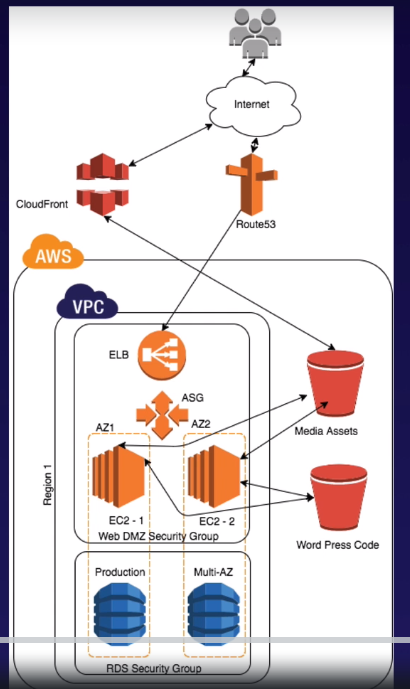
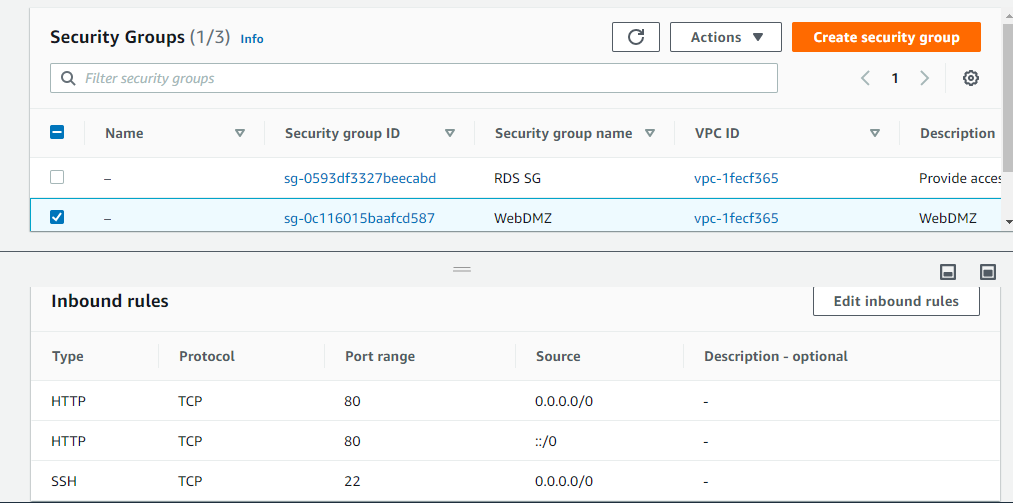
**High Availability Fault Tolerance Architecture**



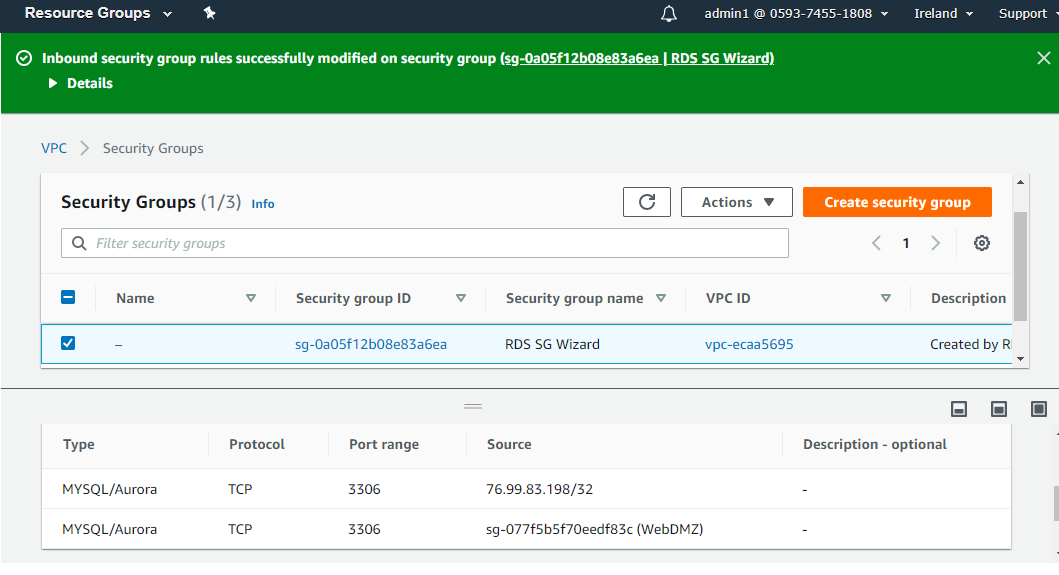
Architecture Design

1. **Create two Security group**:

**WebDMZ** for interacting with internet.



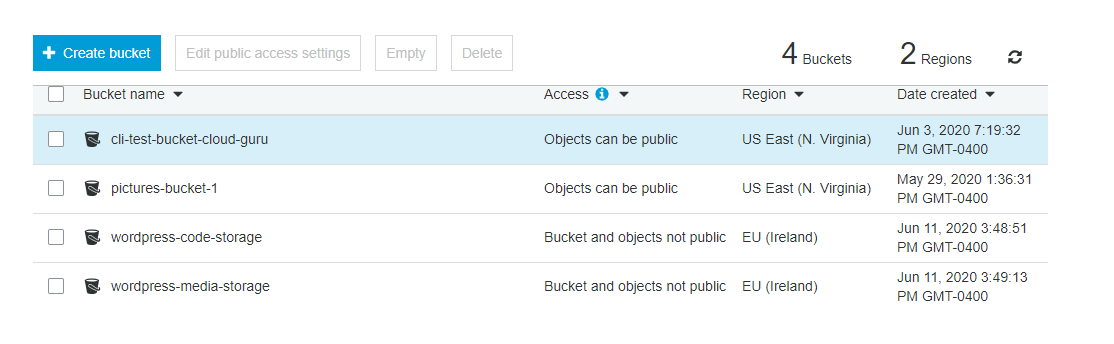
**RDS SG Wizard** to connect to MySQL server that will allow **WebDMZ** SG to connect MySQL/Aruora on 3306 port



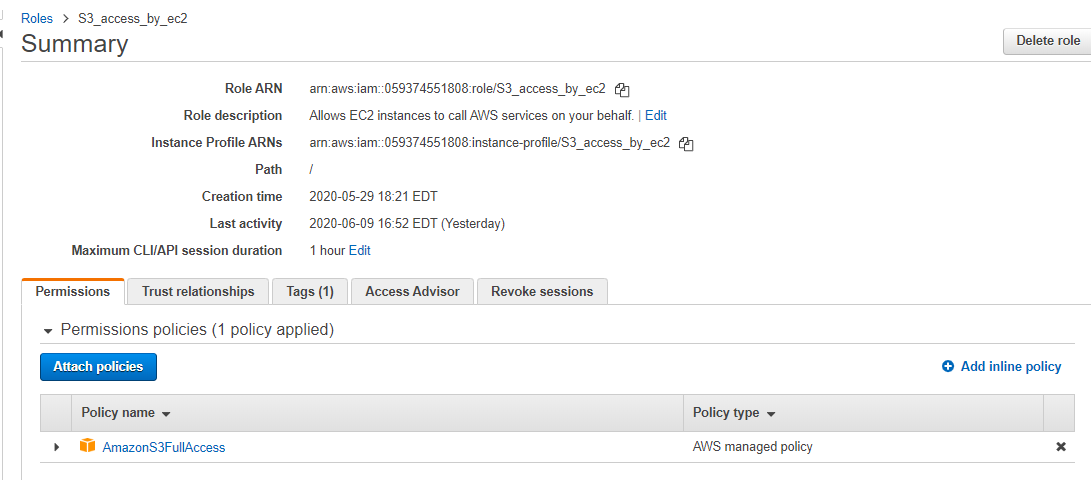
2. **Create two S3 buckets , one for code backup and other for media files in same region (Ireland in this case).**

**Wordpress-code-storage**= for storing code

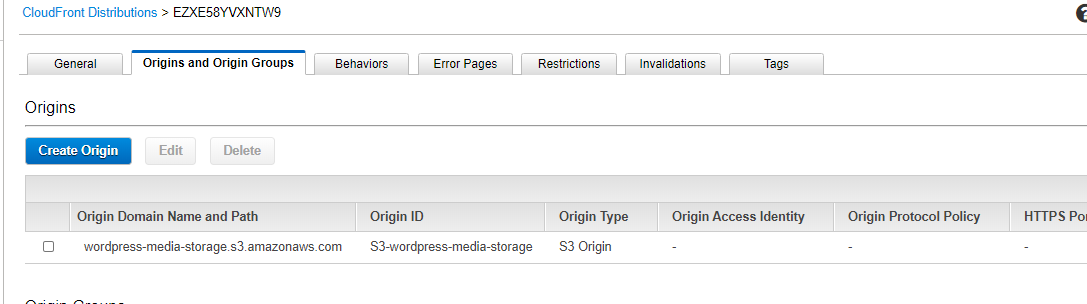
**Wordpress-media-storage**= for storing media files(images, sound etc)

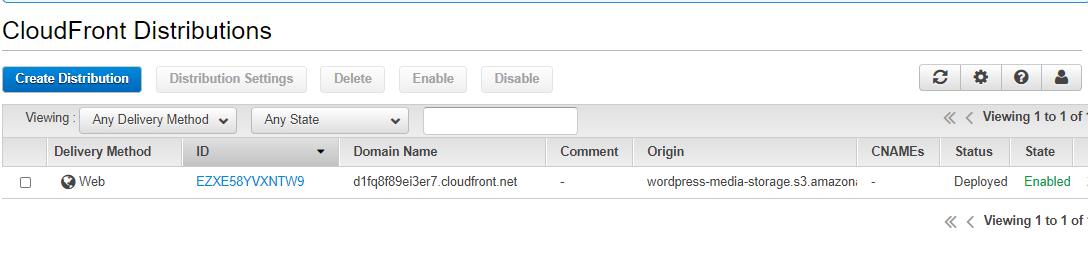


**3. Create Role “*S3\_access\_by\_EC2*” to provide S3 full access to EC2 instance**



4. **Create Cloud Front using the *wordpress-media-storage* S3 bucket as origin. So, that the requests can be served from the edge locations instead of fetching from the site.**





5. Launch the EC2 instance (launched in Ireland region as of now), select the above Role **S3\_access\_by\_EC2** and **WebDMZ** security group.

Paste the bootstrap script provided in the user-data section while configuring the instance or use file user-data Bootstrap.sh

***#!/bin/bash***

***yum update -y***

***yum install httpd php php-mysql -y***

***cd /var/www/html***

***echo "healthy" > healthy.html***

***wget https://wordpress.org/wordpress-5.1.1.tar.gz***

***tar -xzf wordpress-5.1.1.tar.gz***

***cp -r wordpress/\* /var/www/html/***

***rm -rf wordpress***

***rm -rf wordpress-5.1.1.tar.gz***

***chmod -R 755 wp-content***

***chown -R apache:apache wp-content***

***wget https://s3.amazonaws.com/bucketforwordpresslab-donotdelete/htaccess.txt***

***mv htaccess.txt .htaccess***

***chkconfig httpd on***

***service httpd start***

This script will :

1. Update the selected AMI OS.

2. Install apache php and mysql

3. Create a healthy.html file in the cd/var/www/html dir , that will be used by the Elastic Load Balancer for performing health check on the target group nodes. We will provide this healthy.html file name while configuring health check for the ELB.

4. Download the latest version of Wordpress and unzip it , copy to the dir /var/www/html, remove the installed dir and the downloaded zip file.

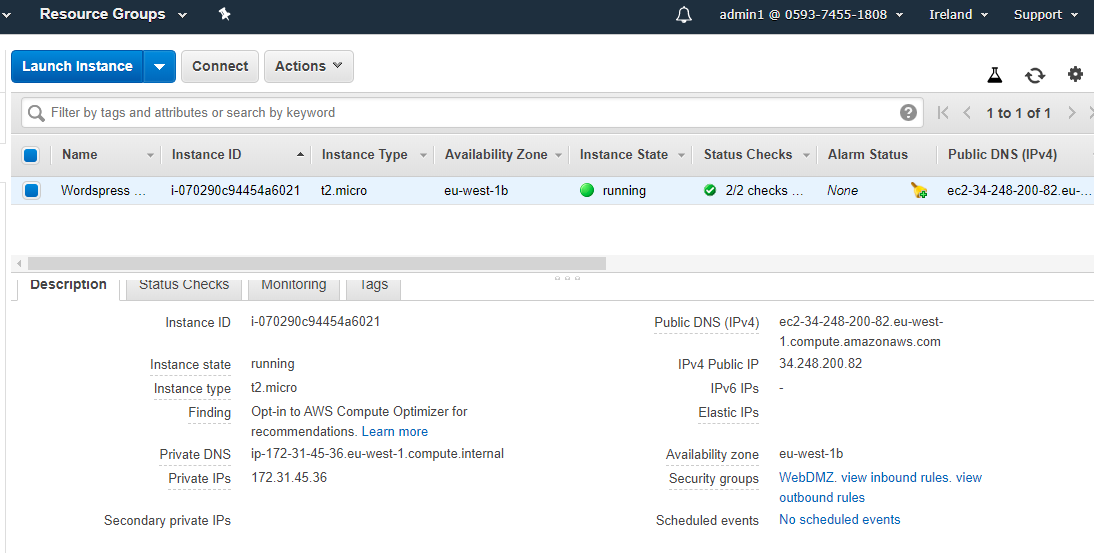
5. Provide write permission to the wp-content dir.

6. Download the htaccess.txt file to perform URL rewrite ( so that media will be served from the CloudFront instead of the site or s3 bucket)

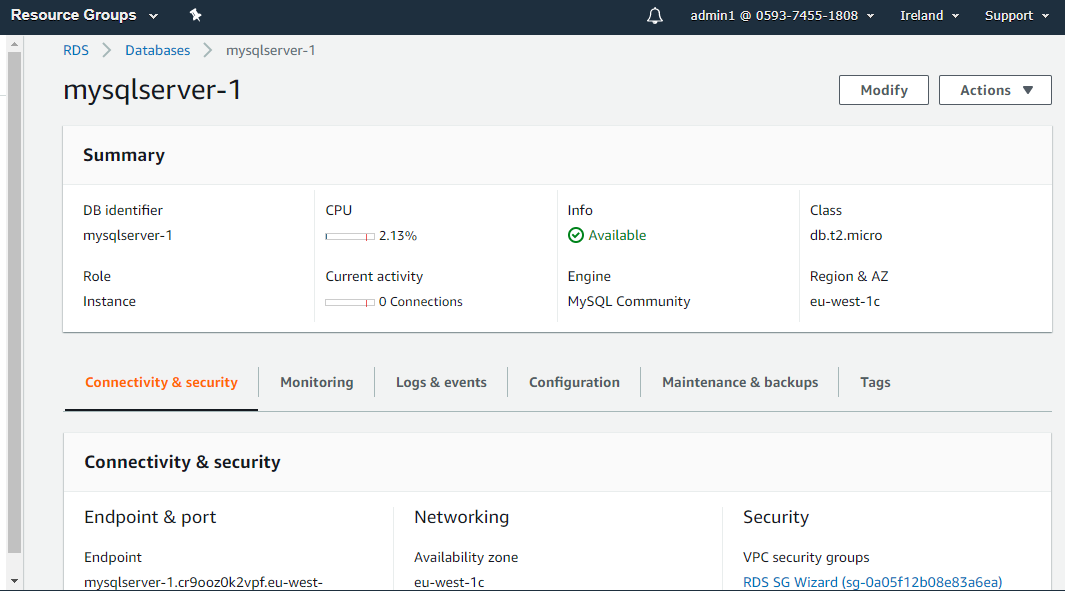
7. Rename file htaccess.txt to **.htaccess,** to make it hidden.

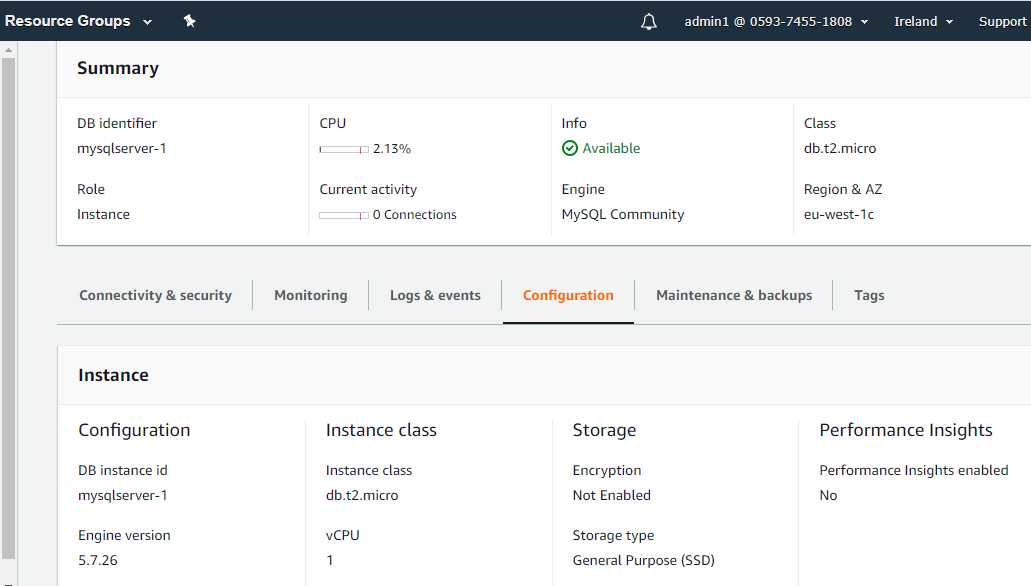
8. *chkconfig httpd* *on* -Make sure that apache will backup, if the Ec2 instance restarts.

9. Start the apache.

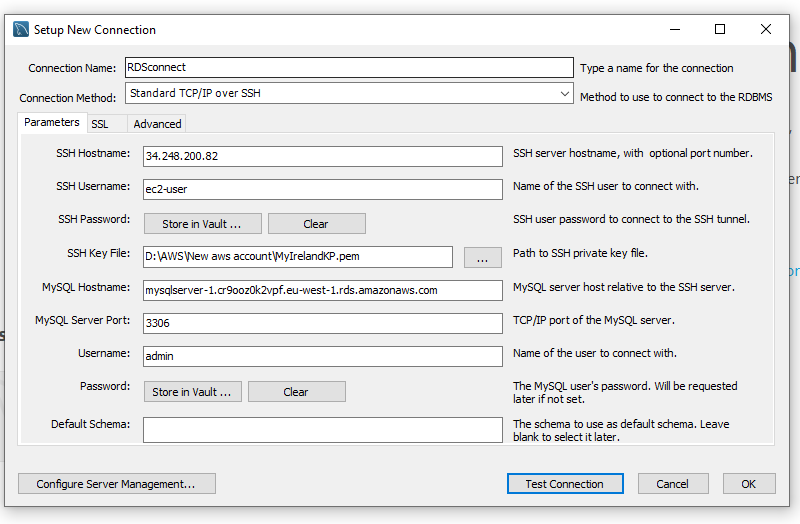


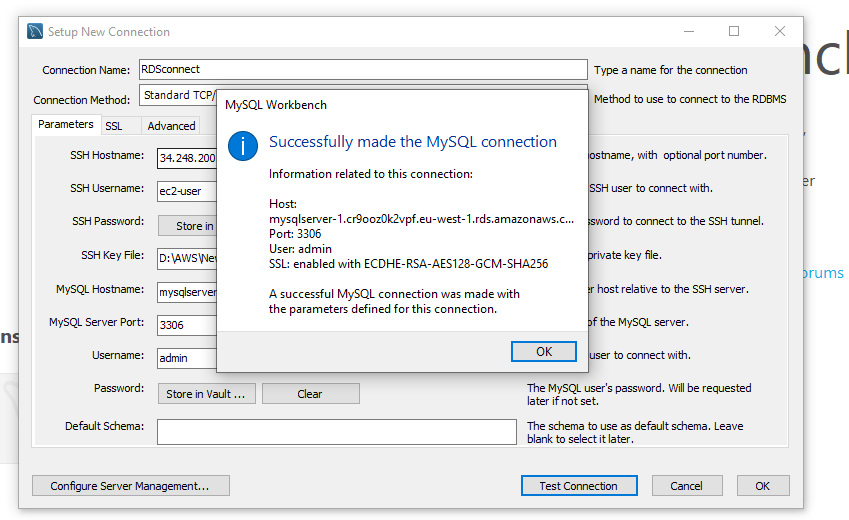
6. **Create a MySQLServer RDS instance to connect with word-press site, use the MySQL Server and DEV mode to launch the RDS instance (Free tier don’t allow multi-AZs option)**

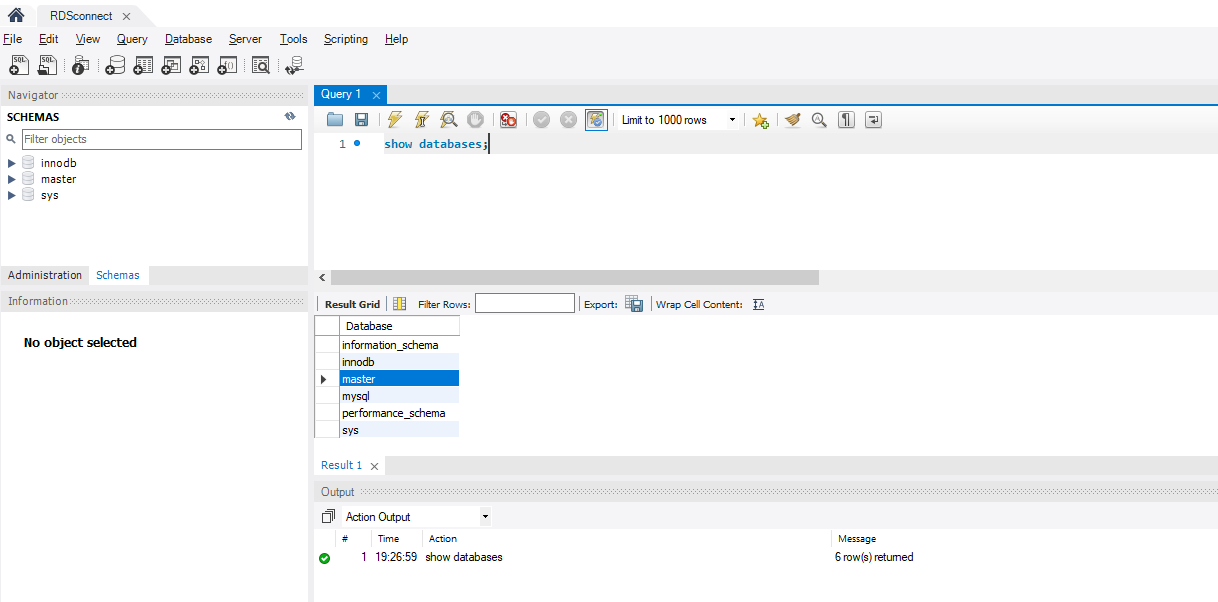




Test connection by filling the SSH details using MySQL Workbech.

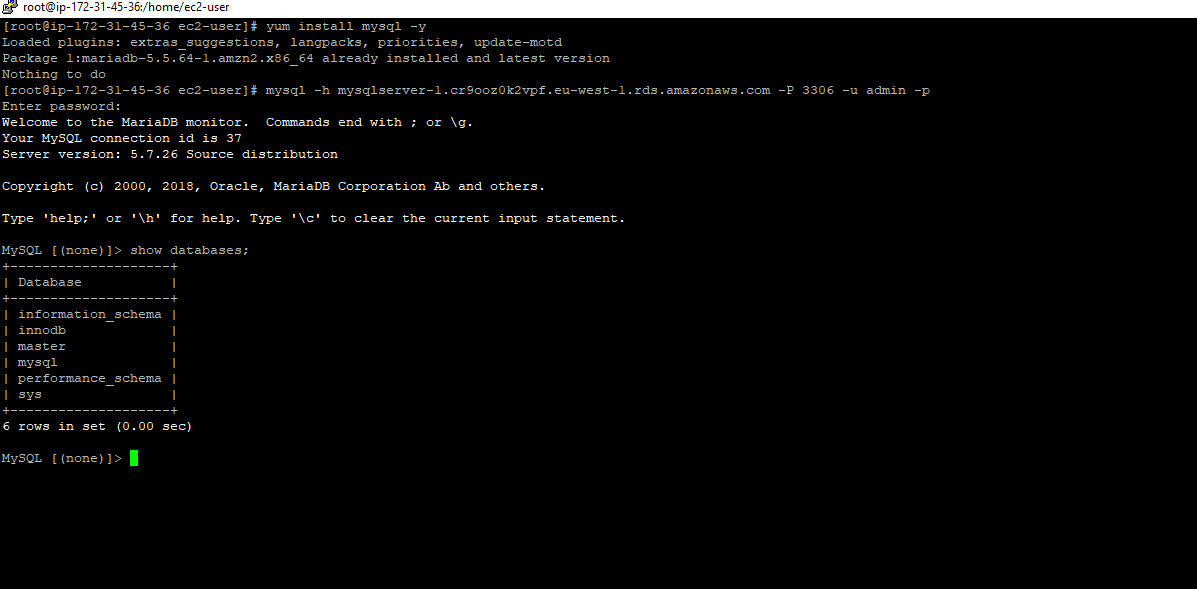






Or using command prompt via ssh into EC2 instance

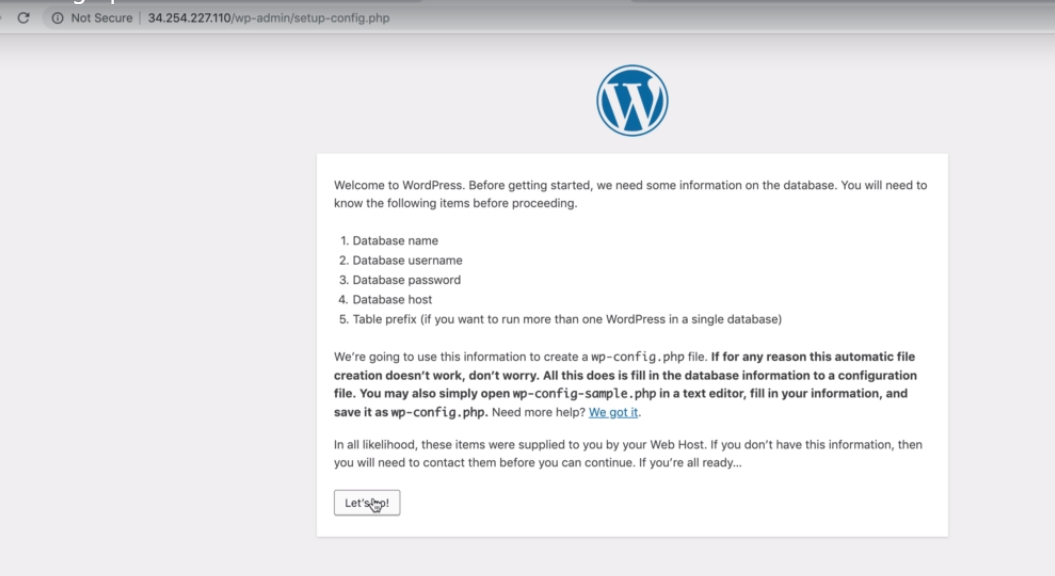
***mysql -h mysqlserver-1.cr9ooz0k2vpf.eu-west-1.rds.amazonaws.com -P 3306 -u admin -p***



**7. Configure the WordPress site.**

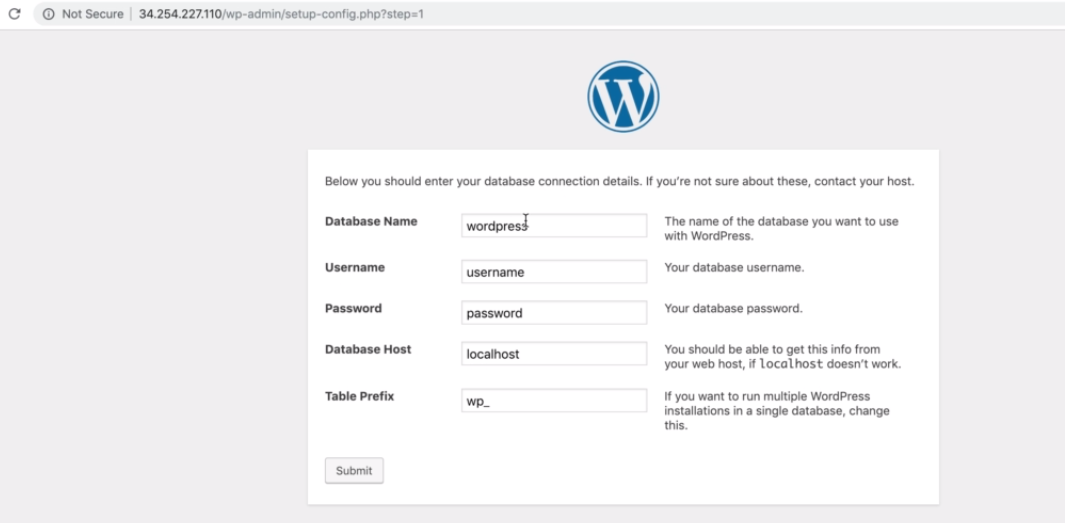
Check the website running on the public IP of the EC2 instance.

After the EC2 instance is up, hit the browser with the public IP address and click on **“Lets Go”.**



Enter the RDS details, like database name, username , password. Enter the RDS instance endpoint in the Database host and click “Submit”.





Next, we need to copy the wp-config.php file and create it the EC2 instance in the dir /car/www/html



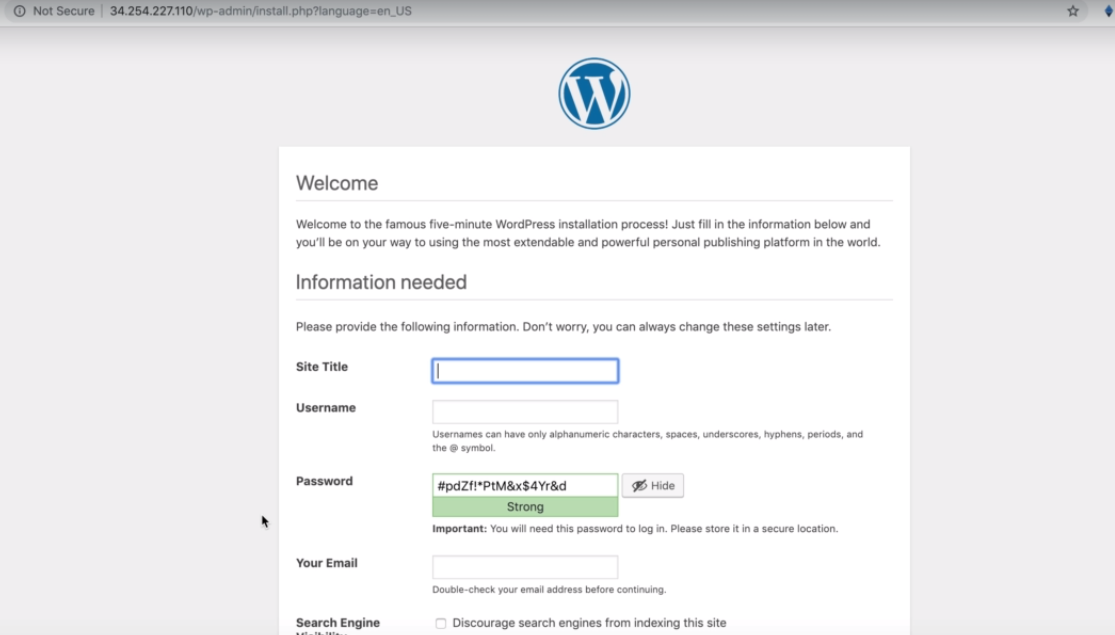
***-cd /var/www/html***

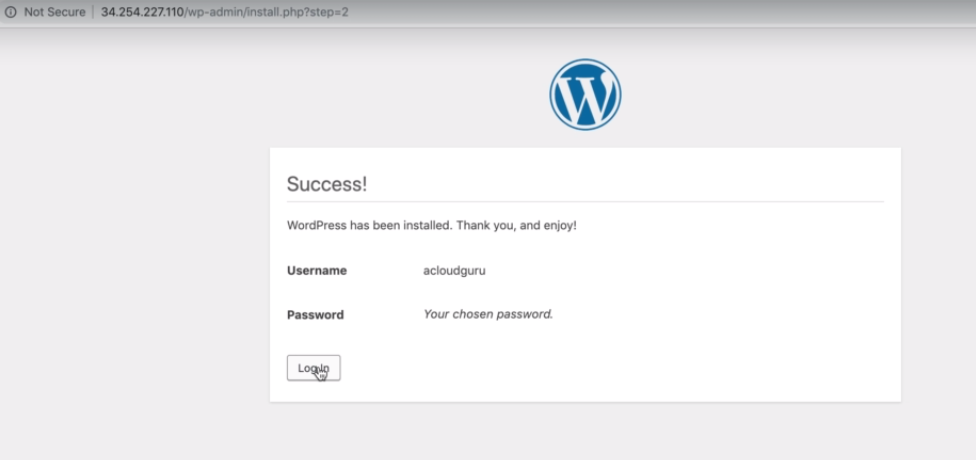
***-nano wp-config.php***

Paste the copied text and Ctrl+x , Yes and enter.

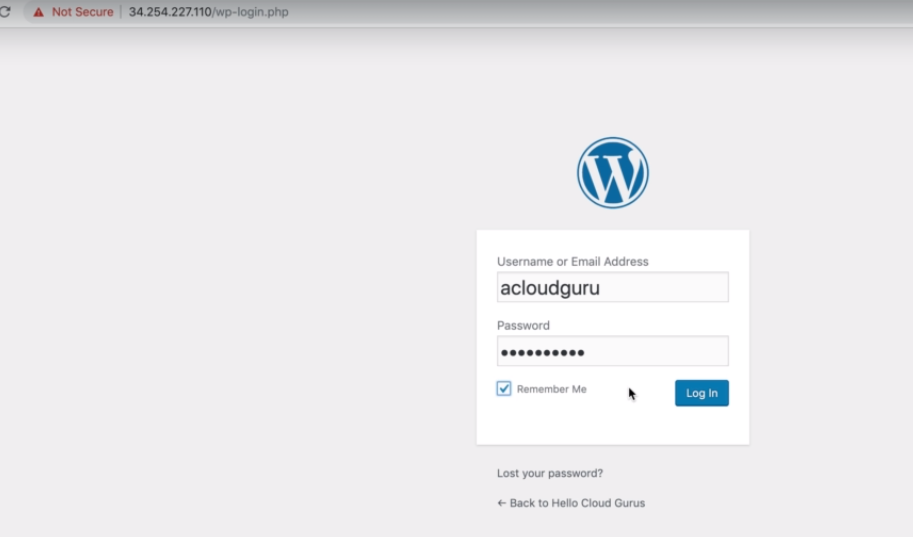
Then click on the **“Run installation”.**

Then, enter the site details and click on **“Install” .**

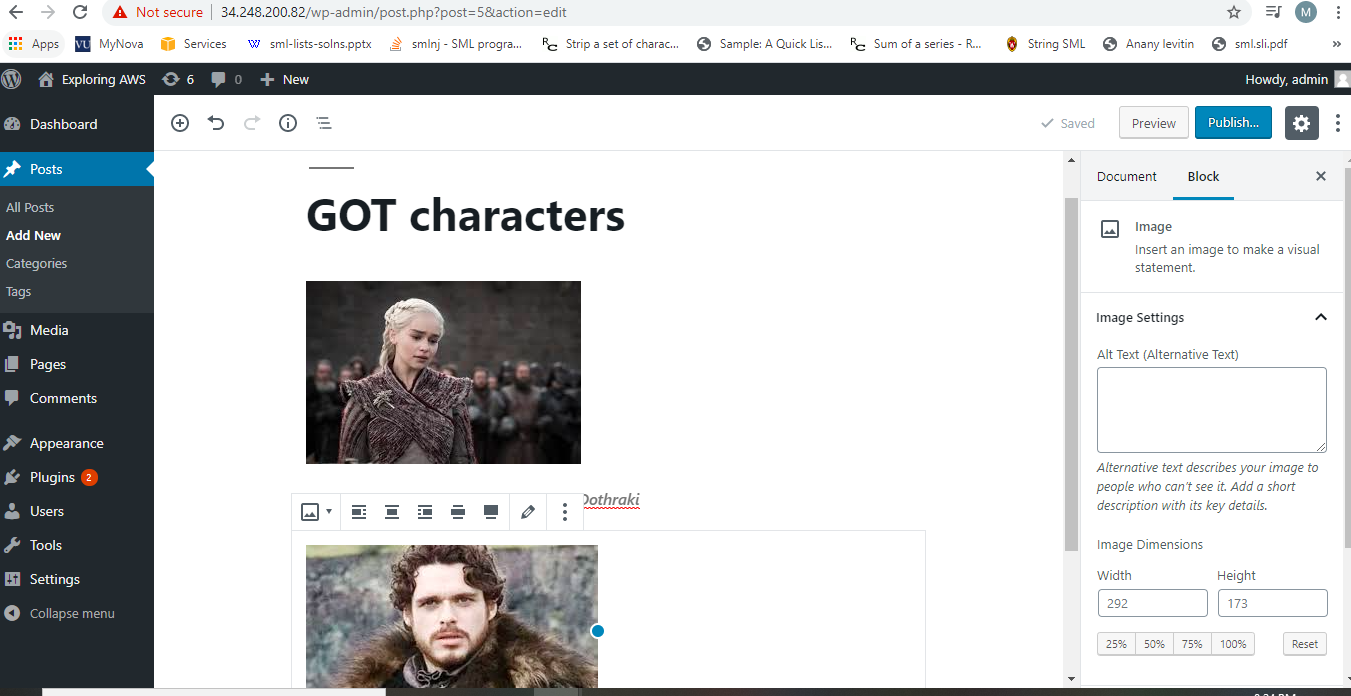




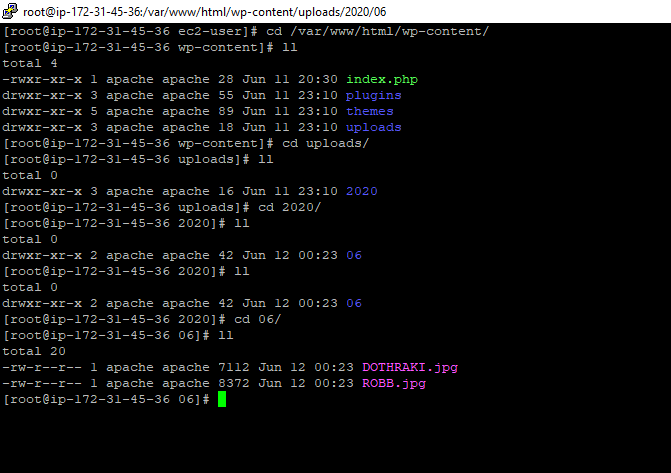
Then, click **“Login”** and enter the username and password.



Create a new post, for eg. I created with the GOT characters and click **Publish.**



Check the uploaded images in word-press content in EC2 instance

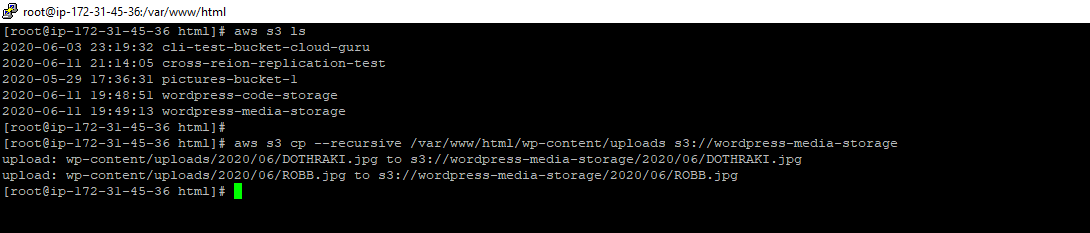


8. **Create Ec2 instance content backup on S3 bucket and enable content to be served from CloudFront Edge location.**

Now we want every time when a file is uploaded on WordPress site , it will be replicated on S3 **wordpress-media-storage** bucket that serve as origin of CloudFront.

And we want CloudFront instance to serve these files rather than EC2 instance so that site will load faster.

We will be able to copy files to S3 using the S3 access role.



Now, we can implement redundancy, so that if the site fail(EC2 instance failure), we can have copy of the site code as well. So, we will backup the site code into S3 bucket **wordpress-code-storage.** This will help in fault tolerance because if the EC2 instance failed where we hosted our site then the Auto-scaling group can run the heath check of the failed nodes and make another EC2 instance up within seconds so, we still have our site running with the same code backed-up at S3 bucket.

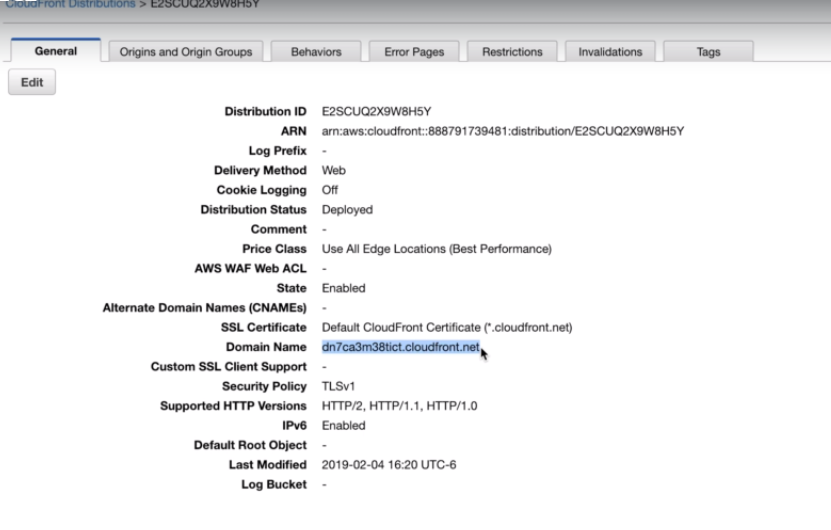
In case of EC2 instance failure, we can have Auto-scaling group that can easily be up and pull the code from the S3 bucket without having to bootstrap the instance again manually.

***aws s3 cp --recusrsive /var/www/html s3://wordpress-code-storage***

This will recursively copy the content from fir /var/www/html to the S3 bucket.  

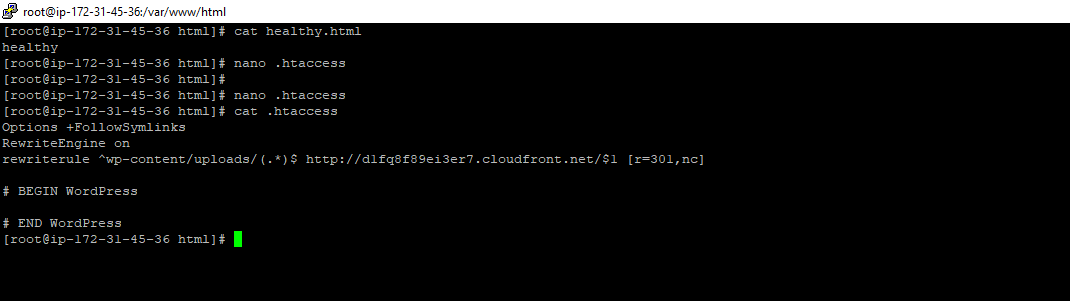

To serve images from CloudFront instead of serving it from EC2 ins. We have a hidden file “htaccess” that needs to be modified and rewrite the rule. It will reroute the URL of all the media/images from the cloudfront.

Copy the CloudFront Domain Name.



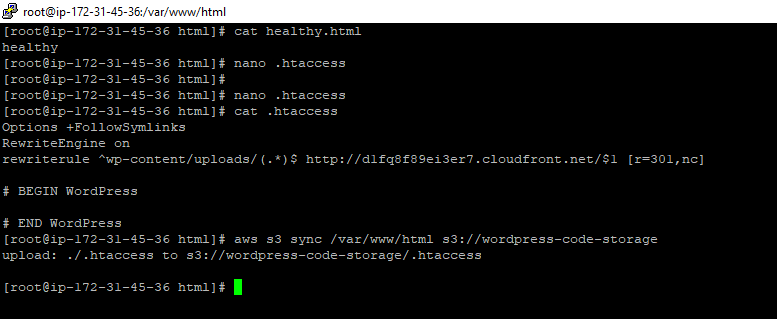
Edit the .htaccess file and replace the Domain name of the CloudFront organization created above in file and save.

***nano .htaccess***



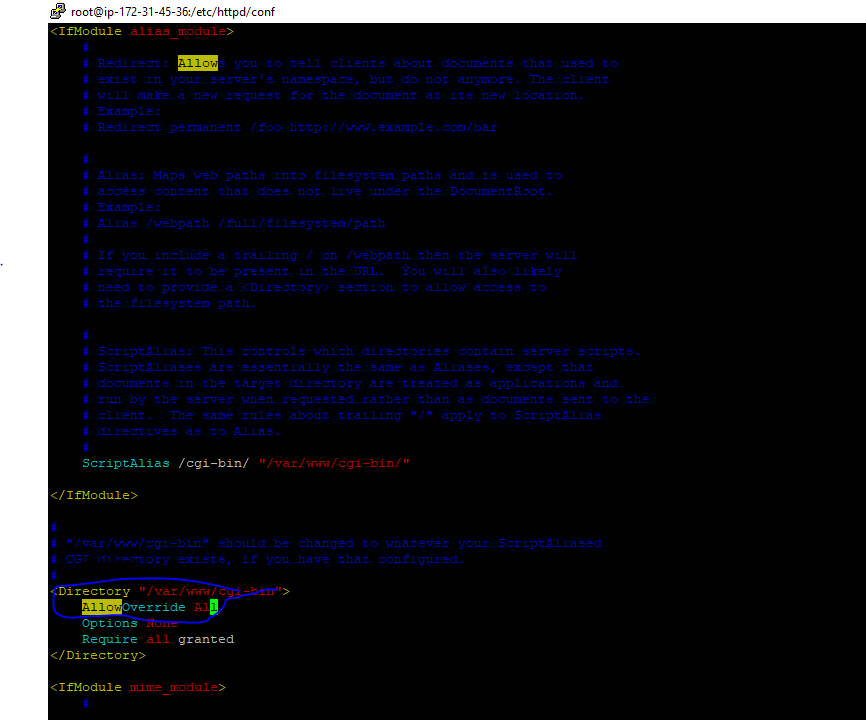
Now sync the modified htaccess file with the backup maintained on S3 bucket

***aws s3 sync /var/www/html s3://wordpress-code-storage***

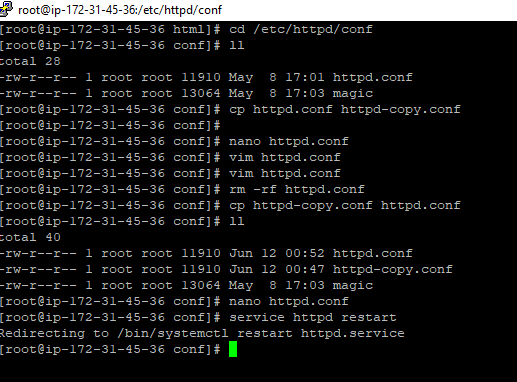


Next step is to, tell apache to enable and allow URL rewrite , go to cd /etc/httpd/config

Edit the httpd.config file and AllowOverride to All



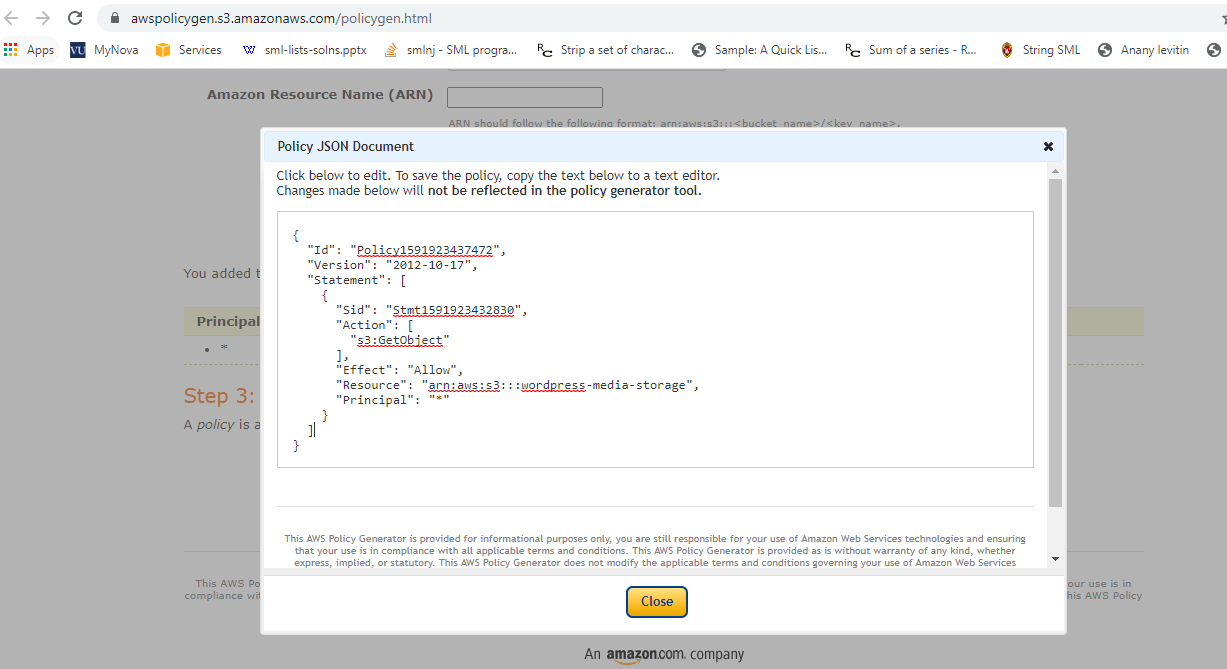
Restart the apache to allow rewrite rules to take effect.



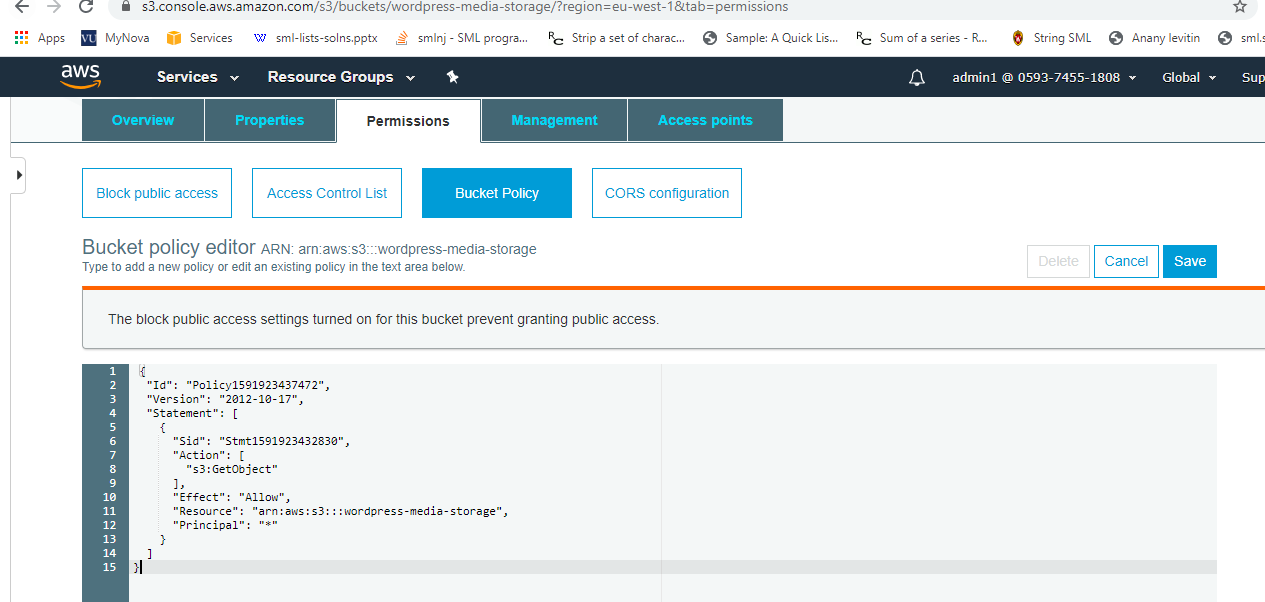
9. **Create bucket policy to allow read/ get objects public from *wordpress-media-storage* S3 bucket.**

Now, create a bucket policy for media S3 bucket to make this bucket public, so that Auto-scaling group can copy the code and make another instance available in case of original site failure:

Use the policy generator to generate the policy or copy the bucket-policy from the file **public bucket policy.txt.** Make sure to update the arn of the bucket you want to make public in the generated policy.

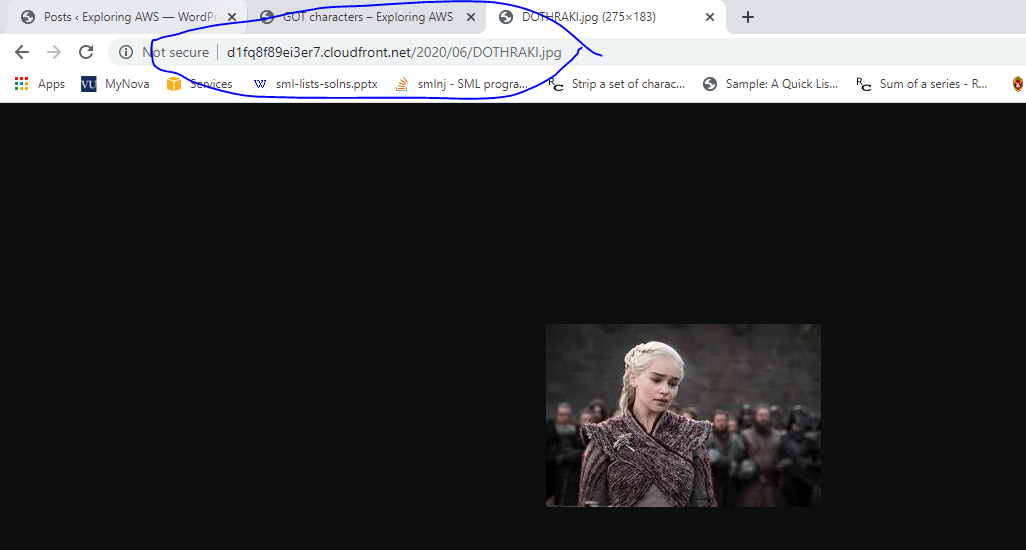


Paste the policy in the bucket policy found inside Permission tab of the bucket.



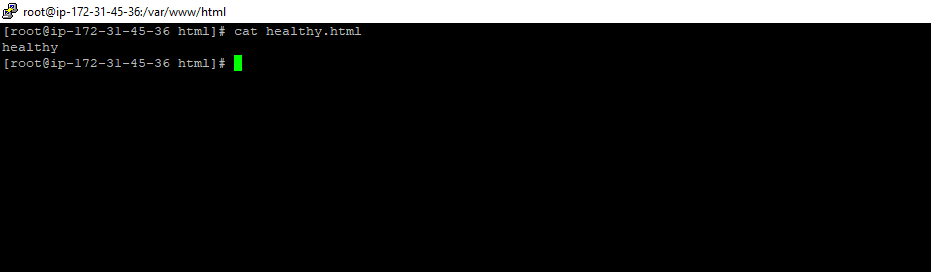


Now, if we check the image address in the EC2 instance site , it will be served from CloudFront endpoint not from the EC2 instance IP.

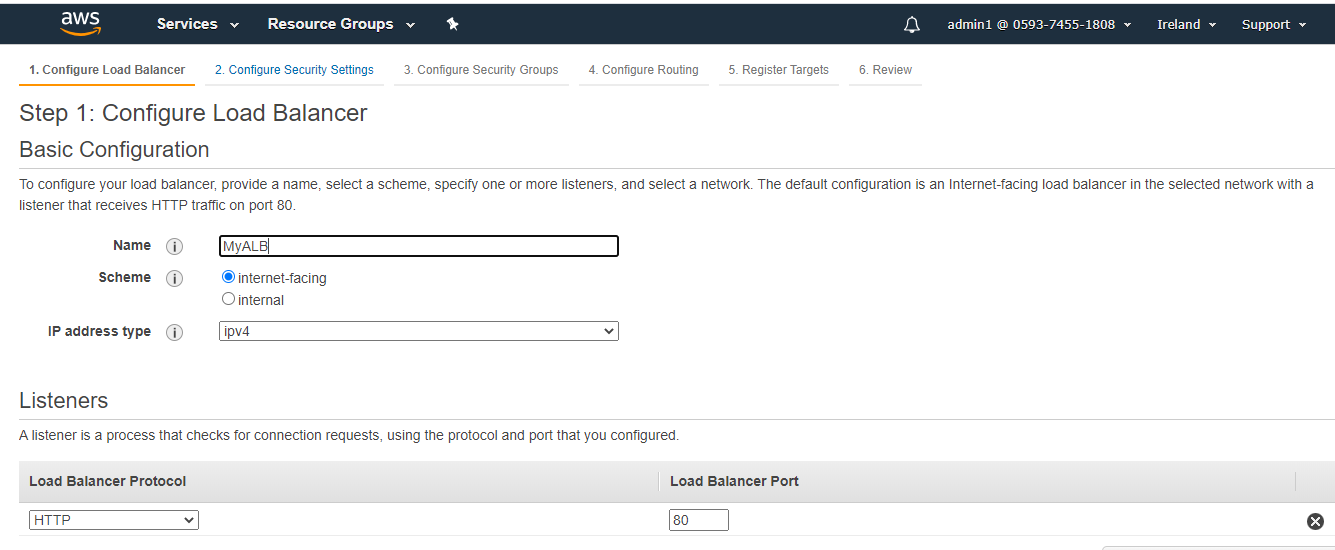


**10. Create Application Load Balancer to distribute traffic over EC2 nodes in the network.**

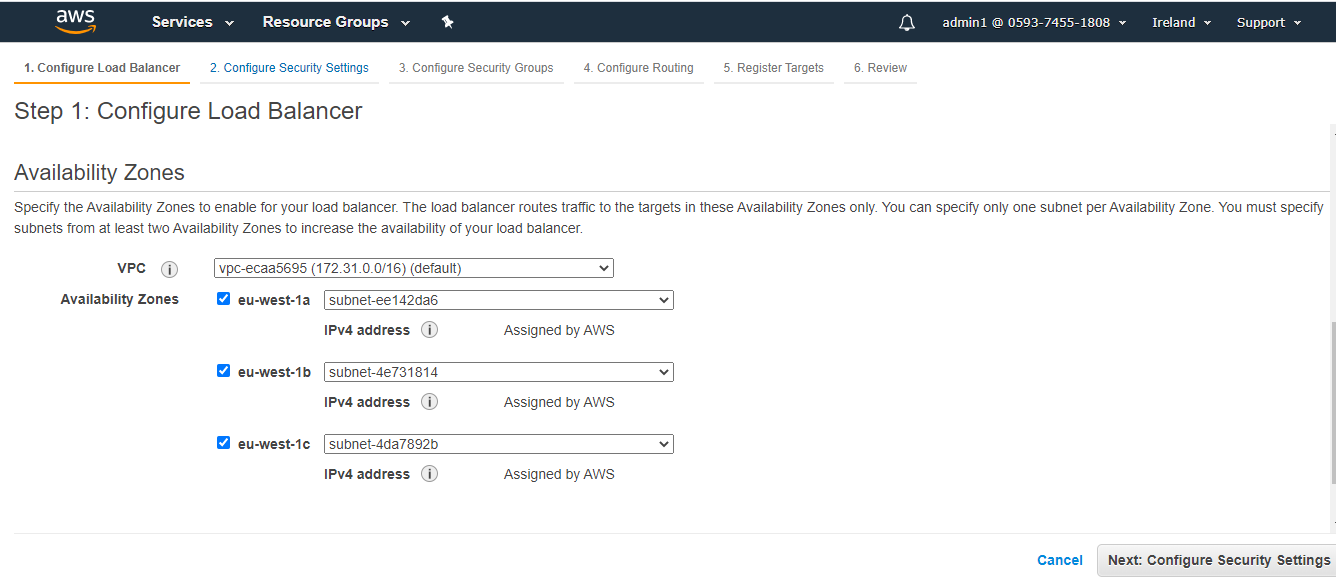
Use healthy.html file at EC2 instance to perform health check of ELBs.



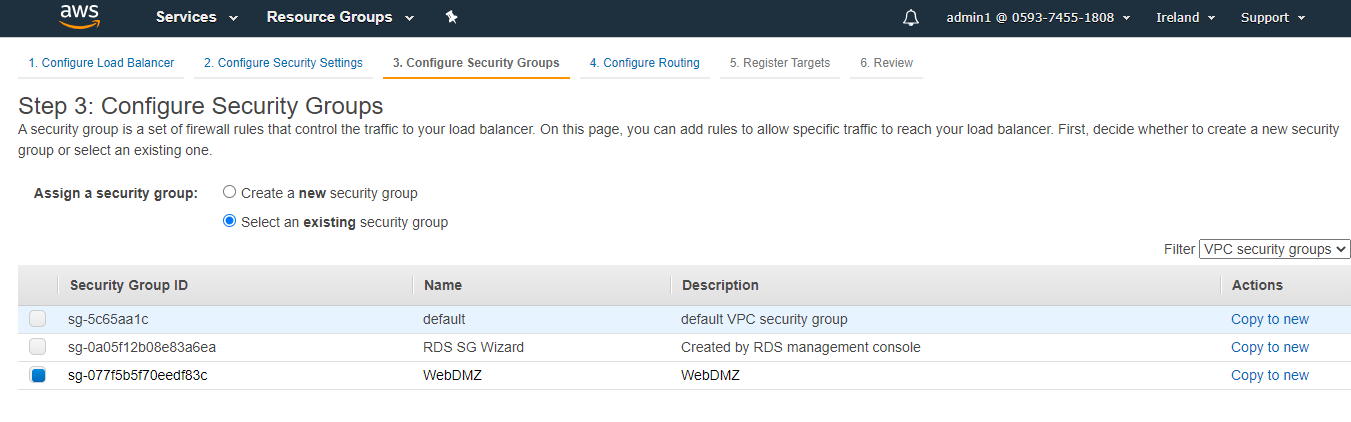
Create an application load balancer in same region Ireland



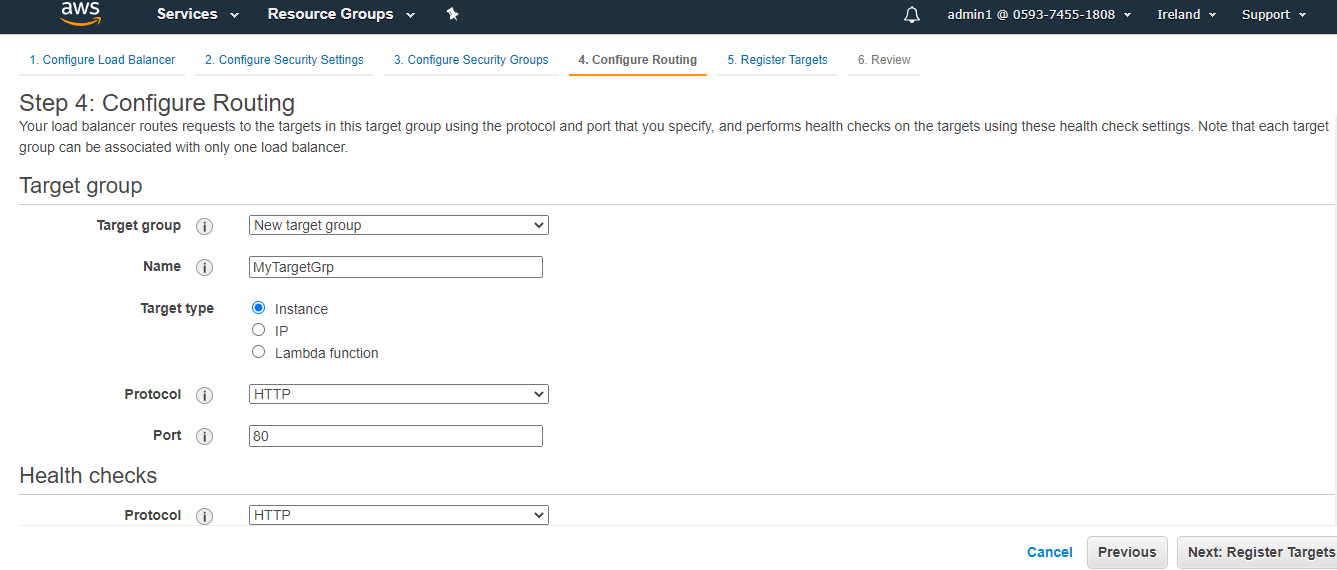
Add all the AZs available in the selected region.



Attach **WebDMZ**  security group.

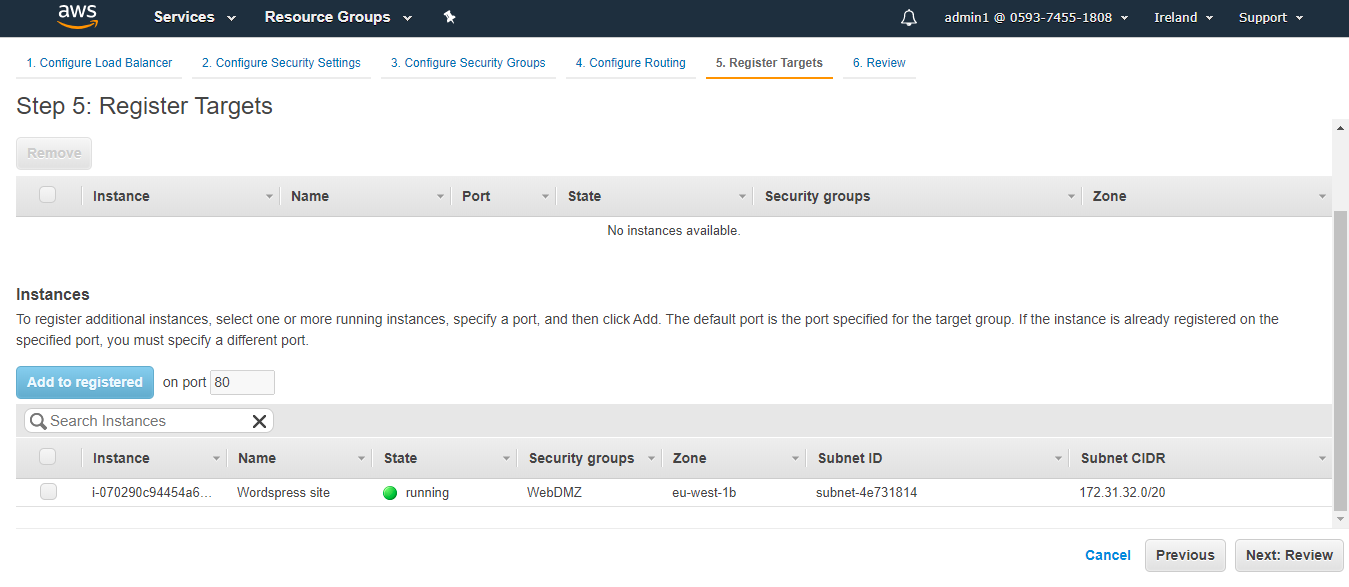


Create target group to add target EC2 instances into that.



Add path to the health check file i.e /healthy.html

Now, register the instances into the target grp

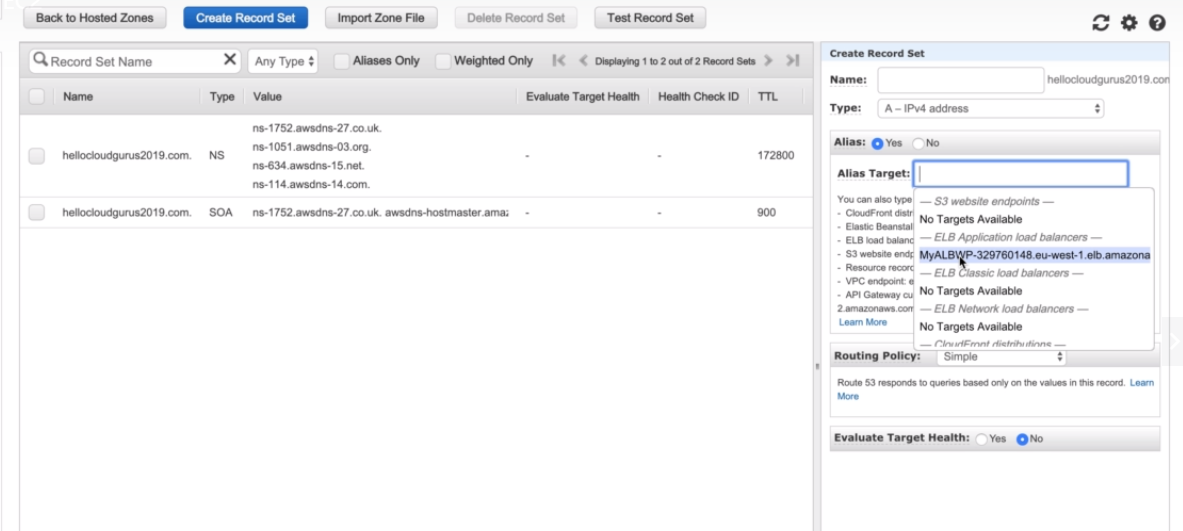


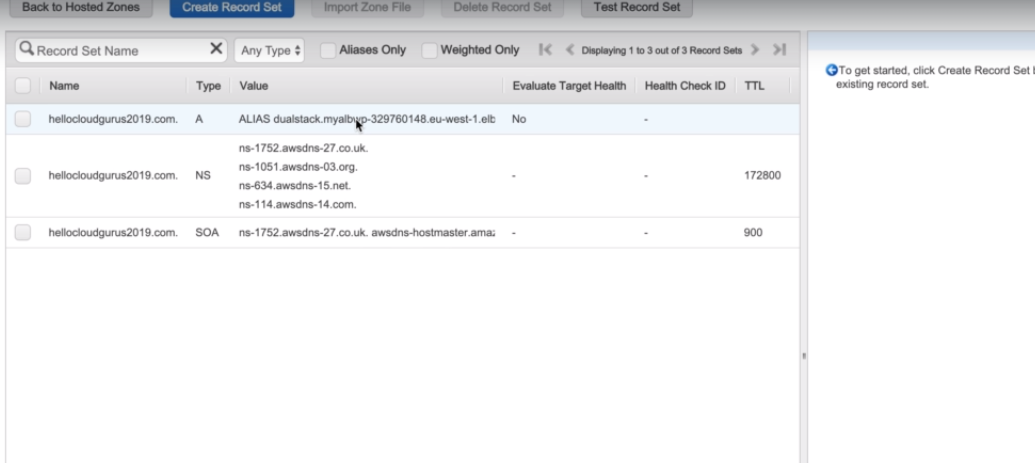
After the ALB is up, we can see the status of the target as healthy.

**11. Setup Route53 to distribute the read and write traffic.**

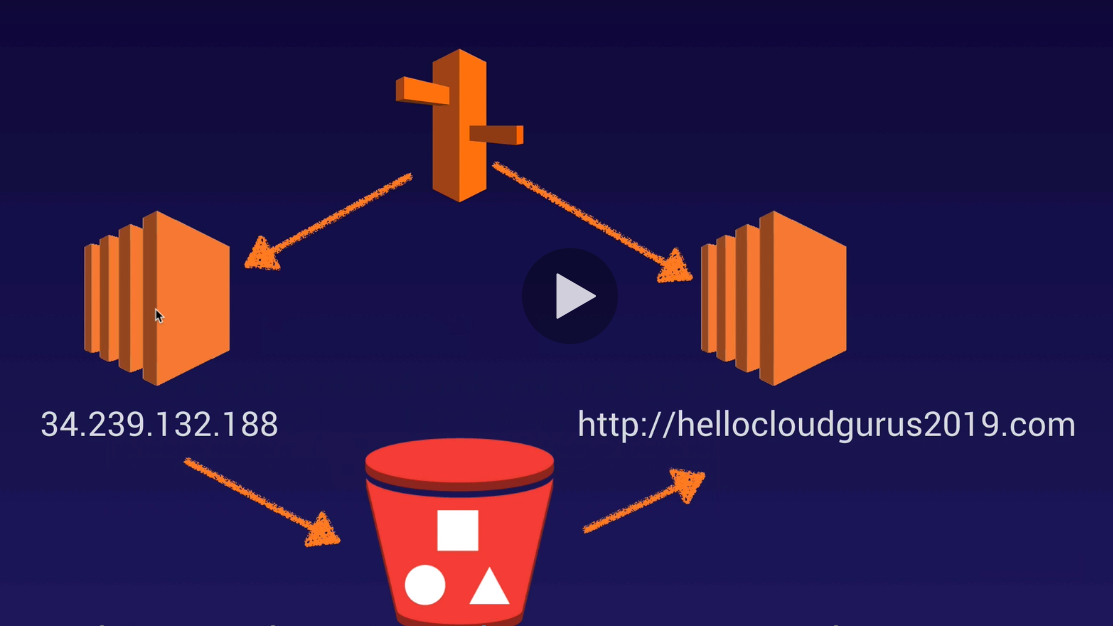
Next, route53 can be created and we can register a domain name for the reader node (http://hellocloudgruru2019.com). Add an alias record (A record) and selec the public IP of the ALB to the DNS created.

DNS name created like http://hellocloudgruru2019.com which will point to the ALB created and then it will serve the WordPress site hosted at EC2 instance(which is one of the member of Target group of Application load balancer) and the images will be served from the CloudFront organization.





To make the **application resilience**, we will follow the below steps



**Read Request-> Route53-> Load Balancer->Autoscaling group->Ec2 reader nodes (Each Reader Node will pool S3 bucket every minute to serve the request with the latest data)**

**Write Request-> Route53-> Ec2 writer node**

34.239.132.188 will be writer node, every write request will be forwarded to this EC2 instance by the Route53 on top of Auto-scaling group and this node will constantly sync the code with the s3 code bucket.

Other node <http://hellocloudgurus2019> will be reader node, every read request will be redirected to this EC2 instance by the route53. This node will constantly sync and copy data from the s3 code bucket where writer node is writing to maintain consistency. Whenever a request come from user it will be routed by route53 to the load balancer which will have an autoscale group and it will be redirected to DNS <http://hellocloudgurus2019> reader node, which is polling the S3 bucket every minute.

**Steps**:

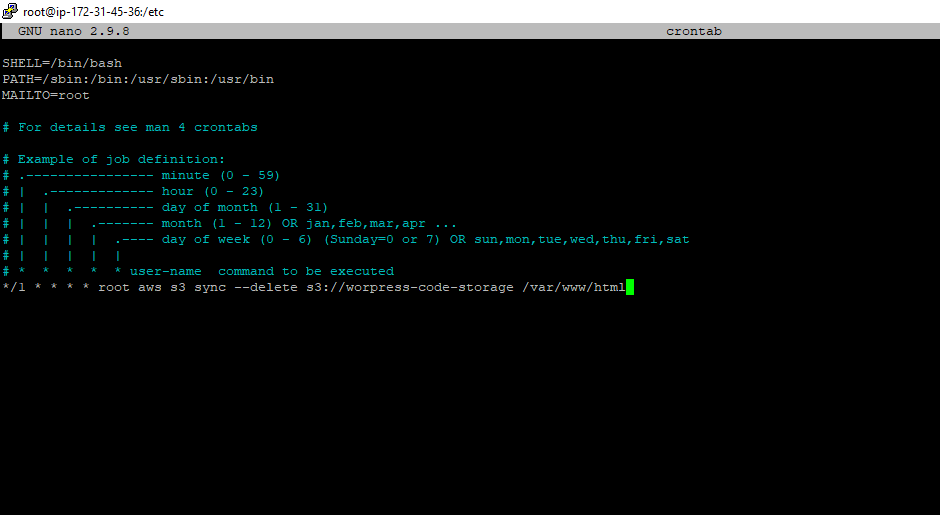
**Step1**. Write a cron in Writer node to copy s3 code bucket data and sync in the /var/www/html dir. So, that any changes made in S3 code bucket will be reflected in the EC2 instance. This is done so that we can make an image from this EC2 instance and launch a new reader node that will constantly pool and copy changes from S3 bucket into itself.

Go to ***cd /ect***

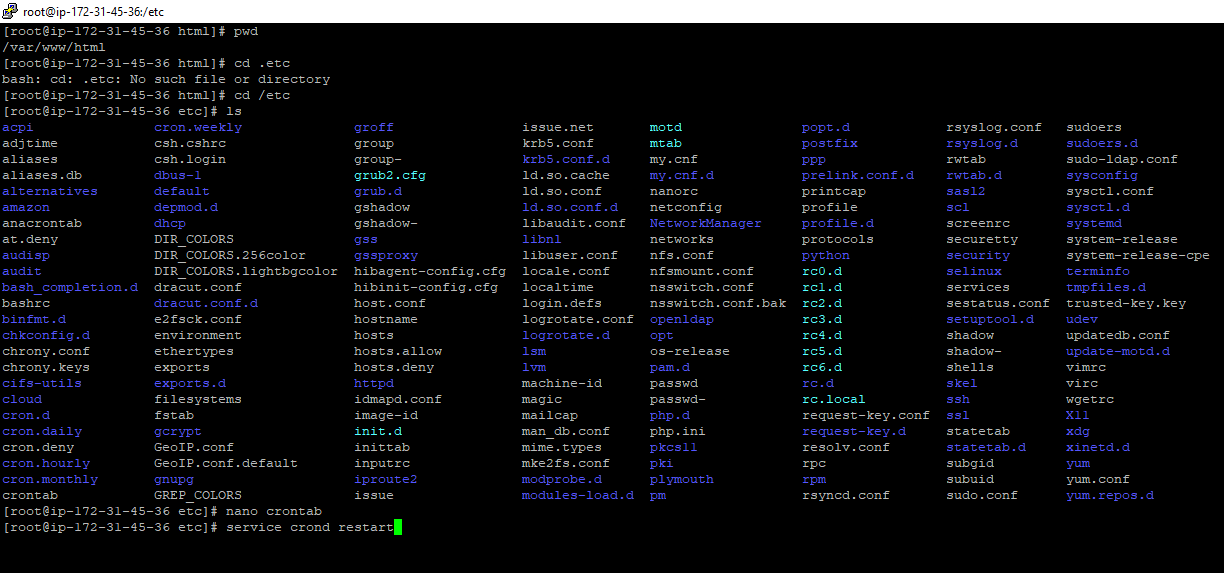
***nano crontab***

***\*/1 \* \* \* \* root aws s3 sync –delete s3://wordpress-code-storage /var/www/html***

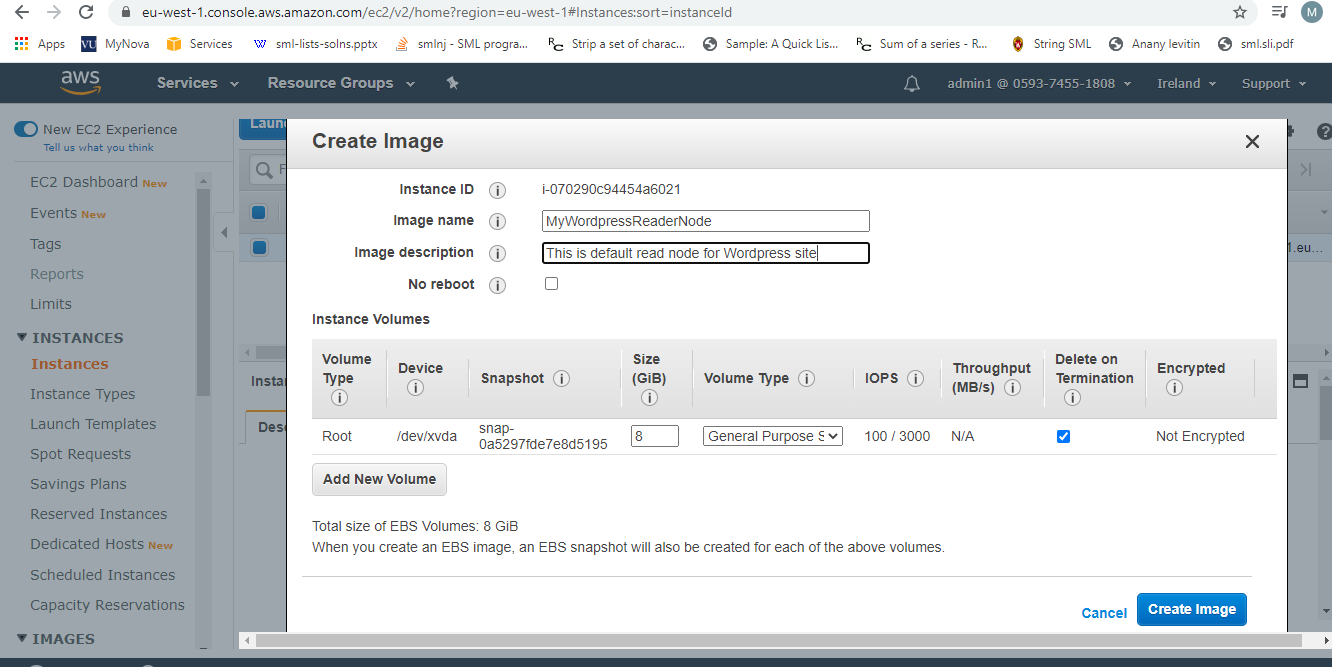
-- delete : allow the exact copy of s3 bucket to be copied into instance and delete the rest of the unmatched file at instance. So, if we delete any file from s3 it will delete them from reader instance too.



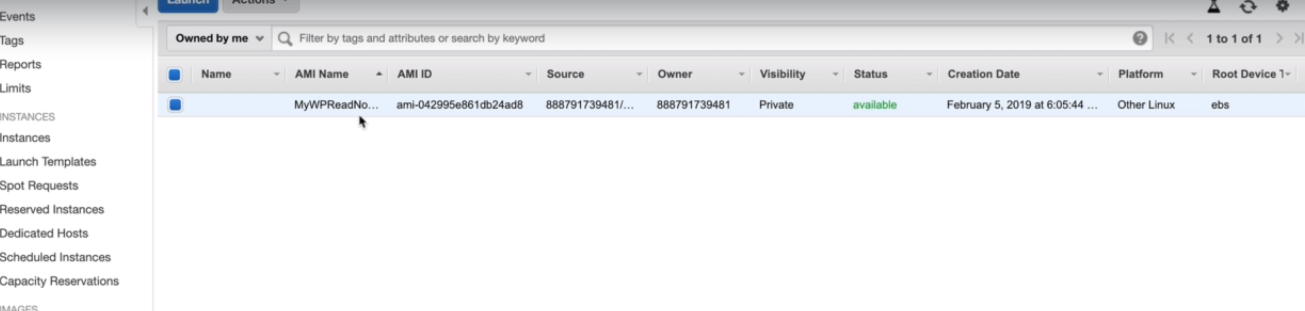
Start the cron : ***service crond restart***



**Step2**: Create the image of EC2 writer instance with name “**MyWPReaderNode**” which will act as Reader Node:



This will reboot the Writer Ec2 instance and wait for it to become available. See, below the created image.



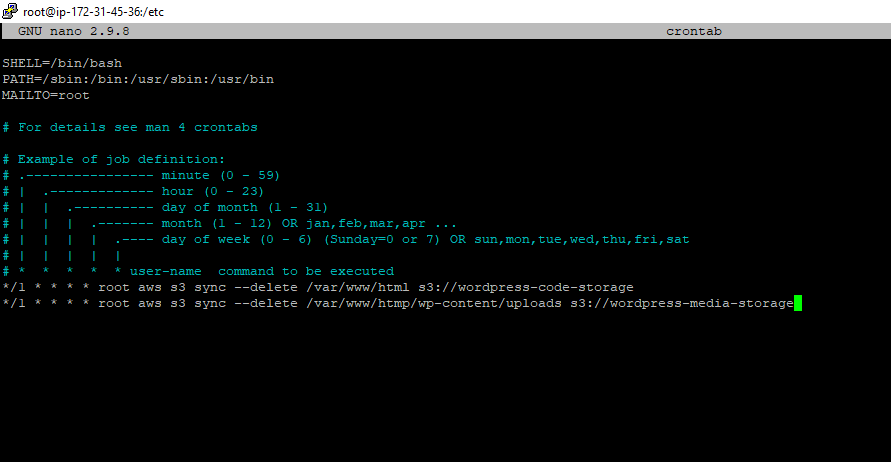
**Step 3**: Go back to Writer node Ec2 instance and update its cron to , copy and sync data into S3 bucket, and media into media s3 bucket: (or copy from file **EC2 Writer Node Cron Bootstrap.sh**)

This is done, so that the Writer Ec2 instance will keep the S3 bucket updated with every modification. (S3 bucket can be used as replica of EC2 writer instance in case of failover)

***\*/1 \* \* \* \* root aws s3 sync --delete /var/www/html s3://wordpress-code-storage***

***\*/1 \* \* \* \* root aws s3 sync --delete /var/www/html/wp-content/uploads s3://wordpress-media-storage***

This cron will copy the data from writer node to respective code and media bucket every minute of every hour of every day or very month of every year.



**Step4**: Put the image AMI of the EC2 instance created into the autoscaling grp, that will sit behind the route53.

**Create a launch configuration**-> select AMI “**My Ami**” ,select MyWordpressAMI (the image of writer EC2 we created).

Select instance type “t2.micro”

Select IAM role to access s3 , select the one we created.

Select the Security Group: “WebDMZ”

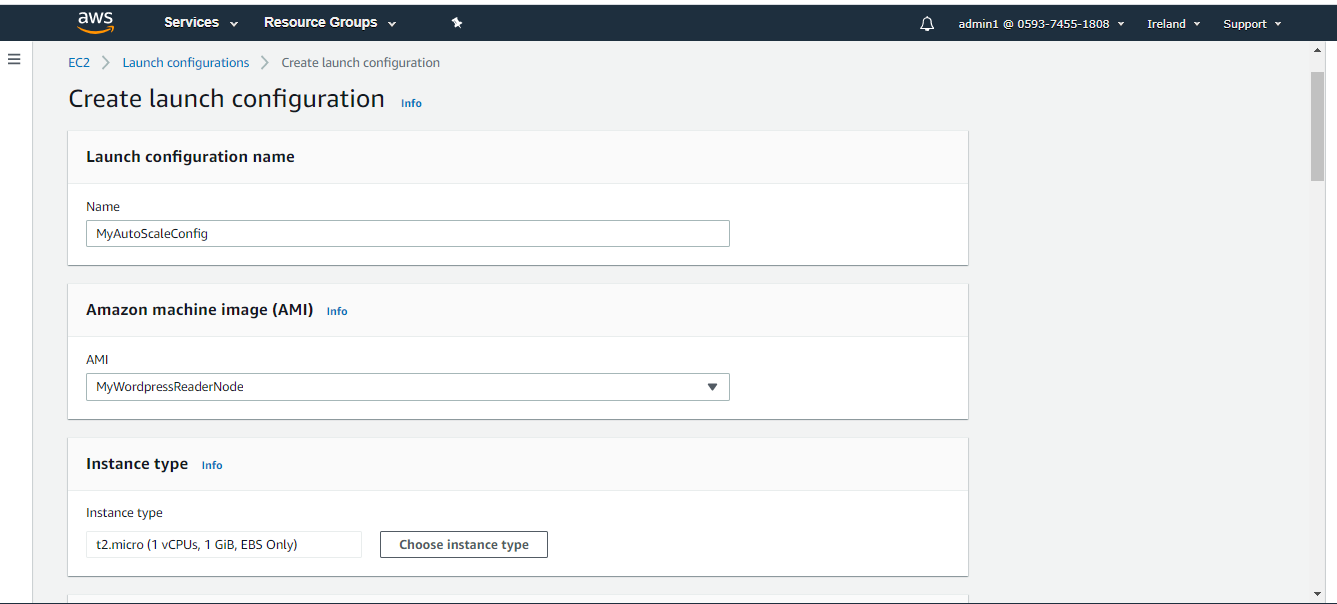
Add the bootstrap userdata: (this will sync the s3 bucket to our reader node) or copy from file : **EC2 Reader Node Cron Bootstrap.sh**

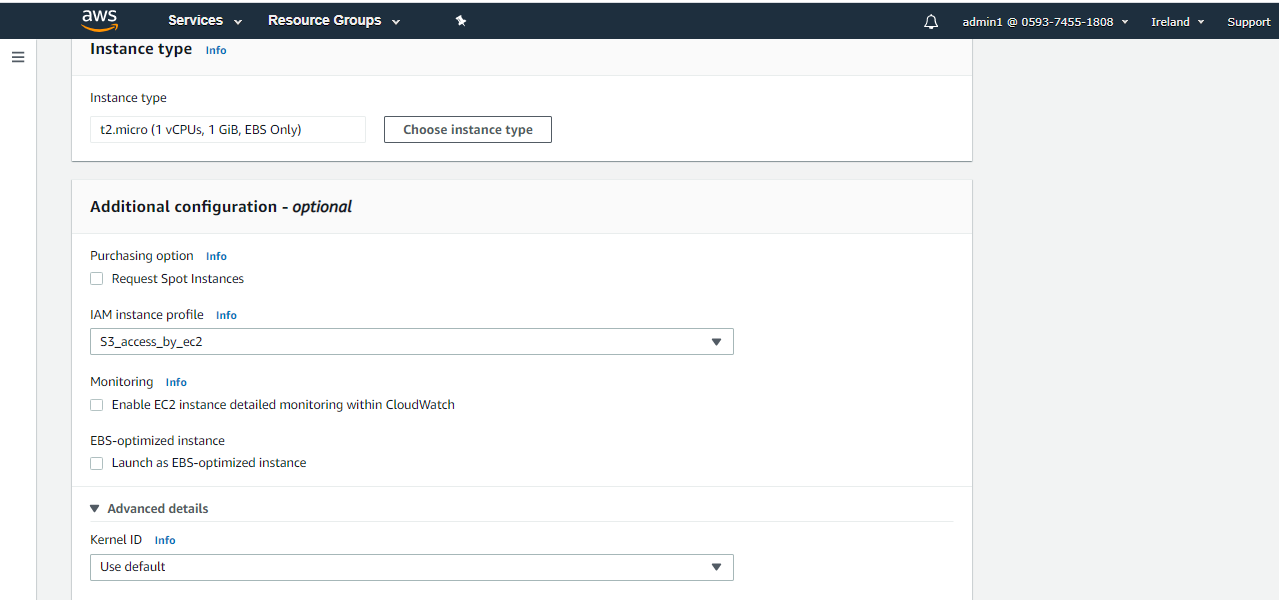
***#!/bin/bash***

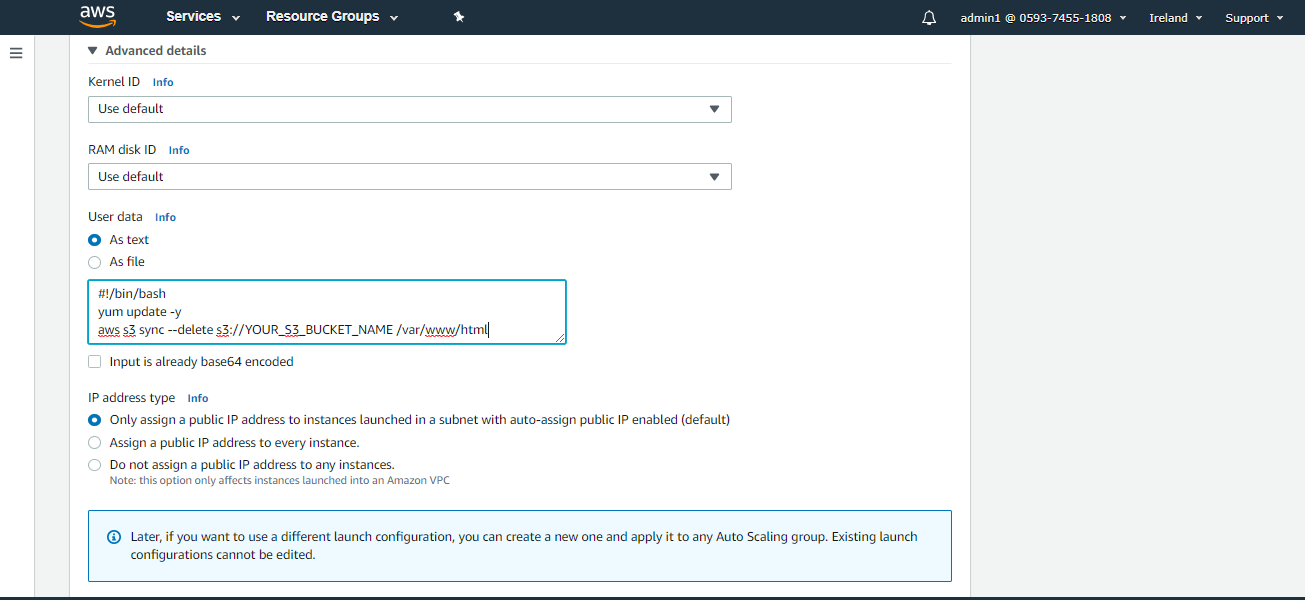
***yum update -y***

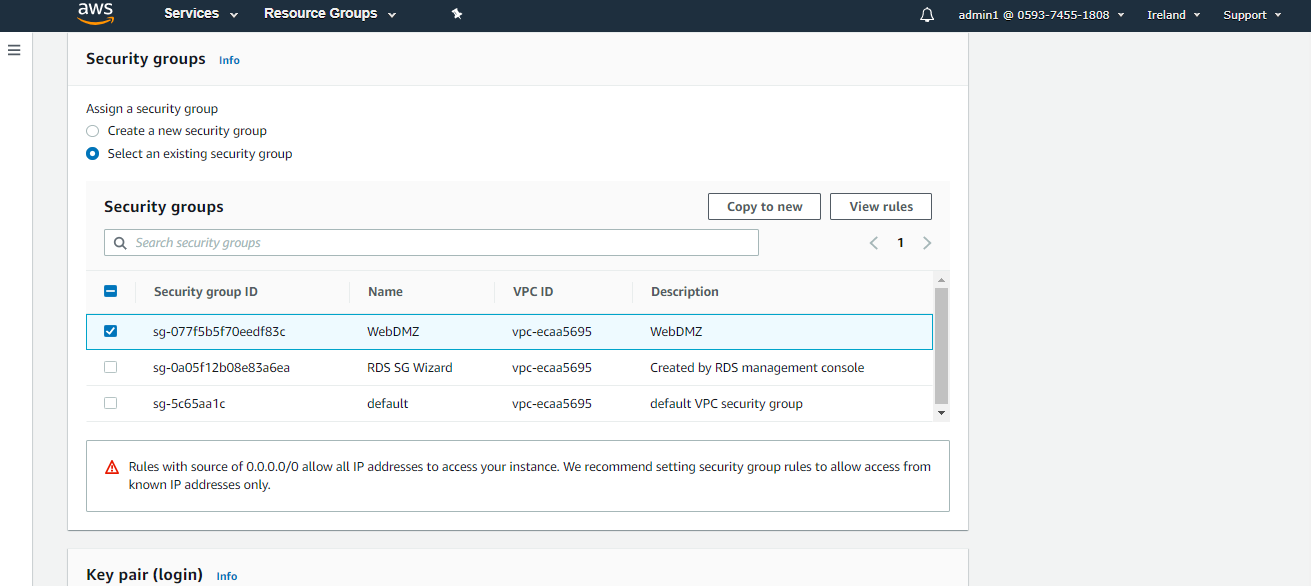
***aws s3 sync --delete s3://wordpress-code-storage /var/www/html***

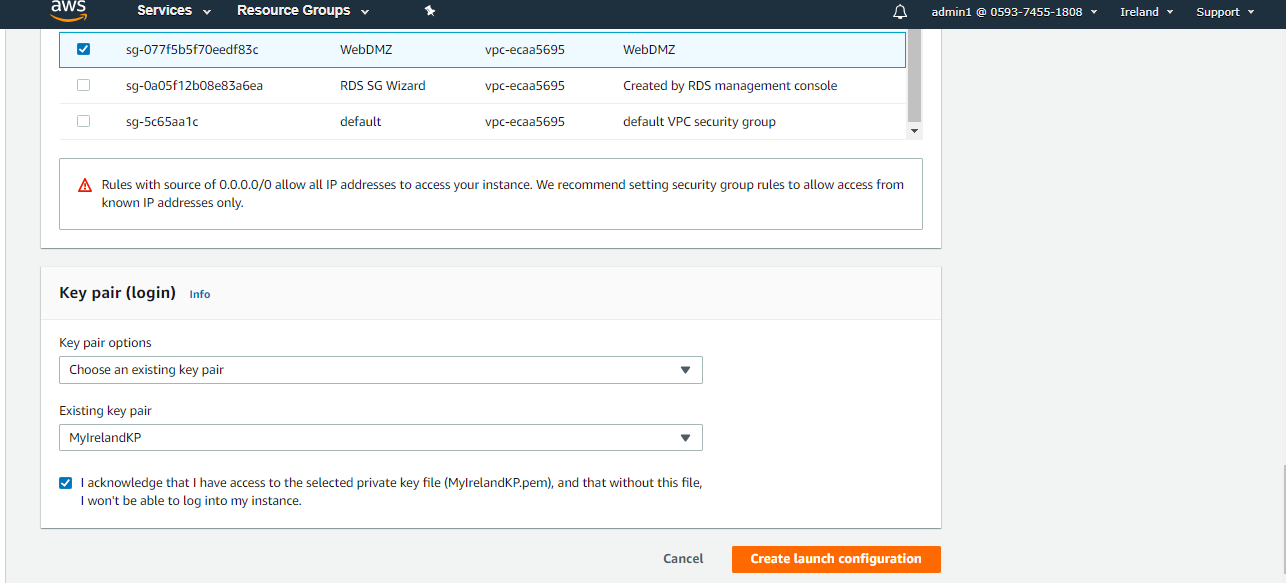
And hit create launch configuration.

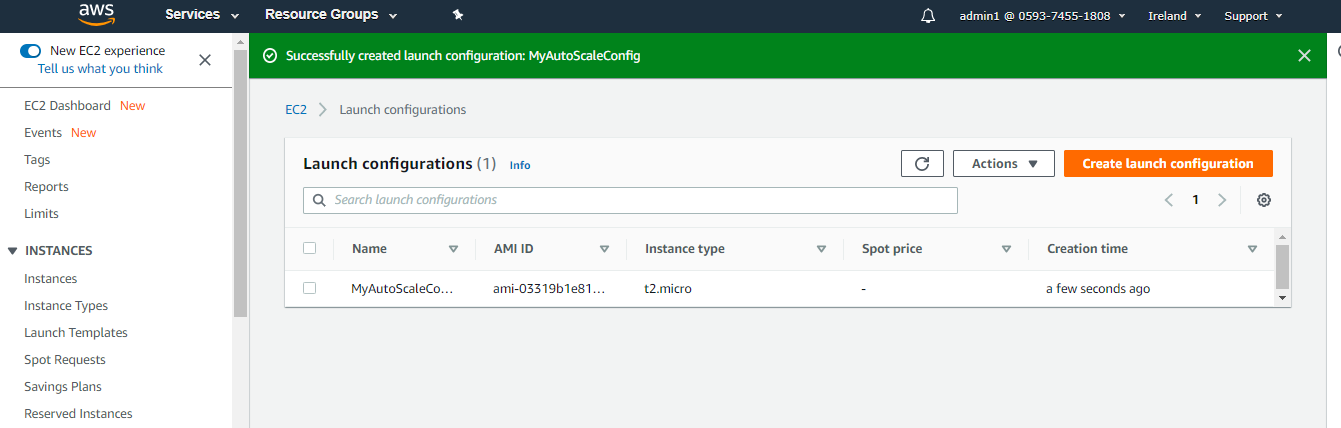








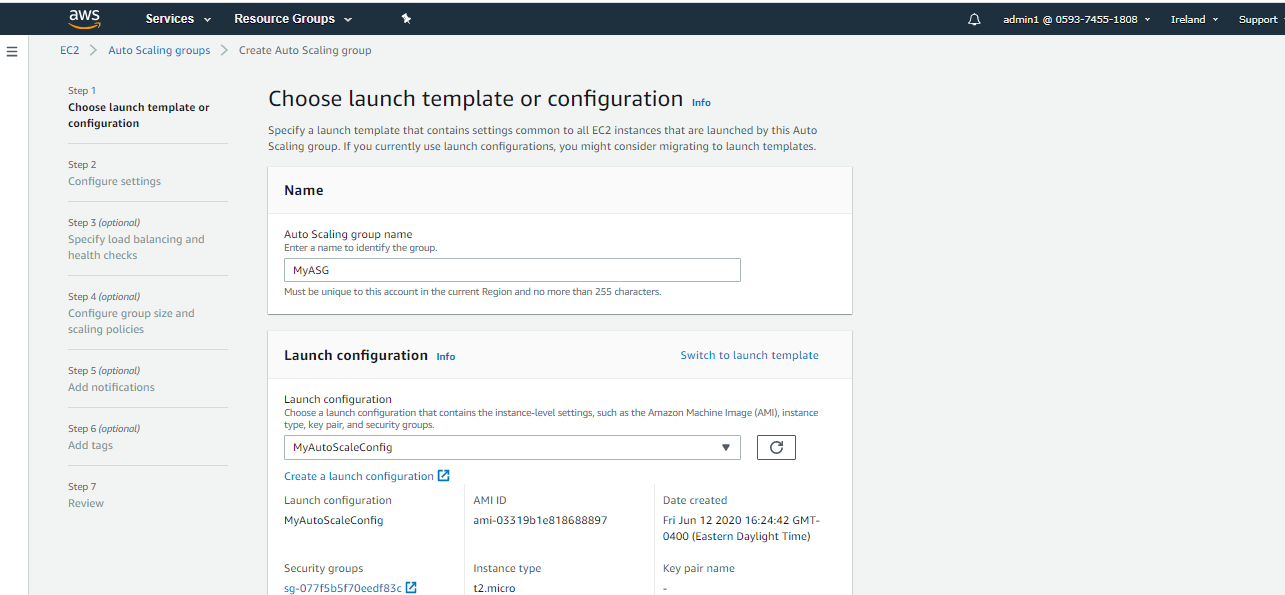


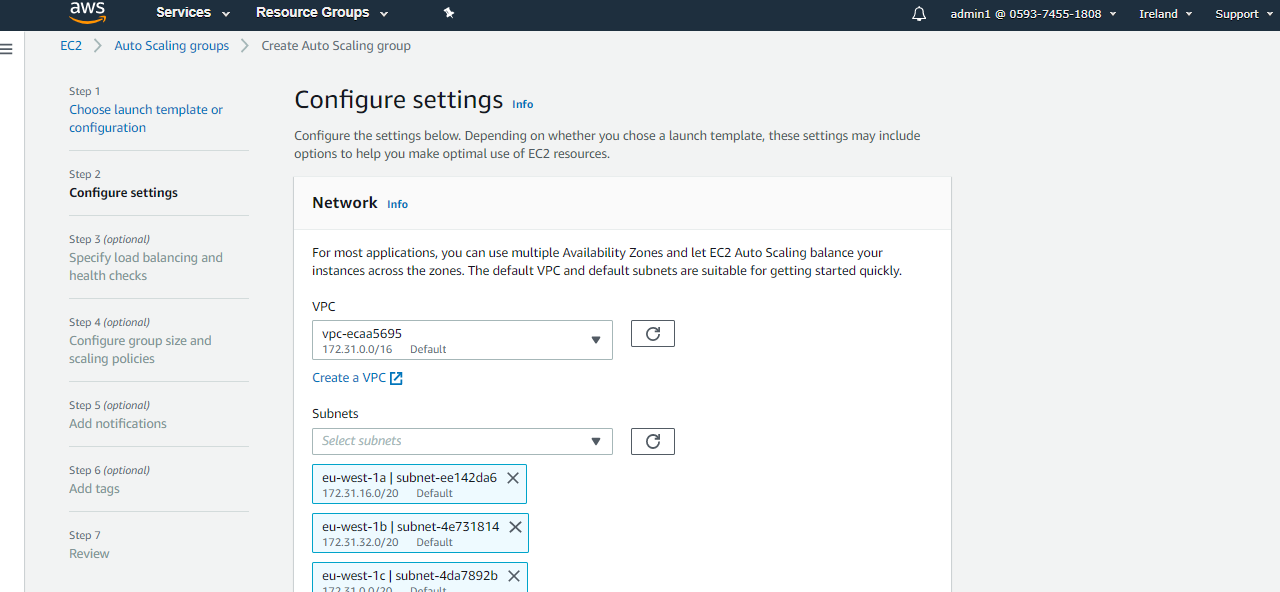


The launch config is created successfully.

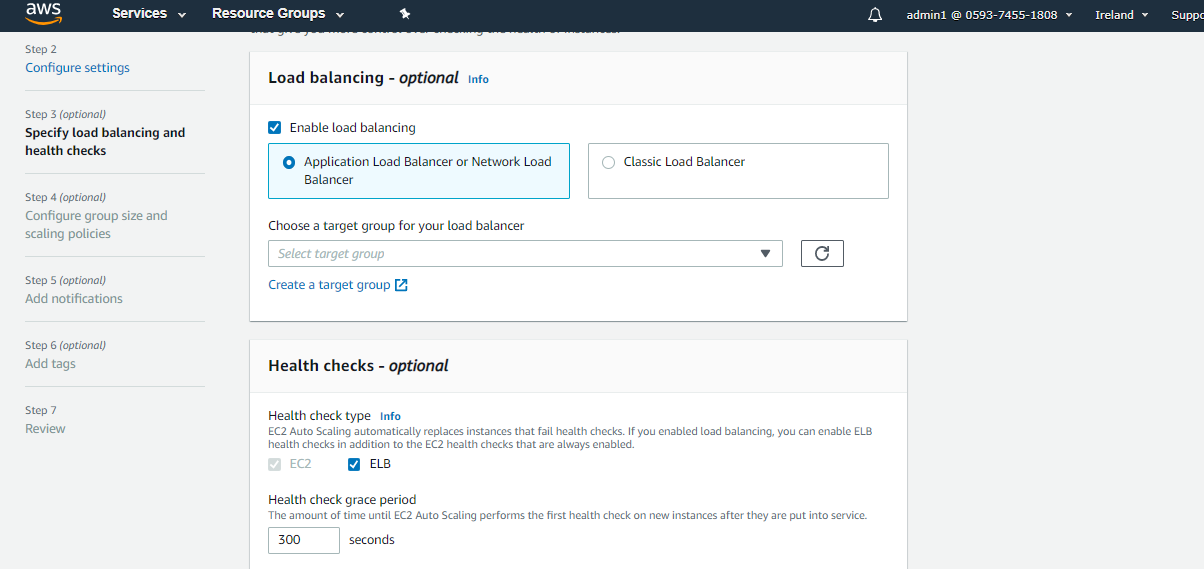
**Step 5: Create Autoscaling group Goto** -> Autoscaling Group in EC2 dashboard.

Select the above created Launch config. And include all subnets, choose default VPC.





Select the target group we created in the Load Balancer.



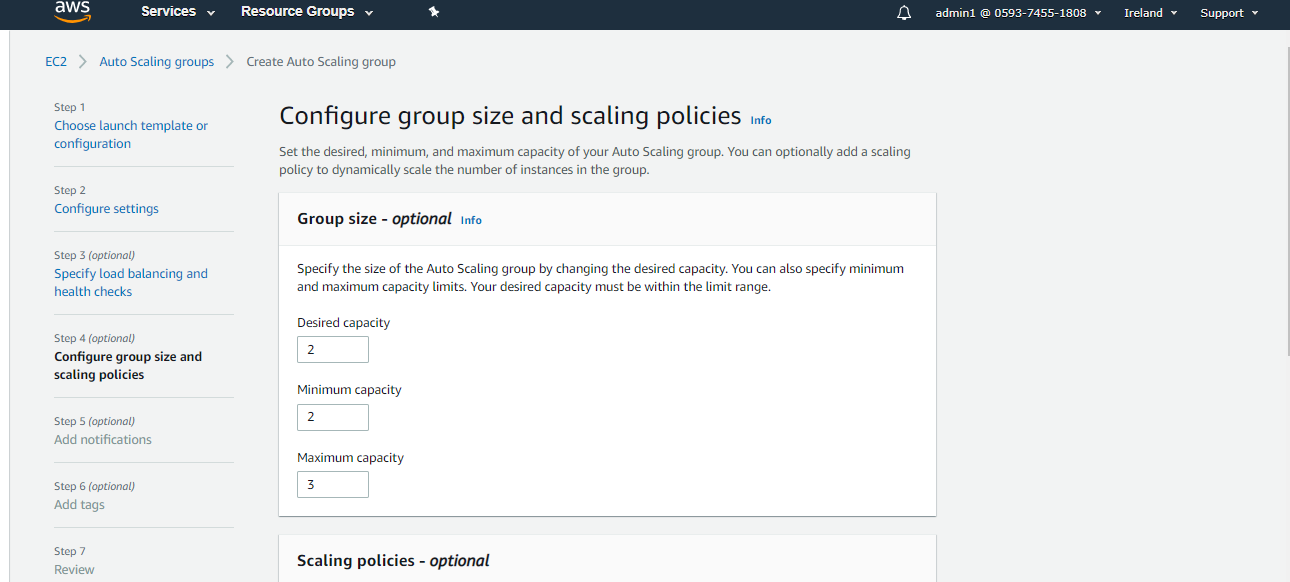
Next, we can add scaling policies, based on the condition/policy the scaling group will scale out(Add more reader Nodes) the EC2 instances. Or leave it default, i.e keep it as its initial size.

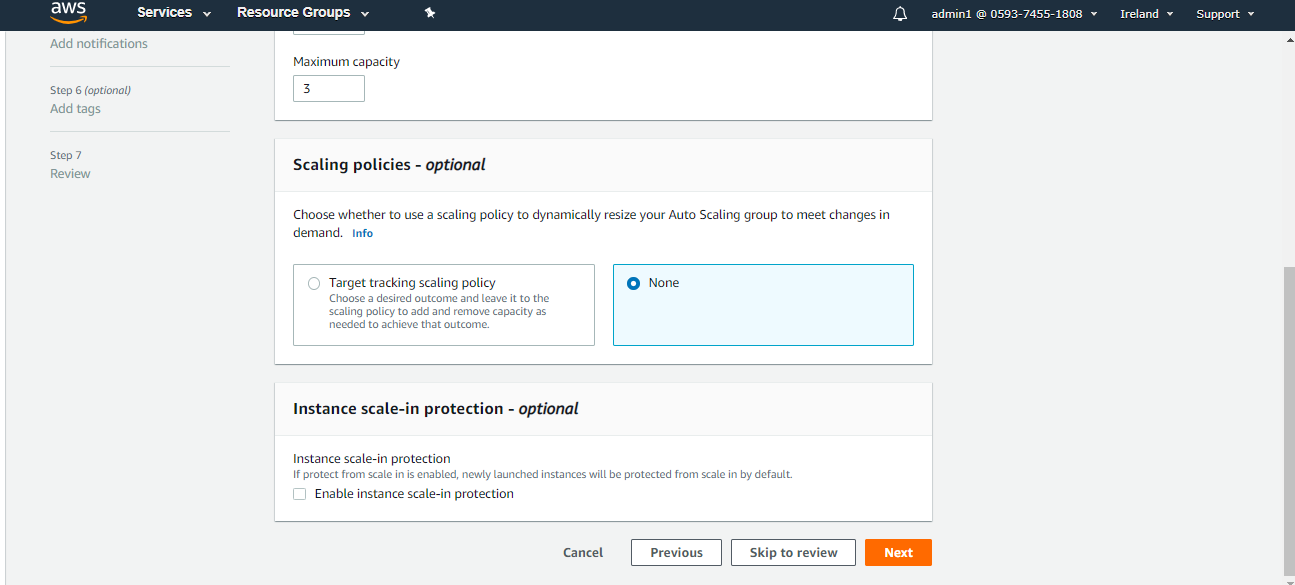
Like add a policy to check if the CPU utilization > 90% then scale out .

Min capacity = 2 ins. (This will create 2 Reader nodes)

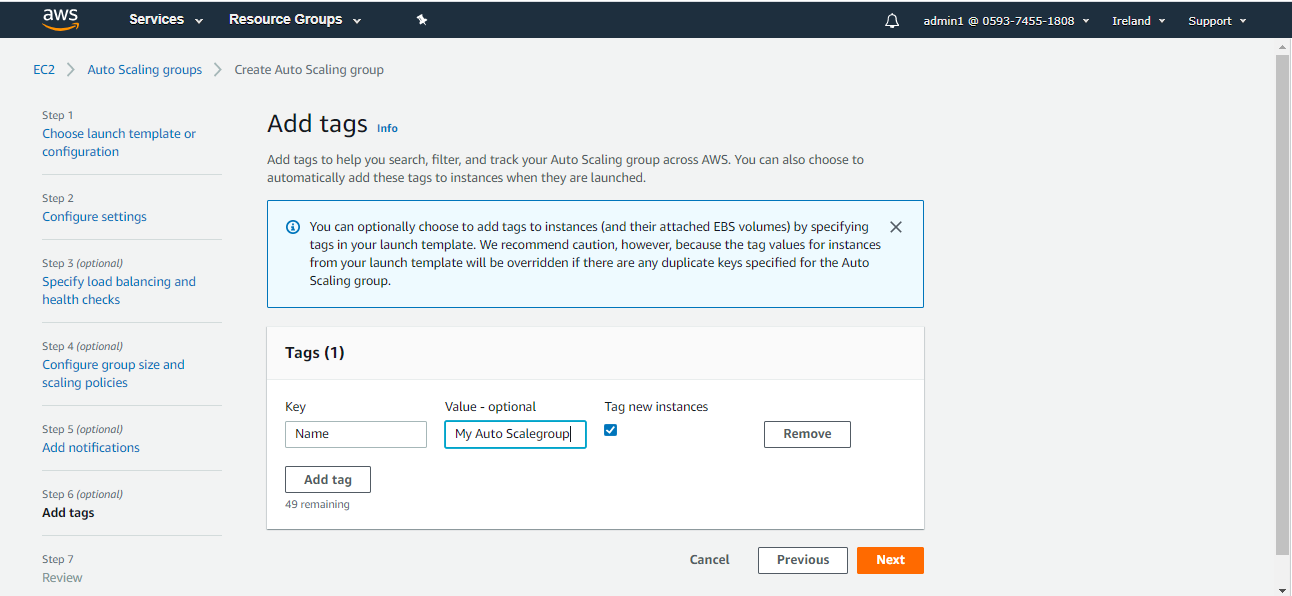
Desired capacity = 2 ins.

Max capacity = 3 ins.

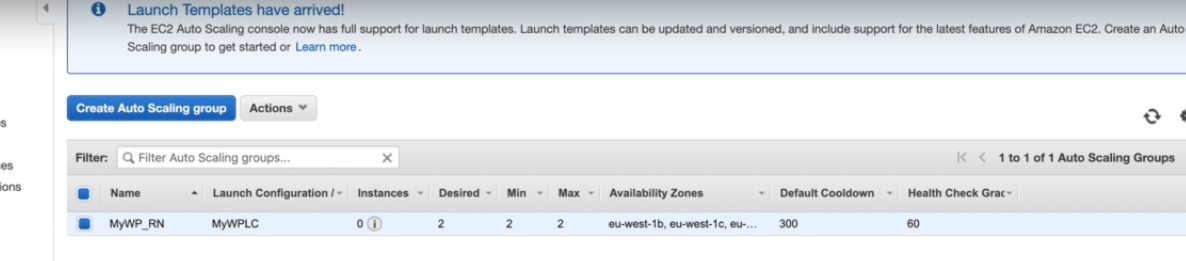




Next Add tags.



And click next to review and create group.



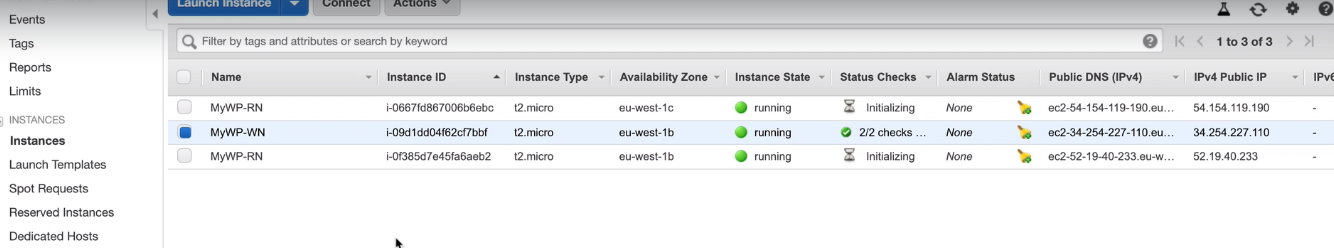
Now go the target group we created and remove the **Write node instance** because we want the Load balancer to distribute traffic only for the read traffic (Reader Nodes).

Goto EC2 dashboard, there will be 3 EC2 insatnces ( notice the Reader nodes are in Diff AZs):

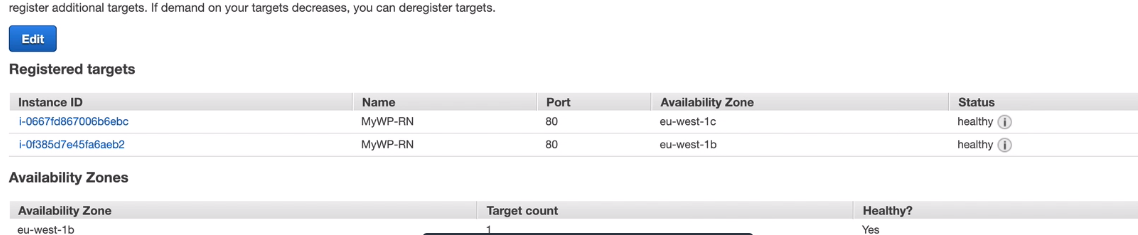
**1**. **Writer node – name: MyWP-WN**

**2. Reader node1 – name: MyWP-RN**

**3. Reader node2- name: MyWP-RN both reader nodes will be in diff AZs**.

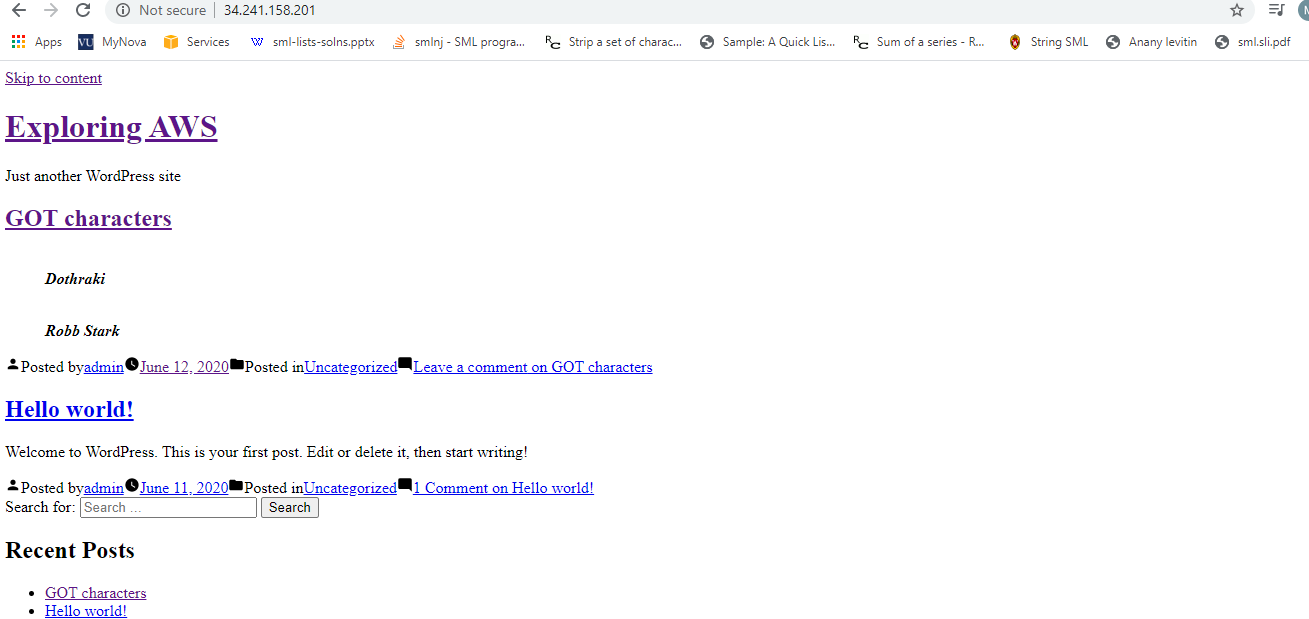


Check the healthy status of Reader nodes in Target group.



So, we will test the scenarios:

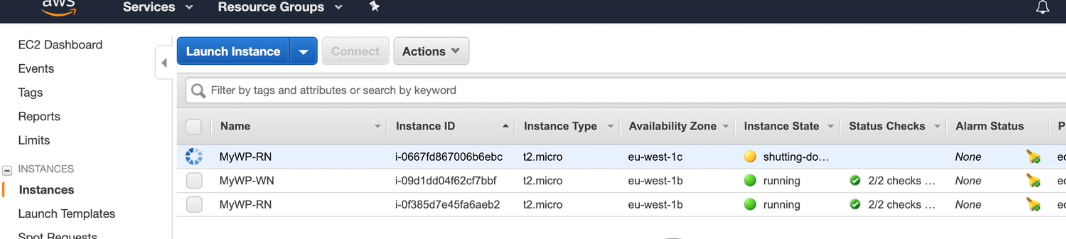
The public IP address of writer EC2 instance will be the admin

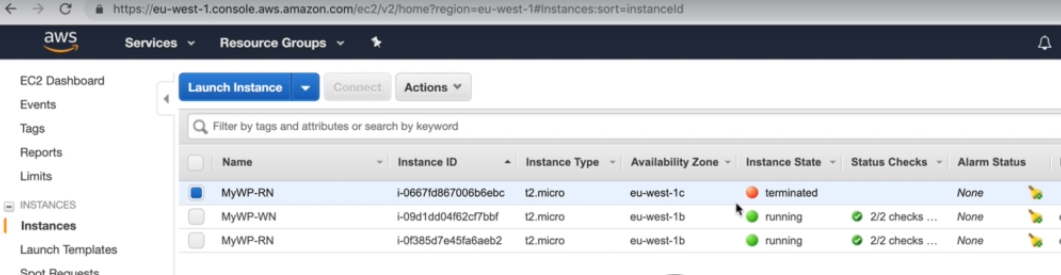


And , the reader nodes will direct to the domain name address : <http://hellocloudgurus2019.com>

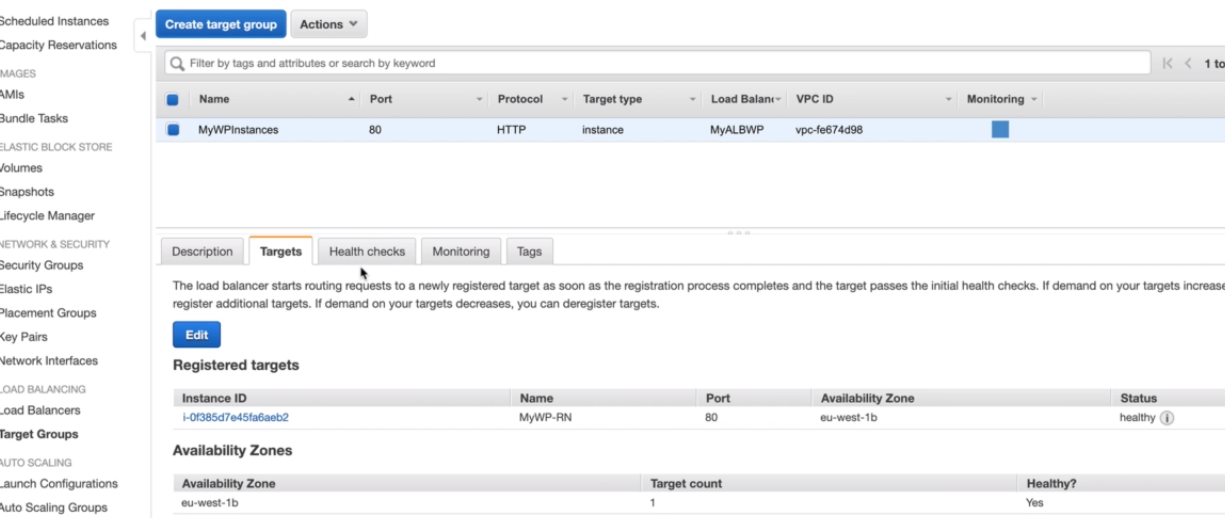
**Simulate High availability behavior (Self healing Reader Nodes)**:

**1. Loss of one AZs. -> go to EC2 dashboard, terminate one of the Reader Node.**

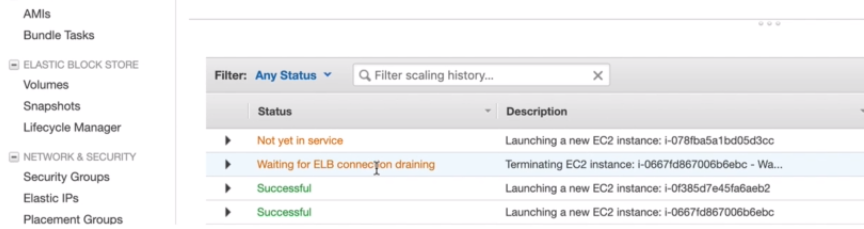




Go to target group and check the health status of nodes. We can see there is only reader nodes now as its not able to reach the /healthy.html of terminated reader node.



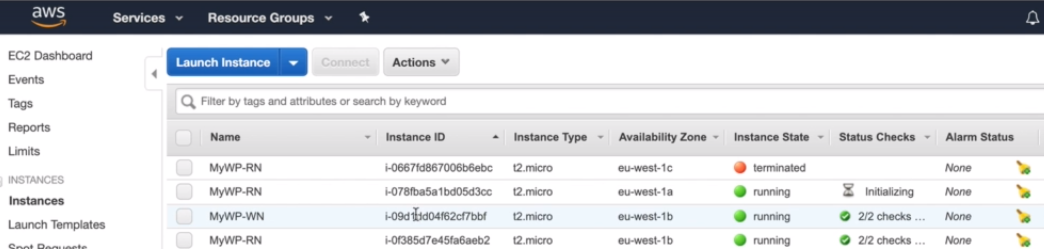
Go to Auto-scaling group and check the Activity history tab, we can see the activity history of instances.



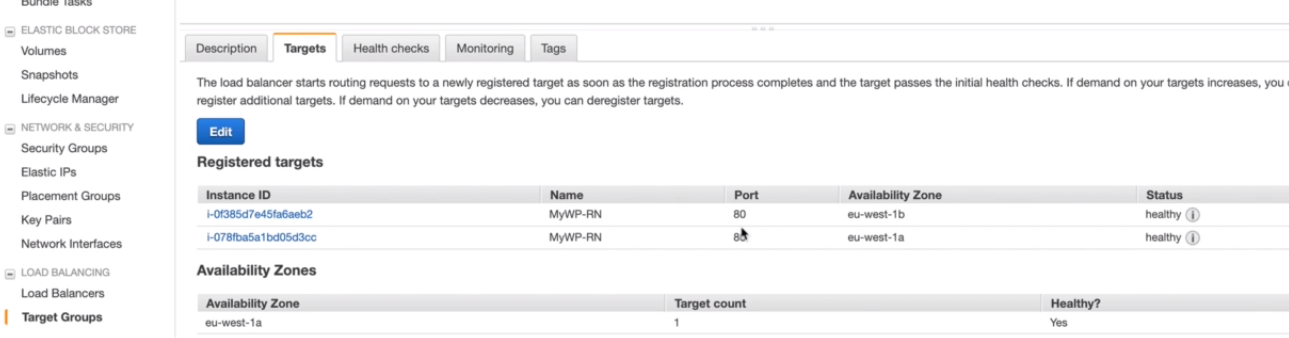
**But if we hit the DNS site , we still get it loaded but with some latency as only one of the reader node is off. The load balancer will redirect all the traffic to only one Reader only, till the time the auto-scalping group make another reader node up.**



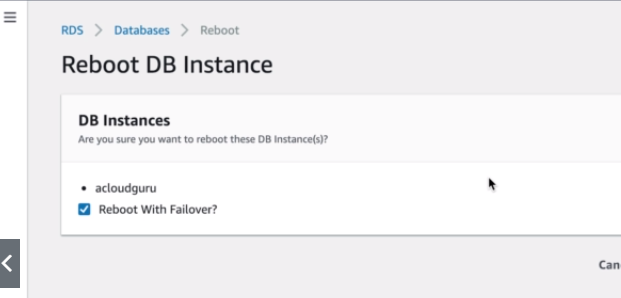
Now , go back to EC2 dashboard to check if auto scale group has launched new instances of reader nodes automatically after the specified timeout to run duplicate instance.

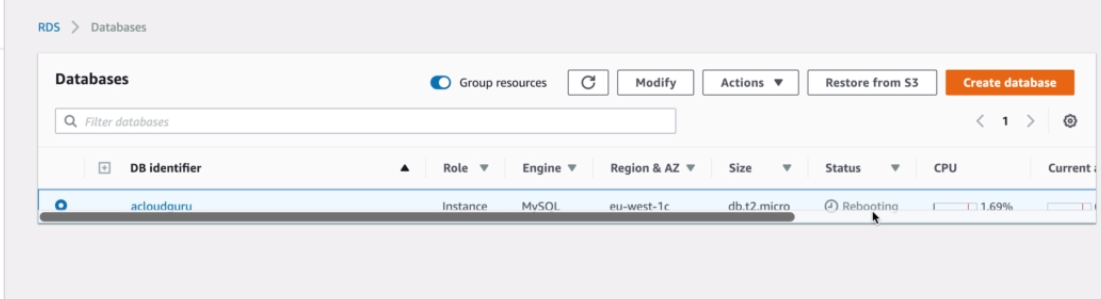


Also, confirm the heath status of newly created Reader Node by Auto-scaling group. Go to-> target group, both nodes will be healthy.

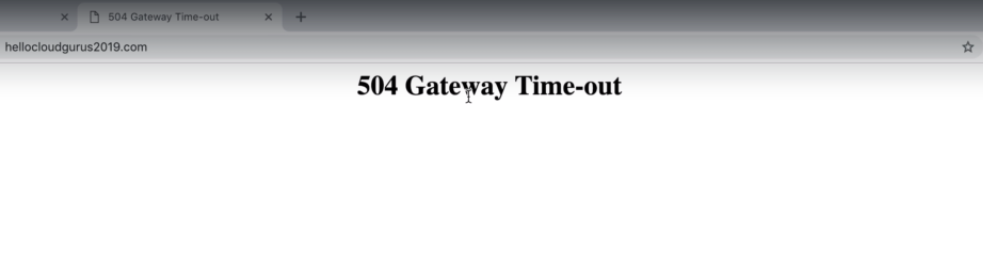


**2. Check how to failover one Az to another for RDS server. Simulate a failure of RDS. RDS can failover to another AZ if we reboot it.**





Now ,if we hit the site it will take few mins to reload, bcz the DB is down n restarting on another AZ.

So, there is issue with the RDS server and its getting up, if we hit it after few seconds site will be up again and RDS will now be in diff AZ.