**4. Re-Architecturing Web App on AWS Cloud**

Scenario: there is one multi tier application running on onpremises/vm/cloud. To maintain it there are separate teams🡪 cloud computing team/ virtualization team/ DC OPS team/Monitoring team/ sys admin team

Text

Description automatically generated

Solution:

Use PAAS/SAAS of cloud

AWS services:

1. EC2
2. Elastic beanstalk
3. S3
4. RDS
5. Elastic cache
6. Active mq
7. Route 53
8. Cloud front

Flow:

Diagram

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

1. Create a key pair (pem file)
2. Create SG 🡪 vprofile-backend-SG

Create one dummy rule 🡪 ssh 22 and then save this SG and edit again with

All traffic select this SG

This is to allow internal traffic in backend servers

BACKEND SERVICES:

* RDS:

1. AWS 🡪 RDS
2. Subnet groups 🡪 create db subnet group 🡪

Name 🡪 vprofile-rds-sub-group

1. Select vpc
2. AZ🡪 select all AZ and all subnets
3. Create
4. Parameter group 🡪 create parameter group 🡪
5. Parameter group family 🡪 mysql 5.6
6. Group name: vprofile-rds-prara-group
7. Create
8. Go to databases 🡪 create database 🡪 standard create
9. Engine type: mysql

\*note: Aurora: its 5 times faster than mysql and 3 times faster than postgres with 1/10 th of the cost

1. Version : mysql 5.6.34
2. Templates: Dev/test

For production select – production

1. Db-instance-identifier : vprofile-rds-mysql
2. Username: admin
3. Pw: autogenerate
4. Db instance class: Burstable : db.t2.micro

For production : memory optimized classes (r series)

1. Enable storage autoscaling
2. Maximum storage threshold : 1000 gb
3. Multi-az deployment: do not create standby instance

For production : create standby instance

1. Additional connectivity configuration
2. Select subnet group
3. Public access : no
4. Select SG
5. Additional configuration
6. Initial database name: accounts
7. Select parameter group
8. Enable enhanced monitoring
9. Granularity: 60 s
10. Log exports: select all
11. Enable deletion protection
12. Create database
13. Copy credentials somewhere

* ElastiCache

1. Parameter group 🡪 create parameter group
2. Family: memcached1.4
3. Name: vprofile-memcached-para-group
4. Create

We can edit these parameters as per need

1. Subnet group 🡪 create
2. Name: vprofile-memcached-sub-group
3. Select vpc
4. Select all zones and subnets
5. Create
6. Memcached 🡪 create
7. Select Memcached
8. Name: vprofile-elasticache-svs
9. Engine version compatibility: 1.4.5
10. Select parameter group
11. Node type: t2.micro
12. Number of nodes: 1

For production : 2

1. Advanced Memcached settings
2. Select subnet group
3. Select Security group
4. Topic for sns 🡪 select ‘default cloud watch alrams topic’
5. Create

* Active mq (rabbit mq):

1. Aws 🡪 active mq 🡪 get started
2. Single instance broker
3. Storage type: durability optimized
4. Next
5. Broker name: vprofile-rmq
6. Broker instance type: mq.t2.micro
7. Set username and pw ---------keep noted
8. Additional settings
9. Logs: select general and audit
10. Select SG—vprofile-backend-sg
11. Public accessibility : no
12. Give tags
13. Create broker

* DB initialization :

1. Copy endpoint of rds and note
2. Launch new ec2 in same region (ubuntu18/t2.micro)
3. In same vpc
4. SG: 22 ssh from my ip (name this SG: mysql\_sg)
5. Create and connect
6. # sudo apt update
7. # sudo apt install mysql-client -y
8. Edit backend SG : allow mysql 3306 from mysql\_sg
9. # mysql -h **rds\_endpoint** -u admin -p**password**
10. > show databases;
11. Exit
12. Clone the source code

$ git clone <http://github.com/devopshydclub/vprofile-project.git>

1. $ cd vprofile
2. $ git checkout aws-Refactor
3. $ cd src/main/resources
4. $ mysql -h **rds\_endpoint** -u admin -p**password** accounts < db\_backup.sql
5. $ mysql -h **rds\_endpoint** -u admin -p**password**
6. > show tables;

* Note down Memcached configuration endpoint
* Note down active mq AMQP endpoint (**after //**)
* Delete that ec2 instance created to initialize rds
* Elastic beanstalk

1. Aws 🡪 beanstalk 🡪
2. Create application
3. Application name: vprofile
4. Platform 🡪 tomcat
5. Configure more option
6. Instances🡪 edit 🡪 select SG (backend sg) 🡪 root volume type: General purpose SSD 🡪 size: 8 GB 🡪 save
7. Capacity 🡪 edit 🡪 environment: load balanced 🡪 min:2 max: 8 🡪 save
8. Rolling updates and deployments: edit 🡪 deployment policy: rolling 🡪 percentage: 50% 🡪 save
9. Security 🡪 edit 🡪 ec2 keypair: select key that we created🡪 save
10. Monitoring: edit 🡪 system : basic (for production: enhanced) 🡪 enable log streaming 🡪 lifecycle: delete logs upon termination 🡪 save
11. Edit tags
12. Create app
13. Open endpoint and test

* Update backend sg:

1. Copy vprofile-bean-inst SG ID
2. Go to backend SG
3. Edit inbound rules
4. Mysql from vprofile-bean-inst SG ID
5. 11211 from vprofile-bean-inst SG ID
6. 5671 from vprofile-bean-inst SG ID
7. Save
8. Environments: vprofile-env 🡪 configuration 🡪 table view 🡪 load balancer : edit 🡪 listeners: add listener 🡪 port: 443 🡪 https 🡪 select certificate 🡪 add
9. Processes 🡪 select default 🡪 actions 🡪 path: /login 🡪 save
10. apply

* Build and deploy artifacts:

1. We’ll build our code locally
2. We need to have maven and jdk installed on our laptop
3. Clone project repo

$ git clone <http://github.com/devopshydclub/vprofile-project.git>

1. $ cd vprofile-project
2. $ git checkout aws-Refactor
3. $ cd src/main/resources
4. $ vim application.properties
5. Remove db01 and replace it with rds endpoint
6. Change pw
7. Replace mc01 with Memcached endpoint : remove endpoint
8. Replace rmq01 with its endpoint: remove endpoint
9. Change pw
10. Save
11. $ cd ../../..
12. $ mvn install
13. Aws 🡪 beanstalk 🡪 Application versions 🡪 upload 🡪 version label: vprofile-v2.5
14. Choose file 🡪 upload vprofile-v2.war
15. Upload
16. Select version 🡪 actions 🡪 deploy
17. Environments 🡪 openendpoint : loginpage shall get open
18. Open go daddy portal to add CNAME record
19. CNAME vprofile endpoint
20. Try to open that website with https
21. Login and check
22. Aws 🡪 beanstalk 🡪 configuration 🡪 load balancer: edit 🡪 processes: select default 🡪 Actions: edit
23. Enable ‘stickyness policy enabled’
24. Save 🡪 apply
25. Refresh url

* Cloudfronts:

1. Aws 🡪 cloudfront 🡪 create distribution 🡪 web: get started 🡪
2. Origin domain name: vprofile.groofy.in
3. Origin protocol policy: match viewer
4. Allowed http methods: GET,HEAD,OPTIONS,PUT,POST,PATCH,DELETE
5. Distribution settings: select price class according to your user location
6. Alternate domain name: vprofile.groofy.in
7. Ssl certificate: custom ssl certificate
8. Chose certificate
9. Security policy : TLSv1
10. Create distribution
11. Wait till status :deployed
12. Open url in firefox browser 🡪 press f12
13. Check ‘via…………….’