**LIFT AND SHIFT APPLICATION WORKLOAD**

* Architecture will consists of:

1. EC2 instances
2. ELB
3. Autoscaling
4. S3
5. Amazon certificate manager
6. Route 53

* Flow of execution:

Graphical user interface, text, application

Description automatically generated

* Login to AWS 🡪 ACM 🡪 validate certificate from domain purchased in Godaddy
* create security groups:

1. ec2 🡪 security groups 🡪 create
2. name: **vprofile-ELB-SG**

allow following inbound rules:

http 80 anywhere

https 443 anywhere

1. name: **vprofile-app-SG**

allow following inbound rules:

http 8080 select vprofile-ELB-SG 🡨 to allow traffic from ELB only

ssh 22 myip

1. name: **vprofile-backend-SG**

allow following inbound rules:

MYSQL/Aurora 3306 select vprofile-app-SG 🡨 to allow traffic from tomcat only

Custom TCP 11211 select vprofile-app-SG 🡨 Memcached

Custom TCP 5672 select vprofile-app-SG 🡨 Rabbitmq

SSH 22 my ip

Save this 3rd SG and again edit inbound rules :

All traffic select vprofile-backend-SG 🡨 **this is to allow internal traffic to flow on all ports**

* create a keypair

ec2🡪 keypairs 🡪 pem

save this in local computer

* create ec2 instances :

1. Mysql db instance
2. centos7 🡪 t2 micro 🡪 copy following user data

#!/bin/bash

DATABASE\_PASS='admin123'

sudo yum update -y

sudo yum install epel-release -y

sudo yum install git zip unzip -y

sudo yum install mariadb-server -y

# starting & enabling mariadb-server

sudo systemctl start mariadb

sudo systemctl enable mariadb

cd /tmp/

git clone -b vp-rem https://github.com/devopshydclub/vprofile-repo.git

#restore the dump file for the application

sudo mysqladmin -u root password "$DATABASE\_PASS"

sudo mysql -u root -p"$DATABASE\_PASS" -e "UPDATE mysql.user SET Password=PASSWORD('$DATABASE\_PASS') WHERE User='root'"

sudo mysql -u root -p"$DATABASE\_PASS" -e "DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1')"

sudo mysql -u root -p"$DATABASE\_PASS" -e "DELETE FROM mysql.user WHERE User=''"

sudo mysql -u root -p"$DATABASE\_PASS" -e "DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%'"

sudo mysql -u root -p"$DATABASE\_PASS" -e "FLUSH PRIVILEGES"

sudo mysql -u root -p"$DATABASE\_PASS" -e "create database accounts"

sudo mysql -u root -p"$DATABASE\_PASS" -e "grant all privileges on accounts.\* TO 'admin'@'localhost' identified by 'admin123'"

sudo mysql -u root -p"$DATABASE\_PASS" -e "grant all privileges on accounts.\* TO 'admin'@'%' identified by 'admin123'"

sudo mysql -u root -p"$DATABASE\_PASS" accounts < /tmp/vprofile-repo/src/main/resources/db\_backup.sql

sudo mysql -u root -p"$DATABASE\_PASS" -e "FLUSH PRIVILEGES"

# Restart mariadb-server

sudo systemctl restart mariadb

#starting the firewall and allowing the mariadb to access from port no. 3306

sudo systemctl start firewalld

sudo systemctl enable firewalld

sudo firewall-cmd --get-active-zones

sudo firewall-cmd --zone=public --add-port=3306/tcp --permanent

sudo firewall-cmd --reload

sudo systemctl restart mariadb

1. select existing SG : vprofile-backend-SG
2. launch
3. connect to vm using gitbash
4. to check user data run following command

# curl <http://169.254.169.254/latest/user-data>

1. check status

# systemctl status mariadb

1. to check processes

# ps -ef

1. login to mysql

# sudo -i

# mysql -u root -p

Pw: admin123

# show databases;

accounts

# use accounts;

# show tables;

Exit

1. Memcached :
2. Ec2 🡪 centos7 🡪 t2.micro
3. Use following user data script

#!/bin/bash

sudo yum install epel-release -y

sudo yum install memcached -y

sudo systemctl start memcached

sudo systemctl enable memcached

sudo systemctl status memcached

sudo memcached -p 11211 -U 11111 -u memcached -d

1. select existing SG : vprofile-backend-SG
2. launch and connect
3. check status of memcahced

# sudo -i

# systemctl status Memcached

To check port

# ss -tunpl | grep 11211

exit

1. Rabbitmq
2. Ec2 🡪 centos7 🡪 t2.micro
3. select existing SG : vprofile-backend-SG
4. Use following user data script

#!/bin/bash

sudo yum install epel-release -y

sudo yum update -y

sudo yum install wget -y

cd /tmp/

wget http://packages.erlang-solutions.com/erlang-solutions-1.0-1.noarch.rpm

sudo rpm -Uvh erlang-solutions-1.0-1.noarch.rpm

sudo yum -y install erlang socat logrotate

wget https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.8.8/rabbitmq-server-3.8.8-1.el6.noarch.rpm

sudo rpm --import https://www.rabbitmq.com/rabbitmq-signing-key-public.asc

sudo rpm -Uvh rabbitmq-server-3.8.8-1.el6.noarch.rpm

sudo systemctl start rabbitmq-server

sudo systemctl enable rabbitmq-server

sudo systemctl status rabbitmq-server

sudo sh -c 'echo "[{rabbit, [{loopback\_users, []}]}]." > /etc/rabbitmq/rabbitmq.config'

sudo rabbitmqctl add\_user test test

sudo rabbitmqctl set\_user\_tags test administrator

sudo systemctl restart rabbitmq-server

1. launch and connect
2. Check status of rabbitmq

# sudo -i

# systemctl status rabbitmq-server

* Route 53

1. Copy private ip’s of above 3 servers in notepad
2. Open Route 23
3. Create hosted zone 🡪 private hosted zone 🡪
4. Domain name: vprofile.in
5. Select region and vpc according to our vm region
6. Create hosted zone
7. Create record
8. Simple routing 🡪

blog : db01

endpoint: ip address

give ip of db01

similarly add 2 more records for

blog: mc01

blog: rmq01

* these are simple Name to IP A records which will be use by tomcat server.
* application.properties file will contain these names and no ip's
* even if we replace backend servers then also there will be no need to make any change in application server
* launch Tomcat ec2 server

1. ec2 🡪 ubuntu18 🡪 t2.micro
2. use following user data script

#!/bin/bash

sudo apt update

sudo apt upgrade -y

sudo apt install openjdk-8-jdk -y

sudo apt install tomcat8 tomcat8-admin tomcat8-docs tomcat8-common git -y

1. use existing SG: vprofile-app-SG

* Build and Deploy artifacts

1. To build project locally we need to install jdk8 and maven
2. Clone project repo
3. # cd /src/main/resources
4. # vim application.properties

Replace db01, mc01, rmq01 with

db01.vprofile.in, mc01 vprofile.in, rmq01 vprofile.in

this is as per our DNS entries in Route53

1. # cd ../../../
2. # mvn install

Target folder will get create

1. To deploy this to aws s3 we need to configure aws cli here in local machine

Open powershell

# choco install awscli -y

1. Go to aws and create IAM user 🡪 programmatic access 🡪 permission 🡪 attach existing policy 🡪 s3 full access
2. Go to project repo
3. Open gitbash

# aws configure

Give access key and secret key🡪 o/p🡪 json

1. To create s3 bucket run following command

# aws s3 mb s3://vprofile-artifact-storage

1. # cd /target
2. # aws s3 cp vprofile-v2.war s3:// vprofile-artifact-storage/ vprofile-v2.war
3. To validate

# aws s3 ls s3:// vprofile-artifact-storage/

1. Create a role :

IAM 🡪 roles 🡪 create a role 🡪 EC2 🡪 permission 🡪 s3 full access 🡪 role name: vprofile-artifact-storage-role 🡪 create role

1. Go to ec2 🡪 vprofile-app01 🡪 Actions 🡪 instance settings 🡪 Modify IAM role 🡪 select vprofile-artifact-storage-role 🡪 save
2. Connect to app01 using gitbash

# systemctl tomcat8 status

# cd /var/lib/tomcat8/webapps

# systemctl stop tomcat8

# rm -rf ROOT

# apt install awscli -y

# cd

# aws s3 cp s3://vprofile-artifact-storage/vprofile-v2.war /var/lib/tomcat8/webapps/ROOT.war

# systemctl start tomcat8

To verify

# ls /var/lib/tomcat8/webapps/

To validate that we are able to connect backend servers or not

# apt install telnet -y

# telnet db01.vprofile.in 3306

* Setup load balancer

1. As we are going to use application load balancer we need to configure target group first
2. Target groups 🡪 create target group 🡪 select ‘instances’🡪 name: vprofile-app-tg
3. Port: 8080
4. Health check path : /login
5. Advanced health check settings : override 🡪 8080
6. Healthy threshold : 3
7. Next
8. Available instances: select ‘vprofile-app01’
9. Select ‘include as pending below’
10. Create target group
11. Load balancers 🡪 create Load balancers 🡪 select ‘Application load balancer’ 🡪 name: vprofile-prod-elb 🡪 protocols: http, https
12. Select all availability zones
13. Configure security setting
14. ‘choose a certificate from ACM’
15. Select certificate
16. SG: vprofile-ELB-SG
17. Configure routing:

Target group: Existing target group

Register targets

1. Review 🡪 create
2. Copy DNS name
3. Go to Godaddy 🡪 dns settings

Type: CNAME

Host: vprofileapp

Points to: paste DNS name

1. Save

Dns purchased: waiin.com

So that ‘vprofileapp.waiin.com’ will be our URL

1. Verify it in browser
2. Username: admin\_vp

Pw: admin\_vp

1. Verify everything

* Autoscaling group:

1. Ec2🡪 select vprofile-app01 🡪 Actions 🡪 Image 🡪 create image 🡪

Name: vprofile-app-image 🡪 create image

1. Launch configurations 🡪 create Launch configurations 🡪 Name: vprofile-app-LC 🡪 select AMI
2. Instance type: t2.micro
3. IAM instance profile: vprofile-artifact-storage-rule
4. Check 🡪 Enable ec2 instance detailed monitoring within cloudwatch
5. Security group: vprofile-app-sg
6. Select keypair
7. Launch configuration
8. Autoscaling group 🡪 create 🡪 name: vprofile-app-ASG 🡪 switch to launch configuration
9. Select our launch configuration 🡪 next
10. Select all subnets 🡪 Next check ‘enable load balancing’
11. Select target group
12. Health check 🡪 check ‘ELB’
13. Next
14. Desired capacity : 1

Min: 1

Max: 4

1. To automatically scale in and scale out select : ‘Target tracking scaling policy’
2. If you don’t want to get ec2 instance deleted automatically i.e to avoid scale in then check

Enable instance scale-in protection

Next

1. Create ASG
2. Delete previous app01 ec2 instance
3. Go to target group 🡪 new instance will get in healthy status

* Validate:

Open url in browser

<https://vprofileapp.waiin.com>