**EC2 instance—window and hosting application**

**1. Instance name**

**2. Select window os (AMI).**

**3. Choose instance type.**

**4. Create keypair**

**5. Allow HTTP and HTTPS by checking the box**

**6. Clink on lunch instance.**

**Select our server->**

**Connect->RDP client->download remote desktop file->click on Get password->using keypair generated password.**

**User name:Administrator.**

**EC2-window application host:-**

**Window->type server manager->add roles and feature->next(3)->Web server(IIS)->next…->install**

**C://inetpub/wwwroot/<past html or application code>**

**EC2 instance—Linux**

**1. Instance name**

**2. Select Linux os (AMI).**

**3. Choose instance type.**

**4. Create keypair**

**5. Allow HTTP and HTTPS by checking the box**

**6. Clink on lunch instance.**

**Select our server. Then copy the public ip address.**

**Connect through MobaXterm (paste ip address, ec2-user, upload key)**

**EC2-linux application host:-**

**sudo yum install httpd -y**

**sudo systemctl status httpd**

**sudo systemctl start httpd**

**sudo wget https://www.free-css.com/assets/files/free-css-templates/download/page292/honey.zip**

**unzip <zipfile>**

**cd <extractedfolderfromzipfile>**

**sudo cp -r \* /var/www/html**

**EC2-linux application host:- need to install httpd server**

**1 ls**

**2 pwd**

**3 ls**

**4 touch file1**

**5 ls**

**6 mkdir myfolder**

**7 ls**

**8 ls -l**

**9 pwd**

**10 cd myfolder**

**11 pwd**

**12 touch file1**

**13 ls**

**14 cd ..**

**15 ls**

**16 pwd**

**17 vi file2**

**18 cat file2**

**19 sudo -i**

**20 sudo vi file2**

**21 cat file2**

**22 ls**

**23 sudo vi file3**

**24 ls**

**25 cat file3**

**26 cat file1**

**27 sudo vi file1**

**28 cat file1**

**29 cd /var/www/html**

**30 sudo yum install httpd**

**31 cd /var/www/html**

**32 pwd**

**33 cd**

**34 pwd**

**35 ls**

**36 sudo wget https://www.free-css.com/assets/files/free-css-templates/download/page292/honey.zip**

**37 ls**

**38 unzip honey.zip**

**39 ls**

**40 cd honey-html**

**41 ls**

**42 cp -r \* /var/www/html**

**43 sudo cp -r \* /var/www/html**

**44 cd**

**45 cd /var/www/html**

**46 ls**

**47 sudo systemctl status httpd**

**48 sudo systemctl start httpd**

**49 sudo systemctl status httpd**

**50 history**

**EBS-Elastic Block Storage for Window Machine**

**Steps**

1. **Create an EC2 instance(Linux server)**
2. **Create an EBS volume**
3. **Click on EBS volume then go to action and attach volume to EC2 instance**
4. **Open the RDP window machine:**
5. **Window+R**
6. **diskmgmt.msc**
7. **Select disk 1 and right click then online**
8. **Right click and select initialize disk**
9. **New->next->derive name->next->finish.**

**EBS-Elastic Block Storage for Linux Machine**

df -hT ->to check our volume

lsblk->list the block

sudo -i

sudo mkfs -t ext4 /dev/xvdf->to create file system

mkdir extrastorage

mount /dev/xvdf /home/ec2-user/<extrastorage>

**Steps**

1. **Create an EC2 instance(Linux server)**
2. **Create an EBS volume**
3. **Click on EBS volume then go to action and attach volume to EC2 instance**
4. **Login server and run the following commands:**

**1. ls**

**2. df -hT**

**3. lsblk**

**4. sudo mkfs -t ext4 /dev/xvdf**

**5. mkdir extras**

**6. cd extras**

**7. pwd**

**8. cd ..**

**9. sudo mount /dev/xvdf /home/ec2-user/extras**

**10. df -hT**

**11. cd extras (extras is our created volume)**

**12. sudo mkdir impfolder**

**EFS-Elastic File System**



EC2



**EC2**



EC2

Steps

1. Create two EC2-instances(Linux server).
2. Create EFS**.( security group of EFS-All traffic)**
3. Login two EC2-instance with different color.

Server 1

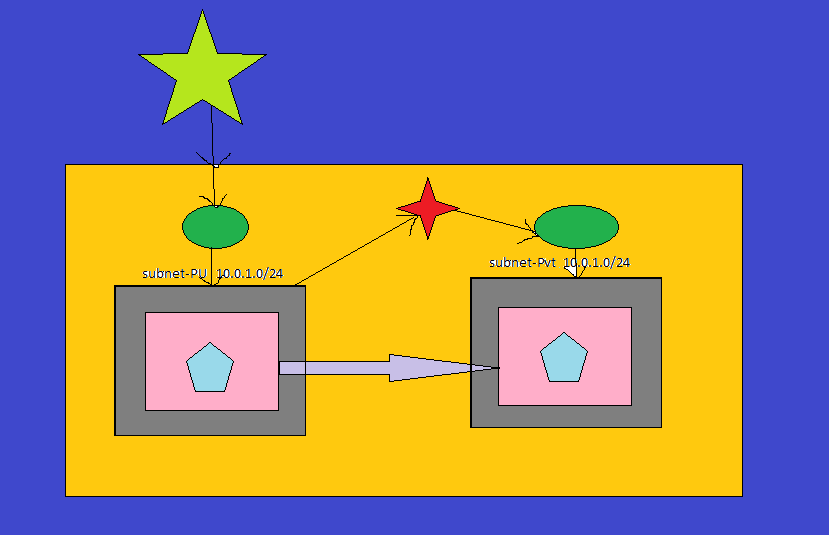
1. mkdir ram
2. Click on EFS and attach then copy NFS client method command.
3. Paste the NFS client command and change as /ram.
4. df -hT
5. cd ram
6. mkdir dailylog

Server 2

1. mkdir efsstorage2
2. Click on EFS and attach then copy NFS client method command.
3. Paste the NFS client command and change as /raj.
4. df -hT
5. cd raj
6. ls

dailylog



**** 

**Private IP address**

**10.0.0.0----class A**

**172.16.0.0 to 172.31.0.0-----class B**

**192.168.0.0-----Class C**

**Step 1: create VPC**

**my-vpc; & IPv4 CIDR: 192.168.0.0/16**

**select vpc ->action->edit vpc setting->dns setting->Enable DNS hostnames**

**Step 2: create two subnet**

**subnet-pu; 192.168.1.0/24; 1a-AZ is public server-Action->edit subnet->Enable Auto assign public IP address & subnet-pvt; 192.168.2.0/24; 1b-AZ-private server**

**Step 3: create InternetGate way**

**my-igw & attach it to my-vpc**

**Step 4: create Route Table for public server**

**MY-RT-PU; select my-vpc**

**Step 5: select MY-RT-PU🡪Route->Edit routes->add route:0.0.0.0/0; Target :my-igw**

**subnet association:edit subnet association->select subnet-pu->save association**

**Step 6: create public EC2 instances**

**Linux; Network:my-vpc; subnet:subnet-pu; Auto assign public IP:Enable; security group:SSH and anywhere.**

**Connect with putty.**

**ping google.com---to check connectivity**

**ctrl+Z**

**Step 7: create private EC2 instances**

**Linux; Network:my-vpc; subnet:subnet-pvt; Auto assign public IP:Disable; security group:SSH and anywhere.**

**To connect it need public server and putty.**

**sudo vi newkey.pem**

**paste pem that opened with notepad.**

**chmod 400 newkey.pem**

**ssh -i newkey.pem ec2-user@<private ip address>**

**ping google.com---to check connectivity**

**ctrl+Z**

**Step 8: create NAT gateway**

**MY-NAT; subnet:sebnet-pu; Elastic IP address: Allocate Elastic IP.**

**Step 9: create Route Table for public**

**MY-RT-PVT; select my-vpc**

**Step 10: select MY-RT-PVT->Route->Edit routes->add route:0.0.0.0/0; Target:nat gateway**

**subnet association:edit subnet association->select subnet-pvt->save association**

**ping google.com**

**How to delete vpc setup**

**1.NAT Gateway**

**2.EC2 instance**

**3.Elastic IP**

**4.Internet gateway**

**5.VPC**

**6.subnet**

**7.route table**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**S3**

**1. Versioning**

**2. Object lock**

**3. Life cycle policy**

**4. Replication policy**

**5. Create Static web hosting — untick Block public access. Object owner->ACLs enable.**

**action->make public using ACls**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CLB---Classical Load Balancer**

**Steps**

1. **Create two EC2-instances with different Availability Zone**
2. **Connect both instances, and run the following commands**

sudo yum install httpd -y

sudo systemctl restart httpd

sudo systemctl enable httpd

cd /var/www/html

sudo vi index.html

enter any code (code for color now)

1. **create CLB (Classic Load Balancer)**
2. **copy the DNS name and browse it.**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<title> Login Page </title>

<style>

body {

font-family: blue;

background-color: pink;

}

button {

background-color: #4CAF50;

width: 100%;

color: orange;

padding: 15px;

margin: 10px 0px;

border: none;

cursor: pointer;

}

form {

border: 3px solid #f1f1f1;

}

input[type=text], input[type=password] {

width: 100%;

margin: 8px 0;

padding: 12px 20px;

display: inline-block;

border: 2px solid green;

box-sizing: border-box;

}

button:hover {

opacity: 0.7;

}

.cancelbtn {

width: auto;

padding: 10px 18px;

margin: 10px 5px;

}

.container {

padding: 25px;

background-color: red;

}

</style>

</head>

<body>

<center> <h1> Student Login Form </h1> </center>

<form>

<div class="container">

<label>Username : </label>

<input type="text" placeholder="Enter Username" name="username" required>

<label>Password : </label>

<input type="password" placeholder="Enter Password" name="password" required>

<button type="submit">Login</button>

<input type="checkbox" checked="checked"> Remember me

<button type="button" class="cancelbtn"> Cancel</button>

Forgot <a href="#"> password? </a>

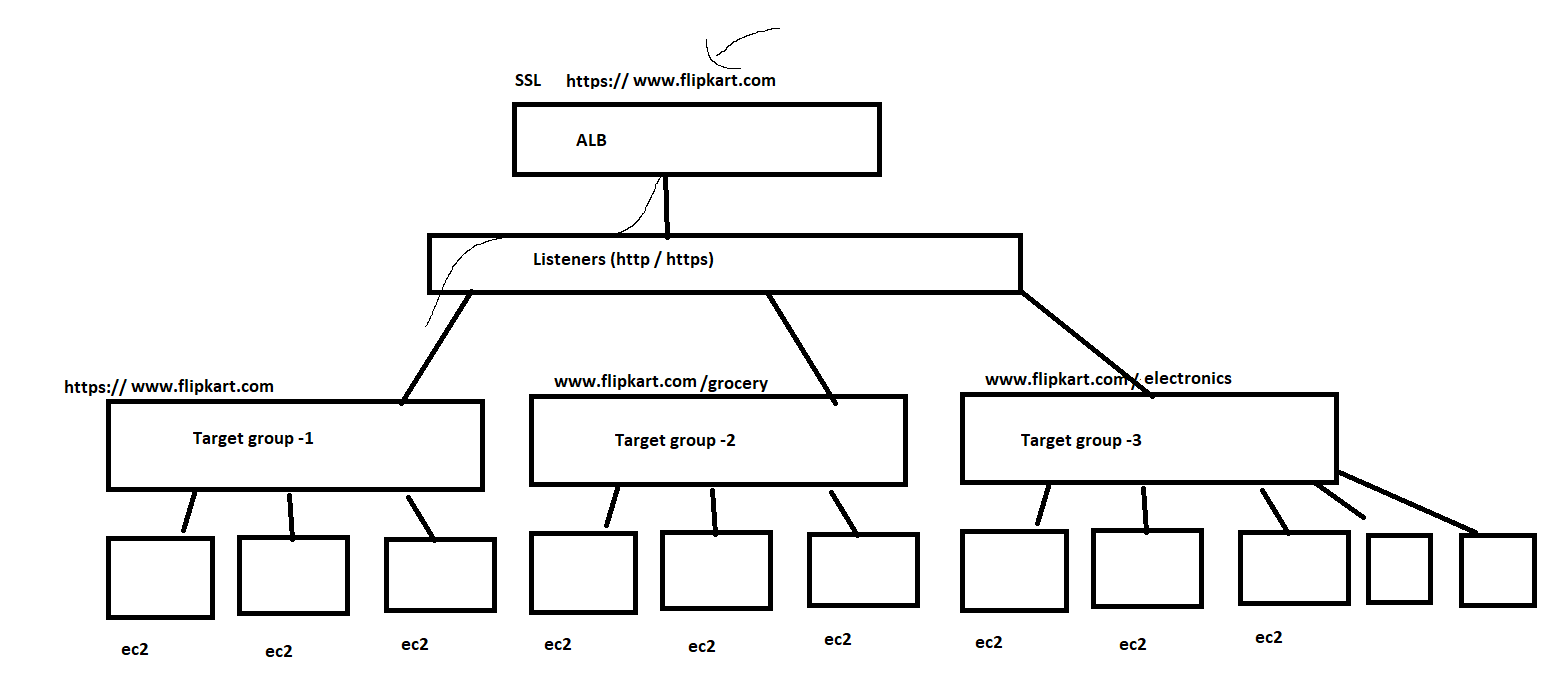
</div>

</form>

</body>

</html>

**ALB---Application LB**

****

**Step1: Create 6 web instance**

* **2 = 2a(1) and 2b(2) (main web)**
* **2 = 2a(3) and 2b(4) (audio web)**
* **2 = 2a(5) and 2b(6) (video web)**

**Host the application:**

**main web(2a-1&2b-2)- #!/bin/bash**

**sudo yum -y install httpd**

**sudo systemctl start httpd**

**sudo systemctl enable httpd**

**cd /var/www/html**

**sudo vi index.html**

**Hyper link (html code)**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Document</title>**

**</head>**

**<body>**

**<h1 style="color:cyan">THIS IS HOME PAGE</h1>**

**<a href="#" target="\_blank">Audio Page</a> <br><br>**

**<a href="#" target="\_blank">Video Page</a>**

**</body>**

**</html>**

**<a href="http://alb-1756842787.ap-south-1.elb.amazonaws.com/audio/index.html">Audio Page</a>**

**<br><br>**

[**http://alb-836133737.ap-south-1.elb.amazonaws.com/audio/index.html**](http://alb-836133737.ap-south-1.elb.amazonaws.com/audio/index.html)

**<a href="http://alb-1756842787.ap-south-1.elb.amazonaws.com/video/index.html">VideoPage</a>**

[**http://alb-836133737.ap-south-1.elb.amazonaws.com/video/index.html**](http://alb-836133737.ap-south-1.elb.amazonaws.com/video/index.html)

**audio web (2a-3&2b-4)- #!/bin/bash**

**sudo yum -y install httpd**

**sudo systemctl start httpd**

**sudo systemctl enable httpd**

**sudo mkdir /var/www/html/audio**

**cd /var/www/html/audio**

**sudo vi index.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Document</title>**

**</head>**

**<body>**

**<h1 style="color:blue; background-color:green;">This is my AUDIO Server page</h1>**

**<a href="#">Tamil hit songs</a> <br><br>**

**<a href="#">English hit songs</a>**

**</body>**

**</html>**

**video web (2a-5&2b-6)- #!/bin/bash**

**sudo yum -y install httpd**

**sudo systemctl start httpd**

**sudo systemctl enable httpd**

**sudo mkdir /var/www/html/video**

**cd /var/www/html/video**

**sudo vi index.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Document</title>**

**</head>**

**<body>**

**<h1 style="color:blue; background-color:red;">This is my VIDEO Server page</h1>**

**<a href="#">Tamil hit Movies</a> <br><br>**

**<a href="#">English hit Movies</a>**

**</body>**

**</html>**

**Step 2: Create ALB**

**target group Name: TG1**

**give target group path : /index.html**

**Add web -1 and web -2**

**Step 3: Create target group : TG2**

**give target group path : /audio/index.html**

**Add web -3 and web -4**

**Step 4: Create target group : TG3**

**give target group path : /video/index.html**

**Add web -5 and web -6**

**Step 5: Go to Listeners - add/edit rules**

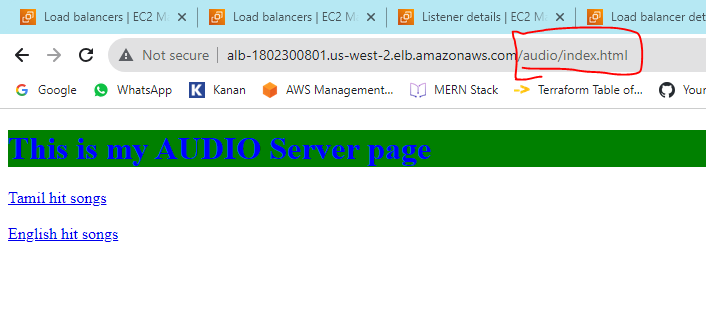
**add target 2**

**path : /audio/index.html**

**and target3**

**path : /video/index.html**

**Copy the DNS name generated and hit.**

****

**Auto Scaling:**

1. **AMI**
2. **SNS - Simple Notification service**
3. **Add this AMI to Launch template**
4. **Create auto scaling group (threshold value)and choose target tracking policies without loadbalancer**
5. **create LB and attach the new instances**
6. **then attach the LB to the auto scaling group**
7. **test in One instance with Stress tool software**

**sudo yum install stress -y**

**sudo stress --cpu 8 --vm-bytes $(awk '/MemAvailable/{printf "%d\n", $2 \* 0.9;}' < /proc/meminfo)k --vm-keep -m 1**

**LAMBDA: EC2 STOP START**

1. **Create two instances**
2. **Create lambda: Function🡪Name:stopstartinstances🡪Author from scratch🡪select Lambda🡪python 3.8🡪Role: lambdaEC2shedule: select: type: AWS services and Lambda then** [**AWSLambdaBasicExecutionRole**](https://us-east-1.console.aws.amazon.com/iam/home#/policies/arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole) **and EC2fullaccess🡪Attach this role and create function.**
3. **Code Test**

[**https://gist.github.com/nktstudios/5150810740be1b8a6f3a31408dc42d42**](https://gist.github.com/nktstudios/5150810740be1b8a6f3a31408dc42d42)

|  |  |
| --- | --- |
|  | **import json**  **import boto3**  **region = 'ap-south-1'**  **ec2 = boto3.client('ec2', region\_name=region)**  **def lambda\_handler(event, context):**  **instances = event["instances"].split(',')**  **action = event["action"]**    **if action == 'Start':**  **print("STARTing your instances: " + str(instances))**  **ec2.start\_instances(InstanceIds=instances)**  **response = "Successfully started instances: " + str(instances)**  **elif action == 'Stop':**  **print("STOPping your instances: " + str(instances))**  **ec2.stop\_instances(InstanceIds=instances)**  **response = "Successfully stopped instances: " + str(instances)**    **return {**  **'statusCode': 200,**  **'body': json.dumps(response)**  **}** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Save then select configure test event in the just right side and name it start EC2 instance🡪paste our EC2 instace ID in below box with comma, action:’start’.**

1. **Create event.**

**{**

**"instances": "PUT YOUR INSTANCE ID's HERE with comma # (No space)",**

**"action": "Start"**

**}**

**Ex. {**

**"instances": "i-0411e61f54ae5736b,i-0430cd255ef01535d", # (No space)**

**"action": "Stop" "action": "Start"**

**}**

1. **Again select configure test event in the just right side and name it stop EC2 instance🡪paste our EC2 instace ID in below box with comma, action:’stop’.**
2. **Create event.**

**{**

**"instances": "PUT YOUR INSTANCE ID's HERE with comma # (No space)",**

**"action": "Start"**

**}**

**Set schedule**

1. **Select Amazon EventBridge**
2. **Create rules**
3. **Name:examplestopEC2instace**
4. **Define pattern: select schedule**
5. **Cron expression: 53 12 \* \* 1-5 \* or** A schedule that runs at a regular rate
6. **Target: Lambda function**

**Select function: startstopinstancefunction**

**Additional Setting**

1. **Configure version/target input: constant JSON text**
2. **Go to our lambda console: click on configure test event: select stopEc2: copy the text.**
3. **{**

**"instances": "PUT YOUR INSTANCE ID's HERE with comma # (No space)",**

**"action": "Stop"**

**}**

**Paste the text in constant JSON box**

**create**

**To start**

**Same as above and Go to our lambda console: click on configure test event: select startEc2: copy the text.**

**{**

**"instances": "PUT YOUR INSTANCE ID's HERE with comma # (No space)",**

**"action": "Start"**

**}**

**Paste the text in constant JSON box**

**create**

**LAMBDA FOR DYNAMODB**











1. **Amazon DynamoDB table.**

**Table Name:** [**jascloudtech\_students**](https://ap-south-1.console.aws.amazon.com/dynamodbv2/home?region=ap-south-1#table?name=jascloudtech_students)

**Partition key is ID**

1. **S3 bucket and upload a CSV file**

**Bucket Name:** [**s3studentdydb1**](https://s3.console.aws.amazon.com/s3/buckets/s3studentdydb1?region=ap-south-1)

**Upload a CSV file: students.csv (.csv stands for comma separator value)**

**1,Ram,Maths**

**2,Ravi,Science**

**3,Raja,English**

**4,Moorthi,Tamil**

**5,Kumar,Computer**

1. **Lambda Function creation**

**Function Name: s3importdydb**

**Python 3.8**

**Attach role🡪 Role Name: lambdarole🡪 DynamoDB full access, S3 full access and CloudWatch full access.**

1. **Test the CSV data import using test in Lambda**

**import boto3**

**s3\_client = boto3.client("s3")**

**dynamodb = boto3.resource("dynamodb")**

**table = dynamodb.Table("JasCloudtech\_students")**

**def lambda\_handler(event, context):**

**bucket\_name = event['Records'][0]['s3']['bucket']['name']**

**s3\_file\_name = event['Records'][0]['s3']['object']['key']**

**resp = s3\_client.get\_object(Bucket=bucket\_name,Key=s3\_file\_name)**

**data = resp['Body'].read().decode("utf-8")**

**Students = data.split("\n")**

**#print(students)**

**for stud in Students:**

**print(stud)**

**stud\_data = stud.split(",")**

**# add to dynamodb**

**try:**

**table.put\_item(**

**Item = {**

**"id" : stud\_data[0],**

**"name" : stud\_data[1],**

**"Subject" : stud\_data[2]**

**}**

**)**

**except Exception as e:**

**print("End of file")**

**then click on deploy.**

1. **Test:**

**Create new events**

**Event name: csv**

**Template: s3-put**

**Event JSON:**

**Region:ap-south-1**

**Change bucket name**

**Change key to our table name:students.csv**

1. **Test S3 event trigger to import data into DynamoDB**

**GO TO DYNAMOdb🡪TABLES🡪EXPLORE ITEMS.**

**EVENT NOTIFICATION:-**

1. **GO TO BUCKET**
2. **SELECT PROPERTIES THEN AND CLICK ON CREATE EVENT NOTIFICATION**
3. **EVENT NAME: CSVEVENT**

**SUFFIX:CSV**

**SELECT ALL OBJECT**

1. **DESTINATION: LAMBDA FUNCTION**

**CHOOSE MYLAMBDADB AND SAVE CHANGES.**

1. **GO TO BUCKET AND UPLOAD ANOTHER CSV FILE.**
2. **TEST OUR LAMBDA AND CHECK DYNAMODB TABLE.**

**NOTE:REGION SHOULD BE SAME**

**DynamoDB full access, S3 full access and CloudWatch full access.**

**DATABASE---RDS(UBUNTU SERVER)**

**Default user name: root**

**step1: create ubuntu server with mysecuritygroup**

**step2: create rds mysql database with create new security group**

**step3: edit rds security group with mysql and mysecuritygroup**

**step4: connect the server and run the following commands**

**sudo apt update**

**sudo apt upgrade**

**sudo apt install mysql-server**

**sudo apt install mysql-client**

**sudo mysql -h database.cilxfswqo8no.ap-south-1.rds.amazonaws.com -u root -p**

**show databases;**

**CREATE DATABASE employeesDB;**

**USE employeesDB;**

**SHOW TABLES;**

**create table employees(ID int, Name varchar(20), Age int, City varchar(20));**

**insert into employees values(1, "Raja", 30, "Chennai");**

**insert into employees values(2, "Ravi", 35, "Chennai");**

**insert into employees values(3, "Rajan", 40, "Selam");**

**insert into employees values(4, "Raman", 32, "Chennai");**

**insert into employees values(5, "Rangan", 25, "Gingee");**

**select \* from persons;**

**DELETE FROM employees WHERE ID=5;**

**DELETE FROM employees WHERE Name=”Raja”;**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Route53**

**1. Get domain Name from AWS Route53 or Godday.**

**2. Create a Host Zone in Route53.**

**3. Copy NS --- 4 lines.**

**4. Edit Domain name with the NS of AWS.**

**5. Create a record in the Host Zone and paste the ip address of the server.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CloudFront:**

**1. Create Bucket with just enabling bucket versioning.**

**2. Create static web hosting**

**3. Upload index.html file**

**4. Cloudfront -> create distribution ->choose s3 bucket ->Legacy access identities -> Create new**

**OAI -> Yes, update the bucket policy -> Redirect HTTP to HTTPS -> Do not enable security**

**protections -> Default root object *- optional -> index.html ->* create distribution.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CloudFormation:**

**1. create stack**

**2. create .yaml file**

**1 EC2Instance.yml**

**Resources:**

**MyInstance:**

**Type: AWS::EC2::Instance**

**Properties:**

**AvailabilityZone: us-east-1a**

**ImageId: ami-0005e0cfe09cc9050**

**InstanceType: t2.micro**

**2 EC2InstanceIP.yml**

**Resources:**

**MyInstance:**

**Type: AWS::EC2::Instance**

**Properties:**

**AvailabilityZone: us-east-1a**

**ImageId: ami-0005e0cfe09cc9050**

**InstanceType: t2.micro**

**MyEIP:**

**Type: AWS::EC2::EIP**

**Properties:**

**InstanceId: !Ref MyInstance**

**3 EC2InstanceMappings.yml**

**Mappings:**

**InstanceMap:**

**us-east-1:**

**dev: t2.micro**

**prod: m1.small**

**us-west-1:**

**dev: m1.small**

**prod: m1.large**

**AMIMap:**

**us-east-1:**

**dev: ami-0005e0cfe09cc9050**

**prod: ami-0005e0cfe09cc9050**

**us-east-2:**

**dev: ami-0cd3c7f72edd5b06d**

**prod: ami-0cd3c7f72edd5b06d**

**Resources:**

**MyInstance:**

**Type: AWS::EC2::Instance**

**Properties:**

**ImageId: !FindInMap [AMIMap, !Ref "AWS::Region", dev]**

**InstanceType: !FindInMap [InstanceMap,!Ref "AWS::Region", dev]**

**4 EC2InstanceParameter.yml**

**Parameters:**

**InstanceTypeParameter:**

**Type: String**

**Default: t2.micro**

**AllowedValues:**

**- t2.micro**

**- m1.small**

**- m1.large**

**Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.**

**Resources:**

**MyInstance:**

**Type: AWS::EC2::Instance**

**Properties:**

**AvailabilityZone: us-east-1a**

**ImageId: ami-0005e0cfe09cc9050**

**InstanceType:**

**Ref: InstanceTypeParameter**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Elastic Beanstack:**

**1. Create Application**

**2. role -> EC2 + Elasticbeanstack-choose EC2 and Elasticbeanstack give elastic enhancementhealth and customerRolePolicy**

**3. PHP platform.**

**4. create zip file**

**5. upload file**

**6. check basic and Managed updates→uncheck activated.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**ElasticCache:**

**1. create server**

**enable 11211 port in the security group inbound rule**

**2. create elasticache -1.memchached**

**a. Design your own cache**

**b. Standard create**

**3. install telnet in our server**

**4. telnet <endpoint of the memcached 11211>**

**set name 0 0 8**

**ramasamy**

**get name**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Redshift:-**

**1. create iam role for redshift-customizable - s3fullacess and redshiftfullaccess**

**2. create bucket without public block, versioning and select owner-> ACLs enable, ACL->read, write then upload**

**the users\_pipe and venue\_pipe (data files).**

**https://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-create-sample-db.html**

**3. connect database - default db name = dev.**

**4. create table**

**create table users(**

**userid integer not null distkey sortkey,**

**username char(8),**

**firstname varchar(30),**

**lastname varchar(30),**

**city varchar(30),**

**state char(2),**

**email varchar(100),**

**phone char(14),**

**likesports boolean,**

**liketheatre boolean,**

**likeconcerts boolean,**

**likejazz boolean,**

**likeclassical boolean,**

**likeopera boolean,**

**likerock boolean,**

**likevegas boolean,**

**likebroadway boolean,**

**likemusicals boolean);**

**create table venue(**

**venueid smallint not null distkey sortkey,**

**venuename varchar(100),**

**venuecity varchar(30),**

**venuestate char(2),**

**venueseats integer);**

**copy users from 's3://redshiftbucketnewjan29/allusers\_pipe.txt'**

**credentials 'aws\_iam\_role=arn:aws:iam::007387495206:role/redshiftrulenew'**

**delimiter '|' region 'us-east-1';**

**copy venue from 's3://redshiftbucketnewjan29/venue\_pipe.txt'**

**credentials 'aws\_iam\_role=arn:aws:iam::007387495206:role/redshiftrulenew'**

**delimiter '|' region 'us-east-1';**

**https://java-runtime-environment.en.softonic.com/download**

**https://doc.microsoft.com/en-us/sql/connect/jdcb/download-microsoft-jdbc-driver-for-sql-server**