Proposal for

online cash-back and shopping awards organization.

TABLE OF CONTENTS

Objective:

Propose the design of the solution, which is cost effective, manageable, secure, scalable, high performance, efficient, highly available, fault tolerant and recoverable architecture for Personal Account.

Problems:

Current solution not supporting sudden spikes of the requests. When breakdown of the server's manual intervention is required. Current design is not scaling based on traffic, resulting in most of the resources not used and paying more.

Goals:

Analyze the online cash-back and shopping awards company requirements and provide solutions using various Amazon Web Services offerings, provide an architecture diagram and state all assumptions made during the design and explicitly state how Amazon Web Services will help overcome the current problems specific to the company.

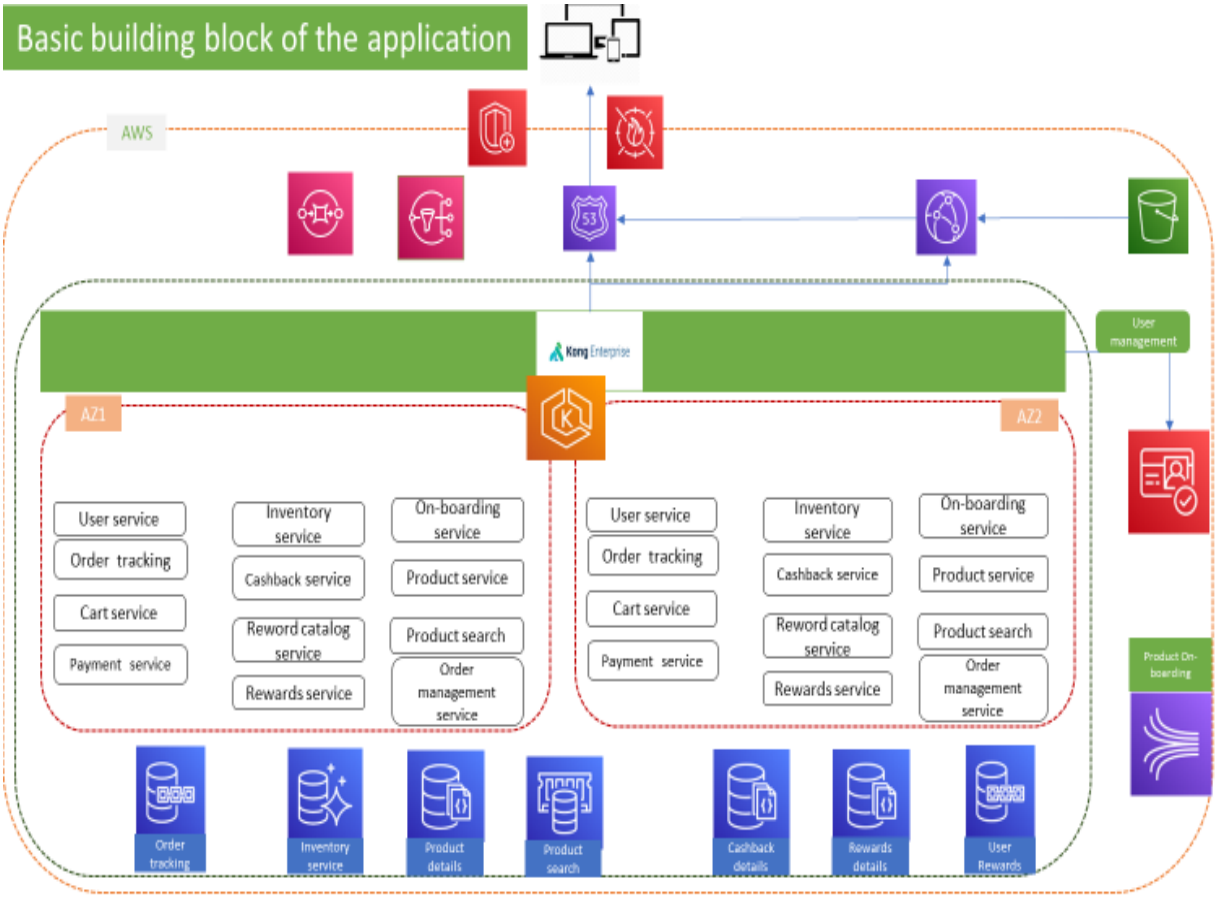
Budget:

One of the major benefits of our platform is that you only pay for what you use (pay-as you-go model). It is fair to say that at the early stages, most of the cost will come from compute and storage. Throughout the time, it will become clearer what the cost is during busy and idle periods.

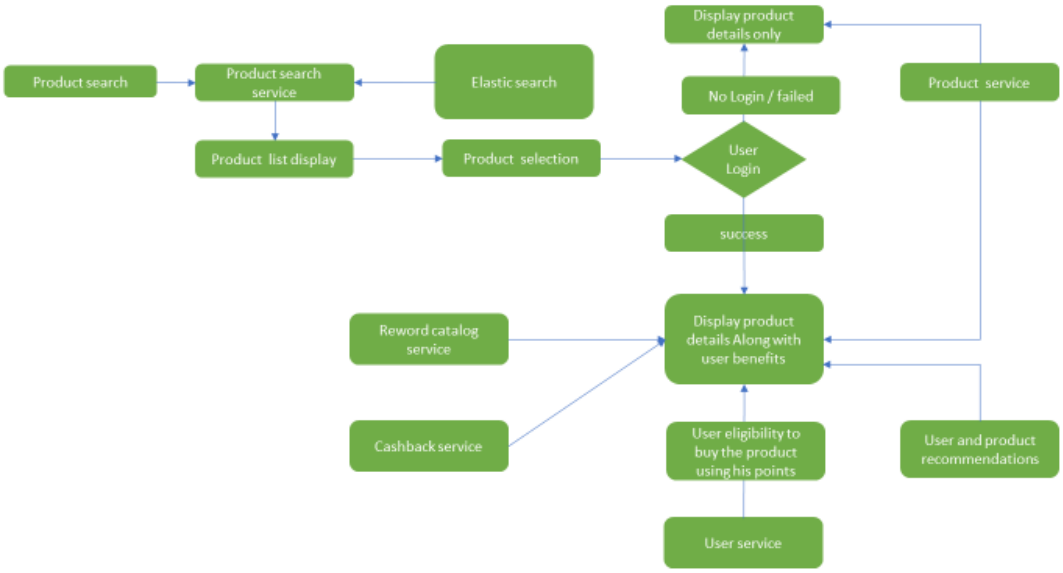
Basic assumptions:

- We will run the reward calculation job every day @12am and product can be returned within how many days can be configurable and based on the product will change. Product return policy will have more terms and conditions
- We are considering deployment will be done on multi availability zones using auto scaling (EKS inbuilt feature we will use)
- We are not considering returning the product flow. While return the order we will remove the reward points entry
- Reward redemption will not be charged, and no rewards points given back
- Recommendations are provided based on the user segment (income, martials status, and his past purchase), not part of design
- Product Recommendations are provided
- Spin new services for special promotions and subscriptions, this will be part of reward catalog service
- Shipping vender management not considered
- we use explicit pod anti-affinity to ensure replicas are distributed between AZs. Active-Active configuration that not required not to maintain the desire minimum or maximum count

design details:

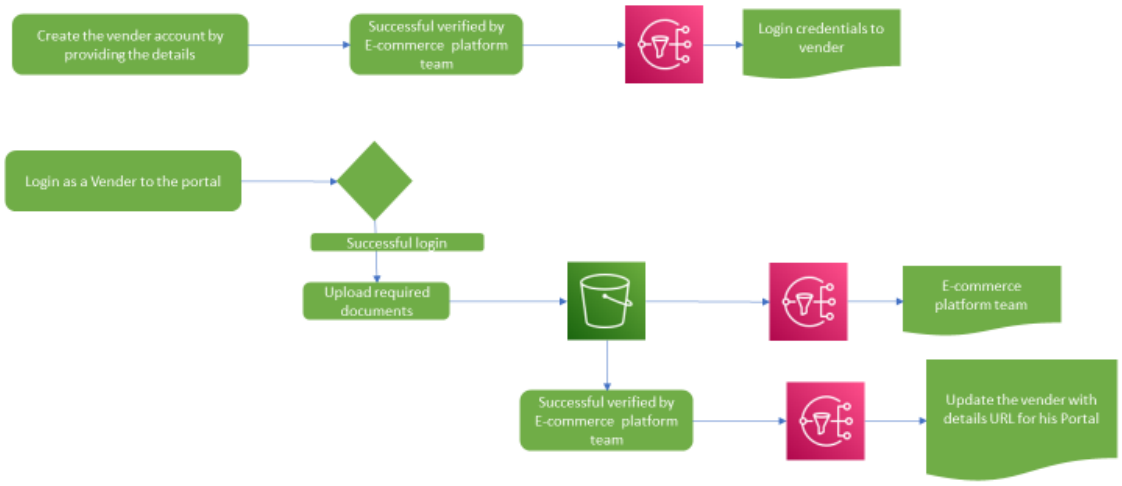


Product selection

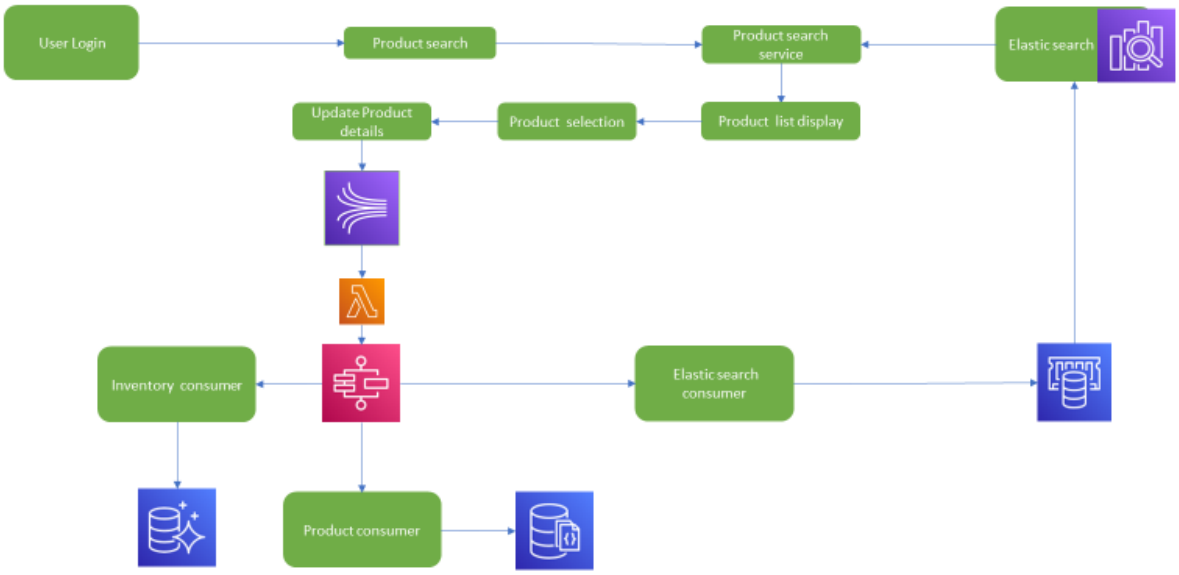


Order placement flow

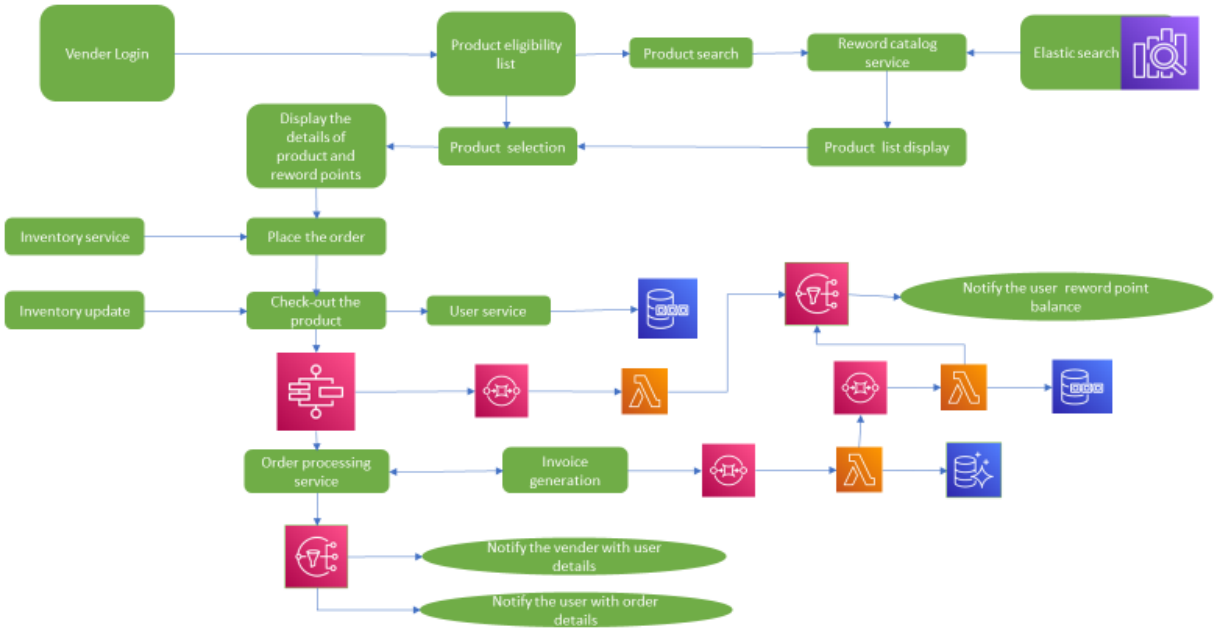
Vender on-boarding process



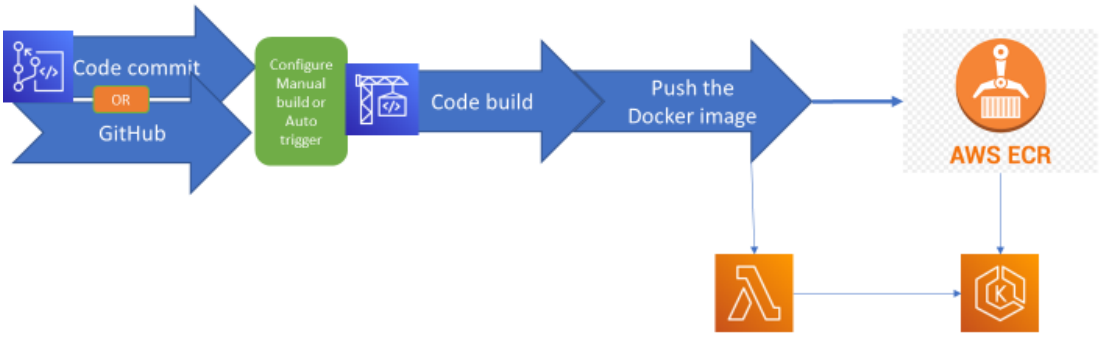
Vender product update



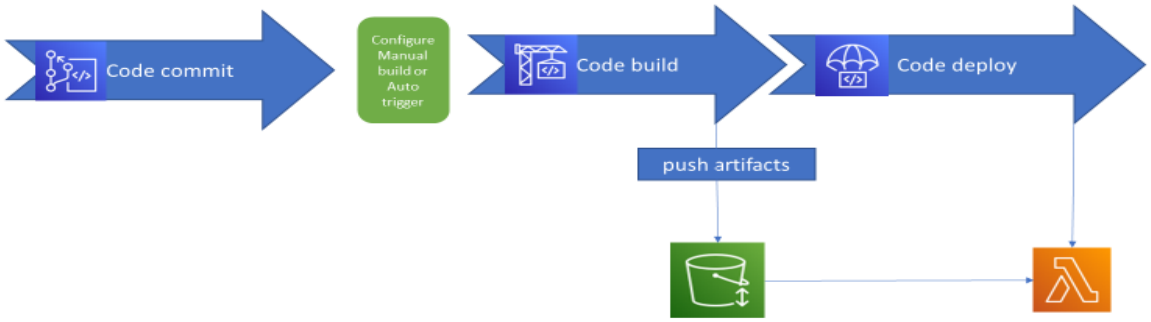
Rewords redemption



CI/CD pipeline for EKS pod development



CI/CD pipeline for Lambda



Analytics Point:

QLDB continuous journal stream:

You can integrate QLDB journal streams with the Amazon Kinesis streaming platform to process real-time journal data.

- Based on this we can recommend support accessories and services availability
- Hourly based discounts
- Which product/ brand/ vender sales is High @some point of time?

Redshift cluster:

We can export the Amazon Quantum Ledger Database to S3 periodically and do the analytics like below:

Customer segmentation based on his purchases we will do recommendations

- Range of products he is buying (economic brand recognition and brand image)
- Cost volumes he is buying (so that we can define his rewards and cashbacks)
- Variety of products for consumer types
- seasonal based product sales, brands so that we can recommend for inventory management
- area-based product sales, brands so that we can concentrate the vender's and Warehouse management effectively

Conclusion:

We focus on the key inputs for your business and deliver them with the right quality and in a timely fashion. Pay-as you-go model we can support the business effectively. Cost may change based reserve resources and on-demand resources type. We use analytics effectiveness to calculate customer frequency so that we reserve the resources it will reduce the cost.