GitHub4Women

Domain 4: Modern Development

Agenda

1. GitHub Codespaces
2. GitHub Copilot
3. GitHub Actions

Codespaces

GitHub Copilot

GitHub Actions

Tudo que você precisa para iniciar seu projeto

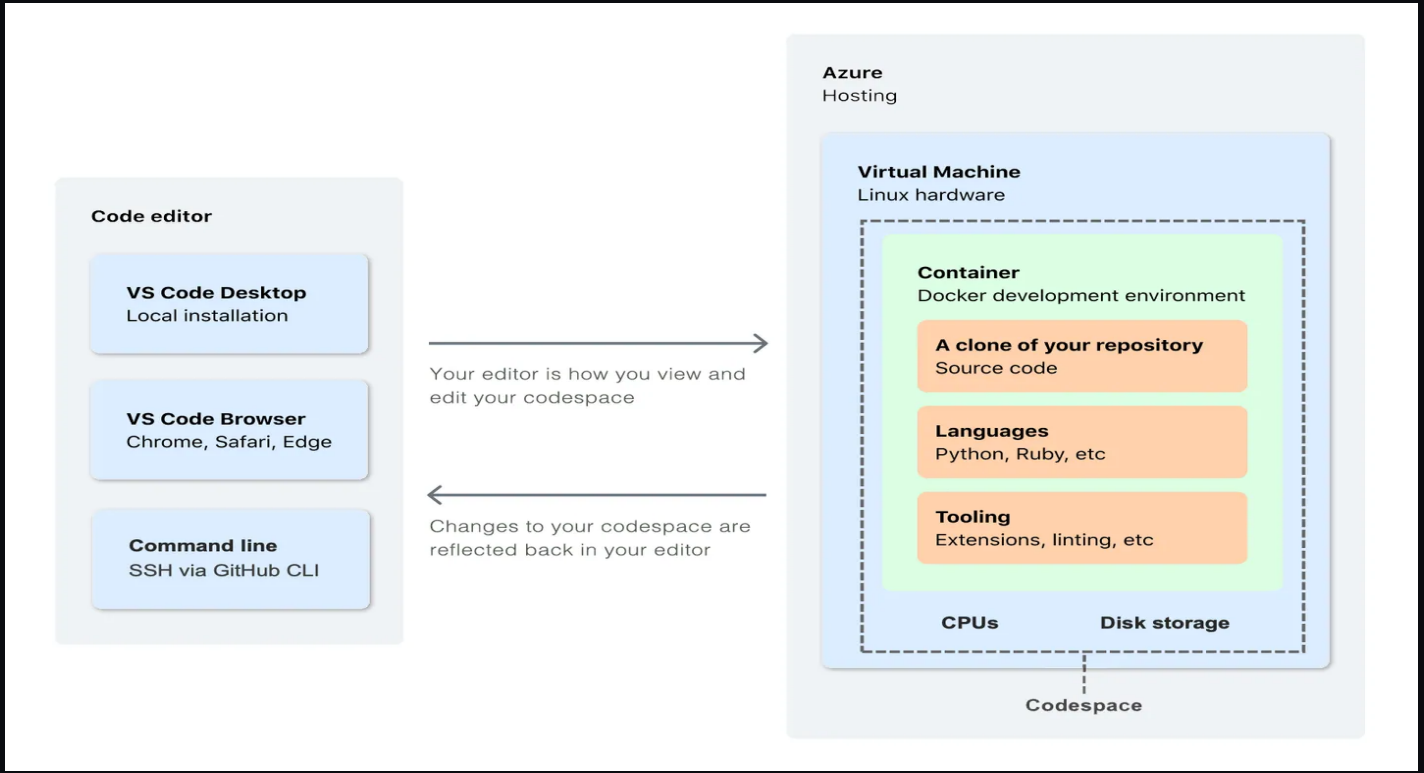
Obtenha o GitHub Copilot gratuitamente para estudantes:
https://aka.ms/copilot-estudantes

GitHub Codespaces

O que é GitHub Codespaces?

GitHub Codespaces é um ambiente de desenvolvimento online hospedado diretamente no GitHub. Ele permite que você desenvolva, execute e depure projetos diretamente no seu navegador web sem a necessidade de configurar um ambiente de desenvolvimento local.

O que é GitHub Codespaces?



Rápido e fácil de criar

Configurar ambientes de desenvolvimento hospedados na nuvem totalmente configurados em minutos

Trabalho remoto mais fácil com ambientes em nuvem

Trabalhe em vários projetos simultaneamente sem ser atrasado por tarefas de computação intensiva

Desenvolva onde quer que você esteja

Edite, execute e depure aplicativos de qualquer dispositivo com um editor que ofereça suporte a repositórios, extensões e CLI do Git

Como criar um Codespace?

Cada Codespace criado é hospedado pelo GitHub em um contêiner do Docker em execução em uma máquina virtual.

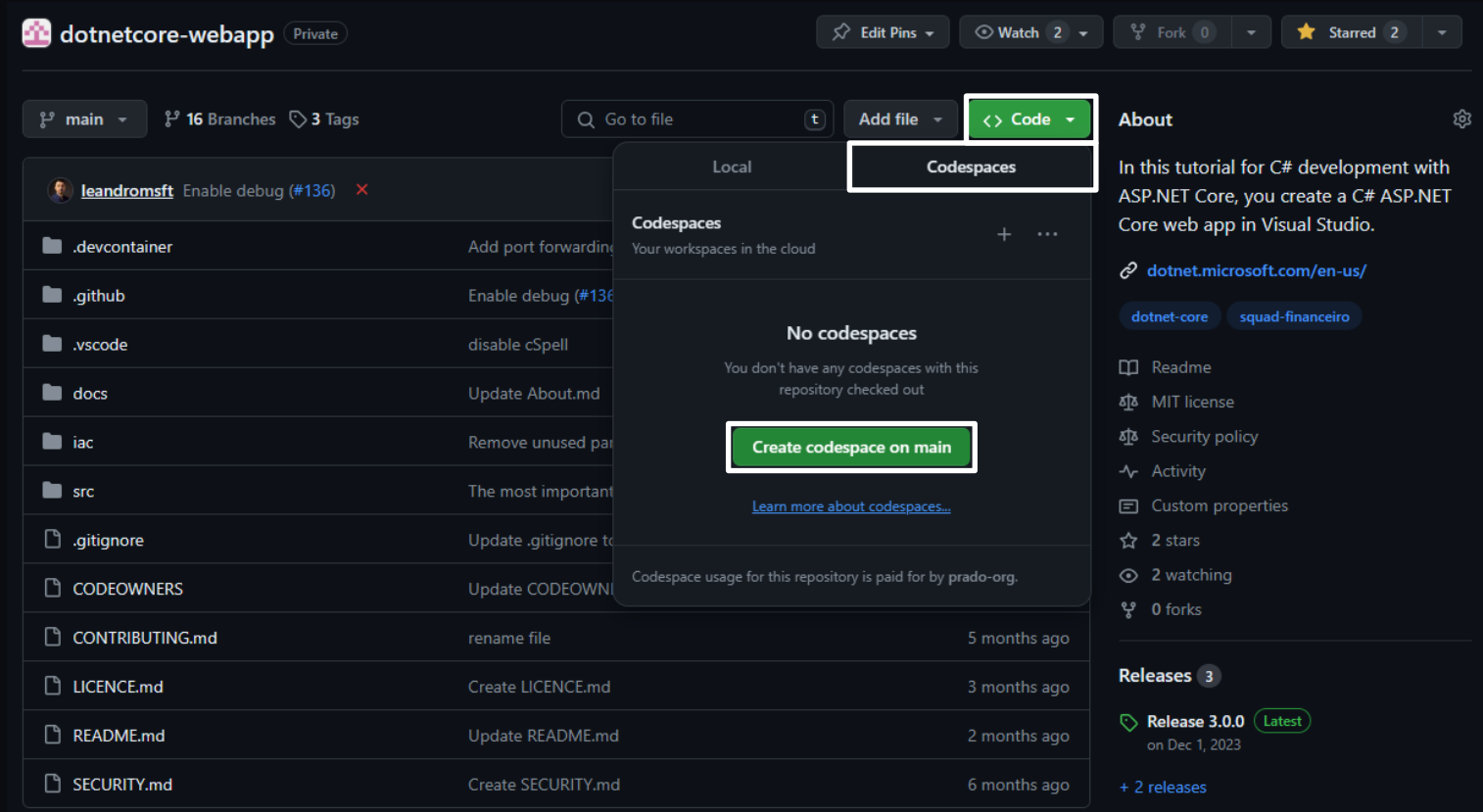
Criado com base em uma imagem do Ubuntu Linux que inclui uma seleção de linguagens e ferramentas populares.

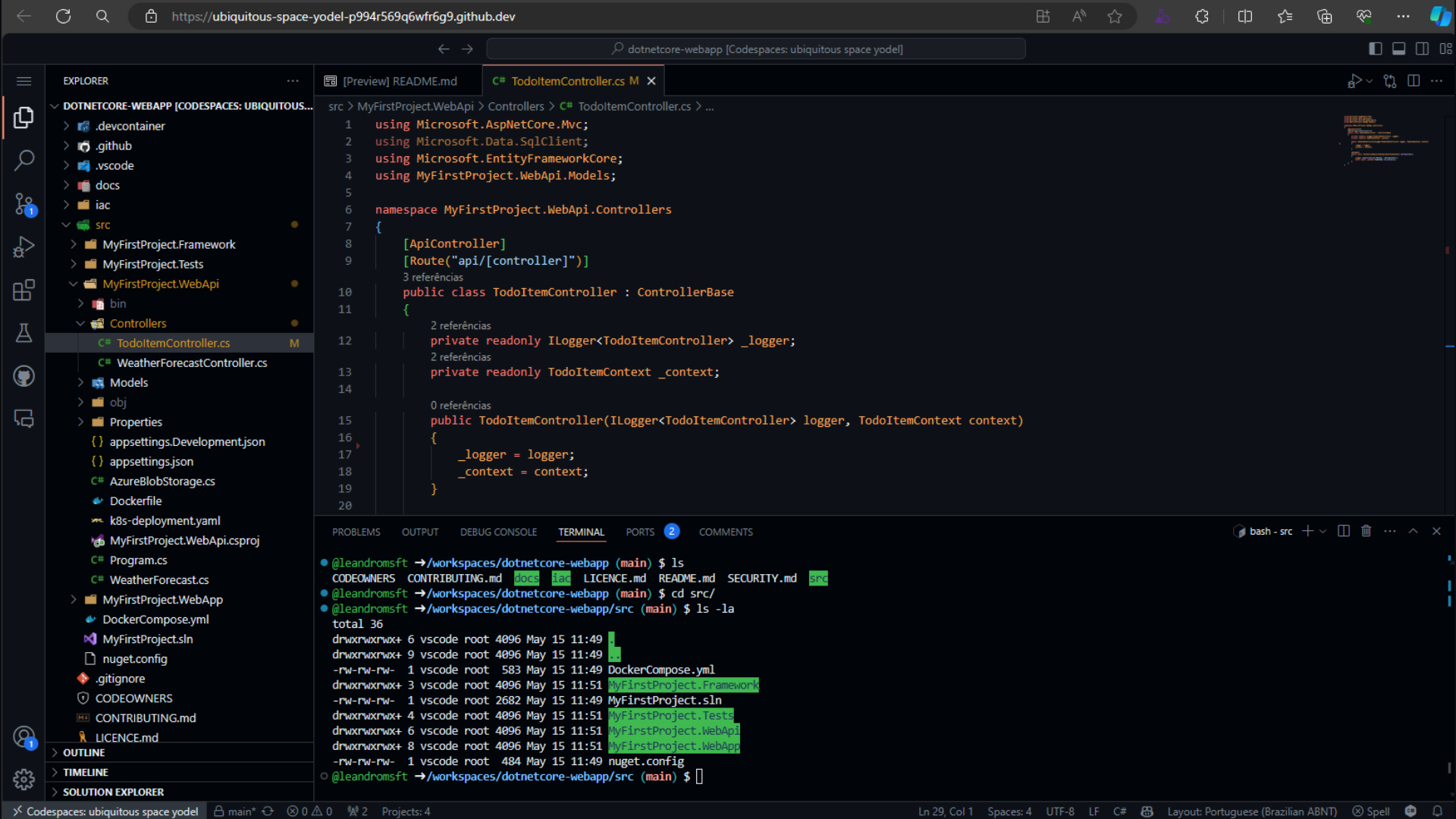
Você pode usar uma imagem com base em uma distribuição do Linux de sua escolha e configurá-la para requisitos específicos.

Independentemente do sistema operacional local, o Codespace será executado em um ambiente Linux.

Não há suporte para sistemas operacionais Windows e macOS para o contêiner de desenvolvimento remoto.

Iniciar um Codespaces




https://ubiquitous-space-yodel-p994r569q6wfr6g9.github.dev
dotnetcore-webapp [Codespaces: ubiquitous space yodel]
EXPLORER
DOTNETCORE-WEBAPP [CODESPACES: UBIQUITOUS...
.devcontainer
.github
.vscode
docs
iac
src
MyFirstProject.Framework
MyFirstProject.Tests
MyFirstProject.WebApi
bin
Controllers
TodoItemController.cs
WeatherForecastController.cs
Models
obj
Properties
appsettings.Development.json
appsettings.json
AzureBlobStorage.cs
Dockerfile
k8s-deployment.yaml
MyFirstProject.WebApi.csproj
Program.cs
WeatherForecast.cs
MyFirstProject.WebApp
DockerCompose.yml
MyFirstProject.sln
nuget.config
.gitignore
CODEOWNERS
CONTRIBUTING.md
LICENCE.md
OUTLINE
TIMELINE
SOLUTION EXPLORER
[Preview] README.md
TodoItemController.cs M
src > MyFirstProject.WebApi > Controllers > TodoItemController.cs > ...
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using MyFirstProject.WebApi.Models;
namespace MyFirstProject.WebApi.Controllers
{
[ApiController]
[Route("api/[controller]")]
3 referências
public class TodoItemController : ControllerBase
{
2 referências
private readonly ILogger<TodoItemController> _logger;
2 referências
private readonly TodoItemContext _context;
0 referências
public TodoItemController(ILogger<TodoItemController> logger, TodoItemContext context)
{
_logger = logger;
_context = context;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS 2
COMMENTS
bash - src
@leandromsft →/workspaces/dotnetcore-webapp (main) $ ls
CODEOWNERS CONTRIBUTING.md docs iac LICENCE.md README.md SECURITY.md src
@leandromsft →/workspaces/dotnetcore-webapp (main) $ cd src/
@leandromsft →/workspaces/dotnetcore-webapp/src (main) $ ls -la
total 36
drwxrwxrwx+ 6 vscode root 4096 May 15 11:49 .
drwxrwxrwx+ 9 vscode root 4096 May 15 11:49 ..
-rw-rw-rw- 1 vscode root 583 May 15 11:49 DockerCompose.yml
drwxrwxrwx+ 3 vscode root 4096 May 15 11:51 MyFirstProject.Framework
-rw-rw-rw- 1 vscode root 2682 May 15 11:49 MyFirstProject.sln
drwxrwxrwx+ 4 vscode root 4096 May 15 11:51 MyFirstProject.Tests
drwxrwxrwx+ 6 vscode root 4096 May 15 11:51 MyFirstProject.WebApi
drwxrwxrwx+ 8 vscode root 4096 May 15 11:51 MyFirstProject.WebApp
-rw-rw-rw- 1 vscode root 484 May 15 11:49 nuget.config
@leandromsft →/workspaces/dotnetcore-webapp/src (main) $
Codespaces: ubiquitous space yodel
main*
Ln 29, Col 1
Spaces: 4
UTF-8
LF
C#
Layout: Portuguese (Brazilian ABNT)
Spell

Customizando um Codespaces

Se você não definir um “**devcontainer**”, o GitHub Codespaces usará uma configuração padrão.

Os arquivos de configuração para um “**devcontainer**” estão contidos em um diretório **.devcontainer** em seu repositório.

devcontainer.json

You, 4 weeks ago | 2 authors (You and others)

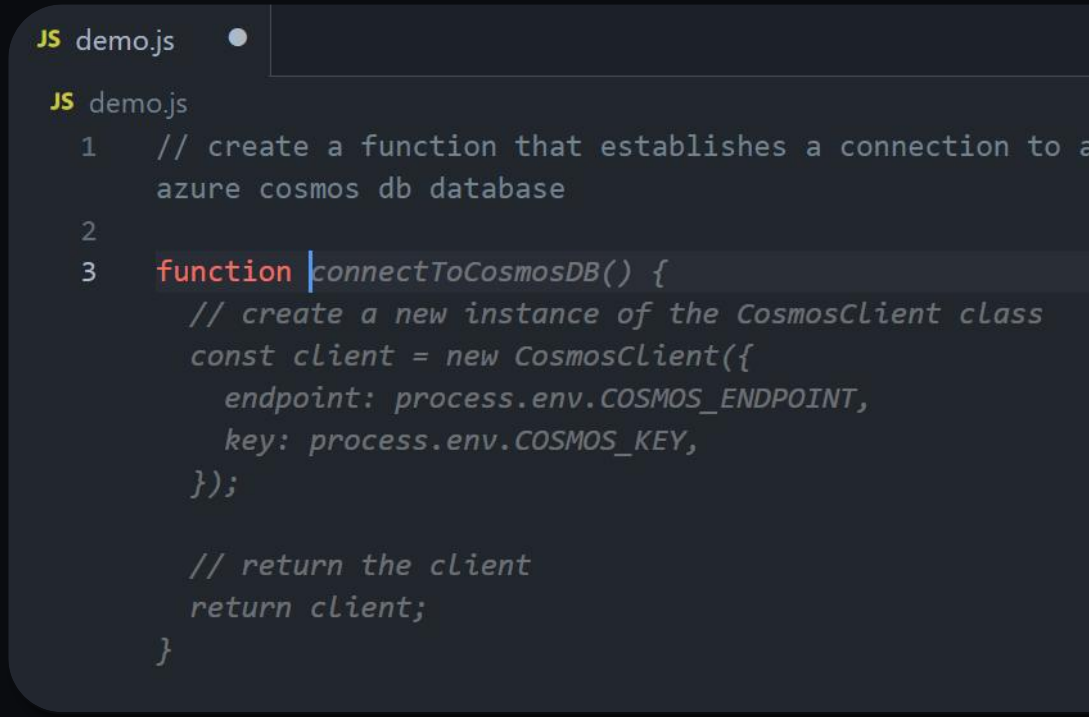
```jsonc
{
    "name": "C# (.NET)",
    "image": "mcr.microsoft.com/devcontainers/dotnet:1-6.0",

    // Features to add to the dev container. More info: https://containers.dev/features.
    "features": {
        "ghcr.io/devcontainers/features/azure-cli:1": {},
        "ghcr.io/devcontainers/features/dotnet:2": {}
    },

    // Use 'forwardPorts' to make a list of ports inside the container available locally.
    "forwardPorts": [5260, 5261],
    "portsAttributes": {
        "5260": {
            "protocol": "http",
            "label": "APP",
            "onAutoForward": "notify"
        },
        "5261": {
            "protocol": "http",
            "label": "API",
            "onAutoForward": "notify"
        }
    },

    // Use 'postCreateCommand' to run commands after the container is created.
    "postCreateCommand": ".devcontainer/postCreateCommand.sh",

    // Configure tool-specific properties.
    "customizations": {
        // Configure properties specific to VS Code.
        "vscode": {
          // Add the IDs of extensions you want installed when the container is created.
          "extensions": [
            "streetsidesoftware.code-spell-checker",
            "vscode-icons-team.vscode-icons",
            "ms-dotnettools.csdevkit",
            "GitHub.copilot",
            "GitHub.copilot-chat"
          ]
        }
      }
}
```

Demo

GitHub Copilot

GitHub Copilot

Focado em fornecer uma experiência para o desenvolvedor que permita que as pessoas façam seu melhor trabalho.



Copilot para todo o ciclo de desenvolvimento

Principais funcionalidades

Do IDE, CLI, e dispositivo móvel

- **Sugestões de código**
- **Chat**
- **Documentar o código com comentários**
- **Criar testes unitários**
- **Fazer perguntas para aprender sobre um novo código**
- **Acelerar o aprendizado de uma nova linguagem ou estrutura de programação**

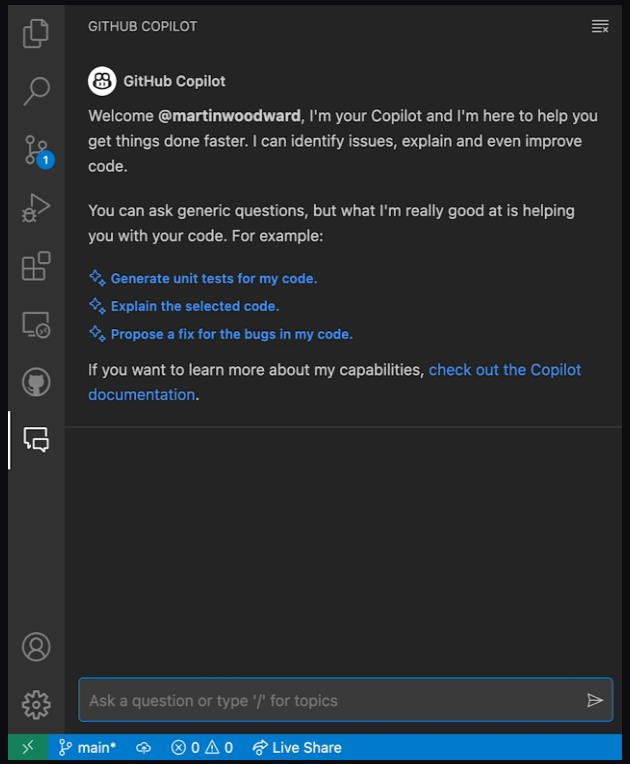
Para a platforma

- **Iniciar uma conversa sobre sua base de código**
- **Copilot Chat no GitHub.com**
- **Resumos de Pull Request**
- **Bases de conhecimento**
- **Copilot no GitHub Mobile**

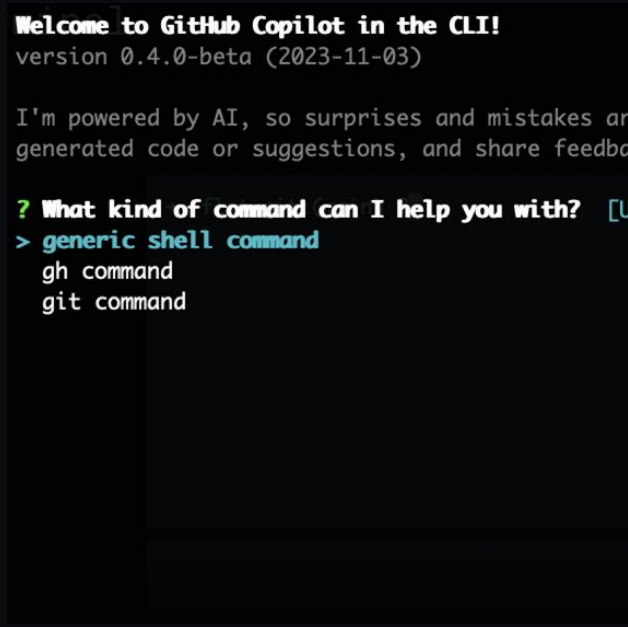
GitHub Copilot

Extensão

Chat



CLI



Demo

GitHub Actions

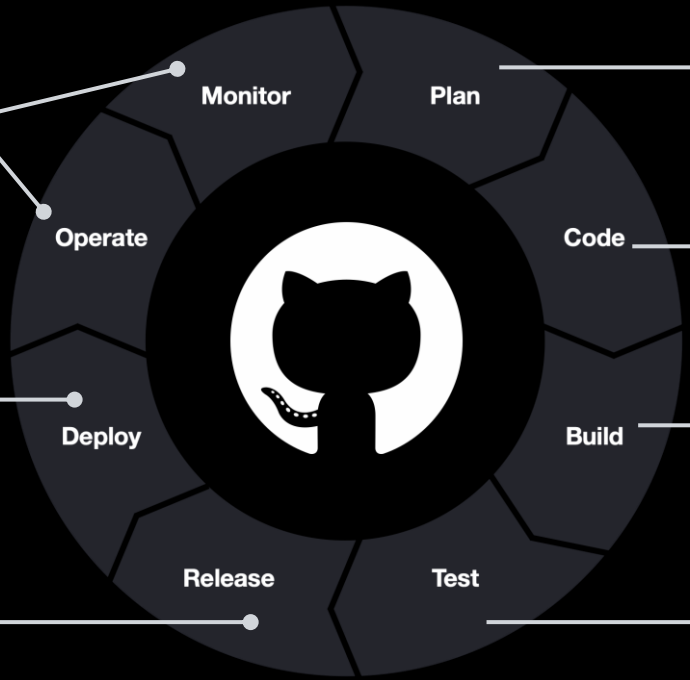
O que é GitHub Actions

O GitHub Actions é um produto do GitHub que permite **automatizar seus fluxos de trabalho.**

- Os fluxos de trabalho são armazenados como arquivos yml
- Totalmente integrado com o GitHub
- Responde aos eventos do GitHub
- Logs em tempo real e visualização da execução do fluxo de trabalho
- Fluxos de trabalho baseados na comunidade
- Runners hospedados no GitHub ou auto-hospedados
- Cofre de segredos incorporado

Casos de uso em todo o seu cilco

- Não é adequado para



- Gerencie as issues
- Boards de Projetos

- Code review
- Auto-merge
- Auto-labeling

- Implantar Aplicações

- Compilação e build
- Guardar artefatos

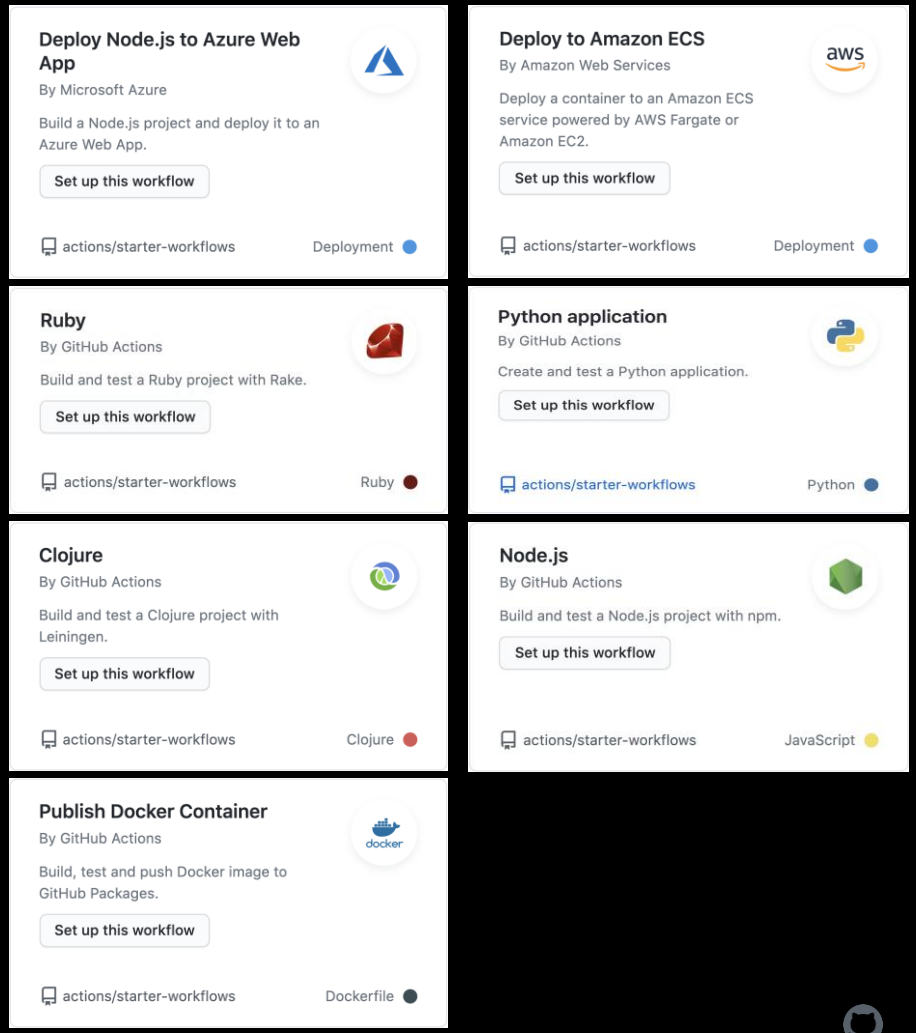
- Publicar versão
- Notas de versão
- Carregar artefatos

- Testes automatizados
- Linters
- CodeQL
- Consumir pacotes

Iniciando com workflows

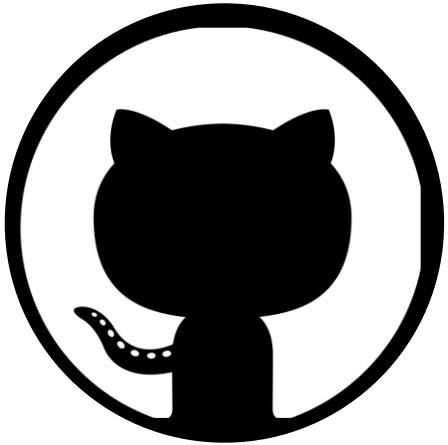
- Pré-configurado para linguagens e estruturas específicas

- O GitHub analisa seu código e sugere os fluxos de trabalho com base em sua linguagem e estrutura



GitHub Actions é mais do que CI/CD

- Mecanismo genérico de workflow
- Automatize tudo com workflows
- 35 eventos podem acionar um workflow
- GitHub Token e permissões do Workflow
- Workflows criados pela comunidade
- Qualquer plataforma, qualquer linguagem, qualquer *provider*

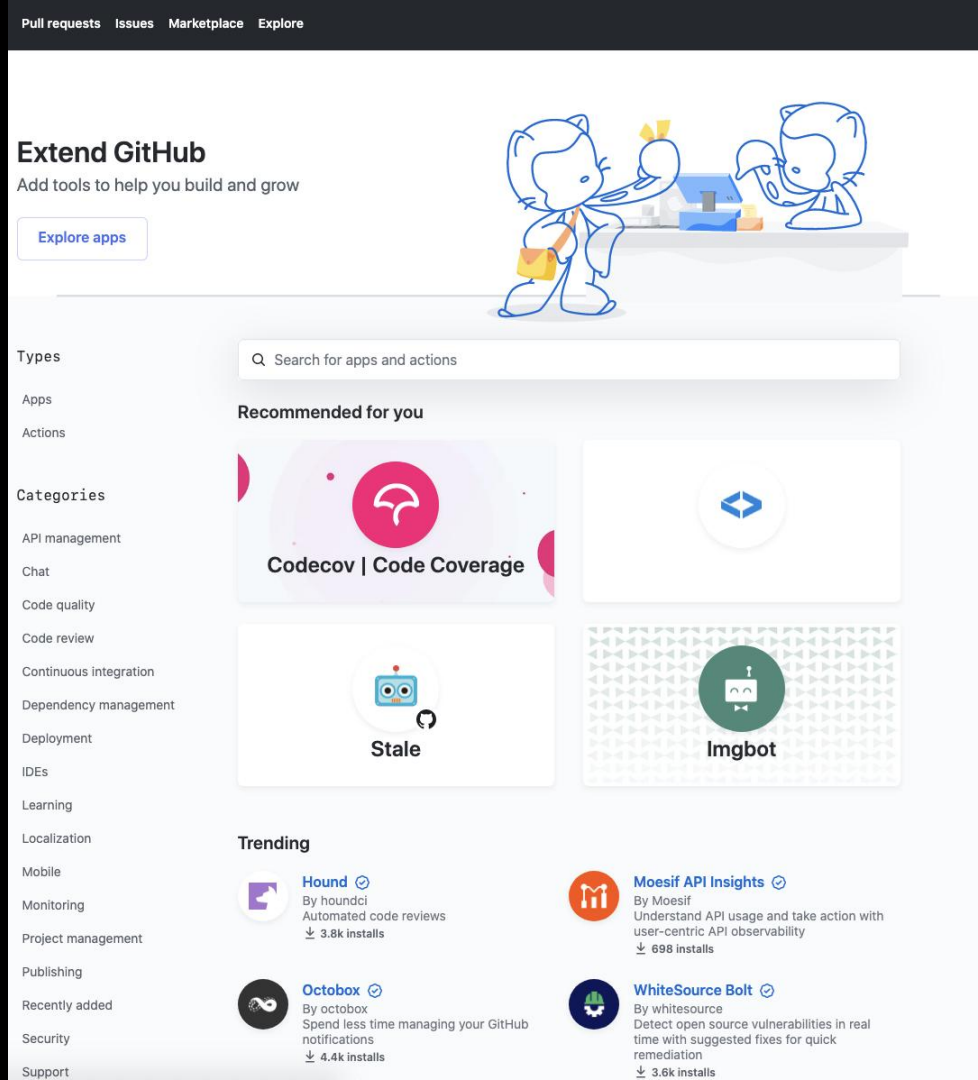


O que é YAML?

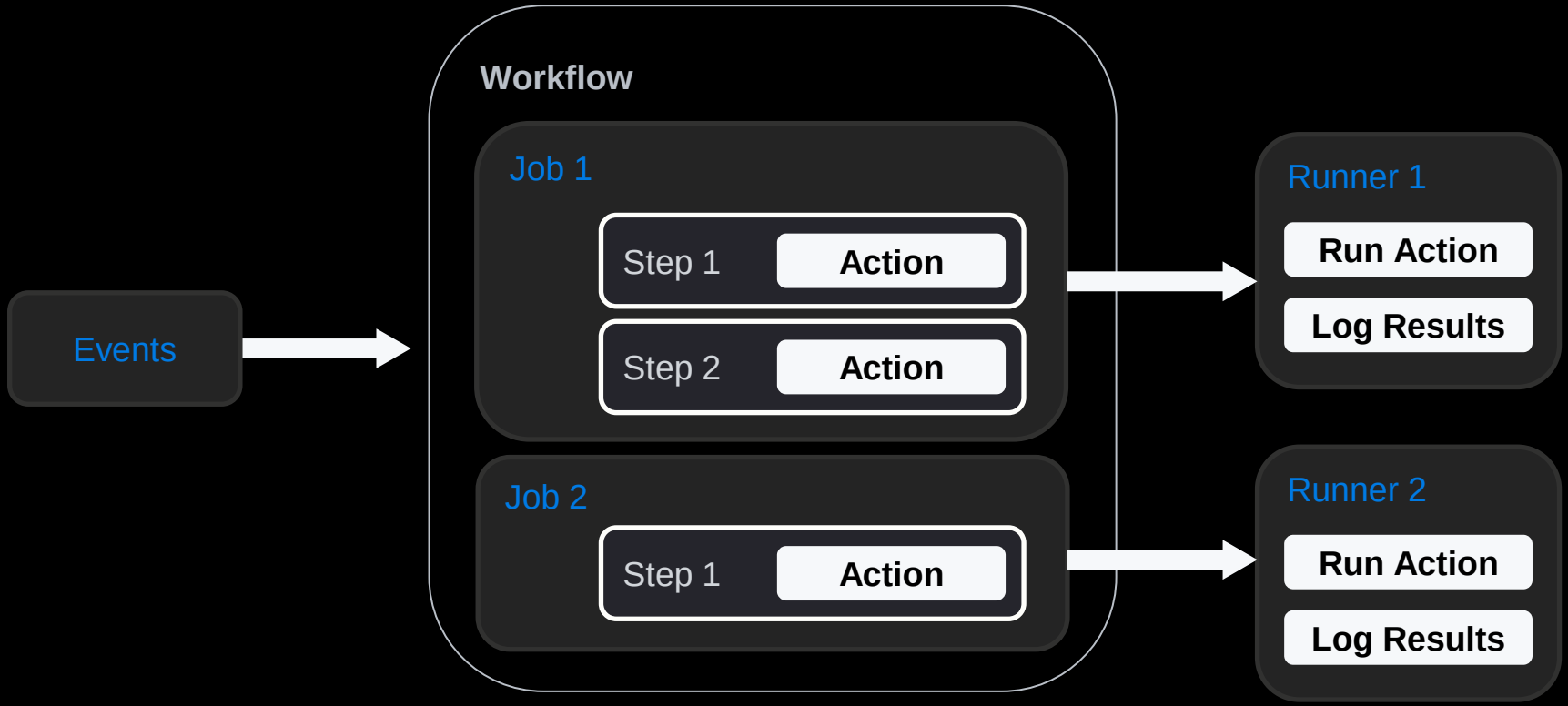
- Extensões: `.yml` ou `.yaml`
- Um superset JSON
- Contém quebras de linha e indentação sintaticamente relevantes em vez de chaves
- Linguagem de serialização de dados, legível e escrita por humanos

GitHub Marketplace

- Encontre várias Actions open-source para múltiplos domínios
- Mais de 18,000 Actions (e contando...)
- Criadores verificados (Domínio do editor e e-mail verificados)
- Utilize essas Actions diretamente do fluxo de Workflow
- Integrado ao editor do GitHub



Principais componentes



Fundamentos de workflows

- Um arquivo de texto em seu repositório (.github/workflows)
- YAML Ain't Markup Language (YAML)
- Eventos acionam workflows (on:)
- Um ou vários jobs
- Executado em um *runner (agente)*
- Contém um ou ou mais *steps (passos)*
- Um *step* reutilizável é um *action (ação)*

AccelerateDevOps / .github / workflows / pr-validation.yml in main

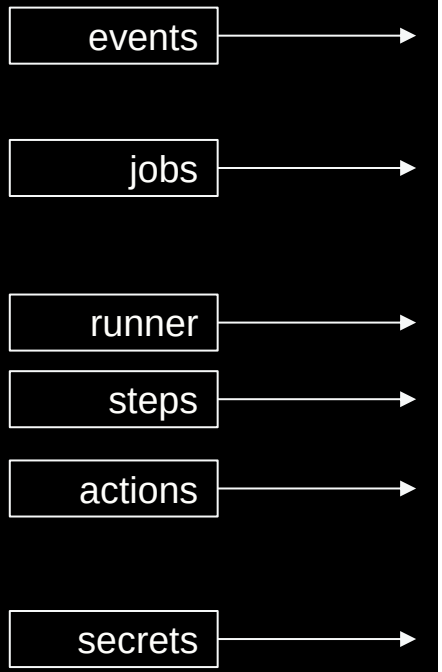
Edit file | Preview changes | Spaces | 2 | No wrap

```yaml
name: PR-Validation

on: [pull_request]

jobs:
  Build:
    runs-on: ubuntu-latest

    steps:
    - name: 'Checkout Github Action'
      uses: actions/checkout@master

    - name: Set up .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: '5.0.x'

    - name: Setup Node
      uses: actions/setup-node@v2.5.1
      with:
        node-version: 10.16.3

    - name: Install dependencies in client app
      working-directory: src/Tailwind.Traders.Web/ClientApp
      run: npm install

    - name: Build and publish with dotnet
      working-directory: src/Tailwind.Traders.Web
      run: |
        dotnet build --configuration Release

```

Use Control + Space to trigger autocomplete in most situations.

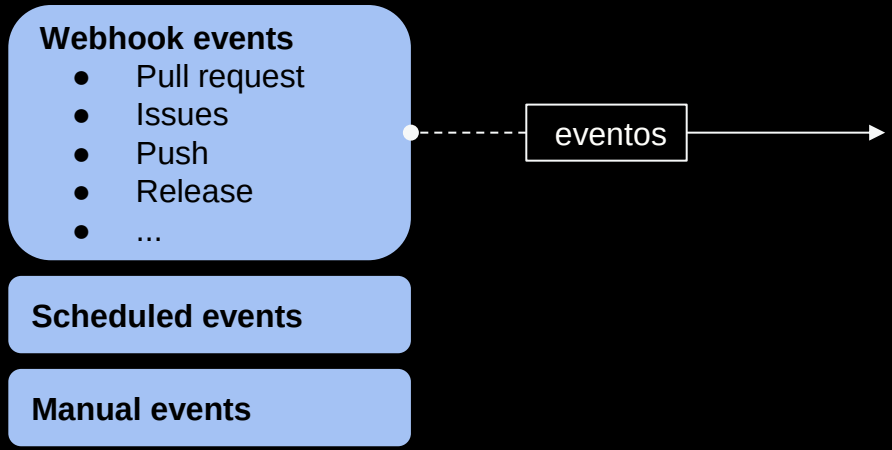
Sintaxe Básica

./.github/workflows/workflow-file-name.yml



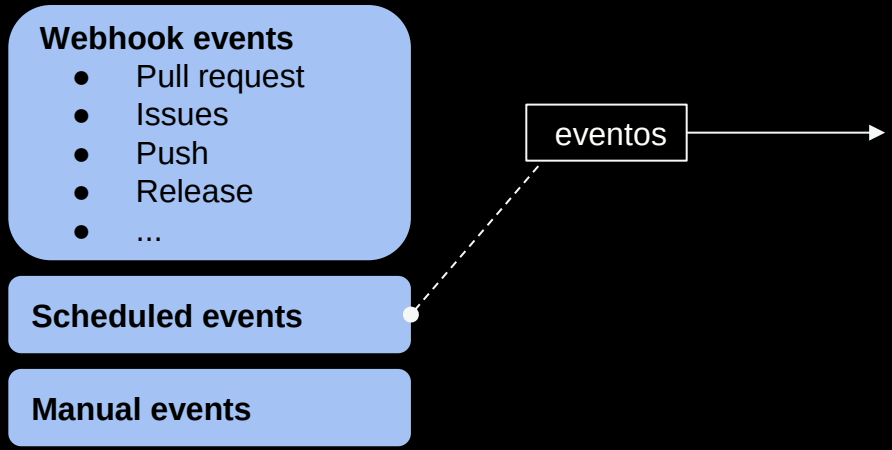

```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Eventos




```yaml
name: Super Linter workflow

on:
  issues:
    types: [closed, reopened]

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Eventos




```yaml
name: Super Linter workflow

on:
  schedule:
    - cron: '30 6 * * 5' # every Friday 06:30 UTC

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

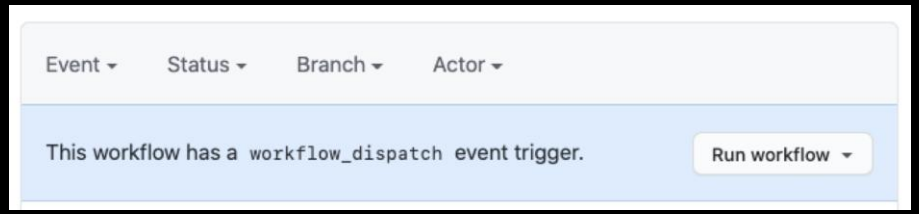
Eventos

Webhook events
- Pull request
- Issues
- Push
- Release
- ...

Scheduled events

Manual events
- workflow_dispatch
- repository_dispatch

eventos



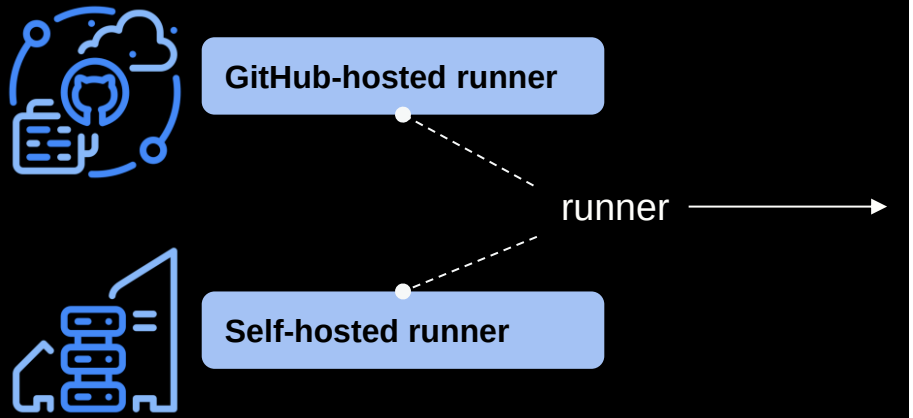

```yaml
name: Super Linter workflow

on:
  workflow_dispatch:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Workflow jobs

- Map – executado em paralelo por padrão
- Pode ser encadeado usando a palavra-chave "needs"
- É executado em um processo por um runner
- Contém uma sequência de etapas
- As etapas podem ser um comando de shell (run) ou uma ação (uses)

```yaml
jobs:
  job_1:
    runs-on: ubuntu-latest

    steps:
      - run: echo "🎉 The job was triggered by a ${{ github.event_name }} event."
      - run: echo "🏠 drive is `${{ github.event.inputs.homedrive }}`."
      - run: echo "🎯 environment is `${{ github.event.inputs.environment }}`."
      - run: echo "📖 log level is `${{ github.event.inputs.logLevel }}`."
      - run: echo "👓 Run the matrix? `${{ github.event.inputs.run_matrix }}`."

  job_2:
    runs-on: ubuntu-latest
    needs: job_1
    steps:
      - run: echo "Status ${{ job.status }}"

  job_3:
    runs-on: ubuntu-latest
    needs: job_1
    steps:
      - run: echo "Services ${{ job.services }}"

  job_4:
    runs-on: ubuntu-latest
    needs: [job_2, job_3]
    steps:
      - run: echo "Status ${{ job.status }}"
```

Runners




```yaml
name: Super Linter workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Runners

GitHub-hosted runner

- SO: ubuntu, windows, ou macOS
- Efêmeros
- 2-core CPU (macOS: 3-core)
- 7 GB RAM (macOS: 14 GB)
- 14 GB SSD
- Softwares Instalados: wget, GH CLI, AWS CLI, Java, …
- Atualmente não disponível no GHES (GitHub Enterprise Server)

Self-hosted runner

- Configuração de hardware personalizada
- Executar em um sistema operacional não suportado pelo runner hospedado no GitHub
- Referenciar o runner usando etiquetas personalizadas
- Podem ser agrupados
- Controlar quais organizações/repositórios têm acesso a quais runners/grupos de runners
- Não use com repositórios públicos!

Actions

Códigos reutilizáveis que podem ser referenciadas em workflow
O GitHub executa no ambiente de runtime Node.js ou em contêineres Docker
Você pode referenciar uma Ação ou executar scripts diretamente
Podem ser referenciadas de três maneiras:

- Repositório público
- O mesmo repositório do seu fluxo de trabalho (ações locais)
- Uma imagem de contêiner Docker publicada no DockerHub

script →

public actions →

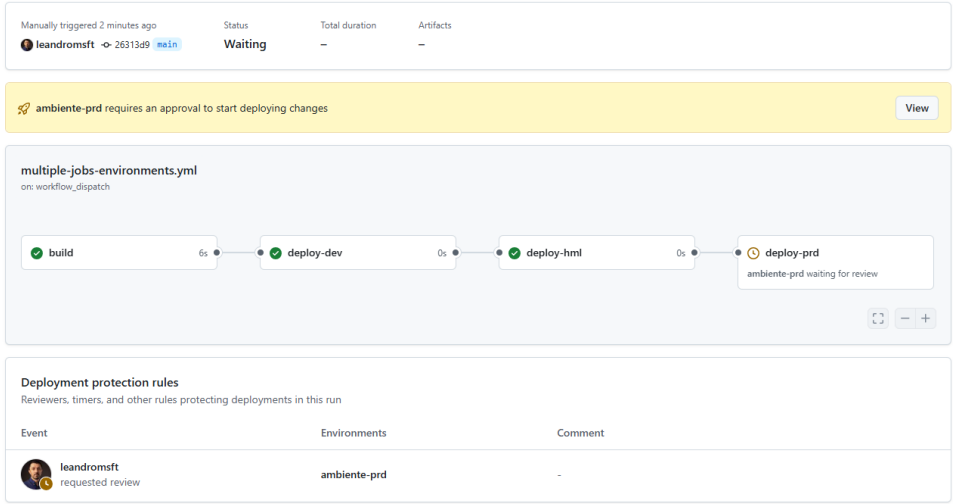
local action →

docker image →

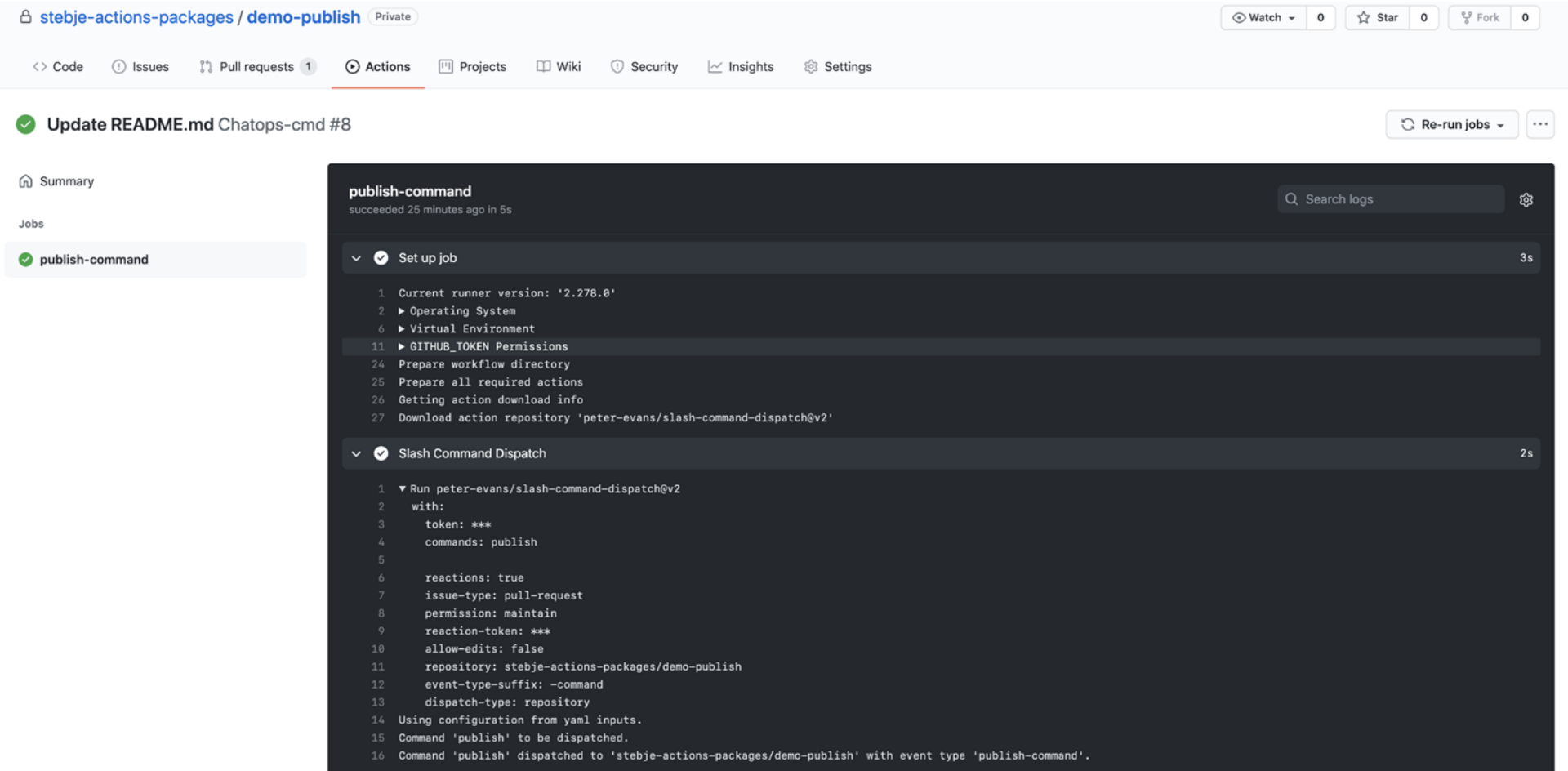
```yaml
name: Super workflow

on:
  push:

jobs:
  lint:
    name: Lint Code Base

    runs-on: ubuntu-latest

    steps:

      - run: echo "Hello World"

      - uses: actions/checkout@v2

      - uses: github/super-linter@v3
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

      - uses: ./path/to/action

      - uses: docker://alpine:3.8
```

Environments

- Os Environments são usados para descrever um destino de implantação geral, como DEV, QA, PROD.
- Você pode configurar ambientes com regras de proteção e segredos.



Workflow logs



Demo

Hora de praticar!

Atividade Módulo 4

Thank you