



## Infraestrutura II

# Criando nosso primeiro Pipeline

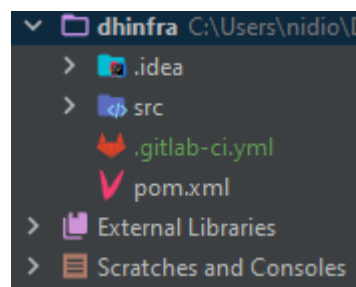
Tendo configurado nossa conta no GitLab e carregado nosso primeiro código, hoje vamos fazer um pipeline com 2 estágios.

### Objetivo final da prática

- Fazer um pipeline.
- Modifique parte do nosso código para ver como isso afeta o pipeline
- Adicione um estágio ao nosso pipeline

## Vamos trabalhar!

O local para definir um pipeline no GitLab é o arquivo `.gitlab-ci.yml`, que deve estar localizado na raiz do nosso projeto. Para isso, abrimos um arquivo com o nome `.gitlab-ci.yml` em **nosso computador** com o editor de texto de nossa escolha.





## 1. Criando os estágios

A definição do pipeline se dá em stage e jobs, estes pertencem aos estágios, ou seja, os primeiros estágios são os de nível hierárquico mais alto.

A execução das etapas é de acordo com como as definimos em nosso arquivo, por convenção para esta primeira prática, vamos construir duas etapas

- **build**
- **test**

Adicionamos as linhas em nosso arquivo `.gitlab-ci.yml`, lembre-se sempre de respeitar a indentação

```
stages:  
  - build  
  - test
```

Embora esta definição esteja correta, basicamente este pipeline não fará nada, pois os estágios não possuem jobs, que é onde os comandos são executados.

## 2. Adicionamos os jobs

Os jobs são definidos no arquivo e sua pseudo estrutura é

```
job_name  
  stage: "etapa ao qual pertence"  
  script:  
    - "comandos a serem executados no job"  
    - echo "hello pipeline"
```

Primeiro vamos definir um job para nosso estágio "build", que compilará o código Java que temos; para isso usaremos o Maven como compilador, o comando para compilar será "mvn compile".

O código do nosso `.gitlab-ci.yml` com o job adicionado deve ser:

```
image: maven:3.6.3  
stages:  
  - build  
  - test  
  
build_job:  
  stage: build  
  script:  
    - echo "Maven compile started"  
    - "mvn compile"
```

Até agora, nós definimos o pipeline, mas o GitLab não está ciente das alterações, é hora de fazê-lo e para isso devemos implementar essas mudanças que ainda estão em nosso computador para o repositório, é por isso que devemos enviar este novo arquivo, executando em **nosso computador**:

```
git add .
git commit -m "Meu primeiro pipeline"
git push
```

### 3. Pipeline em execução!

O GitLab, ao receber as alterações no código, detecta automaticamente a presença do arquivo `.gitlab-ci.yml` e se sua estrutura for válida, **EXECUTA** os passos definidos nele, por isso agora vamos ao nosso **GitLab CI/CD** Pipelines, veremos a lista de Pipelines executados e haverá o correspondente ao último commit.

Nidio Dolfini > infra2 > Pipelines

All 2 Finished Branches Tags Clear runner caches CI lint Run pipeline

Filter pipelines  Show Pipeline ID ▾

Status	Pipeline	Triggerer	Stages
<span>passed</span> 00:00:50 just now	Meu primeiro Pipeline #540 test 3d4dcc20 latest		<span>✓</span>

#### Podemos ver detalhes de sua execução?

Sim, digitamos o número (neste caso de exemplo “#540”) lá veremos os estágios que foram executados

#### Meu primeiro Pipeline

🕒 1 job for test in 50 seconds (queued for 2 seconds)

🚩 latest

🔗 3d4dcc20

🔗 No related merge requests found.

Aqui vemos que apenas o estágio BUILD foi executado, e seu trabalho “build\_job”, se entrarmos observamos todos os detalhes da execução. Clique no botão “build\_job”.

Pipeline Needs Jobs 1 Tests 0

Build

✓ build\_job



```
345 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compilers/2.8.4/plexus-compilers-2.8.4.gom (1.3 kB at 4.9 kB/s)
346 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.4/plexus-utils-2.0.4.jar
347 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.14/plexus-interpolation-1.14.jar
348 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-component-annotations/1.7.1/plexus-component-annotations-1.7.1.jar
349 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.2.1/maven-shared-utils-3.2.1.jar
350 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-incremental/1.1/maven-shared-incremental-1.1.jar
351 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-component-annotations/1.7.1/plexus-component-annotations-1.7.1.jar (4.3 kB at 15 kB/s)
352 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-java/0.9.10/plexus-java-0.9.10.jar
353 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-incremental/1.1/maven-shared-incremental-1.1.jar (14 kB at 47 kB/s)
354 Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/6.2/asm-6.2.jar
355 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.14/plexus-interpolation-1.14.jar (61 kB at 202 kB/s)
356 Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0-M9/qdox-2.0-M9.jar
357 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.4/plexus-utils-2.0.4.jar (222 kB at 712 kB/s)
358 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.8.4/plexus-compiler-api-2.8.4.jar
359 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.2.1/maven-shared-utils-3.2.1.jar (167 kB at 490 kB/s)
360 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.8.4/plexus-compiler-manager-2.8.4.jar
361 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-java/0.9.10/plexus-java-0.9.10.jar (39 kB at 60 kB/s)
362 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.8.4/plexus-compiler-javac-2.8.4.jar
363 Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/6.2/asm-6.2.jar (111 kB at 194 kB/s)
364 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.8.4/plexus-compiler-api-2.8.4.jar (27 kB at 46 kB/s)
365 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.8.4/plexus-compiler-manager-2.8.4.jar (4.7 kB at 7.5 kB/s)
366 Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0-M9/qdox-2.0-M9.jar (317 kB at 415 kB/s)
367 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.8.4/plexus-compiler-javac-2.8.4.jar (21 kB at 25 kB/s)
368 [INFO] Changes detected - recompiling the module!
369 [INFO] Compiling 1 source file to /builds/nidio/infra2/target/classes
370 [INFO] -----
371 [INFO] BUILD SUCCESS
372 [INFO] -----
373 [INFO] Total time: 43.136 s
374 [INFO] Finished at: 2022-04-05T19:23:52Z
375 [INFO] -----
376 $ echo "Hello World from Gitlab"
377 Hello World from Gitlab
378 Cleaning up project directory and file based variables
379 Job succeeded
```

#### 4. E em que momento o Pipeline é executado novamente?

Estando vinculado ao repositório de código, qualquer alteração no meu código-fonte (não apenas no .gitlab-ci.yml) fará com que o pipeline seja executado novamente.

**Vamos ao trabalho:** faça uma alteração em algum arquivo do código fonte (por exemplo a mensagem mostrada em [src/main/java/com/dhinfra/app/App.java](#)) e ao realizar o push observe o comportamento do pipeline.

#### 5. Um passo mais!

Em nosso arquivo .gitlab-ci.yml definimos os dois estágios (build e test), mas fizemos apenas um job para o build, é hora de criar um job para o estágio de teste, daremos apenas um dica, o comando para rodar o teste no Maven é "mvn test",  **você tem coragem de adicionar o job correspondente?**