**Deployment Process of Spring Boot Application**

**to AWS Elastic Beanstalk**

**Project Information**

|  |  |
| --- | --- |
| **Project Name** | **CSTECH Cloud Resource Management** |
| **Project ID** |  |
| **Release Date** | **October 28, 2022** |
| **Creation Date** | **October 27, 2022** |
| **Review Date** | **October 27, 2022** |
| **Author** | **SRIRAM NIDAMANURI** |
| **Reviewer** |  |
| **Approver** | **NIRANJAN KASHIKAR** |

**Document Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Modification Details** | **Author** |
| **October 27, 2022** | **v1.0** | **Created Deployment Process of Spring Boot App to AWS EBS** | **Sriram Nidamanuri** |
|  |  |  |  |

Table of Contents

1. [**Introduction**](#ebs1)**4**
2. [**Deployment Process5**](#ebs2)
   1. [Creating spring boot project5](#ebs3)
   2. [Creating Rest Controller6](#ebs4)
   3. [Testing application locally6](#ebs5)
   4. [Converting JAR file](#ebs6) 6
   5. [Opening Elastic Beanstalk in AWS](#ebs7) 7
   6. [Creating an application8](#ebs8)
   7. [Uploading the files9](#ebs9)
   8. [Changing the port number11](#ebs10)
   9. [Testing the application13](#ebs11)
   10. [Updating the spring boot project14](#ebs12)
3. [**Sign off/Approvals**](#ebs13) **15**

# 

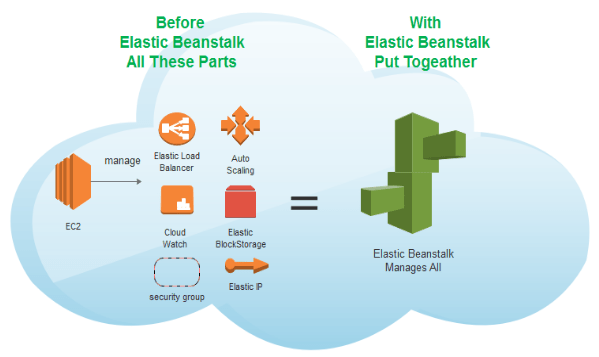
# Introduction

Elastic Beanstalk is a service that provides all the services on AWS that are required to run an application.

Elastic Beanstalk will create EC2 instance behind the scenes and it will install Java or Tomcat based on the application or platform one chosen and it will store jar file or war file on S3 server.

There will be one more component or one more service named as AWS Elastic Load Balancer, this will redirect our application on particular port.

If one doesn’t use Elastic Beanstalk service then they have to do all the configuration steps manually like creating ec2 instance and installing java or tomcat and creating S3 bucket service to store the jar or war file and to configure Elastic Load Balancer etc. separately. One has to do all these steps manually if they don’t use Elastic Beanstalk.

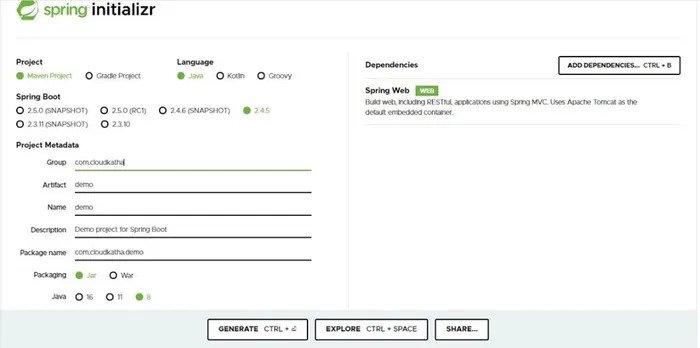


# [Deployment Process of Spring Boot App to AWS Elastic Beanstalk:](#ebs2)

1. Create a Spring Boot project.
2. Add a Rest Controller to be able to test.
3. Test the application locally.
4. Prepare the final jar file.
5. Login to AWS Management Console and open Elastic Beanstalk.
6. Create an Application in AWS Elastic Beanstalk.
7. Upload the JAR file to Elastic Beanstalk.
8. Change the port of the application.
9. Test Spring Boot Application Deployed on Elastic Beanstalk.
10. Update your Spring Boot app.

## [Creating a Spring Boot Project:](#_Creating_a_Spring)

Now all the modern IDE’s like STS or IntelliJ are providing services to create spring boot application in those IDE’s it self or else one can go to Spring Initializer using the link [**https://start.spring.io/**](https://start.spring.io/) to create a Spring Boot app and import it in their IDE’s.



## [Add a Rest Controller to the Spring Boot Application:](#_Add_a_Rest)

To run a spring boot application in web platforms one has to add a rest controller to the spring boot application.

package com.cloudkatha.demo.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class TestController {

@GetMapping("/")

public String health() {

return "Hello & Welcome to CloudKatha !!!";

}

}

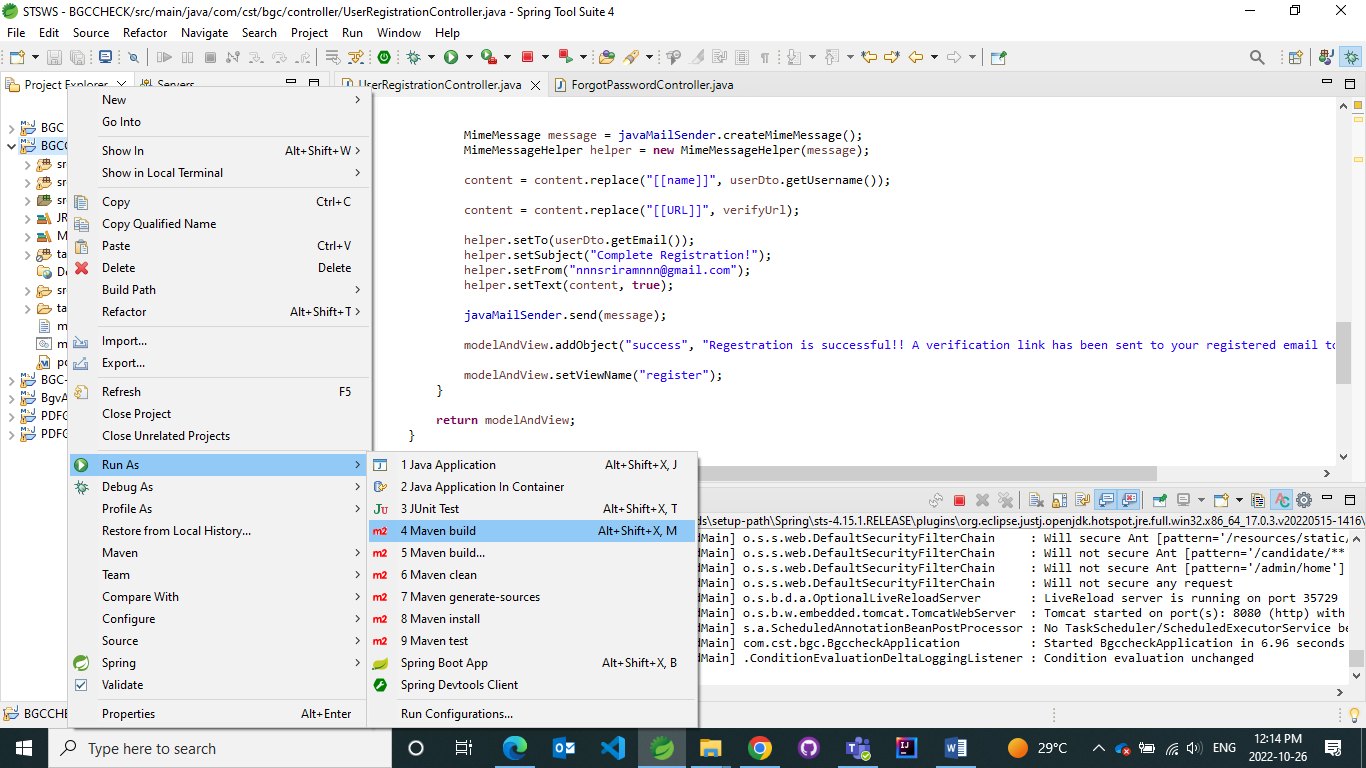
## [Testing the application locally:](#_Testing_the_application)

After adding the rest controller test the application locally to make sure that application is working fine.

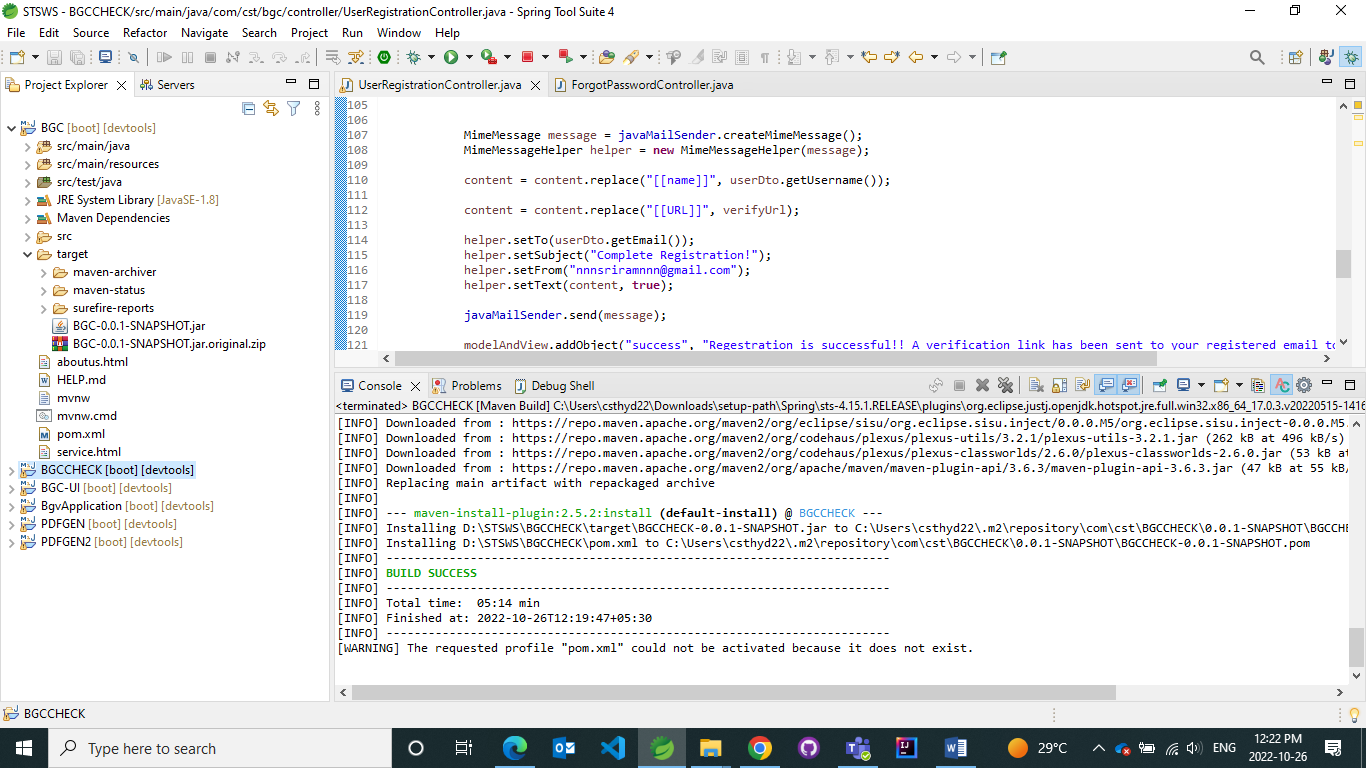


## Converting the Spring Boot Application to JAR File:

After making sure that the application is working fine, right click on the project and Run as Maven Build. It will ask to select goals, mention **clean install** there and click on apply and run the project.



After successfully creating the jar file a message will be shown as **Build Success** in the IDE console and a jar file will be created at the target folder in the project.

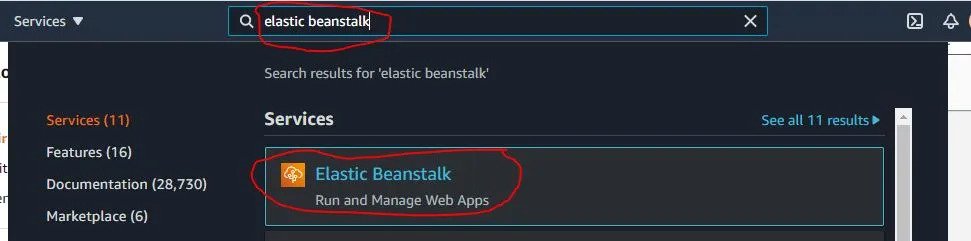


## [Login to AWS Management Console and open Elastic Beanstalk:](#_Login_to_AWS)

Go to the <https://aws.amazon.com/console/> URL and click on search option to search for Elastic Beanstalk.

You can either go to **Services -> Compute -> Elastic Beanstalk** or

Type **elastic beanstalk** in the search bar. Once you see the **Elastic Beanstalk** option, click on that.



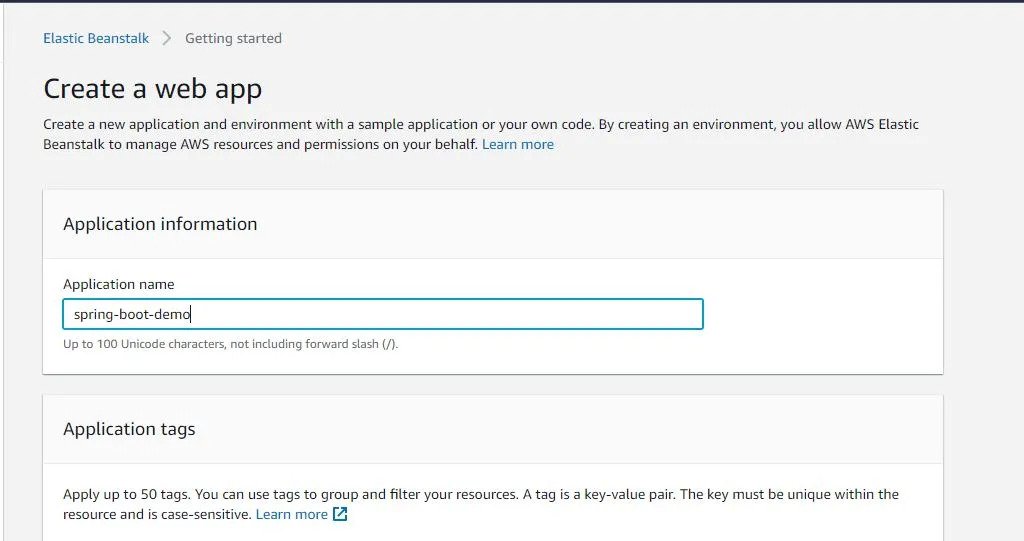
## [Create an application in Elastic Beanstalk:](#_Create_an_application)

An application is the top-level container in Elastic Beanstalk that contains one or more application environments (for example dev, qa and prod etc).

Click on **Create Application.**



Provide an Application name. You can also provide a tag if you want.

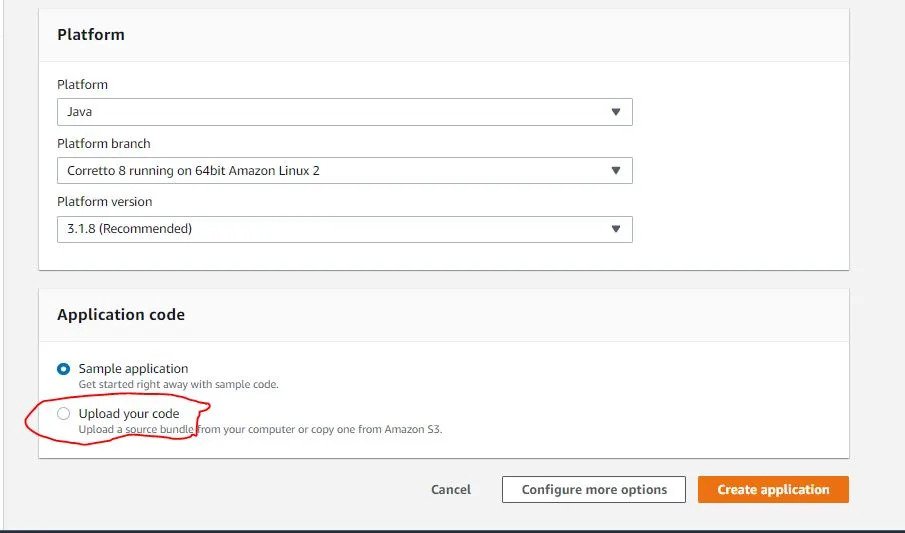


After giving application name scroll down and provide below information in platform section

Platform : Java

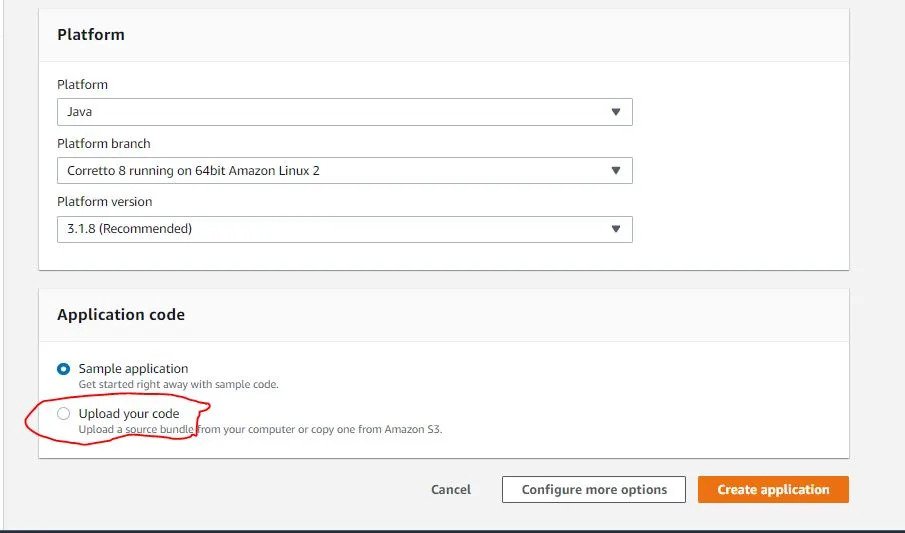
Platform branch: Corretto 8

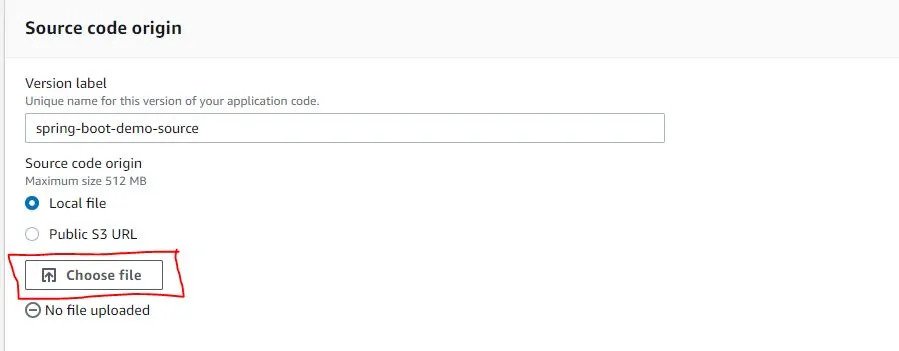
Platform version: Best to choose recommended one



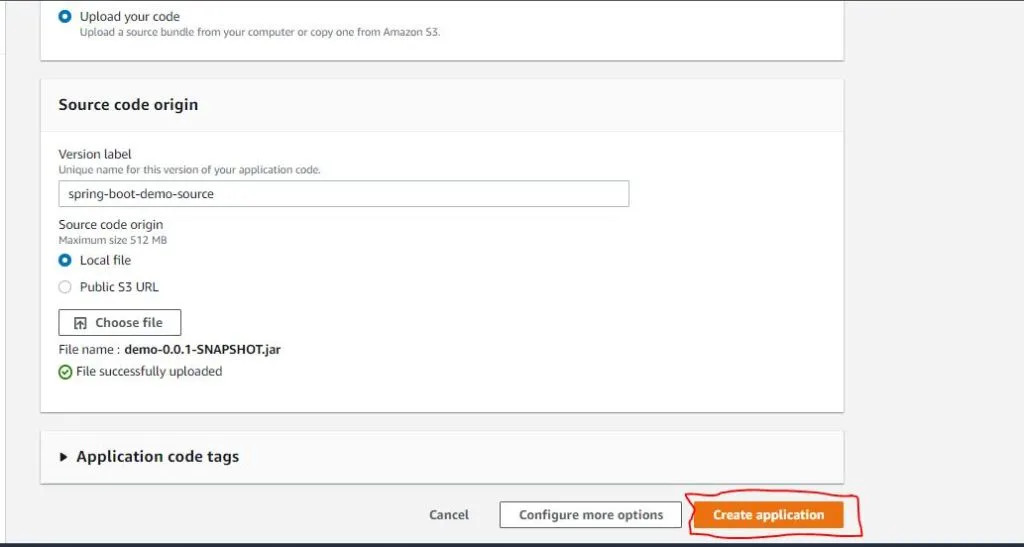
## [Upload the JAR file to Elastic Beanstalk:](#_Upload_the_JAR)

After giving platform information, click on the Upload your code option after that click on **Choose file** and upload the jar file we created. The jar file will be presented in the target folder in spring boot project.

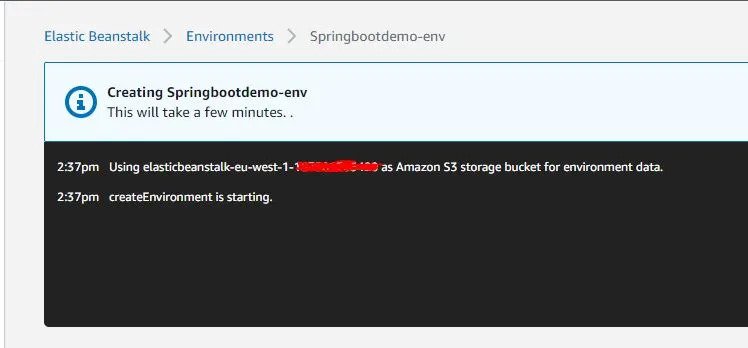




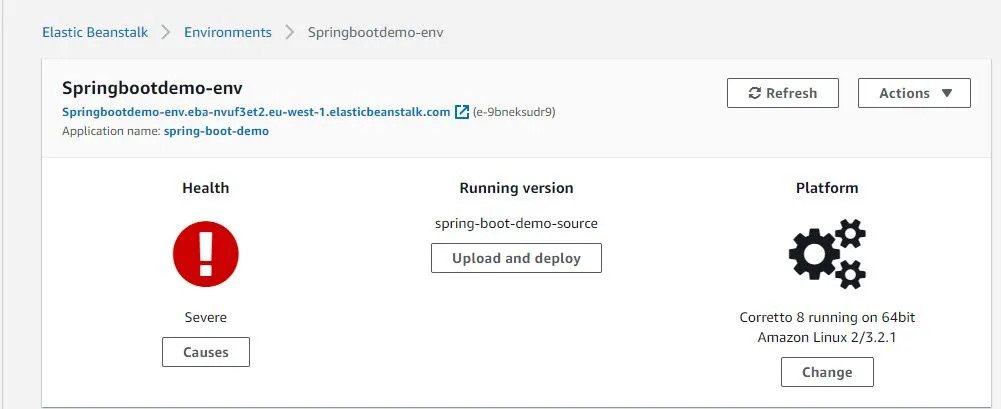
Once the jar is uploaded successfully, Click on **Create application** as shown in the below screenshot.



As soon as you click **Create application**, Elastic Beanstalk starts setting up your application environment. You get to see the log of what’s happening behind the scene on the dashboard like below.



It will take a few minutes for Elastic Beanstalk to provision resources to set up the complete environment. Meanwhile, you can try to understand log on what’s really happening. Once setup is done, you will see the below screen.



Health will give information related to how app is going to work on cloud platform.

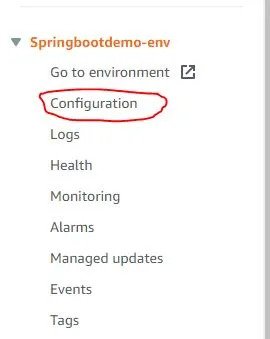
## [Change the PORT number of the Application:](#_Change_the_PORT)

There are quite a few ways of doing this like-

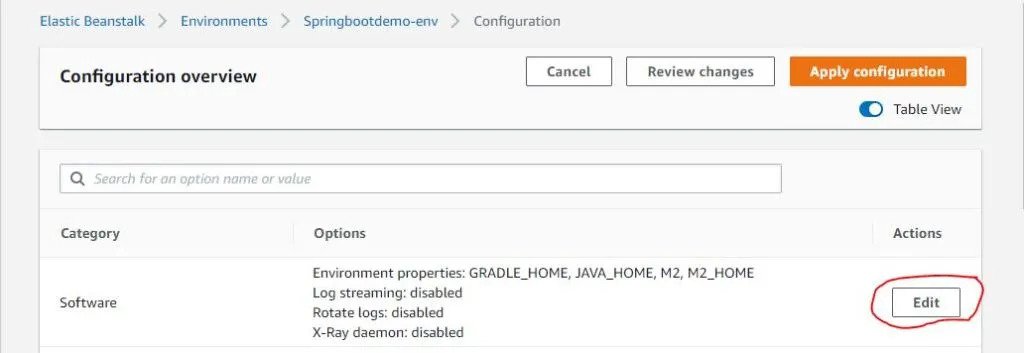
1. Changer **server.port** to 5000 in application.properties file
2. Use Elastic Beanstalk Environment variable to provide SERVER\_PORT

Going with the first option will require changing the application code, building and redeploying. So, let’s go ahead with 2nd option. Let’s change the environment variable to make our spring boot application listen on 5000.

On left-hand side project menu, Click on **Configuration**

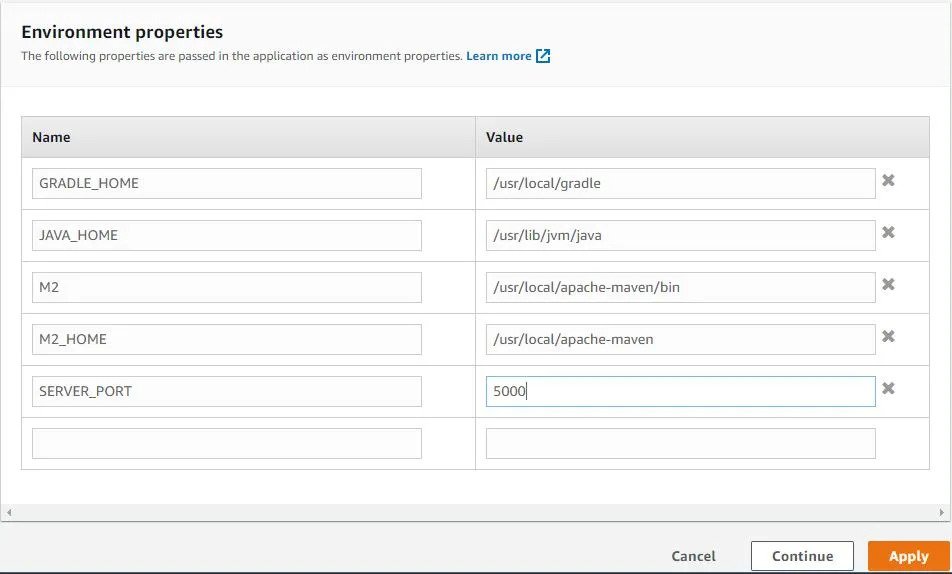


There are various categories displayed here. Environment variables belong to the Software category as you can see below. So, let’s click the edit **Software** category.



Click **Edit** and scroll down to the **Environment properties** section

Add a property with the name **SERVER\_PORT** and **Value** 5000.

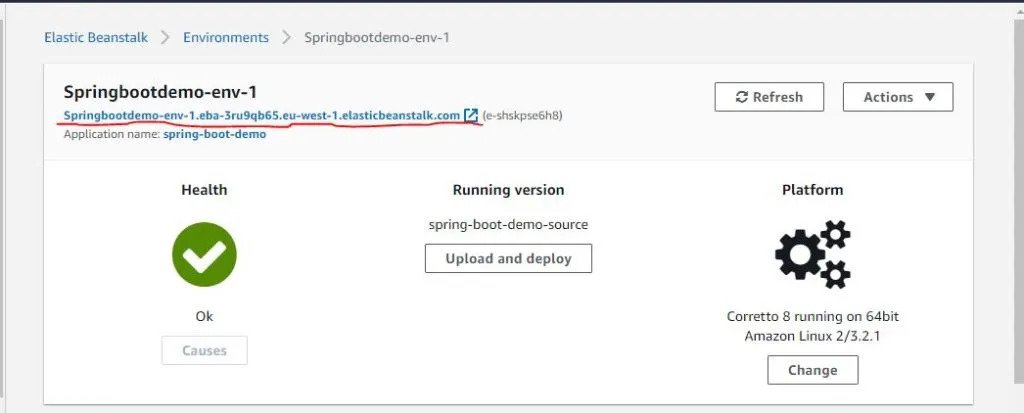


Click **Apply**

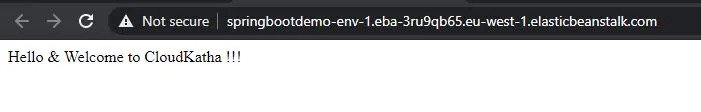
Wait for some time as you see your environment is getting updated. Behind the scene, the application gets restarted and picks up the latest environment variables. Once done, you will see application health updated as green.

## [Testing Spring Boot Application that is deployed in the Elastic Beanstalk:](#_Testing_Spring_Boot)

Click on the application endpoint (Underlined with red on the below screenshot) to check rest controller end point result.



Endpoint result can be seen like mentioned in the below screenshot.



## [Update Spring Boot App:](#_Update_Spring_Boot)

Till now a basic application is created for demo purpose, one can update the app according to their requirements and redeploy to AWS Elastic Beanstalk by following the above procedure.

# 

# [Sign Off / Approvals](#_Sign_Off_/):

|  |  |  |  |
| --- | --- | --- | --- |
| Approver Name | Approver Role | Approved (Y/N) | Date Approved  (MM/DD/YYYY) |
| NIRANJAN KASHIKAR | QA AUTOMATION-LEAD | Y | 10/28/2028 |
|  |  |  |  |