AWS Elastic Beanstalk can be used to quickly deploy and manage applications in the AWS Cloud.

Developers upload applications and Elastic Beanstalk handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

You can use multiple availability zones to improve application reliability and availability.

Considered a Platform as a Service (PaaS) solution.

Supports the following platforms:

* [Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.docker)
* [Multicontainer Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.mcdocker)
* [Preconfigured Docker](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.dockerpreconfig)
* [Go](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.go)
* [Java SE](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.javase)
* [Tomcat](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.java)
* [.NET Core on Linux](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.dotnetlinux)
* [.NET on Windows Server](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.net)
* [Node.js](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.nodejs)
* [PHP](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.PHP)
* [Python](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.python)
* [Ruby](https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.ruby)

Developers can focus on writing code and don’t need to worry about deploying infrastructure.

You pay only for the resources provisioned, not for Elastic Beanstalk itself.

Elastic Beanstalk automatically scales your application up and down.

You can select the EC2 instance type that is optimal for your application.

Can retain full administrative control or have Elastic Beanstalk do it for you.

The Managed Platform Updates feature automatically applies updates for your operating system and platform.

Elastic Beanstalk monitors and manages application health and information is viewable via a dashboard.

Integrated with CloudWatch and X-Ray for performance data and metrics.

Integrates with Amazon VPC and AWS IAM.

Can provision most database instances.

Stores your application files and, optionally, server log files in Amazon S3.

Application data can also be stored on S3.

Multiple environments are supported to enable versioning.

Changes from Git repositories are replicated.

Linux and Windows AMI support.

Code is deployed using a WAR file or Git repository.

Can use the AWS toolkit for Visual Studio and the AWS toolkit for Eclipse to deploy Elastic Beanstalk.

Fault tolerance within a single region.

By default applications are publicly accessible.

Can access logs without logging into application servers.

Provides ISO, PCI, SOC 1, SOC 2, and SOC 3 compliance along with the criteria for HIPAA eligibility.

Supports AWS Graviton arm64-based processors.



## **Elastic Beanstalk Layers**

There are several layers that make up Elastic Beanstalk and each layer is described below:

**Application**:

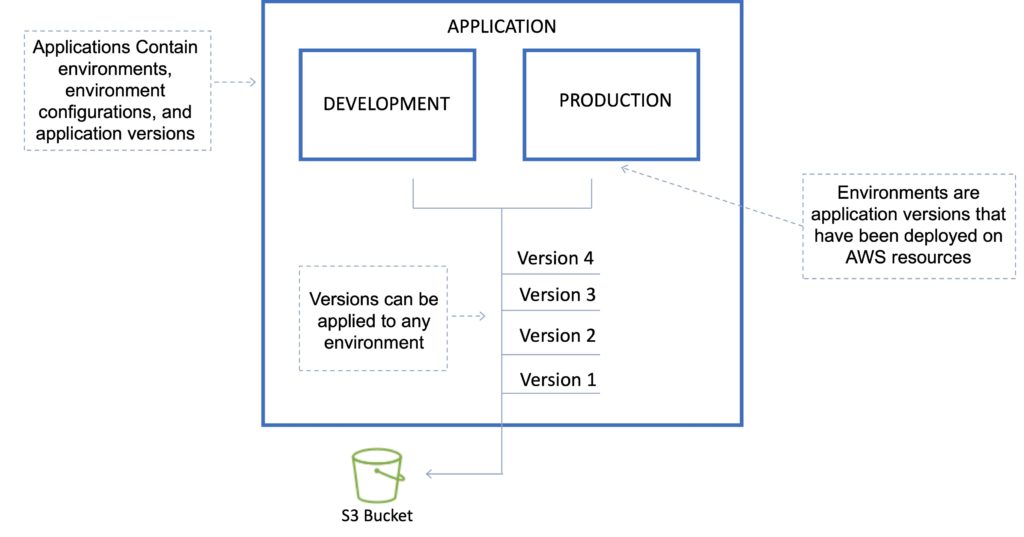
* Within Elastic Beanstalk, an application is a collection of different elements, such as environments, environment configurations, and application versions.
* You can have multiple application versions held within an application.

**Application version**:

* An application version is a very specific reference to a section of deployable code.
* The application version will point typically to an Amazon s3 bucket containing the code.

**Environment**:

* An environment refers to an application version that has been deployed on AWS resources.
* The resources are configured and provisioned by AWS Elastic Beanstalk.
* The environment is comprised of all the resources created by Elastic Beanstalk and not just an EC2 instance with your uploaded code.



**Environment tier**:

* Determines how Elastic Beanstalk provisions resources based on what the application is designed to do.
* **Web servers** are standard applications that listen for and then process HTTP requests, typically over port 80.
* **Workers** are specialized applications that have a background processing task that listens for messages on an Amazon SQS queue.

**Environment configurations**:

* An environment configuration is a collection of parameters and settings that dictate how an environment will have its resources provisioned by Elastic Beanstalk and how these resources will behave.

**Configuration template**:

* This is a template that provides the baseline for creating a new, unique environment configuration.

## **Deployment Options**

AWS Elastic Beanstalk provides several options for how [**deployments**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html) are processed, including deployment policies and options that let you configure batch size and health check behavior during deployments.

### **Deployment options**

Single instance: great for development.

High availability with load balancer: great for production.

### **Deployment policies**

The deployment policies are: All at once, Rolling, Rolling with additional batch, and Immutable.

All at once:

* Deploys the new version to all instances simultaneously.
* All your instances are out of service while the deployment takes place.
* Fastest deployment.
* Good for quick iterations in the development environment.
* You will experience an outage while the deployment is taking place – not ideal for mission-critical systems.
* If the update fails, you need to roll back the changes by re-deploying the original version to all your instances.
* No additional cost.

Rolling:

* Update a few instances at a time (batch), and then move onto the next batch once the first batch is healthy (downtime for 1 batch at a time).
* The application is running both versions simultaneously.
* Each batch of instances is taken out of service while the deployment takes place.
* Your environment capacity will be reduced by the number of instances in a batch while the deployment takes place.
* Not ideal for performance-sensitive systems.
* If the update fails, you need to perform an additional rolling update to roll back the changes.
* No additional cost.
* Long deployment time.

Rolling with additional batch:

* Like Rolling but launches new instances in a batch ensuring that there is full availability.
* The application is running at capacity.
* You can set the bucket size.
* The application is running both versions simultaneously.
* Small additional cost.
* Additional batch is removed at the end of the deployment.
* Longer deployment.
* Good for production environments.

Immutable:

* Launches new instances in a new ASG and deploys the version update to these instances before swapping traffic to these instances once healthy.
* Zero downtime.
* New code is deployed to new instances using an ASG.
* High cost as double the number of instances running during updates.
* Longest deployment.
* Quick rollback in case of failures.
* Great for production environments.

Additionally, Elastic Beanstalk supports blue/green deployment.

Blue / Green deployment:

* This is not a feature within Elastic Beanstalk
* You create a new “staging” environment and deploy updates there.
* The new environment (green) can be validated independently, and you can roll back if there are issues.
* Route 53 can be set up using weighted policies to redirect a percentage of traffic to the staging environment.
* Using Elastic Beanstalk, you can “swap URLs” when done with the environment test.
* Zero downtime.

The following tables summarizes the different deployment policies:

| **Deployment Policy** | **Deploy Time** | **Zero Downtime** | **Rollback** | **Extra Cost** | **Reduction in capacity** |
| --- | --- | --- | --- | --- | --- |
| All at once | Clock | NO | Manual redeploy | NONE | YES (total) |
| Rolling | Clock Clock | YES | Manual redeploy | NONE | YES (batch size) |
| Rolling with additional batch | Clock Clock Clock | YES | Manual redeploy | YES (batch size) | NO |
| Immutable | Clock Clock Clock Clock | YES | Terminate new instances | YES (total) | NO |
| Blue/green | Clock Clock Clock Clock | YES | Swap URL | YES (varies) | NO |

## **Golden AMIs**

When deploying code to Amazon EC2 using Beanstalk, Elastic Beanstalk must resolve application dependencies which can take a long time.

A golden AMI is a method of reducing this time by packaging all dependencies, configuration, and software into the AMI before deploying.

## **Elastic Beanstalk CLI**

There is an additional CLI called “eb cli”.

The EB CLI is a command-line interface for AWS Elastic Beanstalk that provides interactive commands that simplify creating, updating, and monitoring environments from a local repository.

You can use the EB CLI as part of your everyday development and testing cycle as an alternative to the Elastic Beanstalk console.

## **Lifecycle Policies**

Elastic Beanstalk can store at most 1000 application versions.

To phase out old versions use a lifecycle policy:

* Time-based – specify max age.
* Count based – specify max number to retain.

Versions that are in use will not be deleted.

Option to not delete the source bundle in S3 to prevent data loss.

## **Worker environments**

If an application performs tasks that take a long time to complete (long-running tasks), offload to a worker environment.

It allows you to decouple your application tiers.

Can define periodic tasks in the cron.yaml file.

Elastic Beanstalk Worker Environment

## **Elastic Beanstalk Extensions**

You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application’s source code to configure your environment and customize the AWS resources that it contains.

Customization includes defining packages to install, create Linux users and groups, running shell commands, specifying services to enable, configuring a load balancer, etc.

Configuration files are YAML- or JSON-formatted documents with a .config file extension that you place in a folder named .ebextensions and deploy in your application source bundle.

The .ebextensions folder must be included in the top-level directory of your application source code bundle.

All the parameters set in the UI can be configured in the code.

Requirements:

* Must be in the .ebextensions/ directory of the source code.
* YAML or JSON format.
* .config extensions can be included (e.g. logging.config).
* You can modify some default settings using “option\_settings”.
* You can add resources such as RDS, ElastiCache, and DynamoDB.

Resources managed by .ebextensions get deleted if the environment is terminated.

## **Elastic Beanstalk with Amazon Relational Database Service (RDS)**

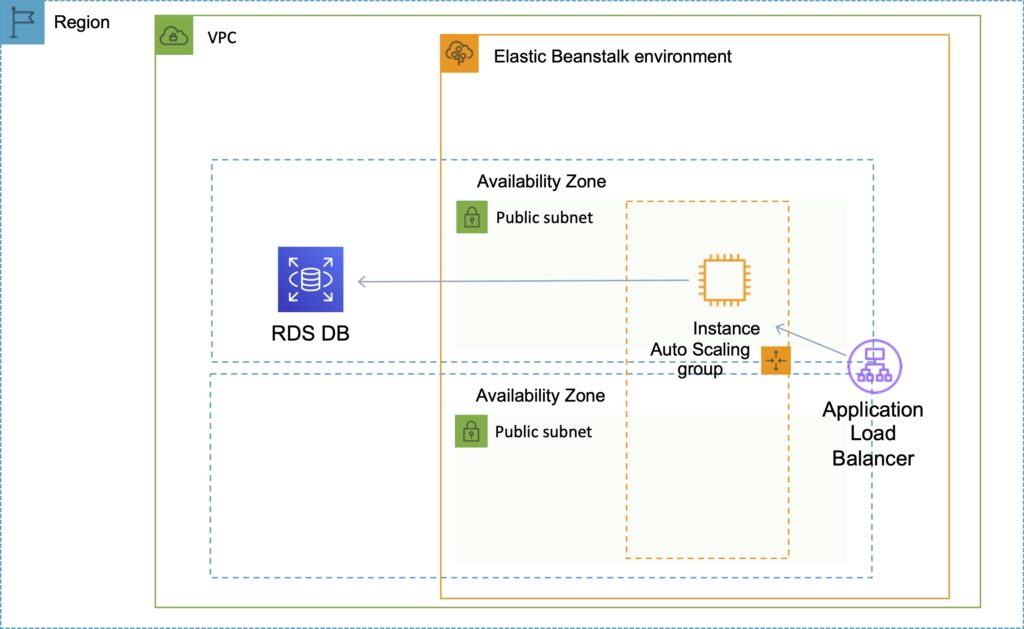
You can deploy Amazon RDS within an Elastic Beanstalk environment as in the diagram below:

Elastic Beanstalk Environment with RDS

However, if you terminate your Elastic Beanstalk environment you also lose the database.

The use case is only for development environments, typically not suitable for production.

For production, it is preferable to create the Amazon RDS database outside of Elastic Beanstalk as in the diagram below:



Steps to migrate from RDS within a Beanstalk environment to standalone RDS:

* Take a snapshot of the RDS DB.
* Enable deletion protection on the RDS DB.
* Create a new environment without an RDS DB and point applications to the existing RDS DB.
* Perform a blue/green deployment and swap the new and old environments.
* Terminate the old environment (RDS will not be deleted due to termination protection).
* Delete the CloudFormation stack (will be in the DELETE\_FAILED state).

### **Connecting to an Amazon RDS database**

When the environment update is complete, the DB instance’s hostname and other connection information are available to your application through the following environment properties:

* RDS\_HOSTNAME – The hostname of the DB instance.
* RDS\_PORT – The port on which the DB instance accepts connections. The default value varies among DB engines.
* RDS\_DB\_NAME – The database name, ebdb.
* RDS\_USERNAME – The user name that you configured for your database.
* RDS\_PASSWORD – The password that you configured for your database.

## **Custom Domain Names**

If you’re using AWS Elastic Beanstalk to deploy and manage applications in the AWS Cloud, you can use Amazon Route 53 to route DNS traffic for your domain, such as example.com, to a new or an existing Elastic Beanstalk environment.

You create either a *CNAME record* or an *alias record*, depending on whether the domain name for the environment includes the Region, such as **us-east-2**, in which you deployed the environment. New environments include the Region in the domain name; environments that were created before early 2016 do not.

If the domain name does NOT include the Region: create a CNAME record.

If the domain name DOES include the Region: create an Alias record.

## **Security**

Elastic Beanstalk works with HTTPS:

* Load the SSL certificate onto the load balancer.
* Can be performed from the console or in code (.ebextensions/securelistener-alb.config).
* SSL certificate can be provisioned using ACM or CLI.

For redirecting HTTP to HTTPS:

* Configure in the application.
* Configure the ALB with a rule.
* Ensure health checks are not redirected.

## **Monitoring and Reporting**

Elastic Beanstalk automatically uses [**Amazon CloudWatch**](https://digitalcloud.training/amazon-cloudwatch/) to help you monitor your application and environment status.

You can navigate to the Amazon CloudWatch console to see your dashboard and get an overview of all your resources as well as your alarms.

You can also choose to view more metrics or add custom metrics.

## **Logging and Auditing**

With CloudWatch Logs, you can monitor and archive your Elastic Beanstalk application, system, and custom log files from Amazon EC2 instances of your environments.

You can also configure alarms that make it easier for you to react to specific log stream events that your metric filters extract.

The CloudWatch Logs agent installed on each Amazon EC2 instance in your environment publishes metric data points to the CloudWatch service for each log group you configure.

Each log group applies its own filter patterns to determine what log stream events to send to CloudWatch as data points.

Log streams that belong to the same log group share the same retention, monitoring, and access control settings.

In addition to instance logs, if you enable [**enhanced health**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/health-enhanced.html) for your environment, you can configure the environment to stream health information to CloudWatch Logs.



## **Authorization and Access Control**

AWS Elastic Beanstalk supports [**identity-based**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.iam.html) policies.

AWS Elastic Beanstalk does not support [**resource-based**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_identity-vs-resource.html) policies.

AWS Elastic Beanstalk has partial support for [**resource-level**](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html) permissions.

When you create an environment, AWS Elastic Beanstalk prompts you to provide two AWS Identity and Access Management (IAM) roles: a service role and an instance profile.

The [**service role**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-service.html) is assumed by Elastic Beanstalk to use other AWS services on your behalf.

The [**instance profile**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-instance.html) is applied to the instances in your environment and allows them to retrieve [**application versions**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.html#concepts-version) from Amazon Simple Storage Service (Amazon S3), upload logs to Amazon S3, and perform other tasks that vary depending on the environment type and platform.

You can also create [**user policies**](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts-roles-user.html) and apply them to IAM users and groups in your account to allow users to create and manage Elastic Beanstalk applications and environments. Elastic Beanstalk provides managed policies for full access and read-only access.

### 

### **Related posts:**

* ****
* ****
* ****