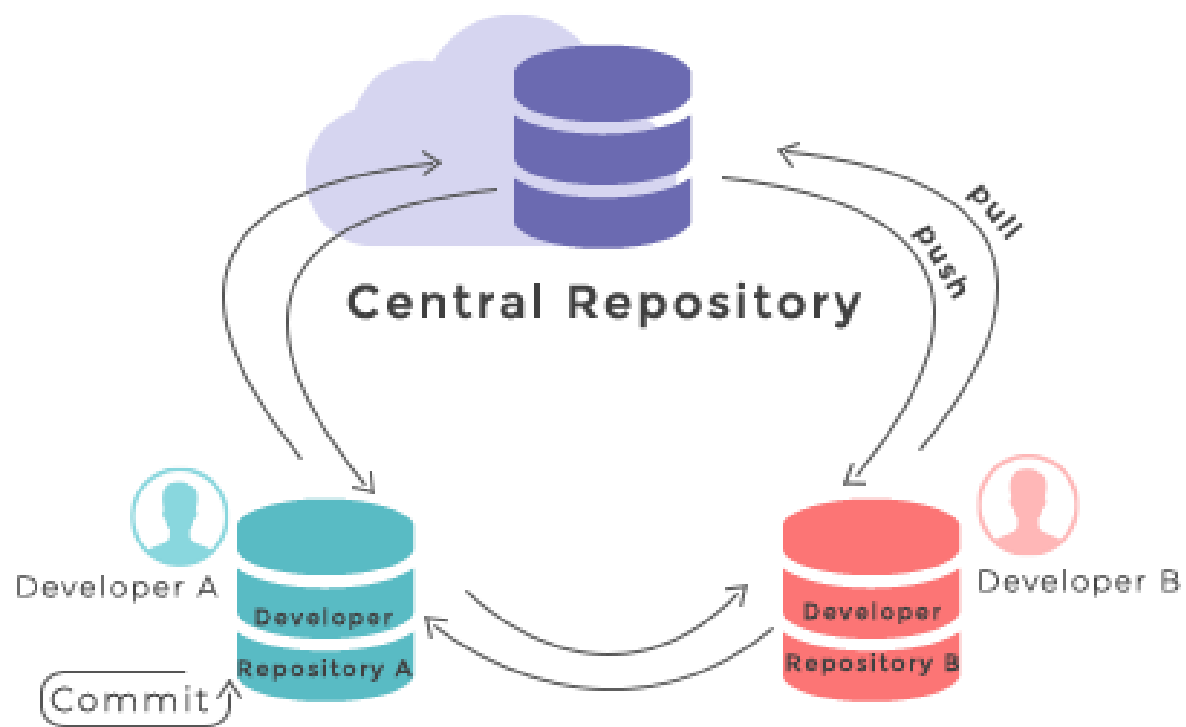
GIT
GITHUB



Já repararam que os aplicativos de celular e demais programas possuem atualizações constantes e suas releases e versões são alteradas?



GIT REPOSITORY



GIT
GITHUB



Os repositórios podem ser tanto locais, ou seja, no computador dos desenvolvedores, quanto na internet.

GIT COMMIT

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT
MESSAGES GET LESS AND LESS INFORMATIVE.

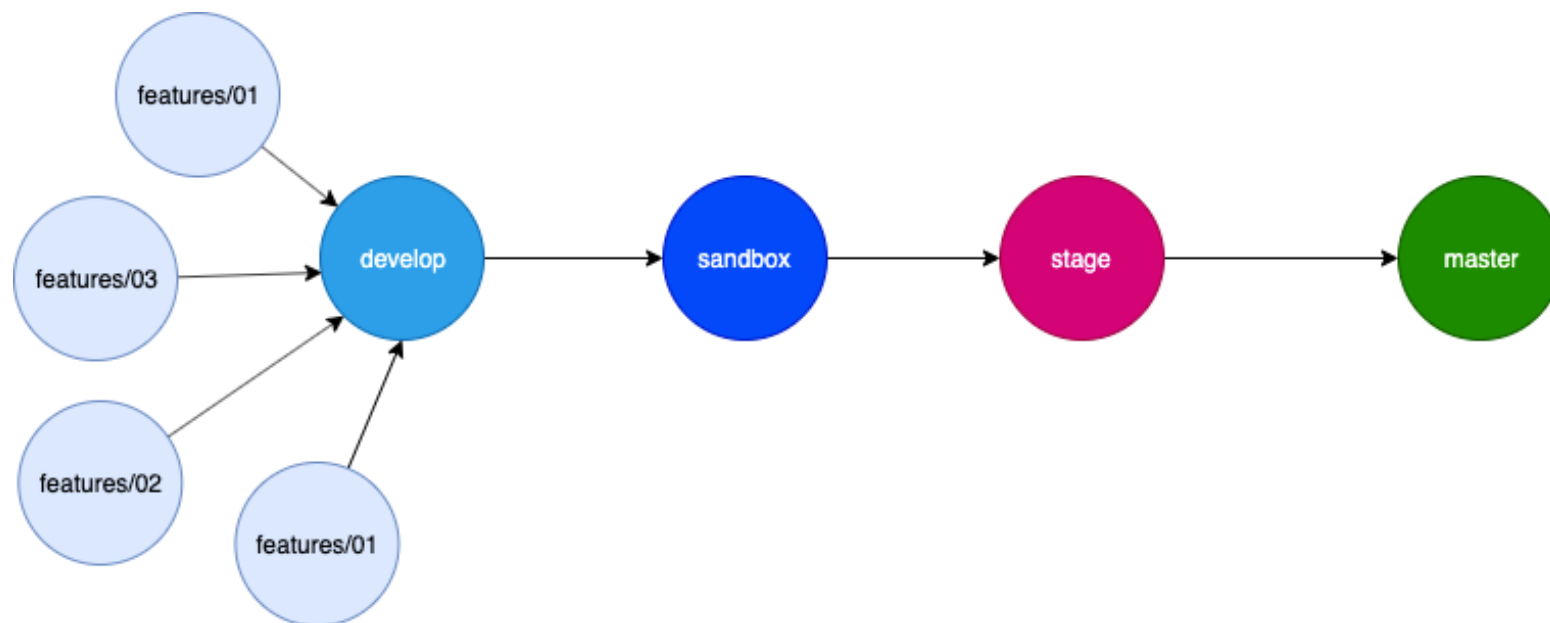


Os “saves” do commit são acompanhados por um comentário, que deverá de forma curta e objetiva, informar o motivo do commit.

GIT
GITHUB



GIT BRANCH



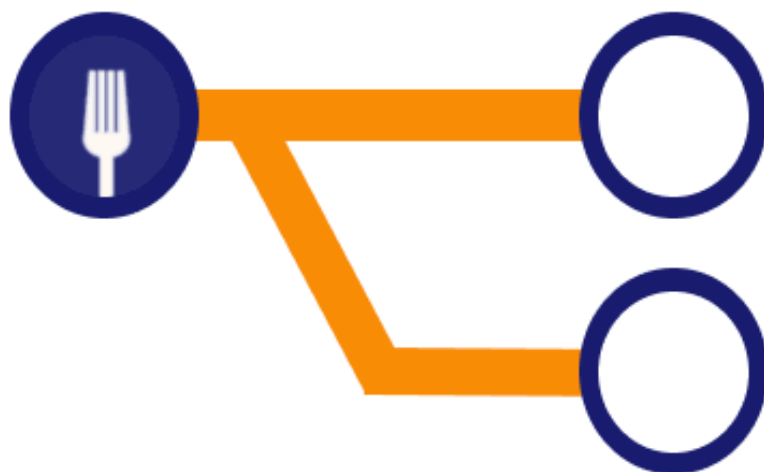
GIT
GITHUB



As branches são divididas em estágios do programa, desde seu desenvolvimento, etapas de testes e validação, até sua entrega em produção.



GIT FORK



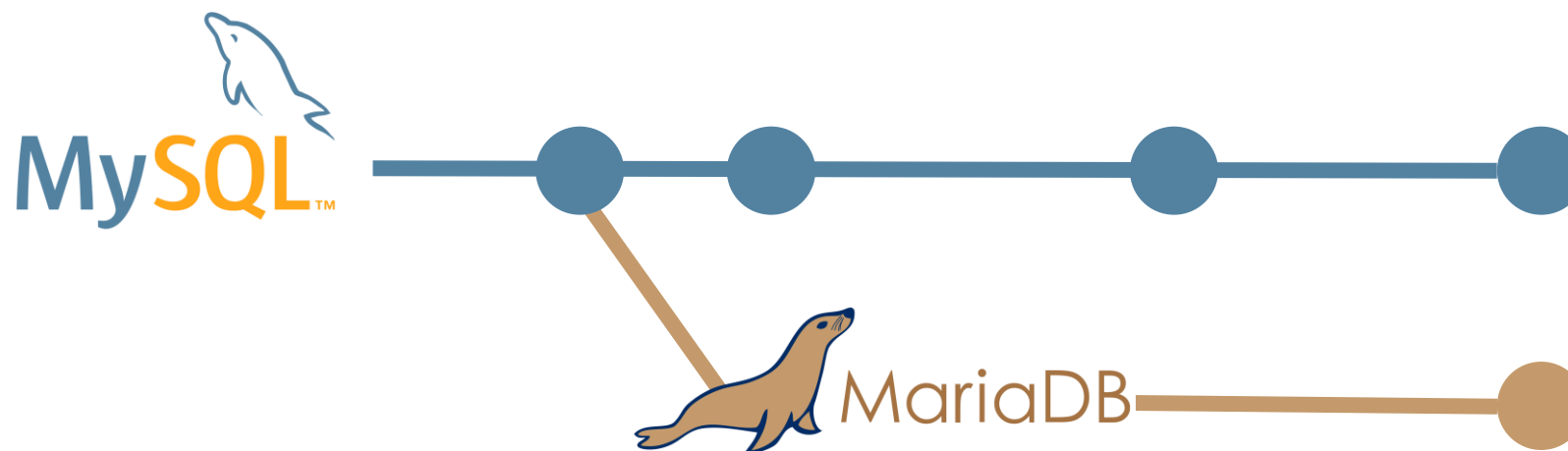
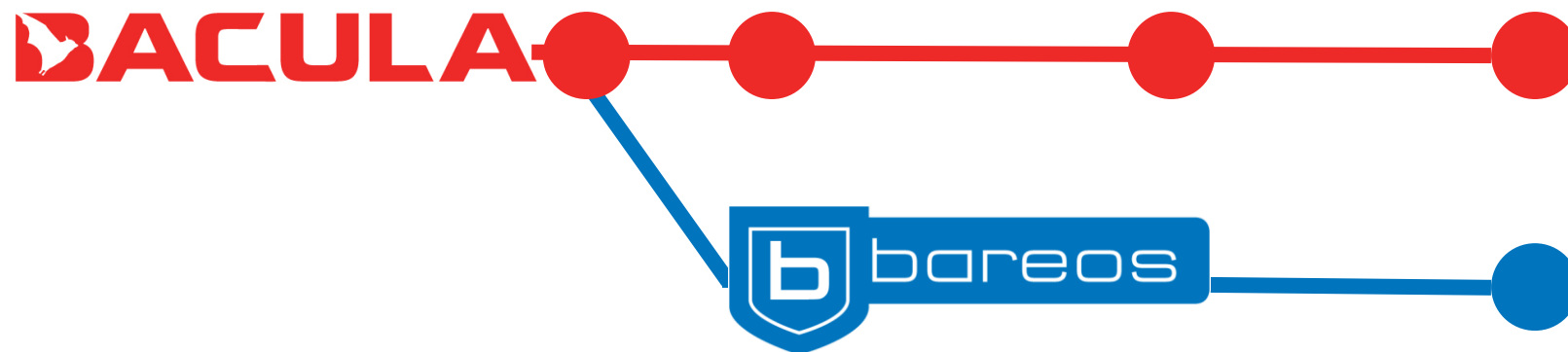
GIT
GITHUB



No FORK, um produto origina outro ou múltiplos produtos, que seguem suas evoluções de forma independente



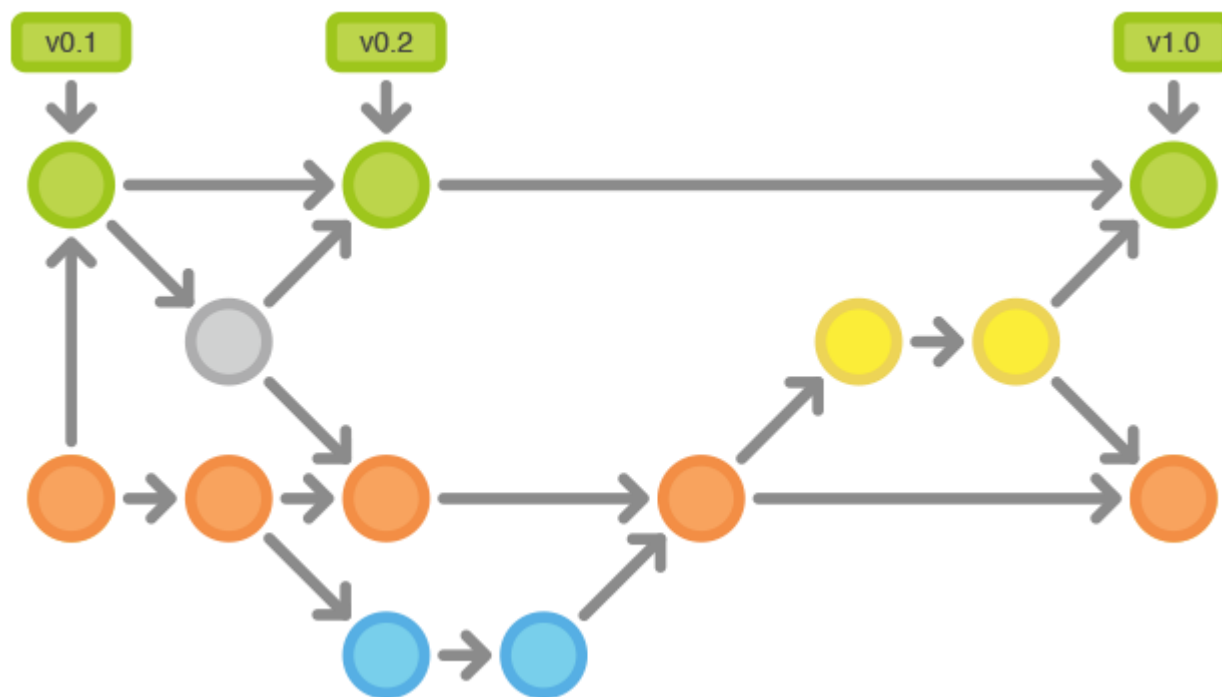
GIT EXEMPLOS FORK



GIT
GITHUB



GIT VERSION (TAG)

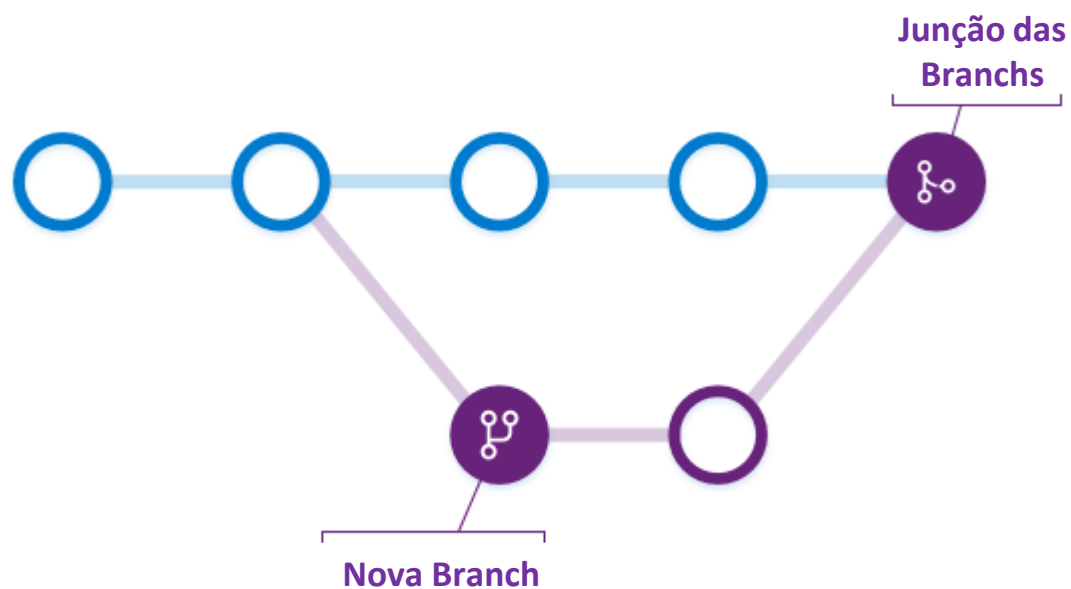


GIT
GITHUB



Pequenas mudanças nas aplicações são chamadas de releases, enquanto que grandes mudanças ou acréscimos de novas funcionalidades são chamadas de versões.

GIT MERGE



Os repositórios podem ser tanto locais, ou seja, no computador dos desenvolvedores, quanto na internet.

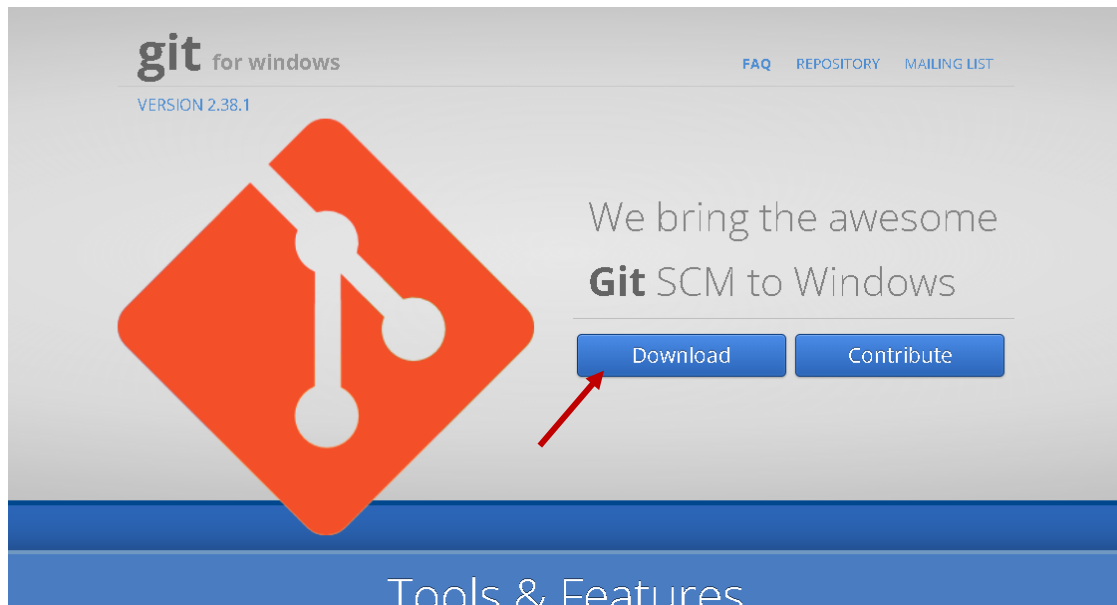


GIT

VAMOS PARA PRÁTICA?

- 1) Passo: Instalação do GIT

Para realizar a instalação do GIT no Windows, acesse o site <https://gitforwindows.org/> e clique no botão “Download”.



GIT
GITHUB

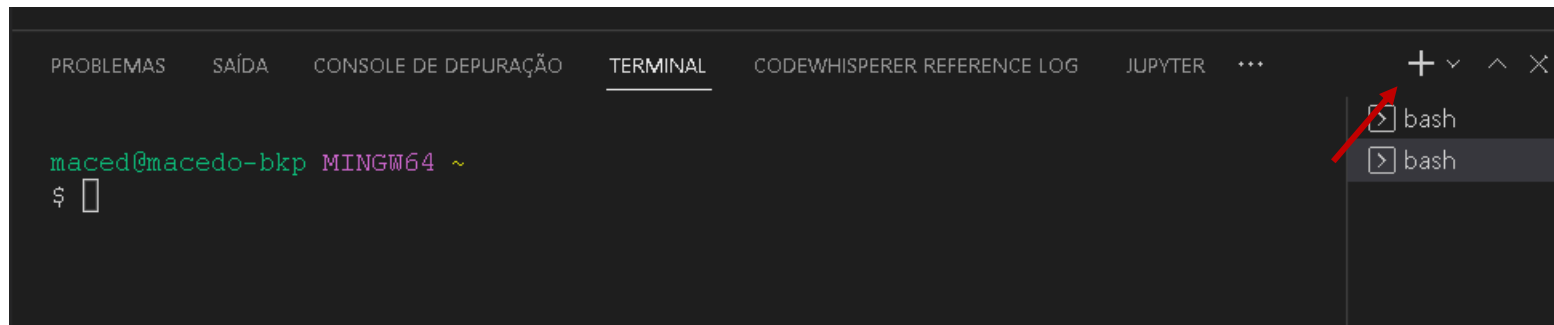


GIT

VAMOS PARA PRÁTICA?

- 2) Passo: Abrir o terminal pelo VSC

Abra o Visual Studio Code, clique no menu “Terminal” e clique na opção “Novo Terminal”. Após aberto, selecione o tipo de terminal “GIT Bash”.



GIT
GITHUB



GIT



Exercício 1

1. Crie uma pasta para o seu projeto do GIT com o comando:
`mkdir nome_pasta`
2. Acesse a pasta criada no terminal com o comando:
`cd nome_pasta`
3. Inicie um repositório para o git na pasta criada com o comando:
`git init .`

Exemplo da mensagem que irá aparecer:

```
maced@macedo-bkp MINGW64 ~/Project1 (master)
$ git init .
Initialized empty Git repository in C:/Users/maced/Project1/.git/

maced@macedo-bkp MINGW64 ~/Project1 (master)
$
```

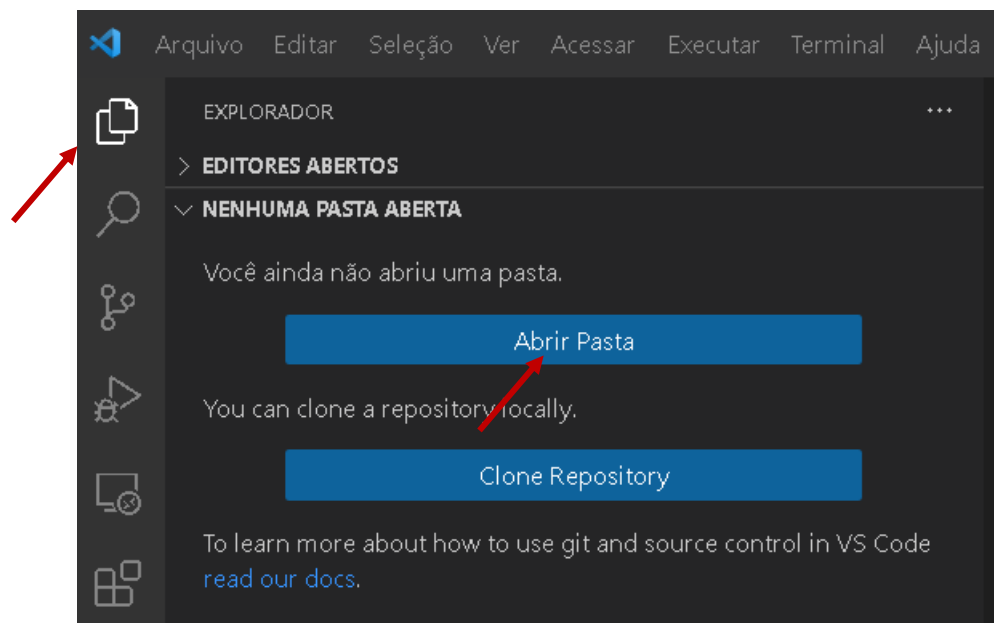


GIT



Exercício 1

4. Abra a pasta que você criou pelo terminal no Visual Code Studio:



GIT



Exercício 1

5. Crie os arquivos html, Javascript e css dentro da pasta
6. Abra o terminal novamente e selecione o “GIT Bash”
7. Vamos criar o seu checkpoint dos arquivos criados. Digite o comando no terminal:
git add .
git commit -am “create files”

```
maced@macedo-bkp MINGW64 ~/Project1 (master)
$ git add .

maced@macedo-bkp MINGW64 ~/Project1 (master)
$ git commit -am "create files"
[master (root-commit) 6654058] create files
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 html.css
create mode 100644 script.js
create mode 100644 stylesheet.css
```



GIT



Exercício 1

8. Agora vamos ver o que fizemos. Digite o comando:
git log

```
maced@macedo-bkp MINGW64 ~/Project1 (master)
$ git log
commit 6654058a75636c6dc6902b2c134e12e6286d4d47 (HEAD -> master)
Author: allanqueirozm <allanqueirozm@gmail.com>
Date:   Sat Nov 19 00:33:59 2022 -0300

    create files
```

9. Agora respondam, quais termos foram aplicados nesse exercício até o momento?



GIT

EXEMPLO GIT LOG

```
$ git log
commit bcb792dcc7dfbfcfd620ee73ed7422295f3d50ca (HEAD -> computer_player, origin/computer_player)
Author: lpenzey <lucaspenezymoog@gmail.com>
Date:   Fri Jul 27 15:19:27 2018 -0500

    cleaned formatting with rubocop

commit e953f0fdbfcf8038afec2a50f72c9d65601d346c
Author: lpenzey <lucaspenezymoog@gmail.com>
Date:   Fri Jul 27 14:55:41 2018 -0500

    updated script

commit d443cc147cf543bc2892a82143e3b0ab016f7847
Author: lpenzey <lucaspenezymoog@gmail.com>
Date:   Fri Jul 27 14:53:12 2018 -0500

    added travisci

commit bf0c6b5362ea8e675a6c866d388aee7867b816ea
Author: lpenzey <lucaspenezymoog@gmail.com>
Date:   Fri Jul 27 14:49:45 2018 -0500

    added travisci

commit ed75abbca061c2c93834d1ee606a1b665175e10d
Merge: 08aa658 a098936
Author: lpenzey <lucaspenezymoog@gmail.com>
Date:   Thu Jul 26 10:15:16 2018 -0500

    Merge branch 'save_resume_game' of https://github.com/lpenzey/Mastermind into save_resume_game
```

GIT
GITHUB



GIT

PRINCIPAIS COMANDOS

- **git init** inicia um repositório do git, como por exemplo, **git init** .
- **git add** é o comando que atualiza o nosso repositório do GIT com as modificações que realizamos em nosso computador, como por exemplo, **git add**
- **git commit** cria um checkpoint das nossas modificações no código e permite que seja inserido um comentário para explicarmos as mudanças, como por exemplo, **git commit -am "mensagem do commit"**
- **git clone** é um comando para baixar o código-fonte existente de um repositório remoto (como, por exemplo, o Github), **git clone <https://link-com-o-nome-do-repositório>**
- **git branch** é um comando para criar uma nova branch, como por exemplo, **git branch <nome-da-branch>**

GIT
GITHUB



Para acessar a documentação do git e entender melhor os comandos ou descobrir novos, acesse: <https://git-scm.com/docs>



GIT

PRINCIPAIS COMANDOS

- **git checkout** é um dos comandos do Git mais usados, pois permite que a gente “entre” na branch, como por exemplo, **git checkout <nome-da-branch>**
- **git status** nos dá todas as informações necessárias sobre a branch atual, como por exemplo, **git status**
- **git push** faz o upload dos commits com as modificações para o repositório remoto do GIT (GITHUB por exemplo), como **git push <repositório-remoto> <nome-da-branch>**
- **git pull** faz o download da última modificação do repositório remoto do GIT para o seu computador de trabalho, como **git pull <repositório-remoto>**
- **git revert** volta as suas alterações para um momento anterior, utilizando um checkpoint, como por exemplo, **git revert 3321844**

GIT
GITHUB



Para acessar a documentação do git e entender melhor os comandos ou descobrir novos, acesse: <https://git-scm.com/docs>



GIT

PRINCIPAIS COMANDOS

- **git merge** realiza a união/mesclagem de duas branches, como por exemplo, **git merge <nome-da-branch-destino>**
- **git log** exibe o histórico das modificações como commits e merges do repositório do GIT, como por exemplo, **git log**
- **git tag** é utilizado para registrar um ponto no histórico do código que é usado para marcar uma versão ou release, sendo imutável, como por exemplo, **git tag 2.0**
- **git help:** abre a documentação do comando informado, por exemplo, **git help commit**

GIT
GITHUB



Para acessar a documentação do git e entender melhor os comandos ou descobrir novos, acesse: <https://git-scm.com/docs>



GIT E AGORA?

Bom, agora você já tem um pequeno conhecimento sobre o que é GIT e como começar a usar. Mas como vamos utilizar essa ferramenta para trabalharmos em colaboração com o nosso time, de uma forma que o código não esteja presente apenas no nosso computador, mas sim que todos possam acessar e subir suas modificações e correções?

Ao mesmo tempo outras perguntas surgem, como:

- Como vou controlar quem tem acesso e permissão ao GIT?
- Como vou compartilhar meus projetos no GIT com o público da internet?
- Como vou automatizar o meu código para subir em produção?
- Como vou ter um repositório pros meus códigos altamente disponível e que não dependa do meu computador ligado?

...



A resposta para todas as perguntas é simples! Por mais que o GIT seja uma ferramenta muito poderosa e possa resolver todos os questionamentos acima, existem formas mais simples e que reduzem bastante o esforço do seu time.

GIT
GITHUB



Parte 2

GITHUB

GITHUB O QUE É?

GitHub é uma plataforma totalmente online onde você pode criar repositórios e hospedar neles seus projetos.

O GitHub permite a hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo.

GIT
GITHUB



Você sabia? O GITHUB foi comprado pela Microsoft por 7,5 bilhões de dólares em 2018, sendo uma plataforma com mais de 85 milhões de projetos e 28 milhões de desenvolvedores ativos.



GITHUB

O QUE É?

Início em
2008

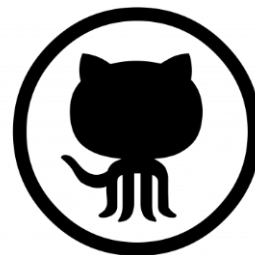
2013

Mascote
Octopuss



github
SOCIAL CODING

GitHub



GIT
GITHUB



GitHub



O logotipo atual representa Octocat - uma simbiose de um polvo e um gato.



GITHUB VANTAGENS

O GitHub serve, fundamentalmente, para facilitar o controle de versões de um software ou aplicação.

As diferenças entre ele e o GIT estão nas interações proporcionadas pelo GitHub: funcionando de modo semelhante a uma rede social, o GitHub é hoje um dos maiores pontos de encontro virtuais entre programadores de todo o mundo. E seus maiores benefícios são:

- Conexão com desenvolvedores de diversas partes do mundo
- Compartilhamento de projetos open-source
- Aprender programação na prática ao observar o avanço do desenvolvimento de aplicações de terceiros;
- Participar de discussões a respeito de novas tecnologias;
- Obter auxílio de outros programadores para resolver problemas relacionados a seus projetos;
- Controlar as diferentes versões de um código com armazenamento em nuvem;
- **Registrar ações e projetos desenvolvidos por você em uma espécie de portfólio online, etc.**

GIT
GITHUB



GITHUB

COMO COMEÇAR?

Para começarmos a utilizar o GITHUB primeiro precisamos realizar a criação de uma conta no site <https://github.com/>

Vamos para a prática?

- **1) Passo: Criação da conta**

Após acessar o site do GITHUB, clique na opção “Sign up”

GIT
GITHUB



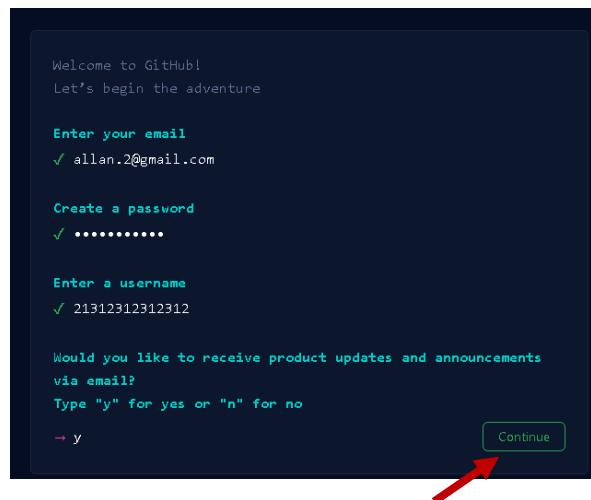
GITHUB

COMO COMEÇAR?

- 2) Passo: Criação da conta

- Insira o seu e-mail e clique no botão “Continue”;
- Crie uma senha e clique novamente no botão “Continue”;
- Digite um nome de usuário (apelido) e clique mais uma vez no botão “Continue”;
- Por fim, informe se você deseja receber anúncios do GITHUB por e-mail, digitando **y** para sim ou **n** para não, clique no botão “Continue” e resolva o captcha para finalizar.

GIT
GITHUB



```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ allan.2@gmail.com

Create a password
✓ .....

Enter a username
✓ 21312312312312

Would you like to receive product updates and announcements
via email?
Type "y" for yes or "n" for no
- y
```

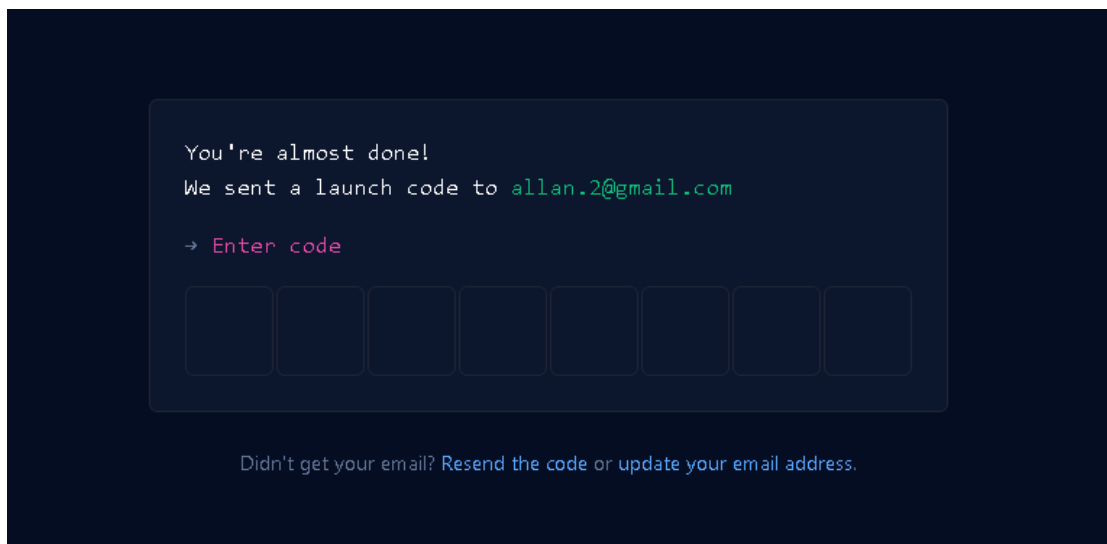
Continue



GITHUB

COMO COMEÇAR?

- **3) Passo: Confirmação da criação da conta**
 - Você irá receber um e-mail do GITHUB informando o código para finalização da criação da conta.



- Pronto! Sua conta foi criada com sucesso!

GIT
GITHUB

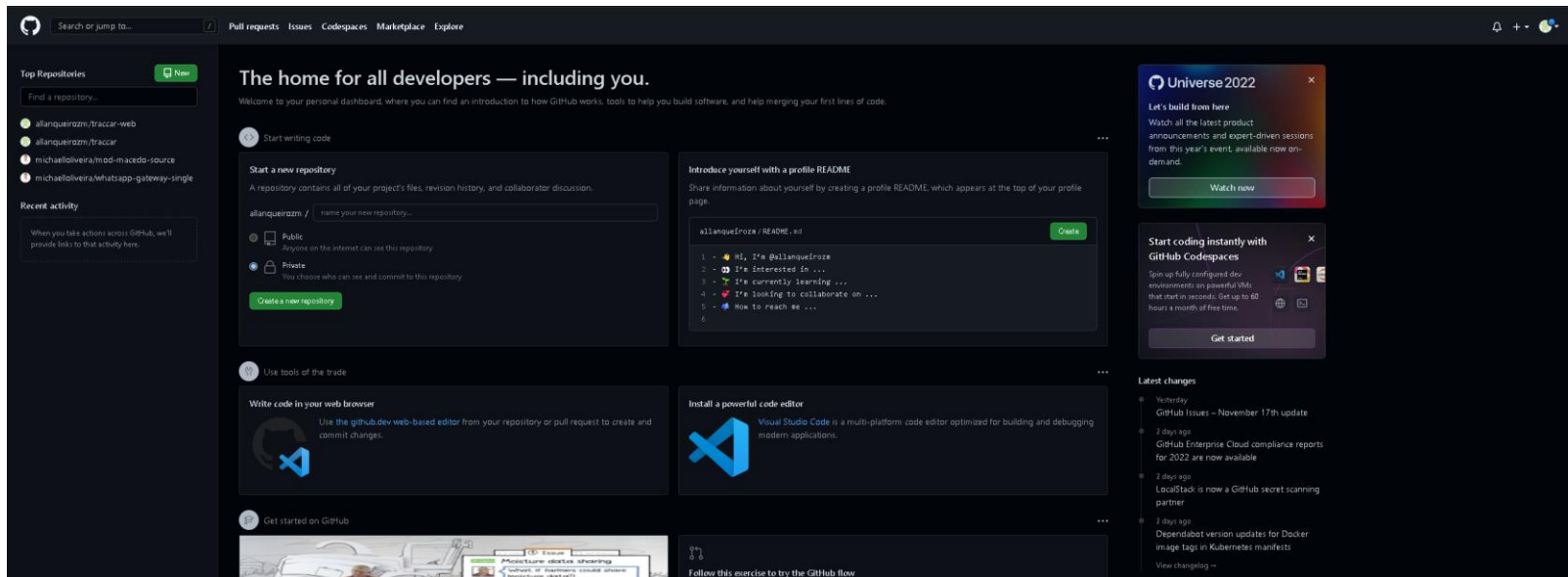


GITHUB COMO COMEÇAR?



```
git clone https://github.com/allanqueirozm/projecft022.git  
cd projecft022  
mkdir nome
```

GIT
GITHUB



GITHUB



Exercício 1

Este exercício iremos realizar em conjunto.

1. Crie um novo repositório no GITHUB
2. Clone o repositório em seu computador
3. Será solicitado as suas credenciais para autenticar no GITHUB.
Insira o seu usuário que senha que acabou de criar
4. Crie os arquivos HTML, JS e CSS no repositório clonado em seu computador
5. Suba os arquivos para o repositório do GITHUB e visualize-os acessando o seu projeto



GITHUB



Exercício 2

1. Clone o projeto de um colega
2. Insira um comentário dentro do arquivo HTML
3. Suba as modificações para o repositório do GITHUB do seu colega

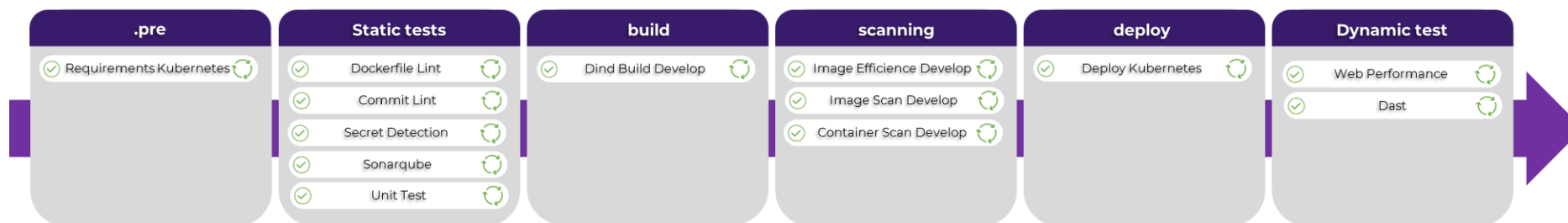


GITHUB

CONSIDERAÇÕES FINAIS E PRÁTICA

Entender o GIT e o GITHUB será fundamental para os desafios que serão passados no decorrer do dia, pois além de ser uma ferramenta muito poderosa, são utilizados por maior parte das empresas de desenvolvimento, assim como a NuageIT, permitindo que o desenvolvimento possa ser realizada de forma colaborativa, em equipes que podem ir de 1 a mais de 100 profissionais para atuar na mesma aplicação.

As funcionalidades do GITHUB vão muito além, permitindo que todo o processo de testes, segurança e entrega do código sejam automatizadas, além de padronizar a entrega das equipes.

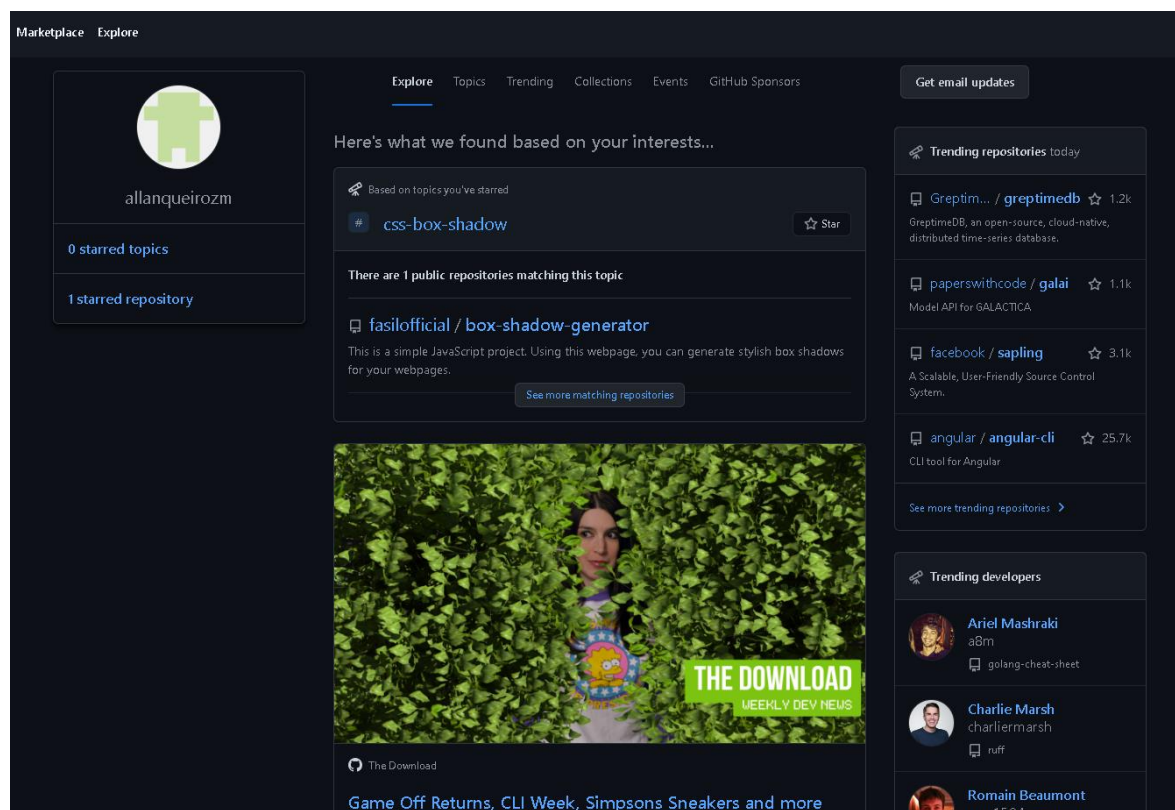


Esteira CI/CD da NuageIT

GITHUB

CONSIDERAÇÕES FINAIS E PRÁTICA

Agora vamos praticar mais um pouco no GITHUB e entender novos termos e funcionalidades que a plataforma trás.



OBRIGAD@



by NUAGE
IT's everywhere.