

## Project 2 - Solution

---

This solution will walk you through step-to-step guide to deploying a High-Availability PHP Application with an External Amazon RDS Database to Elastic Beanstalk.

This project uses a sample application that uses MySQL database to store the user-provided text data. Link for the application is also provided in the Problem statement, here is link for it: <https://github.com/aws-samples/eb-demo-php-simple-app>

Before proceeding forward and execute its steps, let's check the steps we're going to perform, you can directly jump to any topic for your reference if you want:

**Step1: Launch a DB Instance in Amazon RDS**

**Step2: Create an Elastic Beanstalk Environment**

**Step3: Configure Security Groups, Environment Properties, and Scaling**

**Step4: Deploy the Sample Application**

**Step5: Cleanup**

Now let's get started with its steps.

### Launch a DB Instance in Amazon RDS

To use an external database with an application running in Elastic Beanstalk, first launch a DB instance with Amazon RDS. When you launch an instance with Amazon RDS, it is completely independent of Elastic Beanstalk and your Elastic Beanstalk environments and will not be terminated or monitored by Elastic Beanstalk.

Use the Amazon RDS console to launch a Multi-AZ **MySQL** DB instance. Choosing a Multi-AZ deployment ensures that your database will fail over and continue to be available if the master DB instance goes out of service.

#### To launch an RDS DB instance in a default VPC

1. Open the [RDS console](#).
2. Choose **Instances** in the navigation pane.
3. Choose **Launch DB instance**.
4. Choose a database engine. Choose **Next**.
5. Choose a use case, if prompted.
6. Under **Specify DB details**, review the default settings and adjust as necessary.

Pay attention to the following options:

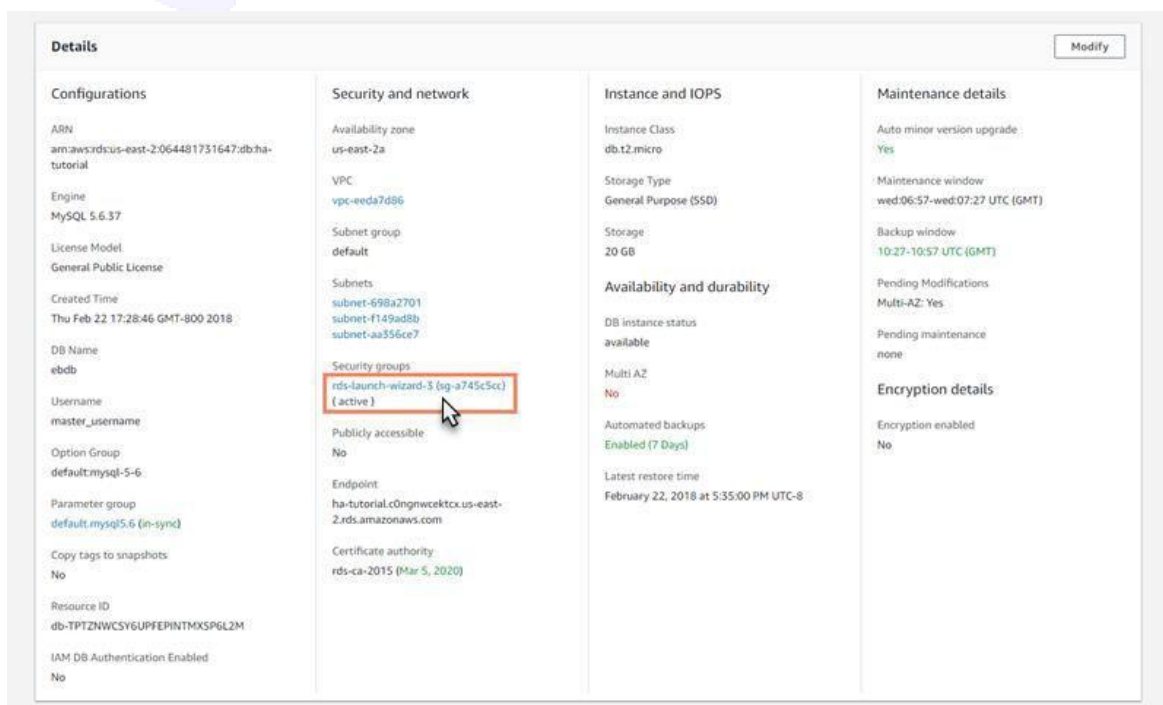
- **DB instance class** – Choose an instance size that has an appropriate amount of memory and CPU power for your workload.

- **Multi-AZ deployment** – For high availability, set to **Create replica in different zone**.
  - **Master username** and **Master password** – The database username and password. Make a note of these settings because you'll use them later.
7. Choose **Next**.
  8. Under **Database options**, for **Database name**, type **ebdb**. Make a note of the **Database port** value for use later.
  9. Verify the default settings for the remaining options and choose **Launch DB instance**.

Next, modify the security group attached to your DB instance to allow inbound traffic on the appropriate port. This is the same security group that you will attach to your Elastic Beanstalk environment later, so the rule that you add will grant ingress permission to other resources in the same security group.

### To modify the ingress rules on your RDS instance's security group

1. Open the Amazon RDS console.
2. Choose **Instances**.
3. Choose the name of your DB instance to view its details.
4. Under **Details** section, note the **Subnets**, **Security groups**, and **Endpoint** shown on this page, so you can use this information later.
5. Under **Security and network**, you can see the security group associated with the DB instance. Open the link to view the security group in the Amazon EC2 console.



The screenshot displays the 'Details' page for an Amazon RDS instance. The page is divided into four main sections: Configurations, Security and network, Instance and IOPS, and Maintenance details. In the 'Security and network' section, the 'Security groups' field is highlighted with a red box, showing 'rds-launch-wizard-3 (sg-a745c5cc) (active)'. A mouse cursor is pointing at this link. Other visible details include the instance class 'db.t2.micro', storage type 'General Purpose (SSD)', and the endpoint 'ha-tutorial.c0ngnwcktcx.us-east-2.rds.amazonaws.com'.

6. In the security group details, choose **Inbound**.
7. Choose **Edit**.
8. Choose **Add Rule**.
9. For **Type**, choose the DB engine that your application uses.
10. For **Source**, type **sg-** to view a list of available security groups. Choose the current security group to allow resources in the security group to receive traffic on the database port from other resources in the same group.



11. Choose **Save**.

Creating a DB instance takes about 10 minutes.

**In the meantime, create your Elastic Beanstalk Environment.**

## Create an Elastic Beanstalk Environment

Use the AWS Management Console to create an Elastic Beanstalk environment. Choose the **PHP** platform and accept the default settings and sample code. After you launch the environment, you can configure the environment to connect to the database, then deploy the sample application that you downloaded from GitHub.

### To launch an environment (console)

1. Open the Elastic Beanstalk console.
2. For **Platform**, choose the platform that matches the language used by your application.
3. For **Application code**, choose **Sample application**.
4. Choose **Review and launch**.
5. Review the available options. When you're satisfied with them, choose **Create app**.

Environment creation takes about 5 minutes and creates the following resources:

- **EC2 instance** – An Amazon Elastic Compute Cloud (Amazon EC2) virtual machine configured to run web apps on the platform that you choose. Each platform runs a specific 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 sits 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 ingress on port 80. This resource lets HTTP traffic from the load balancer reach the EC2 instance running your web app. By default, traffic isn't allowed on other ports.
- **Load balancer** – An Elastic Load Balancing load balancer configured to distribute requests to the instances running your application. A load balancer also eliminates the need to expose your instances directly to the internet.

- **Load balancer security group** – An Amazon EC2 security group configured to allow ingress on port 80. This resource lets HTTP traffic from the internet reach the load balancer. By default, traffic isn't allowed on other ports.
- **Auto Scaling group** – An Auto Scaling group configured to replace an instance if it is terminated or becomes unavailable.
- **Amazon S3 bucket** – A storage location for your source code, logs, and other artifacts that are created when you use Elastic Beanstalk.
- **Amazon CloudWatch alarms** – Two CloudWatch 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 CloudFormation stack** – Elastic Beanstalk uses AWS CloudFormation 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 CloudFormation console.
- **Domain name** – A domain name that routes to your web app in the form *subdomain.region.elasticbeanstalk.com*.

All these resources are managed by Elastic Beanstalk. When you terminate your environment, Elastic Beanstalk terminates all the resources that it contains. The RDS DB instance that you launched is outside of your environment, so you are responsible for managing its lifecycle.

#### Note

The Amazon S3 bucket that Elastic Beanstalk creates is shared between environments and is not deleted during environment termination.

## Configure Security Groups, Environment Properties, and Scaling

Add the security group of your DB instance to your running environment. This procedure causes Elastic Beanstalk to re-provision all instances in your environment with the additional security group attached.

### To add a security group to your environment

- Do one of the following:
  - To add a security group using the Elastic Beanstalk console
    1. Open the Elastic Beanstalk console.
    2. Navigate to the management page for your environment.

3. Choose **Configuration**.
  4. On the **Instances** configuration card, choose **Modify**.
  5. Under **EC2 security groups**, choose the security group to attach to the instances, in addition to the instance security group that Elastic Beanstalk creates.
  6. Choose **Apply**.
  7. Read the warning, and then choose **Confirm**.
- To add a security group using a configuration file, you can use this example file **securitygroup-addexisting.config**: <https://github.com/aws-samples/aws-sns-samples/blob/master/templates/SNS-VPCE-Tutorial-CloudFormation.template>

Next, use environment properties to pass the connection information to your environment. The sample application uses a default set of properties that match the ones that Elastic Beanstalk configures when you provision a database within your environment.

### To configure environment properties for an Amazon RDS DB instance

1. Open the Elastic Beanstalk console.
2. Navigate to the management page for your environment.
3. Choose **Configuration**.
4. On the **Software** configuration card, choose **Modify**.
5. In the **Environment properties** section, define the variables that your application reads to construct a connection string. For compatibility with environments that have an integrated RDS DB instance, use the following.
  - **RDS\_HOSTNAME** – The hostname of the DB instance.  
Amazon RDS console label – **Endpoint** (this is the hostname)
  - **RDS\_PORT** – The port on which the DB instance accepts connections. The default value varies among DB engines.  
Amazon RDS console label – **Port**
  - **RDS\_DB\_NAME** – The database name, ebdb.  
Amazon RDS console label – **DB Name**
  - **RDS\_USERNAME** – The user name that you configured for your database.  
Amazon RDS console label – **Username**
  - **RDS\_PASSWORD** – The password that you configured for your database.

### Environment Properties

The following properties are passed into the application as environment variables. [Learn more.](#)

Property Name	Property Value
RDS_DB_NAME	<input type="text" value="ebdb"/> ✕
RDS_HOSTNAME	<input type="text" value="webapp-db.ixzcb5mpaniu.us-wes"/> ✕
RDS_PORT	<input type="text" value="5432"/> ✕
RDS_USERNAME	<input type="text" value="webapp-admin"/> ✕
<input type="text" value="RDS_PASSWORD"/>	<input type="text" value="kUj5uKxmWDMYc403"/> +

[Cancel](#) [Apply](#)

6. Choose **Apply**.

Finally, configure your environment's Auto Scaling group with a higher minimum instance count. Always run at least two instances to prevent the web servers in your environment from being a single point of failure, and to allow you to deploy changes without taking your site out of service.

#### To configure your environment's Auto Scaling group for high availability

1. Open the Elastic Beanstalk console.
2. Navigate to the management page for your environment.
3. Choose **Configuration**.
4. On the **Capacity** configuration card, choose **Modify**.
5. In the **Auto Scaling Group** section, set **Min instances** to 2.
6. Choose **Apply**.

### Deploy the Sample Application

Now your environment is ready to run the sample application and connect to Amazon RDS. Deploy the sample application to your environment.

**Note:** Download the source bundle from GitHub, if you haven't already: [Click Here](#)



## To deploy a source bundle

1. Open the Elastic Beanstalk console.
2. Navigate to the management page for your environment.
3. Choose **Upload and Deploy**.
4. Choose **Choose File** and use the dialog box to select the source bundle.
5. Choose **Deploy**.
6. When the deployment completes, choose the site URL to open your website in a new tab.

The site collects user comments and uses a MySQL database to store the data. To add a comment, choose **Share Your Thought**, enter a comment, and then choose **Submit Your Thought**. The web app writes the comment to the database so that any instance in the environment can read it, and it won't be lost if instances go out of service.



## Cleanup

When you finish working with Elastic Beanstalk, you can terminate your environment. Elastic Beanstalk terminates all AWS resources associated with your environment, such as Amazon EC2 instances, database instances, load balancers, security groups, and alarms.

## To terminate your Elastic Beanstalk environment

1. Open the Elastic Beanstalk console.
2. Navigate to the management page for your environment.
3. Choose **Actions**, and then choose **Terminate Environment**.
4. In the **Confirm Termination** dialog box, type the environment name, and then choose **Terminate**.



With Elastic Beanstalk, you can easily create a new environment for your application at any time.

In addition, you can terminate database resources that you created outside of your Elastic Beanstalk environment. When you terminate an Amazon RDS database instance, you can take a snapshot and restore the data to another instance later.

### To terminate your RDS DB instance

1. Open the Amazon RDS console.
  2. Choose **Instances**.
  3. Choose your DB instance.
  4. Choose **Instance actions**, and then choose **Delete**.
  5. Choose whether to create a snapshot, and then choose **Delete**.
- 

