MQ v9 & Docker

Guide IBM MQ à Air France Roissy

27 septembre 2016

AM. Mamic G. Trabucco

AIRFRANCE KLM

Summary

What is Docker?

- Quick Introduction
- Containers vs. VMs
- Containers usage logic

IBM MQ & Docker

- Subtleties
- Raw Performance expectations
- Perks

AF/KL Use Case

- Pros & Cons
- Container Creation
- High Availability Example

Questions

Conclusion

AIRFRANCE KLM

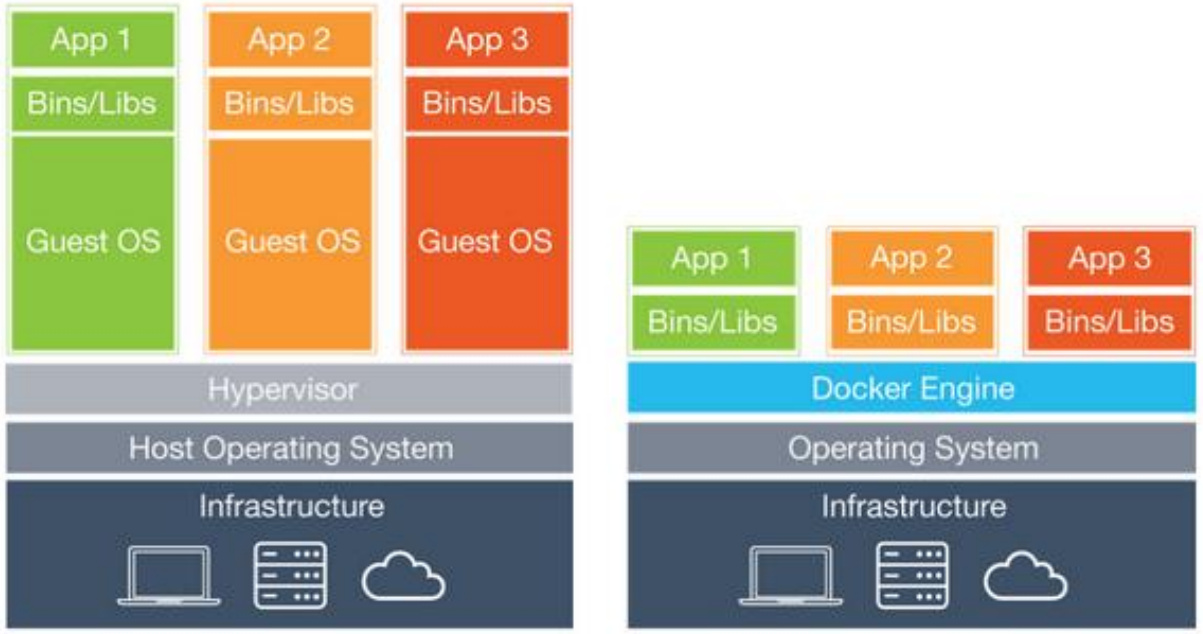
Virtualisation Automation Framework based on LXC *(Linux Containers)*

- Relies on cgroups features available on the Linux Kernel (since V. 2.6.24)

Docker => Freight analogy

Docker's Credo (« build, ship, run »)

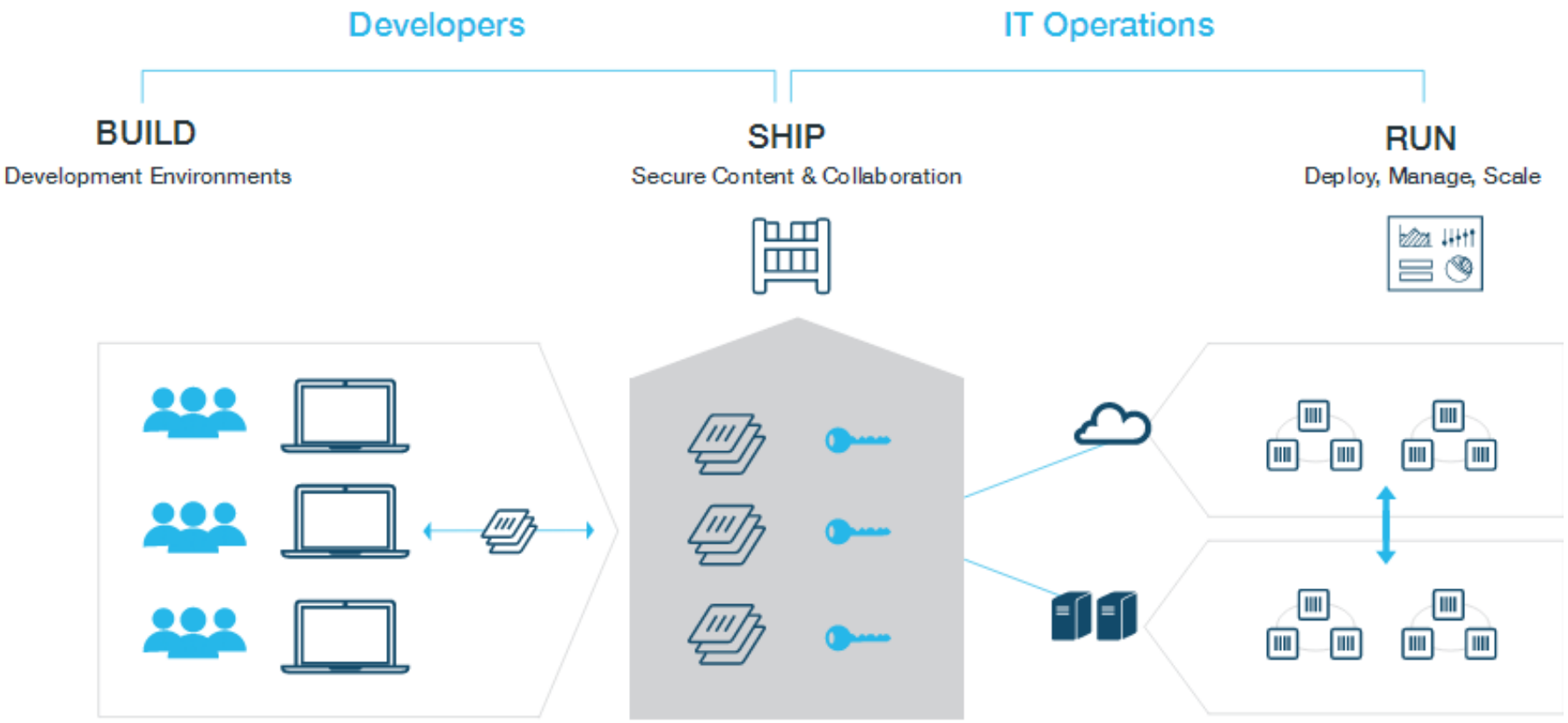
- *Build*
  - Compose and version containers with everything you need for its full functionality
- *Ship*
  - Design the entire cycle of application development, testing and distribution
- *Run*
  - Easily deploy and setup your containers (deployment automation greatly simplified over common systems)



Virtual Machines include the application, the necessary binaries and libraries, and an entire guest operating system -- all of which can amount to tens of GBs.

Containers include the application and all of its dependencies --but share the kernel with other containers, running as isolated processes in user space on the host operating system.
Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.

-> The container binaries are running in the Host OS kernel space


Developers
IT Operations
BUILD
Development Environments
SHIP
Secure Content & Collaboration
RUN
Deploy, Manage, Scale

Software support

- `MQSeriesRuntime, MQSeriesServer, MQSeriesClient, MQSeriesJava, MQSeriesJRE, MQSeriesGSKit, MQSeriesMsg, MQSeriesMan`
- The queue manager data directory *(/var/mqm by default)* must be stored on a Docker volume which keeps persistent state.

  ! Important ! You cannot use the union file system.
- You must either mount a host directory as a data volume, or use a data volume container.
- Applications can only be locally bound to the queue manager (client BINDINGS mode) when running in the same container as that queue manager.
- Security recommendation requires at least Linux kernel version of V3.16 on the Host OS
  - FYI RHEL V7.2 is based on V4.x
- IBM MQ v8 requires at least RHEL V6.x
- IBM MQ v9 requires at least RHEL V7.x

Hardware support

- Recommendations are identical to a bare metal setup

Single `App` instance on a physical box

Multiple `App` instances on a physical box

		Bare Metal	Containers (Docker)	VM
Resources	**Multiple Instances configuration?**	Complex, error prone	Simple (1/container) Mapping/port forwarding	Simple (1/VM) Mapping/port forwarding
	Processes behaviour	All on the same box	All « contained » but sharing the same kernel (shared resources)	All running in separate OSs VMs (isolated)
	CPU *besides OS needs*	100% available	Dynamic/Reserved allocation per container (per core/clock)	HW/SW Dependent
	Memory *besides OS needs*	100% available	Dynamic/Reserved allocation per container	HW/SW Dependent
	Ports *besides OS needs*	100% available	Redirections needed	Redirections needed
	HDD *besides OS needs*	100% available	No rule (still contained)	Dynamic/Reserved allocation per Image
	Software stack versioning	None (autoinst, etc.)	Dockerfile + Docker Trusted Registry	None / snapshots
	Migration over new HW	Manual, error prone	Seamless	"Mostly" seamless
	Migration over new OS	Host OS Dependent	Host OS Agnostic	Host OS Agnostic

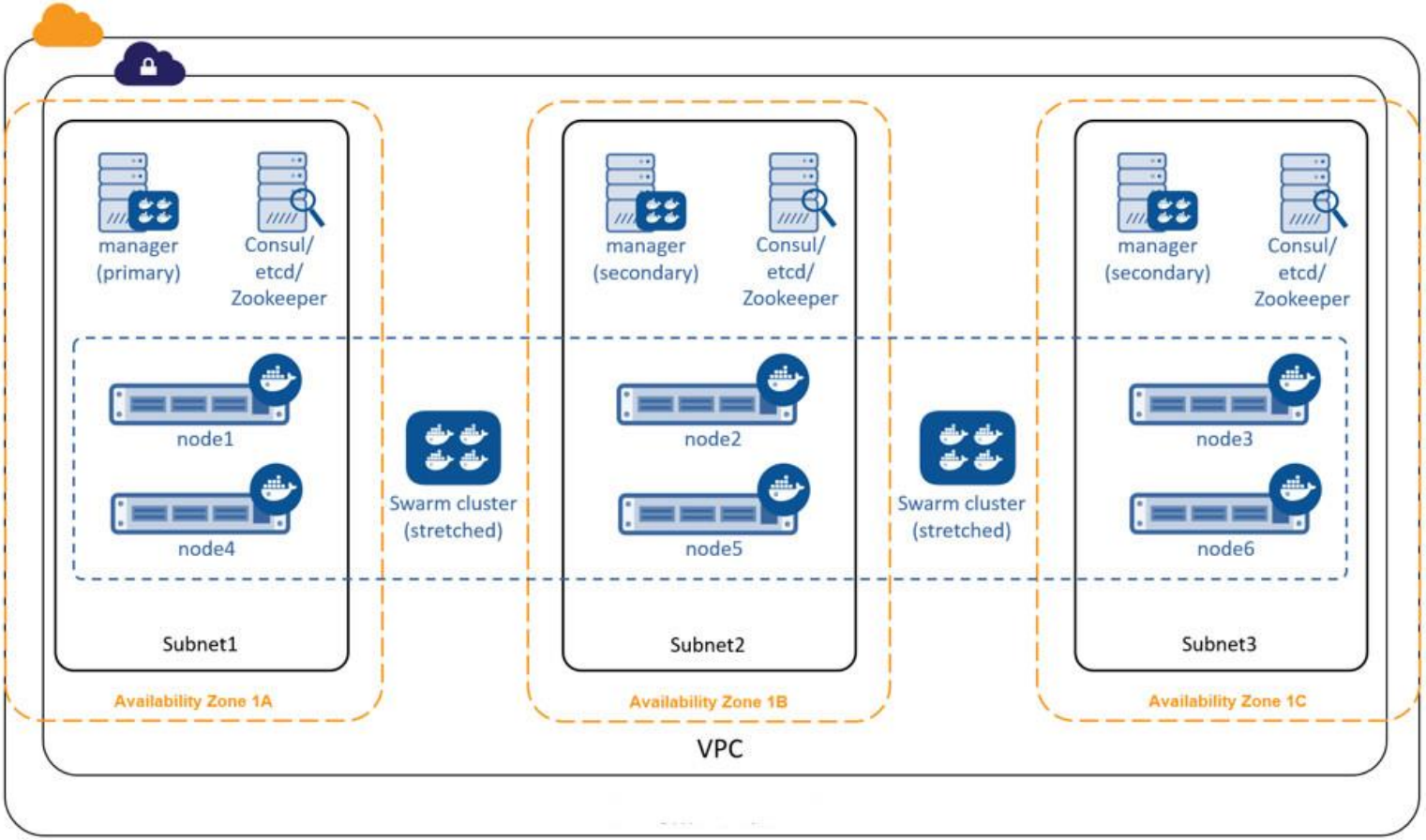
Pros	Cons
IBM MQ (v8, v9) containers already available • Either `config.mqsc` or Remote MQ administration avail.	Needs container creation up front for specific needs
Seamless replication (`docker run <repo/container name>`) • Deploy a specific container and container version • Seamless migration over a HW infrastructure • Multiple Versions Coexistence	Each container runs the full App Stack • wasted resources (RAM, CPU) • requires all libs, even system ones in the container
Versionning (`docker build -t <container name>`) • Create branches, reuse existing containers	Requires a specific version management infrastructure (Docker Trusted Registry)
Simple instanciation and destruction mecanism (`docker run ...` / `docker stop ...`)	Requires an ephemeral build approach • settings are to be either "hard coded" or loaded at startup • data must be stored elsewhere


```
# Dockerfile sample content
# Container image build statements
FROM <AF/KL Docker Trusted Registry> / <DI.MW/MU Repository> /<IBM MQ Docker base image>:<revision>
COPY <qpasa installer folder> <container qpasa install folder>
RUN <qpasa install>
COPY <qpagent installer folder> <container qpagent install folder>
RUN <qpasa install>
RUN …

# Container image run statements
EXPOSE <qpagent port> <archivemq port> …
ENV <key>=<value> <key>=<value> …
CMD <container startup script>
```

03 – AF/KL Use Case

0.3.3 – High Availability Example



Questions

> MQ doker image made by IBM includes OS or not ?

> Can we use the IBM Container and extend it with other necessary software for us (like QPASA) ?
What his your stance regarding customer support in this case ?

> Can we run more than one container instance on the same box (node) ?
Does it change the current licensing approach ? And IBM support ?

- **Is there a recommended list of Docker Engine versions ? And minimal version of Docker Engine ?**

- **Performances impacts (CPU , I/O) ?**

04 – Conclusion

- **Lighter prototyping**
  - Docker run…
- **Lighter versioning**
  - Docker build…
- **Lighter migration steps** *(setup and forget)*
  - Installation ≠ Configuration
  - All-in-one container

  ! Host OS contraints due to IBM MQ versions
- **Useful for IT maintenance**
  - Docker Management console (no user access required)
  - Docker Swarms (MGMT of docker servers farms)
- **Entry costs**
  - Mandatory RHEL V7 (req. for MQ v9)
  - Docker Server deployment on IT infrastructure
  - Docker Universal Control Plane (UCP) deployment
  - Docker Trusted Registry deployment
  - …

AIRFRANCE KLM