MQ and IIB Deployment Patterns using Docker on IBM Cloud Private

Sandeep Chellingi
Head of Hybrid Cloud Integration, Prolifics
Agenda

- Integration Modernization
- Agile Integration Architecture
- SOA vs Micro Services
- Evolution of Agile Architecture
- Containers
- High Availability Scenarios
- IBM Cloud Private
Modernization will impact more than just your software
Agile Integration Architecture drives the change

- Fine grained deployment
  - Speed development & simplify mgmt
- Decentralized Ownership
  - Accelerate agility & innovation
- Cloud native infrastructure
  - Provide resiliency and scalability
Typical benefits sought from a move to microservices

**Agility**
Faster iteration cycles, bounded contexts, autonomous teams

**Scalability**
Elastic scalability, workload orchestration, cloud infrastructure

**Resilience**
Minimized dependencies, discrete failover, fail fast, start fast
Difference between SOA and Micro services

Service oriented architecture (SOA) and microservices architecture relate to different scopes

They are *complementary*, rather than *competing*.

Service oriented architecture (SOA) relates to *enterprise service exposure*.

Microservices relate to application architecture.

Webinar based on above paper (55 mins)  http://ibm.biz/MicroservicesVsSoaFullWebinar
Evolution to agile integration

Centralized ESB

Fine-grained integration deployment

Decentralized integration ownership

Integration as a microservice runtime

Containerization

Application autonomy

Polyglot runtimes
Benefits of a container-based approach

**Build Agility**
- Higher build velocity
- Faster maintenance cycles
- Consistency across environments
- Independent component deployment
- Simplified testing of isolated components

**Fine-grained Resilience**
- Safe independent deployment, removing risk of destabilizing existing components.
- Disposable components enabling rapid start/stop for simple HA and scaling.
- Fit for purpose discrete topologies

**Infrastructure Optimization**
- Maximized component/resource density
- Lower overheads than virtual machine isolation
- Dynamic and elastic provisioning of resources (CPU, memory, persistent volumes)
- Usage based licensing models

**Operational Consistency**
- Standardized infrastructure platform skillset across products
- Platform based load balancing
- Platform based high availability
- Platform based scaling via policy
- Platform based logging/monitoring

**Component Portability**
- Containers can be re-distributed dynamically across nodes within a given cloud
- Images can be built and run on any cloud
- Focus on open containerisation standards such as Docker and Kubernetes
- Enables multi-cloud scenarios

**Scalability & Continuous Availability**
- Fine-grained dynamic scaling of individual functionality
- Implicit high availability based on replication policy and built in re-instatement
- Provided consistently across all types of components
Operational consistency of container based solutions

- Resources
- Artefacts
- Security
- Deployment
- Operations
- Delivery
- Routing

Traditional infrastructure (Pets)

Cloud native infrastructure (Cattle)

Runtime specific

Provided by platform
# The scope of Integration Modernization

<table>
<thead>
<tr>
<th>Integration Modernization</th>
<th>Replatform</th>
<th>Repackage</th>
<th>Refactor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Integration</strong></td>
<td>Containerize application integration infrastructure</td>
<td>Fine-grained integration deployment</td>
<td>Optimize integration deployment for cloud platform</td>
</tr>
<tr>
<td><strong>Messaging &amp; Events</strong></td>
<td>Containerize messaging provider infrastructure</td>
<td>Fine-grained queue deployment</td>
<td>Optimize messaging for cloud platform</td>
</tr>
<tr>
<td><strong>API Management</strong></td>
<td>Containerize API gateway and management infrastructure</td>
<td>Align API component placement</td>
<td>Consumer aligned API exposure</td>
</tr>
</tbody>
</table>

- **DevOps Ownership**
  - Increasingly automated
  - Increasingly decentralized

- **Traditional**
  - Cloud-native

- **Operational consistency**
  - Platform based HA
  - Platform based logging
  - Usage licensing

- **Agility through independence**
  - Granular components
  - Increased isolation
  - Independent release cycles

- **Embrace cloud-native**
  - Image based deploy
  - Automated CI/CD
  - Elastic scalability

- **Increasingly decentralized**
- **Increasingly automated**

---

*MQ Technical Conference v2.0.1.8*
Boundaries make complex environments manageable

**Managed API gateways define and enforce application boundaries**

Microservice component

API

Application boundary

API gateway

Silo application
Running MQ in Containers

- MQ has been supporting Docker containers since 2015 with images on Docker Hub and Docker Store and sample code on Github.

- Recently it has been demonstrating how to get most of docker containers and Kubernetes providers like Redhat open shift, Pivotal container service.

- MQ Advanced is available as a fully supported product with IBM Cloud Private, a Kubernetes-based solution from IBM.
# MQ container orchestration support

<table>
<thead>
<tr>
<th>Component/arch</th>
<th>IBM Cloud Public</th>
<th>IBM Cloud Private</th>
<th>Other Container Services (Docker Hub/Store)</th>
<th>Other Container Orchestration services (e.g. OpenShift)</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQ Advanced Server Production</td>
<td>x86_64</td>
<td>x86_64</td>
<td>ppc64le (POWER)</td>
<td>s390x (z/Linux)</td>
<td>x86_64</td>
</tr>
<tr>
<td>Client (dev &amp; prod)</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>MFT &amp; AMS &amp; MQTT</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>SDK</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>MQ Explorer</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>Salesforce Bridge</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>Blockchain bridge</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>RDQM</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>MQ IPT</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
<td>▲</td>
</tr>
</tbody>
</table>

- **MQ supported with image and sample available**: Supported, image and Helm chart available.
- **MQ supported with sample**: Supported, and you need to build your own image (samples/blog available).
- **MQ supported with no sample**: Supported, and you need to build your own image.
- **Not supported**
Running IIB/ACE in Docker containers

No description, website, or topics provided.

18 commits 1 branch 0 releases 2 contributors EPL-2.0

Branch: master New pull request

dan robinson Merge branch 'master' of https://github.com/ot4i/ace-docker

- 11.0.0.0/ace/ubuntu-1604
  - Merge branch 'master' of https://github.com/ot4i/ace-docker
- CLA.md
  - Update repo layout and add CLA and README
- LICENSE
  - Initial commit
- README.md
  - Further README updates
Helm Charts

### IIB Helm Charts

**IBM INTEGRATION BUS**

IBM® Integration Bus is a market-leading lightweight way for systems and applications to communicate with business value, reduce IT complexity and save money choices, skills and interfaces to optimize the value of

**Introduction**

This chart deploys a single IBM Integration Bus for Developers integrating IBM Cloud Private or other Kubernetes environment.

**Installing the Chart**

To install the chart with the release name `foo`:

```
helm install --name foo ibm-integration-bus-dev --set license=accept
```

---

**Configuration**

The following table lists the configurable parameters of the `ibm-integration-bus-dev` chart and their default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>license</td>
<td>Set to accept to accept the terms of the IBM license</td>
<td>Not accepted</td>
</tr>
<tr>
<td>image.repository</td>
<td>Image full name including repository</td>
<td>ibmcom/iib</td>
</tr>
<tr>
<td>image.tag</td>
<td>Image tag</td>
<td>10.0.0.10</td>
</tr>
<tr>
<td>image.pullPolicy</td>
<td>Image pull policy</td>
<td>IfNotPresent</td>
</tr>
<tr>
<td>image.pullSecret</td>
<td>Image pull secret, if you are using a private Docker registry</td>
<td>null</td>
</tr>
<tr>
<td>service.name</td>
<td>Name of the Kubernetes service to create</td>
<td>qmgr</td>
</tr>
<tr>
<td>service.type</td>
<td>Kubernetes service type exposing ports, e.g. <code>NodePort</code></td>
<td>NodePort</td>
</tr>
<tr>
<td>resources.limits.cpu</td>
<td>Kubernetes CPU limit for the Queue Manager container</td>
<td>2</td>
</tr>
<tr>
<td>resources.limits.memory</td>
<td>Kubernetes memory limit for the Queue Manager container</td>
<td>2048M1</td>
</tr>
<tr>
<td>resources.requests.cpu</td>
<td>Kubernetes CPU request for the Queue Manager container</td>
<td>1</td>
</tr>
<tr>
<td>resources.requests.memory</td>
<td>Kubernetes memory request for the Queue Manager container</td>
<td>512M1</td>
</tr>
<tr>
<td>nodename</td>
<td>IBM Integration Bus integration node name</td>
<td>IIB_NODE</td>
</tr>
<tr>
<td>servername</td>
<td>IBM Integration Bus integration node name</td>
<td>IIB_SERVER</td>
</tr>
</tbody>
</table>
How and why does IIB/ACE use MQ?

- As an asynchronous messaging provider
- By certain nodes to maintain state (Collector, Resync, etc)
- As a co-coordinator for global (two phase commit) transactions
Running MQ and IIB/ACE in containers

- Use cases where IIB/ACE can use Client bindings connections then run MQ and IIB/ACE in separate containers.

- Use cases where IIB/ACE requires local (server) bindings connections can be deployed in different options:
  - Embedded in Same Container
  - Separate Container with same PIS Namespace
Running MQ and IIB/ACE in same containers

- **replicas = n**
  - no persistent volume claim
  - HA by replication (continuous availability)
  - Elastic horizontally scalability
  - Non-durable use of EDA*nodes
  - No 2 Phase Commit.

- **replicas = 1**
  - persistent volume claim
  - HA by reinstatement
  - Manual horizontal scalability
  - Durable use of EDA*nodes
  - 2 Phase Commit
IIB running in Kubernetes

- The IBM Cloud Container Service provides a Kubernetes-based public cloud solution
- IBM Cloud Private provides a Kubernetes-based private cloud solution for running in your own datacenter
MQ HA Availability on Docker/Kubernetes

- **Single resilient queue manager**
  - Cloud manages failover to somewhere with spare capacity
  - Networked storage (block or filesystem), managed by separate subsystem

- **MQ Multi-Instance Queue Manager**
  - Active – Standby pair, MQ Manages Failover
  - Shared Network Storage managed by different subsystem

- **Replicated data queue manager**
  - “Shared Nothing” approach, MQ manages failover
  - Local block storage, synchronously replicated by MQ
  - Not supported on Containers
IIB/ACE HA Availability on Cloud

- **Single resilient Integration Node (v10) or Integration Server (v11)**
  - Cloud manages fail-over to somewhere with spare capacity
  - Networked storage (if needed) managed by separate subsystem

- **Multi-instance Integration Node (v10)**
  - Requires local MQ
  - MQ manages fail-over
  - ACE v11 not supported (yet)
  - Networked storage (filesystem), managed by separate subsystem
IBM Cloud Private Solution

IBM Middleware & Open Source – e.g. Data, Analytics and Developer Services
Cloud-enabled middleware, application runtimes, messaging, databases & analytics to optimize current investments and rapidly innovate

Core Operational Services
To simplify Operations Management, Security, DevOps, and hybrid integration

Kubernetes-based Container Platform
Industry leading container orchestration platform

Cloud Foundry
For prescribed application development & deployment

Terraform (CAM)
Infrastructure as Code for multi-cloud provisioning to public and on-prem private clouds

Runs on existing IaaS: vmware, openstack, Power Systems, System Z, IBM Spectrum

Dell, Cisco, NetApp, Lenovo, ...
IBM Cloud Private Overview

A customer-managed Private Cloud software solution based on Kubernetes, Docker and Cloud Foundry technology that runs on customer-provided infrastructure (or in Public Cloud IAAS)

A platform to run containerized versions of IBM Software such as Datapower, IIB, MQ, DB2, Cloudant, Data Science Experience (Apache Spark), Blockchain

A platform to build Cloud Native, Stateless, 12 Factor apps including powerful developer tools to jump-start projects

A platform to run Modernized and Containerized Legacy Applications including tools and services to help transform code.
IBM Cloud Private Architecture

Workloads
- Cloud Native
  - MicroProfile
  - Node.js
  - Spring
  - Kitura
  - Java
  - JEE
- Containerized Middleware
  - WebSphere
  - MQ
- Data Workloads
  - DB2
  - IBM Integration Broker
  - IBM Cloudant
- DevOps & Tools
  - Jenkins
  - GitHub

Next Generation Management
- Prometheus
- Grafana
- Stack

PaaS (Platform as a Service)
- Cloud Foundry

CaaS (Container as a Service)
- Kubernetes
- Docker
- Istio

Infrastructure
- VMware
- OpenStack

Infrastructure Automation
- Chef
- Terraform

MQ Technical Conference v2.0.1.8
IBM Cloud Private Dashboard

System Overview

Nodes
86% Active
6 Active
1 Inactive

Shared Storage
55% Available
110GB Available
87GB Used
0GB Released
2GB Failed

Deployments
93% Healthy
26 Healthy
2 Unhealthy

Resource Overview

CPU
34

Utilization: 0.79 CPU/12%
Allocation: 0.79 CPU/12%

Memory
121.6 GB

Utilization: 32.5 GB/26%
Allocation: 12.5 GB/12%

GPU
0

Utilization: 0 GPU/0%
Allocation: 0 GPU/0%
IBM Cloud Private Helm Catalog

Catalog

Search for "mq"

Deploy your applications and install software packages

ibm-mqadvanced-server-dev
IBM MQ queue manager.

ibm-charts
Questions & Answers