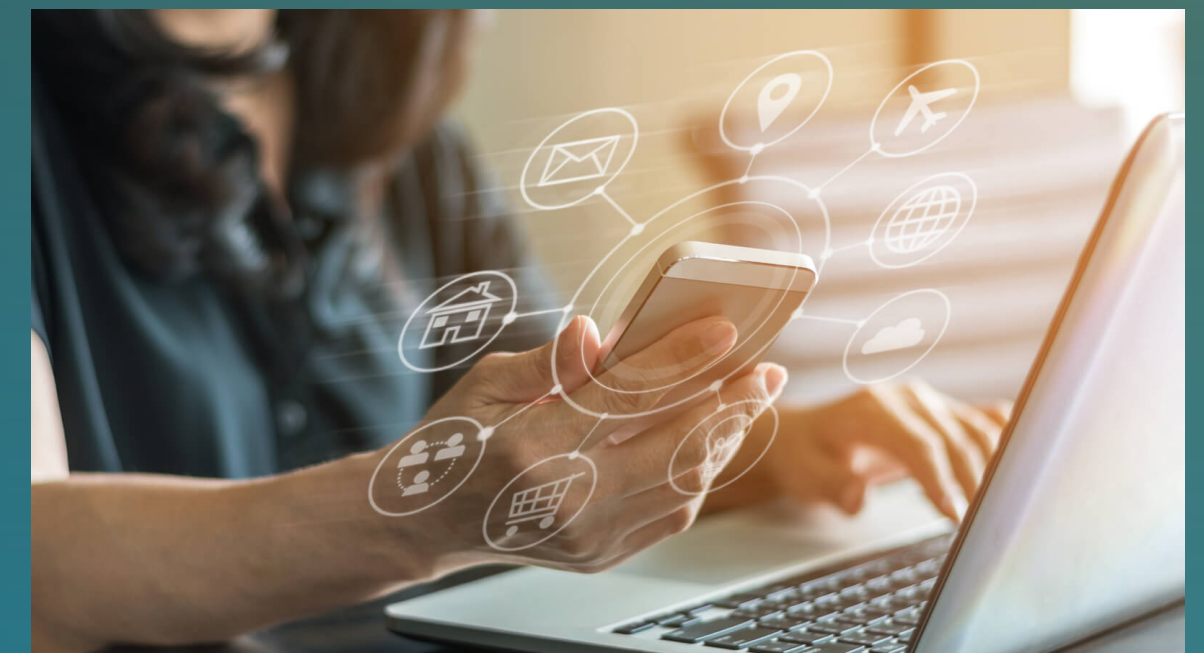


Transactions in your micro-services architecture



Dawn Parzych (CatchPoint)

What ?

- Transaction and Micro-Services
- MicroProfile LRA (Long Running Actions)
- Demo

Rudy De Busscher

- Payara
 - Service team
- Involved in
 - Committer of MicroProfile
 - Committer in Eclipse EE4J groups
 - Java EE Security API Expert group member



@rdebusscher



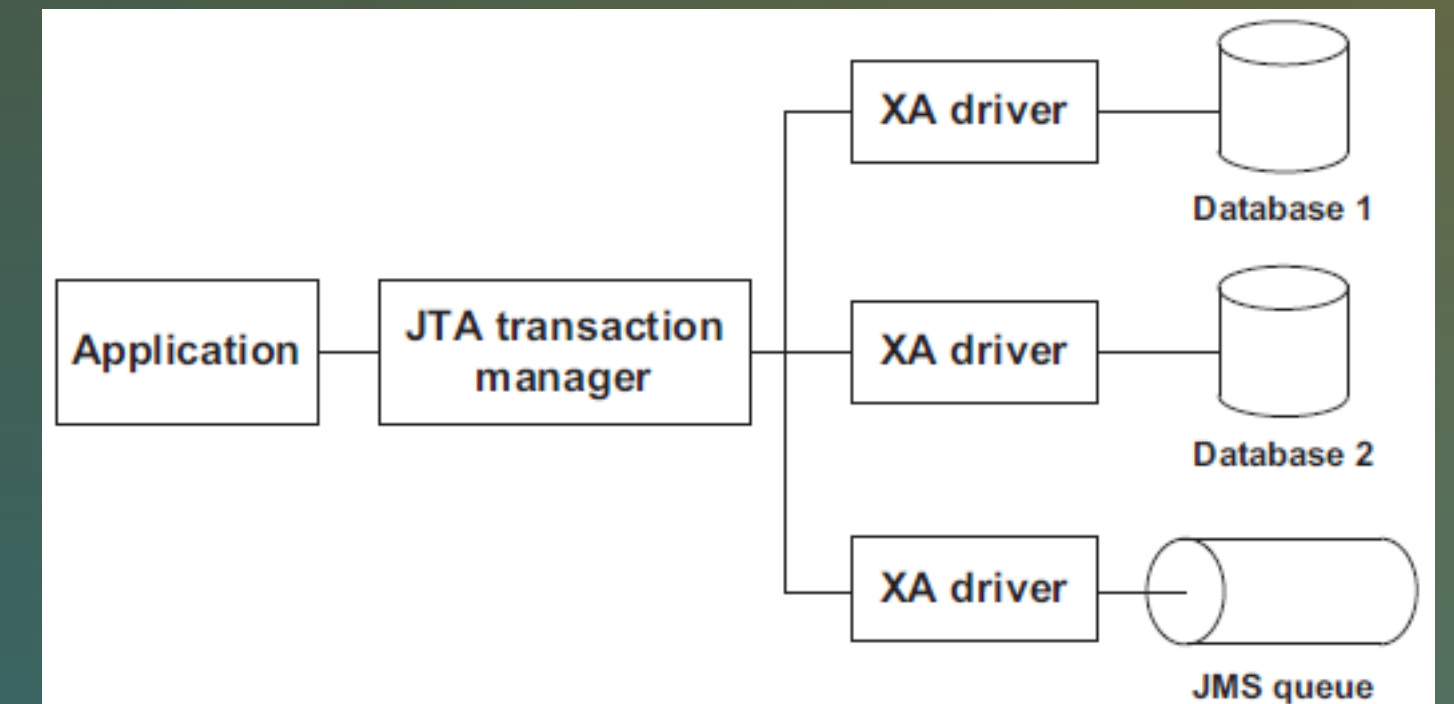
<https://blog.payara.fish/>
<https://www.atbash.be>

Transaction

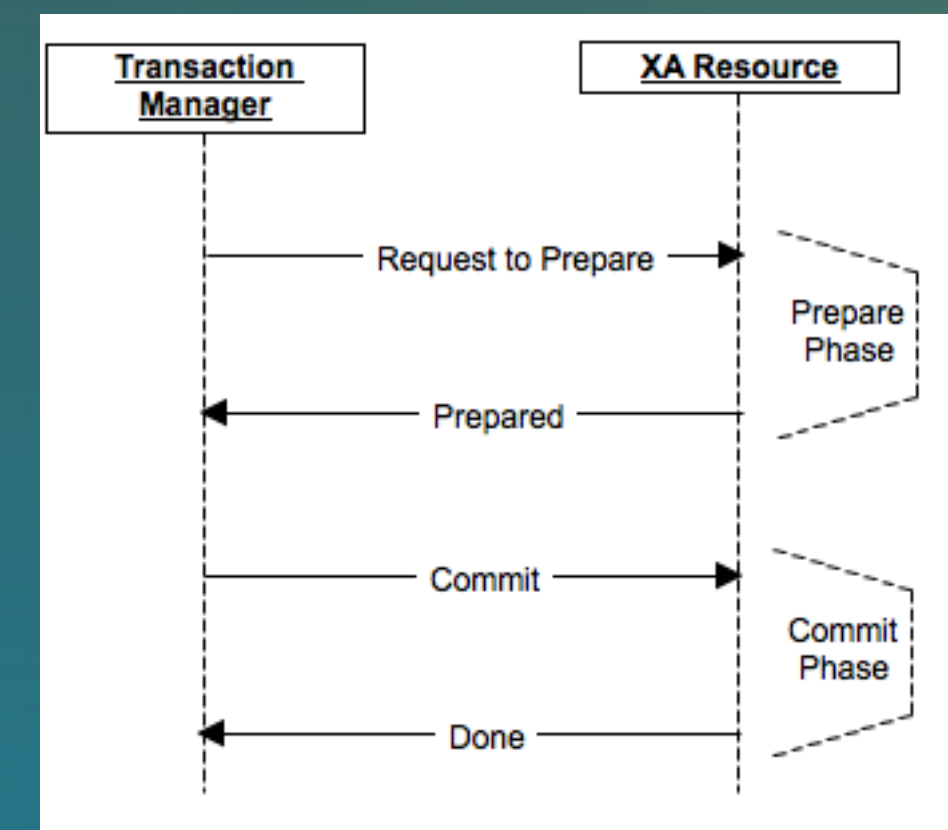
- Typical concept from the 'Monolith'
- Start - End
- Typical short - to avoid locks
- Multiple datasources supported with XA transactions

Transaction

- Distributed - XA transactions
- 2 Phase Commit
- Not scalable
 - Locks



fizalihan - Transactions



DZone - XA transactions

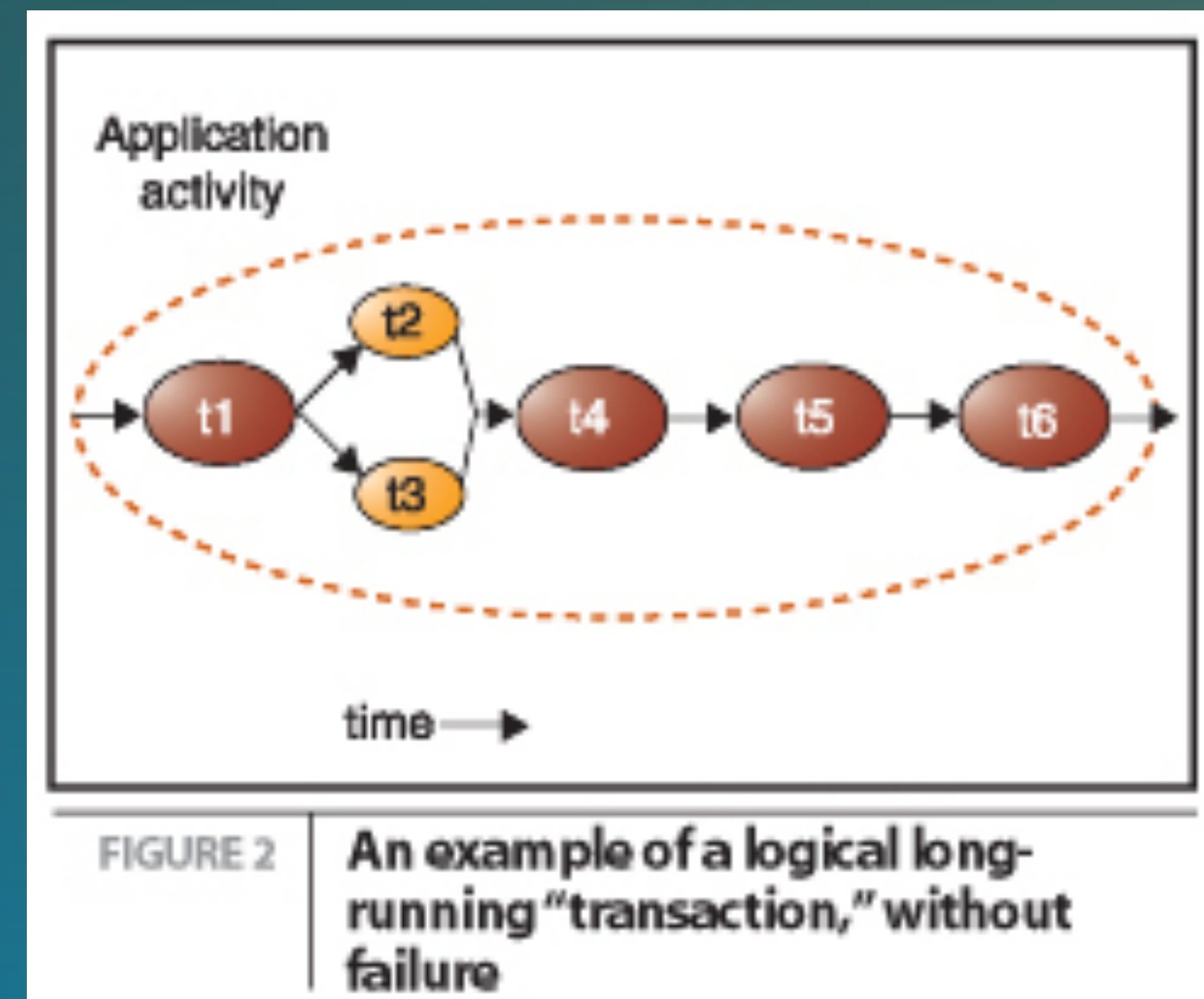
To be ACID or Not

- Atomicity: All or Nothing
- Consistency: All values are aligned
- Isolation: Different transactions can't see 'temporal' values
- Durable: Reliably Stored

- CAP Theorem
 - Consistency - Availability - Partitioning

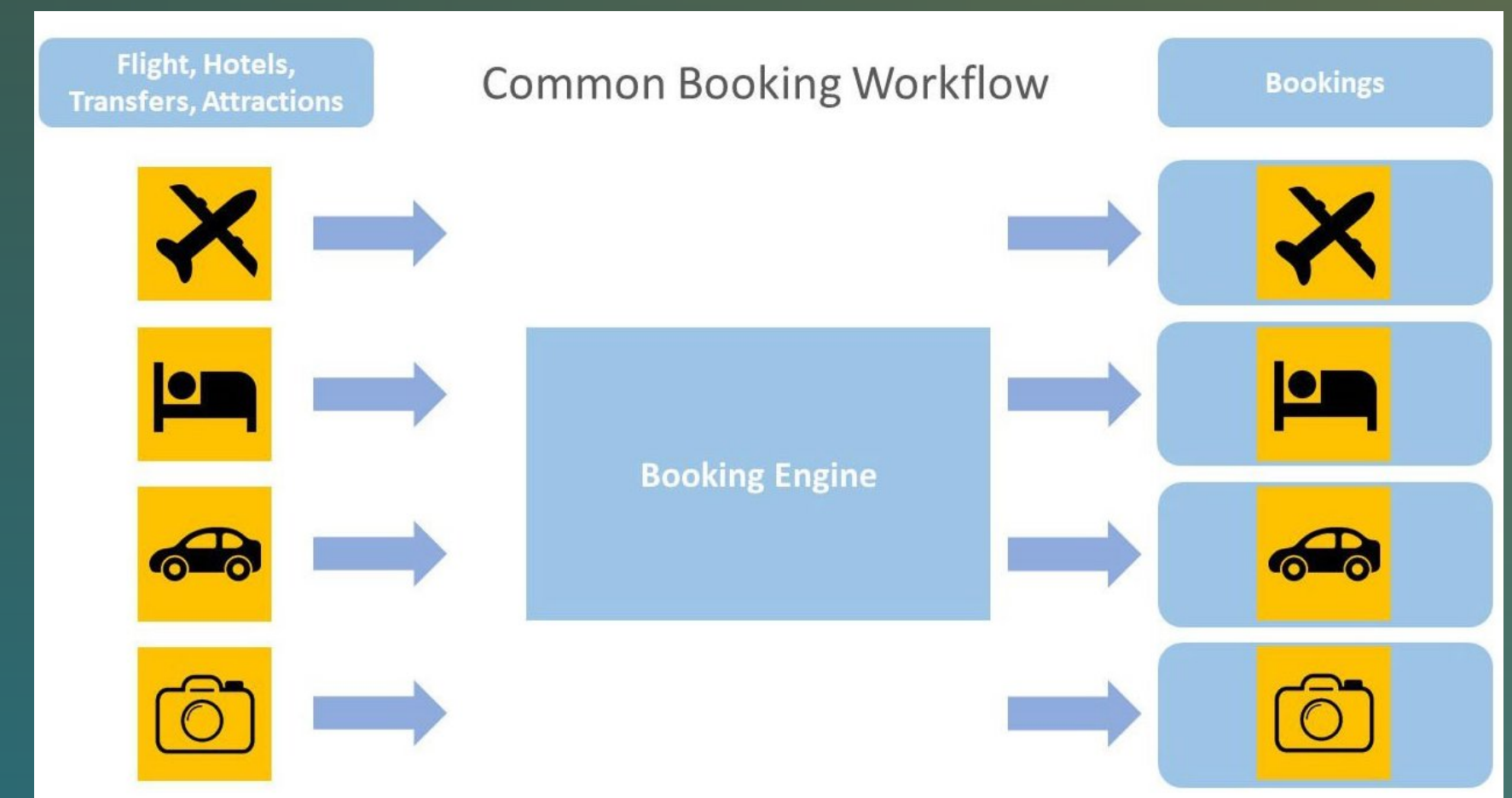
Long Running Transactions

- Multiple Datasource
- Several steps over long period
- Single Unit of Work?



Example

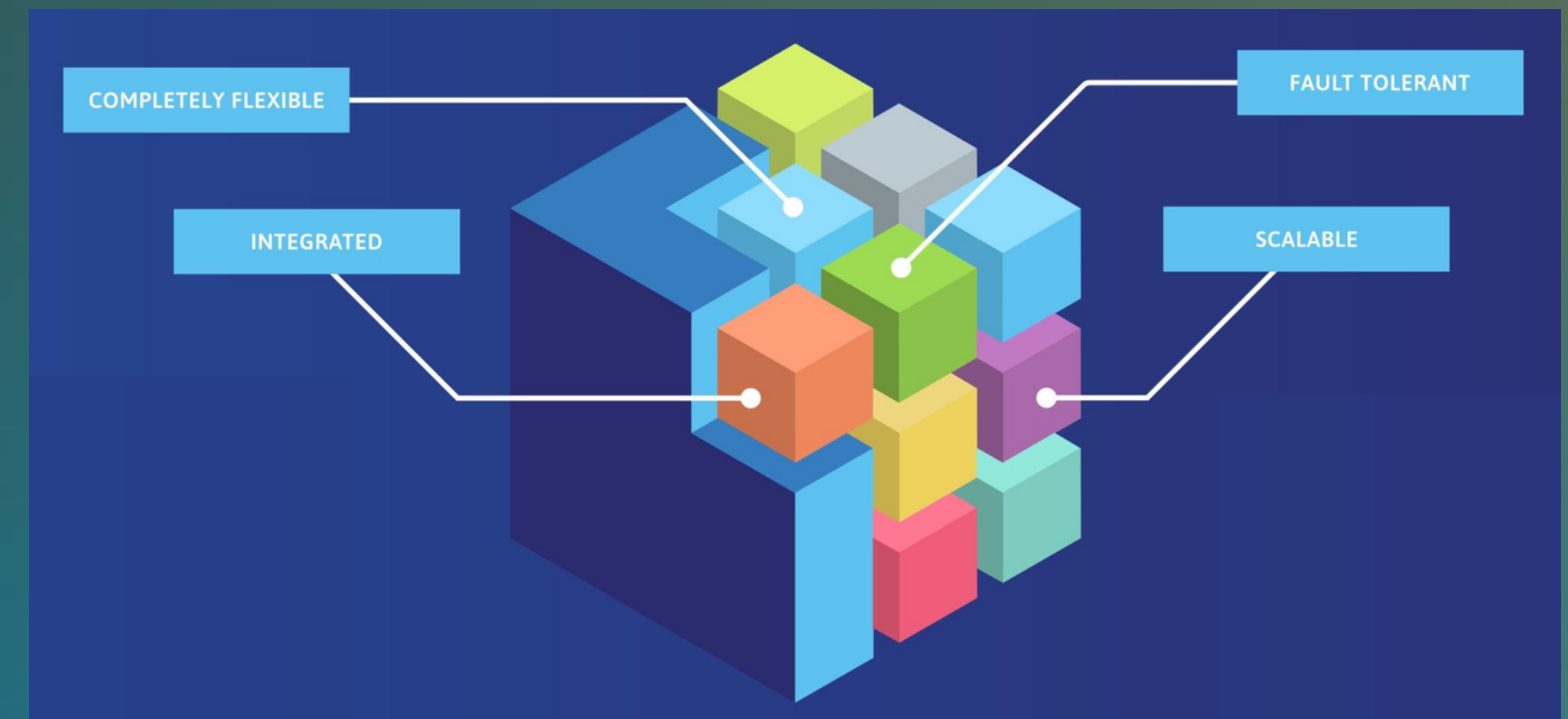
- Book Flight
- Book Hotel
- Payment



AltexSoft

MicroService world

- No **A**tomicity
- Eventual **C**onsistency
- No **I**solation
- **D**urable

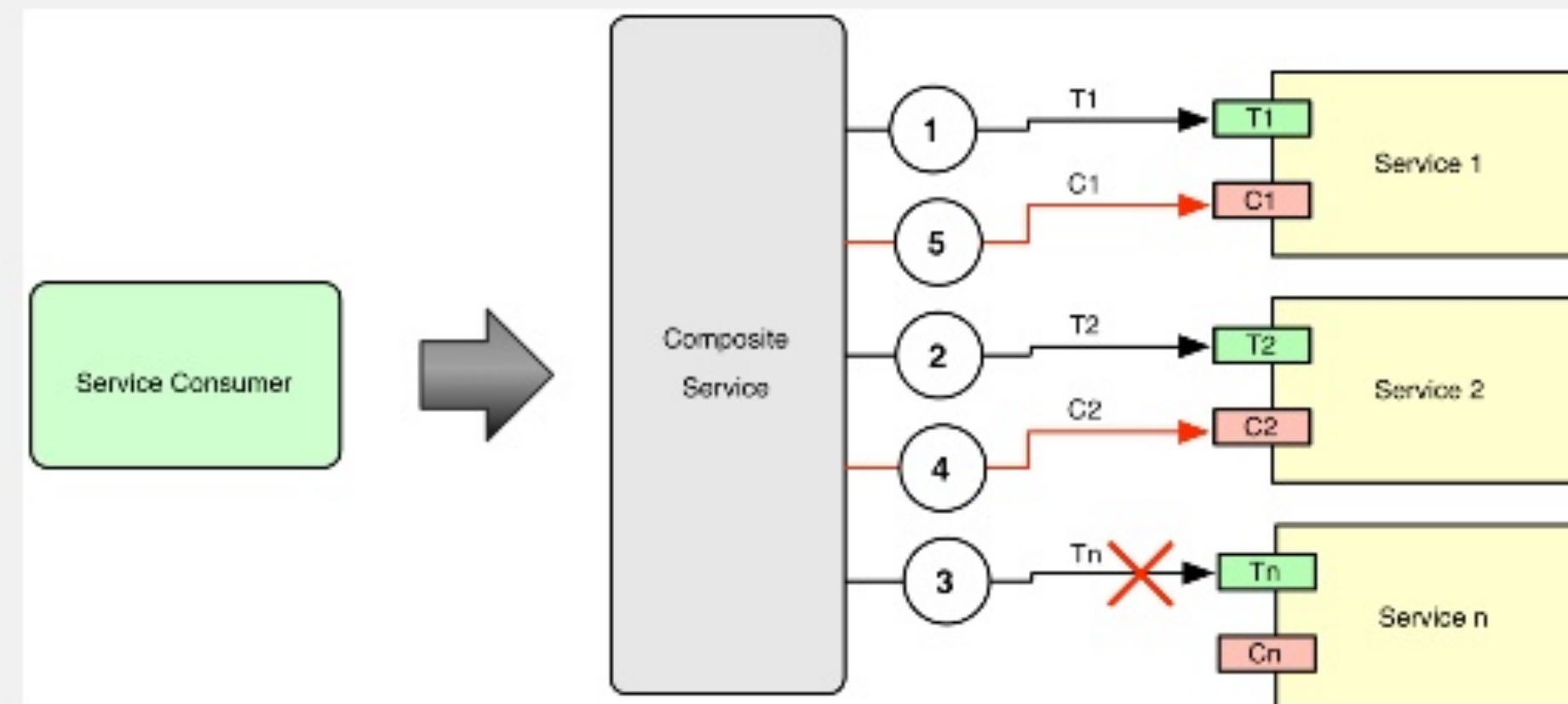


KumuluzEE

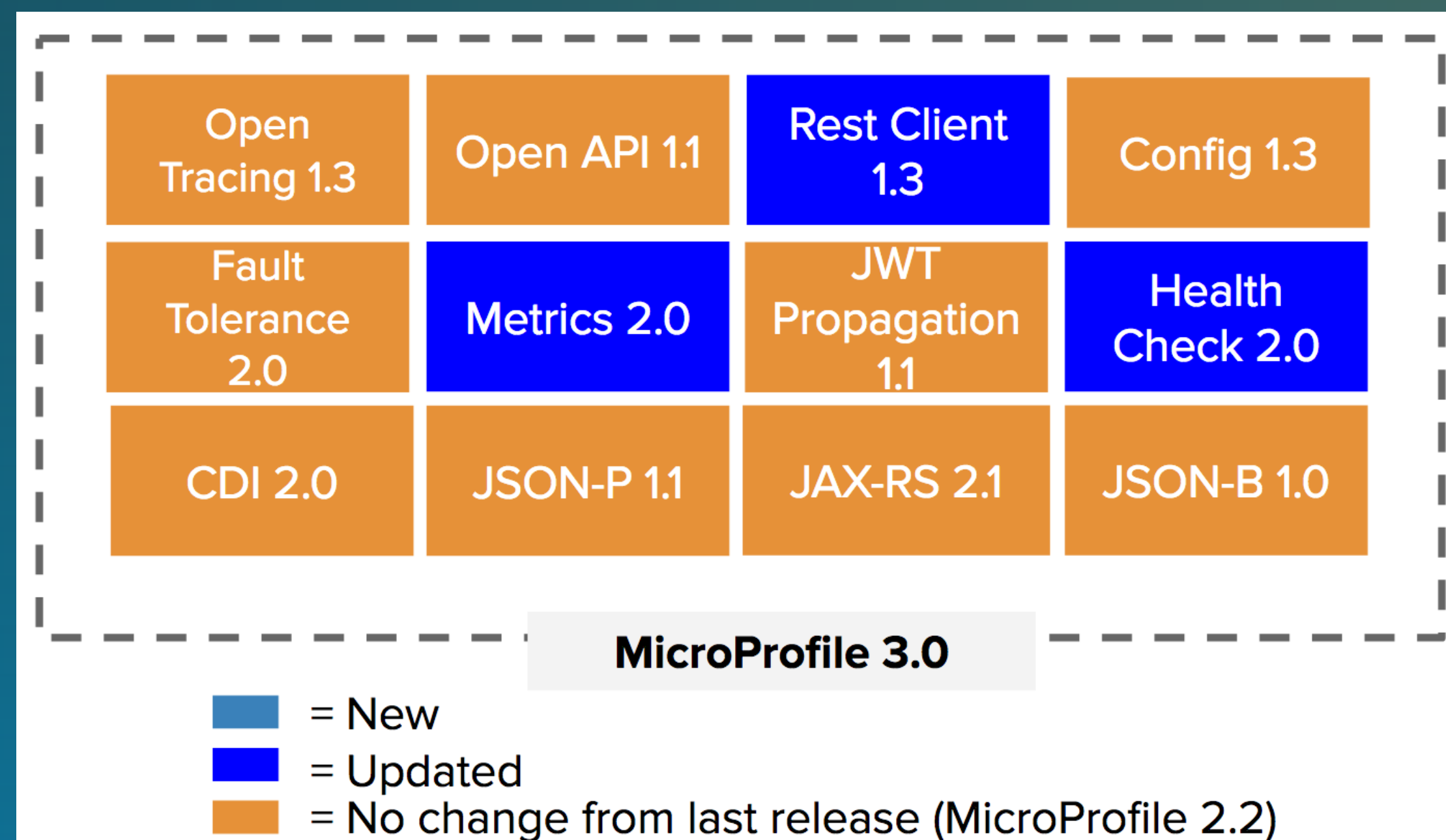
Saga Pattern

Saga Pattern

Ensures that each step of the business process has a compensating action to undo the work completed in the case of partial failures.



- Optimizing Enterprise Java for a Micro-services Architecture
- Based on some Java EE (Jakarta EE) specs



MP Long Running Action

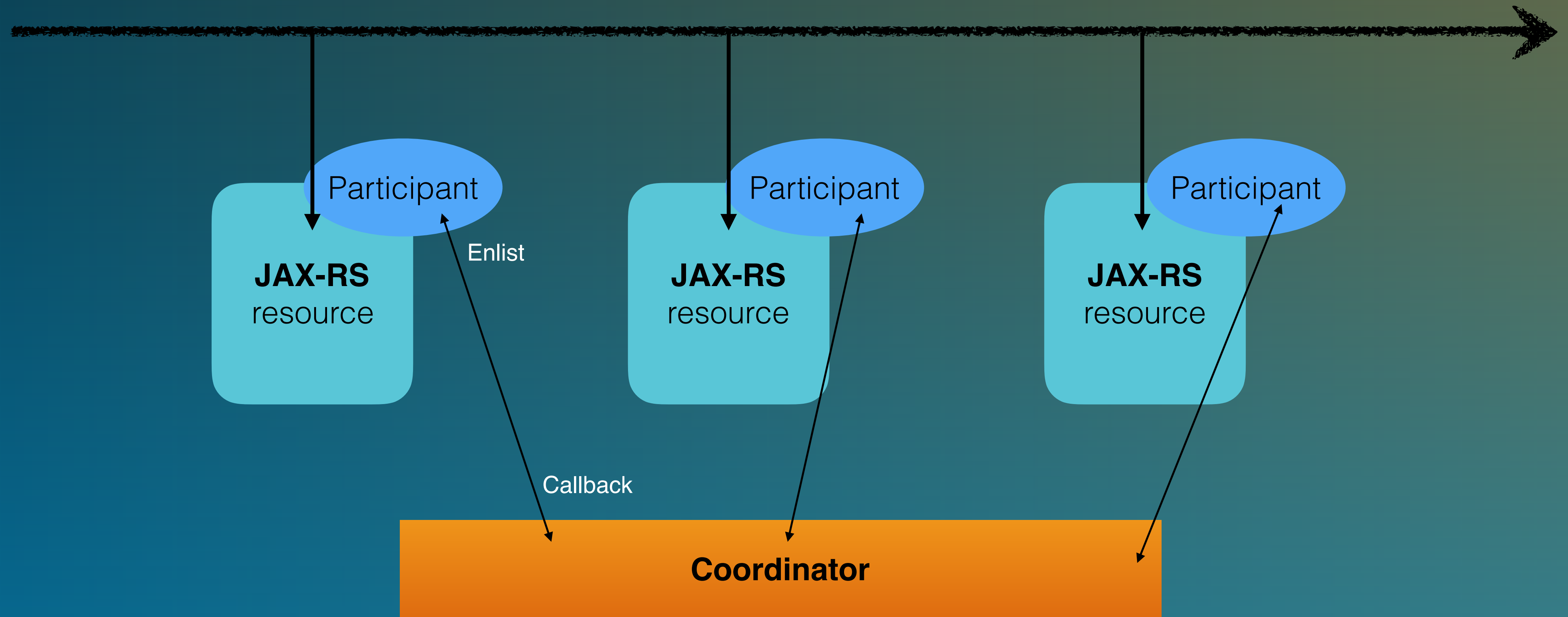


- First release candidate available
- API more or less stable

MicroProfile LRA

- Long Running Actions
- Features
 - Loose coupling
 - Guaranteed a globally consistent outcome
 - Compensatable actions (SAGAs)

LRA Components





Demo

Some Key code concepts

- `@LRA`
 - LRA Level / defines transaction
- `@Compensate`
 - Participant level: Not present, not a participant.
- `@Complete`
 - Participant level
- LRA ID (= URI)
 - Uniquely defines the Long Running Action (transaction)
 - Propagated through Header

LRA Types

- *Required for each JAX-RS resource*

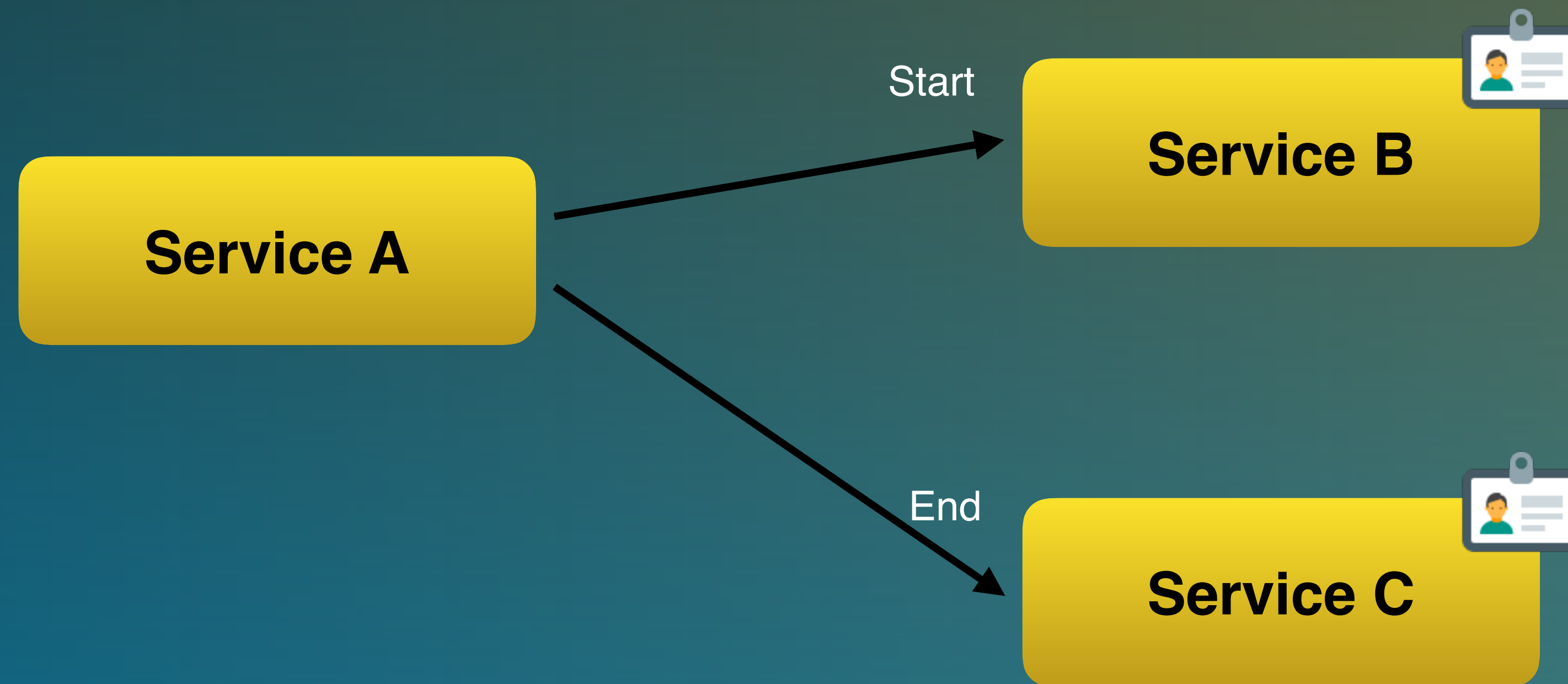
Participant



	Active LRA	No LRA
Required	Use Active	Start New
Requires New	Start New	Start New
Mandatory	Use Active	Error
Supports	Propagate	-
Never	Error	-
Not Supported	Not propagated	-



Distributed



Returning Participant Status

- Immediate
 - Return type *ParticipantStatus*, *void*
- Asynchronous return
 - Return *CompletableFuture<ParticipantStatus>*
- Idempotent *@Complete/@Compensate*
 - Return type *ParticipantStatus*
 - *ParticipantStatus.Completing* / *ParticipantStatus.Compensating*
- Through *@Status* method
 - First call to *@Complete/@Compensate*
 - Following calls to *@Status*

TimeOut

- By Default, LRA runs forever
- But also by Default, immediately closed
- Closed explicitly (end=true)
- Define a timeout -> cancelled / all participants compensated.

Remove Participant

- @Leave
- Remove Participant from LRA
- Developer responsible for cleanup
 - No @Compensate/@Complete/@Status called

Feedback final status

- `@AfterLRA`
 - What was the final outcome
- For participants
- For 'parties'
 - Can be used for any `@LRA`, not `@Compensate` annotated class.

Store LRA Info



- LRA only performs orchestration
- Participant needs to keep track of LRA
- Store LRA Id as part of your business data
- Extension
 - LRAData
 - ParticipantData

Takeaways

- Classic approach for Transactions and Long Running Action not applicable for micro-services
- MicroProfile LRA uses Compensatable actions
- Loosely coupled on top of JAX-RS resources
- Specification : In progress

Code

- Project <https://github.com/eclipse/microprofile-ira>
- Demo code
 - <https://github.com/rdebusscher/mp-ira-demo>

Thank You

Not using the Payara Platform yet? Download the open source software: Payara Server or Payara Micro

<https://payara.fish/downloads>

Need support for the Payara Platform?

<https://payara.fish/support>

