RED HAT SUMMIT

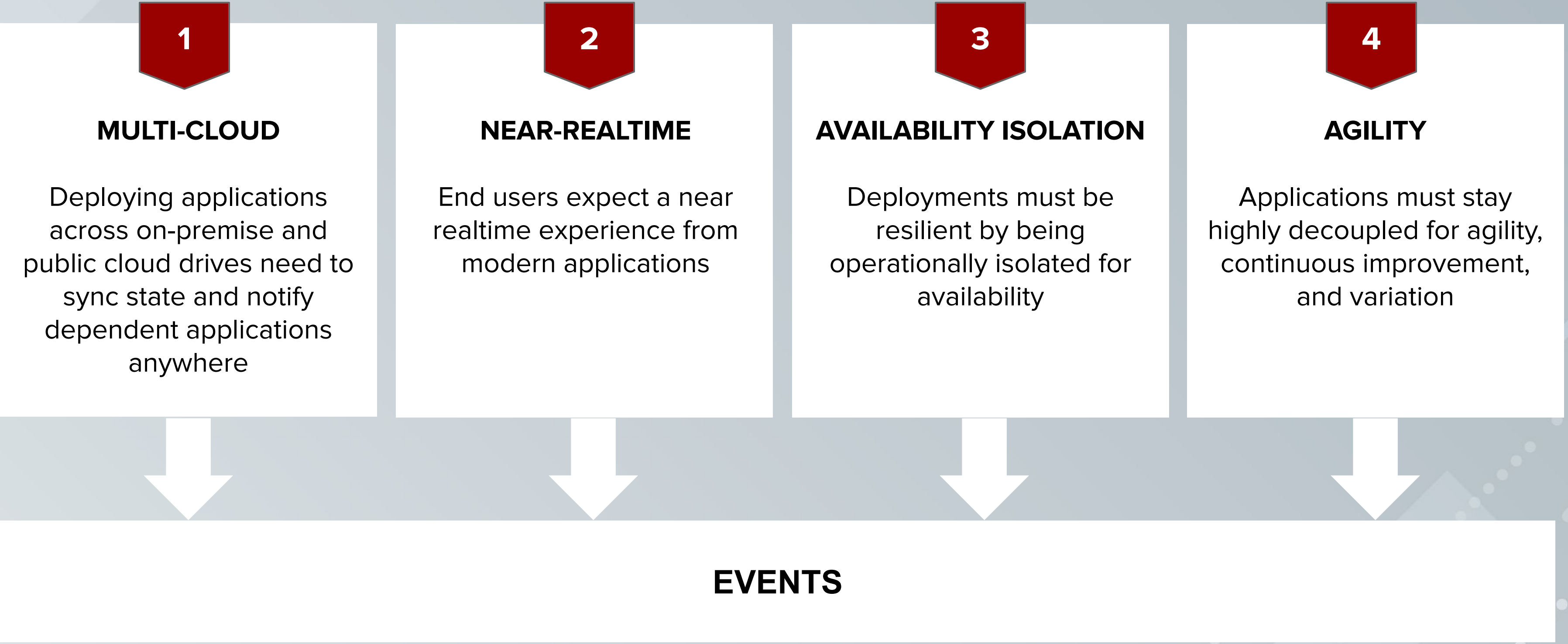
CREATING A GLOBALLY DISTRIBUTED EVENT MESH

MAY 2019

MARIUS BOGOEVICI
Principal Specialist Solution Architect

JOSH WEST
Chief Account Architect, Global Financial Services

RISE IN EVENTS DRIVEN BY MODERN APP-DEV



1

MULTI-CLOUD

Deploying applications across on-premise and public cloud drives need to sync state and notify dependent applications anywhere

2

NEAR-REALTIME

End users expect a near realtime experience from modern applications

3

AVAILABILITY ISOLATION

Deployments must be resilient by being operationally isolated for availability

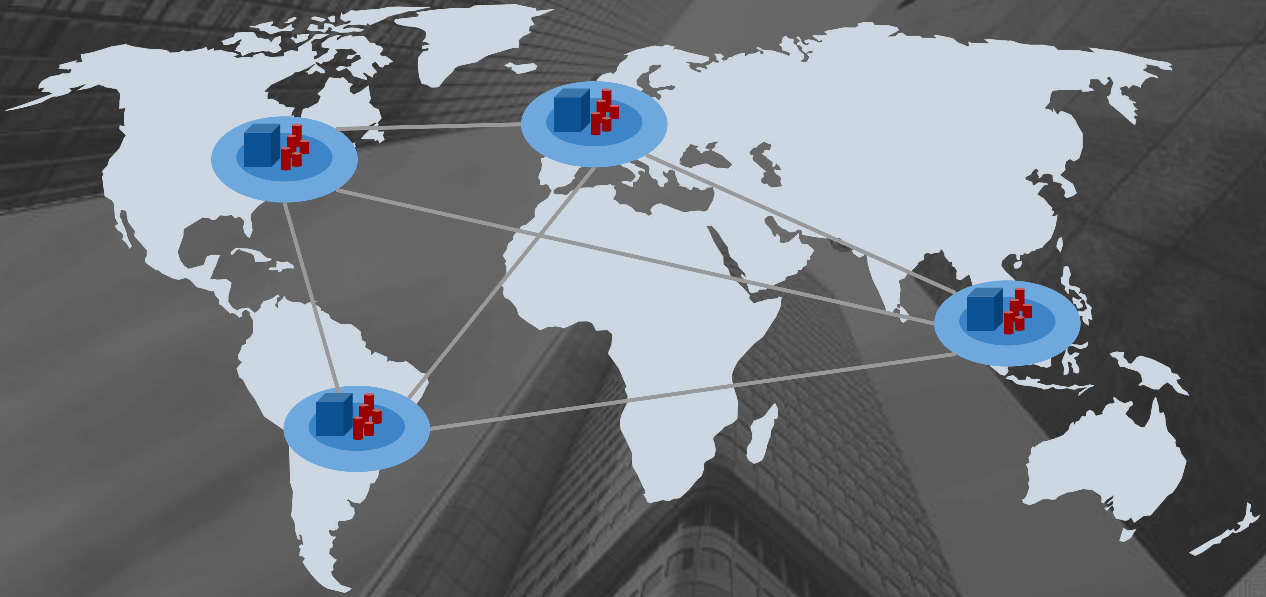
4

AGILITY

Applications must stay highly decoupled for agility, continuous improvement, and variation

EVENTS

GLOBAL NETWORK OF BANKING AND PAYMENTS SERVICES



NATURE OF INTERNATIONAL BANKING

GLOBAL and **REGIONAL** variation to business and technicals

COMPLEX MAZE of interconnected systems

INNOVATION, capabilities, and global expansion drive the TTS business growth

REPUTATION to be maintained as deserving of the business from the world's largest institutions

CORE BANKING tends to be the most connected system

Nearly 1000 applications covering channels, payments, liquidity, and risk & compliance

HIGH VALUE message payloads with severe risks for data loss

SEV1s must be reported to regulators

ORGANIC CHALLENGES WITH MESSAGING

1

- Applications get **coupled to physical** messaging infrastructure and locations
- Apps and teams change at various speeds and shared infrastructure needs to be **isolated**

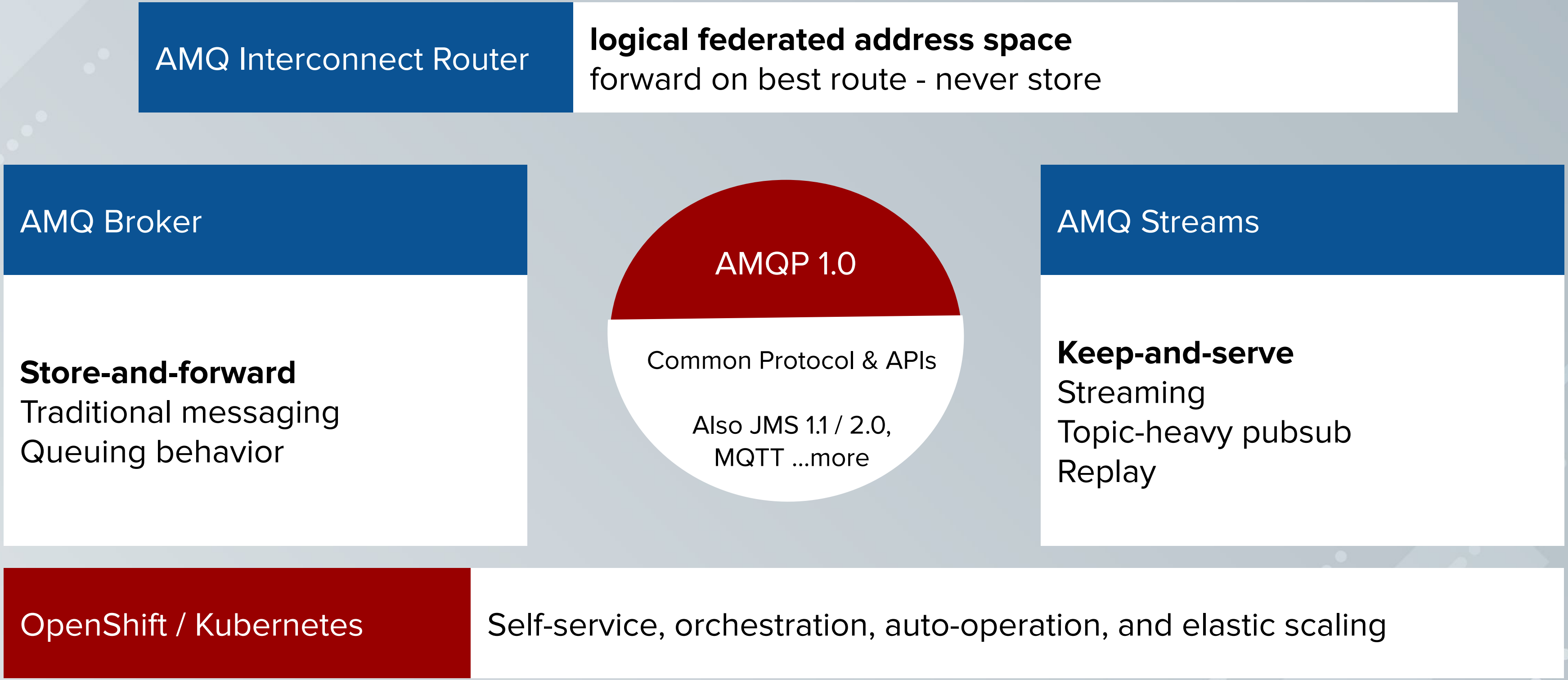
2

- Messaging brokers start to follow **conways laws** - aligning with the organization
- Messages are forwarded / shoveled across organization boundaries
- More shovelling occurs for transformation, isolation based on volumes, and legacy flows
- Special considerations are taken to prevent **circular/recursive routes**

3

- Messaging **governance** takes time and large investment
- App teams start to **avoid** eventing / messaging when they really should
- Messages have a long journey with **many hops** causing difficult **troubleshooting**

BUILDING BLOCKS



AMQ Interconnect Router

logical federated address space
forward on best route - never store

AMQ Broker

Store-and-forward
Traditional messaging
Queuing behavior

AMQP 1.0

Common Protocol & APIs

Also JMS 1.1 / 2.0,
MQTT ...more

AMQ Streams

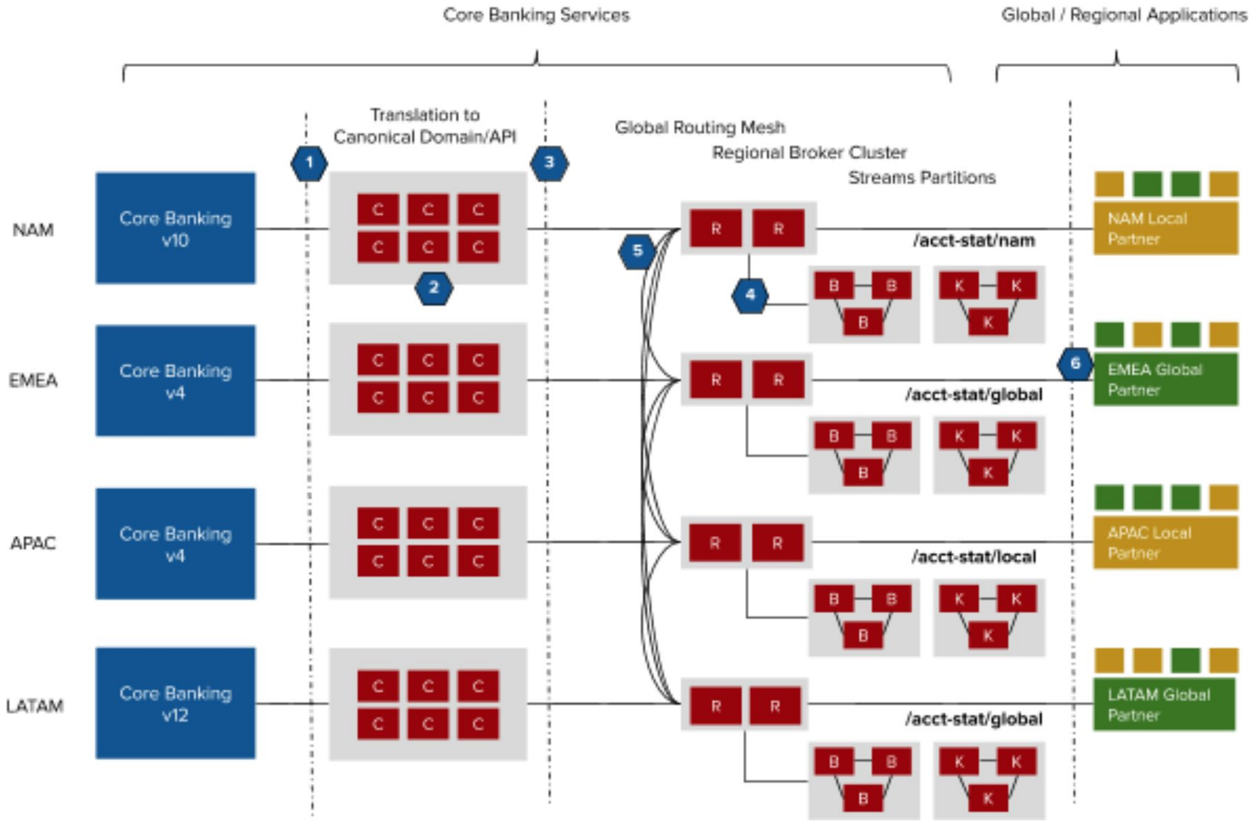
Keep-and-serve
Streaming
Topic-heavy pubsub
Replay

OpenShift / Kubernetes

Self-service, orchestration, auto-operation, and elastic scaling

END-TO-END

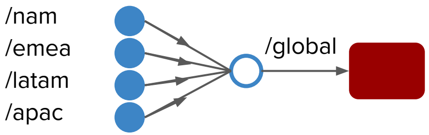
1. **Connect** to unique regional interfaces with integration components and EIPs
2. **Translate** to a canonical data model & API semantics
3. **Serve** up a consistent API to client applications following backwards compatibility and versioning best practices
4. **Deliver** messages reliably and durably over a horizontally scalable messaging broker cluster
5. **Route** and federate messages hiding the complexity and change management of access to the global network from clients
6. **Consume** via a single endpoint from a consistent and simplified address space with complexities hidden



C Fuse Integration (Camel)   R Interconnect Qpid Router   B Broker (Artemis/ EnMasse)   K Kafka Partitions

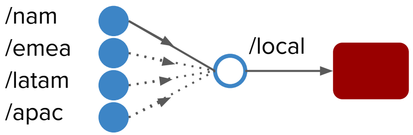
🟨 Local Partner Client Application   🟩 Global Partner Client Application

INTERCONNECT ADDRESSING PATTERNS



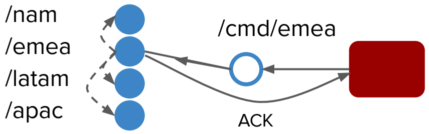
CONSUMER ON GLOBAL AGGREGATE ADDRESS

provide virtual address that aggregates physical message queues across multiple regions in a single address (eg: "/global"), allowing client applications to use a single endpoint and address to reliably receive events from across the global mesh



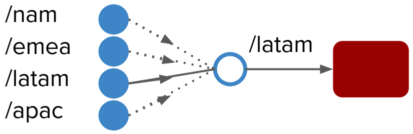
CONSUME ON LOCAL REGIONAL LINK

provide address space for local broker until a named "local' alias, allowing client applications to be programmed/configured homogeneously across regions but receive event contextual to the deployed location



PUBLISH AND DISTRIBUTE COMMAND

provide an address space for submitting commands for target region acknowledgement, allowing client applications to reliably submit mutator requests. This pattern requires a single region to own acknowledgement to a local or regional named address, and then may be distributed across regions.



CONSUME ON NAMED REGIONAL LINK

provide address space specific to a region by name, allowing client applications to subscribe events based on regional business context

● Broker ○ Router ■ Client Application

TRADITIONAL MESSAGING

VS

Advantages

- Store-and-forward
- individual message exchanges (transactionality, acknowledgment, error handling/DLQs), P2P/competing consumer support
- Publish-subscribe support with limitations

Trade-offs

- No replay support
- Requires fast and/or highly available storage infrastructure
- No total ordering

EVENT STREAMING

Advantages

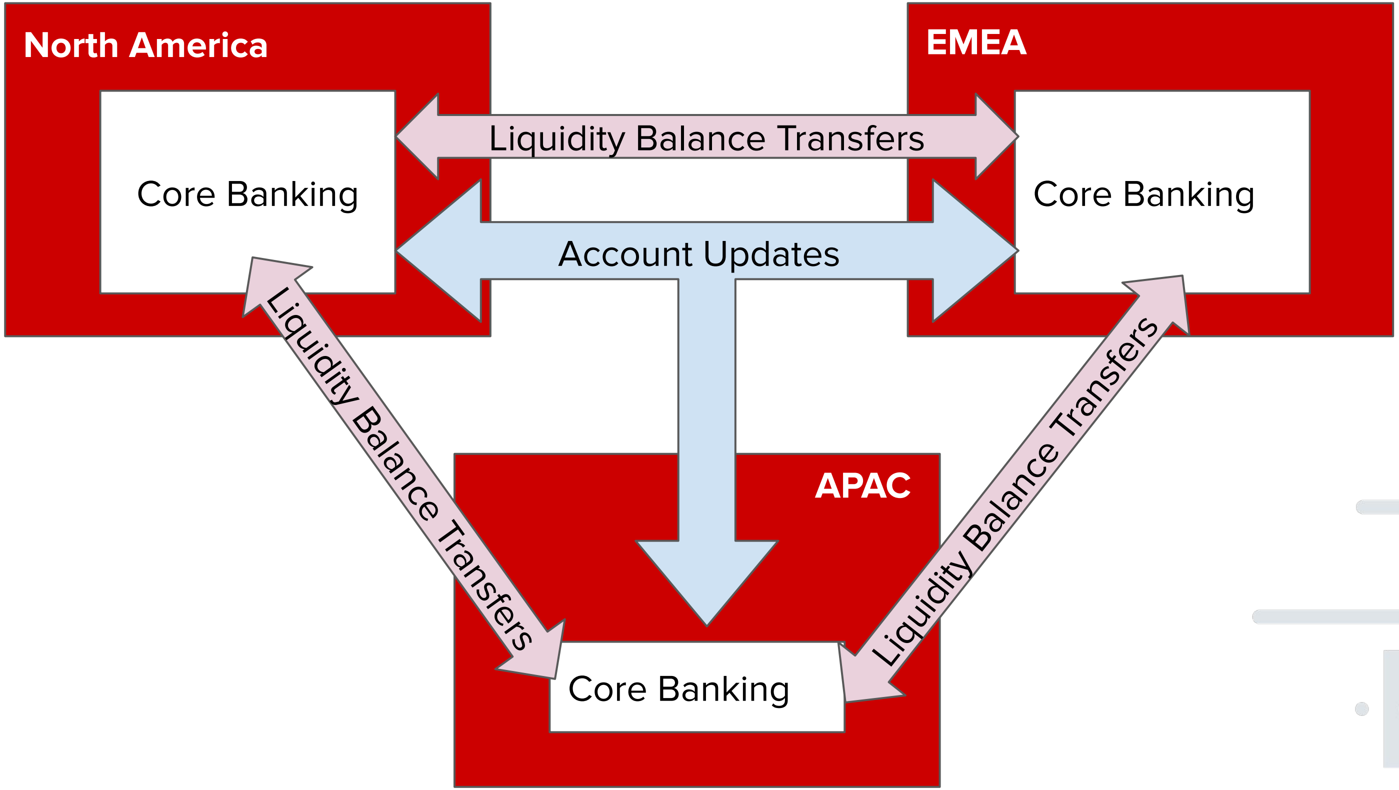
- long-term persistence, replay, semantic partitioning, large publisher/subscriber imbalances, replay and late-coming subscribers
- Shared nothing data storage model
- Total ordering

Trade-offs

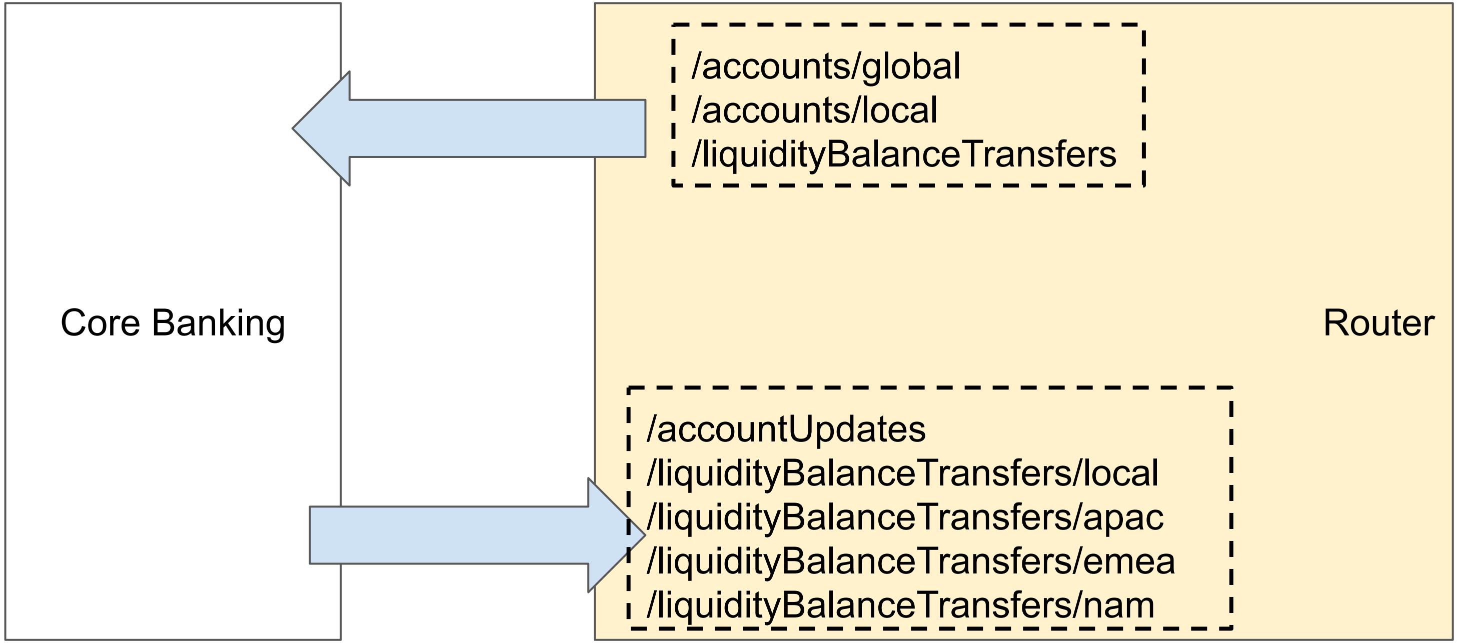
- Weak support for individual message acknowledgment, p2p/competing consumers
- Larger data footprint and extremely fast storage access

DEMO TIME!

AN EXAMPLE USE CASE

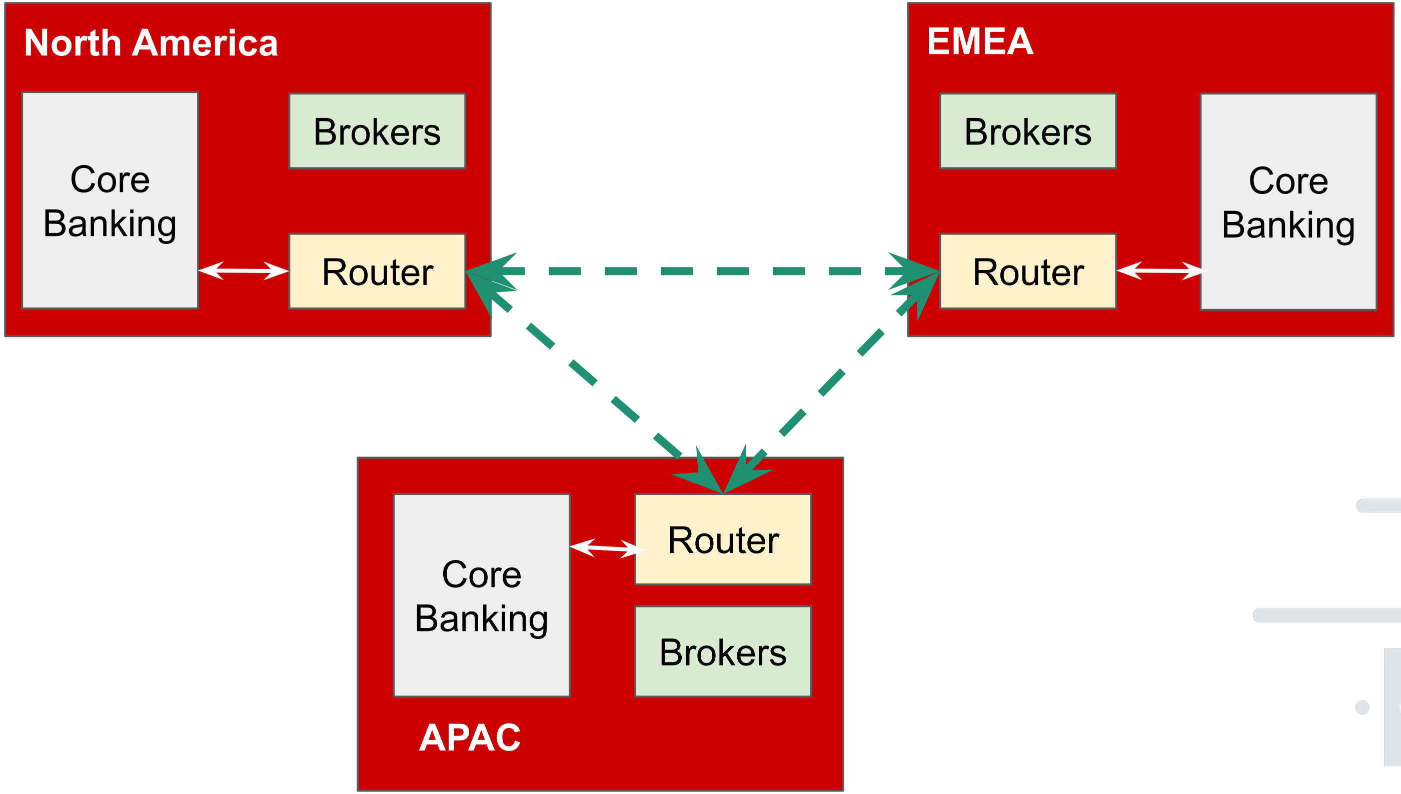

North America
Core Banking
EMEA
Core Banking
Liquidity Balance Transfers
Account Updates
Liquidity Balance Transfers
Liquidity Balance Transfers
APAC
Core Banking

PATTERN: LOCAL CONNECTION ENDPOINTS

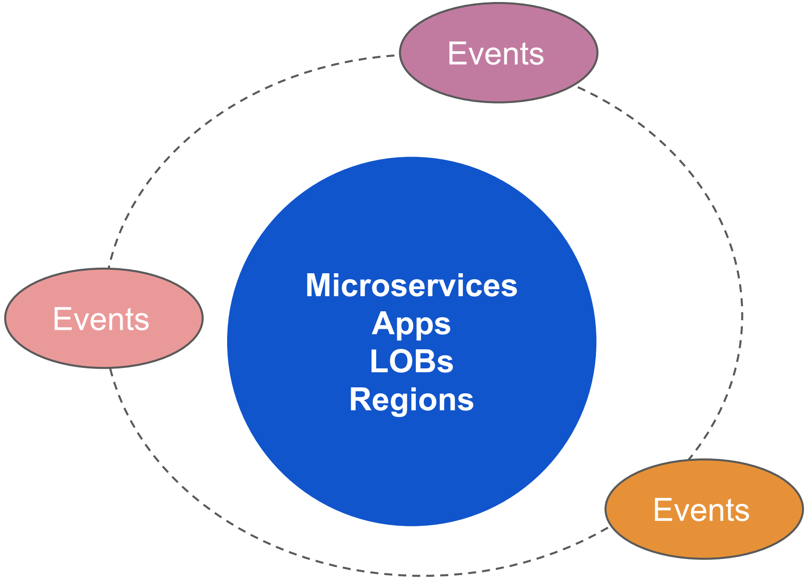


Core Banking
Router
/accounts/global
/accounts/local
/liquidityBalanceTransfers
/accountUpdates
/liquidityBalanceTransfers/local
/liquidityBalanceTransfers/apac
/liquidityBalanceTransfers/emea
/liquidityBalanceTransfers/nam

BUILDING THE MESH

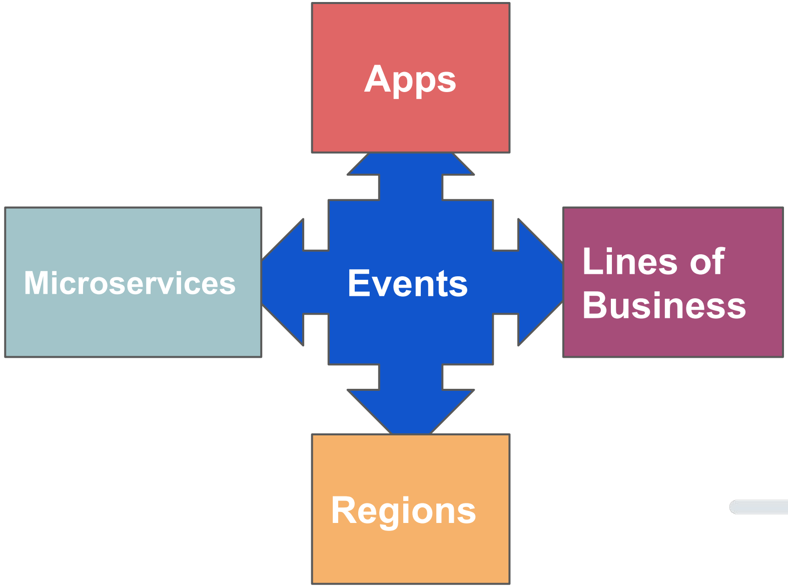


RETHINKING EVENT-DRIVEN ARCHITECTURE



System and data-centric

Events are designed to respond to ad-hoc connectivity needs



Event-centric

Events are first class citizens that describe the interactions in the enterprise

CONCLUSIONS

- Modern app dev is bringing **event-driven architecture** to prominence
  - Near-realtime interaction, extensibility, agility

- **Event distribution**, across multiple interconnected applications, data centers becomes a critical capability
  - **Traditional methods**, such as ETL/message do not scale

- Achievable through a combination of:
  - **Event-centric design**: making events first class citizens
  - **Routing messages with lightweight components** and virtualizing the messaging infrastructure for producers and consumers

Source Code: http://bit.ly/global-mesh-src

RED HAT SUMMIT

THANK YOU

in linkedin.com/company/Red-Hat

f facebook.com/RedHatinc

You Tube youtube.com/user/RedHatVideos

twitter.com/RedHat