Deploying a high-availability WordPress website with an external Amazon RDS database to Elastic Beanstalk

This tutorial describes how to launch an Amazon RDS DB instance that is external to AWS Elastic Beanstalk, then how to configure a high-availability environment running a WordPress website to connect to it.

Running a DB instance external to Elastic Beanstalk decouples the database from the lifecycle of your environment. This lets you connect to the same database from multiple environments, swap out one database for another, or perform a blue/green deployment without affecting your database.

This tutorial was developed with WordPress version 6.0 and PHP 8.0.

Note

For current information about the compatibility of PHP releases with WordPress versions, see PHP Compatibility and WordPress Versions on the WordPress website. You should refer to this information before you upgrade to a new release of PHP for your WordPress implementations.

Topics

Prerequisites

This tutorial assumes you have knowledge of the basic Elastic Beanstalk operations and the Elastic Beanstalk console. If you haven't already, follow the instructions in Getting started using Elastic Beanstalk to launch your first Elastic Beanstalk environment.

To follow the procedures in this guide, you will need a command line terminal or shell to run commands. Commands are shown in listings preceded by a prompt symbol ($) and the name of the current directory, when appropriate.

```
~/eb-project$ this is a command

this is output
```

On Linux and macOS, you can use your preferred shell and package manager. On Windows 10, you can install the Windows Subsystem for Linux to get a Windows-integrated version of Ubuntu and Bash.

Default VPC

The Amazon Relational Database Service (Amazon RDS) procedures in this tutorial assume that you are launching resources in a default Amazon Virtual Private Cloud (Amazon VPC). All new accounts include a default VPC in each AWS

Region. If you don't have a default VPC, the procedures will vary. See Using Elastic Beanstalk with Amazon RDS for instructions for EC2-Classic and custom VPC platforms.

Launch a DB instance in Amazon RDS

When you launch an instance with Amazon RDS, it's completely independent of Elastic Beanstalk and your Elastic Beanstalk environments, and will not be terminated or monitored by Elastic Beanstalk.

In the following steps you'll use the Amazon RDS console to:

- Launch a database with the **MySQL** engine.
- Enable a **Multi-AZ deployment**. This creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

To launch an RDS DB instance in a default VPC

1. Open the RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose **Create database**.
4. Choose **Standard Create**.

   Important

   Do not choose **Easy Create**. If you choose it, you can't configure the necessary settings to launch this RDS DB.
5. Under **Additional configuration**, for **Initial database name**, type `ebdb`.

6. Review the default settings and adjust these settings according to your specific requirements. 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 this to **Multi-AZ DB instance.**
    - **Master username** and **Master password** – The database username and password. Make a note of these settings because you will use them later.
7. Verify the default settings for the remaining options, and then choose **Create database**.

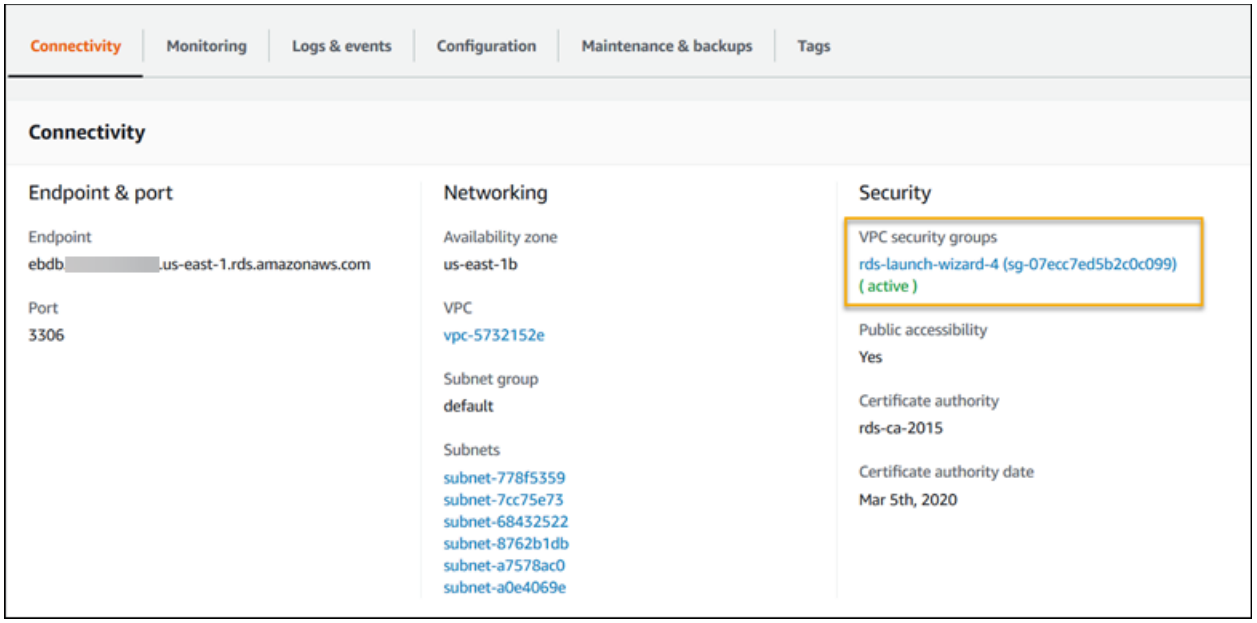
After your DB instance is created, modify the security group attached to it in order to allow inbound traffic on the appropriate port.

Note

This is the same security group that you'll attach to your Elastic Beanstalk environment later, so the rule that you add now will grant ingress permission to other resources in the same security group.

To modify the inbound rules on the security group that's attached to your RDS instance

1. Open the Amazon RDS console.
2. Choose **Databases**.
3. Choose the name of your DB instance to view its details.
4. In the **Connectivity** section, make a note of the **Subnets**, **Security groups**, and **Endpoint** that are displayed on this page. This is so you can use this information later.
5. Under **Security**, you can see the security group that's associated with the DB instance. Open the link to view the security group in the Amazon EC2 console.



Connectivity

Endpoint & port

Endpoint
ebdb[redacted].us-east-1.rds.amazonaws.com

Port
3306

Networking

Availability zone
us-east-1b

VPC
vpc-5732152e

Subnet group
default

Subnets
subnet-778f5359
subnet-7cc75e73
subnet-68432522
subnet-8762b1db
subnet-a7578ac0
subnet-a0e4069e

Security

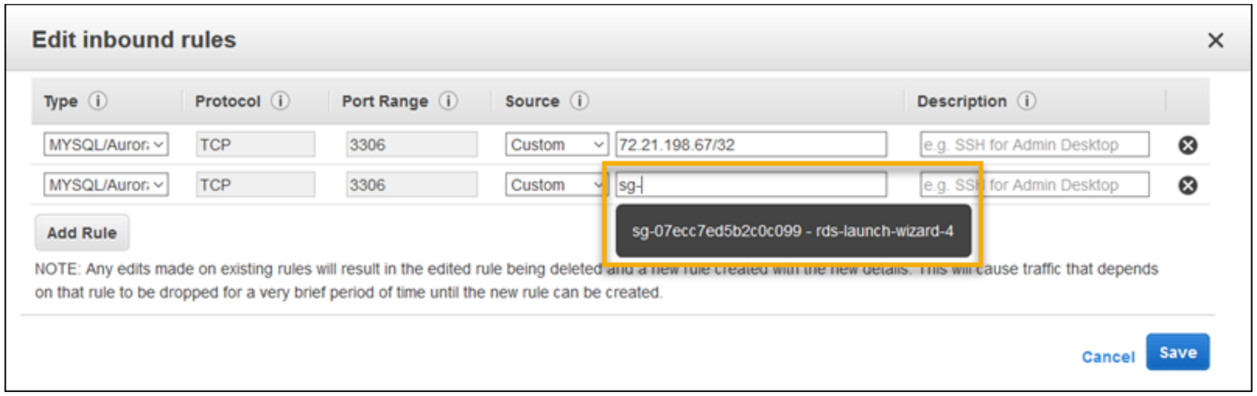
VPC security groups
rds-launch-wizard-4 (sg-07ecc7ed5b2c0c099)
(active)

Public accessibility
Yes

Certificate authority
rds-ca-2015

Certificate authority date
Mar 5th, 2020

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 security group that's associated with the Auto Scaling group that's used with your Elastic Beanstalk environment. This is so that Amazon EC2 instances in the environment can have access to the database.


Edit inbound rules

Type	Protocol	Port Range	Source		Description
MYSQL/Auror:	TCP	3306	Custom	72.21.198.67/32	e.g. SSH for Admin Desktop
MYSQL/Auror:	TCP	3306	Custom	sg-	e.g. SSH for Admin Desktop

sg-07ecc7ed5b2c0c099 - rds-launch-wizard-4

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

11. Choose **Save**.

Creating a DB instance takes about 10 minutes. In the meantime, download WordPress and create your Elastic Beanstalk environment.

Download WordPress

To prepare to deploy WordPress using AWS Elastic Beanstalk, you must copy the WordPress files to your computer and provide the correct configuration information.

To create a WordPress project

1. Download WordPress from wordpress.org.

```
~$ curl https://wordpress.org/wordpress-6.0.tar.gz -o wordpress.tar.gz
```

2. Download the configuration files from the sample repository.

```
~$ wget https://github.com/KavitaMahajan123/EBSWPworkshop/raw/main/eb-php-wordpress-v1.zip
```

3. Extract WordPress and change the name of the folder.

```
4.  ~$ tar -xvf wordpress.tar.gz

5.  ~$ mv wordpress wordpress-beanstalk
```

```
 ~$ cd wordpress-beanstalk
```

6. Extract the configuration files over the WordPress installation.

```
7.  ~/wordpress-beanstalk$ unzip ../eb-php-wordpress-v1.zip

8.   creating: .ebextensions/

9.  inflating: .ebextensions/dev.config

10.  inflating: .ebextensions/loadbalancer-sg.config

11.  inflating: .ebextensions/wordpress.config

12.  inflating: LICENSE

13.  inflating: README.md

14. inflating: wp-config.php
```

Launch an Elastic Beanstalk environment

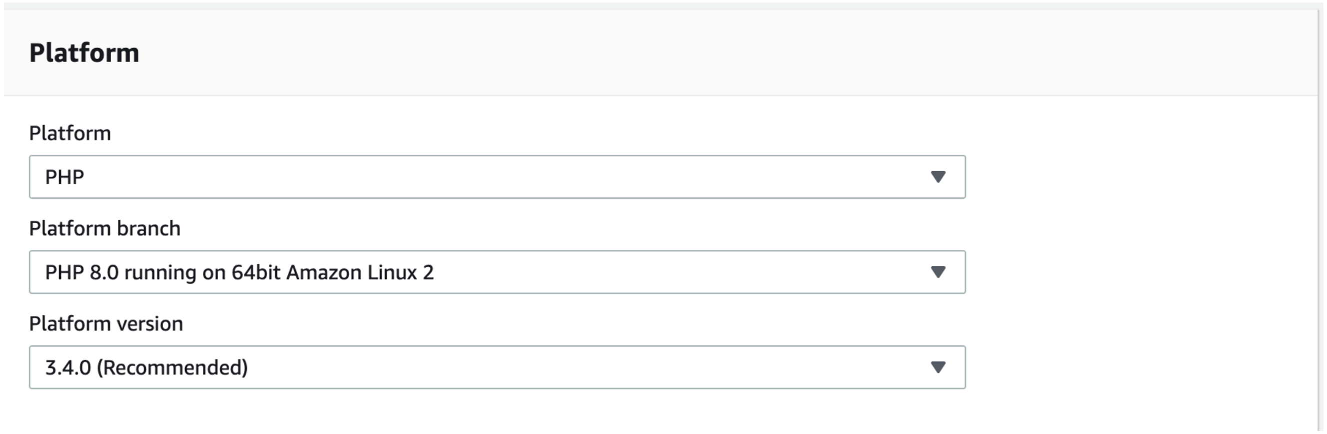
Use the Elastic Beanstalk console to create an Elastic Beanstalk environment. After you launch the environment, you can configure it to connect to the database, then deploy the WordPress code to the environment.

In the following steps, you'll use the Elastic Beanstalk console to:

- Create an Elastic Beanstalk application using the managed **PHP** platform.
- Accept the default settings and sample code.

To launch an environment (console)

1. Open the Elastic Beanstalk console using this preconfigured link: console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
2. For **Platform**, select PHP v8.0.



Platform

Platform: PHP

Platform branch: PHP 8.0 running on 64bit Amazon Linux 2

Platform version: 3.4.0 (Recommended)

3. For **Application code**, choose **Sample application**.
4. Choose **Review and launch**.
5. Review the available options. Choose the available option you want to use, and when you're ready, choose **Create app**.

All of these resources are managed by Elastic Beanstalk. When you terminate your environment, Elastic Beanstalk terminates all the resources that it contains.

Because the Amazon RDS instance that you launched is outside of your environment, 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. For more information, see Using Elastic Beanstalk with Amazon S3.

Configure security groups and environment properties

Add the security group of your DB instance to your running environment. This procedure causes Elastic Beanstalk to reprovision 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, 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 **Instances** configuration category, choose **Edit**.
        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, use the `securitygroup-addexisting.config` example file.

Next, use environment properties to pass the connection information to your environment.

The WordPress 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

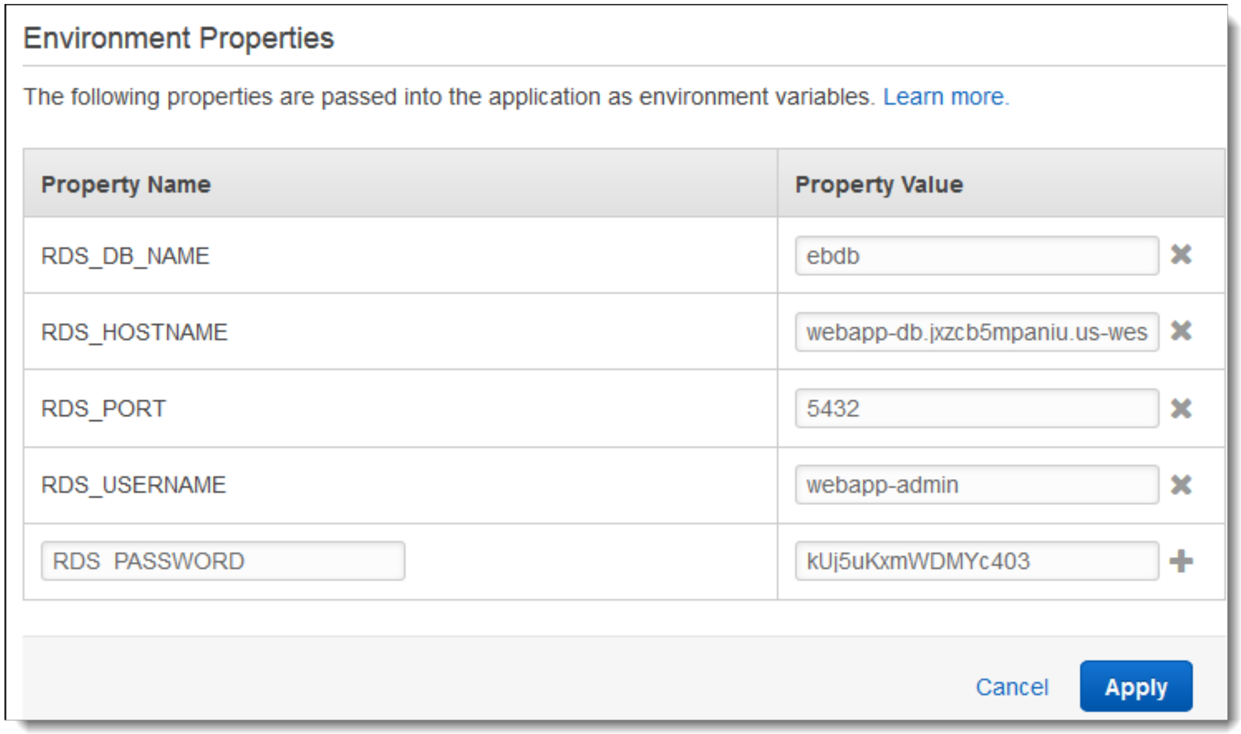
1. Open the Elastic Beanstalk console, and in the **Regions** list, select your AWS Region.
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.

   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 **Software** configuration category, choose **Edit**.
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 names and values. You can find all values, except for your password, in the RDS console.

Property name	**Description**	**Property value**
`RDS_HOSTNAME`	The hostname of the DB instance.	On the **Connectivity & security** tab on the Amazon RDS console: **Endpoint**.
`RDS_PORT`	The port where the DB instance accepts connections. The default value varies among DB engines.	On the **Connectivity & security** tab on the Amazon RDS console: **Port**.
`RDS_DB_NAME`	The database name, `ebdb`.	On the **Configuration** tab on the Amazon RDS console: **DB Name**.
`RDS_USERNAME`	The username that you configured for your database.	On the **Configuration** tab on the Amazon RDS console: **Master username**.
`RDS_PASSWORD`	The password that you configured for your database.	Not available for reference in the Amazon RDS console.



Environment Properties

The following properties are passed into the application as environment variables. Learn more.

Property Name	Property Value
RDS_DB_NAME	ebdb
RDS_HOSTNAME	webapp-db.jxzcb5mpaniu.us-wes
RDS_PORT	5432
RDS_USERNAME	webapp-admin
RDS_PASSWORD	kUj5uKxmWDMYc403

6. Choose **Apply**.

Configure and deploy your application

Verify that the structure of your `wordpress-beanstalk` folder is correct, as shown.

```
wordpress-beanstalk$ tree -aL 1
```

```
.
├── .ebextensions
├── index.php
├── LICENSE
├── license.txt
├── readme.html
├── README.md
├── wp-activate.php
├── wp-admin
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config.php
├── wp-config-sample.php
├── wp-content
├── wp-cron.php
├── wp-includes
├── wp-links-opml.php

├── wp-load.php

├── wp-login.php

├── wp-mail.php

├── wp-settings.php

├── wp-signup.php

├── wp-trackback.php

└── xmlrpc.php
```

The customized `wp-config.php` file from the project repo uses the environment variables that you defined in the previous step to configure the database connection. The `.ebextensions` folder contains configuration files that create additional resources within your Elastic Beanstalk environment.

The configuration files require modification to work with your account. Replace the placeholder values in the files with the appropriate IDs and create a source bundle.

To update configuration files and create a source bundle

1. Modify the configuration files as follows.
   - `.ebextensions/dev.config` – Restricts access to your environment to protect it during the WordPress installation process. Replace the placeholder IP address near the top of the file with the public IP address of the computer you'll use to access your environment's website to complete your WordPress installation.

    Note

    Depending on your network, you might need to use an IP address block.

- `.ebextensions/dev.config` – Identify your default VPC in the Amazon VPC console, and replace **vpc-XXXXXXXXX** with the VPC ID.

2. Create a source bundle containing the files in your project folder. The following command creates a source bundle named `wordpress-beanstalk.zip`.

   ```
   ~/eb-wordpress$ zip ../wordpress-beanstalk.zip -r * .[^.]*
   ```

Upload the source bundle to Elastic Beanstalk to deploy WordPress to your environment.

To deploy a source bundle

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. On the environment overview page, choose **Upload and deploy**.
4. Use the on-screen dialog box to upload the source bundle.
5. Choose **Deploy**.
6. When the deployment completes, you can choose the site URL to open your website in a new tab.

Install WordPress

To complete your WordPress installation

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. Choose the environment URL to open your site in a browser. You are redirected to a WordPress installation wizard because you haven't configured the site yet.
4. Perform a standard installation. The `wp-config.php` file is already present in the source code and configured to read the database connection information from the environment. You shouldn't be prompted to configure the connection.

Installation takes about a minute to complete.

Update keys and salts

The WordPress configuration file `wp-config.php` also reads values for keys and salts from environment properties. Currently, these properties are all set to `test` by the `wordpress.config` file in the `.ebextensions` folder.

The hash salt can be any value that meets the environment property requirements, but you should not store it in source control. Use the Elastic Beanstalk console to set these properties directly on the environment.

To update environment properties

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. On the navigation pane, choose **Configuration**.
4. Under **Software**, choose **Edit**.
5. For `Environment properties`, modify the following properties:
   - `AUTH_KEY` – The value chosen for `AUTH_KEY`.
   - `SECURE_AUTH_KEY` – The value chosen for `SECURE_AUTH_KEY`.
   - `LOGGED_IN_KEY` – The value chosen for `LOGGED_IN_KEY`.
   - `NONCE_KEY` – The value chosen for `NONCE_KEY`.
   - `AUTH_SALT` – The value chosen for `AUTH_SALT`.
   - `SECURE_AUTH_SALT` – The value chosen for `SECURE_AUTH_SALT`.
   - `LOGGED_IN_SALT` – The value chosen for `LOGGED_IN_SALT`.
   - `NONCE_SALT` — The value chosen for `NONCE_SALT`.
6. Choose **Apply**.

Note

Setting the properties on the environment directly overrides the values in `wordpress.config`.

Remove access restrictions

The sample project includes the configuration file `loadbalancer-sg.config`. It creates a security group and assigns it to the environment's load balancer, using the IP address that you configured in `dev.config`. It restricts HTTP access on port 80 to connections from your network. Otherwise, an outside party could potentially connect to your site before you have installed WordPress and configured your admin account.

Now that you've installed WordPress, remove the configuration file to open the site to the world.

To remove the restriction and update your environment

1. Delete the `.ebextensions/loadbalancer-sg.config` file from your project directory.

   ```
   ~/wordpress-beanstalk$ rm .ebextensions/loadbalancer-sg.config
   ```

2. Create a source bundle.

   ```
   ~/eb-wordpress$ zip ../wordpress-beanstalk-v2.zip -r * .[^.]*
   ```

Upload the source bundle to Elastic Beanstalk to deploy WordPress to your environment.

To deploy a source bundle

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. On the environment overview page, choose **Upload and deploy**.
4. Use the on-screen dialog box to upload the source bundle.
5. Choose **Deploy**.
6. When the deployment completes, you can choose the site URL to open your website in a new tab.

Configure your Auto Scaling group

Finally, configure your environment's Auto Scaling group with a higher minimum instance count. Run at least two instances at all times to prevent the web servers in your environment from being a single point of failure. This also allows 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, 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, set **Min instances** to 2.
6. Choose **Apply**.

Upgrade WordPress

To upgrade to a new version of WordPress, back up your site and deploy it to a new environment.

Important

Do not use the update functionality within WordPress or update your source files to use a new version. Both of these actions can result in your post URLs returning 404 errors even though they are still in the database and file system.

To upgrade WordPress

1. In the WordPress admin console, use the export tool to export your posts to an XML file.
2. Deploy and install the new version of WordPress to Elastic Beanstalk with the same steps that you used to install the previous version. To avoid downtime, you can create an environment with the new version.
3. On the new version, install the WordPress Importer tool in the admin console and use it to import the XML file containing your posts. If the posts were created by the admin user on the old version, assign them to the admin user on the new site instead of trying to import the admin user.
4. If you deployed the new version to a separate environment, do a CNAME swap to redirect users from the old site to the new site.

Clean up

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, 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. Choose **Environment actions**, and then choose **Terminate environment**.
4. Use the on-screen dialog box to confirm environment termination.

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 DB 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 **Databases**.
3. Choose your DB instance.
4. Choose **Actions**, and then choose **Delete**.
5. Choose whether to create a snapshot, and then choose **Delete**.

Next steps

As you continue to develop your application, you'll probably want a way to manage environments and deploy your application without manually creating a .zip file and uploading it to the Elastic Beanstalk console. The Elastic Beanstalk Command Line Interface (EB CLI) provides easy-to-use commands for creating, configuring, and deploying applications to Elastic Beanstalk environments from the command line.

The sample application uses configuration files to configure PHP settings and create a table in the database, if it doesn't already exist. You can also use a configuration file to configure the security group settings of your instances during environment creation to avoid time-consuming configuration updates. See Advanced environment customization with configuration files (.ebextensions) for more information.

For development and testing, you might want to use the Elastic Beanstalk functionality for adding a managed DB instance directly to your environment. For instructions on setting up a database inside your environment, see Adding a database to your Elastic Beanstalk environment.

If you need a high-performance database, consider using Amazon Aurora. Amazon Aurora is a MySQL-compatible database engine that offers commercial database features at low cost. To connect your application to a different database, repeat the security group configuration steps and update the RDS-related environment properties.

Finally, if you plan on using your application in a production environment, you will want to configure a custom domain name for your environment and enable HTTPS for secure connections.