Hosting a Wordpress Application on ECS Fargate with RDS, Parameter Store, and Secrets Manager

01:51:02 Exit Lab ✓ Complete Lab

Guided Mode 2 hours 30 minutes duration Practitioner Rate this lab

Videos Guide

Hosting a Wordpress Application on ECS Fargate with RDS, Parameter Store, and Secrets Manager

Introduction

During this hands-on lab you will explore how to deploy a WordPress application using several AWS services, including: Amazon RDS, AWS Systems Manager, Parameter Store, Amazon ECS, Amazon ECR, and Application Load Balancers.

Solution

Log in to the AWS Management Console using the credentials provided on the lab instructions page. Make sure you're using the `us-east-1` Region.

Verify Existing Infrastructure

Before you begin, please verify the below resources exist and if any of these do not exist, please exit the lab and restart.

Type	Name
Application Load Balancer	OurApplicationLoadBalancer
Security Group	ALBAllowHttp
Cloud9 Environment	OurCloud9Environment
IAM Role	OurEcsTaskExecutionRole
IAM Role	OurEcsTaskRole

Create Database Subnet Group

First, we need to create our custom subnet group to host our DB instance.

1. Navigate to the **Amazon RDS** service.
2. Find and select **Subnet groups** from the menu.
3. click **Create DB subnet group**.
4. Name it `database-subnet-group`.
5. Enter the description of your choosing.
6. Choose the **Your Custom VPC** for the *VPC* options.
7. Move to the *Add subnets* section.
8. Under *Availability Zones*, select **us-east-1a**, **us-east-1b**, and **us-east-1c**.
9. For *Subnets*, select the subnets with the CIDRs of `10.0.20.0/24`, `10.0.21.0/24`, and `10.0.22.0/24`.
10. Click on **Create**.

Create the Amazon RDS Instance

Now, we can create our RDS instance.

1. Navigate to the **Amazon RDS** service.
2. Under *Databases* find and select **Create database**.
3. Select **MySQL** from the list of engine types.
4. Leave the default *Edition* selected, which should be **MySQL Community**.
5. For *Engine version* the latest version should populate by default.
6. Under *Templates* select **Free Tier**.
7. Move to the *Settings* section. For *DB cluster identifier*, change the name to `wordpress`.

8. Leave the *Master username* set to `admin`.
9. Check the box for the option to **Manage master credentials in AWS Secrets Manager**.
10. Leave the default encryption key for the Secrets Manager credentials.
11. Move to the *Instance configuration* section. For *DB instance class*, under *Burstable classes* select **db.t4g.micro**.
12. For the *Storage* section, set the *Storage type* to **General Purpose SSD (gp3)**.
13. Set the allocate storage to `20 GiB`.
14. Under the *Connectivity* section, select **Don't connect to an EC2 compute resource**.
15. Place the database in the VPC titled **Your Custom VPC** (*Do not use the default VPC!*)
16. Choose the **database subnet group** from the *DB subnet group* dropdown.
17. Set *Public access* to **No**.
18. For *VPC security group (firewall)* select **Create new**, name it `database-sg`, and then select **No preference** for *Availability Zone*.
19. Skip down to the *Additional configuration* dropdown menu near the bottom of the page.
20. Within the *Database options* set the *Initial database name* to `wordpress`.
21. Leave the rest of the options set to the defaults.
22. Find and select **Create database**. Creation of the DB cluster could take several minutes. While you wait for this to become available, you need to edit the Security Group.
23. Navigate to the **Amazon EC2** console in a new tab.
24. Find and select the **Security Groups** menu.
25. Find and select the `database-sg` Security Group.
26. Select *Inbound rules* and then click **Edit inbound rules**.
27. Delete the existing rule and add the following rule:

TYPE	PROTOCOL	PORT RANGE	SOURCE	VALUE	DESCRIPTION - OPTIONAL
MYSQL/Aurora	TCP	3306	Custom	`10.0.0.0/16`	`Allow MySQL access from VPC`

28. Click **Save rules**.
29. Move on only after the database is **available**!

Create the Parameter Store Parameters and Verify Secrets Manager Secret

Let's create our hidden values and plaintext parameters.

1. Navigate to **Parameter Store** in a new tab.
2. Click **Create Parameter**.
3. Create the following **2** parameters from the table below. You can get your `wordpress` database endpoint from the *Connectivity & security* section of the RDS page.

NAME	DESCRIPTION	TIER	TYPE	DATA TYPE	VALUE
/dev/WORDPRESS_DB_HOST	Wordpress RDS endpoint	Standard	String	text	**YOUR_RDS_ENDPOINT** `:3306` (Example: *wordpress-rds.cc5tzmus2oai.us-east-1.rds.amazonaws.com:3306*
/dev/WORDPRES	Wordpress RDS	Stan	SecureString	text	`wordpress`

NAME	DESCRIPTION	TIER	TYPE	DATA TYPE	VALUE
S_DB_NAME	Database Name	dard	(*default encryption*)		

4. Once you have created the parameters above, open **AWS Secrets Manager** in a new tab.
5. Verify there is a new secret there with a naming convention similar to *rds!db-1380b131-f0b2-4ef3-833f-4ab7a78f29fd*.
6. Select the secret, and then find and click on **Retrieve secret value** to view your admin credentials.
7. To ensure this is the latest version, click on the **Versions** tab and make sure there is only one *AWSCURRENT* tag.
8. Leave these two tabs open, as we will reference them later on for our containers.

Create the ECR Repository and Image

1. Navigate to **Elastic Container Registry** in a new tab.
2. Under *Private registry* on the left-hand side menu, select **Repositories**.
3. Click on **Create repository**.
4. For *Visibility settings* select **Private**.
5. For *Repository name* enter `wordpress` .
6. Leave *Tag immutability* **disabled**.
7. **Enable** the *Scan on push* option.
8. Leave *KMS encryption* **disabled**.
9. Click on **Create repository**.
10. Select your newly created `wordpress` repository.

11. Open a new tab for the **AWS Cloud9** service.
12. Find and select the `OurCloud9Environment` environment.
13. Click on **Open in Cloud9** (*Dismiss any message that may popup initially*).
14. Once loaded, select and expand the bottom terminal session.
15. Now, navigate to **AWS IAM** in a new tab.
16. Find and select **Users** from the menu.
17. Find your `cloud_user` and select it.
18. Click on **Security Credentials**.
19. Under *Access keys*, click on **Create access key**.
20. Select the **Command Line Interface (CLI)** radio button and click the box to confirm you understand the recommendations.
21. Click **Next**.
22. Enter your own description tag value and click **Create access key**.
23. Under the *Retrieve access keys* page, copy the **Access key** value, then navigate back to your Cloud9 tab.
24. In Cloud9, run `aws configure --profile cloud_user`.
25. Paste in your **Access key** and hit enter.
26. Go back to your IAM access key tab and then copy the **Secret access key** value.
27. Navigate back to Cloud9 and paste in the value, then hit enter.
28. Set the default region to `us-east-1`.
29. Set default output to `json`.
30. If you get a popup about *AWS managed temporary credentials*, select **Cancel** and then **Re-enable after refresh**.
31. Test that you can perform an AWS CLIv2 command (Example: `aws s3 ls --profile cloud_user`).
32. Leave your Cloud9 tab running, and Navigate back to the **ECR tab**.
33. Click on **View push commands**.

34. Copy and paste **Step 1** into your Cloud9 terminal. Before entering, add the `--profile cloud_user` to the portion before the pipe. Example below:

    ```
    aws ecr get-login-password --region us-east-1 --profile cloud_user |
    docker login --username AWS --password-stdin 294991935974.dkr.ecr.us-east-
    1.amazonaws.com
    ```

35. You should see a **Login Succeeded** message.
36. Pull the latest Docker image for WordPress running this command:

    ```
    docker pull wordpress:latest
    ```

37. Once complete, tag the image we want to push by copying and pasting **Step 3** from the ECR push commands prompt. (*We don't need to run Step 2 because the image is already built*).
38. Verify the new image exists locally by running the following command:

    ```
    docker image ls
    ```

39. Now run **Step 4** from the ECR push commands.
40. It should push our new image to ECR, which you can verify in the ECR console after completion.
41. After verifying the image is in place, leave the ECR tab open and then move on to the next section!

Create the Amazon ECS Task Definition

1. Navigate to the **Amazon ECS** service in a new tab.
2. Once there, find and select **Task definitions** from the left-hand menu.
3. Click on **Create new task definition**.
4. Enter `wordpress-td` for the *Task definition family*.
5. Under *Infrastructure requirements*, for *Launch type*, select **AWS Fargate**.
6. Leave the other defaults as-is, and then select **OurEcsTaskRole** for the *Task role*.
7. For the *Task execution role*, find and select the already created role called `OurEcsTaskExecutionRole`.

8. Under *Container - 1*, enter the following settings for *Container details*. You can get your `wordpress` image URI from the ECR page.

NAME	IMAGE URI	ESSENTIAL CONTAINER
`wordpress`	Your ECR Image URI from the custom image you pushed (Example: *294991935974.dkr.ecr.us-east-1.amazonaws.com/wordpress:latest*)	Yes

9. Leave the other defaults and find the *Environment variables* section.
10. Click on **Add environment variable** and then fill in the information for each of the below **4** variables. **PLEASE NOTE THE ARN SYNTAX OF THE SECRETS MANAGER SECRET.**

KEY	VALUE TYPE	VALUE
WORDPRESS_DB_HOST	ValueFrom	ARN of the respective Parameter Store parameter (Example: *arn:aws:ssm:us-east-1:552898056824:parameter/dev/WORDPRESS_DB_HOST*
WORDPRESS_DB_NAME	ValueFrom	ARN of the respective Parameter Store parameter (Example: *arn:aws:ssm:us-east-1:552898056824:parameter/dev/WORDPRESS_DB_NAME*
WORDPRESS_DB_USER	ValueFrom	ARN of the respective Secrets Manager RDS secret, specifying the specific key value by adding `:username::` at the end (Example: *arn:aws:secretsmanager:us-east-1:552898056824:secret:rds!db-1380b131-f0b2-4ef3-833f-4ab7a78f29fd-BAsjMA:username::*

KEY	VALUE TYPE	VALUE
WORDPRESS_DB_PASSWORD	ValueFrom	ARN of the respective Secrets Manager RDS secret, specifying the specific key value by adding `:password::` at the end (Example: *arn:aws:secretsmanager:us-east-1:552898056824:secret:rds!db-1380b131-f0b2-4ef3-833f-4ab7a78f29fd-BAsjMA:password::*

11. Click on **Create** at the bottom.

Create the ECS Cluster and Service

1. Within the ECS service, find and select **Clusters**.
2. Click **Create cluster**.
3. For *Cluster name* enter `Wordpress-Cluster` .
4. Under **Infrastructure**, select `AWS Fargate (serverless)` .
5. Click on **Create**.
6. Wait for your cluster to be created before moving on. If you get any service related errors, please navigate to the CloudFormation template that is created for you by the service and retry the deployment.
7. Select your **Wordpress-Cluster**.
8. Under the *Services* tab, click on **Create**.
9. For *Compute options* select **Launch type**.
10. Make sure the *Launch type* is set to **FARGATE**.
11. Make sure *Platform version* is set to **LATEST**.
12. Move to *Deployment configuration*.
13. For *Application type* select **Service**.
14. For *Family*, under *Task definition*, choose your `wordpress-td` task definition from the dropdown and use the `LATEST` version.

15. Name your service `wordpress-service`.
16. Set desired tasks to `1`.
17. Skip to the *Networking* section and select **Your Custom VPC**.
18. For *Subnets*, click **Clear current selection** and only select the ones titled **Private Subnet**.
19. Choose **Create a new security group** for *Security group*.
20. Name it `app-sg` and give it your description of choice.
21. For *Inbound rules for security groups*, enter the following information:

TYPE	PROTOCL	PORT RANGE	SOURCE	VALUES
HTTP	TCP	80	Source group	Security Group of the `ALBAllowHttp`

22. Set *Public IP* to be turned **off**.
23. Move to *Load balancing*, for *Load balancer type*, select **Application Load Balancer**.
24. Choose **Use an existing load balancer**.
25. Find the load balancer named `OurApplicationLoadBalancer`.
26. Set the *Health check grace period* to `30 seconds`.
27. Leave the *Listener* values as default.
28. For *Target group*, name it `wordpress-tg`.
29. For the *Health check path*, enter `/wp-admin/images/wordpress-logo.svg`. (*We do this because we are needing to set up WP still. Otherwise, it will fail the initial health check*).
30. Leave the rest of the values as the defaults.
31. Skip to the bottom and select **Create**.
32. Wait until your service is up and running, then you can move on!

Connect to the Application

1. Once you have a service up and running, you should see a task listed under the *Tasks* tab within the cluster.
2. Navigate to the **Amazon EC2** console.
3. Find and select **Load balancer**.
4. Find the `OurApplicationLoadBalancer` and choose it.
5. Now, copy and paste your ALB DNS name into a new tab, ensuring you use **HTTP**.
6. You should be greeted by the Wordpress setup page!

Conclusion

Congratulations — you've completed this hands-on lab!

Tools

Lab Diagram Instant Terminal

Credentials

How do I connect?

AWS Account

Username

cloud_user

Password

V@PjO2tId7!B#y@0U5ua

Open Link in Incognito Window

Additional Resources

Access the code to complete this lab from the GitHub repo or the lab guide if it is available to you.

Learning Objectives

0 of 12 completed

Optional: Run progress checks to confirm you've completed the objectives

- ✓ Verify Resources and Create Database Subnet Group
- ✓ Create the Amazon RDS Instance
- ✓ Update the RDS Security Group Rules
- ✓ Create the Parameter Store Parameters and Verify Secrets Manager Secret
- ✓ Create the Private ECR Repository
- ✓ Create IAM User Access Keys
- ✓ Configure AWS Cloud9
- ✓ Push Image to ECR Repo from Cloud9
- ✓ Create the Amazon ECS Task Definition
- ✓ Create the Amazon ECS Cluster
- ✓ Create the Amazon ECS Service
- ✓ Test the Application