Assignment 2: Installing WordPress in AWS (PaaS)

Objectives of Assignment 2

In this assignment, you will install a WordPress blog in AWS using *Elastic Beanstalk* and *RDS*.

You will construct this assignment in the *AWS Academy Learner Lab* classroom that has been provided.

The purpose of this assignment is to demonstrate your knowledge of Amazon Web Services gathered from lectures and assigned labs and to show you how you can leverage Platform as a Service (PaaS) to build a web application easily. Some services you configure in this assignment will be familiar; some may be new. It is broken down into 6 tasks:

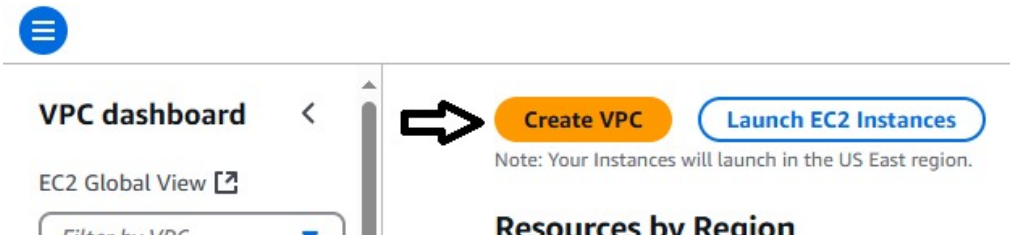
- Task 1: Networking
- Task 2: Database
- Task 3: Wordpress Source Code Modification
- Task 4: Elastic Beanstalk
- Task 5: Site Configuration & Blog Posts

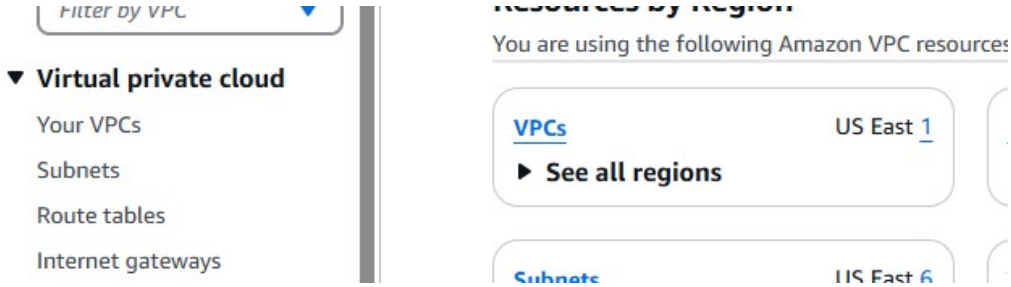
Task 1: Networking

In this task, you will create all the networking required for your new web application.

Virtual Private Cloud

Start your session in the Learner Lab by clicking on the **Start Lab** button. Once the red dot has turned green, click on it to enter the Learner Lab and access the AWS Console interface. You are going to create a new Virtual Private Cloud (VPC). Resources you created last week in Lab 5 will be inaccessible in this VPC. Navigate to VPC (which you may have added to your favourites last week). On the VPC dashboard, click **Create VPC**. See the following screenshot for reference.

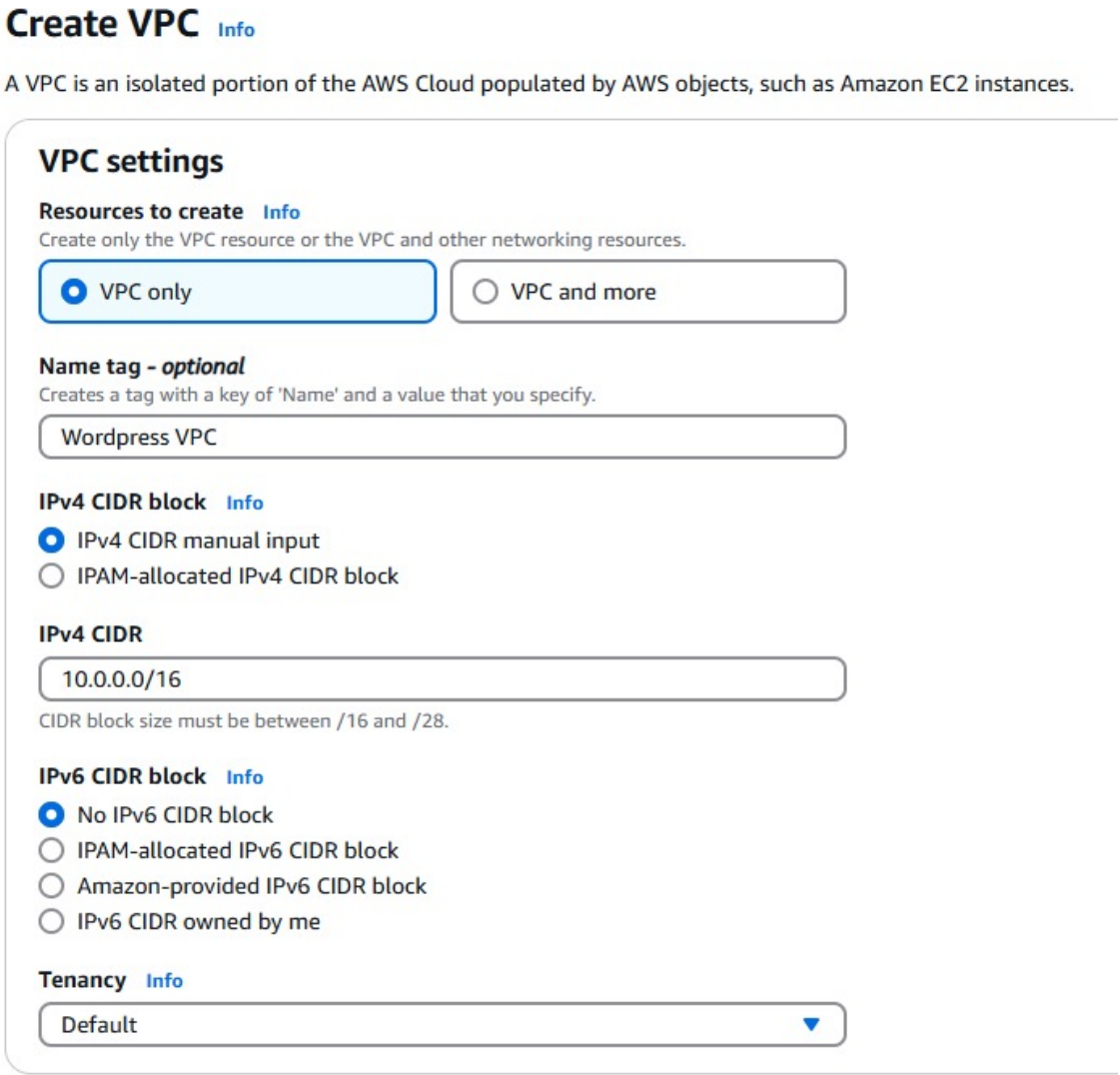




Input the following settings:

1. Select **VPC only**
2. Name tag: **Wordpress VPC**
3. IPv4 CIDR: **10.0.0.0/16**
4. Leave all other settings on default

Confirm the settings match the following screenshot.



Click **Create VPC** at the bottom right.

Once created, click the **Actions** drop down in the top right corner and select **Edit VPC Settings**. Make sure the following boxes are **Checked**:

1. Enable DNS resolution: **Checked**
2. Enable DNS hostnames: **Checked**

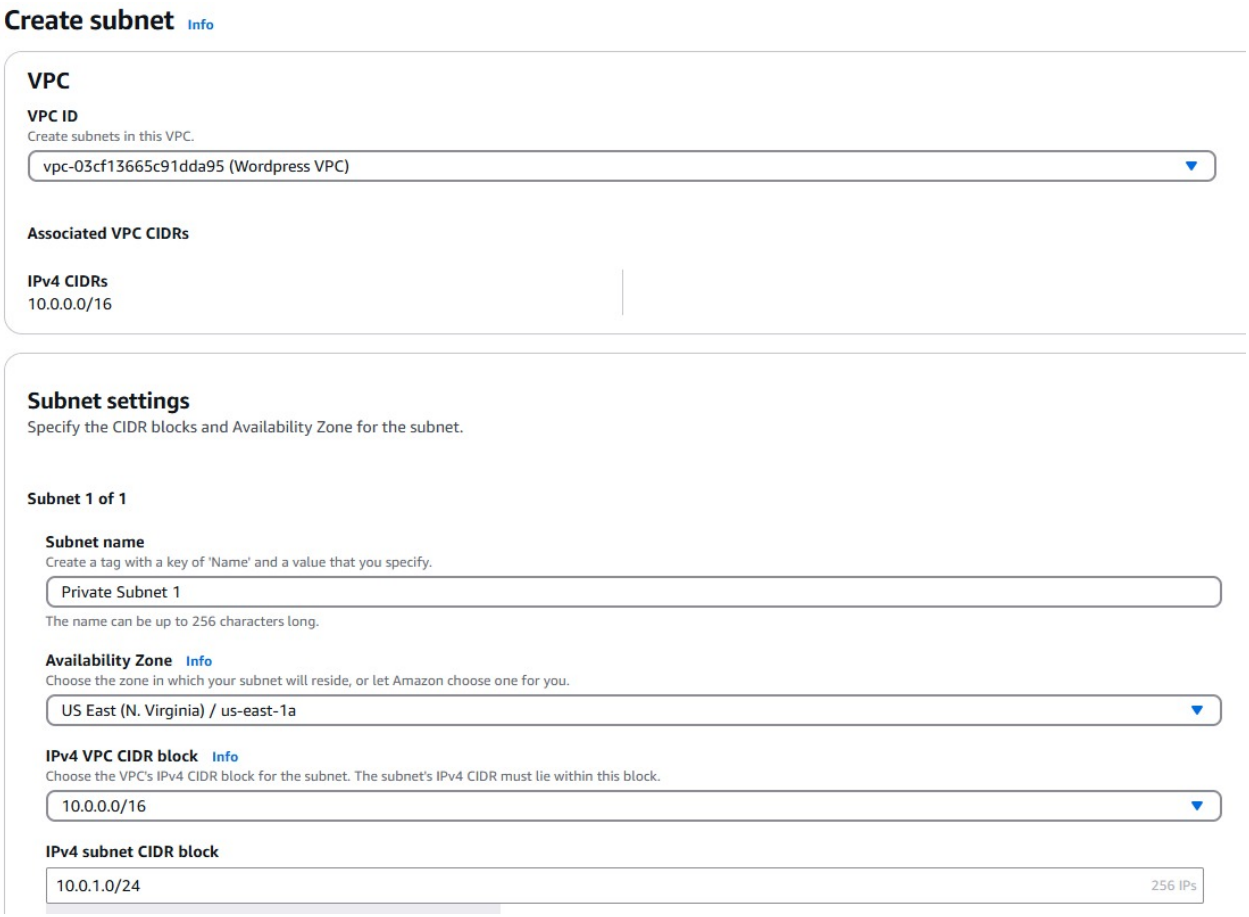
Click **Save**

Subnets

You are going to create 4 subnets in your VPC. Two private subnets, and two public subnets. One of each type (private and public) will be in a different availability zone. This will provide redundancy, ensuring better uptime for servers and applications you create in your VPC.

1. Click on **Subnets** (located on the left side under **Virtual private cloud**).
2. Click **Create subnet** (top right corner).
3. Select **Wordpress VPC** from the VPC ID dropdown.
4. Create a subnet with the following information:

- Subnet Name: **Private subnet 1**
- Availability Zone: **us-east-1a**
- IPv4 VPC CIDR block: **10.0.0/16**
- IPv4 subnet CIDR block: **10.0.1.0/24**
- Your screen should look as follows:



1. Click **Add new subnet** and repeat the process for the following **three** subnets:

Create one private IPv4 subnets in this VPC:

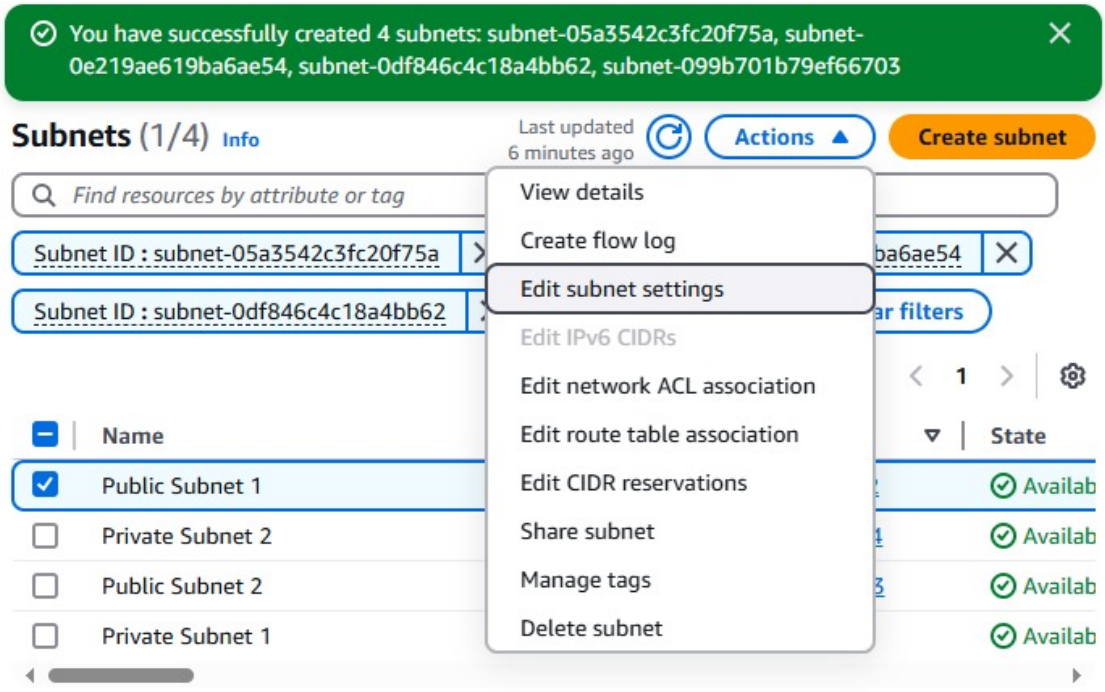
1. **Private Subnet 2** - **10.0.2.0/24** - **us-east-1b**

Create two public IPv4 subnets in this VPC:

1. **Public Subnet 1** - **10.0.11.0/24** - **us-east-1a**
2. **Public Subnet 2** - **10.0.12.0/24** - **us-east-1b**

Once you have confirmed your settings are correct, scroll down and click **Create Subnet** in the bottom right.

Check the box beside **Public Subnet 1**. Click on the **Actions** dropdown (top right) and select **Edit subnet settings**. See the following screenshot.



Make sure the following are **Checked**:

1. Enable auto-assign public IPv4 address: **Checked**
2. Enable resource name DNS A record on launch: **Checked**
3. Click **Save**.
4. Repeat the process for **Public Subnet 2**

Adding an Internet Gateway

Your VPC requires a **Gateway** to access outside resources. There are four types of **gateways**

Internet Gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet. It supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.

An internet gateway enables resources in your public subnets (such as EC2 instances) to connect to the internet if the resource has a public IPv4 address or an IPv6 address. Similarly, resources on the internet can initiate a connection to resources in your subnet using the public IPv4 address or IPv6 address. For example, an internet gateway enables you to connect to an EC2 instance in AWS using your local computer.

Egress-only Internet Gateway

An egress-only internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows outbound communication over IPv6 from instances in your VPC to the internet, and prevents the internet from initiating an IPv6 connection with your instances.

An egress-only internet gateway is for use with IPv6 traffic only. To enable outbound-only internet communication over IPv4, use a NAT gateway instead.

Carrier Gateway

A carrier gateway is a VPC component that allows connectivity between AWS and your on-premises network using AWS Direct Connect or AWS Site-to-Site VPN. It is specifically designed for use with AWS Outposts, enabling communication between your Outposts and the internet, or between your Outposts and other AWS services. The carrier gateway supports both IPv4 and IPv6 traffic and provides a highly available and redundant connection.

A carrier gateway is used when you need to connect your Outposts to the internet or to other AWS services, ensuring that your on-premises applications can communicate seamlessly with AWS resources.

NAT Gateway

A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances.

When you create a NAT gateway, you specify one of the following connectivity types:

- Public – (Default) Instances in private subnets can connect to the internet through a public NAT gateway, but cannot receive unsolicited inbound connections from the internet. You create a public NAT gateway in a public subnet and must associate an elastic IP address with the NAT gateway at creation. You route traffic from the NAT gateway to the internet gateway for the VPC. Alternatively, you can use a public NAT gateway to connect to other VPCs or your on-premises network. In this case, you route traffic from the NAT gateway through a transit gateway or a virtual private gateway.

- Private – Instances in private subnets can connect to other VPCs or your on-premises network through a private NAT gateway. You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You cannot associate an elastic IP address with a private NAT gateway. You can attach an internet gateway to a VPC with a private NAT gateway, but if you route traffic from the private NAT gateway to the internet gateway, the internet gateway drops the traffic.

A NAT gateway is for use with IPv4 traffic only. To enable outbound-only internet communication over IPv6, use an egress-only internet gateway instead.

You are going to create an **Internet Gateway**.

1. Click on **Internet Gateways** (located on the left side under **Virtual private cloud**).
2. Click **Create internet gateway** (located in the top left corner)

Create a new Internet Gateway with the following:

1. Name: **Wordpress Gateway**
2. Click **Create internet gateway**
3. Once created, click on the **Actions** dropdown and select **Attach to VPC**.
4. In the **Available VPCs** input field, select your **Wordpress VPC**.
5. Click **Attach internet gateway** to attach it to your Wordpress VPC.
6. Once completed, your **Wordpress Gateway** should display the following:

Route Tables

You are going to create **Route tables** in your **VPC** to allow traffic from within your VPC to be routed externally through the **Internet Gateway** you created. In the search box at the top, type VPC.

1. Click on **Route Tables** (located on the left side under **Virtual private cloud**. See screenshot for clarity).

2. Click on your **Route table ID**. Find your default route table for your Wordpress VPC and add the name: **VPC-local Route Table**

3. Go back to the main **Route Tables** screen.

4. Click **Create route table** (top right corner).

Create a second route table:

5. Name: **Wordpress Website Route Table**
6. VPC: **Wordpress VPC**

7. Click **Create route table** (bottom right corner).
8. Click **Edit routes** and add the following routes. The first route may already exist.

- Route Entry 1:
  - Destination: **10.0.0.0/16**
  - Target: **local**
- Route Entry 2:
  - Destination: **0.0.0.0/0**
  - Target: **Internet Gateway – Wordpress Gateway**

9. View the following screenshot to confirm your settings are correct. If they are, click **Save changes**.

Security Groups

Security Group settings are located in the left side navigation under **Security** > **Security Groups**. Click on **Security Groups**. Note: You can access **Security groups** through **EC2** as well (as you did in lab 5). The menu they are under is different.

Click on **Create security group** and create a security group with the following settings

1. Security group name: **Wordpress Website SG**
2. Description: **Allows HTTP traffic inbound**
3. VPC: **Wordpress VPC**
4. Inbound Rules:
5. Allow HTTP

- Type: **HTTP**
- Source: **Anywhere – IPv4 (0.0.0.0/0)**

6. Allow SSH

- Type: **SSH**
- Source: **Anywhere – IPv4 (0.0.0.0/0)**

> Warning: Do **not** modify the **outbound** rules.

Verify your inbound rules with the following screenshot.

Click **Create security group** (bottom right).

Repeat the above steps to create another security group with the following settings:

1. Name: **Wordpress Database SG**
2. Description: **Allows MySQL traffic locally**
3. VPC: **Wordpress VPC**
4. Inbound Rule:

- Type: **MYSQL/Aurora**
- Source: **Custom** (Select *Wordpress Website SG* in the search field)

> Warning: Do **not** modify the **outbound** rules.

Verify your inbound rules with the following screenshot.

Editing Public Subnet route table associations

1. Click on **Subnets** under **Virtual private cloud** (left side).
2. Check the box beside **Public Subnet 1** Edit both public subnets' route table associations to: **Wordpress Website Route Table**
3. Click **Actions** > **Edit route table association**
4. Select **Wordpress Website Route Table** in the **Route table ID** dropdown menu. See the following screenshot.
5. Click save.

Repeat the steps for **Public Subnet 2**

Creating a new instance (wordpress)

Create a new instance in AWS (like you did in **Lab 1**), with the following configuration:

1. **Name:** wordpress
2. **OS:** Ubuntu
3. **Amazon Machine Image (AMI):** Make sure Ubuntu Server 24.04 is selected

4. Use your existing key pair. If you lost your key, then generate a new one. Don't lose this one.
5. **Network Settings:** Click **edit**

- **VPC:** Select the **Wordpress VPC** you created.
- **Security Group**: Select the **Wordpress Website Security Group** you created.
- **Subnet**: Select **Public Subnet 1**

1. Leave the default user (**ubuntu**). You do not need to add your user.

Verify your settings are correct and click **Launch Instance**.

Once the instance has created, confirm you can connect to it using:

- EC2 Instance Connect
- From the command line

Screenshots for submission

Take screenshots showing your 4 new subnets (Public Subnet 1, Public Subnet 2, Private Subnet 1 & Private Subnet 2), Route Table and Internet Gateway have been created.

Task 2: Database

Start your session in the Learner Lab by clicking on the **Start Lab** button. Once the red dot has turned green, click on it to enter the Learner Lab and access the AWS Console interface. You are going to create a new RDS instance.

From the **Console Home** navigate to **Database** > **RDS**. See the following screenshot for reference.

Click **Create database** (part way down the screen). Use the following options.

1. Standard create
2. Engine type: **MariaDB**
3. Engine Version: **MariaDB 11.4.4** (or current latest version available)
4. Templates: **Free tier**
5. DB instance identifier: **wordpress-elasticbeanstalk**
6. Master username: **admin**
7. Credentials management: **Self Managed**
8. Auto generate a password: **Checked**
9. DB instance class: **db.t3.micro**
10. Allocated storage: **5 GiB**
11. Enable storage autoscaling: **Unchecked**
12. Virtual private cloud (VPC): **Wordpress VPC**
13. DB subnet group: **Create new DB Subnet Group** (if you're redoing your database creation, there will already be an entry here. Make sure you're using the *Wordpress VPC* in the setting above!)
14. Public access: **No**
15. VPC security group: **Choose existing**
16. Existing VPC security groups:
    i. **Remove default VPC**
    ii. **Add *Wordpress Database SG*** (look to see that it's there below the dropdown after you select it)
17. Availability Zone: **us-east-1a**
18. Monitoring > Enable Enhanced monitoring: **Unchecked**
19. Below the Monitoring section, Additional configuration > Initial database name: **wordpress** (Write the database name down! You will need this later.)
20. Enable automated backups: **Unchecked**
21. Enable encryption: **Unchecked**

Click **Create database**.

This will take a few minutes to create. Once the database has finished creating, click on the *View connection details* button by the green success message at the top of the page. This gives you your database password.

Store the following connection information about your RDS instance in your lab logbook or a saved document. You'll need it later:

1. **Endpoint**
2. **Initial database name**
3. **Master username**
4. **Master password**

Connecting to your database from wordpress

Login to your **wordpress** instance, and issue the following command to connect to your database. Be sure to substitute the credentials you wrote down earlier.

```
mysql -u admin -h **endpoint** -p
```

Enter your Master password when prompted. You should see the following screen indicating a successful connection.

Issue the following command to display the databases.

```
show databases;
```

Disconnect from the database.

```
quit;
```

Task 3: Wordpress Source Code Modification

Explanation

When you install Wordpress, you can simply upload the source code and the first time you load the webpage, you'll be asked for database connector information.

However, Elastic Beanstalk applications are meant to be disposable.

Normally, when you add that database connector info, it is saved in a file called *wp-config.php* on the webserver VM. This is fine for a traditional setup. However, ***in Elastic Beanstalk, changes made to static HTML or PHP are not saved if the Beanstalk application restarts***, which it will do often. Whenever the application restarts, it will reload from the source zip file and the original, empty connector file. If you did this the traditional way, you'd have to constantly re-enter your DB connector info every time you started up your Learner Lab environment.

We *could* add the DB connector info to *wp-config.php* manually before we upload the source code, but there's a much better way.

We use **environment variables** to allow us to put all the info in the Elastic Beanstalk application wizard directly. That way, every time the application restarts and reloads from the source code zip, it'll then read our saved connector information from AWS itself. Read below for details and steps.

Note: All other information, like the Wordpress website name, users, theme settings, blog posts, etc., are saved in the actual database you created in RDS. This database does not get reset when the Elastic Beanstalk application restarts, so your actual blog data will remain intact.

Download and Unzip - Local Computer

1. On your local computer, download the current Wordpress source code from here: https://wordpress.org/latest.zip
2. Unzip the file. You should end up with a *wordpress* directory. (Do not delete the original .zip file)

Modify Wordpress Configuration File

Duplicate and Open Configuration File

1. In the local *wordpress* folder, find a file called: **wp-config-sample.php**
2. Duplicate this file, and call it: **wp-config.php**
3. Open **wp-config.php** in a text editor. You will want something that supports syntax highlighting., such as the default (graphical) text editor in Ubuntu, or something fancier like Visual Studio Code.

Adding Database Connector Info as Environment Variables

In this file (wp-config.php), you will be adding database connector information as ***environment variables***, not the actual connector information. (We'll add that information later.)

Find the following lines and add the bolded values:

1. define('DB_NAME', **getenv('DB_NAME')**);
2. define('DB_USER', **getenv('DB_USER')**);
3. define('DB_PASSWORD', **getenv('DB_PASSWORD')**);
4. define('DB_HOST', **getenv('DB_HOST')**);

Adding Authentication Unique Keys and Salts as Environment Variables

In the same file (wp-config.php), you'll be adding the authentication keys and salts as ***environment variables***.

Find the following lines and add the bolded values:

1. define('AUTH_KEY', **getenv('AUTH_KEY')**);
2. define('SECURE_AUTH_KEY', **getenv('SECURE_AUTH_KEY')**);
3. define('LOGGED_IN_KEY', **getenv('LOGGED_IN_KEY')**);
4. define('NONCE_KEY', **getenv('NONCE_KEY')**);
5. define('AUTH_SALT', **getenv('AUTH_SALT')**);
6. define('SECURE_AUTH_SALT', **getenv('SECURE_AUTH_SALT')**);
7. define('LOGGED_IN_SALT', **getenv('LOGGED_IN_SALT')**);
8. define('NONCE_SALT', **getenv('NONCE_SALT')**);

Figure 1: Adding database connector information to wp-config.php.

Save the file.

Zip As New File and Rename - Local Computer

1. Find the **wordpress** folder on your local computer.
2. *Zip the entire wordpress directory*, not just the files inside. (Use the zip compression protocol. Don't use something else like .rar.)

3. Rename your new zip file: **wordpress-6.7.2-*modded*.zip** (Use whatever version the source zip file has.)

Task 4: Elastic Beanstalk

Navigate to **Compute** > **Elastic Beanstalk**. See the following screenshot for reference.

Click **Create application**, and use the following settings:

Environment Tier

Select: **Web server environment**

Main settings

1. Application name: **wordpress**
2. Environment name: **Wordpress-env**
3. Platform: **PHP**
4. Platform branch: **PHP 8.4** (or current latest)
5. Application code: **Upload your code**
6. Version label: **wordpress-6.7.2** (Use the version from your zip filename)
7. Local file: **wordpress-6.7.2-*modded*.zip** (From your local computer)
8. Presets: **Single instance (free tier eligible)**

Click next

Configure Service Access

Select: Use an existing service role

1. Service role: **LabRole**
2. EC2 key pair: **vockey**
3. IAM instance profile: **LabInstanceProfile**

Click next

Set up networking, database and tags

1. VPC: **Wordpress VPC**

Instance Settings

1. Public IP address: **Checked**
2. **Instance** subnets: **Public Subnet 1, Public Subnet 2** (both checked)

Click next

Database settings

1. **Database** subnets: **Private Subnet 1, Private Subnet 2** (both checked)

Click **Enable database**

1. Username: admin
2. Password: *The password you copied and wrote down earlier*

Click next

Configure instance traffic and scaling

1. EC2 Security Groups: **Wordpress Website SG** & **Wordpress Database SG** (both checked)
2. Leave the rest default

Click next

Configure updates, monitoring and logging

Monitoring

1. System: **Basic**

Managed platform updates

1. Managed updates: **Unchecked**

Email notifications

1. Email notification: **Add your Seneca email**

Platform Software

Before beginning this section, you will need two things:

1. Your database connector information (you saved this, right?)
2. Randomly generated auth keys and salts from here: https://api.wordpress.org/secret-key/1.1/salt/ (it's a good idea to save these in a text file, too)

Container Options

1. Proxy server: **Apache**
2. Document root: **/wordpress**
3. Click **Add environment property** and add the following **Environment properties**
    i. DB_HOST: **your RDS database URL**
    ii. DB_NAME: **wordpress**
    iii. DB_USER: **admin**
    iv. DB_PASSWORD: **your auto-generated database password**
    v. AUTH_KEY: **(use gathered info from salt page)**
    vi. SECURE_AUTH_KEY: **(use gathered info from salt page)**
    vii. LOGGED_IN_KEY: **(use gathered info from salt page)**
    viii. NONCE_KEY: **(use gathered info from salt page)**
    ix. AUTH_SALT: **(use gathered info from salt page)**
    x. SECURE_AUTH_SALT: **(use gathered info from salt page)**
    xi. LOGGED_IN_SALT: **(use gathered info from salt page)**
    xii. NONCE_SALT: **(use gathered info from salt page)**

Hint: None of these values should have single quotes in them. (i.e. ')

*Figure 2: Adding database connector information, auth keys and salts to your Elastic Beanstalk application as static **Environment Variables**.*

Click next.

Review options

Review all settings and ensure they match the instructions above. Once you hit **Submit**, the application will take several minutes to create.

Create the application.

click **Submit** when ready.

While you wait for the creation to complete, check your e-mail to confirm your notification subscription.

Task 5: Site Configuration & Blog Posts

Open the URL presented in the Wordpress EBS instance and begin the site setup.

Site Information

Set the following site information:

1. Site Title: **Your name's blog**
2. Username: ***yourSenecaUsername***
3. Password: **Choose a strong password** (do not reuse the DB password!)
4. Your Email: ***yourSenecaEmailAddress***
5. Search engine visibility: **Unchecked**

> If you get a message indicating a failure to connect, make sure you zipped the **wordpress** folder and it's contents only. You can rezip the file and click **Upload and deploy** if necessary.

Blog Posts:

Delete the first template post. In your own words, answer the following questions in individual blog posts:

Blog Post: 1

How was this assignment? What was the most difficult part of this assignment for you? What was the easiest part for you?

Blog Post: 2

In the context of this assignment, briefly describe the function of the following:

1. VPCs
2. Subnets
3. Security groups
4. Route tables
5. Internet gateways

Assignment Submission

Your work will be evaluated by accessing your site directly.

To formally submit, you must include the following in a word document:

1. The URL of your new site. (The main public landing page, not the admin view.)

Plus screenshots showing:

1. 4 new subnets (Public Subnet 1, Public Subnet 2, Private Subnet 1 & Private Subnet 2)
2. Route Table
3. Internet Gateway have been created.
4. A single full-browser screenshot showing your active and complete Wordpress blog.
5. Attach a single full-browser screenshot showing Blog Post 1.
6. Attach a single full-browser screenshot showing Blog Post 2.

Once submitted, you can leave your Elastic Beanstalk application running, but **shutdown your Learner Lab environment**.

Previous
« Assignment 1