

Mini-Project-2 :

Deploying an Application on AWS Elastic Beanstalk with IAM Roles and Environment Swap

Statement :

- Deploying applications on the cloud requires not only hosting infrastructure but also proper access management. The goal of this project was to deploy a sample application on **AWS Elastic Beanstalk** while ensuring secure and controlled permissions using **IAM roles**.

Elastic Beanstalk :

AWS Elastic Beanstalk is a Platform as a Service (PaaS) that helps developers quickly deploy and manage applications without handling the complexity of the underlying infrastructure.

Steps for creation of beanstalk :

1. During the creation of Beanstalk selected roles which I has created related to requirement :

The screenshot shows the 'Configure service access' step in the AWS Elastic Beanstalk console. The left sidebar lists steps 1 through 6, with step 2 'Configure service access' selected. The main panel shows the 'Service access' section with the following configuration:

- Service role:** Choose an IAM role for Elastic Beanstalk to assume as a service role. The IAM role must have the required IAM managed policies. The selected role is **aws-elasticbeanstalk-service-role**.
- EC2 instance profile:** Choose an IAM instance profile with managed policies that allow your EC2 instances to perform required operations. The selected profile is **aws-elasticbeanstalk-ec2-role**.
- EC2 key pair - optional:** Select an EC2 key pair to securely log in to your EC2 instances. The selected key pair is **For_linux**.

Buttons at the bottom include 'Cancel', 'Skip to review', 'Previous', and 'Next'.

2. Skipped step 3 without modifying anything and in step 4 Added Root General Purpose3(SSD)

The screenshot shows the 'Configure instance traffic and scaling - optional' step in the AWS Elastic Beanstalk console. The left sidebar lists steps 1 through 6, with step 4 'Configure instance traffic and scaling' selected. The main panel shows the 'Instances' section with the following configuration:

- Root volume (boot device):**
 - Root volume type:** The type of storage for the root volume attached to each instance. The selected type is **General Purpose 3(SSD)**.
 - Size:** The number of gigabytes of the root volume attached to each instance. The selected size is **10 GB**.
 - IOPS:** Input/output operations per second for a provisioned IOPS (SSD) volume. The selected IOPS is **3000**.
 - Throughput:** The desired throughput for the Amazon EBS root volume attached to your environment's EC2 instance. The selected throughput is **125 MB/s**.
- Amazon CloudWatch monitoring:** The time interval between when metrics are reported from the EC2 instances. The selected monitoring interval is **5 minute**.

3. Selected Instance type :

▼ Capacity info

Configure the compute capacity of your environment and auto scaling settings to optimize the number of instances used.

Auto scaling group

Environment type

Select a single-instance or load-balanced environment. You can develop and test an application in a single-instance environment to save costs and then upgrade to a load-balanced environment when the application is ready for production. [Learn more](#)

Single instance

Fleet composition

Spot instances are launched at the lowest available price. [Learn more](#)

☒ On-Demand instance

☐ Spot instance

Architecture

The processor architecture determines the instance types that are made available. You can't change this selection after you create the environment. [Learn more](#)

☒ x86_64

This architecture uses x86 processors and is compatible with most third-party tools and libraries.

☐ arm64

This architecture uses AWS Graviton processors. You might have to recompile some third-party tools and libraries.

Instance types

Add instance types for your environment with you include at least two instance types. [Learn more](#)

1. t3.micro

preferred launch order. The order preference only applies to On-Demand instances and Spot instances that use the capacity optimized prioritized allocation strategy. We recommend

Add instance type

AMI ID

Elastic Beanstalk selects a default Amazon Machine Image (AMI) for your environment based on the Region, platform version, and processor architecture that you choose. [Learn more](#)

ami-06b463622da5a26f9

Cancel

Skip to review

Previous

Next

4. Select system in Health reporting :

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

Configure updates, monitoring, and logging - optional

▼ Monitoring info

Health reporting

Enhanced health reporting provides free real-time application and operating system monitoring of the instances and other resources in your environment. The `EnvironmentHealth` custom metric is provided free with enhanced health reporting. Additional charges apply for each custom metric. For more information, see [Amazon CloudWatch Pricing](#)

System

☒ Basic

☐ Enhanced

Health event streaming to CloudWatch Logs

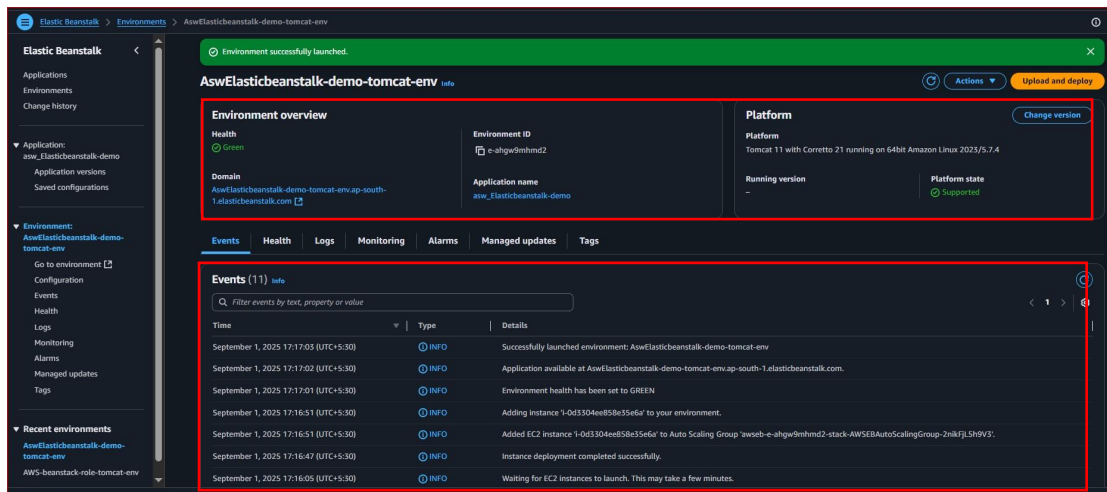
Configure Elastic Beanstalk to stream environment health events to CloudWatch Logs. You can set the retention up to a maximum of ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming

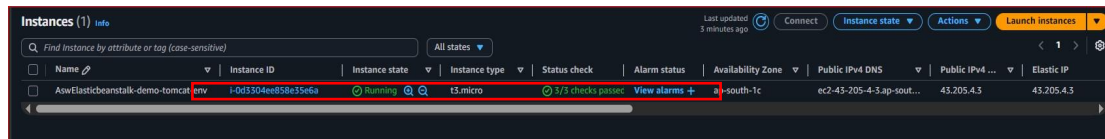
standard CloudWatch charges apply.

☐ Enable

5. Review and create



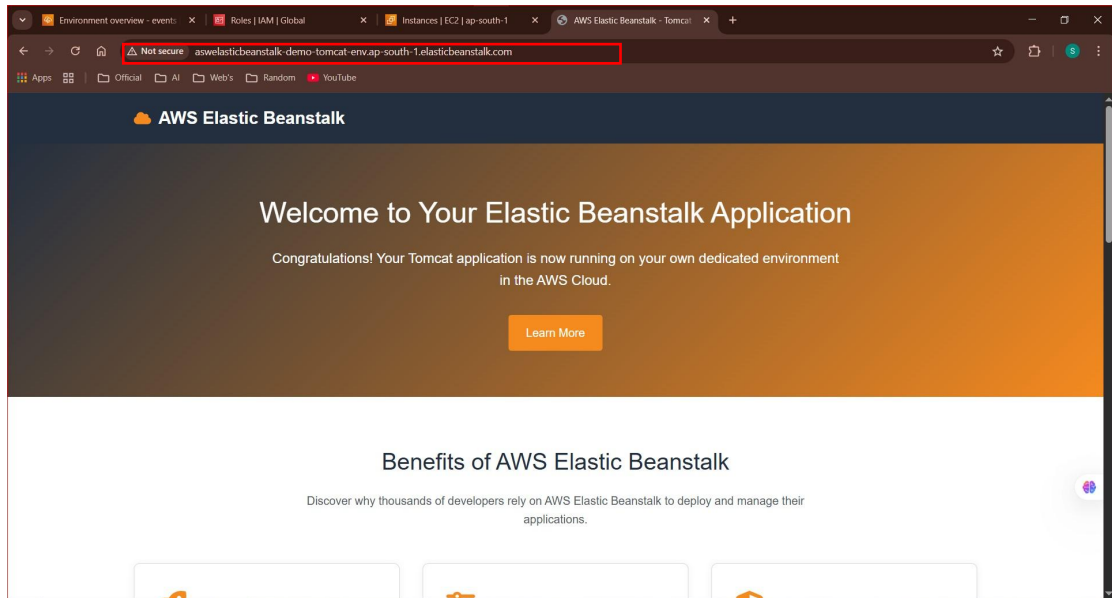
6. Successfully launched instance after creation of Beanstalk



When the environment was created in **AWS Elastic Beanstalk**, several automated steps took place in the background. The Events section shows the sequence of actions:

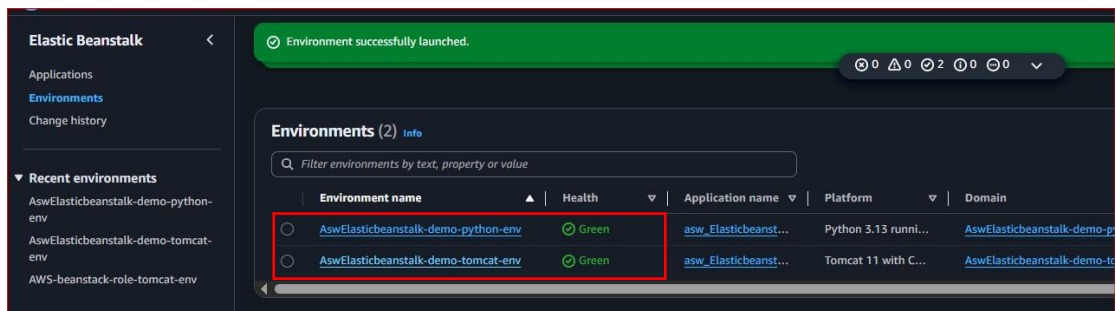
- Environment Launch Initiated** – Beanstalk started creating the environment `AswElasticbeanstalk-demo-tomcat-env`. (There was my typing mistake at giving name)
- Instance Provisioning** – An **EC2 instance** was launched to host the application.
- Auto Scaling Group Update** – The new EC2 instance was automatically added to the Auto Scaling Group, ensuring scalability and high availability.
- Deployment Completed** – Once the EC2 instance was ready, Beanstalk deployed the application code onto it.
- Application Accessible** – The application was made publicly available at the generated **Elastic Beanstalk domain URL**.
- Health Check Performed** – Beanstalk automatically ran health checks, and the environment health was set to **GREEN**, meaning the application is running without issues.

6. Copy DNS name and Search on browser :



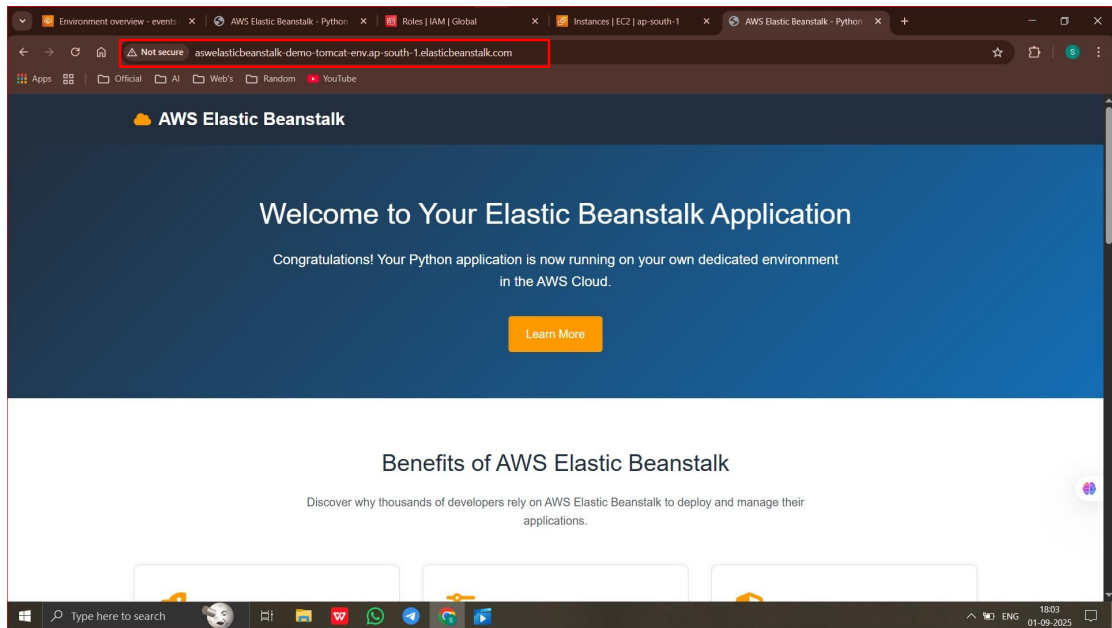
And finally our application is working on browser
Application was used tomcat, Now we swap environment of this application

7.Launched an environment for python application



8.Finally Swapping successfully :

Swap in Elastic Beanstalk changes the traffic from your **old environment (Blue)** to your **new environment (Green)** by Keeping their URLs same, giving you a smooth and risk-free way to update applications.



Stay Connected :



<https://www.linkedin.com/in/siddheshwar-shinde-90b73a353>



<https://github.com/codexshwar>

Thank you .