Basic Setup:-

-I will assume that there is no resource was built before, so I will start to build the full production environment to be serving the word press blogging website.

Solution 1:-

- Create New VPC, after the creation, if we need the RDS instances to be later with a public accessed, there is need to enable DNS Resolution and DNS hostname in the VPC.

- Create Internet gateway and attach it to the vpc.

- Create two public subnets in different availability zones for high availability and fault tolerant.

- Create a public route table and attach it to the internet gateway, attach the subnets to the route tables through subnet association. There is already private route table created with the creation process of the VPC, so no need to create a new one.

**For security:-**

- Subnet level security: - Create NACL that allows inbound and outbound traffic from ssh, MySQL, and HTTP protocols. Then attach the two subnets to this NACL.

- Resources Level Security: - Create a security group that allows terrific from ssh, MySQL, and HTTP. It will be used later with other resources within the vpc.

**Create and launch RDS instance:-**

1 - Will create RDS MySQL database to act as independent centralized database among all the WordPress EC2 instances that will be created within the auto scaling group for best practice.

2 - Choose RDS mysql.

3 - DB instance class, chose the t2.micro type and for the allocated storage, chose 20 GB, but in the real world, we may choose larger types and storage depends on the applications and business needs.

4 - DB instance identifier name.

5- DB Master Username/DB Master password.

6- Choose VPC we created before.

7- Public accessibility – NO. Best Practice to keep the DB instance private and to be accessed only by resources within the VPC, but If we need the DB to be publicly accessed, the VPC created before must have DNS Resolution and DNS hostname enabled.

8- Choose Availability zone.

9- Choose to create new vpc security group for the DB instance.

10 - Database Name.

11- Database Port.(3306).

12- Choose Multi availability Zones for higher availability and fault tolerant.

12- Launch the database.

13- After DB creation, will create a read replica to be used for caching the database read data and for better DB performance, decrease the load on the master DB and in case of DB failure, it will be promoted to be Master DB.

**<u>Prepare the environment to be highly available, scalable and elastic when the audience demand increases:-</u>**

**<u>- Creating an auto scaling group to scale the EC2 instances when needed:-</u>**

**<u>- Create launch configuration for the auto scaling group:-</u>**

1- Choose the AMI that will be used. (I chose Amazon Linux for this challenge).

2- Choose the instance type.( I chose t2.micro for this challenge but in real-world environment it can be different depends on the application and business requirements).

3- Write down launch configuration name.

4- In advanced options, will write down a bootstrap bash script for perquisites of wordpress, so that to be applied to the initial instance in scaling group during the instance creation process as below:-

```
#!/bin/bash

sudo yum update -y

sudo yum install -y httpd php php-mysql

sudo yum install –y wget

Sudo service httpd start

sudo chkconfig httpd on

sudo wget http://wordpress.org/wordpress-latest.tar.gz -P /var/www/html/ force=yes

sudo tar xzf /var/www/html/wordpress-latest.tar.gz -C /var/www/html  --strip-components 1

sudo chmod u+rw -R /var/www/html

sudo chmod go+rw -R /var/www/html

sudo chown apache:apache -R /var/www/html
```

4- Add Storage, won’t add EBS volumes in addition to the default EBS root volume for this challenge, but in a real-world case, we can add more EBS volumes and mount it in the EC2 later depends on the application and business needs.

5- Add Security group, chose the one already created before for this environment which will allow ssh and HTTP and MySQL traffic.

6- Finally, will create a keypair authentication to be used later with the ssh login access to the EC2 instances.

**<u>- Create the auto scaling group:-</u>**

1- Add group name.

2- Add the number of instances to start with, in this case its recommended to start with only one instance, although the best practice is to start with minimum two EC2 instances in two different subnets , but for better solution, will configure all the initial deployment of the wordpress website (as per the bash script bootstrap in the launch configuration + the setup of the wordpress configurations to be ready for blogging) , then we will take an image (AMI) from this word press instance, and will create another EC2 from this AMI in different subnet, and finally the new instance will be added to the scale group created in this section.

3- Choose two different subnets.

4- In advanced options, won't add load balancer for now, and the health check grace period to be 300 seconds.

5- Use scaling policies to adjust the capacity of this group for scaling up and scaling down by using alarms:-

5.1 - Choose simple scaling policy, then create alarm for scaling up:-

5.1.1 SNS topic if available to send notifications by email or sms when the alarm is triggered, set the alarm to trigger when the CPU utilization is in average above 80% for at least 1 period of 5 minutes).

5.1.2 Next step, is to add a number of instances to be launched, will choose 2 instances and then wait for next scale activity for 300 seconds.

5.2- Choose simple scaling policy, then create alarm for scaling down

5.2.1 SNS topic if available to send notifications by email or sms when the alarm is triggered, set the alarm to trigger when the CPU utilization is in average below 40% for at least 1 period of 5 minutes).

5.2.2 Next step, is to remove a number of instances, will choose 2 instances and then wait for next scale activity for 300 seconds.

6- Add name tags to instances as optional, then launch the autoscale group.

**- Setup WordPress on EC2:-**

1- To avoid failure with word press blog from IP changes if the EC2 instance restarted, create an elastic IP to be associated with the WordPress EC2 instance:-

1.1 Create elastic IP, then from actions choose associate.
1.2 Choose the word press EC2 ID, then attach.

2- Open a web browser and write down the Elastic public IP/Public DNS Name.

3- Wordpress start webpage will appear, continue with the configuration for the WordPress.

4- In the WordPress database configuration, write down database name, database host, database username, database password and table prefix if needed.

5- Continue with the installation and voila, the word press blog is now live ☺.

**- Backup the WordPress instance by creating snapshot:-**

1- Create a snapshot, select the volume attached to the WordPress instance.

-Create AMI from the snapshot and launch new WordPress instance:-

1- From the snapshot created above, create an image AMI.

2- Choose the same instance type, storage, and attach the same security group for the WordPress instance created before, then launch.

-Adding the new WordPress instance to the autoscale group:-

1- Select the new WordPress instance after it is created, and from actions choose to attach to autoscaling group.

2- Choose the same autoscaling group for our master WordPress instance.The launch configurations will automatically change the desired instances to two instead of one.

**- Create load balancer to manage traffic to the WordPress instances:-**

-Will use the classic load balancer in this challenge, but in the real world, when the blog will have as an example a separate section for pop music, and other for rock music, we can use the application load balancer with the target group and distribute the traffic by HTTP header host based URL or path URL.

1- Create classic load balancer.

2- Choose a name, Vpc , and configure the load balancer listener to port 80.

3- Choose the two subnets where the WordPress EC2 is allocated.

4- Choose the same security group created before for the WordPress Ec2 instances.

5- Configure the Health Check, will choose that the load balancer will check through TCP protocol to ping port 80.

6- Choose the two WordPress instances to be under the management of the load balancer.

7- Launch the load balancer.

**-Create CloudFront distribution:-**

- For better performance, will use cloud front with the default settings except for the origin domain name will be the load balancer DNS name created before.

OTHER SOLUTIONS: -

Finally, this environment can be created with different methods as well like:-

1- Terraform script to automate the creation of the previous environment by creating all the resources starting from vpc down to load balancer and Ansible playbook to install all the prerequisites for the WordPress installation (like what had been done in the launch configuration bootstrap).
2- Use the cloud formation WordPress template, which comes with multiple availability zones RDS instance and autoscaling group and load balancer ready configured.
3- Follow the amazon documentation https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/php-hawordpress-tutorial.html to create wordpress website in an elasticbean stalk with external RDS instance.