THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

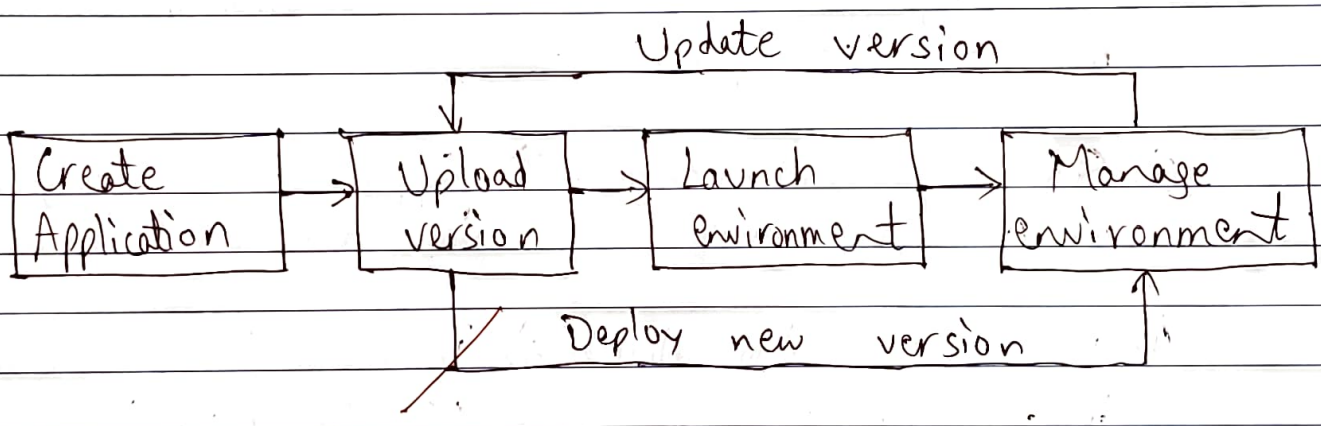
EXPERIMENT NO. 5

Aim :- To study and implement Platform as a service using AWS Elastic Beanstalk Service.

Theory :-

Amazon Web Service (AWS) Elastic Beanstalk Service :

With Elastic Beanstalk, one can quickly deploy and control apps in AWS cloud without having to learn about infrastructure that runs those apps. Elastic Beanstalk reduces restricting choice or ctrl. You simply upload your application, and it automatically handles details of capacity provisioning, load balancing, scaling and app health monitoring.



Elastic Beanstalk supports apps developed in Go, Java, .NET, Node.js, PHP, Python and Ruby. When you deploy your application, it builds selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your apps.

EBS supports a variety of programming languages and frameworks like Java, .NET, Node.js, Python, Ruby, PHP, Go, Docker.

Compare EC2 and Elastic Beanstalk

	EC2		EBS
1)	It provides raw virtual machines (instances) in cloud. Users have full control over OS and networking.	1)	It abstracts away the solution infrastructure details and provides platform as a Service Solution
2)	Users are responsible for deploying application managing s/w updates.	2)	EBS takes care of the underlying infrastructure.
3)	Supports a wide range of OS and services.	3)	Supports specific programming languages and frameworks.
4)	Pricing is based on selected instance type and usage.	4)	Pricing is based on resources used by underlying infra.
5)	More effort needed to handle increased traffic.	5)	Monitors performance and provides easier scalability.

Elastic Load Balancing :

ELB automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more zones. It monitors health of its registered targets, and routes traffic only to healthy targets. Elastic Load Balancing scales your load balancer capacity automatically in response to changes in traffic.

Steps for deploying web apps / web services on AWS Elastic BeanStalk.

Step 1 : Login to AWS console and go to Elastic BeanStalk
Step 2 : Click on create application.
Step 3 : Write app information : Name, tag, Platform, etc.
Step 4 : In app code, select sample application ~~and~~ then click on button create application.
Step 5 : Click on environments → check the health of environments till it becomes 'ok'.
Step 6 : Click the URL.

Conclusion :-

By Performing the above experiment, we were able to understand EBS and its applications and deploy services on EBS.

At
15/3/24

Rishab Mandal
2103110
C23

Cloud Computing Experiment 5

Aim: To study and Implement Platform as a Service using AWS Elastic Beanstalk Service.

Theory:

1. Introduction

This experiment delves into the practical exploration of the Amazon AWS Elastic Beanstalk Service, a prominent Platform as a Service (PaaS) offering within the Amazon Web Services (AWS) ecosystem. The primary focus of this experiment is to gain an in-depth understanding of how Elastic Beanstalk streamlines the deployment, scaling, and management processes of web applications and services. By leveraging Elastic Beanstalk, developers can abstract away the complexities of infrastructure provisioning, allowing them to concentrate more on coding and application development rather than infrastructure management.

2. Amazon AWS Elastic Beanstalk Service

Amazon AWS Elastic Beanstalk Service serves as an advanced cloud deployment service designed to automate the intricate tasks associated with setting up and maintaining the underlying infrastructure required for hosting web applications. This comprehensive platform simplifies the deployment process by orchestrating the provisioning of essential components such as compute instances, load balancers, auto-scaling groups, and networking resources. Through Elastic Beanstalk's intuitive interface and robust automation capabilities, developers can expedite the deployment process and

ensure the seamless operation of their applications without the need for extensive manual intervention.

AWS Elastic Beanstalk Components

- **Application:** Elastic Beanstalk directly takes in our project code. So Elastic Beanstalk application is named the same as your project home directory.
- **Application Environments:** Users may want their application to run on different environments like DEV, UAT, and PROD. You can create and configure different environments to run applications on different stages.
- **Environment Health:** One of the most lucrative features of running applications on AWS or most of the other cloud platforms is automated health checks. AWS runs automatic health checks on all EC-2 deployments (Elastic Beanstalk is a managed EC-2 service) which can be monitored from the AWS console. For example, in the case of web applications AWS will regularly, as scheduled by the developers, ping the application to check if the response is status code 200 and if the application is running as expected. Health check responses:
    - **Red:** The application failed all health tests.
    - **Yellow:** The application failed some of the health tests.
    - **Grey:** The application is updating.
    - **Green:** The application passed the health check successfully.
- **Isolated:** All environments within a single application are isolated from each other (independent of each other's running states). Needless to say, two different applications are also isolated.
- **Scalability:** Using Auto-Scaling within Elastic Beanstalk makes the application dynamically scalable.
- **Elastic Load Balancing:** All the web requests to the application are not directly related to application instances. They first hit the Elastic Load Balancer (ELB), which, as the name suggests, balances the load across all the application instances.

- **Language support:** Elastic Beanstalk supports the applications developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS.
- **Pricing:** There is no extra charge for using Elastic Beanstalk. Users are only required to pay for the services and resources provisioned by Elastic Beanstalk Service.
- **Automatic Provisioning:** Elastic Beanstalk takes away the burden of choosing the right services and configuring their security groups to work together.
- **Impossible to Outgrow:** AWS claims that since Elastic Beanstalk uses the Auto Scaling feature it can, in theory, handle any amount of internet traffic.

3. Supported Languages/Frameworks

One of the notable advantages of Elastic Beanstalk lies in its broad support for various programming languages and frameworks. Developers can choose from a diverse array of options, including Java, Python, Node.js, Ruby, PHP, .NET, and Docker, among others. This extensive language and framework support empowers developers to select the most suitable technology stack for their specific application requirements and proficiency levels. Whether it's a robust Java enterprise application or a lightweight Python web service, Elastic Beanstalk accommodates a wide range of development preferences and project specifications.

Platforms for Programming Languages Provided by Elastic Beanstalk are

- GO
- Java
- Node.js
- PHP
- Python
- Ruby

Platforms for Application Servers Provided by Elastic Beanstalk are

- Tomcat
- Docker

4. Comparison with Amazon EC2

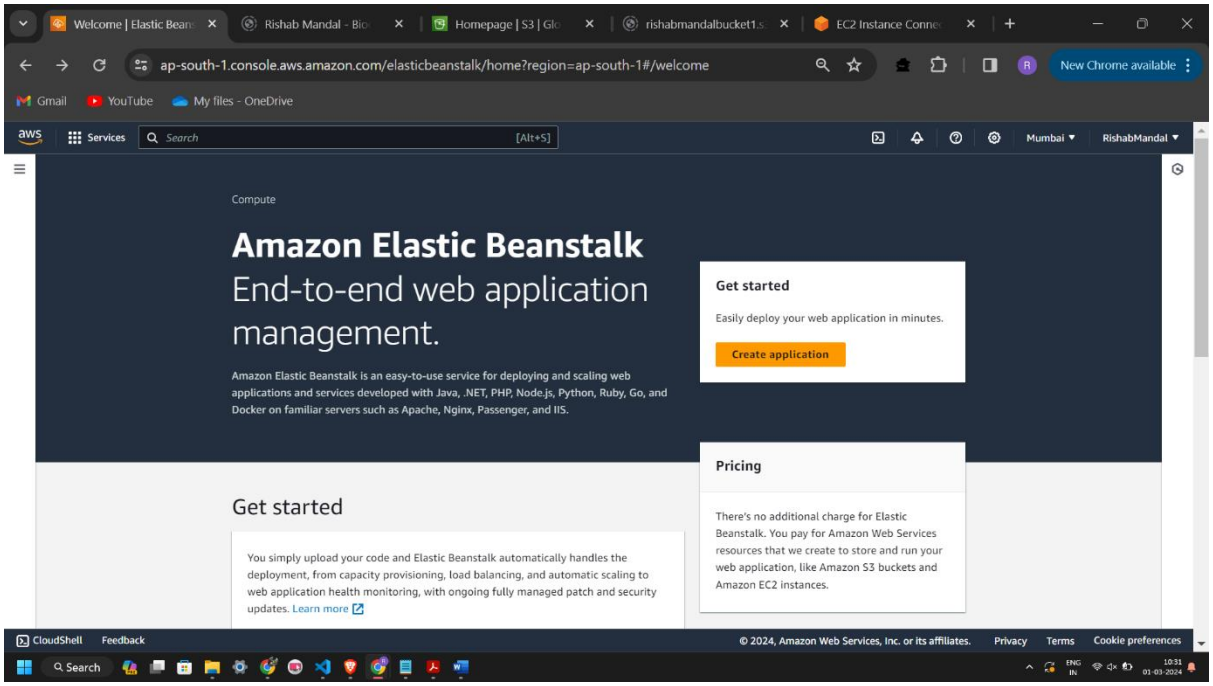
In contrast to the more traditional Infrastructure as a Service (IaaS) model provided by Amazon EC2 (Elastic Compute Cloud), Elastic Beanstalk operates within the Platform as a Service (PaaS) paradigm. While EC2 grants users greater control and flexibility over their cloud infrastructure by offering virtual servers on-demand, Elastic Beanstalk abstracts much of this infrastructure management complexity. By automating deployment, scaling, and monitoring tasks, Elastic Beanstalk simplifies the development process, enabling developers to focus on writing code and delivering value to end-users rather than grappling with infrastructure intricacies.

5. Elastic Load Balancing

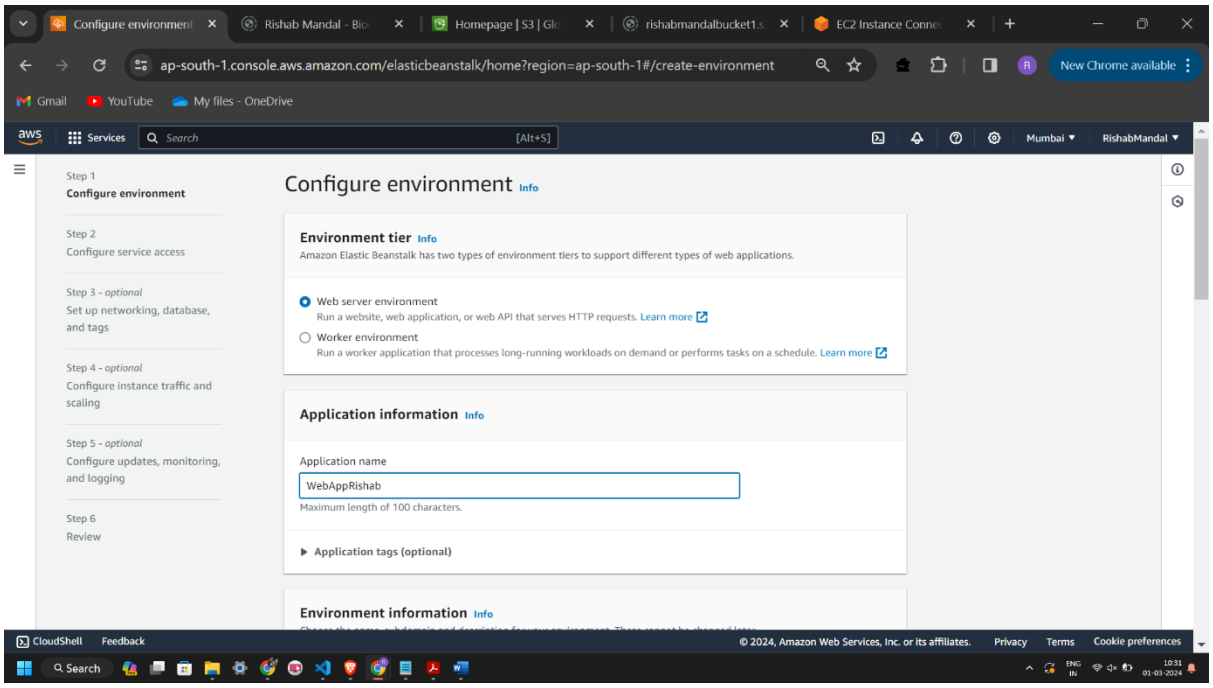
Elastic Beanstalk incorporates native support for elastic load balancing, a critical component for ensuring the availability, fault tolerance, and scalability of web applications. Leveraging elastic load balancing, Elastic Beanstalk efficiently distributes incoming traffic across multiple EC2 instances, thereby mitigating potential bottlenecks and enhancing application performance. By dynamically scaling resources in response to fluctuating demand, elastic load balancing optimizes resource utilization and ensures a seamless user experience even during periods of high traffic volume or sudden spikes in workload.

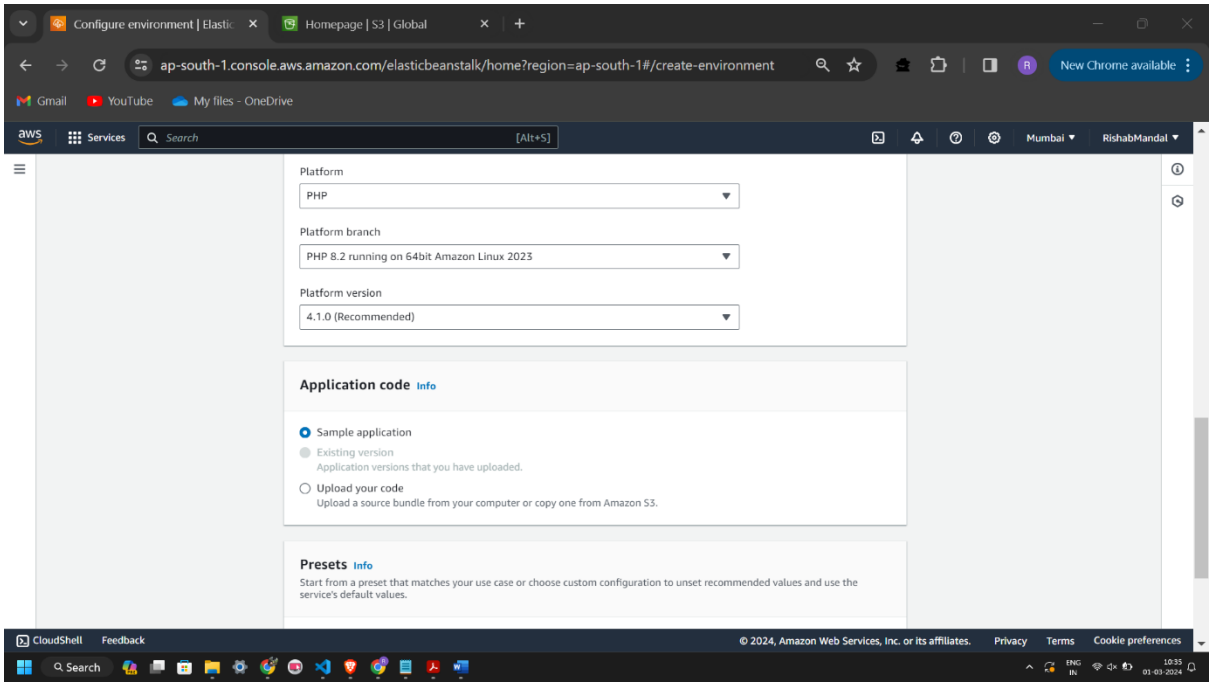
Step 1 : Login to AWS console and go to Elastic Beanstalk

Step 2: Click on Create Application

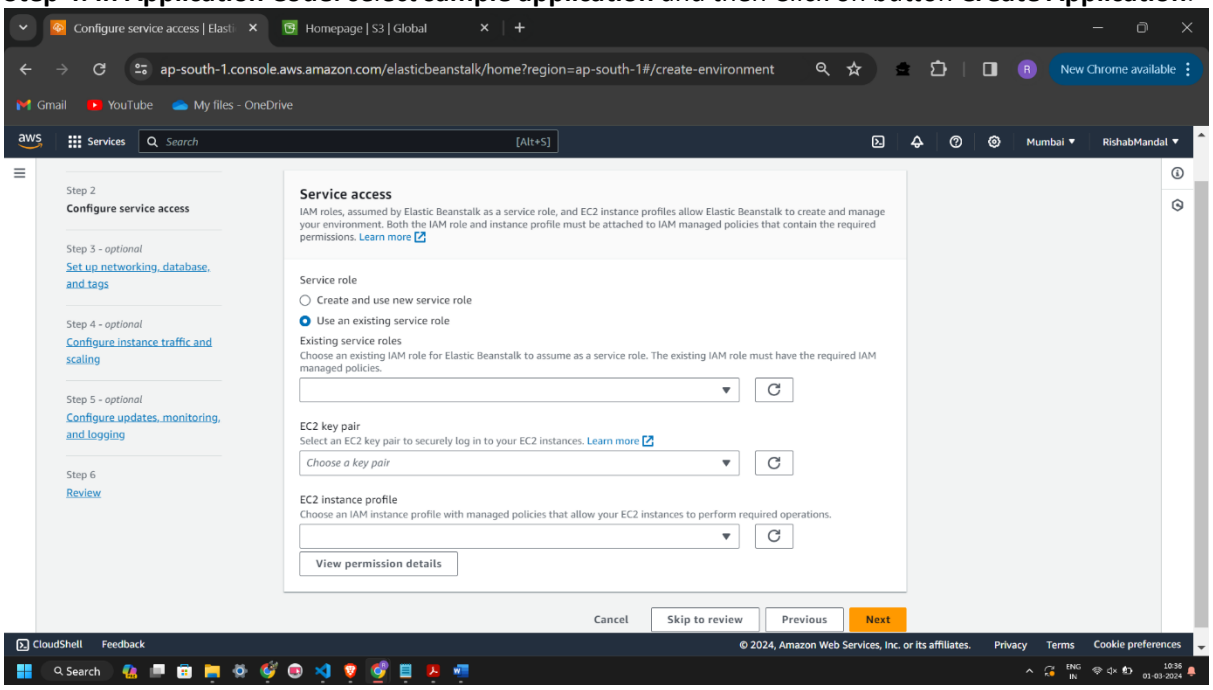


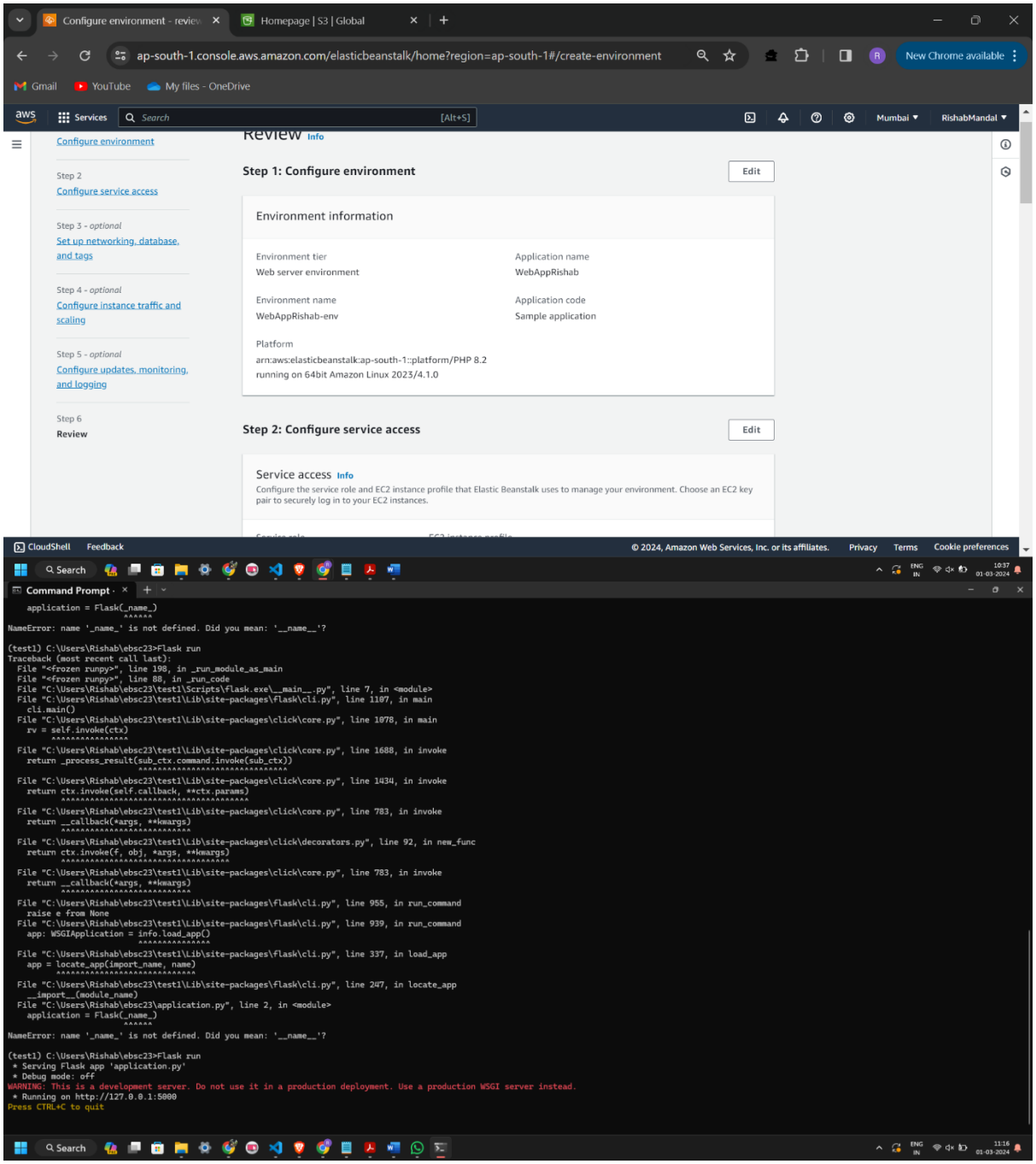
Step 3: Write Application information: Name, Tag, Platform etc.

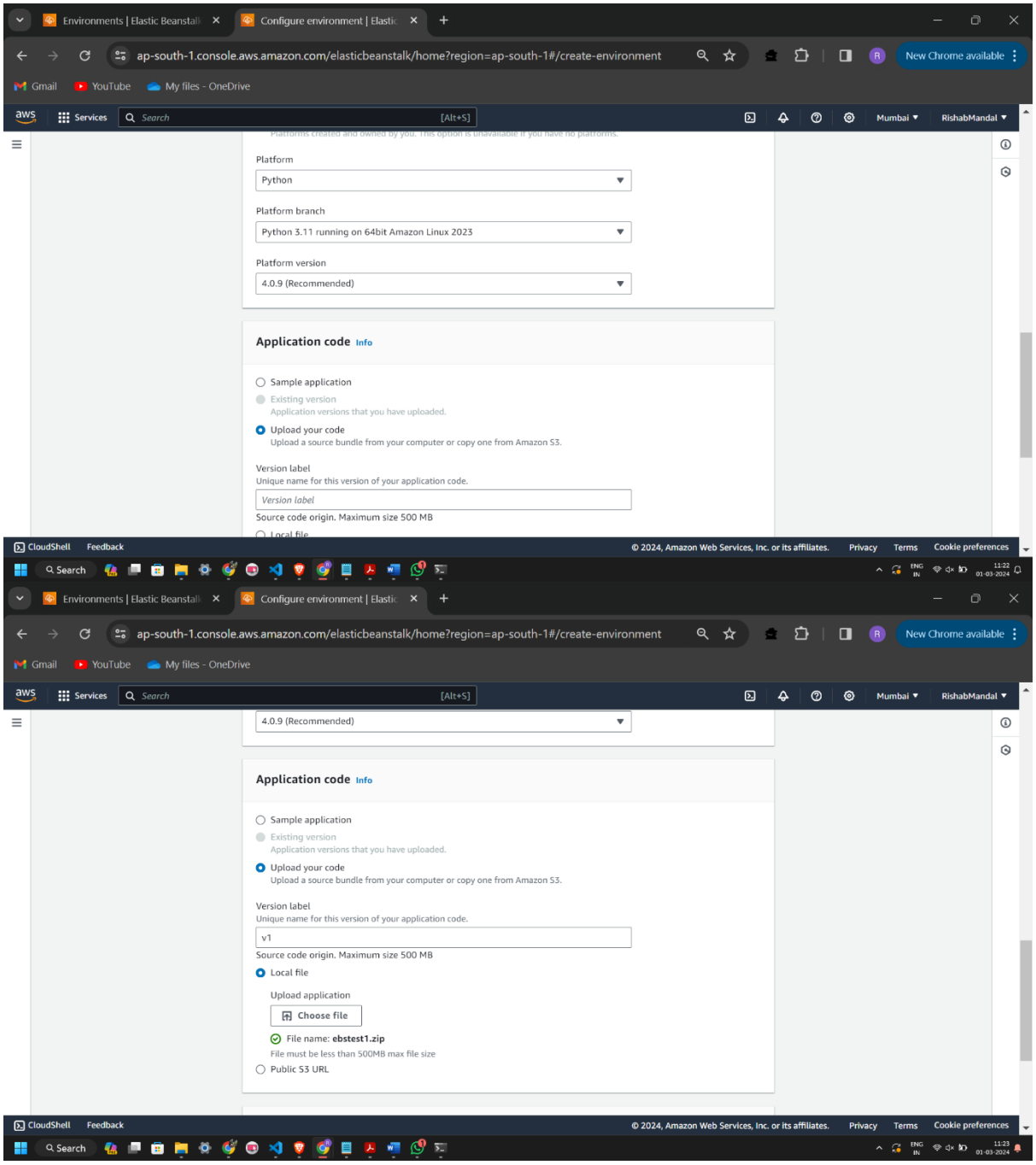




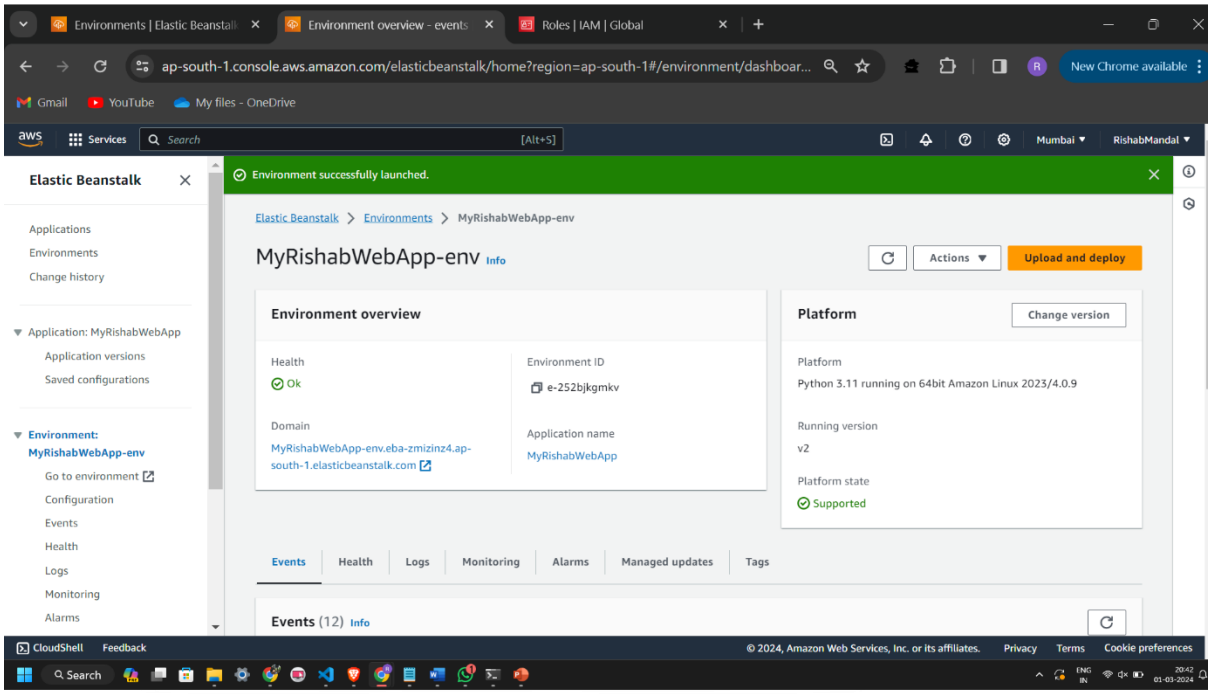
Step 4: In Application Code: select **sample application** and then Click on button **Create Application**.



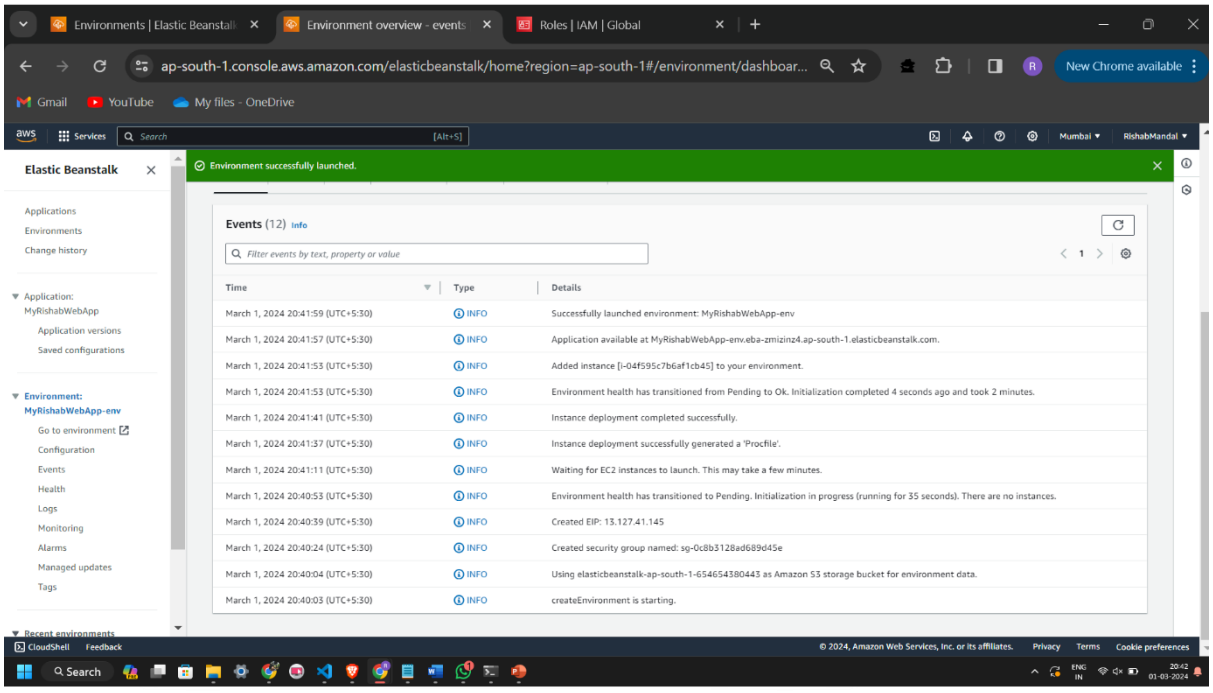

Configure environment - review
Homepage | S3 | Global
ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-environment
New Chrome available
Gmail
YouTube
My files - OneDrive
aws
Services
Search
[Alt+S]
Mumbai
RishabMandal
Configure environment
Step 2
Configure service access
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
Web server environment
Application name
WebAppRishab
Environment name
WebAppRishab-env
Application code
Sample application
Platform
arn:aws:elasticbeanstalk:ap-south-1::platform/PHP 8.2 running on 64bit Amazon Linux 2023/4.1.0
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.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Search
Command Prompt
application = Flask(_name_)
NameError: name '_name_' is not defined. Did you mean: '__name__'?
(test1) C:\Users\Rishab\ebsc23>Flask run
Traceback (most recent call last):
File "<frozen runpy>", line 198, in _run_module_as_main
File "<frozen runpy>", line 88, in _run_code
File "C:\Users\Rishab\ebsc23\test1\Scripts\flask.exe__main__.py", line 7, in <module>
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\flask\cli.py", line 1107, in main
cli.main()
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\core.py", line 1078, in main
rv = self.invoke(ctx)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\core.py", line 1688, in invoke
return _process_result(sub_ctx.command.invoke(sub_ctx))
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\core.py", line 1434, in invoke
return ctx.invoke(self.callback, **ctx.params)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\core.py", line 783, in invoke
return __callback(*args, **kwargs)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\decorators.py", line 92, in new_func
return ctx.invoke(f, obj, *args, **kwargs)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\click\core.py", line 783, in invoke
return __callback(*args, **kwargs)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\flask\cli.py", line 955, in run_command
raise e from None
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\flask\cli.py", line 939, in run_command
app: WSGIApplication = info.load_app()
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\flask\cli.py", line 337, in load_app
app = locate_app(import_name, name)
File "C:\Users\Rishab\ebsc23\test1\Lib\site-packages\flask\cli.py", line 247, in locate_app
__import__(module_name)
File "C:\Users\Rishab\ebsc23\application.py", line 2, in <module>
application = Flask(_name_)
NameError: name '_name_' is not defined. Did you mean: '__name__'?
(test1) C:\Users\Rishab\ebsc23>Flask run
* Serving Flask app 'application.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
Search

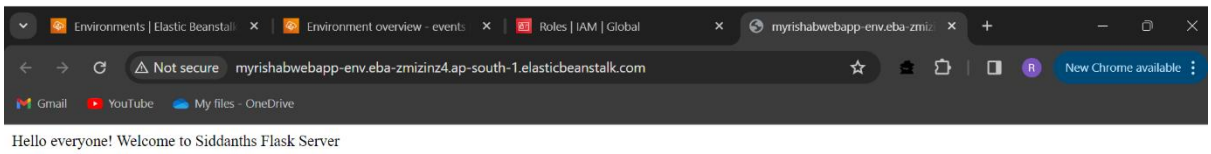

Environments | Elastic Beanstalk
Configure environment | Elastic
ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-environment
New Chrome available
Gmail
YouTube
My files - OneDrive
aws
Services
Search
[Alt+S]
Mumbai
RishabMandal
Platform
Python
Platform branch
Python 3.11 running on 64bit Amazon Linux 2023
Platform version
4.0.9 (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.
Version label
Source code origin. Maximum size 500 MB
Local file
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
11:22
01-03-2024
Environments | Elastic Beanstalk
Configure environment | Elastic
ap-south-1.console.aws.amazon.com/elasticbeanstalk/home?region=ap-south-1#/create-environment
New Chrome available
Gmail
YouTube
My files - OneDrive
aws
Services
Search
[Alt+S]
Mumbai
RishabMandal
4.0.9 (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.
v1
Source code origin. Maximum size 500 MB
Local file
Upload application
Choose file
File name: ebstest1.zip
File must be less than 500MB max file size
Public S3 URL
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Search
ENG
IN
11:23
01-03-2024

Step 5: Click on Environments -> Check the health of Environment wait till it becomes ‘OK’



Step 6: Click the URL





Conclusion:

This experiment demonstrates the benefits of using Amazon AWS Elastic Beanstalk Service for deploying and managing web applications. Its automation features simplify infrastructure management, while support for various languages and frameworks offers flexibility to developers. Additionally, built-in elastic load balancing enhances application performance and scalability.