



# Mojaloop

---

## Technical Overview

By Miguel de Barros  
[miguel.debarros@modusbox.com](mailto:miguel.debarros@modusbox.com)  
v1.0

April 2019





# 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>

Older docs are still found in  
<https://github.com/mojaloop/docs>

(in-process of being migrated to documentation  
repo)

# Mojaloop

---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

# Mojaloop

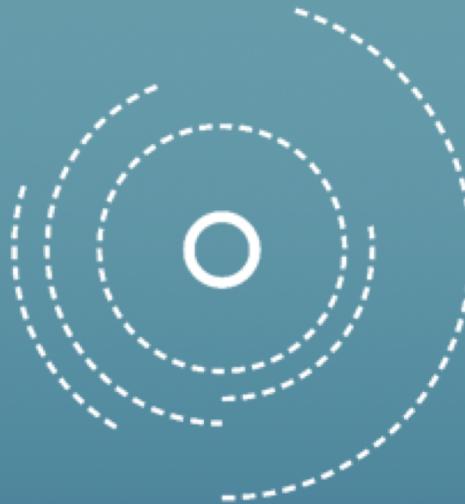
---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. 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



mojaloop

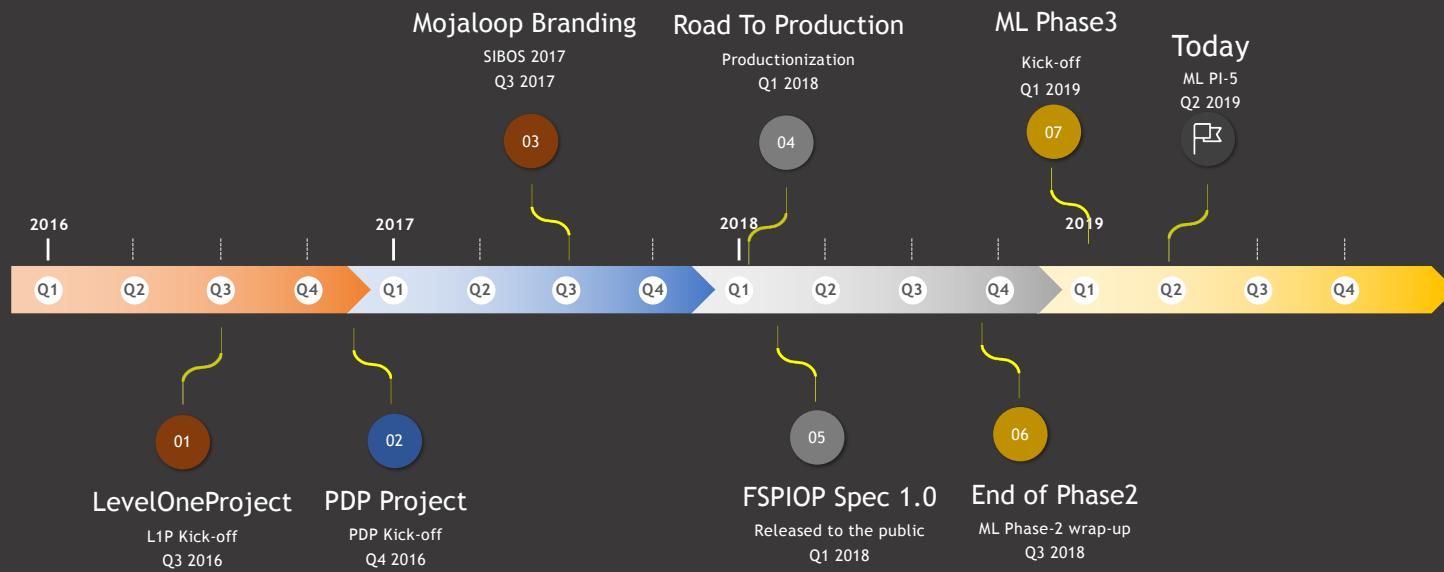
# Mojaloop

---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

# Mojaloop – History & Evolution



April 2019



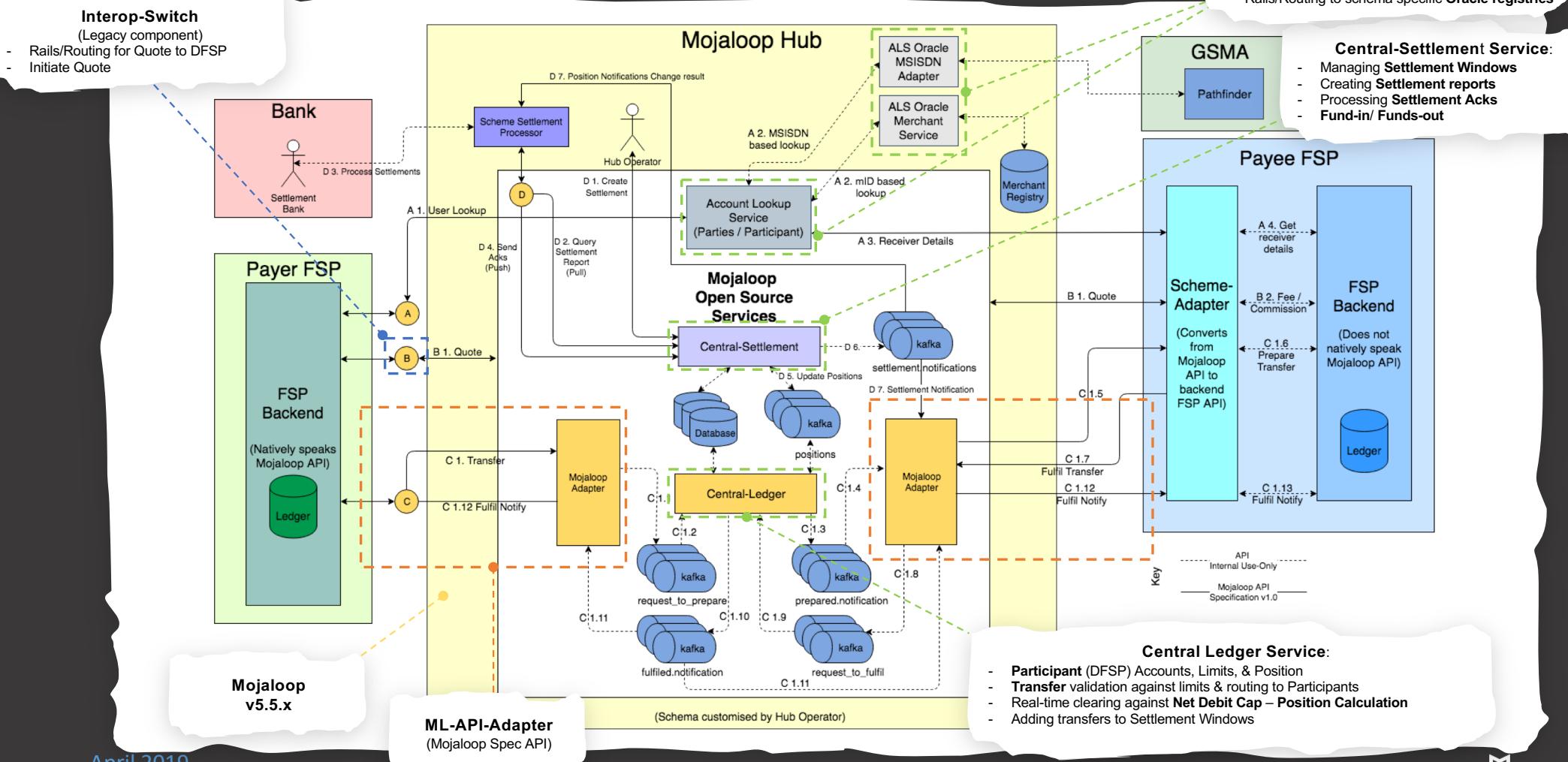
# Mojaloop

---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

# High-level Architecture – Current (PI5)



April 2019

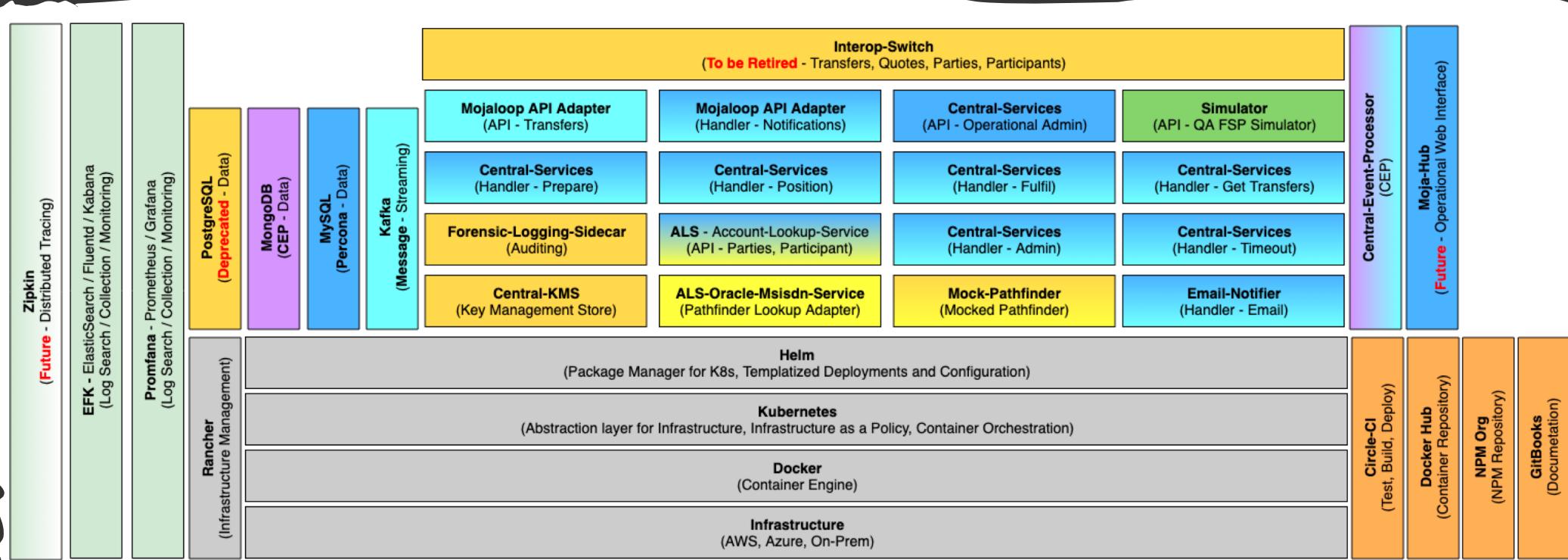
# Mojaloop

---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

# Component Architecture – Current (PI5)





# Deployment Architecture – Helm Overview

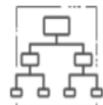
Ref: <http://helm.sh>

## 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?



### Manage Complexity

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

### Easy Updates

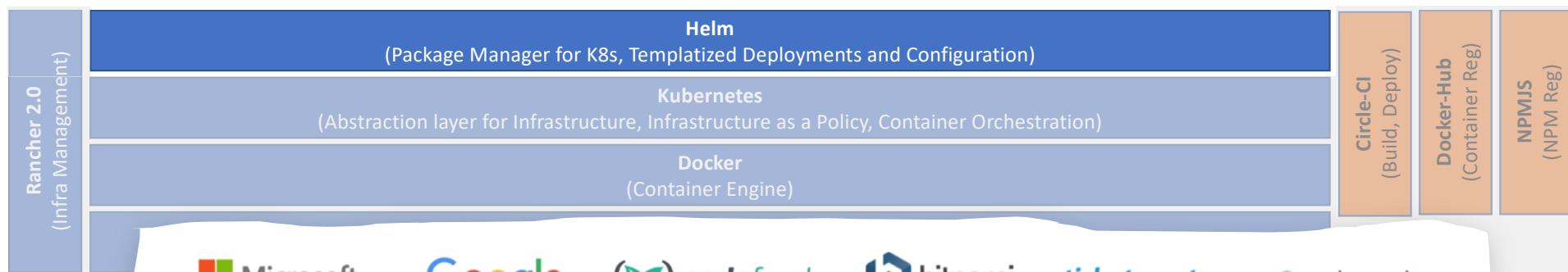
Take the pain out of updates with in-place upgrades and custom hooks.

### 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.



# Deployment Architecture – Kubernetes Overview



## What is Kubernetes?

Open-source system for automating deployment, scaling, and management of containerized applications.

## 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



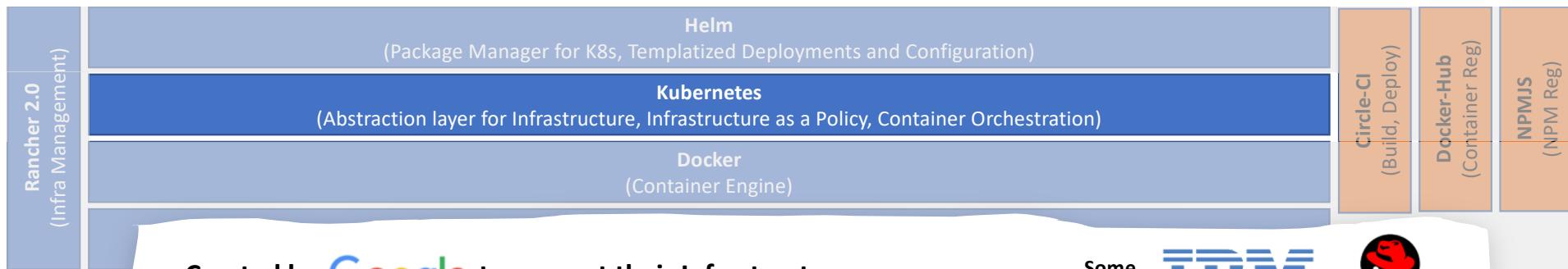
### Security

- 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:





# Deployment Architecture – Rancher Overview

Ref: <http://rancher.com>

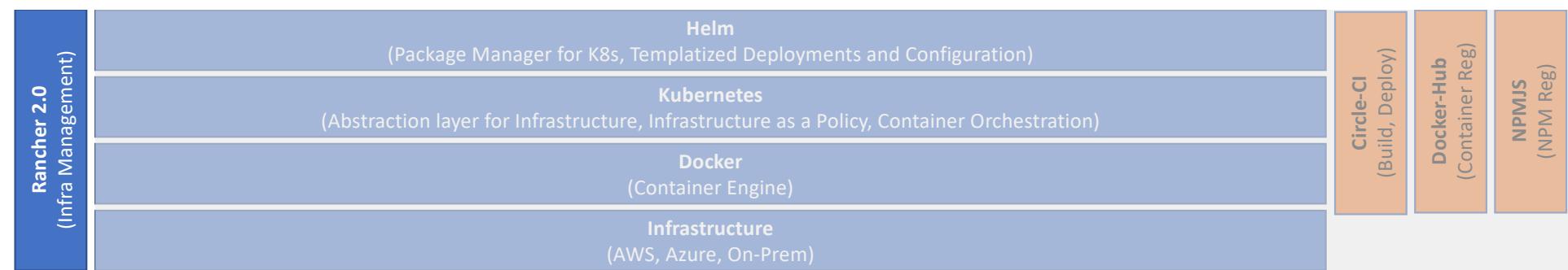
## What is Rancher?

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

Ref: <http://prometheus.io>, <http://Grafana.com>



## What is Metric Instrumentation?

Real-time operational visibility for:

- Performance
- Health
- Alerts

## 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

## What is Promfana?



Leading open-source instrumentation solution for monitoring



The open platform for beautiful analytics and monitoring

### 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



SOUNDCLLOUD



PERCONA

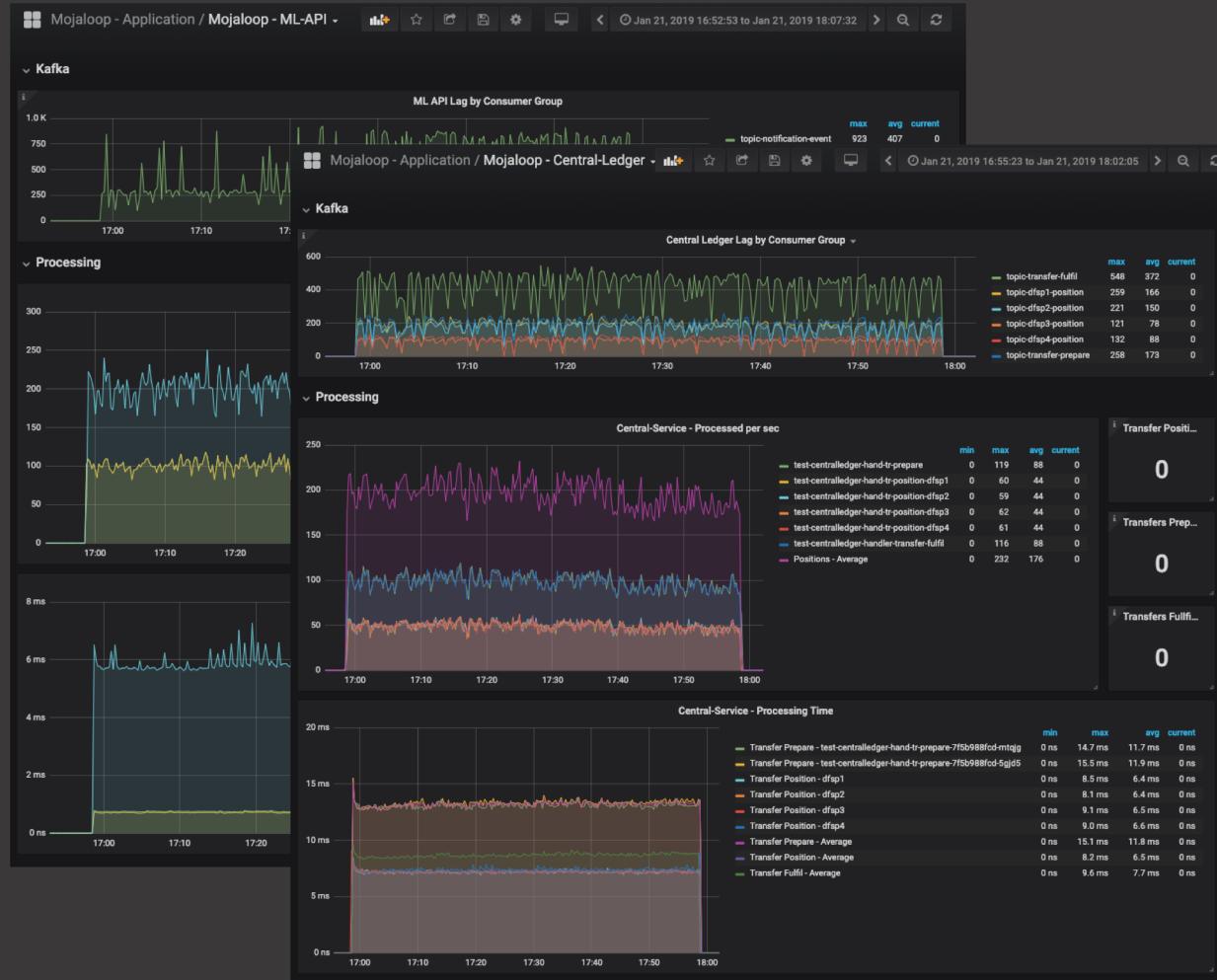


ERICSSON



weaveworks

# Deployment Architecture – Ops Monitoring Dashboards



PI-6

April 2019

## 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



16

# 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.



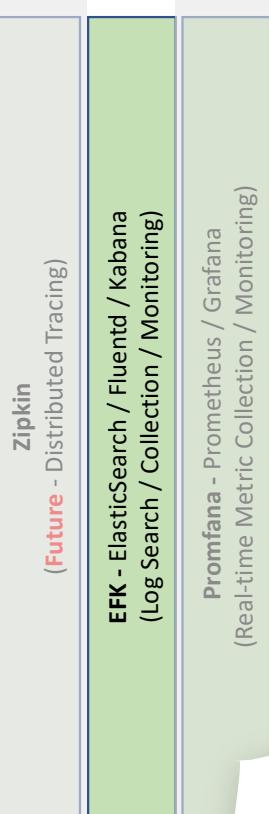
**fluentd**

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.





# Deployment Architecture – CircleCI Overview

## 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.



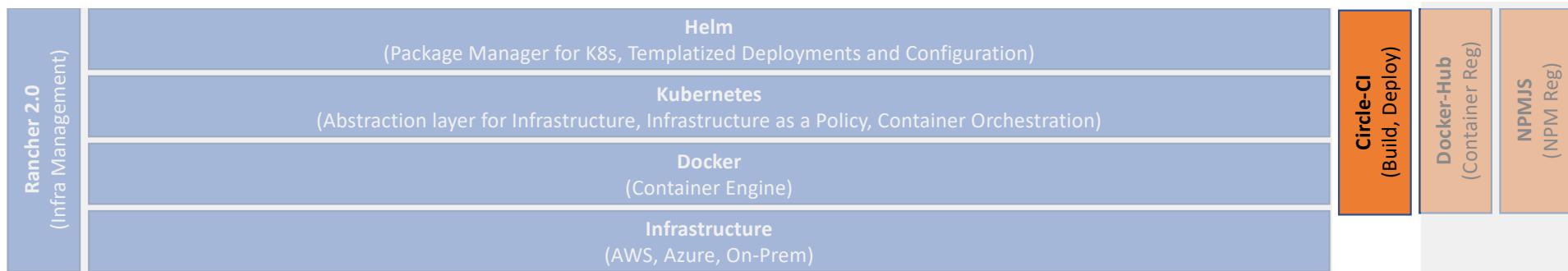
### First-Class Docker Support

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



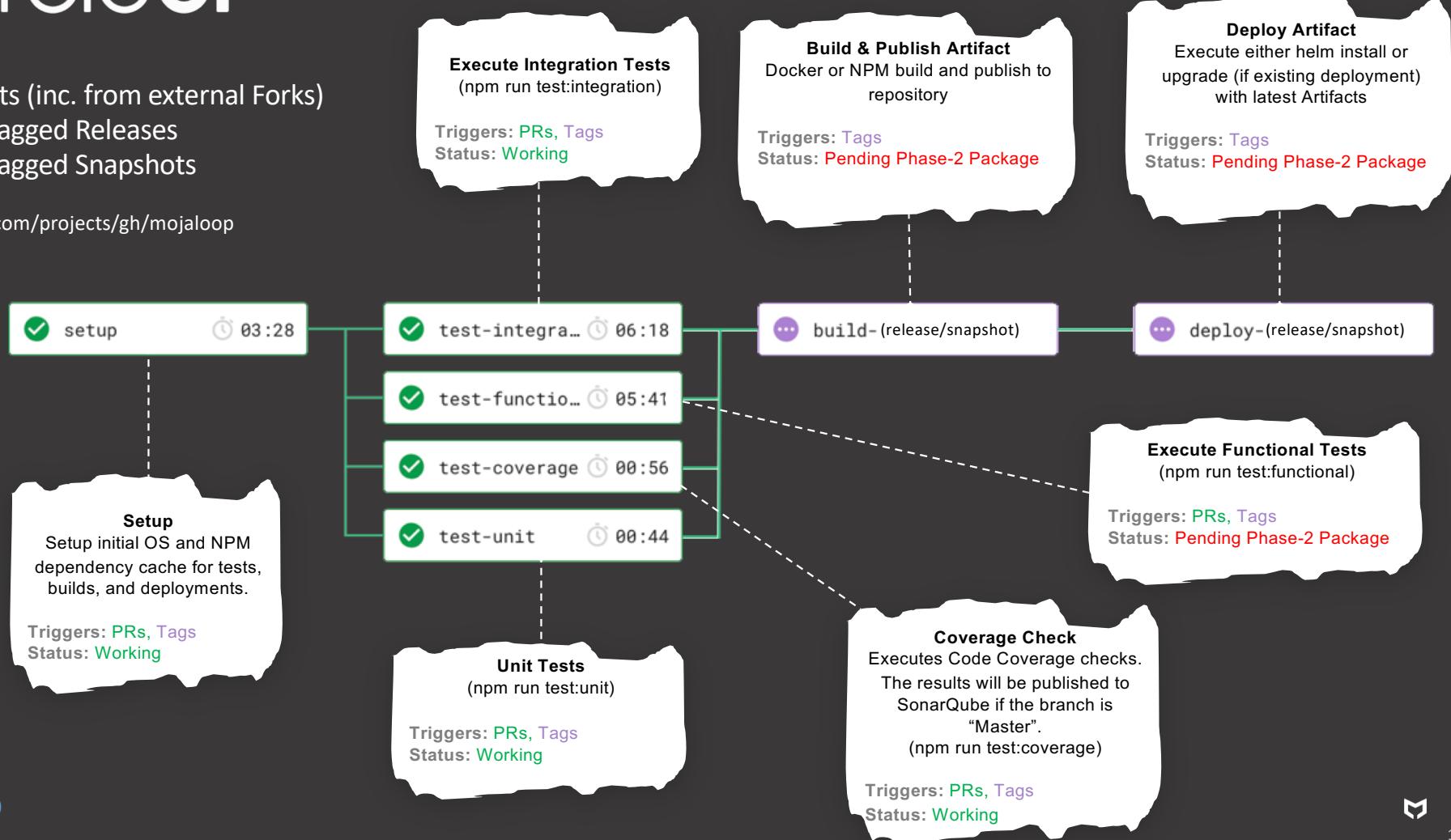
18

# Deployment Architecture – CI/CD Pipeline



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

Ref: <https://circleci.com/projects/gh/mojaloop>



April 2019



19



# Documentation – GitBooks Overview

## 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.

## Why Gitbooks?



### Markdown

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



### Embed Generated Content

Embed generated sequence diagrams, openapi/swagger docs, etc.



### Search

Find what you are looking for.



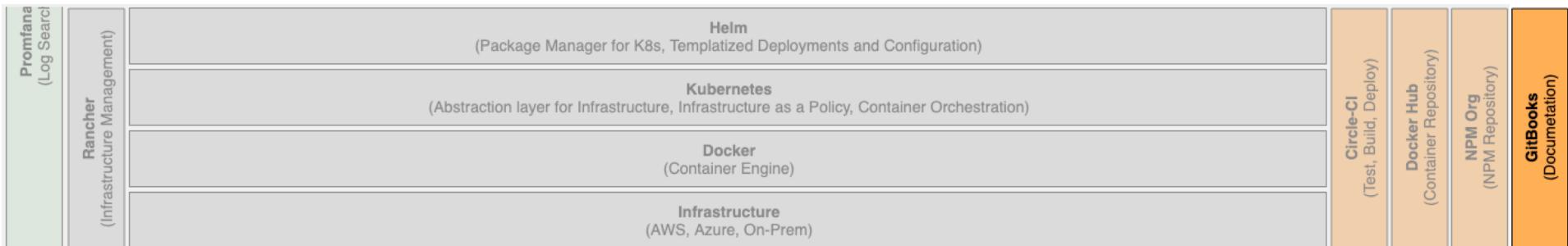
### Plugins

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. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

# Current (PI5) Switch Functionality – Mojaloop Specification

## Mojaloop v1.0 – API Specification

### Transfers

- [●] POST - Prepare
- [●] PUT - Response
- [●] **PUT – Error**
- [●] Outgoing
- [●] **Incoming**
- [●] **GET - Query**

### Parties

- [●] **GET - Request**
- [●] **PUT - Response**
- [●] **PUT - Error**

### Quotes

- [●] POST - Request
- [●] PUT - Response
- [●] PUT - Error
- [○] GET - Query

### Participants

- [●] **POST - Create**
- [●] **PUT - Response**
- [●] **POST - Bulk Create**
- [●] **PUT - Error**
- [●] **DEL - Delete**

### Transactions

- [○] PUT - Response
- [○] GET - Query

### TransactionRequests

- [○] POST - Request
- [○] PUT - Response
- [○] PUT - Error
- [○] GET - Query

### Authorizations

- [○] GET - Request
- [○] PUT - Response
- [○] PUT - Error

### BulkTransfers

- [○] POST - Request
- [○] PUT - Response
- [○] PUT - Error
- [○] GET - Query

### BulkQuotes

- [○] POST - Request
- [○] PUT - Response
- [○] PUT - Error
- [○] GET - Query

### Key

- [●] Fully implemented
- [●] Legacy Code
- [●] Partially implemented
- [●] Not implemented
- [○] Out of Scope for PI5

- Interim:
  - subId not supported currently
  - no validations applied to “update” operations
  - full design for Participants POST – Create is pending

## Current (PI5) Switch Functionality – Operations (1 of 2)

### Operational – Use Cases

#### Participants

- [●] Manage Participants
  - [●] Create Initial Value
  - [●] Query
  - [●] Update
- [●] Manage Participant Limits
  - [●] Create Initial Value
  - [●] Query
  - [●] Update
- [●] Manage Callback URLs
  - [●] Create Initial Value
  - [●] Query
  - [●] Update

#### Settlements

- [●] Open, close Settlement Windows
- [●] Query Settlement Windows
- [●] Query Settlement Report
- [●] Create/Trigger Settlement with Windows
- [●] Process successful Settlement Acknowledgements
- [●] Reconcile Positions based on successful Settlements
- [●] Process failed Settlement Acknowledgements

#### Positions

- [●] Query Positions
- [●] Manage Positions
  - [●] Create Initial Value
  - [●] Query
  - [●] Update

#### Oracles

- [●] Manage Oracles
  - [●] Create
  - [●] Query
  - [●] Update
  - [●] Delete

#### Key

- [●] Fully implemented
- [●] Legacy Code
- [●] Partially implemented
- [●] Not implemented
- [○] Out of Scope



## Current (PI5) Switch Functionality – Operations (2 of 2)

### Central Services – API Specification

#### Participants

- [●] POST - Create
- [●] GET - Query
- [●] PUT - Update

#### Participants Limits

- [●] POST - Set initial Position
- [●] Manage Limits
- [●] POST - Set Limits
- [●] PUT - Update Limits
- [●] GET - Query Limits

#### Positions

- [●] GET - Query by FSP

#### Participants Callback

- [●] POST - Set Callback URIs
- [●] PUT - Update Callback URIs
- [●] GET - Query Callback URIs

### Central Settlements – API Specification

#### Settlement Windows

- [●] POST - Open new window, and close previous
- [●] GET - Query

#### Settlements

- [●] POST – Create/trigger new Settlement with associated Windows
- [●] PUT - Receive Acknowledgments from Settlement Providers
- [●] Process successful Settlement acknowledgements
- [●] Reconcile Positions based on successful Settlements
- [●] GET – Query

#### Key

- [●] Fully implemented
- [●] Legacy Code
- [●] Partially implemented
- [●] Not implemented
- [○] Out of Scope

# Mojaloop Hackathon

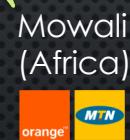
---

## Technical Overview

1. What is Mojaloop?
2. History & Evolution
3. High-level Architecture
4. Component Architecture
5. Switch Functionality
6. Global foot-print & Roll-out

## Mojaloop's Global Foot-print

- \*Implementation Roll-out
- ModusBox OSS Team
- OSS Community Contributors



Mowali  
(Africa)

Bank of Tanzania

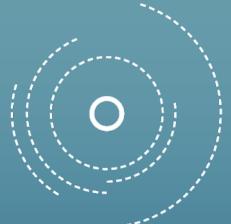


Core OSS  
South African  
Software  
Engineering  
Team

April 2019

\* Non-confidential implementation





mojaloop

# 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://github.com/mojaloop/docs/tree/develop/CentralServices/seq\\_diagrams](https://github.com/mojaloop/docs/tree/develop/CentralServices/seq_diagrams)

# Phase-2: Sequence Diagrams - 1

Topic	Sequence Diagram Name	Filename
Events	<ul style="list-style-type: none"><li>9.1.0. Event Handler Placeholder</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-event-9.1.0.svg</a></li></ul>
Fulfil	<ul style="list-style-type: none"><li>2.1.0. DFSP2 sends a Fulfil Success Transfer request</li><li>2.1.1. Fulfil Handler Consume (Success)</li><li>2.2.1. Fulfil Handler Consume (Reject)</li><li>2.2.0. DFSP2 sends a Fulfil Reject Transfer request</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-fulfil-2.1.0.svg</a></li><li><a href="#">seq-fulfil-2.1.1.svg</a></li><li><a href="#">seq-reject-2.2.1.svg</a></li><li><a href="#">seq-reject-2.2.0.svg</a></li></ul>
Notification	<ul style="list-style-type: none"><li>1.1.4.a. Send notification to Participant (Payer/Payee) (single message)</li><li>1.1.4.b. Send notification to Participant (Payer/Payee) (batch messages)</li><li>5.1.1. Notification Handler for Rejections</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-prepare-1.1.4.a.svg</a></li><li><a href="#">seq-prepare-1.1.4.b.svg</a></li><li><a href="#">seq-notification-reject-5.1.1.svg</a></li></ul>
Participant	<ul style="list-style-type: none"><li>1.0.0 Get Participant Limit Details</li><li>2.06 Design an API to manage NET DEBIT CAP #330</li><li>3.1.0 Add Participant Callback Details</li><li>3.1.0 Get Participant Callback Details</li><li>4.1.0 Get Participant Position Details</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-get-participant-limit-1.1.0.svg</a></li><li><a href="#">seq-manage-participant-limit-1.1.0.svg</a></li><li><a href="#">seq-callback-add-3.1.0.svg</a></li><li><a href="#">seq-callback-3.1.0.svg</a></li><li><a href="#">seq-participants-positions-query-4.1.0.svg</a></li></ul>

# Phase-2: Sequence Diagrams - 2

Topic	Sequence Diagram Name	Filename
Position	<ul style="list-style-type: none"><li>1.0.0 Create initial position and limits for participant</li><li>1.1.2.a. Position Handler Consume (single message)</li><li>1.1.2.b. Position Handler Consume (batch messages)</li><li>1.3.0 Position Handler Consume (single message)</li><li>1.3.1 Prepare Position Handler Consume</li><li>1.3.2 Fulfil Position Handler Consume</li><li>1.3.3 Abort Position Handler Consume</li><li>2.2.2. Position Handler Consume (Reject)</li><li>4.2.0 Get Positions of all Participants</li><li>uc Position Handler</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-participant-limits-1.0.0.svg</a></li><li><a href="#">seq-prepare-1.1.2.a.svg</a></li><li><a href="#">seq-prepare-1.1.2.b.svg</a></li><li><a href="#">seq-position-1.3.0.svg</a></li><li><a href="#">seq-position-1.3.1-prepare.svg</a></li><li><a href="#">seq-position-1.3.2-fulfil.svg</a></li><li><a href="#">seq-position-1.3.3-abort.svg</a></li><li><a href="#">seq-reject-2.2.2.svg</a></li><li><a href="#">seq-participants-positions-query-all-4.2.0.svg</a></li><li><a href="#">use-case-position-handler.svg</a></li></ul>
Prepare	<ul style="list-style-type: none"><li>1.1.0. DFSP1 sends a Prepare Transfer request to DFSP2</li><li>1.1.1.a. Prepare Handler Consume (single message)</li><li>1.1.1.b. Prepare Handler Consume (batch messages)</li></ul>	<ul style="list-style-type: none"><li><a href="#">seq-prepare-1.1.0.svg</a></li><li><a href="#">seq-prepare-1.1.1.a.svg</a></li><li><a href="#">seq-prepare-1.1.1.b.svg</a></li></ul>

# Phase-2: Sequence Diagrams - 3

Topic	Sequence Diagram Name	Filename
Transfer	<ul style="list-style-type: none"> <li>• 1.1.3.a. Transfer Handler Consume (single message)</li> <li>• 1.1.3.b. Transfer Handler Consume (batch messages)</li> <li>• 2.1.3. Transfer Handler Consume (Success)</li> <li>• 2.2.3. Transfer Handler Consume (Reject)</li> <li>• 2.3.0. Transfer Timeout</li> <li>• 2.3.1. Timeout Handler Consume</li> <li>• 2.3.3. Transfer Handler Consume (Timeout)</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">seq-prepare-1.1.3.a.svg</a></li> <li>• <a href="#">seq-prepare-1.1.3.b.svg</a></li> <li>• <a href="#">seq-fulfil-2.1.3.svg</a></li> <li>• <a href="#">seq-reject-2.2.3.svg</a></li> <li>• <a href="#">seq-timeout-2.3.0.svg</a></li> <li>• <a href="#">seq-timeout-2.3.1.svg</a></li> <li>• <a href="#">seq-timeout-2.3.3.svg</a></li> </ul>
Settlement	<ul style="list-style-type: none"> <li>• 6.1.1. Request Settlement Window By Id (getSettlementWindowById)</li> <li>• 6.1.2. Close Settlement Window (closeSettlementWindow)</li> <li>• 6.1.3. Get Settlement Windows By Parameters (getSettlementWindowsByParams)</li> <li>• 6.2.1. Trigger Settlement Event (createSettlement)</li> <li>• 6.2.2. Query Settlements by Parameters</li> <li>• 6.2.3. Get Settlement By Settlement, Participant and Account (getSettlementBySettlementParticipantAccount)</li> <li>• 6.2.4. Get Settlement By Id (getSettlementById)</li> <li>• 6.2.5. Acknowledgement of Settlement Transfer (updateSettlementById)</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">seq-setwindow-6.1.1.svg</a></li> <li>• <a href="#">seq-setwindow-6.1.2.svg</a></li> <li>• <a href="#">seq-setwindow-6.1.3.svg</a></li> <li>• <a href="#">seq-settlement-6.2.1.svg</a></li> <li>• <a href="#">seq-settlement-6.2.2.svg</a></li> <li>• <a href="#">seq-settlement-6.2.3.svg</a></li> <li>• <a href="#">seq-settlement-6.2.4.svg</a></li> <li>• <a href="#">seq-settlement-6.2.5.svg</a></li> </ul>

# 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.

