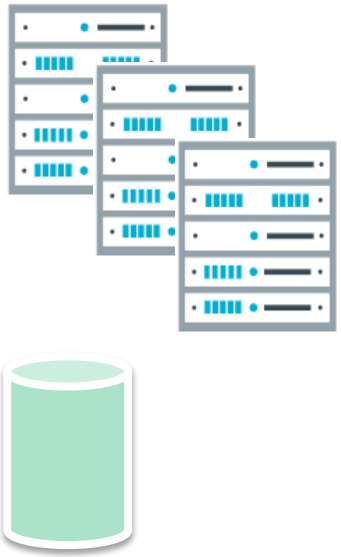
MQ Hybrid Cloud Architectures

Matthew Whitehead
IBM MQ Development
mwhitehead@uk.ibm.com

Agenda

- Topologies
- Connectivity
- Clients & Applications
- Connectivity

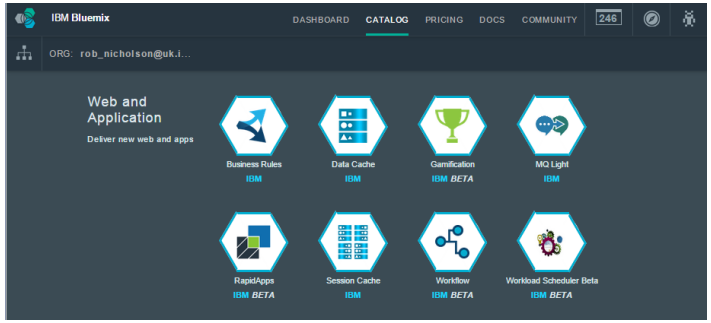
Bluemix Hybrid Messaging – Joining the 2 worlds together



- Systems of record
- Enterprise data
- 24 x 7 x 365 applications

On Premise

- Systems of engagement
- Mobile
- Social
- Analytics & Watson
- Rapid development



Bluemix Cloud

Why Hybrid Messaging?

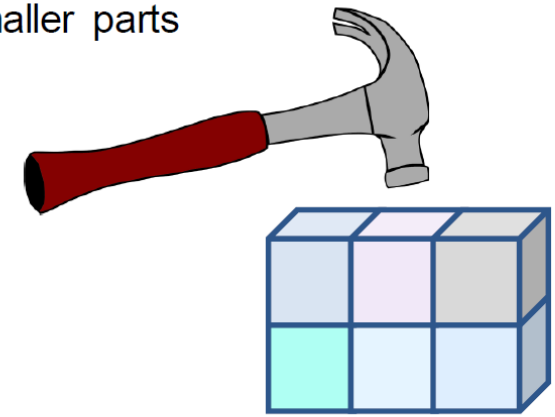
"All the benefits of cloud, with access to your enterprise data"

- Doing more with less
- Being more ready to change
- Making the development process less heavyweight
- Paying for what you use
- Integrating with other cloud services
- Rapidly scaling up and down with demand

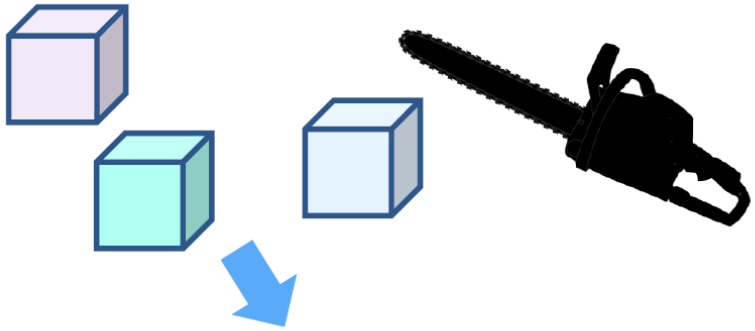
- Customer profiles
- Purchases (online orders)
- Data requests (e.g. insurance quotes)
- Website comments

Micro Service Architecture

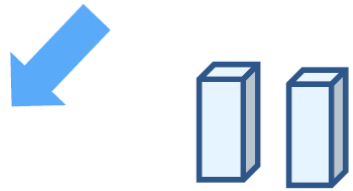
Working in a microservices framework means that applications are broken into smaller parts



So that changes to individual parts can be quickly made



And because they are independent



One change does not affect the other parts

Micro Service Architecture

A supermarket runs an app that allows customers to take advantage of special offers when they are in specific areas of the store

Chris, your developer, decides to make a quick change to the app

IT'S QUICK because he doesn't have to rebuild the entire app

OK, his code breaks, but the rest of app is unaffected

QUICK TO FIX because he doesn't have to rebuild the entire app

Why Hybrid Messaging?

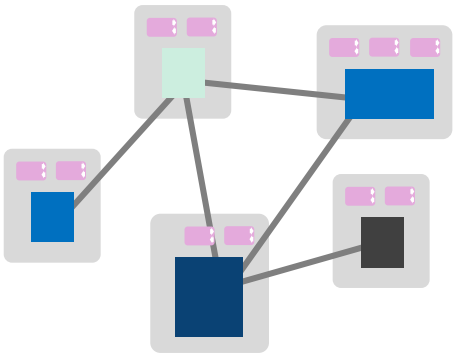
Is it to…

- run you apps, unchanged, in a cheaper environment?
- stage the migration of applications to cloud-native runtimes?
- be able to say you're "in the cloud"?
- move to micro-services model?
- enable developers?

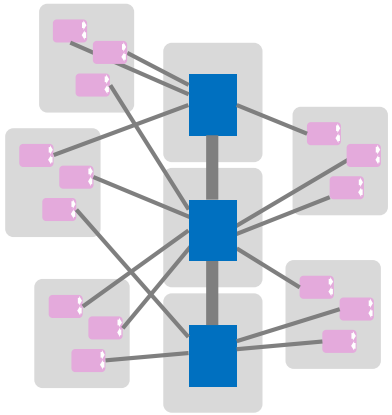
Considerations

- Cost of data egress
- What are your likely data flows
  - Mostly inbound, with small amounts of response data going back to the enterprise?
  - Mostly inbound, with no data leaving the cloud?
  - Similar levels of inbound and outbound
- If cloud apps need to intercommunicate, must they go via on-prem environment?
  - See e.g. message hub slides later, or on-cloud QM

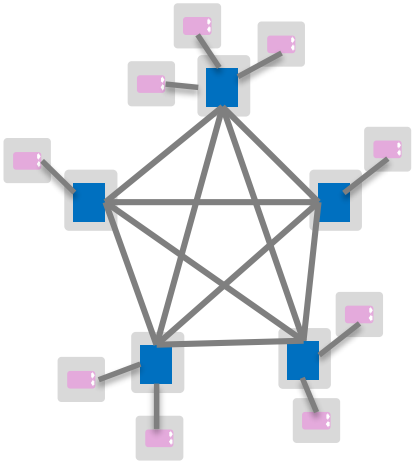
Typical MQ Architectures



Classic



Hub



Decentralized

More suited to cloud scenarios

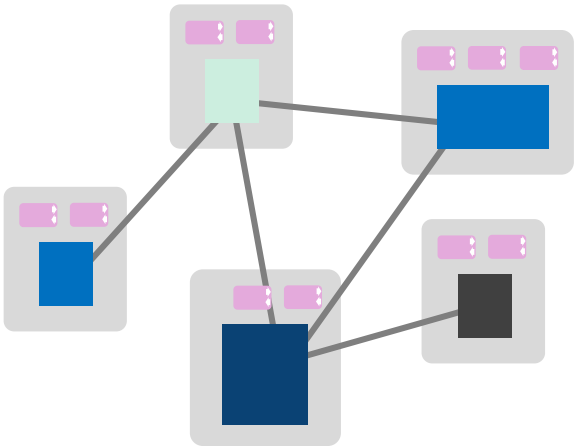
The Classic

Used for connectivity of heterogeneous systems, providing store and forward to overcome system and network outages

Isolation through dedicated queue managers, tightly bound to the application runtimes

This is one of the '*original*' deployment patterns for MQ and has often ended up as bespoke, tuned deployments for individual components

Leads to hard to deploy, manage and maintain systems over time

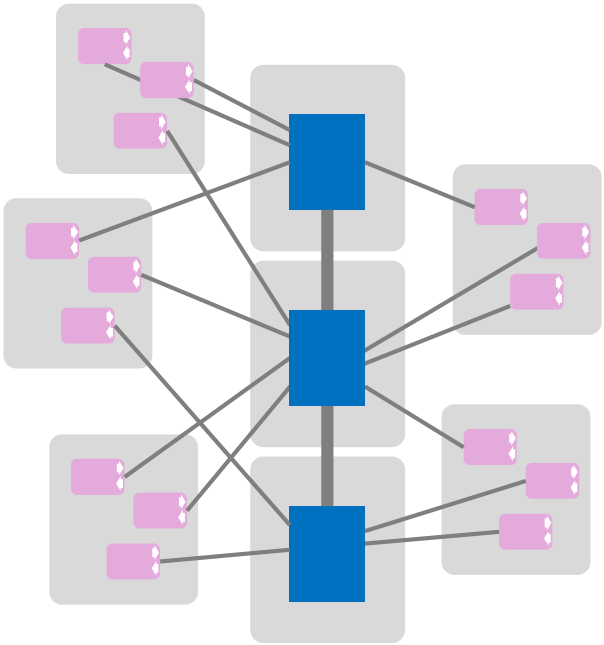


The Hub

A '*hub*' (or backbone) of systems running multiple queue managers, based on a standard deployment

Applications connecting as clients from remote systems. Loser coupling enables simpler deployments and independent scaling and maintenance

This pattern has gained popularity as networks improve and administration costs go up

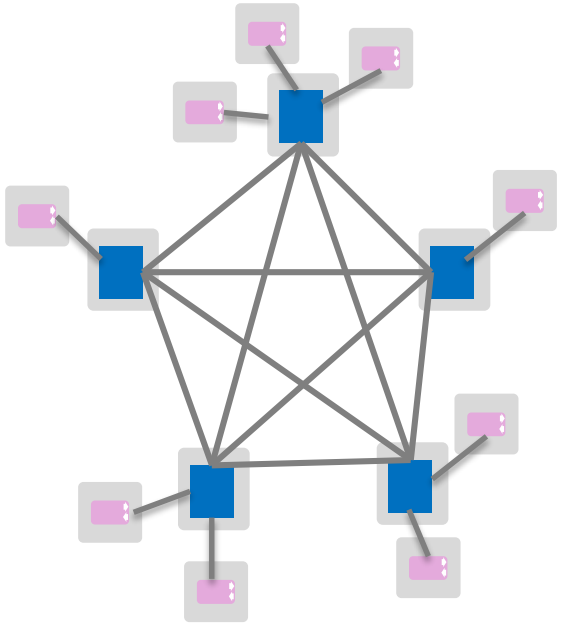


Decentralised

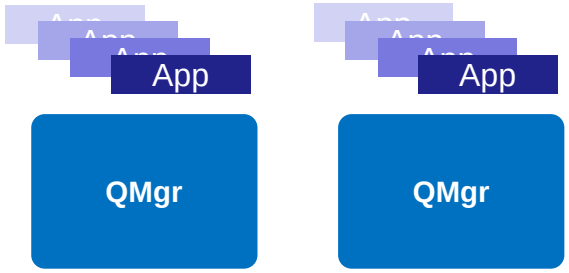
Decentralise the MQ system completely. Each line of business or application has its own infrastructure and therefore own queue managers. Client connections to separate applications from the infrastructure

Remove the central administration as much as possible to reduce bureaucracy and speed up application deployments

Has popularity as a way to satisfy greater autonomy for lines of business



Tenancy



Multi tenant

Potentially lower runtime overheads

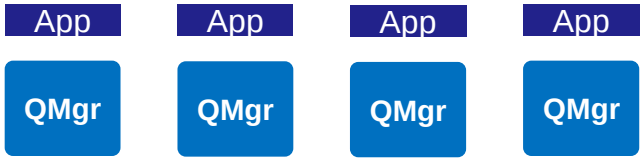
More care needed in configuring to achieve isolation

- Isolation of machine resources not possible

Harder/simpler to monitor

- Depends on your view of more queue managers

Fine grain security required



Single tenant

Simple to configure, maintain and monitor

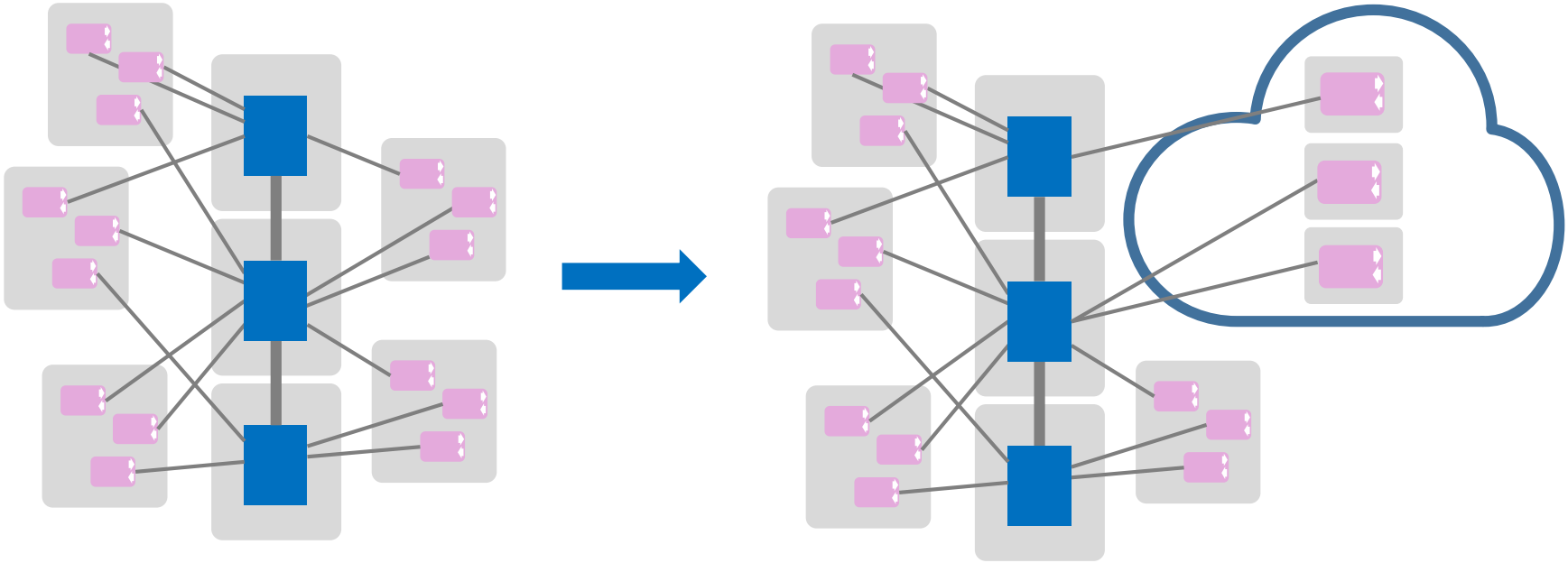
Very good isolation

A proliferation of queue managers

Harder when integration is required

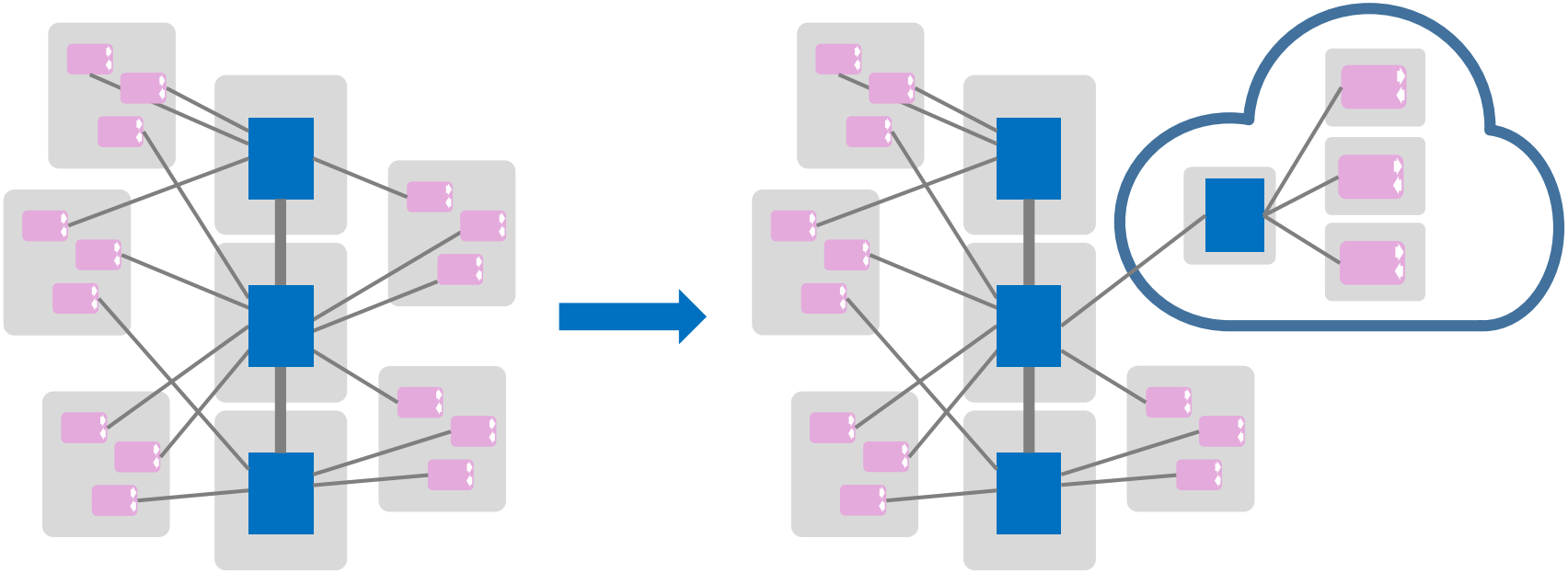
Best suited to scalable, cloud deployments

Tenancy



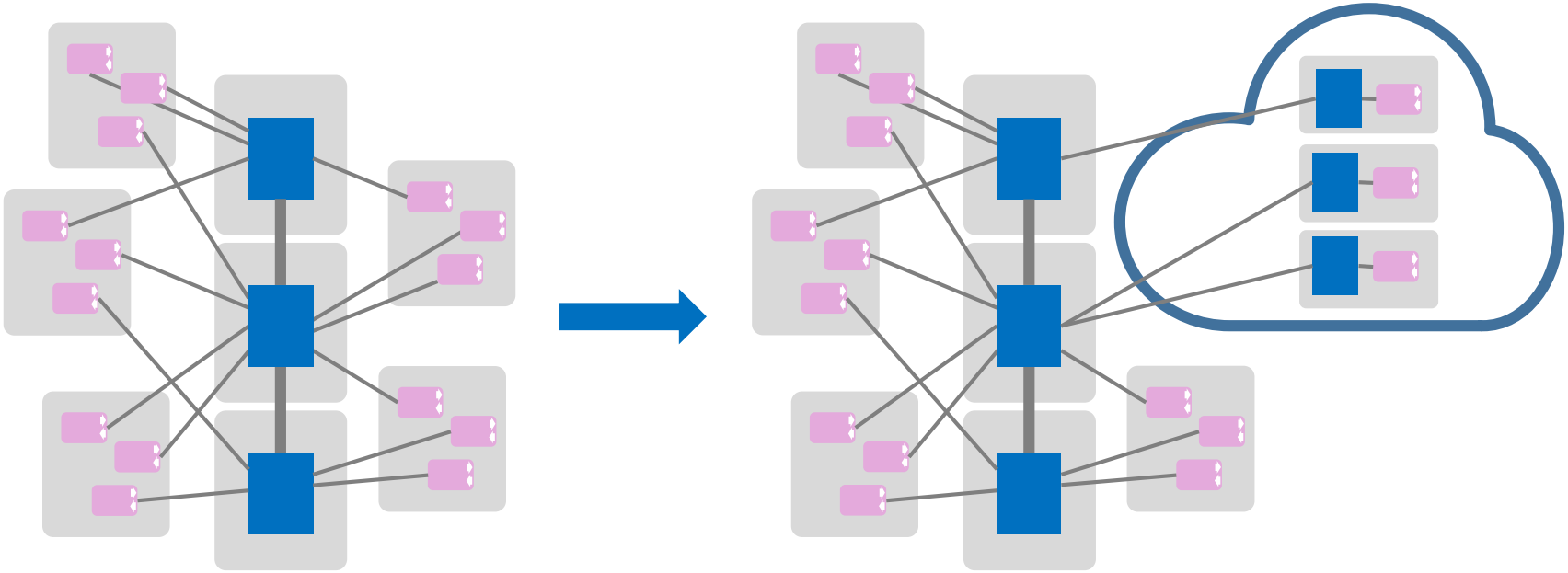
- Run MQ clients in the cloud
- Connect to on-premise hub
- Applications running in container, Cloud Foundry, serverless environment (e.g. Lambda/OpenWhisk), etc…

Tenancy



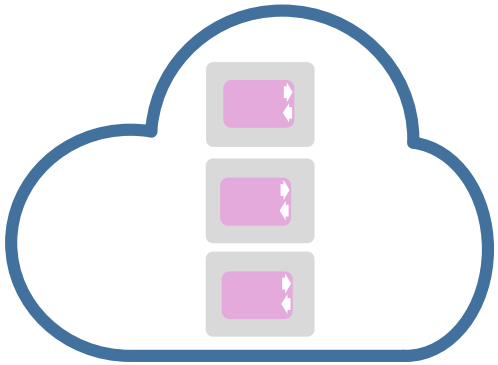
- Single queue manager run in the cloud
- Gateway QM connects to on-premise hub
- Not multi-tenancy - apps are scaled instances
- Allows some communication between cloud apps without going back to on-premise

Tenancy



- Queue managers run in the cloud alongside apps
- Connect to on-premise hub
- Run in **VMs** or **containers**
- Unless you have a good reason to run QMs along side apps this may not be the best architecture for cloud

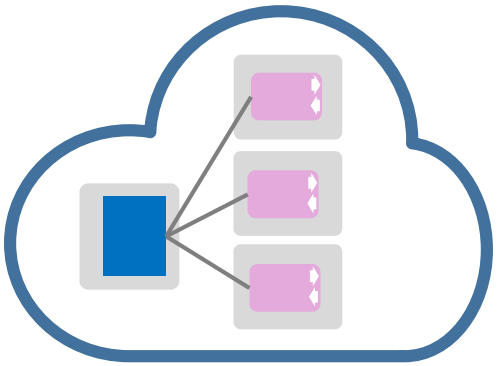
Tenancy



Clients

- Easier to scale ✓
- Stateless ✓
- Less administration ✓

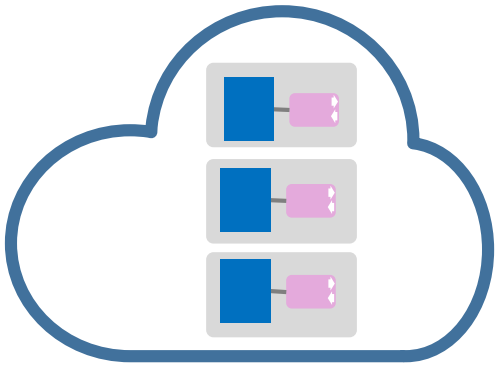
- Need to discover a QM ✗
- Can't operate during network partition ✗



Clients & Gateway QM

- Client service discovery simpler ✓
- QM manages discovery and routing ✓
- Single place to configure connectivity back to the enterprise ✓

- Limits app scalability ✗
- Not very cloudy ✗

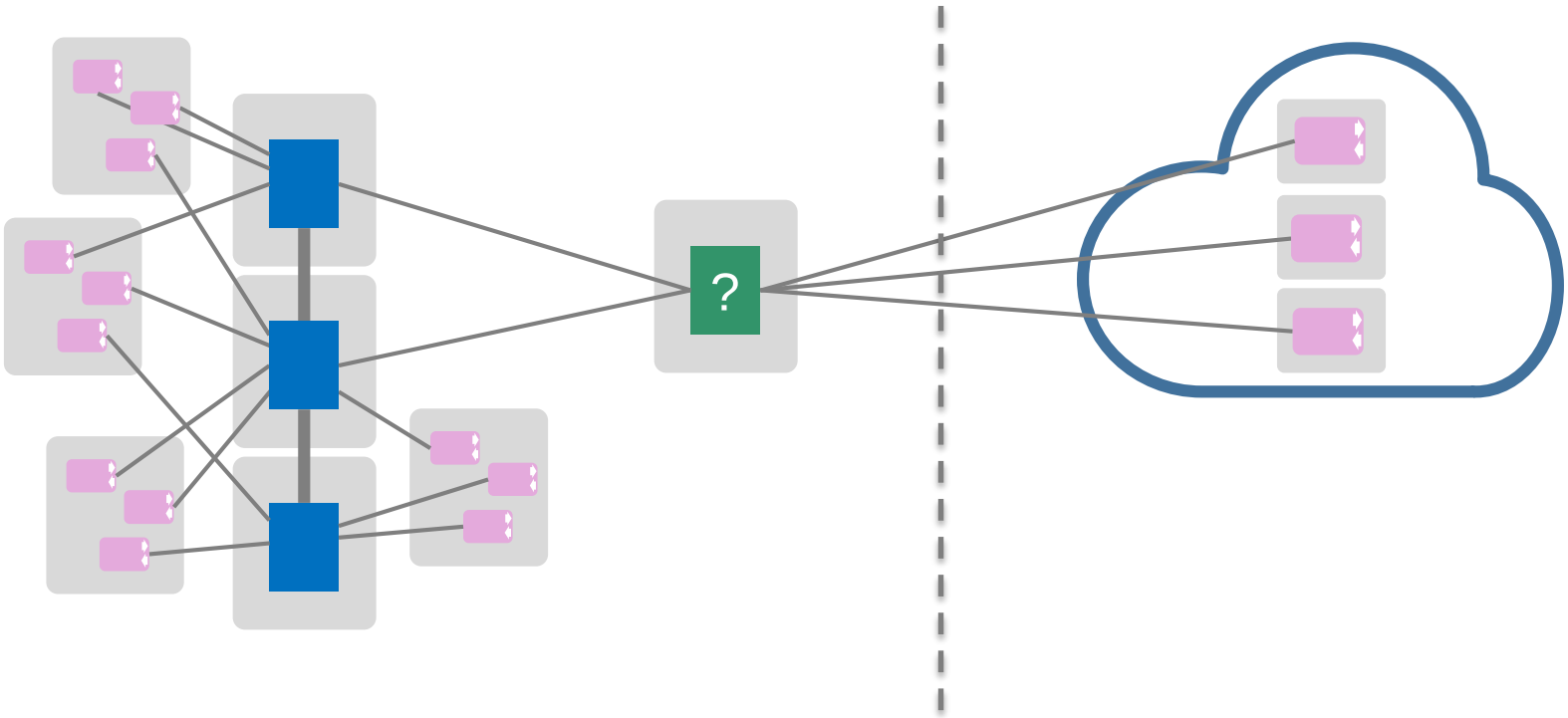


Clients & QMs

- QM buffers messages between outages ✓
- Client service discovery easier ✓

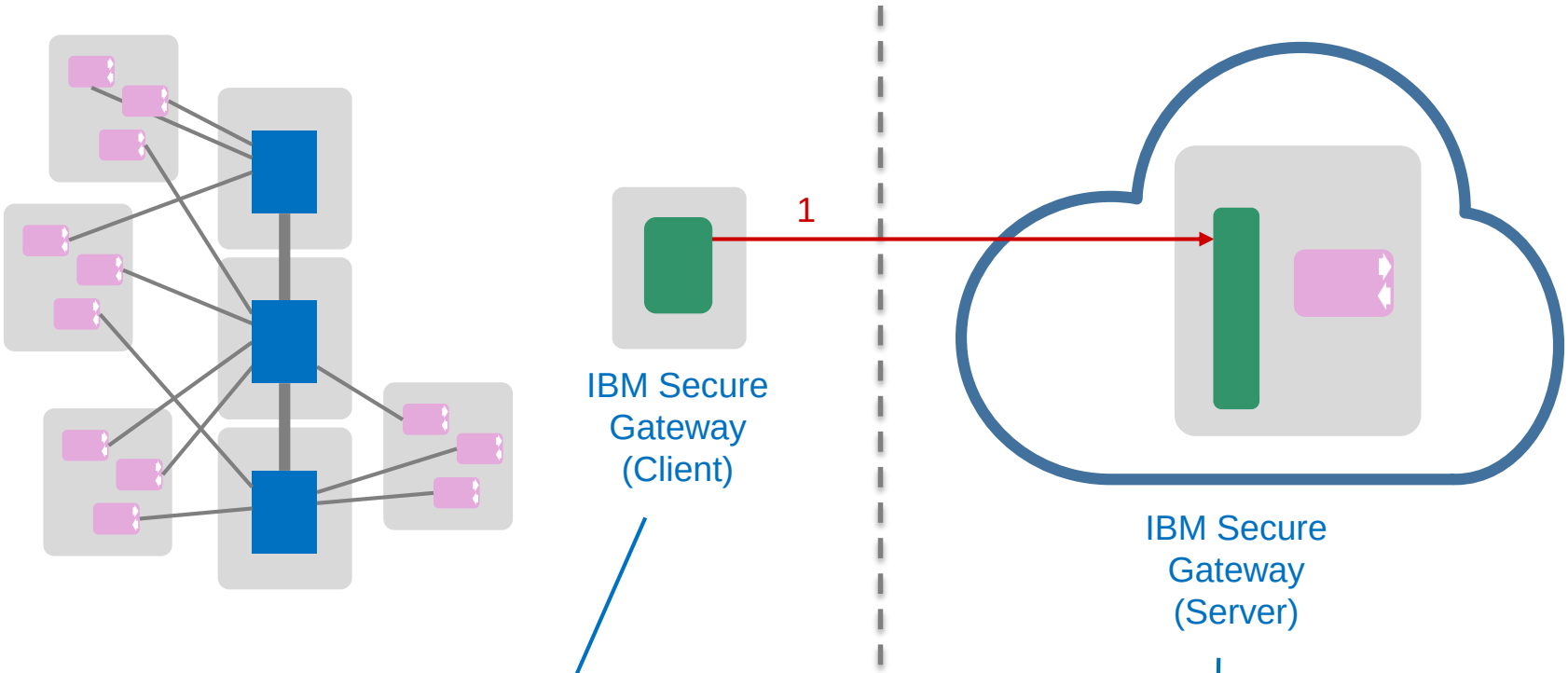
- More admin required ✗
- Need access to each QMs logs ✗
- Harder to scale down ✗
- Can apps really do anything during an outage anyway? ✗

Connectivity



- Like connecting from any other external network, need to route connectivity through firewall/DMZ
- All cloud platforms provide ways to connect on-premise and cloud networks (e.g. IBM SecureGateway, DirectConnect, VPN)

Connectivity – IBM Secure Gateway

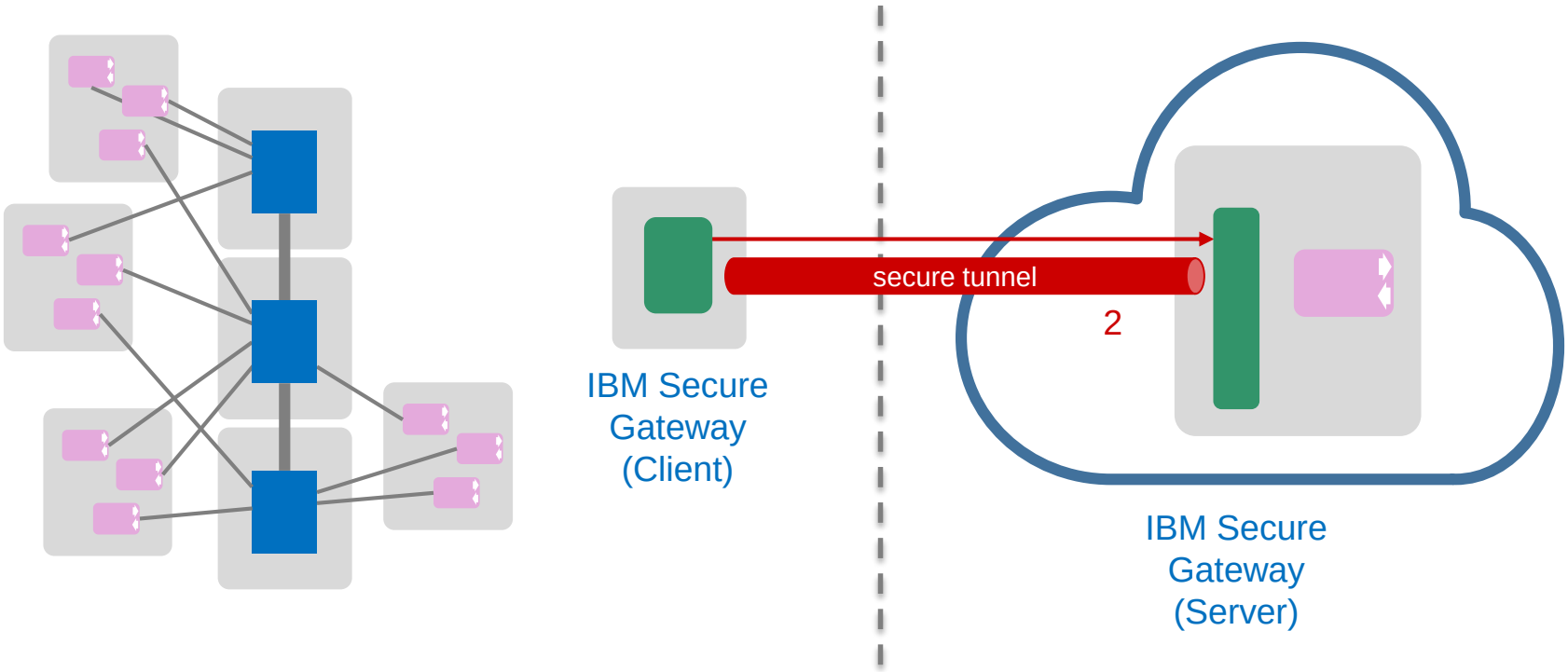


Secure Gateway client runs on-premise

- Native Mac/Linux/Win app
- Docker
- DataPower

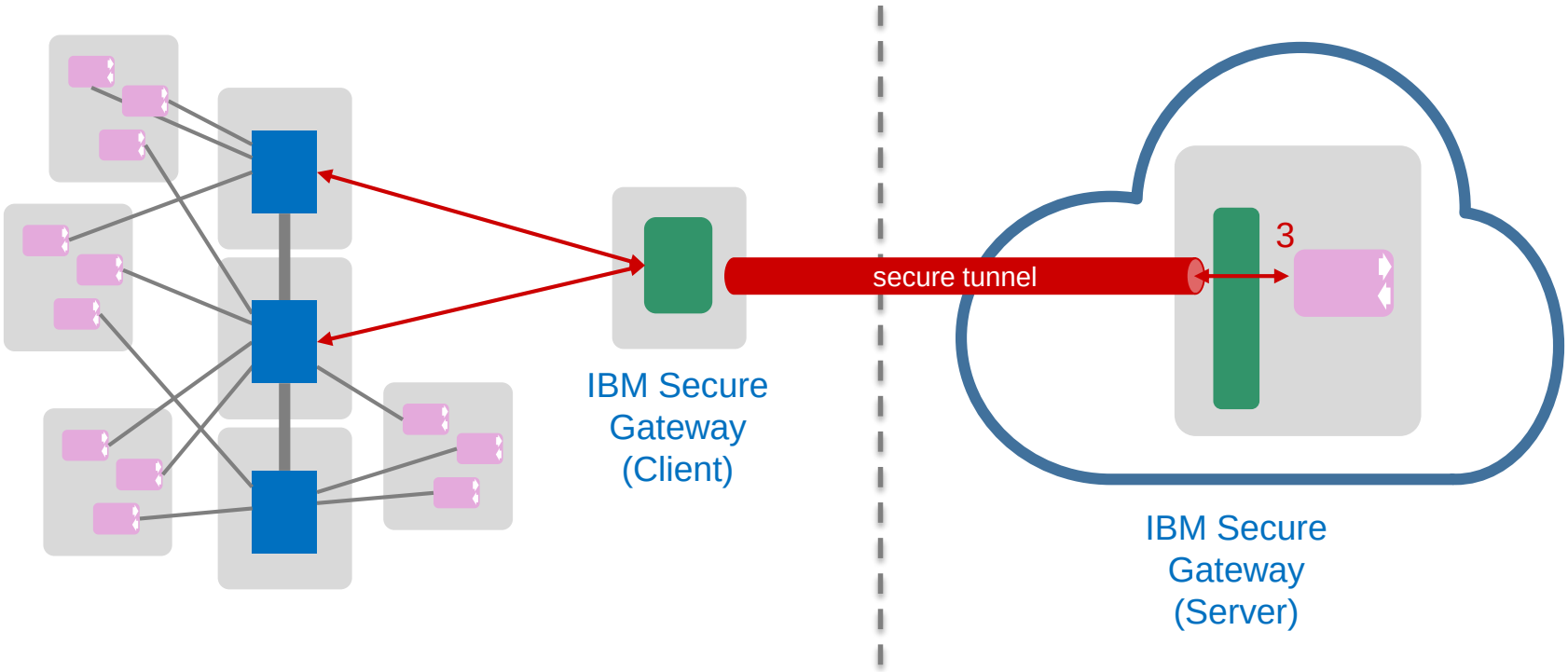
Connects to Secure Gateway server

Connectivity – IBM Secure Gateway



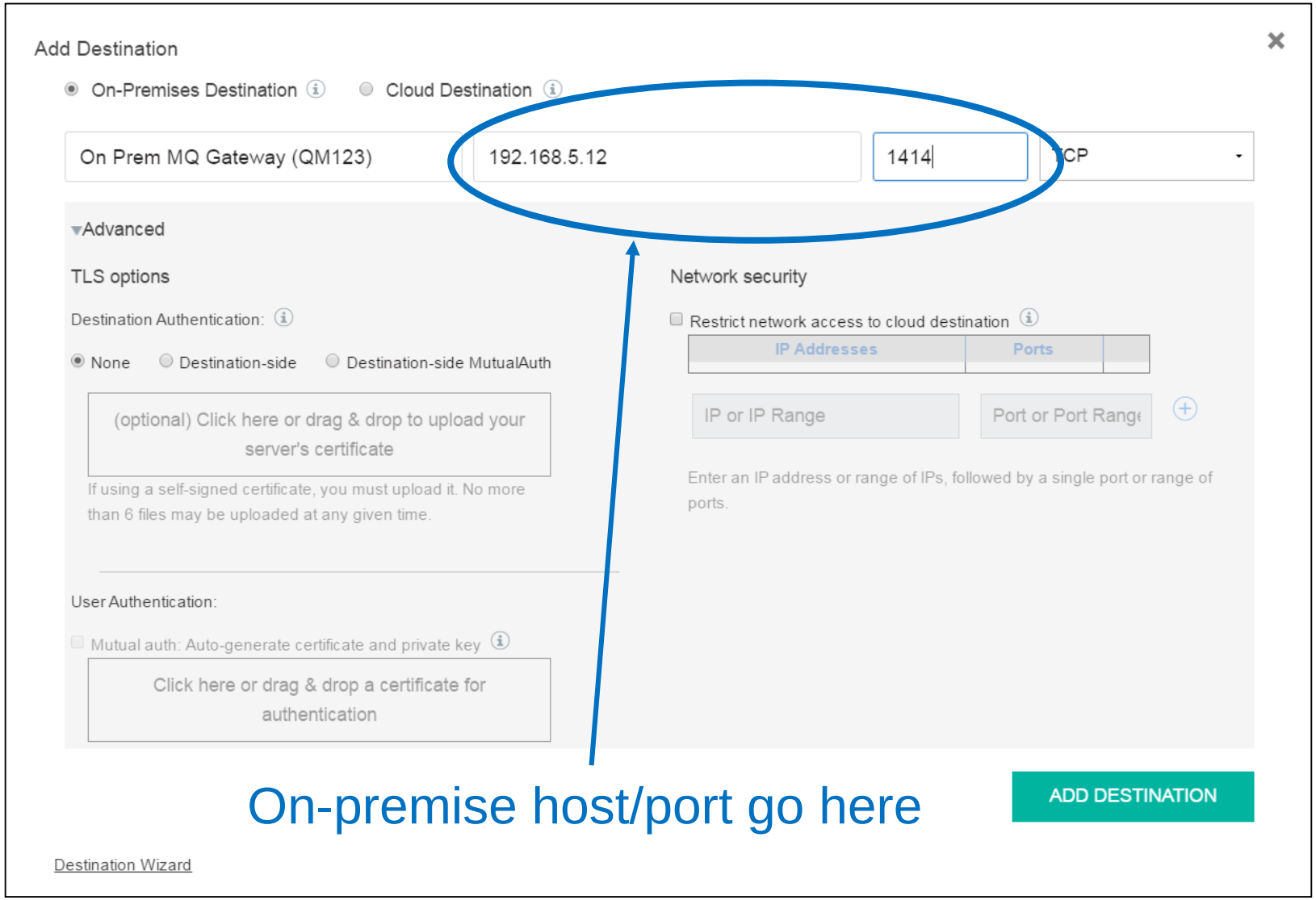
- Secure Gateway sets up a tunnel from cloud network to on-premise client

Connectivity – IBM Secure Gateway



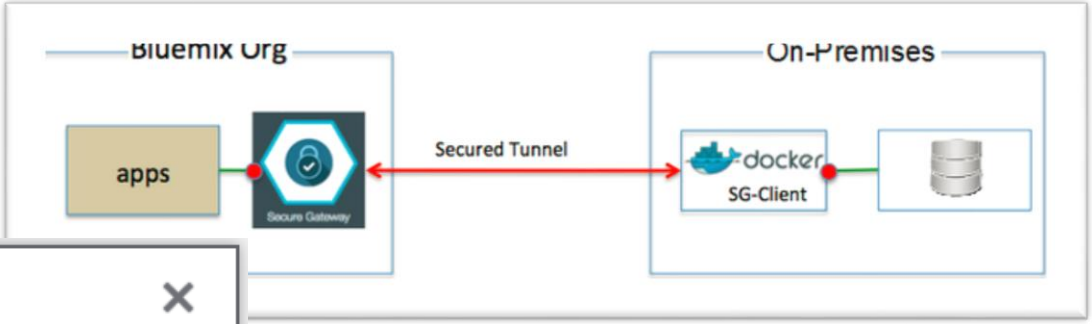
- You configure valid routes from Secure Gateway client to on-premise network interfaces
- Cloud application connects to virtual address in cloud e.g. ***cap-sg-prd-1.integration.ibmcloud.com:17036***
- Secure gateway client routes packets to/from on-premise network

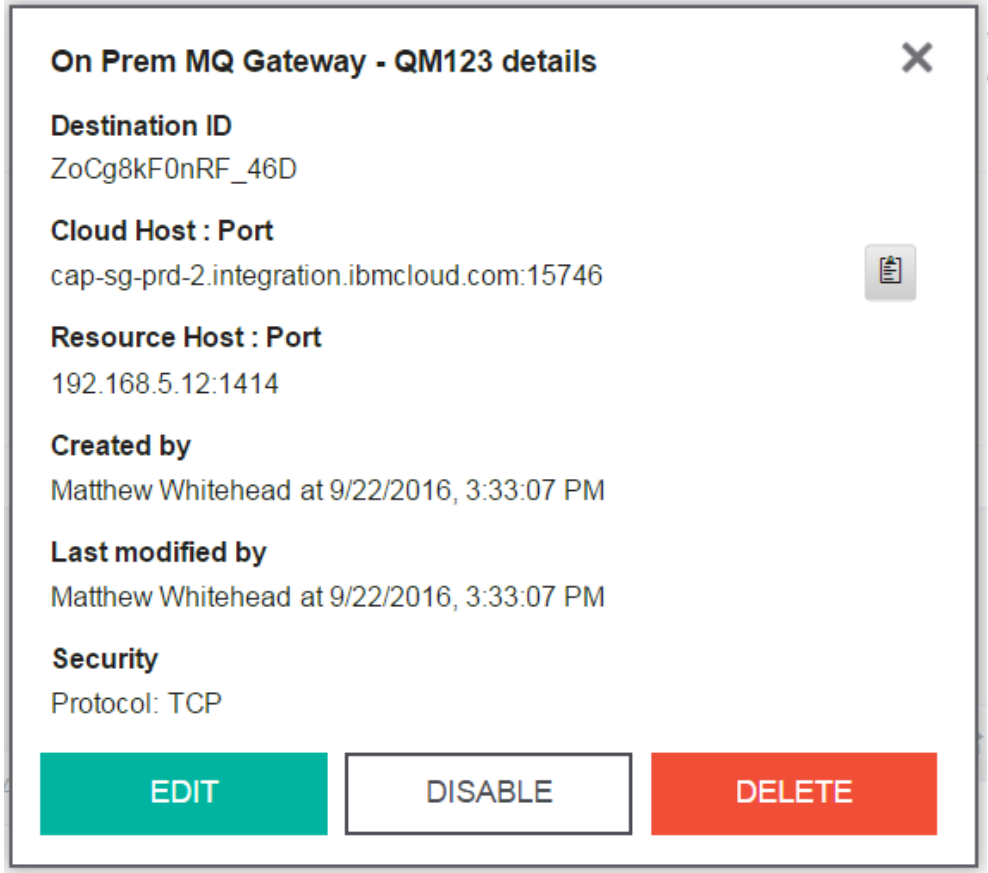
Secure Gateway Destinations



On-premise host/port go here

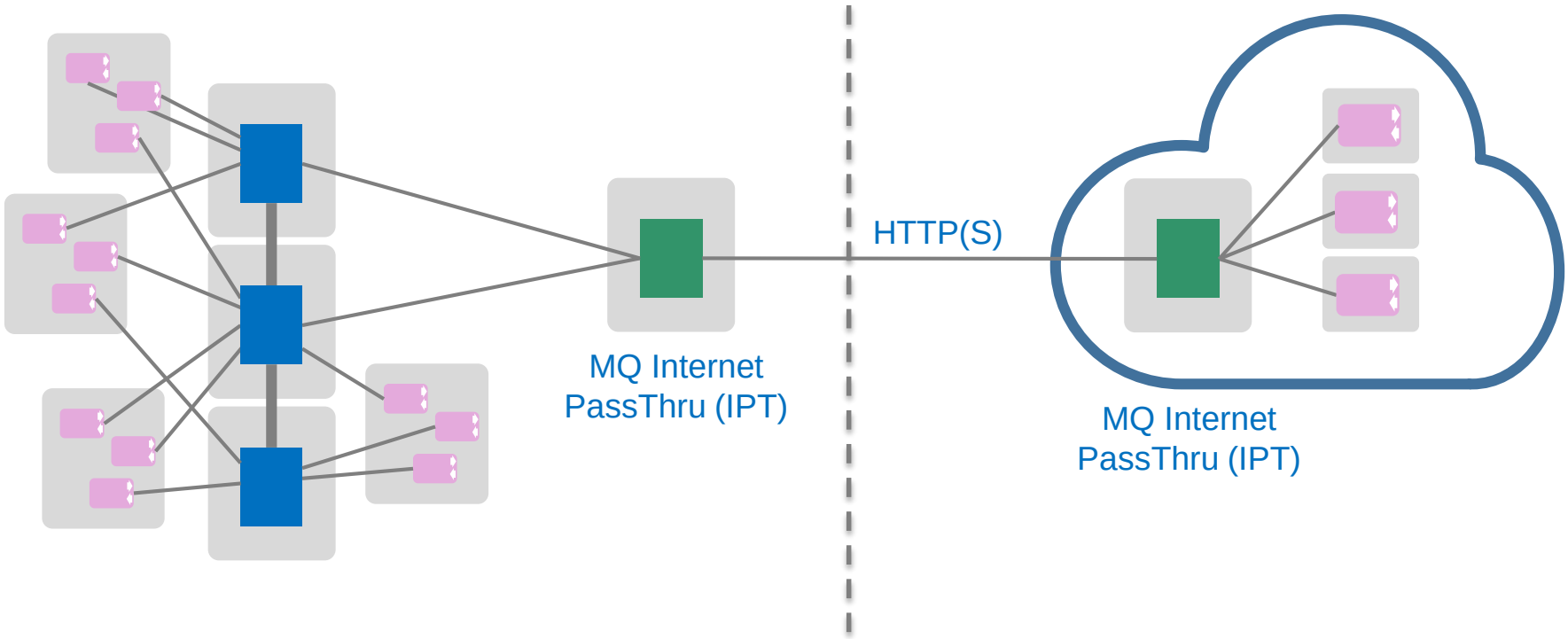
Secure Gateway Destinations





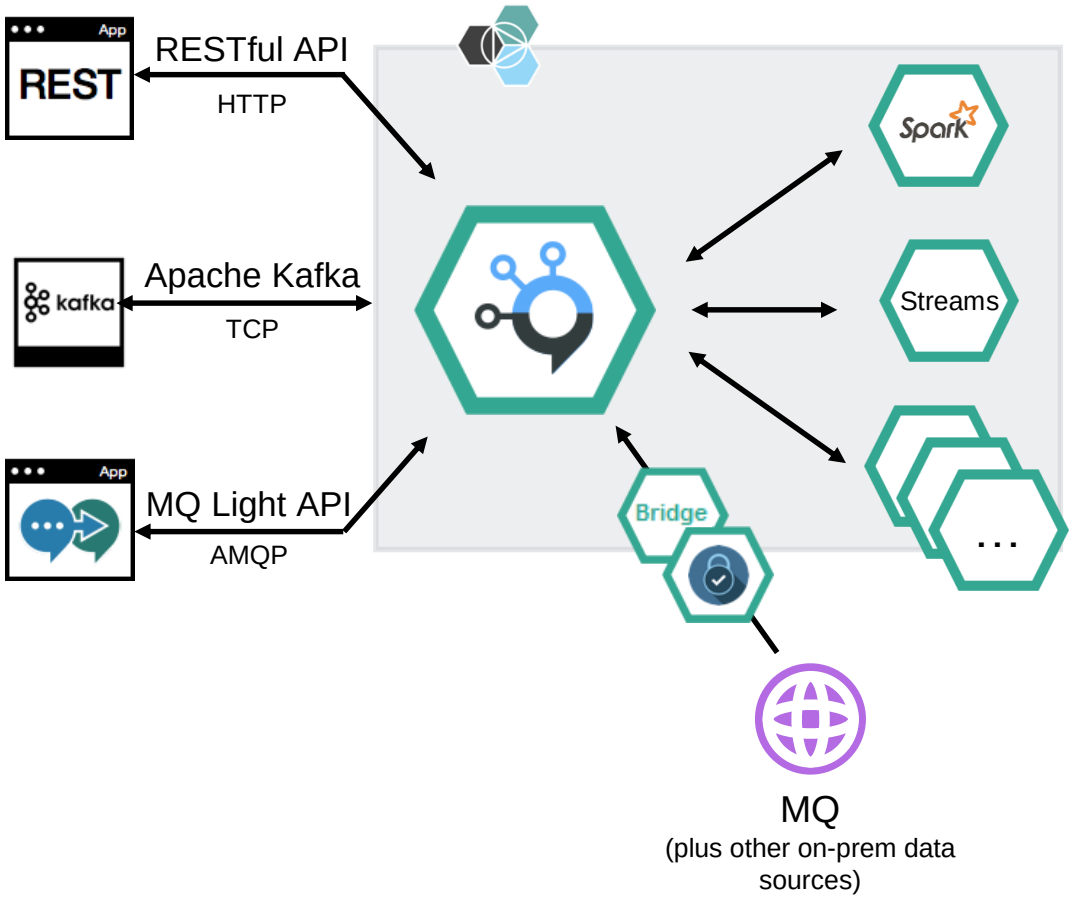
- Once an on-premise destination IP address has been defined, the secure gateway allocates a host name and port
- Your Bluemix application connects to this virtual host name
- The secure gateway routes traffic to the on-premise address 192.168.5.12:1414

Connectivity – MQ IPT



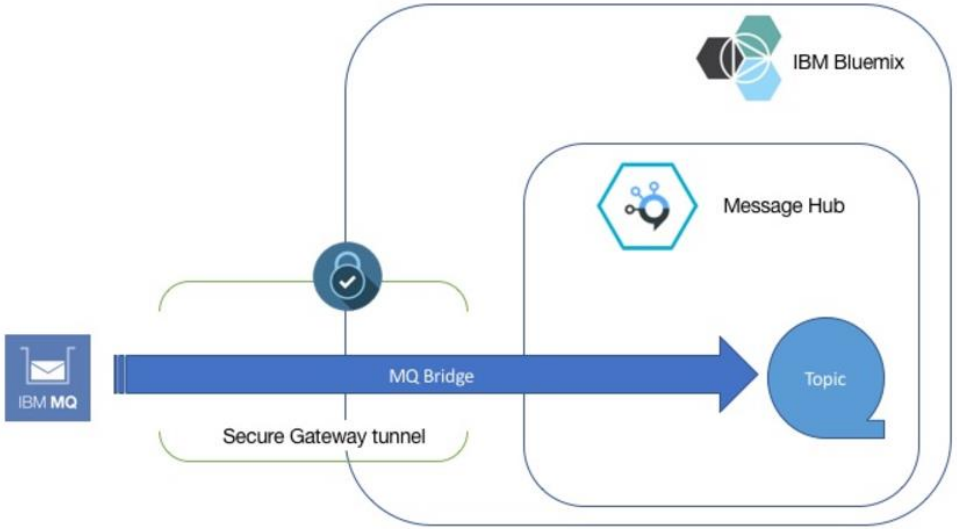
- Avoids the need for a direct TCP connection from cloud to on-prem
- Tunnel MQ traffic over HTTP(S)
- Avoids requirement for more complicated VPN configuration
- Re-use on-prem IPT if you're already using it
- Cloud agnostic

Message Hub



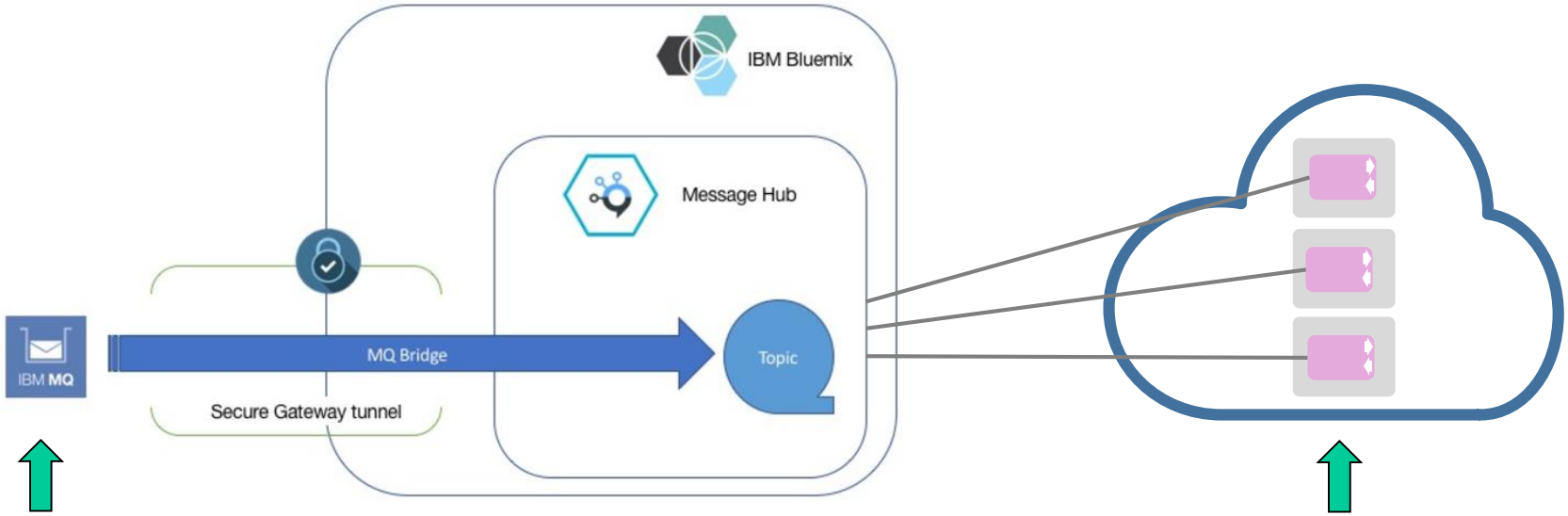
- Max 1 MB message size
- Max 30 day message retention

IBM Message Hub <-> MQ Bridge



- Ingest messages from MQ onto Kafka topic
  - One way only
- Connects as a client to MQ
- Use Secure Gateway to tunnel into on-prem network
- E.g. stream MQ publications to Kafka for realtime analysis with Apache Spark

IBM Message Hub <-> MQ Bridge



Gives you access to your enterprise data here

Gives you a naturally cloud-designed infrastructure here

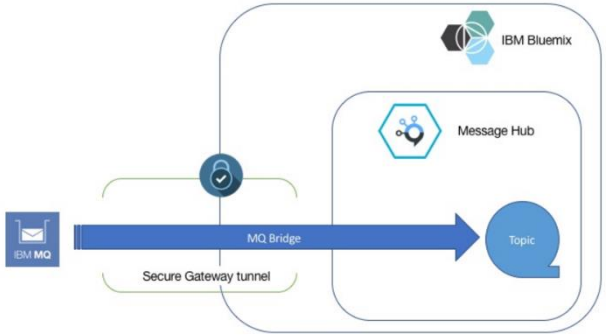
Encourages you to use cloud-native technologies here

(but doesn't suit some traditional messaging topologies e.g. request/reply)

Message Hub Performance

Throughput of the cluster is about 300,000 msg/s

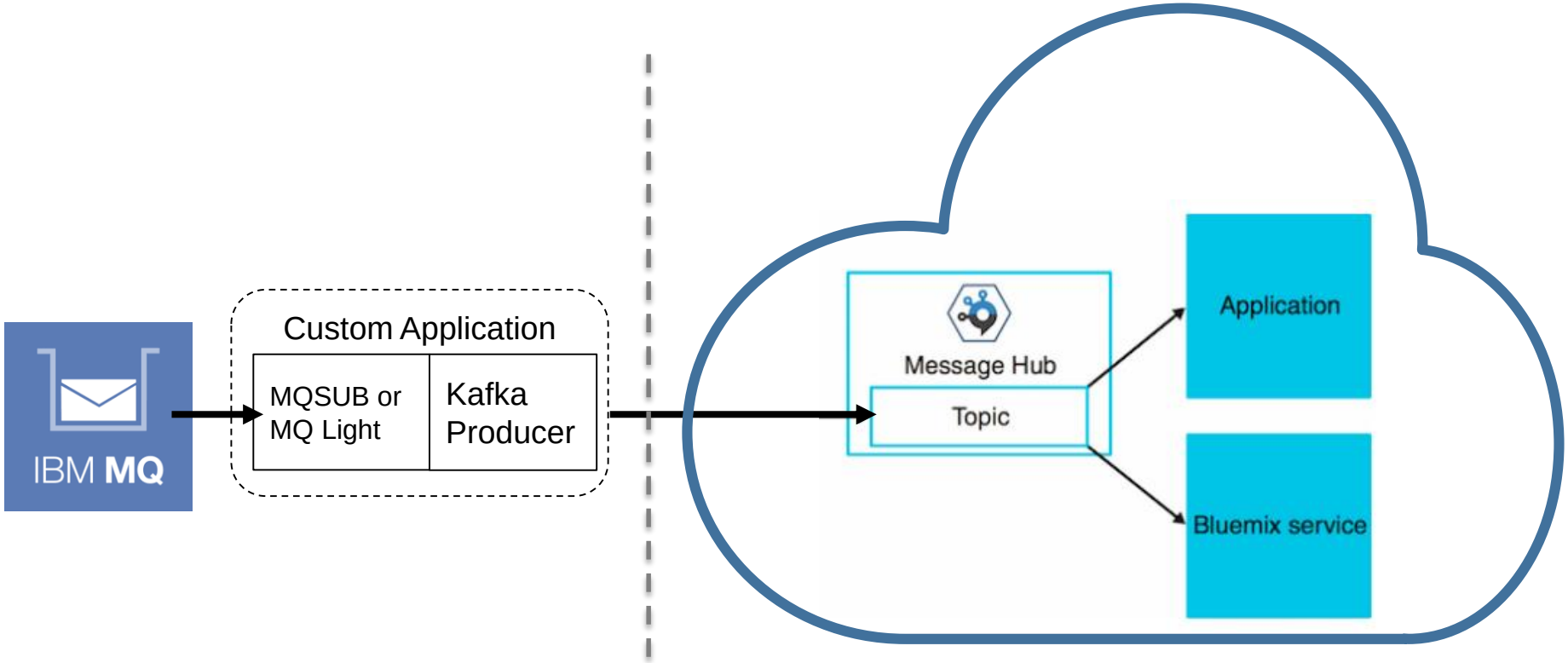
- 100 byte messages, max 1MB
- Secure connection from public network
- Disk encryption of Kafka log
- Secure connections for all user data



Latency ranges from 20ms to 100ms

- Average ~50ms

Other Approaches – Custom application



- Message Hub endpoints are on the public internet
- Complexity of tunnels might be overkill to PoC a cloud deployment for certain applications before configuring more permanent infrastructure
- MQ Light = nodejs/Ruby/Java = quick & easy MQ-to-Kafka bridge

Client Runtimes

- MQ offers a lot of different application runtime options
  - C, C++, JEE, CICS…
- Are there better runtimes for your new cloud-era applications?
- New concepts like serverless programming suit some runtimes over others
- E.g. AWS Lambdas
  - Nodejs
  - Java
  - Python
  - .Net C#

Client Runtimes

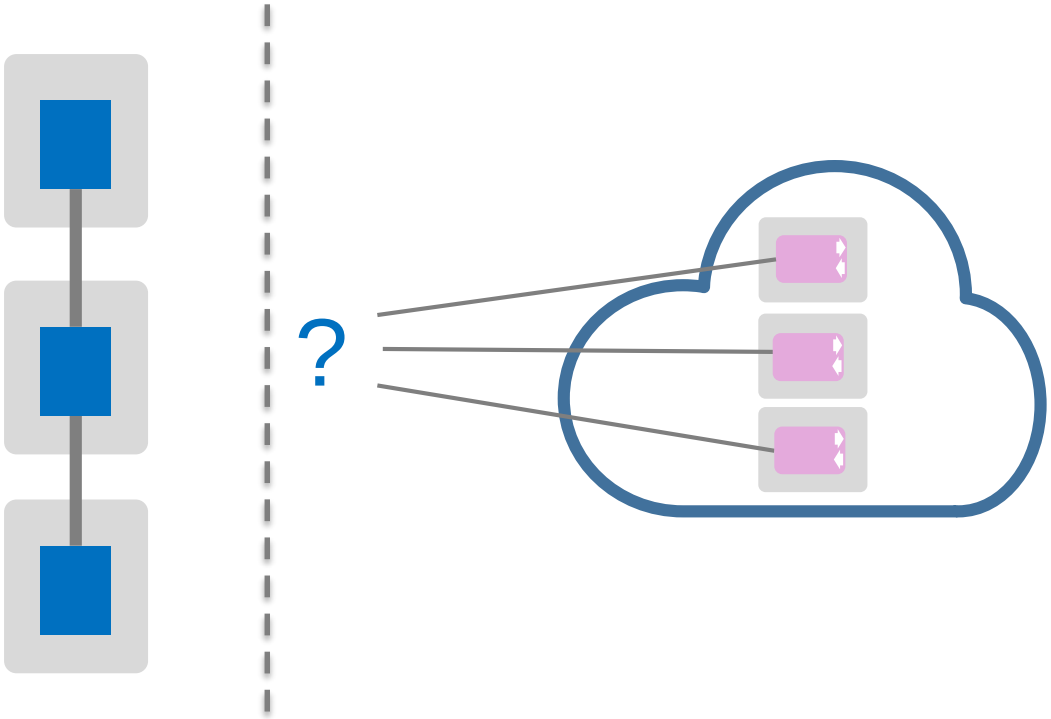
- Bluemix Functions based on Apache OpenWhisk
  - Java
  - Swift
  - Node
  - Python
  - PHP
  - Docker

Client Runtimes

- Bluemix Cloud Foundry supported buildpacks
  - Java
  - .Net Core
  - Nodejs
  - PHP
  - Python
  - Ruby
  - Go
- but you can still push native MQ apps to Cloud runtimes as we'll see later…

Service Discovery

- Clients need to discover where to connect
- Can be done a number of different ways
  - MQSERVER env
  - CCDT (MQCHLLIB & MQCHLTAB, MQCCDTURL)
  - mqclient.ini
  - JNDI
- But also…
  - MQ Light client service lookup (JSON)
  - DNS
  - Key/value store



Cloudifying native apps

- You can still deploy native applications to cloud platforms
  - See binary buildpack for cloudfoundry
  - See OpenWhisk docker support for generic/non-cloud-native languages

A Side Note – MQ Redistributable Clients

fix pack: 9.0.0.0-IBM-MQC-Redist-Win64
IBM MQ C and .NET redistributable client
Click here for product readme Click here for installation instructions

Windows
C & .Net

fix pack: 9.0.0.0-IBM-MQC-Redist-LinuxX64
IBM MQ C redistributable client
Click here for product readme Click here for installation instructions

Linux C

fix pack: 9.0.0.0-IBM-MQC-Redist-Java
IBM MQ JMS and Java redistributable client
Click here for product readme Click here for installation instructions

Java/JMS

- Also available for MFT client libraries (create transfers, query agents etc)
- Create your own redistributable packages by stripping out unused libraries
  - See *genmqpkg.sh*

A Side Note – MQ Redistributable Clients

```
mwhitehead@mrw-ubuntu-1604: ~/redist-mq-client/bin

mwhitehead@mrw-ubuntu-1604:~/redist-mq-client/bin$ ./genmqpkg.sh
Generate MQ Runtime Package
---------------------------
This program will help determine a minimal set of runtime files that are
required to be distributed with a client application. The program will
ask a series of questions and then prompt for a filesystem location to
copy the subset of files to.

Note that IBM can only provide support assistance for an unmodified set
of redistributable runtime files.

Does the runtime require 32-bit application support [Y/N]? n
Does the runtime require support for languages other than English [Y/N]? n
Does the runtime require C++ libraries [Y/N]? n
Does the runtime require COBOL libraries [Y/N]? n
Does the runtime require SSL/TLS support [Y/N]? n
Does the runtime require AMS support [Y/N]? n
Does the runtime require CICS support [Y/N]? n
Does the runtime require any administration tools [Y/N]? n
Does the runtime require any RAS tools [Y/N]? n
Does the runtime require any sample applications [Y/N]? y

Please provide a target directory for the runtime package to be created
/home/mwhitehead/my-redist-client

The redistributable image will be created in

/home/mwhitehead/my-redist-client

Are you sure you want to continue [Y/N]? y

Generation complete !
Redistributable client image copied to '/home/mwhitehead/my-redist-client'

mwhitehead@mrw-ubuntu-1604:~/redist-mq-client/bin$
```

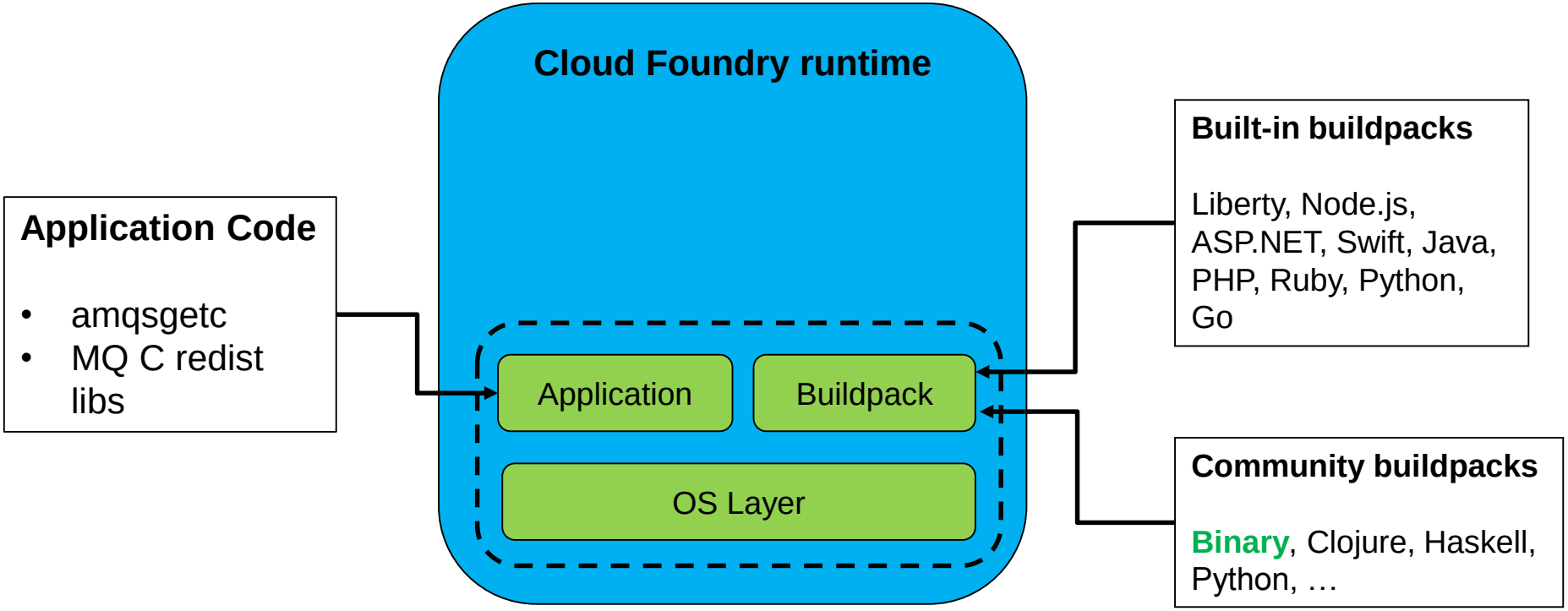
Choose packages to include

Specify a directory to create the package

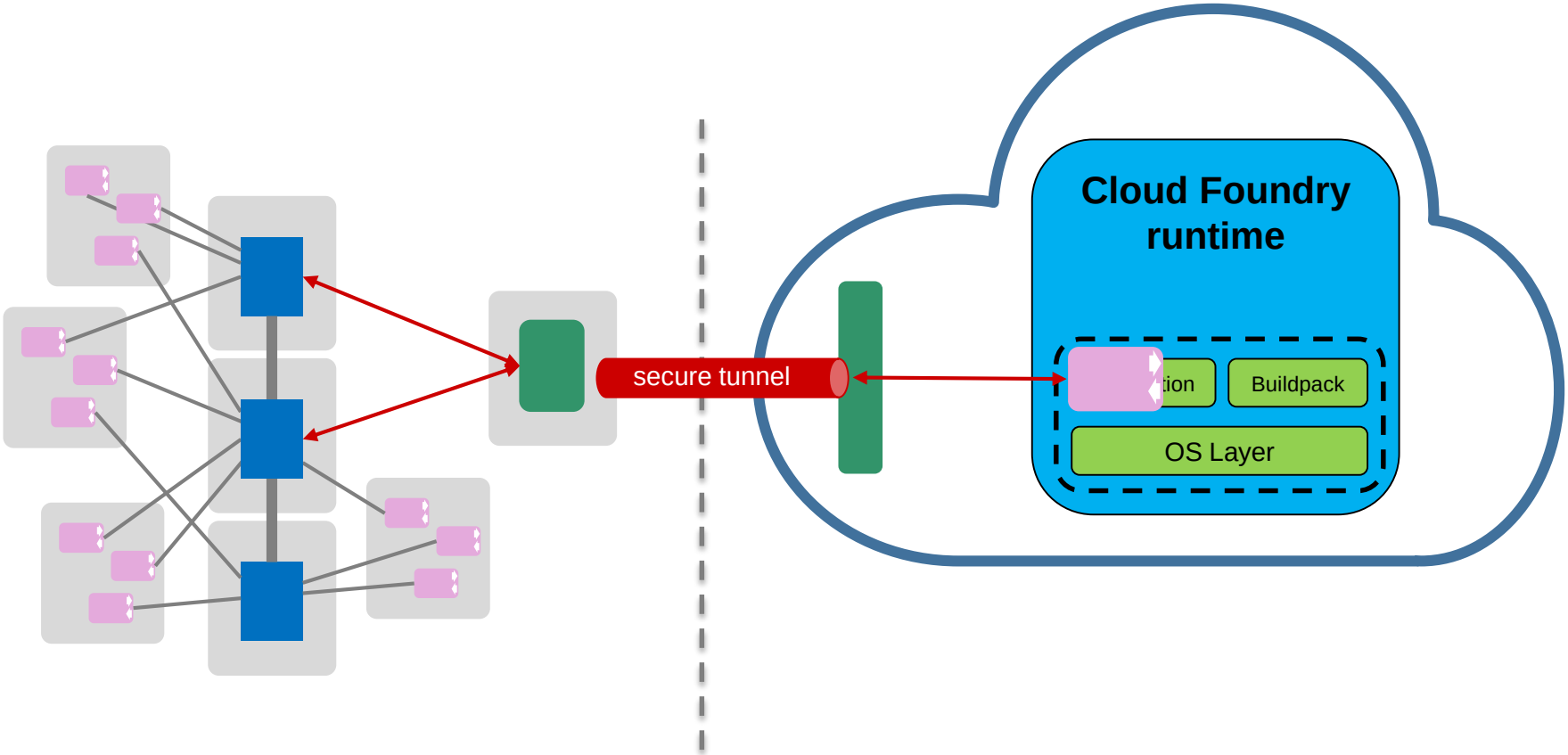
Native applications in Cloud Foundry™

Running an MQ C client in Cloud Foundry™, and connecting it to on-premise MQ

Matthew Whitehead
Published on July 5, 2017 / *Updated on September 15, 2017*



Native applications in Cloud Foundry™



Autoscaling

- Cloud environments build in auto-scaling options
- Cloud Foundry auto-scaling
  - For languages like nodejs and Ruby and Java, there are auto-scaling addons
    - CPU
    - JVM Heap
    - Memory
- AWS EC2 auto-scaling
  - Offers various auto-scaling options
    - Schedules e.g. Monday-Saturday 8am-6pm scale up, otherwise scale back

Serverless Functions

IBM Cloud Functions

AWS Lambdas

Azure Functions

Google Cloud Functions

- Ideal for short-lived application logic
- Only pay for the time functions are executing
- Like PaaS, you don't worry about the OS environment or the application runtime (JVM, nodejs runtime, Pyhon interpreter etc.)
- Just write your function and AWS will invoke it when a defined action occurs
- Scalability and availability is an inherent part of the architecture
  - 1 event = 1 function invocation
  - 10 concurrent events = 10 concurrent function invocations

Serverless Functions

How can you drive MQ servless applications?

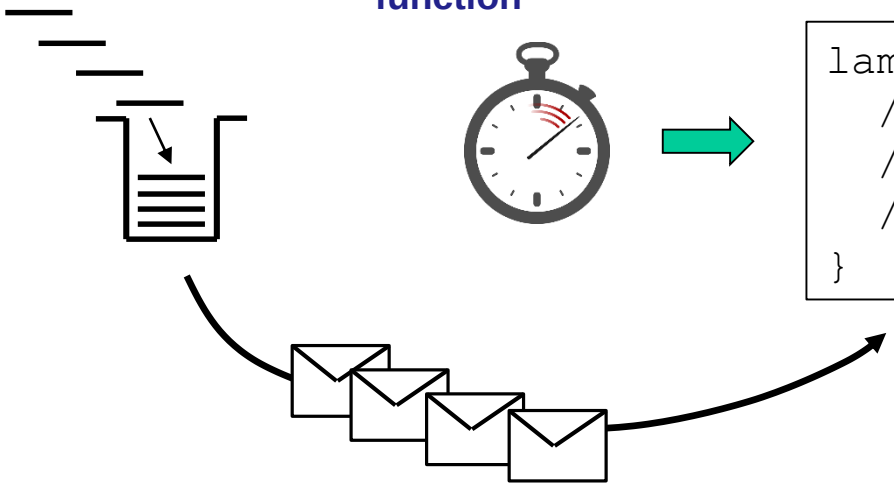
It is difficult since serverless functions don't generally support long-lived connections

One option - use timer events to invoke functions, e.g.

1. Messages arrive for a subscription

2. Timer periodically invokes lambda function

3. Function connects to QM, consumes messages, returns.

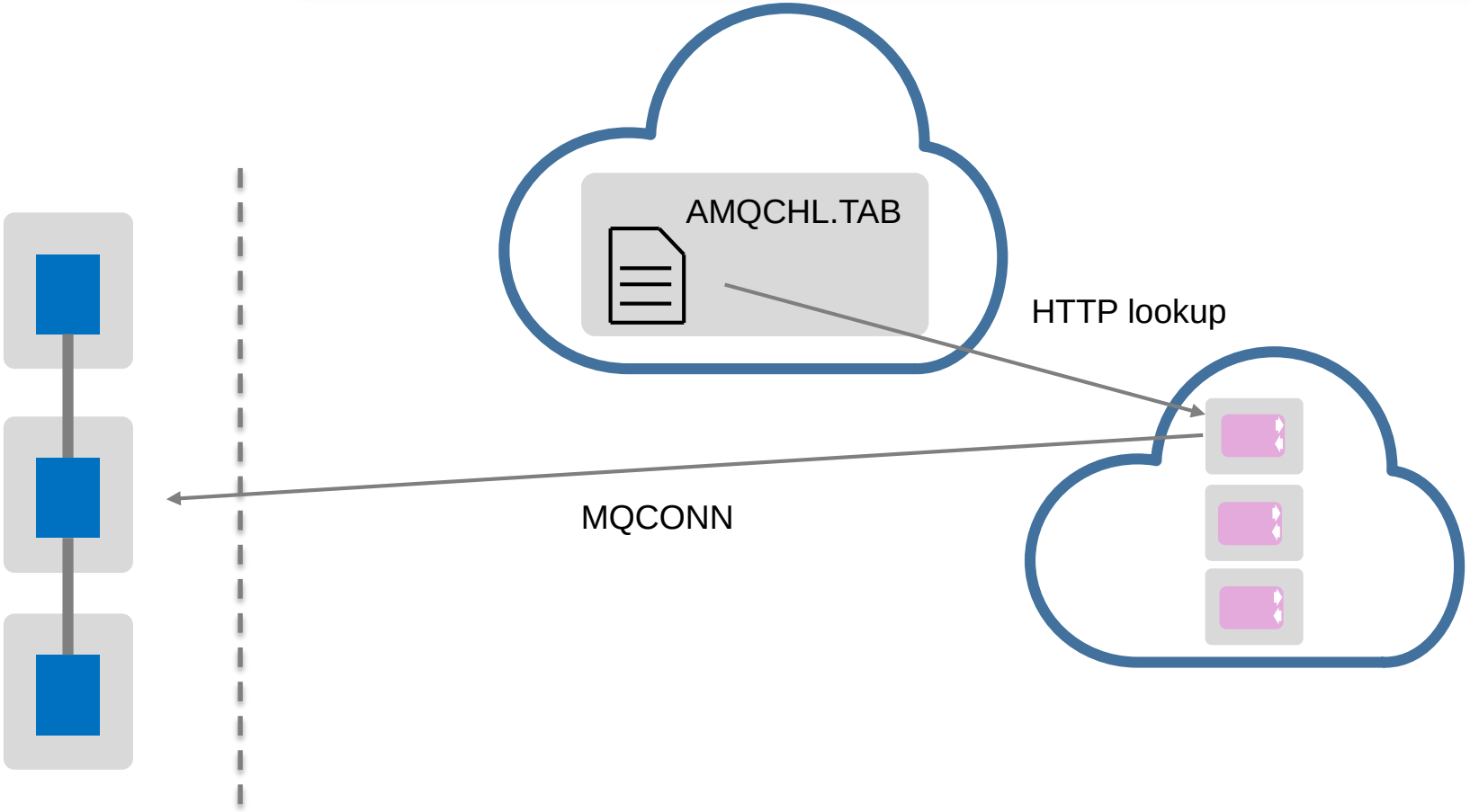


```
lambdaFunction(…) {
  // Connect to MQ
  // Retrieve & process msgs
  // return
}
```

CCDT retrieval over HTTP

MQ on OpenStack, part three: Automated client connection PoC using MQ v9 CCDT URL feature.

RobParker | Aug 17 2016 | Comment (1) | Visits (2714) 1 Like

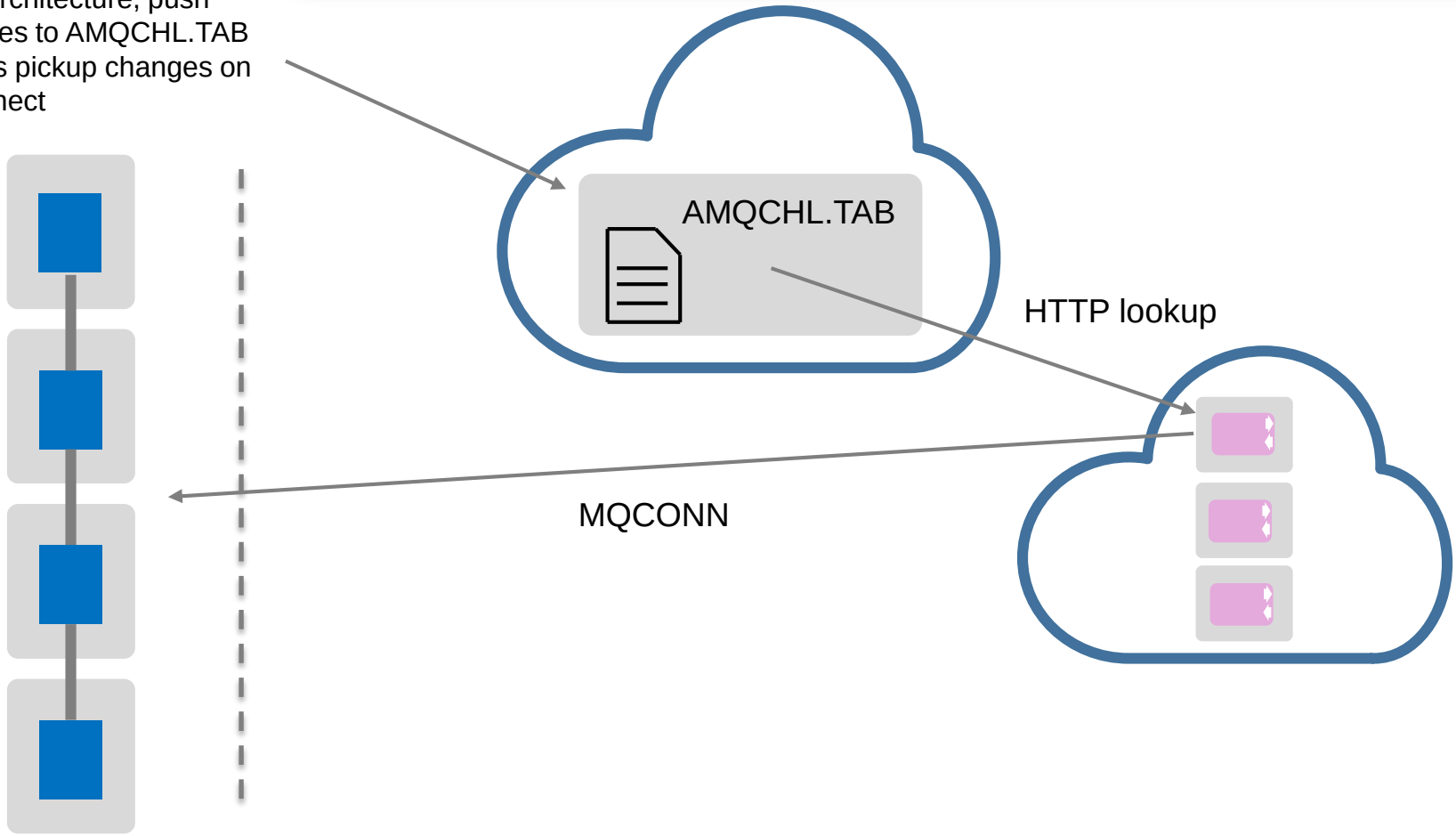


Robs blog on CCDT URLs

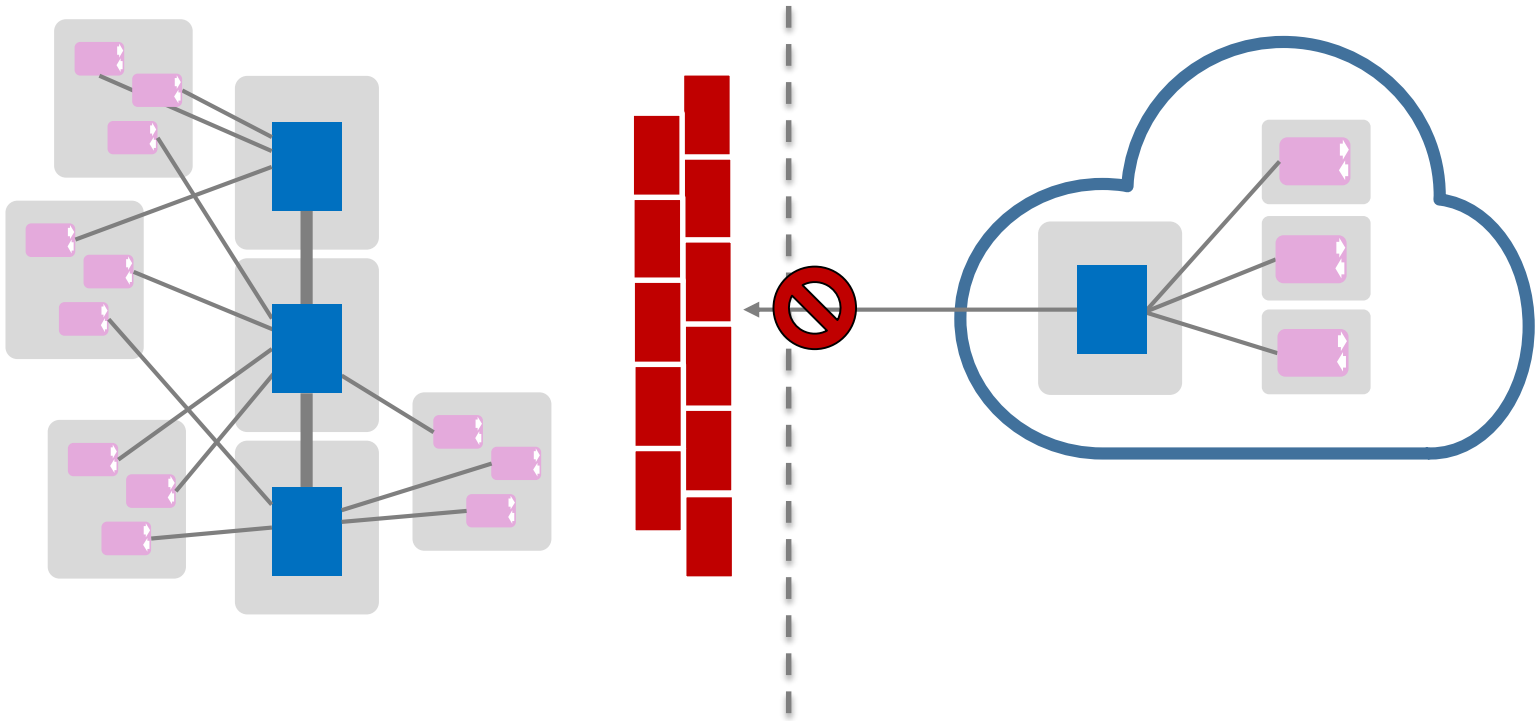
MQ on OpenStack, part three: Automated client connection PoC using MQ v9 CCDT URL feature.

RobParker | Aug 17 2016 | Comment (1) | Visits (2714) 1 Like

- When you need to change your architecture, push changes to AMQCHL.TAB
- Clients pickup changes on reconnect

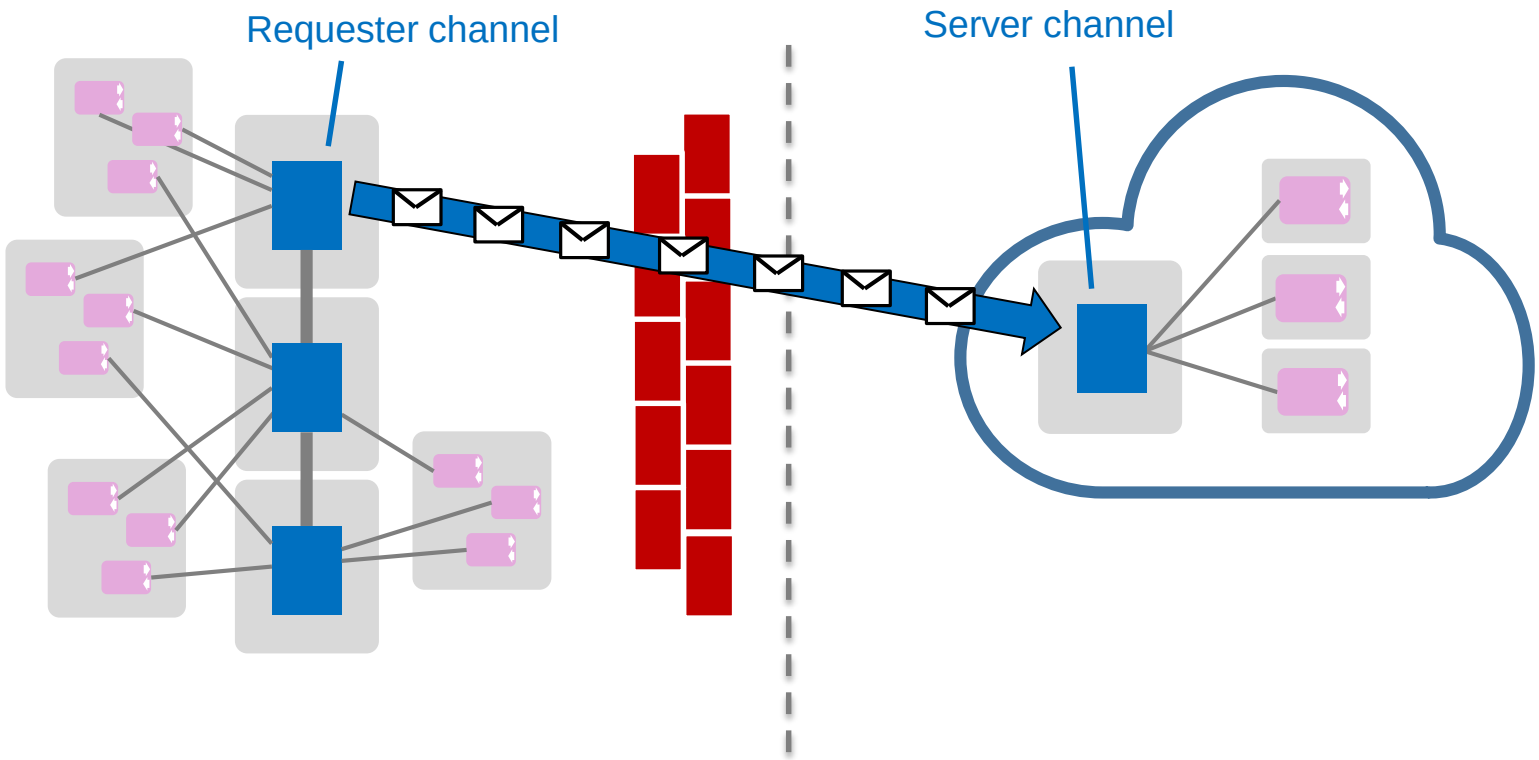


Server/Requester channels



- Enterprise network behind firewall
- Cloud queue manager on public facing IP address
- Cloud can’t connect directly to enterprise QM, but…
  - Enterprise QM can connect to cloud and request data

Server/Requester channels



- Request channel initiates connection to cloud QM
- Server channel sends data back on the connection initiated by the requester channel

Thank You - Questions?

Related sessions:

- MQ in Containers
  - Wednesday 2.30pm (Leopardwood)
- MQ Hybrid Cloud Architectures
  - Tuesday 8.30am (in here)
  - Wednesday 9.50am (in here)

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Trademark Statement

- IBM and the IBM logo are trademarks of International Business Machines Corporation, registered in many jurisdictions. Other marks may be trademarks or registered trademarks of their respective owners.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Red Hat Enterprise Linux is a registered trademark of Red Hat, Inc. in the United States and other countries.
- Ubuntu and Canonical are registered trademarks of Canonical Ltd.
- SUSE and SLES are registered trademarks of SUSE LLC in the United States and other countries
- Mac and OS X are trademarks of Apple Inc., registered in the U.S. and other countries
- Other company, product and service names may be trademarks, registered marks or service marks of their respective owners.
- References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.