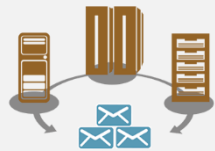


Running and Supporting MQ Light Applications

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

Agenda

- **What is MQ Light?**
- **The MQ Light API**
- **Running MQ Light apps in Bluemix**
- **IBM MQ support for MQ Light**
- **Demos**



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

What is MQ Light?


1. A new messaging API
2. A messaging runtime for on-premise development
3. A PaaS messaging runtime for admin-free cloud deployment (the MessageHub Service in Bluemix)

More on all of these throughout the slides...

MQ Light : Software and Cloud



IBM Bluemix



MQLight
IBM

PUBLISH DATE
30/6/2014

TYPE
Service

[VIEW DOCS](#)

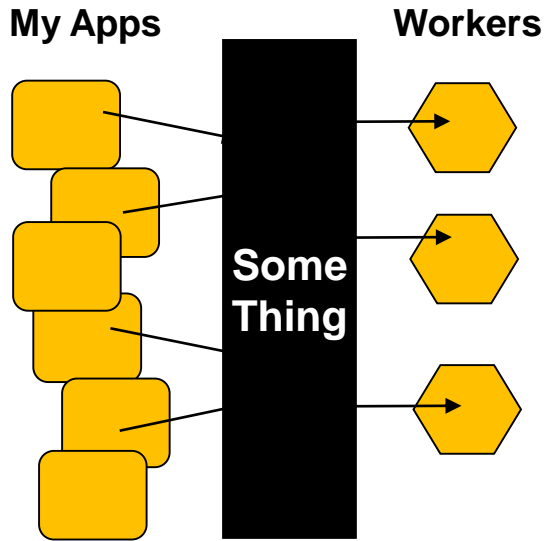
BETA Develop responsive, scalable applications with a fully-managed messaging provider in the cloud. Quickly integrate with application frameworks through easy-to-use APIs.

- **Easy to Use**
Connect applications simply and efficiently so they can off-load work, share data or push events with simple API for Java and JavaScript and zero administration.
- **Robust and Scalable**
Rely on MQLight's data integrity and asynchronous delivery to ensure your distributed applications are loosely-coupled, robust and scalable.

- Messaging that application developers will love to use, helping them make responsive applications that scale easily
- 3 ways to run an MQ Light application:
 - ▶ Bluemix Message Hub service (MQ Light service now deprecated)
 - ▶ MQ Light software download
 - ▶ IBM MQ 8.0.0.4
- Open APIs crafted to feel natural in a growing range of popular languages
- Tooling that makes modular app development easy

The journey that got us here

I want to execute code without taxing my Web app processes

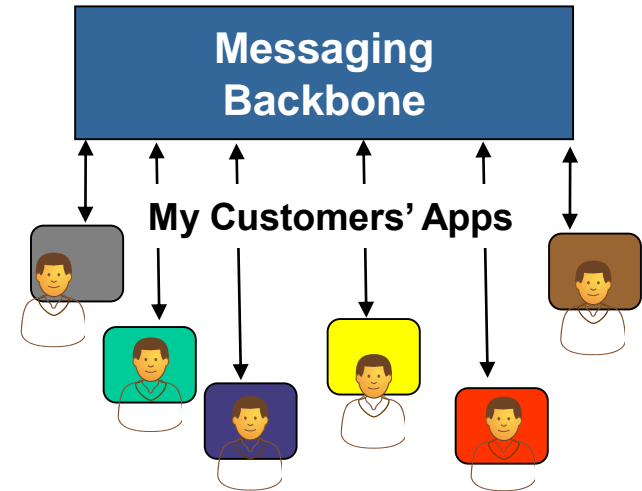


Andy
Developer

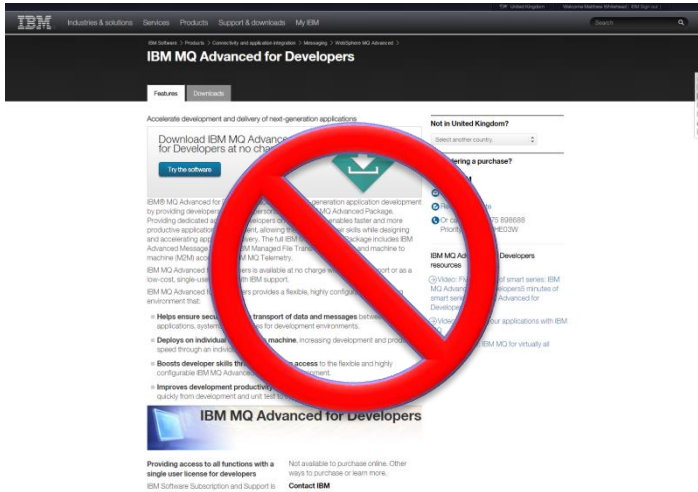


Iain
Infrastructure
Guy

My job is run a communications service for my customers' apps



Corporate-free look & feel



IBM MQ Advanced for Developers

Download IBM MQ Advanced for Developers at no charge

Try the software

IBM MQ Advanced for Developers is a next-generation application development tool that provides dedicated support for developers, allowing them to build and more productive applications faster and more easily while designing and accelerating applications. The full IBM MQ Advanced Package includes IBM Advanced Message Queue (AMQP) Telemetry, IBM Managed File Transfer (MFT) and IBM Machine Access (MMA) and IBM MQ Advanced for Developers.

IBM MQ Advanced for Developers is available at no charge to developers as a low-cost, single-user license with IBM support.

IBM MQ Advanced for Developers provides a flexible, highly configurable environment that:

- Helps ensure secure transport of data and messages between applications, systems and machines, increasing development and production speed through an individual machine.
- Deploys on individual machines, increasing development and production speed through an individual machine.
- Boosts developer skills through access to the flexible and highly configurable IBM MQ Advanced for Developers environment.
- Improves development productivity, quickly from development and unit test environments.

Providing access to all functions with a single user license for developers

IBM Software Subscription and Support is not available to purchase online. Other ways to purchase or learn more.

Contact IBM



IBM Messaging Blog Answers

IBM MQ Light

Downloads Docs Forum

IBM MQ Light

A simple yet powerful AMQP based messaging API. Write apps that run locally, in the cloud, or alongside IBM MQ. Find out more or download the free MQ Light SDK now:

Version 1.0.0.2

Detected OS: Windows [Download for a different OS](#)

(Don't forget to grab a client!)

MQ Light Clients

Got the MQ Light SDK? Now choose your preferred programming language to view sample code and client install instructions:

[node.js](#) [Java](#) [Ruby](#) [Python](#) [Other](#)

```
# Receive:
var mqlight = require('mqlight');
var recvClient = mqlight.createClient({service: 'amqp://localhost'});
recvClient.on('started', function() {
  recvClient.subscribe('news/technology');
  recvClient.on('message', function(data, delivery) {
    console.log(data);
  });
});

# Send:
var mqlight = require('mqlight');
var sendClient = mqlight.createClient({service: 'amqp://localhost'});
sendClient.on('started', function() {
  sendClient.send('news/technology', 'Hello World!');
});
```

To install this client: `npm install mqlight@1.0`

Join The Discussion

Read the latest Blogs from the developers, or ask a question on our Forum or tweet us @mqlight.

Latest Blogs

14 August 2015

"MQ Light on Rails" by SHead [Read More...](#)

13 August 2015

"Node.js – Building messaging systems using IBM MQ Light and Bluemix" by JoshuaCarr [Read More...](#)

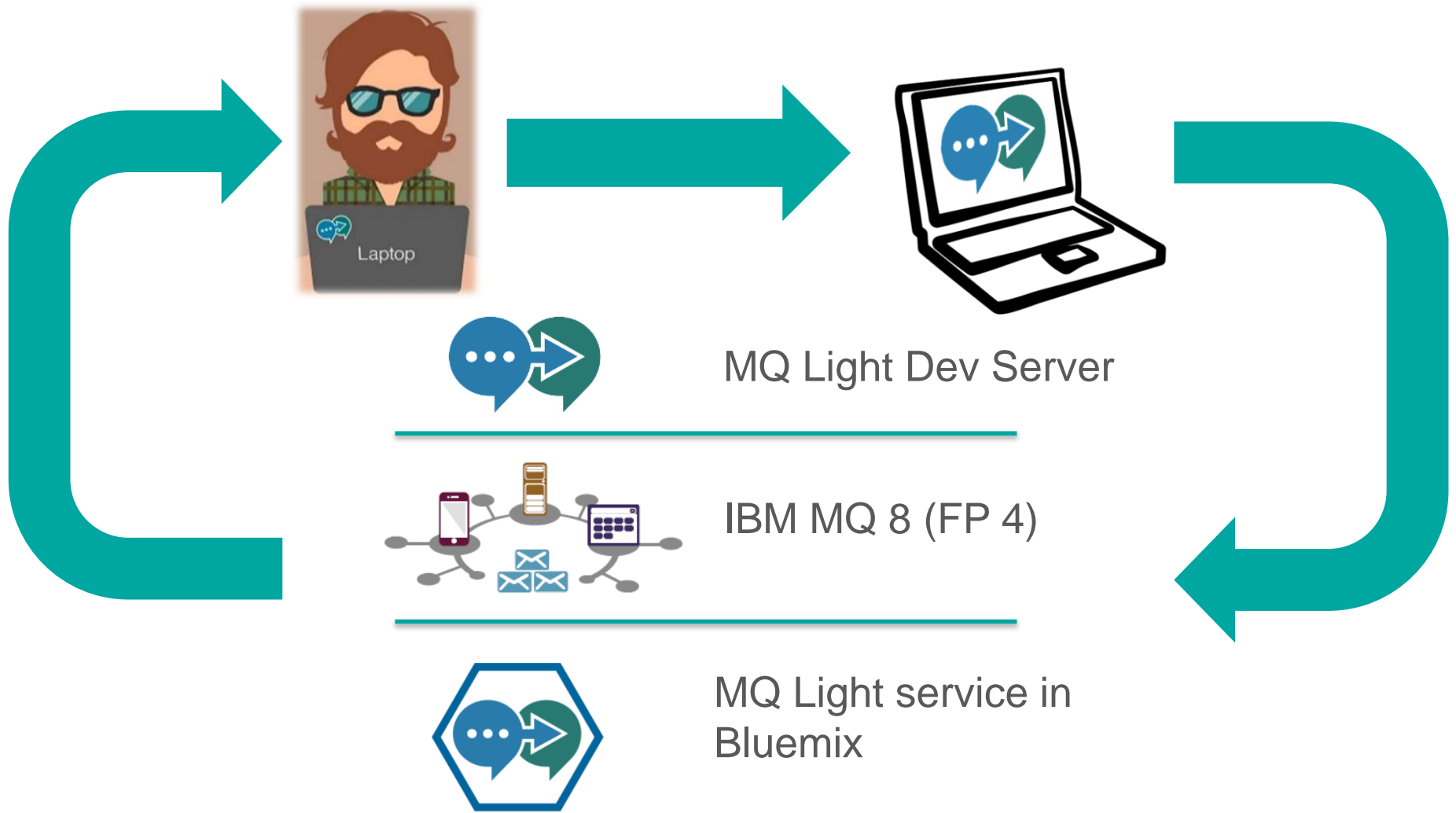
24 July 2015

"MQ Light, Fish Alive and Docker Compose" by David McCann [Read More...](#)

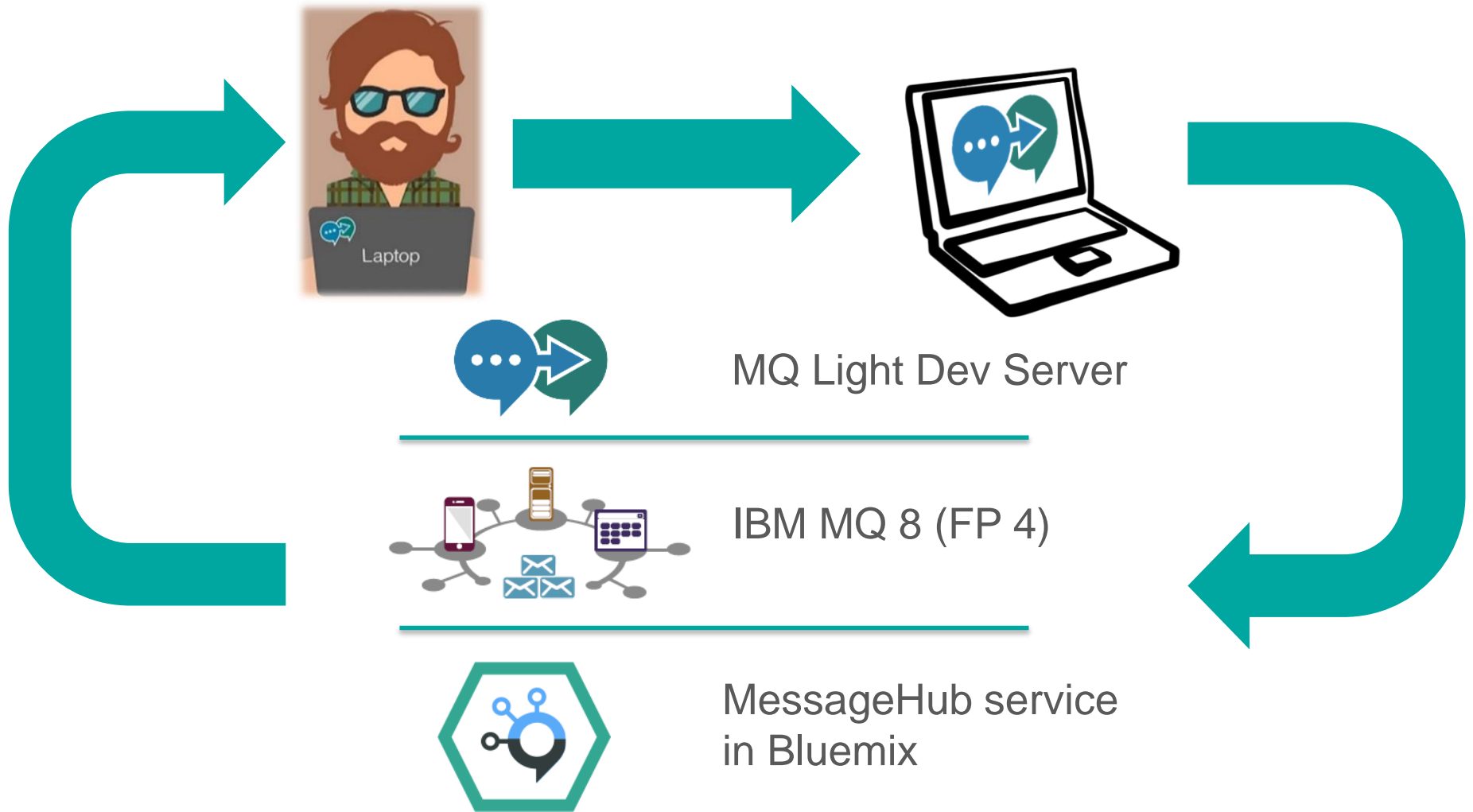
23 July 2015

"Messaging with Bluemix from the Edge of Space, #IBMImp1 – Epilogue" by j0nnymac [Read More...](#)

Deployment Options (last year's slide)



Deployment Options



Web Based Tooling

The screenshot displays the IBM MQ Light web-based tooling interface. At the top left, the title "IBM MQ Light" is shown, with navigation links for "View Messages" and "Documentation". The top right corner features the IBM logo and a status indicator: "Clients: 3 connected 1 disconnected". A timestamp indicates the data is from "0 min" since the last history clear.

The interface is divided into three main sections:

- Senders:** Located on the left, it shows a list of senders. One sender, "send.js", is highlighted with a red square icon and a speech bubble icon containing the number "4".
- Messages:** The central area displays a list of messages. The "Destination" is set to "Any". The messages list includes:
 - Message 1: Received "<1 min" ago, content "etc.", with a status of "2 / 2" and a "Details" link.
 - Message 2: Received "<1 min" ago, content "Here are some football results", with a status of "2 / 2" and a "Details" link.
 - Message 3: Received "<1 min" ago, content "Here is another message", with a status of "2 / 2" and a "Details" link.
 - Message 4: Received "<1 min" ago, content "Hello World", with a status of "2 / 2" and a "Details" link.
- Receivers:** Located on the right, it shows a list of receivers. Two receivers are visible:
 - "recv.js": Status "2 / 2", Destination "public", with a speech bubble icon containing "4".
 - "workers": Status "2 / 2", Destination "public", with a speech bubble icon containing "4" and a "Details" link.

At the bottom of the Receivers section, there is a link for "Privacy Policy Considerations".

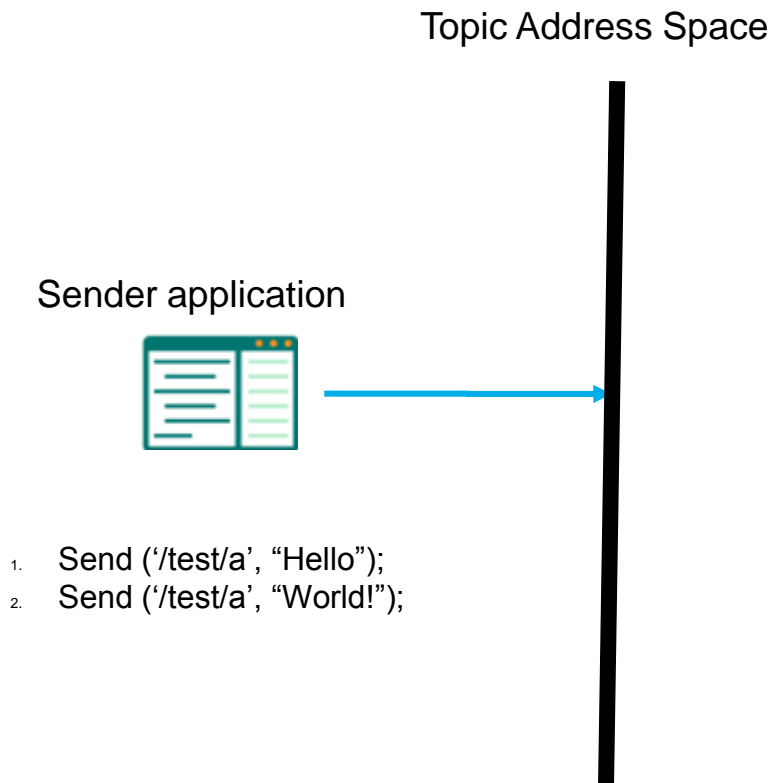
Agenda

- What is MQ Light?
- **The MQ Light API**
- Running MQ Light apps in Bluemix
- IBM MQ support for MQ Light
- Demo

MQ Light API - Language support

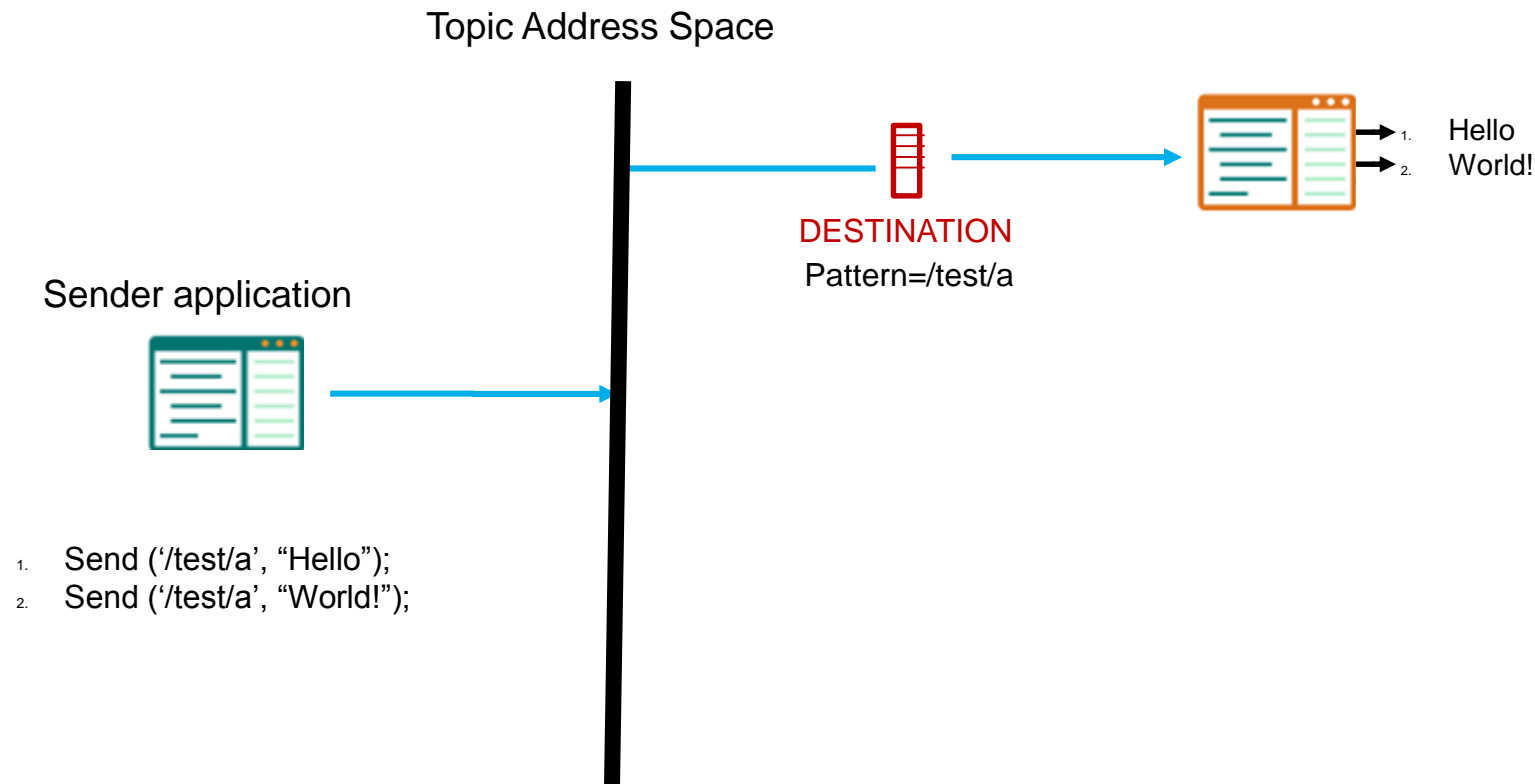
- Simple, programming Language neutral **messaging model**
- Idiomatic language & framework **API Mappings**
 - Frictionless development
- **Open wire protocol**
 - Open Source client libraries
 - Facilitates community drivers for languages & frameworks

MQ Light Messaging Model – Send Messages



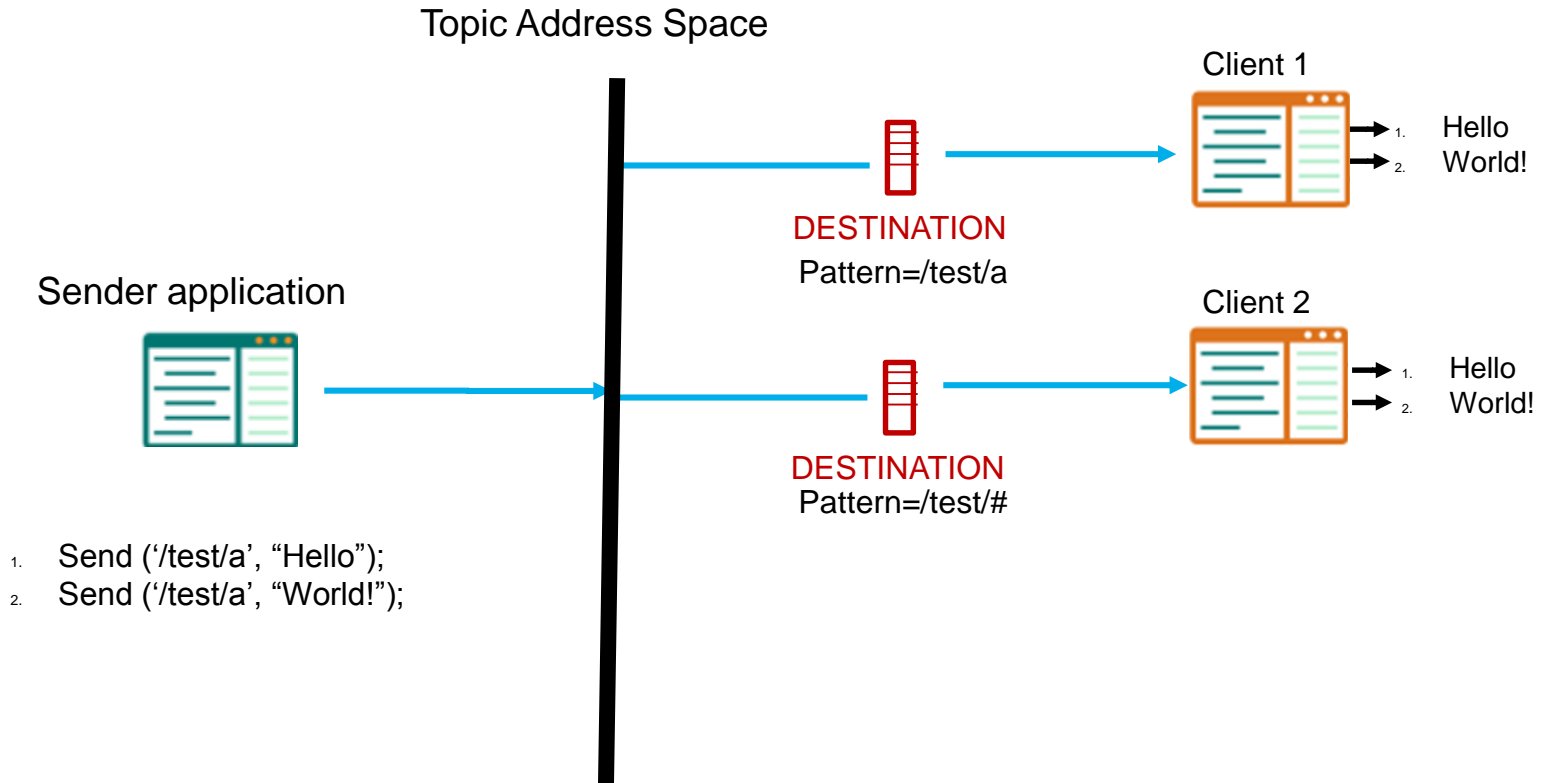
Applications send messages to a **topic**.
A topic is an address in the topic space
either flat or arranged hierarchically.

MQ Light Messaging Model – Simple Receive



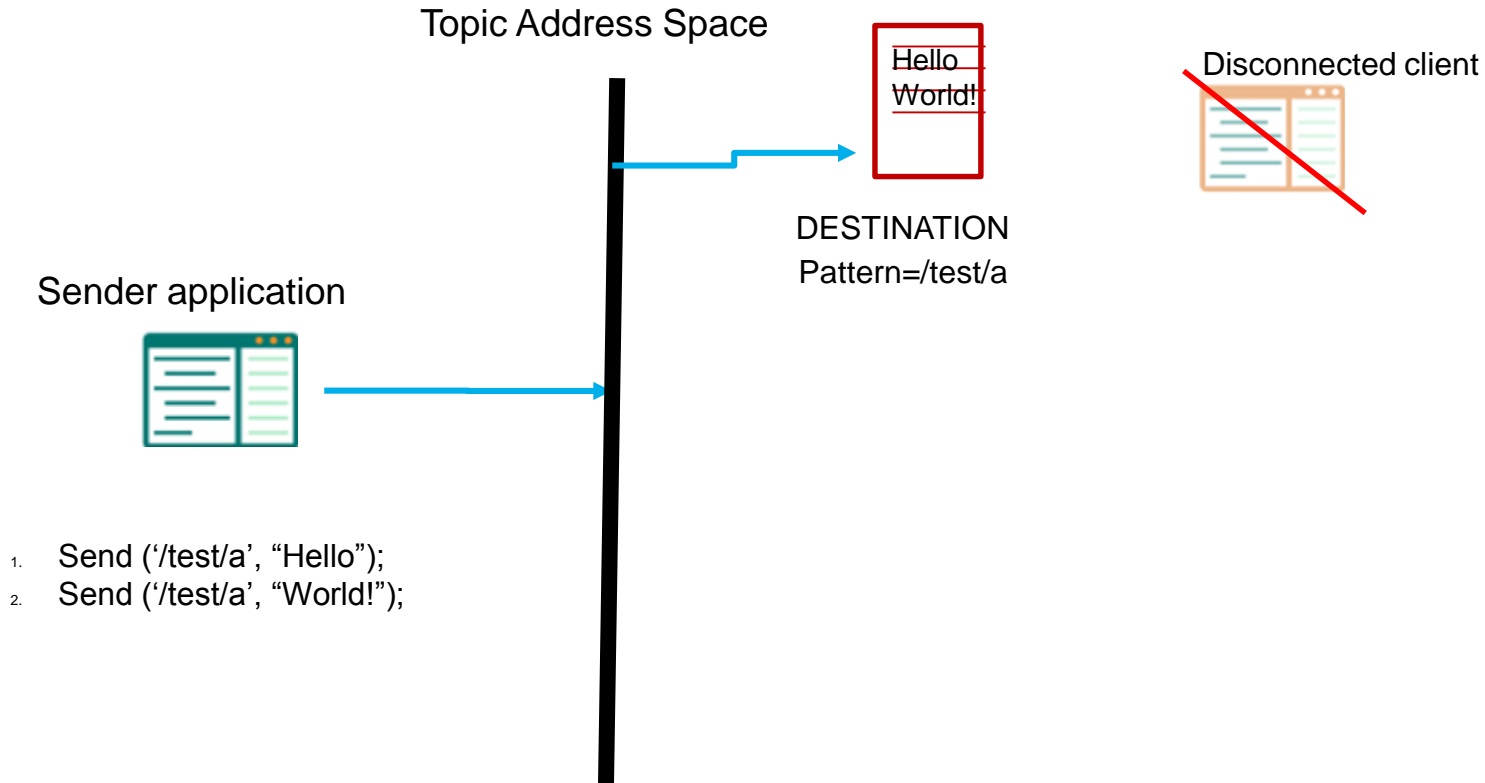
- Applications receive messages by creating a **destination** with a pattern which matches the topics they are interested in.
- Pattern matching scheme based on WMQ.

MQ Light Messaging Model – Pub/Sub



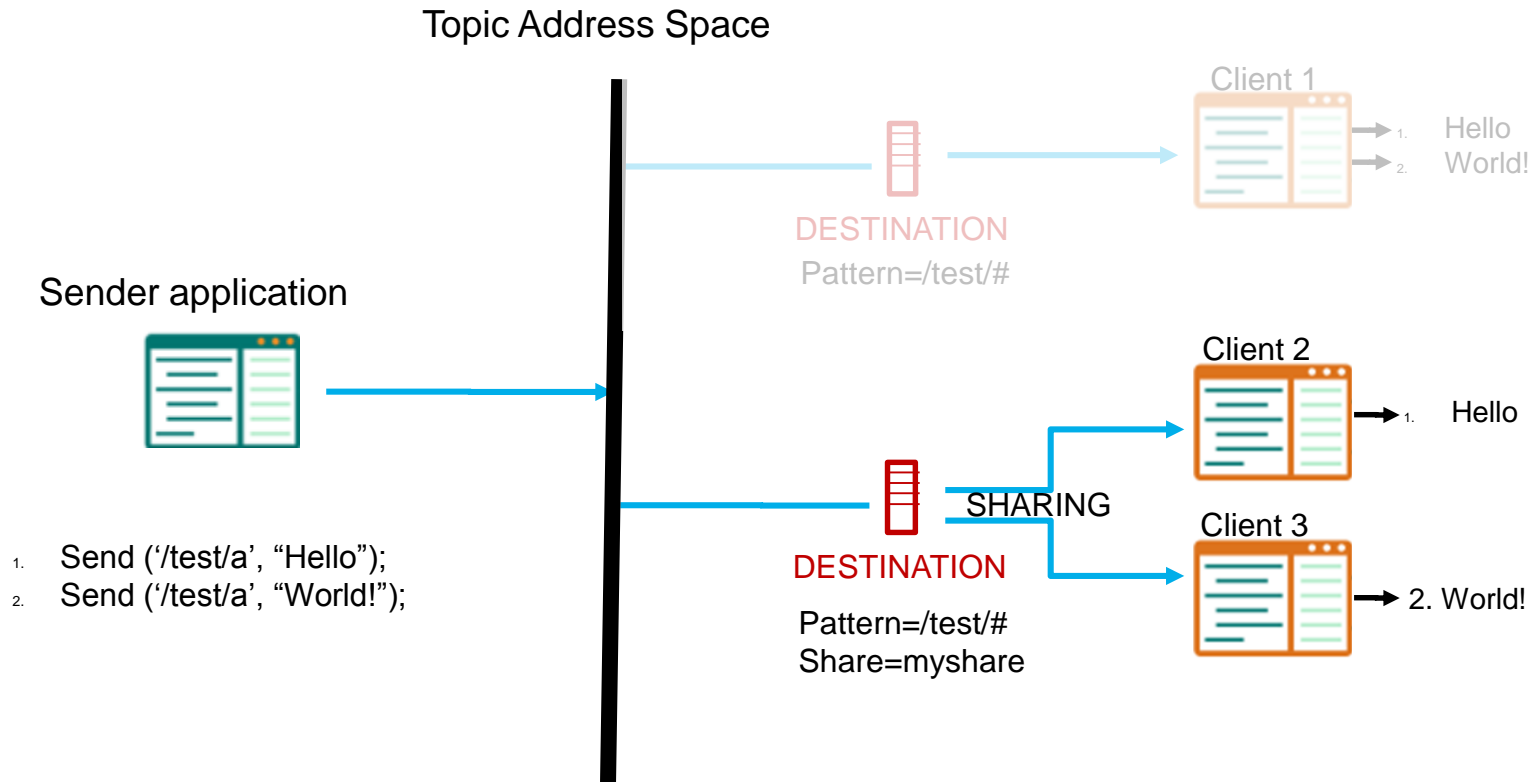
- Multiple destinations can be created which match the same topic
 - Pub/Sub style.

MQ Light Messaging Model – Persistent destinations



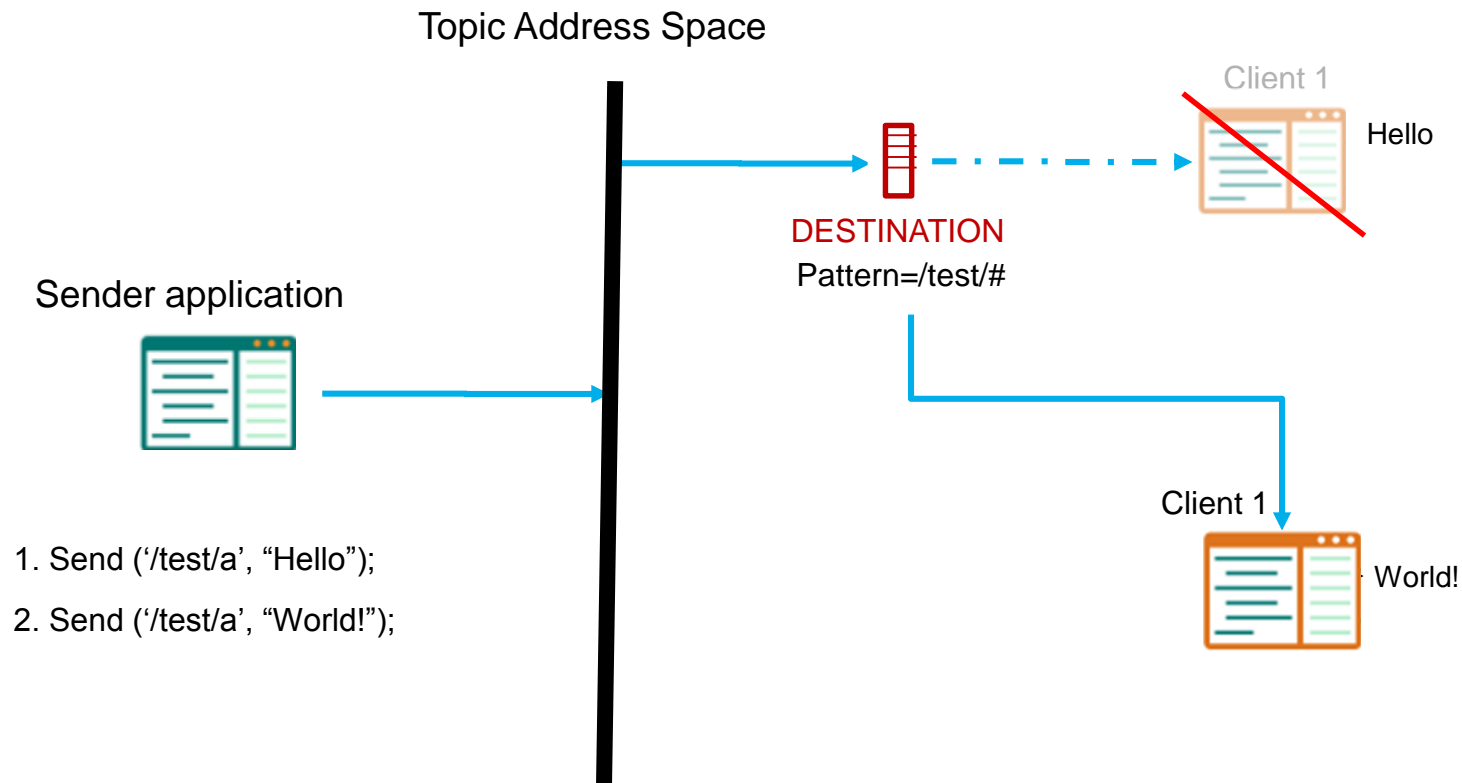
- . Destinations persist for a defined “time to live” after receiver detaches.

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 specify (optional) client ID.
- Re-using the same client ID pre-empts the original connection.
 - Ideal for worker takeover in the cloud.

MQ Light Node.JS API


- Installable from NPM
- Fully non blocking – Node.JS style
- Fluent programming style - wrappable into promises.
- Focussed on code simplicity.
- Client seamlessly retries across cloud endpoints

Receive:

```
var mqlight = require('mqlight');
var rcvClient = mqlight.createClient({service: 'amqp://localhost'});
rcvClient.on('started', function() {
  rcvClient.subscribe('news/technology');
  rcvClient.on('message', function(data, delivery) {
    console.log(data);
  });
});
```

Send:

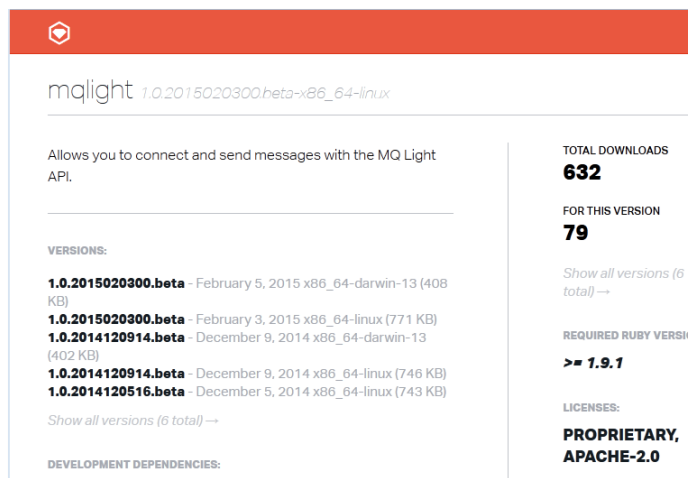
```
var mqlight = require('mqlight');
var sendClient = mqlight.createClient({service: 'amqp://localhost'});
sendClient.on('started', function() {
  sendClient.send('news/technology', 'Hello World!');
});
```



The screenshot shows the NPM package page for 'mqlight'. At the top, there is the NPM logo, a search bar, and a user profile for 'Rob Nicholson'. The package name 'mqlight' is displayed with a star icon. Below it, the description reads 'IBM MQ Light Client Module'. A terminal snippet shows the command '\$ npm install mqlight'. A line graph shows download trends from March 30 to April 13, with a peak around April 13. Below the graph, it states '19 downloads yesterday'. The 'Last Published By' field shows 'dominic.evans'. The 'Maintainers' list includes 'rob_nicholson', 'jnuber', 'matthew1001', and 'dominic.evans'. At the bottom, the version is listed as '0.1.2014041603' with a note 'last updated 3 days ago'.

MQ Light Ruby API

- Installable from rubygems.org
- Synchronous/blocking client.
- Bluemix connection support
- Backlog
 - ▶ Auto reconnect.
 - ▶ Asynchronous non blocking
 - ▶ TLS



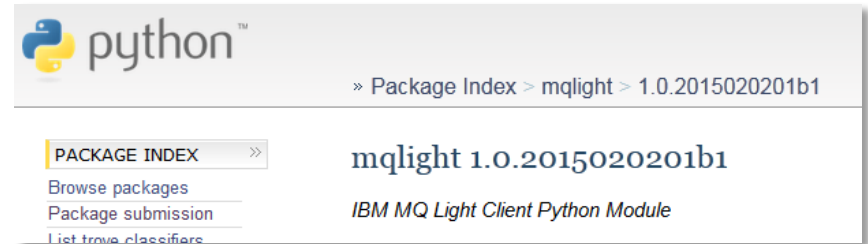
The screenshot shows the RubyGems.org page for the `mqlight` gem. The page title is `mqlight 1.0.2015020300.beta-x86_64-linux`. The description states: "Allows you to connect and send messages with the MQ Light API." The page lists several versions of the gem, including `1.0.2015020300.beta`, `1.0.2015020300.beta`, `1.0.2014120914.beta`, `1.0.2014120914.beta`, and `1.0.2014120516.beta`. The page also shows the total number of downloads (632) and the number of downloads for this version (79). The required Ruby version is `>= 1.9.1`. The licenses are listed as `PROPRIETARY, APACHE-2.0`.

```
# Receive:
require 'mqlight'
client = Mqlight::BlockingClient.new('amqp://localhost')
client.subscribe('news/technology')
delivery = client.receive('news/technology')
puts delivery.data

# Send:
require 'mqlight'
client = Mqlight::BlockingClient.new('amqp://localhost')
client.send('news/technology', 'Hello World!')
```

MQ Light Python API (Beta)

- Installable from pypi.python.org
- Non blocking
- Client seamlessly retries across cloud endpoints
- Backlog
 - ▶ TLS



```
# Receive:
require 'mqlight'
client = Mqlight::BlockingClient.new('amqp://localhost')
client.subscribe('news/technology')
delivery = client.receive('news/technology')
puts delivery.data

# Send:
require 'mqlight'
client = Mqlight::BlockingClient.new('amqp://localhost')
client.send('news/technology', 'Hello World!')
```

MQ Light Non Blocking Java API

- Installable using Maven
- Code is opensource on github.
- Non blocking
- Client seamlessly retries across cloud endpoints



```
void send() {
    NonBlockingClient.create("amqp://localhost", new NonBlockingClientAdapter<Void>() {
        public void onStarted(NonBlockingClient client, Void context) {
            client.send("news/technology", "Hello World!", null);
        }
    }, null);
}

void receive() {
    NonBlockingClient.create("amqp://localhost", new NonBlockingClientAdapter<Void>() {

        public void onStarted(NonBlockingClient client, Void context) {
            client.subscribe("news/technology", new DestinationAdapter<Void>() {
                public void onMessage(NonBlockingClient client, Void context, Delivery delivery) {
                    if (delivery.getType() == Delivery.Type.STRING) System.out.println(((StringDelivery)delivery).getData());
                }
            }, null, null);
        }
    }, null);
}
```

Agenda

- What is MQ Light?
- The MQ Light API
- **Running MQ Light apps in Bluemix**
- IBM MQ support for MQ Light
- Demo

IBM Bluemix

Bluemix is an **open-standards**, cloud-based platform for **building, running, and managing applications.**

Build your apps, your way

Use the most prominent compute technologies to power your app: Cloud Foundry, Docker, OpenStack.

Scale more than just instances

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

Extend apps with services

A catalog of IBM, third party, and open source services allow the developer to stitch an application together quickly.

Deploy and manage hybrid apps seamlessly

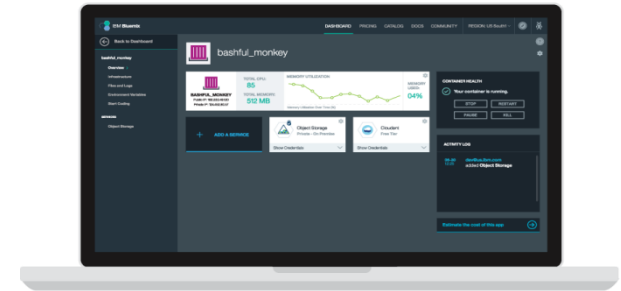
Get a seamless dev and management experience across a number of hybrid implementations options.

Layered Security

IBM secures the platform and infrastructure and provides you with the tools to secure your apps.

Flexible Pricing

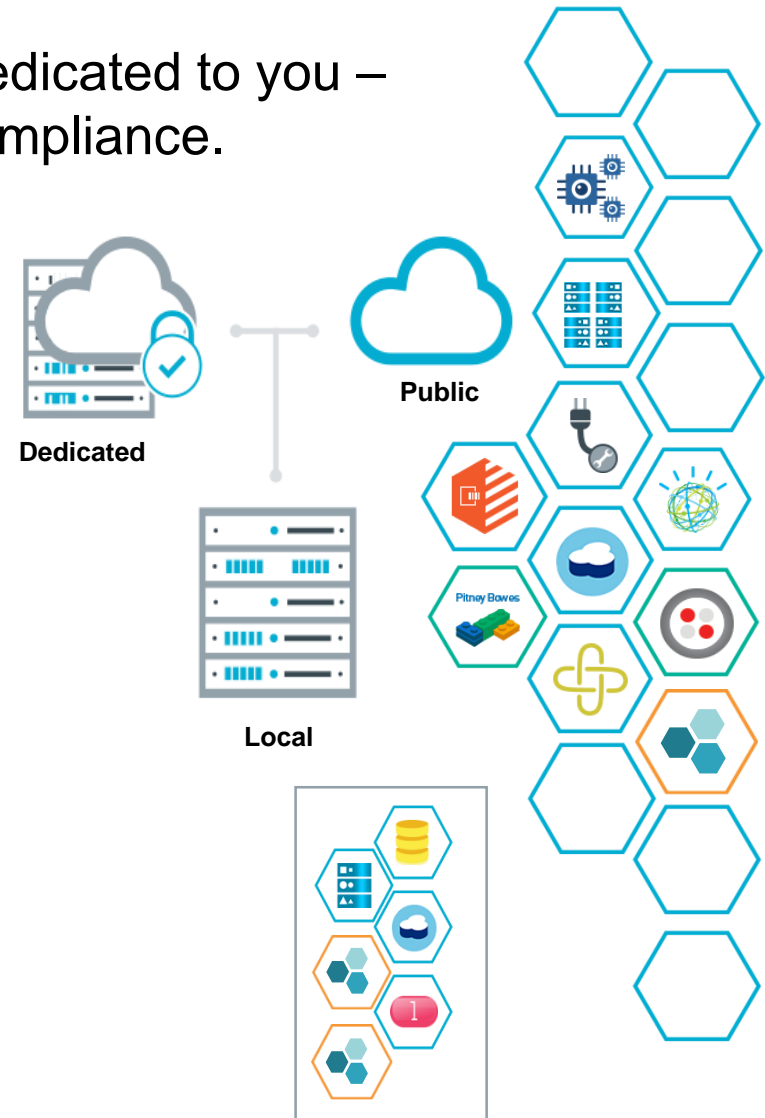
Try compute options and services for free and, when you're ready, pay only for what you use. Pay as you go and subscription models offer choice and flexibility.



Has a “Dedicated to You” option

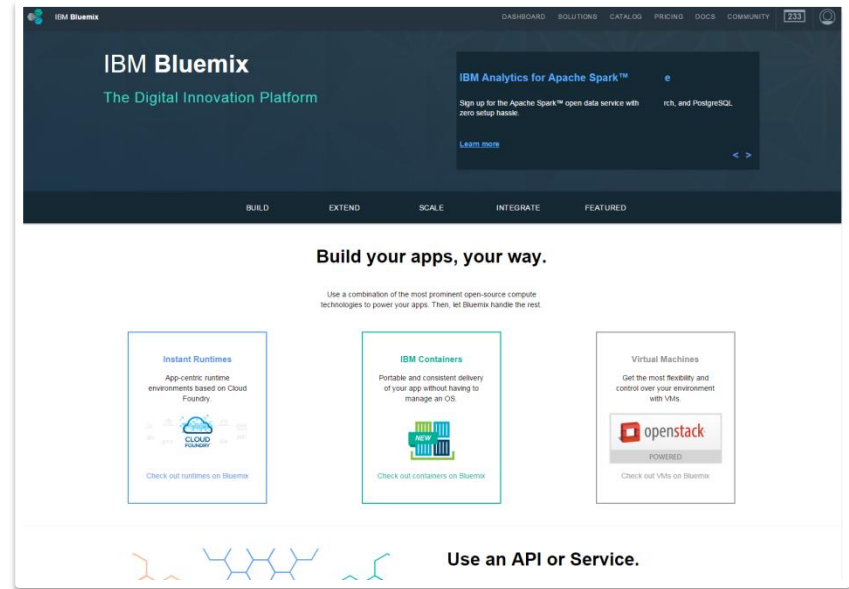
Single tenant hardware that's completely dedicated to you – allowing you to satisfy regulatory & legal compliance.

- The Bluemix platform and dedicated runtimes and services sit on SoftLayer hardware that is dedicated to you
- You still have the ability to connect to all multi-tenant services in the “public” catalog
- Integrated to your LDAP for developer authentication
- Elastic capacity based on your demands.



IBM Bluemix Apps

- Based on Pivotal® Cloud Foundry®
- Uses standard cf commands, e.g.
 - cf push <myapp>
 - cf start <myapp>
 - cf stop <myapp>
 - cf apps
 - cf create-service ...
 - cf bind-service ...



```
Administrator: C:\Windows\system32\cmd.exe
c:\Users\Matthew Whitehead>
c:\Users\Matthew Whitehead>cf apps
Getting apps in org mwhitehead@uk.ibm.com / space dev as mwhitehead@uk.ibm.com...
OK
name                requested state  instances  memory  disk  urls
app.js              stopped         0/1        1G      1G
mql.fishalive.node.backend  stopped         0/2        128M    1G
mql.fishalive.node.frontend started         1/1        128M    1G    mqlight-fishalive-node-unmoody-del
mql.uiworkout.node  stopped         0/1        128M    1G
mrw-demo            stopped         0/1        512M    1G    mrw-demo.mybluemix.net
nodered-custom-mrw  started         1/1        512M    1G    nodered-custom-mrw.mybluemix.net
NRMRW2              started         1/1        512M    1G    NRMRW2.mybluemix.net
web.js              started         1/1        512M    1G    mrwweb.js.mybluemix.net
c:\Users\Matthew Whitehead>
```

IBM Bluemix Services

IBM Bluemix Console

All Categories

Boilerplates

Compute

Network

Storage

Data & Analytics

Watson

Internet of Things

ADP

Blockchain

Business Rules

Message Hub

Workload Scheduler

IBM Beta

IBM

IBM

box

IBM Bluemix Console

View All

Message Hub

IBM Message Hub is a scalable, high-throughput message bus. Wire micro-services together using open protocols. Connect stream data to analytics to realize powerful insights. Feed event data to multiple applications to react in real time. Bridge to your on-premise messaging infrastructure to create a hybrid cloud messaging solution.

Service name: Message Hub-13

Credential name: Credentials-1

Connect to: Leave unbound

Features

- Fast, scalable, fully managed messaging

Tiers	Pricing
1 - 200,000,000	£0.0605 GBP/Million Digital Messages
200,000,001 - 700,000,000	£0.0302 GBP/Million Digital Messages
700,000,001 - 1,700,000,000	£0.0181 GBP/Million Digital Messages
1,700,000,000+	£0.0102 GBP/Million Digital Messages

IBM Bluemix – Binding Apps and Services

Applications are bound to the services they require

This is the dashboard for one of my applications. It's bound to a Message Hub service instance I previously created

The screenshot shows the IBM Bluemix dashboard for the application `mql.uiworkout.node`. The status is "Your app is not running". The dashboard includes tabs for Getting Started, Overview, Runtime, Connections, Logs, and Monitoring. The Runtime section displays the buildpack (SDK for Node.js™), 1 instance (1 Stopped, 0 Running, Health is 0%), and 128 MBS per instance. The Connections section shows 1 connection to the Message Hub-s0 service instance.

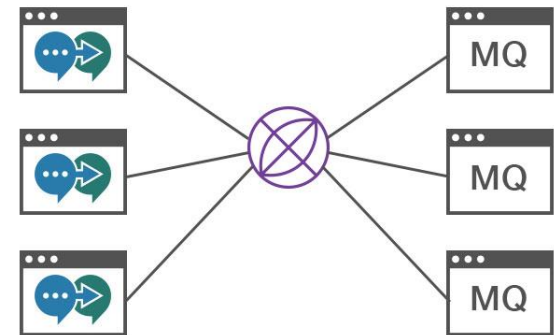
```
cf create-service messagehub standard "Message Hub-7u"  
cf bind-service mql.uiworkout.node "Message Hub-7u"
```

Agenda

- What is MQ Light?
- The MQ Light API
- Running MQ Light apps in Bluemix
- **IBM MQ support for MQ Light**
- Demo

New AMQP channel type

- Introduced in MQ 8.0.0.4
- Supported on distributed platforms (not z/OS/IBM i)
- Adds a channel type of “AMQP”
- Supports a subset of the AMQP 1.0 Oasis specification
- Interoperable with MQ FAP and MQTT applications (see later slides for details)



AMQP channels

- Configuration model

- MQSC and PCF updates allow you to administer AMQP channels in much the same way as other MQ objects

- DISPLAY CHANNEL(*) CHLTYPE(**AMQP**)
- DEFINE CHANNEL(MY.AMQP.CHANNEL) CHLTYPE(**AMQP**) PORT(5673)
- START CHANNEL(MY.AMQP.CHANNEL)
- STOP CHANNEL(MY.AMQP.CHANNEL)
- DISPLAY CHSTATUS(*) CHLTYPE(**AMQP**)

- PCF command types all valid, using MQIACH_CHANNEL_TYPE=**MQCHT_AMQP**:

- MQCMD_CREATE_CHANNEL MQCMD_DELETE_CHANNEL,
MQCMD_CHANGE_CHANNEL MQCMD_START_CHANNEL,
MQCMD_STOP_CHANNEL MQCMD_COPY_CHANNEL,
MQCMD_INQUIRE_CHANNEL MQCMD_INQUIRE_CHANNEL_STATUS

```
mwhitehead@ubuntu-vm: /opt/mqmAMQP
```

DIS CONN command

```
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)
```

Note the new client ID attribute set on the MQ connection


```
mwhitehead@ubuntu-vm: /opt/mqmAMQP
```

DIS CHSTATUS command

```
:
:
display chstatus(*) chlname(AMQP) clientid(*) all
  2 : display chstatus(*) chlname(AMQP) clientid(*) all
AMQ8417: Display Channel Status details.
CHANNEL(MYAMQP)          CLIENTID(recv_fced6d9)★
STATUS(RUNNING)         CONNAME(192.168.56.1)
KAIN(0)                 MCAUSER(mwhitehead)
CLNTUSER( )             MSGSNT(3)
MSGRCVD(0)              INDOUBTIN(0)
INDOUBTOUT(0)           PENDING(0)
LMSGDATE(2015-01-25)    LMSGTIME(03.29.31)
CHLSDATE(2015-01-25)    CHLSTIME(02.46.16)
PROTOCOL(AMQP)
```

Note, DIS CHSTATUS usage varies slightly from MQ channels

Administration – MQ Explorer

AMQP channels displayed alongside existing channels

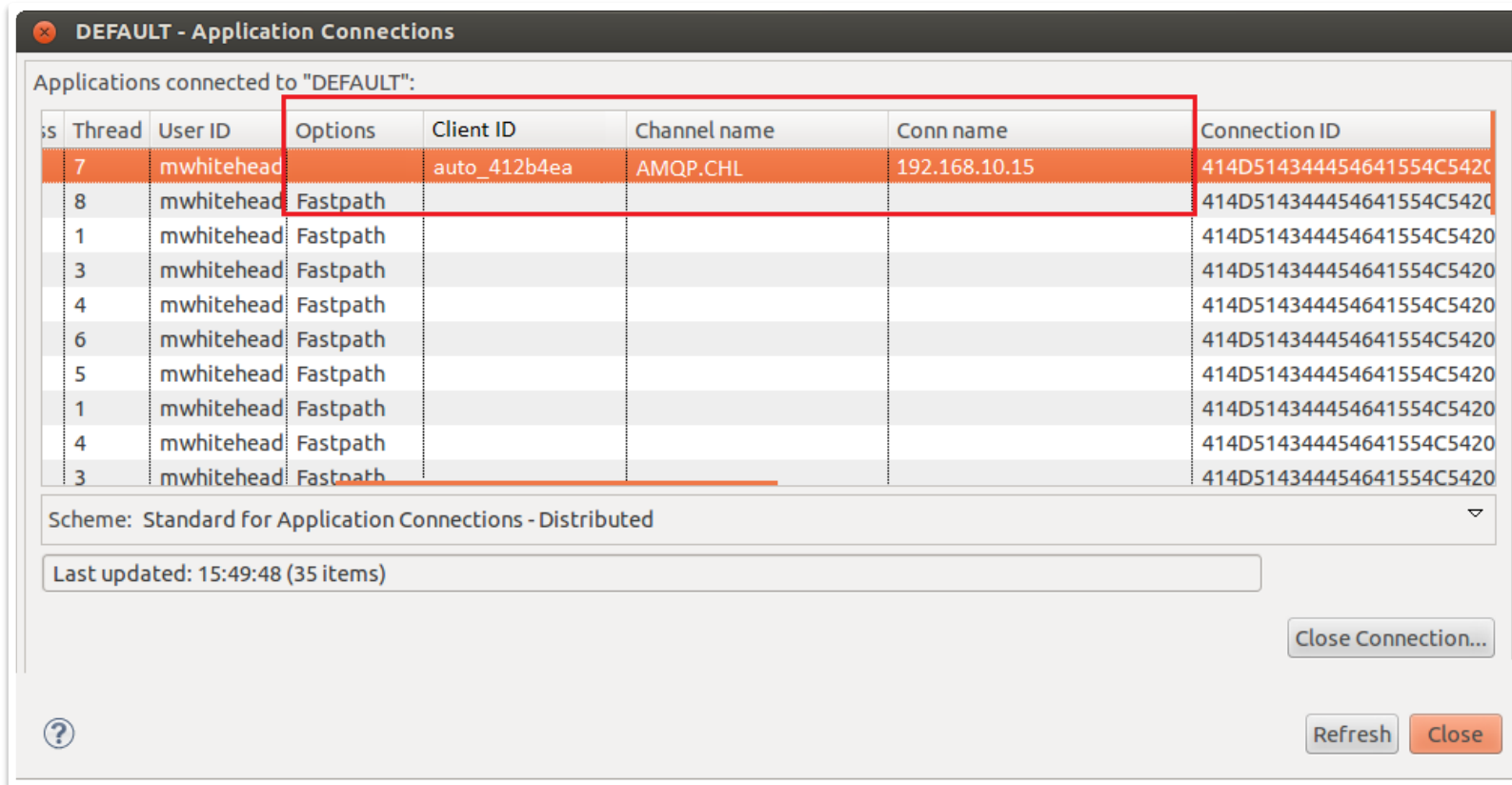
The screenshot shows the IBM WebSphere MQ Explorer interface. On the left, the 'Navigator' pane shows a tree view of the queue manager structure, including 'Queue Managers', 'Queue Managers', 'Queues', 'Topics', 'Subscriptions', and 'Channels'. The 'Channels' folder is expanded, showing 'Client Connections', 'Channel Authentication Records', 'Telemetry', 'Listeners', 'Services', 'Process Definitions', and 'Namelists'. On the right, the 'Content' pane displays a table of channels. The table has three columns: 'Channel name', 'Channel type', and 'Overall chan'. The first row, 'MYAMQPCHANNEL', is highlighted in orange and has an AMQP icon, indicating it is an AMQP channel. The other rows are standard MQ channels with various types like Receiver, Server-connection, Cluster-receiver, Cluster-sender, Requester, Sender, and Server. A black arrow points from the text above to the 'MYAMQPCHANNEL' row in the table.

Channel name	Channel type	Overall chan
MYAMQPCHANNEL	AMQP	Running
SYSTEM.AUTO.RECEIVER	Receiver	Inactive
SYSTEM.AUTO.SVRCONN	Server-connection	Inactive
SYSTEM.DEF.AMQP	AMQP	Running
SYSTEM.DEF.CLUSRCVR	Cluster-receiver	Inactive
SYSTEM.DEF.CLUSSDR	Cluster-sender	Inactive
SYSTEM.DEF.RECEIVER	Receiver	Inactive
SYSTEM.DEF.REQUESTER	Requester	Inactive
SYSTEM.DEF.SENDER	Sender	Inactive
SYSTEM.DEF.SERVER	Server	Inactive
SYSTEM.DEF.SVRCONN	Server-connection	Inactive

Administration – MQ Explorer

Application connections view use to display AMQP clients

- Connection name and channel name populated to show where the client has connected from
- New **Client ID** attribute in the Application Connections view



DEFAULT - Application Connections

Applications connected to "DEFAULT":

Process	Thread	User ID	Options	Client ID	Channel name	Conn name	Connection ID
	7	mwhitehead		auto_412b4ea	AMQP.CHL	192.168.10.15	414D514344454641554C5420
	8	mwhitehead	Fastpath				414D514344454641554C5420
	1	mwhitehead	Fastpath				414D514344454641554C5420
	3	mwhitehead	Fastpath				414D514344454641554C5420
	4	mwhitehead	Fastpath				414D514344454641554C5420
	6	mwhitehead	Fastpath				414D514344454641554C5420
	5	mwhitehead	Fastpath				414D514344454641554C5420
	1	mwhitehead	Fastpath				414D514344454641554C5420
	4	mwhitehead	Fastpath				414D514344454641554C5420
	3	mwhitehead	Fastpath				414D514344454641554C5420

Scheme: Standard for Application Connections - Distributed

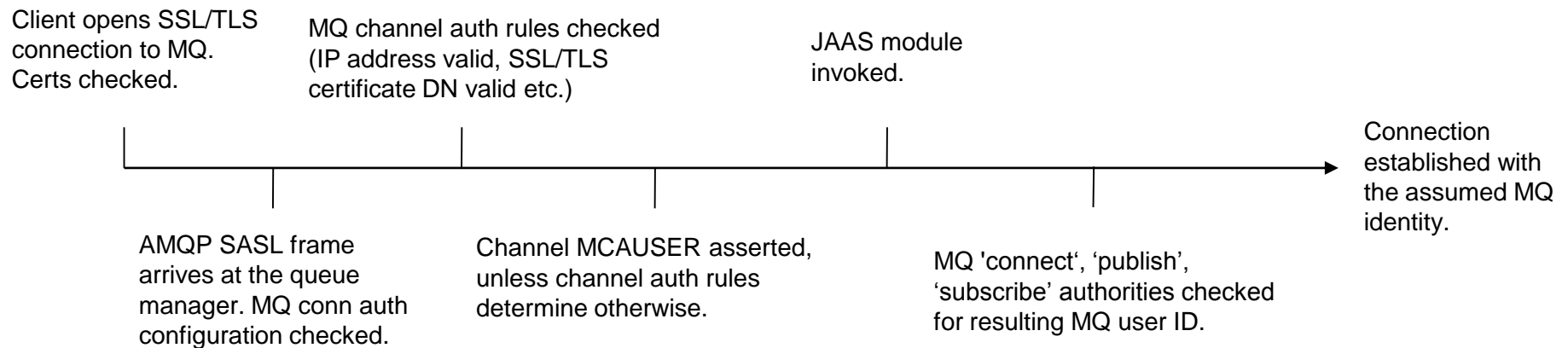
Last updated: 15:49:48 (35 items)

Close Connection...

Refresh Close

Authorities and authentication

- MQ authenticates the client and authorises messaging in a similar way to MQ clients. Similarly administration changes and commands for AMQP channels are authorised in the same way as for other MQ channels. Allows you to specify who has authority to:
 - Start or stop a channel
 - Change a channel's configuration
 - Display a channel's status
 - Delete a channel

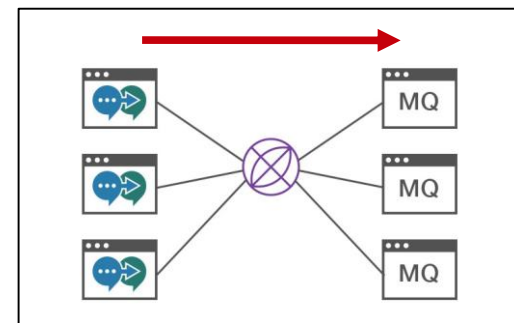


- Events
 - MQ provides events for monitoring different activities
 - Some are available to try in the beta
 - Command events (e.g. request to start a channel)
 - Configuration events (e.g. request to change channel attrs)
 - Some are ones we'd like to do
 - Security events (e.g. an AMQP client failed an authority check)

- Backup/Restore
 - MQ provides tools to saving and restoring queue manager configuration
 - dmpmqcfg and runmqsc
 - These have been updated to include AMQP channel definitions
 - Service may request that custom tuning/service parameters be set in a .properties file. If so, then this file must be manually backed up similar to qm.ini files today
- Logs
 - Located in MQ data path
 - `/var/mqm/qmgrs/QM1/amqp.stdout` and `/var/mqm/qmgrs/QM1/amqp.stderr`
 - `/var/mqm/qmgrs/errors/amqp_*.log`
 - `/var/mqm/trace/amqp_*.trc` (start/end trace using `strmqtrc/endmqtrc`)

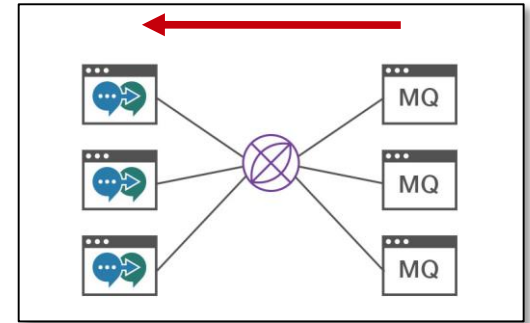
Interoperability

- AMQP publisher to MQ consumer
 - MQMD PutAppIType always set to MQAT_AMQP
 - Some AMQP attributes → MQMD
 - Some AMQP attributes → MQ message properties
 - All AMQP application properties → MQ message properties
 - Simple AMQP binary payload → MQFMT_NONE message
 - Simple AMQP string payload → MQFMT_STRING message
 - All other AMQP payloads → MQFMT_AMQP



Interoperability

- MQ publisher to AMQP consumer
 - Some MQMD fields → AMQP headers
 - Some MQMD fields → AMQP properties
 - All MQ message properties → AMQP application properties
 - MQFMT_NONE message → single AMQP binary data payload
 - MQFMT_STRING message → single AMQP string data payload
 - MQFMT_AMQP message → copy to payload section



Note: All MQ messages are got with MQGMO_CONVERT to convert string data to UTF8

Detail (for reading on the plane home) – AMQP to MQ

Some AMQP **headers** are set as MQMD fields:

AMQP header.ttl	set on MQ message as MQMD.expiry (converted to 10ths of a second)
AMQP header.priority	set on MQ message as MQMD.priority (max value of 9)
AMQP properties.correlation-id	set on MQ message as MQMD.correlid

All AMQP **headers** are set as MQ message properties with a mapped name:

header.durable	set on MQ message as MQ propertyAMQPDurable
header.priority	set on MQ message as MQ propertyAMQPPriority
header.ttl	set on MQ message as MQ propertyAMQPTtl
header.first-acquirer	set on MQ message as MQ propertyAMQPFIRSTAcquirer
header.delivery-count	set on MQ message as MQ propertyAMQPDeliveryCount

All AMQP **properties** are also set as MQ message properties, e.g.

properties.user-id	set on MQ message as MQ propertyAMQPUserId
properties.to	set on MQ message as MQ propertyAMQPTo
properties.subject	set on MQ message as MQ propertyAMQPSubject
properties.reply-to	set on MQ message as MQ propertyAMQPReplyTo
properties.content-type	set on MQ message as MQ propertyAMQPContentType
properties.content-encoding	set on MQ message as MQ propertyAMQPContentEncoding
properties.creation-time	set on MQ message as MQ propertyAMQPCreationTime
properties.group-id	set on MQ message as MQ propertyAMQPGroupId
properties.message-id	set on MQ message as MQ propertyAMQPMessageId
Properties.group-sequence	set on MQ message as MQ propertyAMQPGroupSequence
Properties.absolute-expiry-time	set on MQ message as MQ propertyAMQPAbsoluteExpiryTime
Properties.reply-to-group-id	set on MQ message as MQ propertyAMQPReplyToGroupId

Finally, all AMQP **application-properties** are copied into the MQ message properties in the user space (usr.*) using similar naming conventions with some restrictions on the property length, characters used, and certain keyword restrictions e.g. “JMS”.

Detail (for reading on the plane home) – MQ to AMQP

The following **MQMD fields** are set on the AMQP message as headers, **if and only if** the value in the MQ message is not the same as the AMQP default value for that property.

MQMD.persistence	set on AMQP message as header.durable
MQMD.expiry	set on AMQP message as header.ttl
MQMD.priority	set on AMQP message as header.priority

Some **MQ message properties**, if they exist, are set as AMQP headers:

MQ message property AMQPFirstAcquirer	set on AMQP message as header.first-acquirer
MQ message property AMQPDeliveryCount	set on AMQP message as header.delivery-count

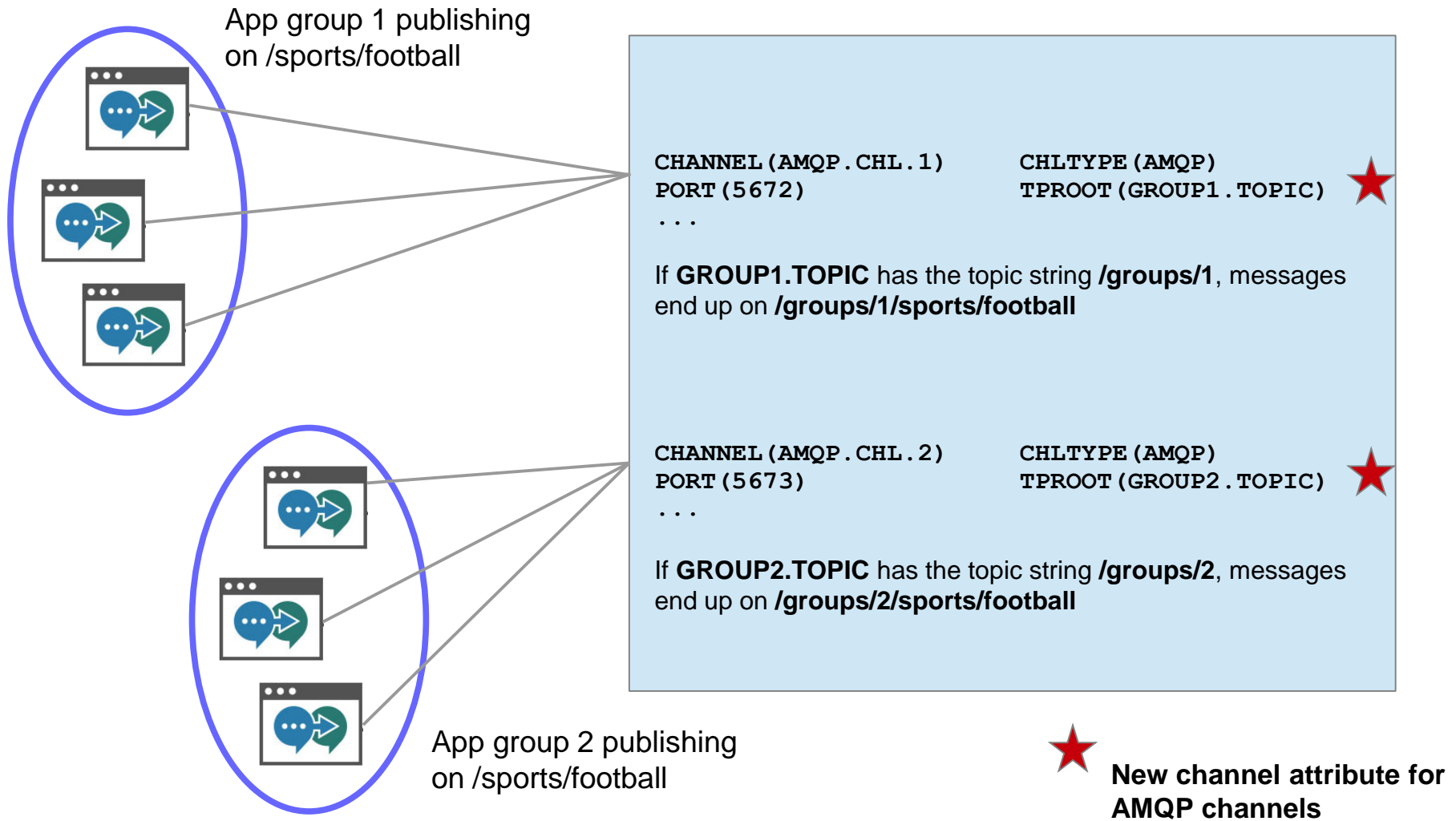
Some **MQ message properties** are set as AMQP properties:

AMQPUserId	set on the AMQP message as properties.user-id
AMQPTo	set on the AMQP message as properties.to
AMQPSubject	set on the AMQP message as properties.subject
AMQPReplyTo	set on the AMQP message as properties.reply-to
AMQPContentType	set on the AMQP message as properties.content-type
AMQPContentEncoding	set on the AMQP message as properties.content-encoding
AMQPCreationTime	set on the AMQP message as properties.creation-time
AMQPGroupId	set on the AMQP message as properties.group-id
AMQPMessageId	set on the AMQP message as properties.message-id
AMQPGroupSequence	set on the AMQP message as properties.group-sequence
AMQPAbsoluteExpiryTime	set on the AMQP message as properties.absolute-expiry-time
AMQPReplyToGroupId	set on the AMQP message as properties.reply-to-group-id

Finally, all **MQ message properties in the user space** (i.e. those which start usr.*) are copied into the AMQP message as application properties.

Managing MQ Light in an MQ Environment

Queue manager



Agenda

- What is MQ Light?
- The MQ Light API
- Running MQ Light apps in Bluemix
- IBM MQ support for MQ Light
- **Demo**

Thank You - Questions?



Related session:

- **Hybrid Messaging with IBM Bluemix**
 - **Tuesday 8.30am (this room)**

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.