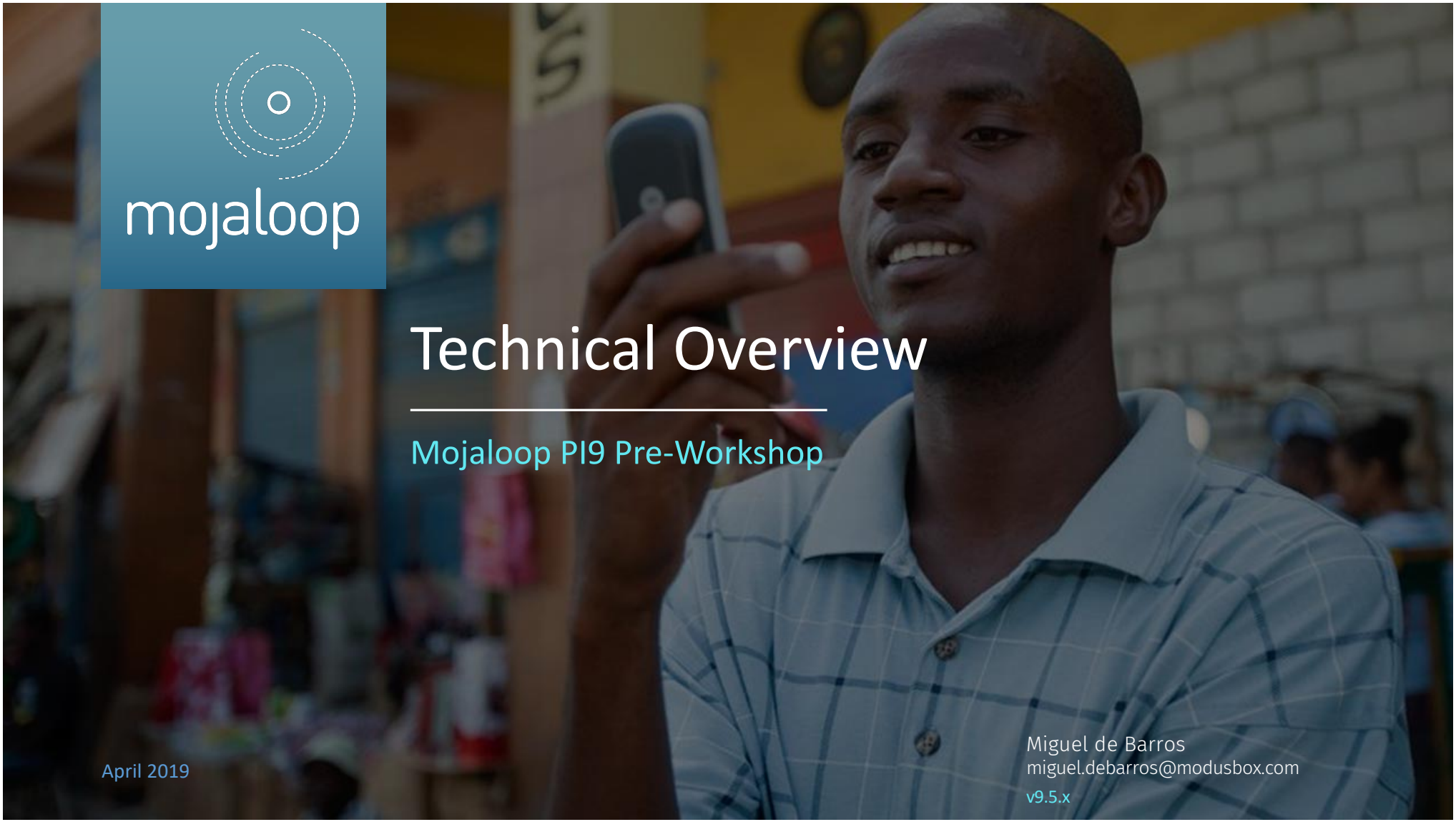


Technical Overview

Mojaloop PI9 Pre-Workshop

April 2019

Miguel de Barros
miguel.debarros@modusbox.com

v9.5.x

mojaloop

Mojaloop Links

Website: http://mojaloop.io

Docs: http://mojaloop.io/documentation

Code: http://github.com/mojaloop

Slack: https://mojaloop-slack.herokuapp.com (self-invite)

Gitter: https://gitter.im/mojaloop/mojaloop

Mojaloop

Technical Overview

1. What is Mojaloop?
2. High-level Overview, Static Demo & Architecture
3. Component Architecture
4. Global foot-print & Roll-out

Mojaloop

Technical Overview

1. What is Mojaloop?
2. High-level Overview, Static Demo & Architecture
3. Component Architecture
4. Global foot-print & Roll-out

What is Mojaloop?

- Open Loop System
- Real-time, Irrevocable, Push-only
- DFSP Governed, Same-day Settlements
- Shared investment in Fraud detection

April 2019

Mojaloop

Technical Overview

1. What is Mojaloop?
2. High-level Overview, Static Demo & Architecture
3. Component Architecture
4. Global foot-print & Roll-out

OSS Functional Overview

Key
[●] Fully implemented
[●] Supported, not tested by OSS
[●] Partially implemented
[●] Not implemented
[○] Future Roadmap

Mojaloop v1.0 Use-cases

Payer-Initiated Transaction
[●] P2P Transfers – Golden Path
  [●] Prepares, Fulfils
  [●] Rejections, Timeouts
  [●] Error Endpoints
[●] Customer-Initiated Merchant Payment
[●] Customer-Initiated Cash-out - Receive Amount
[●] Customer-Initiated Cash-out - Send Amount
[●] ATM-Initiated Cash-out
[●] Refund

Bulk Transactions
[●] Bulk Payments

Payee-Initiated Transaction
[●] Merchant-Initiated Merchant Payment
[●] Agent-Initiated Cash-out
[●] Agent-Initiated Cash-in – Send Amount
[●] Agent-Initiated Cash-in – Receive Amount

Payee-Initiated Transaction using OTP
[●] Merchant-Initiated Merchant Payment Authorized on POS
[●] Agent-Initiated Cash-out Authorized on POS

Additional Supported Use-cases

Payer-Initiated Transaction
[●] P2P Transfers – On-US

Settlements
[●] Funds-in
[●] Funds-out
[●] Open-close Settlement-windows
[●] Trigger Settlement Processing
[●] Receive & Process Settlement Acknowledgements
[●] Reconcile FSP Positions

April 2019

Definition of general terms

1. Hub
- The general term for an operating Mojaloop Switch in the “wild”
- This includes Mojaloop and all supporting services/components/rules required to operate a fully functional payment switch
- Adheres to regulatory requirements when/where applicable

2. Scheme
- The rules around an operating Mojaloop Hub
  - Limits
  - Reporting
  - Regulations
  - Onboarding process
  - etc

3. (D)FSP
- (Digital) Financial Service Provider

4. Participants
- General term for an FSP that utilizes Mojaloop to exact payments

5. Party
- The customer that belongs to an FSP(s) which is represented by an identifier (e.g. MSISDN)

Definition of general terms

6. Cryptographic Receipt (https://interledger.org/)

- Cryptographic receipt used to validate payments which compromised of two parts:
  - Fulfilment ← cryptographically generated from payment information by the Payee FSP
  - Condition ← hash of the Fulfilment (used to validate Transfers when Fulfilment is shared)

7. Participant Position

- A participant (FSP) position is a net position of all activity
- Applies to a switch account for a scheme participant in a specific currency

8. Net Debit Cap Limit

- The limit applied to a participant's position, i.e. a participant is not allowed to trade in the system if their position exceeds their Net Debit Cap Limit.

Stages of a Mojaloop P2P Transaction

1. Discovery
    a. Resolve link between a Party (Customer) and a Participant (FSP)
    b. Lookup Party details (name, account, etc)

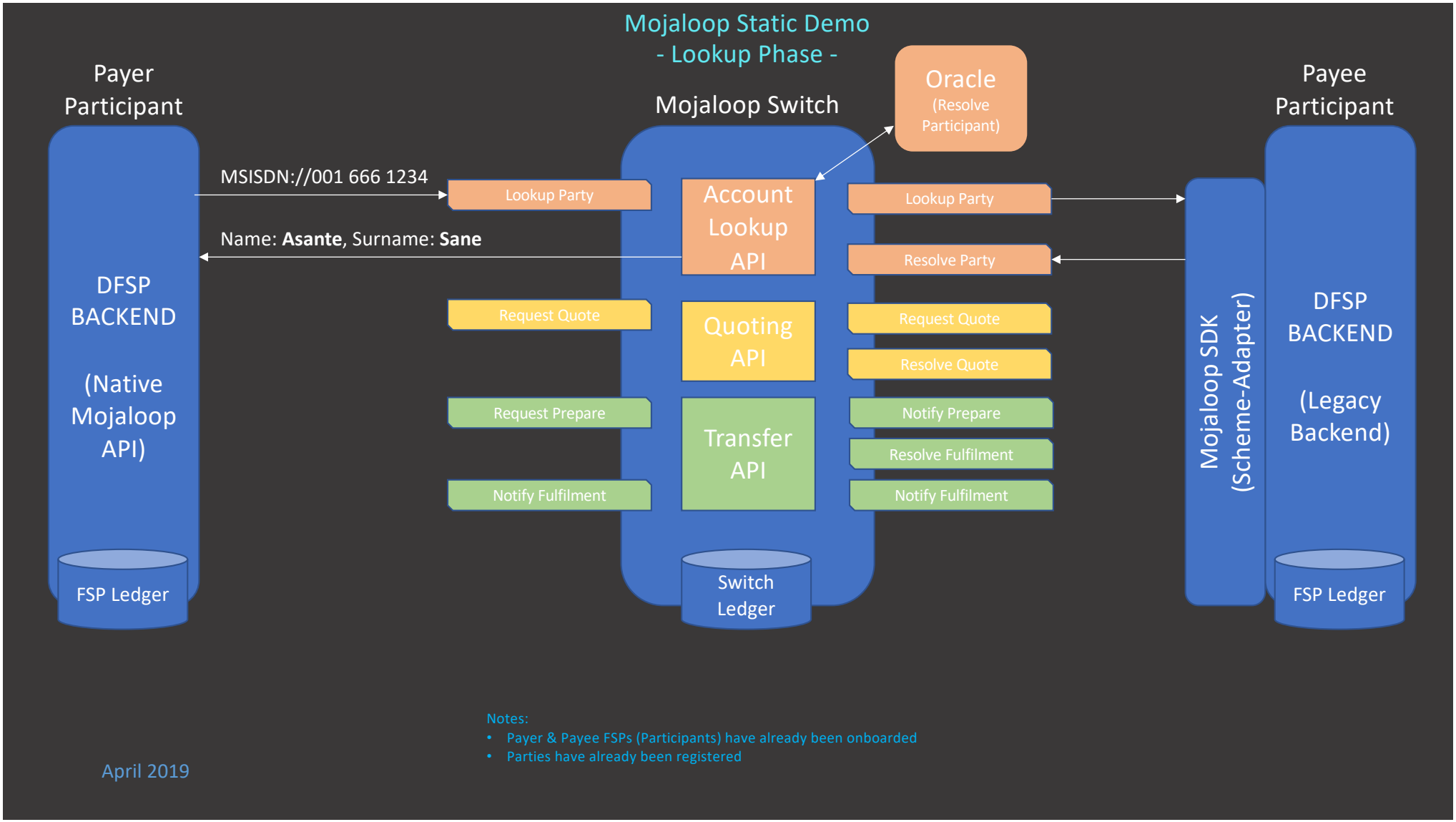
2. Agreement
    a. Participants
    b. Transfer Amount
    c. Fees
    d. Generate Cryptographic Receipt (used for validation on Transfer phase)
        a. Fulfilment → crypto receipt generated from payment information
        b. Condition → hash of the Fulfilment

3. Transfer
    a. Request (Prepare)
        i. Sub-set of Quote information (e.g. participants, amount, etc)
        ii. Condition
    b. Response (Fulfil)
        i. Commit / Abort
        ii. Fulfilment (Crypt-Receipt)

Static Demo: P2P Scenario

- A customer (Sending Party) from an FSP (Payer Participant) wants to make a payment to a friend (*Asante Sane* – Recipient Party) that belongs to another FSP (Payee Participant) ← Payer Participant can exact this by a ***Transfer Request***
- The Payer Participant does not know which FSP *Asante Sane* is a customer of. ← Payer Participant can resolve this first via an ***Account Lookup***
- Payer Participants wants to agree/validate the final transfer (inc. costs, fees, etc) amount with its paying customer (Sending Party) ← Payer Participant can resolve this via a *Quote Request*

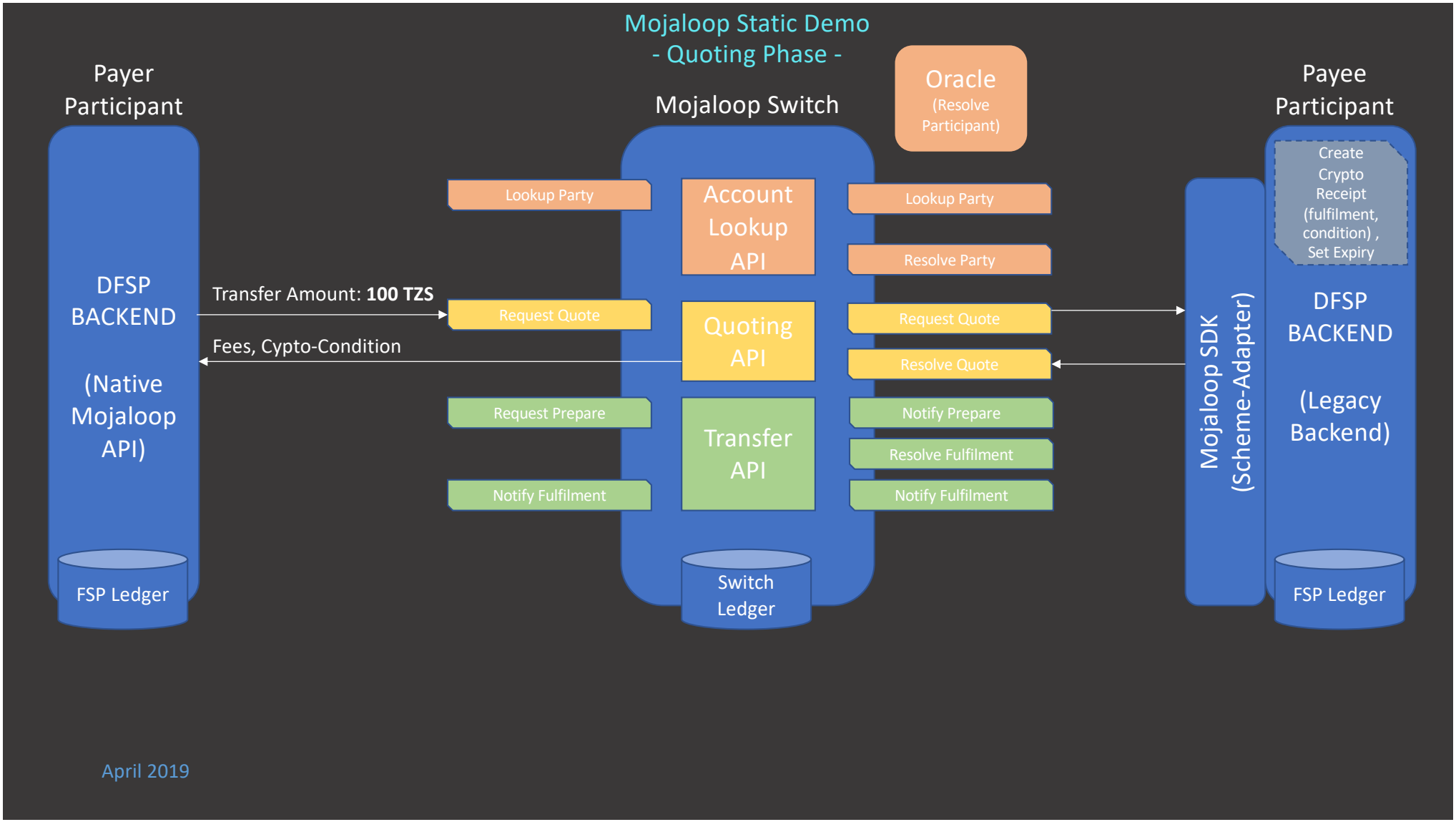
Assumptions:
1. Payer and Payee FSPs are already members of the scheme
2. Payer and Payee FSPs have provided up-front collateral (and Position Limits have been set appropriately)


Mojaloop Static Demo
- Lookup Phase -
Payer Participant
Mojaloop Switch
Oracle (Resolve Participant)
Payee Participant
MSISDN://001 666 1234
Lookup Party
Account Lookup API
Lookup Party
Name: **Asante**, Surname: **Sane**
Resolve Party
DFSP BACKEND
(Native Mojaloop API)
Request Quote
Quoting API
Request Quote
Resolve Quote
Request Prepare
Transfer API
Notify Prepare
Resolve Fulfilment
Notify Fulfilment
Notify Fulfilment
Mojaloop SDK (Scheme-Adapter)
DFSP BACKEND
(Legacy Backend)
FSP Ledger
Switch Ledger
FSP Ledger
Notes:
• Payer & Payee FSPs (Participants) have already been onboarded
• Parties have already been registered
April 2019

Static Demo: P2P Scenario

- A customer (Sending Party) from an FSP (Payer Participant) wants to make a payment to a friend (*Asante Sane* – Recipient Party) that belongs to another FSP (Payee Participant) ← Payer Participant can exact this by a ***Transfer Request***
- The Payer Participant does not know which FSP *Asante Sane* is a customer of. ← Payer Participant can resolve this first via an ***Account Lookup***
- Payer Participants wants to agree/validate the final transfer (inc. costs, fees, etc) amount with its paying customer (Sending Party) ← Payer Participant can resolve this via a *Quote Request*

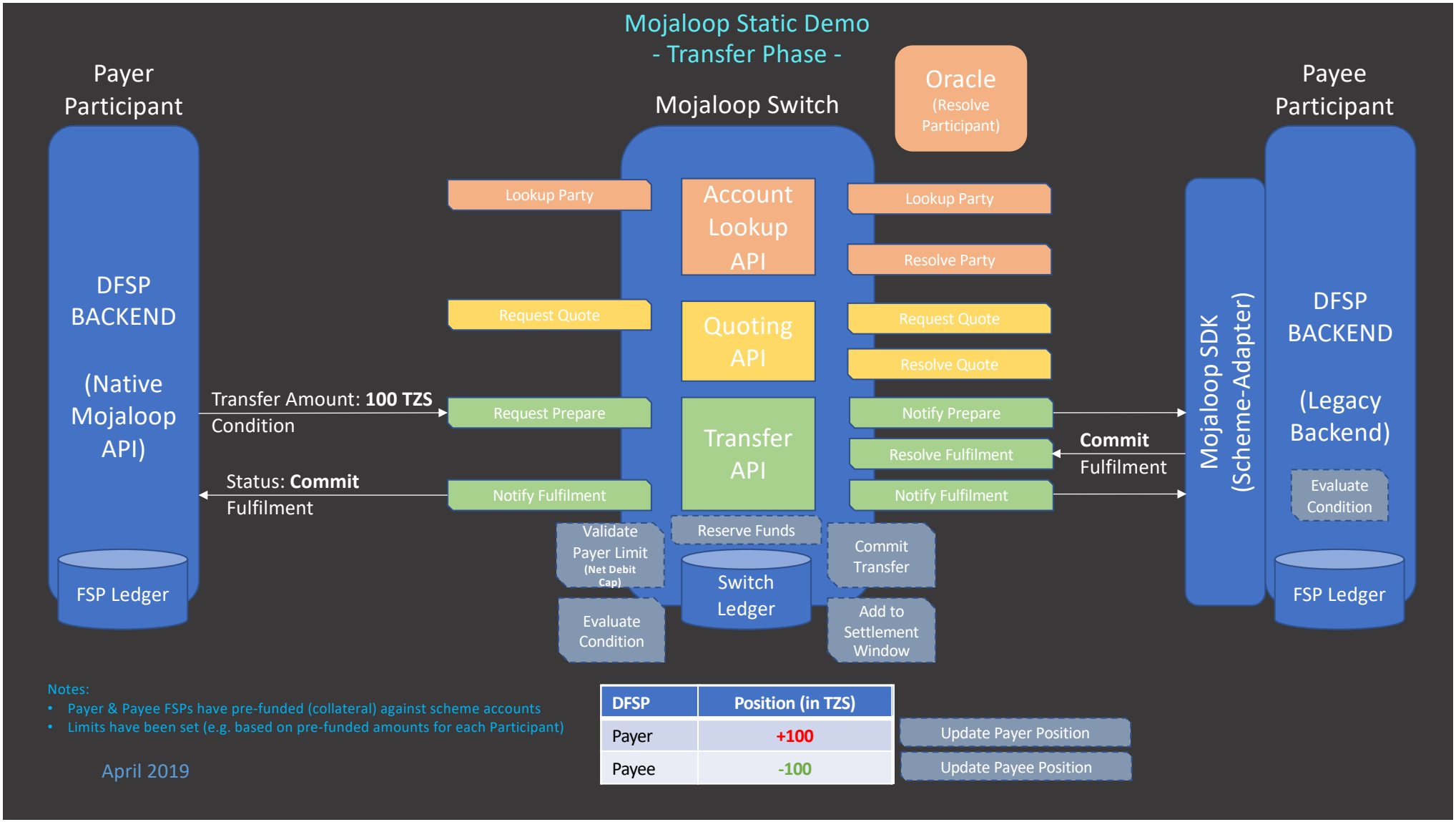
Assumptions:
1. Payer and Payee FSPs are already members of the scheme
2. Payer and Payee FSPs have provided up-front collateral (and Position Limits have been set appropriately)


Mojaloop Static Demo
- Quoting Phase -
Payer Participant
Mojaloop Switch
Oracle (Resolve Participant)
Payee Participant
DFSP BACKEND
(Native Mojaloop API)
Transfer Amount: 100 TZS
Fees, Cypto-Condition
Lookup Party
Request Quote
Request Prepare
Notify Fulfilment
Account Lookup API
Quoting API
Transfer API
Lookup Party
Resolve Party
Request Quote
Resolve Quote
Notify Prepare
Resolve Fulfilment
Notify Fulfilment
Mojaloop SDK (Scheme-Adapter)
Create Crypto Receipt (fulfilment, condition) , Set Expiry
DFSP BACKEND
(Legacy Backend)
FSP Ledger
Switch Ledger
FSP Ledger
April 2019

Static Demo: P2P Scenario

- A customer (Sending Party) from an FSP (Payer Participant) wants to make a payment to a friend (*Asante Sane* – Recipient Party) that belongs to another FSP (Payee Participant) ← Payer Participant can exact this by a ***Transfer Request***
- ~~The Payer Participant does not know which FSP *Asante Sane* is a customer of. ← Payer Participant can resolve this first via an ***Account Lookup***~~
- Payer Participants wants to agree/validate the final transfer (inc. costs, fees, etc) amount with its paying customer (Sending Party) ← Payer Participant can resolve this via a *Quote Request*

Assumptions:
1. Payer and Payee FSPs are already members of the scheme
2. Payer and Payee FSPs have provided up-front collateral (and Position Limits have been set appropriately)


Mojaloop Static Demo
- Transfer Phase -
Payer Participant
Mojaloop Switch
Oracle
(Resolve Participant)
Payee Participant
DFSP BACKEND
(Native Mojaloop API)
FSP Ledger
Lookup Party
Account Lookup API
Lookup Party
Resolve Party
Request Quote
Quoting API
Request Quote
Resolve Quote
Transfer Amount: 100 TZS
Condition
Request Prepare
Transfer API
Notify Prepare
Commit
Fulfilment
Resolve Fulfilment
Status: Commit
Fulfilment
Notify Fulfilment
Notify Fulfilment
Validate Payer Limit (Net Debit Cap)
Reserve Funds
Commit Transfer
Switch Ledger
Evaluate Condition
Add to Settlement Window
Mojaloop SDK (Scheme-Adapter)
DFSP BACKEND
(Legacy Backend)
Evaluate Condition
FSP Ledger
Notes:
• Payer & Payee FSPs have pre-funded (collateral) against scheme accounts
• Limits have been set (e.g. based on pre-funded amounts for each Participant)
April 2019
DFSP
Position (in TZS)
Payer
+100
Payee
-100
Update Payer Position
Update Payee Position

Static Demo: P2P Scenario

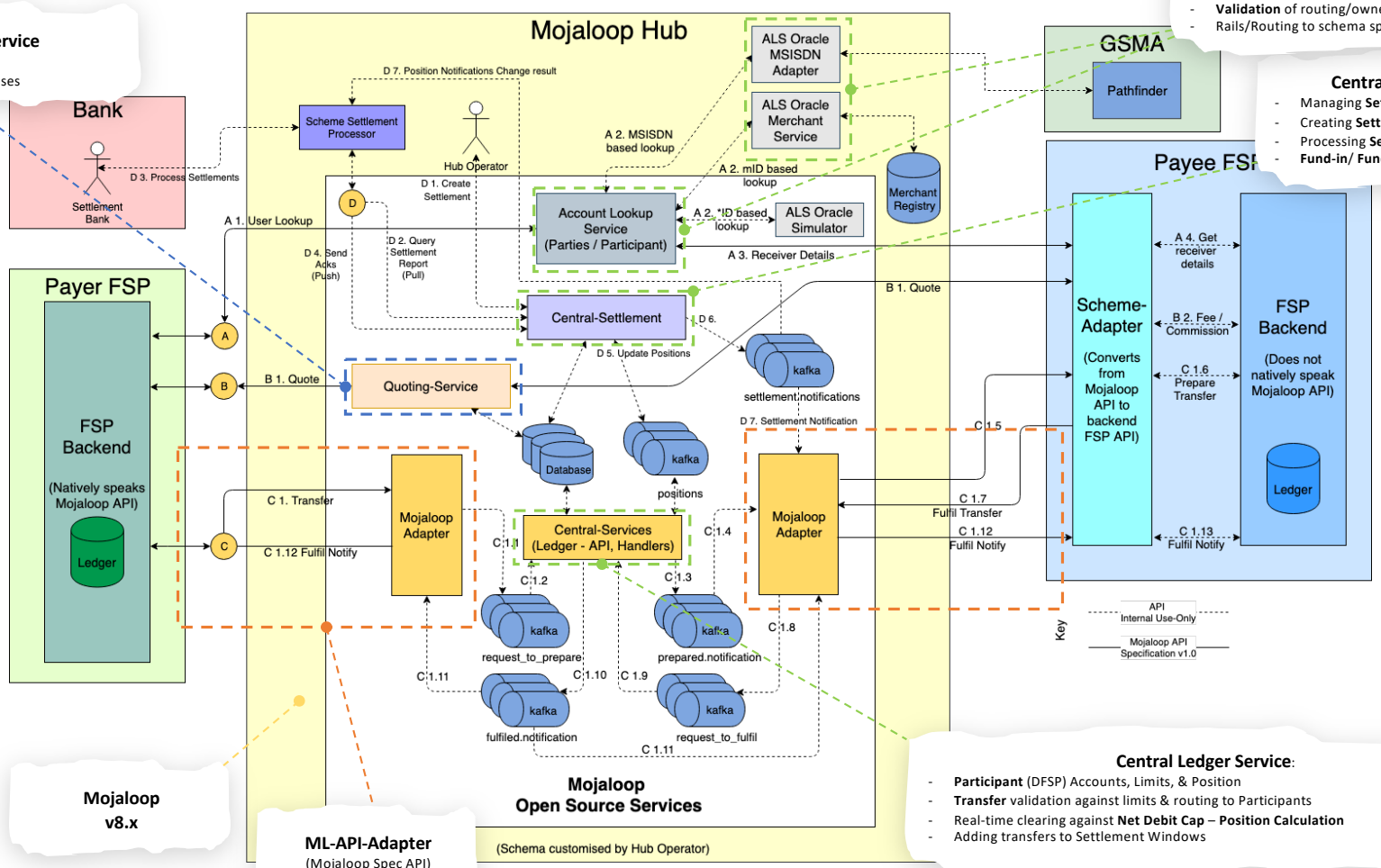
- A customer (Sending Party) from an FSP (Payer Participant) wants to make a payment to a friend (*Asante Sane* – Recipient Party) that belongs to another FSP (Payee Participant) ← Payer Participant can exact this by a ***Transfer Request*** ✓
- ~~The Payer Participant does not know which FSP *Asante Sane* is a customer of. ← Payer Participant can resolve this first via an **Account Lookup**~~
- ~~Payer Participants wants to agree/validate the final transfer (inc. costs, fees, etc) amount with its paying customer (Sending Party) ← Payer Participant can resolve this via a *Quote Request*~~

High-level Architecture

Quoting Service
- Initiate Quote requests
- Resolve Quoting responses

Account Lookup Service:
- Resolve **Participant** (DFSP) Routing
- Resolve **Party** (Customer) details via DFSP
- **Validation** of routing/ownership
- Rails/Routing to schema specific **Oracle registries**

Central-Settlement Service:
- Managing **Settlement Windows**
- Creating **Settlement reports**
- Processing **Settlement Acks**
- **Fund-in/ Funds-out**



Mojaloop v8.x

ML-API-Adapter
(Mojaloop Spec API)

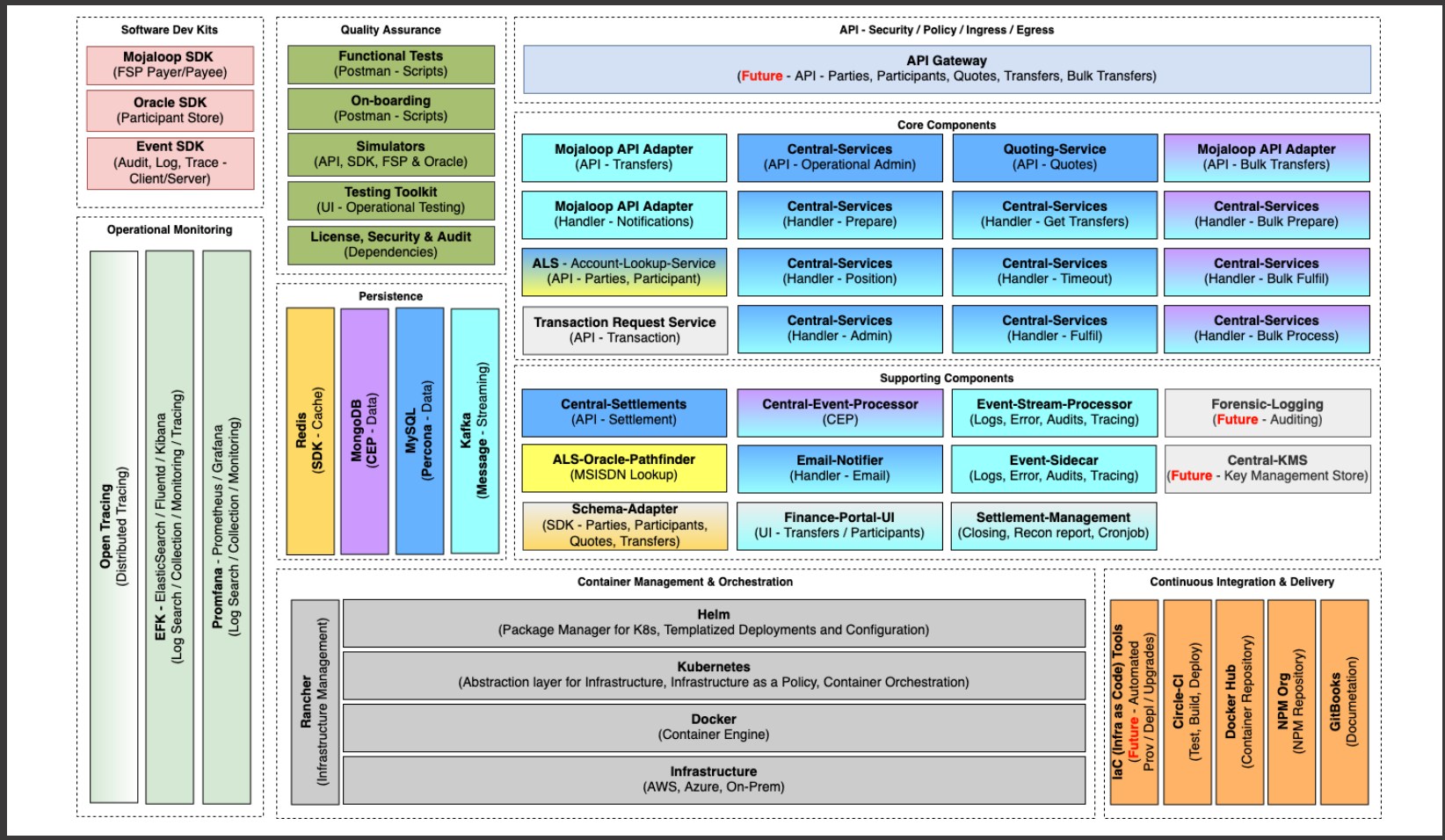
Central Ledger Service:
- **Participant** (DFSP) Accounts, Limits, & Position
- **Transfer** validation against limits & routing to Participants
- Real-time clearing against **Net Debit Cap** – **Position Calculation**
- Adding transfers to Settlement Windows

Mojaloop

Technical Overview

1. What is Mojaloop?
2. High-level overview, Static Demo & Architecture
3. Component Architecture
4. Global foot-print & Roll-out

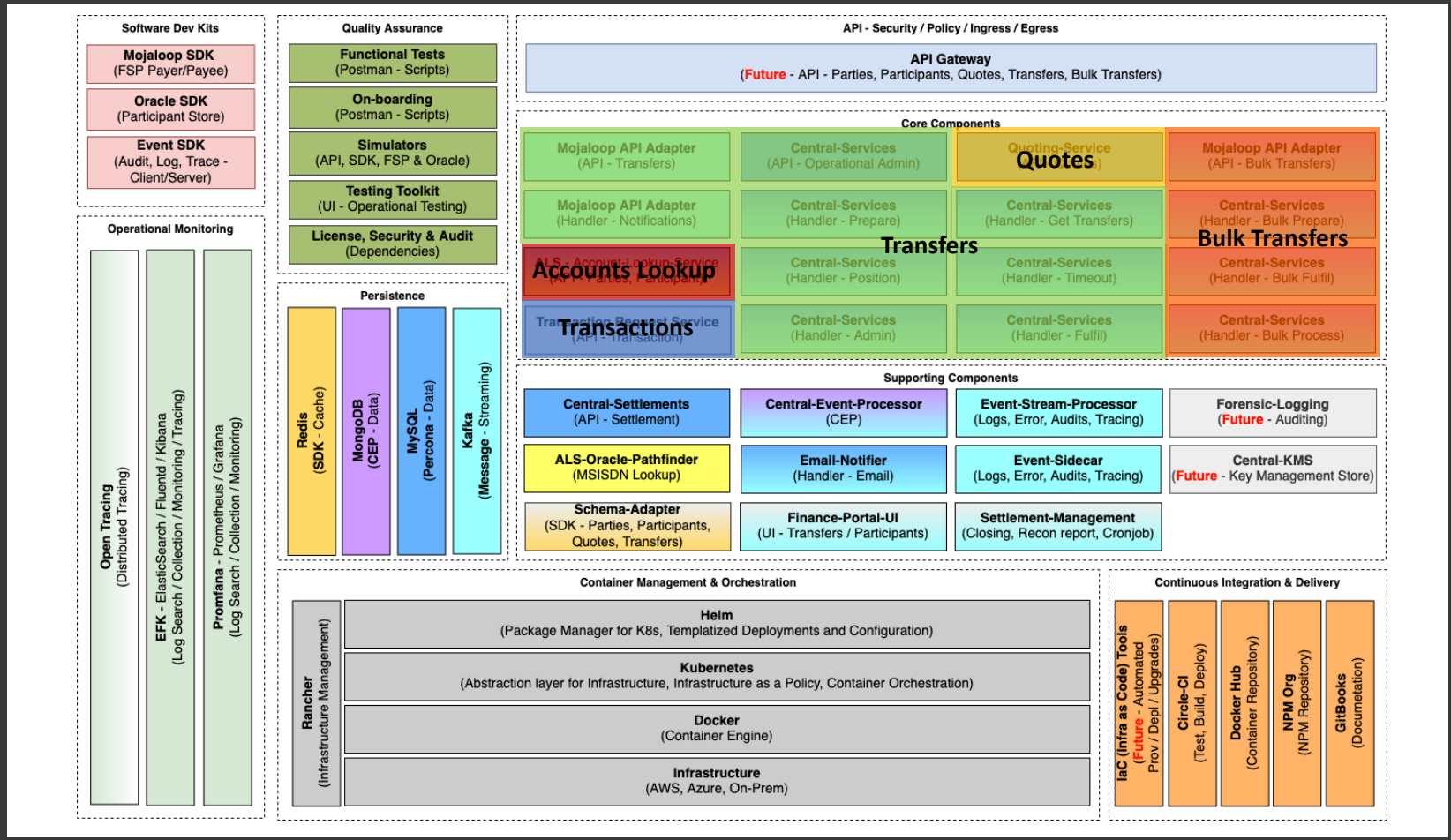
Component Architecture



April 2019

* Color-shading of core & supporting components match dependent persistent stores.

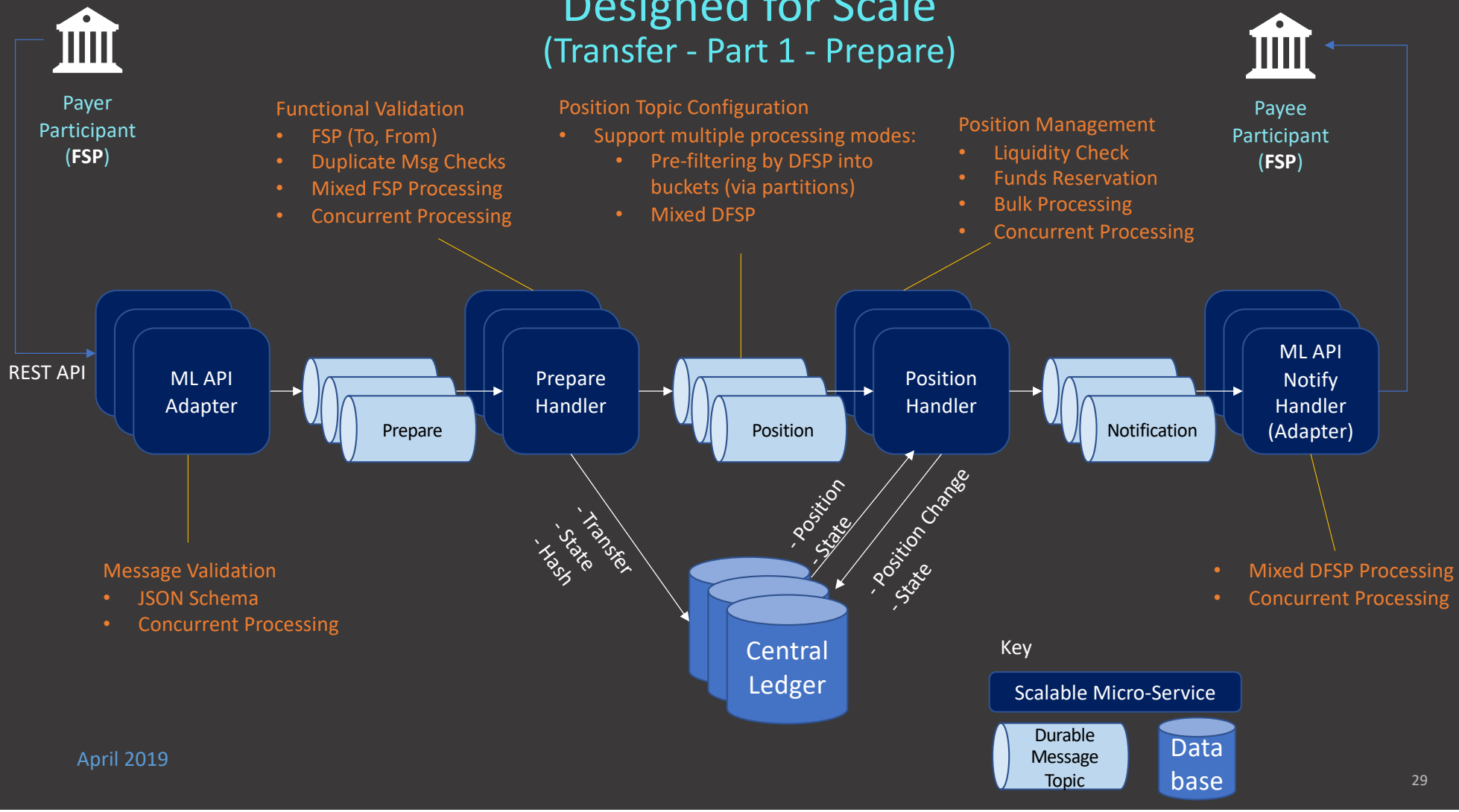
Component Architecture



April 2019

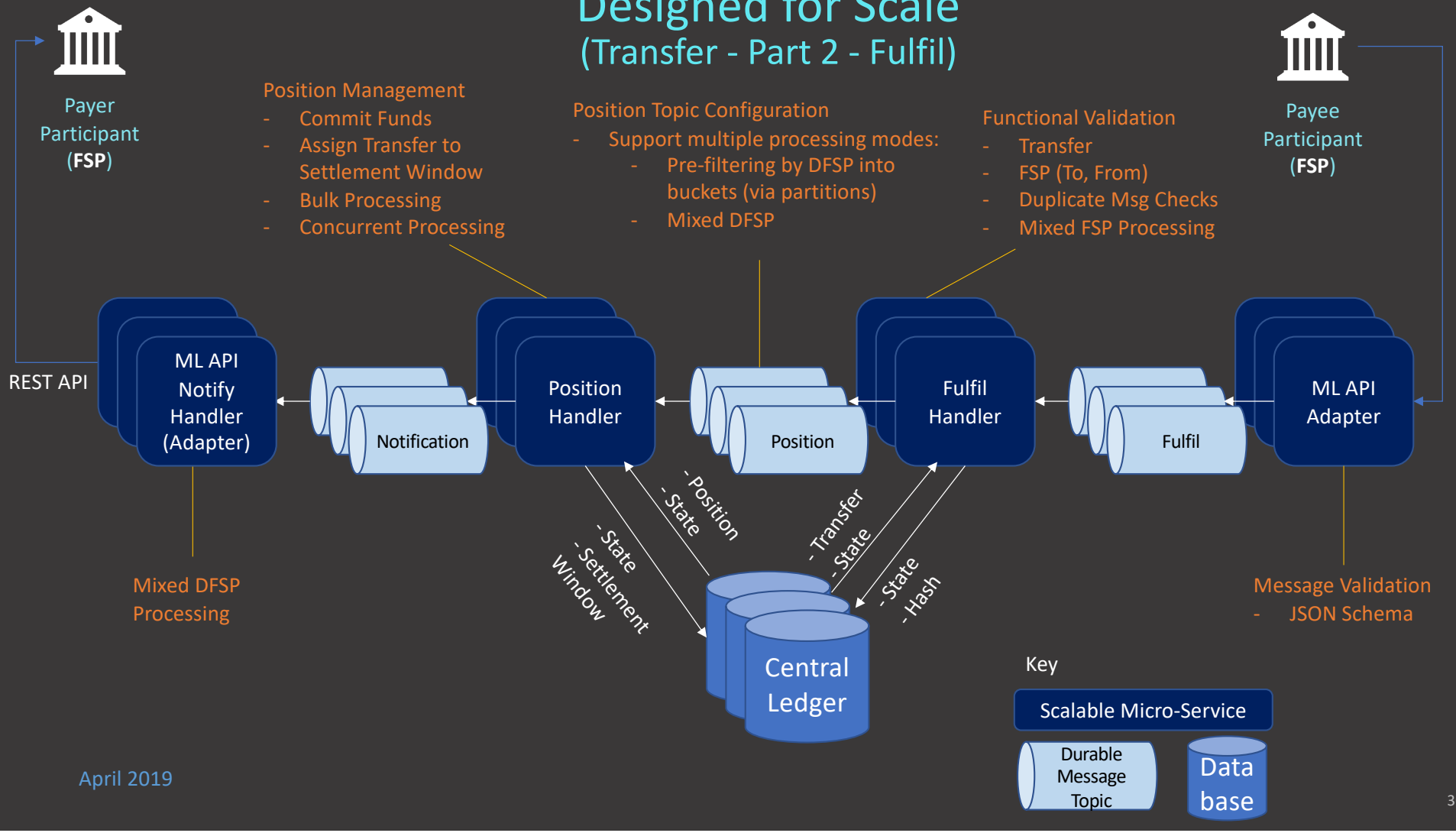
* Color-shading of core & supporting components match dependent persistent stores.

Designed for Scale
(Transfer - Part 1 - Prepare)



April 2019

Designed for Scale
(Transfer - Part 2 - Fulfil)



April 2019

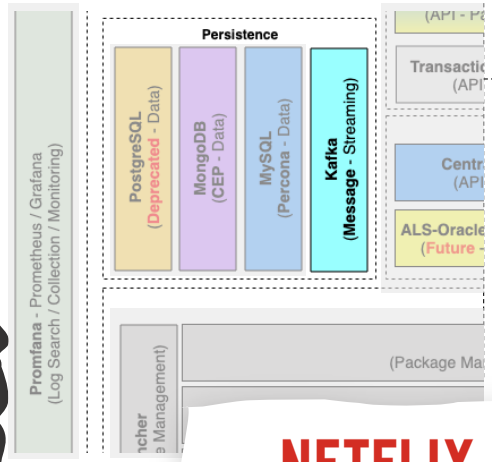
Messaging Platform – Kafka

Ref: *https://kafka.apache.org/*

kafka

What is Kafka?

Apache Kafka is a distributed message streaming platform.

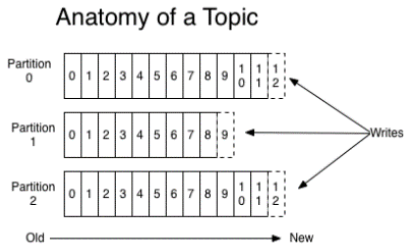


Why Kafka?

Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.

Store streams of records in a fault-tolerant durable way with history being saved for a desired period.

Process streams of records as they occur.



- For each topic, the Kafka cluster maintains a partitioned log.
- Each partition contains a sequence of records that is
  - Ordered; and
  - Immutable
- The records in the partitions are each assigned a sequential id number called the *offset* that uniquely identifies each record within the partition.

NETFLIX PayPal ORACLE IBM Rabobank

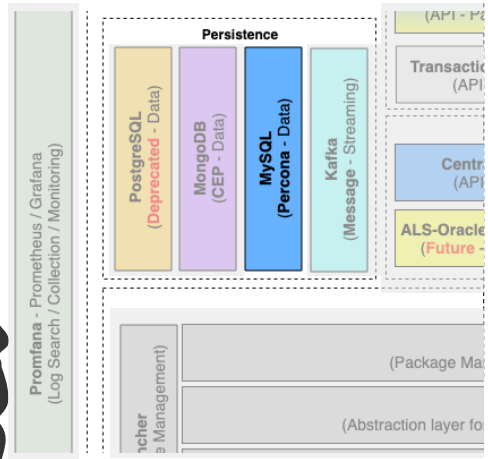
Used By

Storage Platform – Percona XtraDB Cluster

PERCONA

What is Percona XtraDB?

An open source, cost-effective, and robust MySQL clustering solution for businesses.



Ref: *https://www.percona.com/software/mysql-database/percona-xtradb-cluster*

Why Percona XtraDB?

- Cost-effective HA and scalability for MySQL with both Open Source and Enterprise support options
- Increased read/write scalability
- Zero data Loss
- Multi-master replication
- Works on-premises, cloud, hybrid, WAN, LAN, Kubernetes support (Helm)

Used By

Container Management & Orchestration – Helm

What is Helm?

Open Source Package Manager for Kubernetes through the use of Charts.

Charts help you define, install and upgrade releases for Kubernetes deployment via templates and configuration.

Why Helm?

Ref: http://helm.sh

Manage Complexity

Charts describe even the most complex apps; provide repeatable application installation, and serve as a single point of authority.

Easy Updates

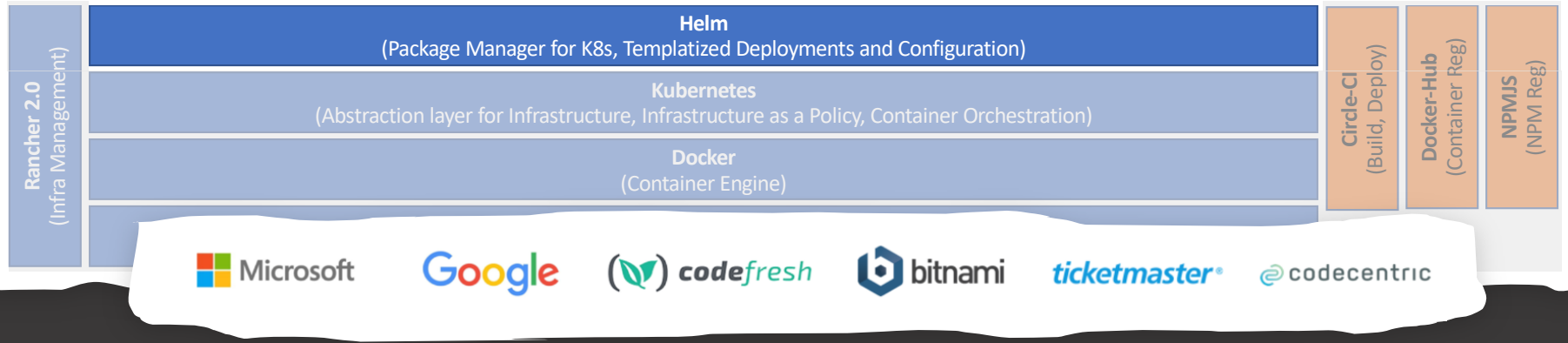
Take the pain out of updates with in-place upgrades and custom hooks.

Simple Sharing

Charts are easy to version, share, and host on public or private servers.

Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease.



Container Management & Orchestration – Kubernetes

What is Kubernetes?
Open-source system for automating deployment, scaling, and management of containerized applications.

kubernetes

Why Kubernetes?

Deploy your applications quickly and predictably
- Infrastructure as a Policy
- Abstraction of Infrastructure (Cloud, On-Prem)

Scale your applications on the fly
- Policy rule based scaling
- Limit hardware & resources by scaling horizontally up/down

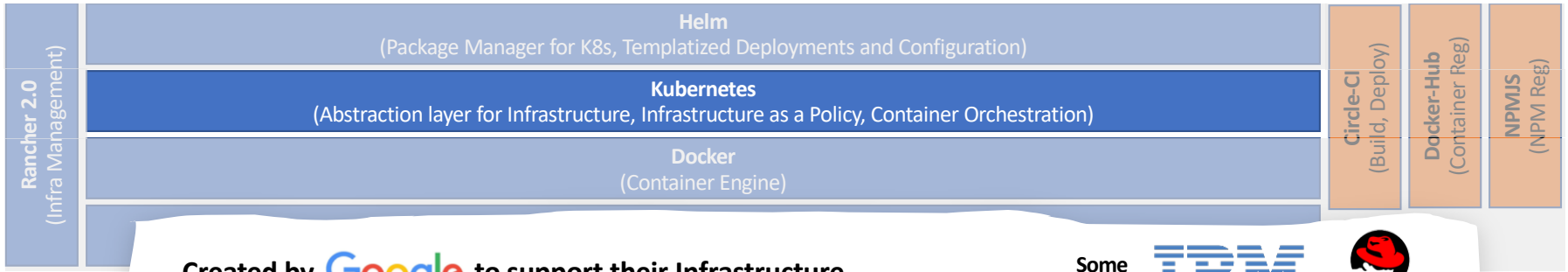
Roll out new features seamlessly
- Rolling updates

Discoverability
- Dynamic service resolution via DNS

Durability
- Self-healing
- Auto-[placement, restart, replication, scaling] based on Policies
- Load Balancing

Security
- Isolation through Containers, Network and Namespaces

Operations
- App config & secrets stored in distributed key-value store (etcd)
- Monitoring of containers



Created by Google to support their Infrastructure.

Some contributors: IBM redhat

April 2019

Deployment Architecture – Rancher Overview

Ref: http://rancher.com

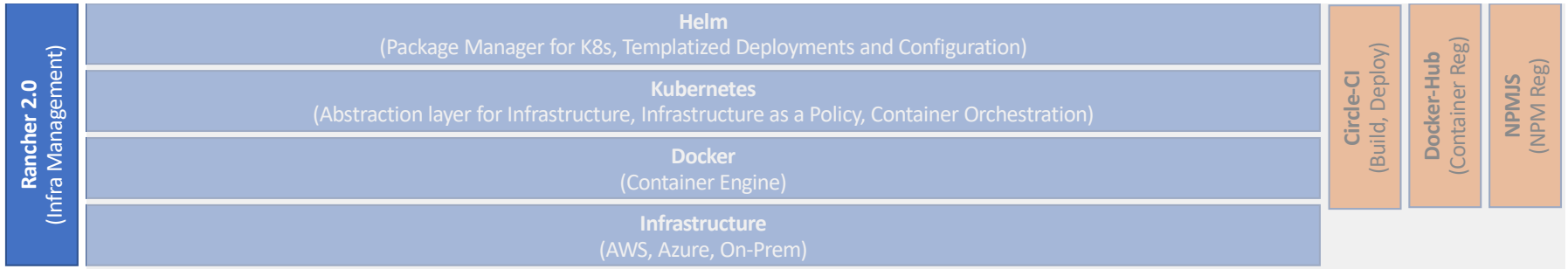
What is Rancher?

Rancher is an enterprise management plane for Kubernetes.

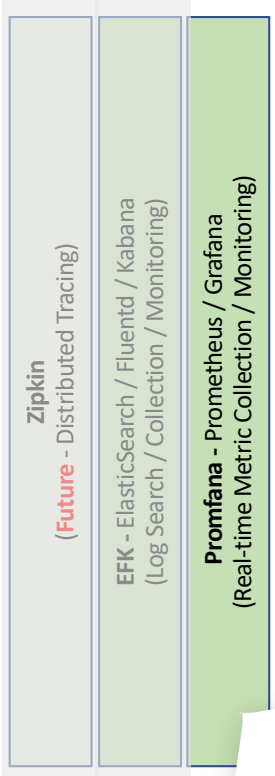
"**Every distro. Every cluster. Every cloud.**" ~ rancher.com

Why Rancher?

- Kubernetes Management
- Container Management
- Access Management (RBAC)
- Helm Repository Management
- Multi-environment Management (multi k8s clusters, On-prem, Azure, Google, AWS, etc)
- Multi-Provider provisioning (On-prem, Azure, Google, AWS, vSphere)
- Easily scale up/down Kubernetes clusters



Deployment Architecture – Ops Monitoring Overview



What is Metric Instrumentation?

Real-time operational visibility for:

- Performance
- Health
- Alerts

Ref: *http://prometheus.io*, *http://Grafana.com*

What is Promfana?

Prometheus

Leading open-source instrumentation solution for monitoring

Grafana

The open platform for beautiful analytics and monitoring

Why Promfana?

- Metric Instrumentation for Mojaloop
- Low overhead on nodejs (histograms + pull metric end-point)
- **Real-time metric visualization** for **Performance** and **Health monitoring** of the Mojaloop Stack

Dimensional data

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries

PromQL allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

Simple operation

Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically linked and easy to deploy.

Precise alerting

Alerts are defined based on Prometheus's flexible PromQL and maintain dimensional information. An alertmanager handles notifications and silencing.

Many client libraries

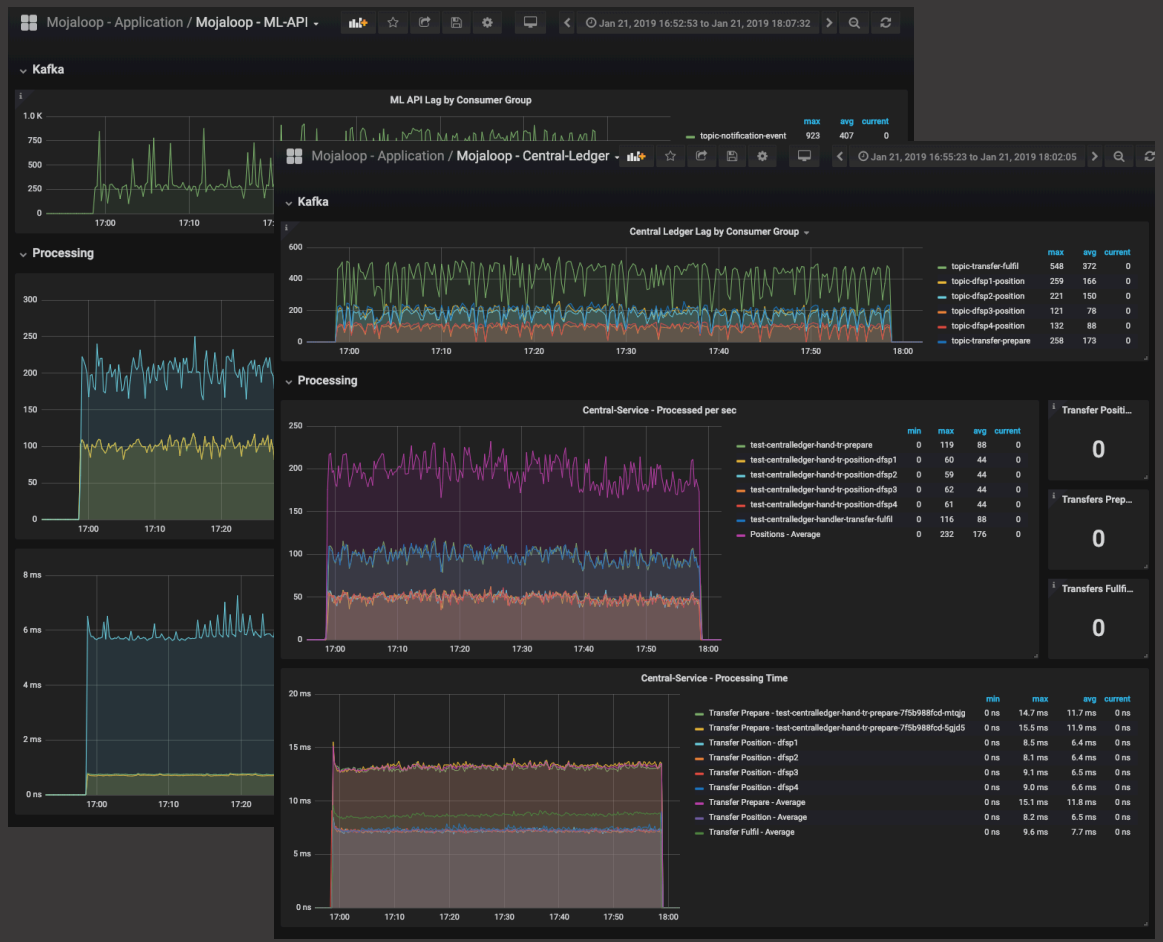
Client libraries allow easy instrumentation of services. Over ten languages are supported already and custom libraries are easy to implement.

Many integrations

Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as Docker, HAProxy, StatsD, and JMX metrics.

CoreOS docker DigitalOcean SOUNDCLOUD PERCONA ERICSSON weaveworks

Deployment Architecture – Ops Monitoring Dashboards



Documentation

http://mojaloop.io/helm/monitoring/

Mojaloop Application

- ML-API-Adapter -- nodejs + application
- Central-Ledger -- nodejs + application
- Simulators -- nodejs + application

Data Store

- MySQL Overview
- PXC Galera Overview
- PXC Galera Graphs

Messaging

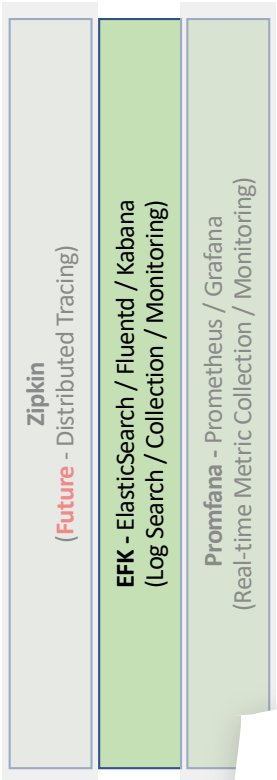
- Kafka Cluster Overview
- Kafka Topic Overview

Kubernetes

- Clusters
- Deployments

April 2019

Deployment Architecture – Ops Logging Overview



Ref: *http://prometheus.io*, *http://Grafana.com*

What is EFK (aka ELK)?

elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine.

fluentd

Open source data collector for unified logging layer, with ingestions into Elasticsearch.

kibana

Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack.

Why EFK?

- Central location and storage of all Mojaloop log files
- Management of log data (persistence, long-term storage, etc)
- Management of alert/events based on log data
- Log files are indexed and searchable
- Assist with tracing & trouble shooting Mojaloop's distributed micro-service logs

Why E-F-K and not E-L-K?

- Fluentd is used instead of Logstash due to its support & seamless integration for Kubernetes (k8s).
- K8s Pods/Containers are easily collected by Fluentd and ingested into ElasticSearch using the underlying K8s logging architecture.

Sprint Rabobank AIRBUS ebay vimeo Fermilab

Deployment Architecture – CircleCI Overview

circleci

What is CircleCI? Cloud based Continuous Integration & Deployment Platform

Ref: http://circleci.com

VCS Integration
CircleCI integrates with GitHub, GitHub Enterprise, and Bitbucket. Every time you commit code, CircleCI creates a build.

Automated Testing
CircleCI automatically tests your build in a clean container or virtual machine.

Automated Deployment
Passing builds are deployed to various environments so your product goes to market faster.

Notifications
Your team is notified if a build fails so issues can be fixed quickly.

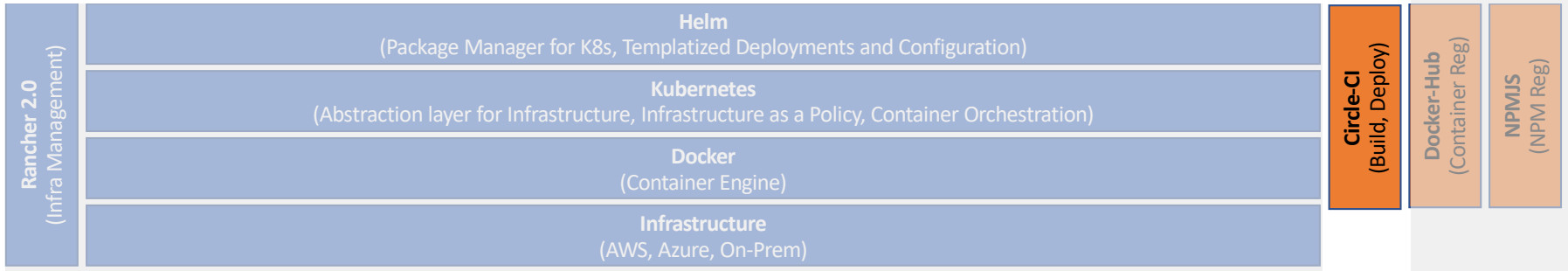
Why CircleCI?

Workflows for Job Orchestration
Orchestrate customizable job execution (such as build, test, deploy), giving complete control over your development process.

Language-Agnostic Support
Supports any language that builds on Linux or macOS, including C++, Javascript, .NET, PHP, Python, and Ruby.

First-Class Docker Support
Run any image from Docker's public/private registry or other common registries. Build Docker images, access Docker layer caching, Compose.

Powerful Caching
Speed up builds with expanded caching options, including images, source code, dependencies, and custom caches. Full control over cache save and restore points for optimal performance.



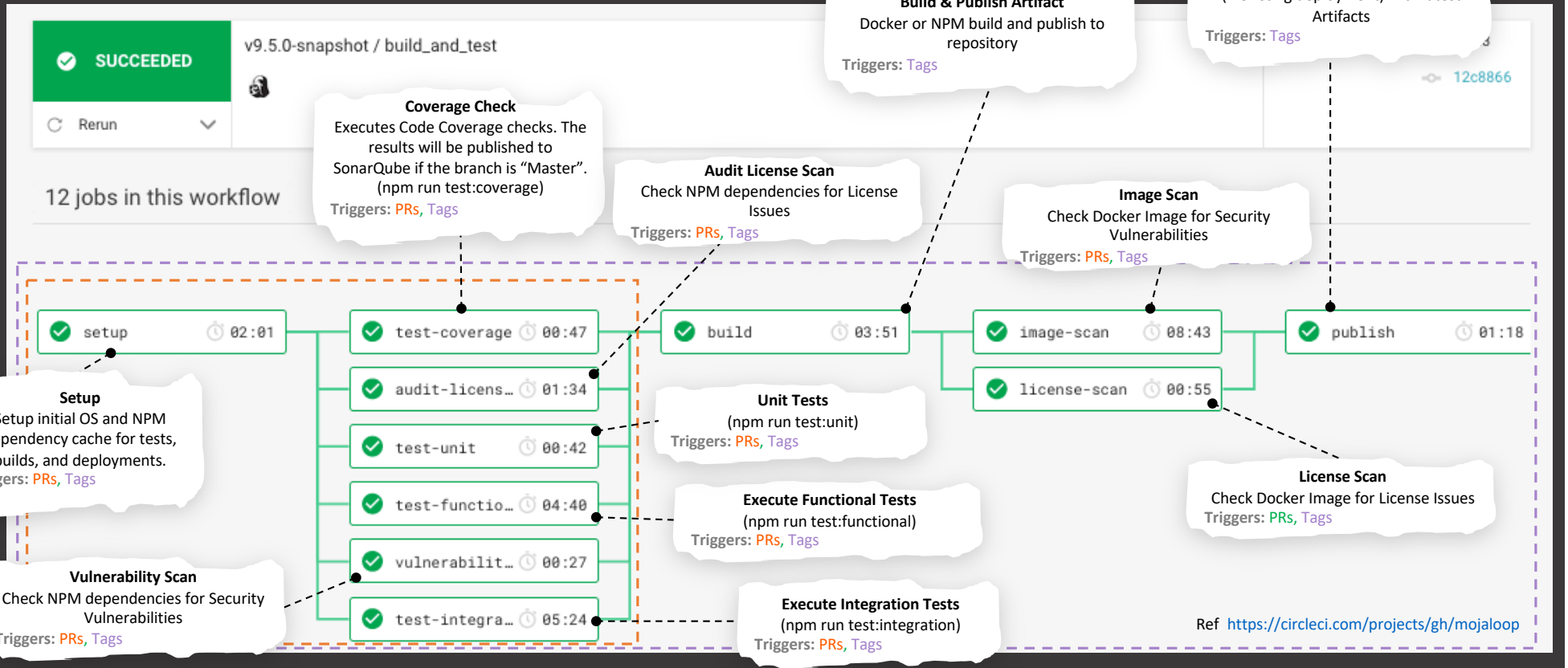
** Forrester names CircleCI a leader (https://www2.circleci.com/circleci-forrester-wave-leader-2017.html)*

April 2019

Deployment Architecture – CI/CD Pipeline

circleci

Triggers:
[●] Pull-Requests (inc. from external Forks)
[●] Publishing tagged Releases / Snapshots, etc



April 2019

Documentation – GitBooks Overview

GitBook

What is Gitbooks?

An open-source open documentation framework where teams can document everything from products, to APIs and internal knowledge-bases based on open-standards with community driven plugins.

Ref: http://gitbook.com
Ref: https://en.wikipedia.org/wiki/Markdown

Why Gitbooks?

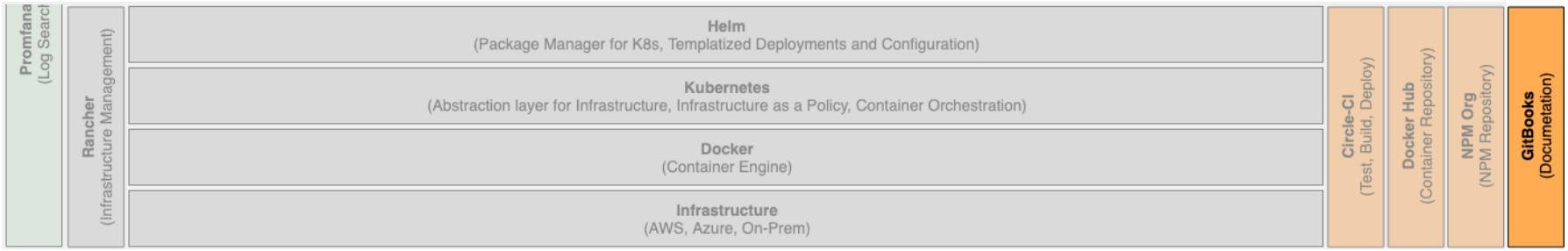
Markdown
Lightweight markup language with plain text formatting syntax supporting standard HTML, and CSS.

Imbed Generated Content
Embed generated sequence diagrams, openapi/swagger docs, etc.

Search
Find what you are looking for.

Plugins
Community plugins for generating content (e.g. plantuml, openapi/swagger docs), providing integration to Github, Slack, etc and themes (e.g. ToCs, Navigation, etc)

Cli
Gitbook-cli to build static-content, with support for local testing. Also supports auto-build sense when changes are made locally when testing.

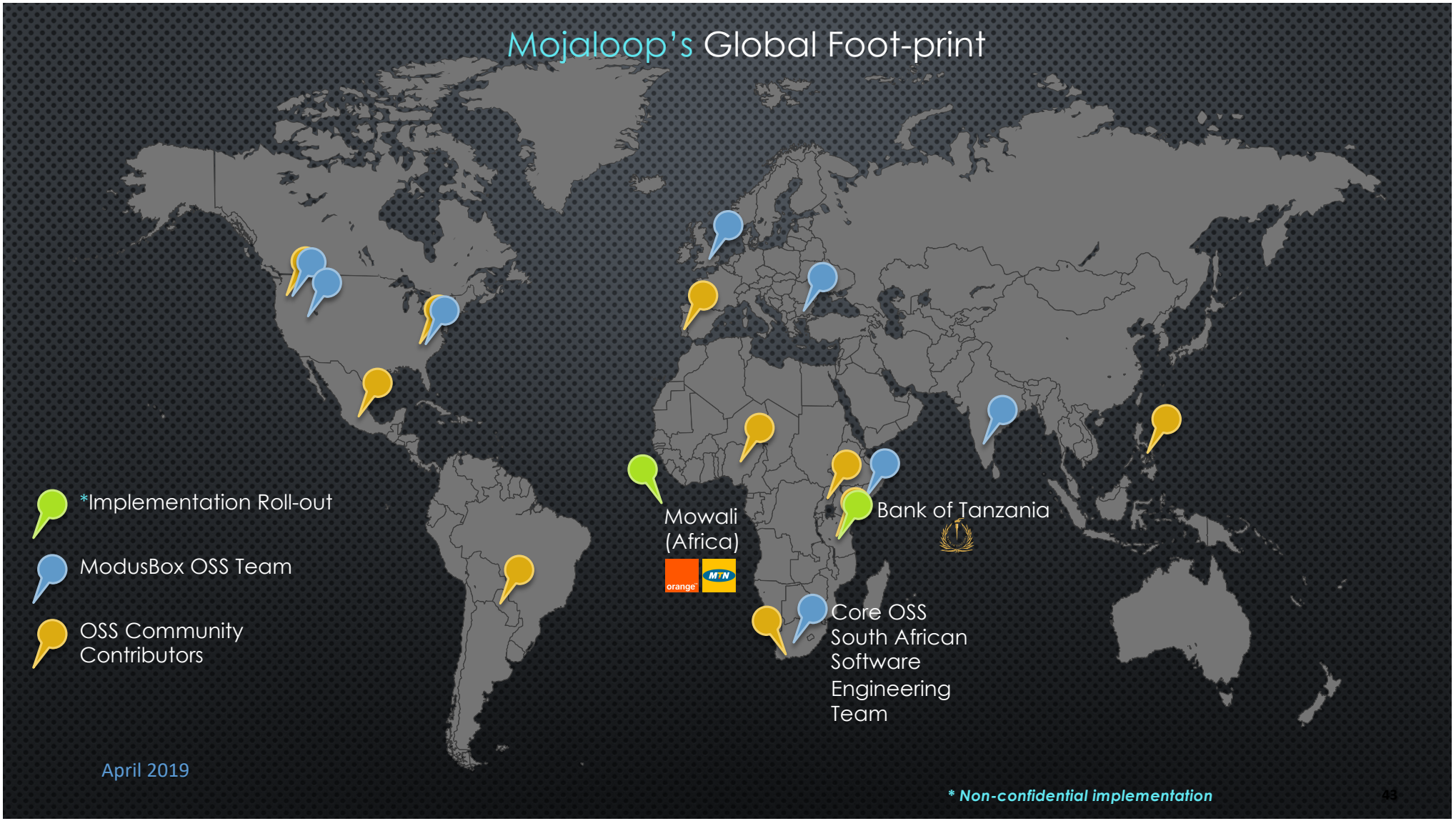


Mojaloop

Technical Overview

1. What is Mojaloop?
2. High-level Overview, Static Demo & Architecture
3. Component Architecture
4. Global foot-print & Roll-out

Mojaloop's Global Foot-print



*** Non-confidential implementation**


mojaloop
Mojaloop
Questions?
April 2019

Mojaloop

Appendix

April 2019

Sequence Diagrams: Overview

1. Overview of Sequence Diagrams
2. High level Design for :
    a. Prepares, Positions, Fulfils, Rejections, Timeouts
    b. Operations: Settlements, Positions, Limits, callback URLs, etc.
3. List of Sequence Diagrams
4. Location: https://mojaloop.io/documentation/mojaloop-technical-overview

API First Approach

Key
[●] Implemented using API First Approach
[●] Not implemented – To be re-factored in future

Background

Legacy Mojaloop API components were implemented using a Code First approach (even if a contract existed).

What is API First Approach?

An Interface Contract is defined up-front, which is then used to generate Base-code which includes validations, routes, documentation, etc.

Benefits?

- Faster implementation time as Base-code is auto-magically created for developers, only requiring logic to be implemented.
- QA Improvement
  - Validations are done auto-magically adhering to contract
  - Routes adhere to contract
  - Documentation adheres to contract

Mojaloop & Operational APIs

[●] Mojaloop-API-Adapter
[●] Central-Ledger
[●] Central-Settlements

Example:
https://github.com/mojaloop/central-settlement

Tools used

Hapi-openapi

- API schema validation.
- Routes based on the OpenAPI document.
- API documentation route.
- Input validation.
- Ref: https://github.com/krakenjs/hapi-openapi

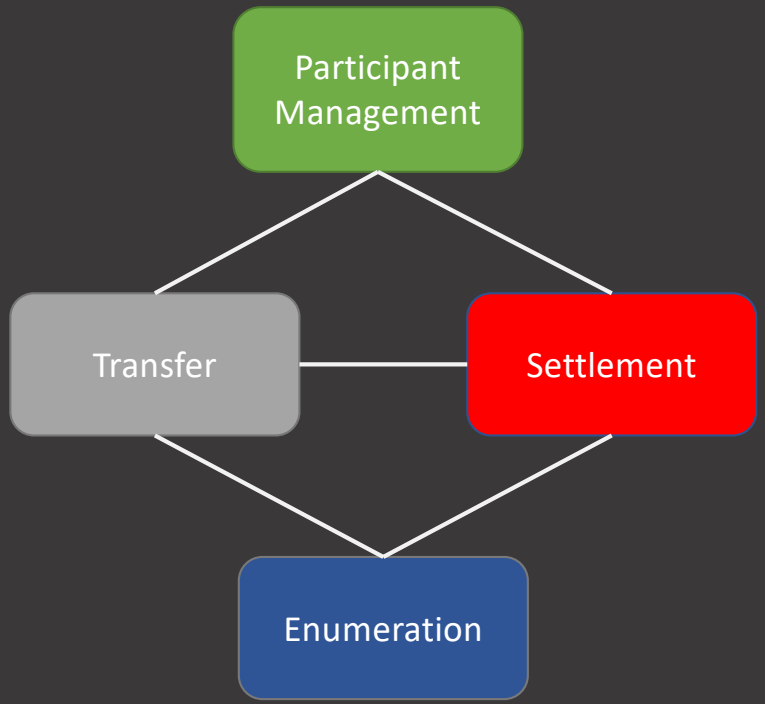
Database Design

Database Design Objectives

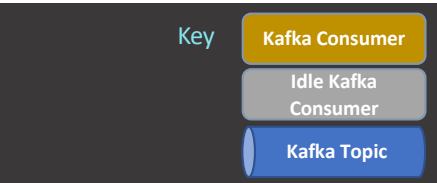
1. Align the data model to support to the Mojaloop specification
2. Design for efficiency and reduce locking
3. Data integrity maintained through a idempotency pattern

How was this achieved?

1. Redesign scheme to reflect the Mojaloop API and support the state transitions.
2. Inserts are used instead of updates to ensure reduced locking, and increased speed of data “updates”
3. Inserts are used to ensure an idempotent pattern for data “updates” is maintained. The result being that all “updates” have a history.
4. Support for batch processing of DFSP position, maintain repeatable and deterministic outcomes of rule processing and “running balance” for positions.

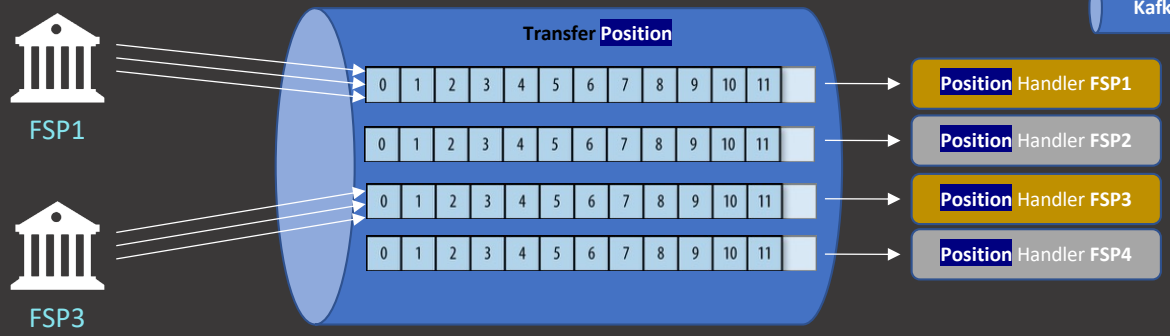


Optimization – Position Handler Processing Modes



Pre-filtering of DFSPs into Buckets

- Partition strategy is keyed for Positions.
- Message keys are used to route each message to a specific Partition (Bucket).
- Dedicated handlers for DFSP specific workloads (Future: Easier In-memory processing)
- *Possible issue/limitation*:
  - Limited scalability by number of FSPs
  - Possible idle handlers (cost implication)
  - Requires process for increasing partitions on Kafka (possible downtime, migration strategy, or temporal performance hit).



Mixed DFSP Processing

- Partition strategy can be configured as Keyed (same as PI4) or Random
- Improved scalability (no limit)
- All Handlers are utilized (efficient hardware usage)
- Future possibility of consolidating handlers (prepare, position & fulfil)
- *Possible issue/limitation*:
  - Increased contention & locking on the database persistence layer.

Definition of terms: 7. participant position

1. A *participant position* is a net position of all activity
2. A position applies to a switch account for a scheme participant in a specific currency.
3. A position represents:
    1. The activity in an account which is a consequence of transfers flowing through the system
    2. Adjustments to the account as directed by the scheme
4. The position is calculated as:
    1. The sum of reservations made for prepared but unfulfilled transfers where the participant is the payer.
    2. The sum of fulfilled transfers where the participant is the payee.
    3. Any changes to the account position directed by the scheme.

April 2019

* This has been blatantly stolen from Michael Richard's Dictionary-O-Mojaloop-Terms

Definition of terms: 7. Net Debit Cap

1. In principle, a participant is not allowed to trade in the system if their position exceeds their Net Debit Cap Limit.
    1. Which means in this context: *if the sum of credits to and debits from their switch account is less than the net of their transfer activities in the account.*
2. The difference between the sum of credits and debits to a participant's switch account and the net of a participant's transfer activities in the account is their *liquidity cover*.
    - ❖ In this case, "transfer activities" includes transfers where the participant is the payer and the transfer has been requested but not yet completed.
3. So the principle is: a participant must always be able to cover their liabilities. We call this a *liquidity cover requirement*
4. Seems clear enough…

April 2019

* This has been blatantly stolen from Michael Richard's Dictionary-O-Mojaloop-Terms