

# **CONTENTS**

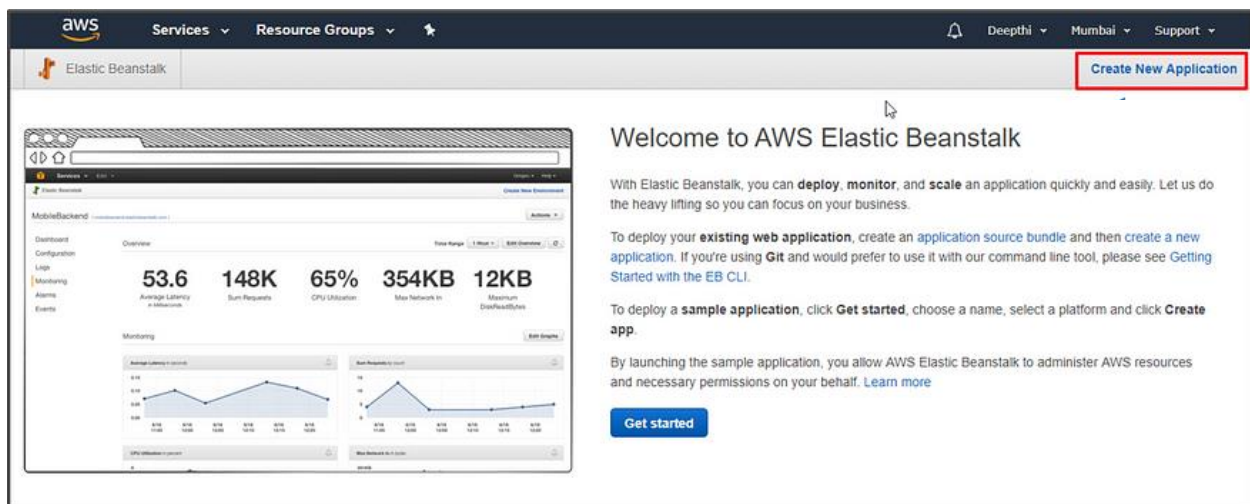
ACKNOWLEDGEMENT.....	1
Certificate .....	2
Candidate's Declaration.....	3

1. Introduction .....	(4-5)
2. Objective .....	(6-7)
3. Theory.....	(8-9)
4. Steps.....	(10-14)
5. Python Optional Codes.....	(15-17)
6. System Analysis.....	(18)

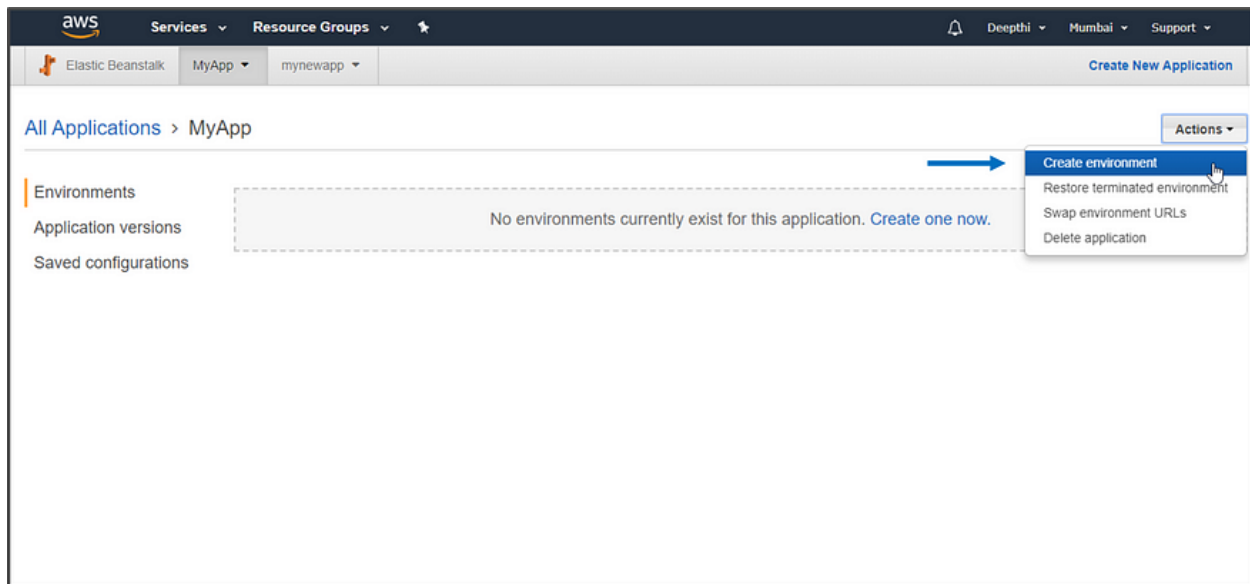
## Deploy an Application on Elastic Beanstalk

Deploying an application on Elastic Beanstalk is a fairly simple process. Let's see how to deploy an application stepwise.

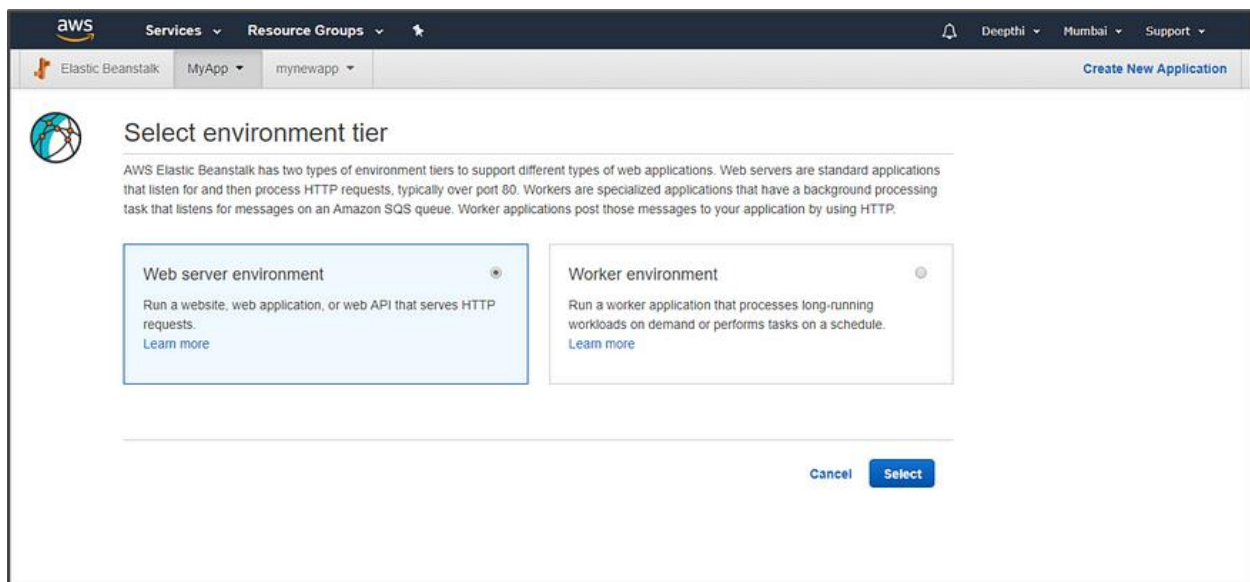
**Step 1:** On the Elastic Beanstalk console click on Create New Application option. A dialog box appears where you can give a name and appropriate description for your application.



**Step 2:** Now that the application folder has been created, you can click on the Actions tab and select Create Environment option. Beanstalk provides you with an option where you can create multiple Environments for your application.



**Step 3:** Choose among two different Environment Tier options. Choose Web Server Environment if you want your application to handle HTTP requests or choose Worker Environment to handle background tasks.



**Step 4:** Another dialog appears, where you need to provide a domain name and description for your application.

aws Services Resource Groups

Elastic Beanstalk MyApp mynewapp Create New Application

### Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)

#### Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Application name MyApp

Environment name Myapp-env

Domain Tomcatapp .ap-south-1.elasticbeanstalk.com Check availability

Tomcatapp.ap-south-1.elasticbeanstalk.com is available.

Description Tomcat Web Application

**Step 5:** Choose a platform of your choice for your application. Elastic Beanstalk will provide you with multiple options. You can choose a sample application provided by Beanstalk, or upload a file which has code for your application.

Base configuration

Platform ☒ Preconfigured platform

Platforms published and maintained by AWS Elastic Beanstalk.

Tomcat

-- Choose a platform --

☒ Generic

Docker

Multi-container Docker

☒ Preconfigured

Elastic Beanstalk Packer Builder

Go

.NET (Windows/IIS)

Java

Node.js

Ruby

PHP

Python

Tomcat App.

☒ Preconfigured - Docker

GlassFish

Go

Python

Upload a source bundle from your computer or copy one from Amazon S3.

Upload ZIP or WAR

Cancel Configure more options Create environment

Beanstalk will take a few minutes to launch an Environment. Once the Environment is launched, on the navigation pane you can see multiple options where you can change the configuration of your application, view log files, and events. Since you're already on Environment page, try exploring different features that Beanstalk offers.

The screenshot displays the AWS Elastic Beanstalk console. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', and user information. The breadcrumb trail shows 'All Applications > MyApp > Myapp-env'. The environment ID is 'e-d5gmmvpajm' and the URL is 'Tomcatapp.ap-south-1.elasticbeanstalk.com'.

**Creating Myapp-env**  
This will take a few minutes.

10:07am Waiting for EC2 instances to launch. This may take a few minutes.  
10:06am Created EIP: 13.232.180.122  
10:06am Created security group named: awseb-e-d5gmmvpajm-stack-AWSEBSecurityGroup-BC28ZICX1XMO  
10:06am Environment health has transitioned to Pending. Initialization in progress (running for 4 seconds). There are no instances.  
10:06am Using elasticbeanstalk-ap-south-1-400572186703 as Amazon S3 storage bucket for environment data.  
10:06am createEnvironment is starting.

**Learn More**  
[Get started using Elastic Beanstalk](#)  
[Modify the code](#)  
[Create and connect to a database](#)  
[Add a custom domain](#)

**Featured**  
[Create your own custom platform](#)

**Command Line Interface (v3)**  
[Installing the AWS EB CLI](#)  
[EB CLI Command Reference](#)

**Overview**

**Health**  
Ok  
[Causes](#)

**Running Version**  
Sample Application  
[Upload and Deploy](#)

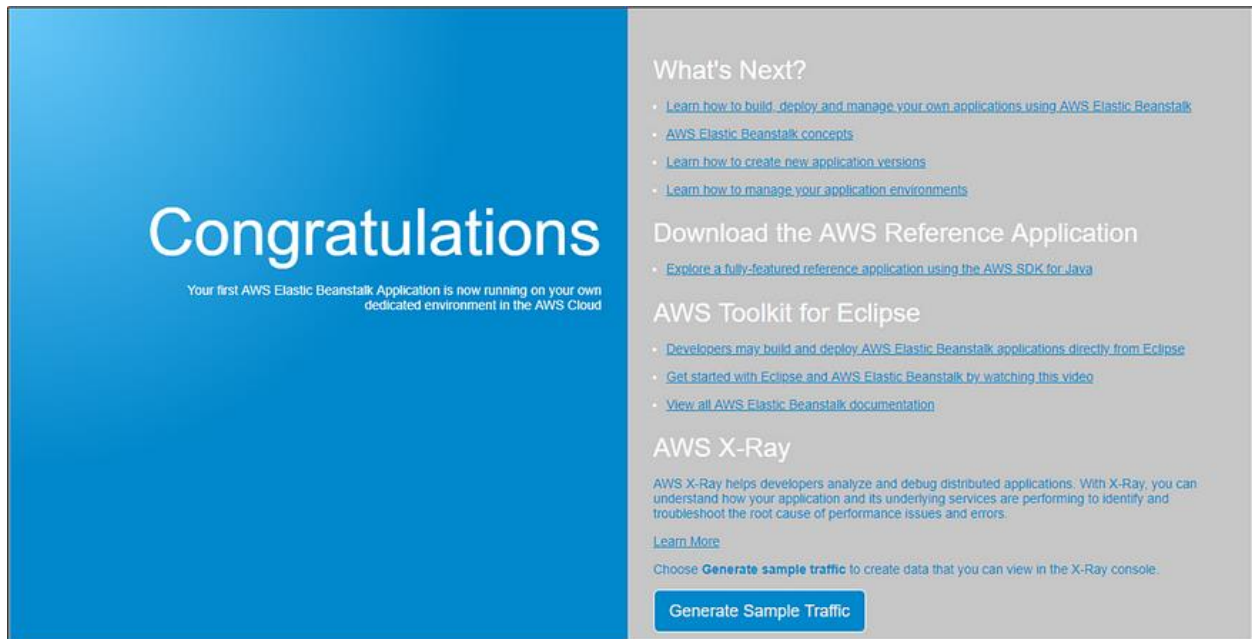
**Configuration**  
Tomcat 8.5 with Java 8 running on 64bit Amazon Linux/3.0.4  
[Change](#)

**Recent Events**  
[Show All](#)

Time	Type	Details
2018-10-13 10:08:29 UTC+0530	INFO	Added instance [i-0960cefc9e5c8db89] to your environment.
2018-10-13 10:08:29 UTC+0530	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 9 seconds ago and took 2 minutes.
2018-10-13 10:08:29 UTC+0530	INFO	Successfully launched environment: Myapp-env
2018-10-13 10:08:28 UTC+0530	INFO	Application available at Tomcatapp.ap-south-1.elasticbeanstalk.com.

**Navigation Pane:**  
Dashboard  
Configuration  
Logs  
Health  
Monitoring  
Alarms  
Managed Updates  
Events  
Tags

**Step 6:** On the top right corner, you will find the URL of your application version. Click on that URL. You will be taken to a page which will confirm that you have successfully launched your application on Elastic Beanstalk.



I hope now you have a clear picture of Elastic Beanstalk and how you can use Beanstalk to deploy your applications.

## **Python Optional Open Source Codes**

Here's a basic example using the Flask web framework and the Serverless Framework to deploy the application to AWS Lambda:

1. Install Flask and Serverless Framework:

```
pip install Flask serverless
```

2. Create a Python script (app.py) for your Flask web application:

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def index():
    return jsonify({"message": "Hello, this is your serverless Flask app on AWS Lambda!"})

if __name__ == '__main__':
    app.run(debug=True)
```

### 3. Create a serverless.yml file for your Serverless Framework configuration:

```
service: my-flask-app
```

```
provider:
```

```
  name: aws
```

```
  runtime: python3.8
```

```
functions:
```

```
  app:
```

```
    handler: app.index
```

```
  events:
```

```
    - http:
```

```
      path: /
```

```
      method: ANY
```

### 4. Deploy your application using Serverless:

```
sls deploy
```

This will package your Flask application and deploy it to AWS Lambda. After deployment, Serverless will provide you with an API Gateway URL. Visit the provided URL in your web browser or use a tool like curl or Postman to test the endpoint.