Migrating Web Application to Cloud using AWS Elastic Beanstalk

**SPL-BE-200-MTELBS-1 - Version 1.0.14**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab Overview**

Your company has an existing web application hosted on-premises, and they have decided to move their existing infrastructure to the cloud with Amazon Web Services (AWS) as the cloud provider. As a Solutions Architect, you are tasked to analyze and evaluate different migration strategies and come up with a solution to migrate the application to the AWS cloud.

In this lab, you re-platform and migrate your web application to the Cloud using AWS Elastic Beanstalk.

ICON KEY

Various icons are used throughout this lab to call attention to certain aspects of the guide. The following list explains the purpose for each one:

* The keyboard icon specifies that you must run a command.
* The clipboard icon indicates that you can verify the output of a command or edited file by comparing it to the provided example.
* The note icon specifies important hints, tips, guidance, or advice.
* Calls attention to information of special interest or importance. Failure to read the note does not result in physical harm to the equipment or data, but could result in the need to repeat certain steps.
* The “i” circle icon specifies where to find more information.
* Suggests a moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.

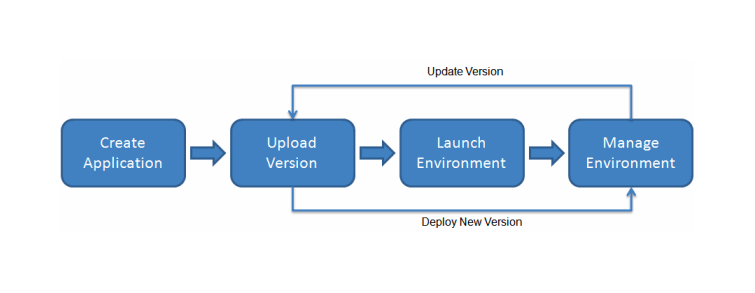
AWS ELASTIC BEANSTALK OVERVIEW

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.

To use Elastic Beanstalk, you 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 information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.



PERMISSIONS

When you create an environment, AWS Elastic Beanstalk prompts you to provide two AWS Identity and Access Management (IAM) roles:

* **Service role:** The service role is assumed by Elastic Beanstalk to use other AWS services on your behalf.
* **Instance profile:** The instance profile is applied to the instances in your environment and allows them to retrieve application versions 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.

 For this lab, the Service Role and the Instance Profile are already created as part of initial lab setup.

TOPICS COVERED

By the end of this lab, you will be able to:

* Prepare your application source code for migration to AWS.
* Upload your application source code version to an S3 Bucket.
* Create an Elastic Beanstalk application and launch an environment.
* Review and manage the Elastic Beanstalk environment.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you should:

* Have access to a computer with Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat).
* Have a modern internet browser such as Chrome or Firefox.
* Be familiar with basic navigation of the AWS Management Console.
* Be comfortable editing scripts using an AWS Cloud9 code editor, or using tools like vi and nano.

DURATION

This lab requires **60** minutes to complete.

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Prepare Source Code**

As Elastic Beanstalk is a managed service, you don’t need to deal with launching and managing servers or any underlying infrastructure. As a developer, you can directly push your codes into Elastic Beanstalk.

In this task, you:

* Install necessary toolset into your source machine.
* Prepare your source code to be uploaded into Elastic Beanstalk.

TASK 1.1: CONFIRM THE INSTALLATION OF NECESSARY TOOLSET INTO YOUR SOURCE MACHINE

In order to prepare the source environment for Elastic Beanstalk migration, you need to install some tools into your source machine. For this lab, those tools were already installed as part of the initial lab setup.

1. In the left navigation menu of the lab instructions page, choose **Download PEM** to download the EC2 key to your local machine.
2. At the top of the AWS Management Console, in the search bar, search for and choose

Cloud9

.

1. On the AWS Cloud9 **Environments** page, in the **Cloud9-Lab-IDE** row, choose the **Open** link.

You now upload the PEM key to the AWS Cloud9 environment and use that to connect to the source web server.

1. In the AWS Cloud9 environment, from the top left menu, choose **File** and then choose **Upload Local Files**.
2. Either drag and drop the PEM file or choose **Select files** and browse to the location where you downloaded the PEM file.
3. Close the **Upload Files** dialog box.
4. In the AWS Cloud9 terminal, run the following command to ensure that the PEM file is uploaded correctly:

cd ~/environment && ls

**Expected output:**

Ec2KeyPair-PEM.pem README.md

1. To change permissions of the PEM file so that only the root user can read it, replace the *PEM filename* in the following command and run the updated command in the AWS Cloud9 terminal:

chmod 400 <PEM Filename>

E.g.

chmod 400 Ec2KeyPair-PEM.pem

1. In the left navigation menu of the lab instructions page, copy the value of **WebServerDNSName**. You use this to connect to the source webserver in the next step.
2. From the AWS Cloud9 terminal, replace the *PEM Filename* and *WebServerDNSName* in the following command and run the updated command to connect to the source webserver using SSH:

ssh -i <PEM Filename> ubuntu@<WebServerDNSName>

E.g.

ssh -i Ec2KeyPair-PEM.pem ubuntu@ec2-35-88-138-8.us-west-2.compute.amazonaws.com

You are prompted with the following message:

Are you sure you want to continue connecting (yes/no)?

1. Enter

yes

 and you are connected to the source webserver.

1. Once connected to the source web server, enter the following commands one by one to confirm that **zip** and **awscli** were successfully installed:

**Command 1:**

zip

**Expected output:**

Copyright (c) 1990-2008 Info-ZIP - Type 'zip "-L"' for software license.

Zip 3.0 (July 5th 2008). Usage:

zip [-options] [-b path] [-t mmddyyyy] [-n suffixes] [zipfile list] [-xi list]

The default action is to add or replace zipfile entries from list, which

can include the special name - to compress standard input.

If zipfile and list are omitted, zip compresses stdin to stdout.

-f freshen: only changed files -u update: only changed or new files

-d delete entries in zipfile -m move into zipfile (delete OS files)

-r recurse into directories -j junk (don't record) directory names

-0 store only -l convert LF to CR LF (-ll CR LF to LF)

-1 compress faster -9 compress better

-q quiet operation -v verbose operation/print version info

-c add one-line comments -z add zipfile comment

-@ read names from stdin -o make zipfile as old as latest entry

-x exclude the following names -i include only the following names

-F fix zipfile (-FF try harder) -D do not add directory entries

-A adjust self-extracting exe -J junk zipfile prefix (unzipsfx)

-T test zipfile integrity -X eXclude eXtra file attributes

-y store symbolic links as the link instead of the referenced file

-e encrypt -n don't compress these suffixes

-h2 show more help

**Command 2:**

aws

**Expected output:**

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]

To see help text, you can run:

aws help

aws <command> help

aws <command> <subcommand> help

aws: error: the following arguments are required: command

 awscli is required to communicate with S3 and upload the source bundle.

TASK 1.2: CREATE A DEVELOPMENT ENVIRONMENT FOR YOUR SOURCE CODE

You need to make updates on your source code. As a best practice, it is recommended to separate your development environment from your production. In this task, you create a new development environment in your source machine.

1. From the AWS Cloud9 terminal, run the following command to create a new folder called *myapp\_beanstalk* under */tmp* directory:

mkdir /tmp/myapp\_beanstalk

1. Run the following command to copy all your application data into the *myapp\_beanstalk* folder:

cp -r /var/www/html/\* /tmp/myapp\_beanstalk/

 You may have separate systems for your dev/test/prod environments. In this workshop, you simulate it by creating a separate folder in the same machine.

TASK 1.3: UPDATE APPLICATION CONFIGURATION AND CREATE SOURCE CODE BUNDLE

In this task, you need to update application configuration with RDS DB details.

1. From the AWS Cloud9 terminal, run the following command to go into the *myapp\_beanstalk* application folder you’ve just created:

cd /tmp/myapp\_beanstalk

1. Run the following command to open the *wp-config.php* file using vi edit:

sudo vi wp-config.php

 You may use other editors like [nano](https://www.nano-editor.org/dist/v2.1/nano.html) in case you are not comfortable with [vi](https://www.tutorialspoint.com/unix/unix-vi-editor.htm).

Once you open the *wp-config.php* file, you see below parameters already set-up for existing DB settings.

**Expected output:**

/\*\* The name of the database for WordPress \*/

define('DB\_NAME', $\_SERVER['RDS\_DB\_NAME']);

/\*\* Database username \*/

define('DB\_USER', $\_SERVER['RDS\_USERNAME']);

/\*\* Database password \*/

define('DB\_PASSWORD', $\_SERVER['RDS\_PASSWORD']);

/\*\* Database hostname \*/

define('DB\_HOST', $\_SERVER['RDS\_HOSTNAME'] . ':' . $\_SERVER['RDS\_PORT']);

1. Update the above lines in the *wp-config.php* file as below. Notice that, there are no hard-coded details like db-username, password in your configuration file anymore. You configure those parameters when creating the application on Elastic Beanstalk console later. Following this approach is a best practice and allows for application changes without modifying application source code.

define('DB\_NAME', $\_SERVER['RDS\_DB\_NAME']);

define('DB\_USER', $\_SERVER['RDS\_USERNAME']);

define('DB\_PASSWORD', $\_SERVER['RDS\_PASSWORD']);

define('DB\_HOST', $\_SERVER['RDS\_HOSTNAME'] . ':' . $\_SERVER['RDS\_PORT']);

1. Save and quit the *wp-config.php* with :wq.
2. Run the following command to create a source code bundle (wordpress-beanstalk.zip) with zip.

zip ../wordpress-beanstalk.zip -r \* .[^.]\*

1. Run the following command to confirm your *wordpress-beanstalk.zip* file is created under *tmp* folder.

cd /tmp/ && ls

**Expected output:**

myapp\_beanstalk

systemd-private-002f60ad779545a4aaa3fa76c97d5241-systemd-resolved.service-GUWFo9

wordpress-beanstalk.zip

systemd-private-002f60ad779545a4aaa3fa76c97d5241-apache2.service-2gpPrN

systemd-private-002f60ad779545a4aaa3fa76c97d5241-systemd-timesyncd.service-IRFVm4

**Congratulations!** You have successfully prepared your source code bundle. In the next task, you create a S3 bucket and upload the source code bundle into S3.

**Task 2: Upload Source Code**

In this task, you create an S3 Bucket for your Elastic Beanstalk application and then upload your source code to the created S3 Bucket.

TASK 2.1: CREATE S3 BUCKET FOR YOUR SOURCE CODE BUNDLE

1. Keep the Cloud9 browser window as it is, and switch back to the previous browser tab to navigate to the AWS Management Console.
2. At the top of the AWS Management Console, in the search bar, search for and choose

S3

.

1. On the **Amazon S3** console, choose **Create bucket**.
2. On the **Create bucket** page, under **General Configuration** section, locate the **Bucket name** textbox and enter

mysourcecodebucket-eb-lab

.

 S3 Bucket name must be unique across all AWS accounts in all AWS regions. For this lab, you can use any name for the S3 Bucket since the above name is only provided as an example and might not be available when you run this lab.

1. Keep default values in the remaining input fields, scroll-down, and choose **Create bucket**.

 Make note of your S3 bucket name. You use it in the next step.

TASK 2.2: UPLOAD YOUR SOURCE CODE TO S3 BUCKET

1. Navigate back to the **AWS Cloud9** terminal window and run the following command to go into tmp folder where your source code bundle *wordpress-beanstalk.zip* is located.

cd /tmp/

1. Replace *NAME\_OF\_YOUR\_S3\_BUCKET* with the actual bucket name that you created earlier and run the following command to copy the *wordpress-beanstalk.zip* file to that S3 bucket.

aws s3 cp wordpress-beanstalk.zip s3://NAME\_OF\_YOUR\_S3\_BUCKET

E.g.

aws s3 cp wordpress-beanstalk.zip s3://mysourcecodebucket-eb-lab

**Expected output:**

upload: ./wordpress-beanstalk.zip to s3://mysourcecodebucket-eb-lab/wordpress-beanstalk.zip

1. Navigate back to **Amazon S3** console and select the link to the bucket you created earlier (e.g. mysourcecodebucket-eb-lab).
2. Select the link to the **wordpress-beanstalk.zip** file and copy the **Object URL** link into your local notepad. You use this link when setting up application source code location in the next section.

**Congratulations!** You have successfully created your source code bundle and uploaded it into S3 bucket. In the next task, you launch your Elastic Beanstalk environment and deploy your code directly on it.

**Task 3: Launch Environment**

In this task, you create an application and then launch an environment using the Elastic Beanstalk console.

TASK 3.1: CREATE ELASTIC BEANSTALK ENVIRONMENT

1. At the top of the AWS Management Console, in the search bar, search for and choose

Elastic Beanstalk

.

You are prompted with the **Amazon Elastic Beanstalk** getting started page. This page is displayed because the Amazon Elastic Beanstalk service has not yet been configured in this region.

1. On the **Get started** section, choose **Create Application**.
2. On the **Configure environment** page, under **Environment tier**, **Application information** and **Platform** sections, configure the following:

* **Environment tier:** Choose

Web server environment

* **Application name:** Enter

my-eb-app

* **Platform:** Select **PHP** from the drop-down menu.
* **Platform branch:** Select **PHP 8.1 running on 64bit Amazon Linux 2** from the drop-down menu.
* **Platform version:** Select **3.7.x (Recommended)** from the drop-down menu.

1. Under **Application code** section, choose  **Upload your code**.
2. For **Version label**, enter

v1

1. Under **Source code origin** section, choose  **Public S3 URL** and paste the **Object URL** link that was copied during the previous task.
2. For **Presets**, choose  **Custom configuration**.
3. Choose **Next**.
4. On the **Configure service access** window, for **EC2 key pair**, choose the key pair that is available.
5. Choose **Next**.

TASK 3.2: CONFIGURE NETWORK SETTINGS

1. On the **Set up networking, database, and tags** page, for **VPC**, choose the **TargetVPC**.
2. In the **Instance settings** section, for **Instance subnets**, choose:

* **TargeVPC-public-a**
* **TargeVPC-public-b**

1. Scroll to the bottom of the screen, then choose **Next**.
2. On the **Configure instance traffic and scaling** page, choose **Next**.

TASK 3.3: CONFIGURE SOFTWARE SETTINGS

In this task, you configure your software to connect to the RDS database which was already created for this lab.

1. Scroll down to the **Platform software** section and choose **Add environment property**, then add the following environment variables:

* **Name:** Enter

RDS\_DB\_NAME

**Value:** Enter

wordpress-db

.

* **Name:** Enter

RDS\_HOSTNAME

**Value:** Copy and paste the value of **MyRDSInstanceEndpoint** from the left navigation menu of the lab instructions page.

* **Name:** Enter

RDS\_USERNAME

**Value:** Enter

admin

.

* **Name:** Enter

RDS\_PASSWORD

**Value:** Copy and paste the value of **RDSPassword** from the left navigation menu of the lab instructions page.

* **Name:** Enter

RDS\_PORT

**Value:** Enter

3306

.

1. Choose **Next**.
2. Choose **Submit**.

Elastic Beanstalk now automatically starts creating your environment and later deploy and run your application.

 Note: Creating environment may take few minutes. Use them to review other configuration options of Elastic Beanstalk in AWS Console or checking [Elastic Beanstalk documentation](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-containers.html) describing this topic.

Once your environment is ready, you should see its health status as  Ok.

TASK 3.4: UPDATE RDS SECURITY GROUPS

In order to provide connectivity between your application and RDS, you update the RDS security group to allow MySQL traffic from the security group created by Elastic Beanstalk.

1. At the top of the AWS Management Console, in the search bar, search for and choose

RDS

.

1. Choose **Databases** from the left menu and select the link for **targetrdsdatabase** DB.
2. Locate the **Connectivity & security** tab and select link to the **DB-SG** security group in **VPC security groups** section.
3. On the **Security Groups** screen, select the link to the **Target DB SG** from the **Security group ID** column.
4. On the **DB-SG** security group screen, choose **Edit inbound rules**.
5. Choose **Add rule** and configure the following:

* Locate the **Type** drop-down menu and choose **MYSQL/Aurora**.
* Locate the **Source** drop-down menu and choose **Custom**. Using the search menu, select the Security Group that Elastic Beanstalk created during the application creation process. The security group should have a tag **My-eb-app-env**.
* Locate the **Description** textbox and enter

Allow MySQL traffic from the Elastic Beanstalk app instance

.

1. Choose **Save rules**.

**Congratulations!** You have successfully launched your Elastic Beanstalk environment and deployed your application. Go to the next task to review and manage it.

**Task 4: Review and Manage Environment**

Your application is now running on AWS Elastic Beanstalk. In this task, you navigate through Elastic Beanstalk console and review application details.

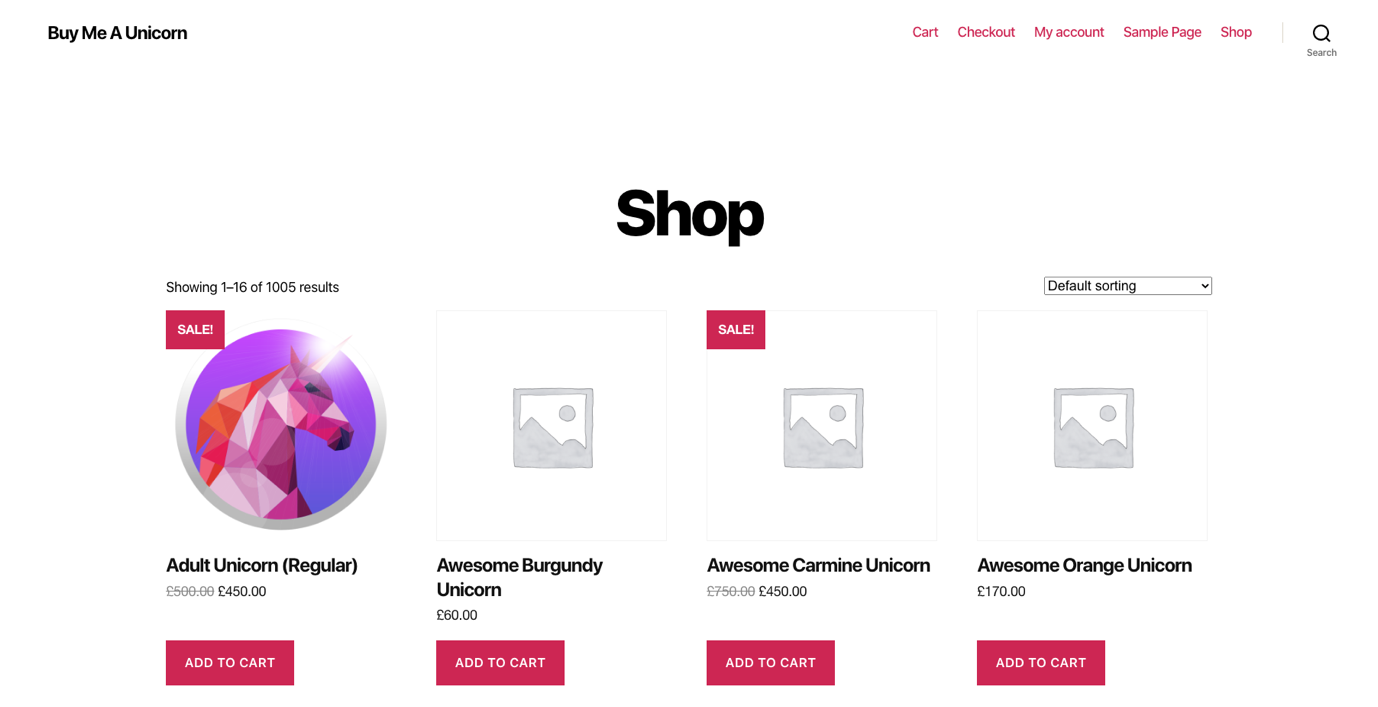
1. At the top of the AWS Management Console, in the search bar, search for and choose

Elastic Beanstalk

.

1. On the left navigation menu choose **Environments**.
2. Locate the **My-eb-app-env** environment and select the URL under **Domain** column to go to your application’s public page. This takes you to your application’s web interface as follows:

**Expected output:**

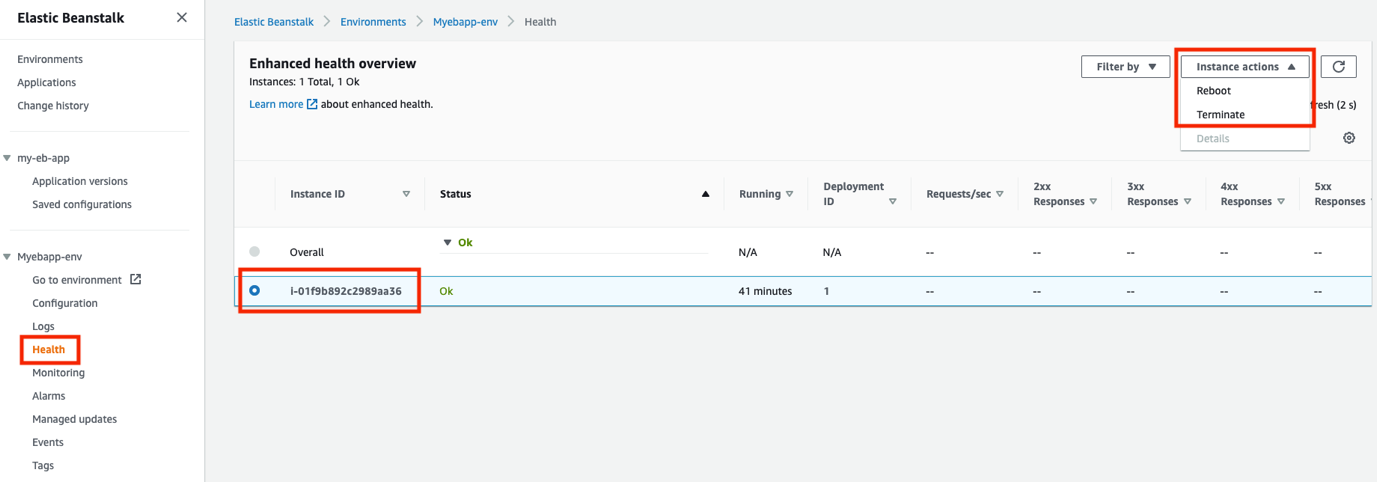


1. Navigate back to the Elastic Beanstalk console and select the link to **My-eb-app-env** environment from the **Environment name** column. You see detailed information about your environment, including:

* Health status
* Recent events
* Platform details
* Public URL
* Currently running version (You can upload and deploy your application code from here.)

Elastic Beanstalk creates and manages all underlying infrastructure for your applications. You can see the instance details from the Elastic Beanstalk console.

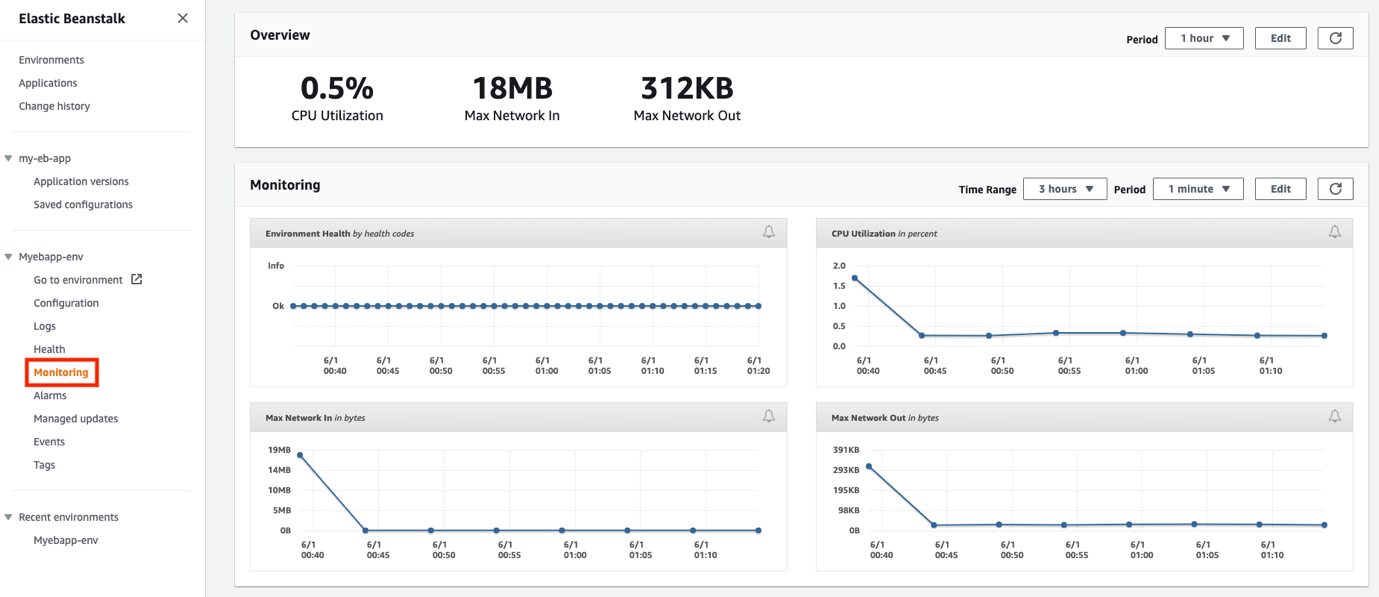
1. From the left navigation menu, expand the **My-eb-app-env** environment and choose **Health**.
2. On the environment health screen, check your Instance ID where your application is running on.
3. You can also reboot/terminate your instance directly by choosing **Instance actions** button from the environment health screen.



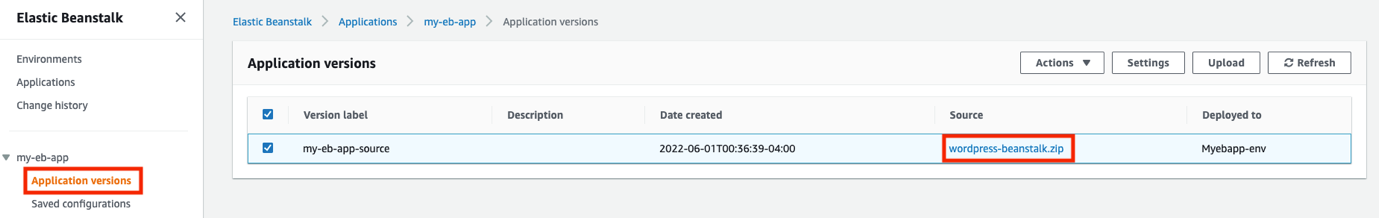
You can also monitor your environment from Elastic Beanstalk console directly.

1. From the left navigation menu, expand the **My-eb-app-env** environment and choose **Monitoring**.

Here you see graphical representation of different metrics for your Elastic Beanstalk environment like Environment Health, CPU Utilization, Max Network In and Max Network Out.



1. From the left navigation menu, expand the **my-eb-app** application and choose **Application versions**.



 Note: You can manage your application versions through Elastic Beanstalk and deploy them into different environments easily without dealing with any underlying infrastructure provisioning.

**Congratulations!** You have successfully reviewed and managed your Elastic Beanstalk environment.

**Task 5: Challenge Task (Optional)**

This is an optional challenge task where you can:

* Make changes in your application source code and deploy the new version into your environment.
* Perform Blue/Green deployments.
* Explore how to deploy your application into multiple availability zones with Elastic Loadbalancer.

**Conclusion**

**Congratulations!** You now have successfully:

* Prepared your application source code for migration to AWS.
* Uploaded your application source code version to an S3 Bucket.
* Created an Elastic Beanstalk application and launched an environment.
* Reviewed and managed the Elastic Beanstalk environment.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

* [Blue/Green deployments with Elastic Beanstalk](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.CNAMESwap.html)
* [Elastic Beanstalk Troubleshooting](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/troubleshooting.html)
* [Creating and deploying PHP applications on Elastic Beanstalk](https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_PHP_eb.html)

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).