Moscow
2018

Dmitry Patsura
Fintier

Microservice Architecture

Dmitry Patsura

Пишу на много чем)

https://github.com/ovr

GHubber

PHPSA

StaticScript

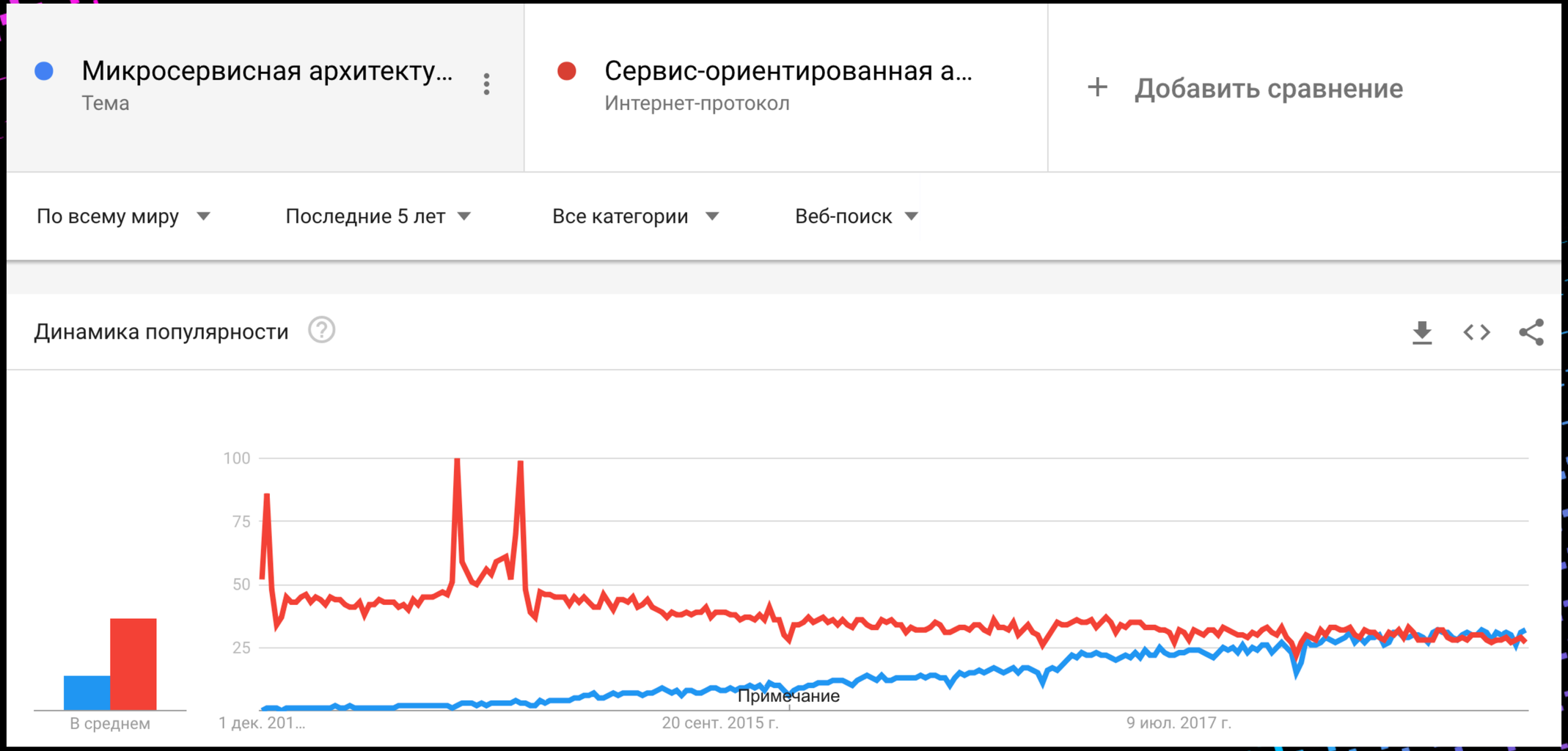
FINTIER

Helping institutions and organisations to develop a marketplace, provide innovative financial services and accept payments worldwide.

FundSource MobisCard intonet Creditin Insidex

Microservice Architecture?

Microservice Architecture?



Hype....

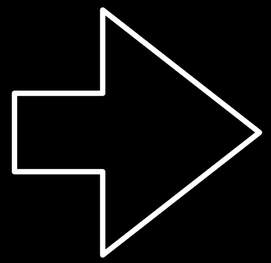
Hype....

From my experience

From my experience

From my experience

From my experience




HOLY
JS/...

Plan

Plan

» Architectures overview

Plan

» Architectures overview

» Microservice Architecture?

Plan

» Architectures overview
» Microservice Architecture?
» Positive and negative sides

Plan

» Architectures overview
» Microservice Architecture?
» Positive and negative sides
» Teamwork

Plan

» Architectures overview
» Microservice Architecture?
» Positive and negative sides
» Teamwork
» Dependecy control

Plan

» Architectures overview
» Microservice Architecture?
» Positive and negative sides
» Teamwork
» Dependecy control
» Internal communication in MSA

Plan

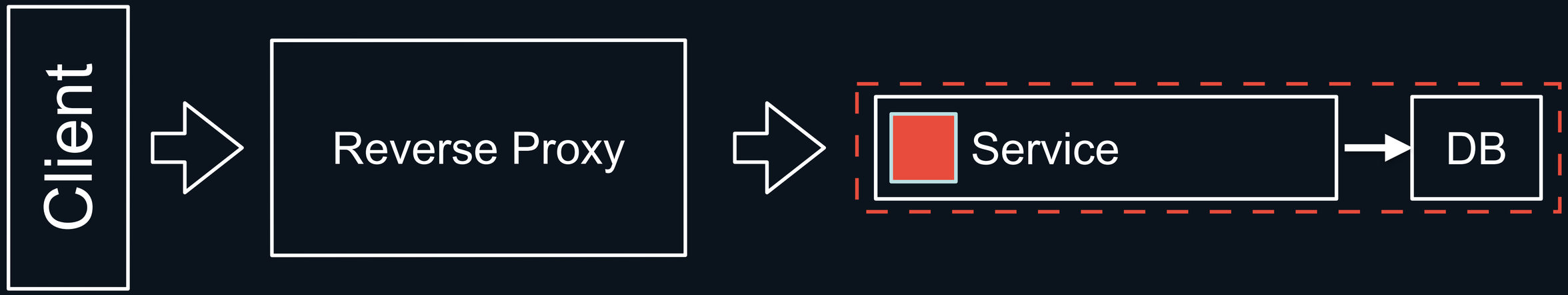
» Architectures overview
» Microservice Architecture?
» Positive and negative sides
» Teamwork
» Dependecy control
» Internal communication in MSA
» Service Architecture (code examples)

Plan

» Architectures overview
» Microservice Architecture?
» Positive and negative sides
» Teamwork
» Dependecy control
» Internal communication in MSA
» Service Architecture (code examples)
» Delusions

Architecture overview

Monolithic Backend



Problems

Problems

» Fault tolerance

Problems

» Fault tolerance
» Horizontal scaling

Problems

» Fault tolerance
» Horizontal scaling
» One technology/language/stack

Problems

» Fault tolerance
» Horizontal scaling
» One technology/language/stack

Performance

Problems

» Fault tolerance
» Horizontal scaling
» One technology/language/stack
» All code in one place -> hard to refactoring/legacy

Performance

Problems

» Fault tolerance
» Horizontal scaling
» One technology/language/stack
» All code in one place -> hard to refactoring/legacy
» Teamwork (how to scale?)

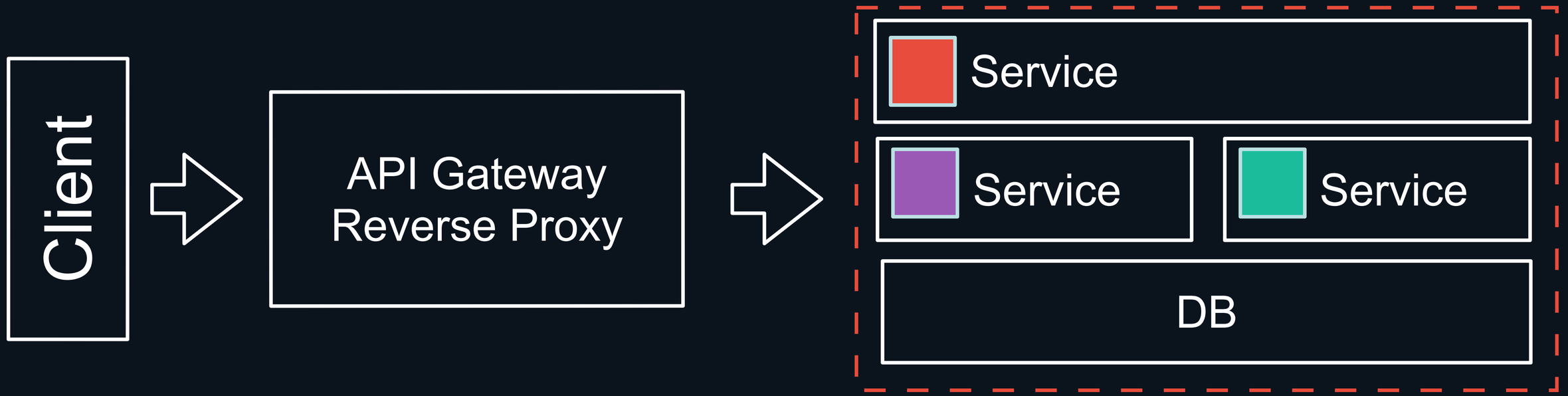
Performance

Problems

» Fault tolerance
» Horizontal scaling
» One technology/language/stack
» All code in one place -> hard to refactoring/legacy
» Teamwork (how to scale?)
» Reusage

Performance

${PROJECT_NAME} Backend



Problems

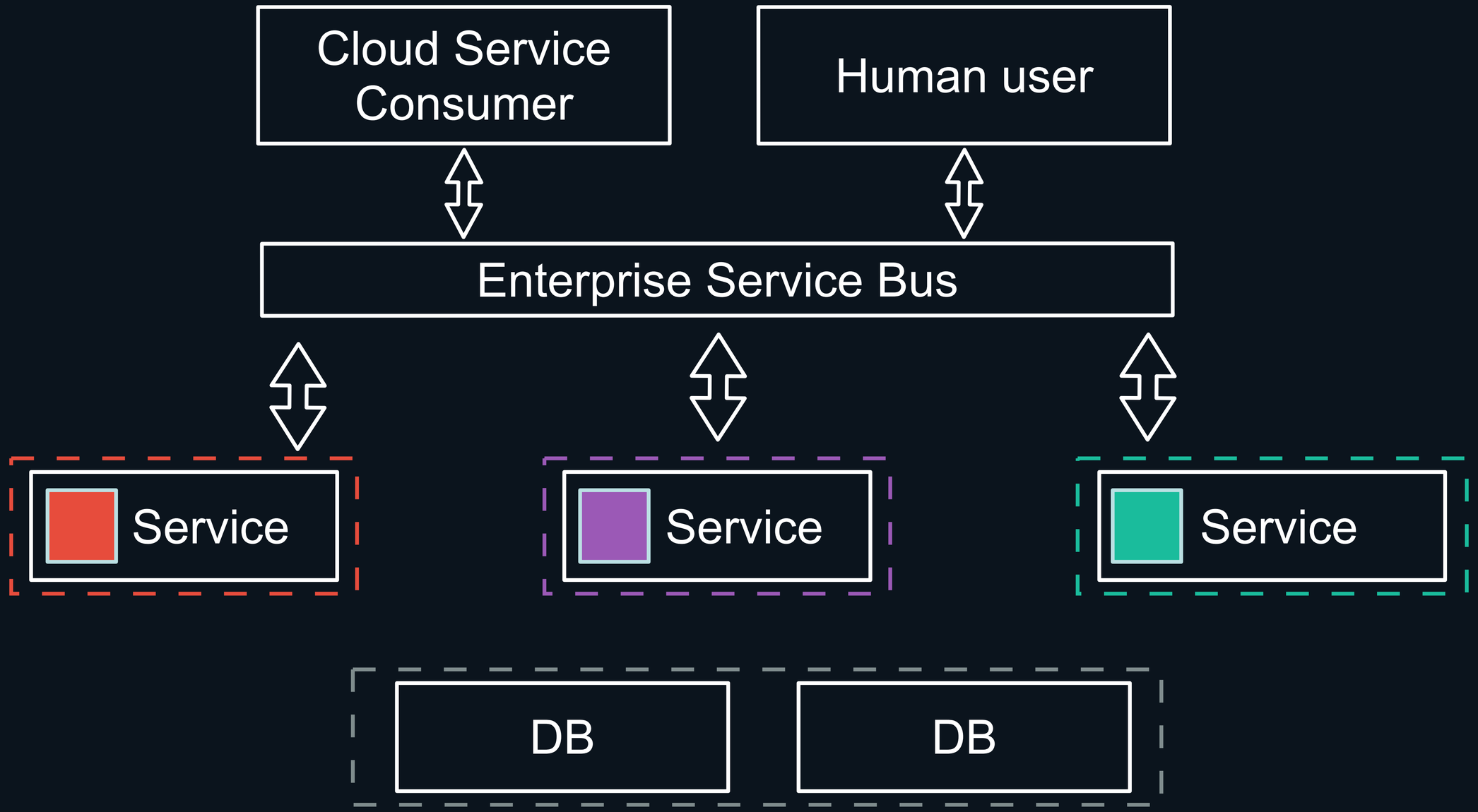
» ~~Fault~~ tolerance
» ~~Horizontal~~ scaling
» ~~One~~ technology/language/stack
» All code in one place -> hard to refactoring/legacy
» Teamwork (how to scale?)
» Reusage

Performance

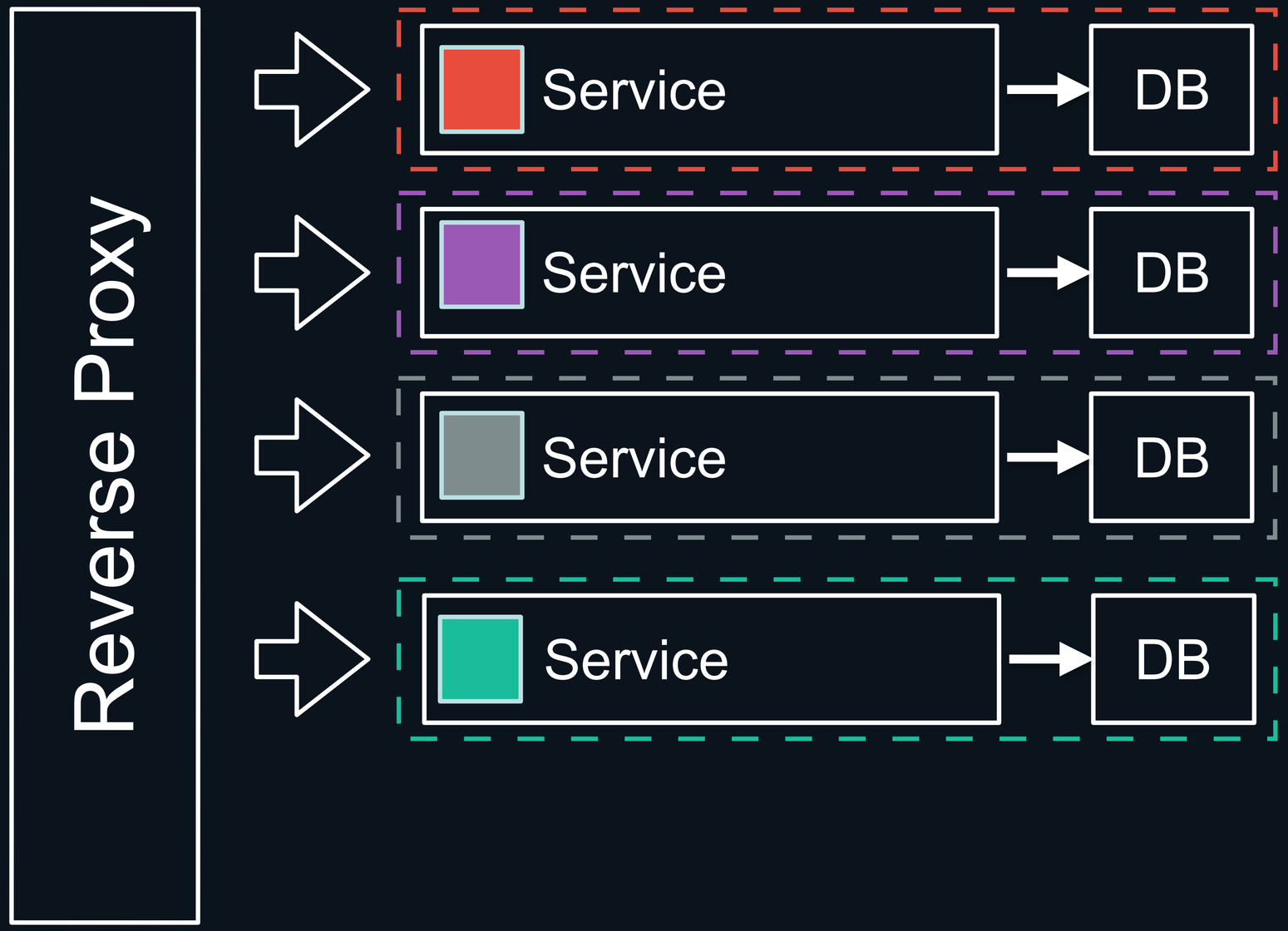
Microservice architecture

вариант сервис-ориентированной архитектуры

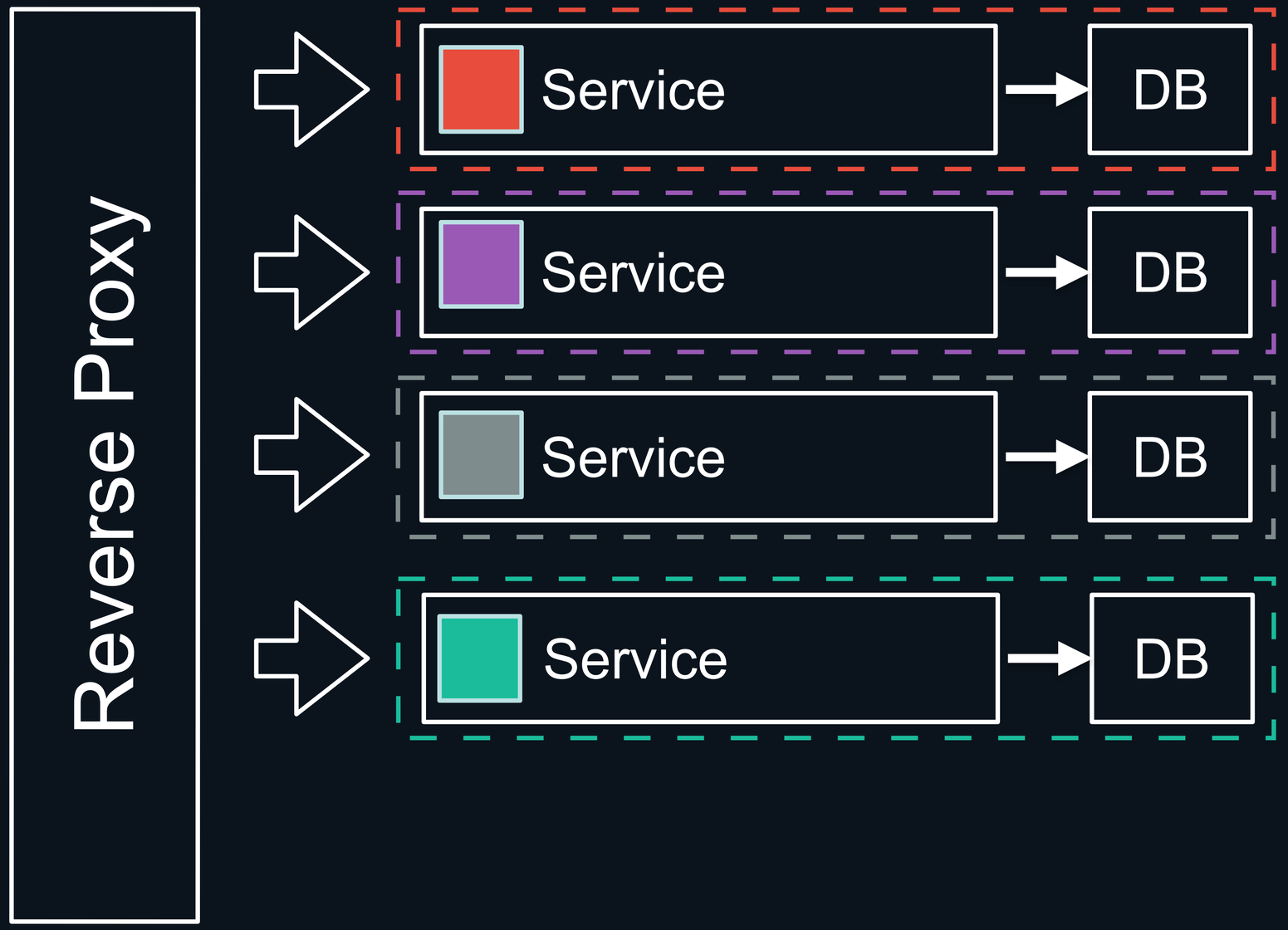
Service-oriented architecture



Microservice architecture

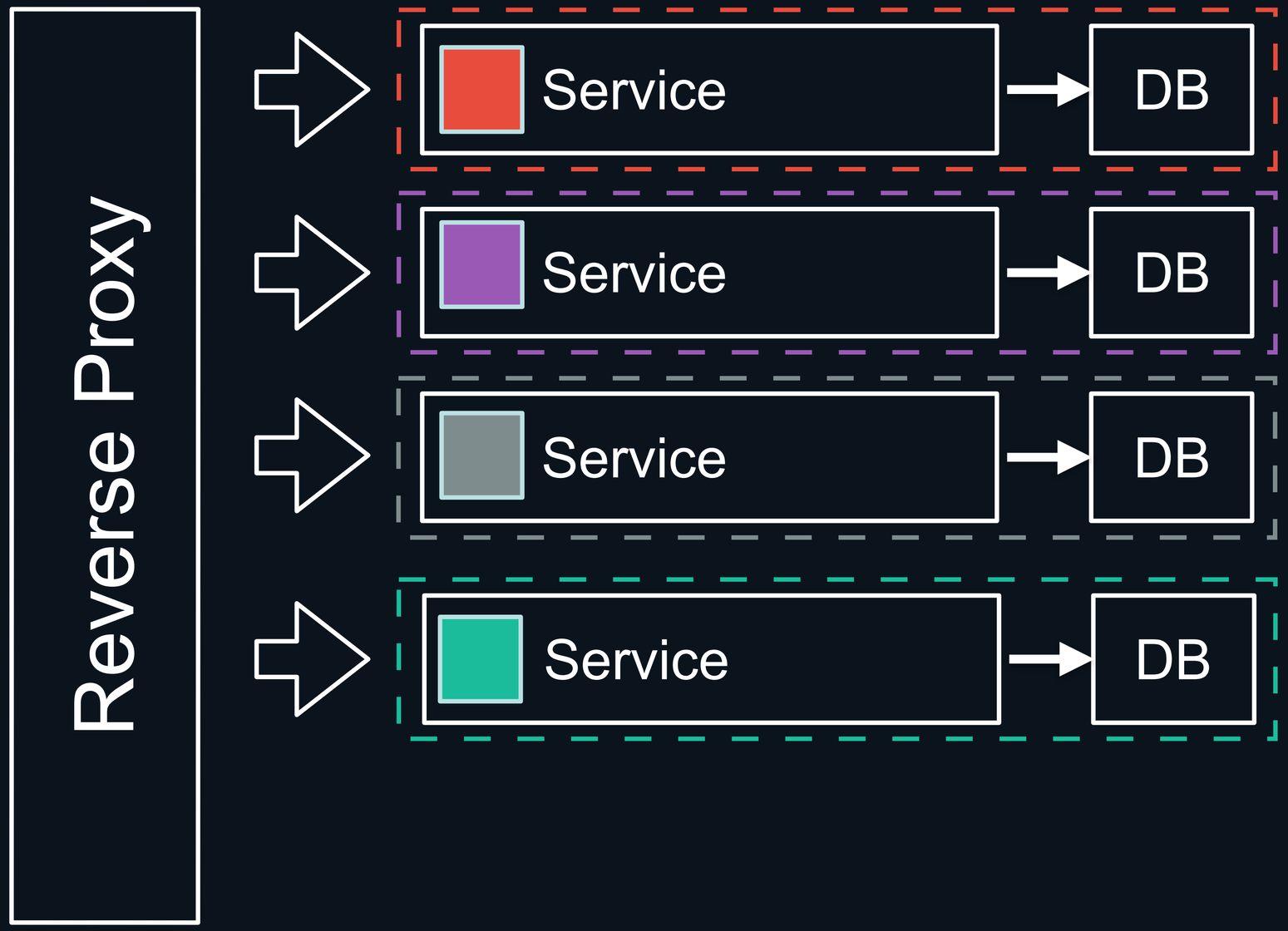


Microservice architecture



Isolation

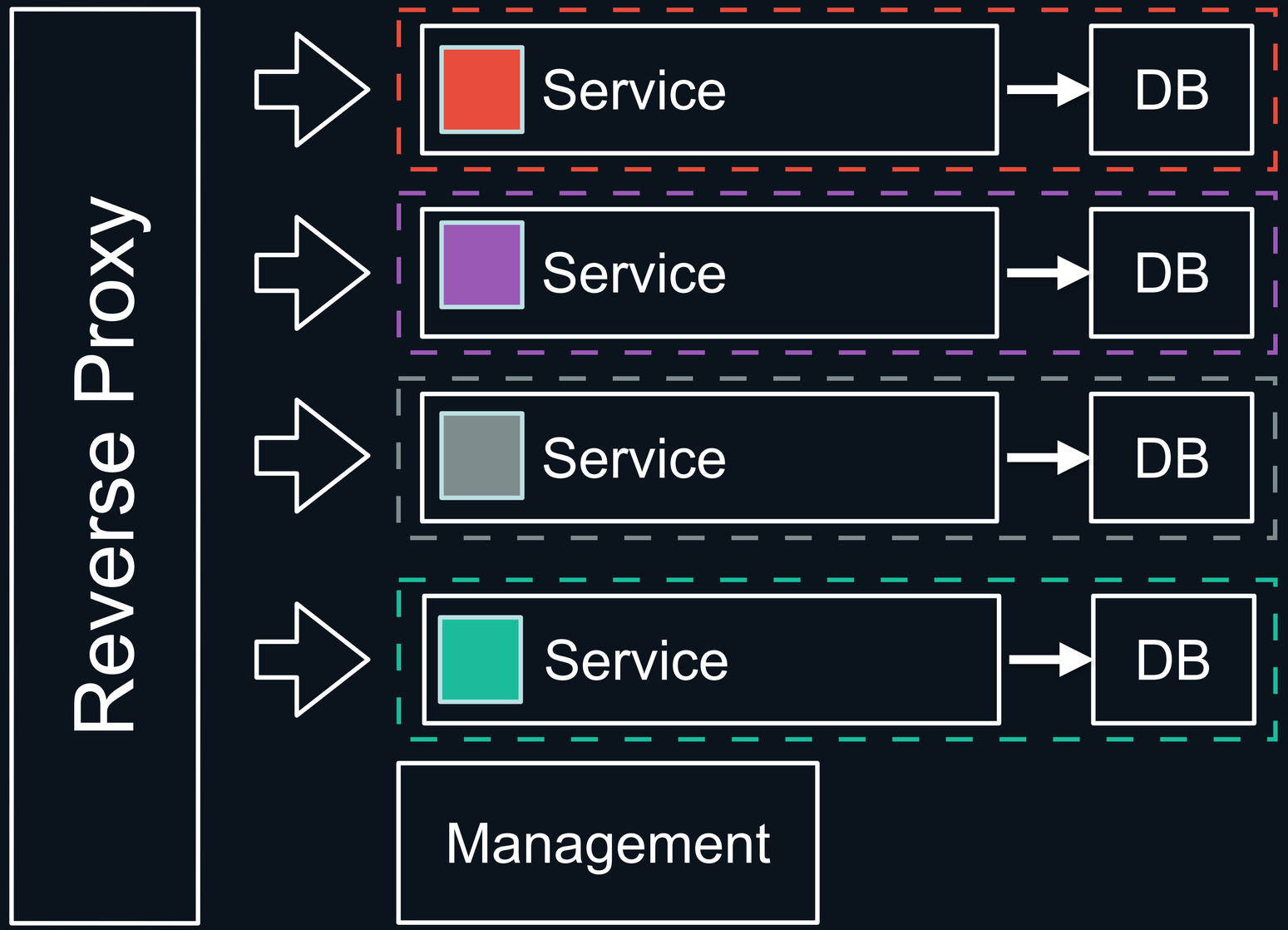
Microservice architecture



Isolation

Distributed

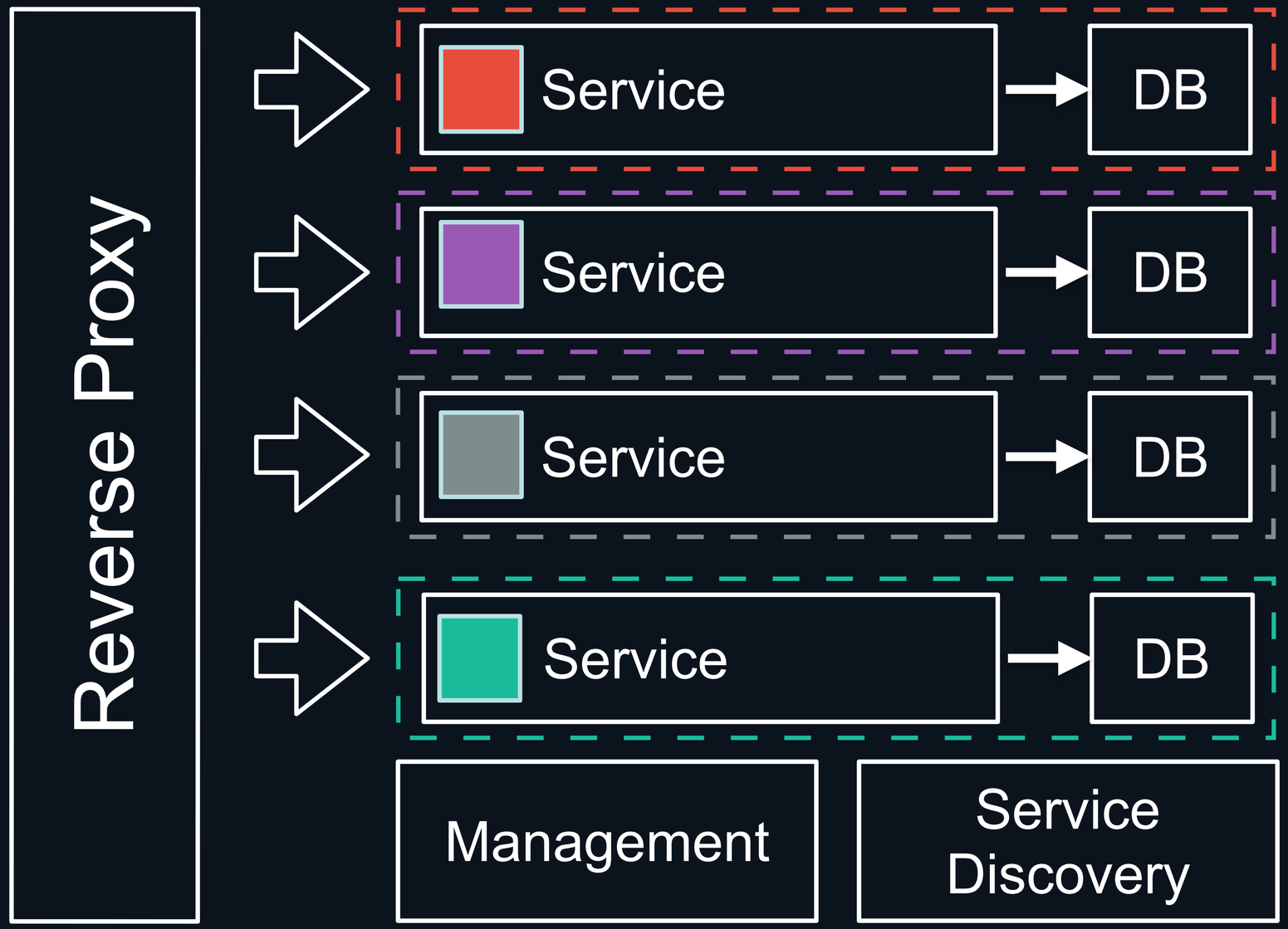
Microservice architecture



Isolation

Distributed

Microservice architecture



Isolation

Distributed

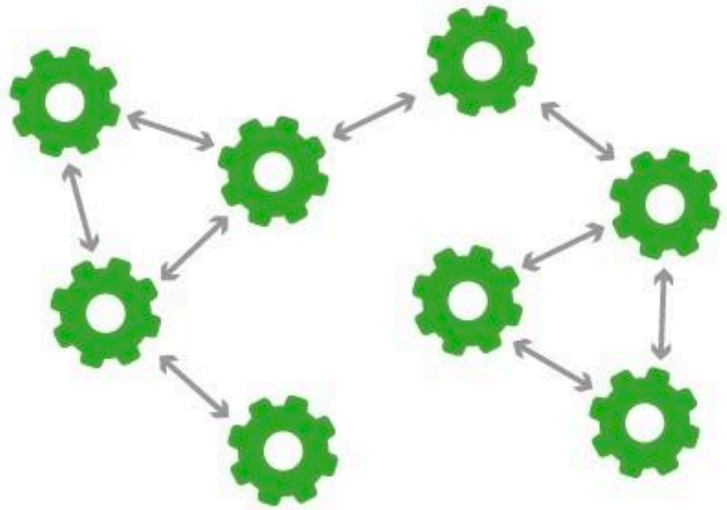
SOA vs MSA

2000's

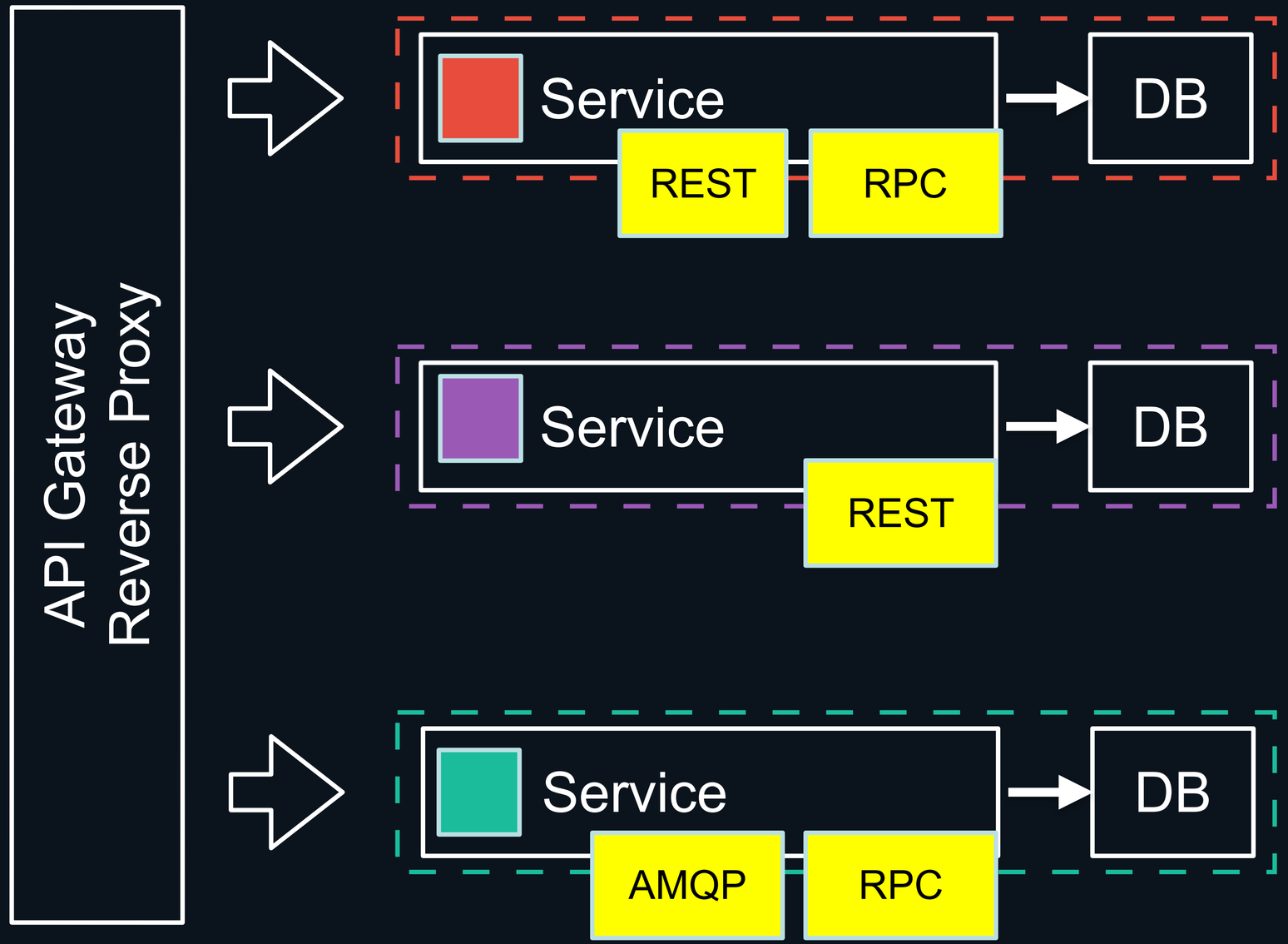
SERVICE ORIENTED ARCHITECTURE

2010's

MICROSERVICES ARCHITECTURE



Communication



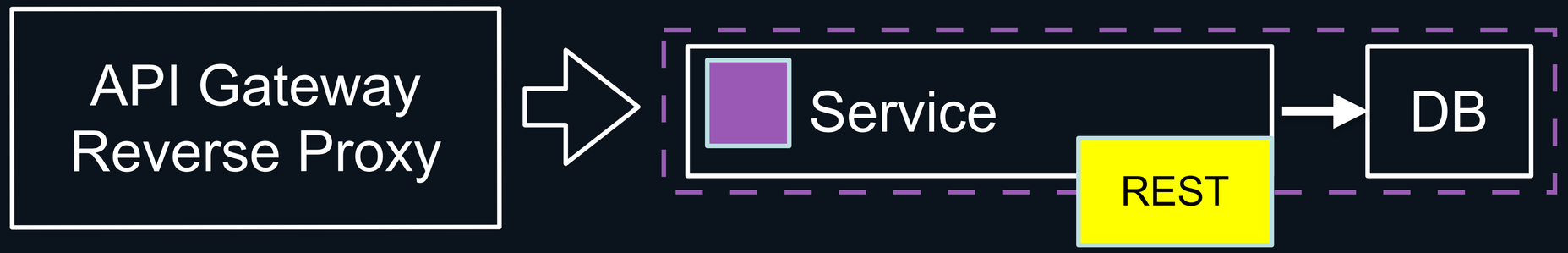
REST

RPC

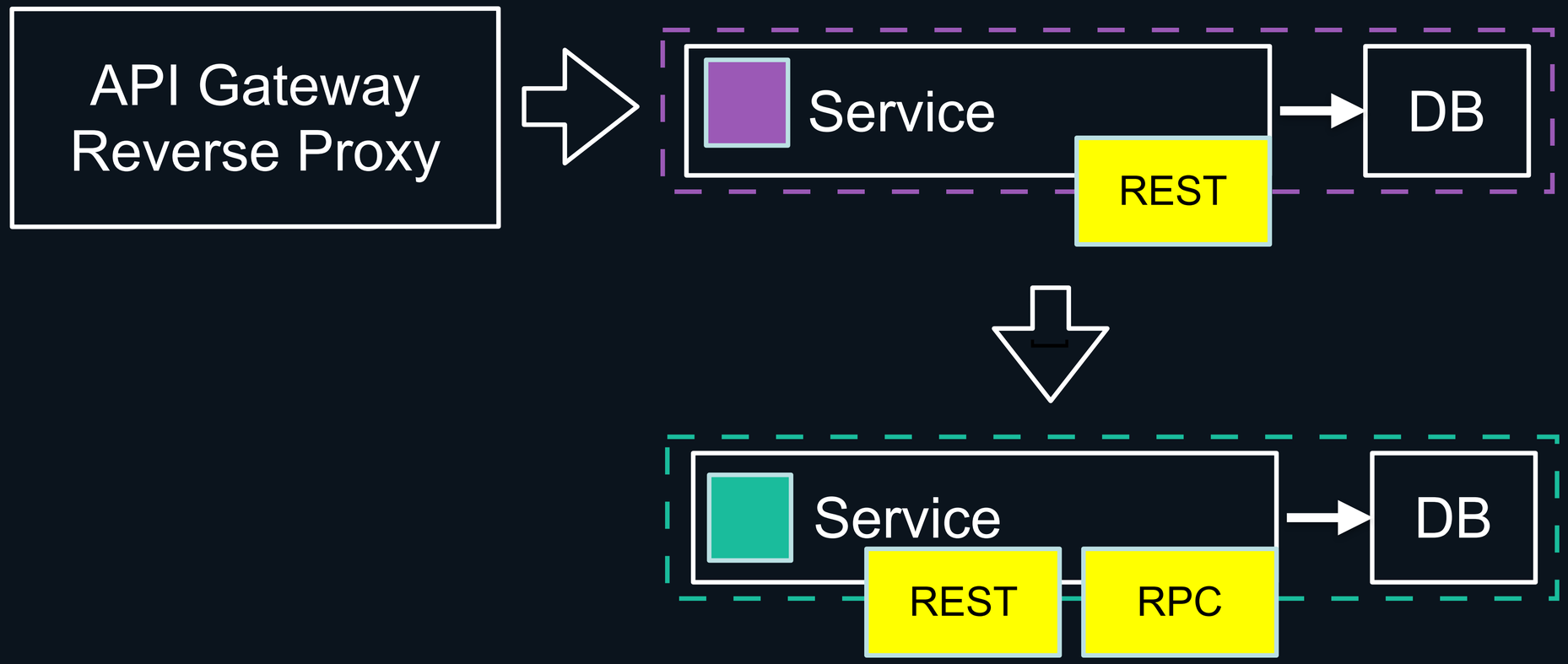
AMQP

Kafka

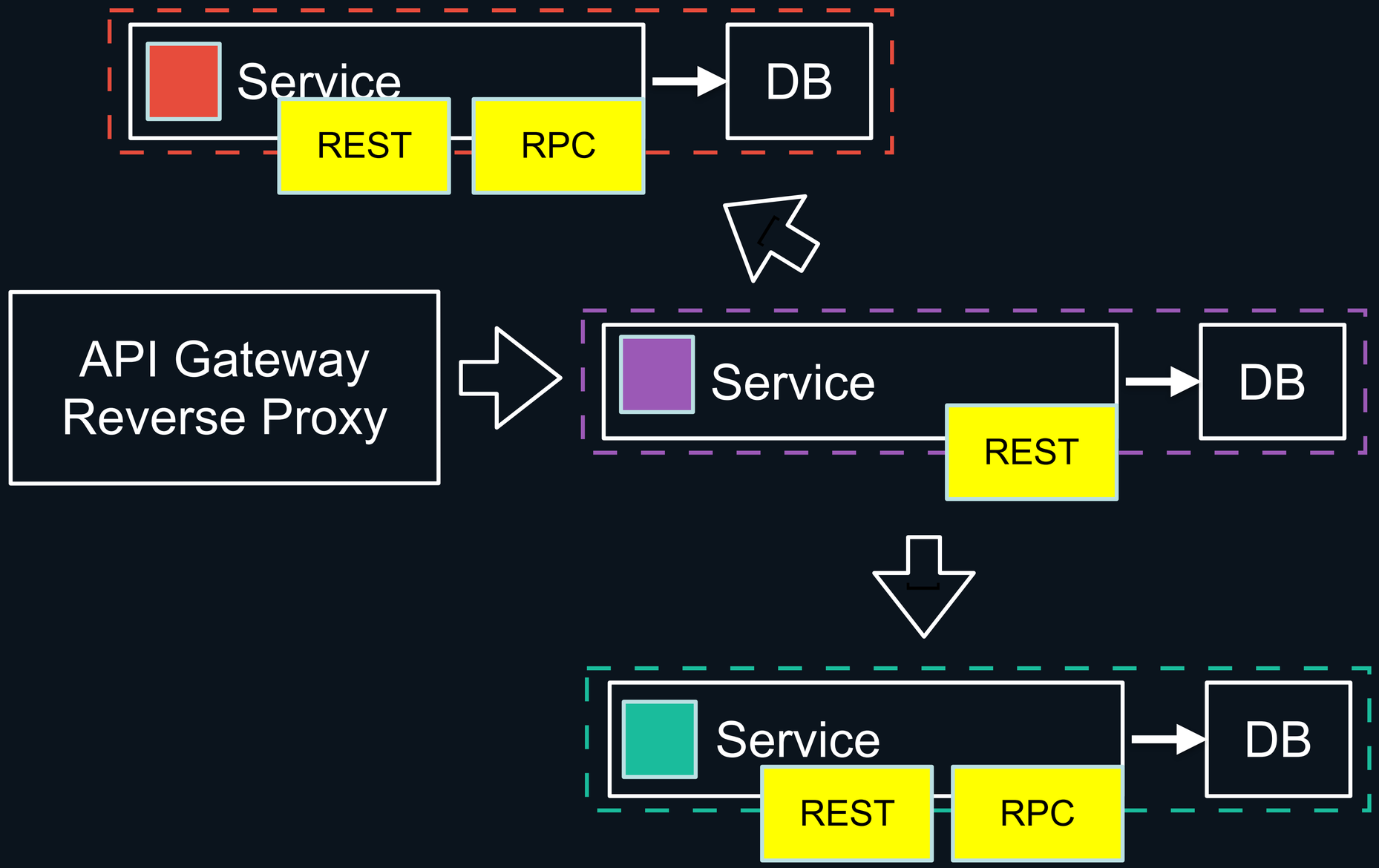
Request Execution Path



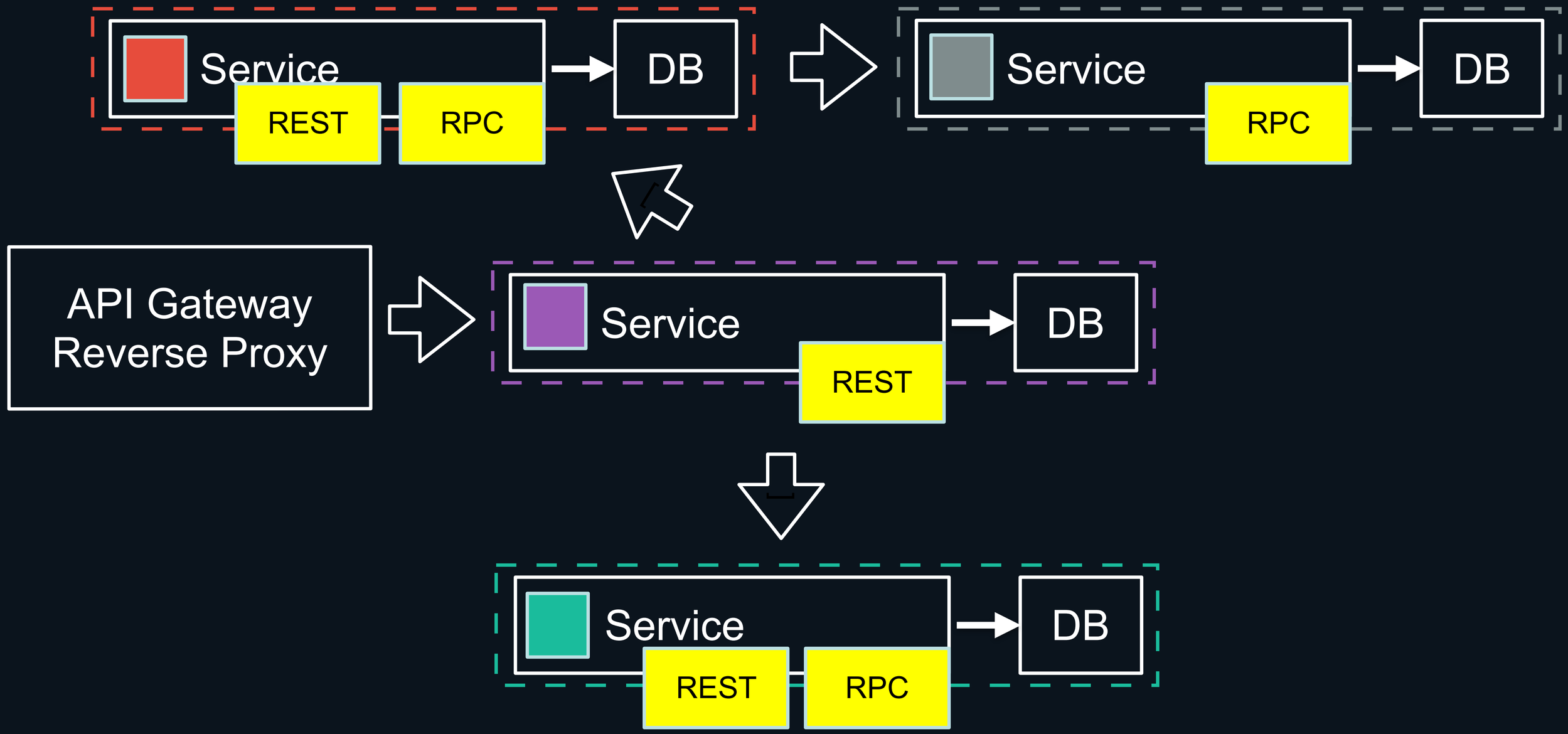
Request Execution Path



Request Execution Path



Request Execution Path



Service splitting

Service splitting

» Business context / Reusage / DRY

Service splitting

» Business context / Reusage / DRY
» Internal communications

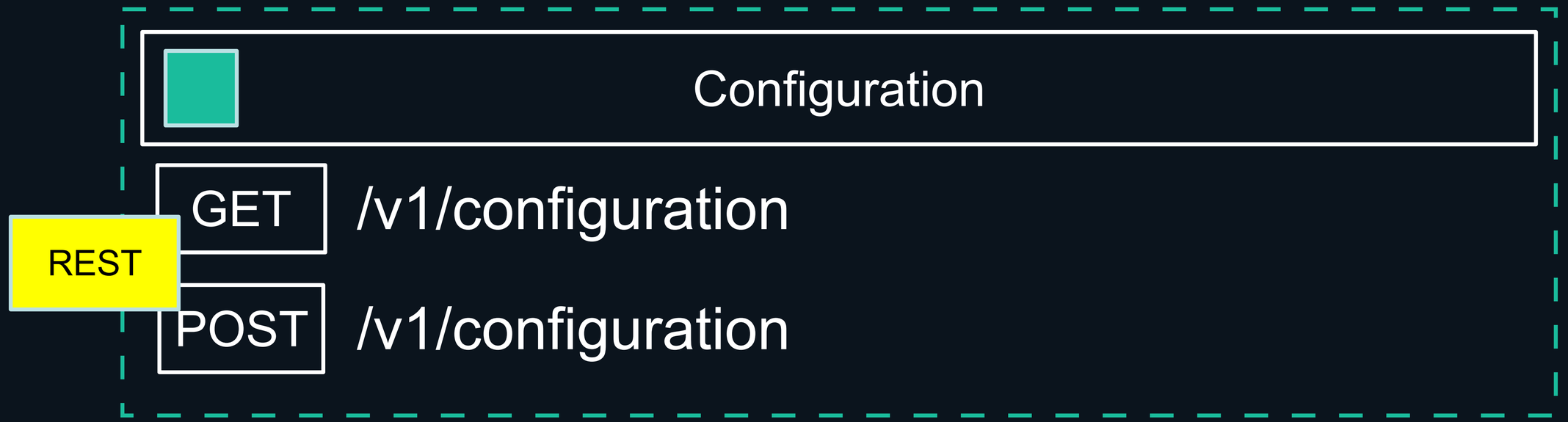
Service splitting

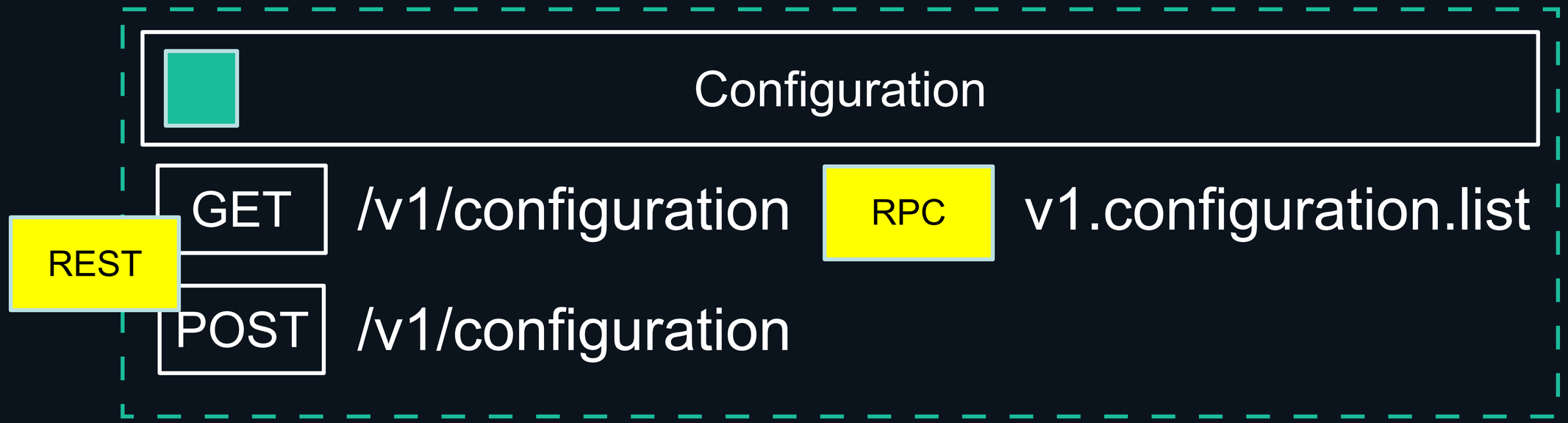
» Business context / Reusage / DRY
» Internal communications
» One DB for transactions

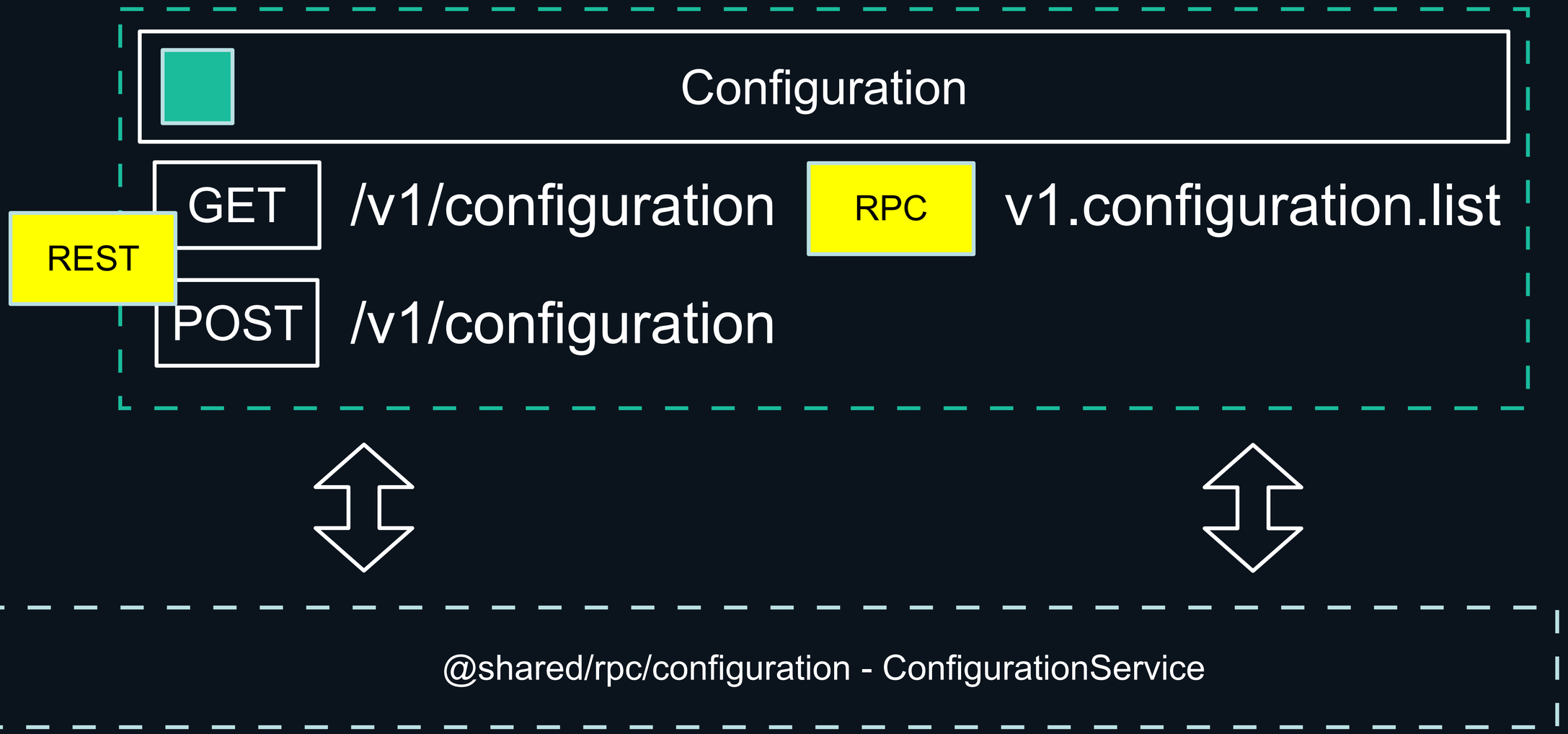
When we extract new microservice?

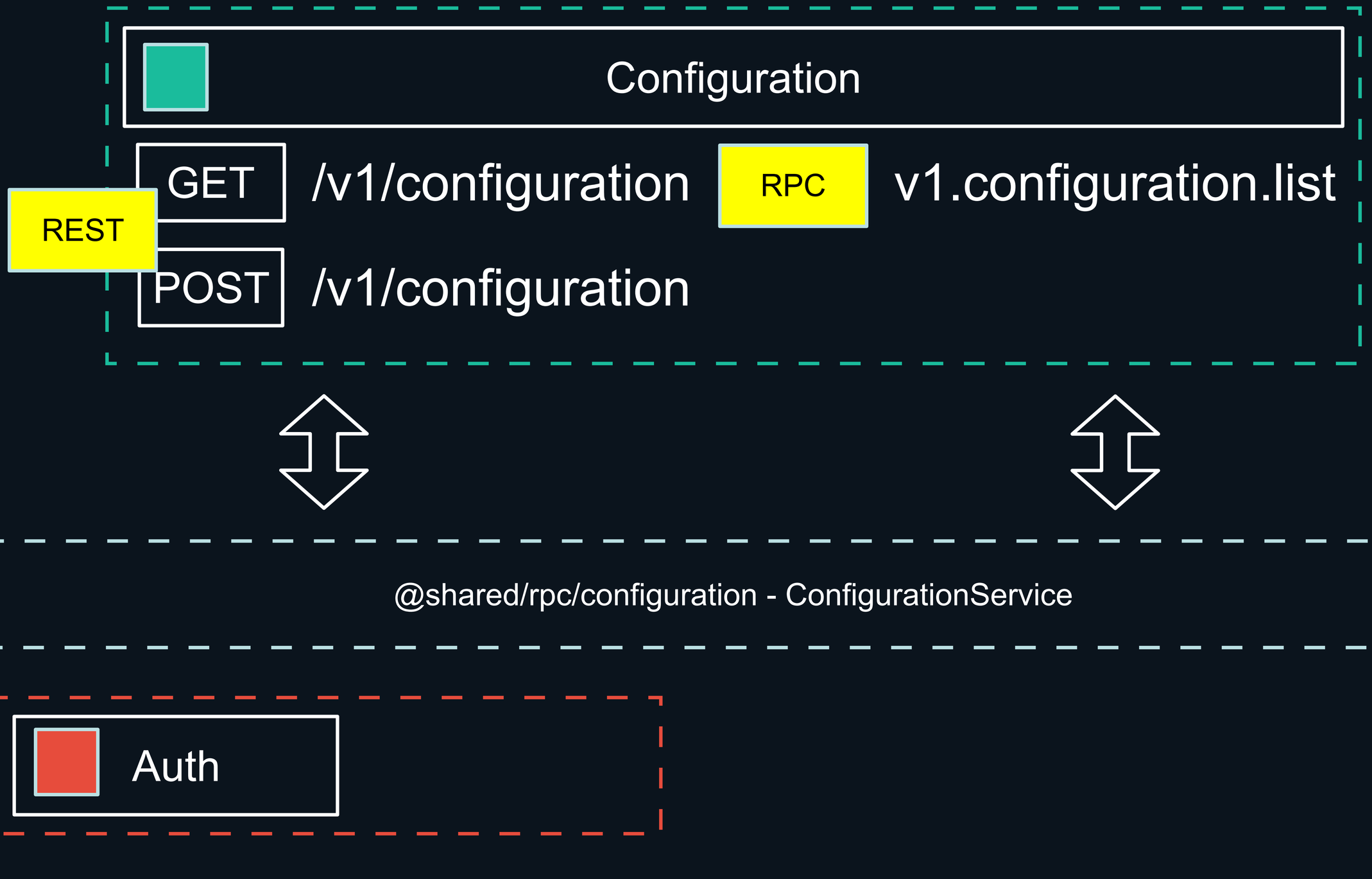
When we extract new microservice?

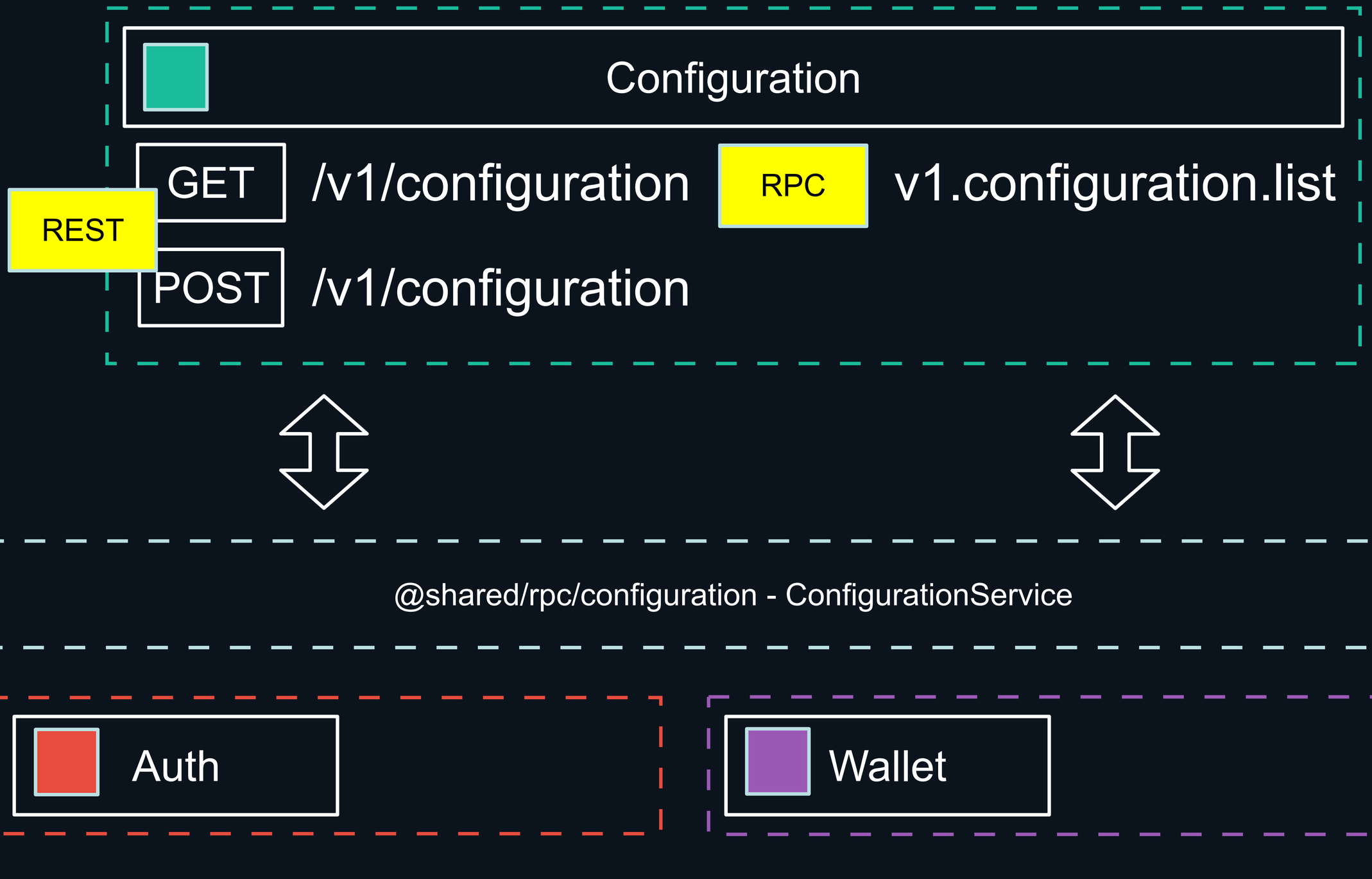
When we extract new microservice?


Configuration
GET
/v1/configuration
REST
POST
/v1/configuration


Configuration
GET
/v1/configuration
RPC
v1.configuration.list
REST
POST
/v1/configuration


Configuration
GET
/v1/configuration
RPC
v1.configuration.list
REST
POST
/v1/configuration
@shared/rpc/configuration - ConfigurationService


Configuration
GET
/v1/configuration
RPC
v1.configuration.list
REST
POST
/v1/configuration
@shared/rpc/configuration - ConfigurationService
Auth


Configuration
GET
/v1/configuration
POST
/v1/configuration
REST
RPC
v1.configuration.list
@shared/rpc/configuration - ConfigurationService
Auth
Wallet

Microservice architecture

Positive and negative sides


Service
DB

Distributed + Isolation

Distributed + Isolation

» Architecture

Distributed + Isolation

» Architecture

» Code

Distributed + Isolation

» Architecture
» Code
» Database

Architecture

Distributed

Distributed

» Overhead costs on communication

Distributed

» Overhead costs on communication
» CI/CD is important and should invest time

Distributed

- Overhead costs on communication
- CI/CD is important and should invest time
- Infra is more important (docker / orchestration / service discovery / secure store (vault) / tracing / monitoring / logging) and should invest time

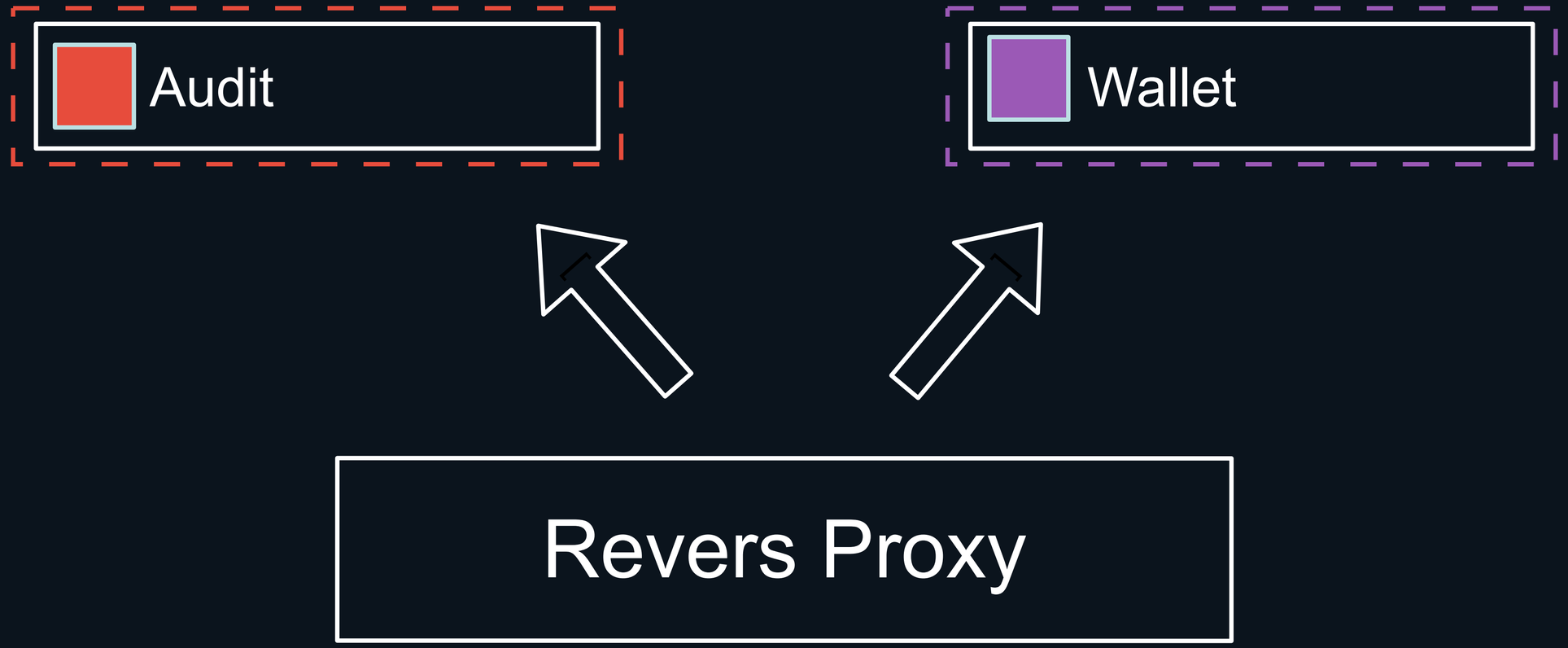
Distributed

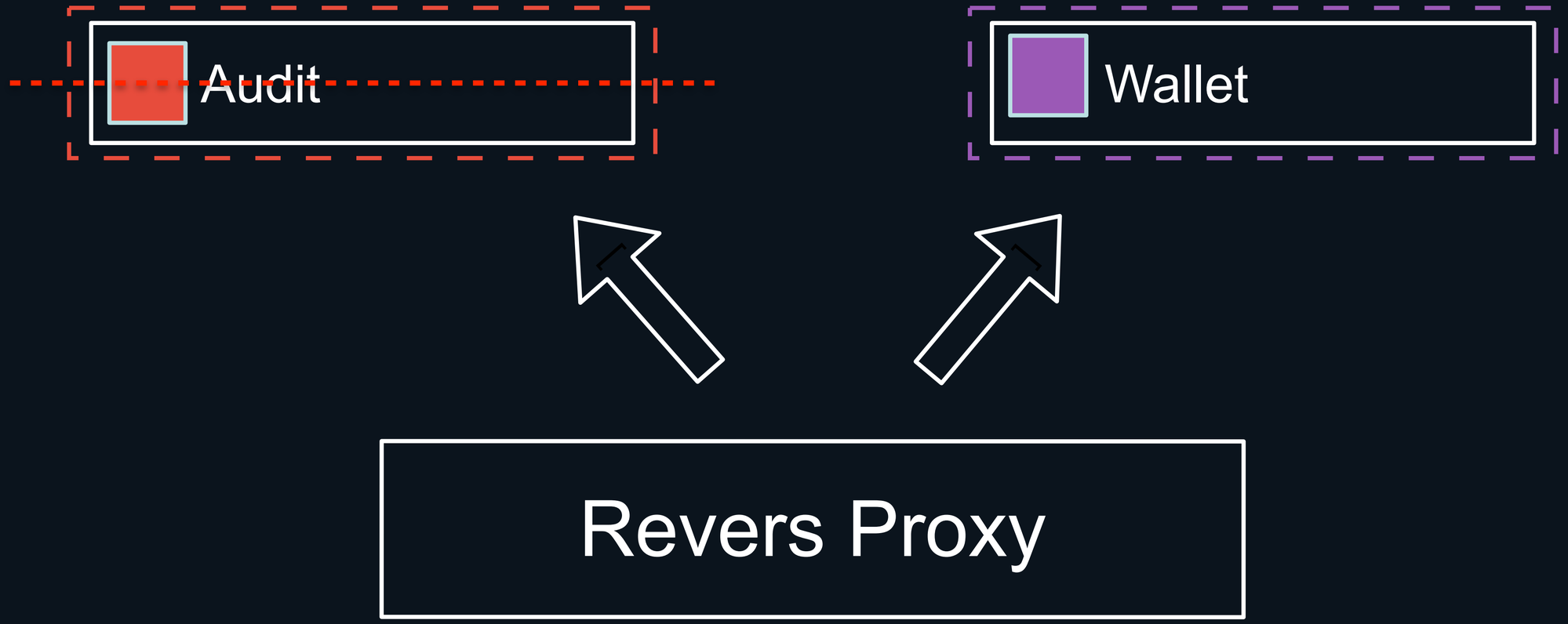
- Overhead costs on communication
- CI/CD is important and should invest time
- Infra is more important (docker / orchestration / service discovery / secure store (vault) / tracing / monitoring / logging) and should invest time
- Organization

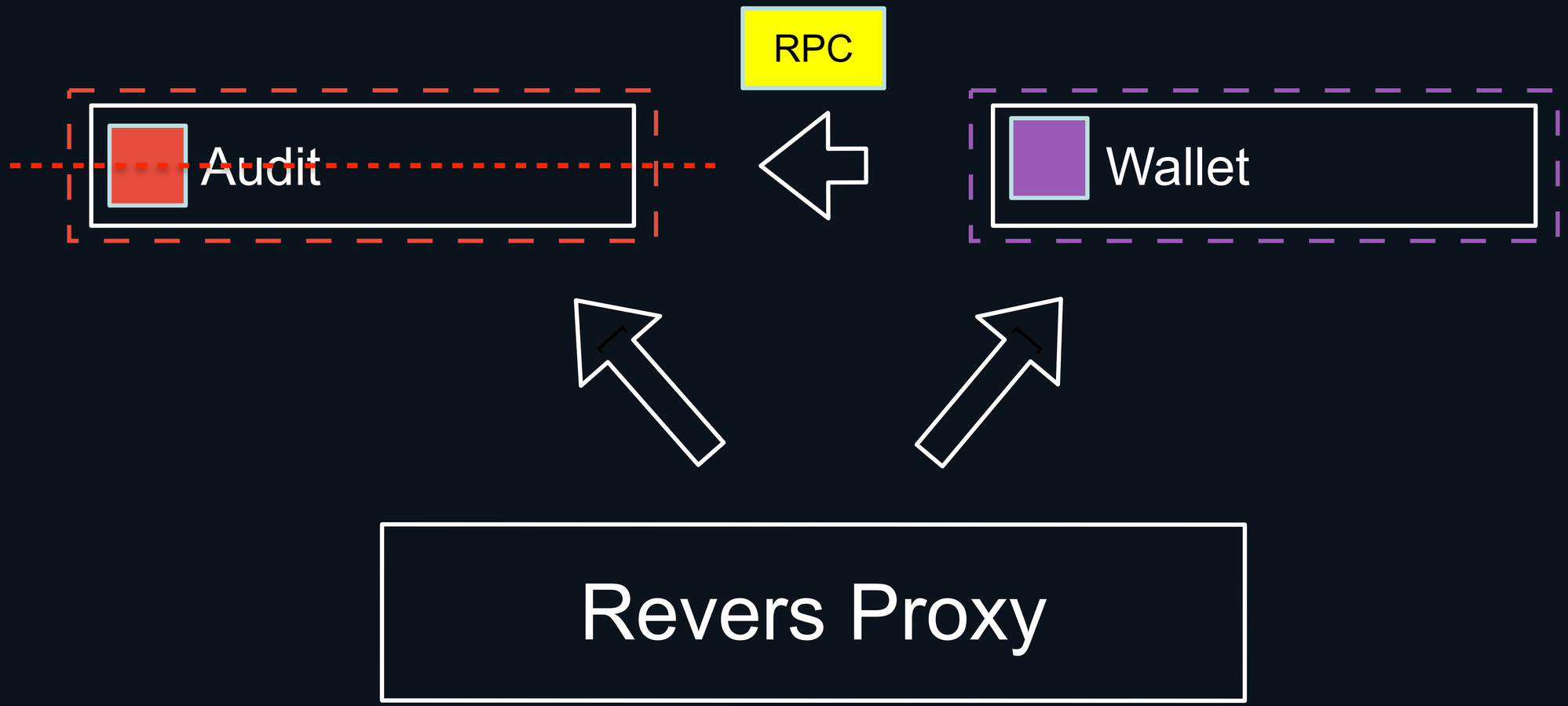
Fault tolerance

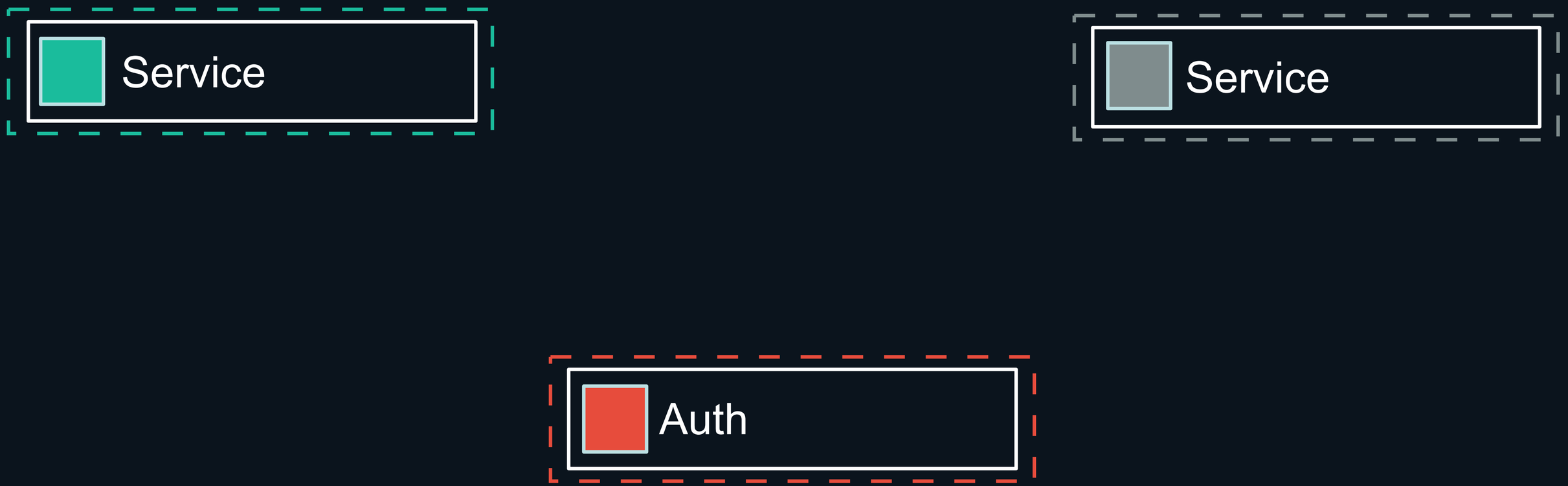
Fault tolerance

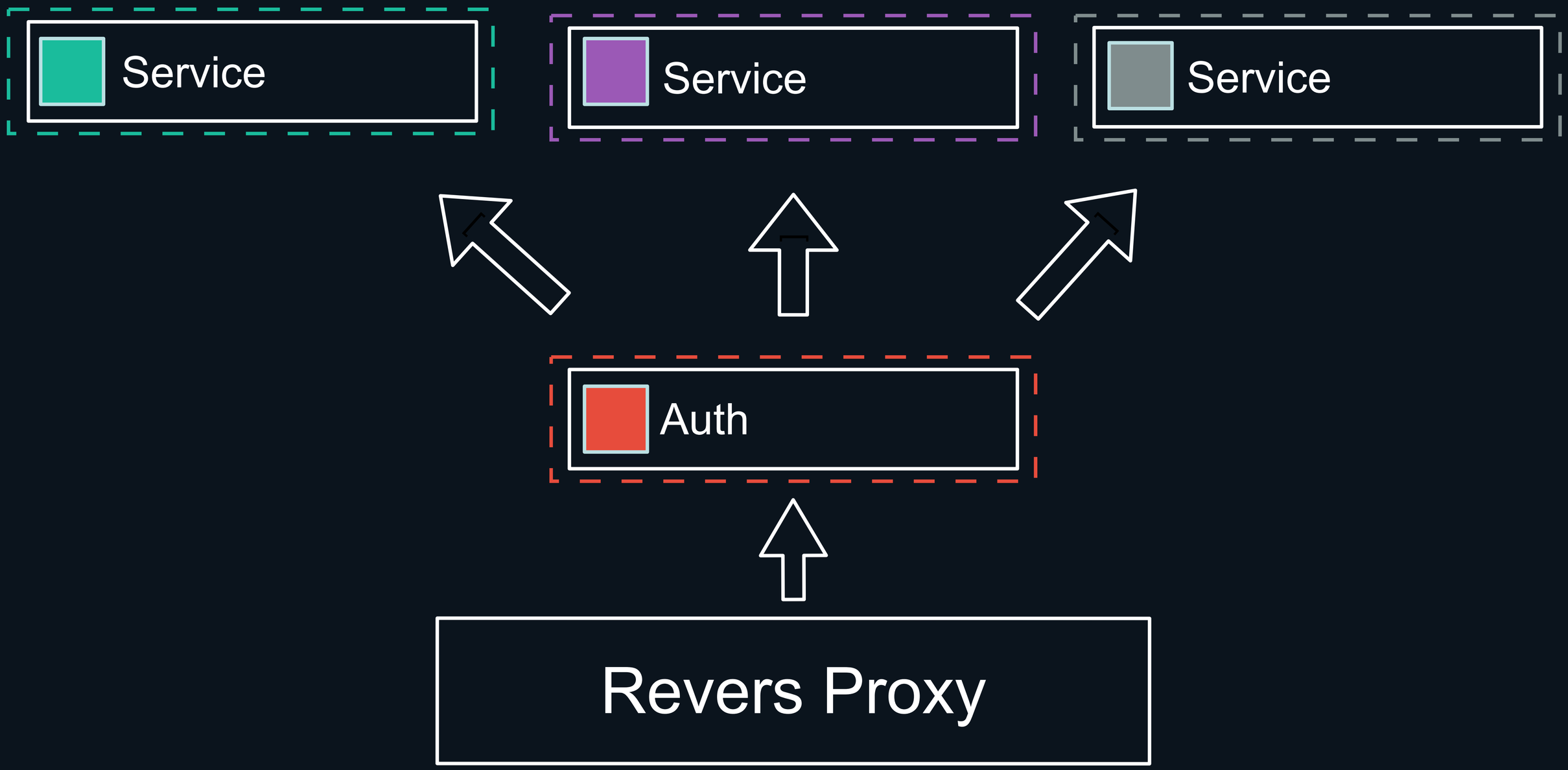
Downtime of one service will not down whole system

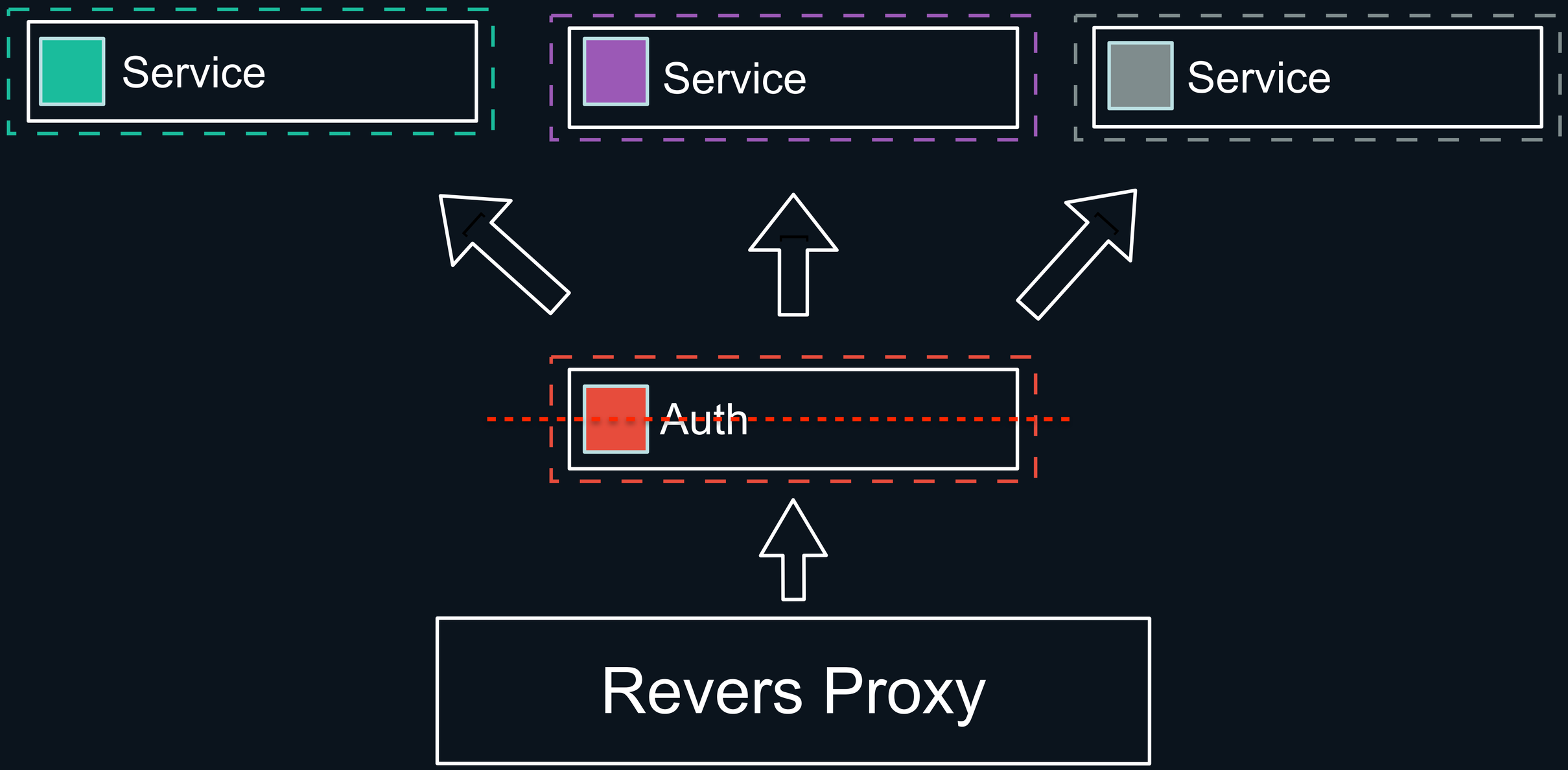

Audit
Wallet
Revers Proxy


Audit
Wallet
Revers Proxy


RPC
Audit
Wallet
Revers Proxy


Service
Service
Auth


Service
Service
Service
Auth
Revers Proxy


Service
Service
Service
Auth
Revers Proxy

TECH TRAIN 2018

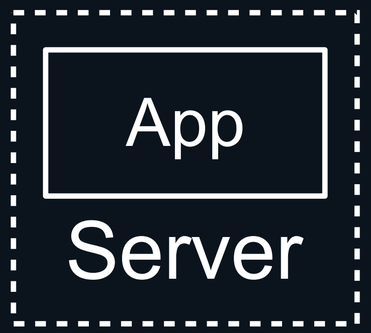
Олег Анастасьев

Одноклассники

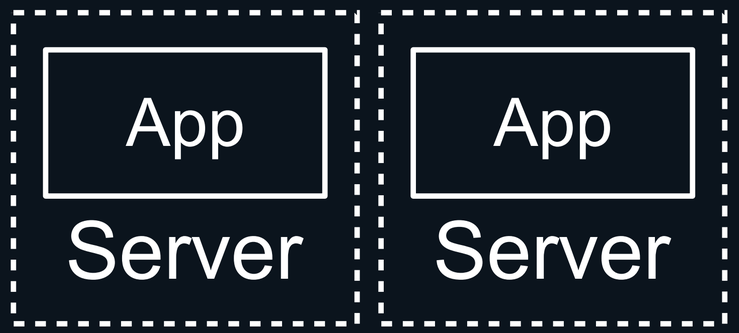
Надежность в распределенных системах

Scalability

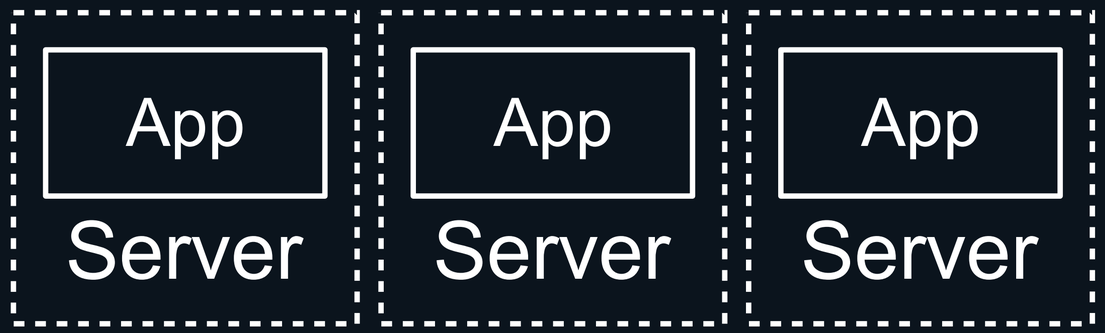
Before



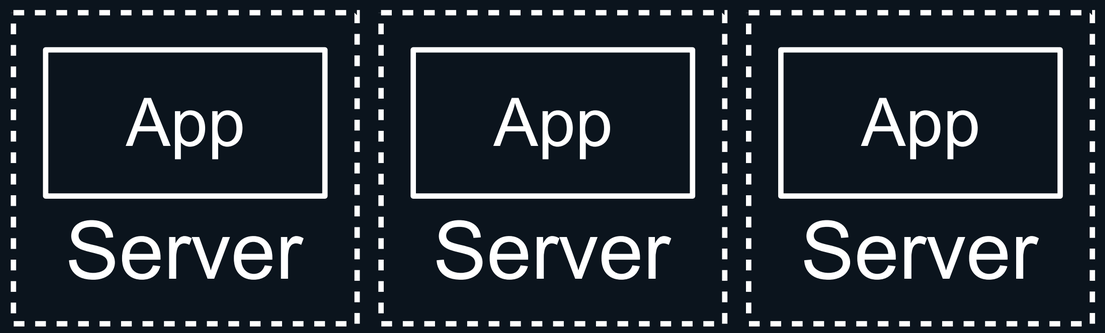
Before



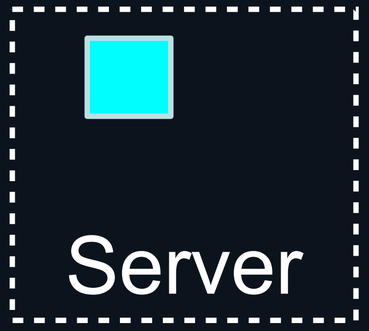
Before



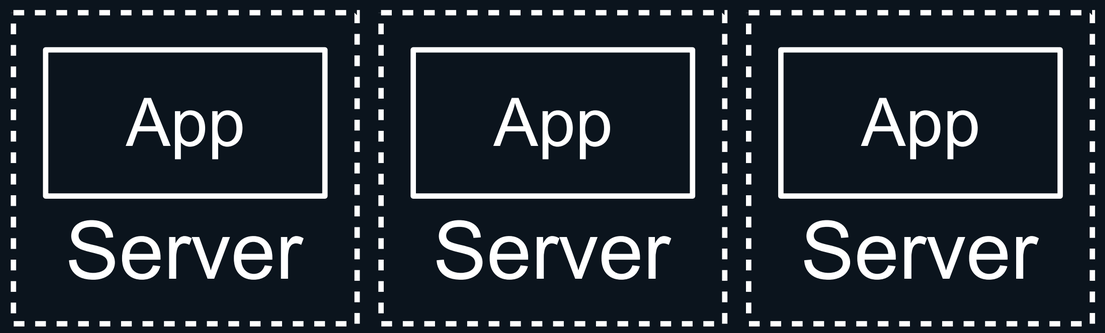
Before



After

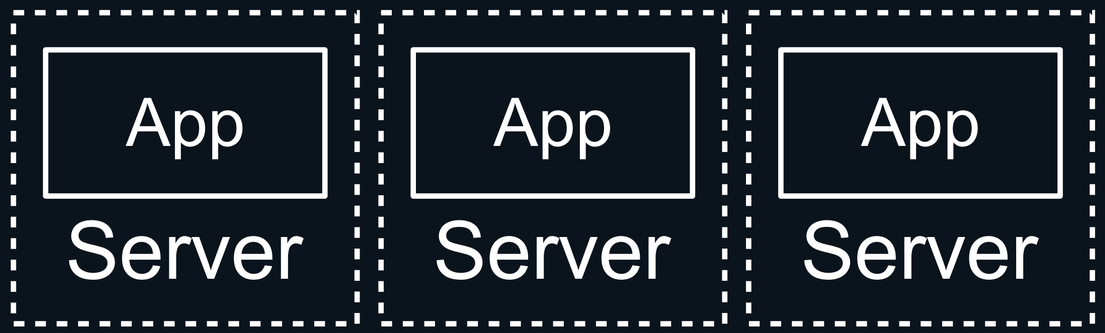


Before

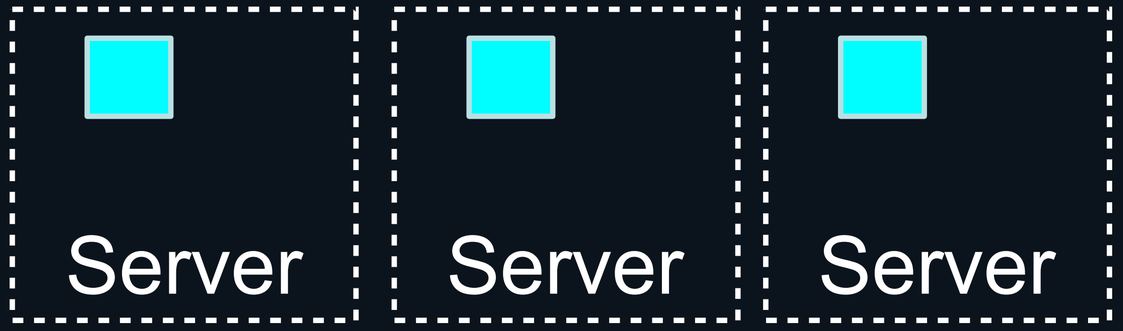


After

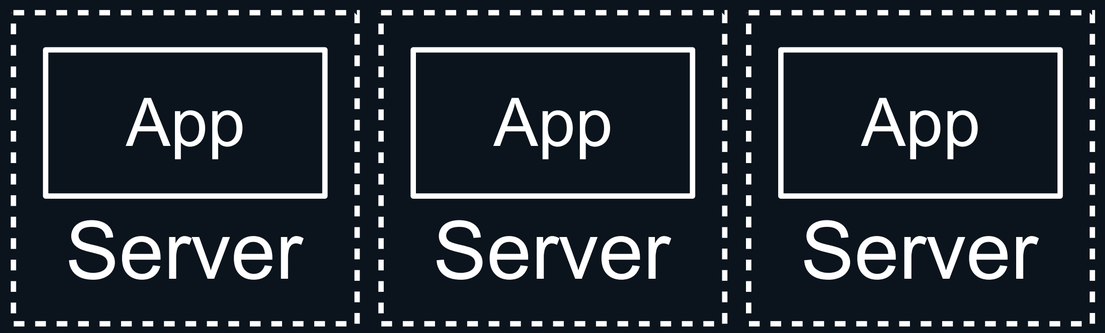
Before



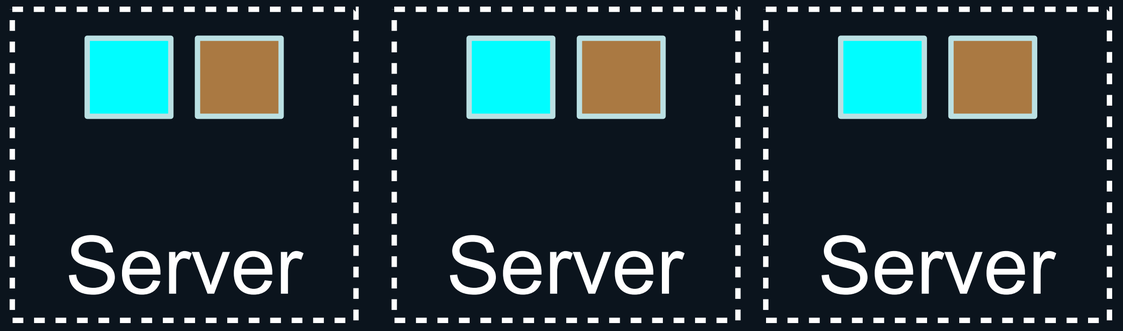
After



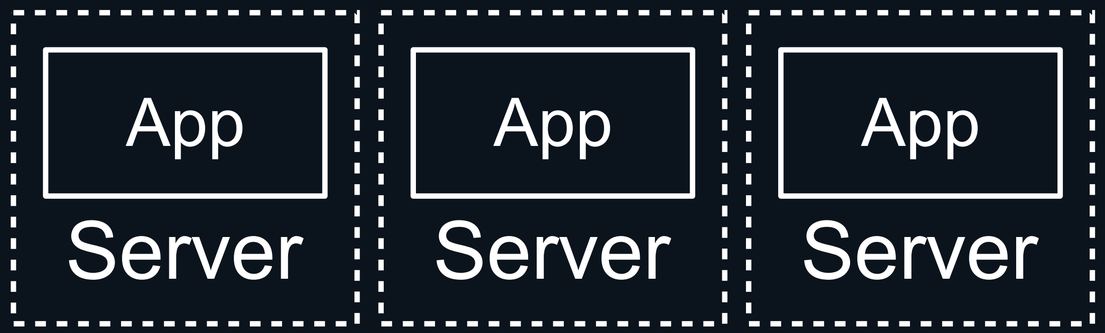
Before



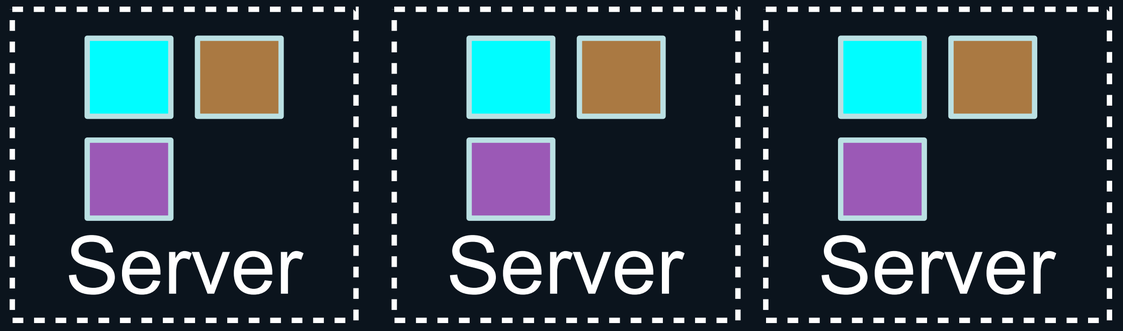
After



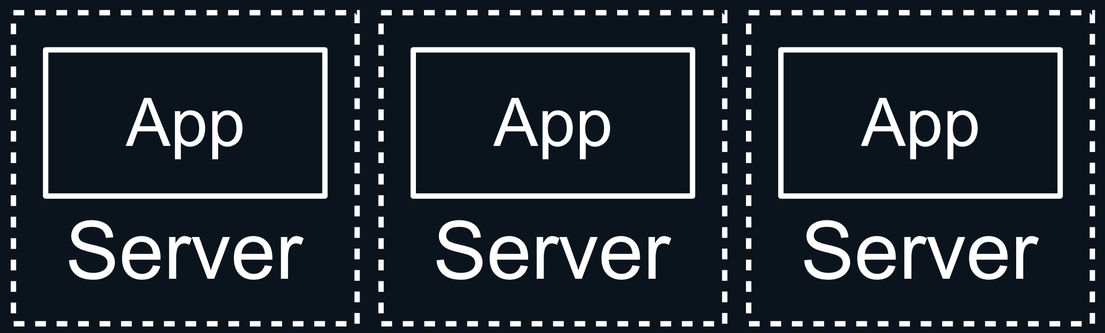
Before



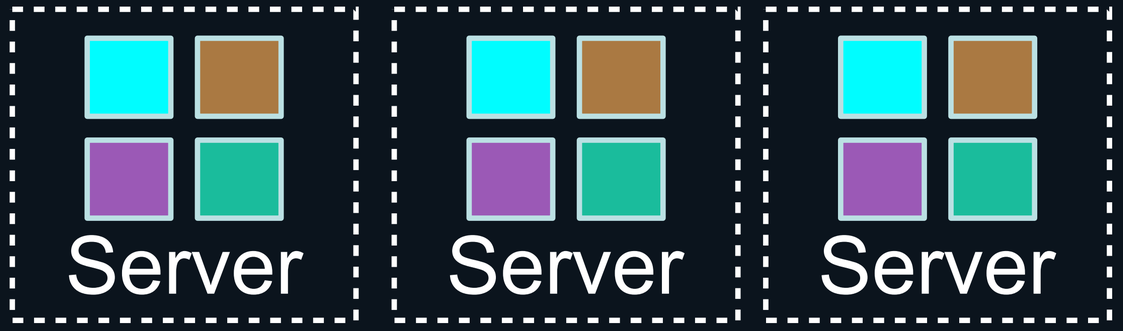
After



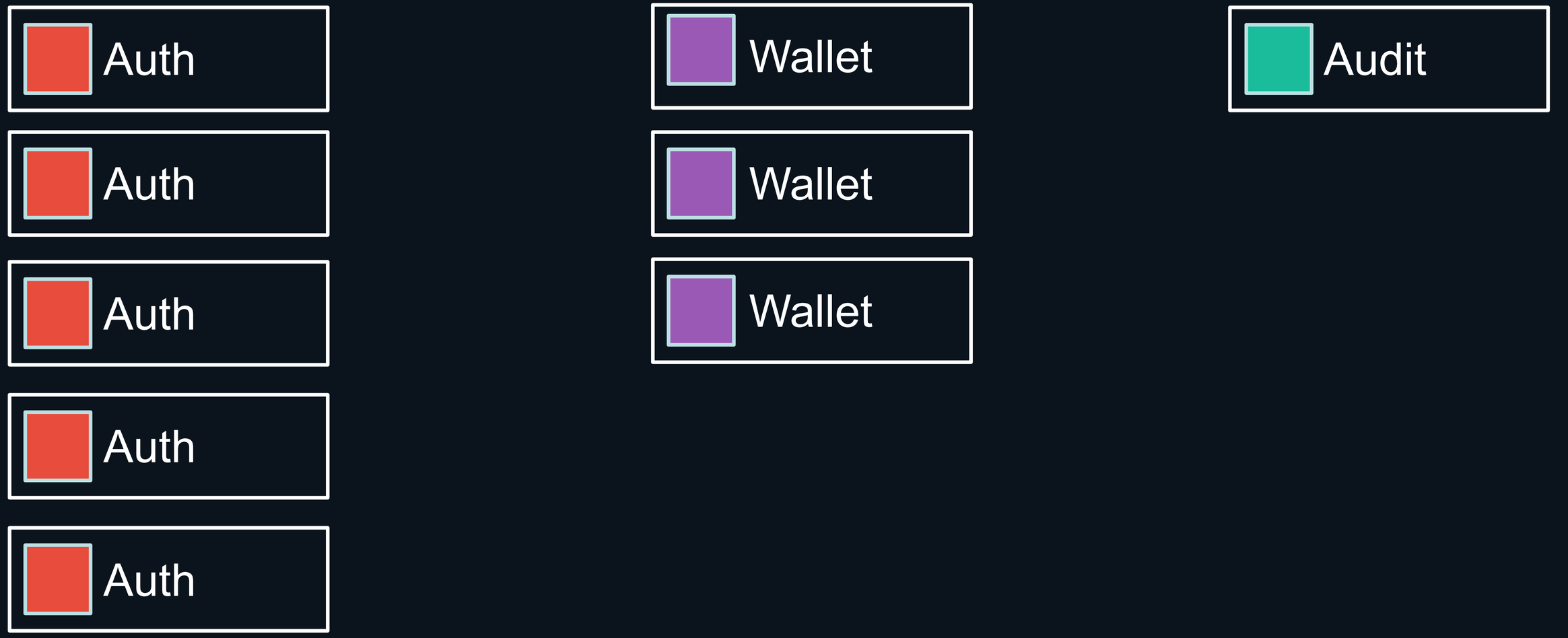
Before



After



Scalability



Developing

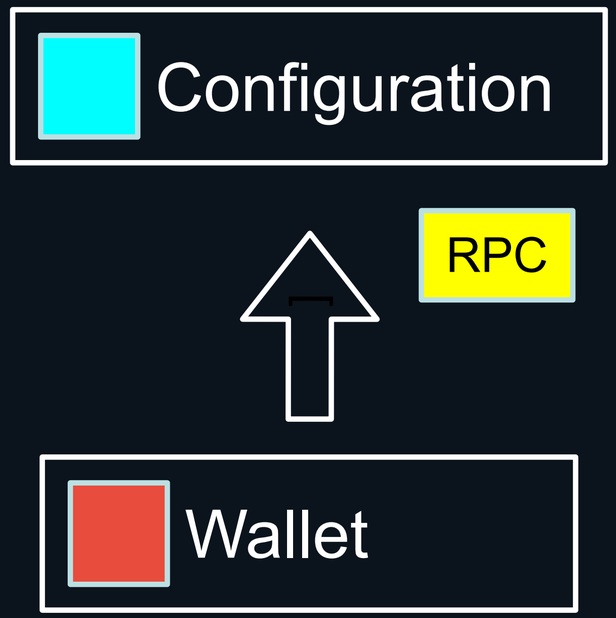
Developing

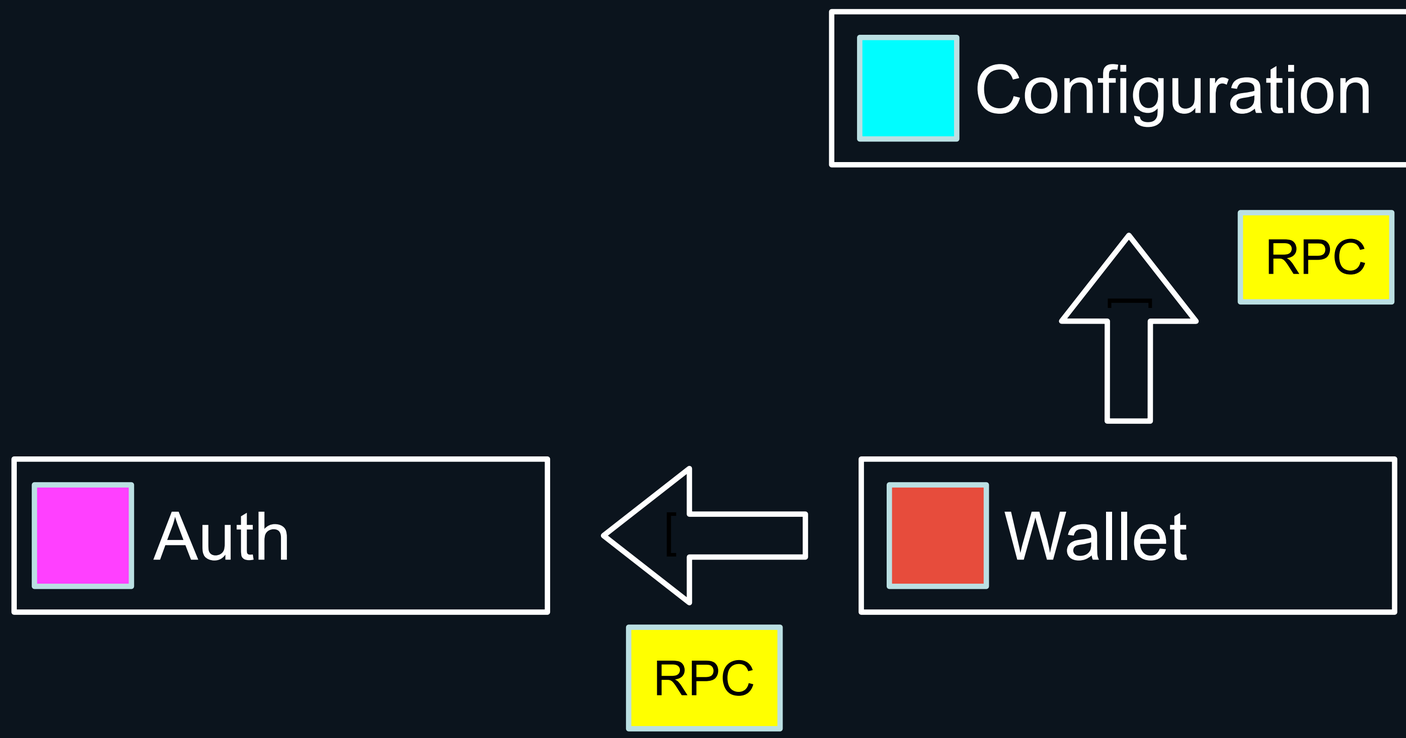
Developing

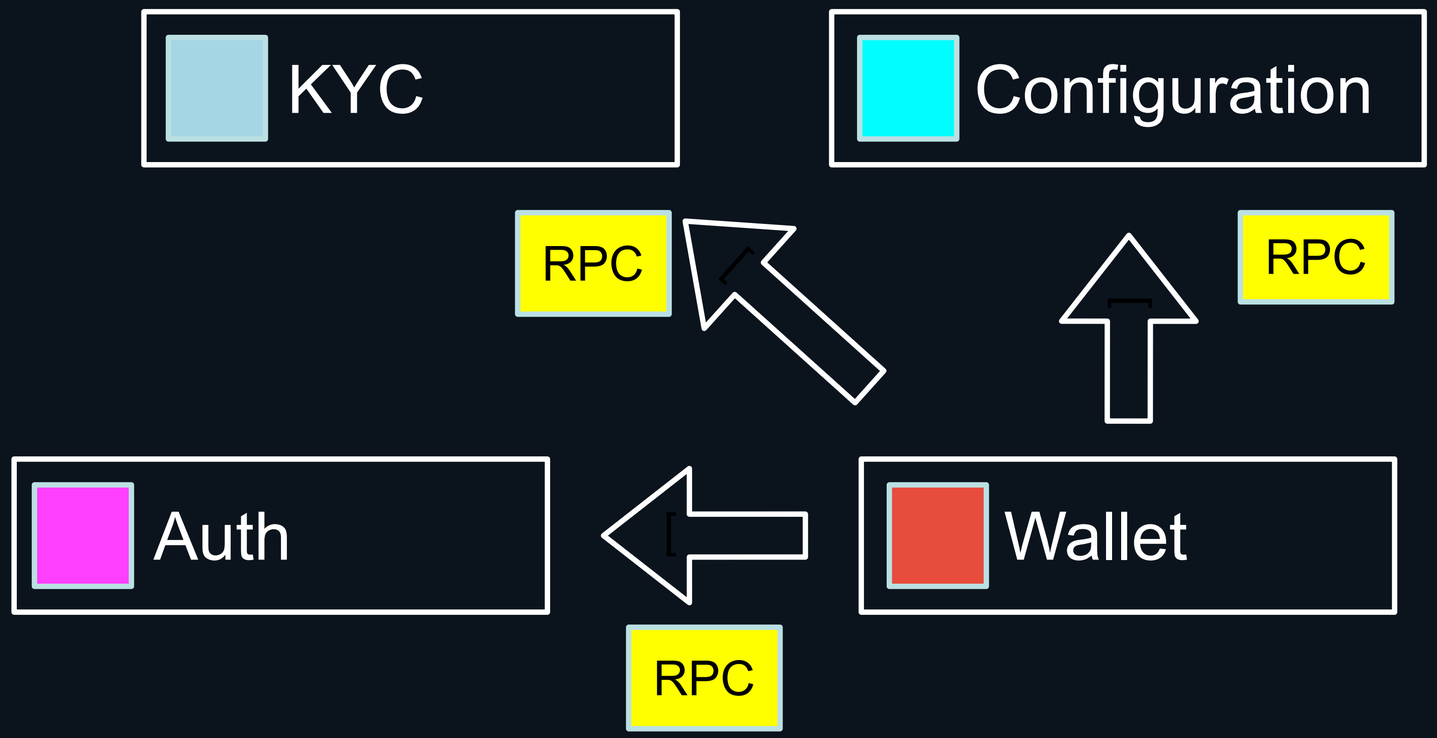
Developing

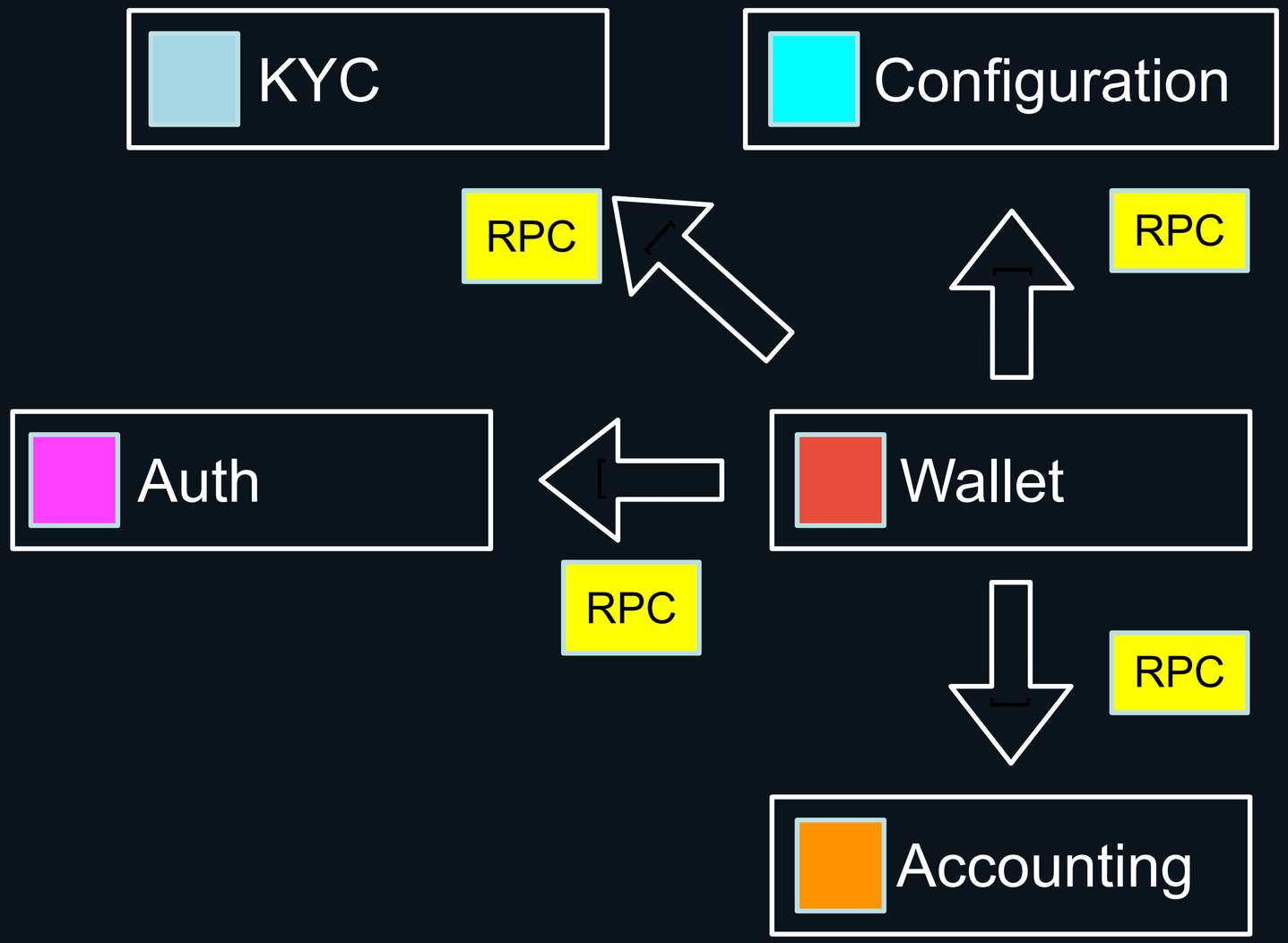
Developing

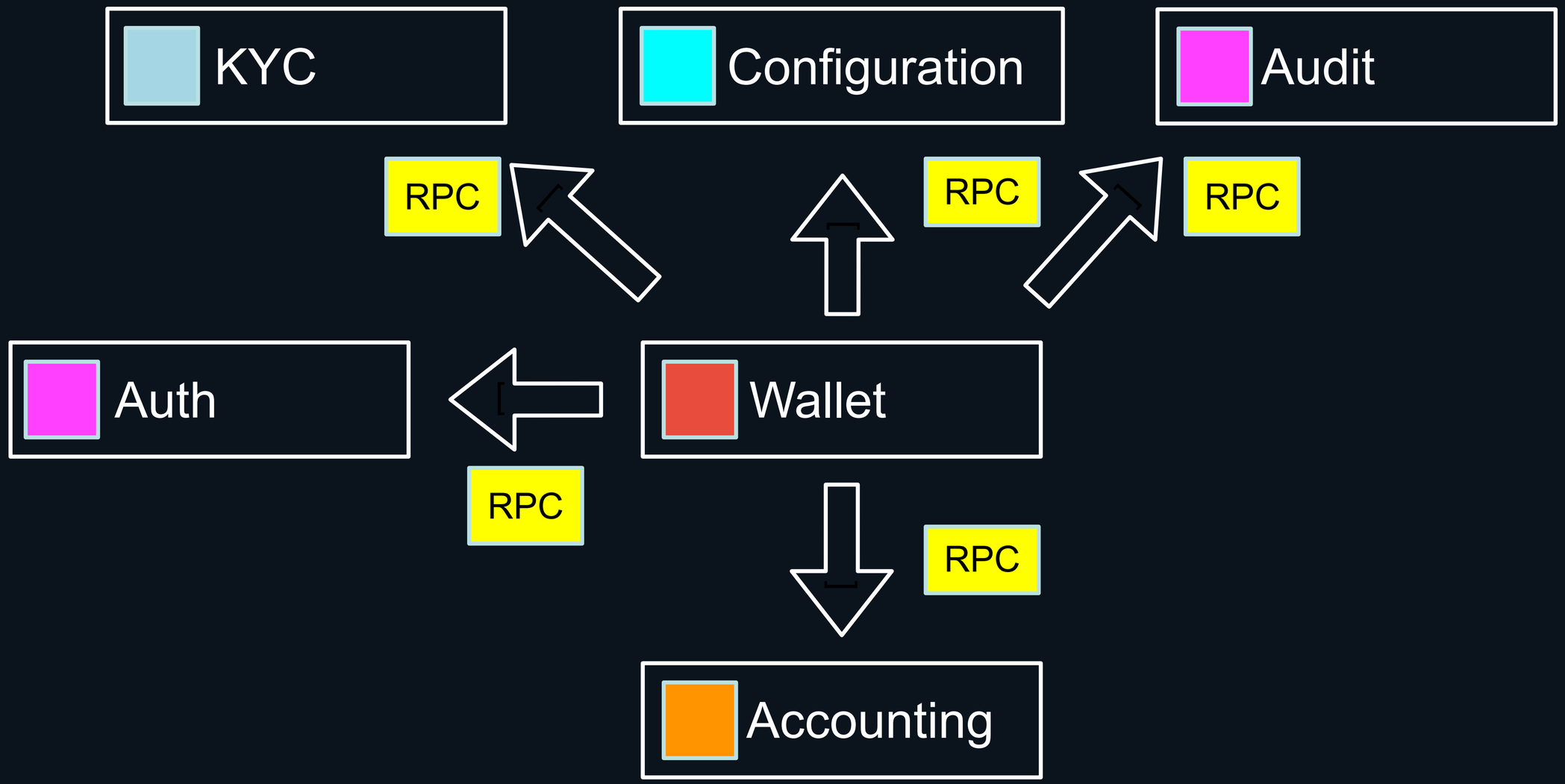

Wallet

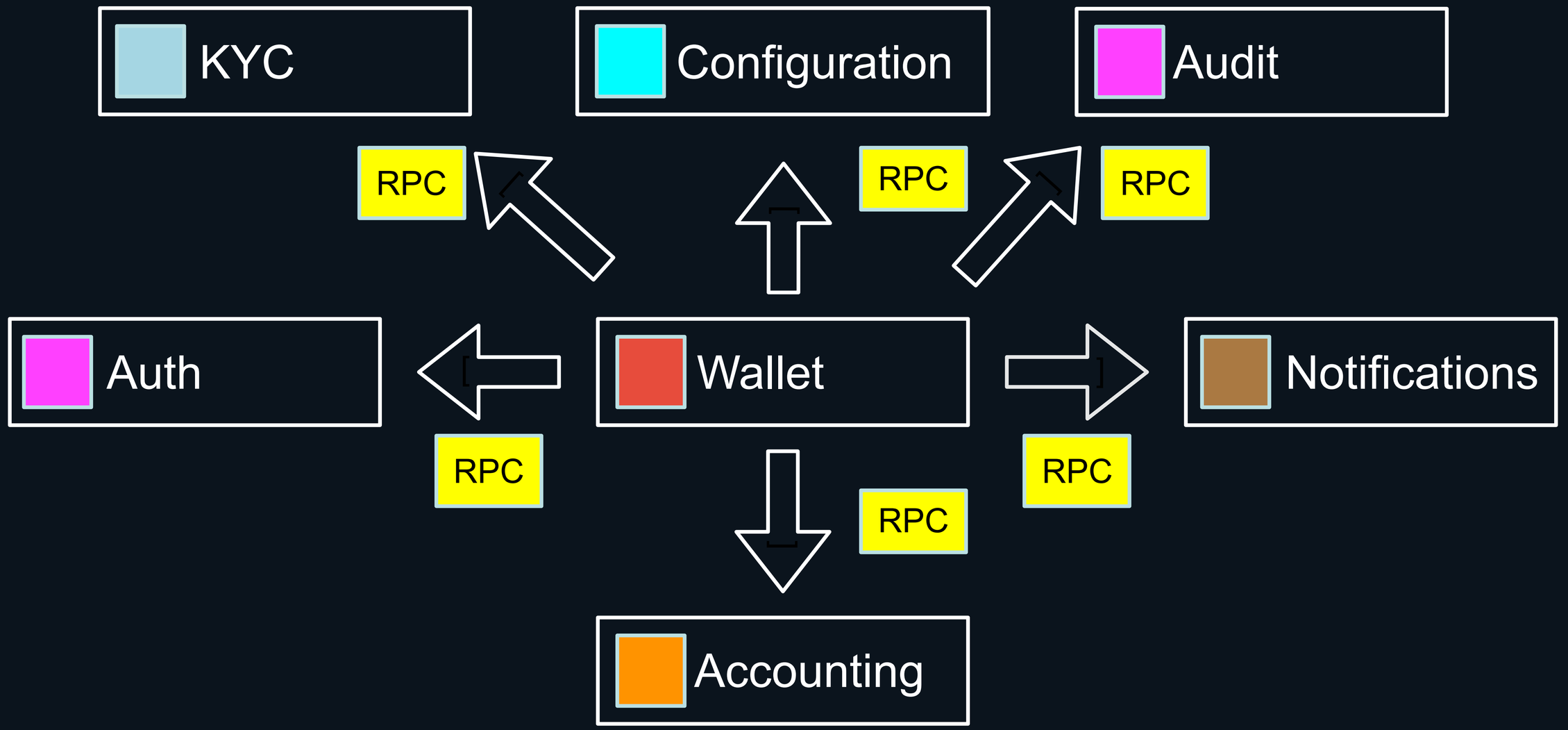

Configuration
RPC
Wallet


Configuration
RPC
Auth
Wallet
RPC


KYC
Configuration
RPC
RPC
Auth
Wallet
RPC


KYC
Configuration
RPC
RPC
Auth
Wallet
RPC
RPC
Accounting


KYC
Configuration
Audit
RPC
RPC
RPC
Auth
Wallet
RPC
RPC
Accounting


KYC
Configuration
Audit
RPC
RPC
RPC
Auth
Wallet
Notifications
RPC
RPC
RPC
Accounting

Problems

Problems

» You need a lot of microservices to execute

Problems

» You need a lot of microservices to execute

» You need working infrastructure on local machine

Problems

» You need a lot of microservices to execute
» You need working infrastructure on local machine
» You need to update services + configuration


Yeah, big bada boom.


Yeah, big bada boom.

How to test?

Developer (Manual)

Transfer Person A -> Person B

Transfer Person A -> Person B

» Create User A

Transfer Person A -> Person B

» Create User A

» Create User B

Transfer Person A -> Person B

» Create User A
» Create User B
» Create User Company

Transfer Person A -> Person B

» Create User A
» Create User B
» Create User Company
» Create Agent

Transfer Person A -> Person B

» Create User A
» Create User B
» Create User Company
» Create Agent
» Emission money to System - Master Account

Transfer Person A -> Person B

» Create User A
» Create User B
» Create User Company
» Create Agent
» Emission money to System - Master Account
» Populate money to company - Repository Account

Transfer Person A -> Person B

» Create User A
» Create User B
» Create User Company
» Create Agent
» Emission money to System - Master Account
» Populate money to company - Repository Account
» CASH_IN Money to UserA

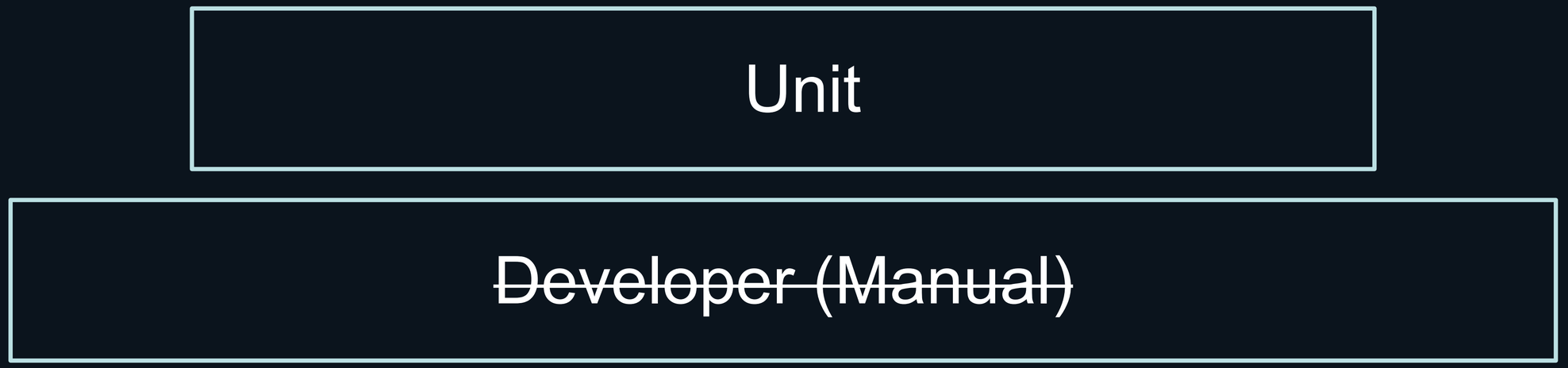
Transfer Person A -> Person B

- Create User A
- Create User B
- Create User Company
- Create Agent
- Emission money to System - Master Account
- Populate money to company - Repository Account
- CASH_IN Money to UserA
- Transfer User A -> User B

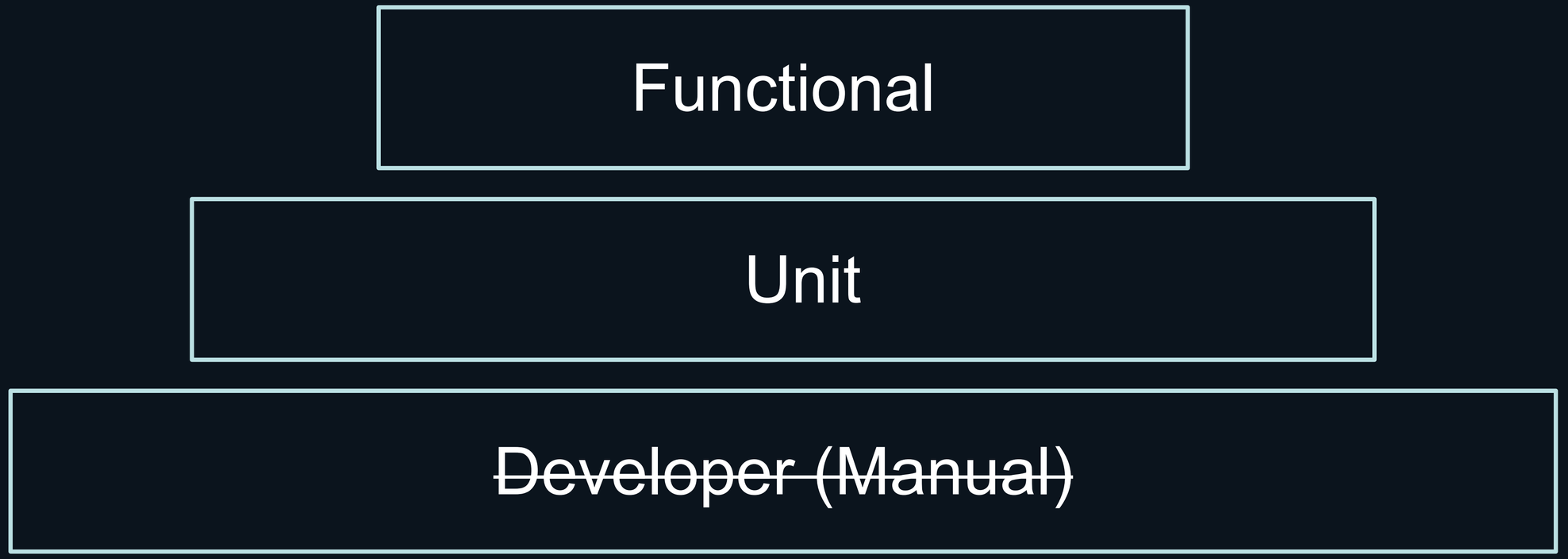
How to test?

~~Developer (Manual)~~

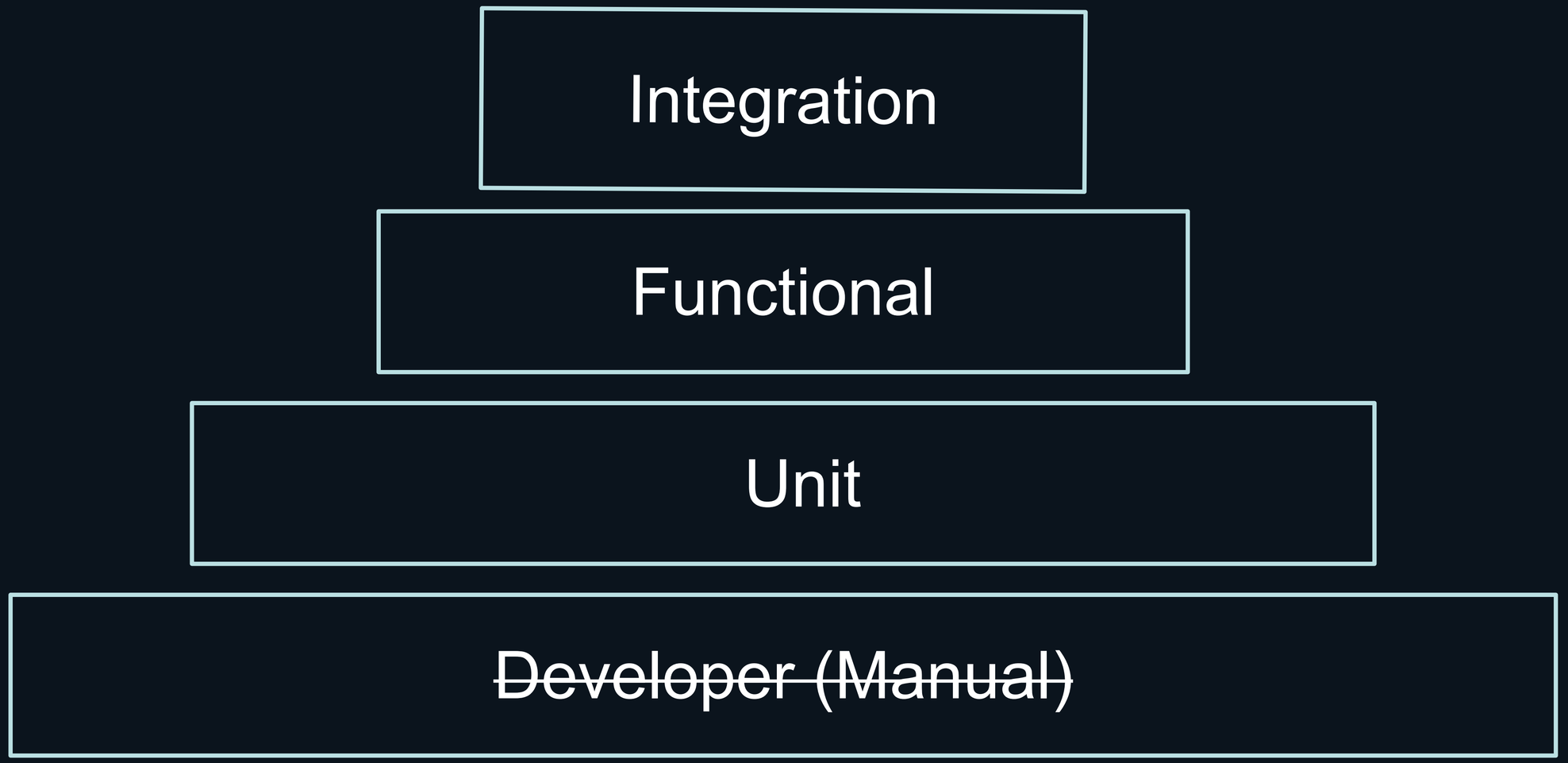
How to test?



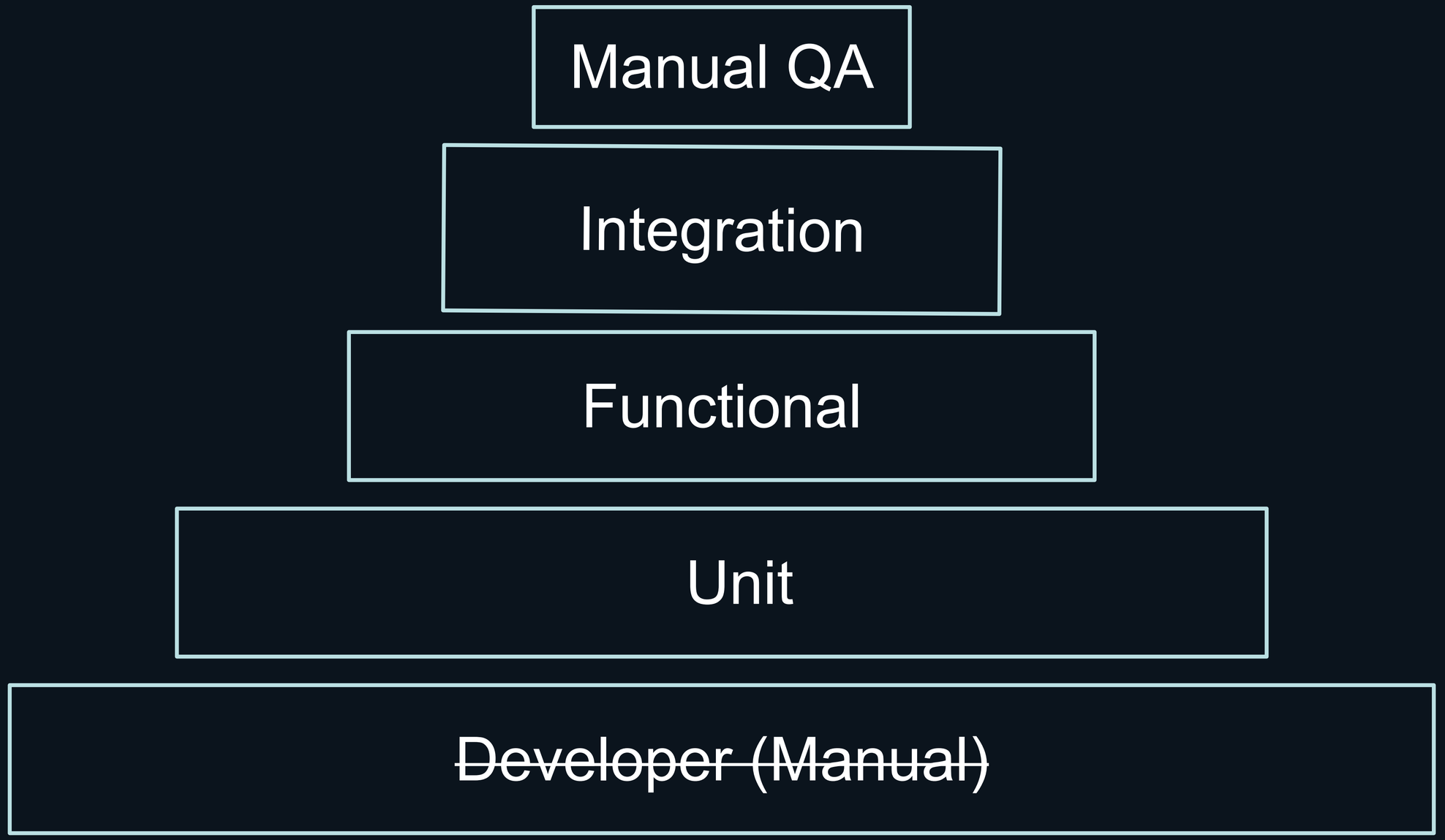
How to test?



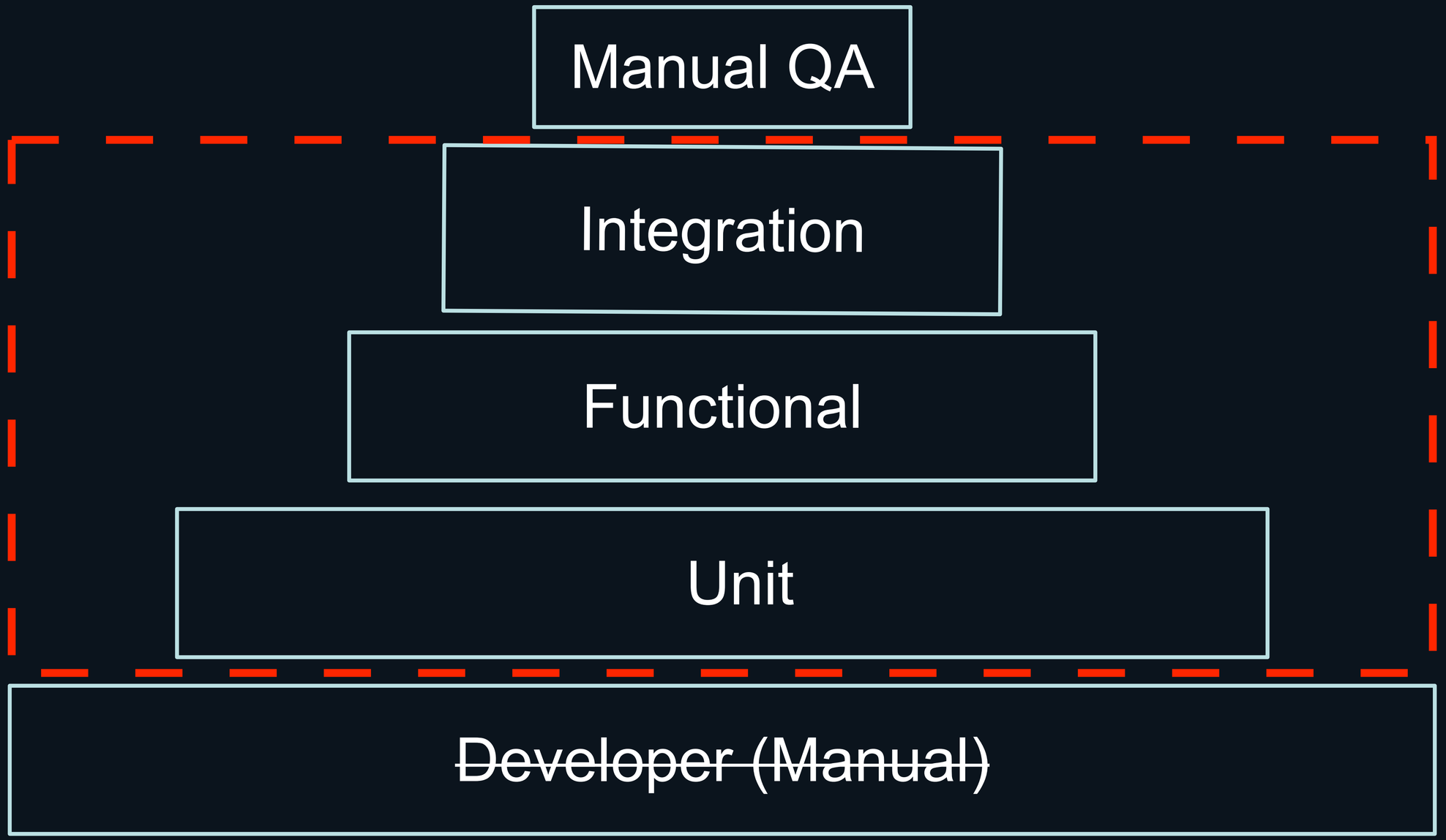
How to test?



How to test?



How to test?

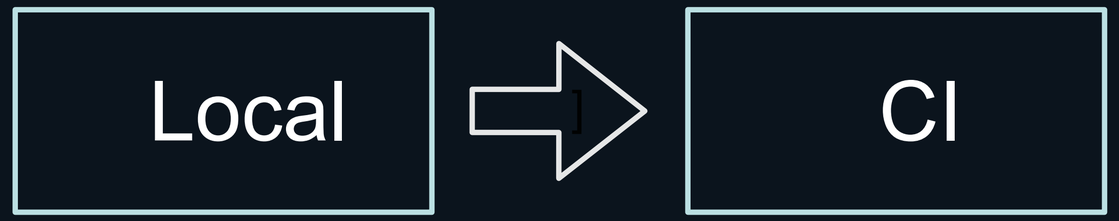


Testing cycle

Testing cycle

Unit

Testing cycle



Unit

Unit
Functional
Integration
CLI Testing

Testing cycle

latest

Unit

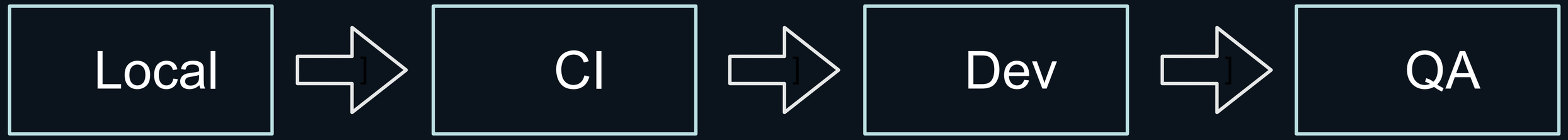
Unit
Functional
Integration
CLI Testing

Developer
(manual)

Testing cycle

latest

x.x.x



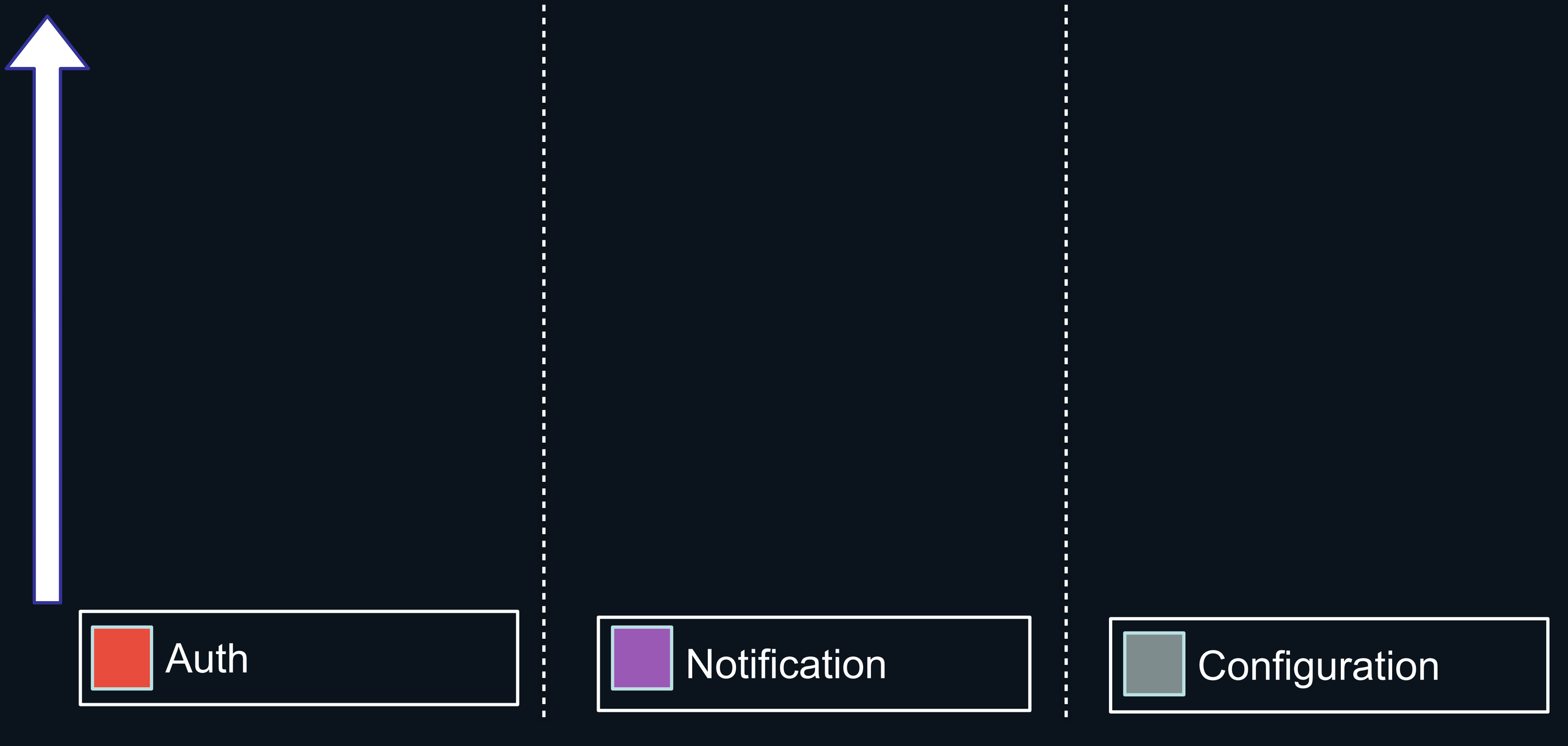
Unit

Unit
Functional
Integration
CLI Testing

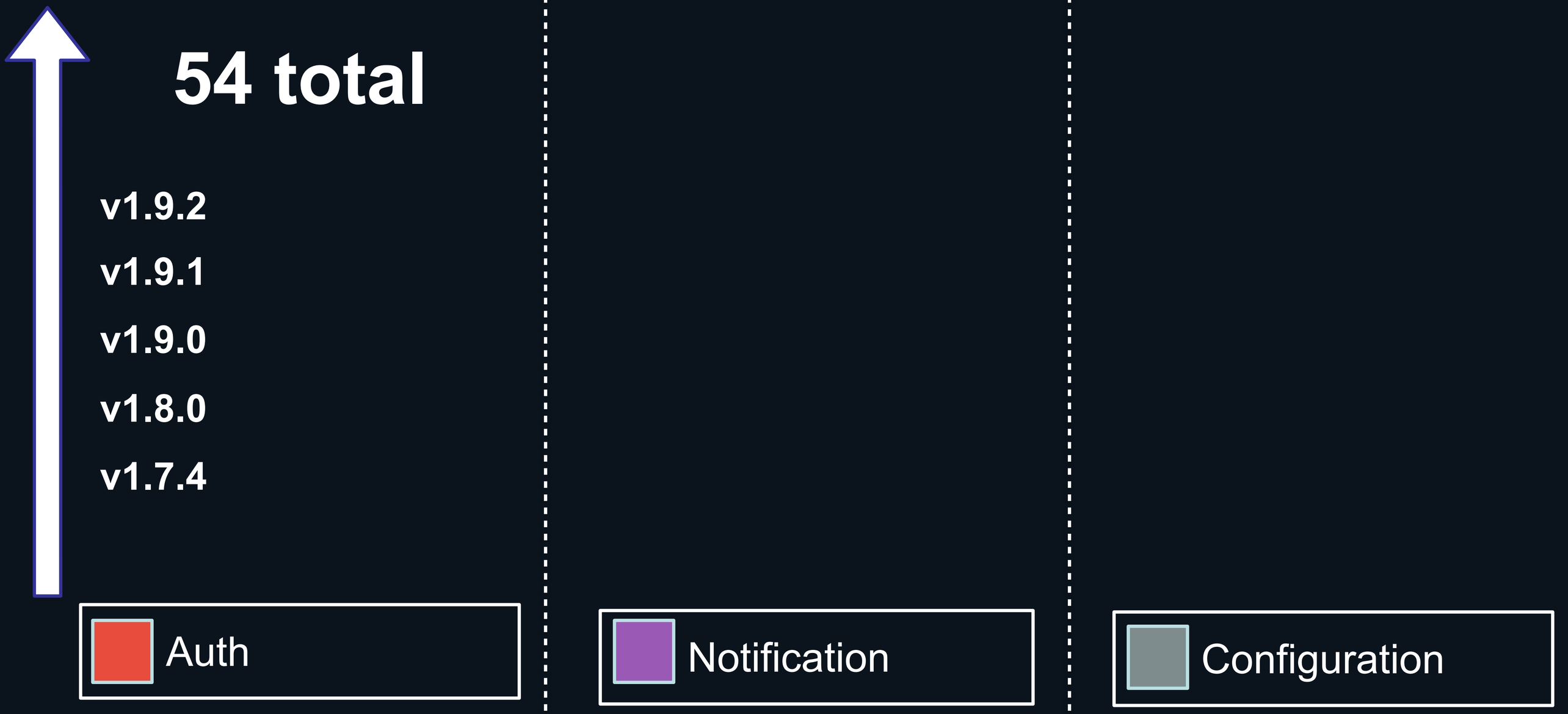
Developer
(manual)

QA Manual
(manual)

Independent development/deploy



Independent development/deploy



Independent development/deploy

54 total

v1.9.2

v1.9.1

v1.9.0

v1.8.0

v1.7.4

12 total

v1.1.0

v1.0.3

v1.0.2 v2.0.0

v1.0.1 v1.2.1

v1.0.0 v1.2.0

Auth

Notification

Configuration

Independent development/deploy

54 total

v1.9.2

v1.9.1

v1.9.0

v1.8.0

v1.7.4

Auth

12 total

v1.1.0

v1.0.3

v1.0.2 v2.0.0

v1.0.1 v1.2.1

v1.0.0 v1.2.0

Notification

18 total

v1.5.0

v1.4.0

v1.2.0

v1.1.0

v1.0.1

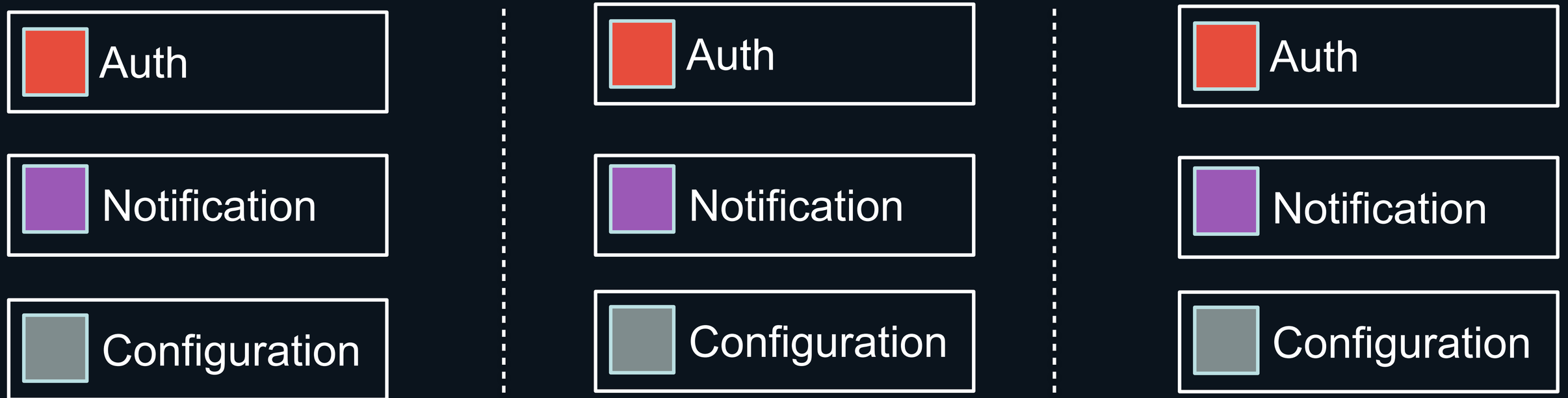
v1.0.0

Configuration

Reusage

Project A | Project B | Project C

Reusage

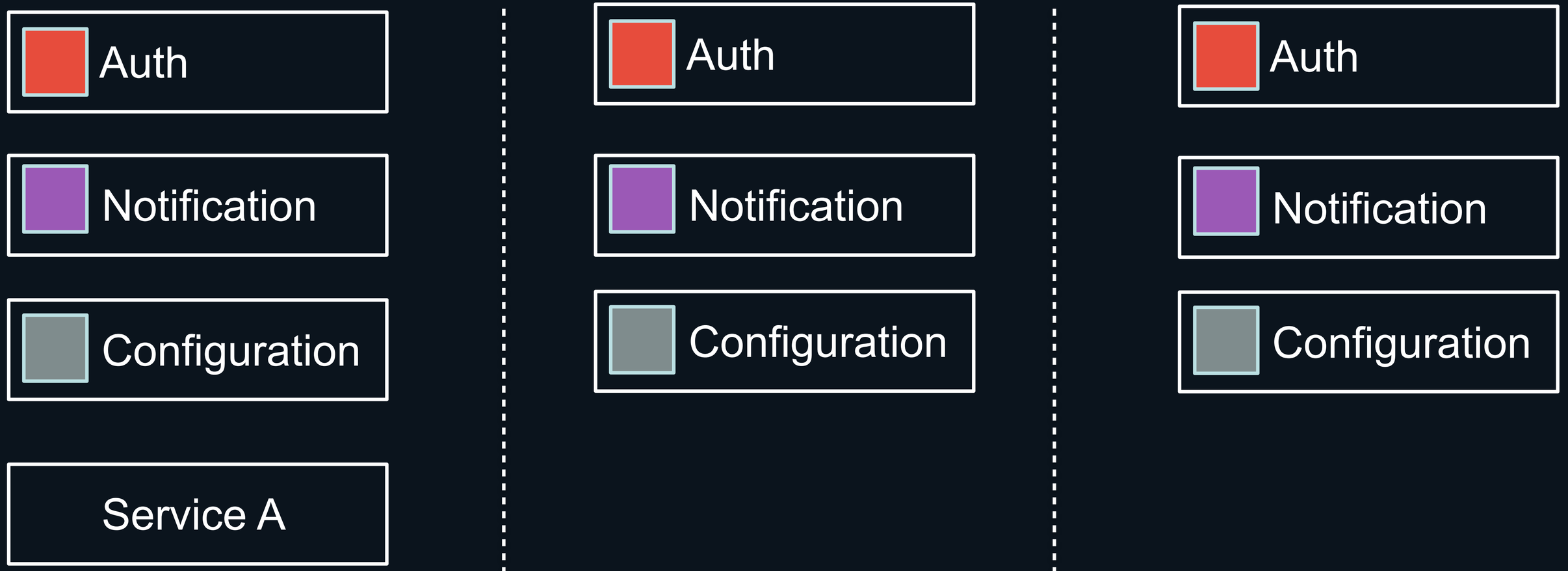


Project A

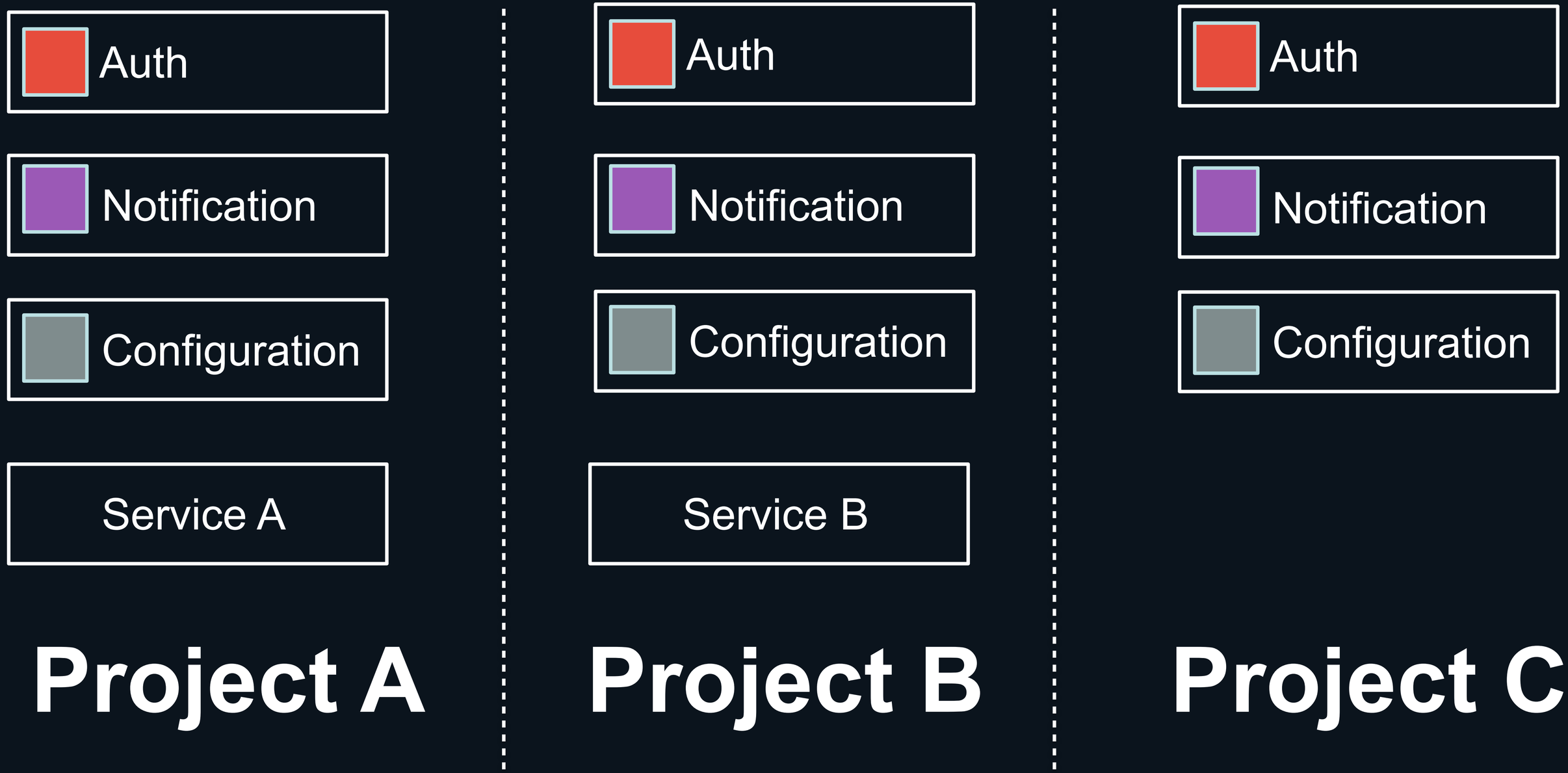
Project B

Project C

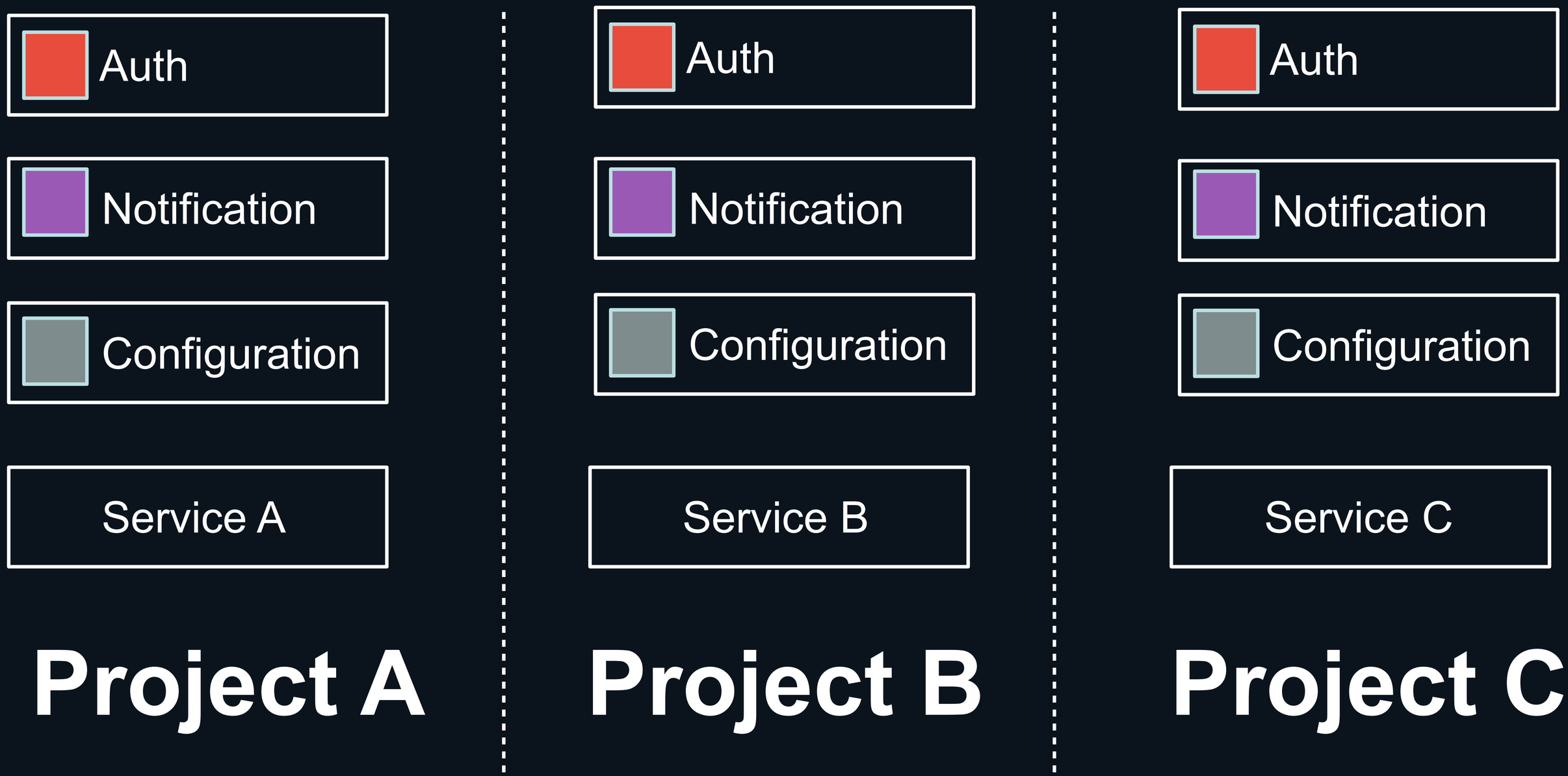
Reusage


Auth
Notification
Configuration
Service A
Auth
Notification
Configuration
Auth
Notification
Configuration
Project A
Project B
Project C

Reusage


Auth
Notification
Configuration
Service A
Auth
Notification
Configuration
Service B
Auth
Notification
Configuration
Project A
Project B
Project C

Reusage

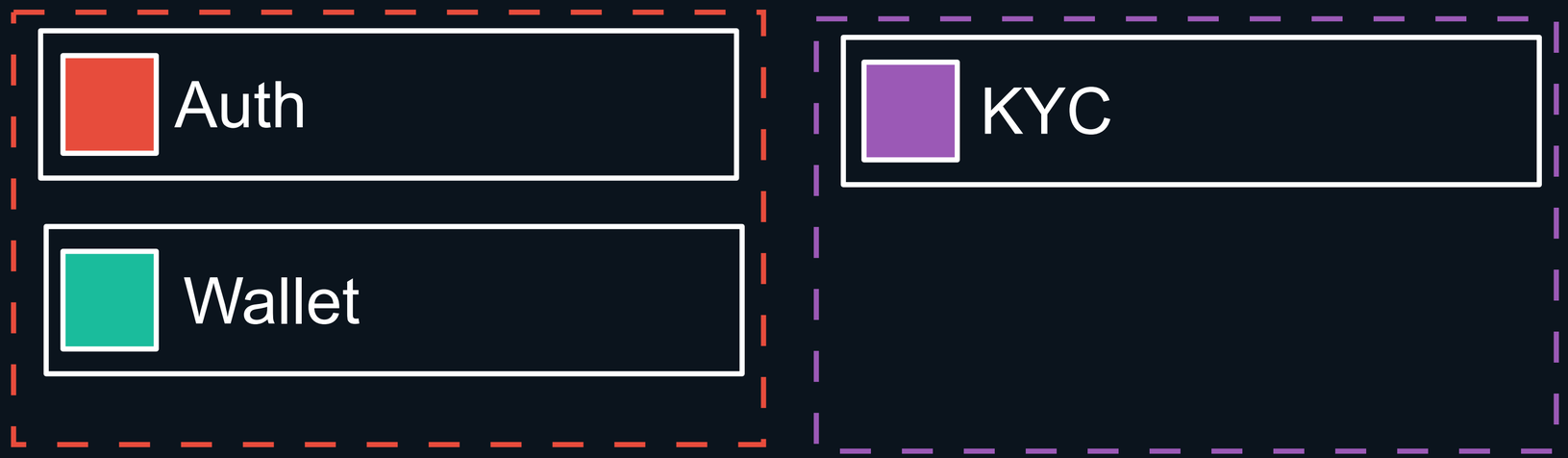

Auth
Notification
Configuration
Service A
Auth
Notification
Configuration
Service B
Auth
Notification
Configuration
Service C
Project A
Project B
Project C

Code

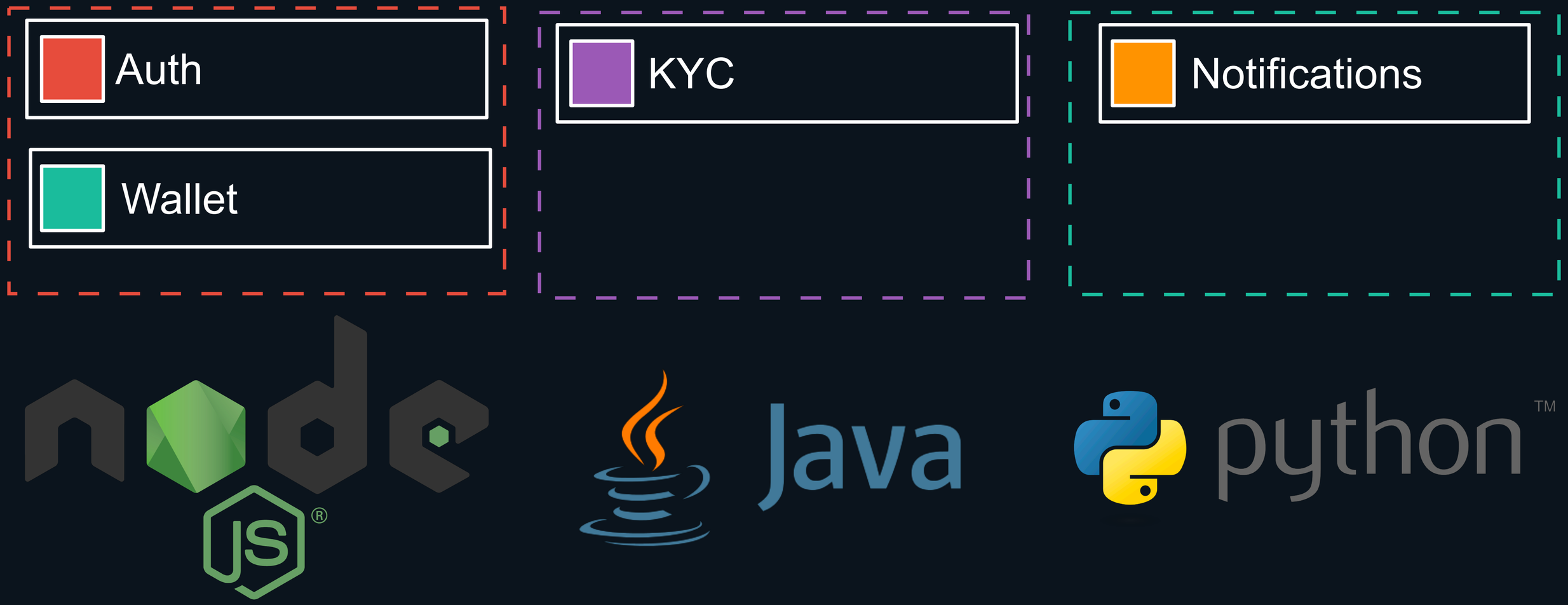
Technological heterogeneity

Technological heterogeneity

Technological heterogeneity



Technological heterogeneity

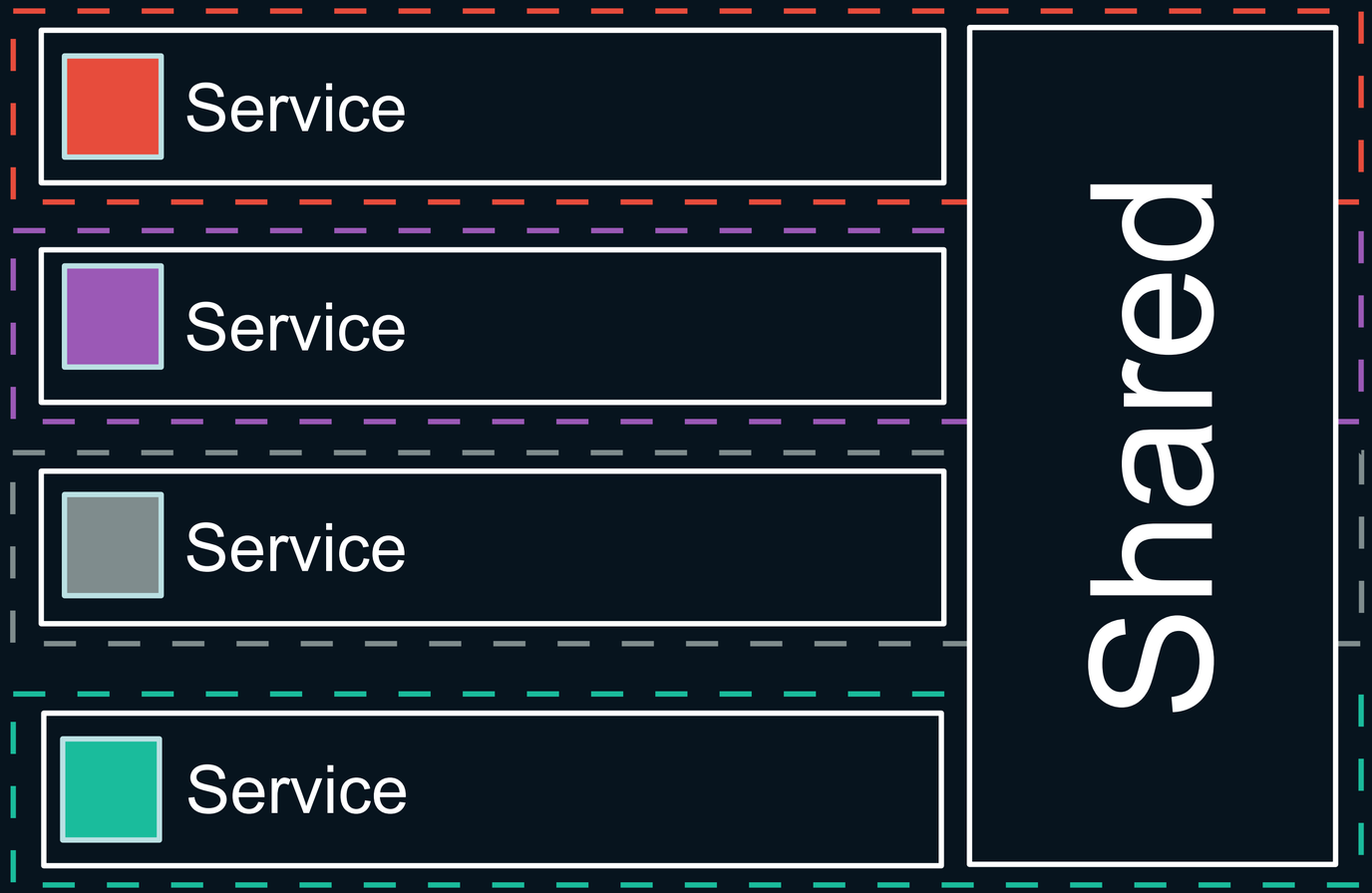


Maintainability

» 100500 repositories

Dependency control

One shared package

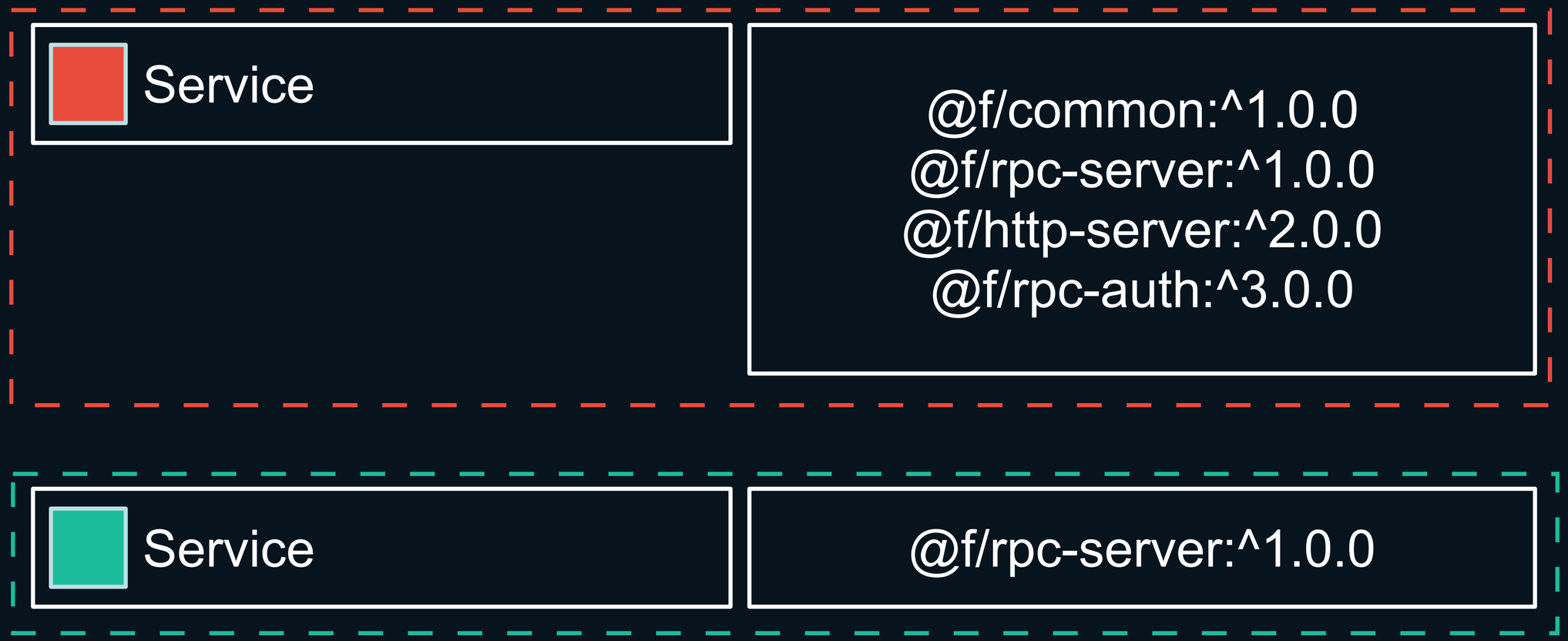


Versions usage

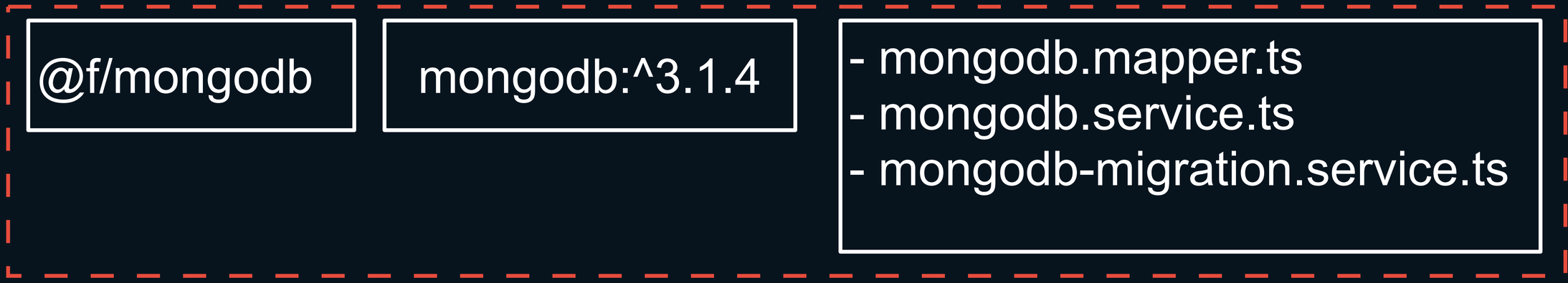
SemVer

Major > Minor > Patch

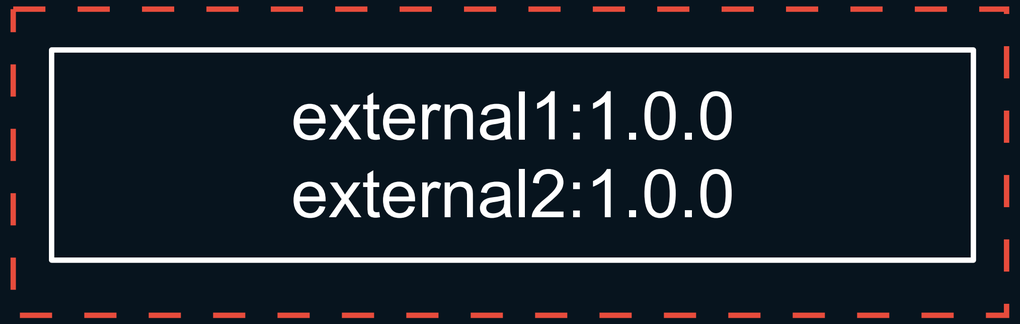
bc bc


Service
@f/common:^1.0.0
@f/rpc-server:^1.0.0
@f/http-server:^2.0.0
@f/rpc-auth:^3.0.0
Service
@f/rpc-server:^1.0.0

Invert you deps to packages

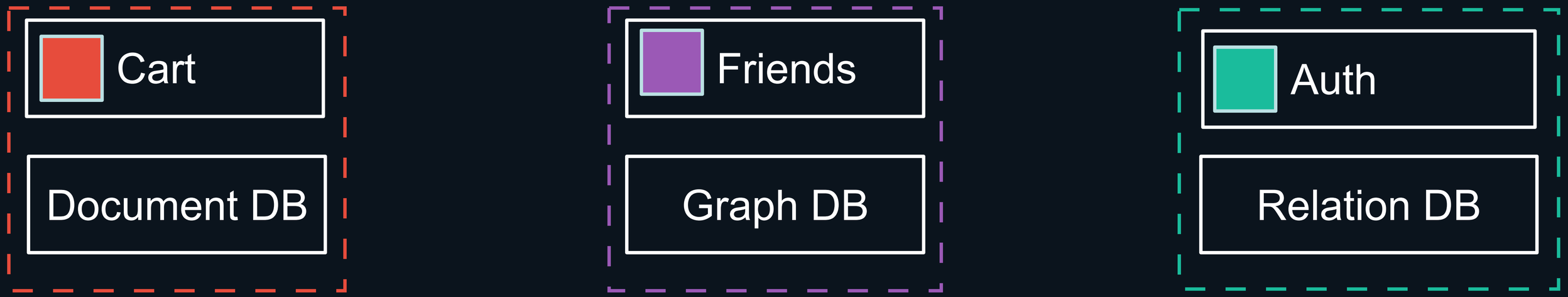


Static versions for external

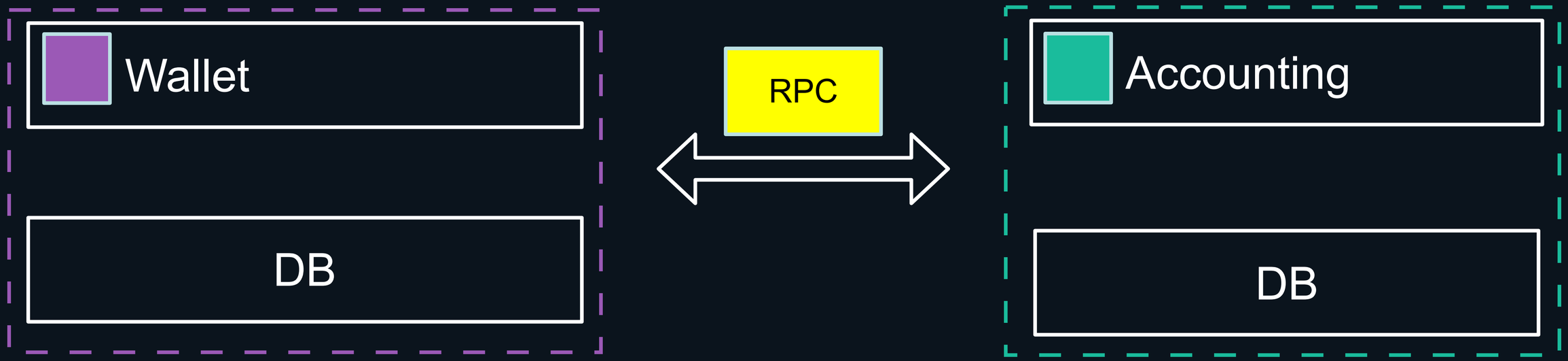


Database

Technological heterogeneity



Security



https://ru.wikipedia.org/wiki/PCI_DSS

Report

DEPARTMENT	ROLE	ACTIVE	BLOCKED	DELETED	TOTAL
Verified	WALLET_MEMBER	11	1	9	21
	ADMIN	1	0	0	1
	USERS_MANAGER	1	0	0	1
	TOTAL	23	13	1	9
Unverified	WALLET_MEMBER	235	1	56	292
	TOTAL	292	235	1	56

Report

DEPARTMENT	ROLE	ACTIVE	BLOCKED	DELETED	TOTAL
Verified	WALLET_MEMBER	11	1	9	21
	ADMIN	1	0	0	1
	USERS_MANAGER	1	0	0	1
	TOTAL	23	13	1	9
Unverified	WALLET_MEMBER	235	1	56	292
	TOTAL	292	235	1	56

Aggregation Values

Report

DEPARTMENT	ROLE	ACTIVE	BLOCKED	DELETED	TOTAL
Verified	WALLET_MEMBER	11	1	9	21
	ADMIN	1	0	0	1
	USERS_MANAGER	1	0	0	1
	TOTAL	23	13	1	9
Unverified	WALLET_MEMBER	235	1	56	292
	TOTAL	292	235	1	56

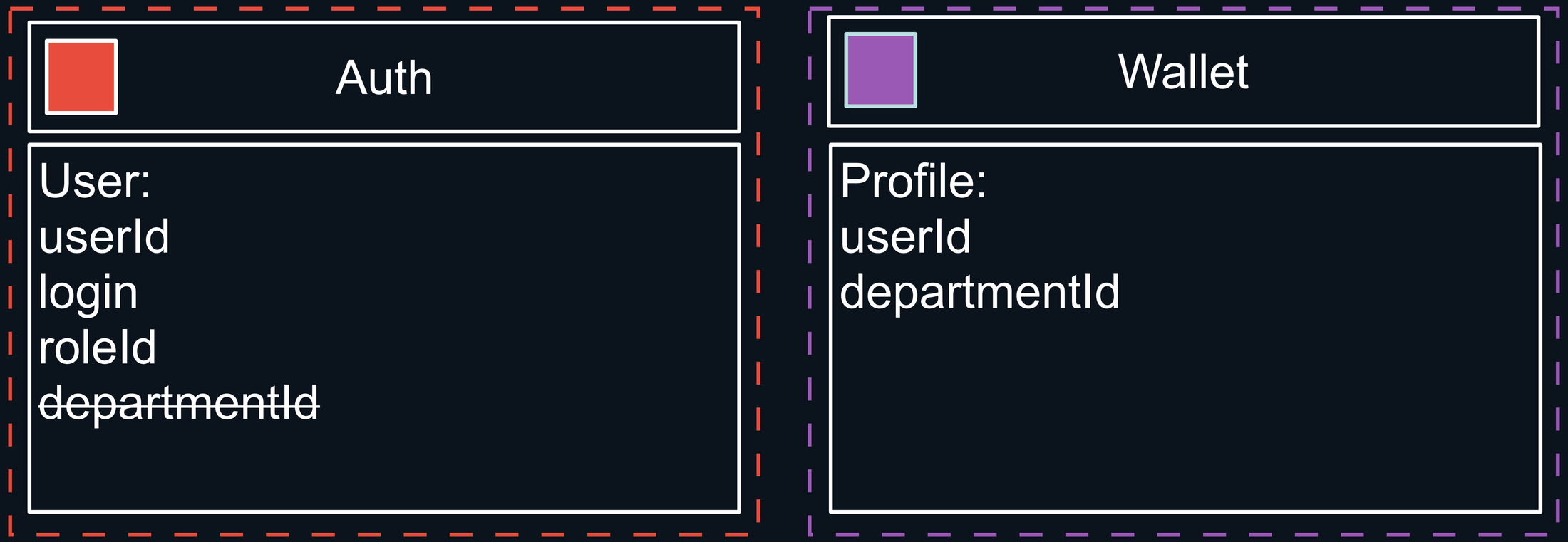
Aggregation Values

Report

DEPARTMENT	ROLE	ACTIVE	BLOCKED	DELETED	TOTAL
Verified	WALLET_MEMBER	11	1	9	21
	ADMIN	1	0	0	1
	USERS_MANAGER	1	0	0	1
	TOTAL	23	13	1	9
Unverified	WALLET_MEMBER	235	1	56	292
	TOTAL	292	235	1	56

Aggregation Values

Problem with Distributed



How to solve

How to solve

» Denormalization, store one data in different services

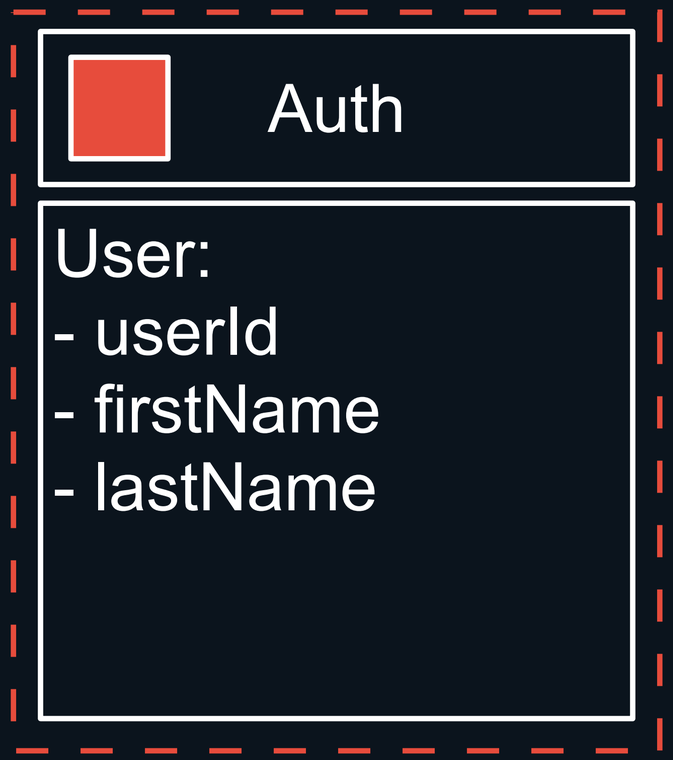
How to solve

» Denormalization, store one data in different services
» RPC to service when you needed data

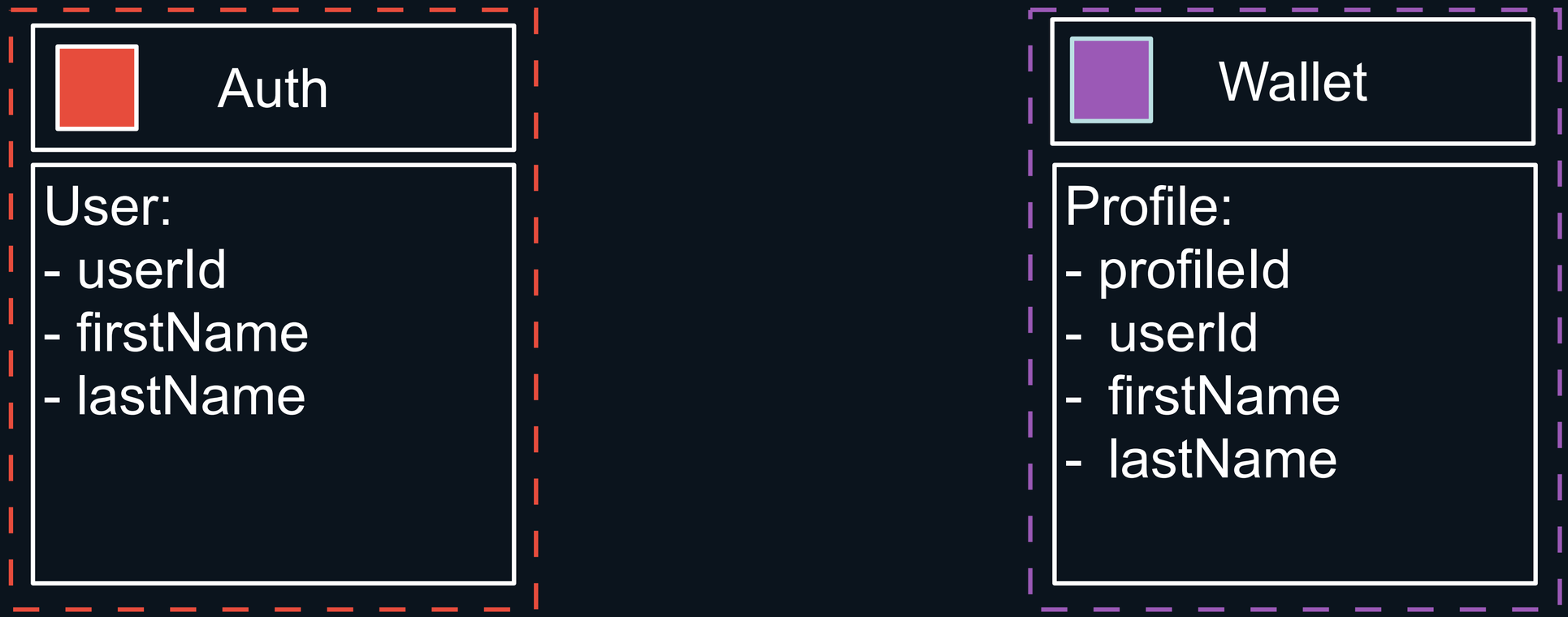
How to solve

» Denormalization, store one data in different services
» RPC to service when you needed data
» Merge two different databases to one, when it’s needed

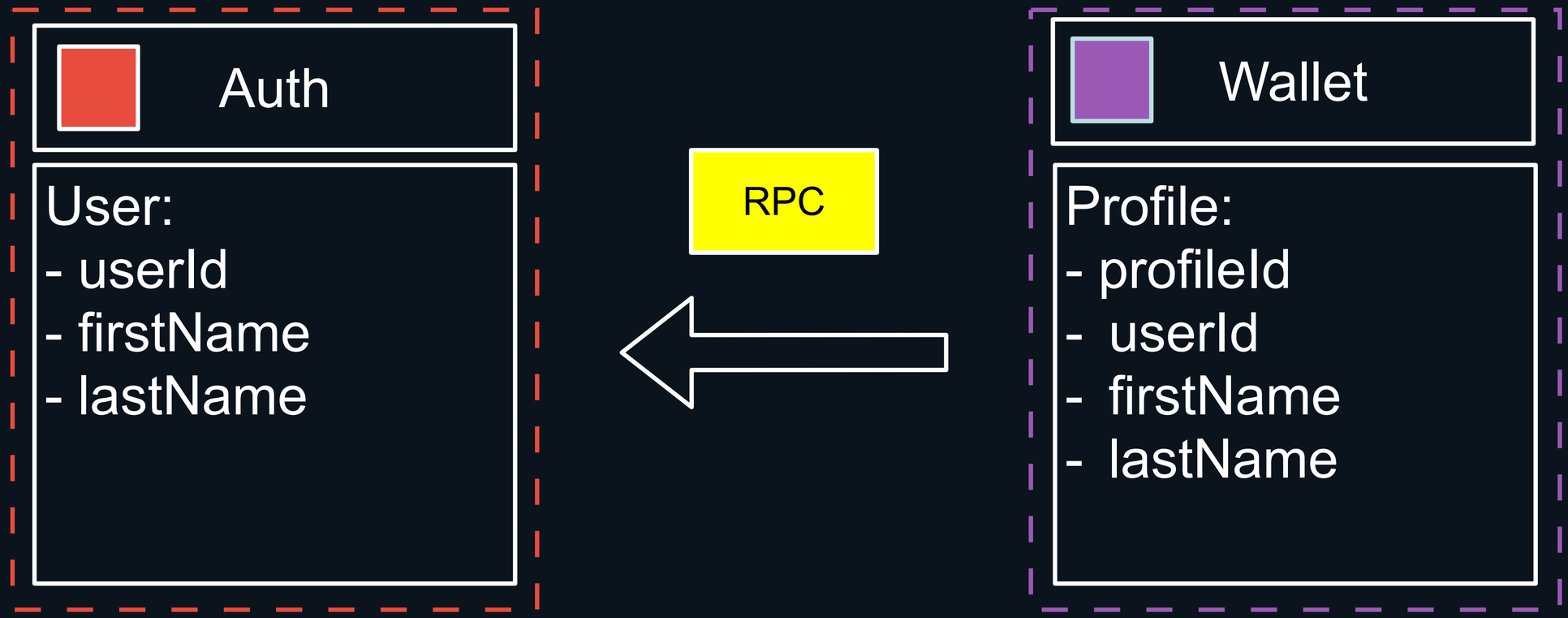
Migration



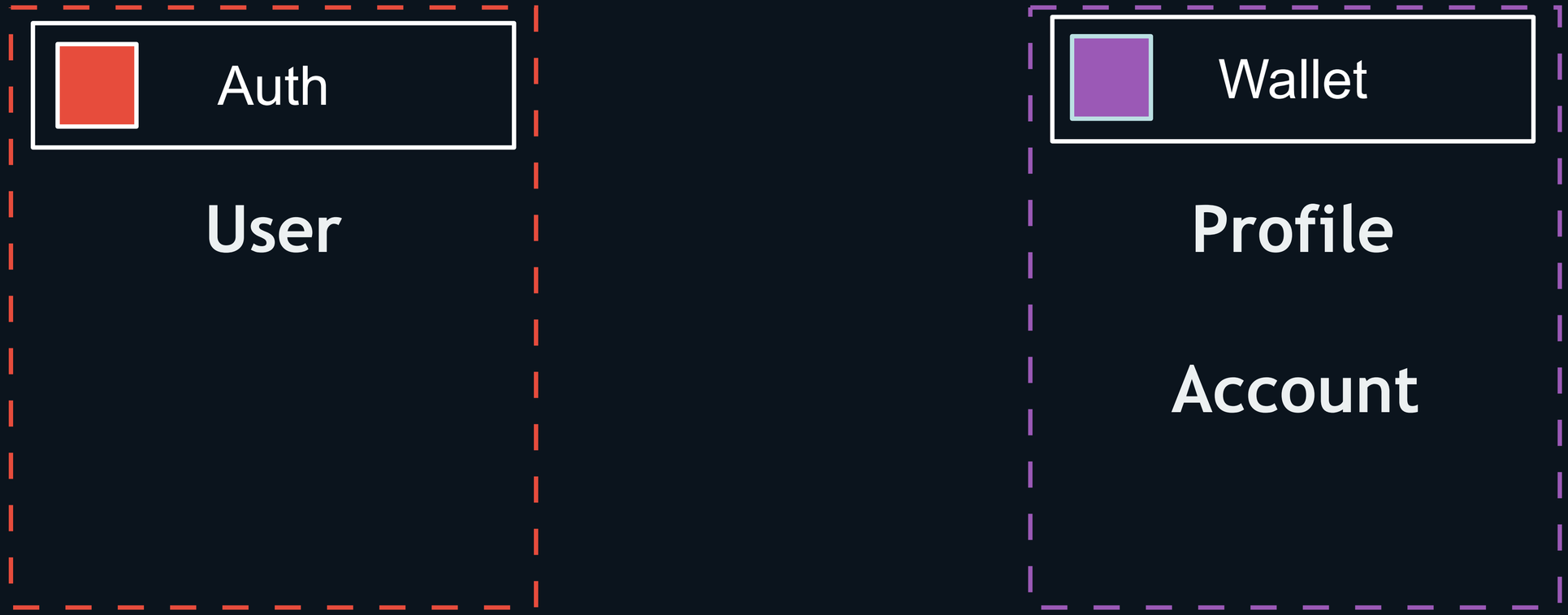
Migration


Auth
User:
- userId
- firstName
- lastName
Wallet
Profile:
- profileId
- userId
- firstName
- lastName

Migration


Auth
User:
- userId
- firstName
- lastName
RPC
Wallet
Profile:
- profileId
- userId
- firstName
- lastName

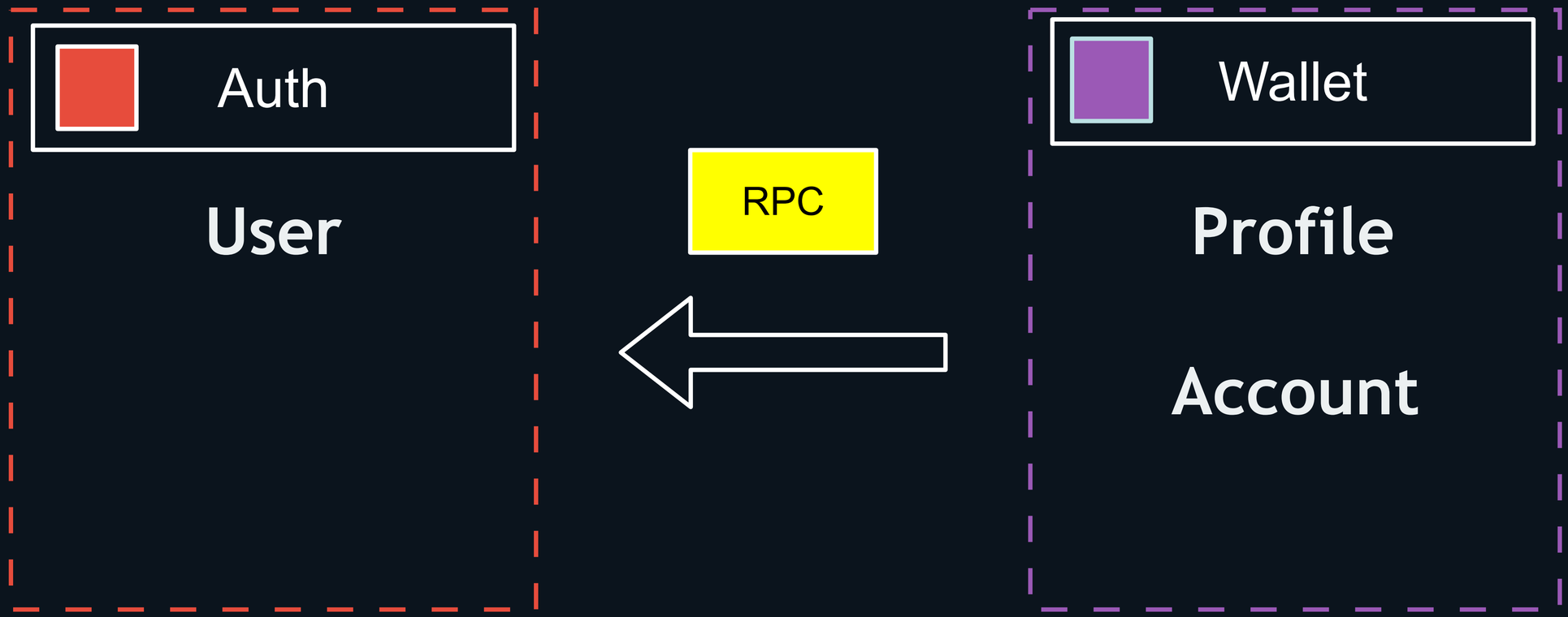
Complex



Complex

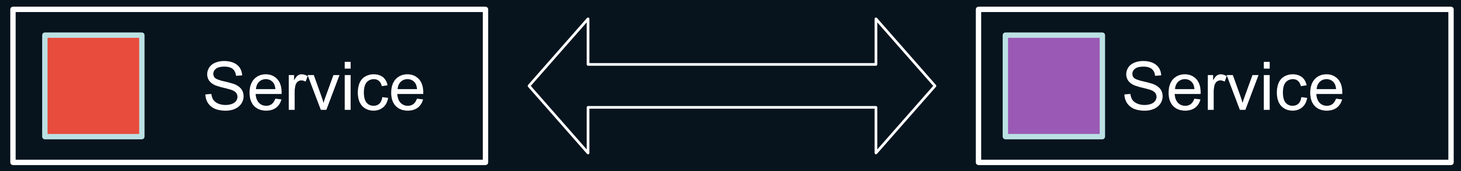
How to rollback?

Complex

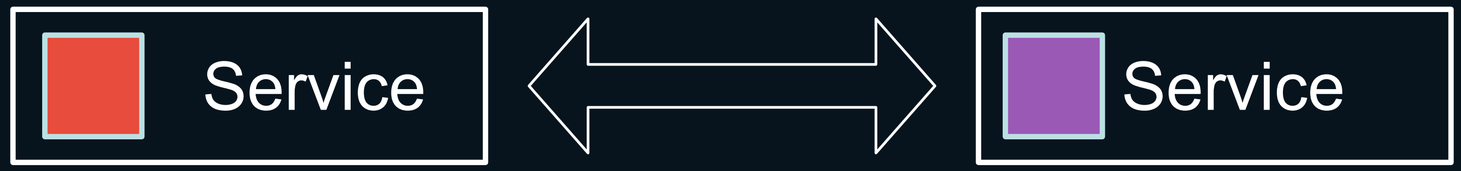


How to rollback?

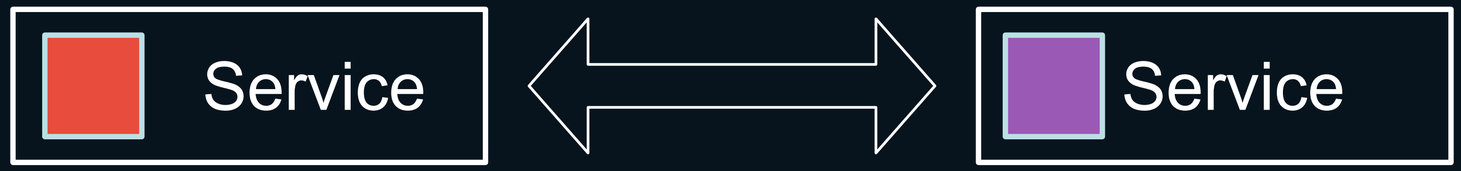
Internal communication in MSA


Service
Service

Contract

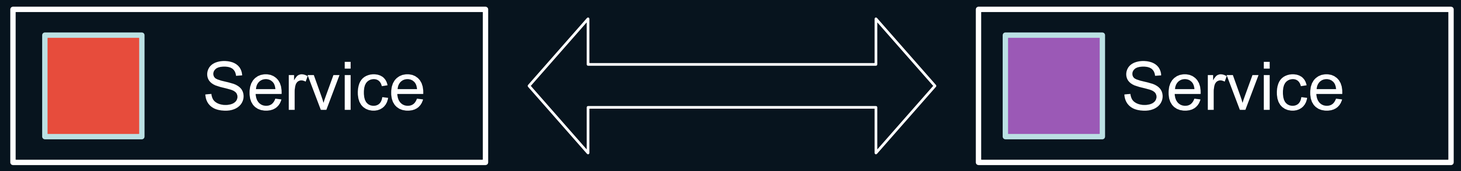


Contract



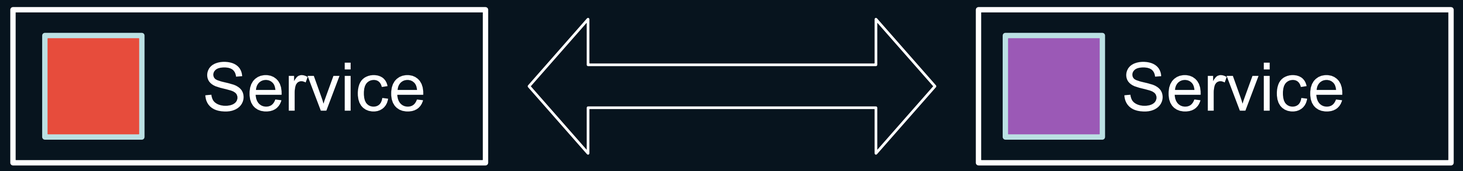
» Protocol

Contract

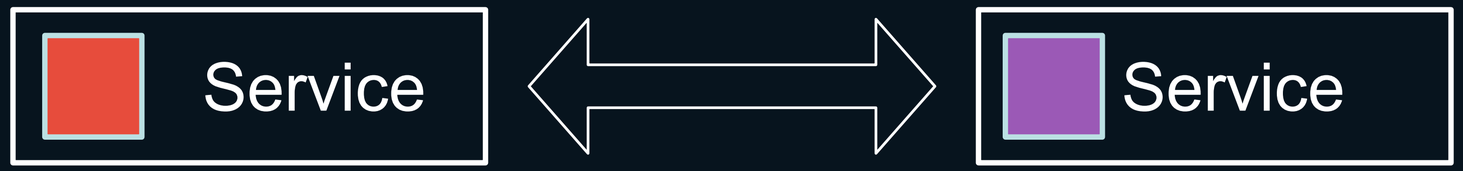


» Protocol

» Data validation


Service
Service

Contract as ~~JSON + JSON-SCHEMA~~



JSON + JSON-SCHEMA

JSON + JSON-SCHEMA

» It’s not strict

JSON + JSON-SCHEMA

» It’s not strict
» No protocol

JSON + JSON-SCHEMA

» It’s not strict
» No protocol
» No generation for clients & servers

JSON + JSON-SCHEMA

» It’s not strict
» No protocol
» No generation for clients & servers
» Slow serialization/deserialization

JSON + JSON-SCHEMA

» It’s not strict
» No protocol
» No generation for clients & servers
» Slow serialization/deserialization
» Traffic, because it’s not binary

But what is requirements?

But what is requirements?

» Strict

But what is requirements?

» Strict
» One protocol generation for both client and server

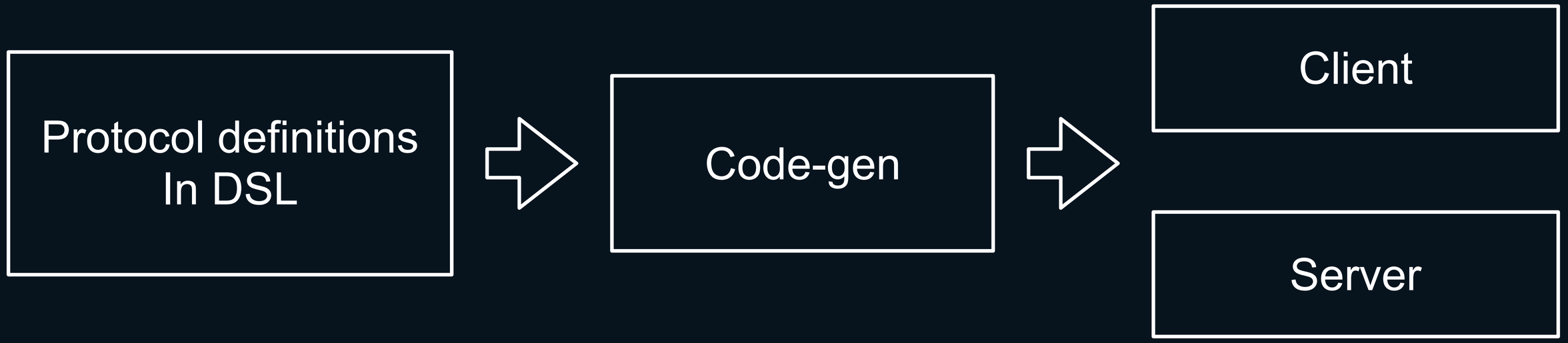
But what is requirements?

» Strict
» One protocol generation for both client and server
» Codegen for any language

But what is requirements?

» Strict
» One protocol generation for both client and server
» Codegen for any language
» Performance ;)

Thrift / FlatsBuffers / Protocol Buffers



Apache Thrift

```
enum Operation {
  PLUS = 1,
}

struct Work {
  1: i32 num1 = 0,
  2: i32 num2,
  3: Operation op,
}

exception InvalidOperation {
  1: i32 whatOp,
  2: string why
}

service Calculator {
   i64 plus(1:i32 num1, 2:i32 num2),
   i32 calculate(1:Work w) throws (1:InvalidOperation ouch),
}
```

Teamwork organization

How to split teams?

By language/technology

By language/technology

JS Backend

By language/technology

JS Backend

Java Backend

JS Team

JS Team

Developer1

JS Team

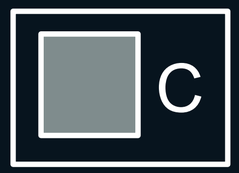
Developer1

Developer2

JS Team

Developer1

Developer2



Developer1

JS Team

Developer1

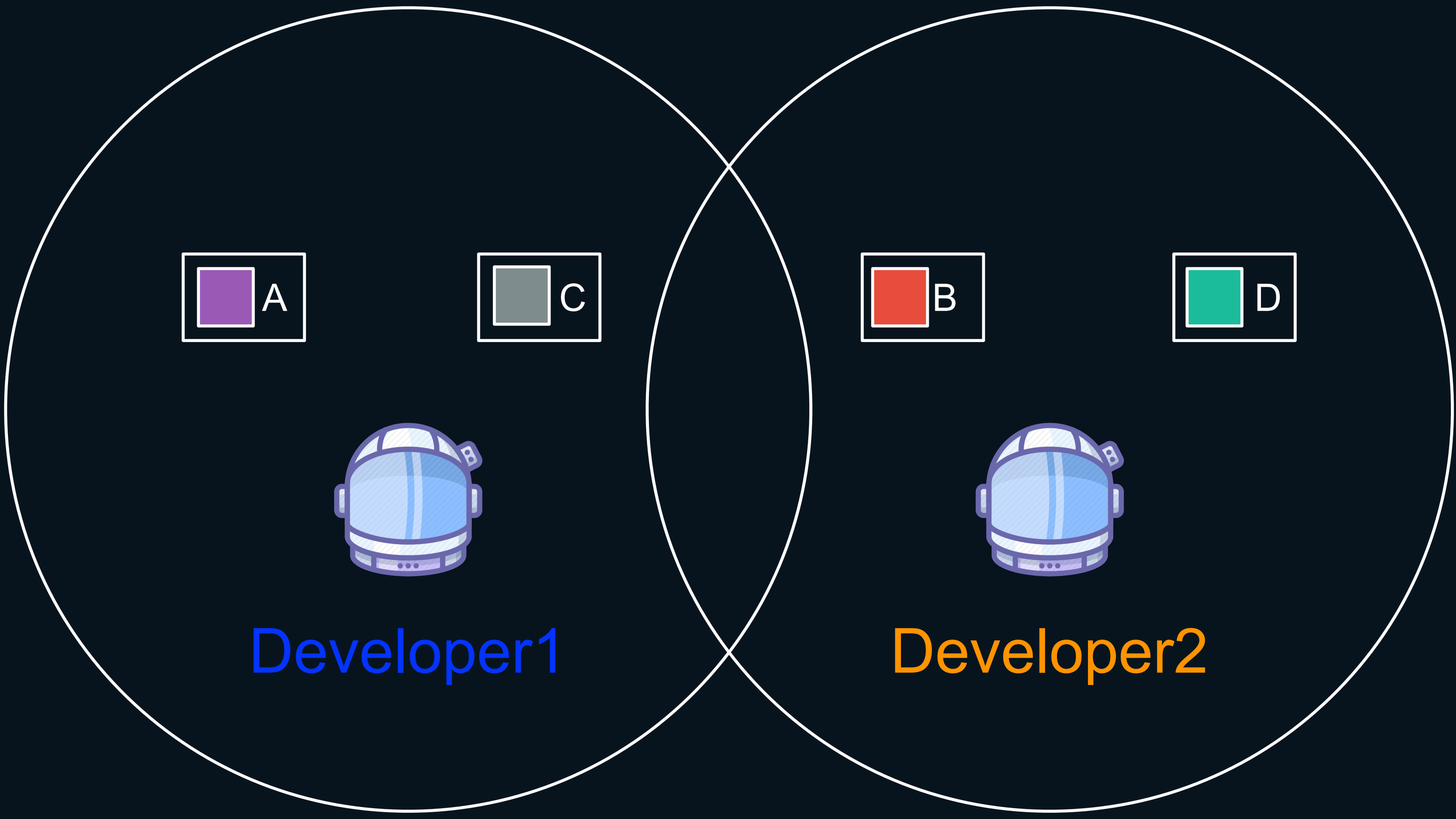
Developer2

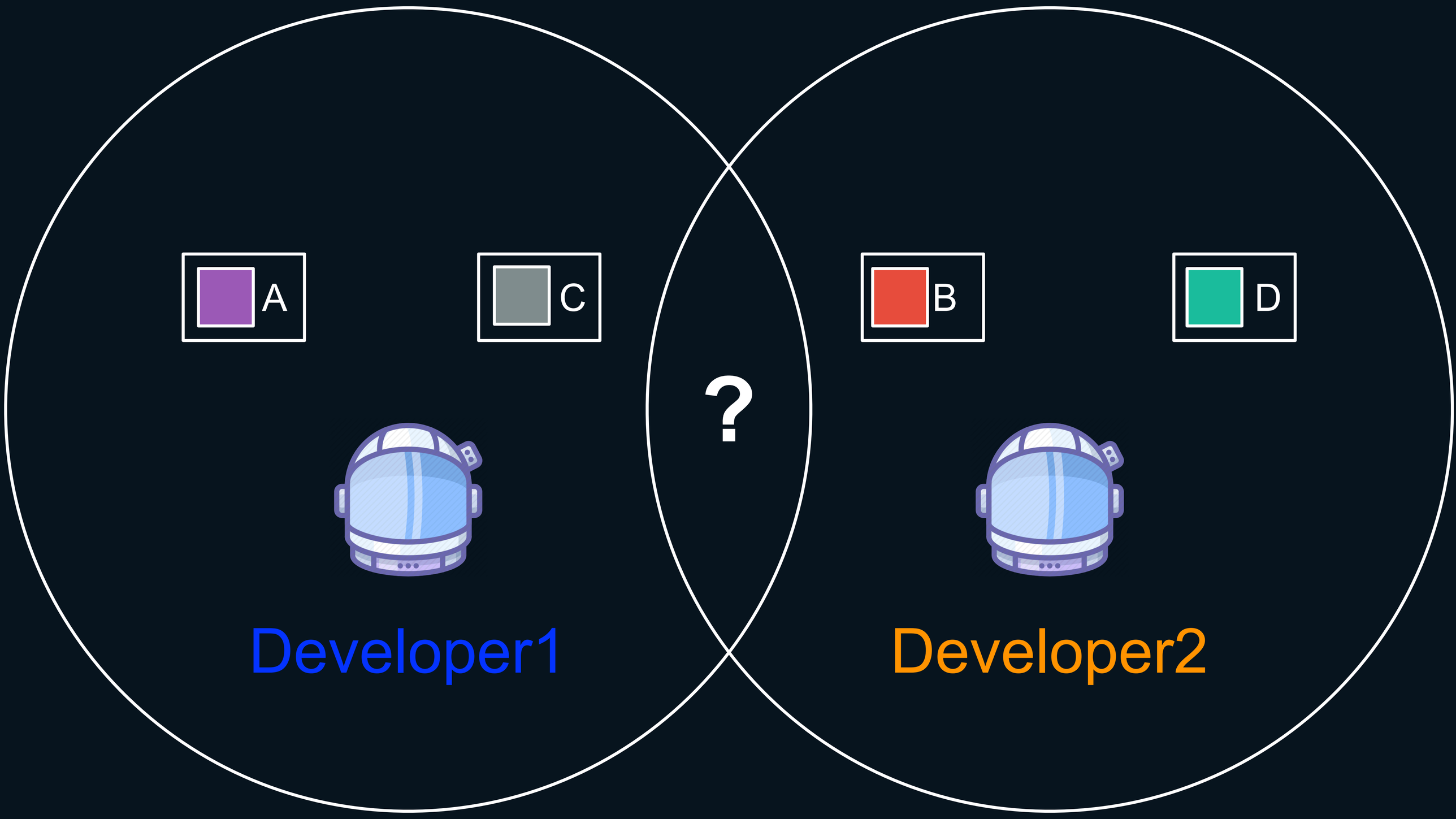
Developer1

Developer2

Developer1

Developer2


A
C
B
D
Developer1
Developer2


A
C
?
B
D
Developer1
Developer2

— Мелвин Конвей (Melvin E. Conway)"

Организации, проектирующие системы, ограничены дизайном, который копирует структуру коммуникации в этой организации.

— Мелвин Конвей (Melvin E. Conway)"

Communication is epic important

Communication is epic important

The intersection of knowledge

The intersection of knowledge

» One language for microservice

The intersection of knowledge

» One language for microservice
» One template/organization for each microservice

The intersection of knowledge

» One language for microservice
» One template/organization for each microservice
» Shared code

The intersection of knowledge

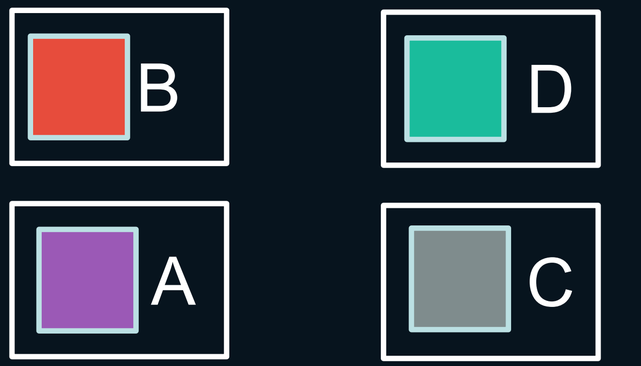
» One language for microservice
» One template/organization for each microservice
» Shared code
» CI/CD

The intersection of knowledge

» One language for microservice
» One template/organization for each microservice
» Shared code
» CI/CD
» Communication (Protocol)

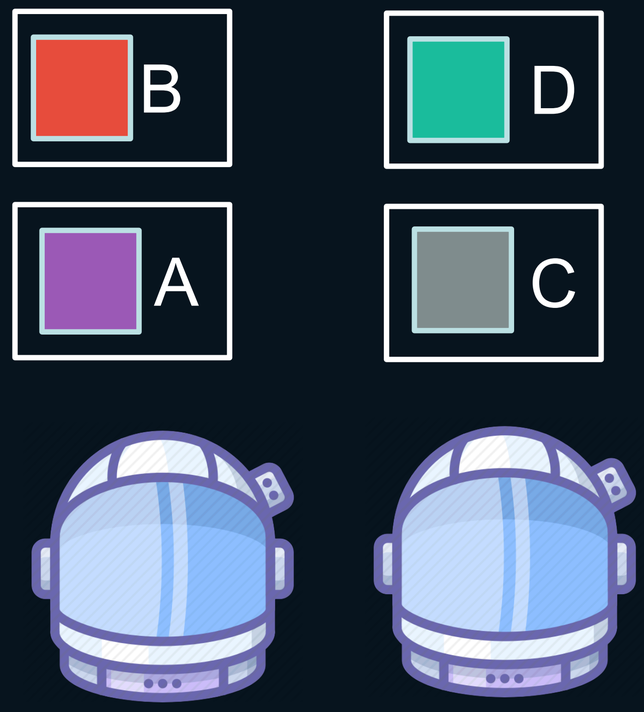
The intersection of knowledge

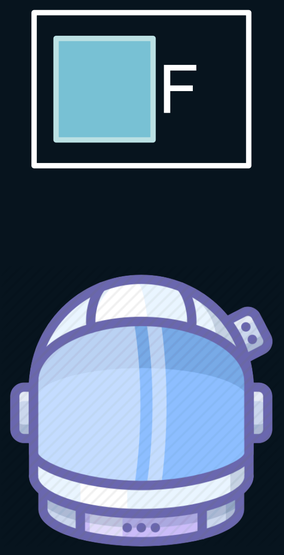
» One language for microservice
» One template/organization for each microservice
» Shared code
» CI/CD
» Communication (Protocol)
» Documentation

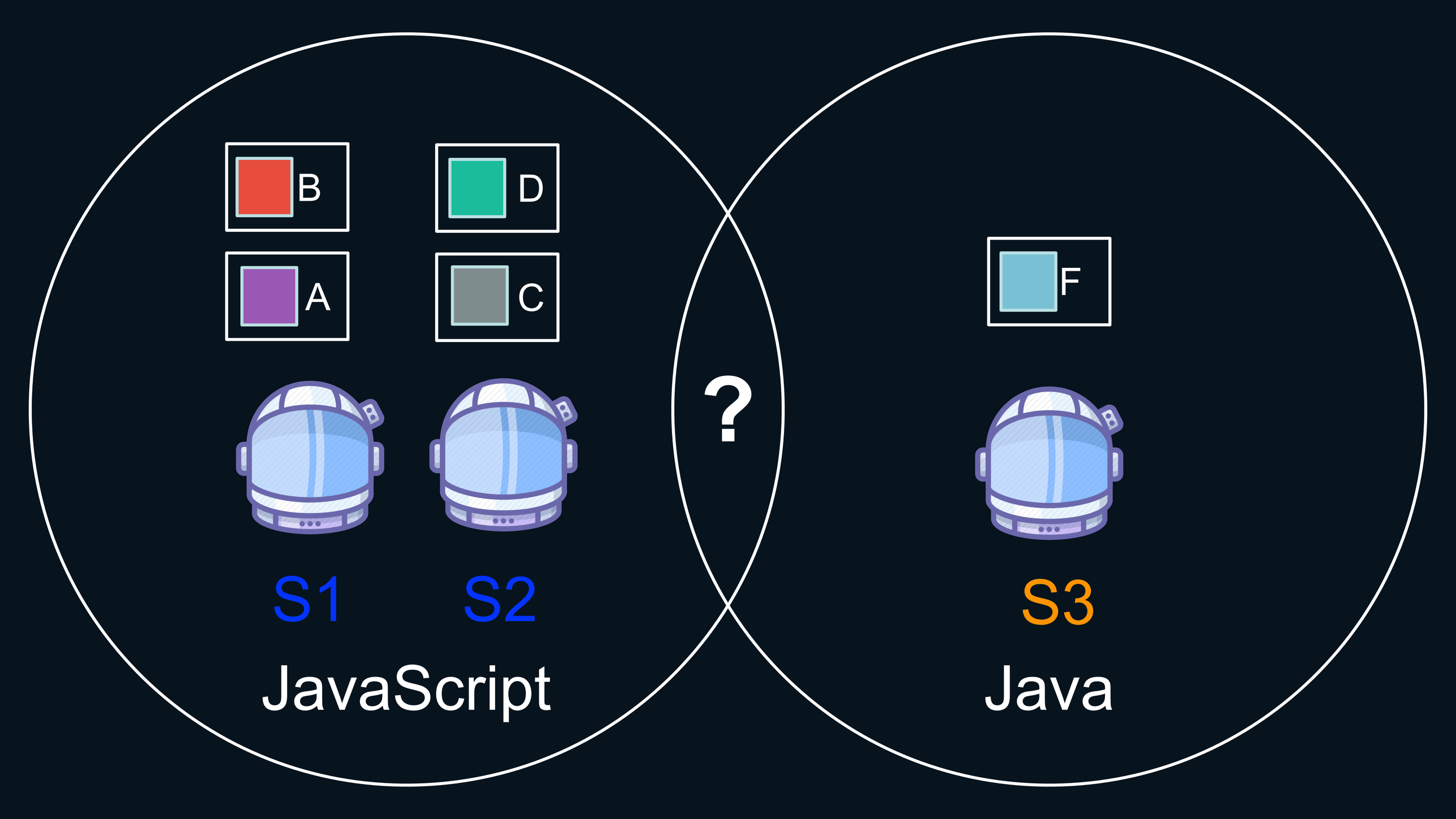


S1 S2

JavaScript


B
D
A
C
S1
S2
JavaScript


F
S3
Java


B
D
A
C
?
F
S1
S2
S3
JavaScript
Java

The intersection of knowledge

The intersection of knowledge

» ~~One language for microservice~~

The intersection of knowledge

» ~~One language for microservice~~
» ~~One template/organization for each microservice~~

The intersection of knowledge

» ~~One language for microservice~~
» ~~One template/organization for each microservice~~
» ~~Shared code~~

The intersection of knowledge

- ~~One language for microservice~~
- ~~One template/organization for each microservice~~
- ~~Shared code~~
- CI/CD

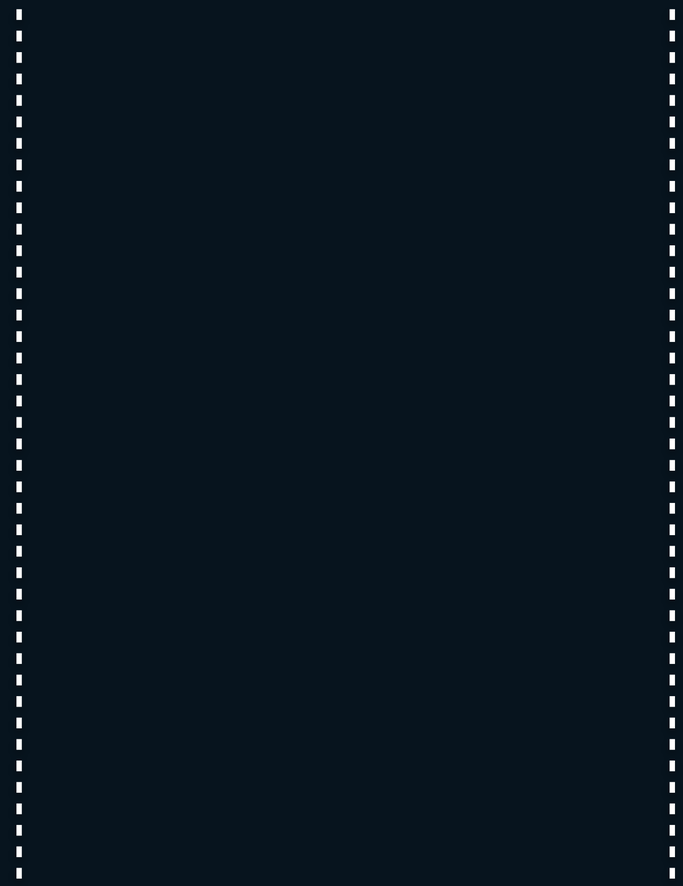
The intersection of knowledge

- ~~One language for microservice~~
- ~~One template/organization for each microservice~~
- ~~Shared code~~
- CI/CD
- Communication (Protocol)

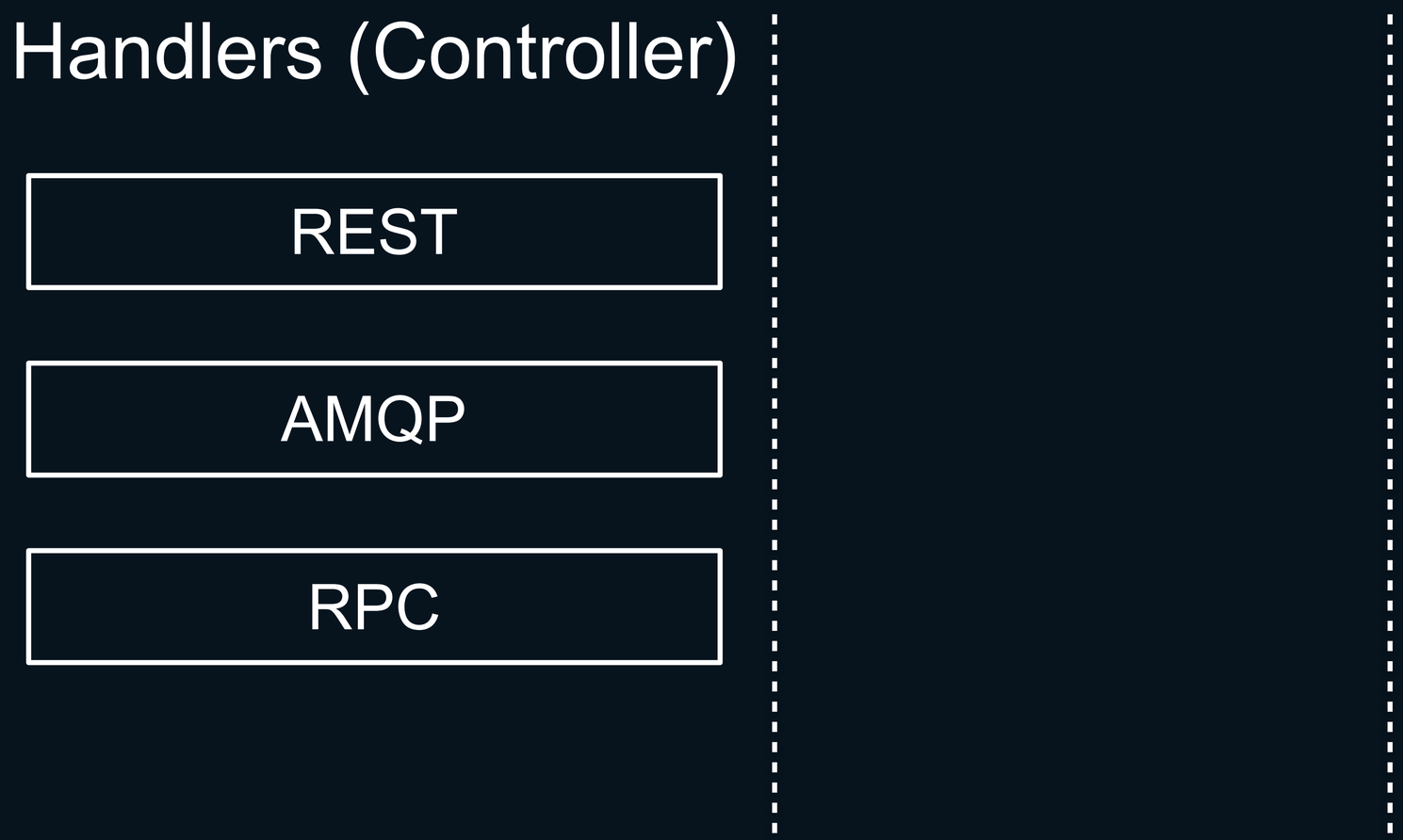
The intersection of knowledge

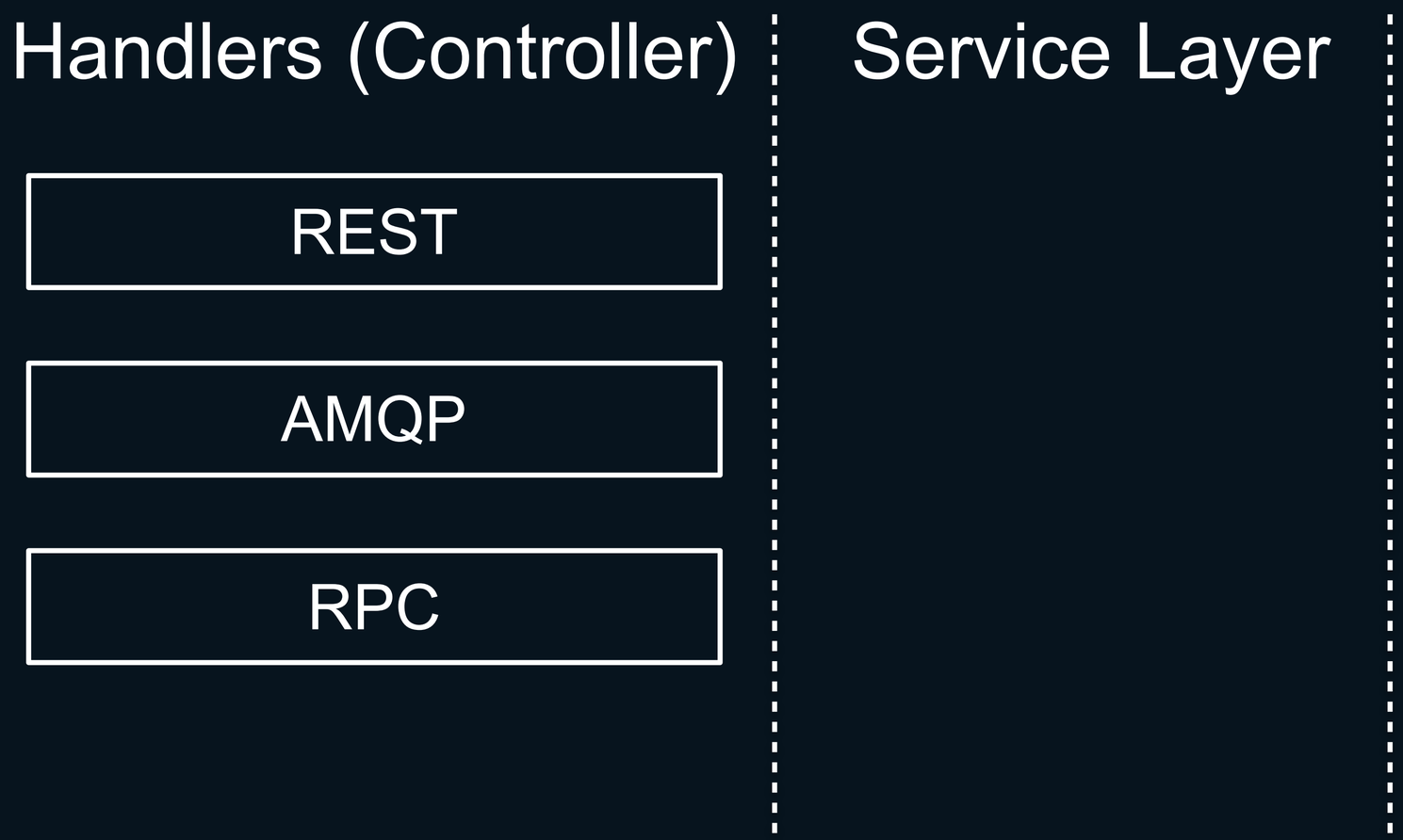
- ~~One language for microservice~~
- ~~One template/organization for each microservice~~
- ~~Shared code~~
- CI/CD
- Communication (Protocol)
- Documentation

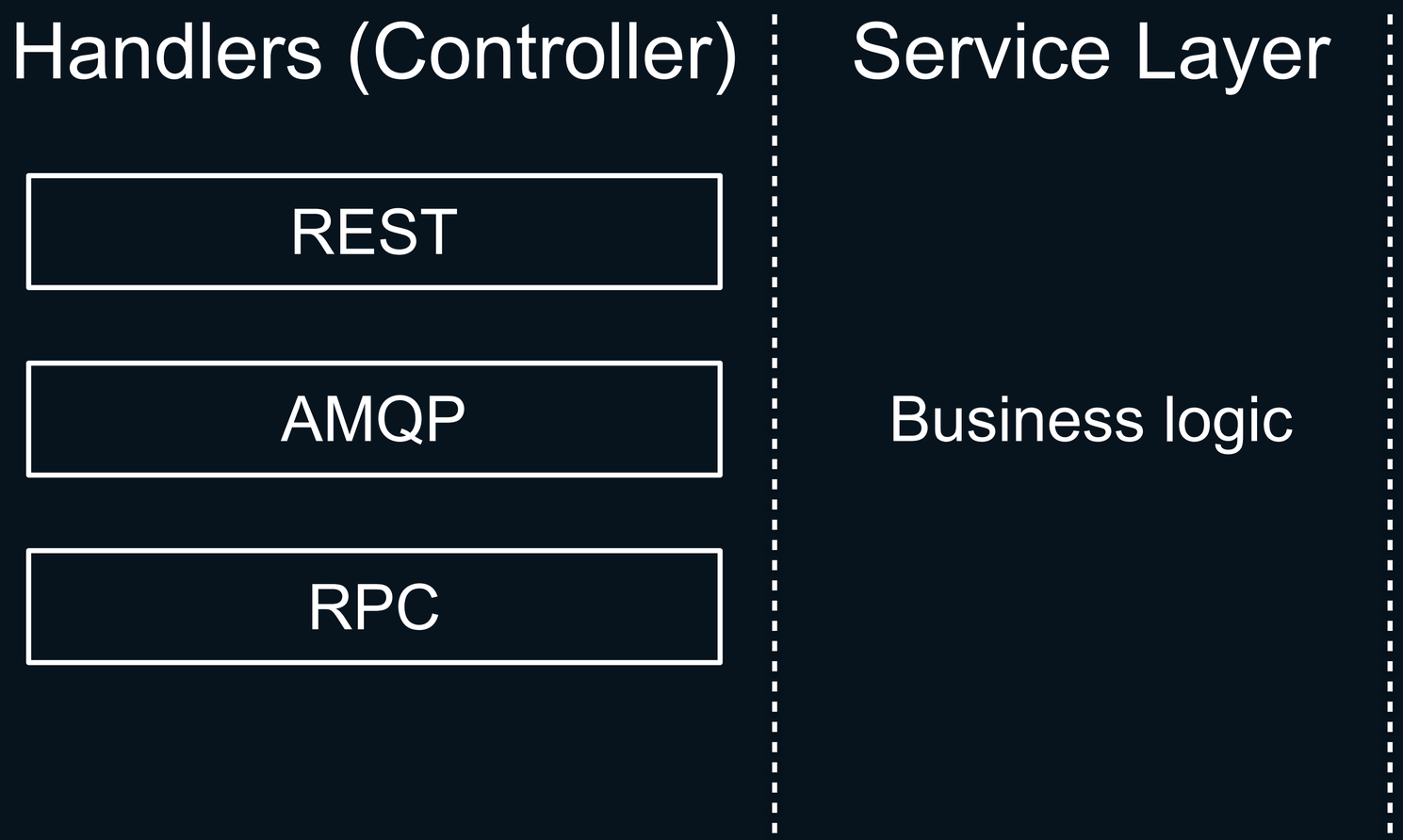
Service Architecture

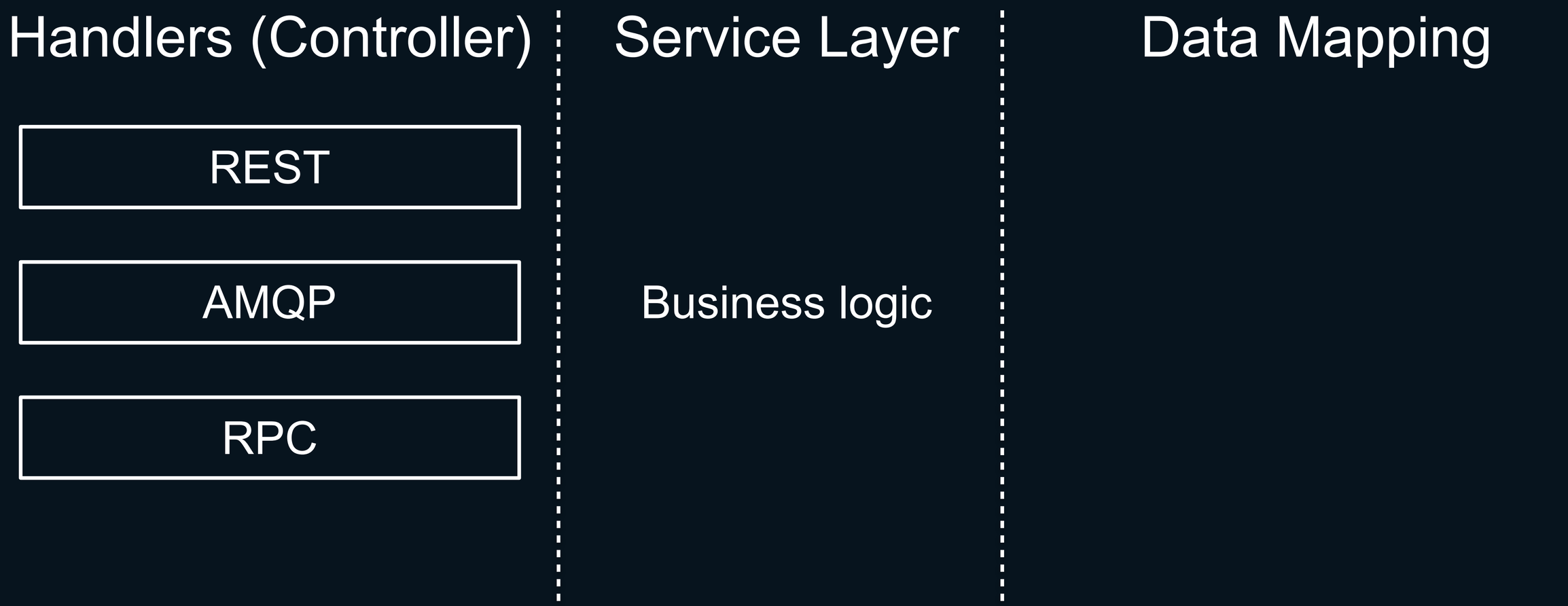


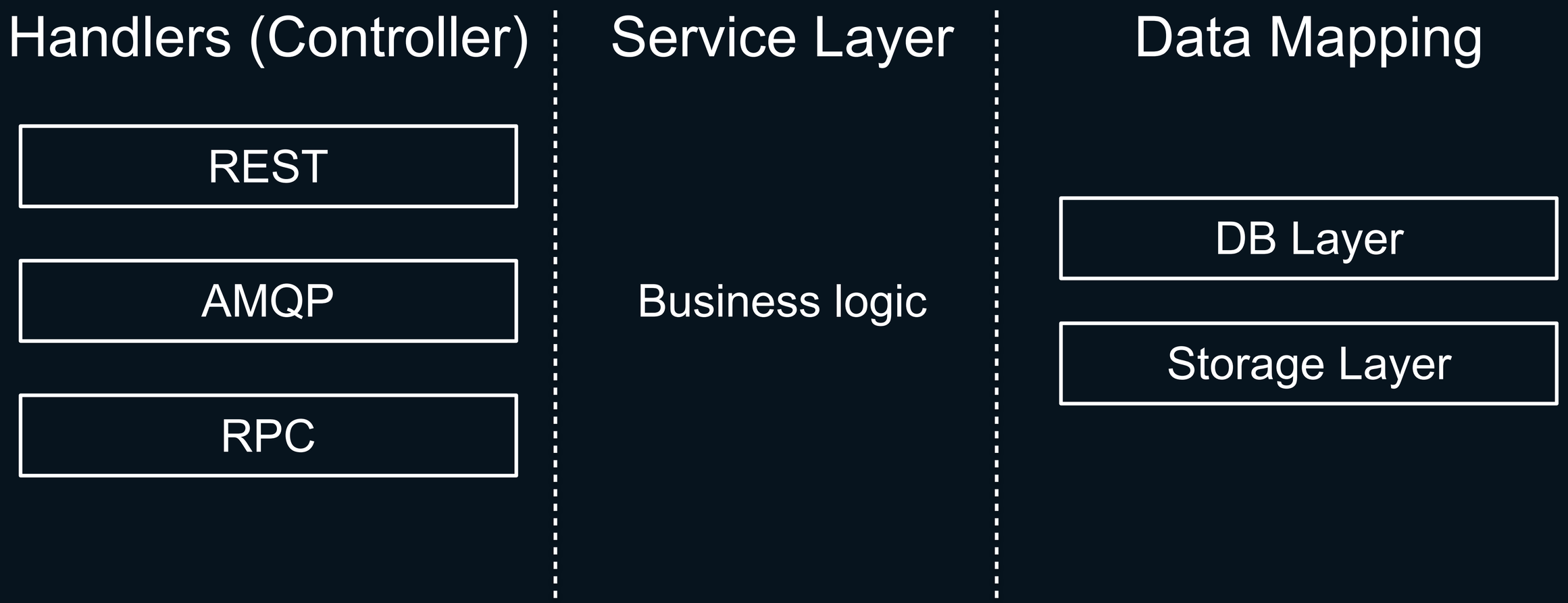
Handlers (Controller)


Handlers (Controller)
REST
AMQP
RPC

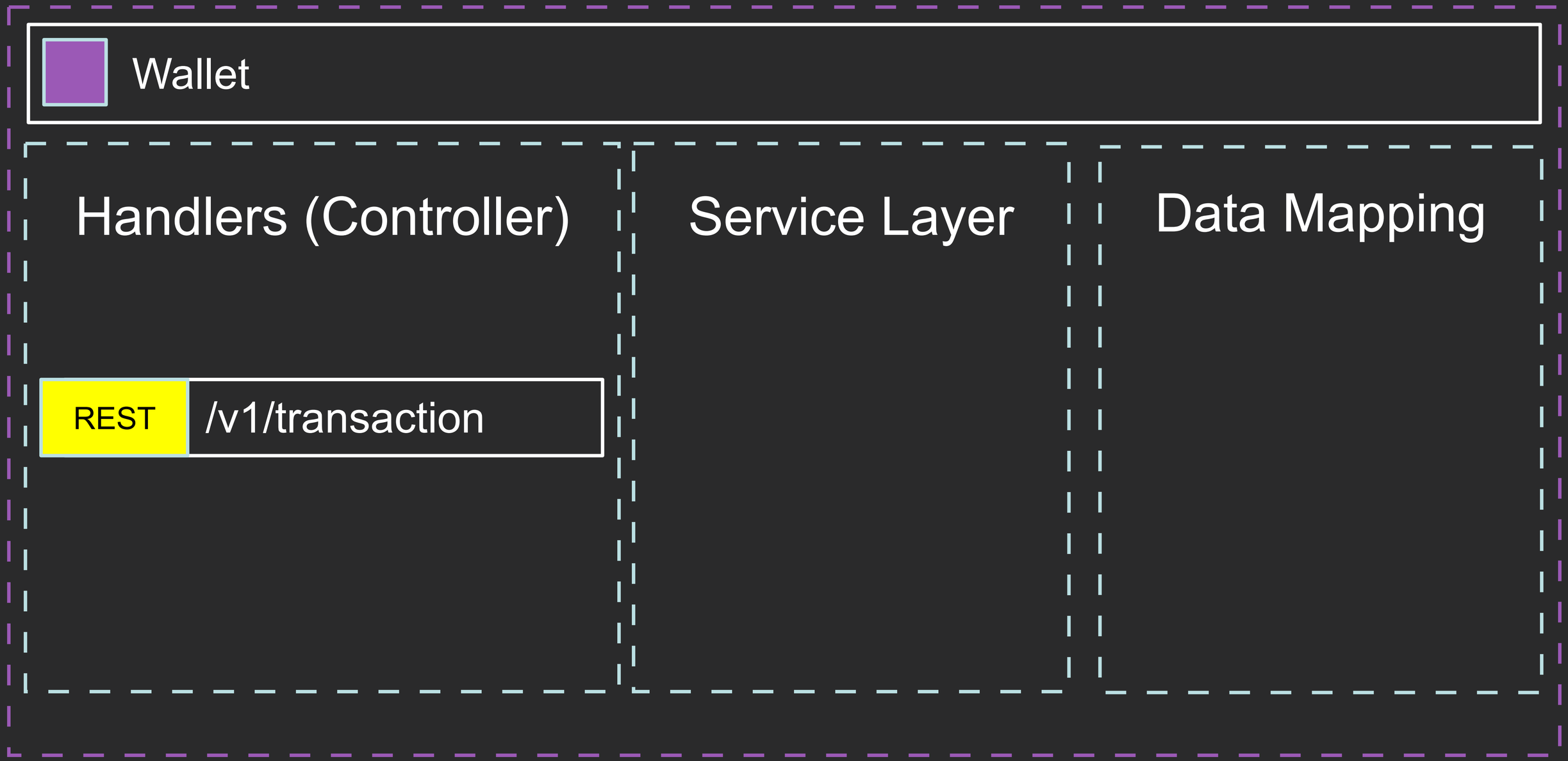

Handlers (Controller)
Service Layer
REST
AMQP
RPC


Handlers (Controller)
Service Layer
REST
AMQP
RPC
Business logic


Handlers (Controller)
Service Layer
Data Mapping
REST
AMQP
RPC
Business logic

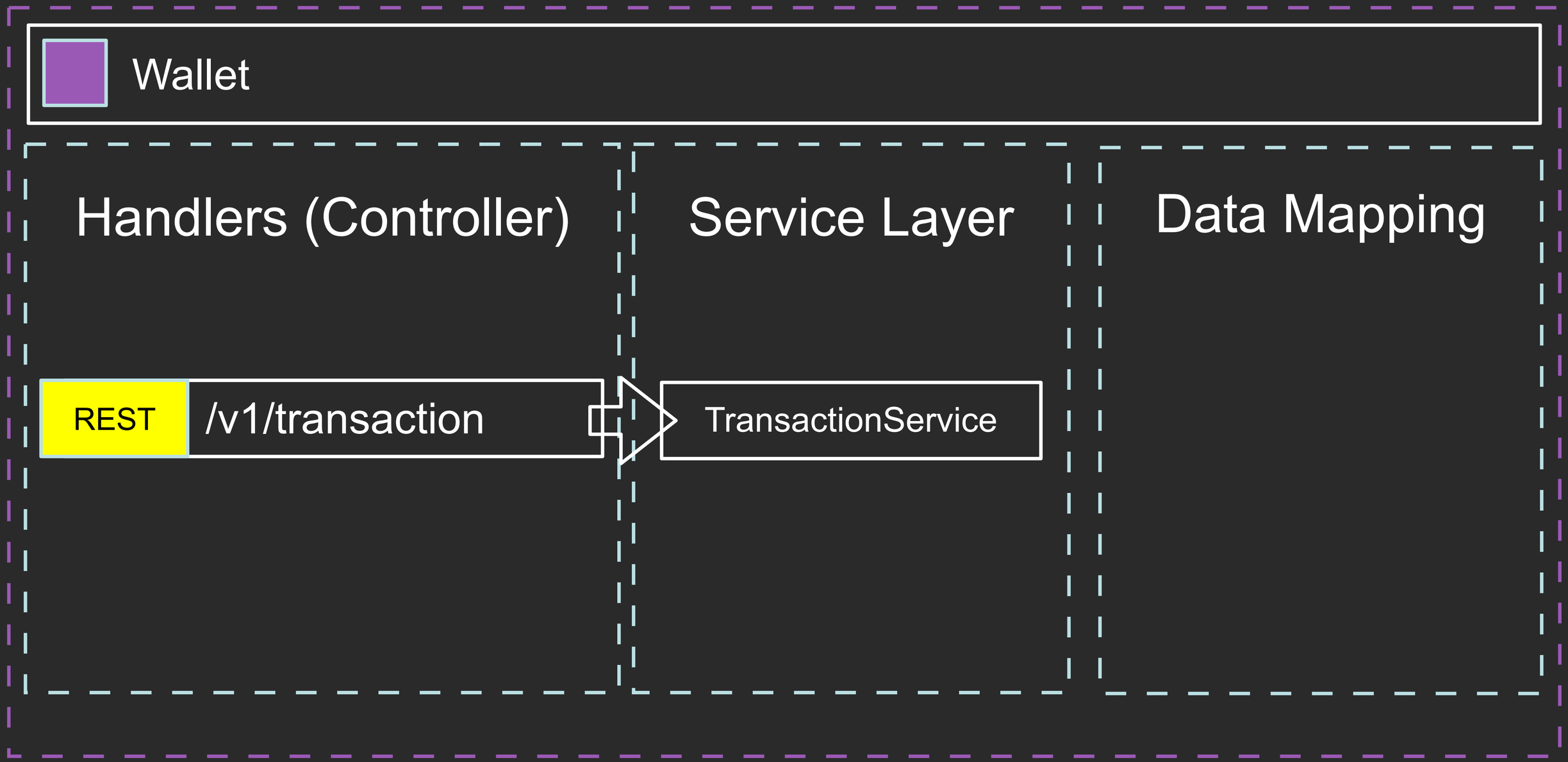

Handlers (Controller)
Service Layer
Data Mapping
REST
AMQP
RPC
Business logic
DB Layer
Storage Layer

REST GET /transaction/{id}



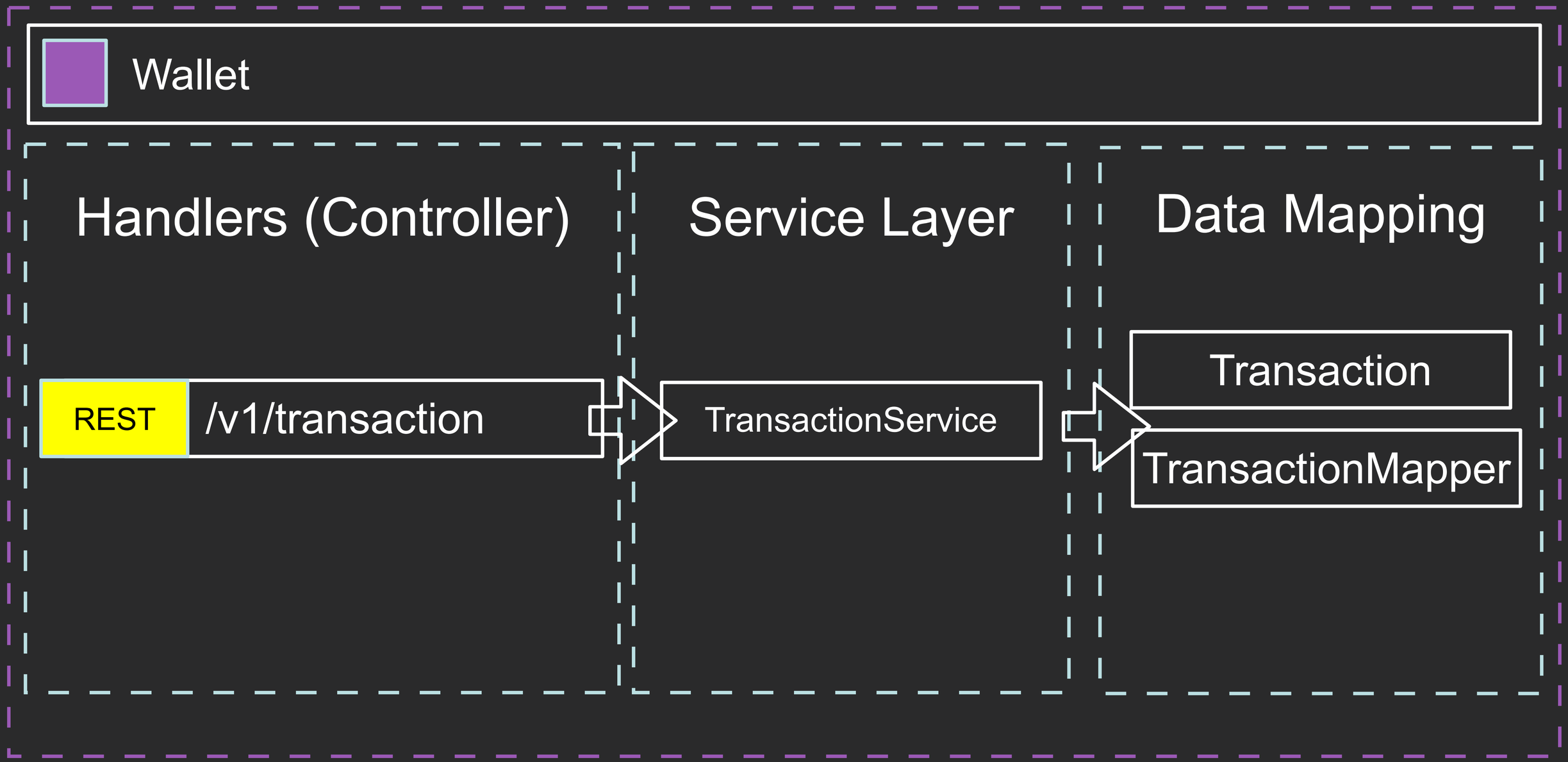
REST

GET /transaction/{id}

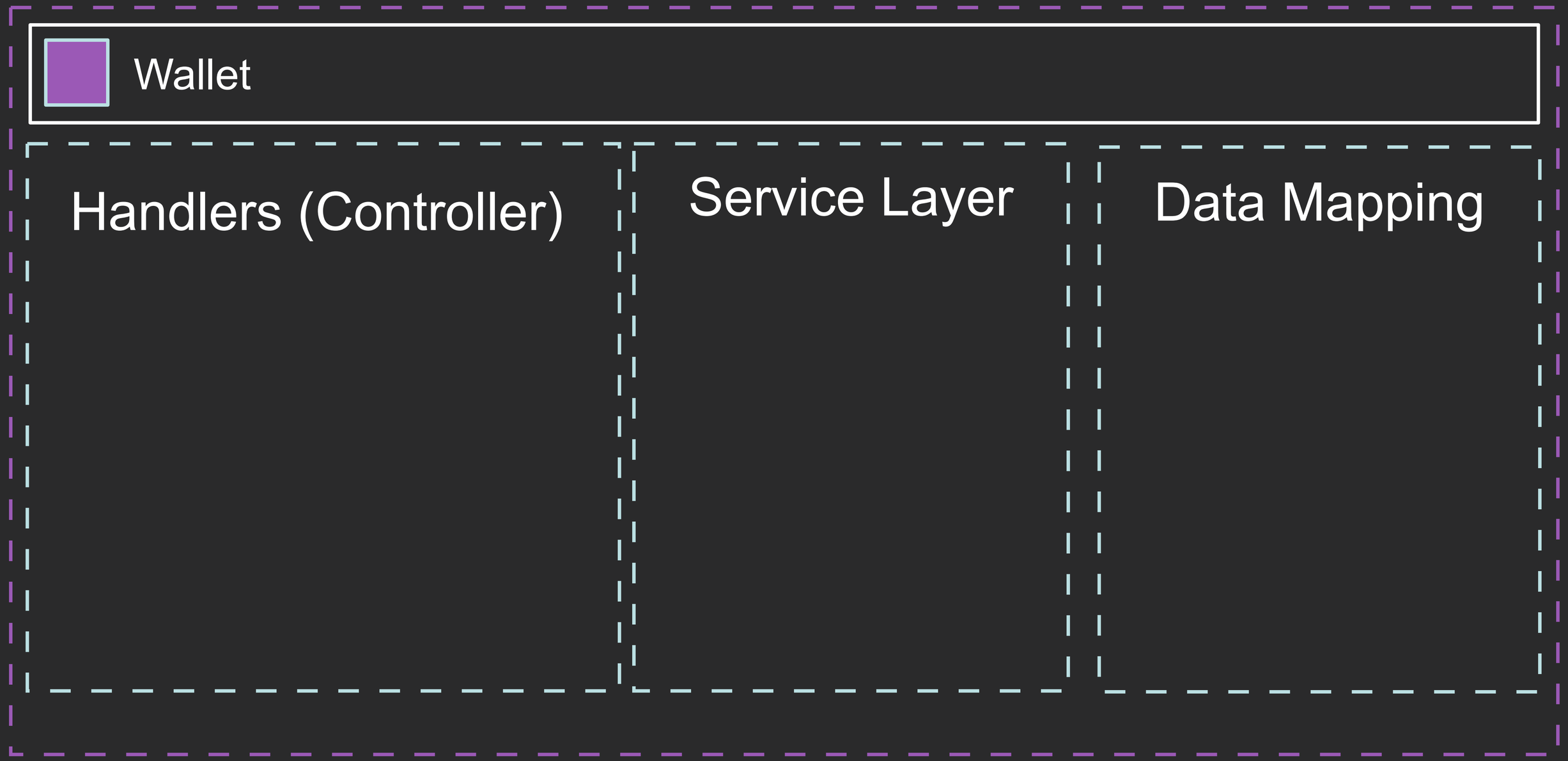


REST

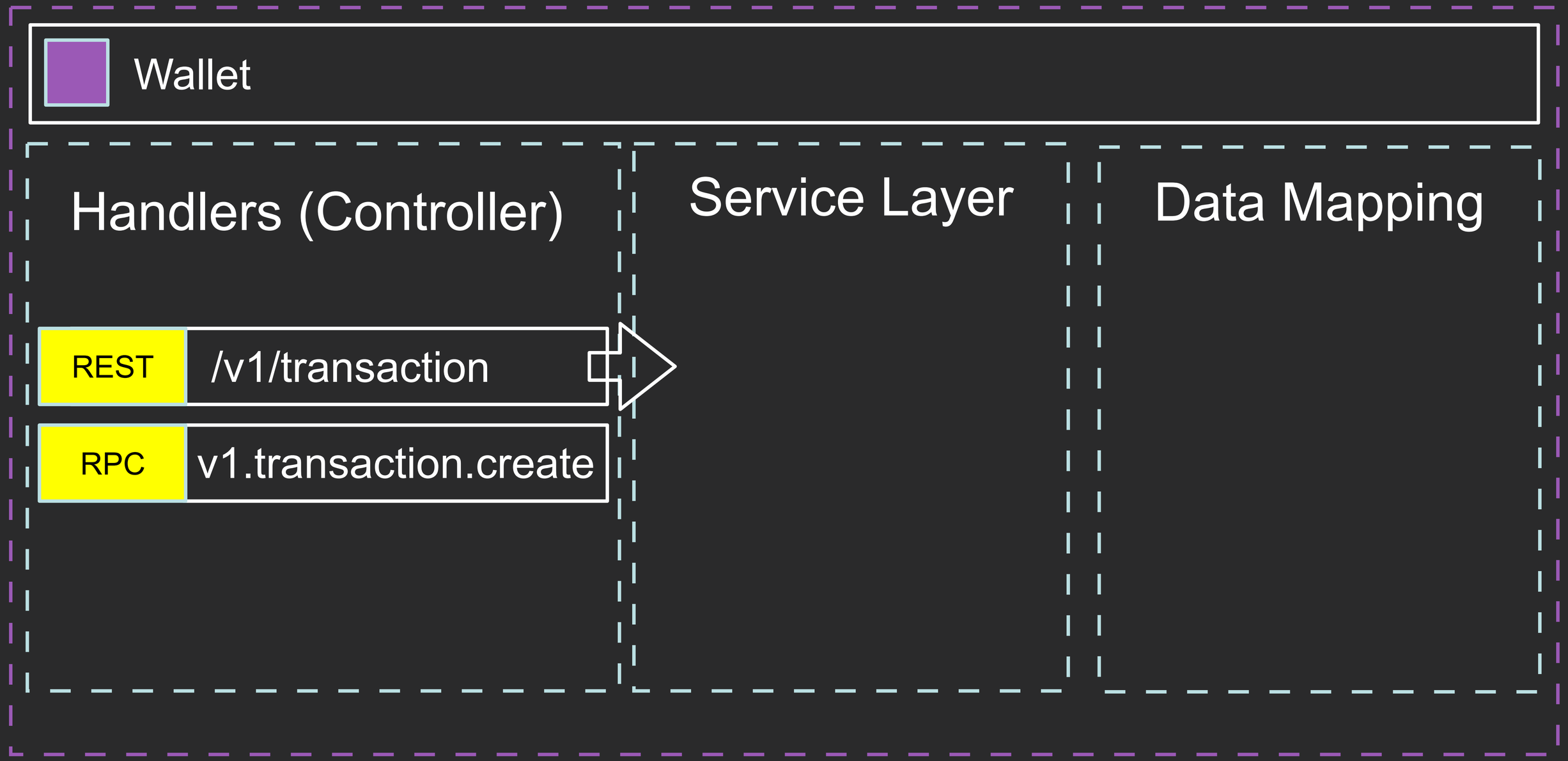
GET /transaction/{id}



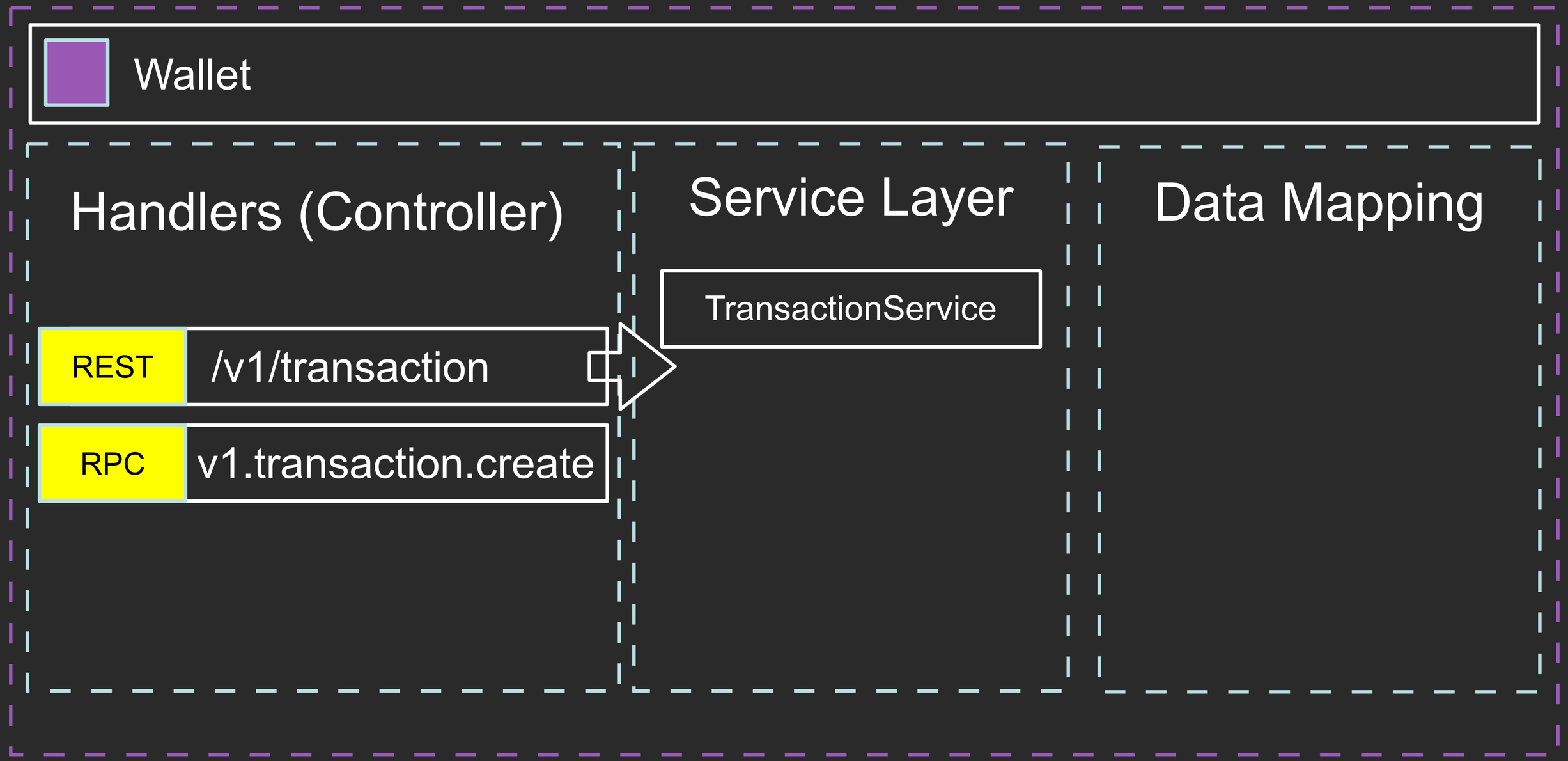
It can be more complex



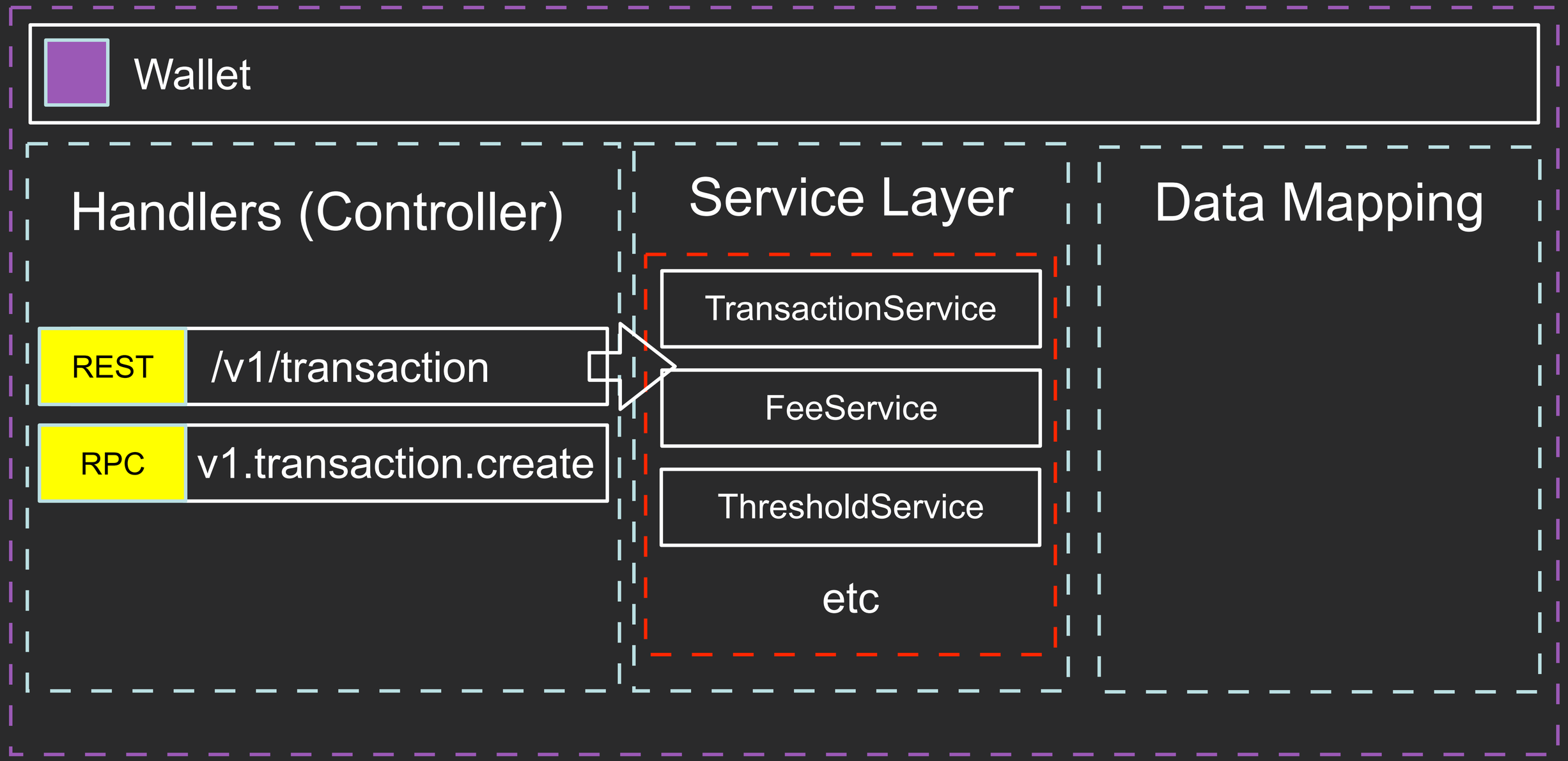
It can be more complex



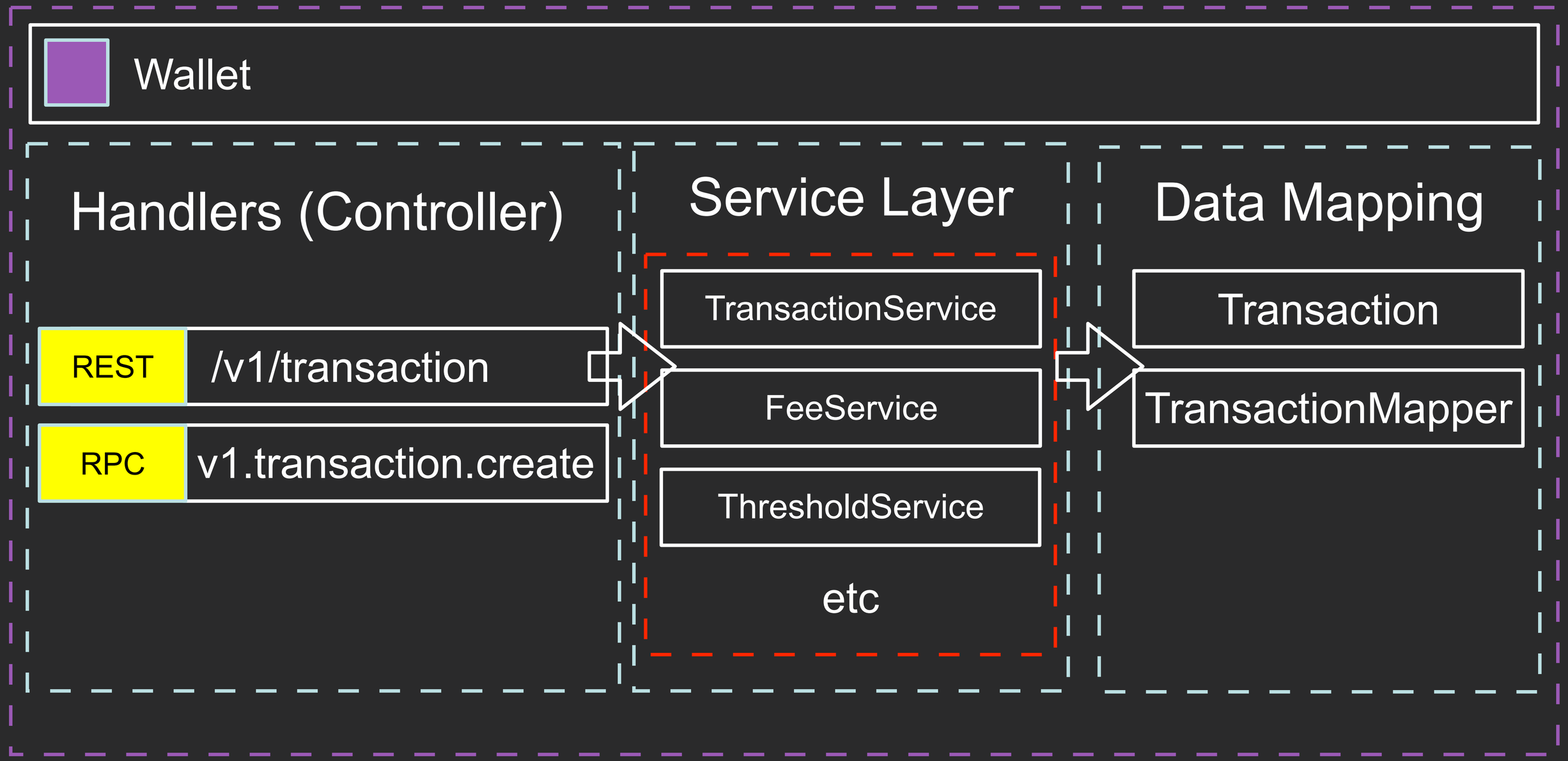
It can be more complex



It can be more complex



It can be more complex



Data Transfer Object (DTO)

A **data transfer object** (**DTO**[1][2]) is an object that carries data between processes. The motivation for its use is that communication between processes is usually done resorting to remote interfaces (e.g., web services), where each call is an expensive operation.

```
@Inject()
class HolyJSService {
    @Inject
    public readonly holyJSMapper: HolyJSMapper;

    public async create(payload: any, ctx: Context): Promise<any> {
        return await this.holyJSMapper.create(payload, ctx);
    }
}
```

Don’t forget, Node is async language

Handlers (Controller)

Request #1

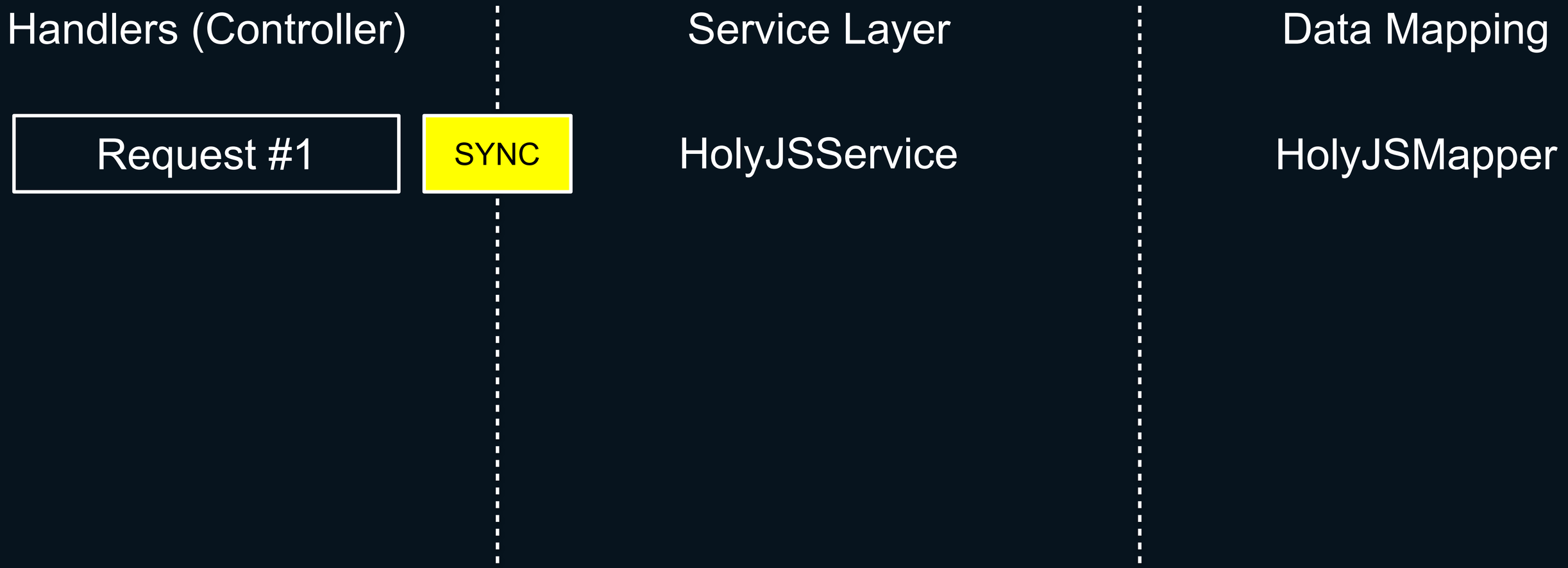
Service Layer

HolyJSService

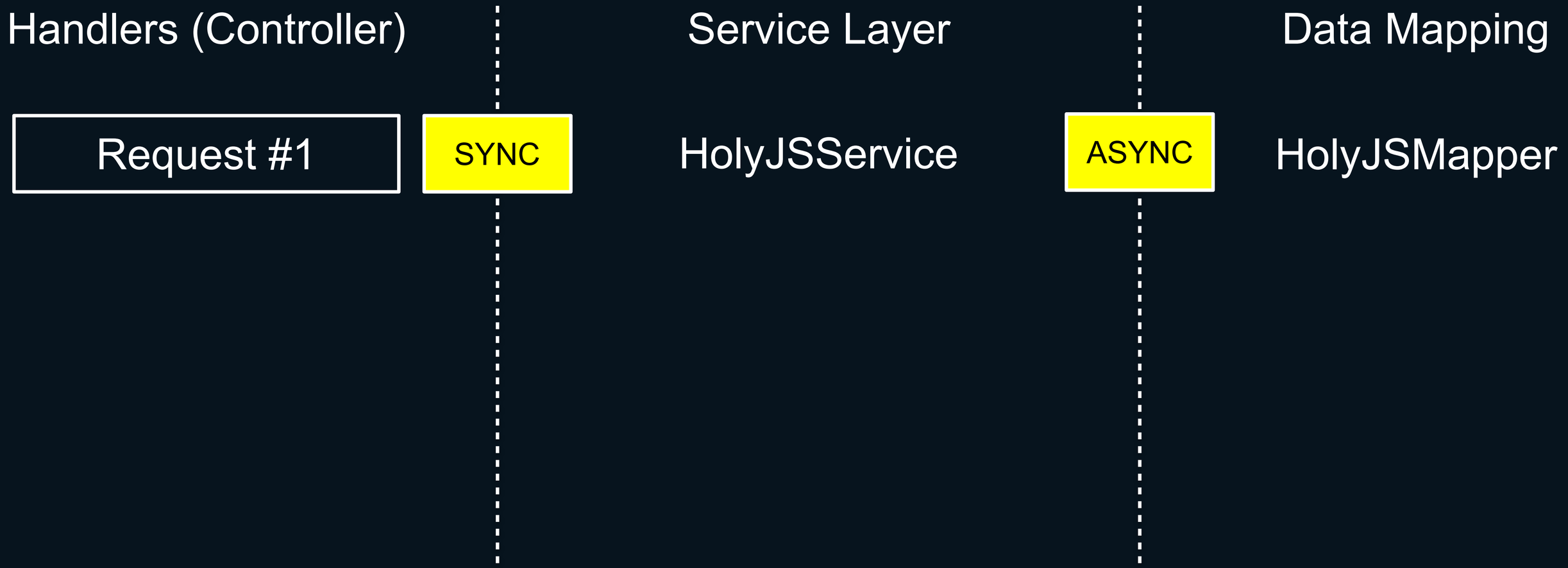
Data Mapping

HolyJSMapper

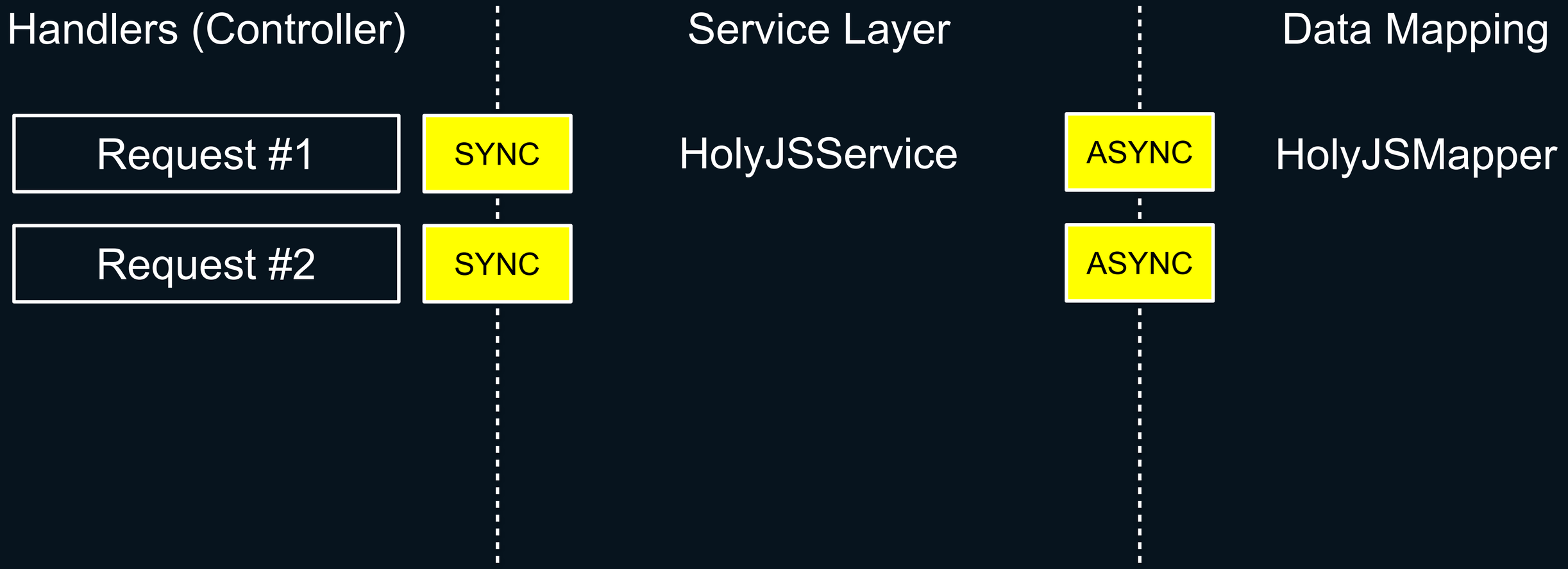
Don’t forget, Node is async language



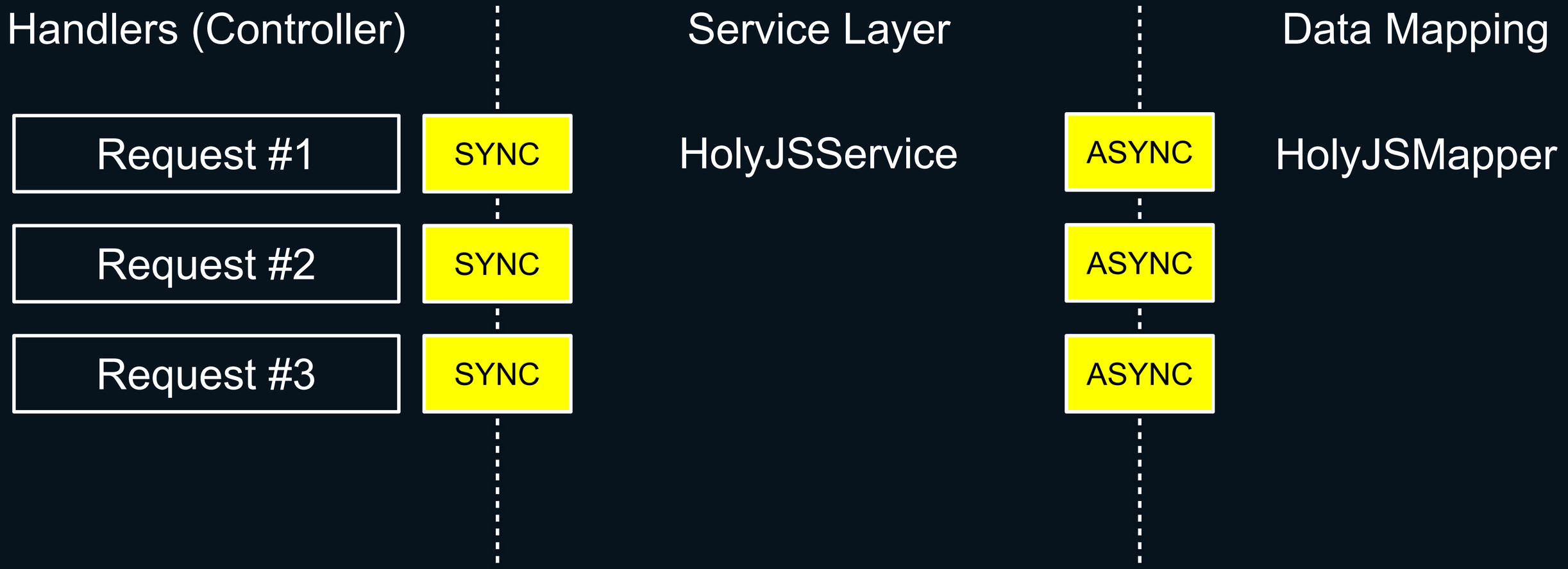
Don’t forget, Node is async language



Don’t forget, Node is async language



Don’t forget, Node is async language



How to Organize

How to Organize

Stateless

How to Organize

Stateless

Statefull

Statefull

Handlers (Controller)

Request #1

Service Layer

HolyJSService#1

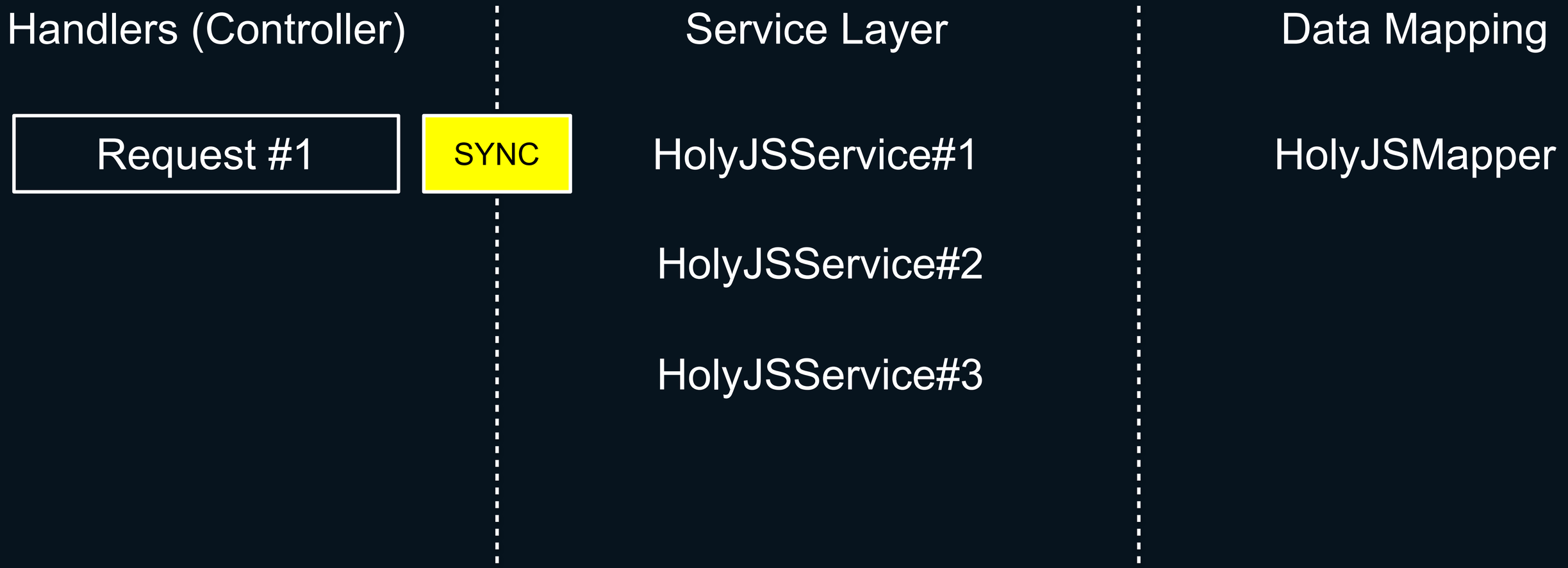
HolyJSService#2

HolyJSService#3

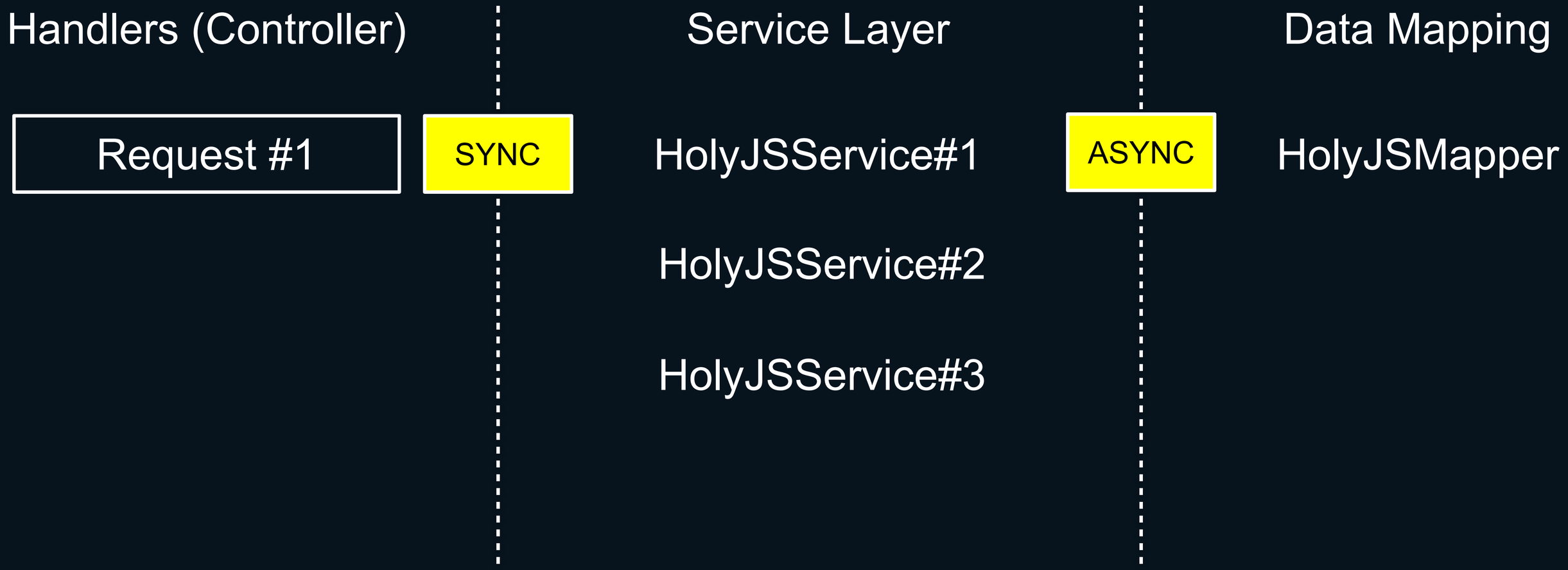
Data Mapping

HolyJSMapper

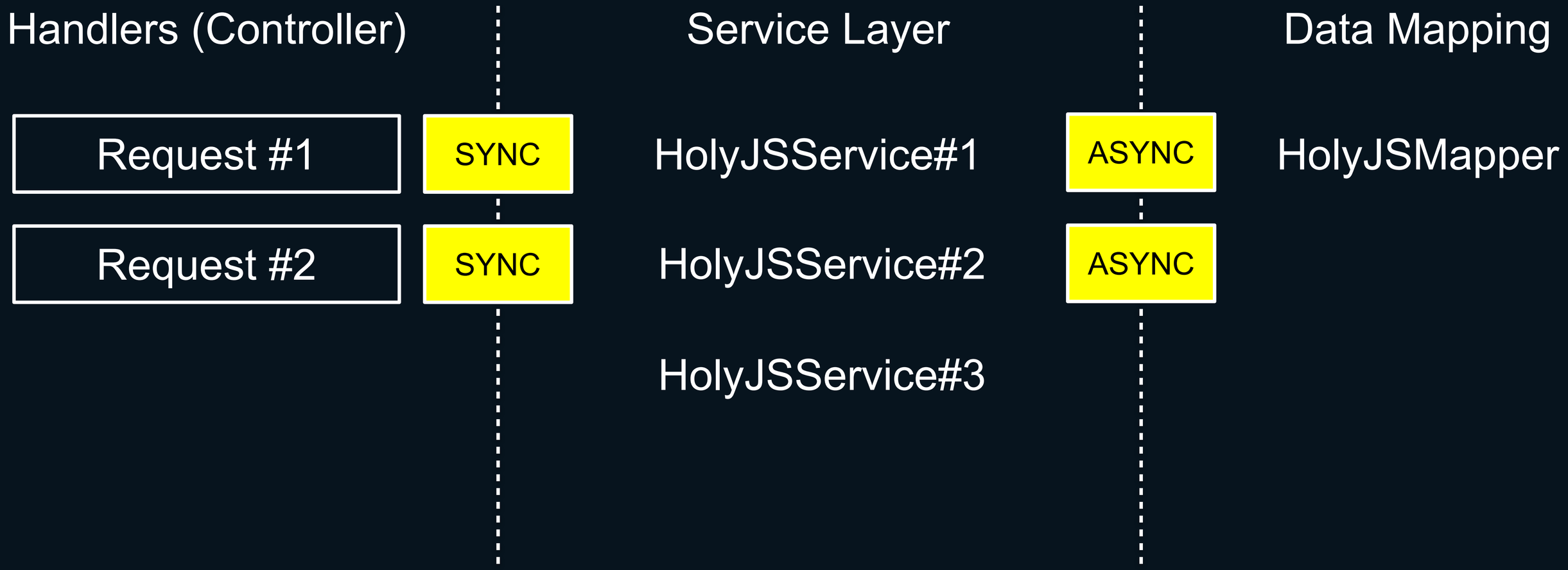
Statefull


Handlers (Controller)
Service Layer
Data Mapping
Request #1
SYNC
HolyJSService#1
HolyJSService#2
HolyJSService#3
HolyJSMapper

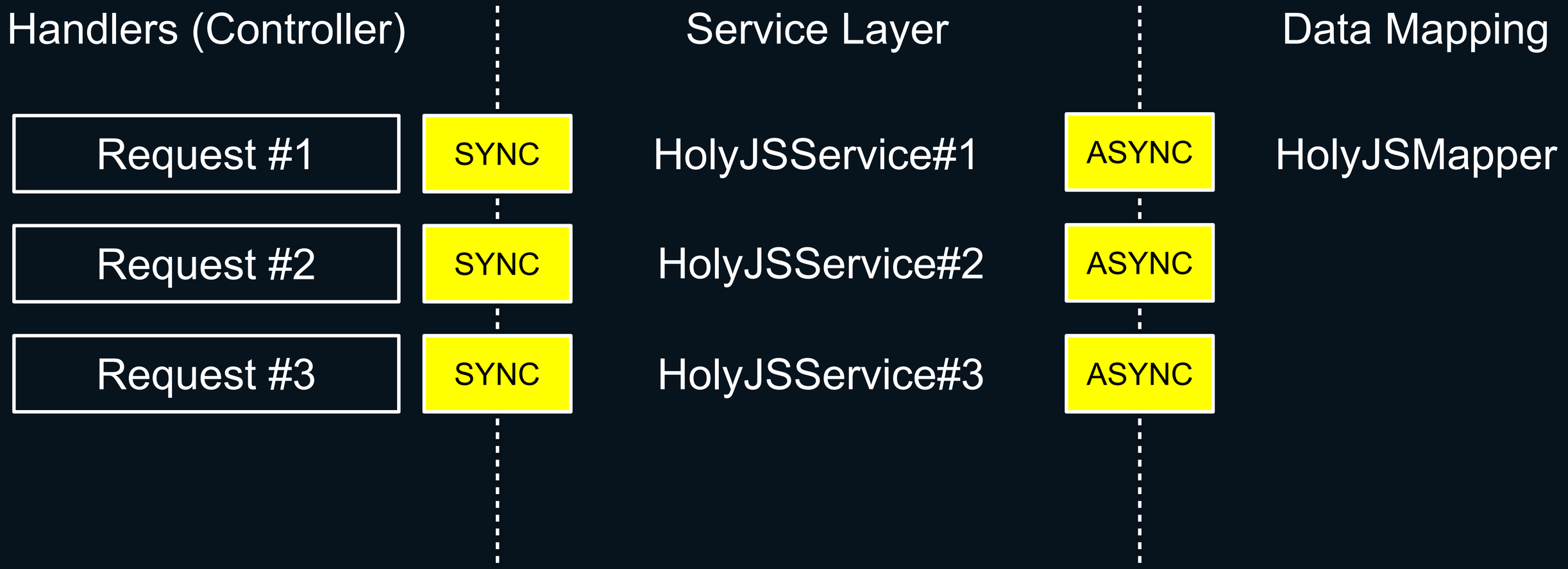
Statefull


Handlers (Controller)
Service Layer
Data Mapping
Request #1
SYNC
HolyJSService#1
HolyJSService#2
HolyJSService#3
ASYNC
HolyJSMapper

Statefull


Handlers (Controller)
Service Layer
Data Mapping
Request #1
Request #2
SYNC
SYNC
HolyJSService#1
HolyJSService#2
HolyJSService#3
ASYNC
ASYNC
HolyJSMapper

Statefull


Handlers (Controller)
Service Layer
Data Mapping
Request #1
Request #2
Request #3
SYNC
SYNC
SYNC
HolyJSService#1
HolyJSService#2
HolyJSService#3
ASYNC
ASYNC
ASYNC
HolyJSMapper

Statefull Service

Statefull Service

```
abstract class AbstractService {
    protected requestId: string;
    protected userId: string;
    protected userAgent: string;

    public setRequestId(requestId: string) { this.requestId = requestId; }
    public setUserId(userId: string) { this.userId = userId; }
    public setUserAgent(userAgent: string) { this.userAgent = userAgent; }
}

@Inject()
class HolyJSService extends AbstractService {
    @Inject
    public readonly holyJSMapper: HolyJSMapper;

    public async create(payload: any): Promise<any> {
        return await this.holyJSMapper.create(payload);
    }
}
```

Statefull Service

Statefull Service

```
export const create = httpHandler(async (req: Request, res: Response) => {
    const holyjsService: HolyJSService = Container.get(HolyJSService);
    populateServiceFromRequest(holyjsService, Request);

    res.status(201).json(
        await holyjsService.create(req.body, createContextFromHttpRequest(req))
    );
});
```

Stateless

Handlers (Controller)

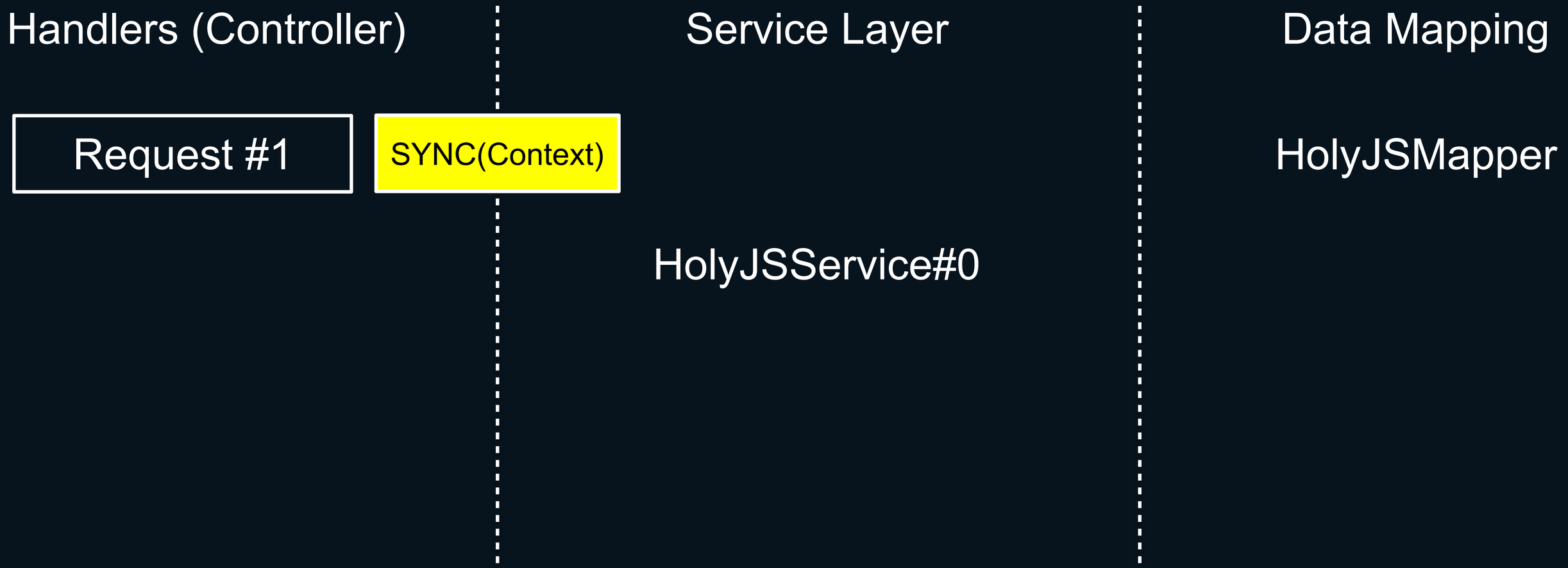
Service Layer

HolyJSService#0

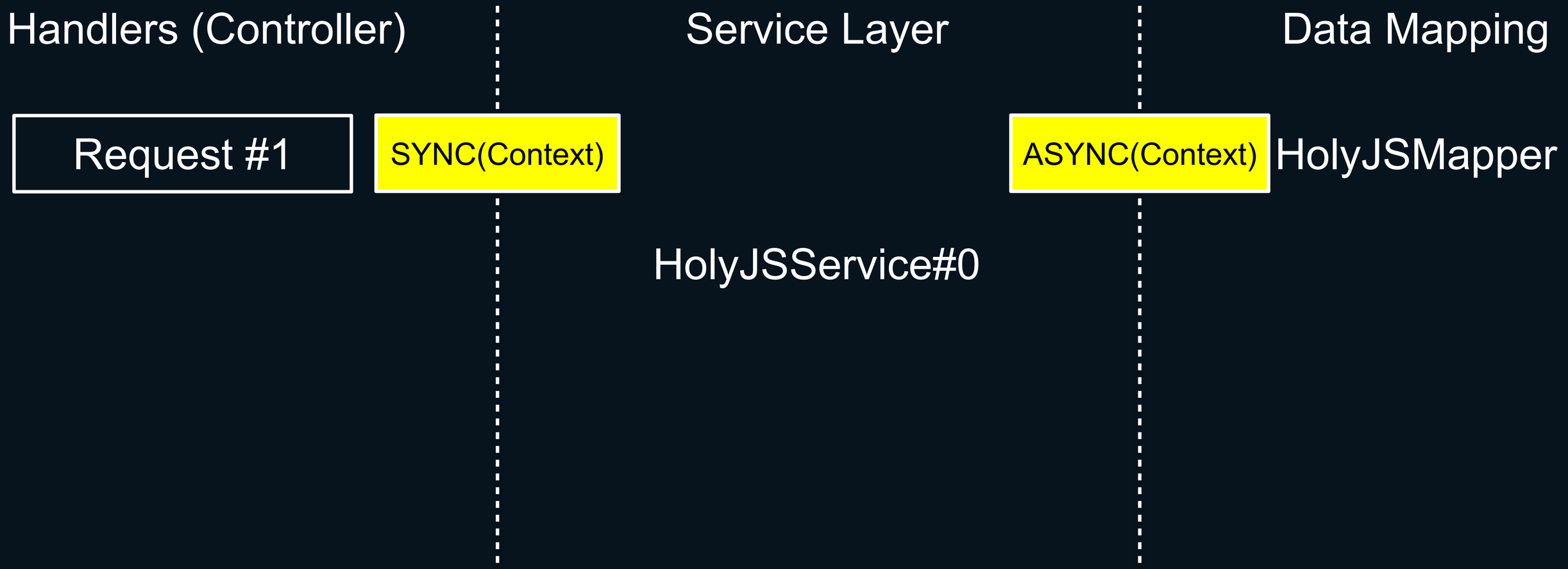
Data Mapping

HolyJSMapper

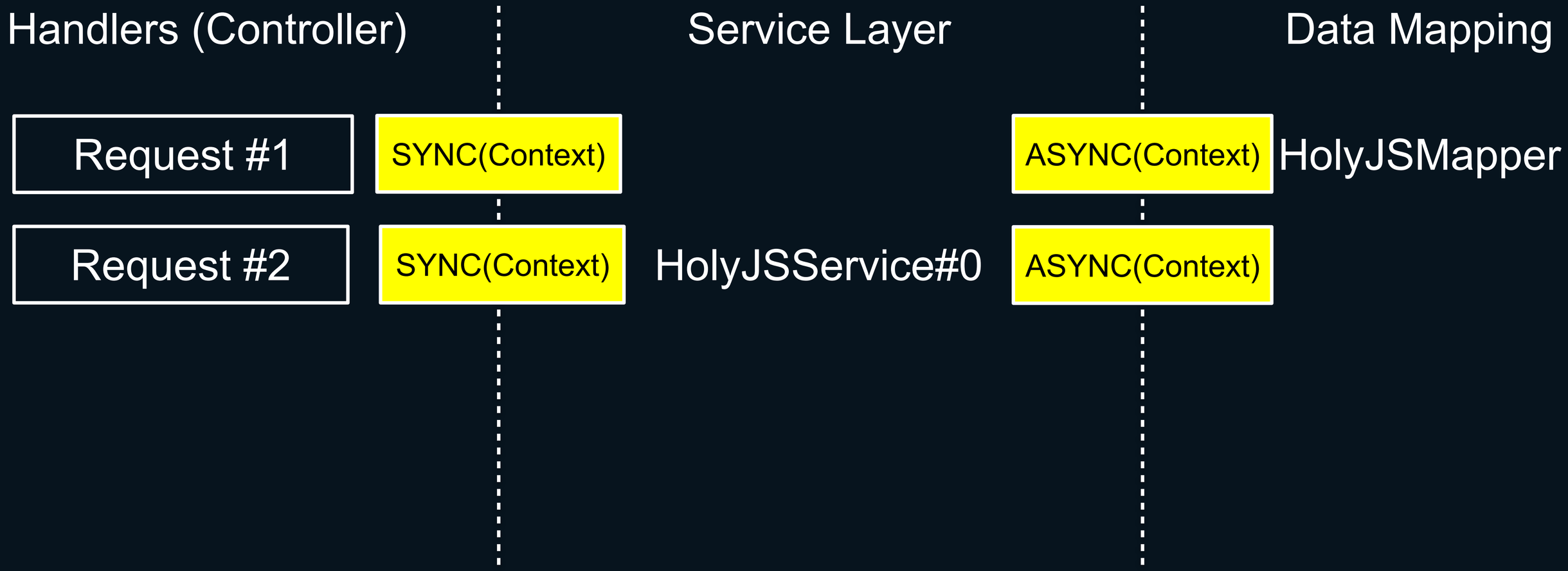
Stateless


Handlers (Controller)
Service Layer
Data Mapping
Request #1
SYNC(Context)
HolyJSMapper
HolyJSService#0

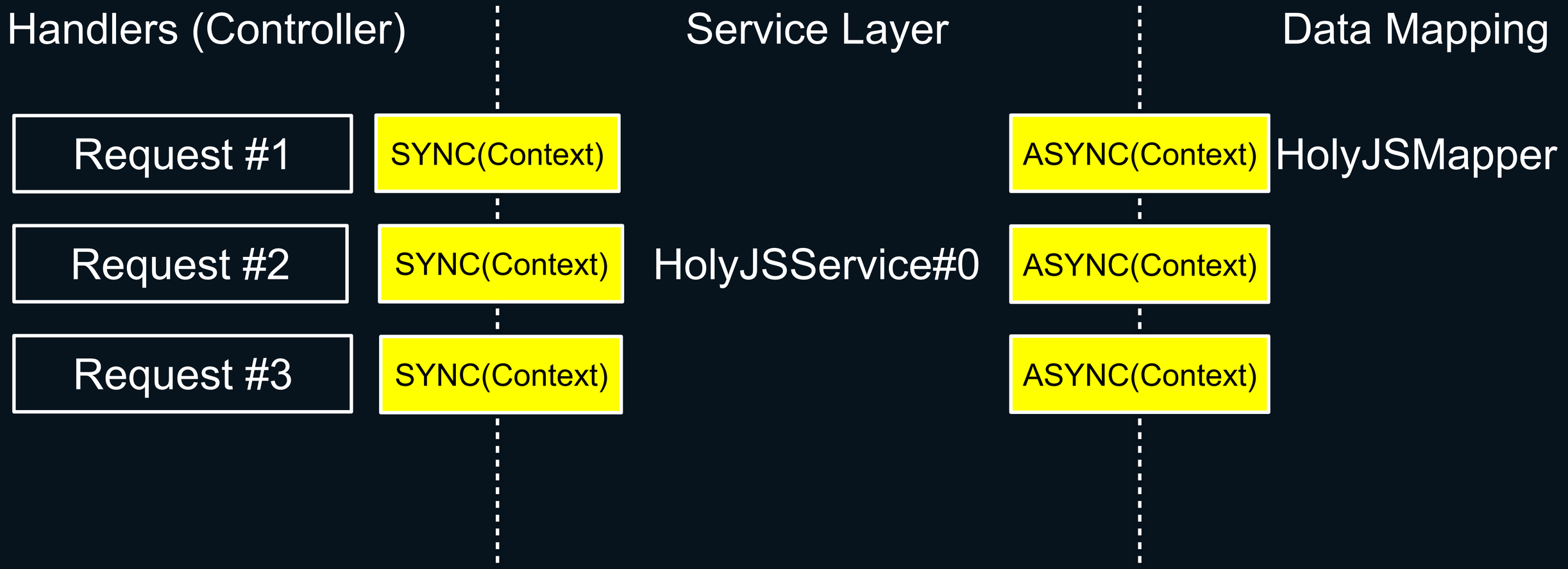
Stateless


Handlers (Controller)
Service Layer
Data Mapping
Request #1
SYNC(Context)
ASYNC(Context)
HolyJSMapper
HolyJSService#0

Stateless


Handlers (Controller)
Service Layer
Data Mapping
Request #1
SYNC(Context)
ASYNC(Context)
HolyJSMapper
Request #2
SYNC(Context)
HolyJSService#0
ASYNC(Context)

Stateless


Handlers (Controller)
Service Layer
Data Mapping
Request #1
Request #2
Request #3
SYNC(Context)
SYNC(Context)
SYNC(Context)
HolyJSService#0
ASYNC(Context)
ASYNC(Context)
ASYNC(Context)
HolyJSMapper

Context

Context

```
type Context = {
    userAgent: string;
    requestId: string;
    currentUserId: string;
};

export const createContext = (req: Request): Context => ({
    userAgent: <string>req.headers['user-agent'],
    requestId: <string>req.headers['X-Request-Id'],
    currentUserId: <string>req.headers['X-Current-User-Id']
});
```

Stateless Service

```
@Inject()
class HolyJSService {
    @Inject
    public readonly holyJSMapper: HolyJSMapper;

    public async create(payload: any, ctx : Context): Promise<any> {
        return await this.holyJSMapper.create(payload, ctx );
    }
}
```

Usage of stateless Service

```
const holyjsService: HolyJSService = Container.get(HolyJSService);

export const create = httpHandler(async (req: Request, res: Response) => {
    res.status(201).json(
        await holyjsService.create(req.body, createContextFromHttpRequest(req))
    );
});
```

REST + AMQP

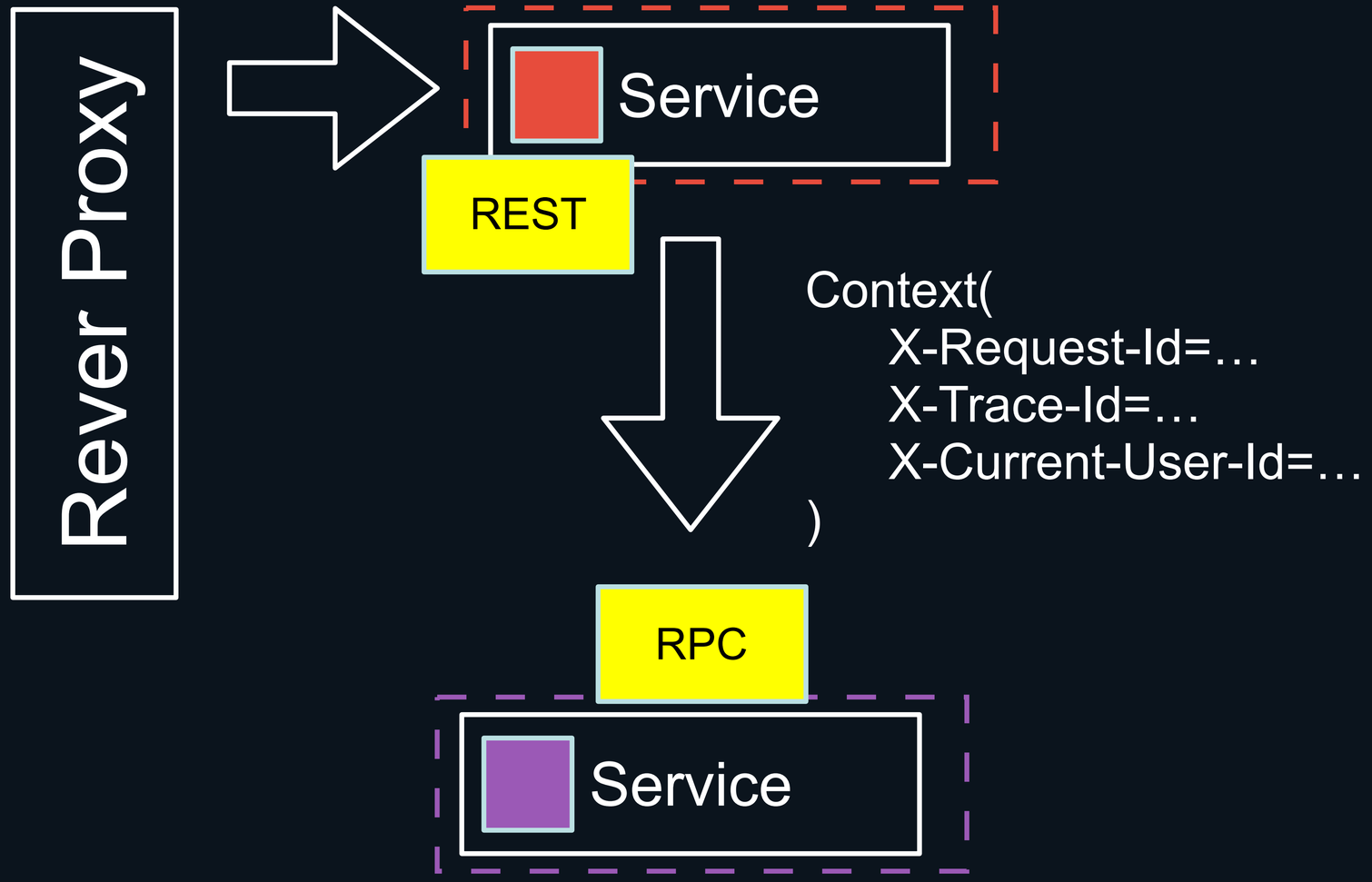
REST

```
# transaction.amqp.handler.ts

const holyjsService: HolyJSService = Container.get(HolyJSService);

export const create = amqpHandler(async (req: Request, res: Response) => {
    res.status(201).json(
        await holyjsService.create(req.body, createContextFromHttpRequest(req))
    );
});
```

AMQP

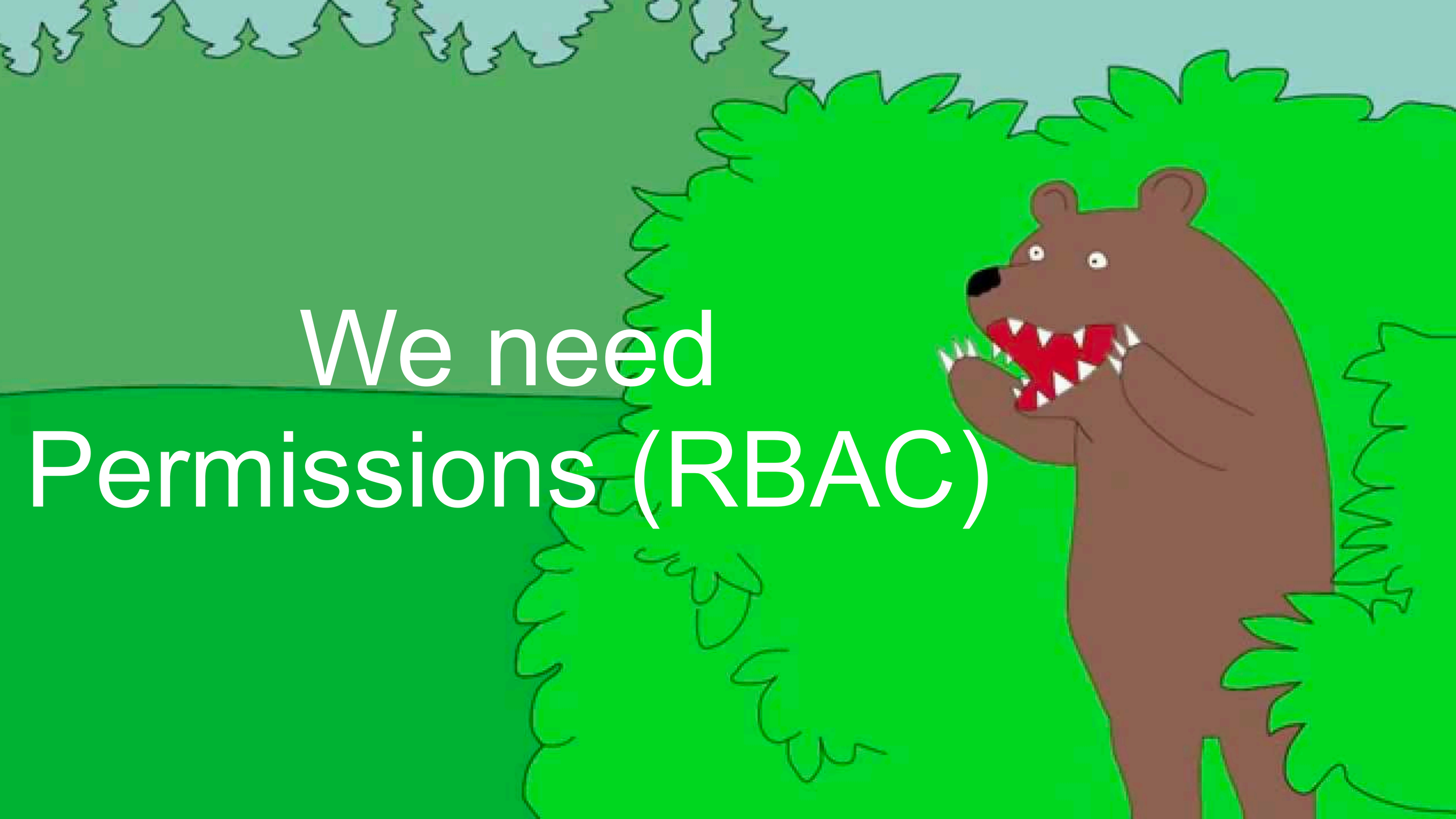
```
# transaction.rest.handler.ts

const holyjsService: HolyJSService = Container.get(HolyJSService);

export const create = amqpHandler(async (req: AMQPRequest, res: AMQPResponse) => {
   return await holyjsService.create(req.body, createContextFromAmqpRequest(req))
});
```

Context Handling



OPENTRACING

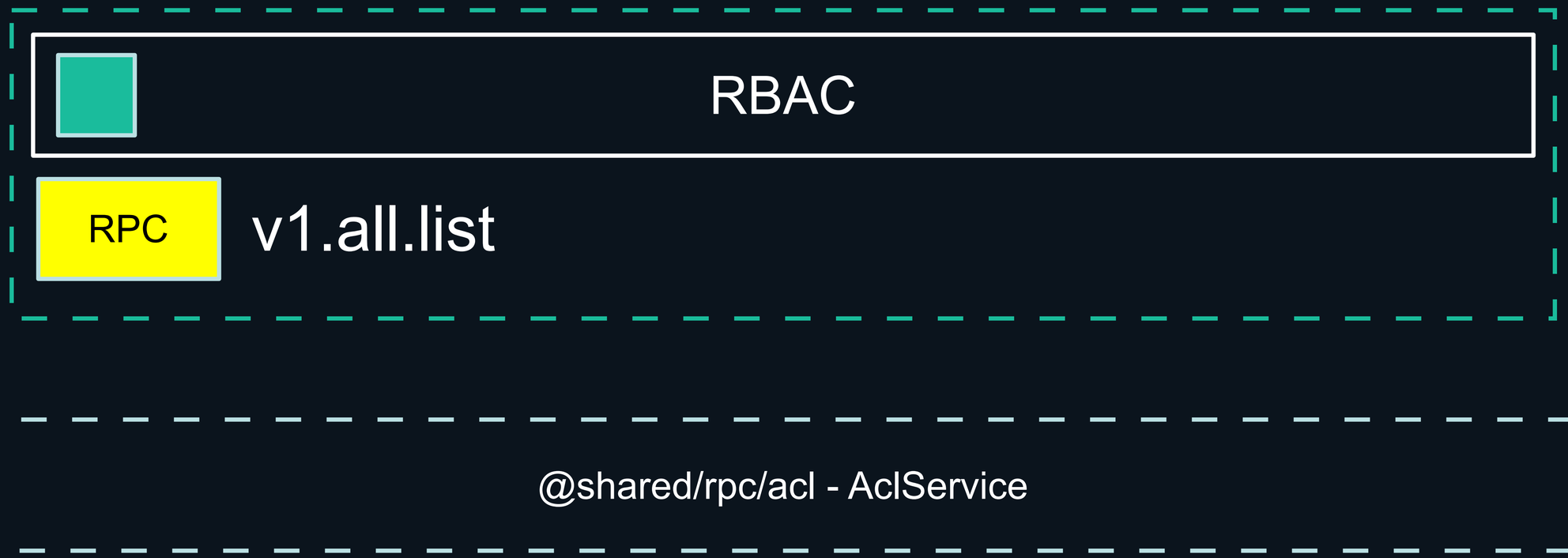
MicroService + Service

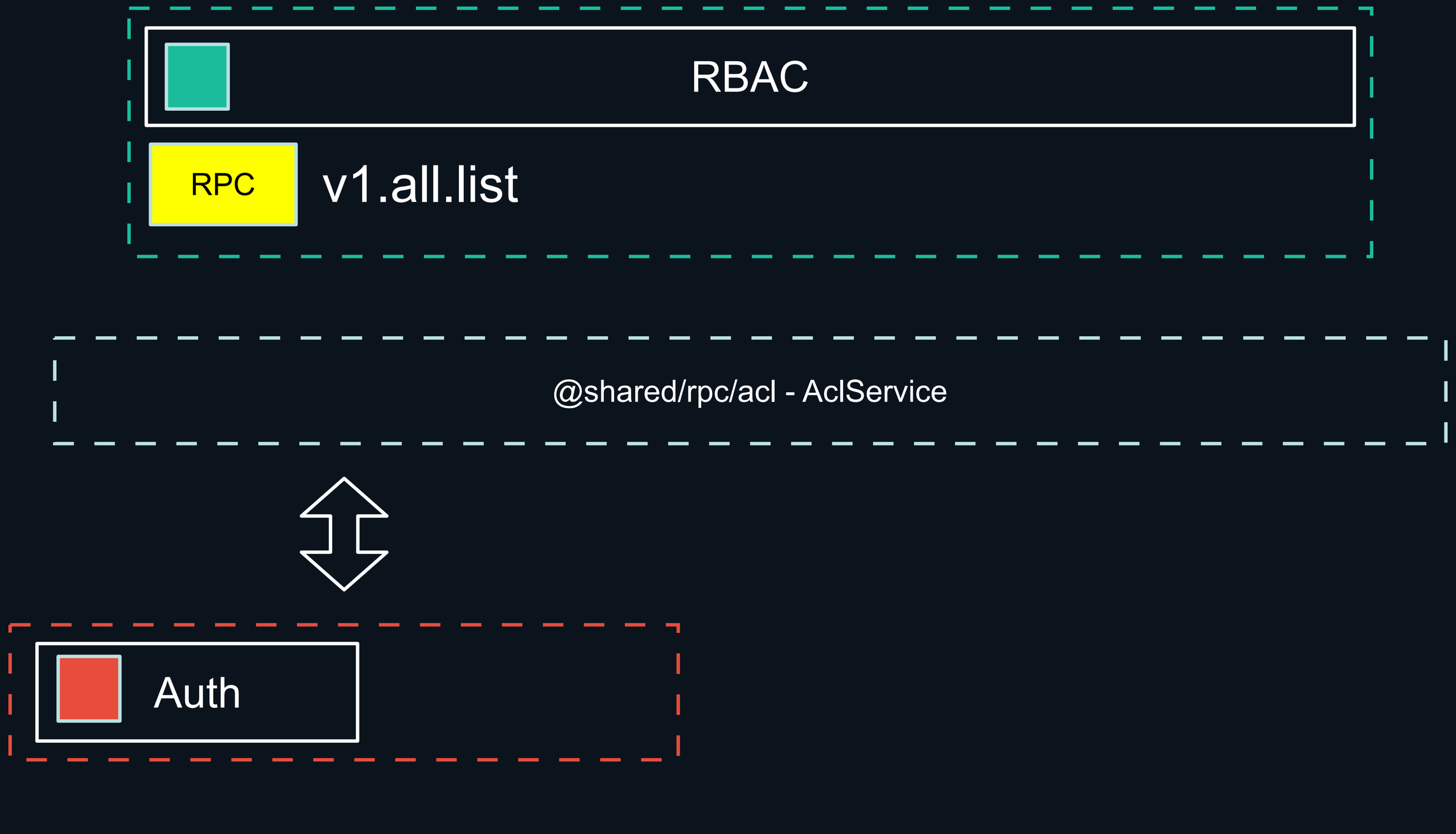

We need
Permissions (RBAC)

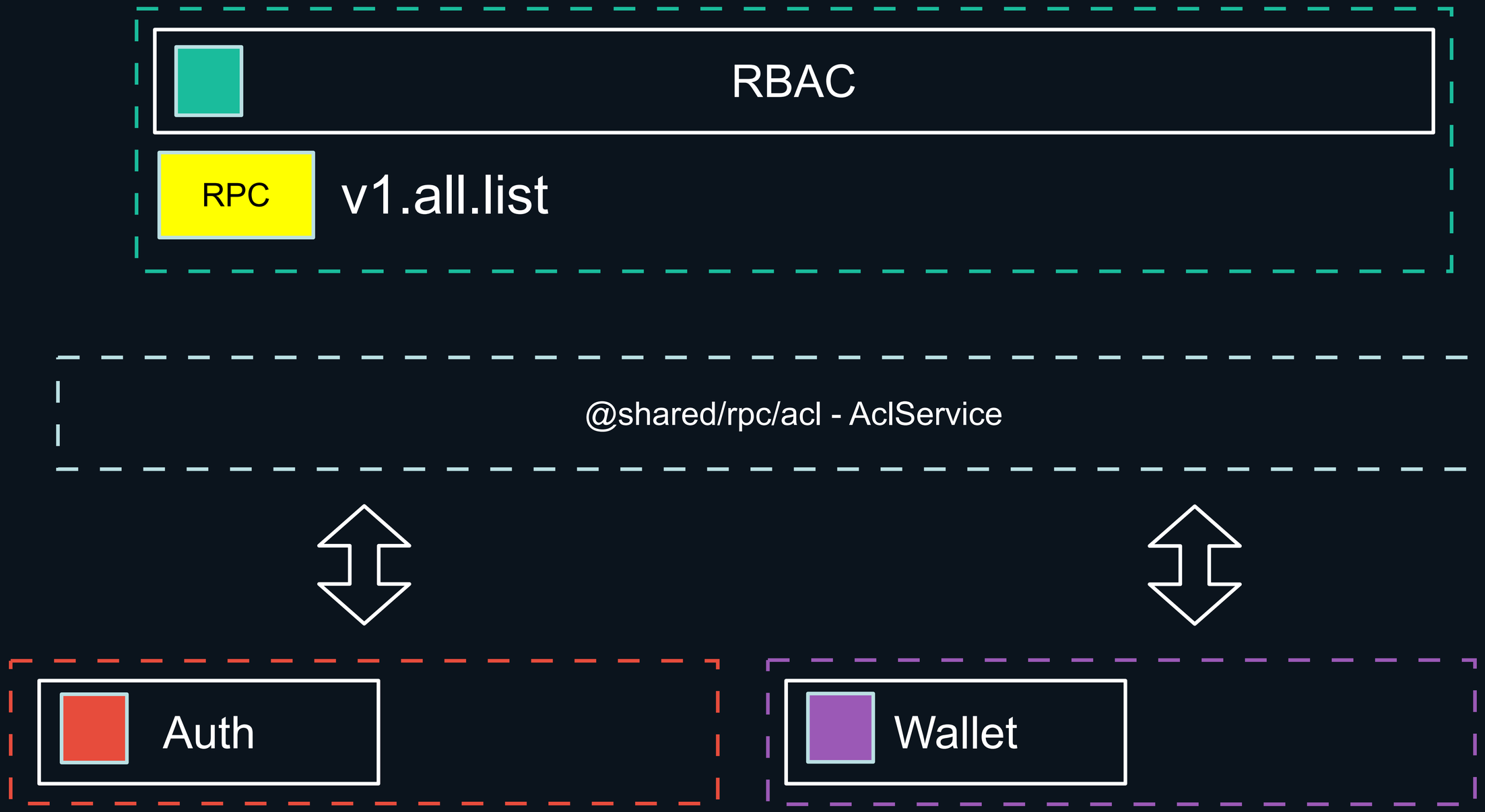

RBAC
Role
Resource
Rule


RBAC

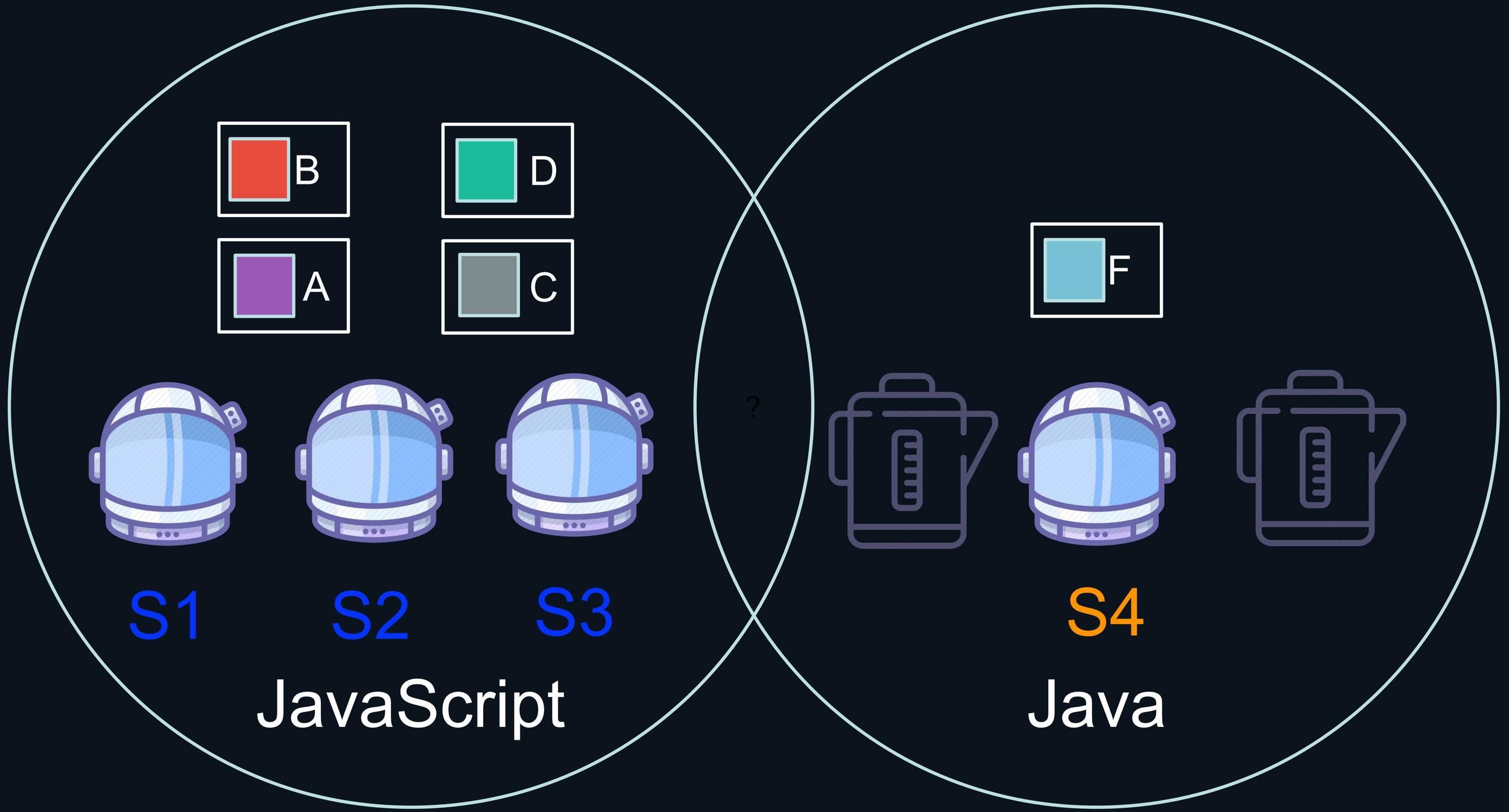

RBAC
RPC
v1.all.list


RBAC
RPC
v1.all.list
@shared/rpc/acl - AclService

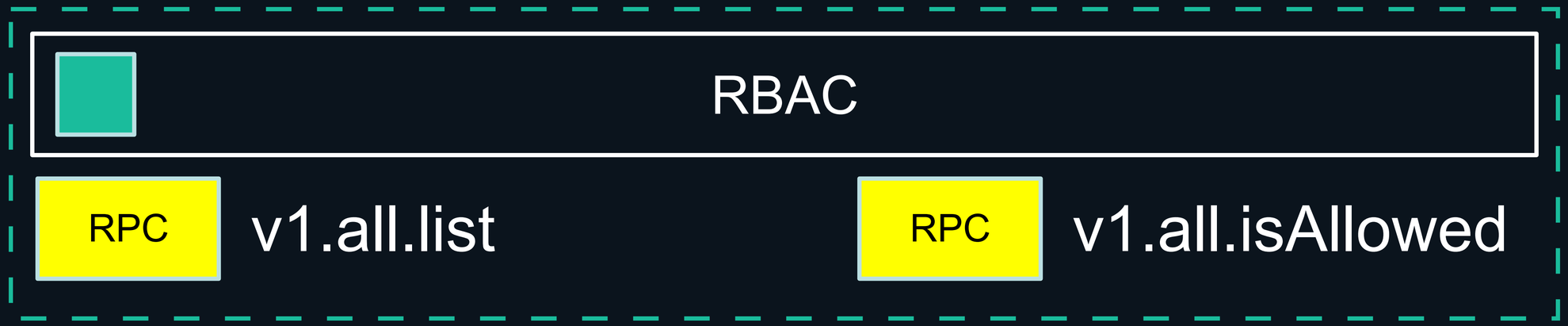

RBAC
RPC
v1.all.list
@shared/rpc/acl - AclService
Auth

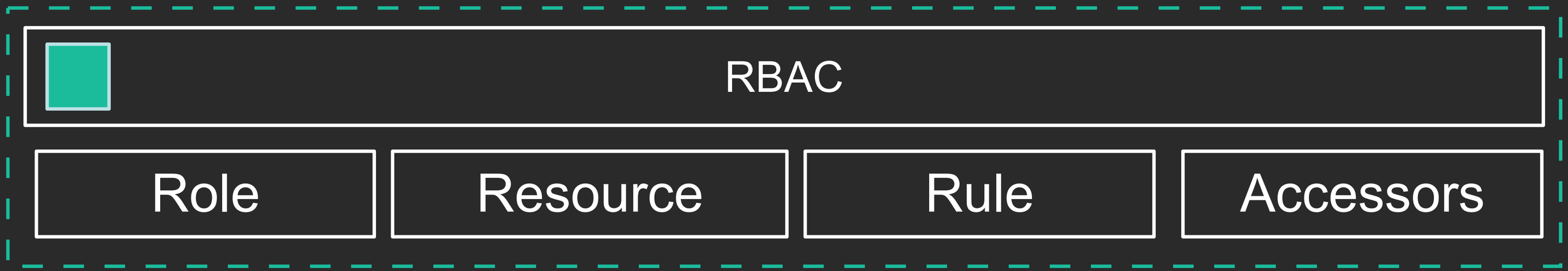

RBAC
RPC
v1.all.list
@shared/rpc/acl - AclService
Auth
Wallet


```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
}
```

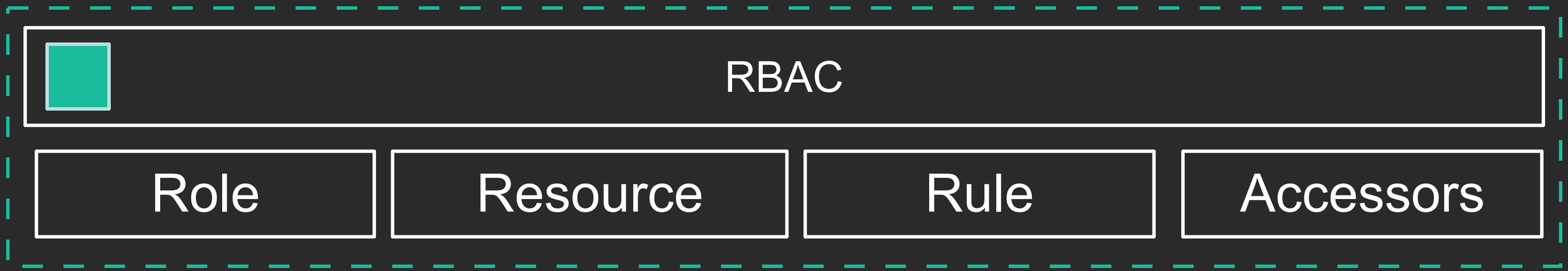

B
D
A
C
F
?
S1
S2
S3
S4
JavaScript
Java


RBAC
RPC
v1.all.list


RBAC
RPC
v1.all.list
RPC
v1.all.isAllowed

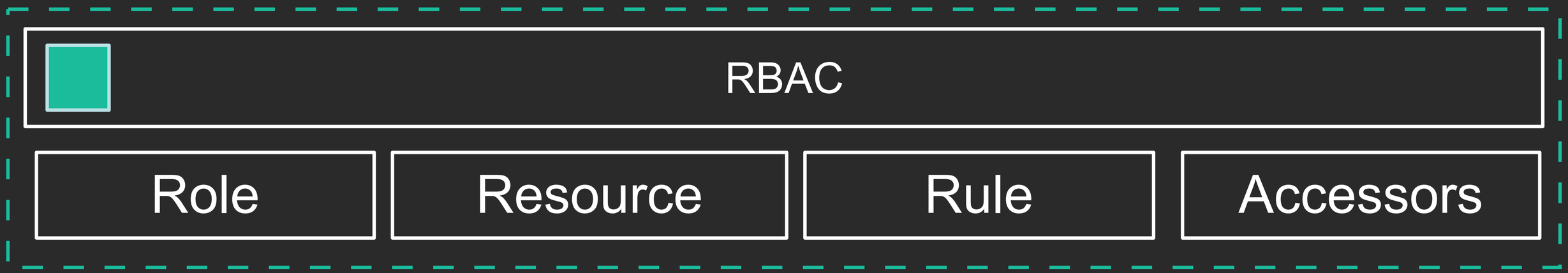



```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
    filter(ctx: Context, query: any, resource: string, action: string);
}
```




```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
    filter(ctx: Context, query: any, resource: string, action: string);
}
```

But we cannot use RPC for filter 😭




```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
    filter(ctx: Context, query: any, resource: string, action: string);
    condition(ctx: Context, query: any, resource: string, action: string);
}
```


```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
    filter(ctx: Context, query: any, resource: string, action: string);
    condition(ctx: Context, query: any, resource: string, action: string);
}
```

But we cannot use RPC for filter 😭


```
interface RBACServiceInterface {
    isAllowed(ctx: Context, resource: string, action?: string);
    filter(ctx: Context, query: any, resource: string, action: string);
    condition(ctx: Context, query: any, resource: string, action: string);
}
```

But we cannot use RPC for filter 😭

But we cannot use RPC for condition 😭

We are using GitLab

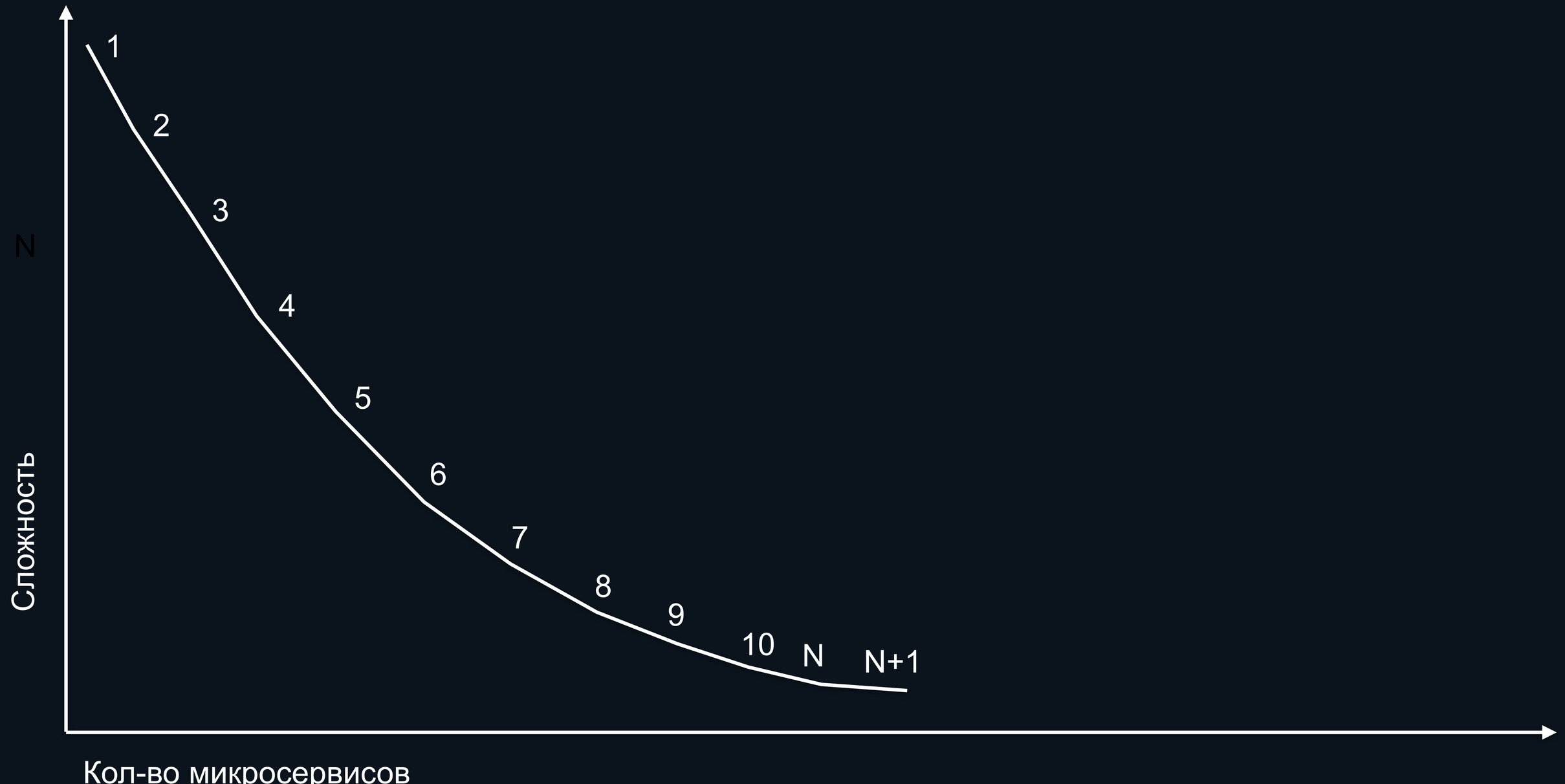
GitLab

details in discussion zone

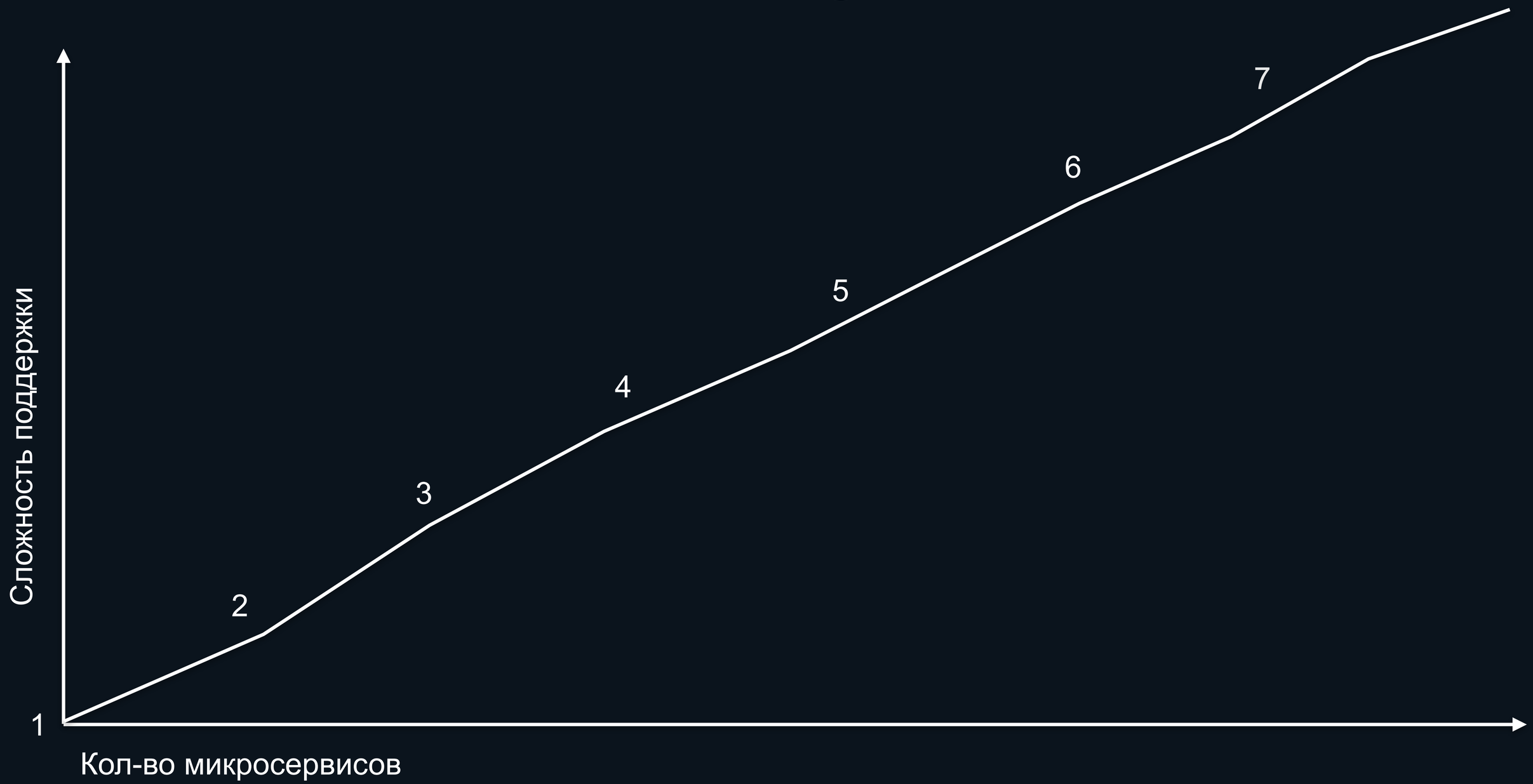
Delusion

#1 More easy

New service creation complexity



Support Complexity



#2 Better performance

Use microservices?

Use microservices?

positive > negative

Вопросы?)

https://github.com/ovr

talk@dmtry.me

https://telegram.me/ovrweb

Monolith

Microservice