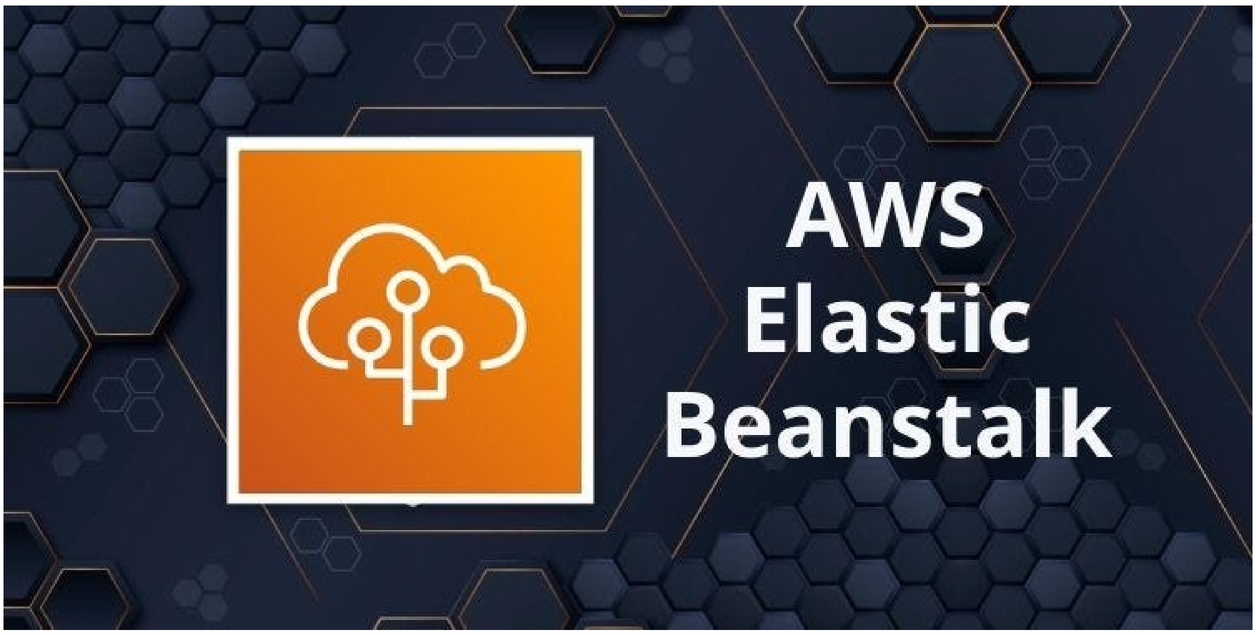
The Project Report on

"Deploy a Scalable Web App with Elastic Beanstalk"



Github Repository Link :

https://github.com/KPranit-2105/AWS-Projects

By :

Pranit P. Kolamkar

Abstract

The project, "Deploy a Scalable Web App with Elastic Beanstalk," focuses on utilizing Amazon Web Services (AWS) Elastic Beanstalk to simplify the deployment and management of a scalable web application. Elastic Beanstalk is a Platform-as-a-Service (PaaS) solution that abstracts the complexities of infrastructure provisioning, deployment, monitoring, and scaling, allowing developers to focus on writing code.

The project involves deploying a web application written in a supported programming language, such as Python, Node.js, or Java, and leveraging Elastic Beanstalk's capabilities to handle load balancing, auto-scaling, and monitoring. Key features like integration with AWS services, such as S3 for storage and RDS for databases, are implemented to demonstrate the platform's flexibility and reliability.

This project highlights the ease of automating deployment pipelines and scaling web applications dynamically to accommodate varying traffic loads. It also demonstrates best practices in managing application environments, monitoring performance using AWS CloudWatch, and ensuring high availability. The outcome is a fully functional, scalable, and cost-effective web application that can adapt to changing user demands with minimal manual intervention.

Introduction

Deploying and managing web applications in a scalable and efficient manner can be challenging, especially when dealing with infrastructure provisioning, scaling, and monitoring. AWS Elastic Beanstalk, a Platform-as-a-Service (PaaS) offering, simplifies this process by providing a managed service that automates the deployment and scaling of web applications while allowing developers to focus on building application functionality.

This project, "Deploy a Scalable Web App with Elastic Beanstalk," explores the deployment of a web application using Elastic Beanstalk, leveraging its features to handle tasks such as load balancing, auto-scaling, and health monitoring. Elastic Beanstalk supports multiple programming languages, including Python, Java, Node.js, and more, making it a versatile tool for various application types. The service also integrates seamlessly with other AWS offerings like Amazon S3 for storage and Amazon RDS for databases, further enhancing the application's functionality and scalability.

Through this project, developers can gain practical experience in deploying a web application that can dynamically scale to accommodate varying traffic loads while maintaining high availability. By automating infrastructure management, Elastic Beanstalk provides an efficient, reliable, and cost-effective way to host modern web applications in the cloud.

Problem Statement :

Developing and deploying a web application that can handle varying traffic loads while ensuring high availability, performance, and scalability is a complex task. Traditional methods of managing infrastructure require significant manual effort, expertise in server configuration, and constant monitoring to maintain optimal performance. These challenges are amplified when applications need to scale dynamically to accommodate unpredictable traffic spikes, often leading to inefficient resource utilization and increased operational costs.

There is a need for a solution that simplifies the deployment and management of web applications while automating tasks like infrastructure provisioning, load balancing, auto-scaling, and monitoring. The solution should also integrate seamlessly with storage and database services to enhance functionality and provide a cost-effective way to host web applications without compromising reliability or scalability.

Objectives :

-**Simplify Deployment**: To deploy a web application using AWS Elastic Beanstalk, minimizing the manual effort required for server configuration and infrastructure management.

-**Enable Scalability**: To leverage Elastic Beanstalk's auto-scaling features to dynamically adjust resources based on application traffic, ensuring optimal performance during peak loads.

-**Ensure High Availability**: To implement load balancing and health monitoring features to maintain the availability and reliability of the application.

-**Integrate AWS Services**: To incorporate additional AWS services, such as Amazon S3 for file storage and Amazon RDS for database management, to enhance application functionality.

-**Optimize Cost Efficiency**: To demonstrate how Elastic Beanstalk can reduce operational costs by automating infrastructure management and efficiently utilizing resources.

-**Improve Application Monitoring**: To utilize AWS CloudWatch for tracking application performance, enabling proactive issue resolution and ensuring system stability.

-**Provide a Scalable Framework**: To deliver a deployment framework that can be easily adapted for future applications requiring similar scalability and reliability.

Scope :

The scope of this project, "Deploy a Scalable Web App with Elastic Beanstalk," encompasses the deployment, management, and scaling of a web application using AWS Elastic Beanstalk. This project focuses on demonstrating how Elastic Beanstalk automates infrastructure provisioning, application deployment, load balancing, auto-scaling, and monitoring to simplify the management of web applications.

The project includes configuring a supported web application (e.g., Python, Node.js, or Java) for deployment and integrating it with AWS services such as Amazon S3 for file storage and Amazon RDS for database management. It also involves implementing load balancers, health checks, and auto-scaling policies to ensure high availability and optimal performance during varying traffic conditions.

The scope extends to configuring AWS CloudWatch for monitoring application health, performance, and usage metrics to enable proactive issue detection and resolution. This project is tailored for developers seeking to deploy scalable, cost-effective, and reliable web applications without the complexity of manually managing infrastructure. It highlights best practices for using Elastic Beanstalk and provides a framework for future projects requiring similar scalability and reliability.

Software Requirements and Specifications :

Software Requirements

-Operating System:Any OS that supports AWS CLI and Elastic Beanstalk CLI (e.g., Windows, macOS, Linux).

-Programming Language:Python, Node.js, Java, or any other language supported by AWS Elastic Beanstalk.

-AWS Tools:AWS CLI (Command Line Interface) for managing AWS resources.
Elastic Beanstalk CLI for deploying and managing the application.

-Web Application Framework:Frameworks such as Flask (Python), Express.js (Node.js), or Spring Boot (Java).

-Integrated Development Environment (IDE):Visual Studio Code, IntelliJ IDEA, PyCharm, or any IDE of choice for development.

-AWS Services:
Elastic Beanstalk: For deployment and scaling of the application.
Amazon S3: For file storage integration (if required).
Amazon RDS: For database management (if required).
AWS CloudWatch: For monitoring application health and performance.

-Dependencies:
Python/Node.js/Java runtime environment and libraries specific to the application.
Package managers like pip (Python) or npm (Node.js).

-Version Control System:
Git for managing application code and deployment pipelines.

Software Specifications :

-Application Architecture: Follows a multi-tier architecture with Elastic Beanstalk managing the application layer, and optional integration with RDS for the database layer.

-Scalability: The application must support auto-scaling to handle varying traffic loads.

-Load Balancing:Integrated load balancers to distribute traffic across multiple instances for high availability.

-Security:IAM roles and policies for secure access to AWS resources. HTTPS support for secure communication.

-Monitoring: CloudWatch configured to track key metrics like CPU utilization, memory usage, and application health.

-Deployment: Deploy using Elastic Beanstalk CLI or the AWS Management Console with zero-downtime updates.

-Data Storage:
Amazon RDS for relational database management (optional).
Amazon S3 for static file storage or backups (optional).

-Environment Management:
Multiple environments (e.g., development, staging, and production) supported by Elastic Beanstalk.

Project Plan :

Software Development Life-cycle :

1. Requirement Gathering and Analysis

Objective: Understand the needs of the web application and define the project scope.

Activities:

-Identify application requirements, including supported programming languages, frameworks, and integrations with AWS services.

-Determine the expected traffic load, scalability needs, and monitoring requirements.

-Document functional and non-functional requirements, such as performance, availability, and security standards.

2. System Design

Objective: Design the architecture and workflow of the application deployment.

Activities:

-Create an architecture diagram involving Elastic Beanstalk, load balancers, auto-scaling groups, and optional AWS services like S3 and RDS.

-Define the deployment strategy (rolling updates, blue/green deployment).

-Design access control policies using IAM roles and security groups.

-Plan the logging and monitoring setup with AWS CloudWatch.

3. Implementation (Development)

Objective: Develop the web application and configure Elastic Beanstalk for deployment.

Activities:

-Build the web application using the chosen programming language and framework.

-Package the application for deployment (e.g., zip files with dependencies).

-Configure the Elastic Beanstalk environment using the CLI or AWS Management Console.

-Set up optional integrations with S3 (for static file storage) and RDS (for database).

-Write scripts or automation pipelines for deployment.

4. Testing

Objective: Validate the application’s functionality, scalability, and performance.

Activities:

-Unit Testing: Test individual application components to ensure correctness.

-Integration Testing: Verify interactions between Elastic Beanstalk, S3, RDS, and other AWS services.

-Performance Testing: Test the application under various traffic loads to ensure auto-scaling works as expected.

-Security Testing: Check IAM roles, access policies, and HTTPS configurations for vulnerabilities.

-Load Testing: Simulate high traffic scenarios to validate the load balancer and scalability setup.

5. Deployment

Objective: Deploy the web application to the production environment.

Activities:

-Use Elastic Beanstalk CLI or the AWS Management Console for deployment.

-Configure rolling updates or blue/green deployment for minimal downtime.

-Ensure the environment is secure with proper access controls and encryption.

-Validate deployment by testing the live environment.

6. Monitoring and Maintenance

Objective: Ensure the application remains operational, scalable, and cost-efficient over time.

Activities:

-Monitor application performance and health using AWS CloudWatch metrics.

-Update the application or environment as required (e.g., adding new features or patching vulnerabilities).

-Optimize auto-scaling and lifecycle policies to control costs.

-Address system alerts or issues proactively to ensure high availability.

7. Evaluation and Feedback

Objective: Assess the project's success and identify areas for improvement.

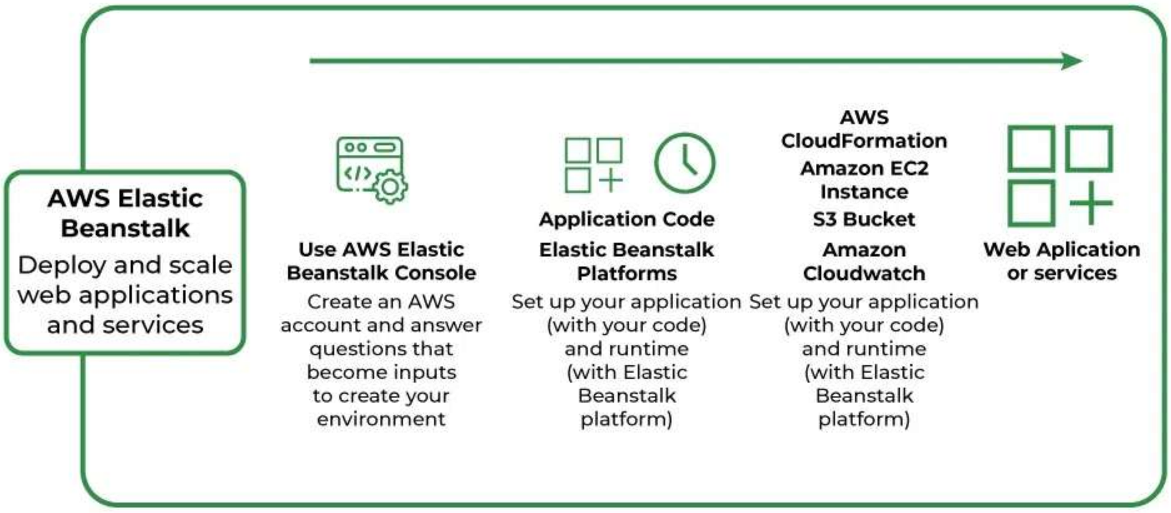
Activities:

-Gather feedback from stakeholders or users about application performance and functionality.

-Review metrics and logs to evaluate the efficiency of the deployment and scaling strategies.

-Plan future updates or enhancements based on feedback and performance data.

Architecture Diagram :


AWS Elastic Beanstalk
Deploy and scale web applications and services
Use AWS Elastic Beanstalk Console
Create an AWS account and answer questions that become inputs to create your environment
Application Code
Elastic Beanstalk Platforms
Set up your application (with your code) and runtime (with Elastic Beanstalk platform)
AWS CloudFormation
Amazon EC2 Instance
S3 Bucket
Amazon Cloudwatch
Set up your application (with your code) and runtime (with Elastic Beanstalk platform)
Web Aplication or services

The architectural diagram for "Deploy a Scalable Web App with Elastic Beanstalk" illustrates the flow of components in the AWS ecosystem. At the core, AWS Elastic Beanstalk manages the deployment and operation of the web application. It automatically provisions resources, including EC2 instances, within an auto-scaling group to ensure scalability based on traffic demands.
A load balancer distributes incoming traffic across multiple EC2 instances to maintain high availability and optimize performance. For data storage, Amazon RDS provides relational database support, while Amazon S3 is used for storing static assets or backups.
Monitoring and logging are facilitated by AWS CloudWatch, which tracks application health, performance metrics, and system logs. Security is ensured through IAM roles and security groups, controlling access to resources. The architecture is designed for reliability, scalability, and cost efficiency while maintaining a seamless user experience.

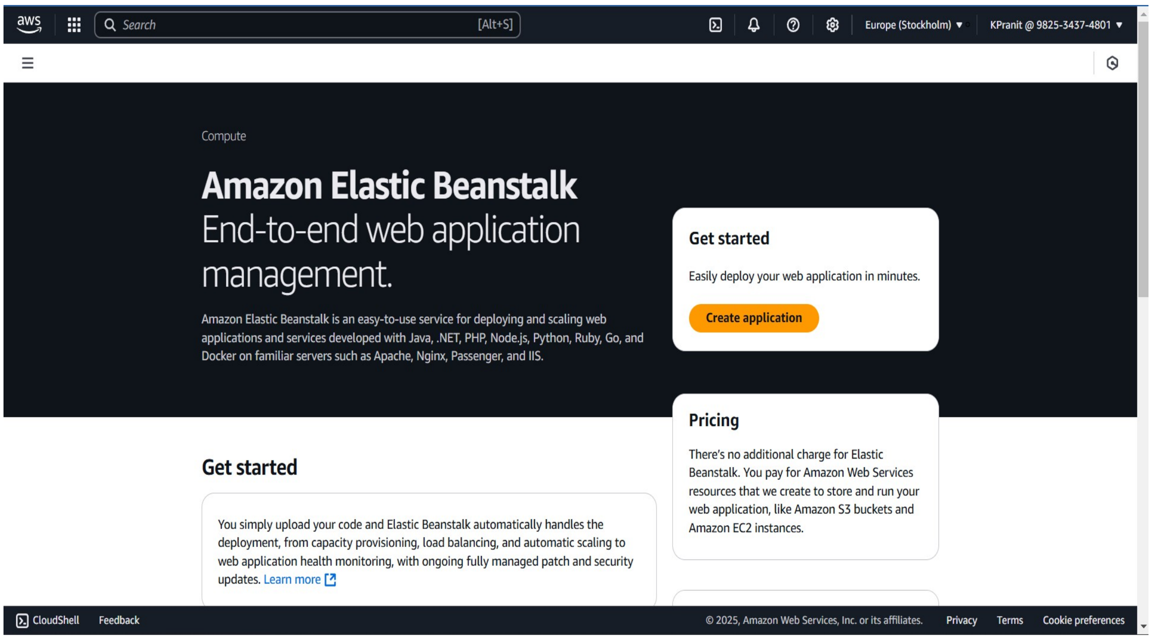
Result :

The deployment of a scalable web application using AWS Elastic Beanstalk successfully achieved the objectives of simplicity, scalability, and high availability. The application was deployed seamlessly, with Elastic Beanstalk automating the provisioning of infrastructure, including EC2 instances, load balancers, and auto-scaling groups.

The system dynamically adjusted resources to handle varying traffic loads, ensuring consistent performance during peak periods. Integration with Amazon RDS and S3 enhanced the functionality of the application, providing reliable database support and secure storage for static assets. Monitoring through AWS CloudWatch enabled real-time performance tracking and proactive issue resolution. Overall, the project demonstrated how Elastic Beanstalk simplifies the deployment and management of scalable web applications, reducing manual effort while maintaining cost efficiency and operational reliability.

Screenshots :

Step 01 : Go to the Elastic Beanstalk dashboard in AWS.



Amazon Elastic Beanstalk
End-to-end web application management.

Amazon 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.

Get started

Easily deploy your web application in minutes.

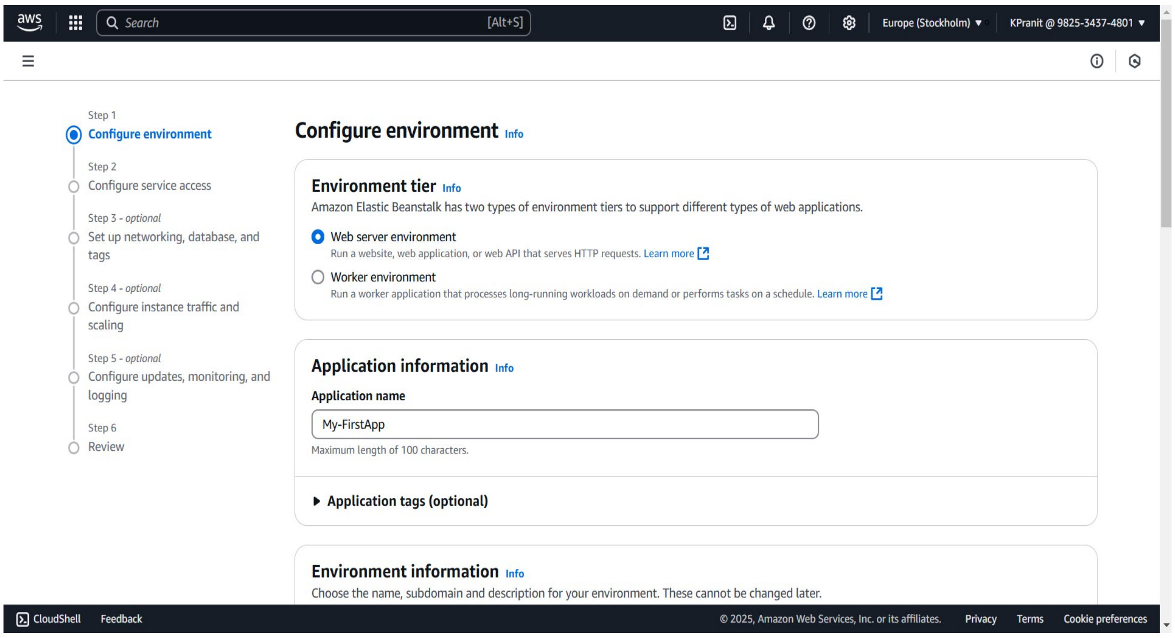
Pricing

There's no additional charge for Elastic Beanstalk. You pay for Amazon Web Services resources that we create to store and run your web application, like Amazon S3 buckets and Amazon EC2 instances.

Get started

You simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, and automatic scaling to web application health monitoring, with ongoing fully managed patch and security updates. Learn more

Step 02 : By clicking on the creating the application we come to the configure environment section.



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-FirstApp

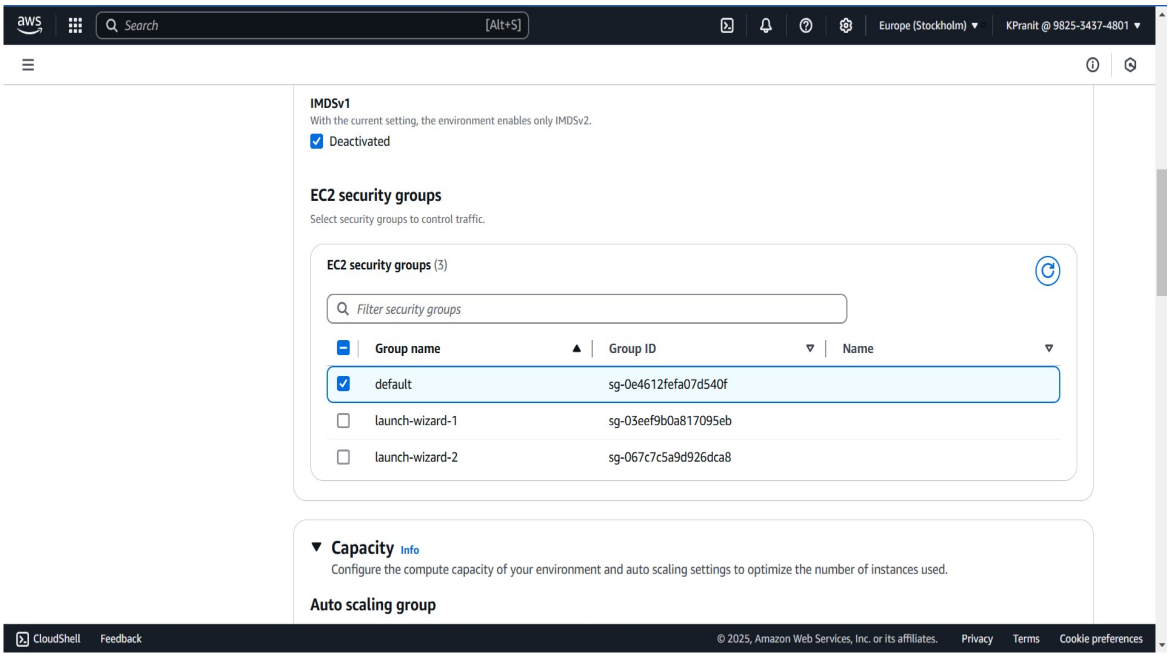
Maximum length of 100 characters.

▸ Application tags (optional)

Environment information Info

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

Step 03 : Setup the service access section according to you.


IMDSv1

With the current setting, the environment enables only IMDSv2.

☑ Deactivated

EC2 security groups

Select security groups to control traffic.

EC2 security groups (3)

	Group name	Group ID	Name
☑	default	sg-0e4612fefa07d540f	
☐	launch-wizard-1	sg-03eef9b0a817095eb	
☐	launch-wizard-2	sg-067c7c5a9d926dca8	

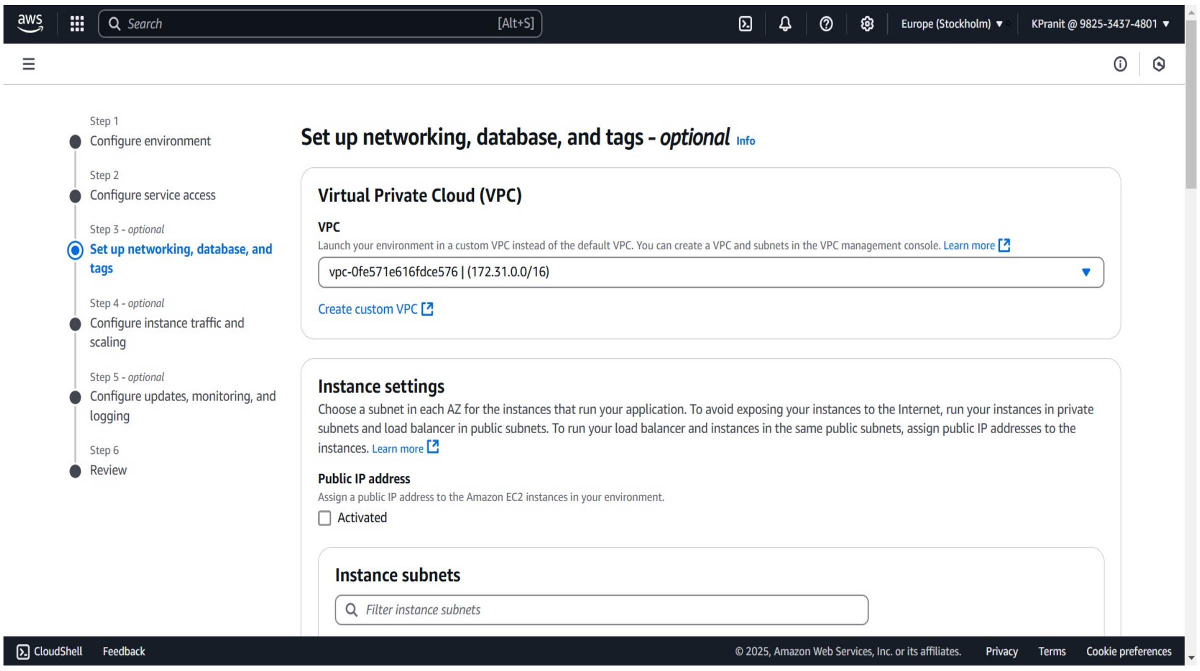
Capacity Info

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

Auto scaling group

© 2025, Amazon Web Services, Inc. or its affiliates.

Step 04 : Set up the networking ,database and tags section to proceed further.


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

Set up networking, database, and tags - *optional* Info

Virtual Private Cloud (VPC)

VPC

Launch your environment in a custom VPC instead of the default VPC. You can create a VPC and subnets in the VPC management console. Learn more

vpc-0fe571e616fdce576 | (172.31.0.0/16)

Create custom VPC

Instance settings

Choose a subnet in each AZ for the instances that run your application. To avoid exposing your instances to the Internet, run your instances in private subnets and load balancer in public subnets. To run your load balancer and instances in the same public subnets, assign public IP addresses to the instances. Learn more

Public IP address

Assign a public IP address to the Amazon EC2 instances in your environment.

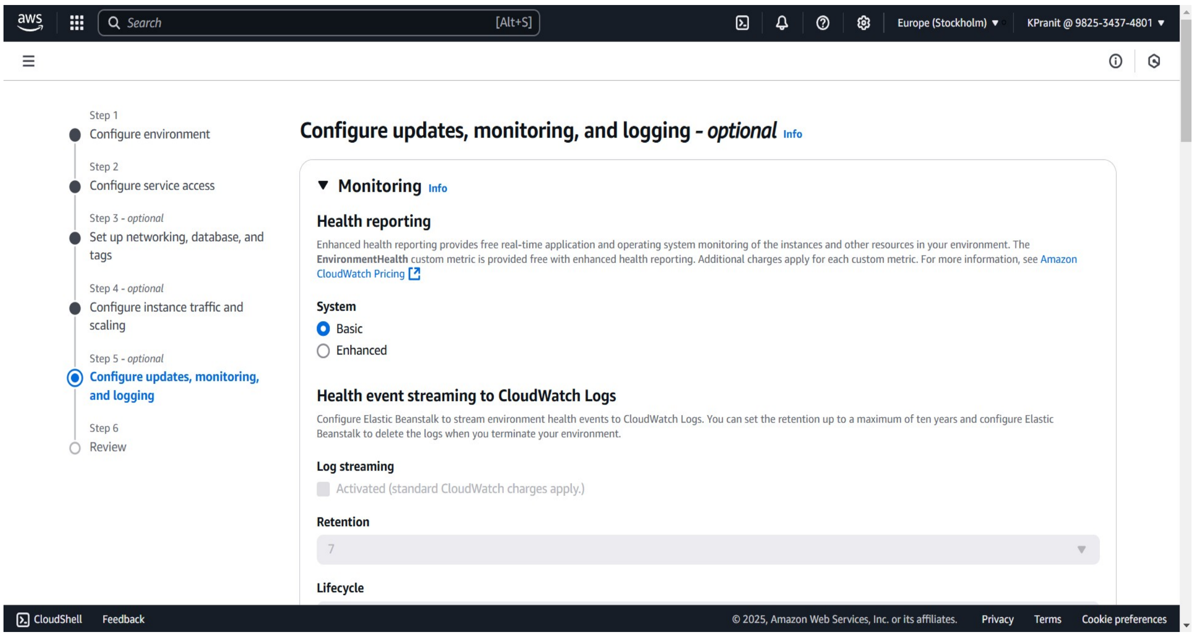
☐ Activated

Instance subnets

© 2025, Amazon Web Services, Inc. or its affiliates.

PRANIT KOLAMKAR

Step 05 : Setup the configure updates ,monitoring and logging section.



Configure updates, monitoring, and logging - *optional* Info

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

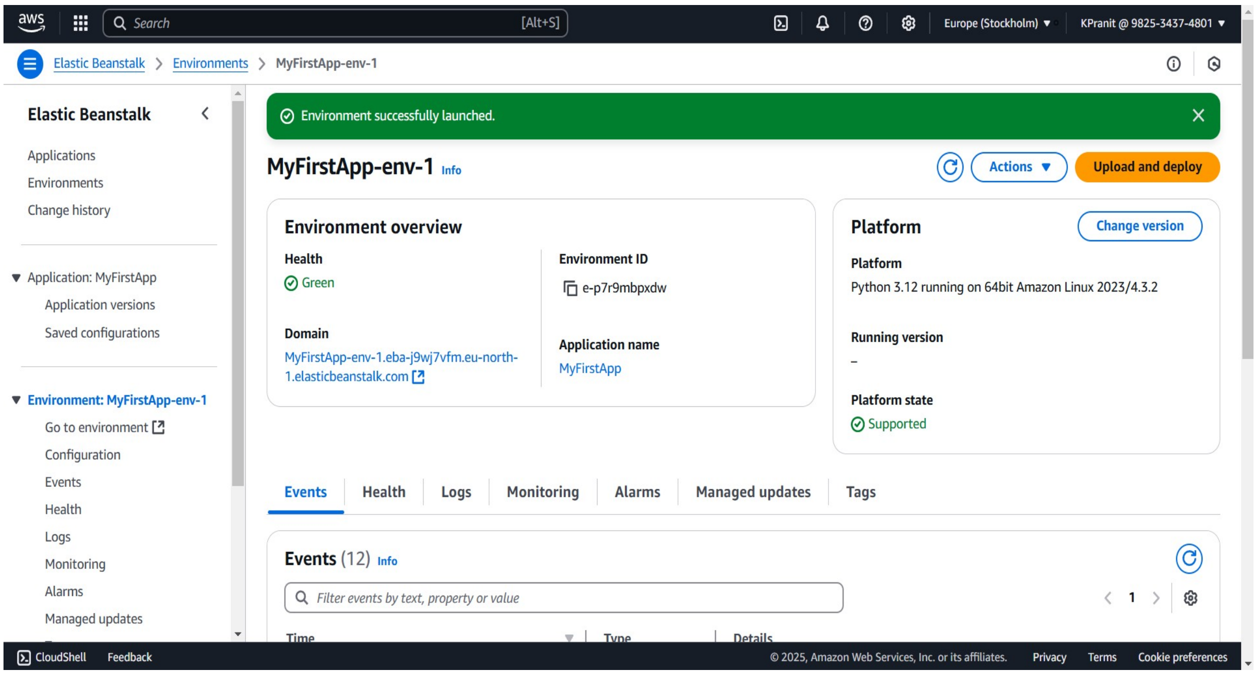
☐ Activated (standard CloudWatch charges apply.)

Retention

7

Lifecycle

Step 06 : After reviewing all the sections launch the environment securely it will take few minutes so be patient.



Environment successfully launched.

MyFirstApp-env-1 Info

Environment overview

Health: Green

Environment ID: e-p7r9mbpxdw

Domain: MyFirstApp-env-1.eba-j9wj7vfm.eu-north-1.elasticbeanstalk.com

Application name: MyFirstApp

Platform

Platform: Python 3.12 running on 64bit Amazon Linux 2023/4.3.2

Running version: –

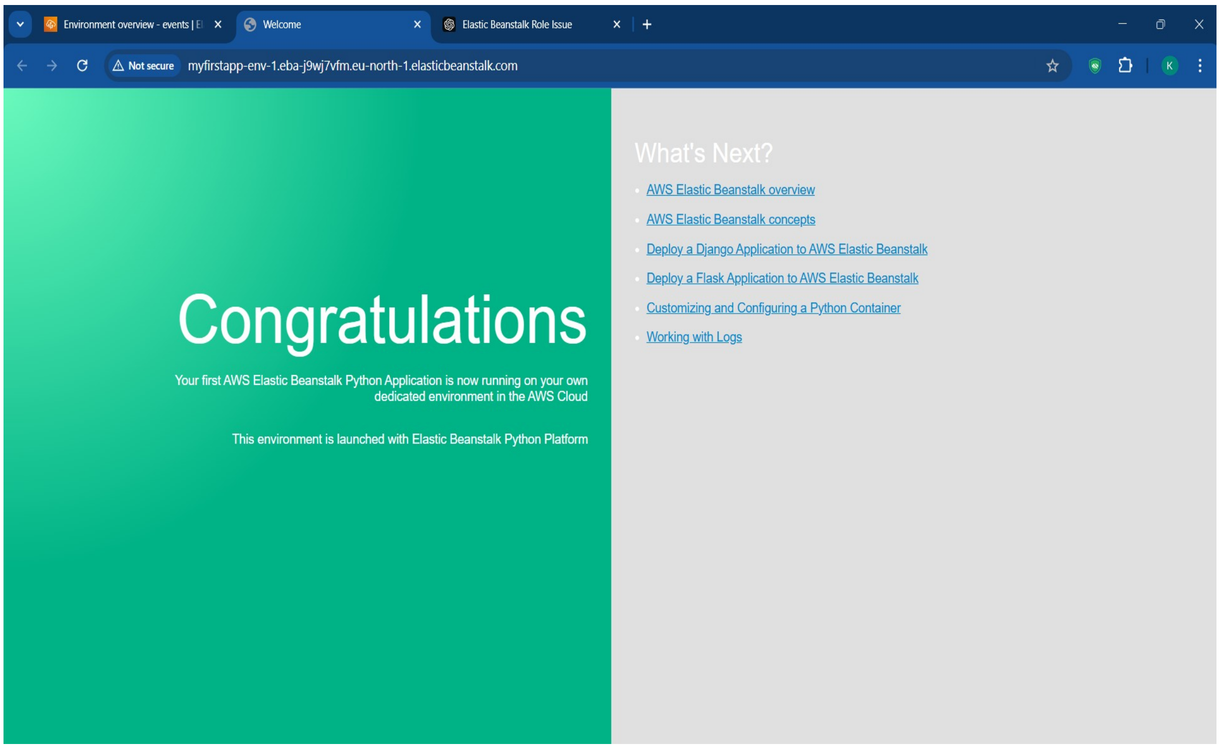
Platform state: Supported

Events (12) Info

Filter events by text, property or value

PRANIT KOLAMKAR

Step 07 : After deploying the application we can access it through the URL provided in the domain section.


Environment overview - events | E
Welcome
Elastic Beanstalk Role Issue
Not secure myfirstapp-env-1.eba-j9wj7vfm.eu-north-1.elasticbeanstalk.com

Congratulations

Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud

This environment is launched with Elastic Beanstalk Python Platform

What's Next?

- AWS Elastic Beanstalk overview
- AWS Elastic Beanstalk concepts
- Deploy a Django Application to AWS Elastic Beanstalk
- Deploy a Flask Application to AWS Elastic Beanstalk
- Customizing and Configuring a Python Container
- Working with Logs

Conclusion :

The project successfully demonstrated the deployment of a scalable and reliable web application using AWS Elastic Beanstalk. By automating infrastructure management, Elastic Beanstalk simplified the process of provisioning, scaling, and monitoring resources, ensuring high availability and performance. Integrations with AWS services like RDS and S3 enhanced functionality, while CloudWatch provided effective monitoring. This project highlights the efficiency and cost-effectiveness of Elastic Beanstalk as a platform for deploying modern web applications with minimal manual intervention.