

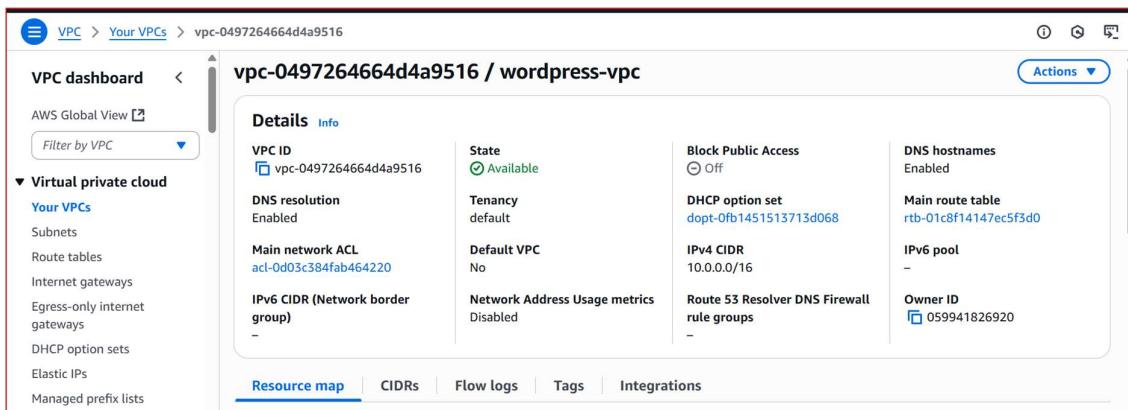
Architecture Overview

- **ECS Fargate:** Serverless container hosting for WordPress
- **RDS MySQL:** Managed database service
- **ALB:** Application Load Balancer for traffic distribution
- **VPC:** Virtual Private Cloud for network isolation
- **ECR:** Elastic Container Registry for WordPress image

Step 1: Create VPC and Networking

1.1 Create VPC

1. Go to **VPC Console** → **Create VPC**
2. Choose **VPC and more**
3. Configure:
 - **Name:** wordpress-vpc
 - **IPv4 CIDR:** 10.0.0.0/16
 - **Availability Zones:** 2
 - **Public subnets:** 2
 - **Private subnets:** 2
 - **NAT gateways:** 1 per AZ
 - **VPC endpoints:** None
4. Click **Create VPC**



1.2 Create Security Groups

Database Security Group

1. Go to **EC2 Console** → **Security Groups** → **Create Security Group**

2. Configure:

- **Name:** wordpress-db-sg
- **Description:** Security group for RDS MySQL
- **VPC:** Select wordpress-vpc

3. Inbound Rules:

- Type: MySQL/Aurora (3306)
- Source: Custom (select ECS security group - we'll create this next)

4. Click **Create Security Group**

The screenshot shows the AWS Security Groups console for a security group named 'sg-024dadcc4495e82b27 - wordpress-db-sg'. The 'Details' section includes fields for Security group name (wordpress-db-sg), Security group ID (sg-024dadcc4495e82b27), Description (Security group for RDS MySQL), VPC ID (vpc-0497264664d4a9516), Owner (059941826920), Inbound rules count (1 Permission entry), and Outbound rules count (1 Permission entry). Below the details, there are tabs for Inbound rules, Outbound rules, Sharing - new, VPC associations - new, and Tags. The 'Inbound rules' tab is selected, showing one rule: sgr-00f6ad1a3a8b0f1c1 (Name) allowing TCP (Protocol) port 3306 (Port range) from source sg-026f826.

ECS Security Group

1. Create another security group:

- **Name:** wordpress-ecs-sg
- **Description:** Security group for ECS tasks
- **VPC:** Select wordpress-vpc

2. Inbound Rules:

- Type: HTTP (80), Source: Custom (select ALB security group - we will create this next)
- Type: HTTPS (443), Source: Custom (select ALB security group - we will create this next)

3. Click **Create Security Group**

sg-026f8261b27938d80 - wordpress-ecs-sg

Details

Security group name wordpress-ecs-sg	Security group ID sg-026f8261b27938d80	Description Security group for ECS tasks	VPC ID vpc-0497264664d4a9516
Owner 059941826920	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing - new](#) | [VPC associations - new](#) | [Tags](#)

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-01acdde3cb54efc90	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-02ea0c0163c8b0b6	IPv4	HTTPS	TCP	443	0.0.0.0/0

Load Balancer Security Group

1. Create another security group:
 - **Name:** wordpress-alb-sg
 - **Description:** Security group for ALB
 - **VPC:** Select wordpress-vpc
2. **Inbound Rules:**
 - Type: HTTP (80), Source: 0.0.0.0/0
 - Type: HTTPS (443), Source: 0.0.0.0/0
3. Click **Create Security Group**

sg-05a8470aaa07ae6e6 - wordpress-alb-sg

Details

Security group name wordpress-alb-sg	Security group ID sg-05a8470aaa07ae6e6	Description Security group for ALB	VPC ID vpc-0497264664d4a9516
Owner 059941826920	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry	

[Inbound rules](#) | [Outbound rules](#) | [Sharing - new](#) | [VPC associations - new](#) | [Tags](#)

Inbound rules (2)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-09297ebeee98258ad	IPv4	HTTPS	TCP	443	0.0.0.0/0
-	sgr-04e5024f00bec5e65	IPv4	HTTP	TCP	80	0.0.0.0/0

1.3 Update Database and ECS Security Group

1. Go back to wordpress-db-sg
2. Edit inbound rules and change the source to wordpress-ecs-sg
3. Go back to wordpress-ecs-sg
4. Edit inbound rules and change the source to wordpress-alb-sg

Step 2: Create RDS MySQL Database

2.1 Create DB Subnet Group

1. Go to **RDS Console** → **Subnet Groups** → **Create DB Subnet Group**

2. Configure:

- **Name:** wordpress-db-subnet-group
- **Description:** Subnet group for WordPress database
- **VPC:** Select wordpress-vpc
- **Availability Zones:** Select both AZs
- **Subnets:** Select both private subnets

3. Click **Create**

The screenshot shows the AWS RDS Subnet Groups page. The left sidebar has links for Aurora and RDS, Dashboard, Databases, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL Integrations, Events, and Event subscriptions. The main content area shows the details for the subnet group 'wordpress-db-subnet-group'. It includes fields for VPC ID (vpc-0497264664d4a9516), ARN (arn:aws:rds:ap-south-1:059941826920:subgrp:wordpress-db-subnet-group), Supported network types (IPv4), and Description (Subnet group for WordPress database). Below this is a table titled 'Subnets (2)' showing two entries:

Availability zone	Subnet name	Subnet ID	CIDR block
ap-south-1b	wordpress-subnet-private2-ap-south-1b	subnet-02e718ea5d0b2f83a	10.0.144.0/20
ap-south-1a	wordpress-subnet-private1-ap-south-1a	subnet-0b930fb47ba0ba01	10.0.128.0/20

2.2 Create RDS Instance

1. Go to **RDS Console** → **Create Database**

2. **Engine Options:**

- **Engine:** MySQL
- **Version:** Latest (e.g., 8.0.35)
- **Template:** Free tier (or Production for production use)

3. **Settings:**

- **DB Instance Identifier:** wordpress-db
- **Master Username:** admin
- **Master Password:** Create a secure password (save this!)

4. **DB Instance Class:**

- **Free tier:** db.t3.micro
- **Production:** db.t3.small or larger

5. **Storage:**

- Storage Type: General Purpose SSD (gp2)
- Allocated Storage: 20 GB (minimum)

6. **Connectivity:**

- **VPC:** wordpress-vpc
- **DB Subnet Group:** wordpress-db-subnet-group
- **VPC Security Groups:** wordpress-db-sg
- **Availability Zone:** No preference
- **Public Access:** No

7. **Additional Configuration:**

- **Initial Database Name:** wordpress
- **Backup Retention:** 7 days
- **Monitoring:** Enable Enhanced Monitoring (optional)

8. Click **Create Database**

Note: Database creation takes 10-15 minutes. Note down the endpoint when ready.

The screenshot shows the AWS Aurora and RDS console with the database named 'wordpressdb'. The 'Summary' tab is active, displaying information such as the DB identifier (wordpressdb), status (Available), CPU usage (3.22%), role (Instance), current activity (0 connections), engine (MySQL Community), and region & AZ (ap-south-1b). The 'Connectivity & security' tab is selected, showing the endpoint (wordpressdb.cnw6q604gt5c.ap-south-1.rds.amazonaws.com), port (3306), VPC (wordpress-vpc), and subnet group (wordpress-db-subnet-group). Other tabs include Monitoring, Logs & events, Configuration, Zero-ETL integrations, Maintenance & backups, and Data migration.

Step 3: Create Application Load Balancer

1. Go to **EC2 Console** → **Load Balancers** → **Create Load Balancer**

2. Choose **Application Load Balancer**

3. **Basic Configuration:**

- **Name:** wordpress-alb
- **Scheme:** Internet-facing
- **IP Address Type:** IPv4

4. Network Mapping:

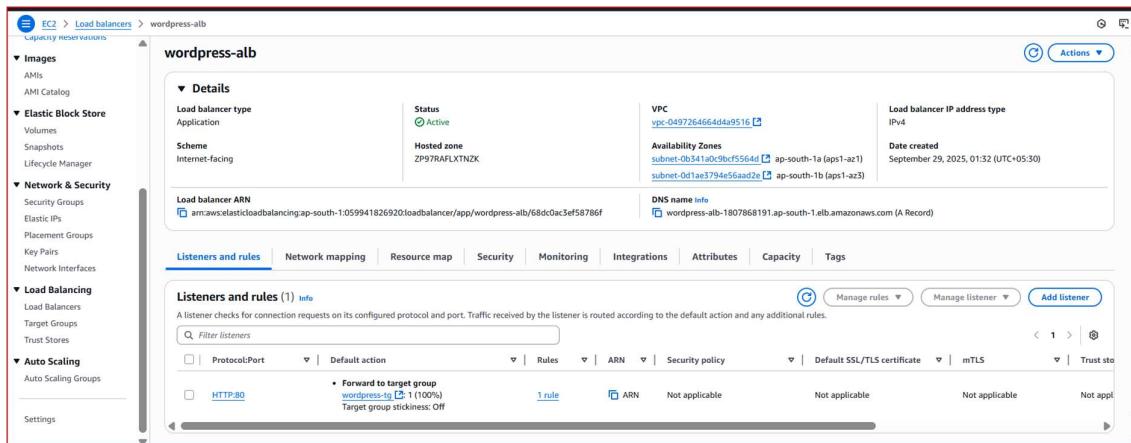
- **VPC:** wordpress-vpc
- **Mappings:** Select both public subnets

5. Security Groups: Select wordpress-alb-sg

6. Listeners and Routing:

- **Protocol:** HTTP, **Port:** 80
- **Default Action:** Create new target group
 - **Target Group Name:** wordpress-tg
 - **Target Type:** IP
 - **Protocol:** HTTP, **Port:** 80
 - **VPC:** wordpress-vpc
 - **Health Check Path:** /

7. Click Create Load Balancer



Step 4: Create ECS Cluster

1. Go to ECS Console → Clusters → Create Cluster

2. Cluster Configuration:

- **Cluster Name:** wordpress-cluster
- **Infrastructure:** AWS Fargate (serverless)

3. Monitoring: Enable Container Insights (optional)

4. Click Create

The screenshot shows the AWS ECS console interface. On the left, a sidebar lists 'Clusters', 'Namespaces', 'Task definitions', and 'Account settings'. Below that are links for 'Amazon ECR', 'Repositories', 'AWS Batch', 'Documentation', 'Discover products', and 'Subscriptions'. A 'Tell us what you think' feedback link is also present.

The main area displays the 'Cluster overview' for 'wordpress-cluster'. It shows the ARN (arn:aws:ecs:ap-south-1:059941826920:cluster/wordpress-cluster), Status (Active), CloudWatch monitoring (Default), and Registered container instances (none listed). Below this is a table for 'Services' with one entry: 'wordpress-task-service-4ibbxjr' (ARN: arn:aws:ecs:ap-south-1:059941826920:service/wordpress-cluster, Status: Active, Task definition: REPLICAS, Deployment: wordpress-task-2, Last deployment: September 29, 2025, 01:59 (UTC+5:30)).

Step 5: Create Task Definition

5.1 Create Task Definition

1. Go to **ECS Console** → **Task Definitions** → **Create New Task Definition**
2. **Task Definition Configuration:**
 - **Task Definition Family:** wordpress-task
 - **Launch Type:** AWS Fargate
 - **Operating System:** Linux/X86_64
 - **CPU:** 0.5 vCPU (512)
 - **Memory:** 1 GB (1024)
 - **Task Role:** ecsTaskExecutionRole (create if doesn't exist)
 - **Task Execution Role:** ecsTaskExecutionRole

5.2 Configure Container

1. **Container Details:**
 - **Name:** wordpress
 - **Image URI:** wordpress:latest
 - **Port Mappings:**
 - Container Port: 80
 - Protocol: TCP
 - Port Name: wordpress-80-tcp
2. **Environment Variables:**

WORDPRESS_DB_HOST=<RDS_ENDPOINT>:3306

WORDPRESS_DB_USER=admin

WORDPRESS_DB_PASSWORD=<YOUR_DB_PASSWORD>

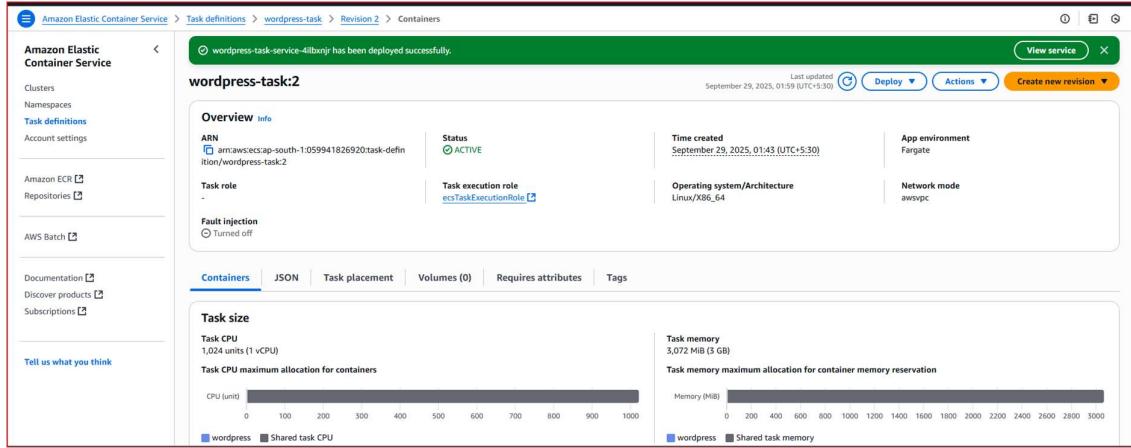
WORDPRESS_DB_NAME=wordpress

Replace <RDS_ENDPOINT> with your actual RDS endpoint and <YOUR_DB_PASSWORD> with your database password.

3. Logging (Optional):

- Enable **Use log collection**
- **Log Driver:** awslogs
- **Log Group:** Create new: /ecs/wordpress-task

4. Click **Create**



Step 6: Create ECS Service

1. Go to **ECS Console** → **Clusters** → Select wordpress-cluster

2. **Services** tab → **Create Service**

3. Deployment Configuration:

- **Application Type:** Service
- **Task Definition:** Select wordpress-task
- **Service Name:** wordpress-service
- **Desired Tasks:** 2

4. Networking:

- **VPC:** wordpress-vpc
- **Subnets:** Select private subnets
- **Security Groups:** Select wordpress-ecs-sg

- **Public IP:** Enabled (required for Fargate in private subnets with internet access)

5. Load Balancing:

- **Load Balancer Type:** Application Load Balancer
- **Load Balancer:** Select wordpress-alb
- **Target Group:** Select wordpress-tg
- **Container:** wordpress 80:80

6. Click Create Service

The screenshot shows the AWS ECS console with the following details:

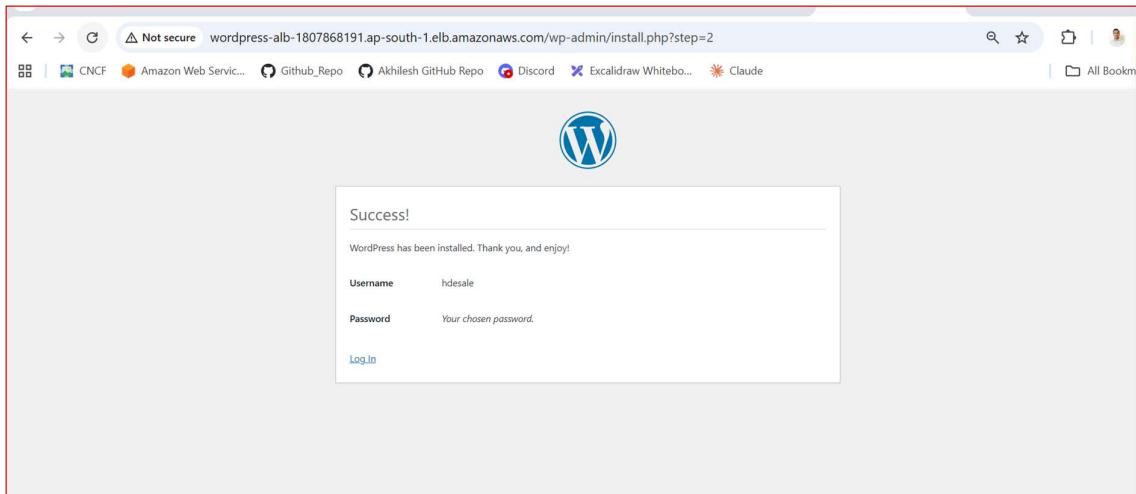
- Service name:** wordpress-task-service-4ilbxnjr
- Status:** Active
- Tasks (2 Desired):** 0 Pending | 2 Running
- Task definition:** revision wordpress-task-2
- Deployment status:** Success
- Health and metrics:** Tab selected.
- Service ARN:** arn:aws:ecs:ap-south-1:059941826920:service/wordpress-cluster/wordpress-task-service-4ilbxnjr
- Deployments current state:** 2 Completed tasks
- Created at:** September 29, 2025, 01:47 (UTC+5:30)
- Load balancer health:**
 - Load balancer:** wordpress-alb
 - Type:** Application Load Balancer
 - Port:** wordpress:80
 - Listeners:** HTTP:80
 - Target group:** wordpress-tg
 - Target health:** 2 Healthy | 0 Unhealthy

Step 7: Access WordPress using ALB DNS

The screenshot shows a web browser displaying the WordPress installation screen. A language selection dropdown is open, showing the following options:

- English (United States)
- Afrikaans
- አማርኛ
- Aragonés
- عربیة
- ଓଡ଼ିଆ
- অসমীয়া
- گوئندي اردو
- Azərbaycan dilı
- Беларуская мова
- Български
- বাংলা
- ດ້ວຍ
- Bosanski
- Català
- Čeština
- Cymraeg
- Dansk
- Deutsch (Österreich)
- Deutsch
- Deutsch (Sise)

A "Continue" button is visible at the bottom of the dropdown menu.



The Security Flow :

Internet → Load Balancer → WordPress Containers → Database

✓ Allowed

✓ Allowed

✓ Allowed

Internet → Database (Direct)

✗ BLOCKED by wordpress-db-sg

