Improving Service Availabiliy with Rule-Based Adaptation

SSP 2018, Hildesheim

Marc Adolf Reiner Jung & Lars Blümke

8th November 2018

iObserve

Kieker

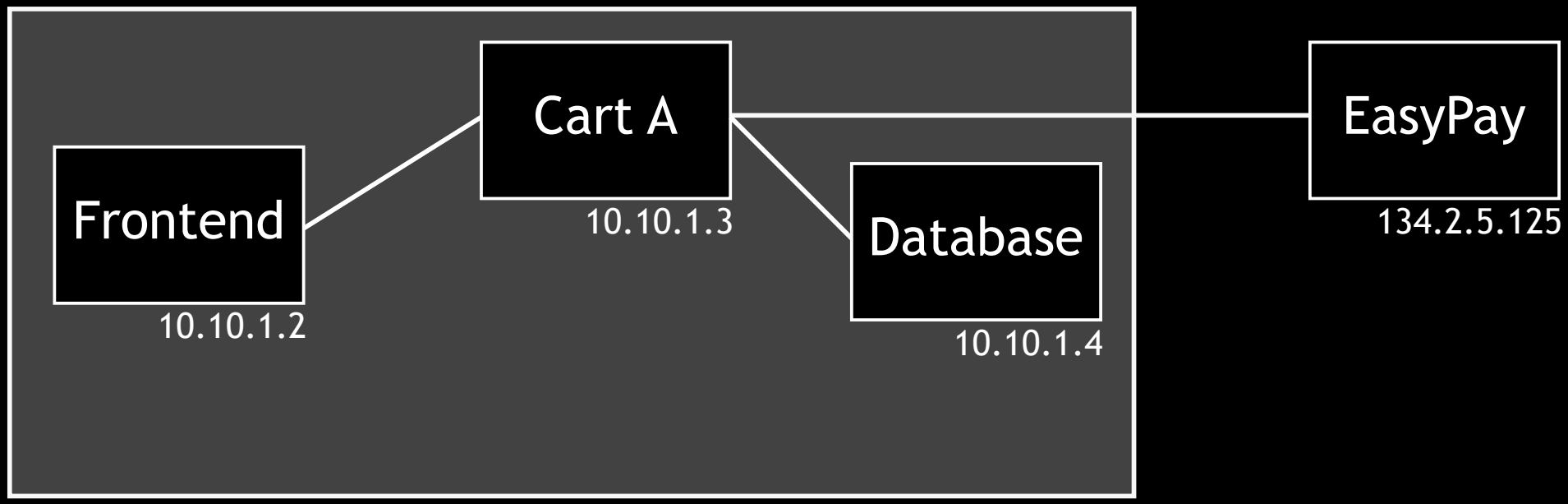
Introduction

Cloud application rely on adaptation

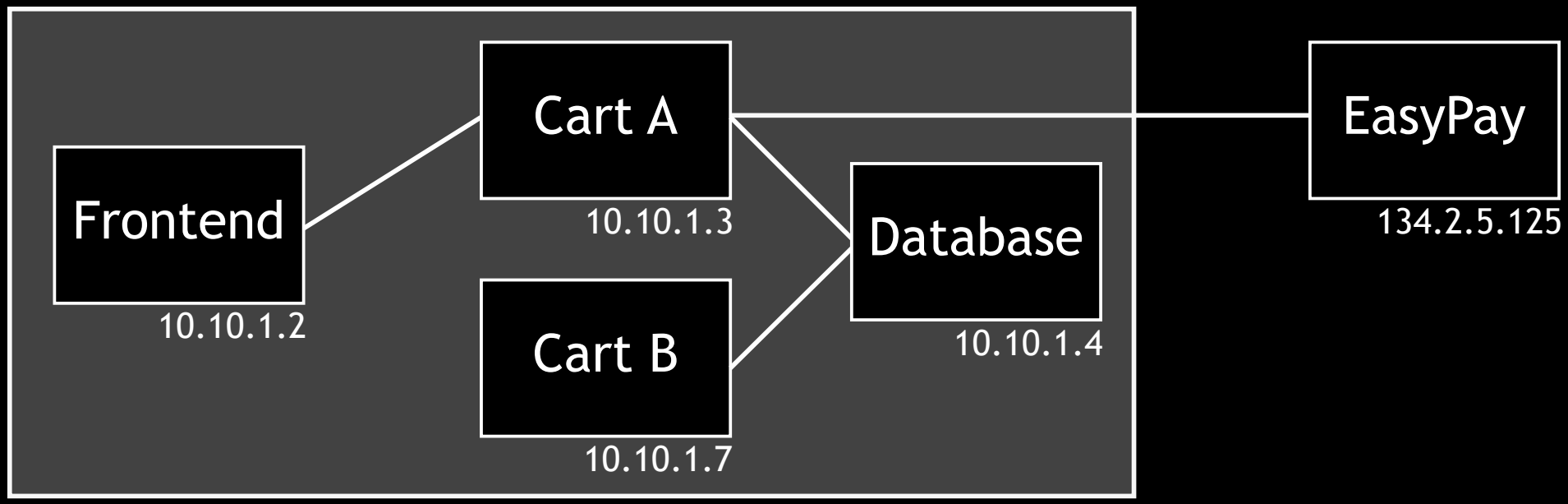
- Up- and Down-Scaling
  - (De-)Replication
  - Migration
- Reconfiguration

$\Rightarrow$ Potential information loss during scaling and reconfiguration

Example Scenarios

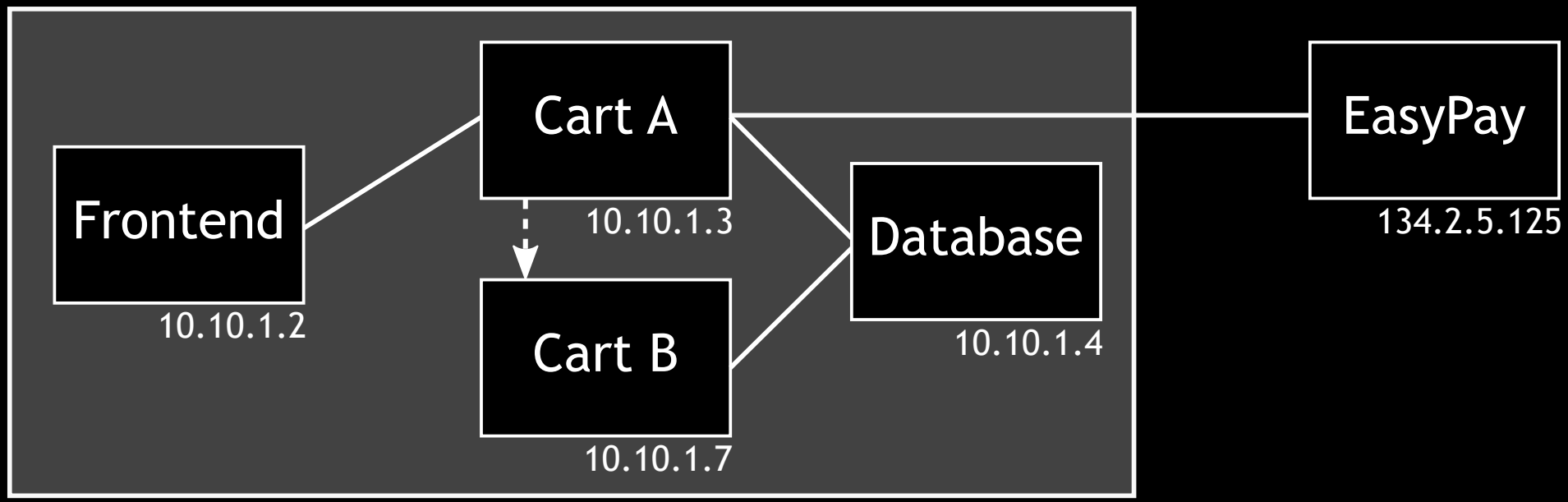


Example Scenarios

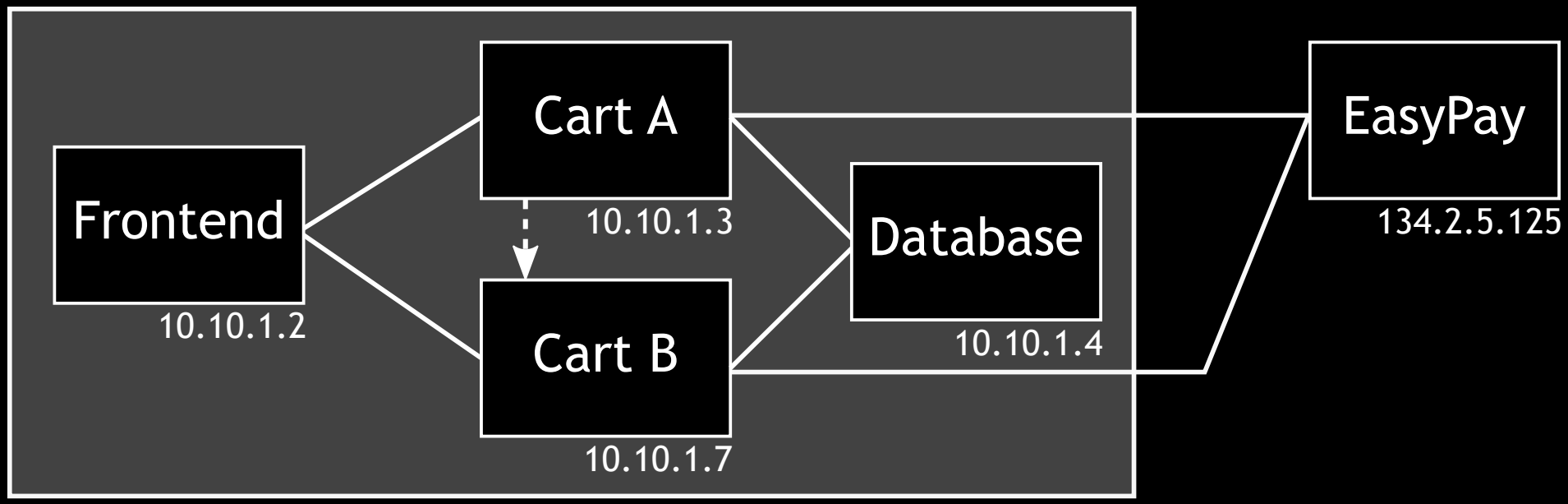


deploy Cart B, configure Database

Example Scenarios

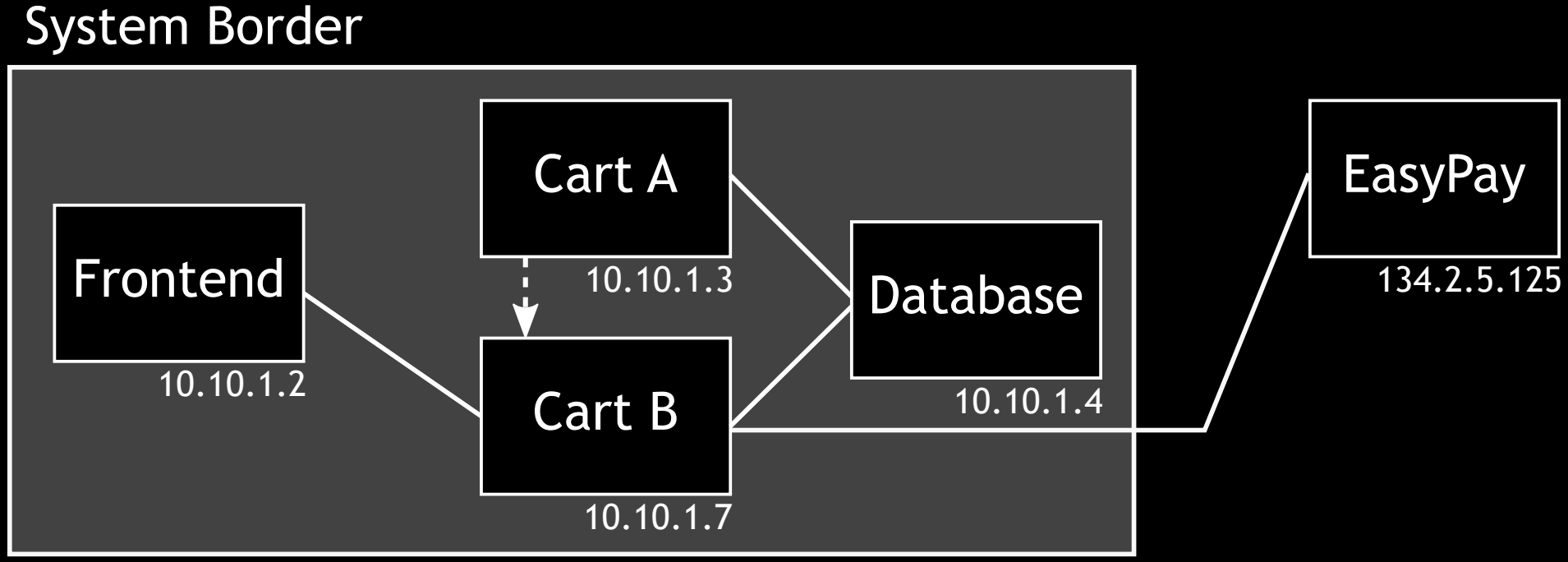


Example Scenarios



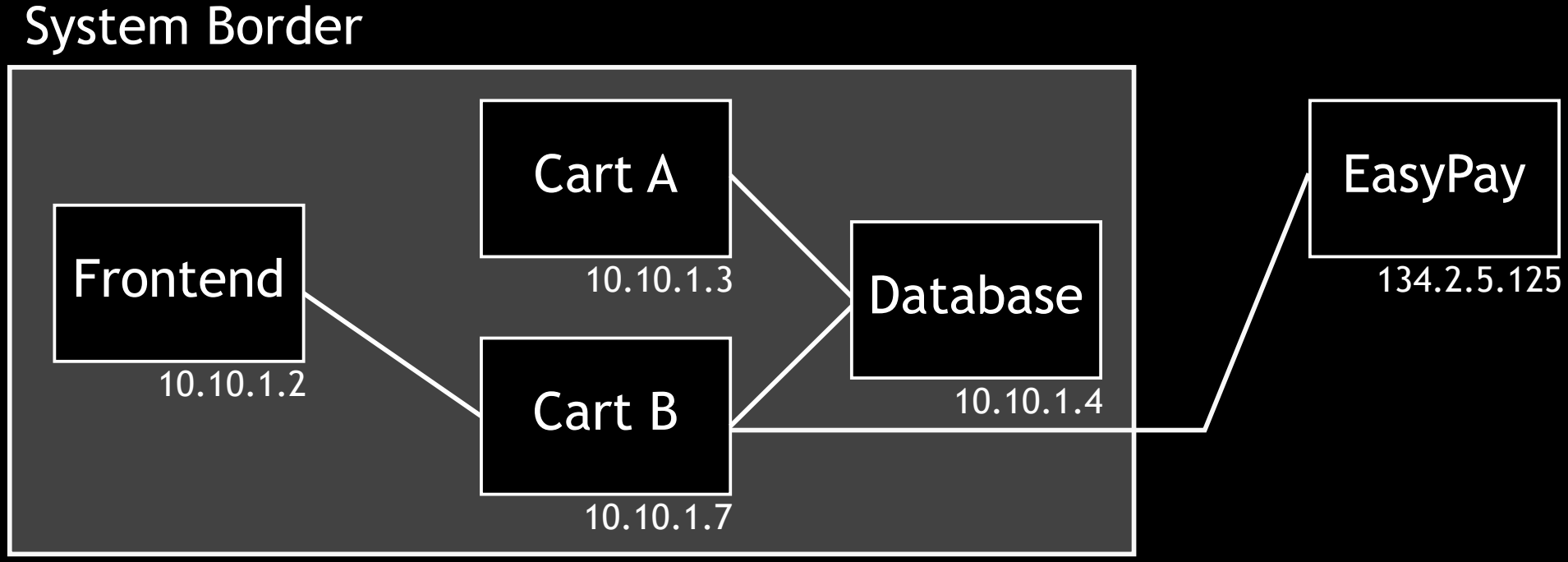
connect Cart B, update state

Example Scenarios

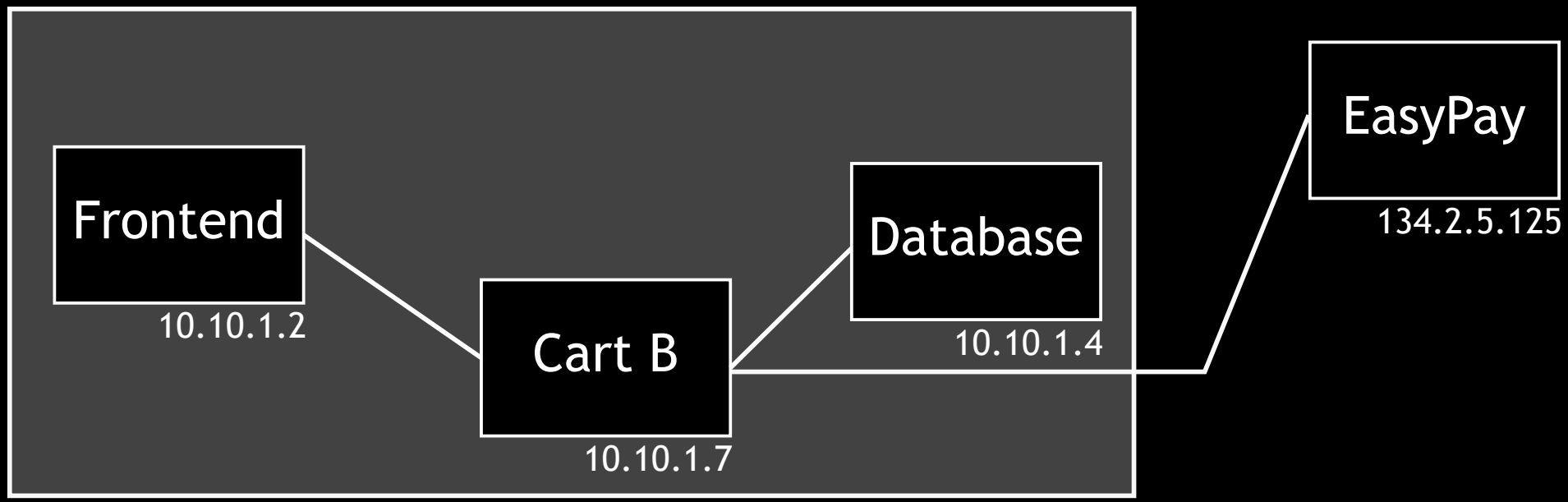


disconnect Cart A

Example Scenarios

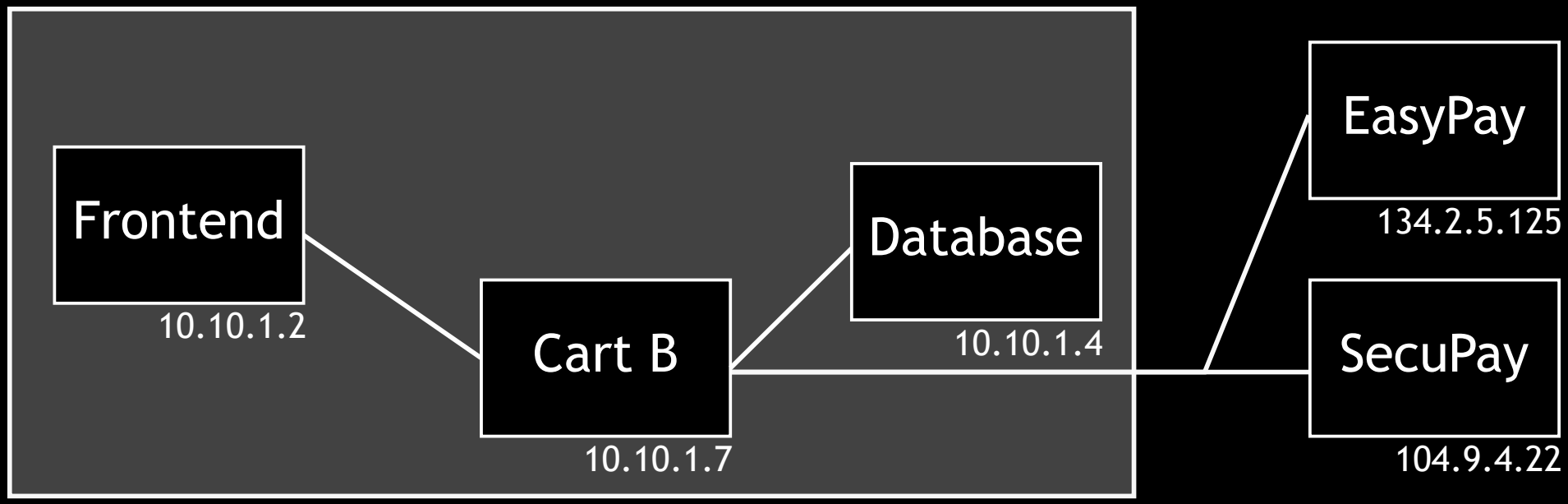


Example Scenarios



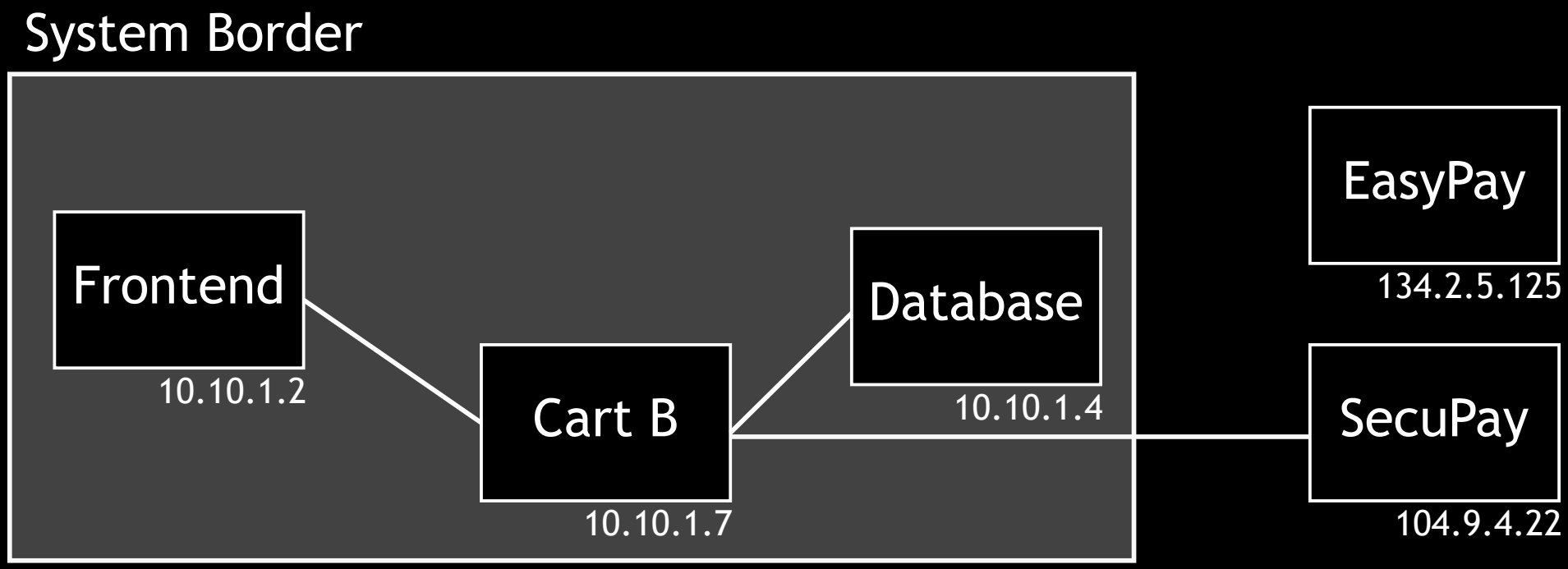
undeploy Cart A

Example Scenarios



configure Cart B to use SecuPay

Example Scenarios

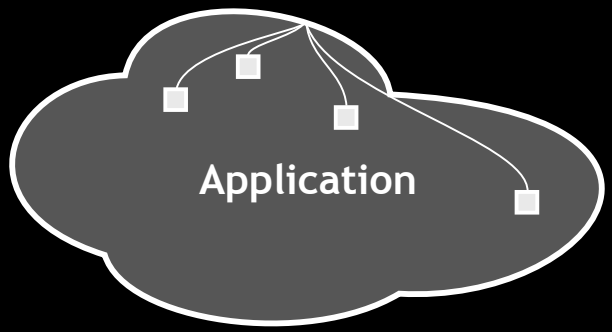


confiugure Cart B to no longer use EasyPay

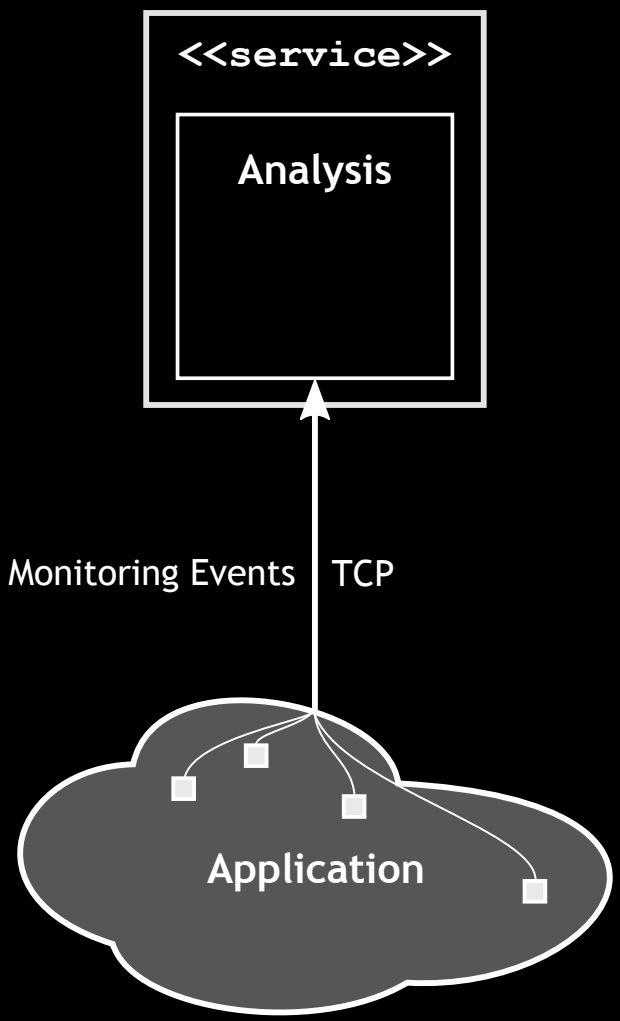
Solution

Divide adaptions actions in smaller execution actions

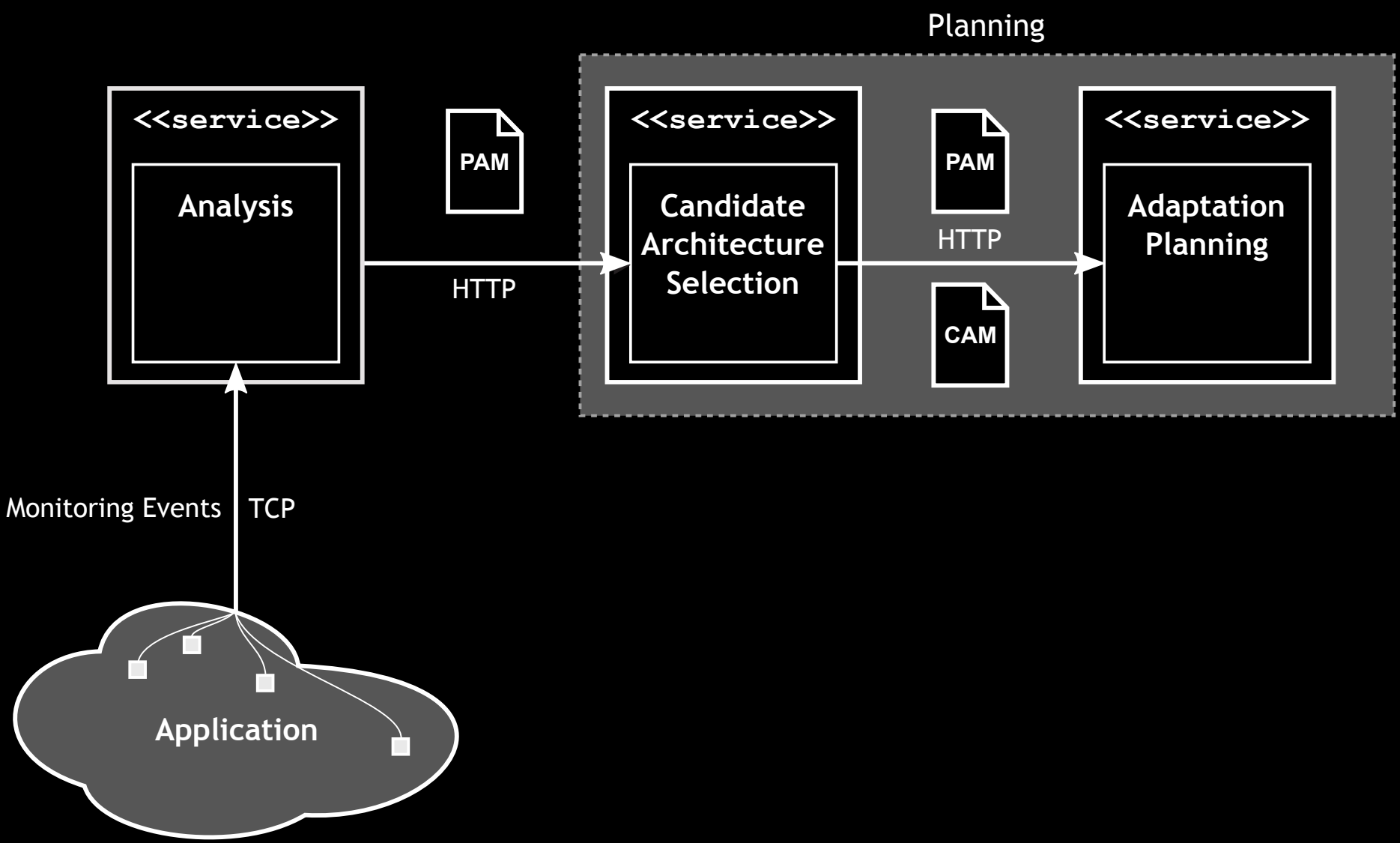
iObserve



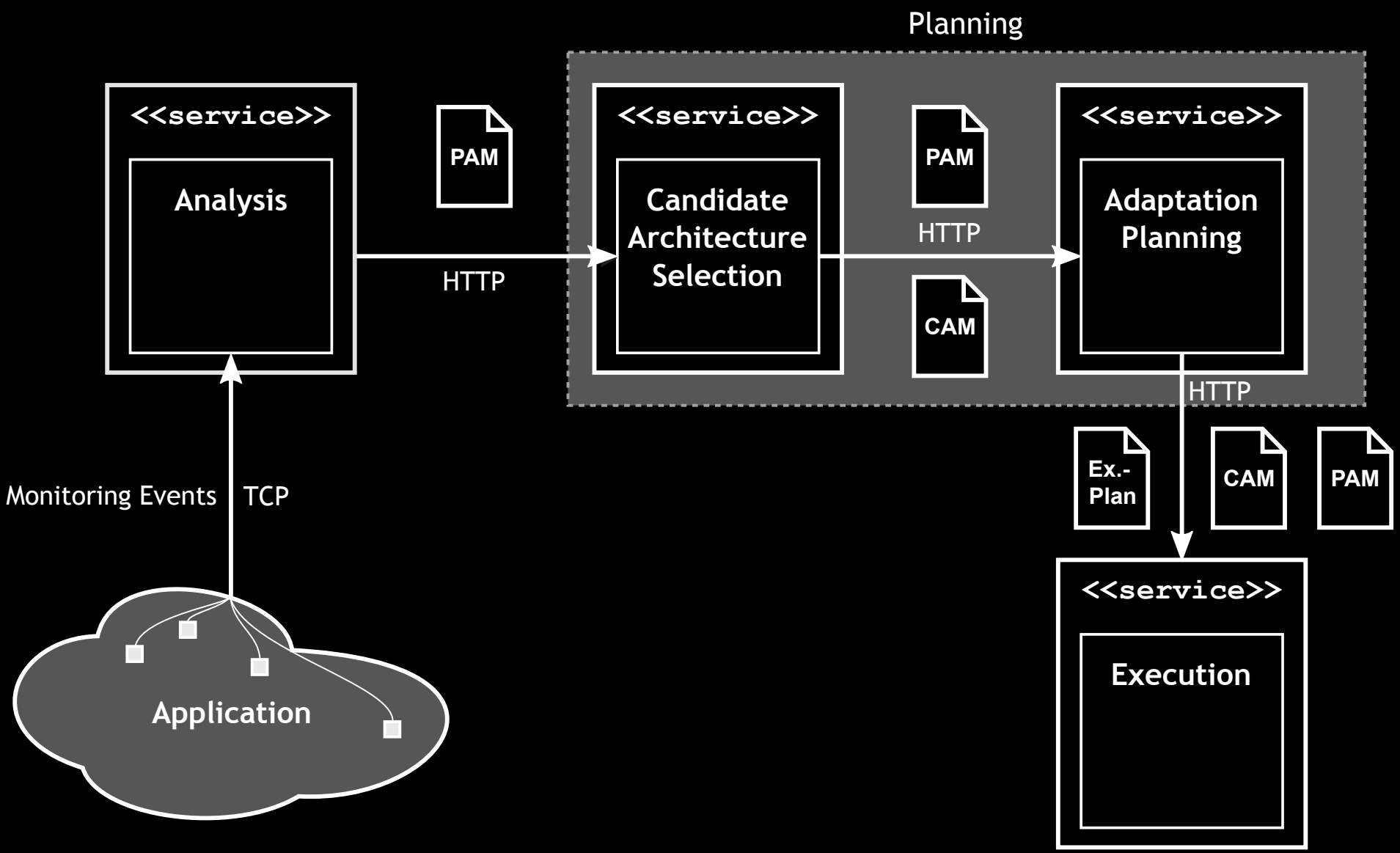
iObserve



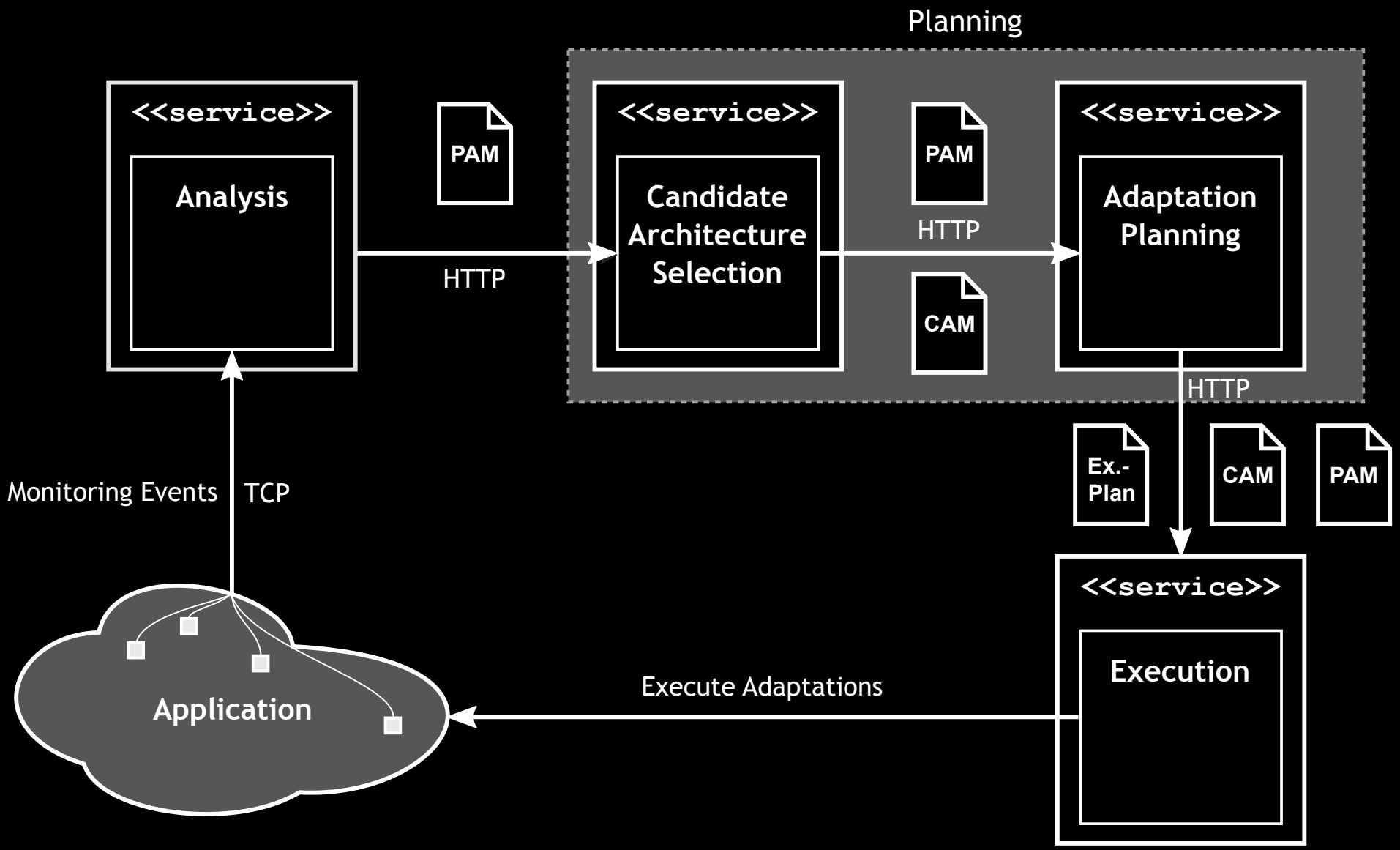
iObserve


Planning
<<service>>
Analysis
PAM
HTTP
<<service>>
Candidate Architecture Selection
PAM
HTTP
CAM
<<service>>
Adaptation Planning
Monitoring Events
TCP
Application

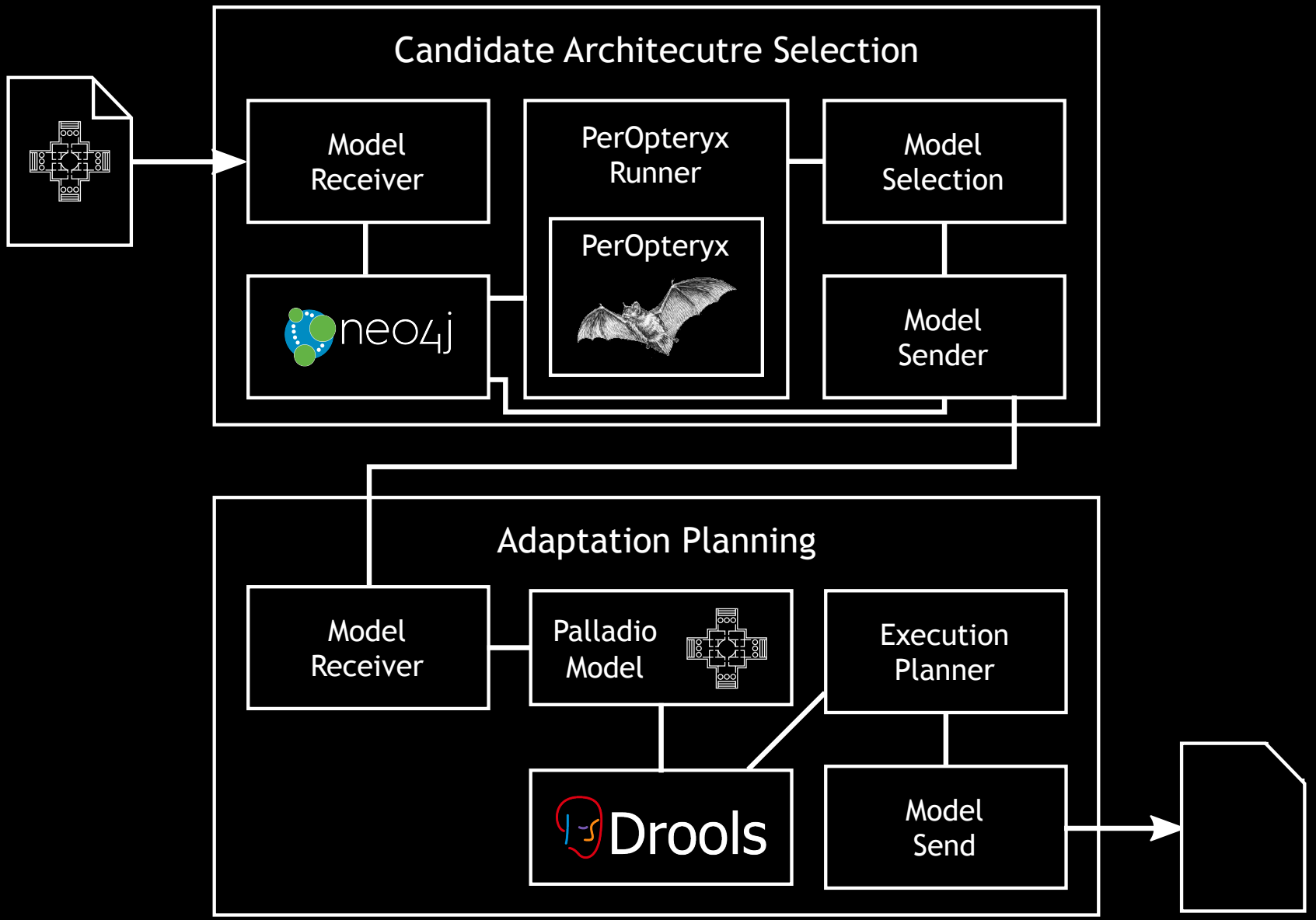
iObserve


Planning
<<service>>
Analysis
PAM
HTTP
<<service>>
Candidate Architecture Selection
PAM
CAM
HTTP
<<service>>
Adaptation Planning
HTTP
Ex.-Plan
CAM
PAM
<<service>>
Execution
Monitoring Events
TCP
Application

iObserve


Planning
<<service>>
Analysis
PAM
<<service>>
Candidate Architecture Selection
PAM
HTTP
<<service>>
Adaptation Planning
HTTP
CAM
HTTP
Ex.-Plan
CAM
PAM
Monitoring Events
TCP
<<service>>
Execution
Execute Adaptations
Application

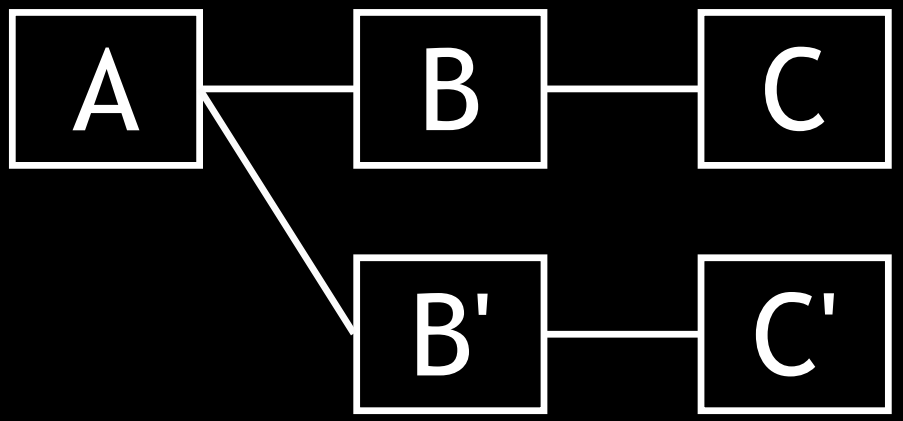
Planning


Candidate Architecutre Selection
Model Receiver
PerOpteryx Runner
PerOpteryx
Model Selection
neo4j
Model Sender
Adaptation Planning
Model Receiver
Palladio Model
Execution Planner
Drools
Model Send

Adaptation Planning

Compute composed adaptation actions

- Difference PAM vs. CAM
- Consider data dependencies
- Consider state



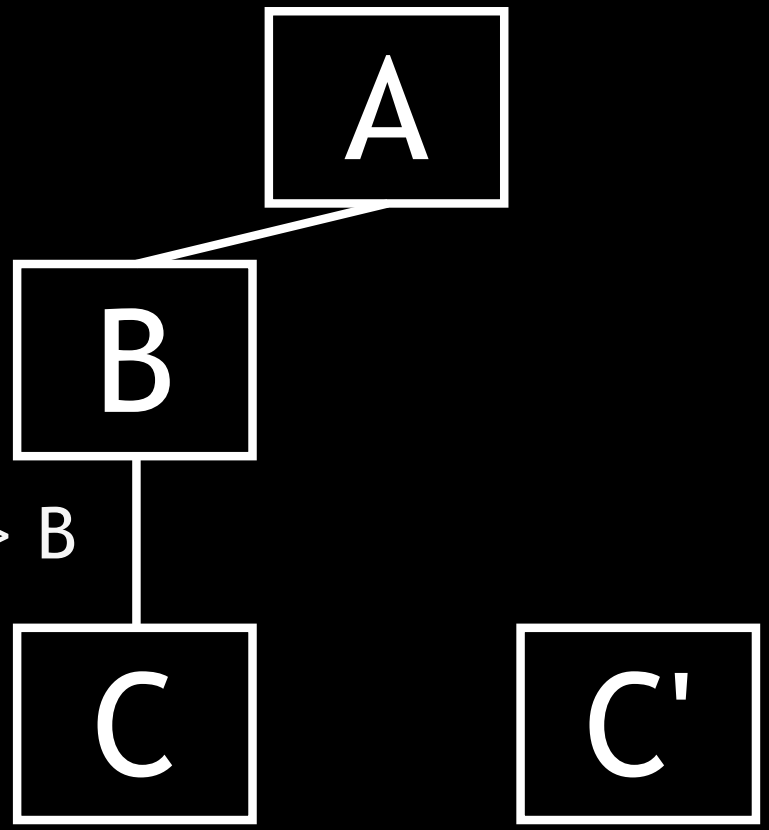
↓B	↓B	↓B	↓B	↓B	↓B	…	↑C'
↑B'	↑B'	↓C	↓C	↑C'	↑C'		↑B'
↓C	↑C'	↑B'	↑C'	↑B'	↓C		↓B
↑C'	↓C	↑C'	↑B'	↓C	↑B'		↓C

4! = 24 options, only **two** safe

Generating atomic actions

Migration of B to B'

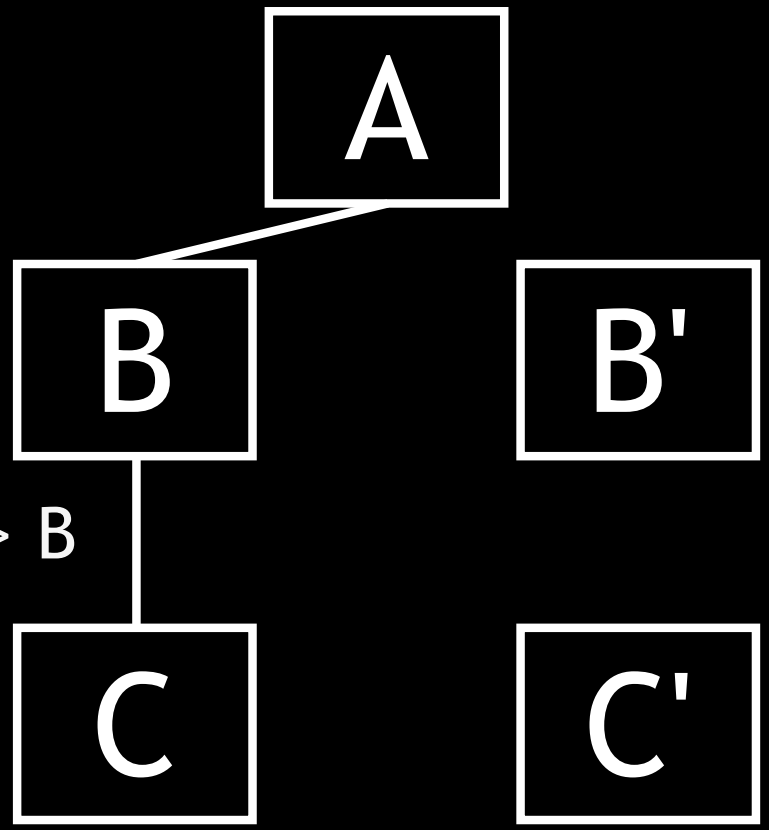
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

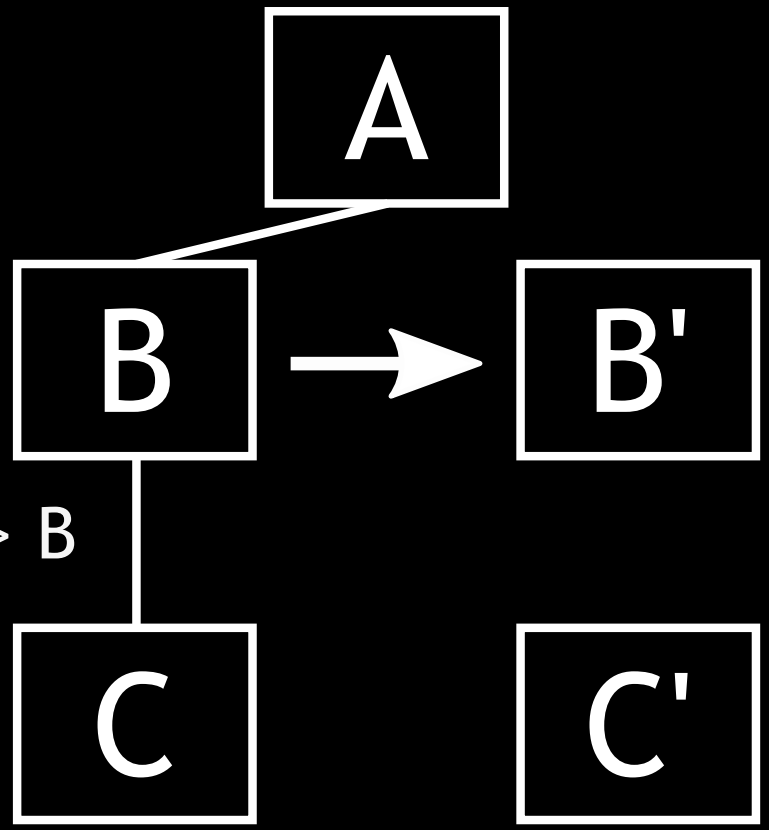
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

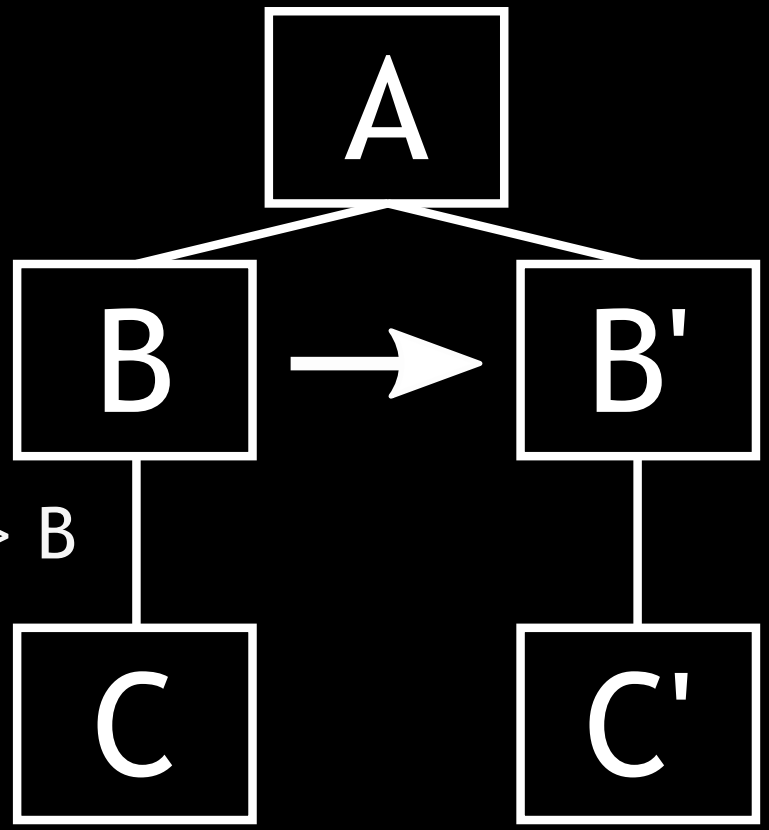
- deploy B'
- start state sync
  - connect B' -> C'
  - connect A -> B'
  - disallow new sessions from A -> B
  - wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

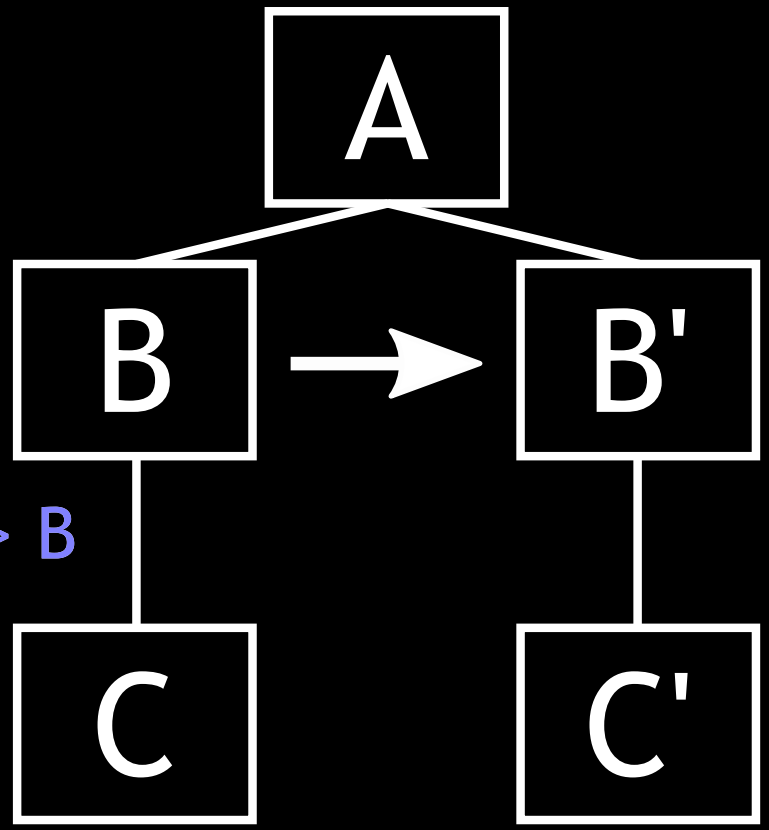
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

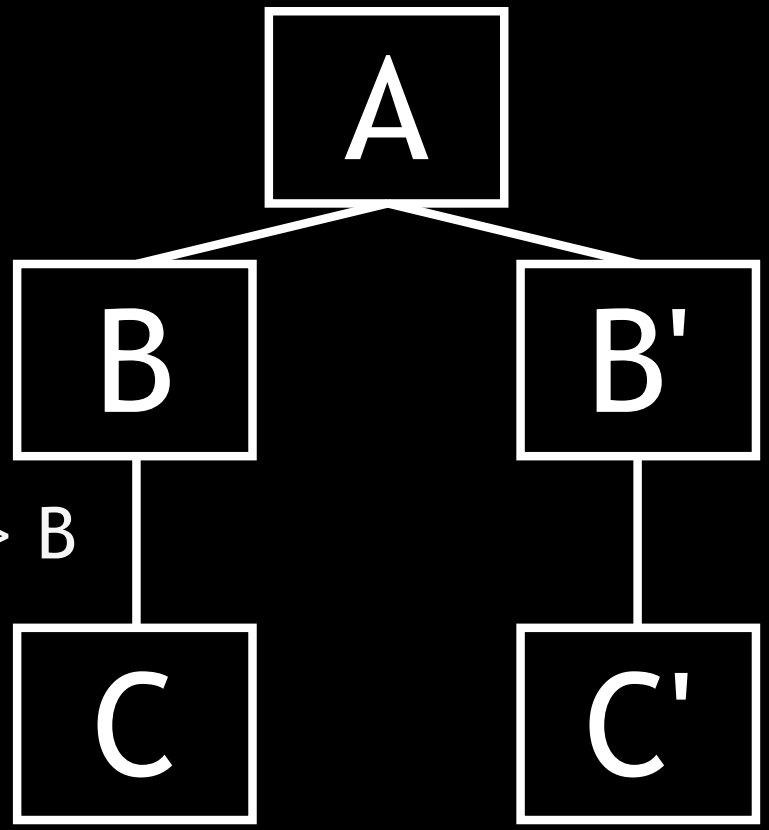
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

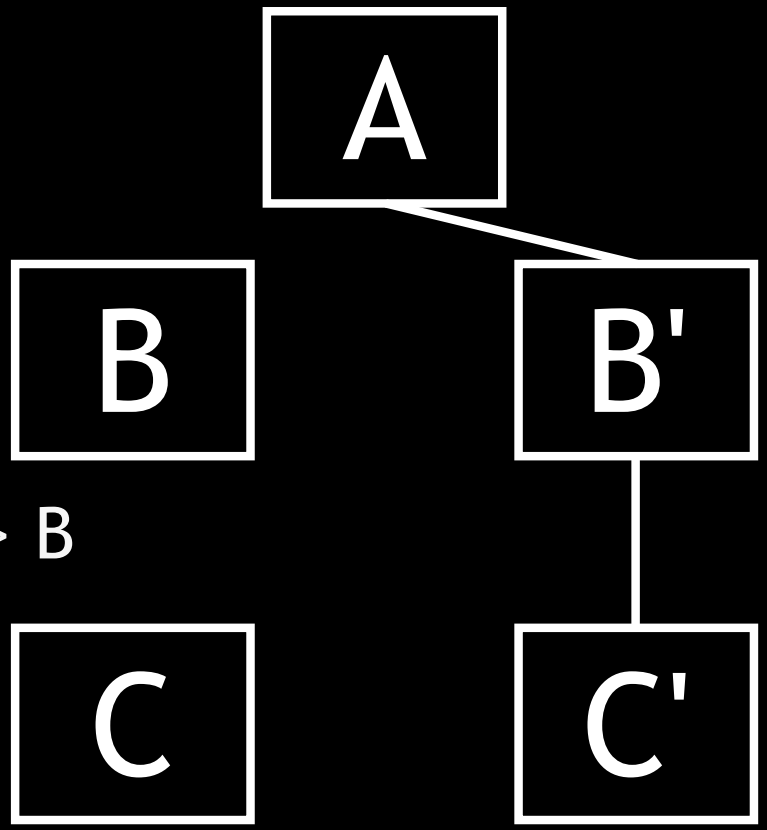
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

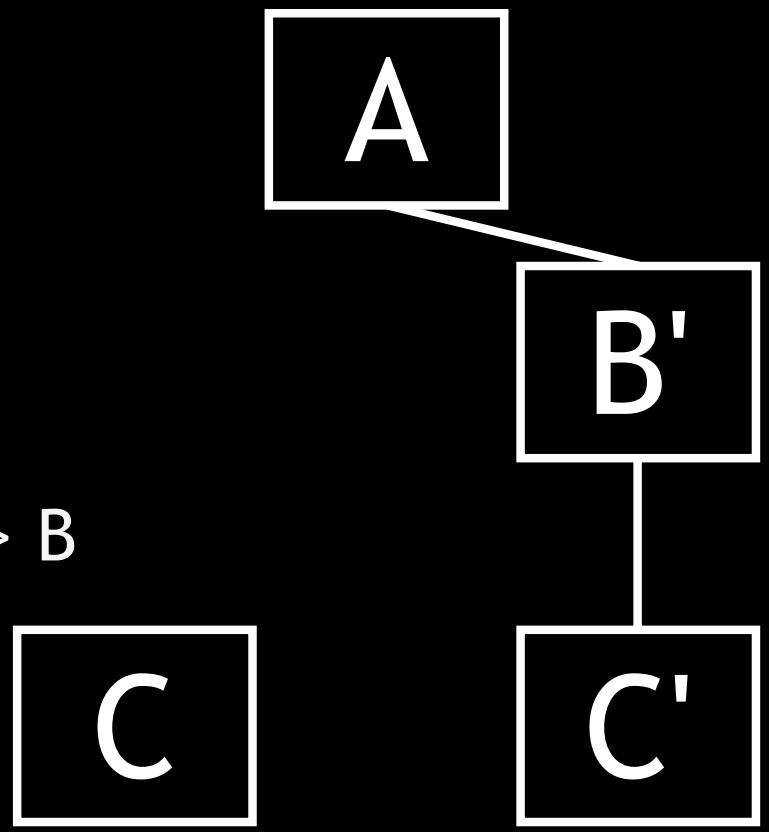
- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Generating atomic actions

Migration of B to B'

- deploy B'
- start state sync
- connect B' -> C'
- connect A -> B'
- disallow new sessions from A -> B
- wait for session completion
- complete sync
- disconnect A -> B
- disconnect B -> C
- undeploy B



Execution

Service

- Map atomic action to cloud API calls
- Execute actions without violating data constraints
  - Currently sequential execution
  - Future concurrent execution

Mapping

- Depends on cloud API
- Depends on service features
  - State transfer
  - Availability checks

Conclusion

Summary

- Introduced iObserve services for MAPE-K
  - Candidate Selection
  - Planning
  - Execution
- Rule-based adaption with Drools

Code

`https://github.com/research-iobserve/iobserve-analysis`

Outlook

- Concurrent execution
- Additional atomic actions
- More executors supporting different techologies
- Use of Kieker probes to check availability of services