

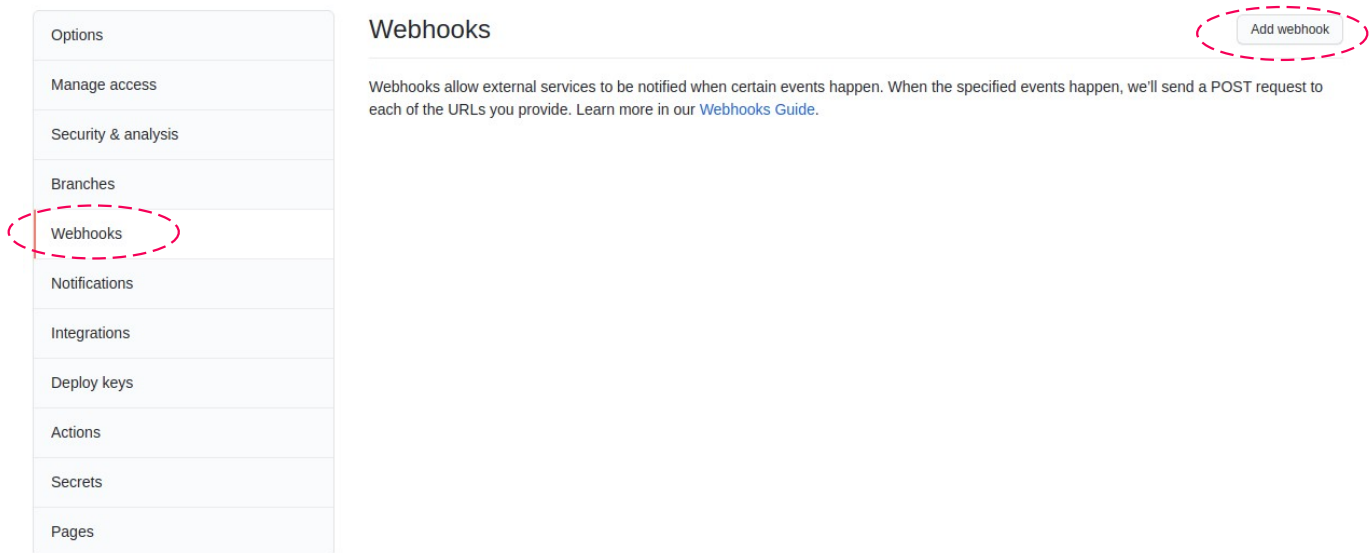
Tutorial

Servidor de Testes – Atividades e Labs

Siga as etapas deste tutorial para que o seu repositório de atividades da disciplina possa ser testado automaticamente. Após o cadastro, sempre que soltar uma **release** ou **tag** de versão, o teste será feito automaticamente e em questão de segundos você poderá saber se falhou ou não.

1. Cadastro do webhook

Acesse o repositório das atividades no github e acesse as **configurações** ou **settings** (ex: <https://github.com/insper-classroom/23-1-sishard-pedrods/settings>). No menu esquerdo, escolha a opção **Webhooks** e em seguida a opção **Add webhook**.



Será necessário preencher:

- **Payload URL:** `http://3.142.157.80/webhook/sishard/test/23-2-bcc`
- **Content type:** `application/json`
- **Secret:** deixe vazio!

- Which events would you like to trigger this webhook?: Escolha “Let me select individual events” e na sequência marque APENAS AS OPÇÕES:
 - Branch or tag creation
- Ao final, deixe a opção Active ativada.

General

Access

Collaborators and teams

Code and automation

Branches

Tags

Actions

Webhooks

Pages

Security

Code security and analysis

Deploy keys

Secrets and variables

Integrations

GitHub apps

Email notifications

Webhooks / Manage webhook

Settings Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://3.142.157.80/webhook/sishard/test/23-1

Content type

application/json

Secret

Which events would you like to trigger this webhook?

☐ Just the push event.
 ☐ Send me **everything**.
 ☒ Let me select individual events.

☒ **Branch or tag creation**

Branch or tag created.

☐ **Branch or tag deletion**

Branch or tag deleted.

☐ **Branch protection rules**

Branch protection rule created, deleted or edited.

☐ **Check runs**

Check run is created, requested, rerequested, or completed.

☐ **Check suites**

Check suite is requested, rerequested, or completed.

☐ **Code scanning alerts**

Code Scanning alert created, fixed in branch, or closed.

☐ **Repositories**

Repository created, deleted, archived, unarchived, publicized, privatized, edited, renamed, or transferred.

☐ **Repository imports**

Repository import succeeded, failed, or cancelled.

☐ **Repository vulnerability alerts**

Dependabot alert (aka dependency vulnerability alert) created, resolved, or dismissed on a repository.

☐ **Secret scanning alerts**

Secrets scanning alert created, resolved, or reopened

☐ **Stars**

A star is created or deleted from a repository.

☐ **Statuses**

Commit status updated from the API.

☐ **Team adds**

Team added or modified on a repository.

☐ **Visibility changes**

Repository changes from private to public.

☐ **Watches**

User stars a repository.

☐ **Wiki**

Wiki page updated.

☐ **Workflow jobs**

Workflow job queued, requested or completed on a repository.

☐ **Workflow runs**

Workflow run requested or completed on a repository.

☒ **Active**

We will deliver event details when this hook is triggered.

Add webhook

Pronto! Com isto o seu repositório já poderá ser testado automaticamente! Siga para as etapas 2, 3 e 4.

2. Gerando uma primeira release

Com o repositório clonado em sua máquina, abra o terminal e lance uma tag qualquer. Vamos lançar (propositalmente) uma tag para uma atividade inexistente.

Agra o terminal na raiz do repositório e digite os seguintes comandos:

```
git tag -a atv0.0.0 -m "atv0.0.0"
git push origin atv0.0.0
```

Acesse no github a aba de issues do seu repositório, você deve encontrar um retorno do teste, informando que a atividade não existe!

The screenshot shows the GitHub interface for a repository. The top navigation bar includes 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. Below the navigation bar, there's a search bar with the filter 'is:issue is:open'. The main content area shows a list of issues with one open issue titled 'Problemas na atv0.0.0' by user 'macielcalebe'. The issue details page is also visible, showing the title 'Problemas na atv0.0.0 #1', the status 'Open', and a comment from 'macielcalebe' stating 'Atividade atv0 não encontrada na lista de exercícios!'. The comment also includes a note to check the tag name and read the README.

Como não temos atv0 disponível, o teste não pode ser realizado!

Para conferir as atividades disponíveis, podemos abrir uma aba no navegador e acessar a URL: http://3.142.157.80/webhook/sishard/test/svg/23-2-bcc/insper-classroom/<repository_name>, trocando <repository_name> pelo seu repositório. Por exemplo:

<http://3.142.157.80/webhook/sishard/test/svg/23-2-bcc/insper-classroom/23-2-bcc-sishard-beatrizv>

3. Imagem SVG dos resultados

Agora iremos alterar o README.md para trazer o status atual dos testes do seu repositório. Edite o seu README.md e adicione no início uma chamada à API, informando o seu usuário github e o seu repositório, respeitando maiúsculas e minúsculas:

<> Edit file Preview

```
1 # 23-1-sishard
2
3 Este é o repositório da Beatriz!
4
5 # Status dos testes
6 ![svg](http://3.142.157.80/webhook/sishard/test/svg/23-1/insper-classroom/23-1-sishard-beatrizv)
```

- E altere <repository_name> para o seu repositório! Ex:

`![svg](http://3.142.157.80/webhook/sishard/test/svg/23-2-bcc/insper-classroom/23-2-bcc-sishard-beatrizv)`

Pronto! Ao acessar a raiz do seu repositório você deve ver a imagem contendo o status de todas as versões vigentes!

Status dos Testes

atv1 On time Pass

Os possíveis valores para a coluna do meio são:

- **Delayed:** quando você falha em passar nos testes até o deadline de entrega da atividade.
- **On time:** quando ainda há prazo até o deadline da versão ou se você passou nos testes dentro do prazo.

Os possíveis valores para a terceira coluna são:

- **Error:** quando ocorre algum erro ao executar os testes em sua release.
- **Failed:** quando o resultado dos testes é diferente do esperado.
- **To do:** quando você ainda não soltou nenhuma release da versão.
- **Pass:** quando passou em todos os testes.

4. Faça uma release para testar!

Solte uma release da `atv1`, seguindo a nomenclatura oficial (ex: `atv1.0.5`) e verifique se tudo ocorre como deveria. Os resultados esperados são:

- Passa nos testes e a imagem que contém o status da versão muda para **Pass**.
- Falha nos testes e uma issue é criada automaticamente no repositório.

Caso falhe em algum teste, solte uma nova release ou tag corrigindo o problema e feche a issue, pois caso falhe novamente o programa de testes criará uma nova automaticamente.

Atenção: o programa de testes testa uma tag ou release uma única vez, não sendo possível testar outra com o mesmo nome no futuro, logo, siga a nomenclatura oficial e vá incrementando (`atv2.0.0`, `atv2.0.1`, `atv2.0.2` ...).

Qualquer problema, entre em contato com o professor pelo e-mail carloosedm@insper.edu.br.