# What's the cloud going to do to my MQ network?

*Chris Leonard*
*IBM UK*

*Session 17055*
*Tuesday 3rd March 2015*

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

# Agenda

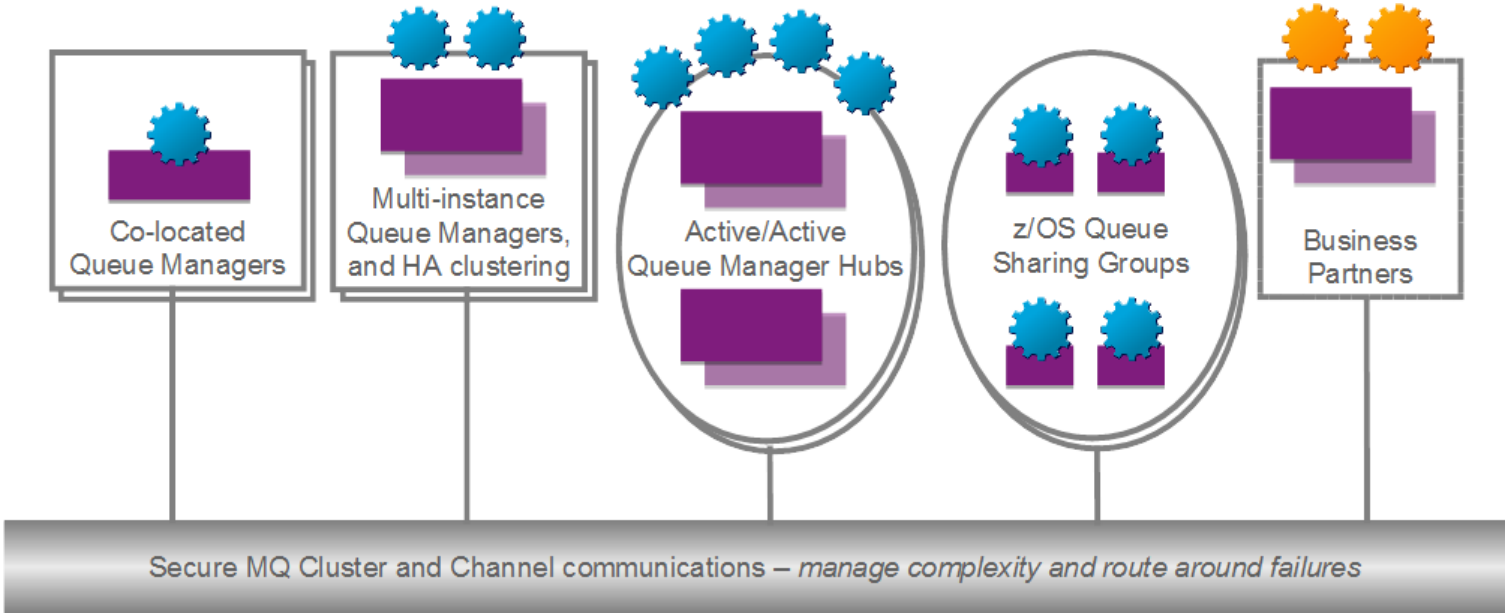
- Does my traditional MQ network make sense in the cloud era?
- Client user growing pains
- Rapid development
- Scalability
- Demo – Provisioning MQ for z/OS

# Does my traditional MQ infrastructure make sense in the cloud era?

The main challenges of the cloud era:

- Managing increasing numbers of concurrently connected users and applications, especially pervasive devices
- Rapid deployment to respond quickly to user demand and maximise return on investment
- Provide dynamic capacity to manage demand effectively with efficient management of idle resources

# Does my traditional MQ infrastructure make sense in the cloud era?



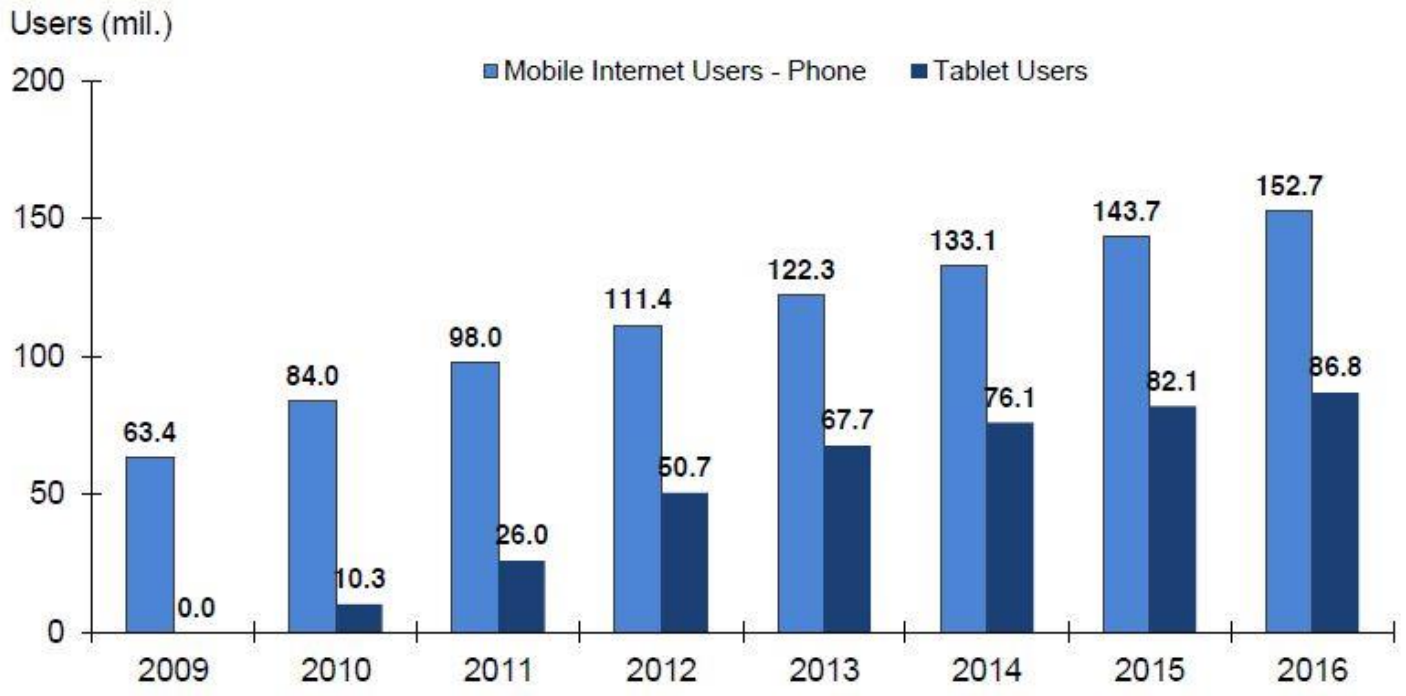

- Maybe.
  - MQ remains the market leader in enterprise messaging, designed to provide flexible, scalable solutions
    - Diverse platform and API coverage
    - Integration with other enterprise products such as application servers and databases

# Agenda

- Does my traditional MQ network make sense in the cloud era?
- Client user growing pains
- Rapid development
- Scalability
- Demo – Provisioning MQ for z/OS

# Client user growing pains

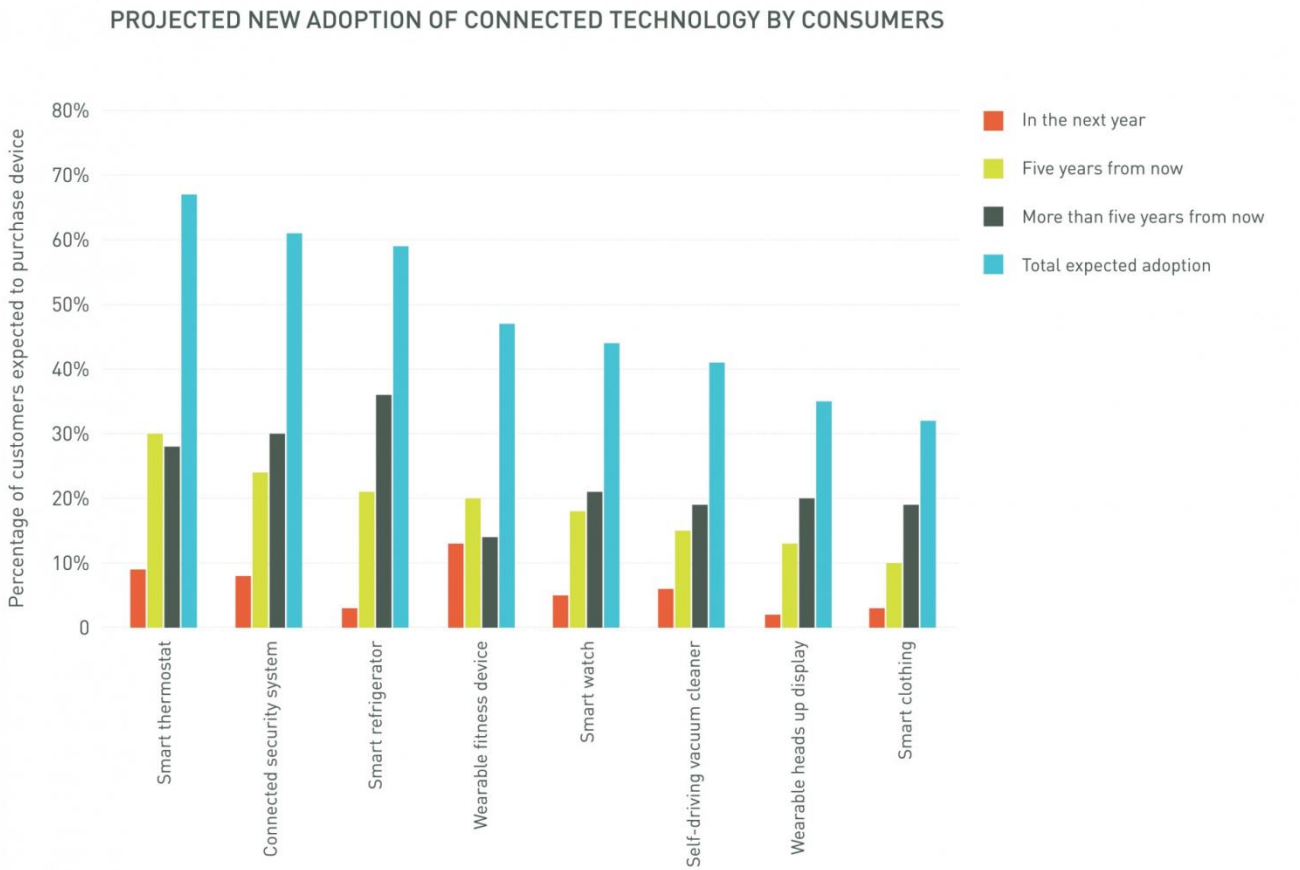
- Internet device adoption growth continues, for example in North America:




Source: Forrester Research Mobile Advertising Forecast, 2011-2016 (US)

# Client user growing pains

- It's not just personal devices:




Source: Acquity Group 2014 Internet of Things Study via Forbes

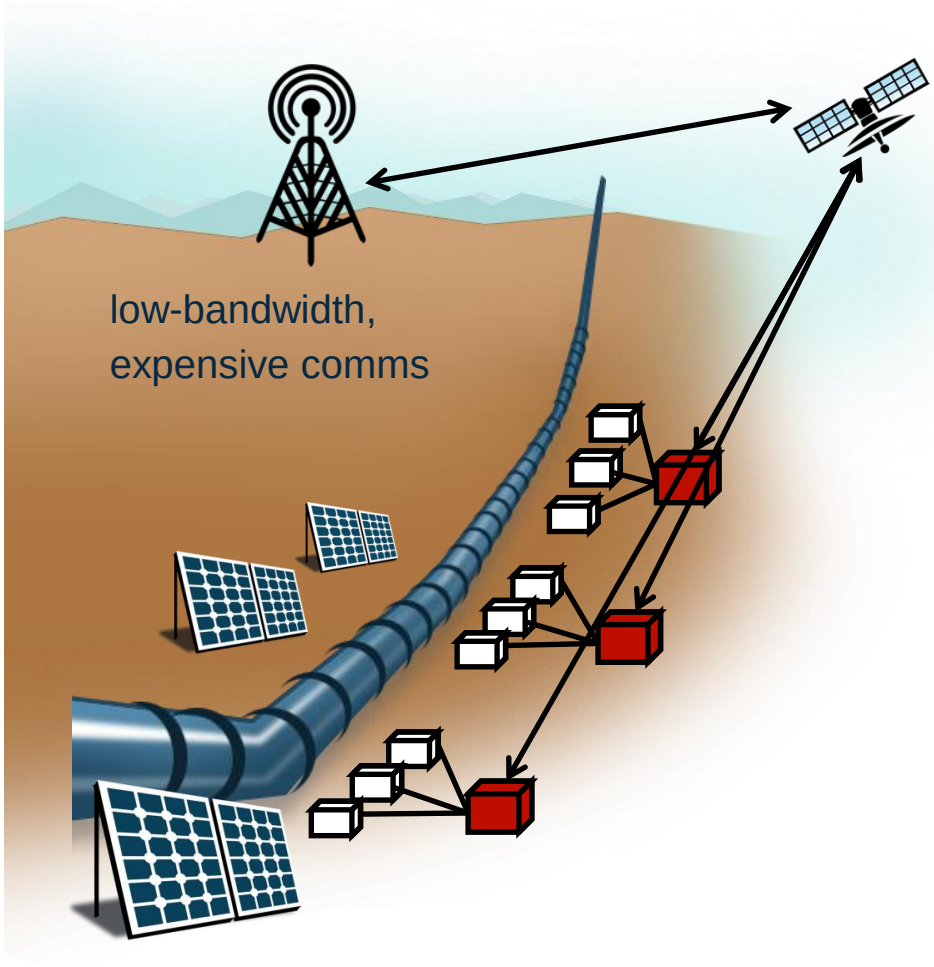
# Lightweight messaging with MQTT

- Handling interconnected devices is not an unforeseen problem!

- To save inventing a new protocol every time a new embedded device came along, a common protocol was needed.

- MQ Telemetry Transport (MQTT) is that protocol.
  - It traces its roots back to 1999, where Dr Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech) devised the protocol.

# Lightweight messaging with MQTT




Design goals of MQTT:

- Works over unreliable communication networks
- Minimal data overhead (low bandwidth)
- Capable of supporting large numbers of devices
- Simple to interface the data with the traditional IT world
- Simple to developers to write applications to use

# Lightweight messaging with MQTT


Key capabilities:

- Expects and caters for frequent network disruption – built for low bandwidth, high latency, unreliable, high cost networks
- Expects that client applications may have very limited resources available.
- Publish/subscribe messaging paradigm as required by the majority of SCADA and sensor applications.
- Provides traditional messaging qualities of service where the environment allows.
- OASIS standard for ease of adoption by device vendors and third-party client software.

# MQTT header

- MQTT header can be as little as 2 bytes:

| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Byte 1 | Message Type | | | | DUP flag | QoS Level | | RETAIN |
| Byte 2 | Remaining Length (at least one byte) | | | | | | | |

- Contrast with WebSphere MQ MQMD header structure:

```
struct tag MQMD {      MQCHAR4   StrucId;           // Structure identifier
                       MQLONG    Version;           // Structure version number
                       MQLONG    Report;            // Options for report messages
                       MQLONG    MsgType;           // Message type
                       MQLONG    Expiry;            // Message lifetime
                       MQLONG    Feedback;          // Feedback or reason code
                       MQLONG    Encoding;          // Numeric encoding of message data
                       MQLONG    CodedCharSetId;    // Character set identifier of message data
                       MQCHAR8   Format;            // Format name of message data
                       MQLONG    Priority;          // Message priority
                       MQLONG    Persistence;       // Message persistence
                       MQBYTE24  MsgId;             // Message identifier
                       MQBYTE24  CorrelId;          // Correlation identifier
                       MQLONG    BackoutCount;      // Backout counter
                       MQCHAR48  ReplyToQ;          // Name of reply queue
                       MQCHAR48  ReplyToQMgr;       // Name of reply queue manager
                       MQCHAR12  UserIdentifier;    // User identifier
                       MQBYTE32  AccountingToken;   // Accounting token
                       MQCHAR32  ApplIdentityData;  // Application data relating to identity
                       MQLONG    PutApplType;       // Type of application that put the message
                       MQCHAR28  PutApplName;       // Name of application that put the message
                       MQCHAR8   PutDate;           // Date when message was put
                       MQCHAR8   PutTime;           // Time when message was put
                       MQCHAR4   ApplOriginData;    // Application data relating to origin
                       MQBYTE24  GroupId;           // Group identifier
                       MQLONG    MsgSeqNumber;      // Sequence number of logical message within group
                       MQLONG    Offset;            // Offset of data in physical message from start of logical message
                       MQLONG    MsgFlags;          // Message flags
                       MQLONG    OriginalLength;    // Length of original message    }
```

# MQTT header

**Message Types:**

| | |
|---|---|
| CONNECT | CONNACK |
| PUBLISH | PUBACK |
| PUBREC | PUBREL |
| PUBCOMP | SUBSCRIBE |
| SUBACK | UNSUBSCRIBE |
| UNSUBACK | PINGREQ |
| PINGRESP | DISCONNECT |

**DUP flag:**
Used to indicate a redelivery message for one of the message types:
PUBLISH, PUBREL, SUBSCRIBE, UNSUBSCRIBE

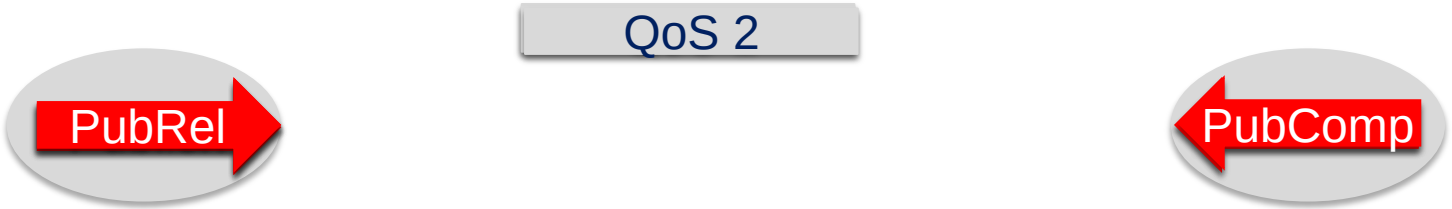
| bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Byte 1** | Message Type | | | | DUP flag | QoS Level | | RETAIN |
| **Byte 2** | Remaining Length (at least one byte) | | | | | | | |
| ***Byte 3*** | *Remaining Length (msg up to 16KB)* | | | | | | | |
| ***Byte 4*** | *Remaining Length (msg up to 2MB)* | | | | | | | |
| ***Byte 5*** | *Remaining Length (msg up to 256MB)* | | | | | | | |

Variable **length** message (127 bytes maximum for the single byte length field), up to a maximum of 256MB for 4 length byte fields.

Indicates if a message should be **retained**, to be sent to new subscribers.

# MQTT qualities of service

- QoS 0: At most once delivery (non-persistent)
  - No retry semantics are defined in the protocol.
  - The message arrives either once or not at all.
- QoS 1: At least once delivery (persistent, duplicate messages possible)
  - Client sends message with Message ID in the message header
  - Server acknowledges with a PUBACK control message
  - Message resent with a DUP bit set If the PUBACK message is not seen
- QoS 2: Exactly once delivery (persistent)
  - Uses additional flows to ensure that message is not duplicated
  - Server acknowledges with a PUBREC control message
  - Client releases message with a PUBREL control message
  - Server acknowledges completion with a PUBCOMP control message

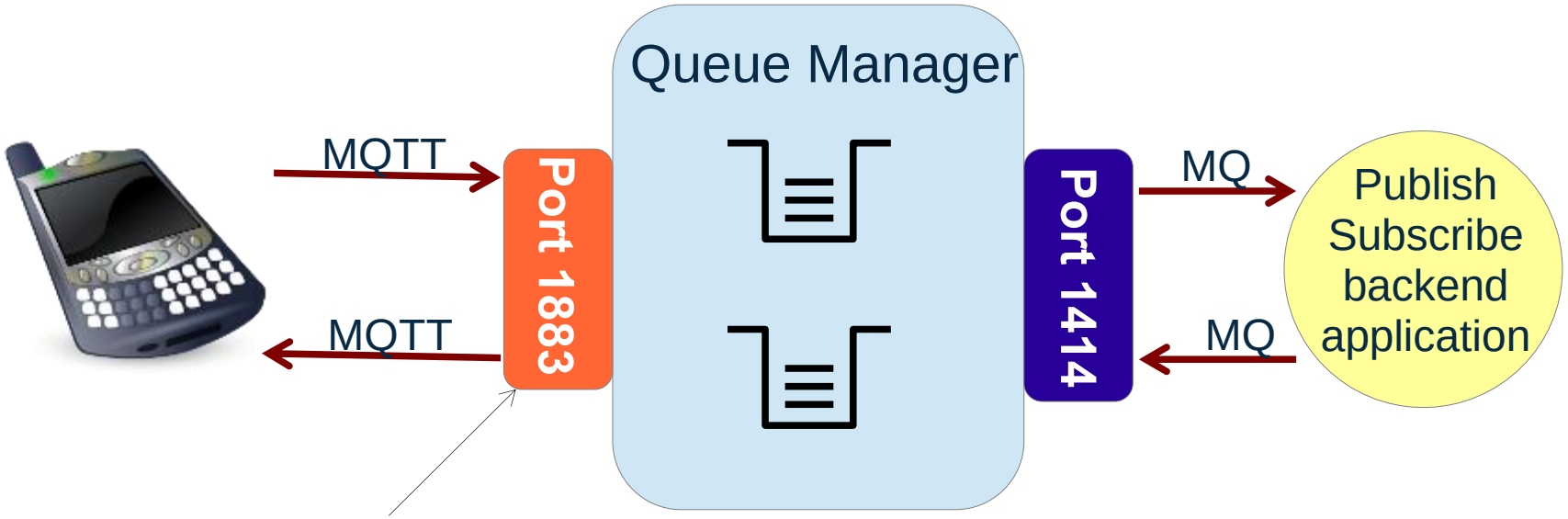


# MQ Telemetry service (MQXR)

- Supplied as a component of MQ V7.1 and later releases on distributed platforms, under the component name “MQ Extended Reach” (or MQXR).

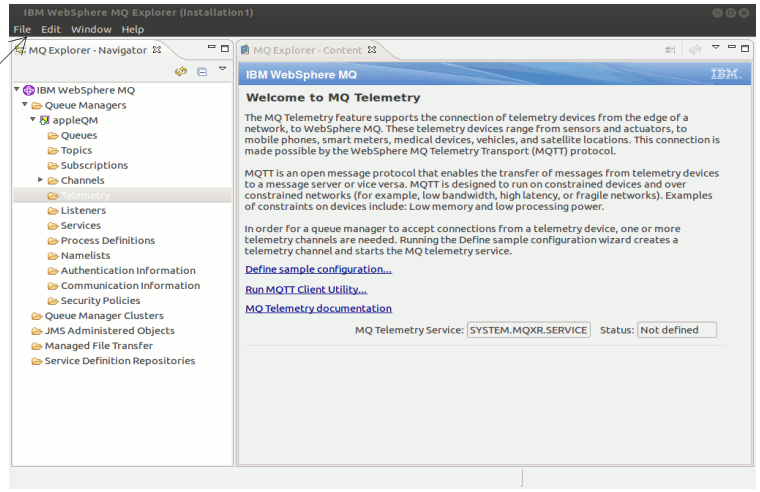
- MQXR brings MQTT protocol functionality to MQ
  - Highly scalable : tested with 200,000+ clients
  - Security : SSL channels, JAAS authentication, WMQ OAM
  - Ships with reference Java and C clients
  - Small footprint clients
  - Supports other APIs and implementations of MQTT clients available via 3rd parties

# MQ Telemetry Service (MQXR)




WebSphere MQ MQTT Listener
IANA registered ports:
1883, 8883 for MQTT over SSH

Use WebSphere MQ Explorer to administer the WebSphere MQ Telemetry service – define Channels, start and stop the MQTT service.
Alternatively, it can be configured through 'runmqsc' commands.



# MQTT through Javascript

As of MQ 7.5.0.1, the MQXR component has support for MQTT v3.1 protocol over WebSockets.

This enables the use of MQTT through a WebSocket supporting web browser, meaning that MQTT can be used without preinstalling any software on a browser equipped device.

```
<script type="text/javascript" src="mqttws31.js"></script>
<script type="text/javascript">
  var clientId = "MyUniqueClientID";
  var client;

  function publishMessage() {
    client = new Messaging.Client(location.hostname, Number(1883), clientId);
    client.onConnectionLost = onConnectionLost;
    client.onMessageArrived = onMessageArrived;
    client.connect({onSuccess:onConnect});
  }

  function onConnect() {
    // Once a connection has been made, make a subscription and send a message.
    console.log("onConnect");
    client.subscribe("/TopicLocation");
    message = new Messaging.Message("My publish text!");
    message.destinationName = "/TopicLocation";
    client.send(message);
  }

</script>

<button type="button" onclick="publishMessage()" name="Connect">Publish a Message</button>
```

Reference the WMQ supplied MQTT javascript file

Connect to the MQTT server, and register callback functions

Subscribe to the Topic, and publish a message.

Invoke the Javascript function from HTML

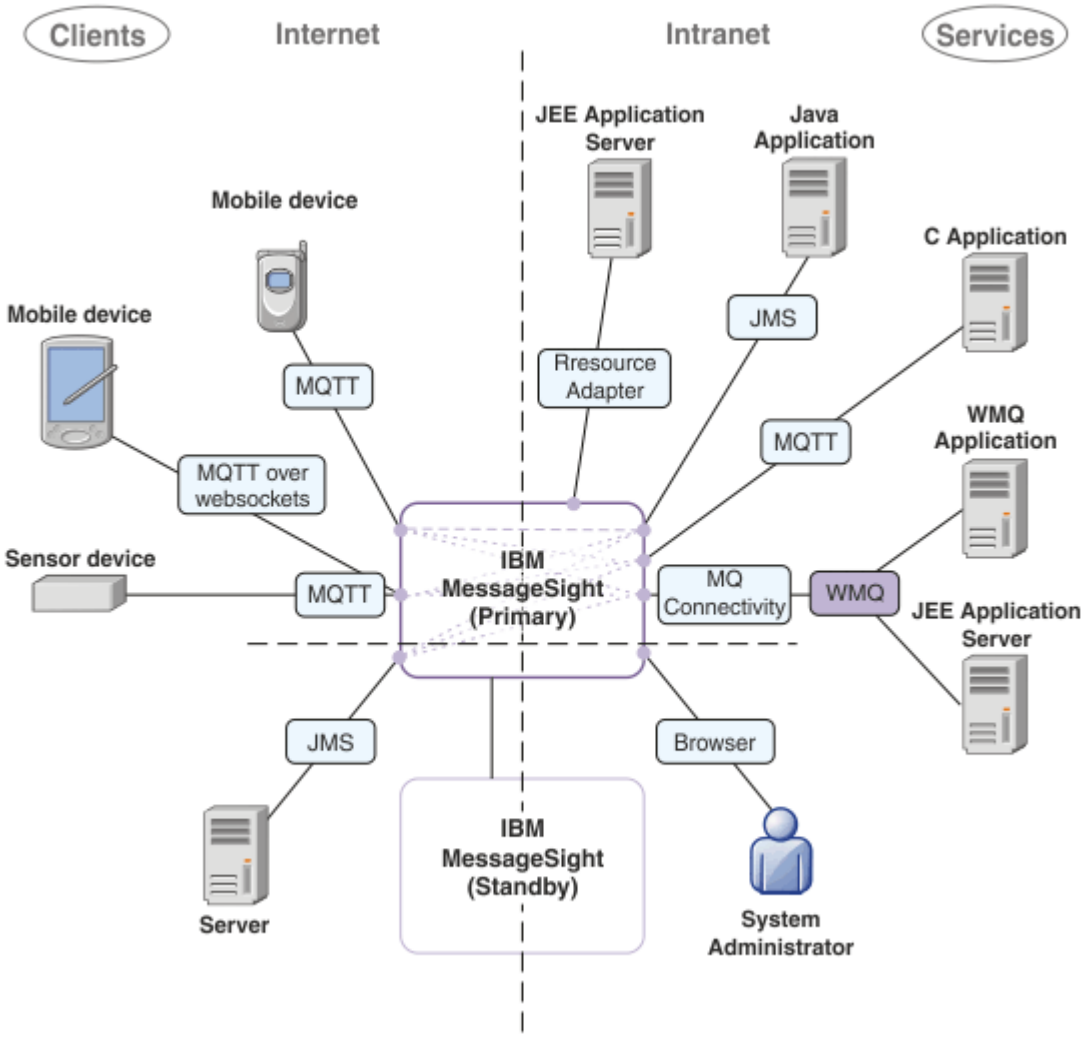
Write callback functions here

# IBM MessageSight – big connectivity in a box


- A secure messaging server appliance optimised to meet the demands of massive scale messaging of machine-to-machine and mobile use cases.

- How massive? One appliance can achieve:
  - 1 million concurrent connections
  - 13 million non-persistent msg/sec
  - 400K persistent msg/sec

# IBM MessageSight and MQ



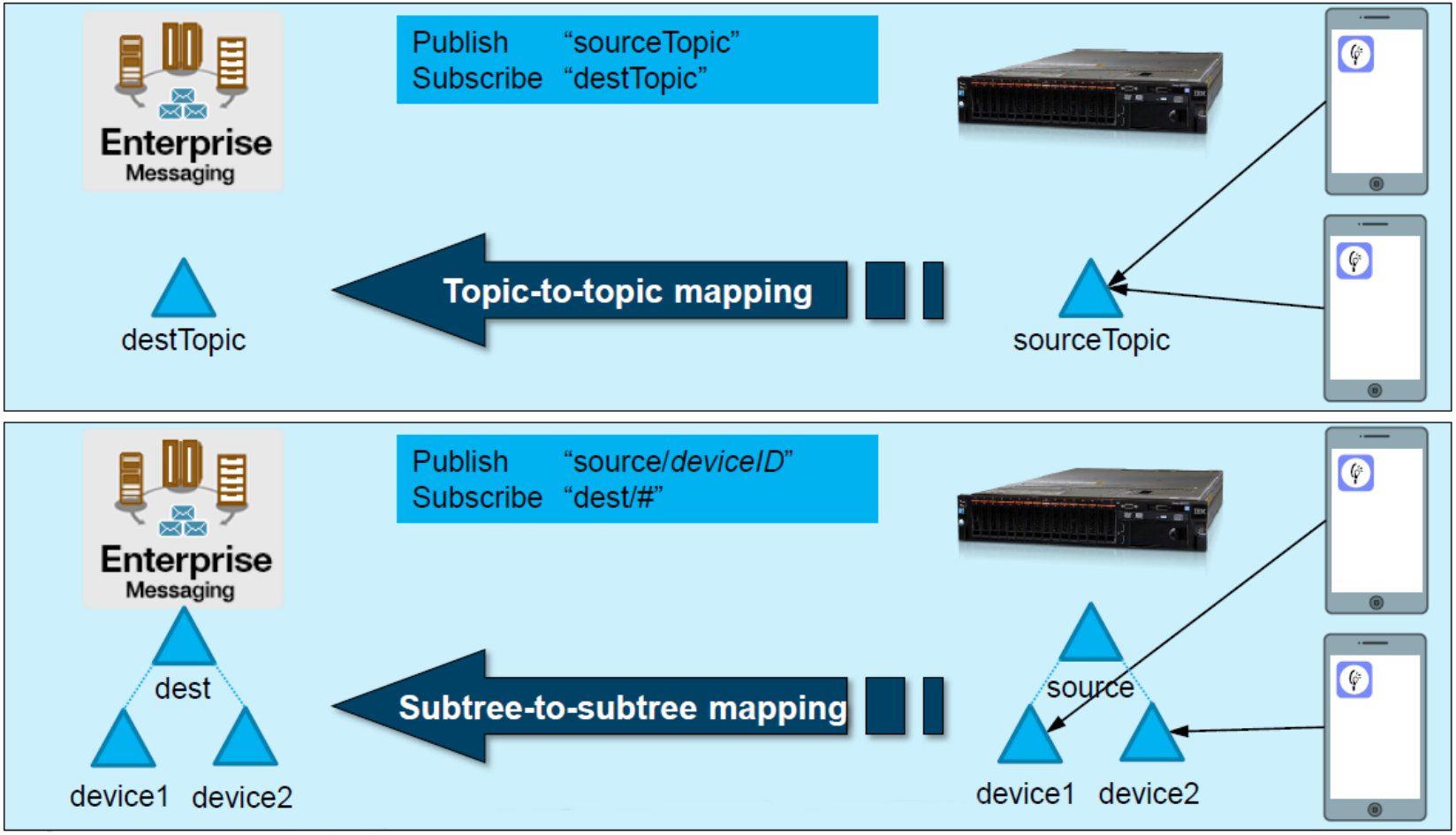
# IBM MessageSight and MQ

- Built-in MQ connectivity
- Securely extends existing enterprise messaging infrastructures
- Accelerate massive fan-out message delivery to huge numbers of devices
- Reliable bi-directional messaging enabling intelligent decisions based on real-time events

# IBM MessageSight and MQ

- Very simple to set up
- Queue manager connection
  - Defines how to connect to a queue manager
    - Queue manager name
    - Connection name – host name, port
    - Channel name – server-connection channel
    - SSL cipher specification (optional)
- Destination mapping rule
  - Defines source and target of messages
    - Rule type – topic-to-topic, topic-to-queue, …
    - Queue manager connection – one or more
    - Maximum messages to buffer for transmission
    - Retained messages?

# IBM MessageSight and MQ



# MQTT links

- MQTT homepage:
  - http://mqtt.org
- MQTT Specification
  - http://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html
- WebSphere MQ and MQ Telemetry
  - http://www-01.ibm.com/software/integration/wmq/
- Mobile Messaging & M2M Client Pack
  - http://www.ibm.com/developerworks/mydeveloperworks/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/download
- MQTT: the Smarter Planet Protocol
  - http://andypiper.co.uk/2010/08/05/mqtt-the-smarter-planet-protocol/
- Lotus Expeditor (MQTT microbroker)
  - http://www.ibm.com/software/lotus/products/expeditor/
- IBM MessageSight
  - www.ibm.com/software/products/gb/en/messagesight/

# Agenda

- Does my traditional MQ network make sense in the cloud era?
- Client user growing pains
- Rapid development
- Scalability
- Demo – Provisioning MQ for z/OS

# Rapid development of messaging applications

Enterprise Messaging

Deliver Messaging Backbone for Enterprise

Focus on traditional MQ values, rock-solid enterprise-class service, ease-of-operation, breadth of platform coverage, availability, z/OS exploitation

Application Messaging

Enable Developers to build more scalable, responsive applications

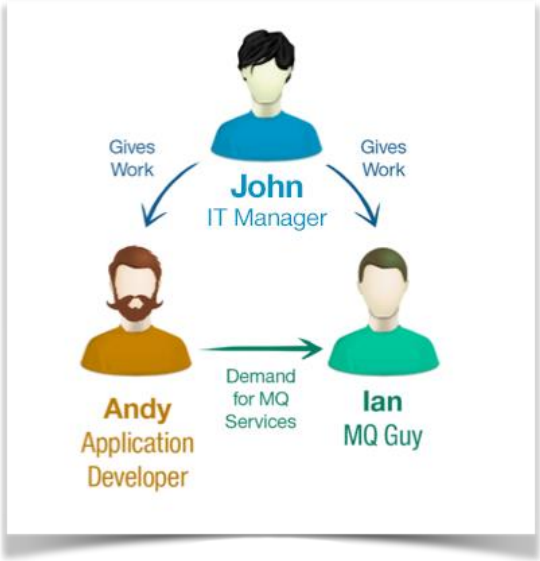
Focus on app use cases, breadth of languages, ease-of-deployment, micro services, integration with developer frameworks

# Meet Andy – Messaging application developer

- Wants to product applications that can be field tested in the minimum time possible
- Discovers technologies that are prevalent in his communities
- Uses the best tool for the job
- Intolerant of process / imposed technologies that do not obviously and immediately benefit his application

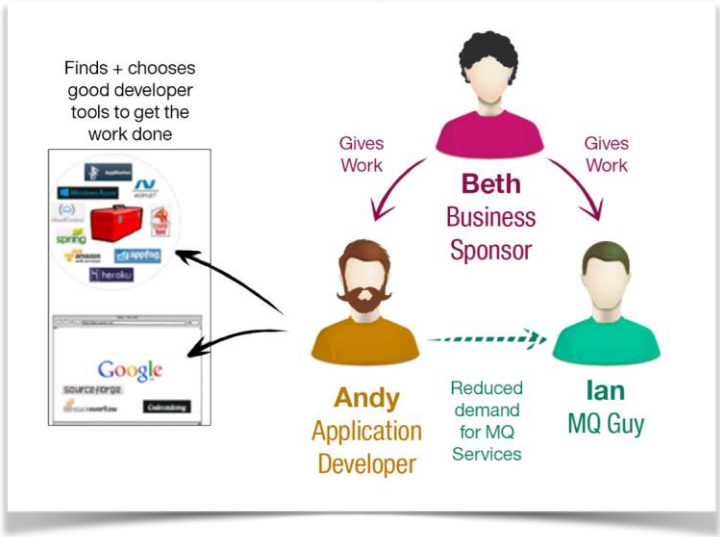
# Organisational changes

From:
Centrally planned IT Architecture



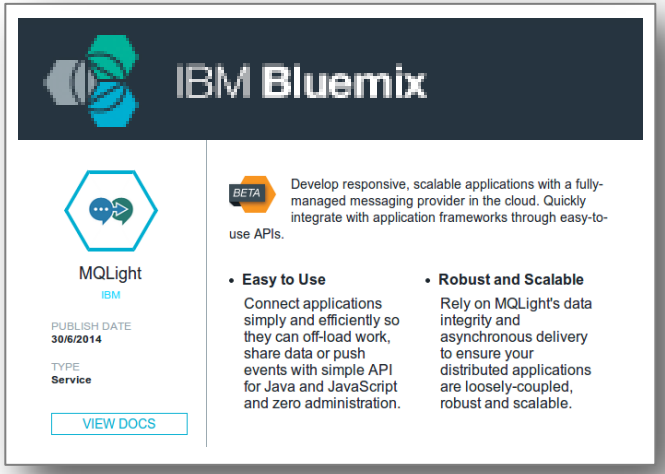

Centrally controlled common standards
Planned projects delivering core systems
Focused on skills and investment reuse

Emerges:
Business sponsor driven
Developer led architecture



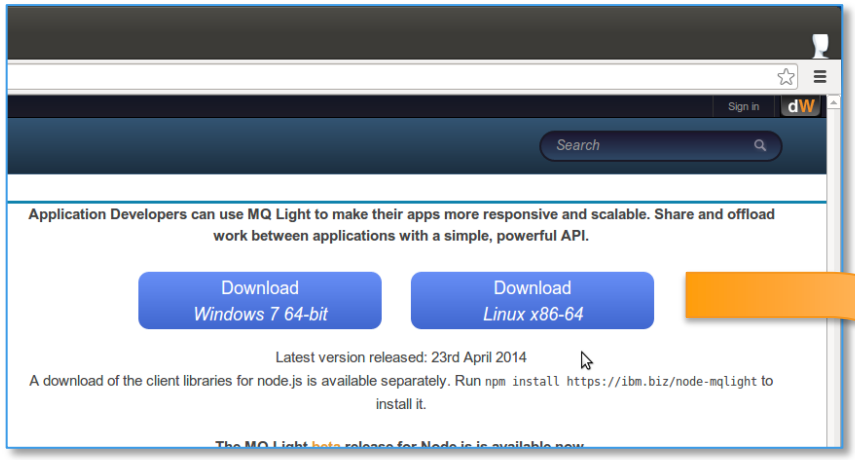

Developers select tools to get the job done
Focused on trying new apps and concepts in the market

# MQ Light : Software and Cloud

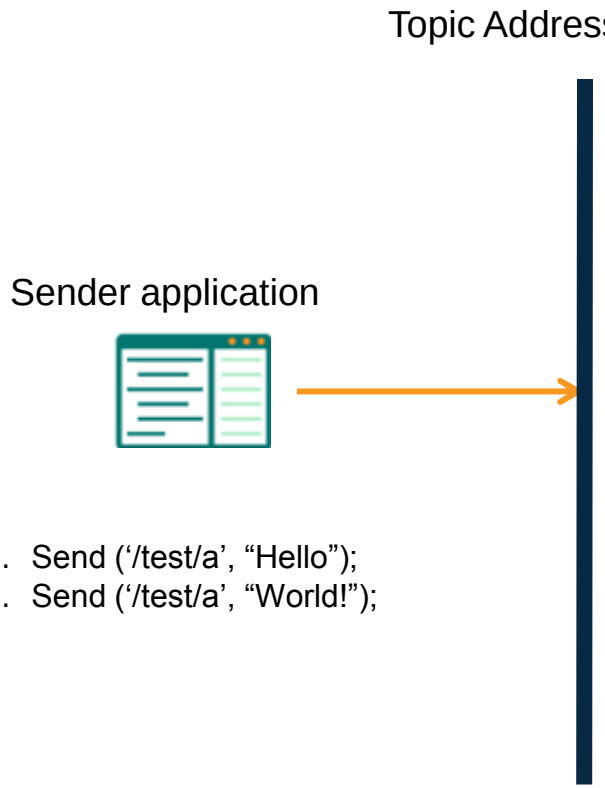

- Messaging that application developers will love to use, helping them make responsive applications that scale easily
- 3 ways to get it:
  - MQ Light software download
  - Bluemix service
  - *Statement of Direction for support in MQ Version 8.*
- Open APIs crafted to feel natural in a growing range of popular languages
- Tooling that makes modular app development easy

# Software download - First five minute experience



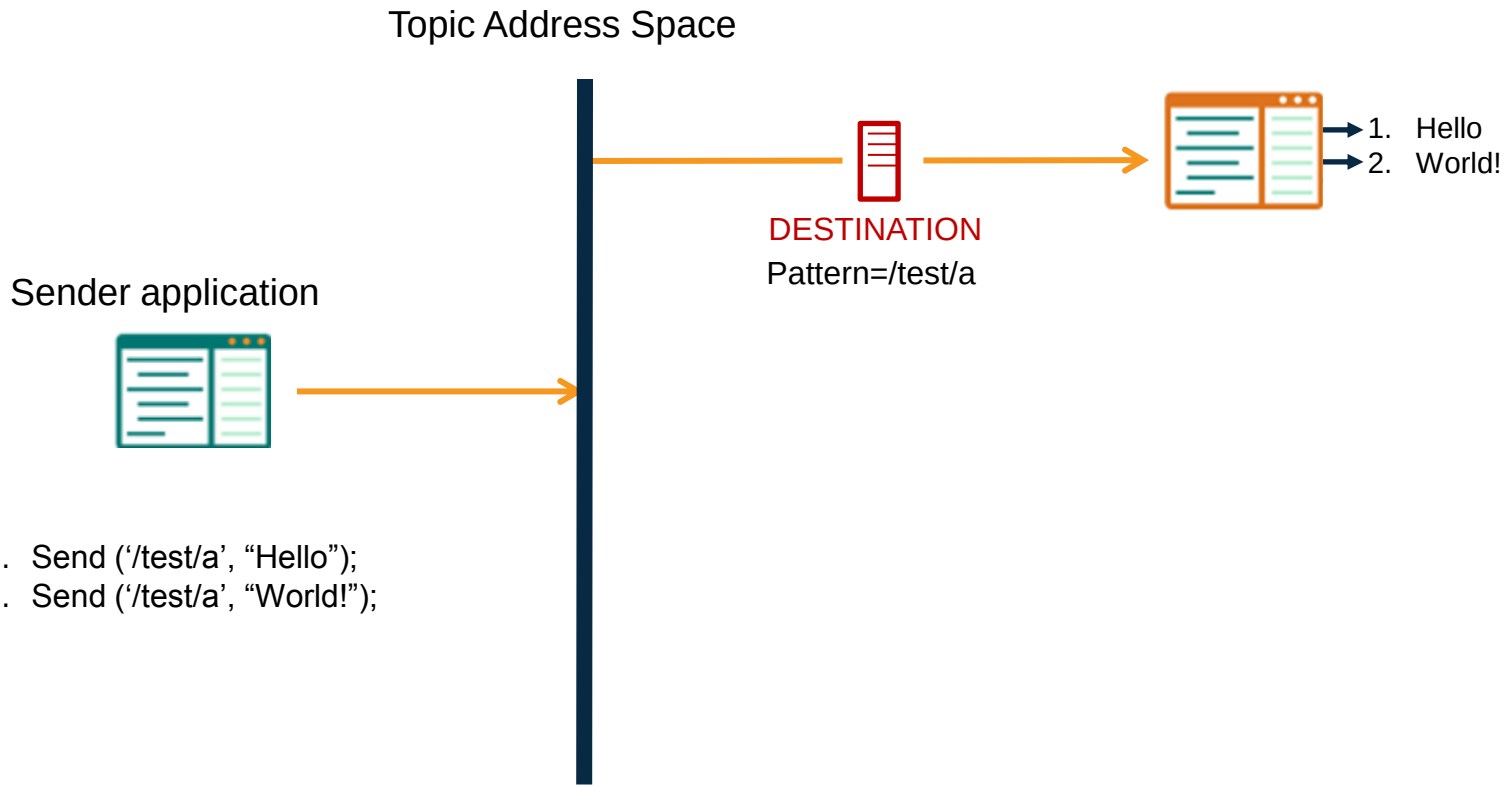

- Download and get coding within 5 minutes
  - Linux-x86-64, Windows7 64 bit, Mac OSX
  - Unzip install
  - Unlimited time developer license (unsupported).
- No administration; just code and go
- Node API client libraries installed using npm package manager
- Tutorials and examples in multiple languages, relevant to actual use

# MQ Light Messaging Model – Send Messages



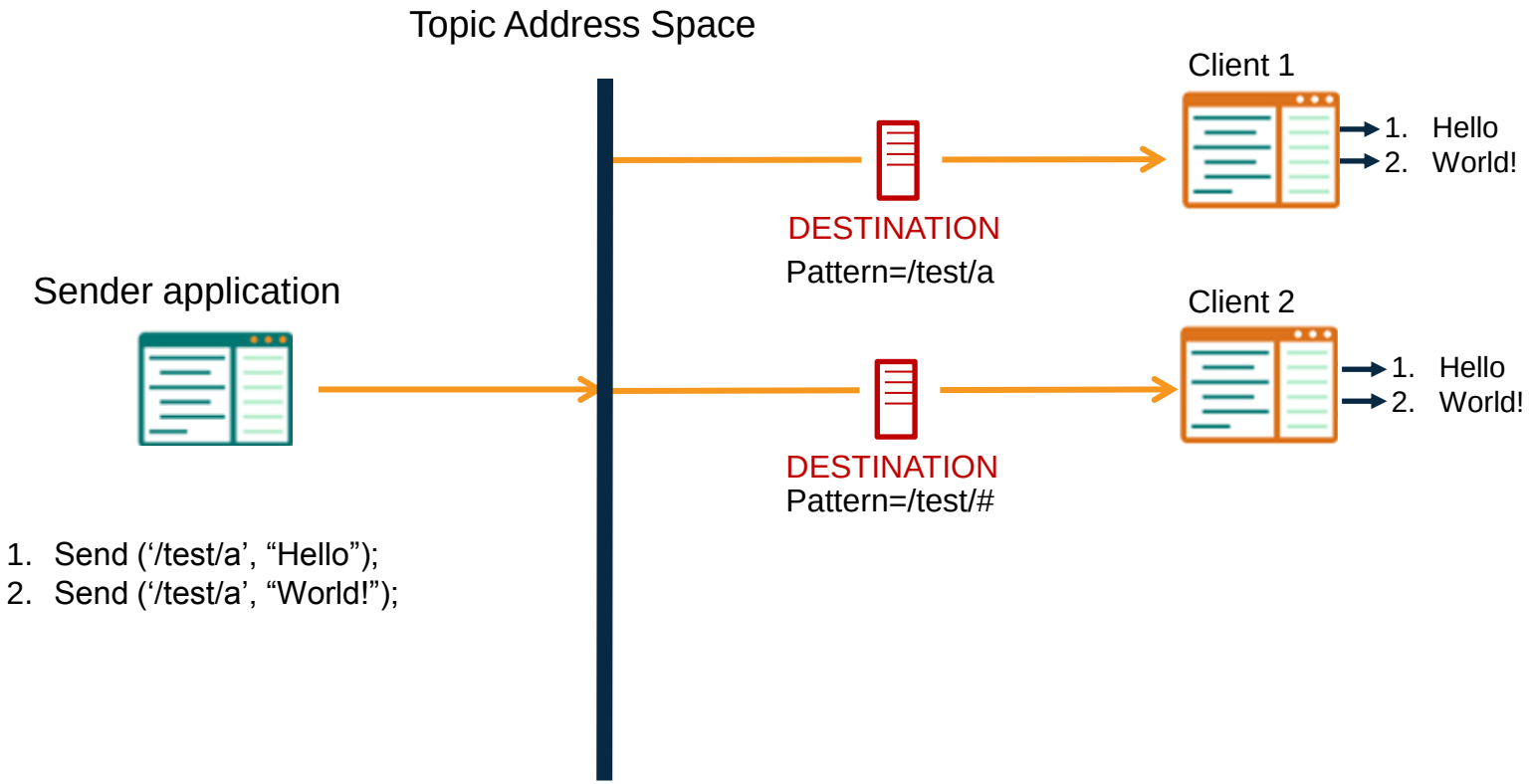

- Applications send messages to a topic.
- A topic is an address in the topic space
  - Either flat or arranged hierarchically.

# MQ Light Messaging Model – Simple Receive



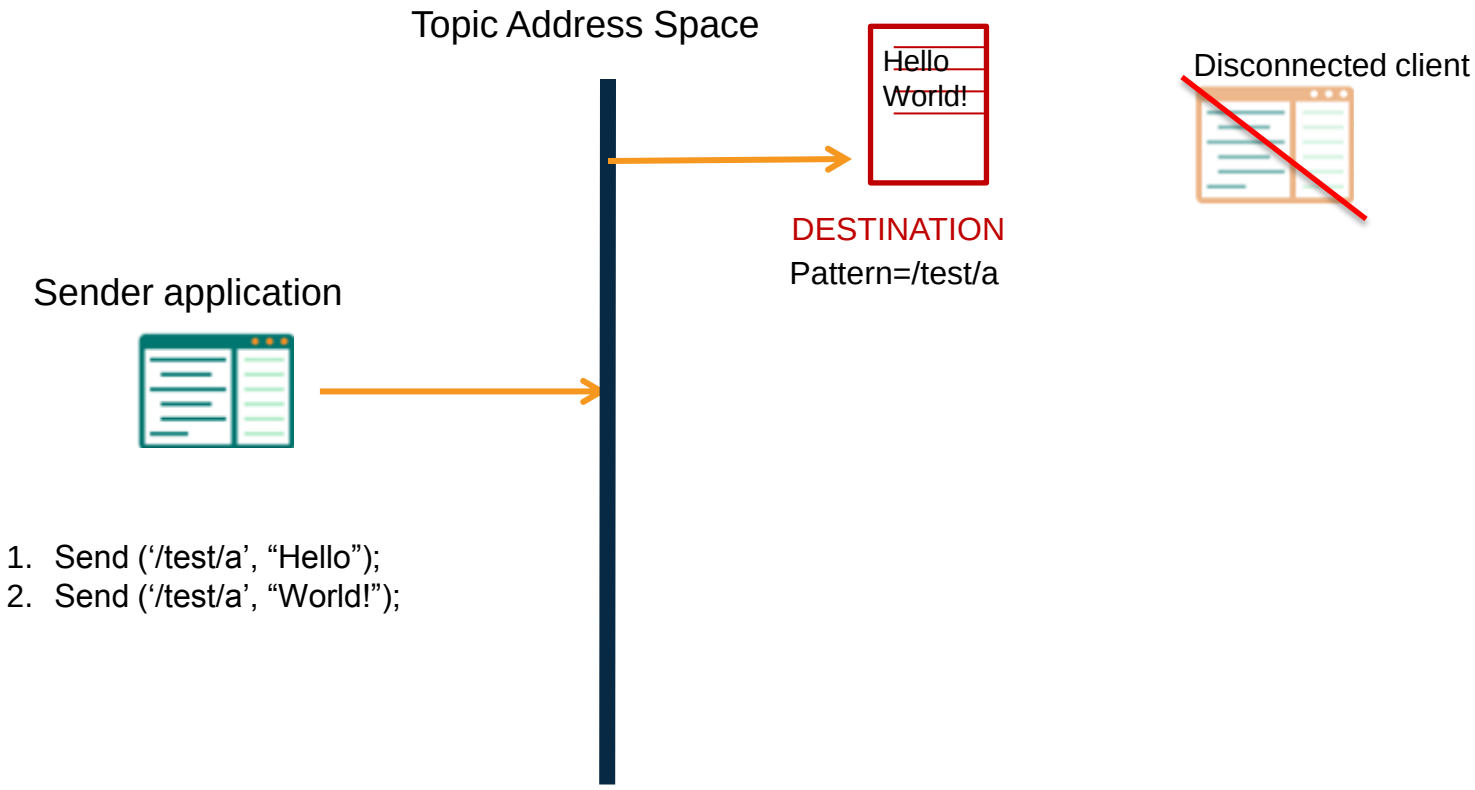

- Applications receive messages by creating a destination with a pattern which matches the topics they are interested in.
- Pattern matching scheme based on MQ.

# MQ Light Messaging Model – Pub/Sub



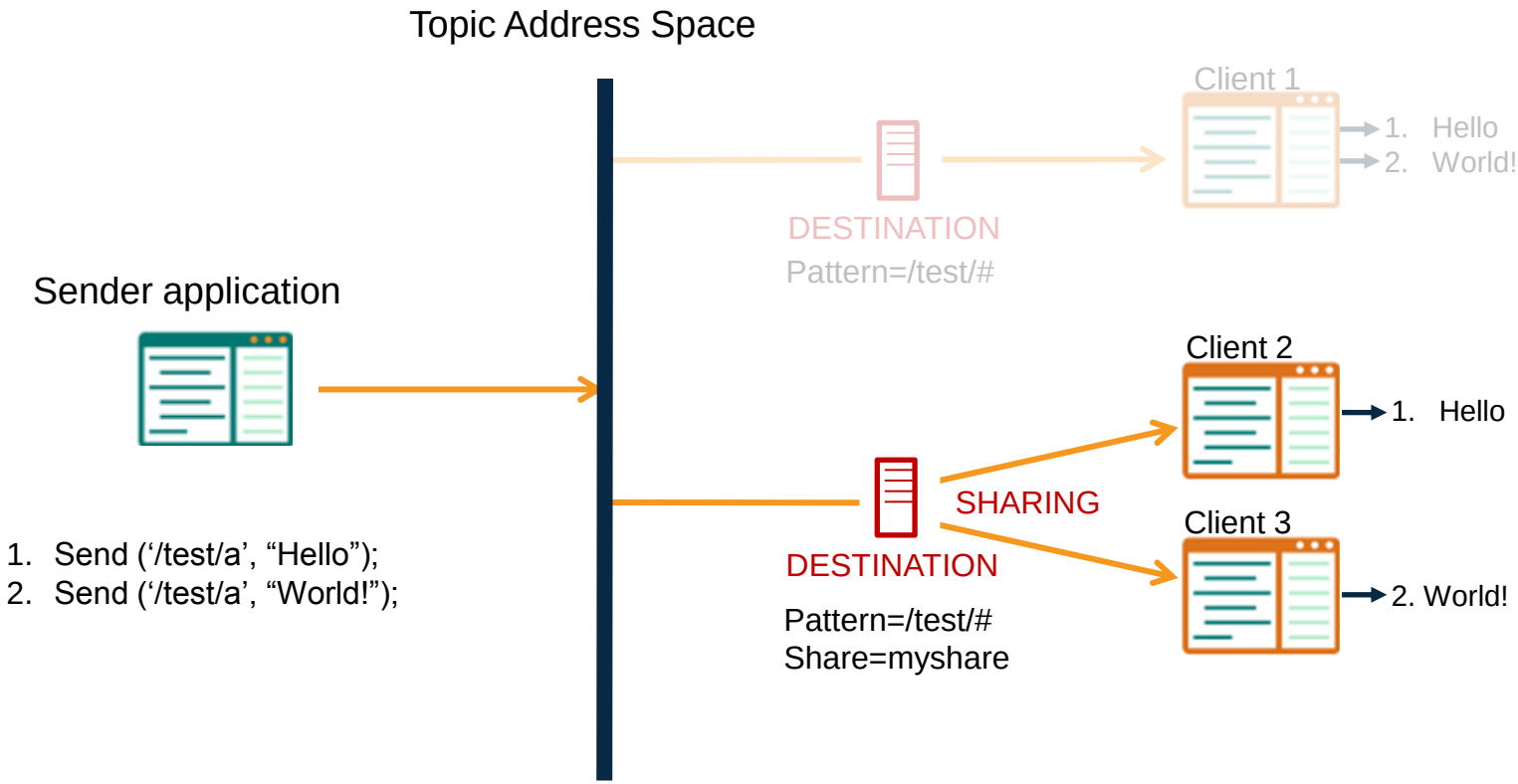

- Multiple destinations can be created which match the same topic
  - Pub/Sub style.

# MQ Light Messaging Model – Persistent destinations



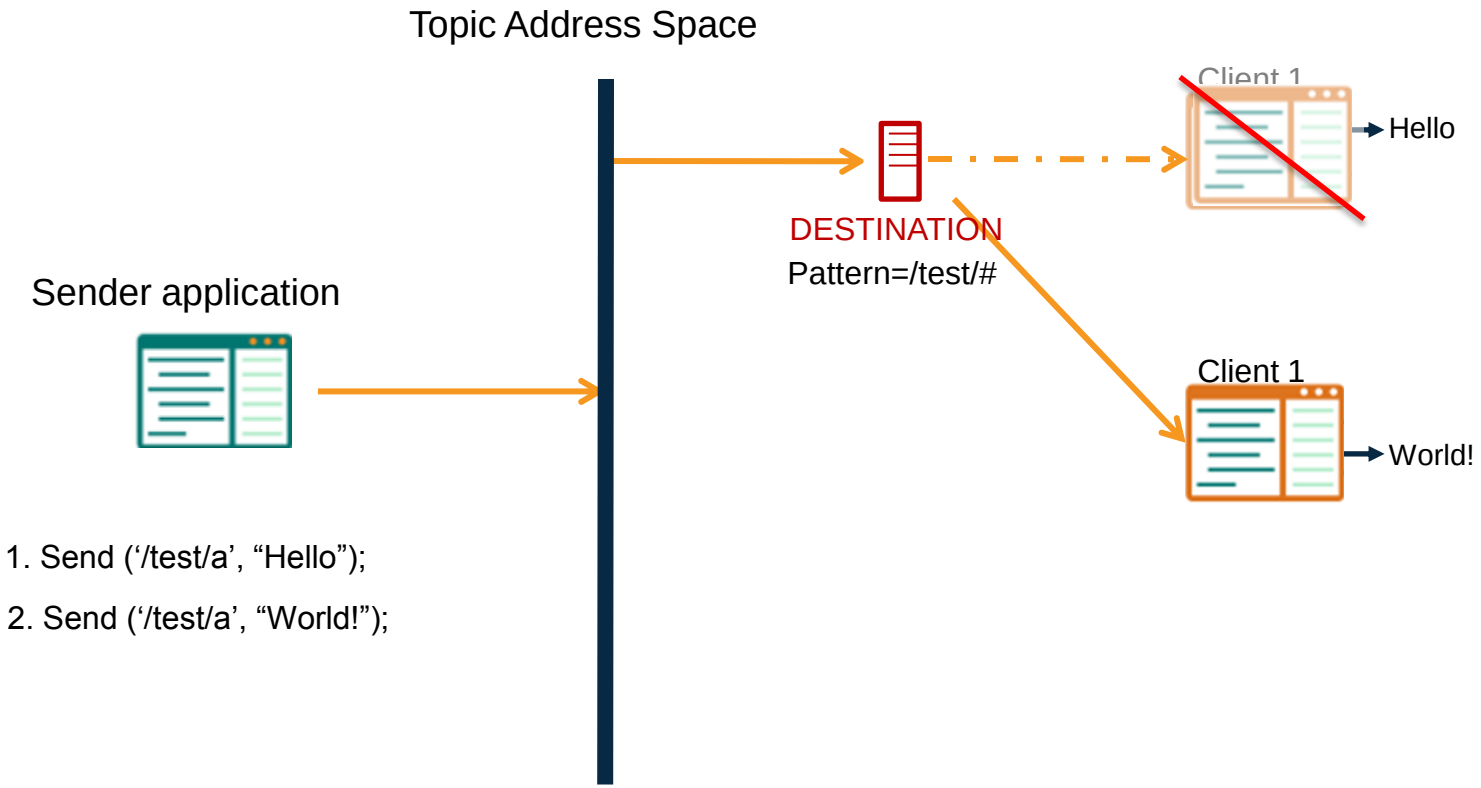

- Destinations persist for a defined "time to live" after receiver detaches.

# MQ Light Messaging Model – Sharing




- Clients attaching to the same topic pattern and share name attach to the same shared destination.

# MQ Light Messaging Model – Client takeover




- Applications connect to MQ Light service specifying (optional) client ID.
- Re-using the same client ID takes over the original connection.
  - Ideal for worker takeover in the cloud.

# MQ Light Messaging Model

- Messages
  - Payload is either Text or Binary.
  - Content-type is used by clients to transfer JSON
  - Per message time to live.

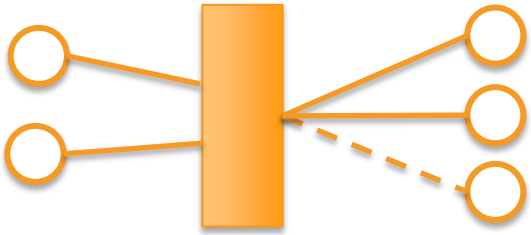
- Message delivery model
  - At most once delivery (QoS 0)
  - At least once delivery (QoS 1)
  - Acknowledge & Reject messages
  - Control over the number of unacknowledged messages delivered. (link credit)

# Use Cases

## Worker Offload

Intensive work offloaded and distributed amongst worker processes to be performed asynchronously

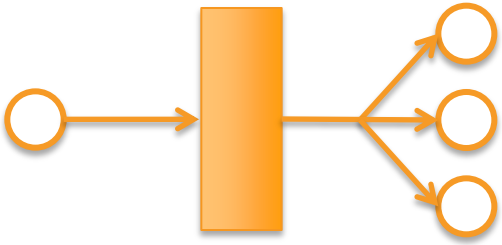
- Processing images or videos
- Performing text analytics



## Event Driven

Take one or more actions when something interesting happens

- Email logs and update dashboards when build finishes
- Upload videos once finished transcoding

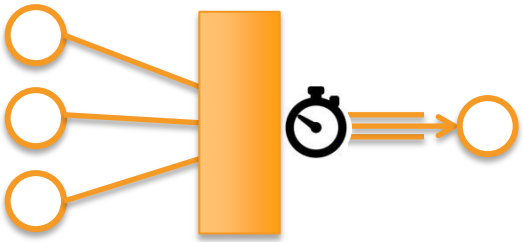


# Use Cases

## Delayed Processing

Schedule a task to happen at a specific point in time

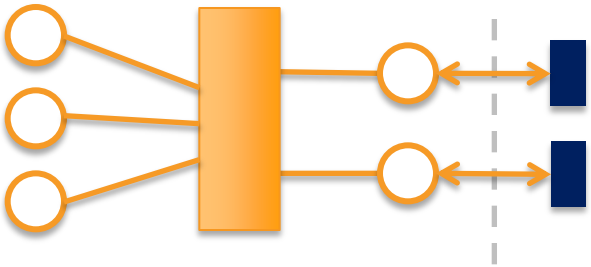
- Run in detailed reports when app use is low
- Generate end of day summary



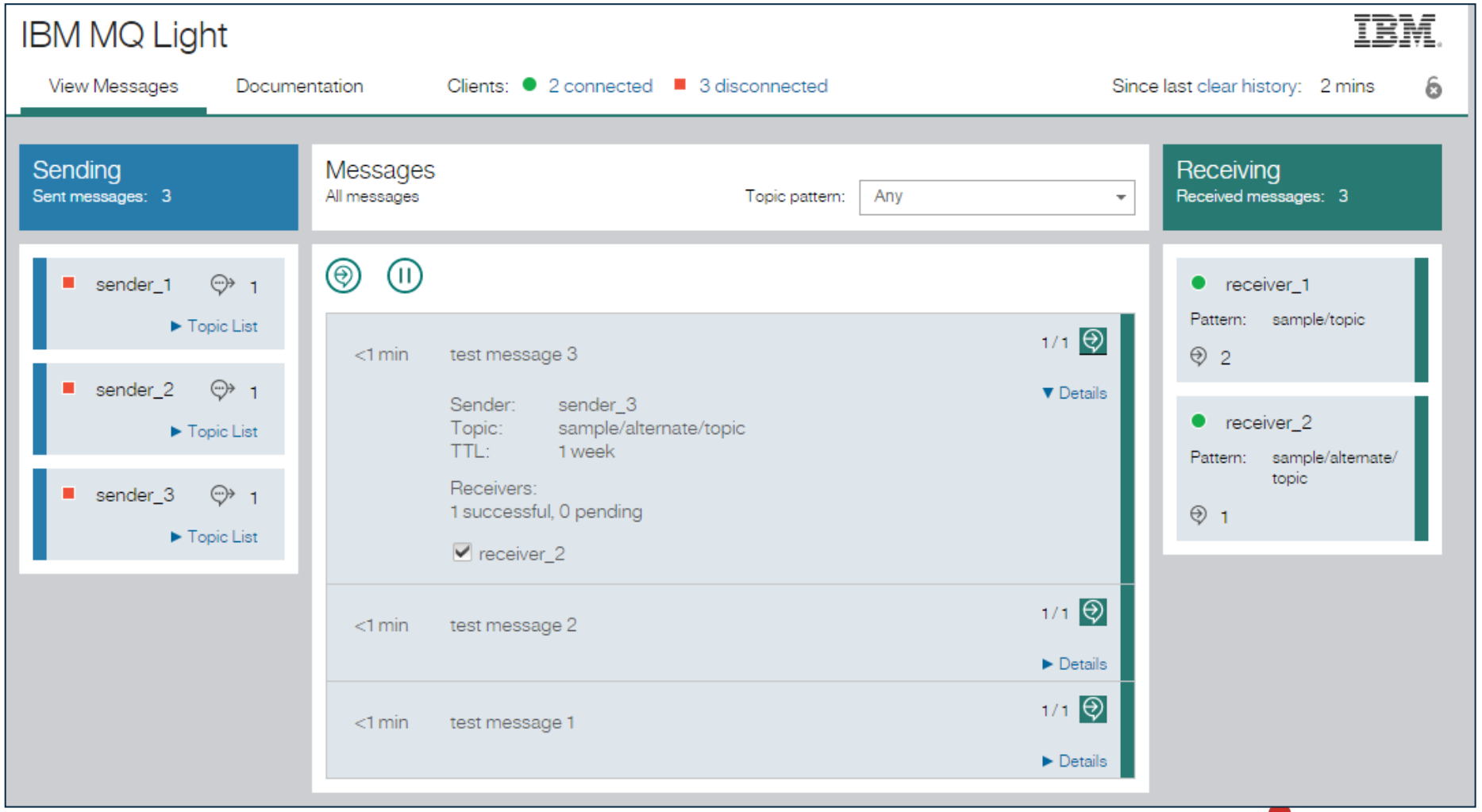
## 3rd Party Integration

Ensure applications remain responsive even when 3rd party system are not available or responding fast enough

- Updating existing CRM system
- Booking appointment



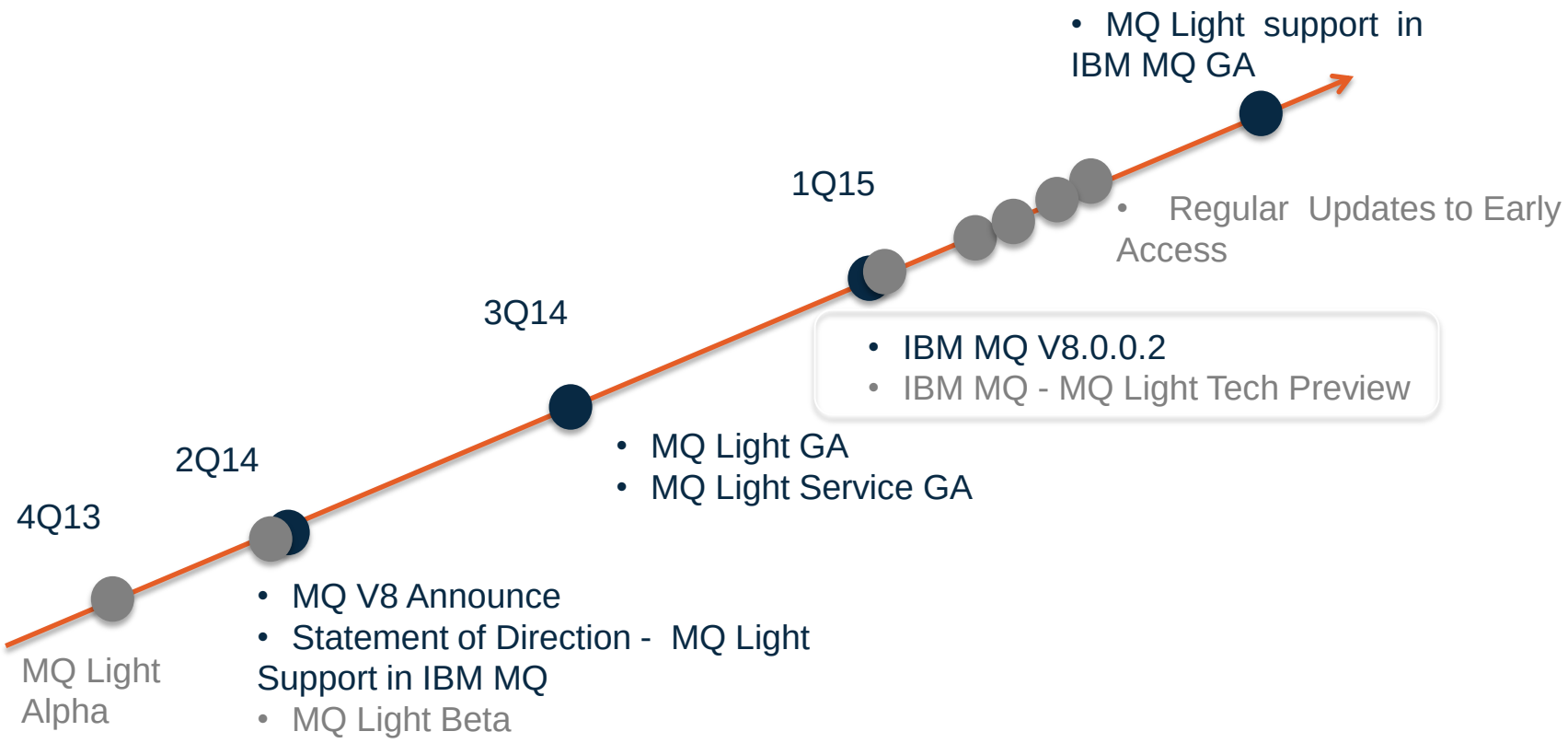
# MQ Light - WebUI



# Agenda

- Does my traditional MQ network make sense in the cloud era?
- Client user growing pains
- Rapid development
- Scalability
- Demo – Provisioning MQ for z/OS

# MQ Light Support in IBM MQ

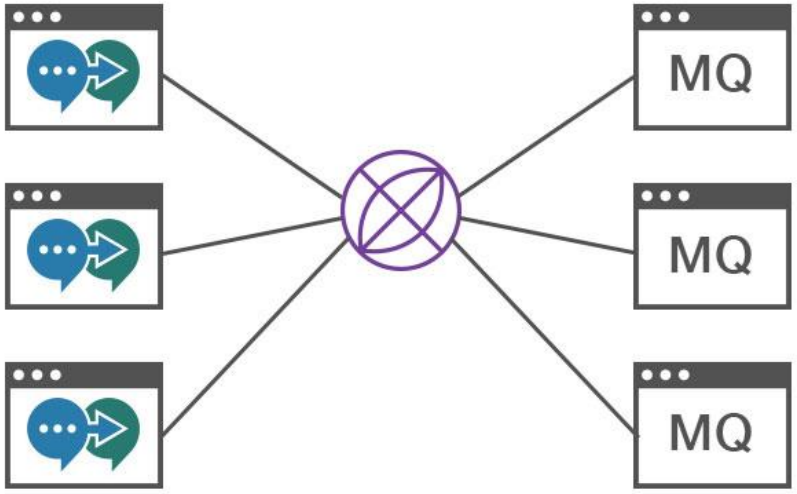


# IBM MQ – MQ Light Tech Preview

- Platforms
  - Windows 64 Bit
  - Linux x86_64
- Beta Installation
  - Prereq is IBM MQ v8.0.0.2
  - Add Tech Preview install media
    - Linux – RPM which is installed along side the other MQ RPMs
    - Windows – Zip which is manually extracted to an MQ installation

# New AMQP channel type

- Adds a channel type of “AMQP”
- Support a subset of the AMQP 1.0 Oasis specification
- Interoperable with MQ FAP and MQTT applications (see later slides for details)



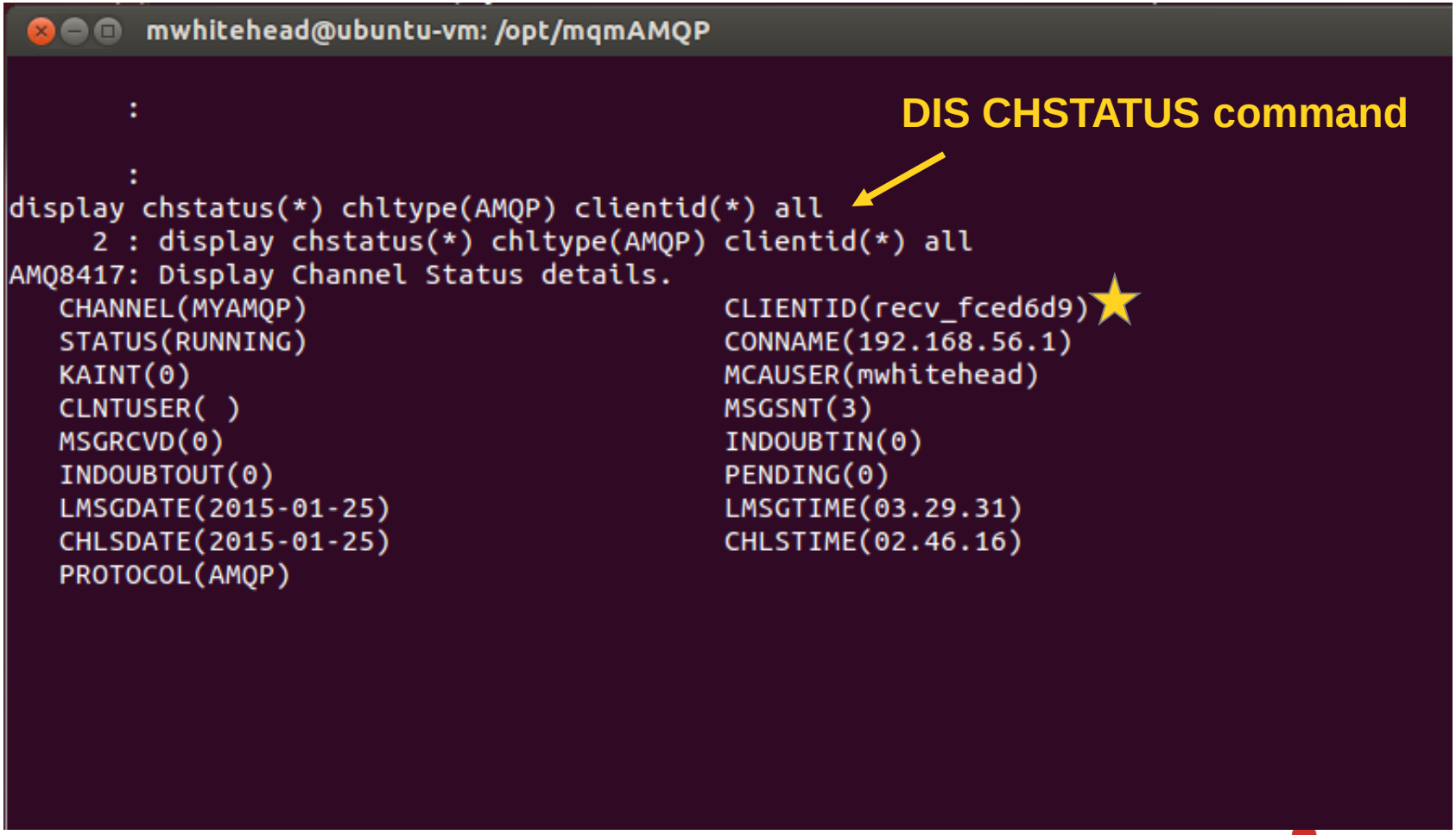
# AMQP channels

```
mwhitehead@ubuntu-vm: /opt/mqmAMQP
      :
      :
dis conn(*) type(conn) where (channel eq myamqp) all
    10 : dis conn(*) type(conn) where (channel eq myamqp) all
AMQ8276: Display Connection details.
   CONN(9658C45405250020)
   EXTCONN(414D5143514D35202020202020202020)
   TYPE(CONN)
   PID(3446)                                TID(8)
   APPLDESC(WebSphere MQ Advanced Message Queuing Protocol)
   APPLTAG(java)                            APPLTYPE(SYSTEMEXT)
   ASTATE(STARTED)                          CHANNEL(MYAMQP)
   CLIENTID(recv_729b7a3)                   CONNAME(192.168.56.1)
   CONNOPTS(MQCNO_HANDLE_SHARE_NO_BLOCK,MQCNO_FASTPATH_BINDING)
   USERID(mqm)                              UOWLOG( )
   UOWSTDA( )                               UOWSTTI( )
   UOWLOGDA( )                              UOWLOGTI( )
   URTYPE(QMGR)
   EXTURID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
   QMURID(0.0)                              UOWSTATE(NONE)
```

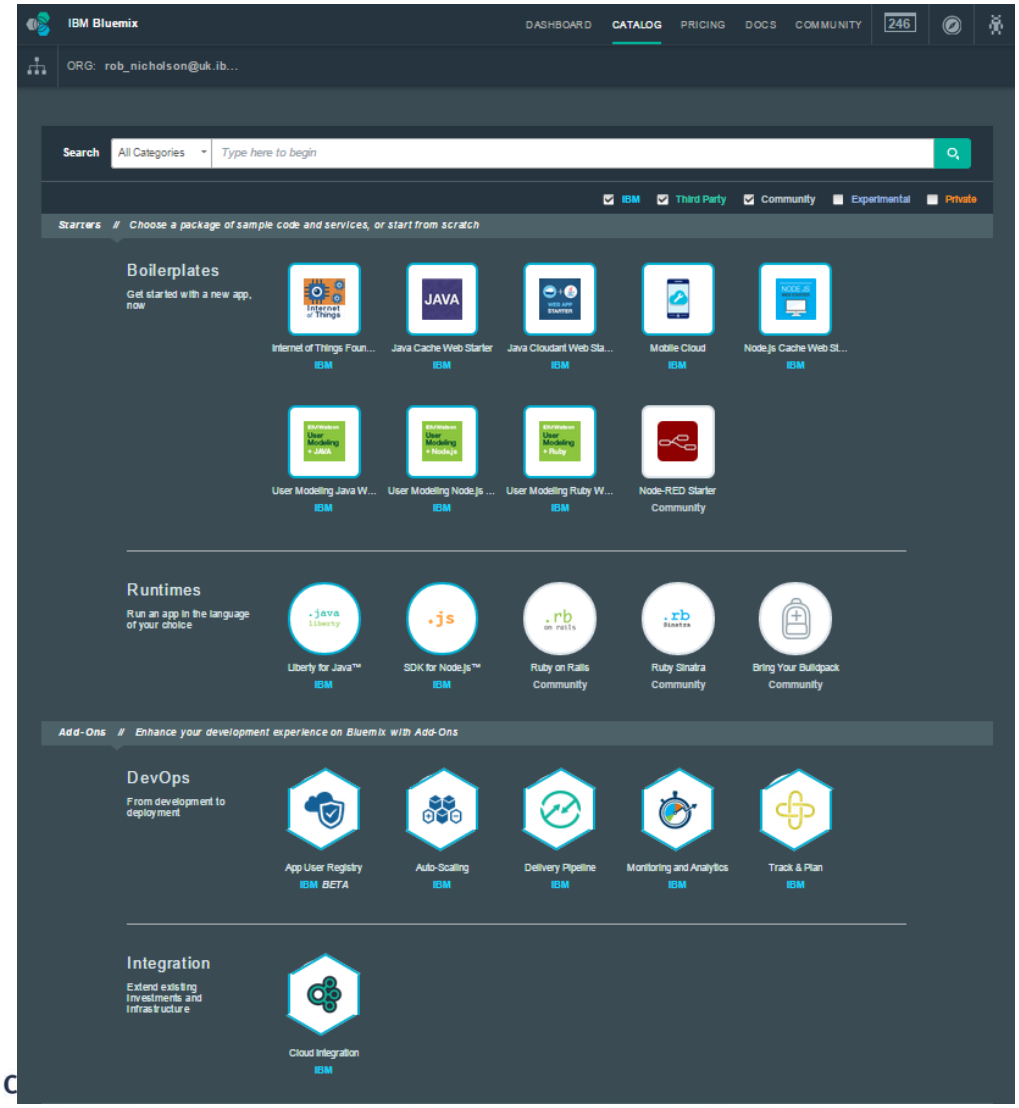
DIS CONN command

**Note the new client ID attribute set on the MQ connection**

# AMQP channels



# MQ Light on IBM Bluemix




## Run Your Apps

The developer can chose any language runtime or bring their own. Just upload your code and go.

## DevOps

Development, monitoring, deployment and logging tools allow the developer to run the entire application

## APIs and Services

A catalog of open source, IBM and third party APIs services allow a developer to stitch together an application in minutes.

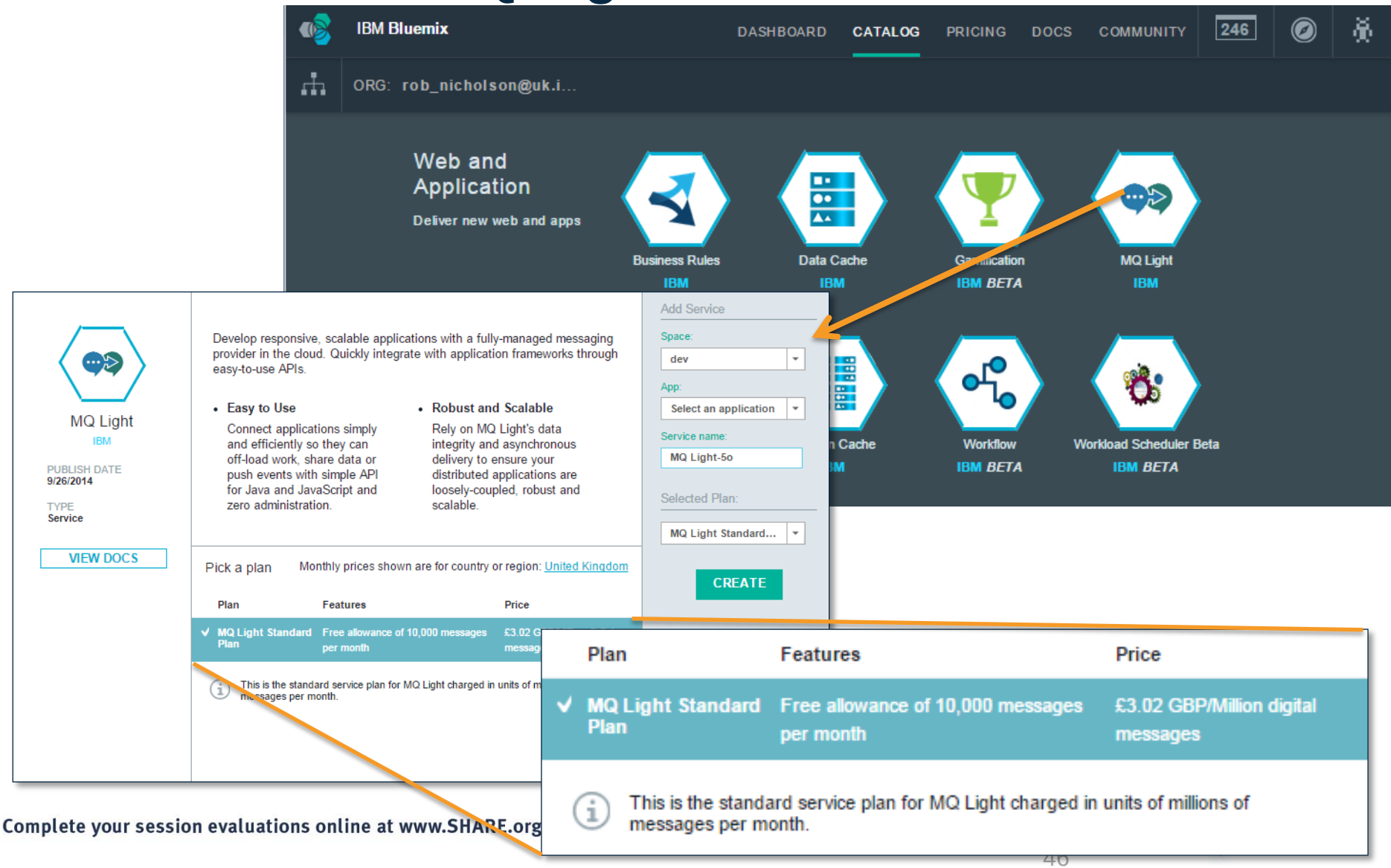
## Cloud Integration

Build hybrid environments. Connect to on-premises systems of record plus other public and private clouds. Expose your own APIs to your developers.

## Built on IBM SoftLayer

Runs automatically on top of IBM's leading infrastructure as a service. No need to worry about provisioning or managing infrastructure.

# Introduction to MQ Light Service



# Some other example cloud deployment patterns for MQ…

- Hypervisor editions
- IBM PureSystems patterns
- MQ on Azure

# MQ Hypervisor edition


- MQ Hypervisor Editions allow automation and standardisation of the traditional approach to provisioning messaging systems, which combined with IWD/PureApp gives many benefits:
  - Standardization of software images reduces risk and uncertainty
  - Automated provisioning reduces errors and speeds time to value
  - Repeatable configuration across sets of machines is quicker and less error-prone
  - Applying software maintenance is simpler and quicker using IWD/IPAS GUI or CLI
  - Comprehensive history/audit is maintained
  - License tracking is integrated

# MQ on Azure

- Recently announced offering allows enterprises to deploy existing IBM licences into managed systems in the Azure cloud

- MQ is now available as a deployable pattern

- Enables scalable MQ deployments using Azure's platform- and infrastructure-as-service capabilities

# Links

- MQ Light
  - https://developer.ibm.com/messaging/mq-light/
- MQ Light for Bluemix
  - http://www.bluemix.net
- MQ Hypervisor edition for Red Hat
  - http://www.ibm.com/software/products/en/wmq-hyper-redhat/
- MQ Hypervisor edition for AIX
  - http://www.ibm.com/software/products/en/wmq-hypervisor-aix
- MQ on Microsoft Azure
  - https://msopentech.com/blog/2014/11/04/ibm-websphere-mq-db2-now-microsoft-azure/
- Extending IBM WebSphere MQ WebSphere Message Broker to the Clouds
  - https://ibm.biz/BdENPZ

# Agenda

- Does my traditional MQ network make sense in the cloud era?
- Client user growing pains
- Rapid development
- Scalability
- Demo – Provisioning MQ for z/OS

# This was Session 17055. The rest of the week ……

| | Monday | Tuesday | Wednesday | Thursday | Friday |
|---|---|---|---|---|---|
| 08:30 | | | 17060: Understanding MQ Deployment Choices and Use Cases | 17051: Application Programming with MQ Verbs [z/OS & Distributed] | |
| 10:00 | 17036: Introduction to MQ - Can MQ Really Make My Life Easier? [z/OS & Distributed] | | 17052: MQ Beyond the Basics - Advanced API and Internals Overview [z/OS & Distributed]<br>17035: MQ for z/OS, Using and Abusing New Hardware and the New V8 Features [z/OS] | 17054: Nobody Uses Files Any More do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB [z/OS & Distributed] | 17057: Not Just Migrating, but Picking up New Enhancements as You Go - We've Given You the Shotgun, You Know Where Your Feet Are [z/OS & Distributed] |
| 11:15 | 17041: First Steps with IBM Integration Bus: Application Integration in the New World [z/OS & Distributed] | | 16732: MQ V8 Hands- on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs! | 17046: Paging Dr. MQ - Health Check Your Queue Managers to Ensure They Won't Be Calling in Sick! [z/OS] | 17053: MQ & DB2 – MQ Verbs in DB2 & InfoSphere Data Replication (Q Replication) Performance [z/OS] |
| 01:45 | 17037: All About the New MQ V8 [z/OS & Distributed] | 17034: MQ Security: New V8 Features Deep Dive [z/OS & Distributed] | 17040: Using IBM WebSphere Application Server and IBM MQ Together [z/OS & Distributed] | 17062: End to End Security of My Queue Manager on z/OS [z/OS] | All sessions in Seneca unless otherwise noted. |
| 03:15 | 17042: What's New in IBM Integration Bus [z/OS & Distributed] | 17065: Under the hood of IBM Integration Bus on z/OS - WLM, SMF, AT-TLS, and more [z/OS] | 17043: The Do's and Don'ts of IBM Integration Bus Performance [z/OS & Distributed] | 17039: Clustering Queue Managers - Making Life Easier by Automating Administration and Scaling for Performance [z/OS & Distributed] | |
| 04:30 | 17059: IBM MQ: Are z/OS & Distributed Platforms like Oil & Water? [z/OS & Distributed] | 17055: What's the Cloud Going to Do to My MQ Network? | 17044: But Wait, There's More MQ SMF Data Now?!?! - Monitoring your Channels Using V8's New Chinit SMF Data [z/OS] | 17068: Monitoring and Auditing MQ [z/OS & Distributed] | |