Cypress com Pepiline Azure DevOps Integração contínua — CI

[Rosangela Fernandes da Costa](https://medium.com/@rosangelafernandesdacosta?source=post_page-----386741e832c7----------------------)

[Sep 1](https://medium.com/@rosangelafernandesdacosta/cypress-com-pepiline-azure-devops-integra%C3%A7%C3%A3o-cont%C3%ADnua-ci-386741e832c7?source=post_page-----386741e832c7----------------------) · 4 min read

Cypress é uma ferramenta para automação de testes de última geração para front-end e APIs REST.  
É open source escrita em javascript e utiliza o NODE.JS para instalação. Com essa ferramenta é possível escrever testes de forma rápida.

Inicialmente escrevi os testes e executava localmente na minha máquina, o que ficava sempre com a dependência para executar os testes. Como a ideia é que a execução seja de forma automática e que rode várias vezes, onde todos da equipe possam acompanhar e visualizar o report dos testes sempre que necessário, partimos para a integração continua usando o Pepiline Azure.

**Repositório**O código fonte do projeto precisa estar em um repositório na nuvem, no meu caso utilizei o repositório Azure Git.

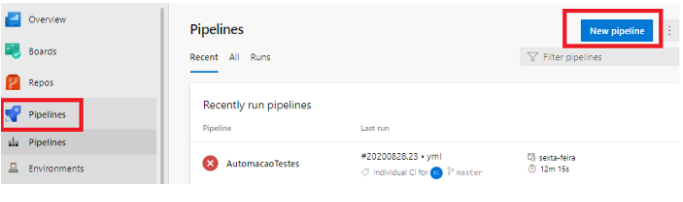


**Criar pipeline para execução dos testes Cypress no Azure DevOps**

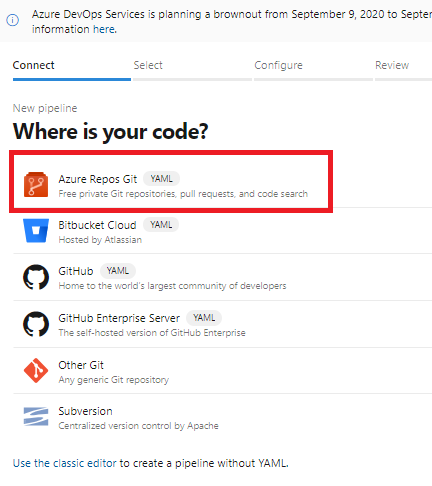
É necessário que tenha um projeto criado na Azure e possua acesso para criar um pipeline. Primeiro precisa criar um servidor NODE, conforme as etapas abaixo:

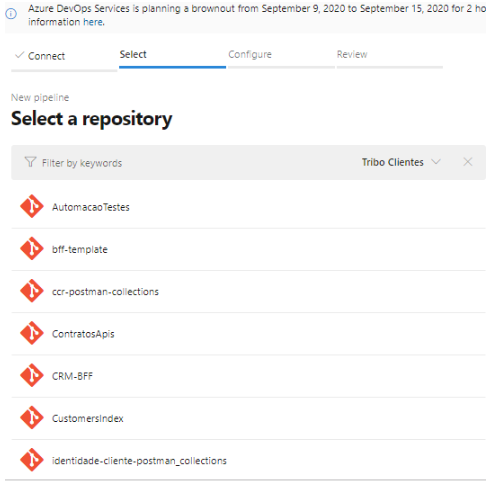
1. Entre na sua organização do Azure DevOps e navegue até seu projeto

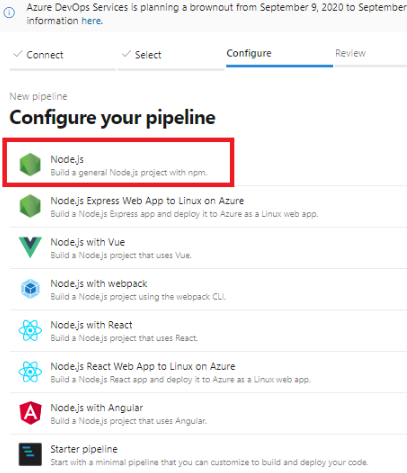
2. Em seu projeto, navegue até a página **Pipelines**. Em seguida, escolha a ação para criar um novo pipeline.



3. Siga as etapas do assistente selecionando a opção **Azure Repos Git**como o local do repositório do seu código-fonte. Pode escolher uma das opções abaixo, o importante que seja onde se encontra o repositório do projeto que será automatizado.

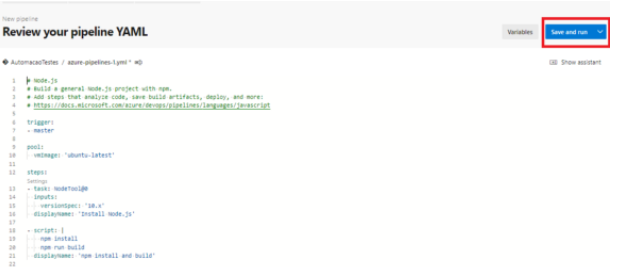


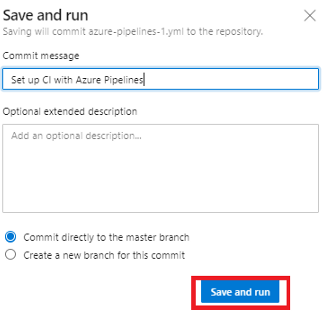
4. Quando a lista de repositórios aparecer, selecione o repositório onde encontra-se seu código fonte.  
  
  
  
  
5. O Azure Pipelines analisará o código em seu repositório e recomendará o modelo Node.js para seu pipeline. Selecione-o.



6. O Azure Pipelines gerará um arquivo YAML para o seu pipeline. Selecione **Salvar e executar e** , a seguir, **Confirmar diretamente para a ramificação mestre** e, a seguir, **Salvar e executar** novamente.  
Código do arquivo padrão criado

# Node.js  
# Build a general Node.js project with npm.  
# Add steps that analyze code, save build artifacts, deploy, and more:  
# <https://docs.microsoft.com/azure/devops/pipelines/languages/javascript>trigger:  
- masterpool:  
 vmImage: 'ubuntu-latest'steps:  
- task: NodeTool@0  
 inputs:  
 versionSpec: '10.x'  
 displayName: 'Install Node.js'- script: |  
 npm install  
 npm run build  
 displayName: 'npm install and build'





7. Configuração do arquivo .yml

O arquivo yml precisa ser configurado de acordo com o projeto do cypress que será executado na Azure. No exemplo que estamos publicando, o arquivo ficou com o seguinte código.

# Node.js  
# Build a general Node.js project with npm.  
# Add steps that analyze code, save build artifacts, deploy, and more:  
# <https://docs.microsoft.com/azure/devops/pipelines/languages/javascript>trigger:  
- masterpool:  
 vmImage: 'ubuntu-latest'steps:  
- task: NodeTool@0  
 inputs:  
 versionSpec: '10.x'  
 displayName: 'Install Node.js'- script: |  
 cd bff-ura-midway && npm ci  
 displayName: 'Install project dependencies'  
- script: |  
 cd bff-ura-midway && npm run cypress:verify  
 displayName: 'Verify if Cypress can be executed'  
- script: |  
 cd bff-ura-midway && npm run cypress:run  
 displayName: 'Executed test Cypress'  
- task: PublishTestResults@2  
 displayName: 'Publish the test reports'  
 inputs:  
 testResultsFiles: '\*\*/test-\*.xml'  
 condition: succeededOrFailed()  
 continueOnError: true

**Configuração do projeto Cypress report JUnit**

Precisamos fazer algumas configurações no projeto para exibir o resultado dos testes usando o reporte do JUnit.  
No arquivo **package.json**acrescentar em “scripts” as seguintes linhas, que serão usadas no arquivo .yml para execução dos testes da Azure.

"cypress:verify": "cypress verify",  
"cypress:run": "cypress run"

No arquivo “**cypress.json**” acrescente as seguintes linhas, que serão o report do JUnit e onde serão armazenados os resultado dos arquivos de testes.  
No nome dos arquivos xml, usamos o [hash], pois o cypress sempre gera o mesmo nome de arquivo para o resultado dos scripts executados.  
Ao adicionar a parte “hash”, ele irá gerar um valor hash aleatório e atribuí-lo a cada nome de arquivo xml, de forma que os arquivos não sejam substituídos.

"reporter": "junit",  
"reporterOptions": {  
 "mochaFile": "cypress/reports/junit/test-results-[hash].xml",  
 "toConsole" : true  
}

**Reporte dos testes executados no pipeline**  
Com essa configuração e o pipeline criado e configurado, sempre que for realizado um push no repositório git onde foi criado o pipelene os testes serão executados.  
O resultado é esse dashboard com os percentuais dos testes de passaram e falharam, além de mostrar todos os cenários executados e seus resultados.

