What can you achieve with MQ clusters?

A. Simplification, availability and scalabiltiy

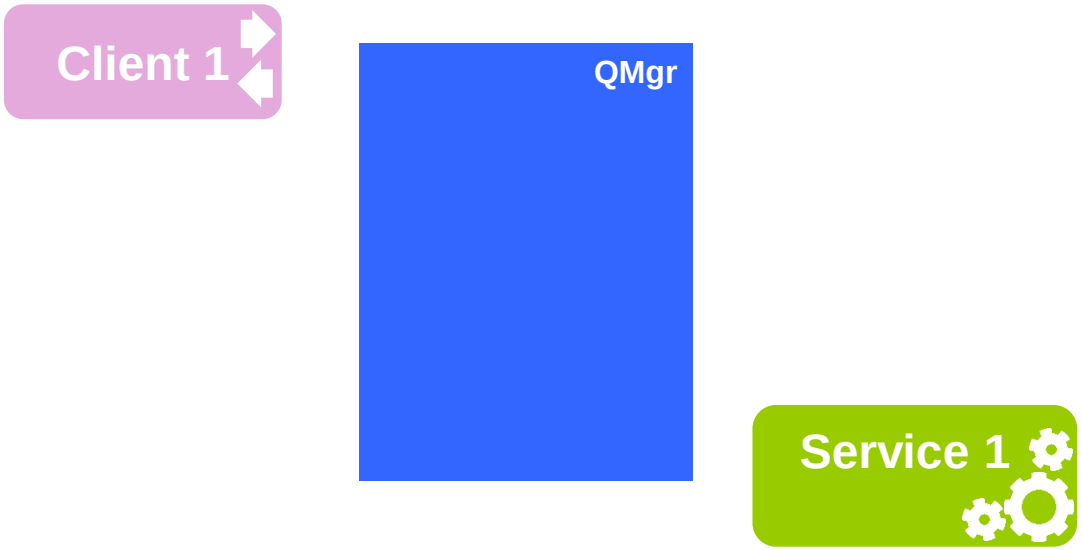
David Ware
IBM MQ

Agenda

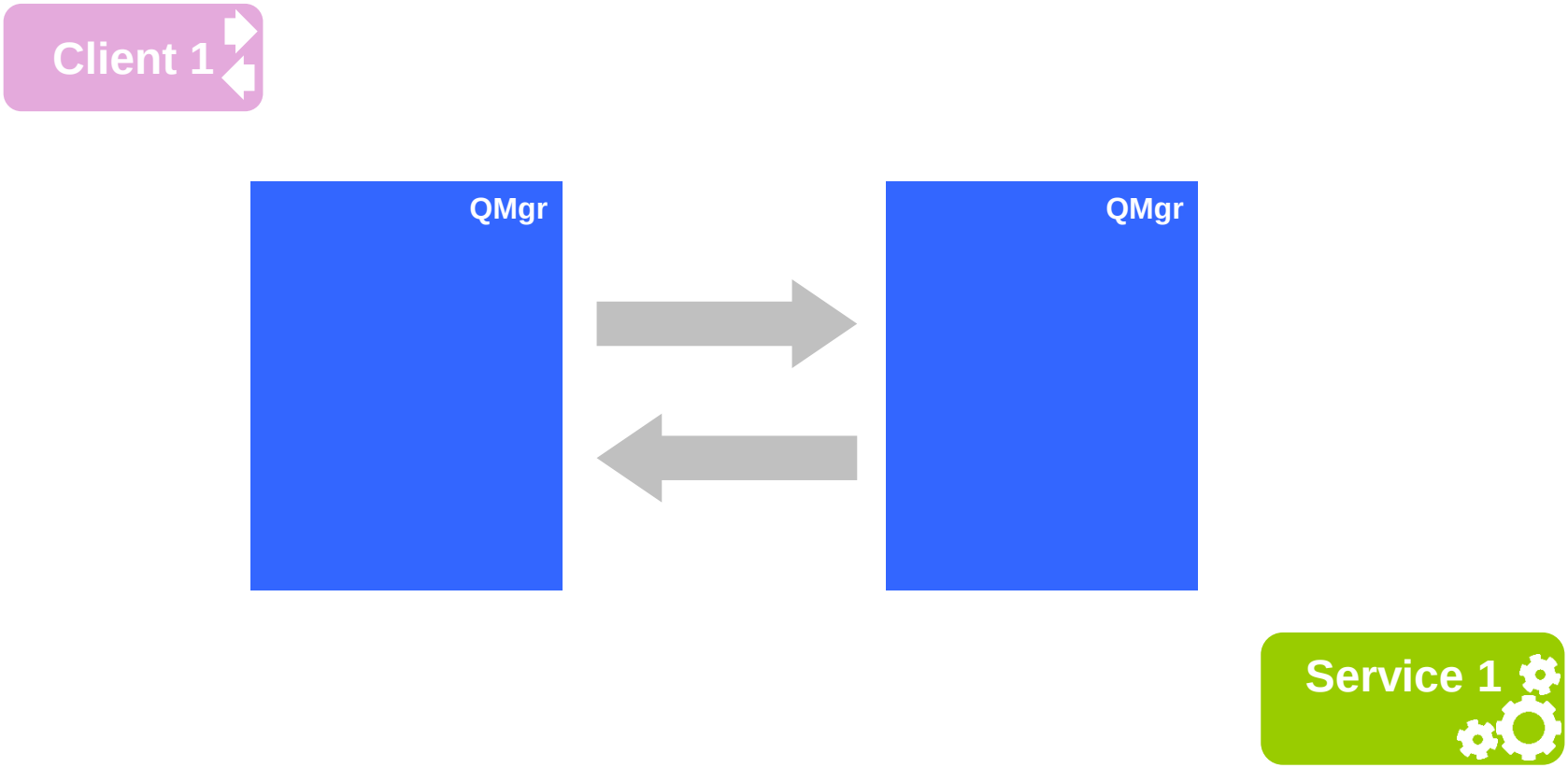
- My first cluster
  - Starting to scale
- Service availability
  - Routing around failures
- Location dependency
  - Active active scenarios with ties to home
- Avoiding interference
  - What are we sharing?
- When things go wrong
  - Levels of DR and clustering implications

My First Cluster

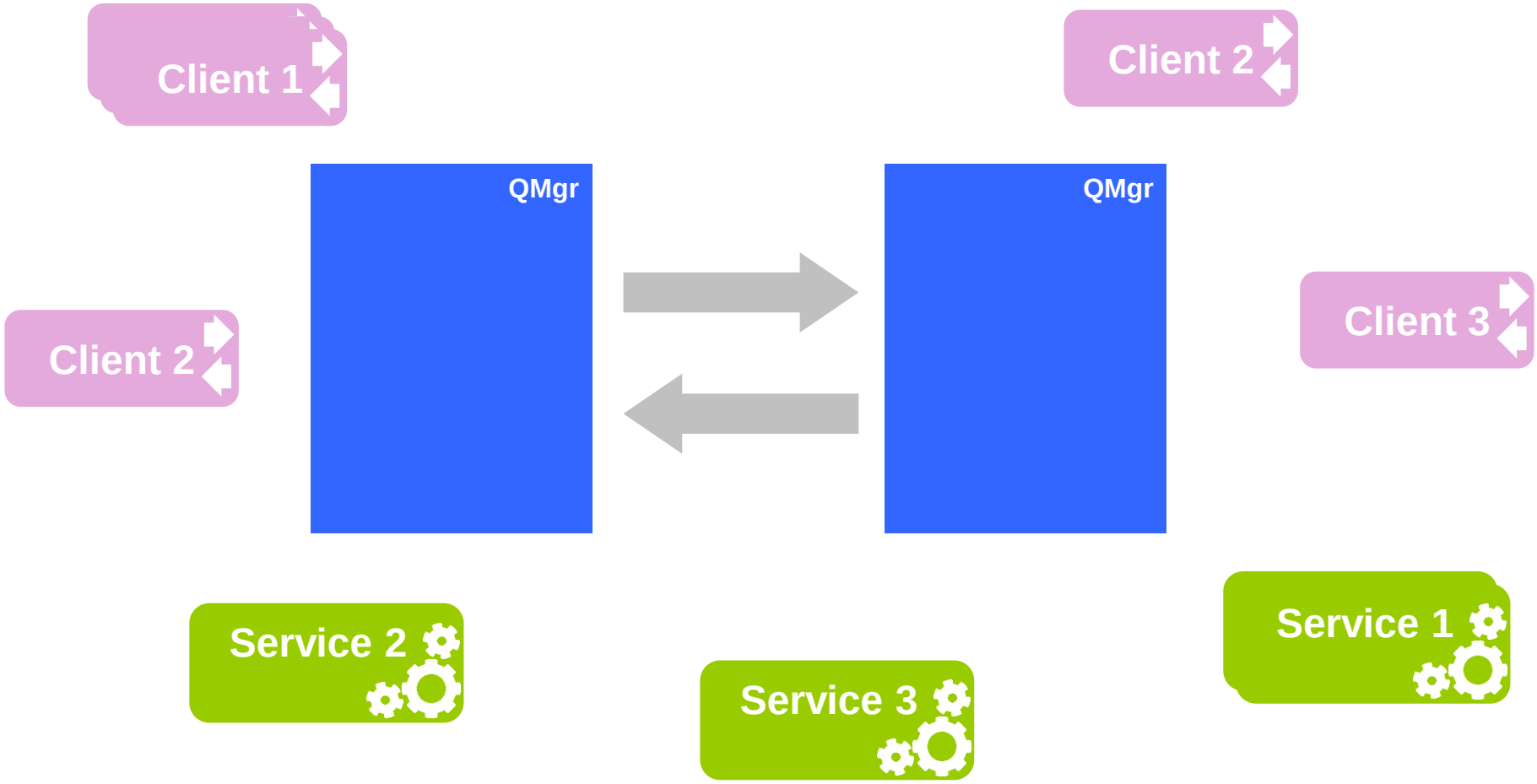
Where it all begins…



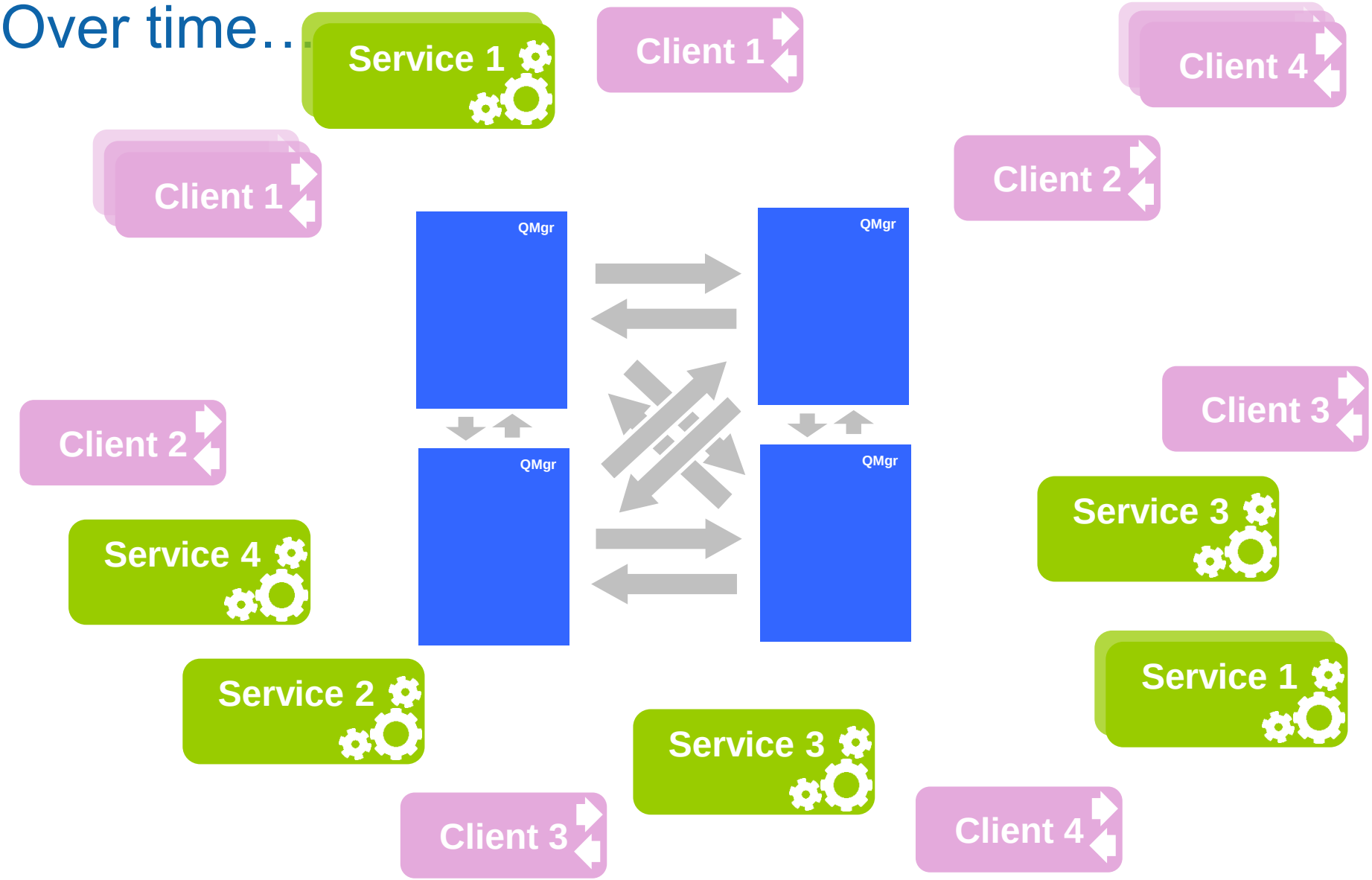
Where it all begins…



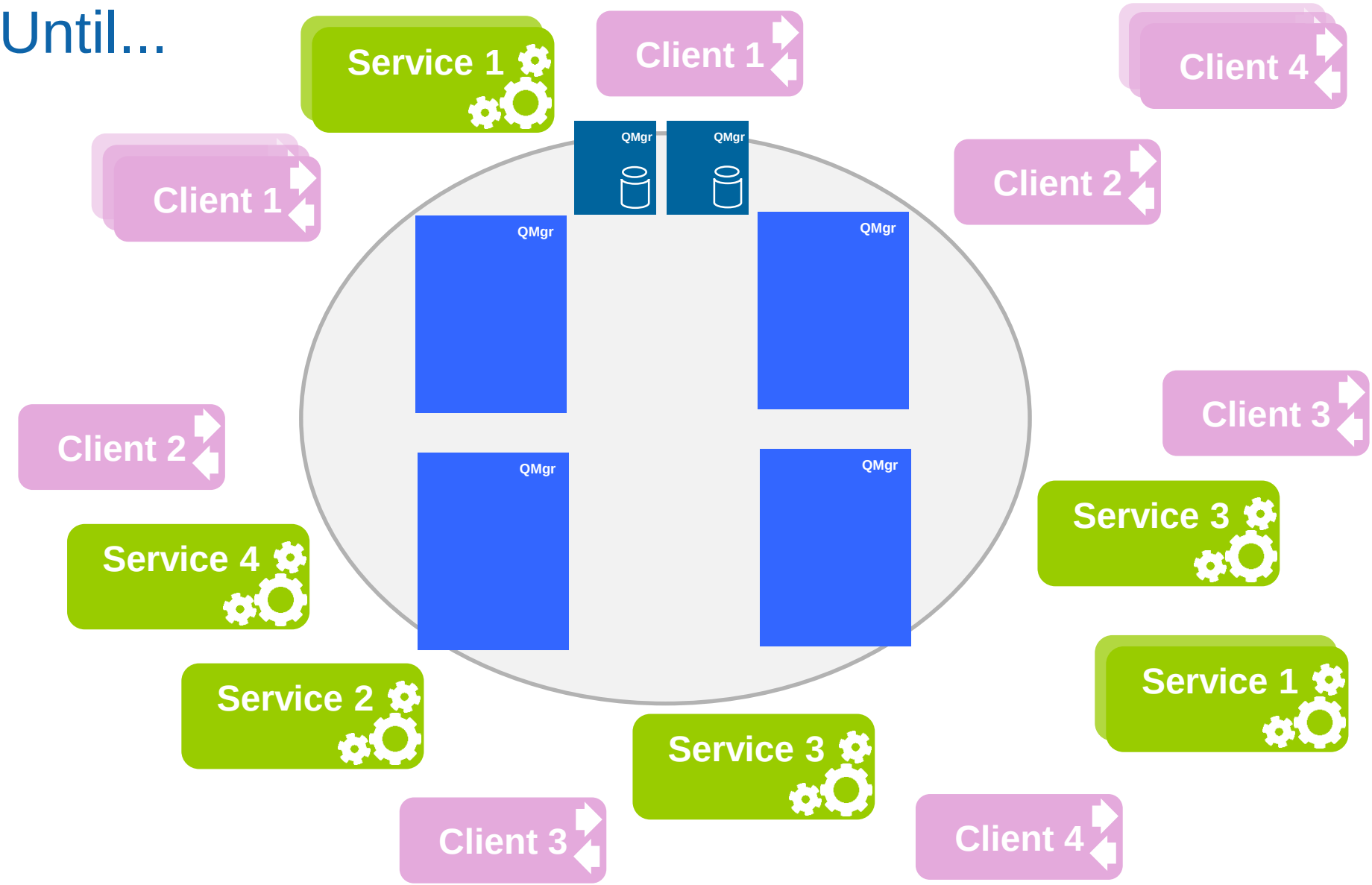
Over time…



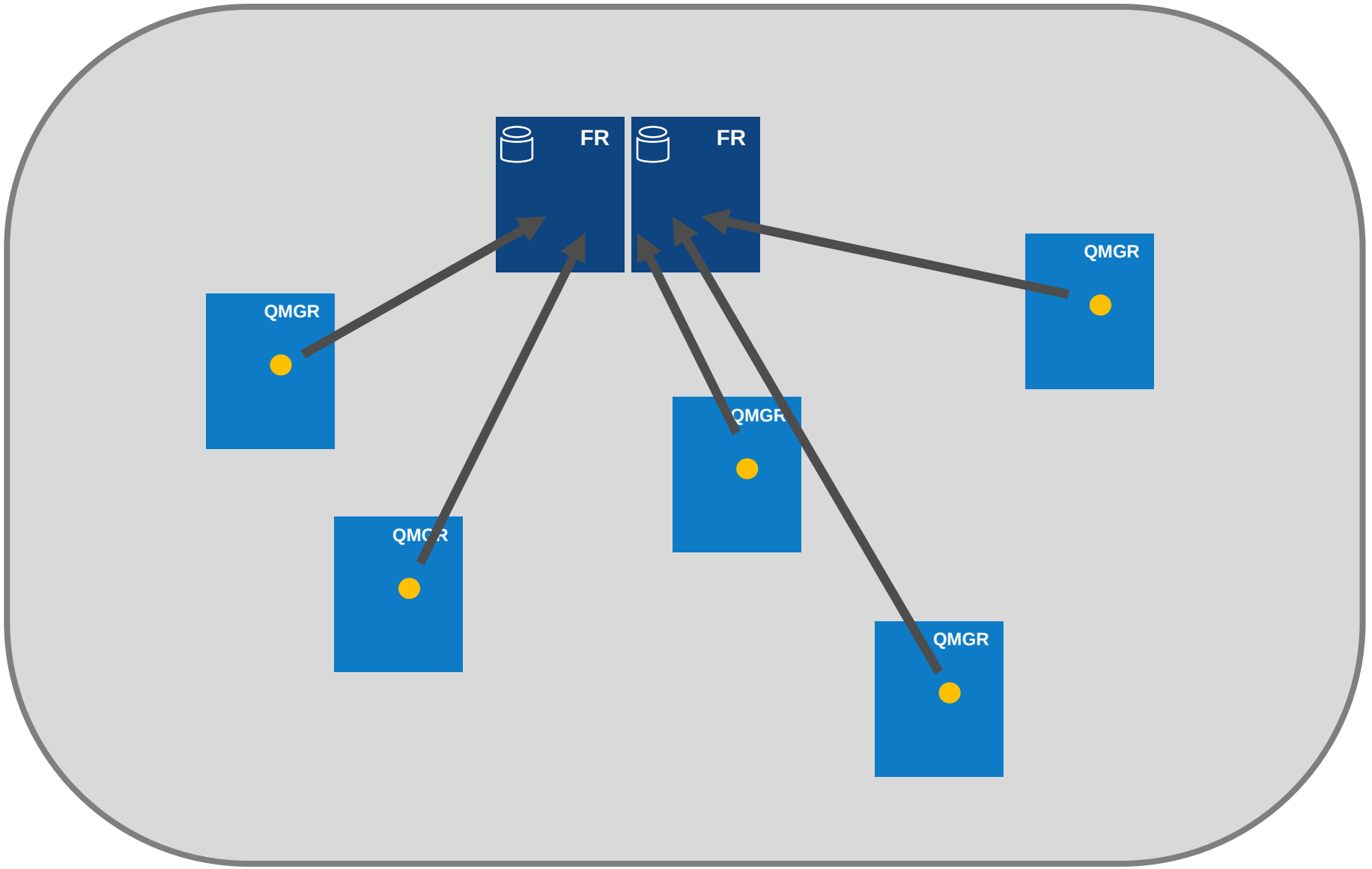
Over time..



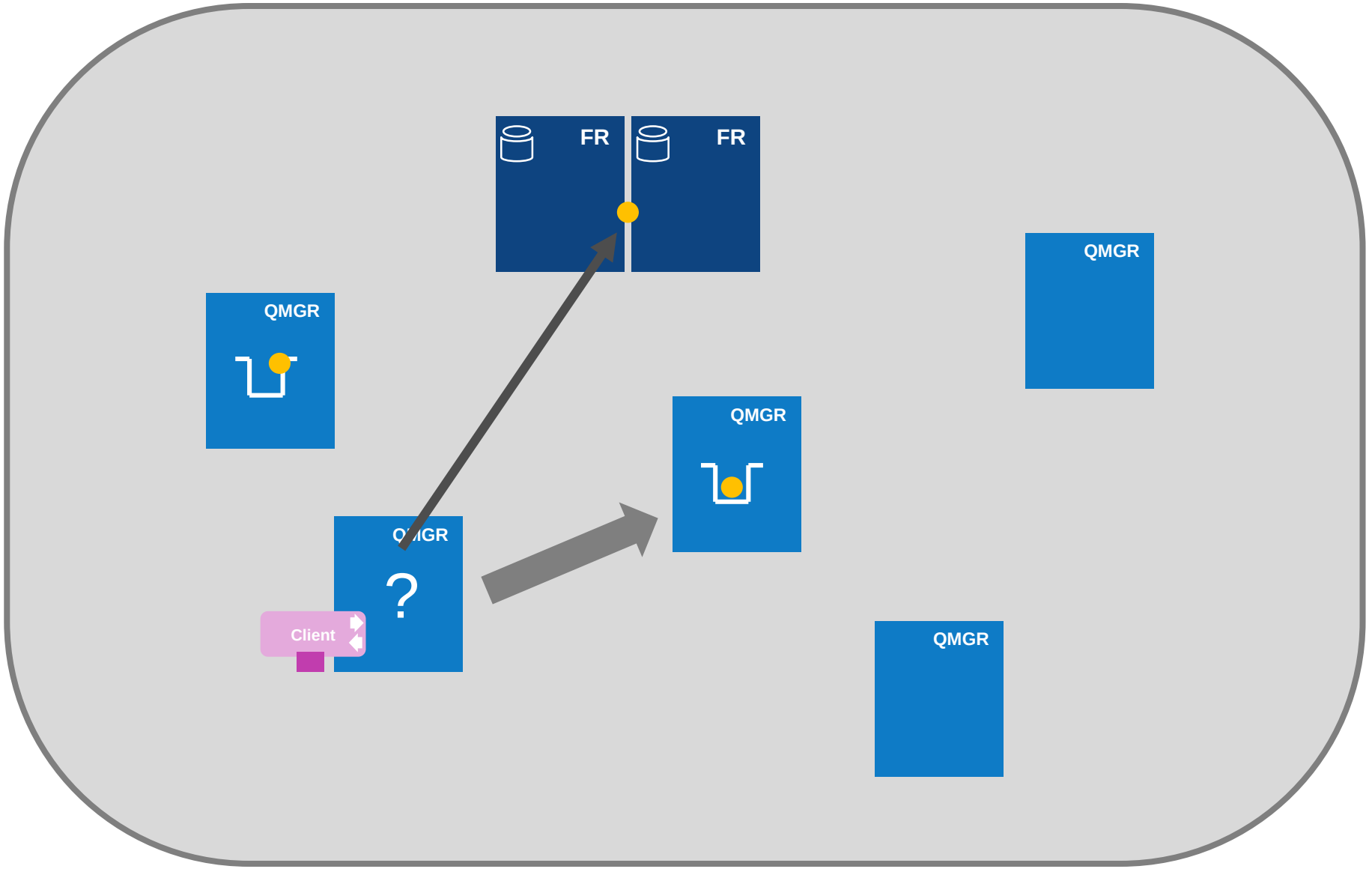
Until...



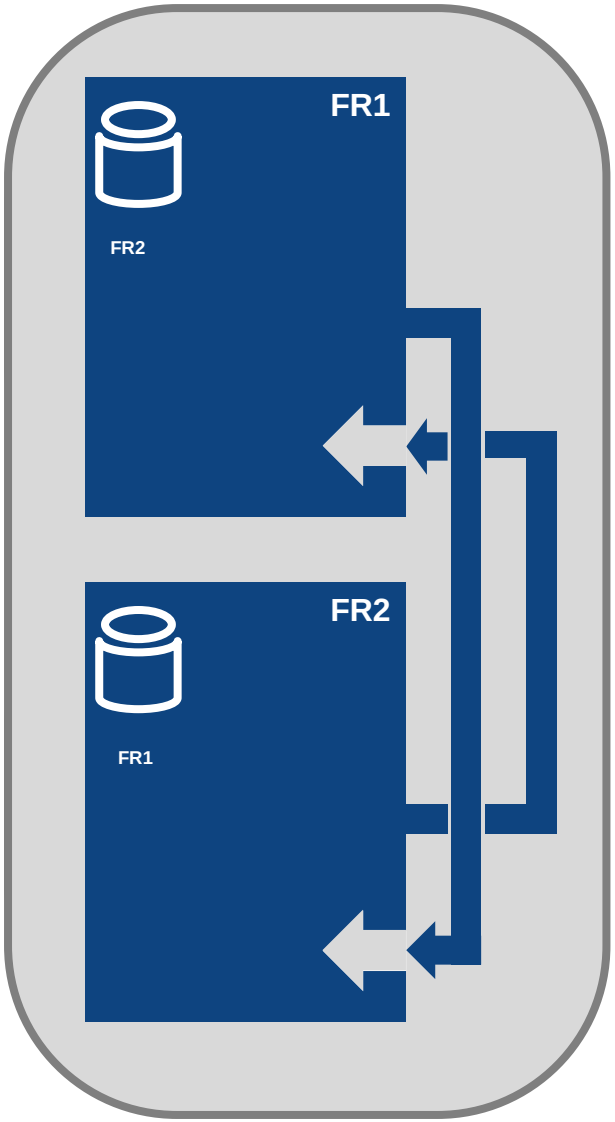
Building the cluster



Building the cluster



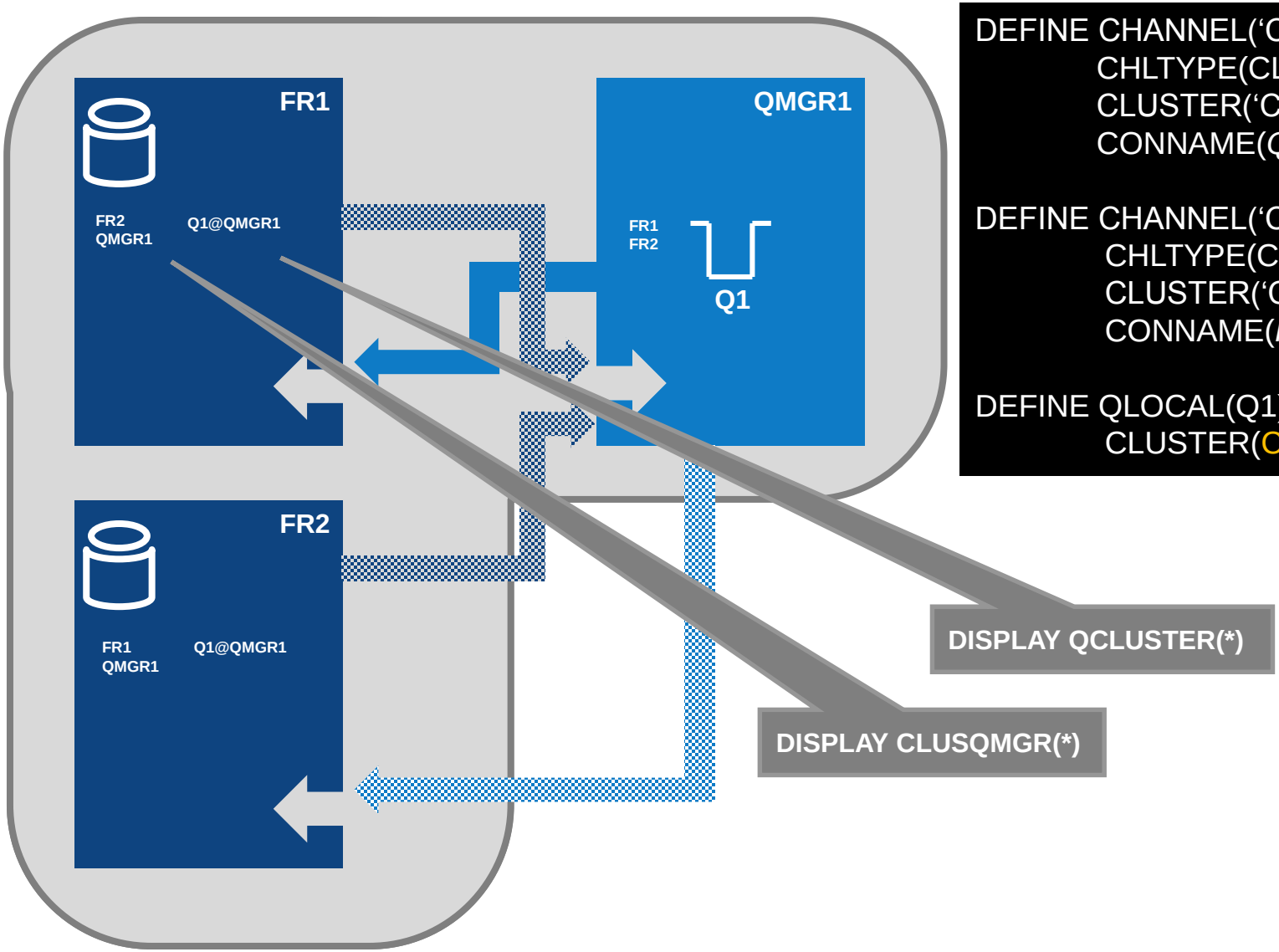
Step 1: Create your two full repositories




```
ALTER QMGR REPOS('CLUS1')

DEFINE CHANNEL('CLUS1.FR1') CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1') CONNAME(FR1 location)

DEFINE CHANNEL('CLUS1.FR2') CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1') CONNAME(FR2 location)
```

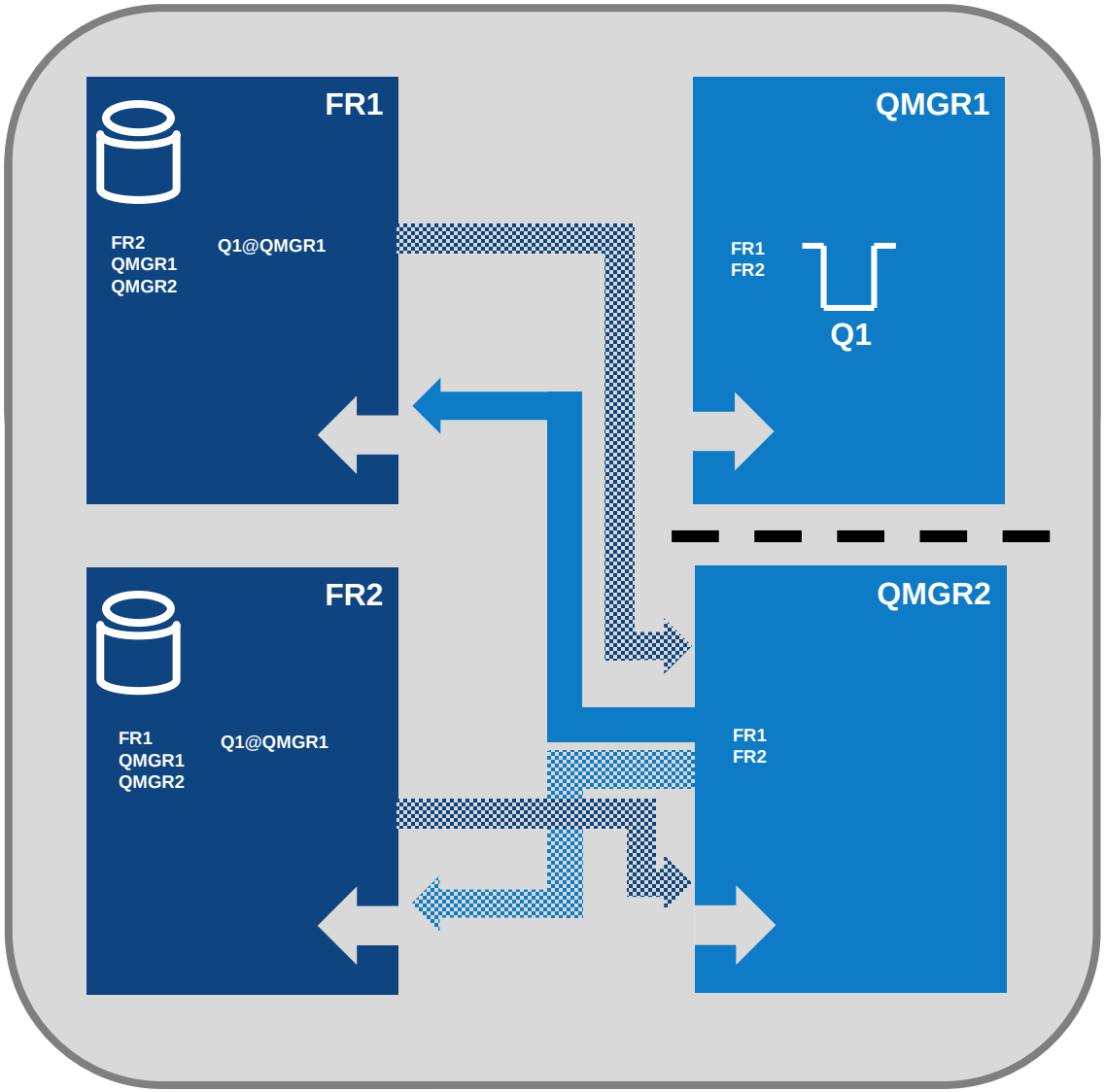
```
ALTER QMGR REPOS('CLUS1')

DEFINE CHANNEL('CLUS1.FR2') CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1') CONNAME(FR2 location)

DEFINE CHANNEL('CLUS1.FR1') CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1') CONNAME(FR1 location)
```


Step 2: Add in more queue managers




```
DEFINE CHANNEL('CLUS1.QMGR1')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR1 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)

DEFINE QLOCAL(Q1)
       CLUSTER(CLUS1)
```

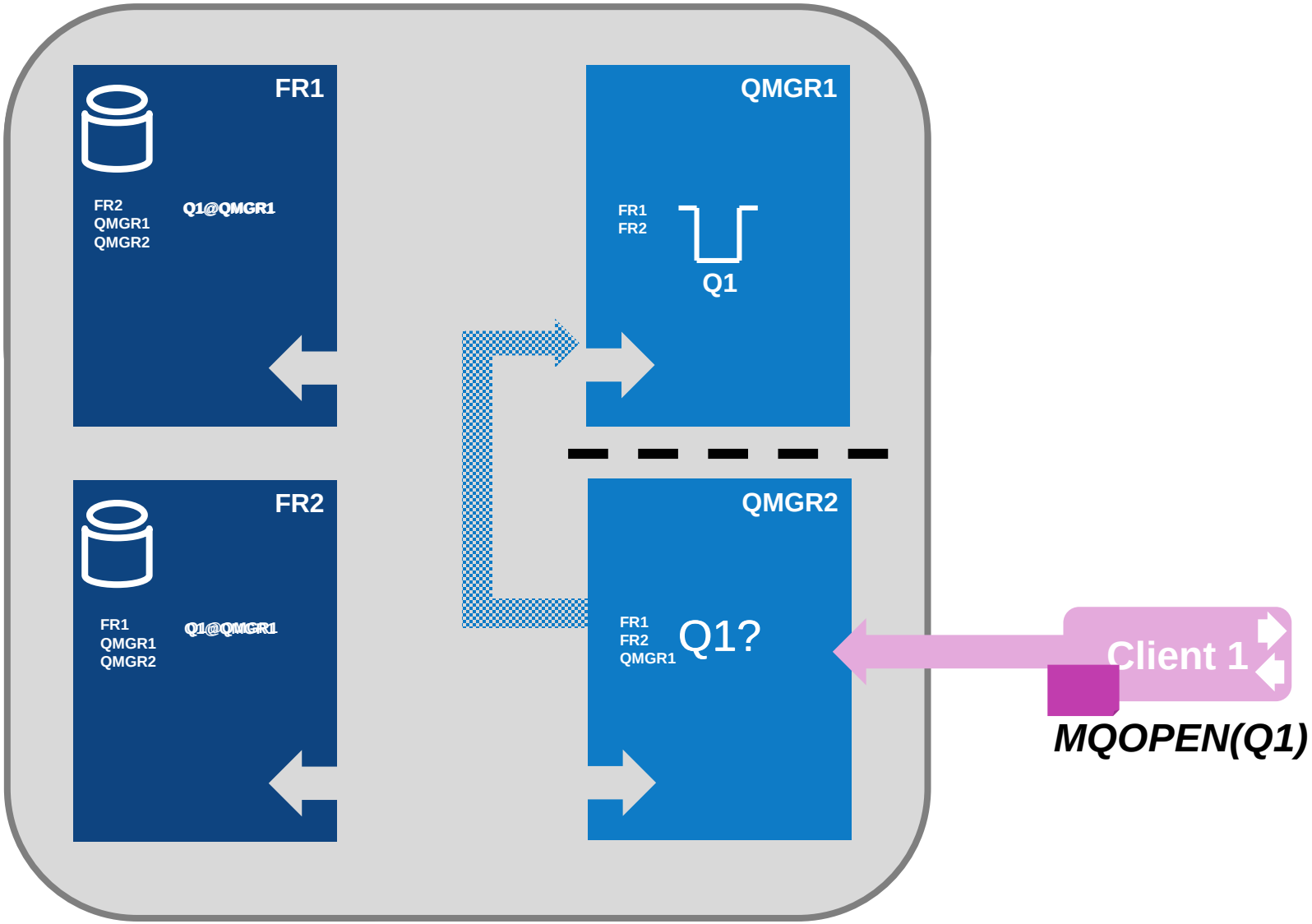

Step 2: Add in more queue managers



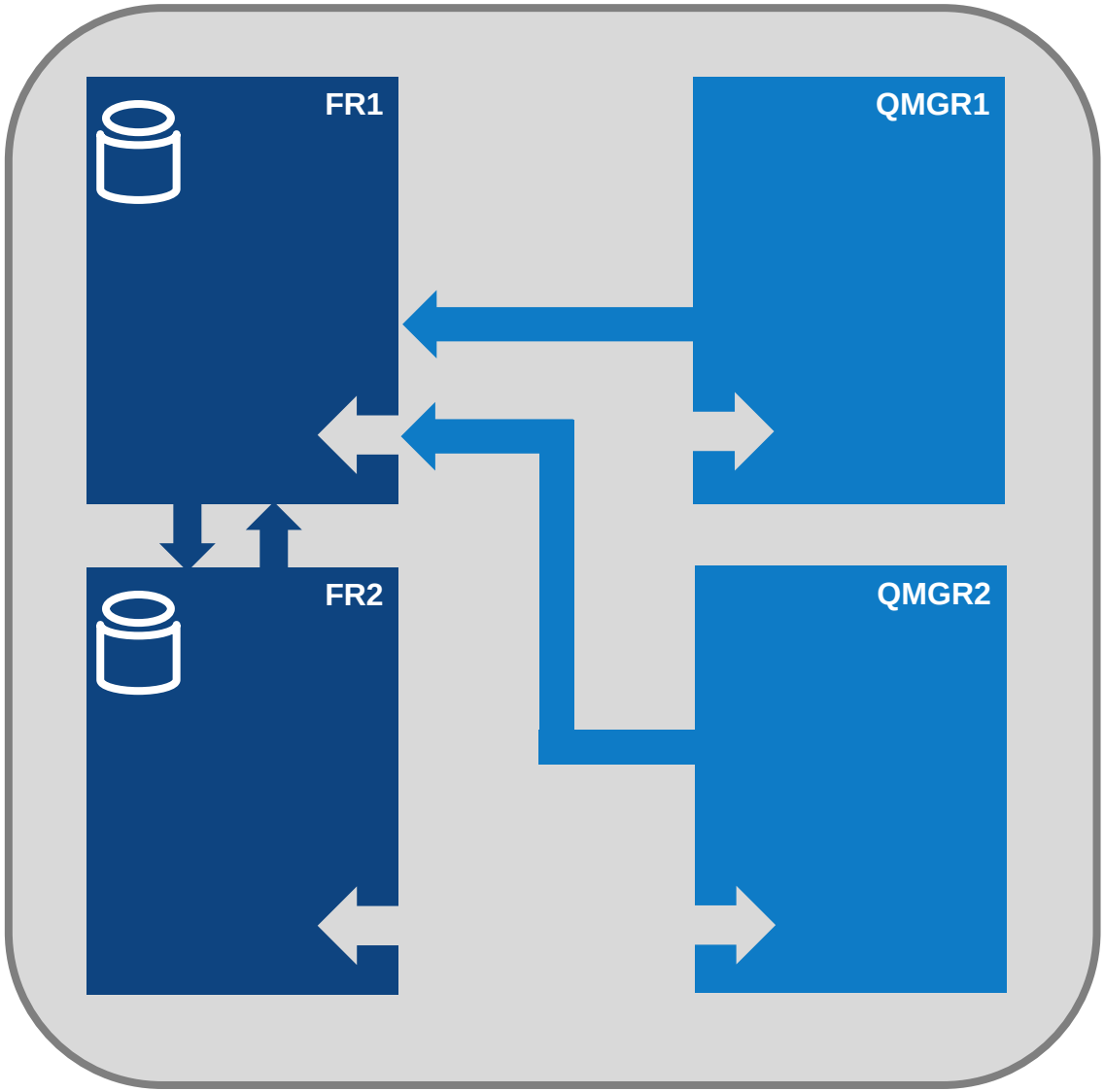

```
DEFINE CHANNEL('CLUS1.QMGR1')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR1 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)

DEFINE QLOCAL(Q1)
       CLUSTER(CLUS1)
```

```
DEFINE CHANNEL('CLUS1.QMGR2')
       CHLTYPE(CLUSRCVR)
       CLUSTER('CLUS1')
       CONNAME(QMGR2 location)

DEFINE CHANNEL('CLUS1.FR1')
       CHLTYPE(CLUSSDR)
       CLUSTER('CLUS1')
       CONNAME(FR1 location)
```


Step 3: Start sending messages



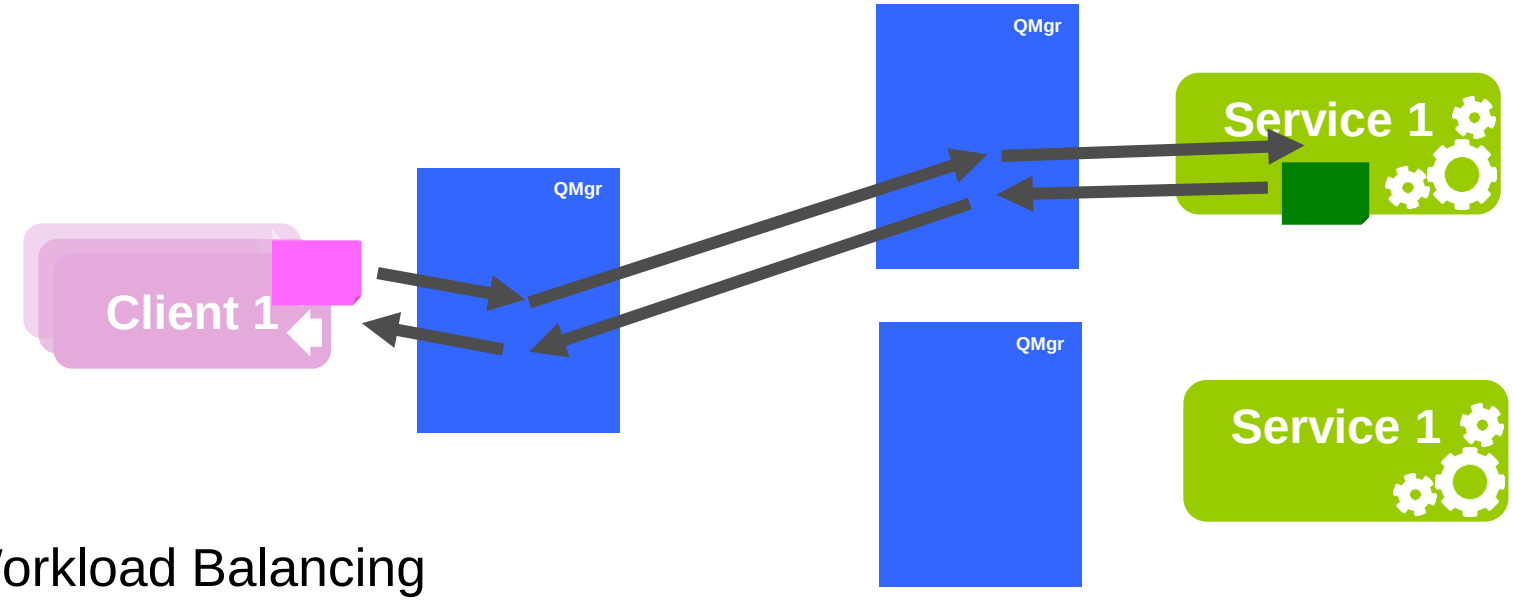
So all you needed…



- Two full repository queue managers
- A cluster receiver channel each
- A single cluster sender each
- No need to manage pairs of channels between each queue manager combination or their transmission queues
- No need for remote queue definitions

That’s just for starters…

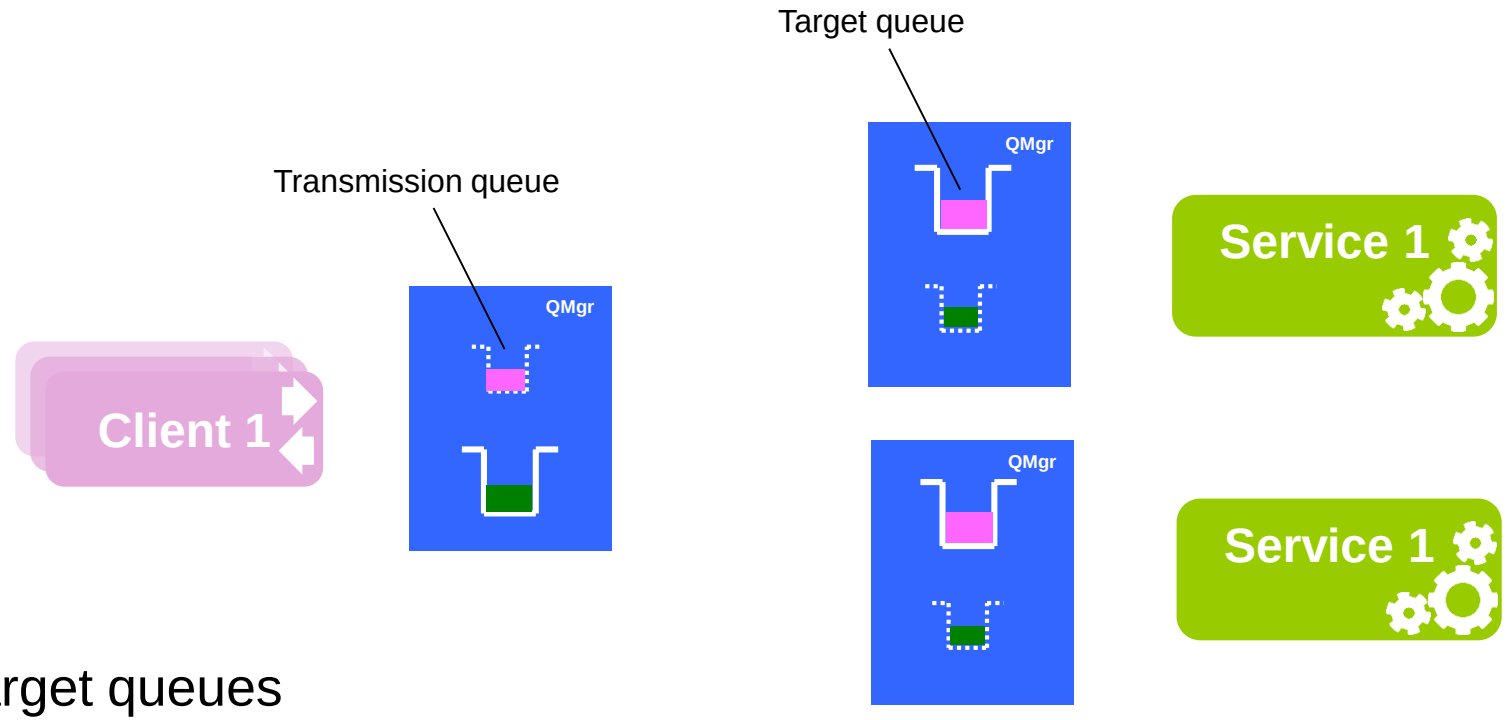
Starting to scale horizontally…



- Workload Balancing
- Service Availability

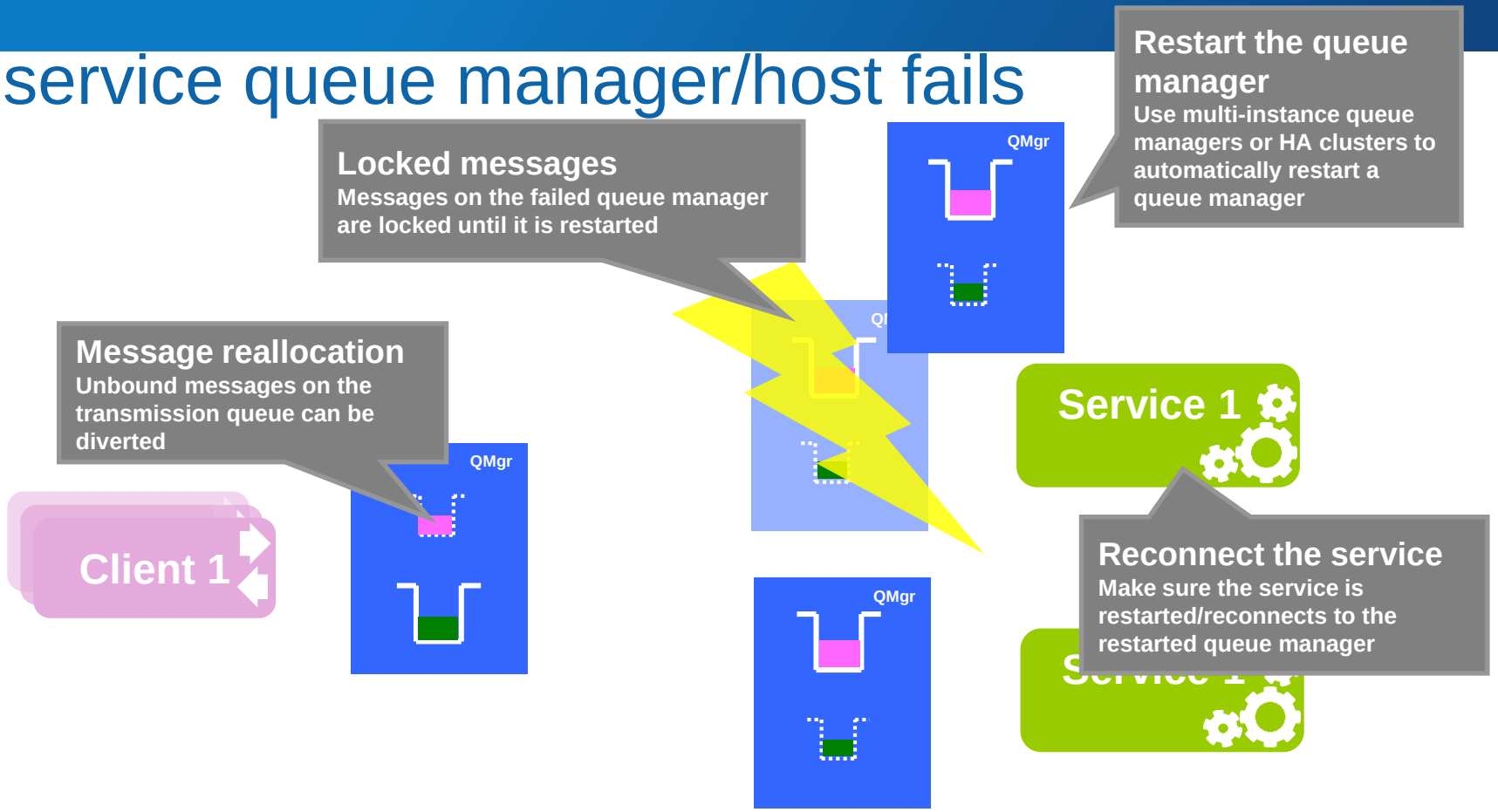
Availability

Where can the messages get stuck?



- Target queues
- Transmission queues

The service queue manager/host fails

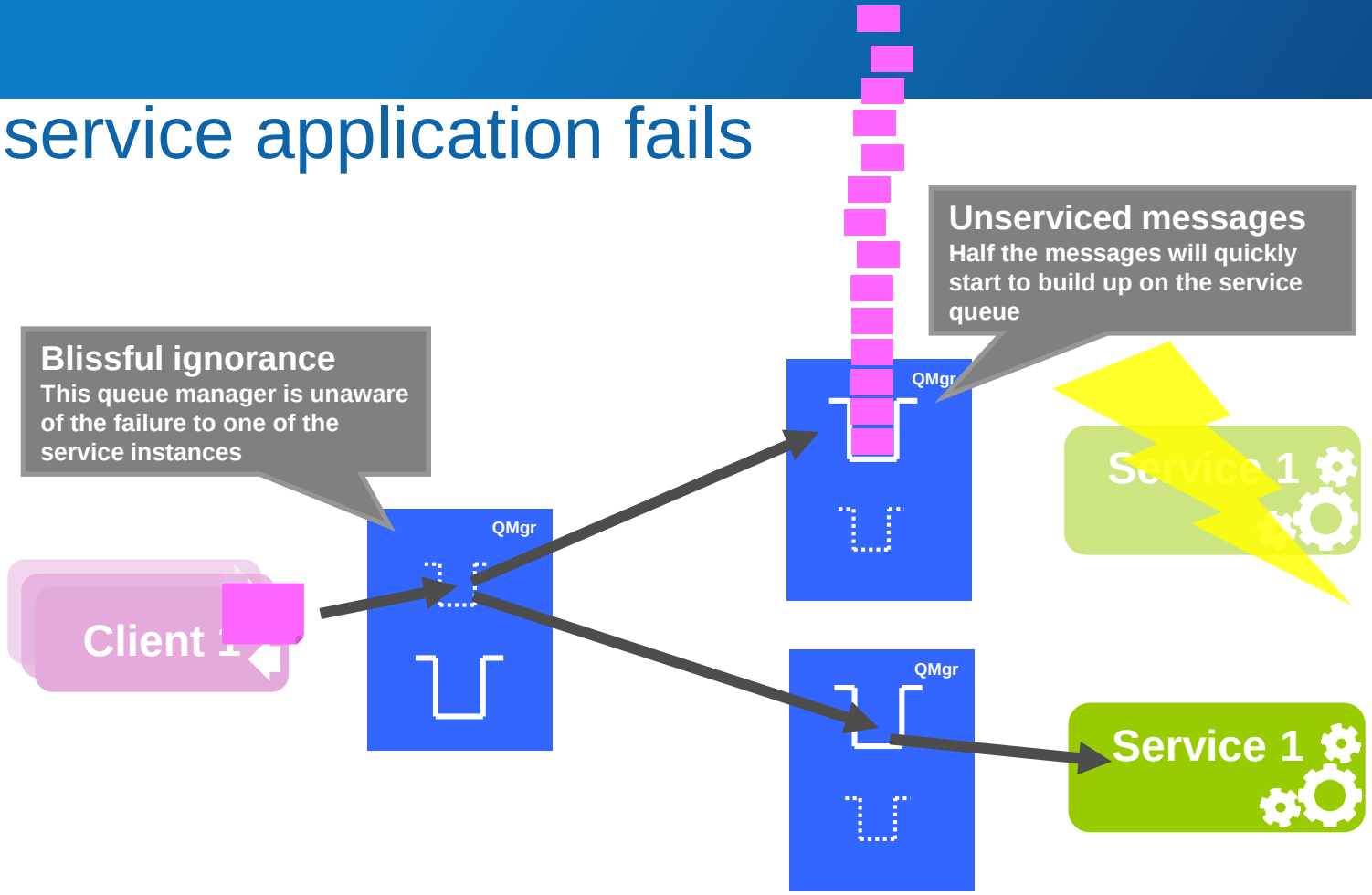


When a queue manager fails:

- Ensure messages are not **bound** to it
- Restart it to release queued messages

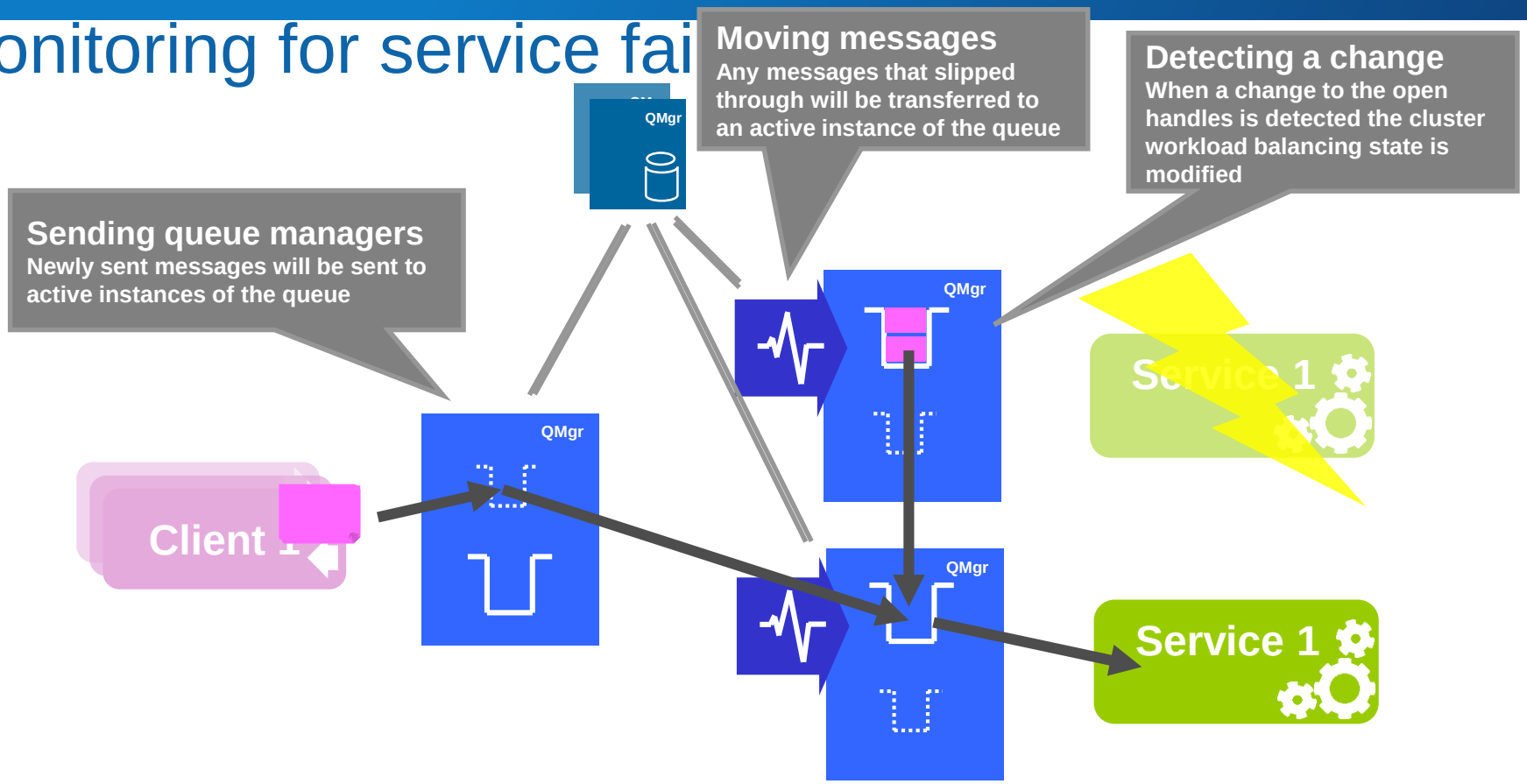
Service application availability

The service application fails



- Cluster workload balancing does not take into account the availability of receiving applications.
- Or a build up of messages.

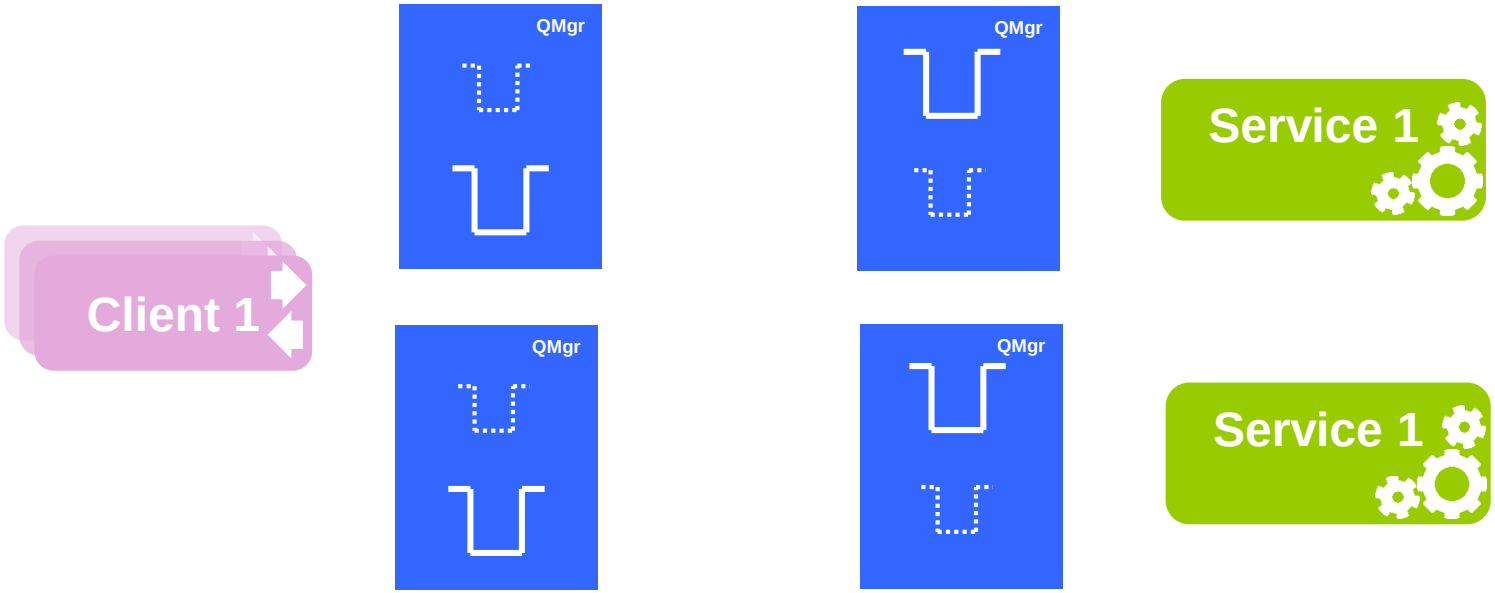
Monitoring for service fai



- WebSphere MQ provides a sample monitoring service
- Regularly checks for attached consuming applications
- Generally suited to ***steady state*** service applications

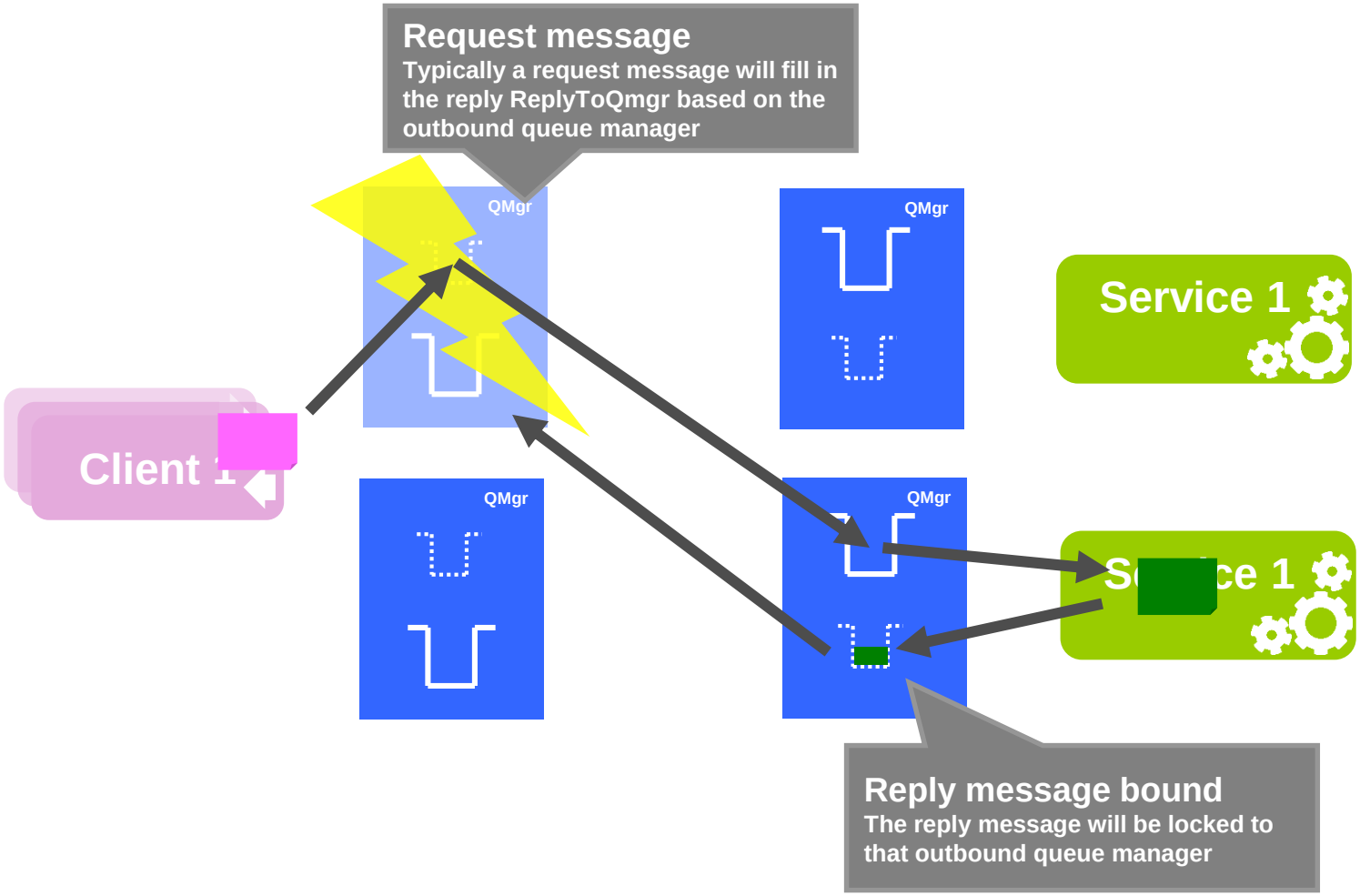
Client failures

Client availability



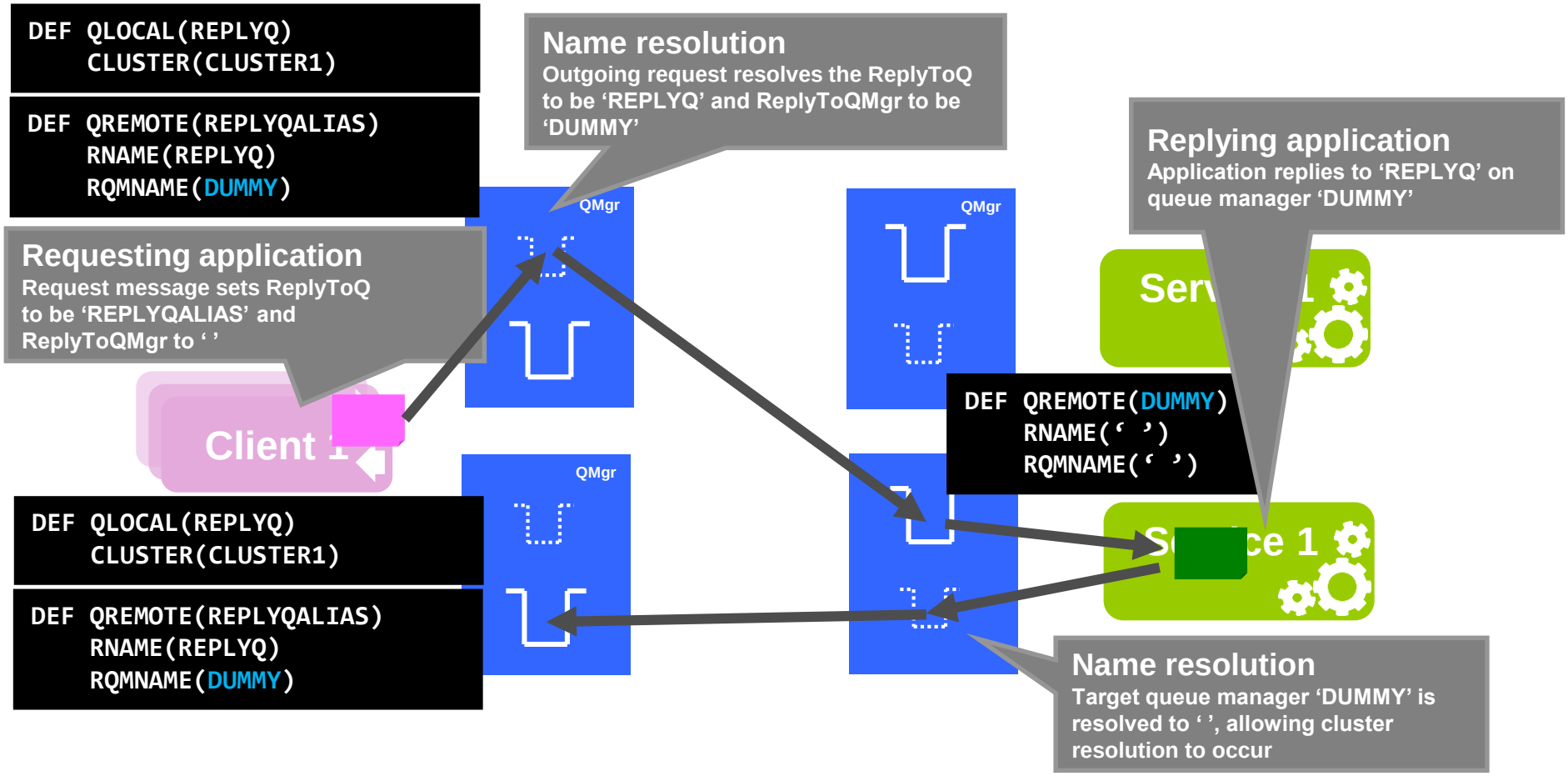
- Multiple locations for a client to connect to
  - Allows new requests when one queue manager is unavailable.
- What happens to replies after a failure?

Client host failure with an in flight request/response



- Reply messages are bound to the originating queue manager, with no ability to redirect.

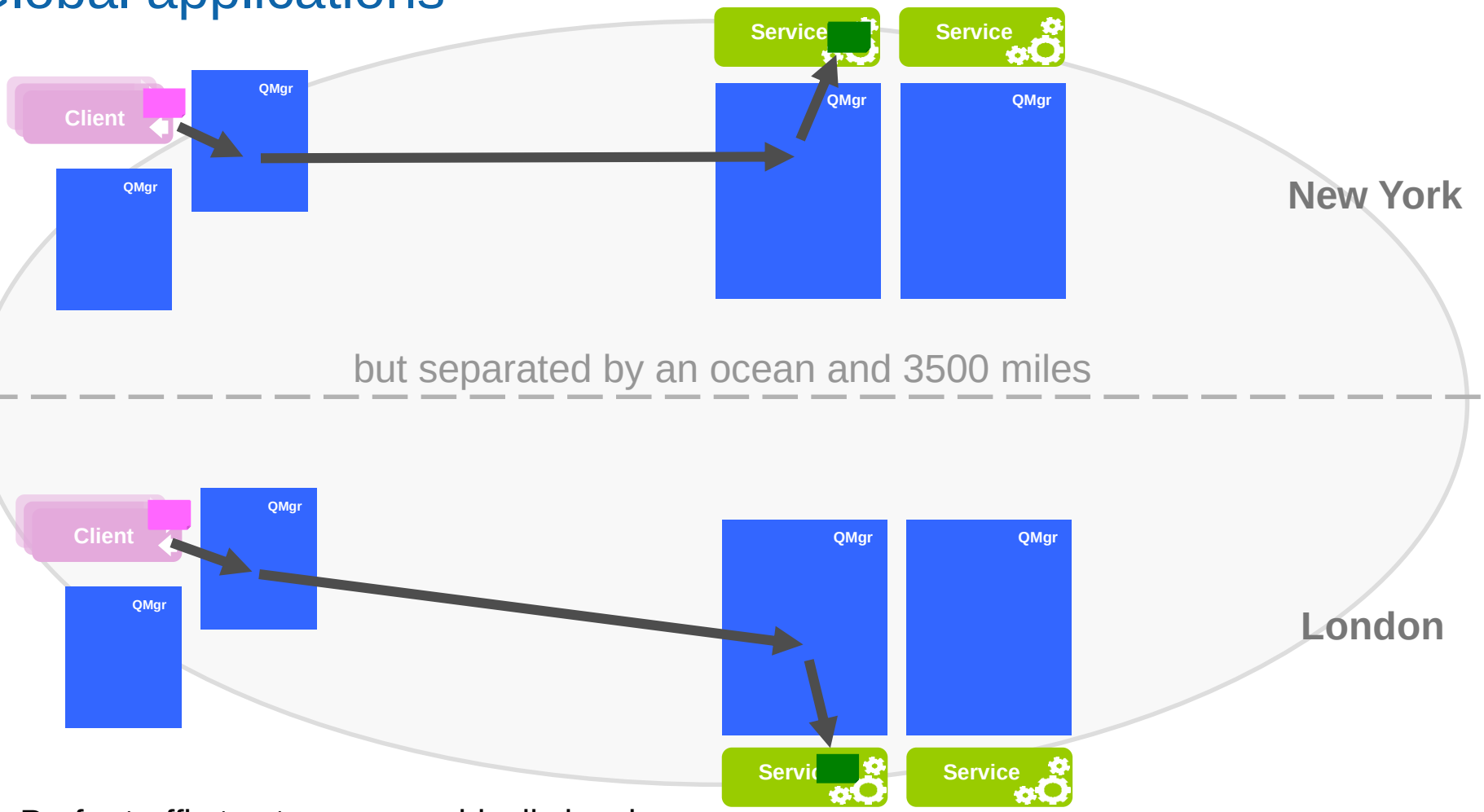
Client host failure with an in flight request/response



- ***Reply-to queue aliases*** and ***reply-to queue manager aliases*** can be used to blank out the outbound resolution of the ReplyToQMgr field.
- *Typically, under normal running, you require the originating queue manager to receive the replies, cluster workload balancing configuration from before can help to provide this.*

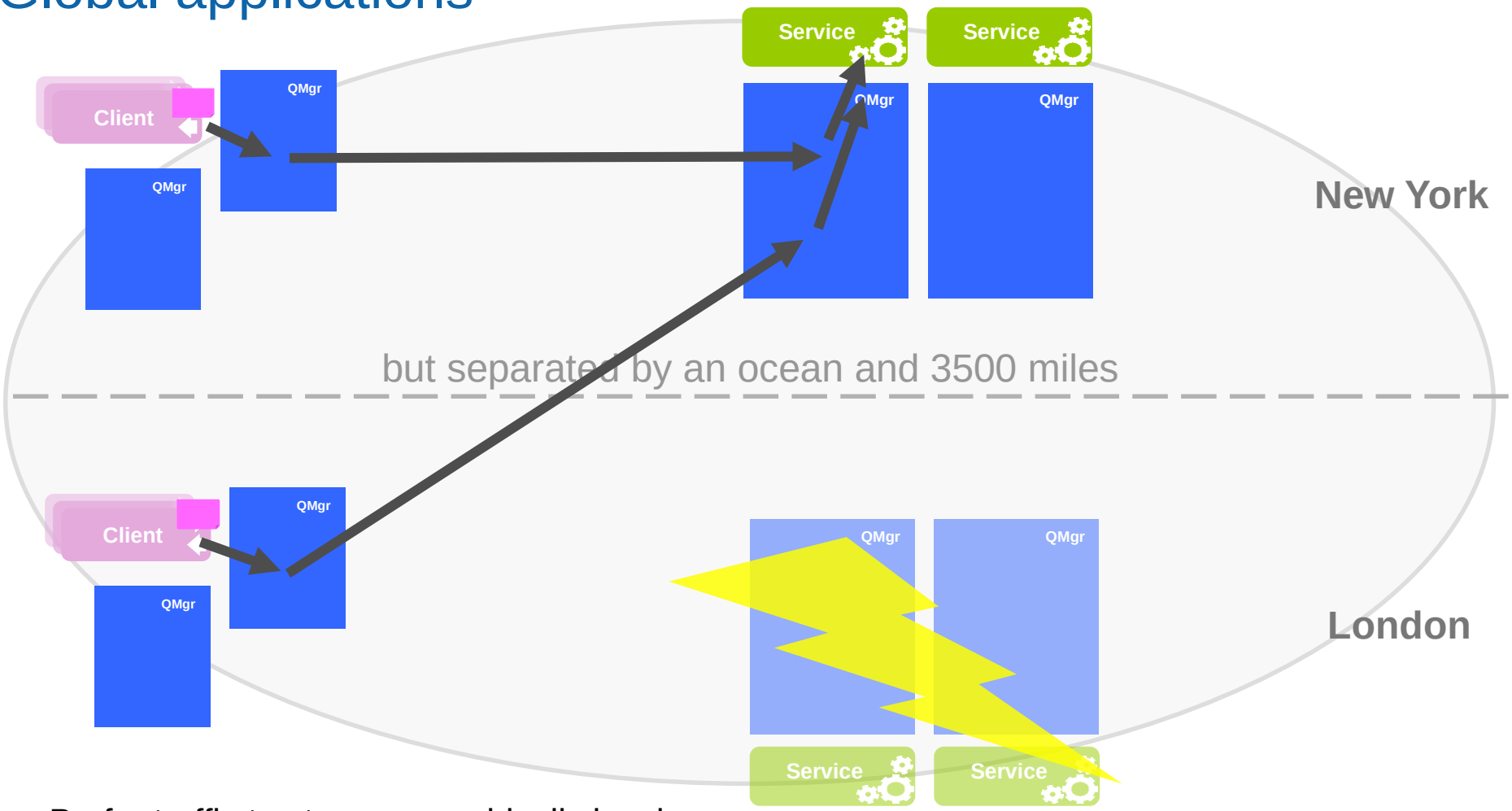
Location Dependency

Global applications



- Prefer traffic to stay geographically local

Global applications



- Prefer traffic to stay geographically local
- Except when you have to look further afield
- Clusters can be used to span geographies

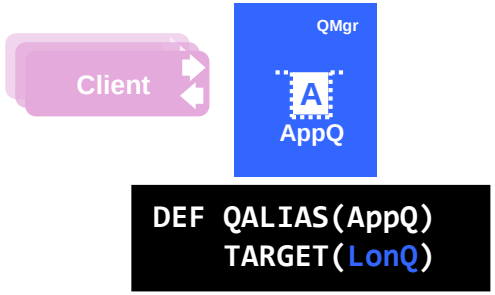
Setting this up

One cluster


```
DEF QALIAS(NYQ)
    TARGET(ReqQ)
    CLUSTER(Global)
    CLWLPRTY(9)
```

```
DEF QALIAS(LonQ)
    TARGET(ReqQ)
    CLUSTER(Global)
    CLWLPRTY(4)
```

New York



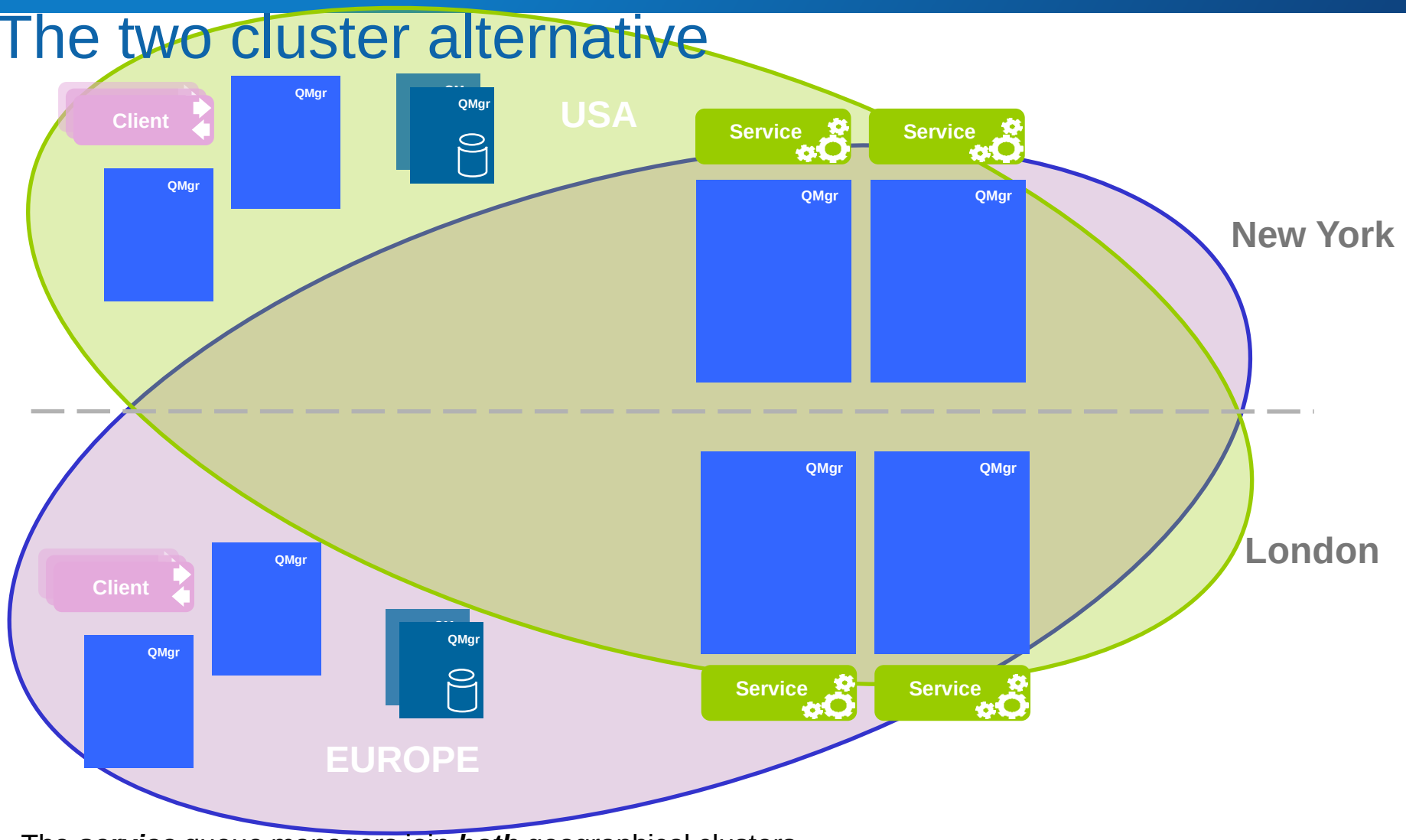

```
DEF QALIAS(LonQ)
    TARGET(ReqQ)
    CLUSTER(Global)
    CLWLPRTY(9)
```

```
DEF QALIAS(NYQ)
    TARGET(ReqQ)
    CLUSTER(Global)
    CLWLPRTY(4)
```

London

- Clients always open ***AppQ***
- Local alias determines the preferred region
- Cluster workload priority is used to target geographically local cluster aliases
- Use of CLWLPRTY enables automatic failover
  - CLWLRANK can be used for manual failover

The two cluster alternative



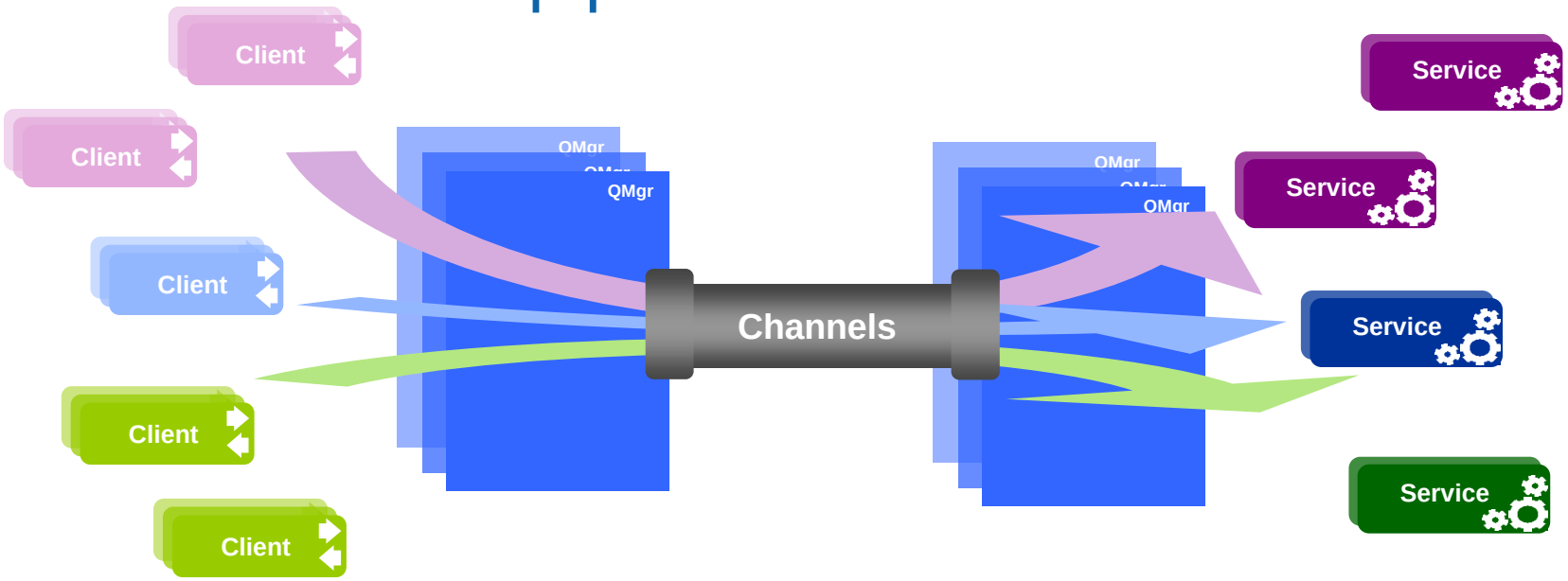
- The ***service*** queue managers join ***both*** geographical clusters
  - Each with separate cluster receivers for each cluster, at ***different cluster priorities***. Queues are clustered in ***both*** clusters.
- The ***client*** queue managers are in their ***local*** cluster only.

Avoiding interference

The cluster as a pipe

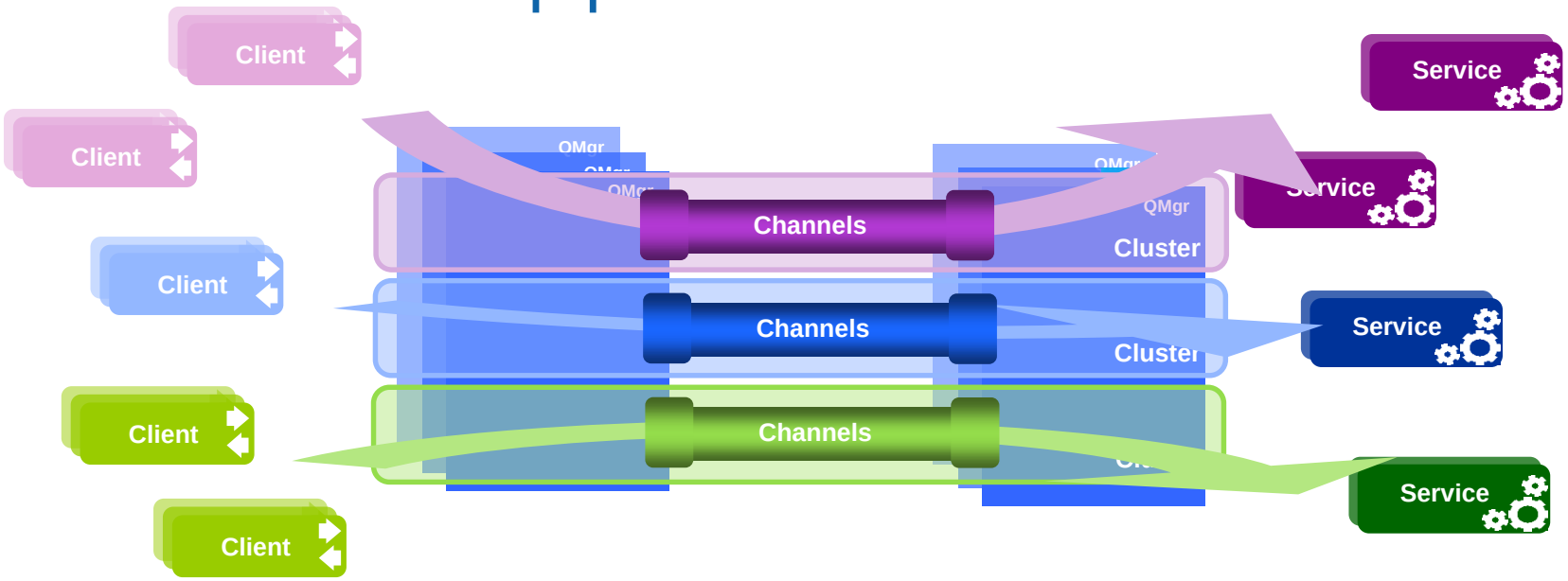
- Often a WebSphere MQ backbone will be used for multiple types of traffic

The cluster as a pipe



- Often a WebSphere MQ backbone will be used for multiple types of traffic
- When using a single cluster and the same queue managers, messages all share the same channels
- Even multiple cluster receiver channels in the same cluster will not separate out the different traffic types

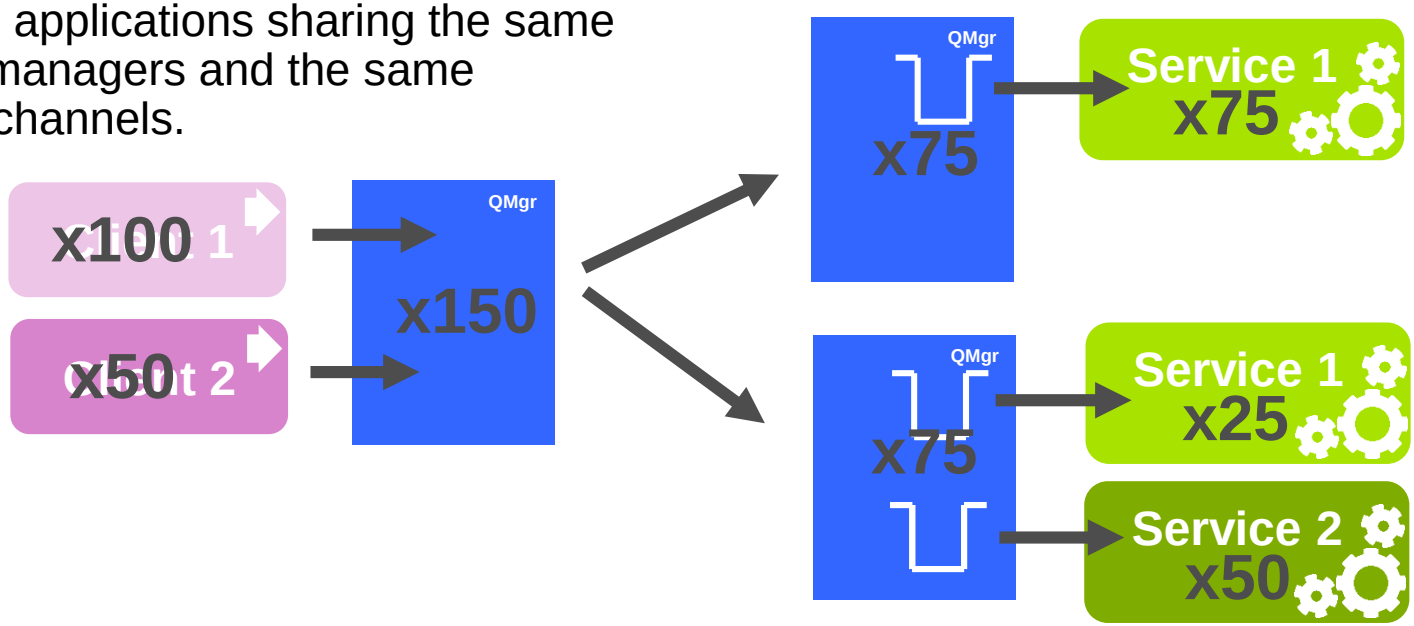
The cluster as a pipe



- Often a WebSphere MQ backbone will be used for multiple types of traffic
- When using a single cluster and the same queue managers, messages all share the same channels
- Even multiple cluster receiver channels in the same cluster will not separate out the different traffic types
- Multiple overlaid clusters with different channels enable separation

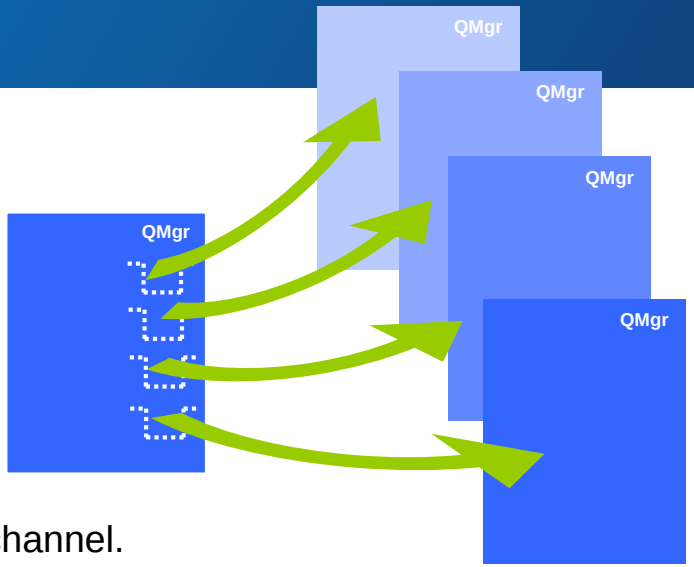
Workload balancing level interference

- Multiple applications sharing the same queue managers and the same cluster channels.



- Cluster workload balancing is at the **channel** level.
  - Messages sharing the same channels, but to different target queues will be counted together.
- The two channels here have an even 50/50 split of messages…
- *…but the two instances of Service 1 do not!*
- *Split Service 1 and Service 2 queues out into separate clusters, queue managers or customise workload balancing logic.*

Cluster transmit queue



- **Separation of Message Traffic**
  - With a single transmission queue there is potential for pending messages for cluster *ChannelA* to interfere with messages pending for cluster *ChannelB*
- **Management of messages**
  - Use of queue concepts such as *MAXDEPTH* not useful when using a single transmission queue for more than one channel.
- **Monitoring**
  - Tracking the number of messages processed by a cluster channel currently difficult/impossible using queue.
- **Performance?**
  - In reality a shared transmission queue is not always the bottleneck, often other solutions to improving channel throughput *(e.g. multiple cluster receiver channels)* are really what's needed.
- **A much requested feature…**
  - **Multiple cluster transmission queues**

V7.5 Distributed **V8** z/OS & IBM i

Multiple cluster transmit queues: Automatic

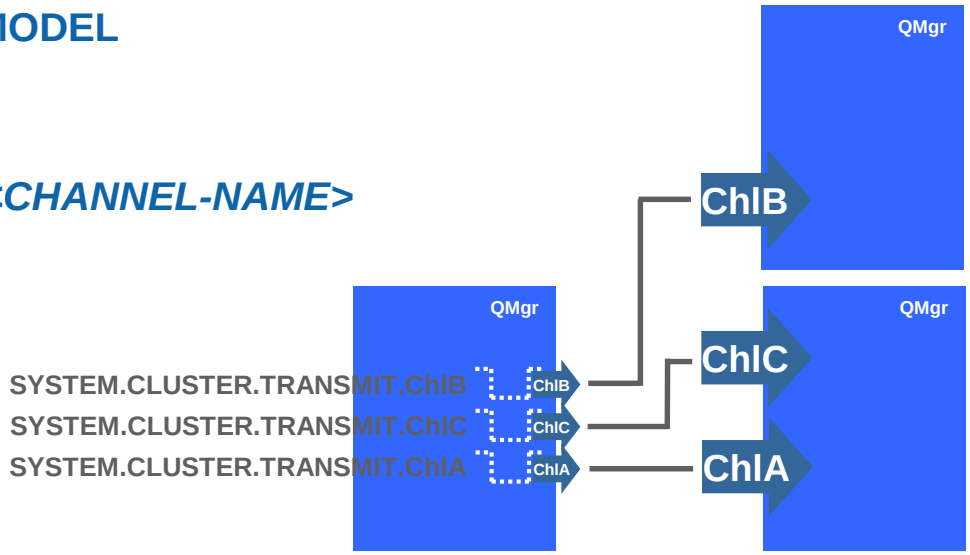
- Configured on the ***sending queue manager***, not the owners of the cluster receiver channel definitions.

- Queue Manager switch to automatically create a dynamic transmission queue per cluster sender channel.
  ALTER QMGR DEFCLXQ(SCTQ | CHANNEL)

- Dynamic queues based upon model queue.
  SYSTEM.CLUSTER.TRANSMIT.MODEL

- Well known queue names.

  SYSTEM.CLUSTER.TRANSMIT.*<CHANNEL-NAME>*



Multiple cluster transmit queues: Manual

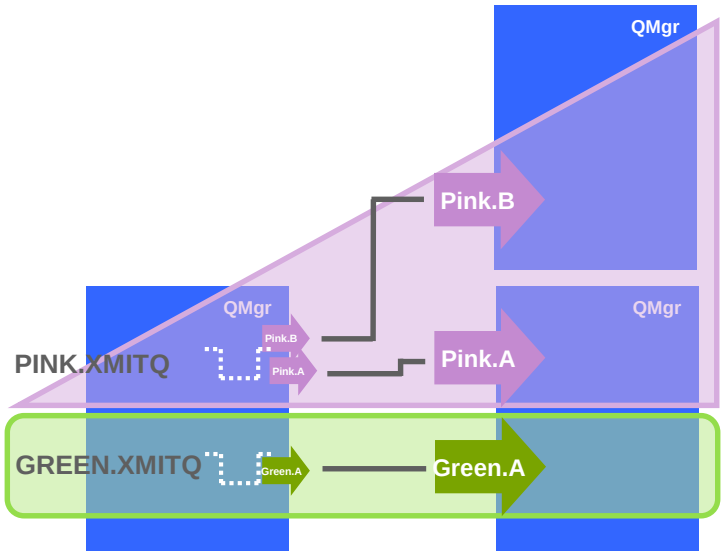
- *Still* configured on the ***sending queue manager***, not the owners of the cluster receiver channel definitions.

- Administratively define a transmission queue and configure which cluster sender channels will use this transmission queue.

  DEFINE QLOCAL(GREEN.XMITQ) CLCHNAME(GREEN.*) USAGE(XMITQ)

  - Set a channel name *pattern* in CLCHNAME
  - Single/multiple channels (wildcard)
    - E.g. all channels for a specific cluster (assuming a suitable channel naming convention!)

- Any cluster sender channel not covered by a manual transmission queue defaults to the DEFCLXQ behaviour



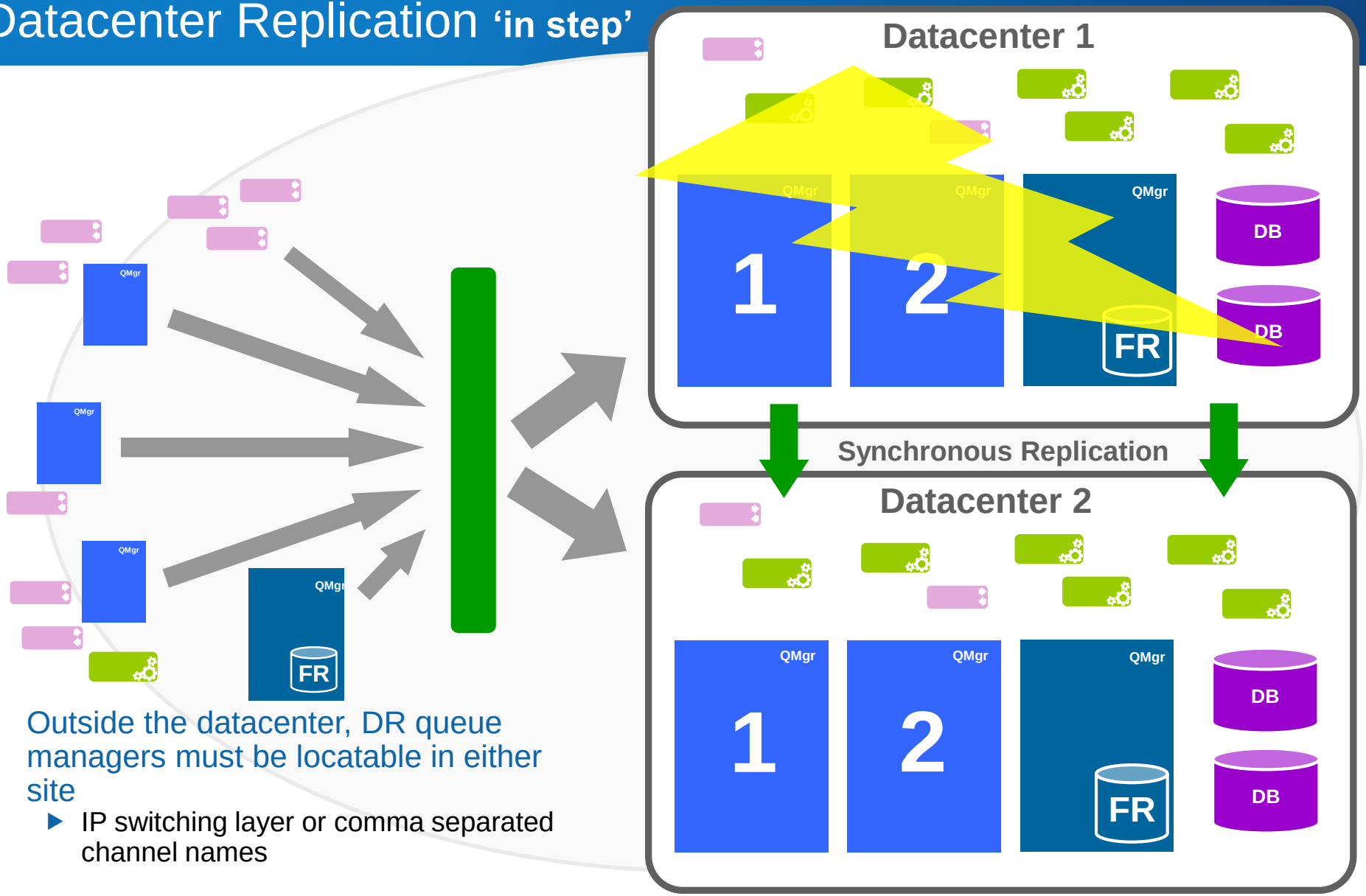
Disaster Recovery and clusters

MQ Clusters and Disaster Recovery

- Everybody seems to do disaster recovery slightly differently
- MQ Clusters can be made to fit with *most* setups
- But some more easily than others…

Synchronous replication

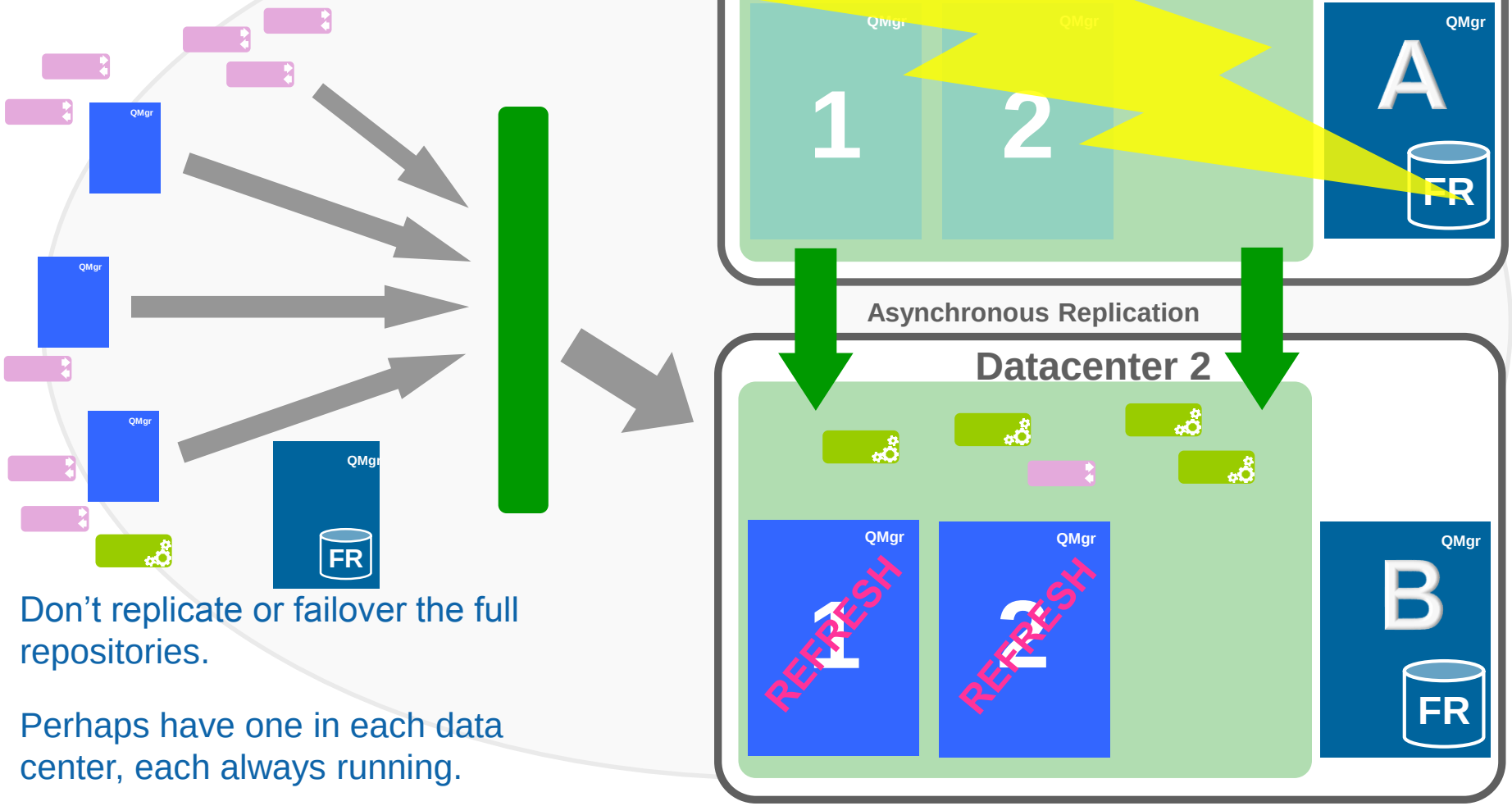
Datacenter Replication 'in step'



- Outside the datacenter, DR queue managers must be locatable in either site
  - IP switching layer or comma separated channel names
- After failover, everything continues as if nothing happened

Asynchronous replication

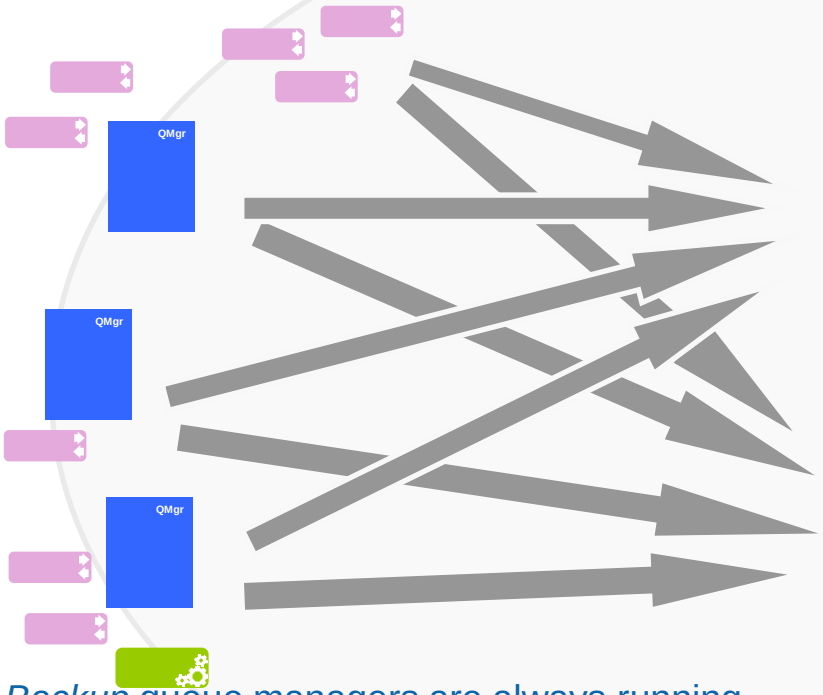
Datacenter Replication ‘async’

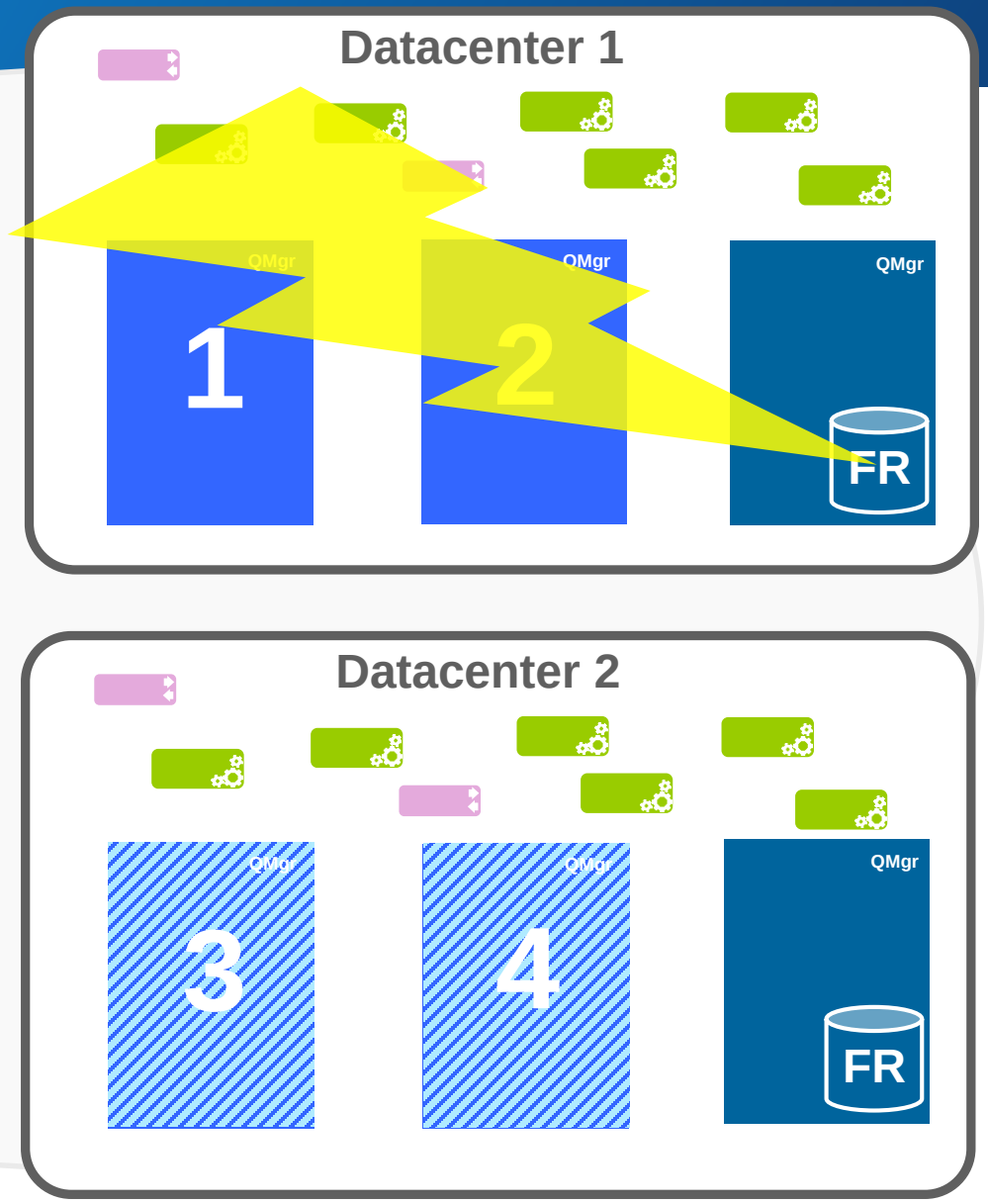


- Don’t replicate or failover the full repositories.
- Perhaps have one in each data center, each always running.
- No need to refresh them, and always at least one full repository is available

No replication

No Replication 'warm standby'





- *Backup* queue managers are always running.
  - Cluster workload balancing used to direct
- Failed connections are detected and messages routed to *secondary* qmgrs.
- Messages will be trapped on *failed* queue managers in the event of a failure.
- The applications and the system must be designed to accommodate this configuration.

Summary

- My first cluster
- Service availability
- Location dependency
- Avoiding interference
- When things go wrong

Questions & Answers

Legal Disclaimer