# **Description**

Deploying a Multi-Tier, Auto scalable and Load Balanced Web Application using EC2 and RDS.

# **Acceptance criteria**

By the end of this case study you would have created a production ready sample Java application that is highly available and on fault tolerant infrastructure. This application uses Amazon RDS as a managed database to store the data.

The main services covered in this lab are

- VPC

- RDS - MySQL

- EC2 Auto Scaling, Application Load Balancer

- SNS

***PS: This lab is in continuation from Case 1 - VPC, we consider you already have the VPC and other components from Case 1 - VPC except NAT Gateway. We also have divided this lab in two parts. We will create the required services manually (single point of failure) in the first part and remediate that in part two.***

***Also edit all the Security Groups from the previous labs and replace RDP ports with SSH, since we are going to create Linux instances in this lab.***

# **Instructions**

**Case Study 02 Part 01 of 02**

**Step 01 Creating an RDS instance**

Let us create a database, but before we do that we need to tell the service as to which subnets to be used for creating RDS instances, we do it by creating a Subnet group.

1. Find the RDS service in the management console and click on it.
2. Expand the left hand side navigation pane (three horizontal lines) and click on Subnet groups.
3. Click Create subnet group, name it my-db-sg. You can paste the same in the description.
4. Choose MyVPC from the VPC dropdown.
5. In the Add subnets section, select the first AZ and the corresponding private subnet from next dropdown. Click on Add subnet button.
6. Change the AZ and select the other corresponding private subnet and click on Add subnet button.
7. You should see both of your private subnets in the group now.
8. Click on Create.

***Let us now create an RDS MySQL instance.***

1. Click on Create database. You should see the six choices supported by RDS as of now. We will select MySQL.
2. Choose Dev/Test MySQL on the next page. Click Next
3. Fill the below details, leave defaults what is not explicitly mentioned here.

- DB instance class: db.t2.micro

- Multi-AZ deployment: No (Chargeable if you say yes)

- Storage type: General Purpose (SSD)

- Allocated storage: 20 GB

***You will see a monthly cost below; this will be charged in case you are not in free tier period.***

1. Settings

- DB instance identifier: inventory-db

- Master username: master

- Master password: lab-password

1. Click Next

- Virtual Private Cloud (VPC): MyVPC

- Subnet group: my-db-sg

- Public accessibility: No

- Availability zone: No preference

- VPC security groups: Choose existing VPC security groups Add My-DB-SG

- Remove the Default one

1. In the Database options section, configure:

- Database name: inventory

- Click Create database

It will take a little time now to create your RDS instance, we can continue with the rest steps. Open another tab to do other steps and leave this page open, we will revisit soon.

**Step 02 - Creating a Role**

1. Open IAM in a new tab and click on Roles Create role
2. Select type of trusted entity: EC2, go next to permission page
3. Search for AmazonSSMFullAccess and go next, skip the tags and go next Review
4. Role name: My\\_SSM\\_Role, Create.

This role will allow the application to use the System Manger feature, more on this later.

**Step 03 - Creating App Server (EC2 instance)**

Let us create our application servers now. Go to EC2 Dashboard and click on Launch Instance.

1. AMI: Amazon Linux 2 - Instance Type: t2.micro

On the Configure Instance Details page select the below mentioned points and leave everything else as default.

1. Network: MyVPC
2. Subnet: MyPublicSubnet01
3. IAM Role: My\\_SSM\\_Role
4. Expand the Advance Details section and paste the following script in the user data section. The format of the script is very important, please copy it in text editor and ensure there are no extra line breaks or spaces and then paste in the user data section.

***\*\*\\*\\*\\* VV Imp, your lab will fail if you do not do it properly\\*\\*\\*\*\****

|  |
| --- |
| #!/bin/bash  # Install Apache Web Server and PHP  yum install httpd mysql -y  amazon-linux-extras install -y php7.2  # Download Lab files  wget https://us-west-2-tcprod.s3.amazonaws.com/courses/ILT-TF-100-ARCHIT/v6.2.1/lab-1-webapp/scripts/inventory-app.zip  unzip inventory-app.zip -d /var/www/html/  # Download and install the AWS SDK for PHP  wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip  unzip aws -d /var/www/html  # Turn on web server and ensure running on reboot  service httpd start  chkconfig httpd on |

1. This script will

- Install an Apache web server and the PHP

- Download the Inventory application and the AWS SDK

- Activate the Web server and configure it to automatically start on boot

1. Click on Next: Add Storage

Your instance will come with a root volume of 8 GB as you can see in this screen. We can add additional EBS volumes if need be, as of now click on

1. Next: Add Tags

- Create a Tag with Key: Name and Value: MyAppServer

1. Click on Next: Configure Security Group

- Click on the Select existing security group, find and select My-App-SG

1. Click on Review and Launch.

- On the next page ensure that your AMI is free tier eligible and Instance Type is showing as t2.micro.

1. Click on Launch.

You may use your existing key pair from previous lab.

Go to the EC2 dashboard and see your server should be launching.

Once the server is launched, copy its public IP address and open in a browser window, you should see a minimal web page. Click on the settings button to set up database connection.

1. Go to the RDS dashboard and find out the database connection endpoint
2. Return to the browser tab with the Inventory application and enter the below information:

- Endpoint: Paste the endpoint you copied earlier

- Database: inventory

- Username: master

- Password: lab-password

1. Click Save.

The application will now connect to the database and will load some pre-fed sample data. Feel free to add, edit and delete inventory information using the web application.

So as of now you have one single EC2 instance serving a web application, it is storing the data in a RDS database. But this instance is a manually created and what will happen if it goes down? The data might remain saved in the database but will be unavailable till the time the app server is/are brought back. The process has to be an automated one rather than manual, EC2 auto scaling is the feature that comes to rescue here!

Terminate the instance, we are simulating a disaster now

**Case Study 02 Part 02 of 02**

You will now be launching this application using a Launch Configuration into an Auto Scaling Group, the ASG will automatically grow and shrink the number of your servers based on the user defined threshold. The requests to your application will be distributed by Application Load Balancer.

**Step 01 - Creating an Auto Scaling Group**

1. Go to the Auto Scaling section in your EC2 dashboard and click on Create Launch Configuration.

- AMI Amazon Linux AMI - 2

- Instance Type t2.micro

- Name MyAppServer\\_V01\\_LC

- IAM Role My\\_SSM\\_Role

- Expand Advance Details Section and paste the same user data script from above.

1. Go next.

- No additional storage, go next.

- On the security group page, choose My-App-SG

- Click on Create launch configuration

- Create a new key pair or select the existing and Create launch configuration

1. Your Launch Configuration is created, let us now create the auto scaling group. Click on Create an Auto scaling group using this Launch configuration.

- Group name MyApp\\_ASG

- Group size Start with 2 instances

- Network MyVPC

- Subnet Select both the public subnets here.

- Configure scaling policies - Use scaling policies to adjust the capacity of this group

- Scale between 2 and 5 instances.

- Target value 60

- Instances need 10

- Next Configure Notification

- Add Notification Create Topic

- Send a notification to MyASG\_Topic

- With these recipients - your email ID

- Next Create a Tag with Key: Name; and Value: MyAppServer

- Review Create Auto Scaling group

1. Click on Close, you would be directed to the Auto Scaling Groups Dashboard. Explore the Activity History and other tabs.

You have just launched our highly available inventory application in an ASG. You can open the public IP addresses of both the instances in separate browser and see what happens. You should be seeing the webpage with same information but look at the bottom. You should see the instance ID. This way you can identify which server your request is being served from.

Also check if you received an email from SNS topic, you need to confirm the subscription.

But the end users would not have IP addresses to your server right? They should have a domain name to go to. Let us create a Load balancer that will divert the traffic to both these instances in weighted round robin method.

**Step 02 - Creating an Application Load Balancer**

1. Go to the Load Balancing section of EC2 dashboard and click on Target Group

- Create Target Group

- Target group name MyTG

- VPC MyVPC

- Leave rest defaults and click Create.

1. Click on Load Balancers Create Load Balancers
2. From next screen, create an Application Load Balancer

- Name MyALB

- Scroll down to the Availability Zones Section

- Select the VPC in which you have launched the ASG

- Select Public Subnets from both AZs. This is a critical step, reconfirm before going forward.

1. Next - Configure Security Settings Ignore the warning, it is recommending to have SSL certificate.

- Next - Configure Security Groups. Select My-ALB-SG from existing ones.

- Next Configure Routing

- Target group Existing Target Group

- Name MyTG

- Leave rest defaults - Register Targets Review Create

Click on close and it will take you to the load balancer dashboard, you should see the DNS endpoint of your load balancer in Description Tab. ALB takes a little time to come up. Refresh till you see the state as active.

Let us register our instances in ASG with the MyTG target group. Select your ASG and go to action dropdown and click on edit. You will find a field for target group in the lower section. Click on the empty field and assign MyTG. Save it (save button is towards the top right of lower section)

Open the DNS address of your ALB in a browser and notice what it shows. It is now diverting the traffic to both your instances. You can see the behavior of load balancer while you refresh the page and notice the instance ID.

So at this point of time your application can be reached by the load balancer endpoint as well as direct IPs of the instances. This is not an ideal behavior; we should not allow our app servers to accept traffic from anywhere else apart from the load balancer in order to ensure security.

Can you restrict it?

**Step 02 Modify the Security Groups to ensure security on incoming traffic**

Change the My-App-SG security group settings as shown below.

1. My-App-SG

| Type | Protocol | Port Range | Source | |

| :---: | :---: | :---: | :---: | :---: |

| HTTP | TCP | 80 | Custom | \<My-ALB-SG> |

| HTTPS | TCP | 443 | Custom | \<My-ALB-SG> |

| RDP | TCP | 3389 | Anywhere | 0.0.0.0/0 |

If your application is reachable by the load balancer endpoint and not through visiting the IP addresses or EC2 instances in browser, you have done it well.

You can also now try deleting one/more server in order to verify whether the auto scaling feature is able to spin up instances in response.

If the objectives have successfully completed, then take the below snapshots -

1. FrontPage of your application while it is connected to the database and you have made a few changes in the existing data showing there.
2. RDS dashboard page showing the endpoint of your database.
3. Auto scaling Group page while the Activity History tab is clicked.

You can upload these in git, attach them while submitting this lab completion.

**Step 03 Clean up steps**

Delete the resources in the below order

1. RDS (do not create a snapshot and do not retain the automated backup files)
2. ALB
3. Auto Scaling Group (takes little time to delete)
4. Target Group Launch Configuration.

***All the services used in this lab are eligible and covered within the free tier account. There should not be any charge if you delete all the resources within a couple of hours of creation provided you have monthly limits left.***