London Stock Exchange Group

Assignment

Capital Markets Applications

Keerthana Vijekumar

Contents

Introduction and the solution

A **capital market** is an organized market in which both individuals and business entities buy and sell debt and equity securities. It refer to the places where savings and investments are moved between suppliers of capital and those who are in need of capital. Here security and reliability is an essential thing to be considered. Therefore, my choice is to go with AWS.
AWS stands for Amazon Web Services. The AWS service is provided by Amazon that uses distributed IT infrastructure to provide different IT resources available on demand.
AWS is a cloud computing platform which is globally available. It is based on the concept of Pay-As-You-Go. Global infrastructure is a region around the world in which AWS is based. 77 Availability Zones within 24 geographic regions around the world and has announced plans for 18 more Availability Zones and 6 more AWS Regions in Australia, India, Indonesia, Japan, Spain, and Switzerland. Each region consists of 2 or more availability zones. Availability Zones, Region, Edge locations and Regional Edge Caches are components in a global infrastructure.

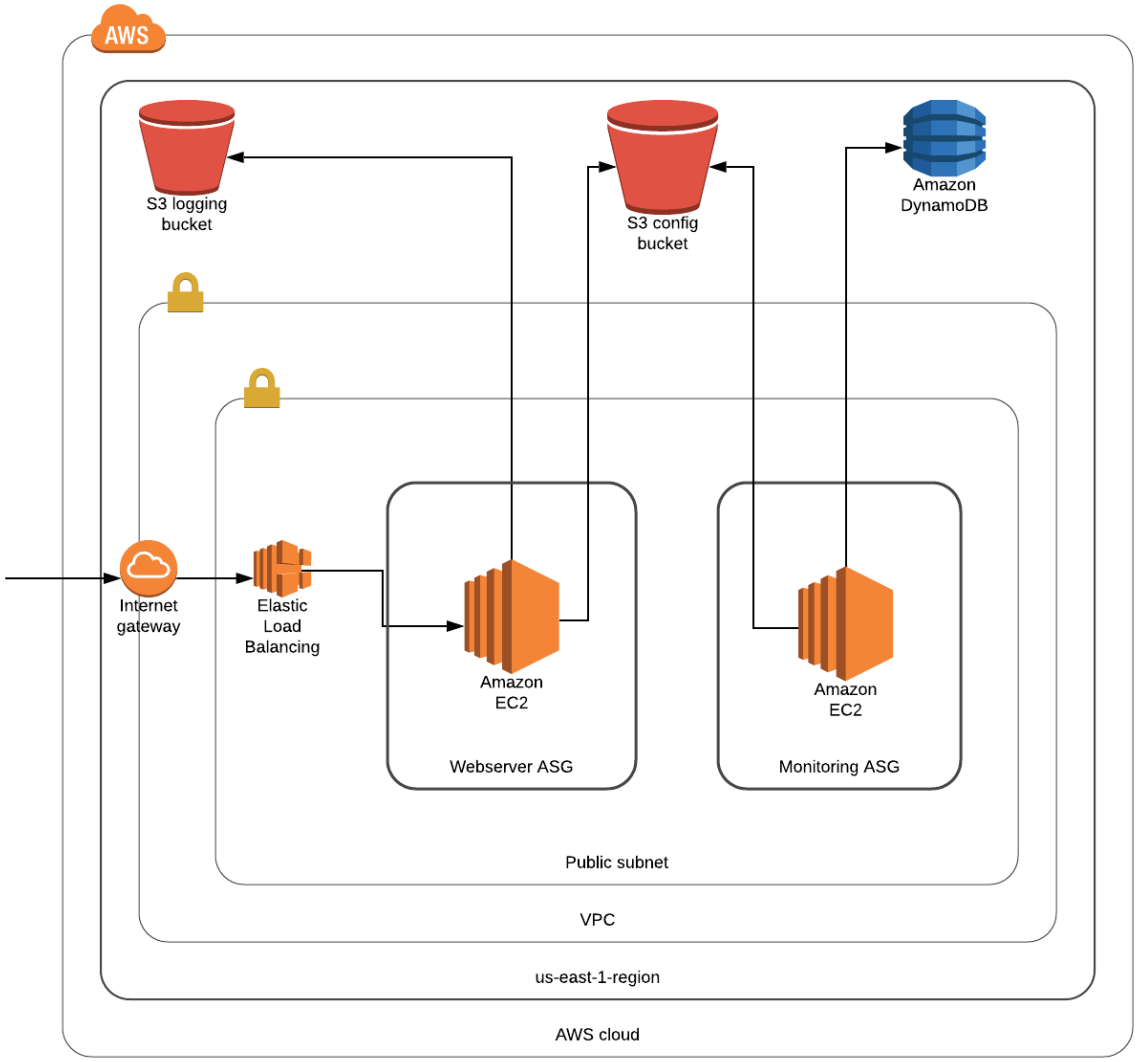
Using AWS infrastructure our team can have benefits such as scalability, availability, reliability and specially cost-effectiveness. AWS requires no upfront investment, long-term commitment, and minimum expense when compared to traditional IT infrastructure that requires a huge investment.

Some of key points

- To ensure availability and reliability of this setup, used an ASG, Launch Config with fully automated userdata scripts and Load Balancer which would run health checks every 5 mins, in case any instance fails the health check, it will be kicked out and replaced by a new instance.
- Automated EC2 instance bootstrap using BASH and EC2 UserData
- Web server can be scaled to N number of instances without any issue, monitoring server script is written in a dynamic way which allows it to monitor all the new and old instances without any script change.
- In order to make sure availability under load, added an ASG and divided traffic using a Load Balancer. An ELB (Elastic Load Balancer) balances the incoming traffic.
- Tested all scripts vigorously using a test EC2 instance to make sure it's reliable.
- To make the website available only 9*5, I added a Scheduled Scale Down policy under ASG.
- If configured, Auto scaling and elastic load balancing can automatically scale up or down, when demand increases or decreases respectively. ASGs are ideal for handling unpredictable or very high loads. Because of this we can reduce cost and increase user satisfaction.

- In order to be cost effective I used cheaper DynamoDB which is fast and cheap to store monitoring statuses.
- To decrease the cost furthermore used small EC2 Spot Instances and Scheduled Scale Down when not needed to reduce cost.
- To secure the setup, Encrypted Private S3 buckets were used to store keys and configs
- Used proper Security Group rules for instances by opening only the required ports.
- Created proper IAM roles for the instances, giving permission only to the resources and actions that are required.
- Access to SSH is limited only to the developers machine

Architecture Diagram



Steps

Pre-Steps

1. Configured AWS CLI for my WSL2.0 ubuntu subsystem to easily copy files to S3.

Web Server

1. Created private S3 bucket with the name “capital-market-logging-bucket” in us-east-1 region to store collected logs.
2. Created private S3 bucket with the name “capital-market-config-bucket” in us-east-1 region to store configurations and keys.
3. Prepared script to collect, compress and push logs and content to s3.

```
#!/bin/bash

#Create name for the object to be stored in s3
dateName=$(date -u +%Y-%m-%d)
# Change to log script directory which was created in the userdate
script
cd /etc/logging/
# Delete if the log directory is already there. Redirect all output to
/dev/null to keep the script clean
rm -rf logDir &> /dev/null
# Create Fresh Directory
mkdir logDir
# Copy httpd Server logs to Log Directory
cp /var/log/httpd/* ./logDir/
# Get Server Content
curl localhost > ./logDir/content.html
# Compress Collected log files and content. Date name to create unique
log files
tar -cvzf "log-$dateName.tar.gz" ./logDir
# Delete temp logDir Directory
rm -rf ./logDir

# Copy Compressed log files to S3
aws s3 cp "/etc/logging/log-$dateName.tar.gz" s3://capital-market-
logging-bucket/webserver/

# Check if AWS S3 Copy Failed
if [ $? -ne 0 ]
then
cat <<EOF | ssmtp keerthuvije@gmail.com
Subject: Log File Copy Failed
```

```
Log file copy to AWS S3 for the date : $dateName has failed.
EOF
else
   rm "/etc/logging/log-$dateName.tar.gz"
fi
```

4. Copied created script into S3 Config Bucket under the container “webserver”.
5. Created SSMTP configurations.

```
root=<YOUR EMAIL>
mailhub=smtp.gmail.com:587
hostname=Monitoring
AuthUser=<YOUR EMAIL>
AuthPass=<YOUR EMAIL PASSWORD>
UseTLS=YES
UseSTARTTLS=YES
FromLineOverride=YES
TLS_CA_File=/etc/pki/tls/certs/ca-bundle.crt
```

6. Copied SSMTP configuration to config S3 bucket under the container “webserver”.
7. Created User Data script which would run at the start of the instance to configure and start httpd server.

```
#!/bin/bash

## COMMON
# update yum packages and install httpd server
sudo yum update -y
sudo yum install httpd -y

## EMAIL
# Enable Rhel Yum repository
# Need to enable since Amazon Yum repo doesn't have ssmtp package
sudo amazon-linux-extras install epel -y
# Install Email client
sudo yum install ssmtp -y
# Delete current ssmtp conf if available
if [ -f /etc/ssmtp/ssmtp.conf ]
then
   sudo rm /etc/ssmtp/ssmtp.conf
   sudo rm /etc/ssmtp/revaliases
fi
# Copy email config from config s3 bucket
sudo aws s3 cp s3://capital-market-config-bucket/webserver/ssmtp.conf
/etc/ssmtp/ssmtp.conf
```

```
sudo aws s3 cp s3://capital-market-config-bucket/webserver/revaliases
/etc/ssmtp/revaliases

## WEBSERVER
# Create index.html with Hello World Header at /var/www/html/index.html
echo "<h1>Hello World</h1>" > /var/www/html/index.html
# Start httpd server
sudo systemctl start httpd
# Enable Auto start of httpd server in case of restarts
sudo chkconfig httpd

## LOGGING
# Create Logging Script dir
sudo mkdir -p /etc/logging/
# Copy logging script from config s3 bucket
sudo aws s3 cp s3://capital-market-config-bucket/webserver/logging.sh
/etc/logging/script.sh
# Allow execute permission to script.sh
sudo chmod 755 /etc/logging/script.sh
# Configure crontab to run daily at 23:30
# Redirect and append script logs under /var/log/cron.log
cat <<EOF > /cron.txt
30 23 * * * /etc/logging/script.sh >> /var/log/cron.log
EOF
sudo crontab /cron.txt
```

8. Tested the scripts using a test EC2 instance will Admin privileges. Deleted the instance afterwards.
9. Created the following IAM Policy which would allow web server instance to get config data from config s3 bucket and also will allow the instance to write log files to logging bucket.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "VisualEditor0",
           "Effect": "Allow",
           "Action": "s3:GetObject",
           "Resource": "arn:aws:s3:::capital-market-config-
bucket/webserver/*"
       },
```

```
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:RestoreObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::capital-market-logging-
bucket/webserver/*"
        }
    ]
}
```

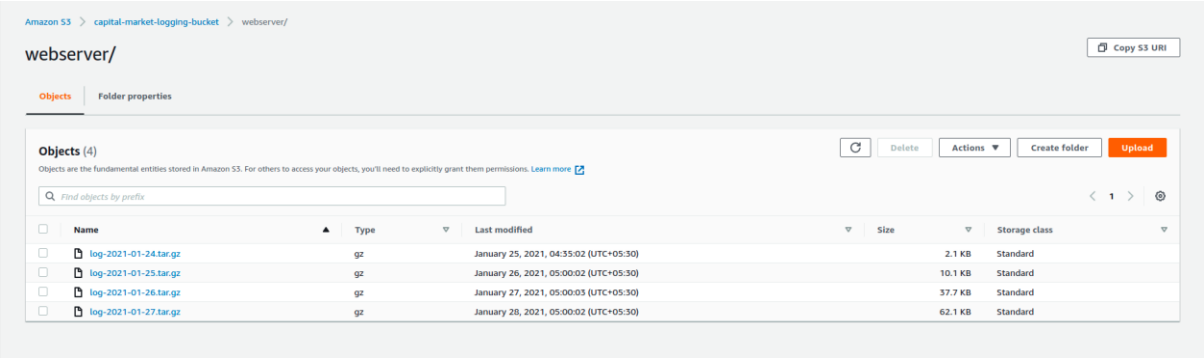
10. Created an IAM role using the above policy with the name “webserver”
11. Created a keypair called “keerthan-key.pair” and saved it in the config s3 bucket under the container “monitoring”
12. Created a launch configuration which uses the AmazonLinux2 AMI on t2.micro instances and also gave the created role and keypair which were created in the previous steps.
    a. Added the prepared userdata script.
    b. Used Spot Instances to Reduce cost.
    c. Used Default VPC and Subnets under the Launch Configuration to make the setup simple.
    d. Allowed the following rules under the Security Group
        i. Allowed HTTP (80) traffic from anywhere
        ii. Allowed SSH (22) traffic only from my IP to preserve security
        iii. Allowed SMTP (25) traffic
        iv. Allowed Port 587, which is gmail SMTP server port
13. Using this launch configuration, created an Auto Scaling Group named “webserver”. Used an Auto Scaling group coupled with a Load Balancer to make sure high availability and Scalability.
14. Configured a Load Balancer named “webserver-lb” and forwarded HTTP traffic to the ASG instances using a target group.
15. Configured ELB Based Health checks to the Auto Scaling Group to maintain high durability.
16. Configured Schedules actions under the created Auto Scaling Group to scale down to 0 instances at 5.15PM and scale back out to 1 instance at 8.45AM.
17. Confirmed Instance created in the AWS Console and made sure that instance is created with correct values.
18. Using the downloaded “keerthan-key.pair”, SSH ed into one of the created instances and made sure that following is working
    a. Checked if httpd server is running. *sudo systemctl status httpd*
    b. Checked if the cronjob has been created. *sudo crontab -l*
    c. Ran logging script manually and checked if compressed logs fails are available in the logging S3 bucket. *sudo /etc/logging/script.sh*

d. Checked if email configuration is correct by sending a dummy test email. *echo "test"* | *ssmtp* keerthuvije@gmail.com
e. check /var/log/cron.log for any script errors.

19. Checked if httpd server is reachable from public by sending a request to the created Load Balancer's AWS DNS.

Results

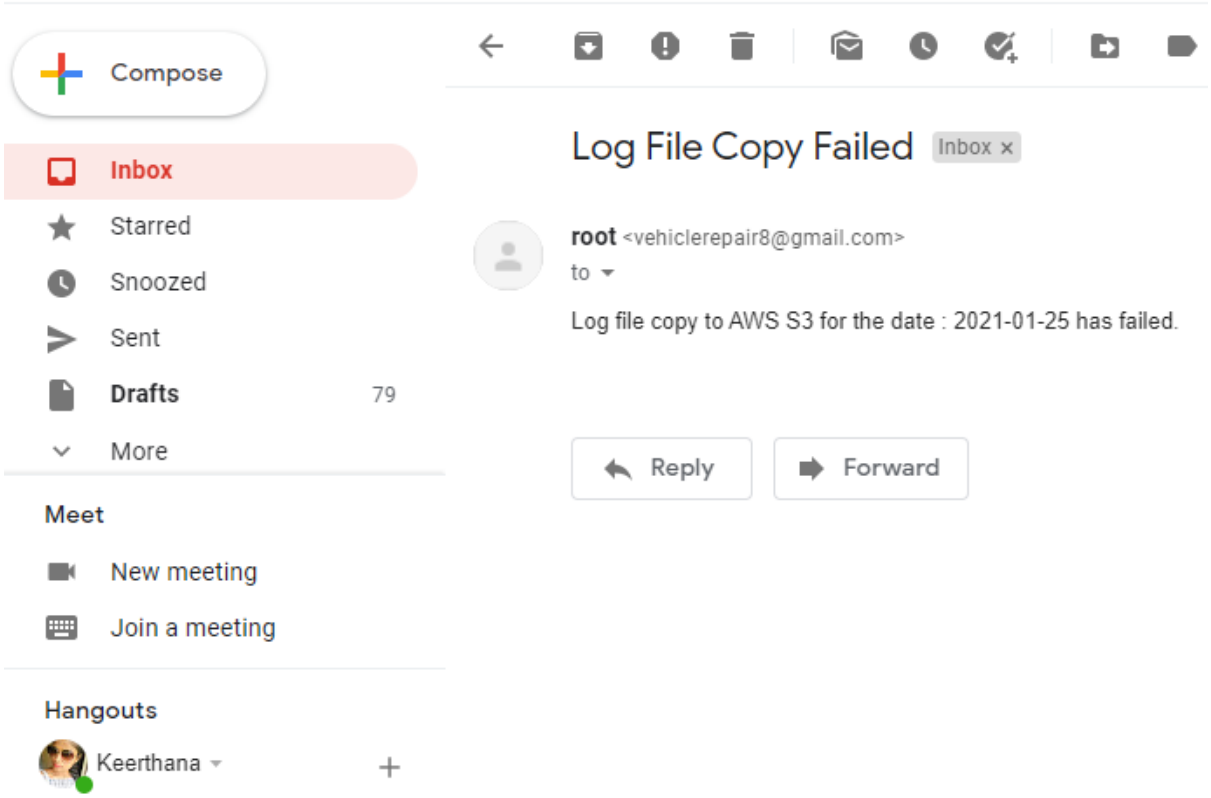
1. Log files saved in S3 daily



2. Web Server reached via AWS Load Balancer DNS from Public Browser.

   http://webserver-lb-27343380.us-east-1.elb.amazonaws.com/

3. Email received when Log File Copy to S3 Failed. (Simulated test by removing access to S3 for that specific instance)



Monitoring

1. Created a DynamoDB table in the us-east-1 region with the name “webserver-monitoring‘.
2. Used DynamoDB since its the cheapest database solution available in AWS.
3. Prepared the following monitoring script using BASH to check if the httpd server is running and in the case if its not running, run it.

```
#!/bin/bash

# Function to send emails
function email () {
cat <<EOF | ssmtp keerthuvije@gmail.com
Subject: Monitoring Script Failed

$1
EOF
}

# DynamoDB Insert Array String start
```

```
strJson="["
# Get timestamp to be used as primary key
timestamp=$(date -u +%Y-%m-%d:%H:%M:%S)

# Get list of all instances in webserver-asg Autoscaling group
for line in $(aws autoscaling describe-auto-scaling-groups --
output json --region us-east-1 --auto-scaling-group-names
webserver-asg | jq -c ".AutoScalingGroups | .[0] | .Instances |
.[]")
do
   # Default result is success
   result="success"
   # Get Instance ID from ASG describe output
   inId=$(echo $line | jq -r ".InstanceId")
   # Get Ip of Instance using gathered Instance Id
   inIp=$(aws ec2 describe-instances --region us-east-1 --
instance-ids $inId --output text --query
'Reservations[*].Instances[*].[PrivateIpAddress]')
   # Debug print statement
   echo "Ip of instance $inId is $inIp"
   # SSH into specific instance and check if httpd is running
else start
  # StrictHostKeyChecking is disabled since inside internal
network
   ssh -i "/etc/monitoring/keerthan-key.pem" -
oStrictHostKeyChecking=no -oIdentitiesOnly=yes ec2-user@$inIp
"systemctl is-active httpd || sudo systemctl start httpd"
   # Check if SSH Failed, if so send mail
   if [ $? -ne 0 ]
   then
       # Set result to SSH failed
       result="Failed to SSH"
       # Send Email
       email "Failed for Instance $inId. Due to SSH Fail"
   fi
   # Try to get web content
   resp=$(curl -o /dev/null -s -w "%{http_code}\n" $inIp)
   # Check if http code is 200 (Success)
   if [ $resp -ne 200 ]
   then
       # Update result
       result="Failed to get web content"
       # Send Email
```

```
        email "Failed for Instance $inId. Due to bad status code
when trying to get content : $resp"
    fi
    echo $result
    # Format StrJson according to DynamoDB required format

strJson="$strJson{\"M\":{\"code\":{\"S\":\"$resp\"},\"ip\":{\"S\"
:\"$inIp\"},\"result\":{\"S\":\"$result\"}}},"
done

# Remove last comma from json
strJson=${strJson::-1}
# Add ending ] to str
strJson="$strJson]"
echo $strJson

# Add Output to dynamoDB
aws dynamodb put-item --region us-east-1 --table-name webserver-
monitoring --item
"{\"timestamp\":{\"S\":\"$timestamp\"},\"response\":{\"L\":$strJs
on}}" --region us-east-1
if [ $? -ne 0 ]
then
    # Send Email
    email "Failed to add data into DynamoDB at $timestamp"
fi
```

4. Copied the created script to the config s3 bucket under the container “monitoring”.
5. Created the User Data Script to pull in config from the configuration bucket and to initialize the cronjob which run the above script hourly.

```
#!/bin/bash

configBucket="capital-market-config-bucket"


## COMMON
# update yum packages
sudo yum update -y
sudo yum install jq -y

## EMAIL
# Enable Rhel Yum repository
sudo amazon-linux-extras install epel -y
```

```
# Install Email client
sudo yum install ssmtp -y
# Delete current ssmtp conf if available
if [ -f /etc/ssmtp/ssmtp.conf ]
then
   sudo rm /etc/ssmtp/ssmtp.conf
   sudo rm /etc/ssmtp/revaliases
fi
# Copy email config from config s3 bucket
sudo aws s3 cp s3://$configBucket/monitoring/ssmtp.conf
/etc/ssmtp/ssmtp.conf
sudo aws s3 cp s3://$configBucket/monitoring/revaliases
/etc/ssmtp/revaliases


## MONITORING
# Create Script dir
sudo mkdir -p /etc/monitoring/
# Copy script from config s3 bucket
sudo aws s3 cp s3://$configBucket/monitoring/monitoring.sh
/etc/monitoring/script.sh
# Copy ssh key from config bucket
sudo aws s3 cp s3://$configBucket/monitoring/keerthan-key.pem
/etc/monitoring/keerthan-key.pem
sudo chmod 400 /etc/monitoring/keerthan-key.pem

# Allow execute permission to script.sh
sudo chmod 755 /etc/monitoring/script.sh
# Configure crontab to run fail at 23:30
cat <<EOF > /cron.txt
5 0-23 * * * /etc/monitoring/script.sh >> /var/log/cron.log
EOF
sudo crontab /cron.txt
```

6. Tested the scripts using a test EC2 instance will Admin privileges. Deleted the instance afterwards.
7. Copied the email config which was created for the webserver to the config bucket under the “monitoring” container.
8. Created a IAM policy which would allow the instance the following actions
    a. Pull all config under the monitoring container in the config s3 bucket.
    b. Push status updates to “webserver-monitoring” DynamoDB.
    c. Read instance data from Auto Scaling Group to get private instance IP’s dynamically.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "VisualEditor0",
           "Effect": "Allow",
           "Action": [
               "autoscaling:DescribeAutoScalingInstances",
               "ec2:DescribeInstances",
               "ec2:DescribeScheduledInstances",
               "autoscaling:DescribeAutoScalingGroups",
               "ec2:DescribeInstanceAttribute",
               "ec2:DescribeInstanceStatus"
           ],
           "Resource": "*"
       },
       {
           "Sid": "VisualEditor1",
           "Effect": "Allow",
           "Action": [
               "s3:GetObject",
               "dynamodb:PutItem"
           ],
           "Resource": [
               "arn:aws:dynamodb:us-east-
1:861710225209:table/webserver-monitoring",
               "arn:aws:s3:::capital-market-config-bucket/monitoring/*"
           ]
       }
   ]
}
```

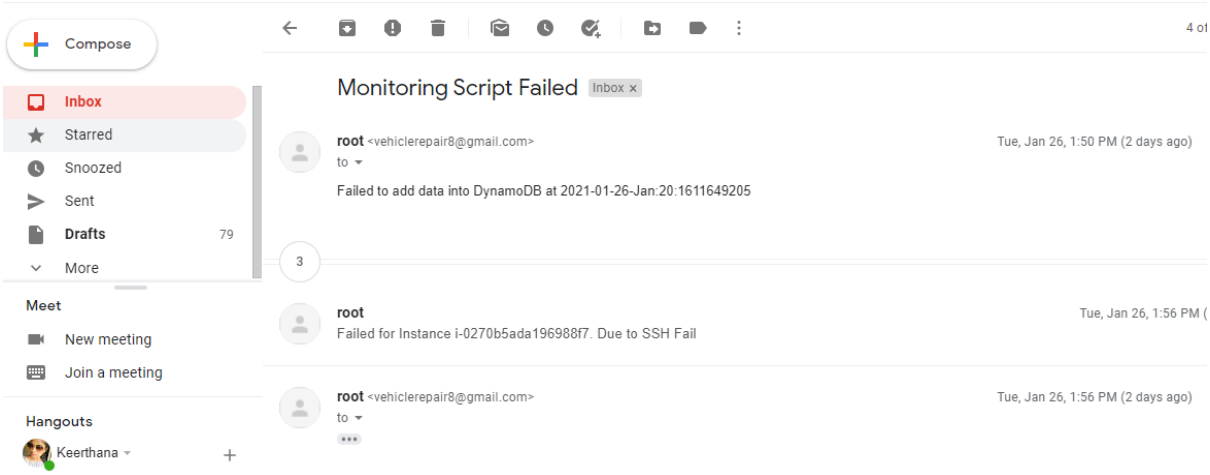
9. Created an IAM Role from the above policy with name as “monitoring”.
10. Created a Launch Configuration with the following properties
    a. Used t2.micro instances to reduces costs
    b. Used AmazonLinux2 AMI
    c. Used keerthan-key pair
    d. Used spot instances
    e. Used monitoring IAM instance Role
    f. Added the above script as userdata script
    g. Used default VPC and subnets to make the setup simpler
    h. Added the following rules under the security group
        i. Allowed SSH (22) only from my IP

ii. Allowed SMTP (25)
iii. Allowed port 587 which is port of gmail's SMTP server

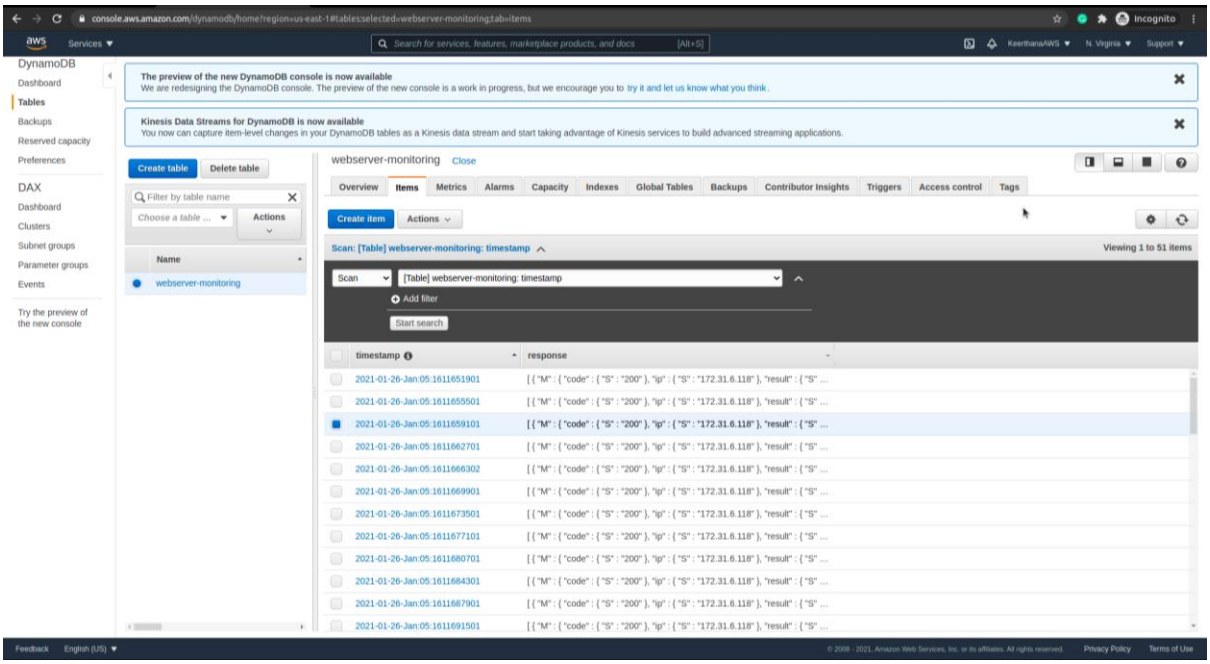
11. Created an ASG using the above created launch configuration. Used an ASG to make sure always one instance is running. Configured EC2 health check.
12. Allowed SSH(22) into the web server instances from the monitoring server using the security group id of the monitoring server.
13. Validated the setup
    a. Checked if the cronjob has been created. *sudo crontab -l*
    b. Ran monitoring script manually and checked if status is updated in dynamoDB *sudo /etc/monitoring/script.sh*
    c. Checked if email configuration is correct by sending a dummy test email. *echo "test"* | *ssmtp* keerthuvije@gmail.com
    d. Checked /var/log/cron.log for any script errors

Results

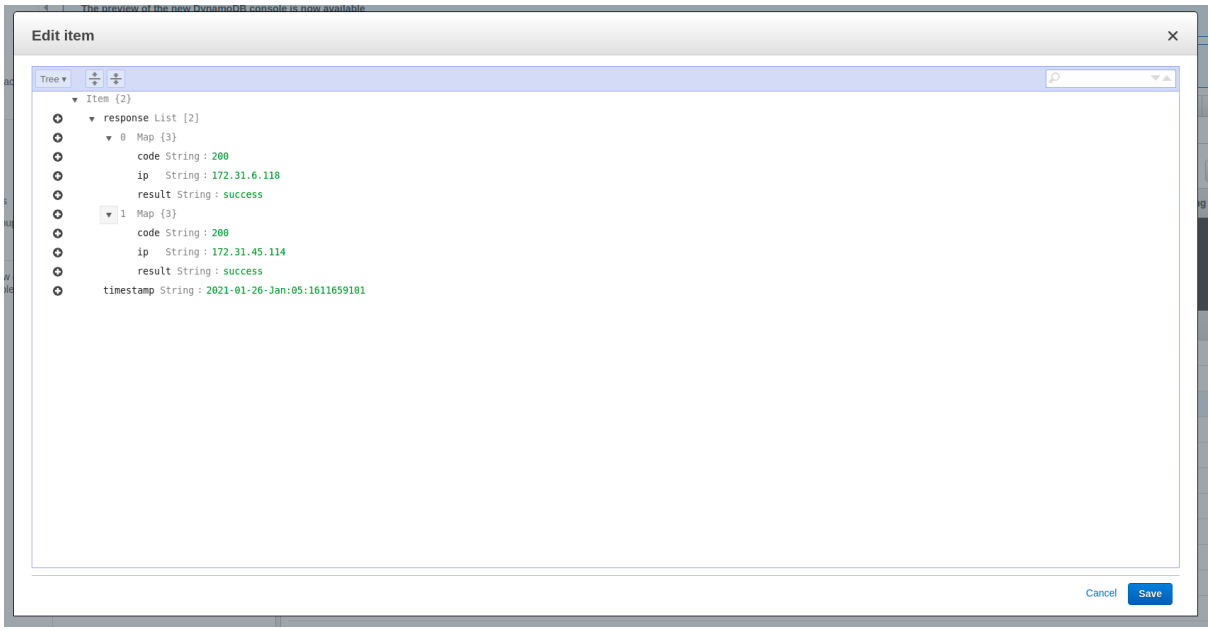
1. Received error email when status update to dynamoDB failed (Simulated Test by removing permissions)



2. List of status items in DynamoDB with timestamp as the primary key



3. Specific status Item in DynamoDB



Instructions to set up in any environment and the steps to execute the script

NOTE :

- If any name changes or region changes are required. Make sure to change the references in the script and policies accordingly.
- " " wil be used to highlight names.
- Port 587 is used by Gmail SMTP Server. Make sure to whitelist your server correctly

Pre-requists :

1. Configured AWS Account
2. us-east-1 region will be used (Or Script needs to be edited)

To Do

1. Update ssmtp.conf and revalaliases under both monitoring and webserver directory with your email username and password
2. Create a config s3 bucket named "capital-market-config-bucket"

Common

1. Configured AWS Account.
2. us-east-1 region selected as the region.
3. Create a config s3 bucket named "capital-market-config-bucket" - in order to store all the configs.

Webserver

1. Create a logging s3 bucket named as "capital-market-logging-bucket" - to store collected logs

2. Copy files under webserver directory to s3 bucket "s3://capital-market-config-bucket/webserver/"

3. Create Key Pair with the name "keerthan-key" and download the key - To SSH the instance.

4. Create IAM Role using iam-policy.json file inside webserver directory. - To allow the web server instance to get config data from config s3 bucket and also will allow the instance to write log files to logging bucket.

5. Create Launch configuration. Make sure to follow below steps;
   1. Use the above created keypair and IAM instance Role.
   2. Add webserver/userdata.sh files contents under Userdate in Launch configuration - userdata.sh is the file which has all the configured details when starting a webserver.
   3. Use Amazon Linux 2 AMI -OS

4. Make sure to use a public subnet (Default VPC, Default Subnet works) - to check we need to use SSH. to allow we used a public
5. Allow Port 80 and 587 from anywhere in Security Group - to allow requests from public and 587 is to allow if a mail request comes from gmail to allow.
6. (Optional) Allow Port 22 from your IP for debugging - SSH

6. Create ASG from above created launch config - lauch configuration is kind of a blue print, ASG is will help to create an instance when a server died. ASG will create a new server based on the launch config
1. Create Internet Facing load balancer with ASG as the target for port 80 - if a ASG has an instance, we requested that IP instance. then if ASG creates a new instance IP will change but the user who is requesting to the server doesn't know, so Load Balancer will devide load and forward to the correct IP.

7. Validation
1. SSH into the instance using the public ip and the downloaded key
2. Check if scripts are present under /etc/logging
3. Run script and check if the compressed log files are copied to S3
4. use crontab -l command and check if the crontab is present to run the script daily
5. Use the Load Balancers DNS to make sure you can reach httpd server in instance
6. Script logs will be added to /var/log/cron.log

Monitoring

1. Copy files under monitoring directory to s3 bucket "s3://capital-market-config-bucket/monitoring/"
2. Copy downloaded webserver key (Step: Webserver:3) to s3 bucket as "keerthan-key.pem" "s3://capital-market-config-bucket/monitoring/keerthan-key.pem"
3. Create a DynamoDB table named as "webserver-monitoring" with primary key as "timestamp"
4. Create a new keypair
5. Create IAM Role using iam-policy.json file inside monitoring directory.
6. Create Launch configuration. Make sure to follow below steps;
1. Use the above created keypair and IAM instance Role
2. Add monitoring/userdata.sh files contents under Userdate in Launch configuration
3. Use Amazon Linux 2 AMI
4. Make sure to use a public subnet (Default VPC, Default Subnet works)
5. Allow Port 587 from anywhere in Security Group
6. (Optional) Allow Port 22 from your IP for debugging
7. Create ASG using created Launch Config
8. Validation
1. SSH into the instance using the public ip and the downloaded key
2. Check if scripts are present under /etc/monitoring
3. Run script and check if result has been added to DynamoDB
4. use crontab -l command and check if the crontab is present to run the script Hourly
5. Script logs will be added to /var/log/cron.log

Conclusion

Using AWS infrastructure as a solution mainly has benefits for cost effectiveness, and also when it's come to the security AWS provides end-to end security and privacy to customers. Therefore, we can guarantee our stored information credential. AWS works with a pay-as-you-go concept which means we can pay as per usage scheme. Altogether this is highly beneficial architecture to be used.

Source code repository link :- https://github.com/KeerthuVije/capital-market

References

1. AWS documentation
2. A cloud guru
3. Stack Overflow
4. Youtube videos
5. Random Google websites
6. KT from a cloud expert