# Mini Project-2

**Task: Deploying a Two-Tier Application (WordPress) Using Amazon ELB**

**Steps: 1)** Create a personalized VPC with two public and two private subnets in at least two different availability zones.

2) Deploy a Bastion host in a public subnet through which we will SSH to WordPress servers placed in private subnets.

3) Deploy two temporary WordPress instances on the same private subnet in the same availability zone as that of the bastion host. We create a new Security Group for all WordPress servers that allow SSH inbound connection, with Source from the Security Group of the Bastion host.

4) Create a NAT gateway to solve the problem where the instance itself is unable to connect to the Internet to download software or make critical updates. Deploy a NAT Gateway in the same public subnet with the Bastion host and assign an Elastic IP for the NAT Gateway.

5) Create two RDS databases with a new security group and in the inbound rules, select MYSQL/Aurora with Source from the Security Group of the WordPress app. With this option, only WordPress servers can access the RDS.

6) Create two EFS shared drives to store all configurations and data for the WordPress application with a new security group that allows incoming NFS traffic directly from the Security Group of WordPress servers. Deploy on private subnets. Click on the created EFS, then

click attach on the right top corner and store the command to attach it using nfs client.

7) In the case of an amazon linux machine, execute the following commands on both the temporary instances after ssh via the Bastion host.

a) sudo yum update -y

```
sudo yum install php -y
sudo yum install php-mysqli -y
sudo yum install mariadb105 -y
```

b) After installing the necessary libraries, start the following services.

```
sudo systemctl start httpd
sudo systemctl enable httpd

sudo systemctl start php-fpm
sudo systemctl enable php-fpm
```

c) Then modify the permissions to access the apache public folder.

```
sudo usermod -a -G apache ec2-user
sudo chown -R ec2-user:apache /var/www/html
sudo chown -R apache:apache /var/www/html
```

d) Paste the command from step 6, one into each instance. Instead of efs, end the command with /var/www/html/.

e) sudo vim /etc/fstab

f) Append the following line to the end of the file and save it.

```
fs-020a682c5ba2f0fc3.efs.us-west-1.amazonaws.com:/
/var/www/html/ nfs defaults,_netdev 0 0
```

Replace fs-020a682c5ba2f0fc3.efs.us-west-1.amazonaws.com: with the dns from step 6.

g)
```
wget https://wordpress.org/latest.zip
unzip latest.zip
sudo cp -R wordpress/* /var/www/html/
```

h)
```
mysql -h <respective rds endpoint> -P 3306 -u admin -p
```

i)
```
create databases project;
```

j)
```
show databases;
```

8) Select the temporary instances one by one, in “Actions,” choose “Image and Templates,” and then select “Create image.”

9) Once it is done, go to EC2 and create a launch template.

10) In the “Application and OS Images” option, select “My AMIs” and then choose your newly created instance images.

11) Select the existing WordPress key pair for the launch template and the existing Security Group for the WordPress instance.

12) Navigate to the EC2 console and select “Auto Scaling Groups.” From there, choose the recently created launch templates one by one.

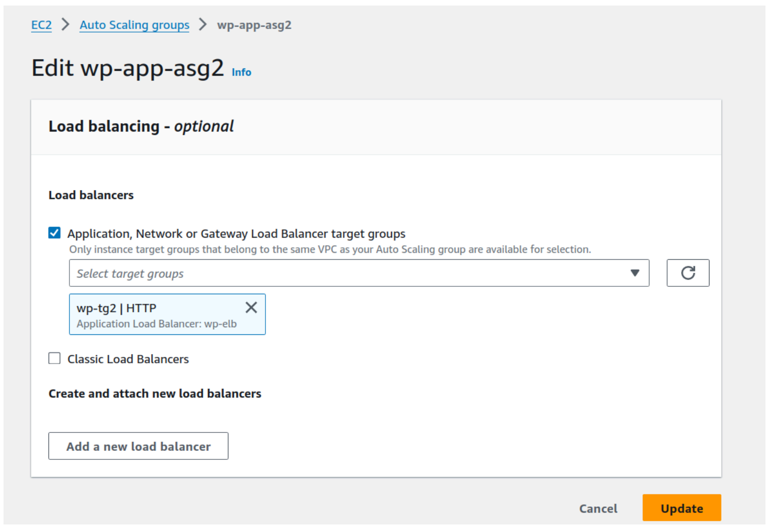
13) In the “Network” option, select the private subnets in both Availability Zones.

14) This creates two instances each in the auto scaling groups for both the instances.

15) Create two target groups for the load balancer. Do not add running instances to the target group.

16) Create an application load balancer and configure it to listen to both the target groups 50-50%. Ensure that the Load Balancer is placed in all public subnets, and the security group for the Load Balancer is configured to listen to both HTTP and HTTPS connections.

17) Go back to the autoscaling groups and attach them to different target groups which are attached to the same load balancer.


EC2 > Auto Scaling groups > wp-app-asg2

Edit wp-app-asg2 Info

**Load balancing - *optional***

**Load balancers**

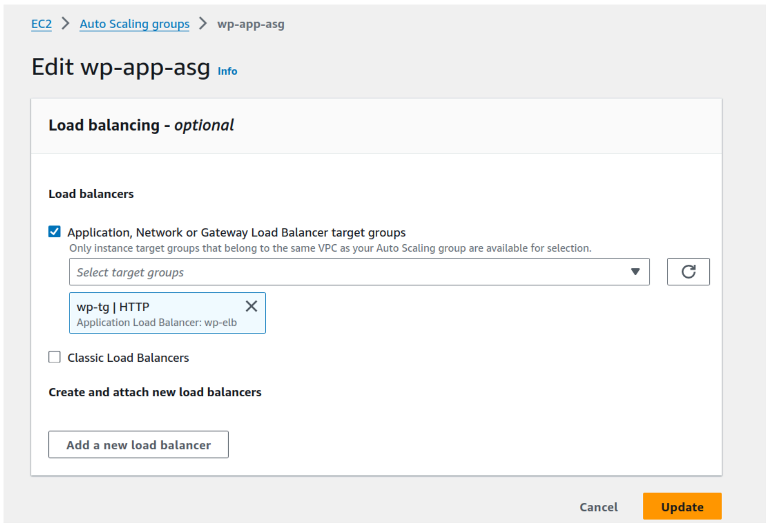
☑ Application, Network or Gateway Load Balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

*Select target groups*

wp-tg2 | HTTP
Application Load Balancer: wp-elb

☐ Classic Load Balancers

**Create and attach new load balancers**


EC2 > Auto Scaling groups > wp-app-asg

Edit wp-app-asg Info

**Load balancing - *optional***

**Load balancers**

☑ Application, Network or Gateway Load Balancer target groups
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

*Select target groups*

wp-tg | HTTP
Application Load Balancer: wp-elb

☐ Classic Load Balancers

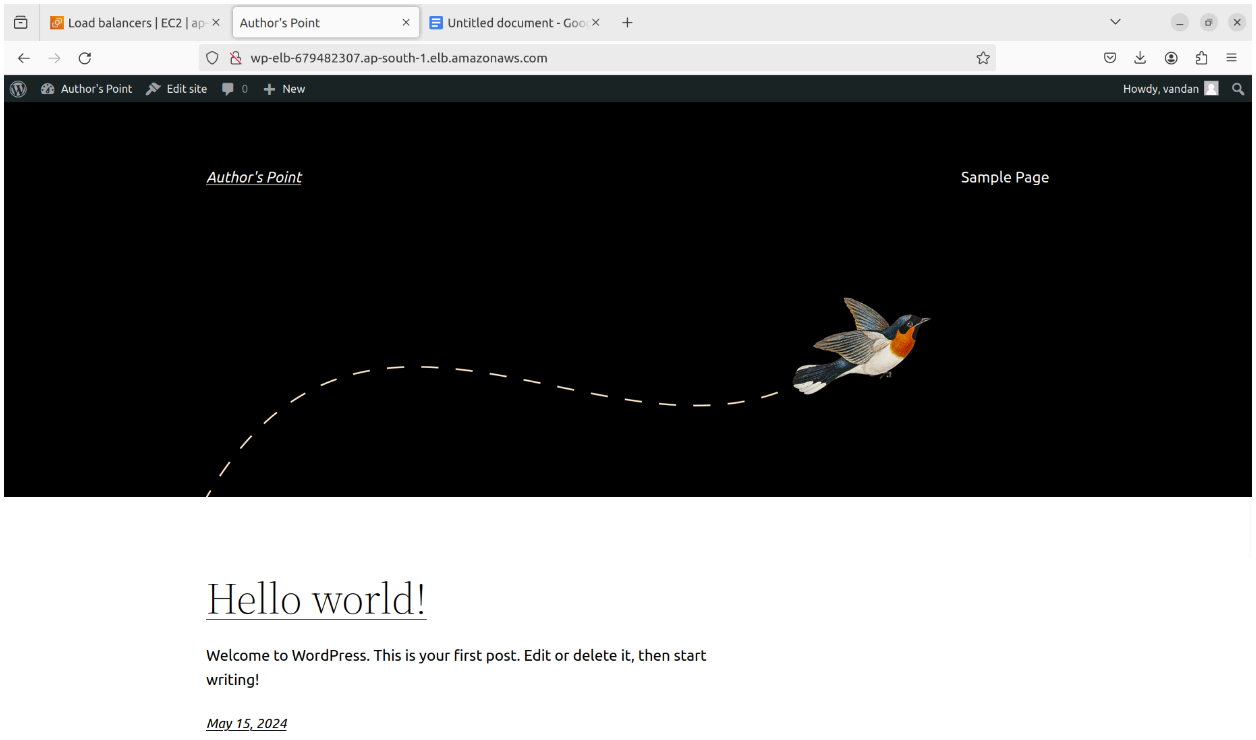
**Create and attach new load balancers**

18) To allow incoming traffic from the Load Balancer, navigate to the Security Group associated with the Auto Scaling group. Add inbound rules for both HTTP and HTTPS protocols, specifying the source as the Security Group of the Load Balancer.

19) To test the new Load Balancer, copy its endpoint and paste it into your browser. We'll get different wordpress websites after reloading each time.

20) Setup username and password for both the wordpress websites. We can edit the websites using /wp-admin at the end of the load balancer url every time.

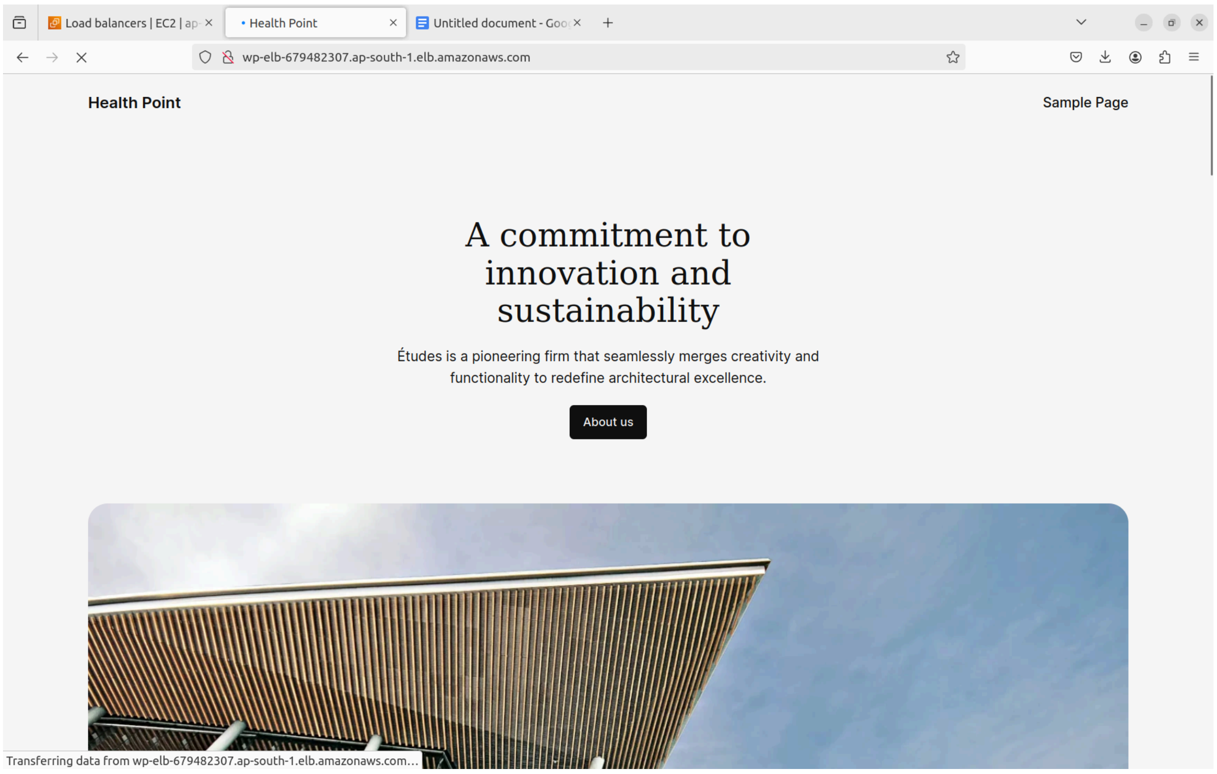
21) Task is accomplished.



Hello world!

Welcome to WordPress. This is your first post. Edit or delete it, then start writing!

May 15, 2024


Load balancers | EC2 | ap
Health Point
Untitled document - Goo
wp-elb-679482307.ap-south-1.elb.amazonaws.com
Health Point
Sample Page

A commitment to innovation and sustainability

Études is a pioneering firm that seamlessly merges creativity and functionality to redefine architectural excellence.

Transferring data from wp-elb-679482307.ap-south-1.elb.amazonaws.com…