IEN: - **Batch: -**

Name of the Student: - **Div.: -**

Date of Performance: -

Course Outcome: - CSL 605:

Experiment No.:5

Aim: To study and Implement Platform As A Service using AWS Elastic Benstalk

Software Required:
Theory :-
AWS 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.

You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

There is no additional charge for Elastic Beanstalk - you pay only for the AWS resources needed to store and run your applications.

Steps To use AWS Elastic Benstalk :
Setting up: Create an AWS account

If you're not already an AWS customer, you need to create an AWS account. Signing up enables you to access Elastic Beanstalk and other AWS services that you need.

To sign up for an AWS account

1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.

Step 1: Create an example application

In this step, you create a new application starting from a pre-existing example application. Elastic Beanstalk supports platforms for different programming languages, application servers, and Docker containers. You choose a platform when you create the application.

Create an application and an environment

To create your example application, you'll use the **Create a web app** console wizard. It creates an Elastic Beanstalk application and launches an environment within it. An environment is the collection of AWS resources required to run your application code.

To create an example application

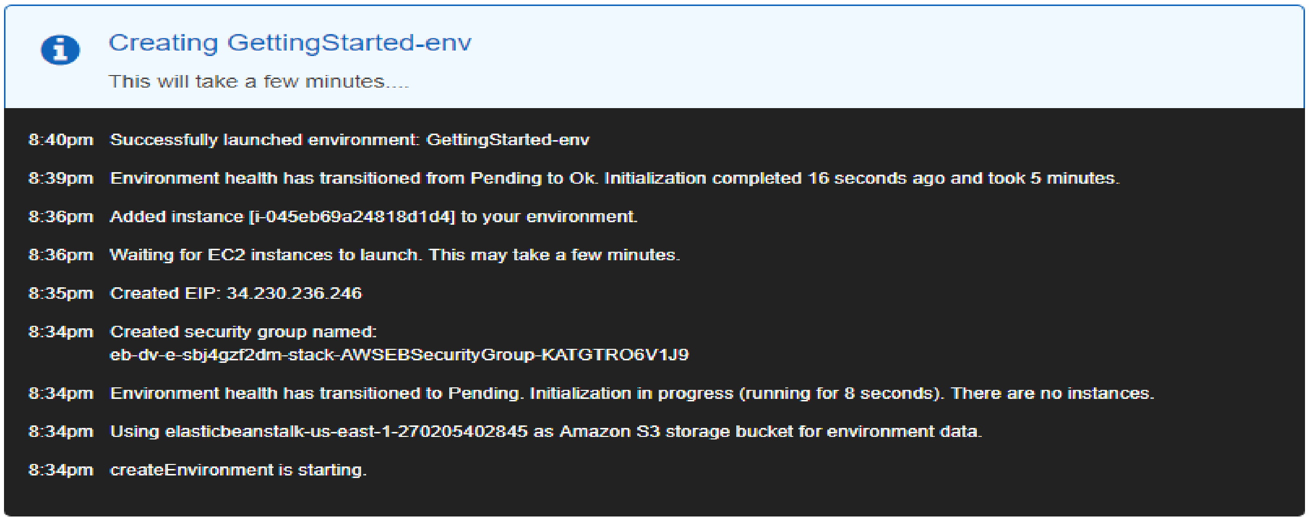
1. Open the Elastic Beanstalk console using this link: https://console.aws.amazon.com/elasticbeanstalk/home#/gettingStarted?applicationName=getting-started-app
2. Optionally add application tags.
3. For **Platform**, choose a platform, and then choose **Create application**.

To run the example application on AWS resources, Elastic Beanstalk takes the following actions. They take about five minutes to complete.

1. Creates an Elastic Beanstalk application named **getting-started-app**.
2. Launches an environment named **Getting Started App-env** with these AWS resources:
   - An Amazon Elastic Compute Cloud (Amazon EC2) instance (virtual machine)
   - An Amazon EC2 security group
   - An Amazon Simple Storage Service (Amazon S3) bucket
   - Amazon Cloud Watch alarms
   - An AWS Cloud Formation stack
   - A domain name

   For details about these AWS resources, see AWS resources created for the example application.
3. Creates a new application version named **Sample Application**. This is the default Elastic Beanstalk example application file.
4. Deploys the code for the example application to the **Getting Started App-env** environment.

During the environment creation process, the console tracks progress and displays events.



```
8:40pm   Successfully launched environment: GettingStarted-env
8:39pm   Environment health has transitioned from Pending to Ok. Initialization completed 16 seconds ago and took 5 minutes.
8:36pm   Added instance [i-045eb69a24818d1d4] to your environment.
8:36pm   Waiting for EC2 instances to launch. This may take a few minutes.
8:35pm   Created EIP: 34.230.236.246
8:34pm   Created security group named:
         eb-dv-e-sbj4gzf2dm-stack-AWSEBSecurityGroup-KATGTRO6V1J9
8:34pm   Environment health has transitioned to Pending. Initialization in progress (running for 8 seconds). There are no instances.
8:34pm   Using elasticbeanstalk-us-east-1-270205402845 as Amazon S3 storage bucket for environment data.
8:34pm   createEnvironment is starting.
```

When all of the resources are launched and the EC2 instances running the application pass health checks, the environment's health changes to Ok. You can now use your web application's website.

AWS resources created for the example application

When you create the example application, Elastic Beanstalk creates the following AWS resources:

- **EC2 instance** – An Amazon EC2 virtual machine configured to run web apps on the platform you choose.

Each platform runs a different set of software, configuration files, and scripts to support a specific language version, framework, web container, or combination thereof. Most platforms use either Apache or nginx as a reverse proxy that processes web traffic in front of your web app, forwards requests to it, serves static assets, and generates access and error logs.

- **Instance security group** – An Amazon EC2 security group configured to allow incoming traffic on port 80. This resource lets HTTP traffic from the load balancer reach the EC2 instance running your web app. By default, traffic is not allowed on other ports.
- **Amazon S3 bucket** – A storage location for your source code, logs, and other artifacts that are created when you use Elastic Beanstalk.
- **Amazon Cloud Watch alarms** – Two Cloud Watch alarms that monitor the load on the instances in your environment and are triggered if the load is too high or too low. When an alarm is triggered, your Auto Scaling group scales up or down in response.
- **AWS Cloud Formation stack** – Elastic Beanstalk uses AWS Cloud Formation to launch the resources in your environment and propagate configuration changes. The resources are defined in a template that you can view in the AWS Cloud Formation console.
- **Domain name** – A domain name that routes to your web app in the form *subdomain.region.elasticbeanstalk.com*.

Step 2: Explore your environment

To see an overview of your Elastic Beanstalk application's environment, use the environment page in the Elastic Beanstalk console.

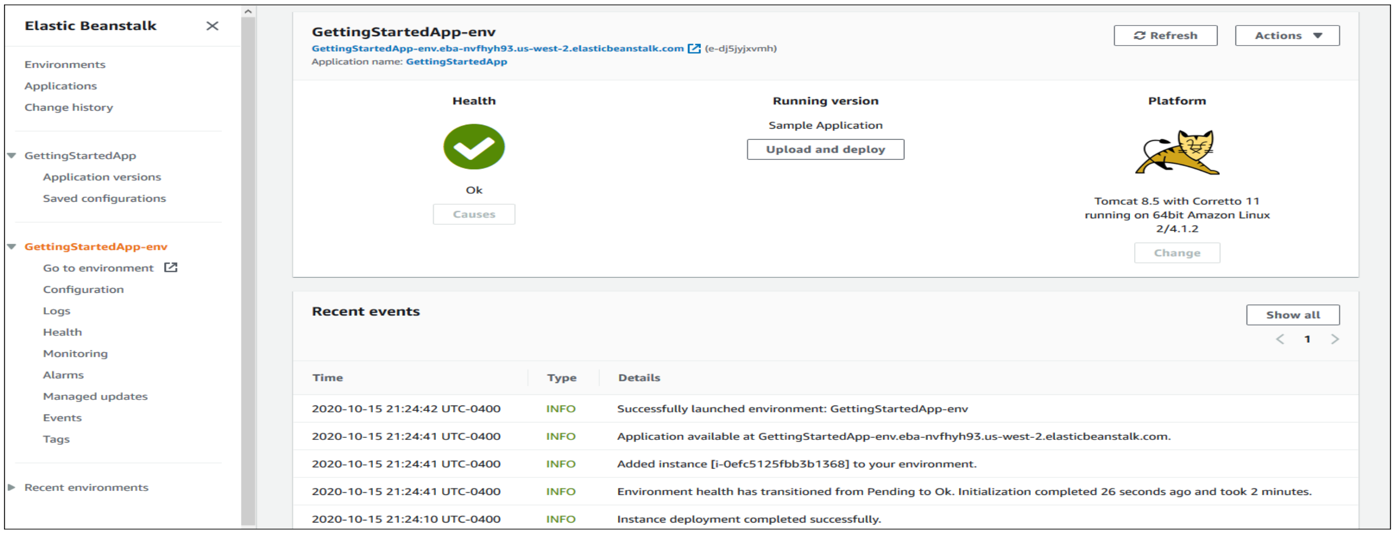
To view the environment overview

1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.

Note:- If you have many environments, use the search bar to filter the environment list.

The environment overview pane shows top level information about your environment. This includes its name, its URL, its current health status, the name of the currently deployed application version, and the platform version that the application is running on. Below the overview pane you can see the five most recent environment events.

To learn more about environment tiers, platforms, application versions, and other Elastic Beanstalk concepts, see Elastic Beanstalk concepts.



GettingStartedApp-env
GettingStartedApp-env.eba-nvfhyh93.us-west-2.elasticbeanstalk.com (e-dj5jyjxvmh)
Application name: GettingStartedApp

Health: Ok

Running version: Sample Application

Platform: Tomcat 8.5 with Corretto 11 running on 64bit Amazon Linux 2/4.1.2

Recent events

Time	Type	Details
2020-10-15 21:24:42 UTC-0400	INFO	Successfully launched environment: GettingStartedApp-env
2020-10-15 21:24:41 UTC-0400	INFO	Application available at GettingStartedApp-env.eba-nvfhyh93.us-west-2.elasticbeanstalk.com.
2020-10-15 21:24:41 UTC-0400	INFO	Added instance [i-0efc5125fbb3b1368] to your environment.
2020-10-15 21:24:41 UTC-0400	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 26 seconds ago and took 2 minutes.
2020-10-15 21:24:10 UTC-0400	INFO	Instance deployment completed successfully.

While Elastic Beanstalk creates your AWS resources and launches your application, the environment is in a Pending state. Status messages about launch events are continuously added to the overview.

The environment's **URL** is located at the top of the overview, below the environment name. This is the URL of the web application that the environment is running. Choose this URL to get to the example application's *Congratulations* page.

The navigation page on the left side of the console links to other pages that contain more detailed information about your environment and provide access to additional features:

- **Configuration** – Shows the resources provisioned for this environment, such as the Amazon Elastic Compute Cloud (Amazon EC2) instances that host your application. You can configure some of the provisioned resources on this page.

- **Health** – Shows the status of and detailed health information about the Amazon EC2 instances running your application.
- **Monitoring** – Shows statistics for the environment, such as average latency and CPU utilization. You can use this page to create alarms for the metrics that you are monitoring.
- **Events** – Shows information or error messages from the Elastic Beanstalk service and from other services whose resources this environment uses.
- **Tags** – Shows environment tags and allows you to manage them. Tags are key-value pairs that are applied to your environment.

Step 3: Deploy a new version of your application

Periodically, you might need to deploy a new version of your application. You can deploy a new version at any time, as long as no other update operations are in progress on your environment.

The application version that you started this tutorial with is called **Sample Application**.

To update your application version

1. Download the sample application that matches your environment's platform. Use one of the following applications.
    - **Docker** – docker.zip

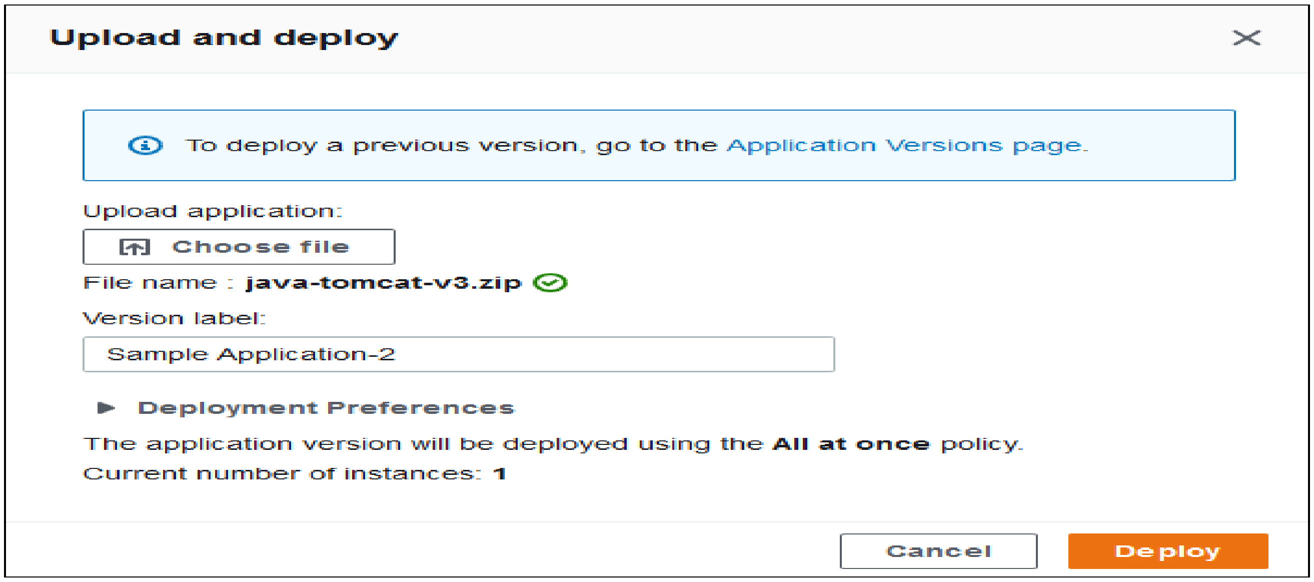
- **Multicontainer Docker** – docker-multicontainer-v2.zip
- **Preconfigured Docker (Glassfish)** – docker-glassfish-v1.zip
- **Go** – go.zip
- **Corretto** – corretto.zip
- **Tomcat** – tomcat.zip
- **.NET Core on Linux** – dotnet-core-linux.zip
- **.NET** – dotnet-asp-v1.zip
- **Node.js** – nodejs.zip
- **PHP** – php.zip
- **Python** – python.zip
- **Ruby** – ruby.zip

2. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
3. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.

   Note

   If you have many environments, use the search bar to filter the environment list.

4. On the environment overview page, choose **Upload and deploy**.
5. Choose **Choose file**, and then upload the sample application source bundle that you downloaded.



Upload and deploy

To deploy a previous version, go to the Application Versions page.

Upload application:

Choose file

File name : java-tomcat-v3.zip

Version label:

Sample Application-2

Deployment Preferences

The application version will be deployed using the **All at once** policy.

Current number of instances: 1

Cancel Deploy

   The console automatically fills in the **Version label** with a new unique label. If you type in your own version label, ensure that it's unique.

6. Choose **Deploy**.

While Elastic Beanstalk deploys your file to your Amazon EC2 instances, you can view the deployment status on the environment's overview. While the application version is updated, the **Environment Health** status is gray. When the deployment is complete, Elastic Beanstalk performs an application health check. When the application responds to the health check, it's considered healthy and the status returns to green. The environment overview shows the new **Running Version**—the name you provided as the **Version label**.

Elastic Beanstalk also uploads your new application version and adds it to the table of application versions. To view the table, choose **Application versions** under **getting-started-app** on the navigation pane.

Step 4: Configure your environment

You can configure your environment to better suit your application. For example, if you have a compute-intensive application, you can change the type of Amazon Elastic Compute Cloud (Amazon EC2) instance that is running your application. To apply configuration changes, Elastic Beanstalk performs an environment update.

Some configuration changes are simple and happen quickly. Some changes require deleting and recreating AWS resources, which can take several minutes. When you change configuration settings, Elastic Beanstalk warns you about potential application downtime.

Make a configuration change

In this example of a configuration change, you edit your environment's capacity settings. You configure a load-balanced, scalable environment that has between two and four Amazon EC2 instances in its Auto Scaling group, and then you verify that the change occurred. Elastic Beanstalk creates an additional Amazon EC2 instance, adding to the single instance that it created initially. Then, Elastic Beanstalk associates both instances with the environment's load balancer. As a result, your application's responsiveness is improved and its availability is increased.

To change your environment's capacity

1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.

Note:- If you have many environments, use the search bar to filter the environment list.

3. In the navigation pane, choose **Configuration**.
4. In the **Capacity** configuration category, choose **Edit**.
5. In the **Auto Scaling group** section, change **Environment type** to **Load balanced**.
6. In the **Instances** row, change **Max** to **4**, and then change **Min** to **2**.
7. Choose **Apply**.
8. A warning tells you that this update replaces all of your current instances. Choose **Confirm**.
9. In the navigation pane, choose **Events**.

   The environment update can take a few minutes. To find out that it's complete, look for the event **Successfully deployed new configuration to environment** in the event list. This confirms that the Auto Scaling minimum instance count has been set to 2. Elastic Beanstalk automatically launches the second instance.

   Verify the configuration change

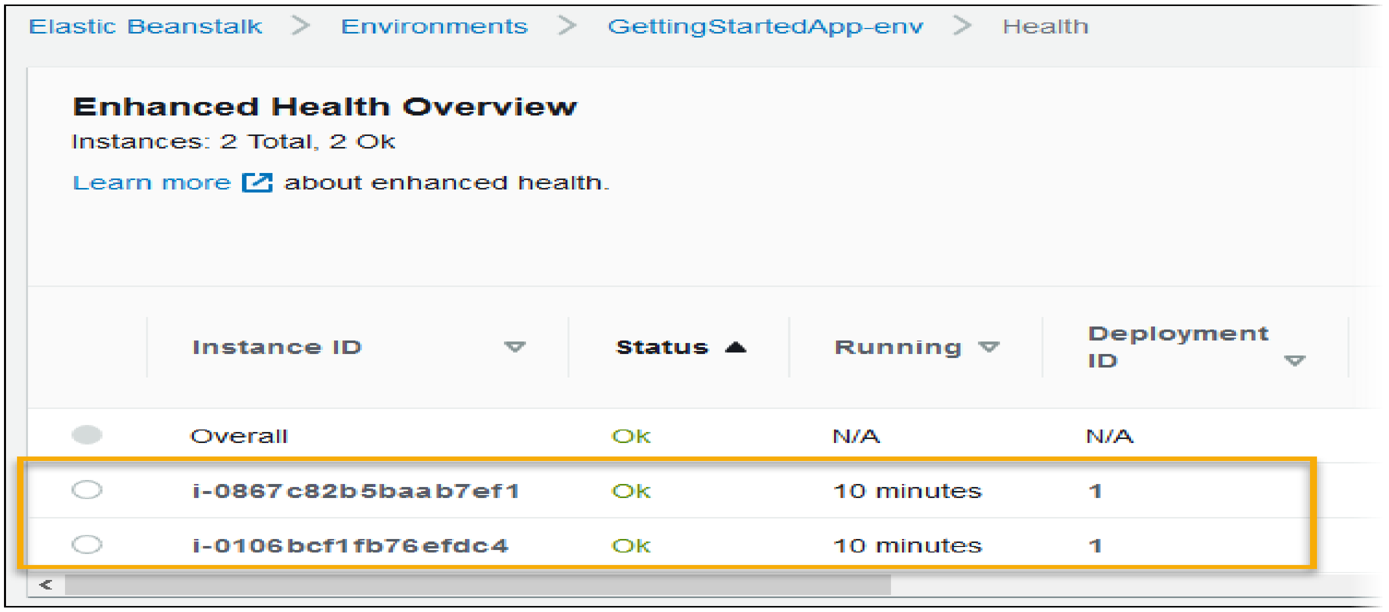
When the environment update is complete and the environment is ready, verify your change.

To verify the increased capacity

1. In the navigation pane, choose **Health**.

2. Look at the **Enhanced health overview** page.

   You can see that two Amazon EC2 instances are listed following the **Overall** line. Your environment capacity has increased to two instances.



Elastic Beanstalk > Environments > GettingStartedApp-env > Health

Enhanced Health Overview

Instances: 2 Total, 2 Ok

Learn more about enhanced health.

	Instance ID	Status	Running	Deployment ID
	Overall	Ok	N/A	N/A
	i-0867c82b5baab7ef1	Ok	10 minutes	1
	i-0106bcf1fb76efdc4	Ok	10 minutes	1

Step 5: Clean up

Congratulations! You have successfully deployed a sample application to the AWS Cloud, uploaded a new version, and modified its configuration to add a second Auto Scaling instance. To ensure that you're not charged for any services you aren't using, delete all application versions and terminate the environment. This also deletes the AWS resources that the environment created for you.

To delete the application and all associated resources

1. Delete all application versions.
   a. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
   b. In the navigation pane, choose **Applications**, and then choose **getting-started-app**.
   c. In the navigation pane, find your application's name and choose **Application versions**.
   d. On the **Application versions** page, select all application versions that you want to delete.
   e. Choose **Actions**, and then choose **Delete**.
   f. Turn on **Delete versions from Amazon S3**.
   g. Choose **Delete**, and then choose **Done**.
2. Terminate the environment.
   a. In the navigation pane, choose **getting-started-app**, and then choose **Getting Started App-env** in the environment list.
   b. Choose **Environment actions**, and then choose **Terminate Environment**.
   c. Confirm that you want to terminate **Getting Started App-env** by typing the environment name, and then choose **Terminate**.
3. Delete the getting-started-app application.

a. In the navigation pane, choose the **getting-started-app**.
b. Choose **Actions**, and then choose **Delete application**.
c. Confirm that you want to delete **getting-started-app** by typing the application name, and then choose **Delete**.

Conclusion:

Thus we have successfully uploaded static php website using AWS EBS and concluded the following points:

○ Platform as a Service provides an environment for the developers to create, host, and deploy an application.

○ The cloud platform companies remove the complexities by configuring and managing the elements such as database and servers.

○ This helps the customer to focus on the application without thinking of other problems.

○ The vendor modifies the development tools as per their requirement.

Experiment Rubric:

Evaluation Criteria	**Marks**	**Signature of Instructor with Date**
Lab Performance		
Topic Knowledge		
Task Conclusion		
Attainment Level (Out of 3)		