Project 3 - Deploying a Java application using PAAS/SAAS services

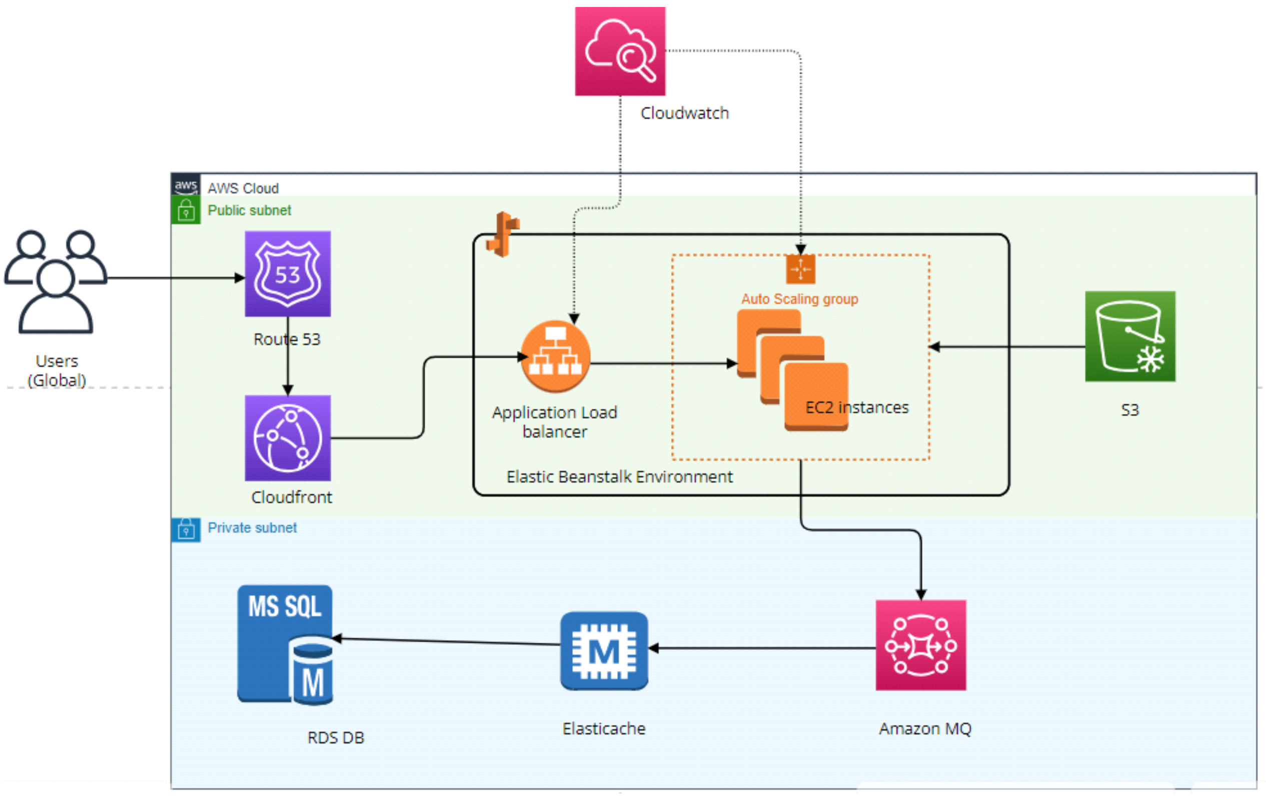
03 June 2024 19:47

Public subnet
- Beanstalk (one each for Tomcat, nginx,) + LB
- Autoscaling
- S3, EFS

Backend services (Private subnet)
- RDS
- Elastic cache
- Active MQ
- Route 53
- Cloudfront
- ACM

Architectural diagram

Below is the architectural diagram for this project.



Flow of execution
- Create keypairs for beanstalk instance
- Create security groups for backend services
- Create
    - RDS
    - Amazon Elastic cache
    - Amazon MQ
- Create elastic beanstalk environment
- Update SG of backend to allow traffic from beanstalk security group
- Update SG of backend to allow internal traffic
- Launch EC2 instance for Initializing RDS DB
- Login to the instance and initialize RDS DB

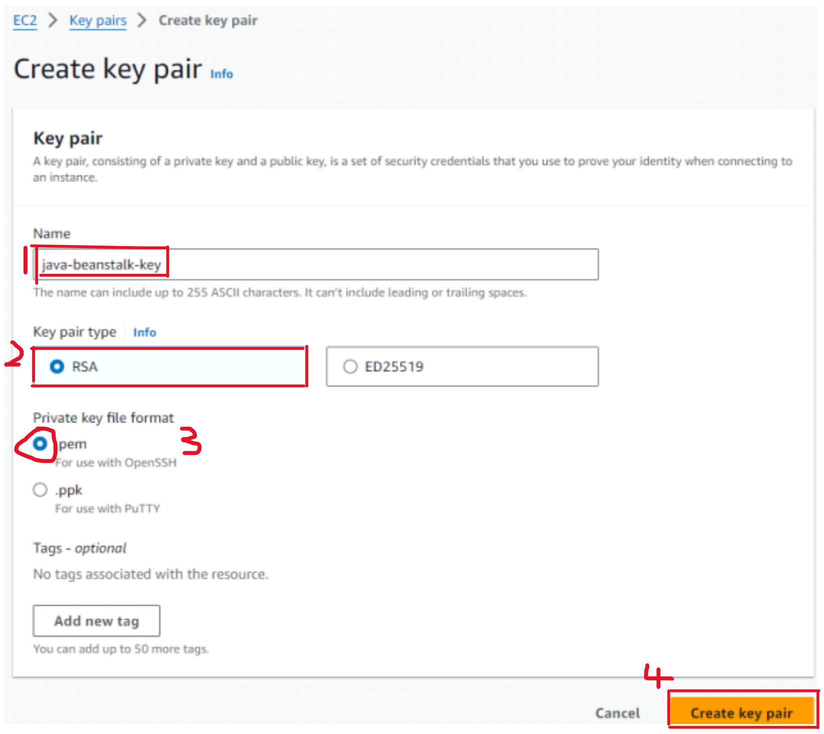
- Change health-check on beanstalk to /login
- Add 443 https listener to ELB
- Build artifiact with backend information
- Deploy artifact to beanstalk
- Create CDN with SSL cert
- Update entry in GoDaddy DNS zones
- Test URL

Requirements (needed for the project but not covered in this documentation):

- Before commencing, it is worth noting that a domain name (iamugo.de)has been purchased and an SSL certificate is already in place for this project. The certificate is managed in AWS Certificate Manager (ACM) and will be used to validate the HTPS connections to the application. I obtained the certificate from GoDaddy when I purchased a domain.
- Installation of jdk11 on pc
- Maven install
- Purchase of a domain name (not free domains)

Step 1: create keypairs for beanstalk

- I logged on to the console and went to the ec2 dashboard to create the keypair


EC2 > Key pairs > Create key pair

Create key pair Info

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

1 java-beanstalk-key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info

2 [✓ RSA] ED25519

Private key file format

3 [✓ .pem] For use with OpenSSH

.ppk For use with PuTTY

Tags - *optional*

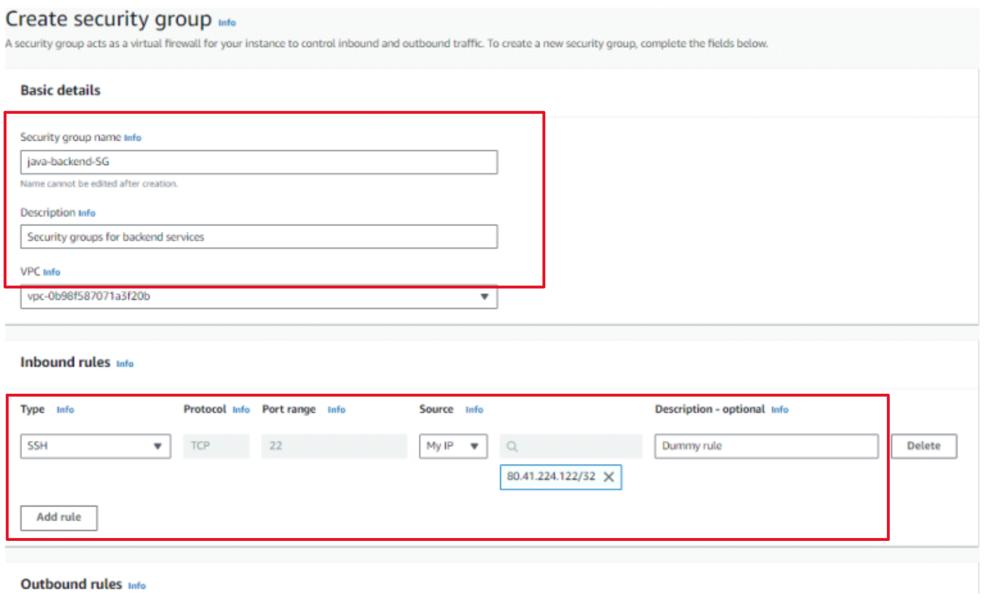
No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel 4 Create key pair

Step 2: create security groups for backend services


Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name: java-backend-SG

Name cannot be edited after creation.

Description: Security groups for backend services

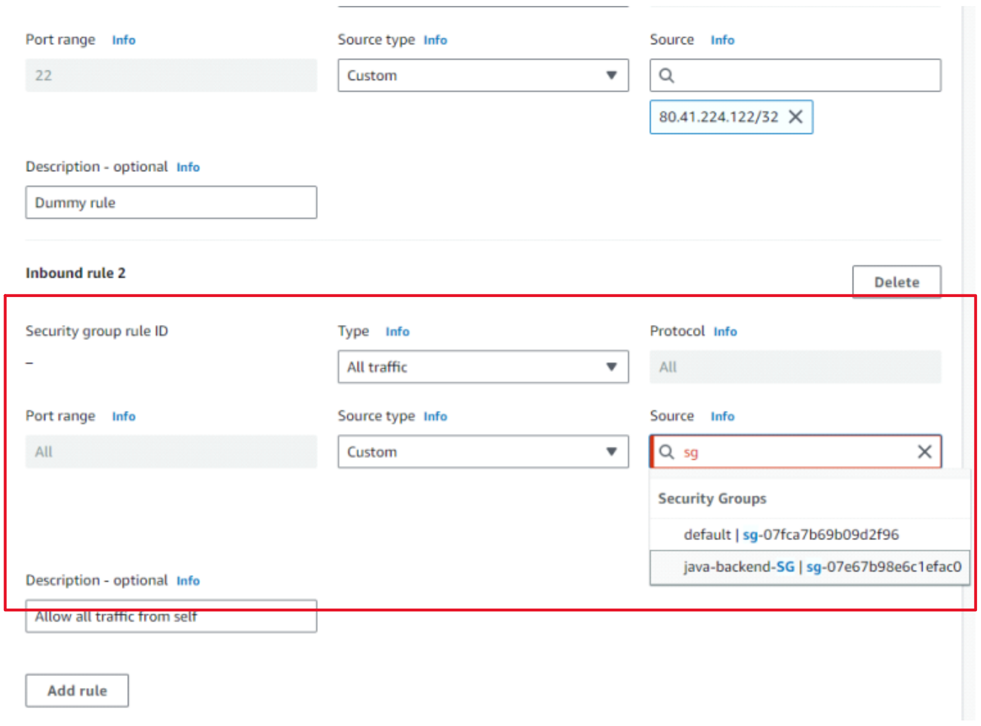
VPC: vpc-0b98f587071a3f20b

Inbound rules

Type	Protocol	Port range	Source		Description - optional
SSH	TCP	22	My IP	80.41.224.122/32	Dummy rule

I first of all had to create a dummy rule (allow ssh from my IP) in order to first create the security group, and then edit it to allow all traffic from itself.

I did this because all the backend services will share the same security group and it is vital they are able to communicate with each other. I will edit this security group once all the backend services are created to include access to their relevant port numbers.


Port range: 22

Source type: Custom

Source: 80.41.224.122/32

Description - optional: Dummy rule

Inbound rule 2

Security group rule ID: –

Type: All traffic

Protocol: All

Port range: All

Source type: Custom

Source: sg

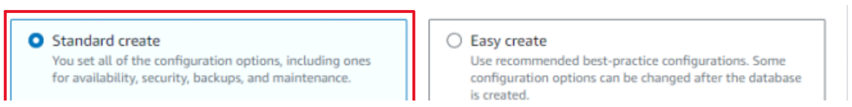
Security Groups
- default | sg-07fca7b69b09d2f96
- java-backend-SG | sg-07e67b98e6c1efac0

Description - optional: Allow all traffic from self

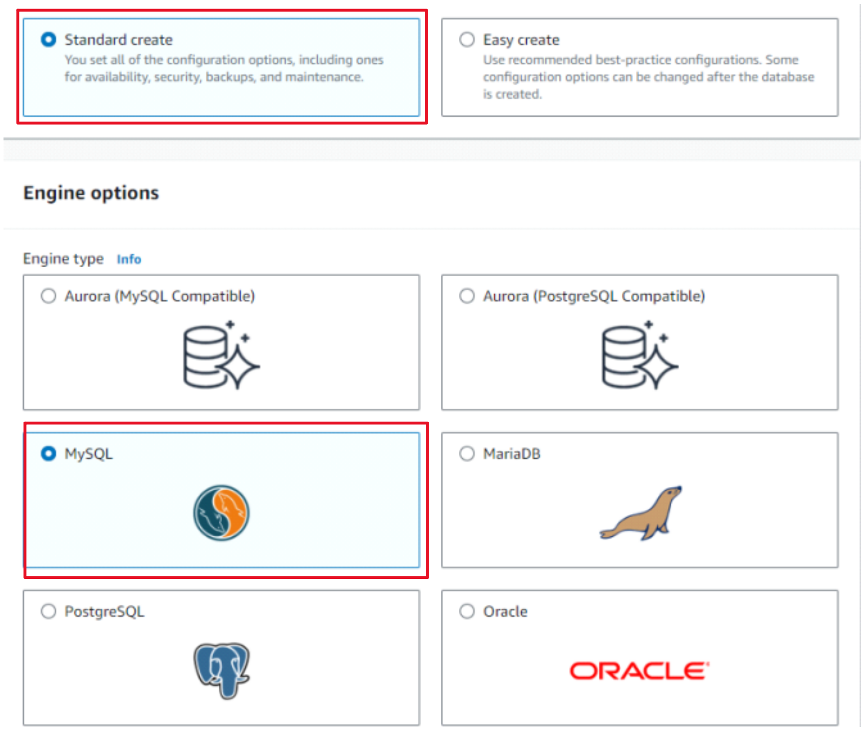
Step 3a: create RDS

Firstly, I created a subnet because the RDS instance will be in a private subnet. The subnet is Highly available as I used all the available availability zones in the N.Virginia region. In real-time, it will be wise to only use the AZ where the RDS will serve users.

- Create a subnet group for the rds cluster. This is because the rds cluster will be in a private subnet
- Create parameter groups
- Create RDS db. The option to use aurora here also works as it is cheaper and faster but for the sake of this project, I stuck with mysql


[✓ Standard create] You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create: Use recommended best-practice configurations. Some configuration options can be changed after the database is created.


[✓ Standard create]
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

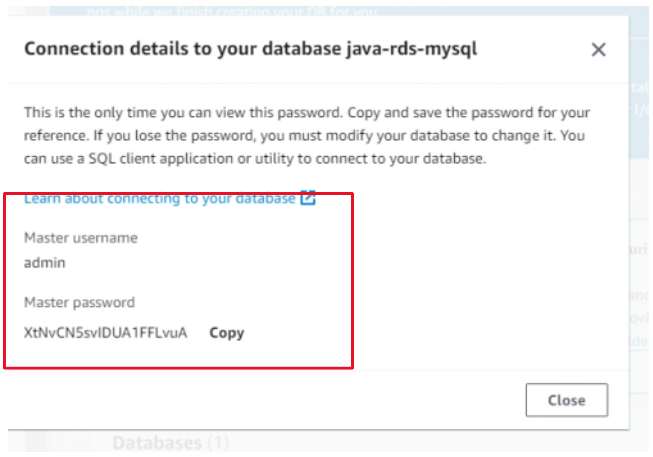
Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

- Aurora (MySQL Compatible)
- Aurora (PostgreSQL Compatible)
- [✓ MySQL]
- MariaDB
- PostgreSQL
- Oracle

- I used the free tier setup which has the following configuration: 20GB, 1GB RAM, t3.micro. The database name is accounts.
- When the db is created, I copied the master password into a notepad


Connection details to your database java-rds-mysql

This is the only time you can view this password. Copy and save the password for your reference. If you lose the password, you must modify your database to change it. You can use a SQL client application or utility to connect to your database.

Learn about connecting to your database

Master username
admin

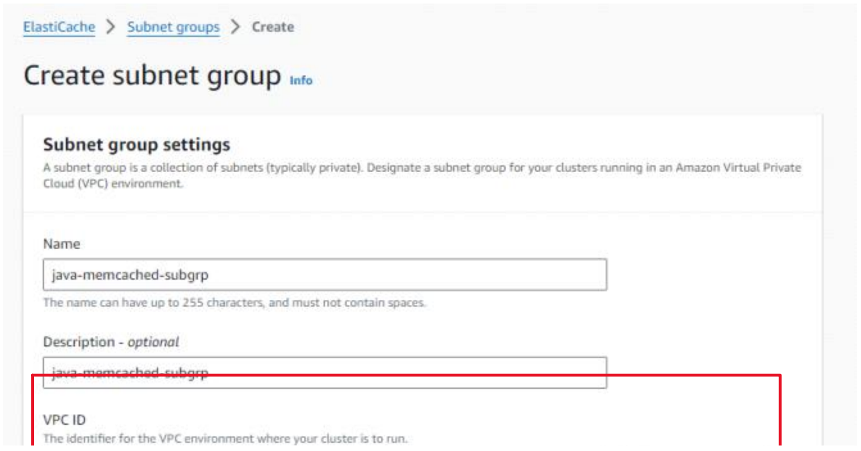
Master password
XtNvCN5svIDUA1FFLvuA Copy

Close

Step 3b: create ElastiCache

I navigated to the ElastiCache dashboard

- Created subnet group


ElastiCache > Subnet groups > Create

Create subnet group Info

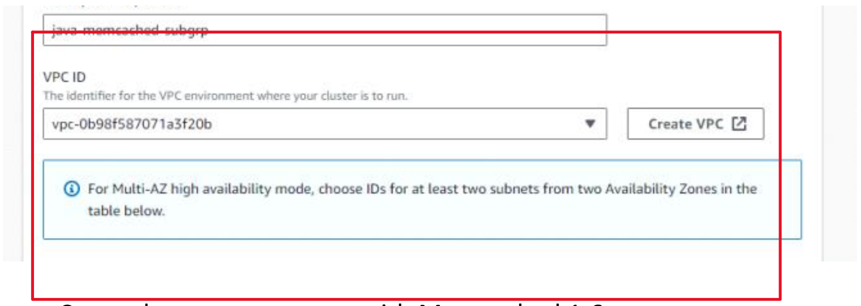
Subnet group settings

A subnet group is a collection of subnets (typically private). Designate a subnet group for your clusters running in an Amazon Virtual Private Cloud (VPC) environment.

Name
java-memcached-subgrp
The name can have up to 255 characters, and must not contain spaces.

Description - *optional*
java-memcached-subgrp

VPC ID
The identifier for the VPC environment where your cluster is to run.


java-memcached-subgrp

VPC ID

The identifier for the VPC environment where your cluster is to run.

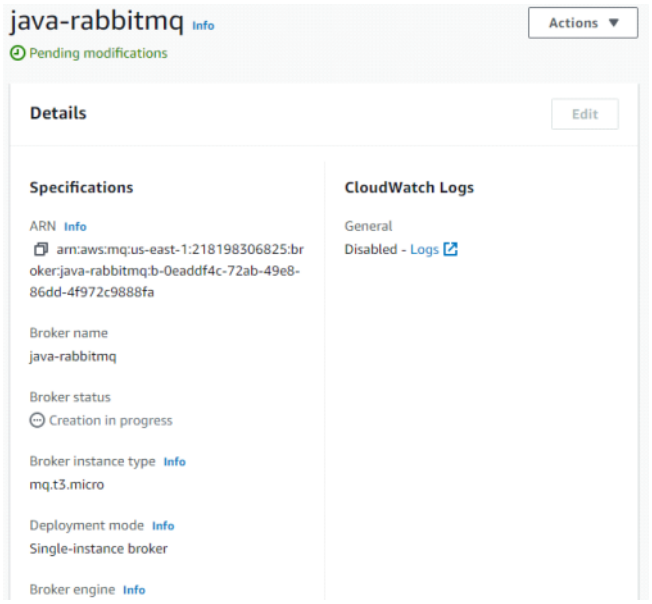
vpc-0b98f587071a3f20b

For Multi-AZ high availability mode, choose IDs for at least two subnets from two Availability Zones in the table below.

- Created parameter group with Memcached 1.6
- Created the ElastiCache cluster (using standard create to have options to stay within the free tier) with t2.micro in the subnet group created earlier
- Copied out the endpoint of the ElastiCache cluster (without the 'amqps' at the beginning and the port number at the end) to a notepad

Step 4: create Amazon MQ (RabbitMQ)

- This is the message brokering service for the applications
- Configuration: t3.micro, Rabbitmq, private connection. Used the backend security groups configured earlier


java-rabbitmq

Pending modifications

Details

Specifications

ARN

arn:aws:mq:us-east-1:218198306825:broker:java-rabbitmq:b-0eaddf4c-72ab-49e8-86dd-4f972c9888fa

Broker name

java-rabbitmq

Broker status

Creation in progress

Broker instance type

mq.t3.micro

Deployment mode

Single-instance broker

Broker engine

CloudWatch Logs

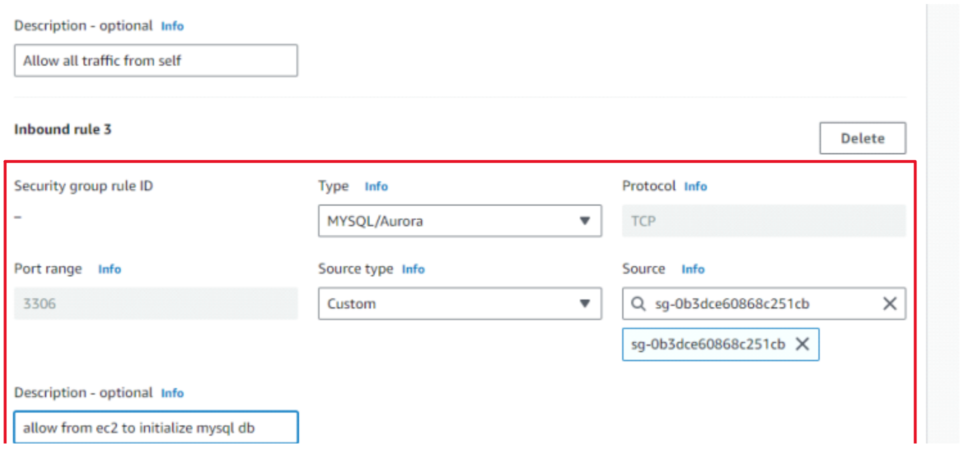
General

Disabled - Logs

- When created, I copied the endpoint (without the port number at the end) of the Amazon MQ broker and saved in a notepad

Step 5: initialize DB

- Next was to launch an ec2 instance in the same VPC as the RDS instance. This instance will be used to initialize the DB using some SQL queries. The SQL queries used will be found in this repository.
  - Configuration of instance - t2.micro, ubuntu 22, with a different security group (22 from my ip)
- Edited the security group of the backend services to allow connection on port 3306 (the RDS mysql database) from the ec2 security group


Description - optional

Allow all traffic from self

Inbound rule 3

Security group rule ID

–

Type

MYSQL/Aurora

Protocol

TCP

Port range

3306

Source type

Custom

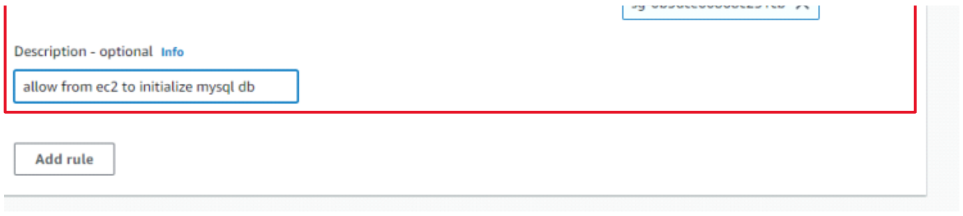
Source

sg-0b3dce60868c251cb

sg-0b3dce60868c251cb

Description - optional

allow from ec2 to initialize mysql db


Description - optional Info

allow from ec2 to initialize mysql db

Add rule

- SSH to the ec2 instance using the public IP and the downloaded keypair
- Installed the mysql-client on the ec2 instance using the command below:

```
Sudo apt update && sudo apt install mysql
```

- Logged into the db using the command below:

```
 mysql -h rds_endpoint -u admin -pXtNvCN5svIDUA1FFLvuA accounts
```

  At the moment when you run the '`show tables;`' command in the mysql prompt, no tables are present.

- Next, I cloned the repo containing the SQL file - db_backup.sql and push the file to the db

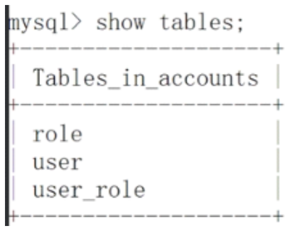
```
git clone https://github.com/hkhcoder/vprofile-project.git
```

  To push the SQL file into the db and initialize it, run the command:

```
mysql -h rds_endpoint -u admin -pXtNvCN5svIDUA1FFLvuA accounts
< src/main/resources/db_backup.sql
```

  The above command sends the db_backup.sql (from the repo) to the account database and runs the sql queries

- Re-logged in to the RDS instance and ran the '`show tables;`' command



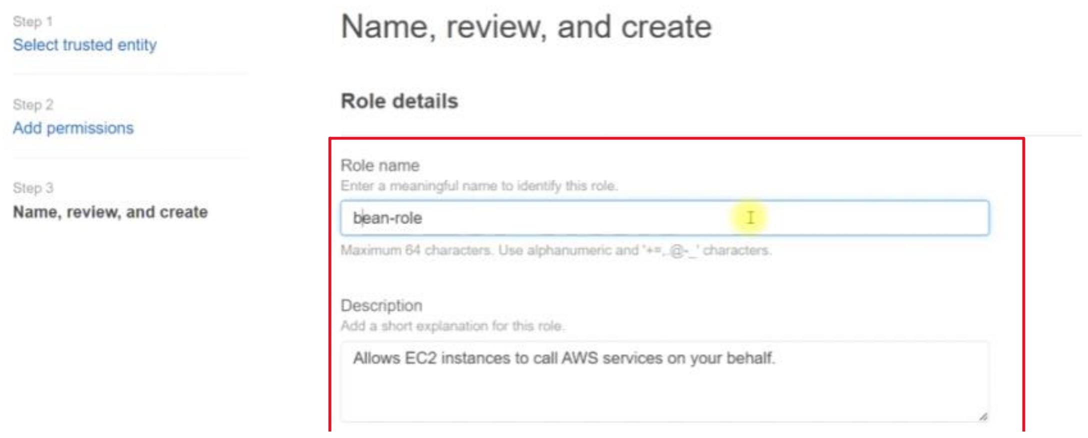
```
mysql> show tables;
+--------------------+
| Tables_in_accounts |
+--------------------+
| role               |
| user               |
| user_role          |
+--------------------+
```

  Next, I deleted the ec2 instance used to initialize the db

Step 6: Setup Elastic Beanstalk

AWS Elastic Beanstalk is the environment where the application will run. It is a managed service that allows application to be easily deployed

- Created beanstalk roles for ec2 instance having the following policies
  - AWSElasticBeanstalkWebTier
  - AdministratorAccess-AWSElasticBeanstalk
  - AWSElasticBeanstalkCustomPlatformforEC2Role
  - AWSElasticBeanstalkRoleSNS

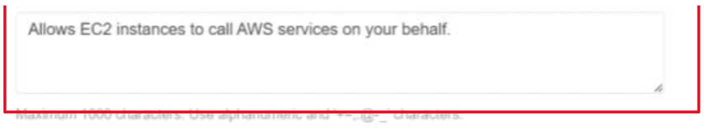

Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
bean-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

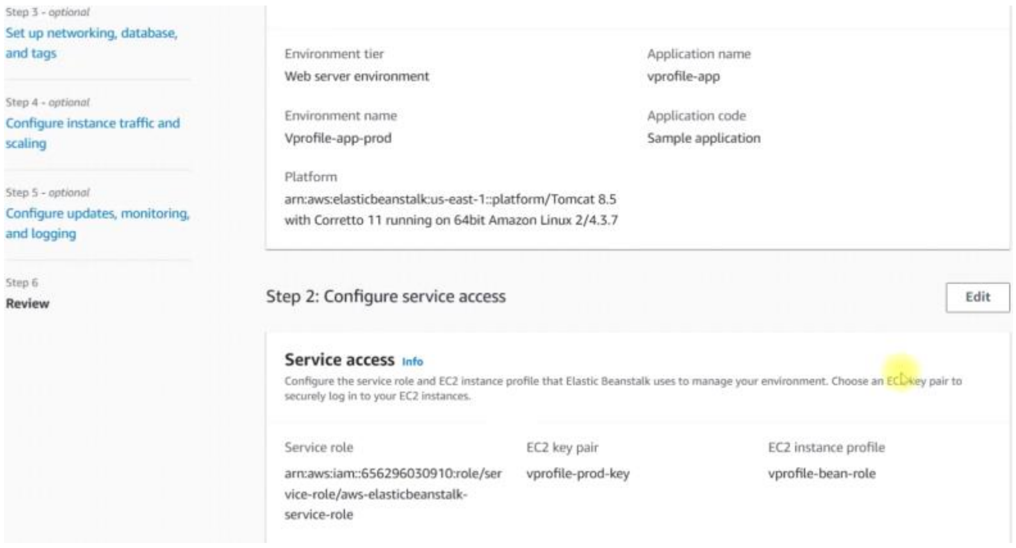
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.


Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Ps: I deleted the aws-elasticbeanstalk-service-role available from a previous job in IAM. If not done, this will create problems as Beanstalk will recreate this role during its initialization

- Configure beanstalk application
  - Choose a unique domain name
  - Tomcat environment with jdk11, since the application requires it
  - When prompted, I chose the roles for I created earlier for the EC2 instance profile and chose to allow Beanstalk to create a service role
  - Select the keypair I created earlier
  - I gave some tags - Name, project,
  - Chose to use an application loadbalancer (just for the project)
  - For ec2: t3.micro
  - NetworkOut is used as the autoscaling metric, since it's a web application that will be serving users. The scaling policy will be based on the number of users accessing the application (Network Out
  - Health check: the application uses /login for its healthcheck
  - I also enabled stickiness
  - Rolling updates & deployment: Rolling at a percentage, probably 10%

- 
Step 3 - optional
Set up networking, database, and tags
Step 4 - optional
Configure instance traffic and scaling
Step 5 - optional
Configure updates, monitoring, and logging
Step 6
Review
Edit

Environment tier
Web server environment

Application name
vprofile-app

Environment name
Vprofile-app-prod

Application code
Sample application

Platform
arn:aws:elasticbeanstalk:us-east-1::platform/Tomcat 8.5 with Corretto 11 running on 64bit Amazon Linux 2/4.3.7

Step 2: Configure service access

Service access Info

Configure the service role and EC2 instance profile that Elastic Beanstalk uses to manage your environment. Choose an EC2 key pair to securely log in to your EC2 instances.

Service role
arn:aws:iam::656296030910:role/service-role/aws-elasticbeanstalk-service-role

EC2 key pair
vprofile-prod-key

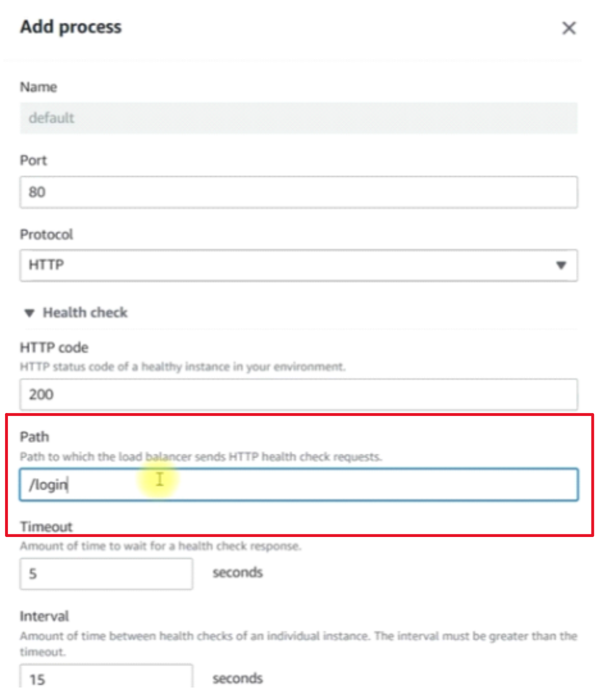
EC2 instance profile
vprofile-bean-role

After creation, clicking on the URL shows you the default application, showing that the beanstalk creation worked. I will be deploying the application

Step 7: Edit ACL on S3 bucket, Beanstalk health check & HTTPS Listener

S3 bucket was used to save the artifact of the application build.

- Opened the S3 service. The S3 bucket created by elastic beanstalk is here. I have to enable to ACL on it.
  - Clicked on the Edit button of the object ownership. Clicked on Enable
  - ACL gives the root user permission to its S3 account or other AWS accounts
- Revisited the Beanstalk Configuration and in the 'Instance traffic and scaling' settings (Processes), edited the health check to `/login`.


Add process

Name: default

Port: 80

Protocol: HTTP

Health check

HTTP code
HTTP status code of a healthy instance in your environment.
200

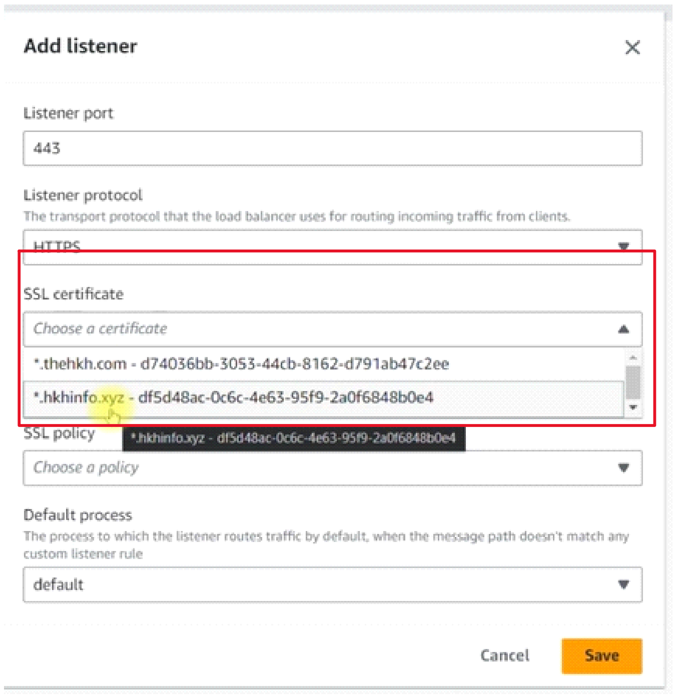
Path
Path to which the load balancer sends HTTP health check requests.
/login

Timeout
Amount of time to wait for a health check response.
5 seconds

Interval
Amount of time between health checks of an individual instance. The interval must be greater than the timeout.
15 seconds

- Also enabled stickiness. Stickiness is a cookie stored in the user's browser that always sends the user to a particular instance of the application.
- The health check and stickiness settings are requirements of this application. Not all applications will require these settings.

- Enabled HTTPS Listener.
  - Still within the instance tracking and scaling settings, enable HTTPS (port 443) with your own SSL certificate


Add listener

Listener port
443

Listener protocol
The transport protocol that the load balancer uses for routing incoming traffic from clients.
HTTPS

SSL certificate
Choose a certificate
*.thehkh.com - d74036bb-3053-44cb-8162-d791ab47c2ee
*.hkhinfo.xyz - df5d48ac-0c6c-4e63-95f9-2a0f6848b0e4

SSL policy
Choose a policy

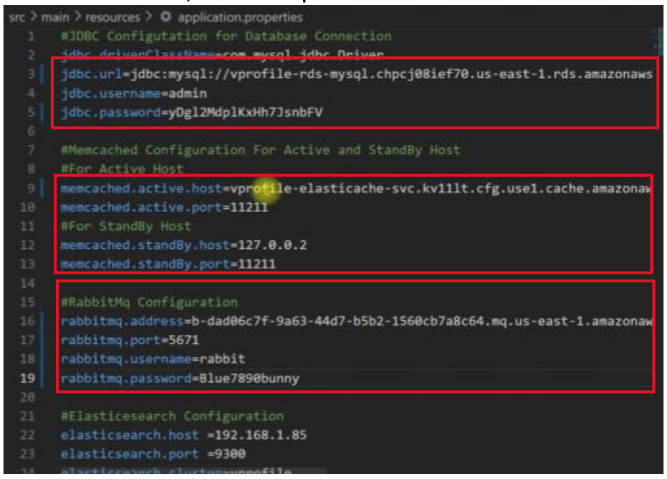
Default process
The process to which the listener routes traffic by default, when the message path doesn't match any custom listener rule
default

  - Scroll down and click apply

- Edit the backend security group settings to allow all traffic from the newly created Beanstalk instance security group

Step 7: Build and deploy application
- Clone the repository in a folder on my pc (using vscode). Switch to the main branch
- Edited the src/main/resources/application.properties file with the endpoints, username & password for the RDS,

Memcache, rabbitmq

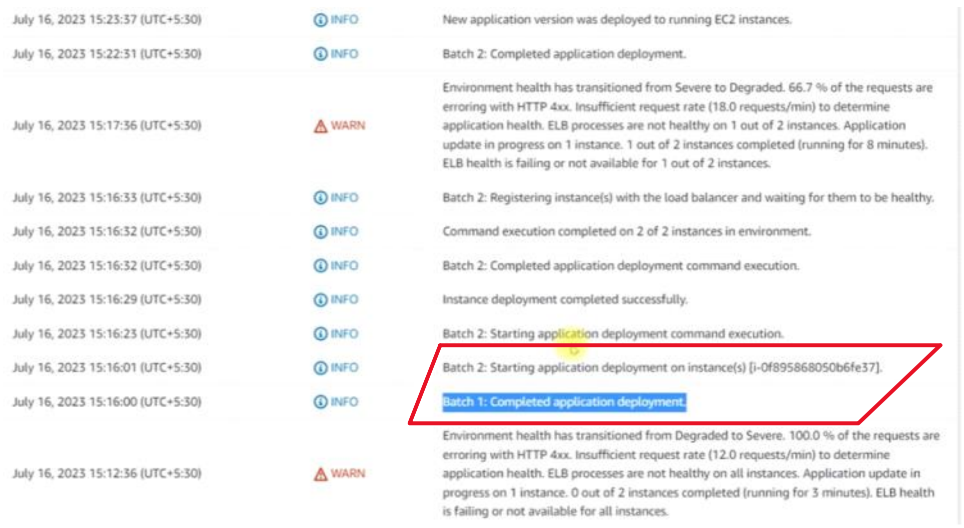


src > main > resources > application.properties

```
#JDBC Configutation for Database Connection
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://vprofile-rds-mysql.chpcj08ief70.us-east-1.rds.amazonaws
jdbc.username=admin
jdbc.password=yDgl2MdplKxHh7JsnbFV

#Memcached Configuration For Active and StandBy Host
#For Active Host
memcached.active.host=vprofile-elasticache-svc.kv11lt.cfg.use1.cache.amazonaw
memcached.active.port=11211
#For StandBy Host
memcached.standBy.host=127.0.0.2
memcached.standBy.port=11211

#RabbitMq Configuration
rabbitmq.address=b-dad06c7f-9a63-44d7-b5b2-1560cb7a8c64.mq.us-east-1.amazonaw
rabbitmq.port=5671
rabbitmq.username=rabbit
rabbitmq.password=Blue7890bunny

#Elasticsearch Configuration
elasticsearch.host =192.168.1.85
elasticsearch.port =9300
elasticsearch.cluster=vprofile
```

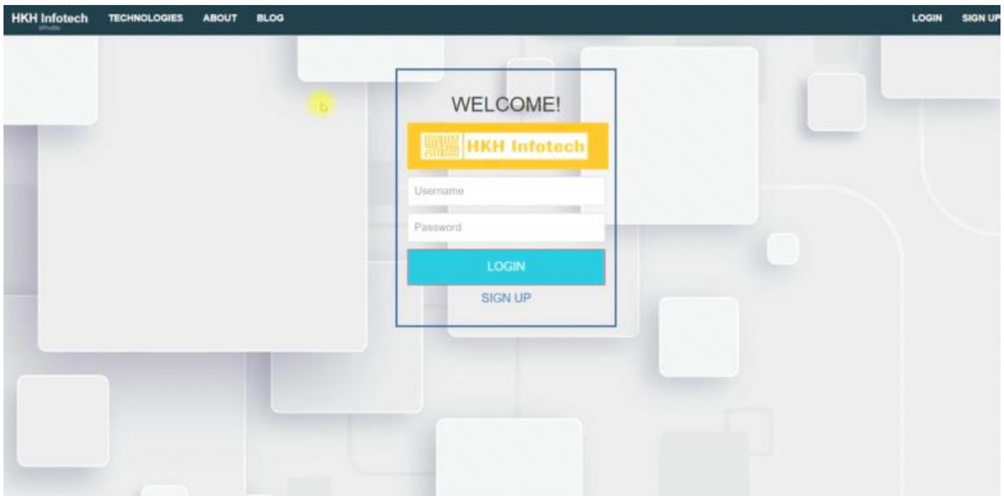
- In the folder where the github repo was cloned, I executed the following to generate the artifact:
  - `mvn --version`
  - `mvn install`

- I then uploaded artifact to the beanstalk environment. Inspecting the Beanstalk logs, you see the effect of rolling updates - the application is deployed one after the other on the ec2 instances. Because the deployment takes time, the health check fails at the beginning but after while passes through.



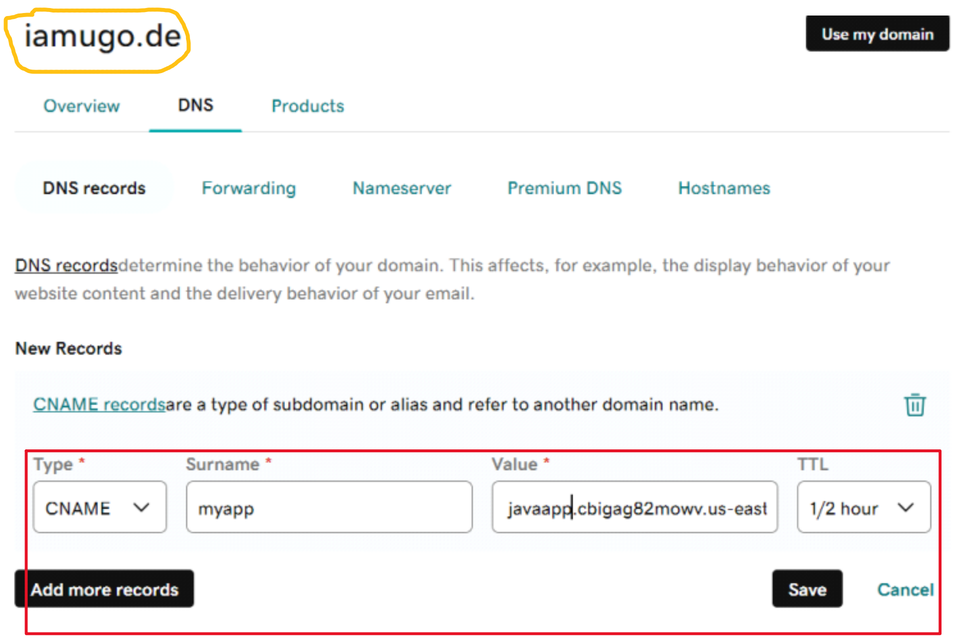
```
July 16, 2023 15:23:37 (UTC+5:30)   INFO   New application version was deployed to running EC2 instances.
July 16, 2023 15:22:31 (UTC+5:30)   INFO   Batch 2: Completed application deployment.
July 16, 2023 15:17:36 (UTC+5:30)   WARN   Environment health has transitioned from Severe to Degraded. 66.7 % of the requests are
                                           erroring with HTTP 4xx. Insufficient request rate (18.0 requests/min) to determine
                                           application health. ELB processes are not healthy on 1 out of 2 instances. Application
                                           update in progress on 1 instance. 1 out of 2 instances completed (running for 8 minutes).
                                           ELB health is failing or not available for 1 out of 2 instances.
July 16, 2023 15:16:33 (UTC+5:30)   INFO   Batch 2: Registering instance(s) with the load balancer and waiting for them to be healthy.
July 16, 2023 15:16:32 (UTC+5:30)   INFO   Command execution completed on 2 of 2 instances in environment.
July 16, 2023 15:16:32 (UTC+5:30)   INFO   Batch 2: Completed application deployment command execution.
July 16, 2023 15:16:29 (UTC+5:30)   INFO   Instance deployment completed successfully.
July 16, 2023 15:16:23 (UTC+5:30)   INFO   Batch 2: Starting application deployment command execution.
July 16, 2023 15:16:01 (UTC+5:30)   INFO   Batch 2: Starting application deployment on instance(s) [i-0f895868050b6fe37].
July 16, 2023 15:16:00 (UTC+5:30)   INFO   Batch 1: Completed application deployment.
July 16, 2023 15:12:36 (UTC+5:30)   WARN   Environment health has transitioned from Degraded to Severe. 100.0 % of the requests are
                                           erroring with HTTP 4xx. Insufficient request rate (12.0 requests/min) to determine
                                           application health. ELB processes are not healthy on all instances. Application update in
                                           progress on 1 instance. 0 out of 2 instances completed (running for 3 minutes). ELB health
                                           is failing or not available for all instances.
```

Clicking on the endpoint of the beanstalk shows the deployed application.



Step 8: Domain name mapping

- I then needed to map the endpoint of the Beanstalk environment to the domain name I created before - iamugo.de
  - In Godaddy, I created a CNAME record using the endpoint



iamugo.de

Use my domain

Overview DNS Products

DNS records Forwarding Nameserver Premium DNS Hostnames

DNS recordsdetermine the behavior of your domain. This affects, for example, the display behavior of your website content and the delivery behavior of your email.

New Records

CNAME recordsare a type of subdomain or alias and refer to another domain name.

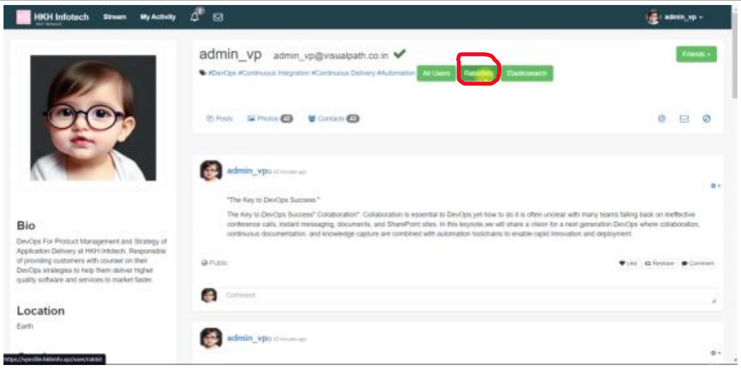
Type *	Surname *	Value *	TTL
CNAME	myapp	javaapp.cbigag82mowv.us-east	1/2 hour

Add more records Save Cancel

  - The url should now become https://myapp.iamugo.de, which shows the certificate is valid

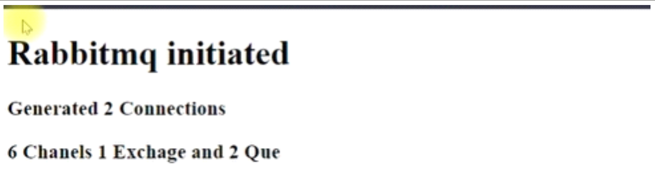
Step 9: Testing

- To test the application, login with the username & password (admin_vp) showing that the RDS database is well connected



User logged in --> validates working RDS DB.

Click on RabbitMQ at the top



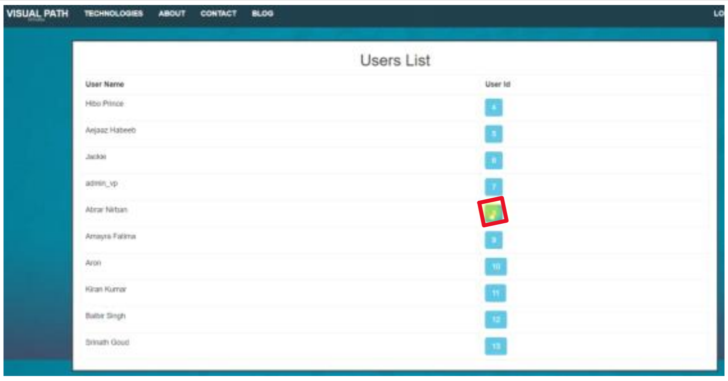
Rabbitmq initiated

Generated 2 Connections

6 Chanels 1 Exchage and 2 Que

Validates working RabbitMQ (AmazonMQ) service

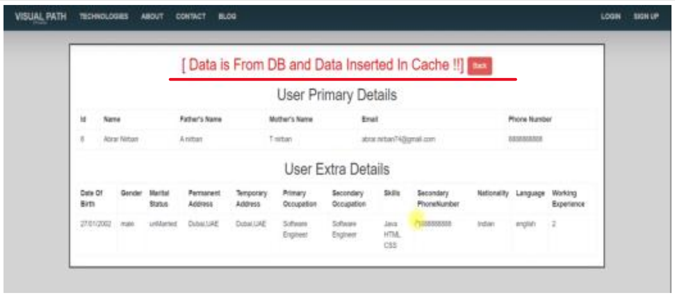
Return to the previous page.
Select All users



Users List

User Name	User Id
Hibo Prince	4
Aejaaz Habeeb	5
Jackie	6
admin_vp	7
Abrar Nirban	8
Amayra Fatima	9
Aron	10
Kiran Kumar	11
Balbir Singh	12
Srinath Goud	13

Select a user (user Id 8)



[Data is From DB and Data Inserted In Cache !!]

User Primary Details

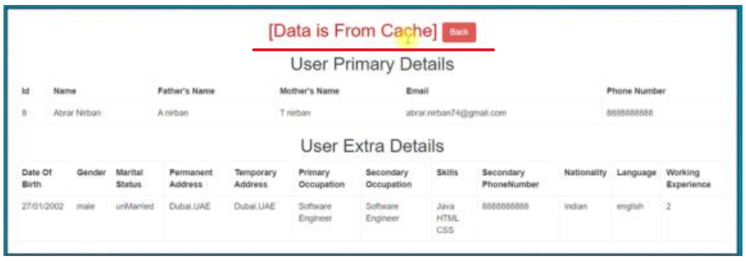
Id	Name	Father's Name	Mother's Name	Email	Phone Number
8	Abrar Nirban	A nirban	T nirban	abrar.nirban74@gmail.com	8888888888

User Extra Details

Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation	Secondary Occupation	Skills	Secondary PhoneNumber	Nationality	Language	Working Experience
27/01/2002	male	unMarried	Dubai,UAE	Dubai,UAE	Software Engineer	Software Engineer	Java HTML CSS	7888888888	Indian	english	2

Validates RDS DB + Memcached connectivity

Return to previous page. Click on same user


[Data is From Cache]

User Primary Details

Id	Name	Father's Name	Mother's Name	Email	Phone Number
8	Abrar Nirban	A nirban	T nirban	abrar.nirban74@gmail.com	8888888888

User Extra Details

Date Of Birth	Gender	Marital Status	Permanent Address	Temporary Address	Primary Occupation	Secondary Occupation	Skills	Secondary PhoneNumber	Nationality	Language	Working Experience
27/01/2002	male	unMarried	Dubai,UAE	Dubai,UAE	Software Engineer	Software Engineer	Java HTML CSS	8888888888	indian	english	2

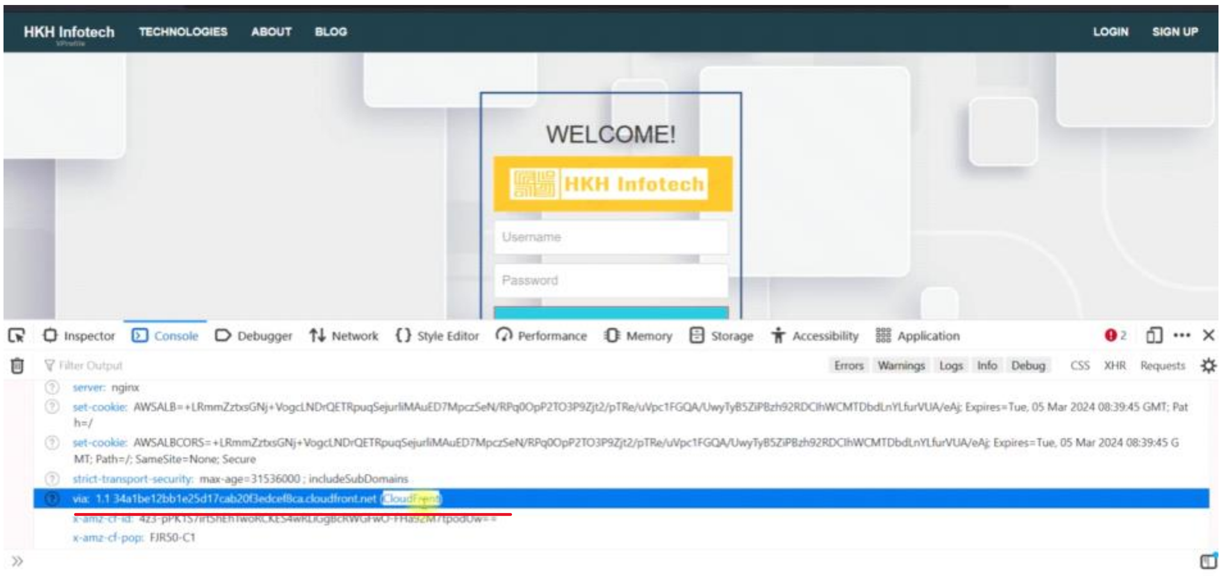
Validates cache working.

Step 10: Setup CloudFront

Cloudfront allows us to cache the application at AWS edge locations, giving users low latency access to our application. Firstly, I needed to create the CloudFront service and then added a Godaddy CNAME record for the application to use the CloudFront endpoint

- Visited CloudFront and created a new distribution
  - Next, I selected the elastic beanstalk load balancer for the application I just created
  - Allowed all the HTTP methods, legacy cache settings
  - For real-time purposes, it is advisable to enable WAF (not selected for this project because it falls outside free-tier)
  - In the alternate domain name (CNAME) I give the FQDN domain name I want to use (myapp.iamugo.de)
  - I then copied the URL of the domain name and added to a new CNAME record (in Godaddy) using this URL (remove the preceeding https://)

- To validate, I accessed the site using another browser and inspected the get request response



```
server: nginx
set-cookie: AWSALB=+LRmmZztxsGNj+VogcLNDrQETRpuqSejurliMAuED7MpczSeN/RPq0OpP2TO3P9Zjt2/pTRe/uVpc1FGQA/UwyTyB5ZiPBzh92RDClhWCMTDbdLnYLfurVUA/eAj; Expires=Tue, 05 Mar 2024 08:39:45 GMT; Pat
h=/
set-cookie: AWSALBCORS=+LRmmZztxsGNj+VogcLNDrQETRpuqSejurliMAuED7MpczSeN/RPq0OpP2TO3P9Zjt2/pTRe/uVpc1FGQA/UwyTyB5ZiPBzh92RDClhWCMTDbdLnYLfurVUA/eAj; Expires=Tue, 05 Mar 2024 08:39:45 G
MT; Path=/; SameSite=None; Secure
strict-transport-security: max-age=31536000 ; includeSubDomains
via: 1.1 34a1be12bb1e25d17cab20f3edcef8ca.cloudfront.net (CloudFront)
x-amz-cf-id: 4z3-pPK15/irtShEhTwoRCKES4wRLJGgBcRWGFwO-FHa52M7tpodOw==
x-amz-cf-pop: FJR50-C1
```

Challenges:

- Decision to use either Redis or Memcached was a challenge but since the application just required a simple, high-performance caching solution that and can trade off advanced features and persistence for simplicity and memory efficiency, I decided to use Memcached as against Redis. For applications that require advanced data structures, persistence replication and pub/sub messaging and transactions, I would have chosen Redis.