Vamos avaliar agora como efetuar a execução de nossos testes de API REST automatizada através do Newman, utilizando a pipeline do Github Actions. Os repositórios (incluindo o projeto da API) são disponibilizados no **GitHub**, basta clicar [neste link](https://github.com/carloseduardo1984/api_with_github_actions) para ter acesso ao clone.

**Mas o que são GitHub Actions?**

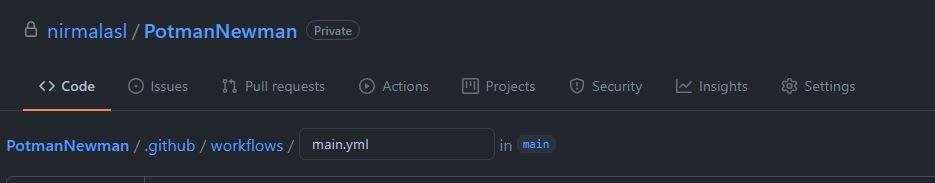
As GitHub Actions (ações do GitHub) são fluxos de trabalho, mais conhecidos na área de projetos de software como pipelines, que permitem o uso de um conjunto de ações através da identificação de determinados eventos  no repositório. Um evento no repositório pode ser simples, por exemplo "quando um push é feito para o branch master" ou "Uma solicitação de pull é levantada no branch master". O GitHub Actions nos permite automatizar certas respostas a esses eventos usando fluxos de trabalho (workflows) chamados de GitHub Actions. Essas ações reagem a um ou vários eventos e executam determinadas tarefas, por exemplo. "Construir e executar os testes", "Implantar o código", "Mesclar o código de uma ramificação para outra".

Podemos construir nossa integração contínua (CI) e implantação (CD) de ponta a ponta para o código no repositório diretamente no GitHub.

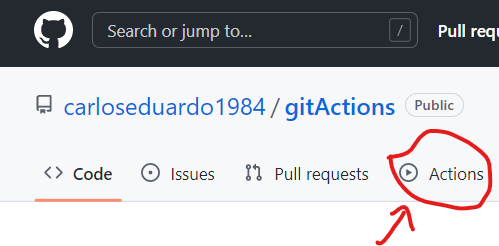
**Configurando o  GitHub Actions**

Adicionando uma AÇÃO

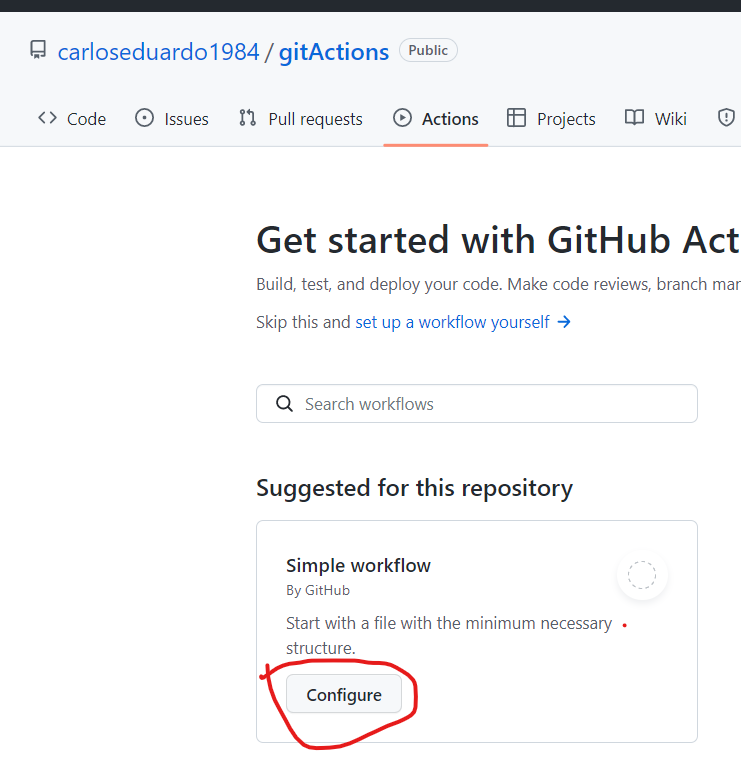
As ações do Github Actions são definidas em um arquivo no formato yaml (YAML é uma linguagem de serialização de dados legível por humanos. É comumente usada para arquivos de configuração e em aplicativos onde os dados estão sendo armazenados ou transmitidos) e estão localizadas na pasta do GitHub > **.github/workflows**



Para começar, crie o repositório com nossos arquivos exportados de Coleção e Ambiente, e logo após clique em Actions:

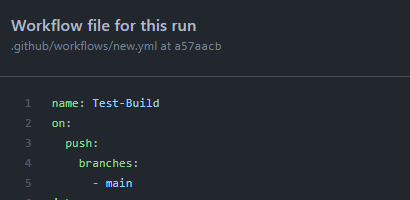


Na sequência, clique em fluxo simples, na opção configurar:

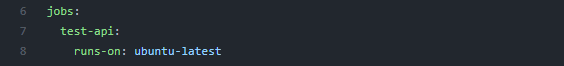


**Criando o arquivo de Workflow (Fluxo de trabalho)**

Começamos adicionando o gatilho (trigger) pelo qual nosso fluxo de trabalho GitHub Action será executado. Isso é feito no arquivo yaml, conforme mostrado abaixo. As linhas de código abaixo definem o nome do fluxo de trabalho e o que será executado quando ocorrer um evento de push no branch main/principal do repositório.



Agora criamos o trabalho que realmente executará a coleção POSTMAN.

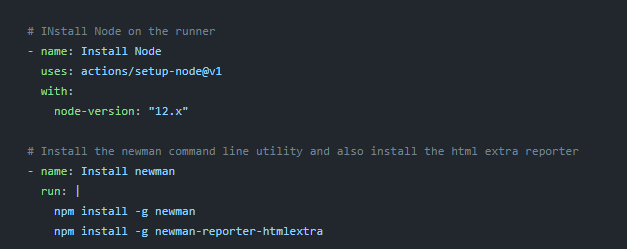


Agora, começamos a definir as etapas individuais que serão executadas dentro do trabalho.

**Instalando o Node e o Newman no arquivo de fluxo de trabalho — Workflow GitHub Action**

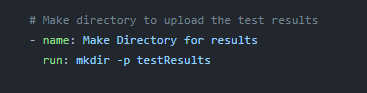
O Postman pode ser executado dentro dos agentes CI/CD, usando o utilitário de prompt de comando chamado newman. Para instalar o newman, primeiro precisamos instalar o node.js no runner. É feito como mostrado abaixo.

Observação: o newman-reporter-htmlextra instala um repórter personalizado que gerará relatórios visualmente atraentes para nós.



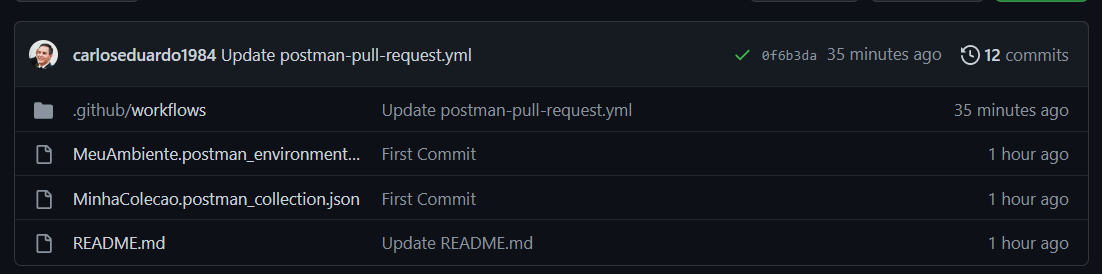
**Diretório para resultados de teste**

Como um processo personalizado em CI/CD, precisamos de um espaço de trabalho para fazer upload dos artefatos gerados durante a execução. Esses artefatos podem ser os logs gerados pelo agente, logs personalizados escritos em etapas, resultados de testes etc. GitHub Actions fornece uma maneira de fazer upload dos arquivos do trabalho para o espaço de trabalho. O truque é criar um diretório e enviar os resultados para ele. O diretório pode ser criado usando a etapa a seguir.

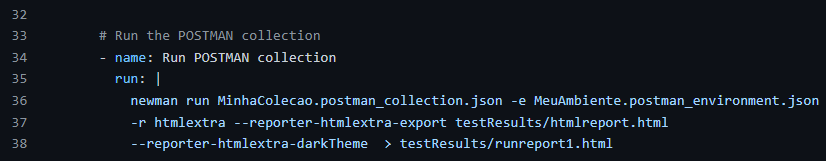


**Configurando a execução da coleção Postman no Workflow do GitHub Actions**

Nossa próxima tarefa é executar os testes definidos na coleção do Postman. O arquivo de coleção do Postman (.json) está disponível no repositório conforme mostrado abaixo.

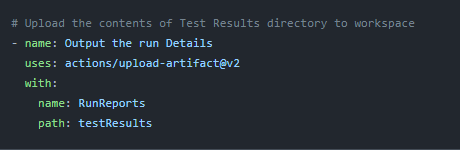


A etapa mostrada abaixo executa a coleção do Postman e cria o relatório html como resultado da execução do teste.



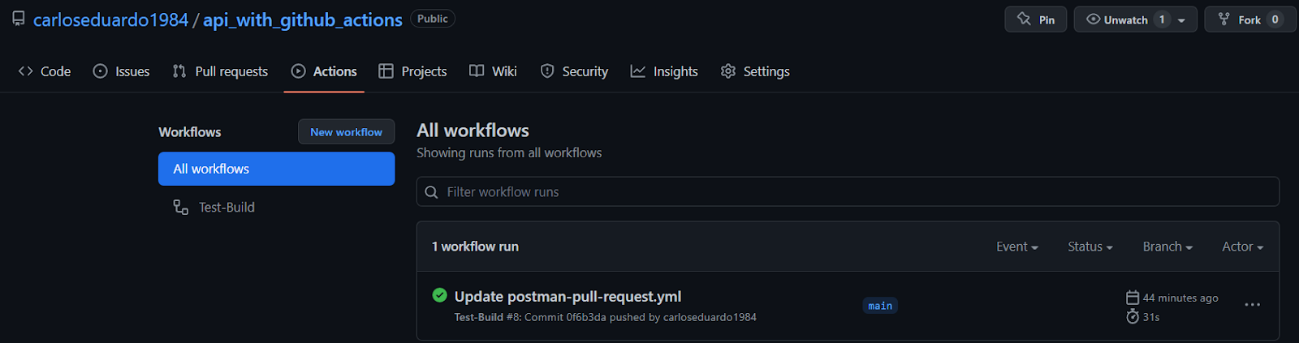
**Fazendo upload de artefatos (relatórios, coleções e ambientes)**

Como última etapa, fazemos upload dos artefatos para o espaço de trabalho. Isso pode ser feito usando o actions/upload-artifact@v2, conforme mostrado abaixo. Estamos carregando todo o conteúdo do diretório testResults que criamos anteriormente.

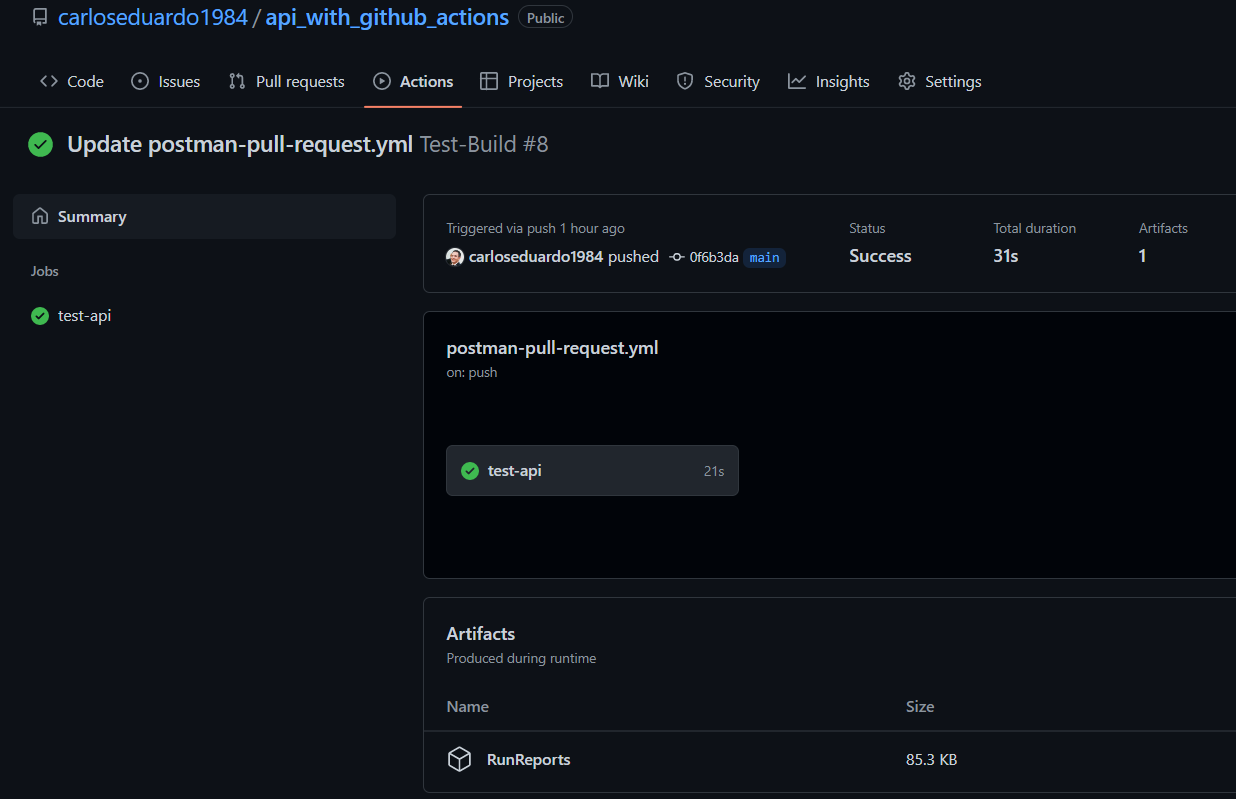


**O teste em si**

Quando um push é feito para o repositório principal, o fluxo de trabalho é acionado. A saída de cada compilação está disponível no painel Ações do repositório e podemos detalhar cada fluxo para ver por que ele falhou. Para executar novamente, basta fazer uma atualização no github, por exemplo um comentário no arquivo “README.md” e efetuar um commit. Assim, a execução do Github Action recebe o evento e inicia a nova execução do Workflow configurado.

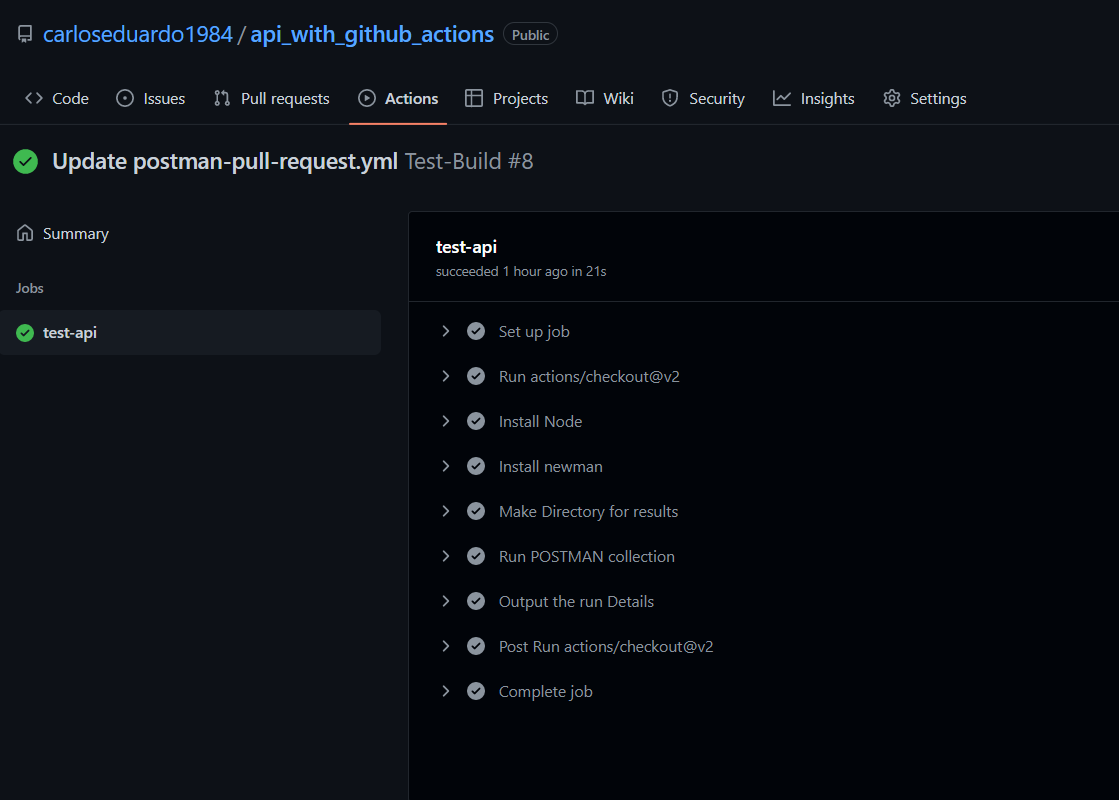


Ao clicar na execução (Update postman-pull-request.yml) vemos os detalhes da mesma:

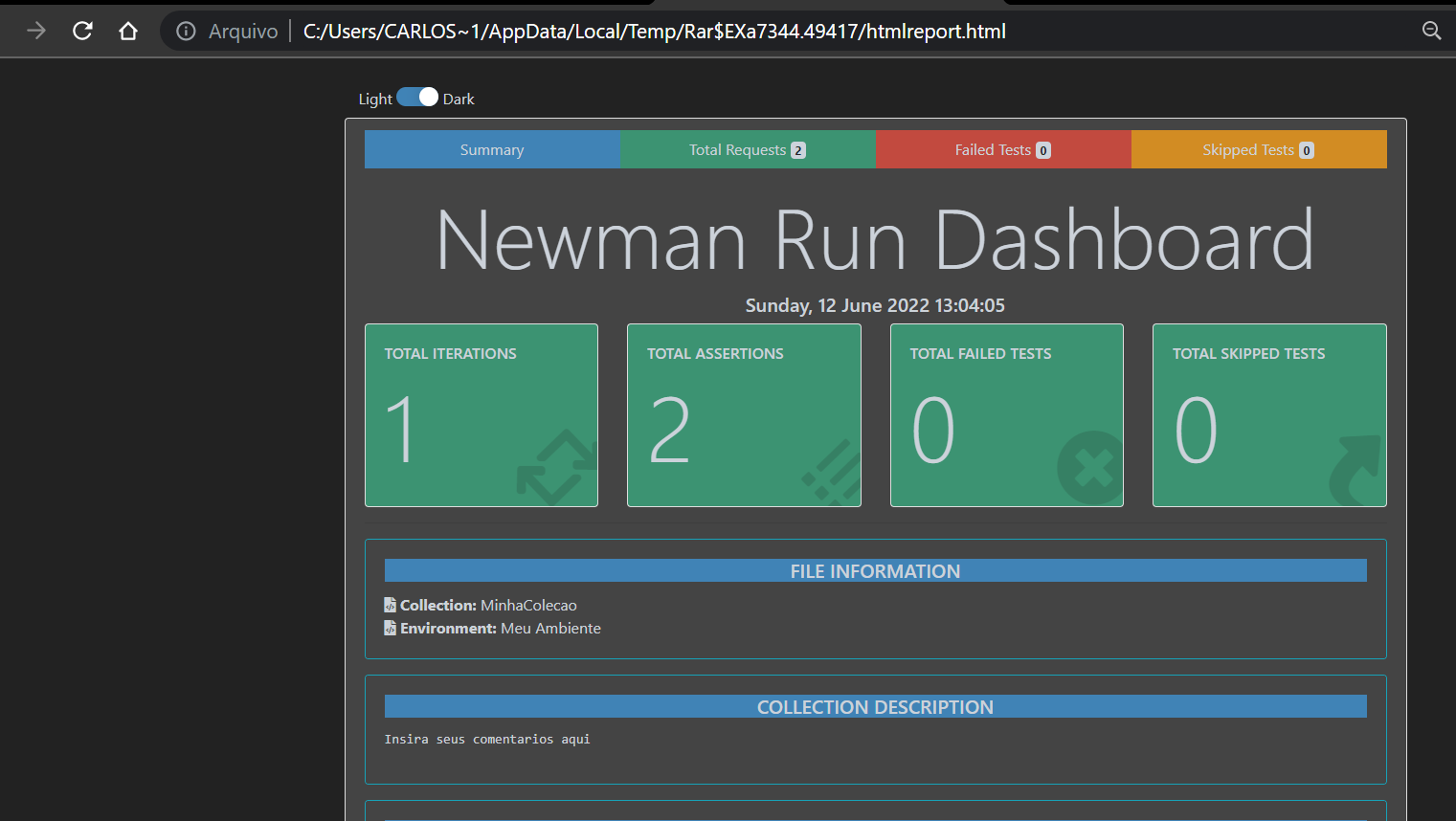


Acima temos a execução do Newman e abaixo o relatório em HTML no formato zip.

**Execução do Newman**



**Relatório**



AQUI O ARQUIVO YML NA ÍNTEGRA

name: Test-Build

on:

push:

branches:

- main

jobs:

test-api:

runs-on: ubuntu-latest

steps:

# Checks-out your repository under $GITHUB\_WORKSPACE, so your job can access it

- uses: actions/checkout@v2

# INstall Node on the runner

- name: Install Node

uses: actions/setup-node@v1

with:

node-version: "12.x"

# Install the newman command line utility and also install the html extra reporter

- name: Install newman

run: |

npm install -g newman

npm install -g newman-reporter-htmlextra

# Make directory to upload the test results

- name: Make Directory for results

run: mkdir -p testResults

# Run the POSTMAN collection

- name: Run POSTMAN collection

run: |

newman run MinhaColecao.postman\_collection.json -e MeuAmbiente.postman\_environment.json -r htmlextra --reporter-htmlextra-export testResults/htmlreport.html --reporter-htmlextra-darkTheme > testResults/runreport1.html

# Upload the contents of Test Results directory to workspace

- name: Output the run Details

uses: actions/upload-artifact@v2

with:

name: RunReports

path: testResults