MQ, IIB, Docker, Kubernetes & IBM Cloud

MQ in Containers

IBM CHAMPION

MQ in the Cloud Content

- Containers
- Container Software Stack
- Open Systems Interconnection (OSI) Layers
- Differing Perspectives
- Challenges of porting networks into Containers
- Summary

MQ in the Cloud - Containers

Containers

The new Application Runtime Environment

Containers

What are Containers?

- They are a type of Virtual Machine.
- They are very lightweight; more like a “Virtual Thread”.
- They all are based on Linux.

Where did Containers Come From?

- First released in 2013.
- Adopted by Amazon in 2014.
- 2016 Contributors:
  - Docker, Cisco, Google, Huawei, IBM, Microsoft, Red Hat.

Why are Containers Important?

- Standardized configuration allows “run anywhere” behavior.
- Enable massive horizontal scaling.
- Foundational technology for Amazon, Microsoft, & IBM Clouds.

How Containers are Built

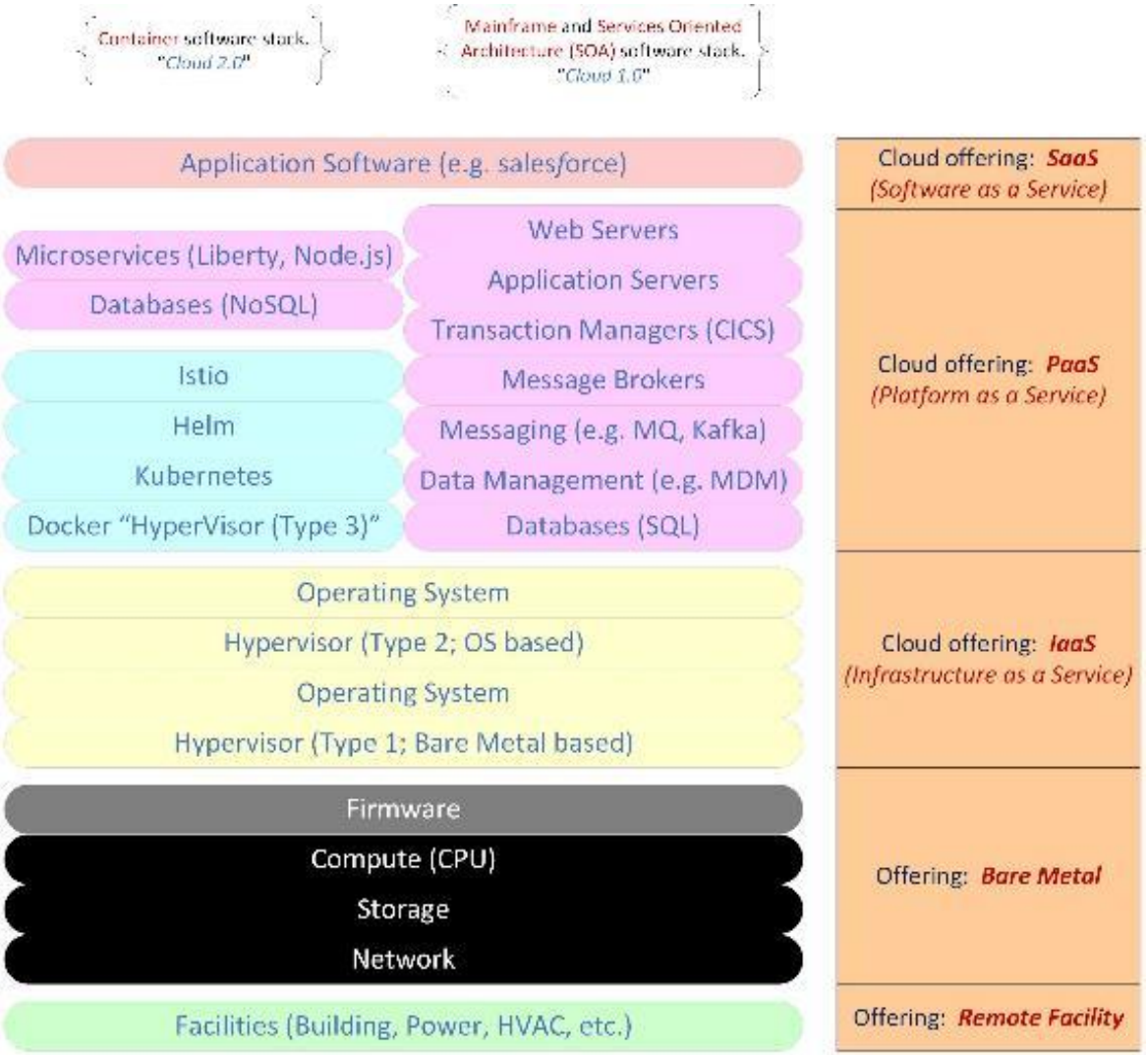
- **Containers are defined by a "Dockerfile"**
  - The Dockerfile is a build script
  - The Dockerfile defines a container as a series of layers
    - The Initial layer is required to be a Linux image
    - The second layer could be, for example, the MQ binaries
  - Containers have a defined command/script to be executed at startup
- **Two Methods for adding a Queue Manager to a Container**
  - Add a named Queue Manager and it's MQ objects as a layer
    - But every instance of the container would contain the same Qmgr!
  - Define a new Queue Manger in the startup script
    - But how to define communications to and from the Qmgr?
  - To cover this dilemma in more depth, additional background is needed

MQ in the Cloud – Cloud Software Stack

Container Software Stack

It's a whole new ballgame

Cloud Software Container Stack



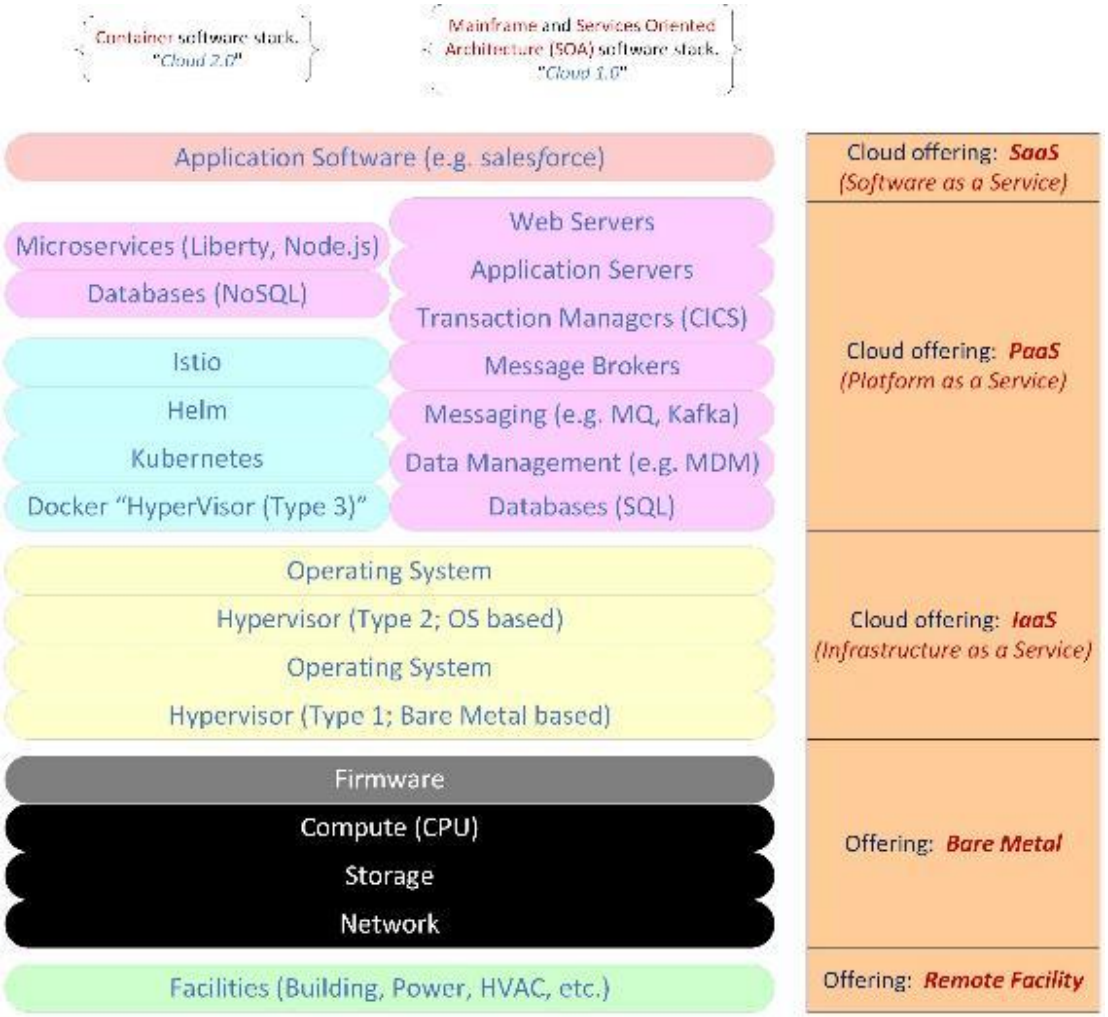
- **Remote**
  - ✓ *Facilities*
- **Bare Metal**
  - ✓ + *Network*
  - ✓ + *Storage*
  - ✓ + *Compute*
- **IaaS**
  - ✓ + OS
- **PaaS**
  - ✓ + DB & Mgmt
  - ✓ + Middleware
  - ✓ + App Hosting
- **SaaS**
  - ✓ + Application

Container Stack

- **Docker**
  - The engine that runs a container (a new kind of hypervisor)
  - Dockerfile defines which Ports of the Container are exposed
  - Provides communication support within a server
- **Kubernetes**
  - Kubernetes is a Container manager
  - Monitors health and restarts containers
  - Provides dynamic horizontal scaling of containers
  - Provides communications support across servers
- **Helm**
  - Helm is a Kubernetes package manager
- **Istio**
  - A Services Mesh
  - Provides communications support through a Control Plane

Docker

- Developed by Solomon Hykes
- Released in 2013
- Uses Linux features
  - cgroups
  - Namespaces
  - "Union" file system
- Union file system
- Open Source
  - Open Container Initiative
  - Cloud Native Computing Foundation



Docker Notes - I

N O T E S

- **Conceptual Framework**
  - Software executes in “Containers”
  - Containers are based upon native Linux capabilities
  - A Container is a single isolated & encapsulated thread
    - ✓ Everything necessary to execute (i.e. libraries)
  - A Container is a run-time instance of an “Image”
    - ✓ Images stored in Docker registries
- **Containers are managed by a daemon**
  - dockerd (Docker container daemon)
  - containerd (Open Source container daemon)
  - Container isolated from all other non-kernel processes
  - Scope of daemon is only server wide

Docker Notes - II

N

O

T

E

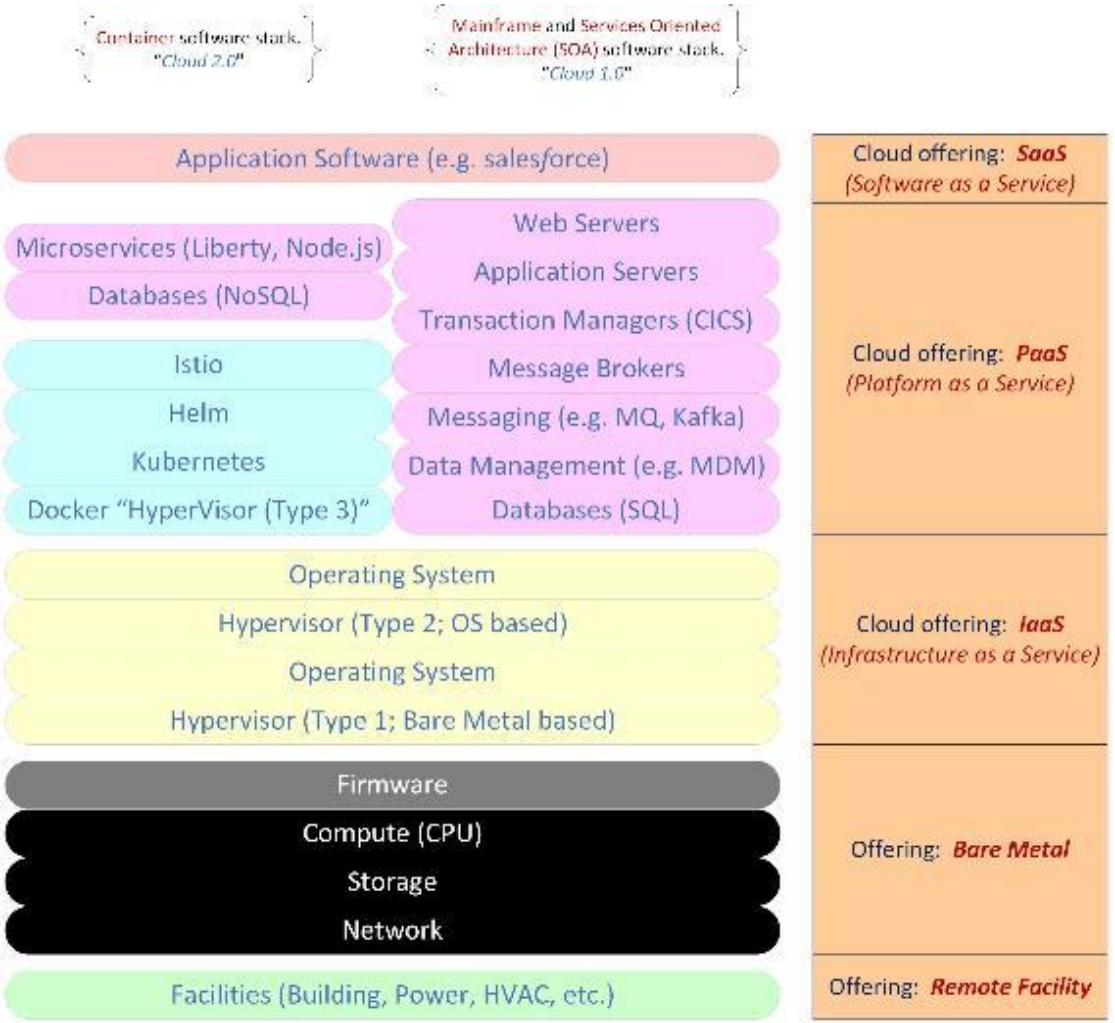
S

❖ Virtual Machines versus Containers

- ❑ Virtual Machines
  - ✓ Implement a “virtual” Operating System
  - ✓ General purpose
  - ✓ Multi-threaded
  - ✓ Shared resources for multiple processes
  - ✓ Slow to start up and shut down
- ❑ Containers
  - ✓ Implement a “virtual” Thread
  - ✓ Execute a single program
  - ✓ Single-threaded (Single Linux thread)
  - ✓ Resources dedicated to the software image
  - ✓ Extremely fast to start up and shut down

Kubernetes

- Developed by Google
- Released in 2015
- Turned over to the Cloud Native Computing Foundation (CNCF)
- ”Clustering for Containers”
- Docker Swarm and Apache Mesos are competing products



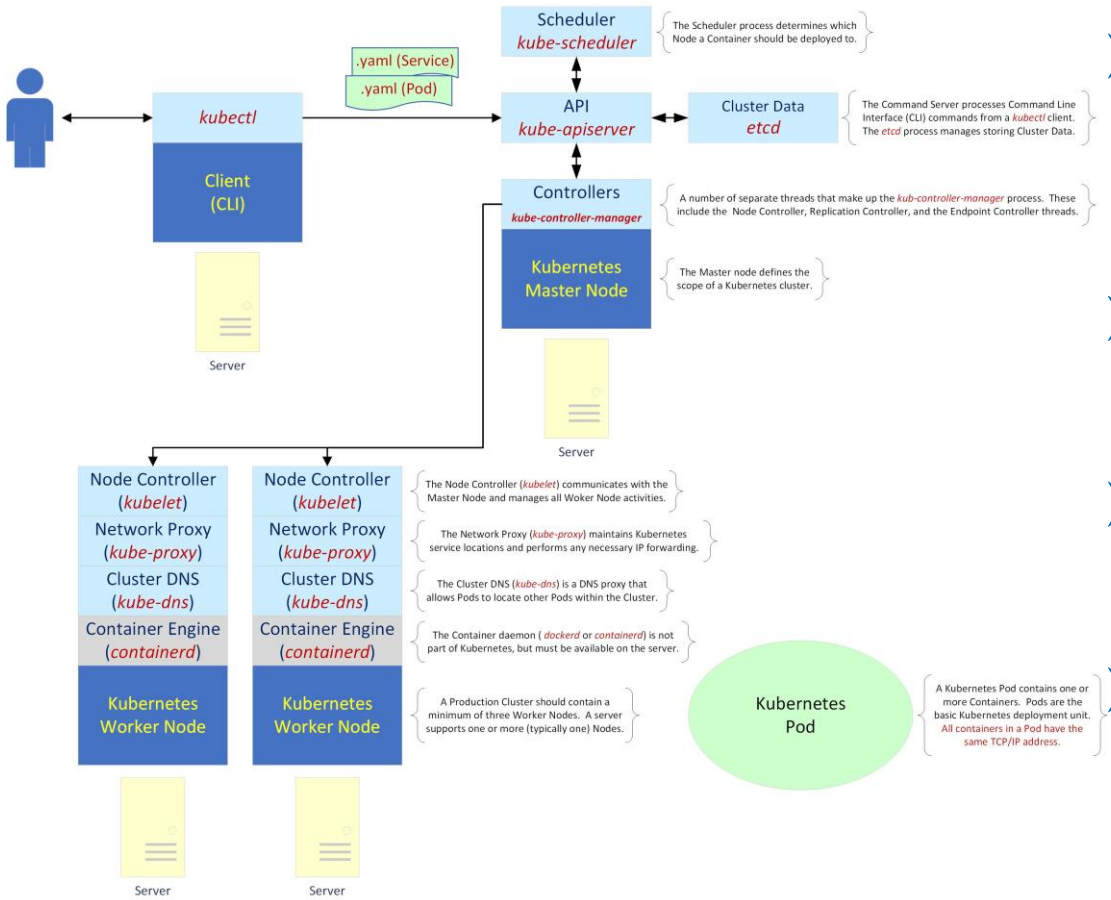
Kubernetes Notes

N O T E S

- **Container Orchestration**
  - Cluster Management
  - Container Scheduling
  - Service Discovery
  - Dynamic Scaling (Managing Container instances)
  - Health Maintenance (Health Checking & Repair)
- **Single Docker instance only spans one server**
- **Kubernetes deploys “Pods” of Containers**
  - Pods contain one or more containers
  - Pod instances deployed across multiple servers
  - Number of Pod instances monitored and managed

Kubernetes Architecture

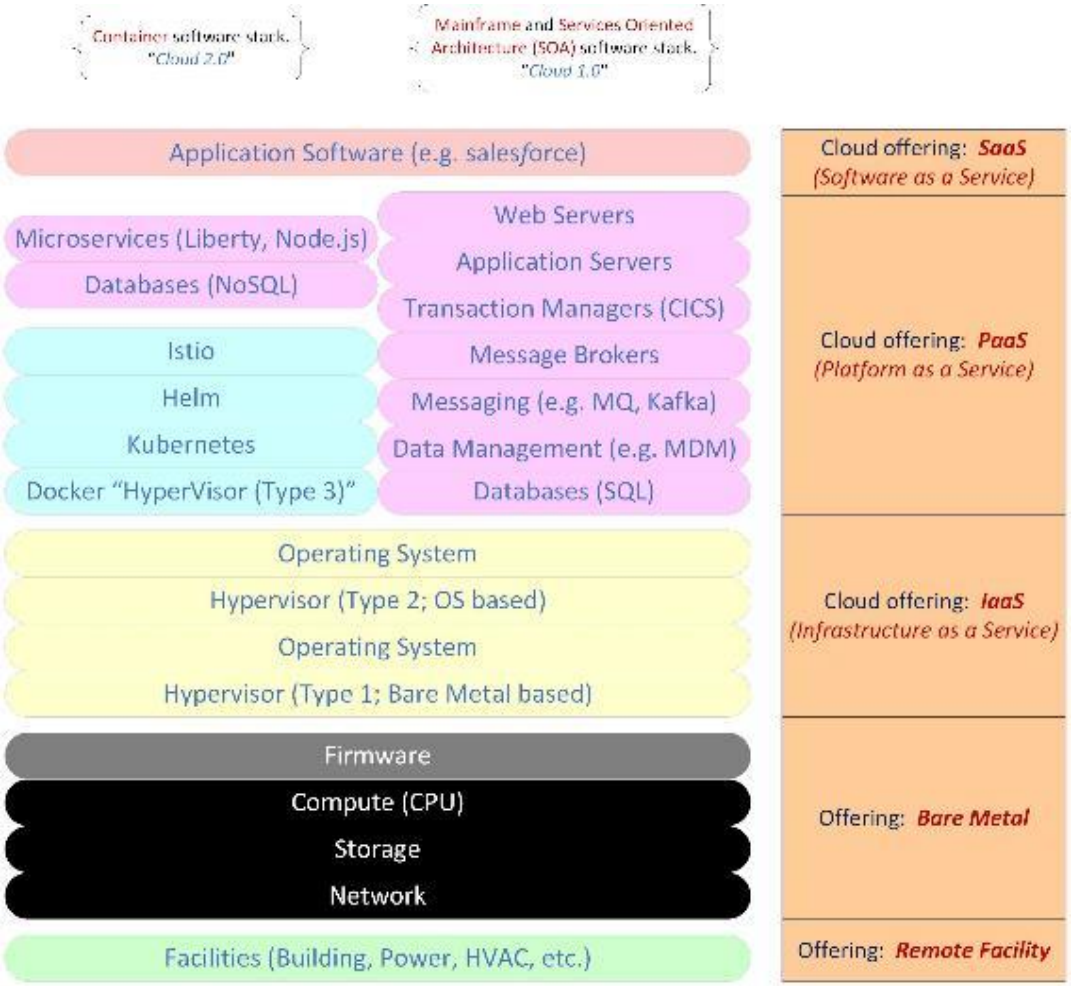
N O T E S



- Kubernetes Cluster defined by Master node.
- Pods distributed across Worker nodes.
- Client control interface.
- Defined Pods and Services.

Helm

- Developed at Deis
- Released in 2015
- ”Packaging for Kubernetes”
- Turned over to the Cloud Native Computing Foundation (CNCF)
- Initial development started with a short Deis hackathon



Helm Notes

N O T E S

- **Package Manager for Kubernetes**
  - Provides "Helm" Charts
    - ✓ A Helm Chart is a zipped directory (chart name = directory)
    - ✓ Package multiple Kubernetes components into one chart
      - Pods
      - Services
      - Ingress
      - Volumes
    - ✓ Separate Manifest data from Environment data
    - ✓ Charts can be stored and versioned in a repository
    - ✓ A "Release" is an instance of a Chart
  - Simplifies managing deployments
    - ✓ Combines multiple Kubernetes actions into a single chart
    - ✓ Creates a single reusable set of deployed objects (manifest)
    - ✓ Isolates Environment settings for simplified deployment migration (e.g. from Development to Production)

Helm Directory Structure

N

O

T

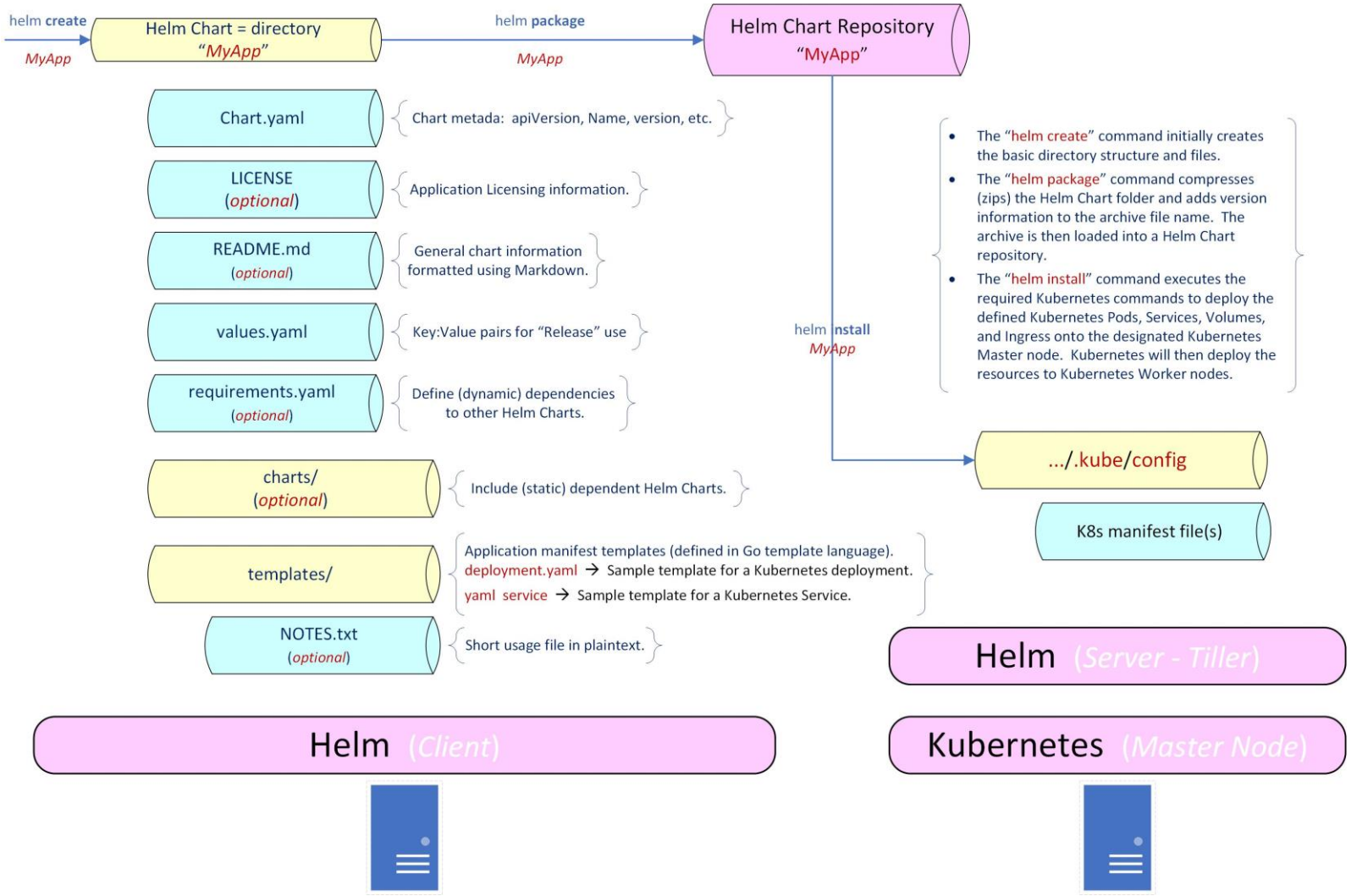
E

S

- **Helm Chart Directory**
  - Chart.yaml (*Chart metadata; YAML format*)
  - LICENSE (*L*) - *optional*
  - README.md (*Text file formatted using Markdown*) - *optional*
  - templates (*Resource manifests; Directory*)
    - NOTES.txt (*Text file*)
    - _helpers.tpl (*Text file*)
    - configmap.yaml (*YAML file*)
    - deployment.yaml (*YAML file*)
    - pvc.yaml (*YAML file*)
    - secrets.yaml (*YAML file*)
    - svc.yaml (*YAML file*)
  - values.yaml (*Release Keys and Values; YAML format*)

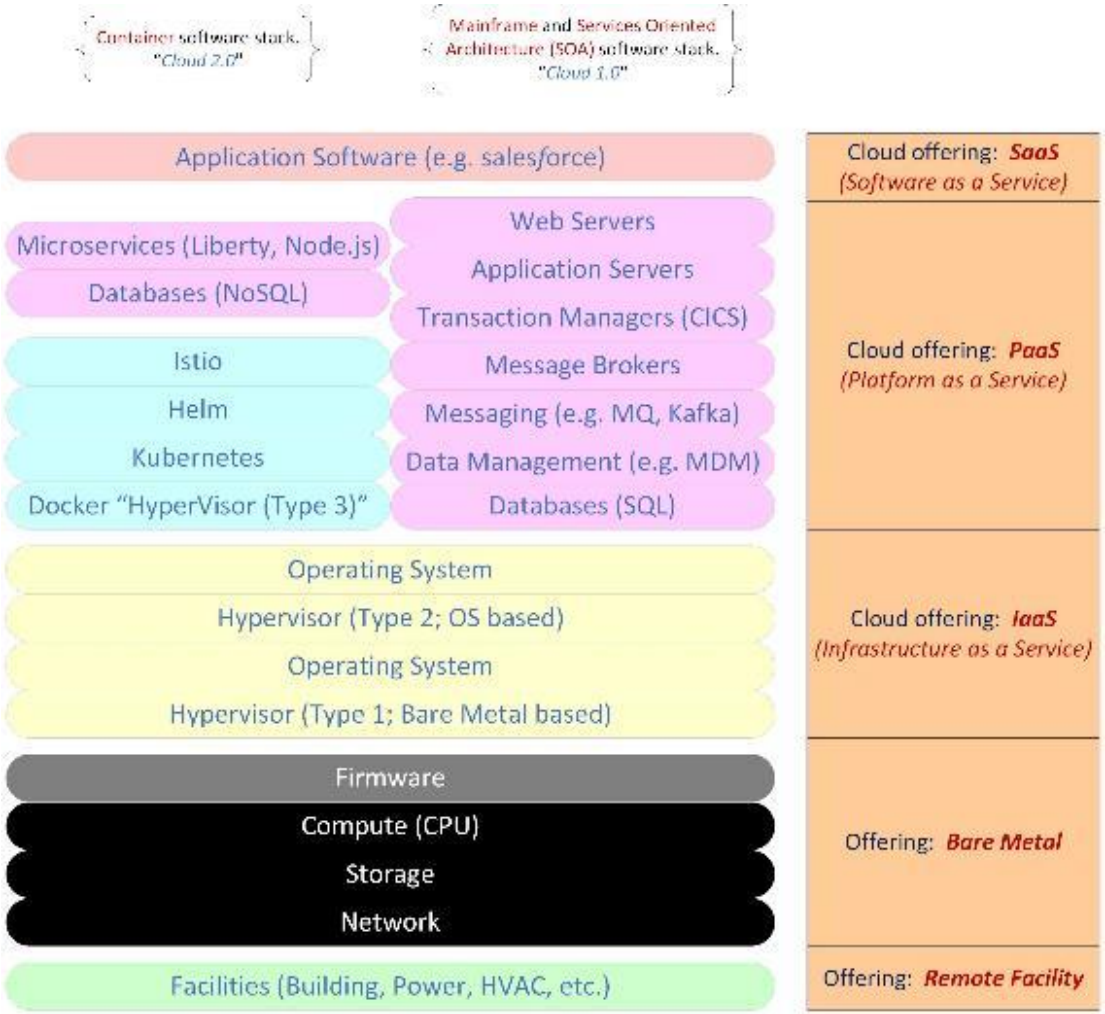
Helm Architecture

N O T E S



Istio

- Developed by IBM, Google, & Lyft
- Released in 2017
- Service Discovery ("Dynamic DNS") for the Cloud
- Consolidation of the Amalgam8 (IBM), Service Control (Google), and Envoy Proxy (Lyft) projects



Istio Notes

N O T E S

- **The Problem:**
  - How can the location of a Service be determined?
- **The Answer:**
  - A Service Mesh
    - ✓ Envoy Proxies are added as “sidecars” to Docker containers
    - ✓ These sidecars are deployed as part of the Kubernetes Pod
    - ✓ TCP requests routed through the Proxies.
    - ✓ Proxies announce their existence to the “Control Plane”
      - This allows them to receive inbound traffic
    - ✓ Proxies route their requests through the “Control Plane”
      - This allows them to receive inbound traffic
    - ✓ Control Plane may also enforce Policies (Security, Traffic, etc.)

Istio Architecture

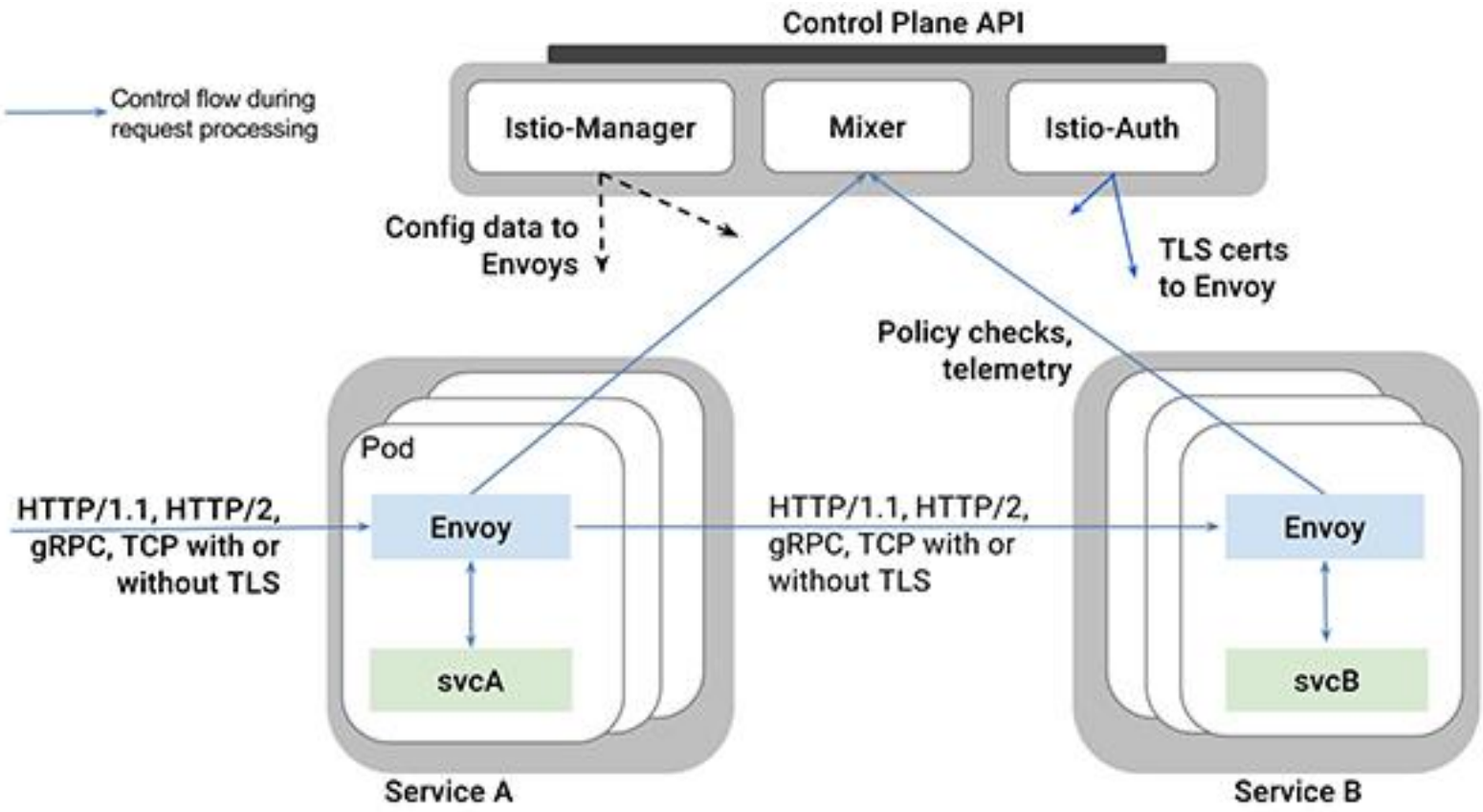
N

O

T

E

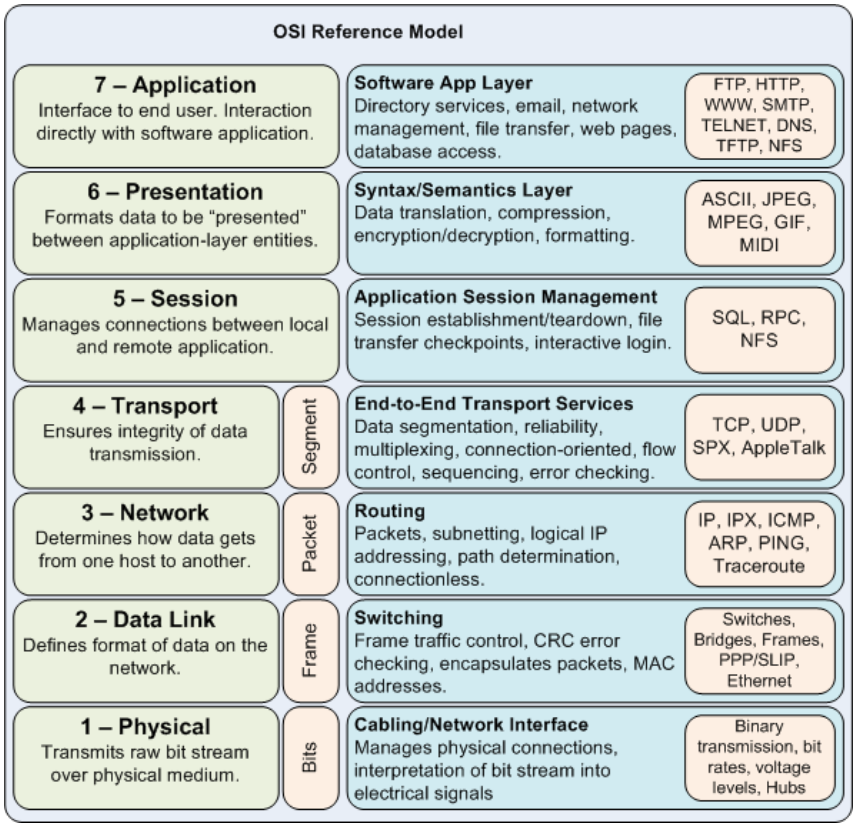
S



Open Systems Interconnection Layers

Decomposing the Software Stack

Open Systems Interconnection Layers



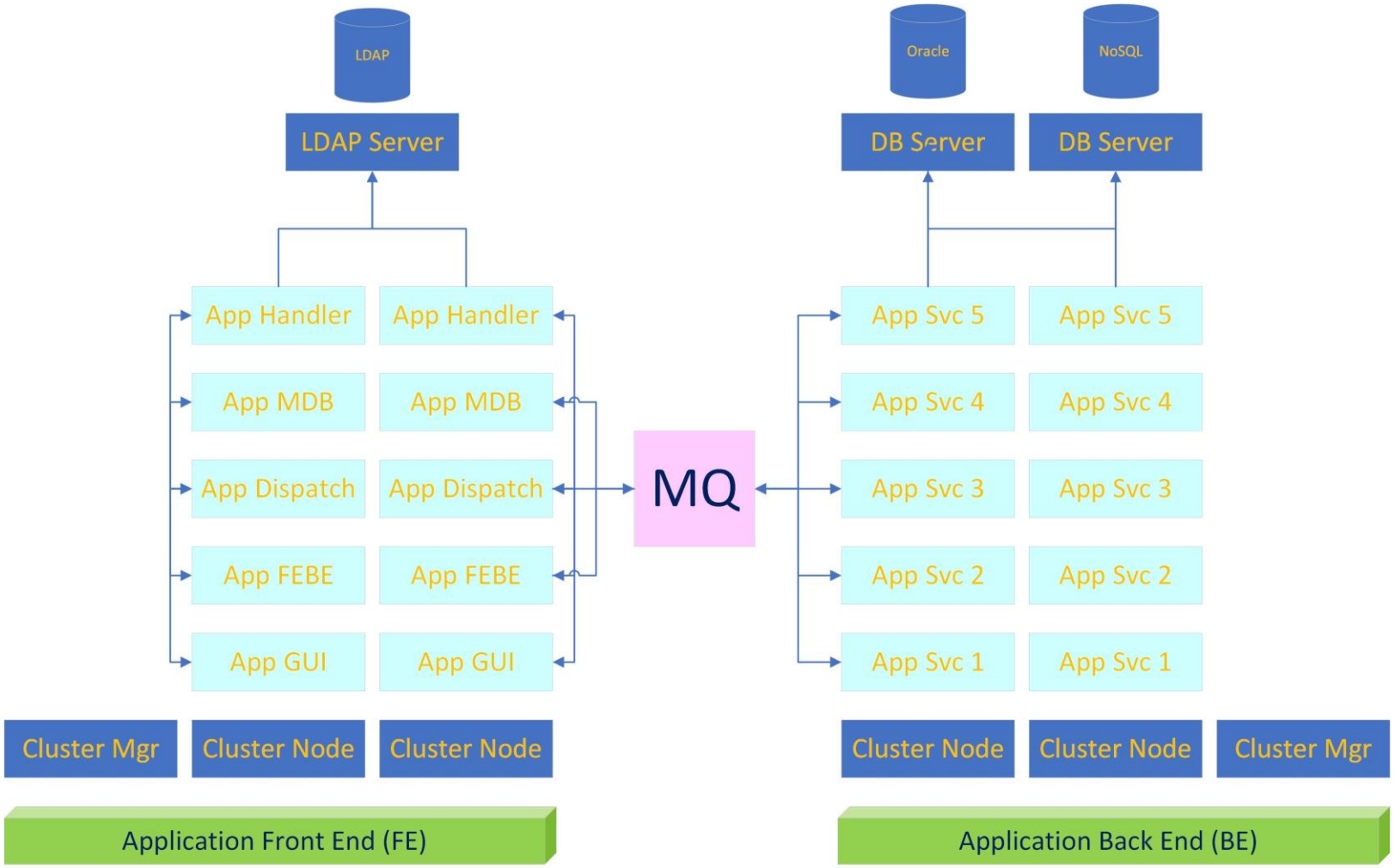
- Application perspective is OSI Layer 7
- Apps are all about "function"
- MQ is an OSI Layers 4 & 5 product
- Reasoning that what is "good" for Layer 7 will be good for other layers IS NOT VALID!

MQ in the Cloud - Containers

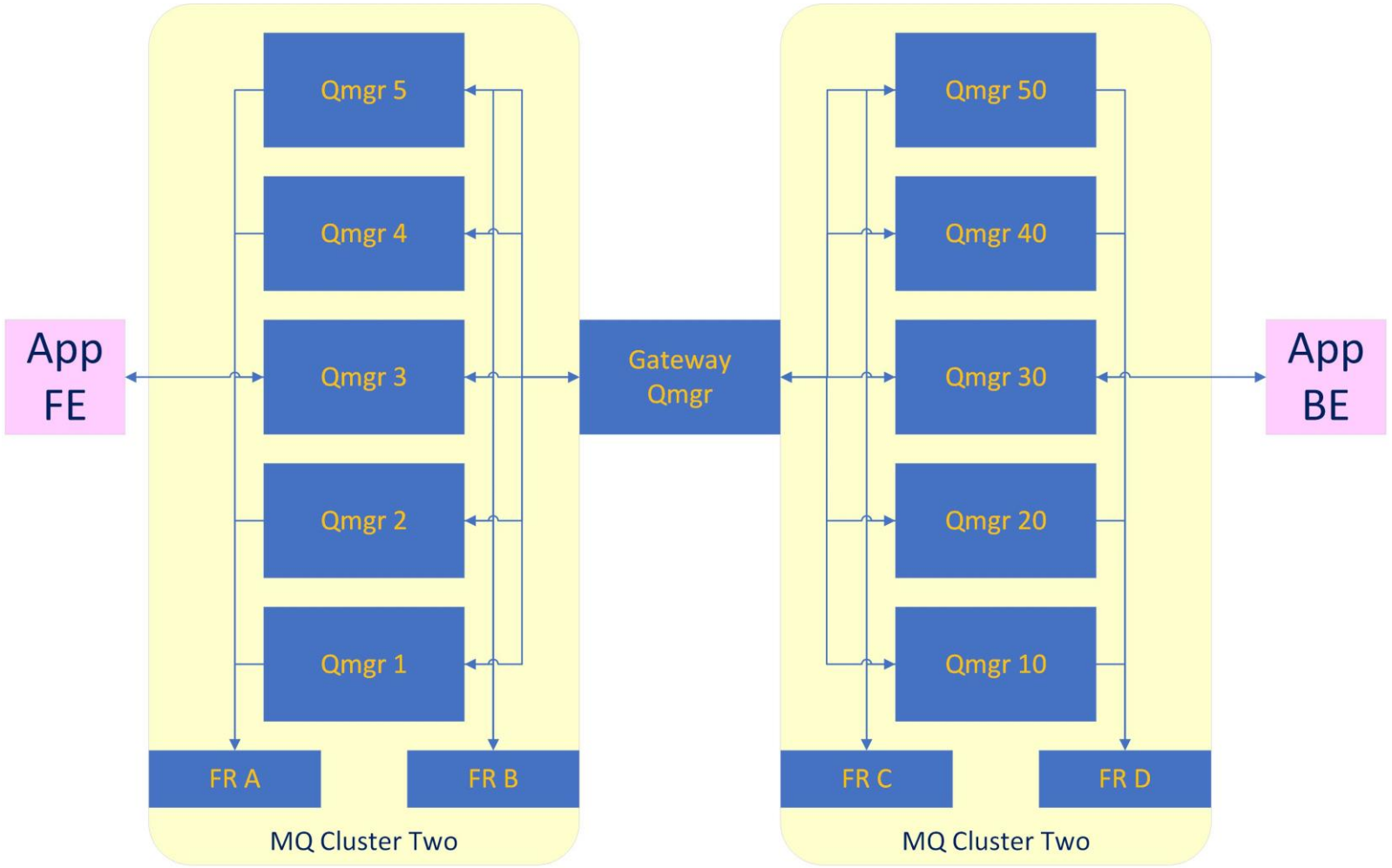
Differing Perspectives

What you don't know seems simple

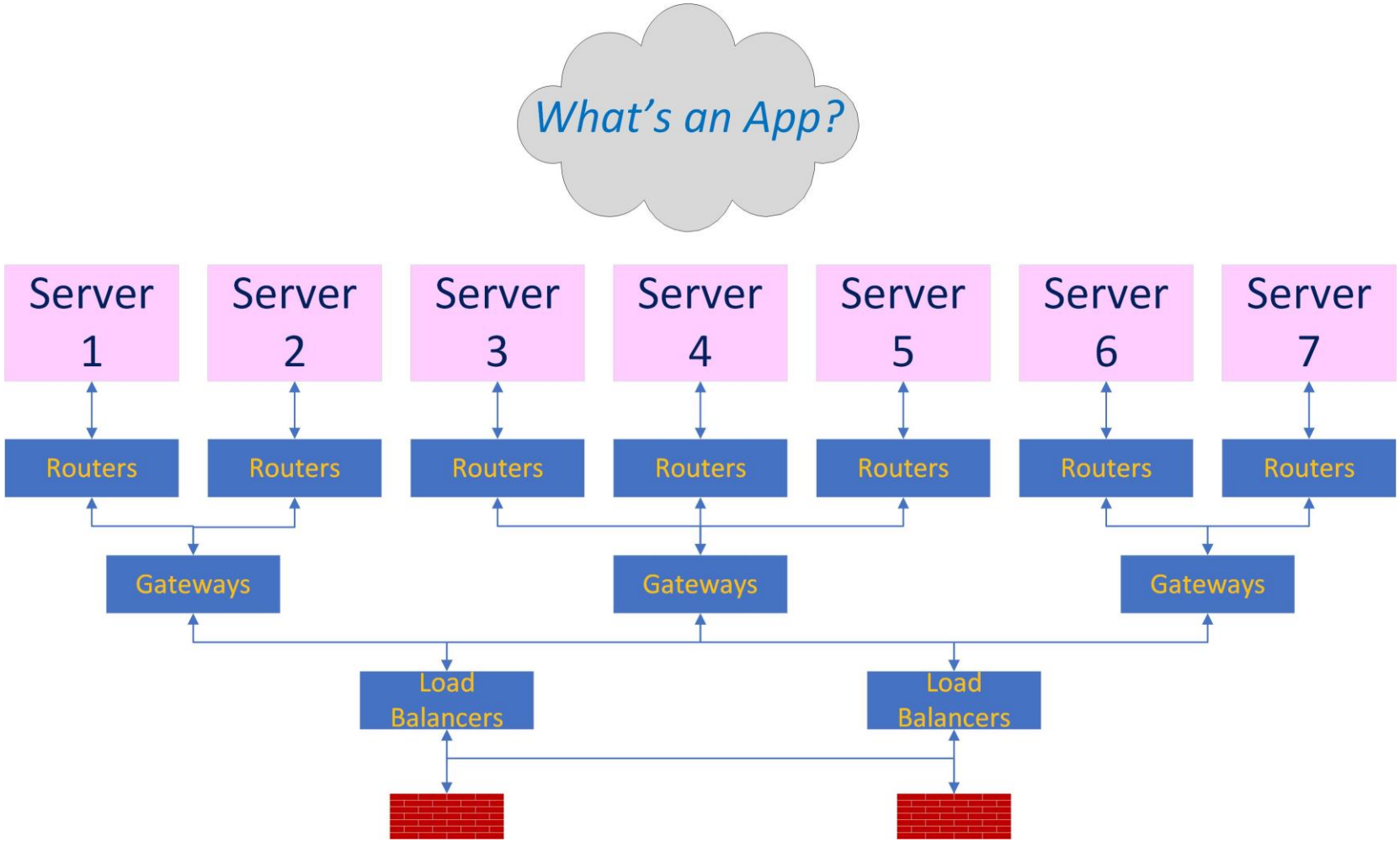
Application View of MQ



MQ View of Applications



Network View of Servers



Perspectives - I

❖ Application Developers

- Containers provide a standardized runtime environment (through Dockerfiles)
  - Every Container is, by definition, configured identically
- Containers provide horizontal scaling (through Kubernetes)
- Containers provide HA (through Kubernetes)
- Influenced by Web Front-ends using clustered horizontal scaling

❖ Enterprise Architects

- Back-end software may also leverage horizontal scaling
- Read-only resources may be horizontally scaled (multiple instances)
- Single-instance resources, however, are placed in their own container
  - Transactional Databases may only have one logical copy

Perspectives - II

❖ MQ Administrators

- See Application Front-Ends
- See Application Back-Ends
- See MQ as a Network, not as a Service

❖ Network Engineers

- See the actual network
- Understand routing and load balancing AT THE NETWORK LEVEL
- **Do Not** see routing and load balancing AT THE SERVER LEVEL

Perspectives & Skills

❖ Multiple roles required to deploy Cloud Applications

- Cloud developer (Programmer) & Cloud deployer (DevOps)
- Infrastructure, Middleware, & Cloud Administrators
- Network Engineers & Security Administrators

❖ You Don't Know What You Don't Know

- You know your owner complexities
- You may not know, or may underestimate, complexities of other roles
- It takes a village to nurture software and the village is growing

❖ MQ Administrators MUST learn about the Cloud

- It's NOT just another computing location
- It's an entirely new Software Stack; from top to bottom
- The only thing in common is Linux, but even that doesn't translate
- Network knowledge is even more important now

Challenges of Porting Networks into Containers

A Square Peg in a Round Hole

MQ in the Cloud (in Containers) - I

- **MQ designed for a server based environment**
  - Communications through DNS or TCP addresses
  - Server addresses (both DNS and IP) relatively static
  - Server address changes not expected to update in real time
- **Containers designed for a very dynamic environment**
  - Container instances continuously created and destroyed
  - Container instances running across multiple servers & data centers
  - Most containers don’t need to persist data
  - Data persistence requires shared disk
- **MQ Challenges**
  - Channel definitions (CONNAME)
  - Cluster membership
  - Queue Manager location for Applications to read messages

MQ in the Cloud (in Containers) - II

❖ Three patterns of Application Communications

- Asynchronous Writers/Publishers
- Request/Reply processing
- Asynchronous Readers/Subscribers

❖ MQ Writer/Publisher and Request/Reply Processing

- From an Application perspective, MQ provides all of the “magic” to make Asynchronous Writes and Request/Reply processing work!
- Therefore, these seem logical to Application developers and architects to containerize.
- But how will these containers communicate with other Qmgrs?
  - Built-in Sender channels?
  - How about, if needed, Receiver channels?

MQ in the Cloud (in Containers) - III

❖ MQ Reader/Subscriber and Processing

- From an Application perspective, how can the Application connect to an unknown Queue Manager at an unknown location?
- But how will these containers communicate with other Qmgrs?
  - Built-in Sender channel definitions assume destination stability.
  - How about, if needed, Receiver channels? How would the corresponding Sender channels be defined?

❖ All Queue Managers

- Persisting messages requires usage of shared disk
- Shared disk limits location of servers that can host the container
- Much closer integration with network configuration
  - e.g. F5 Global Traffic Management (GTM) DNS routing
  - e.g. FT Local Traffic Management (LTM) Load Balancing

IIB/ACE in the Cloud (in Containers)

- Both MQ and TCP Design considerations
- If MQ is only used locally by IIB/ACE then no issues
- If MQ connects with other Qmgrs, then same MQ issues
- TCP issues can be handled by incoming Load Balancing
- TCP based services require a “Global” front-end address
- Standard Istio Service Mesh processes could handle local Service registration

MQ in the Cloud - Containers

Summary

Anchored at the shore of the New World

The Potential for Containers

❖ High Availability

- A specific Queue Manager, using shared disk, could potentially run on any server capable of connecting to the shared disk
- Kubernetes managed HA would potentially seem to be highly desirable

❖ Horizontal Scaling (up to Extremely Large scales)

- Already possible for some MQ Communication Patterns
  - Asynchronous Writer/Publisher
  - Request Reply
- Already possible for IIB TCP based Services
  - Global Service Address
  - Local registration using Istio

Questions & Answers

Presenter

- Glen Brumbaugh
  - Glen.Brumbaugh@TxMQ.com
- Computer Science Background
  - Lecturer in Computer Science, University of California, Berkeley
  - Professorial Lecturer in Information Systems, Golden Gate University, San Francisco
- WebSphere MQ Background (25 years plus)
  - IBM Business Enterprise Solutions Team (BEST)
    - Initial support for MQSeries v1.0
    - Trained and mentored by Hursley MQSeries staff
  - IBM U.S. Messaging Solutions Lead, GTS
  - Platforms Supported
    - MVS aka z/OS
    - UNIX (AIX, Linux, Sun OS, Sun Solaris, HP-UX)
    - Windows
    - iSeries (i5OS)
  - Programming Languages
    - C, COBOL, Java (JNI, WMQ for Java, WMQ for JMS), RPG


Thank
You