TresVista
Catalyzing Your Impact

Product Development

Case Studies

Product Development

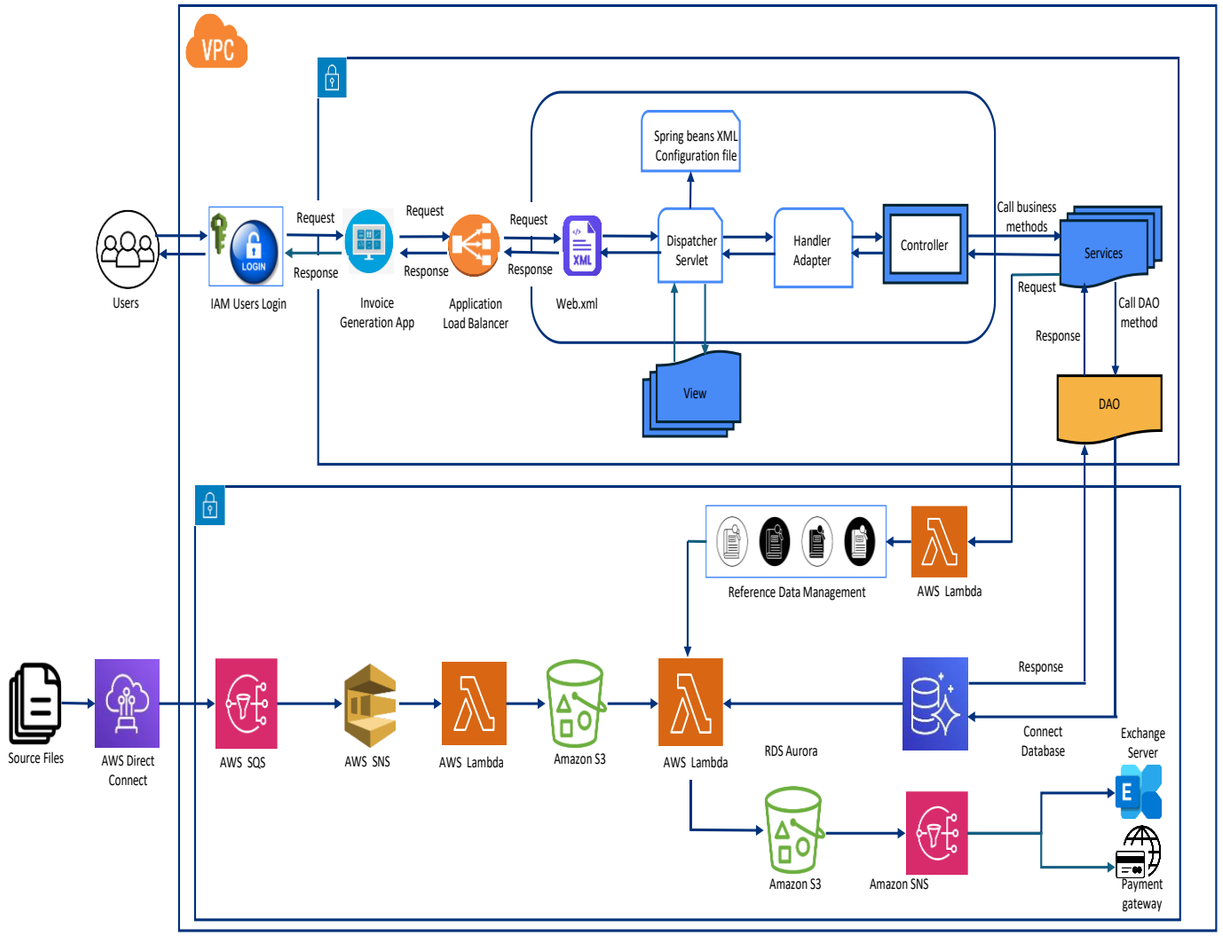
Automated Invoice Generation - Intelligent and Efficient Billing

Problem Statement

- Create high performance Billing application that consumes multiple services and reference data sources.
- Integrate the Billing application with Source Data, Reference systems and Payment gateway and Notification systems
- Support 100+ concurrent users, 2k+ daily active users, and 10k+ monthly active users with invoice generation time of less than 1 minute.

Solution Overview

- An invoice generation application built using Spring MVC architecture hosted on AWS solves the challenges of manual invoice generation
  - We load invoice raw data from various source files into the data lake using AWS Direct Connect. Upon file upload, necessary AWS services invoke the lambda function utilizing primary data from the data lake and additional reference data(e.g., GST rates, exchange rates) from Reference data management to generate standard invoices and store them in the data lake
  - Following invoice generation, notifications are sent to the downstream Payment application through the payment gateway, and to Exchange server.
  - IAM authenticated users access the web application to retrieve invoices, leveraging services that engage reference data management and database via DAO



Tools/Technology used: Spring MVC architecture, Java, AWS services, Payment gateway, Aurora, Python

Product Development

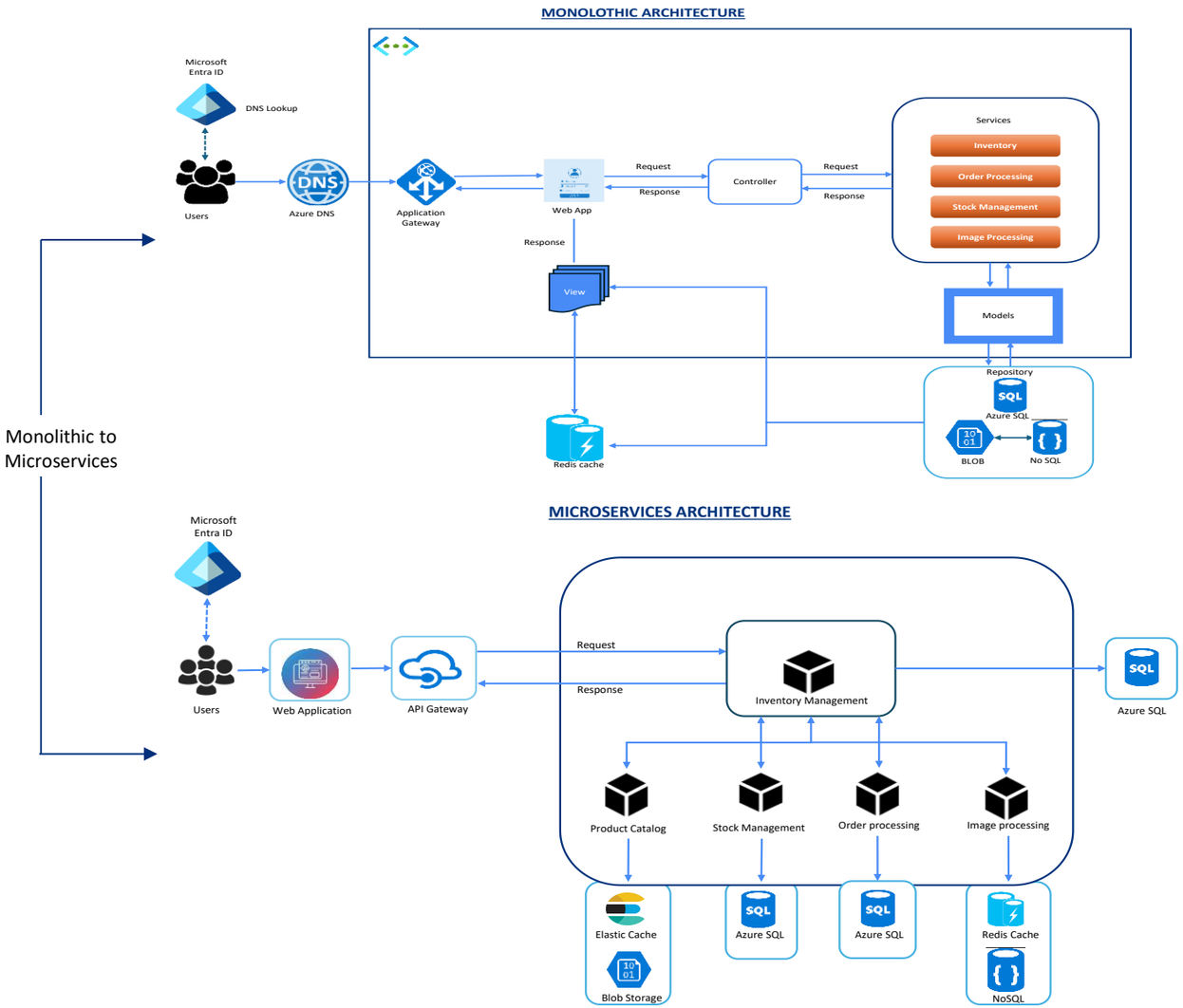
Inventory Management system – Monolithic and Microservices Architecture

Problem Statement

- With the augmentation of features in our client's Inventory Management Application, scaling became challenging as the application grew. The architecture's reliance on single technology stack limited flexibility. Additionally Monolithic applications lacked resilience and fault isolation, making it difficult to optimize resource allocation and handle failures efficiently.
- Furthermore, monolithic architectures are prone to cascading failures, where an issue in a single component can escalate and affect the entire system.

Solution Overview

- The inventory management system was redesigned for microservices architecture to enhance the scalability, technology diversity, resilience, maintainability, and support for continuous deployment
  - Entra ID user logs in to the web application, and the request is directed to the Load balancer and application gateway for routing to the master service, which contains metadata and is linked to other slave services
  - Based on the application requirements, services are designed and can be scaled independently because they are loosely coupled
  - The databases pertaining to the services are chosen based on the availability and consistency across the applications. For example: Redis cache with NoSQL is used for image processing



Tools/Technology used: Java, React, Microservices, Azure, Elastic cache, Azure SQL, Redis Cache, NOSQL SB

Product Development

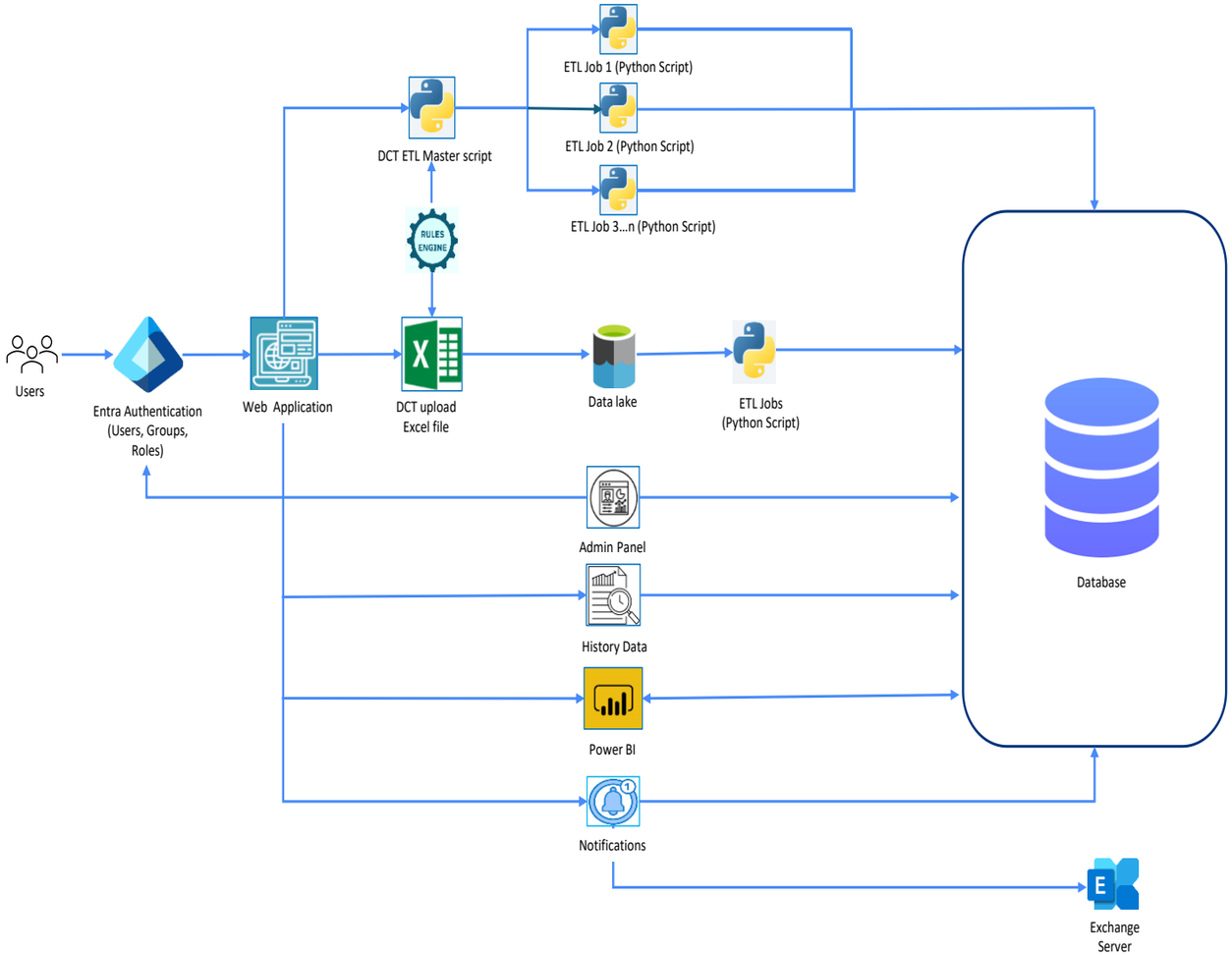
Three tier Web Based Application for Portfolio Management

Problem Statement

- Managing portfolio is cumbersome process which demand multiple features packaged in single solution. The lack of real-time data and advanced analytics further complicates decision-making, as portfolio managers struggle to respond promptly to market changes.
- Additionally, the complexity in customization and scalability hinders the ability to tailor solutions to specific portfolio needs.

Solution Overview

- A three tier Web application was built to provide a comprehensive and integrated solution that enhances efficiency, Security, Rule based validations and customization
  - Entra authenticated users logs in the web application and they are provided with the feature of uploading the portfolio details which is validated in the rule-based engine and the necessary screens based on the groups/roles are displayed. The data is transformed using the Python ETL scripts based on the business logics and are loaded into the Database which is further leveraged for Dashboarding.
  - Customized access controls and rule-based engine was implemented to enhance the efficiency based on the customization needs of the customer.



Tools/Technology used: Azure services, Web App, Python, Database, Power BI

salessupport@tresvista.com | www.tresvista.com