

What is AWS Elastic Beanstalk?

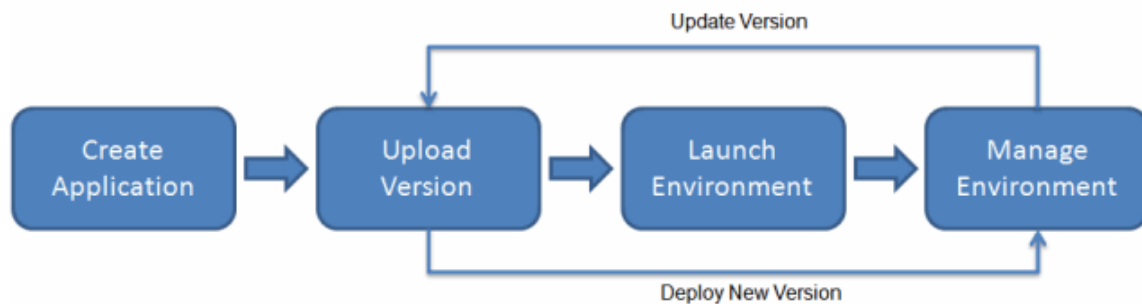
Amazon Web Services (AWS) has over hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility as we want to manage our AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, we can quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. We upload our application and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling and application health monitoring. Elastic Beanstalk uses highly reliable and scalable services that are available in the AWS Free Tier.

Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python and Ruby, as well as different platform configurations for each language. A configuration defines the infrastructure and software stack to be used for a given environment. When we deploy our application, Elastic Beanstalk provisions one or more AWS resources, such as Amazon EC2 instances in order to run the application. The software stack that runs on our Amazon EC2 instances depends on the configuration. For example, Elastic Beanstalk supports two configurations for the Java SE platform: one running Java 7, and the other running Java 8. We can interact with Elastic Beanstalk by using the AWS Management Console or the AWS Command Line Interface (AWS CLI).

To use Elastic Beanstalk, we create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run our code. After our environment is launched, we can then manage our environment and deploy new versions of the application.

The following diagram illustrates the workflow of Elastic Beanstalk.



After we create and deploy our application, information about the application—including metrics, events and environment status—is available through the AWS Management Console, APIs or Command Line Interfaces (CLI).

Elastic Beanstalk provides developers and systems administrators an easy, fast way to deploy and manage their applications without having to worry about AWS infrastructure. Once our AWS resources are deployed, we can modify and update them in a controlled and predictable way, providing the same sort of version control over our AWS infrastructure that we exercise over our software.

Note: There is no additional charge for Elastic Beanstalk. We pay only for the underlying AWS resources that our application consumes.

Benefits of AWS Elastic Beanstalk

Below are some benefits that AWS Elastic Beanstalk offers over other PaaS services

Offers Quicker Deployment: Elastic Beanstalk offers developers the fastest and simplest way to deploy their application. Within minutes, the application will be ready to use without users having to deal with the underlying infrastructure or resource configuration.

Supports Multi-Tenant Architecture: AWS Elastic Beanstalk makes it possible for users to share their applications across different devices with high scalability and security. It provides a detailed report of application usage and user profiles.

Simplifies Operations: Beanstalk provisions and operates the infrastructure and manages the application stack. Developers have to just focus on developing code for their application rather than spending time on managing and configuring servers, databases, firewalls, and networks.

Offers Complete Resource Control: Beanstalk gives developers the freedom to select the AWS resources, like EC2 instance type, that are optimal for their application. It allows developers to retain full control over AWS resources and access them at any time.

AWS Elastic Beanstalk Components

There are certain key concepts which we will come across frequently when we deploy an application on Beanstalk. Let us have look at those concepts:

Application:

- An application in Elastic Beanstalk is conceptually similar to a folder
- An application is a collection of components including ***environments***, ***versions*** and ***environment configuration***

Application Version:

- An application version refers to a specific, labeled iteration of deployable code for a web application
- An application version points to an Amazon S3 object that contains the deployable code such as a Java web application WAR file

Environment:

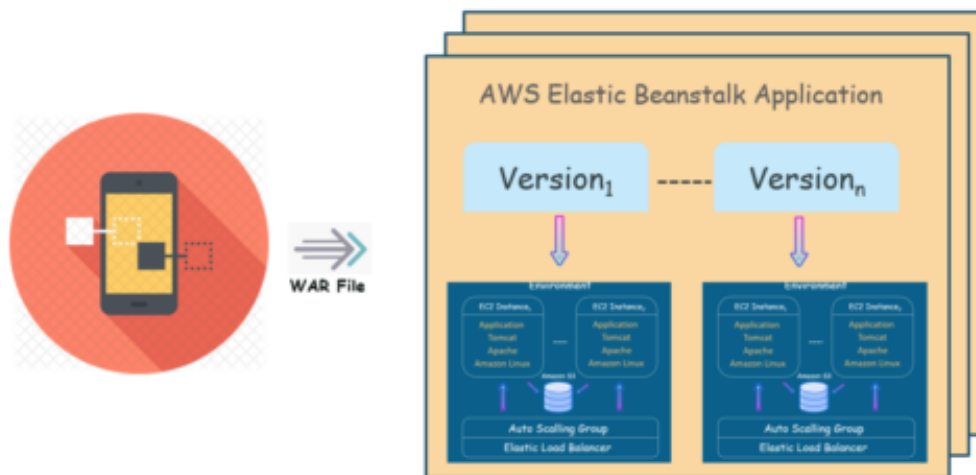
- Environments within Elastic Beanstalk Application is where the current version of the application will be active
- Each environment runs only a single application version at a time. But it is possible to run same or different versions of an application in many environments at the same time

Environment Tier:

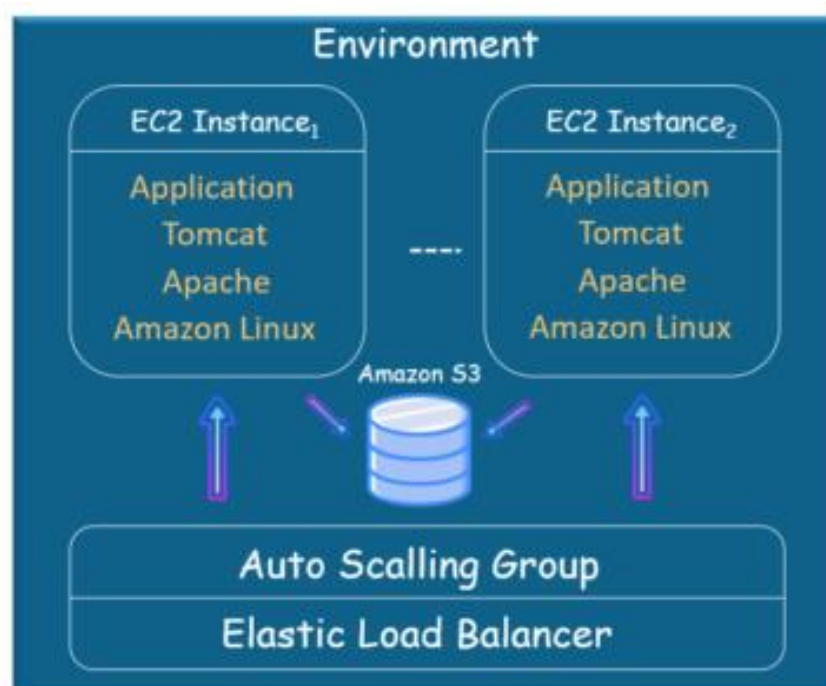
Based on requirement beanstalk offers two different Environment tiers: Web Server Environment, Worker Environment

- Web Server Environment: Handles HTTP requests from clients
- Worker Environment: Processes background tasks which are resource consuming and time intensive

Relation between Application, Application version and Environments:



How Beanstalk Environment using default container type looks like:



Now that we know about various key concepts pertaining to Elastic Beanstalk, let us understand the architecture of Elastic Beanstalk.

AWS Elastic Beanstalk Architecture

Before getting into AWS Elastic Beanstalk architecture, let us answer the most frequently asked question.

What is an Elastic Beanstalk Environment?

Environment refers to the current version of the application. When we launch an Environment for our application, Beanstalk asks us to choose among two different Environment Tiers i.e, *Web Server Environment* or *Worker Environment*. Let us understand them one by one.

Web Server Environment

Application version which is installed on the Web Server Environment handles HTTP requests from the client.

Beanstalk Environment

The Environment is the heart of the application. When you launch an Environment, Beanstalk assigns various resources that are needed to run the application successfully.

Elastic Load Balancer

When the application receives multiple requests from a client, Amazon Route53 forwards these requests to the Elastic Load Balancer. The load balancer distributes the requests among EC2 instances of Auto Scaling Group.

Auto Scaling Group

Auto Scaling Group automatically starts additional Amazon EC2 instances to accommodate increasing load on your application. If the load on your application decreases, Amazon EC2 Auto Scaling stops instances, but always leaves at least one instance running.

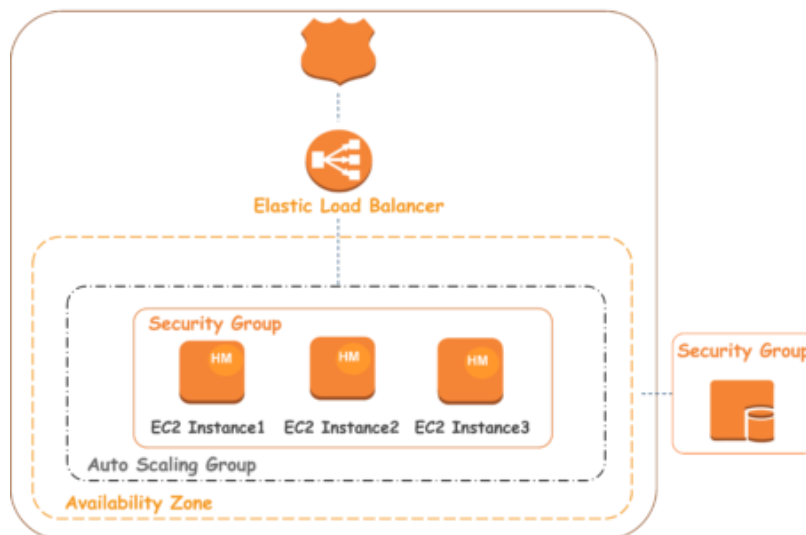
Host Manager

It is a software component which runs on every EC2 instance that has been assigned to your application. The host manager is responsible for various things like

- Generating and monitoring application log files
- Generating instance level events
- Monitoring application server

Security Groups

Group is like a firewall for our instance. Elastic Beanstalk has a default security group, which allows the client to access the application using HTTP Port 80. It also provides us with an option where we can define security groups to the database server as well. The below image summarizes what we have learned about Web Server Environment.



If the application version installed on Web Server Tier keeps denying multiple requests because it has encountered time intensive and resource consuming tasks while handling a request? This is where Worker Tier comes into the picture.

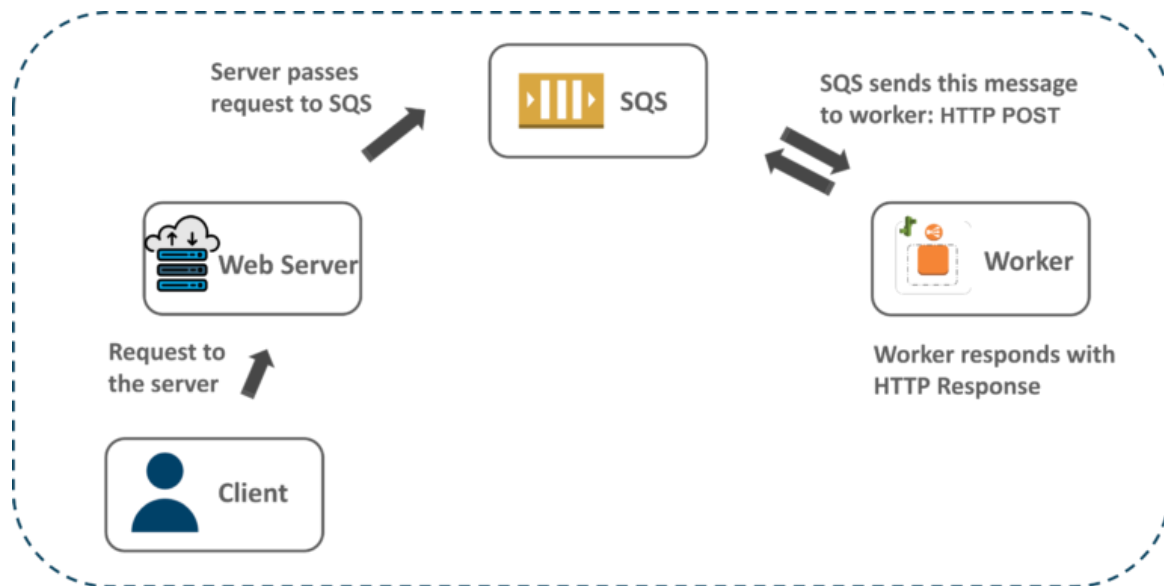
Worker Environment

A worker is a separate background process that assists Web Server Tier by handling resource-intensive or time-intensive operations. In addition, it also emails notifications, generates reports and cleans up databases. This makes it possible for the application to remain responsive and handle multiple requests.

How does Worker process know which tasks to handle and when? How do these two Environment tiers communicate? For that, we use a message queuing service by AWS called Amazon Simple Queue Service (SQS). The image below gives us a rough idea of how the worker process receives and handles background tasks.

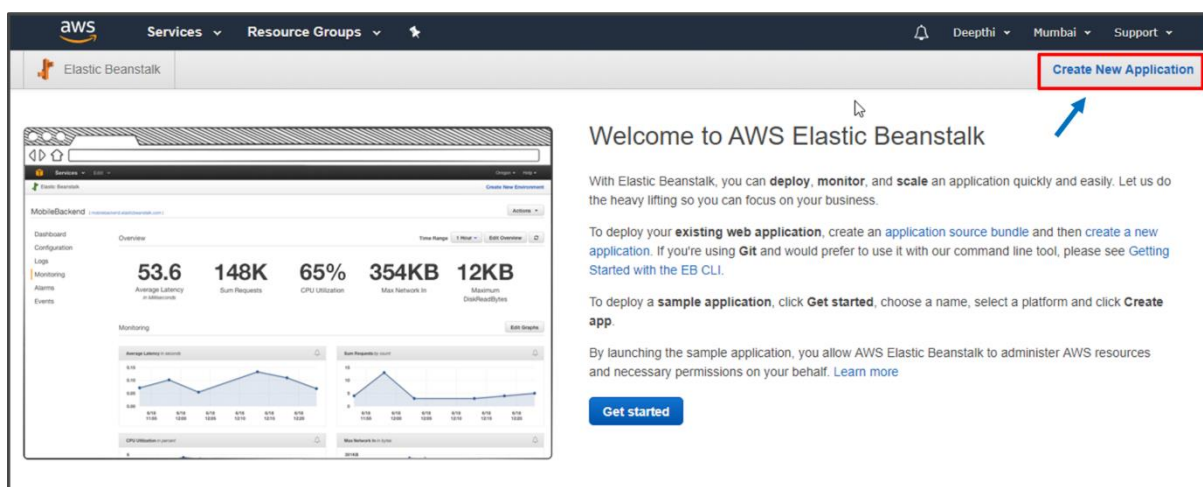
The workflow of the worker process is fairly simple. When we launch a Worker Environment tier, Elastic Beanstalk installs a daemon on each EC2 instance in the Auto Scaling group. The daemon pulls requests sent from an Amazon SQS queue. Based on the queue's priority, SQS will send the message

via a POST request to the HTTP Path of the Worker Environment. The worker on receiving the message executes the tasks and sends an HTTP response once the operation is done. SQS on receiving response message deletes the message in the queue. If it fails to receive a response, it will continuously retry sending the messages.

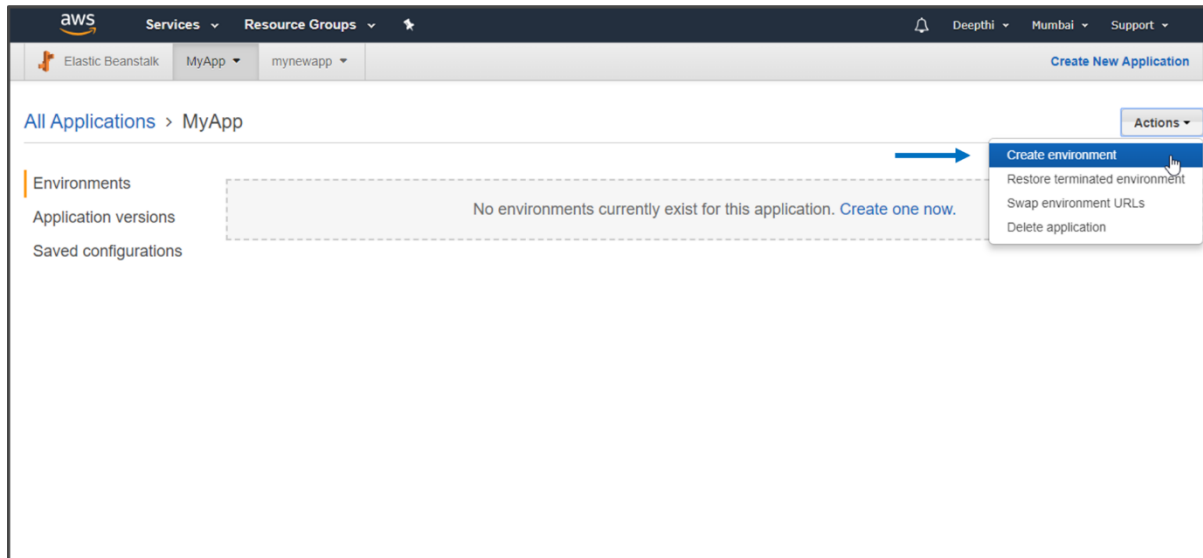


Deploy an Application on Elastic Beanstalk

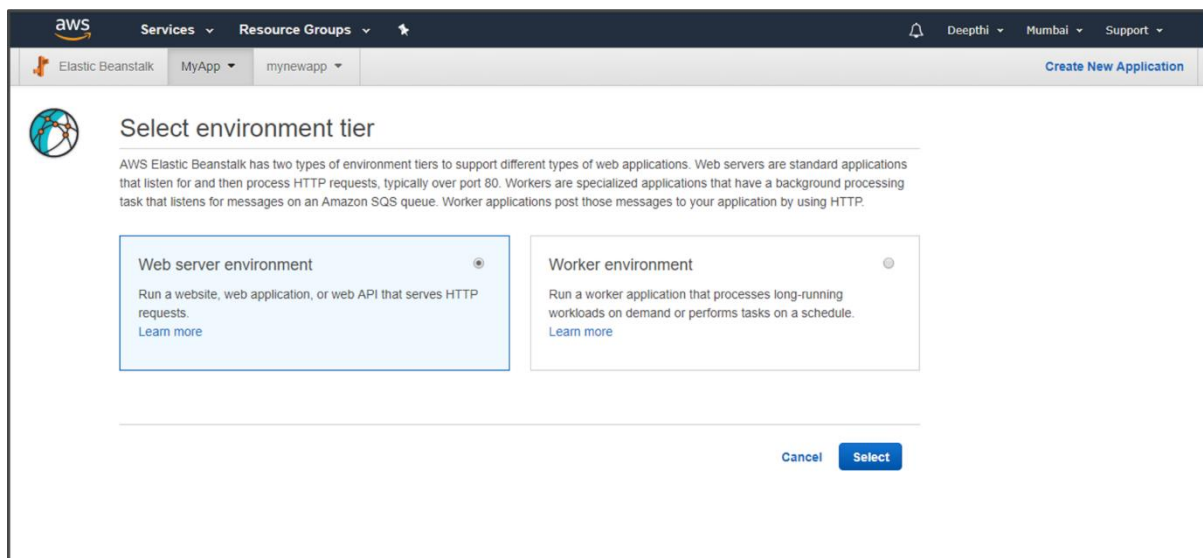
Step 1: On Elastic Beanstalk console click on *Create New Application* option. A dialog box appears where we can give a name and appropriate description for our application.



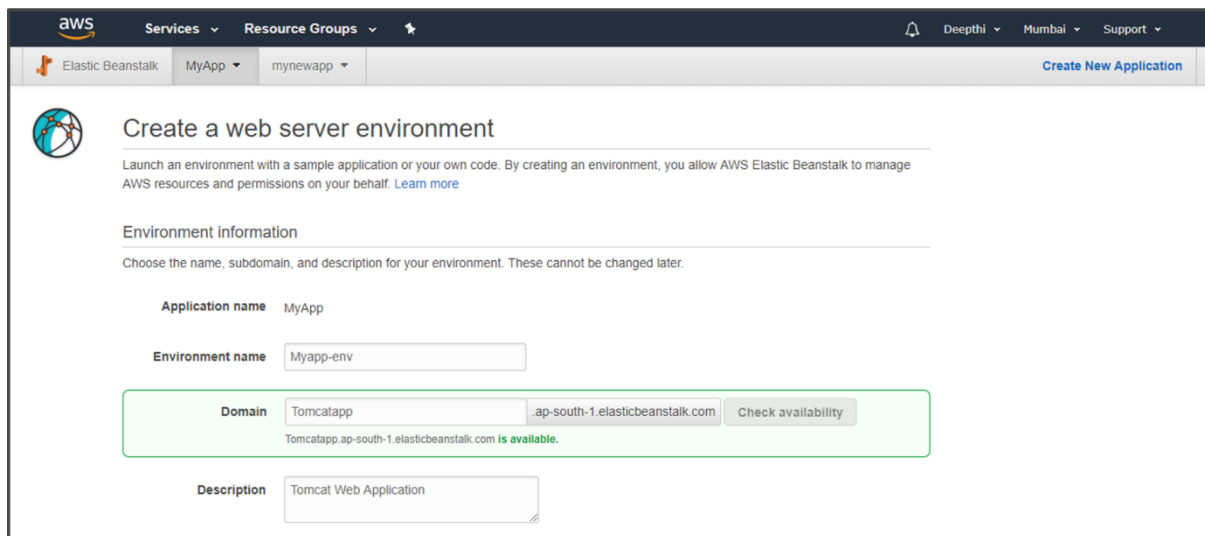
Step 2: Now that the application folder has been created, we can click on the *Actions tab* and select *Create Environment* option. Beanstalk provides us with an option where we can create multiple Environments for our application.



Step 3: Choose among two different Environment Tier options. Choose Web Server Environment if we want our application to handle HTTP requests or choose Worker Environment to handle background tasks.



Step 4: Another dialog appears, where we need to provide a domain name and description for our application.

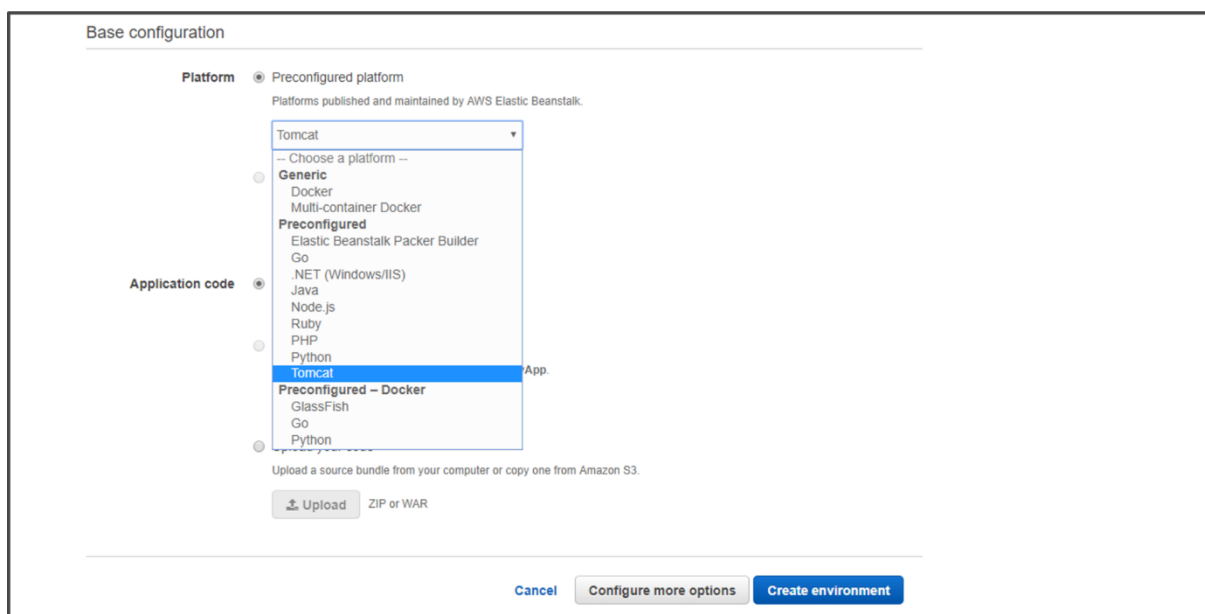


The screenshot shows the AWS Elastic Beanstalk console interface for creating a new environment. The page title is "Create a web server environment". Below the title, there is a brief description: "Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. [Learn more](#)".

The "Environment information" section contains the following fields:

- Application name:** MyApp
- Environment name:** Myapp-env
- Domain:** Tomcatapp, .ap-south-1.elasticbeanstalk.com. A "Check availability" button is next to it. Below the domain field, a message states: "Tomcatapp.ap-south-1.elasticbeanstalk.com is available."
- Description:** Tomcat Web Application

Step 5: Choose a platform of our choice for your application. Elastic Beanstalk will provide us with multiple options. We can choose a sample application provided by Beanstalk, or upload a file which has code for our application.



The screenshot shows the "Base configuration" dialog in the AWS Elastic Beanstalk console. It has two main sections: "Platform" and "Application code".

Platform: The "Preconfigured platform" radio button is selected. Below it, a dropdown menu is open, showing the following options:

- Tomcat
- Choose a platform --
- Generic**
- Docker
- Multi-container Docker
- Preconfigured**
- Elastic Beanstalk Packer Builder
- Go
- .NET (Windows/IIS)
- Java
- Node.js
- Ruby
- PHP
- Python
- Tomcat
- Preconfigured – Docker**
- GlassFish
- Go
- Python

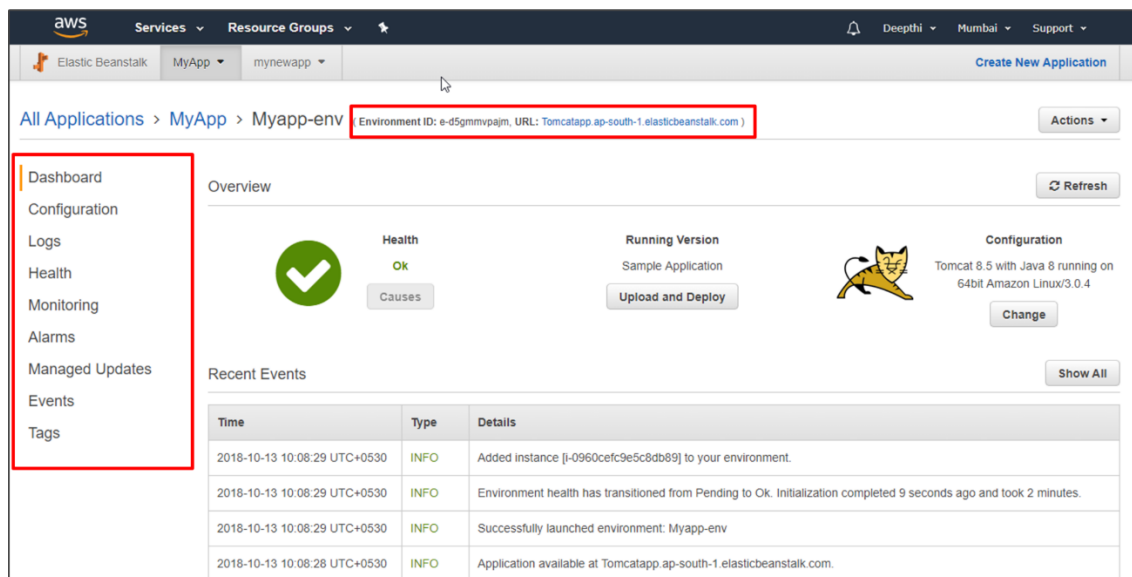
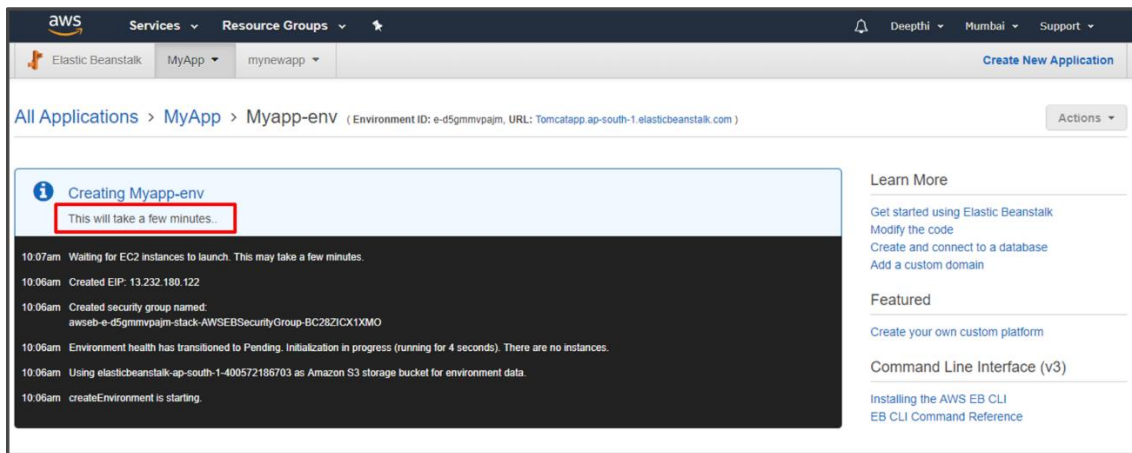
Application code: The "Preconfigured" radio button is selected. Below it, a dropdown menu is open, showing the following options:

- Tomcat
- Preconfigured – Docker
- GlassFish
- Go
- Python

At the bottom of the dialog, there is an "Upload" button and a note: "Upload a source bundle from your computer or copy one from Amazon S3." Below the "Upload" button, there is a "ZIP or WAR" label.

At the bottom of the dialog, there are three buttons: "Cancel", "Configure more options", and "Create environment".

Beanstalk will take a few minutes to launch an Environment. Once the Environment is launched, on the navigation pane we can see multiple options where we can change the configuration of our application, view log files, and events. Since we are already on Environment page, try exploring different features that Beanstalk offers.



Step 6: On the top right corner, you will find the URL of your application version. Click on that URL. You will be taken to a page which will confirm that you have successfully launched your application on Elastic Beanstalk.

