**SOP for Resilience - Basik Marketing - End-to-End Application Level**

1. **Purpose**

The purpose of this Standard Operating Procedure (SOP) is to **outline best practices** for resilience at the end-to-end application level. This SOP will cover topics related to Recovery Time Objective (RTO), Recovery Point Objective (RPO), and the recovery process for core components in a customer architecture that includes AWS EKS, Traefik API Gateway and Ingress, Application Load Balancers, EC2 Nodegroups, and Worker Nodes deployed across multiple availability zones (AZs) within a region.

2. **Introduction**

Resilience refers to the ability of a system to continue functioning even in the face of failures or disruptions. In the context of an end-to-end application architecture, it is crucial to design for resilience to ensure high availability and minimize downtime in the event of failures.

3. **Recovery Time Objective (RTO) & Recovery Point Objective (RPO)**

* Recovery Time Objective (RTO): The RTO is the targeted time within which a system or application must be restored after a failure occurs. For our customer architecture, the RTO target is set to **5 minutes** in case of an availability zone (AZ) failure.
* Recovery Point Objective (RPO): The RPO is the targeted maximum data loss that is acceptable in case of a failure. For our customer architecture, the RPO target is set to **10 minutes** with the help of MongoDB databases (stetefulsets) backed by Amazon Elastic Block Store (EBS) volumes across multiple AZs.

4. **Recovery Process for Core Components**

4.1 **EKS Control Plane - Managed by AWS with external ETCD DB**

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

Amazon EKS runs and scales the Kubernetes control plane across multiple AWS Availability Zones to ensure high availability. Amazon EKS automatically scales control plane instances based on load, detects and replaces unhealthy control plane instances, and automatically patches the control plane. After you initiate a version update, Amazon EKS updates your control plane for you, maintaining high availability of the control plane during the update.

This control plane consists of at least two API server instances and three etcd instances that run across three Availability Zones within an AWS Region. Amazon EKS:

Actively monitors the load on control plane instances and automatically scales them to ensure high performance.

Automatically detects and replaces unhealthy control plane instances, restarting them across the Availability Zones within the AWS Region as needed.

Leverages the architecture of AWS Regions in order to maintain high availability. Because of this, Amazon EKS is able to offer below service levels and the credits in case of outage for the EKS Control Plane and the ETCD:-

Service Credits are calculated as a percentage of the charges paid by you for the EKS Cluster(s) that did not meet the Monthly Uptime Percentage commitment in a billing cycle in accordance with the schedule below.

| **Monthly Uptime Percentage** | **Service Credit Percentage** |
| --- | --- |
| Less than 99.95% but greater than or equal to 99.0% | 10% |
| Less than 99.0% but greater than or equal to 95.0% | 25% |
| Less than 95.0% | 100% |

4.1 **EKS Worker Nodes**

* EKS worker nodes are deployed across multiple AZs using EC2 Nodegroups to achieve high availability.
* In the event of an AZ failure, EKS will automatically handle the scheduling of pods to healthy nodes in other AZs.
* Kubernetes Cluster Autoscaler and Karpenter are configured to push alternate nodes to different AZs to maintain pod availability.

4.2 **MongoDB Databases - kubernetes STS with helm charts**

* MongoDB databases are deployed as “StatefulSets” in Kubernetes with persistent volumes backed by EBS volumes across multiple AZs.
* MongoDB deployment was in Master/Master mode to ensure HA.
* The RPO of 10 minutes is achieved by ensuring frequent replication of data to EBS volumes in other AZs.
* In the event of a failure, MongoDB pods are rescheduled (with same name- feature of the statefulset) to healthy nodes in other AZs, and the data is recovered from the replicated EBS volumes in respective AZs.

4.3 **Microservices - kubernetes Deployment with replicasets**

* Microservices are deployed as “Deployment” in Kubernetes without any persistent volumes on the pods
* The RPO of 10 minutes (with back-end mongodb service connectivity) is achieved by ensuring re-deployment of the microservices pods in case of failures with the help of kubernetes native features
* In the event of AZ failure, Microservices pods are rescheduled to healthy nodes in other AZs.

4.4 **Traefik API Gateway and Ingress**

* Traefik API Gateway and Ingress DaemonSet with pods are deployed across multiple AZs to ensure availability.
* In the event of an AZ failure, Kubernetes and Traefik automatically handle the routing of traffic to healthy pods in other AZs.

5. **Conclusion**

By following the best practices and resilience strategies outlined in this document, the end-to-end application architecture, comprising AWS EKS, Traefik API Gateway and Ingress, Application Load Balancers, EC2 Nodegroups, and Worker Nodes, will be equipped to handle failures effectively, ensuring minimal downtime and maintaining high availability for their applications.