**Designing a Cloud Solution – A Medical Company Startup**

Sahil Jaswal, Shubham Prasad, Zhaode Ouyang

2021 Spring ITC6480 SEC 1 AWS Cloud Architecting

Northeastern University

July 1, 2021

**Introduction**

A Medical Company is a startup software as a service (SaaS) company. They have built an online medical social networking and diagnosis assistance applicationfor users in APAC, the US, and Europe. The Medical company wants to launch the application with some requirements. This Project is the solution to deploy the Application on cloud with customers’ requirements by using AWS technologies in the right manner. And Translate customer requirements into a proposed technical solution.

**Executive Summary**

In this project, we are going to move the current architecture from datacenter to the cloud. Based on the requirement, we determined the current architecture is eligible to move to the cloud. The Cloud architecture not only providing a higher security and resiliency and achieve the requirement of supporting rapid growth, but delivering data with low latency, high transfer speed, all within a user-friendly environment.

**Overall Requirements and Assumptions**

***Overall Requirements***

The Medical Company needs to achieve the current function of the architecture, which is to allow online appointments, remote consultation, remote diagnosis, electronic prescription transfer, and payment services.

Besides, the architecture needs to achieve high availability for all tiers to reduce downtime, for users in APAC, the US, and Europe, control access to the application and limit public entry points, allow access management for different users and administrators, minimize IP address usage to reduce the attack surface, and host the development, test, and production environments.

Other additional requirements are needed are to allow customers to upload datasheets and images. The datasheet should automatically be extracted into a database, and the images should be converted into multiple formats. Besides, the architecture needs to handle doubling the number of servers to support its rapid growth.

***Overall Assumptions***

We assume the architecture will help the Medical Company achieve all the requirements. But more importantly, the architecture will help them save costs, providing a greater economy of scale, high availability, high security, and better disaster recovery and reliability.

**Solution**

In order to achieve the requirement mentioned above, we going to first determine what cloud services we going to use and make a cloud architecture diagram based on their current architecture and additional requirements. Second, we set the fine-grained user authentication, such as each group of users has different access. Third, we configured the network of the cloud architecture, including VPC, subnets, and IP address. Forth, we selected the type of instances and configured the autoscaling. Finally, we configuration the autoscaling group and include monitoring and auditing services.

## ***Identify AWS Services***

EC2 - A web service that provides secure, resizable compute capacity in the cloud

IAM - Enables you to manage access to AWS services and resources securely

RDS - An easy to set up, operate, and scale a relational database in the cloud

VPC - A service that lets you launch AWS resources in a logically isolated virtual network that you define

Route 53- A highly available and scalable cloud Domain Name System (DNS) web service

Auto Scaling- Monitors your applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost

Elastic Load Balancer - It automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, Lambda functions, and virtual appliances

AWS Lambda - A serverless compute service that lets you run code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes

Amazon CloudFront - A web service that speeds up distribution of your static and dynamic web content to users

Amazon S3 - A storage for the Internet. It is designed to make web-scale computing easier

Amazon SNS - A fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication

Amazon Cloud watch- A monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers

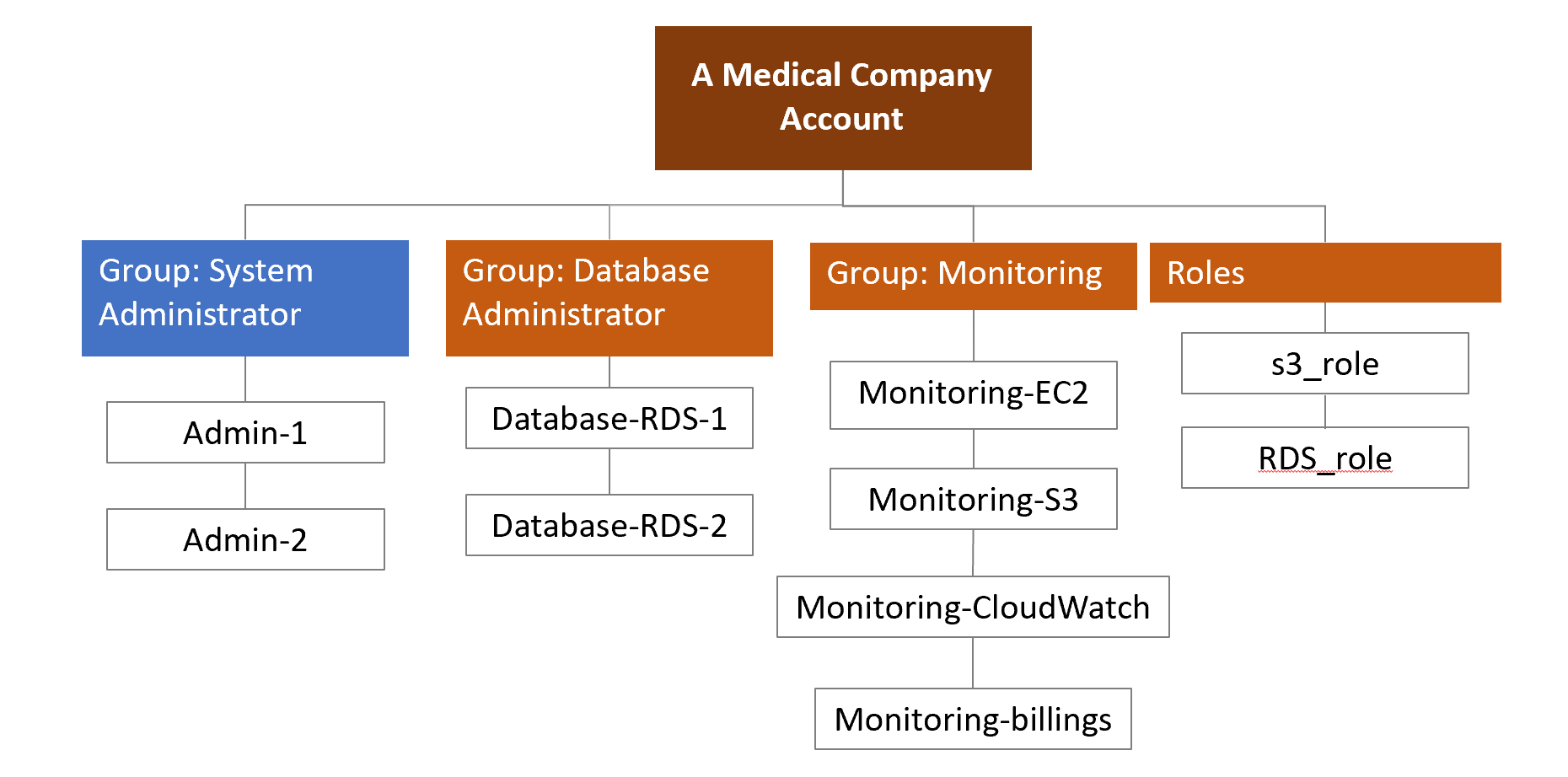
Nat Gateway- To let instances in a private subnet can connect to services outside your VPC, but external services cannot initiate a connection with those instances

**User Authentication**

In Table 1 and 2, follow AWS best practices for assigning permissions, there are three user groups with AWS Management Console access. Also, there are two roles for the application tier to access RDS and S3.

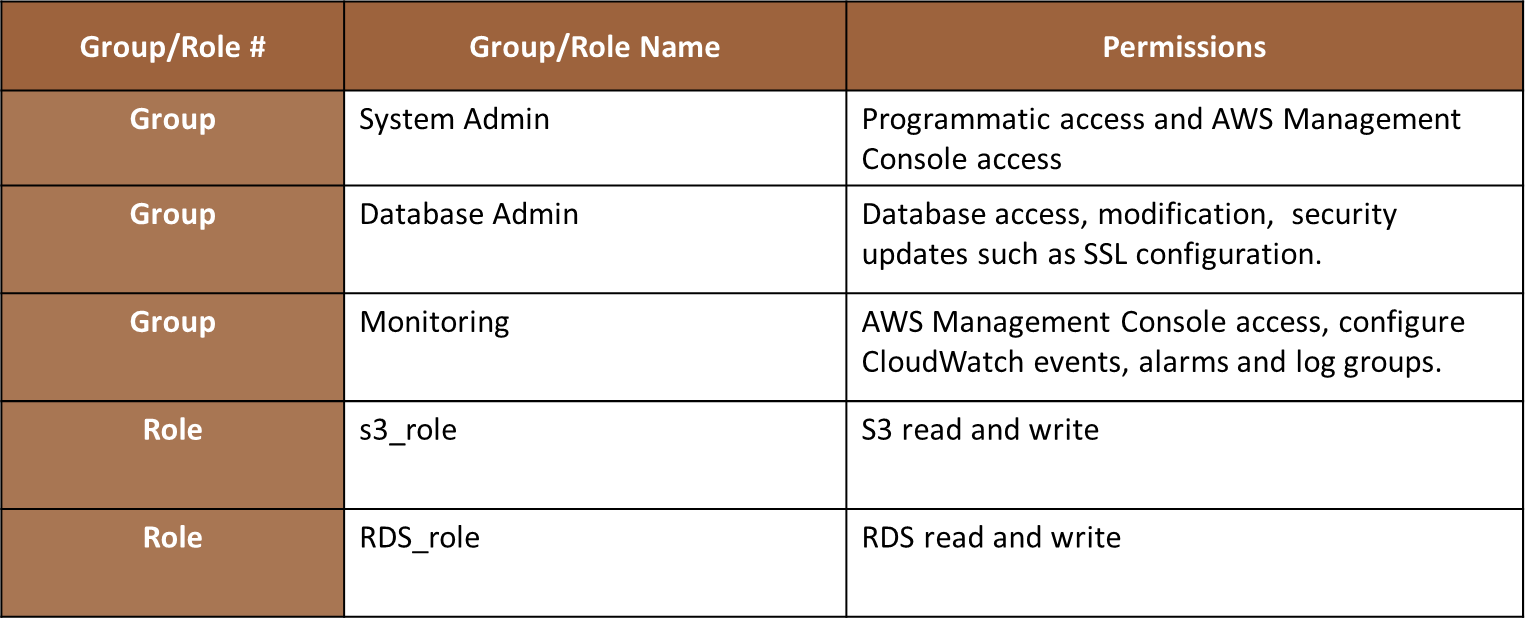
**Table 1**

*Groups and Role Chart*

****

**Table 2**

*Groups, Role name, Permissions*

****

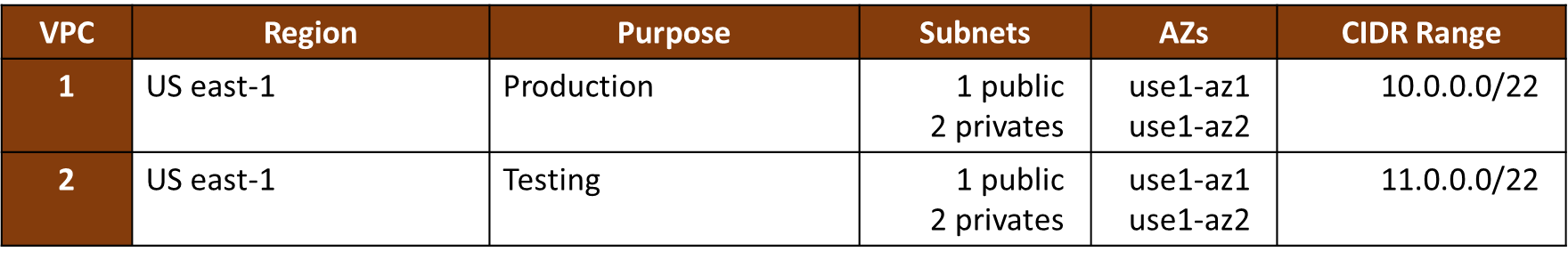
**Network and Security**

***VPC***

The whole architecture will have two VPCs. One for production, and one for testing. They are all in the US east-1 region.

**Table 3**

*VPCs Overview*



In Table 3, we can see each VPC have total of 3 subnets including 1 public subnet, and 2 private subnets. Each VPC will across two Availability Zones for high availably, and they are use1-az1 and use1-az2.

Continue in Table 3 in CIDR Range. For assigning IP address, we will use CIDR block of /22, which mean we have 1,024 IP address for each VPC (not include the 5 AWS reserves IP addresses). We decided to use CIDR block of /22 in order to minimize IP address usage to reduce the attack surface but still have enough for the future growth.

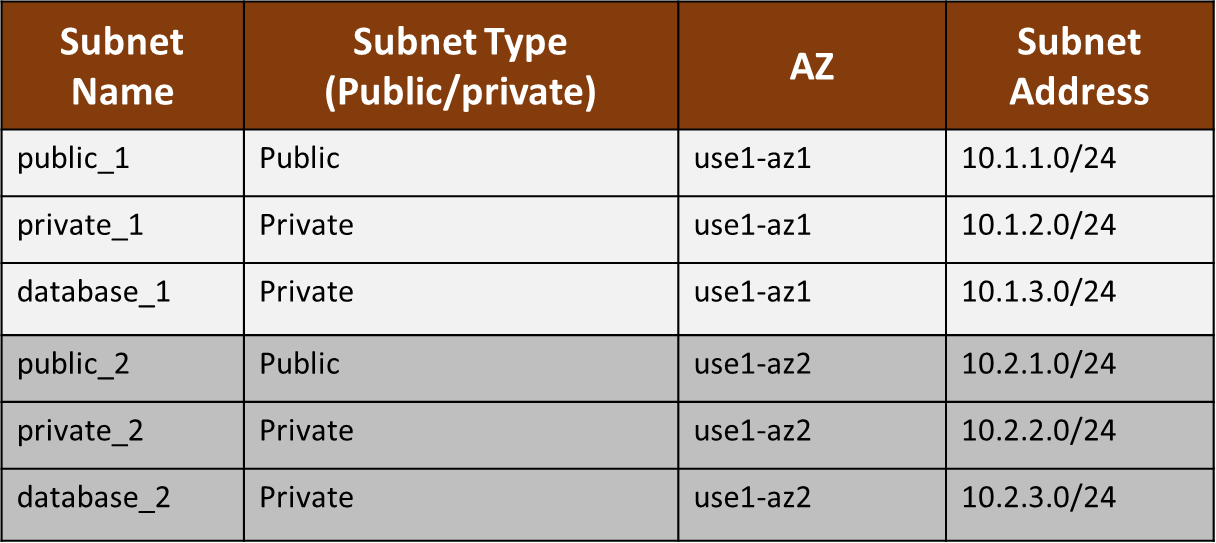
***Subnets***

Tables 4 is the detail subnet configuration of the production VPC. The public subnets are to host a NAT gateway and the web-tier. For the other two pair of private subnets, one is for hosting the application Tier, and the other is for hosting the database. We decided to use two sets of private subnets because we want the RDS to achieve higher security and fine-grained control. For IP address of subnets, each subnet will have a unique group of IP address to prevent overlap. They are all having a CIDR block of /24, which is 251 available IP address, and that is enough for the autoscaling.

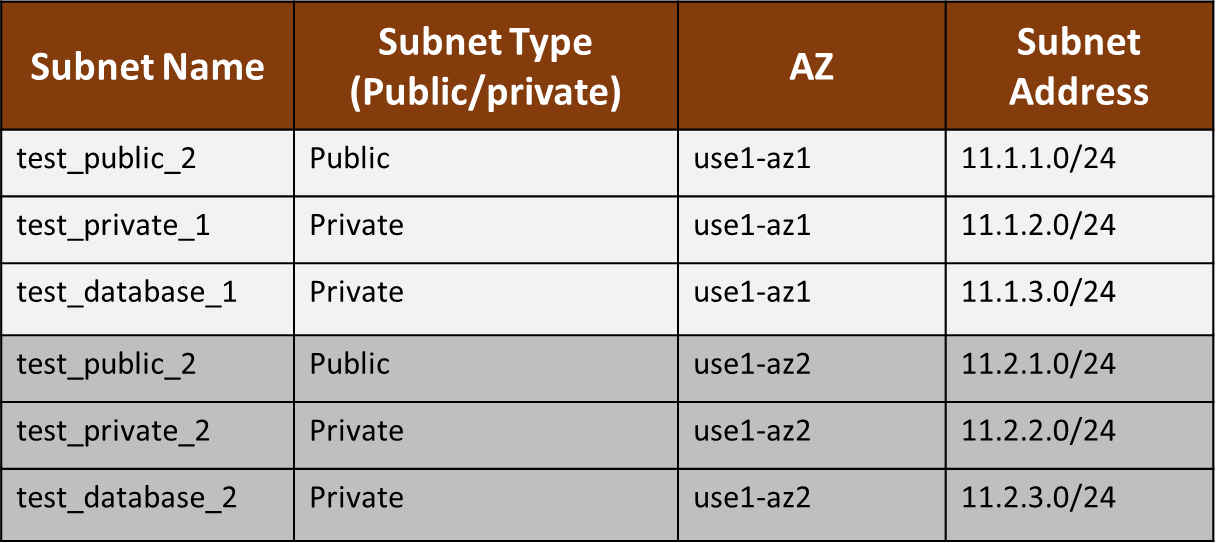
Tables 5 is the VPC for testing. The two VPC are similar, except for the subnet name and IP address.

**Table 4**

*Production VPC Subnets Configuration*



**Table 5**

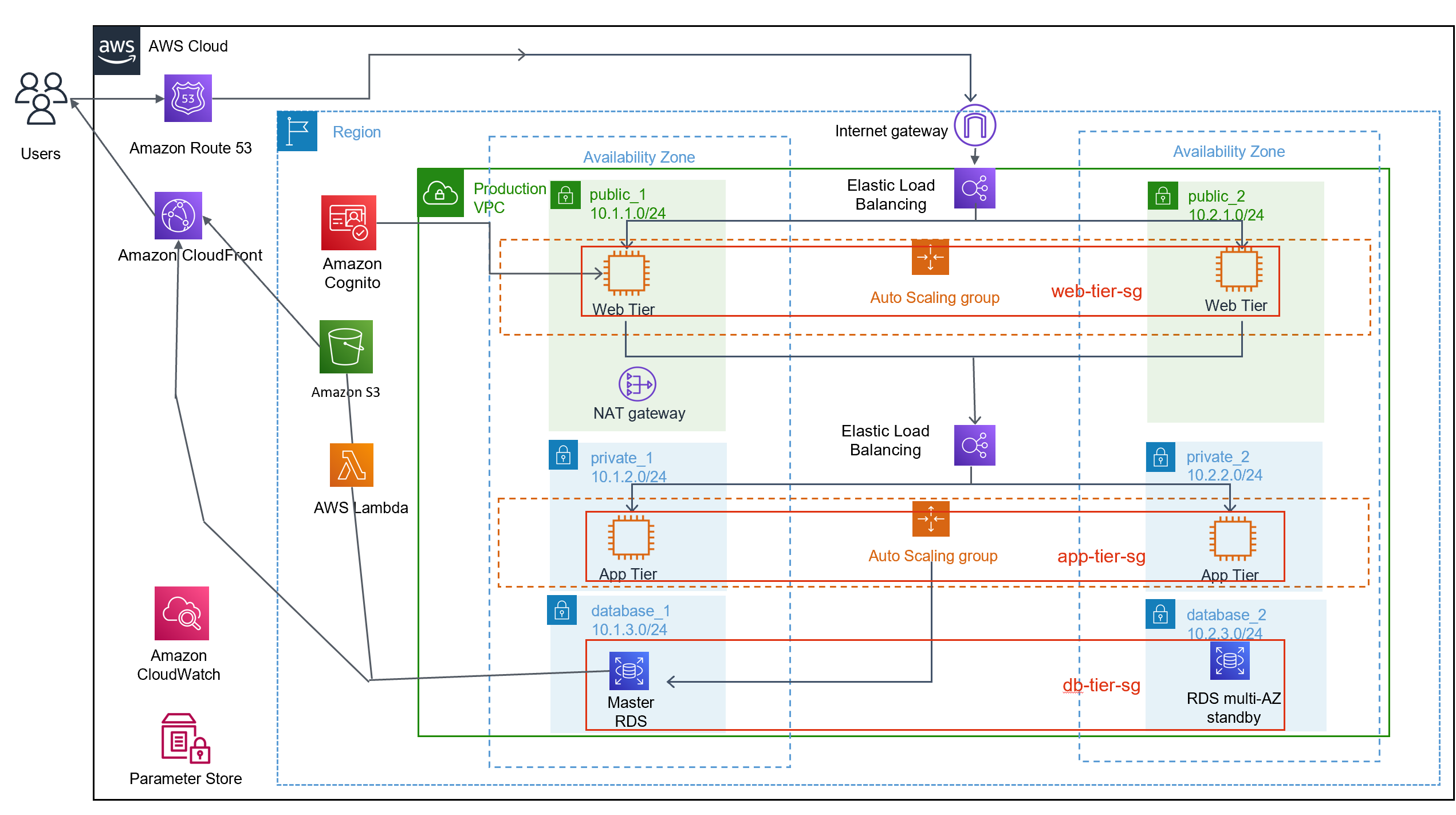
*Testing VPC Subnets Configuration*

**Web and Application Tier - Architecture Diagram**

In Future 1, users can access the services through the domain name which is provided by the service named 'route 53'. Route 53 helps in registering the name, DNS routing, and health checks. The request goes to the web tier elastic load balancer through the Internet gateway. The web tier load balancer will handle the traffic load and distribute the request into multiple availability zones. After that, the request will send to one of the web tiers and Cognito will help in signup or login. The Autoscaling group helps in maintaining the number of instances according to load. NAT gateway helps the resources in the private subnet to access the internet for the purposes like updating packages. The application tier load balancer will send the request to the application tier which is the middleware of the application. Data will be retrieved from the master and send to the user with the help of the Amazon CloudFront service. Amazon CloudFront helps in delivering the content faster and securely. AWS Lambda functions will automatically be executed when users uploads datasheet and photos. Datasheets will be extracted into the database, and the images will be converted into multiple formats. CloudWatch service provides us with data and actionable insights to maintain our application.

**Figure 1**

*Architecture Diagram*

****

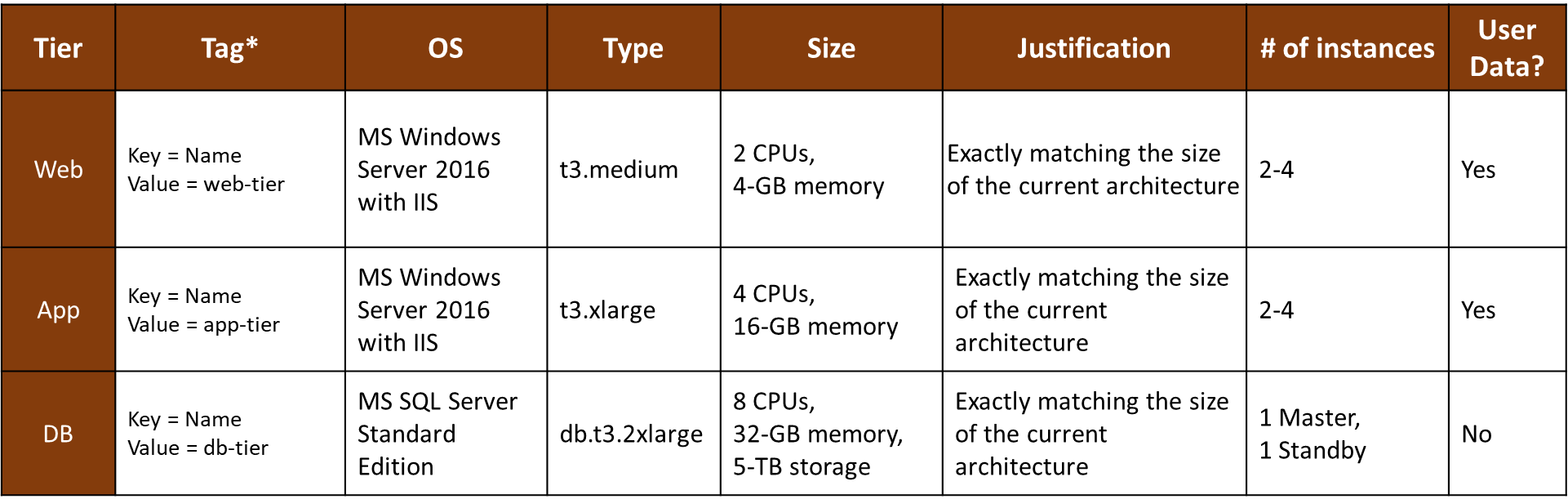
**Web and Application Tier**

***Instances Type***

AWS providing many different types of instances for the different workloads. Luckily, AWS is able to match the size of the current architecture. We going to use t3.medium for web-tier, t3.xlarge for application tier, and db.t3.2xlarge for the database.

**Table 4**

*Auto Scaling Group Configuration*

****

In Table 4, it shows the operating systems we choose will also match the current environment. They are Microsoft windows server 2016 with IIS for both web and application tiers. For database, since RDS is a fully managed service, and it doesn’t need to choose the operating system. Instead, we only need to choose the engine type and edition, which is the Microsoft SQL server standard edition.

Therefore, we can choose MS windows server 2016 AMI for both Web and Application Tier servers. However, they still need to have IIS installed, so using user data is the quickest and easiest way. With user data, the instances will automatically run the script when they start.

For the number of instances, we will have 2-4 for the web and application tier. The Business Continuity section will explain more details about autoscaling below. For the database, the size of RDS is enough, so we not going to use auto scaling. So, it has one master and one standby.

***Load Balancer and Security Groups***

There are two elastic load balancers in a VPC. The web elastic load balancer is to receive traffic from the internet, and the application elastic load balancer is to distribute traffic from the Web tier to the Application tier. In Table 5, it shows the security group setting of the two load balancers. The web load balancer should open the HTTP port from anywhere, and the application load balancer should only open port 8080 from the web-tier, which is a listening port.

**Table 5**

*Load Balancer Configuration*

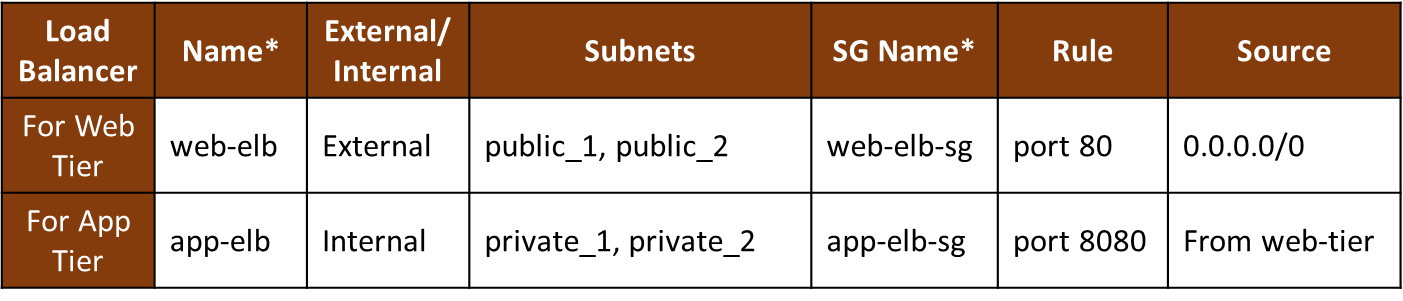
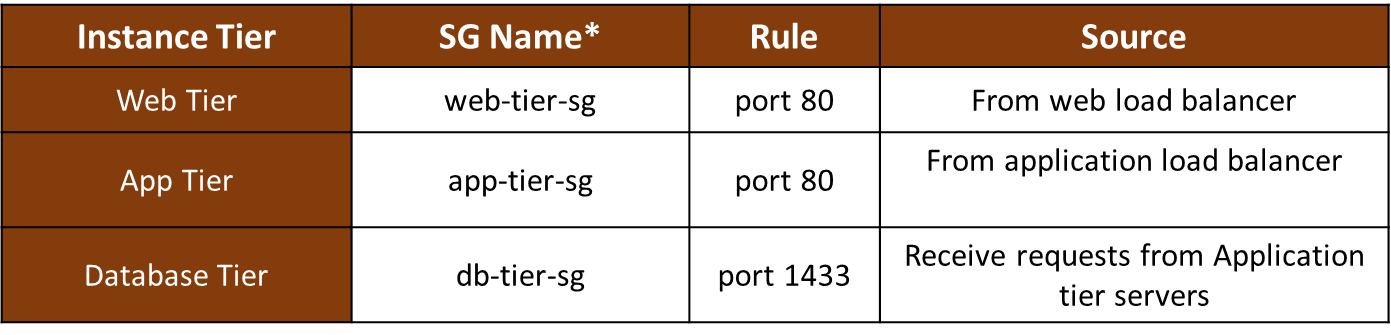


Table 6 shows the security groups in each tier. The web tier should open port 80 to receive requests from web tier load balancer, the application tier should open port 80 to receive requests from the application load balancer, and the database tier should open port 433 to receive requests from Application tier servers. All three tiers should permit administrative access on port 22, which is from SSH, but only from the medical company’s network.

**Table 6**

*Instance Tiers’ Security Group Configuration*



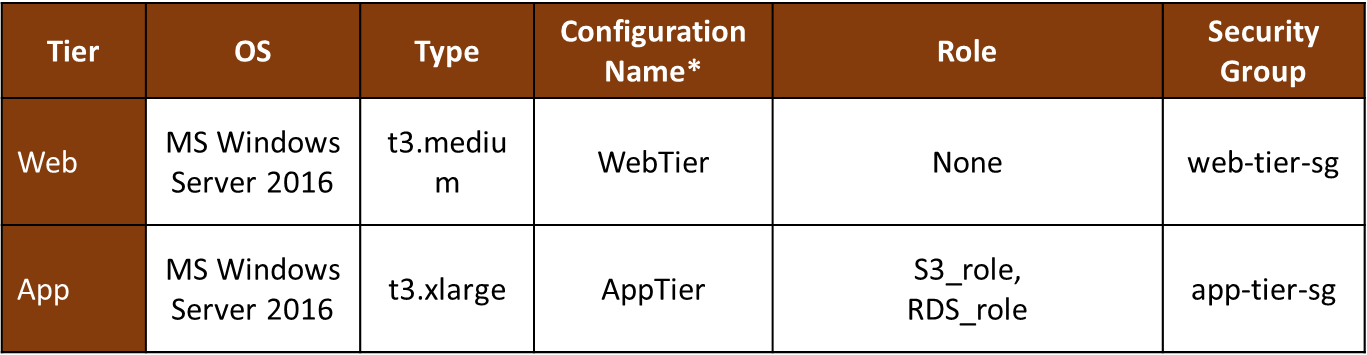
**Business Continuity**

***Auto Scaling***

The architecture will be having two auto scaling groups. One for the web tier, and one for the application tier.

**Table 7**

*Auto Scaling Group Launch Configuration*



For the auto scaling group launch configuration in Table 7, both tiers will have the same operating system and same instance type as mentioned above in the Web and Application Tier section. The IAM role for both tiers is EMR\_EC2\_DefaultRole.

**Table 8**

*Auto Scaling Group Configuration*

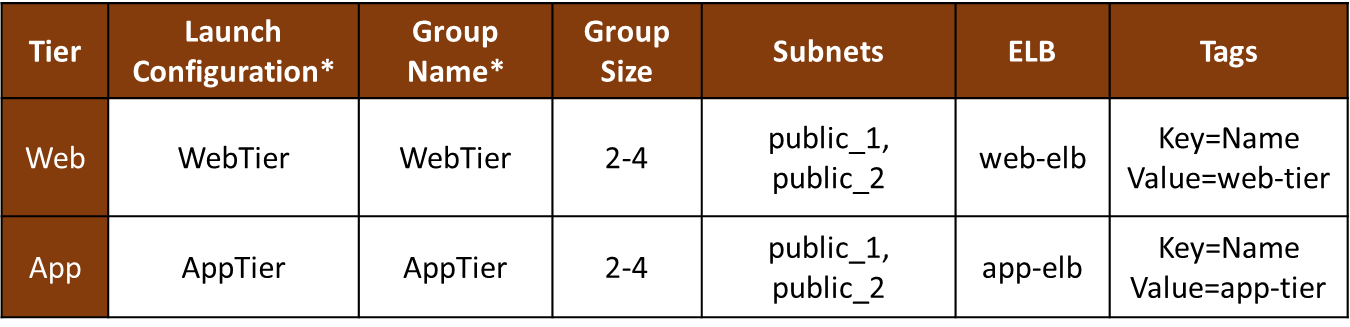


Table 8 is the auto scaling group configuration, in order to handle the double amount of current traffic, we will use autoscaling to scale between 2-4 for both web and application tiers.

**Monitoring and Auditing**

***AWS CloudWatch***

CloudWatch enables collecting logs, monitor the performance, trigger auto scaling alarms, and analyze metric data for fine-tune across all resources, applications, and services that run on AWS.

***AWS CloudTrail***

CloudTrail is used to recording and storing events for user activity in the AWS account. It also keeps auditing and generates alarms when security vulnerabilities are detected.

**Next Steps and Conclusion**

In conclusion, the architecture has the ability to protect data, systems, and assets to improve the security of the environment. IAM Permissions let users restrict their roles and access their services. User authentication techniques filtered the audience and allow them to utilize the services. Cost optimization in AWS enabled controlling spending on unused services.

After finish building the architecture, we will use a Proof of Concept (POC) approach for our customers to validate and test the architecture in a virtual cloud environment before the migration.

In the next step, we will consider slowly transfer the operating system to Linux OS with Apache installer for a better cost solution. Also, we will use containerized applications and deploy Kubernetes clusters. Furthermore, we will improve security by adding an SSL certificate.