AQUAQ
Analytics

AquaQuarantine:

kdb+ in a containerised solution

Experts in fast data solutions

for demanding environments

AQUAQ
Analytics

- Established in 2011
- Headquarters in Belfast, N.Ireland
- Headcount of 160 staff
- 2016 US Subsidiary launched
- 2018 Singapore subsidiary launch
- 2020 Hong Kong subsidiary launch

NYC
Belfast HQ
London
Tokyo
Hong Kong
Singapore
Sydney

What do we do?

Technology Consultancy Services

Altair Panopticon Professional Services

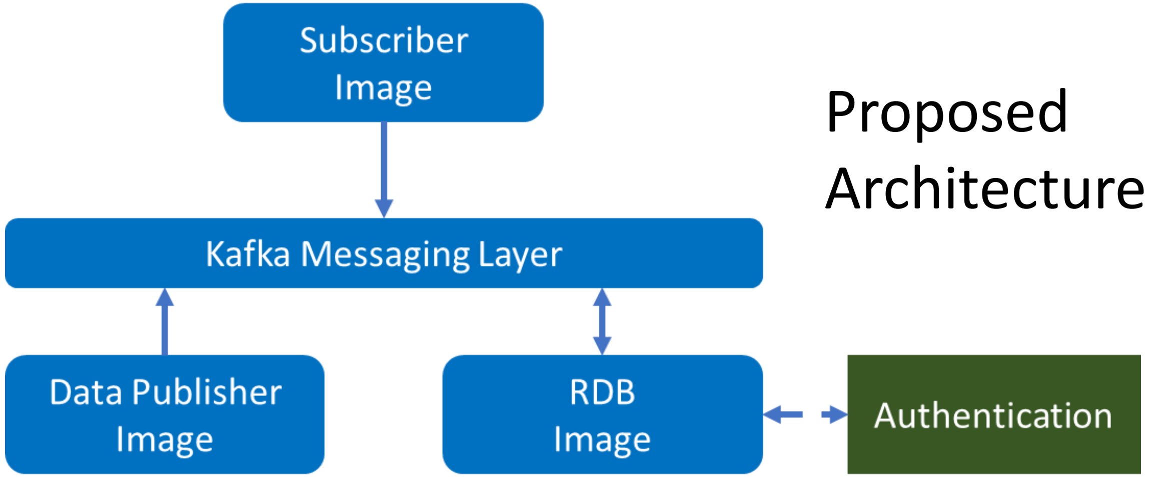
Altair

Remote (24/7) Support Centre of Excellence

Objective

The proof of concept is to demonstrate the following capabilities:

- kdb+ in containers
- Entitlements interaction with kdb+ in containers
- Integration of kdb+ containers within data pub sub setup



Subscriber Image

Kafka Messaging Layer

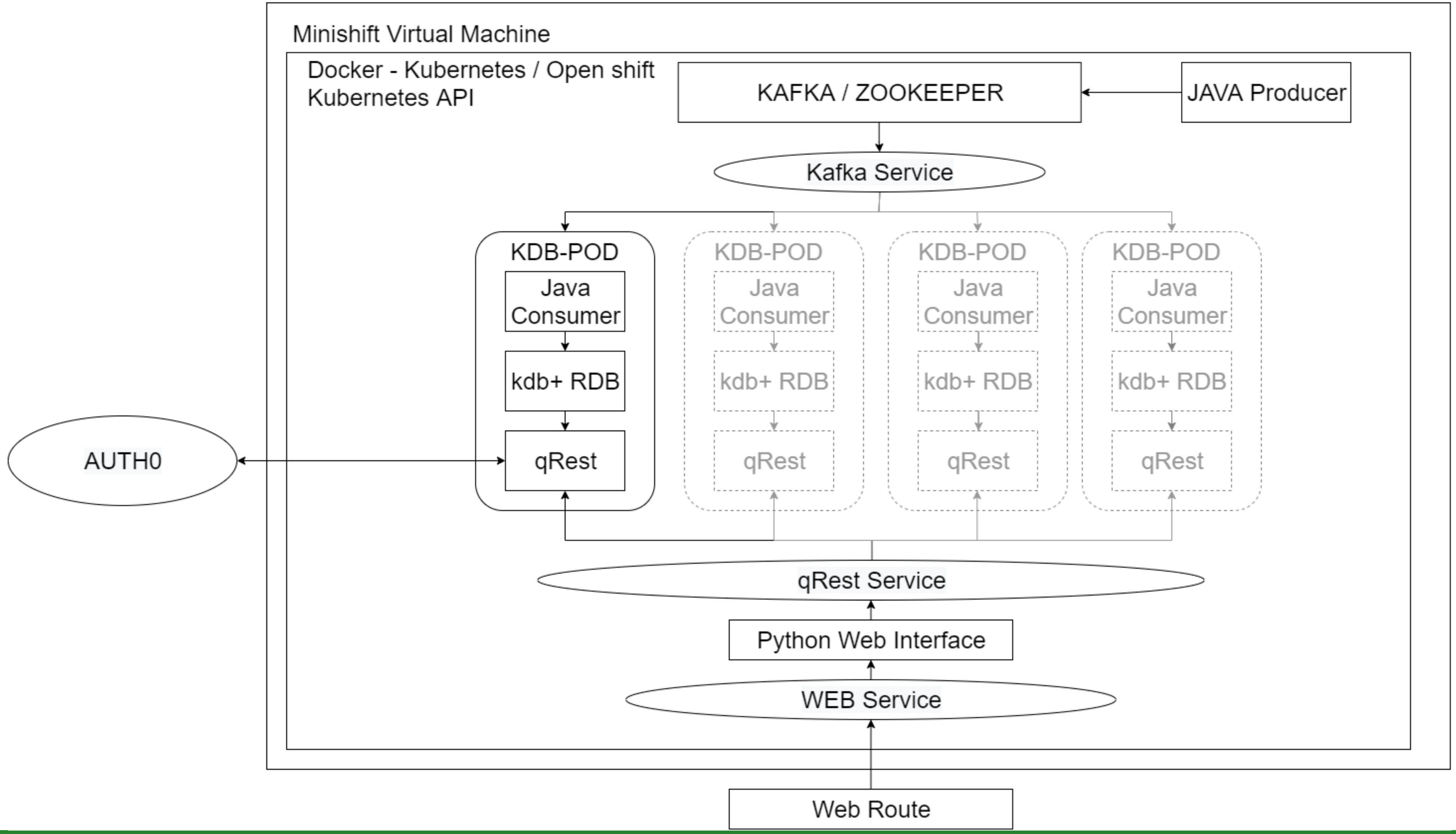
Data Publisher Image

RDB Image

Authentication

Proposed Architecture

Architecture



Technologies

Technology [Version]	Function
Minishift	Virtual Machine running full OKD Cluster
Openshift [3.11] Kubernetes [1.11] Docker	Containers and container management
AUTH0	Authentication and user role management
kdb+	Real-time database
Springboot Java	• Data generation (quote/trade) • Push to Kafka topics (1 per dataset) • Data consumption and push to kdb+ (Feedhandler)
Python JS / HTML /CSS	Web front end
Kafka	Message handling system
qRest	• Open Source Restful Interface for kdb • Modified for token handling

Roles & Entitlements

Roles, entitlements and authentications through AUTH0 (https://auth0.com/)

User	Role	Description
demo0@aquaq.co.uk	realtime	Full access to all data in real time
demo1@aquaq.co.uk	delay_15	Access to data provided it is >=15 minutes old (Time filtering)
demo2@aquaq.co.uk	delay_05, xlon	Access to only London data and aged >=5 minutes (Row and Time filtering)
demo3@aquaq.co.uk	delay_05, xams, xmil	Access to only Amsterdam and Milan data, aged >=5 minutes
demo4@aquaq.co.uk	no_ex	User is not allowed to see the exchange data and therefore it's columns are not visible (Column filtering)
demo5@aquaq.co.uk	no_trade	User cannot see the trade table (Table filtering)

q-REST

kdb+ queries are through the q-REST API (https://www.aquaq.co.uk/q/q-rest/). This has been modified to handle tokens.

- Simple RESTful web services
- Integration from any client
- Connects to any kdb+ database
- Seamless integration with TorQ
- Supports synchronous and deferred sync connections for concurrency

https://github.com/AquaQAnalytics/q-REST

Demo

AQUAQ Analytics

kdb-containers PoC

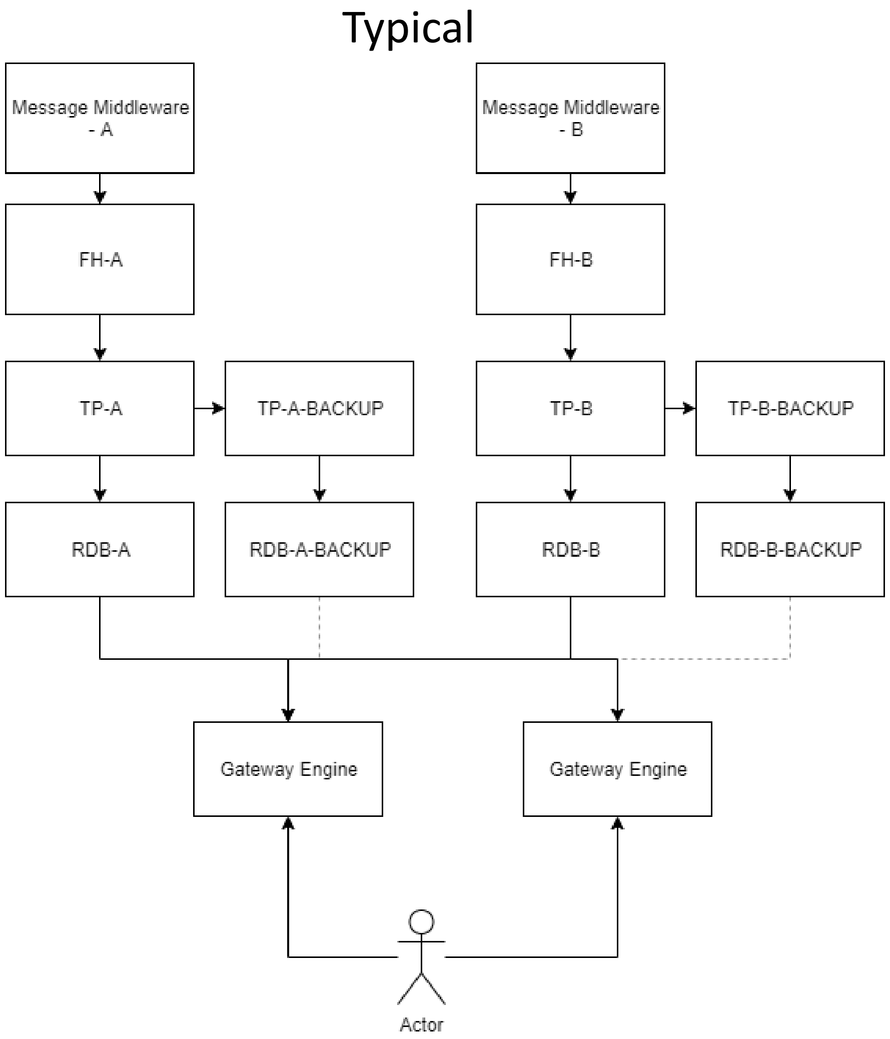
Success Criteria

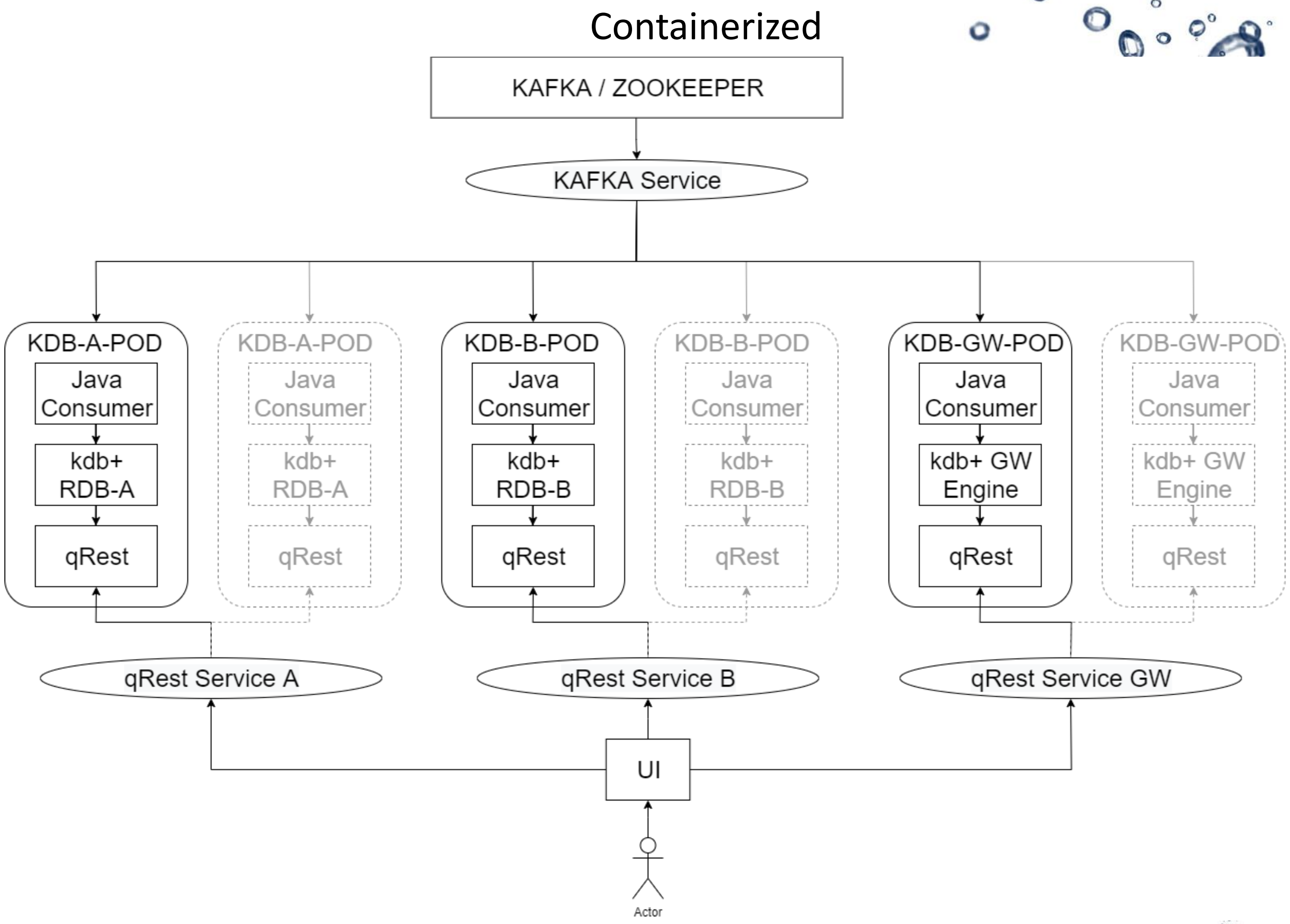
Scenario	Comments
Simulated market data will be generated in the publisher	✓ PoC uses Java/Springboot container
Publisher will publish data onto a Kafka Topic	✓ Kafka topic per dataset trade/quote ✓ Publisher sends to appropriate topic
RDB will consume data from the Kafka Topic	✓ RDB consumes via Java Feedhandler (FH) ✓ RDB and FH are housed in replicable pod ✓ Pods allows for horizontal scalability
Subscriber will simulate connection from querying client	✓ Python web front end with q-Rest integration ✓ Querying client read-only
Subscriber will simulate connection from a client subscription	✓ UI interface allows for polling based subscription ✓ Client applications should consume from kafka
Subscriber subject to authentication	✓ Role based access via AUTH0 ✓ Single entry point, no kdb+ ports exposed
Entitlements demonstration against kdb+ service	✓ Users have roles determining data access
Chaos scenario, overloaded RDB	✓ RDB Crashes, comes back up as went inactive

Containerised vs Non

Characteristic	Containerised	Non-Containerised
Architecture	• Potential to simplify through no tickerplants, kafka direct replay • Consumption from Kafka, rather than kdb+ -> kdb+ • Current kdb implementations would need re-defined	• Typical kdb tick architecture
Resources	• Reduced CPU consumption due to tickerplant removal • Reduced disk usage due to no tickerplant log • Memory footprint of RDB the same	
Scalability	• qRest provides a single entry point to access all available pods • Pods can be scaled for additional RDBs • Pod monitoring can ensure n number of pods are active	• Chained tickerplants and RDBs would be required, these require manual interaction and configuration with additional monitoring added
Recoverability	• Pods can be spawned easily and automatically without need for user switching • Recovery time will still be dependent on amount of messages to consume	• Requires manual intervention and user notification/impact without additional architecture (Gateways) • Recovery time dependent on amount of messages to consume

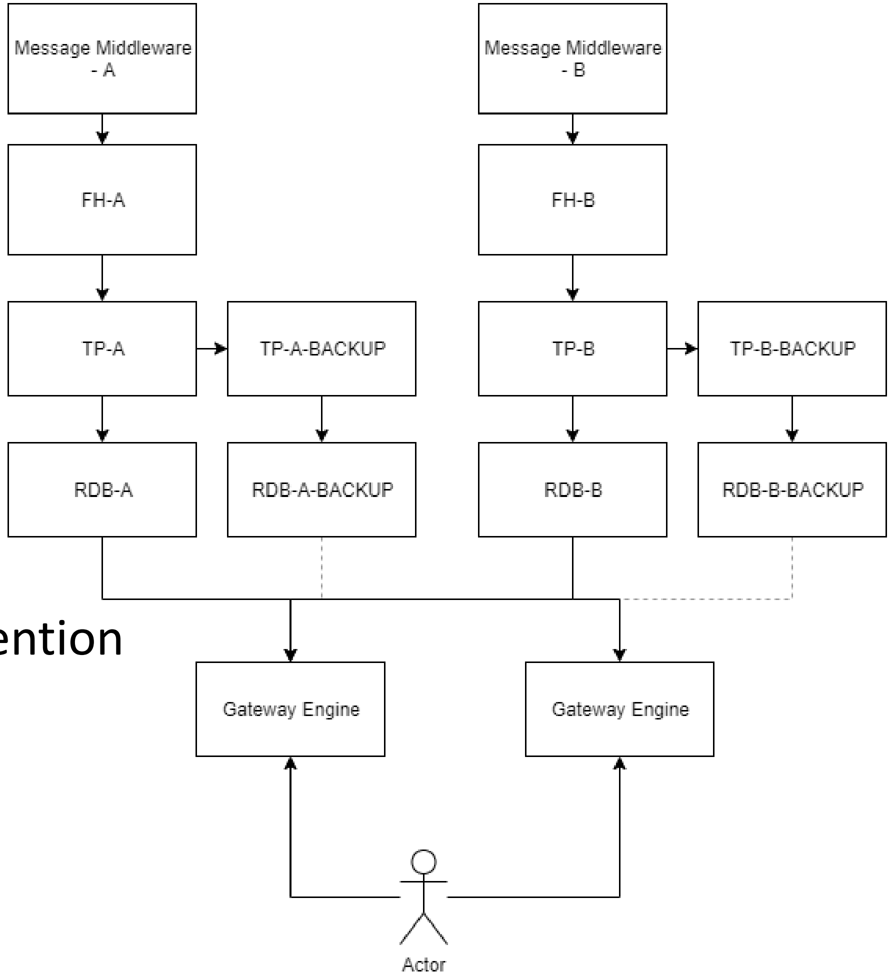
A Potential Re-Architecture





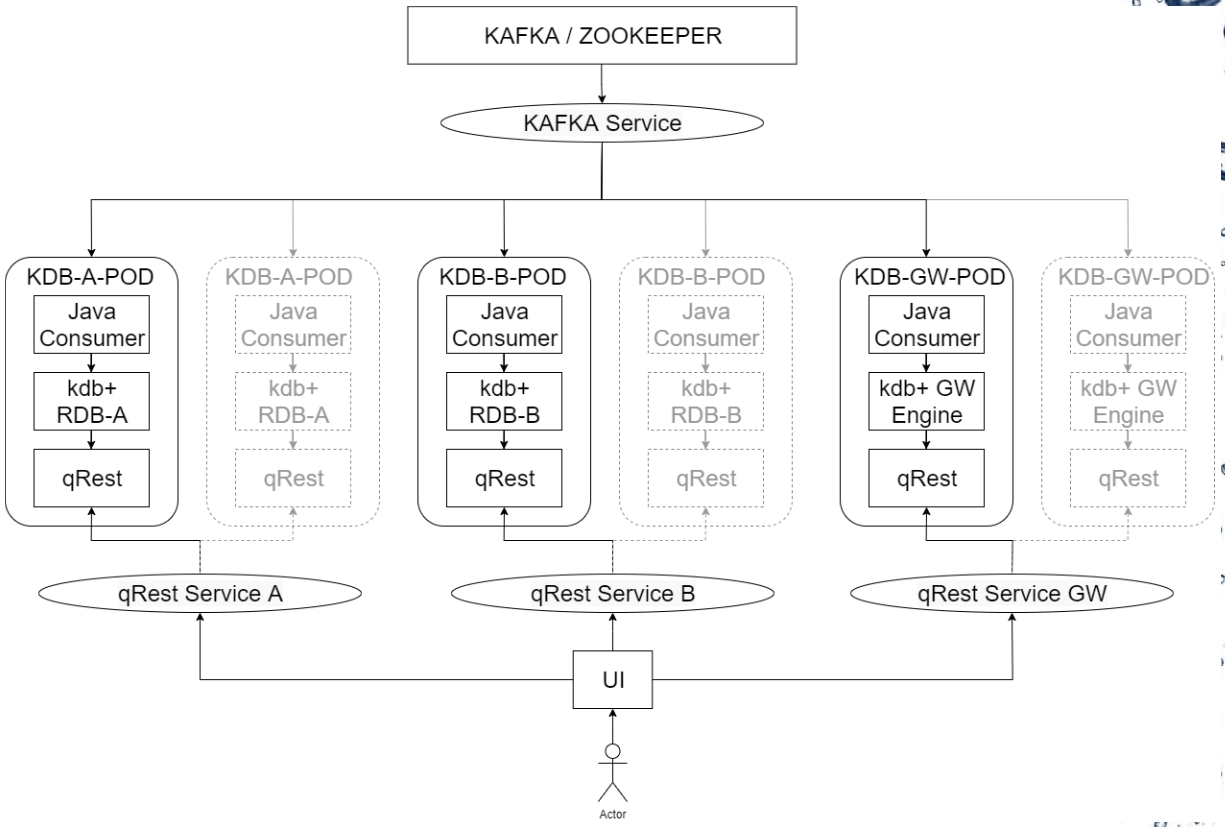
Potential Re-Architecture:Typical

- Individually managed and supported components
- Open access to all elements
- No scalability without config/code implementation
- Chained Tickerplants connected to main chain
- Susceptible to change in server configuration
- Migration across hardware is difficult
- Many potential points of failure (TP/RDB/GW/FH)
- Loss of TP results in full chain failure
- Loss of RDB impacts GW and requires manual intervention
- Access to RDBs could impact users on GW
- Manual restart of elements in any failure
- kdb+ upgrade requires interaction with all elements, dependencies and clients of each element.



Potential Re-Architecture:Typical

- Pods managed through Openshift
- Single entry point to each kdb+ through qRest endpoints.
- No open access to kdb+ processes
- No chained tickerplants, each pod is independent
- Instant scalability
- Migration is simplified (no server specific config)
- Microservice architecture (auto recovery of pods)
- No Tickerplant, recovery from Kafka
- No Tickerplant, no risk of slow consumer
- GW is separate, if RDB goes down no impact.
- Potential increased memory/data duplication by separating GW



Potential Re-Architecture:Typical

- kdb+ in containers
  - Scalable
  - Recoverable
  - Server Agnostic
  - Secure

- Entitlements interaction with kdb+ in containers
  - Single entry point
  - Unit of deployment/access is now pod not individual kdb+ process

- Integration of kdb+ containers within data pub sub setup
  - Integration with kafka
  - Potential removal of tickerplant
  - Integration with Restful interface

Thanks!

Q+A

Upcoming Talks:

- Memverge Memory Machine – 4th June