Stretch Cluster

What is Stretch Cluster ?

Kafka when run as a cluster of one or more servers that can span across multiple Datacenters or cloud regions, such clusters are called as stretch cluster.

A *stretched cluster* is a single logical cluster comprising several physical ones. Replicas are evenly distributed between physical clusters using the rack awareness feature of Kafka, while client applications are unaware of multiple clusters.

Why Multi-Region/DC?

- **Region failure disaster recovery** - When a region experiences a region-wide failure, a multi-region architecture allows failover to a different region to decrease RTO and RPO.
- **Global operations with minimized latency** - End users in one locality, for example in North America, will connect to a North American region, whereas end users in another locality, for example in Europe, will connect to a European region. This ensures each end user will experience the lowest possible latency.
- **Data sovereignty** - End-user data within certain sovereignties must never leave the sovereignty or must be aggregated or anonymized when leaving the sovereignty.
- **Data governance** - Similar to data sovereignty, some applications require that certain data never leave a certain region or network but perhaps not because of a sovereignty's laws.
- **Network isolation** - Some applications use Confluent Platform within or through a network DMZ to increase network security.

Features of Stretch Cluster :

- This is NOT a multi-cluster: it is just one cluster which has broker servers on different Datacenters.
- Stretch clusters are intended to protect the Kafka cluster from failure in the event an entire datacenter fails. This is done by installing a single Kafka cluster across multiple Datacenters.
- Strong consistency due to the synchronous data replication between clusters.
- Kafka's normal replication mechanism is used, as usual, to keep all brokers in the cluster in-sync.

When to use stretch cluster?

- For applications which need ideal scenario RPO=0 and RTO=0.
- Use a stretched cluster when the application only needs to survive a single region failure at a time.
- The most common environment where stretched clusters are deployed is in public cloud regions with three availability zones (AZ).
- Used for applications located within a single country or smaller continent.

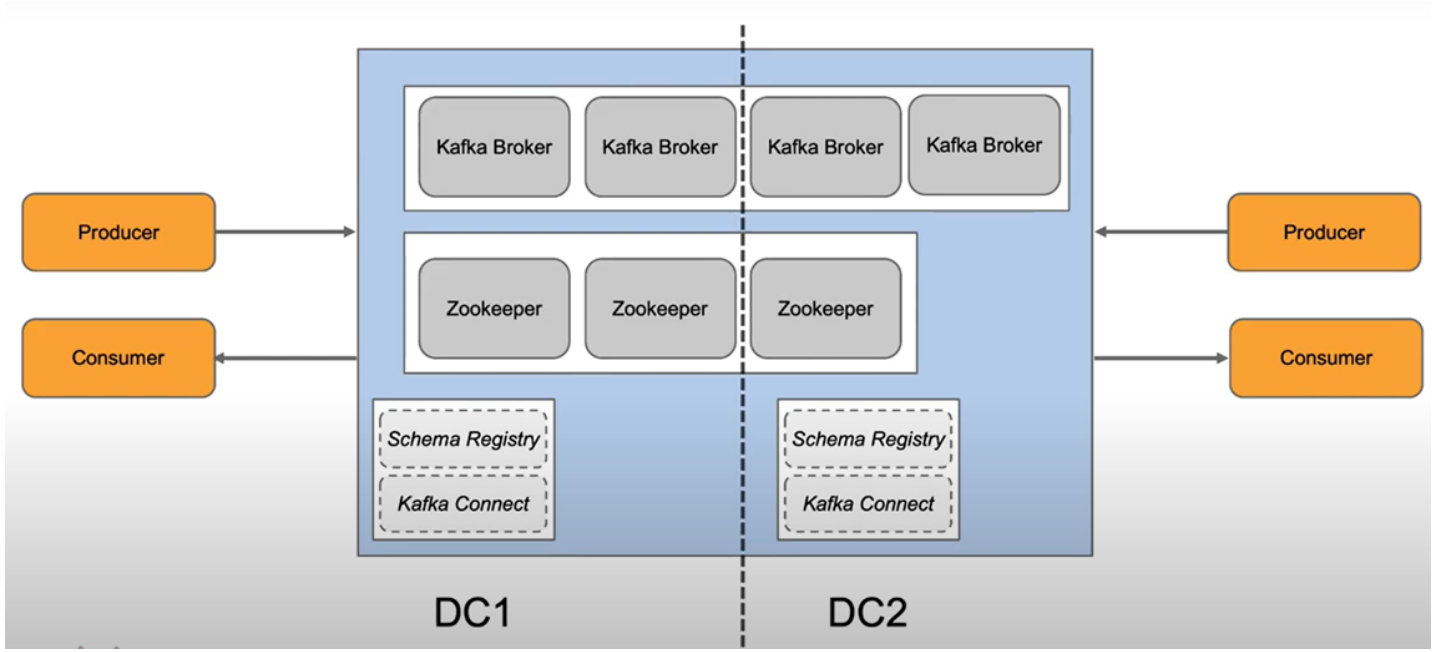
Advantages of stretch cluster

- Strong consistency due to the synchronous data replication between clusters. 0 data loss as In case of a single cluster failure, other ones continue to operate with no downtime.
- Ordering of Kafka messages is preserved across datacenters.
- Consumer offsets are preserved.
- In event of a disaster in a datacenter, new leaders are automatically elected in the other datacenter for the topics configured for synchronous replication, and applications proceed without interruption, achieving very low RTOs and RPO=0 for those topics.
- Unawareness of multiple clusters for client applications. Consumers can leverage data locality for reading Kafka data, which means better performance and lower cost.
- High Availability Zero Data loss **and** Zero downtime Automatic client failover works well in cloud (with different availability reagion).
- No External tools (like mirror maker needed).

Disadvantages of stretch cluster

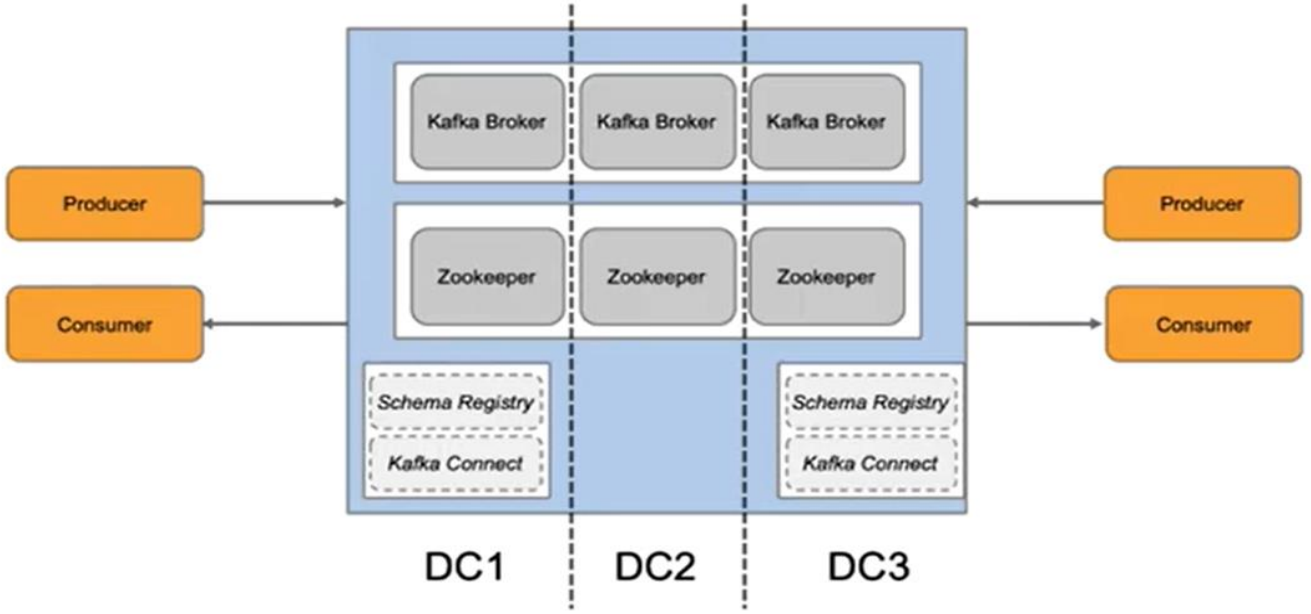
- Cluster failure is still a disaster. Any issue related to cluster there will be no cluster available for streaming data.
- Provided only in confluent edition
- Complex to configure and operate especially when it fails. Complex to maintain topic topology and overhead for operations teams. Not recommended.
- 2.5 datacenter topology or a tie-breaker data center is required.
- This model features high latency due to synchronous replication between clusters. So, it's recommended to use such deployment only for clusters with high network bandwidth. End to end should be <10s
- Within the stretched cluster model, minimum three clusters are required. Maintenance of three cluster is again an operational overhead.
- On-Premises is not recommended.
- System incurs additional fixed overhead as Zookeeper nodes negotiate consistency on whatever metadata changes might happen in the cluster. Particularly, the write performance of Zookeeper decreases rapidly as latency between the members of the quorum increases.

Two data center stretch cluster



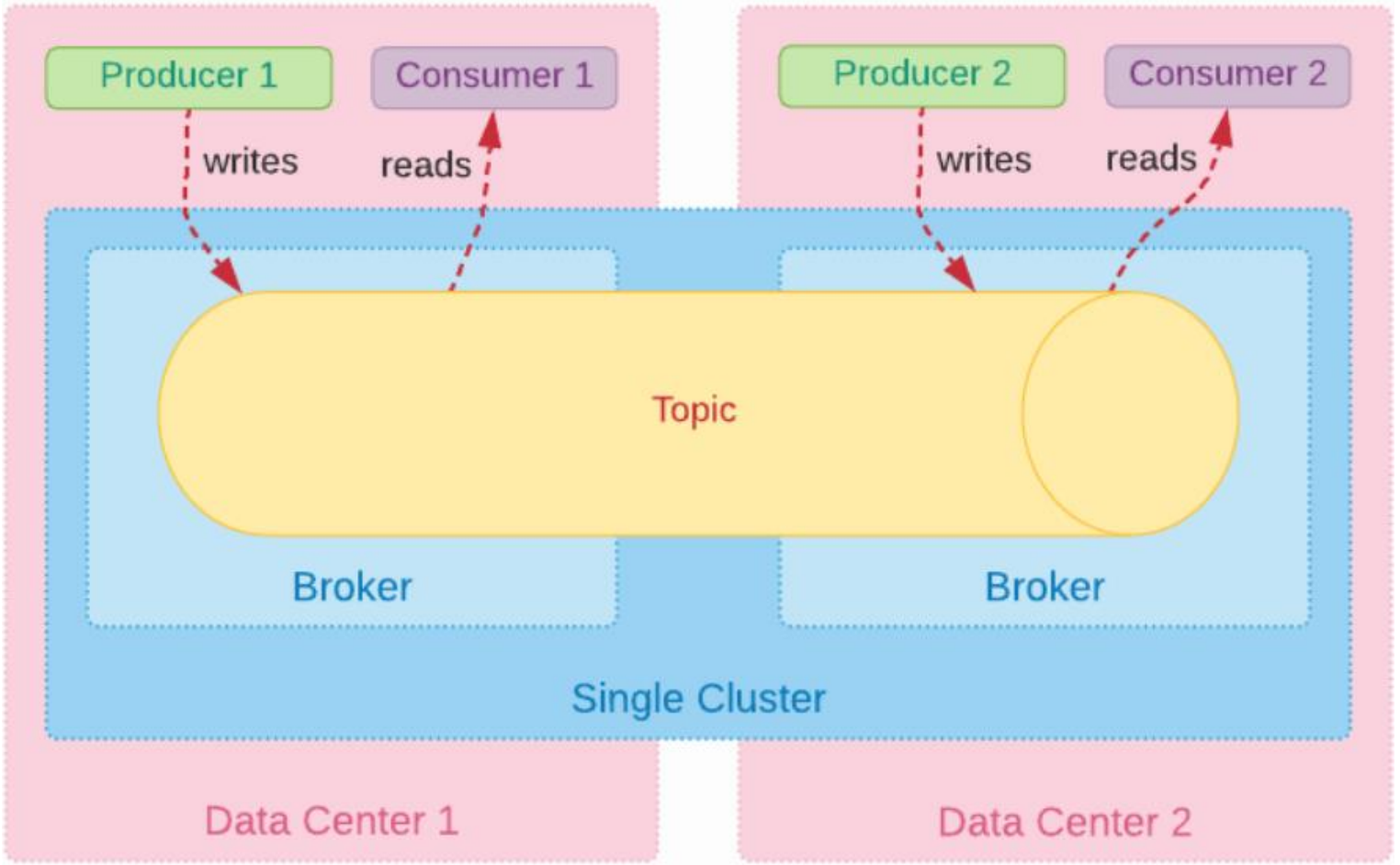
- High availability (Survives DC outage).
- Zero data loss OR zero downtime.
- Automatic client fail-over.
- Stopgap solution for on premise (if only 2 DCs available). 2.5DC as workaround.
- Requires "good latency".
- Quorum in 2DCs not possible.
- Complex to configure and operate.

Three data center stretch cluster

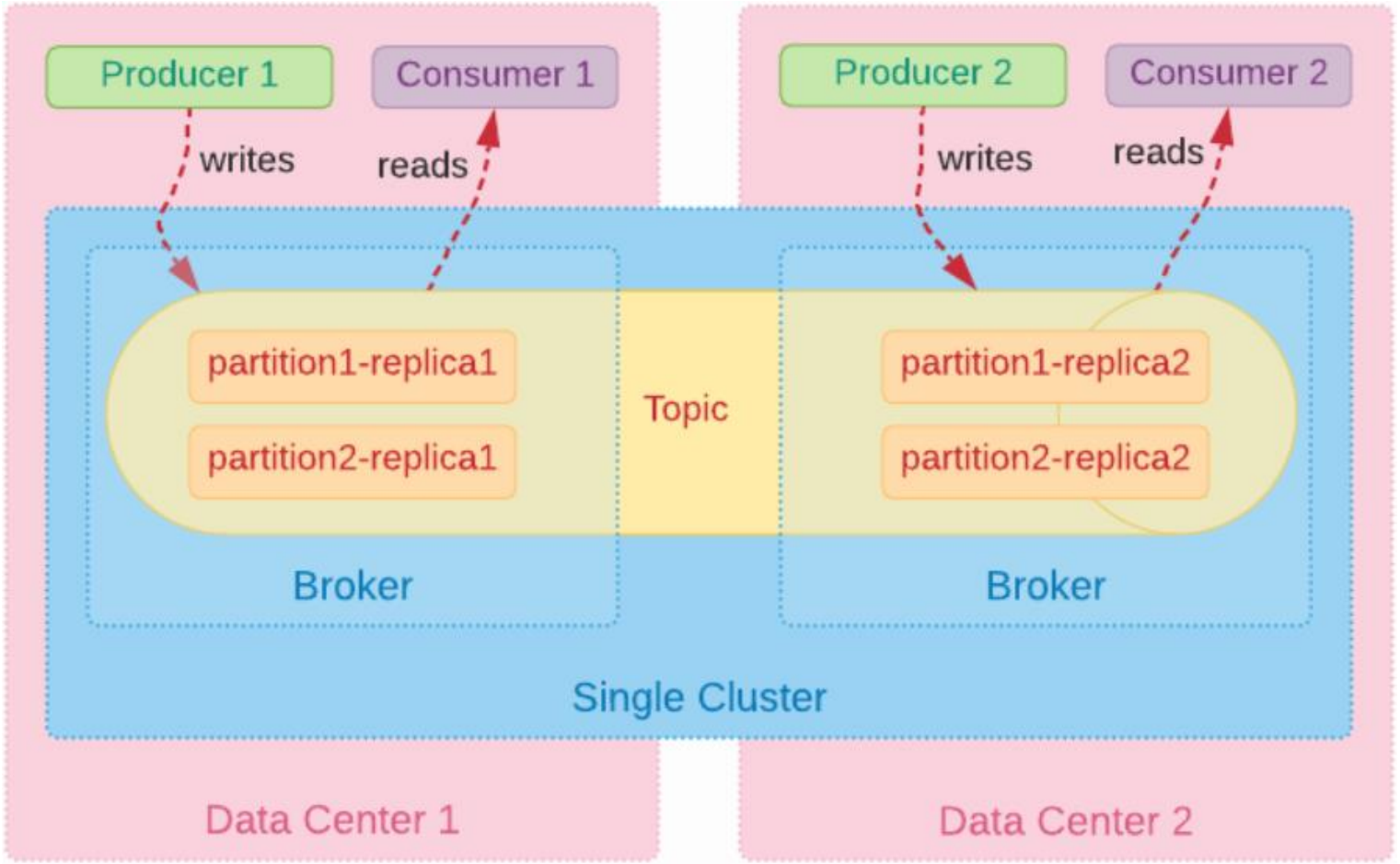


- High availability (Survives DC outages)
- Zero data loss AND zero downtime
- Automatic client fail-over
- Works well in cloud and on premise.
- No External tools needed.
- Requires low latency.
- Requires 3 DCs (quorum/split brain)
- Complex to configure and operate

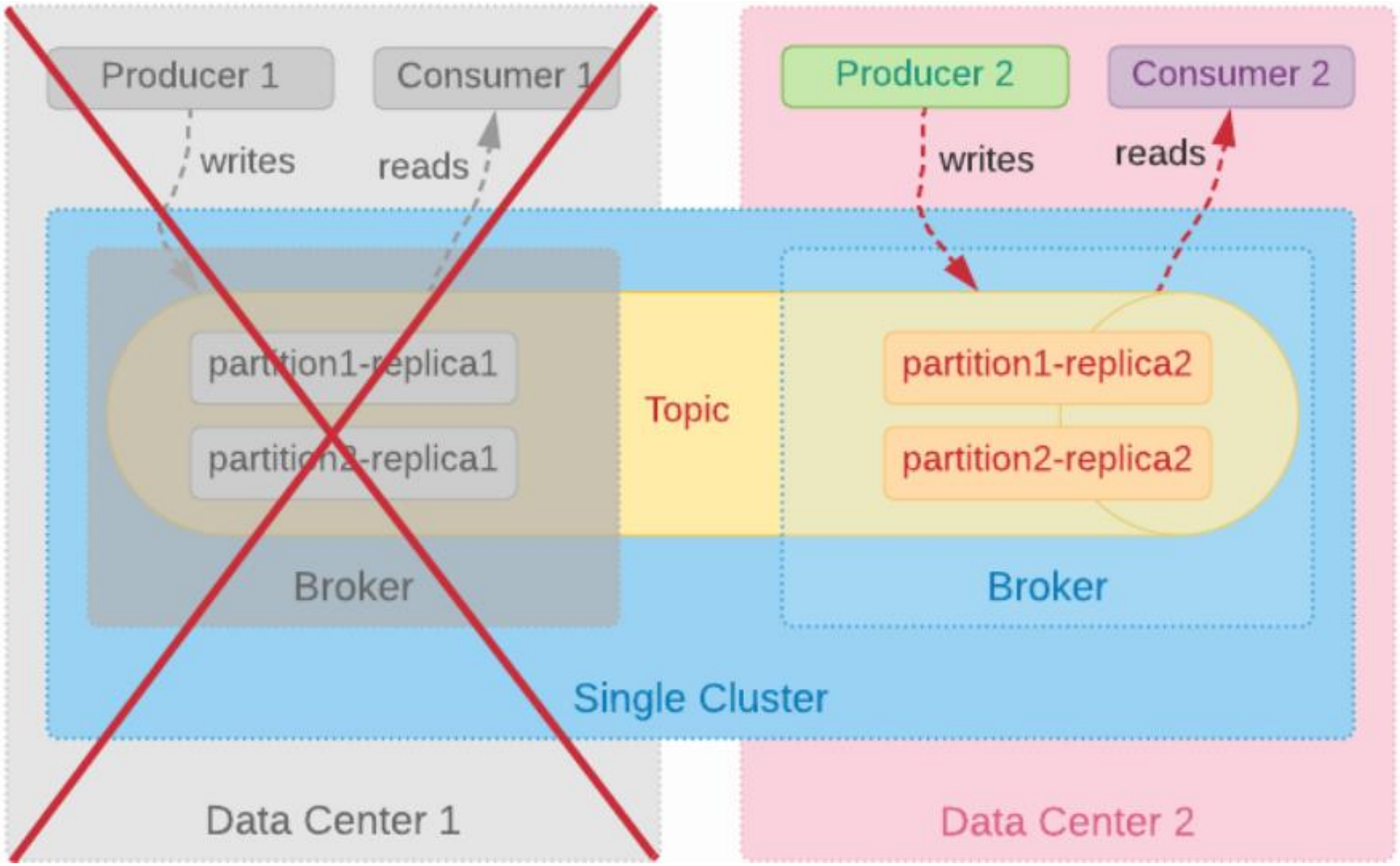
Stretch Cluster



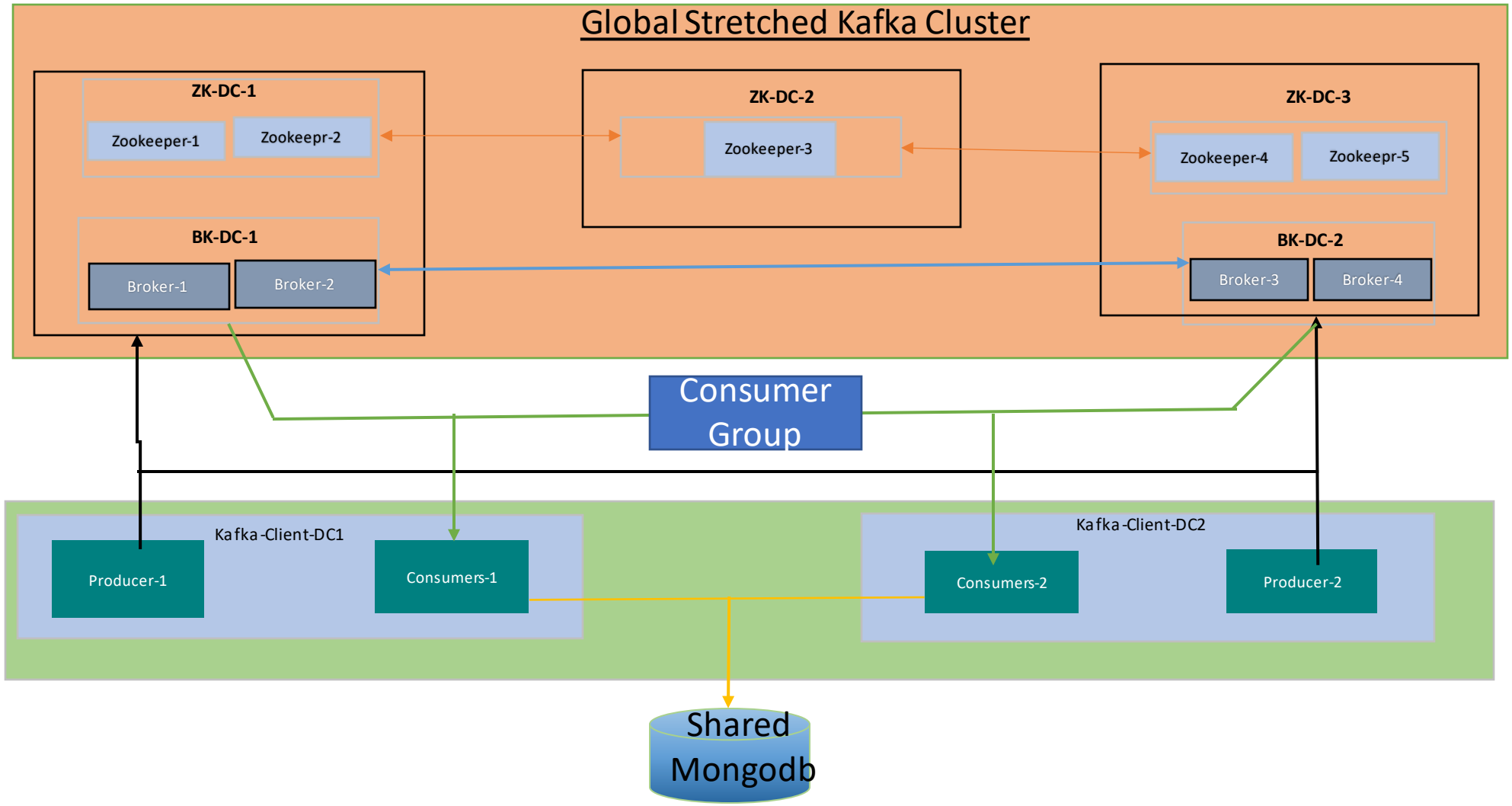
Stretch Cluster – Rack Awareness



Stretch Cluster – Disaster in one DC.

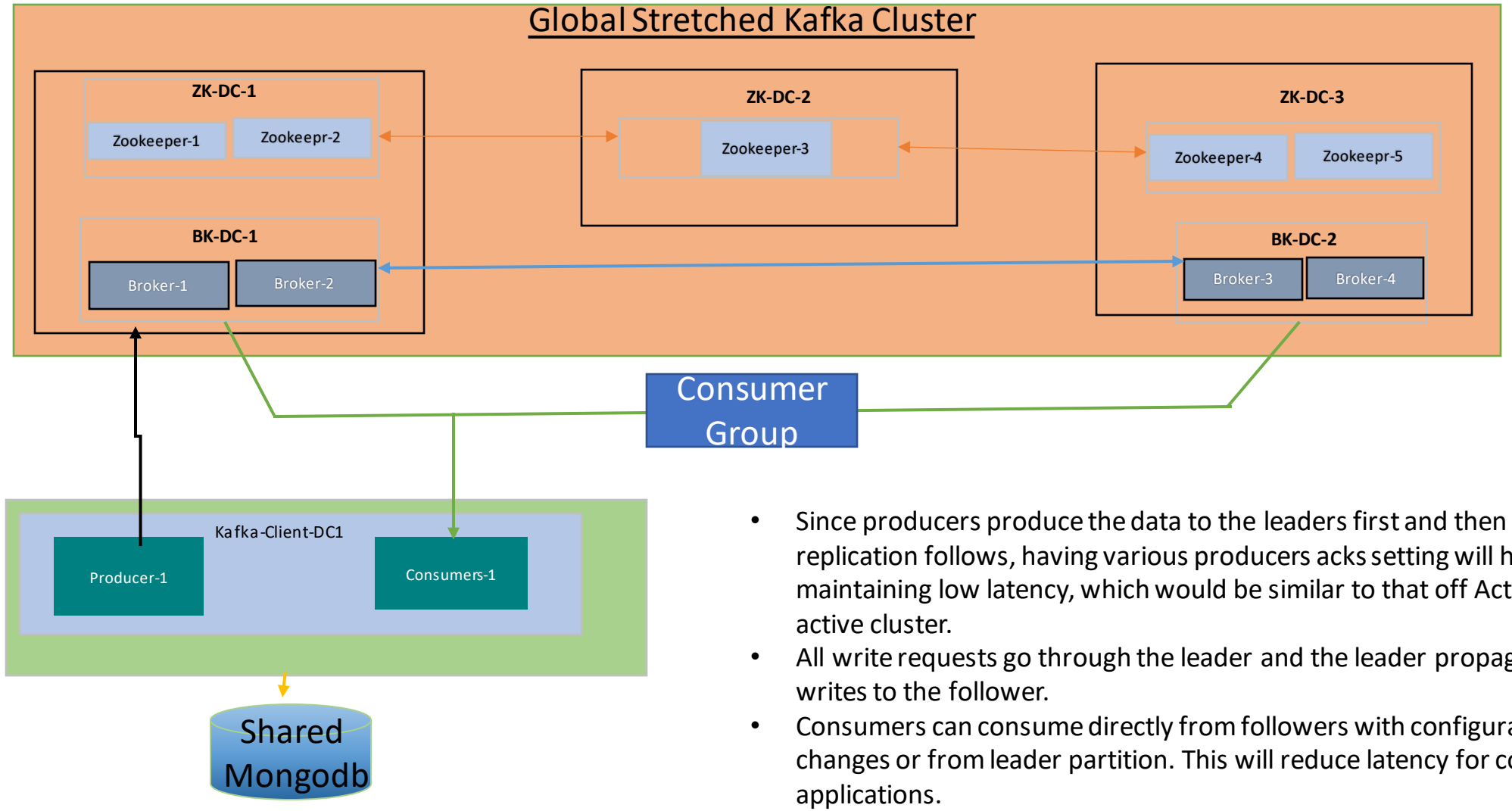


Proposed Stretch cluster setup



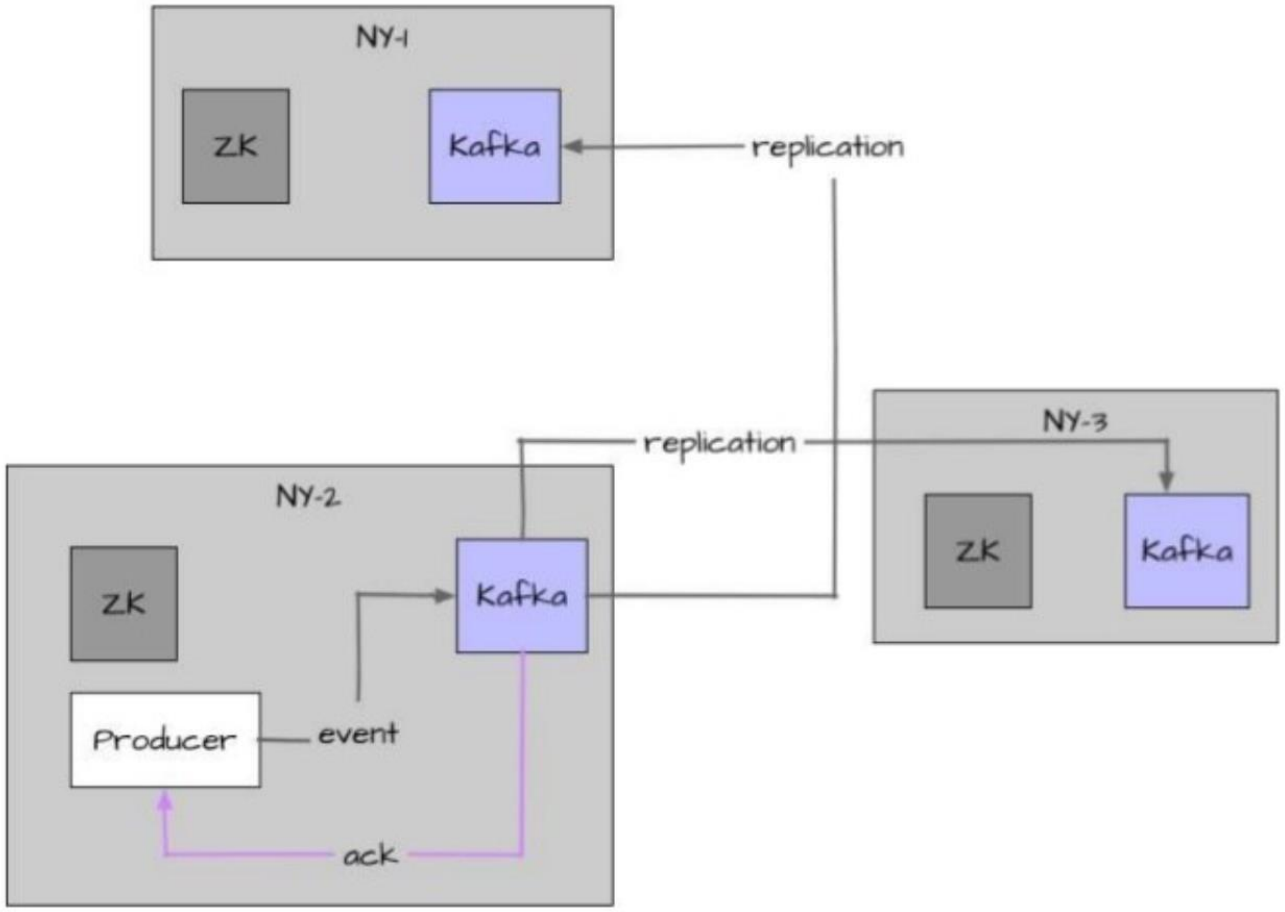
Network latency – We have to make the in-sync replica between one among the DCs
Consumer Group – Same consumer group can consume data across the DCs
In Sync replica – Any network failure in the second DC might lose the message (Check the commit time, any such facility available Kafka can commit after a definite time)
Write operation might be slow
Support one DC failure

Proposed Stretch cluster setup with topics having leaders on mainly DC1 data center



- Since producers produce the data to the leaders first and then replication follows, having various producers acks setting will help in maintaining low latency, which would be similar to that off Active-active cluster.
- All write requests go through the leader and the leader propagates the writes to the follower.
- Consumers can consume directly from followers with configuration changes or from leader partition. This will reduce latency for consumer applications.
- Operational additional overhead to maintain the topics leaders in one single DC.

Simplified proposed architecture



Proposed Stretch cluster setup with topics having leaders on mainly DC1 data center (Synchronous replication)

- To reduce Latency further and increase throughput with **acks = all**, (synchronous replication)
- Design topics to have all the leaders aligned to one particular Data region.
- All producers to be placed in DC1 alone which already gives an added advantage of lower latency.
- Consumers present in only in DC1 and consuming from any replica/leader.
- Broker.rack awareness is enabled.

Parameter name (component)	Stretched cluster value	Advantages
acks (Producer)	acks=all	Ordering of messages
min.insync.replicas (Broker)	3	Data durability
Replication Factor	3	Lower latency among acks = all configuration for producer
Delivery.timeout.ms (Producer)	Integer.MAX_VALUE	Lower latency for consumers.
Idempotence (producer)	true	End to End Latency high
max.in.flight.requests.per.connection	5	Exactly once writes to partitions
Retries (producer)	>1	Reliability gurantee
client.rack (consumer - RackAwareReplicaSelector)	Set	No Duplication
transactional.id (producer)	_UNIQUE-ID_	

Proposed Stretch cluster setup with topics having leaders on mainly DC1 data center (Synchronous replication)

- To reduce Latency further and increase throughput with **acks = all**, (Synchronous replication)
- Design topics to have all the leaders aligned to one particular Data region.
- All producers to be placed in DC1 alone which already gives an added advantage of lower latency.
- Consumers present in only in DC1 and consuming from any leader/replica.
- Broker.rack awareness is enabled.

Parameter name (component)	Stretched cluster value	Advantages
acks (Producer)	acks=all	Ordering of messages
min.insync.replicas (Broker)	3	Data durability
Replication Factor	3	Lower latency among acks = all configuration for producer
Delivery.timeout.ms (Producer)	Integer.MAX_VALUE	Lower latency for consumers.
Idempotence (producer)	false	End to End Latency high
max.in.flight.requests.per.connection	1	Exactly once writes to partitions
Retries (producer)	0	Reliability guarantee
client.rack (consumer - RackAwareReplicaSelector)	Set	No Duplication
transactional.id (producer)	_UNIQUE-ID_	

Proposed Stretch cluster setup with topics having leaders on mainly DC1 data center (Asynchronous replication)

- To reduce Latency further and increase throughput with **acks = 1/0**, (Asynchronous replication)
- Design topics to have all the leaders aligned to one particular Data region.
- All producers to be placed in DC1 alone which already gives an added advantage of lower latency.
- Consumers present in only in DC1 and consuming from any replica/leader.
- Broker.rack awareness is enabled.

Parameter name (component)	Stretched cluster value	Advantages
acks (Producer)	acks=1/0	Lower latency among acks = all configuration for producer
min.insync.replicas (Broker)	3	Lower latency for consumers.
Replication Factor	3	End to End Latency slightly lower
Delivery.timeout.ms (Producer)	Integer.MAX_VALUE	
Idempotence (producer)	NA	
max.in.flight.requests.per.connection	1	
Retries (producer)	>1	
client.rack (consumer - RackAwareReplicaSelector)	Set	
transactional.id (producer)	_UNIQUE-ID_	

Stretch Cluster additional requirements :

- Topics to have leader replicas to stay in their preferred DC after a broker or a rack failure.
- Producers need to set acknowledgment level to ensure that data is replicated in both the DCs.
- To satisfy these requirements, there is a need to have a process running which will reduce **min.ISR** if a DC failure occurs. Also this process will set leader replicas in the preferred DC for a topic in broker or rack failure scenario**.**

Implementation Challenges :

- Replacement of leader replicas of a partition in their preferred DC after a broker or a rack failure**.**
- Producers and consumers may connect across DCs so it will impact performance.
- If across DC connection will happened there will be more chances of rebalance of consumers. It will cause duplicate data entry in target which need to be deal with application.
- If across DC connection will happened there will be chances of more timeout which trigger retry of producer which again multiple duplicates data write in topic.

Solution not recommended, why?

Solution

- 2 ZK nodes in each DC and “observer” somewhere else
- 3 ZK nodes in each DC and manually reconfigure quorum on failover
- One Cluster.

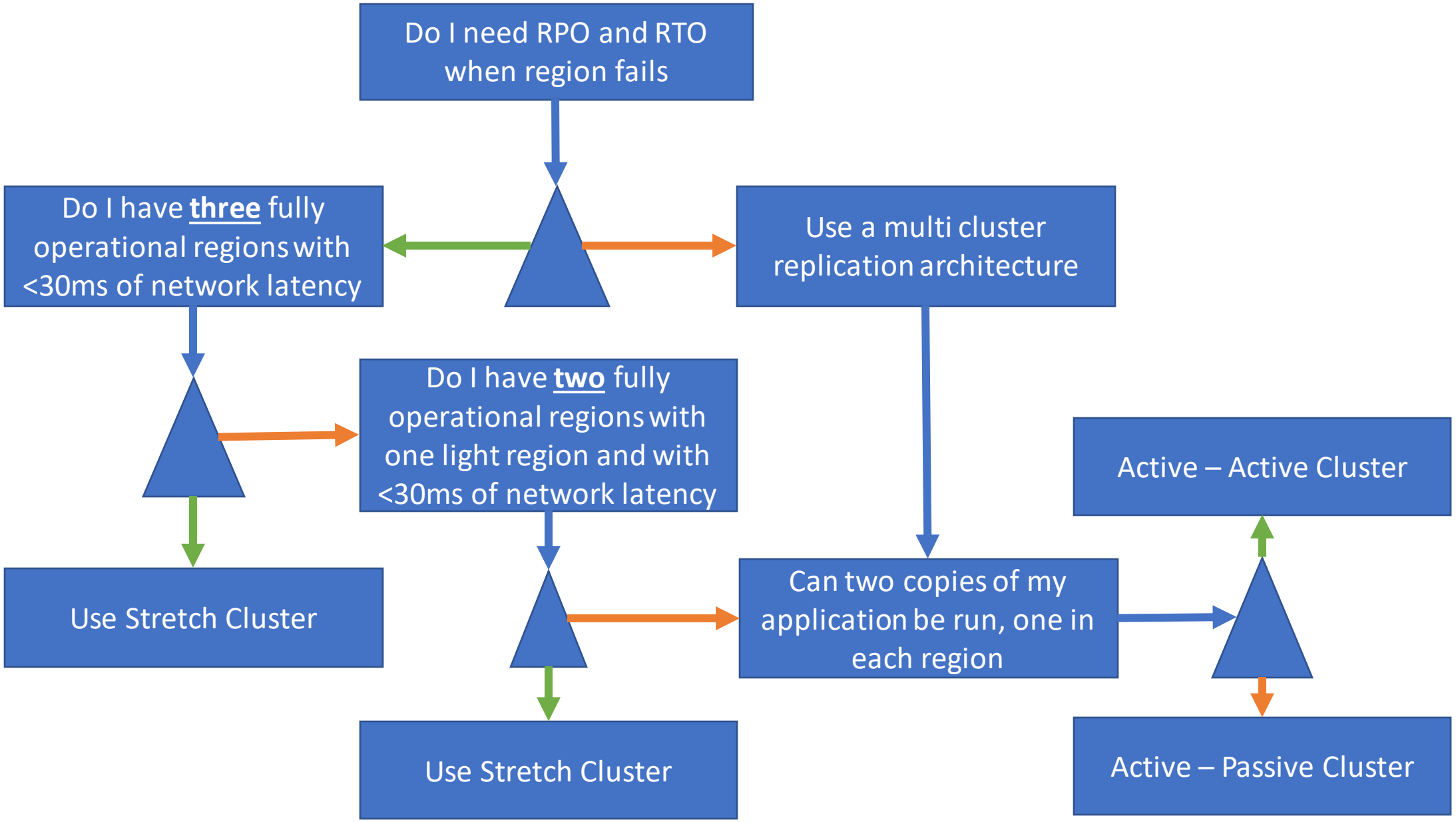
Details

- Not a tried and tested option and only available in Confluent version
- We might lose ZK updates during failover. Requires manual intervention.
- Not safe enough

Comparison between existing and proposed stretch cluster

Type of Cluster/Agreements	Stretched Cluster	Active-Active Cluster
Advantages	Strong consistency due to the synchronous data replication between clusters.	Resources are fully utilized in both clusters.
	Zero downtime and Zero data loss in case of a single cluster failure.	Zero downtime in case of a single cluster failure.
	In case of a single cluster failure, other ones continue to operate with no downtime.	Network bandwidth between clusters doesn't affect performance.
	Unawareness of multiple clusters for client applications.	Easy to maintain and operate.
	Cluster resources are utilized to the full extent.	Clusters/data independent of each other data center.
		Low latency and strong consistency.
		Isolation, Tuning, Convenience and organization structure.
		Both clusters are equivalent.
Disadvantages	Cluster failure is still a disaster. Any issue related to cluster there will be no cluster available for streaming data.	Eventual consistency due to asynchronous mirroring between clusters. Bidirectional mirroring required.
	Various implementation challenges.	
	Within the stretched cluster model, minimum three clusters are required.	Possible data loss in case of a cluster failure due to asynchronous mirroring.
	Complex to configure and operate especially when it fails.	Awareness of multiple clusters for client applications
	Three data centers required.	Monitor replication and avoid loops.
	This model features high latency due to synchronous replication between clusters.	
	Operational overhead.	
	Once zookeeper is removed in latest version of kafka, the stretch cluster design might need a change.	
	Costly infrastructure	

Which architecture to be considered?



Main Links Referred

- Architecture Patterns for Multi-Region Clusters
- Youtube - Architecture patterns for distributed and global apache kafka
- Prodcast by TIM Berglund
- Kafka-stretch-clusters-By-Sonam
- Multi-cluster-deployment-options-for-apache-kafka-pros-and-cons
- Multi-Region Clusters with Confluent Platform 5.4

THANK YOU