AWS Migration Redesign/Refactor Playbook

Contents

[Introduction 3](#_Toc529338538)

[Scope 3](#_Toc529338539)

[Playbook 3](#_Toc529338540)

[Oracle DB Migration to AWS RDS Oracle 4](#_Toc529338541)

[Oracle DB Migration to AWS Aurora DB 5](#_Toc529338542)

[3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon MQ and Oracle RDS 6](#_Toc529338543)

[3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon MQ and Aurora DB 7](#_Toc529338544)

[3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon SQS and Oracle RDS 8](#_Toc529338545)

[3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon SQS and Aurora DB 8](#_Toc529338546)

[AWS Migration Architecture for Pega BPM Tool 10](#_Toc529338547)

[AWS Migration Architecture for Oracle BPEL Tool 10](#_Toc529338548)

[AWS Migration Architecture for Oracle Data Warehouse (DWH) Solution 10](#_Toc529338549)

[AWS Migration scenarios using NetApp 10](#_Toc529338550)

# Introduction

The purpose of this document is to develop a technical playbook for the migration of on premise/data center based apps to the AWS cloud. The design consideration here take more of a green field approach when it comes to recommending services on the AWS cloud, to leverage for the migration efforts. There is another parallel playbook being developed which will address AWS cloud migration more from an incremental evolution perspective. This alternate playbook can be seen as providing lower risk and quick win strategy, however the tradeoff with that strategy is it might not be able to leverage all the rich features of a cloud-first or cloud-native approach. It will nonetheless offer a first-step or pathway to migrate from the physical data center to the cloud. The recommendations in this playbook propose using more of the cloud-native or cloud-first approach solution choices for refactoring/retooling, but does come with longer cycle/development times as new services are recommended which have not been used before. It may also potentially require a lot of different systems/applications to be refactored, where some of the applications touched may go deep into the customers’ suite of applications and its network, to incorporate the recommended AWS services and integrate with them.

# Scope

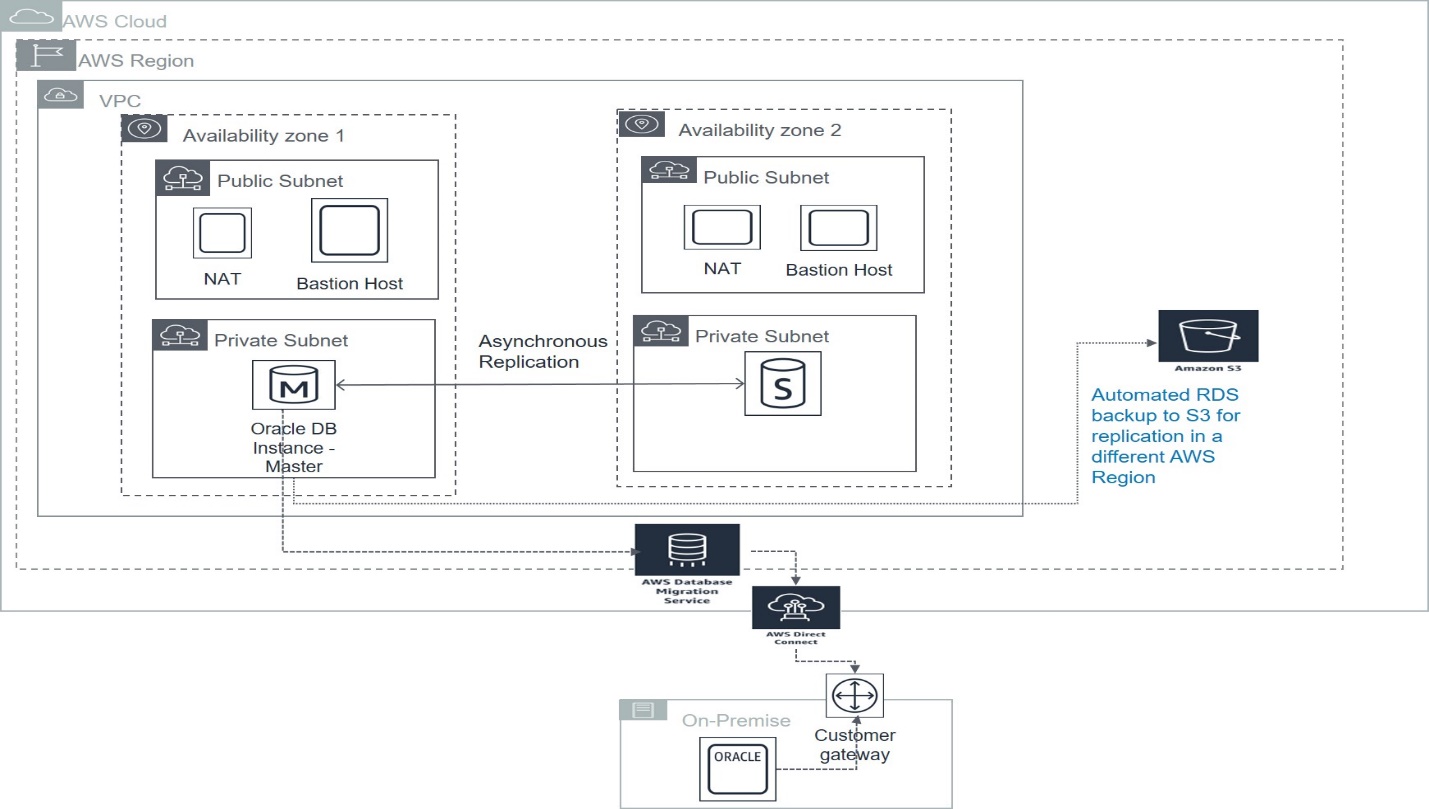
Cloud migration is a huge area for one single playbook to cover in its entirety. This is more a limited purpose playbook, with the goal to recommend alternate options for migration, when the customer solutions are heavily reliant on Oracle-based tools with their application development and deployment, starting off with Oracle database and perhaps also using middleware tools from Oracle such as Fusion, Oracle BPEL, etc. We will also assess paths to migrate away from any Oracle Data warehouse solutions that may be part of the customers’ enterprise suite and architecture.

# Playbook

The playbook will take a similar (not exact) approach to a design patterns catalog with using a phrase/sentence to describe the solution each playbook recommendation is targeting, and will be organized as a collection of recommendations as they evolve, starting off with migrating away from the Oracle database, and slowly moving up the chain to other parts of the application stack. The recommendations while heavily focused on migration away from Oracle tools, are however not limited to just the Oracle suite tools. The focus will also include using open source alternatives to development, an understanding of the customers’ enterprise architecture strategies and toolsets, as well as basing the solutions on using cloud-first or cloud-native approaches. These alternative may lead to longer development cycles but will align with more long-term cloud migration goals, and realizing its full benefits, instead of being driven based off of short-term goals with limited benefits. Each playbook recommendation will come with an architecture/blueprint diagram as well as a synopsis of the features of the solution and its benefits. There are always multiple design choices when it comes to developing cloud migration strategies, and there is no one-size that fits all. Our goal here is to propose alternatives that provide the best value to their long-term goals, while being cognizant of the added development costs that these proposed solutions may require. The intent of the playbook is not to review every possible design alternative that maybe available, but propose a reasonable set of strategies to cloud migration, keeping the required criteria in mind. The scope as outlined before is also to exclusively focus on migrating away from an Oracle toolset, further narrowing the playbook recommendations discussed in this whitepaper. This whitepaper is also limited in scope, by focusing on pilots or migration of test environments, and not targeted for production workloads. The goal essentially here is to prove these migration architectures out in a test environment, before recommending them for further consideration in a production environment.

## Oracle DB Migration to AWS RDS Oracle

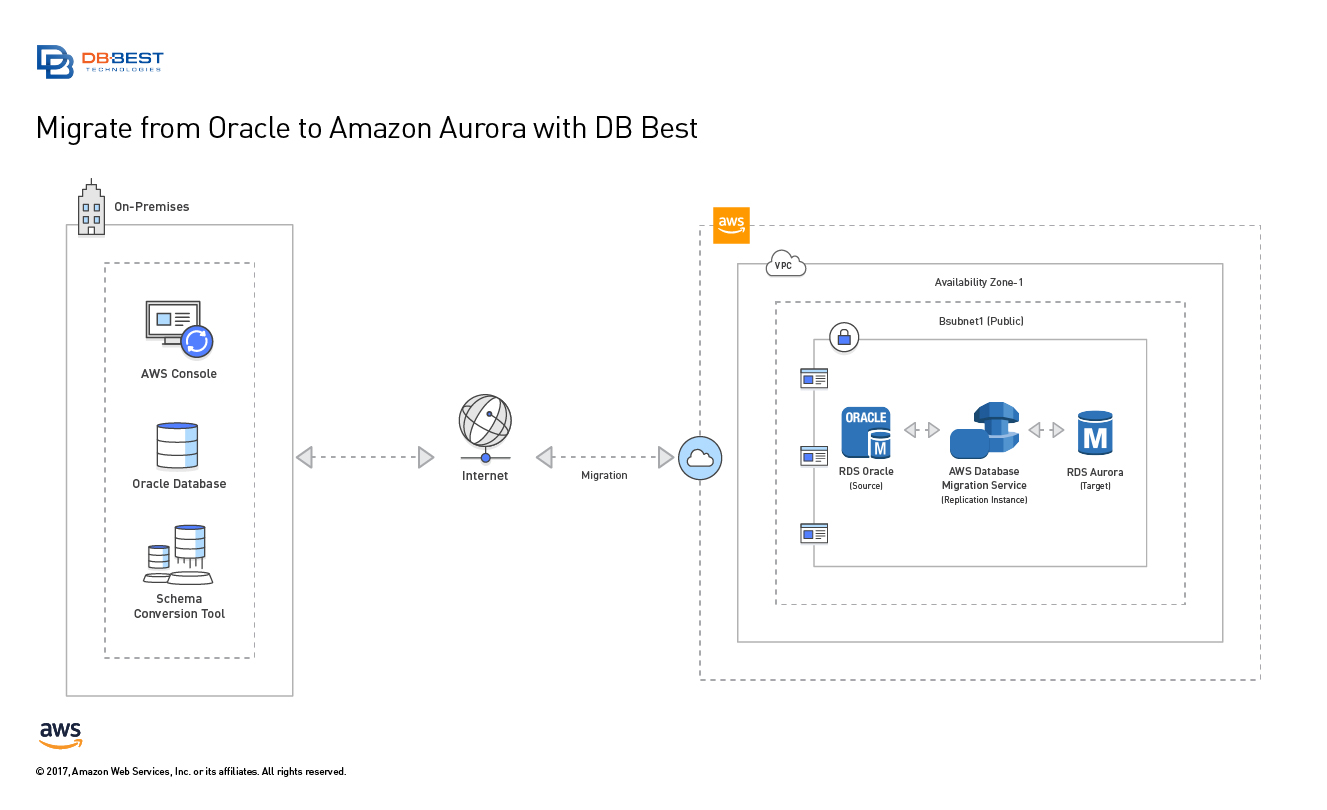
We recommend using a managed Multi-AZ Oracle highly available RDS solution as part of migrating away from enterprise applications which have an Oracle backend/database. This eliminates the need for reinstalling Oracle from scratch on an EC2 instance or starting off from an AMI, and performing lot of the administration yourself. This level of control maybe needed in certain scenarios, but using RDS and having AWS do all the management and administration, will liberate the organization from doing more mundane IT/administration tasks, and focus on delivering the needed application functionality. The proposed architecture also recommends using AWS data migration service to transition data from the customer’s data center to RDS over Direct Connect or a VPN connection, whichever maybe available. The architecture diagram shows Direct Connect, but it could easily be used over a VPN connection as well. The data from the RDS primary instance is asynchronously replicated over to the secondary which will become available in case of a failover. The managed RDS service also automatically backs up the data from the primary RDS instance over to S3, which could be used to build a new RDS instance in a different region for full disaster recovery. The multi-AZ managed RDS solution does provide failover within a region, but if there is a broader regional outage, then having a cross-region failover capability will be more useful here and also meet the needs of the customers. The NAT and bastion host instances are typically provided for outward and inward access from and to the instances. The RDS instance should always be hosted in private subnet, and access through the bastion should be the only mechanism to reaching the managed RDS instance from the outside. This is a security best practice that is a common/recurring theme across AWS configurations, to making instances accessible/available to the outside world.



**Figure 1: Migrate an on premise Oracle database to a multi-AZ managed AWS RDS Oracle setup**

## Oracle DB Migration to AWS Aurora DB

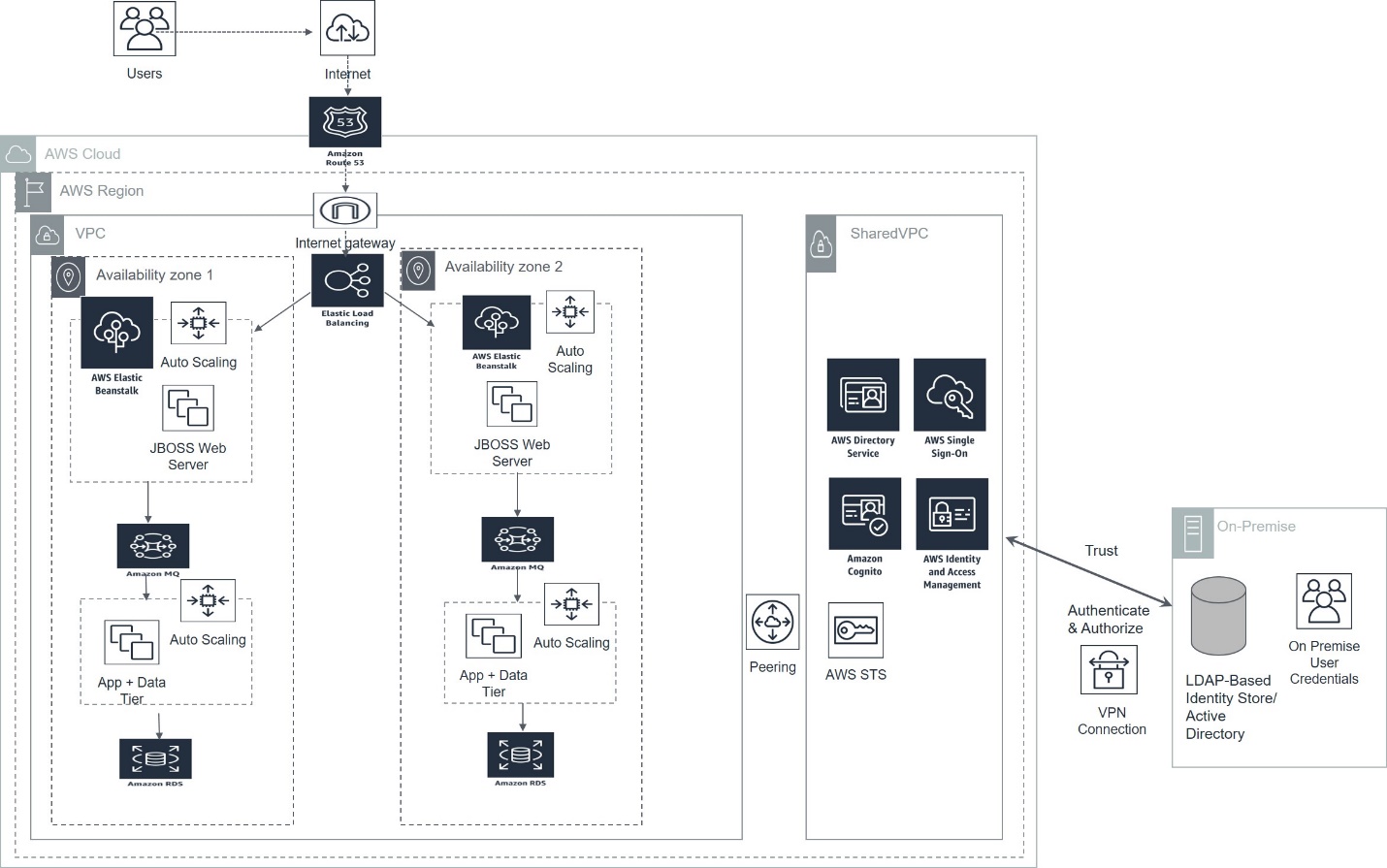
**<< Stock Photo from AWS Site – Need new diagram and associated content describing the scenario>>**



**Figure 2: Migration of an on premise oracle database backend to AWS Aurora DB**

## 3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon MQ and Oracle RDS

The playbook recommends migrating from any proprietary development/development such as IBM WebSphere Application Server (WAS), Oracle Application Developer Framework (ADF), Oracle Fusion Middleware, IBM MQ Broker, Oracle database backend with more open source tools and technologies. The recommendation is to migrate Java applications to the JBOSS App/Web Server using AWS Elastic Beanstalk to produce containerized deployments for the different application tiers (Web/Presentation, App, Data) with auto scaling built into the solution. The deployment and management of applications leveraging a cloud-native approach using Elastic Beanstalk, makes auto scaling a seamless and salient feature of the deployed applications. Auto scaling triggers such as measuring CPU utilization on EC2 instances, etc., can be conveniently integrated into application deployment, by leveraging AWS CloudWatch and other AWS monitoring technologies and services. The proposed architecture also recommends replacing any message brokering technologies used by the applications with managed Amazon MQ service, and also replacing Oracle backend with managed RDS service. The migration from an on premise Oracle database backend to managed AWS RDS service is already a documented pattern or scenario in this playbook. We will showcase multiple variants of the 3-zone/tier architecture for the migration to the AWS platform in this technical playbook document, which will also include using Amazon SQS in lieu of Amazon MQ, as well as AWS Aurora in lieu of AWS Oracle RDS instance. This also serves to illustrate the importance of playbook scenarios that address full stack migration of an on premise application to the AWS platform.



**Figure 3: 3-zone/tier architecture using Elastic Beanstalk, Amazon MQ and Oracle RDS**

## 3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon MQ and Aurora DB

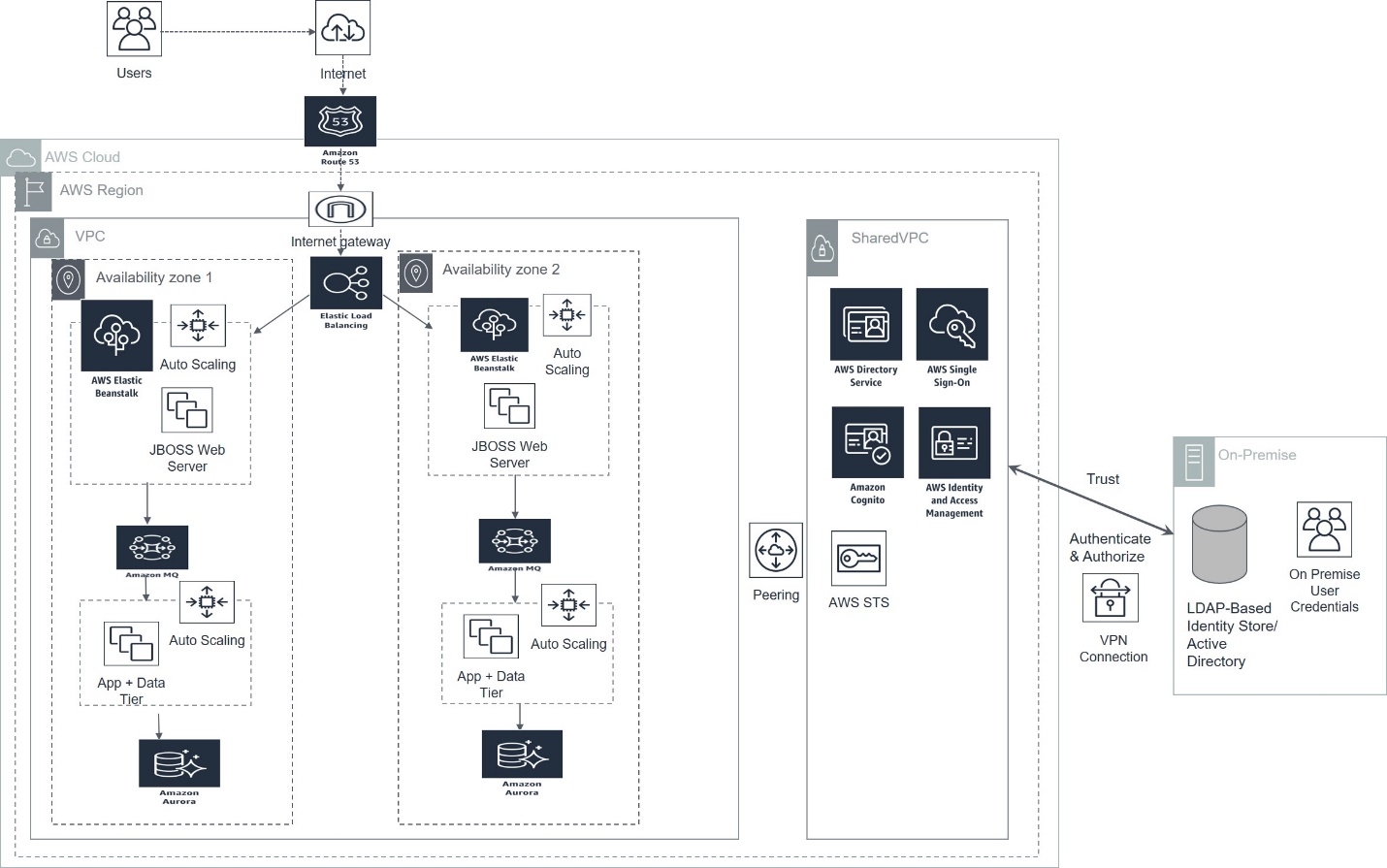
This is essentially the same solution as described previously except it replaces using AWS Oracle RDS managed service with Aurora DB managed service.

Below is a snapshot of the features of the Aurora DB managed service and is directly quoted from the AWS website describing the service:

*Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.*

*Amazon Aurora is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases. It provides the security, availability, and reliability of commercial databases at 1/10th the cost. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups.*

*Amazon Aurora features a distributed, fault-tolerant, self-healing storage system that auto-scales up to 64TB per database instance. It delivers high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones (AZs)*.



**Figure 4: 3-zone/tier architecture using Elastic Beanstalk, Amazon MQ, and Aurora DB**

## 3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon SQS and Oracle RDS

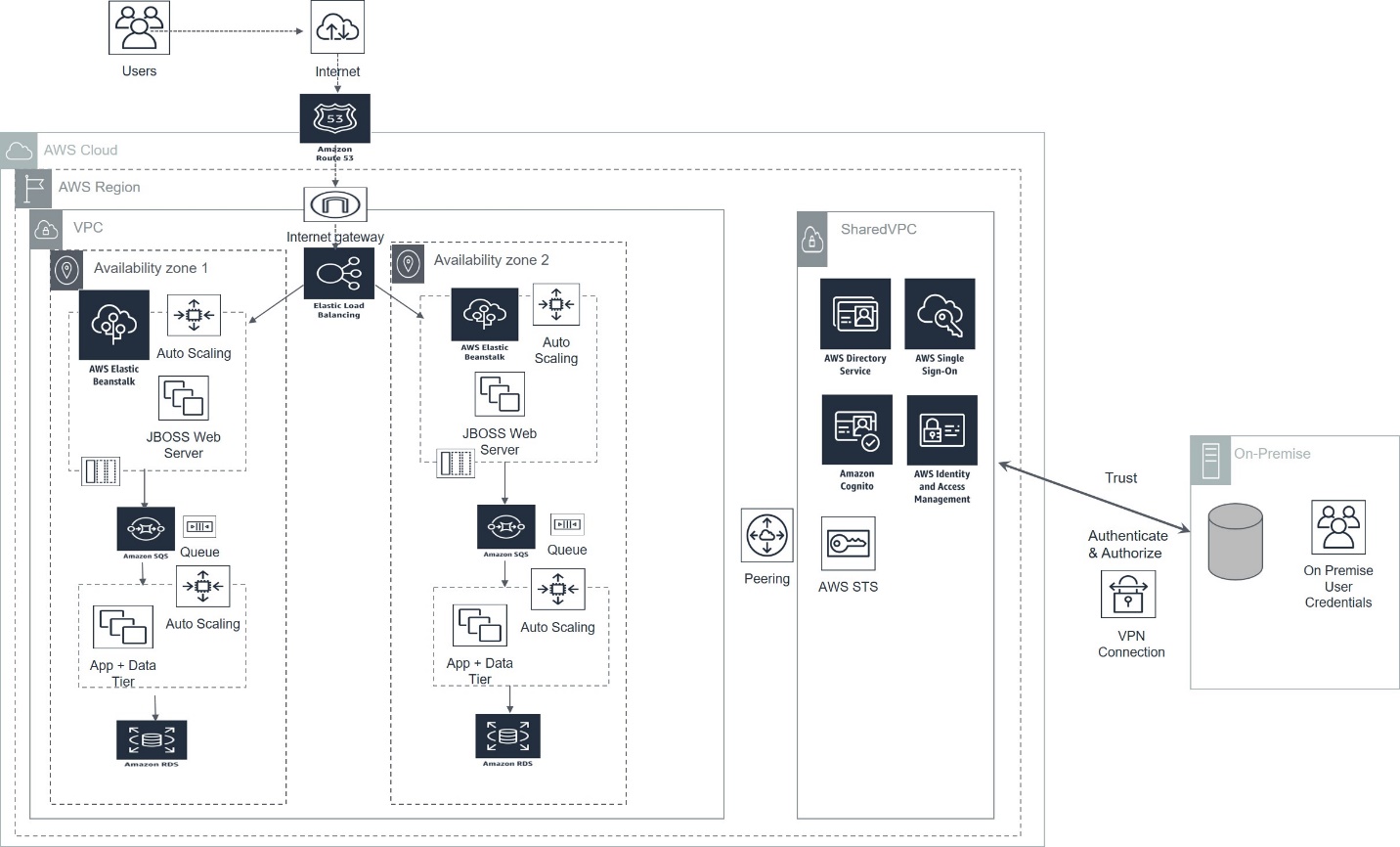
Provided next is another variant of the 3-zone/tier architecture with Elastic Beanstalk, Amazon SQS and using Oracle RDS. The difference between the 1st scenario described for the 3-zone/tier architecture and the one described here is we recommend completely replacing any message brokering technology with a fully cloud native messaging service that doesn’t require the installation or setup of a dedicated broker.

Provided below is a description of how AWS compares and contrasts Amazon MQ with Amazon SQS and is directly quoted from their website:

***How Is Amazon MQ Different from Amazon SQS or Amazon SNS?***

*Amazon MQ is a managed message broker service that provides compatibility with many popular message brokers. We recommend Amazon MQ for migrating applications from existing message brokers that rely on compatibility with APIs such as JMS or protocols such as AMQP, MQTT, OpenWire, and STOMP.*

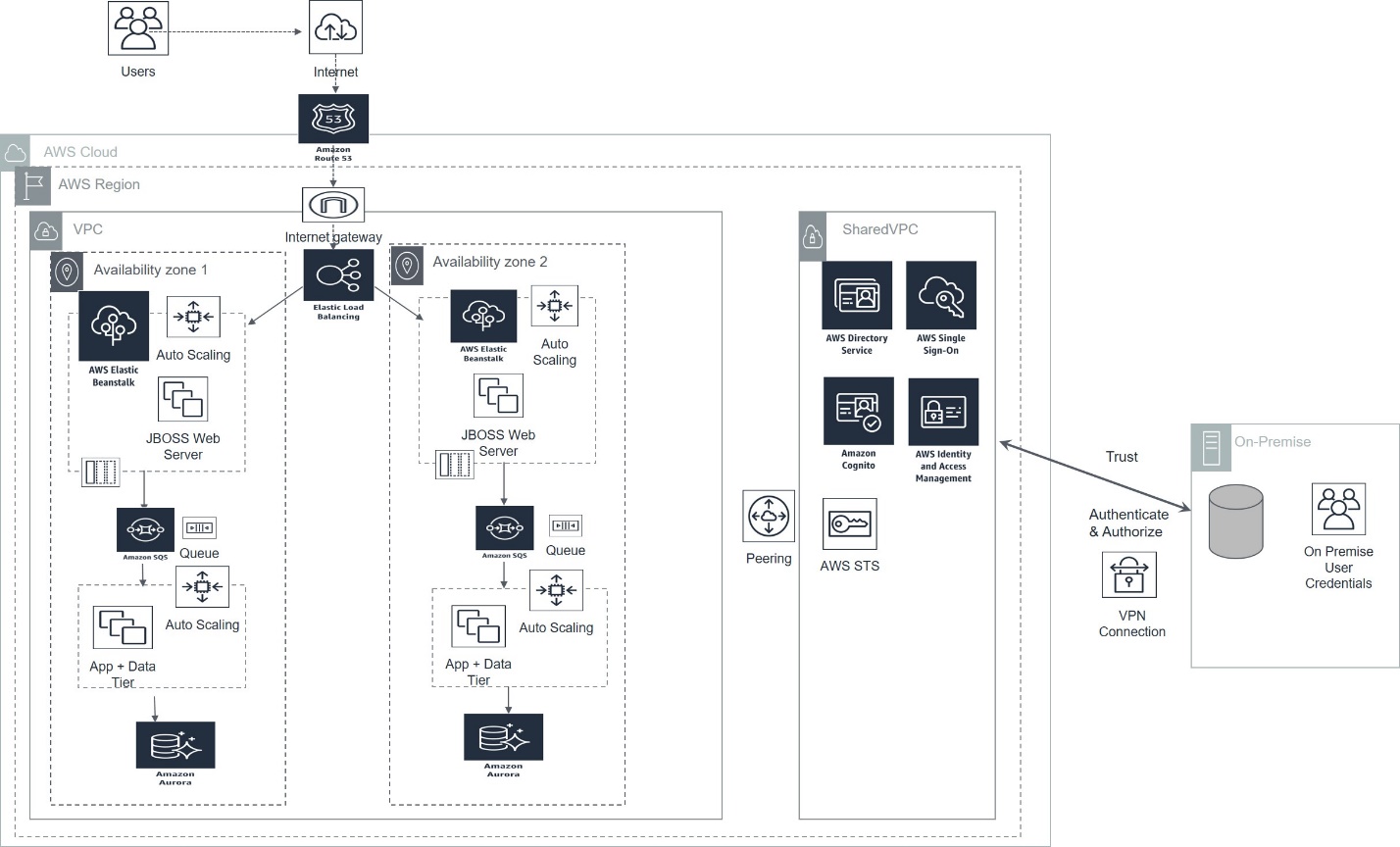
[*Amazon SQS*](https://aws.amazon.com/sqs/)*and*[*Amazon SNS*](https://aws.amazon.com/sns/)*are queue and topic services that are highly scalable, simple to use, and don't require you to set up message brokers. We recommend these services for new applications that can benefit from nearly unlimited scalability and simple APIs*.



**Figure 5: 3-zone/tier architecture using Elastic Beanstalk, Amazon SQS and Oracle RDS**

## 3-zone/tier architecture migrating to AWS Elastic Beanstalk, Amazon SQS and Aurora DB

This is the final version in the set of 4 patterns or design approaches, for the migration of a 3-zone/tier application architecture to the AWS platform. As is evident right from the caption/phrase for the approach, the recommendation is to use all cloud-first/cloud-native AWS services, to reengineer/build and deploy all tiers/parts of a 3-zone/tier application. The benefits of using SQS and Aurora DB have already been captured by the previous 2 playbook scenarios and won’t be reiterated again.



**Figure 6: 3-zone/tier architecture using Elastic Beanstalk, Amazon SQS and Aurora DB**

## AWS Migration Architecture for Pega BPM Tool

**<< Needs Diagram and Content>>**

## AWS Migration Architecture for Oracle BPEL Tool

**<< Needs Diagram and Content>>**

## AWS Migration Architecture for Oracle Data Warehouse (DWH) Solution

**<< Needs Diagram and Content>>**

## AWS Migration scenarios using NetApp

**<< Needs Diagram and Content>>**