

School of Computer Science and Engineering (SCOPE)

Fall Semester 2025-26

CBS3005 - Cloud, Microservices and Applications

LAB ASSESSMENT 2

Submitted by-

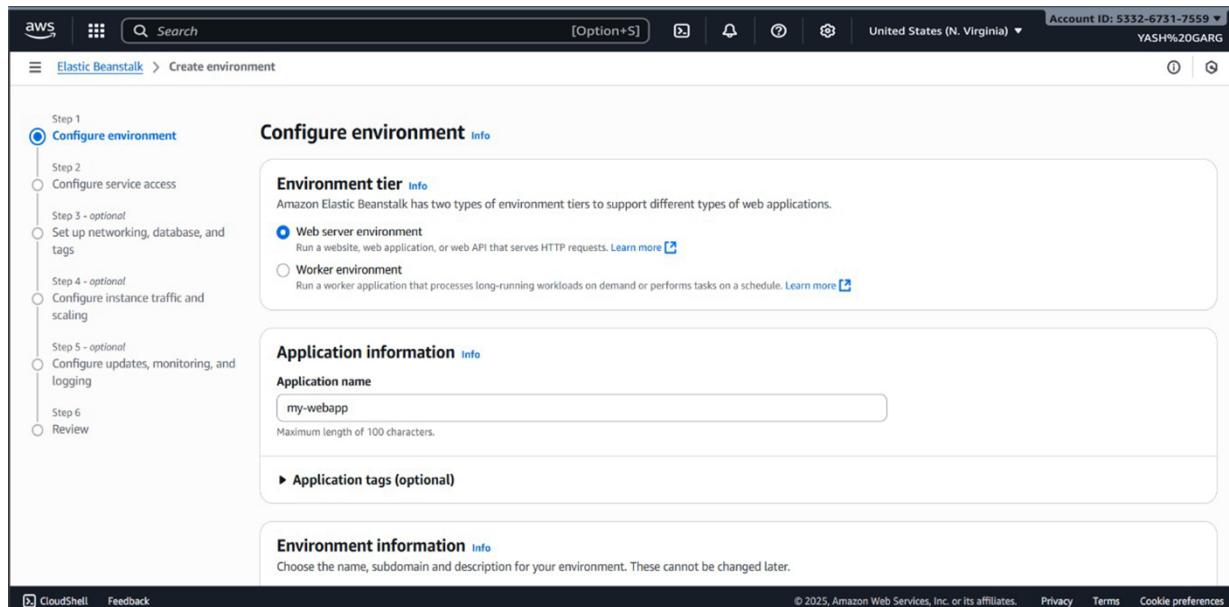
YASH GARG

22BBS0183

Q1. Create a simple web application using your preferred programming language and framework (e.g., Node.js, Python, Java etc.). Ensure the application is fully functional and ready for deployment. Initialize an AWS Elastic Beanstalk environment for your application. Choose the appropriate platform (e.g., Node.js, Python, Java) and configure the environment settings. Package your web application and deploy it to the Elastic Beanstalk environment. Access the deployed web application via the Elastic Beanstalk URL provided. Test its functionality to confirm that the deployment was successful and that the application is accessible and performs as expected.

Step 1 – Launching the Application on Elastic Beanstalk

- Log in to your AWS Management Console.
- Navigate to **Elastic Beanstalk** and select **Create Application**.
- Provide an application name, e.g., *my-webapp*.
- For the platform, choose **Python** (suitable for serving static HTML).
- Under **Application code**, select **Upload your code** and upload the .zip package containing your project files.
- Click **Create Application** and wait a few minutes (usually 2–5) for the environment to be initialized.



Configure environment [Info](#)

Environment tier [Info](#)
Amazon Elastic Beanstalk has two types of environment tiers to support different types of web applications.

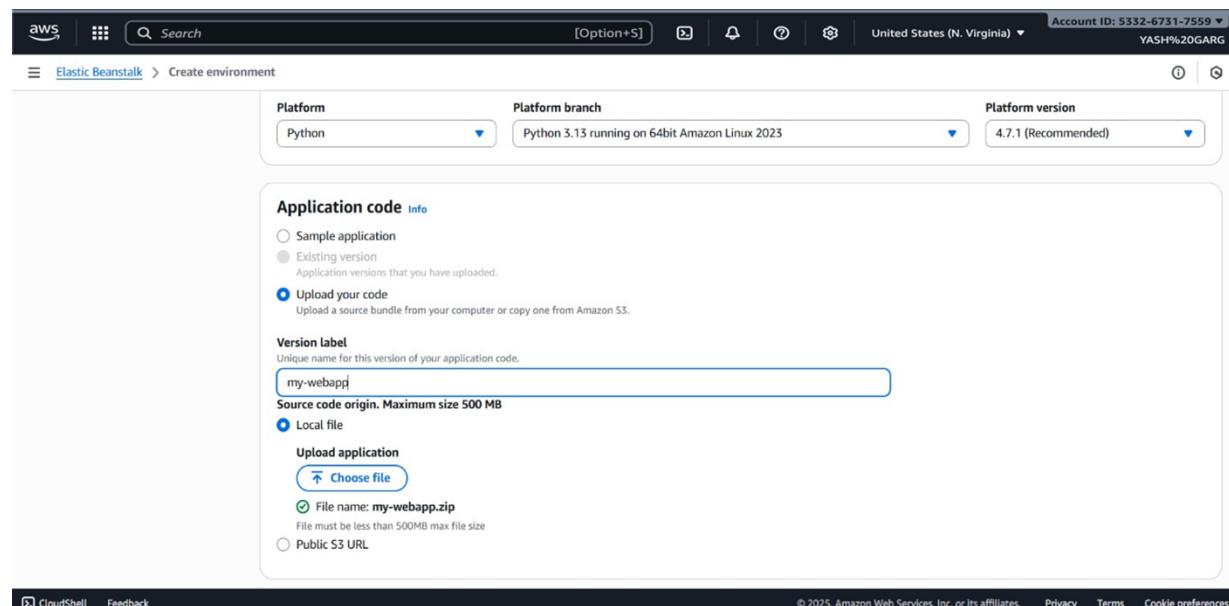
Web server environment
Run a website, web application, or web API that serves HTTP requests. [Learn more](#)

Worker environment
Run a worker application that processes long-running workloads on demand or performs tasks on a schedule. [Learn more](#)

Application information [Info](#)

Application name
my-webapp
Maximum length of 100 characters.

Environment information [Info](#)
Choose the name, subdomain and description for your environment. These cannot be changed later.



Platform	Platform branch	Platform version
Python	Python 3.13 running on 64bit Amazon Linux 2023	4.7.1 (Recommended)

Application code [Info](#)

Sample application

Existing version
Application versions that you have uploaded.

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

Version label
Unique name for this version of your application code.
my-webapp

Source code origin Maximum size 500 MB

Local file

Upload application

File name: my-webapp.zip
File must be less than 500MB max file size

Public S3 URL

Step 2 – Setting Up the Service Role

- On the service role section, click **Create role** (this opens the IAM console in a new tab).
- In the IAM console, choose **Elastic Beanstalk** as the trusted service.
- Attach the recommended managed policy:

AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy

- Provide a descriptive name (e.g., *aws-elasticbeanstalk-service-role*).
- Click **Create role** to finalize.

The screenshot shows the AWS IAM Roles page. At the top, a green banner indicates that the role 'aws-elasticbeanstalk-service-role' has been created. The main table lists three roles: 'aws-elasticbeanstalk-service-role' (selected), 'AWSServiceRoleForTrustedAdvisor', and 'X.509 Standard'. The 'aws-elasticbeanstalk-service-role' row shows 'AWS Service: trustedadvisor (Service)'. Below the table, there are three sections: 'Access Anywhere', 'X.509 Standard', and 'Temporary credentials'. The 'Access Anywhere' section is expanded, showing options for 'Access AWS from your non AWS workloads' and 'X.509 Standard'. The 'Temporary credentials' section is also visible. The left sidebar shows navigation links for IAM, Access management, Access reports, and CloudShell.

Step 3 – Creating the EC2 Instance Profile

- Back in the Beanstalk configuration screen, under **EC2 instance profile**, click **Create role** (opens IAM again).
- This time, select **EC2** as the service.
- Attach the following managed policies:
 - AWSElasticBeanstalkWebTier
 - AWSElasticBeanstalkWorkerTier
 - AWSElasticBeanstalkMulticontainerDocker (*not required for static HTML but useful for future scalability*)
- Name the profile something like *aws-elasticbeanstalk-ec2-role*.
- Create the role.

Service access [Info](#)
Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role	EC2 instance profile
arn:aws:iam::217553593570:role/aws-elasticbeanstalk-service-role	aws-elasticbeanstalk-ec2-role

Step 3: Set up networking, database, and tags [Edit](#)

Networking, database, and tags [Info](#)
Configure VPC settings, and subnets for your environment's EC2 instances and load balancer. Set up an Amazon RDS database that's integrated with your environment.

No options configured

Tags

Key	Value
No tags There are no tags defined	

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 1
Configure environment

Step 2
Configure service access

Step 3 - optional
Set up networking, database, and tags

Step 4 - optional
Configure instance traffic and scaling

Step 5 - optional
Configure updates, monitoring, and logging

Step 6
Review

Review [Info](#)

Step 1: Configure environment [Edit](#)

Environment information

Environment tier	Application name
Web server environment	my-webapp
Environment name	Application code
My-webapp-env	my-webapp.zip
Platform	
arm:aws:elasticbeanstalk:us-east-1::platform/Python 3.13 running on 64bit Amazon Linux 2023/4.7.1	

Step 2: Configure service access [Edit](#)

Service access [Info](#)
Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role	EC2 instance profile
--------------	----------------------

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 4 – Verifying Deployment

- Once the environment status shows **Health: OK**, a URL will be generated, e.g.,

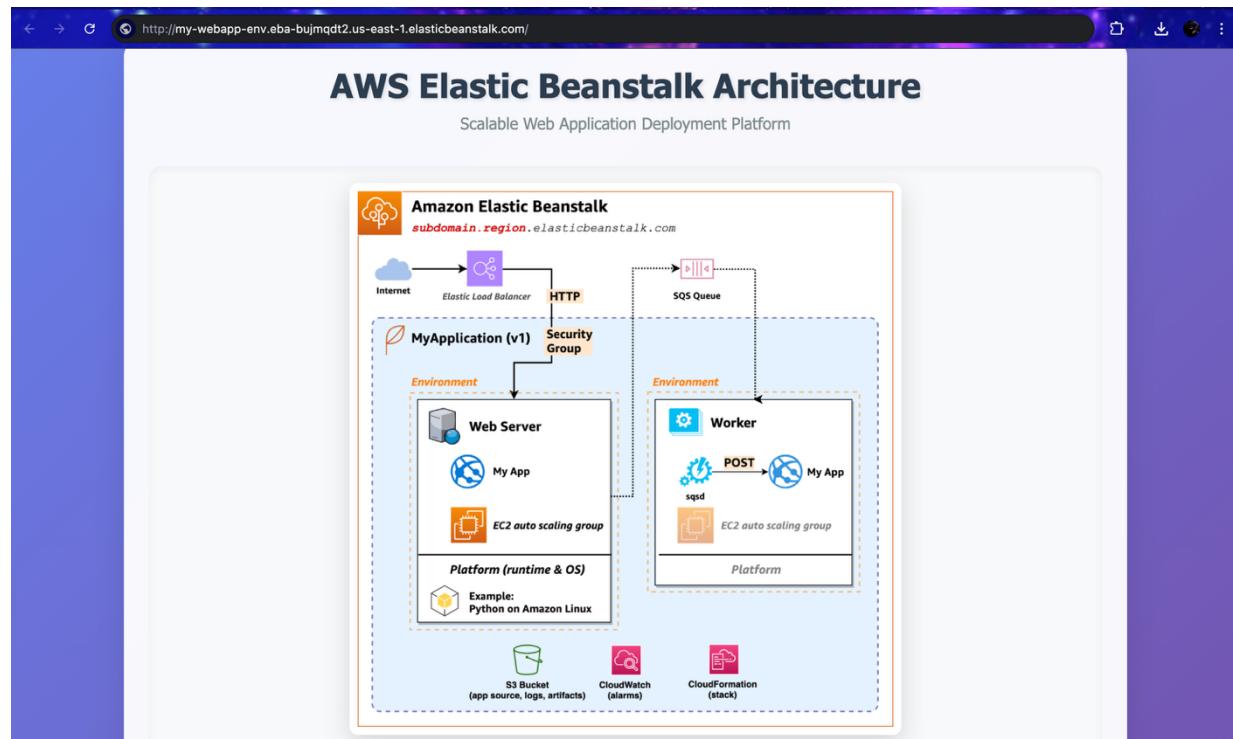
<http://my-webapp.us-east-1.elasticbeanstalk.com>

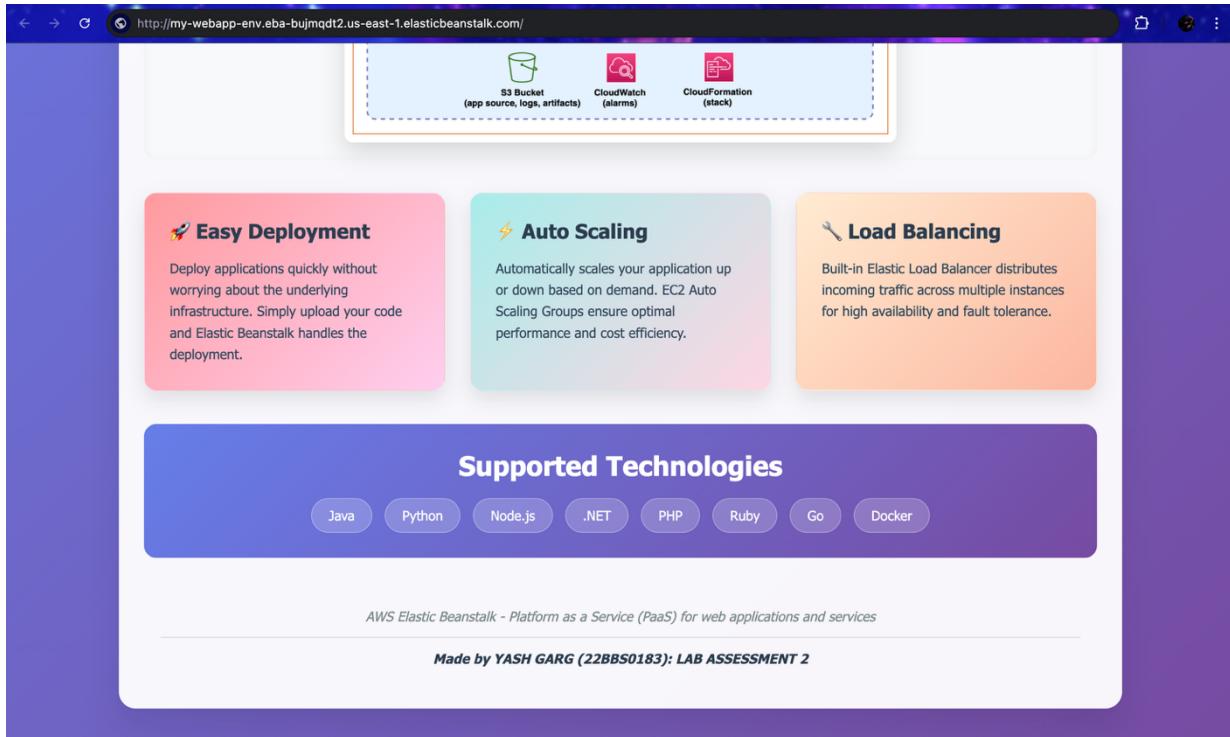
- Open the URL to access your deployed site.
- Test core features to ensure everything works properly:
 - Navigation links (Home, About, Contact).
 - Buttons and JavaScript functionality.
 - Page styling and images (CSS applied correctly).

Output:

My web application is now live and accessible via the Elastic Beanstalk-provided URL. Functionality and styling confirm that the deployment has been successful.

Link: <http://my-webapp-env.eba-bujmqdt2.us-east-1.elasticbeanstalk.com/>





Thank You