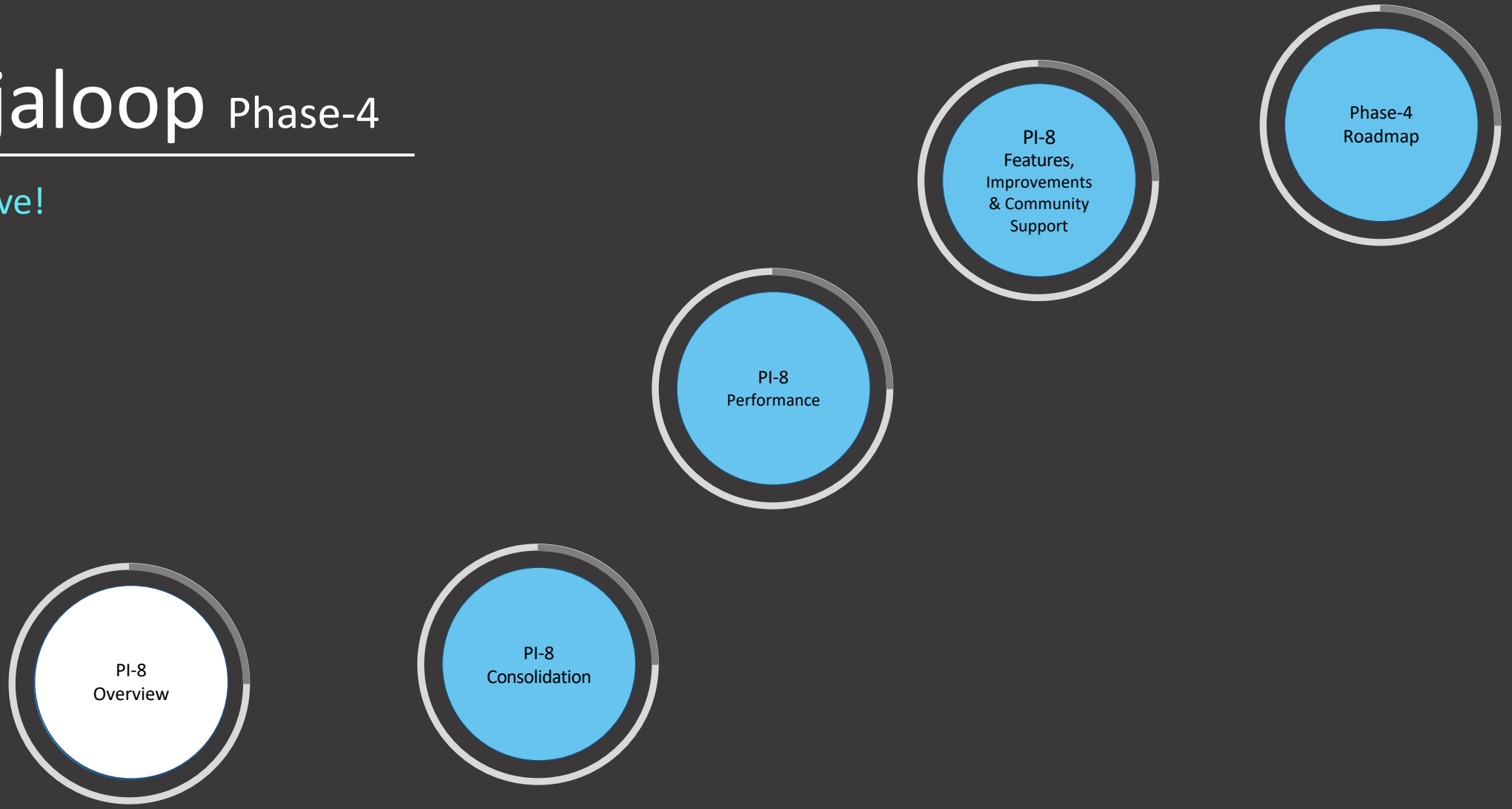
mojaloop

Mojaloop Phase-4

Going live!

Mojaloop Phase-4

Going Live!



Mojaloop PIs Overview

Timeline	Summary
Phase-1 (2016 - 17)	**Level One Project** • Reference Implementation • 6 Program Increments (PIs)
Phase-2 (2018)	**Road To Productionization** • 1 – 4 Program Increments
Phase-3 (2019 Jan - Dec)	**Supporting Adoption & Deployment** • PI-5 (Feb – April): Account lookup, QA Framework, Streamlined CI, Release process, Error endpoints, Documentation, Node Upgrade, Bug Fixes & Community support, Bulk Transfers Design • PI-6 Event handling framework, Bulk Transfers PoC, API Gateway, OSS Settlements API, Quoting Service, ALS • PI-7 Event & Error Handling framework, Packaging, OSS Settlements, Performance testing capabilities, QA • **PI-8 Consolidation, Performance, Community Support**
Phase-4 (2020 Jan -)	**Going Live** • **PI-9** (Jan/Feb – April) • PI-10 (May – July) • PI-11 • PI-12

PI-8: Focus Areas

1. Consolidation of the Mojaloop OSS codebase
2. Performance testing and addressing regression
3. Community Support

Switch Functionality – Mojaloop End-points (PI7→PI8)

Mojaloop v1.0 – API Specification

Transfers*
[●] POST - Prepare
[●] PUT - Response
[●] PUT – Error
[●] Out…ping
[●] …ing
[●] …Query

1. **Performance improvements**

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
[●] PoC / Initial Version
[●] Partially implemented
[●] Not implemented
[○] Future Roadmap

* Enhancements for PI8

Switch Functionality – Mojaloop (End of Phase-3)

Mojaloop v1.0: Focus Use-case – P2P

Payer-Initiated Transaction

[●] P2P Transfers – Golden Path
- [●] Prepares, Fulfils
- [●] Rejections, Timeouts
- [●] Error Endpoints

Mojaloop v1.0 – End Points for P2P

Transfers

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

[●] GET - Query

Quotes

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

Parties

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

Participants

[●] POST - Create
[●] PUT - Response

Key
[●] Fully implemented
[●] Legacy Code
[●] PoC / Initial Version
[●] Partially implemented
[●] Not implemented
[○] Out of Scope

Switch Functionality – Mojaloop Use Cases (End of Phase-3)

Mojaloop v1.0 – Use-cases

Payer-Initiated Transaction

[●] P2P Transfers
- [●] 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

Key
[●] Fully implemented
[●] Supported, not tested
[●] Proof of Concept
[●] Not implemented
[○] Out of Scope

End-of-PI8 Switch Functionality – Operations

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

Oracles

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

Settlements v**1.0** [Supports some forms]

[●] 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

Settlements v**2**

[●→●] Settlement models
[●→●] Designs
[●→●] Settlement by Currency
[ ] Interchange Fees
[ ] Continuous Gross Settlement

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

PI-8 Mojaloop OSS Community - 1

Contributions, Collaboration

1. Cross network/currency
2. Code quality and Security improvements
3. Finance portal (Settlement)
4. ISO Integration
5. Quality Assurance (bugs, coverage, Tests with mojaloop-simulator)
6. Settlement v2

PI-8 Mojaloop OSS Community - 2

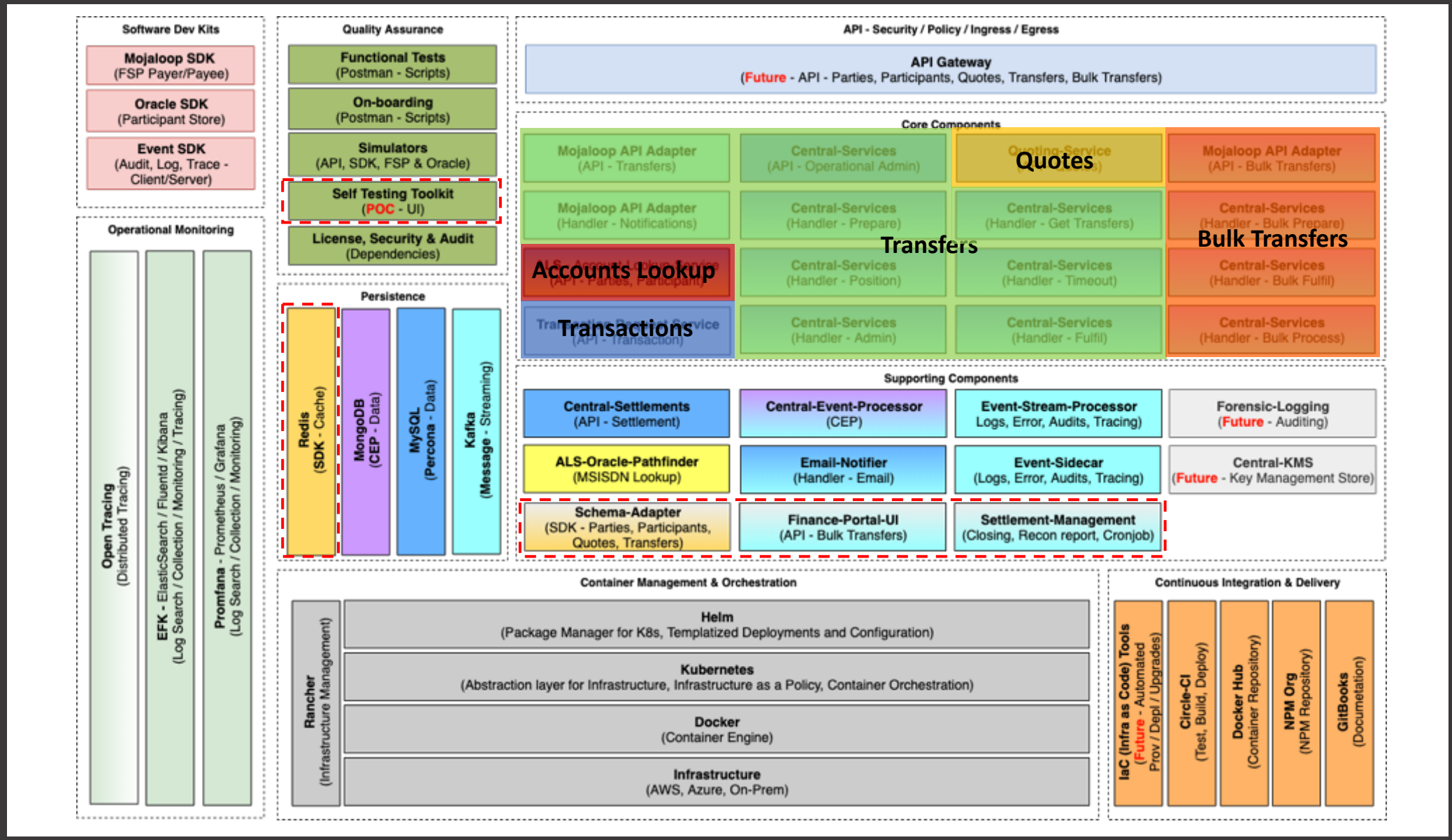
Contributions, Collaboration

1. ALS oracle for Pathfinder adapter
2. Documentation
3. Resource implementation */authorizations*

PI-8: Features

1. Event Framework
2. Settlements v2
3. Enhancements, Standardization:
    1. Account lookup service
    2. Transactions requests service
    3. Bulk Transfers
4. Testing toolkit (API Specification)

PI-8: Component Architecture



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

Mojaloop Phase-4

Going Live!

Consolidation: Mojaloop Helm Releases

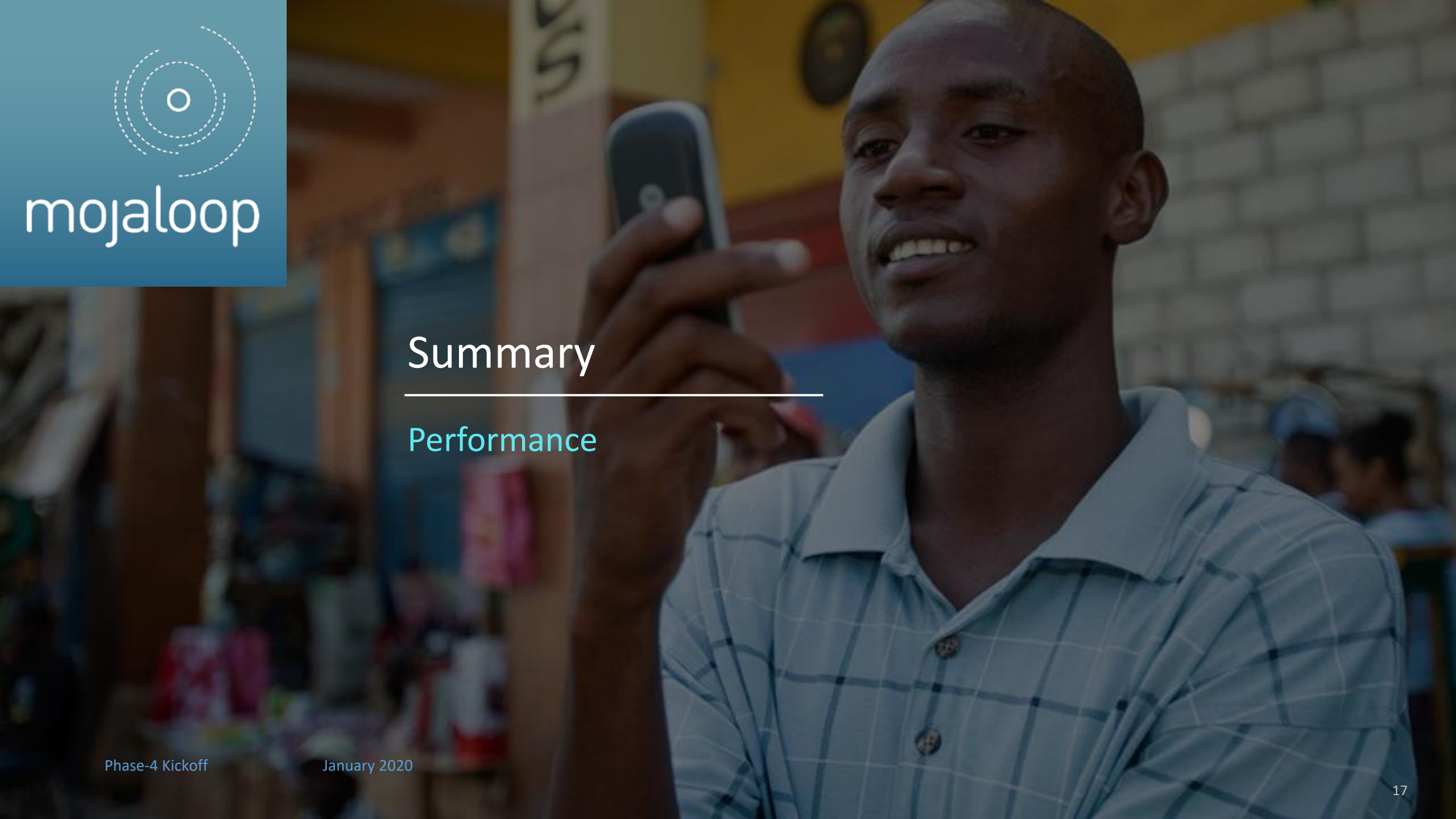
1. Mojaloop Helm releases: https://github.com/mojaloop/helm/releases
2. Helm releases in PI8
    a. v8.1.0 - https://github.com/mojaloop/helm/releases/tag/v8.1.0
    b. v8.4.0 - https://github.com/mojaloop/helm/releases/tag/v8.4.0
    c. v8.7.0 -https://github.com/mojaloop/helm/releases/tag/v8.7.0

PI-8 Consolidation: Bug Fixes

1. At the time of and after **v8.7.0 release** – all open bugs were resolved
2. Total of 49 bugs resolved in PI8
3. As part of bug resolution
    a. Along with validation after deployment,
    b. Test cases / automated tests added to scripts
4. QA: Postman collection for deployments using Mojaloop-simulator

Mojaloop Phase-4

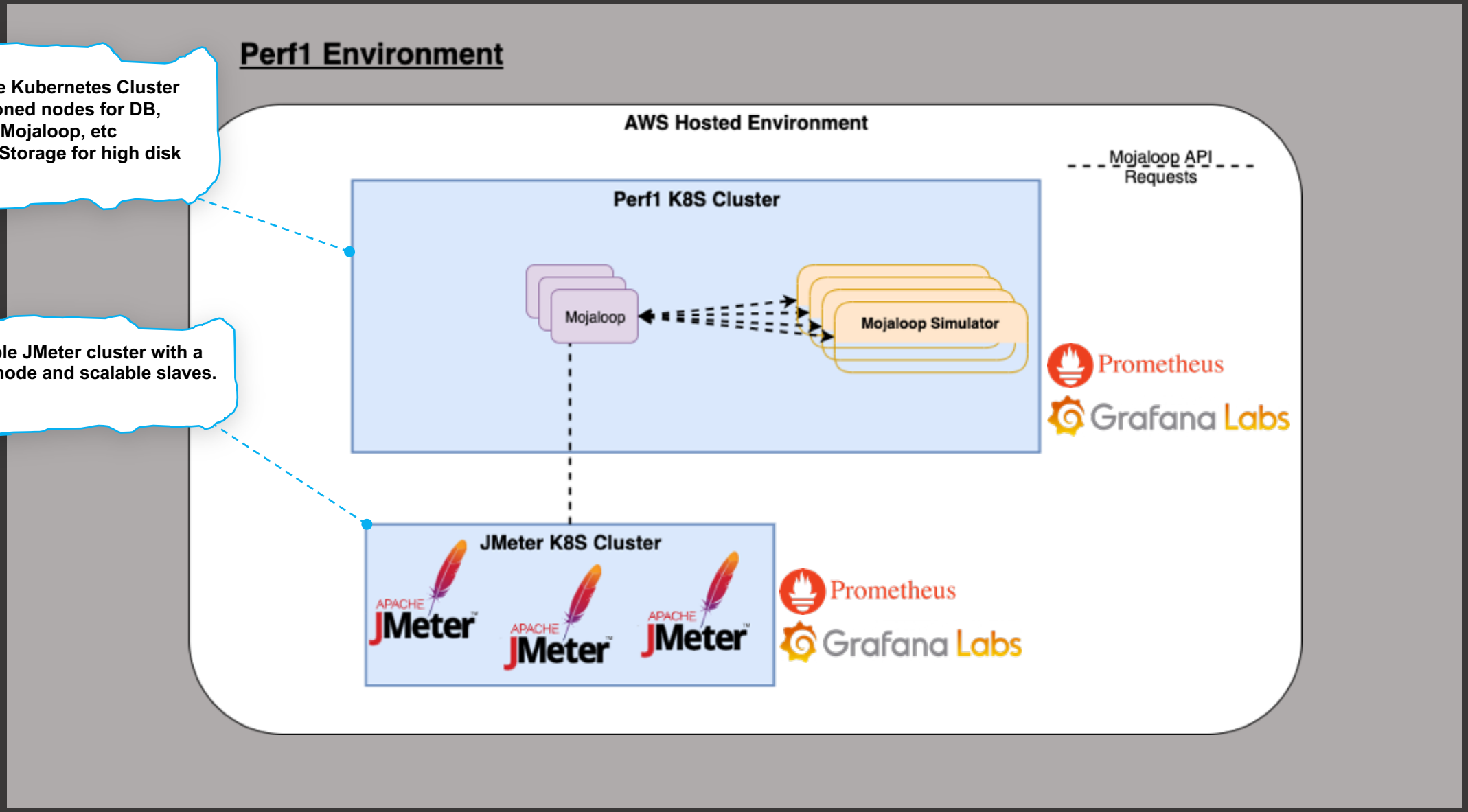
Going Live!



Summary

Performance

Perf1: Architecture Overview



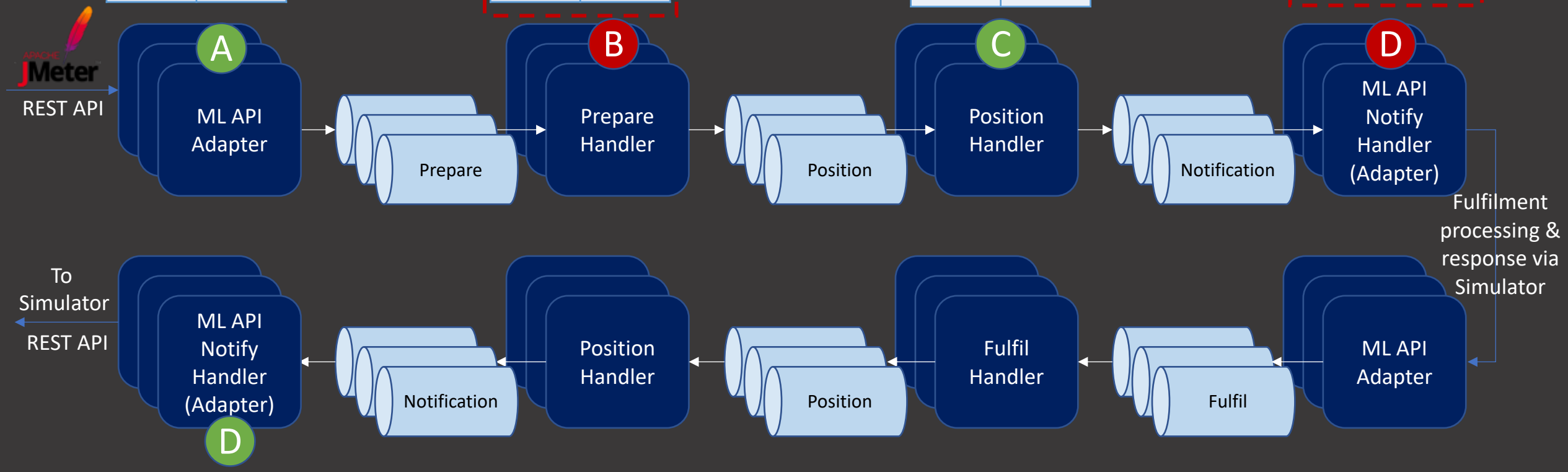
Observed Transfer Performance

TPS	Scale
286	1
536	2
752	3

TPS	Scale
225	4
242	8
242	12

TPS	Scale
225	12
242	16

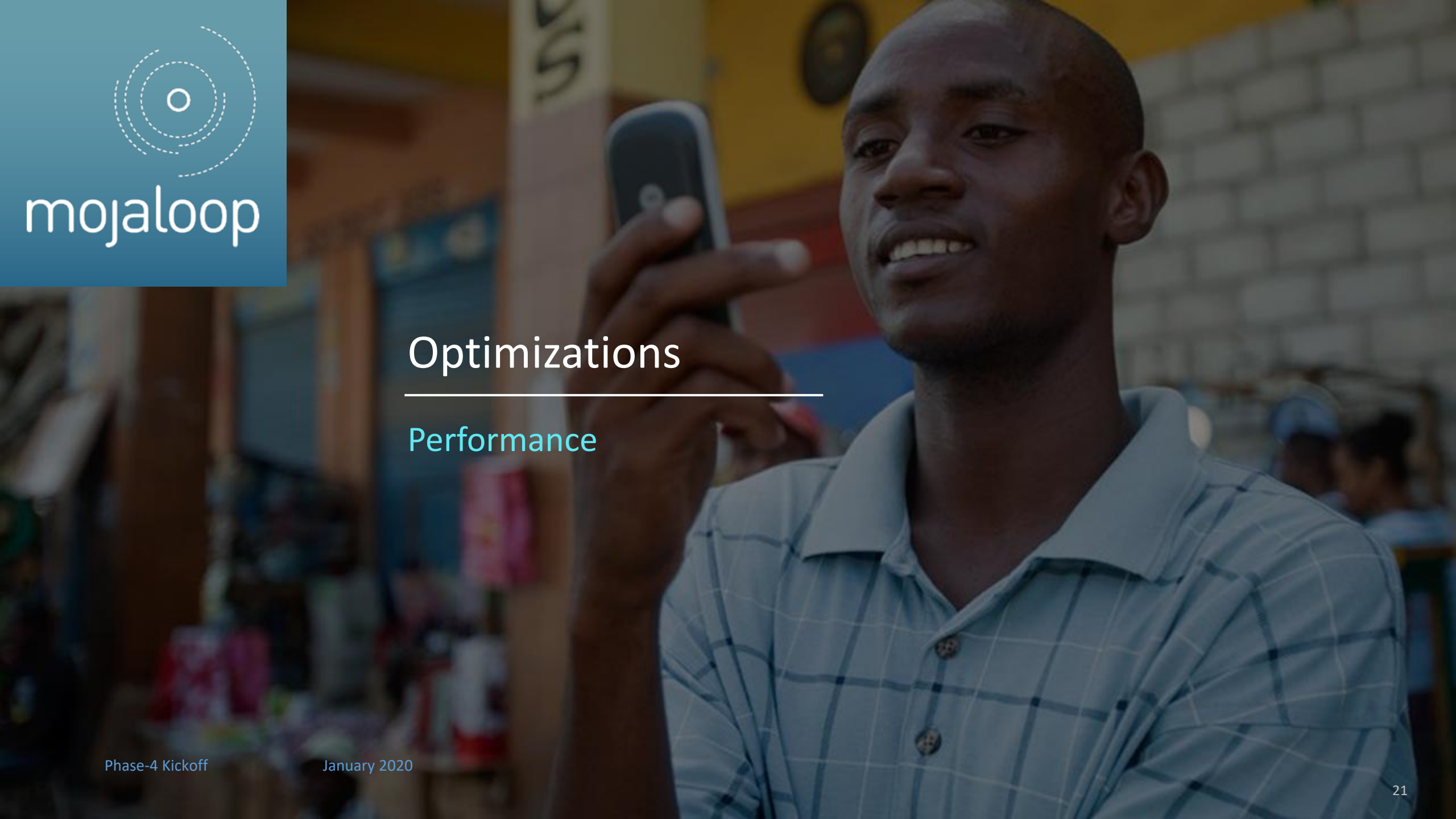
TPS	Scale
150	8
151	16



TPS: 150,
Shortest: 62ms, Longest: 2646ms, Average: 153ms, % > 1 Sec: 0.11%

ML Performance: Future Strategy

1. Performance focused work-stream
2. Understand performance characteristics, constraints based on data
    A. Determine Performance Characteristics at both Macro & Micro levels
        a. Micro – isolate components (e.g. Notification Handler, DB, Kafka)
        b. Macro – Used to determine focus for Micro optimizations and validate Micro optimizations once integrated.
3. Decide best optimizations to resolve identified issues
    a. PoC optimization
    b. Implement optimization if proven beneficial



Optimizations

Performance

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.

TO BE DONE REFACTORED?

Optimization – IO

Description:

- Slow response times observed on Handlers due to high IO

Resolution:

- Handlers:
  - Manual Commits were causing excessive high IO thread contention which was drastically reduced by enabling auto-commits
  - Optimized Kafka configurations to reduce unnecessary IO
  - Loggers were moved into an async process (20% improvement)

Considerations:

- Additional logic will need to be added in Handlers as the auto-commits allows for re-processing of messages. The handlers will not re-apply any business logic as the "state" of the transactions will have moved on, however it is possible that there will be unwanted notifications sent to the FSPs. Thus we will have to store the Kafka index for each message to validate if the message is a re-request from FSP or a re-processing of a previous message.

PoC Producer - Commit = Manual

Number of unique matched entries: **10000**
Total difference of all requests in milliseconds: 1308
Shortest response time in millisecond: 0
Longest response time in millisecond: 3
Mean/The average time a transaction takes in millisecond: **0.1122**
Standard **deviation** in milliseconds: **0.3206**
Number of entries that took longer than a second: 0
% of entries that took longer than a second: 0.00%
Average **transactions per second**: **7645.26**

PoC Producer - Commit = Auto

Number of unique matched entries: **10000**
Total difference of all requests in milliseconds: 1234
Shortest response time in millisecond: 0
Longest response time in millisecond: 3
Mean/The average time a transaction takes in millisecond: **0.1023**
Standard **deviation** in milliseconds: **0.3080**
Number of entries that took longer than a second: 0
% of entries that took longer than a second: 0.00%
Average **transactions per second**: **8103.73**

PoC Consumer - Commit = Manual

Number of unique matched entries: **10000**
Total difference of all requests in milliseconds: 16154
Shortest response time in millisecond: 0
Longest response time in millisecond: 3
Mean/The average time a transaction takes in millisecond: **0.1116**
Standard **deviation** in milliseconds: **0.3187**
Number of entries that took longer than a second: 0
% of entries that took longer than a second: 0.00%
Average **transactions per second**: **619.04**

PoC Consumer - Commit = Auto

Number of unique matched entries: **10000**
Total difference of all requests in milliseconds: 8852
Shortest response time in millisecond: 0
Longest response time in millisecond: 3
Mean/The average time a transaction takes in millisecond: **0.1018**
Standard **deviation** in milliseconds: **0.3063**
Number of entries that took longer than a second: 0
% of entries that took longer than a second: 0.00%
Average **transactions per second**: **1129.69**

1. Partially Implemented in PI3 Master Branch (async Loggers were deferred by DA →
   a. https://github.com/mojaloop/design-authority/issues/11
   b. https://www.dropbox.com/s/cu9vam5jlqsea35/Screenshot%2020199-10-09%2012.31.55.png?dl=0
2. DA decision was to defer async logging until the auditing framework (e.g. Event Framework) was in place.
3. Sync vs Async logging can be configured by each Components & Helm chart

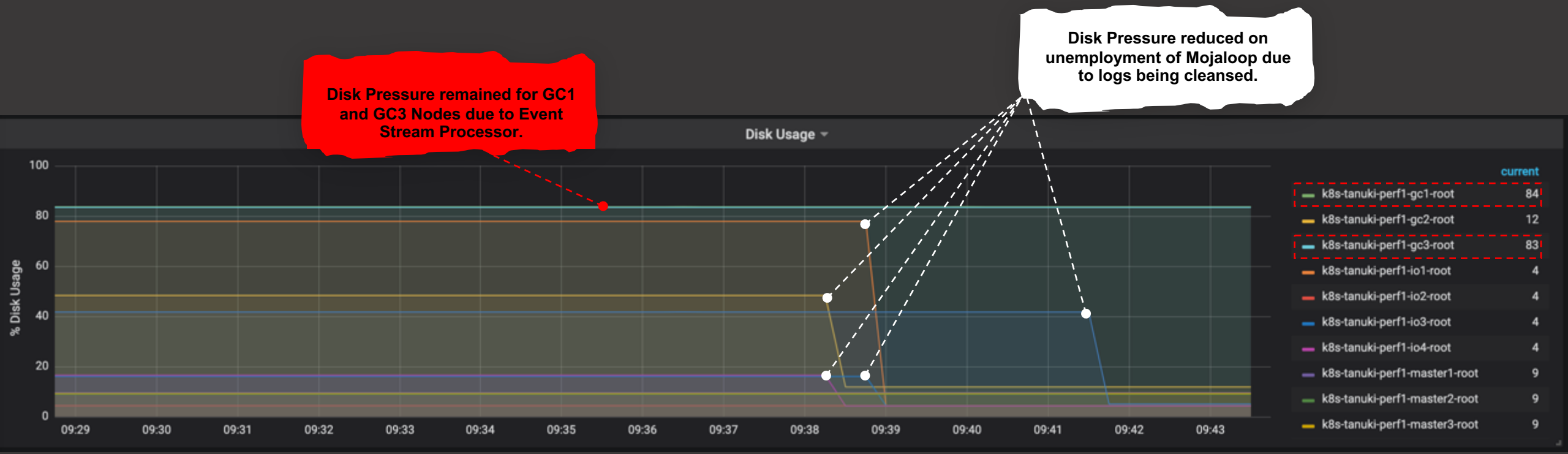
Optimization – K8S Disk Pressure

Description:

- High disk space usage on specific K8S Node due to excessive log file sizes.
- FluentD (Log collector) process experienced issues in collecting large log files
- Verbosity of Event logging identified to be the main cause

Resolution:

- Event-Sidecar: Verbosity of Event Stream logs reduce to display meta-data only.
- Event-Stream-Processor: Verbosity of Event Stream logs reduce to display meta-data only.



Optimization – General Resolved Issues

Caching of Long-Lived-Temporal Data

1. Optimization – Caching of FSP Participant Information. ← Reduction in DB QPS

SDK-Schema-Adapter

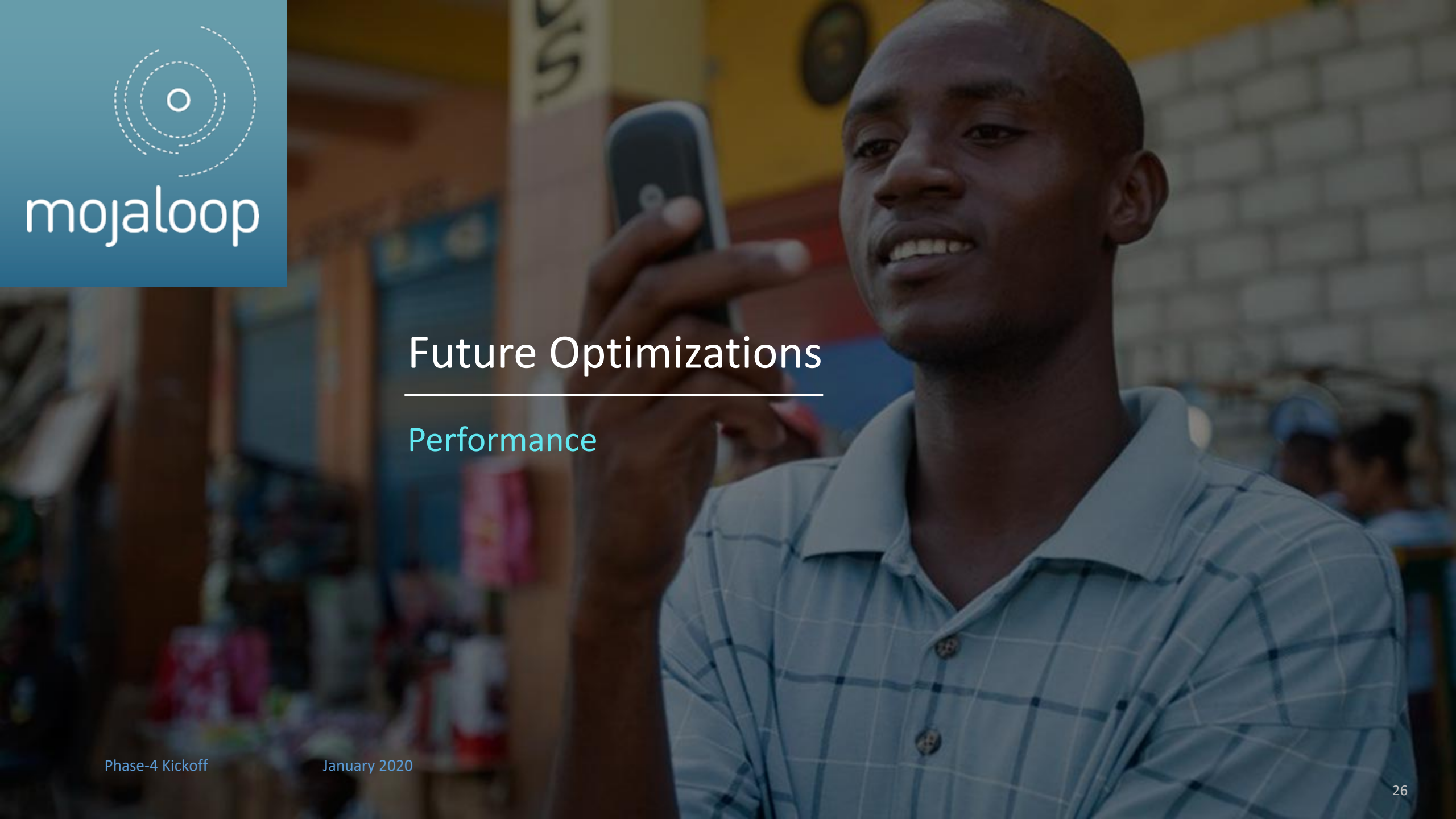
1. Stability - Connection Management for Redis Cache (https://github.com/mojaloop/sdk-scheme-adapter/pull/103)
2. Stability - Timeout subscribers failing (https://github.com/mojaloop/sdk-scheme-adapter/pull/108)
3. Stability - Redis cache data retention issue (https://redis.io/commands/expire - https://github.com/mojaloop/project/issues/1134)

Central-Services-Stream

1. Stability - Kafka Client Producers were failing with “Full Queue Error” (https://github.com/mojaloop/project/issues/1114)

Grafana Dashboard Metric Miss-match

1. Monitoring – Fixed metric calculation on JMeter Dashboard for TPS calculation (did not match ML-API rate)



Future Optimizations

Performance

Future Optimization – PRISM

Meaning:

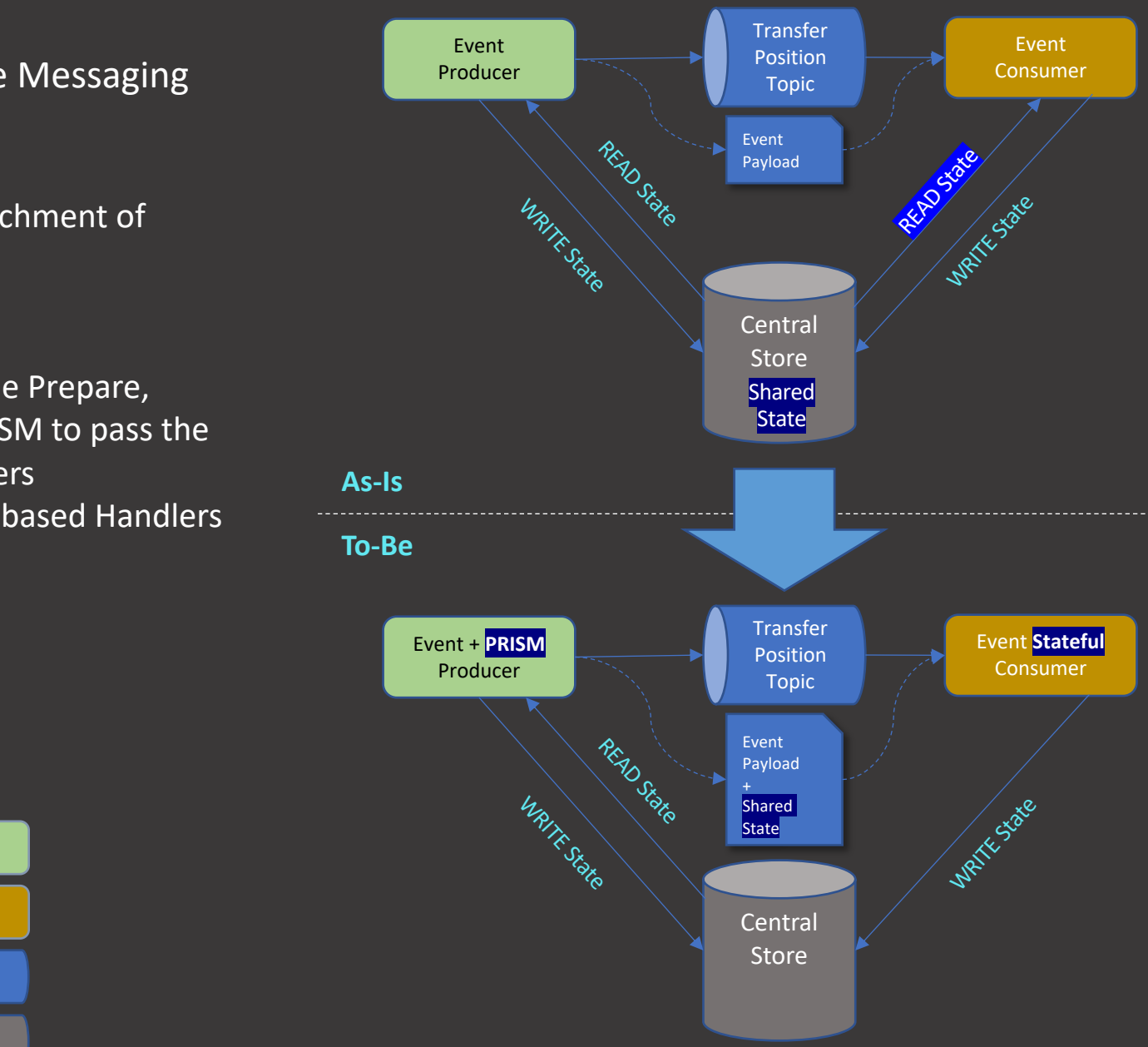
- PRISM - Persisted Replicated Internal State Messaging

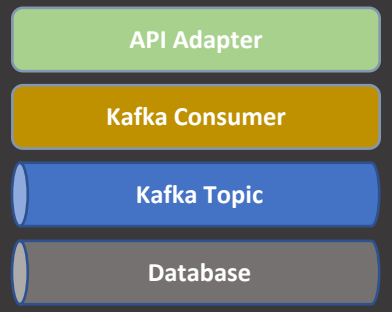
Description:

- Kafka messaging system is used for stateful enrichment of messages to down-stream handlers

Considerations:

- We can minimize unnecessary locks between the Prepare, Position, Fulfil and Timeout Handlers by using PRISM to pass the necessary information to the down-stream handlers
- General improvement to most Kafka Consumer based Handlers (e.g. Prepare, Positions, Fulfil, Notifications, etc)





Future Optimization – Position Handlers Concurrency

Objective: Compete historic state record at time of rule execution, while never breeching liquidity checks

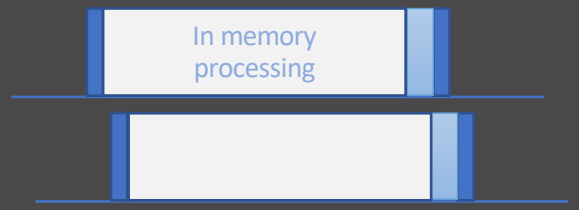
1. Position Handler Turnstile – single instance, bulk transfer set for processing

Single Position Handler

1. DFSP's Position is held in READ/UPDATE lock while processing transfer set read from topic
2. Means there can only be one active handler per DFSP

2. Position Reserved Handler – Reserved Position, multi-instances

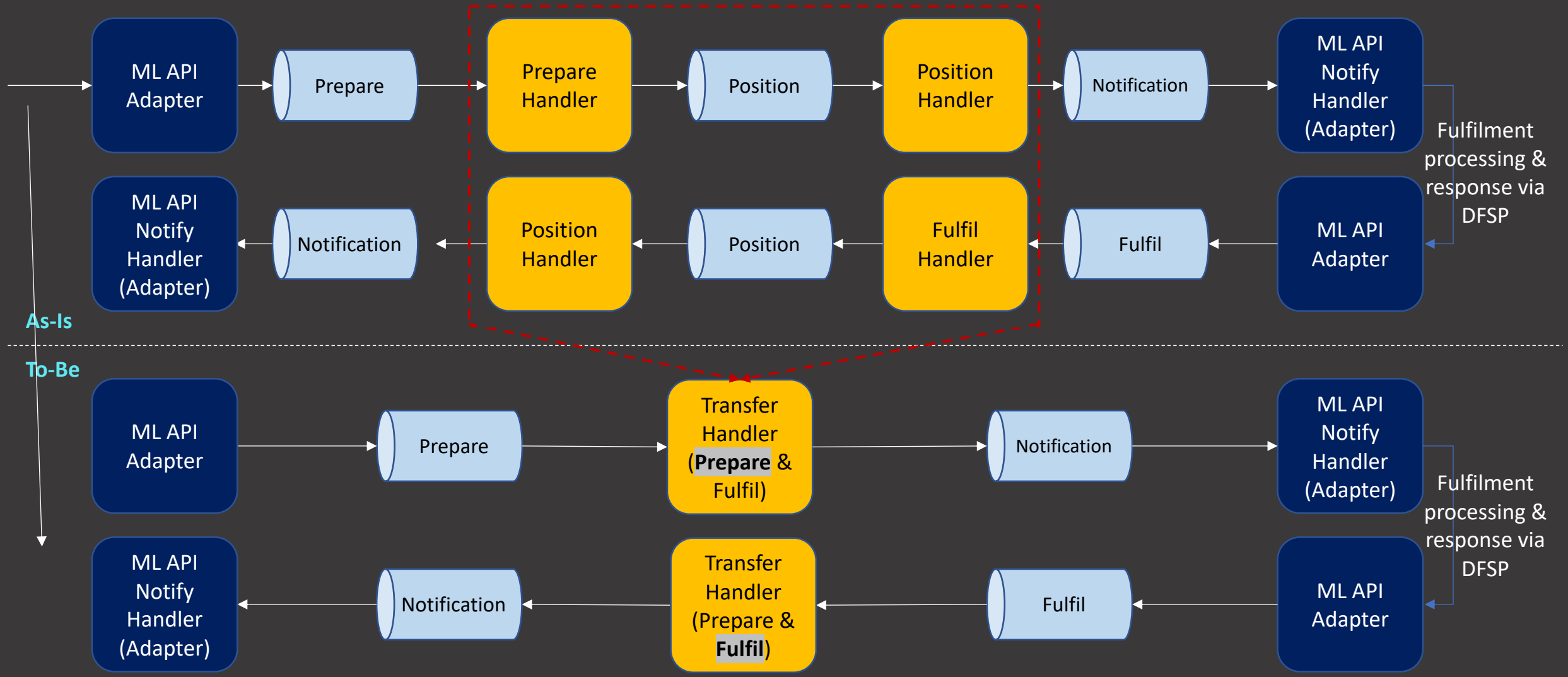
Multiple Position Handlers



Position Reserve and Position Finalise, separated with in memory processing

1. Position/Liquidity is reserved for total of transfer set
2. After in memory processing:
   a. Finalised transfer states are INSERTed, and
   b. DFSP Liquidity is finalised with UPDATE
3. Works in mixed or per DFSP processing mode
4. Multiple active handlers

Future Optimization – Consolidation of Handlers



- Reduction in DB reads/write (state kept in memory for each leg – prepare & fulfil)
- Reduction in latency (less hops over network)

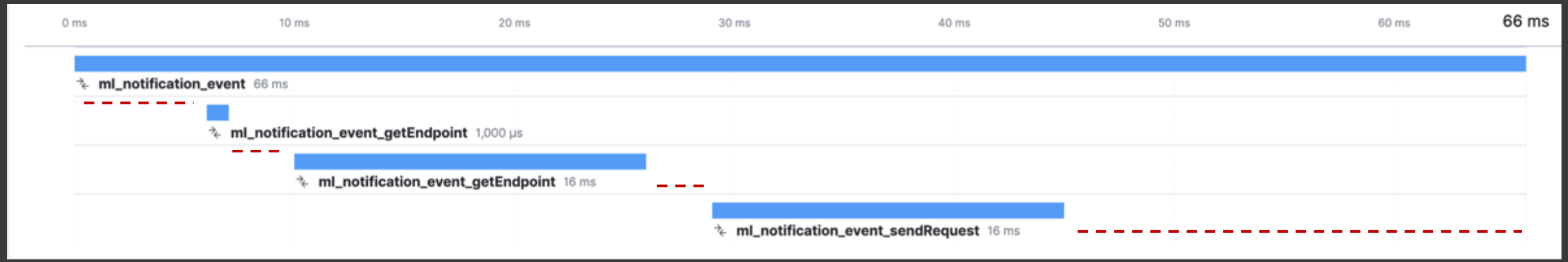
Future Optimization – Notification Handler

Description:

- Notification Handler trace results indicate that a large portion of the end-to-end transfer processing time is being spend within this component.

Potential Optimizations:

- Parallelize sendRequests – Currently the Mojaloop HTTP Requests for Transfer PUT on Fulfilment is ordered (PayeeFSP followed by PayerFSP). This assumes ordering of the requests are not important.
- Async sendRequests – We can avoid the blocking IO of the sendRequests if we do not wish to utilize the kafka rollback mechanism for replays. This can be rather done as part of the sendRequest function itself thereby improving the efficiency of processing.



Future Optimization – Timeout Handler

Current processing of timeouts:

1. Retrieve transferStateChangeId up to which transfers have been processed during previous run (segment table)
2. Cleanup the list of expiring transfers (transferTimeout table) from finilised transfers [reads transferStateChange]
3. Determine current max transferStateChangeId up to which the current run will proceed (transferStateChange) [read]
4. Populate transferTimeout table with newly added transfers, which are not finilised [reads transferStateChange]
5. At this point transferTimeout holds records only for expiring transfers with their respective expiriationDate
6. Timeout transfers for which funds where not yet reserved by inserting multiple records in transferStateChange-EXPIRED_PREPARED [reads transferStateChange]
7. Mark for timeout transfer with reserved finds by inserting multiple records in transferStateChange-RESERVED_TIMEOUT [reads transferStateChange]
8. Store current max transferStateChangeId to be used for the next run (segment table)
9. Read extended info for all expired transfers, including latest state changes [reads transferStateChange]
10. Queue individual message for each transfer in the previous list to NotificationHandler/PostionHandler
11. For each transfer, PositionHandler reads from db info to change position [reads transferStateChange]
12. For each transfer, PositionHandler updates the position, writes transferStateChange and queues notification message
13. This process is currently configured to repeat every 15s

Pitfalls:

1. Too many multi record reads during TimeoutHandler from transferStateChange (6)
2. Two bulk inserts in transferStateChange happen inside a transaction during TimeoutHandler processing
3. PositionHandler updates position and inserts transferStateChange using transaction for each expiring transfer

Disadvantages:

1. Possible table level locks for transferStateChange
2. TimeoutHandler can not be scaled

Proposed Enhancement:

1. Create TimeoutPrepareHandler which does not use segment and transferTimeout tables, but rather a new table expiringTransfer (expiringTransferId, transferId, expirationDate, createdDate) indexed by expirationDate
2. PrepareHandler will create a new record in expiringTransfer for each incoming transfer
3. FulfilHandler will not delete records from expiringTransfer, but it is an option that might be considered
4. Upon run, TimeoutPrepareHandler will first delete all records for which a matching transferId exists in tranfserFulfilment and then select using index all transfers for which expirationDate < currentTimestamp. For each transfer in the selected set a message will be produced.
5. Create TimeoutProcessingHandler that will query current transfer state and
   a. Stop processing if transfer is finalized (and delete expiringTransfer entry), or
   b. Insert transferStateChange-EXPIRED_PREPARED and produce message to notification-topic (-expiringTransfer), or
   c. Insert transferStateChange-RESERVED_TIMEOUT and produce message to position-topic

In case of 5c PositionHandler processes the message as with the current TimeoutHandler.
6. PositionHandler to clear processed exprintTransfer etnry

Advantages:

1. Reads from transferStateChange are per transfer
2. TimeoutProcessingHandler could be scaled when necessary

Disadvantages:

1. One extra insert during PrepareHandler processing, but expirationDate index maintenance should not be heavy as records get cleaned from it by TimeoutPrepareHandler

Stories:

1. https://github.com/mojaloop/project/issues/1089

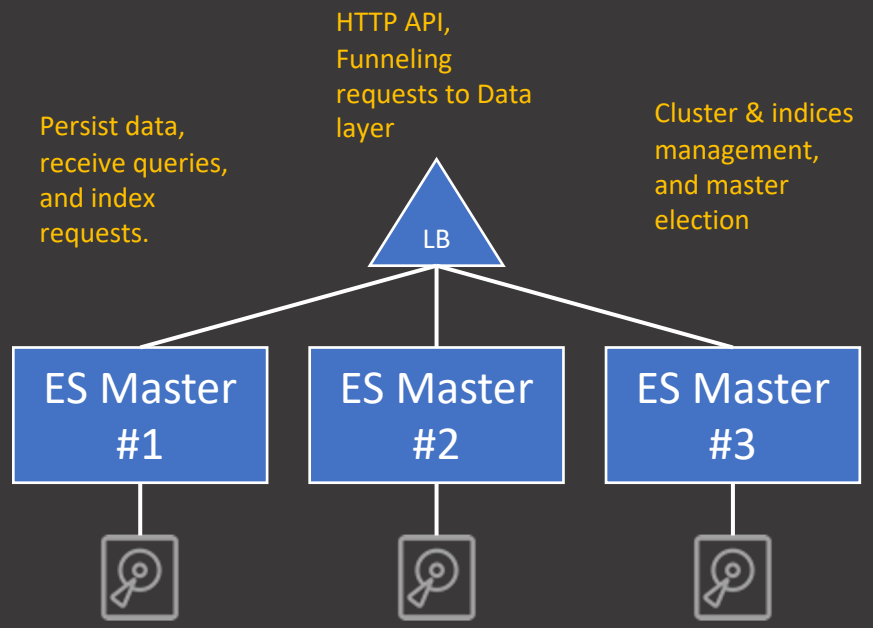
Future Optimization – ElasticSearch Data Ingestion

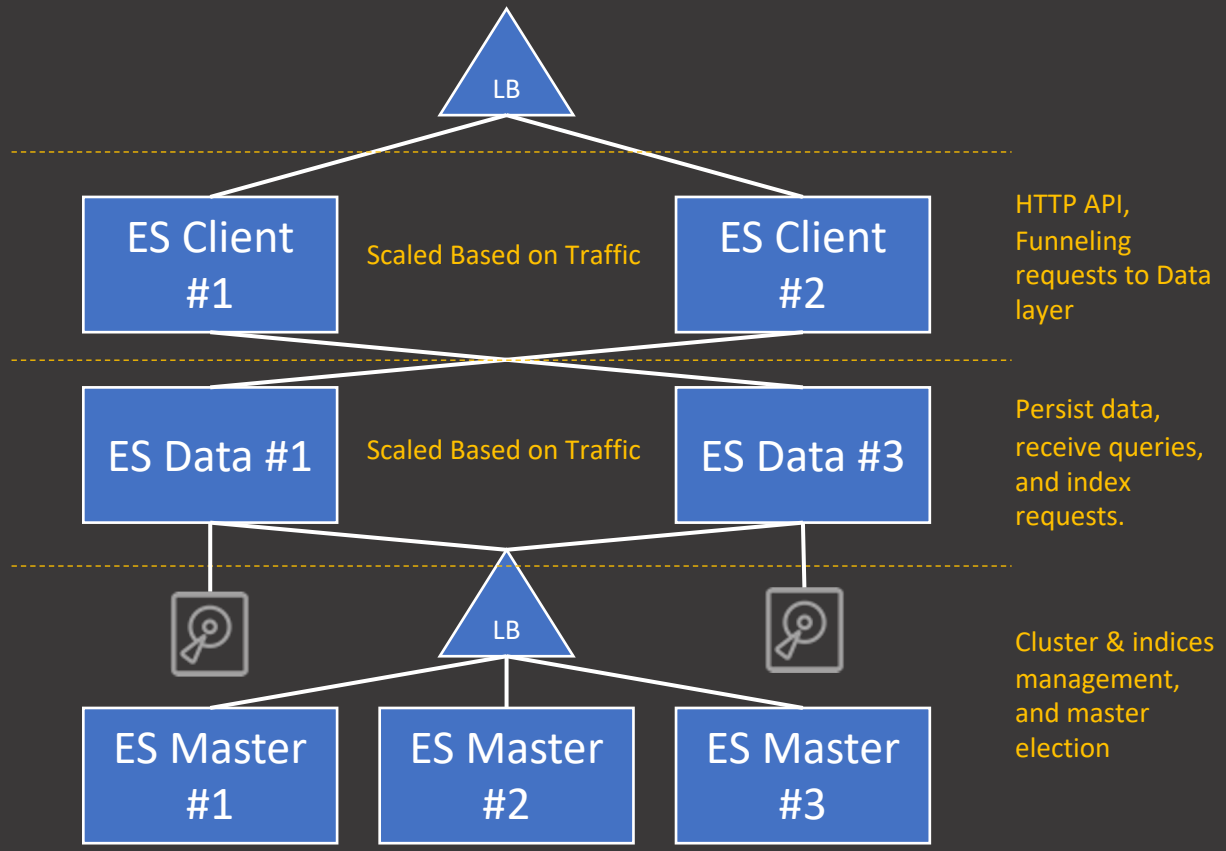
Description:

- ElasticSearch is unable to ingest the amount of data being produced by the logs and event stream.

Resolution:

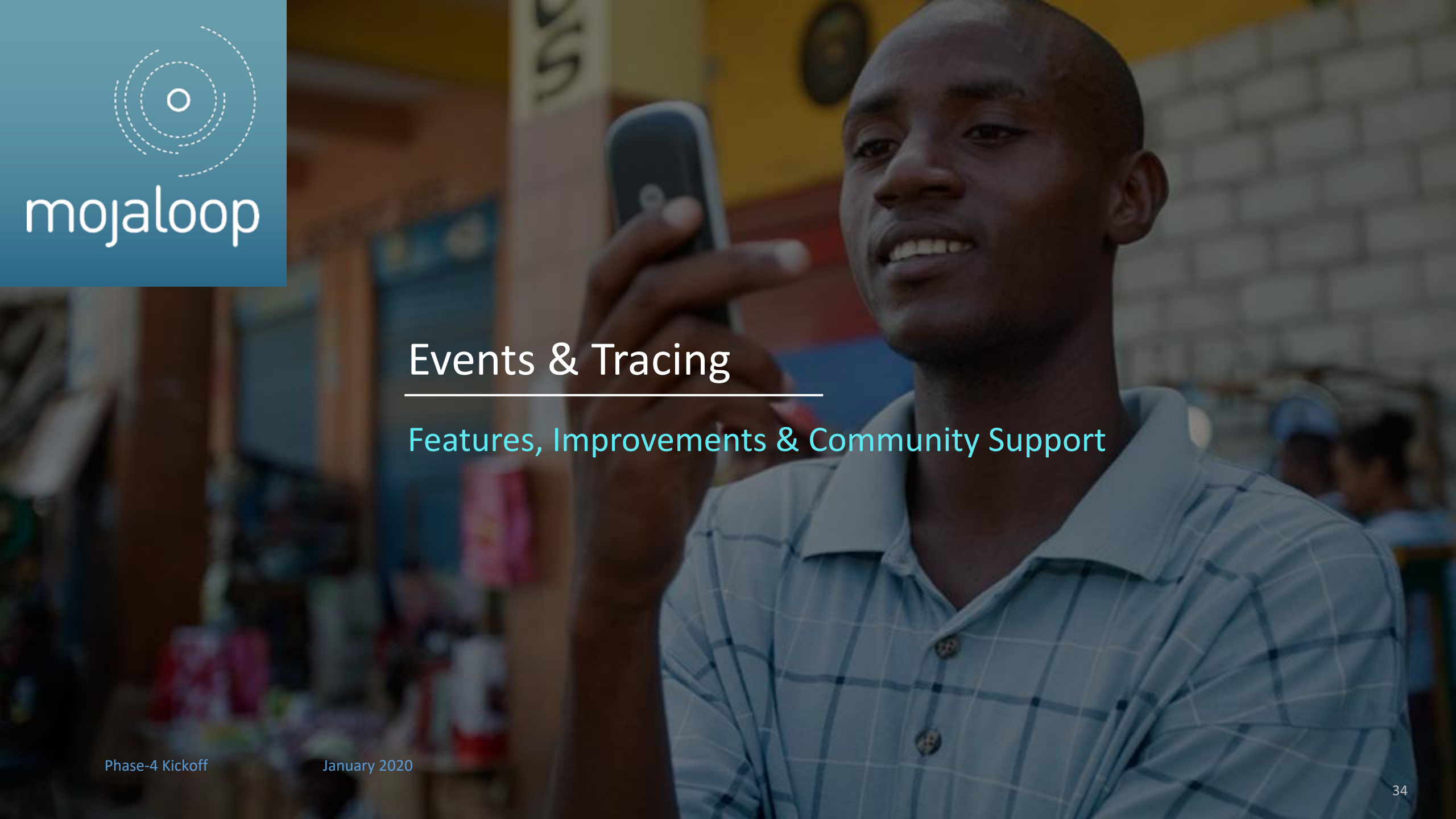
- Event-stream-processor – Configuration to throttle event stream processing
- ElasticSearch - Deployment to be optimized into layered topology to support scaling requirements





Mojaloop Phase-4

Going Live!



Events & Tracing

Features, Improvements & Community Support

Event Monitoring Framework

1. Overview
2. Functionality
3. SDK Examples
4. Roadmap
5. Demo

Event Monitoring Framework – Overview

What

1. Unified framework to capture all Mojaloop events and ingest them appropriately
2. Event Types:
   a. Logs
   b. Errors
   c. Audits
   d. Traces

Why

1. Operational monitoring of requests end-to-end
   a. End-to-end request visualization
   b. Enabler for alerts
   c. Issue resolution
2. Enabler for auditing and fraud management

How

1. Standardized framework for capturing events (types, actions, metadata, etc)
2. Every request is given a trace-id at the boundary.
3. Standard common (Event-SDK) library that will publish events to a sidecar component utilising a light-weight highly performant protocol (*e.g. gRPC*)
4. Sidecar module will publish to a Kafka messaging stream for all events utilising Event-SDK
5. Each Mojaloop component will have its own tightly coupled Sidecar
6. Leverage on open source standards and solutions where possible (*e.g. APM, OpenTracing, Zipkin, etc*)

Event Framework – Functionality

REFERENCE

Event SDK Features

1. Spans
   1. Create new span
   2. Create child spans
2. Event Types:
   1. Logs (info, debug, error, performance, warning, verbose)
   2. Audit (default, start, finish, ingress, egress)
   3. Traces (Spans)
3. Client/Server:
   1. gRPC client
   2. gRPC server
4. Recorders:
   1. Async gRPC (event-sidecar)
   2. Sync gRPC (event-sidecar)
   3. Logger (Own Logger Instance)
5. Supports arbitrary tags
   1. Add additional information as part of the trace context (e.g. transferId)
6. Context
   1. Inject Context into Message
   2. Extract Context from Message
   3. Create Child Spans from Context
   4. Support for Messaging (e.g. Kafka) & HTTP Transports (WC3 Tracing *standard)
   5. Support for ”traceState” header from WC3 Tracing *Standard

* https://www.w3.org/TR/trace-context-1/

Event Sidecar

1. Publishes Messages to Kafka Event Topic

Event Stream Processor

1. Record **logs** & **audits** to **EFK**
2. Record **traces** to APM server

Helm

1. Helm Charts with Event-sidecar support:
   a. ML-API-Adapter
   b. Central-Ledger
   c. Quoting-Service
   d. SDK-Schema-Adapter
   e. Legacy Simulator

Improvements in PI8

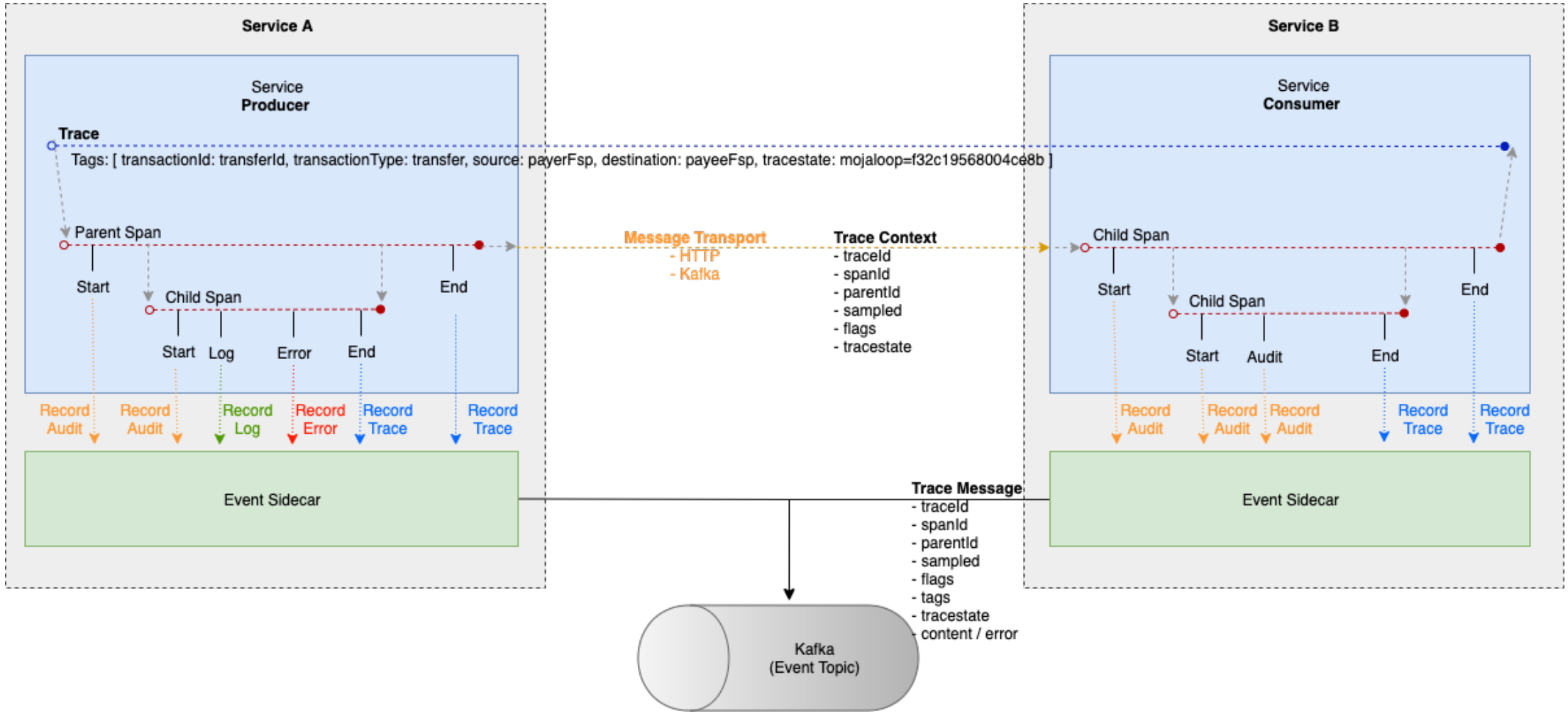
1. Enhanced WC3 Tracing Standard Support
   a. Tracestate header
2. General
   a. Sync - Async configuration for Event levels (trace, audit, log)
   b. Precise control over logging (enable/disable, local/remote & metadata)
   c. Improved stability

Future Roadmap

1. Roll-out
   a. Account-Lookup-Service
2. Performance testing
   a. What is the impact of enabling Tracing
   b. What is the optimum/acceptable level of tracing/logging

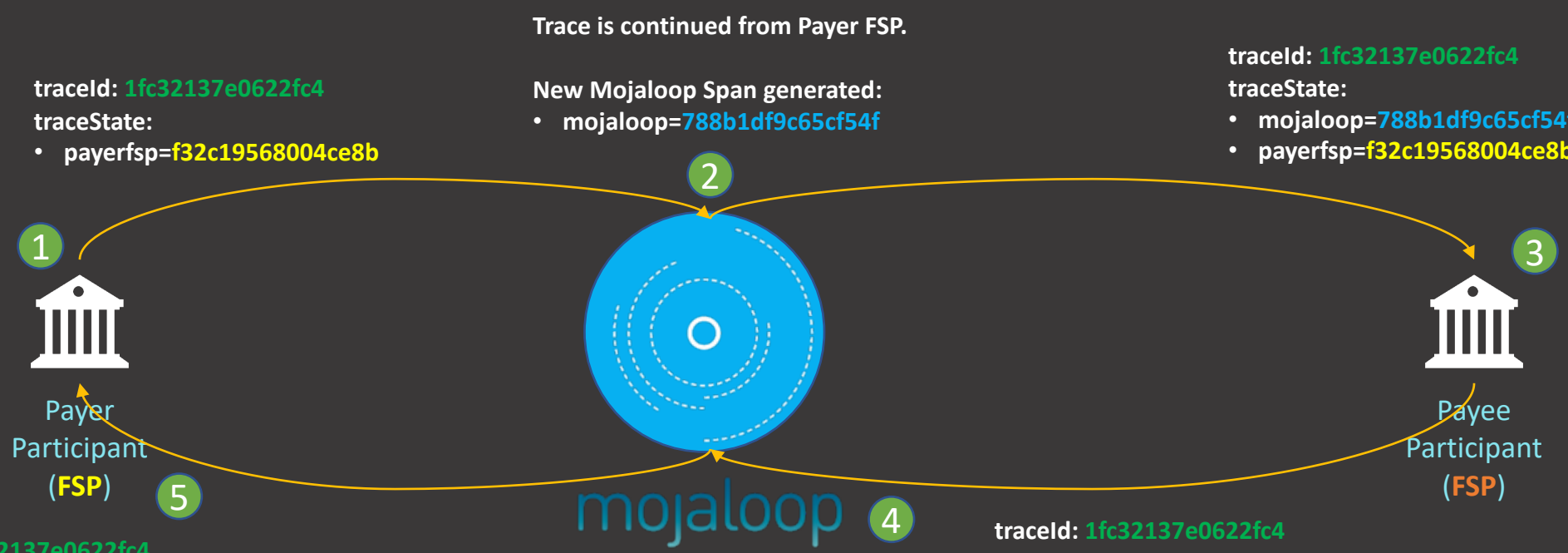
Event Monitoring Framework – Trace Architecture

REFERENCE



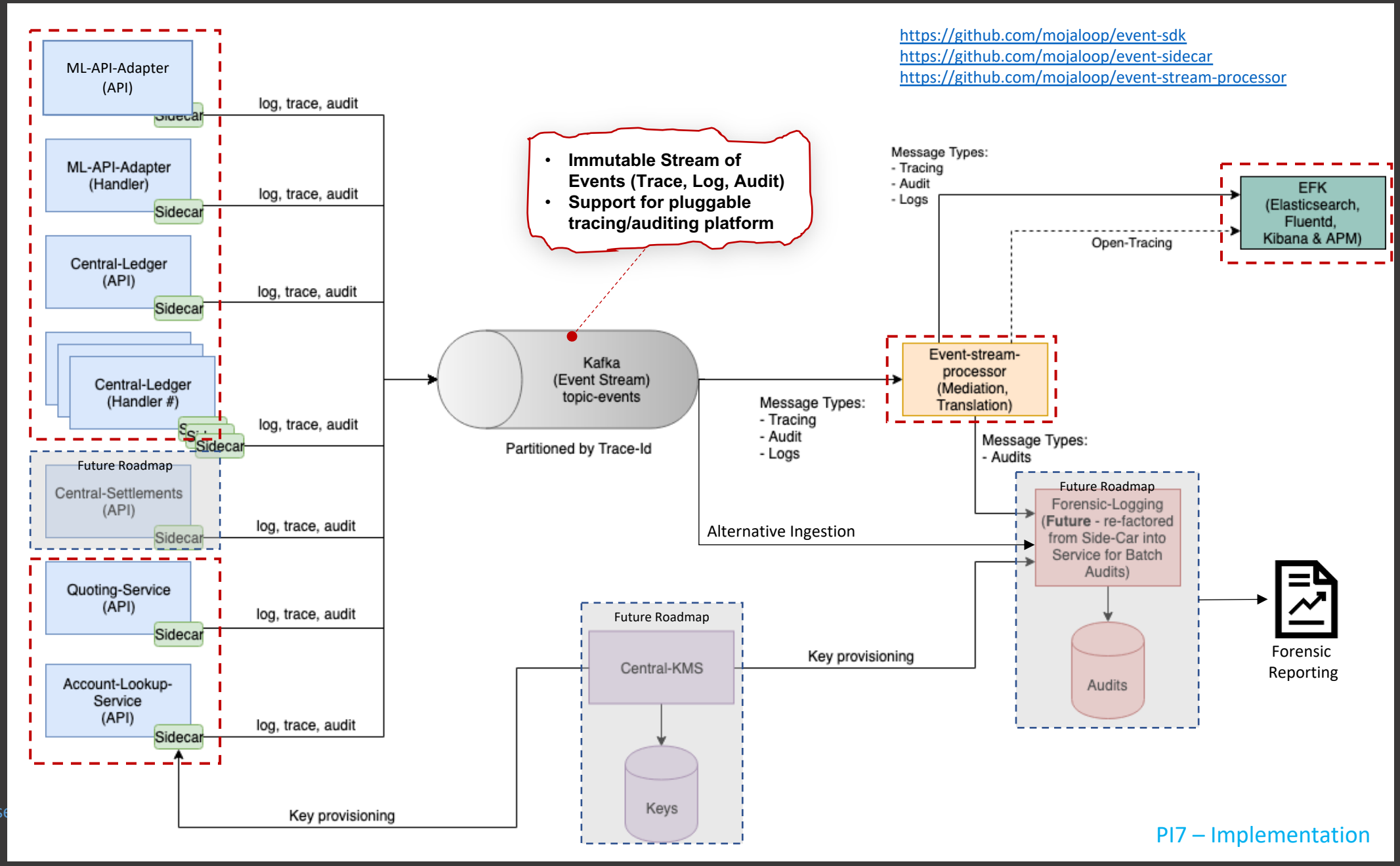
Event Flow of Events

Tracing Enhancement – Federated Tracing



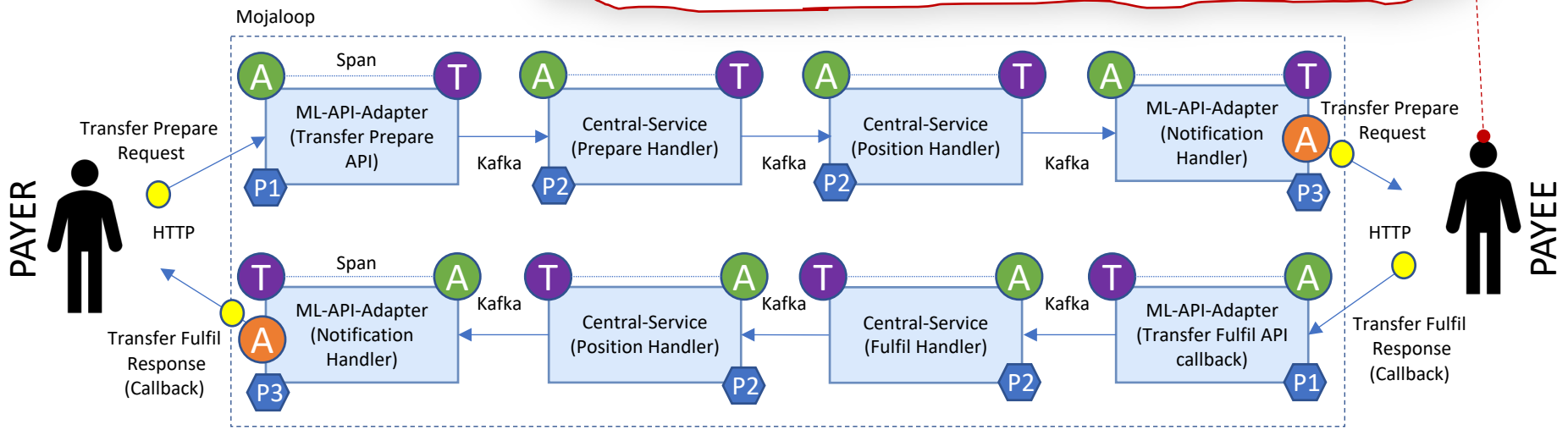
REF: https://www.w3.org/TR/trace-context-1/#relationship-between-the-headers

Event Framework – Current Functional Overview



Event Monitoring Framework Use-Case Example – Success End-to-end Scenario

- Optional support for FSPs to propagate/partake in Tracing via **Wc3 Trace Headers**
- Support for **Federated Tracing** (3rd Party Vendors) via "**tracestate**" header
- Required for "end-to-end" trace **or** Prepare and Fulfil legs will be separate Traces



Event Key

- **Audit** - Start
- **Audit** - Egress
- **Trace** - Finish Span
- **Log** - Error
- **Trace-State:** Vendor Trace propagated

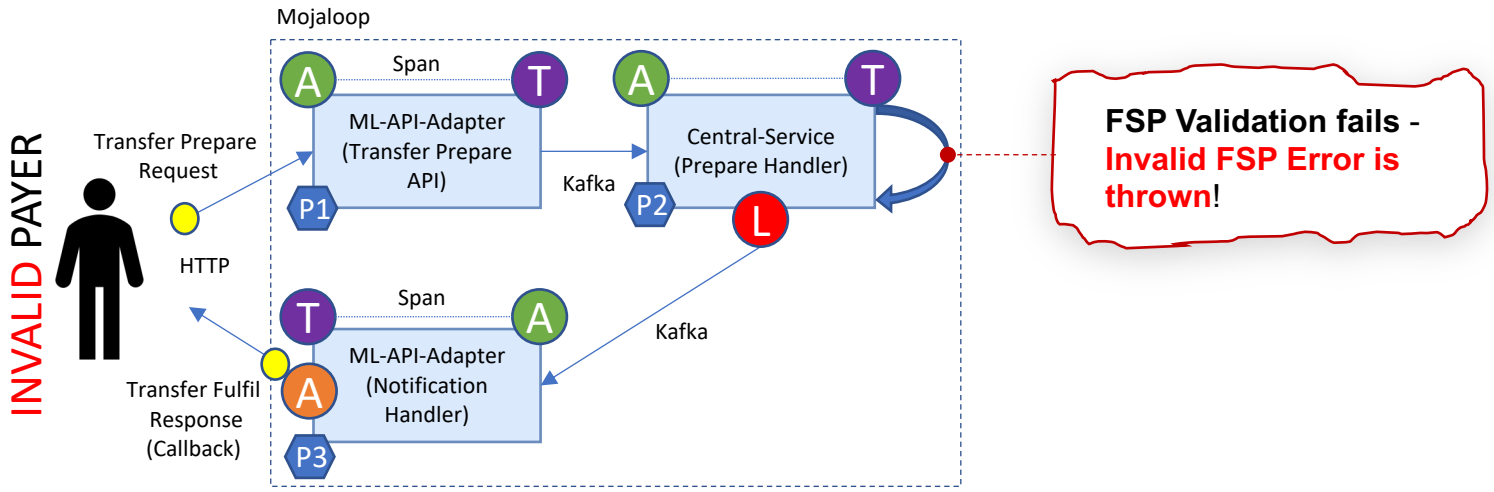
Processing Key

P1
- **Trace** is **created** with an initial Span or **continued** if **WC3 Trace Headers** are found
- **HapiJS plugin** provides consistent mechanism to manage incoming Traces
- **Trace-Context injected** into Message

P2
- **Trace-Context extracted** from message
- **Child span created**
- **Trace-Context injected** into Message

P3
- **Trace-context** is **injected** into **WC3 Trace Headers**
- **HapiJS plugin** provides consistent mechanism to manage outgoing Traces

Event Monitoring Framework Use-Case Example – Failed End-to-end Scenario



Event Key

- **Audit** - Start
- **Audit** - Egress
- **Trace** - Finish Span
- **Log** - Error
- **Trace-State:** Vendor Trace propagated

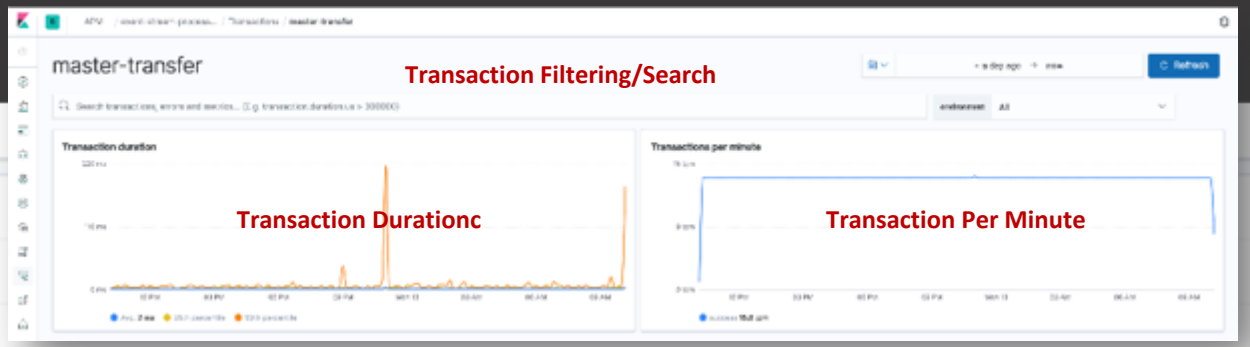
Processing Key

P1
- **Trace** is **created** with an initial Span or **continued** if **WC3 Trace Headers** are found
- **HapiJS plugin** provides consistent mechanism to manage incoming Traces
- **Trace-Context injected** into Message

P2
- **Trace-Context extracted** from message
- **Child span created**
- **Trace-Context injected** into Message

P3
- **Trace-context** is **injected** into **WC3 Trace Headers**
- **HapiJS plugin** provides consistent mechanism to manage outgoing Traces

Event Monitoring Framework – Dashboard Sample



APM / event-stream-process... / Transactions / master-transfer

Transactions duration distribution

24000 trans.
12000 trans.
0 trans.
Transaction Duration Distribution
0 ms
50 ms
100 ms
150 ms
200 ms
250 ms

Transaction sample

Actions
View full trace

Timestamp
2 minutes ago (January 13th 2020, 10:17:56.231)

URL
N/A

Duration	% of trace	Result	Errors	User ID
259 ms	100.0%	success	None	N/A

Timeline | Metadata

Transaction Visualization

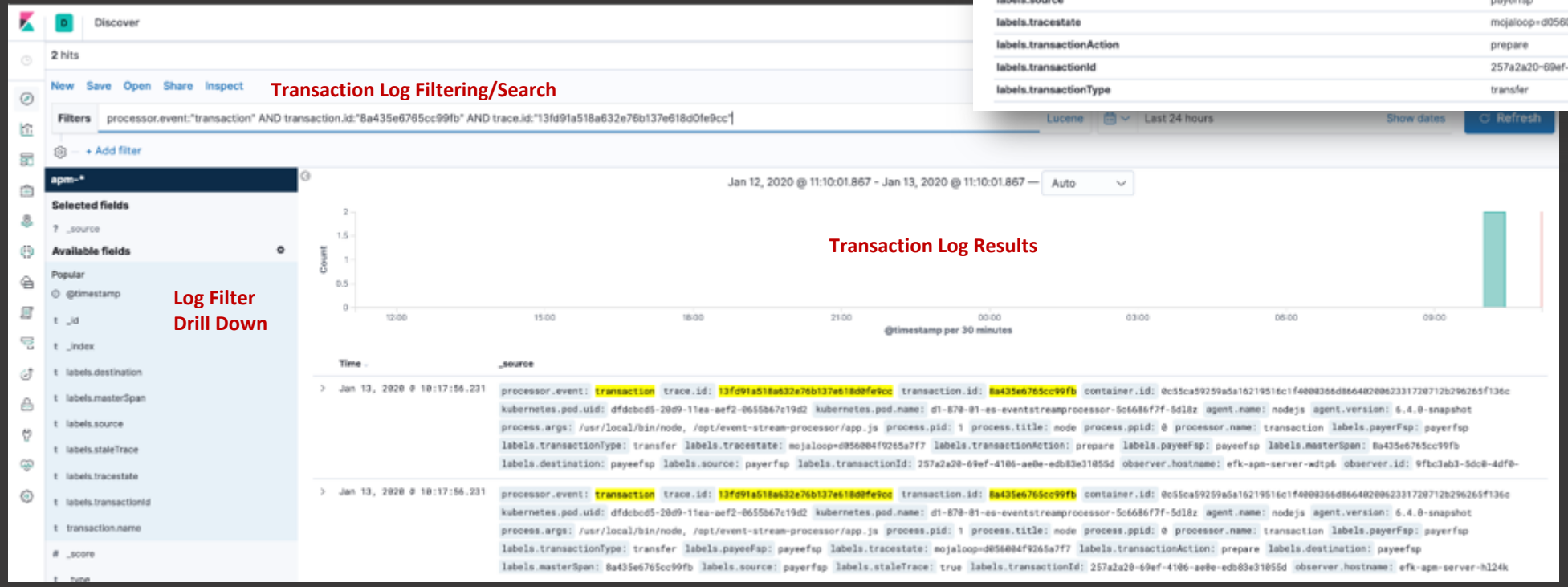
Services ● event-stream-processor

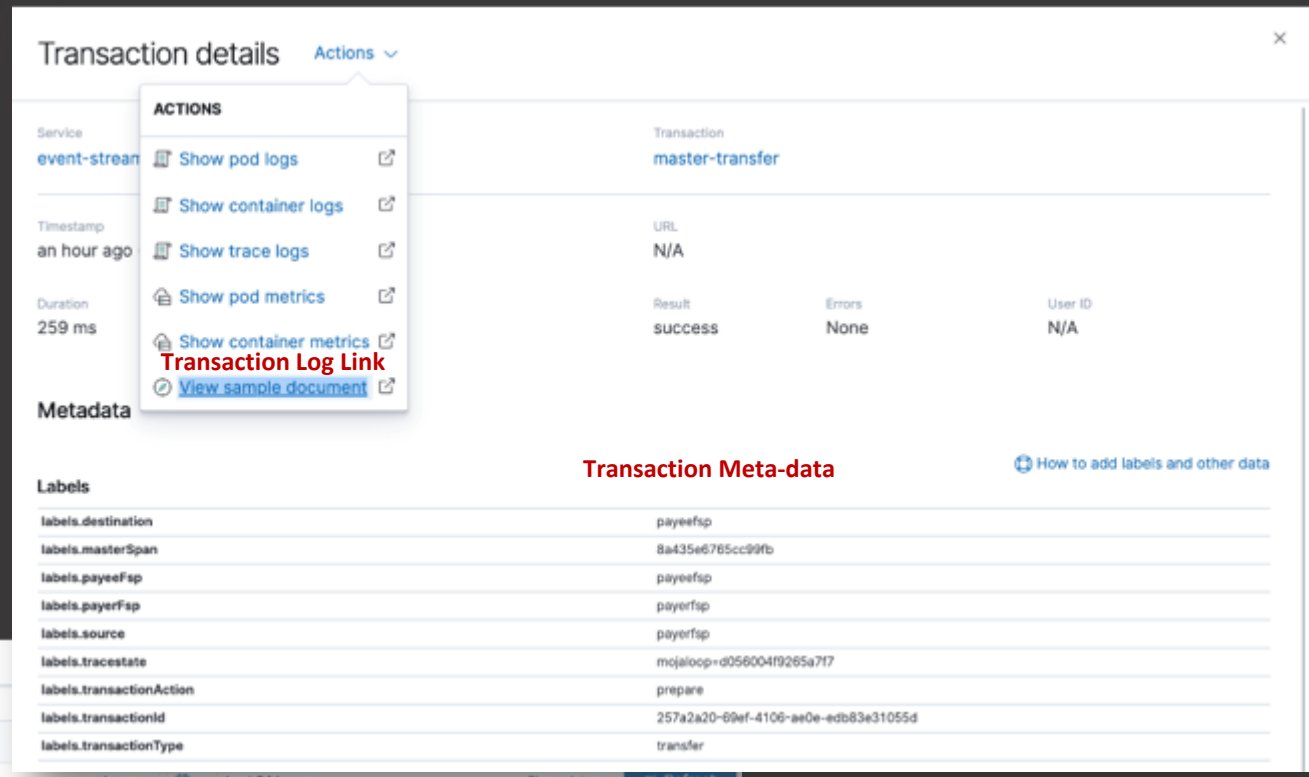
0 ms | 50 ms | 100 ms | 150 ms | 200 ms | 259 ms

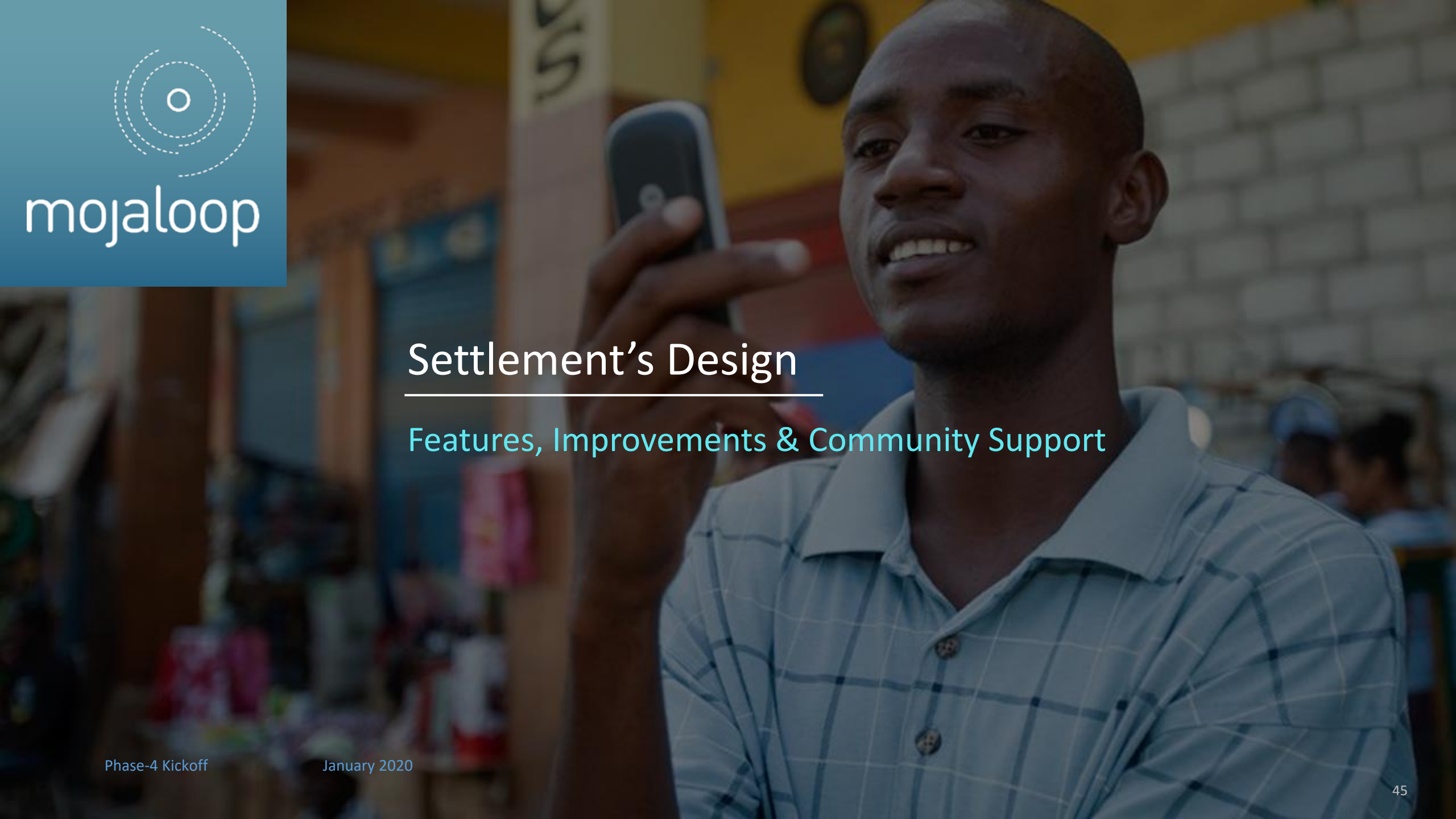
- master-transfer 259 ms
- ml_transfer_prepare 9,000 µs
- cl_transfer_prepare 26 ms
- span
- cl_transfer_position 56 ms
- ml_notification_event 65 ms
- ml_transfer_fulfil 8,000 µs
- cl_transfer_fulfil 29 ms
- cl_transfer_position 16 ms
- ml_notification_event 43 ms

Event Monitoring Framework – Dashboard Sample (Continued)

Direct link from Tracing Dashboard to Processing & Tracing logs







Settlement's Design

Features, Improvements & Community Support

OSS Settlement FSD Outline

Available at Mojaloop Documentation

- https://mojaloop.io/documentation/mojaloop-technical-overview/central-settlements/oss-settlement-fsd.html

1. Definition of terms: ledger types, states, models
2. Business rules: how it is working currently?
3. Proposed enhancements
    1. Request settlement by currency
    2. Support continuous gross settlement
    3. Processing interchange fees

Defining scheme settlement models

1. Choosing Granularity: GROSS, NET
2. Interchange: BILATERAL, MULTILATERAL
3. Delay: IMMEDIATE, DEFERRED
4. Currency: NULLABLE
5. Ledger account type: POSITION, INTERCHANGE_FEE, etc.
6. Model name
7. Usage: during create settlement event

Settlement windows handling

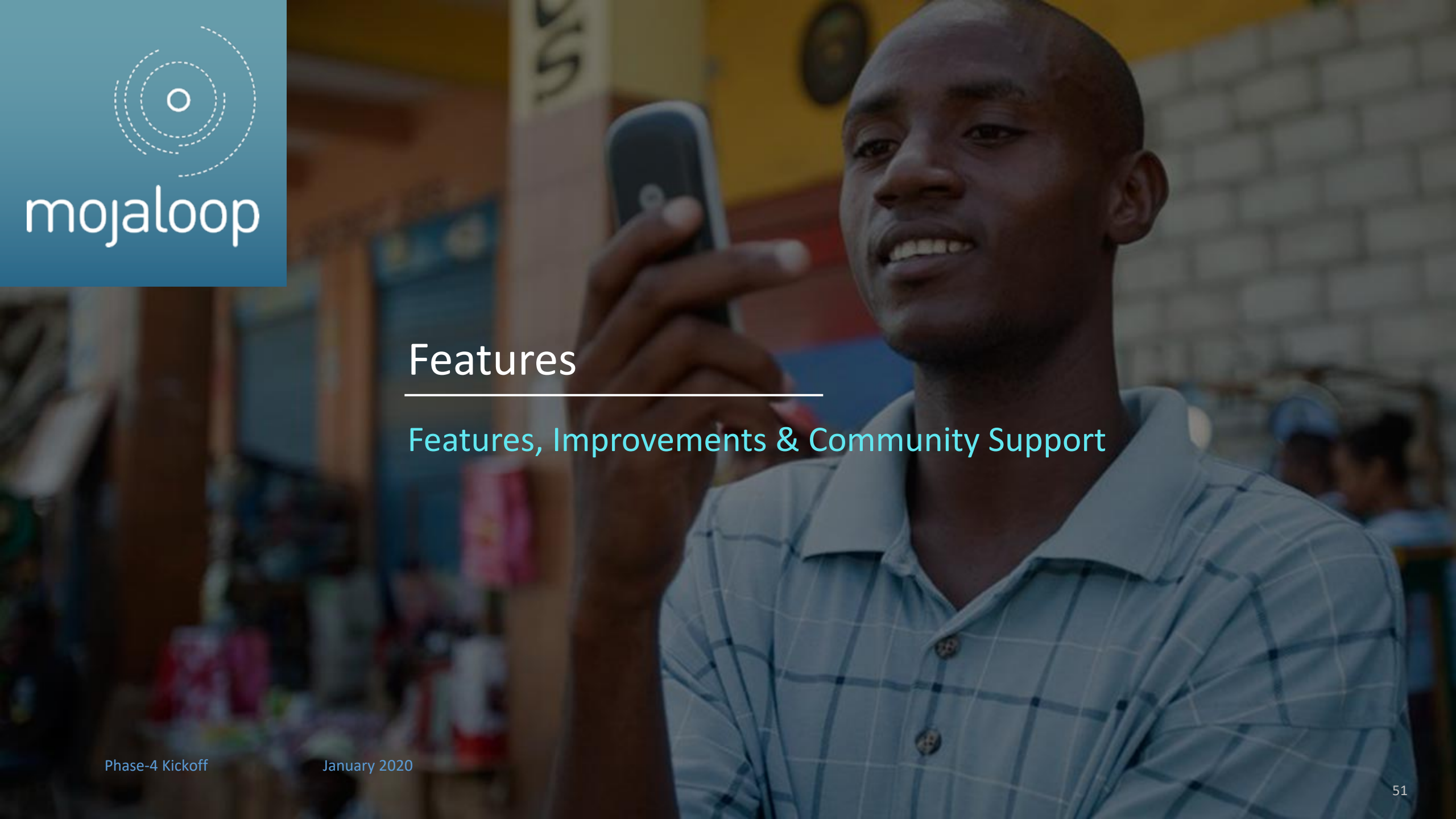
1. Current state transitions: OPEN-CLOSED
2. New state dynamics: OPEN-PROCESSING-CLOSED
3. Generation of window content and aggregations
4. Asynchronous processing and retries
5. Settlement windows state management and reporting

Working Example - JSON request/response

1. Setup
    1. Two DFSPs with POSITION accounts in USD and TZS
    2. Two transfers for the amount of 1 in each currency in one window
    3. Closed window and create a settlement
2. Scenarios
    1. Current master implementation
    2. Settlement-v2: DEFERRED_NET_USD
    3. Settlement-v2: DEFERRED_NET

Settlement v2 Roadmap

1. Descrete settlement by currency and account type
2. Continuous gross settlement
3. Processing of interchange fees
4. Settlement transfer and automatic reset of positions
5. Quality assurance and OSS standards
6. V2 migration strategy

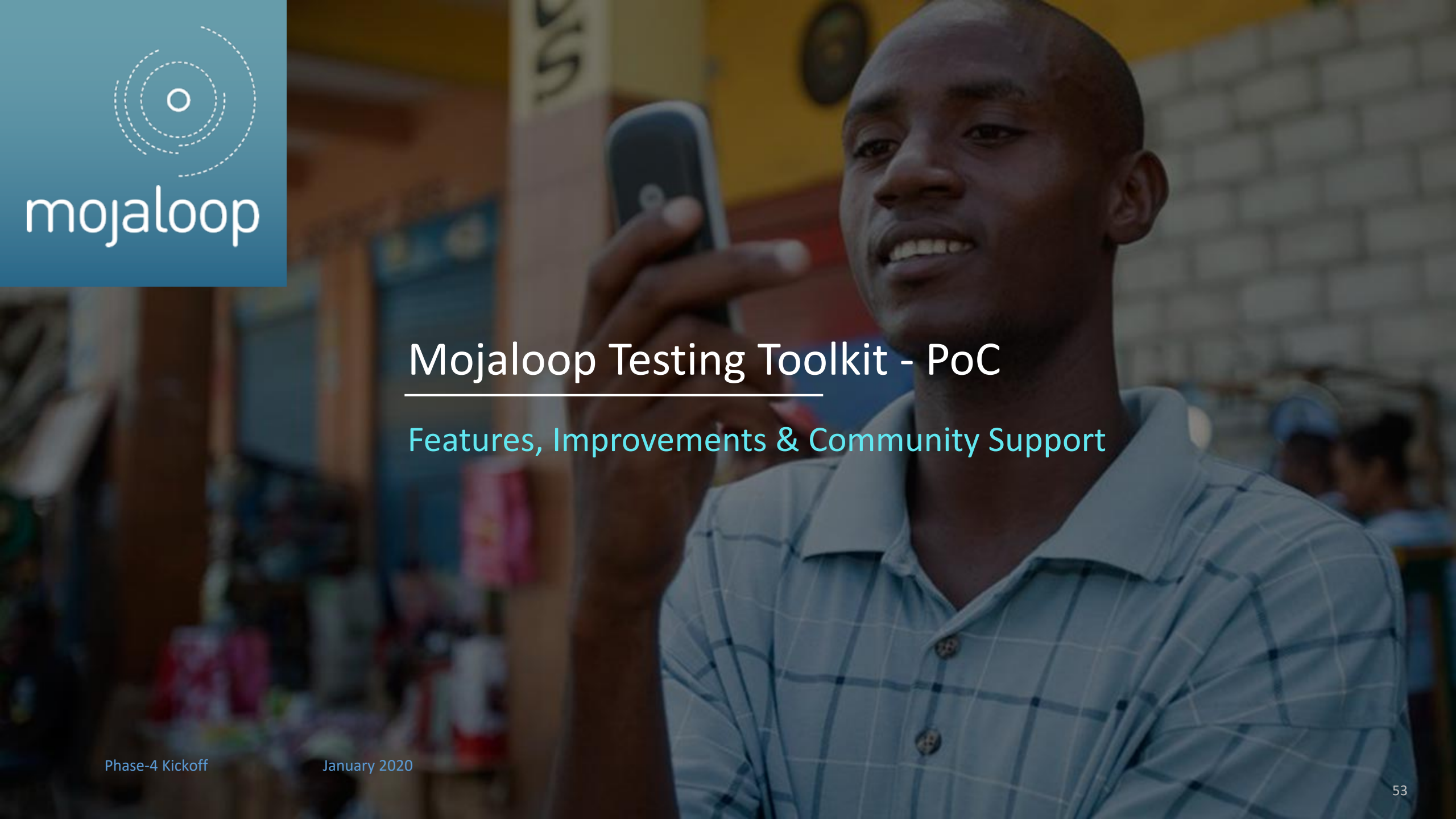


Features

Features, Improvements & Community Support

PI8 Update: Other features

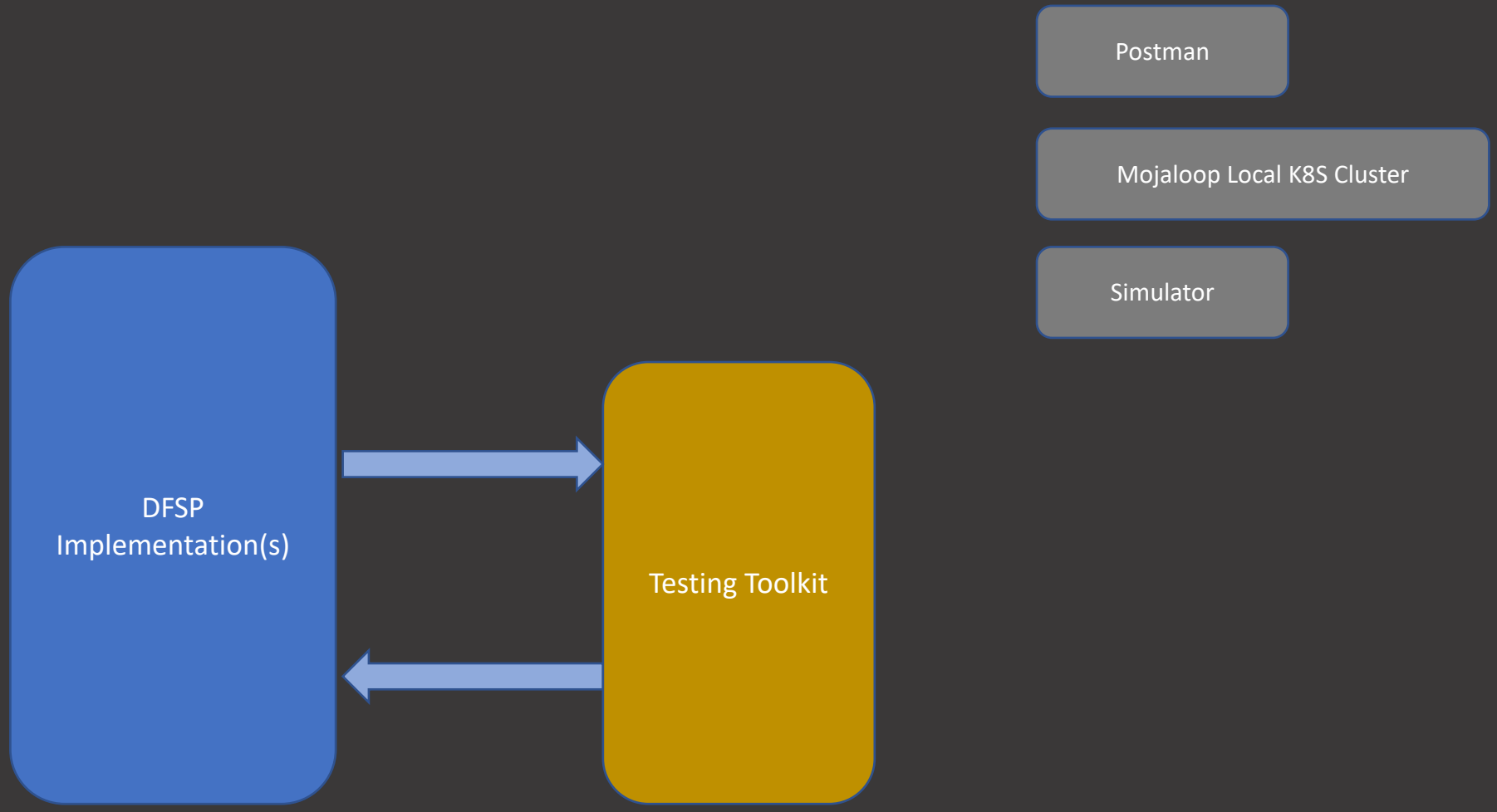
1. ALS SubId functionality
2. Quoting service standardization (from snapshot to release)
    a. Automated tests, coverage
    b. Fixing known issues
    c. Functional (postman) tests
3. Transaction Requests service
    a. Automated tests, coverage
    b. Postman tests
    c. Helm charts
4. Bulk transfers:
    1. Duplicate handling added
    2. Bug fixes



Mojaloop Testing Toolkit - PoC

Features, Improvements & Community Support

What is the Testing Toolkit



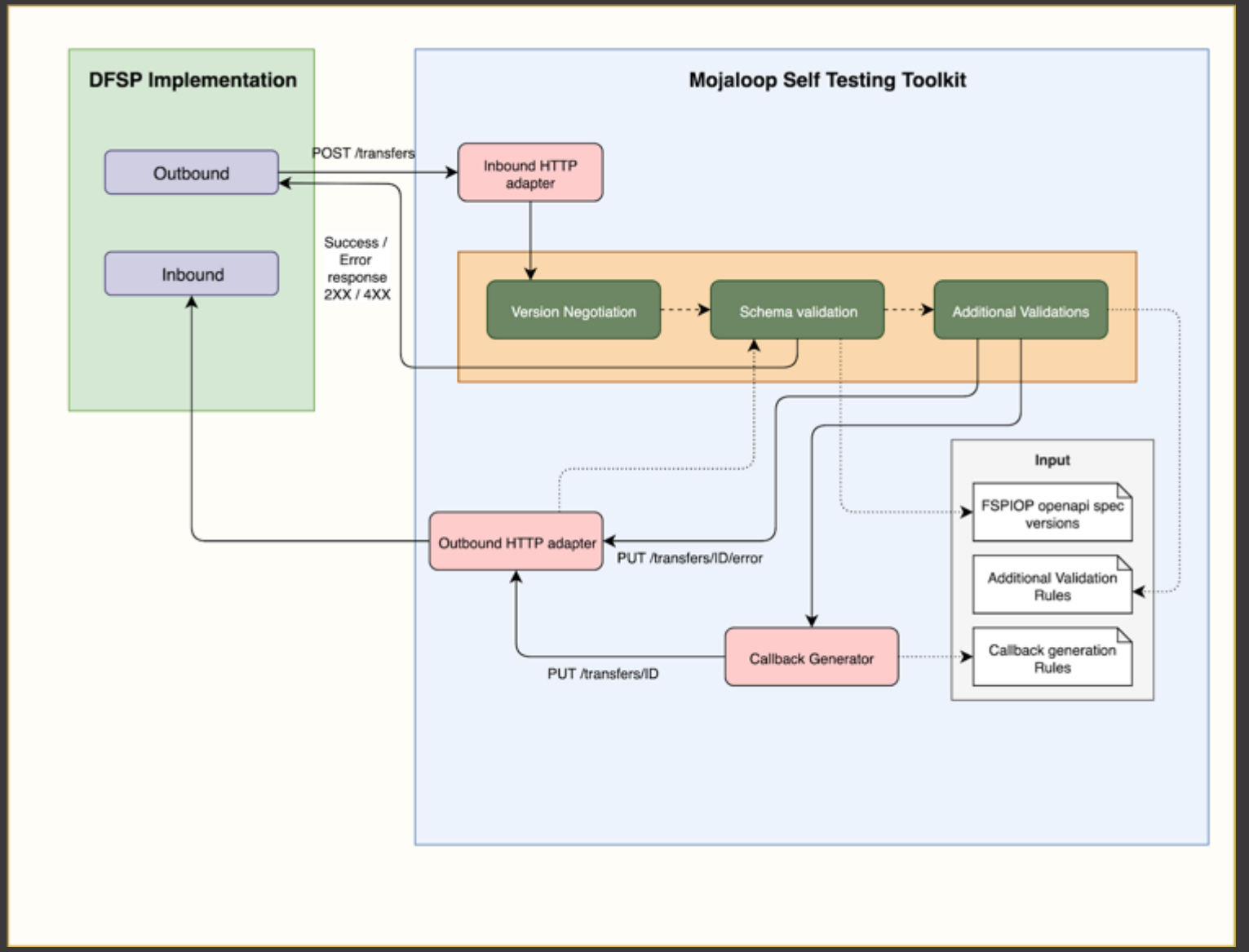
Testing toolkit: Goals

1. Test the Mojaloop API implementations - PoC
2. Simple to use
3. Support different versions of Mojaloop API
4. Highly configurable (Configurations portable)
5. Can validate Inbound requests
6. Can generate Outbound requests

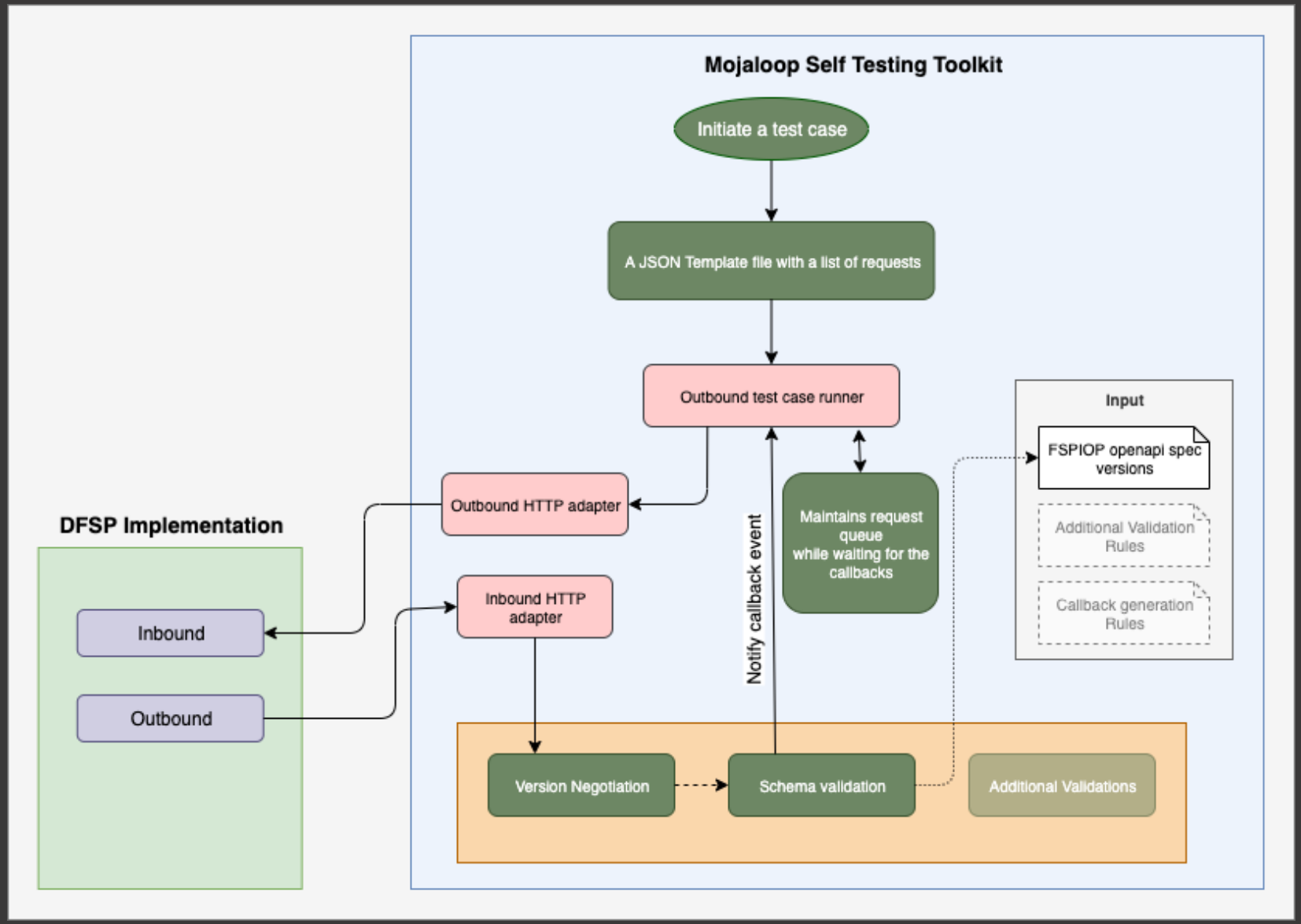
Features

1. Version validation and negotiation
2. Schema validation
3. Additional validation and error callback generation
4. Dynamic callback generation based on rules
5. Initiation of use cases (outbound)

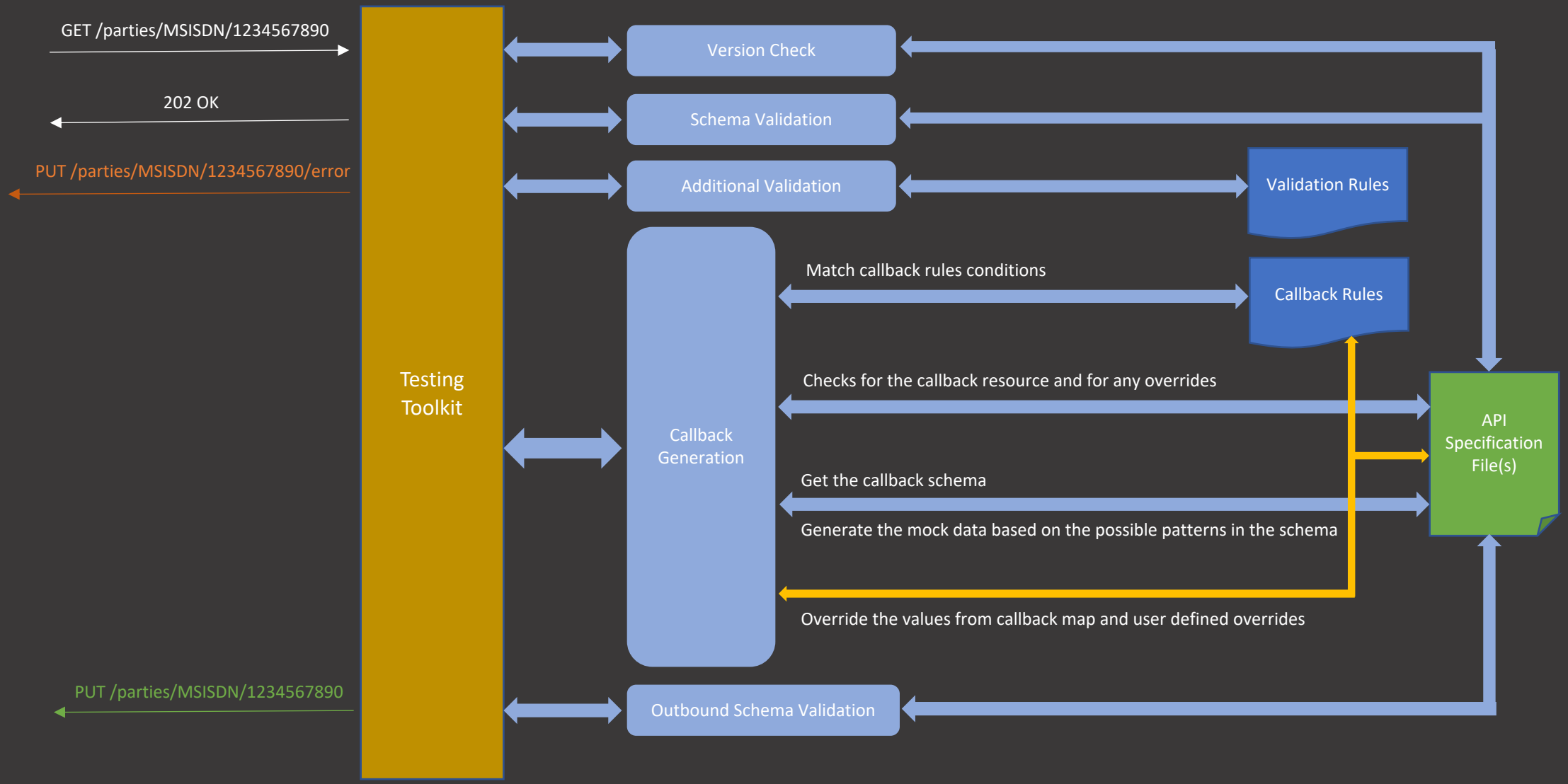
Architecture Diagram – Part 1



Architecture Diagram – Part 2



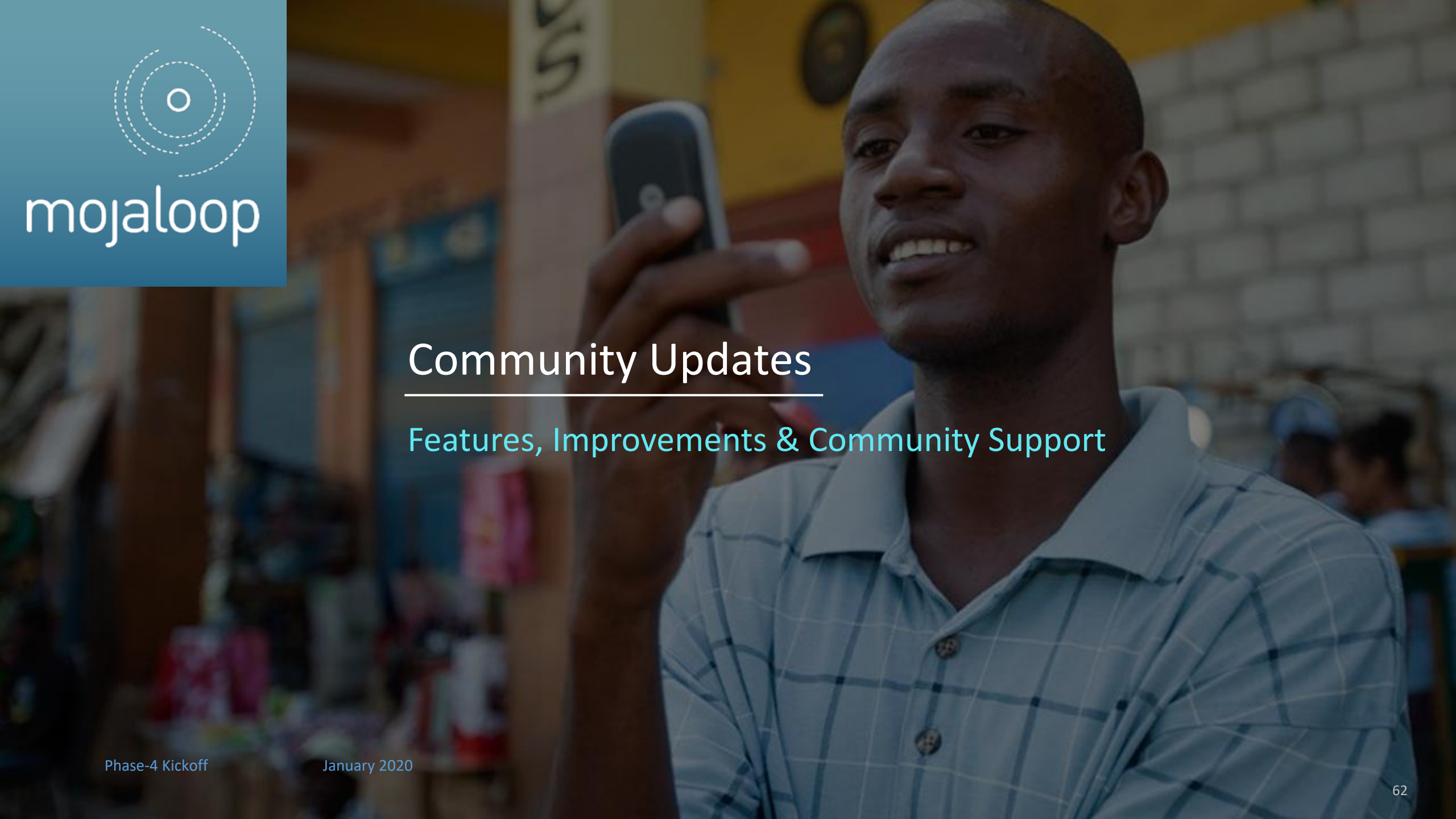
How Testing Toolkit Works



Demo - Separate

Roadmap

1. QA – Automated tests, functional tests
2. Import and Export capability of rules as files
3. JWS validation and generation and support TLS (using SDK standard components)
4. Validate association of a transfer request with an existing quote
5. Expand support for all the resources in API spec file
6. A dashboard for statistics
7. Incorporate prioritized items from feedback, community input



Community Updates

Features, Improvements & Community Support

Mojaloop OSS: Community

1. Change Control Board (CCB)
2. Design Authority (DA)
3. Weekly scrum-of-scrums
4. Slack channels #*general*, #*announcements* , #*help-mojaloop*, #*design-authority* *(Gitter)*

ML OSS Community: Design Authority

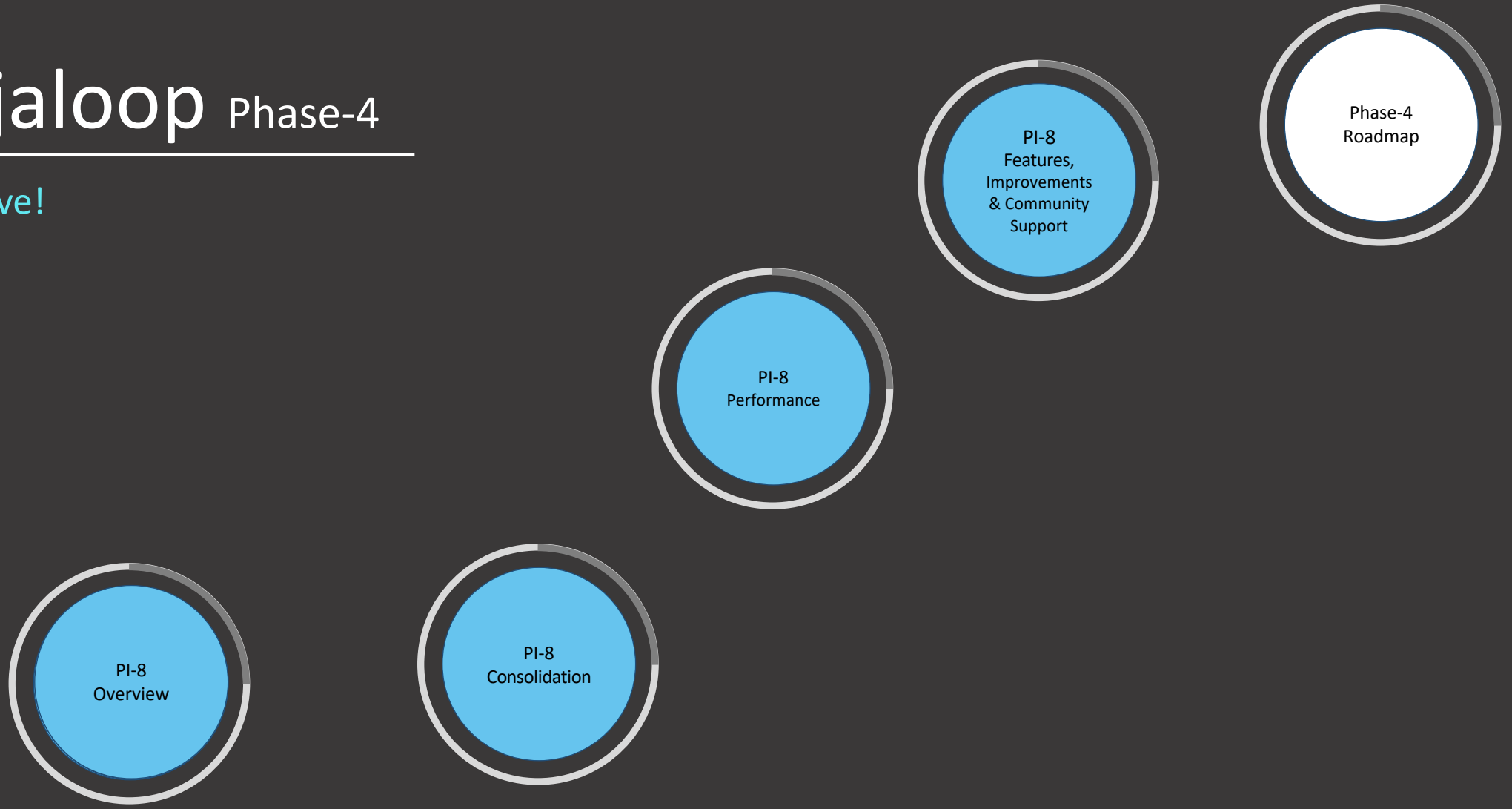
1. Goals, purpose
2. Membership
3. Frequency of meetings, boards used
4. Functioning overview
5. Example topics, discussions covered as part of the DA

ML OSS Community: Change Control Board

1. Goals, purpose
2. Current Membership
   i. BMGF (Matt Bohan, Miller Abel)
   ii. Ericsson (Henrik Karlsson)
   iii. Huawei (Chen Hill)
   iv. Mahindra Comviva (Ritvik Sinha)
   v. ModusBox [Non-voting] (Michael Richards, Sam Kummary)
   vi. Mowali (John Mark Ssebunya)
   vii. Telepin (RJ Wilson)
   viii. TIPS (Mutashobya Mushumbusi)
3. Frequency of meetings, boards used
4. Change requests, Solution proposals, Bugs
5. Versioning

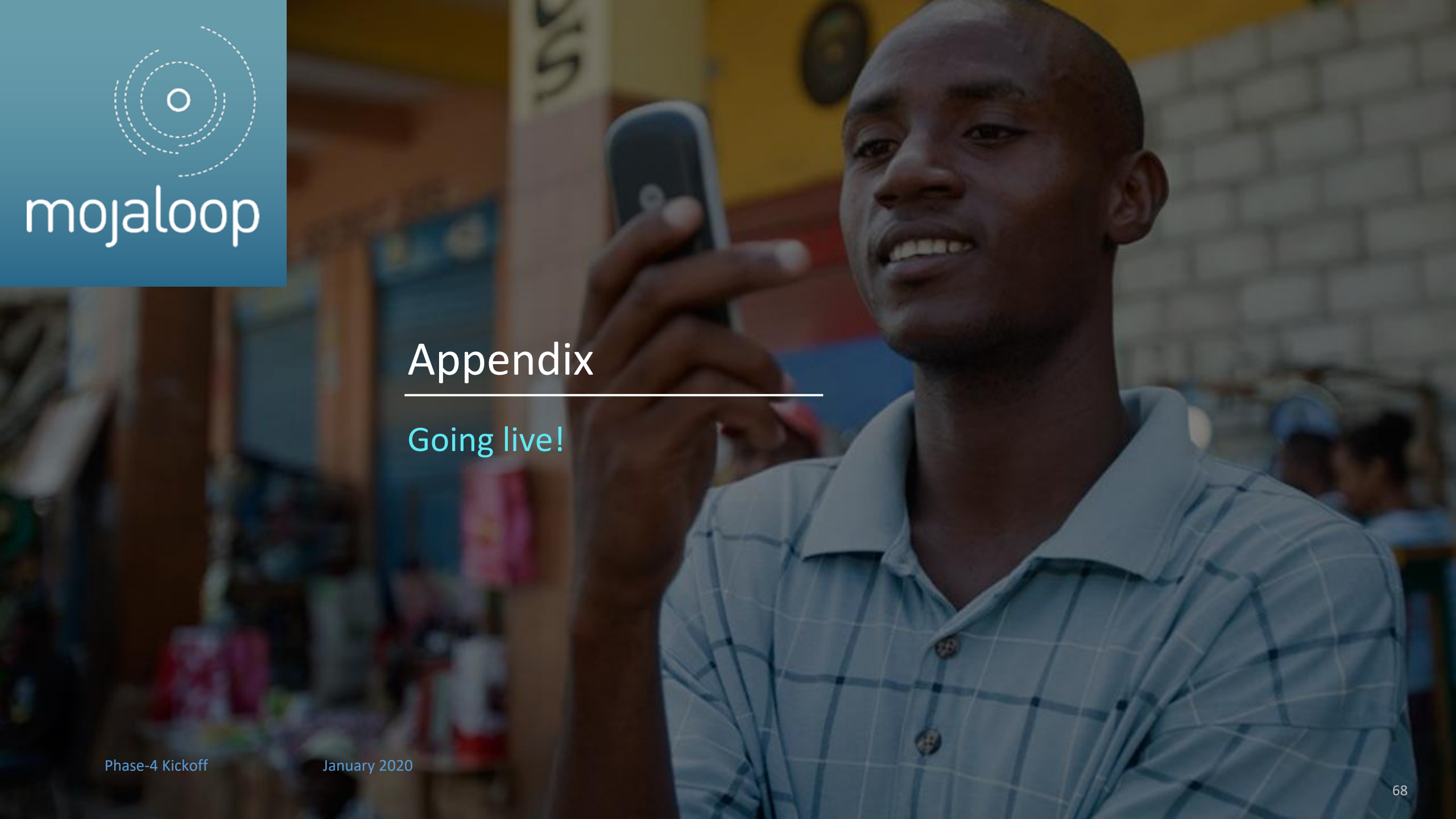
Mojaloop Phase-4

Going Live!



Phase-4: Roadmap

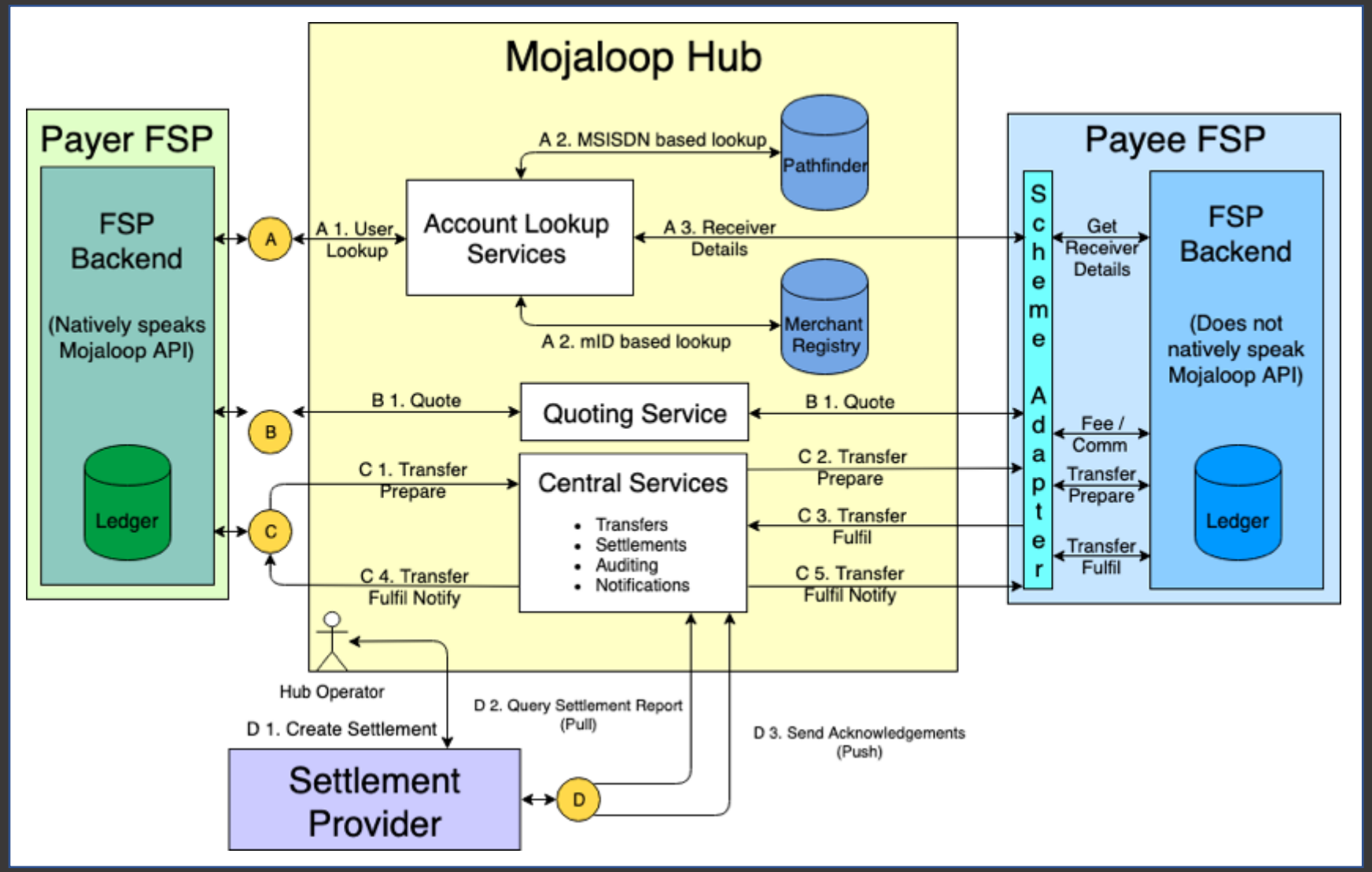
1. Settlement-v2 changes
2. Versioning standards, version maintenance
3. Cross-currency changes
4. Security
5. Performance improvements
6. Standardizing bulk transfers, QA
7. Merchant payment support
8. Standardize operations (Admin) API
9. Portal for Hub Operations and Hub TechOps
10. Secure auditing / Forensic Logging
11. PoC for a ticketing system
12. Design for reliable and tiered storage
13. Maintenance
    a. Upgrading Helm to v3
    b. Upgrading all NodeJS services to Node v12 (recent dependencies are not compatible with Node v10)



Appendix

Going live!

Mojaloop Overview



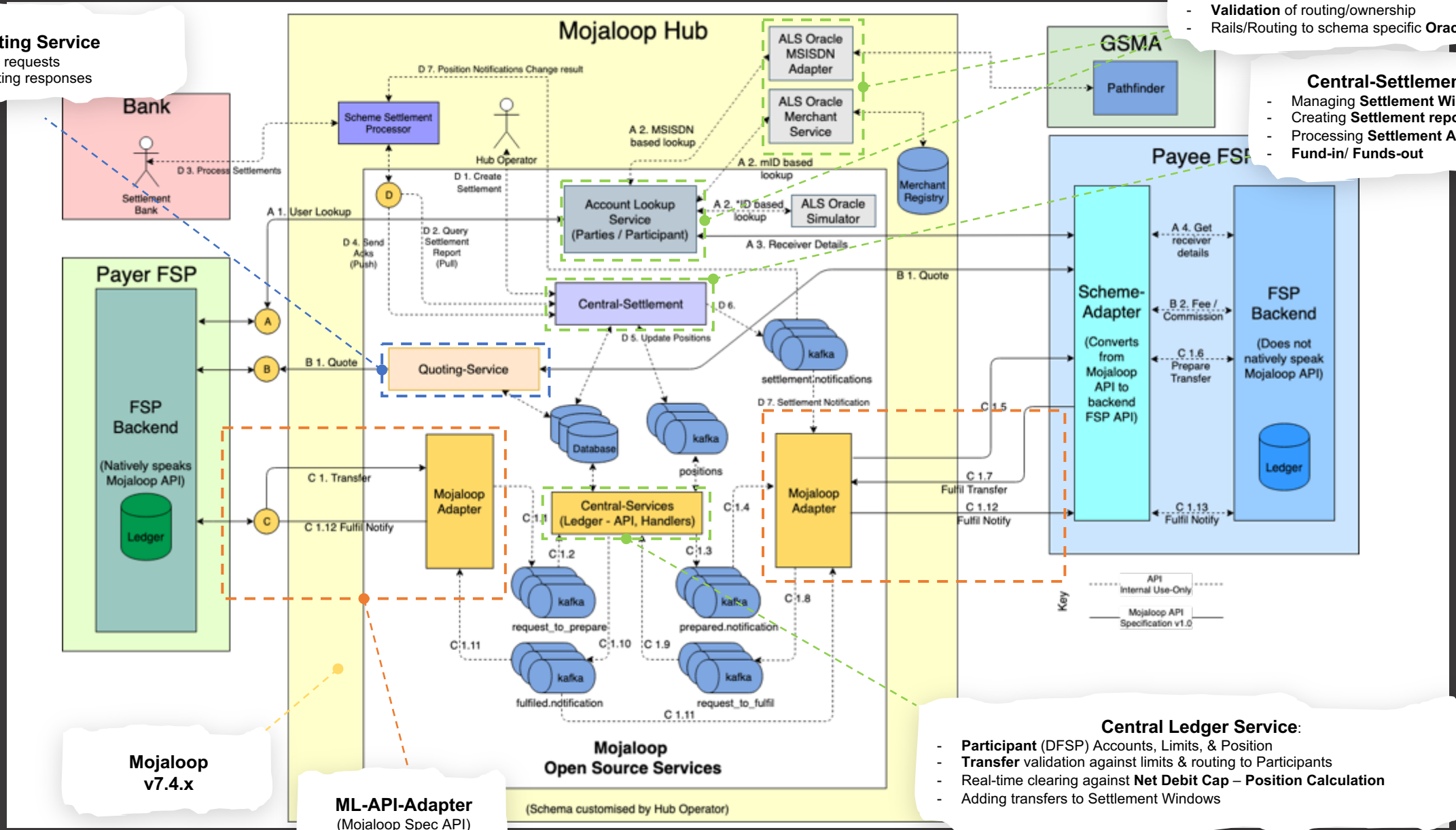
DEMO: http://mojaloop.io/docs/CentralServices/mojaloop-architecture-static-demo/index.html

High-level Architecture (PI7)

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**

Quoting Service
- Initiate Quote requests
- Resolve Quoting responses

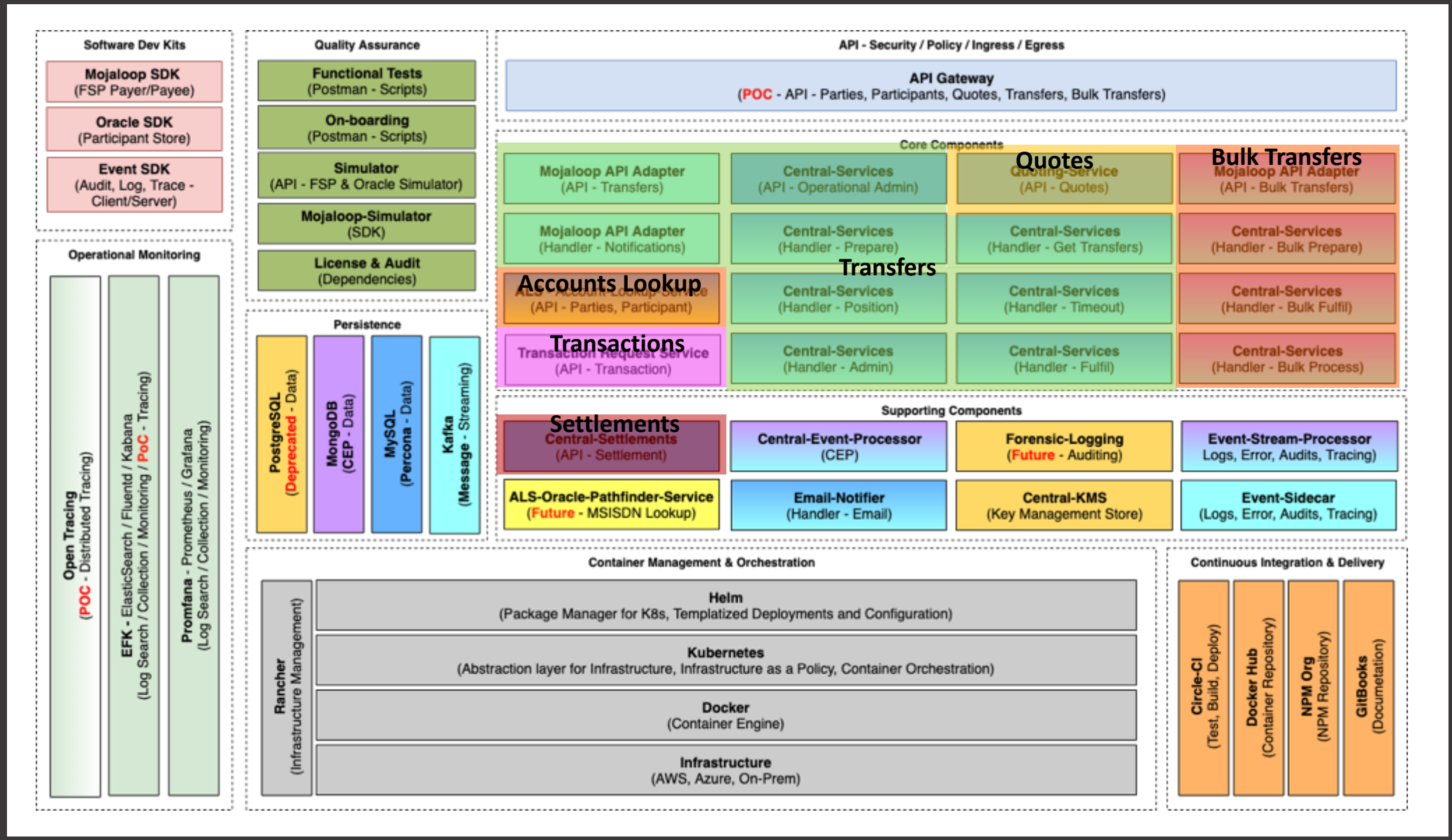


Mojaloop v7.4.x

ML-API-Adapter
(Mojaloop Spec API)

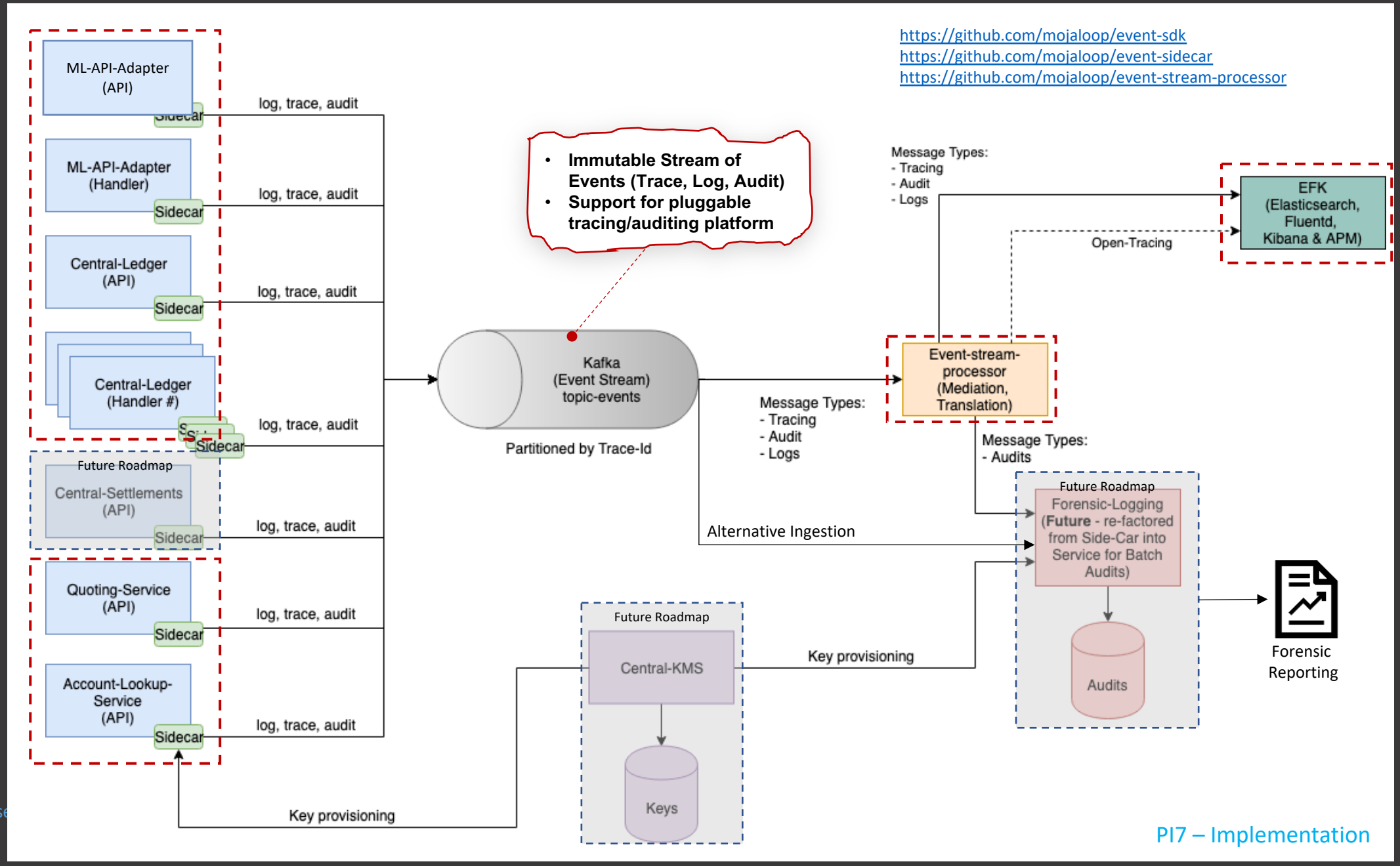
Central Ledger Service:
- **Participant** (DFSP) Accounts, Limits, & Position
- **Transfer** validation against limits & routing to Participants
- Real-time clearing against **Net Debit Cap** – **Position Calculation**
- Adding transfers to Settlement Windows

Component Overview (PI7)



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

Event Framework – Current Functional Overview



Run-time Platform – 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
- Elastic scaling (horizontally up/down)
- Limit hardware & resources via Policies

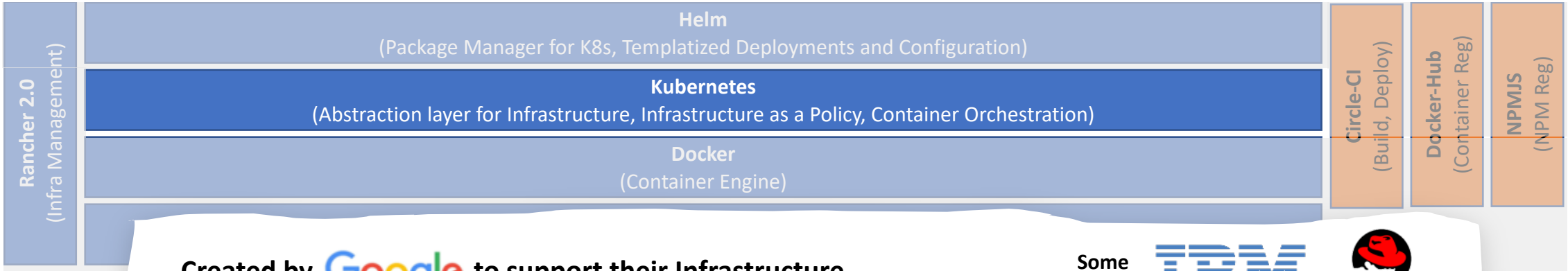
Roll out new features seamlessly
- Rolling updates

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.

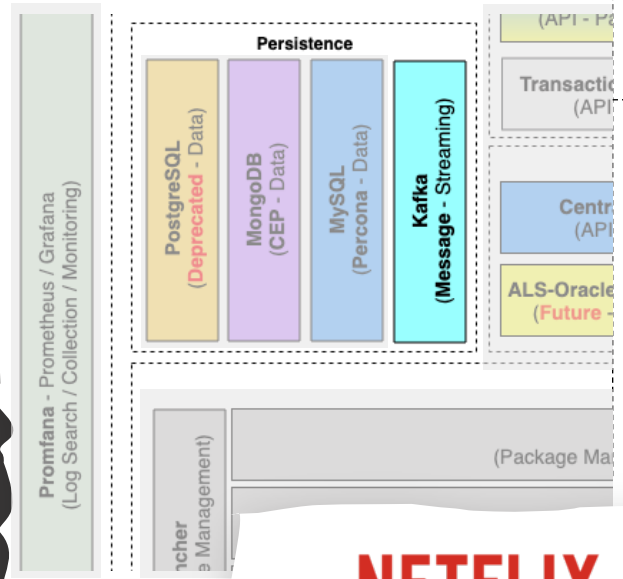
Some contributors: IBM redhat

Messaging Platform – Kafka

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

What is Kafka?

Apache Kafka is a distributed message streaming platform.

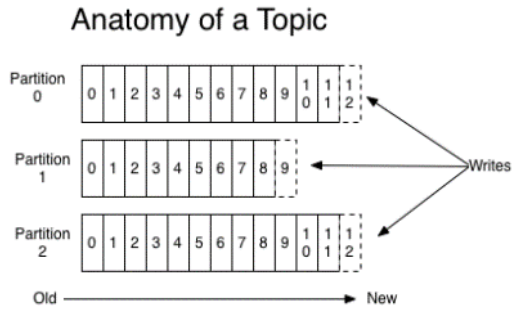


Why Kafka?

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.

Used By

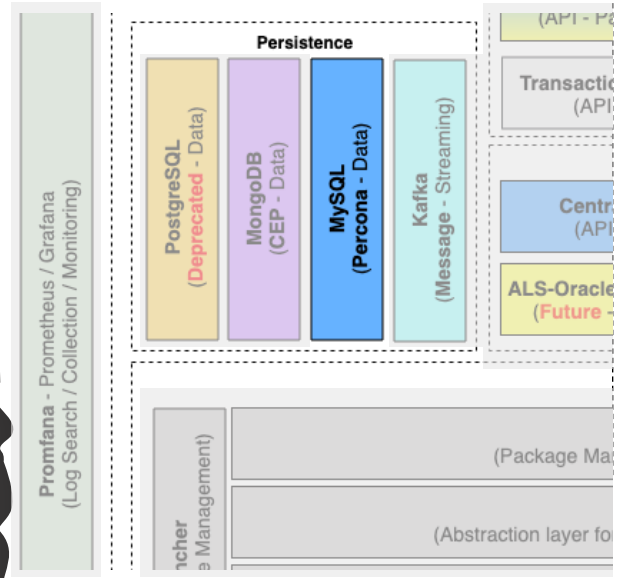
NETFLIX PayPal ORACLE IBM Rabobank

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

Deployment Architecture – Rancher Overview

Ref: http://rancher.com

What is Rancher?

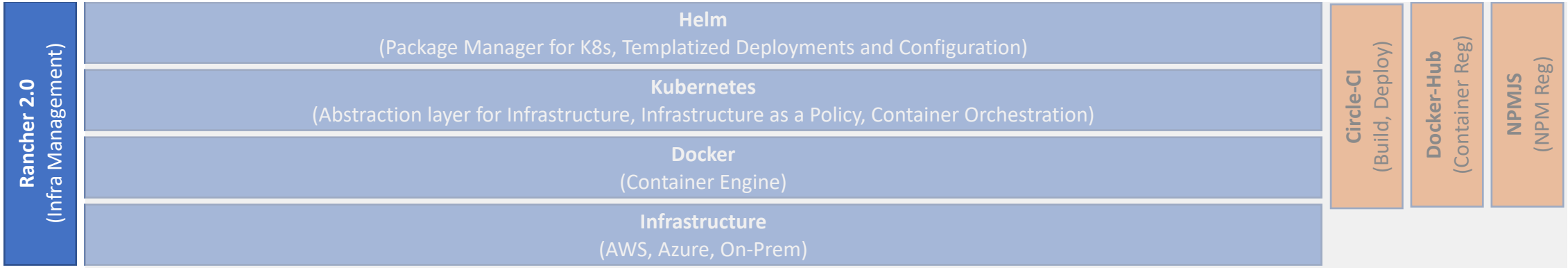
Rancher is enterprise management for Kubernetes.

Every distro. Every cluster. Every cloud.

Why Rancher?

- Kubernetes Management (v1.8 to *v1.9)
- 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

** New in Rancher v2.x*



Deployment Architecture – Helm

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.

Microsoft Google codefresh bitnami ticketmaster codecentric

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.

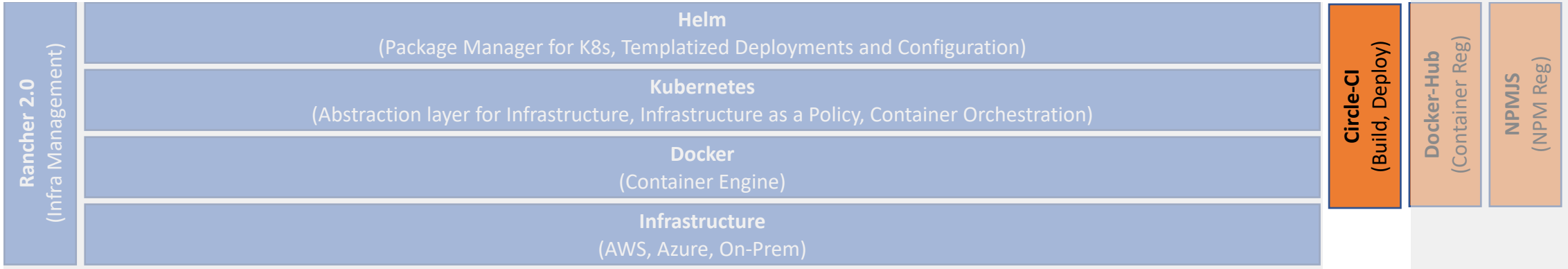
Why CircleCI?

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)*

Documentation – GitBooks Overview

GitBook

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

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?

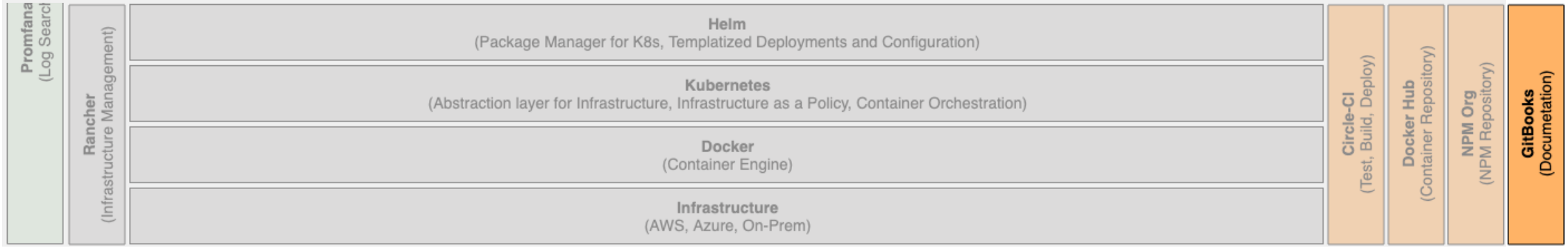
Markdown
Lightweight markup language with plain text formatting syntax supporting standard HTML, and CSS.

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)

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

Search
Find what you are looking for.



79

Deployment Architecture – CI/CD Pipeline

circleci

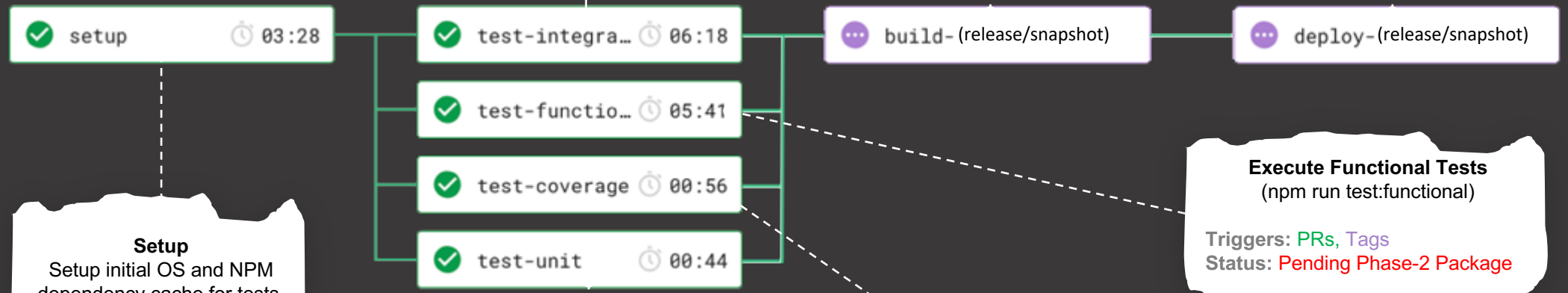
Triggers:

[●] Pull-Requests (inc. from external Forks)

[●] Publishing tagged Releases

[●] Publishing tagged Snapshots

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



Execute Integration Tests
(npm run test:integration)

Triggers: PRs, Tags
Status: Working

Build & Publish Artifact
Docker or NPM build and publish to repository

Triggers: Tags
Status: Pending Phase-2 Package

Deploy Artifact
Execute either helm install or upgrade (if existing deployment) with latest Artifacts

Triggers: Tags
Status: Pending Phase-2 Package

Setup
Setup initial OS and NPM dependency cache for tests, builds, and deployments.

Triggers: PRs, Tags
Status: Working

Unit Tests
(npm run test:unit)

Triggers: PRs, Tags
Status: Working

Execute Functional Tests
(npm run test:functional)

Triggers: PRs, Tags
Status: Pending Phase-2 Package

Coverage Check
Executes Code Coverage checks. The results will be published to SonarQube if the branch is "Master".
(npm run test:coverage)

Triggers: PRs, Tags
Status: Working

Event Monitoring Framework – SDK Example

REFERENCE

New Span:
- Generate **Trace Context** (**traceId** & **spanId**)

Finish Span:
- Close **Span**
- Record **Trace**

```
// Creates a new parent span for given service
// this sets new traceId and new spanId.
let parentSpan = Tracer.createSpan( service: 'parent service')
```

```
// Finish the span. This also sends the trace context to the tracing platform. All further operations are forbidden after the span is finished.
await parentSpan.finish(event)
```

Recorded events
- **Logs** (info, warn, debug, verbose, perf, error)
- **Audit**

```
// Logs message with logging level info from the parent span
await parentSpan.info(event)
await parentSpan.warning( message: 'event')
await parentSpan.error('event')
await parentSpan.debug('message')
await parentSpan.verbose( message: 'message')
await parentSpan.performance( message: 'message')
await parentSpan.audit( message: 'message')
// Logs message with logging level debug from the parent span
await parentSpan.debug('this is debug log')
```

Add Tags
- Add arbitrary **tags** for **metadata** as part of **Trace Context**

```
// Set tags to the span
IIChildSpan.setTags({ one: 'two' })
```

Child Span
- **Create Child Span** from a parent span
- Generate **spanId**, and set **parentId**

```
// Creates child span from the parent span with new service name.
// The traceId remains the same. The spanId is new and the parentSpanId is the spanId of the parent.
let IIChildSpan = parentSpan.getChild( service: 'child II service')
```

Trace Context
- **Inject Trace Context** into a Message
- **Extract Trace Context** from a Message
- **Create Child Span** from **Trace Context**

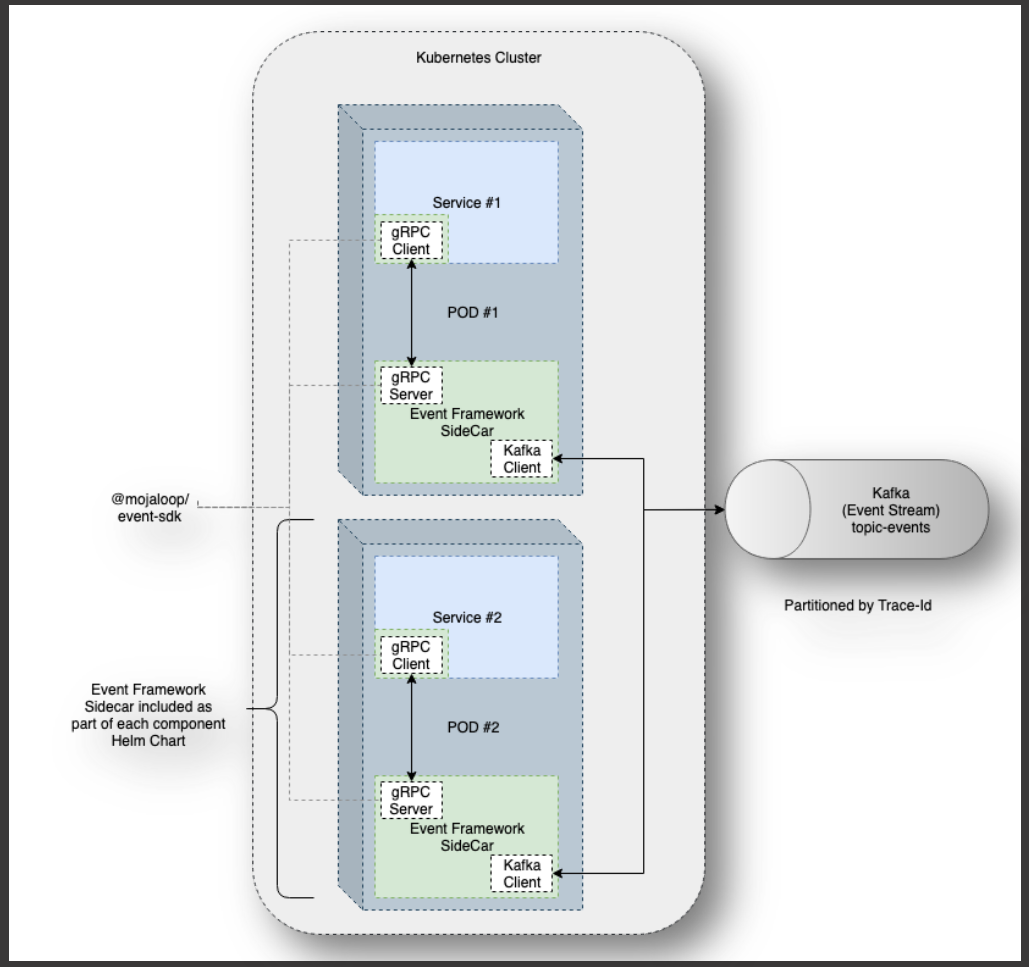
```
// Injects trace context to a message carrier. When the trace is carried across few services, the trace context can be injects in the carrier that transports the data.
let messageWithContext = await IIChildSpan.injectContextToMessage(event)
// await sleep(2000)

// Extracts trace context from message carrier. When the message is received from different service, the trace context is extracted by that method.
let contextFromMessage = Tracer.extractContextFromMessage(messageWithContext)

// Creates child span from extracted trace context.
// let IIIChild = Tracer.createChildSpanFromContext('child III service', contextFromMessage, { defaultRecorder: new DefaultLoggerRecorder() })
let IIIChild = Tracer.createChildSpanFromContext( service: 'child III service', contextFromMessage)
```

Janua

Event Framework – Deployment Arch & Roadmap



Deployment Architecture

Future Roadmap

1. Design considerations:
   a. Audit requirements / functionality
   b. Crypto signatures for Audit logs (single & batch)
   c. Opentracing / Zipkin dashboards if EFK is not adequate
   d. Sidecar inter-lock
2. Implementation:
   a. Audits sig & processing
   b. Official releases
   c. Mojaloop Helm Chart integration

GitHub PoC Repositories*

1. https://github.com/mojaloop/event-sdk
2. https://github.com/mojaloop/event-sidecar
3. https://github.com/mojaloop/event-stream-processor
4. https://github.com/mojaloop/apm-agent-nodejs
5. https://github.com/mojaloop/apm-agent-nodejs-opentracing
6. https://github.com/mojaloop/opentracing-javascript

** Note:*

1. *Event PoC code currently in feature branches*
2. *Snapshot releases currently available*