

Deploying wordpress on AWS/Azure with RDS using Terraform, and Docker.

- Create a Terraform script to deploy a VM. The VM should be in public subnet.
- In the same VM Create a Dockerfile to deploy Apache webserver + PHP + Wordpress. Use COPY, ARG, RUN, ENTRYPOINT, CMD, WORKDIR in Dockerfile.
- Build the docker image, tag the image and push the docker image to Docker Hub and run the container.
- Deploy the RDS on AWS/Azure and it should be on private subnet.
- Connect your Wordpress container with RDS database.
- Setup the Wordpress and attach all the screenshot and code to Github.

Note- Please add the GitHub link while submitting the assignment.

Step 1: Setting Up the Local Environment

- Installed Terraform on your local machine.
- Installed the AWS CLI (Command Line Interface).
- Configured my AWS account using the aws configure command to set your access key, secret key, default region, and output format.

Step 2: Terraform Infrastructure as Code

- Wrote a Terraform script (main.tf) to define AWS resources for the project. The script included the following components:
 - Provider configuration
 - VPC (Virtual Private Cloud) declaration
 - Public subnet
 - Private subnet
 - Internet Gateway (IGW)
 - Network Address Translation (NAT) Gateway
 - Route tables
 - Security groups
 - EC2 instance (placed in the public subnet)
- After completing the Terraform script, ran the following Terraform commands:
 - terraform init: Initialised your working directory and prepared it for creating and managing the infrastructure.
 - terraform plan: Reviewed the execution plan to ensure it matched your intended changes.
 - terraform apply -auto-approve: Applied the changes to create the infrastructure. Used the -auto-approve flag for a non-interactive

deployment. This command deployed a virtual machine (EC2 instance) in the public subnet.

Step 3: Deploying Docker Container

- Wrote a Dockerfile on the deployed EC2 instance created using the Terraform script to customize the Docker image.
- Built the customized Docker image using the docker build command.
- Logged into your Docker Hub account using docker login.
- Pushed the Docker image to Docker Hub using the docker push command.
- Ran the Docker container on the EC2 instance in detached mode (-d) using the docker run command.

Step 4: Setting Up RDS (Relational Database Service)

- Set up an RDS instance using the AWS Management Console in a private subnet.

Step 5: Connecting the WordPress Application

- Connected the WordPress application to the Docker container hosting the RDS database, ensuring that the WordPress configuration was correctly set to use the RDS database endpoint.

Screenshots (screenshot and source code has been shared to github account)
screenshots for each of the following steps to accompany my documentation:

AWS CLI configuration.

```
diwakarraajanna@Diwakars-MacBook-Pro ~ % aws configure
AWS Access Key ID [None]: AKIA2LJG3WMFI45FAQGP
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
```

Terraform script (main.tf) showing the defined resources.

Wrote it in a visual studio code

```
main.tf x DOCKERFILE apache-config.conf
main.tf
1 provider "aws" {
2   region = "ap-south-1"
3 }
4
5 # VPC declaration
6 resource "aws_vpc" "my-wordpress-vpc" {
7   cidr_block = "20.0.0.0/16"
8   tags = {
9     Name = "wordpress-vpc"
10  }
11 }
12
13 # Public subnet
14 resource "aws_subnet" "public1" {
15   vpc_id      = aws_vpc.my-wordpress-vpc.id
16   cidr_block   = "20.0.1.0/24" # Adjusted CIDR block
17   availability_zone = "ap-south-1a"
18
19   tags = {
20     Name = "public-subnet1"
21   }
22 }
23
24 # Private subnets
25 resource "aws_subnet" "private1" {
26   vpc_id      = aws_vpc.my-wordpress-vpc.id
27   cidr_block   = "20.0.2.0/24" # Adjusted CIDR block
28   availability_zone = "ap-south-1a"
29
30   tags = {
31     Name = "private-subnet1"
32   }
33 }
34
35 # Internet Gateway
36 resource "aws_internet_gateway" "igw" {
37   vpc_id = aws_vpc.my-wordpress-vpc.id
38
39   tags = {
40     Name = "IGW"
41   }
42 }
43
44 # Elastic IP
45 resource "aws_eip" "eip" {
46   domain = "vpc"
47 }
48
49 # NAT Gateway
50 resource "aws_nat_gateway" "nat" {
51   allocation_id = aws_eip.eip.id
52   subnet_id     = aws_subnet.public1.id
53
54   tags = {
```

Terraform apply command output.

```
Plan: 12 to add, 0 to change, 0 to destroy.
aws_eip.eip: Creating...
aws_vpc.my-wordpress-vpc: Creating...
aws_eip.eip: Creation complete after 1s [id=eipalloc-0291f0ee53c3aba0d]
aws_vpc.my-wordpress-vpc: Creation complete after 2s [id=vpc-03bdf958858c41cd2]
aws_internet_gateway.igw: Creating...
aws_subnet.private1: Creating...
aws_subnet.public1: Creating...
aws_security_group.allow_ssh_http: Creating...
aws_internet_gateway.igw: Creation complete after 0s [id=igw-0f6c5c51e9dee4d1b]
aws_route_table.public: Creating...
aws_subnet.public1: Creation complete after 0s [id=subnet-0e70ade3e4a2d7542]
aws_subnet.private1: Creation complete after 0s [id=subnet-0934e913febbcf2eb]
aws_nat_gateway.nat: Creating...
aws_route_table.public: Creation complete after 1s [id=rtb-0e4c33662217a5565]
aws_route_table_association.public: Creating...
aws_route_table_association.public: Creation complete after 0s [id=rtbassoc-0ffc93e28039a8c86]
aws_security_group.allow_ssh_http: Creation complete after 2s [id=sg-0990eb5dd26abd8f8]
aws_instance.wordpress: Creating...
aws_nat_gateway.nat: Still creating... [10s elapsed]
aws_instance.wordpress: Still creating... [10s elapsed]
aws_nat_gateway.nat: Still creating... [20s elapsed]
aws_instance.wordpress: Still creating... [20s elapsed]
aws_nat_gateway.nat: Still creating... [30s elapsed]
aws_instance.wordpress: Still creating... [30s elapsed]
aws_instance.wordpress: Creation complete after 32s [id=i-04dc0c29f4cf795a3]
aws_nat_gateway.nat: Still creating... [40s elapsed]
aws_nat_gateway.nat: Still creating... [50s elapsed]
aws_nat_gateway.nat: Still creating... [1m0s elapsed]
aws_nat_gateway.nat: Still creating... [1m10s elapsed]
aws_nat_gateway.nat: Still creating... [1m20s elapsed]
aws_nat_gateway.nat: Still creating... [1m30s elapsed]
aws_nat_gateway.nat: Still creating... [1m40s elapsed]
aws_nat_gateway.nat: Creation complete after 1m45s [id=nat-05eee49fca7eec884]
aws_route_table.private: Creating...
aws_route_table.private: Creation complete after 2s [id=rtb-0fb3710dfdf48fa7d]
aws_route_table_association.private: Creating...
aws_route_table_association.private: Creation complete after 0s [id=rtbassoc-03f58d44204573f90]

Apply complete! Resources: 12 added, 0 changed, 0 destroyed.
```

The screenshot displays the AWS Management Console interface for the 'ap-south-1' region. The left-hand navigation pane shows various services, with 'Instances' selected. The main content area shows a list of instances, with one instance 'wordpress-ubuntu' (ID: i-04dc0c29f4cf795a3) in a 'Running' state. Below this, the 'Details' tab for the instance is expanded, showing a summary of its configuration. The instance is running on the 't2.micro' instance type, using the 'Ubuntu (Inferred)' AMI. It is associated with the 'vpc-03bdf958858c41cd2' VPC and the 'subnet-0e70ade3e4a2d7542' subnet. The instance has a public IP address of 13.233.90.155 and a private IP address of 10.0.0.1. The instance is also associated with the 'sg-0990eb5dd26abd8f8' security group. The console footer shows the copyright notice for Amazon Web Services India Private Limited.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4
wordpress-ubuntu	i-04dc0c29f4cf795a3	Running	t2.micro	2/2 checks passed...	No alarms	ap-south-1a	-	13.233.90.155

Instance: i-04dc0c29f4cf795a3 (wordpress-ubuntu)

Instance summary

Property	Value
Instance ID	i-04dc0c29f4cf795a3 (wordpress-ubuntu)
IPv6 address	-
Host name type	ip-20-0-1-21.ap-south-1.compute.internal
Answer private resource DNS name	-
Auto-assigned IP address	13.233.90.155 [Public IP]
IAM Role	-
IMDSv2	Optional
Platform	Ubuntu (Inferred)
Platform details	-
Public IPv4 address	13.233.90.155 open address
Instance state	Running
Private IP DNS name (IPv4 only)	ip-20-0-1-21.ap-south-1.compute.internal
Instance type	t2.micro
VPC ID	vpc-03bdf958858c41cd2 (wordpress-vpc)
Subnet ID	subnet-0e70ade3e4a2d7542 (public-subnet1)
AMI ID	ami-0f5ee92e2d653afc18
AMI name	-
Private IPv4 addresses	20.0.1.21
Public IPv4 DNS	-
Elastic IP addresses	-
AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more
Auto Scaling Group name	-
Monitoring	disabled
Termination protection	-

<input checked="" type="checkbox"/>	Name	Instance ID	In:
<input checked="" type="checkbox"/>	wordpress-ubuntu	i-04dc0c29f4cf795a3	<input checked="" type="checkbox"/>

Instance: i-04dc0c29f4cf795a3 (wordpress-ubuntu)

Details | Security | **Networking** | Storage | Status

▼ Networking details [Info](#)

Public IPv4 address
 13.233.90.155 | [open address](#)

Public IPv4 DNS
-

Subnet ID
 [subnet-0e70ade3e4a2d7542 \(public-subnet1\)](#)

Availability zone
-

Vpc

VPC dashboard

EC2 Global View

Filter by VPC

Select a VPC

▼ Virtual private cloud

Your VPCs [New](#)

Subnets

Route tables

Internet gateways

Egress-only internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

▼ Security

Network ACLs

Security groups

▼ DNS firewall

Rule groups

Domain lists

▼ Network Firewall

Firewalls

Firewall policies

Network Firewall rule groups

TLS inspection configurations

Network Firewall resource groups

Your VPCs (1/2) [Info](#)

Find resources by attribute or tag

☐

-

vpc-0b64ab95b1d45cd95

Available

172.31.0.0/16

-

dopt-02324c505c8be9c5e

rt

☒

wordpress-vpc

vpc-03bdf958858c41cd2

Available

20.0.0.0/16

-

dopt-02324c505c8be9c5e

rt

vpc-03bdf958858c41cd2 / wordpress-vpc

Details | **Resource map** [New](#) | CIDRs | Flow logs | Tags

VPC [Show details](#)

Your AWS virtual network

wordpress-vpc

Was the resource map helpful today?

Give us feedback as often as possible. We are improving continually.

Subnets (5)

Subnets within this VPC

ap-south-1a

public-subnet1

private-subnet1

ap-south-1b

private-subnet2

public-subnet2

ap-south-1c

private-subnet3

Route tables (3)

Route network traffic to resources

public-rt

rtb-09b810f38556018eb

private-rt

Network connections (2)

Connections to other networks

IGW

nat

Security group

The screenshot shows the AWS Management Console interface. The left sidebar contains navigation links for VPC dashboard, EC2 Global View, and various VPC and Security services. The main content area displays the details for a security group named 'allow_ssh_http' (ID: sg-0990eb5dd26abd8f8). The details section includes the security group name, ID, description ('Allow SSH and HTTP inbound traffic'), VPC ID, owner, inbound rules count (3), and outbound rules count (2). Below this, the 'Inbound rules' tab is selected, showing a table of three rules: one for SSH (port 22), one for HTTP (port 80), and one for Custom TCP (port 8080). All rules are for IPv4 and have a source of 0.0.0.0/0.

sg-0990eb5dd26abd8f8 - allow_ssh_http

Details

Security group name allow_ssh_http	Security group ID sg-0990eb5dd26abd8f8	Description Allow SSH and HTTP inbound traffic	VPC ID vpc-03bdf958858c41cd2
Owner 711435465482	Inbound rules count 3 Permission entries	Outbound rules count 2 Permission entries	

Inbound rules (3)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source
-	sgr-04416b59514b85391	IPv4	SSH	TCP	22	0.0.0.0/0
-	sgr-04da8f2d7510946a2	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-07b5d272a1a56e4a7	IPv4	Custom TCP	TCP	8080	0.0.0.0/0

Dockerfile.

```
main.tf DOCKERFILE X apache-config.conf
DOCKERFILE > ...
1 FROM php:7.4-apache
2
3 #setting arg for Wordpress version and url
4 ARG WORDPRESS_VERSION=5.7
5 ARG WORDPRESS_URL=https://wordpress.org/wordpress-${WORDPRESS_VERSION}.tar.gz
6
7 # packages to install
8 RUN apt-get update && apt-get install -y \
9     zlib1g-dev \
10    libzip-dev \
11    && docker-php-ext-install zip
12
13 ENV APACHE_DOCUMENT_ROOT /var/www/html
14 # Set the working directory to /var/www/html
15 WORKDIR $APACHE_DOCUMENT_ROOT
16
17 # Enable mod_rewrite for WordPress permalinks
18 RUN a2enmod rewrite
19
20 # Download and install WordPress https://wordpress.org/wordpress-${WORDPRESS_VERSION}.tar.gz
21 RUN curl -o wordpress.tar.gz -SL $WORDPRESS_URL && \
22     tar -xzf wordpress.tar.gz --strip-components=1 && \
23     rm wordpress.tar.gz && \
24     chown -R www-data:www-data ${APACHE_DOCUMENT_ROOT}
25
26 # Copy a custom Apache config to enable .htaccess and set AllowOverride
27 COPY apache-config.conf /etc/apache2/sites-available/000-default.conf
28
29 # Expose port 80
30 EXPOSE 80
31
32 # Define the ENTRYPOINT and CMD
33 ENTRYPOINT ["apache2-foreground"]
34 CMD ["-D", "FOREGROUND"]
```

Docker image build and push commands.

```

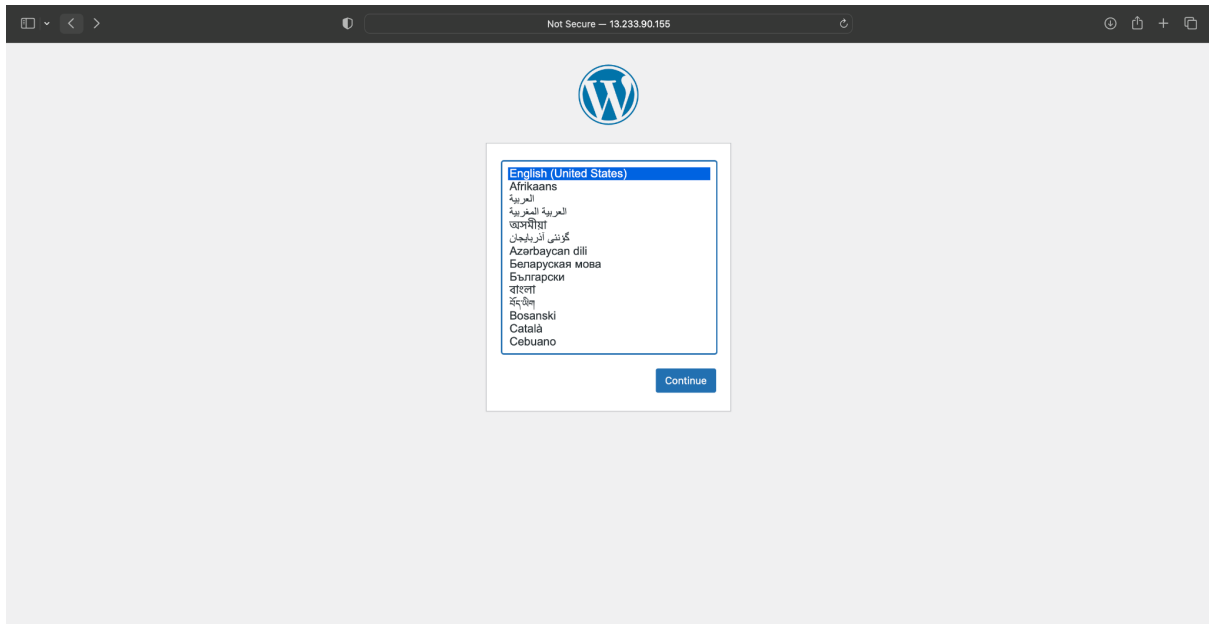
ubuntu@ip-20-0-1-21:~/Docker$ docker build -t diwakarjanna/my-wordpress-image:latest .
[*] Building 1.6s (12/12) FINISHED
[+] Internal load build definition from Dockerfile
=> transferring dockerfile: 1.01kB
[+] Internal load .dockerignore
=> transferring context: 2B
[+] Internal load metadata for docker.io/library/php:7.4-apache
[auth] library/php:pull token for registry-1.docker.io
[+] [1/6] FROM docker.io/library/php:7.4-apache#sha256:c9d7608f73832673479770d66aacc810001lec751d1905ff63fae3fe2a0ca6d
[+] Internal load build context
=> transferring context: 40B
=> CACHED [2/6] RUN apt-get update && apt-get install -y zlib-dev unzip-dev && docker-php-ext-install zip
=> CACHED [3/6] WORKDIR /var/www/html
=> CACHED [4/6] RUN chmod a+rwx /var/www/html
=> CACHED [5/6] RUN curl -O wordpress.tar.gz -SI https://wordpress.org/wordpress-5.7.tar.gz && tar -xzf wordpress.tar.gz --strip-components=1 && rm wordpress.tar.gz && chown -R www-data:www-data /var/w
=> CACHED [6/6] COPY apache-config.conf /etc/apache2/sites-available/000-default.conf
=> exporting to image
=> exporting layers
=> writing image sha256:e40cd63b7a281ea672ad37dd0efa6f17a49424a49bc44829369671c5fa09ce549
=> naming to docker.io/diwakarjanna/my-wordpress-image:latest
=> naming to docker.io/diwakarjanna/my-wordpress-image:latest
ubuntu@ip-20-0-1-21:~/Docker$ docker images
REPOSITORY              TAG          IMAGE ID       CREATED      SIZE
diwakarjanna/my-wordpress-image   latest       e40cd63b7a28   20 minutes ago   525MB
my-wordpress-image             latest       e40cd63b7a28   20 minutes ago   525MB
ubuntu@ip-20-0-1-21:~/Docker$ docker push diwakarjanna/my-wordpress-image:latest
The push refers to repository [docker.io/diwakarjanna/my-wordpress-image]
b3e302e2ef8d: Pushed
e09978ed0910: Pushed
d4b2a1d21b95: Pushed
5f70bf18a086: Pushed
81610e43ab4d: Pushed
3633242bf117: Mounted from library/php
529016336083: Mounted from library/php
5464bcc3f1c2: Mounted from library/php
28192e867e79: Mounted from library/php
d173e76df32e: Mounted from library/php
0e2e8c4f8fd8: Mounted from library/php
30fa0c430434: Mounted from library/php
a538c5a6e4e0: Mounted from library/php
e5840f84dc43: Mounted from library/php
4414e371cc97: Mounted from library/php
797a7c0590e0: Mounted from library/php
f60117696410: Mounted from library/php
ec4a38999118: Mounted from library/php
latest: digest: sha256:b3eb738c0cd0656f658976d8158e1770837508ffa02122d4ee986e5229e1e190 size: 4079
ubuntu@ip-20-0-1-21:~/Docker$

```

Docker container running on the EC2 instance.

```
ubuntu@ip-20-0-1-21:~/wordpress-image$ docker images
REPOSITORY          TAG         IMAGE ID       CREATED        SIZE
my-wordpress-image  latest      e40cd63b7a28  2 minutes ago  525MB
ubuntu@ip-20-0-1-21:~/wordpress-image$ docker run -d -p 8080:80 --name my-wordpress-container my-wordpress-image
b6bd170a7bbc98dc693963278fe27f1bca6fcd19a33d6f796a0900d28d998d
ubuntu@ip-20-0-1-21:~/wordpress-image$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  STATUS    PORTS
b6bd170a7bbc   my-wordpress-image                  "apache2-foreground ..." 5 seconds ago  Up 3 seconds    0.0.0.0:8080->80/tcp, :::8080->80/tcp
my-wordpress-container
```

Wordpress webapplication



RDS setup in the AWS Management Console.

The screenshot displays the AWS Management Console interface for the 'ap-south-1' region. A green notification banner at the top states: 'Successfully created database wordpress-db. You can use settings from wordpress-db to simplify configuration of suggested database add-ons while we finish creating your DB for you. How was your experience creating an Amazon RDS database? Provide feedback'. Below this, the 'Database Details - RDS Management Console' page is shown for the instance 'wordpress-db'. The left sidebar contains navigation links for Amazon RDS services. The main content area shows the 'Summary' tab for the database instance, which is in an 'Available' state. The summary table lists the following details:

DB identifier	CPU	Status	Class
wordpress-db	5.86%	Available	db.t2.micro
Role	Current activity	Engine	Region & AZ
Instance	0 Connections	MySQL Community	ap-south-1a

Below the summary, the 'Connectivity & security' tab is selected, showing details for the endpoint, port, networking, and security settings.

Endpoint & port	Networking	Security
Endpoint wordpress-db.cvgnrwdtuf73.ap-south-1.rds.amazonaws.com	Availability Zone ap-south-1a	VPC security groups allow_ssh_http (sg-0990eb5dd26abd8f8) Active
Port 3306	VPC wordpress-vpc (vpc-03bdf958858c41cd2)	Publicly accessible No
	Subnet group wordpress-subnet-db	Certificate authority info rds-ca-2019
	Subnets subnet-0934e913febcbf2eb subnet-0329c08f3380a5c66 subnet-0f6f35dc3f86493ef	Certificate authority date August 22, 2024, 22:38 (UTC+05:30)
	Network type	DB instance certificate expiration date August 22, 2024, 22:38 (UTC+05:30)

WordPress application successfully connected to the RDS database.

The screenshot shows the WordPress installation screen in a web browser. The WordPress logo is at the top. The main content area displays a welcome message and a list of database configuration items that need to be provided:


1. Database name
2. Database username
3. Database password
4. Database host
5. Table prefix (if you want to run more than one WordPress in a single database)

Below the list, a message states: 'We're going to use this information to create a wp-config.php file. If for any reason this automatic file creation doesn't work, don't worry. All this does is fill in the database information to a configuration file. You may also simply open wp-config-sample.php in a text editor, fill in your information, and save it as wp-config.php. Need more help? We got it.'

At the bottom, a message says: 'In all likelihood, these items were supplied to you by your Web Host. If you don't have this information, then you will need to contact them before you can continue. If you're all ready...'

A 'Let's go!' button is located at the bottom of the configuration box.

Not Secure — 13.233.90.155



Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	<input type="text" value="wordpress"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="username"/>	Your database username.
Password	<input type="password" value="password"/>	Your database password.
Database Host	<input type="text" value="localhost"/>	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

Submit