Continuous Integration con Travis CI

Table of Contents

Introduzione a Travis CI

Far riferimento alle seguenti presentazioni:

- https://www.slideshare.net/aneteknake/travis-ci-46453062
- https://www.slideshare.net/bsiggelkow/travis-ci

Scaricare il progetto

- Effettuare il fork del seguente progetto:
  - https://github.com/dduportal/dw-demo-app
- Effettuare il checkout del progetto presente nel repository personale
- Provare ad eseguire il processo di build del progetto eseguendo:
  - `mvn install`

Esecuzione dell'applicazione

L'artefatto generato dall'applicazione è un `jar` standalone.

NOTE	`jar` = "Java Archive". E' un archivio zip che contiene le classi Java compilate e i metadata

Questo jar contiene al suo interno un *embedded application server*. Per questo l'esecuzione dell'applicazione non richiede un *Java application server* come Tomcat o Jetty

NOTE	Questo tipo di jar è conosciuto anche come "Uber-JAR"

L'applicazione verrà eseguita in 2 modi:

- **On the metal**: eseguendo l'embedded application server "as-is".
- **Tramite Docker**: build di una Docker image ed esecuzione di un Docker Container

Esecuzione "on the metal"

Andiamo ad eseguire l'applicazione "as is". Per un problema di porte non sarà possibile accedere all'applicazione da un web-browser, sarà comunque possibile verificare l'esecuzione da lina di comando.

IMPORTANT	L'obiettivo è di evidenziare com'è possibile eseguire dei test di Integrazione e degli Smoke test da un server senza interfaccia grafica.

- Eseguire il comando java con i seguenti parametri:
  - Path al jar file `-jar'
  - Il parametro "server" come primo argomento
  - Il path al file di configurazione YAML di DropWizard
- Il comando risultante è:

```
java -jar ./target/demoapp.jar server ./hello-world.yml
```

- Una volta avviato verrà visualizzato il log nel terminale.
- Leggere le ultime linee di log:

```
...
INFO  [2016-08-31 16:38:12,271] org.eclipse.jetty.server.ServerConnector: Started
application@674658f7{HTTP/1.1}{0.0.0.0:8080}
INFO  [2016-08-31 16:38:12,272] org.eclipse.jetty.server.ServerConnector: Started
admin@5c8eee0f{HTTP/1.1}{0.0.0.0:8081}
INFO  [2016-08-31 16:38:12,272] org.eclipse.jetty.server.Server: Started @4018ms
```

- L'applicazione è attiva nella porta `8080`, e la consolle di amministrazione di DropWizard è attiva nella porta `8081`
- Eseguire il seguente comando:

```
curl -v http://localhost:8080/
```

Risultato atteso:

```
*   Trying ::1...
* Connected to localhost (::1) port 8080 (#0)
> GET / HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.50.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Wed, 31 Aug 2016 16:44:37 GMT
< Last-Modified: Wed, 31 Aug 2016 14:07:30 GMT
< Content-Type: text/html; charset=UTF-8
< Vary: Accept-Encoding
< ETag: "8f2181178186417ce600f7d5716bb124"
< Content-Length: 345
<
<!doctype html>

<head>
  <title>Hello Butler</title>
  <script src="angular.min.js"></script>
  <script src="hello.js"></script>
</head>



    <p>The ID is {{greeting.id}}</p>
    <p>The content is {{greeting.content}}</p>







* Connection #0 to host localhost left intact
```

- Ritornare al primo terminale. Verificare se viene visualizzato il seguente messaggio di log:

```
...
0:0:0:0:0:0:0:1 - - [31/Aug/2016:16:44:37 +0000] "GET / HTTP/1.1" 200 345 "-"
"curl/7.50.1" 5
...
```

- Terminare l'appicazione digitando la seguente combinazione di tasti: **CTRL-C**

```
...
^C
INFO  [2016-08-31 16:47:38,233] org.eclipse.jetty.server.ServerConnector: Stopped
application@674658f7{HTTP/1.1}{0.0.0.0:8080}
INFO  [2016-08-31 16:47:38,236] org.eclipse.jetty.server.ServerConnector: Stopped
admin@5c8eee0f{HTTP/1.1}{0.0.0.0:8081}
INFO  [2016-08-31 16:47:38,237] org.eclipse.jetty.server.handler.ContextHandler:
Stopped i.d.j.MutableServletContextHandler@66e889df{/,null,UNAVAILABLE}
INFO  [2016-08-31 16:47:38,239] org.eclipse.jetty.server.handler.ContextHandler:
Stopped i.d.j.MutableServletContextHandler@383790cf{/,null,UNAVAILABLE}
```

Continuous Integration con Travis CI

Di seguito vengono definiti i passi per configurare la continuous integration con Travis CI

Abilitare Travis CI ad accedere ai repository in GitHub

Seguendo la guida presente a questo indirizzo, abilitare Travis CI ad accedere ai repository presenti nel account GitHub.

Seguire i punti: * How * How > Getting Started

Configurare l'esecuzione del processo di Build in Travis CI

Travis CI Job lifecycle

Come descritto nella documentazione, il processo di Build in Travis CI viene configurato nel un file `.travis.yml` che deve essere presente nel repository del progetto.

Il processo di build è composto dalle seguenti fasi:

1. *OPTIONAL* Install apt addons
2. *OPTIONAL* Install cache components
3. *OPTIONAL* before_install
4. **install**: permette di installare le dipendenze richieste per eseguire il processo di build
5. before_script
6. **script**: permette di definire i comandi del processo di build
7. *OPTIONAL* before_cache (for cleaning up cache)
8. after_success or after_failure
9. *OPTIONAL* before_deploy
10. *OPTIONAL* deploy

11. *OPTIONAL* after_deploy
12. after_script

Come descritto nella documentazione il processo di build può fallire nel seguente modo:

- **errored**: se viene ritornato un non-zero exit code nelle prime 4 fasi
- **failed**: se viene ritornato un non-zero exit code nelle fase script o se le fasi successive sono in time-out

Caratteristiche dell'ambiente di esecuzione del processo di build

Le caratteristiche degli ambienti dov'è possibile eseguire i processi di CI è specificato in questa pagina di documentazione.

Configurare il processo di build per il progetto di esempio

- Aprire il progetto
- Creare il file `.travis.yml` contenente il seguente contenuto:

```
dist: trusty

group: edge

language: java

sudo: required

jdk: openjdk8

cache:
  directories:
    - $HOME/.m2
script:
  - mvn clean install
```

Il processo di build eseguirà in un ambiente con le seguenti caratteristiche:

- **Sistema Operativo**: Ubuntu Trusty 14.04
- **Linguaggio del progetto**: Java in questo modo nell'ambiente sarà presente maven (vedi qui)
- **Il tipo di Jdk**: open jdk 8
- Viene specificata di preservare la cartella .m2 (con le dipendenze)
- viene definito **il processo di build** composto dal comando:
  - `mvn clean install`

Se tutto è andato a buon fine, all'indirizzo https://travis-ci.org/ sarà possibile vedere l'esecuzione e lo stato della build.

Abilitare le notifiche

Come definito nella pagina di documentazion, Travis CI invierà di default una mail al autore dei commit per i seguenti eventi:

- Quando una build è fallita o con errori
- Quando viene ripristinato il corretto funzionamento di una build

Come definito nella documentazione è possibile attivare diversi meccanismi di notifica, tra cui:

- Canali IRC
- Chat (campfire, HipChat, Slack, etc)
- Web Hook

L'esito della build può essere inserito anche attraverso un'immagine in una delle pagine del repository. Per maggiori informazioni vedi qui.

NOTE	Nel processo di build possono essere integrati diversi componenti come sonarcloud e coveralls. Per maggiori dettagli vedi la parte di *Integrations and Notifications*.