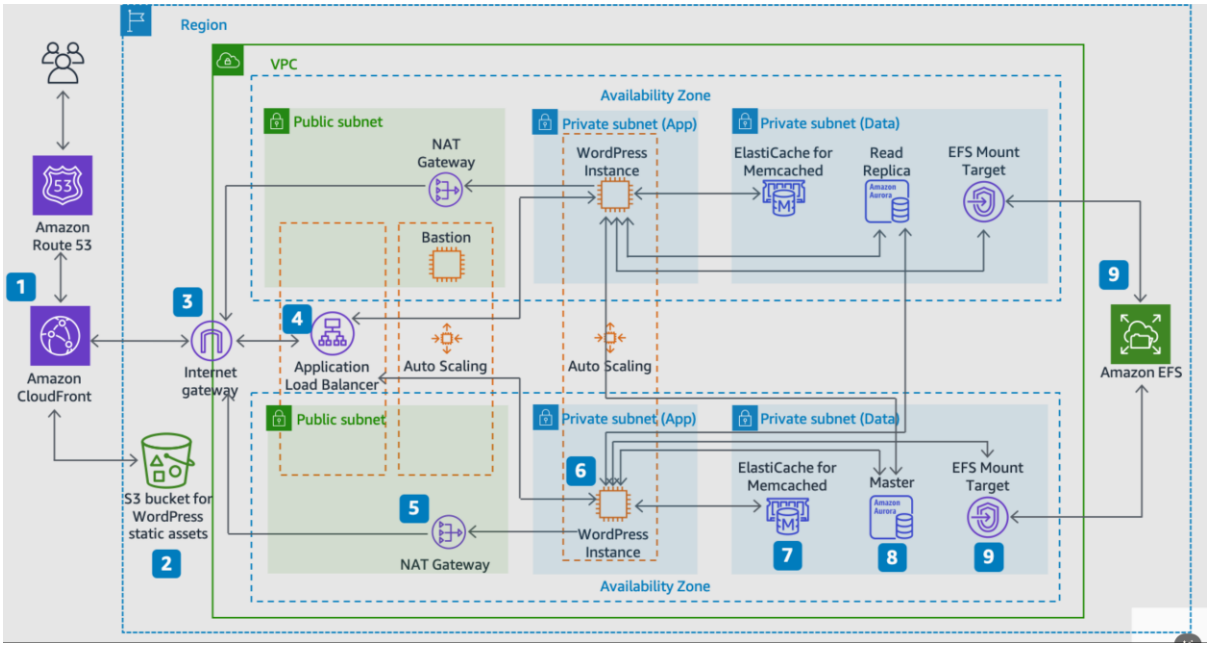
Best Practices for WordPress on AWS

WHAT IS WORDPRESS:

WordPress is an open-source blogging tool and content management system (CMS) based on PHP and MySQL that is used to power anything from personal blogs to high-traffic websites. When the first version of WordPress was released in 2003, it was not built with modern elastic and scalable cloud-based infrastructures in mind. Through the work of the WordPress community and the release of various WordPress modules, the capabilities of this CMS solution are constantly expanding. Today, it is possible to build a WordPress architecture that takes advantage of many of the benefits of the AWS Cloud.



Architecture components

The reference architecture illustrates a complete best practice deployment for a WordPress website on AWS.

- It starts with edge caching in **Amazon CloudFront** (1) to cache content close to end users for fasterdelivery.

- CloudFront pulls static content from an **S3 bucket** (2) and dynamic content from an **Application Load Balancer** (4) in front of the web instances.
- The web instances run in an **Auto Scaling group** of **Amazon EC2instances** (6).
- An **ElastiCache** cluster (7) caches frequently queried data tospeed up responses.

  An **Amazon Aurora** MySQL instance (8) hosts the WordPress database.
- The WordPress EC2 instances access shared WordPress data on an **Amazon EFS** file system via an **EFS Mount Target** (9) in each Availability Zone.
- An **Internet Gateway** (3) enables communication between resources in your VPC and the internet.
- **NAT Gateways** (5) in each Availability Zone enable EC2 instances in private subnets (App and Data) to access the internet.

Within the **Amazon VPC** there exist two types of subnets: public (**Public Subnet**) and private (**App Subnet** and **Data Subnet**). Resources deployed into the public subnets will receive a public IP address and will be publicly visible on the internet. The **Application Load Balancer** (4) and a Bastion host for administration are deployed here. Resources deployed into the private subnets receive only a private IP address and hence are not publicly visible on the internet, improving the security of those resources. The **WordPress web server instances** (6), **ElastiCache cluster instances** (7), **Aurora MySQL database instances** (8), and **EFS Mount Targets** (9) are all deployed in private subnets.

Simple deployment : For low-traffic blogs or websites without strict high availability requirements, a simple deployment of a single server might be suitable. This deployment isn't the most resilient or scalable architecture, but it is the quickest and most economical way to get your website up and running.

Available approaches

AWS has a number of different options for provisioning virtual machines. There are three main ways to host your own WordPress website on AWS:

• Amazon Lightsail

• Amazon Elastic Compute Cloud (Amazon EC2)

• AWS Marketplace

USE PUTTY for connectivity of your instance to your system

Commands used while deploying the architecture:

1. Install Apache server on Ubuntu

sudo apt install apache2

2. Install php runtime and php mysql connector

sudo apt install php libapache2-mod-php php-mysql

3. Install MySQL server

sudo apt install mysql-server

4. Login to MySQL server

sudo mysql -u root

5. Change authentication plugin to mysql_native_password (change the password to something strong)

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by 'Testpassword@123';

6. Create a new database user for wordpress (change the password to something strong)

CREATE USER 'wp_user'@localhost IDENTIFIED BY 'Testpassword@123';

7. Create a database for wordpress

CREATE DATABASE wp;

8. Grant all privilges on the database 'wp' to the newly created user

GRANT ALL PRIVILEGES ON wp.* TO 'wp_user'@localhost;

9. Download wordpress

cd /tmp

wget https://wordpress.org/latest.tar.gz

10. Unzip

tar -xvf latest.tar.gz

11. Move wordpress folder to apache document root

sudo mv wordpress/ /var/www/html

12. Command to restart/reload apache server

sudo systemctl restart apache2

OR

sudo systemctl reload apache2

13. Install certbot

sudo apt-get update

sudo apt install certbot python3-certbot-apache

14. Request and install ssl on your site with certbot

sudo certbot –apache