CONFLUENT

Disaster Recovery Options running Apache Kafka On Kubernetes

Jennifer Snipes
Staff Customer Success Technical Architect

Rema Subramanian
Customer Success Technical Architect

Contents

Resilient Kafka Architectures

Know your RTO & RPO

RPO

RPO (Recovery Point Objective) is about ***how much data you can afford to lose before it impacts business operations***. For example, for a banking system, 1 hour of data loss can be catastrophic as they operate live transactions.

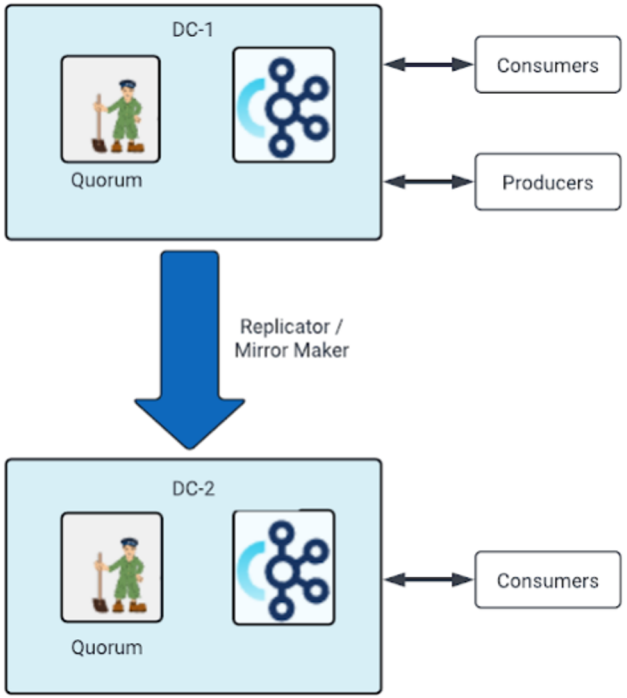
RTO

RTO (Recovery Time Objective) is ***the timeframe within which an application and systems must be restored*** after an outage.

Resilient Kafka Architectures

Active DC-1 / Passive DC-2

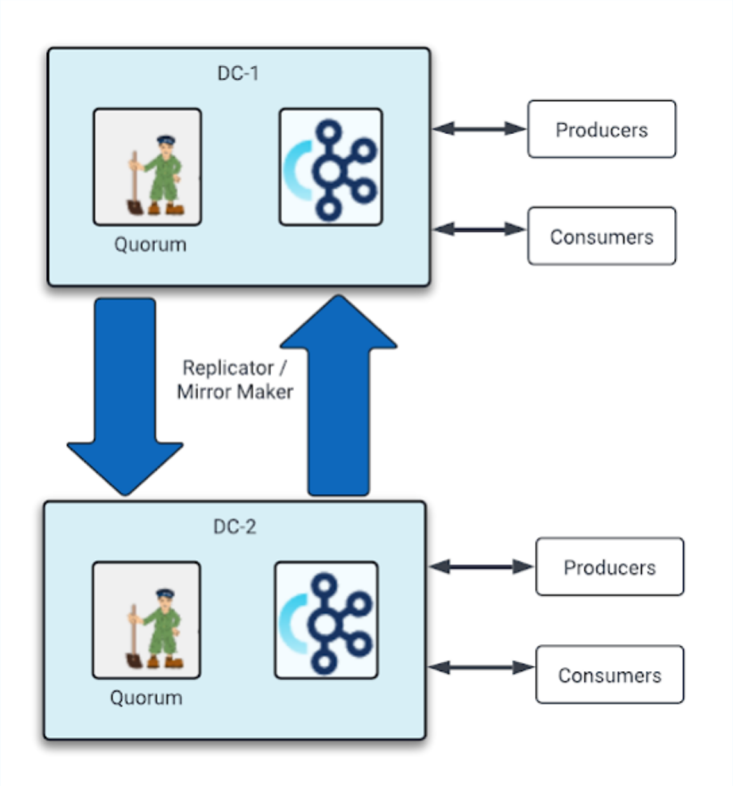
- Two independent clusters in different Data Centers / Regions
- Producers only in one Data Center
- Consumers in both Data Centers
- Multi-cloud / Multi-region
- One way Replication
- Asynchronous Replication
- RPO >0, RTO >0



Resilient Kafka Architectures

Active DC-1 / Active DC-2

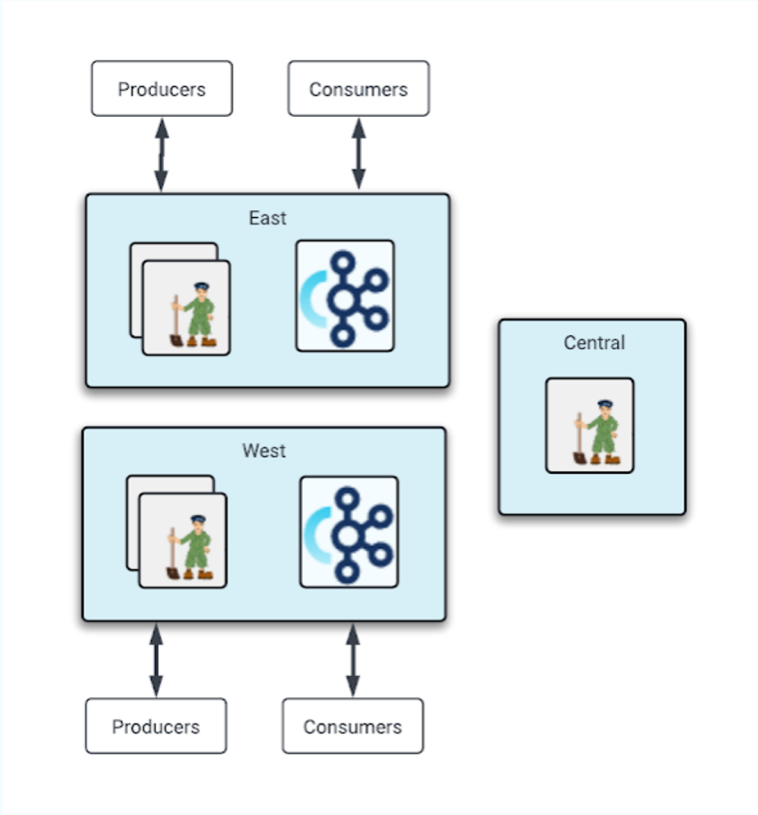
- Two independent clusters in different Data Centers / Regions
- Producers in both Data Centers
- Consumers in both Data Centers
- Multi-region / Multi-cloud
- Bi-Directional Replication with Provenance Headers
- Asynchronous Replication
- RPO >0, RTO >0



Resilient Kafka Architectures

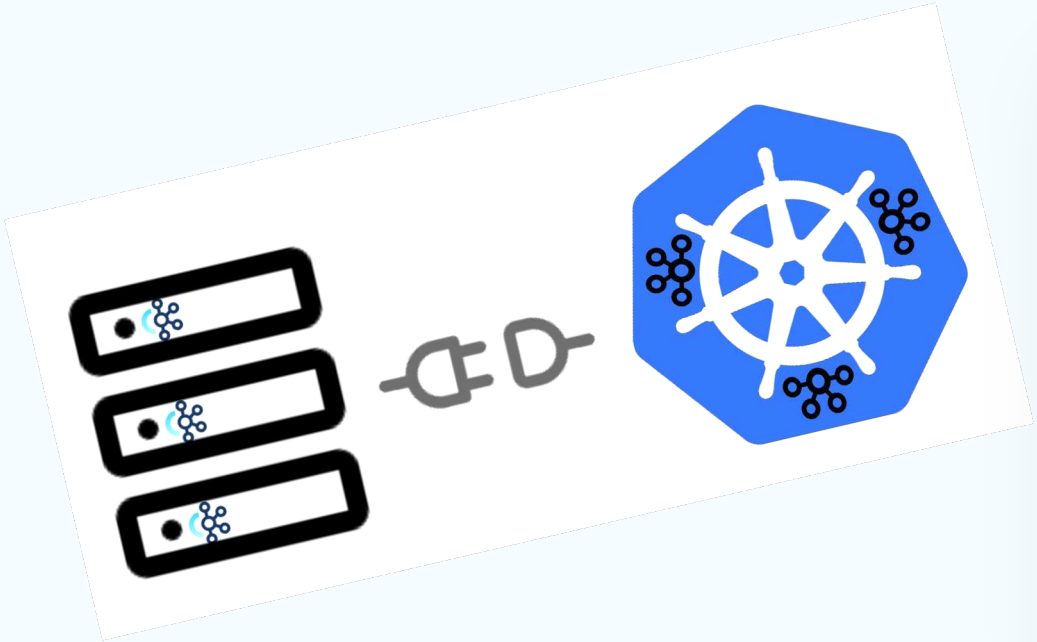
Stretch Cluster

- Single Cluster stretched across different Data Centers
- Producers write transparently across Data Centers
- Consumers in all Data Centers
- RPO = 0, RTO near 0
- Synchronous Replication native to cluster
- Asynchronous Replication with Observers*
- Replica & Observer placement defines Active-Active vs Active-Passive*
- Auto Observer Promotion*
- Multi-region*



Kafka & Kubernetes

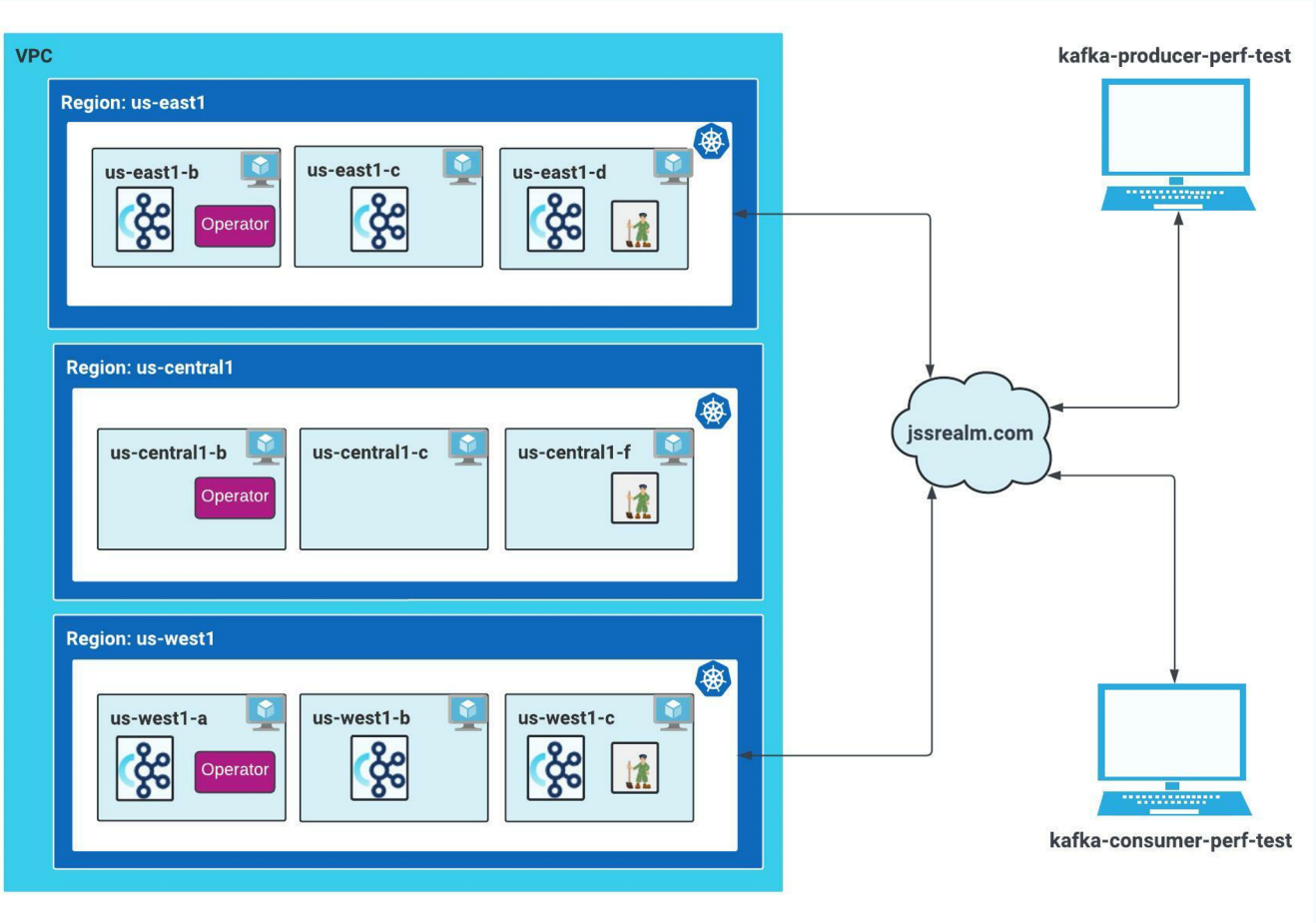
Kafka Deployment Arenas



Traditional vs K8s

- Broker
  - Hostnames /IPs
  - Placement across DCs
  - Communication across DCs
  - Failure
- Broker and ZK co-location
- Multi-tenancy

Stretch Cluster on Kubernetes



Built-in Disaster Recovery on Kubernetes

Kubernetes

Operator

Kafka Stretch Cluster

Chaos Testing / Monitoring

Getting Started

Building the K8s Cluster

1

GCP VPC Native Cluster

- Alias IP address range for nodes, pods and services
- Requires non-overlapping CIDR ranges

2

GKE Cluster

- Separately Managed Node Pool
- Node machine type
- Configurable number of nodes distributed across AZs
- Distinct namespace per cluster

3

Networking

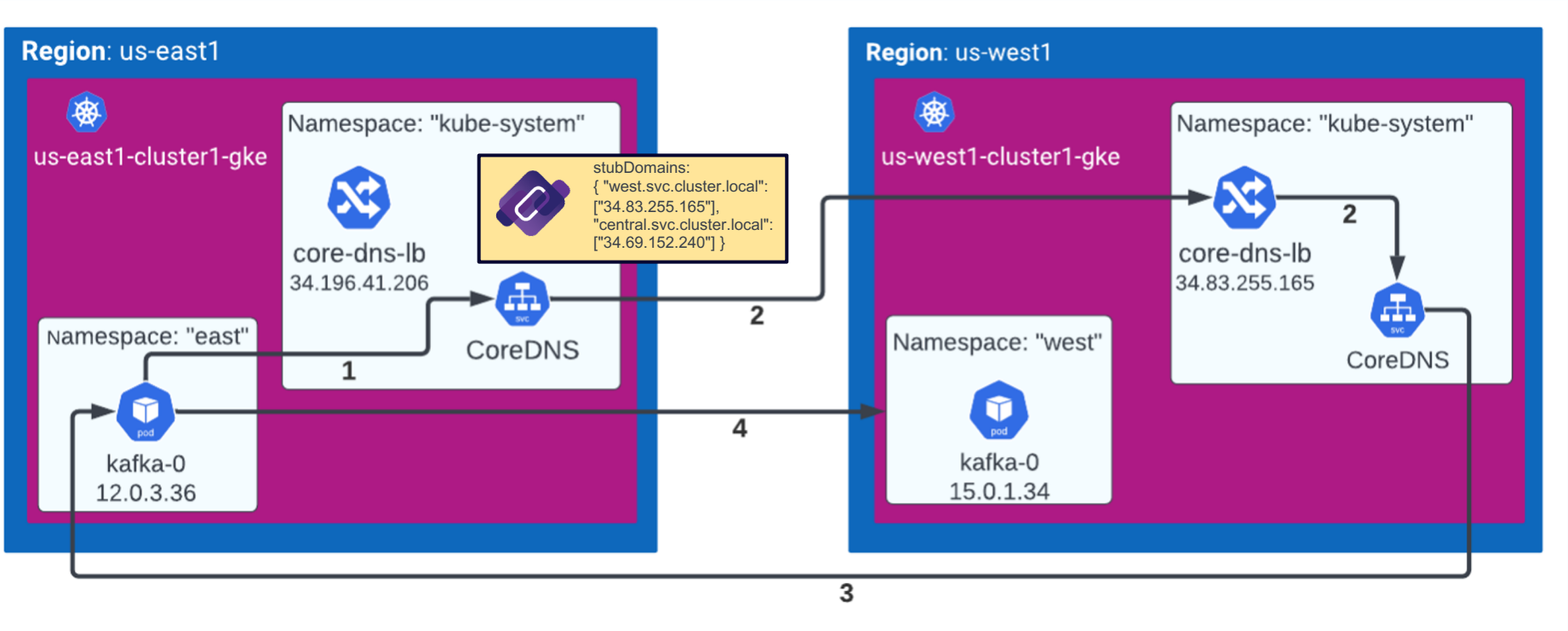
- VPC Native cluster installs routing
- Firewall rules
  - allow tcp between k8s clusters
  - allow access to 2181, 2888, 3888, 9092, 7778, 3000

4

StorageClass

- provisioner: kubernetes.io/gce-pd
- allowVolumeExpansion : true
- type: pd-ssd
- fstype: ext4
- reclaimPolicy: Retain
- volumeBindingMode: WaitForFirstConsumer

Networking between Kubernetes Clusters



Kubernetes Operator

Operator

1

CRDs

- Define various application components
- Medium to tie to kafka server.properties

2

Controller

- CRD's behavior
- Reconciliation loop

3

Services

- Headless Service
- Expose individual pods as external services
- Bootstrap LB service to get metadata

4

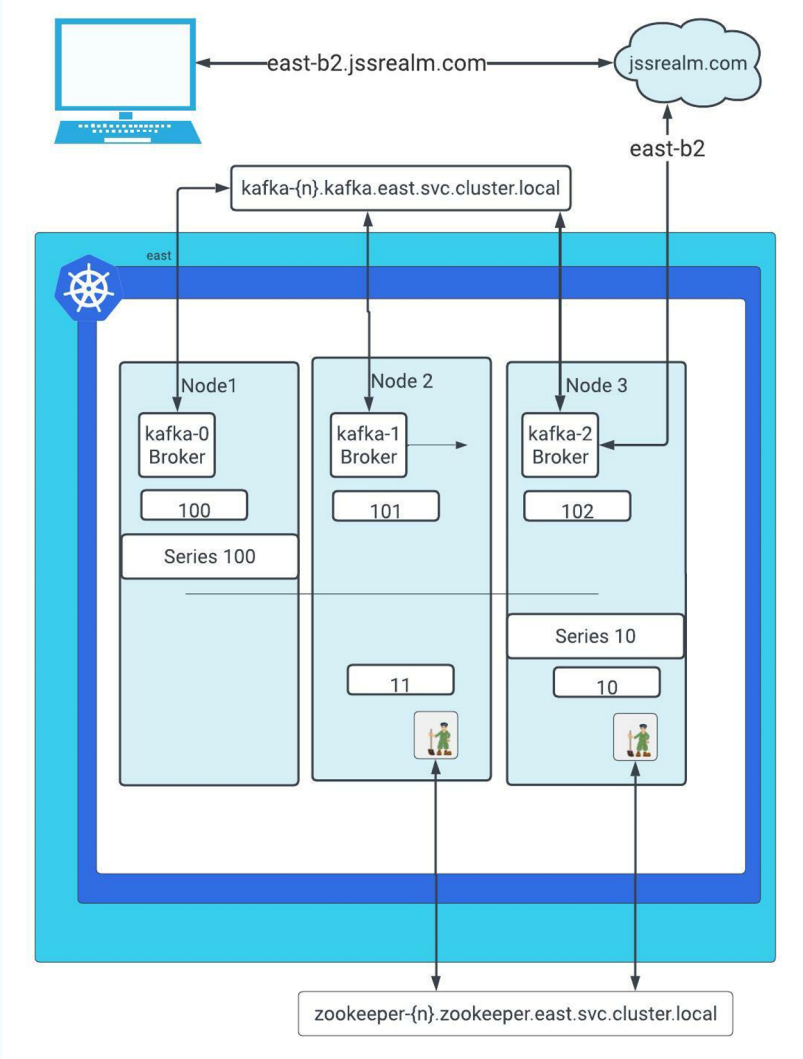
StatefulSets

- PVC claim

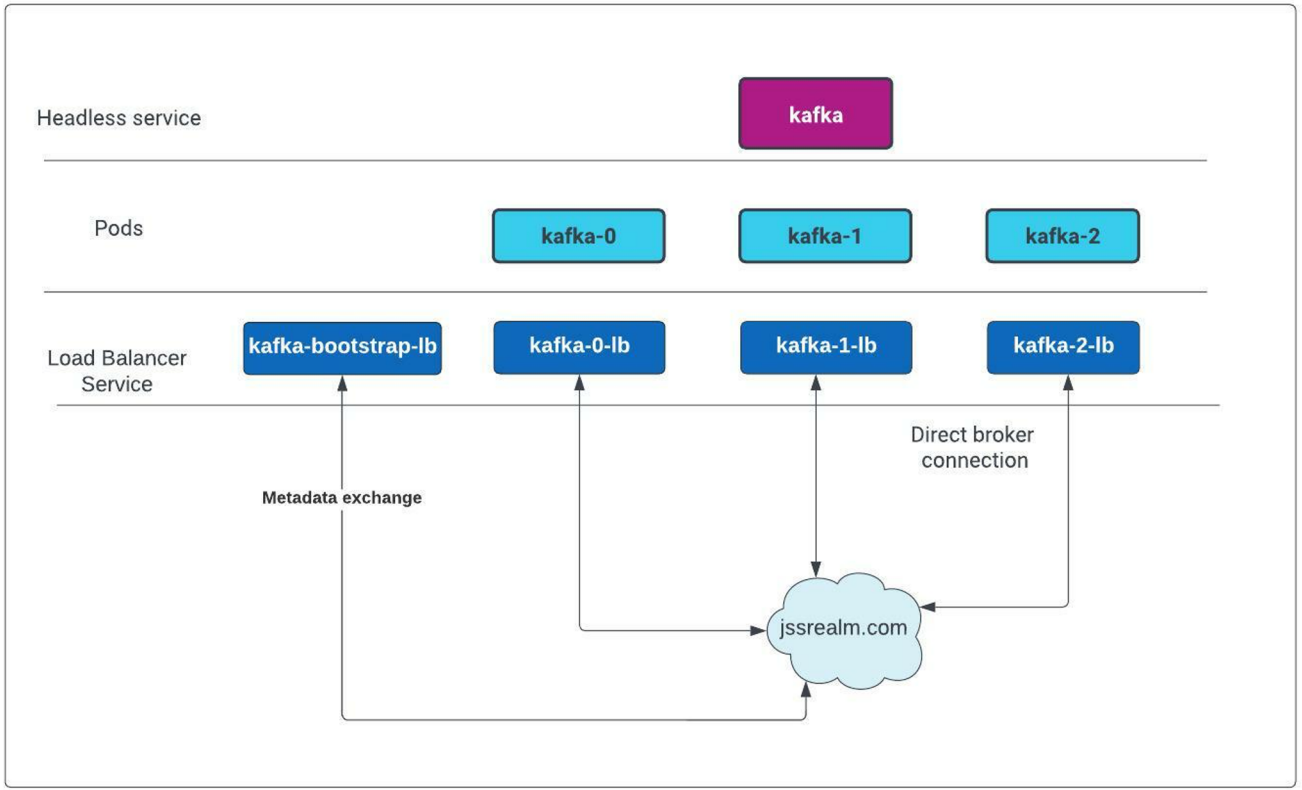
Pod Accessibility

Identifying the kafka pods

- Unique broker IDs
- Internally, each pod resolves
  kafka-{n}.kafka.east.svc.cluster.local
- Externally, broker prefixes to map to pod ordinals
  {region}-b{n}.{domain}

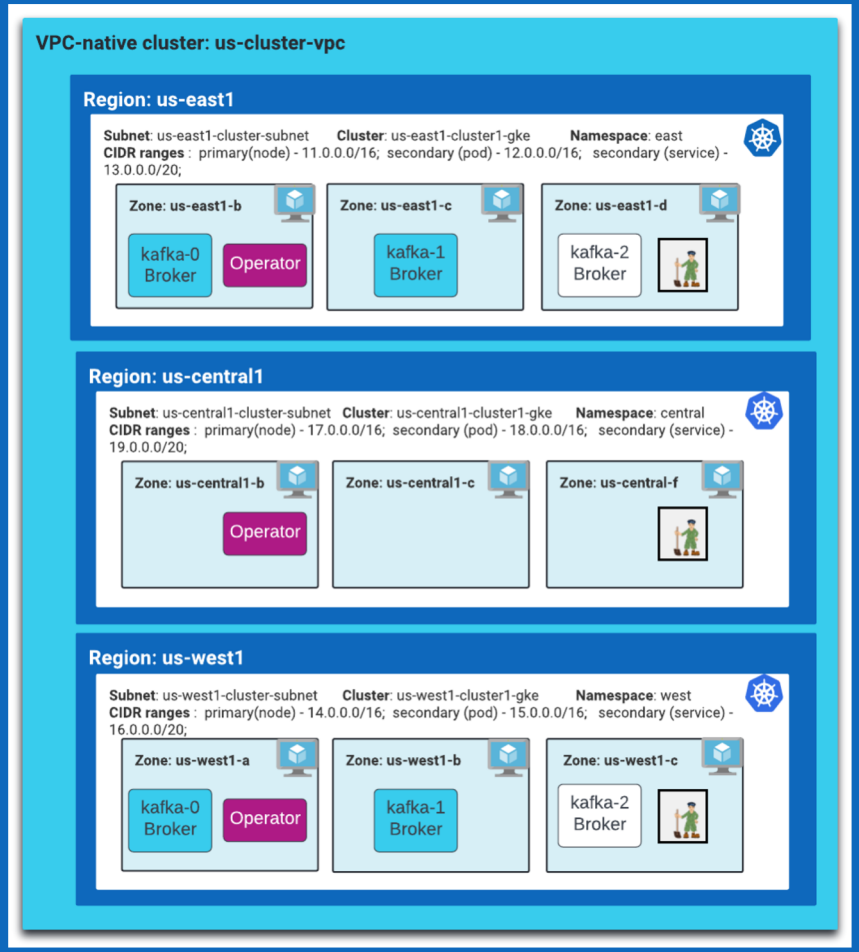


Kafka CRD Services Galore



Stretch Cluster on Kubernetes

Multi Region (Stretch) Cluster



East Cluster

- Single Kubernetes Cluster (us-east1-cluster1-gke)
- 3+ Brokers
- Single Zookeeper

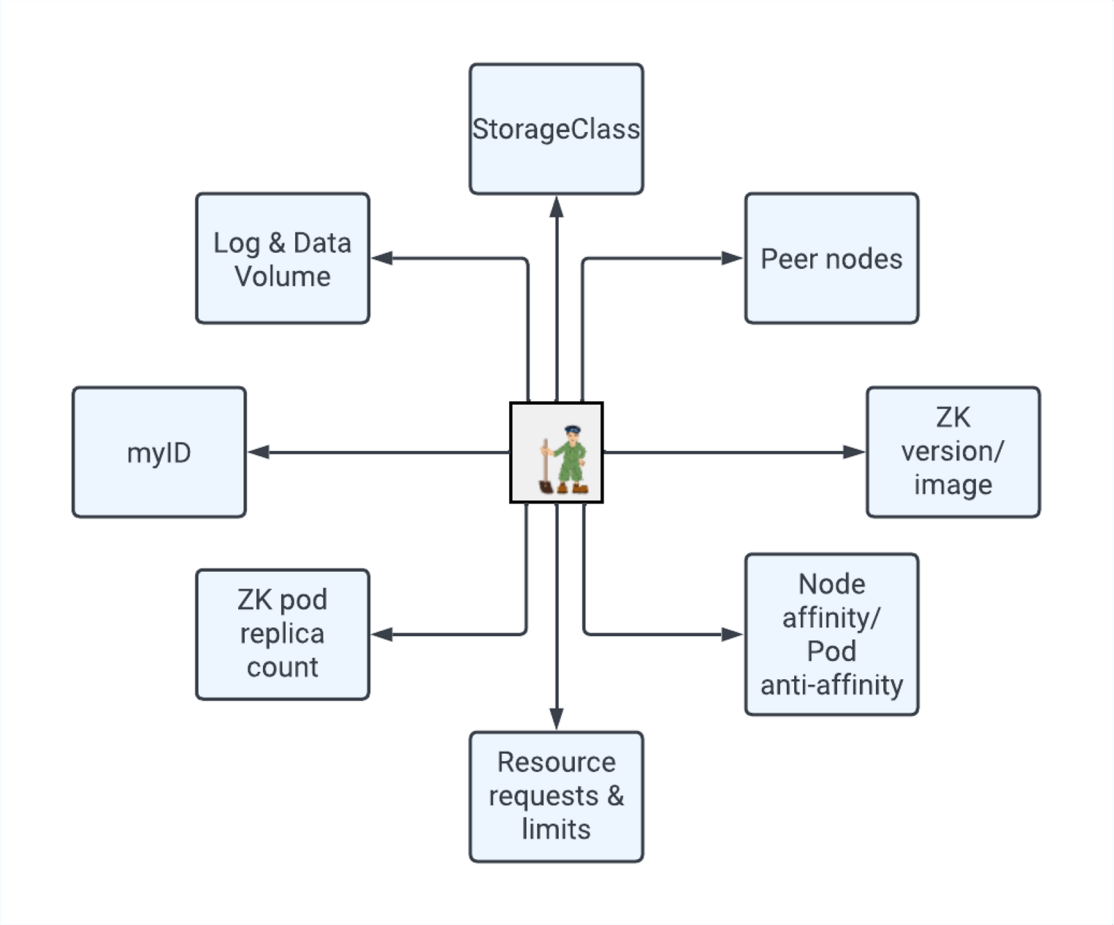
Central Cluster

- Single Kubernetes Cluster (us-central1-cluster1-gke)
- Single Zookeeper

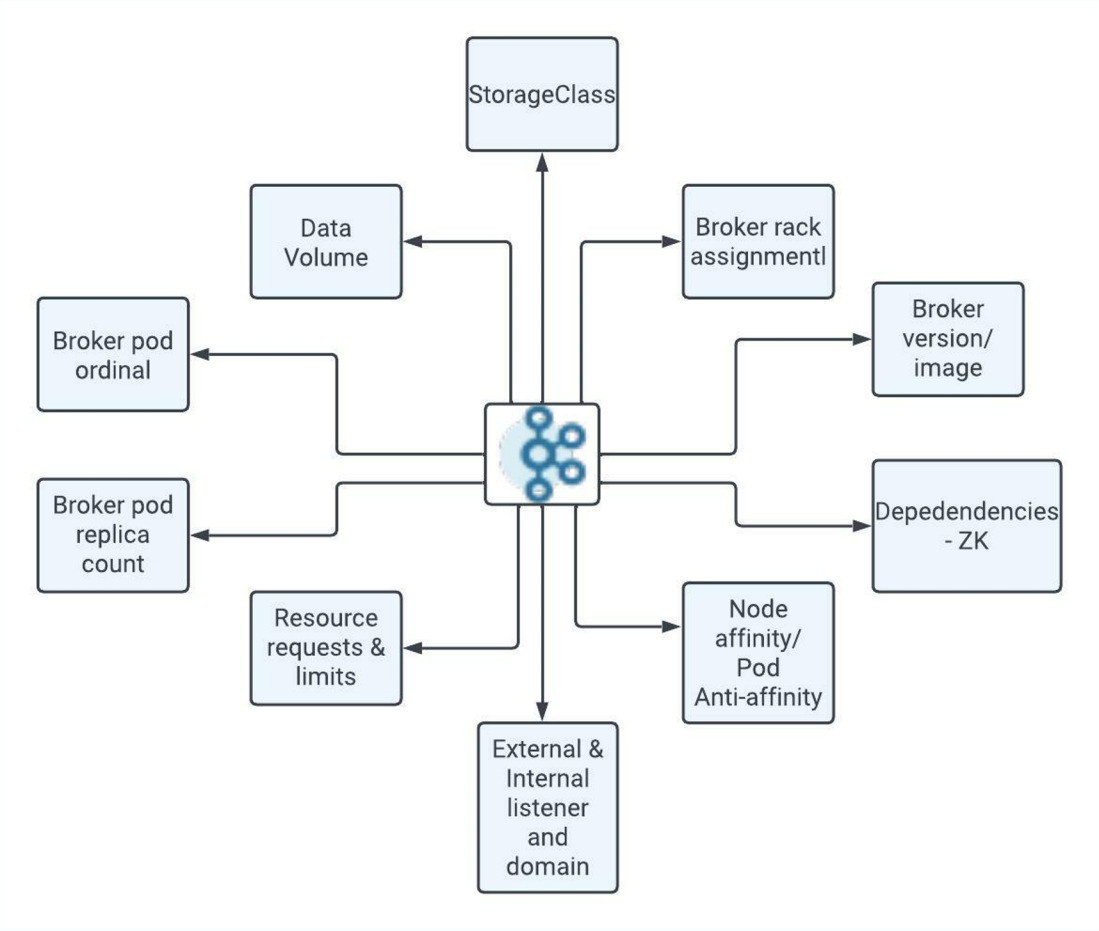
West Cluster

- Single Kubernetes Cluster (us-west1-cluster1-gke)
- 3+ Brokers
- Single Zookeeper

ZooKeeper Configuration



Kafka Configuration



Topic Replica Placement*

```
{
    "version": 2,
    "replicas": [{
            "count": 1,
            "constraints": {
                "rack": "us-east1-b"
            }},
        {   "count": 1,
            "constraints": {
                "rack": "us-east1-c"
            }},
        {   "count": 1,
            "constraints": {
                "rack": "us-west1-a"
            }},
        {   "count": 1,
            "constraints": {
                "rack": "us-west1-b"
            }}],
     "observers": [{
            "count": 1,
            "constraints": {
                "rack": "us-east1-d"
            }},
        {
            "count": 1,
            "constraints": {
                "rack": "us-west1-c"
            }}],
     "observerPromotionPolicy": "under-replicated"
}
```

- **Broker Rack Awareness**
- **Synchronous Replicas**
- **Asynchronous Observers***
- **Observer Promotion Policy***

Putting it to the Test

Testing

Pod Kill

- Deleted a pod

Node Fault

- Node VM down
- Auto-scaler off & Node VM down

Pod Failure

- Introduced pod failures with chaos-mesh

ZK Quorum Failure

- 2 ZK nodes down - disrupt quorum

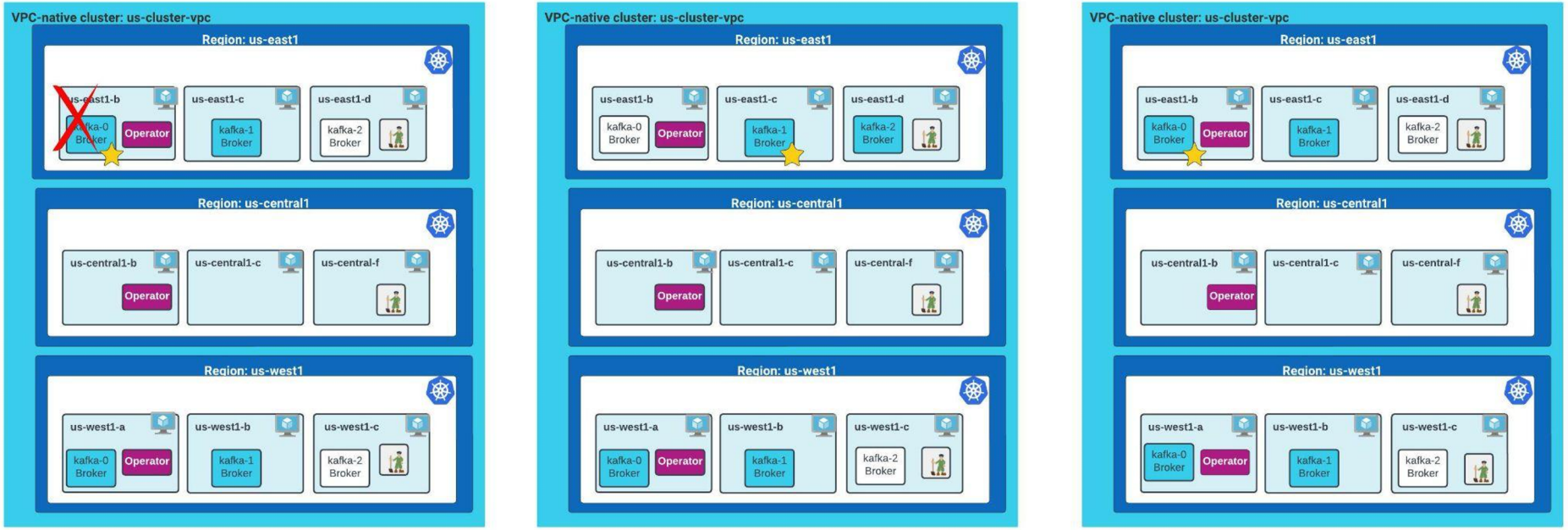
Region Down

- Producers don't stop
- Controller broker and topic leader broker move to west
- ZK west is accessible for write

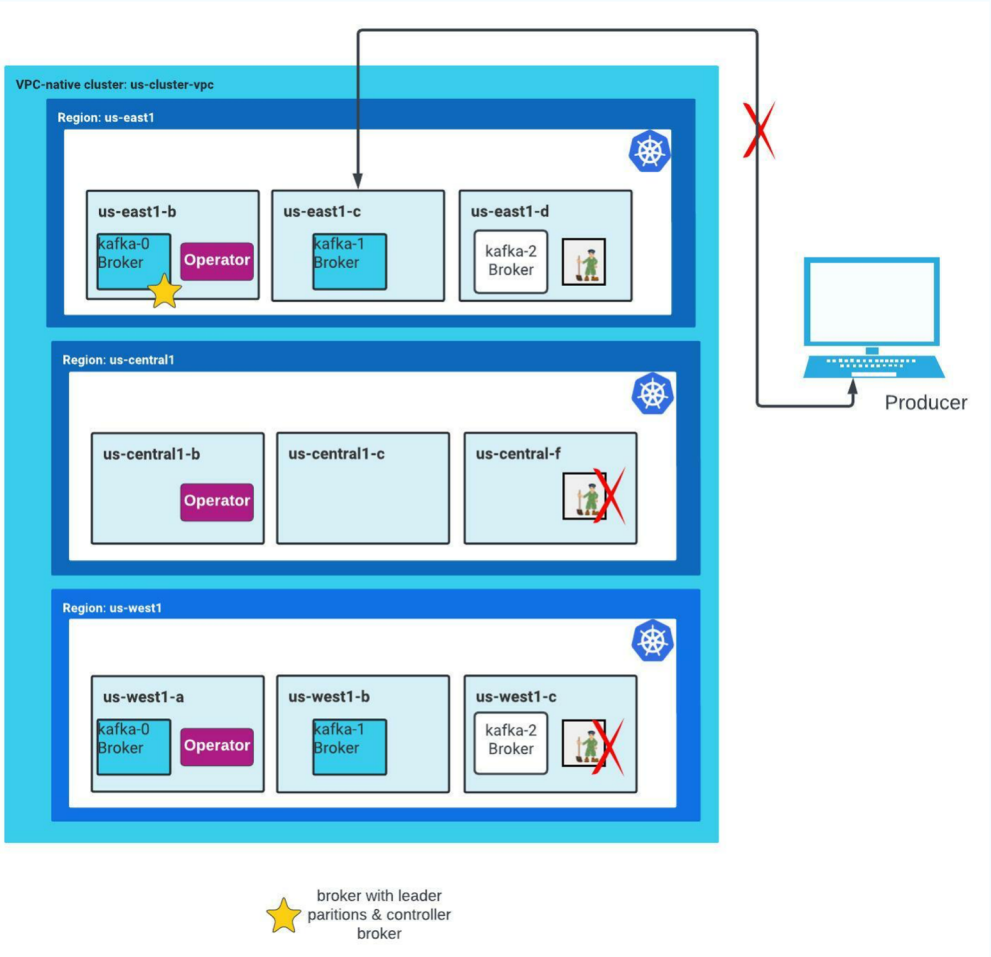
Network Fault

- Edited kube-dns stub-domain

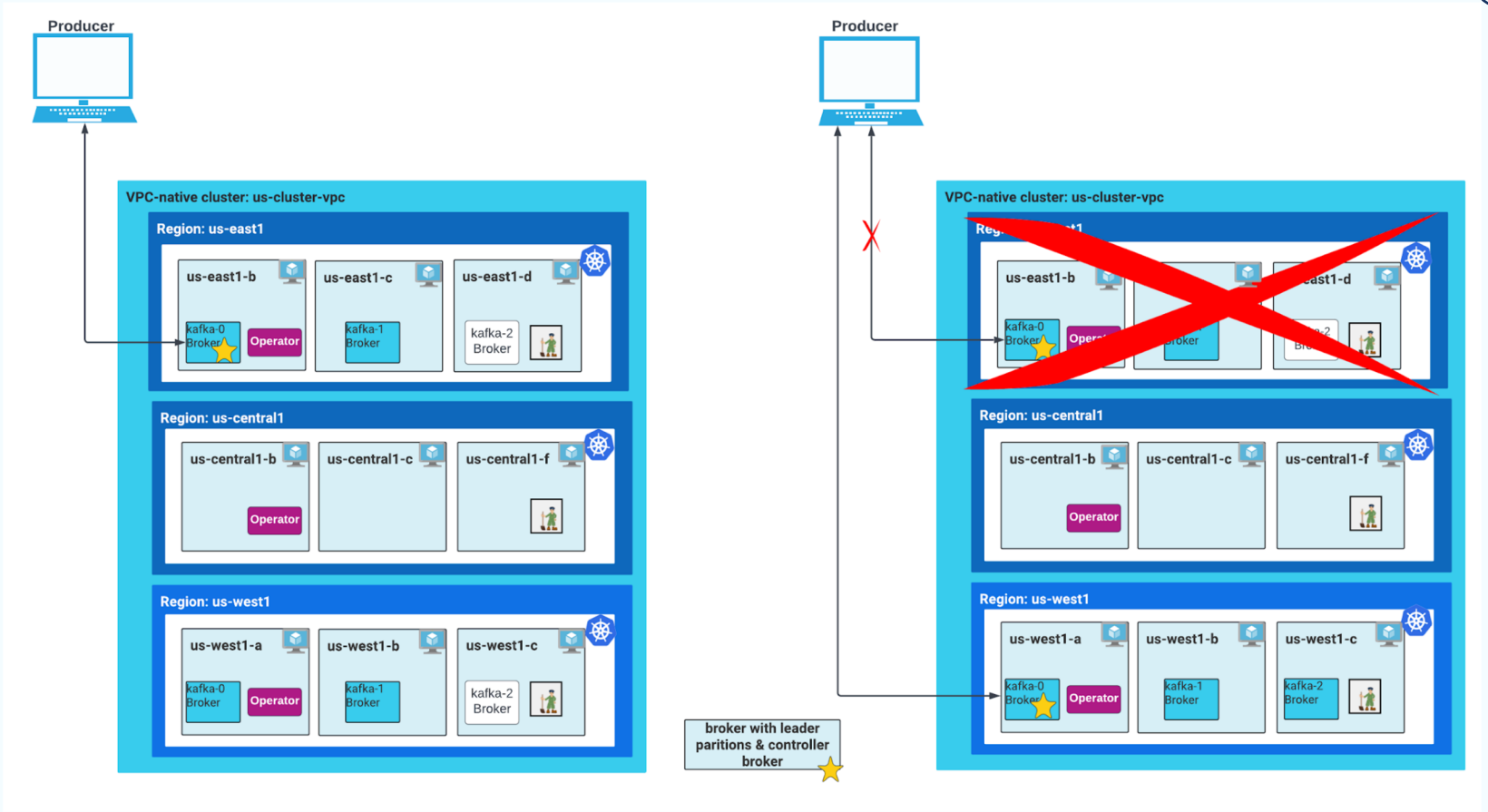
Pod Failure/Node Fault



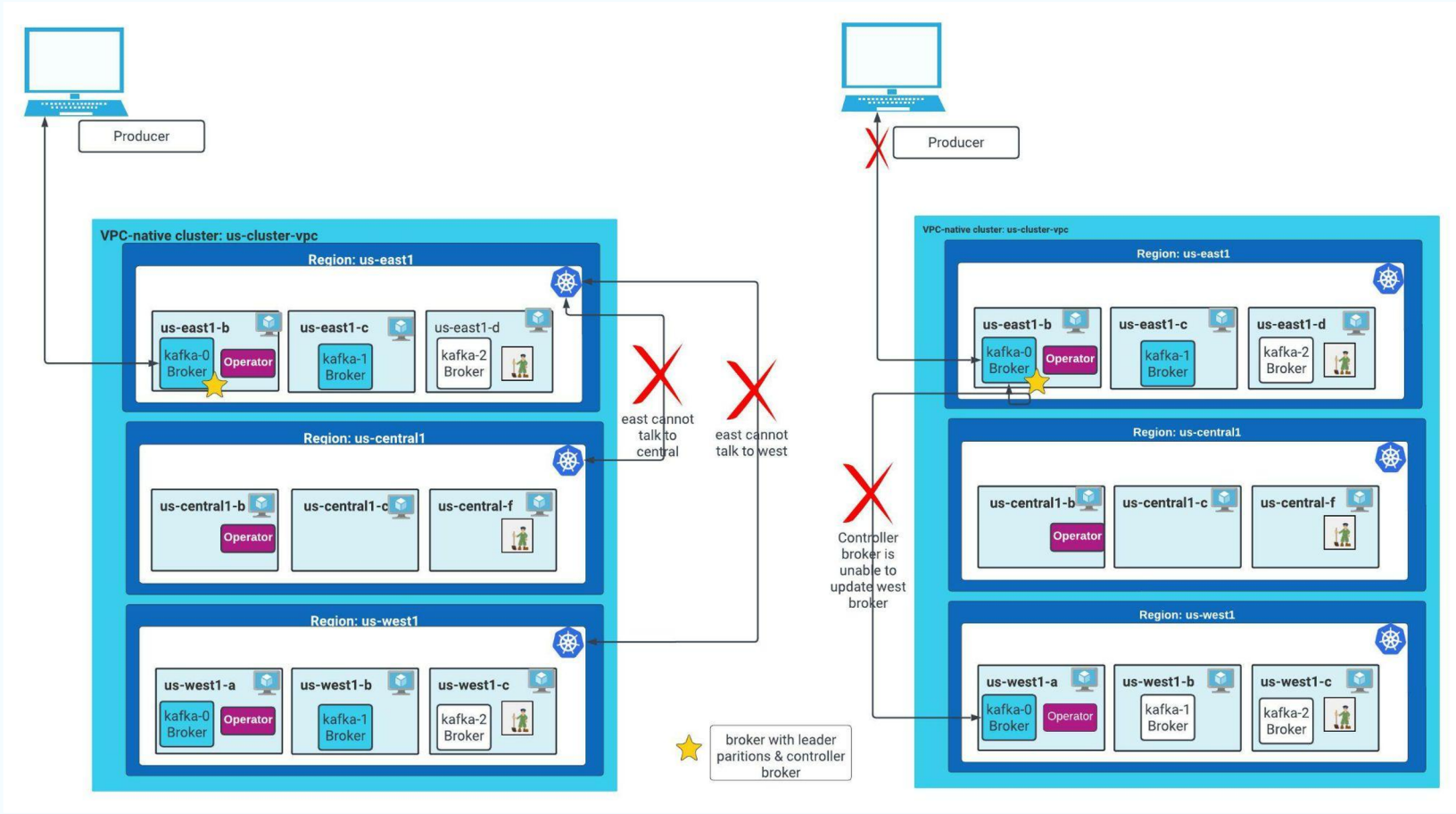
ZK Quorum Failure



Region Down


Producer
VPC-native cluster: us-cluster-vpc
Region: us-east1
us-east1-b
us-east1-c
us-east1-d
kafka-0 Broker
Operator
kafka-1 Broker
kafka-2 Broker
Region: us-central1
us-central1-b
us-central1-c
us-central1-f
Operator
Region: us-west1
us-west1-a
us-west1-b
us-west1-c
kafka-0 Broker
Operator
kafka-1 Broker
kafka-2 Broker
broker with leader paritions & controller broker
Producer
VPC-native cluster: us-cluster-vpc
us-east1-b
kafka-0 Broker
Region: us-central1
us-central1-b
us-central1-c
us-central1-f
Operator
Region: us-west1
us-west1-a
us-west1-b
us-west1-c
kafka-0 Broker
Operator
kafka-1 Broker
kafka-2 Broker

Network fault



Watching the Metrics

Wrapping it Up

Best Practices - Node to Pod Ratio

Quantity

- Use eventsizer.io to derive number of broker pods
- Adjust count to balance across AZs
- Each AZ is a rack

Capacity

- Eventsizer.io output will derive CPU and memory per pod
- Set CPU and memory limits and requests

Nodes

- Memory optimized node type
- Evaluate capacity of node based on how average load
- Enable auto scaling with average and peak range

Pod **Pod** **Nodes**

Best Practices

- Use Confluent for Kubernetes
- Choose the right storage that guarantees reliability, efficiency and speed e.g. SSD
  - Refer to ‘Building the K8s Cluster’ slide for other storage best practices
- Run health/liveness checks on:
  - Individual pod and bootstrap LBs
  - kube-system LBs
  - kube-dns service
- Use node affinity/pod anti-affinity to strategically place broker and ZK pods across AZs
- Use automation so infrastructure and CRDs can be deployed multi-region across all of your environments
- Follow best practices for tuning tcp socket buffers, replica fetcher, and clients for optimal performance with stretch clusters
- Minimum Durability Configuration
  - 2 Replicas and 2 Observers in each region
  - min.ISR=3
- Monitor everything!

- Requires separate IP CIDR ranges
- CoreDNS exposed externally
- Restricted to single K8s implementation
- CRDs may get stuck if the finalizer logic in Operator is not finishing
- Manually restart stateful set if pods are erroring - known issue
- GCP VPCs are global, subnets are regional

Achieving your Desired Resilience

Active-Passive

- RTO > 0, RPO > 0
- Replicas in one Region
- Observers in another Region
- Under-replicated AOP

Active-Active

- RTO ~ 0, RPO = 0
- 2 Replicas in each region
- 2 Observers in each region
- Under-replicated AOP

Demo and Q&A

References

1. https://assets.confluent.io/m/69c5ce7aff462f44/original/20180619-WP-Recommendations_for_Deploying_Apache_Kafka_on_Kubernetes.pdf
2. https://cloud.google.com/kubernetes-engine/docs/how-to/alias-ips
3. https://learn.hashicorp.com/tutorials/terraform/gke
4. https://chaos-mesh.org/docs/
5. https://docs.confluent.io/operator/current/overview.html
6. https://docs.confluent.io/platform/current/multi-dc-deployments/multi-region.html
7. https://www.confluent.io/en-gb/events/kafka-summit-americas-2021/a-tale-of-2-n-data-centers-tuning-apache-kafka-clusters-to-combat-latency/

CONFLUENT
Developer

Your Apache Kafka® journey begins here

developer.confluent.io


CONFLUENT