

14 de julio de 2017 | Introducción a GitLab CI con Maven

En ConSol usamos GitLab (<https://about.gitlab.com>) como nuestro servidor central de Git y estoy muy contento con su funcionalidad. Últimamente, he estado jugando con GitLab CI (<https://about.gitlab.com/features/gitlab-ci-cd/>) con el objetivo de descubrir si podemos usarlo en lugar de Jenkins (<https://jenkins.io>) , nuestro servidor de CI actual de elección.



Como la mayoría de nuestros proyectos usan Maven (<https://maven.apache.org>) , estaba particularmente interesado en configurar un trabajo de construcción Maven simple.

Para abreviar una larga historia, sí, usaría GitLab CI en mi próximo proyecto. Más adelante veremos por qué, pero primero quiero dar un tutorial rápido de GitLab CI.

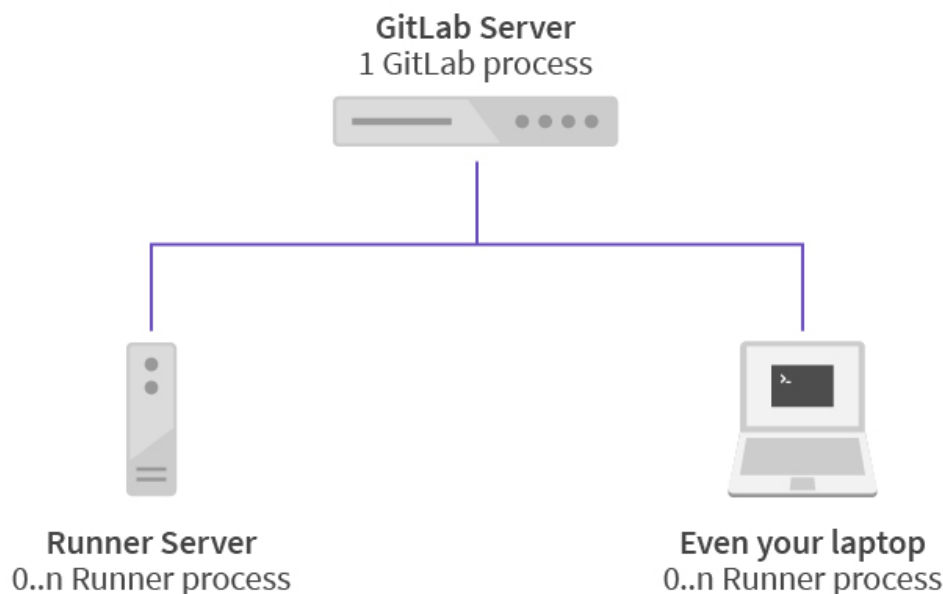
Requisitos

Para poder probarlo usted mismo necesita:

- Un servidor GitLab con Pipelines activado
- Un usuario de GitLab con `Master` rol en un proyecto
- Un servidor de repuesto donde los agentes de GitLab CI pueden ejecutarse (puede ser su propia computadora)

Cómo funciona GitLab CI

GitLab sigue un enfoque basado en el agente, es decir, en cualquier lugar donde desee ejecutar una compilación, necesita instalar un llamado GitLab Runner (<https://docs.gitlab.com/runner/>) . Puede instalar tantos GitLab Runners como desee y cada Runner puede ejecutar compilaciones simultáneas de 1 n, que es configurable.



En resumen, necesita dos cosas para comenzar con GitLab CI (aparte del propio GitLab):

1. Un archivo llamado `.gitlab-ci.yml` en la raíz del proyecto, que contiene las instrucciones de la línea de comandos para su compilación
2. Instale y registre al menos un GitLab Runner

`.gitlab-ci.yml` Configuración

He creado un proyecto público de GitLab donde puedes encontrar el código: `gitlabci-maven` (<https://gitlab.com/gucce/gitlabci-maven>) .

Nota Si bifurca el proyecto, no bifurcará a los Runners y deberá proporcionarle sus propios Runners.

Versión simple muerta

Para una compilación simple, esto es tan fácil como puede ser, poco más que el comando Maven mismo. Los siguientes contenidos son todo lo que necesita para ejecutar una compilación de Maven:

```
.gitlab-ci.yml
```

```
maven_build:
  script: mvn verify
```

Si la definición de CI es tan breve, ¿necesita confiar en una gran cantidad de definiciones implícitas que usted dice? ¡Por supuesto, hay mucha magia involucrada!

En primer lugar, GitLab aloja su código, por lo que obviamente sabe cómo clonarlo. Por lo tanto, no necesitamos agregar ninguna configuración relacionada con el repositorio de git.

En segundo lugar, suponemos que `mvn` (y, por lo tanto, también una JVM) está instalada (y en el `PATH`) en nuestro sistema de destino.

Versión mejorada

Un `.gitlab-ci.yml` aspecto más sofisticado y explícito podría verse de la siguiente manera (con comentarios que explican el significado):

```
# These are the default stages.
# You don't need to explicitly define them.
# But you could define any stages you want.
stages:
  - build
  - test
  - deploy

# This is the name of the job.
# You can choose it freely.
maven_build:
  # A job is always executed within a stage.
  # If no stage is set, it defaults to 'test'.
  stage: test
  # Since we require Maven for this job,
  # we can restrict the job to runners with a certain tag.
  # Of course, it is our duty to actually configure a runner
  # with the tag 'maven' and a working maven installation
  tags:
    - maven
  # Here you can execute arbitrate terminal commands.
  # If any of the commands returns a non zero exit code the job fails.
  script:
    - echo "Building project with maven"
    - mvn verify
```

Para obtener una especificación completa de todos los comandos posibles, consulte la documentación de GitLab CI YAML (<https://docs.gitlab.com/ee/ci/yaml/>)

Instalar un GitLab Runner

Ahora que tenemos nuestra `.gitlab-ci.yml` configuración, necesitamos un GitLab Runner.

Los corredores están escritos en Go (<https://golang.org>) y están disponibles para varias plataformas, incluidas Linux, Windows, MacOS, FreeBSD y Docker.

Consulte la documentación de GitLab sobre cómo instalar un GitLab Runner (<https://docs.gitlab.com/runner/install/>) .

Verifique su instalación:

```
$ gitlab-runner list
Listing configured runners                                ConfigFile=/etc/gitlab-runner/config.t
```

Como puede ver, hay cero GitLab Runners configurados, así que creemos uno (ya que `root` , de lo contrario, el runner no se puede instalar como servicio)

NOTA Puede obtener los parámetros necesarios para la inscripción en su proyecto GitLab bajo

Settings > Pipelines > Specific Runners

```
# gitlab-runner register
Running in system-mode.
```

```
Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com/):
https://gitlab.com/
Please enter the gitlab-ci token for this runner:
abcdefghijklm
Please enter the gitlab-ci description for this runner:
[build-n]: build-n-shell
Please enter the gitlab-ci tags for this runner (comma separated):
maven, java, ubuntu
Whether to run untagged builds [true/false]:
[false]:
Whether to lock Runner to current project [true/false]:
[false]: true
Registering runner... succeeded runner=HzUGN97U
Please enter the executor: parallels, virtualbox, docker+machine, docker, docker-ssh, shell
shell
Runner registered successfully. Feel free to start it, but if it's running already the con
```

Poco después de que haya registrado su corredor que aparece en la **Settings** > **Pipelines** > **Specific Runners** vista.

Runners activated for this project

 **c169ccd6**  





build-n-shell

java **maven** **ubuntu**








Ejecutando la compilación

Ahora el Runner está listo para usar. Cada vez que presiona una confirmación en su servidor GitLab, se activará la canalización. Si solo ha configurado un Runner con la etiqueta **maven**, siempre se ejecutará en este. De lo contrario, GitLab seleccionará al azar un Runner disponible.

Cada confirmación tendrá un icono adjunto que representa el estado de la tubería (ver también en gitlabci-maven (<https://gitlab.com/gucce/gitlabci-maven/commits/master>)).

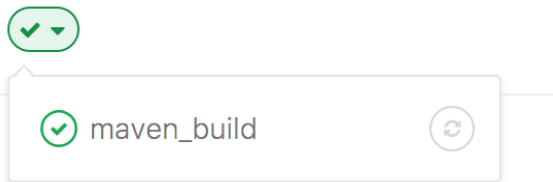
running	failed	passed
 e5537d02 	 47897476 	 71f2a8ce 

A continuación se muestra una descripción general de todas las canalizaciones que se han ejecutado **Pipelines** (ver en gitlabci-maven (<https://gitlab.com/gucce/gitlabci-maven/pipelines>)).


Project Repository Issues 0 Merge Requests 0 Pipelines Wiki Snippets Settings					
Pipelines Jobs Schedules Environments Charts					
All 4	Pending 0	Running 0	Finished 4	Branches	Tags
<div>Run Pipeline</div> <div>CI Lint</div>					
Status	Pipeline	Commit	Stages		
passed	#9811080 by  latest	master -> 71f2a8ce Update GitLab CI YAML ...	✓	00:00:10 about 3 hours ago	
failed	#9810889 by  latest	master -> 71f2a8ce Update GitLab CI YAML ...	✗	00:00:01 about 3 hours ago	
failed	#9810814 by 	master -> e5537d02 Add simple maven proje...	✗	48 minutes ago	
failed	#9807790 by 	master -> 47897476 Basic .gitlab-ci.yml file f...	✗	00:00:52 about 5 hours ago	

En esta página de resumen también puede ver qué trabajos se han ejecutado dentro de una tubería.

Stages



Y, por supuesto, puede inspeccionar los registros (ver gitlabci-maven (<https://gitlab.com/gucce/gitlabci-maven/-/jobs/22214945>)).

passed Job #22214945 in pipeline #9811080 for 71f2a8ce from master by  gucce about 3 hours ago

Retry job

Job details

Duration: 10 seconds
Finished: about 3 hours ago
Runner: #175037

Raw Erase

Commit title
Update GitLab CI YAML to use tag 'maven'

Tags
maven

maven_build maven_build

```
130/203 KB 221/221 KB
160/203 KB 221/221 KB
164/203 KB 221/221 KB
168/203 KB 221/221 KB
172/203 KB 221/221 KB
176/203 KB 221/221 KB
180/203 KB 221/221 KB
184/203 KB 221/221 KB
188/203 KB 221/221 KB
192/203 KB 221/221 KB
196/203 KB 221/221 KB
200/203 KB 221/221 KB
203/203 KB 221/221 KB

Downloaded: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0/plexus-utils-3.0.jar (221 KB at 2794.6 KB/sec)
Downloaded: https://repo.maven.apache.org/maven2/commons-lang/commons-lang/2.1/commons-lang-2.1.jar (203 KB at 2444.9 KB/sec)
[INFO] Building jar: /home/gitlab-runner/builds/c169cc68/gucce/gitlabci-maven/target/gitlabci-maven-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.577 s
[INFO] Finished at: 2017-07-12T18:06:31+02:00
[INFO] Final Memory: 18M/181M
[INFO] -----
Job succeeded
```

En mi opinión, esto está perfectamente integrado en la interfaz de usuario de GitLab.

Envolver

Eso es todo, ahora tiene una compilación de Maven que funciona con GitLab CI. Por supuesto, esto es solo el comienzo y debe consultar la documentación bien escrita de GitLab sobre sus flujos de trabajo de CI. Es posible que desee comenzar con la página README de GitLab CI (<https://docs.gitlab.com/ce/ci/README.html>) . También revisa GitLab's `.gitlab-ci.yml` para GitLab (<https://gitlab.com/gitlab-org/gitlab-ce/blob/master/.gitlab-ci.yml>) (perdón por el trabalenguas, no pude resistirme).

Jenkins vs. GitLab CI sobre Maven

Volvamos ahora a mi declaración inicial para preferir GitLab sobre Jenkins en mi próximo proyecto.

Desventajas de GitLab CI

Comencemos primero con sus desventajas.

- Solo basado en la línea de comandos, que no es para todos (en realidad lo prefiero)
- No es compatible con Solaris, HP-UX, etc. donde Jenkins se ejecuta debido a Java (en realidad tenemos proyectos ejecutándose en Solaris)
- Dependencia de un proveedor
- No hay sistema de complemento (pero puede usar cualquier comando de terminal disponible en su sistema de destino)
- No personalizable

Ventajas de GitLab CI

- Un archivo por proyecto contiene toda la compilación (la falta de ella es mi principal problema con Jenkins)
 - Versión controlada automáticamente
 - Diferentes ramas pueden tener diferentes construcciones
 - Las revisiones antiguas tienen automáticamente el archivo de compilación correspondiente
- Soporte Docker listo para usar (aunque no lo mencioné en el artículo)
- Desarrollo activo y reflexivo (los complementos de Jenkins suelen ser un desastre)
- Definición de compilación declarativa (que prefiero sobre el plugin Pipeline de Jenkins o Job DSL)

Y ahora pruébalo por ti mismo. Puedes bifurcar mi proyecto gitlabci-maven (<https://gitlab.com/gucce/gitlabci-maven>) si quieres.

Autor: Christian Guggenmos

Etiquetas: gitlab (/tags/gitlab) , integración continua (/tags/continuous integration)

Categorías: desarrollo (/categories/development)

