Number to Word

DECEMBER 13 **2019**

Console, REST-API endpoint, highly scalable

Developed by: Selva Vadivel
vselvakumaranv@hotmail.com

Documentation

QUICK DEMO

1. **OPTION 1: View live demo at AWS on my personal site https://dice.red:**

a. **Access REST API using the end-point in a tool like Post-man for GET:**

https://api.numtoword.dice.red/numtoword/123

SAMPLE OUTPUT:


```
{
    "numberInputStr": "123",
    "words": "One hundred and twenty three"
}
```


b. **Web-app Console URL:**

https://console.numtoword.dice.red/numtoword-mvc-dash/

SAMPLE OUTPUT:

After clicking "Convert" button, it will take you to the results page. It is configured at AWS using Route 53 – geolocation policy setting and routed to corresponding load balancers with non-scaling and auto-scaling groups behind it with EC2 instances. It is best practice to set both the US and non-US users routing load balancer to point to auto-scaling with different min and max instances for scaling instead of making one load balancer point to non-scaling set of instance and the other to auto scaling group. While it is using only http here, it is best practice to use the https for having an encrypted communication which also works for both the console URL and REST-API endpoint. The console is setup to auto-forward from http to https

COMMON TROUBLE SHOOTING:

You may get a 404 error. Please try again using http://console.numtoword.dice.red/numtoword-mvc-dash/ for the console or the

REST API endpoint. **This can be built in as a feature to try twice on a 404 error but it is not included in this present release.**

2. **OPTION 2: Setup demo on your own:**

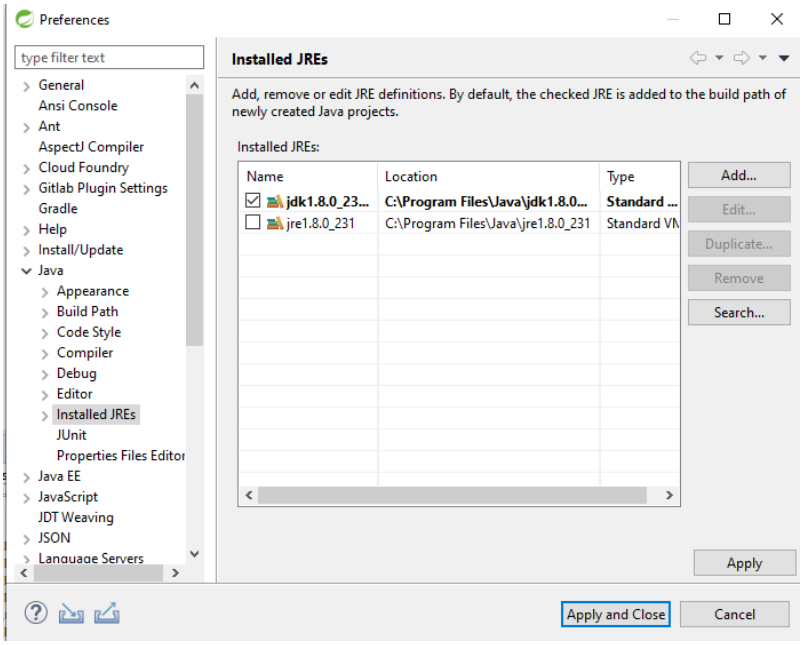
Recommended Environment: Use Eclipse IDE, Maven, TestNG and Spring Tool Suite 3.9

Step 1: Check out code from repo

https://repo.dice.red/vselvakumaranv/sonatype

Common Troubleshooting Issue: Compiler not found.

Solution: Add JDK as below from Window - > Preferences.



Step 2: Run - > Maven clean and Maven - > Project Update in your IDE

Step 3: pom.xml of numtowords-service-word: Run - > Maven Install

This will generate the JAR file after running the JUNIT tests

Step 4: pom.xml of numtowords-api: Run - > Maven Install

- **This will generate the JAR file for the REST-API.**

Then run the following command

java -jar {above generated filename}.jar

This will start the API. You can access the GET endpoint using a tool like POST-MAN using the

http://localhost:8080/numbertoword/123

where 123 is the number and it will return a JSON with the "words".

You can further configure the file for the port number by the following:

- **In the same folder of the JAR file, create a folder called "config". Inside the config folder create a file called application.properties.**
- **For the application.properties file of numtowords-api use the following. You may change these ports and name as you deem appropriate for each running instance:**

```
server.port=8081
spring.application.name=NumToWordAPIproduces
appInstance.name=instanceAPIapp01
```

Generating the WAR file for the Online Dashboard/Console:

Step 1: Run -> Maven Install on numtoword-mvc-dash for generating the WAR file.

Step 2: Update the URL in application.properties file inside the WAR file of numtoword-mvc-dash, update the api.hostname variable to the REST-API endpoint.

For example:

http://localhost:8080/numbertoword-mvc-dash/

Step 3: Place the WAR file at the webapps folder in case you are using Tomcat.

Please refer to the Common Troubleshoot Issues at page 23 for possible issues you may face.

Step 4: TESTING WAR file:

Run -> Maven Test in the Eclipse on the pom.xml file of "numtowords-console-loadtesting".
This will do the testing of several test cases of Selenium with TestNG. The results can be accessed at target/surefire-reports/index.html from Eclipse IDE. Remember to update the URL with the location of your REST-API at "testng.xml"

SPECIFICATIONS

I DESCRIPTION

This tool converts numbers into its corresponding words. This tool can be used using

1. The REST-API GET endpoint or using the console online.
2. The console consumes the REST-API endpoint.

II REQUIREMENTS

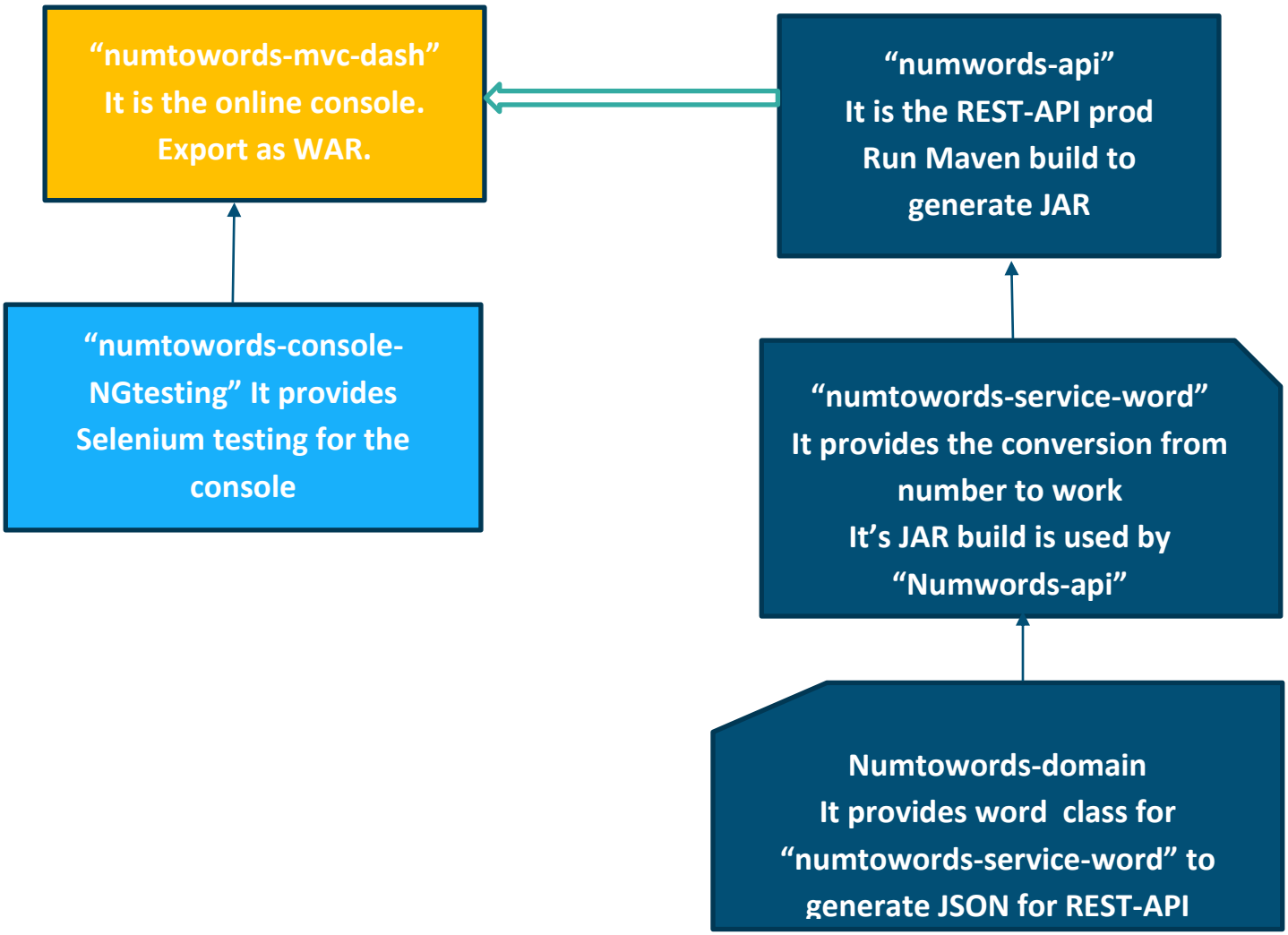
Commas and hyphens (indicating negative input) are the only non-numeric characters that are considered valid input. With commas, there are extra rules, however. For example, 1,000 is valid input, but 10,00 would be invalid. Only need to support whole numbers. Negative numbers are valid input as well. However, fractional, irrational, and decimal numbers can be treated as invalid input. Only need to support values that fit within a signed 32-bit integer. For java, then the range would be from java.lang.Integer.MIN_VALUE to java.lang.Integer.MAX_VALUE (inclusive). There is no specific set of frequent numbers. Reviewers will try various combinations to try to crash your submission and to make sure that all values are properly supported. An API alone isn't quite enough, would like some means included that lets a user execute the program. A single number at a time is what is expected as input at a time, other forms of input is invalid. An end-user will not expect to use the program without an internet connection. There is no authorization requirement or rate limiting. There will be no analytics attached to the program. Users will typically only submit a single request at a time. The peak time for requests is during school hours in the US - in particular, during the hours when all schools are in session (roughly 11am to 3pm eastern time). The only future direction is that for the program to be eventually included in popular, free Linux distributions such as Debian, Ubuntu, and Fedora.

III ANALYSIS OF REQUIREMENTS

To meet above requirement, the program was developed in Java using Spring framework. It was deployed in AWS with Auto-scaling for US users and setting Route 53 with policy for routing users by the geolocation to non-scaling EC2 instances and

auto-scaling groups behind load balancers from 11 am to 3 pm EST during school hours in the US, which is 16:00 - 20:00 UTC Mon-Fri. There is really no need for logging as part of the Java application. Also, Spring Security dependency is not required as there is no authorization and authentication requirements. AWS Inspector to test vulnerabilities for both the network and application is only needed in addition to Application level testing using Junit, Selenium with TestNG and input pattern validation

MULTI-MODULE MAVEN SETUP at Eclipse IDE



The console is one project and all other modules are one other project as a multi-module maven in the same workspace. It is setup in this way for the convenience of building the tool.

IV CODE

The code repository of this tool is at Gitlab. You may download the tool and open it in an IDE like Eclipse. The tool was developed using Spring Tool Suite 3 for Eclipse. The tool has been setup as a multi-module Maven project.

'numtowords-api' produces the REST-API GET endpoint. Please look into its application.properties file for configuration settings. 'numtowords-api' has a dependency on 'numtowords-service-word'. 'numtowords-service-word' provides the core functionality to convert number to words. It has optimized number of JUnit test cases to validate the boundary conditions for both valid and invalid inputs.

'numtowords-mvn-dash' consumes the REST-API GET provided by 'numtowords-api'. It has numberConsole.jsp which will be used by end users to enter the numbers. Then this tool validates user input for the format. The text box is already limited to 14 characters which is the maximum character size of the input as per the set boundary conditions and provides validation with feedback to enhance user experience. The api and console are separated for both software functionality and hardware resource so to support loosely coupled architecture.

Value Stream: Developing success measurements and having only the needed tools is a challenge in today's DevOps which is flooded with several alternatives. So very often, tools can become redundant and not being part of the value stream. One such a case is that for having application logging in this case.

"Logging: For this application, AWS CloudWatch is sufficient as no additional information is needed, so logging like logback is not part of the application"

DOWNLOAD AND RUN

Refer Page 4, this is only more details

To get started right away with setting up a demo on own machine, follow the steps in page 4 and below instructions. Remember to modify the application.properties file in each of them. This is only further more details if it is not clear in page 4

Build on your own:
If you would like to build and generate the WAR and JAR files, follow the step below:
Step 1: Meeting the following system requirements:

a. JDK 8
b. Spring Tool Suite 3 - Eclipse IDE
c. Dependencies are added in pom.xml. In particular:
- Undertow is included as dependency in the pom.xml for API and the default Tomcat is excluded.

d. Apache Maven 3.6.3

Step 2: The code can be downloaded from the GitLab repo.
Step 3: Open in an IDE like Eclipse. It is recommended to use Spring Tool Suite 3 for Eclipse as the tool was built using it. Java 8 JDK, Apache Tomcat 9.0.29-windows-x64 and Apache Maven 3.6.3 were used for developing this tool. Tomcat was later excluded, and Undertow was included as dependency in pom.xml. This was done to improve the performance of the tool and extend the support for Event Driven Architecture.
Step 4: If you are going to test locally, first run the 'numtowords-api' as Springboot app. Then run 'numtoword-mvc-dash' as a Java Application. There is one application.properties file for each 'numtowords-api' and 'numtoword-mvc-dash'. These application.property files have the settings for these two modules.
Step 4: Set the goal to 'clean install' in case you have not already set it for any of the modules. Maven install 'numtowords-api' and it will create a runnable jar that can be deployed where you would like to run it. Select Maven clean if there is any error which is common because of IDE/Eclipse cache settings. Similar it can be done for 'numtoword-mvc-dash' as well.
Note:

For one of the requirements where the usage is expected to be higher from 11:00 am to 3:00 pm EST in the US, the auto-scaling group is set for that time period at AWS live demo setup.

Other Details:

- If you would like to use the default Tomcat instead of Undertow, you may remove the <exclusion> tag for Tomcat and remove dependency tag for Undertow from pom.xml to default to Tomcat. This application was tested using Apache Tomcat 9.0.29-windows-x64 as well to compare for performance.

Testing WAR file: Refer page 5

TESTING AND QUALITY

The following are the different ways by which this tool was tested, and high-quality tool was developed. The following modules are opened in Eclipse IDE to run the tests.

I. **REST-API ENDPOINT TESTING**

1. JUnit at 'numtowords-service-word' module performs 13 different test cases. It will be run every time Maven install is run.

II. **APPLCATION CONSOLE TESTING**

- Install TestNG for Eclipse IDE. If you are new reference links below.
- User Maven clean, Maven Test with the pom.xml file at "numtowords-console-loadtesting".
- **TO DO: Change the value of the URL parameter at testng.xml with the URL you are using for testing.**

  It is currently set as:

```
<parameter name="url-param" value="http://console.numtoword.dice.red/numtoword-mvc-dash/" />
```

2. Selenium webdriver for Google Chrome browser is setup in the module 'numtowords-console-NGtesting'. TestNG library is added to the build path. Run Maven Test to do this test and generate the report. The report can be viewed at /test-output/index.html
3. Validation at numtoword-mvc-dash module for the input text field. @Pattern annotation and other special character checks as performed. Appropriate error message is displayed on the webpage if the pattern is not matched. Also, the textbox is limited to 14 characters to meet the boundary conditions.

Report generated for Test Results of Selenium with testNG are provided in page 25 from target/surefire-reports/index.html from Eclipse IDE.

New to TestNG?

- To add this library to your Eclipse IDE, please refer to the following link: https://www.techbeamers.com/create-testng-test-case-using-selenium/
- If you are new to TestNG, please read the following documentation to get TestNG: https://www.techbeamers.com/testng-tutorials/

SCREENSHOTS

In case you prefer using the Maven CLI, then just use the regular mvn commands to clean and run the build. If you prefer using the IDE, see below.

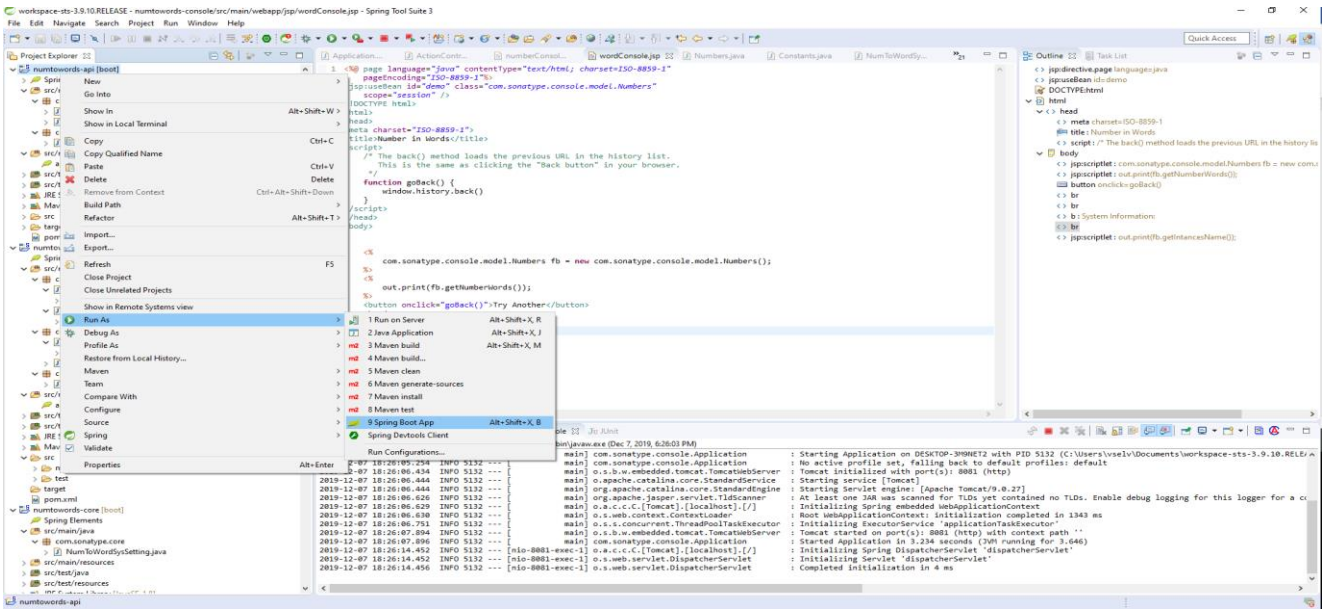


Figure 1: The above screenshot shows how to start the API from Eclipse IDE with Spring Tool Suite 3

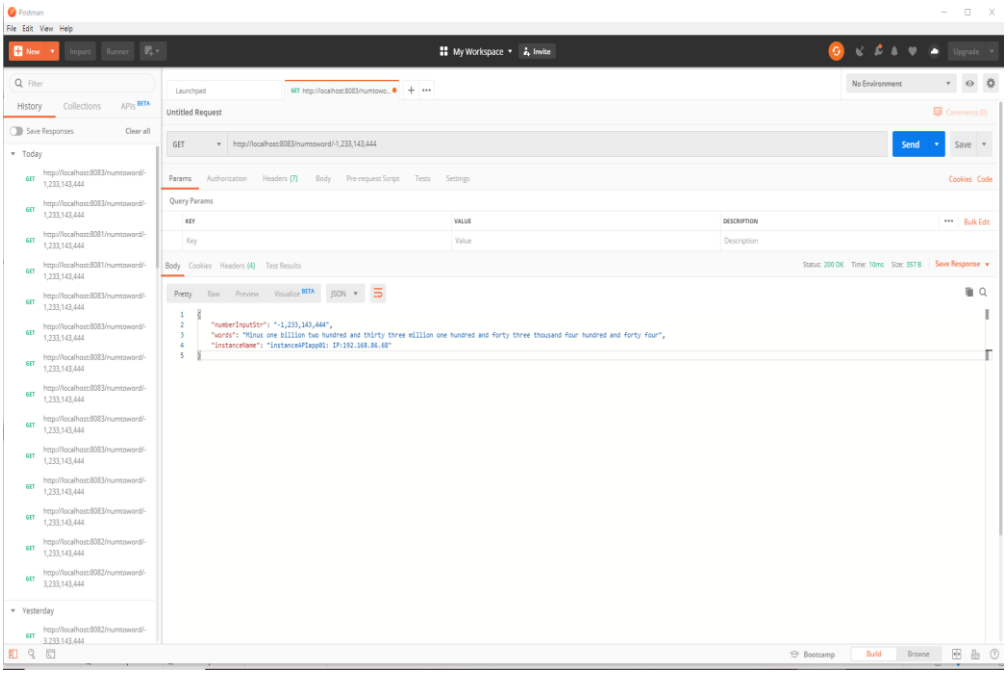


Figure 2: The above screenshot shows the response for REST-API endpoint at POSTMAN

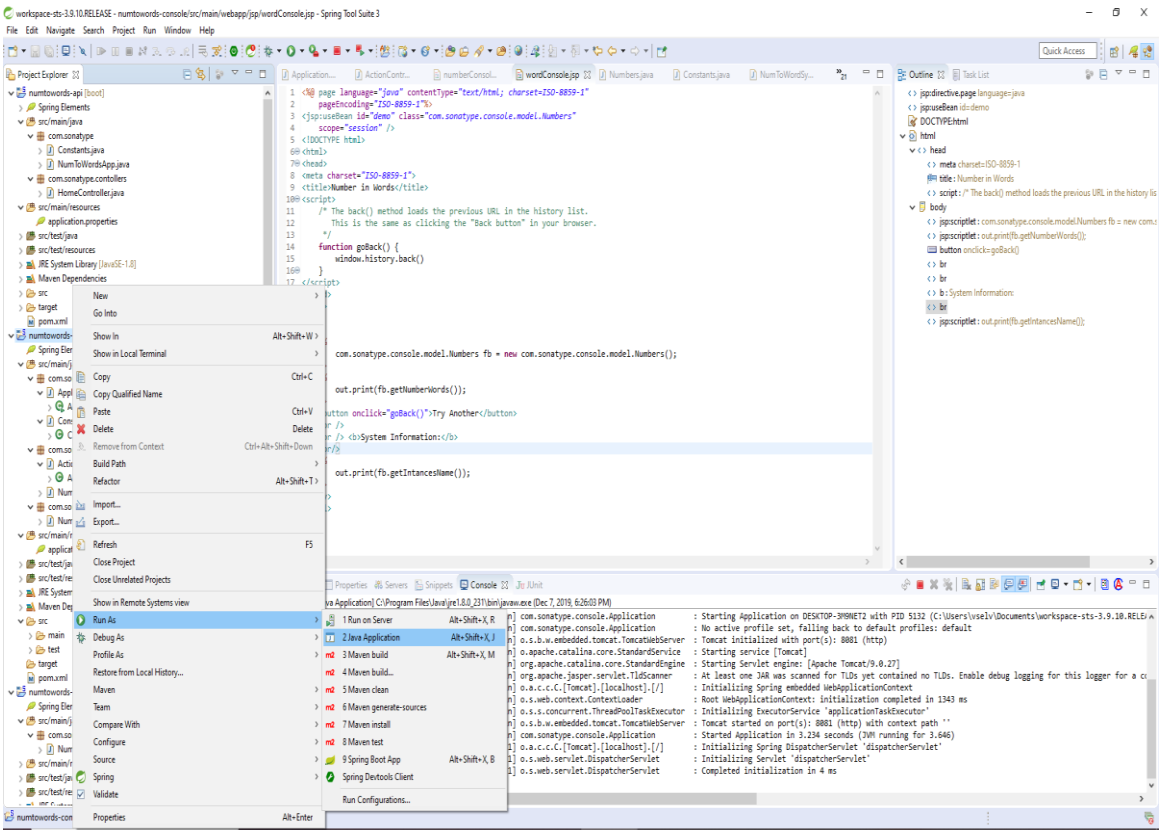


Figure 3: The above screenshot shows how to start the console from Eclipse IDE. You may also export it as a WAR file, or do a Maven build.

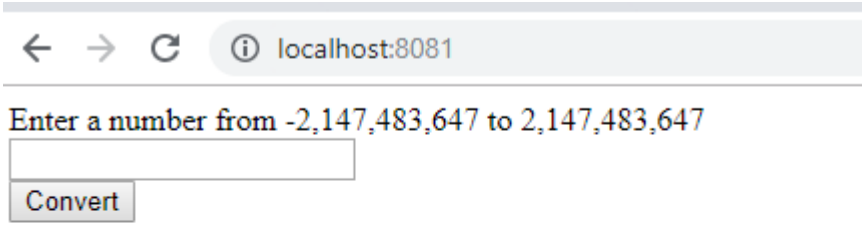


Figure 4: The above screenshot shows console at Google Chrome browser.

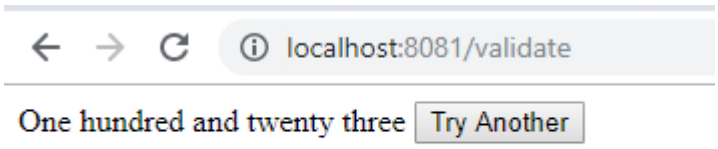
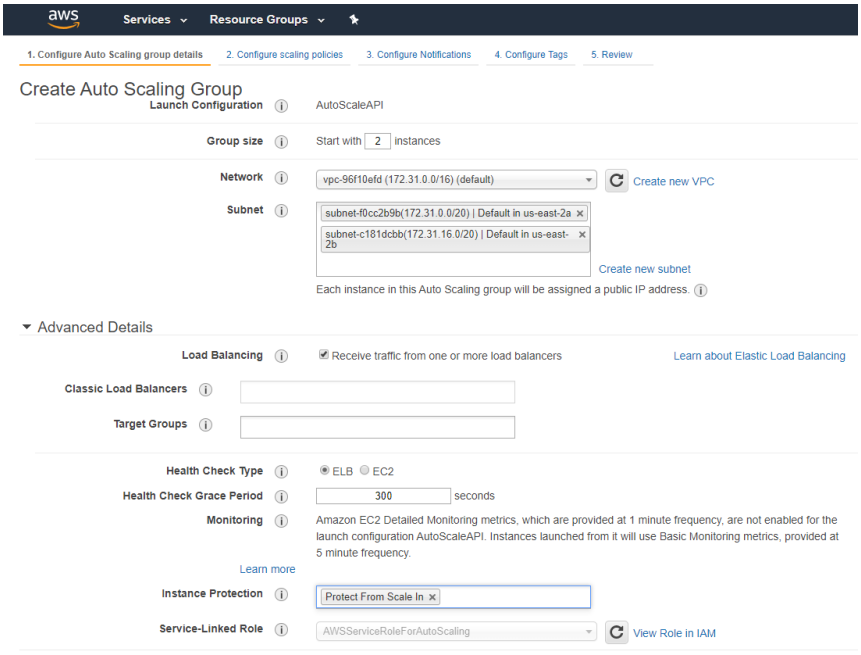


Figure 5: The above screenshot shows console result page at Google Chrome browser.

Even though the load balancer will auto scale only for US users and not in the case of other non-US users for the current setup, the instances between non-scaling and auto-scaling is shared by the US users. However, it is best practice to auto-scale both non-US and US users' load balancers accordingly.

One of the requirements is to have the tool be able to serve growing number of users in the US between 11:00 am and 3:00 pm eastern. To address this, the following setup at AWS to auto scale by cron 16:00-20:00 UTC time using route 53 geolocation routing is done.

Auto Scaling Group Setup at AWS:
When creating the image of the AMI when setting the configuration of auto-scaling group, it is important to include the bash shell script and related Linux commands to auto-start jar file for REST-API.
There is no information on predicted amount of traffic. So, the number of instances for maximum is selected to a low number and alarm is set so that this can be updated once the utilization has increased and it is within budget to increase the number of instances.



Setting up Auto Scaling Group Setup

Create Auto Scaling group | Actions

Filter: Filter Auto Scaling groups...

Name	Launch Configuration /	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grac
APIAutoScale...	api-reboot-auto-config-...	2	2	1	3	us-east-2a, us-east-2b	300	300
autoScaleUS	AutoScaleAPI	2	2	1	2	us-east-2a, us-east-2b	300	300

Add policy

Decrease Group Size

Policy type: Step scaling

Execute policy when: awsec2-autoScaleUS-High-CPU-Utilization
breaches the alarm threshold: CPUUtilization <= 50 for 300 seconds
for the metric dimensions AutoScalingGroupName = autoScaleUS

Take the action: Remove 1 capacity units when >= CPUUtilization > -infinity

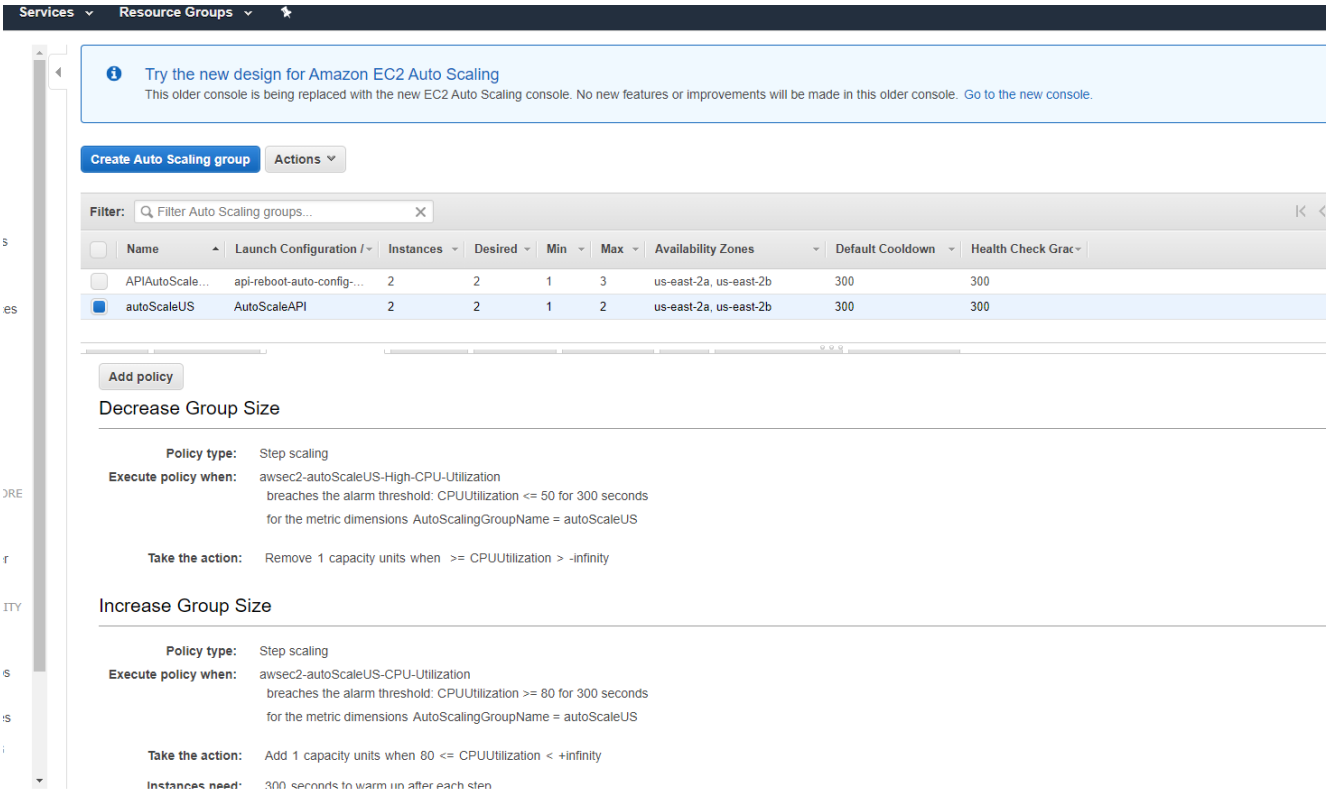
Increase Group Size

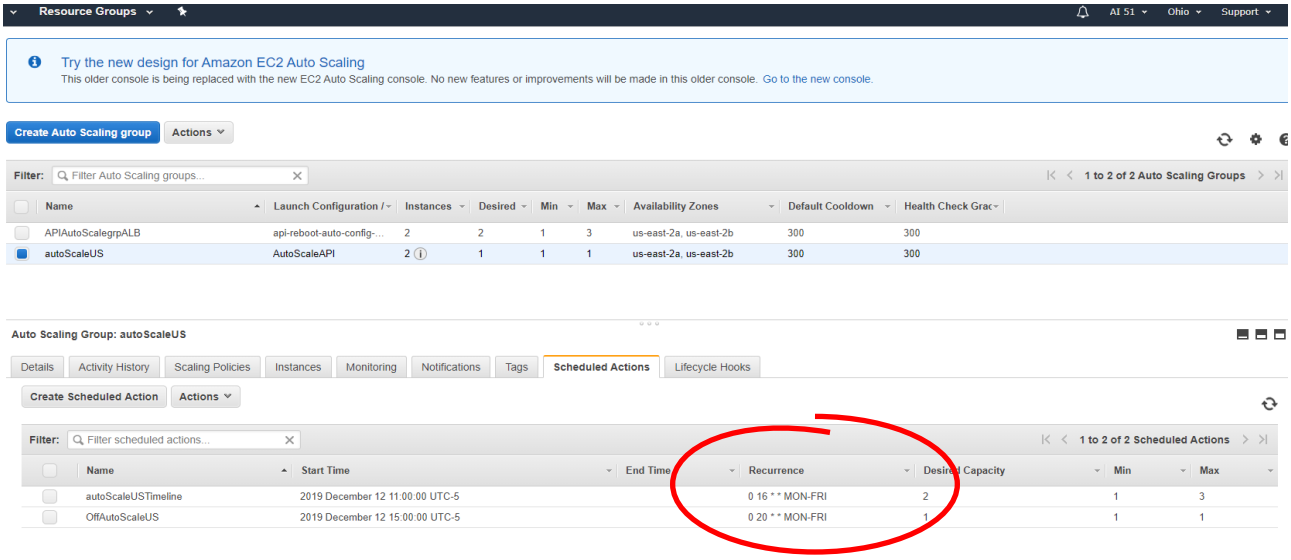
Policy type: Step scaling

Execute policy when: awsec2-autoScaleUS-CPU-Utilization
breaches the alarm threshold: CPUUtilization >= 80 for 300 seconds
for the metric dimensions AutoScalingGroupName = autoScaleUS

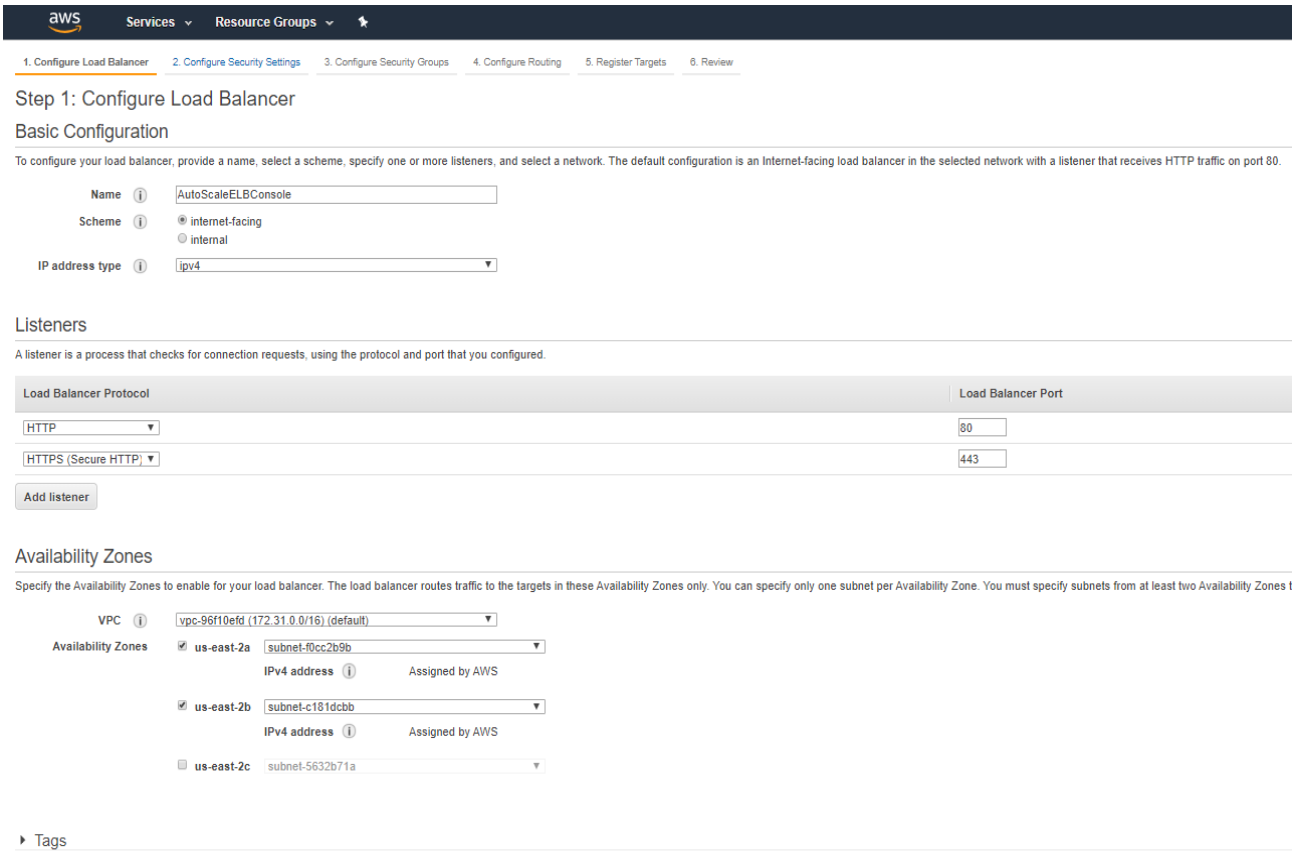
Take the action: Add 1 capacity units when 80 <= CPUUtilization < +infinity

Instances need: 300 seconds to warm up after each step

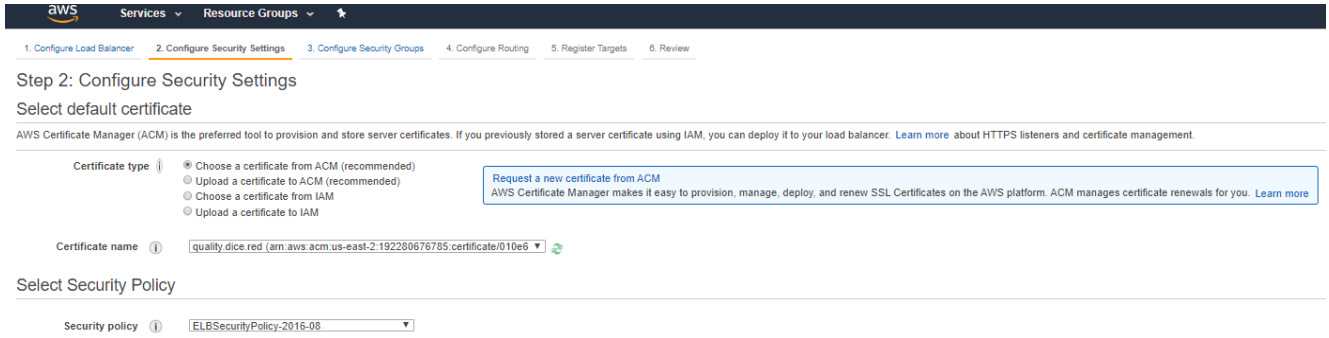




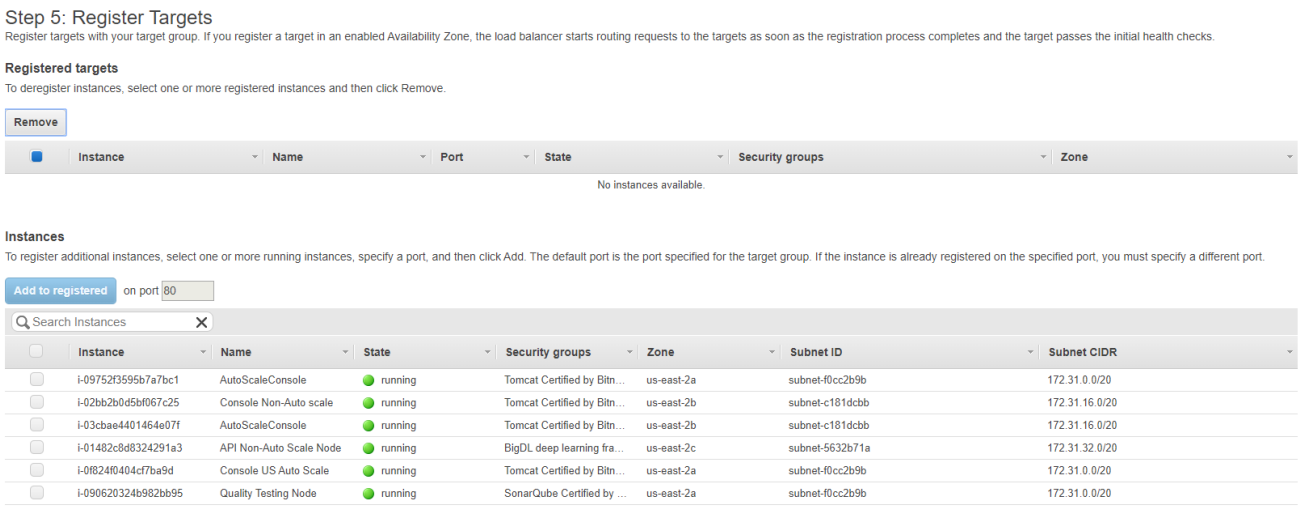
The above figure shows that the auto-scaling of the cron is set between 11:00 am (increase size) and 3:00 pm EST (decrease size) i.e. 16:00-20:00 UTC Mon-Fri to meet the requirement, traffic requirement during school hours.



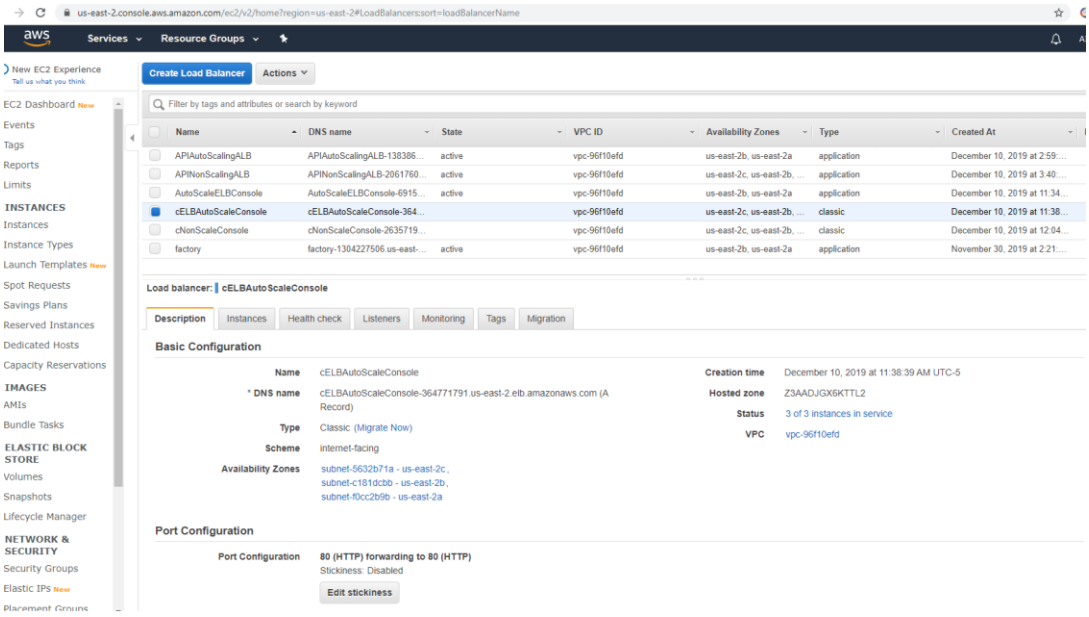
Configuring Load Balancer



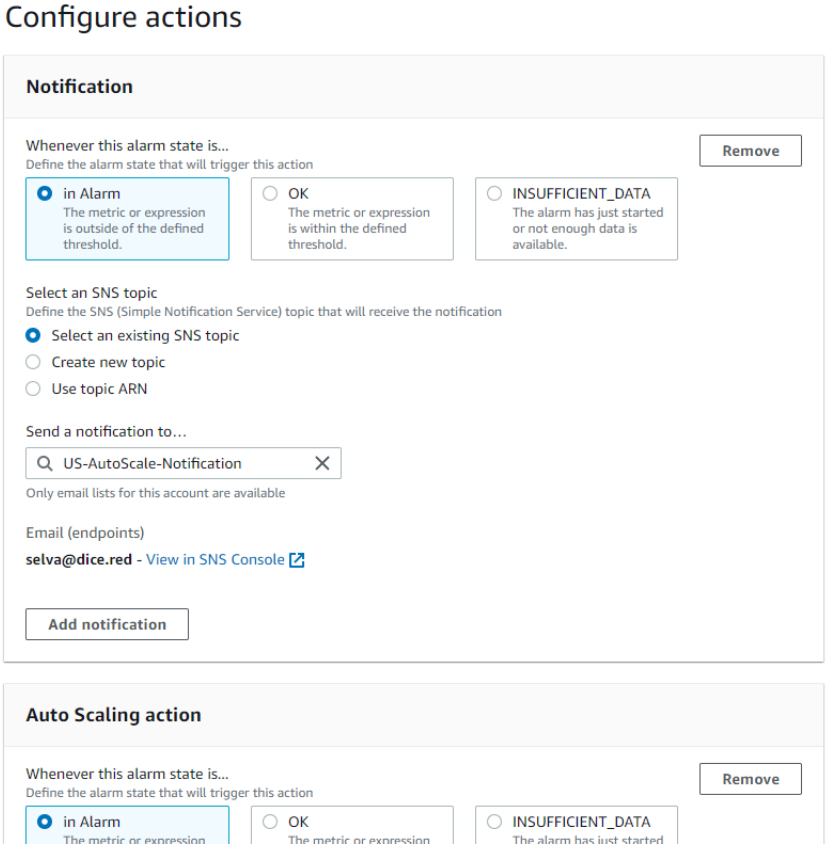
Configuring Load Balancer



Configuring Load Balancer

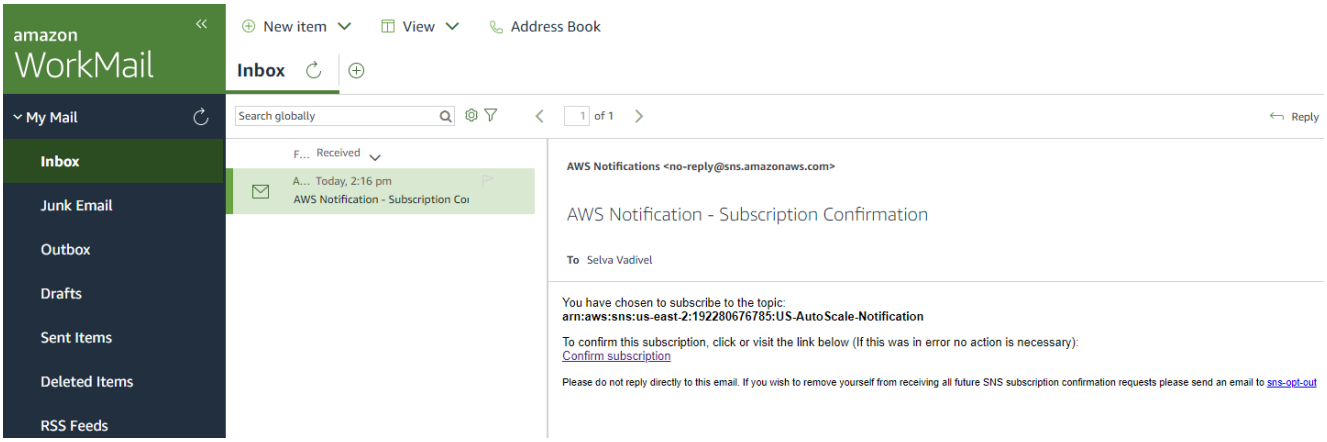


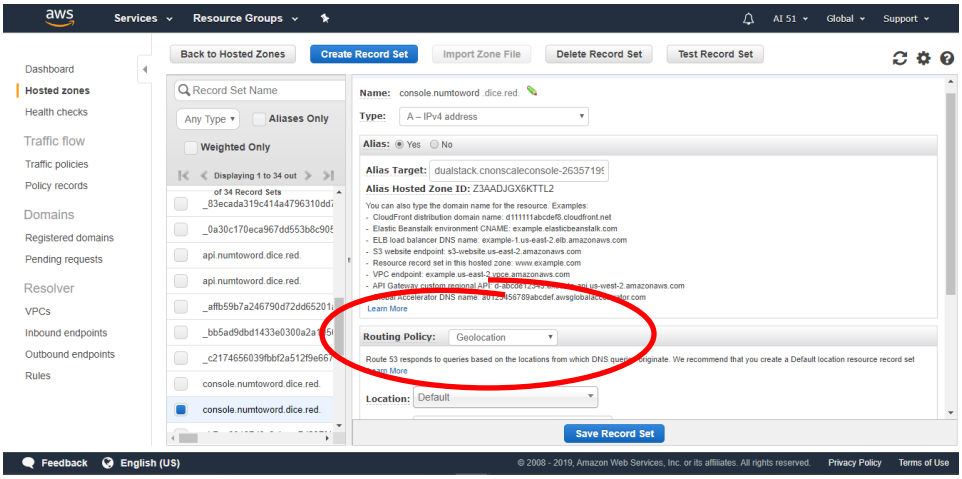
Configuring Load Balancer



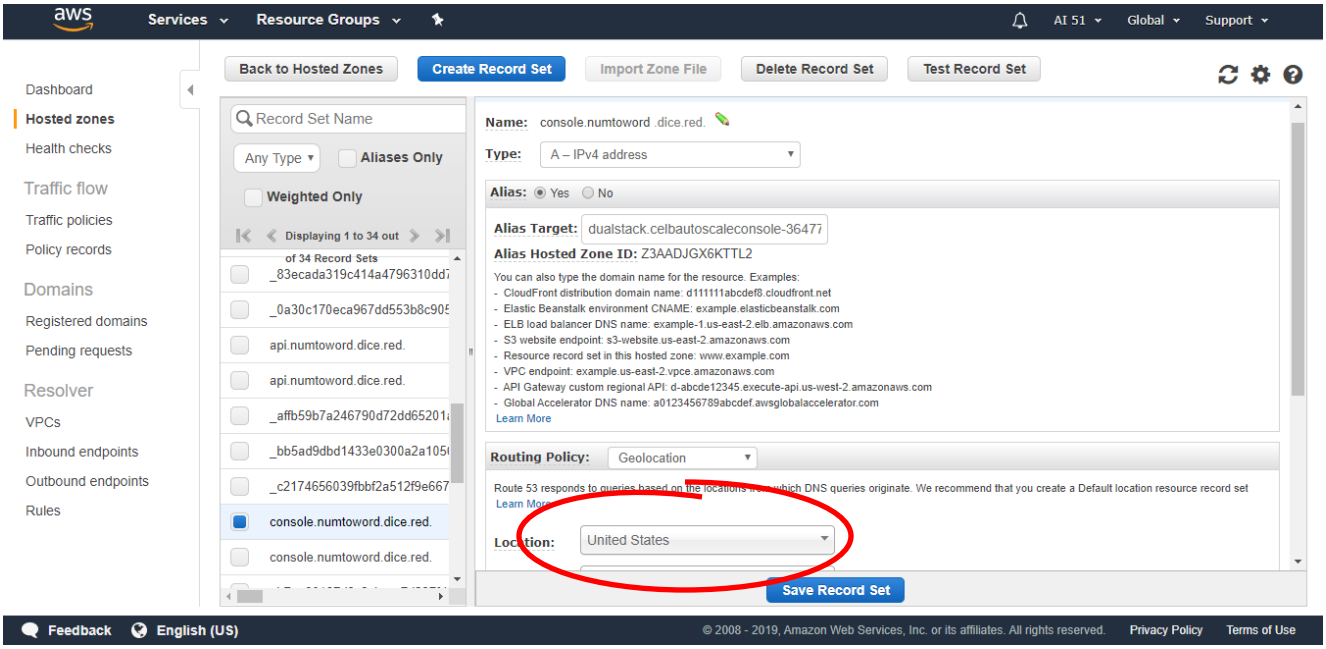
Setting Alarm at Cloud Watch

The above screenshot shows Alarm set at Cloud Watch to send email through SNS - topic and subscription is added for this Auto Scale. Confirm subscription for the confirmation email as below:





The above shows setting the URL to point to non-scaling load balancer for default users based on geolocation, it will include US users as well. It is best practice to auto-scale both the non-US and US users as it will not cost more.



The above shows setting the URL to point to auto-scaling load balancer for the US users based on geolocation.

COMMON TROUBLESHOOTING ISSUES

A Stackoverflow-like Issues

1. Goals not set when building in IDE like Eclipse.
   Solution: Set the goals to “clean install” and run Maven build.
2. No compiler is provided in the environment
   Solution:
   1. On your Eclipse IDE, go into Window > Preferences > Java > Installed JREs > and check your installed JREs. You should have an entry with a JDK there.
   2. Select the Execution Env as show below. Click OK
   3. Then Right-Click on your Project -> Maven -> Update Project
3. One or more files showing error
   Solution: Run a Maven clean and Maven -> Update Project
4. The REST-API GET endpoint provides valid JSON as response for any valid number. and it returns appropriate JSON for invalid requests as well. However, there may be cases where it will return a 404 error or no response for a very poorly constructed GET request. A REST-API endpoint is expected to be used programmatically, so further validations are expected to be provided by the program that will be consuming this end-point.
5. Missing custom jar files
   Solution: In case there is issue, then run maven install for pom.xml of numtowords-service-word first and then run mavan install for pom.xml of numtowords-api. It is because numtowrods-api uses the jar file of service. The console, numtowords-mvc-dash, is a separate project and is not part of the multi-module maven for api. For the console project, you can either export it as a WAR file or run maven build and that should generate the WAR file. Do not forget to update the URL in the application.properties file inside WAR file to point to the REST-API endpoint where you are running it. Also to use custom ports for the api, create a config folder in the same parent folder where the api jar is going to be run and then place the application.properties file inside the config folder, and then enter the service.port number that you would like to use. Both the api and console have application.properties file that you can customize to meet your requirements.
6. In case Instance ID is different from previously validated instance IDs, it is possible that the auto scaling group set a new instance. Also auto-scaling and

non-scaling groups share the same instance as load balancing is based on capacity utilization and is not on round robin.

7. When setting up the hostname for REST API application, the DNS of load balancer may resolve 404 as error because REST API end point is a variable. Solution: Set both 200 and 404 as valid response
8. Running Commands on Your Linux Instance at Launch:
   Auto scaling may not start the jar service on reboot. So include the following command when launching a new EC2 instance or for the image of AMI for help auto-scale restart the service for API endpoint when it tries to maintain the fleet of EC2 instances using the image it has in its configuration.

```
#!/bin/bash
cd /home/ubuntu/sonatype
java -jar numtowords-api-0.0.1-SNAPSHOT.jar
```

numtowords-api-0.0.1-SNAPSHOT is the name of the generated jar file for the REST-API. Please update the path as per your Linux instance. The above command is not needed for a WAR file because the Tomcat will install it.

9. Selenium testing in Eclipse IDE:

https://www.seleniumeasy.com/testng-tutorials/how-to-install-testng-step-by-step

APPENDIX

SELENIUM TEST RESULTS