# SVKM'S NMIM'S Nilkamal School of Mathematics, Applied Statistics & Analytics
# Master of Science (Data Science)

Practical-2 Platform as a service using AWS.

**Date:-23/01/2024 Submission Date:- 30/01/2024**

**Writeup:-**

- **Platform as a service**

PaaS, or Platform-as-a-Service, is a cloud computing model that provides customers a complete cloud platform—hardware, software, and infrastructure—for developing, running, and managing applications without the cost, complexity, and inflexibility that often comes with building and maintaining that platform on-premises.

The PaaS provider hosts everything—servers, networks, storage, operating system software, databases, development tools—at their data center. Typically customers can pay a fixed fee to provide a specified amount of resources for a specified number of users, or they can choose 'pay-as-you-go' pricing to pay only for the resources they use. Either option enables PaaS customers to build, test, deploy run, update and scale applications more quickly or inexpensively they could if they had to build out and manage their own on-premises platform.

Every leading cloud service provider—including Amazon Web Services (AWS), Google Cloud, IBM Cloud and Microsoft Azure—has its own PaaS offering. Popular PaaS solutions are also available as open source projects (e.g. Apache Stratos, Cloud Foundry) or from software ventors (e.g. Red Hat OpenShift and Salesforce Heroku).

- **Elastic Beanstalk**

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them. With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health

monitoring. Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application. You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or eb, a high-level CLI designed specifically for Elastic Beanstalk.

- **Components of beanstalk**

The five main components that make Elastic Beanstalk are:-

### Deployment

Elastic Beanstalk, for the most part, simplifies the process of deploying an application on the Amazon cloud. The service allows developers the ability to upload and manage different versions of their apps, and switch between them in different environments like development, test, and production.

### Application

An application in Elastic Beanstalk is basically a collection of environments, versions, and everything else related to them, like events. In other words, an Elastic Beanstalk application is conceptually similar to a folder. Most users normally create a separate EB application for each of their applications, and although this is not required, it does help streamline management.

### Version

A version is the deployable code of an application. Depending on your programming platform of choice, you will have a file, or a set of files that you upload, with a label and description. You can then see where it is deployed, in which environment, and even download the file or files, if needed.

### Environment

As you may have guessed, an environment is a deployed version on specific instances, load balancers, and scaling groups, etc. A typical workflow is creating one environment for testing, and another for production. Though you can, of course, create as many as you need, as much as your budget allows. Amazon provides access to your environment via a specific URL, and provide different health status so you can quickly get an idea of how things are up there. Green is okay, yellow when your environment has not responded within the last 5 minutes, red if it hasn't responded for more than 5 minutes, and gray, unknown.

## Events

Events tell you what is going on with your environments. They are either informative, warnings, or errors, letting you know details like when an environment successfully launches, or an instance is close to utilizing its resources. You can view the events in a web console, or have them sent to you via email.

- ## IAM

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform.

IAM gives you the following features:

- **Shared access to your AWS account**

You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.

- **Granular permissions**

You can grant different permissions to different people for different resources. For example, you might allow some users complete access to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Redshift, and other AWS services. For other users, you can allow read-only access to just some S3 buckets, or permission to administer just some EC2 instances, or to access your billing information but nothing else.

- **Secure access to AWS resources for applications that run on Amazon EC2**

You can use IAM features to securely provide credentials for applications that run on EC2 instances. These credentials provide permissions for your application to access other AWS resources. Examples include S3 buckets and DynamoDB tables.

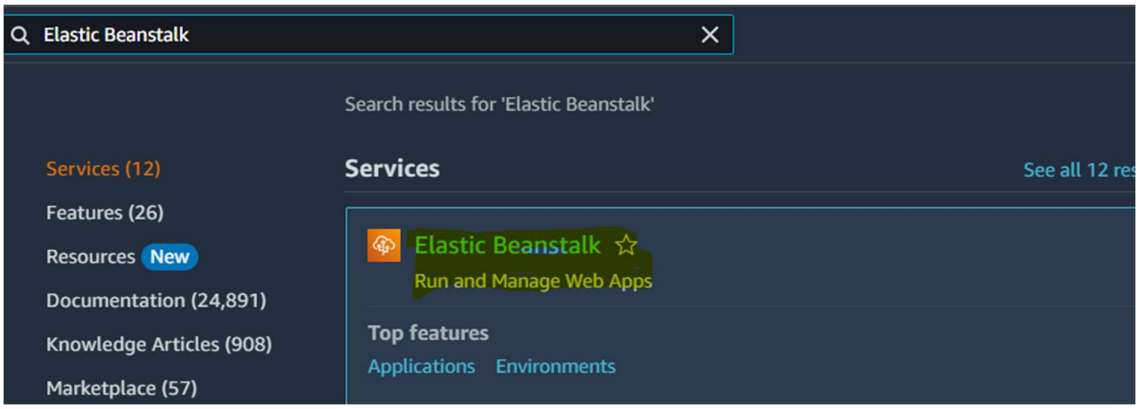
- **Multi-factor authentication (MFA)**

You can add two-factor authentication to your account and to individual users for extra security. With MFA you or your users must provide not only a password or access key to work with your account, but also a code from a specially configured device. If you already use a FIDO security key with other services, and it has an AWS supported configuration, you can use WebAuthn for MFA security.

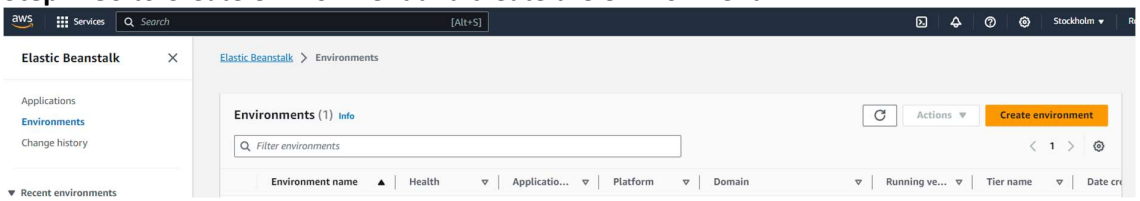
- **Implement PAAS using elastic beanstalk for the following.**

1. **Server**

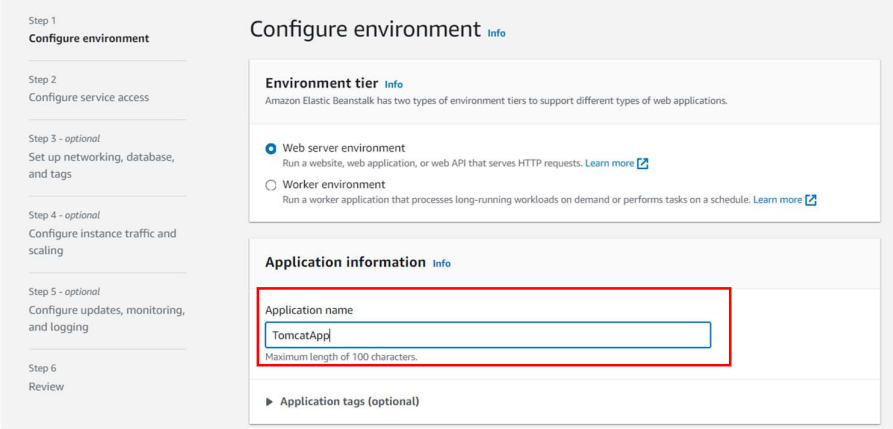
**Step1:** Login to AWS and search for Elastic Beanstalk



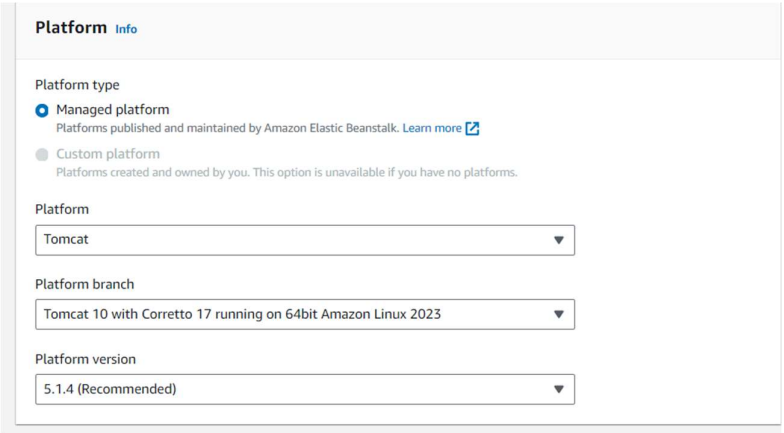

**Step2:** Go to Create environment and create the environment ->



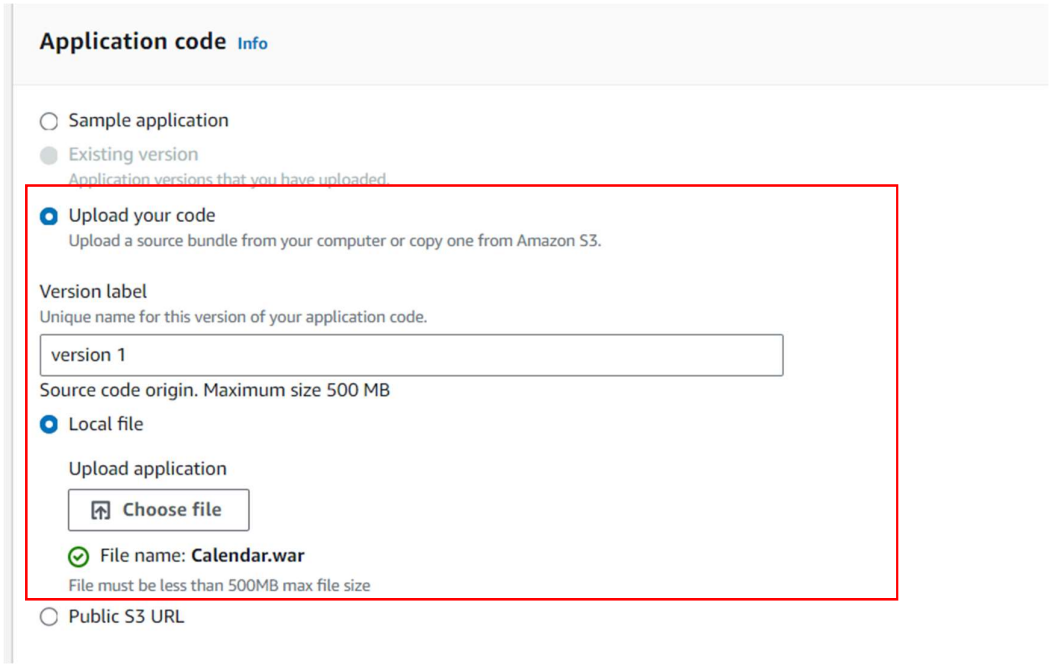

Insert application name ->



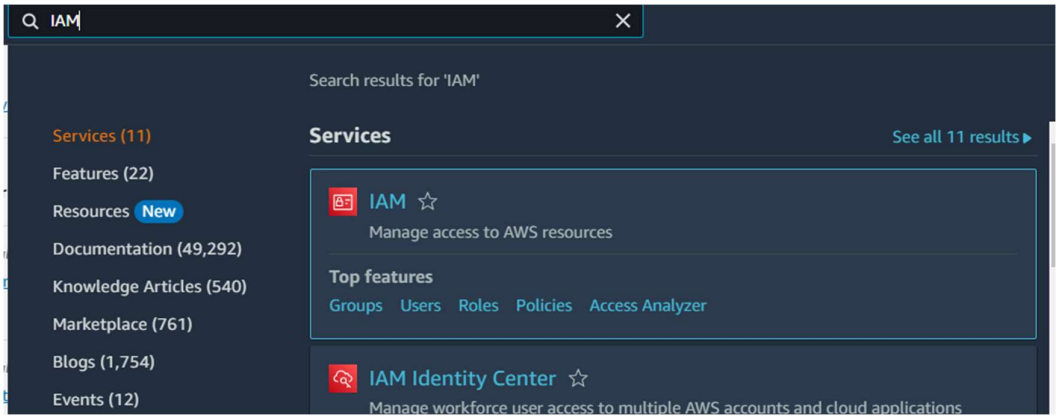

Select preferred platform ->

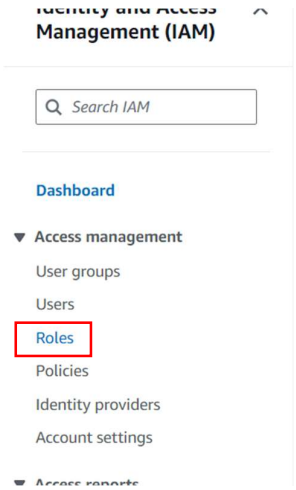



Upload the code file and Type the version label ->

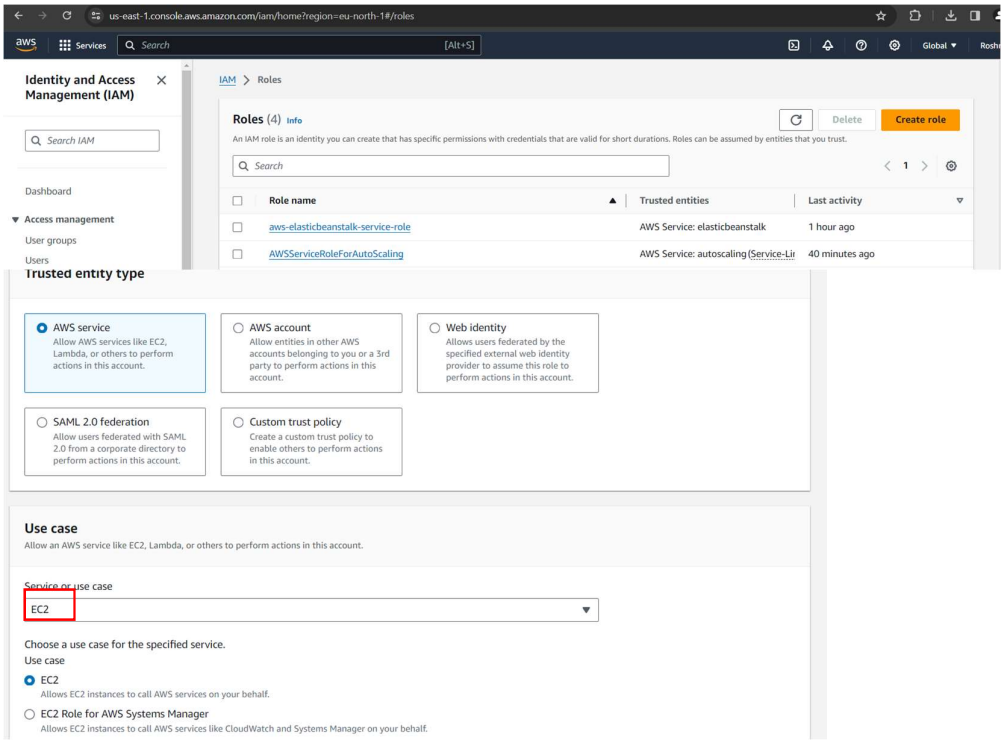



**Step3:** Search for IAM and open in new tab

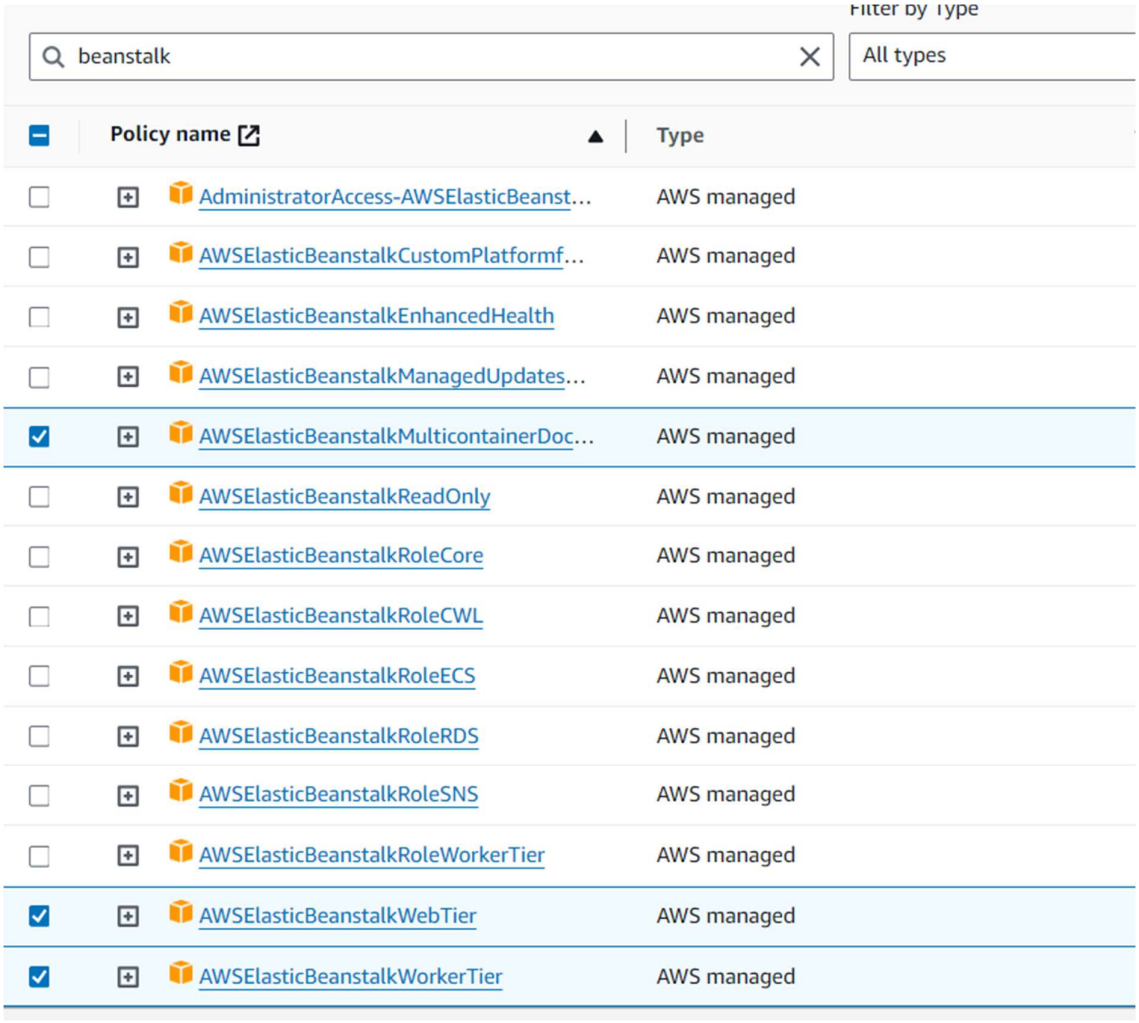




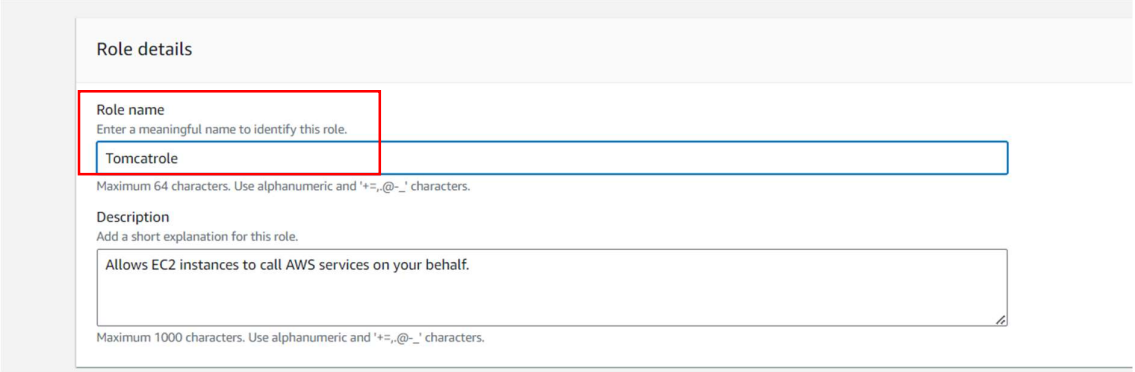

Go to Create Role and select EC2 -> Next

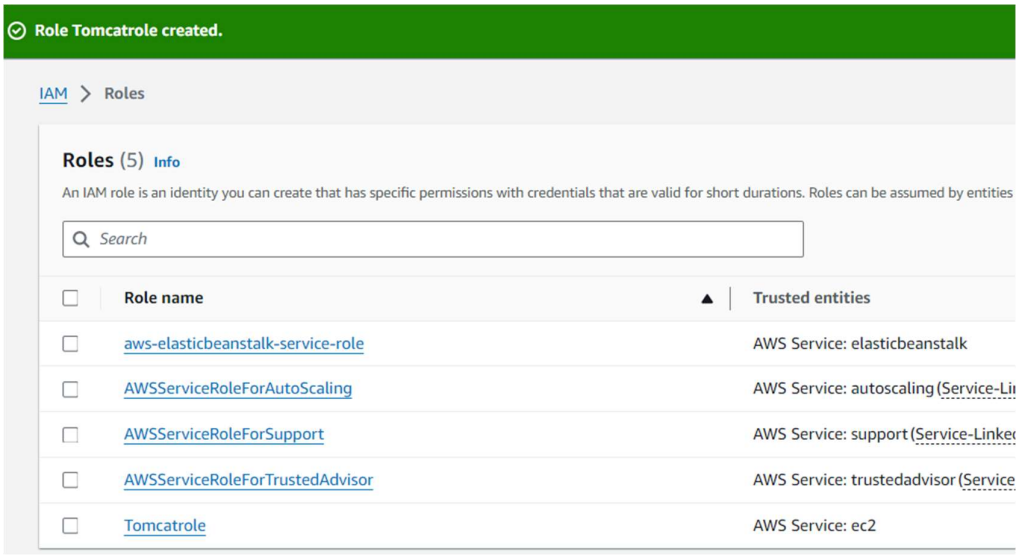


Search for beanstalk and select the options show below

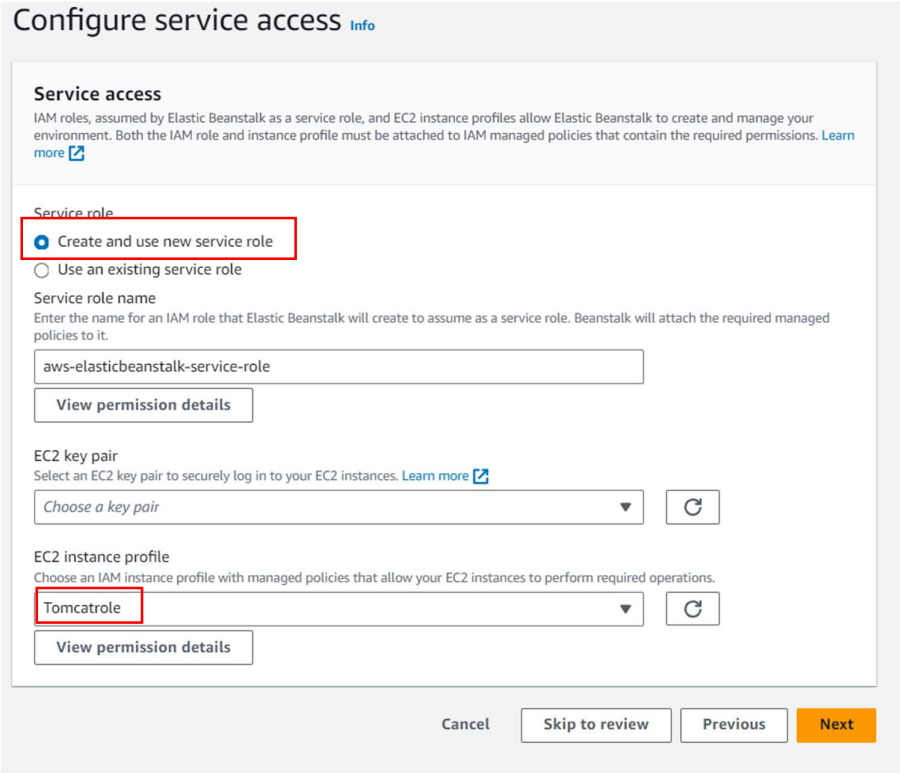



Insert the role name -> create role

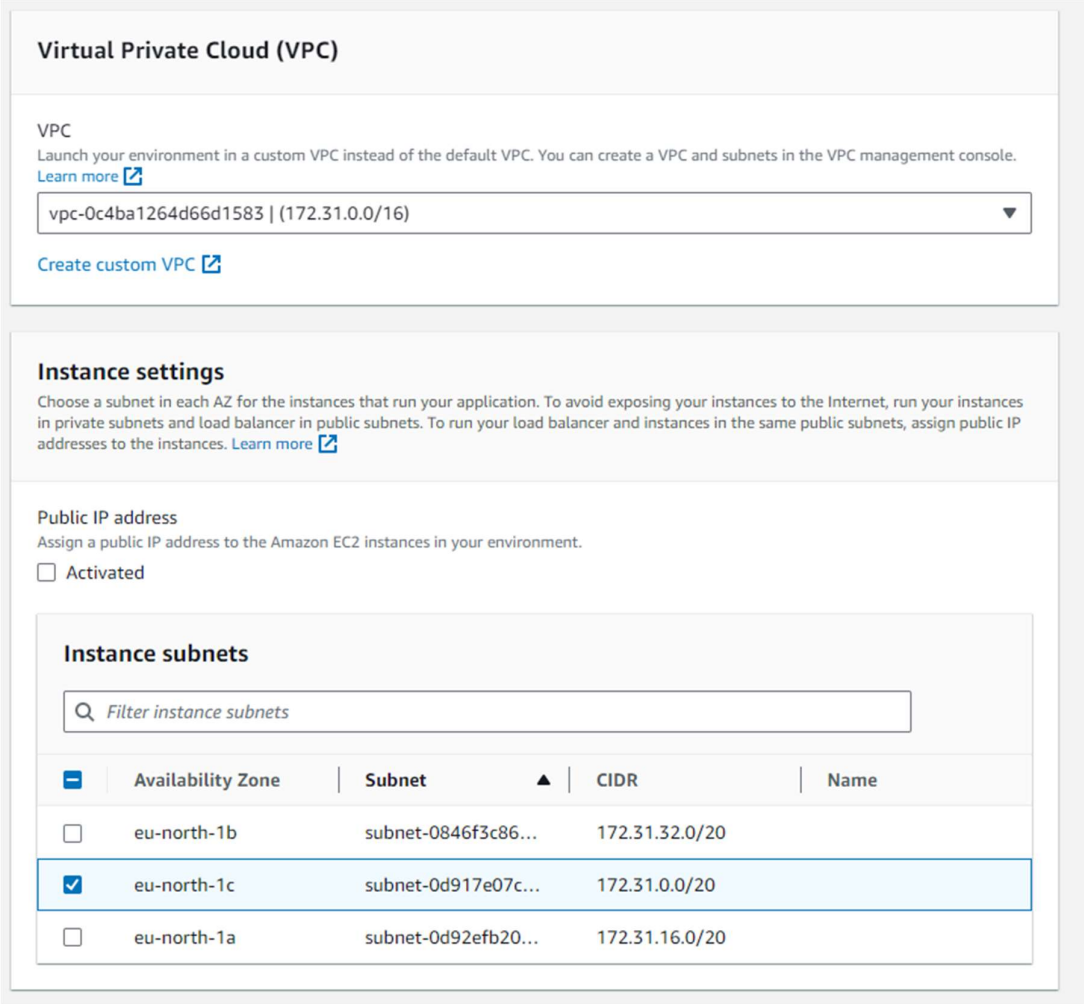



**Step4:** Continue step 2 after creating environment  Configure service access -> select create and use new service role and select the role created in EC2 instance profile -> Next

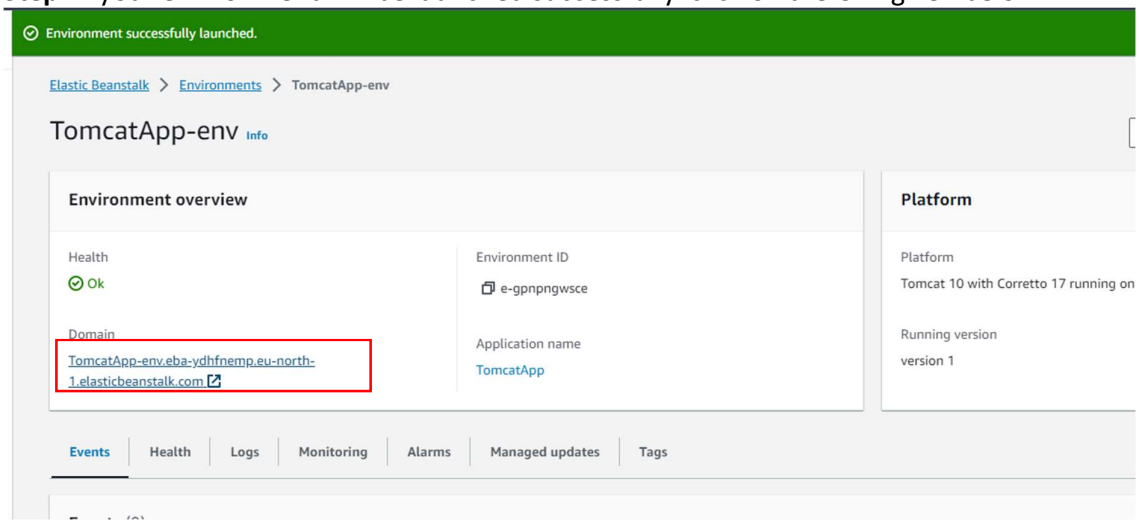


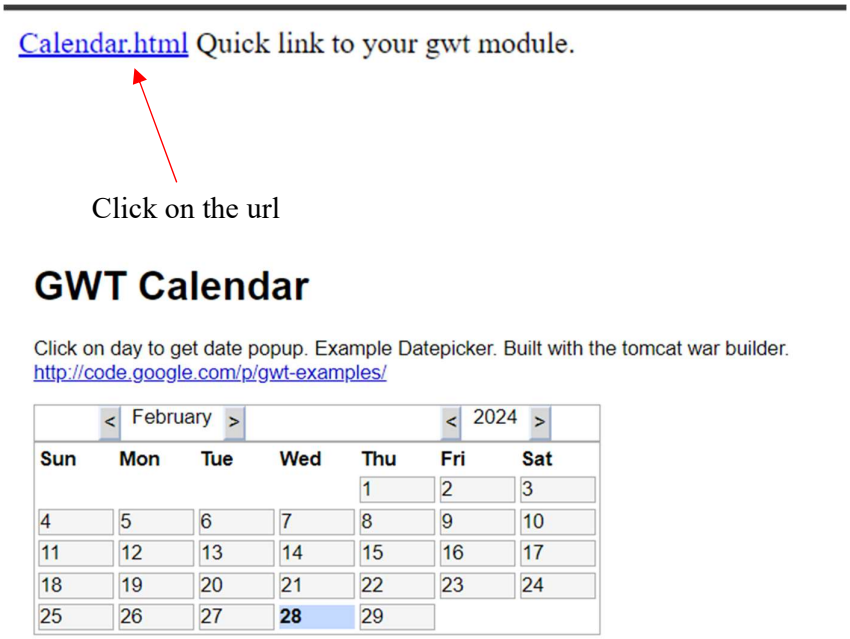

**Step5:** Select the options given below -> Keep the Database settings as default -> Next




**Step6:** Keep the Configure instance traffic and scaling – optional as default then click on skip to review and then submit is there is any error go to step 4 and select the option as Use an existing service role

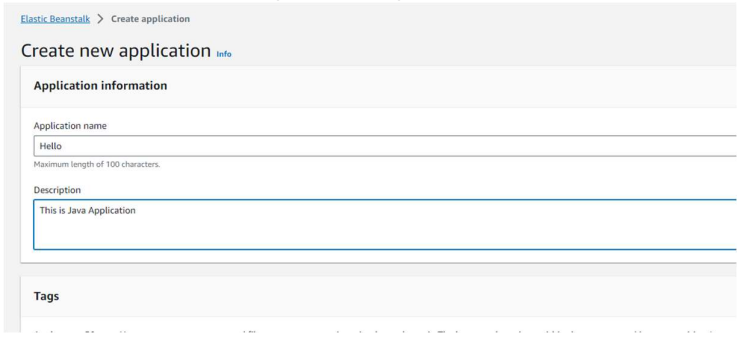
**Step7`:** your environment will be launched successfully click on the URL given below



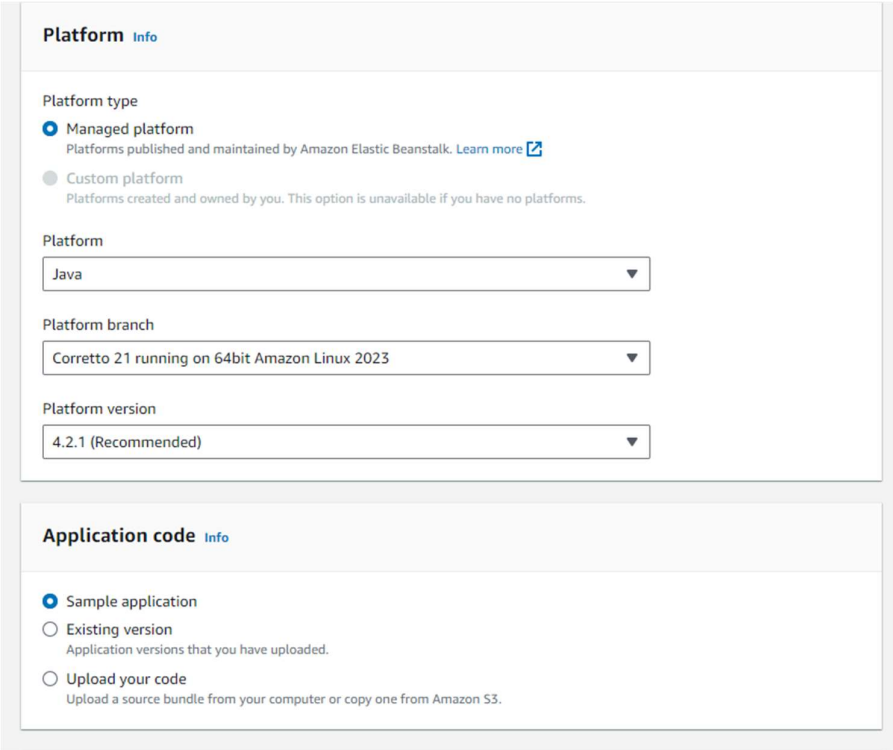



## 2. Java

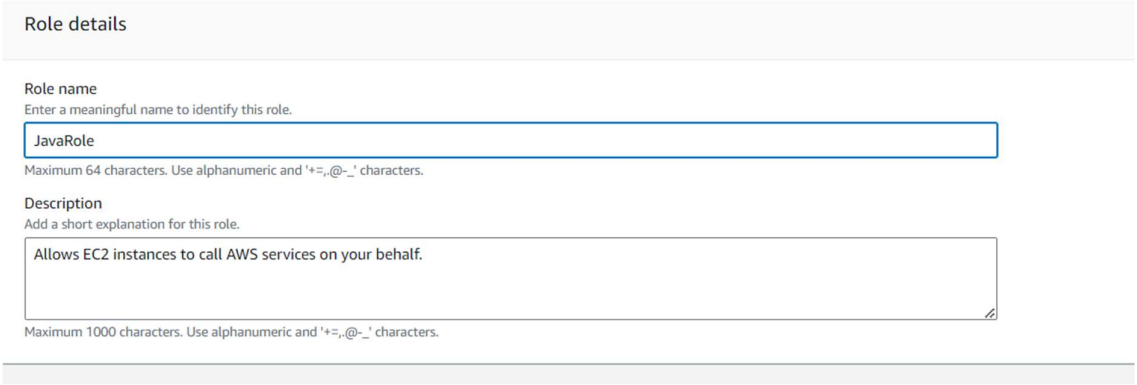
### Follow the same steps till step 2 of Server



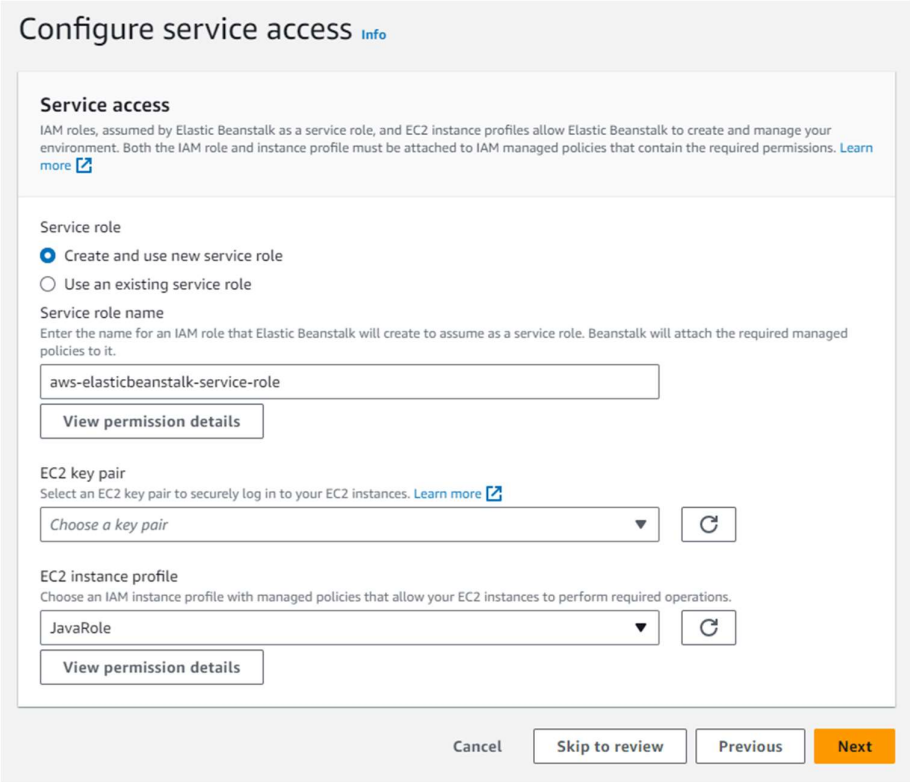
**Step 1: Only Change the platform as Java**
**And keep the Application code to Sample application ->Next**




**Step2:- Follow Step 3 of Server till the role details -> enter the role name**

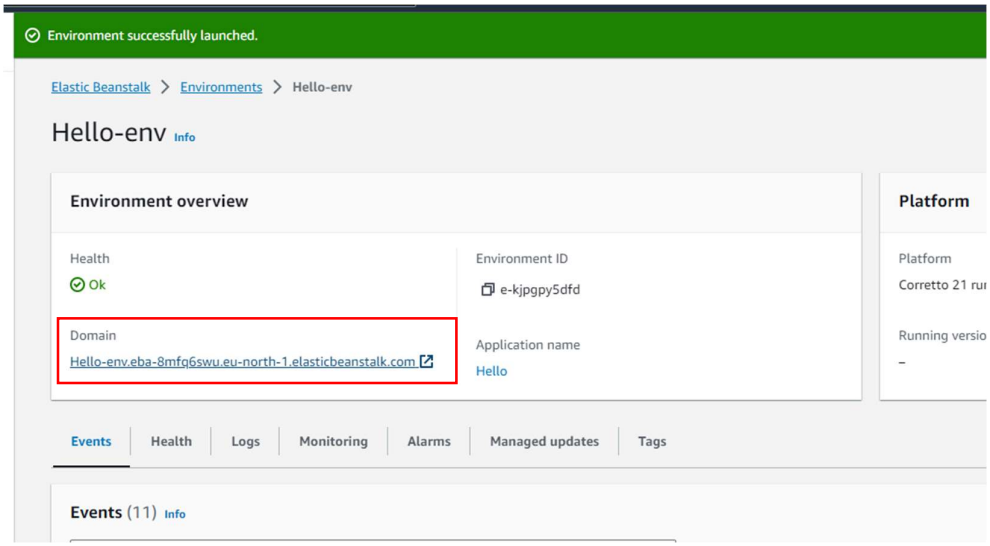


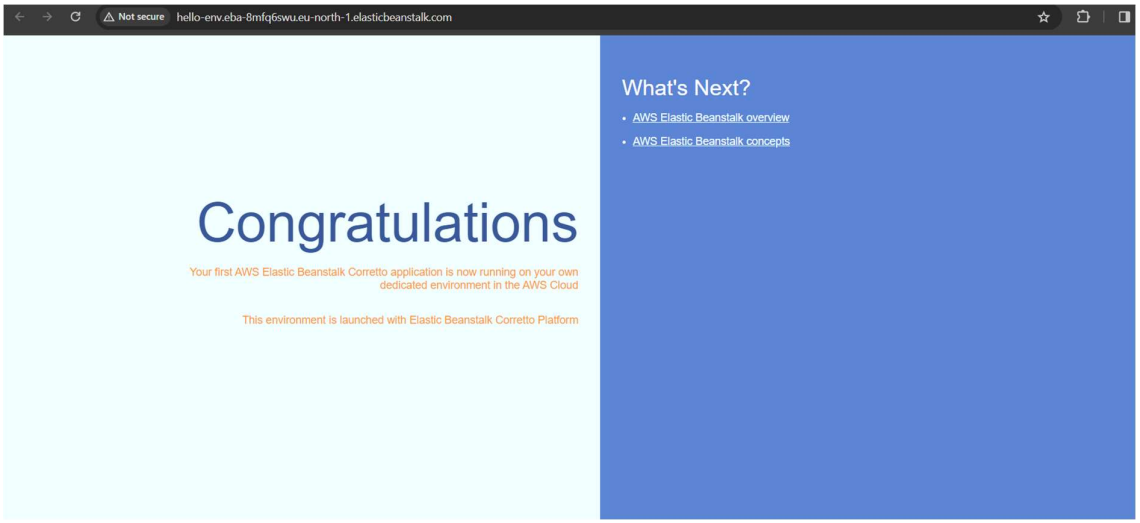
**Step3:** Continue step 2 after creating environment  Configure service access -> select create and use new service role and select the role created in EC2 instance profile -> Next




**Step4:- Follow the steps of step 5 and 6 of Server**

**Step 5:-** your environment will be launched successfully click on the URL given below




**3. Python**

**Follow the same steps given above**

**4. Node.js**

**Follow the same steps given above**