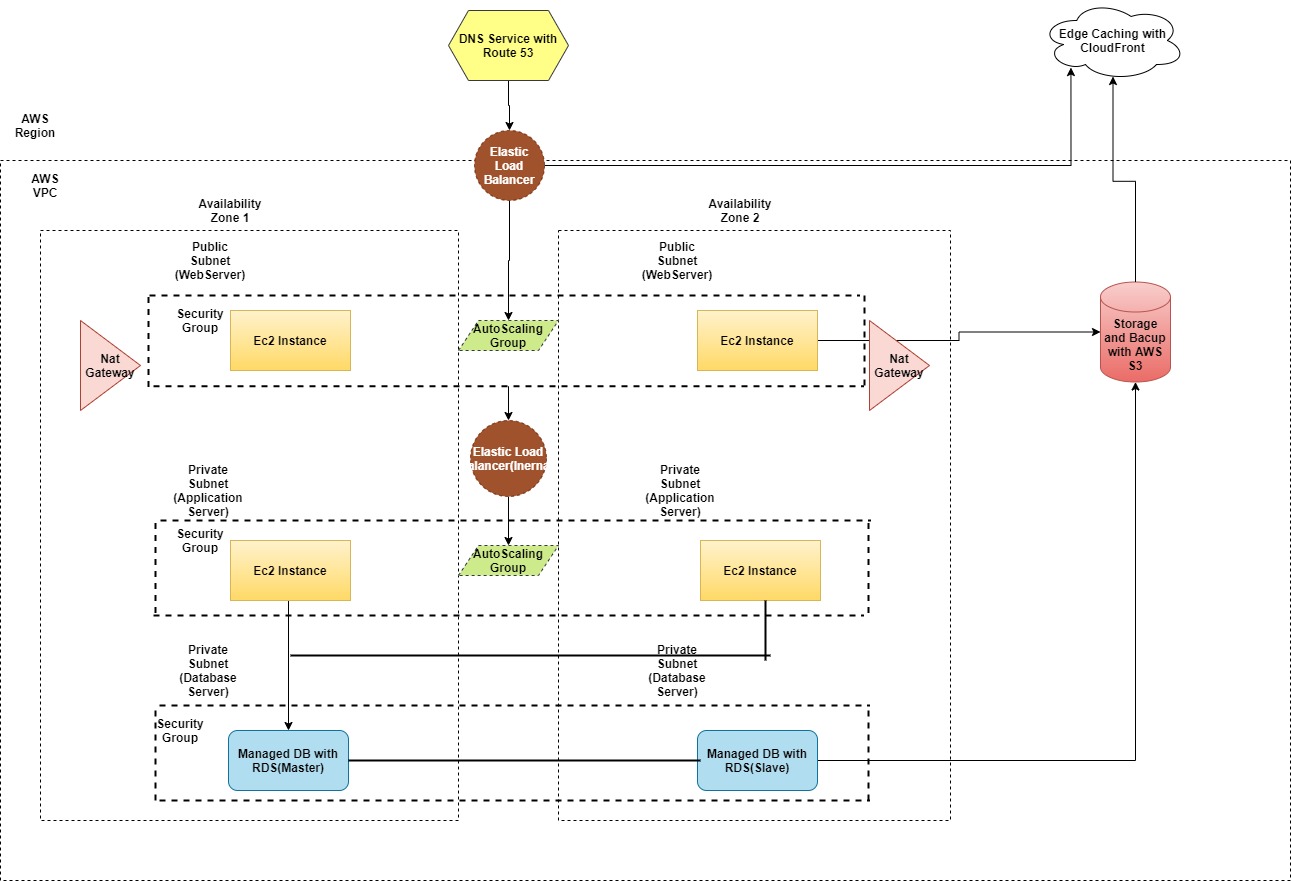
%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3Bdashed%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22628%22%20y%3D%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3Bdashed%3D1%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22628%22%20y%3D%2290%22%20width%3D%22392%22%20height%3D%22590%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E2290%22%20width%3D%22392%22%20height%3D%22590%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

**Challenge 1**

**AWS 3 tier Architecture**

****

AWS 3 tier architecture consists of 3 layer,

1) Public Layer

2) Application Layer

3) Database Layer.

**Public Subnets:-**

10.0.0.0/24 -> local

0.0.0.0/0 -> internet-gateway-id

**Private Subnets:-**

10.0.0.0/24 -> local

0.0.0.0/0 -> NAT-gateway-id

Above diagram shows AWS 3 tier architecture. Below are the steps to create 3 tier architecture in AWS management console.

Step 1:- Create VPC with public and private subnets for multiple Availability Zones as shown in the above design.

* Web servers are placed in a public subnet.
* Application and Database servers are placed in a private subnet.
* NAT Gateways are used to provide internet access to the resources in the private subnet i.e. for installing patches etc.
* You can manage access and security restrictions using security groups and network access control lists.
* AWS VPC gives you fine control to manage inbound and outbound traffic rules.

Step 2:- Create Load Balancers for Web and Application servers.

* Web and application servers are spread across multiple availability zones, the system needs a load balancer to distribute the incoming traffic.
* In the above design, there are two load balancers, the first one receives internet traffic and route it to the web servers.
* Second load balancer is used as an internal one that cannot be accessed by the external traffic and only route the requests from the web servers to the application servers.
* The internal load balancers serve the purpose of an added layer of security between the external traffic and the applications servers.

Step 3:- Create EC2 instances within Auto Scaling groups.

* The Auto Scaling groups enable the application to replace instances based on their health checks and also scale automatically in case it is overloaded by requests.
* It maintains the minimum amount of instances and scales whenever required. The automatic increase and decrease in the number of instances can be set by using scaling policies.

Step 4:- Setting up the Database tier.

* The next step is to setup the persistence layer.
* It can only be accessed by the second tier in which the application servers reside.
* I have used AWS Relational Database Service (RDS) instances. It also provides easy to set up, operate and scale a relational database in the cloud.
* RDS is highly available and secure. It also offers use-case specific instances e.g. performance optimized, high I/O or memory intensive workloads.

Step 5:- Setup DNS service with AWS Route 53.

* In order to receive the internet traffic and route it to the web servers, we use AWS Route 53 as a DNS service.
* It is simple, secure, scalable and highly available service that routes end user traffic to the internet applications based on multiple criteria i.e. latency, geolocation etc.
* You can also use the Cloud Front which is a Content Delivery network service to increase the performance of your web application by caching the most requested content in a nearby Edge Location.
* Above architecture achieves availability, security, performance, reliability and cost optimization.

**Challenge 2**

To view all categories of instance metadata from within a running instance, we use the URI.

**URI: - http://169.254.169.254/latest/meta-data/**

The IP address 169.254.169.254 is a link-local address and is valid only from the instance.

1. Bash scripting command, these below scripts gives the instance profile id and instance profile arn.

[ec2-user ~] **$ curl http://169.254.169.254/latest/meta-data/iam/info**

**Output:-**

**{**

**"Code" : "Success",**

**"LastUpdated”: "2018-08-15T23:19:38Z",**

**"InstanceProfileArn”: "arn:aws:iam::111122223333:instance-profile/MyInstanceProfile",**

**"InstanceProfileId" : " AIPA1234567890ABCDEFG "**

**}**

1. These below bash scripts gives the current region and account.

[ec2-user ~] **$ curl -s http://169.254.169.254/latest/dynamic/instance-identity/document | jq -r .accountId**

**{**

**"Code”: "Success",**

**"LastUpdated”: "2018-08-15T23:19:38Z",**

**"InstanceRegion”: "us-west-2",**

**"InstanceAccountId”: “32547985214095”**

**}**

1. These below bash scripts gives the top level metadata items.

[ec2-user ~] **$ curl http://169.254.169.254/latest/meta-data/**

CHALLENGE 3

object = {“a”:{“b”:{“c”:”d”}}}

key = a/b/c

object = {“x”:{“y”:{“z”:”a”}}}

key = x/y/z

value = a