mojaloop

Mifos Mojaloop Payment Hub Update

April 20, 2020 - Virtual Zanzibar, Tanzania

Ed Cable | Istvan Molnar | Kristof Jozsa | Zoltan Nebli | Zoltan Mezei | Adam Saghy

mojaloop

Agenda

- Mifos Overview & Vision
- Payment Hub Grant Objectives
- Accelerating Community Adoption
- PI-9 Update
  - Implementing Production-Ready Payment Hub EE
    - Payment flows
    - Auditing
    - Operations Applications
- Roadmap for Payment Hub EE

Mifos Overview

Who is Mifos?

FinTech non-profit leveraging the cloud, mobile, and open source community to transform the delivery of digital financial services to the world's 3 billion underbanked and unbanked.

Mifos Initiative & DPC

- 501(c)3 non-profit guiding Mifos OS community advancing Apache Fineract
- Stewards of roadmap & collaborative center
- Industry thought leader and HFOSS pioneer
- Maintain Ecosystem of Solutions & Network of Partners
- 12 million clients reached across 350 orgs

- 20+ years in IT training, consultancy, software development
- Experience with Instant Payment Systems (Singapore FAST, Hungary HCT Inst, SEPA Instant)
  - Including clearing house solutions, payment hubs, shadow balance solutions from key vendors
  - Developing central clearing house prototype, simulator for participants, payment hub

dpc consulting

Edward Cable

Mifos Initiative

FINANCIAL INNOVATION AWARDS 2019 — Highly Commended — Best financial inclusion or outreach initiative — Presented by 140 The London Institute of Banking & Finance

István Molnár

Kristóf Józsa

Ádám Sághy

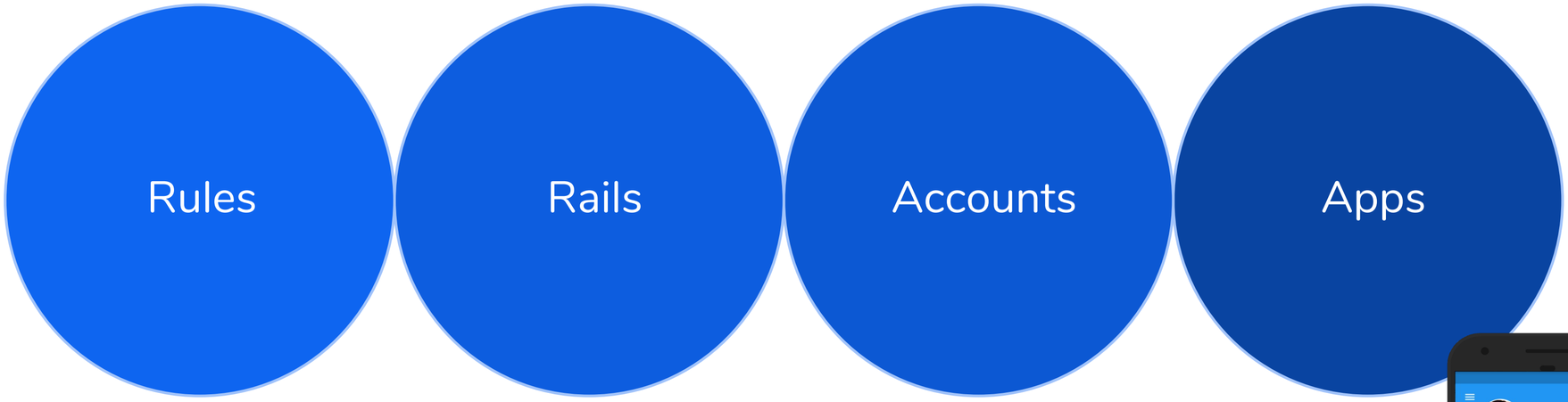
Zoltán Nébli

Zoltán Mezei

Our Vision

End to End Open Source Stack for Digital Financial Services

- Open Stack
  - OS L1-Aligned Payment Switch - Mojaloop
  - OS Bridge - Payment Hub
  - OS Account Management System - Mifos/Fineract
  - OS Reference Mobile Apps - Mobile Banking, Mobile Wallet



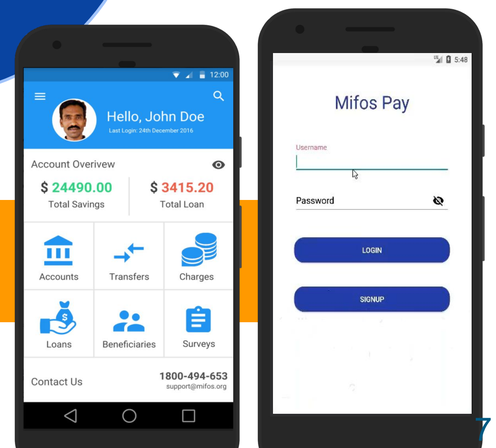
THE LEVEL ONE PROJECT

mojaloop

Payment Hub

Mifos®

Open Banking APIs



Four Layers of APIs at 2 Different Levels

Rules

Rails

Accounts

Apps

THE LEVEL ONE PROJECT

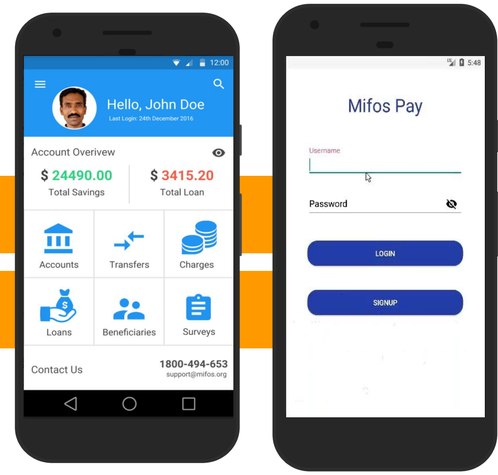
mojaloop

Payment Hub

Mifos®

Open Banking APIs

GSMA Mobile Money APIs



Infrastructure

Application

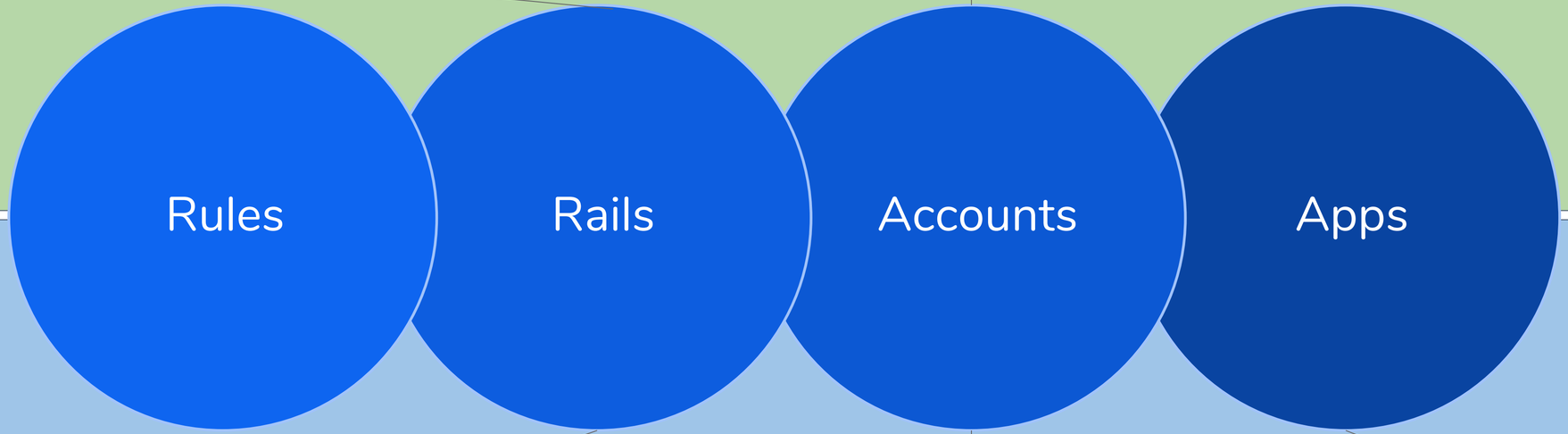
Enabling Access & Meaningful Usage of DFS

MFIs can digitize and digitally transform.

Payment Hub allows simple and low-cost participation in scheme.

Mifos X provides flexible, open production-ready system to digitize all providers, formal & informal.

- Directly offer digital financial services
- Use Open Banking API to partner with fintechs



Payment Hub facilitates easy connection to Mojaloop

Mobile Wallet Management System with Core Banking Built-in

MMOs can easily roll out adjacent loan & savings services

MMOs can evolve to become payment & service platforms

Moving into Production

An extensible, enterprise-grade integration with Mojaloop

Accelerate DFSP Mojaloop Participation in Tanzania, Ethiopia, Myanmar, & BCEAO

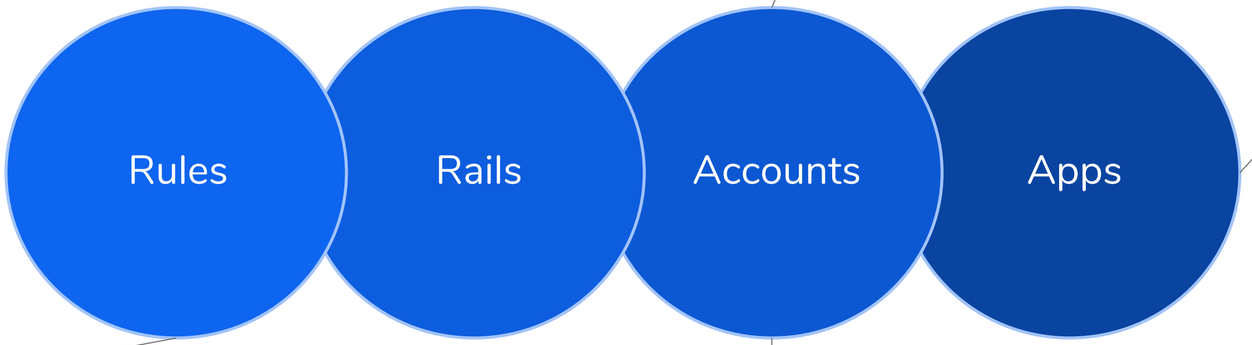
BCEAO
BANQUE CENTRALE DES ETATS DE L'AFRIQUE DE L'OUEST

MODUSBOX

የኢትዮጵያ ብሔራዊ ባንክ
NATIONAL BANK of ETHIOPIA

Payment Hub EE provides integration solution for banks to to connect to their CBS.

Open Banking API Layer Allows Third Parties to access SACCO & MFI members



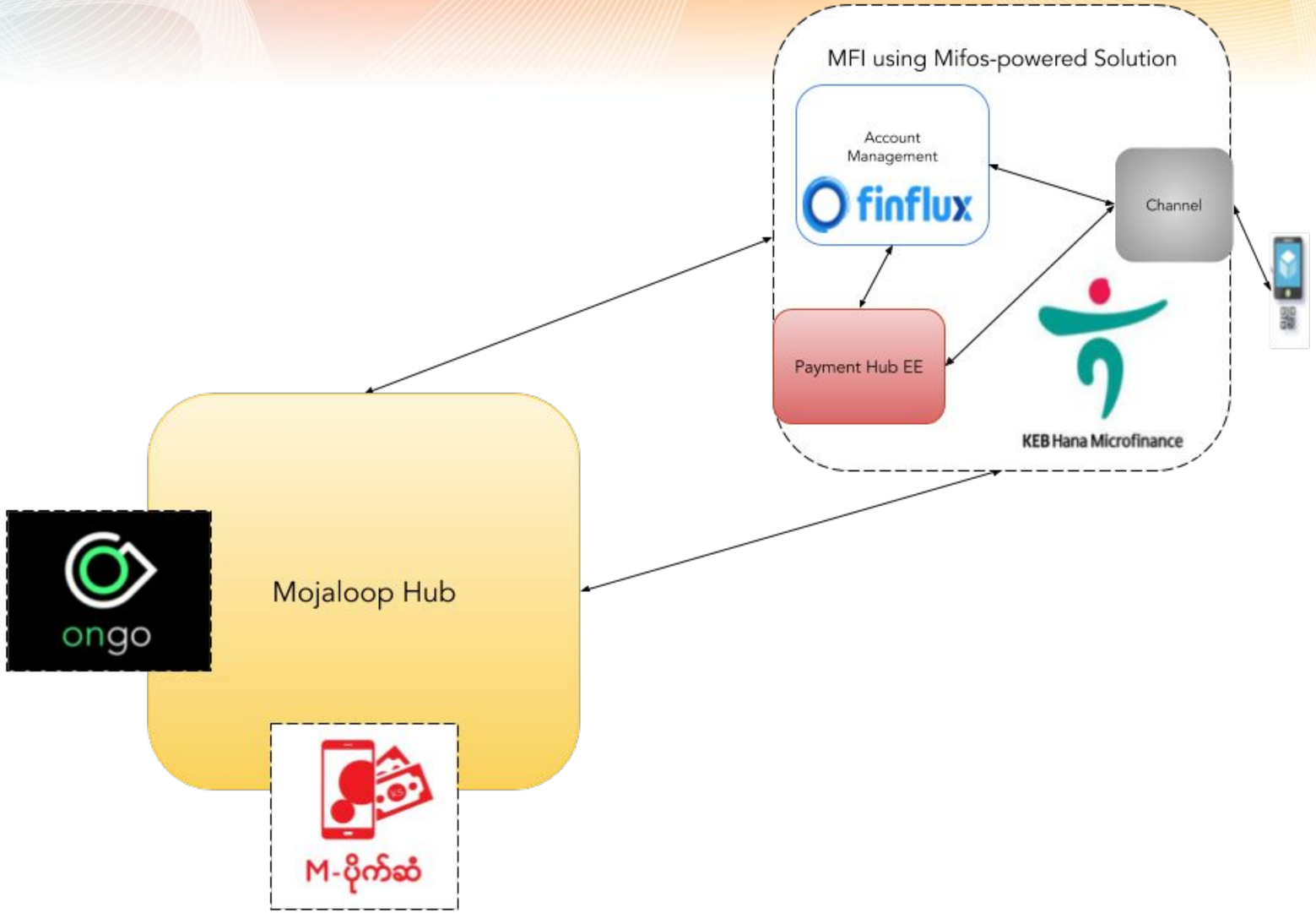
In coordination with ModusBox, FSDT, and local associations advocate for participation of MFIs and SACCOs in Mojaloop and L1P systems.

Build Connector in Payment Hub EE to integrate via Mojaloop Adapter.

Provide turnkey cloud-hosted core banking solution of Mifos integrated with Mojaloop along with Deployment Playbook

Network of local Mifos partners to deploy CBS and integrate with Mojaloop

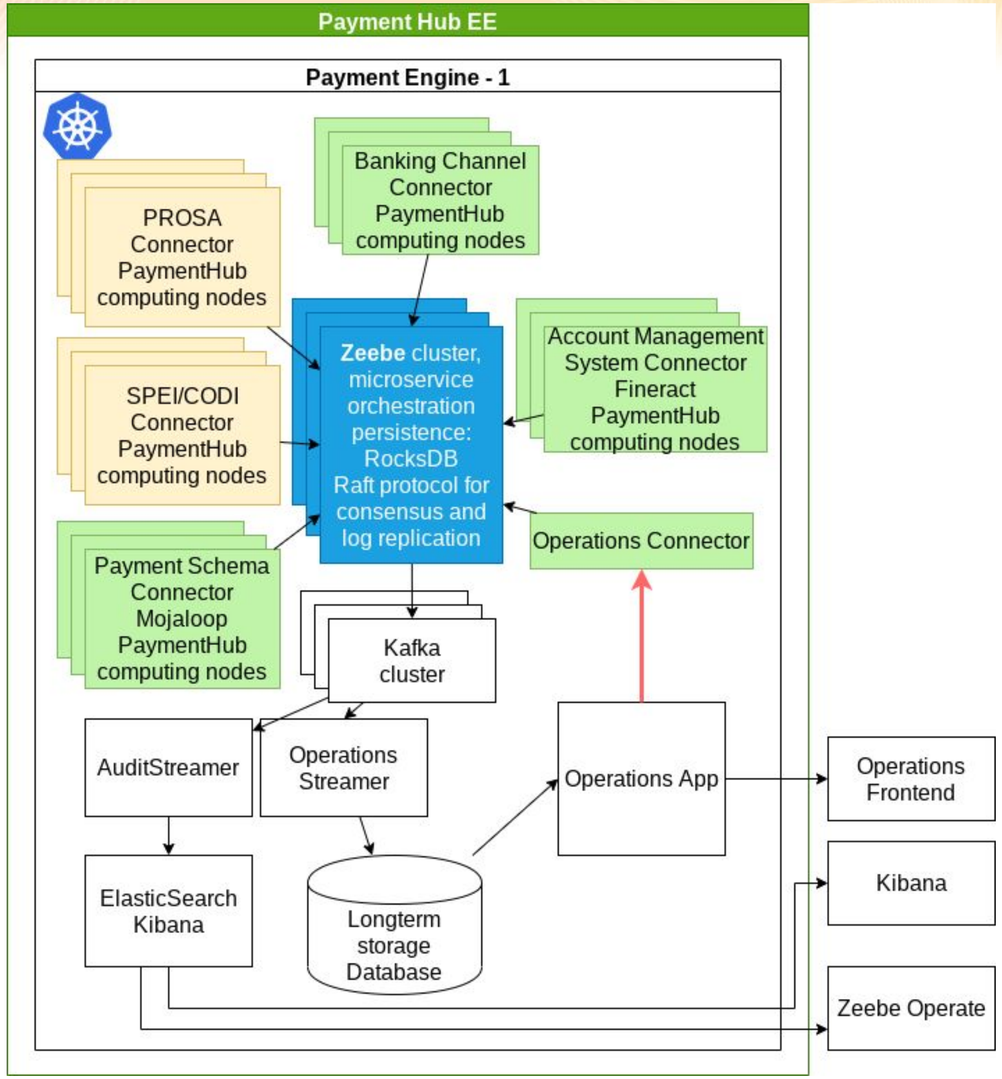
UNCDF Integration Pilot in Myanmar



Within the Mifos Ecosystem

Mexico

- Financial Inclusion Initiative led by Federal Public Policies based on open source where Mifos I/O (Gen 3) will be at the heart of providing accounts to tens of millions.
  - Phase 1 - Payment Hub EE to connect to SPEI and Codi
    - Disburse of COVID-19 stimulus loans via SPEI
  - Future - Mojaloop to enable efficiencies across payment systems and connect SOCIOs to Codi



An Immersive User Experience

Robust & Accessible Lab Environment

Assist in Credit Union Digital FI Project in Philippines & Indonesia

MODUSBOX

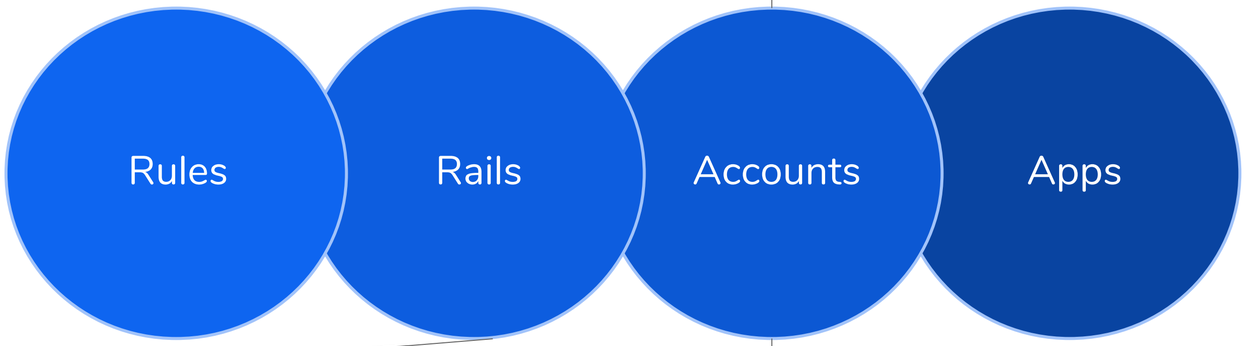
PAYSYS

Rich, working system integrated with Mojaloop for pilots/lab to test use cases

Payment Hub Architecture for CUs to integrate with existing systems.

End to End Production Ready Solution for Credit Unions who need System

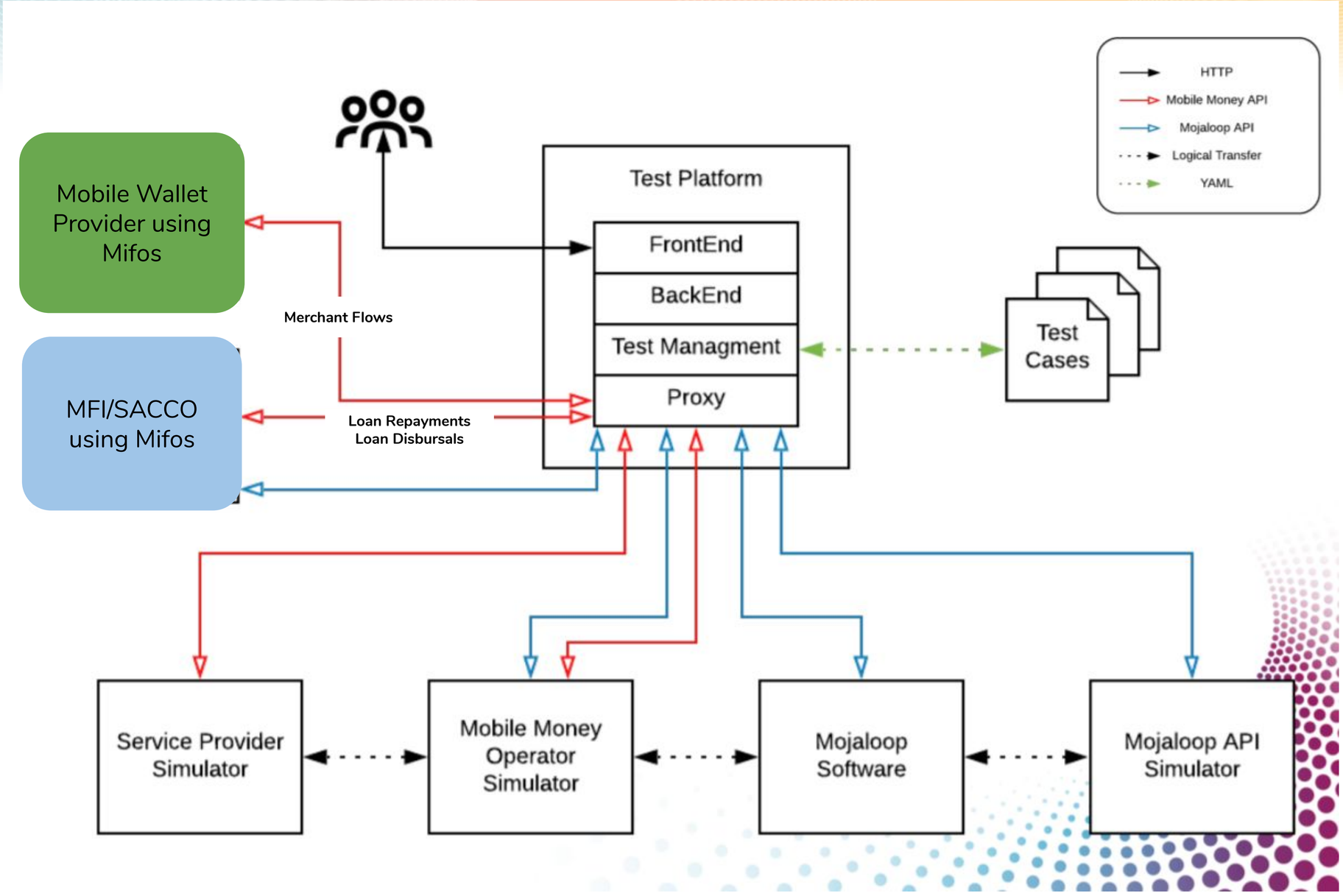
Existing Mifos Core Banking System supports credit union needs



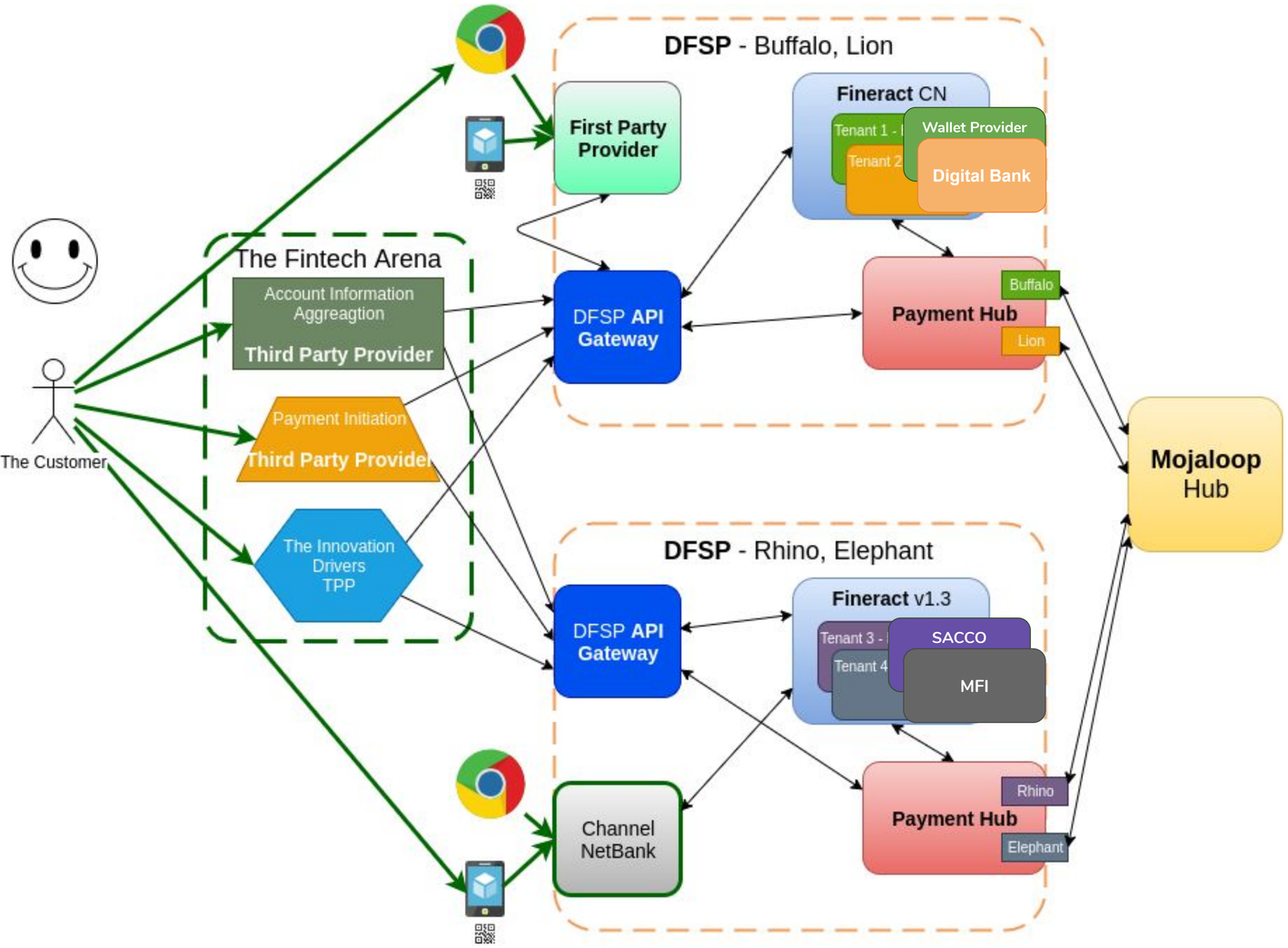
Understand needs and uses for credit unions to prioritize Mojaloop APIs to support in Payment Hub EE

Leverage on the ground network of local Mifos partners to help with future deployment and rollout.

Integrating with GSMA Interoperability Test Platform



Sandbox Environment for Hackathons



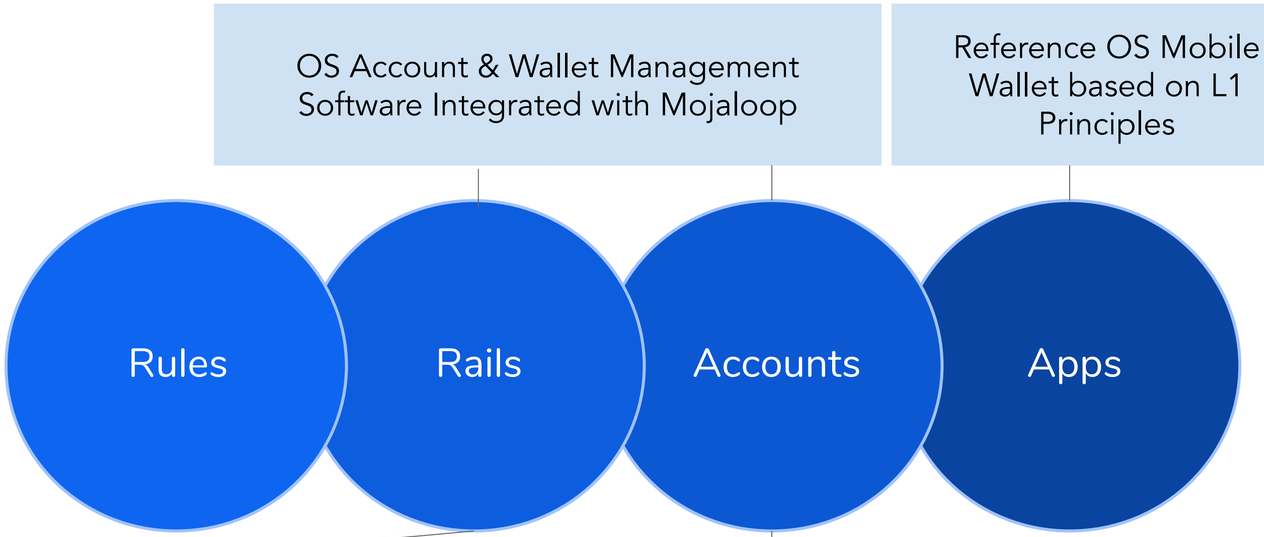
Audience	API Layer	OS Toolset
Fintechs	**- Open Banking API Layer** **- GSMA Mobile Money API (in progress)**	➢ Mobile Wallet App ➢ Mobile Banking App ➢ Online Banking App ➢ Open Banking Fintech App
DFSPs Banks, MFIs, SACCO, MMOS	**Mifos/Fineract API Layer** - Identity & KYC - Wallet/Account Management - Loan & Savings Management - Accounting & Ledger - Reporting	➢ API-Driven Open Source Core Banking Platform ➢ Angular Web UI for staff ➢ Android Mobile UI for staff
Hub Operators, Regulators	**Mojaloop API Layer** - Peer to Peer - Merchant Proximity Payment - Merchant Request to Pay (in progress) - Bulk Payment/Transfer (upcoming)	➢ Payment Hub EE ○ Operations UI ○ BPMN Workflow Engine

Explore How Mifos Can Support Vision & Goals of GSMA Lab

Advance adoption of GSMA Mobile Money API

Enable Interoperability & Merchant Acceptance of Mobile Money

Fuel delivery of adjacent products and services on top of rails



Leverage Payment Hub to integrate with Mojaloop and GSMA Mobile Money APIs

Identify a pilot to help MFIs and SACCOs undergo digital transformation

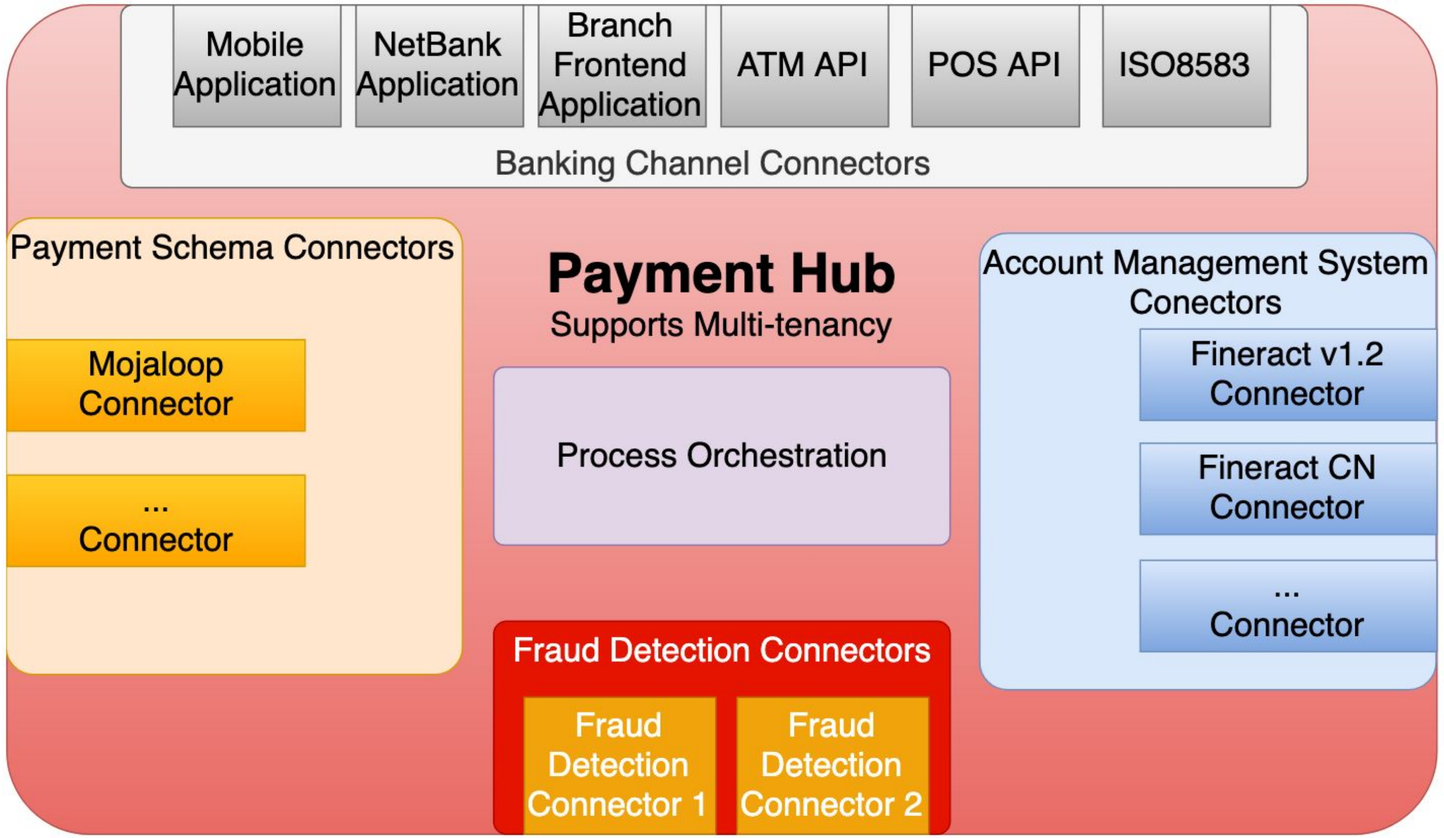
PI-9 Progress

PI-9 Accomplishments

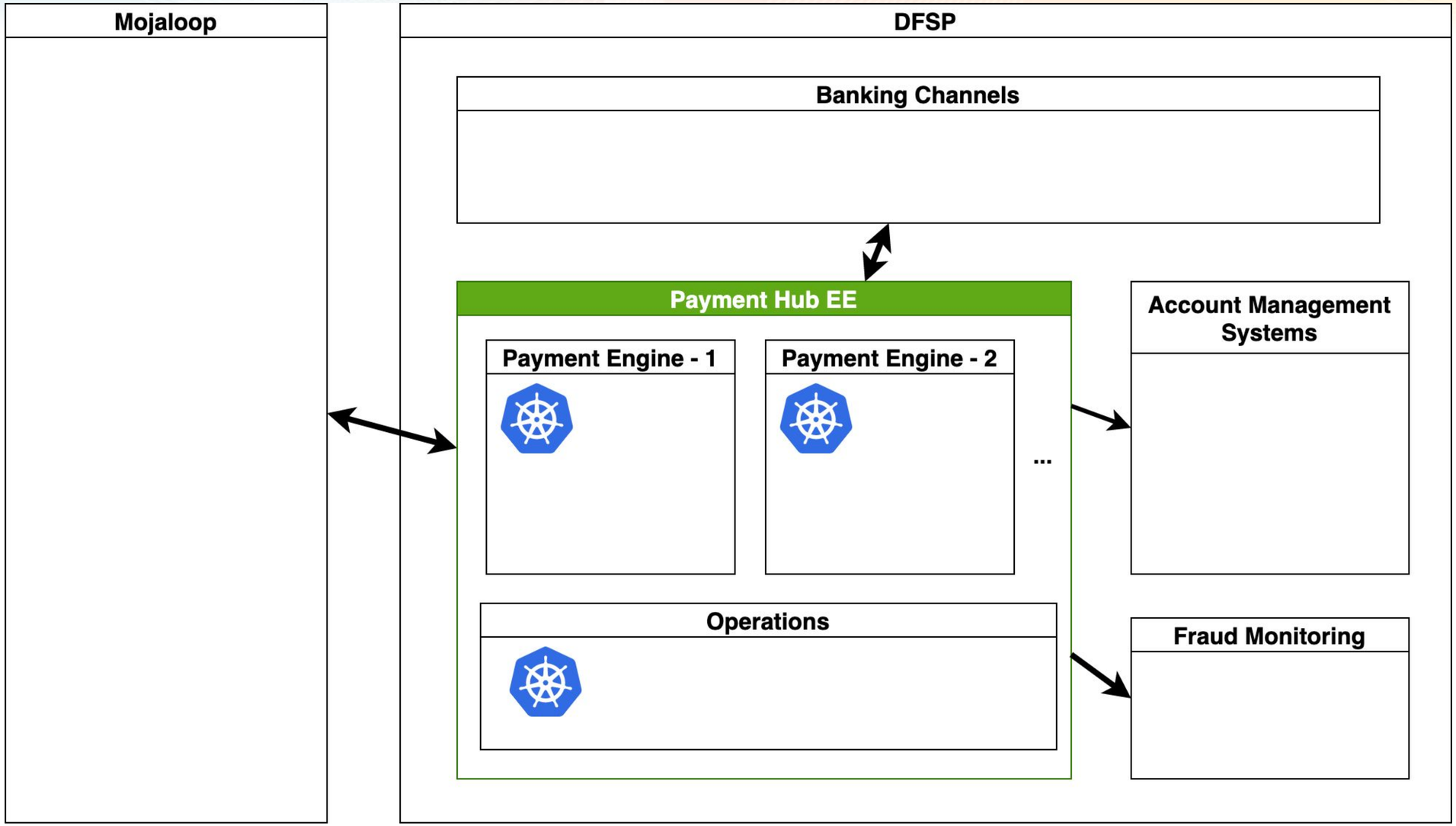
- Implementation of Production-Ready Payment Hub EE
  - Creating all Fineract and Payment instances with CI/CD on Azure
  - Payment flows are handling special cases and operations
  - Auditing the complete flow from the point of view of the DFSP
  - Operations Applications providing Search, Summary and Detailed information on particular processes
  - Error handling is sophisticated and evolving

Payment Hub as an Open Source Asset for the Community

- The role of a payment hub to connect:
  - Financial Institution channels (Mobile, Internet, Branch, Callcenter, ATM, POS, API Gateways)
  - Account Management Systems (AMS / Core banking platform), optionally fraud monitoring tools
  - Payment Schemes, such as Mojaloop
- Need
  - Consistent Way to Connect to Mojaloop
  - Effective Operational Participation
- Additional Capabilities
  - DFSP-level fraud monitoring
  - Bulk Transfer Campaign Management
  - Operational Monitoring
  - Manages the identifier – account relation
  - Trigger notifications
- Built on proven open-source technology:
  - Java, SpringBoot, Kafka, Elasticsearch
  - Apache Camel, Camunda Zeebe
  - Kubernetes

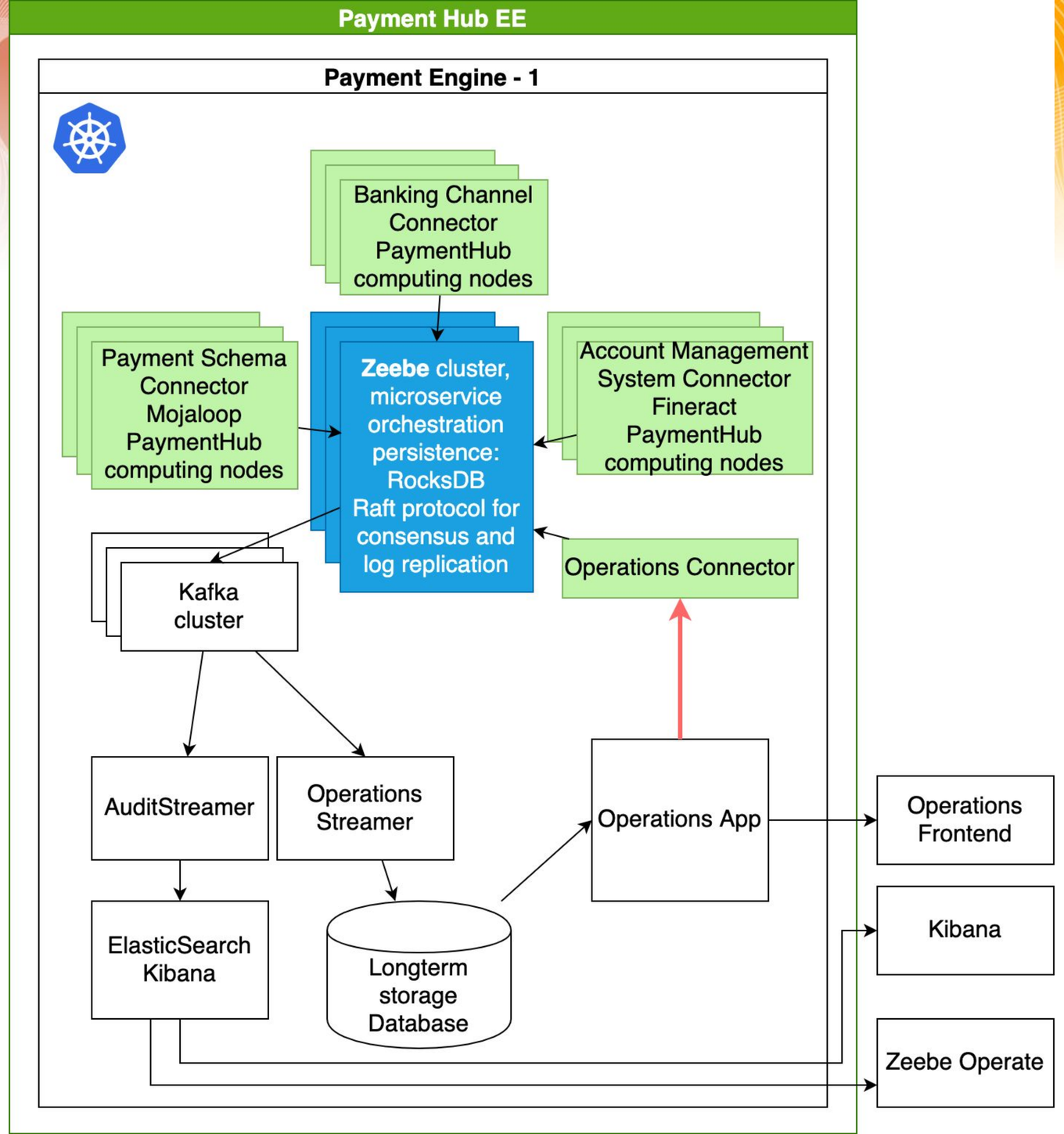


Payment Hub EE in the Payment Context

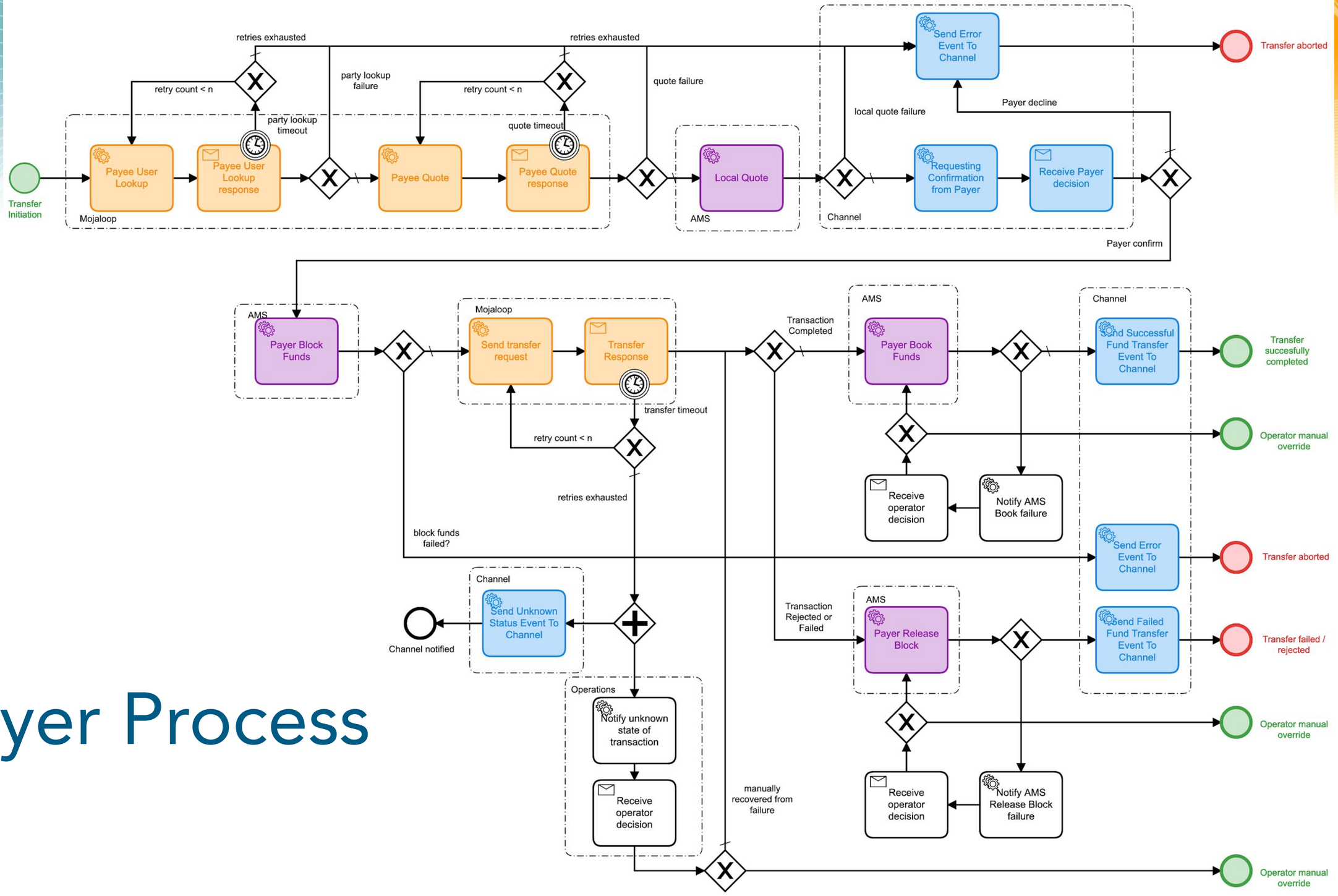


PH EE Internals

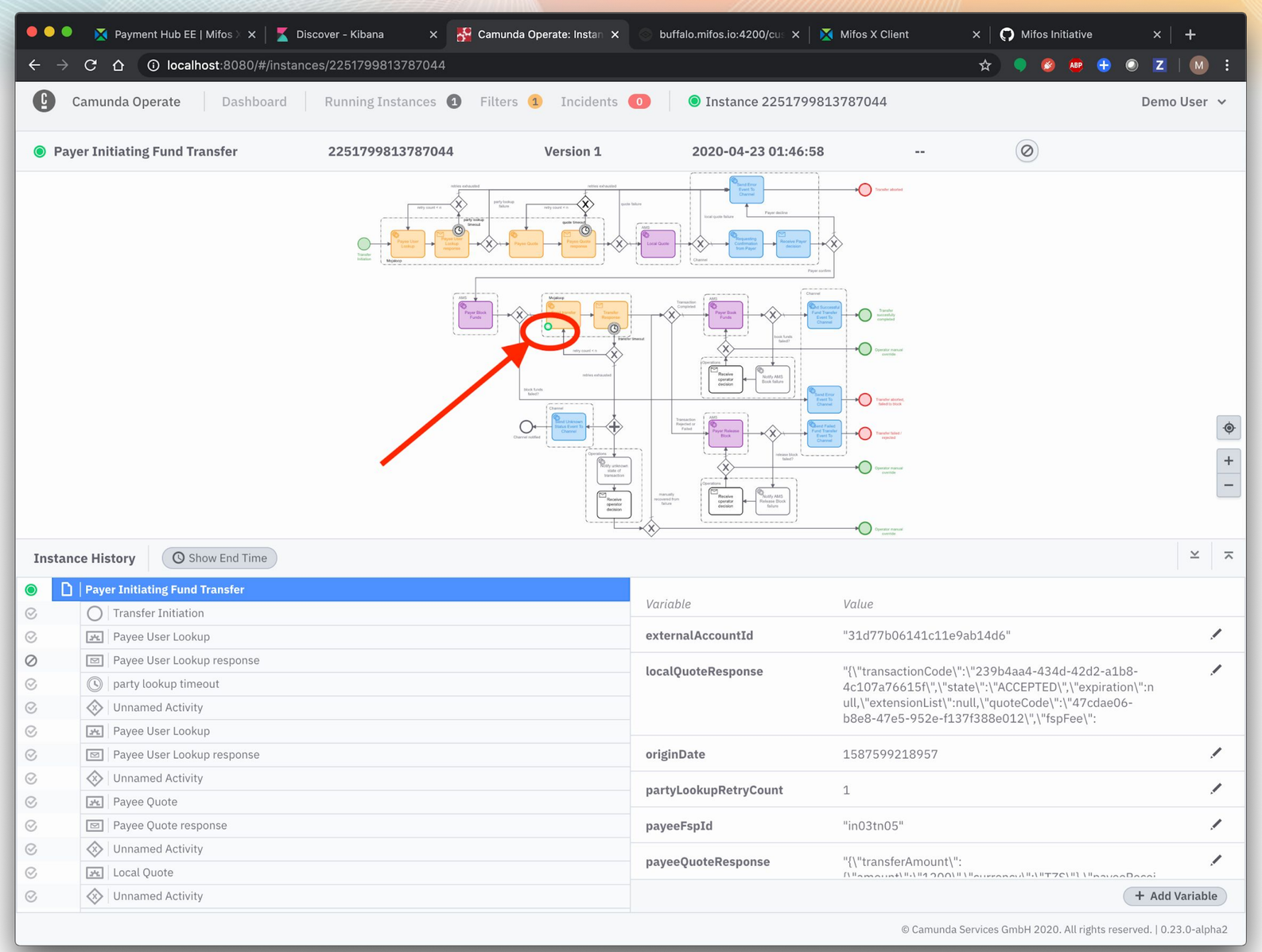
- Process orchestration by Zeebe
- Mojaloop connector
- Channel connector
- AMS connector
- Audit Streamer
- Operations Streamer
- Operations App
- Operations Frontend



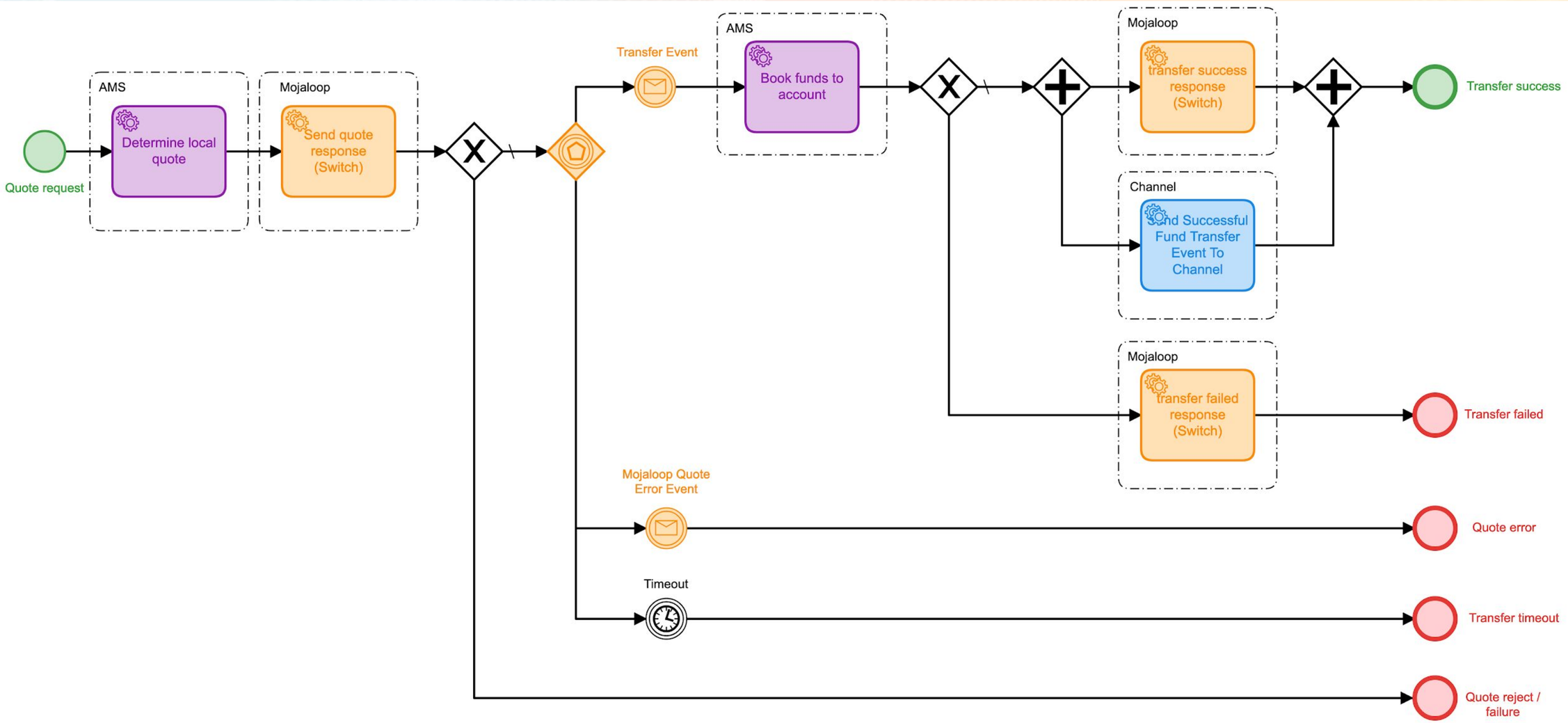
Payer Process



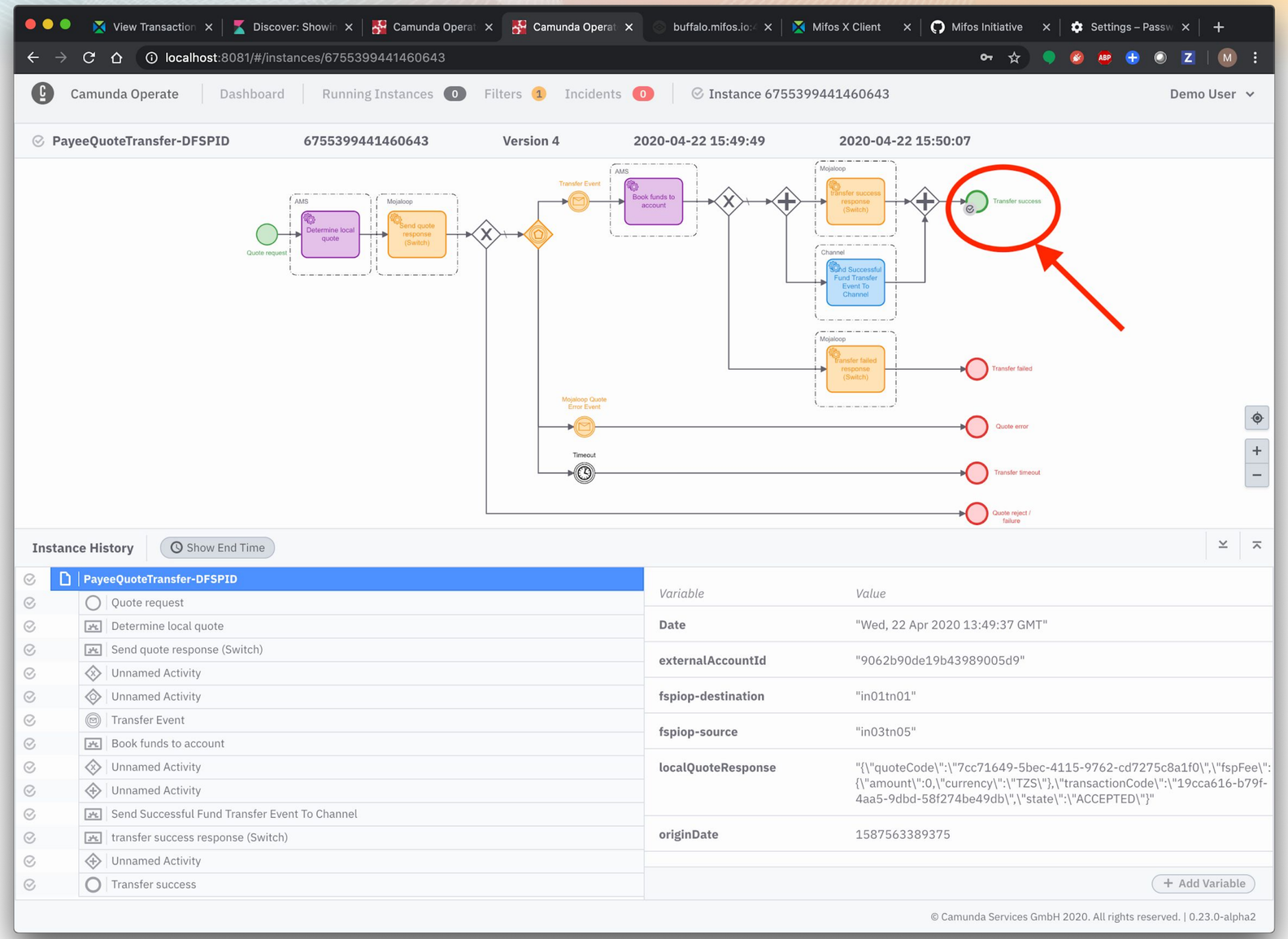
Payer Process in Production



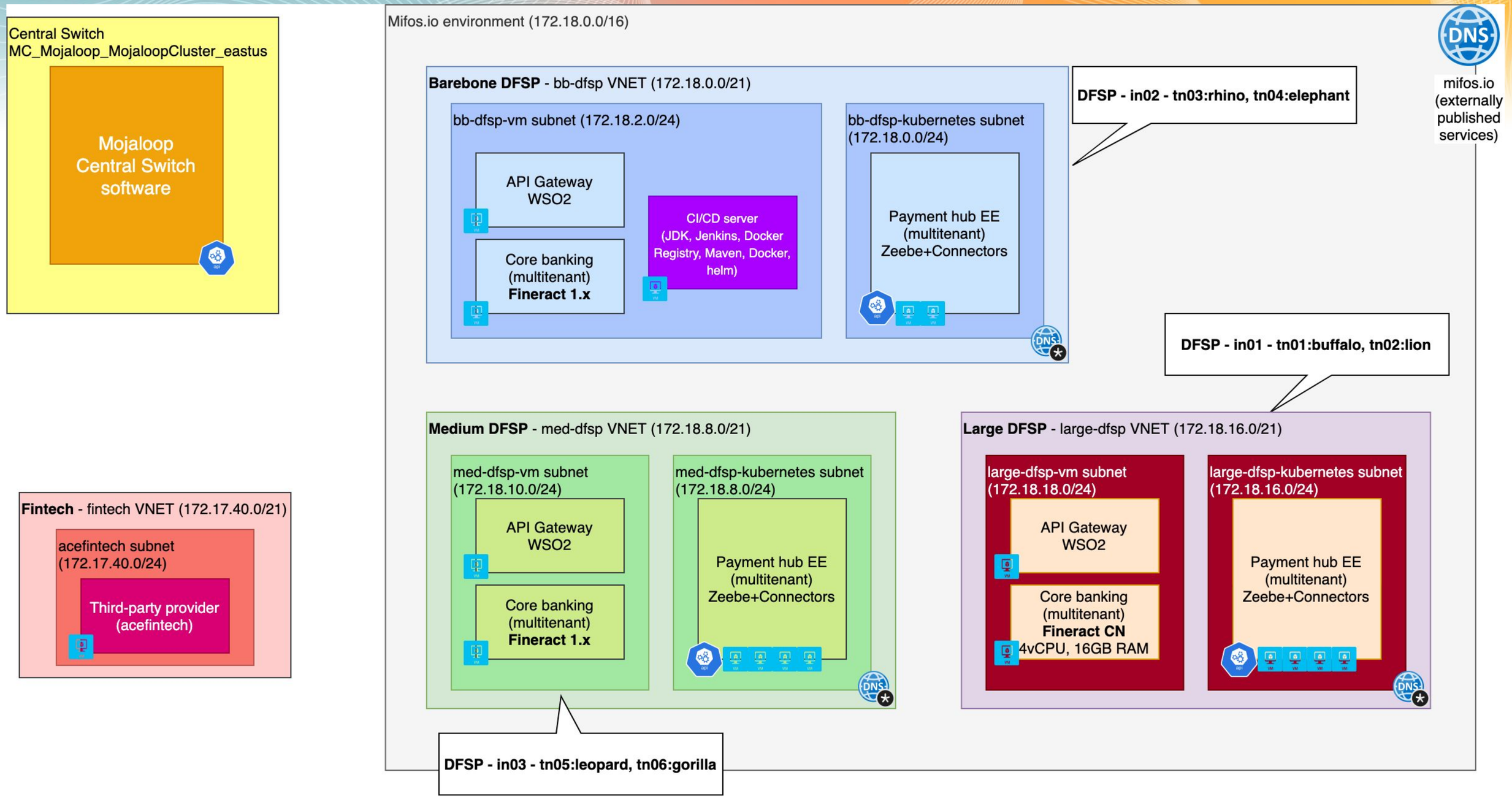
Payee Process



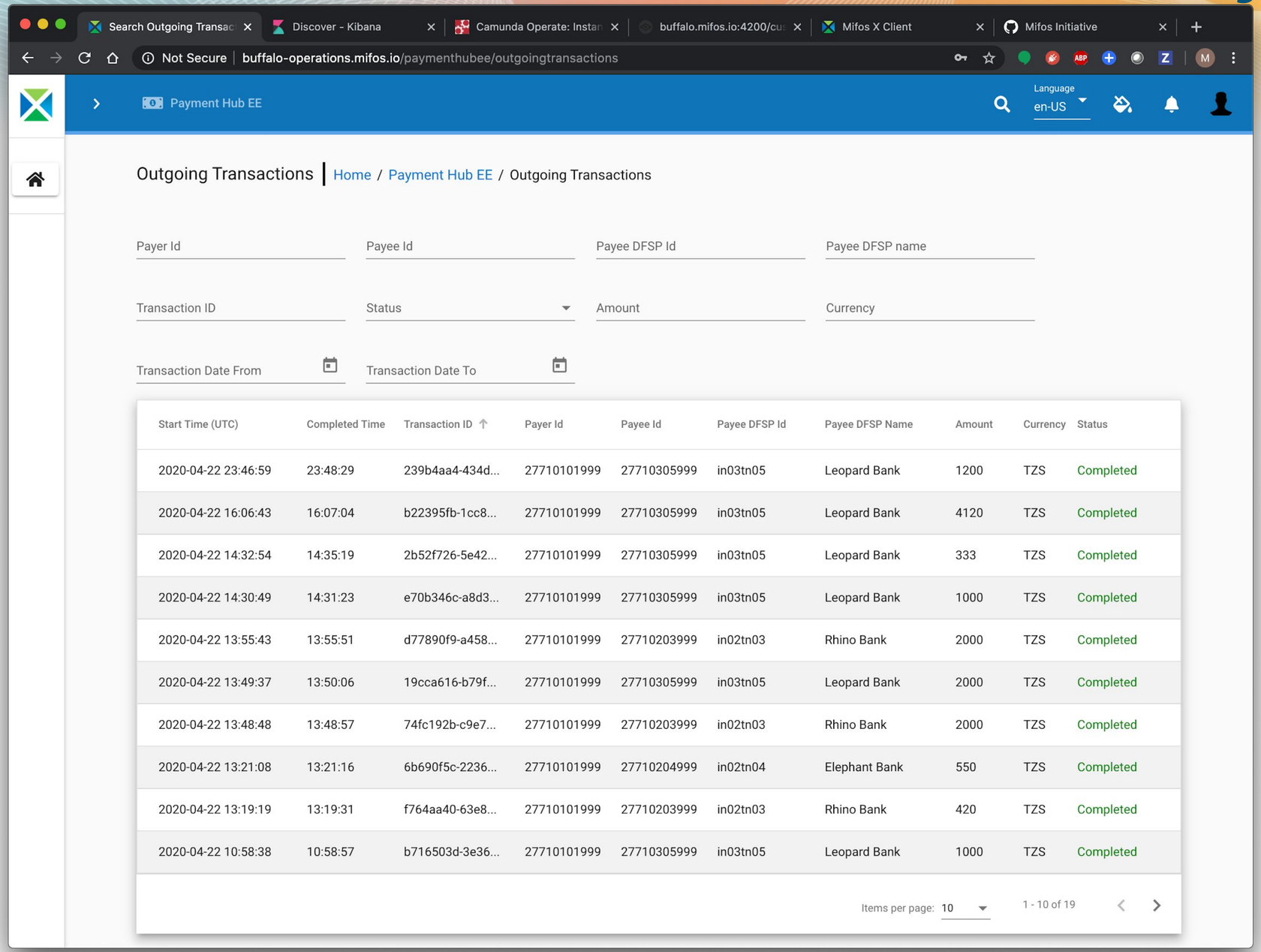
Payee Process in Production



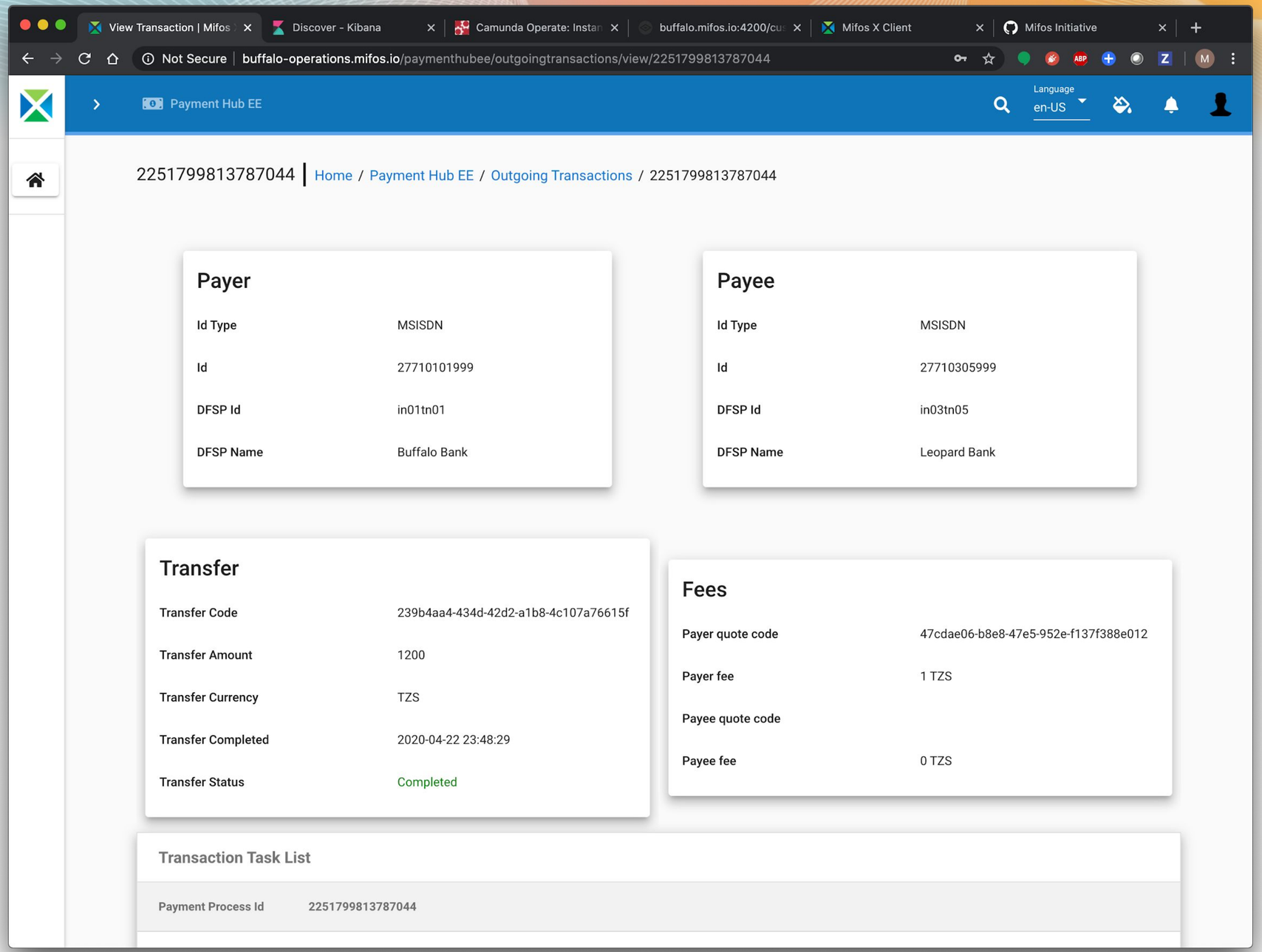
Mifos Lab - Hackathon Environment



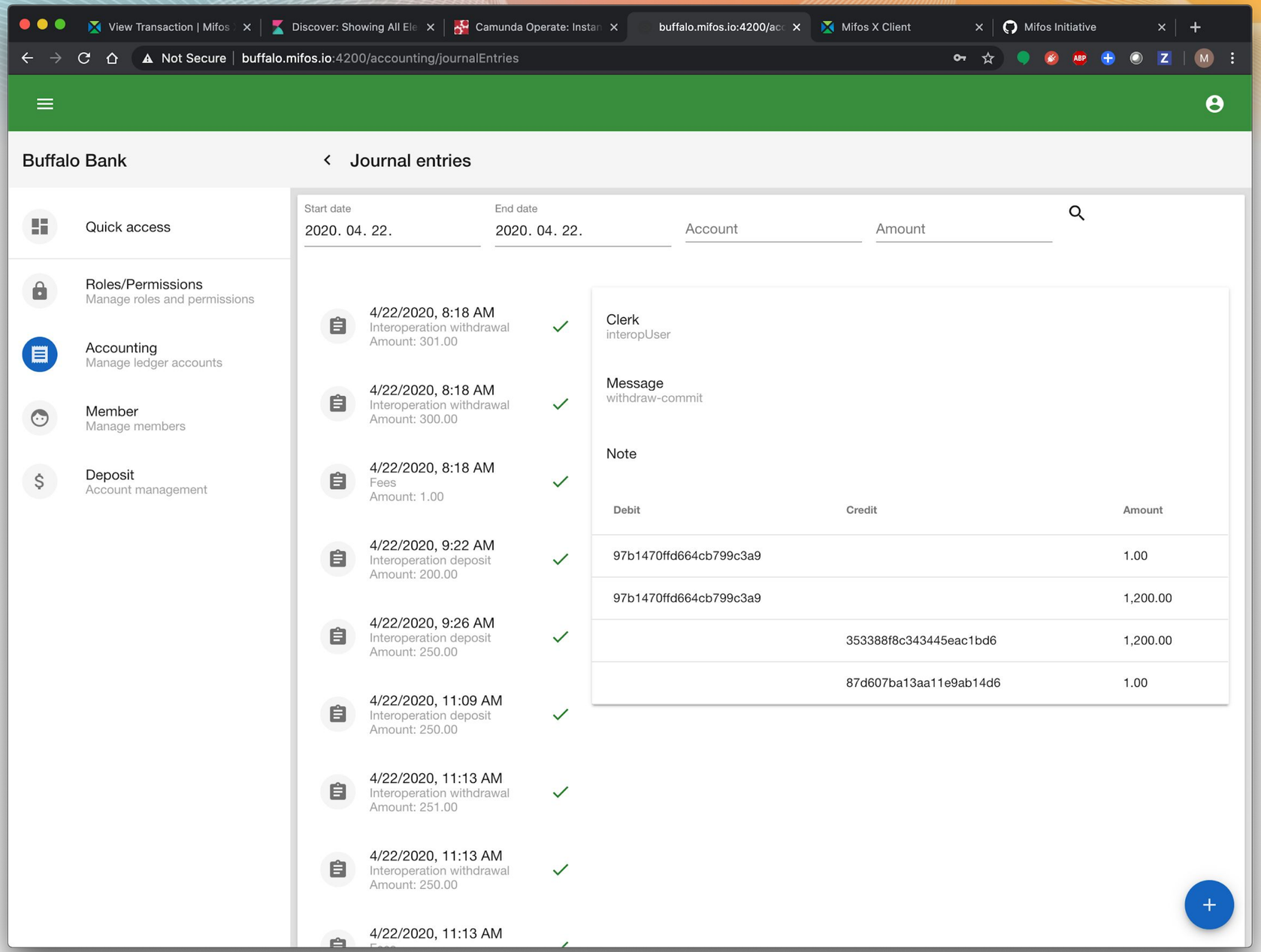
Operations User Interface - Summary



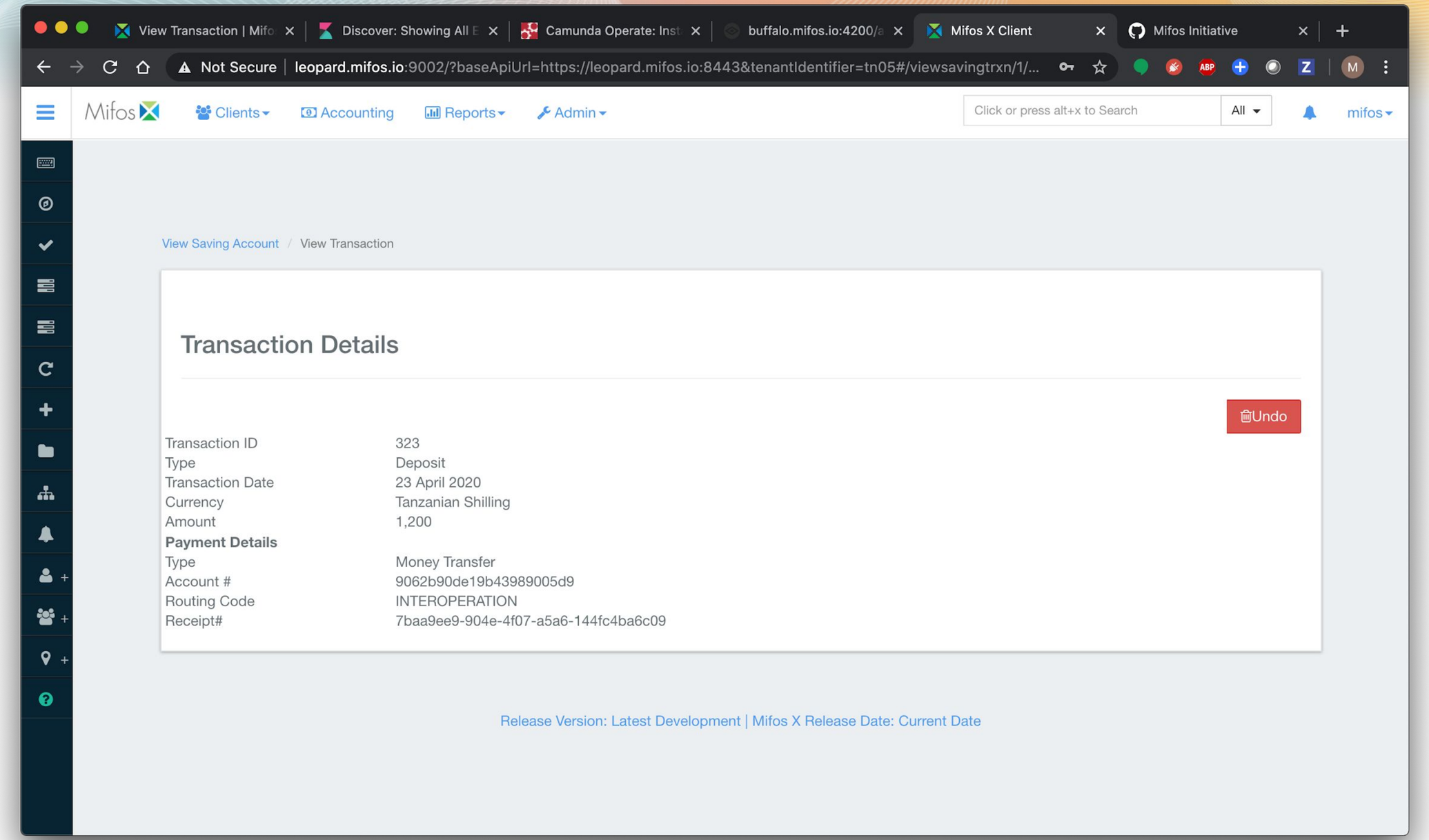
Operations User Interface - Details



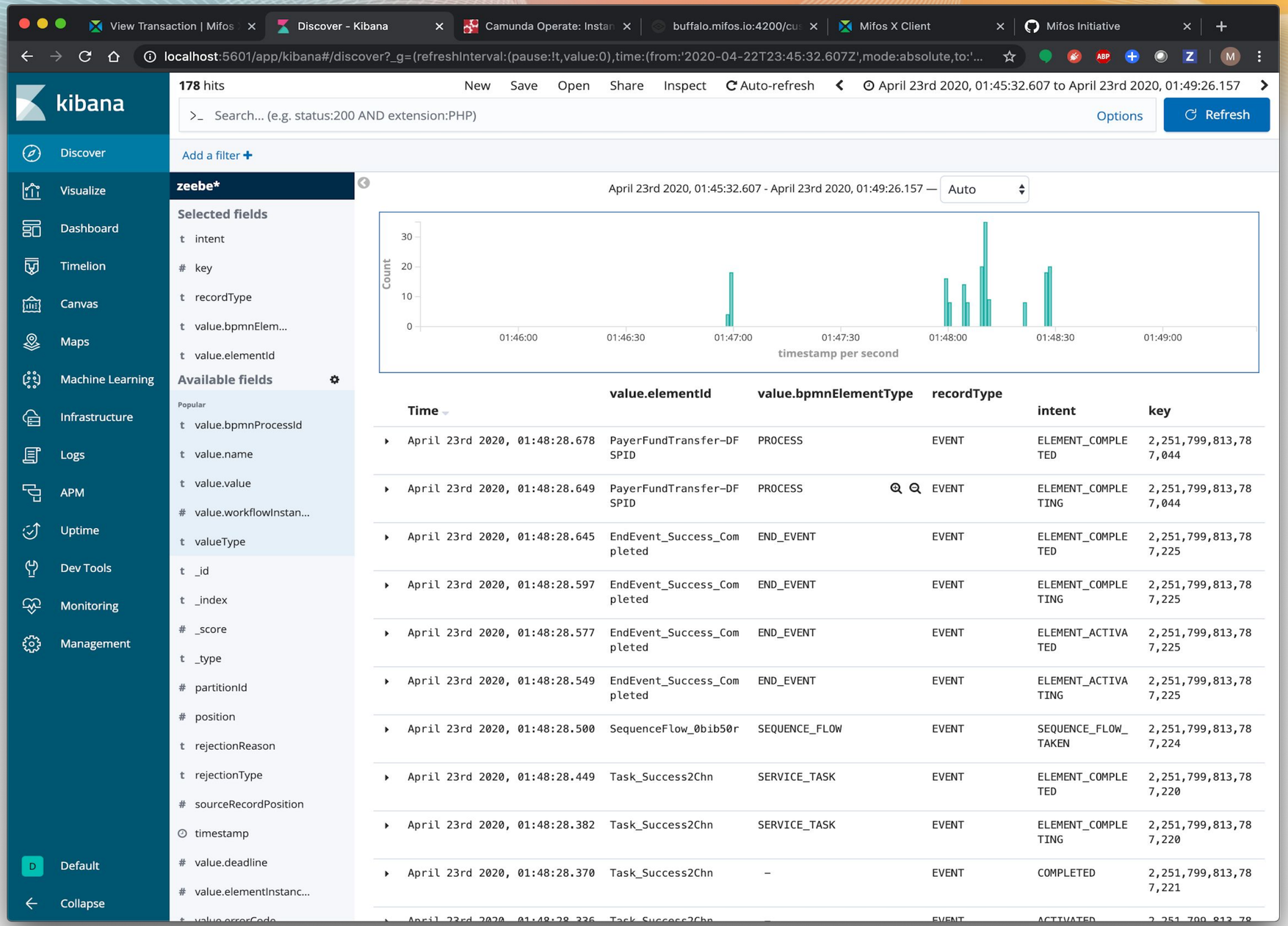
Payer's Account



Payee's Account



Audit records



Roadmap

Work in progress

- Operations actions
  - Enable DFSP operators to resolve problematic cases
    - Notifications collected on the Operations UI
    - Provide Actions for the Authorised Personnel
      - Force Complete, Force Fail, Manual Override, Retry - depending on context
      - Initiate recall of transactions
    - Selected action is fed back to the process to execute the changes
- Finalizing the request to pay flows
- Support the party id (e.g. MSISDN) registration process
- Optimize and fully integrate Mifos channel apps into lab
- Performance testing and tuning

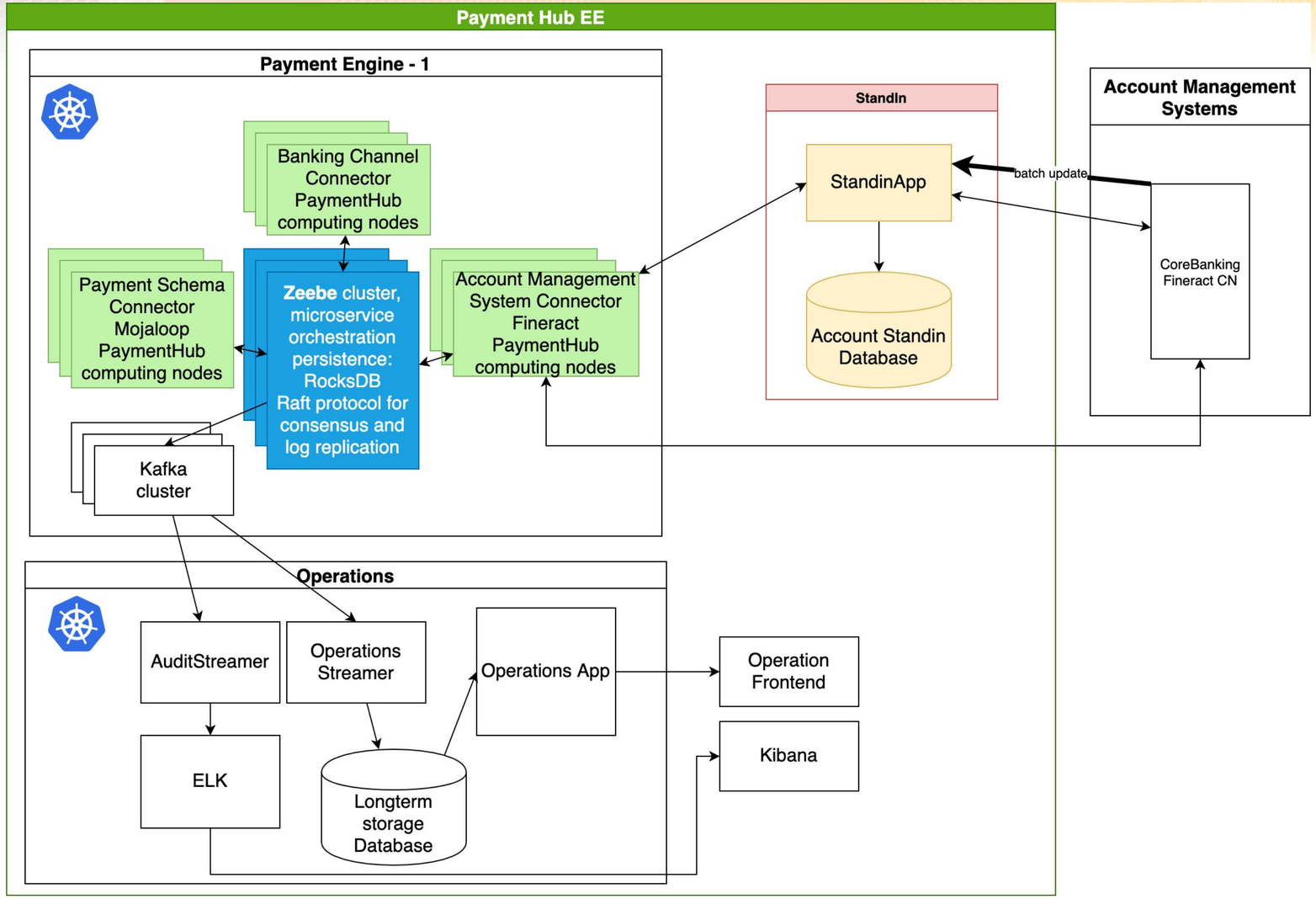
Roadmap for Payment Hub EE

- Stand-in solution
  - Provide 7x24 availability for a DFSP while core banking platform is under maintenance or not accessible while running end of day jobs
- Preprocessing bulk payments
  - Lookup Payee's DFSP ID for the transactions to determine target DFSP
  - Splitting the incoming bulk into smaller batches per target DFSP
    - On-us transactions can be handled differently
  - Manage communication with Mojaloop for the batches
  - Aggregate incoming results to provide response to the bank's channel
  - In case transfering from a single account (pension, aid), booking can be individual, aggregated by target DFSP or single grand total
- Integration with new APIs for 3rd parties (PISP APIs)
- Mojaloop Adapter Integration, Mobile Money API integration

Stand-in

Integrated into the Payment Hub EE solution a stand-in system could provide functionality in case the Account Management System is not available.

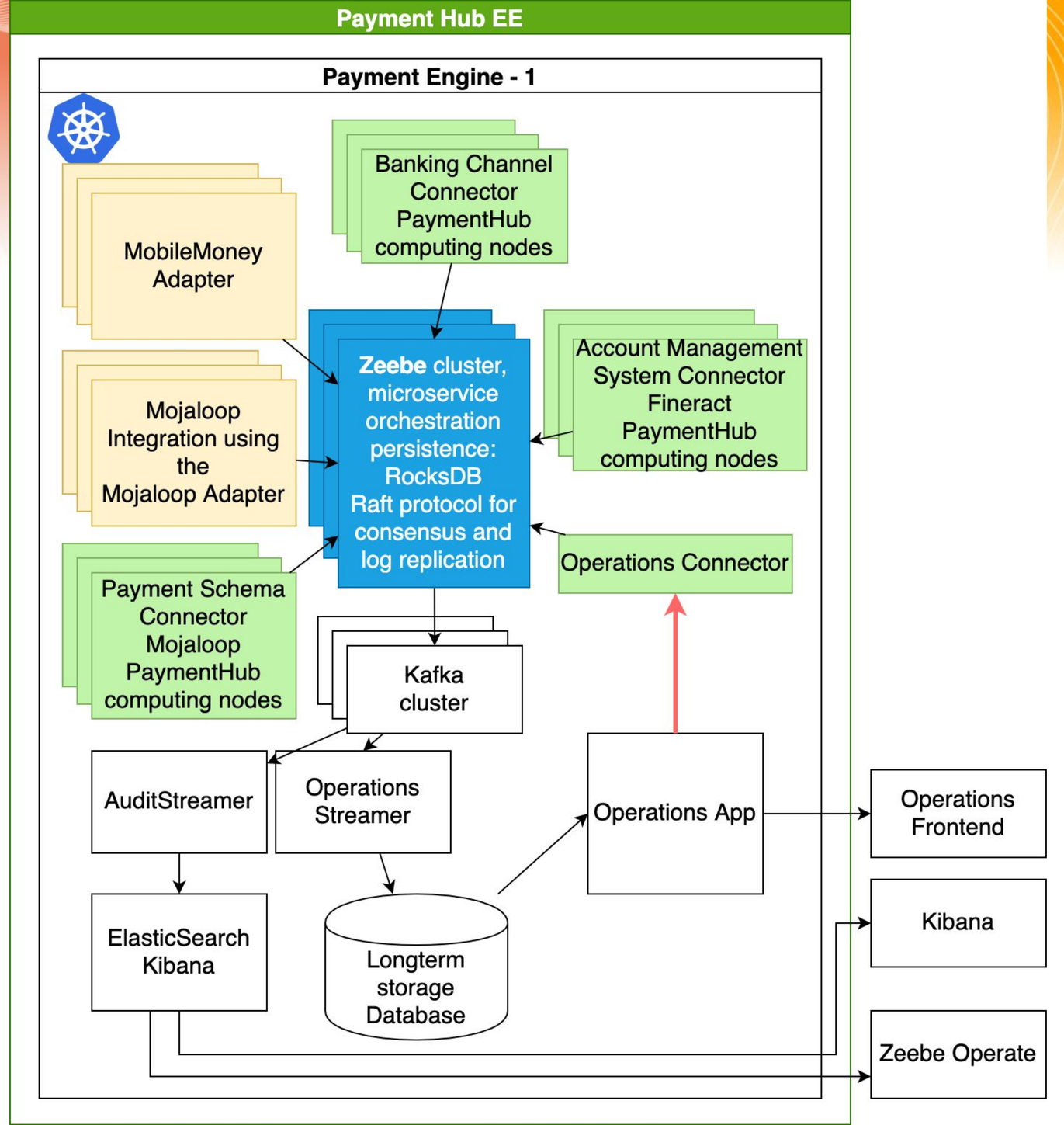
- only incoming transactions - simple solution, requires only synchronizing the valid account and party ids
- both incoming and outgoing transactions - requires more complex solution to minimize risk of overdraft



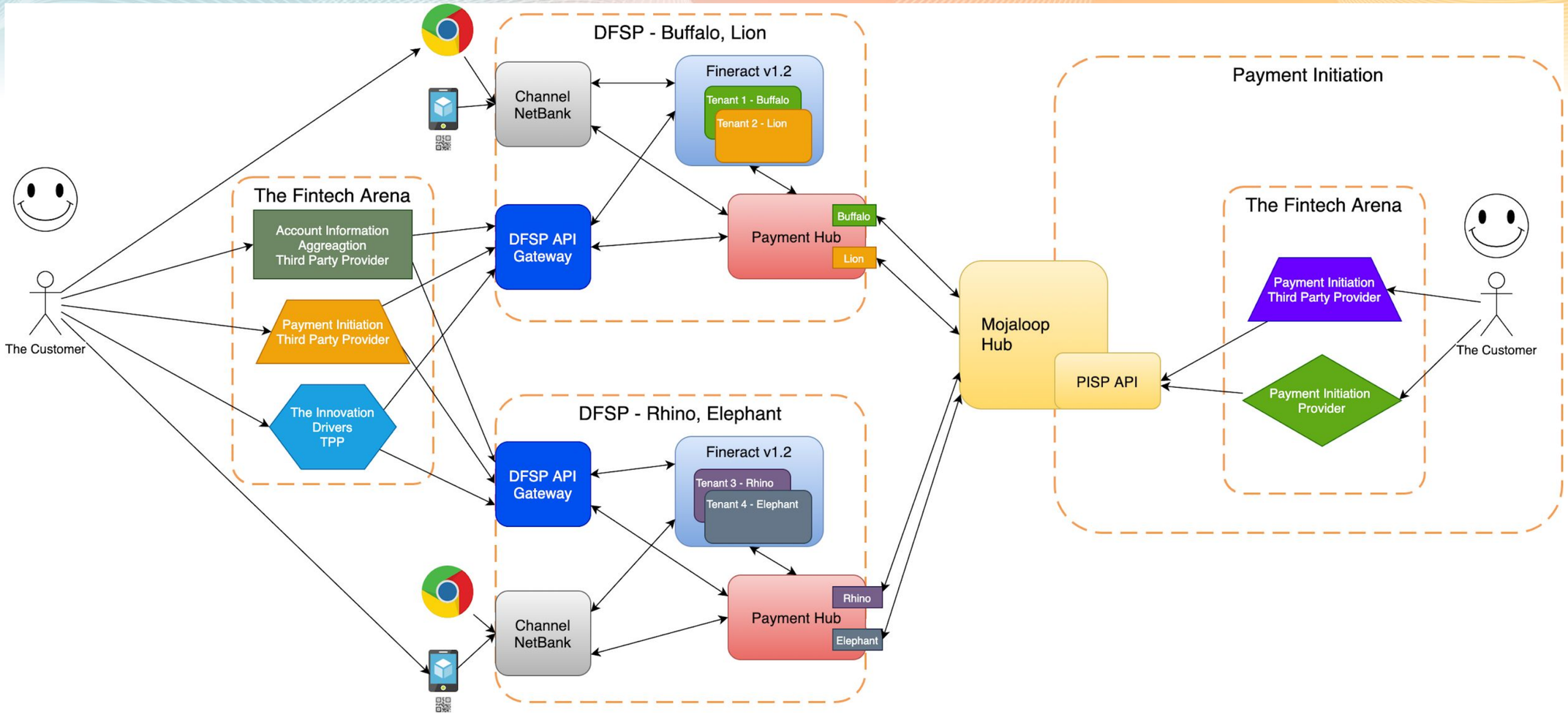
New Schema Connectors

Provide new connectors:

- Mojaloop Connector based on the Mojaloop Adapter
- Mobile Money API adapter for DFSPs who wish to manage funds at the Mobile Money providers



Supporting the PISP APIs



- Support the new APIs, so PISPs could execute the transactions

Summary

- Simplifies integration
  - robust platform to handle not only Mojaloop Integration, but the internal processes and systems of the DFSP
- Customizable platform
  - (e.g. multiple account management systems, no real-time interface for core banking, stand-in capability, fraud monitoring)
- Complete audit log of all transactions
- Full Control for DFSP Operators
  - Force Complete, Force Fail, Manual Override, Retry, Initiate recall
- Pre-process bulk transfers

Summary

The Payment Hub EE simplifies integration for DFSPs by providing a robust platform to handle not only Mojaloop Integration, but the internal processes and systems of the DFSP

Customizable platform to be easily adapted to the DFSP special requirements (e.g. multiple account management systems, no real-time interface for core banking, stand-in capability, fraud monitoring)

Complete audit log of all transactions from the DFSP point of view, and quick actions for DFSP payment operators

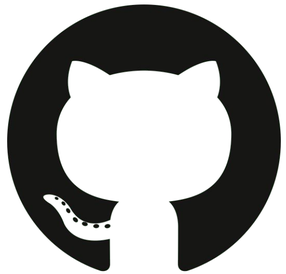
Ready to pre-process bulk transfers to support sending out huge volumes either as real-time transactions or batches

Thank You

- Miller Abel, Kim Walters & Ariel Delaney
- Core OSS Team

Edward Cable
edcable@mifos.org
https://mifos.org

Istvan Molnar
istvan.molnar@dpc.hu
https://dpc.hu



github.com/openMF
github.com/apache/fineract
https://fineract.apache.org

Appendix

Grant Objectives

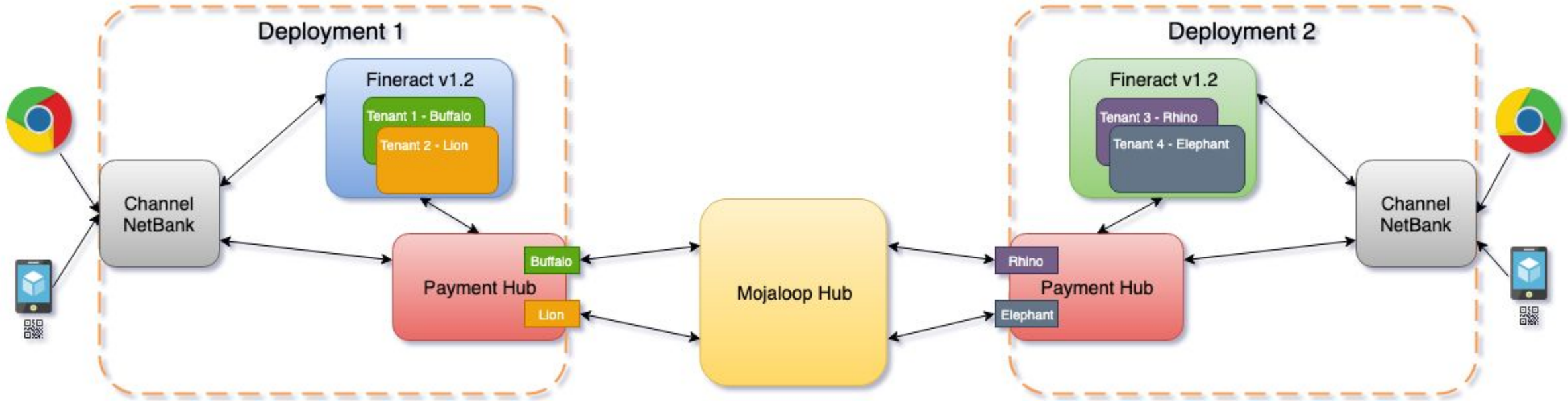
- Update Payment Hub to Incorporate Latest Mojaloop Components
- Achieve production readiness of Payment Hub to support deployments
  - Operational UI for monitoring, investigation and problem resolution at DFSP
  - **Persistence management for recoverability and auditability**
  - Identifier (MSISDN) - Account assignment management
  - Notification handling (push, sms, email)
  - **Advanced error handling, compensation management**
  - **High Availability, Fault Tolerance**
  - Optimization for High Performance
  - Implement necessary tests and QA procedures
  - Baseline documentation to set up, configure, and extend Payment Hub.

Grant Objectives cont.

- Build Support for the additional use cases provided by the Mojaloop APIs
  - Request To Pay
  - Bulk Payment
  - Agent initiated cash out; cash out authorized on POS
  - Customer initiated cash out
  - Merchant Initiated Merchant Payment; Authorized on POS
  - ATM initiated cash out
  - Refund
- Update Mifos Reference Apps to Provide Demonstrable User Interface
  - Mobile Banking, Mobile Wallet and Online Banking Apps
  - Web app for staff
- Additional Advanced Capabilities of Payment Hub
  - Bulk Transfer Campaign Management & Processing

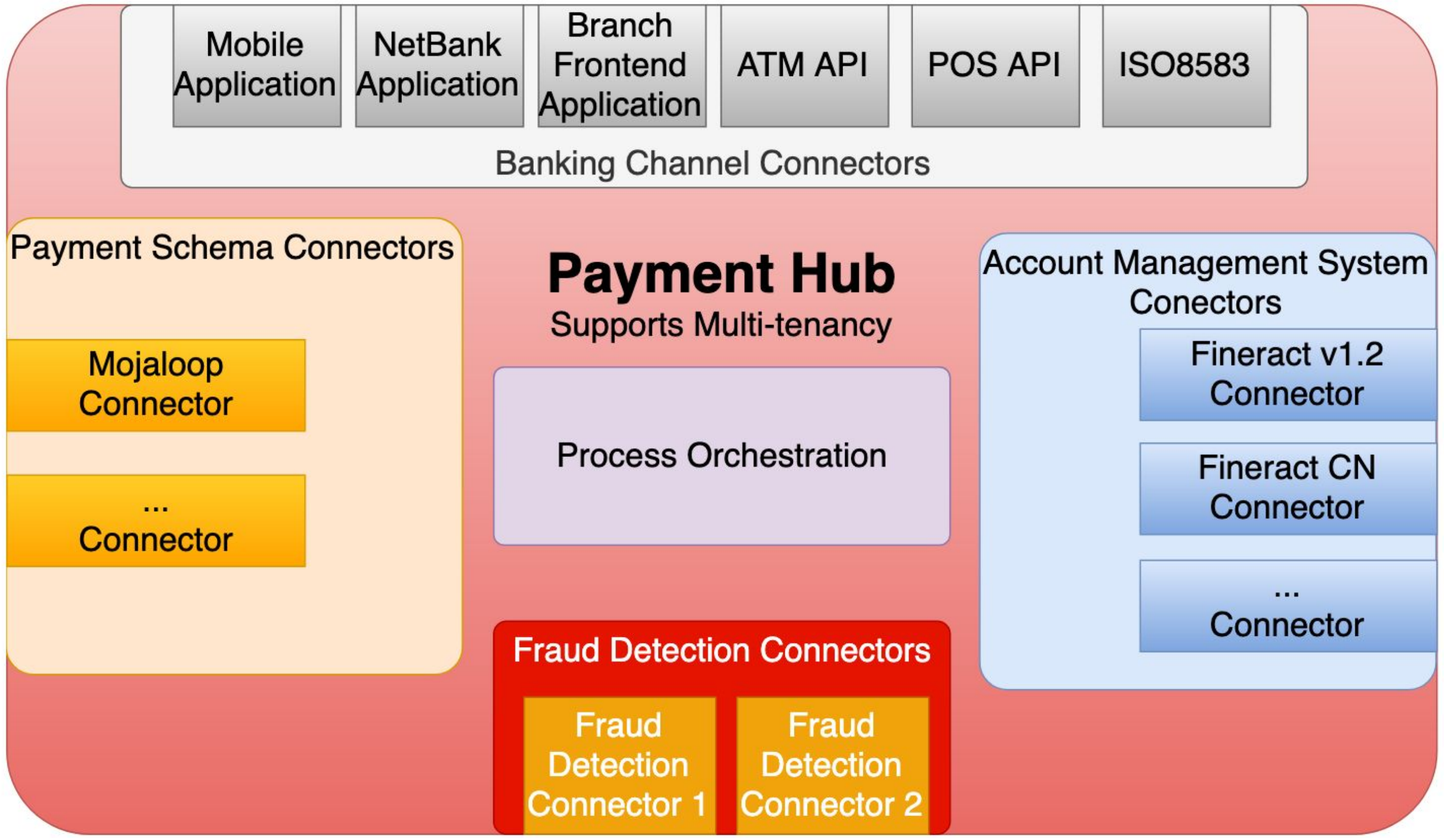
Architecture & Design for Production-Readiness - Payment Hub EE

Logical model of a test environment



Payment Hub as an Open Source Asset for the Community

- The role of a payment hub to connect:
  - Financial Institution channels (Mobile, Internet, Branch, Callcenter, ATM, POS, API Gateways)
  - Account Management Systems (AMS / Core banking platform), optionally fraud monitoring tools
  - Payment Schemes, such as Mojaloop
- Need
  - Consistent Way to Connect to Mojaloop
  - Effective Operational Participation
- Additional Capabilities
  - DFSP-level fraud monitoring
  - Bulk Transfer Campaign Management
  - Operational Monitoring
  - Manages the identifier – account relation
  - Trigger notifications
- Built on proven open-source technology:
  - Java, SpringBoot, Kafka, Elasticsearch
  - Apache Camel, Camunda Zeebe
  - Kubernetes



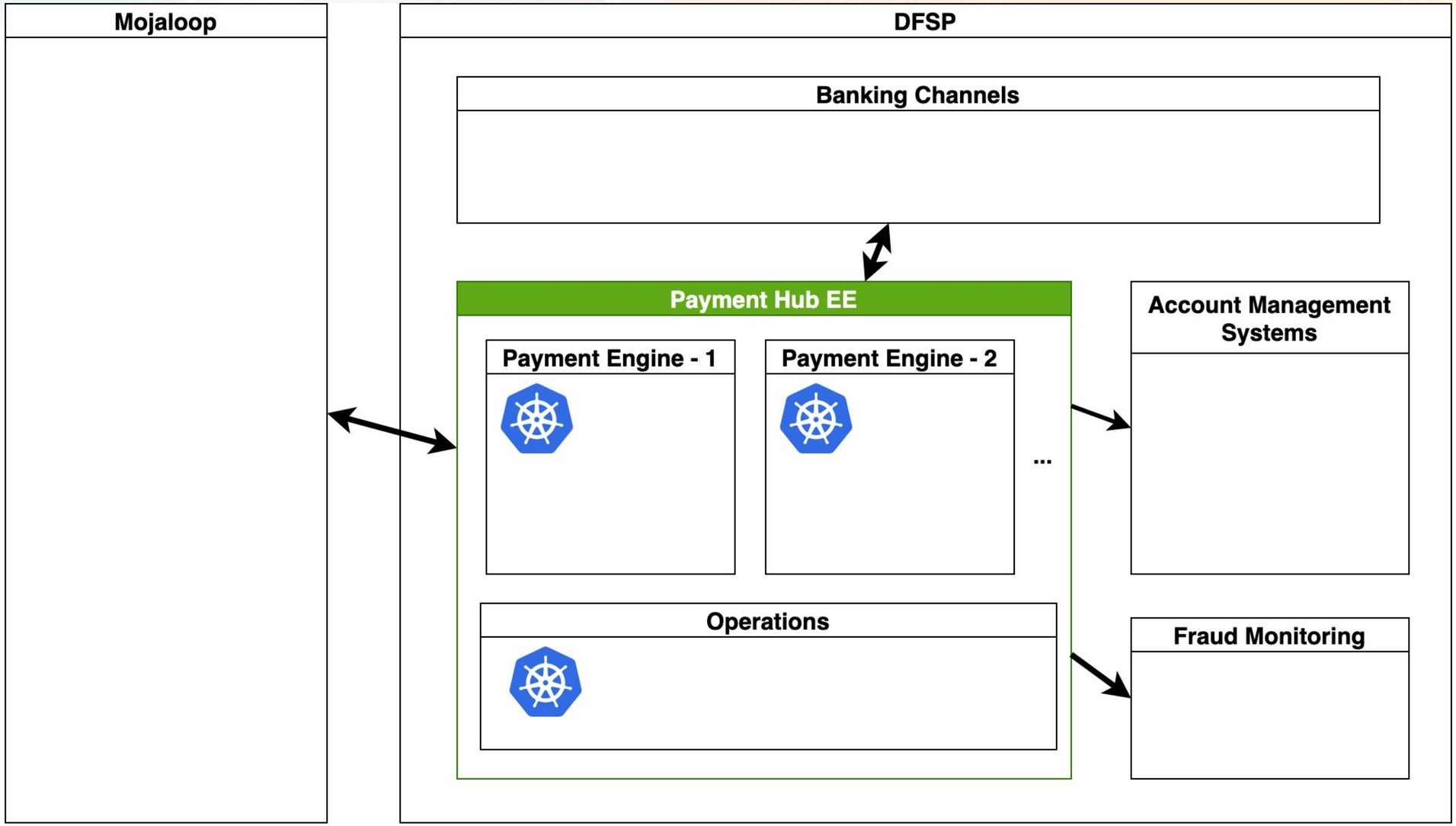
Key architectural considerations of Payment Hub EE 1/2

- Running on **on-premises** and **cloud** infrastructures, utilizing Kubernetes
- Separate **real-time engine** and **operations backend** systems
- The **Payment Realtime Engine** is **self contained, highly available and fault tolerant**, can handle complete transaction flows
- **Orchestrating** the microservices are desired
  - handling timeouts
  - correlations of asynchronous events
  - handling error and exception scenarios with the necessary compensations
  - overview of in-flight transactions
- A Payment Realtime Engine manages the **state machine using a microservice orchestration** layer
    - Using Raft for consensus and log replication
    - Using RocksDB for key-value store of states
    - Does not use external NoSQL or Relational DB, which hinders scalability

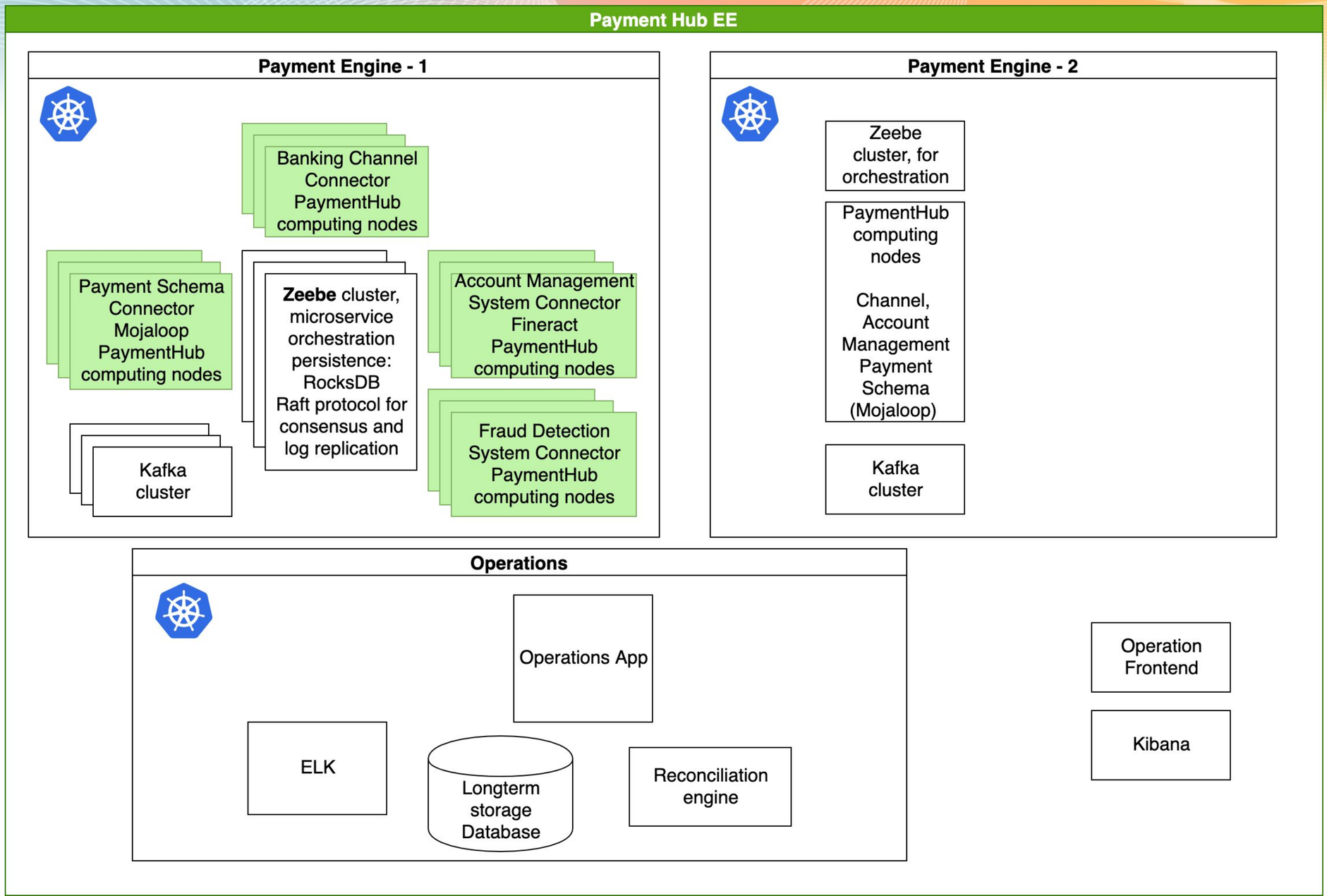
Key architectural considerations of Payment Hub EE 2/2

- **One engine is bounded to a single data center**, for high performance, low latency and minimizing issues from network failure scenarios
- **Multiple realtime engines could run parallel and independent**
  - Protect against complete site failure
  - 24x7, 99.99+% availability operations require version upgrades, certificate changes, hardware replacement, ... without interruption of the service
  - If payment network supports the notion of independent engines at a single DFSP, than routing could happen at the Switch, otherwise, the Payment Hub realtime engines could hand over the callbacks amongst each-other internally
- Systems for **backoffice operations**, including long term storage database, audit logging, monitoring, reconciliation - **detached in a separate cluster**
  - These components **could be offline without any payment service interruption** and data loss - the realtime engines Kafka layer keeps the records
  - “Operations systems” are not performance critical
  - Backoffice user access allowed only to these components, not the realtime engines

Payment Hub EE in the Payment Context



Payment Hub Logical Components



Deployment models

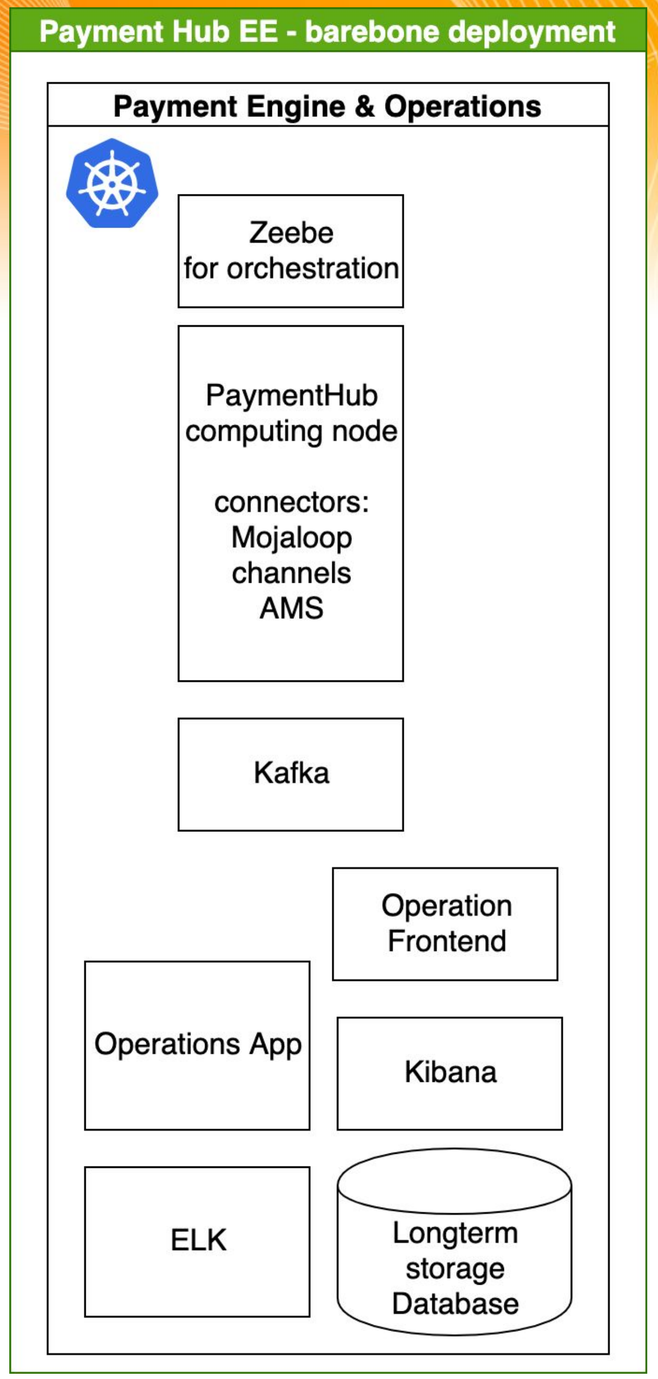
On-premises and any of the cloud providers

Shared service serving multiple DFSPs run by an aggregator (multiple SACCOs, credit unions on a multitenant setup) or dedicated setup

Depending on the DFSP requirements it could be deployed as

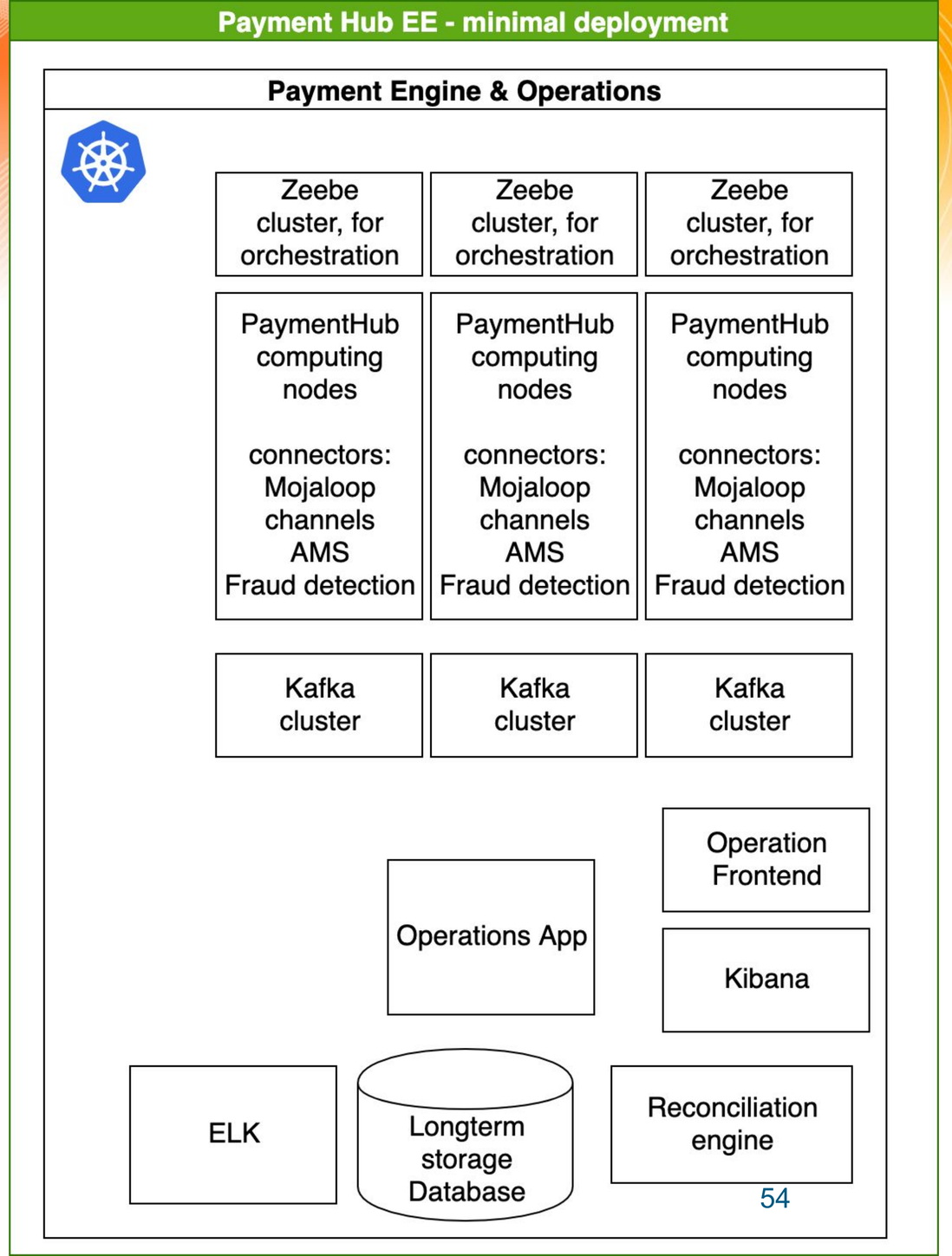
- **barebone** - single instance of components, minimized resource usage, no loss of functionality Might not run on a feature phone, but we will get there.
- **minimal** - single realtime engine
- **fully scaled** - multiple realtime engines

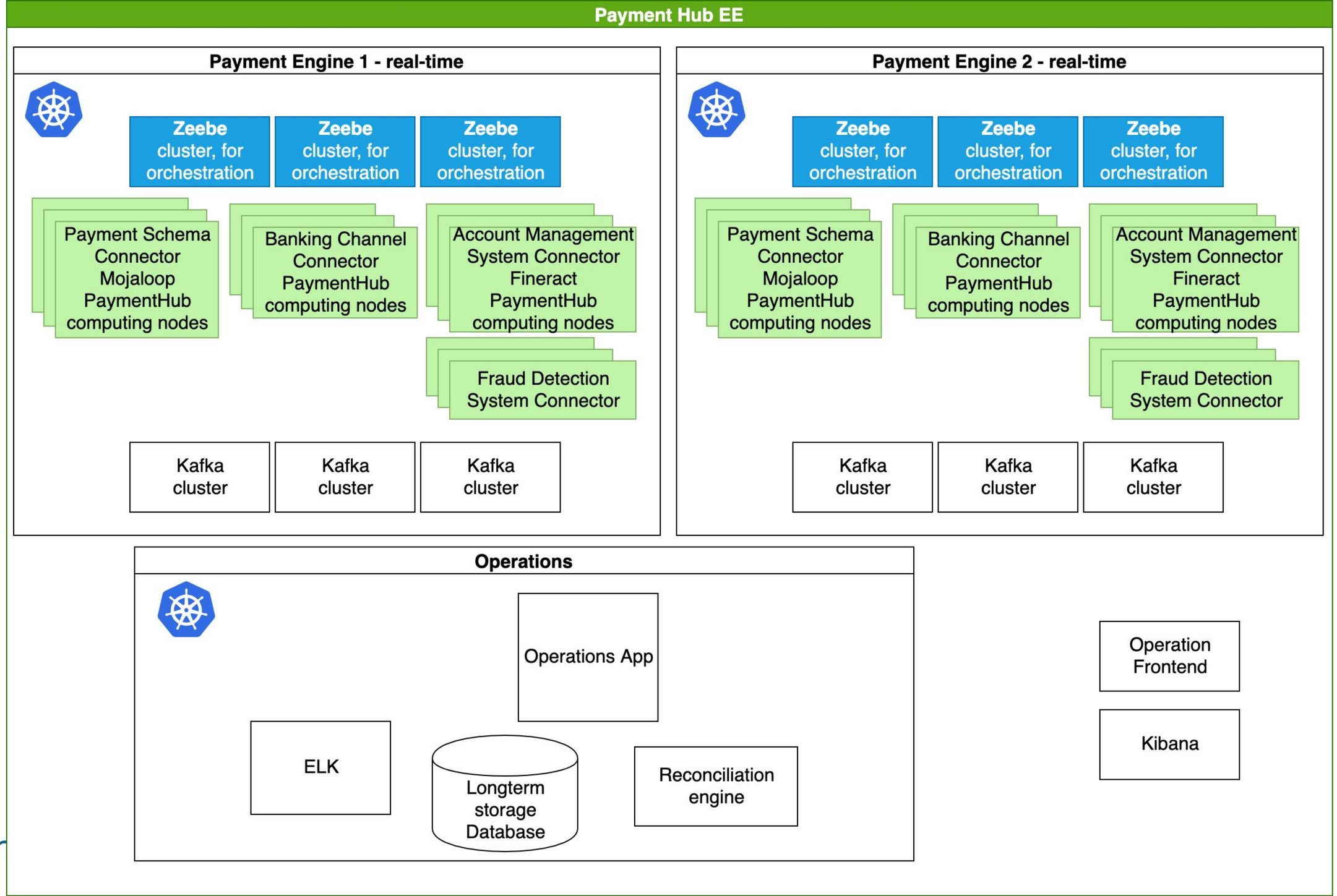
The difference is in availability, fault tolerance and the volume of transactions, which can be handled.



Deployment model - minimal

- Minimal deployment
  - Single kubernetes cluster to contain all the necessary components
  - Fault tolerance provided by the clustered components
  - Stretched installation across data centers possible, but not ideal
- Full scale deployment
  - Multiple independent payment engines (a single engine is collocated for performance), enabling complete version upgrades without service interruptions
  - Running in different data centers on independent network connections (high availability, fault tolerant even in case of disaster scenarios)
  - Partitioning the load across the engines



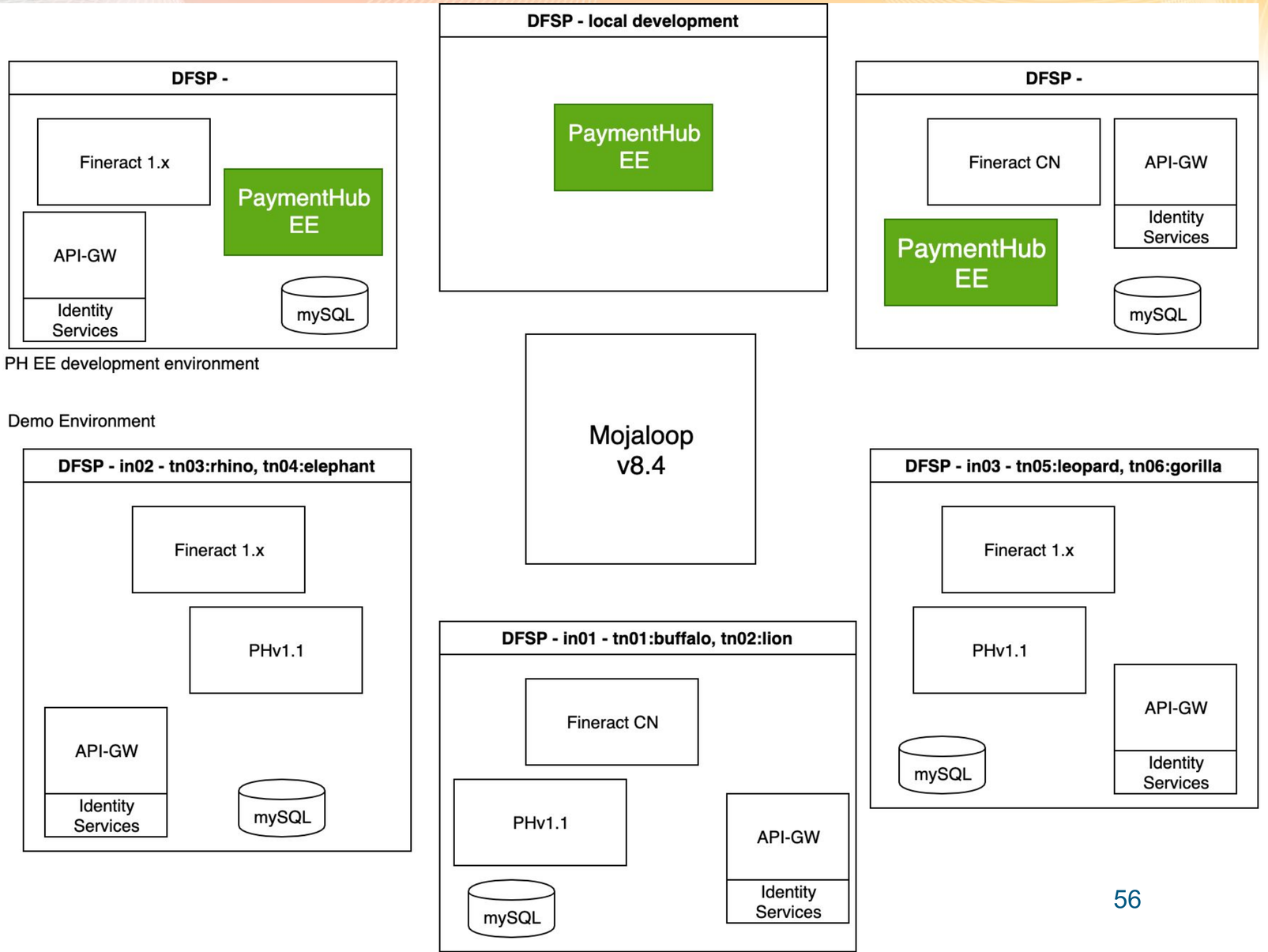

Payment Hub EE
Payment Engine 1 - real-time
Zeebe cluster, for orchestration
Zeebe cluster, for orchestration
Zeebe cluster, for orchestration
Payment Schema Connector Mojaloop PaymentHub computing nodes
Banking Channel Connector PaymentHub computing nodes
Account Management System Connector Fineract PaymentHub computing nodes
Fraud Detection System Connector
Kafka cluster
Kafka cluster
Kafka cluster
Payment Engine 2 - real-time
Zeebe cluster, for orchestration
Zeebe cluster, for orchestration
Zeebe cluster, for orchestration
Payment Schema Connector Mojaloop PaymentHub computing nodes
Banking Channel Connector PaymentHub computing nodes
Account Management System Connector Fineract PaymentHub computing nodes
Fraud Detection System Connector
Kafka cluster
Kafka cluster
Kafka cluster
Operations
Operations App
ELK
Longterm storage Database
Reconciliation engine
Operation Frontend
Kibana

Lab environment

New full instances running the PaymentHub EE engine along with Fineract versions.

Using the 3 deployment models.

Keeping the previous lab environment instances for the existing use cases, hackathons.



What is Zeebe and why to use it?

Opensource microservice orchestration engine, implemented in **highly scalable**, **highly performant** and **fault tolerant** way using state of the art, proven technology building blocks

- Using Raft protocol for consensus and replication
- Using RocksDB, an embeddable high performance key-value store, for its persistency
- All process instances are auditable, and audit records can be sent to Elasticsearch or Kafka
- Using an efficient, high performance binary protocol (gRPC) for the clients to connect and receive tasks for execution

Key features of Zeebe

Workflow definition	BPMN 2.0, YAML - provides integrators with an easy to use tool to customize workflows
Visual workflow representation	Yes, BPMN workflows are both executable and have visual representation
State storage for active workflow instances	Stored on the machines running Zeebe using embedded RocksDB (no external storage required)
Scalability	Horizontally scalable via partitions (no DB bottleneck)
Fault tolerance	Yes, via replication factor (3 node, 1 can fail, or 5 node and 2 can fail)
Supported programming languages	Ships with Java and Go clients, plus NodeJS, C# and Ruby clients available, using gRPC for the clients
Historic workflow data	Can be streamed to storage systems via exporters (Elasticsearch and Apache Kafka is available). Complete auditable log.

Focus on system integrators

- BPMN let implementation team **focus on the business flow changes inside the given DFSP**, making adoption and customization much easier
- **Multi programming language support for component implementation**, let implementation team to create the necessary connector components in any of the supported languages, the engine is able to orchestrate the steps. (e.g. multiple AMS integration is required), using gRPC protocol for efficient communication
- **Connector components are stateless, easy and simple implementation**
- No additional components required in the realtime engines, no databases to maintain, all active process states are stored in the embedded RocksDB data store. Auditlogs, and other events are stored in Kafka clusters before storing them
- Scalability - using partitioning (sharding), the system scales horizontally to create thousands of workflow instances per second
- Fault tolerance enables, that system recovers from a failure without data loss, and using replication to define the number of copies for all instances
  In case of failures, “leader election” is taking place for the given partitions to continue operations uninterrupted (using Raft Consensus and Replication Protocol)

What is important

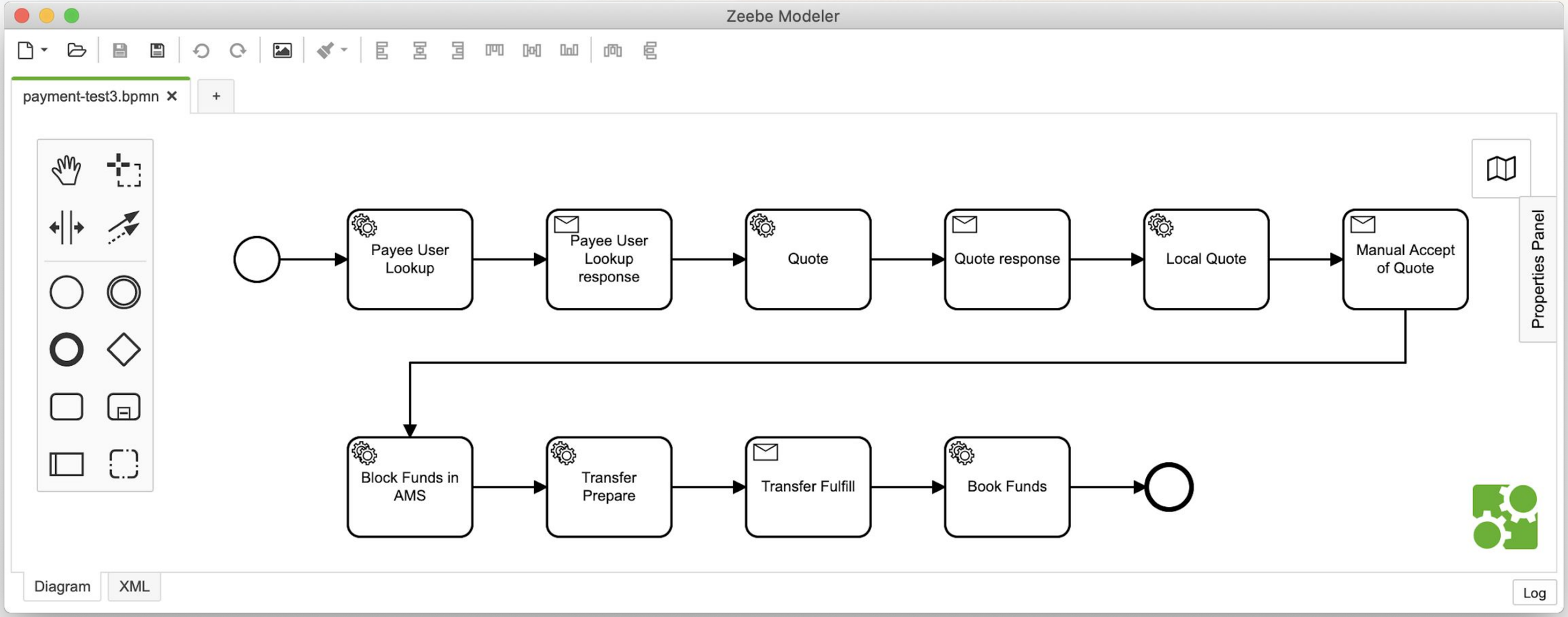
Timers surviving node failures without interruption

Correlation of incoming messages to existing workflow instances across the cluster nodes

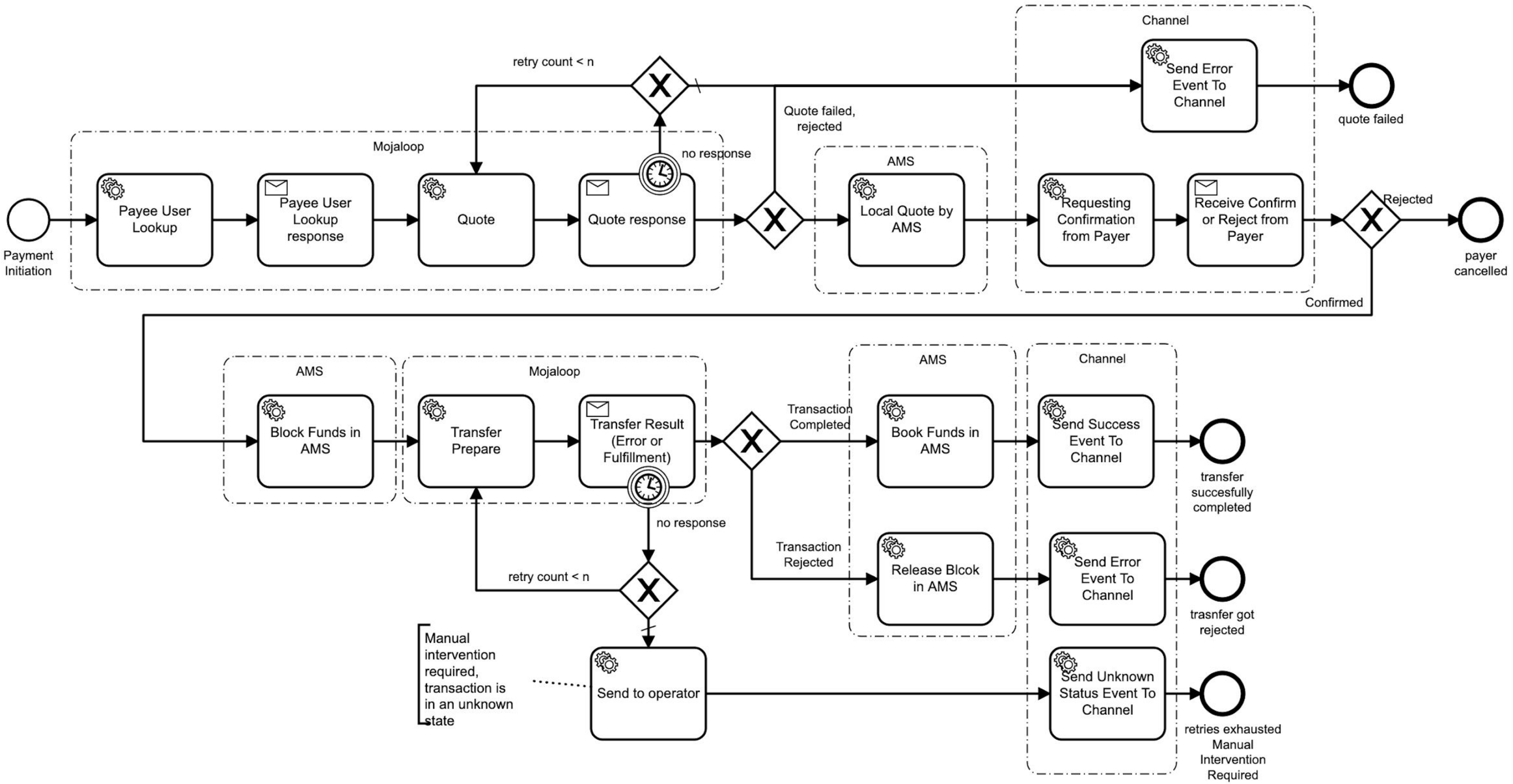
Capability to handle non correlating cases - important for cross payment hub engine flows

Handle failover situation without interruption (3 nodes: 1 can fail, or 5 nodes: 2 can fail) - utilizing Raft protocol for consensus, hand over “leadership” in case failed nodes and replication of state

The vanilla payment flow from the Payer perspective



Payer's flow with some error handling



Error Handling

Error Handling

Connect to the new error handling APIs

Update AMS to return with the most specific error

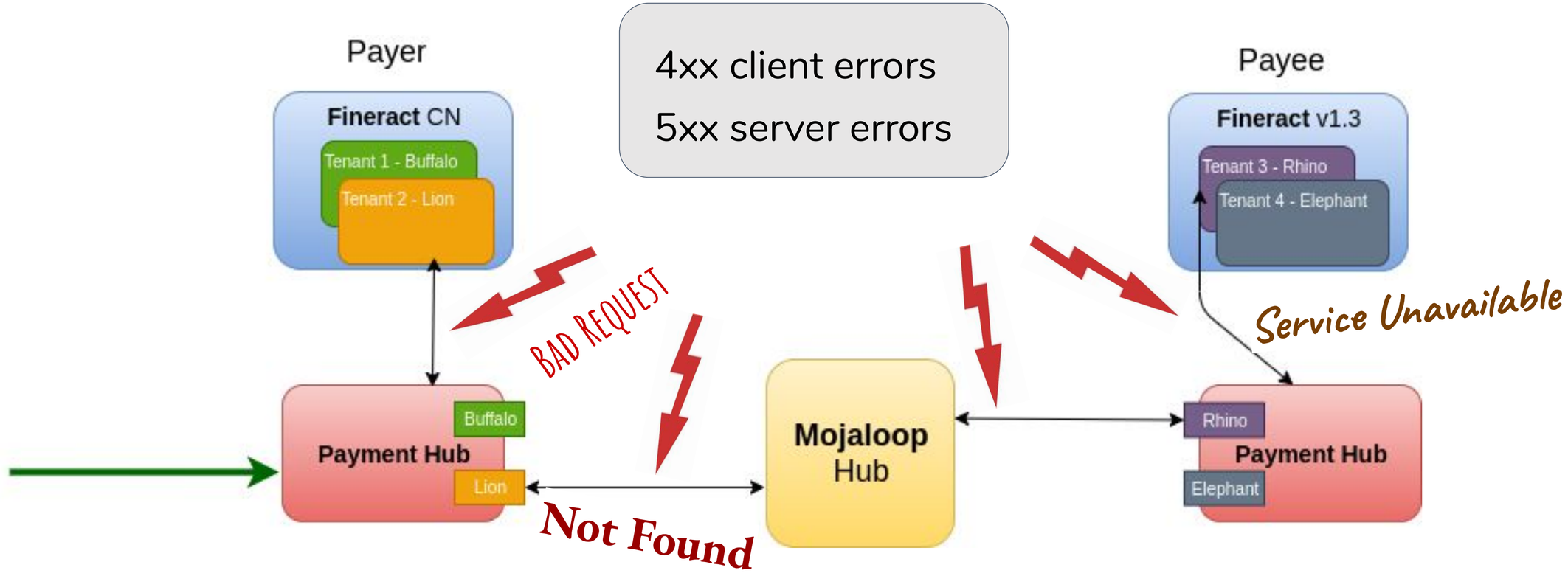
First party provider improvements to show error scenarios

Audit logs

Reports

Provide an administration interface

Standard HTTP errors

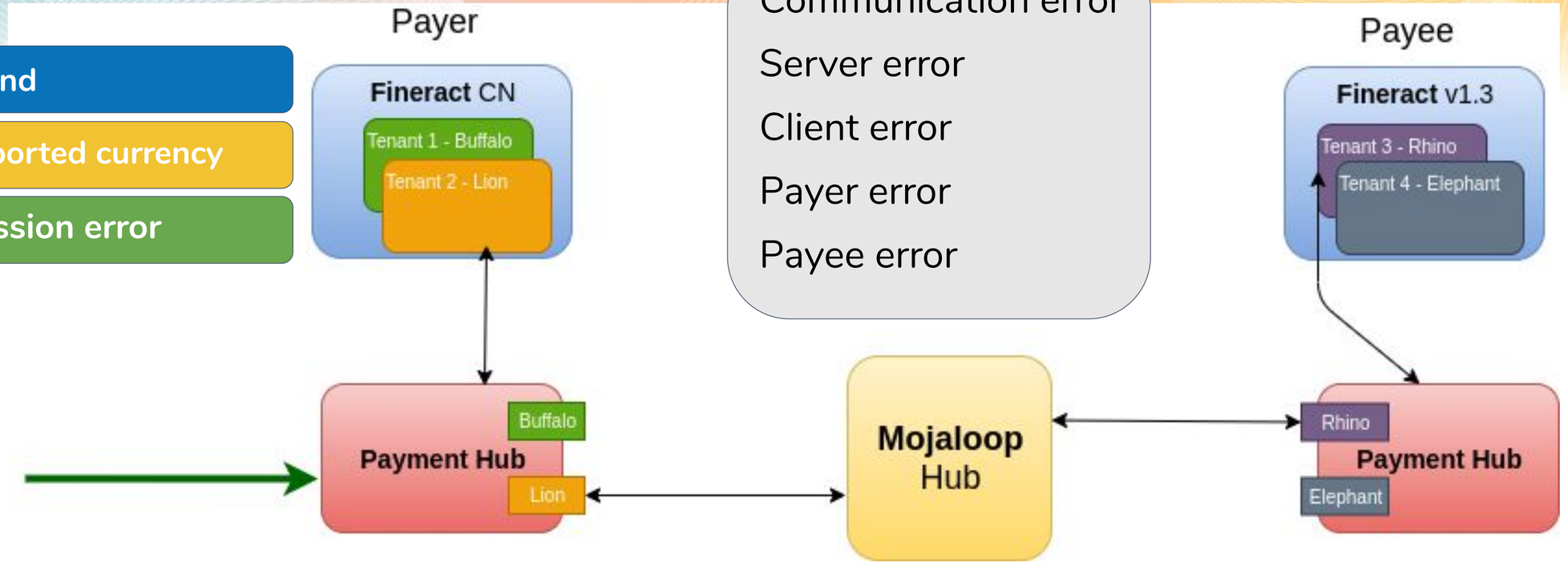


Interoperability Framework errors

Party not found

Payer unsupported currency

Payer permission error



Communication error

Server error

Client error

Payer error

Payee error

Unacceptable version requested
Missing mandatory element
Transfer expired

Payer FSP ID not found
Payee FSP ID not found
Party not found

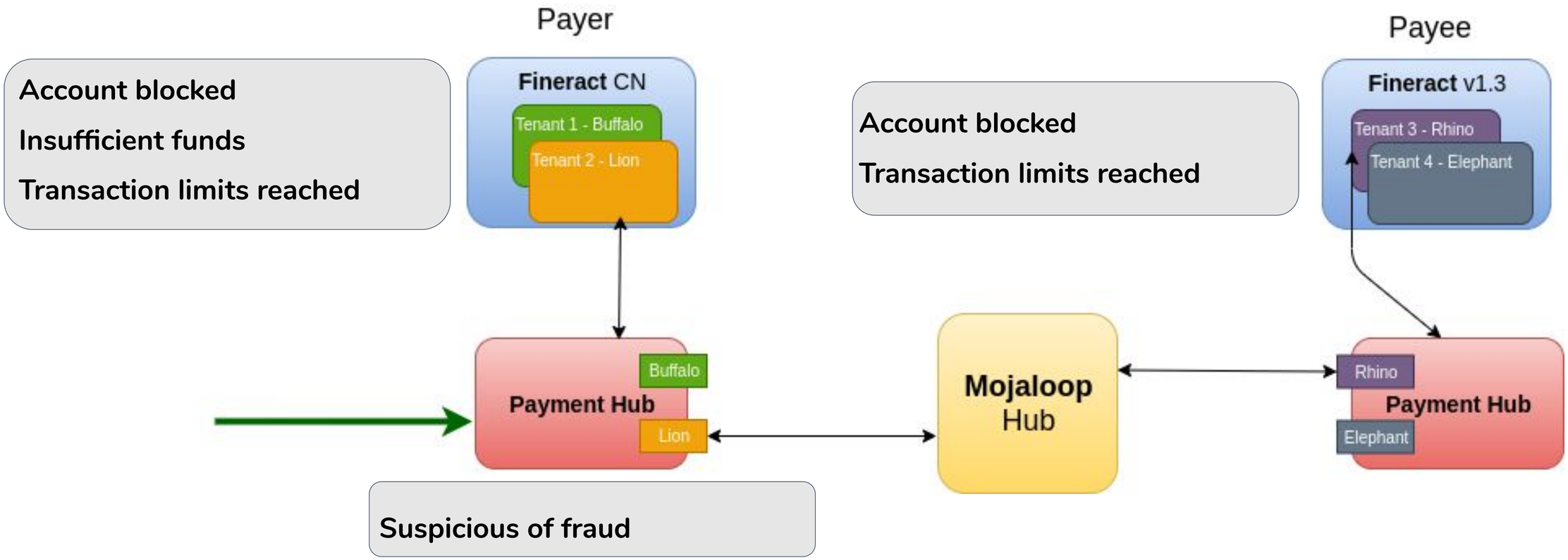
Payer limit error
Quote ID not found
Payee unsupported currency

Payer FSP insufficient liquidity
Transfer ID not found
Transfer expired

Interoperability errors in Payment Hub

- Communication error
- Destination communication error
- Generic server error
- Internal server error
- Not implemented
- Service currently unavailable
- Server timed-out
- Generic client error
- Unacceptable version requested
- Unknown URI
- Add Party information error
- Generic validation error
- Malformed syntax
- Missing mandatory element
- Modified request
- Generic ID not found
- Payer FSP ID not found
- Payee FSP ID not found
- Party not found
- Quote ID not found
- Transaction ID not found
- Transfer ID not found
- Transaction request expired
- Quote expired
- Transfer expired
- Payee unsupported currency
- Generic Payer error
- Payer FSP unsupported transaction type
- Payer unsupported currency
- Payer limit error
- Payer permission error
- Payer FSP insufficient liquidity
- Payee unsupported currency
- Payee limit error
- Payee permission error
- Payee FSP rejected quote
- Payee unsupported currency
- Payee FSP unsupported transaction type
- Payee unsupported currency
- Payee limit error
- Payee permission error
- Payee FSP rejected quote

Errors occuring at the DFSPs



Error Handling in Payment Hub EE

Automatic

- log each transaction step for audit purposes
- recovering from errors:
  retry steps according to schema rules and monitoring timeouts and continue the workflow without error
- eliminate the inconsistent state:
  releases blocked amount in AMS, if transaction gets rejected
- notify the payer
- notify back office operations
- send error report

Error Handling in Payment Hub EE

Manual

use-cases:

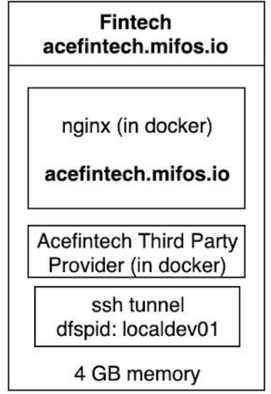
- blocked funds was not released because neither “transaction fulfillment notification” nor error was received on transaction execution, after multiple retries
- error happened after the Payee booked the transfer amount

Administration page

- recovering by operators
- search transaction and check details of executed flows
- reports
- API for Account Management Service (AMS)

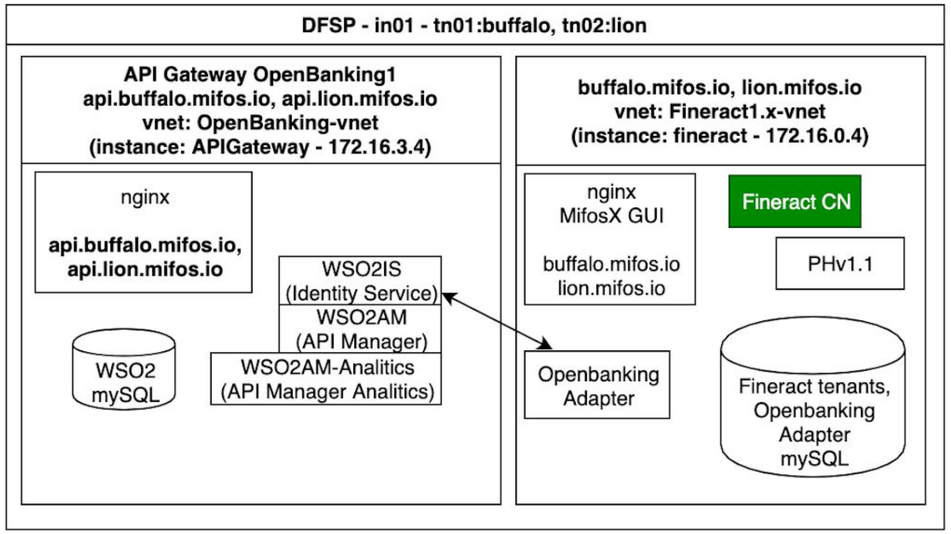
Lab environment in Azure

Azure configuration

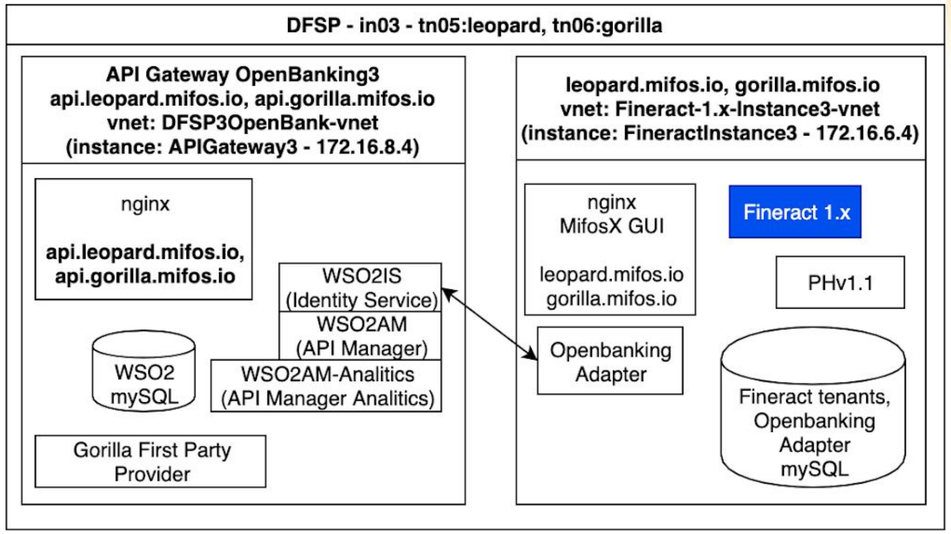


TPP

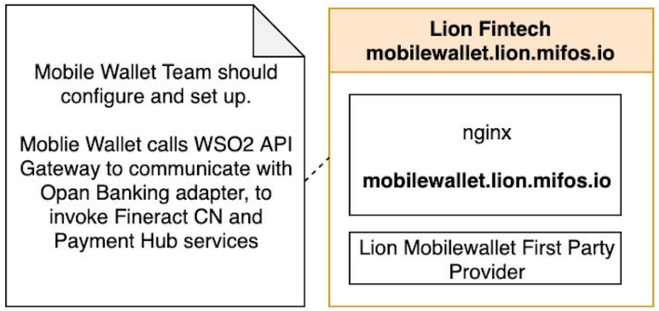
Instance 1

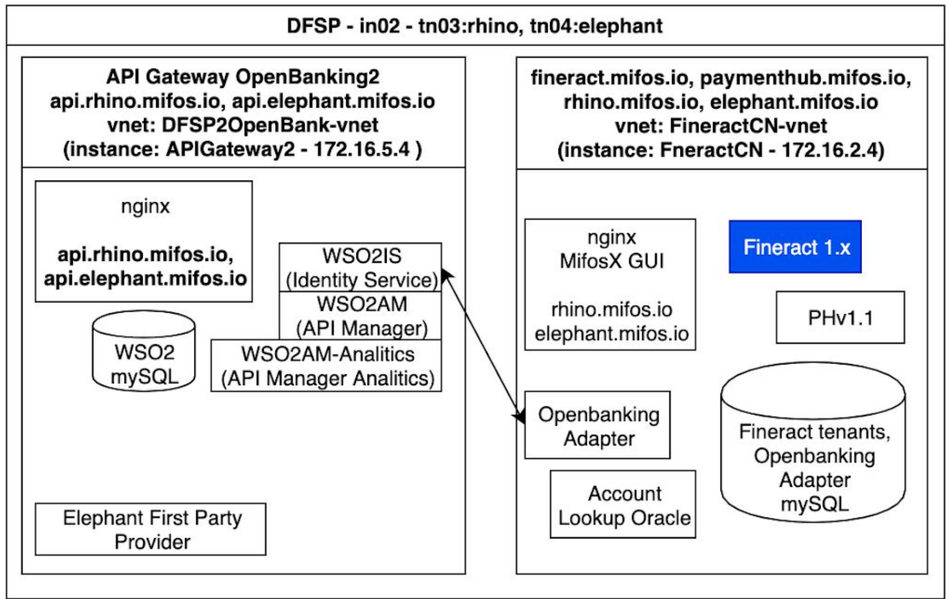


Instance 3



Instance 2





Azure configuration

Fintech
acefintech.mifos.io

nginx (in docker)

acefintech.mifos.io

Acefintech Third Party Provider (in docker)

ssh tunnel
dfspid: localdev01

4 GB memory

TPP

Mojaloop
v8.4.0

Instance 1

DFSP - in01 - tn01:buffalo, tn02:lion

Instance 3

DFSP - in03 - tn05:leopard, tn06:gorilla

Mobile Wallet Team should configure and set up.

Moblie Wallet calls WSO2 API Gateway to communicate with Opan Banking adapter, to invoke Fineract CN and Payment Hub services

Lion Fintech
mobilewallet.lion.mifos.io

nginx

mobilewallet.lion.mifos.io

Lion Mobilewallet First Party Provider

Instance 2

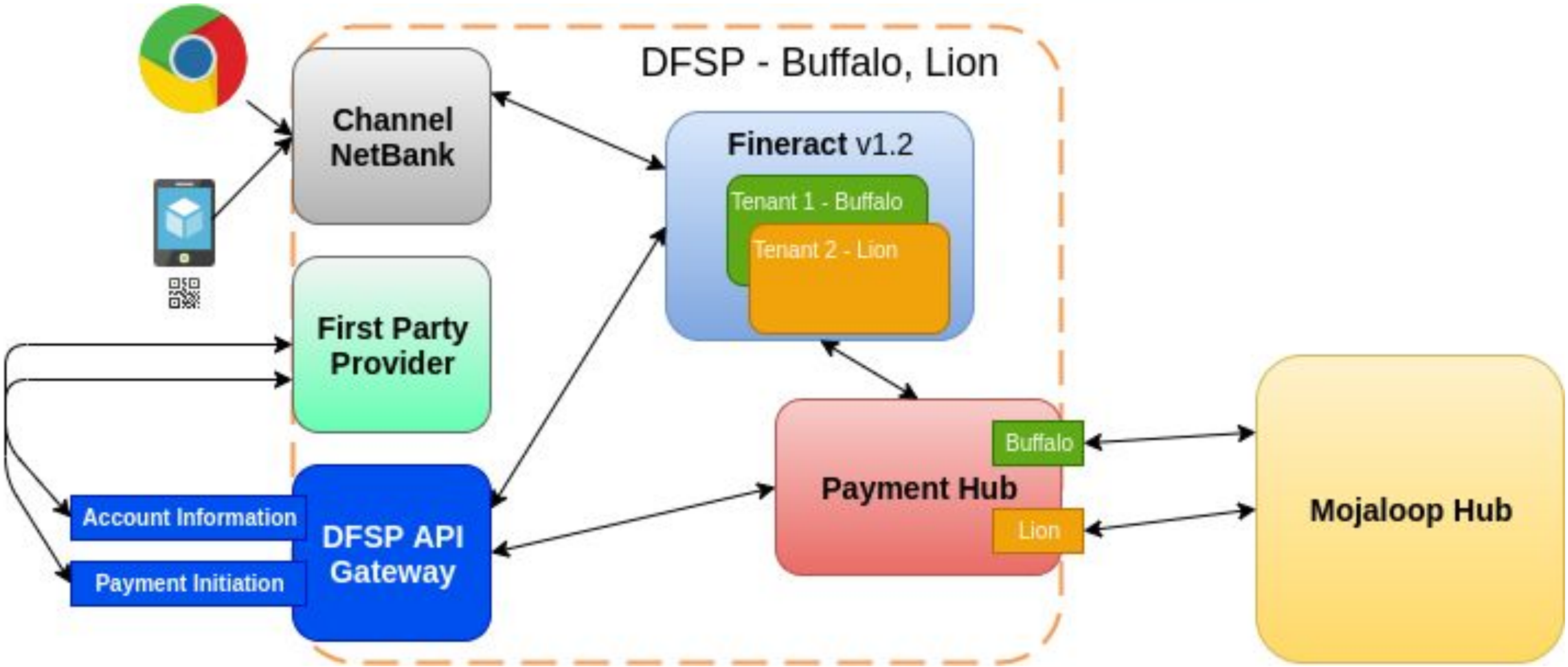
DFSP - in02 - tn03:rhino, tn04:elephant

Elephant First Party Provider

Lookup Oracle

mySQL

Azure infrastructure - Instance 1

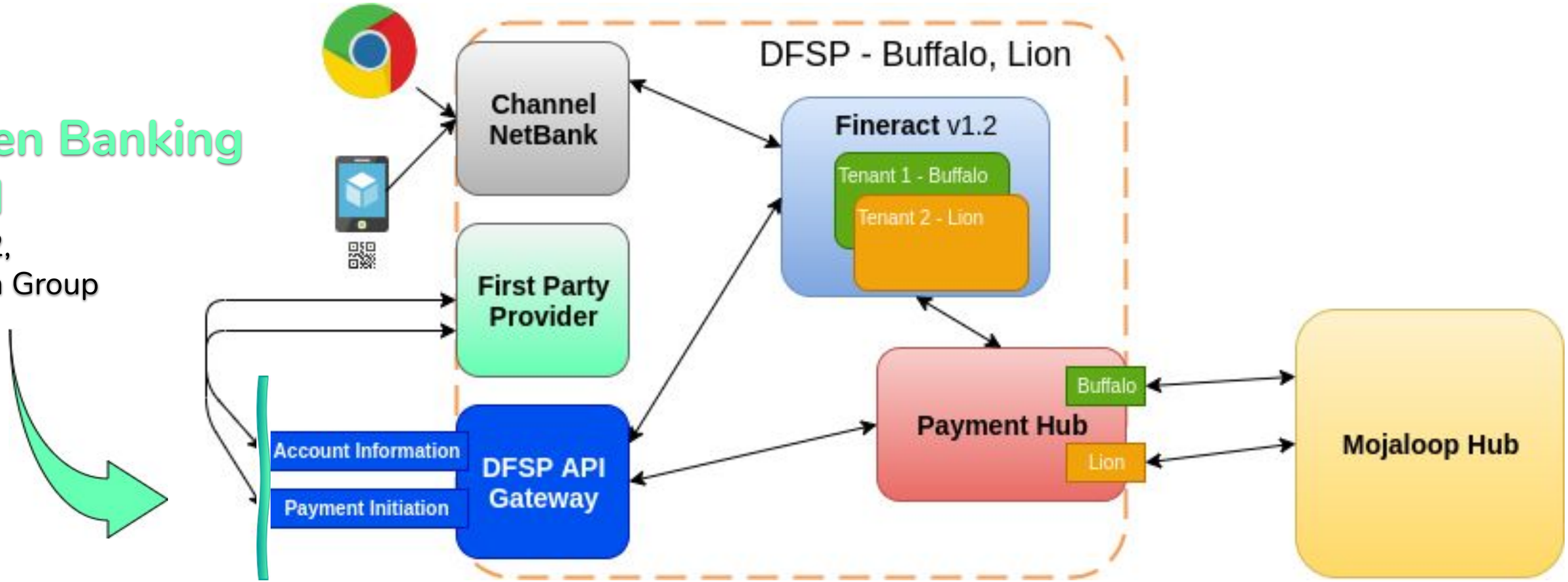


API Gateway

API Gateway - WSO2

Open Banking API

PSD2,
Berlin Group



Open Banking API

“Open Banking is the secure way to give providers access to your financial information.”

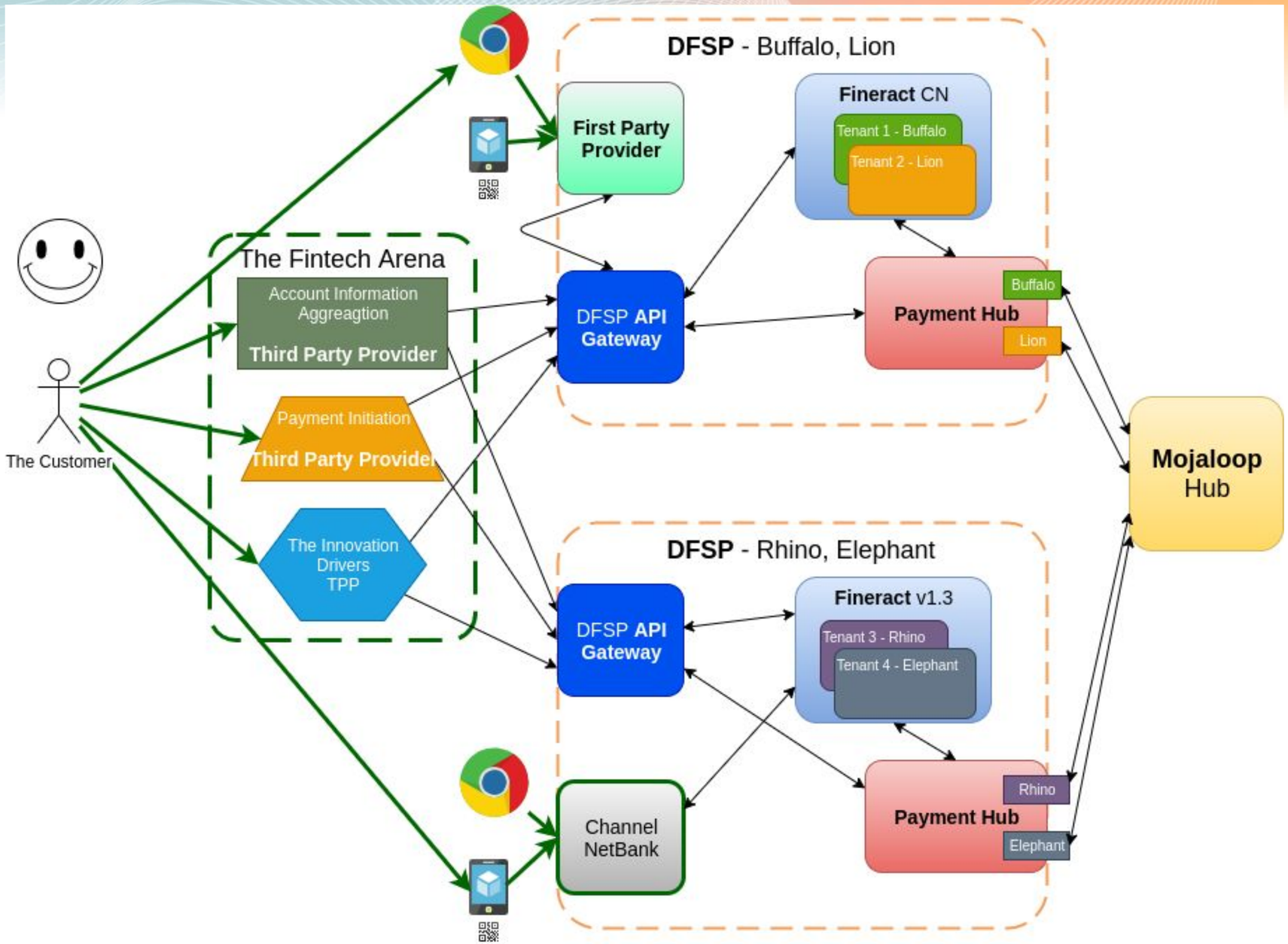
“It opens the way to new products and services that could help customers and small to medium-sized businesses get a better deal. It could also give you a more detailed understanding of your accounts, and help you find new ways to make the most of your money.”

Enable third parties (fintechs) to access multiple DFSPs APIs using a standardised mechanism.

Terminology used:

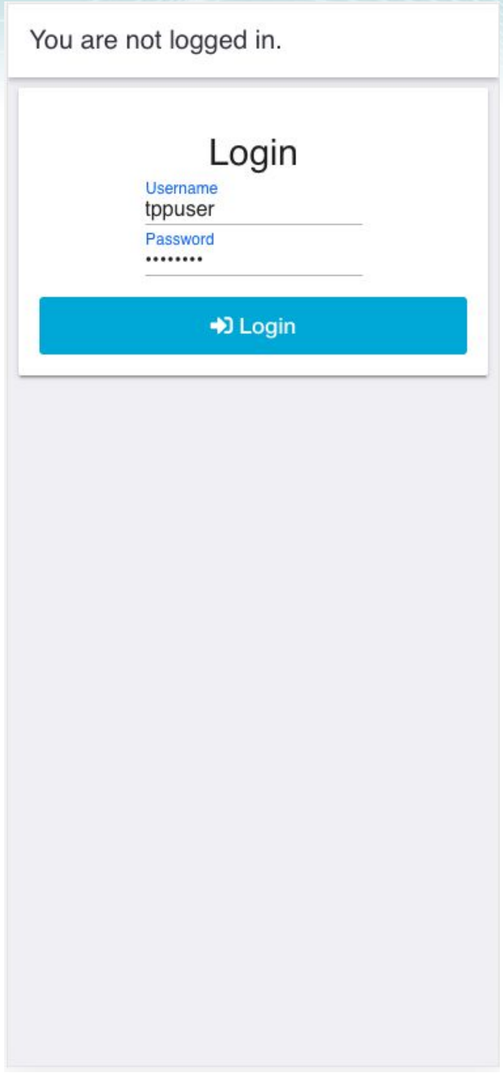
- ASPSP - Account Servicing Payment Service Provider - DFSP
- PSU - Payment Service User - account holder at one or multiple ASPSP
- AISP - Account Information Service Provider
- PISP - Payment Initiation Service Provider

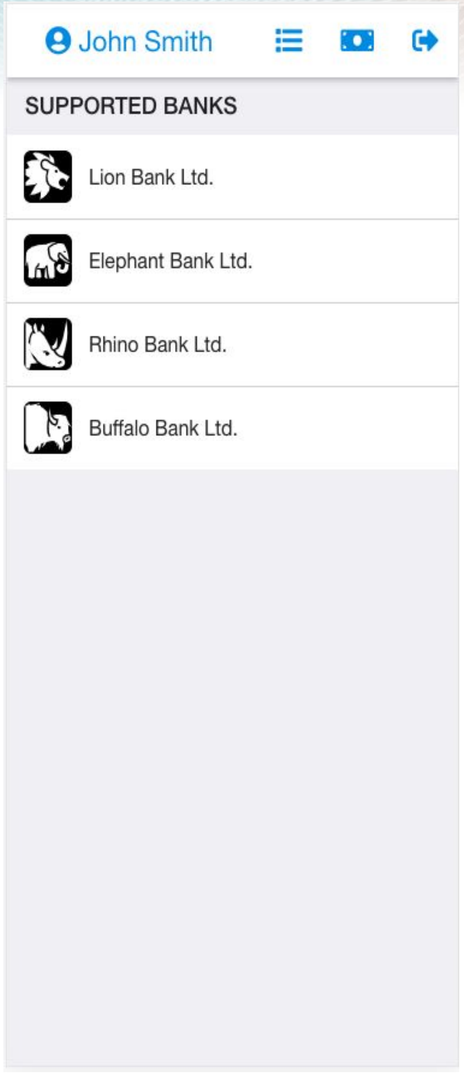
API Gateway - WSO2

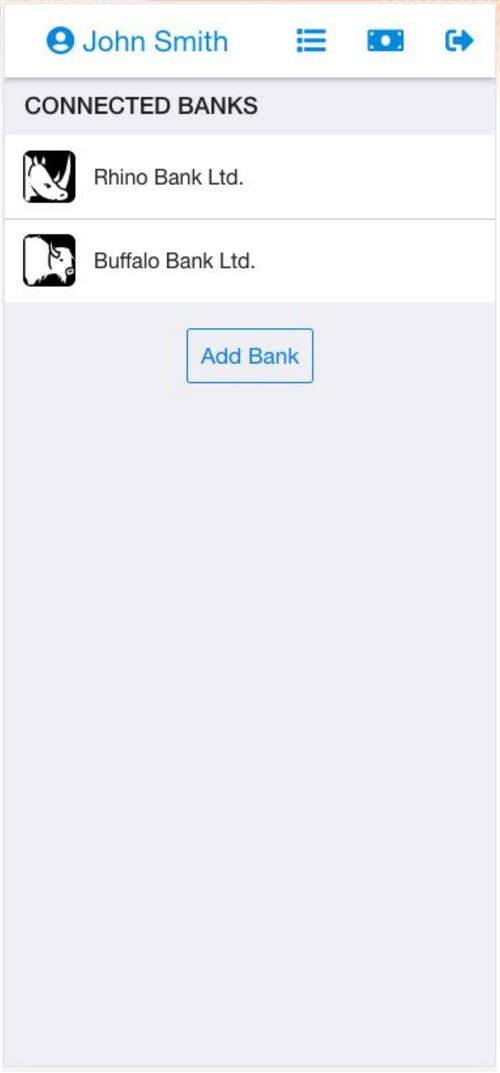


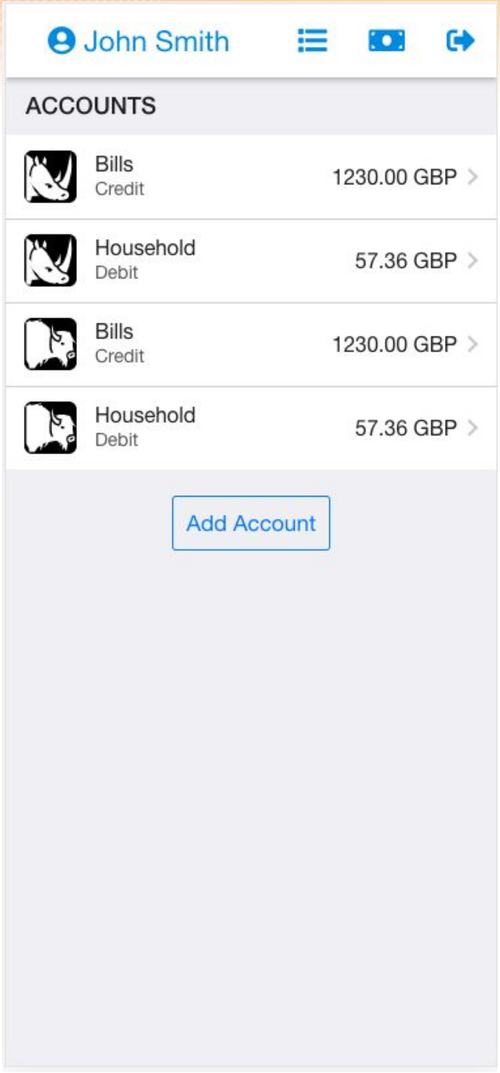
- Enable TPP to access multiple DFSPs, self service onboarding
- Using standardised Open Banking APIs (PSD2, Berlin Group)
- Client and user authentication only once, authorization (OAuth2)
- User gives consent on account, action type, transaction level
- Input validation, fraud monitoring, load balancing, metrics

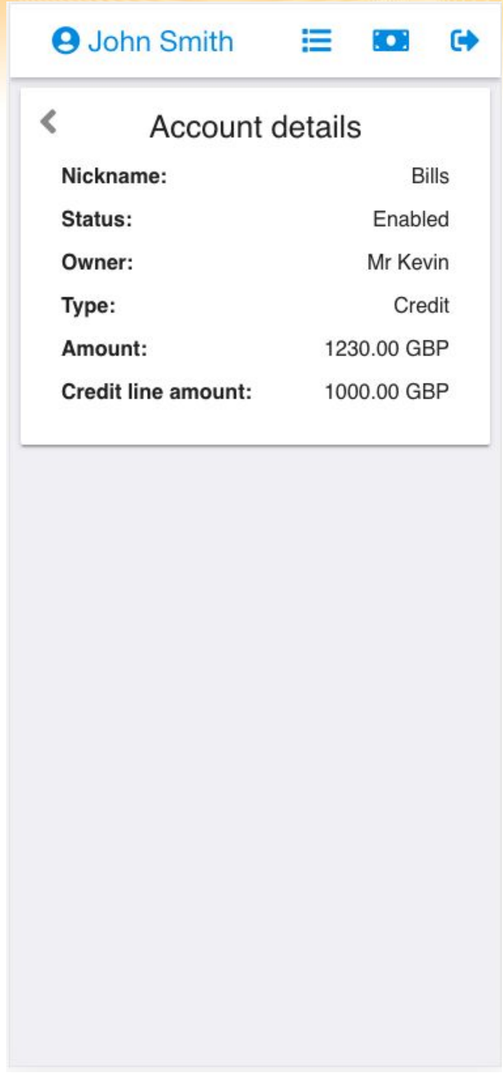
Interaction with the Payment Service User











Ideas on Google Pay Integration

If community decide to support Authentication, Authorization flows by Mojaloop, standardizing the approach for the participating DFSPs, the lab environment with the API Gateways and DFSP based Identity managers could be a good candidate to demonstrate the new functionality.

This enables all 3rd party Fintechs, including Google Pay and even the first party DFSP applications (like mobile banking) to easily join the platform with a simple, secure and standardized consent management flow and user experience.