# Confluent Kafka well Architectured stories

## Creating single node pool :

The current setup involves individual node pools for each Kafka service, resulting in suboptimal resource utilization. The proposal is to consolidate these separate node pools into a single, shared node pool. This consolidation enhances resource efficiency and requires careful configuration of resource requests and limits for pods to ensure optimal allocation, resource isolation, and overall cluster stability.

**Definition of Done:**

1. Successfully consolidate individual node pools into a single shared node pool.
2. Ensure optimal resource utilization through appropriate configuration of resource requests and limits.
3. Rigorously test resource isolation and cluster stability under various scenarios.

* Right sizing the nodes:

The existing architecture utilizes large node sizes for each Kafka service. To achieve cost savings and improve resource utilization, the plan is to merge multiple Kafka services into a single node. This consolidation reduces the number of nodes in use, leading to approximately 40% cost reduction. Careful consideration is given to avoiding small capacity instances, as they may consume CPU and memory for OS and Kubelets without significant benefits.

**Definition of Done:**

1. Merge multiple Kafka services into a single, appropriately sized node.
2. Achieve approximately 40% cost reduction by reducing the number of nodes in use.
3. Optimize resource configurations for CPU, memory, and other parameters.
4. Thoroughly test and validate the stability of the cluster with the new node configuration.

* Auto Scaling:

Integrate Cluster AutoScaler or Karpenter to enable dynamic management of node groups based on resource usage. This automation ensures that the cluster scales up or down according to demand, optimizing resource allocation and minimizing costs.

**Definition of Done:**

1. Successfully integrate Cluster AutoScaler or Karpenter.
2. Demonstrate dynamic management of node groups based on resource usage.
3. Thoroughly test and validate automated scaling actions under varying workloads.

* Pod Balancing:

Implement Kubernetes Descheduler to optimize the distribution of pods across nodes. This tool prevents underutilization of resources by intelligently redistributing pods, improving overall cluster efficiency.

**Definition of Done:**

1. Implement Kubernetes Descheduler for optimal pod distribution across nodes.
2. Verify tools' effectiveness in preventing underutilization and improving overall cluster efficiency.
3. Conduct thorough testing to ensure balanced pod distribution under different scenarios.

* Making services High Available:

Address the current limitation in Zookeeper's high availability by increasing the number of Zookeeper nodes from 3 to 5. This adjustment enhances the quorum's resilience, allowing the cluster to withstand the failure of up to two Zookeeper nodes while maintaining availability..

**Definition of Done:**

1. Increase the number of Zookeeper nodes from 3 to 5.
2. Demonstrate enhanced quorum resilience, allowing the cluster to withstand the failure of up to two Zookeeper nodes.
3. Rigorously test and validate the resilience and availability of the Zookeeper quorum.

## Remove Rest proxy:

Propose the decommissioning of the Rest Proxy service in the Stockholm cluster due to its underutilization. If there is a future demand for this service, it can be reintroduced. However, for the present scenario, stopping support for Rest Proxy results in resource optimization and streamlining the cluster.

**Definition of Done:**

1. Successfully decommission Rest Proxy service from the Stockholm cluster.
2. Notify relevant stakeholders about the cessation of Rest Proxy support.
3. Implement monitoring to ensure no disruption occurs due to the removal of Rest Proxy.

## Running Control Center with managed only mode:

Leverage the existing data and metrics streaming to Datadog by running Control Center in managed-only mode. This mode significantly reduces the resources consumed by Control Center, leading to improved efficiency and cost savings. All relevant data and metrics will continue to be available through Datadog, ensuring continued monitoring capabilities.

**Definition of Done:**

1. Configure Control Center to operate in managed-only mode.
2. Demonstrate significant reduction in resource consumption.
3. Ensure continued availability of relevant data and metrics through Datadog.

## Use Ingress as Load Balancer in daemon sets.

Implement Ingress as a Load Balancer within daemon sets to efficiently manage and distribute incoming traffic across the application's components. This approach will enhance scalability, simplify routing, and improve resource utilization.

**Definition of Done:**

1. Implement Ingress as a Load Balancer within daemon sets.
2. Enhance scalability, simplify routing, and improve overall resource utilization.

## Use Tiered storage for topics with large data.

Optimize storage management by implementing a tiered storage approach specifically designed for topics dealing with large volumes of data. This involves categorizing data based on access frequency, ensuring cost-effectiveness, and improving overall system performance.

**Definition of Done:**

1. Optimize storage management with tiered storage for large data topics.
2. Categorize data based on access frequency to ensure cost-effectiveness and improved system performance.

## Reducing EBS volume: Rohit did a demo and shared the document.

Following Rohit's demo and shared documentation, the initiative involves carefully reducing Elastic Block Store (EBS) volumes. This optimization aims to achieve resource efficiency, cost savings, and better alignment with actual storage needs.

**Definition of Done:**

1. Successfully reduce Elastic Block Store (EBS) volumes as demonstrated by Rohit.
2. Achieve resource efficiency, cost savings, and alignment with actual storage needs.

## Increase the payload size to 8 MB.

Enhance system capabilities by increasing the permissible payload size to 8 MB. This adjustment facilitates the smooth exchange of larger sets of data between system components, meeting the growing demands of data-intensive processes.

**Definition of Done:**

1. Enhance system capabilities by increasing the permissible payload size to 8 MB.
2. Ensure smooth exchange of larger sets of data between system components.

## Have proper naming convention for Datadog dashboard.

Establish a standardized and meaningful naming convention for Datadog dashboards to facilitate easy identification, navigation, and management of monitoring and analytics visualizations. This naming convention will improve collaboration and maintainability.

**Definition of Done:**

1. Establish a standardized and meaningful naming convention for Datadog dashboards.
2. Facilitate easy identification, navigation, and management of monitoring and analytics visualizations.

## Automate cert generation process or increase the expiry date to like 2 years till Stockholm platform is decommission.

Streamline the certificate generation process or extend the expiration date to a suitable duration, such as 2 years, to ensure smooth operation until the Stockholm platform is decommissioned. Automation will enhance efficiency and reduce manual intervention in certificate management.

**Definition of Done:**

1. Streamline the certificate generation process or extend the expiration date to 2 years.
2. Ensure smooth operation until the Stockholm platform is decommissioned.
3. Implement automation for efficient certificate management.

## Automate root ca distribution.

Implement an automated process for distributing the root Certificate Authority (CA) to enhance security and streamline certificate validation across the system. This automation ensures consistent and secure distribution of the root CA, minimizing the risk of errors.

**Definition of Done:**

1. Implement an automated process for distributing the root Certificate Authority (CA).
2. Enhance security and streamline certificate validation across the system.

## Create Run book different possible scenario.

Develop a comprehensive runbook detailing various operational scenarios, including common issues, troubleshooting steps, and resolution procedures. This runbook serves as a valuable reference for support teams, enabling them to respond effectively to different situations.

**Definition of Done:**

1. Develop a comprehensive runbook detailing various operational scenarios.
2. Include common issues, troubleshooting steps, and resolution procedures.
3. Serve as a valuable reference for support teams to respond effectively to different situations.

## Leverage confluent tool to get all possible logs which confluent support team may require.

Utilize Confluent tools to capture and provide all necessary logs that the Confluent support team may require for efficient troubleshooting and issue resolution. This proactive approach ensures timely and accurate support, minimizing system downtime.

**Definition of Done:**

1. Utilize Confluent tools to capture and provide all necessary logs for efficient troubleshooting.
2. Ensure proactive support with timely and accurate information for issue resolution**.**

## Add performance testing using Jmeter or some inbuilt confluent tools.

Integrate performance testing into the system using tools like JMeter or built-in Confluent tools. This initiative aims to assess and optimize the system's scalability, responsiveness, and overall performance under varying load conditions.

**Definition of Done:**

1. Integrate performance testing into the system using JMeter or built-in Confluent tools.
2. Assess and optimize system scalability, responsiveness, and performance under varying load conditions.

## Enable Rack awareness while replicating data in AZs.

Implement rack awareness strategies to enhance data replication across Availability Zones (AZs). This approach ensures data resilience and reliability in the event of AZ failures, promoting high availability and fault tolerance in the system architecture.

**Definition of Done:**

1. Implement rack awareness strategies to enhance data replication across Availability Zones (AZs).
2. Ensure data resilience, reliability, high availability, and fault tolerance in the system architecture.