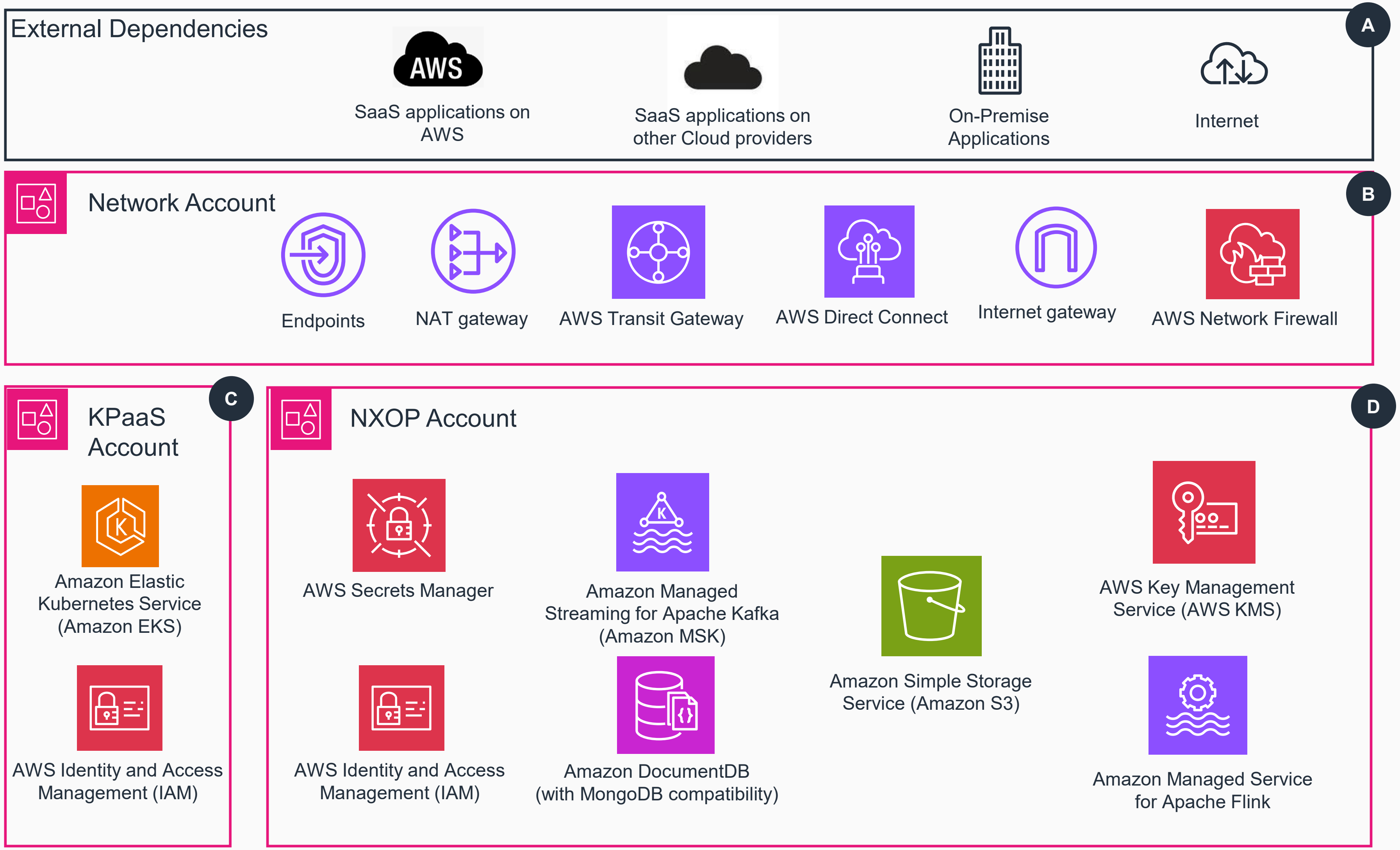
NXOP MVP Design Doc

System Context

This diagram shows the overall system context calling out the core/critical components that together make up the NXOP platform


External Dependencies
A
AWS
SaaS applications on AWS
SaaS applications on other Cloud providers
On-Premise Applications
Internet
Network Account
B
Endpoints
NAT gateway
AWS Transit Gateway
AWS Direct Connect
Internet gateway
AWS Network Firewall
KPaaS Account
C
Amazon Elastic Kubernetes Service (Amazon EKS)
AWS Identity and Access Management (IAM)
NXOP Account
D
AWS Secrets Manager
Amazon Managed Streaming for Apache Kafka (Amazon MSK)
Amazon Simple Storage Service (Amazon S3)
AWS Key Management Service (AWS KMS)
AWS Identity and Access Management (IAM)
Amazon DocumentDB (with MongoDB compatibility)
Amazon Managed Service for Apache Flink

A External systems and connectivity that NXOP integrates with including SaaS applications on AWS and other clouds, AA on-premise legacy systems, and internet connectivity for monitoring and external APIs.

B Centralized networking infrastructure providing secure connectivity for NXOP including Transit Gateway for cross-account routing, NAT/Internet gateways for external access, Direct Connect for on-premise connectivity, Network Firewall for security inspection, and VPC endpoints for private AWS service access.

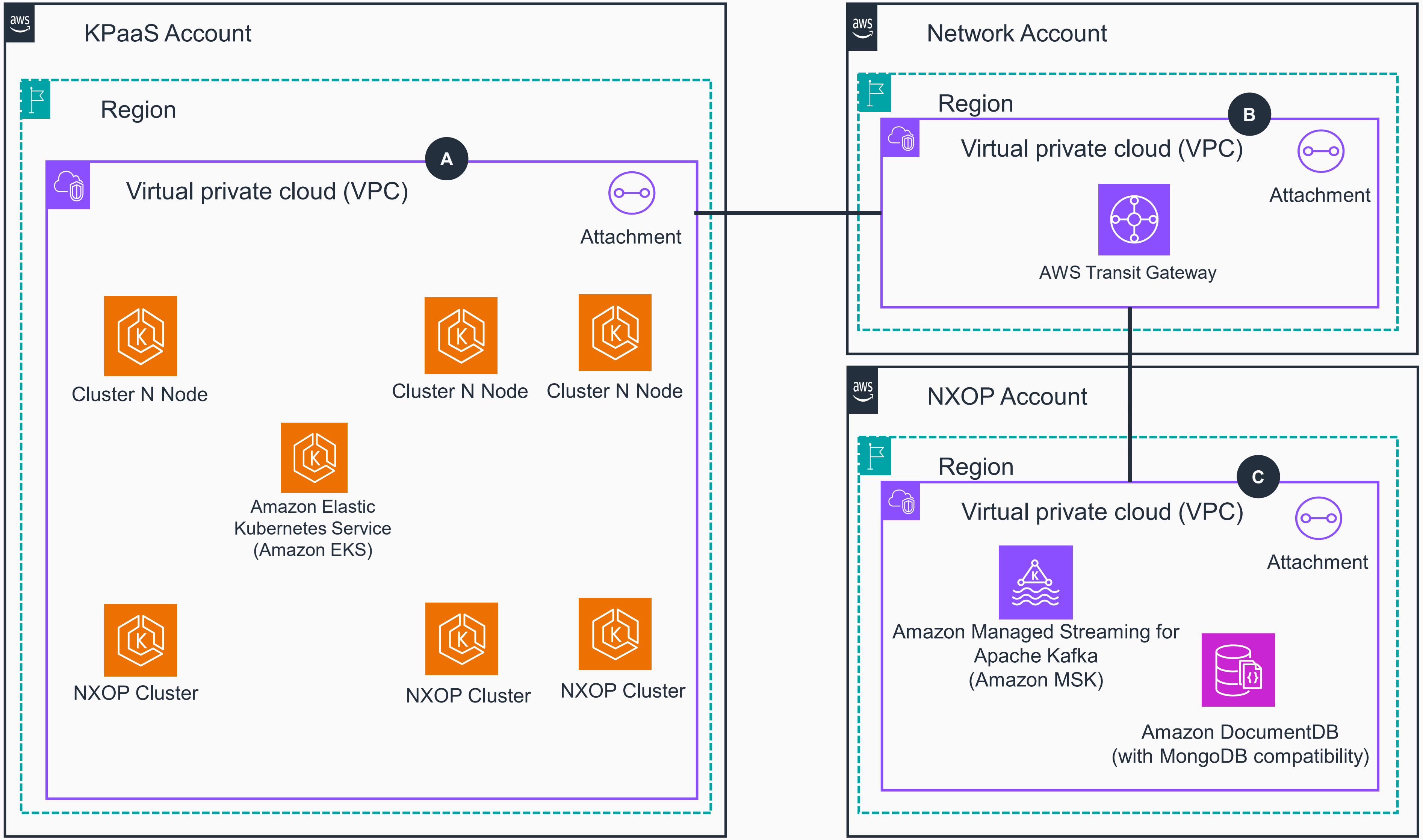
C Kubernetes platform services hosting NXOP applications including Amazon EKS clusters running RabbitMQ consumers, flight data validators, and transformers/processors, plus IAM services managing pod identities and cross-account access permissions.

D Data and security services supporting NXOP operations including MSK for flight data streaming, Secrets Manager for FlightKeys credentials, KMS for encryption, IAM for cross-account access control, S3 and DocumentDB for data persistence.

Reviewed for technical accuracy, 2025

EKS Architecture – Cluster in Separate/Shared Account

This diagram essentially is shows the setup where the Amazon EKS clusters reside in separate and possible shared KPaaS account and the services required for the NXOP platform to function reside in the NXOP workload account.



KPaaS Account
Region
Virtual private cloud (VPC)
A
Attachment
Cluster N Node
Cluster N Node
Cluster N Node
Amazon Elastic Kubernetes Service (Amazon EKS)
NXOP Cluster
NXOP Cluster
NXOP Cluster
Network Account
Region
Virtual private cloud (VPC)
B
Attachment
AWS Transit Gateway
NXOP Account
Region
Virtual private cloud (VPC)
C
Attachment
Amazon Managed Streaming for Apache Kafka (Amazon MSK)
Amazon DocumentDB (with MongoDB compatibility)

A **Amazon Virtual Private Cloud** (Amazon VPC) hosts multiple **Amazon Elastic Kubernetes Service** (Amazon EKS) clusters in the shared KPaaS account.

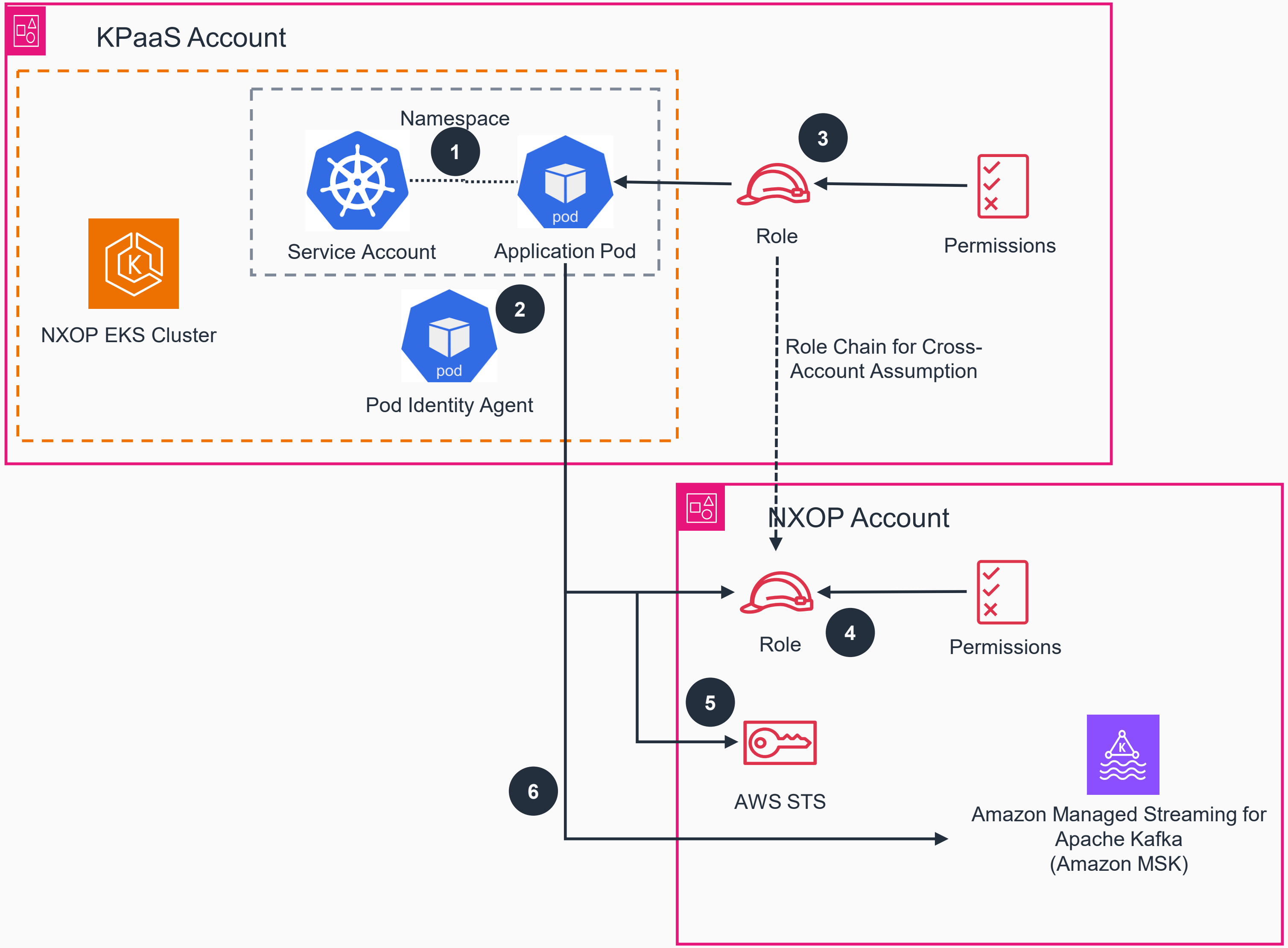
B The shared Network account hosts the **AWS Transit Gateway** with attachments to all other VPCs to enable across-account network connectivity.

C The NXOP Account hosts all the necessary services for the NXOP workload to operate like **Amazon DocumentDB** and **Amazon Managed Streaming for Apache Kafka** and many others.

Reviewed for technical accuracy, 2025

Security – Pod Identity – Role Chaining

This diagram shows the Pod Identity flow for cross-account assumption of roles



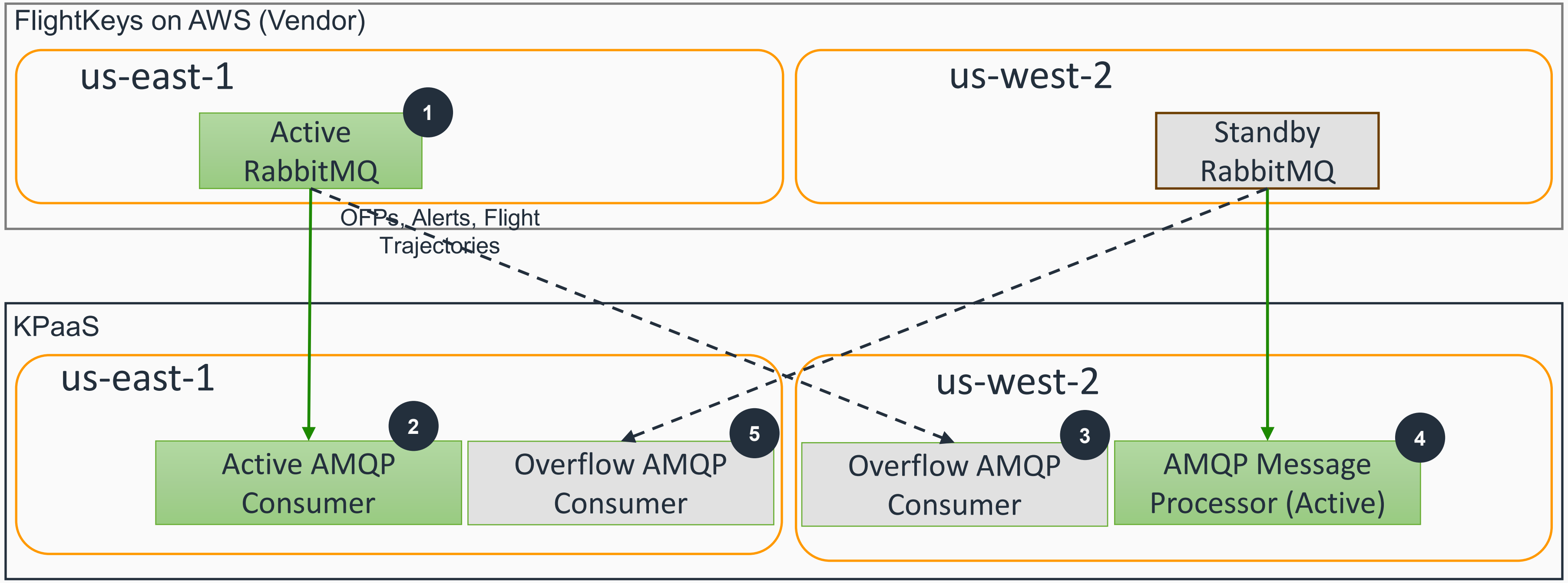
1. NXOP Application pod starts in EKS cluster with assigned service account for MSK access
2. EKS Pod Identity agent exchanges service account token for AWS credentials
3. Pod Identity assumes KPaaS account IAM role with permissions for cross-account access
4. Final role in workload account with specific MSK topic permissions, and cross-account trust policy for KPaaS IAM Role.
5. STS provides temporary credentials for workload account resource access
6. NXOP application accesses MSK topics using temporary credentials for consuming/producing messages

aws

Reviewed for technical accuracy, 2025

Overflow RabbitMQ Consumer Pattern

This shows a comprehensive Health aware overflow scaling pattern, where the overflow consumer scales out depending on a health criteria.



1 FlightKeys 5D engine publishes flight optimization data (OFPs, alerts, trajectories) to RabbitMQ queue in us-east-1 (Active Region)

2 Active AMQP Consumer consuming FlightKeys messages, validating schema, and transforming for downstream systems integrated to consume from the RabbitMQ queue in us-east-1

3 A similar deployment of the same application exists in us-west-2 but scaled down to Zero. The deployment scales up the number of Pods depending on a comprehensive health aware policy.

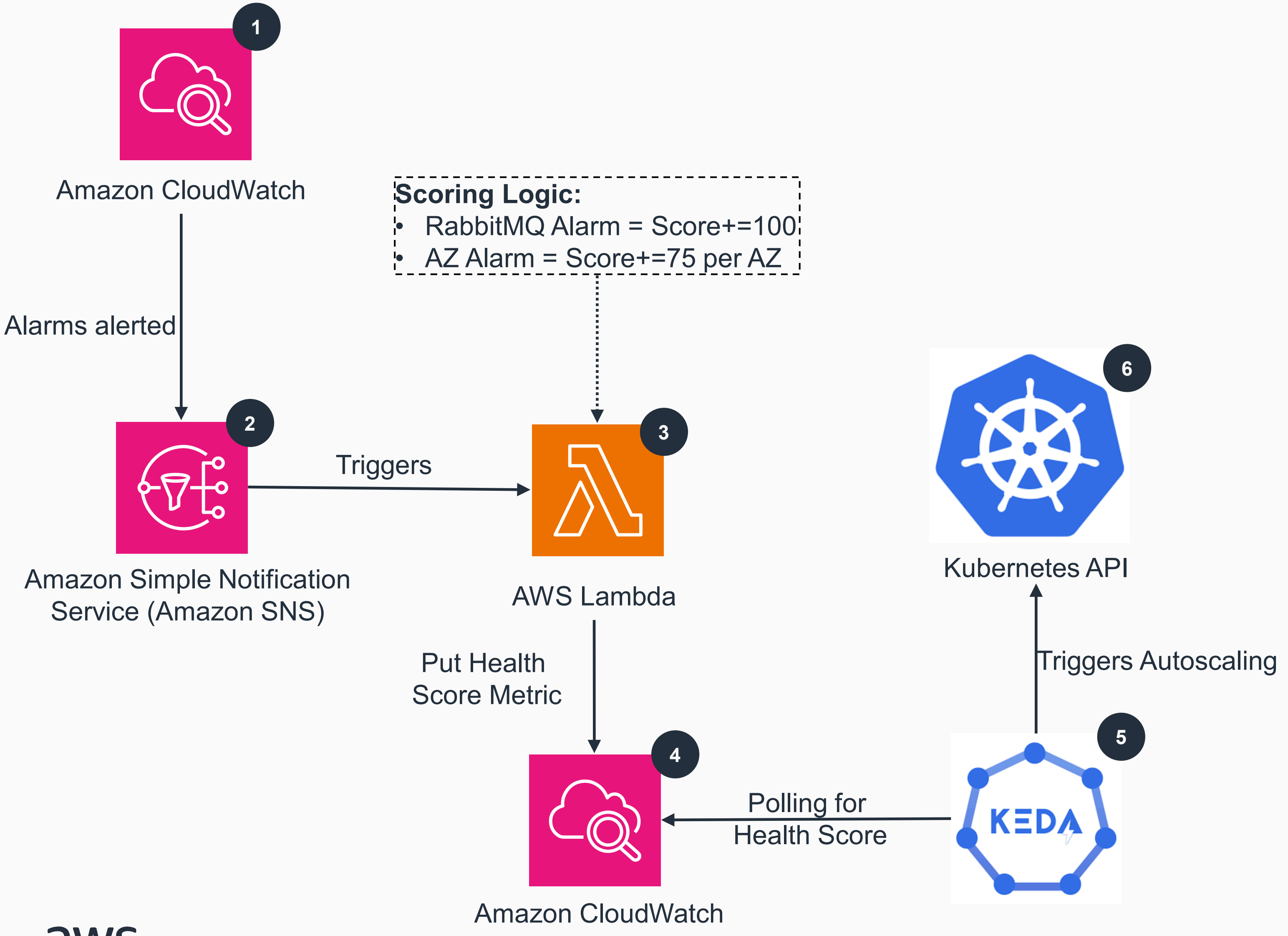
4 Flight Keys RabbitMQ in us-west-2 is in standby mode, with no messages being produced actively until FlightKeys fails over.

5 A similar deployment of the same application exists in us-east-1 but scaled down to Zero. The deployment scales up the number of Pods depending on a comprehensive health aware policy.

Reviewed for technical accuracy, 2025

Overflow Consumer Application Scaling – Alarms to Scaling behavior

This shows a flow diagram of how to connect the CloudWatch Alarms to drive autoscaling of the overflow consumers



1 **Amazon CloudWatch** alarms are configured for publishing alerts to SNS Topics when the alarm is in **ALARM** state.

2 **Amazon Simple Notification Service (Amazon SNS)** topics are configured with permissions route the messages to subscriptions, and also support filters as needed.

3 **AWS Lambda** function is triggered and receives the SNS Topic messages and calculate the comprehensive Health score which is then also published to **Amazon CloudWatch** as a custom metric.

4 **Amazon CloudWatch** allows to configure alerting on the custom metric along with all the other alarms.

5 A KEDA Scaling configuration on the overflow consumer deployment polls **Amazon CloudWatch** in configurable interval for the Health score and performs a scaling math based on desired vs target values to determine the scaling behavior.

6 KEDA uses the derived scaling behavior information to interact with Kubernetes API to scale up/down the overflow consumer application Pods as needed.

Reviewed for technical accuracy, 2025

KEDA Scaling Configuration Example

This shows an example of KEDA Scaled Object configuration that uses the Health Score from Amazon CloudWatch

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: overflow-consumer-optimal
spec:
  scaleTargetRef:
    name: overflow-amqp-consumer
  minReplicaCount: 0
  maxReplicaCount: 10
  triggers:
  - type: aws-cloudwatch
    metadata:
      awsRegion: us-east-1
      namespace: Custom/FlightPipeline
      metricName: HealthScore
      dimensionName: Flow
      dimensionValue: FlightKeys_To_NXOP
      targetMetricValue: “50”
      activationTargetMetricValue: “25”
```

1. For the overflow consumer application, the minimum replica count is set to Zero and Maximum to 10

2. References the actual custom metric in **Amazon CloudWatch** for the Health score that was calculated by the **AWS Lambda** function.

3. The targetMetricValue determines the scaling behavior along with the health score. KEDA uses the logic of

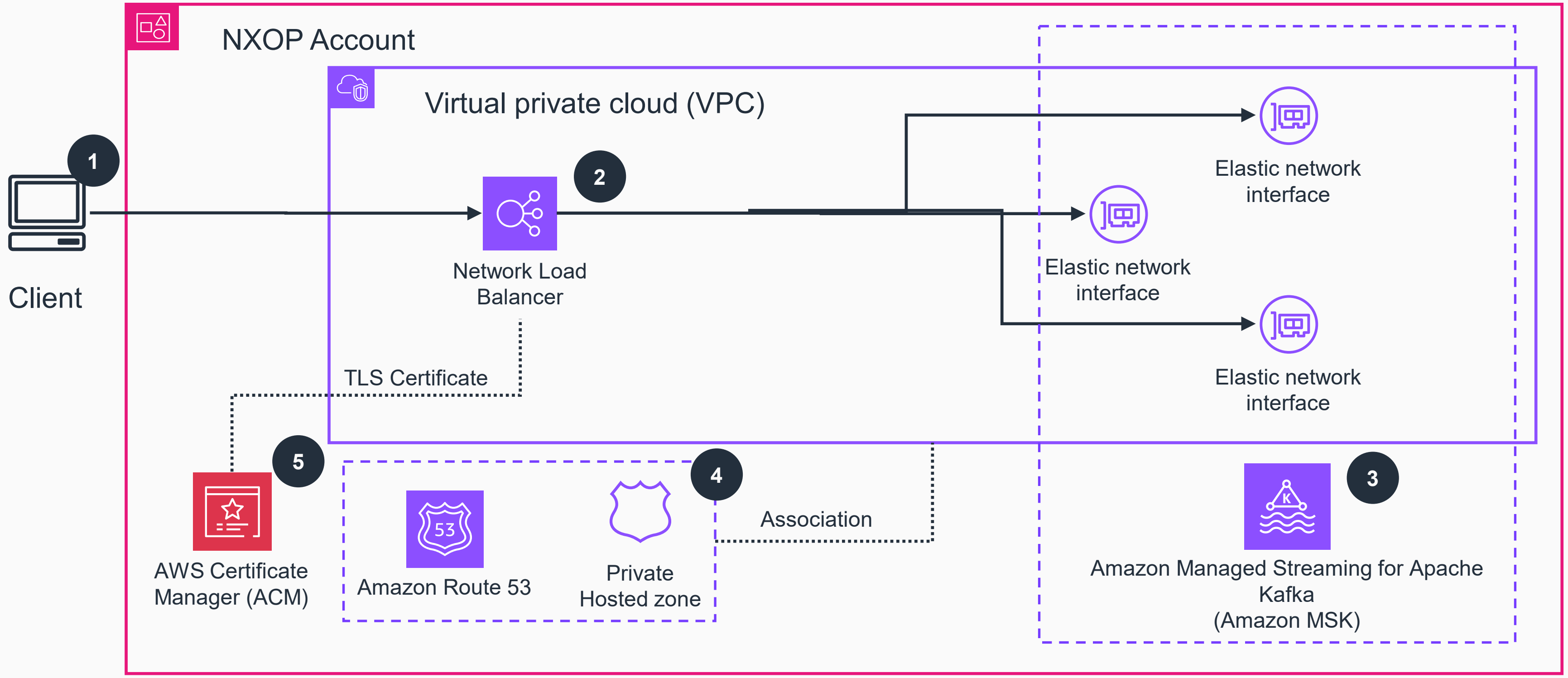
   replicas = ceil(health_score / targetMetricValue)

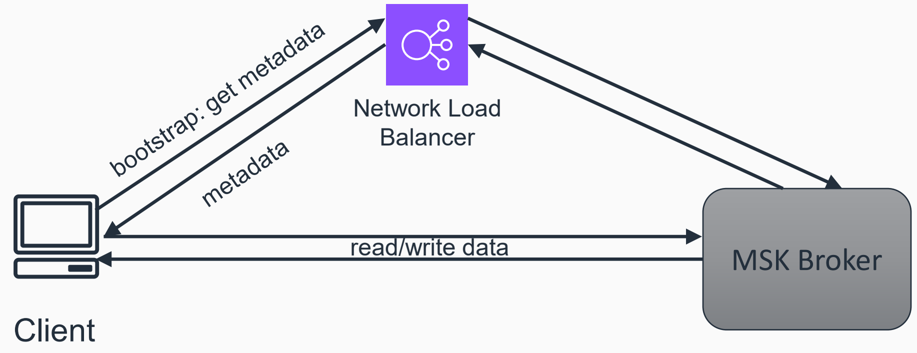
   Examples:

   - When Healthy
     health_score=0, then replicas=0
   - When 1 AZ impact
     health_score=75, then replicas=2
   - When RabbitMQ impact
     health_score=100, then replicas=2
   - When RabbitMQ + 1 AZ impact
     health_score=175, then replicas=4
   - When RabbitMQ + 2 AZ impact
     health_score=250, then replicas=5
   - When RabbitMQ + 3 AZ impact
     health_score=325, then replicas=7

MSK – Client Connectivity

This diagram shows the approach for setting up NLB for the initial connection bootstrap to MSK brokers


NXOP Account
Virtual private cloud (VPC)
1
Client
2
Network Load Balancer
Elastic network interface
Elastic network interface
Elastic network interface
TLS Certificate
5
AWS Certificate Manager (ACM)
Amazon Route 53
Private Hosted zone
4
Association
3
Amazon Managed Streaming for Apache Kafka (Amazon MSK)



1 External clients initiate connection to the custom domain for MSK bootstrap and metadata retrieval.

2 The AWS Network Load Balancer receives the client request and applies TLS certificate from ACM (Step 5) to enforce transport layer encryption for secure communication.

3 MSK brokers are deployed in 3 AZs to provide high availability

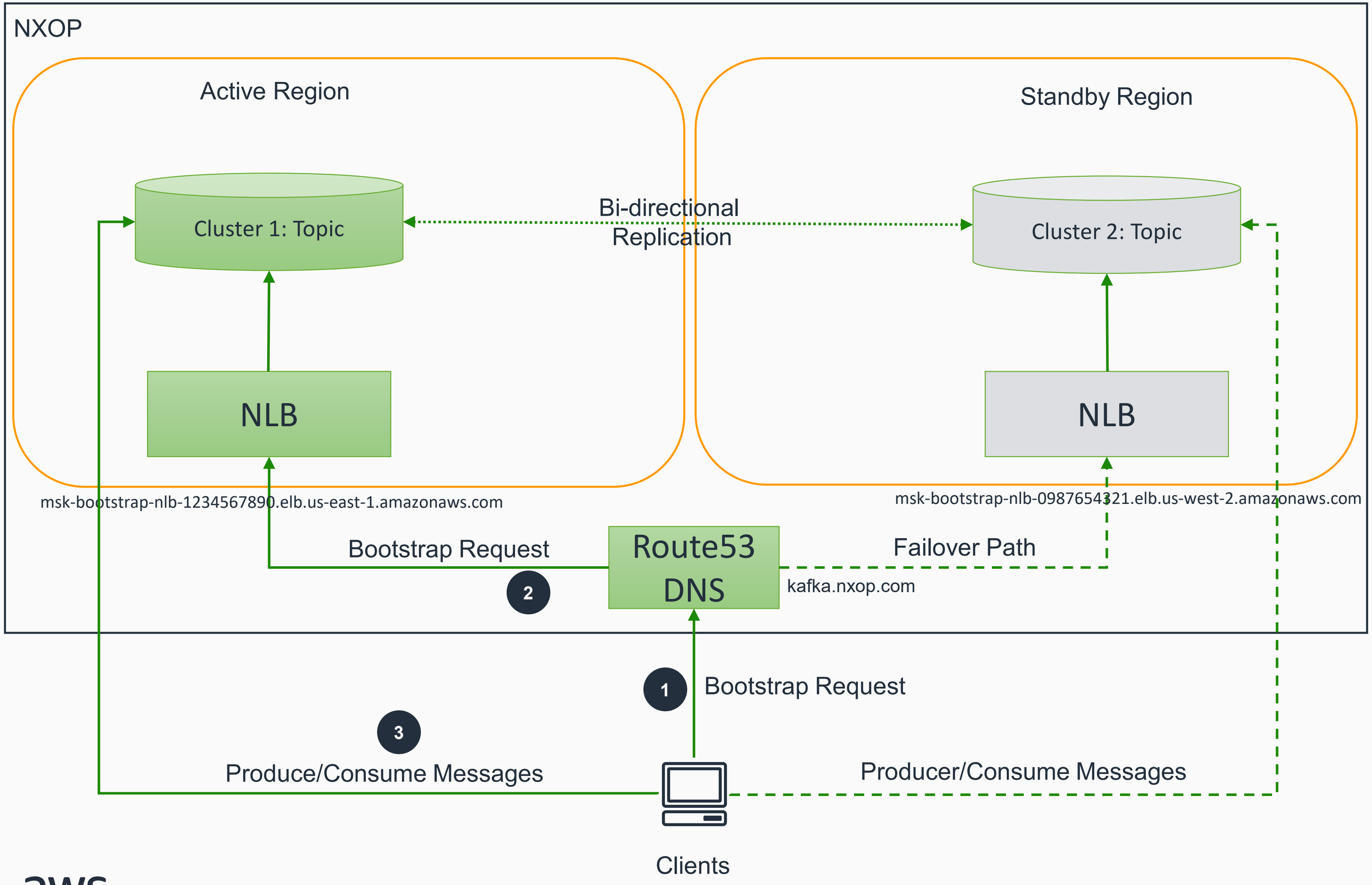
4 Route 53 Private Hosted Zone helps with DNS resolution of the custom domain and maintains the association between the custom domain and the Network Load Balancer endpoint.

5 AWS Certificate Manager (ACM) provides the TLS certificate used by the Network Load Balancer to secure client connections during the bootstrap process.

Reviewed for technical accuracy, 2025

MSK Producers/Consumers Network Path

This shows the network connectivity path for MSK Producers/Consumers



1 MSK Clients connect to MSK via the Route53 DNS layer for resolving the actual brokers and get connected to the Active region

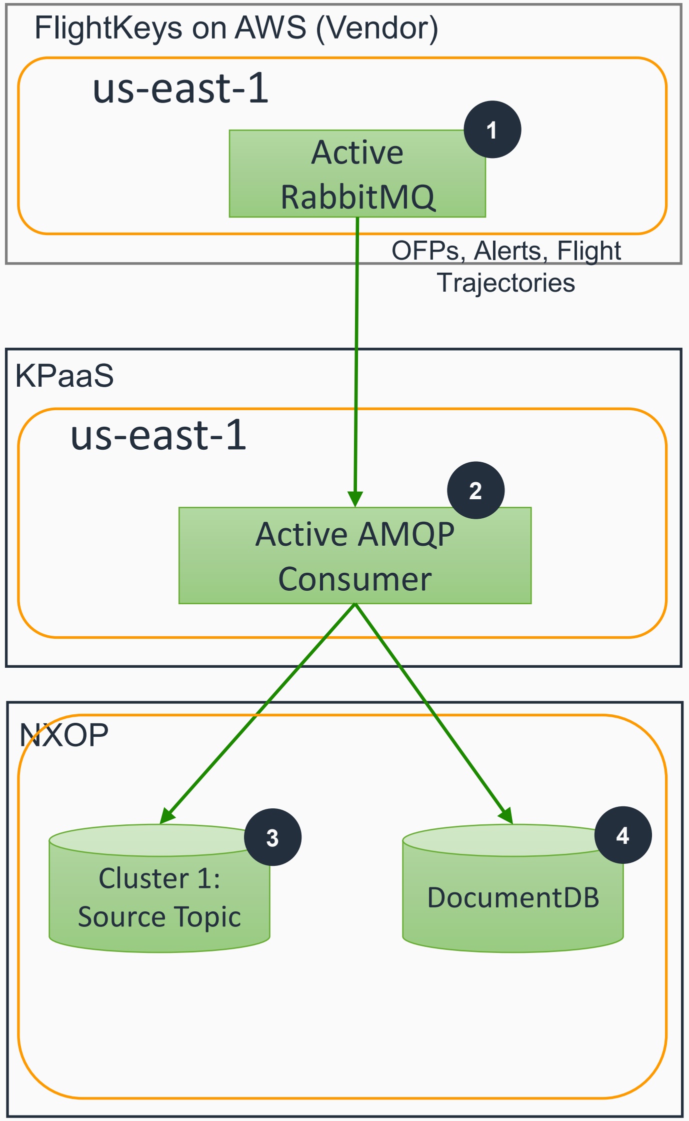
2 The Route53 DNS record resolves the request to the NLB in the active Region, and thereby enabling the bootstrap request to be handled by the MSK brokers in the active Region.

3 The MSK clients connect directly to the broker in the Active Region to produce/consume messages.

Reviewed for technical accuracy, 2025

FXIP MVP for 11/14 – Single Region

This diagram shows the single Region setup for 11/14 MVP deadline



1 FlightKeys 5D engine publishes flight optimization data (OFPs, alerts, trajectories) to RabbitMQ queue in us-east-1 (Active Region)

2 Active AMQP Consumer consuming FlightKeys messages, validating schema, and transforming for downstream systems integrated to consume from the RabbitMQ queue in us-east-1

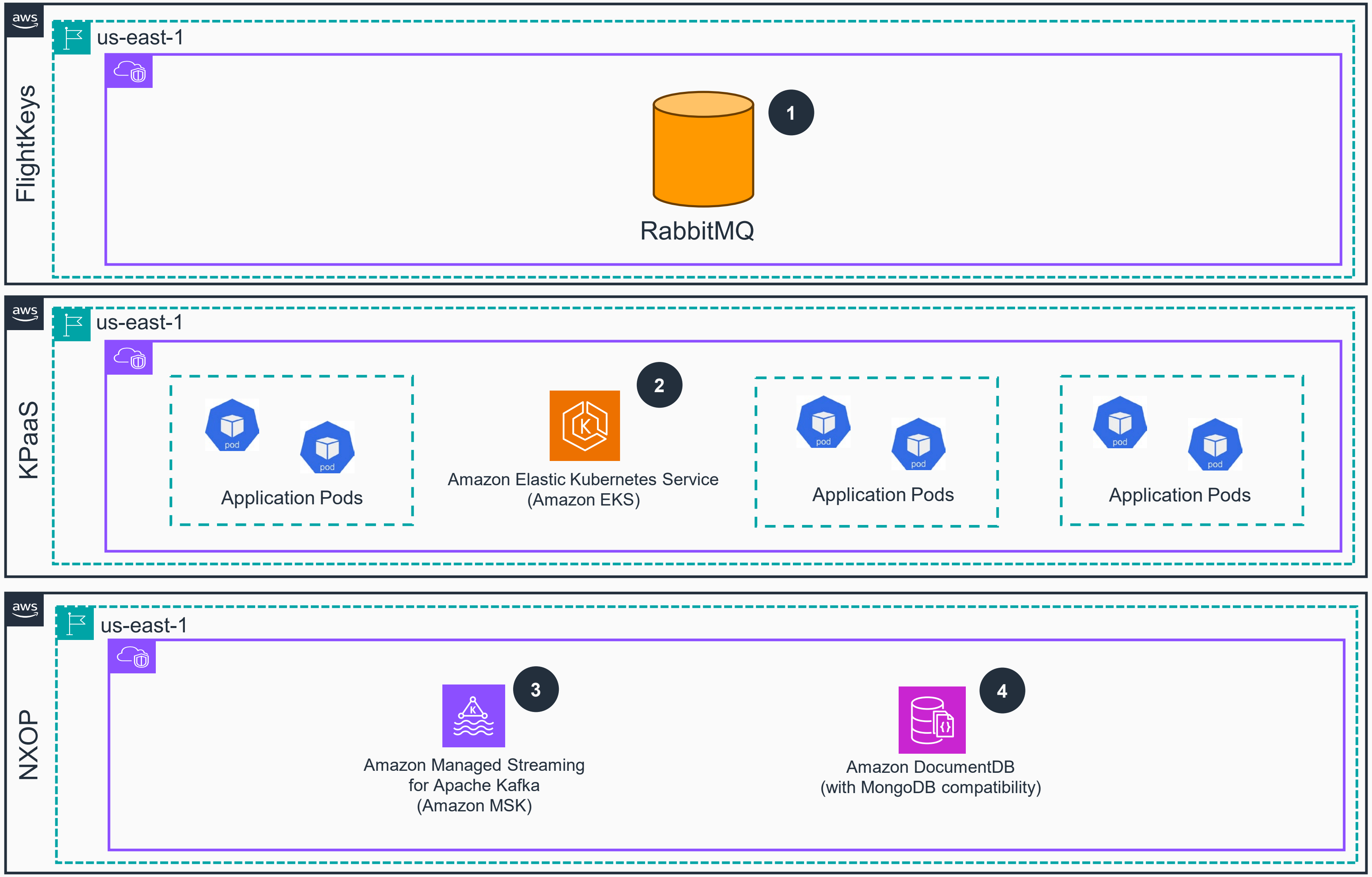
3 MSK cluster in us-east-1 receiving validated flight data from active FXIP processor, in Source Topic

4 Application also writes to DocumentDB in the same Region

Reviewed for technical accuracy, 2025

FXIP MVP for 11/14 – Single Region

This diagram shows the single Region setup for 11/14 MVP deadline


FlightKeys
us-east-1
RabbitMQ
1
KPaaS
us-east-1
Application Pods
2
Amazon Elastic Kubernetes Service (Amazon EKS)
Application Pods
Application Pods
pod
NXOP
us-east-1
3
Amazon Managed Streaming for Apache Kafka (Amazon MSK)
4
Amazon DocumentDB (with MongoDB compatibility)

1 FlightKeys 5D engine publishes flight optimization data (OFPs, alerts, trajectories) to RabbitMQ queue in us-east-1 (Active Region)

2 Application Pods deployed on EKS Cluster consume messages from FlightKeys RabbitMQ and produce to MSK cluster as well as write to DocumentDB.

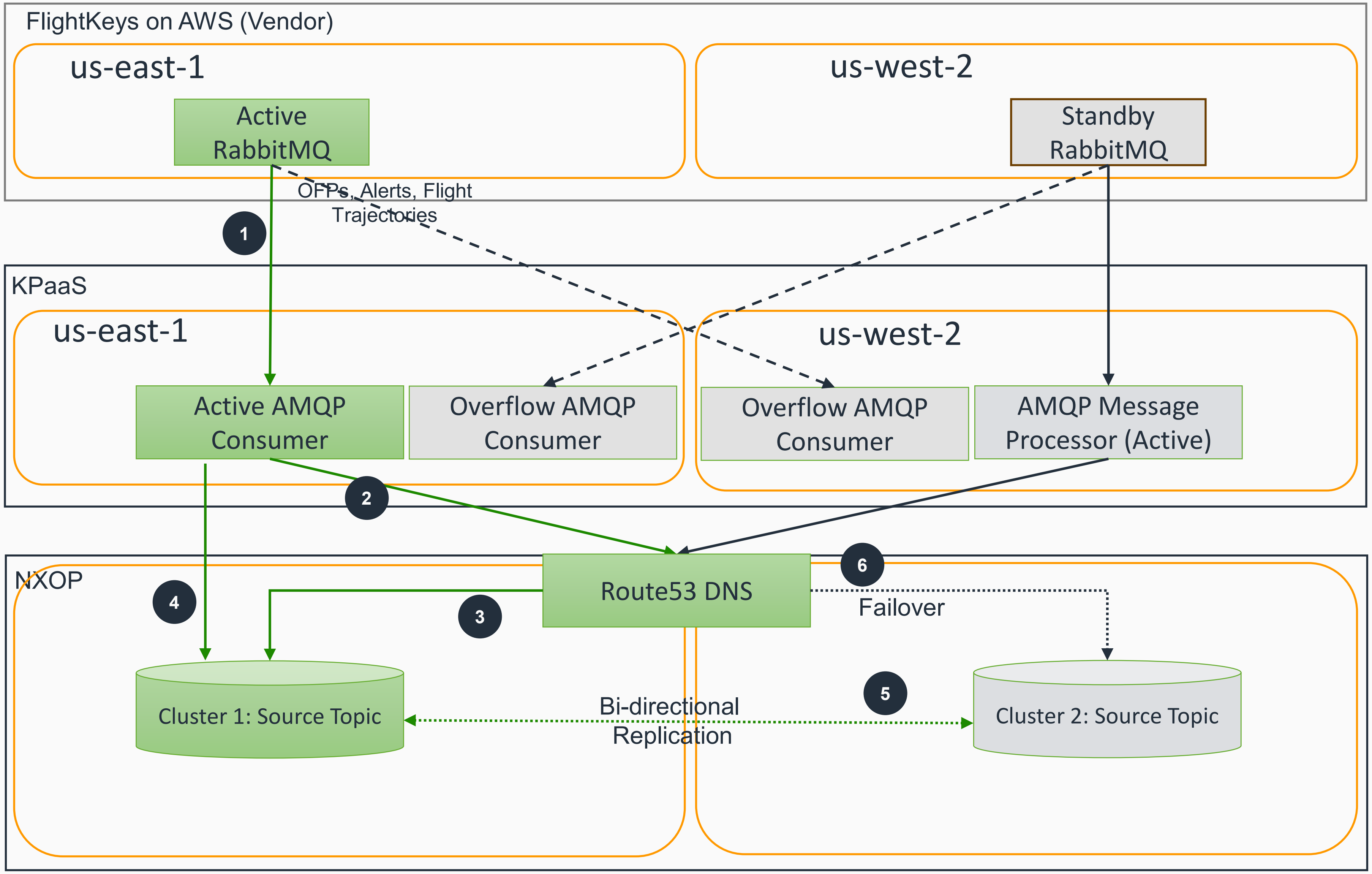
3 MSK cluster in us-east-1 receiving validated flight data from application pods.

4 The application on EKS also writes data to DocumentDB.

Reviewed for technical accuracy, 2025

FXIP - Data Flow – FlightKeys to NXOP (Inbound)

This diagram shows the data flow for FXIP across the various systems/components from Flight Keys to MSK integration layer



1 Active AMQP Consumer in us-east-1 subscribes and receives messages from FlightKeys RabbitMQ in us-east-1.

2 The Active AMQP Consumer get routed to the Active MSK cluster by using the bootstrap process that gets routed to the Route53 DNS Record.

3 The Route53 DNS record resolves to the NLB in the active Region which then enables the request to resolve to the MSK brokers behind the NLB.

4 The Active AMQP Consumer produce data directly to the active MSK cluster

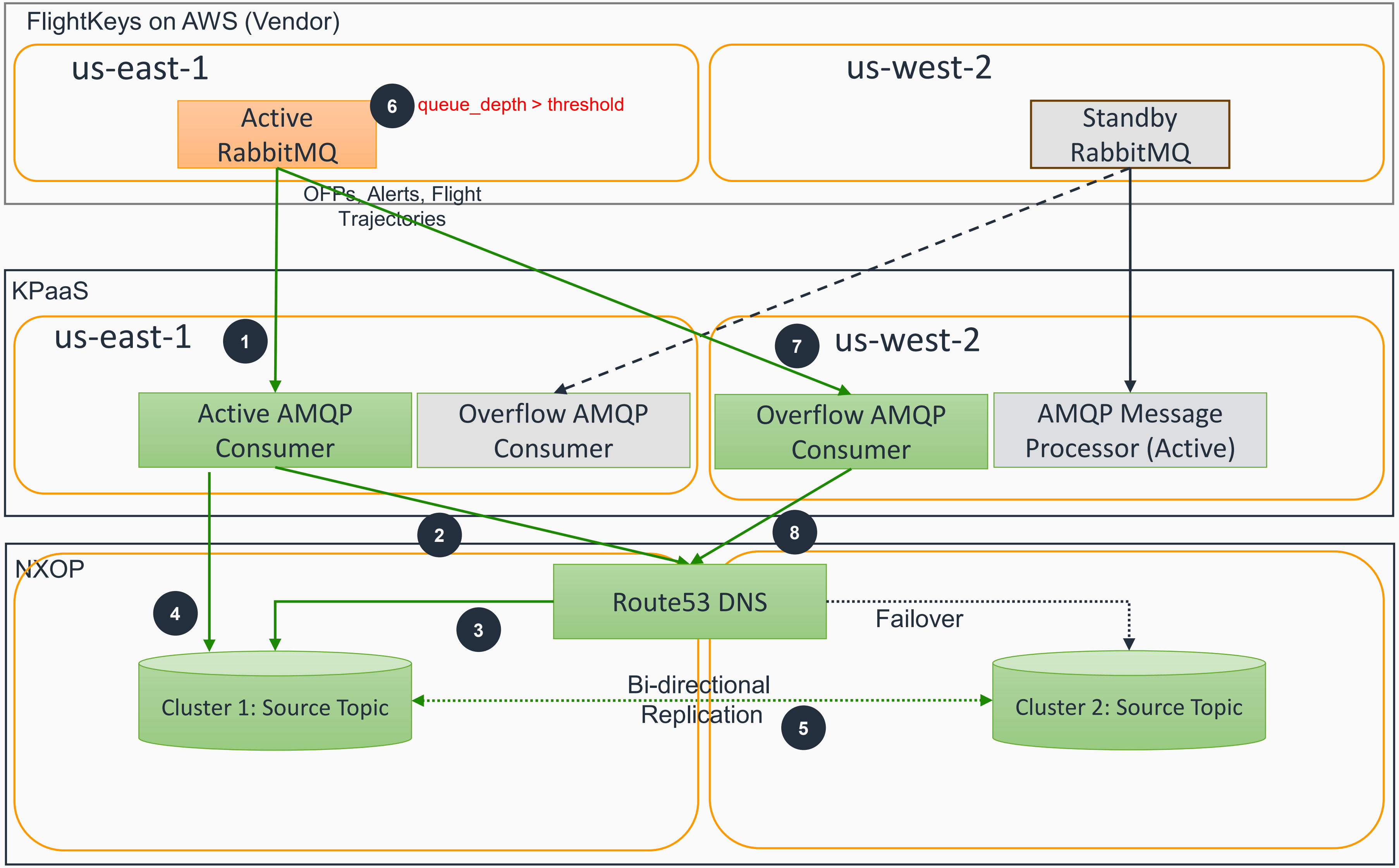
5 The Source Topic in the active MSK Cluster is replicated to the Source Topic in the Standby region MSK Cluster

6 During a failover, the route53 DNS record will resolve the bootstrap requests to the MSK cluster in the standby Region.

Reviewed for technical accuracy, 2025

FXIP - Data Flow – FlightKeys to NXOP (Inbound) – Failure Scenario 1

This diagram shows the data flow for FXIP across the various systems/components from Flight Keys to MSK integration layer when the RabbitMQ queue depth increases beyond a threshold



1 Active AMQP Consumer in us-east-1 subscribes and receives messages from FlightKeys RabbitMQ in us-east-1.

2 The Active AMQP Consumer get routed to the Active MSK cluster by using the bootstrap process that gets routed to the Route53 DNS Record.

3 The Route53 DNS record resolves to the NLB in the active Region which then enables the request to resolve to the MSK brokers behind the NLB.

4 The Active AMQP Consumer produce data directly to the active MSK cluster

5 The Source Topic in the active MSK Cluster is replicated to the Source Topic in the Standby region MSK Cluster

6 The RabbitMQ queue depth increases beyond threshold due to multiple possible reasons, and triggers an alarm.

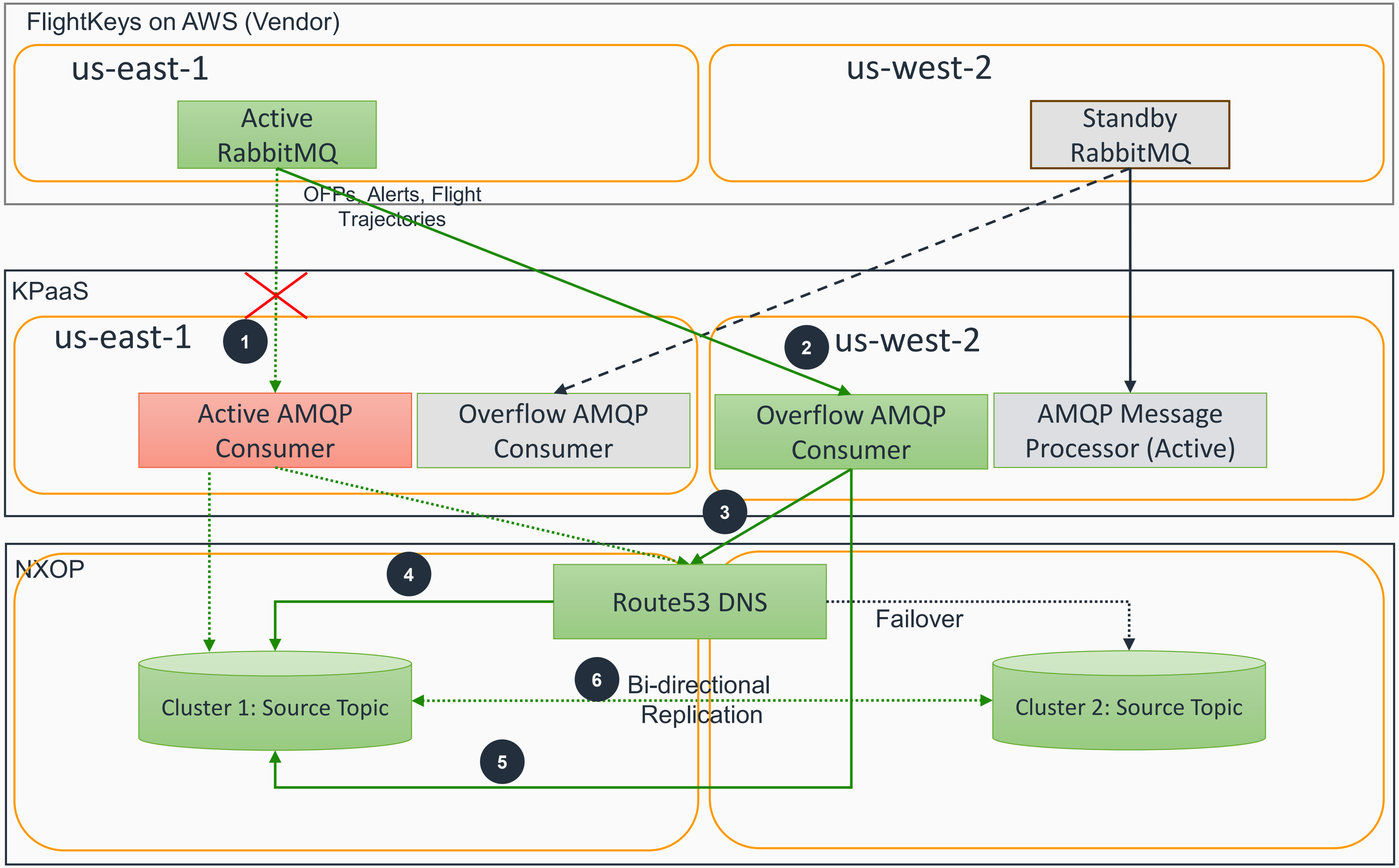
7 The alarm affects the overall health score of the system and triggers the overflow AMQP consumer in us-west-2 to scale out to also start consuming from the RabbitMQ in us-east-1.

8 Like the active AMQP consumer, the overflow AMQP consumer also follows the same connectivity process to connect to the active MSK cluster.

Reviewed for technical accuracy, 2025

FXIP - Data Flow – FlightKeys to NXOP (Inbound) – Failure Scenario 2

This diagram shows the data flow for FXIP across the various systems/components from Flight Keys to MSK integration layer when the RabbitMQ queue depth increases beyond a threshold, due an issue with the EKS components

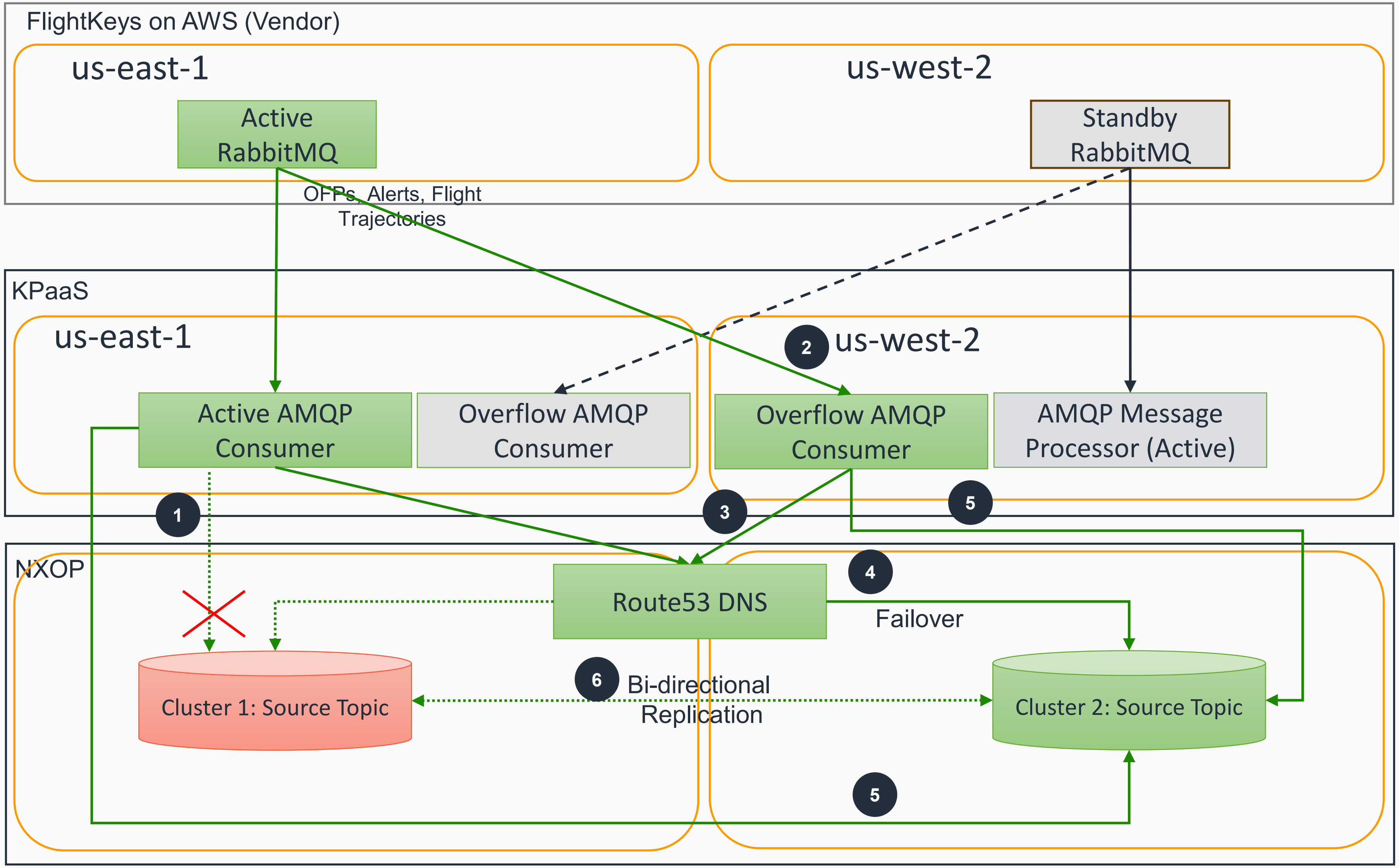


1. Active AMQP Consumer in us-east-1 has an impairment preventing it from consuming messages from the FlightKeys RabbitMQ in us-east-1, and triggers an alarm.
2. The alarm affects the overall health score of the system and triggers the overflow AMQP consumer in us-west-2 to scale out to also start consuming from the RabbitMQ in us-east-1.
3. Like the active AMQP consumer, the overflow AMQP consumer also follows the same connectivity process to connect to the active MSK cluster via the Route53 DNS Record.
4. The Route53 DNS record resolves to the NLB in the active Region which then enables the request to resolve to the MSK brokers behind the NLB.
5. The Overflow AMQP Consumer produce data directly to the active MSK cluster
6. The Source Topic in the active MSK Cluster is replicated to the Source Topic in the Standby region MSK Cluster

Reviewed for technical accuracy, 2025

FXIP - Data Flow – FlightKeys to NXOP (Inbound) – Failure Scenario 3

This diagram shows the data flow for FXIP across the various systems/components from Flight Keys to MSK integration layer when the RabbitMQ queue depth increases beyond a threshold, due to the MSK cluster impairment in us-east-1



1. The MSK in us-east-1 has an impairment preventing the Active AMQP Consumer in us-east-1 from producing message to the cluster topic and triggers an alarm.

2. The alarm affects the overall health score of the system and triggers the overflow AMQP consumer in us-west-2 to scale out to also start consuming from the RabbitMQ in us-east-1.

3. Like the active AMQP consumer, the overflow AMQP consumer also follows the same connectivity process to connect to the active MSK cluster via the Route53 DNS Record.

4. The Route53 DNS record resolves to the NLB in the Standby Region which then enables the request to resolve to the MSK brokers behind the NLB.

5. The Overflow AMQP Consumer produce data directly to the standby MSK cluster, and once the Active AMQP consumer is able to recover starts producing the Standby MSK cluster.

6. The Source Topic in the standby MSK Cluster is replicated to the Source Topic in the Active region MSK Cluster

Reviewed for technical accuracy, 2025

Overflow Consumer Scaling Criteria Matrix

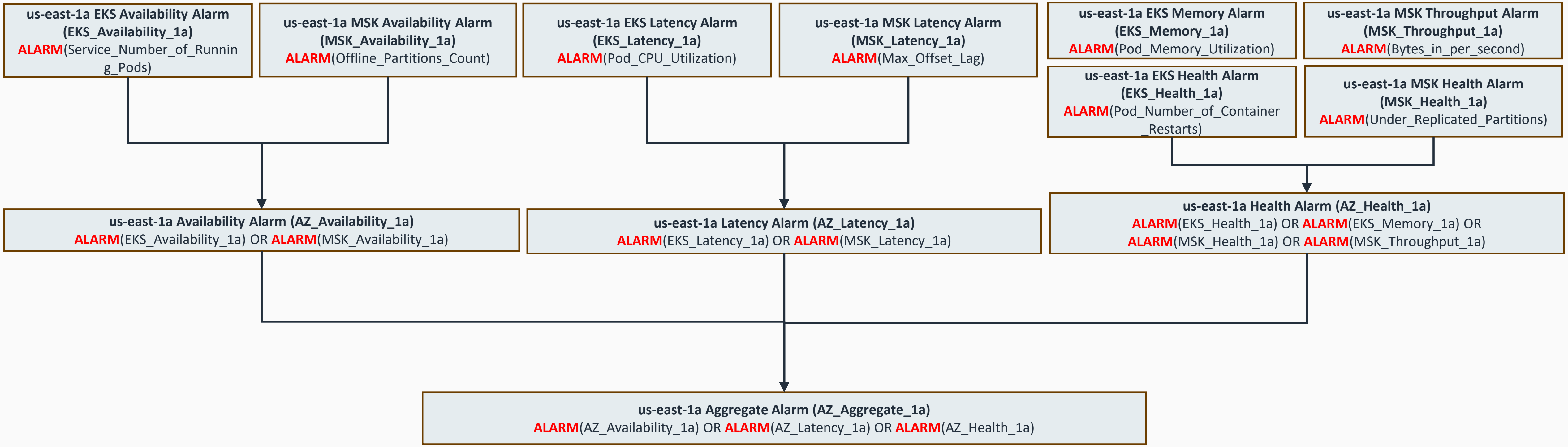
This shows a comprehensive suite of metrics that will be used for scaling criteria for the overflow consumers

Component	Metrics	Threshold	Weight	Scaling Action	Rationale	Custom/Out of the Box
RabbitMQ	Queue Depth	> 10,000 messages	Critical	Scale to 10	Message Backlog indicates slow/stalled processing	Custom (due to Vendor ownership)
	Connection Count	< 2 connections	Critical	Scale to 10	Low/No connections = No message consumption	Custom (due to Vendor ownership)
EKS Pods	Number of running Pods	< 2 Pods	High	Scale to 5	Insufficient processing capacity	EKS Container Insights (Out of the box)
	Pod Number of Container Restarts	> 5 /min	High	Scale to 3	Pod instability	EKS Container Insights (Out of the box)
	Pod CPU Utilization	> 80%	Medium	Scale to 2	Resource pressure	EKS Container Insights (Out of the box)
	Pod Memory Utilization	> 85%	Medium	Scale to 2	Memory pressure	EKS Container Insights (Out of the box)
MSK	Consumer Lag	3000	High	Scale to 5	Downstream processing behind	MSK Metrics (Out of the box)
	Offline Partition Count	> 0	Critical	Scale to 10	Data unavailability	MSK Metrics (Out of the box)
	Under replicated Partitions	> 0	Medium	Scale to 3	Reduced redundancy	MSK Metrics (Out of the box)
	Bytes in per second	< 50% of baseline	Medium	Scale to 2	Reduced throughput	MSK Metrics (Out of the box)

Reviewed for technical accuracy, 2025

Alarm Structure for AZ Fault Detection

This shows a comprehensive Alarm Structure by Region to detect AZ faults



us-east-1a EKS Availability Alarm (EKS_Availability_1a)
ALARM(Service_Number_of_Running_Pods)

us-east-1a MSK Availability Alarm (MSK_Availability_1a)
ALARM(Offline_Partitions_Count)

us-east-1a EKS Latency Alarm (EKS_Latency_1a)
ALARM(Pod_CPU_Utilization)

us-east-1a MSK Latency Alarm (MSK_Latency_1a)
ALARM(Max_Offset_Lag)

us-east-1a EKS Memory Alarm (EKS_Memory_1a)
ALARM(Pod_Memory_Utilization)

us-east-1a MSK Throughput Alarm (MSK_Throughput_1a)
ALARM(Bytes_in_per_second)

us-east-1a EKS Health Alarm (EKS_Health_1a)
ALARM(Pod_Number_of_Container_Restarts)

us-east-1a MSK Health Alarm (MSK_Health_1a)
ALARM(Under_Replicated_Partitions)

us-east-1a Availability Alarm (AZ_Availability_1a)
ALARM(EKS_Availability_1a) OR ALARM(MSK_Availability_1a)

us-east-1a Latency Alarm (AZ_Latency_1a)
ALARM(EKS_Latency_1a) OR ALARM(MSK_Latency_1a)

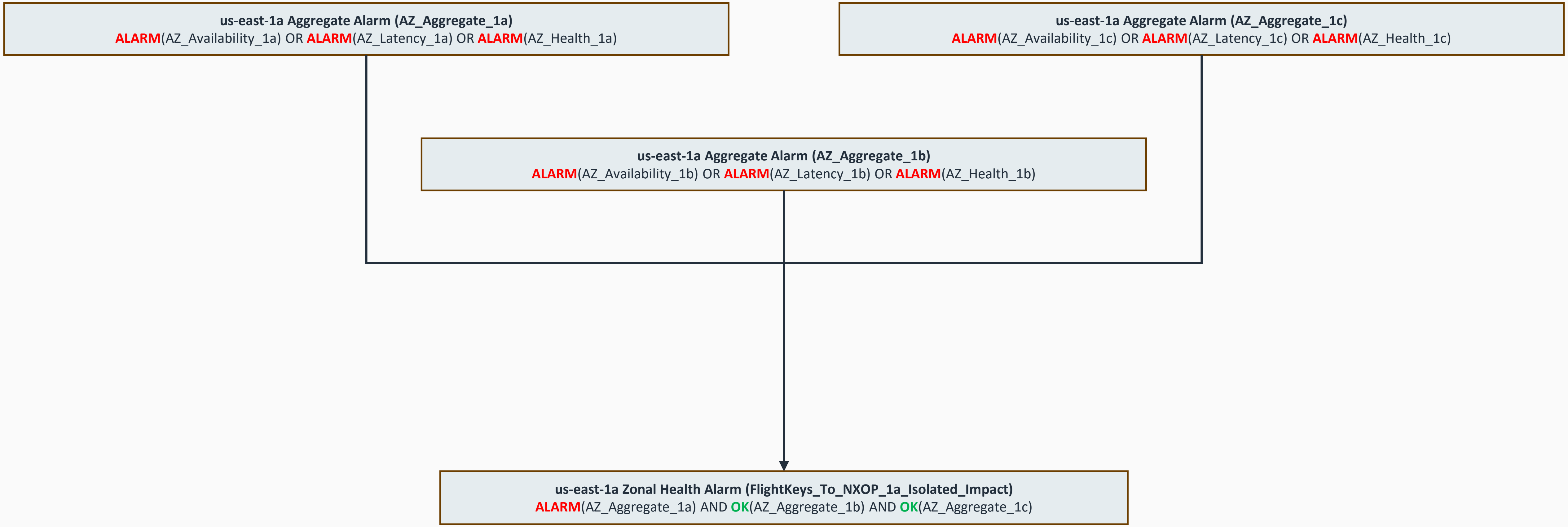
us-east-1a Health Alarm (AZ_Health_1a)
ALARM(EKS_Health_1a) OR ALARM(EKS_Memory_1a) OR ALARM(MSK_Health_1a) OR ALARM(MSK_Throughput_1a)

us-east-1a Aggregate Alarm (AZ_Aggregate_1a)
ALARM(AZ_Availability_1a) OR ALARM(AZ_Latency_1a) OR ALARM(AZ_Health_1a)

Reviewed for technical accuracy, 2025

Alarm Structure for AZ Health – FlightKeys to NXOP Data Flow

This shows a comprehensive Alarm Structure by Region to detect AZ faults



us-east-1a Aggregate Alarm (AZ_Aggregate_1a)
ALARM(AZ_Availability_1a) OR ALARM(AZ_Latency_1a) OR ALARM(AZ_Health_1a)

us-east-1a Aggregate Alarm (AZ_Aggregate_1c)
ALARM(AZ_Availability_1c) OR ALARM(AZ_Latency_1c) OR ALARM(AZ_Health_1c)

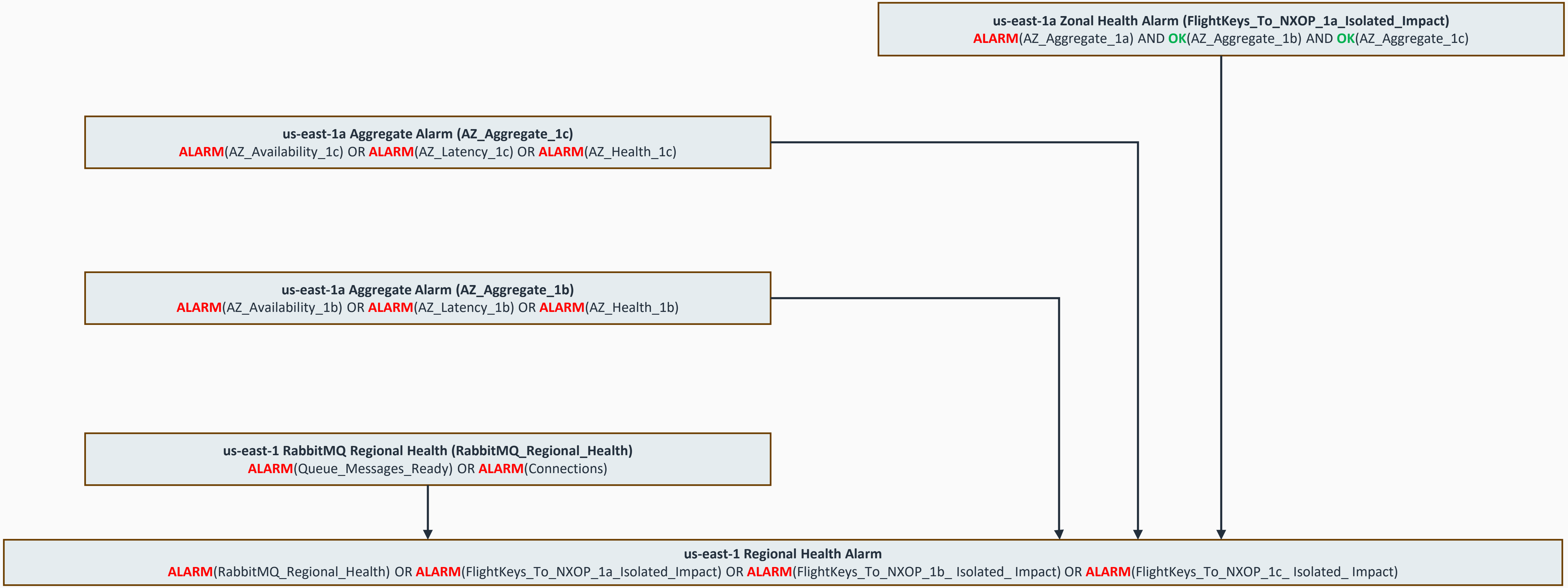
us-east-1a Aggregate Alarm (AZ_Aggregate_1b)
ALARM(AZ_Availability_1b) OR ALARM(AZ_Latency_1b) OR ALARM(AZ_Health_1b)

us-east-1a Zonal Health Alarm (FlightKeys_To_NXOP_1a_Isolated_Impact)
ALARM(AZ_Aggregate_1a) AND OK(AZ_Aggregate_1b) AND OK(AZ_Aggregate_1c)

Reviewed for technical accuracy, 2025

Alarm Structure for Regional Health – FlightKeys to NXOP Data Flow

This shows a comprehensive Alarm Structure for Regional Health of FlightKeys to NXOP Data Flow



us-east-1a Zonal Health Alarm (FlightKeys_To_NXOP_1a_Isolated_Impact)
ALARM(AZ_Aggregate_1a) AND OK(AZ_Aggregate_1b) AND OK(AZ_Aggregate_1c)

us-east-1a Aggregate Alarm (AZ_Aggregate_1c)
ALARM(AZ_Availability_1c) OR ALARM(AZ_Latency_1c) OR ALARM(AZ_Health_1c)

us-east-1a Aggregate Alarm (AZ_Aggregate_1b)
ALARM(AZ_Availability_1b) OR ALARM(AZ_Latency_1b) OR ALARM(AZ_Health_1b)

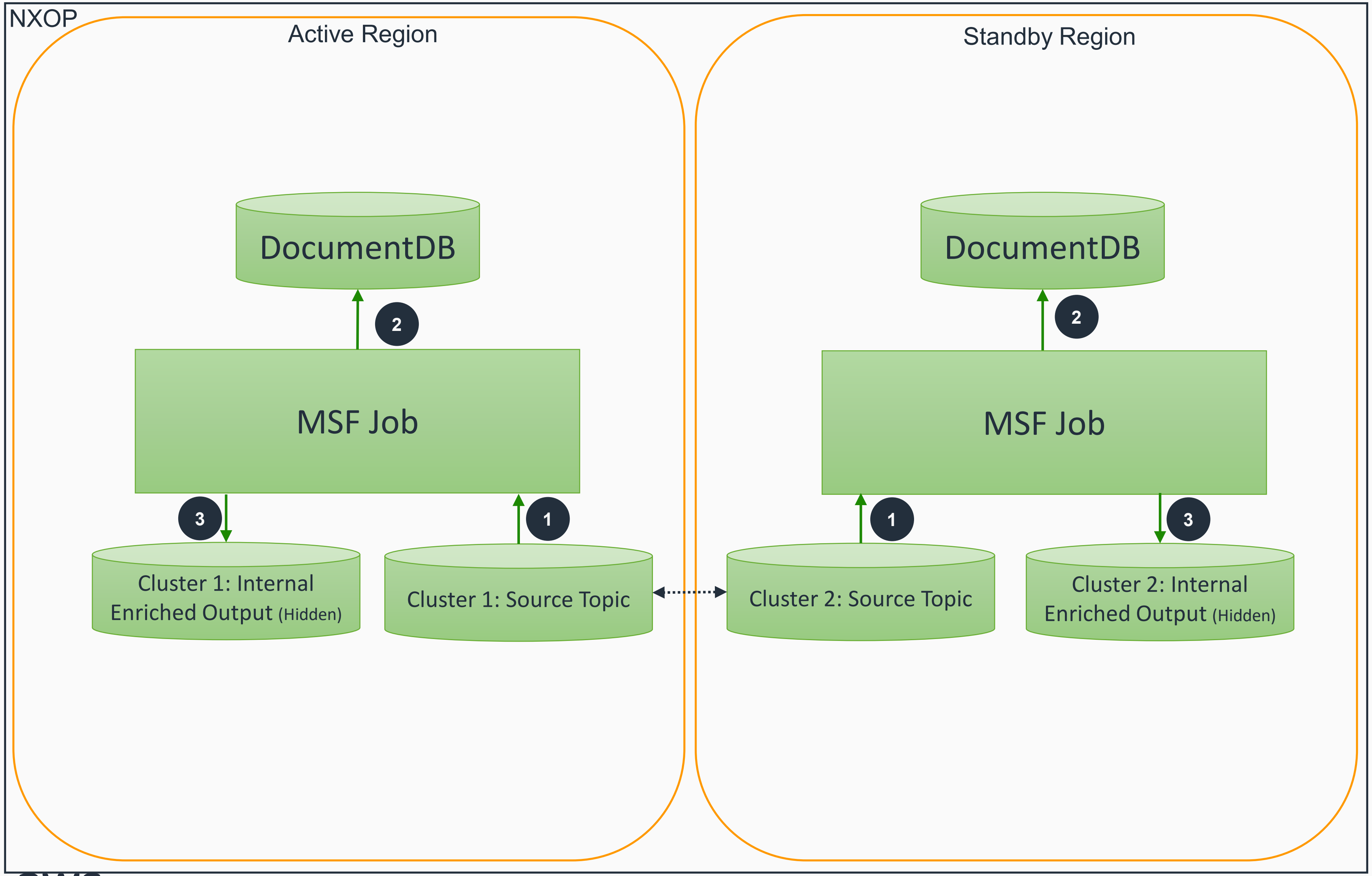
us-east-1 RabbitMQ Regional Health (RabbitMQ_Regional_Health)
ALARM(Queue_Messages_Ready) OR ALARM(Connections)

us-east-1 Regional Health Alarm
ALARM(RabbitMQ_Regional_Health) OR ALARM(FlightKeys_To_NXOP_1a_Isolated_Impact) OR ALARM(FlightKeys_To_NXOP_1b_ Isolated_ Impact) OR ALARM(FlightKeys_To_NXOP_1c_ Isolated_ Impact)

Reviewed for technical accuracy, 2025

NXOP – ASM Data Flow

This diagram shows the data flow for ASM



1 MSF jobs consume data from the source topic in MSK cluster in the same Region.

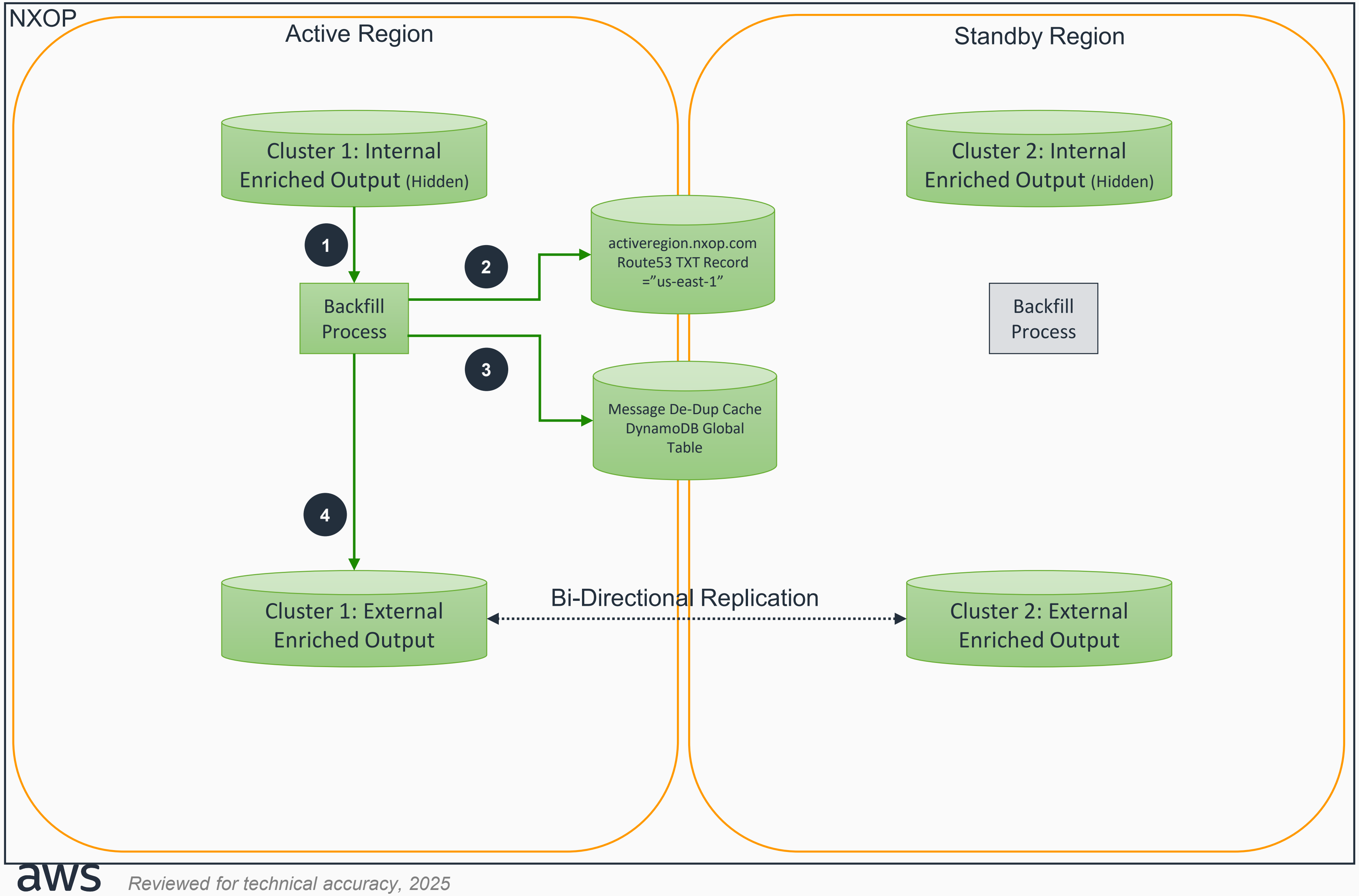
2 The job enriches the data and writes to DocumentDB cluster in region.

3 The job also writes to an enriched output topic which is internal and not replicated across Regions.

Reviewed for technical accuracy, 2025

NXOP – ASM Data Flow

This diagram shows the data flow for ASM



NXOP
Active Region
Standby Region
Cluster 1: Internal Enriched Output (Hidden)
1
Backfill Process
2
activeregion.nxop.com Route53 TXT Record =”us-east-1”
3
Message De-Dup Cache DynamoDB Global Table
4
Cluster 1: External Enriched Output
Bi-Directional Replication
Cluster 2: Internal Enriched Output (Hidden)
Backfill Process
Cluster 2: External Enriched Output

1 Backfill process in the Active Region consumes data from the Internal Enriched Output topic.

2 The Backfill process validates if the current Region is still the Active region by performing a Route53 TXT Record DNS lookup

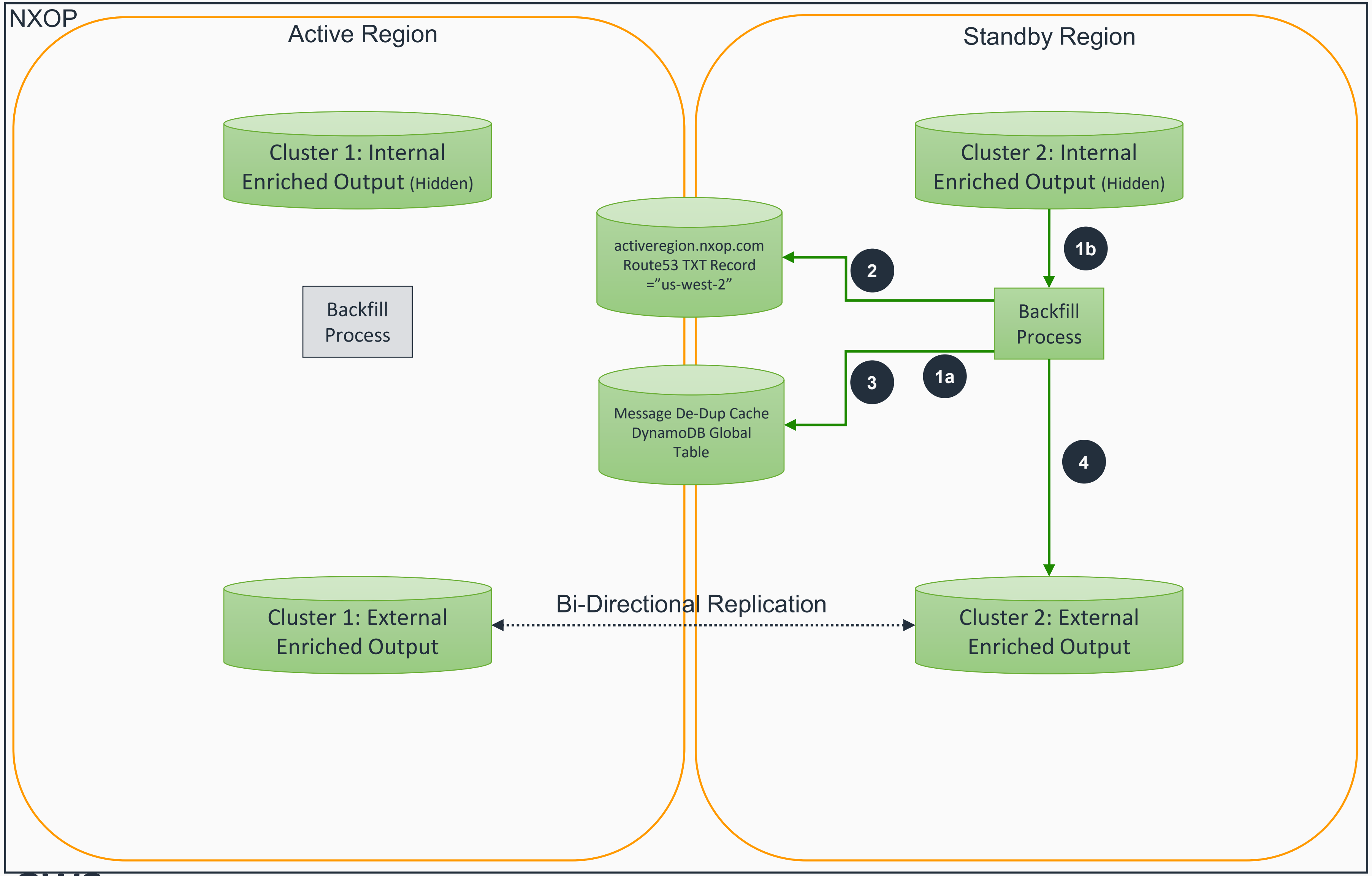
3 If still operating in the Active Region, proceed to validate if the current message being processed is a duplicate using the DynamoDB Global Table. If not a duplicate, insert an Item into the Table with the message metadata

4 Proceed to write to the External Enriched Output topic which is bi-directionally replicated across Regions, and then update the item in the DynamoDB Global Table.

Reviewed for technical accuracy, 2025

NXOP – ASM Data Flow - Failover

This diagram shows the data flow for ASM


NXOP
Active Region
Standby Region
Cluster 1: Internal Enriched Output (Hidden)
Cluster 2: Internal Enriched Output (Hidden)
activeregion.nxop.com Route53 TXT Record =”us-west-2”
Backfill Process
Backfill Process
1b
2
3
1a
4
Message De-Dup Cache DynamoDB Global Table
Cluster 1: External Enriched Output
Bi-Directional Replication
Cluster 2: External Enriched Output

1a Backfill process in the Standby Region looks up the last processed message timestamp from the de-dup cache table.

1b The process seeks the offset corresponding to the timestamp from the internal enriched output topic.

2 Validates that the current Region is still the Active Region.

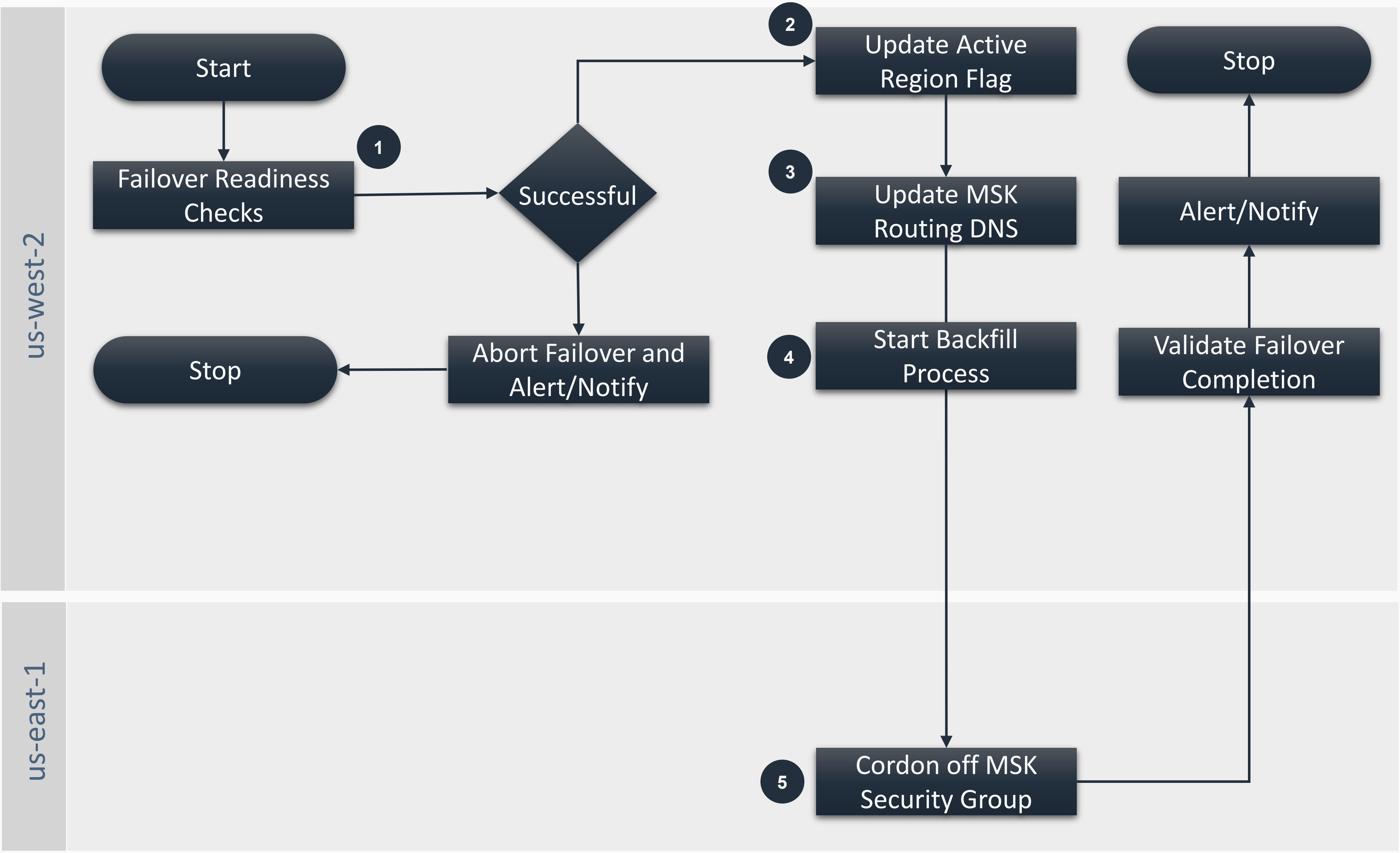
3 Proceeds to lookup the message metadata to validate for duplication, and makes an entry into the Table if not a duplicate

4 Produces the message to MSK in the Standby Region to the External Enriched Output topic which is bi-directionally replicated, and then updates the Item in the DynamoDB Global Table that the Item has been processed.

Reviewed for technical accuracy, 2025

NXOP – Region Failover Steps

This shows the steps involved in orchestrating a Region failover when failing over to us-west-2



us-west-2
us-east-1
Start
Failover Readiness Checks
1
Successful
Abort Failover and Alert/Notify
Stop
2
Update Active Region Flag
3
Update MSK Routing DNS
4
Start Backfill Process
5
Cordon off MSK Security Group
Validate Failover Completion
Alert/Notify
Stop

1. Validate
   - Us-west-2 Health
   - MSK Cluster Ready
   - Backfill Jobs health
   - DynamoDB Health
2. Update Route53 TXT record with the new value for active Region i.e; us-west-2
3. In Application Recovery Controller Routing Control Panel
   - Disable us-east-1 health check routing control
   - Enable us-west-2 health check routing control

   Validate DNS resolution to us-west-2 NLB
4. Start the Backfill process in the us-west-2 Region
5. Cross-Region Lambda invocation to remove inbound rules from MSK Security Group and block ports 9092, 9094

Reviewed for technical accuracy, 2025

Reviewed for technical accuracy, 2025