



# THE DEVELOPER'S CONFERENCE

## Estratégias de Qualidade no seu Fluxo de Entrega Contínua

Mensurando a Integridade de suas APIs

# ALEX ALVES



THE  
DEVELOPER'S  
CONFERENCE



- **MG** <3 **SP**
- Engenheiro de Software @ **XP Inc.**
- Fundador @ **TI\_Itaúna**

✉ alexalves2501@hotmail.com

🔗 [linktr.ee/alexalvess](https://linktr.ee/alexalvess)

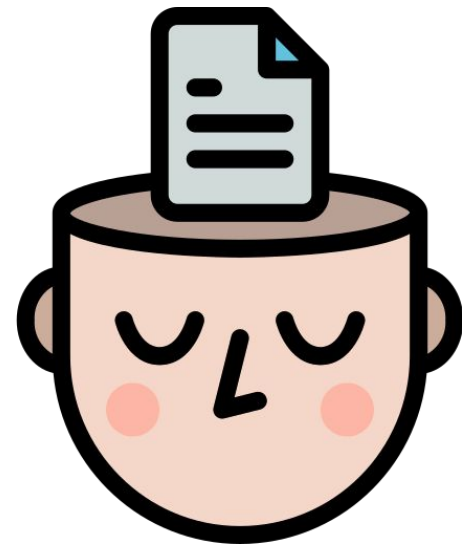


@alexalvess

# Contexto



- Iniciativa de construir testes integrados
- Como mensurá-los?



# O que é DevOps?



- Cultura
- Pessoas
- Ferramentas



# CI/CD o que são?



THE  
DEVELOPER'S  
CONFERENCE

- Integração Contínua
- Entrega Contínua
- Implantação Contínua



# Importância do CI/CD e DevOps



- Crescimento da empresa
- Entrega de valor
- Feedback imediato
- Resiliência



# Como garantir a qualidade?



# Por que Testes?



THE  
DEVELOPER'S  
CONFERENCE

- Segurança
- Integridade
- Confiabilidade
- O barato que **não** sai caro

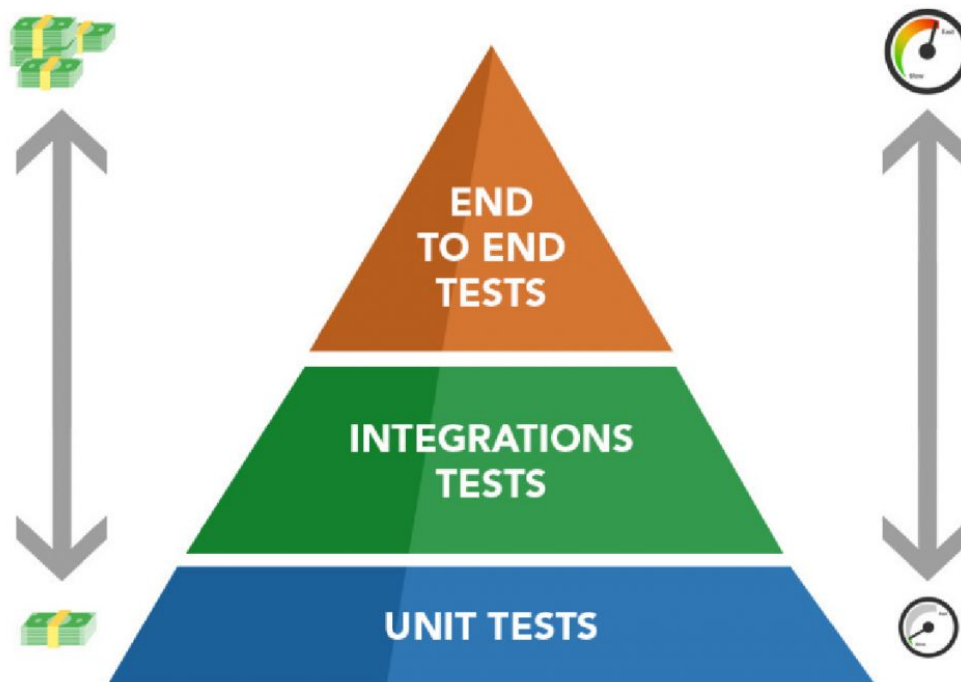




# Tipos de Testes



THE  
DEVELOPER'S  
CONFERENCE



# Onde que entra os Testes?



THE  
DEVELOPER'S  
CONFERENCE



# Como validar meus testes?



THE  
DEVELOPER'S  
CONFERENCE



# Testes Integrados



THE  
DEVELOPER'S  
CONFERENCE



# Case de Teste Integrado



THE  
DEVELOPER'S  
CONFERENCE



# Sem Métricas, Sem Avaliação?



THE  
DEVELOPER'S  
CONFERENCE



Que se faça a Luz



THE  
DEVELOPER'S  
CONFERENCE



# Programador é Preguiçoso



THE  
DEVELOPER'S  
CONFERENCE





# Identificando padrões



THE  
DEVELOPER'S  
CONFERENCE

```
→ Não seguro | aurora-project.azurewebsites.net/swagger/v1/swagger.json
{
  "openapi": "3.0.1",
  "info": {
    "title": "Aurora API",
    "description": "API REST created on ASP.NET Core 3.1",
    "version": "v1"
  },
  "paths": {
    "/api/users": {
      "post": {
        "tags": [
          "Worker"
        ],
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/CreateWorkerModel"
              }
            },
            "text/json": {
              "schema": {
                "$ref": "#/components/schemas/CreateWorkerModel"
              }
            },
            "application/*+json": {
              "schema": {
                "$ref": "#/components/schemas/CreateWorkerModel"
              }
            }
          }
        },
        "responses": {
          "200": {
            "description": "Success"
          }
        }
      }
    }
  }
}
```

```
UnitReport.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <testsuites name="Aurora Integration Tests" tests="2" time="33.779">
3   <testsuite name="GET /api/users" id=
4     "57d261a6-bd9a-43b2-a6b5-a92342c32871" timestamp=
      "2020-11-23T15:10:43.176Z" tests="1" failures="0" errors="0" time=
        "33.214">
5     <testcase name="Status code is OK" time="33.214" classname=
      "AuroraIntegrationTests"/>
6   </testsuite>
7   <testsuite name="DELETE /api/users/{id}" id=
      "b5efb0cc-e959-4c46-8060-782a13d731cf" timestamp=
        "2020-11-23T15:10:43.176Z" tests="1" failures="0" errors="0" time=
          "0.565">
8     <testcase name="Necessary to inform a valid ID to delete" time="0.565"
          classname="AuroraIntegrationTests"/>
9   </testsuite>
10 </testsuites>
```

# Let's Start!



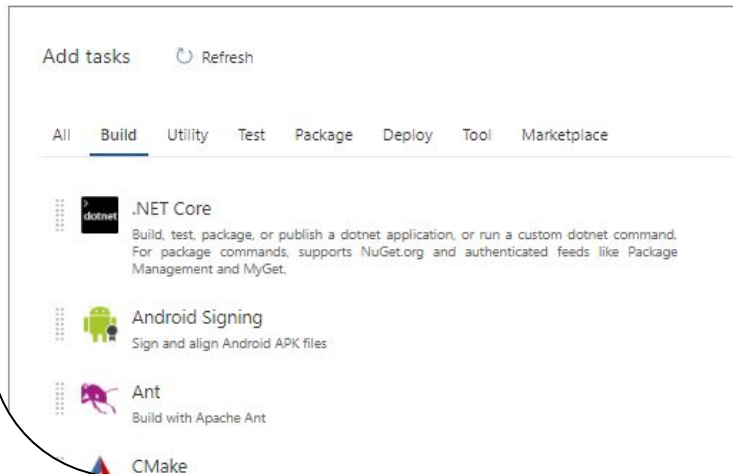
THE  
DEVELOPER'S  
CONFERENCE

## Add a custom pipelines task extension

03/04/2021 • 19 minutes to read • 

[Azure DevOps Services](#) | [Azure DevOps Server 2020](#) | [Azure DevOps Server 2019](#) | [TFS 2018](#) - [TFS 2017](#)

Learn how to install extensions to your organization for custom build or release tasks in Azure DevOps. These tasks appear next to Microsoft-provided tasks in the **Add Step** wizard.



## Install azure-pipelines-task-lib

We provide a library, *azure-pipelines-task-lib*, that should be used to create tasks. Add it to your library.

```
npm install azure-pipelines-task-lib --save
```

## Add typings for external dependencies

Ensure that TypeScript typings are installed for external dependencies.

```
npm install @types/node --save-dev
npm install @types/q --save-dev
```

Create a `.gitignore` file and add `node_modules` to it. Your build process should do an `npm install` and a `typings install` so that `node_modules` are built each time and don't need to be checked in.

```
echo node_modules > .gitignore
```



# Custom Task



THE  
DEVELOPER'S  
CONFERENCE



**Coverage API Test** |  Reports |  Manage

Alex Alves |  13 installs |      (0) | Free

Tools for generate the coverage by test result of a integration test.

Get it free

# The Code !



THE  
DEVELOPER'S  
CONFERENCE

```
71 async function processAnalysis(file: string, inputData: InputDataModel) {
72     log(`Reading file: ${file}`);
73     const testResultsFile = fs.readFileSync(file, "utf8");
74     const parser = new Parser(testResultsFile, {error: any, result: any} => {
75         if (error) {
76             throw error;
77         } else {
78             let endpointsTested: Array<EndpointModel> = new Array<EndpointModel>();
79             let endpointsExists: Array<EndpointModel> = new Array<EndpointModel>();
80             let coverage: CoverageModel = new CoverageModel();
81
82             if (inputData.testType === TestType.TestSuite) {
83                 testSuiteMap(result.testsuite, endpointsTested);
84             } else {
85                 testCaseMap(result.testsuite, endpointsTested);
86             }
87
88             coverage.tested = EndpointModel.totalEndpoints(
89                 "Endpoints tested",
90                 endpointsTested
91             );
92
93             log(`Reading Swagger of API: ${inputData.url}`);
94             makeGetRequest(inputData.url)
95                 .then((response: any) => {
96                     endpointsMap(response.data, endpointsExists);
97
98                     coverage.existed = EndpointModel.totalEndpoints(
99                         "Endpoints found",
100                         endpointsExists
101                     );
102
103                     coverage.uncover = findUncoverEndpoints(
```

The status bar at the bottom shows the file is 'index.ts' in the 'typescript' language, with 781 lines, 0 errors, 0 warnings, 0 suggestions, and 5 fixes. It also shows the current cursor position as 'Ln 71, Col 21' and various settings like 'Spaces: 4', 'UTF-8', 'CRLF', 'TypeScript', '4.2.3', 'ESLint', and 'Prettier'.

# API Coverage Test



THE  
DEVELOPER'S  
CONFERENCE

API Coverage Test ⓘ

[Link settings](#) [View YAML](#) [Remove](#)

Task version 0.\* ▾

Display name \*

API Coverage Test

Api url \* ⓘ

https://aurora-project.azurewebsites.net/

PATH to JSON of Swagger \* ⓘ

/swagger/v1/swagger.json

Tests Result Path \* ⓘ

\$(System.DefaultWorkingDirectory)

Where is the tests name \* ⓘ



Test Suite



Test Case

Minimum Quality Gate Percentage \* ⓘ

0

Webhook ^

Webhooks ⓘ

Application Name ⓘ

Test.Postman

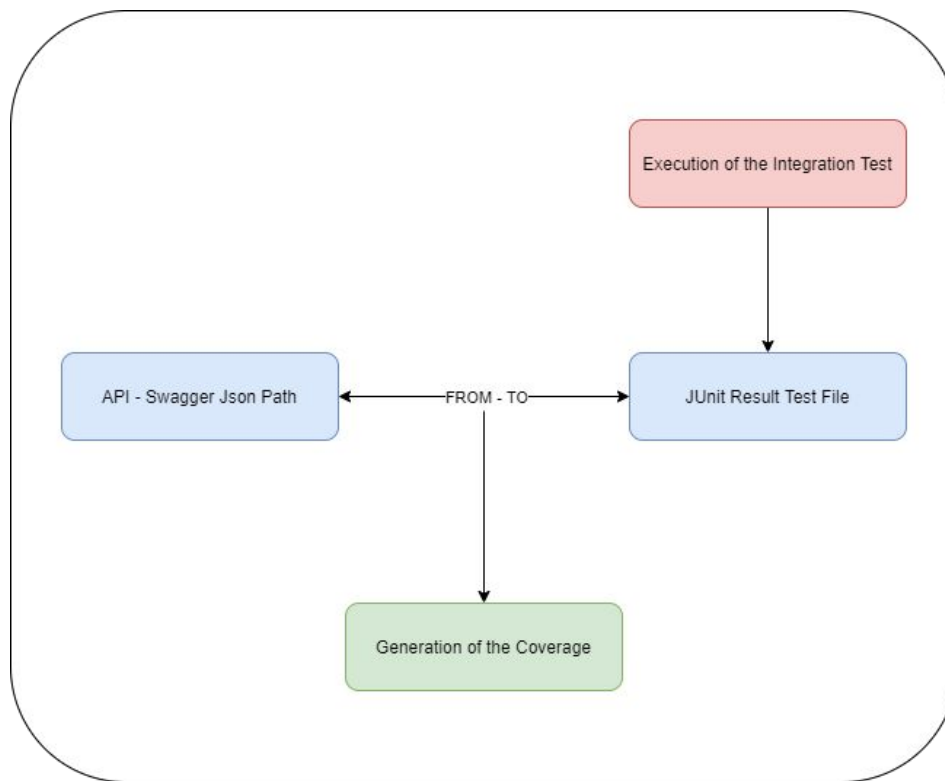
Build Number ⓘ

\$(Build.BuildNumber)

# A Arquitetura



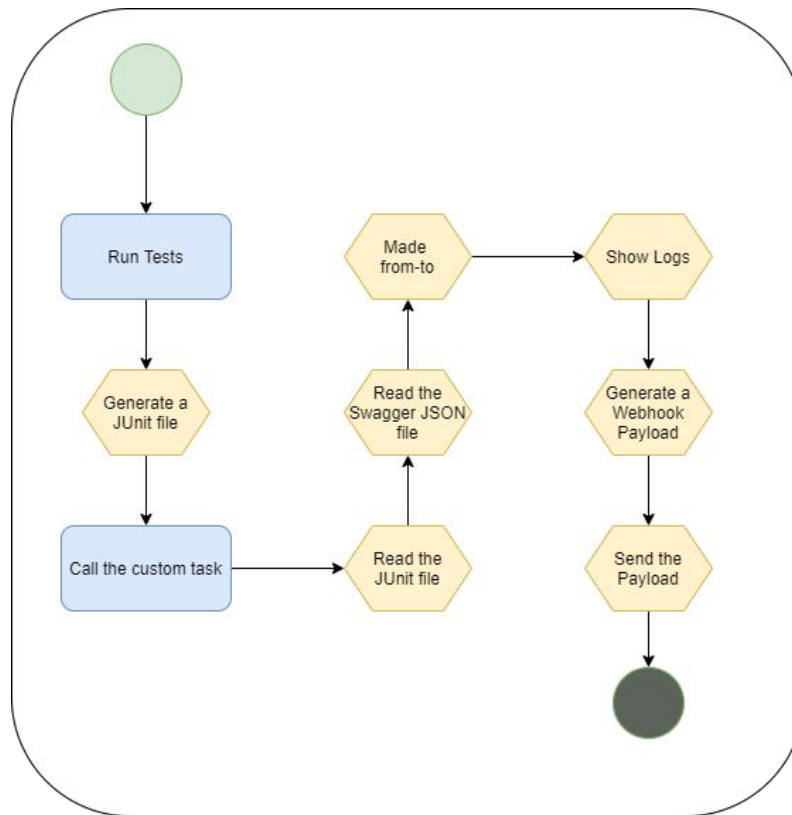
THE  
DEVELOPER'S  
CONFERENCE



# O Fluxo!



THE  
DEVELOPER'S  
CONFERENCE



# Que dados eu gero?



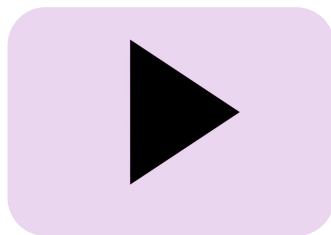
THE  
DEVELOPER'S  
CONFERENCE







THE  
DEVELOPER'S  
CONFERENCE

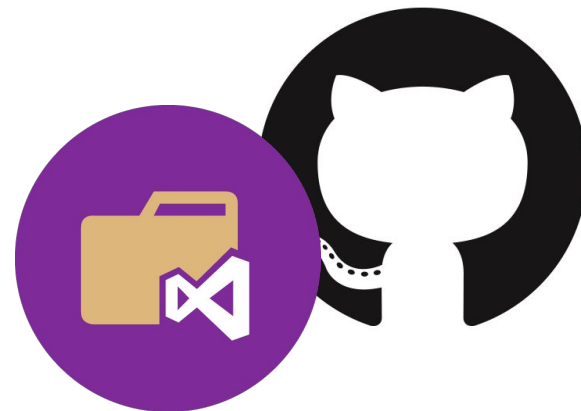


< Show me All! />



THE  
DEVELOPER'S  
CONFERENCE

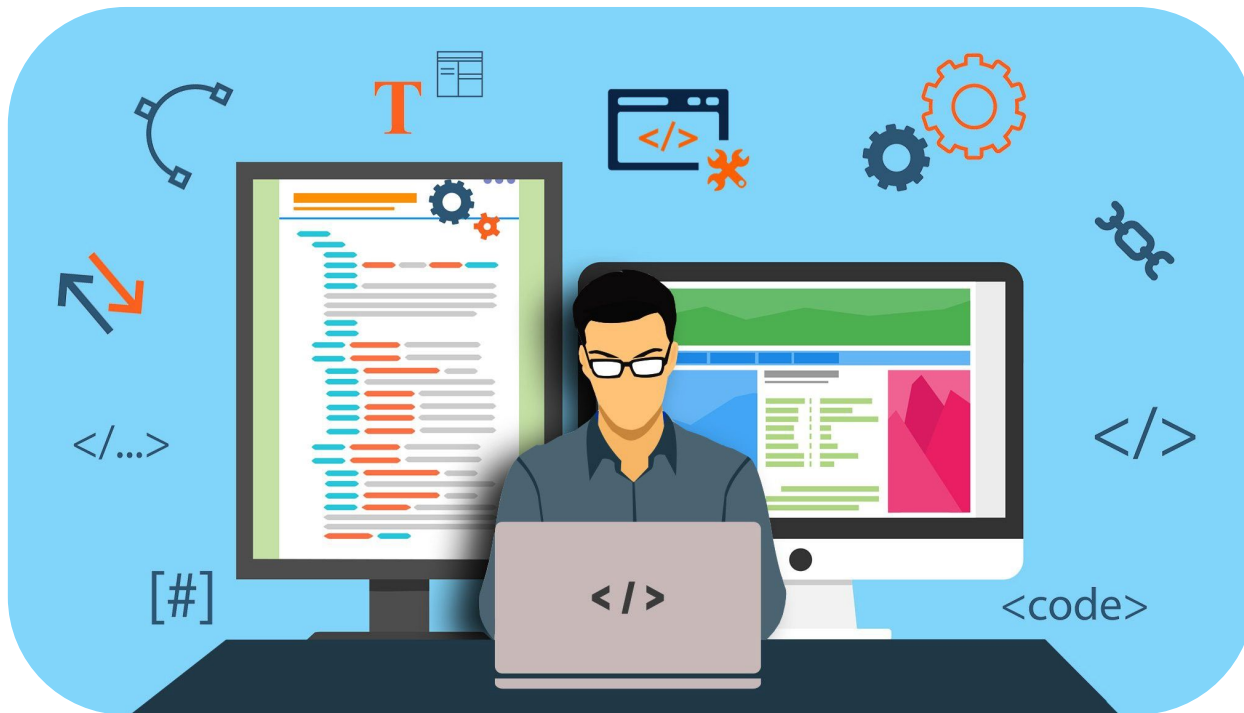
A Extensão se encontra no [Visual Studio Market Place](#)  
O Código se encontra no [GitHub](#)



# Conclusão



THE  
DEVELOPER'S  
CONFERENCE





THE  
DEVELOPER'S  
CONFERENCE

# Dúvidas?





# Obrigado!



**KEEP  
CALM  
and  
WRITE  
CODE**

# Referências



- [Add a custom pipelines task extension](#)
- [Criando Tasks Customizadas para Pipelines no Azure DevOps](#)
- [Testes: O que são? Aonde vivem?](#)

