Coding Challenge - Support Document

Contents

[To Run/Test the Anagramstring Standalone Application 2](#_Toc44952042)

[To Run the HttpServer AnagramEndpoint project 3](#_Toc44952043)

[To Test the HttpServer AnagramEndpoint Project 4](#_Toc44952044)

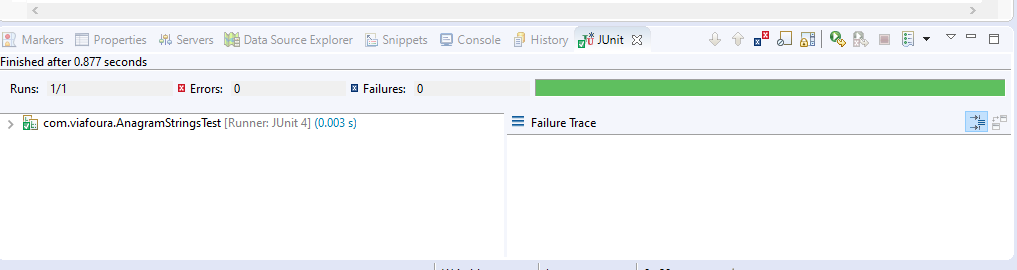
[To deploy/Build the Project in Docker Container Using Dockerfile: 8](#_Toc44952045)

[To deploy/Build the Project in Docker Container Using docker-compose.yml: 9](#_Toc44952046)

# To Run/Test the Anagramstring Standalone Application

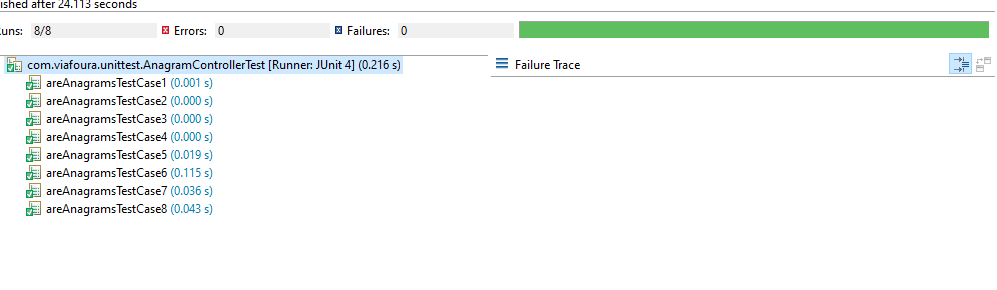
Step By Step for the coding challenge 1 and 2

1. Download Anagramstring project and save in C:\temp directory in Windows machine
2. Open Eclipse and import this zip file by navigating File🡪Import🡪General🡪Existing Project into workspace -> Select root directory🡪Browse🡪Anagramstring
3. Build the project and run com.viafoura.Anagram\_Strings as a Java application
4. The above standalone application takes two strings from the junit test class(com.viafoura.AnagramStringsTest) and checks whether the two strings are anagrams of each other and returns Boolean value.
5. Run com.viafoura.AnagramStringsTest as a JUNIT Test.



# To Run the HttpServer AnagramEndpoint project

1. Download Anagramendpoint project and save in C:\temp directory in Windows machine
2. Open Eclipse and import this zip file by navigating File🡪Import🡪Maven🡪Existing maven project into workspace🡪 Select root directory🡪Browse🡪 Anagramendpoint
3. Build the project and run com.viafouraAnagramMainApplication as a Java application
4. Run com.viafoura.unittest.AnagramControllerTest as a JUNIT Test

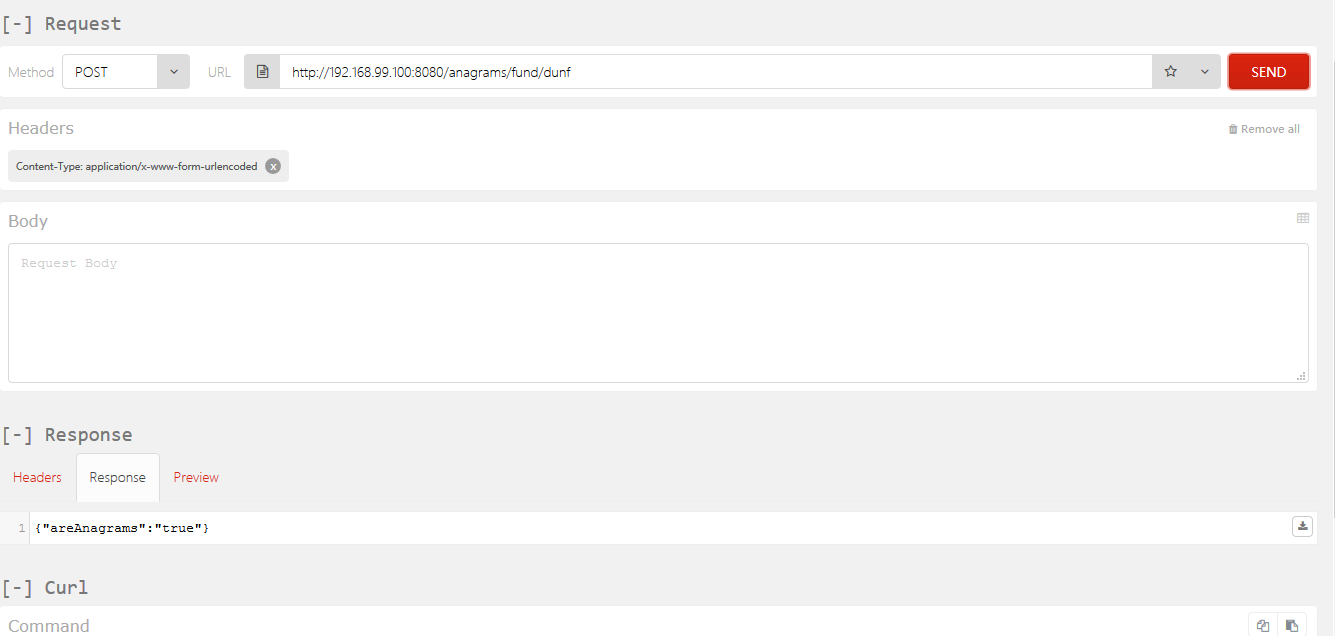


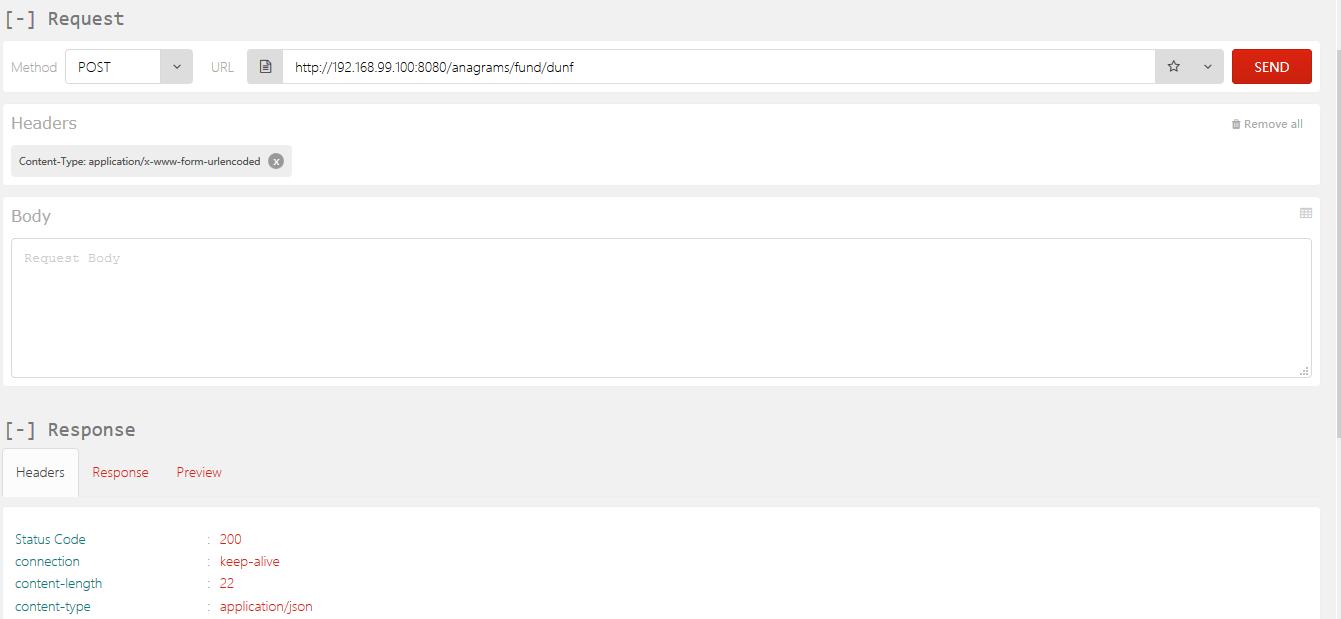
# To Test the HttpServer AnagramEndpoint Project

1. Launch Fiddler/Postman/any REST Client and compose the POST request as below,
   1. URL – [Listed](http://localhost:8080/anagrams/test) below in test scenarios
   2. Set the Content-Type as application/x-www-form-urlencoded
   3. Request type should be POST
   4. Click on Send button
2. Test Scenario – Passing two Strings
   1. TRUE when the strings are anagrams
      1. <http://localhost:8080/anagrams/fund/dunf>

we can also use the docker container ip and the port ,

<http://192.168.99.100:8080/anagrams/fund/dunf>

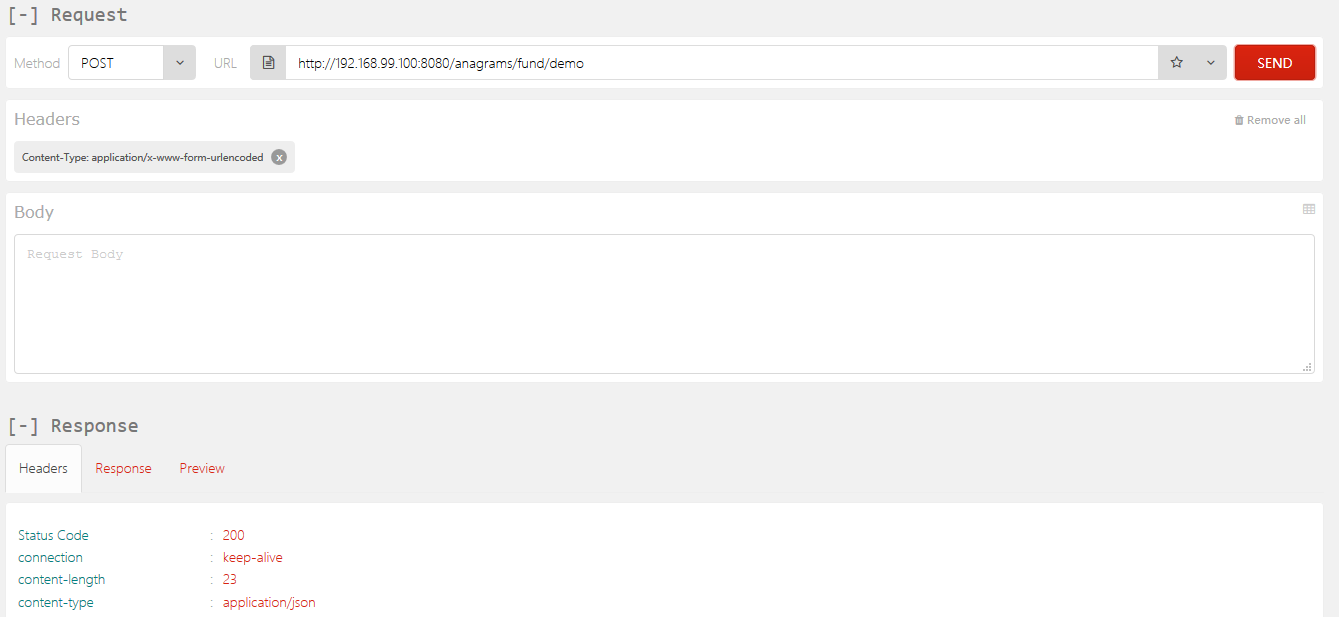


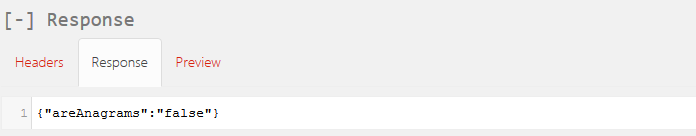


* 1. FASLE when the strings are not anagrams
     1. http://localhost:8080/anagrams/fund/demo
     2. <http://localhost:8080/anagrams/fund/fun>

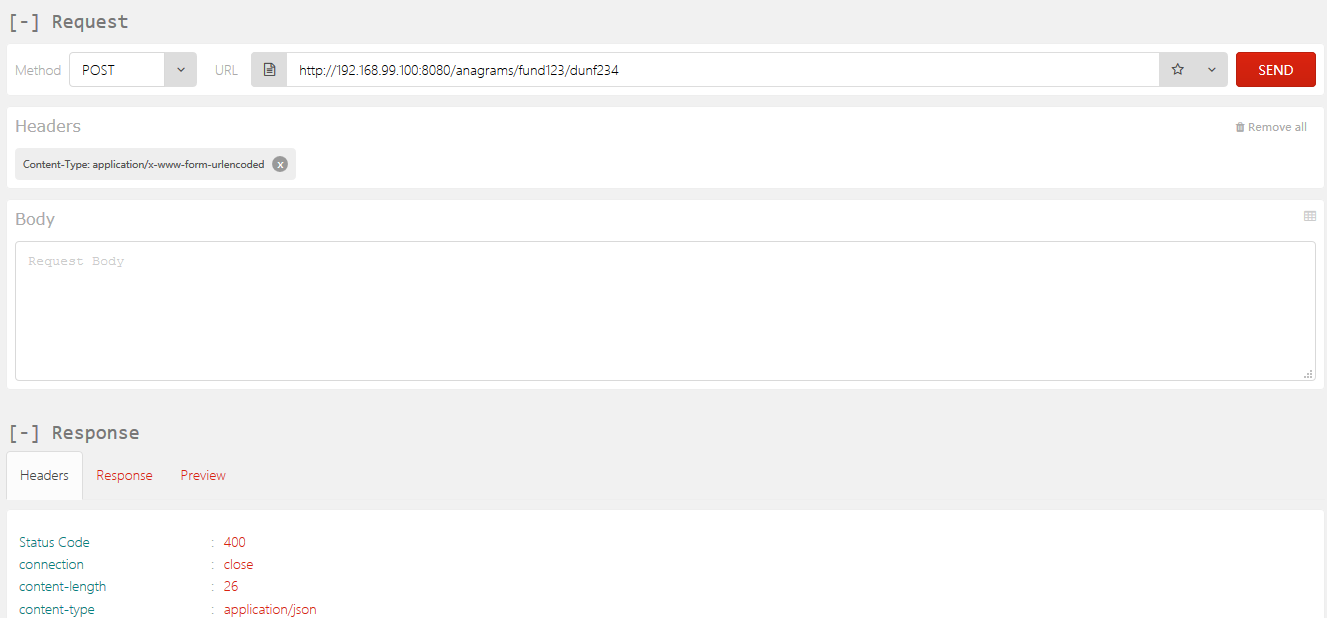
Docker machine ip and url:

<http://192.168.99.100:8080/anagrams/fund/demo>



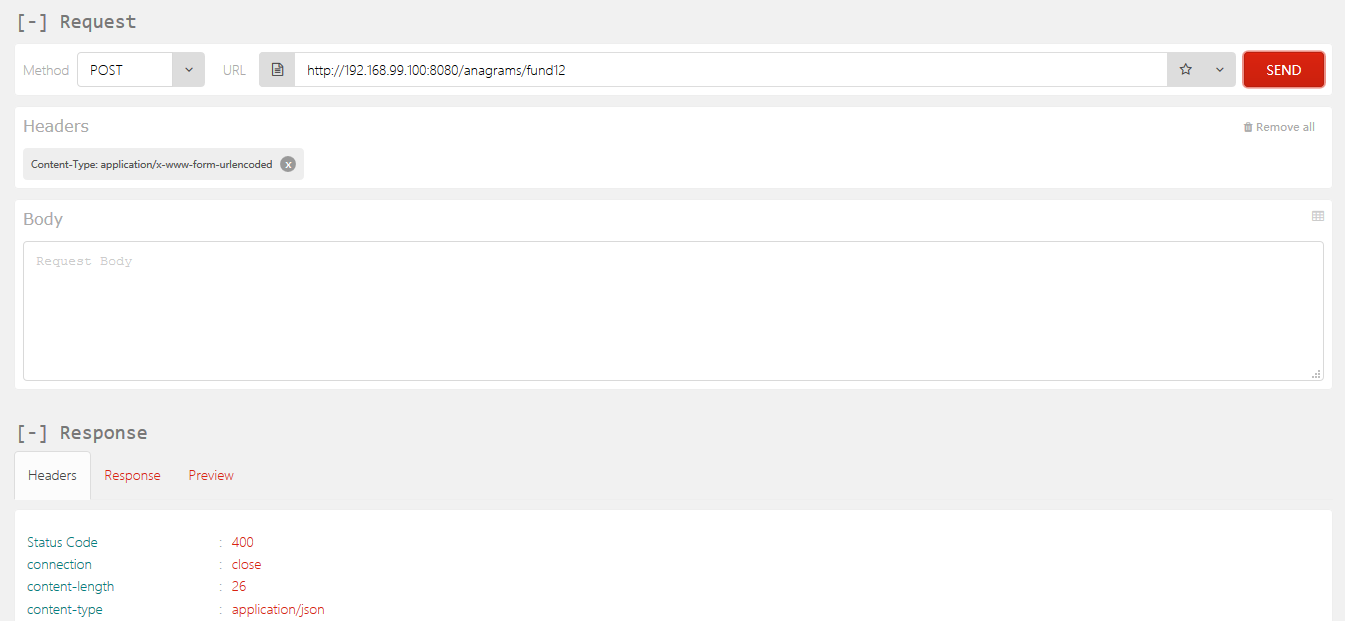


* 1. BAD REQUEST
     1. http://localhost:8080/anagrams/fund123/dunf
     2. http://localhost:8080/anagrams/fund/dunf123
     3. <http://localhost:8080/anagrams/fund123/dunf234>



1. Test Scenario – Passing one String
   1. BAD REQUEST
      1. <http://localhost:8080/anagrams/fund12>

http://192.168.99.100:8080/anagrams/fund12



* 1. RETURN all possible anagrams
     1. <http://localhost:8080/anagrams/fund>

<http://192.168.99.100:8080/anagrams/fund>



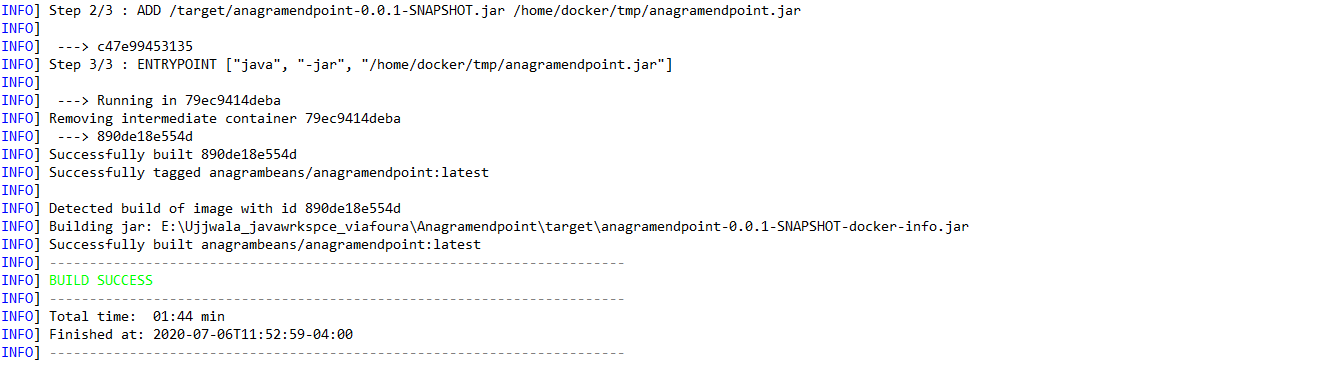
# To deploy/Build the Project in Docker Container Using Dockerfile:

Basic Steps to start/stop the docker, ip, to check the docker container files and folders:

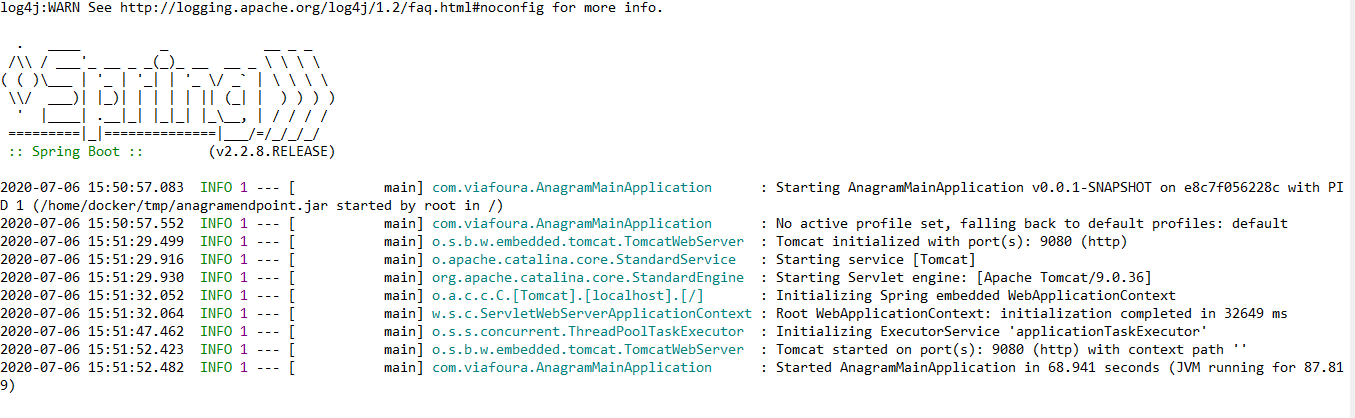
1. docker-machine start
2. docker-machine stop
3. docker-machine ip
4. docker-machine ssh default

Steps for building the project using dockerfile:

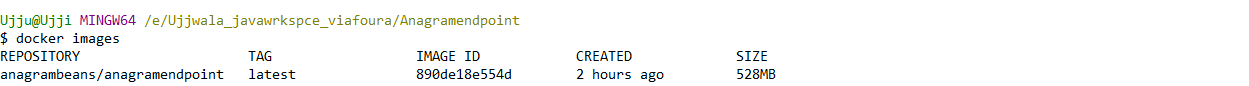
1. mvn clean install dockerfile:build



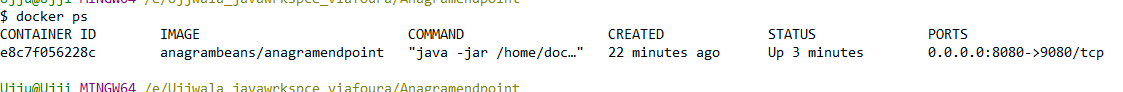
1. docker run -p 8080:9080 -t anagrambeans/anagramendpoint --name anagramendpoint-image



1. docker images



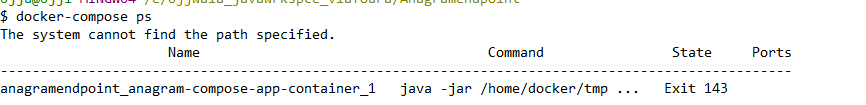
1. docker ps



# To deploy/Build the Project in Docker Container Using docker-compose.yml:

1. docker-compose config
2. docker-compose up –build





1. docker-compose down