SVKM'S NMIMS Nilkamal School of Mathematics, Applied Statistics & Analytics

Master of Science (Statistics & Data Science)

Practical-5 Platform as a service using AWS

Name: Shreya Nadkarni

SAP ID: 86062300047

Roll No: A042

1)Platform as a Service (PaaS)

Platform as a Service (PaaS) is a cloud computing model that provides a ready-to-use environment for developers to build, run, and manage applications without worrying about the underlying infrastructure. PaaS offers pre-configured tools, libraries, and frameworks that simplify the development and deployment process, helping developers focus on coding and application logic rather than server management and infrastructure maintenance. Examples of PaaS include AWS Elastic Beanstalk, Google App Engine, and Microsoft Azure App Service.

2)AWS Elastic Beanstalk

AWS Elastic Beanstalk is an easy-to-use PaaS solution that automates the deployment, scaling, and management of applications in the AWS Cloud. With Elastic Beanstalk, developers can deploy applications written in popular programming languages (like Python, Java, Node.js, PHP, Ruby, and more) by simply uploading code. Beanstalk handles tasks such as provisioning the necessary infrastructure, managing load balancing, auto-scaling, monitoring, and more. It’s particularly useful for fast deployments and quick iterations while maintaining flexibility over the underlying AWS services.

3)Components of Elastic Beanstalk

Elastic Beanstalk comprises several key components that work together to create and manage the environment in which applications run. These include:

1. Application: A container that holds different environments. Each environment runs a version of the application, allowing for testing and production environments within the same application container.

2. Environment: The specific infrastructure resources and settings used to run an application version. Each environment is associated with a URL, load balancer, and other resources that Elastic Beanstalk configures based on the application's needs.

3. Environment Configuration: Settings applied to the environment, such as instance type, security groups, scaling policies, and software versions. These configurations can be updated as the application evolves.

4. Application Version: A deployable version of the code. Multiple versions can be stored and rolled back if necessary.

5. Environment Tier: The type of environment, either Web Server (used for web applications) or Worker (for asynchronous, background tasks).

6. Elastic Beanstalk CLI (EB CLI): A command-line tool that facilitates the management of applications and environments directly from the command line.

4) IAM (Identity and Access Management)

AWS Identity and Access Management (IAM) is a security service that enables users to securely control access to AWS services and resources. With IAM, users can manage permissions and access controls for various AWS

resources, ensuring that only authorized individuals or applications can perform specific actions. Key IAM components include:

1. Users: Individual user accounts representing a person or service.

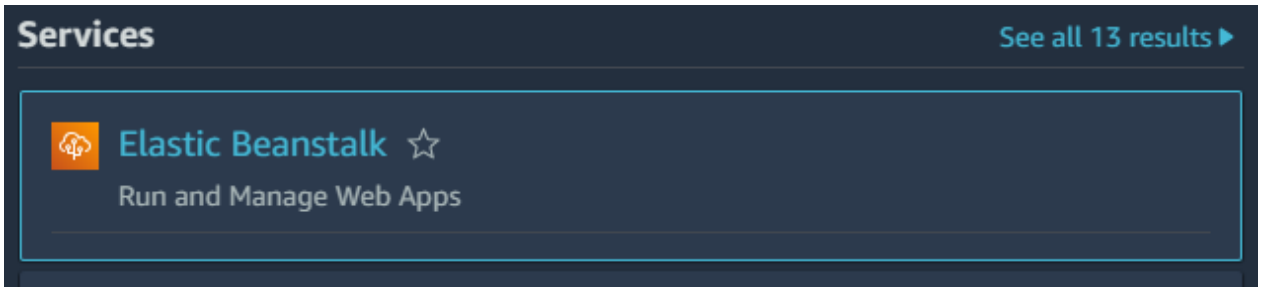
2. Groups: Collections of users to simplify permission management. Permissions applied to a group apply to all members.

3. Roles: Roles allow users or services to assume permissions temporarily, ideal for cross-account or temporary access.

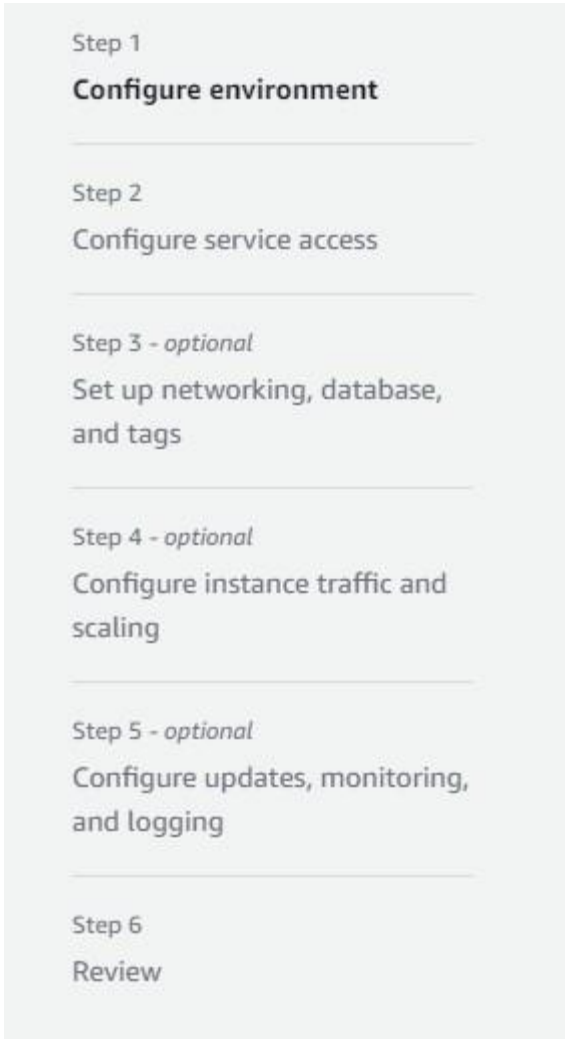
4. Policies: JSON-based documents that define permissions and can be attached to users, groups, or roles to control access.

IAM plays a crucial role in Elastic Beanstalk by securing access to application environments and associated resources, allowing developers to manage permissions granularly.

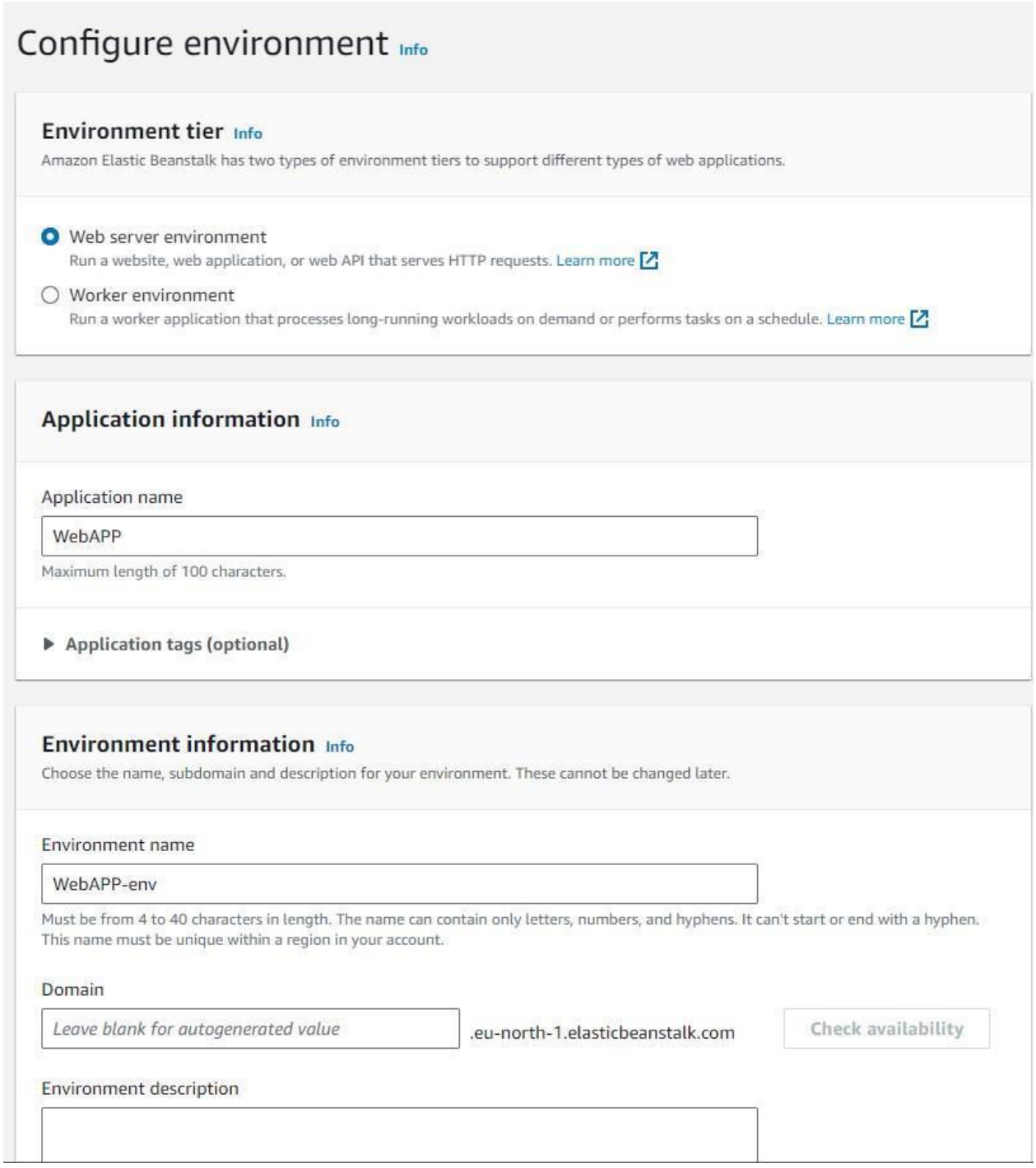
1. We'll be using Elastic Beanstalk - Python



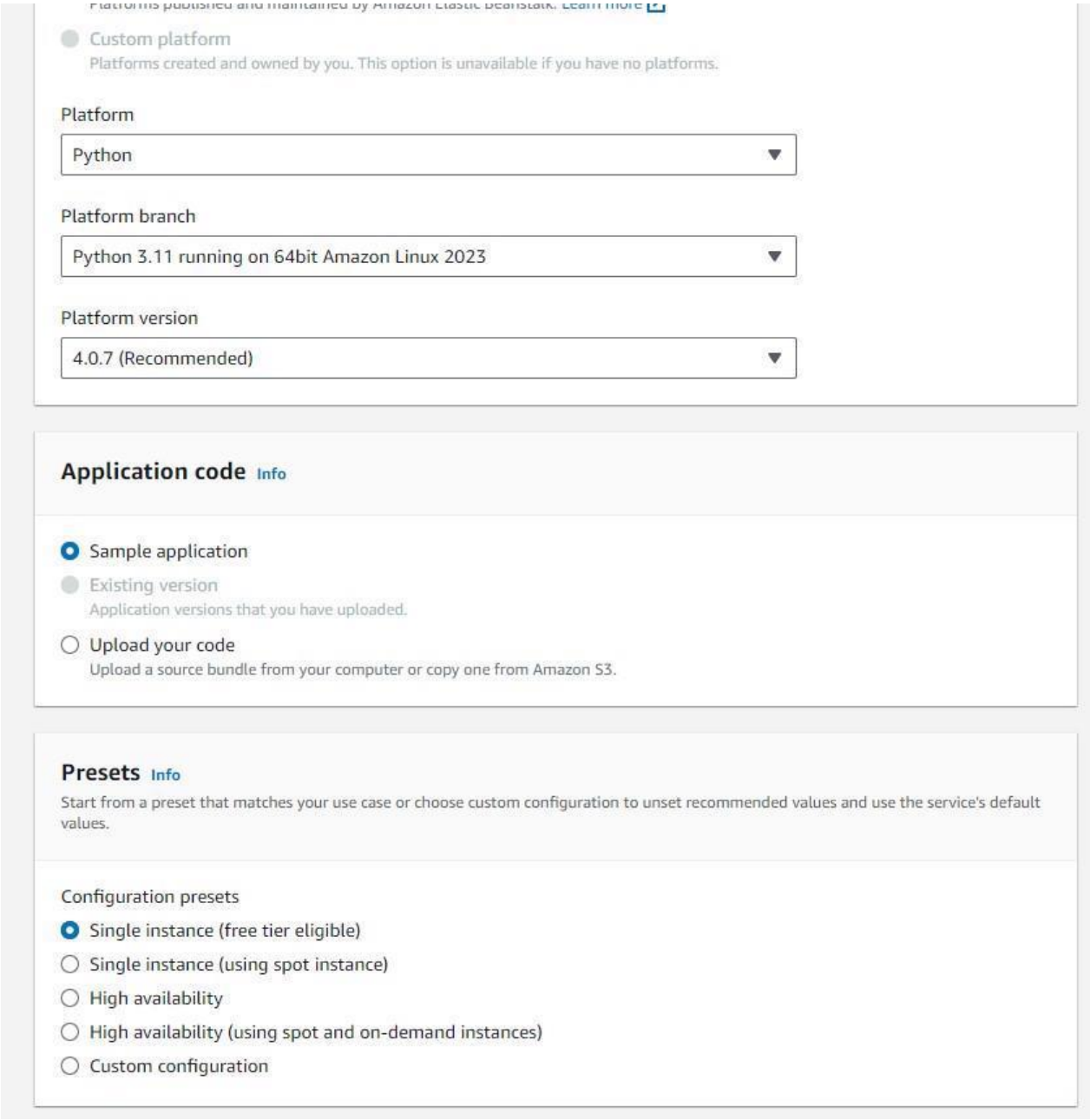
2. There will be the steps



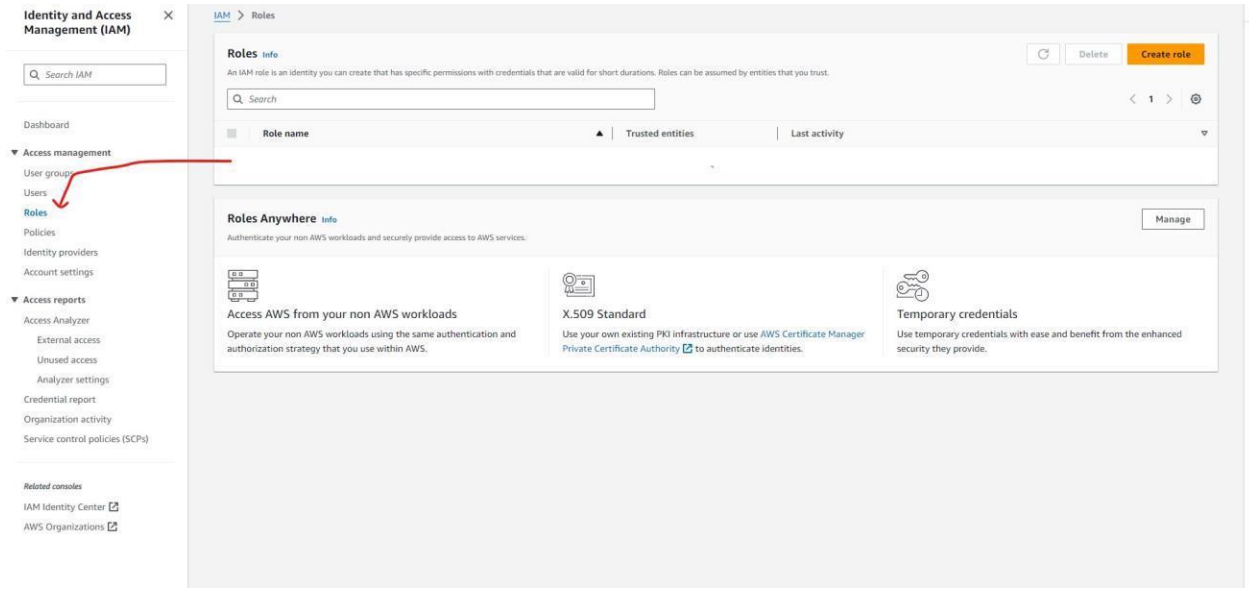
3. Fill in the details



4. Select the platform



5. Go to the IAM services -> Create role



Trusted entity type

- (●) **AWS service**
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ( ) **AWS account**
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ( ) **Web identity**
  Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ( ) **SAML 2.0 federation**
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ( ) **Custom trust policy**
  Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

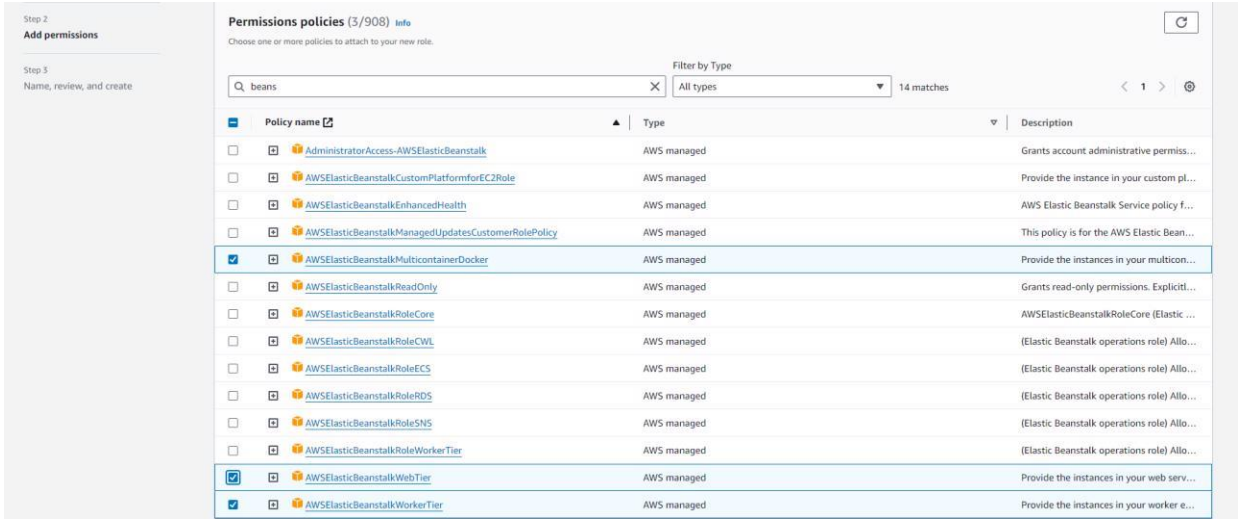
Service or use case

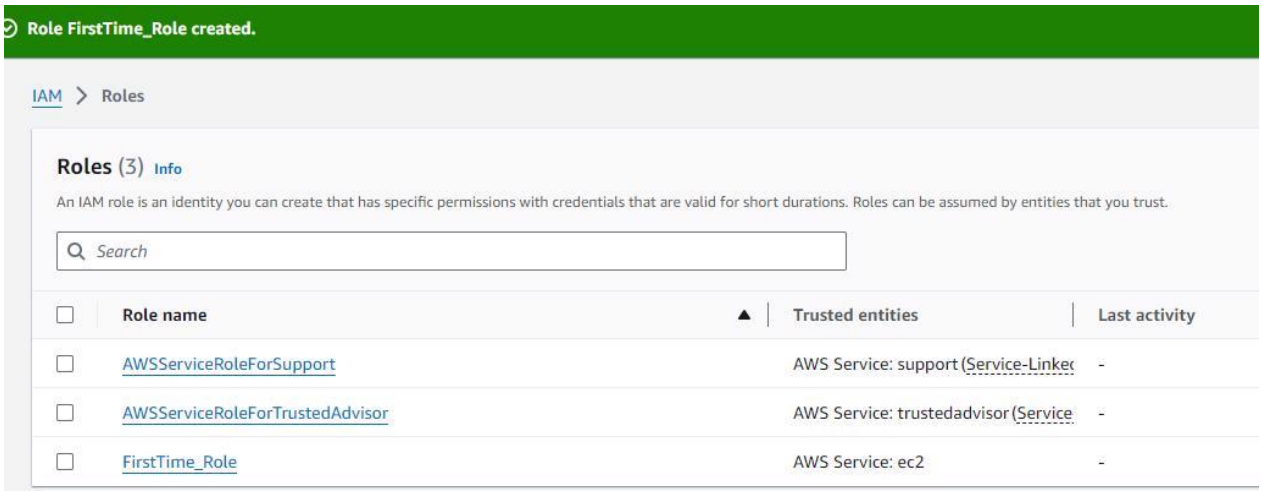
EC2

Choose a use case for the specified service.

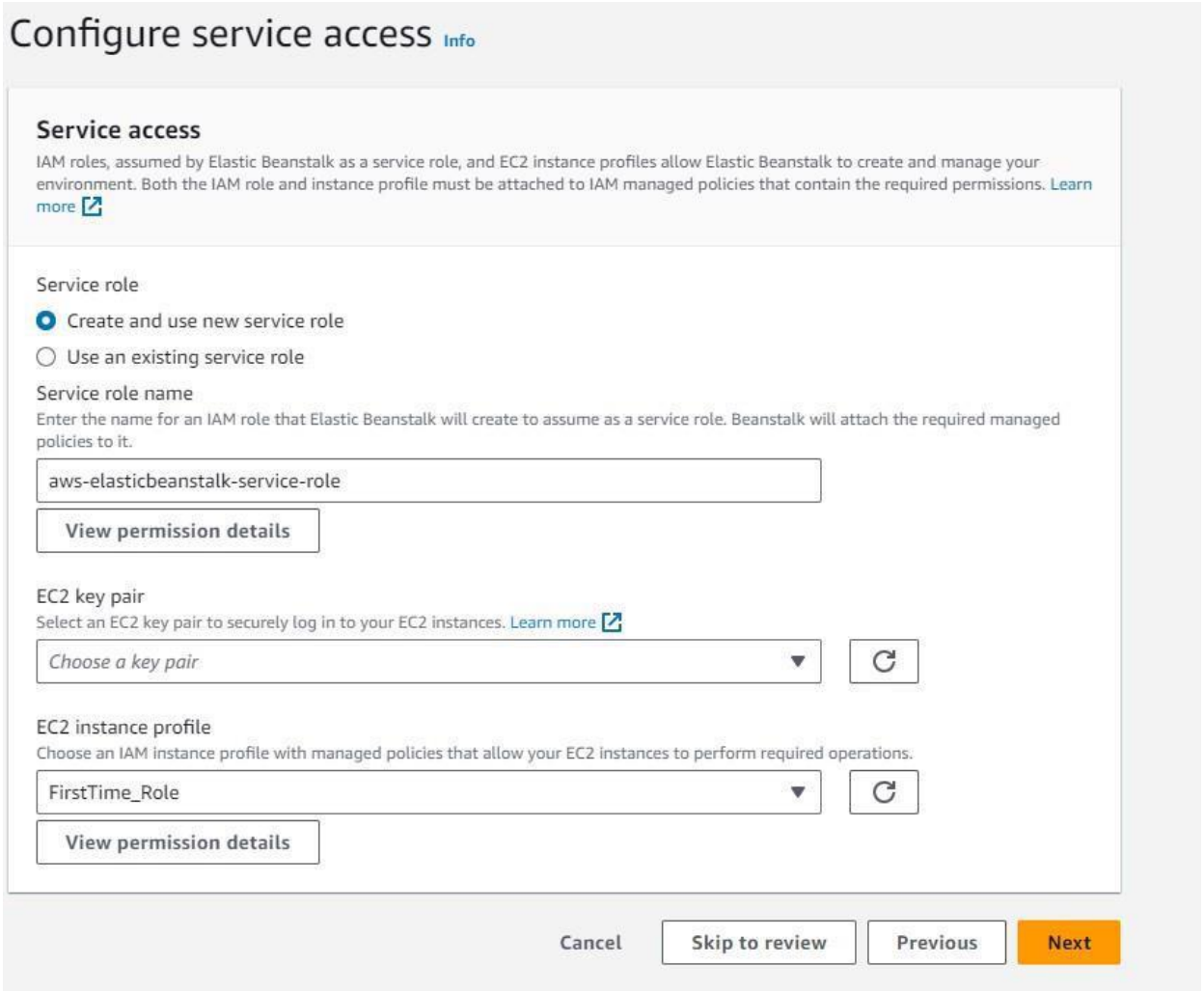
Use case

- (●) **EC2**
  Allows EC2 instances to call AWS services on your behalf.
- ( ) **EC2 Role for AWS Systems Manager**
  Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- ( ) **EC2 Spot Fleet Role**
  Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- ( ) **EC2 - Spot Fleet Auto Scaling**
  Allows Auto Scaling to access and update EC2 spot fleets on your behalf.

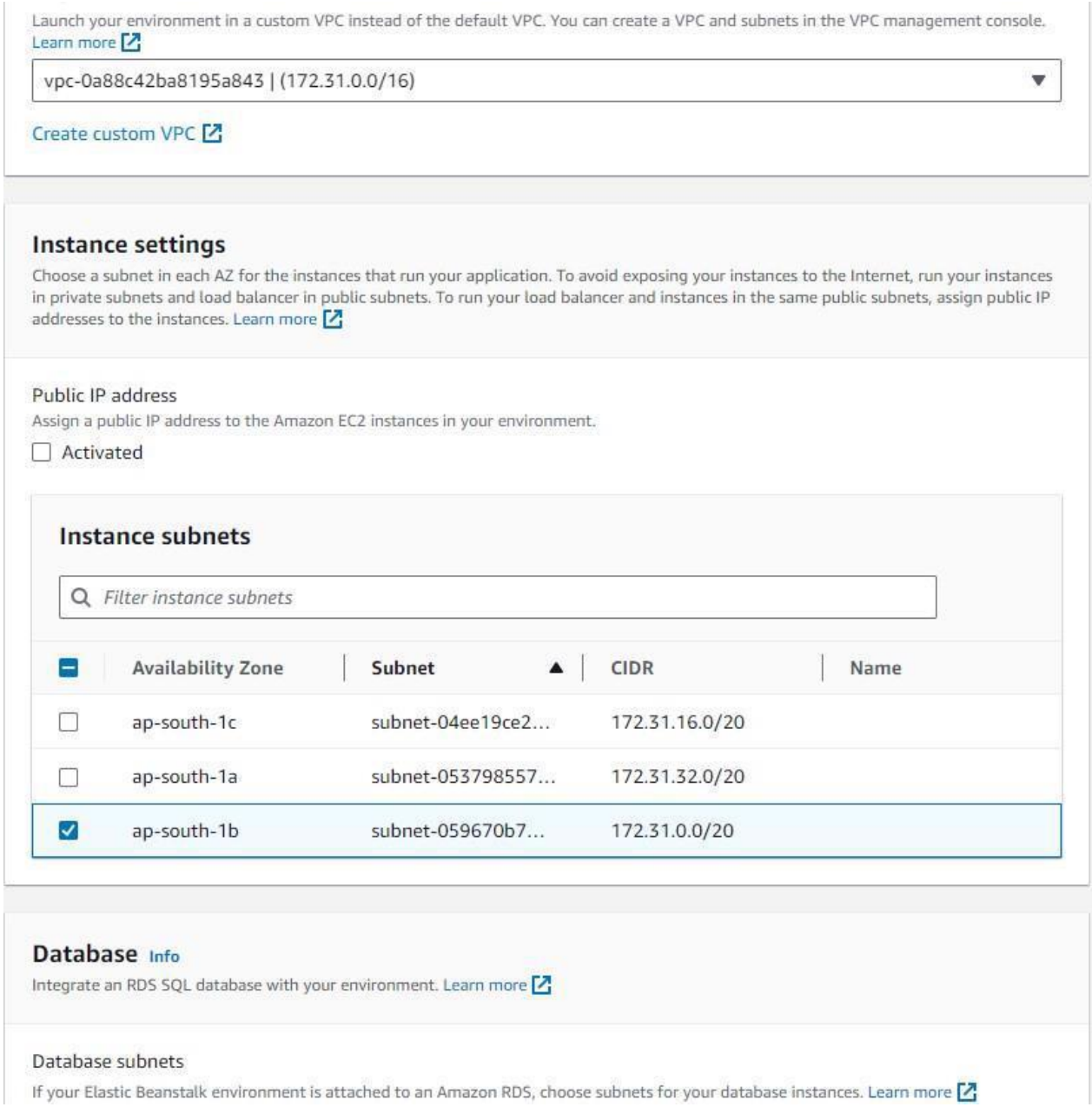




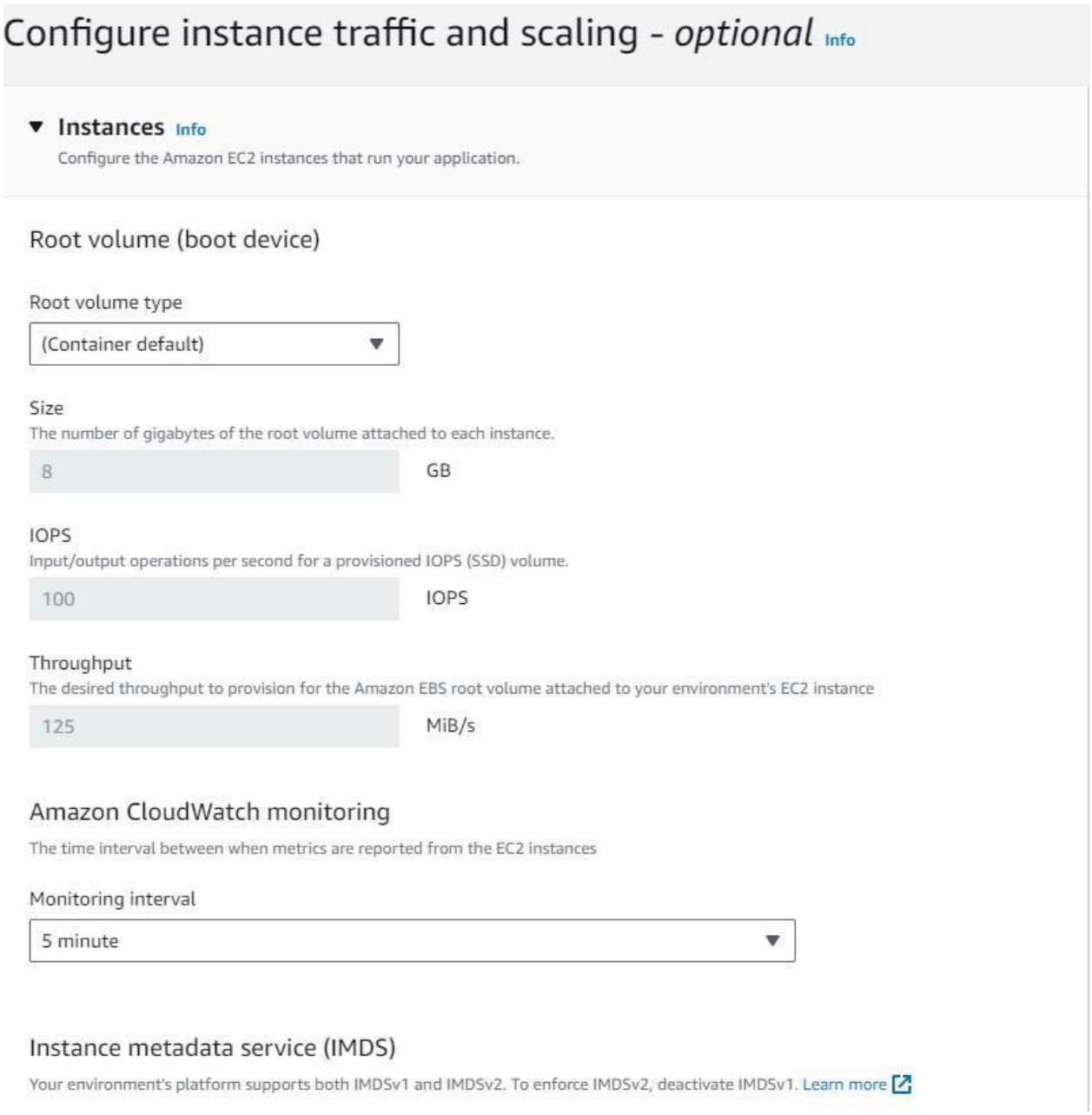
6. Create new service role, and select the EC instance profile



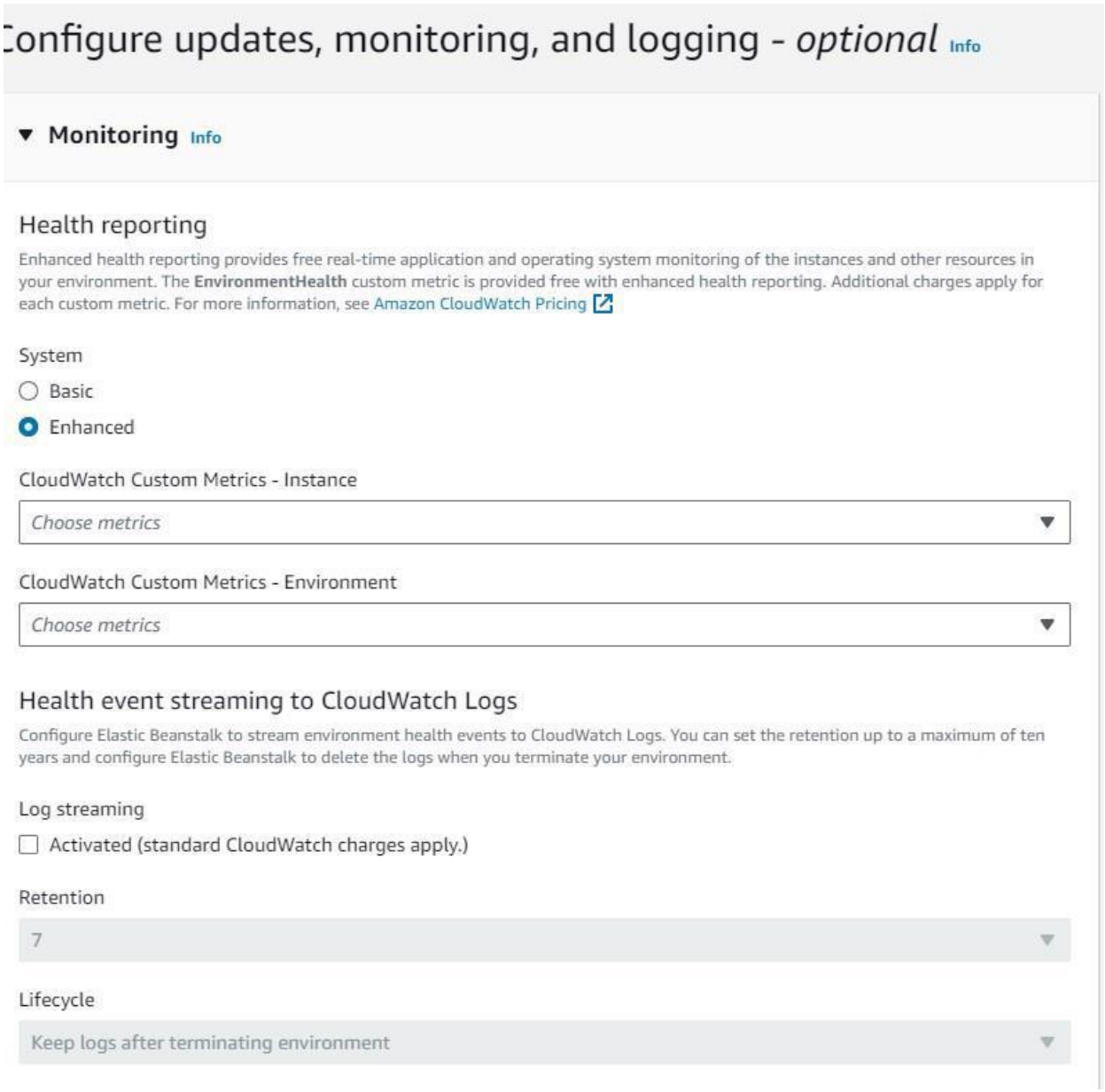
7. Set up network and database



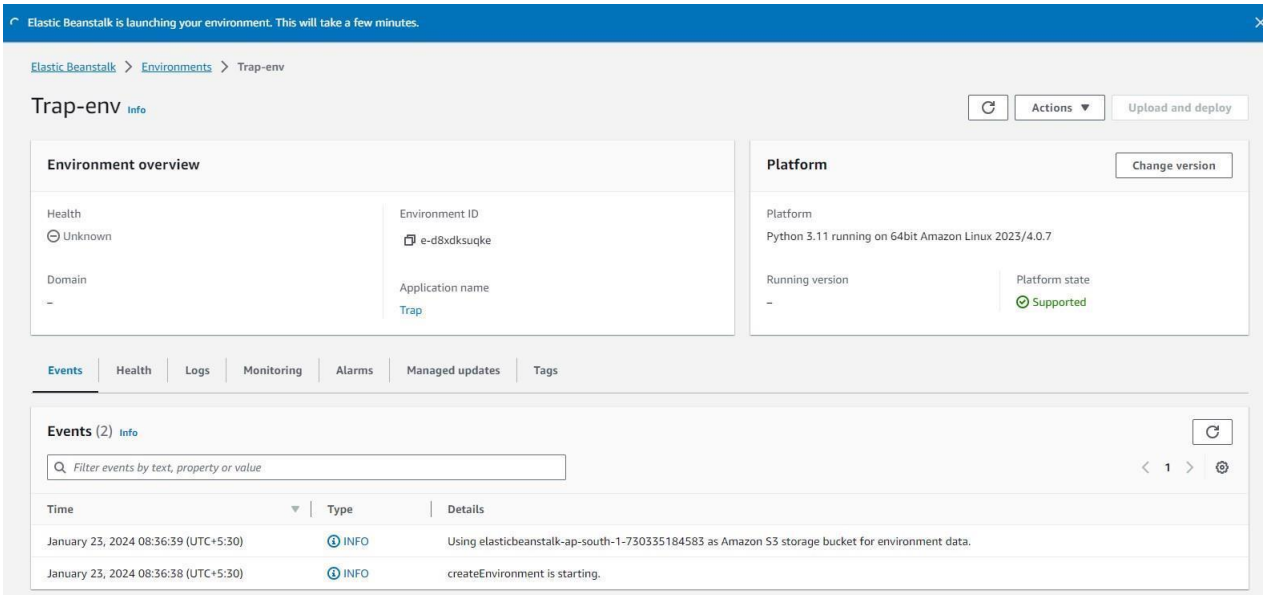
8. Keep the settings default

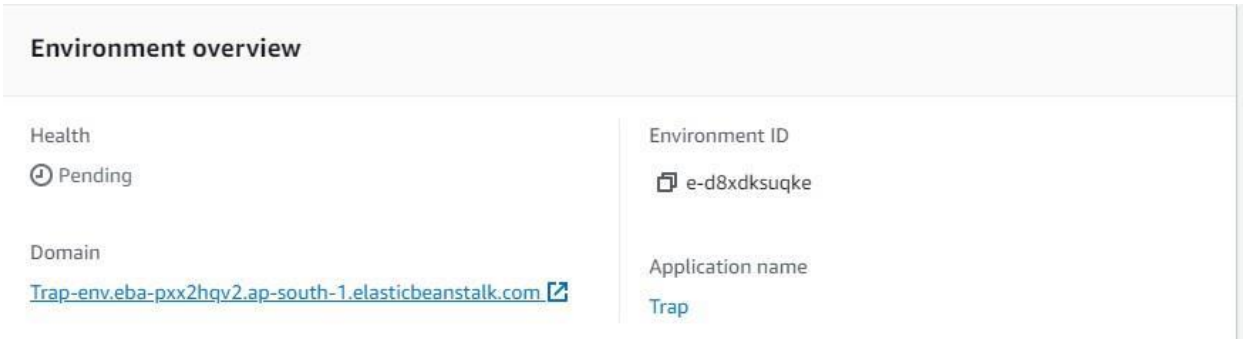


9. Keep the settings default

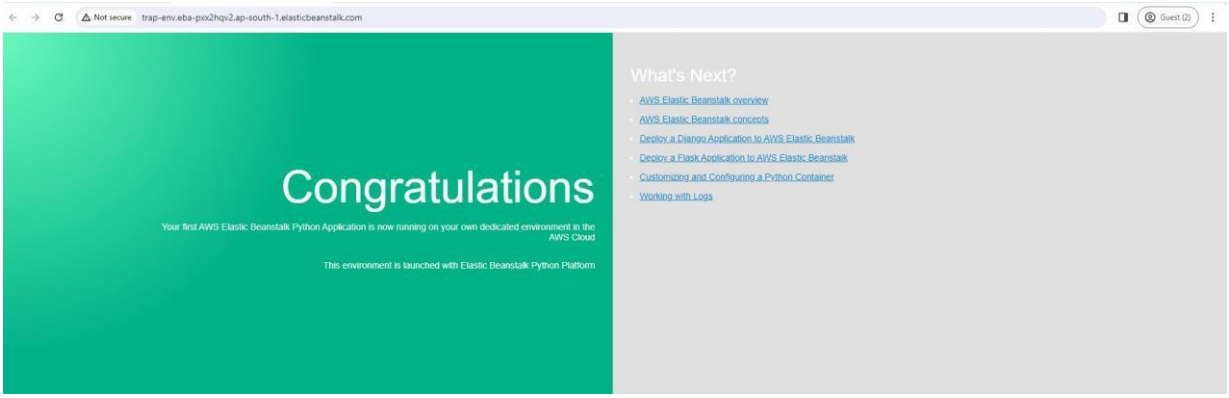


10. Review and submit!
11. The environment will start launching

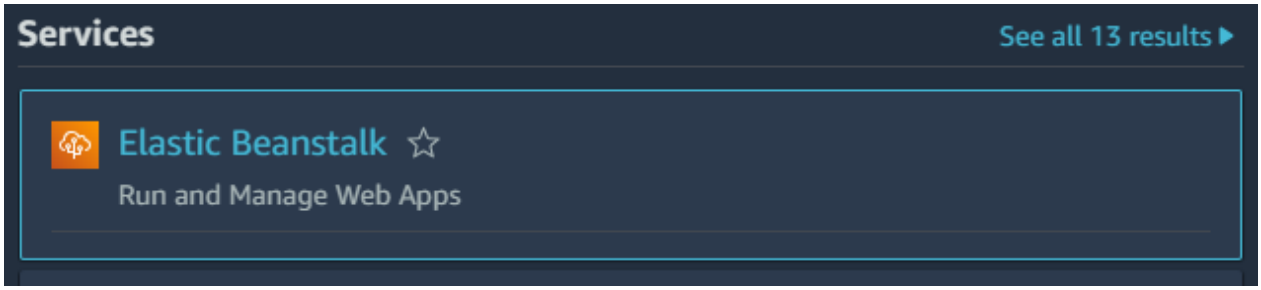




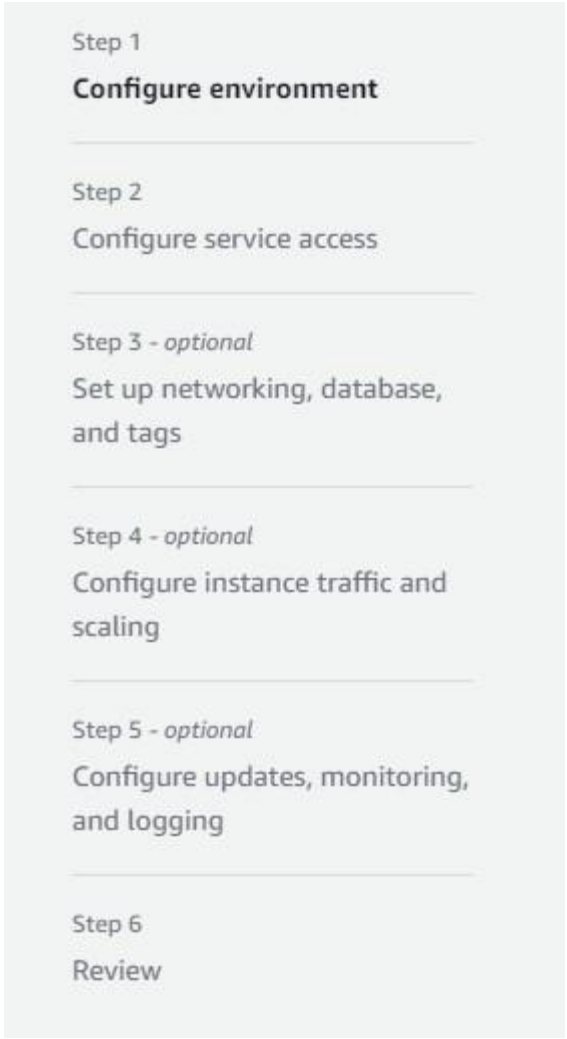
The webapp is launched!

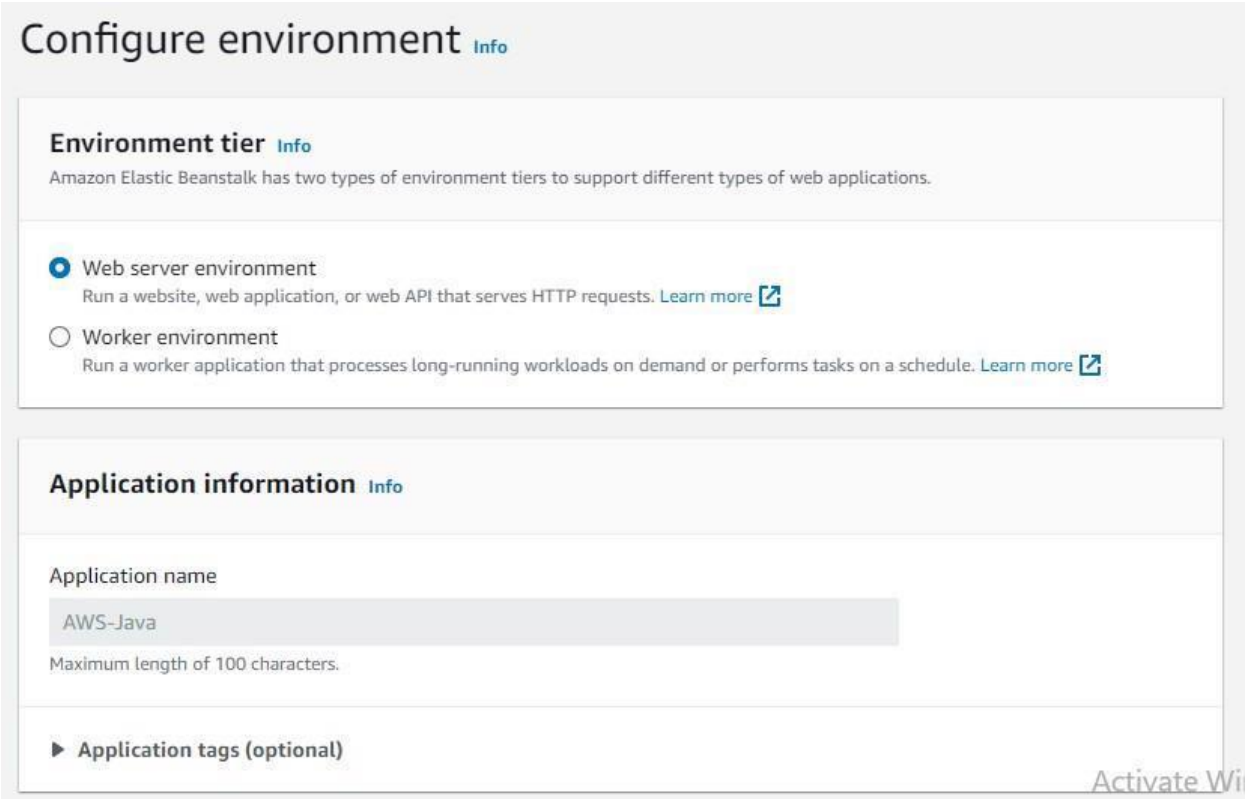


1. We'll be using Elastic Beanstalk - Java

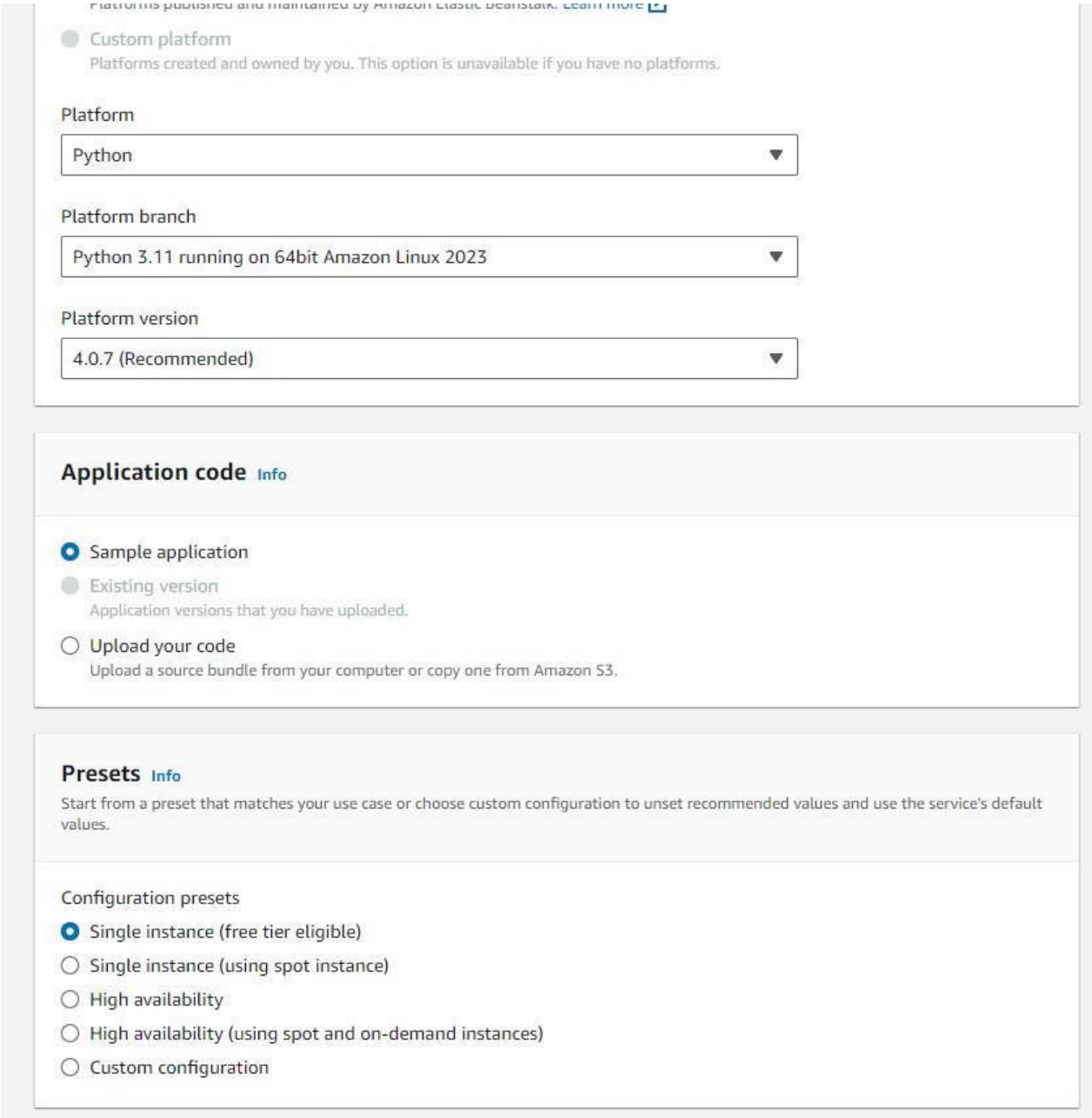


2. There will be the steps

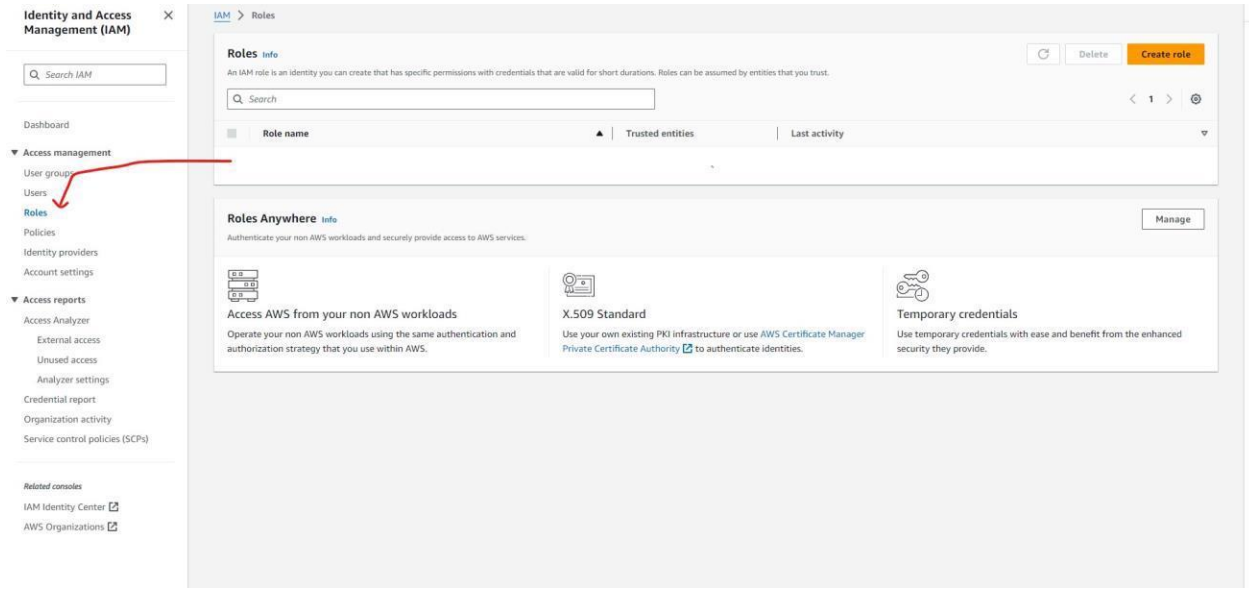




12. Select the platform



13. Go to the IAM services -> Create role



Trusted entity type

- (●) **AWS service**
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ( ) **AWS account**
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ( ) **Web identity**
  Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ( ) **SAML 2.0 federation**
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- ( ) **Custom trust policy**
  Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

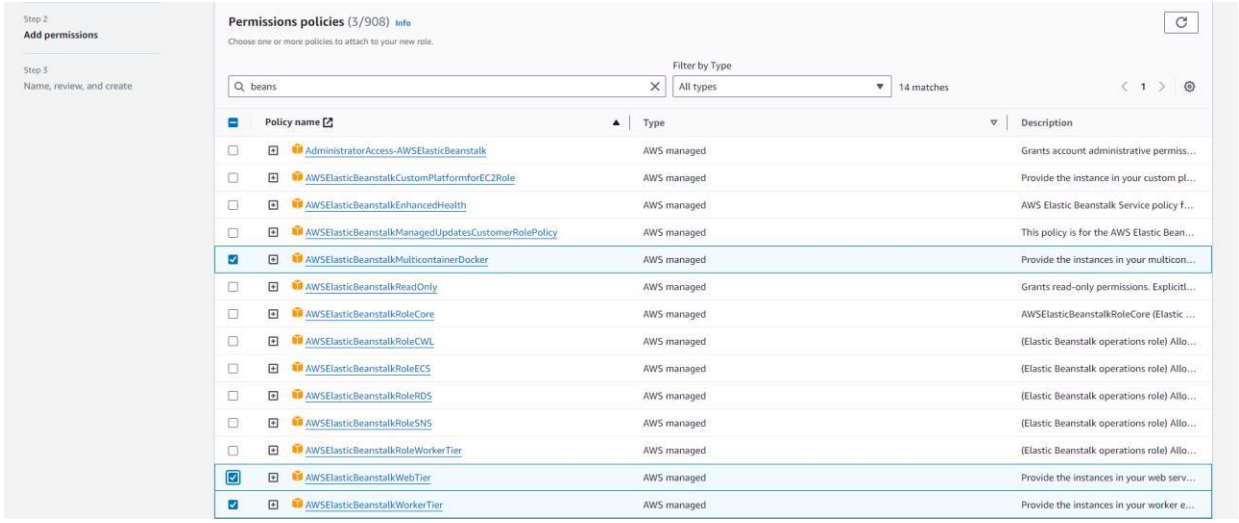
Service or use case

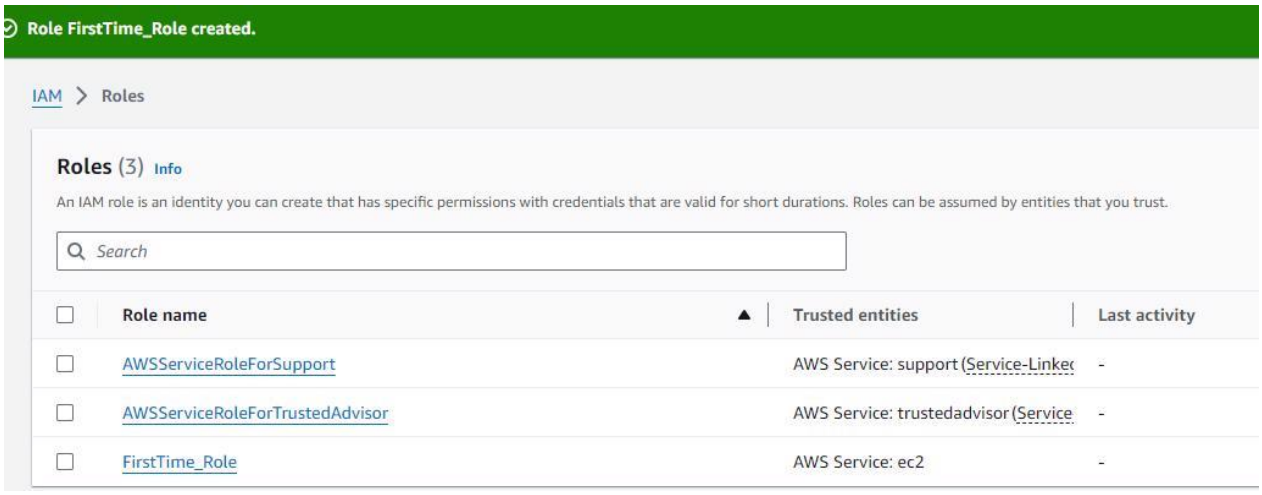
EC2

Choose a use case for the specified service.

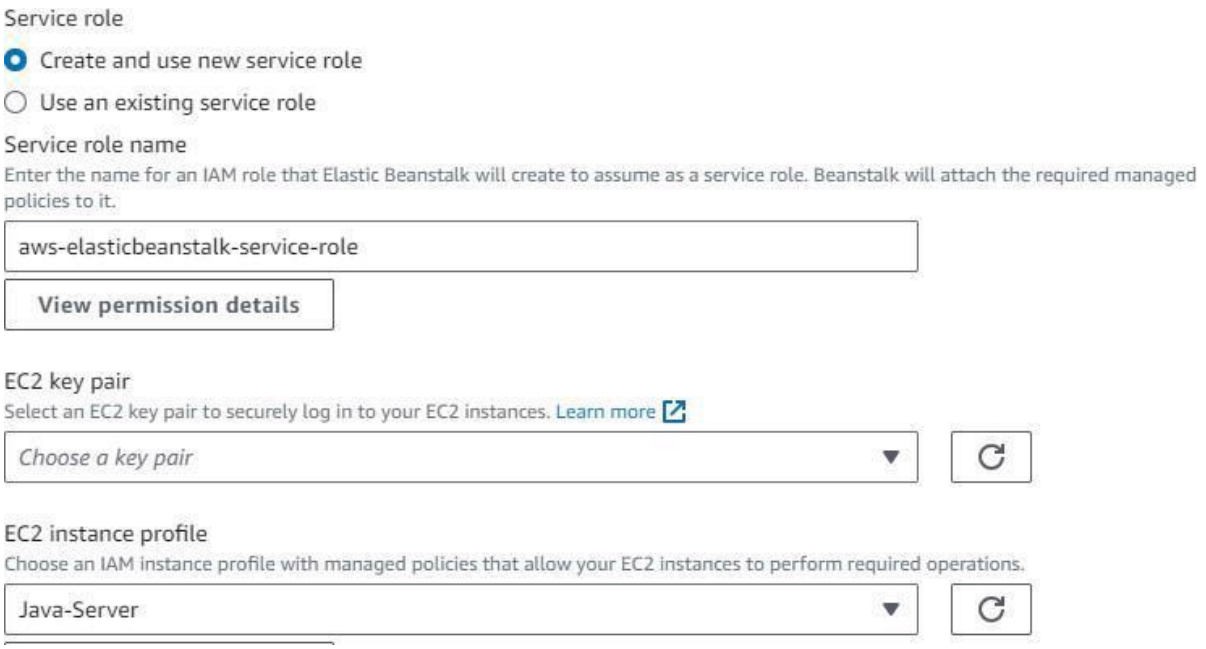
Use case

- (●) **EC2**
  Allows EC2 instances to call AWS services on your behalf.
- ( ) **EC2 Role for AWS Systems Manager**
  Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- ( ) **EC2 Spot Fleet Role**
  Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- ( ) **EC2 - Spot Fleet Auto Scaling**
  Allows Auto Scaling to access and update EC2 spot fleets on your behalf.





14. Create new service role, and select the EC instance profile



15. Set up network and database

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-0a88c42ba8195a843 | (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

Filter instance subnets

	Availability Zone	Subnet	CIDR	Name
☐	ap-south-1c	subnet-04ee19ce2...	172.31.16.0/20	
☐	ap-south-1a	subnet-053798557...	172.31.32.0/20	
☑	ap-south-1b	subnet-059670b7...	172.31.0.0/20	

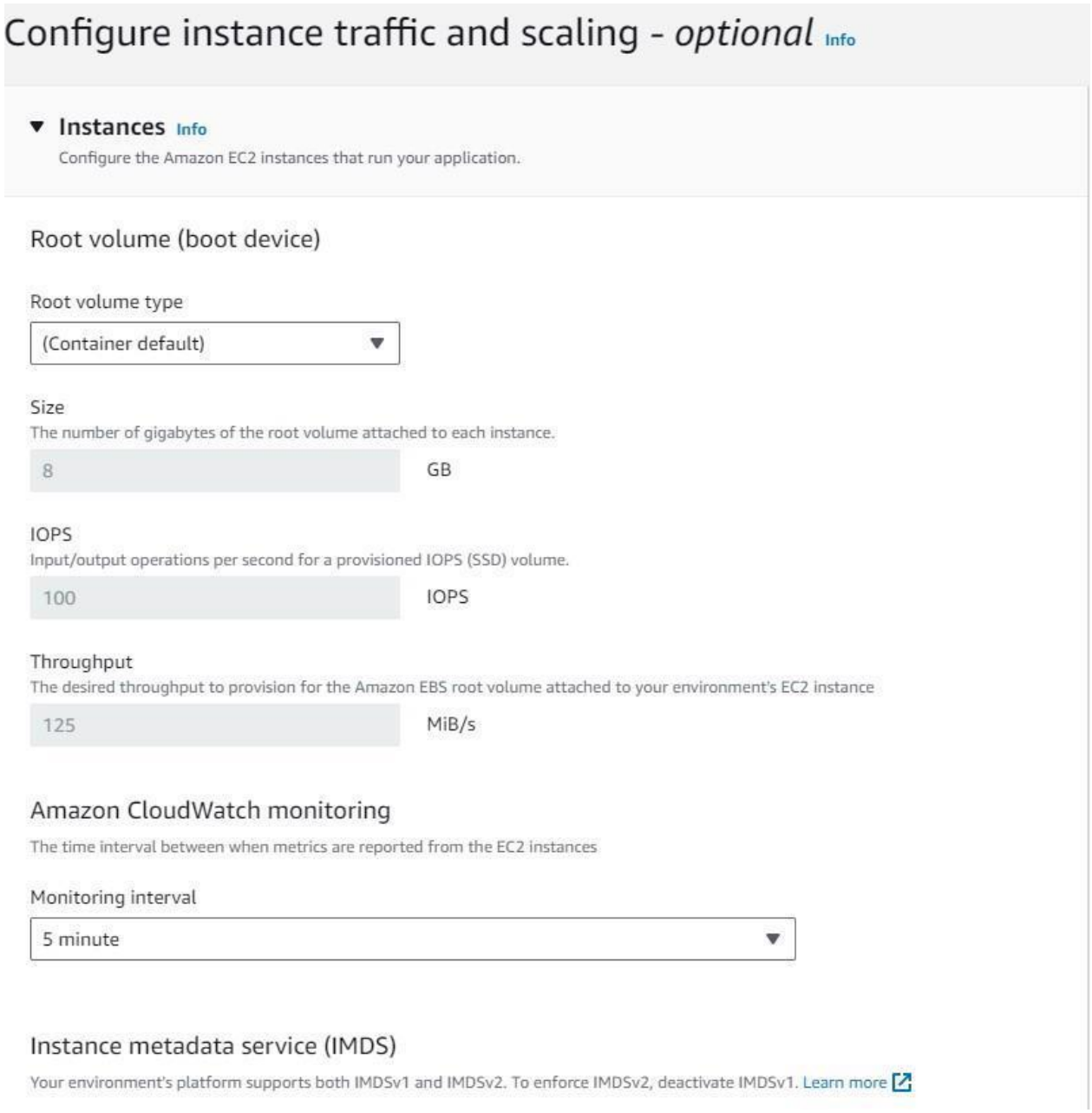
Database Info

Integrate an RDS SQL database with your environment. Learn more

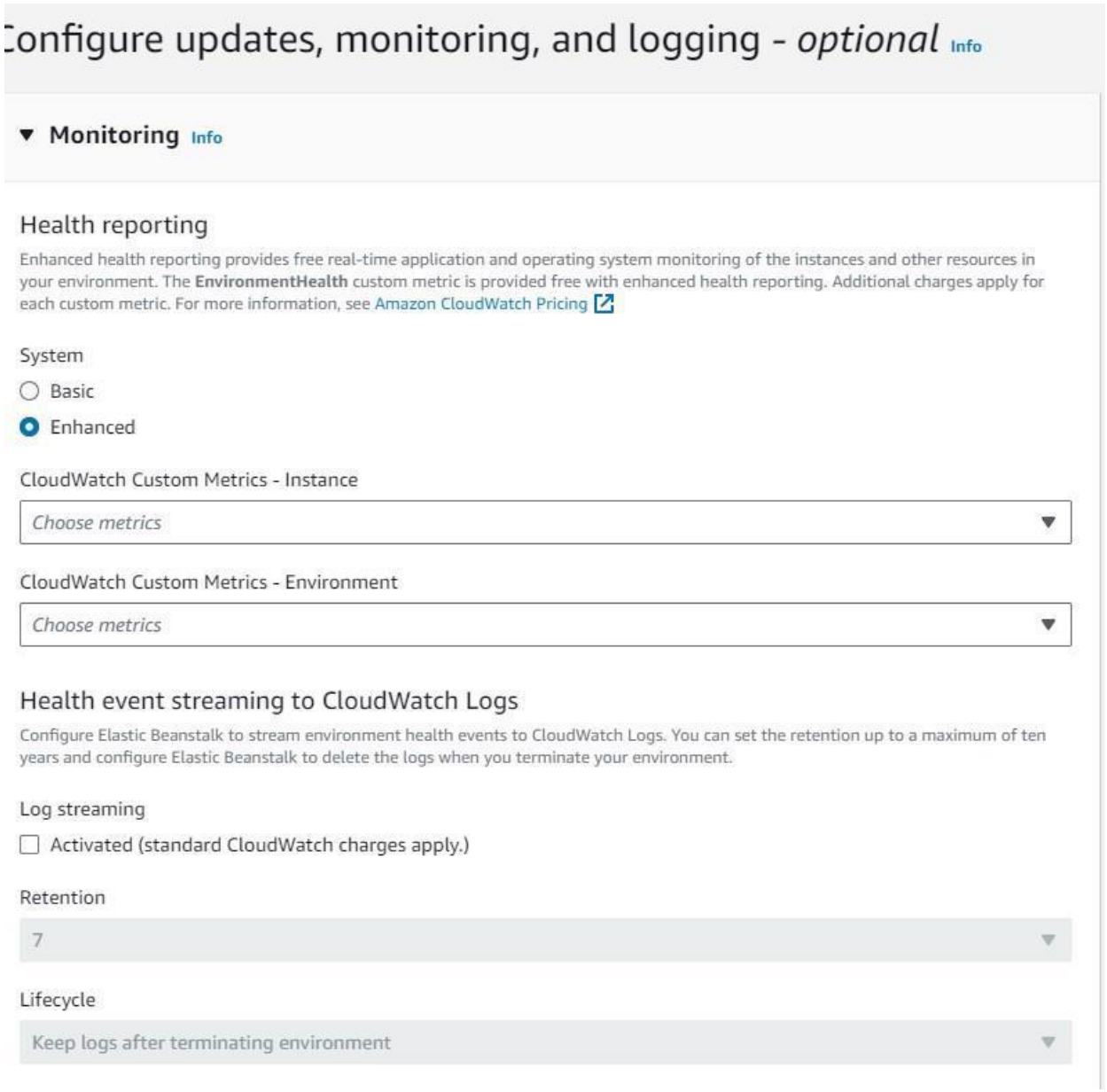
Database subnets

If your Elastic Beanstalk environment is attached to an Amazon RDS, choose subnets for your database instances. Learn more

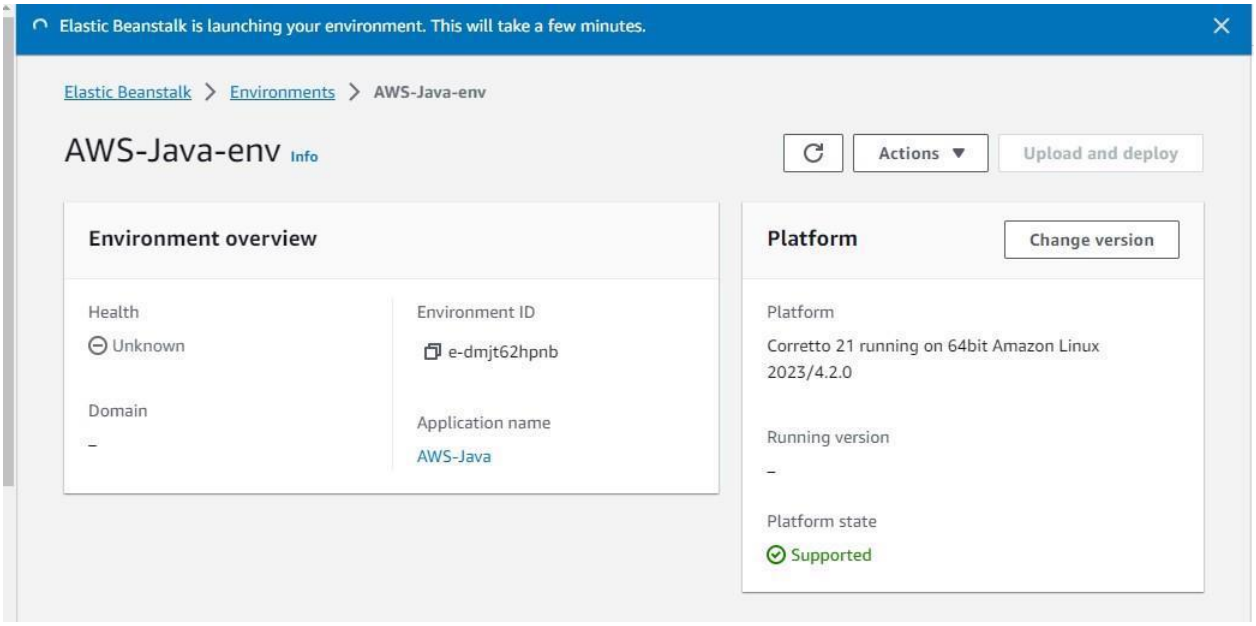
16. Keep the settings default

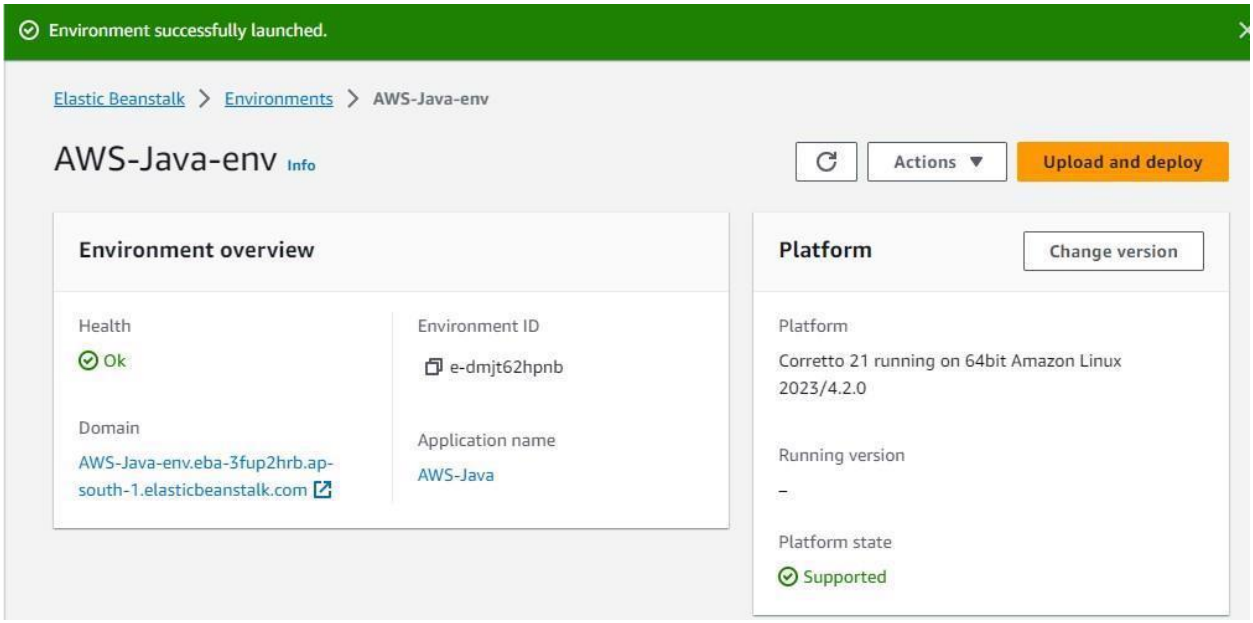


17. Keep the settings default

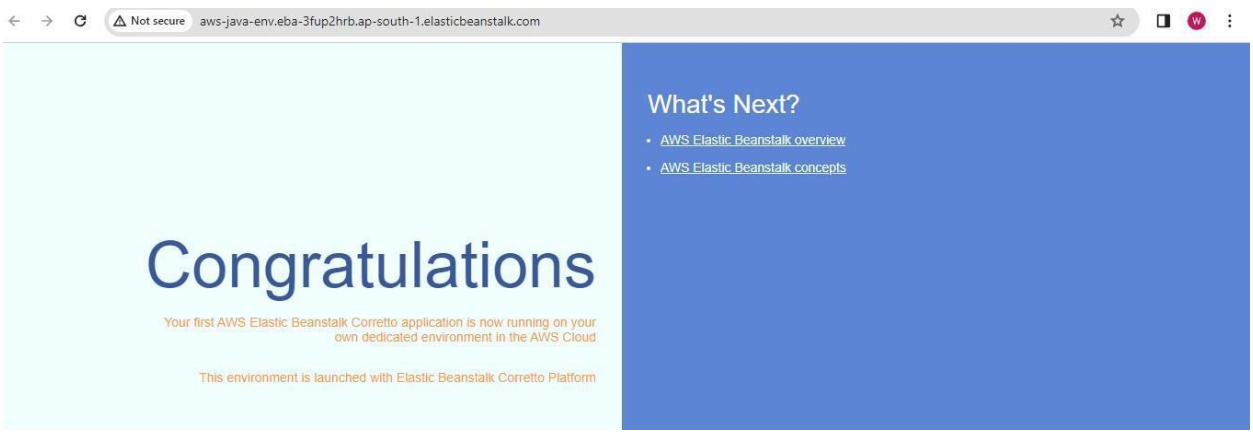


18. Review and submit!
19. The environment will start launching





The webapp is launched!


Not secure
aws-java-env.eba-3fup2hrb.ap-south-1.elasticbeanstalk.com
Congratulations
Your first AWS Elastic Beanstalk Corretto application is now running on your own dedicated environment in the AWS Cloud
This environment is launched with Elastic Beanstalk Corretto Platform
What's Next?
AWS Elastic Beanstalk overview
AWS Elastic Beanstalk concepts