# HOW TO:

# GUIDE TO GAIN FAMILIARITY WITH AWS

BY NUTAN SONALE

HOW TO:

# GUIDE TO GAIN FAMILIARITY WITH AWS

BY NUTAN SONALE

This short guide aims to give basic exercises that makes one familiar and curious with fundamental functionality of AWS services

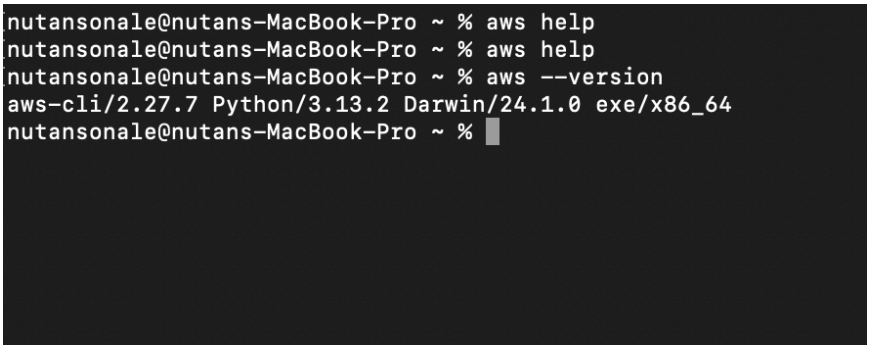
nutan sonale
www.nutansonale.in

# INDEX

# BASICS

Before starting doing anything, one should create an AWS root account. This will give access to was console via web browser. If the user wishes to do perform the tasks from CLI they have to configure their AWS CLI with the user token.

- Install AWS CLI package from CLI Documentation page.
- After following the installation guide for your operating system you should be able to view the CLI version. For CLI options use command "*aws help*"



# 1.

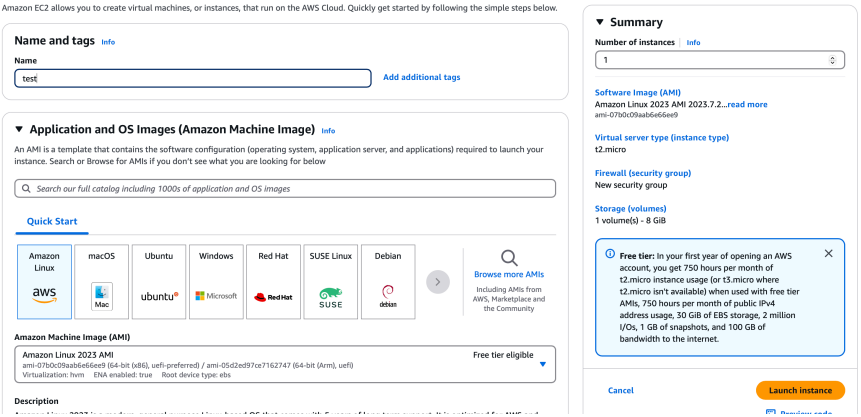
# LAUNCH A LINUX AND WINDOWS SERVER WITH VPC AND ROUTE TABLE

1. At the top of console search bar type EC2 and this will lead to EC2 control panel.

2. In navigation pane select **Instances**.

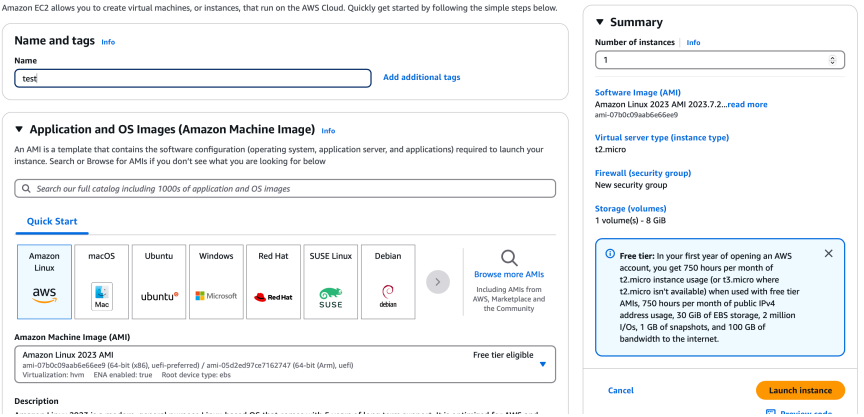
3. choose **Launch Instances**

4. Select OS type for linux select **Amazon linux 2023 AMI** and for windows select **Microsoft Windows server 2022 base**



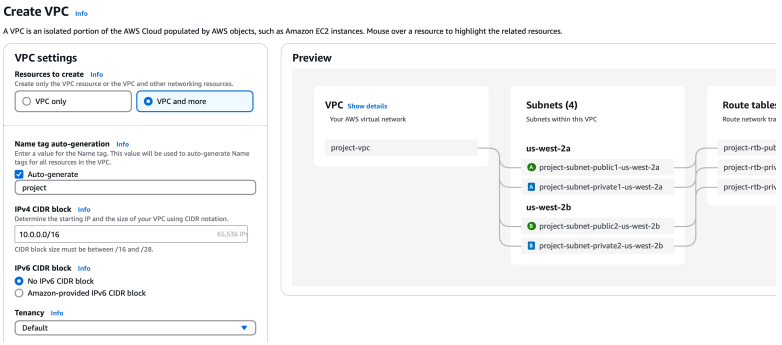

5. Make sure you select t2.micro instance type and it shows free tier eligible.

6. For Linux VM you need to create key pairs for login via SSH. For windows you need to use RDP so you can login using the UI.



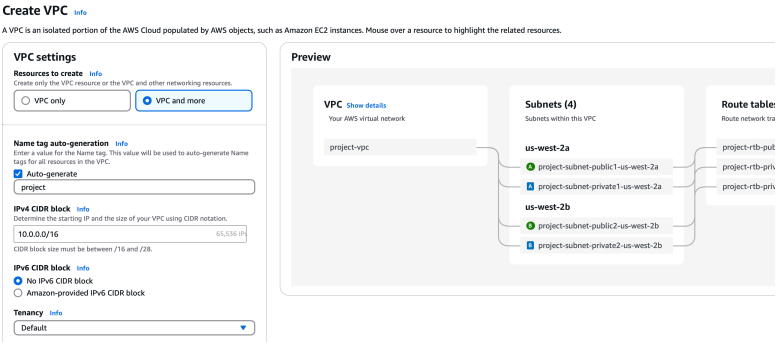

7. For creating a VPC with subnet and its route table, choose VPC in console search. Select **Create VPC.**

8. Under VPC settings sections choose **VPC and more** and for making the VPC available in multiple availability zones you need to select **Number of Availability Zone** as **2,** for Subnets we need 1 **public subnet** and 1 **private subnet.**




9. We need 1 **NAT Gateway** and 1 **Internet Gateway**

10. If we want to access S3 buckets from the VPC we need to enable **VPC Endpoints.**




11. The Subnets will be associated with route tables the public subnet should be routed to internet gateway as target, the private subnet should be routed to NAT gateway. The NAT gateway will be in public subnet all the traffic will be translated/transferred to internet gateway from NAT gateway. The inbound network to private subnet from the internet will be denied.

12. The EC2 instances should be assigned with these VPC configuration

13. After completing all these configuration add SSH and RPD ports to accept traffic from internet(all hosts).

14. After these changes launch the instances, once the instance status shows running you should be able to access the instance via SSH or RDP login.

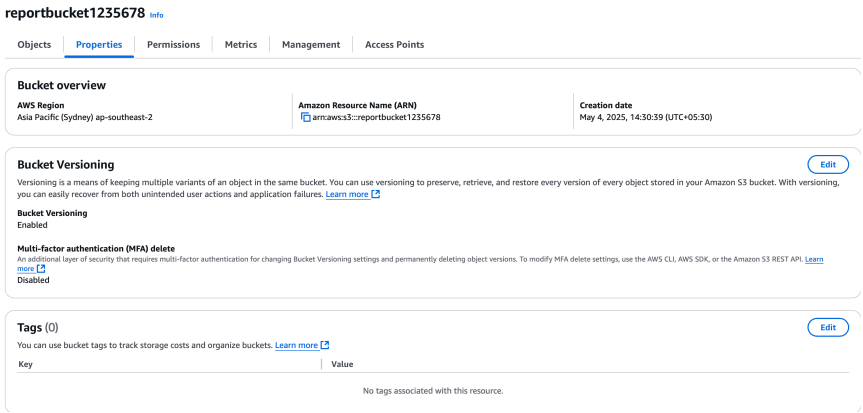
# 2.

# CREATE STORAGE SPACE USING S3

1. Select S3 service in AWS console

2. Choose **Create Bucket.** Bucket names must be between 3 and 63 characters long and consist of only lowercase letters, numbers, or hyphens. The bucket name must be globally unique across all of Amazon S3, regardless of account or region, and cannot be changed after the bucket is created

3. In object ownership section select ACL(Access Control List)



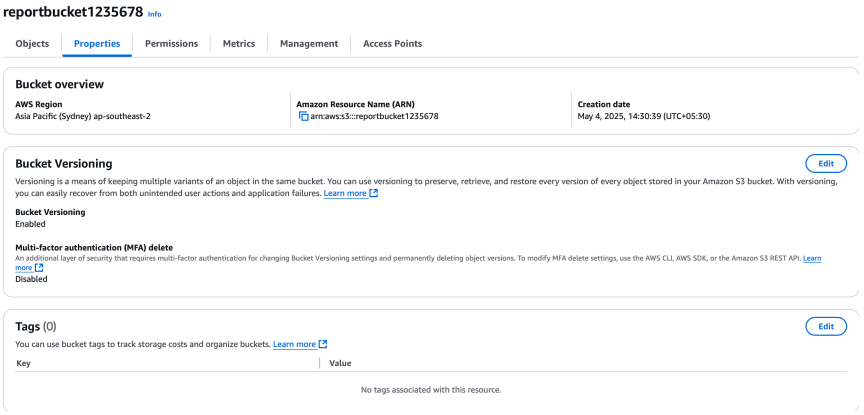

4. Upload an object into the bucket, copy the URL of the object. Try accessing the object from web browser you will receive access denied.

5. To make the object publicly accessible first we have to disable the bucket policy for blocking all public access

6. Next we have to select the object and choose **Action** select **Make public via ACL**.

7. Then we can access the object via the URL without any authentication.

8. To maintain backup of old version of the object in bucket policies we have to enable versioning

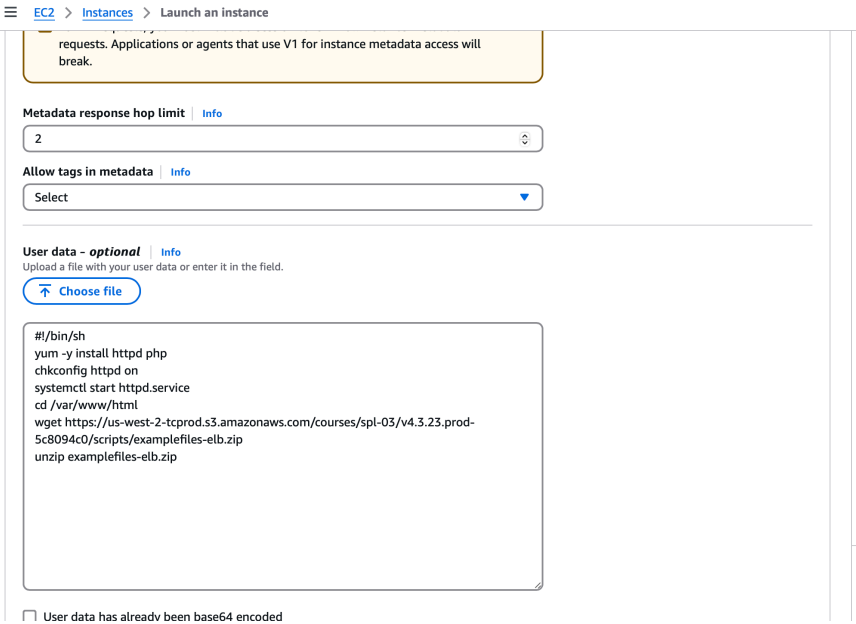


# 3.

# DEMONSTRATE LOAD BALANCER AND ELASTIC IP

1. Create two EC2 instances as shown in chapter 1 with VPC configuration. While creation of EC2 in the field of **user data** in **Advance details** section paste the below code.

```
#!/bin/sh
yum -y install httpd php
chkconfig httpd on
systemctl start httpd.service
cd /var/www/html
wget https://us-west-2-tcprod.s3.amazonaws.com/courses/spl-03/v4.3.23.prod-5c8094c0/scripts/examplefiles-elb.zip
unzip examplefiles-elb.zip
```




2. After launching the EC2 in the property of the instance copy **Public ipv4 DNS** and view the page in the browser. The output of zone and instance ID should be shown.

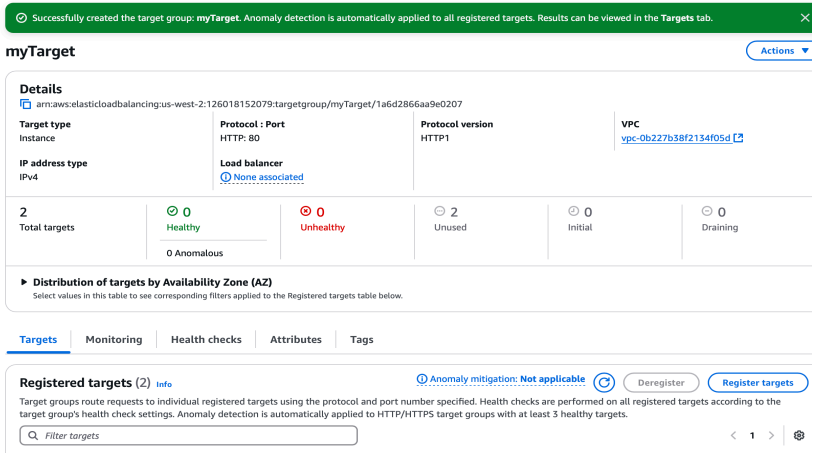

3. In the main page of the EC2 service the navigation pane holds a section for load balancers, select load balancer.

4. Choose **Create load balancer,** this will load a page where we have to select the type of load balancer we want to create, choose **Application load balancer.**

5. Enter the name of the load balancer and select the VPC that was created earlier, make sure the public subnet is selected.

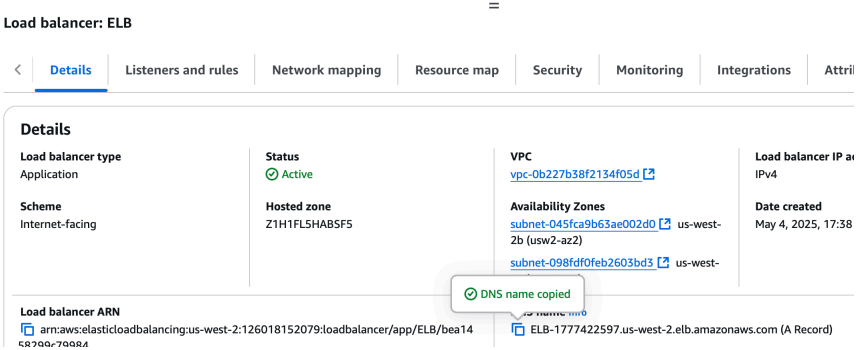
6. Assign a security group where the web port 80 are reachable from any host.

7. In **listener and routing** section choose **create target group,** here enter name to the target group and in the **Register target** select the EC2 instance with the web application running.

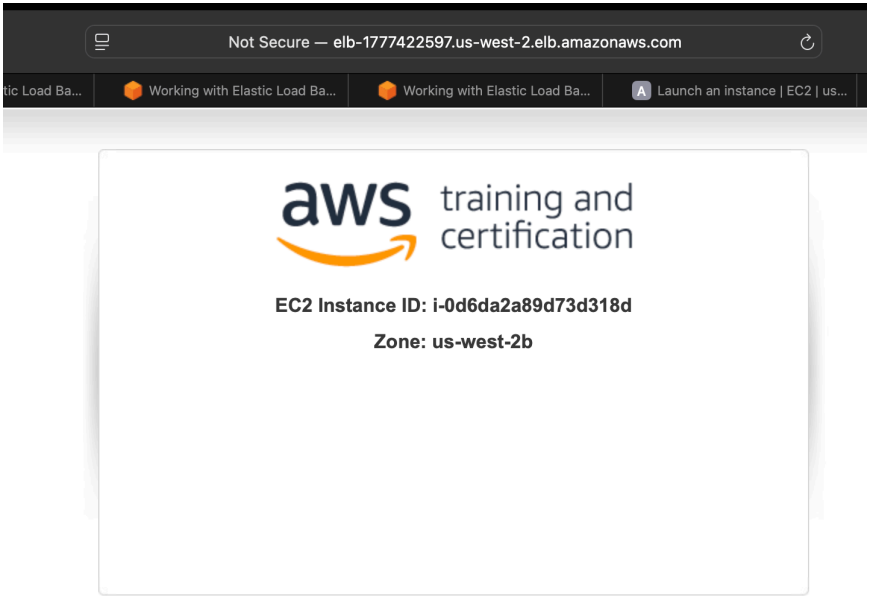


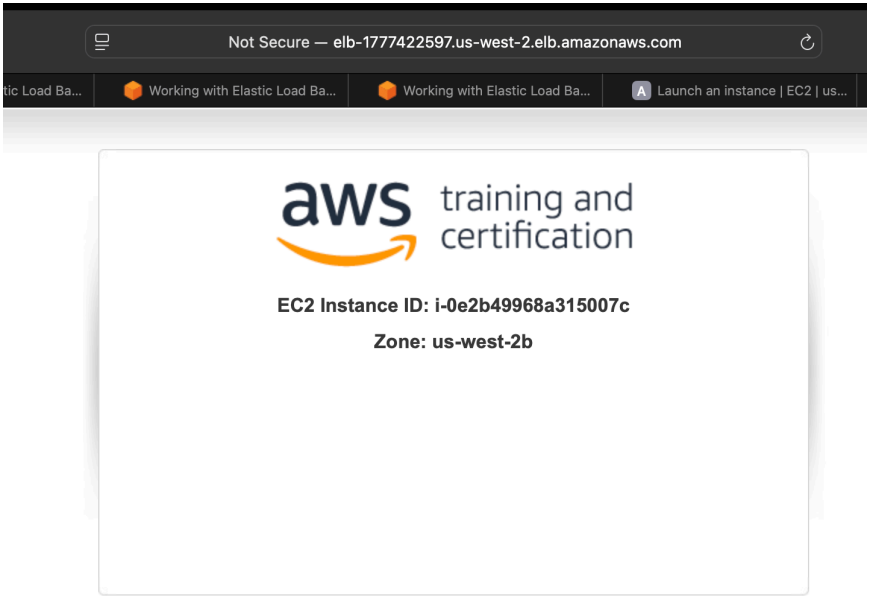

8. In the load balancer page select the newly created target group and launch the load balancer by confirming the configurations.

=



9. Now copy the **DNS name** from the load balancer details section and view the page in the browser, if you refresh the page multiple times it will show different instance ID which proves the response is coming from different instances.



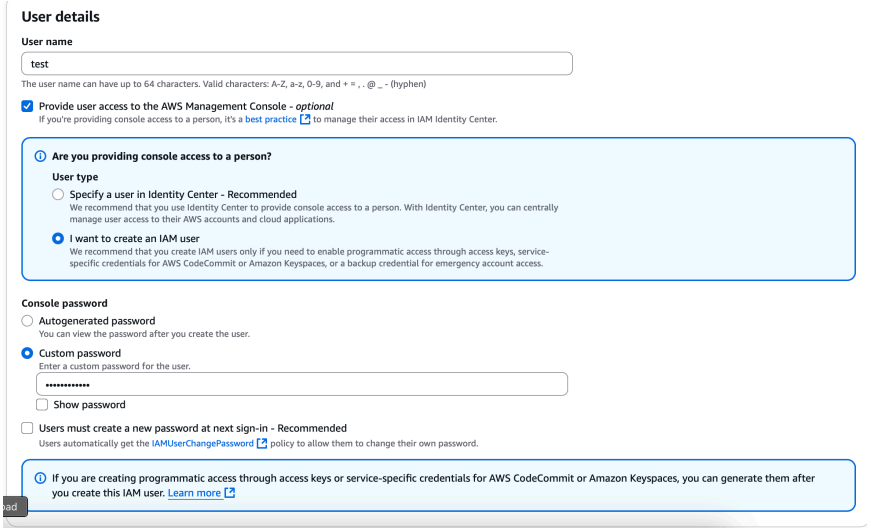
# 4.

# CREATE NEW USER FROM ROOT FROM IAM

1. Login to the AWS console with root account.

2. Go to IAM control panel and choose users from navigation panel

3. Choose **Create user** and enter username for the new user

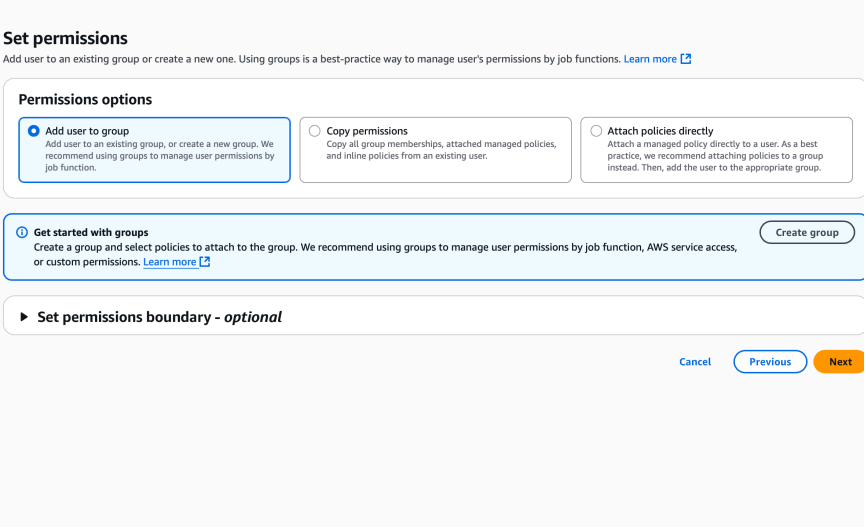


4. To create a traditional IAM user check the box where it says “provide user access to management console”

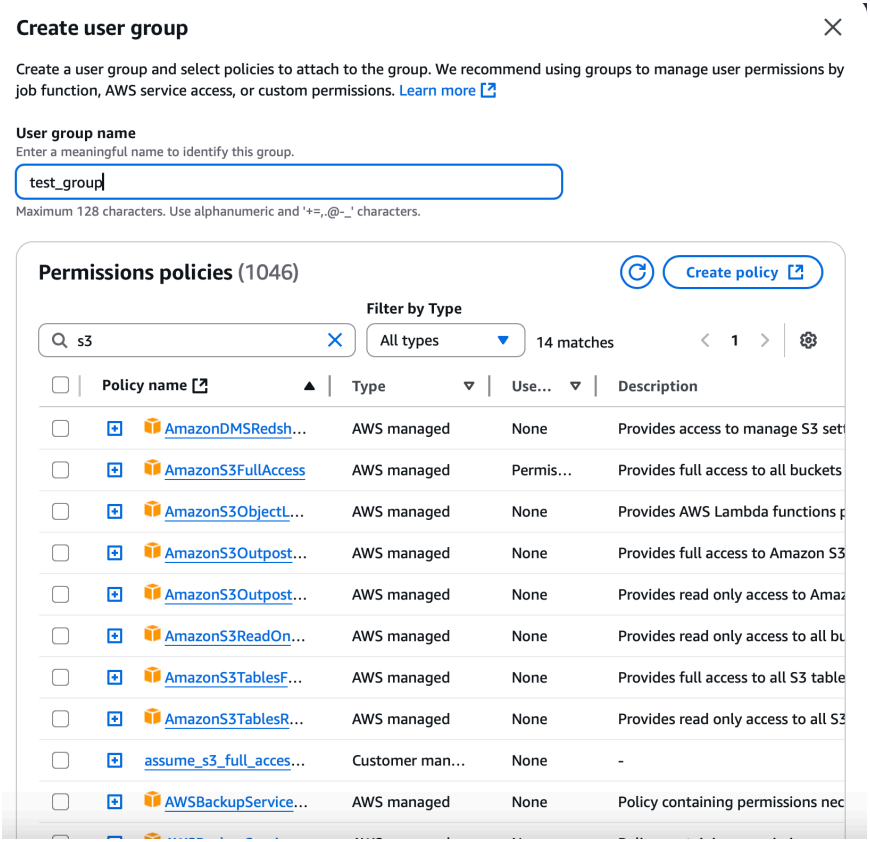
5. Then in the next field select “I want to create an IAM user”

6. Then select the password handling approach by which the user should login first time the choose next

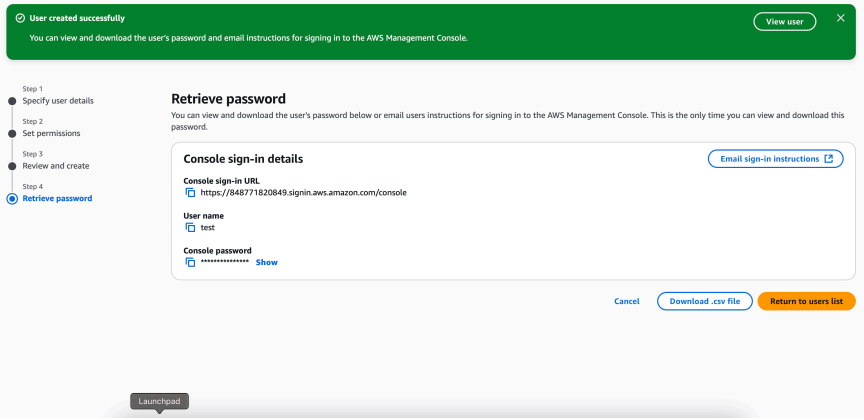
7. In next step we need to add permission to the user, it is recommended to assign a user to a group




8. If there is no existing user group create a new group, you can add policy to the group or can create a custom policy



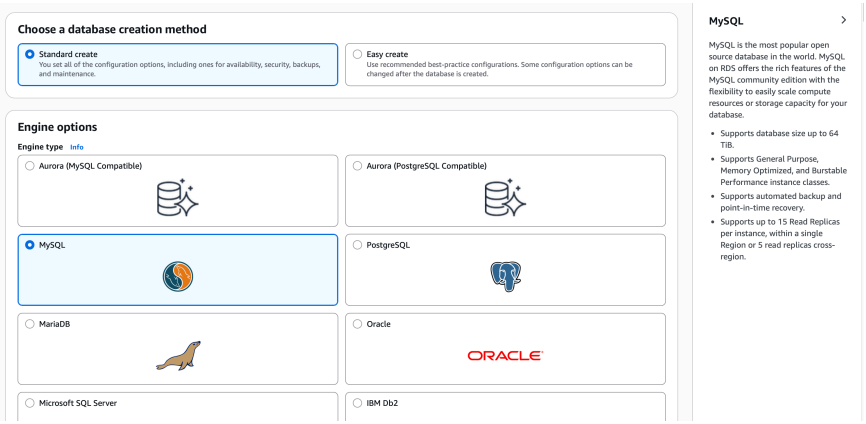

9. After adding user to a group click on next to review the user and create the user account, the user login details can be shared via email through the console.



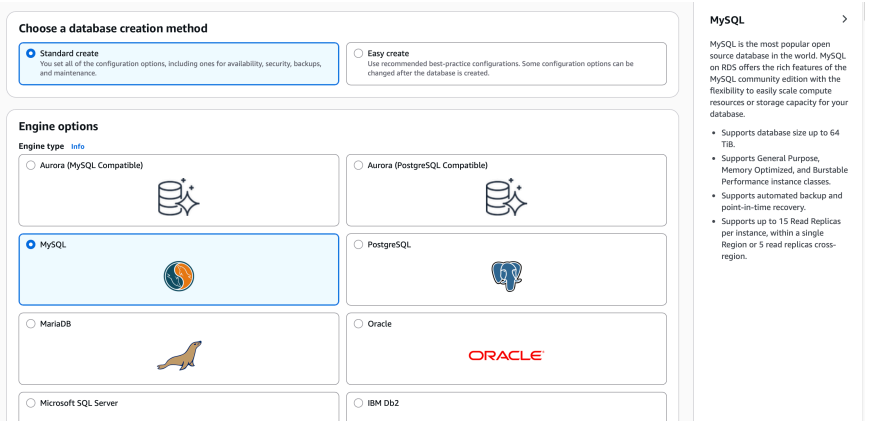
# 5.

# CREATE RDS SERVER AND CONNECT USING MYSQL WORKBENCH

1. In AWS management console go to Aurora RDS control panel
2. In navigation panel select databases and choose **Create database**
3. In **choose a database creation method** select standard create

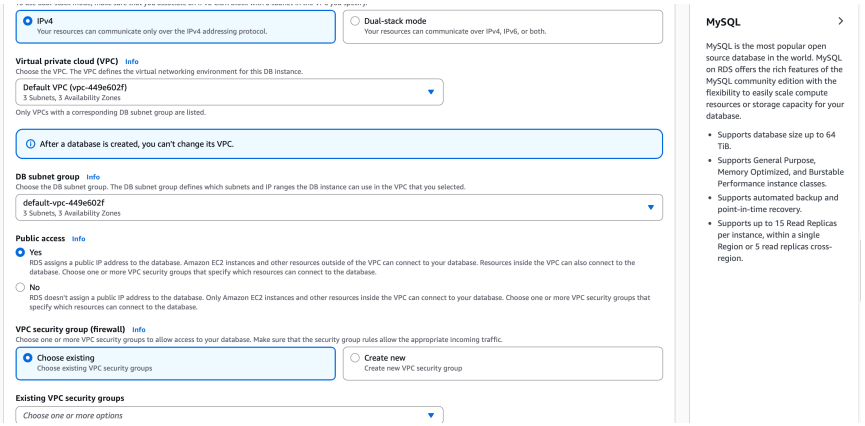



4. For **Engine options** select Mysql
5. In **Templates section** choose free tire



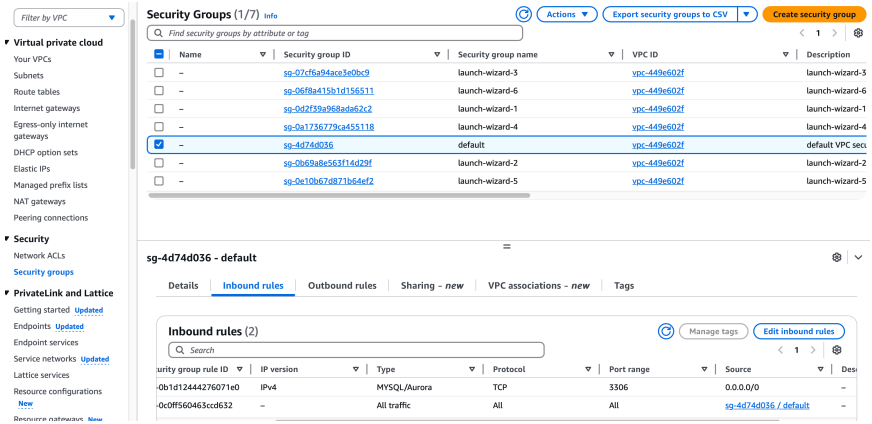

6. Set admin user credentials in **credential settings**

7. In **Connectivity** section enable public access and disable EC2 connectivity.

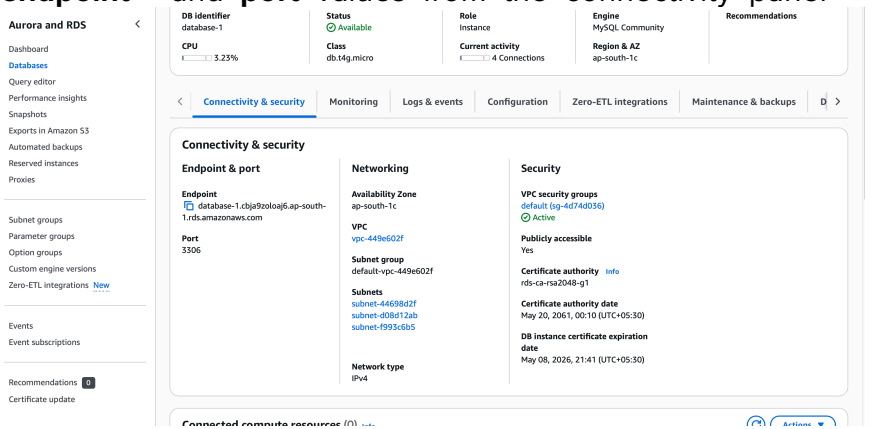



8. Leave other things to default and click **Create database**

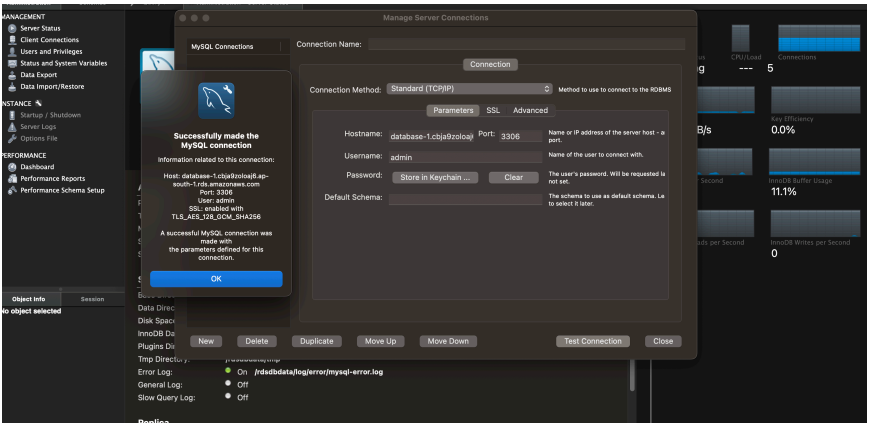
9. After the database is available go to its security group configuration, and in details panel verify its inbound rule. We have to add a MYSQL/Aurora rule to be accepted from all the source or your local machines public IP.



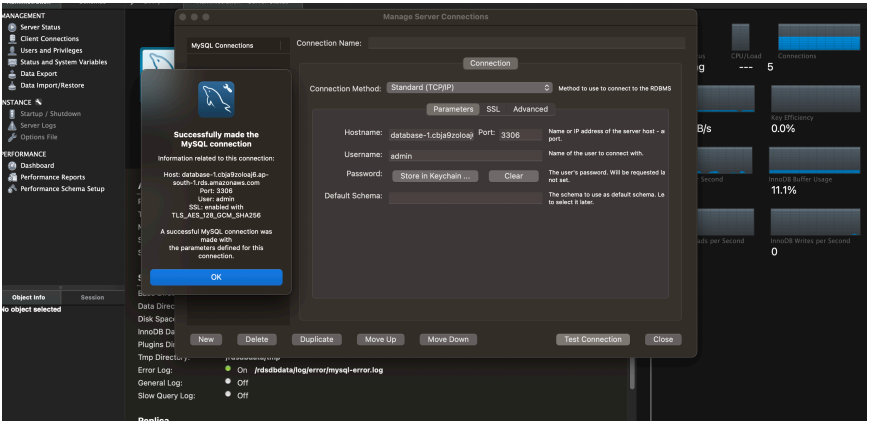

10. After this step go to the database panel and copy the **endpoint** and **port** values from the connectivity panel




11. In MYSQL workbench in databases option add the endpoint as host name and the port as port in the connection setting, with your admin credentials.




12.Test your connection once its successful connect to the database



# 6.

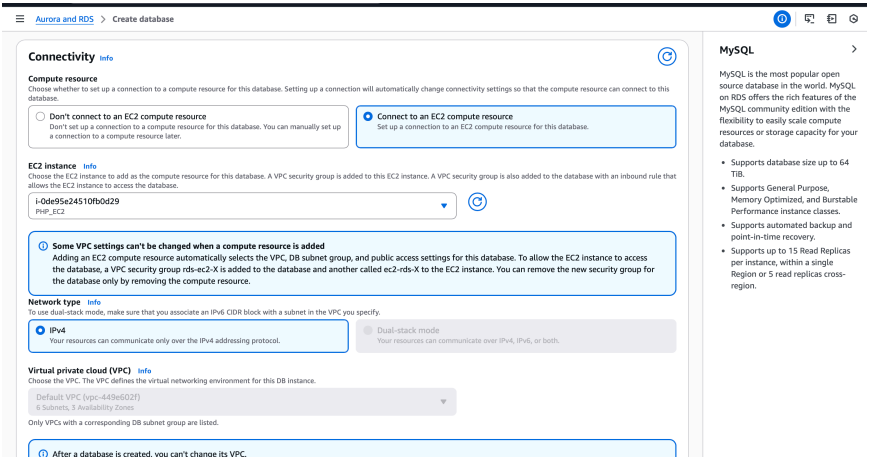
# RUN PHP CODE ON EC2 INSTANCE THAT RETRIEVE DATA FROM RDS SERVER

1. Create EC2 instance with similar configurations in earlier exercise

2. In user data section add the below commands, at this stage you should see error for connecting to the database

```
#!/bin/sh
yum -y install httpd php php-mysqlnd
chkconfig httpd on
systemctl start httpd.service
cd /var/www/html
wget https://github.com/nutansonale/PHP_sample_resource/archive/refs/heads/main.zip
unzip main.zip
cp PHP_sample_resource/* .
systemctl restart httpd.service
```

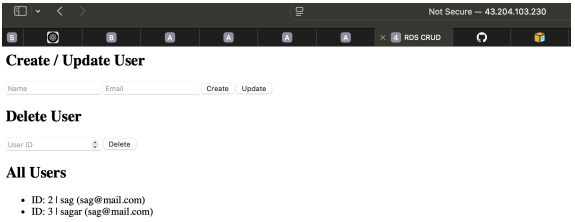
3. Go to RDS control panel and follow initial steps of selecting the RDS instance type and it has to be MYSQL as our PHP code is written for MYSQL engine




4. In connectivity section you have to select Connect ti an EC2 compute resource

5. Keep other selections default as it will create a security group suitable for the the EC2 to connect to the RDS but RDS server wont be available for public access.

6. The output for the configuration should be like this

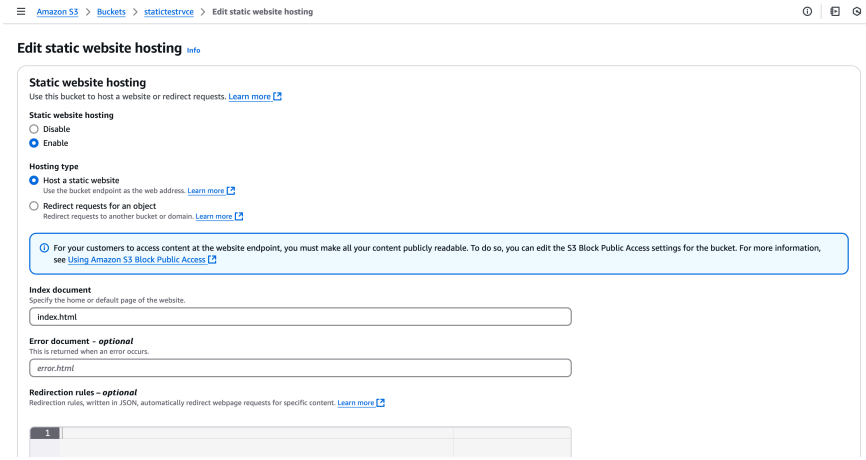


# 7.

# BUILD OWN STATIC WEBSITE AND HOSTING APPLICATION

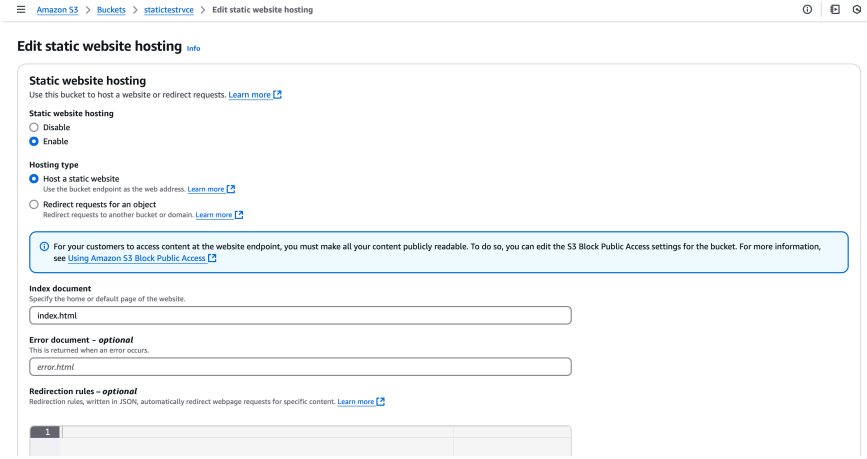
1. Create a general purpose S3 bucket, similar to that of in earlier exercise

2. In the S3 control panel click on the bucket you have created and go to properties tab, enable **Static website hosting**



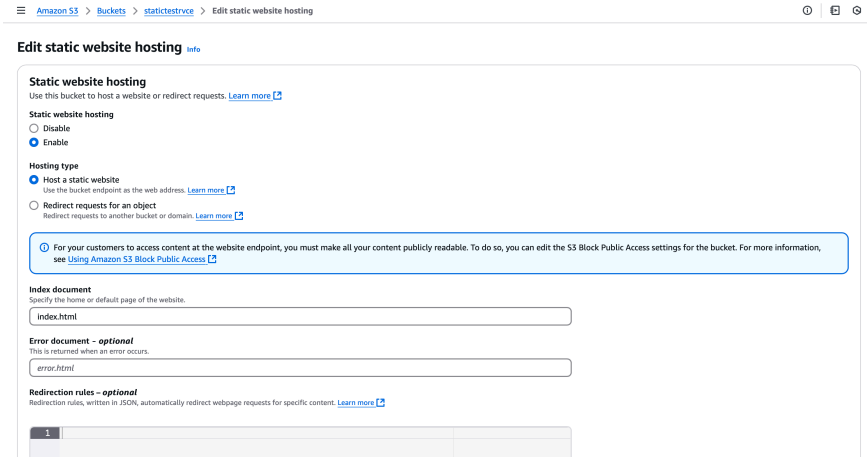

3. While enabling this feature make sure you give the landing page of the the web site, remember if you need to

add any dynamic features it need to be via client side script.

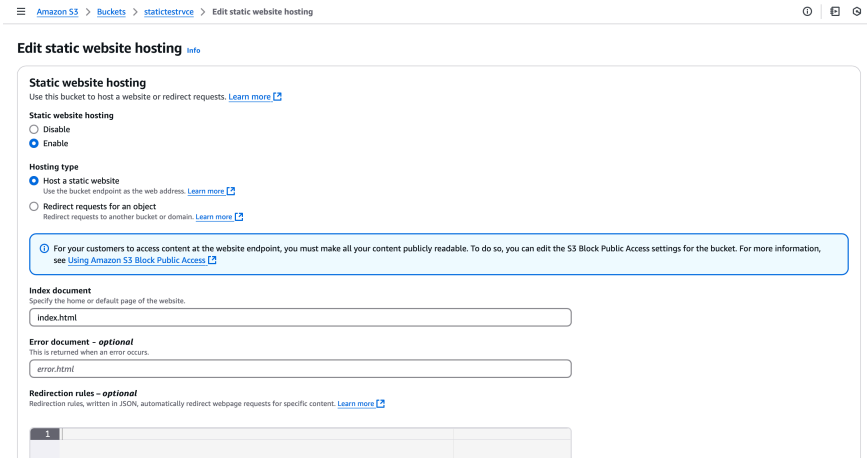



4. Now add the html page into the bucket using upload function

5. Now the buck should be made publicly accessible, so in permissions uncheck all blocking permissions for public access




6. Finally be need to add the below bucket policy to allow all users to have get access to the objects in the bucket this will not be handled by above steps





```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::statictestrvce/*"
        }
    ]
}
```

# 8.

# DEMONSTRATE ECLIPSE INTEGRATION WITH CLOUD

AWS earlier had AWS toolkit support for eclipse IDE for enable faster development, as this has reached its end of life by 2023 developers are using AWS SDKs as they contain prebuilt functions.

Toolkits are still available for Intelij IDEA platform but its a professional platform that need to be subscribed.

# 9.

# RUN JAVA APPLICATION BY CONNECTING TO RDS SERVER

1. Follow the same steps to create RDS and EC2 instance as previously created for PHP script, but there is a slight difference. I have used serverlet approach to keep dependencies as small as possible

2. Here we need to install tomcat externally

#!/bin/sh

yum -y install tomcat10

Wget https://repo1.maven.org/maven2/com/mysql/mysql-connector-j/8.3.0/mysql-connector-j-8.3.0.jar

Wget https://github.com/nutansonale/PHP_sample_resource/archive/refs/heads/java.zip

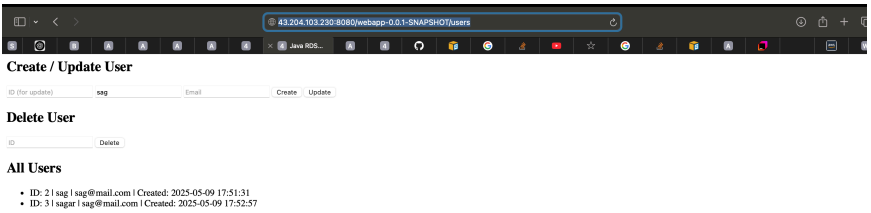
Unzip java.zip

cp PHP_sample_resource/webapp-0.0.1-SNAPSHOT.war /var/lib/tomcat10/webapps/

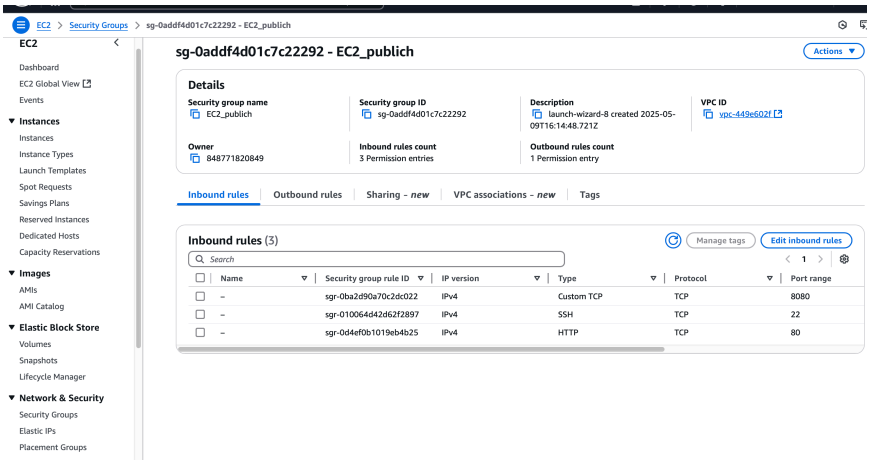
cp mysql-connector-j-8.3.0.jar /usr/share/tomcat10/lib/

Systemctl enable tomcat10

Systemctl start tomcat10




3. Note tomcat runs at 8080 port hence the inbound rule in security group of the EC2 should allow that port similar to 80



# 10.

# DEMONSTRATE AUTO SCALING GROUP

1. Open the Amazon EC2 console and choose Auto Scaling Groups from the navigation pane.
2. On the navigation bar at the top of the screen, choose the same AWS Region that you used when you created the launch template.
3. Choose Create an Auto Scaling group.
4. On the Choose launch template or configuration page, do the following:
    - For Auto Scaling group name, enter a name for your Auto Scaling group.
    - For Launch template, choose an existing launch template.
    - For Launch template version, choose whether the Auto Scaling group uses the default, the latest, or a specific version of the launch template when scaling out.
    - Verify that your launch template supports all of the options that you are planning to use, and then choose Next.
5. On the Choose instance launch options page, if you're not using multiple instance types, you can skip the Instance type requirements section to use the EC2 instance type that is specified in the launch template.

6. Under Network, for VPC, choose a VPC. The Auto Scaling group must be created in the same VPC as the security group you specified in your launch template.

7. For Availability Zones and subnets, choose one or more subnets in the specified VPC. Use subnets in multiple Availability Zones for high availability.

8. If you created a launch template with an instance type specified, then you can continue to the next step to create an Auto Scaling group that uses the instance type in the launch template.
   Alternatively, you can choose the Override launch template option if no instance type is specified in your launch template or if you want to use multiple instance types for auto scaling.

9. Choose Next to continue to the next step.
   Or, you can accept the rest of the defaults, and choose Skip to review.