Medical Company AWS Cloud Architecture

By: Rajeev Kumar Maram

Shivam Chauhan

Hoang Uyen Le

Introduction

This project aims to build and optimize a web service architecture on the AWS platform that achieves customer's business requirements. This time, our customer is a medical start-up company that provides software as a service for their customer. They're handling an online medical social networking and diagnosis assistance application reaching customers from various regions globally, including APAC, the US, and Europe. Some main tasks frequently generated on this application are setting up appointments between doctors and patients, then remoting consultations and diagnosis, as well as transferring prescriptions. The platform is also highly expected to provide the most convenient and stable tools for customers' best medical examination experiences. Therefore, it covers some additional functions of handling pictures provided by patients, doctors' documents, and direct online payment via this application. AWS Cloud architecture is designed to address the requirements of a medical application/company. This cloud architecture can host the development, test, and production environments.

Executive Summary

The medical company is planning to deploy its current environment to expand business coverage with a considerable number of users utilizing their platform shortly, therefore it is necessary to improve their infrastructure to ensure the application will adequately be launched. Additionally, the company strongly believes that cloud technologies will adequately support their business's rapid growth so that AWS web services will be a potential solution for the current issue. Our approach, first, is learning from their existing architecture to point out where its cons limiting application's functioning are. Meantime, communicating with the company's engineers to savvy their expectations and desired business approaches to improve the infrastructure comprehensively. Lastly, apply architecture techniques to create an ideal and optimal solution, not only for business workflow but also for cost-efficiency.

Requirements

Based on the discussions and current architecture diagrams, medical company and we arrived at the following customer requirements*:*

- Configure access permissions to conform to AWS best practices.
- The solution follows the AWS best practices
- The application should be secure to keep all the medical information safe.
- Resilient architecture (connects patients to doctors)

- No external access to the app or database tiers.
- Keep track of all actions
- Build an architecture that matches the current 3-tier architecture at the server hosting company, and that can handle double the number of servers.
- High security is required since all medical information and patients' payment method are especially sensitive, personally identifiable information (PII).

Assumptions

To build the best practice AWS environment, we work with some assumption for future plan:

- All the services are available in used regions
- Expecting many users from the US, Europe, and Pacific Asia regions.
- Expecting company's future business will grow rapidly

Solutions

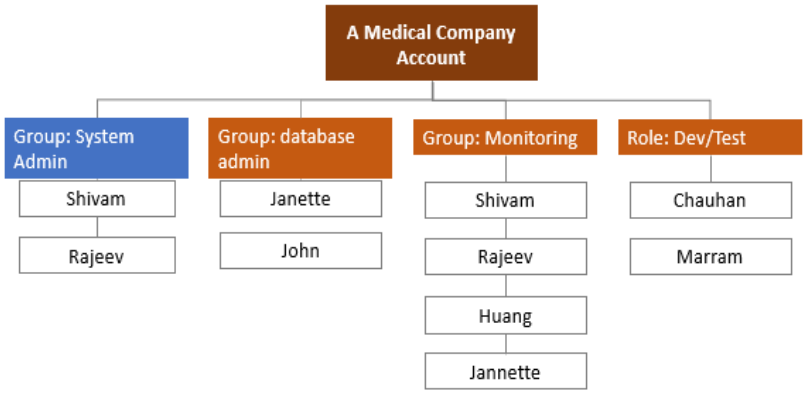
AWS Services which we have considered using for our architecture solution are:

- **EC2 Instances:** For the computation and running our application.
- **Elastic Load Balancer:** To divide the load among the instances.
- **IAM:** To manage access securely to resources and AWS accounts.
- **Amazon S3:** To store any amount of data.
- **SNS**: To send out notifications using SMS, and email to doctors and admins.
- **Amazon RDS**: Relational database to store structured data.
- **CloudFront**: To secure and accelerate your WebSocket traffic as well as API calls.
- **CloudTrail**: It enables governance, compliance, operational auditing, and risk auditing of the AWS account.
- **AWS Shield**: To protect from Distributed Denial of Service (DDoS) and safeguard applications running on AWS.
- **Amazon Redshift**: Data Warehouse for the large data-generating application.
- **Route 53**: For highly available and scalable cloud Domain Name System (DNS) web service.

Solution - User Authentication

These are the requirements for the user group for this company.

- System Administrator Group: 2 users
- Database Administrator Group: 2 users
- Monitoring Group: (monitors 4 users)
    - infrastructure resources (EC2, S3, RDS for the app)



As per the requirement of the company, we have 2 system administrators, 2 database administrators, and 4 users in the monitoring group. We also have defined the dev/test role where users will be able to access the resources used in the development environment.

The below table shows the groups/role and permission associated with the respective groups or roles.

Group/Role #	Group/Role Name	Permissions
Group	System Admin	Administration Access
Group	Database Admin	AmazonRDSFullAccess, AmazonS3FullAccess
Role	Dev/Test	WebTierFullAccess, ApptierFullAccess, DevFullAccess, AmazonEC2FullAccess
Group	Monitoring	AWSCloudTrailFullAccess, AWSCloudWatchFullAccess, AmazonS3FullAccess, AWSRDSFullAccess, AmazonEC2FullAccess

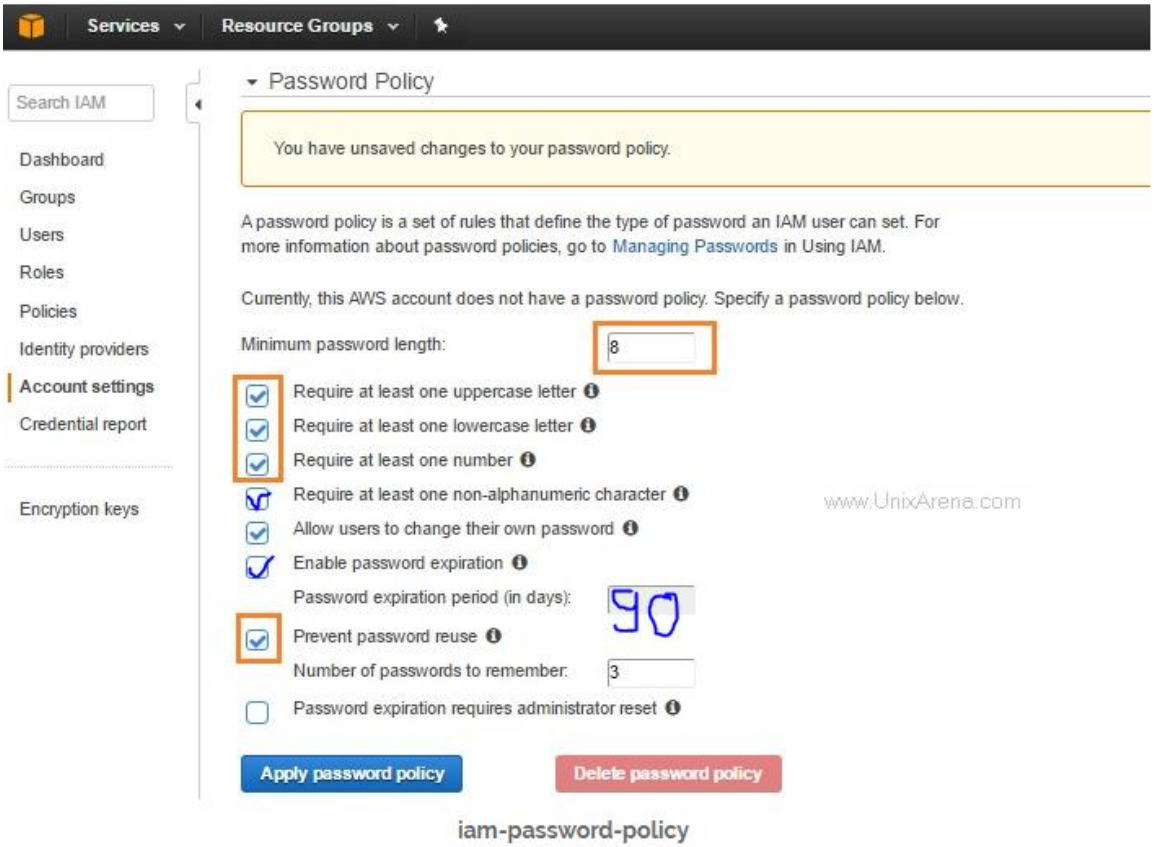
- For the system admin group, am Administration Access permission is created for full system access.
- For Databases Admin, Full access to RDS and S3 is given to handle the data.
- For the Monitoring group, Full access to CloudTrail, CloudWatch, S3, RDS, and EC2 is given for monitoring and tracking all the logs.
- Another role Dev/Test is created for the development and testing environment.

Requirements for password policy.

- Administrators require programmatic access and AWS Management Console access.
- When signing into the console, each administrator is required to provide a user name, a password, and a randomly generated code provided by the Virtual MFA.
- All other users should only have AWS Management Console access, using a combination of username and password.

- A password with at least 8 characters, 1 uppercase and 1 lowercase letter, 1 number, and 1 special character
- Forced password change every 90 days
- No re-use of the previous three passwords

Requirement	Solution
Should be at least 8 characters and have 1 uppercase, 1 lowercase, 1 special character, and a number.	Set a password policy.
Change passwords every 90 days and ensure that the previous three passwords can't be re-used.	Set mandatory password rotation periods of 90 days by updating Password Policy page.
All administrators require programmatic access	Configure programmatic access includes access that is internal to your workload, and access to AWS related resources.
Administrator sign-in to the AWS Management Console requires the use of Virtual MFA.	MFA is active for the password policy



iam-password-policy

IAM users can set their password (with at least 8 characters, 1 uppercase and 1 lowercase letter, 1 number, and 1 special character). We have also enabled password expiration in 90 days. Also, we have enabled password reuse and the number or passwords to remember is 3 as per the requirements.

Solution – Network & Security

The requirements for network design are the following:

- Networks that conform to AWS best practices while providing all the necessary network services to the application in their different environments.
- An architecture that matches the current architecture at the server hosting company and that can handle double the number of servers.
- Security for all medical information, as medical information usually contains highly sensitive personally identifiable information (PII).
- Load balancers for web tier and application tier that must support HTTP, HTTPS, TCP protocols plans to move their application into AWS.

AWS best practices that new architecture must conform:

- Achieve high availability for all tiers to reduce downtime.
- Control access to the application and limit public entry points.
- Minimize the IP address used to reduce the attached surface.
- Maintain separate networks for A Medical Company's development/testing environment and the production environment.
- The web tier load balancer can receive requests from the Internet on port 443.
- Web tier servers can receive requests from the web tier load balancer only on port 443.
- The Application Load Balancer can receive requests from the application tier load balancer only on port 443.
- Database servers can receive requests from application servers only on port 433.

VPC	Region	Purpose	Subnets	AZs	CIDR Range
1	us-east	Main for for US and Europe	4	2	10.0.0.0/16
2	ap-southeast	Main for APAC countries	4	2	10.1.0.0/16

This company is expecting users from around the world i.e from the USA, Europe, and APAC countries. To make this application Highly Available and well structured that can serve globally, we designed a multi-region architecture. Two regions we have used are us-east which is the USA and ap-southeast which is Singapore. Both the regions will have 2 Availability zones with the IPv4 CIDR block of 65,536 IP addresses for each region.

The Dev/Test subnet solution.

Subnet Name	VPC	Subnet Type (Public/private)	AZ	Subnet Address
Us-NAT	#1	public	Us-east-1,us-east-2	10.0.1.0/24
Us-Web	#1	private	Us-east-1,us-east-2	10.0.2.0/24
Us-App	#1	private	Us-east-1,us-east-2	10.0.3.0/24
Us-Database	#1	private	Us-east-1,us-east-2	10.0.4.0/24
Ap-NAT	#2	public	ap-southeast-1, ap-southeast-2	10.1.1.0/24
Ap-Web	#2	private	ap-southeast-1, ap-southeast-2	10.1.2.0/24
Ap-App	#2	private	ap-southeast-1, ap-southeast-2	10.1.3.0/24
Ap-Database	#2	private	ap-southeast-1, ap-southeast-2	10.1.4.0/24

Development and Test group has full access to all the subnet which will support the functioning of development and test environment. There are a total of 8 subnets (2 Public & 6 Private) and 4(1 Public & 3 Private) subnets in each region.

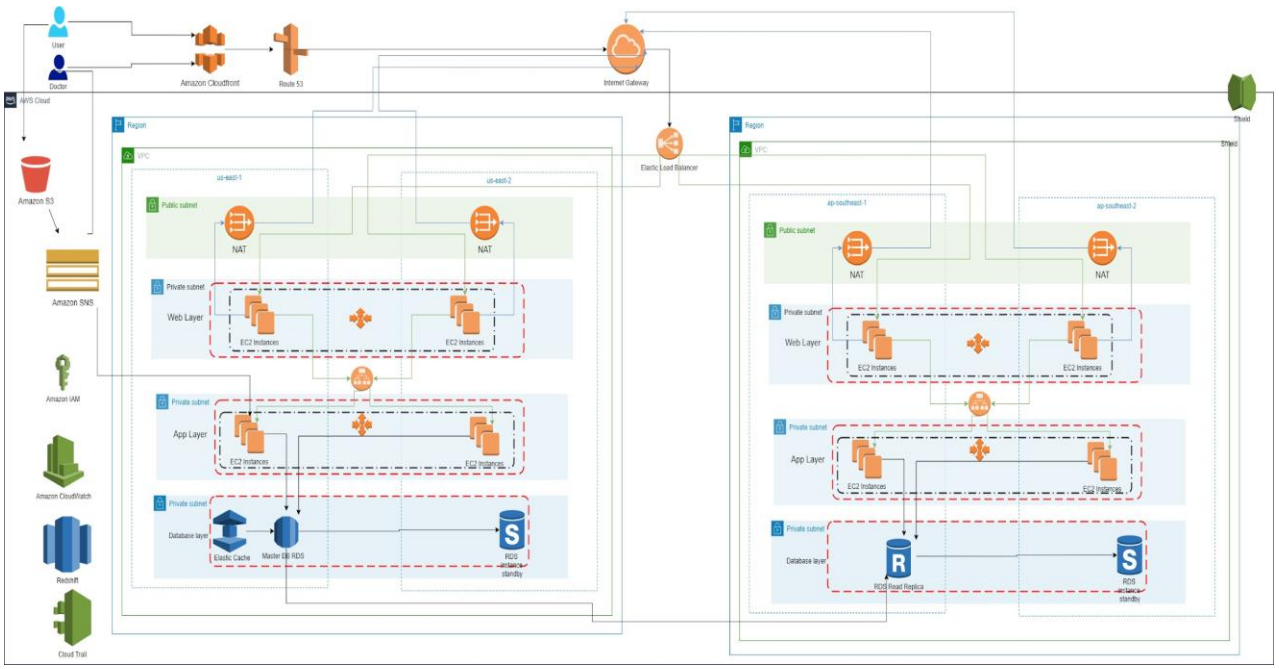
The Production subnet solution.

Subnet Name	VPC	Subnet Type (Public/private)	AZ	Subnet Address
Us-Web	#1	private	Us-east-1,us-east-2	10.0.2.0/24
Us-App	#1	private	Us-east-1,us-east-2	10.0.3.0/24
Us-Database	#1	private	Us-east-1,us-east-2	10.0.4.0/24
Ap-Web	#2	private	ap-southeast-1, ap-southeast-2	10.1.2.0/24
Ap-App	#2	private	ap-southeast-1, ap-southeast-2	10.1.3.0/24
Ap-Database	#2	private	ap-southeast-1, ap-southeast-2	10.1.4.0/24

Production will have access to 3 tier web, app, and database because it is a startup with fewer people, most of the employees have full access to most of the subnet.

The IPv4 CIDR block for all the 8 subnets is /24 i.e it supports 256 IP addresses.

Solution – Web and Application Tier



This application connects patients and doctors. You can see two end-users in the above diagram. These users will request to use this application via URL. DNS resolution can be achieved using Amazon Route 53. Moreover, our first priority is the security of protected health information, so to make sure that we do not use any protected health information in the URL.

This application also allows the patient to upload documents to Amazon S3. Once the document is uploaded, SNS sends a notification via SMS and email to the user's respective doctor and the application will extract text from the document and store data in RDS. Now doctors will have access to that information. For fast content delivery and reduce latency, we have also used Amazon CloudFront.

Route 53 sends the user request through an internet gateway to the web-elb Elastic Load Balancer. It supports HTTP, HTTPS, TCP protocols to ensure secure communication. This ELB is the primary entity and is internet facing which receives requests from the Internet on port 443 and sends requests to instances in the back end using HTTPS on port 443.

The first Public Subnet which has NAT gateway is an extra layer of security. This application should not have external access, so we have included this layer. It prevents the internet from initiating a connection with the instances.

The next later is web tier in the auto-scaling group for HA. The instances in this group are placed in a security group to secure the access to these instances. accept requests only from the designated web-elb load balancer.

There is an app-elb load balancer with HTTPS listener configured. This receives a request from the web tier and sends it to app layer instances in the app tier.

The app layer is also on the private subnet with auto-scaling and security groups. This will ensure that our app is stable and highly responsive.

All the instances in the app tier support EBS optimization as per the requirement.

In the database layer, we have used Amazon RDS. To ensure security we have kept this layer in the security group. It has access only from the application tier. Encryption is enabled because it is storing PHI data. We have used Elastic cache so that we don't lose the session data. We have one RDS read replica for back-up and 2 RDS standby instances in case the master DB fails.

Requirements for web and application tier:

- All the instances in the application tier must support EBS optimization
- Load balancers for web tier and app tier must support HTTP, HTTPS, TCP protocols
- All web tier instance names should be tagged as key as Name and value as web tier
- All application tier instance names should be tagged as key as name and value as app tier.

Tier	Tag*	OS	Type	Size	Justification	# of instances	User Data?
Web	Key = Name Value = web-tier	Amazon Linux	t2.medium	2vCPU/4 GiB	Network Performance: Low to Moderate	2	AMI
App	Key = Name Value = app-tier	Amazon Linux	t2.xlarge	2vCPU/16 GiB	Network Performance: Moderate	2	AMI
DB	Key = Name Value = db-tier	Amazon Linux	t2.2xlarge	4vCPU/36 GiB	Network Performance: Moderate	4	AMI

- For web-tier the ec2 instance that we used is t2.medium which has 2vcpu/4GBmemory and has network performance in between low and moderate.
- For the app-tier, the ec2 instance that we used is t2.xlarge which has 2vcpu/16GB memory and has network performance is moderate.
- For the DB-tier the ec2 instance that we used is t2.2xlarge which has 4vcpu/36GB memory and has network performance is moderate.
- All these tiers will run on Amazon Linux operating systems.
- All these instances support Amazon EBS encryption.

The load balancer and instance security group details:

Load Balancer	Name*	External/Internal	Subnets	SG Name*	Rule	Source
For Web Tier	web-elb	External	public	web-elb-sg	HTTPs–port 443	Internet Gateway
For App Tier	app-elb	Internal	private	app-elb-sg	HTTPS- port 443	Web-tier

Instance Tier	SG Name*	Rule	Source
Web Tier	web-tier-sg	Receives on port 443	Internet Gateway
App Tier	app-tier-sg	Receives on port 443	web-elb
Database Tier	db-tier-sg	Receives on port 443	app-elb

For accessing web tier with elastic load balancer, we used a public subnet to route patients from their network through an internet gateway using HTTPs- port 443. For application tier, the source would be web tier using the same HTTPs- port 443 that is placed in a private subnet and is external.

Solution – Business continuity

Requirements Business continuity

- The web and app tiers should be resilient and designed for business continuity so that If a server becomes unavailable it will be replaced by a new server.
- It is considered as unavailable if the os or application fails to respond
- The database tier should support Multi-AZ deployment and it should handle doubling the number of servers to support its rapid growth

AWS business continuity outlines measures to avoid and reduce environmental disruptions. It includes operational details about the steps that we take before, during and after an event. It helps businesses recover from a disaster and resume operations with as little disruption as possible.

Tier	OS	Type	Size	Configuration Name*	Role	Security Group
Web	Amazon Linux	t2.medium	2vCPU/4GiB	WebTier	System admin	web-tier-sg
App	Amazon Linux	t2.xlarge	2vCPU/16GiB	AppTier	System admin	app-tier-sg
Database	SQL Server	t2.2xlarge	4vCPU/36GiB	dbTier	Database admin	db-tier-sg

To satisfy business continuity requirements, the autoscaling groups are configured as mentioned in the above table with 2 CPUs and 4GB in the web tier with amazon Linux as the operating system and 2CPUs and 16GB in the application tier. In the database tier, SQL server with 4 CPUs/36GB and only the database administrator has access to it.

Tier	Launch Configuration*	Group Name*	Group Size	VPC	Subnets	ELB	Tags
Web	WebTier	WebTier	Min-2,Max-4	#1,#2	us-web, ap-web	web-elb	Key = Name Value = web-tier
App	AppTier	AppTier	Min-2,Max-4	#1,#2	us-app, Ap-app	App-elb	Key = Name Value = app-tier
Data base	DatabaseTier	DbTier	Min-2,Max-4	#1,#2	us-database, Ap-database	-	Key = Name Value = database-tier

Solution – Auditing

Requirements for auditing

- Three user groups who have access will continuously monitor, and retain account activity related to actions across your AWS infrastructure.
- Logging and auditing the history of user activity including actions from the management console, SDKs, command-line tools.
- We should ensure that it is an audit trail for all API calls and logs should be saved in a secure location.

Solution

We can continuously monitor the working of all the services using Amazon cloud watch and Amazon Cloud trails for api audit logging.

- Evaluate the ability of AWS services to meet information security objectives and ensure future deployments within the AWS cloud are done in a secure and compliant way
- Assess your existing organizational use of AWS and to ensure it meets security best practices
- Develop AWS usage policies or validate that existing policies are being followed
- To assist customers when they evaluate the security controls required by their specific industry or governing body.

Next steps

We would implement a serverless architecture lambda and use AWS glue for extract-transform-load. We would also implement automated infrastructure to cut down most human work.

Conclusion

With the architecture that we designed it is easy to migrate medical company's whole infrastructure to AWS cloud which made patients and doctors online interaction easy, they will be able to use social networking application and patients will be able to make online appointments, doctors can take general information about patients, can provide electronic prescriptions and also payment can also be done. Our architecture met the company's hardware and software requirements. We have used a wide range of services in which the most significant ones are EC2 instances, Elastic Load Balancer, IAM, S3, RDS, CloudFront, CloudTrail, Shield, Redshift and Route 53. We met requirements like user authentication where we divided the user groups and gave specific access to everyone. We will request every user to create their own passwords which meets all requirements and MFA enabled. We made our ec2 instances highly available so that it can be accessed across US, Europe and APAC around the world. We have given top priority to security and the patients who are registered can only access the online service through route 53 and then internet gateway. We have added a business continuity module to make sure all the services are working 24/7 and the damaged server will automatically get replaced. Finally, we met user authentication, password policy, network, security, web and application tiers, business continuity, auditing and monitoring requirements.

References:

- aws (May 2020) - *Architecting for HIPAA Security and Compliance on Amazon Web Services* Retrieved from: https://d0.awsstatic.com/whitepapers/compliance/AWS_HIPAA_Compliance_Whitepaper.pdf

- Engdahl, S. (2008). Blogs. Retrieved June 27, 2020, from https://aws.amazon.com/blogs/startups/architecting-your-healthcare-application-for-hipaa-compliancepart-2/
- https://aws.amazon.com/rds/features/multi-az/