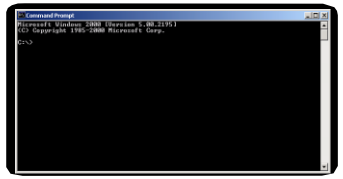
Open Source Monitoring for IBM MQ

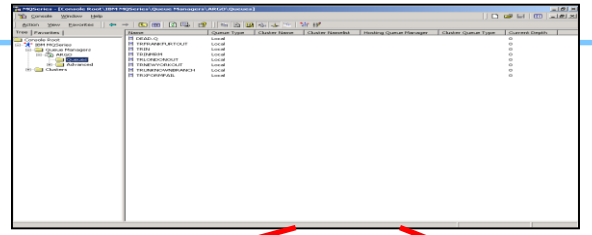
Mark Taylor
marke_taylor@uk.ibm.com

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.**

MQ Administration





MQSC commands

OS commands

PCF commands

OS configuration

ini files

Queue Manager

Events (async)

Status (polled)

Log files

These are all documented and supported interfaces – it has encouraged a management ecosystem over many years

IBM MQ - MQSC

Old Example: AIX smit panels

- Command line interface
- V8 enhanced runmqsc
  - Make it world-executable
  - Enable direct client-connection
- MQSC intended for human consumption
  - Parsable by eye, less easy in programs
  - For example, **DESCR('This is 'a' description with quote & paren(')**
  - No guaranteed ordering in runmqsc, two-column output
- Despite awkwardness, basis for many script-based admin tools
  - echo "DISPLAY Q(X) IPPROCS" | runmqsc QM1
- Same commands – different front-end (CSQUTIL) – for z/OS

IBM MQ - PCF

- A "self-describing" MQ message used for administrative operations
- Your programs can send commands and get responses using PCF
  - Equivalent to "DISPLAY QSTATUS" or "ALTER CHANNEL"
- MQ emits events in PCF format
  - "Queue is getting full"
- PCF intended for programs – usually C or Java
  - Can tell exactly what the parameter is for, its length and value
  - But cannot easily be scripted
- Approximately one-one mapping between MQSC commands and PCF
- Remember that PCF invented before formats like JSON or XML
  - And there are many MQ apps that are built on PCF

An event message

```
****   Message length - 300 of 300 bytes ***

00000000:  0000 0007 0000 0024 0000 0003 0000 0063 '.......$.......c'
00000010:  0000 0001 0000 0001 0000 0000 0000 096C '...............l'
00000020:  0000 0002 0000 0014 0000 0010 0000 1F41 '...............A'
00000030:  0000 0004 0000 0004 0000 0020 0000 0BE5 '........... ...å'
00000040:  0000 0333 0000 000C 6D65 7461 796C 6F72 '...3....metaylor'
00000050:  2020 2020 0000 0003 0000 0010 0000 03F3 '    ...........ó'
00000060:  0000 0001 0000 0004 0000 0044 0000 0BE7 '...........D...ç'
00000070:  0000 0333 0000 0030 5638 3030 335F 4120 '...3...0V8003_A '
00000080:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000090:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
000000A0:  2020 2020 2020 2020 0000 0003 0000 0010 '        ........'
000000B0:  0000 03FD 0000 005A 0000 0014 0000 0010 '...ý...Z........'
000000C0:  0000 1F42 0000 0004 0000 0004 0000 0018 '...B............'
000000D0:  0000 0BFB 0000 0000 0000 0001 5800 0000 '...û........X...'
000000E0:  0000 0003 0000 0010 0000 03F8 0000 0001 '...........ø....'
000000F0:  0000 0006 0000 0024 0000 0BF9 0000 0000 '.......$...ù....'
00000100:  0000 0001 0000 0008 6D65 7461 796C 6F72 '........metaylor'
00000110:  0000 0000 0000 0005 0000 0018 0000 045C '...............\'
00000120:  0000 0002 0000 000B 0000 0009           '............    '
```

An event message

```
****   Message length - 300 of 300 bytes ***

00000000:  0000 0007 0000 0024 0000 0003 0000 0063 '.......$.......c'
00000010:  0000 0001 0000 0001 0000 0000 0000 096C '...............l'
00000020:  0000 0002 0000 0014 0000 0010 0000 1F41 '...............A'
00000030:  0000 0004 0000 0004 0000 0020 0000 0BE5 '........... ...å'
00000040:  0000 0333 0000 000C 6D65 7461 796C 6F72 '...3....metaylor'
00000050:  2020 2020 0000 0003 0000 0010 0000 03F3 '    ...........ó'
00000060:  0000 0001 0000 0004 0000 0044 0000 0BE7 '...........D...ç'
00000070:  0000 0333 0000 0030 5638 3030 335F 4120 '...3...0V8003_A '
00000080:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000090:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
000000A0:  2020 2020 2020 2020 0000 0003 0000 0010 '        ........'
000000B0:  0000 03FD 0000 005A 0000 0014 0000 0010 '...ý...Z........'
000000C0:  0000 1F42 0000 0004                     '...B............'
000000D0:                                          '...û........X...'
000000E0:  0000 0003 0000 0010 0000 03F8 0000 0001 '...........ø....'
000000F0:  0000 0006 0000 0024 0000 0BF9 0000 0000 '.......$...ù....'
00000100:  0000 0001 0000 0008 6D65 7461 796C 6F72 '........metaylor'
00000110:  0000 0000 0000 0005 0000 0018 0000 045C '...............\'
00000120:  0000 0002 0000 000B 0000 0009           '............    '
```

TYPE (cfst)	LEN (24)		
PARM (MQCA…)	CCSID (0)	LEN (1)	DATA

Event formatting C sample in V8.0.0.4

- No sample previously shipped to format all "standard" events
  - Authorisation, queue full, service interval, command/config etc
  - Other samples are available for acct/stats, activity reports
  - Several SupportPacs but product only has out-of-date source code in the KC
- The **amqsevt** program formats events into readable English-ish text
  - Option to stay with full MQI constant name instead of making it look nice
  - Uses MQCB to read from multiple event queues. No polling required
  - Can connect as client to any remote queue manager including z/OS
  - Source code included
- Includes C header file to help convert MQI numbers to strings
  - Similar to Java MQConstants.lookup() capability for all sets of constants

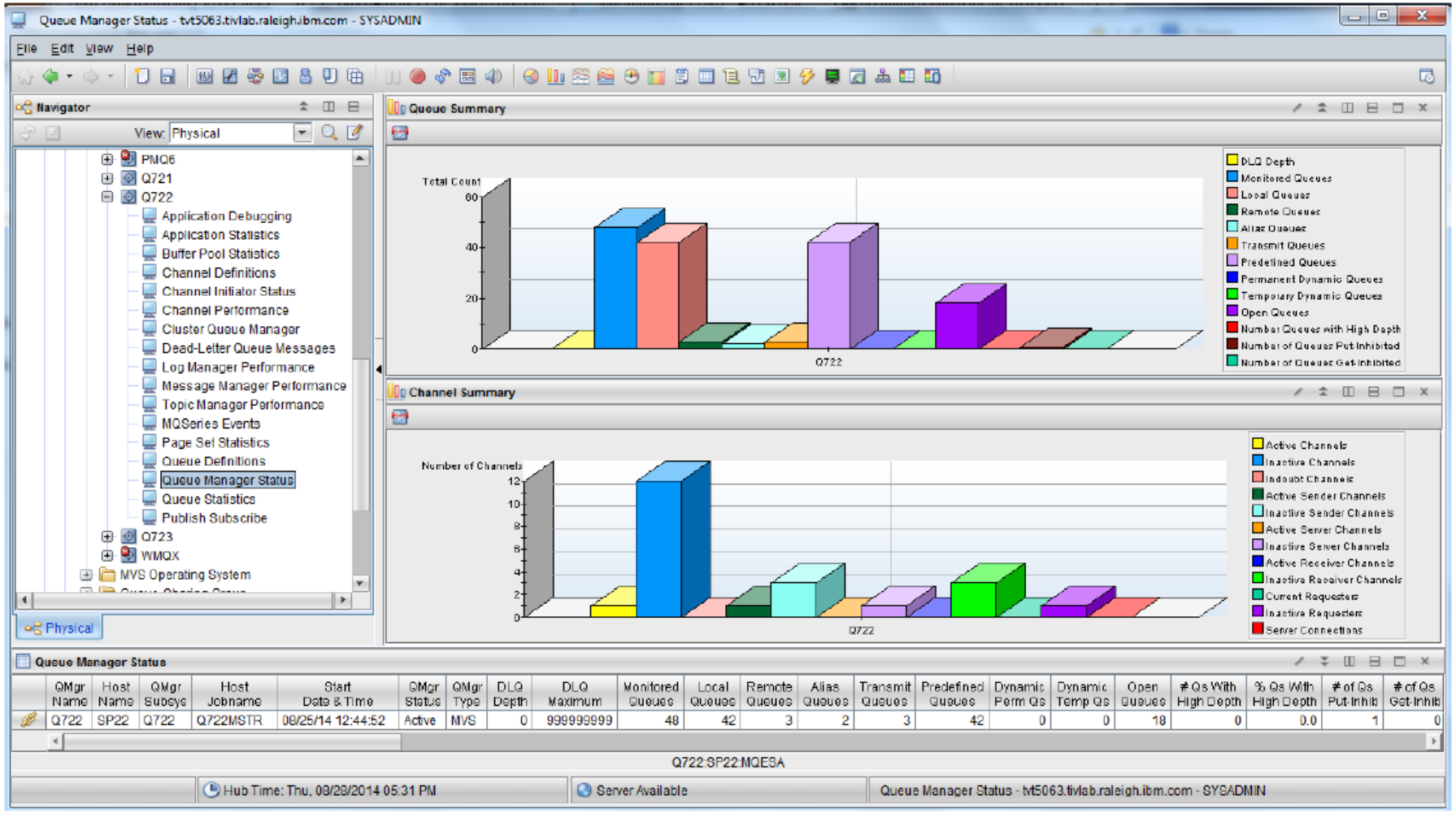
```
printf("Error is %s\n",MQRC_STR(2035));
```

An event message decoded

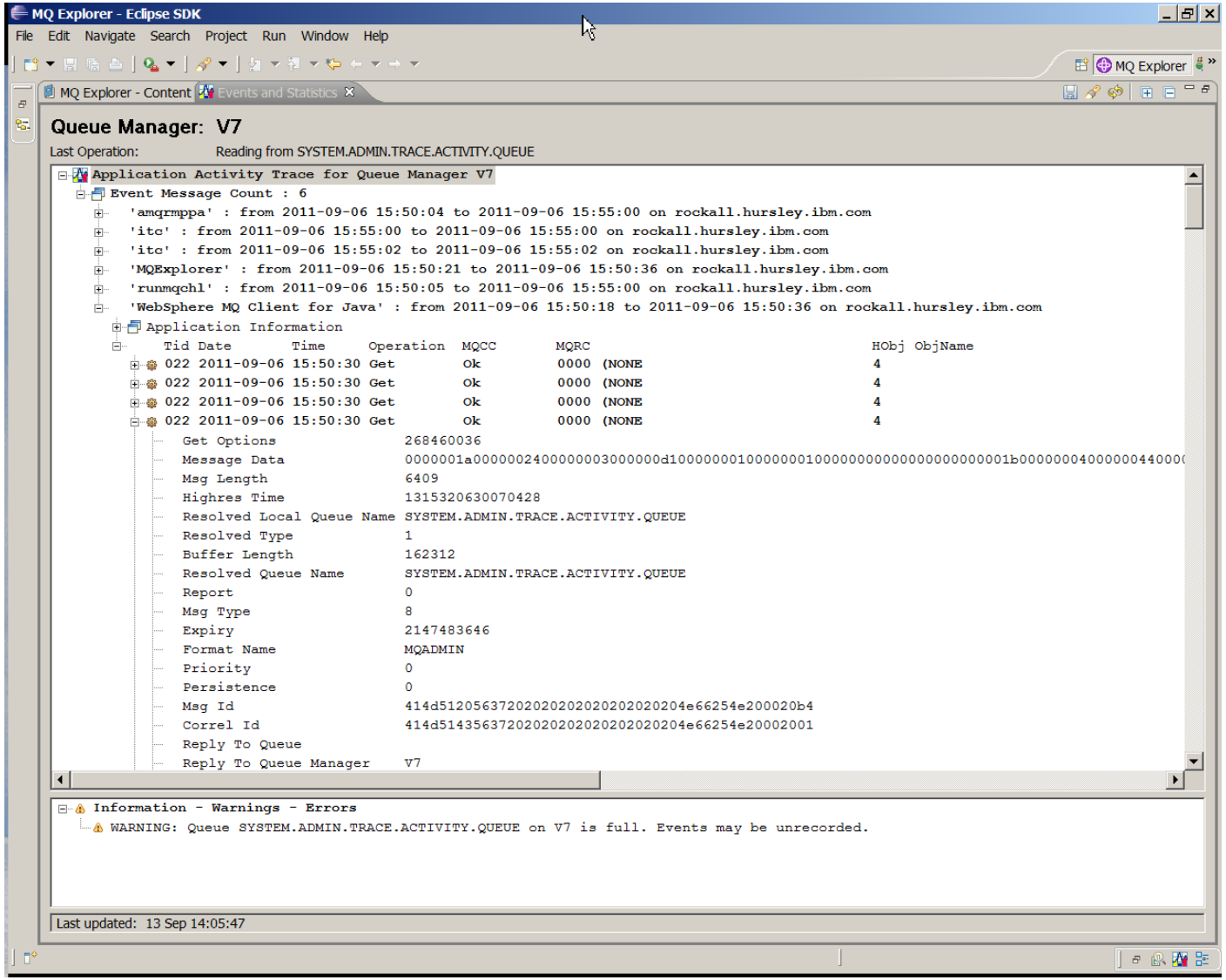
```
Event Type                    : Command Event
Reason                        : Command MQSC
Event created                 : 2015/06/03 13:28:20.51 GMT
Correlation ID : 414D5120563830303 35F412020202020556F00F120001E05
  COMMAND CONTEXT
    Event User Id             : metaylor
    Event Origin              : Console
    Event Queue Mgr           : V8003_A
    Command                   : Set Auth Rec
  COMMAND DATA
    Auth Profile Name         : X
    Object Type               : Queue
    Principal Entity Names: metaylor
    Auth Add Auths            : Output
                              : Input
```

Third-party solutions

- Many vendor products – this screenshot from ITCAM/Omegamon



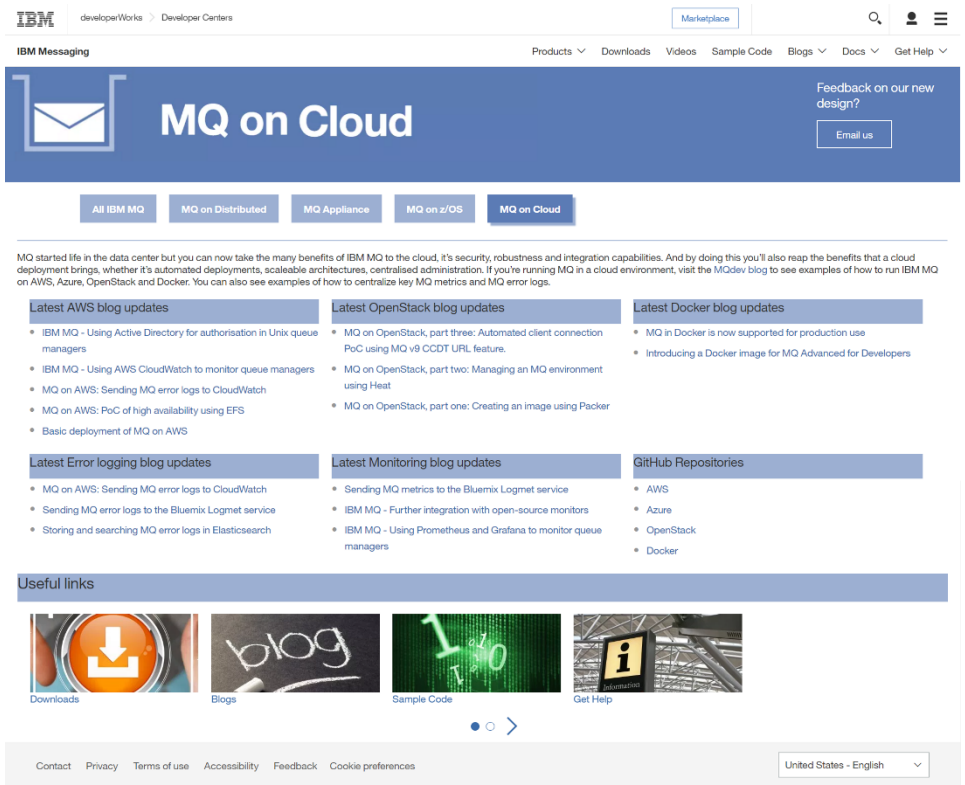
Application Activity inside MQ Explorer using MS0P



Many people now using different tools

- Because they are using those tools for other products
- And because MQ is being used in more environments

- Therefore MQ has to be able to be integrated with them



https://developer.ibm.com/messaging/mq-on-cloud/

Decided to demonstrate MQ integration

- Using the V9 resource statistics data
- Feeding a variety of monitoring tools
- And doing it in public – Github, blog articles etc
  - See github.com/ibm-messaging/mq-golang
  - Video at youtube.com/watch?v=Pi_jHCiqTgU

System Monitoring with V9

- More statistics available via a pub/sub model

- Includes CPU and Disk usage
  - As well as many MQ statistics
  - Not full replacement for accouting/statistics events but many key values

- Subscribe to meta-topic to learn which classes of statistics are available
  - $SYS/MQ/INFO/QMGR/<qmgr>/Monitor/METADATA/CLASSES
  - Then subscribe to specific topics
  - See amqsrua sample program

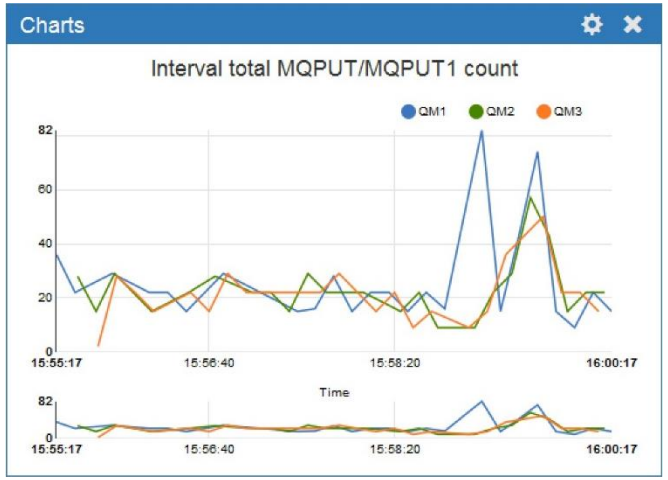
- Distributed platforms only

RFE 71123

System Monitoring Example

```
$ amqsrua -m V9000_A
CPU : Platform central processing units
DISK : Platform persistent data stores
STATMQI : API usage statistics
STATQ : API per-queue usage statistics
Enter Class selection
==> CPU
SystemSummary : CPU performance - platform wide
QMgrSummary : CPU performance - running queue manager
Enter Type selection
==> SystemSummary
Publication received PutDate:20160411 PutTime:10465573
User CPU time percentage 0.01%
System CPU time percentage 1.30%
CPU load - one minute average 8.00
CPU load - five minute average 7.50
CPU load - fifteen minute average 7.30
RAM free percentage 2.02%
RAM total bytes 8192MB
```

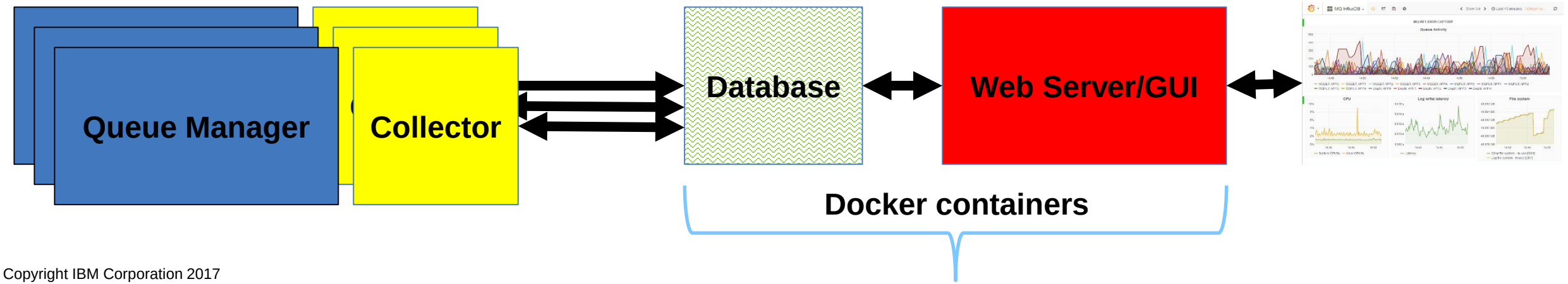
This capability already underpins the charting in the MQ Appliance WebUI



Monitoring Architecture

- Architecture is split – database and user interface
  - The database is usually a "time-series" DB, not traditional SQL
  - Designed and optimised for {timestamp, metric, value} storage and queries

- These databases include Prometheus, InfluxDB, OpenTSDB
- Collection architecture may have intermediate layers – collectd



Started with Prometheus

- Seemed to be one of the most popular
- Which does have its own limited GUI
- Model is "pull" – calls a collector program at intervals via http
  - Most other DBs are "push" where collector sends to DB at interval
- Standard API for getting data to Prometheus is in Go
  - And we had no Go API for MQ …

The Go API for MQ

- So first off, I had to create a new language binding
  - Based on full MQI rather than a "simplified" version
  - But not all function implemented
  - Trying to make it look natural to Go programmers

```
if err == nil {
  putmqmd := ibmmq.NewMQMD()
  pmo     := ibmmq.NewMQPMO()
  pmo.Options = ibmmq.MQPMO_SYNCPOINT | ibmmq.MQPMO_NEW_MSG_ID
  putmqmd.Format = "MQSTR"
  msgData := "Hello from Go"
  buffer := []byte(msgData)
  err = qObject.Put(putmqmd, pmo, buffer)
  if err != nil {
    fmt.Println(err)
  }
}
```

Working with the Go API

- Ensured bindings had functions I needed including PCF generation and parsing
- Started with RESET QSTATS as PoC for hooking to Prometheus
  - But rapidly went to full amqsrua-style metadata subscriptions
- After first release of Go bindings, extensions made for more verbs and options
  - Including client connections via MQCNO/MQCD structures
  - MQSET
  - Information on building for Windows
- Still subject to change

Collector configurations

- Collector subscribes to all data for qmgr (cpu, disk etc) and nominated queues
  - Command line parameters name the queues with wildcards
- Started via MQ Service definition and shell script
- Can connect as client to remote queue managers including MQ appliance
  - Any system that supports the resource statistics
  - One collector instance per queue manager

```
/usr/local/bin/mqgo/mq_prometheus -ibmmq.queueManager=QM1
    -ibmmq.monitoredQueues=APP.*,MYQ.*
    -ibmmq.httpListenPort=9157
    -log.level=error
```

Prometheus configuration

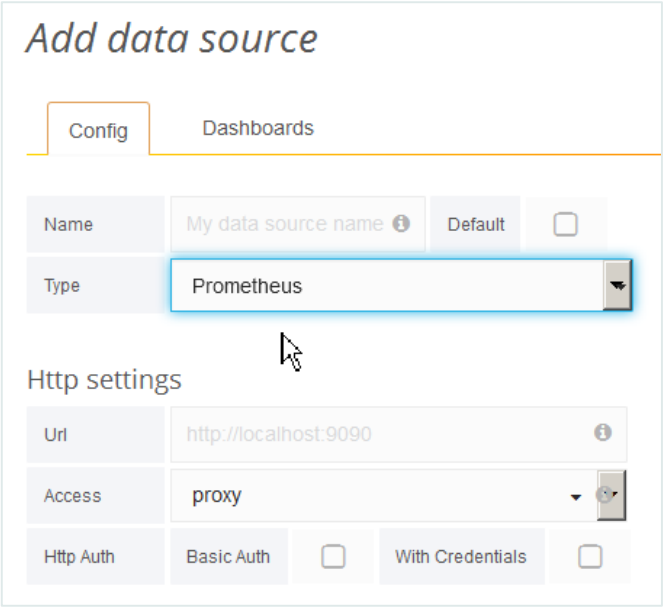
- File prometheus.yml defines configuration
  - Built copy of this into Docker image
  - Two targets for two collectors on this system (queue manager, Salesforce bridge)

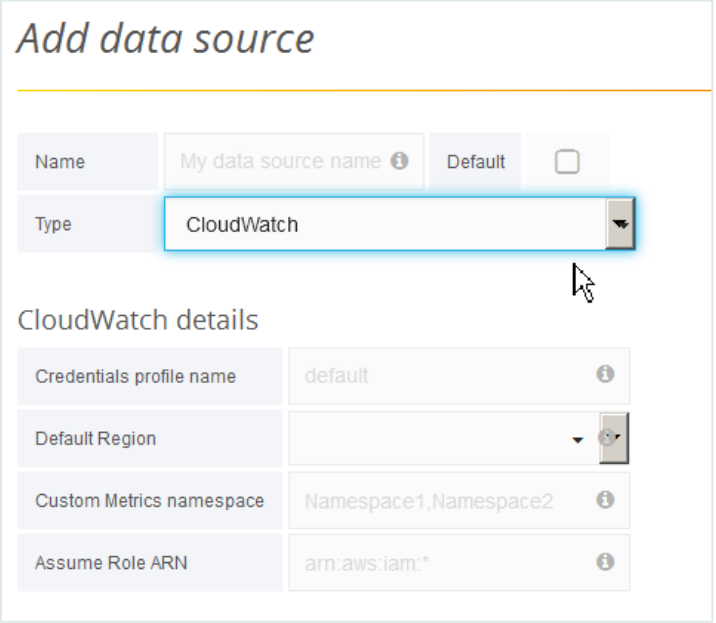
```
scrape_configs:
  # Job name added as label `job=<job_name>` to any timeseries scraped from this config
  - job_name: 'prometheus'
    # Override the default and scrape targets from this job every 5 seconds.
    scrape_interval: 5s
    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'ibmmq'
    scrape_interval: 5s
    static_configs:
       - targets: ['klein.hursley.ibm.com:9157', 'klein.hursley.ibm.com:9158']
```

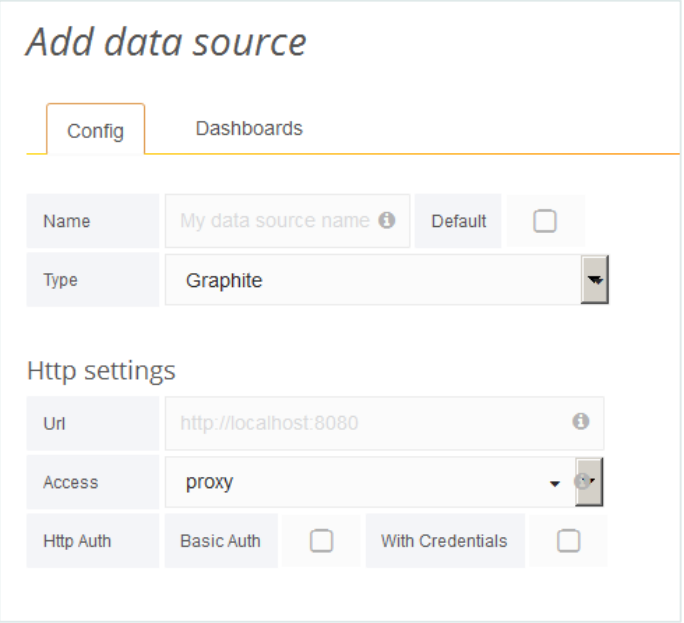
Grafana

- Although Prometheus has a GUI it is not very sophisticated

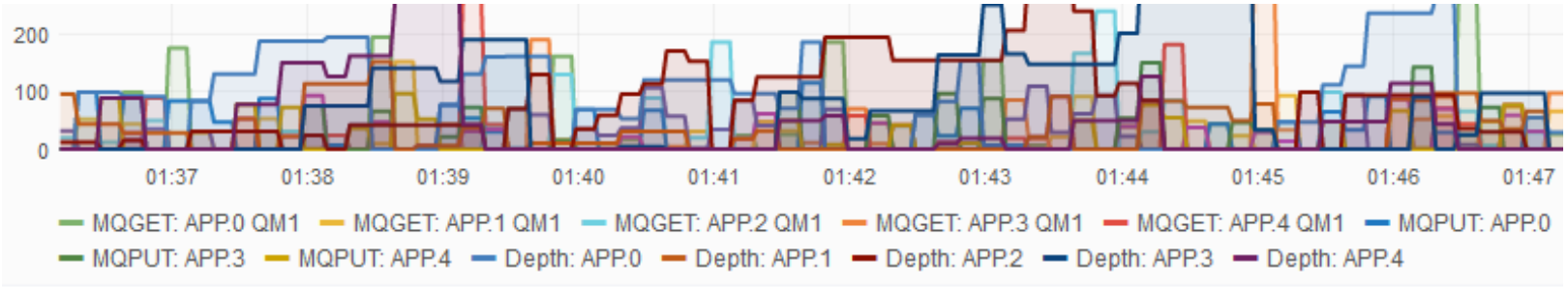
- Instead, prefer to use Grafana as visualisation tool
  - Supports many different backend databases
  - Understands the metric names, query capabilities etc of each

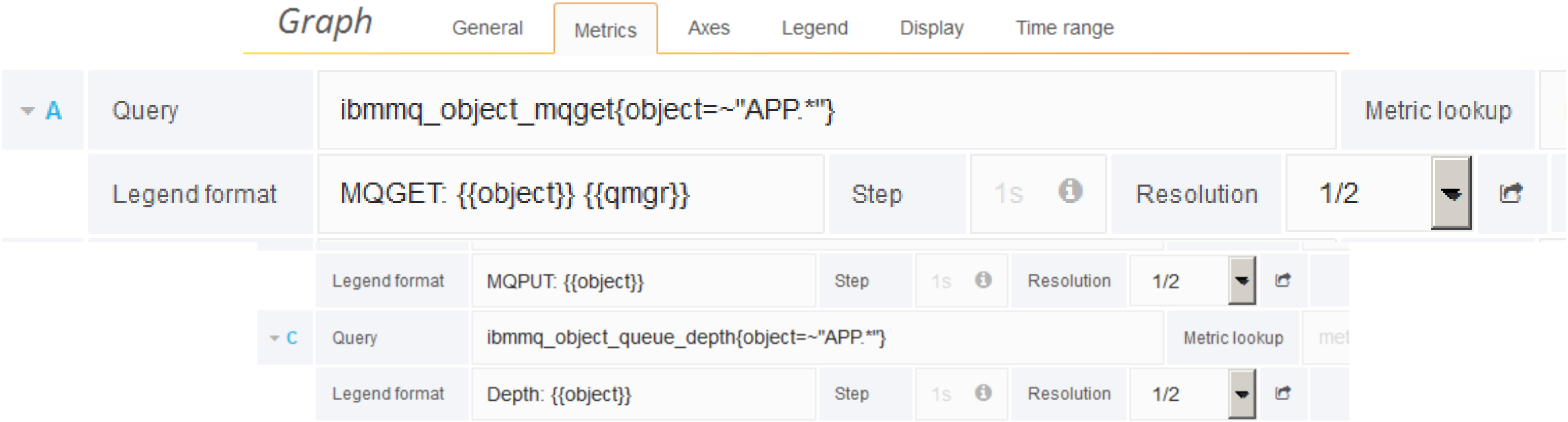




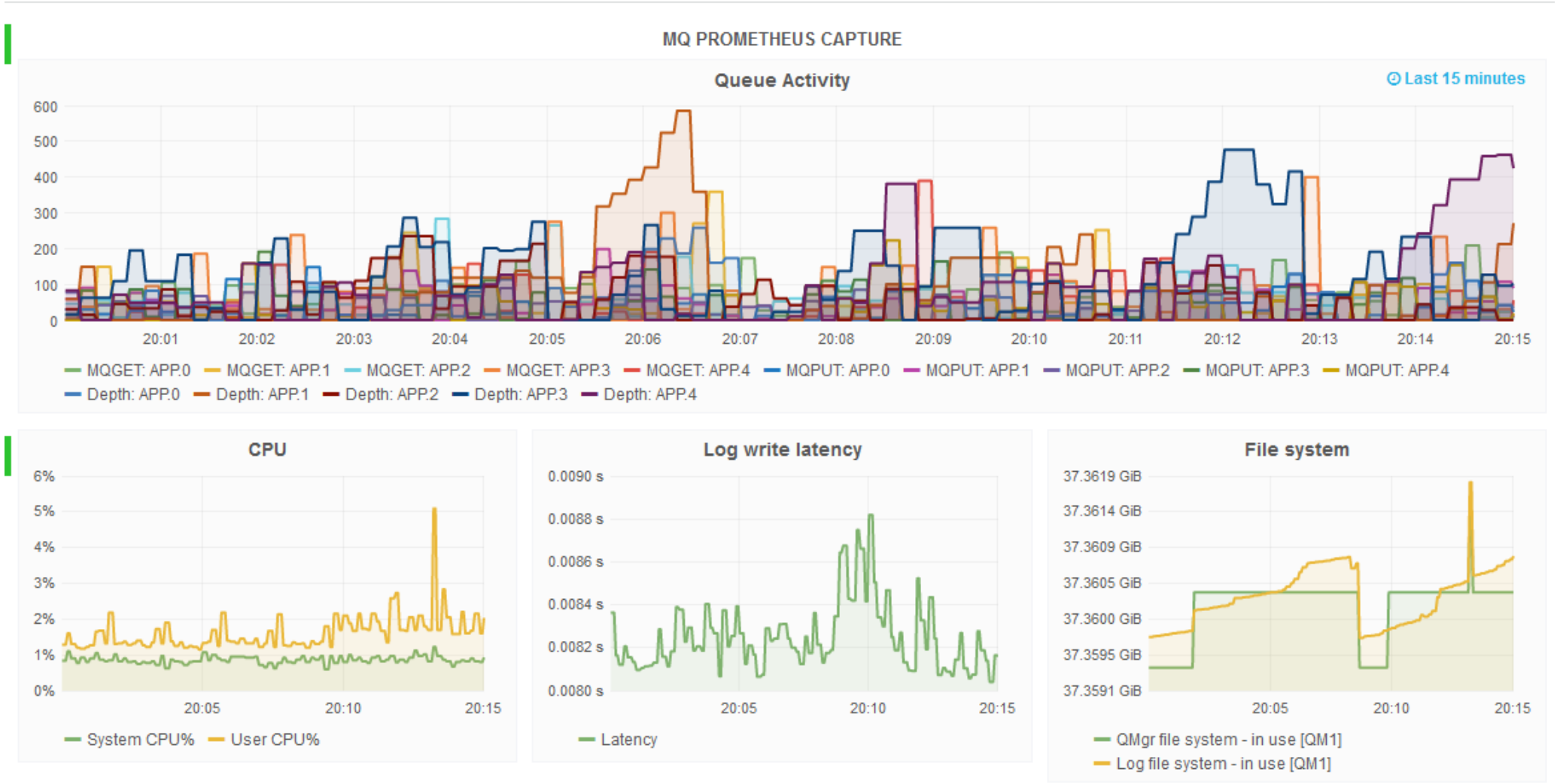


Accessing queue stats from Prometheus in Grafana





Grafana dashboard

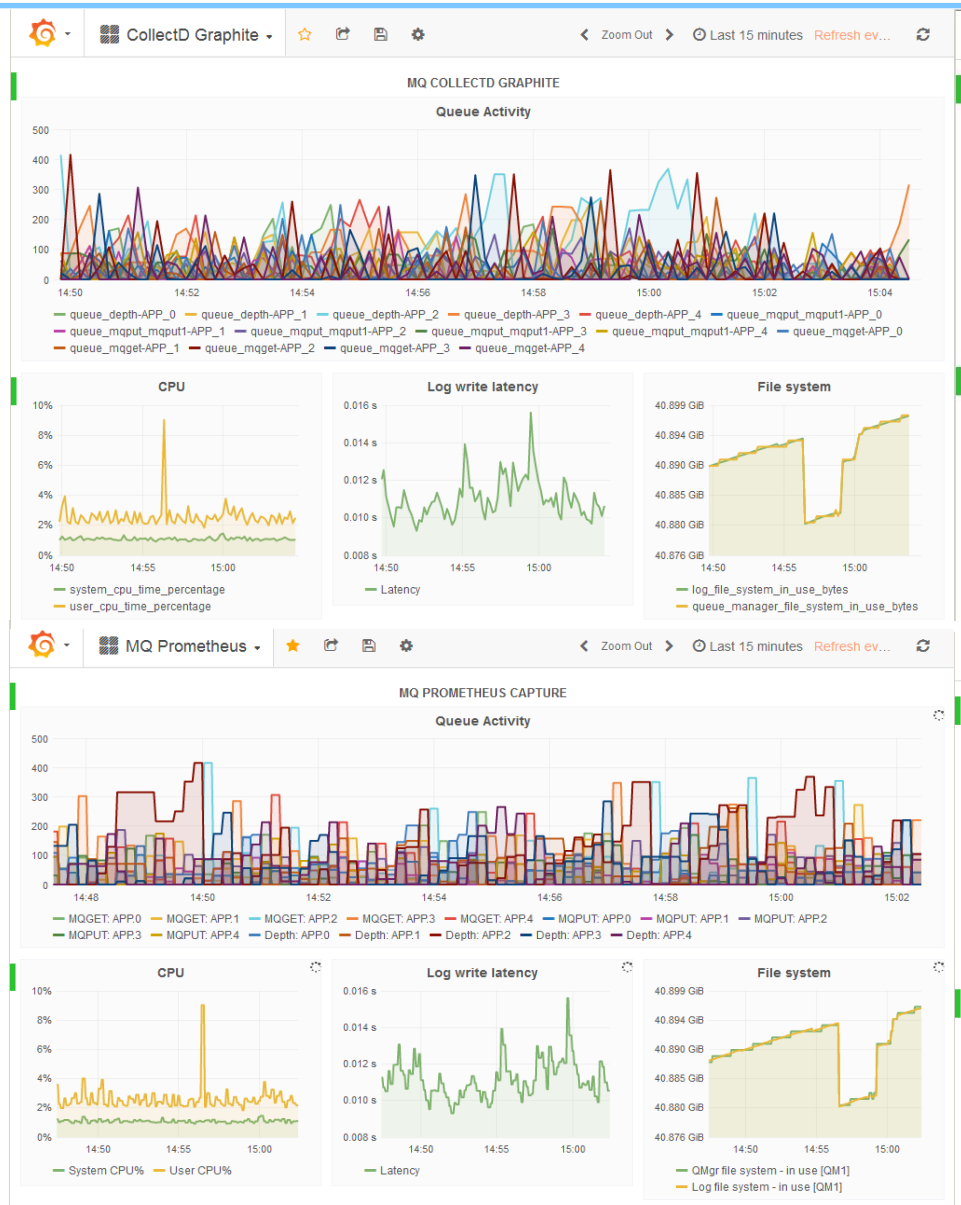


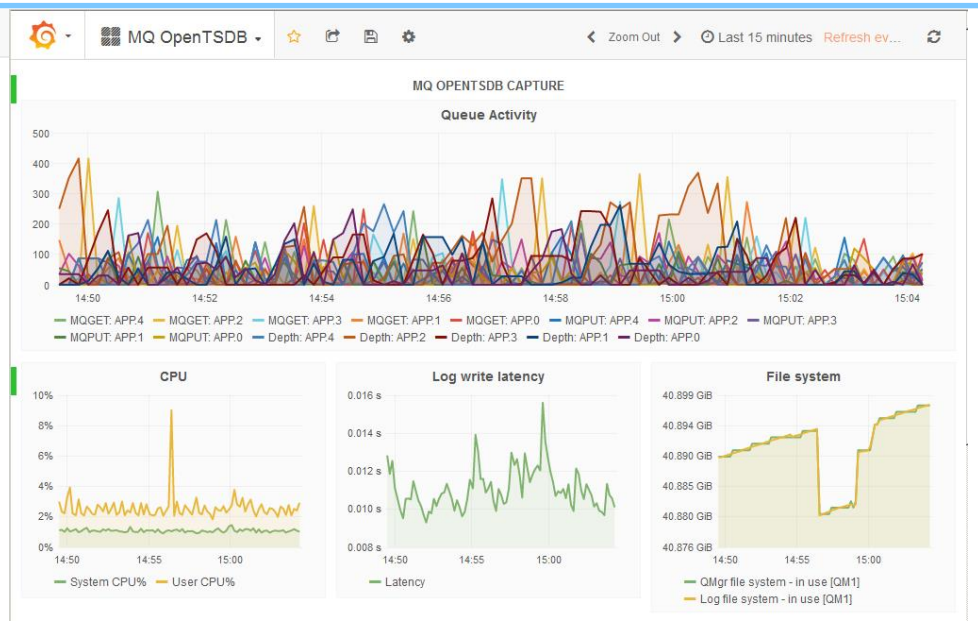
Then added more variants

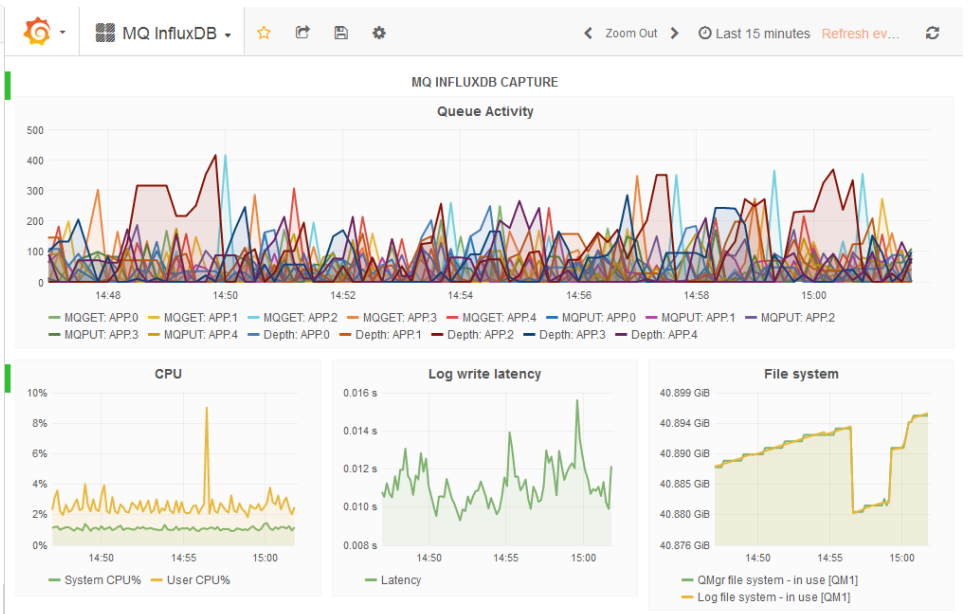
- Rapidly added support for influx, opentsdb
  - Different collectors with slightly different parameters
- Graphite is another database, but fed via collectd
  - collectd can also feed the database used by bluemix
- Also added an AWS collector for CloudWatch
- Generic JSON formatting

```
{ "collectionTime" : {
      "timeStamp" : "2016-11-07-T15:00:55Z"
      "epoch" : 1478527255   },
   "points" : [
      { "queueManager" : "QM1", "ramTotalBytes" : 15515735206 },
      { "queueManager" : "QM1", "userCpuTimePercentage" : 1.33 }
   ]
}
```

Four equivalent Grafana dashboards

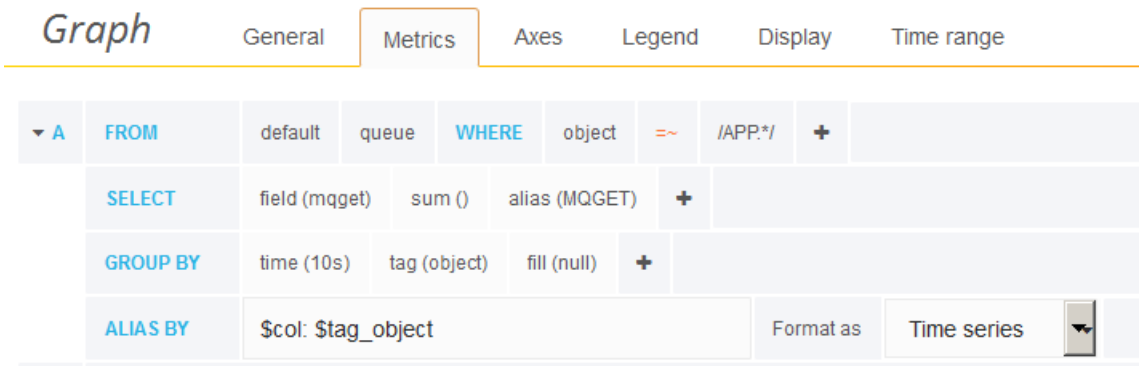






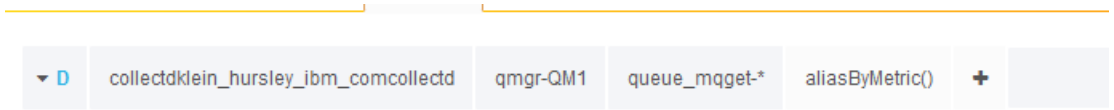
Metric Queries

- Influx

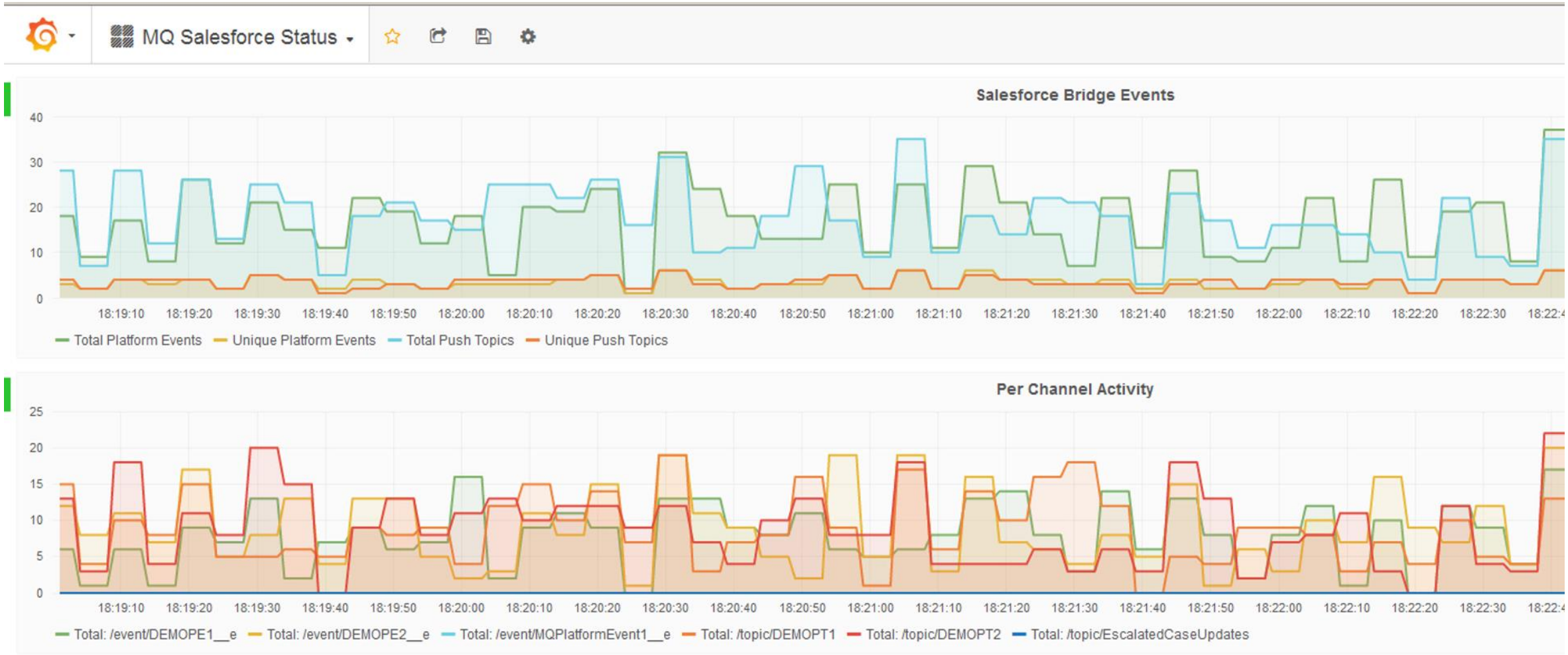


- OpenTSDB

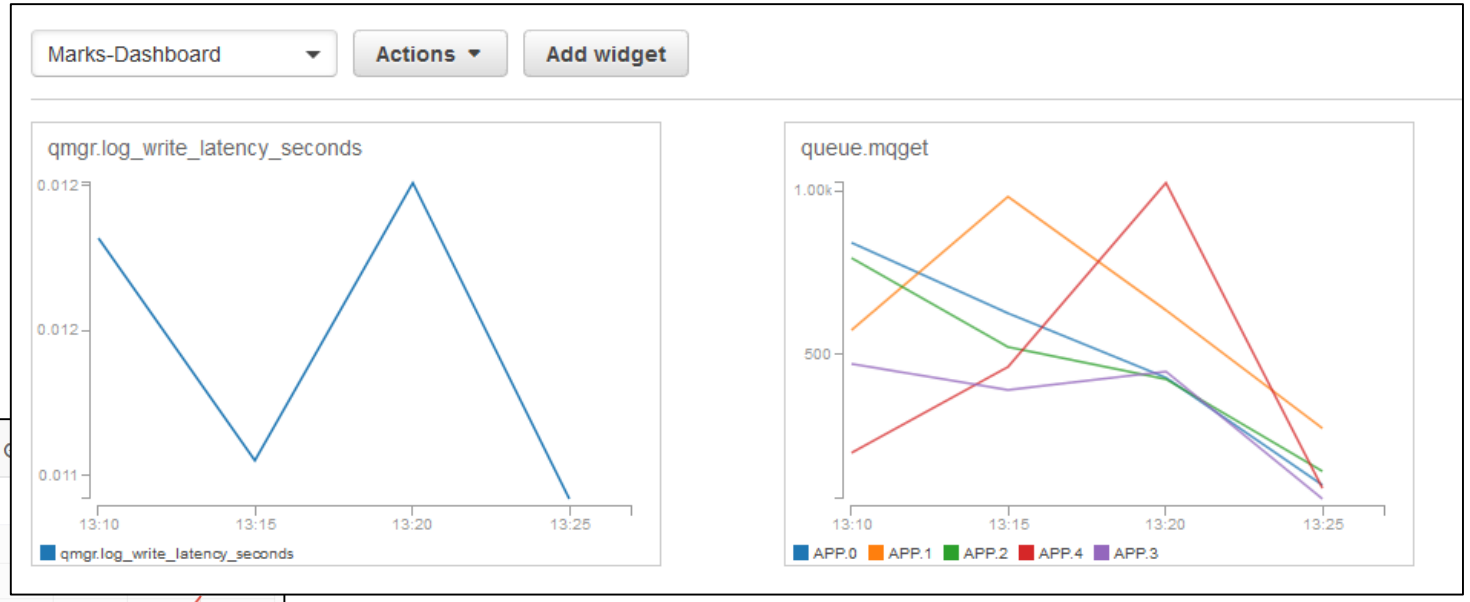
- Graphite/Collectd

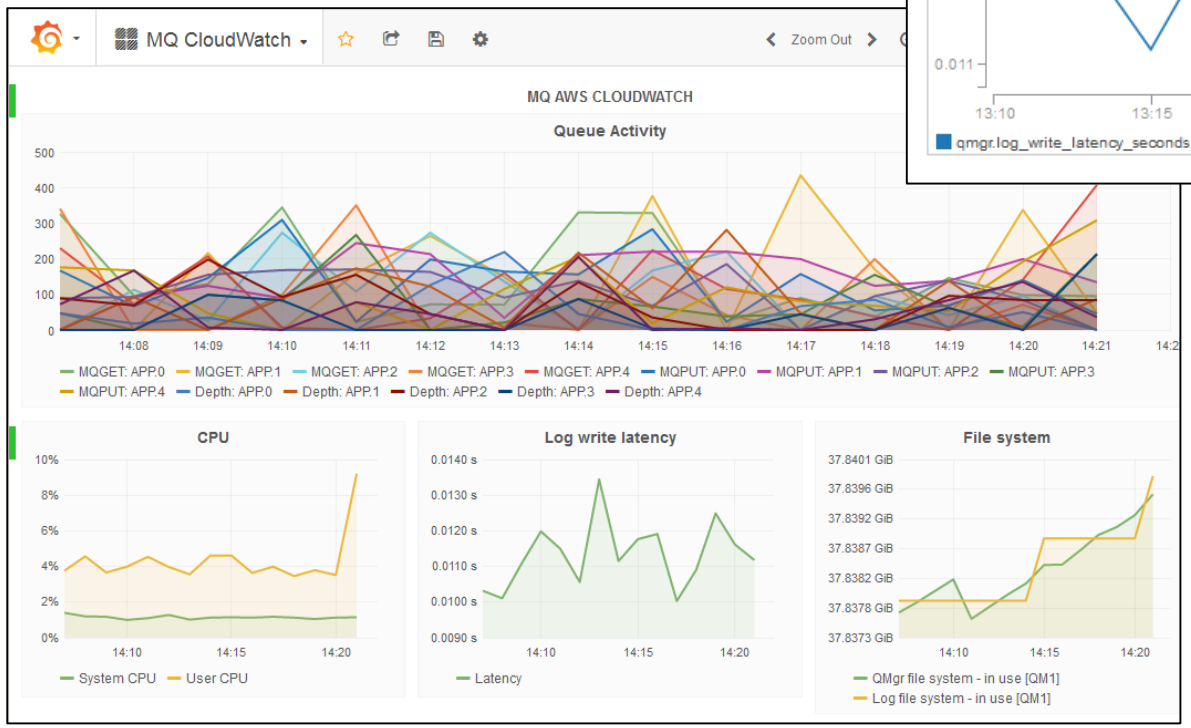


More resources – the MQ Bridge to Salesforce



AWS Cloudwatch





What are differences? Which is best?

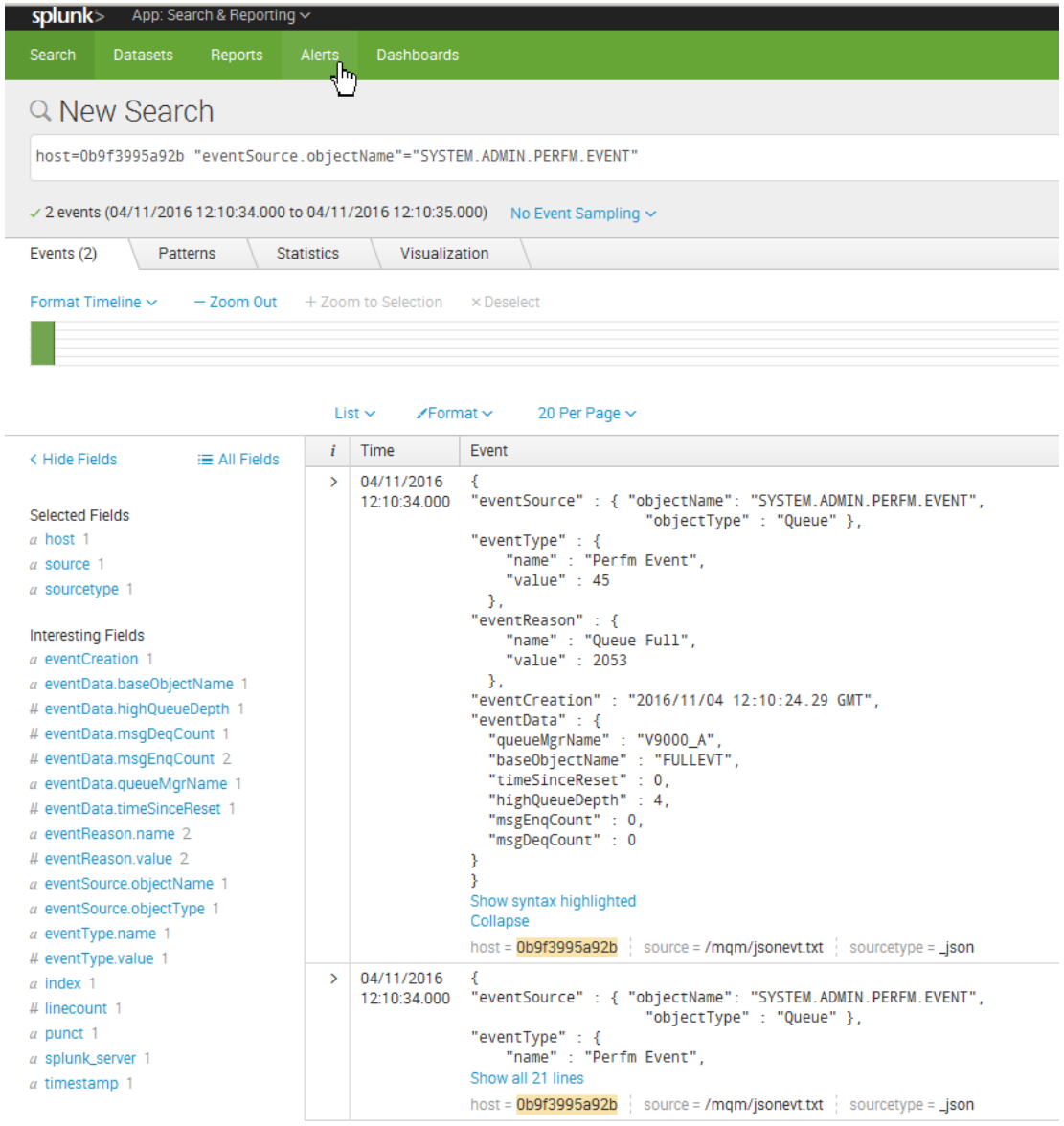
- Differences are generally in
  - The names and formats of metrics ("ibmmq_mqobject_mqget")
  - Naming for individual resources such as the queue name
  - Query capabilities to select and display chosen metrics
    - Can you use wildcards on object names
  - Creating labels on graphs
    - Can it be automatic based on the query?

- The best is going to be whatever you are already using!
  - But I found the Prometheus/Grafana combination to be flexible and usable

Processing other MQ events

- Already shown amqsevt as shipped in MQ
- Also available is JSON variety
  - Available at https://gist.github.com/ibmmqmet/fabd57f4ff5c6e1b8d78284b2bc65f9e
- Used to feed JSON consumers such as splunk

MQ events in splunk



Error log collection

- MQ error logs can also be fed to monitors
  - Define filters to extract interesting information from the error messages

- Several articles published on using Bluemix (Kibana) and Cloudwatch

https://www.ibm.com/developerworks/community/blogs/messaging/entry/Sending_MQ_logs_to_the_Bluemix_Logmet_service?lang=en

https://www.ibm.com/developerworks/community/blogs/messaging/entry/mq_aws_cloudwatch_logs?lang=en

https://www.ibm.com/developerworks/community/blogs/messaging/entry/Monitoring_and_Exploring_IBM_MQ_AMQERR_logs_on_Bluemix_using_logmet?lang=en

Analysing MQ error logs in Bluemix

IBM Bluemix Monitoring & Logging | Grafana | Kibana 3 | Kibana 4

kibana — Discover | Visualize | Dashboard | Settings — 5 minutes | Last 24 hours

IBM MQ AMQERR Dashboard

Overview

IBM MQ *AMQERR log entries* dashboard overview

> *This dashboard has a collection of Visualizations that present aspects of IBM MQ queue manager log entries in useful ways.*

The Visualizations present the entries in following ways:

Area chart. Number of log entries per queue manager in 30 min intervals

Shows the **rate at which the queue managers are writing entries to their AMQERR01.LOG** file. Information is shown for only the 25 queue managers that logged the most entries.

- *Click a line in the split bar chart area to view activity for one queue manager.*

Bar chart. Number of entries per logged entry code (AMQnnn), split by queue manager

Ranks the most frequently occurring entry codes in your IBM MQ network. Information is shown for only the 25 queue managers that logged the most entries.

- *Click a split bar (queue manager color) to view activity for one queue manager.*

Data table. Host, queue manager name, and log entry count

Shows the number of log entries per queue manager and host.

Information is shown for only the 25 queue managers that logged the most entries. If multiple queue managers are running on a single host, then the list shows data from all the queue managers on that host.

- *Click a queue manager or host name to view information about their activity.*

Data table. A complete list of log files for all queue managers and hosts in an interval.

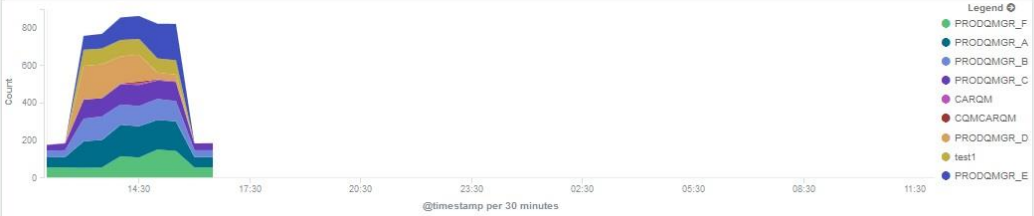
*A list of all **AMQ log files** that are used for generating the chart and table data in the three other visualizations in this dashboard. The most significant fields are displayed.*

- *Click a row in the table to view more information about the log.*

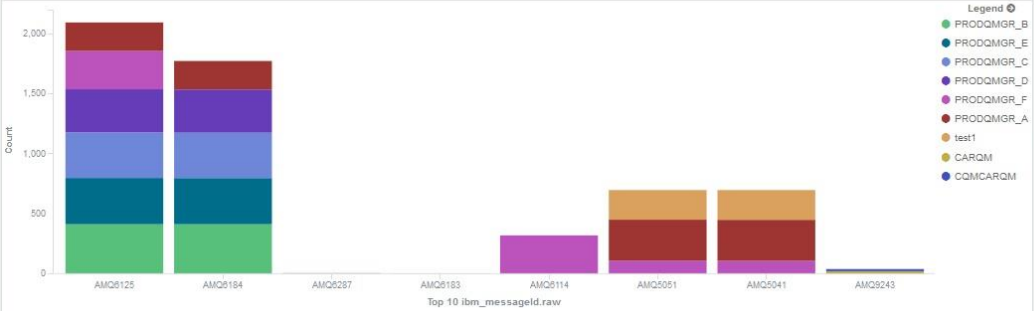
Note

- If you cannot find a queue manager or host in the Visualizations, it might be because they are not among the 25 queue managers or hosts with the most log entries.

Log entries per queue manager in 30 min intervals



Number of entries per logged entry code (AMQnnn), split by queue manager



Host, queue manager name, and log entry count

Top 25 host.raw	Top 25 ibm_qmgrName.raw	Count
Server1	PRODQMGR_A	1,162
Server1	PRODQMGR_B	825
Server1	PRODQMGR_C	758
Server1	CARQM	8
Server1	CQMCARQM	8
Server3	PRODQMGR_E	773
Server3	test1	256

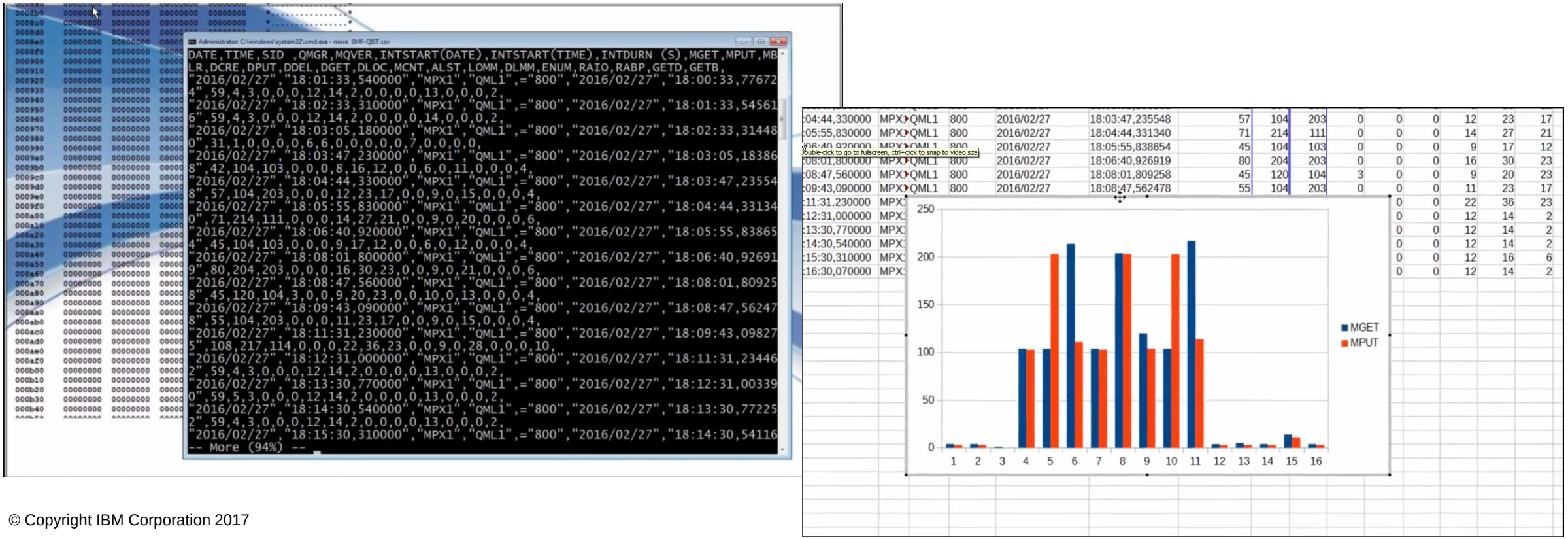
Complete list of log entries for all queue managers

1 2 3 4 5 ...10 »

Time	host	ibm_qmgrName	ibm_messageId	message	ibm_userAction
March 12th 2017, 16:57:51.000	Server1	PRODQMGR_A	AMQ5051	The queue manager task 'DUR-SUBS-MGR' has started.	None.
March 12th 2017, 16:57:50.000	Server1	PRODQMGR_A	AMQ5041	The queue manager task 'DUR-SUBS-MGR' has ended.	None.
March 12th 2017, 16:57:50.000	Server1	PRODQMGR_B	AMQ6125	An internal IBM MQ error has occurred.	Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the MQ Support site: http://www.ibm.com/software/integration/wmq/support/, or IBM Support
March 12th 2017, 16:57:50.000	Server1	PRODQMGR_B	AMQ6184	An internal IBM MQ error has occurred on queue manager PRODQMGR_A.000.	Use the standard facilities supplied with your system to record the problem

Similar resource data available on z/OS but via SMF

- By popular demand … open source tool to format MQ z/OS SMF records for easy import to spreadsheets and databases
  - http://github.com/ibm-messaging/mq-smf-csv
  - http://youtube.com/marktaylorhursley



How it looks in DB2

Control Center - DB2COPY1

Control Center Selected Edit View Tools Help

Object View

- Control Center
  - All Systems
  - All Databases
    - MYDB
      - Tables
      - Views
      - Aliases
      - Nicknames
      - Cache Objects
      - Triggers
      - Schemas
      - Indexes
      - Table Spaces
      - Event Monitors
      - Buffer Pools
      - Application Objects
      - User and Group Objects
      - Federated Database Object
      - XML Schema Repository (X
    - TOOLSDB

METAYLOR2016 - DB2 - MYDB - Tables

Name	Schema	Table space	Comment	Index table spa
QEST	MQSMF	USERSPACE1		
QIST	MQSMF	USERSPACE1		
QJST	MQSMF	USERSPACE1		
QLST	MQSMF	USERSPACE1		
QMAC	MQSMF	USERSPACE1		
QMST	MQSMF	USERSPACE1		
QPST	MQSMF	USERSPACE1		
QSST	MQSMF	USERSPACE1		
QTST	MQSMF	USERSPACE1		
WQ	MQSMF	USERSPACE1		
WTAS	MQSMF	USERSPACE1		

22 of 152 items displayed

Table - QPST

Schema : MQSMF
Creator : METAYLOR
Columns : 26

Actions:
- Open
- Query
- Show Related
- Create New

Columns

Key	Name	Data type
	DATE	DATE
	TIME	CHARACTER
	LPAR	CHARACTER
	QMGR	CHARACTER
	MQ_VERSION	CHARACTER
	INTERVAL_START_DATE	DATE
	INTERVAL_START_TIME	CHARACTER
	INTERVAL_DURATION	INTEGER
	"BUFFERPOOL"	INTEGER
	BUFFER_COUNT	INTEGER
	LOWEST_STEALABLE	INTEGER
	CURRENT_STEALABLE	INTEGER
	GETP_OLD_REQUESTS	INTEGER
	GETP_NEW_REQUESTS	INTEGER
	DASD_READ	INTEGER
	SET_WRITE_PAGES	INTEGER

Open Table - QPST

METAYLOR2016 - DB2 - MYDB - MQSMF.QPST

Edits to these results are performed as positioned UPDATEs and DELETEs. Use the Tools Settings notebook to change the form of editing.

DATE	TIME	LPAR	QMGR	MQ_VERSION	INTERVAL_START
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:10:04,930000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	21:39:58,000000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:09:51,200000	H019	MQPC	800	No
Nov 23, 2015	22:39:44,100000	H019	MQPC	800	No

Add Row
Delete Row

Commit Roll Back Filter Fetch More Rows

Automatically commit updates

100 row(s) in memory

Close Help

Example queries

- What was my largest message size retrieved for this queue?
  - SELECT MAX(Get_Max_Msg_Size) from MQSMF.WQ where (Base_Name= ‘LYNS.TEST.QUEUE');
  - Result was 11,189 (application people insisted it was 3,800)
- How many MQPUTs and MQPUT1s were completed?
  - SELECT SUM (Put_Count), SUM (Put1_Count) from MQSMF.WQ where (Base_Name = ‘LYNS.TEST.QUEUE');
  - Results:

Properties | SQL Results | Data Project Explorer

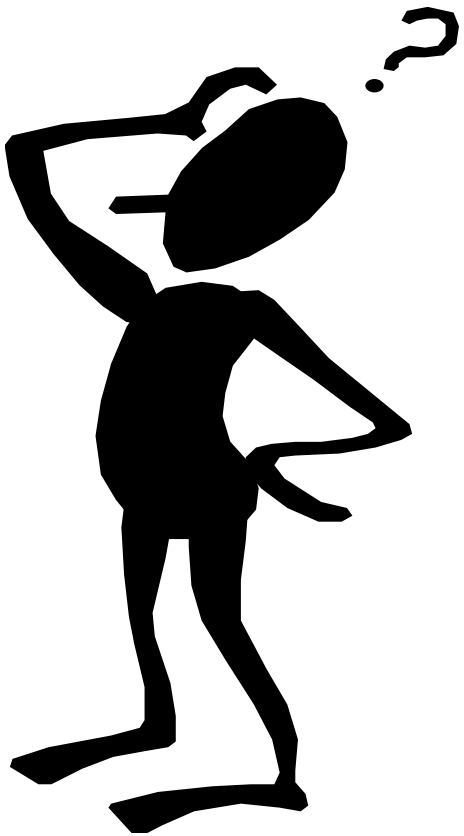
	1	2
1	0	3121

Summary

- MQ can be easily integrated with a variety of tools

- The pub/sub model for statistics makes it easy to add new consumers
  - Without disrupting any existing monitors
  - And makes it possible to add your own producers

- Using github for repository of code enables easy modification and sharing
- And the MQDev blog for documenting what we have done



Any questions?