

MQ Automation: Config Management using Amazon S3

T.Rob Wyatt

Change is the only constant

This presentation is a snapshot in time as of *27 September 2017*.

For the latest version, please check the author's web page:

<https://t-rob.net/links>

What is it?

Shell scripting tools

To build queue managers

On Linux

Reliably, repeatably, consistently

With a single command

How to build a queue manager

```
ksh -c "QMGR=BIRCH;$(curl --stderr /dev/null  
https://mqtc.s3.amazonaws.com/MQTC-SCRIPTS/s3-bootstrapmq.ksh)"
```

Session is companion guide to the scripts

- **Much of the session will be live demo**
 - ▶ Yes, the scripts are really that trustworthy
- **This deck is meant as companion notes/docs for the scripts, not as a stand-alone tutorial. Go get the scripts!**
- **Download the scripts to your UNIX/Linux VM, workstation, or server using the commands you find a few slides down.**
- **Feel free to keep using these (scripts and templates will be read-only) or set up your own S3 bucket and try creating your own configurations.**

This is only a bite-size chunk!

This session focuses entirely on build-time tasks because that's where many defects are introduced. However, a big part of the reason to do this is the benefit to day-to-day run-time tasks, and decommissioning assets at the end of their lifecycle (whether due to migration or sunsetting).

Having a system of record for the MQ configuration and reconciling back to it on-demand and in real time helps to find organic configuration “drift,” provides intrusion detection capabilities, and greatly reduces the burden of audit proof.



Project Background

Why?

- Because any system that relies primarily on human vigilance as a control will eventually fail.
- To reduce the incremental cost of building a new queue manager to exact specifications, including monitoring agents and certificates, to near zero.
- To reduce defects to near zero.
- To reduce request servicing time to near zero.
- To reduce the skill threshold required to perform routine tasks.
- To better manage the scale and dynamicity of cloud and virtual environments.
- To shift MQ Admin time away from tactical triage tasks to focus more on strategic tasks.

Design goals

- **Minimal prereqs to set up host image**
 - ▶ Ksh, curl, openssl, and MQ installed
 - ▶ AWS key so that files need not be made public access.
- **Single command populates config files and builds queue manager**
- **Cache configuration files locally in case remote access fails**
- **Facilitate automation of entire QMgr life cycle**
- **Make metadata about build spec queryable by instrumentation**
- **Proof of concept with low-cost, ubiquitous, Amazon S3 hosting**
- **Highly portable across Linux/UNIX platforms**
- **Multi-step self-signed cert distribution and other workflows supported**
- **Automatically capture logs of all activity in the tool**

Yes, its in Production

Customized versions of these scripts are currently running in Production at several Fortune 500 companies who are current or former clients. Thousands of queue managers have been built.

This is a sanitized version of those same tools, stripped-down for classroom use. In particular, this is limited to build-time tasks and configured to create a cluster on a single host.

Missing from this session are run-time scripts that compare the as-specified configuration to the as-running configuration and report on discrepancies. This can be helpful for intrusion detection, audit proof, or to help identify the source of configuration “drift” in shops with delegated admin control.



Minimal Prereqs

Before building one or more QMgrs...

- UNIX/Linux with modern ksh or bash, plus curl and openssl
- MQ has been installed
- Entries for the QMgrs to be built are defined in the configuration file on S3
- The necessary baseline and pattern files exist on S3

Manual setup to run Proof of Concept

Amazon Web Services IAM users are provisioned with public and secret keys that allow fine-grained permissions to be granted on S3 files.

These IAM keys cannot be distributed by the bootstrap script since its components would then be required to be world-readable. Ordinarily, these static artifacts are provisioned using the cloud or virtualization automation.

For purposes of this session, if you want to run the scripts you will be required to manually create the key files used by the scripts. These will be read by the scripts to sign S3 requests and format headers.

```
mkdir ~/.aws
cd ~/.aws
echo mqtc >.awsbucket
echo AKIAJHSJBNHSOVAVAVWQ >.awsid
echo "j6/WAqAkmcBc0NCxIPTED6dnj7SmN/0St1Zu20u/" >.awskey
echo us-east-1 >.awsregion
cd
```

Work in progress

Most of the work in preparing for this session was in sanitizing the scripts and documentation to remove confidential client information, names and details. As a result, some sections are pretty bare. I'll continue to update the scripts and templates until things like pattern files are more representative.

Currently the cluster pattern doesn't build. The versions in Prod all rely on signed certificates so are able to deploy a static set of signer certs to make it all work. I started converting for self-signed but there is no good way to demo that. Instead, I'll build CA automation into one of the repositories. At that time I'll fix the cluster AUTHREC entries and channels properly.


All of this needs to be completed before IBM Think, if they accept my session proposals, so it **will** get done. ;-)



Operation Steps

After the <enter> key

- Name of QMgr to be built is passed to a KSH session
- Bootstrap script is downloaded by curl and piped to the same ksh session
- The bootstrap script calls the sync script that builds the directory structure, downloads the script files and templates
 - ▶ After the first run, the local cache holds the last known configuration and script set
 - ▶ This and other scripts always try to refresh the cache before proceeding.
- The bootstrap script resumes and checks the configuration file integrity
- The queue manager is built
 - ▶ The ini file is edited as part of the baseline. Sorry, ini file tuning for patterns is not (yet?) in this toolset. Obviously since tuning is what patterns are about, it will be required and so I'll add it soon.
 - ▶ The baseline configuration is applied
 - ▶ The pattern configuration is applied
- The QMgr certificate or cert request file is loaded to S3
- The local log file is uploaded to S3



Detail Notes

Local directory structure

```
.__MQTC-CERT
|__MQTC-CERTTEMP
|__MQTC-LOGS
|__MQTC-SCRIPTS
| |__s3-bootstrapmq.ksh
| |__s3sync.ksh
| | # s3-bash library
| |__s3-common-functions
| |__s3-delete
| |__s3-get
| |__s3-put
|
|__MQTC-TEMPLATES
| |__MQTCBASE.8007.v01.ini
| |__MQTCPATT.8007.v01.ini
| |__qmgrs.ini
|
|__~mqm
| |___.aws
| | |___.awsbucket
| | |___.awsid
| | |___.awskey
| | |___.awsregn
```

Approved Certificates
Temporary holding area for new certs
Log files from script runs
Scripts go here
Builds QMgrs from central spec files
Initialize, sync local files from S3
External package for s3 functions
S3-bash core library functions
S3-bash deletion script
S3-bash get to stdout
S3-bash put a file to S3

Configuration specifications
MQ v1 Baseline for MQ v8.0.0.7
MQ v1 Pattern for MQ v8.0.0.7
QMGr instance definitions

Admin or mqm home directory
Hidden AWS directory. Owner-read only.
AWS S3 Bucket Name
AWS S3 Public ID for user
AWS S3 secret key for user
AWS S3 hosting region

These are built by the sync script

These are provisioned into the host image.
Or in our case, you manually create them as part of the exercise.

To obtain the scripts

Paste the following into a Linux session while logged in as mqm:

```
mkdir ~/.aws
cd ~/.aws
echo mqtc >.awsbucket
echo AKIAJHSJBNHSOVAVAVWQ >.awsid
echo "j6/WAqAkmcBc0NCxIPTED6dnj7SmN/0St1Zu20u/" >.awskey
echo us-east-1 >.awsregion
cd
```

Run the following command:

```
ksh -c "$(curl --stderr /dev/null https://mqtc.s3.amazonaws.com/MQTC-SCRIPTS/s3sync.ksh)"
```

I'll post them to Github when I get them fleshed out a bit more.

Open Source

The scripts use the s3-bash4 library maintained by Chi Vinh Le on Github:

<https://github.com/wikiwi/s3-bash4>

The author writes:

s3-bash4 is a small collection of Bash scripts to do simple interaction with Amazon S3 using AWS Signature Version 4. The advantage of using s3-bash4 is that it's extremely lightweight and easy to use. No need to setup Python, Java, Ruby and co.

This is inspired by the discontinued s3-bash from cosmin. I was in need of a Bash version that supports the newer AWS Signature Version 4.

S3-bash4 is available under an Apache license.

Note: The version of the scripts used here has been modified to include an ACL that gives the bucket owner full control of files uploaded to S3. The modifications have not been integrated into the original source at this time.

S3 Bucket policies

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "mqtc-user-get",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::{my S3 ID here}:user/mqtc"
    },
    "Action": "s3:GetObject",
    "Resource": [
      "arn:aws:s3::mqtc/MQTC-SCRIPTS/*",
      "arn:aws:s3::mqtc/MQTC-TEMPLATES/*",
      "arn:aws:s3::mqtc/MQTC-CERT/*"
    ]
  },
  {
    "Sid": "mqtc-user-put-logs-certs",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::{my S3 ID here}:user/mqtc"
    },
    "Action": "s3:PutObject",
    "Resource": [
      "arn:aws:s3::mqtc/MQTC-LOGS/*",
      "arn:aws:s3::mqtc/MQTC-CERTTEMP/*"
    ],
    "Condition": {
      "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
      }
    }
  }
],
```

Make sure our lab user can access all these files. Note that CERTTEMP and LOGS are omitted because ordinary users should not normally be able to read all those files.

This allows the MQTC lab user to upload log files and raw certificate files for processing. The Condition statement makes sure that the bucket owner has full access to the files which is not the default.

S3 Bucket policies

```
{
  "Sid": "deny-other-actions",
  "Effect": "Deny",
  "NotPrincipal": {
    "AWS": "arn:aws:iam::{my S3 ID here}:root"
  },
  "NotAction": [
    "s3:PutObject",
    "s3:PutObjectAcl"
  ],
  "Resource": [
    "arn:aws:s3:::mqtc/MQTC-LOGS/*",
    "arn:aws:s3:::mqtc/MQTC-CERTTEMP/*"
  ]
}
```

This says that if you are not the bucket owner, and you try to do something that is not PUT an object or the ACL that lets the bucket owner read the object, and you are trying to do it in the logs or cert temp directory, the action is rejected.

This ensures that only the bucket owner and object owner can read uploaded log or certificates. That's a lot more significant when different people or different departments work from different S3 user IDs.

Tutorials and more

People kept asking where to find the slides, videos. Here ya go...

- YouTube tutorials: <https://www.youtube.com/tdotrob>
- Twitter
 - ▶ @deepqueue (MQ & security)
 - ▶ @tdotrob (MQ & security + politics, humor, autism)
- LinkedIn: <https://www.linkedin.com/in/tdotrob/>
- Blogging on general IT, security, malvertising. How to hire me: <https://ioptconsulting.com>
- MQ web site and blog: <https://t-rob.net> (Slides are uploaded here)

All my web sites are linked together in the nav bar. Go to Ask-An-Aspie for autism content, or The Odd is Silent for everything that's not autism or IT.

Questions & Answers

