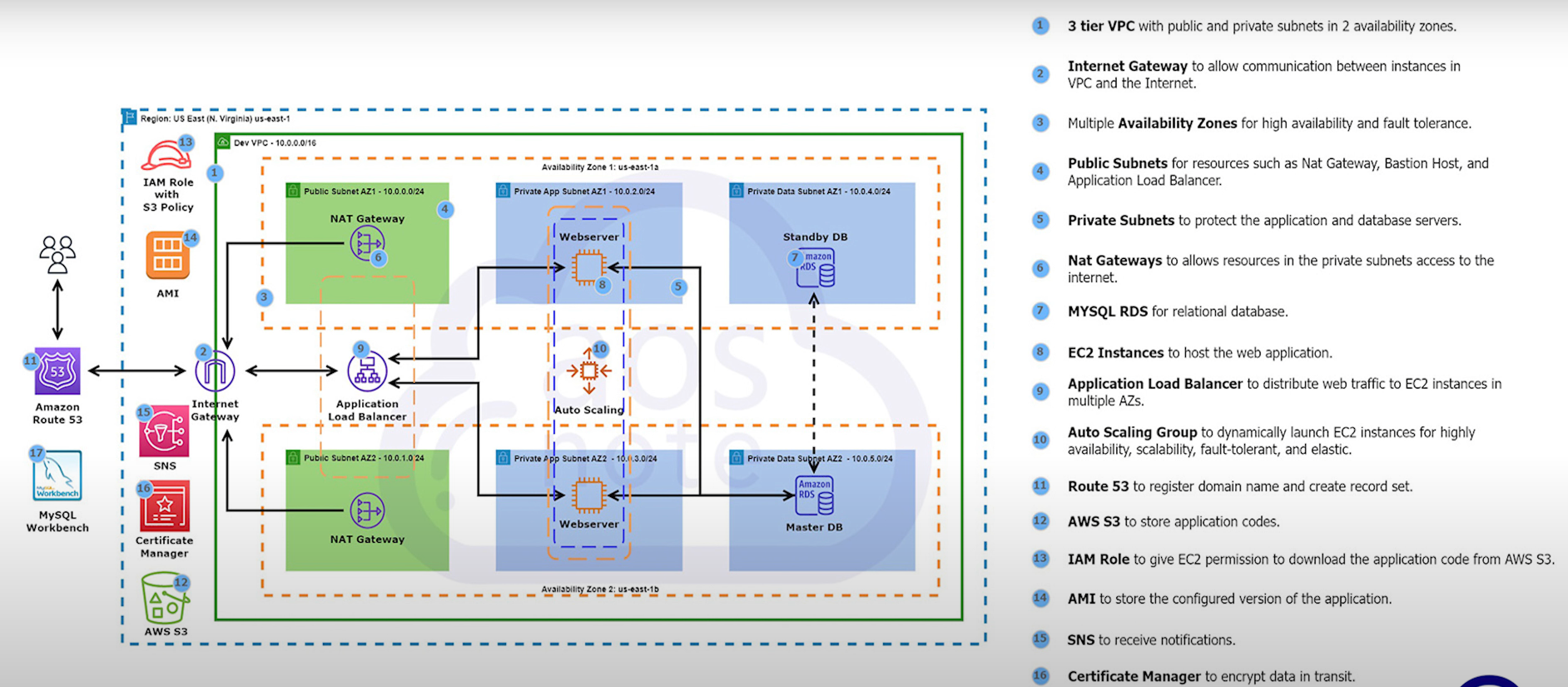
**Dynamic web application hosted on AWS**

Hi! This document is how we host a dynamic web application hosted on AWS following the architecture shown in below Image.



**3 Tier Architecture:**

Tier 1- Public Subnet for Nat Gateway, Bastion Host and Load Balancer

Tier2- Private Subnet for Webserver

Tier 3- Private Subnet for Database

STEPS to follow:

1. Create a VPC and name it Demo VPC. [Don’t forget to enable DNS hostnames through Actions > Edit VPC settings and enable DNS hostnames after launching VPC.]
2. Create IGW and attach it to the Demo VPC.
3. Create Public Subnet AZ1 in US east 1a [10.0.0.0 /24] and Public Subnet AZ2 in US east 1b [10.0.1.0 /24]. [Don’t forget to enable **Assign** **Public** **IPv4** through Actions > Edit subnet settings and enable Assign Public IPv4 after launching subnets.]
4. Create route table in Dev VPC also name it Public Route VPC and edit route and add route to internet via IGW into the table and then associate the Public Route table to 2 public subnets by editing subnet association.
5. Create Private App Subnet AZ1 in US east 1a [10.0.2.0 /24] and Private App Subnet AZ2 in US east 1b [10.0.3.0 /24] and 2 more database private subnet in both AZ in the same VPC. Name 2 Private Data Subnet AZ1 [10.0.4.0 /24] and Private Data Subnet AZ2 [10.0.5.0 /24]
6. Create 2 Nat Gateways in both AZ1 and AZ2 in public subnet 1 and 2 respectively and create private route table for AZ 1 and associate to subnet 3 and 5 that route to internet via NAT GATEWAY in AZ1. Similarly create private route table in AZ2 and associate with AZ2 private subnets and provide internet access via NAT Gateway in AZ2.
7. We would create 4 SGs:
   1. ALB Security Group
      * 1. Port: 80 and 443; Source: 0.0.0.0/0
   2. SSH Security Group
      * 1. Port: 22; Source: My IP address
   3. Web Server Security Group
      * 1. Port: 80 and 443; Source: ALB Security Group id
        2. Port: 22; Source: SSH Security Group id
   4. Database Security Group
      * 1. Port: 3306; Source: Webserver Security Group id
8. Create a RDS MySQL DBMS.

STEPS:

1. Create a **subnet group** in which the RDS will be deployed.
2. Go to Databases and click **create database**.
3. Choose **Standard Create** -> **MySQL** -> Under **Edition** select latest version.
4. Under **Templates** Choose between **Production**, **Dev/Test**, **Free Tier**.
5. Under **Availability and durability** Choose among **Multi-AZ DB Cluster**, **Multi-AZ DB instance**, **Single DB instance**.
6. Under **Settings** -> **Provide DB cluster identifier** (unique name)-> under **Credential** **settings** provide **Master username** and under **credential management** choose either **Managed in AWS Secrets Manager** or **Self-Managed.** You would be asked to create password on selecting self-managed.
7. Under **Instance configuration** Select out of 3 classes Standard, Memory optimized, Compute optimized.
8. Under **Storage type** among 3 options **General Purpose Provisioned, IOPS SSD(io1) or (io2).** Enter Allocated storage in Gib and provisioned IOPS. Select Storage Autoscaling(not available for Multi AZ DB cluster.)
9. Under connectivity -> Don’t connect to an ec2 compute resources. Select VPC and subnet group. Public access > No. Select Security Group.
10. After leaving te rest default create db.
11. Create S3 and upload file:
12. Create IAM role with s3 full access policy.
13. Create key pair and save it in same path and the main folder that shows in PowerShell. For e.g.: PS C:\Users\jinda> [In this system].
14. Launch EC2 instance in public subnet and name it, Setup-Server and we will use the instance for 2 tasks. First to import the SQL data for our application into the RDS database and secondly will use the setup server to install and configure the application. Make sure to use the key pair created and saved in step XI.

Select ALB SG, Webserver SG, SSH SG during creating EC2. Also assign S3 role created in step X. IAM role setting is in under additional setting.

1. Install MySQL Workbench on the system if it does not exist. Download it from [www.mysql.com](http://www.mysql.com), you dot require to sign in or login.

Once dialogue box comes up select custom and then click next. Expand Application > Expand MySQL Workbench and then select the latest version and move it using arrow towards right and then click next and then click execute > Next > Execute and then it will start installing it on the system.

1. Open the application on top menu click Database> Connect to Database > Connection Method drop down select Standard TCP Ip over SSH. Add Public IP DNS server address in the SSH Hostname area. For SSH username add username of setup server for e.g. for Amazon Linux we use ec2-user. Add private key after browsing to location where we saved the key i.e C:\Users\jinda in our case. Copy Endpoint of RDS under Connectivity & security on the console and paste it in MySQL Hostname and enter 3306 port and provide DB username and password. Click OK twice and successfully connect to RDS DB.
2. Import Data from local system to RDS using MySQL work bench. Now on the application tab next to Schemas click Data Import and select import from self- contained file. Browse to location we downloaded the db file> select and click open. Under Default target Schema select the name of the DB we want to work with. Click Start Import the data and work done successfully.
3. Host a Dynamic Website on EC2 server(LAMP)

Take SSH using powershell( key sould be in te same location powersell path opens, in our system C:\Users\jinda.

**Command**: ssh -i name\_of\_keypair.pem ec2-user@public\_ip\_of\_ec2.

example: ssh -i my-ec2key.pem [ec2-user@3.45.65.82](mailto:ec2-user@3.45.65.82)

Then follow below steps:

* 1. **update ec2 instance**

sudo yum update -y

* 1. **install Apache**

sudo yum install -y httpd httpd-tools mod\_ssl

[installs apache webserver + includes useful tools like htpasswd + adds ssl/tls support]

sudo systemctl enable httpd

sudo systemctl start httpd

* 1. **install php 7.4**

sudo amazon-linux-extras enable php7.4

sudo yum clean metadata

sudo yum install php php-common php-pear -y

sudo yum install php-{cgi,curl,mbstring,gd,mysqlnd,gettext,json,xml,fpm,intl,zip} -y

* 1. **install mysql5.7**

sudo rpm -Uvh https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

sudo yum install mysql-community-server -y

sudo systemctl enable mysqld

sudo systemctl start mysqld

* 1. **download the rentzone zip from s3 to the html directory on the ec2 instance**

sudo aws s3 sync s3://rentzone123bucket /var/www/html

* 1. **unzip the rentzone zip folder**

cd /var/www/html

sudo unzip rentzone.zip

* 1. **move all the files and folder from the rentzone directory to the html directory**

sudo mv rentzone/\* /var/www/html

* 1. **move all the hidden files from the rentzone directory to the html directory**

sudo mv rentzone/.well-known /var/www/html

sudo mv rentzone/.env /var/www/html

sudo mv rentzone/.htaccess /var/www/html

* 1. **delete the rentzone and rentzone.zip folder**

sudo rm -rf rentzone rentzone.zip

* 1. **enable mod\_rewrite on ec2 linux**

sudo sed -i '/<Directory "\/var\/www\/html">/,/<\/Directory>/ s/AllowOverride None/AllowOverride All/' /etc/httpd/conf/httpd.conf

* 1. **set permissions**

sudo chmod -R 777 /var/www/html

sudo chmod -R 777 storage/

* 1. **add database credentials**

sudo vi .env

edit below in the file:

DB\_CONNECTION=mysql

DB\_HOST=rentzone-db.cdaqms8qunqr.ap-south-1.rds.amazonaws.com

DB\_PORT=3306

DB\_DATABASE=new\_db\_rentzone

DB\_USERNAME=admin

DB\_PASSWORD=easypassword1

* 1. **restart server**

sudo service httpd restart

1. Create AMI for the setup-server that we would use to launch a server in private server and the new server would already have the application installed with every setup done in setup-server. Before we launch the server, we can terminate the previous setup server.
2. In next step we would create ec2 server in private subnet using AMI created in previous step.en Create an application Load Balancer

*First*, we would launch the ec2 in private subnet. Confiuration: **Name**- webserver az1, **Ami**: under **My AMI:** select AMI created in *XVII step*. Select Instance type, select **private** subnet you want to launch webserver to then select **webserver** **security** **group** created in step VII.

*Secondly*, create a target group with webserver instance. Configuration:

Target type: Instances, provide target group name, select the vpc, protocol version: Http1, Healt check select Advance health check and Add success codes for Https as well that is: 301,302 them create target group.

*Then*, last you just have to create a load balancer. In Ec2 service in left tab go to Load Balancer and open. Select application load balancer and click create. Basic configuration: give it name. Select Internet facing and select IPv4. Select your VPC and public subnet AZs.

Select ALB Security group from previous step. Under **Listener and routing** setting select your target group from drop down and on port 80 for http protocol.

You would also provide Https protocol on port 443 after creating SSL certificate. Click create after leaving rest settings on default.

***You should be able to access website on load balancer DNS if settings are done correctly.***

1. Register a New Domain in Route 53

Under Register Domain check if required domain name is available. Once find scroll down and click continue> provide contact info> Continue > enable if you want to renew > complete order. Might take max of 3 days or min of 15 minutes.

When registered you can see in Registered domains

1. Create a Record Set in Route 53

Route 53> Hosted Zones> Create Record > Record name: www, Record type: A-Route traffic, alias: enable then add route traffic: select ALB> provide region ALB is created in and then select ALB by name > Create record set.

***Now, you can access website through the domain name you registered in Route 53.***

1. Register an SSl certificate in AWS certificate manager.

Open Certificate Manager > Request a public certificate > Enter fully qualified domain name + add another name to this certificate: enter \*.domain\_name.com > Select DNS validation > click Request. Create record in route 53 once request gets completed(select both domain name created)> Create record.

1. Secure a website or web application with ACM SSL certificate

To secure our website we will add new listener on load balancer that listens to traffic on port 443.

Click Load Balancer> Open created load balancer > Scroll down to listener> Add Listener > ADD listener port to HTTPS and on 443> Default action: Forward: Select Target group> Scroll down to Seccure Listener setting> Default SSL/TLS certificate> From ACM: select Certificate created in XXI> Click Add.

Next edit Port:80 listener and redirect traffic to HTTPS: Port 443.

***You now have to SSH to webserver ec2 and make some changes in configuration file for website to work properly.***

1. SSH into an ec2 instance in the private subnet
2. we launch bastion host in public subnet. While launching make sure to use the same key as of webserver.

EC2>launch Bastion host in public subnet> add same key created for webserver ec2>Select VPC and provide security group SSH created already.

* + - 1. Next take ssh of Bastion host using PowerShell and set up SSH agent in it. We need to launch PowerShell as administrator to set up SSH agent.

Search PowerShell in search box and right click and run as administrator

# Check if ssh-agent is running

Get-Service ssh-agent

# Start the service

Start-Service ssh-agent

# This should return a status of Running

Get-Service ssh-agent

# Now load your key files into the ssh-agent

ssh-add C:\Users\Admin\my-ec2key.pem

# SSH into the instance (bastion host) in the public subnet. Remember to allow agent forwarding

After loading private key into ssh agent you need to close powershell and open again without admin rights to take ssh into private server.

* + - 1. Command to ssh: ssh-A ec2-user@public ip of bastion host

Then you type ssh private ip address of webserver host.

* + - 1. Update .env file after ssh : cd /var/www/html > sudo vi .env > press “I” to enter insert/edit mode in the file > Edit APP\_ENV= local and change local to production. Also need to change URL=<paste websites Https url(don’t forget to remove / at the last of the url)>. Next Press esc then type :wq! Save and exit
      2. Next we need to add https setting in AppServiceProvider.php for our application framework is Lavarel we need to do this step to allow https url to be accessible.
      3. We need to change directory in var/www/html > cd app > cd Providers > In here run command <# sudo vi AppServiceProvider.php > to open the app service provider file and do following changes:
      4. Need to do some changes in the file so enter “I” after opening the file > Take cursor down to

Public function boot

{ #add this if statement inside this function

if(env(‘APP\_ENV’=== ‘production’ ))

{

\Illuminate\Support\Facades\URL::forcescheme(‘https’);

}

}

* + - 1. Last after making these changes and saving the AppServiceProvider.php. You need to restart http service. Run Command: sudo service httpd restart

1. Create another AMI:

To reflect changes made I configuration files and webserver in previous step.

Go to ec2 > Select the webserver ec2 > click actions > Imae and template > Create image > name Rentzone AMI version 2 > Under Tags, select Tag image and snapshot together > Key: name, Value: Rentzone AMI version 2

1. Create auto scaling group.

After creating AMIs delete existing ec2 and ceate auto scaling policy using the AMI version 2 and you will have the configuration done on the ec2 launched using the version 2 AMIs and website will be up and running on Https url.

Steps to set up ASG:

Open ASG tab > Click Create launch template > Name Dev launch template > Give description: launch template for ASG > For AMI select the AMI created under my AMIs (select the latest version of AMI) > Select instance type > select the key pair created: myec2-key > Select webserver SG under existing security group > Create Launch template.

After creating LT > create Auto Scaling group > give it name> select LT created > Under Network: select Dev-VPC and private app subnet AZ1 and AZ2 > Click next > Under Load Balancing attach existing Dev-LB and provide/select LB target group > Under health check select ELB checkbox and provide Health check grace period in seconds > Monitoring : check box for group metrics collection with CloudWatch > click Next > Provide desired capacity: min capacity and max capacity > Under Notifications> click create topic > Send a notification name > Enter email id that needs to be notified> select event types u need notification for : launch, terminate, fail to launch and fail to terminate > Add Tags: The new launched ec2 will have this tag (check the tag new instances)> Click Next to review then Create new Auto Scaling Group.