Blue/Green Deployment of Node.js Application

-Sriganesh Pera MS in CS -University of Missouri-Columbia

Project: Blue/Green Deploying a Node.js Application with DynamoDB to Elastic Beanstalk

Blue/Green Deployments: Blue/green deployment is a technique for releasing applications by shifting traffic between two identical environments running different versions of the application. Blue/green deployments can mitigate common risks associated with deploying software, such as downtime and rollback capability.

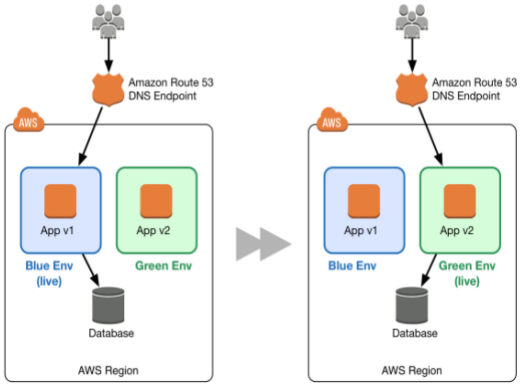


Figure 1: Basic blue/green example

The blue environment represents the current application version serving production traffic. In parallel, the green environment is staged running a different version of your application. After the green environment is ready and tested, production traffic is redirected from blue to green.

ELASTIC BEANSTALK:

Traditionally, deploying a web application on AWS requires provisioning following services: EC2 ELB, Auto scaling ,VPC and IAM. Elastic Beanstalk removes the need to manually build an infrastructure for the developer and makes it possible to quickly deploy and manage web applications of any scale.

Developers just need to upload and deploy code and configuring AWS services is taken care of.

Elastic Beanstalk supports applications developed in Java, PHP, .NET, Node.js, Python, and Ruby, as well as different container types for each language.

Lets do a simple lab to see how Elastic Beanstalk helps us provision all the infrastructure we need:

Lab 1: Deploying a Node.js Application with DynamoDB to Elastic Beanstalk

Step 1: Download source code for the project

Download the zip file following the link given below

https://github.com/perasr/node-ebs.git

Step 2: Install Unixutils and set the path and check it in command prompt by typing zip

(i) Download Unixutils:

https://sourceforge.net/projects/unxutils/

(ii) Copy the folder in C:\Program Files (x86).

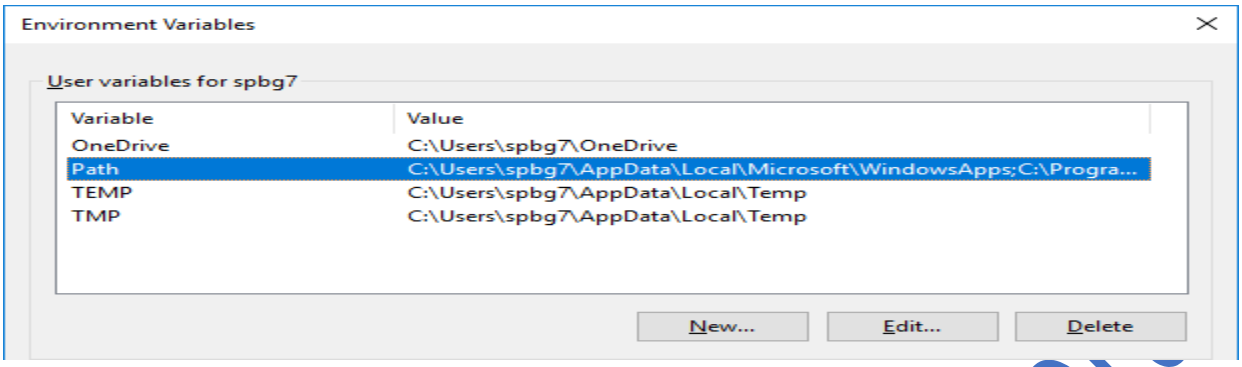
and extract it and save it there.Now copy the path

`C:\Program Files (x86)\UnxUtils\usr\local\wbin`.

(iii) Add it in Path variables

Now go to control panel > system and security > advanced system settings

Now a window pops out ,now click on advanced > environment variables and click on Path and edit

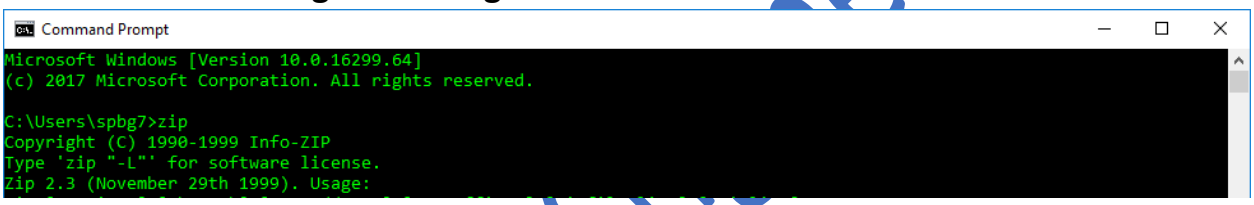


Now click on edit again and paste the path you copied :

```
C:\Program Files (x86)\UnxUtils\usr\local\wbin.
```

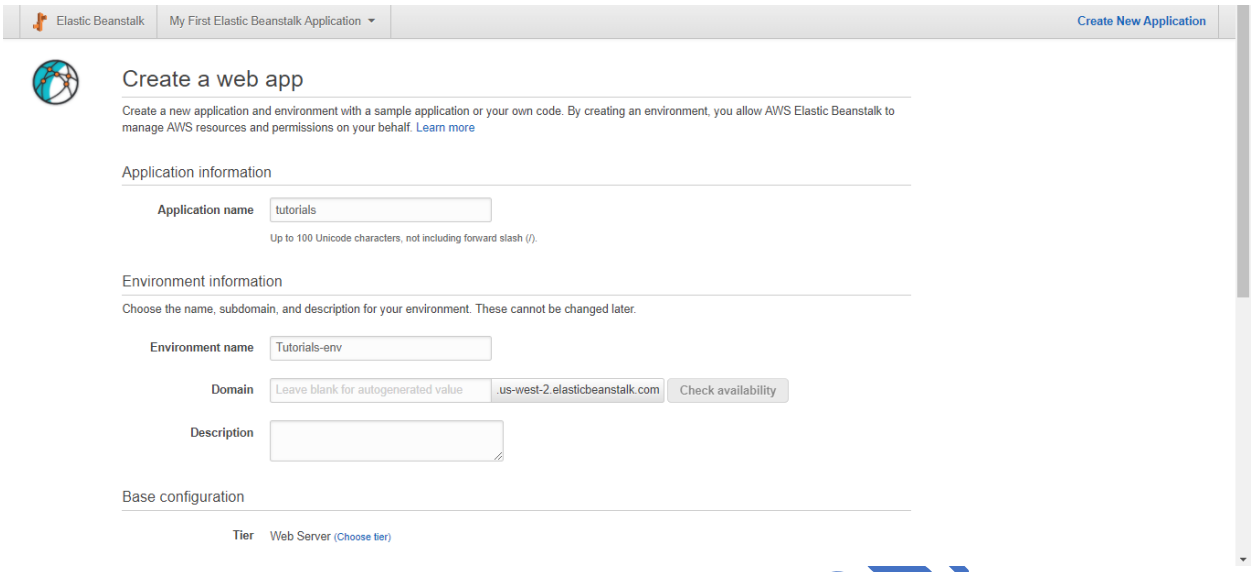
(iv) Now open command prompt in windows and type zip

You should be seeing something like this:



Step 3: Launch the environment using this pre-configured link

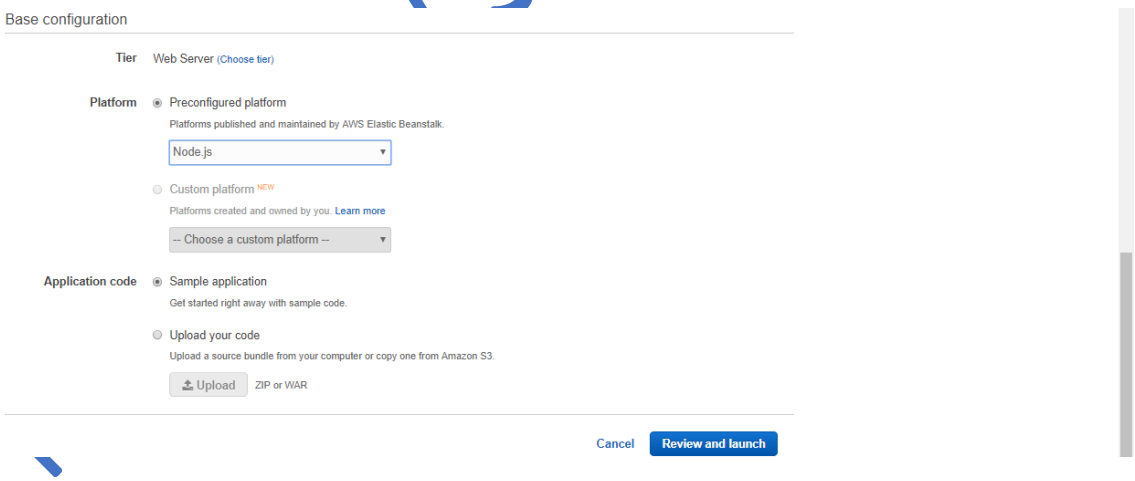
console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced



<u>And under the base configuration:</u>

Chose **platform** as Node.js as we gonna launch our app using Node.js

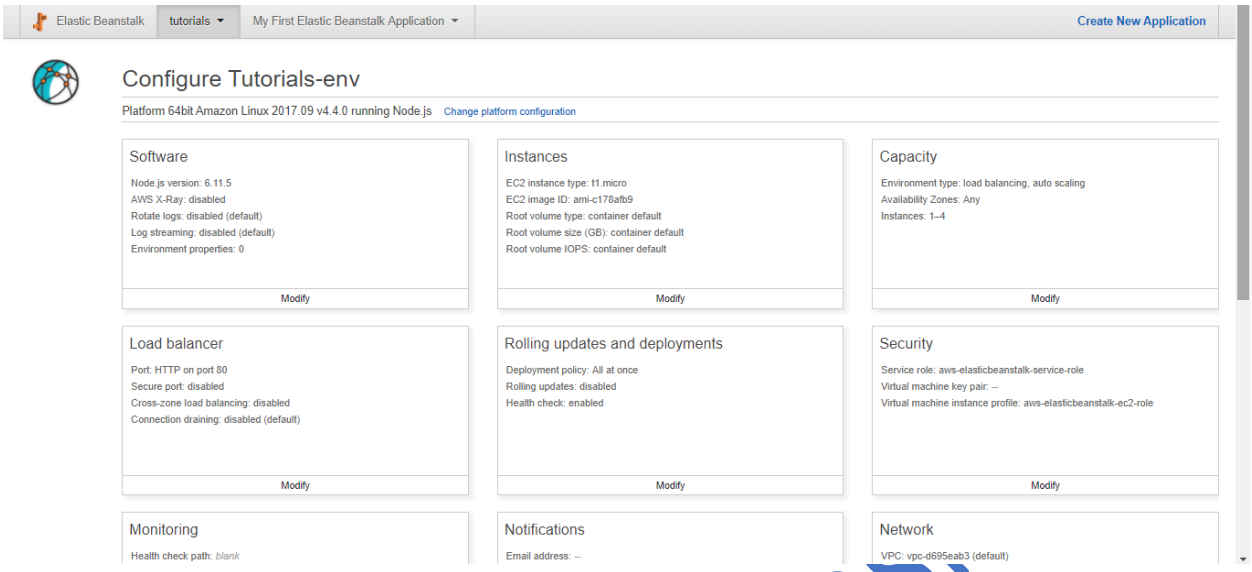
And for **application code** chose sample because before we upload and deploy our code we gonna make some few tweaks which is why we are using sample appliction this time.



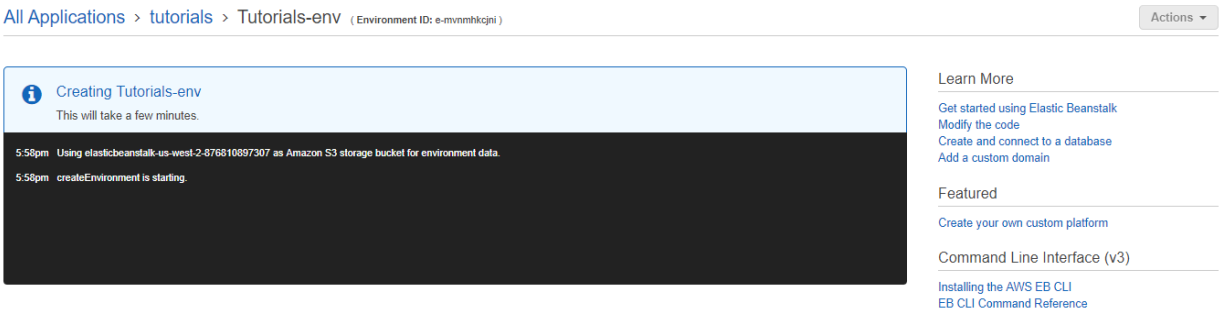
Now click on **<u>'Review and Launch'.</u>**

<u>Now you will see configuration page where all the infrastructure is provisoned :</u>

Ec2 instances,Auto scaling group,security group,Load Balancers,VPC and even DynamoDB table.



Now click on create app.You'll find that our app is being created.It will take several minutes.



<u>Elastic Beanstalk takes about five minutes to create the environment with the following resources:</u>

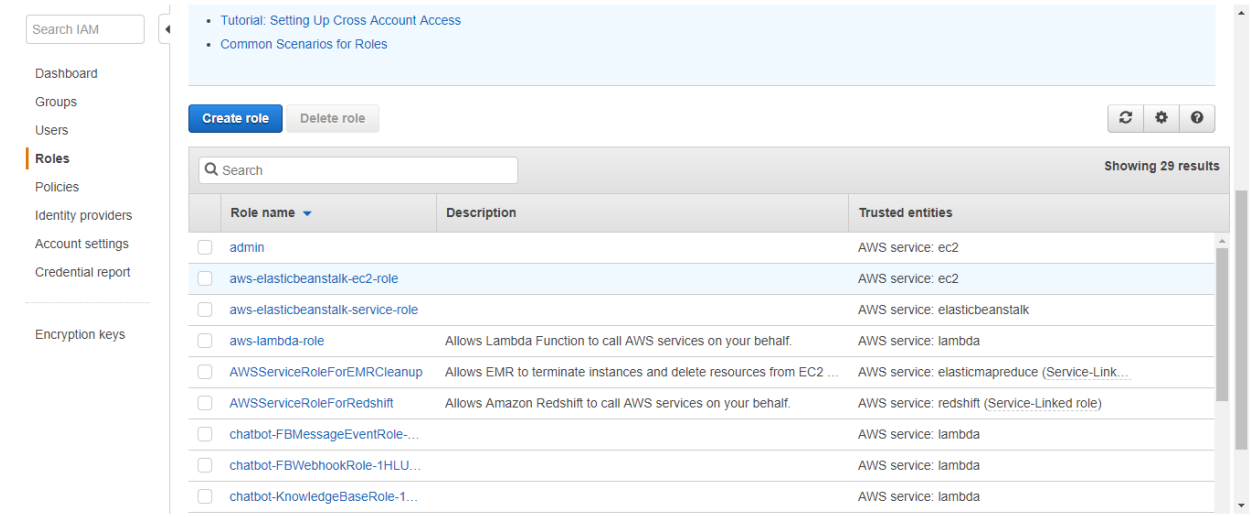
EC2 instance - Instance security group – Load balancer – Auto Scaling group Amazon S3 bucket – Amazon CloudWatch alarms – AWS CloudFormation stack –

<u>Step 4: Now Add Permissions to Your Environment's Instances using IAM roles.</u>

When the app receives a request that requires it to use AWS services, the application uses the permissions of the instance it runs on to access those services.

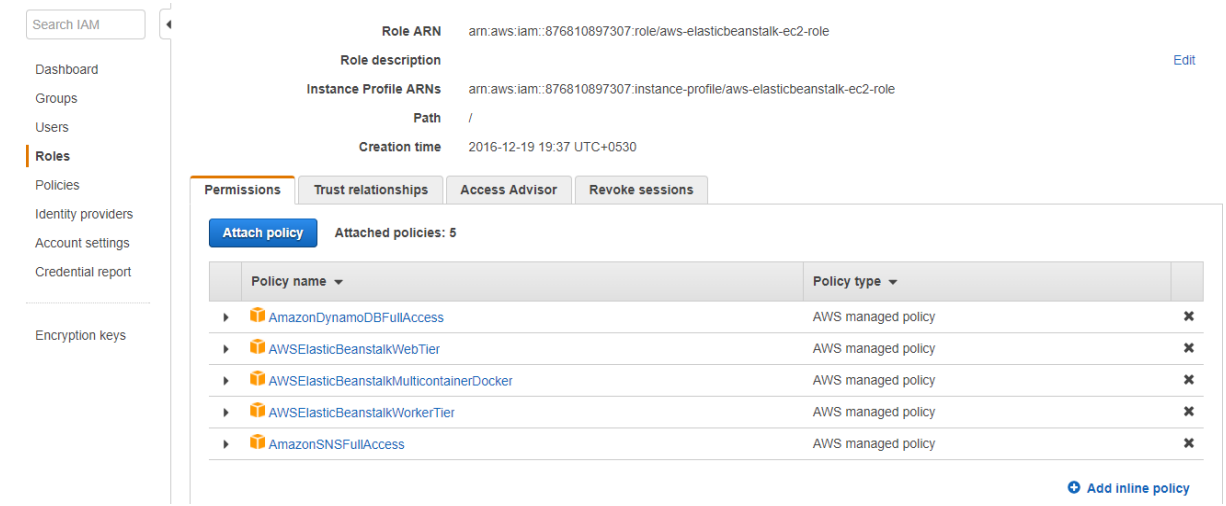
The sample application uses instance permissions to write data to a DynamoDB table, and to send notifications to an Amazon SNS topic

Now Go to IAM and roles and chose aws-elasticbeanstalk-ec2-role.



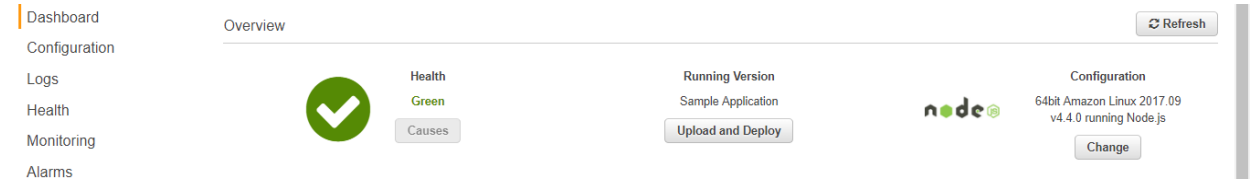
And click permissions manage permissions and attach these policies to the role

- **AmazonDynamoDBFullAccess**
- **AmazonSNSFullAccess**

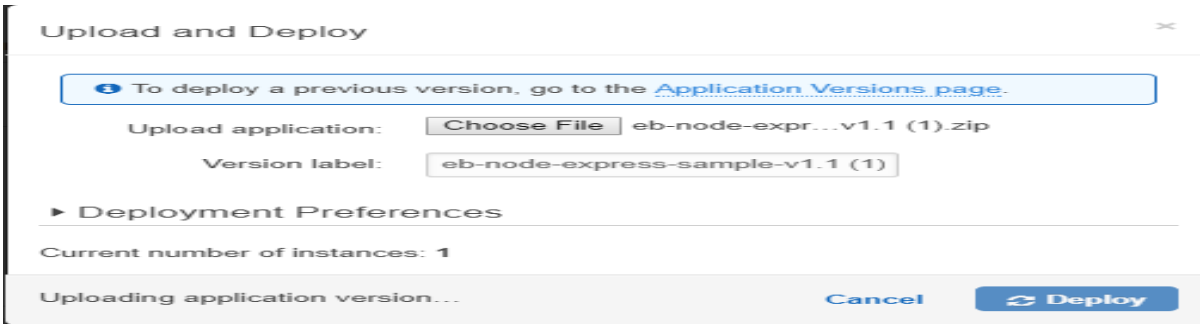


Step 6: Deploy the sample app

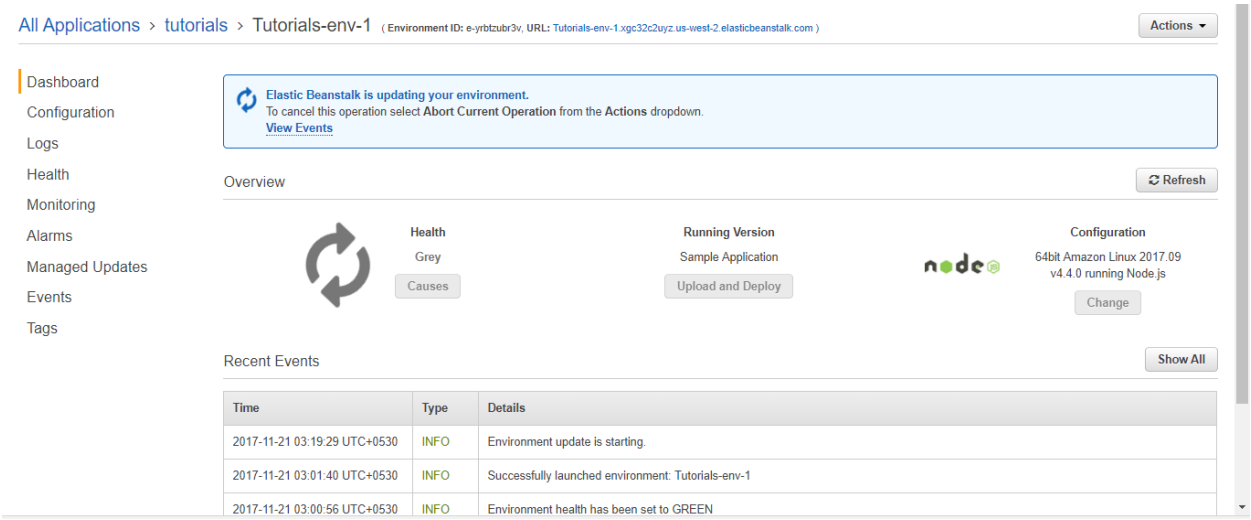
Go to Elastic Beanstalk dashboard and click on the application Tutorials-env.Now you will see a 'upload and deploy' button.



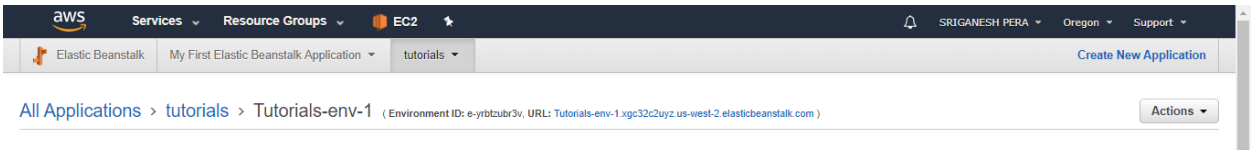
Click on that and upload and Deploy button and chose the file that you've downloaded in the first step and hit deploy.



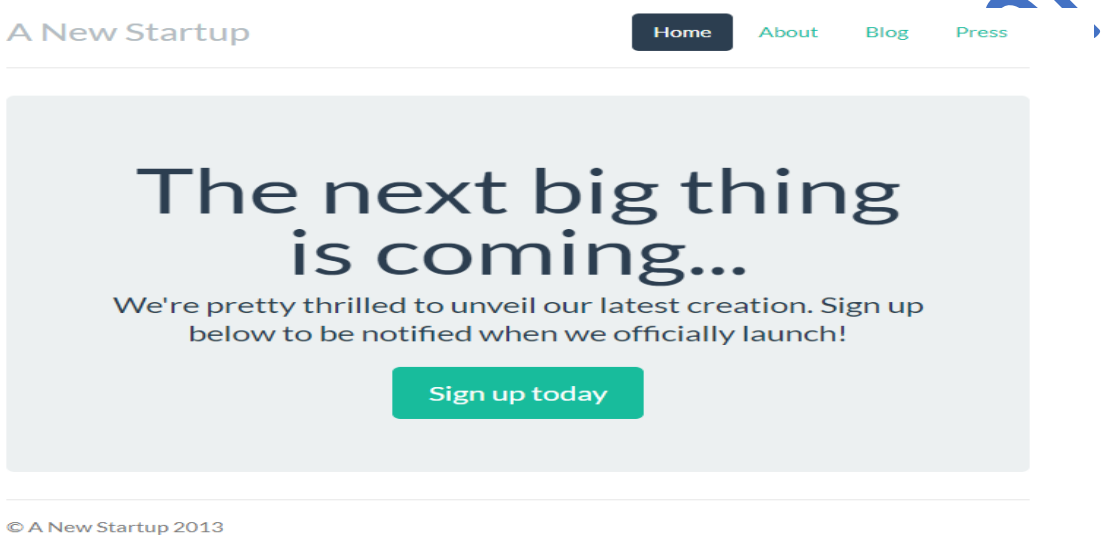
It takes several minutes for the application to be deployed and ready to operated on.You can monitor the status in 'Recent Events'



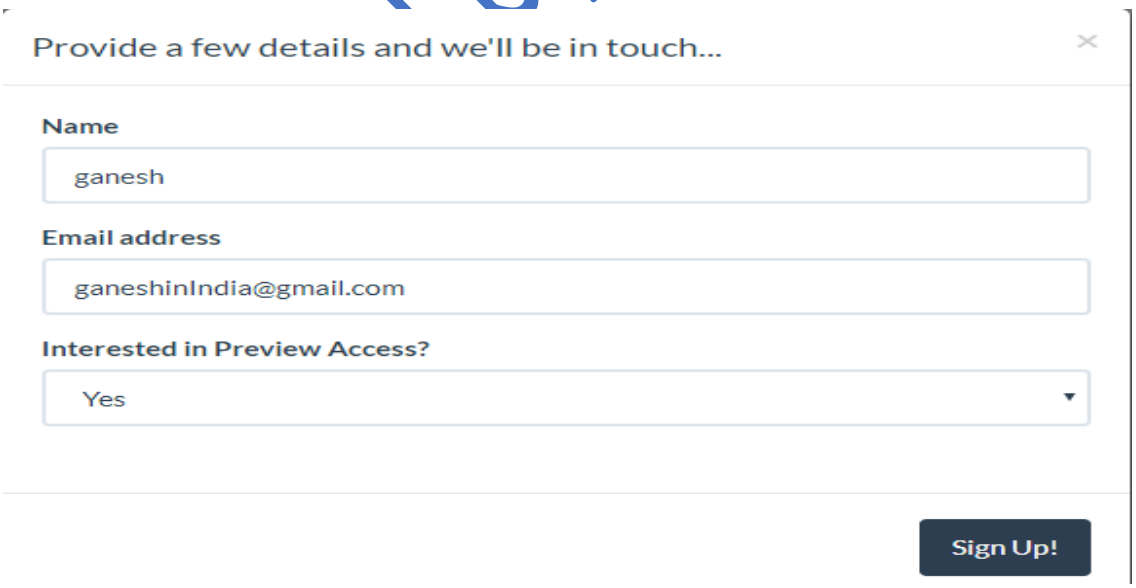
Once the health turns green.Click on the url as shown in the dashboard.



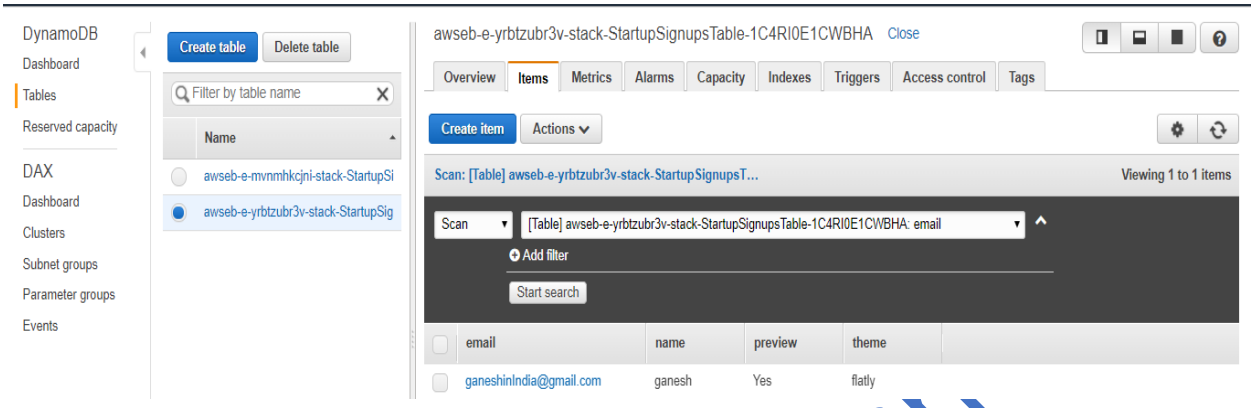
Now you will see a webapp built on Node.js and deployed on Elastic Beanstalk.



Now click on signup today and enter details



Now go back to your DynamoDB and you will find the details entered in the item of your table that is created by Elastic beanstalk.



Now go to SNS console and Open the Topics page and you'll find the topic that the application created. The name starts with **'awseb'** and contains **NewSignupTopic**.

<u>Choose the topic to view its subscriptions.</u>

Step 7: Create new tables in DynamoDB

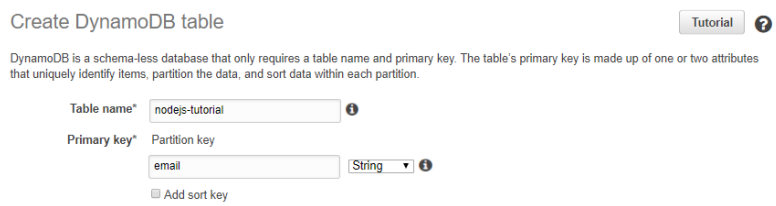
Create a table with the following settings:

Table name – nodejs-tutorial

Primary key – email

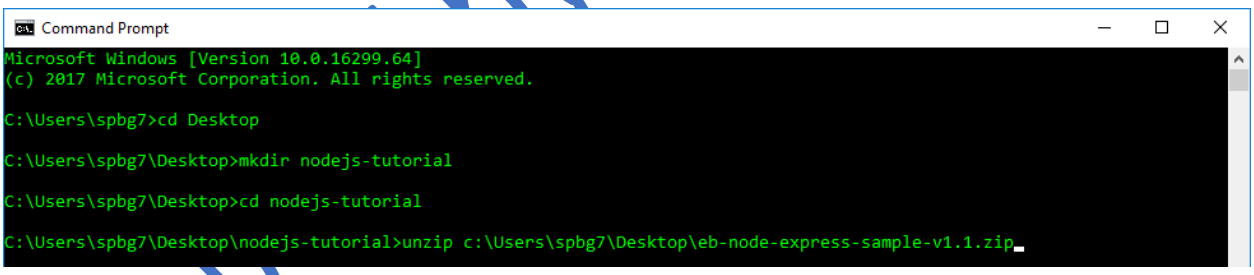
Primary key type – String

and click Create



Step 8: Update the application configuartion files

a)Type the following command as shown.File path needs to be changed to your own path.



```
Command Prompt
Archive:  c:/Users/spbg7/Desktop/eb-node-express-sample-v1.1.zip
f53517593522821 8a8c3b0aca5cf2b383dc9a5d8
   creating: .ebextensions/
  inflating: .ebextensions/create-dynamodb-table.config
  inflating: .ebextensions/create-sns-topic.config
  inflating: .ebextensions/options.config
  inflating: .gitignore
  inflating: LICENSE
  inflating: README.md
  inflating: app.js
  inflating: iam_policy.json
   creating: misc/
  inflating: misc/theme-flow.png
  inflating: npm-shrinkwrap.json
  inflating: package.json
   creating: static/
   creating: static/bootstrap/
  inflating: static/bootstrap/LICENSE
   creating: static/bootstrap/css/
  inflating: static/bootstrap/css/jumbotron-narrow.css
   creating: static/bootstrap/css/theme/
   creating: static/bootstrap/css/theme/amelia/
  inflating: static/bootstrap/css/theme/amelia/bootstrap.css
   creating: static/bootstrap/css/theme/default/
  inflating: static/bootstrap/css/theme/default/bootstrap.css
   creating: static/bootstrap/css/theme/flatly/
  inflating: static/bootstrap/css/theme/flatly/bootstrap.css
   creating: static/bootstrap/css/theme/slate/
  inflating: static/bootstrap/css/theme/slate/bootstrap.css
   creating: static/bootstrap/css/theme/united/
```

b) Now Open .ebextensions/options.config and change the values of the following settings:

NewSignupEmail – Your email address.

STARTUP_SIGNUP_TABLE – nodejs-tutorial

```
1  option_settings:
2    aws:elasticbeanstalk:customoption:
3      NewSignupEmail: gannubabuus@gmail.com
4    aws:elasticbeanstalk:application:environment:
5      THEME: "flatly"
6      AWS_REGION: '`{"Ref" : "AWS::Region"}`'
7      STARTUP_SIGNUP_TABLE: nodejs-tutorial
8      NEW_SIGNUP_TOPIC: '`{"Ref" : "NewSignupTopic"}`'
9    aws:elasticbeanstalk:container:nodejs:
10     ProxyServer: nginx
11   aws:elasticbeanstalk:container:nodejs:staticfiles:
12     /static: /static
```

This configures the application to use the **nodejs-tutorial** table instead of the one created by .ebextensions/create-dynamodb-table.config, and sets the email address that the Amazon SNS topic uses for notifications.

c) Now remove the `.ebextensions/create-dynamodb-table.config`.

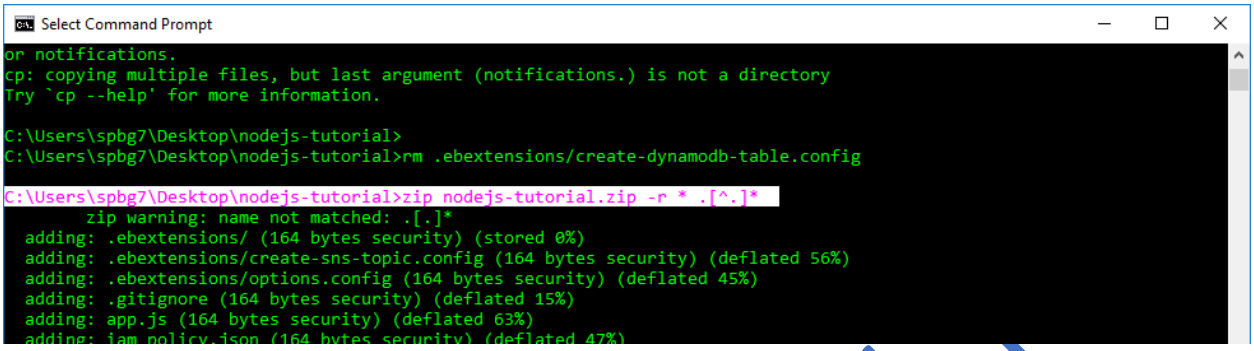
```
C:\Users\spbg7\Desktop\nodejs-tutorial>
C:\Users\spbg7\Desktop\nodejs-tutorial>rm .ebextensions/create-dynamodb-table.config

C:\Users\spbg7\Desktop\nodejs-tutorial>
```

The next time you deploy the application, the table created by this configuration file will be deleted.

d) Create a source bundle from the modified code.

Copy and paste this on command line:

```
zip nodejs-tutorial.zip -r * .[^.]*
```



<u>Step 9: Deploy the nodejs-tutorial.zip source bundle to your environment.</u>

You will find the zip file inside the nodejs-tutorial folder itself.

Go to Elastic Beanstalk console and click on your application and click upload and deploy ,then locate the file and click deploy.

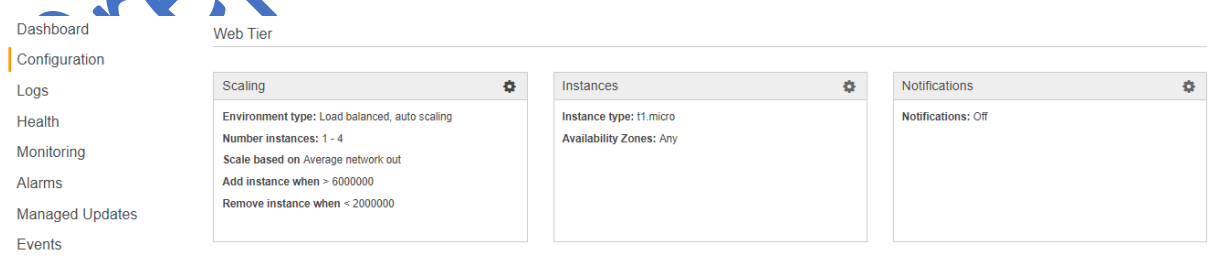
When you deploy,Elastic Beanstalk updates the configuration of the Amazon SNS topic and deletes the DynamoDB table that it created when you deployed the first version of the application.

Now after you find the health in green.Open the Elastic Beanstalk URL and click on signup and enter the details.

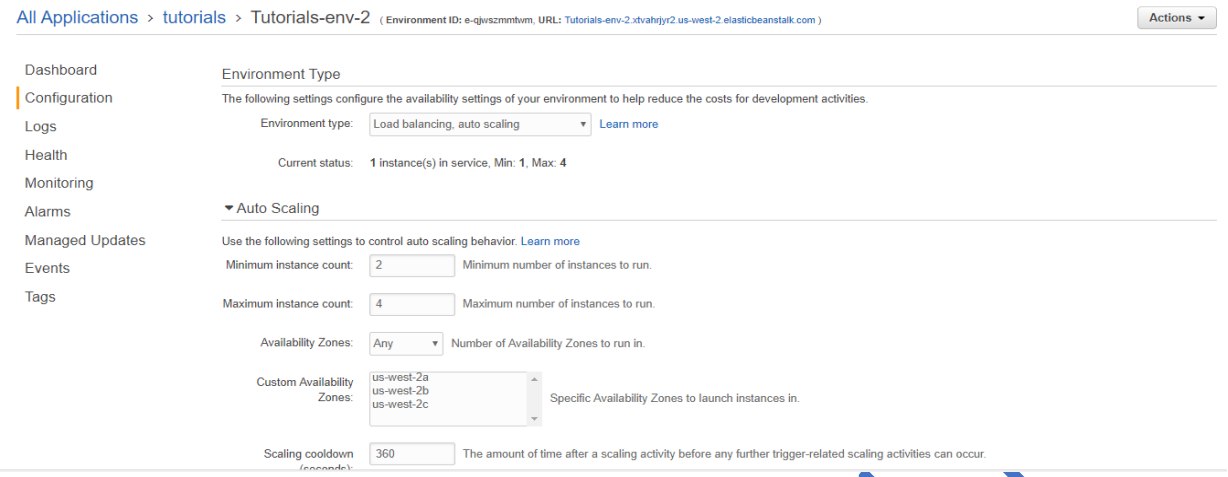
Take a guess? In which table those new details were saved.

<u>Stp 10: Configure for high-availiability</u>

Now to go to Elastic Beanstalk dashboard and click on configuration under dashboard and click on settings button in scaling block.



Click on Autoscaling and change the minimum instance count to 2. Thus you can ensure high-availiabilty for your application.

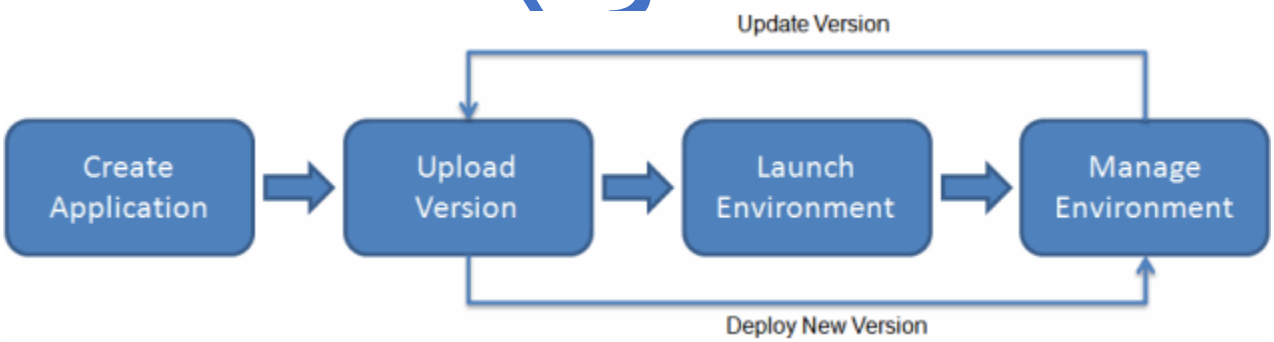


Step 11 : Clean up!

Now go back to Elastic Beanstalk dashbaord.In action click on delete application.Elastic beanstalk deletes both the application and environment.

But this wont delete the DynamoDB,and you have to manually delete it.

Application deployment and lifecycle management of Elastic Beanstalk:



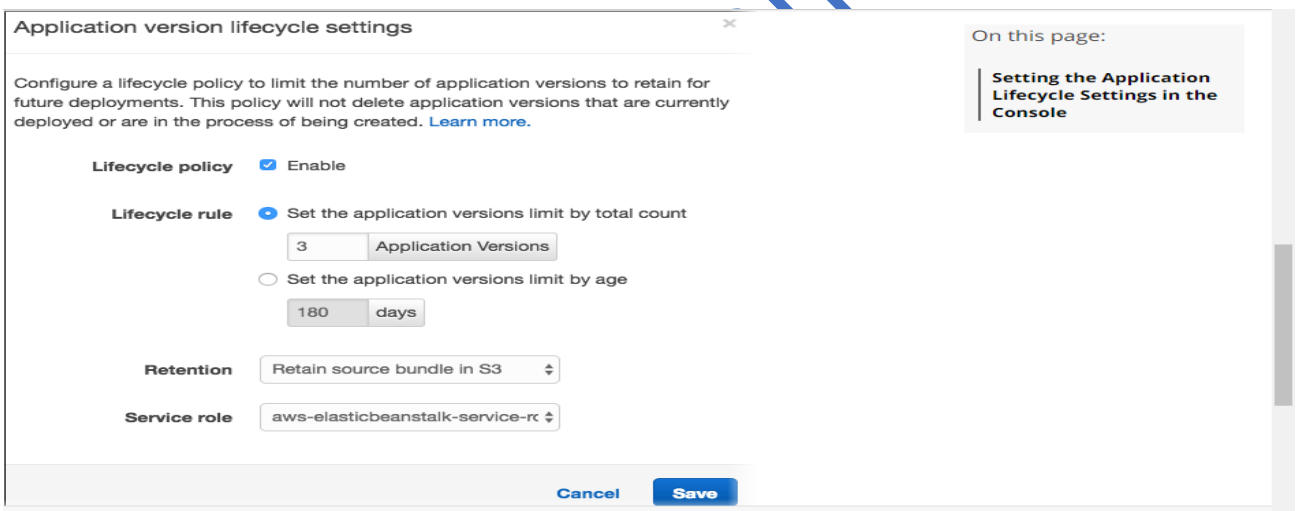
Initially,a web application is developed with preferred programming language on developer's machine.Once it is developed,the source code is converted into a source bundle,for example in Java,source bundle is converted into .war file.

Once the initial version of source bundle has been uploaded ,Elastic Beanstalk automatically launches and configures the underlying infrastructure for running the source bundle

With time,as business requirement changes,it is also possible to upload a newer version of a web application.This helps developers to focus just on their application development rather than on infrastrutcure components.

Version lifecycle limits: A newer application version is created when a newer source code is uploaded.Creating a newer version and deleting the old unwanted application version leads to hitting application limit.

Resource	Default limit
Applications	75
Application versions	1000
Environments	200



Elastic Beanstalk Components:

The following are the various components that work together to make it possible to deploy and manage custom applications easily on AWS cloud:

Application:

This is a logical collection of environment,versions and configuration variables.Just like a folder that consists of code.

Application Version:

This refers to specific source code version for a web application like a .war file in Java.An application can have multiple source codes.

Generally,application runs with the latest code version.At times,multiple versions of an application may run simultaneously for catering to users in different geography or for testing purposes.

Environment:

There are two types of environment:

1.Webserver: Listens and process HTTPS requests

2.Worker: Process background tasks and listens on messages in AWS SQS queue

Each environment runs only a single application version at a time.

Environment configuration:

This is a set of parameters and settings,it defines how an environment and its associated resources will behave.Elastic Beanstalk will automatically apply changes from the environment configuration to the exsisiting resources.If required,it may delete exsisting resources and create new ones to match environment configuration change.

Configuration template:

This is a starting point for creating unique environment configurations.

Architectural concepts:

As you noticed environment is the most essential component to deploy a web application.Creating a new environment requires selecting the appropriate environment tier,platform and environment type.

These environment tiers are divided into two types:

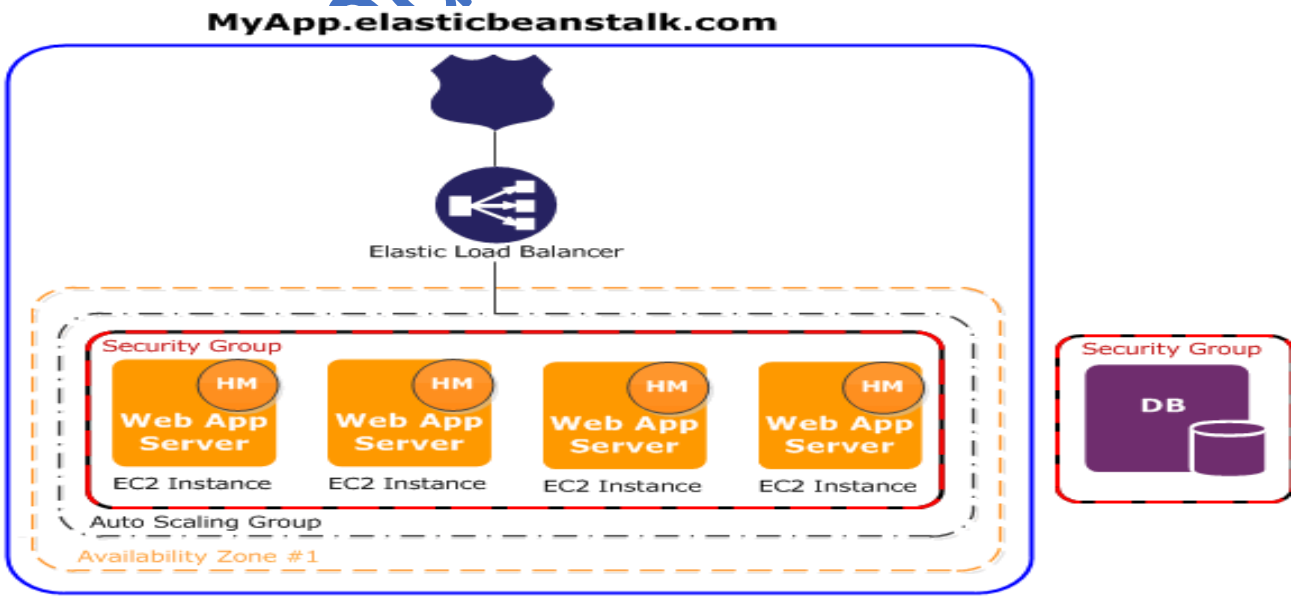
Web server environment: This hosts a web application and handles HTTP(S) requests.This environment is called web server tier

Worker environment: This hosts a web application and handle long-running or scheduled background processing tasks.This environment is called a worker tier.

At any point of time,each environment can support either of the environment tier.The reason for this limitation is that ,Elastic Beanstalk can handle only one Auto Scaling group.Each of the environment tiers requires one dedicated Auto Scaling group to host a web application.

Web server environment:

This following diagram illustrates an example Elastic Beanstalk architecture for a web server environment tier and shows how the components in that type of environment tier work together:



The solid blue line that you see is the environment within which your application is created. This environment provisions all the AWS resources that your application requires to run.

The resources here are: EC2 instances, Auto Scaling groups, ELB.

Every environment like this one has a CNAME and an alias in the Amazon Route 53 pointing to ELB, where this ELB is a part of Auto Scaling group and it sits in front of EC2 instances. The registered domain name will forward the end user's request to access web application on the CNAME.

The software stack running on the Amazon EC2 instances is dependent on the container type. A container type defines the infrastructure topology and software stack to be used for that environment.

The software stack may include one or more components such as programming language(Python,PHP,Java and so on) a web server(Apache,Tomcat etc) and also web container and Docker container.

As you can see there's a software component called the **host manager (HM)** runs on each Amazon EC2 server instance.

Host manager is responsible for the following:

1. Deploying the application
2. Aggregating events and metrics for retrieval
3. Generating instance-level events
4. Monitoring the application server and log files for critical errors
5. Patching instance components
6. Rotating your application's log files and publishing them to Amazon S3

The host manager reports metrics, errors and events, and server instance status.

By default,Elastic Beanstalk creates a security group which allows everyone to connect using port 80 (HTTP).

You can customize security groups as per application's requirement.

Worker environment tiers:

The worker environment tier includes an Auto Scaling group, at least one EC2,and an IAM role. Optionally, it also creates an AWS SQS queue.

When you launch a worker environment tier, Elastic Beanstalk installs the necessary support files for your programming language of choice and a daemon on each EC2 instance in the Auto Scaling group. The daemon is responsible for pulling requests from an Amazon SQS queue and then sending the data to the web application running in the worker environment tier that will process those messages. If you have multiple instances in your worker environment tier, each instance has its own daemon, but they all read from the same Amazon SQS queue.