Recording available at:
http://www.slideshare.net/DavidWare1/ame-2272-mq-publish-subscribe-network-pdf

Publish/Subscribe in an MQ network

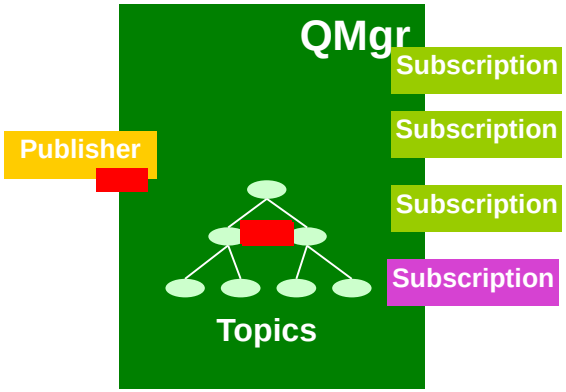
David Ware
Lead Architect, IBM MQ Distributed
dware@uk.ibm.com

Agenda

- The basics
- The possibilities
- Compare and contrast
- Scenarios

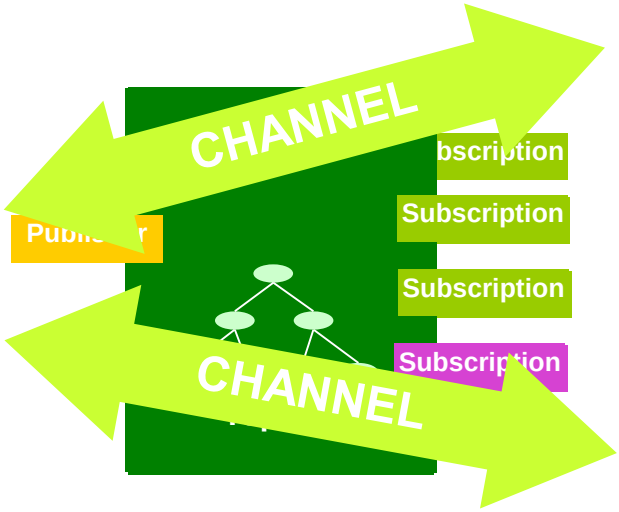
Distributed publish/subscribe

- Everything revolves around the topic tree, dynamically built up in a queue manager



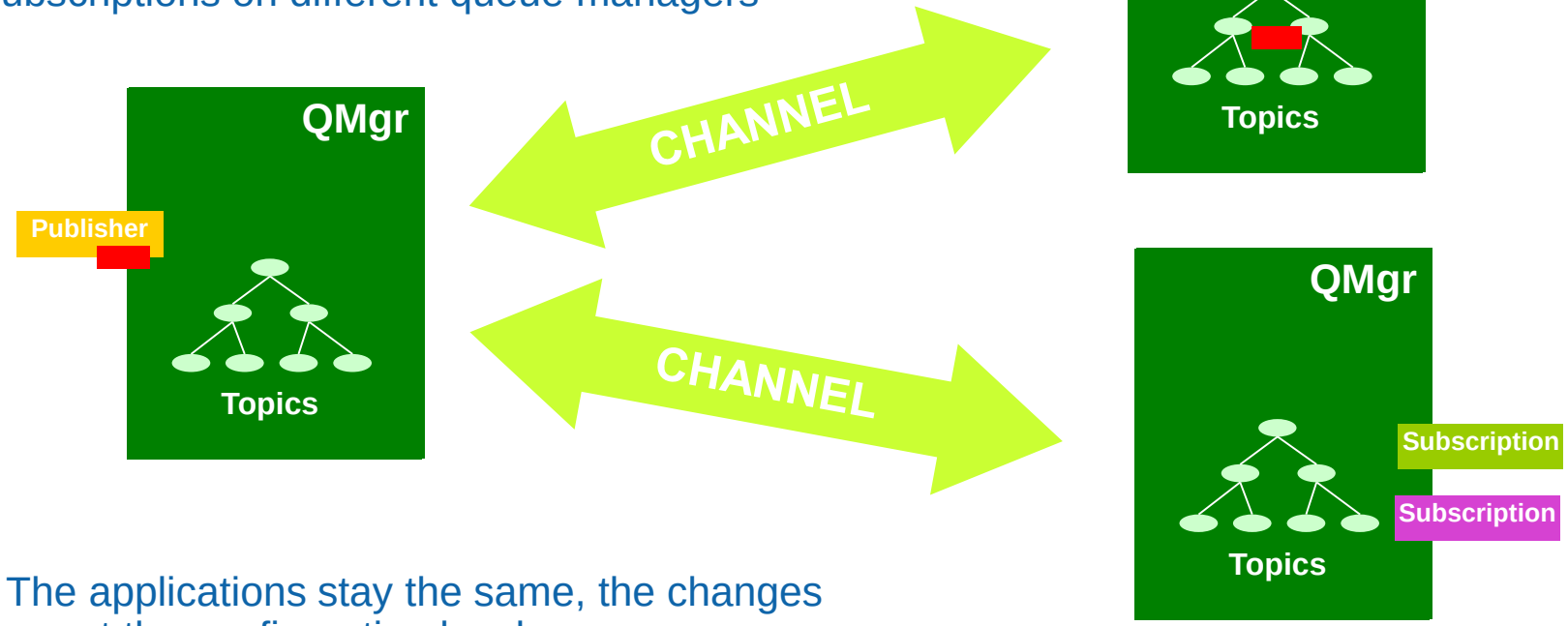
Distributed publish/subscribe

- Everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them



Distributed publish/subscribe

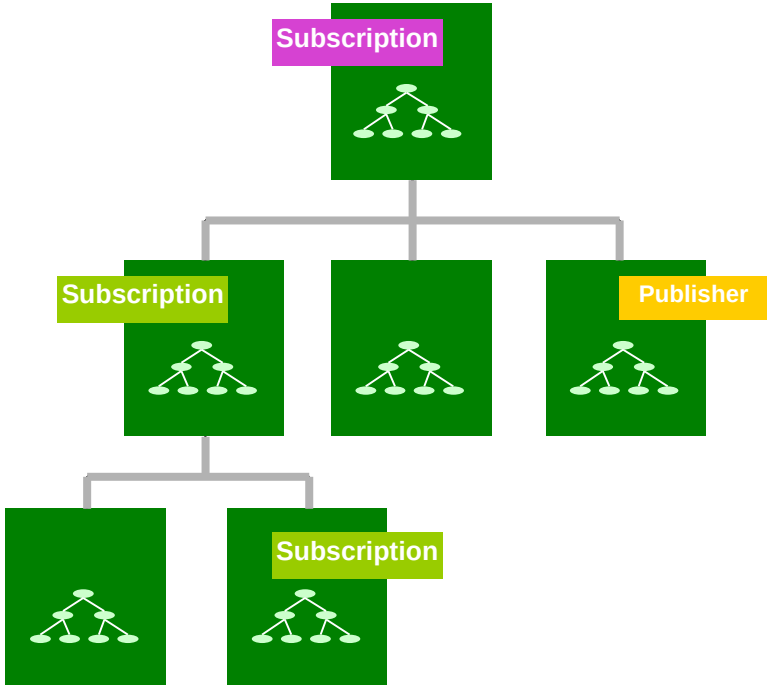
- Everything revolves around the topic tree, dynamically built up in a queue manager
- Queue managers can work together to share their topic tree knowledge between them
- Enabling publications to be propagated to subscriptions on different queue managers

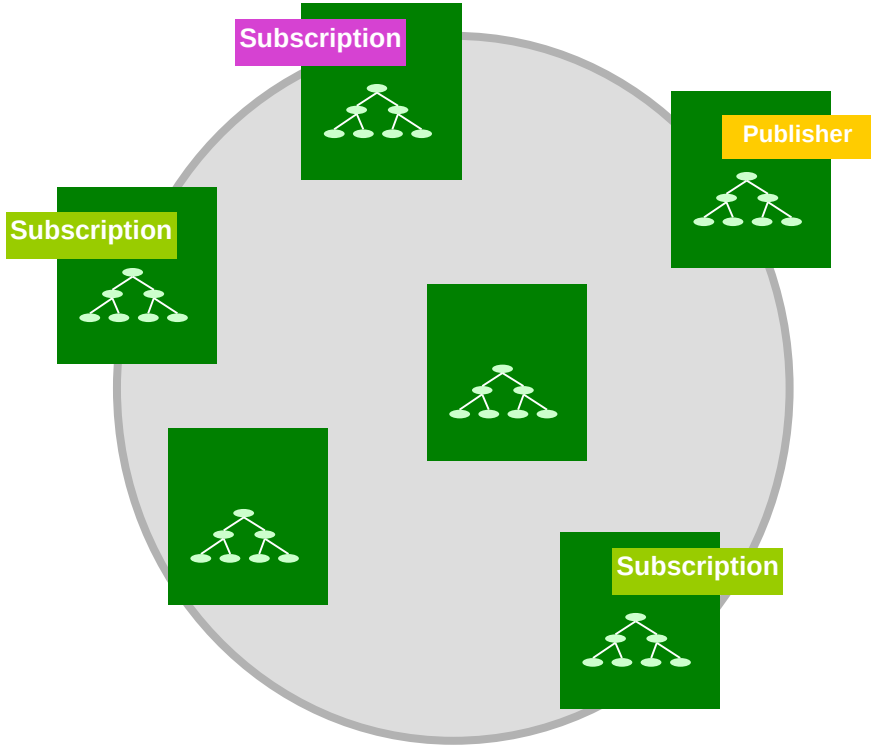


- The applications stay the same, the changes are at the configuration level.

Distributed publish/subscribe topologies

- Publish/subscribe topologies can either be created as a defined ***hierarchy***
- or more dynamically as a ***cluster***
- In both cases, publish/subscribe ***configuration is required***…

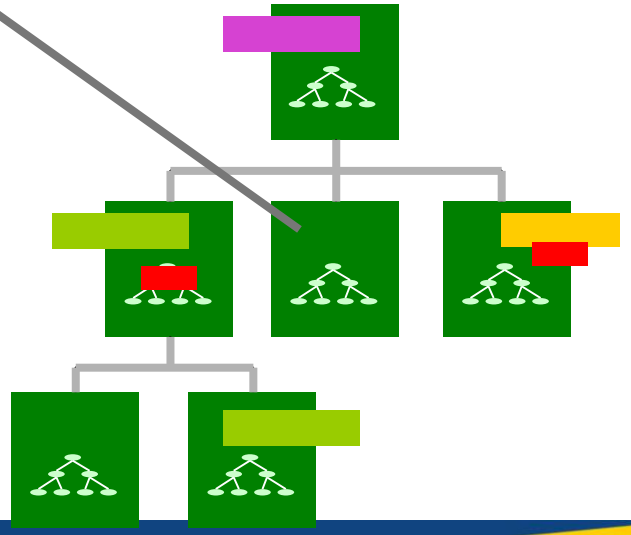




Configuring

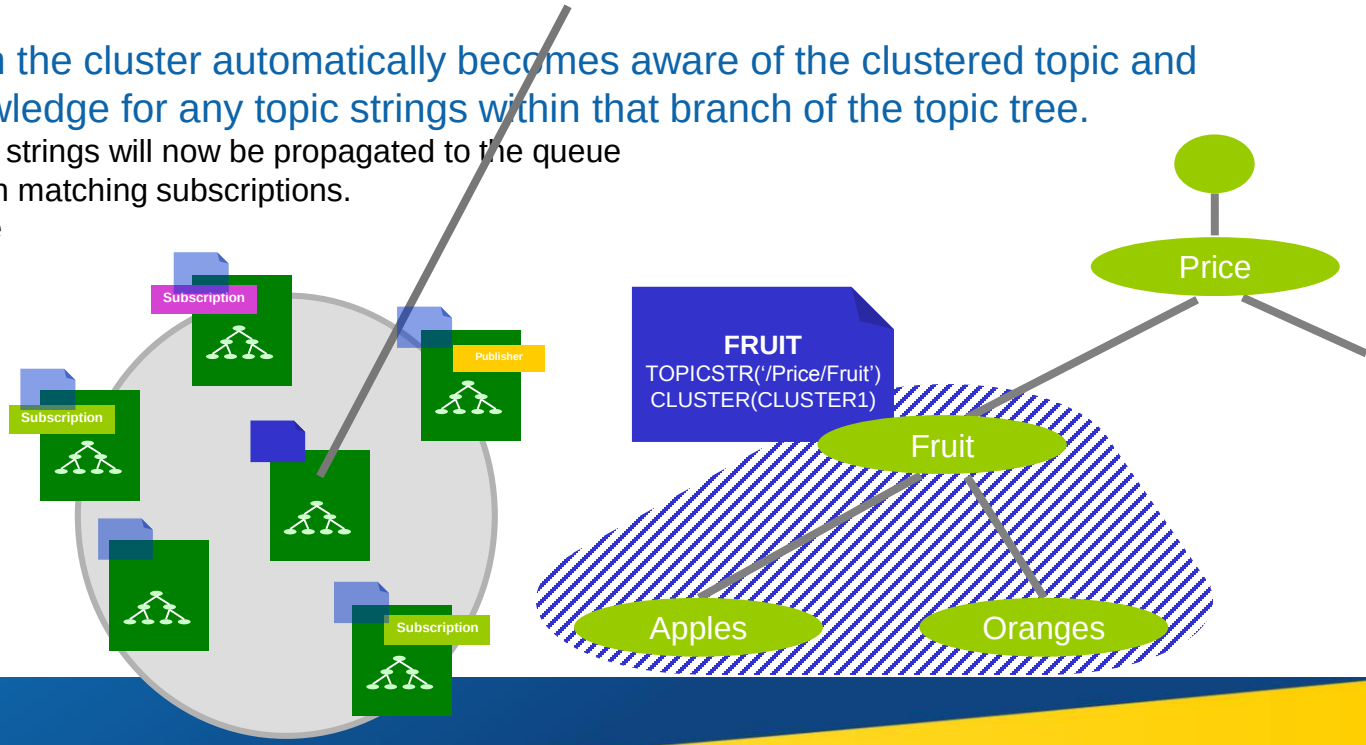
Hierarchies

- Hierarchies have been supported since the original WebSphere MQ publish/subscribe broker support.
- Every queue manager defines its *parent* in the hierarchy.
  - (except for the root queue manager)
  - ***ALTER QMGR PARENT(QMGR1)***
- Each pair of directly related queue managers must be able to connect to each other using WebSphere MQ channels.
  - Manual channels, with queue manager named transmission queues or queue manager aliases.
  - They are both in the same cluster.
- Use **DISPLAY PUBSUB** to check the relationships
- By default all subscription knowledge is shared across the hierarchy.
  - No requirement to define administered topic objects or modify applications.
  - Controlled by scoping down topics.
    - ***Subscription Scope*** and ***Publication Scope***.



Publish/Subscribe Clusters

- No additional connectivity configuration is required (that's handled by the cluster)
- ***By default, no subscription knowledge is shared between members of a cluster.***
  - Meaning no publications automatically flow between queue managers.
- Enabling distributed pub/sub in a cluster is initially achieved by ***clustering*** an administered topic definition on ***one*** of the queue managers in the cluster.
  - Set the cluster name to the name of the cluster.
  - ***DEFINE TOPIC(FRUIT) TOPICSTR('/Price/Fruit') CLUSTER(CLUSTER1)***
- ***Every*** queue manager in the cluster automatically becomes aware of the clustered topic and shares subscription knowledge for any topic strings within that branch of the topic tree.
  - Publications for those topic strings will now be propagated to the queue managers in the cluster with matching subscriptions.
  - No application changes are required.

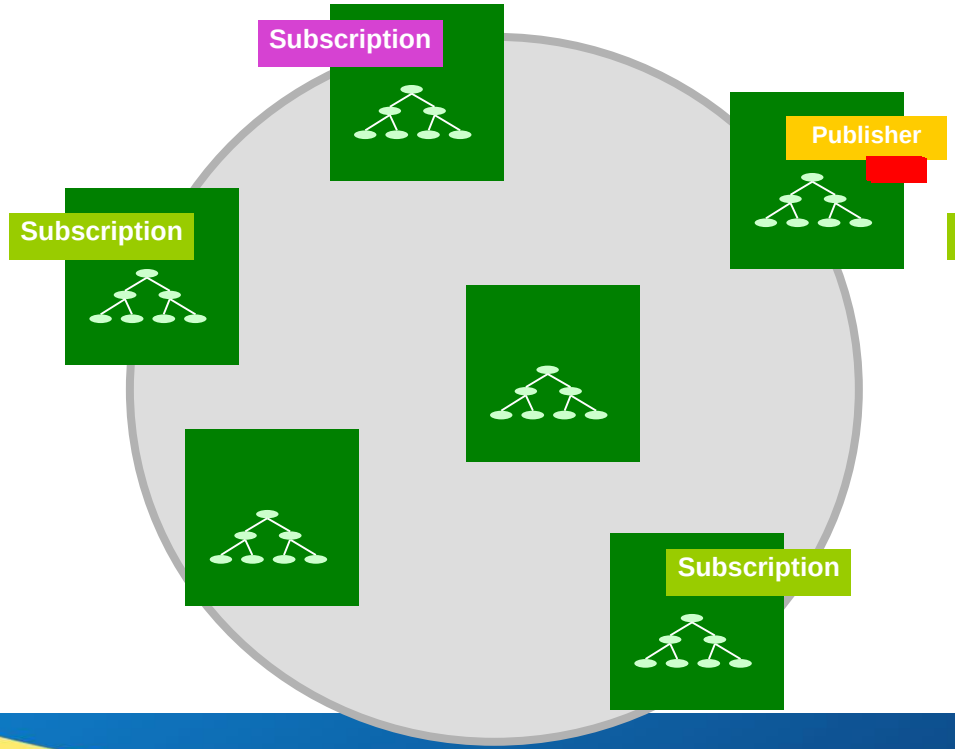


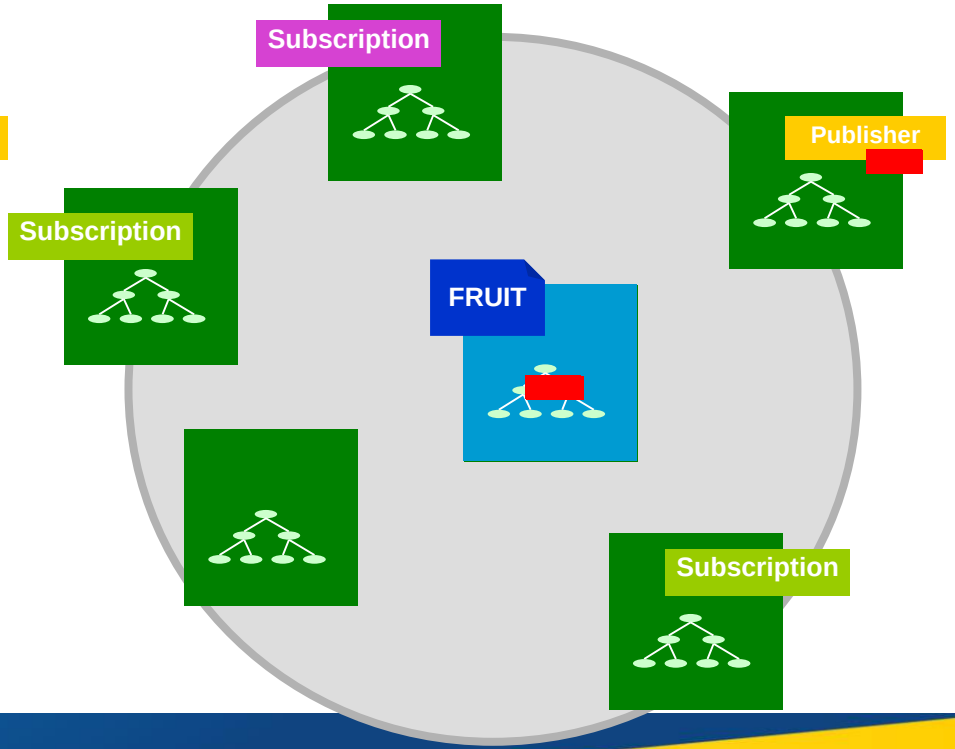
Clusters

- Publish/subscribe in clusters now has two modes of routing publications.
  - **DIRECT**: Publications are sent directly from the publisher's queue manager to every queue manager in the cluster with a matching subscription.
  - **TOPIC HOST**: Publications are routed via one or more queue managers in the cluster. These are the queue managers where the clustered topic definitions were defined.

V8

- ***DEFINE TOPIC … CLUSTER(…) CLROUTE(DIRECT|TOPICHOST)***





Which is best?

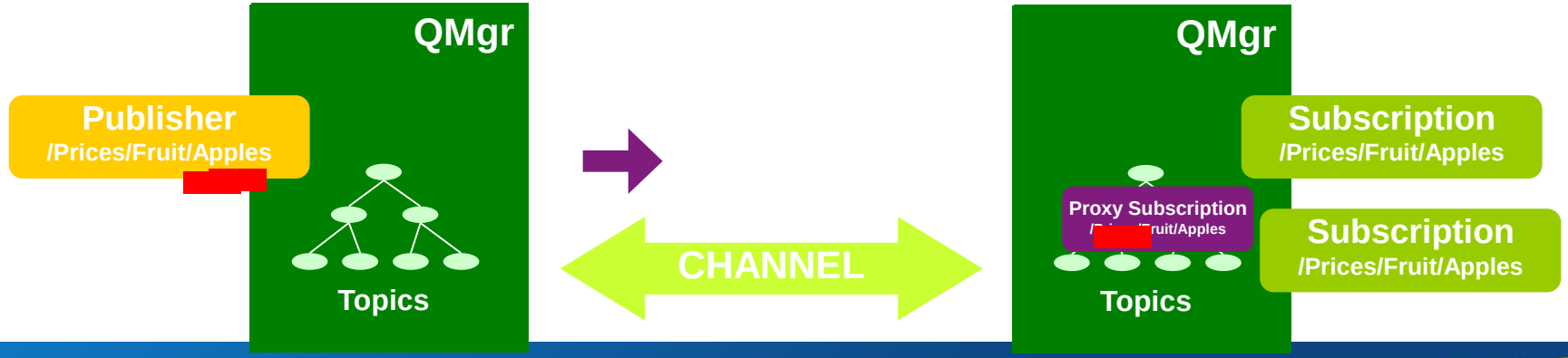
- Should I use a ***hierarchy***, a ***direct*** or a ***topic host cluster topic***?
- *Guess what, it depends…*

- Many factors will influence the design of a pub/sub topology:
  - Number of queue managers in the topology
  - Size and shape of those queue managers
    - Their capacity, availability, lifetime, interconnectivity…
  - Number and lifetime of subscriptions
  - Number and lifetime of individual topic strings
  - Publication traffic
  - Distribution pattern of publications to subscribers
  - …
- One size does ***not*** fit all.
- First, you need to understand more about how it works…

The mechanics

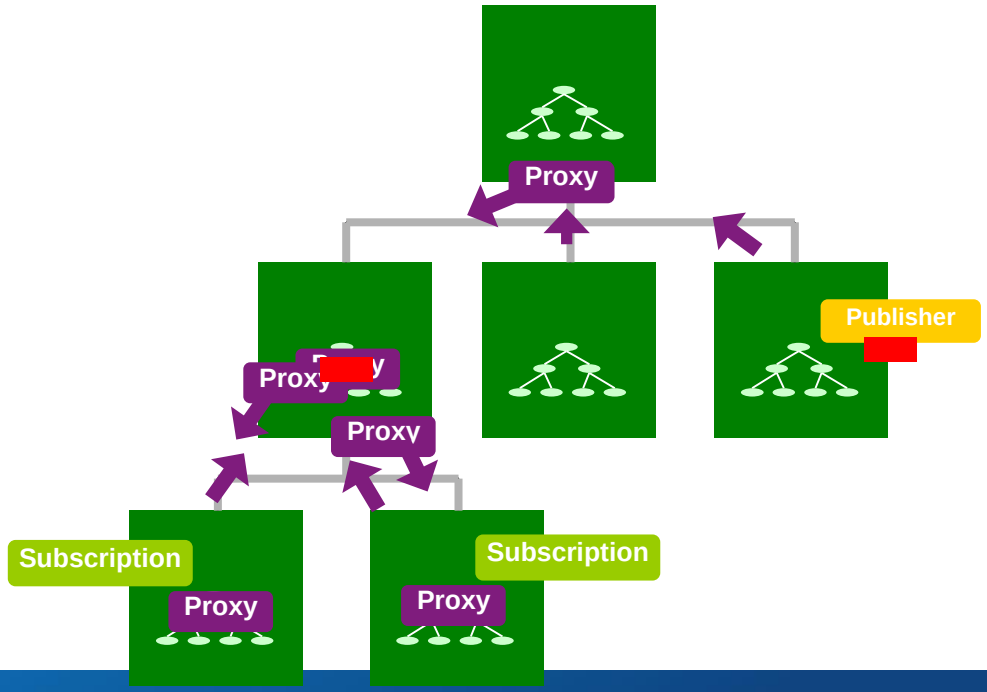
How it works: subscription propagation

- A subscription for a topic string is created on a queue manager.
- That queue manager tells others that it is interested in that topic string.
- Those queue managers register ***proxy*** subscriptions to represent the interest.
- On a queue manager, when a message is published to the topic string, a copy of the message is sent, over MQ channels, to each queue manager that a proxy subscription is held for.
- The receiving queue manager processes the message and gives a copy to each subscription it holds *(including any proxies…).*
- Additional subscriptions on the same topic string do not require additional proxy subscriptions to be flowed.



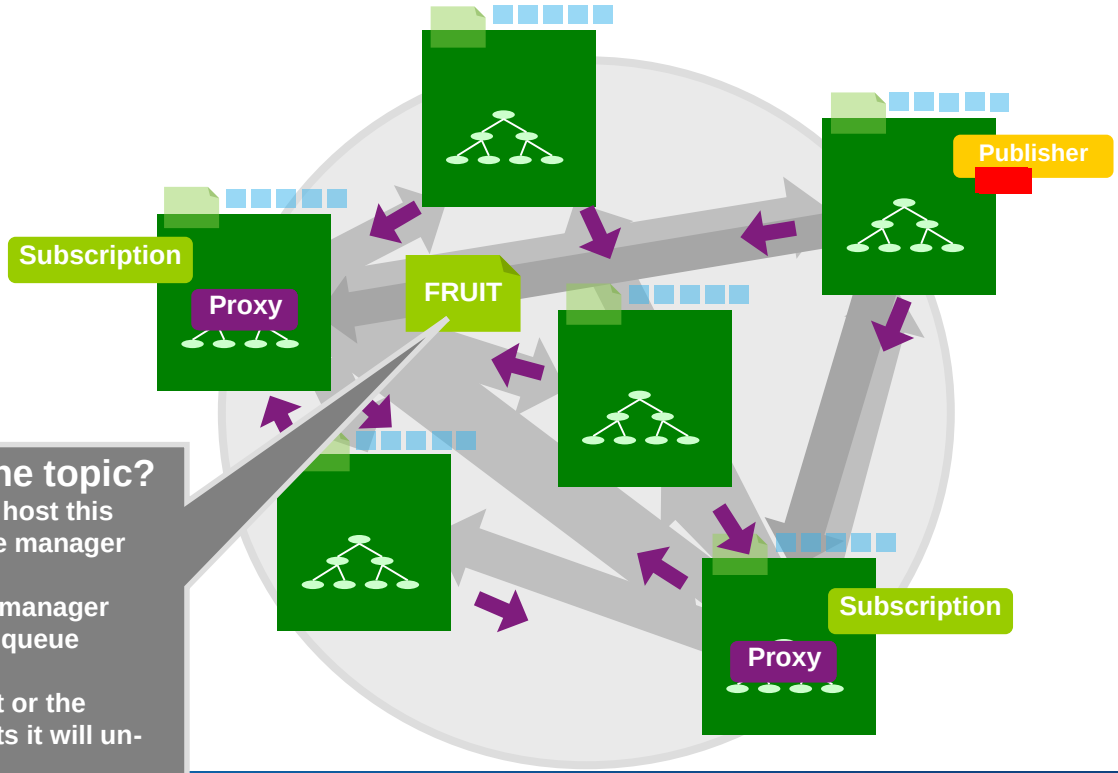
How it works: in a *hierarchy*

- Proxy subscriptions are sent from a queue manager to every ***directly*** connected queue manager in the hierarchy.
- They in turn send proxy subscriptions onto any further relations.
- Until everyone in the hierarchy is aware of the topic string being subscribed to.
- Publications are then sent back down the path of proxy subscriptions.



How it works: *Direct* routed clustered topics

- Proxy subscriptions are sent from a subscribing queue manager directly to ***every*** other queue manager in the cluster.
- First, define the clustered topic object.
  - *Every queue manager becomes aware of the topic and everyone else in the cluster.*
- Any publication from a queue manager is sent directly to those queue managers which sent proxy subscriptions.

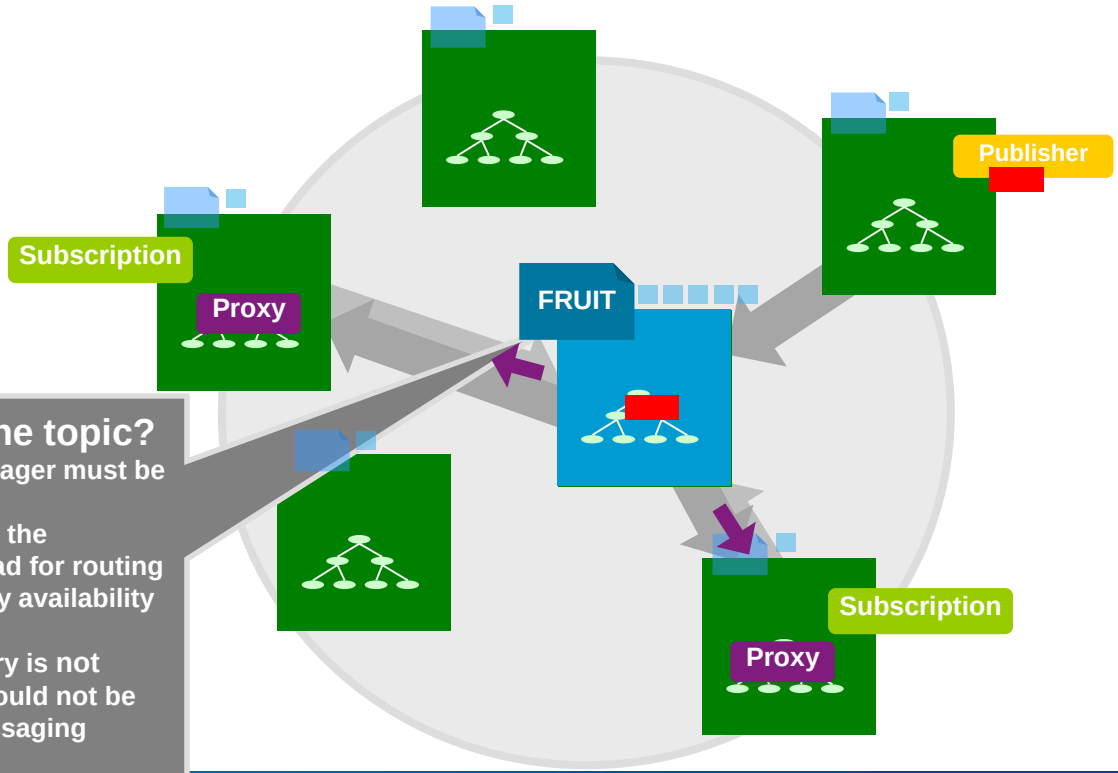


Where to define the topic?
Any queue manager can host this topic object, every queue manager will act the same.
Pick a *long living* queue manager such as a full repository queue manager.
Deleting this topic object or the queue manager that hosts it will un-cluster the topic.

How it works: *Topic host* routed clustered topics

V8

- Proxy subscriptions are sent from a subscribing queue manager ***only*** to queue managers that host a definition of the clustered topic.
- Define the clustered topic object on a suitable queue manager.
  - Only the topic hosts become aware of everyone else.
- Any publication from a queue manager is ***always*** sent to one of the topic hosts, who then forwards the message to any subscribing queue managers.



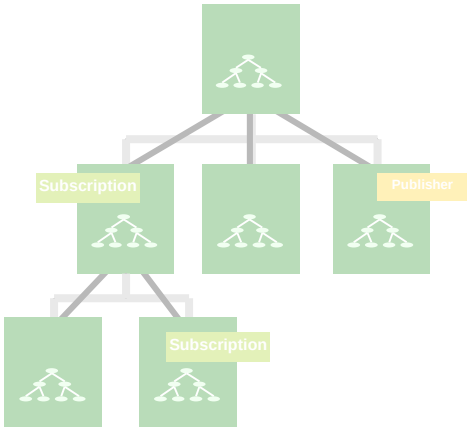
Where to define the topic?

This time the queue manager must be carefully considered.
It must be able to handle the additional messaging load for routing and be able to satisfy any availability requirements.
This time, a full repository is not recommended – they should not be used for application messaging traffic.

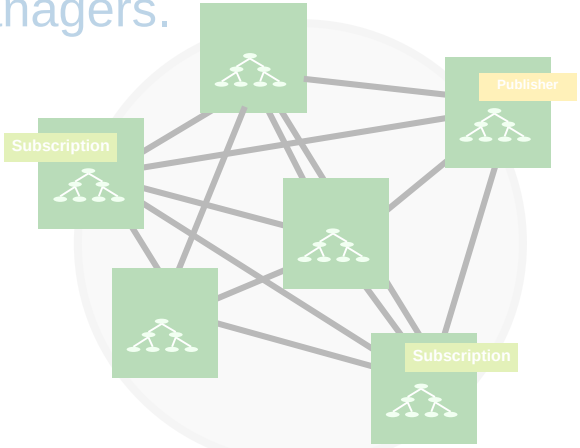
Side-by-side

Side-by-side: scaling

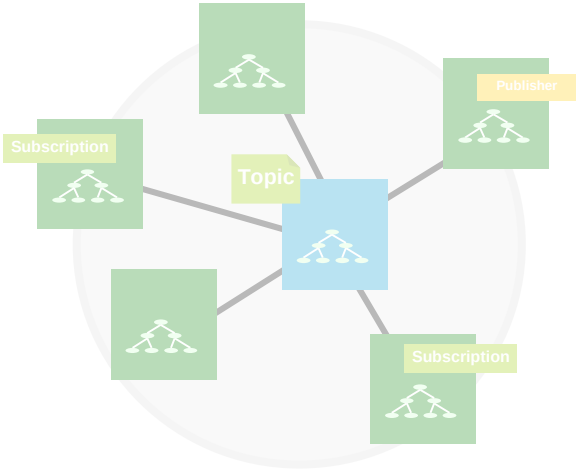
- Channel connectivity is restricted in both topic host routed clusters and hierarchies.
- Direct routed clusters will result in many queue manager channels being established, even to queue managers with no publishers or subscribers.
  - ***This can be a concern for large clusters***.
- Hierarchies and direct routed clusters share subscription knowledge across **all** queue managers.



Hierarchy



Direct Cluster



Topic host cluster

Side-by-side: publication flows

- Direct routed clusters ensure publications take the shortest path between publishers and subscribers.
- Topic host routed clusters can introduce an additional hop, with hierarchies relying on even more.
  - In both of these, careful placing of subscribers and publishers can achieve the same single-hop route that direct clusters provides.

Hierarchy:1,2,3… hops

Direct Cluster: 1 hop

Topic host cluster: 1 or 2 hops

Side-by-side: configuration

- Hierarchies require fine grain planning and configuration of queue managers.
  - Harder to alter the hierarchical structure.
- Clusters allow queue managers to join and leave without too much consideration once the overall plan is defined.
  - Careful planning of topic host routers is required.

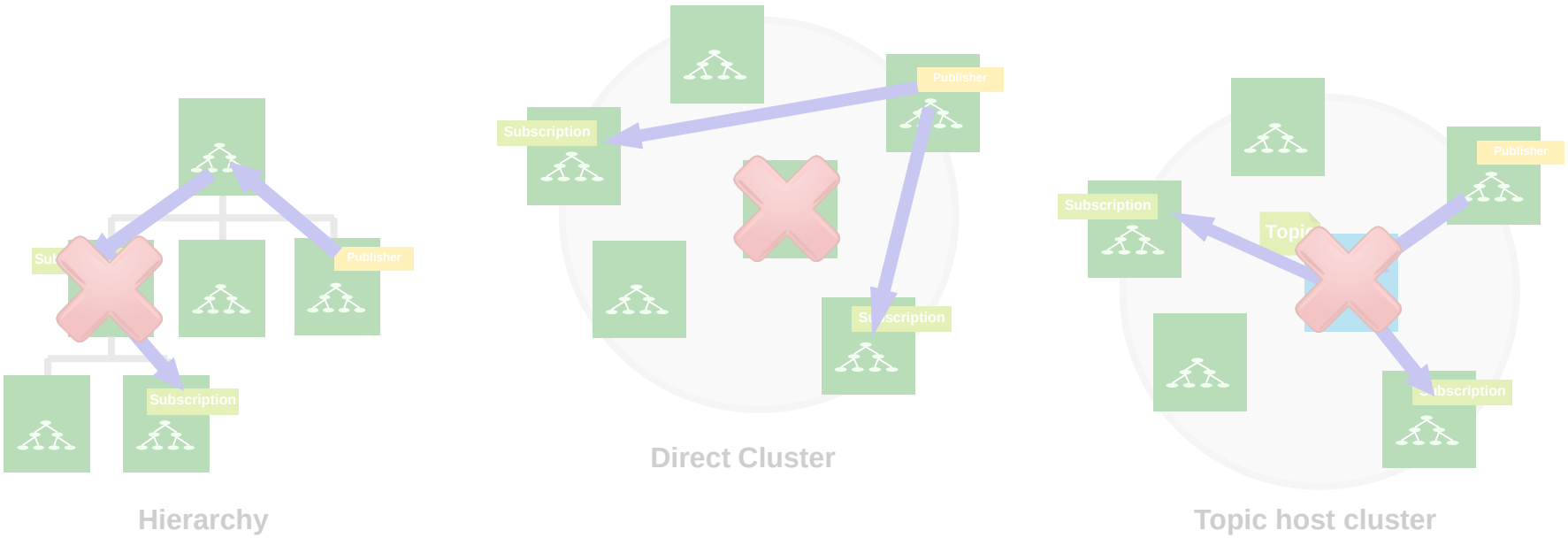
Hierarchy

Direct Cluster

Topic host cluster

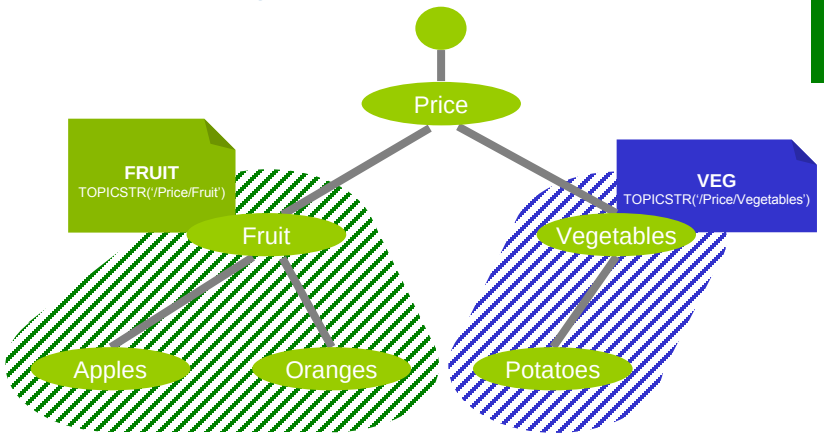
Side-by-side: availability

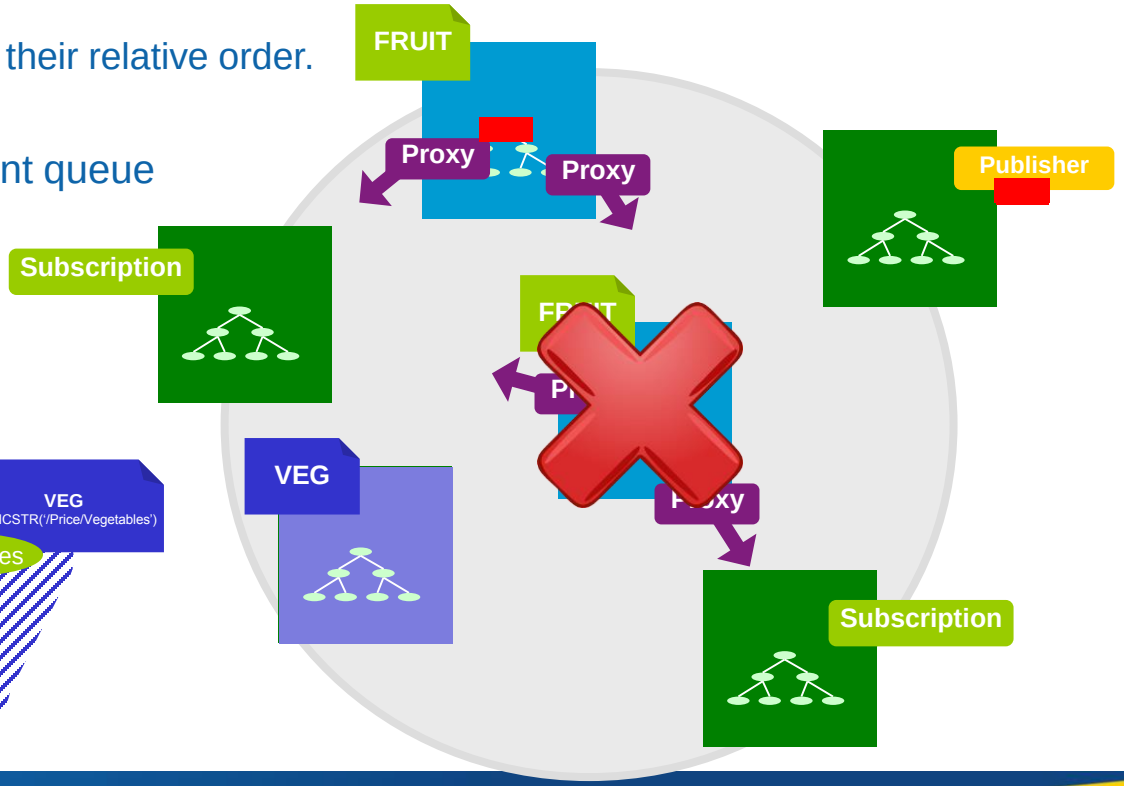
- An unavailable queue manager in a hierarchy can cut off many others.
- In a direct routed cluster it'll only affect the subscribers and publishers on that queue manager.
- In a topic host routed cluster it can affect more if the queue manager hosts a topic definition.
  - ***But this can be avoided…***



Topic host routing

- Multiple queue managers can host the *same* clustered topic.
  - With the same name, topic string, etc.
- Publications are routed based on availability and workload balancing.
- This increases availability and scalability of routes.
  - Multiple topic hosts for a topic enable workload balancing of publication traffic.
- Publications will no longer maintain their relative order.
- Hosting different topics on different queue managers can also be used to increase availability and scaling.





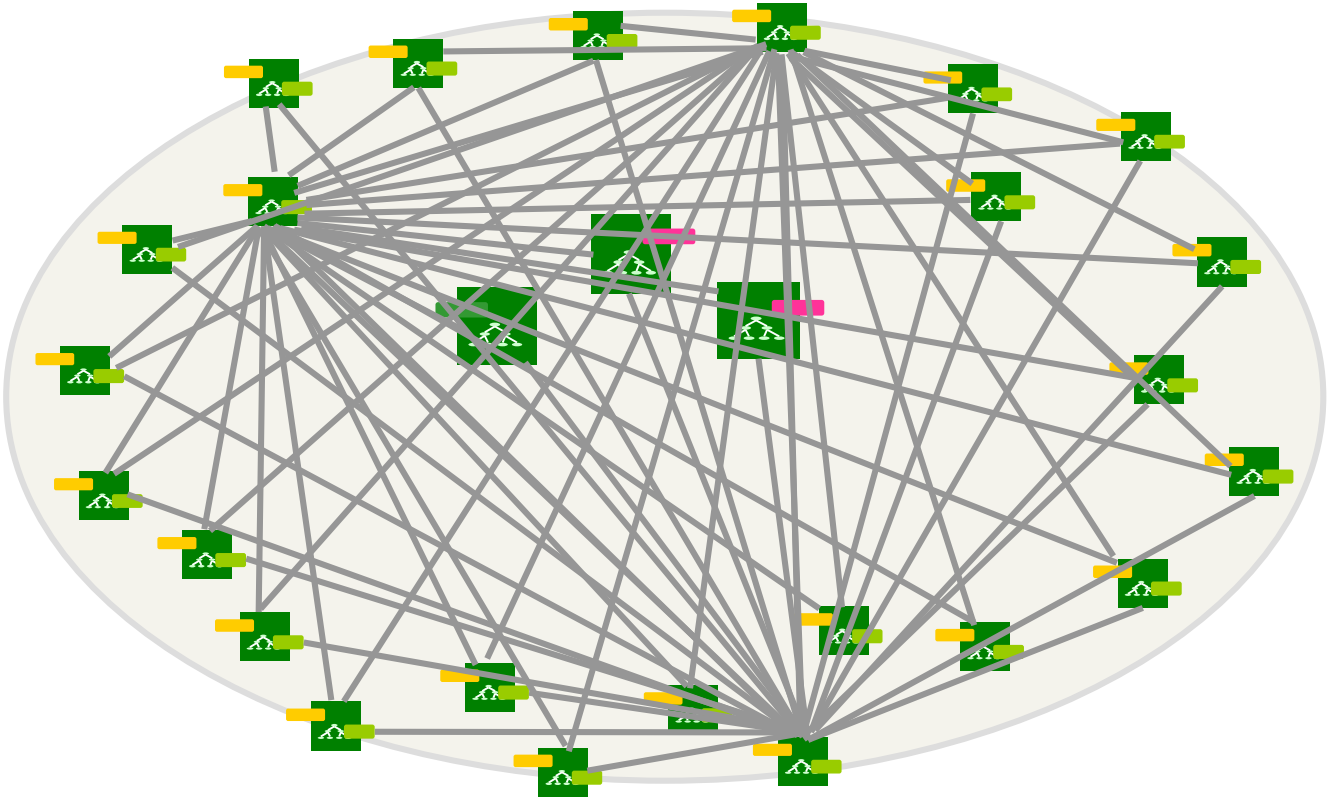
Comparison summary

- **Scaling**
  - Topic host routed clusters provide the best options for scaling.
- **Availability**
  - Clusters can be configured to avoid single points of failures, unlike hierarchies.
- **Publication performance**
  - All topologies use very similar inter queue manager techniques for transferring messages, and each can be designed to minimise routes between queue managers.
- **Configuration**
  - Clustering is the most dynamic and simplest solution – especially where clusters are already in use.

Putting it into practice

A big ‘hub and spoke’ cluster

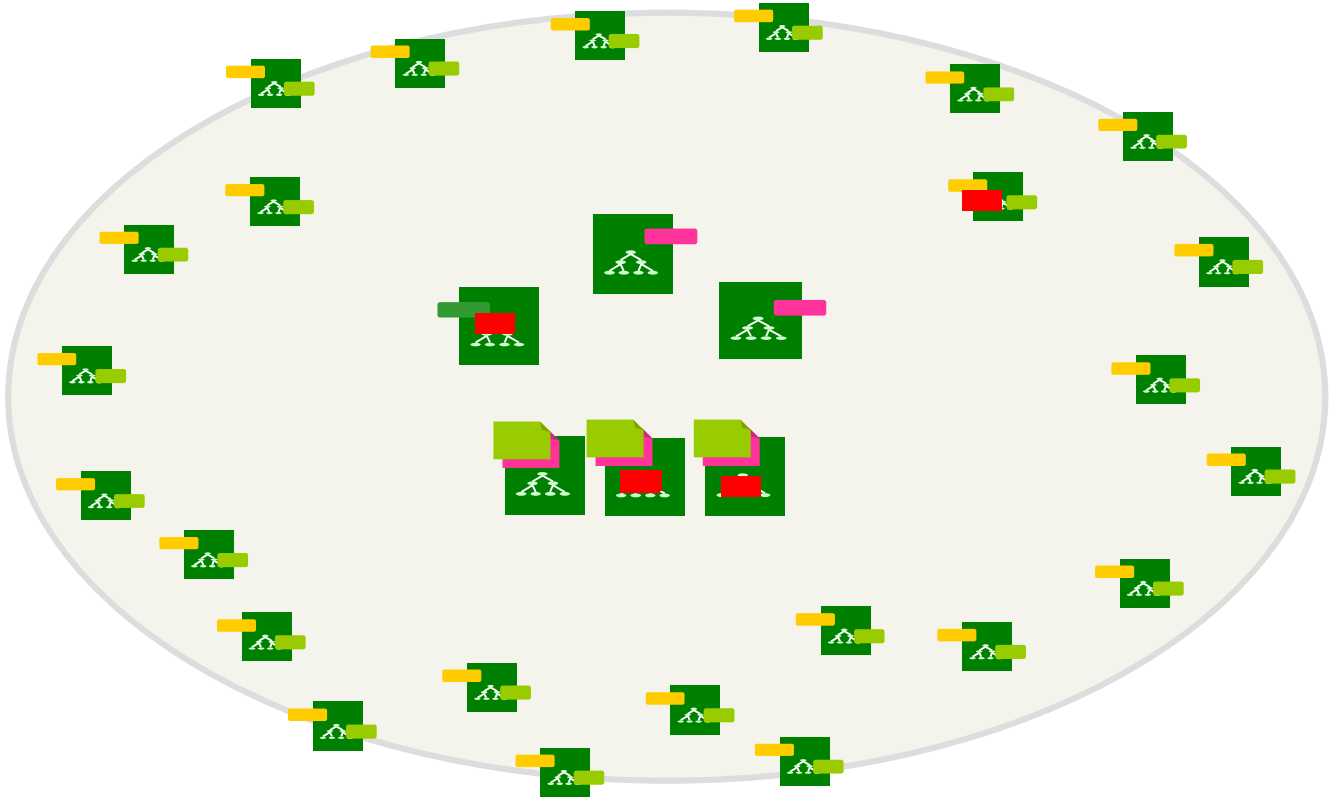
- A thousand queue managers, all talking to a few in HQ, but rarely to each other.
- Each with its own subscriptions and publishers.



- ***Do not use a direct cluster topic!***

A big ‘hub and spoke’ cluster

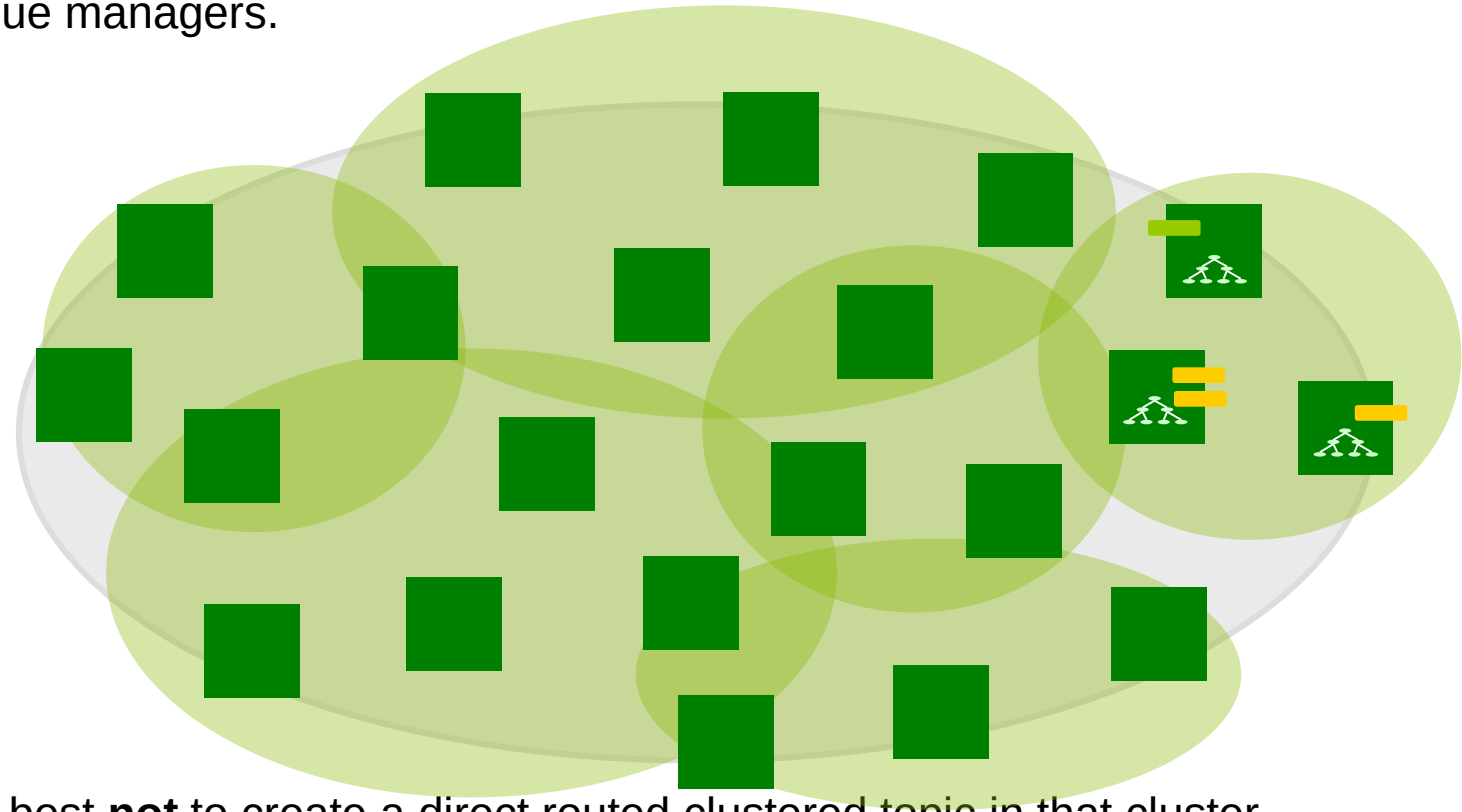
- A thousand queue managers, all talking to a few in HQ, but rarely to each other.
- Each with its own subscriptions and publishers.



- Instead, use ***topic host*** routed clustered topics, possibly on a dedicated set of routing queue managers, sized to take the load.

A small amount of pub/sub within a much larger cluster

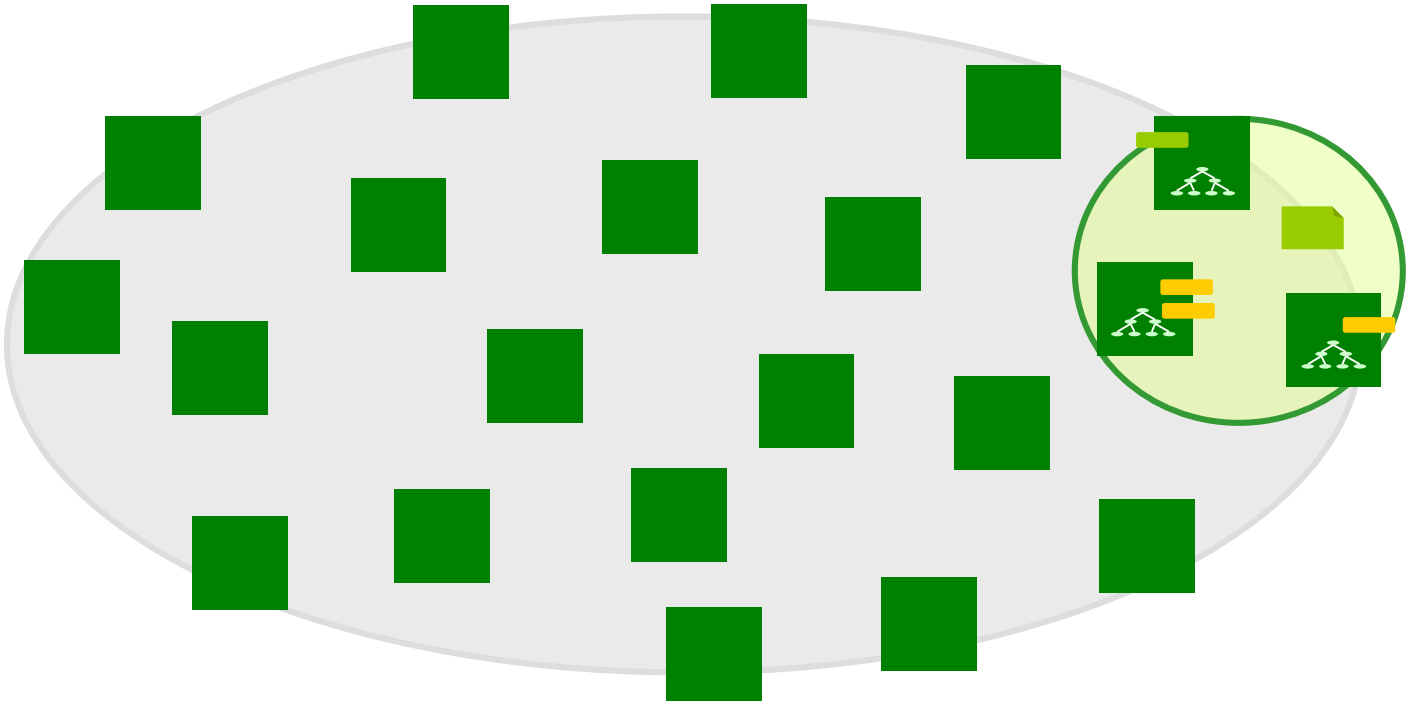
- An existing single cluster containing hundreds or thousands of queue managers. Each connected to a few others in the cluster, as required.
- A ***few*** applications need to support inter queue manager pub/sub on a handful of queue managers.



- It is best **not** to create a direct routed clustered topic in that cluster.
  - All queue managers can potentially create channels to connect to each other.

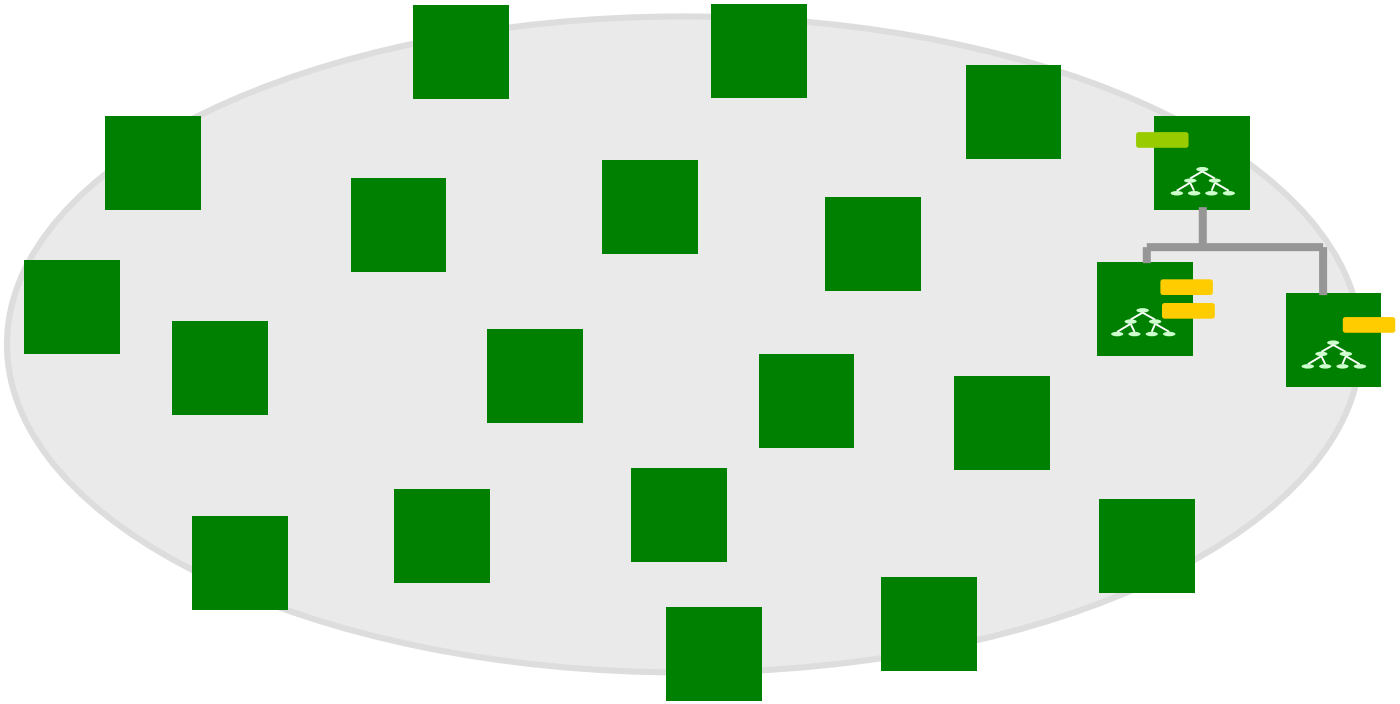
A small amount of pub/sub within a much larger cluster

- One solution is to create a dedicated overlaid cluster for the queue managers requiring publish/subscribe and clustering a **direct** topic.
- If you're nervous, you can use the queue manager property **PSCLUS** to block the other queue managers from accidently clustering a topic



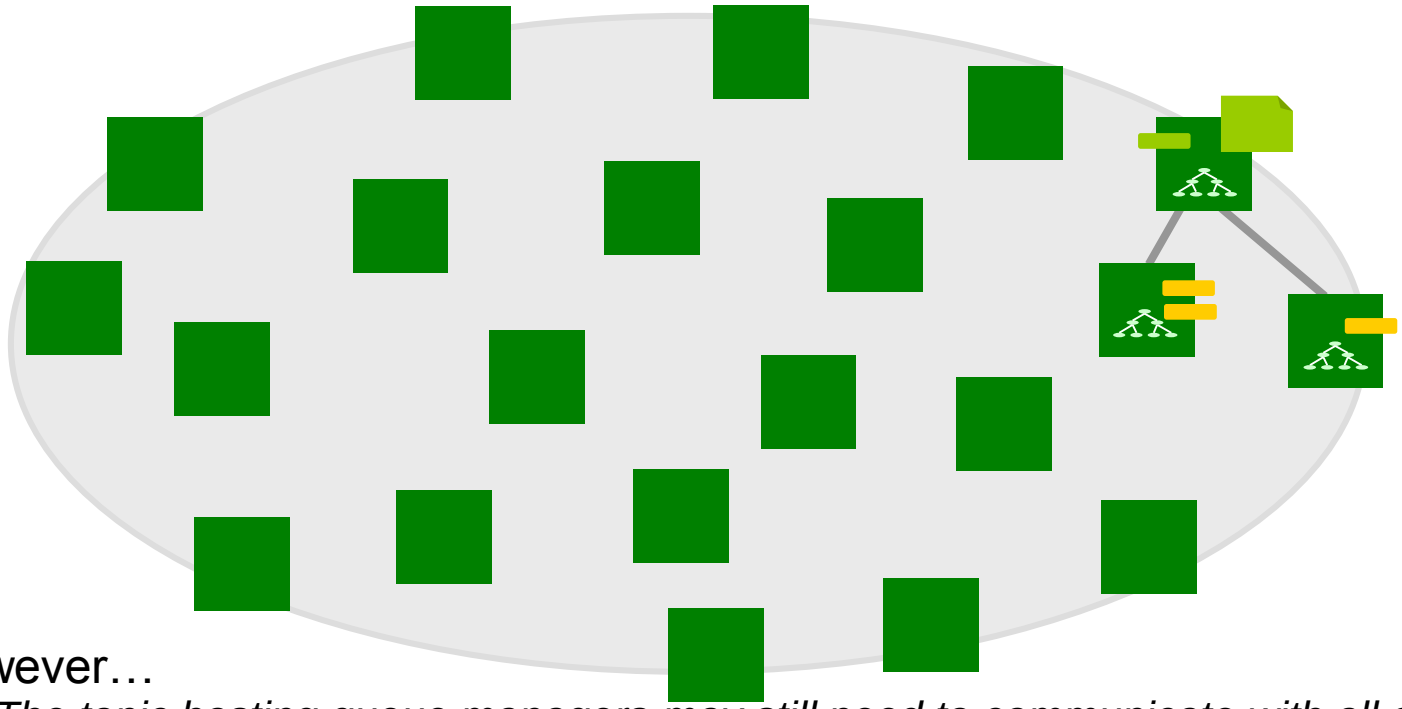
A small amount of pub/sub within a much larger cluster

- Alternatively, create a hierarchy and **don't** cluster the topic



A small amount of pub/sub within a much larger cluster

- Or use a topic host routed clustered topic.
  - All queue managers become aware of the topic but only those using it will communicate with the topic host queue manager(s).

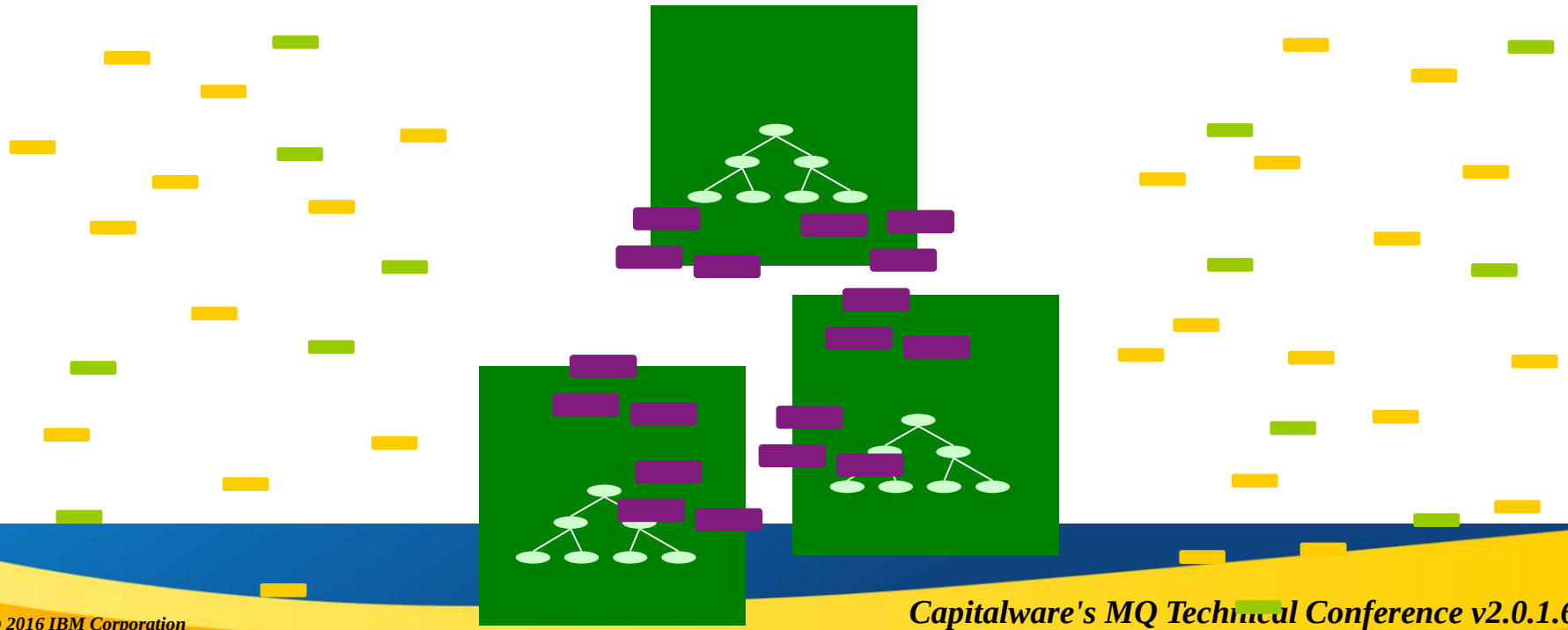


- However…
  - *The topic hosting queue managers may still need to communicate with all other queue managers under certain error circumstances.*
  - *And you can't use PSCLUS to protect the whole cluster*

A very dynamic setup

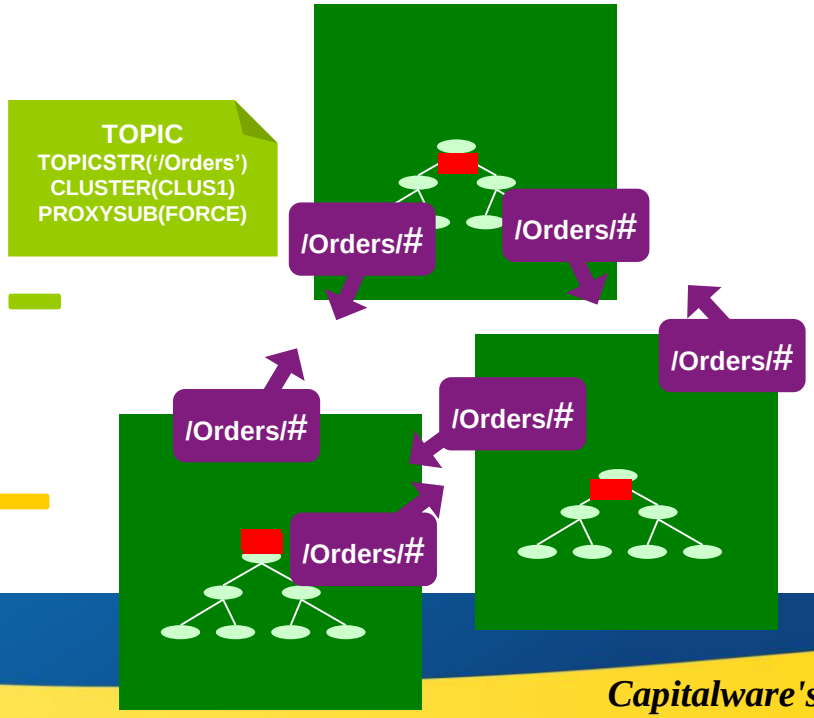
A very dynamic setup

- Thousands of loosely connected dynamic subscriptions, on thousands of different topic strings, continually coming and going, along with corresponding publishers.
- A few queue managers to provide availability and subscription load sharing.
- This introduces a special problem….
  - *Proxy subscription '**churn**'*



Publication broadcast

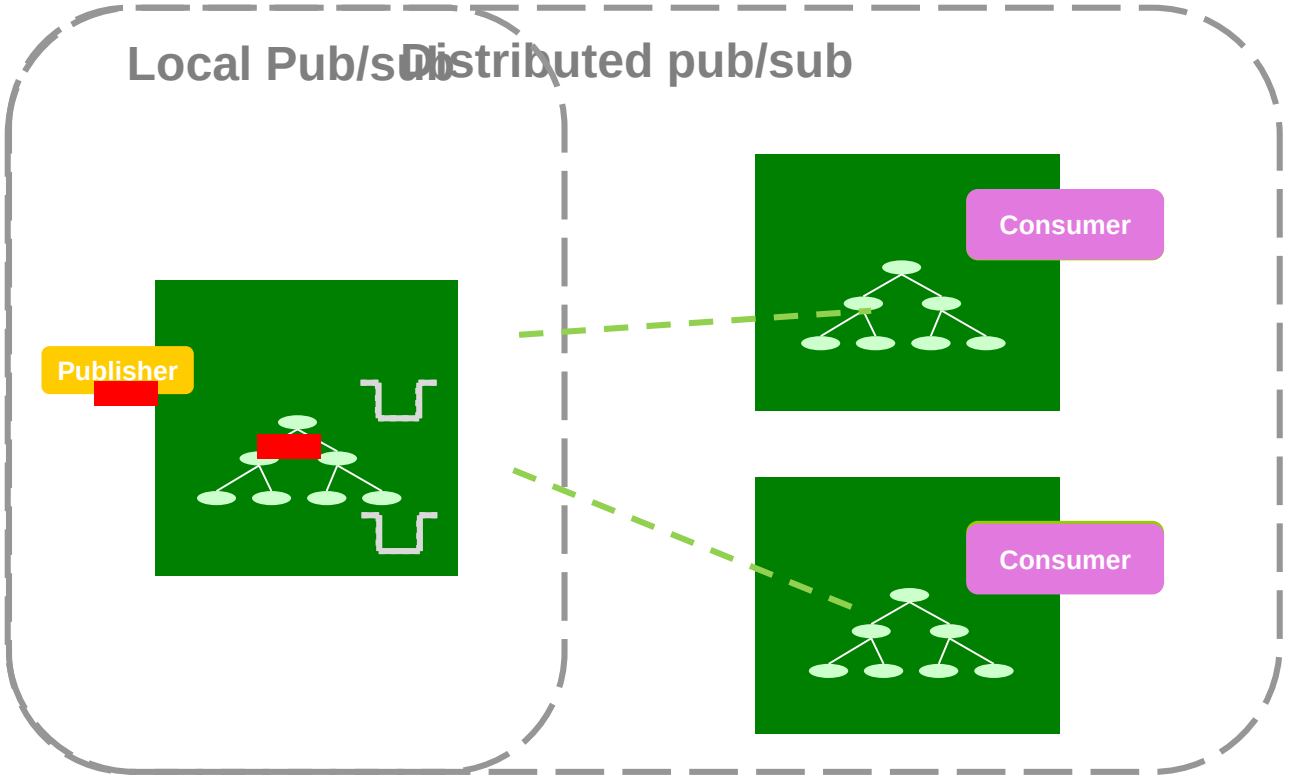
- Setting ***proxy subscription*** to ***force*** generates a wildcarded proxy subscription on all queue managers where that topic is seen.
  - **All** queue managers when in a cluster.
  - Only the queue manager where it is defined when in a hierarchy.
- Every publication is always sent to every queue manager and onto the subscriptions.
  - If no local subscriptions exist for a publication, it is simply *discarded*.
- Once that is in place there is no longer a need for individual proxy subscriptions for specific topic strings.
  - *This is automatic in IBM MQ V8, but requires intervention prior to this* V8



Pub/sub in a network, but not in that way

Distributed pub/sub, without hierarchies or clustered topics

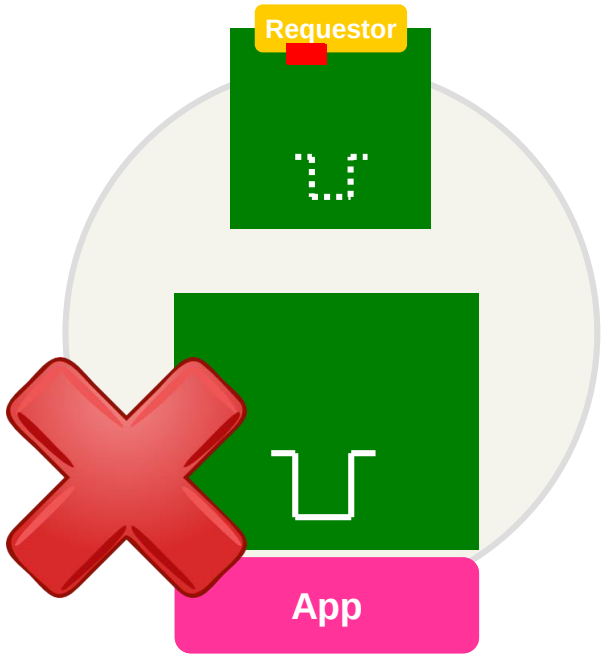
- Simple setups to use pub/sub to distribute messages across queue managers
- Subscriptions need to be defined where the publishers are connected
- Consuming applications interact with the target queue, not the subscription



Subscription availability

The continuously available point-to-point application

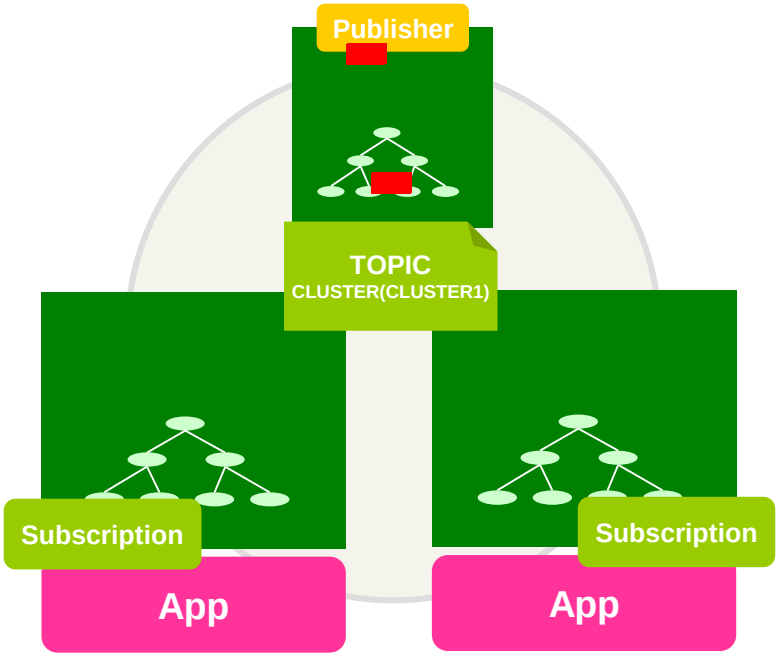
- When we need a point-to-point application to be continuously available we can cluster and duplicate its request queue.



- How do we get the same with a publish/subscribe application?....

The continuously available subscription

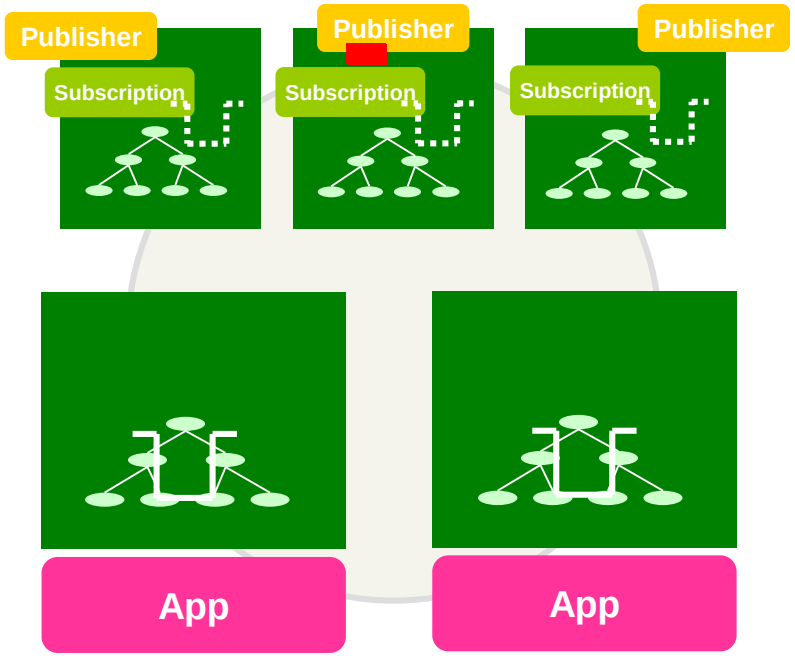
- The aim is to configure the system so that an application that processes publications can run multiple instances, across multiple queue managers.
- The instinct is to cluster the topic and duplicate the subscription.



- This is *not* the solution, it will lead to multiple copies of each publication

The continuously available subscription

- **Don't** define the topic as clustered *(this is **really** important).*
- Configure matching clustered queues, as for point-to-point and open these queues instead of subscribing.
- Define the same subscription on ***every*** queue manager where a publisher can connect.
  - Point those subscriptions at the **cluster queue**, leaving the target queue manager **blank.**



- Publish to the topic, and consume from the queues…

Hints and tips

Problem determination

- Double check your configuration.
- When in a cluster, check each queue manager's cluster knowledge.
  - In **V8** check the ***cluster state*** of the clustered topics
- When in a hierarchy, check your relationships.
- Check the error logs.
- Check your channels.
- Check your proxy subscriptions.
- Check your topic status.
- Watch for publication movement.
- Look for a build up of messages.

Summary

- The basics
- The possibilities
- Compare and contrast
- Scenarios

Questions & Answers

Legal Disclaimer