MQ Automation: Config Management using Baselines, Patterns and Apps

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

Cloud is good!

All the advantages we used to like about SOA and ESB:

- Availability
- Network ubiquity
- Ephemeral instances
- Location independence
- Horizontal scale-out, scale-back
- Dynamic, run-time resolution of endpoints

Cloud is different

Reconsider “best” practices in light of

- Organic growth or,
- Plethora of patterns or,
- Ultra-personalization or,
- Multi-tenancy shared hubs or,
- Decentralized administration or,
- Swarms of ephemeral instances or,
- …

If you do what you've always done,

You'll get what you've always got.

Right?

If you administer MQ

in the cloud

the way it's always been done,

you get the network

you always had.

How do you manage and secure clusters

with ephemeral nodes?

Do all your QMgrs have the same name?

How do you ensure access is granted on a least-privilege, need-to-know basis

How do you reconcile dynamicity, manageability and consistency?

How do you make sure that

monitoring and operational instrumentation

is consistently installed

and correctly configured?

How do you provide

consistent, reliable, repeatable, dynamic,

defect-free configurations

at scale

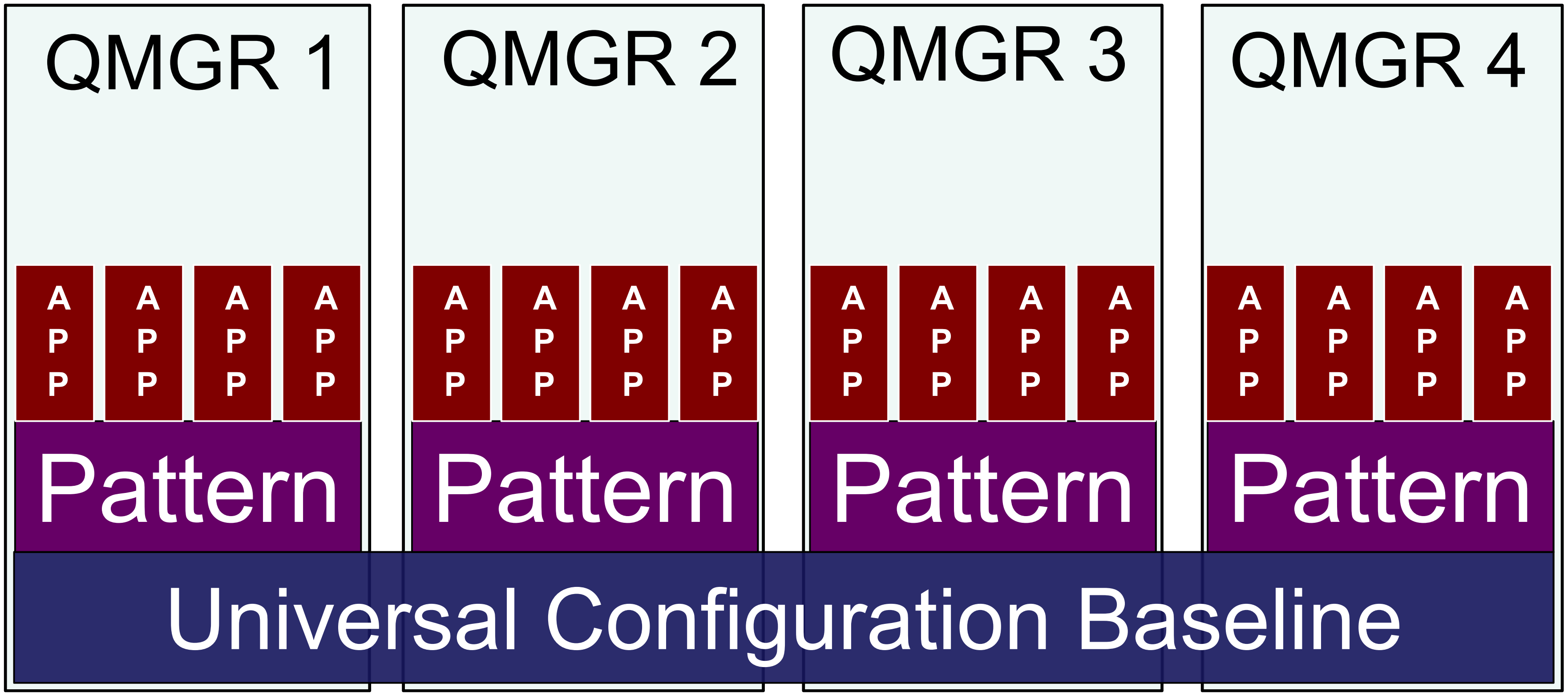
and in budget?

Design for manageability

Configuration considerations include:

- Static vs. Dynamic
- Consistency vs. volatility
- Functional demarcation
- Ownership and stakeholders
- Quality vs. Speed vs. Cost vs. scale

Layered configuration architecture

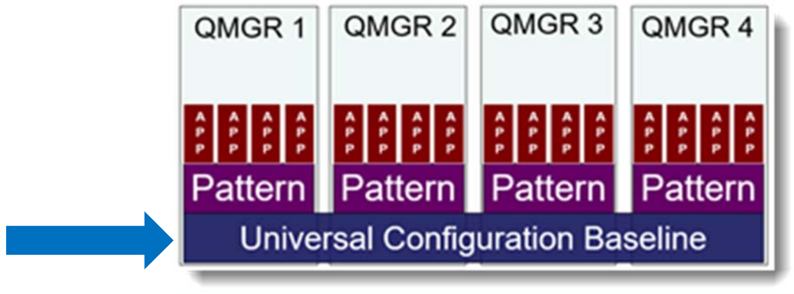


Configuration Baseline

Elements that exist on all nodes

Possibilities include:

- Certificates
- Event settings
- Monitoring agents
- Cluster membership
- Provision admin access
- Derivation of QMgr name

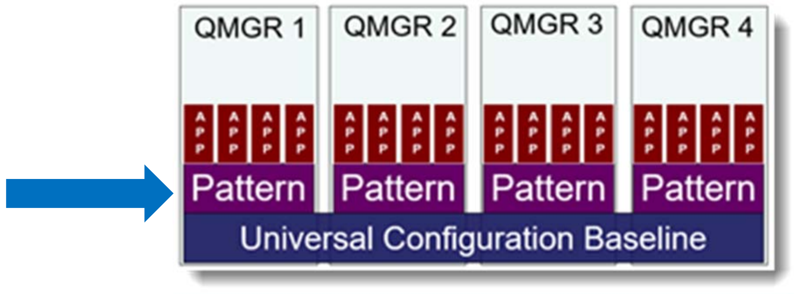


Functional patterns

Tuning specific to classes of service or topology

Possibilities include:

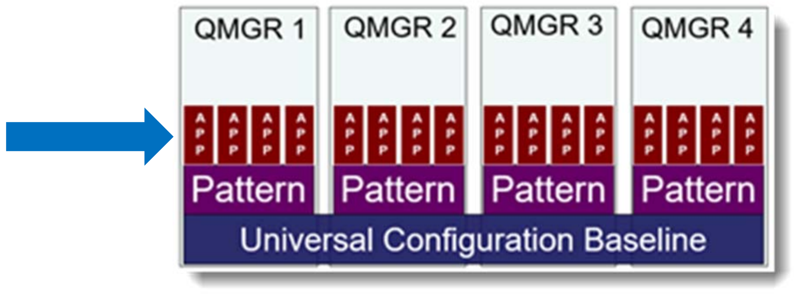
- System of record (Highly available, deep well known queues, etc.)
- Client concentrator (Few or no well-known queues, lots of channels, etc.)
- Low-latency failover (Few & small logs, low MAXDEPTH, etc.)
- Shared Hub
- B2B Gateway
- Order preservation
- Managed File Transfer
- Advanced Message Security



Applications

Objects specific to each app

- **Each QMgr can host multiple apps**
- **Supports same app on more than one MQ node**
- **Configuration variability within a given app is supported**
- **Inheritance of local attributes into app names supported**
  - Dev, SIT, UAT, etc.
  - Versioning
- **Portable across QMgrs**



In practice

- **Baseline maintained by MQ Administration team**
  - Collaboration of MQ Admin/Architect team & stakeholder teams, possibly including:
    - Enterprise Security/Audit
    - Accounts administration
    - Platform SA
    - Operations
  - Represent platform (Appliance, UNIX, Windows, z/OS, etc.)
- **Patterns are offerings to LOB customers**
  - Collaboration between LOB teams and MQ Admin/Architect team
  - Compromise between need for standardization vs unlimited customization
  - Represent classes of service based on function
  - Represent topology (Gateway, client concentrator, enclave, etc.)
  - Represent platform (Appliance, UNIX, Windows, z/OS, etc.)
- **App configurations tailored to business requirements**
  - Collaboration between LOB teams and MQ Admin/Architect team
  - Supports business functions
  - Supports app-specific user access, monitoring
  - Inherits local metadata values

Configuration versioning

- **Configuration files once deployed are static objects.**
  - Copy, edit and increment version number to make changes
  - A given node can have one and only one version of a configuration
  - Multiple versions can co-exist across nodes to support migration, resource availability
  - Redeploy prior version to fallback after deploy new version

- **Configuration version is inherited metadata**
  - Namelist object name corresponds to baseline/pattern/app name
  - Metadata stored in namelist entries
  - Queryable at run time by automation, interactive user
  - Queryable offline in object backup files

- **Version metadata can be propagated to object names where appropriate**
  - For example, where queues represent service endpoints

Portability

- **Queue managers are containers for business objects**
  - Apps should not be QMgr-aware
  - Apps can be QMgr alias aware
- **Location independence**
  - Endpoints can be resolved at run-time from the configuration database
  - Virtual IP or DNS can be used for redirection to physical instance
  - IP sprayers may be suitable for MQ client connections
- **Introspection**
  - Inheritance of locally relevant metadata
  - Inheritance of local configuration differences in lower layers
  - Use of MQSCX to query local configuration can greatly reduce number of configuration files and versions.

MQ Version migration

My personal "Recommended Practice" makes MQ version migration easy

- **Pool of nodes at -1, Current, and +1 version**
  - Example: v7.5, v8.0 and v9.0
- **The -1 version is deprecated, the +1 version is targeted**
  - Exception process for apps that can't get at least to -1 version
  - Possibly with premium chargeback
- **Constant migration forward across all application teams**
  - Each app on its own schedule
  - Supports high density multi-latency
  - Provides smooth, easy fallback
- **As nodes in the -1 pool become idle, redeploy as Current or +1 version**
  - Manage workload, capacity by leapfrogging

Consistency over time

Reconciliation of definitive As-Specified config to As-Running

- **Intrusion detection if configuration changes unexpectedly**
- **Enforce standards where required**
- **Configuration “drift” may highlight unmet needs**
- **Configuration “drift” may reveal skill gaps**
  - For example, tuning that conflicts with the underlying pattern
  - Working outside the configuration management system
- **Central configuration database**
  - Shows “estate at a glance”
  - Edit configurations offline using text files, templates, or tools

Automation is essential

Scale up without increasing admin overhead or defects

- **Minimal prereq on host node is almost all static**
  - MQ installed
  - IBM GSKit and/or OpenSSL
  - KSH, BASH or other script interpretation engine
- **Automation detects local customization that determines what to build**
  - May be env vars, IP address, DNS name, host name, parm files
  - Typically specified in the virtualization or deployment package dashboard
- **Automation pulls down boot script, automation libraries, config modules**
  - Use local parameters to resolve which configuration to use
  - Apply substitutions from local parms and config files
  - Build to precise spec
- **On-host operator activities minimized**
  - Execute a single boot script
  - Drastically reduced defects, turnaround time, and cost

Multiple Successful Prod deployments

- **Developed and refined across several client engagements**
- **Can be customized heavily without breaking the architecture**
- **Gem Software’s MQSCX greatly multiplies benefits relative to cost**
- **Can be integrated into anything with an API**
  - For config management system of record
  - For outbound notifications – “You’ve got MQ!”

A **very quick and dirty version is presented in the companion session: MQ Automation: Config Management using Amazon S3**

Aloeswood room

- Monday 15:50
- Wednesday 8:30

Questions & Answers