Axon Training

Module 7 – CQRS & Distributed Systems

AxonIQ

Agenda

Week 1

1. DDD and CQRS Fundamentals
2. Command Model
3. Event Handling & Projections
4. Sagas and Deadlines

Week 2

1. Snapshotting and Event Processors
2. Preparing for Production
3. **CQRS and Distributed Systems**
4. Monitoring, Tracing, Advanced Tuning

Together we can achieve great things...

Distributed Systems

“Evolutionary” Microservices

- “Microservices are a journey, not a destination”
- Build microservices, monolith-first
  - Separate components as requirement comes up
  - Ensure correct abstraction of monolith’s components

Location Transparency

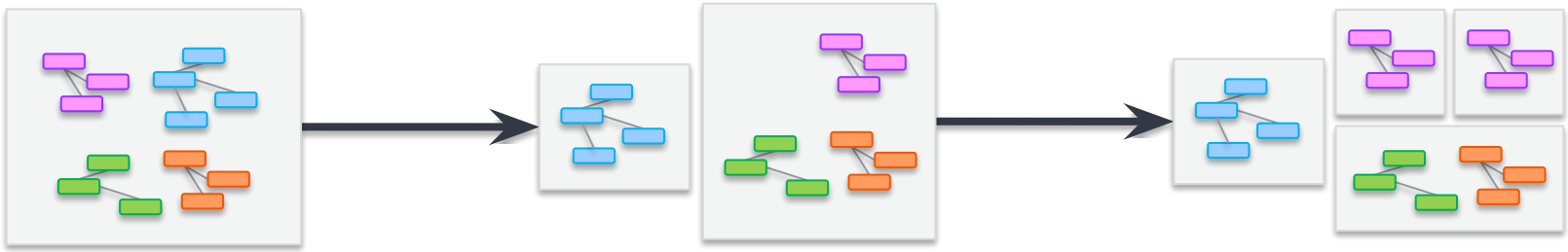
- A Component should not be aware, nor make any assumptions, of the physical location of Components it interacts with.
- Beware of APIs & method signatures:
  - Not location transparent:

```
public Result doSomething(Request request) {...}
```

  - Location transparent alternatives:

```
public void doSomething(Request request, Callback<Response> callback) {...}
public CompletableFuture<Result> doSomething(Request request) {...}
```

Location Transparency

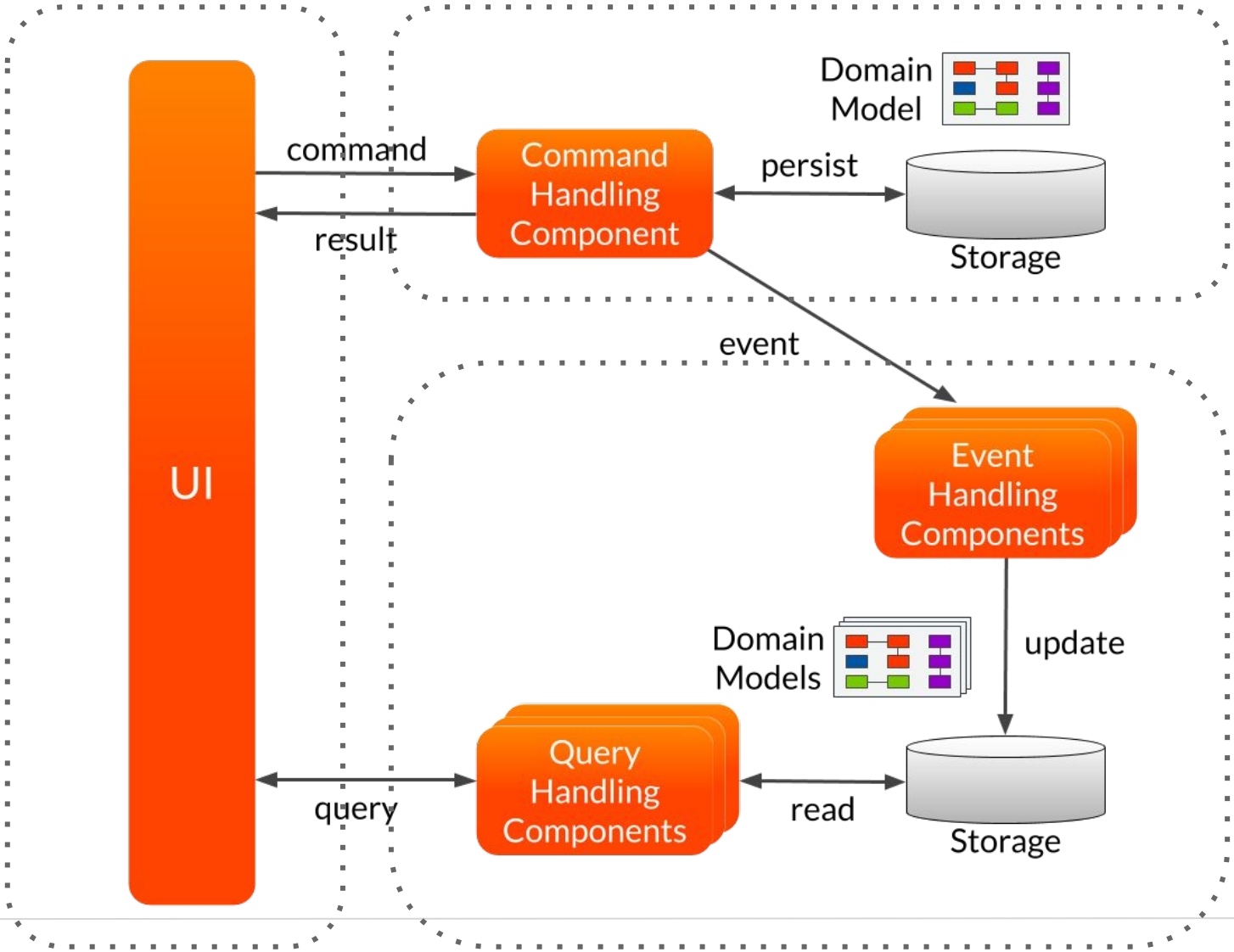


1 – Build a monolith

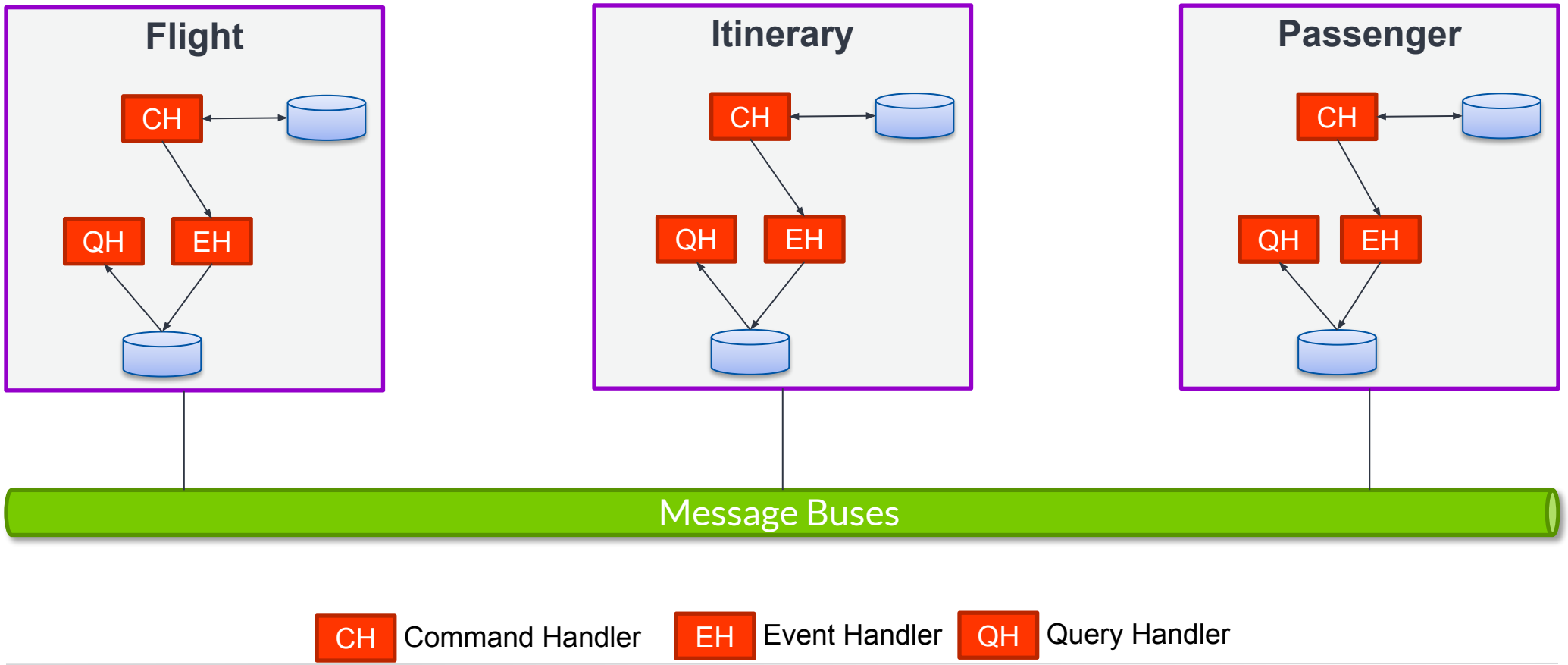
2 – Extract components

3 – Extract more
4 – Scale as needed

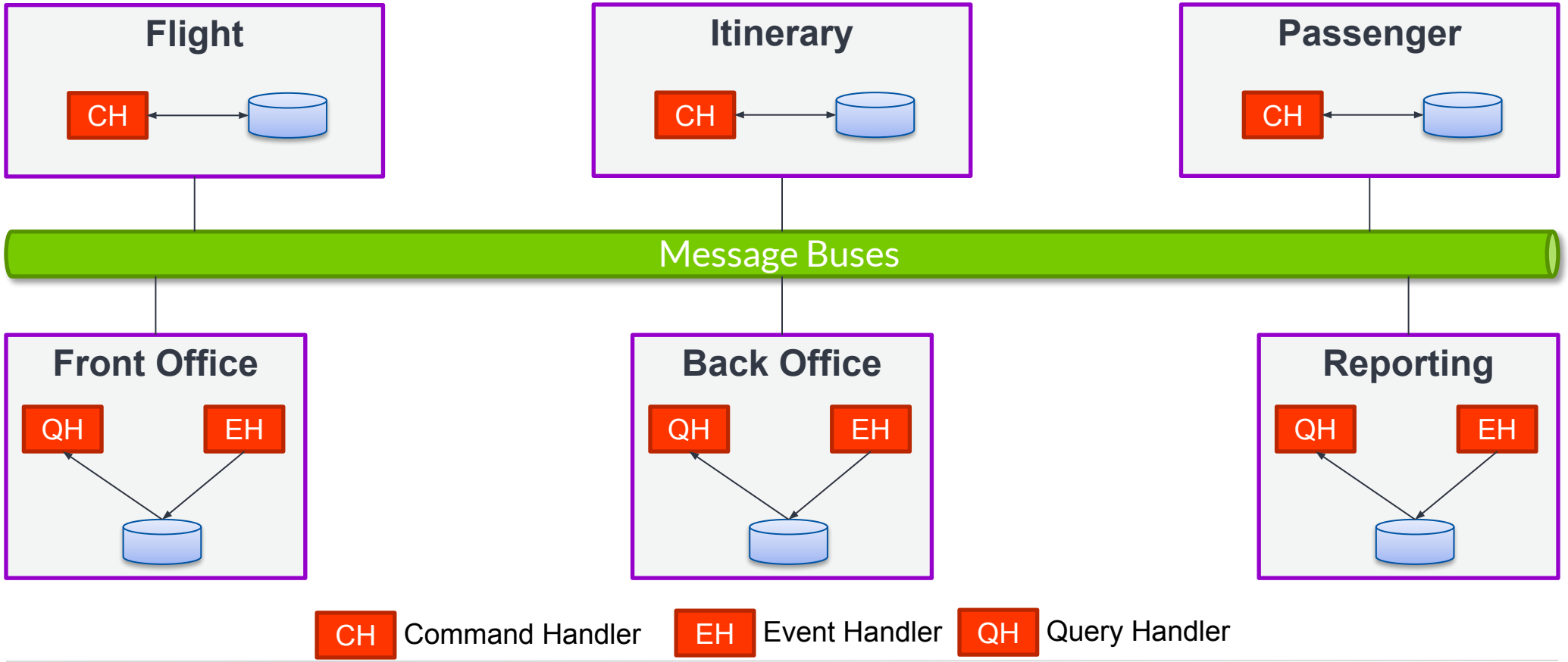
Location Transparency boundaries



Scaling scenario – Bounded Context



Scaling scenario – Separate audience



Sharing Events between nodes

- ”Proper” Event Store, like Axon Server

- Embedded Event Store with shared data source
  - Tracking processors will track all stored events
- Message Broker
  - Publish all events messages to broker
  - Register Message Broker as message source for Processors
  - Spring AMQP: `SpringAMQPPublisher`
- Beware the “contract”!

Distributing Command Messages

- Dedicated message routing solution: Axon Server
  - Unified solution for *all* messages

- `DistributedCommandBus`
  - `CommandRouter`
  - `CommandBusConnector`
    - Spring Cloud Discovery & Jgroups implementations available

Distributing Query Messages

- Dedicated message routing solution: Axon Server
  - Unified solution for *all* messages

The more the merrier?

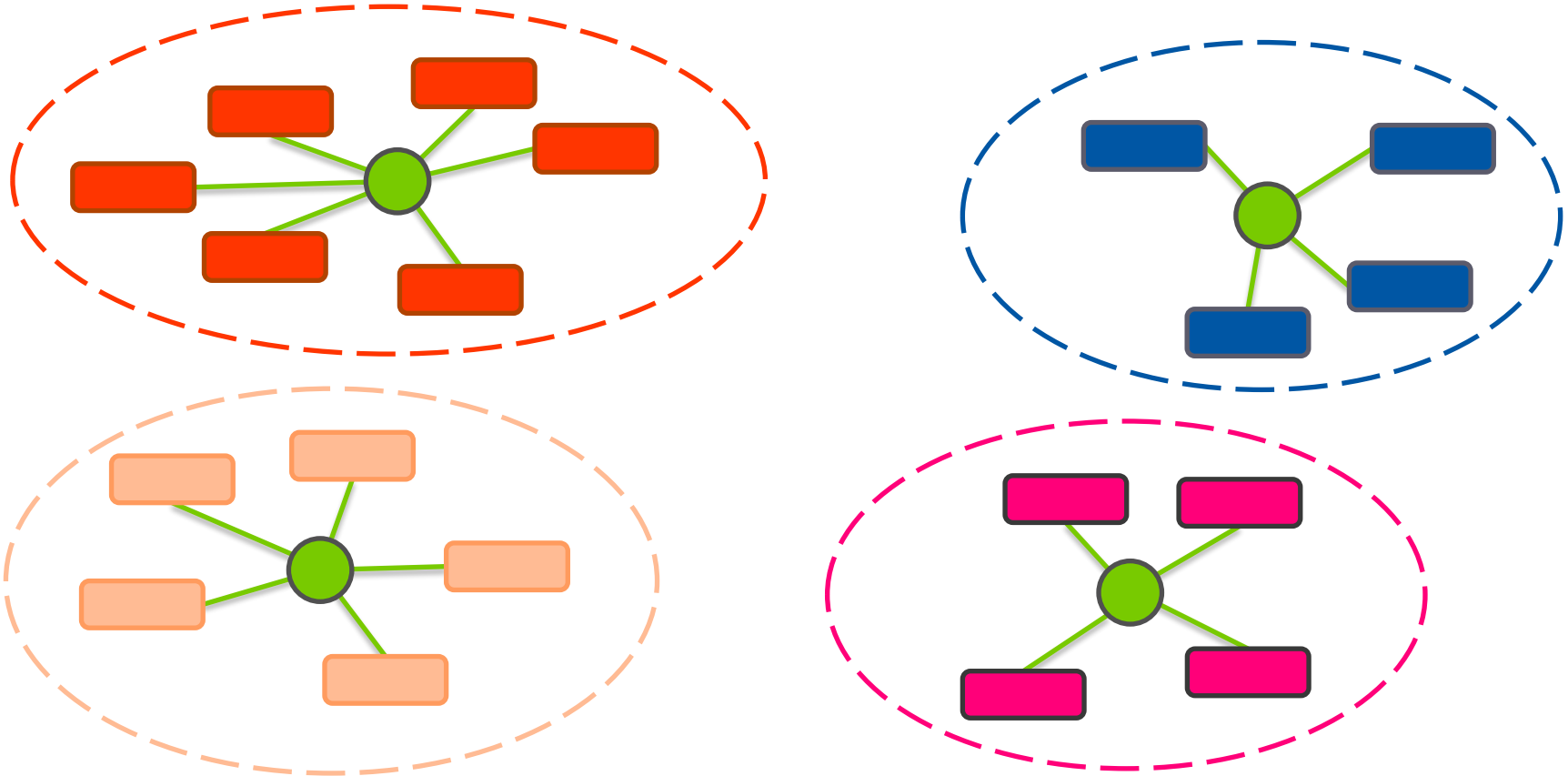
Large Scale Distributed Systems

Unmanageable mess

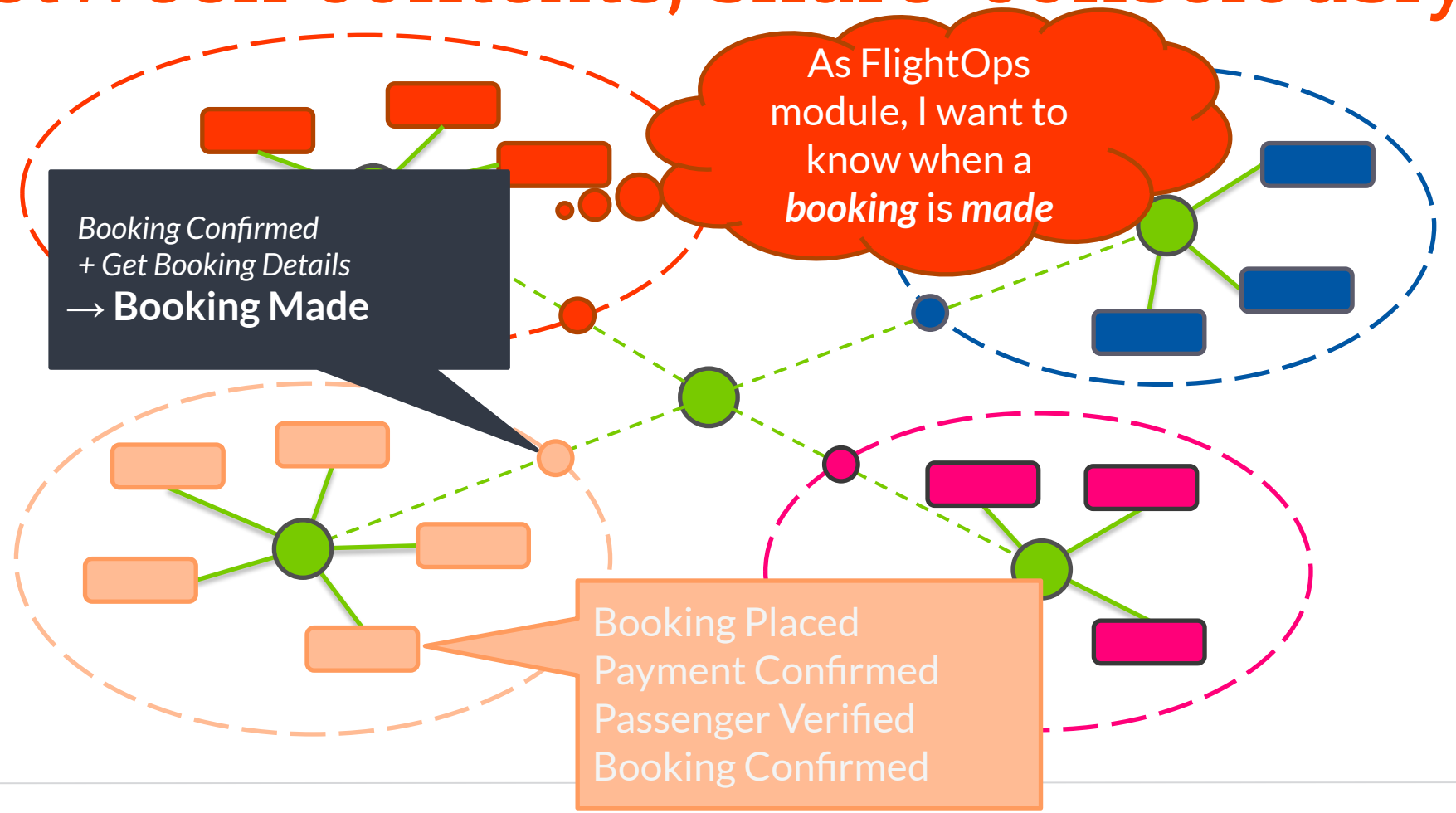
Bounded context

> Explicitly define the context within which a model applies. Explicitly set boundaries in terms of team organization, usage within specific parts of the application, and physical manifestations such as code bases and database schemas. Keep the model strictly consistent within these bounds, but don't be distracted or confused by issues outside.

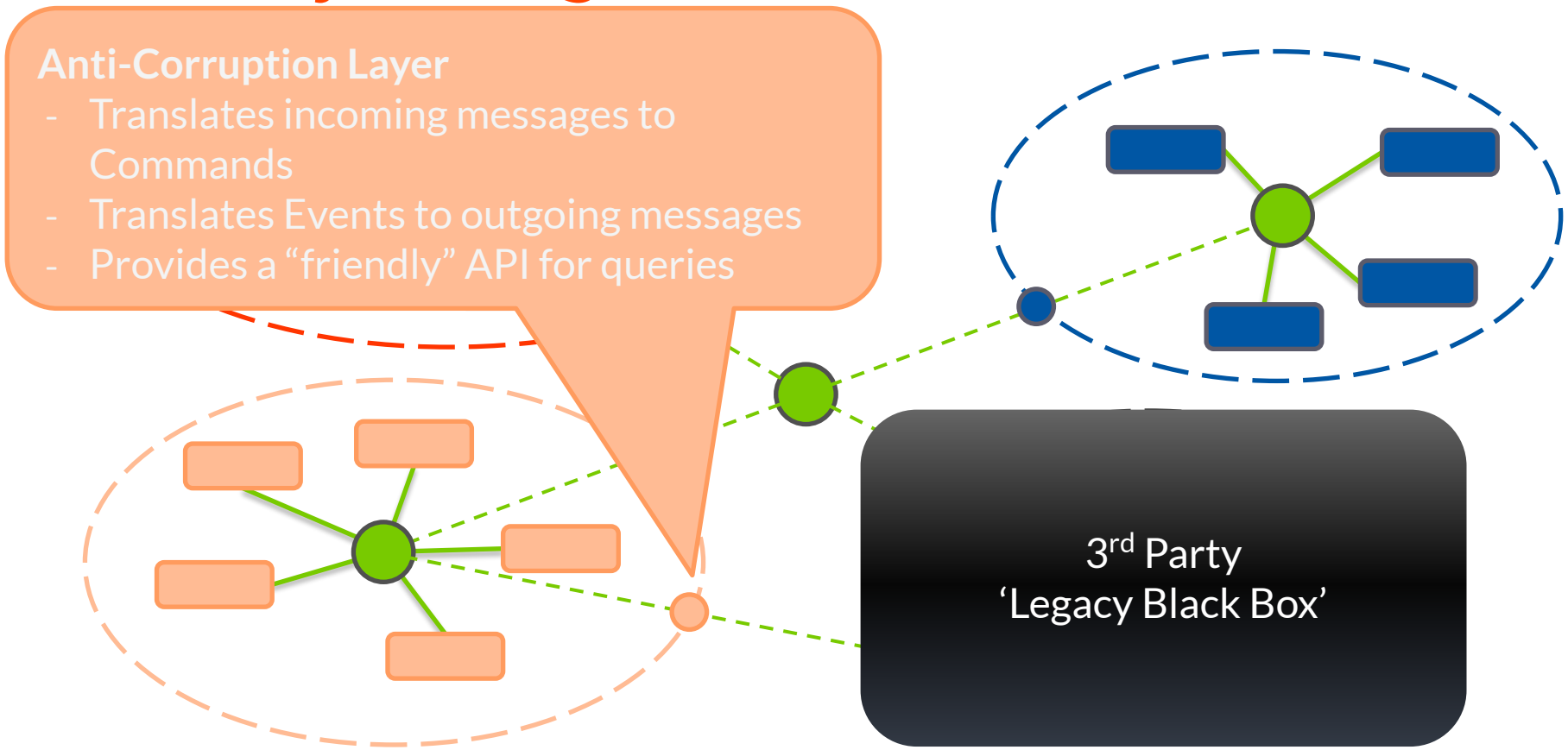
Within a context, share ‘everything’



Between contexts, share 'consciously'

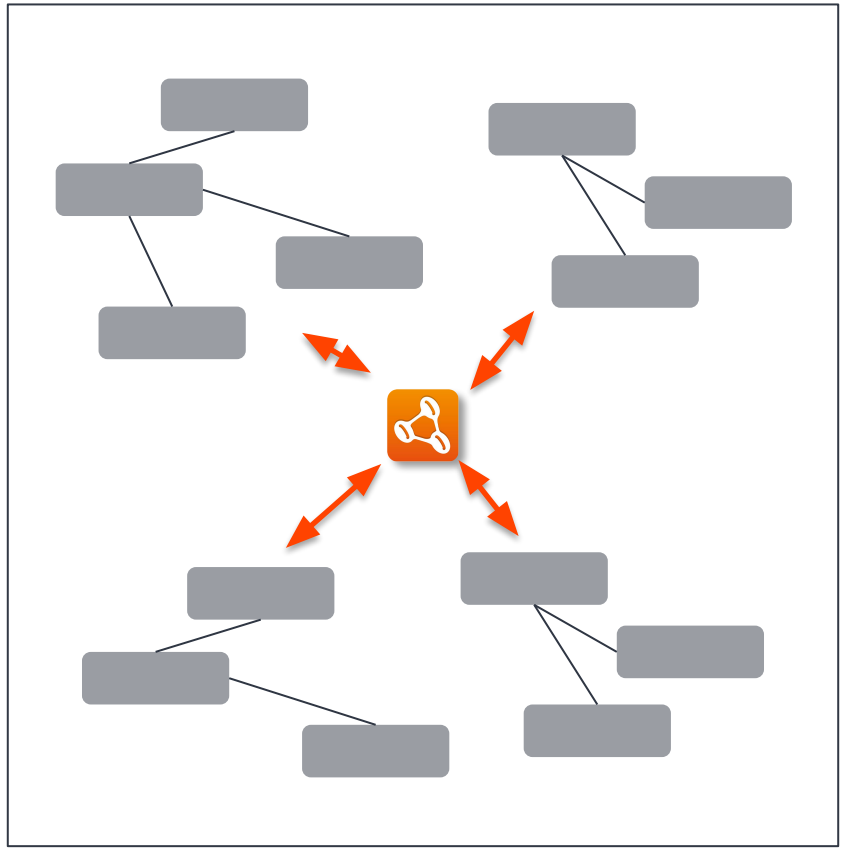


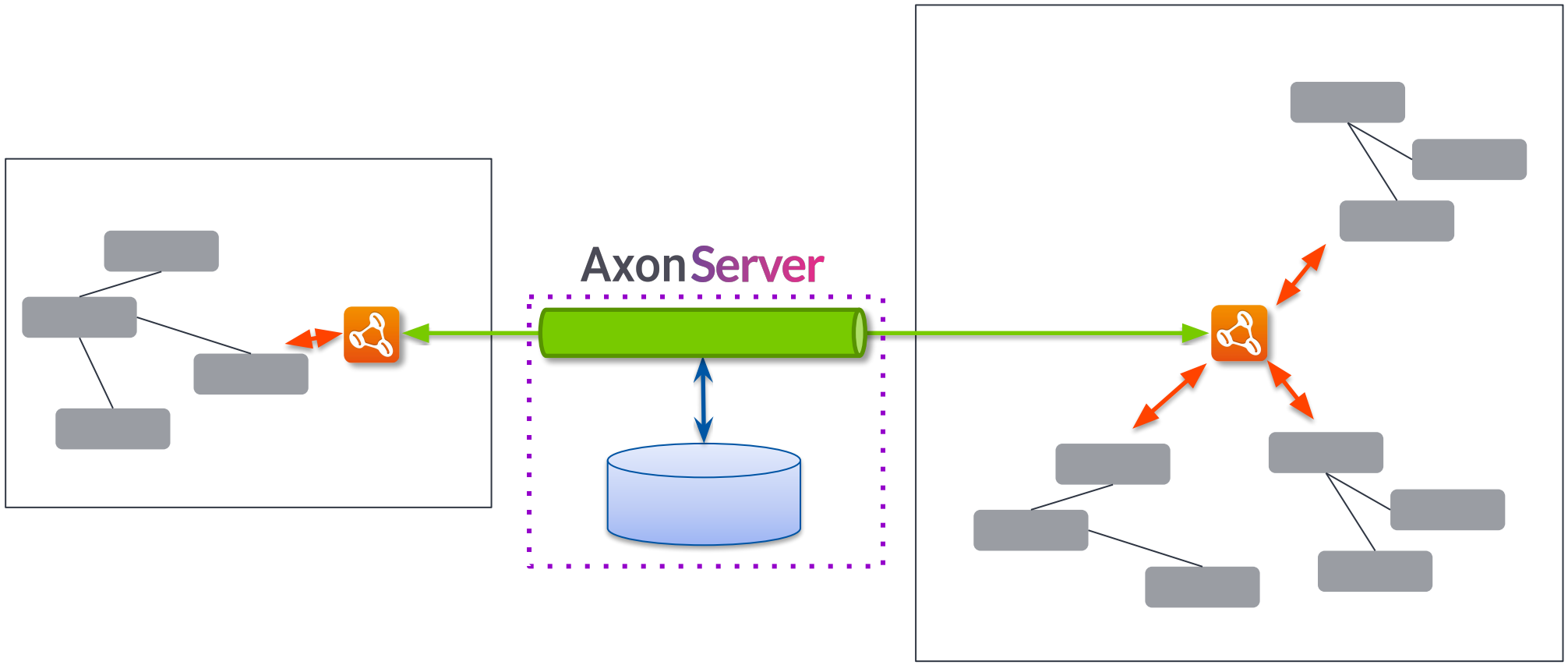
3rd Party Integration

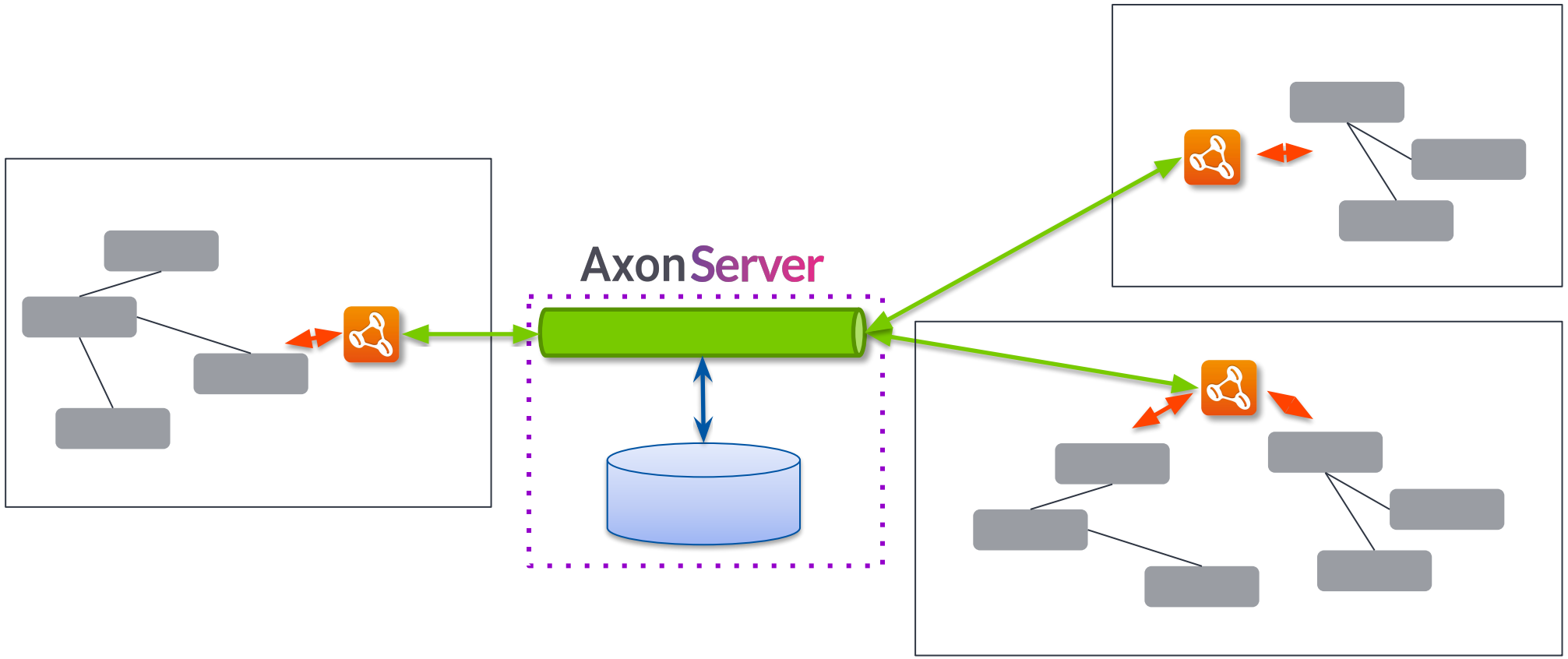


The only thing you'll ever need...

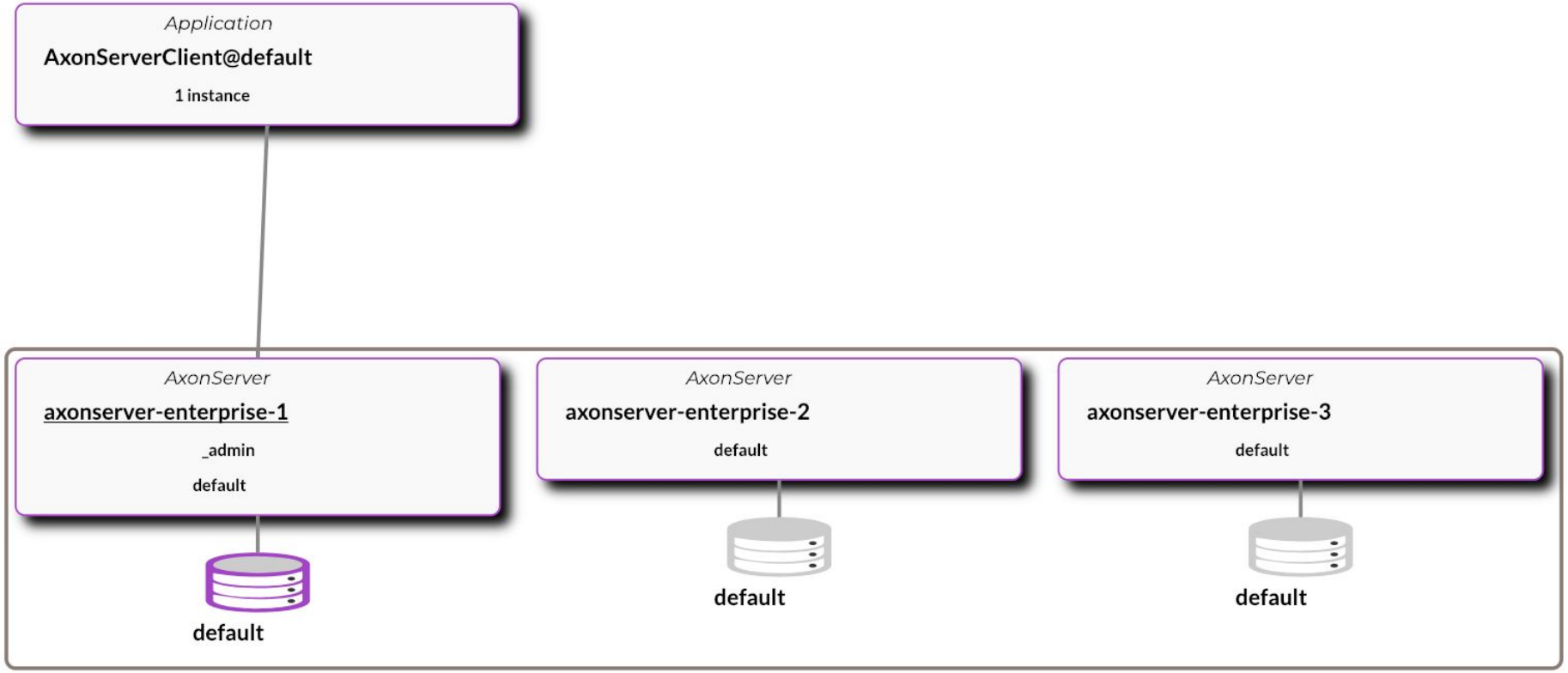
Axon Server




AxonServer


AxonServer

Please allow me to introduce myself



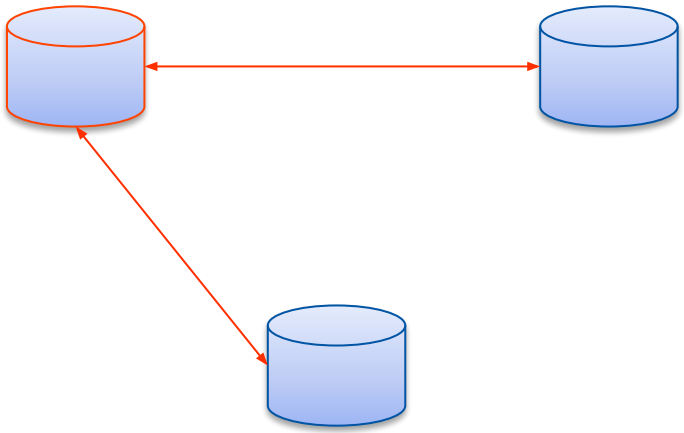
(Auto)Clustering

- Increases availability
  - As long as majority of nodes are up cluster is available
- One leader per context
  - Appending events goes through the leader of the context
  - Other operations are balanced on other nodes
- Incremental cluster changes
- Automatic initialization

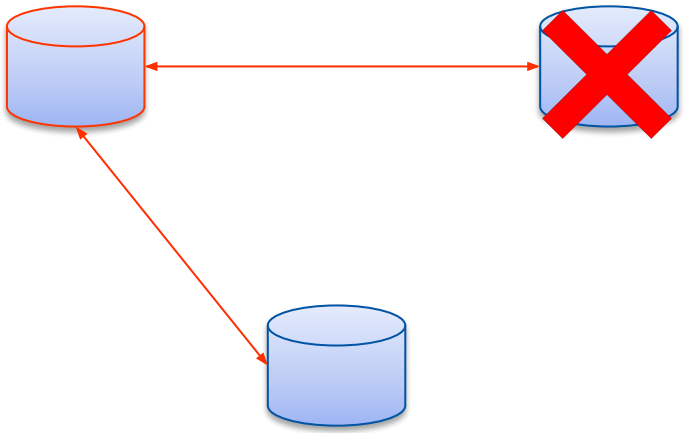
Event replication

- RAFT protocol
- Single leader
- Needs majority to confirm in order to commit

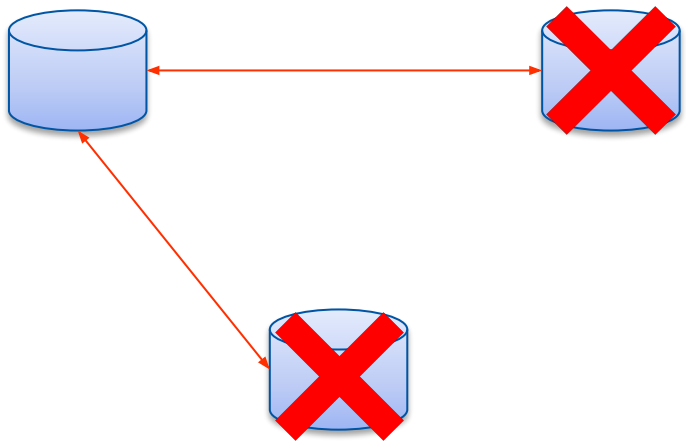
Cluster scenarios – all nodes up



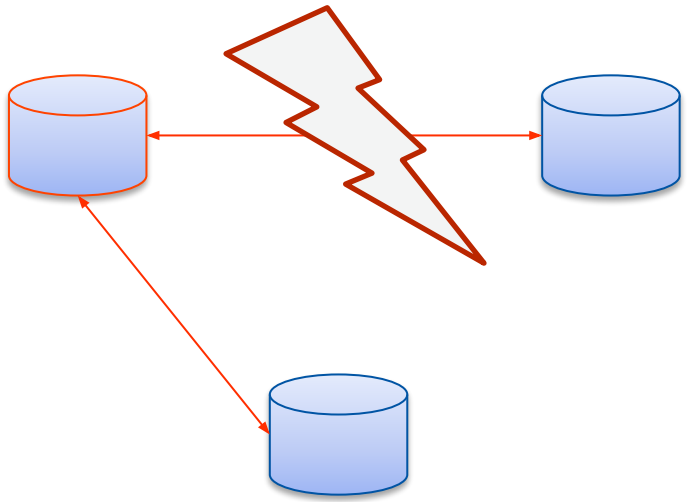
Cluster scenarios – minority down



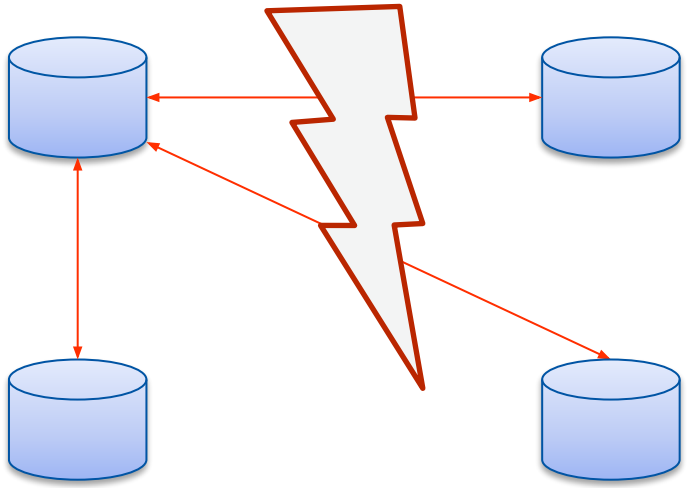
Cluster scenarios – majority down



Cluster scenarios – network partition



Cluster scenarios – network partition



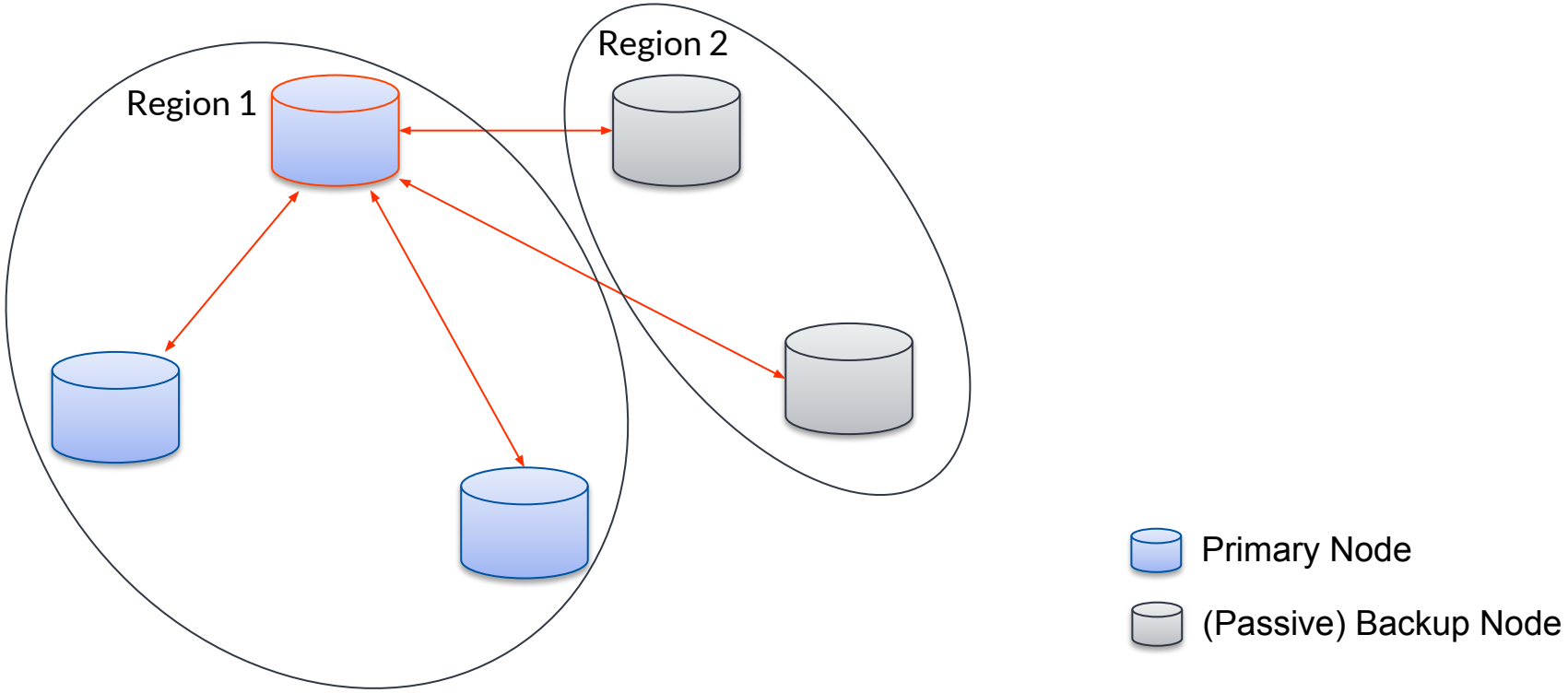
Node Roles

- PRIMARY – Participate in Transaction ACKs
- ACTIVE_BACKUP – Track transactions. Participates in ACKs
- PASSIVE_BACKUP – Track transactions. Does not participate in ACKs
- MESSAGING_ONLY – Does not track transactions.
- SECONDARY - Offloading location for recent events

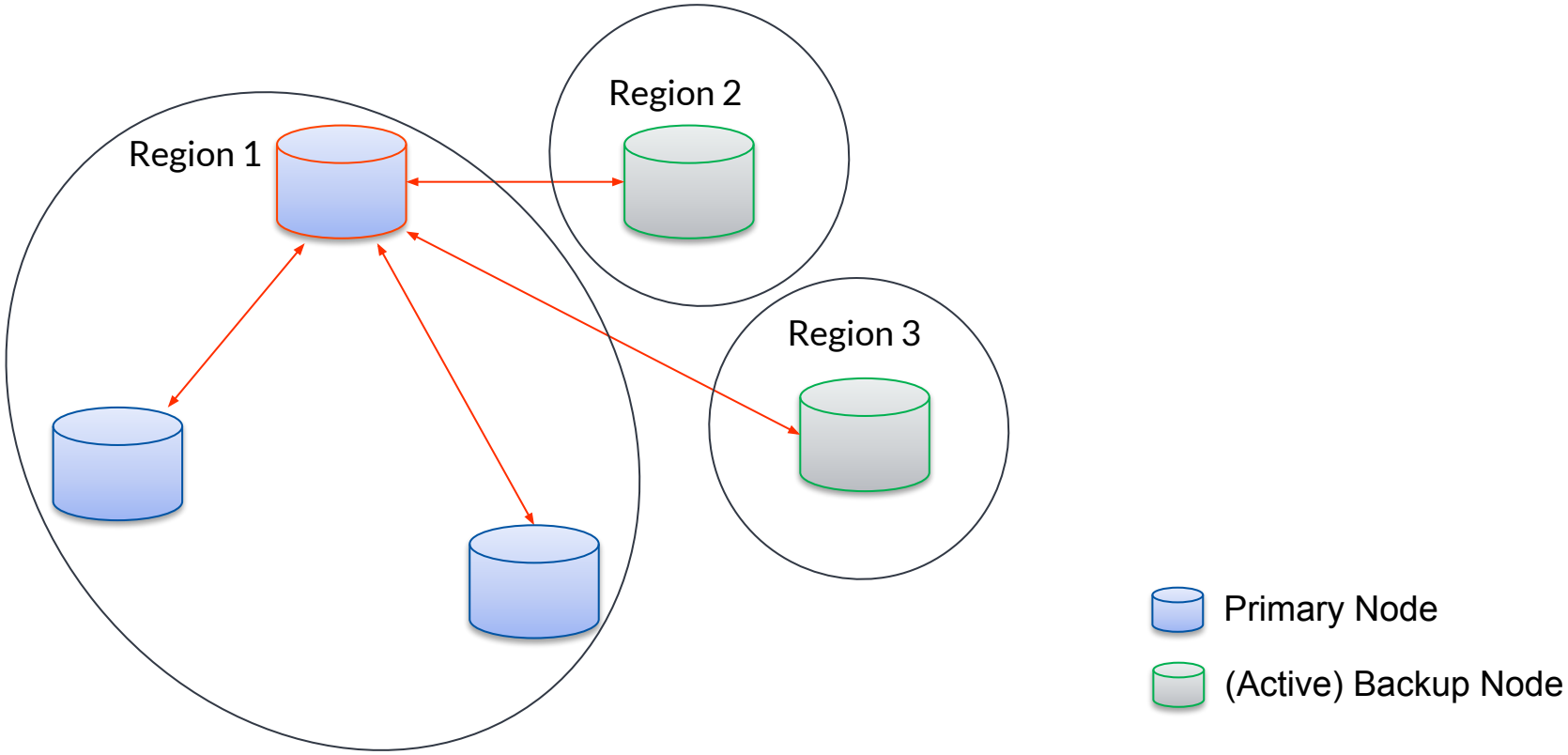
Deployment Patterns

- Odd number of nodes (e.g. 3 or 5) per context
- Don’t put all (majority of) eggs in one basket (AZ)

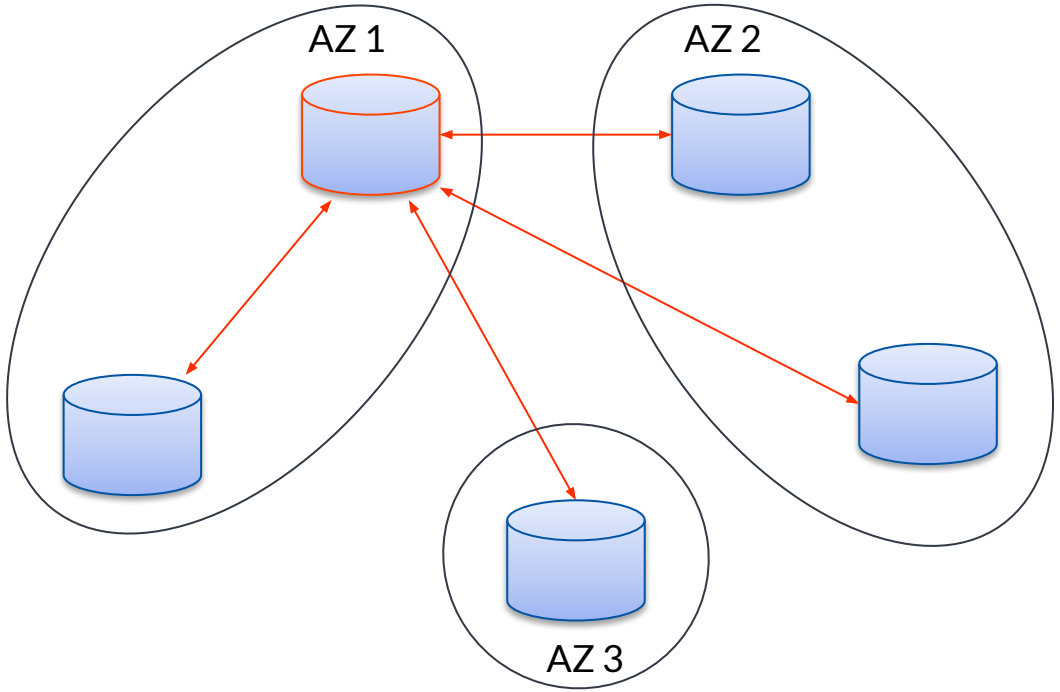
Disaster Recovery - Passive Backup



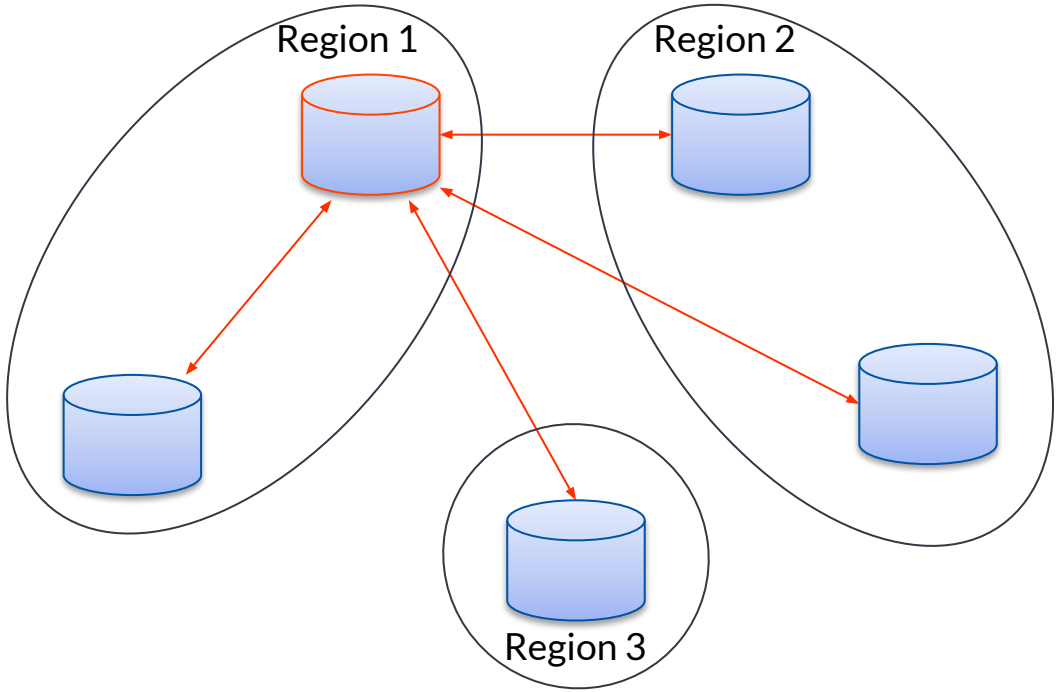
Disaster Recovery - Active Backup

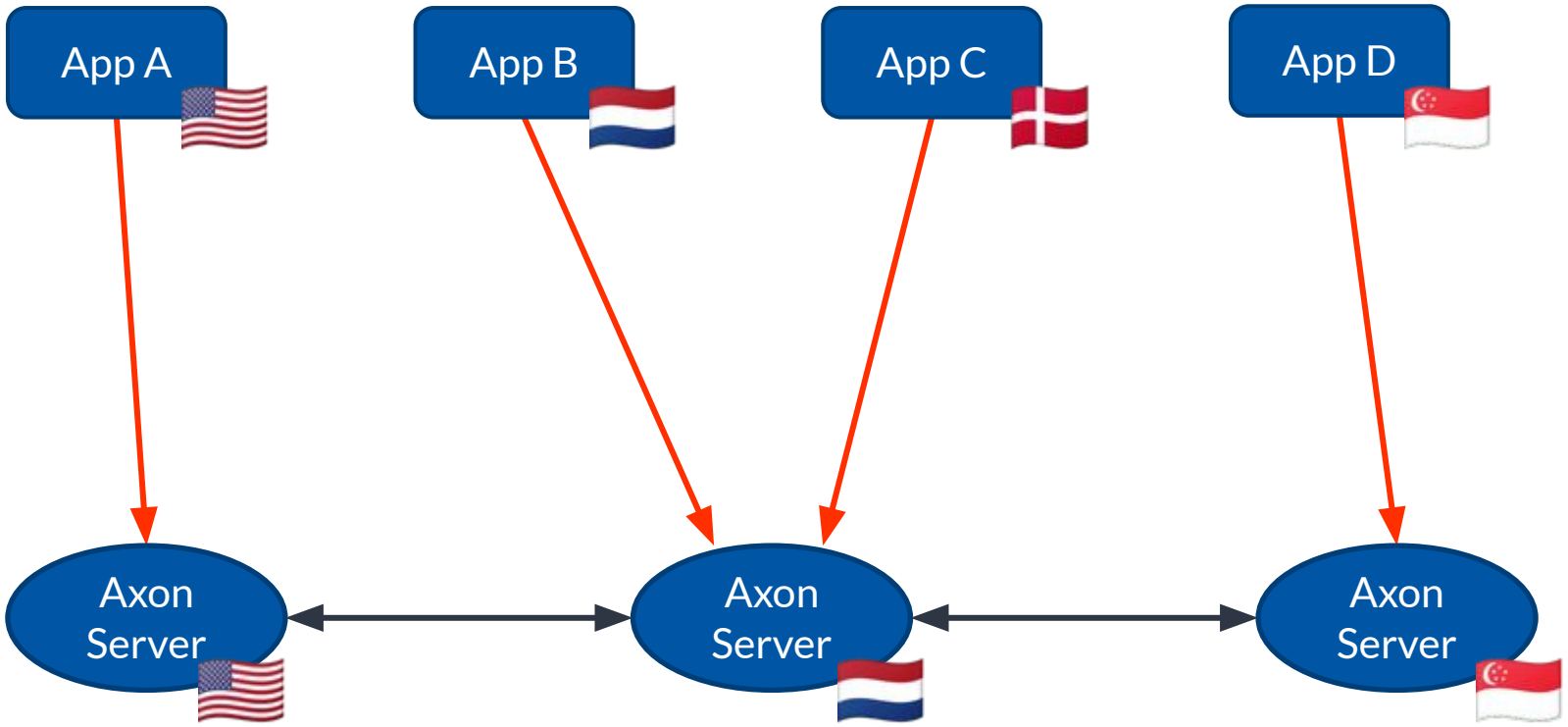


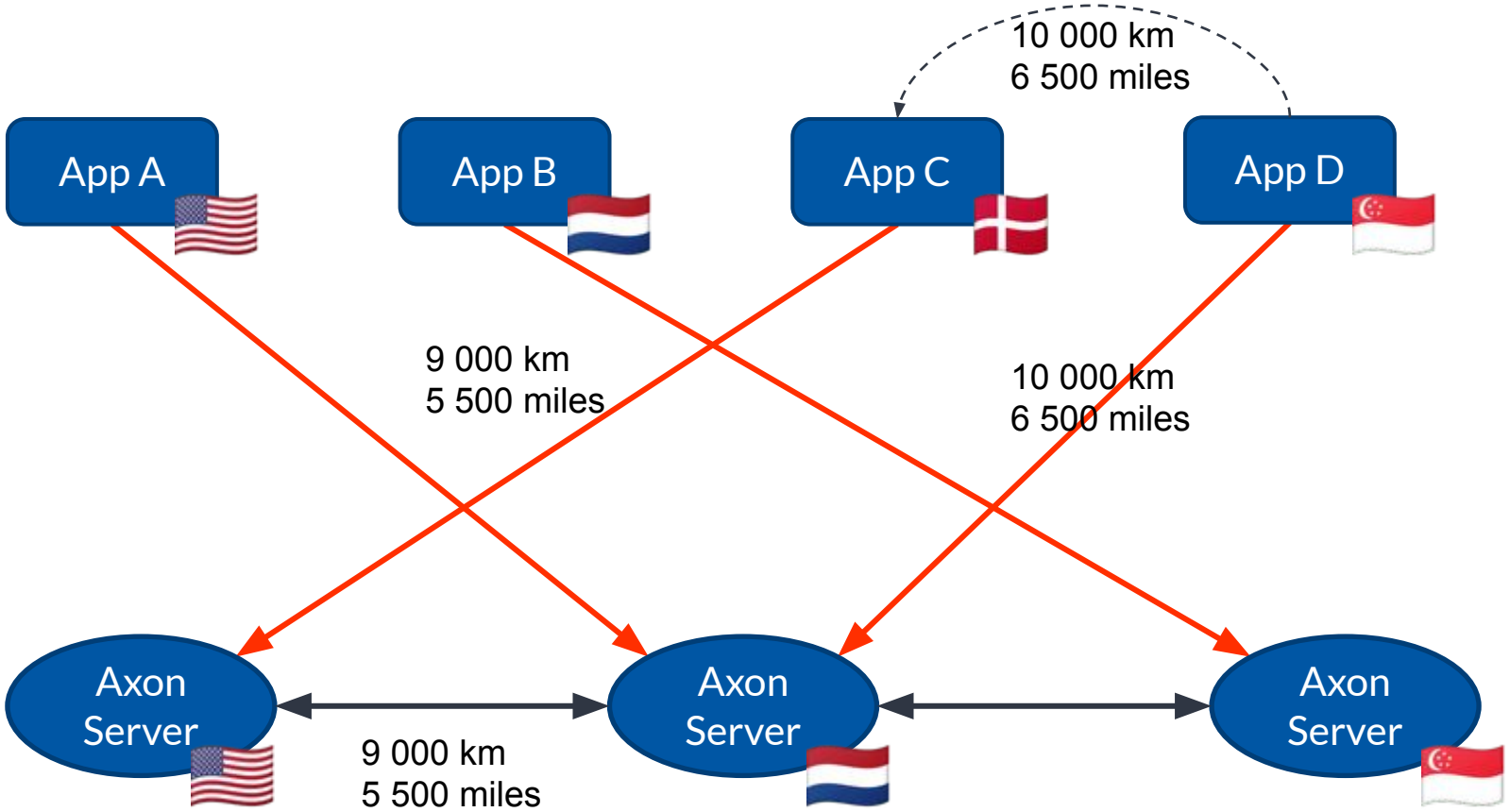
High Availability – Multi-AZ



High Availability – Multi-Region

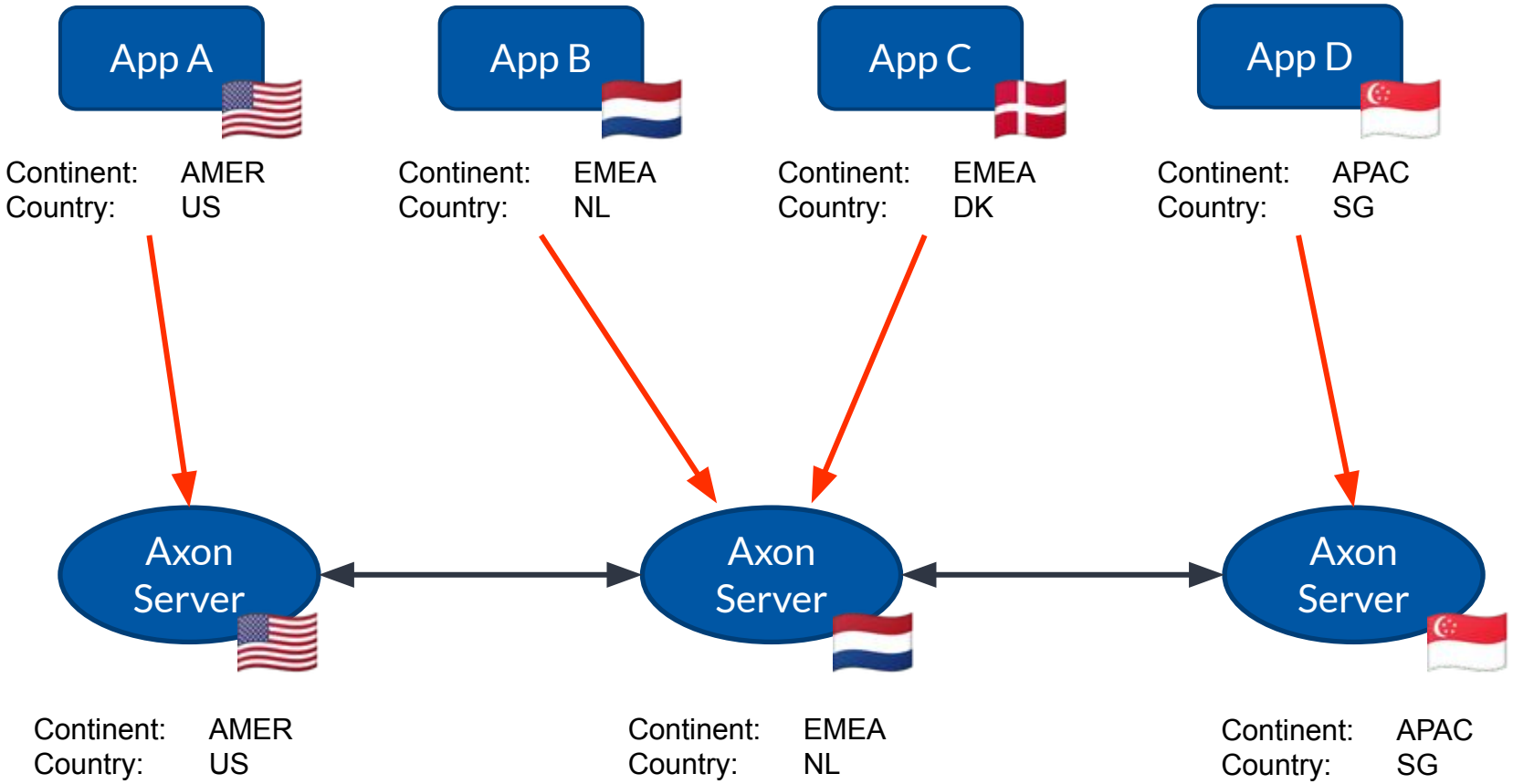



App A
App B
App C
App D
Axon Server
Axon Server
Axon Server


10 000 km
6 500 miles
App A
App B
App C
App D
9 000 km
5 500 miles
10 000 km
6 500 miles
Axon Server
Axon Server
Axon Server
9 000 km
5 500 miles

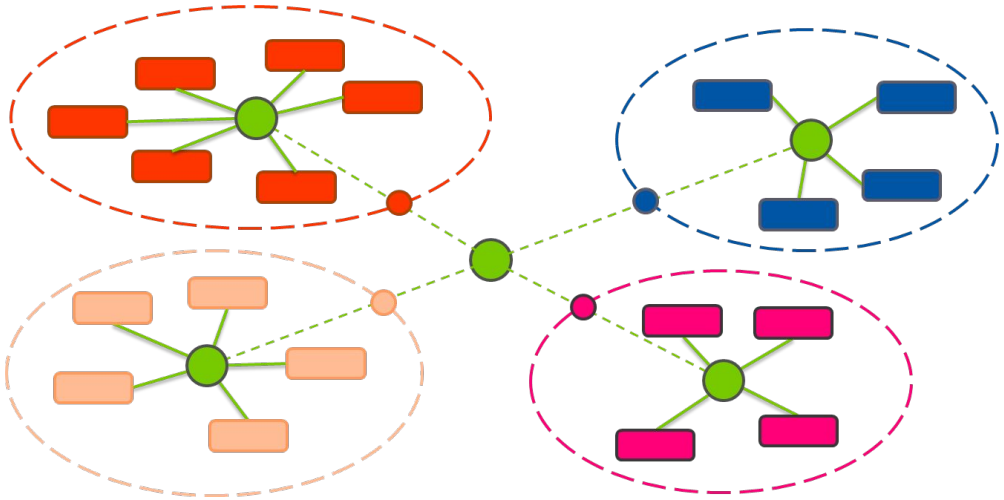
Location Awareness

- (Location) Tags
  - Axon Server: *axoniq.axonserver.tags.[tag-name]=[tag-value]*
  - Axon Application: *axon.tags.[tag-name]=[tag-value]*


App A
Continent: AMER
Country: US
App B
Continent: EMEA
Country: NL
App C
Continent: EMEA
Country: DK
App D
Continent: APAC
Country: SG
Axon Server
Continent: AMER
Country: US
Axon Server
Continent: EMEA
Country: NL
Axon Server
Continent: APAC
Country: SG

Multi Context

- Events stored in directory per context
- By default, RAFT group per context
  - Contexts may be grouped together under a Replication Group
- *_admin* context



Access Control

- Axon Server
  - *axoniq.axonserver.accesscontrol.enabled=true*
- Application Token
  - *axon.axonserver.token=[Token]*
- Cluster
  - *axoniq.axonserver.accesscontrol.internal-token=[Token]*

Roles

- ADMIN
- CONTEXT_ADMIN
- DISPATCH_COMMANDS
- DISPATCH_QUERY
- MONITOR
- PUBLISH_EVENTS
- READ_EVENTS
- SUBSCRIBE_COMMAND_HANDLER
- SUBSCRIBE_QUERY_HANDLER
- USE_CONTEXT

Backup

- Data availability naturally supported in clustered environment.

- *Backup endpoints*
  - *POST - /v1/backup/createControlDbBackup*
  - *GET - /v1/backup/filenames [EVENT / SNAPSHOT]*
  - *GET - /v1/backup/log/filenames*

Whatever else you wanted to know...

Questions