R SYSTEMS

PROCTOR FINANCIAL

CASE STUDY:

Cloudification & Migration

On-Premise App to AWS Cloud

PROCTOR FINANCIAL Improved User Experience & Satisfaction By Leveraging AWS Capabilities, Delivered by **R SYSTEMS**

www.rsystems.com

Client Overview

Proctor Financial, Inc. is a leader of lender-placed hazard and flood insurance, including hazard tracking for the mortgage servicing community. Since 1970s, Proctor has blended comprehensive insurance programs with innovative tracking & reporting services to meet specific servicing needs of mortgage servicers, community banks, credit unions, sub-servicers, distressed asset investors, and the U.S. Government. The company provides portfolio construction, retirement & education planning, and investment management services to individuals, charitable organizations, estates & small businesses.

Problem Statement

The rising brand competition in the marketplace compelled Proctor to migrate to a robust and scalable rewards platform, that was required to endorse the brands of many insurance providers through their Electronic Health Records (EHR) partners, worldwide.

R Systems Solution

Delivered cloud-based migration solution using various AWS tools & services e.g. EC2, VPC, VPN, Application Discovery Service, Transit Gateway,, Server Migration Service, Database Migration Service, etc. to connect every family with dedicated primary care.

R Systems leveraged its global delivery model to provide required services for this project. The engagement roadmap was planned with due consideration to the client's goals. Agile Scrum methodology was adopted to expedite the app development, migration and faster time-to-market. R Systems' teams focused on providing rich Quality of Service (QoS) & Experience (QoE) to the client via innovation and by adopting industry best practices.

Activities Performed By **R Systems** To Achieve Desired Results:

- Used OAuth2 security to authorize/authenticate apps
- Implemented versioning restful services for backward compatibility
- Deployed CodePipeline, Docker & code repository to provide robust continuous integration & delivery solutions for quick deployment from staging to production environment
- Created VPN tunnel between AWS & client datacenter
- Implemented AWS transit gateway
- Used CloudWatch for centralized logs & dashboards to debug problems easily

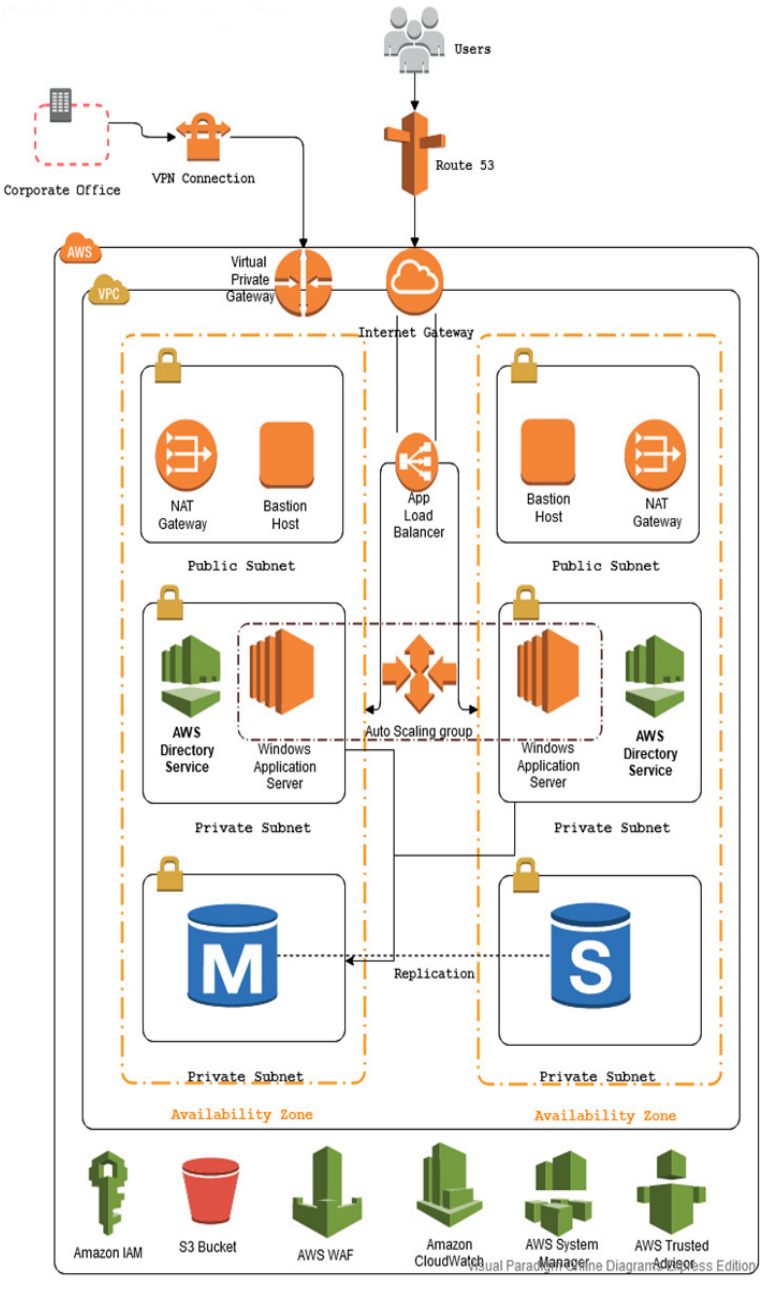
Continued...

Continued...

- Finetuned performance using AWS CloudWatch & CloudTrail and configured CloudWatch events for instant possible remediation
- Used CloudWatch matrix for monitoring, and an early identification & resolution of problem areas
- Deployed API monitoring with API Gateway metric which further sends data to CloudWatch
- Integrated AWS API Gateway
- Implemented application status health check that'd give various health status for each service and data for monitoring & performance tuning of services
- Configured Auto Scaling groups in relation to CloudWatch events for dynamic scale up and scale down capabilities of EC2 instances
- Verified response times, request count & stability of the system under increasing user loads with the help of CloudWatch Agent
- Introduced AWS CloudFront content distribution to decrease the latency between user and the app
- Leveraged Trusted Advisor to keep the Amazon EC2 Windows updated with AWS-provided Win. drivers
- Used AWS Systems Manager SSM AWS Support-Upgrade Windows AWS Drivers to easily apply the updates across our instances
- Performed tests using EC2 solution to ensure all instance types are properly configured, instance topology is appropriate for the workload and the high-performance features are leveraged
- Created SSO to access AWS accounts & cloud apps in the AWS SSO user portal for on-premises AD, and adhered to 2-way trust relationship, along with an AD connector
- Employed AWS Systems Manager to scan & install missing patches to large groups of instances using Amazon EC2 tags

- Automated a variety of maintenance & deployment tasks through AWS Systems Manager
- Defined the configuration options & policies centrally for managed instances
- Implemented short living utility tasks (like create & update DB/utilities) to perform further actions for facilitating a quick deployment

Architectural Diagram



Services Overview

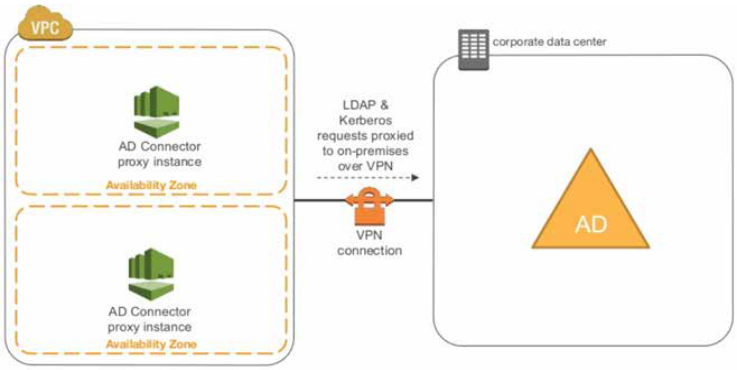
- **EC2 –** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity on cloud. For web tier, we chose t2.xlarge instances running MS Windows 2019 & IIS in Auto Scaling to provide elastic compute capacity based on demand. For app tier, chose t2.2xlarge instances as the app was compute intensive. App servers were also configured to leverage Auto Scaling

Best practices were followed to ensure flexibility, operational excellence & availability of:

- **Security & Network –** All users were accessing AWS console/resources through their individual IAM roles. Multi-factor Authentication (MFA) was made compulsory the AWS management console login.

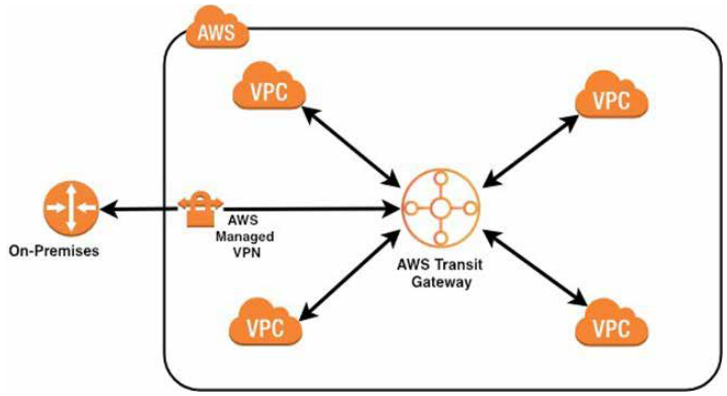
1. Configured Apps on EC2 to either use IAM roles or AWS managed directory (need based)
2. Configured security groups were with least permissive rules
3. Configured VPN to secure the traffic coming from Proctor on-premises datacenter to AWS
4. Set-up all servers including web & app were in private subnet, so they aren't accessible to the outside world directly
5. Hardened bastion hosts were configured to access Ec2 servers in private subnets
6. Patched all servers' operating systems regularly using System Manager

- **AWS AD Connector with SSO –** AWS AD connector was configured to proxy directory requests, to on-premises Microsoft AD to use their existing on-premise creds to access AWS apps

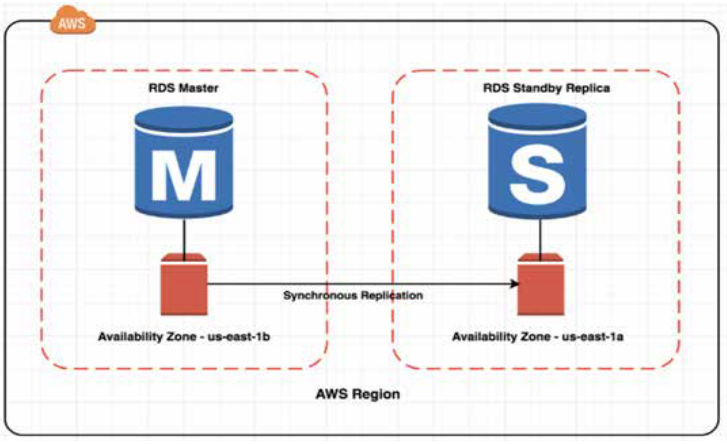


- **AWS VPN Tunnel –** AWS VPN tunnel was configured between AWS VPC & on-prem datacenter to connect with the network, and set up virtual private gateways & customer gateways for VPN

- **AWS Transit Gateway –** AWS transit gateway was set up for network connectivity with multiple accounts, VPC, on-prem datacenter



- **RDS –** AWS RDS service was used for database; MS SQL server 2017 Enterprise edition was configured in Multi-AZ environment with synchronous replication and configured automated daily backups



Migration:
Tools & Technologies

- AWS Technology Stack
- AWS CloudWatch
- OAuth 2
- API Gateway
- AWS EC2
- AWS CloudTrail
- App Load Balancer
- CodePipeline
- Git Code Repository
- NAT Gateway
- Auto Scaling Groups
- AWS CloudFront
- Internet Gateway
- AWS S3

Benefits
Reward Platform

- Improved user experience & customer satisfaction
- Improved adaptability services for multiple use cases
- Simplified process to build & maintain apps
- Enabled quick identification and resolution of the root cause of poor application performance
- Simplified log analysis & correlation tasks
- Increased flexibility & scalability with optimized costs

Challenges Faced
Before Implementation

- Maintaining seamless backward compatibility with the existing processes
- Migrating heavy apps to a light-weight microservices architecture
- Providing continuous integration and delivery solutions for faster development & deployment
- Monitoring & identifying problems automatically
- Configuring maintenance of all components across the environments
- Performing dynamic scale-up/-down of microservices architecture in the cloud
- Logging & dashboarding centrally, for easy debugging
- Providing optimum solution architecture, designs & ensuring full support with minimum turnaround time
- Managing minimal latency between the users & app
- Establishing connection between the AWS VPC, and client datacenter
- Integrating cutting-edge technologies with key tools