**Step-1: Create Keypair for Beanstalk EC2 Login**

* We will create a key pair to be used with Elastic Beanstalk. Go to EC2 console, on left menu select KeyPair -> Create key pair.

Name: vprofile-bean-key

* Remember where to download the private key, it will be used when logging in to EC2 via SSH.

**Step-2: Create Security Group for ElastiCache, RDS and ActiveMQ**

* Create a Security Group with name vprofile-backend-SG. Once it is created we need to edit Inbound rules:

All Traffic from `vprofile-backend-SG`

**Step-3: Create RDS Database**

**Create Subnet Group:**

* First we will create Subnet Groups with below properties:

Name: vprofile-rds-sub-grp

AZ: Select All

Subnet: Select All

**Create Parameter Group**

* We will create a parameter group to be used with our RDS instance. If we want to use default parameter group we don't need to create one. With parameter group, we are able make updates to default parameter for our RDS instance.

Parameter group family: mysql5.7

Type: DB Parameter Group

Group Name: vprofile-rds-para-grp

**Create Database**

* We will create RDS instance with below properties:

Method: Standard Create

Engine Options: MySQL

Engine version: 5.7.33

Templates: Free-Tier

DB Instance Identifier: vprofile-rds-mysql

Master username: admin

Password: Auto generate psw

Instance Type: db.t2.micro

Subnet grp: vprofile-rds-sub-grp

SecGrp: vprofile-backend-SG

No public access

DB Authentication: Password authentication

Additional Configuration

Initial DB Name: accounts

Keep the rest default or you may add as your own preference

* After clicking Create button, you will see a popup. Click View credential details and note down auto-generated db password. We will use it in our application config files.

**Step-4: Create ElastiCache**

**Create Parameter Group**

* We will create a parameter group to be used with our ElastiCache instance. If we want to use default parameter group we don't need to create one. With parameter group, we are able make updates to default parameters for our ElasticCache instance.

Name: vprofile-memcached-para-grp

Description: vprofile-memcached-para-grp

Family: memcached1.4

**Create Subnet Group:**

* First we will create Subnet Groups with below properties:

Name: vprofile-memcached-sub-grp

AZ: Select All

Subnet: Select All

**Create Memcached Cluster**

* Go to Get Started -> Create Clusters -> Memcached Clusters

Name: vprofile-elasticache-svc

Engine version: 1.4.5

Parameter Grp: vprofile-memcached-para-grp

NodeType: cache.t2.micro

# of Nodes: 1

SecGrp: vprofile-backend-SG

**Step-5: Create Amazon MQ**

* We will create Amazon MQ service with below properties:

Engine type: RabbitMQ

Single-instance-broker

Broker name: vprofile-rmq

Instance type: mq.t3.micro

username: rabbit

psw: bunnyhole789

Additional Settings:

private Access

VPC: use default

SEcGrp: vprofile-backend-SG

* Do not forget to note down tour username/pwd. You won't be able to see your Password again from console.

**Step-6: DB Initialization**

* Go to RDS instance copy endpoint.

vprofile-rds-mysql.chrgxmhxkprk.us-east-1.rds.amazonaws.com

* Create an EC2 instance to initialize the DB, this instance will be terminated after initialization.

Name: mysql-client

OS: ubuntu 18.04

t2.micro

SecGrp: Allow SSH on port 22

Keypair: vprofile-prod-key

Userdata:

#! /bin/bash

apt update -y

apt upgrade -y

apt install mysql-client -y

* SSH into mysl-client instance. We can check mysql version

mysql -V

* Before we login to database, we need to update vprofile-backend-SG Inbound rule to allow connection on port 3306 for mysql-client-SG After updating rule, try to connect with below command:

mysql -h vprofile-rds-mysql.chrgxmhxkprk.us-east-1.rds.amazonaws.com -u admin -p<db\_password>

mysql> show databases;

* Next we will clone our source code here to use script to initialize our database. After these commands we should be able to see 2 tables role, user, and user\_role.

git clone https://github.com/rumeysakdogan/vprofileproject-all.git

cd vprofileproject-all

git checkout aws-Refactor

cd src/main/resources

mysql -h vprofile-rds-mysql.chrgxmhxkprk.us-east-1.rds.amazonaws.com -u admin -padvPtIYOfqGe4T41MUXk accounts < db\_backup.sql

mysql -h vprofile-rds-mysql.chrgxmhxkprk.us-east-1.rds.amazonaws.com -u admin -padvPtIYOfqGe4T41MUXk accounts

show tables;

**Step-7: Create Elastic Beanstalk Environment**

* Our backend services are ready now. We will copy their endpoints from AWS console. These information will be used in our application.properties file

RDS:

vprofile-rds-mysql.chrgxmhxkprk.us-east-1.rds.amazonaws.com:3306

ActiveMQ: amqps://b-b7d7bbcb-3894-4af7-8048-726a9ceabc43.mq.us-east-1.amazonaws.com:5671

ElastiCache:

vprofile-elasticache-svc.eqmmsw.cfg.use1.cache.amazonaws.com:11211

**Create Application**

* Application in Elastic Beanstalk means like a big container which can have multiple environments. Since out app is Running on Tomcat we will choose Tomcat as platform.

Name: vprofilejavaapp-prod-rd

Platform: Tomcat

keep the rest default

Configure more options:

- Custom configuration

\*\*\*\*Instances\*\*\*\*

EC2 SecGrp: vprofile-backend-SG

\*\*\*\*Capacity\*\*\*\*

LoadBalanced

Min:2

Max:4

InstanceType: t2.micro

\*\*\*\*Rolling updates and deployments\*\*\*\*

Deployment policy: Rolling

Percentage :50 %

\*\*\*\*Security\*\*\*\*

EC2 key pair: vprofile-bean-key

**Step-8: Update Backend SecGrp & ELB**

* Our application instances created by BeanStalk will communicate with Backend services. We need update vprofile-backend-SG to allow connection from our appSecGrp created by Beanstalk on port 3306, 11211 and 5671

Custom TCP 3306 from Beanstalk SecGrp(you can find id from EC2 insatnces)

Custom TCP 11211 from Beanstalk SecGrp

Custom TCP 5671 from Beanstalk SecGrp

* In Elastic Beanstalk console, under our app environment, we need to clink Configuration and do below changes and apply:

Add Listener HTTPS port 443 with SSL cert

Processes: Health check path : /login

**Step-9: Build and Deploy Artifact**

* Go to directory that we cloned project, we need to checkout aws-refactor branch. Update below fields in application.properties file with correct endpoints and username/pwd.

vim src/main/resources/application.properties

\*\*\*\*\*Updates\*\*\*\*\*

jdbc.url

jdbc.password

memcached.active.host

rabbitmq.address

rabbitmq.username

rabbitmq.password

* Go to root directory of project to the same level with pom.xml file. Run below command to build the artifact.

mvn install

**Upload Artifact to Elastic Beanstalk**

* Go to Application versions and Upload the artifact from your local. It will autmatically upload the artifact to the S3 bucket created by Elasticbeanstalk.
* Now we will select our uploaded application and click Deploy.
* Let's check if our application deployed successfully.

**Step-10: Create DNS Record in Route53 for Application**

* We will create an A record which aliasing Elastic Beanstalk endpoint.
* Now we can reach our application securely with DNS name we have given.

**Step-11: Create Cloudfront Distribution for CDN**

* Cloudfront is Content Delivery Nettwork service of AWS. It uses Edge Locations around the world to deliver contents globally with best performance. We will to CloudFront and create a distribution.

Origin Domain: DNS record name we created for our app in previous step

Viewer protocol: Redirect HTTP to HTTPS

Alternate domain name: DNS record name we created for our app in previous step

SSL Certificate:

Security policy: TLSv1

* Now we can check our application from browser.