Availability of Kafka - Beyond the Brokers

Andrew Borley

Emma Humber

Apache Kafka

Kafka Summit Europe 2021

High availability

Kafka has guarantees around the number of server failures a cluster can tolerate

What if the environment becomes unavailable?

Many layers: Applications, Kafka, OS, Network, Storage

Design into the system

Constraints

What guarantees do you need

- Consistency, availability, performance, manual intervention

What resources do you have

- Datacenters, network, cost

Consistency OR availability

Stretch Clusters

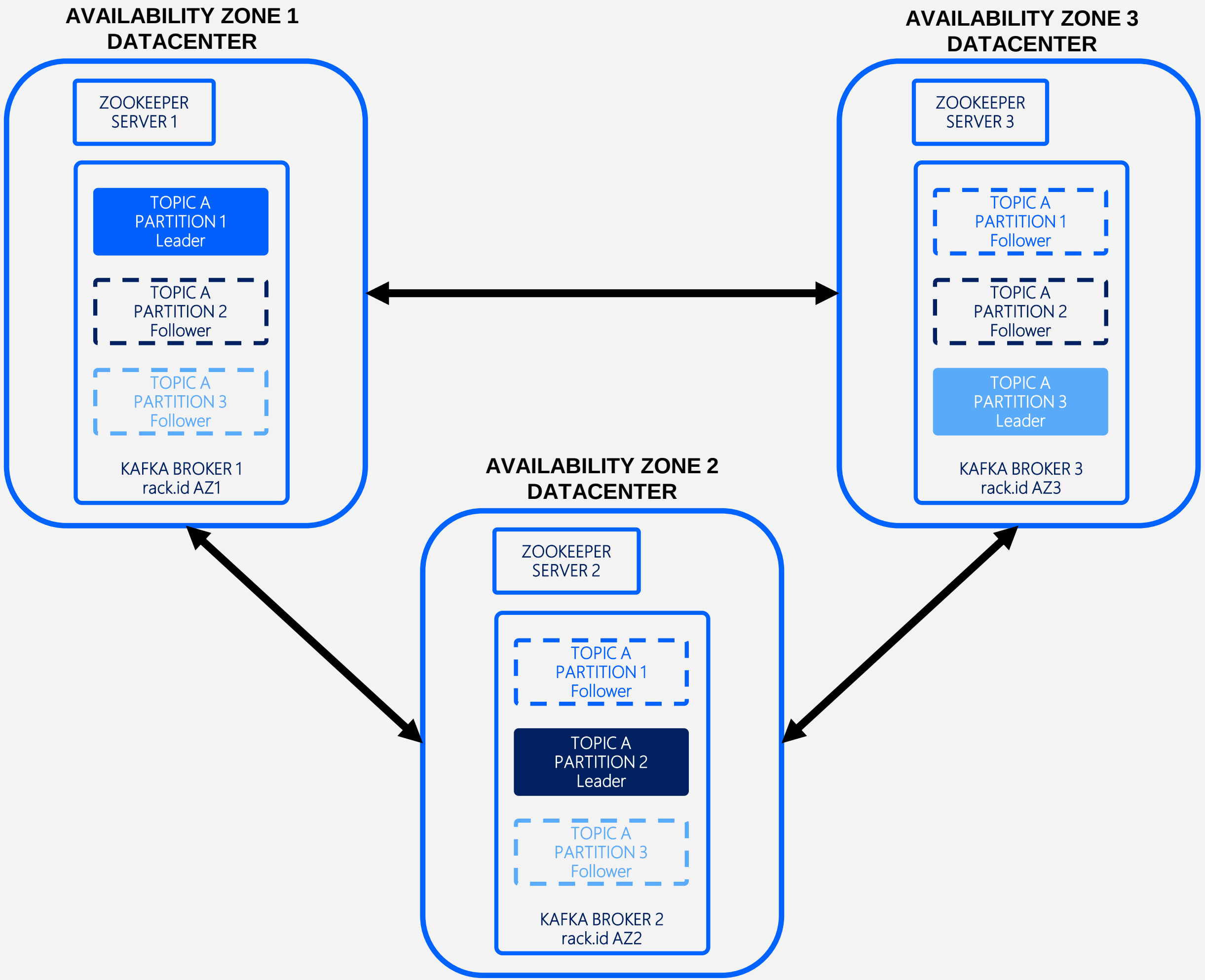
Availability zones

Set of isolated infrastructure

- Compute, storage and network connectivity + associated power and cooling
- Limits the blast radius of an infrastructure problem

A datacenter, or a failure domain within a datacenter

Geographic regions (eg Central Europe) often support multiple availability zones



AVAILABILITY ZONE 1 DATACENTER

ZOOKEEPER SERVER 1

TOPIC A PARTITION 1 Leader

TOPIC A PARTITION 2 Follower

TOPIC A PARTITION 3 Follower

KAFKA BROKER 1
rack.id AZ1

AVAILABILITY ZONE 3 DATACENTER

ZOOKEEPER SERVER 3

TOPIC A PARTITION 1 Follower

TOPIC A PARTITION 2 Follower

TOPIC A PARTITION 3 Leader

KAFKA BROKER 3
rack.id AZ3

AVAILABILITY ZONE 2 DATACENTER

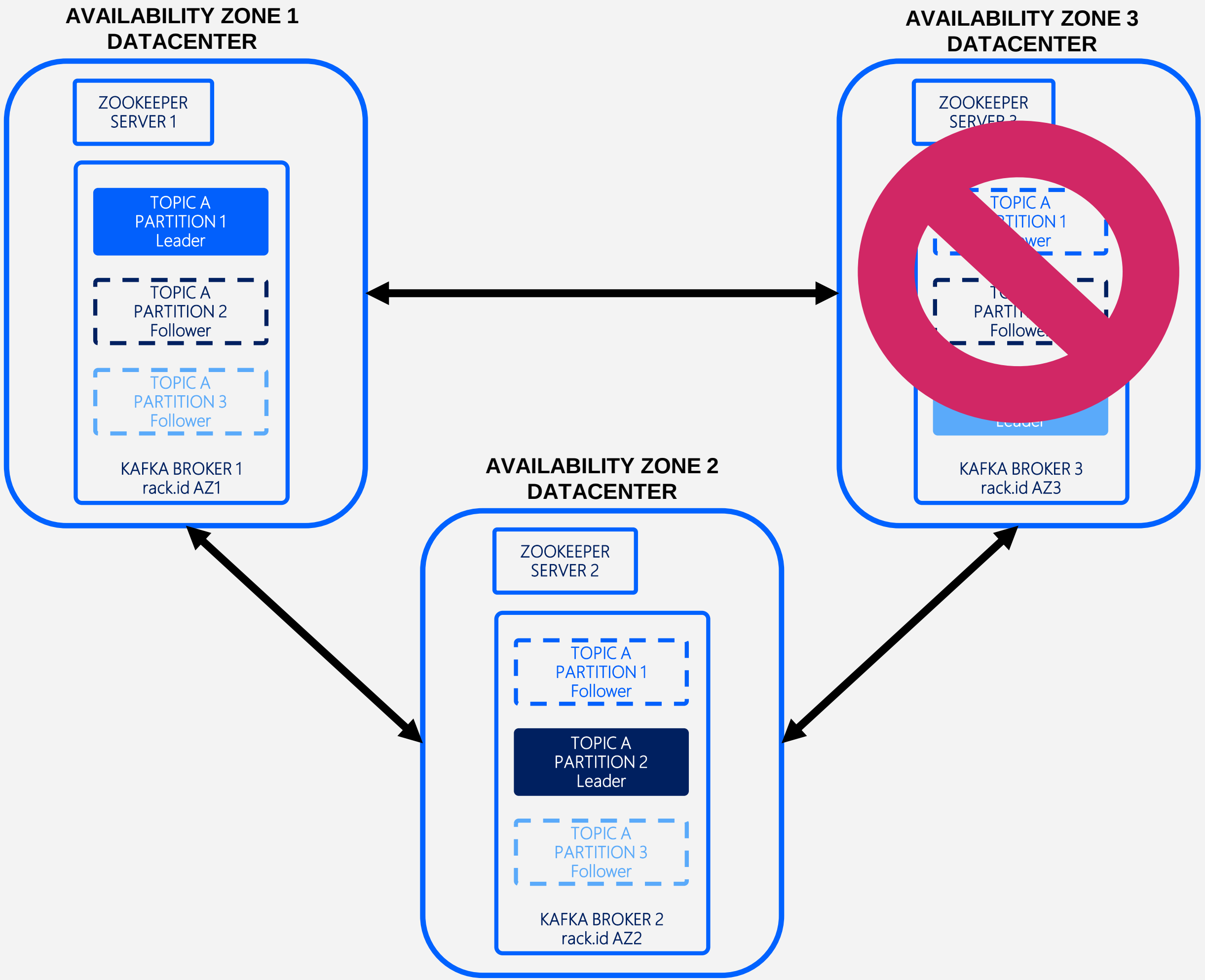
ZOOKEEPER SERVER 2

TOPIC A PARTITION 1 Follower

TOPIC A PARTITION 2 Leader

TOPIC A PARTITION 3 Follower

KAFKA BROKER 2
rack.id AZ2



AVAILABILITY ZONE 1
DATACENTER

ZOOKEEPER
SERVER 1

TOPIC A
PARTITION 1
Leader

TOPIC A
PARTITION 2
Follower

TOPIC A
PARTITION 3
Follower

KAFKA BROKER 1
rack.id AZ1

AVAILABILITY ZONE 3
DATACENTER

ZOOKEEPER
SERVER 3

TOPIC A
RTITION 1
wer

TO
PARTI
Followe

Leader

KAFKA BROKER 3
rack.id AZ3

AVAILABILITY ZONE 2
DATACENTER

ZOOKEEPER
SERVER 2

TOPIC A
PARTITION 1
Follower

TOPIC A
PARTITION 2
Leader

TOPIC A
PARTITION 3
Follower

KAFKA BROKER 2
rack.id AZ2

Configuration

Zone aware platforms automatically set up clusters in multiple availability zones

- Each Kubernetes node assigned a zone and labelled with `failure-domain.beta.kubernetes.io/zone`

Kafka's `rack.id : zone`

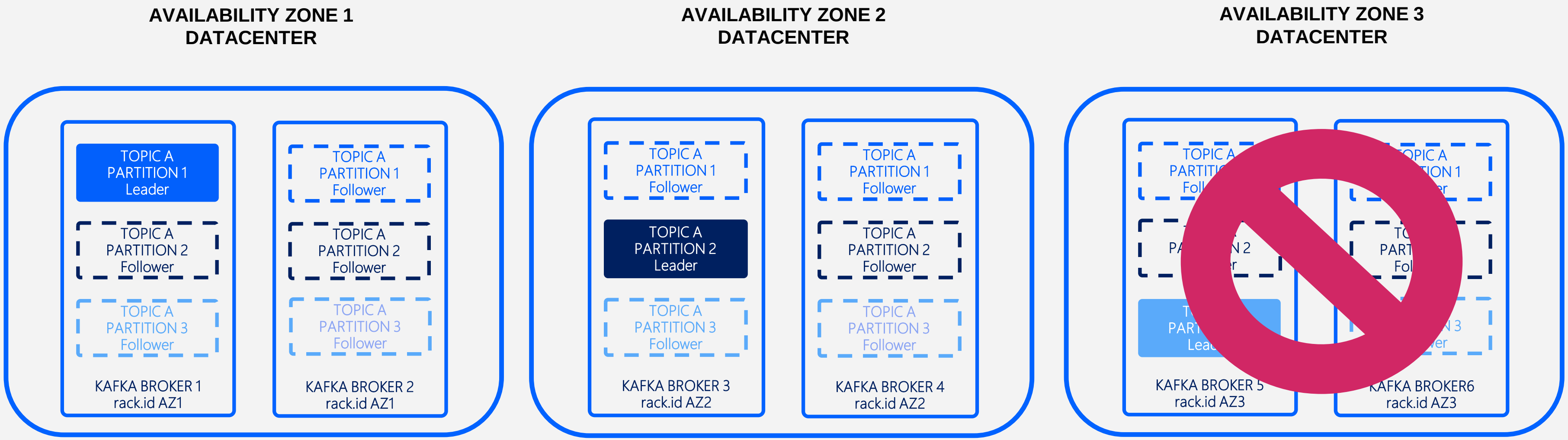
Low latency a MUST

Look at timeouts and client configuration

`zookeeper.connection.timeout.ms`

`replica.lag.time.max.ms`

`acks=all`


AVAILABILITY ZONE 1
DATACENTER
AVAILABILITY ZONE 2
DATACENTER
AVAILABILITY ZONE 3
DATACENTER
TOPIC A PARTITION 1 Leader
TOPIC A PARTITION 2 Follower
TOPIC A PARTITION 3 Follower
KAFKA BROKER 1 rack.id AZ1
TOPIC A PARTITION 1 Follower
TOPIC A PARTITION 2 Follower
TOPIC A PARTITION 3 Follower
KAFKA BROKER 2 rack.id AZ1
TOPIC A PARTITION 1 Follower
TOPIC A PARTITION 2 Leader
TOPIC A PARTITION 3 Follower
KAFKA BROKER 3 rack.id AZ2
TOPIC A PARTITION 1 Follower
TOPIC A PARTITION 2 Follower
TOPIC A PARTITION 3 Follower
KAFKA BROKER 4 rack.id AZ2
KAFKA BROKER 5 rack.id AZ3
KAFKA BROKER6 rack.id AZ3

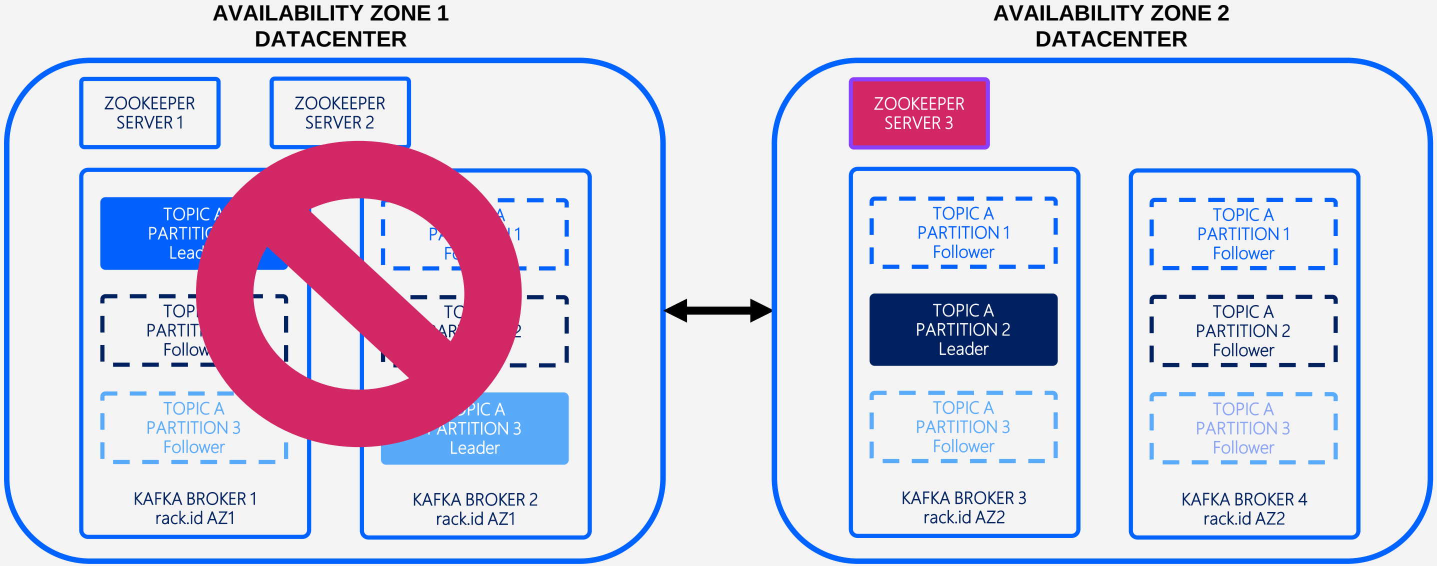
`Replication factor` > `min.insync.replicas`

Ensure there are sufficient replicas to cover all zones

`client.rack` tags which node the client application is running on

- Consumers to fetch from the closest replica

Zookeeper


AVAILABILITY ZONE 1
DATACENTER
ZOOKEEPER SERVER 1
ZOOKEEPER SERVER 2
TOPIC A PARTITIO Lead
TOP PARTITI Follow
TOPIC A PARTITION 3 Follower
KAFKA BROKER 1
rack.id AZ1
A PA 1 F
TO AR 2
PIC A ARTITION 3 Leader
KAFKA BROKER 2
rack.id AZ1
AVAILABILITY ZONE 2
DATACENTER
ZOOKEEPER SERVER 3
TOPIC A PARTITION 1 Follower
TOPIC A PARTITION 2 Leader
TOPIC A PARTITION 3 Follower
KAFKA BROKER 3
rack.id AZ2
TOPIC A PARTITION 1 Follower
TOPIC A PARTITION 2 Follower
TOPIC A PARTITION 3 Follower
KAFKA BROKER 4
rack.id AZ2

Stretch clusters

Kafka and Zookeeper replication ensures data is highly available

Data consistency

Exactly once processing possible

Event order can be preserved

Guards against the loss of a data center

Simple client configuration

No offset lag or offset translation

Utilizes all brokers

Considerations

Stable, low latency, high-bandwidth connection is a must : care if crossing regions

No 0 downtime upgrades

Doesn't protect against whole cluster failure

Cross-availability zone data transfer fees will apply : especially ingress/egress

Third datacenter required

Complexity of configuration

Multiple Clusters

Multiple clusters

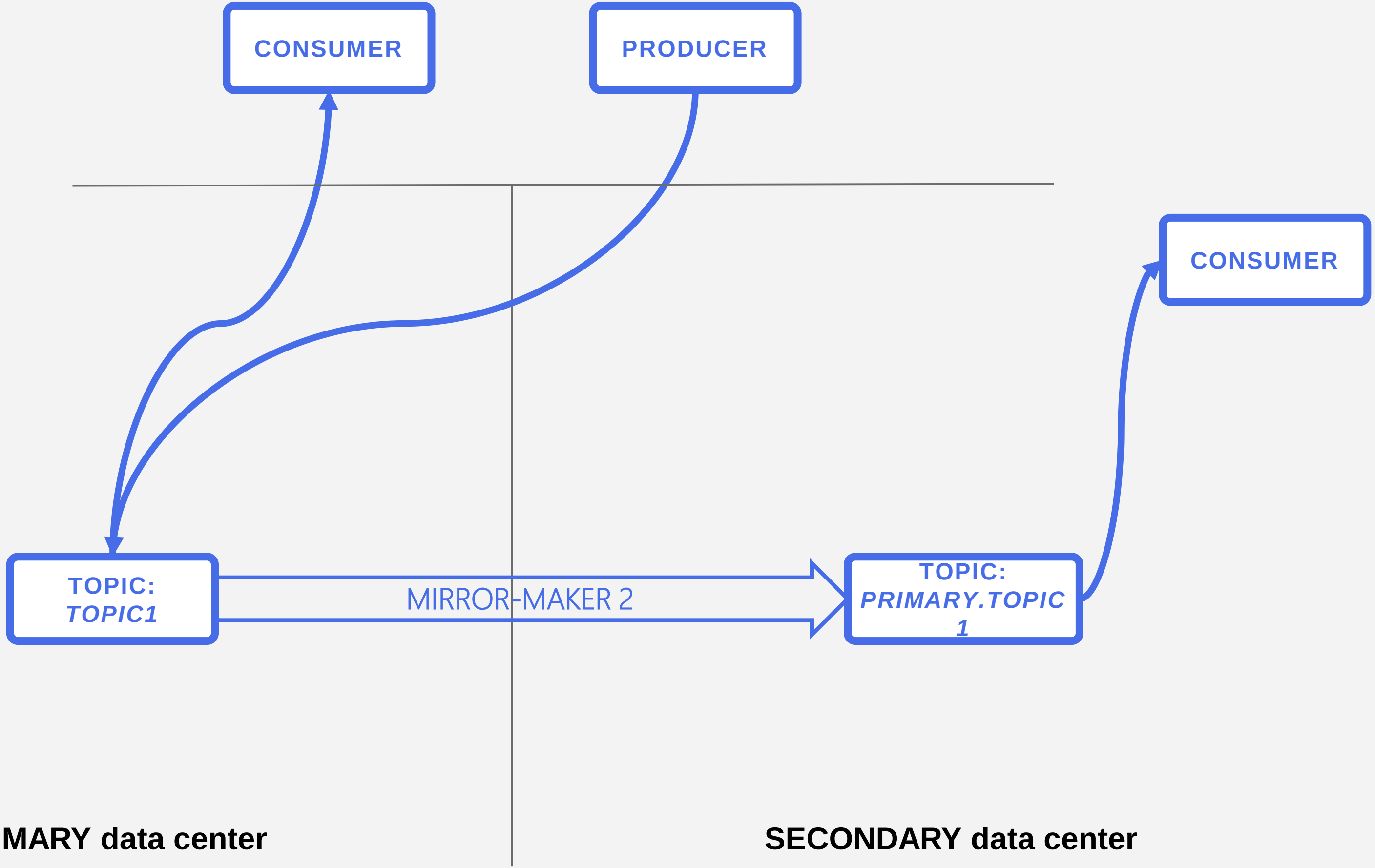
Multiple independent Kafka clusters in different regions. Topic data is mirrored across clusters

- **Active/passive**: Produce to primary cluster, consume from any
- **Active/active**: Produce to and consume from any cluster
- **Federated**: Central cluster with multiple regional clusters

A **mirror making** technology mirrors the data between clusters

- Typically runs at the target cluster
- Data is consumed remotely and produced locally

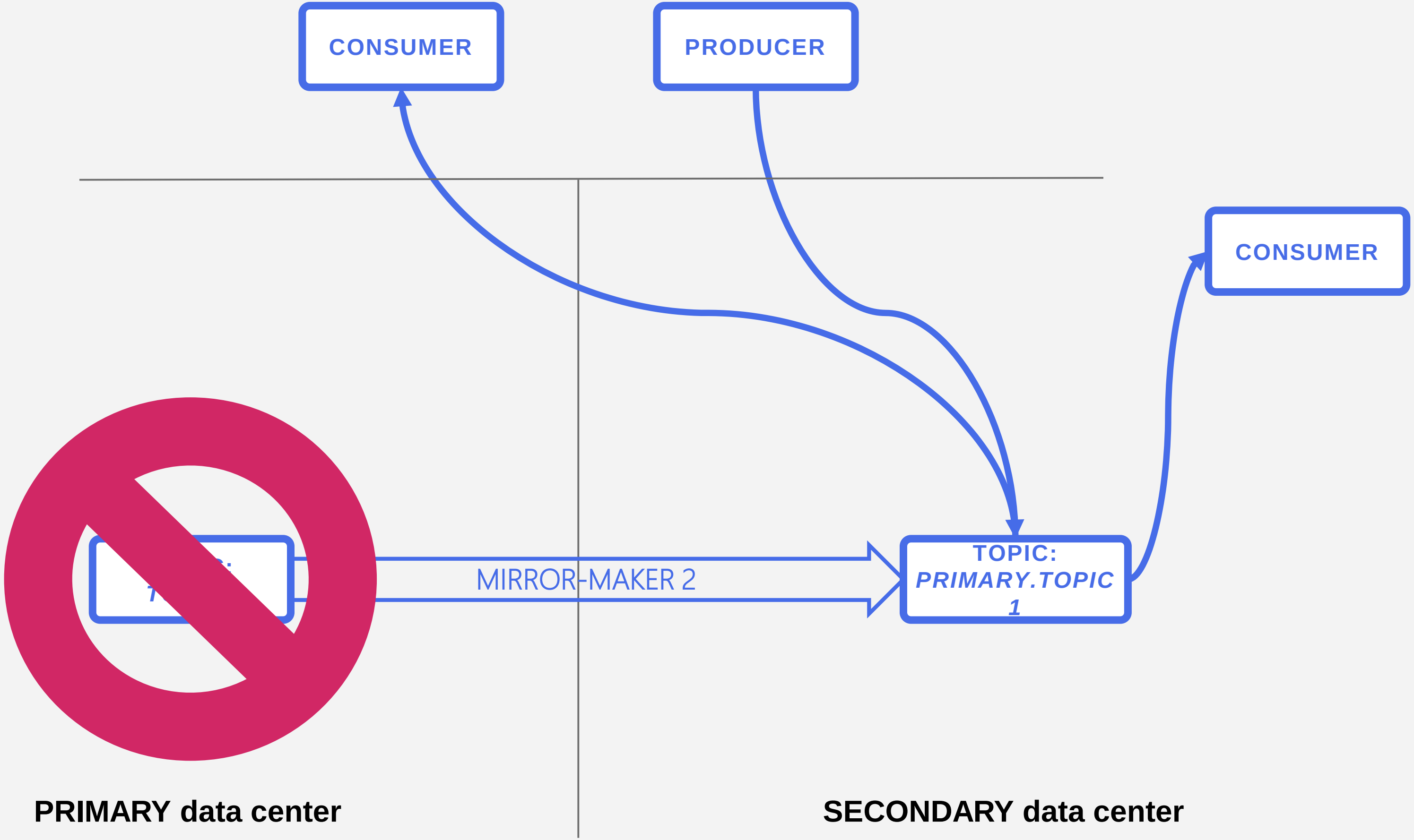
Active/passive clusters



PRIMARY data center

SECONDARY data center

Active/passive clusters



Active/passive clusters

Data back-up to multiple destinations

Disaster recovery fail-over after loss of infrastructure

Data migration:

- Moving to a new cluster
- Moving from a staging environment to a production environment

Clusters are independent

Message keys used for partitioning, so order is preserved on a per-key basis

Source cluster must be the 'owner' of the data. Target cluster is essentially read-only

Considerations

Target cluster will lag behind source cluster - mirroring is asynchronous, so monitor that it's not too far behind

- High traffic topics could mean 1000’s of un-mirrored messages, even if there is only a few milliseconds of lag

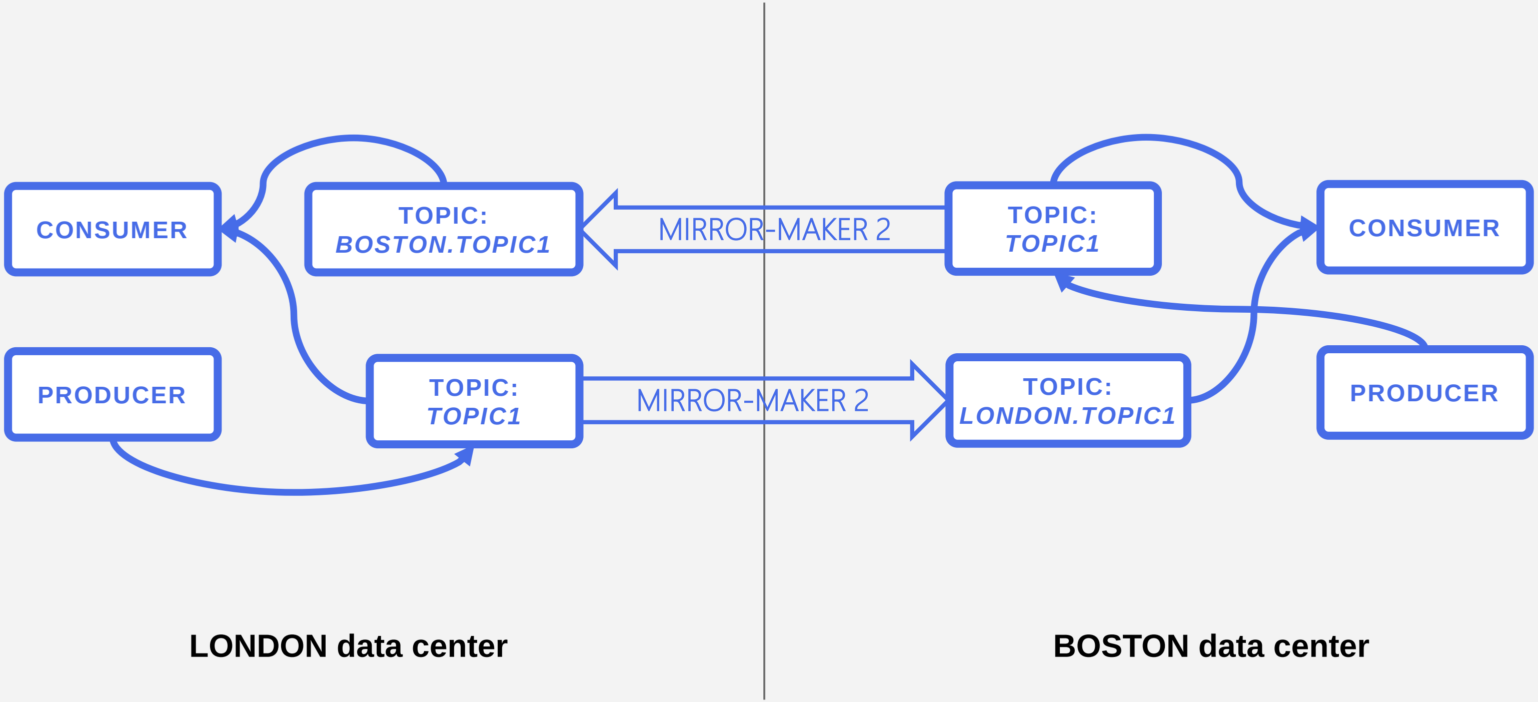
Message offsets in the source and target topics may not match up

- On fail-over consumers need to be updated to start at the correct offset
- Returning to the source cluster will require a similar process

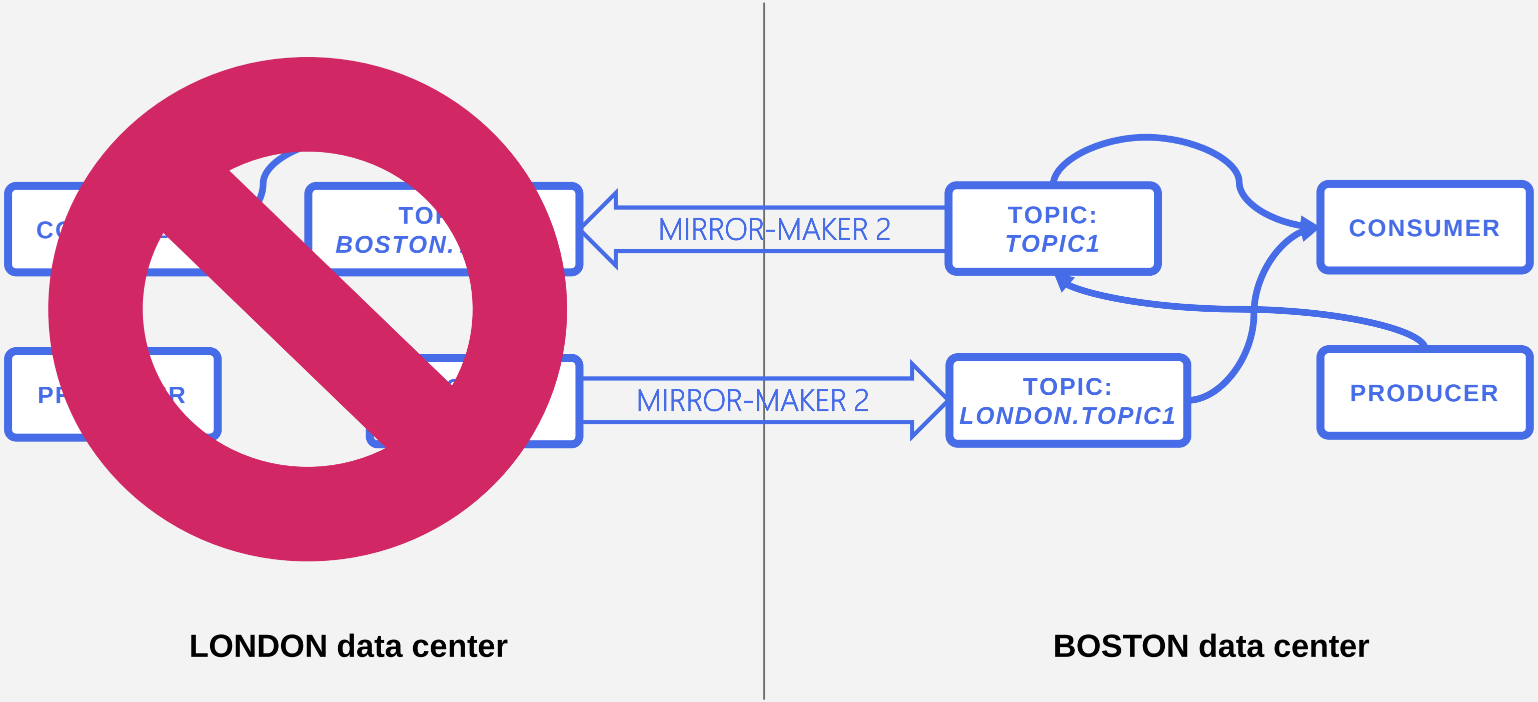
Configuration changes on the source may need to be propagated to the target

Target cluster may be unused (until it’s needed)

Active/active clusters



Active/active clusters



LONDON data center

BOSTON data center

Active/active clusters

Implicit disaster recovery – data centers can operate independently

Data back-up to multiple destinations

A ‘virtual’ topic shared across geographies

Serve users from a nearby data center to increase performance

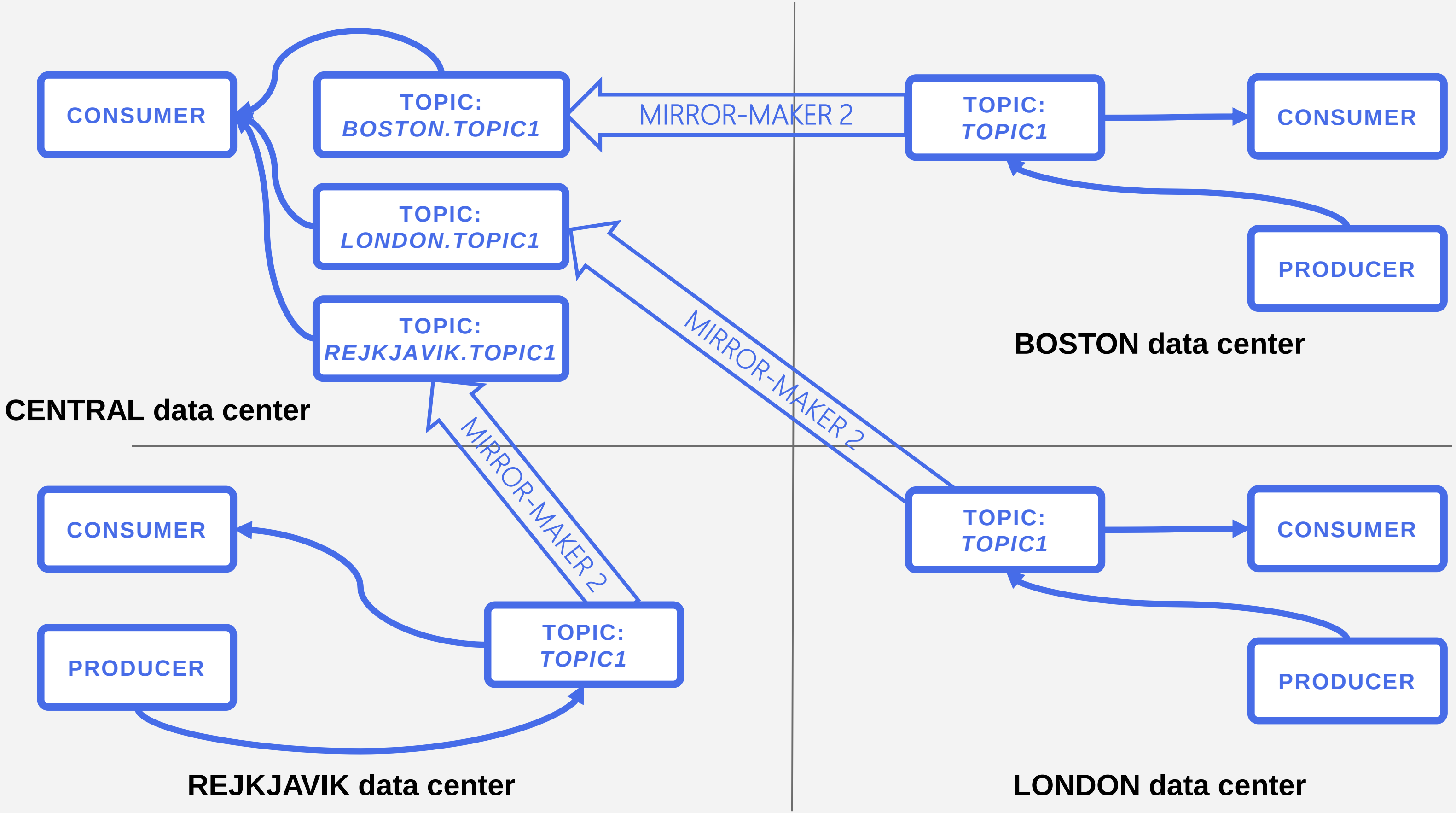
Redundancy and resilience - can easily do a network redirect on failure to route all traffic to surviving cluster

Considerations

Possible to accidentally configure loops of data

Lag and ordering issues may arise if trying to consume from a different data center to where the data was produced

Federated (hub and spoke) clusters



Federated (hub and spoke) clusters

Each region has a Kafka cluster to handle data for the region

There is no requirement for one region to know about data for other regions

Serve users from a nearby data center to increase performance

Central cluster consolidates data from each regional center

- Can be used when central processing requires access to the full set of data

Mirroring is single direction

- Makes it simple to configure, deploy and monitor

Be aware

Not useful if all regions need access to the data

Central cluster is for secondary processing or back-up

Lag and ordering issues at the central cluster may arise if regional data is co-dependent

Planning multi-cluster architectures for fail-over

Consider how applications will fail-over to different clusters

- Bootstrap address / DNS
- Certificates, user credentials, permissions
- Consumer offsets
- Topic subscriptions
- Message ordering

Consider effect of duplicates/lost messages

- Idempotent writes

Monitor that data is arriving at remote clusters as well as your primary system health

Practice your fail-over and fail-back

Summary

Kafka provides a good degree of high availability

Some applications require additional guarantees

Dependent on your requirements and your infrastructure

Thank you

Andrew Borley
Emma Humber

IBM Event Streams

—

borley@uk.ibm.com
emma.humber@uk.ibm.com