



Infraestructura II

Realizamos nuestro primer Pipeline

Ya habiendo configurado nuestra cuenta en GitLab y habiendo subido nuestro primer código, en el día de hoy vamos a realizar un pipeline con 2 etapas.

Objetivo final de la práctica

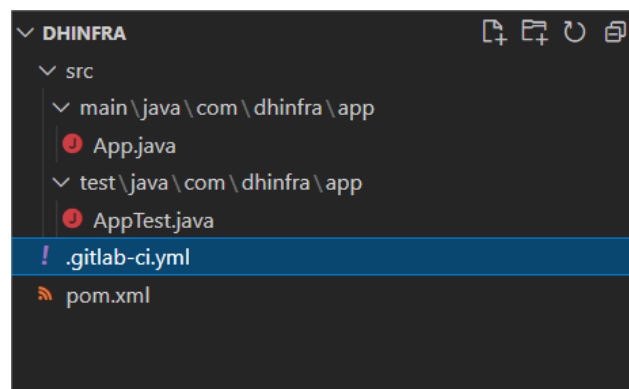
- Realizar un pipeline.
- Modificar parte de nuestro código para observar cómo afecta al pipeline
- Agregar una etapa a nuestro pipeline

¡Manos a la obra!

Como vimos en el [video](#), el lugar de definición de un pipeline en GitLab es el archivo `.gitlab-ci.yml`, el cual debe estar alojado en la raíz de nuestro proyecto.

Para ello, abrimos un Shell en **nuestro equipo**, nos dirigimos a la raíz de nuestro proyecto (o sea, dentro de la carpeta `dhinfra`) y allí creamos con el editor de texto de nuestra preferencia el archivo **`.gitlab-ci.yml`**

Ubicación del archivo `.gitlab-ci.yml` (Captura de Visual Studio Code)





1. Creamos los stages

La definicion del pipeline se da en stage y jobs, estos pertenecen a los stages, es decir, los primeros stages son los que estan a mayor nivel jerarquico.

La ejecucion de los stages es de acuerdo a como los definamos en nuestro archivo, por convencion para esta primer practica, construiremos dos stages

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

Agregamos en nuestro archivo `.gitlab-ci.yml` las lineas, recordemos siempre respetar el identado

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

Si bien esta definicion es correcta, basicamente este pipeline no hara nada ya que los stages carecen de jobs, que es donde se ejecutan los trabajos.

2. Agregamos los jobs

Los jobs se definen en el archivo y su pseudo estructura es

```
nombre_job  
  stage: "stage a la que pertenece"  
  script:  
    - "comandos a ejecutar en el job"  
    - echo "hola pipeline"
```

Nosotros en primer lugar vamos a definir un job para nuestro stage "build", el cual compilara el codigo Java que tenemos; para ello usaremos Maven como compilador, con lo cual el comando sera "mvn compile".

El codigo de nuestro `.gitlab-ci.yml` con el agregado del job debera ser:

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

Hasta ahora, definimos el pipeline, pero aun GitLab no se ha enterado de los cambios, es momento que lo haga y para ello debemos impactar estos cambios que aun estan en nuestro equipo en el repositorio, es por eso que debemos hacer push de este nuevo archivo, ejecutando en **nuestro equipo**:

```
git add .
git commit -m "Mi Primer Pipeline"
git push
```

3. Pipeline ejecutandose!

GitLab, al recibir los cambios en el codigo, detecta automaticamente la presencia del archivo .gitlab-ci.yml y si su estructura es valida, **EJECUTA** los pasos definidos en el, es por ello que ahora si vamos a nuestro GitLab → CI/CD → Pipelines, observaremos el listado de Pipelines ejecutados y alli estara el correspondiente al ultimo commit

Emilio Dorio > infra2v > Pipelines

All 7	Finished	Branches	Tags	Clear runner caches	CI lint	Run pipeline
Filter pipelines				Q	Show Pipeline ID ▾	
Status	Pipeline	Triggerer	Stages			
<div>passed</div> <div>00:00:49</div> <div>5 minutes ago</div>	Mi Primer Pipeline #22 test 6c52efff <div>latest</div>					

¿Podemos ver detalles de su ejecucion?

Si, ingresamos en el numero (en este caso de ejemplo el "#22") alli veremos los stages que se ejecutaron

Mi Primer Pipeline

1 job for **test** in 49 seconds

latest

[6c52efff](#)

No related merge requests found.

Pipeline Needs Jobs 1 Tests 0

Build

build_job

Aquí vemos que solo se ejecuto el stage BUILD, y su job "build_job", si ingresamos en el observaremos todo el detalle de la ejecucion.



Detalles de la ejecución del “build_job”, en donde se observa el BUILD SUCCESS

```

xus-compiler-api-2.8.4.jar
357 Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.2.1/m
aven-shared-utils-3.2.1.jar (167 kB at 524 kB/s)
358 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-manager/2.8.
4/plexus-compiler-manager-2.8.4.jar
359 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 70 kB/s)
360 Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.8.4/p
lexus-compiler-javac-2.8.4.jar
361 Downloaded from central: https://repo.maven.apache.org/maven2/org/ow2/asm/asm/6.2/asm-6.2.jar (111 kB at 191 kB/
s)
362 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-api/2.8.4/plex
us-compiler-api-2.8.4.jar (27 kB at 46 kB/s)
363 Downloading 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 8.0 kB/s)
364 Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0-M9/qdox-2.0-M9.jar
(317 kB at 418 kB/s)
365 Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-compiler-javac/2.8.4/pl
exus-compiler-javac-2.8.4.jar (21 kB at 25 kB/s)
366 [INFO] Changes detected - recompiling the module!
367 [INFO] Compiling 1 source file to /builds/testdh/infra2v/target/classes
368 [INFO] -----
369 [INFO] BUILD SUCCESS
370 [INFO] -----
371 [INFO] Total time: 42.480 s
372 [INFO] Finished at: 2022-03-30T06:38:41Z
373 [INFO] -----
375 Cleaning up project directory and file based variables
377 Job succeeded
  
```

build_job

Duration: 49 seconds
 Finished: 10 minutes ago
 Timeout: 1h (from project)
 Runner: #1 (nRjwKrU) GitLabDH

Commit 6c52efff
 Mi Primer Pipeline

Pipeline #22 for test

build

→ build_job

4. ¿Y en que momento se ejecuta de vuelta el Pipeline?

Al estar vinculado con el repositorio de código, ante cualquier cambio en mi código fuente (no solo en el .gitlab-ci.yml) el pipeline se volverá a ejecutar.

Manos a la obra: realizar un cambio en algún archivo del código fuente (por ejemplo el mensaje mostrado en src/main/java/com/dhinfra/app/App.java) y al realizar el push observar el comportamiento del pipeline.

5. ¡Un paso más!

En nuestro archivo .gitlab-ci.yml definimos los dos stage (build y test), pero solo hicimos un trabajo para el build, es momento de crear un trabajo para el stage de test, te vamos a dar solo una pista, el comando para ejecutar el test en Maven es “mvn test”, ¿se animan a agregar el trabajo correspondiente?