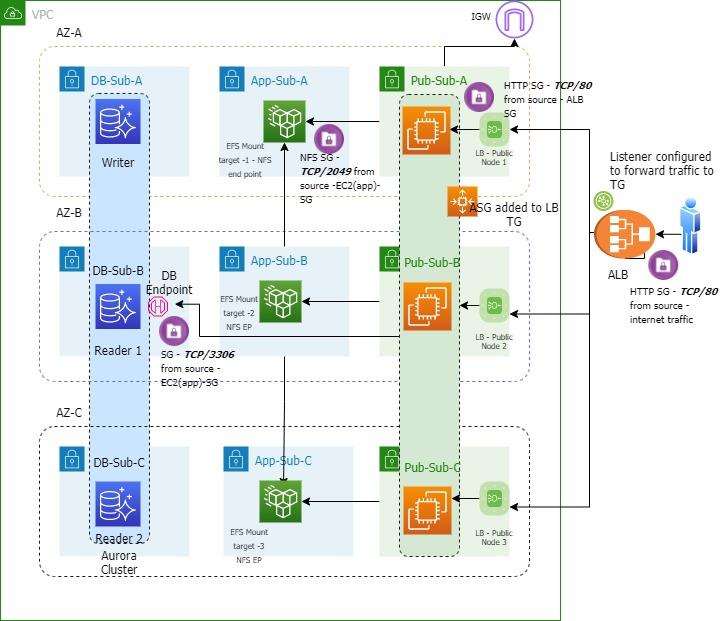
WordPress application deployed in HA, Scalable, Self Healing environment:



1. Created **VPC** in us-east-1 region with 3 Azs , each containing 2 private subnets (DB , Application) and 1 public subnet
2. Public subnets of each Azs, are associated with Network Route table which is configured to direct internet(0.0.0.0/0) bound traffic to **Internet gateway**
3. Created 4 SG upfront as depicted in the diagram to allow traffic in.

* DB SG → Allows traffic from word press application connect to DB endpoint at port 3306
* EFS SG → Allows traffic from WordPress application connect to EFS mount target(NFS endpoint) at port 2049
* WordPress(EC2) SG → Allows traffic from LBSG connect to EC2 instance at port 80
* LBSG → Allows traffic from internet connect to Load balancer

1. Created **SSM Parameter** store to store DB password, DB Name,

DB endpoint, EFS ID, DB user and Load balancer endpoint

1. Created regional **Elastic File System (Managed NFS)** with mount target across 3 Availability zones in Application Subnets (App-subnet-A, App-subnet-B, App-subnet-C). EFS will store images that are uploaded into WordPress application while post,comments are to be inserted into Aurora DB.
2. Created **Launch template** to provision EC2 instances to host WordPress application , across 3 AZs in Public subnet with userdata as mentioned below,

#!/bin/bash -xe

ALBDNSNAME=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/ALBDNSNAME --query Parameters[0].Value)

ALBDNSNAME=`echo $ALBDNSNAME | sed -e 's/^"//' -e 's/"$//'`

EFSFSID=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/EFSFSID --query Parameters[0].Value)

EFSFSID=`echo $EFSFSID | sed -e 's/^"//' -e 's/"$//'`

DBPassword=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/DBPassword --with-decryption --query Parameters[0].Value)

DBPassword=`echo $DBPassword | sed -e 's/^"//' -e 's/"$//'`

DBRootPassword=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/DBRootPassword --with-decryption --query Parameters[0].Value)

DBRootPassword=`echo $DBRootPassword | sed -e 's/^"//' -e 's/"$//'`

DBUser=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/DBUser --query Parameters[0].Value)

DBUser=`echo $DBUser | sed -e 's/^"//' -e 's/"$//'`

DBName=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/DBName --query Parameters[0].Value)

DBName=`echo $DBName | sed -e 's/^"//' -e 's/"$//'`

DBEndpoint=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/DBEndpoint --query Parameters[0].Value)

DBEndpoint=`echo $DBEndpoint | sed -e 's/^"//' -e 's/"$//'`

yum -y update

yum -y upgrade

yum install -y httpd wget amazon-efs-utils

yum install stress -y

systemctl enable httpd

systemctl start httpd

mkdir -p /var/www/html/wp-content

chown -R ec2-user:apache /var/www/

echo -e "$EFSFSID:/ /var/www/html/wp-content efs \_netdev,tls,iam 0 0" >> /etc/fstab

mount -a -t efs defaults

wget http://wordpress.org/latest.tar.gz -P /var/www/html

cd /var/www/html

tar -zxvf latest.tar.gz

cp -rvf wordpress/\* .

rm -R wordpress

rm latest.tar.gz

sudo cp ./wp-config-sample.php ./wp-config.php

sed -i "s/'database\_name\_here'/'$DBName'/g" wp-config.php

sed -i "s/'username\_here'/'$DBUser'/g" wp-config.php

sed -i "s/'password\_here'/'$DBPassword'/g" wp-config.php

sed -i "s/'localhost'/'$DBEndpoint'/g" wp-config.php

usermod -a -G apache ec2-user

chown -R ec2-user:apache /var/www

chmod 2775 /var/www

find /var/www -type d -exec chmod 2775 {} \;

find /var/www -type f -exec chmod 0664 {} \;

cat >> /home/ec2-user/update\_wp\_ip.sh<< 'EOF'

#!/bin/bash

source <(php -r 'require("/var/www/html/wp-config.php"); echo("DB\_NAME=".DB\_NAME."; DB\_USER=".DB\_USER."; DB\_PASSWORD=".DB\_PASSWORD."; DB\_HOST=".DB\_HOST); ')

SQL\_COMMAND="mysql -u $DB\_USER -h $DB\_HOST -p$DB\_PASSWORD $DB\_NAME -e"

OLD\_URL=$(mysql -u $DB\_USER -h $DB\_HOST -p$DB\_PASSWORD $DB\_NAME -e 'select option\_value from wp\_options where option\_id = 1;' | grep http)

ALBDNSNAME=$(aws ssm get-parameters --region us-east-1 --names /Wordpress/ALBDNSNAME --query Parameters[0].Value)

ALBDNSNAME=`echo $ALBDNSNAME | sed -e 's/^"//' -e 's/"$//'`

$SQL\_COMMAND "UPDATE wp\_options SET option\_value = replace(option\_value, '$OLD\_URL', 'http://$ALBDNSNAME') WHERE option\_name = 'home' OR option\_name = 'siteurl';"

$SQL\_COMMAND "UPDATE wp\_posts SET guid = replace(guid, '$OLD\_URL','http://$ALBDNSNAME');"

$SQL\_COMMAND "UPDATE wp\_posts SET post\_content = replace(post\_content, '$OLD\_URL', 'http://$ALBDNSNAME');"

$SQL\_COMMAND "UPDATE wp\_postmeta SET meta\_value = replace(meta\_value,'$OLD\_URL','http://$ALBDNSNAME');"

EOF

chmod 755 /home/ec2-user/update\_wp\_ip.sh

echo "/home/ec2-user/update\_wp\_ip.sh" >> /etc/rc.local

/home/ec2-user/update\_wp\_ip.sh

Above script is going to do following,

* Populate environment variables after fetching data from Parameter store

## Download and extract Wordpress

## Configure the wordpress wp-config.php file

## Fix Permissions on the filesystem

* Mount EC2 instance with EFS mount target(NFS end point) which is pointed by EFSID parameter

Updated /**etc/fstab** file to do the same

* Updated WordPress tables to point to load balancer DNS name

1. Created **internet facing Application Load Balance**r (ALB) with a target group (no instance attached) . It will create Load balancer nodes in each of subnets within AZs. **Load balancer will provide resiliency and will abstract underlying architecture from user**
2. Created **Auto Scaling Group (ASG)** with min:Desired:max **1:2:4** pointing to **Launch template created at step 5 and attached load balancer (step 6)**  with ELB health check enabled (in addition with default EC2 health check. ASG will provide **scalable, self healing capability to WordPress applications.**
3. Created simple scaling policy with cloudwatch alarm to trigger alarm if CPU utilization is > 40% . If there is an alarm (CPU utilization > 40%) , ASG will add 1 capacity unit . It means desired capacity will be adjusted to provision 1 EC2 instance

Another scaling policy is created with cloudwatch alarm to trigger an alarm if CPU utilization is < 40% . If there is an alarm (CPU utilization < 40%) , ASG will remove 1 capacity unit . It means desired capacity will be adjusted to terminate 1 EC2 instance

1. Simulated scaling task with **stress utility** to generate HIGH CPU load . It can successfully scale out and scale in based on CPU utilization.
2. Created **Aurora DB cluste**r (Writer end point, Reader endpoint → load balancing reader 1, reader 2 (in the diagram specified) ). Have WriterEndpoint DNS mapped to an environment variable (DBEndpoint) . This DBEndpoint is configured to the WordPress config (wp-config.php) file, providing a **high available capability** of the database.

Thus we have a 3 tier WordPress application deployed into **Highly available, loosely coupled , scalable , self healing capable**, architecture.