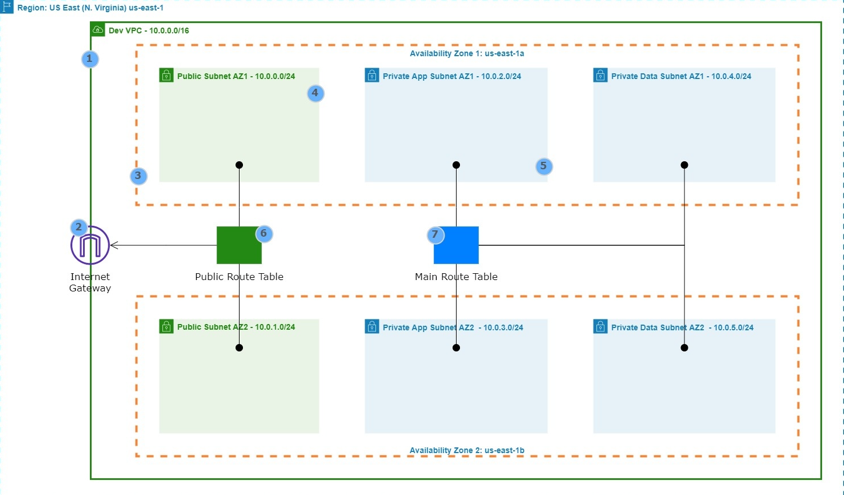
Deploy a WordPress Website on AWS

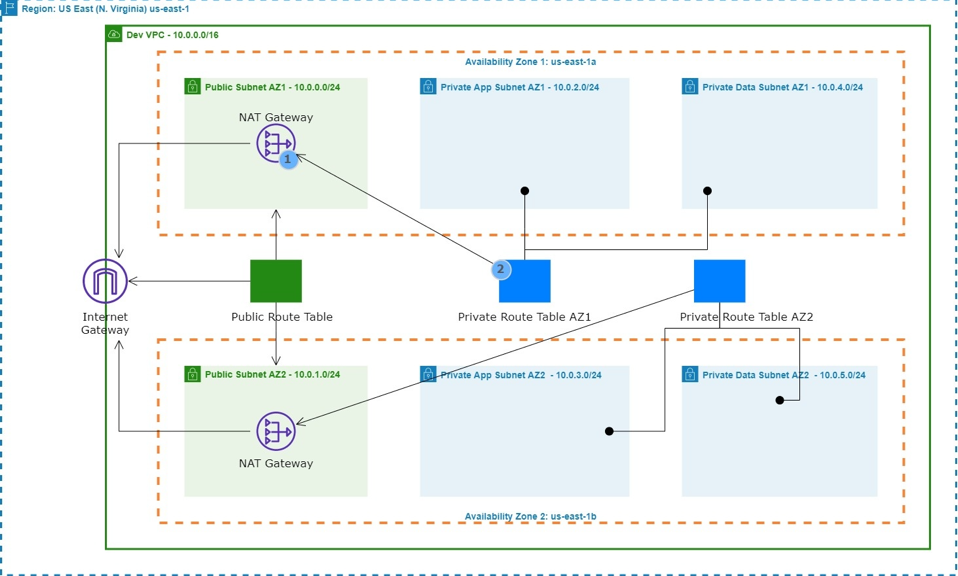
I recently finished a DevOps project where I hosted an HTML website on AWS, utilizing the resources listed below. I have uploaded the reference diagram and scripts I used to deploy the website on an EC2 instance to a GitHub repository for the project. Please use the information to create a readme file for the project

1. Start by building a three-tier AWS VPC with public and private subnets in 2 availability zones for high availability and fault tolerance.
   1. Tier 1 contains the public subnet which holds resources such as Nat Gateway, Load Balancer and Bastion Host
   2. Tier 2 contains the private subnet which holds the webservers (EC2 instances)
   3. Tier 3 contains another private subnet which holds the database.
2. Create first VPC: Dev VPC
3. Enable DNS Hostnames in the VPC:
4. Create Internet Gateway to allow communication between instances in the VPC and the internet.
5. Create public subnets in 2 Availability Zones. Then enable Auto Assign IP settings to have any EC2 instance starting within the public subnet associated with an IPv4 address.
6. Create a **public** route table. One thing to note, a main **private** route table is created by default when creating a VPC.
   1. Add public route to the table with destination 0.0.0.0/0 (anywhere on the internet) with the Internet Gateway created earlier as the Target
   2. Associate to the two public subnets with the table.
7. Create 4 **private** subnets. 2 subnets for the webservers and 2 subnets for the database.
   1. Subnets will be associated with the main private route table created with the VPC by default. The private route table only routes traffic locally in the VPC
8. Reference Architecture:

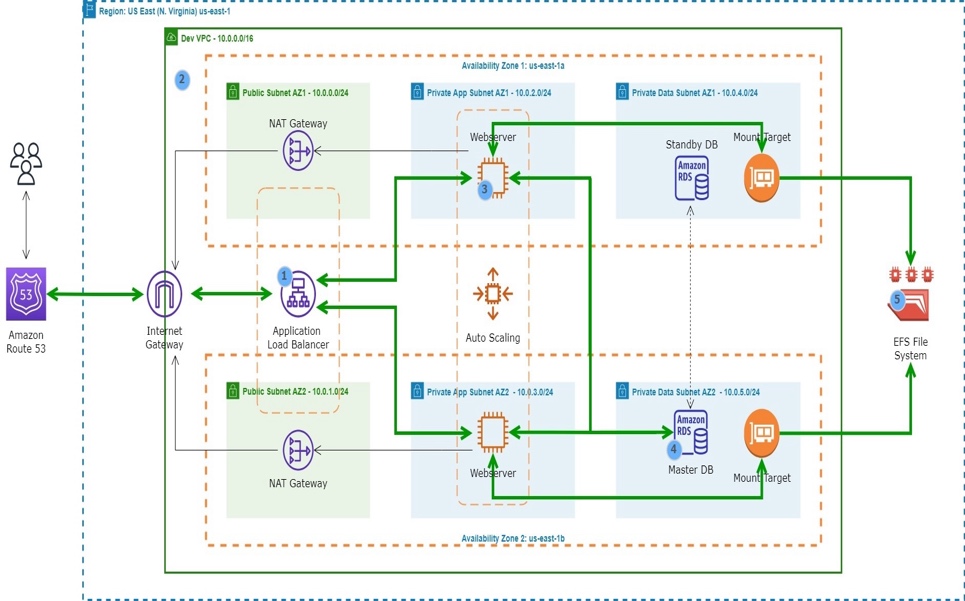


* 1. Create a NAT Gateway in Public AZ1 and AZ2. This allows instances in the private App subnets and Data subnets to access the internet.
  2. Make sure you’re in the correct region you created the VPC
  3. Must allocate an elastic IP to the NAT gateway.

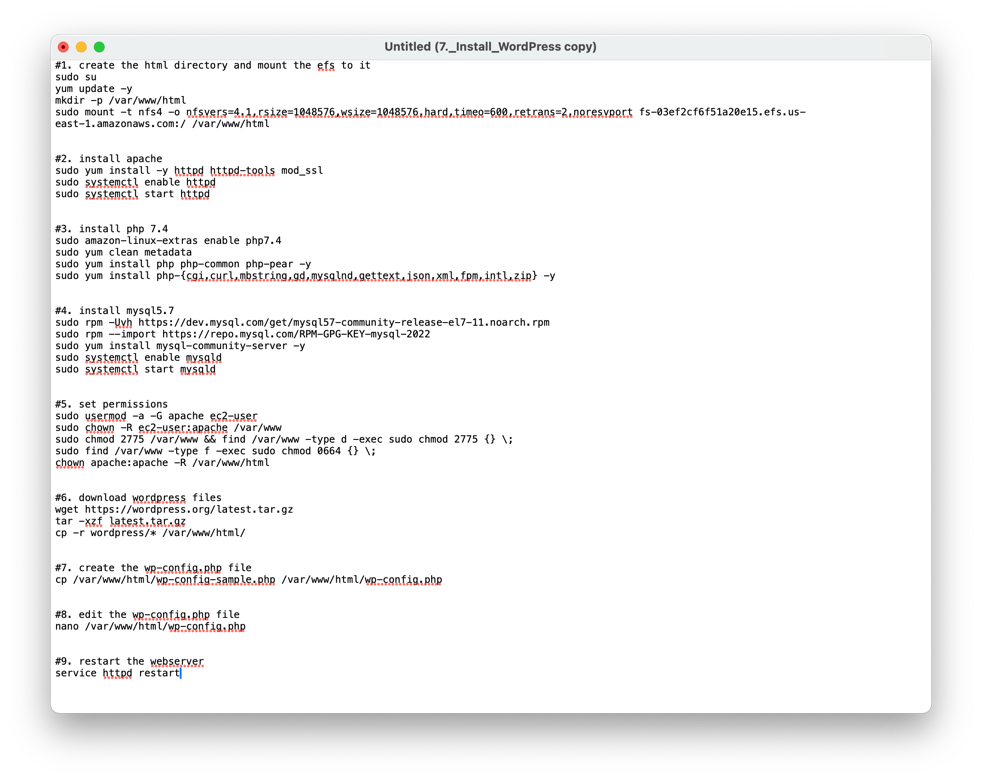
1. Create private route table AZ1. Associated with the private subnets and routes traffic to the internet through the NAT Gateway
   1. Create a **private** route table AZ1 to route traffic to the internet through the NAT Gateway in the **public** subnet AZ1
   2. Choose destination 0.0.0.0/0 and the NAT Gateway AZ1 (in the public subnet) created earlier as the target.
   3. Must allocate an elastic IP to the NAT gateway.
   4. Associate table with private app subnet AZ1 and private data subnet AZ1
2. Create NAT Gateway in public subnet AZ2. Associated with the private subnets and routes traffic to the internet through the NAT Gateway
   1. Create a route **private** route table AZ2 to route traffic to the internet through the NAT Gateway in the **public** subnet AZ2
   2. Choose destination 0.0.0.0/0 and the NAT Gateway AZ2 created earlier as the target.
   3. Must allocate an elastic IP to the NAT gateway.
   4. Associate table with private app subnet AZ2 and private data subnet AZ2
3. Reference Architecture:



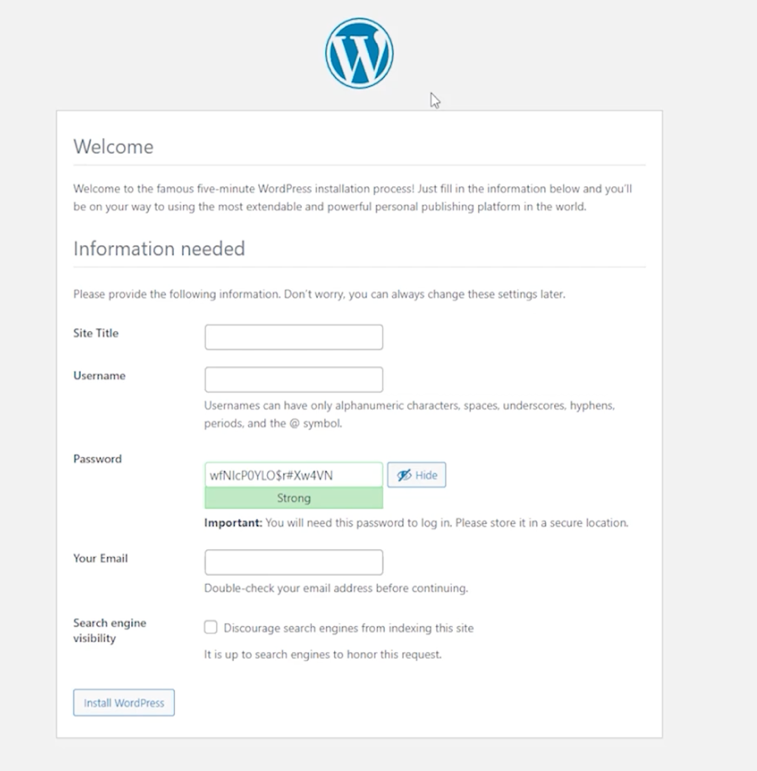
1. Create Setup Security Groups that act as a firewall for the instances (Inbound Rules)
   1. Application Load Balancer SG open ports 80 and 443 and source = anywhere (0.0.0.0/0)
   2. SSH Security Group open port 22 and source = Your IP Address
   3. Webserver Security Group **(EC2 Instances on the Private App Subnets)** open ports 80, 443, and 22 with ALB and SSH SG as the source. This will ONLY allow traffic on 80 and 443 if it comes from the Load Balancer. This will ONLY allow traffic on 22 if it is coming from the SSH Security Group.
   4. Database Security Group opens port 3306 (MySQL/Aurora) with source = Webserver SG.
   5. EFS Security Group opens port 2049 (NFS) and 22 with source = Webserver SG and SSH SG. And another rule for NFS and limit the source to the EFS SG
2. Reference Architecture:



1. Create RDS Instance in private data subnets
   1. First create subnet groups. This allows us to specify which subnets we want to create our RDS Database in.
   2. Select databases
      1. Standard Create
      2. My SQL
      3. Select the proper version
      4. Put in credentials
      5. Select instance configuration
      6. Select VPC
      7. Make sure correct subnets are selected
      8. Selected existing database security group
      9. Select AZ where you want the master DB (east-1b)
      10. Make sure password authentication is selected
      11. Under additional configuration make sure to create a database name
2. After database has been created, select and look under connectivity and security for the Endpoint. Will need this info to connect database to our EC2 instance.
3. Create Elastic File System with mount targets in each AZ in the VPC. The webservers will use the mount targets to connect to the EFS.
   1. Select File Systems under EFS
   2. Create then select customize
   3. Create a name
   4. If you encrypt data, you will be charged.
   5. Select Dev VPC
   6. Select the private data subnets (AZ1 and AZ2) and EFS SG
   7. File System policy leave as default
   8. Select attach to see the information to mount the system.
4. Create an EC2 instance in Public Subnet AZ1 in the Dev VPC. Choose SSH, ALB, and Webserver security groups.
5. SSH into Instance
6. Install WordPress
   1. Here are the scripts used: First have to update mount information of EFS



1. Finish setting up wordpress site through public IPv4 link. Entire any site title name, and entire admin name in the username section. After entering email, install wordpress, then login.



1. Create ALB to route traffic to the EC2 instance in the private subnet.
   1. Launce first instance on private app subnet AZ1 and AZ2 using the below script in the user data section.(we
      1. #!/bin/bash
      2. yum update -y
      3. sudo yum install -y httpd httpd-tools mod\_ssl
      4. sudo systemctl enable httpd
      5. sudo systemctl start httpd
      6. sudo amazon-linux-extras enable php7.4
      7. sudo yum clean metadata
      8. sudo yum install php php-common php-pear -y
      9. sudo yum install php-{cgi,curl,mbstring,gd,mysqlnd,gettext,json,xml,fpm,intl,zip} -y
      10. sudo rpm -Uvh https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm
      11. sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
      12. sudo yum install mysql-community-server -y
      13. sudo systemctl enable mysqld
      14. sudo systemctl start mysqld
      15. echo "fs-03c9b3354880b36a6.efs.us-east-1.amazonaws.com:/ /var/www/html nfs4 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 0 0" >> /etc/fstab
      16. mount -a
      17. chown apache:apache -R /var/www/html
      18. sudo service httpd restart
   2. Select Webserver Security Group
   3. Create target group (Dev-TG) for the previous 2 created instances to allow ALB to route traffic to it.
      1. Under success codes, type 200, 301, 302
   4. Now create Application Load Balancer
      1. Name is Dev-ALB
      2. Internet facing, IPv4
      3. Select Dev VPC
      4. Under mappings select Public Subnet AZ1 and AZ2
      5. Select ALB Security Group
      6. Select Dev-TG for the Listener on port 80
   5. You can now access website with DNS name of ALB
2. Whenever you change the domain address, you have to go into WordPress settings and change DNS address there as well.
3. Since we have setup the 2 instances and the ALB, we can now terminate the setup server.
4. Register domain name in Route 53
5. Register for an SSL Certificate in AWS Certificate Manager
6. Launch Bastion Host to SSH into the EC2 instance in the private subnet.
   1. First launch EC2 instance in public subnet. SSH into the public subnet, then SSH into the private subnet thru the Bastion host. Use the SSH Security Group.
7. To ssh into an instance in the private subnet using a mac computer, run the following commands from the directory you saved your private key pair.

Please note that the following commands assume you are using an Amazon Linux instance.

Command 1:

ssh-add --apple-use-keychain <the-name-of-your-private-key.pem>

Example

ssh-add --apple-use-keychain myec2key.pem

Command 2:

ssh -A ec2-user@<the-public-ipv4-ip-of-your-bastion-host>

Example:

ssh -A ec2-user@54.162.137.241

Command 3:

ssh ec2-user@<the-private-ipv4-ip-of-the-instance-in-the-private-subnet>

Example:

ssh [ec2-user@18.232.135.220](mailto:ec2-user@18.232.135.220)

1. Create an HTTPS Listener for the ALB previously created
   1. Under listeners tab, add listener HTTPS port 443
   2. Select forward to Target Group created earlier
   3. Then select the SSL certificate
   4. Now edit HTTP listener to redirect traffic to the HTTPS 443
2. Now SSH into one of your private subnet instances to modify WPconfig file. Use script

/\* SSL Settings \*/

define('FORCE\_SSL\_ADMIN', true);

// Get true SSL status from AWS load balancer

if(isset($\_SERVER['HTTP\_X\_FORWARDED\_PROTO']) && $\_SERVER['HTTP\_X\_FORWARDED\_PROTO'] === 'https') {

$\_SERVER['HTTPS'] = '1';

}

1. Create Auto Scaling Group
   1. Before that can be done, the EC2 instances running must be terminated.
   2. Then a launch template must be created. This contains configuration of our EC2 instances the ASG will use to launch new instances in the Private App Subnet.
   3. Select Auto Scaling Guidance
   4. Select Amazon Linux AMI
   5. Select Instance type
   6. Select Keypair
   7. Select Webserver Security Group
   8. Under advanced details, paste the bash script in the “user data” section.
   9. When creating the ASG, select the launch template created earlier and the corresponding VPC.
   10. Select Private Subnet AZ1 and Private Subnet AZ2
   11. Attach the Dev-TG previously created.
   12. Check all options under health checks (EC2 and ELB) and enable group metrics within CloudWatch
   13. Desired Capacity to 2, Min Capacity to 1, Max Capacity to 4
   14. Add notification and select an SNS topic