

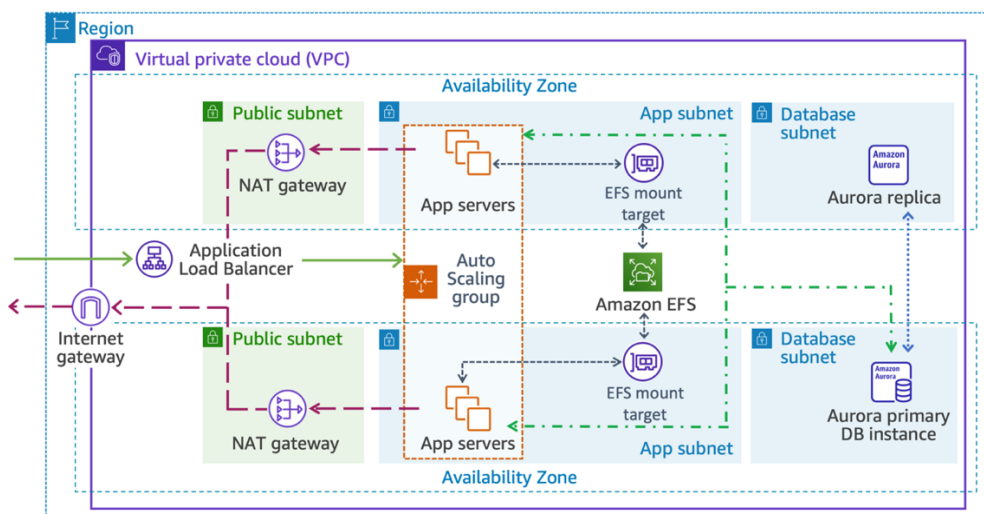
# AWS Capstone Project

Miles Mathiason

## Introduction

In this project I was tasked with applying my AWS solution architecture knowledge to create and configure a cloud environment, utilizing various AWS resources to solve a specific business scenario. This project consisted of configuring a multi- Availability Zone network environment consisting of six subnets spanning two Availability Zones. And, deploying and configuring resources including from Amazon EC2, Amazon EFS, AWS CloudFormation, Amazon RDS, and Amazon ELB. Ultimately, the goal of this project was to architect a cloud environment capable of hosting a client-facing application which customers would use to access their accounts, create marketing plans, and run data analysis on their marketing campaigns. The solution had to be durable, scalable, fast and more cost-effective than their existing on-premises infrastructure.

Below is the design of the cloud environment I aimed to deploy and configure in this project.



## Reviewing and running a preconfigured CloudFormation template

In order to deploy a highly available and scalable deployment of the WordPress application stack, I used a pre-made CloudFormation template to deploy the networking resources required to support the application. This template consisted of the CIDR range for the Amazon Virtual Private Cloud with two public subnets, two private application subnets, and two private database subnets. Below are the steps I took to deploy this template.

- First, I opened and reviewed the pre-made IaC CloudFormation template in Visual Studios.

```

! Task1.yaml
Users > milesmathiason > Downloads > ! Task1.yaml
1 AWSTemplateFormatVersion: 2010-09-09
2
3 Description: Lab7 Task 1 template which builds VPC, supporting resources, a basic networking structure, and some Security groups for use in later tasks.
4
5 Parameters:
6   VPCCIDR:
7     Description: CIDR Block for VPC
8     Type: String
9     Default: 10.0.0.0/16
10    AllowedValues:
11      - 10.0.0.0/16
12
13   PublicSubnet1Param:
14     Description: Public Subnet 1
15     Type: String
16     Default: 10.0.0.0/24
17     AllowedValues:
18       - 10.0.0.0/24
19
20   PublicSubnet2Param:
21     Description: Public Subnet 2
22     Type: String
23     Default: 10.0.1.0/24
24     AllowedValues:
25       - 10.0.1.0/24
26
27   AppSubnet1Param:
28     Description: App Subnet 1
29     Type: String
30     Default: 10.0.2.0/24
31     AllowedValues:
32       - 10.0.2.0/24
33
34   AppSubnet2Param:
35     Description: App Subnet 2

```

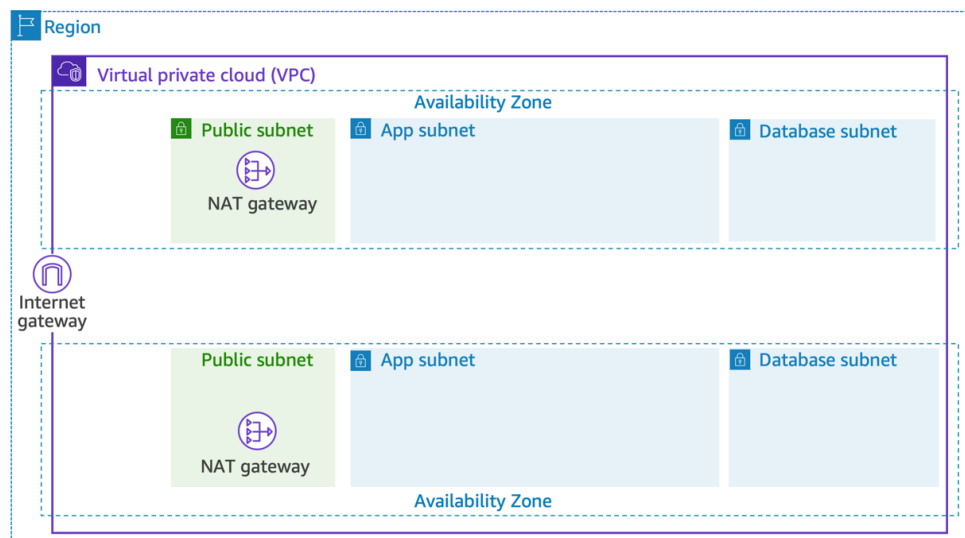
- Then, I uploaded the template through the CloudFormation console which included assigning resource identifiers for all new resources and naming the stack.
- Here I ran into an error creating the change set as my new resources didn't have a DeletionPolicy attribute specified in the template.



#### There was an error creating this change set

The following resources to import [DatabaseSubnet1, PublicRouteTable, RDSSecurityGroup, DatabaseSubnet2, AttachGateway, DatabaseSubnet1RouteTableAssociation, LabVPC, AppSubnet2, AppSubnet1, PublicRoute, AppInstanceSecurityGroup, PublicSubnet2, EFSMountTargetSecurityGroup, AppSubnet2RouteTableAssociation, PrivateRouteTableAZ1, PrivateRouteTableAZ2, PublicSubnet1, LabInternetGateway, PrivateRouteAZ1, PrivateRouteAZ2, AppSubnet1RouteTableAssociation, DatabaseSubnet2RouteTableAssociation, PublicSubnet1RouteTableAssociation, ElasticIPAddress2, PublicSubnet2RouteTableAssociation, ElasticIPAddress1, NATGateway2, NATGateway1] must have DeletionPolicy attribute specified in the template.

- To resolve this issue I had to back track and change my stack creation preferences. I had to specify I was creating a stack with new resources which I had missed in my first attempt.
- Now that my network infrastructure is up and running I checked out all my new resources from the AWS console.



## Creating an Amazon RDS database

For the next step in this project I created and deployed an Amazon Aurora database using Amazon RDS. I also created a subnet group to support the database instances.

First, I created a DB subnet group inside the VPC I deployed previously through the RDS console. I had to configure which Availability Zones this multi-AZ database would span across and assign CIDR blocks to each AZ. I assigned the CIDR blocks 10.0.4.0/24 and 10.0.5.0/24 to the Availability Zones eu-west-2a and eu-west-2b respectively

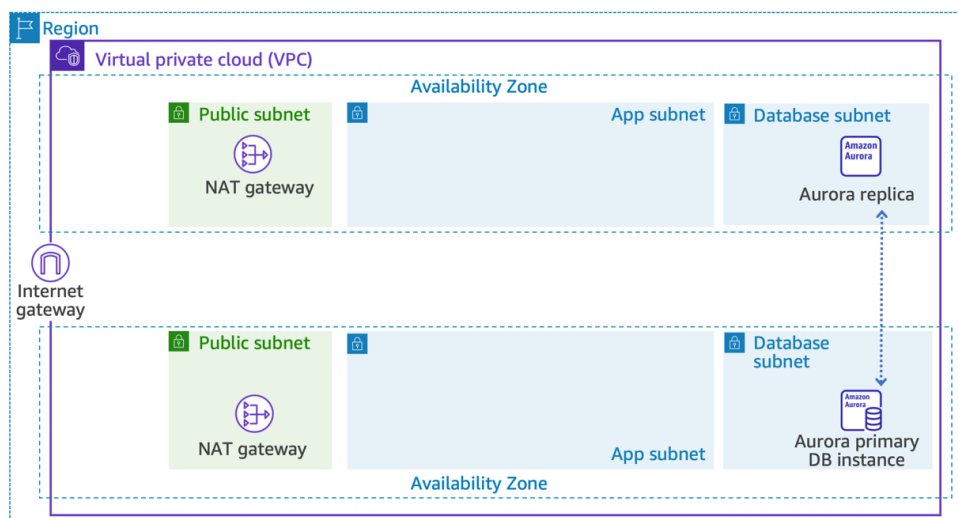
db\_subnet\_group

Subnet group details		
VPC ID	vpc-0fe00d80a87ed24be	
ARN	arn:aws:rds:eu-west-2:086600671338:subgrp:db_subnet_group	
Supported network types	IPv4	
Description	Database subnet group	

Subnets (2)		
Availability zone	Subnet ID	CIDR block
eu-west-2a	subnet-03b9c35fef5eaf4e5	10.0.4.0/24
eu-west-2b	subnet-0c895dad9ba930a76	10.0.5.0/24

Next, I created my database instances. For this I took care to configure according to the requirements of the client. This involved creating an Aurora DB which is compatible with MySQL and is a burstable-performance DB instance. I created this Aurora DB with instance size db.t3.medium, and Encryption, Enhanced monitoring, Enable auto minor version upgrade, Enable deletion protection all turned off as requested by the client. I also configured the networking and security which involved deploying the database into the VPC I deployed previously as well as configuring the subnet group with database subnets and configuring with the RDS security group set up by the clients security team.



## Creating an Amazon EFS file system

In order to speed up the clients ability to onboard new customers and expand the business, EFS was chosen as the storage solution for this project. The client needed to be able to confirm that the backup policies and encryption settings meet their internal and regulatory compliance requirements.

When creating this EFS file system I had to enter the customisations page. In the page I configured this file system to match all of the clients' requirements. For example, regional availability & durability, general purpose performance (to control costs), bursting throughput (to scale the system size), turning off automatic backups & encryption of data at rest, deploying in the VPC deployed previously, mounting to availability zones eu-west-2a and eu-west-2b, assigning AppSubnet1 & AppSubnet2 as subnet IDs, assigning the Security Group configured by the client's security team to the EFS Mount Target. After creating this file system, it was important to note down the File system ID for use later.

### Review and create

#### Step 1: File system settings

[Edit](#)

##### File system

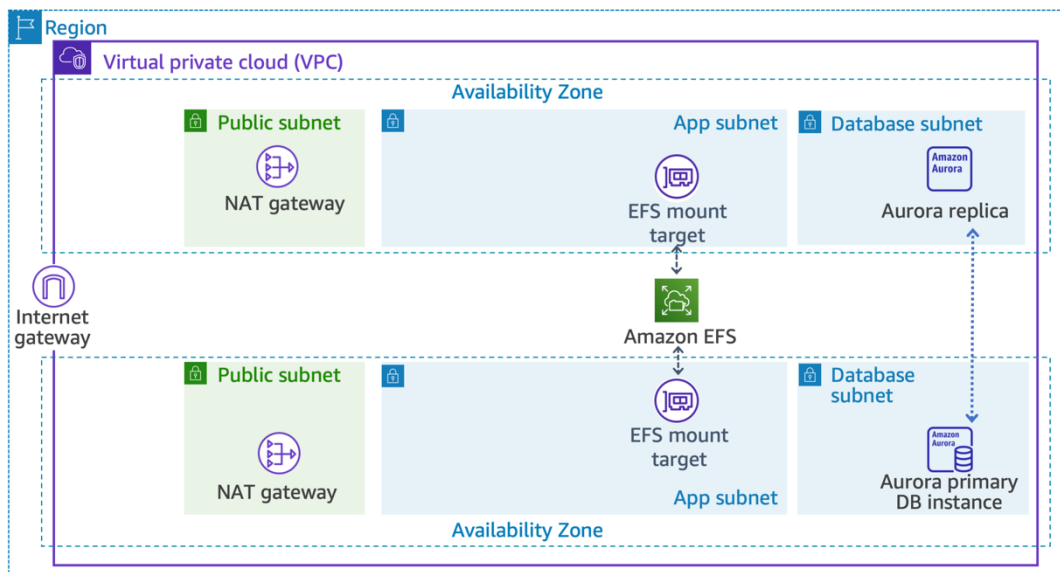
Field	Value	Is editable?
Name	Lab_FS	Yes
Performance mode	General Purpose	No
Throughput mode	Bursting	Yes
Encrypted	No	No
KMS Key ID	-	No
Lifecycle management	Transition into Infrequent Access (IA): 30 day(s) since last access Transition into Archive: None Transition into Standard: None	Yes
Automatic backups	No	Yes
VPC ID	vpc-0fe00d80a87ed24be (LabVPC)	Yes
Availability Zone	Regional	No

#### Step 2: Network access

[Edit](#)

##### Mount targets

Availability zone	Subnet	IP address	Security groups
eu-west-2a	subnet-0cef19ddd0626f042	-	sg-016c99b1c240d5b6f
eu-west-2b	subnet-0cb687430fef4c7f7	-	sg-016c99b1c240d5b6f



## Creating an Application Load Balancer

In order to confront frequent outages caused by varying and unexpected traffic loads, experienced by the clients current on-premises set up, I set up an Application Load Balancer for the application servers in the private subnet with a health check.

To create the Application Load balancer, I first had to create and configure a target group. I configured the target group as shown below. This involved configuring health check settings in a way that suited the client. For example, I chose the Healthy threshold, Unhealthy threshold, and the Timeout & Interval times.

Details					
arn:aws:elasticloadbalancing:eu-west-2:086600671338:targetgroup/myWPTargetGroup/f13ed0f781faacfb					
Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC <a href="#">vpc-0fe00d80a87ed24be</a>		
IP address type IPv4	Load balancer <a href="#">None associated</a>				
0 Total targets	<div> <div>✓ 0</div> <div>Healthy</div> </div> <div>0</div> <div>Anomalous</div>	<div> <div>✗ 0</div> <div>Unhealthy</div> </div>	<div> <div>⋯ 0</div> <div>Unused</div> </div>	<div> <div>⌚ 0</div> <div>Initial</div> </div>	<div> <div>⌚ 0</div> <div>Draining</div> </div>

Now, for the Application Load balancer, I had to configure with the VPC deployed in the CloudFormation template and in the public subnets. This involved Selecting all Public subnets, choosing a previously configured Security Group, and configure a Listener to connect with the target group. At this point it was important to note down the Load Balancers DNS name for later use.

myWPAppALB

Actions

▼ Details

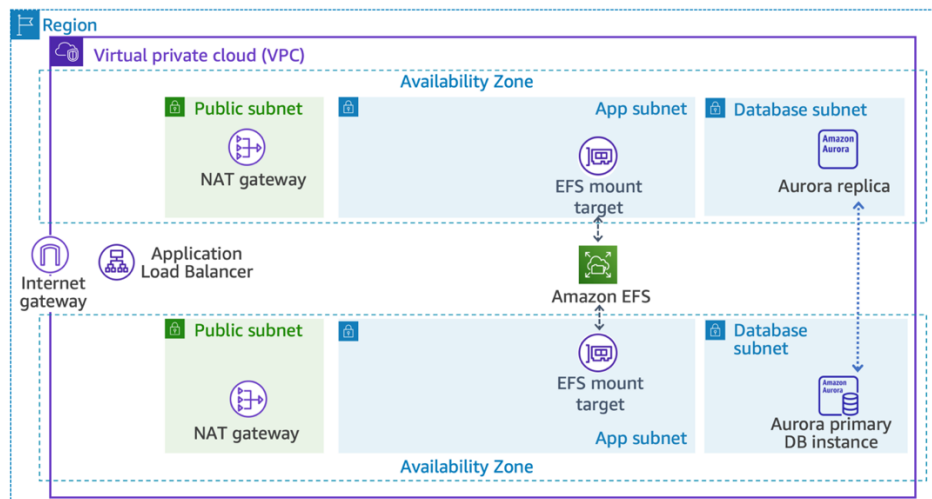
Load balancer type	Status	VPC	IP address type
Application	⏸ Provisioning	<a href="#">vpc-0fe00d80a87ed24be</a>	IPv4
Scheme	Hosted zone	<a href="#">subnet-018d49ebe6ca3a703</a>	Date created
Internet-facing	ZHURV8PSTC4K8	<a href="#">eu-west-2b (euw2-az3)</a>	February 29, 2024, 17:12 (UTC+00:00)
		<a href="#">subnet-0022a6dfe0e0db8f3</a>	
		<a href="#">eu-west-2a (euw2-az2)</a>	

Load balancer ARN

arn:aws:elasticloadbalancing:eu-west-2:086600671338:loadbalancer/app/myWPAppALB/dbfe925132be197c

DNS name [Info](#)

myWPAppALB-1089303226.eu-west-2.elb.amazonaws.com (A Record)



## Creating a launch template using CloudFormation

To combine all the network resources, database, EFS file system, and the load balancer I deployed a pre-configured CloudFormation template.






First, I opened the template in Visual Studios to review it. Then, during the stack creation process I updated parameter values with values noted from previous deployments. For example, the database name, database endpoint, database username & password, etc.

```

Users > milesmathiason > Downloads > / Task5.yaml
1  AWS::CloudFormation::Interface
2
3  Description: Stack to create a launch configuration for wordpress EC2 servers.
4
5  Metadata:
6  AWS::CloudFormation::Interface:
7  ParameterGroups:
8  - Labels:
9    default: Database Parameters
10     Parameters:
11     - DatabaseName
12     - DatabaseHostName
13     - DatabaseUsername
14     - DatabasePassword
15  - Labels:
16    default: WordPress Parameters
17     Parameters:
18     - Username
19     - Password
20     - Email
21  - Labels:
22    default: Other Parameters
23     Parameters:
24     - EC2ServerInstanceType
25  ParameterLabels:
26  DatabaseName:
27    default: DB name
28  DatabaseHostName:
29    default: Database endpoint
30  DatabaseUsername:
31    default: Database User Name
32  DatabasePassword:
33    default: Database Password
34  EC2ServerInstanceType:
35    default: Instance Type
36  Username:

```

During this process I ran into an issue. I forgot to note down the databases master password when creating it. To solve this I had to open the database modify settings of the Aurora DB instances to change the master password. This solved my problem and the template was successfully deployed.

Overview 	
Stack ID  <a href="#">arn:aws:cloudformation:eu-west-2:086600671338:stack/WPLaunchConfigStack/198cc170-d729-11ee-a110-064a9f8342cb</a> 	Description Stack to create a launch configuration for wordpress EC2 servers.
Status  <b>CREATE_IN_PROGRESS</b>	Status reason -
Root stack -	Parent stack -
Created time 2024-02-29 17:36:41 UTC+0000	Deleted time -
Updated time -	
Drift status  <b>NOT_CHECKED</b>	Last drift check time -
Termination protection Deactivated	IAM role -


## Creating the application servers by configuring an Auto Scaling group and a scaling policy

The last step was to create the application servers by configuring an Auto Scaling group and scaling policy. To do this I created a scaling group with the launch template from the previous step. I configured with the VPC and the two app subnets. I then had to configure

some advanced options such as choosing from load balancer target groups, attaching to existing load balancer, turning on Elastic Load Balancing health check and enabling group metric collection with CloudWatch. Then, I configured the group size and scaling policies as follows: Desired capacity = 2, Minimum capacity = 2, and Maximum capacity = 4.

Group details

Edit

Auto Scaling group name	Desired capacity	Desired capacity type	Amazon Resource Name (ARN)
WP-ASG	2	Units (number of instances)	 arn:aws:autoscaling:eu-west-2:086600671338:autoScalingGroup:384e7237-1277-4714-a0e3-04943681b8b0:autoScalingGroupName/WP-ASG
Date created	Minimum capacity	Status	
Thu Feb 29 2024 17:45:34 GMT+0000 (Greenwich Mean Time)	2	-	
	Maximum capacity		
	4		

Finally, I checked my WordPress application was successfully launched by visiting the URL below. Below is a screen shot of the website I launched.

<http://mywpappalb-1089303226.eu-west-2.elb.amazonaws.com/wp-login.php>

WordPress on AWS | 0 | New | Howdy, wpadmin

Dashboard | Screen Options | Help

# Welcome to WordPress!

[Learn more about the 6.4.3 version.](#)

Author rich content with blocks and patterns

Block patterns are pre-configured block layouts. Use them to get inspired or create new pages in a flash.

[Add a new page](#)

Customize your entire site with block themes

Design everything on your site — from the header down to the footer, all using blocks and patterns.

[Open site editor](#)

Switch up your site's look & feel with Styles

Tweak your site, or give it a whole new look! Get creative — how about a new color palette or font?

[Edit styles](#)

Site Health Status

No information yet...

Site health checks will automatically run periodically to gather information about your site. You can also [visit the Site Health screen](#) to gather information about your site

Quick Draft

Title

mywpappalb-1089303226.eu-west-2.elb.amazonaws.com/wp-admin/index.php