DYNAMIC TOPOLOGIES TO DELIVER MICRO SERVICES, MICRO BROKERS, AND MICRO FLOWS

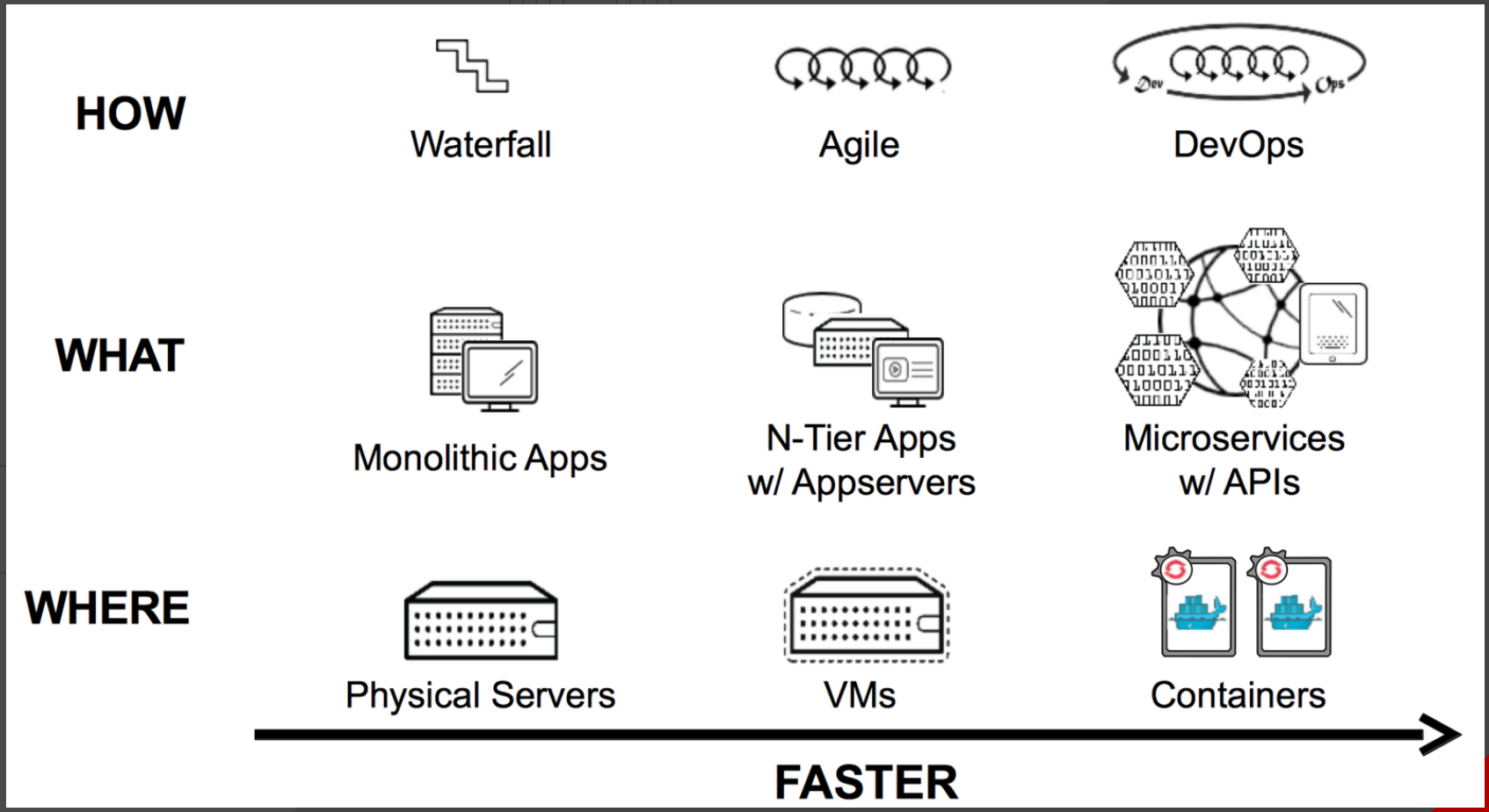
Ashwin Karpe

Apache Camel Committer, Camel PMC Member

and

Red Hat Enterprise Integration Practice Lead, North America

THE NEED FOR SPEED



MICROSERVICES ARCHITECTURE

AN ARCHITECTURAL PATTERN FOR DESIGNING APPLICATIONS AS

- a collection of capabilities, hosted in interesting ways
- designed to scale on demand with full location transparency
- capabilities organized around domains, namespaces and coordinates
- designed to discover and interact with integration endpoints and messaging systems
- identically provisioned, governed and managed using API's

TOPOLOGY BUILDING BLOCKS

MICROSERVICE

A fine-grained capability, containing everything from

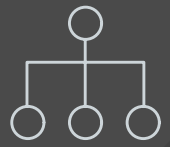
- *runtime and dependencies* *[JBoss Fuse]*
- *platform or framework* *[J2SE/JRE]*
- *software container* *[Docker, OpenShift]*

packaged as one unit of execution

MICROBROKER

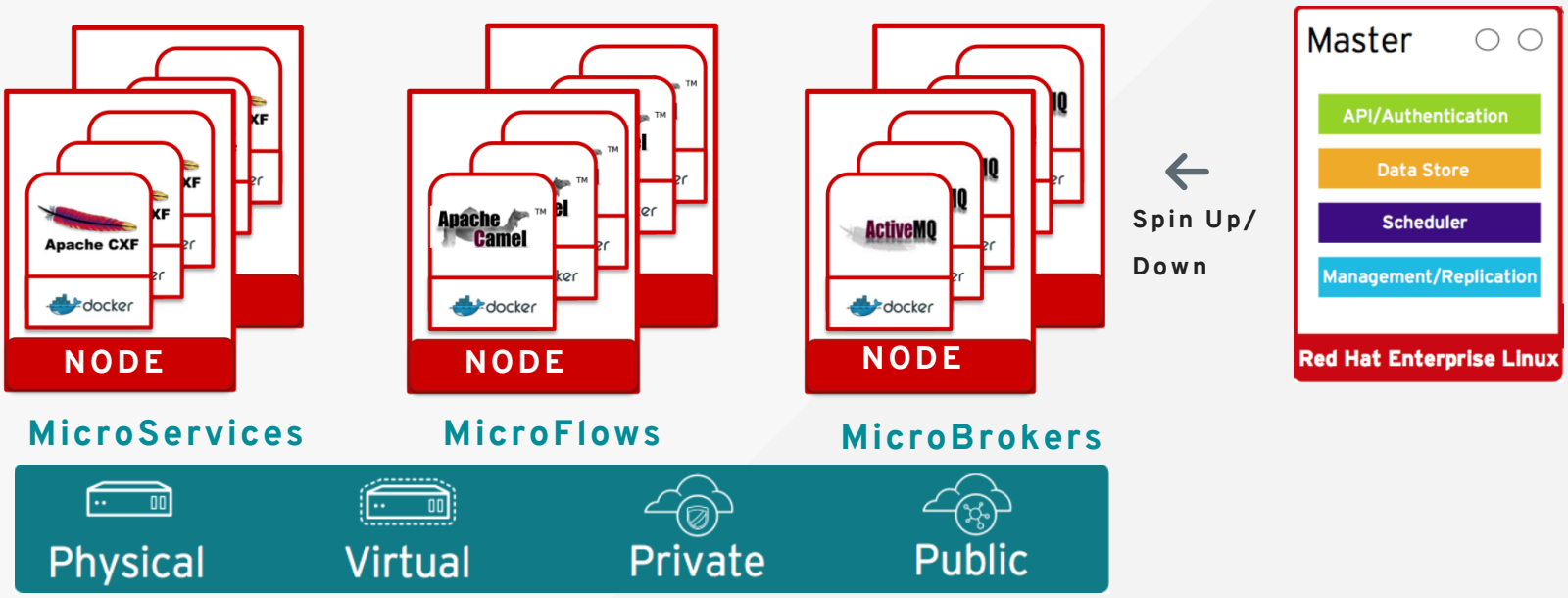
A Broker based Messaging Microservice deployed on demand to drive data between systems

MICROFLOW



An integration flow or route developed To implement Enterprise Integration Patterns as a Microservice

DEPLOYMENT



- SERVICES SPUN/DOWN AS DOCKER PODS USING KUBERNETES
- ON ALL MANNER OF ENVIRONMENTS
- AUTOMATICALLY ON DEMAND OR MANUALLY

DESIGN CONSIDERATIONS

MICROSERVICES *(example: Apache CXF Rest Services)*

- CLONEABLE AND VERSIONABLE
- LOCATION TRANSPARENT, AUTO SCALED AND EASILY RELOCATED

MICROBROKERS *(example: Apache ActiveMQ with Fabric8MQ)*

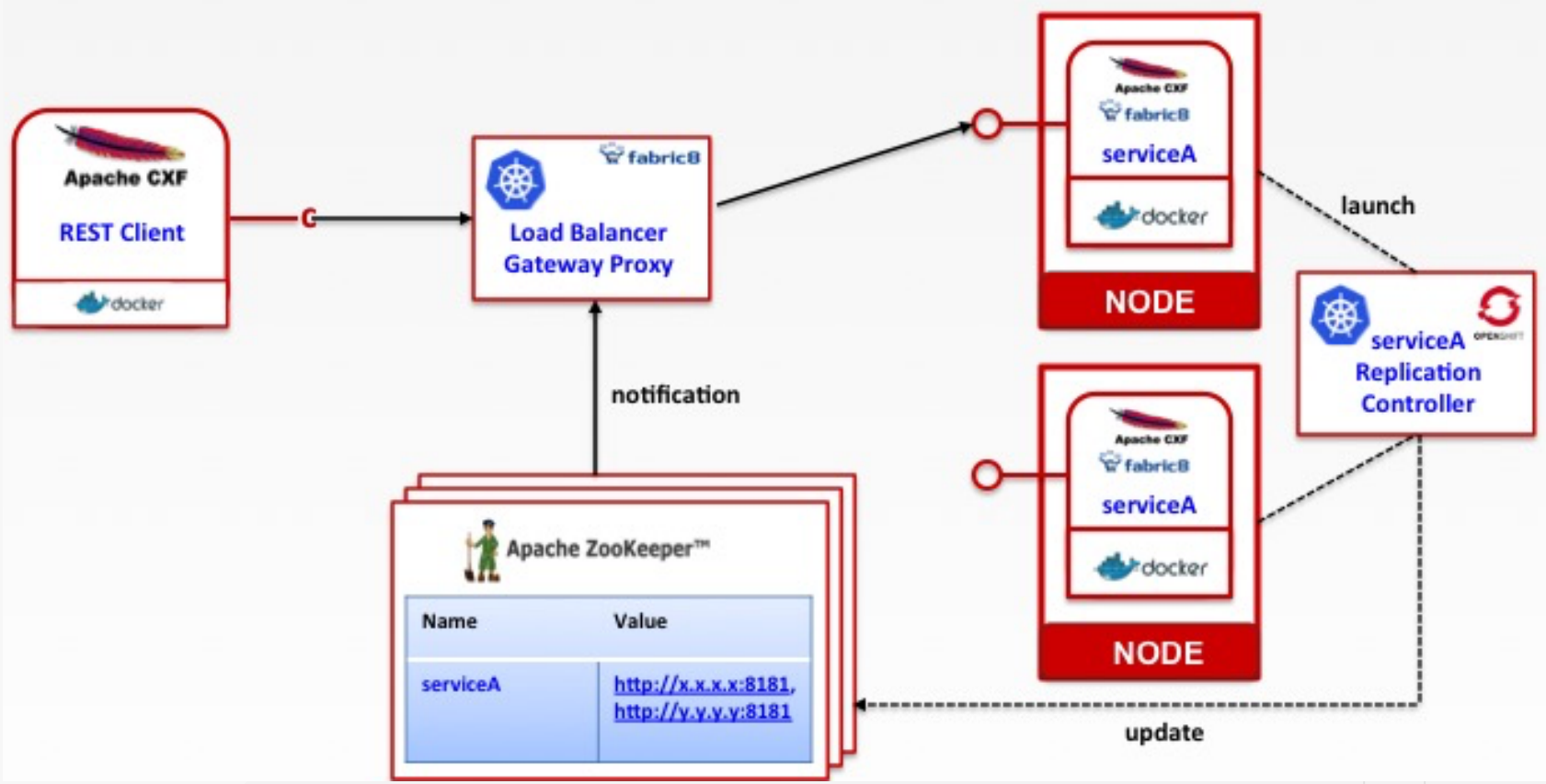
- TOPOLOGY AWARE, SMART AND AUTO SCALING VIA KUBERNETES
- EVENT AND NOTIFICATION AWARE VIA ZOOKEEPER/ETCD WATCHERS
- HIGHLY AVAILABLE, FAILOVER & DISASTER RECOVERY PROOFED

MICROFLOWS *(example: Apache Camel with Fabric8 & OpenShift)*

- TEMPLATE DRIVEN WITH NO BOILER PLATE LOGIC
- TEMPLATE AND ENDPOINTS APPLIED AT RUNTIME USING WATCHERS
- EVENT AND NOTIFICATION AWARE, AUTO SCALED VIA KUBERNETES

DYNAMIC MICROSERVICES

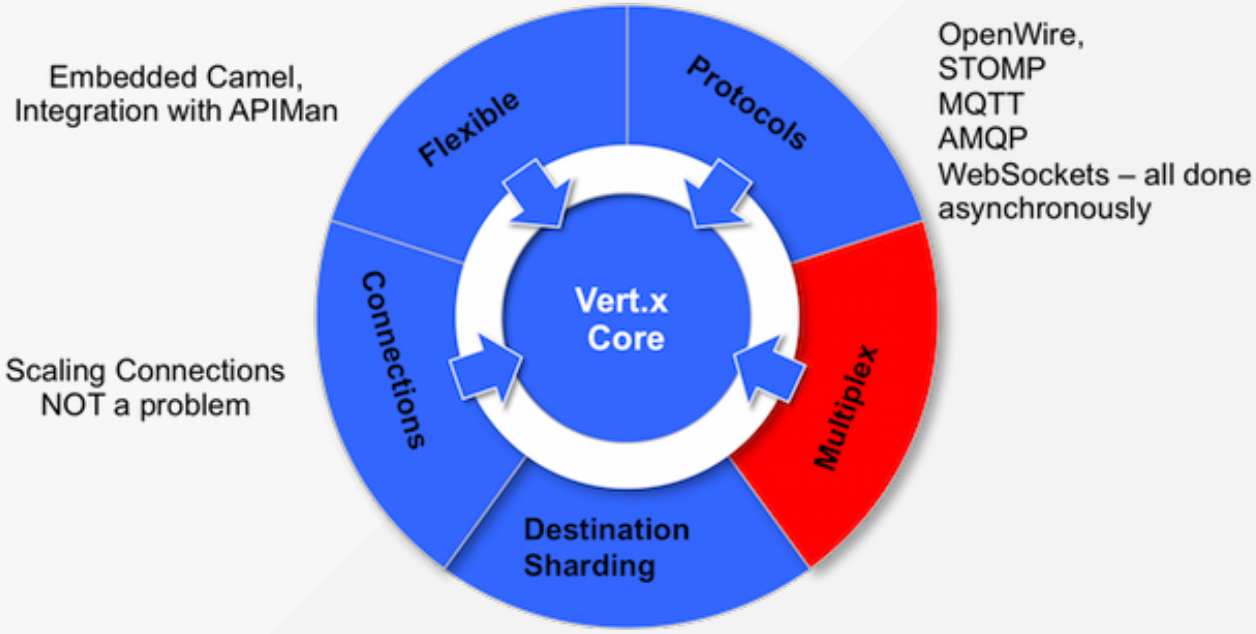
MICROSERVICE AUTO DETECTION AND TRANSPARENT LOAD BALANCING



DYNAMIC MICROBROKERS

SMART, SCALABLE MICROBROKERS OFFERING MESSAGING AS A SERVICE

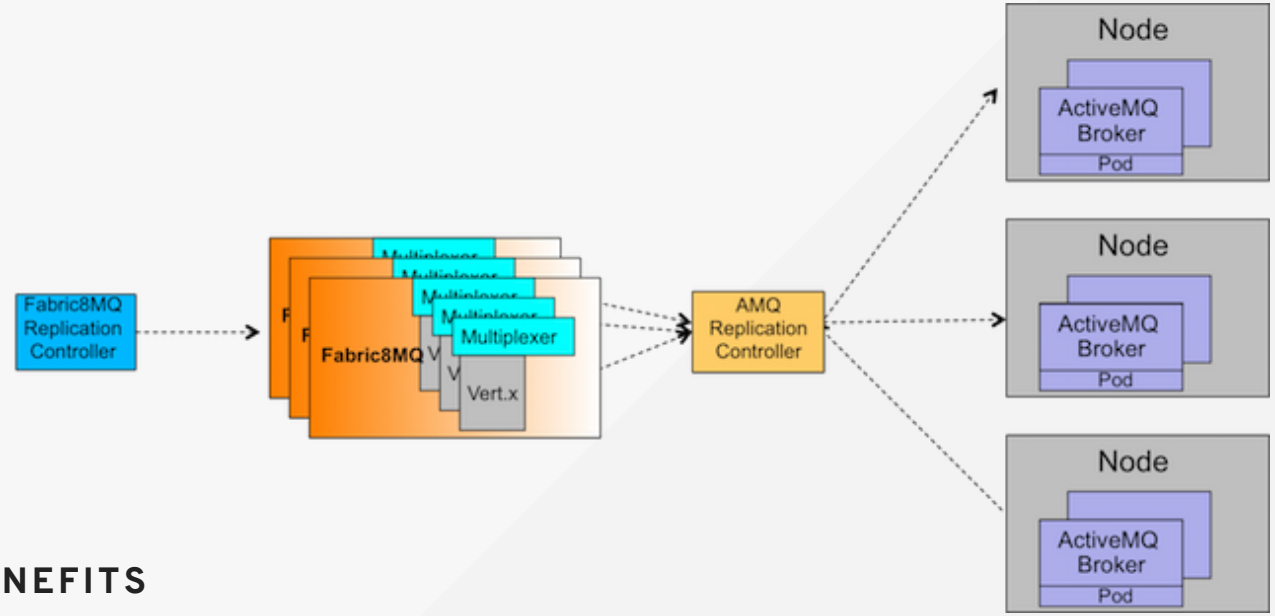
- Messaging as a Service using a smart kubernetes based messaging proxy and auto-sharded ActiveMQ Brokers



- Uses **Vert.x** core to provide highly scalable asynchronous connection handling

DYNAMIC MICROBROKERS

SMART, SCALABLE MICROBROKERS OFFERING MESSAGING AS A SERVICE

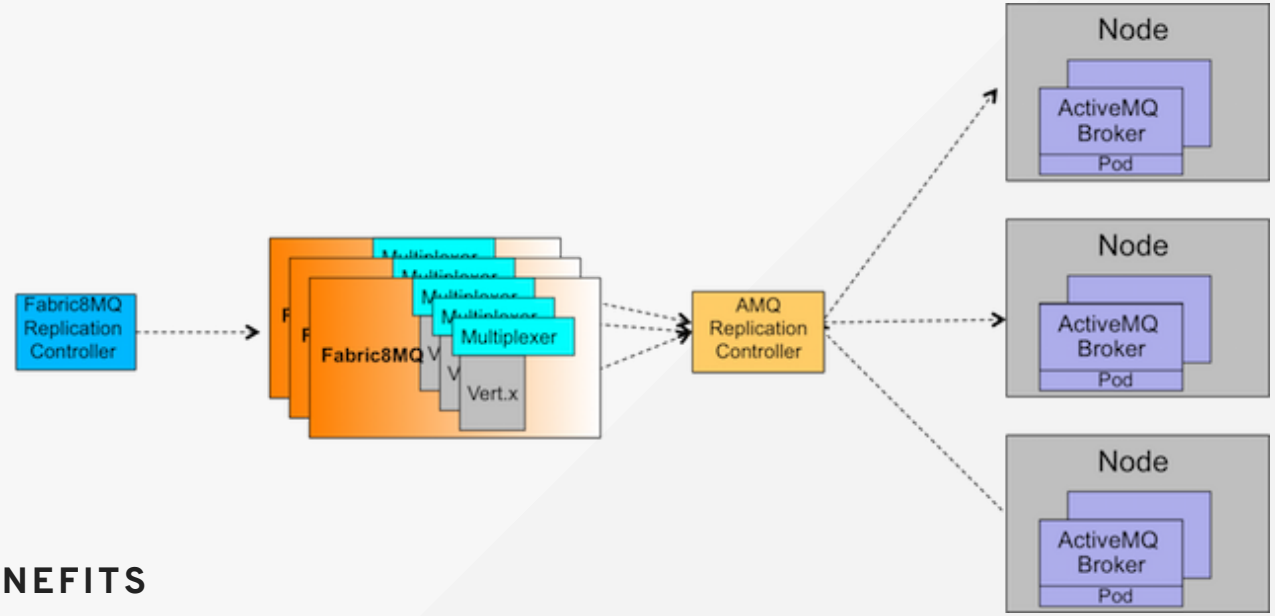


BENEFITS

- **Protocol Conversion:** MQTT, STOMP, AMQP protocol conversion outside of the ActiveMQ broker
- **Co-located Camel Router**: convert on the fly between Topics and Queues
- **API Management:** apply security and rate limiting policies to destinations

DYNAMIC MICROBROKERS

SMART, SCALABLE MICROBROKERS OFFERING MESSAGING AS A SERVICE

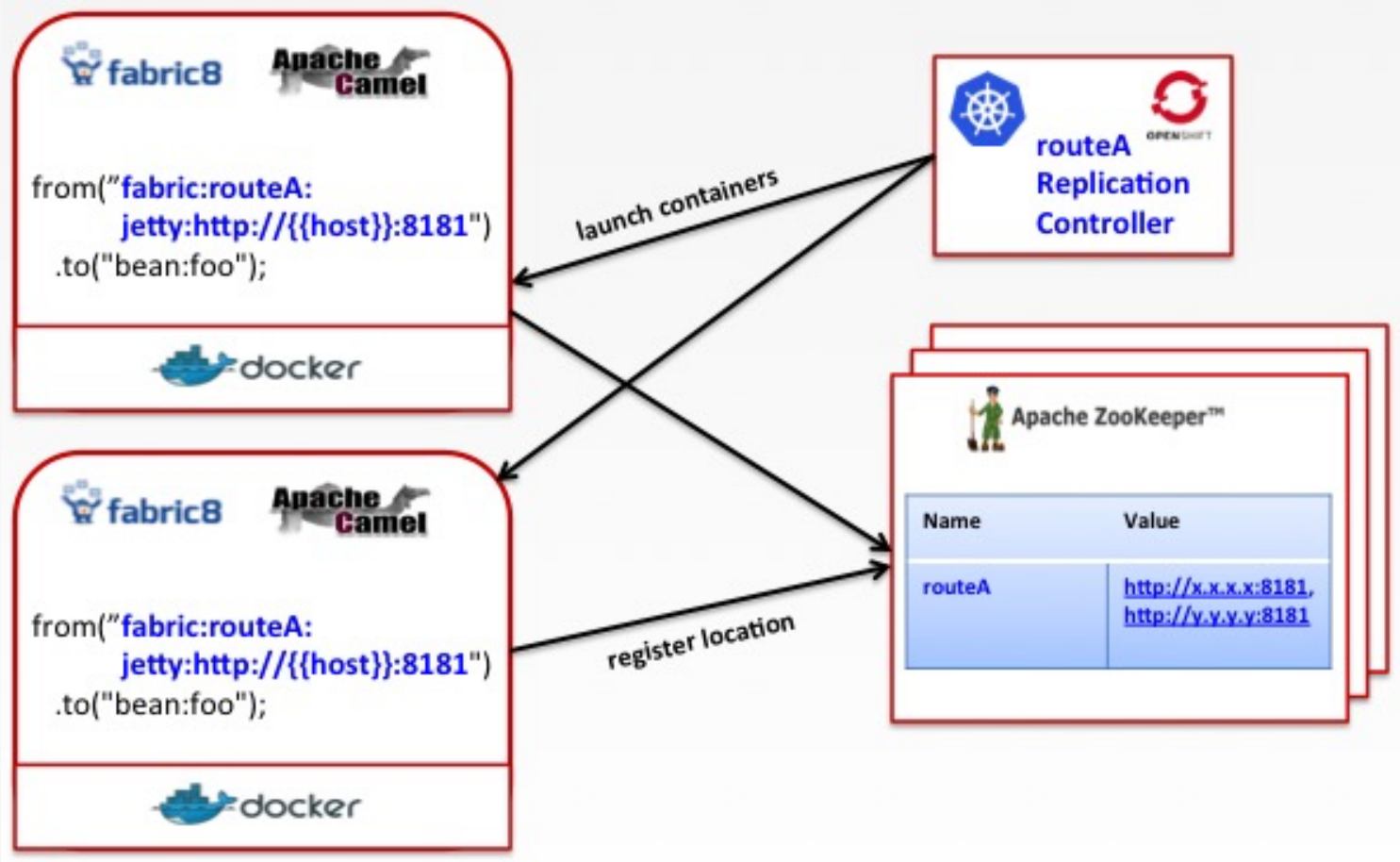


BENEFITS

- **Multiplexer :** lowers the amount of physical connections an ActiveMQ broker and improves throughput
- **Destination Sharding:** Destination based broker sharding, transparent message migration as brokers spin up or down
- **Broker Control:** demand based broker scaling, control and health monitoring

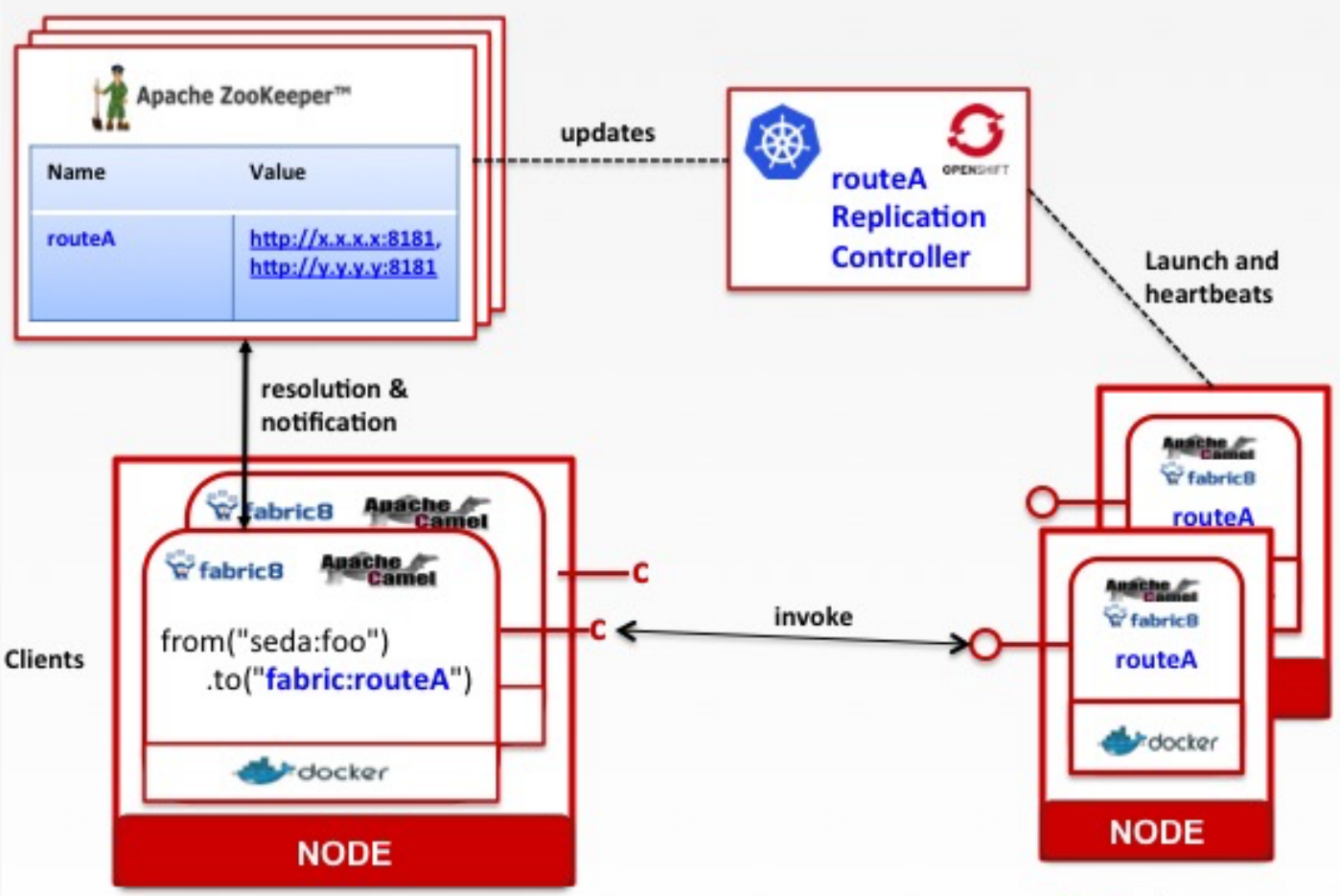
DYNAMIC MICROFLOWS

SCALABLE TEMPLATED MICROFLOWS WITH LOAD BALANCING AND DYNAMIC ENDPOINT INJECTION



DYNAMIC MICROFLOWS

SCALABLE TEMPLATED MICROFLOWS WITH LOAD BALANCING AND DYNAMIC ENDPOINT INJECTION



SUMMARY

- MICROSERVICES ARCHITECTURE CONSISTS OF KEY BUILDING BLOCKS SUCH AS
  - MICROSERVICES
  - MICROBROKERS
  - MICROFLOWS
- MICROSERVICES CAN BE LAUNCHED USING REDHAT OPENSHIFT IN DOCKER PODS AND MANAGED BY KUBERNETES REPLICATION CONTROLLERS
- SMART AND DYNAMIC TOPOLOGIES CAN BE BUILT AROUND FABRIC8 AND SERVICE REGISTRIES SUCH AS APACHE ZOOKEEPER OR ETCD.
- TEMPLATED AND CLONEABLE SERVICES CAN DRIVE STANDARDIZED AND HIGHLY EFFICIENT AND SCALABLE ENVIRONMENTS

redhat.

THANK YOU!