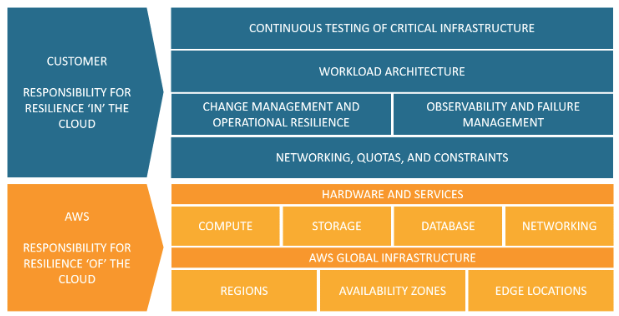
**SOP - Resilience Discussion and Best Practices for End-to-End Application Level**

1. Introduction

The purpose of this SOP is to guide consultants in conducting resilience discussions with customers at the end-to-end application level, specifically focusing on Amazon Elastic Kubernetes Service (EKS), Traefik API Gateway, and other components deployed across multiple availability zones (AZs) within an AWS region. The goal is to ensure that the application architecture is resilient, with a target Recovery Time Objective (RTO) and Recovery Point Objective (RPO) in the event of an AZ failure.

However in general Resiliency is a shared responsibility between AWS and the customer as depicted below :-

**

2. Overview

Resilience is a critical aspect of application design, ensuring that the application remains highly available and operational in the face of potential failures. In this discussion, we will focus on the following key components:

* Amazon EKS - Managed Kubernetes service for container orchestration.
* Traefik API Gateway and Ingress - Responsible for routing and load balancing traffic to the application services.
* MongoDB Databases - Running as statefulsets in Kubernetes, backed by Elastic Block Store (EBS) volumes across multiple AZs.
* EC2 Nodegroups and Worker Nodes - Providing compute resources for the application.

3. RTO & RPO Target

For the application's resilience strategy, we aim to achieve an RTO target of 5 minutes in case of an AZ failure and an RPO of 10 minutes. This means that in the event of an AZ failure, we aim to recover the application and restore its functionality within 5 minutes. Additionally, we aim to limit data loss to a maximum of 10 minutes.

4. Recovery Process

To achieve the target RTO and RPO, the following recovery processes are proposed for core components:

4.1 EKS Worker Nodes:

* Utilize multiple nodegroups spread across different AZs to distribute compute resources and ensure high availability.
* Implement a proactive node replacement strategy with the help of Karpenter, configured to push the replacement node to a different AZ than the failing node.

4.2 Traefik API Gateway Pods:

* Deploy Traefik API Gateway pods across multiple AZs to ensure redundancy and fault tolerance.
* Utilize Kubernetes DaemonSets/Deployments to automatically reschedule pods to healthy nodes in case of failures.

4.3 Microservices:

* Deploy Microservices as kubernetes deployments with required set of replicas across nodes in different AZ’s
* Use node/pod affinity, taints & tolerations in case if it needs specific workloads

4.4 MongoDB Databases:

* Deploy MongoDB databases as statefulsets, ensuring unique identity and persistent storage for each instance.
* Use EBS volumes to back the MongoDB databases, replicating data across multiple AZs to achieve data durability.
* Implement read and write concern strategies to ensure data consistency during failure scenarios.

5. High Availability & Load Balancing

To achieve high availability and load balancing, we will utilize the following mechanisms:

* Deploy application services across multiple replicas and distribute them across different AZs using Kubernetes Deployments.
* Implement Traefik Ingress Controllers with an Application Load Balancer (ALB) to route and distribute traffic across healthy replicas in different AZs.

6. Failover Testing

Regularly perform failover testing to ensure the resilience strategy is effective and meets the RTO and RPO targets. Conduct testing during maintenance windows to minimize customer impact.

7. Monitoring & Alerting

Implement comprehensive monitoring and alerting for all components. Monitor key performance indicators, application health, and resource utilization. Set up alarms to detect any potential failures or degraded performance.

8. Documentation & Communication

Document the resilience strategy and architecture decisions. Communicate the strategy to the customer, ensuring mutual understanding and alignment with the proposed solution.

9. Conclusion

Resilience discussions are crucial for ensuring the application's high availability and operational stability. By adhering to these best practices and targets for RTO and RPO, we can create a robust and resilient end-to-end application architecture for the customer.

10. Revision History

Maintain a revision history for this SOP to track updates, improvements, or changes over time.

By following this SOP and discussing resilience strategies with customers, we can deliver highly available and reliable applications on AWS EKS and other services, meeting customer expectations and maintaining service uptime even during AZ failures.