**A picture containing drawing

Description automatically generated**

Spring Boot Microservice component deployment on AWS Elastic Beanstalk

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr no** | **Version** | **Author** | **Date** | **Reviewed by** | **Comments** |
| 1 | 1.0 | jagadish.sanjeevi |  | manoj.kumar.boyeena | Initial draft |
|  |  |  |  |  |  |

MICROSERVICES SPRING BOOT EXPERTS

Table of Contents

1. [Agenda 3](#Agenda)
2. [Why AWS Elastic Beanstalk](#WhyAWSElasticBeanstalk) [3](#Processtofollow" \o "How to Enable Virtualization)
3. [Benefits of using EBS](#BenefitsofusingEBS) [4](#Processtofollow" \o "How to Enable Virtualization)
4. [Process to follow](#Processtofollow) [5](#Processtofollow" \o "How to Enable Virtualization)
5. [Application creation](#ApplicationCreation) [5](#Processtofollow" \o "How to Enable Virtualization)
6. [EBS Deployment](#EBSdeployment) [6](#Processtofollow" \o "How to Enable Virtualization)
7. [Running multiple deployments with Load balancer](#RunningMultipledeploymentswithLoad)  [9](#Processtofollow" \o "How to Enable Virtualization)

**Agenda**

Deploy a Spring boot microservice component on AWS environment using the AWS Elastic Beanstalk [EBS] deployment service.

Modify the configurations of EBS to build an ECO system around the component like provisioning multiple EC2 Instances on demand and resource utilization.

Provisioning a Load balancer, Auto Scaling groups etc.

**Why AWS Elastic Beanstalk?**

When it comes to cloud infrastructure providers, AWS has always been the market leader and their services make it fast and easy to deploy a cloud application.

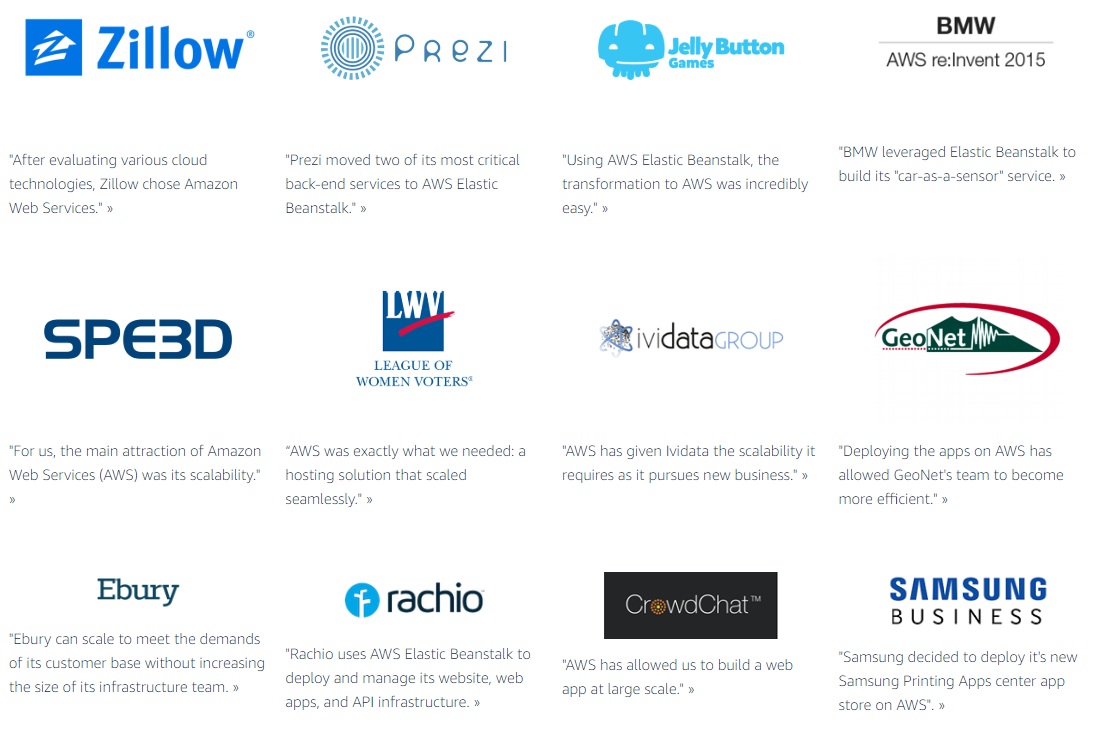
Elastic Beanstalk is a deployment and provisioning service that allows you to deploy an entire infrastructure with a few clicks by automating the process of getting applications set up on AWS infrastructure.

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

There is no additional charge for Elastic Beanstalk - you pay only for the AWS resources needed to store and run your applications.

**Featured AWS customers using EBS**



**Benefits of using EBS**

1. **Fast and simple to begin**

Elastic Beanstalk is the fastest and simplest way to deploy your application on AWS. You simply use the AWS Management Console, to upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Within minutes, your application will be ready to use without any infrastructure or resource configuration work on your part.

1. **Developer productivity**

Elastic Beanstalk provisions and operates the infrastructure and manages the application stack (platform) for you, so you don't have to spend the time or develop the expertise. It will also keep the underlying platform running your application up-to-date with the latest patches and updates. Instead, you can focus on writing code rather than spending time managing and configuring servers, databases, load balancers, firewalls, and networks.

1. **Impossible to outgrow**

Elastic Beanstalk automatically scales your application up and down based on your application's specific need using easily adjustable Auto Scaling settings. For example, you can use CPU utilization metrics to trigger Auto Scaling actions. With Elastic Beanstalk, your application can handle peaks in workload or traffic while minimizing your costs.

1. **Complete resource control**

You have the freedom to select the AWS resources, such as Amazon EC2 instance type, that are optimal for your application. Additionally, Elastic Beanstalk lets you "open the hood" and retain full control over the AWS resources powering your application. If you decide you want to take over some (or all) of the elements of your infrastructure, you can do so seamlessly by using Elastic Beanstalk's management capabilities.

**Process to follow**

Below are the step by step process to follow for deploying the Spring boot micro service component on AWS EBS environment and configure the settings to build the ECO system around the component.

**Application Creation**

First let’s create a simple Spring boot microservice component with CRUD operation.

To make it simple we are not using any database in the DAO layer. The DAO layer is a stub, basically it’s a in memory DAO.

Here i created the Country-Population-Registration microservice component which has rest end points to perform:

Registration of a new country.

Deletion of existing country registration details.

Update the existing country registration details.

Retrieving country registration details with registration number.

Retrieving all the country registration details.

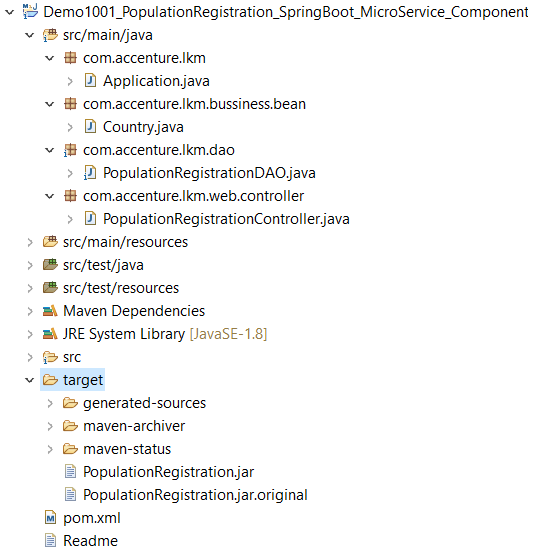
Refer the Demo “**Demo1001\_PopulationRegistration\_SpringBoot\_MicroService\_Component**”.

Import the application to eclipse and observe the code and start the deployment.

Using postman client check all the Rest end points. [If required refer to “**Readme.txt**” file under the root folder in the Demofor execution stepswith end point URL’s and sample JSON data for Post and Put request.]

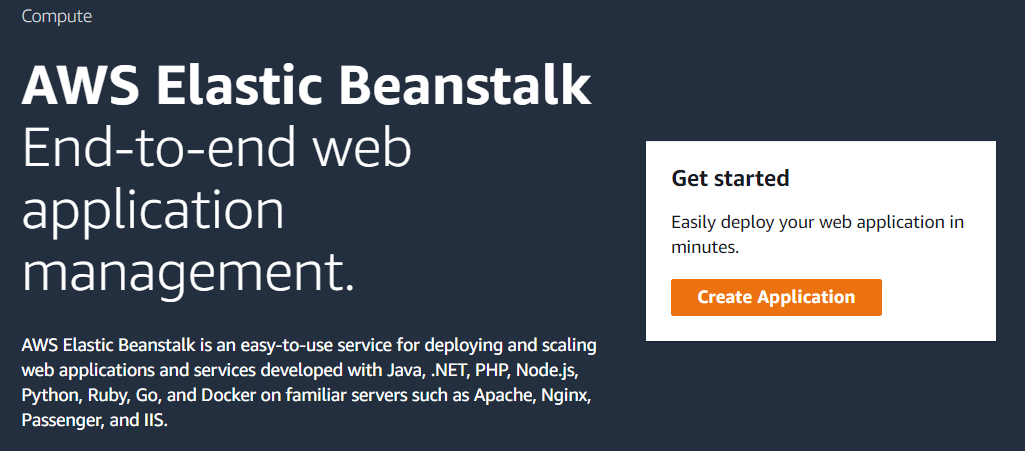
Clean and build the Fat jar using mvn clean package goal.

**Project Structure**

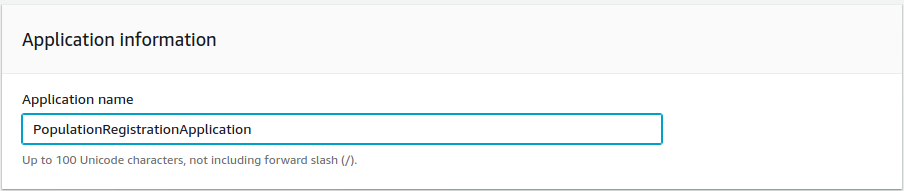


**EBS deployment**

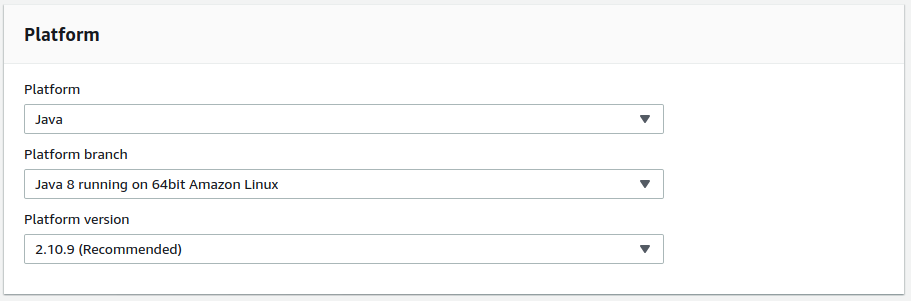
1. To start application deployment on AWS, first we need an AWS account. You can use your existing account or else create your own free tier account <https://aws.amazon.com/free>.
2. Login to the account.
3. From the AWS management console click and open Elastic Beanstalk.



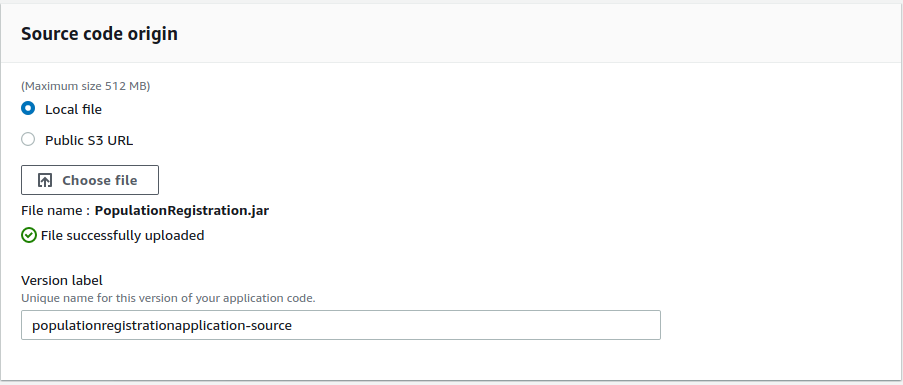
1. Click on create application, provide the application name [Ex: PopulationRegistrationApplication].

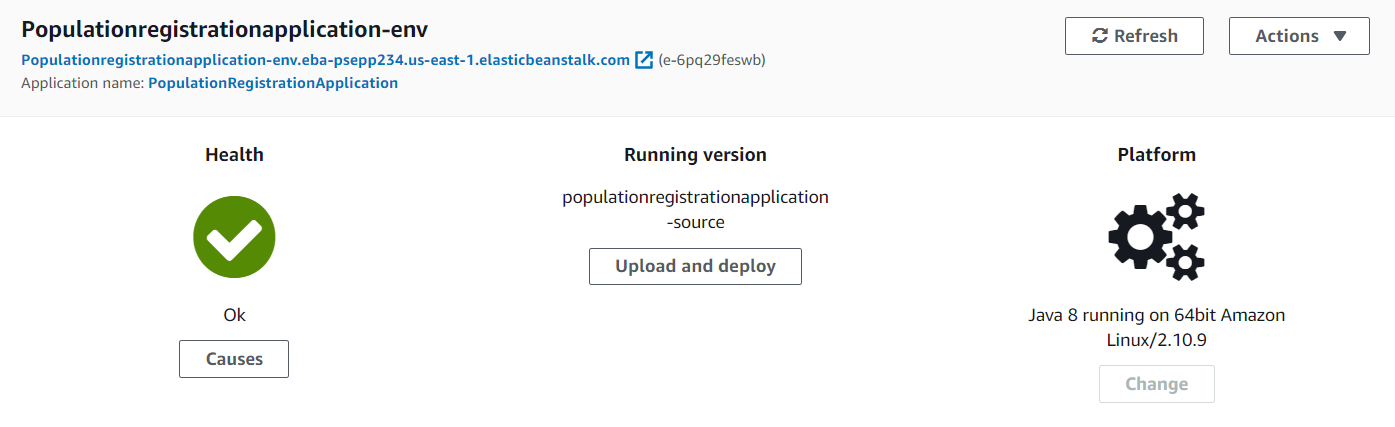


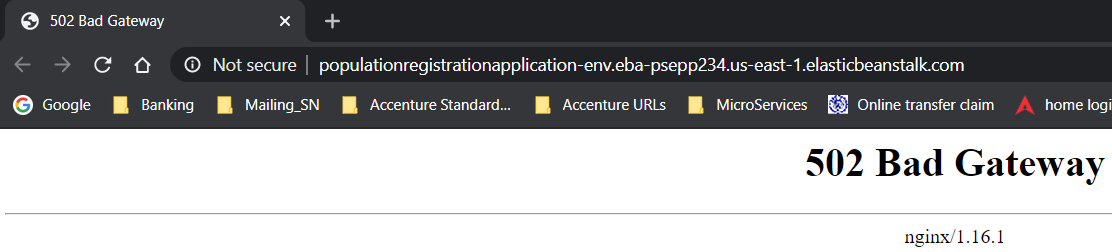
1. Under platform choose Java, branch as Java8, platform version choose recommended version as shown below.



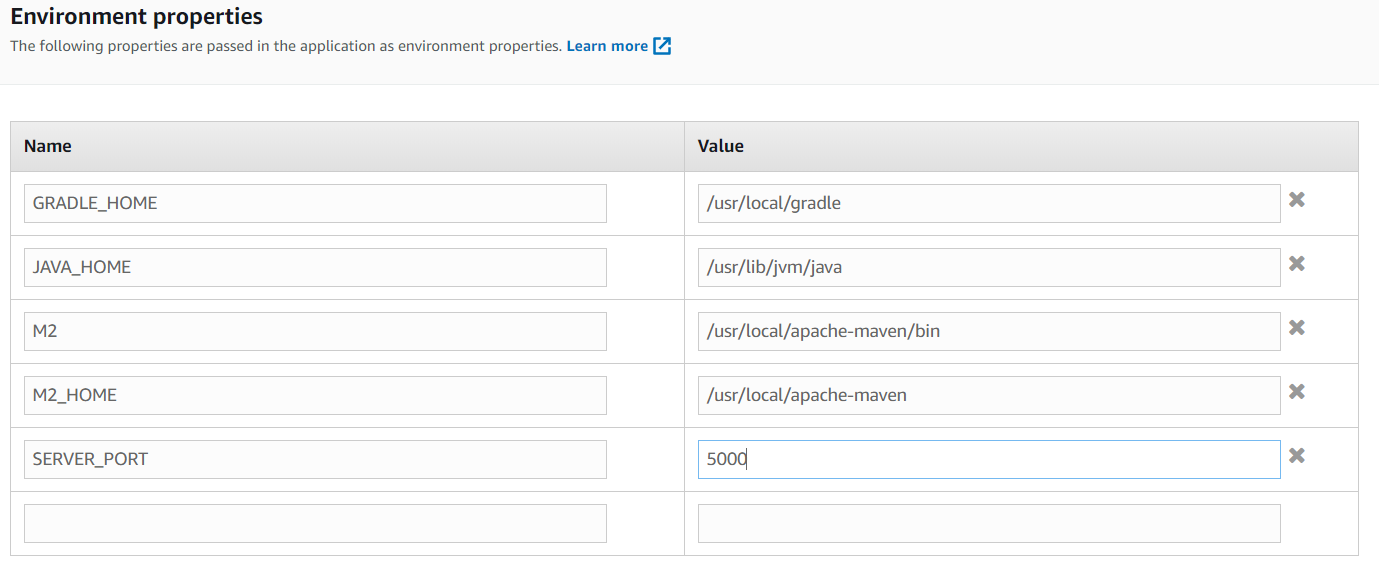
1. Under source code origin, select local file click choose file and browse and upload the Jar file.



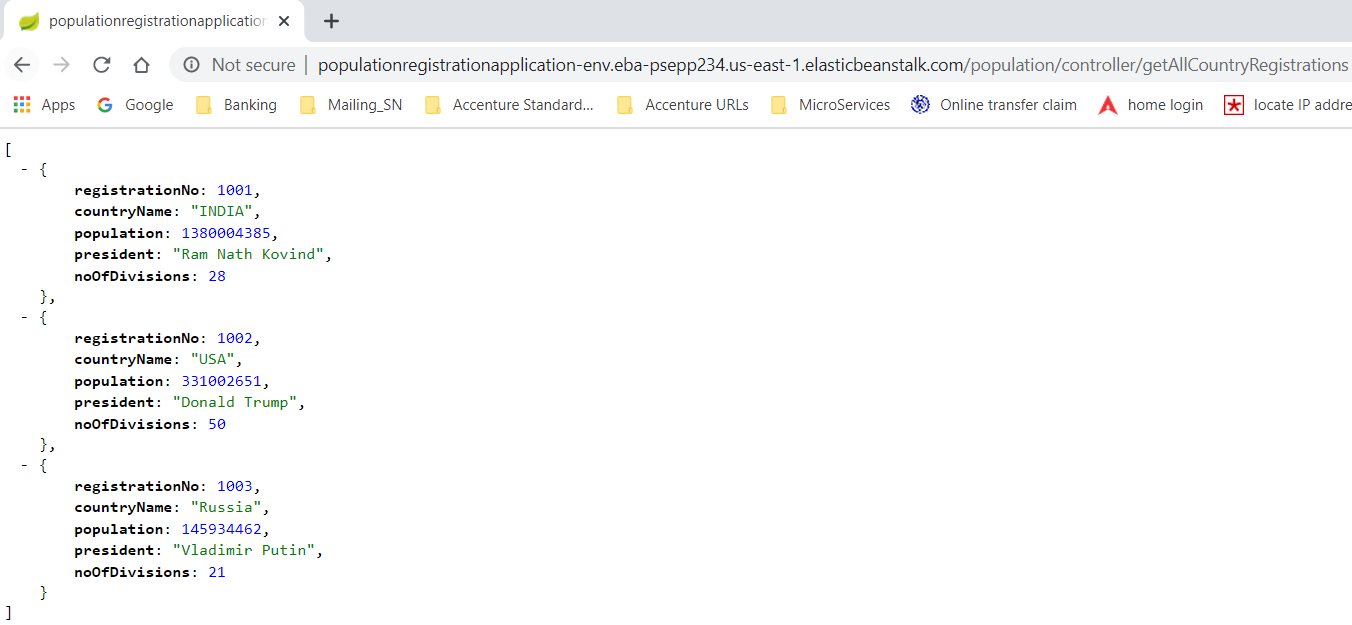
1. Click on create application and wait till everything is setup. Observe the logs, it will create a Security group, Elastic IP, EC2 instance, S3 etc.
2. Once everything is done you will see. 
3. Access the application by clicking on the link highlighted in red above.



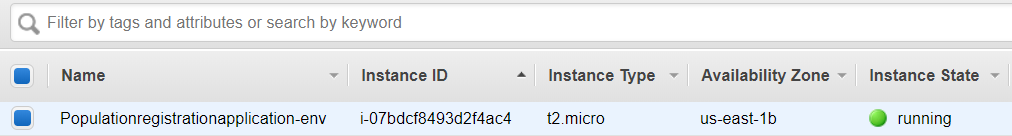
1. Right now, you get a 502-error response, which is expected. By default, Elastic Beanstalk runs the application on port 5000, while Spring Boot executes on port 8080. Let’s direct your app run on port 5000 to fix the issue.
2. Click on Configuration the dashboard menu, Edit software configurations, scroll down to Environment properties and add SERVER\_PORT : 5000 as shown below. Click on apply and wait for operation to complete.

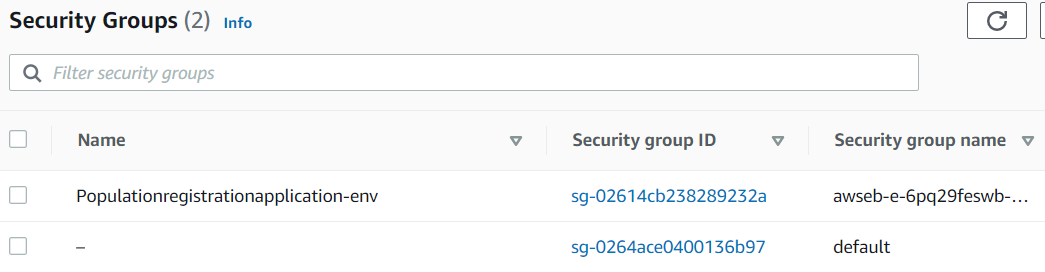


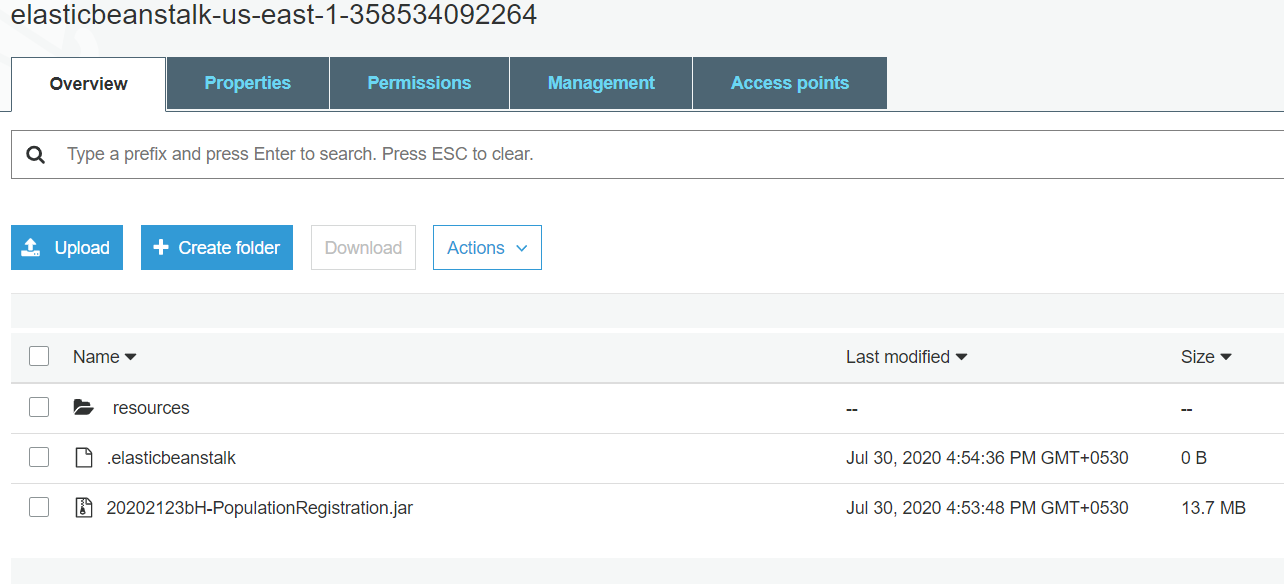
1. Go back and access the application with the URL provided, append the rest end point to the URL you can see the application is accessible.



1. That’s all deploying the component through Beanstalk is done. Observe the EC2 Instance provisioned and running, Security Group created, S3 bucket etc.







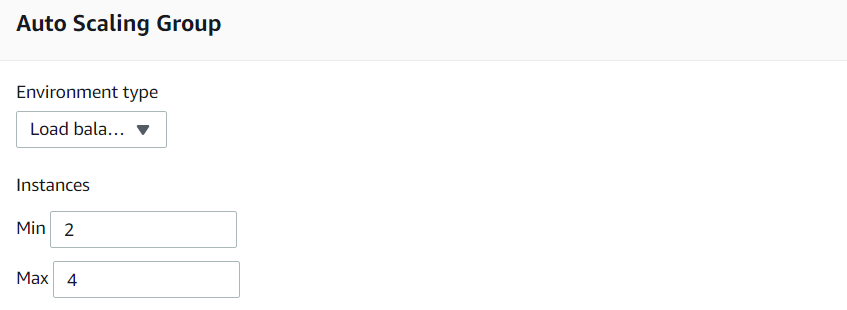
You can observe one EC2 instance is running with application deployment, one security group created, one S3 bucket where the Jar file is stored. So as said complete infrastructure is provided without, we are managing it.

**Running Multiple deployments with Load balancer**

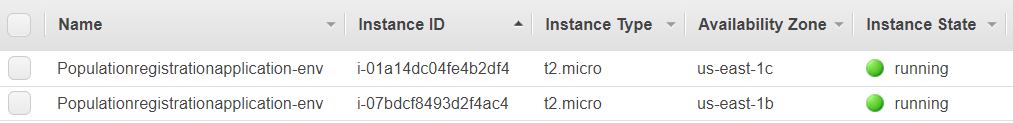
1. Suppose in EBS environment if we want multiple EC2 to run in parallel with load balancer we can do it just by just changing the configurations.
2. It will automatically provide the load balancer etc. stuff.

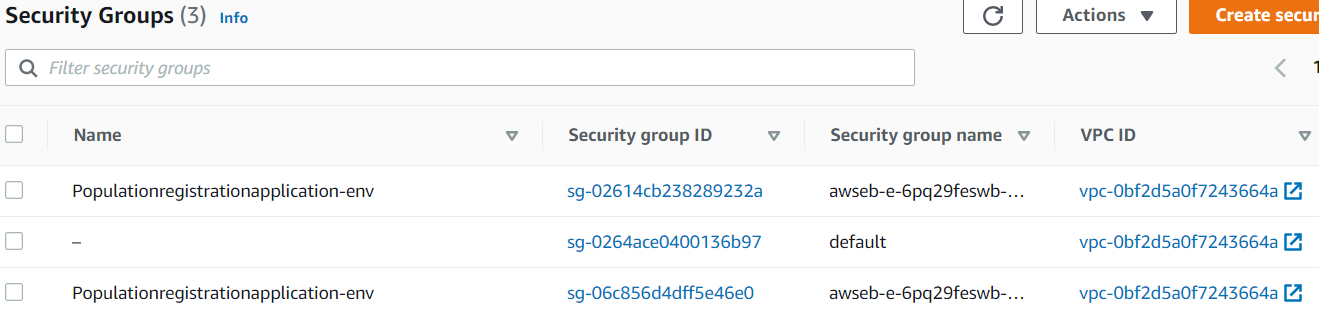
**Let’s begin**

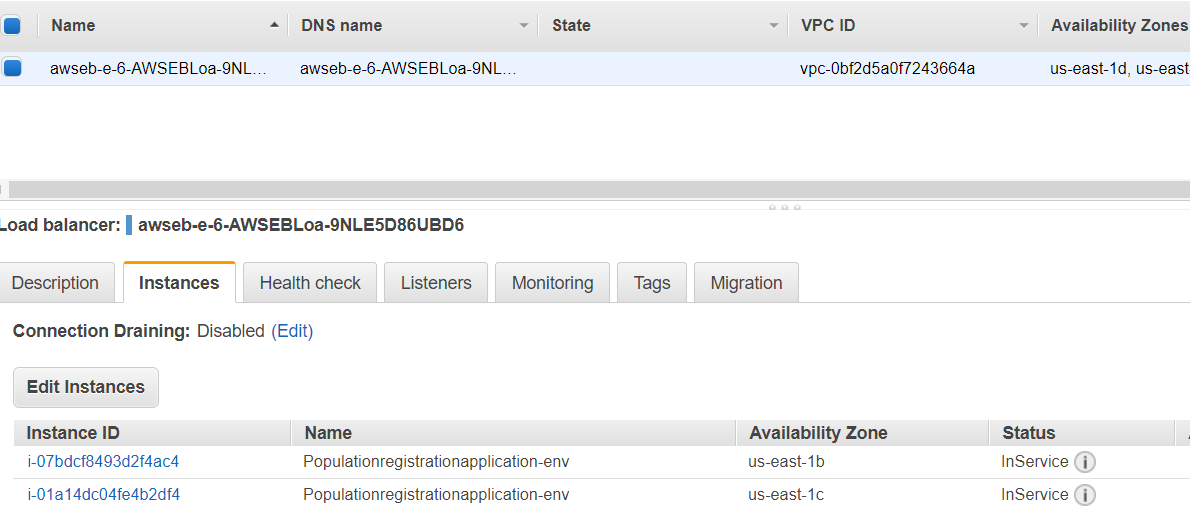
1. Click on Configuration the dashboard menu, edit capacity, change environment type to load balanced, min-2 , max-4, leave every thing else default. Click on apply. On any warning click confirm. Wait for the operation to complete.

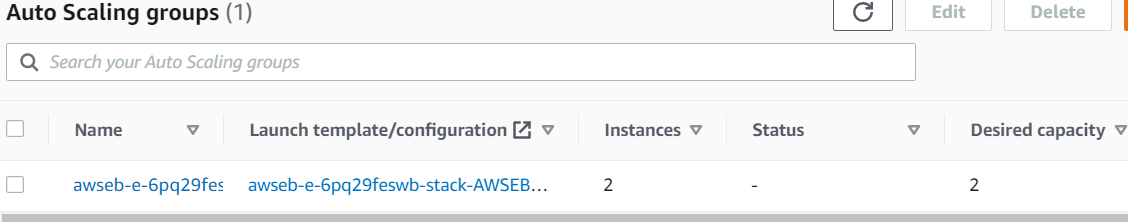


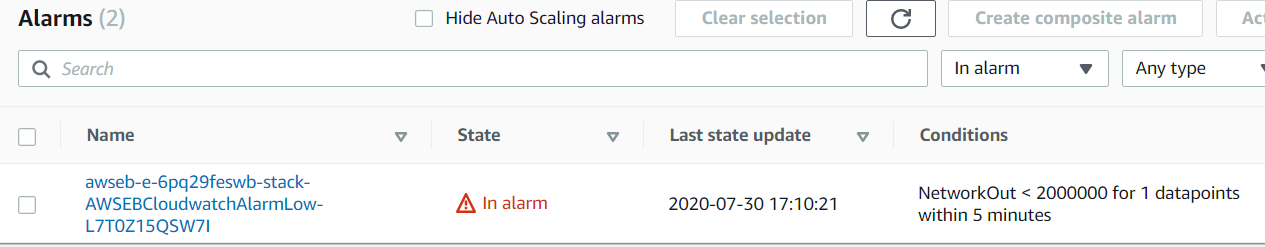
1. Go and observe min 2 EC2 Instance will be running, a new classic load balancer is created with 2 EC2 instances as targets, a new auto scaling group is created, a new security group is created, a new cloud watch alarm as shown below in screen shots.



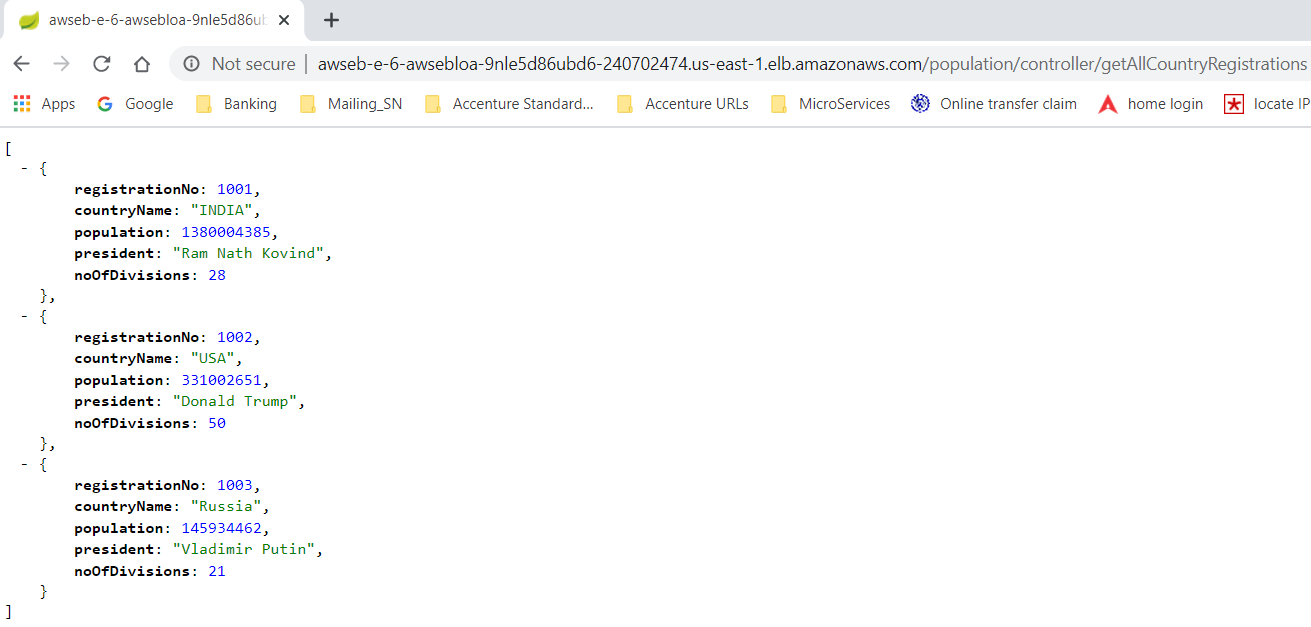








1. Pick the DNS of the load balancer append the rest end point URI and access the application.



1. That’s all we are done with Multiple deployments and load balancer.