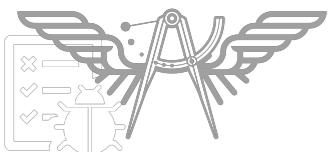


Microservices





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...
Timeline About Friends 3,138 Photos More

When did you work at Opendream?
... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro
Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

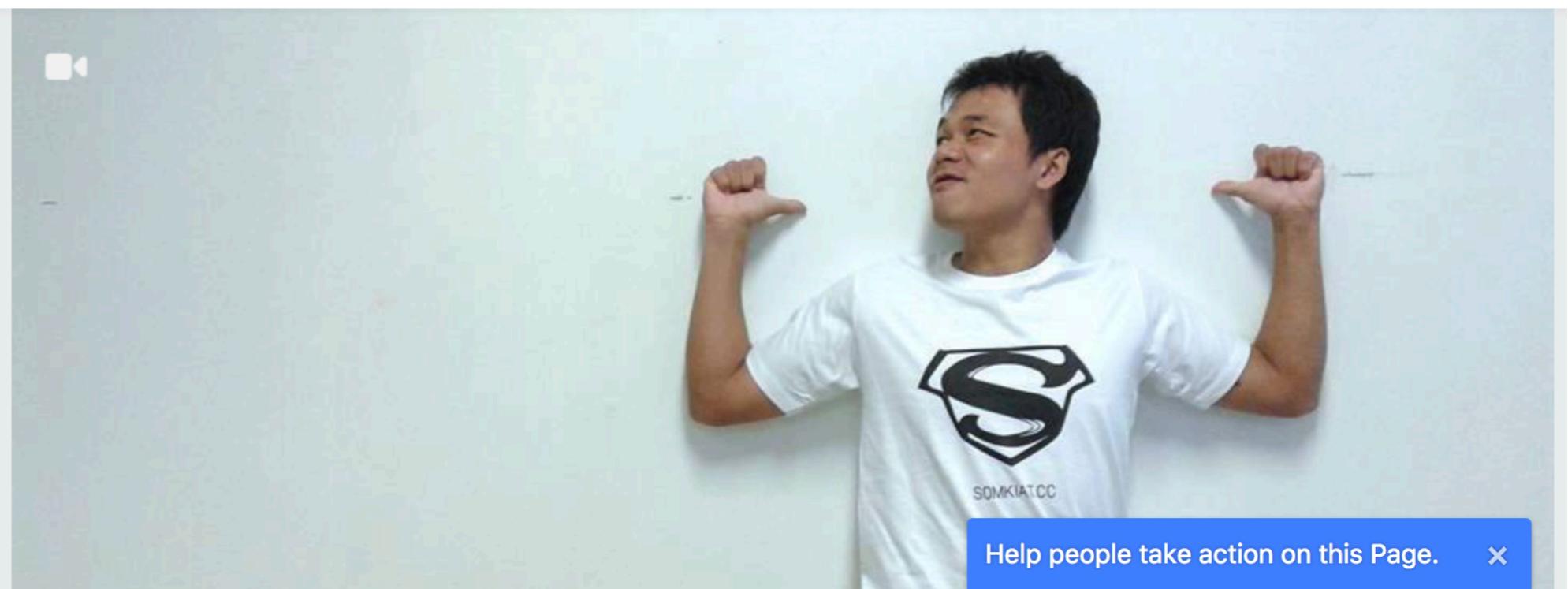
@somkiat.cc

Home

Posts

Videos

Photos



Module 1 : Design

Cloud Native Application
Evolution of architecture
Microservice architecture
How to decompose app to Microservice
Communication between service
Data consistency
Workshop



Module 2 : Develop

Recap Microservice

Properties of Microservice

Microservice 1.0 - 4.0

How to develop Microservice ?

How to test Microservice ?

12-factors app

Workshop



Module 3 : Deploy

How to deploy Microservice ?
Continuous Integration and Delivery
Practices of Continuous Integration
Deployment strategies
Working with containerization (Docker)
Workshop



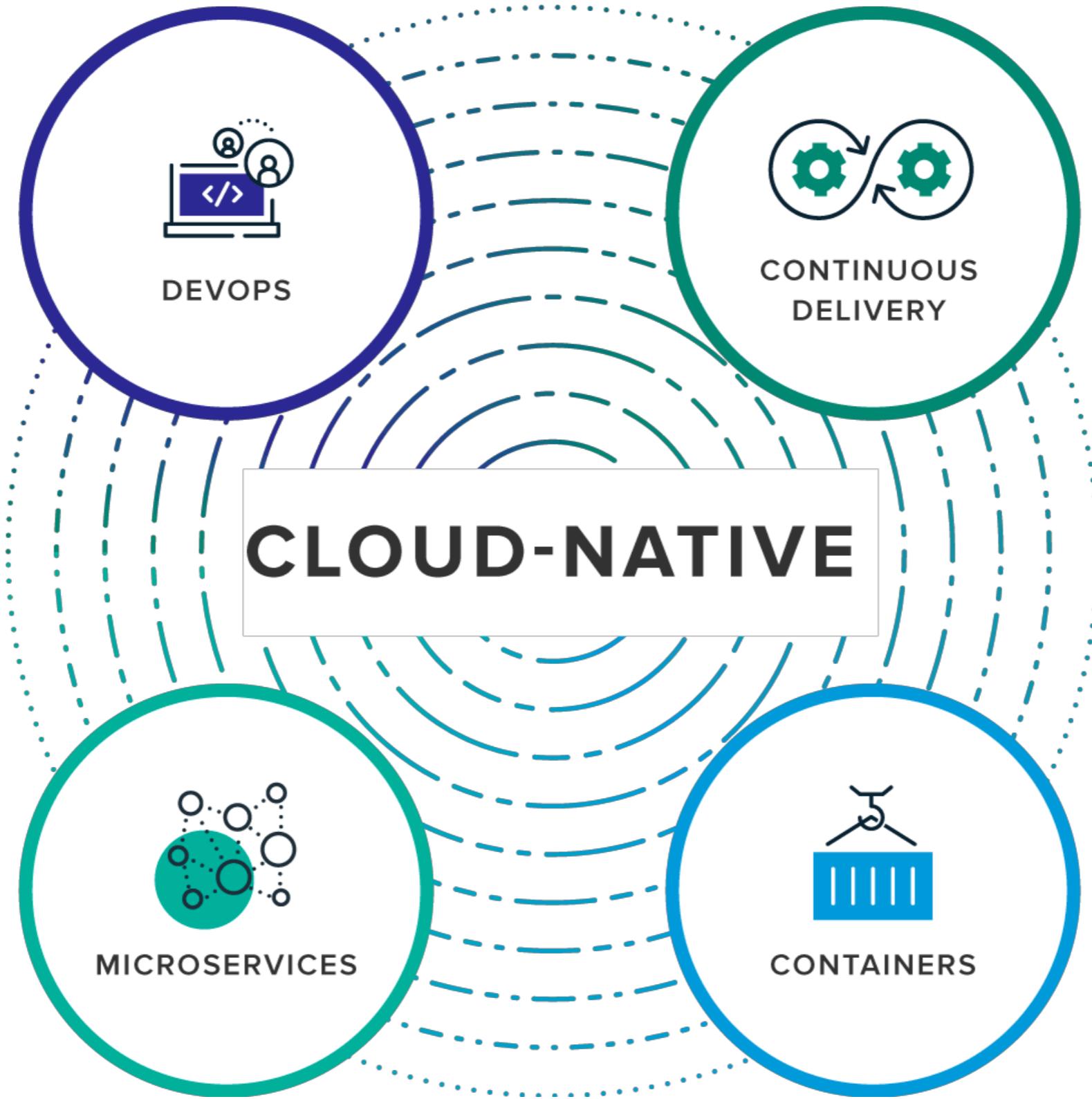
Module 1 : Design



"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"

- *Melvin Conwey, 1968* -





<https://pivotal.io/cloud-native>



Evolution of Architecture

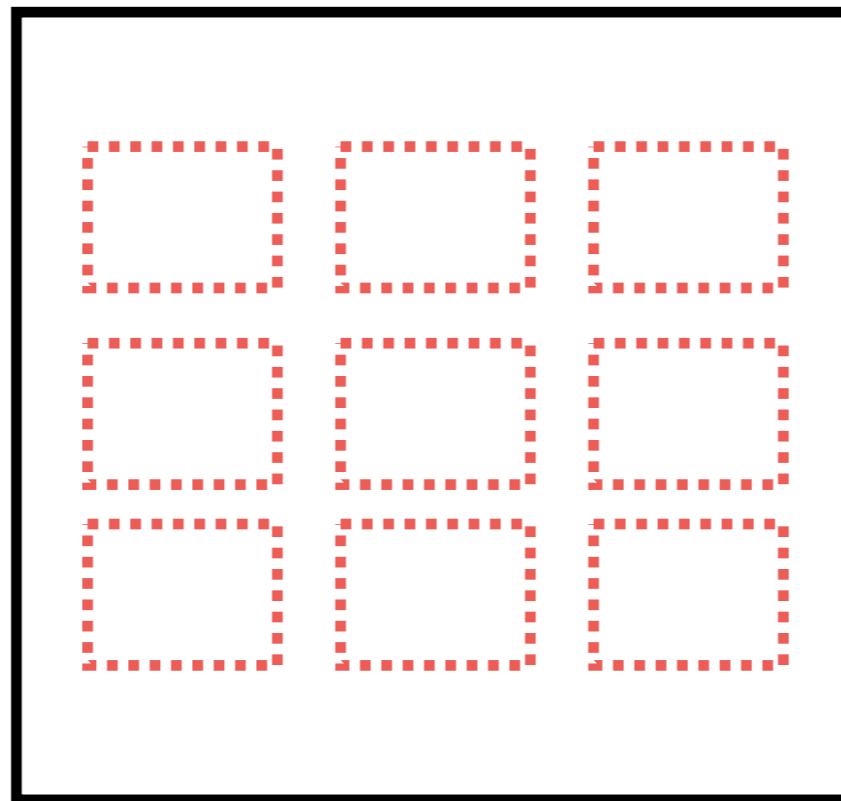


Monolith

App



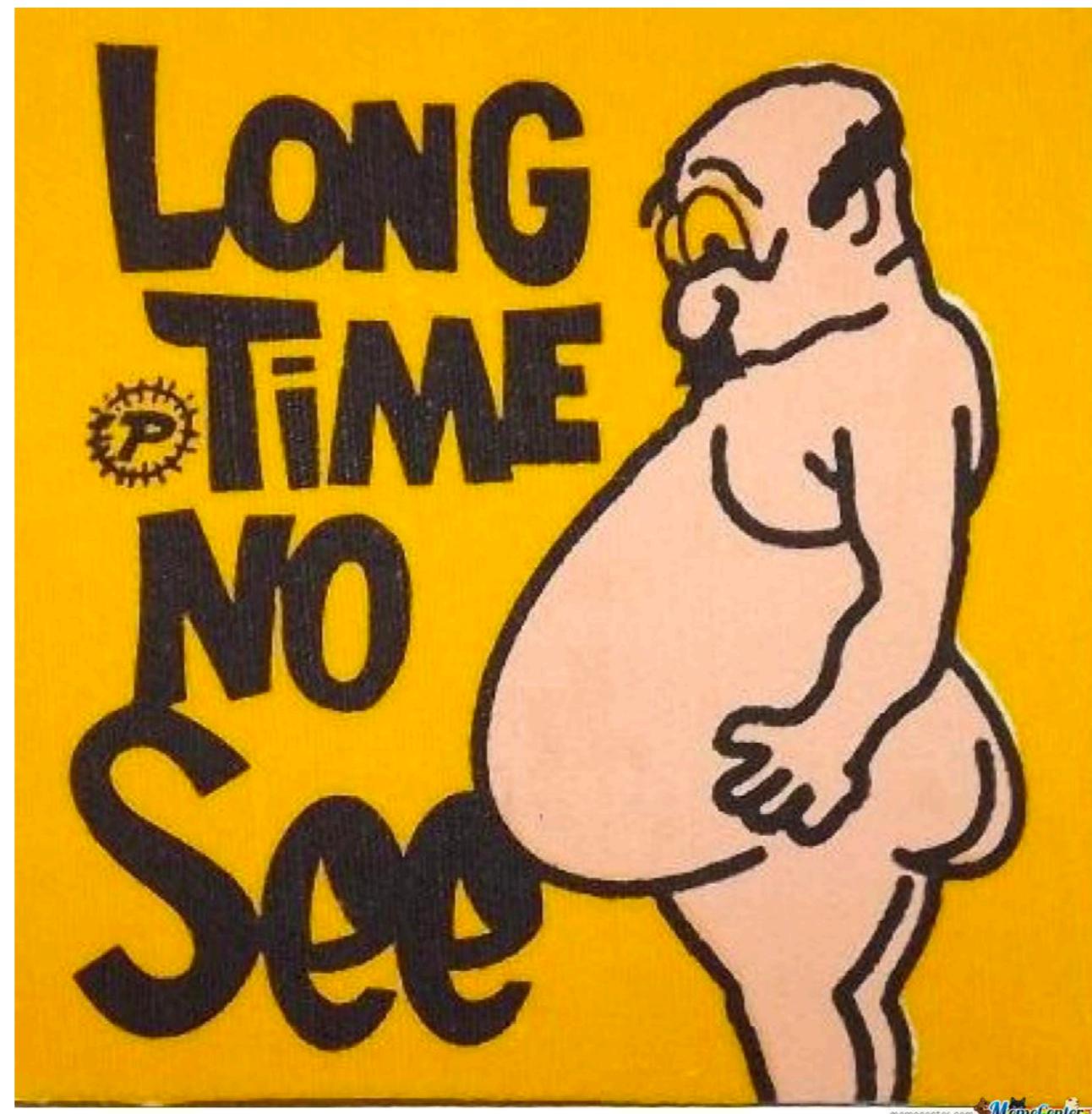
Modules



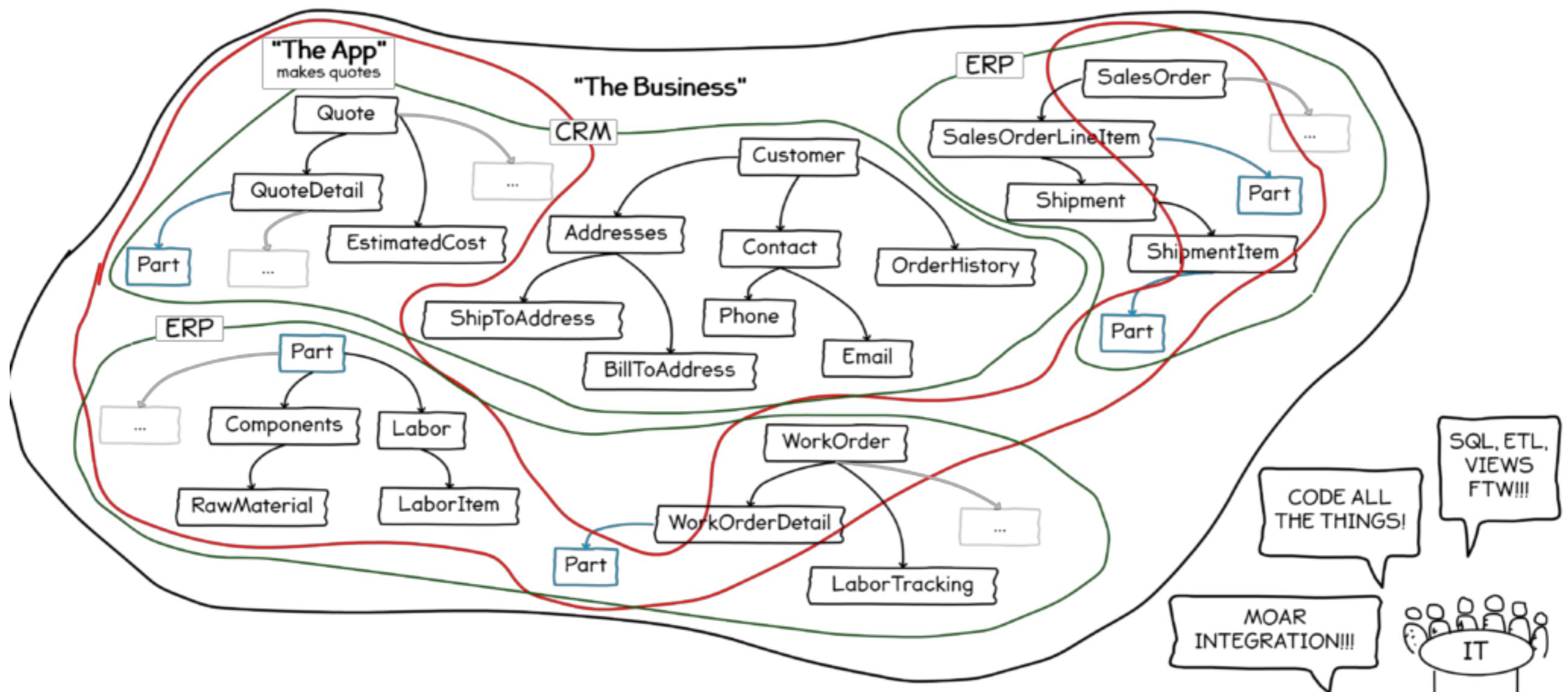
Monolith

Easy to develop
Easy to change
Easy to testing
Easy to deploy
Easy to scale



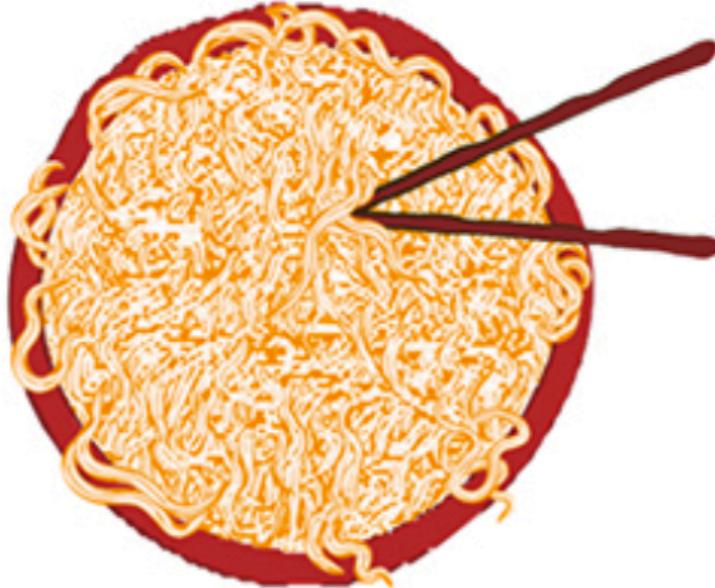


Monolith Hell



1990s and earlier

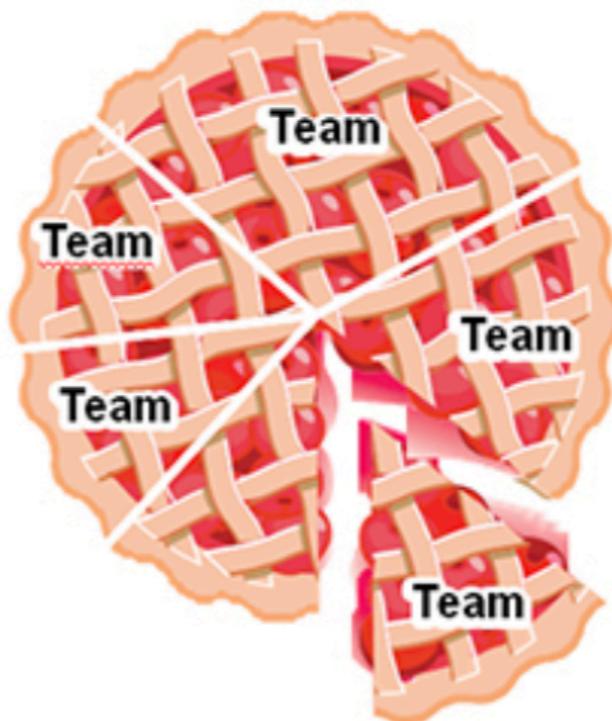
Pre-SOA (monolithic)
Tight coupling



For a monolith to change, all must agree on each change. Each change has unanticipated effects requiring careful testing beforehand.

2000s

Traditional SOA
Looser coupling



Elements in SOA are developed more autonomously but must be coordinated with others to fit into the overall design.

2010s

Microservices
Decoupled



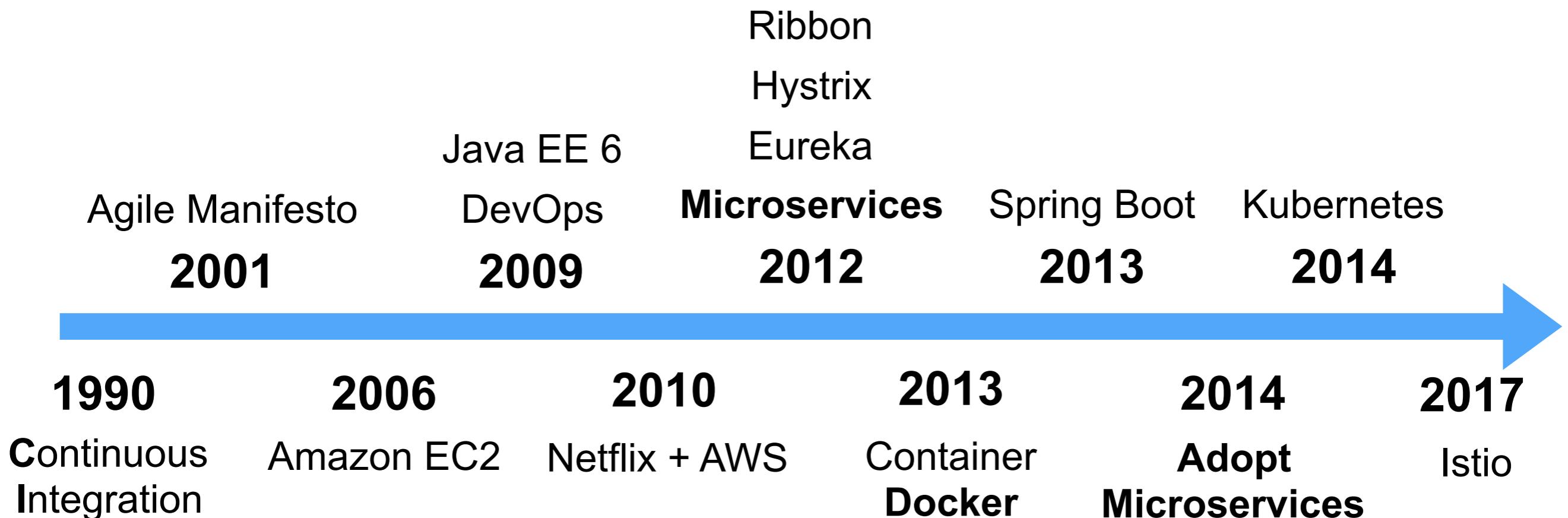
Developers can create and activate new microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new or modified services possible.



Microservice



Microservice history



Microservice

Small, Do one thing (Single Responsibility)

Modular

Easy to understand

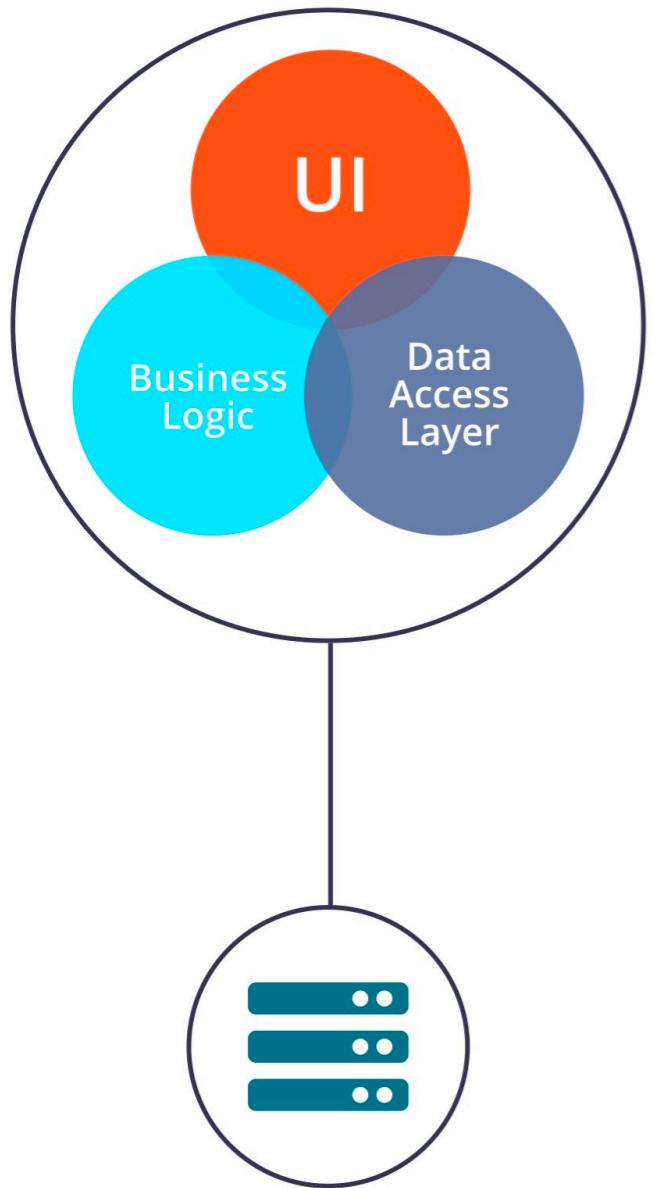
Easy to develop

Easy to deploy

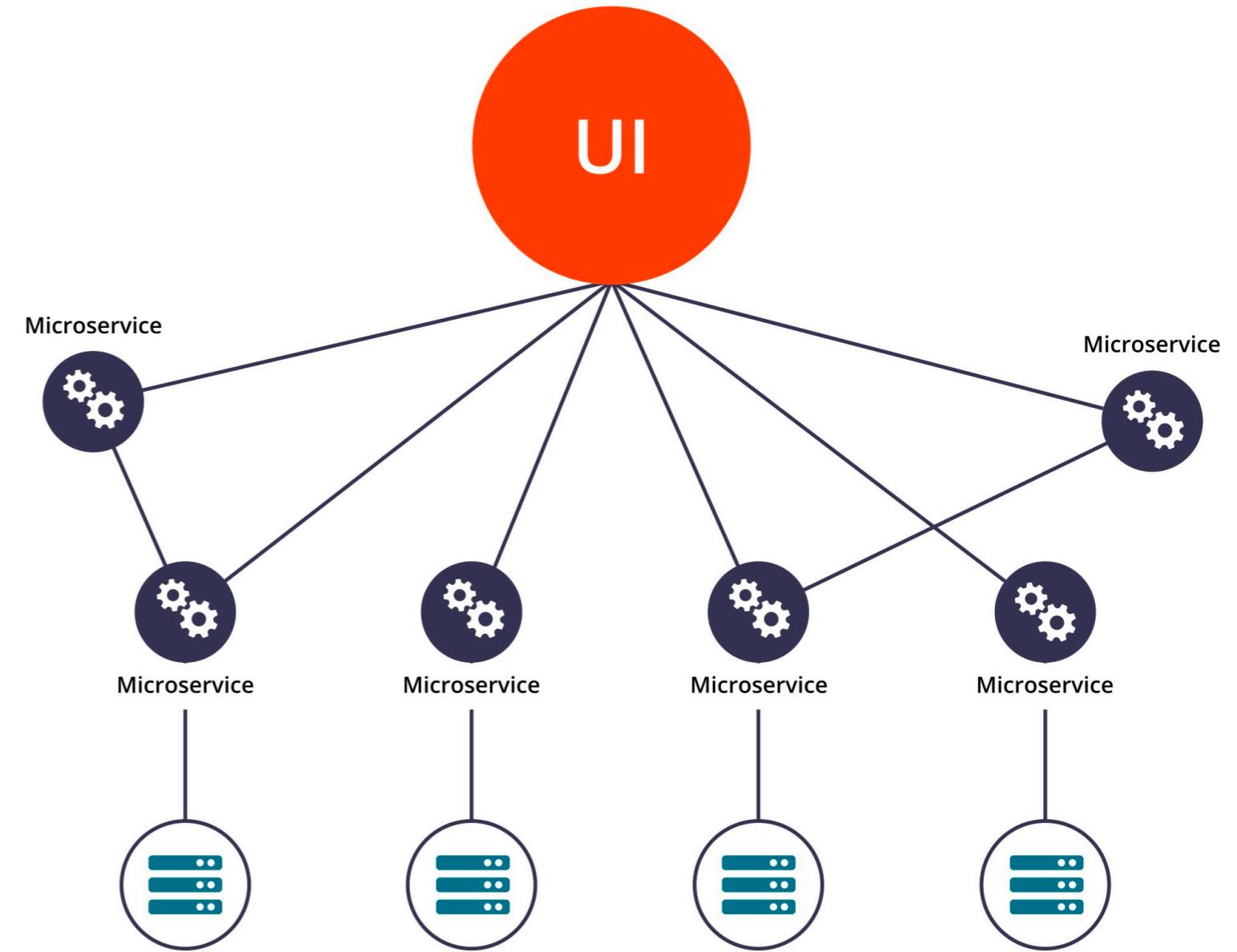
Easy to maintain

Scale independently





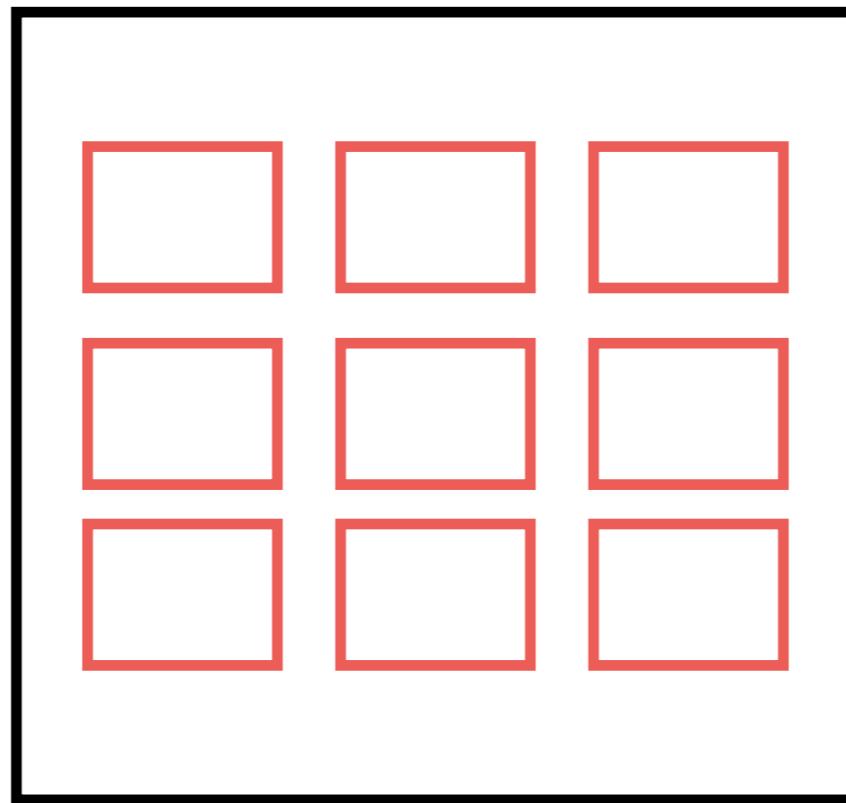
Monolithic Architecture



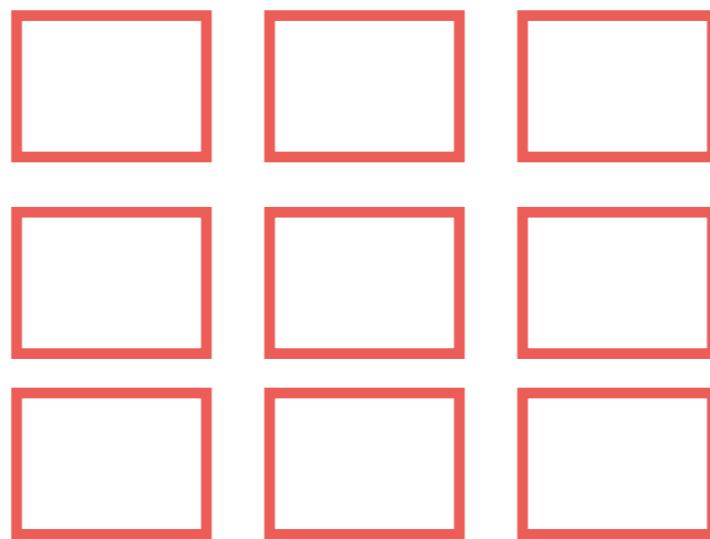
Microservice Architecture



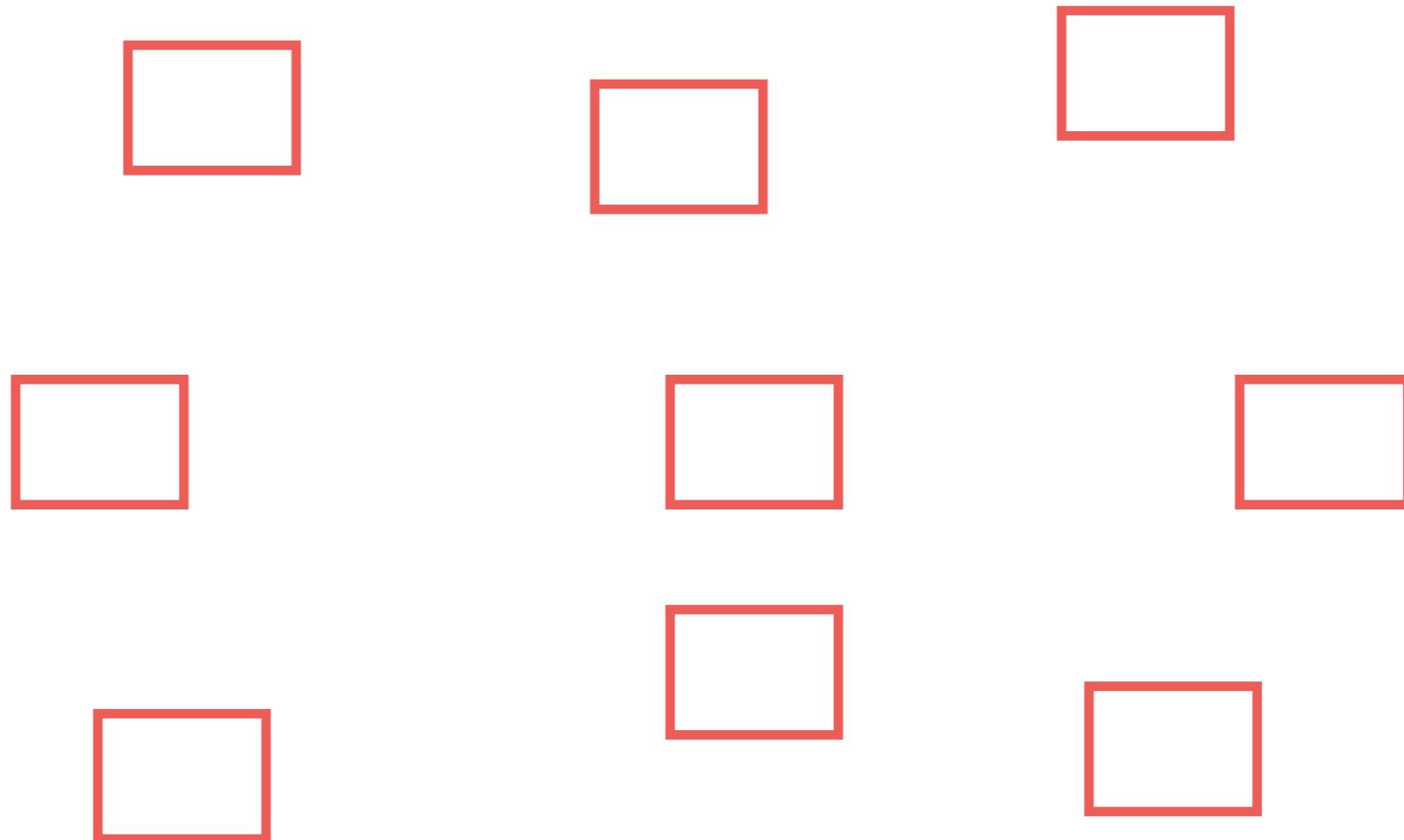
Microservice



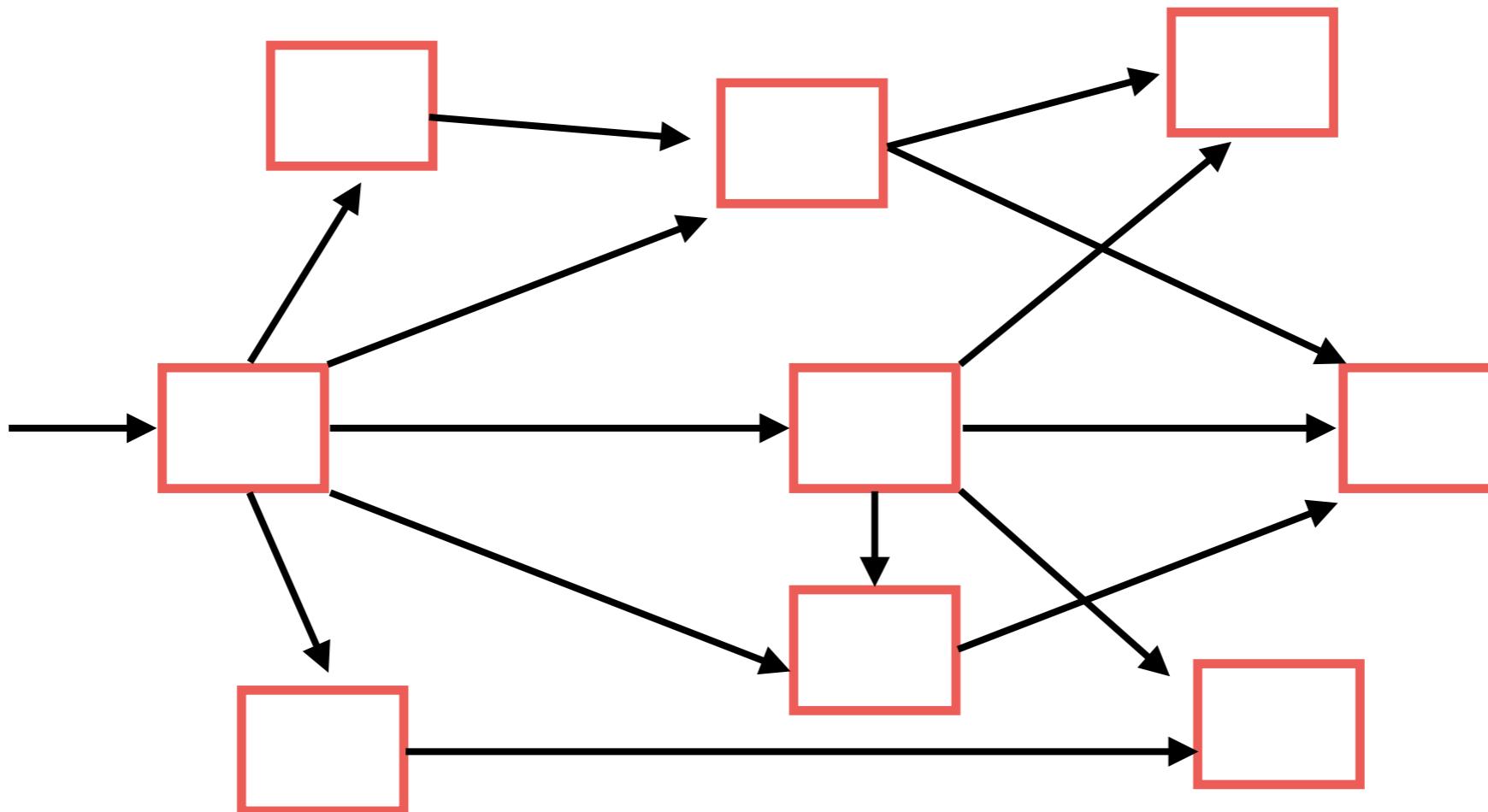
Microservice



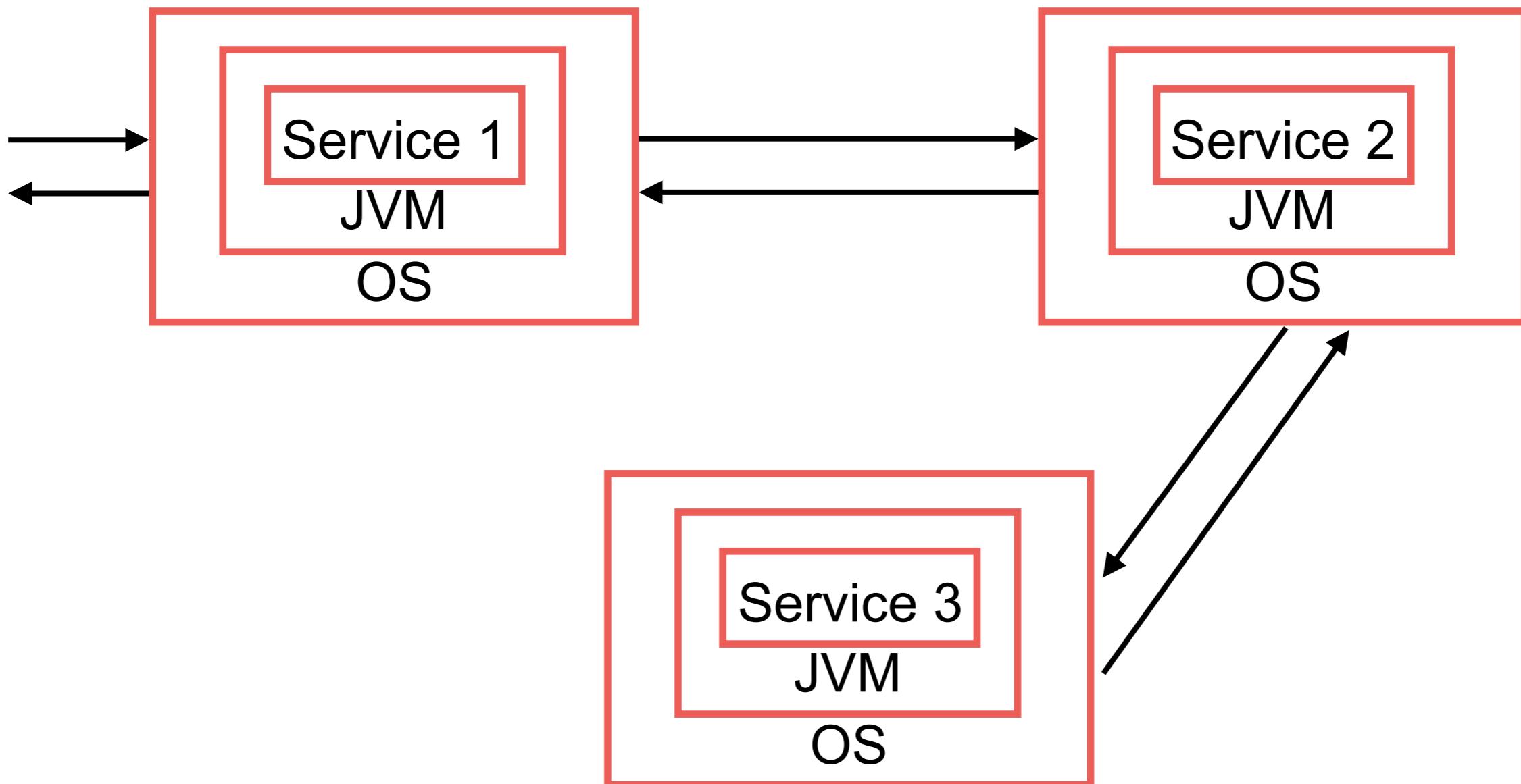
Microservice



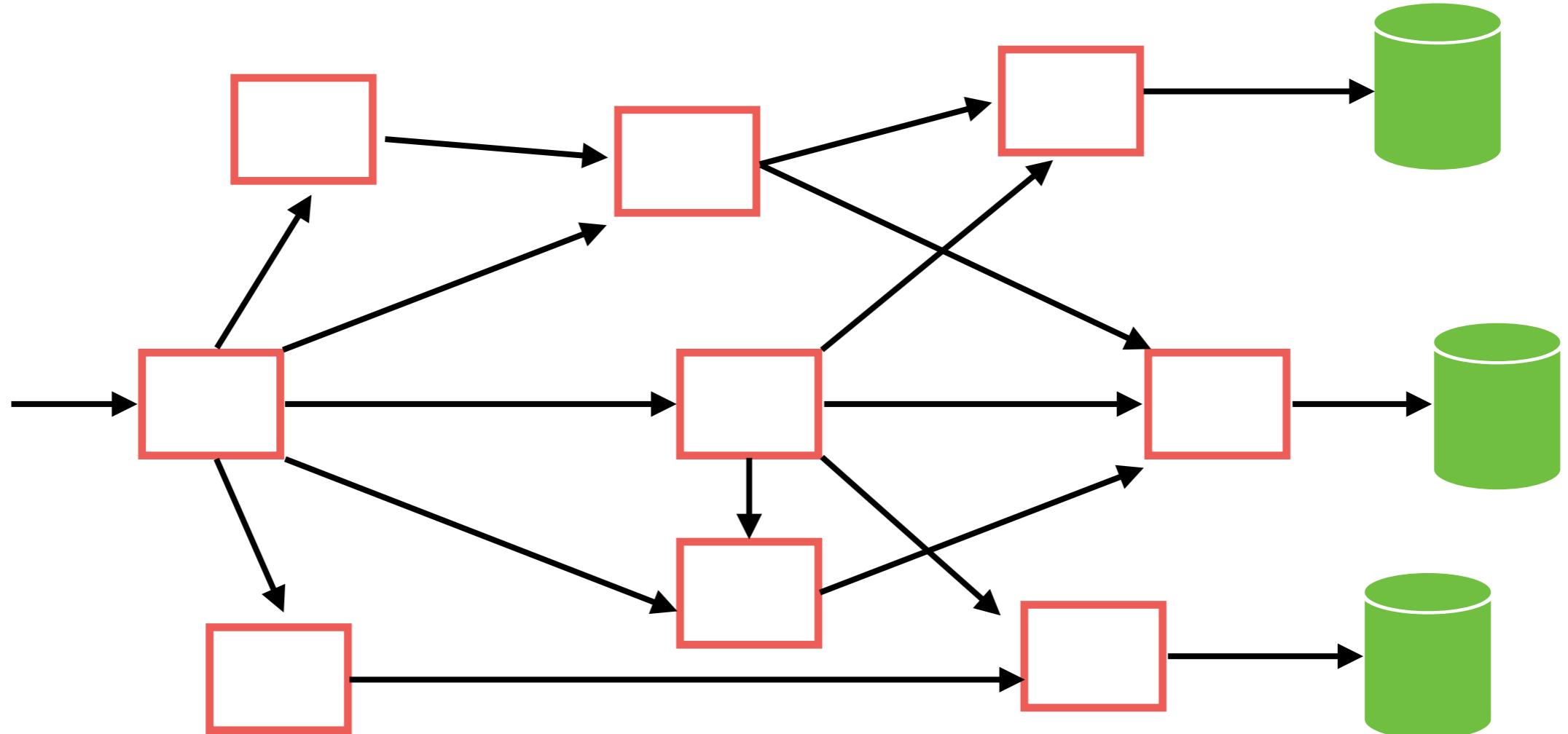
Microservice == Distributed



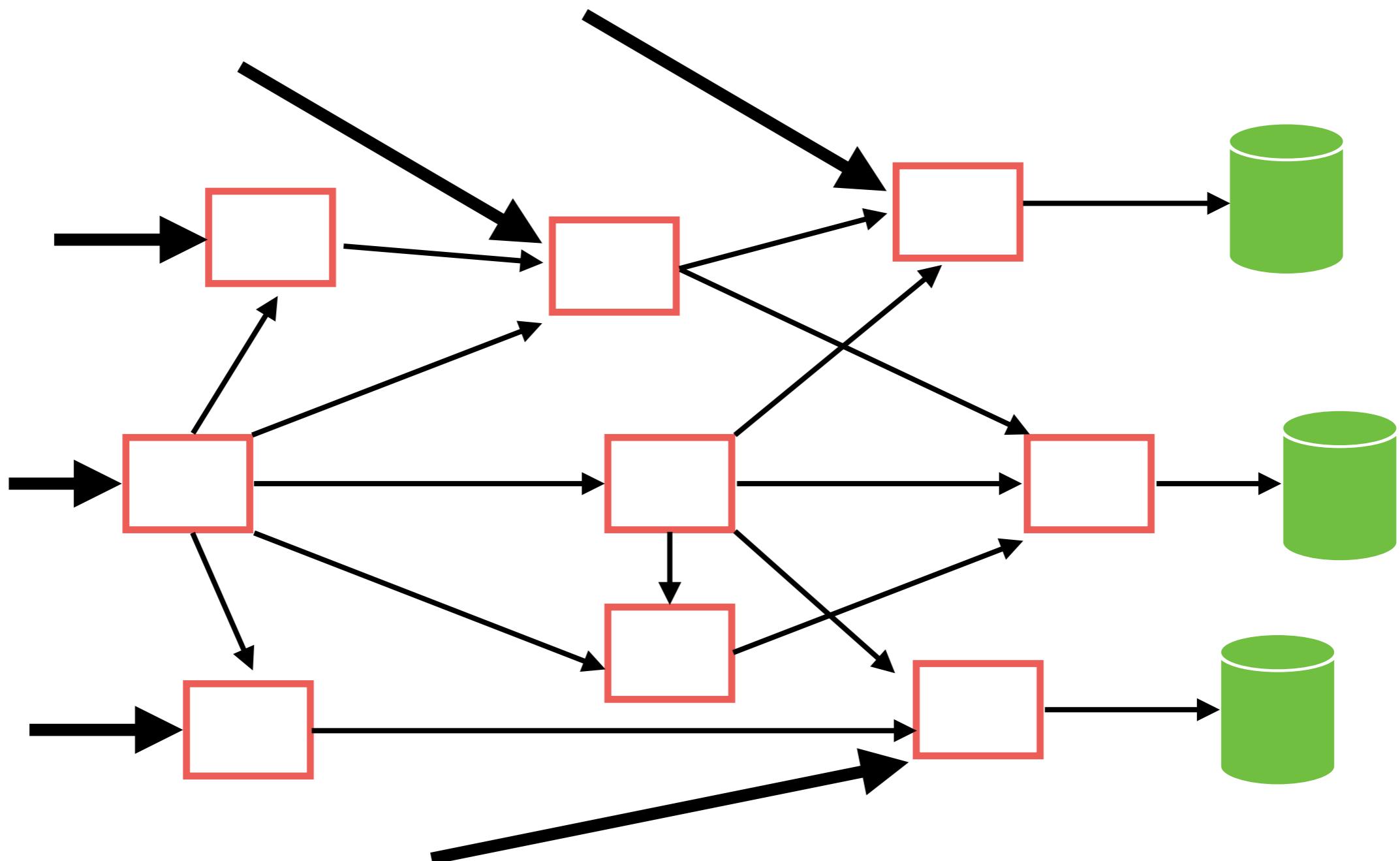
Network of services



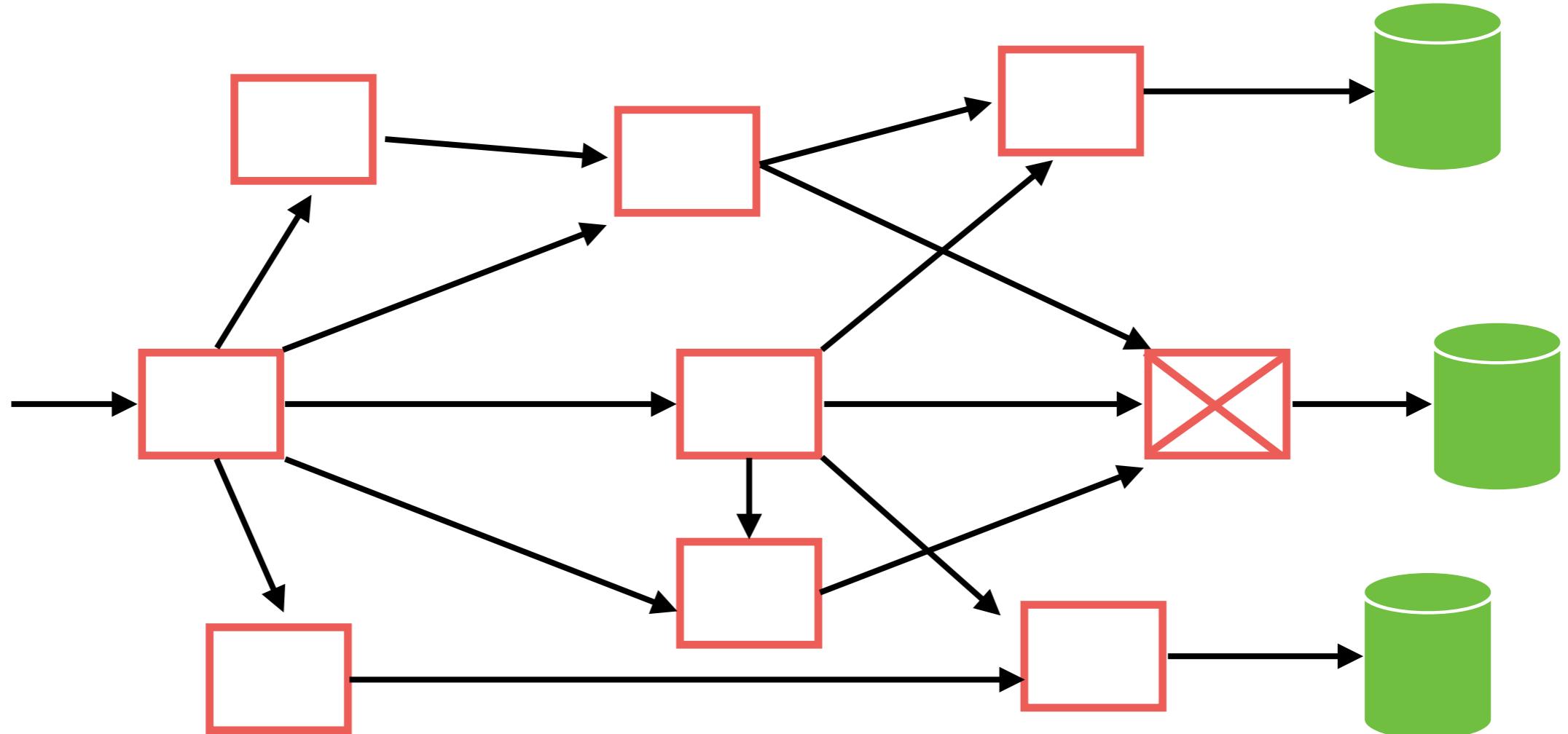
Own data



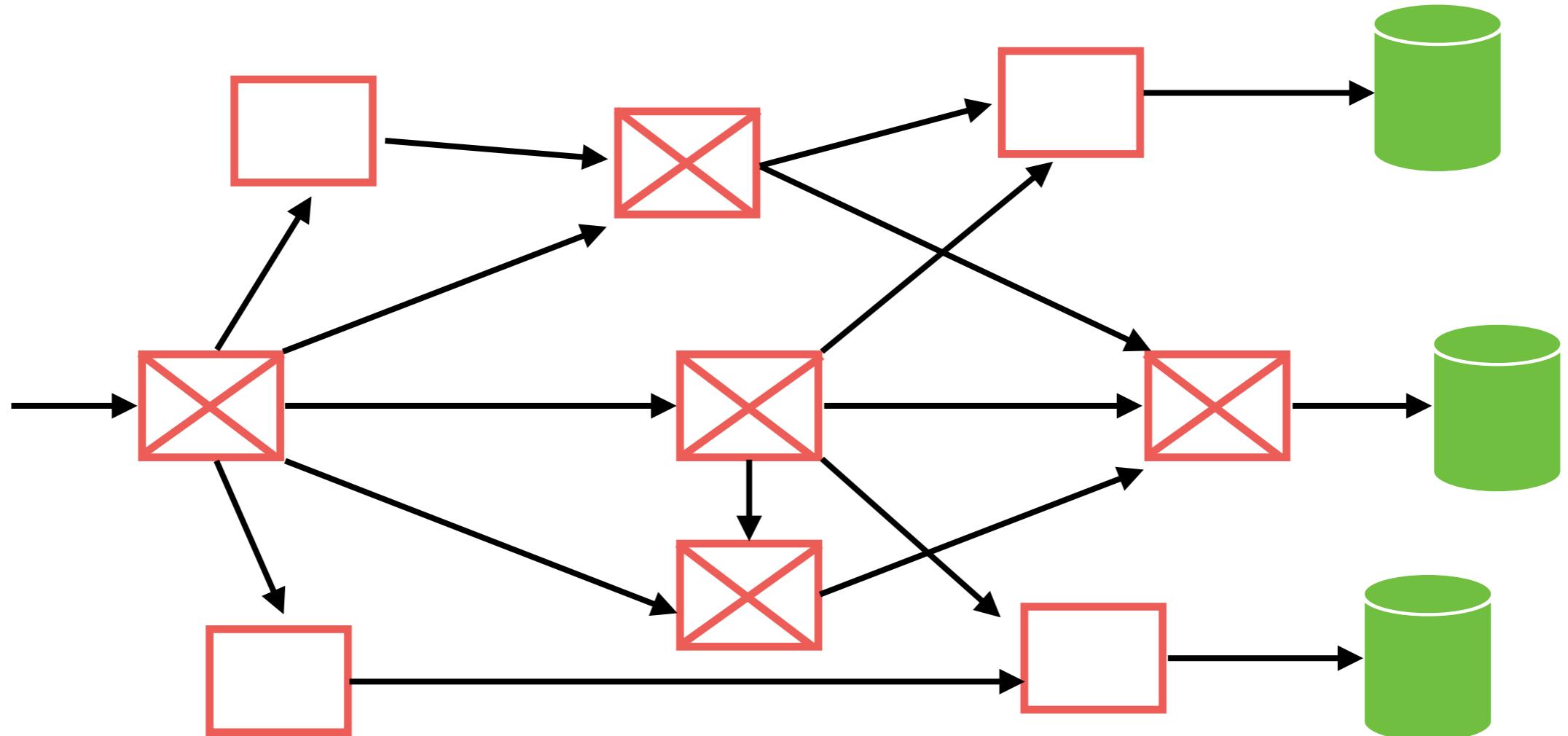
Multiple entry points



Failure !!



Cascading Failure !!

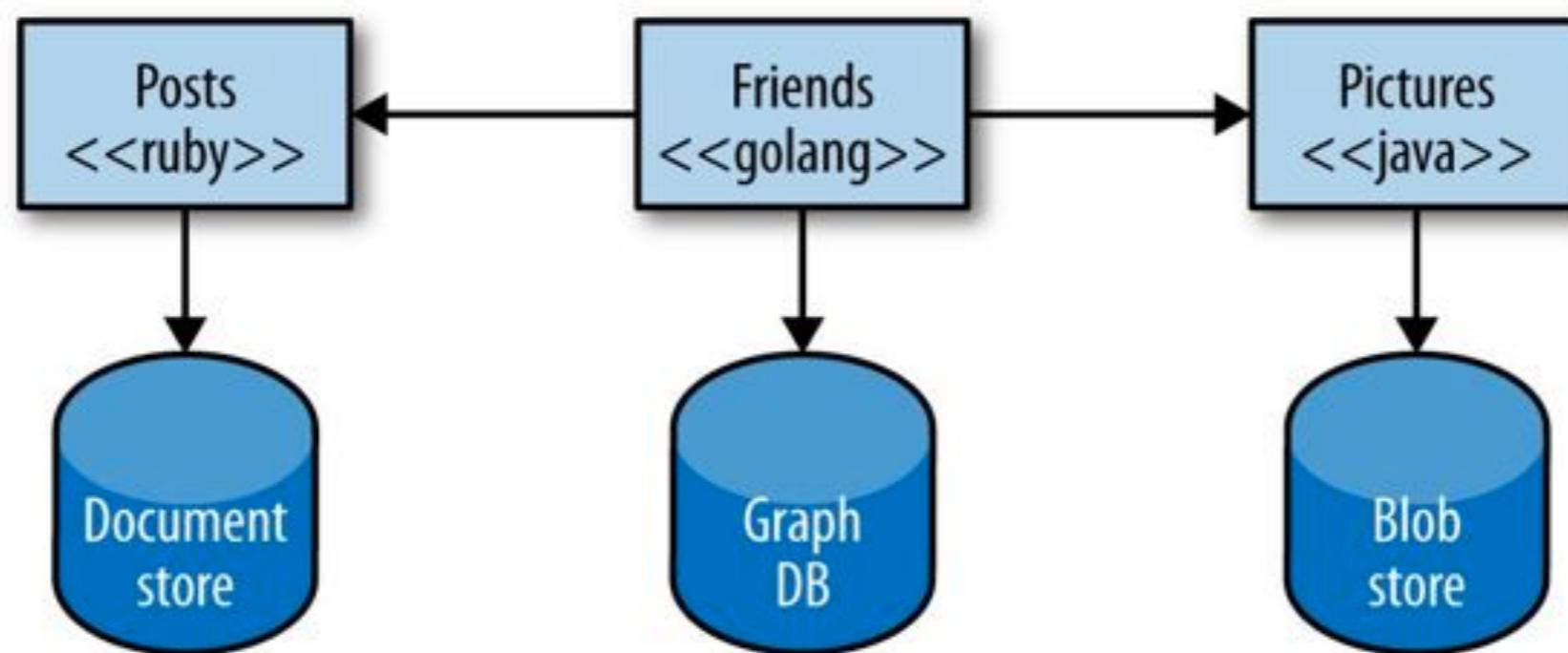


Key Benefits

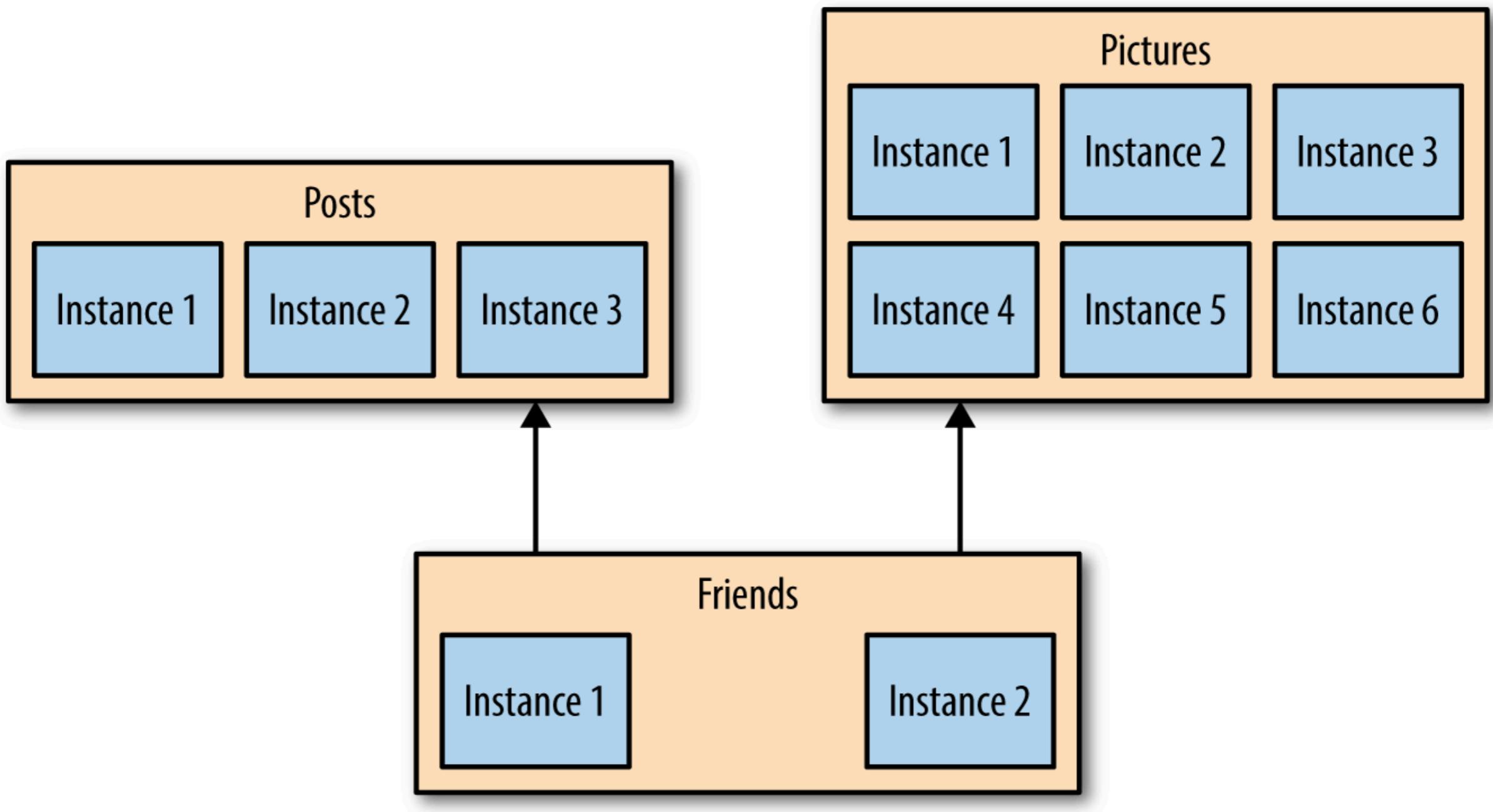


1. Technology heterogeneous

The right tool for each job
Allow easy experimenting and adoption of new technology

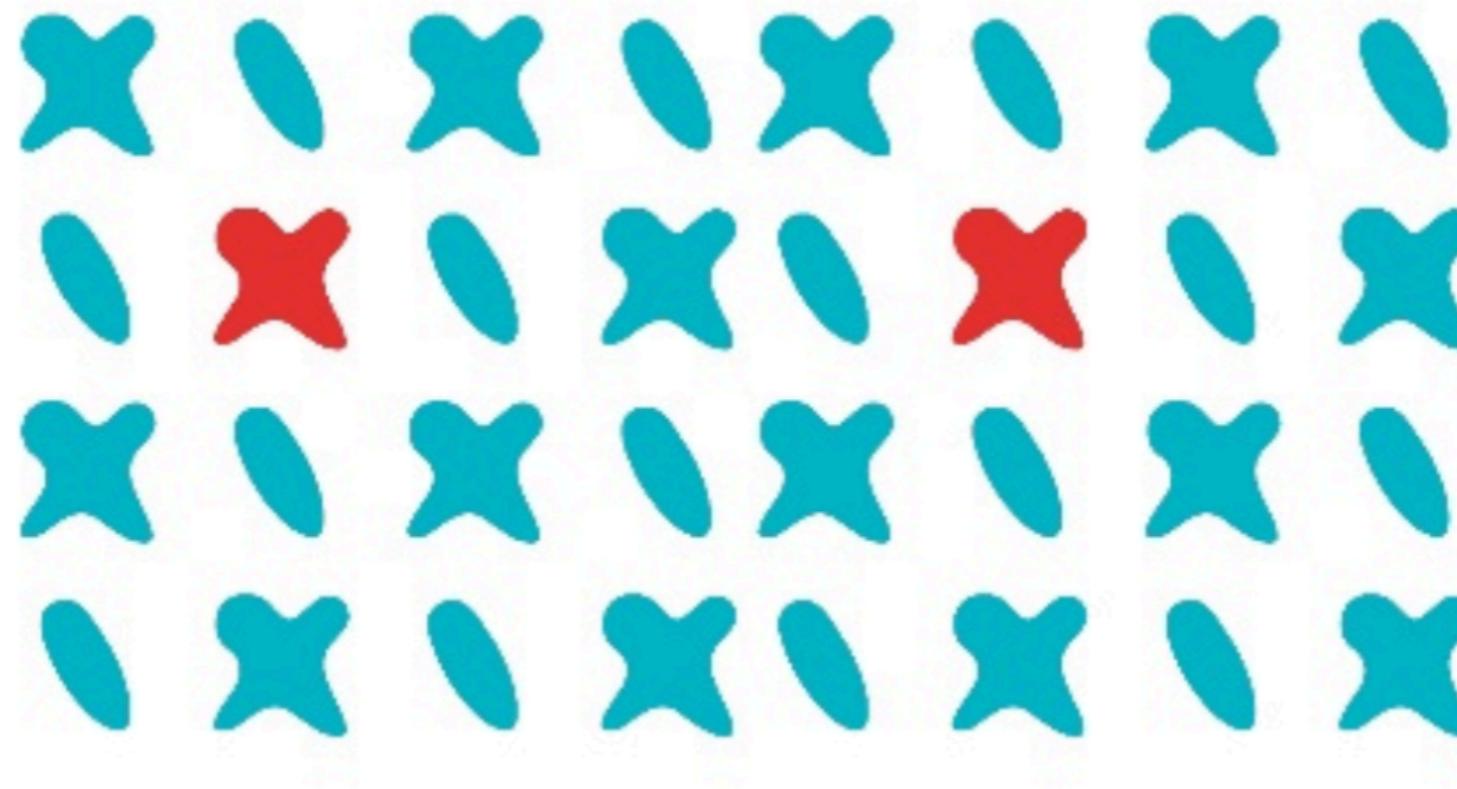


2. Scaling

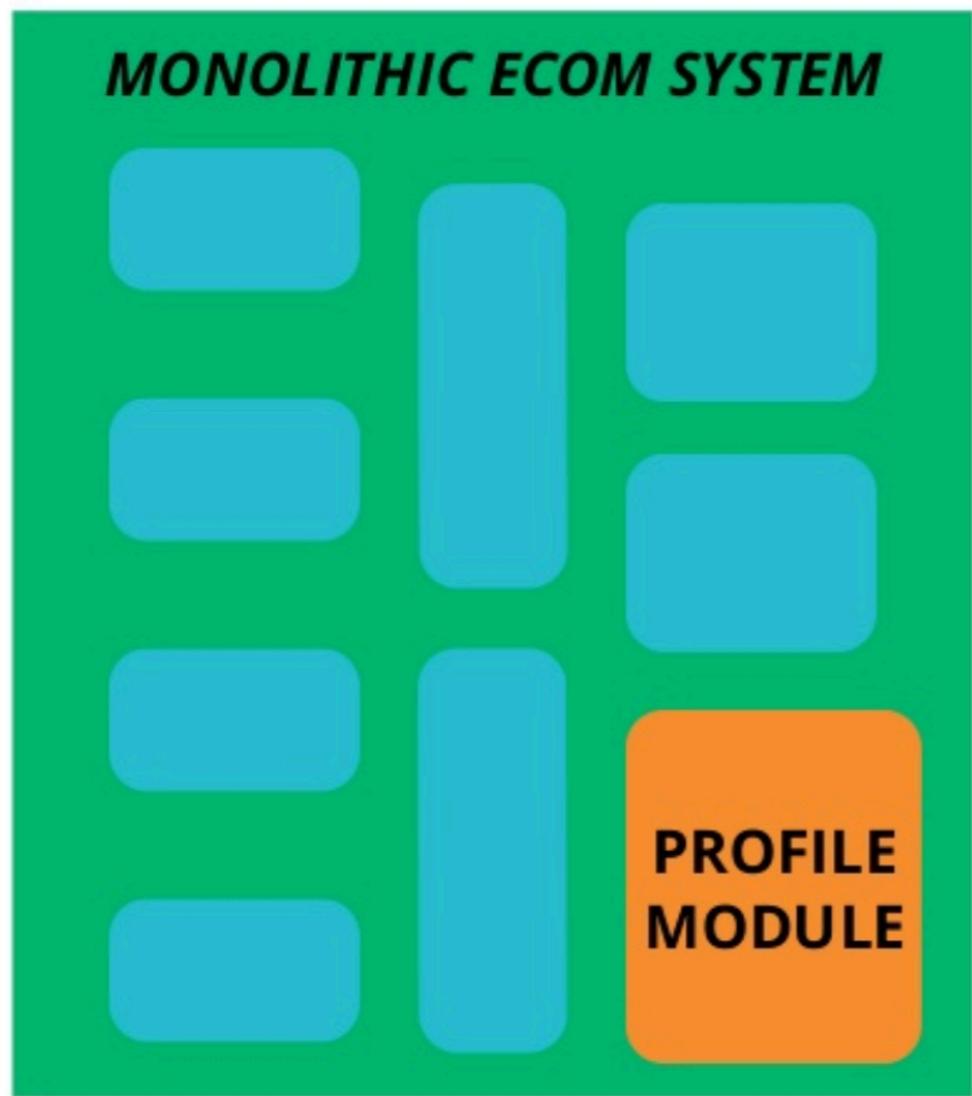


3. Ease of deployment

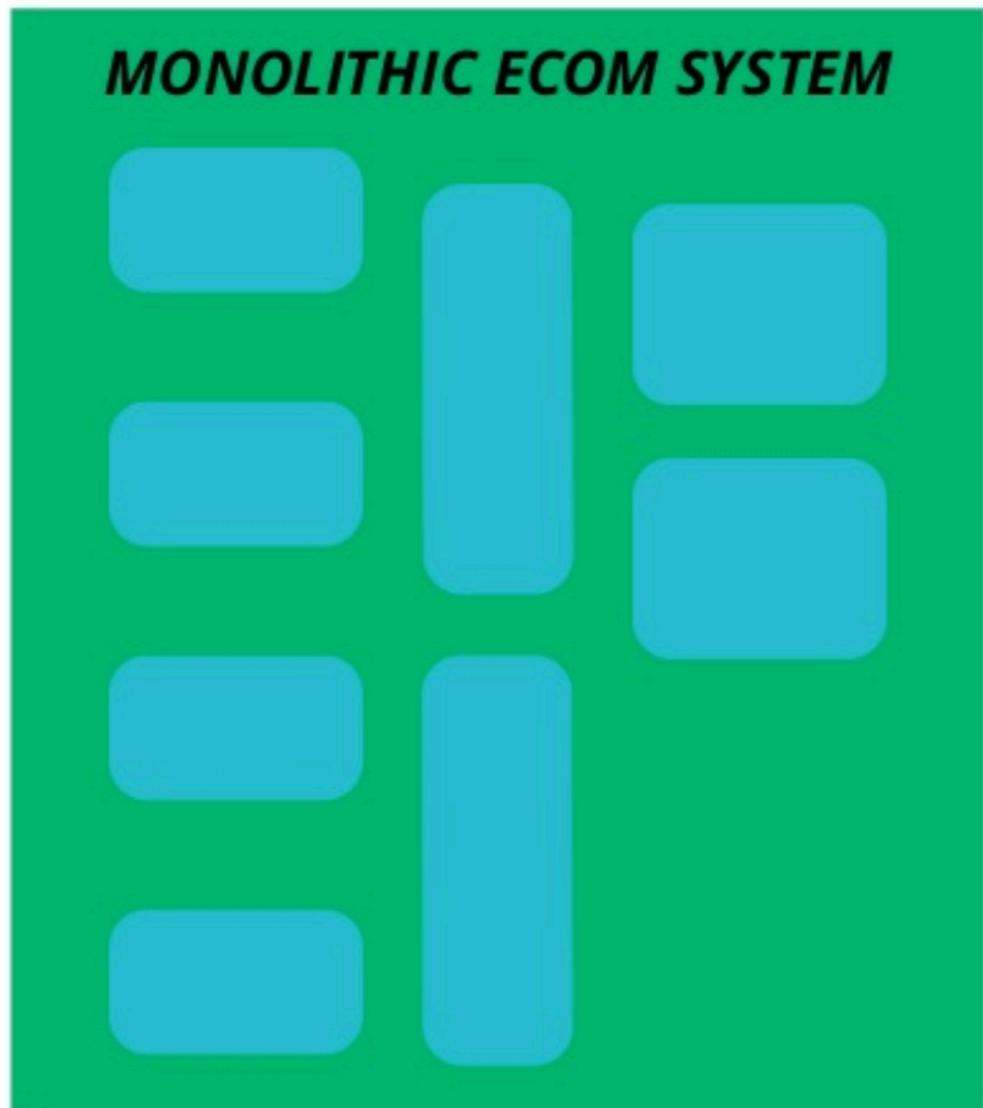
Deploys are faster, independent and problems can be isolated more easily



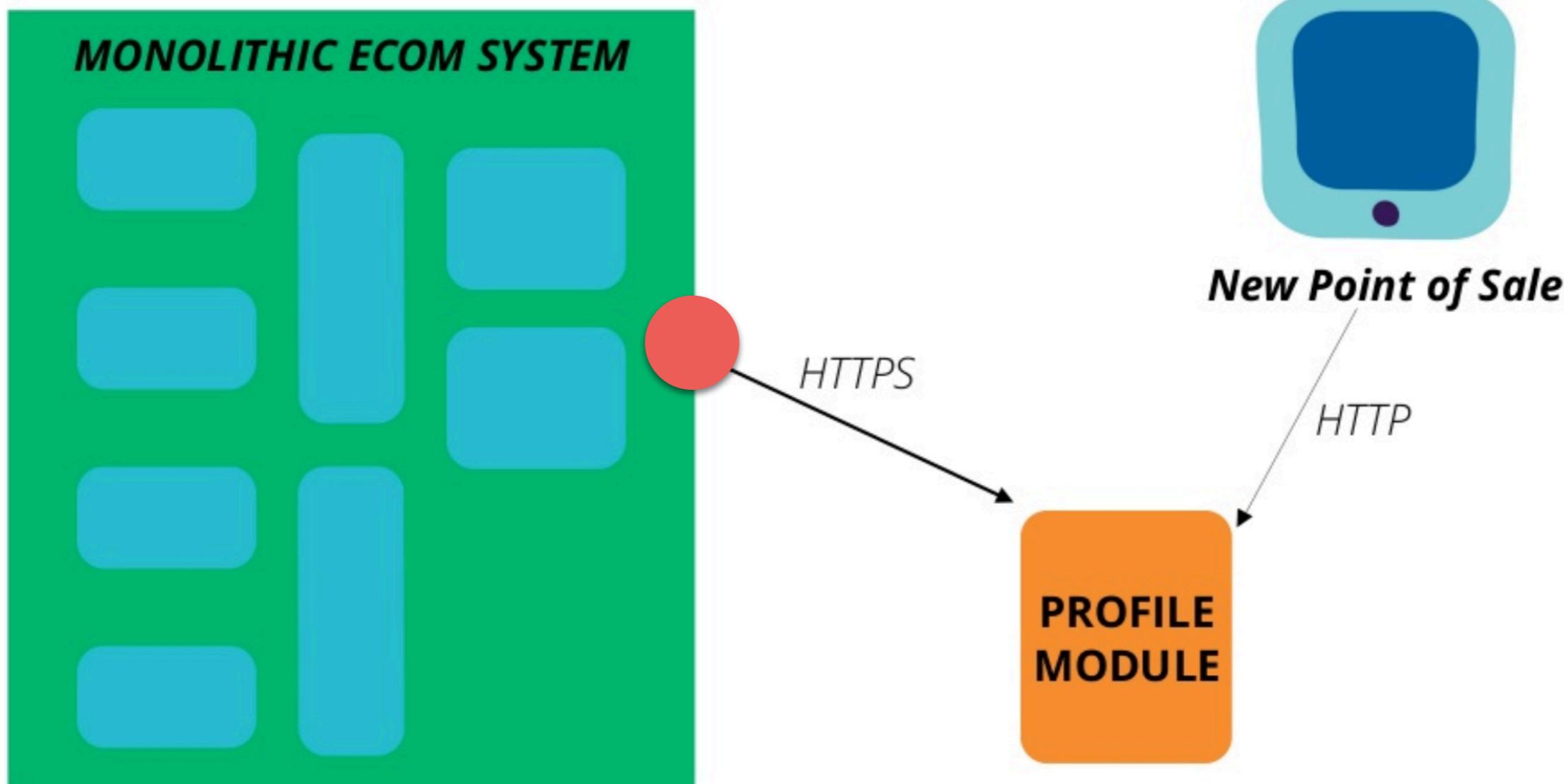
4. Composability and replaceability



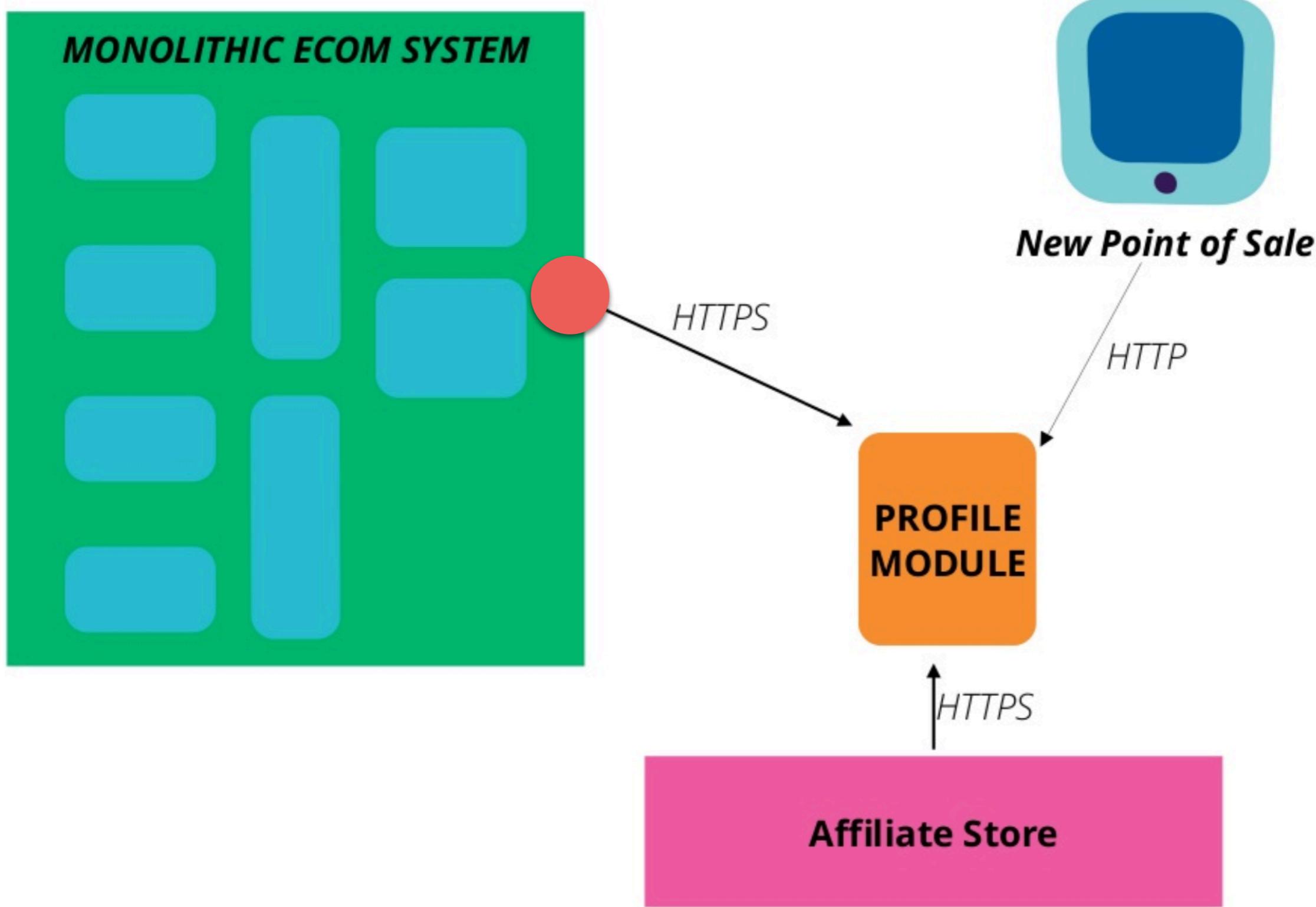
4. Composability and replaceability



4. Composability and replaceability



4. Composability and replaceability

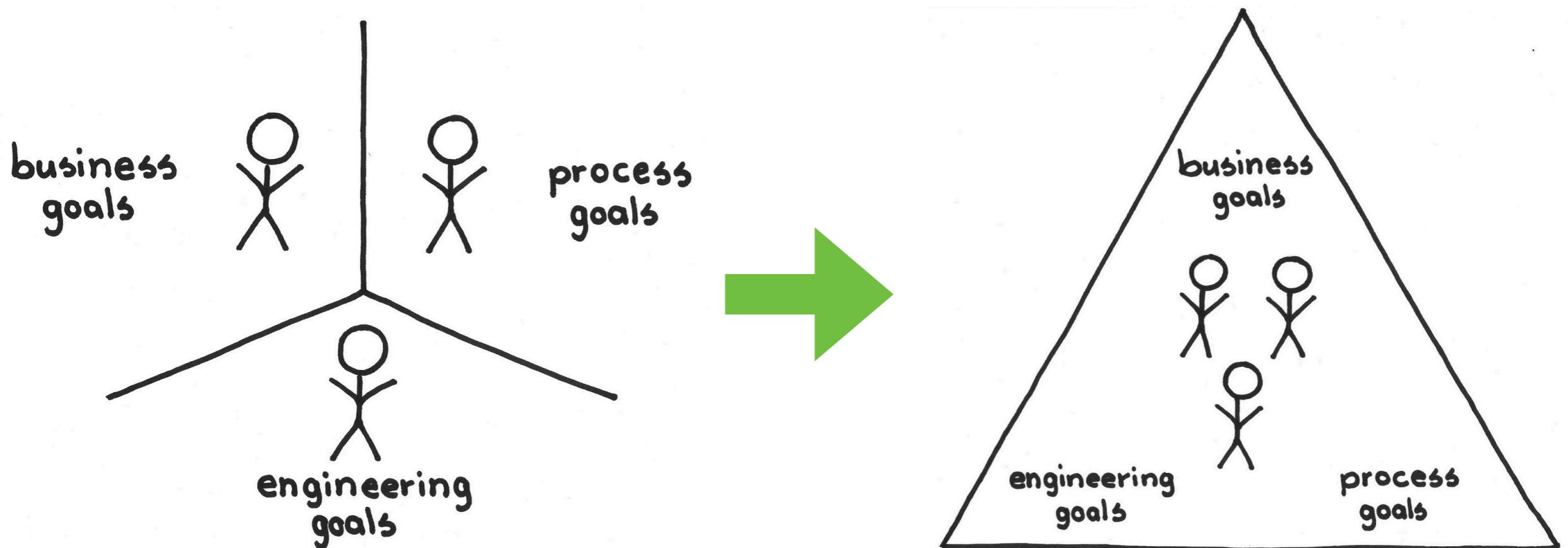


5. Organization alignment

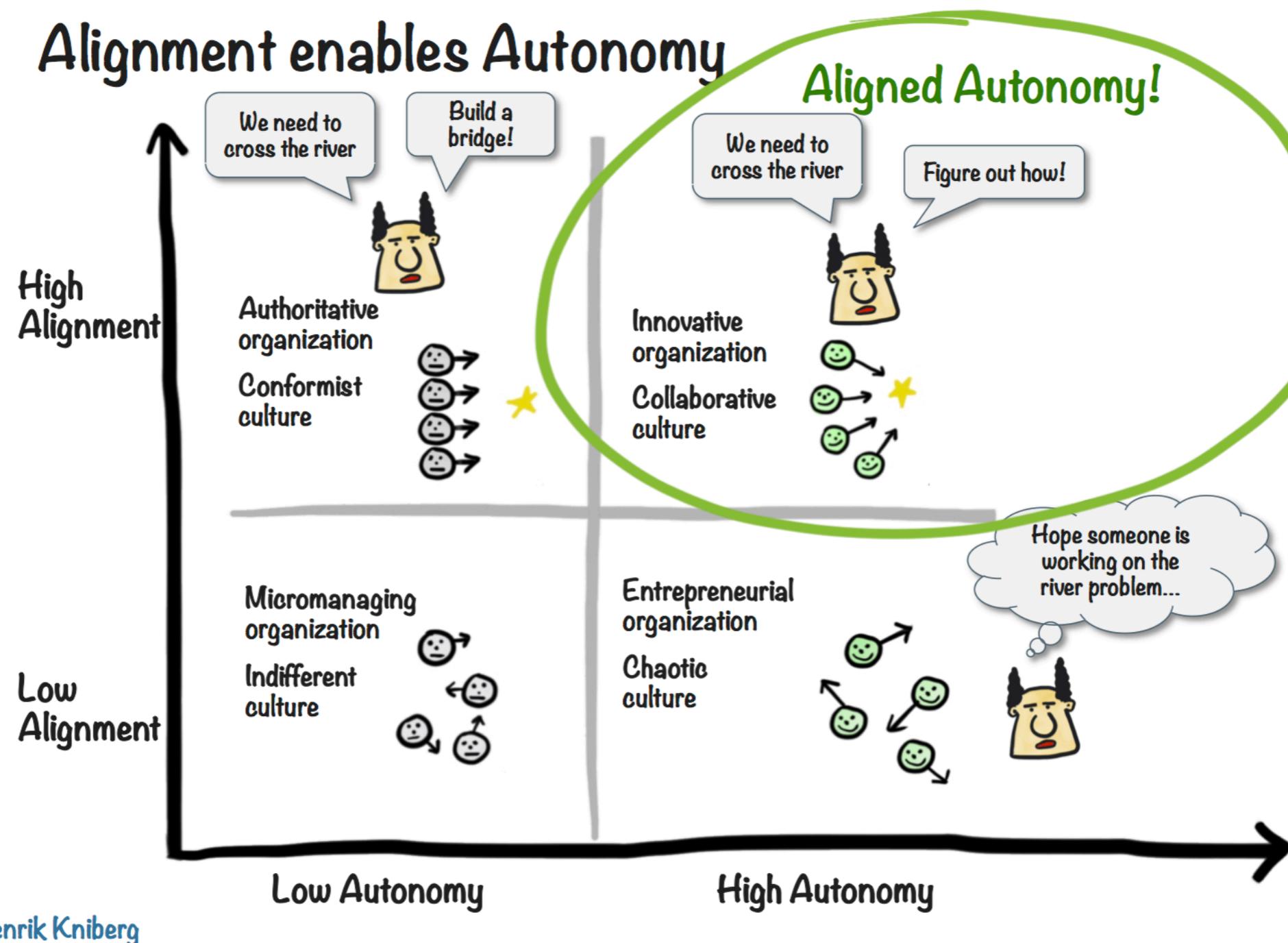
Small teams and smaller codebases



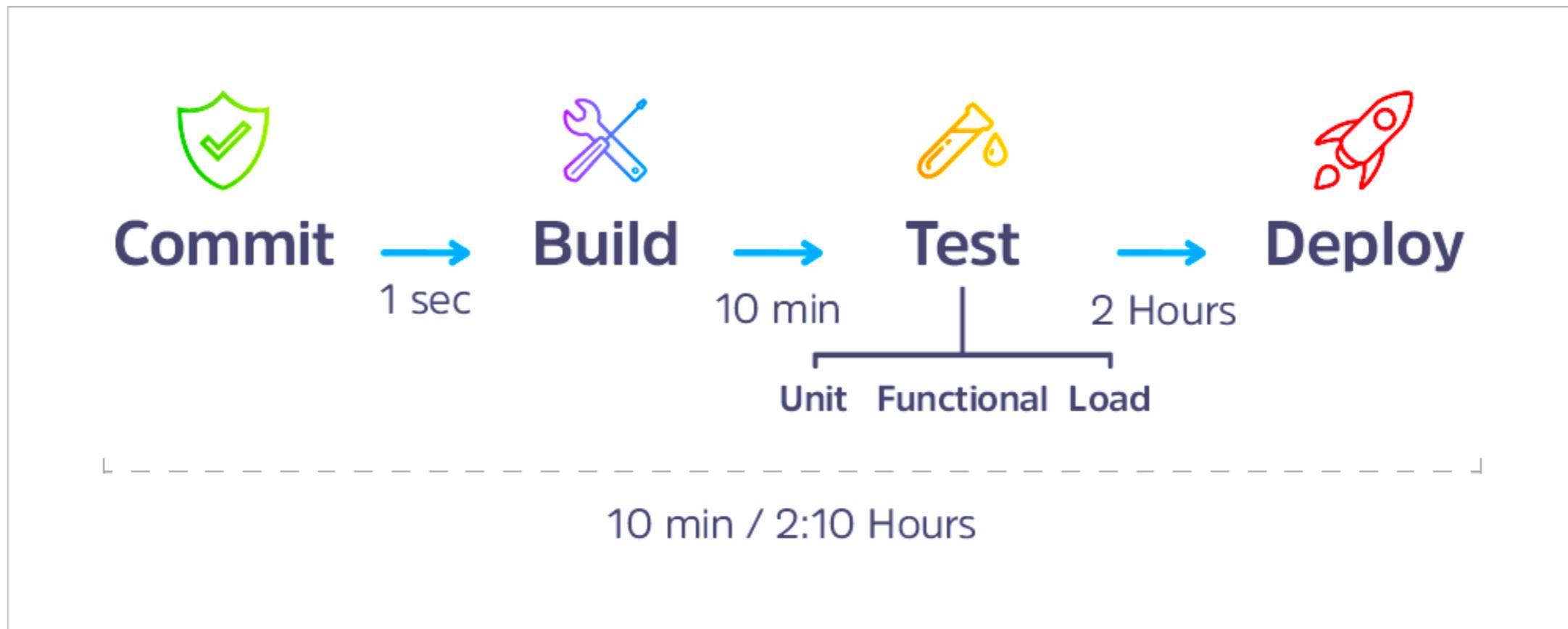
Autonomous team



Autonomous team



Autonomous team



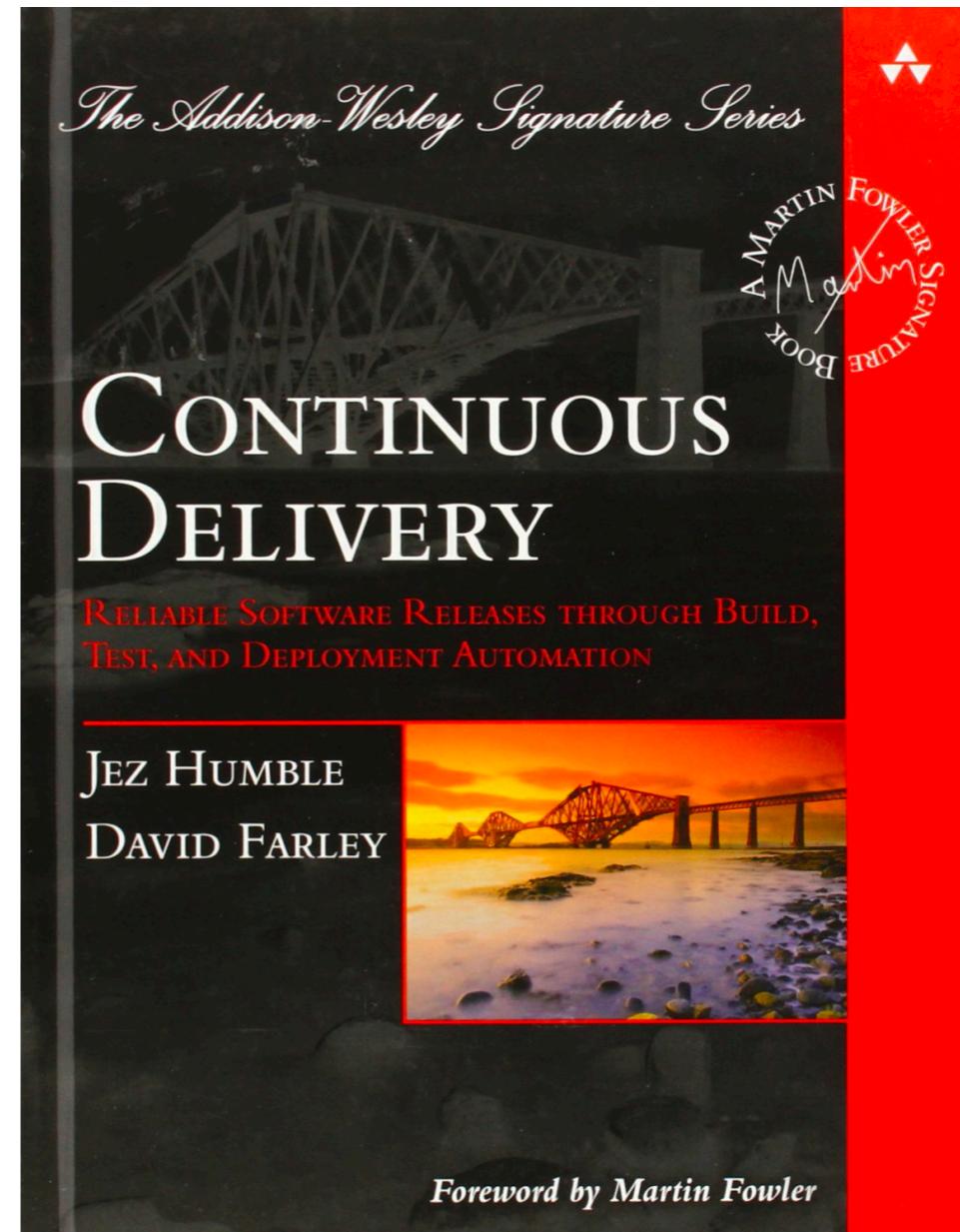
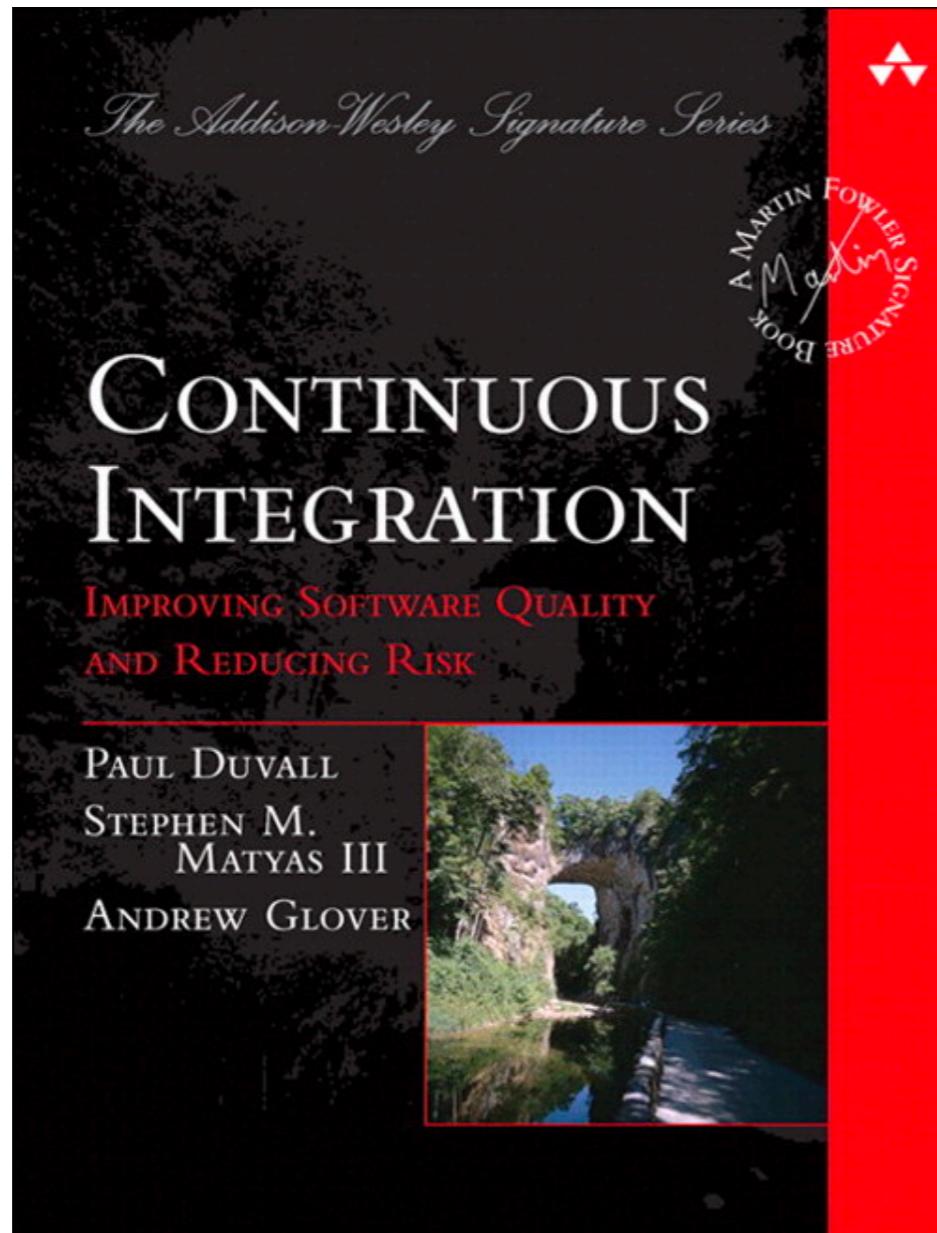
6. Enable Continuous Delivery

A part of DevOps

Set of practices for the **rapid, frequent and reliable delivery** software



Continuous Delivery/Deployment



Microservices

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Continuous Delivery/Deployment

Testability
Deployability

Autonomous team and loose coupling



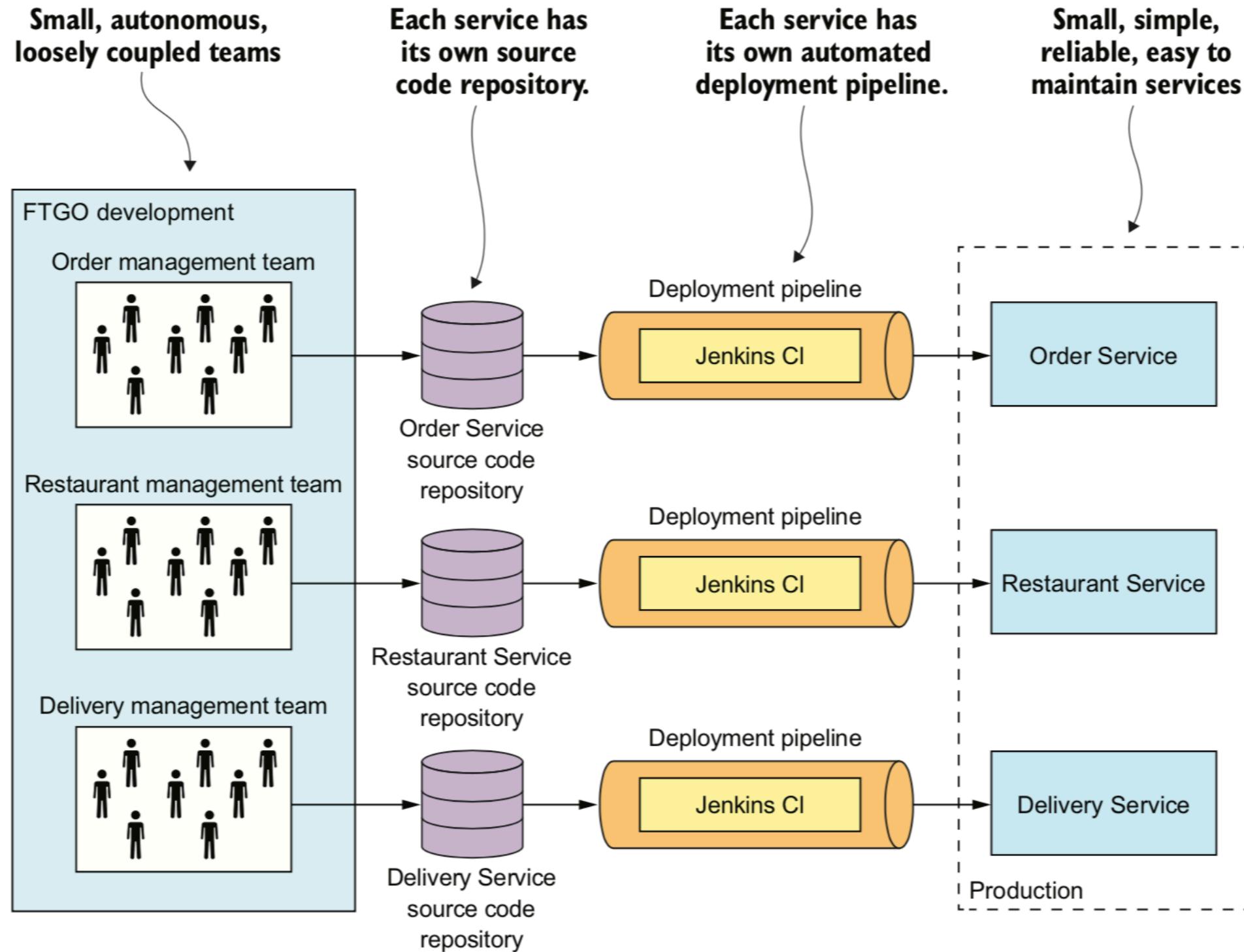
Benefits of Continuous Delivery

Reduce time to market

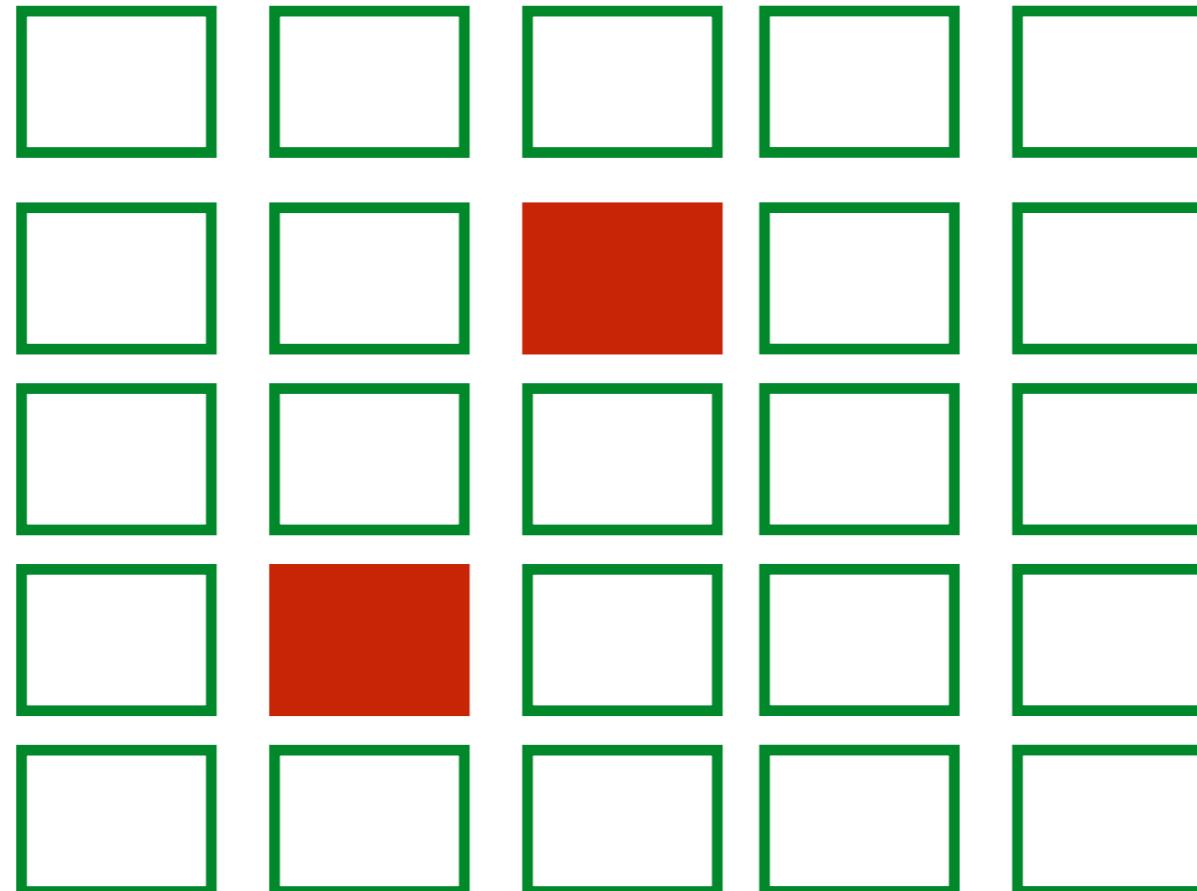
Enable **business** to provide reliable service
Employee satisfaction is higher



Continuous Delivery for service



7. Better Fault isolation



Drawbacks of Microservice



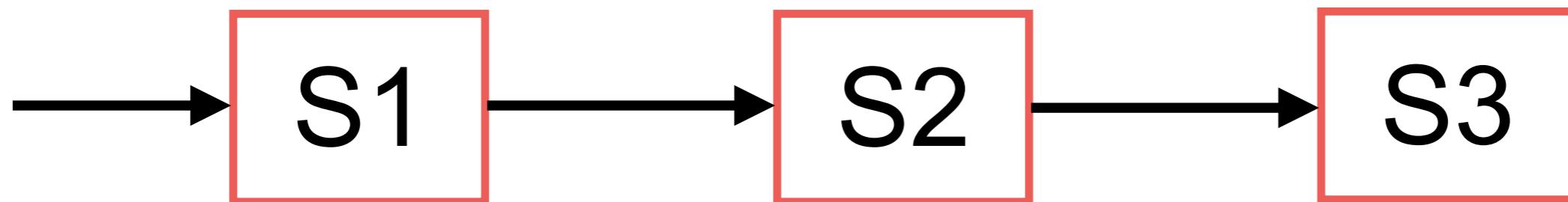
Drawbacks of Microservice

Find the right set of services
Distributed systems are complex
How to develop, testing and deploy ?



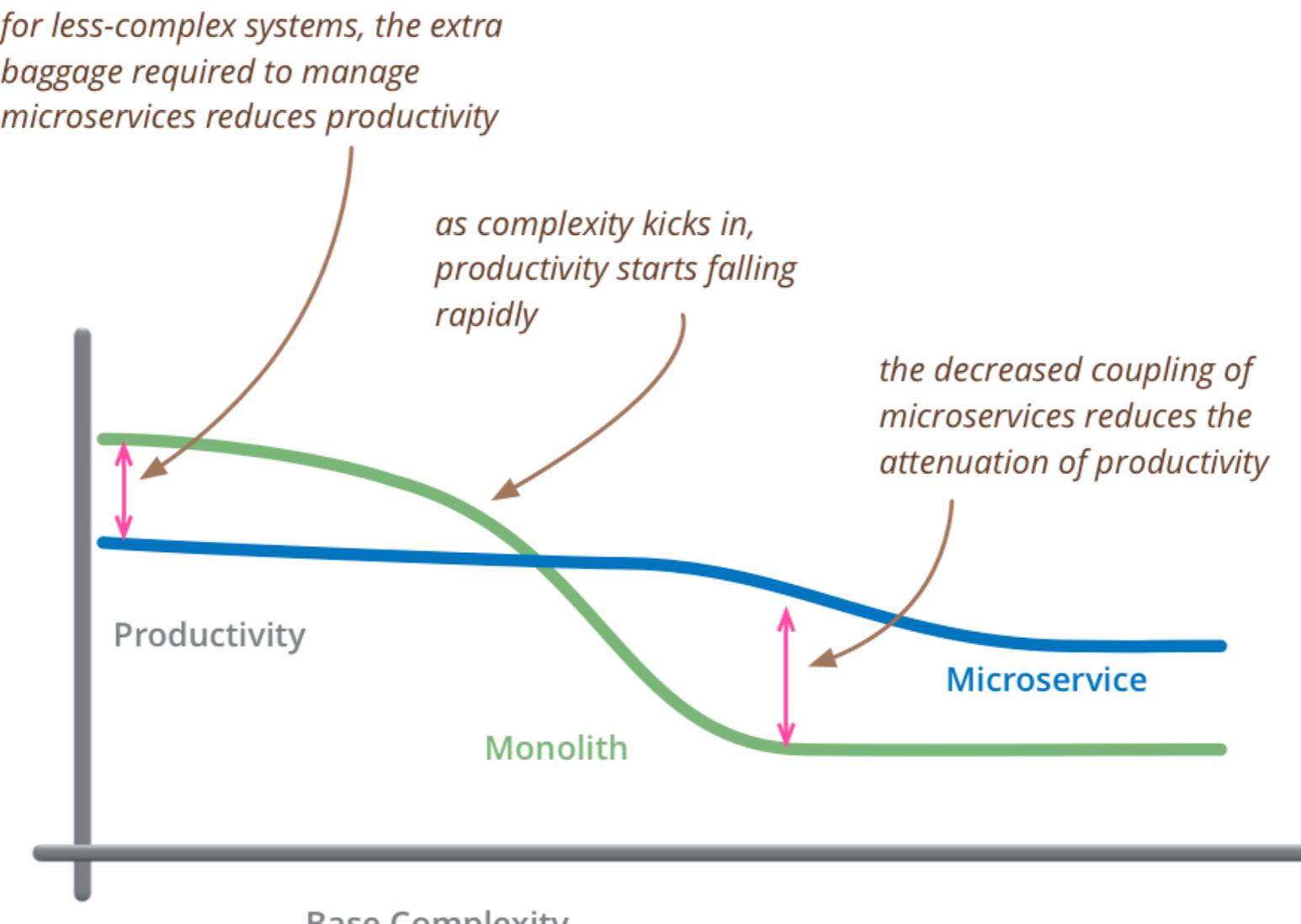
Drawbacks of Microservice

Deploy feature that required multiple service ?



Drawbacks of Microservice

Decide to use when adopt is difficult !!



but remember the skill of the team will outweigh any monolith/microservice choice



Microservice architecture patterns

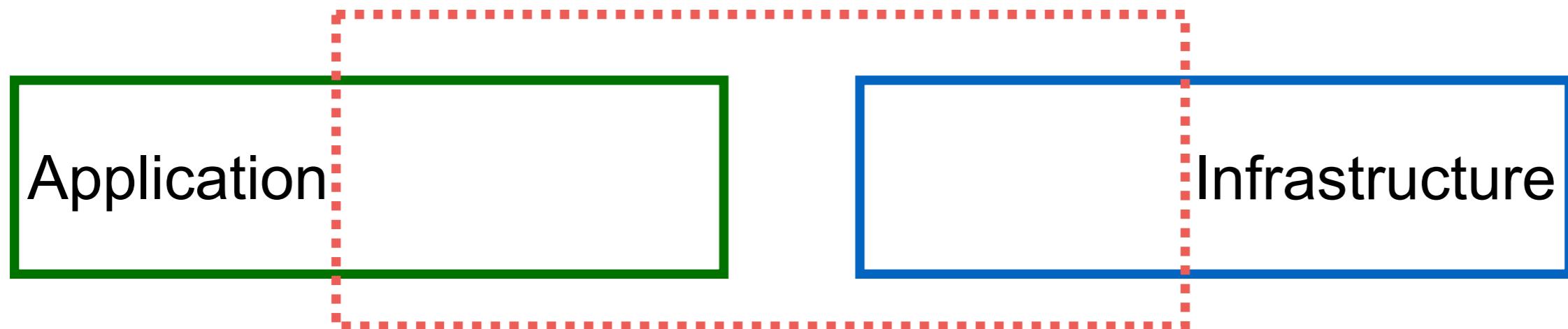


3 Layers of service patterns

Application patterns

Application infrastructure pattern

Infrastructure pattern



Patterns for Microservice

Decompose application into services

 Communication patterns

 Data consistency patterns

 Service deployment patterns

 Observability patterns

 Automated testing of services

 Security patterns



Decompose application into services



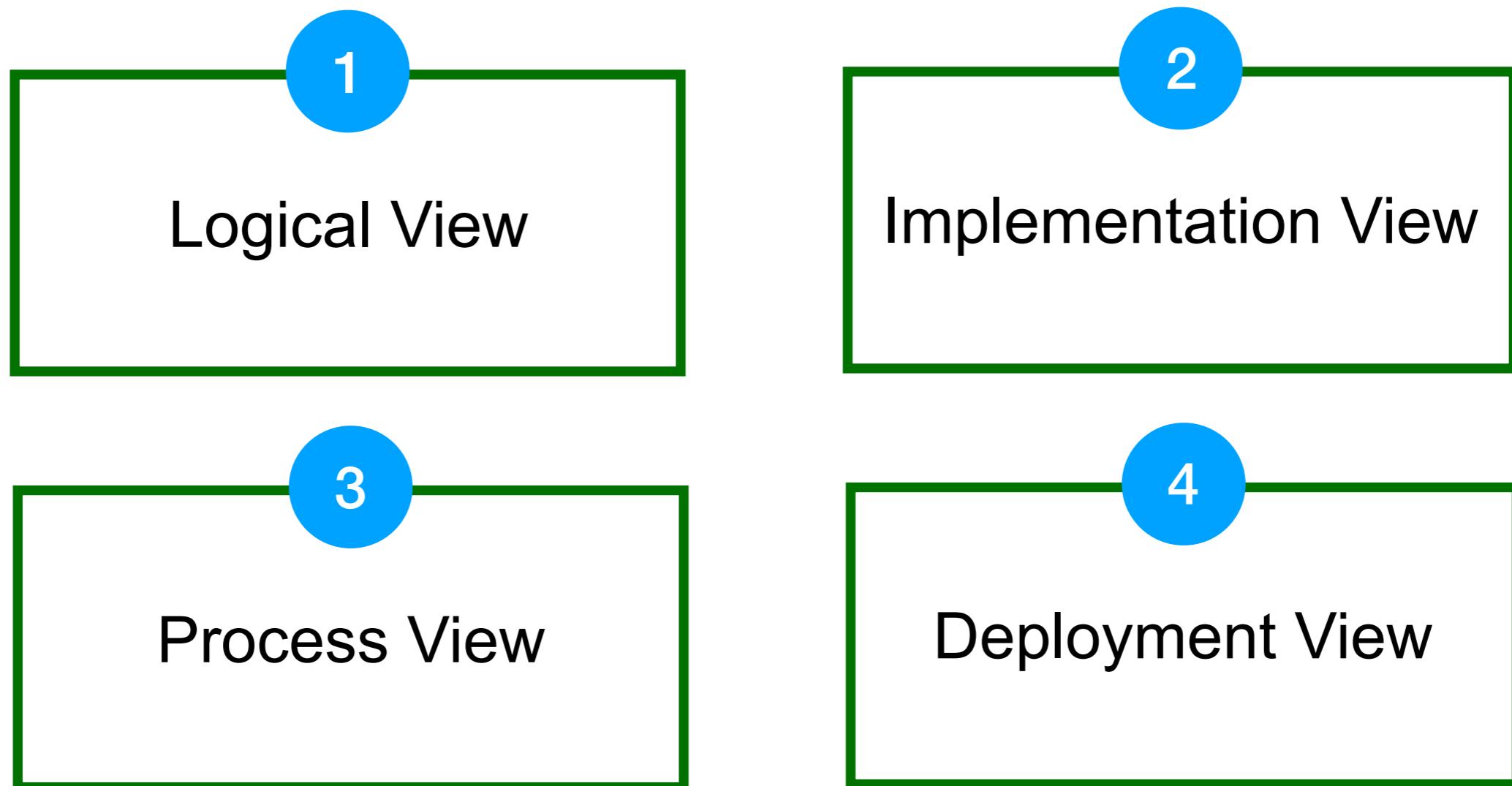
Premature splitting is the root of
all evil.



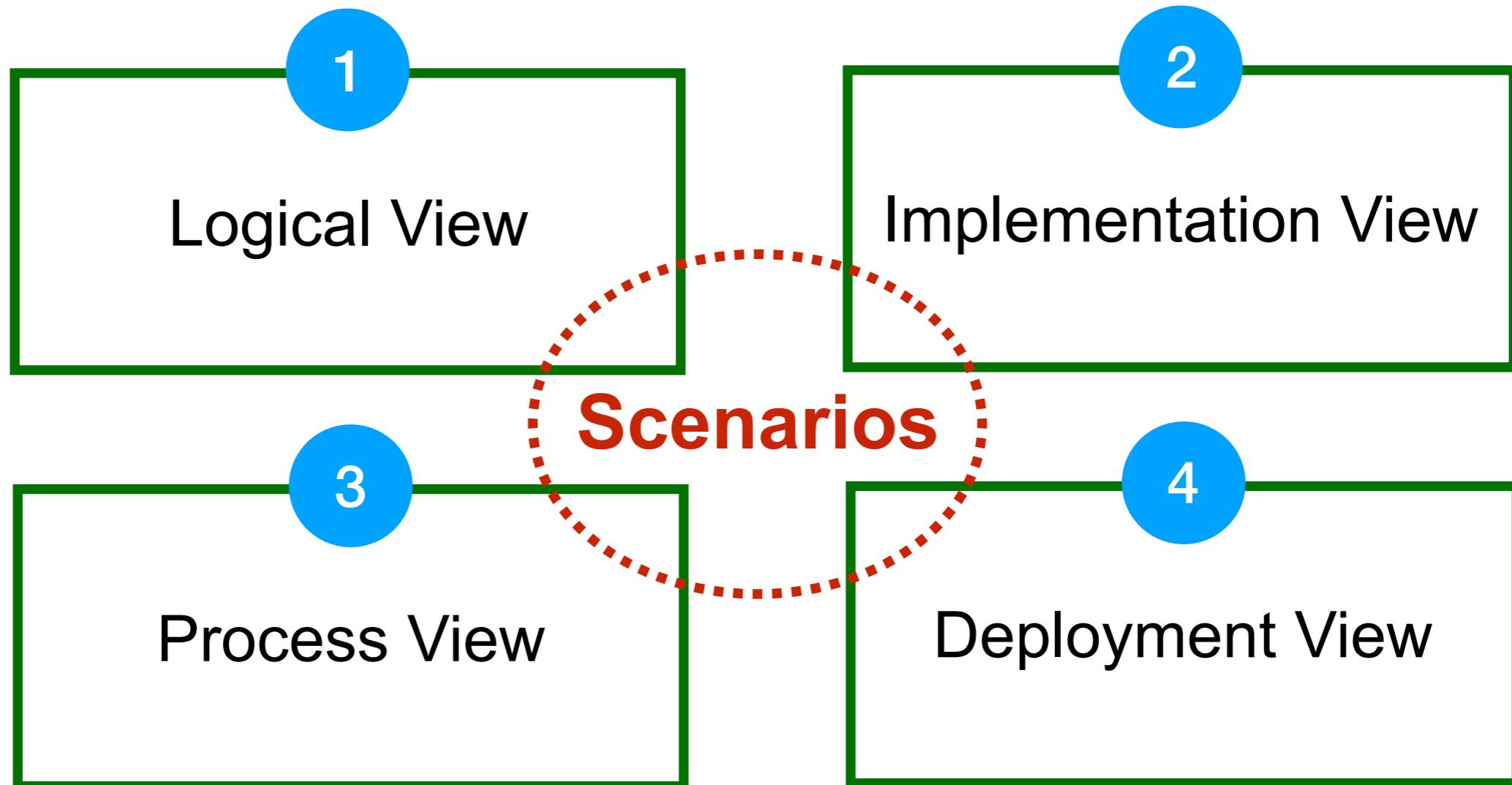
Every time you make the decision
to split out a new microservice,
there's a **risk** of ending up with a
bloated app.



4 View model of Software Architecture



4 View model of Software Architecture



What is service ?

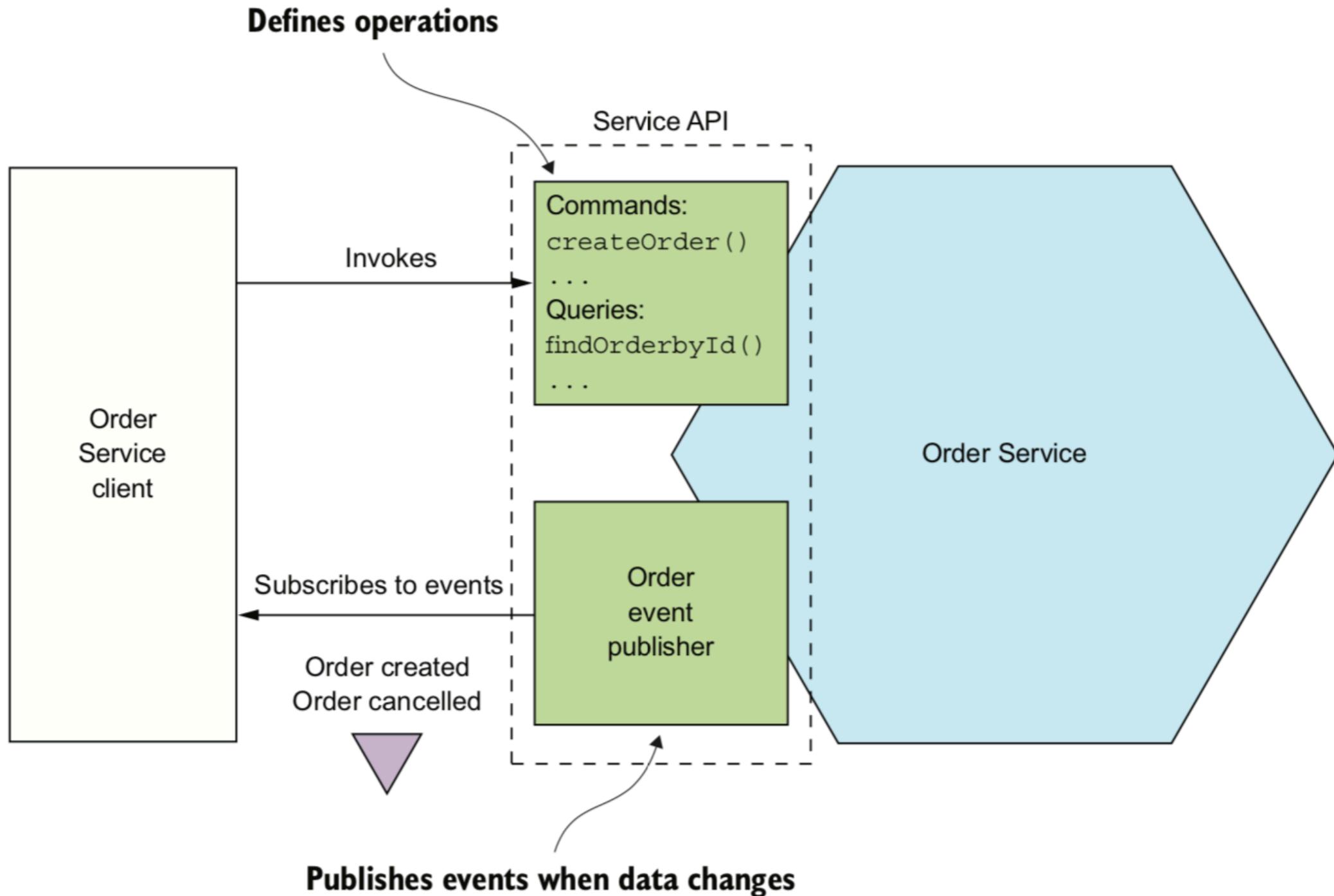
Standalone and loosely couple

Independently deployable software component

Implement some useful functionality



Example of service



Decompose application into services

By business capability ?
By subdomain/technical ?



Step to define services

1. Identify system operations
2. Identify services
3. Define service APIs and collaboration

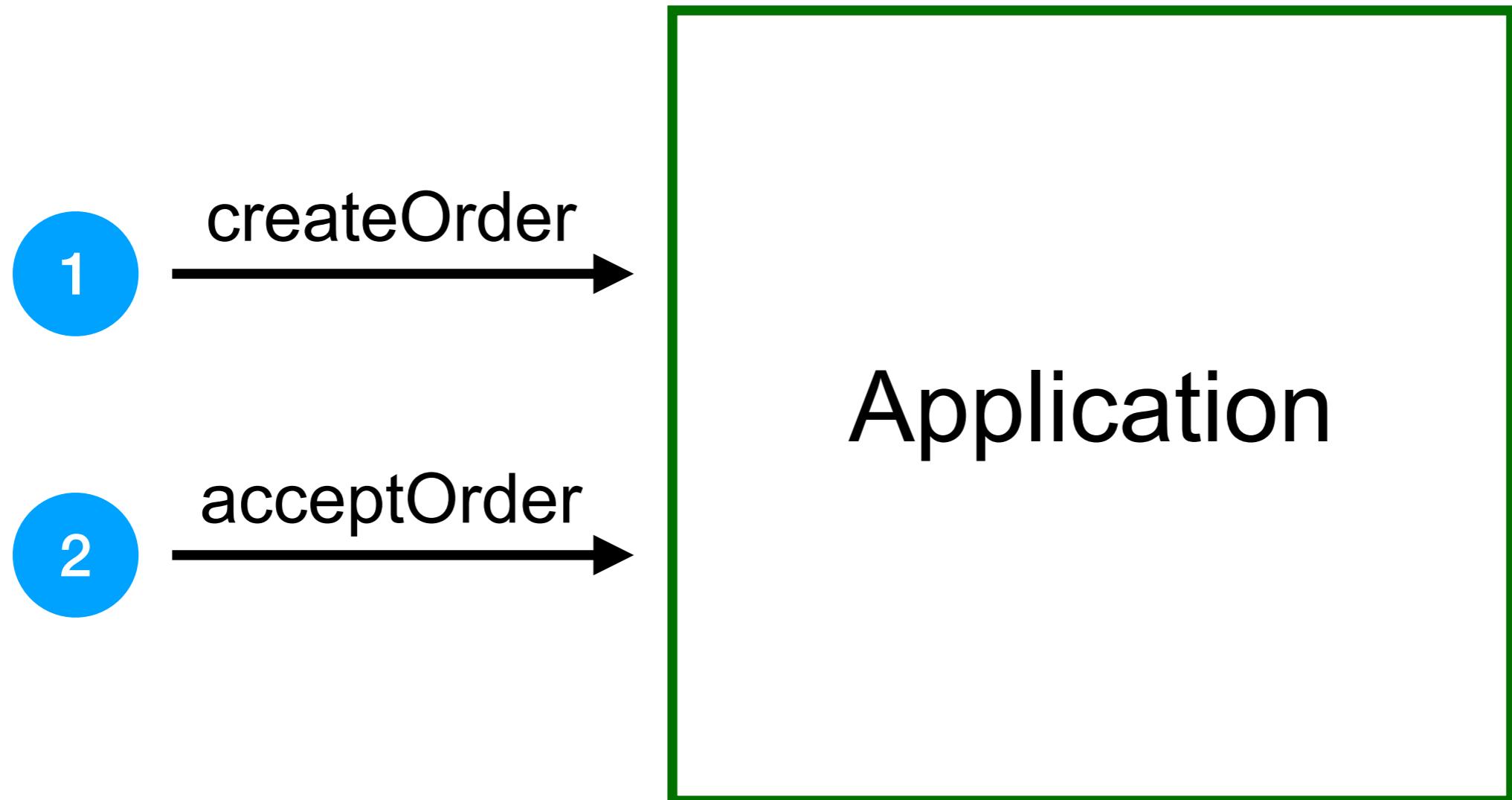


1. Identify system operations

Start with functional requirements
User story



1. Identify system operations



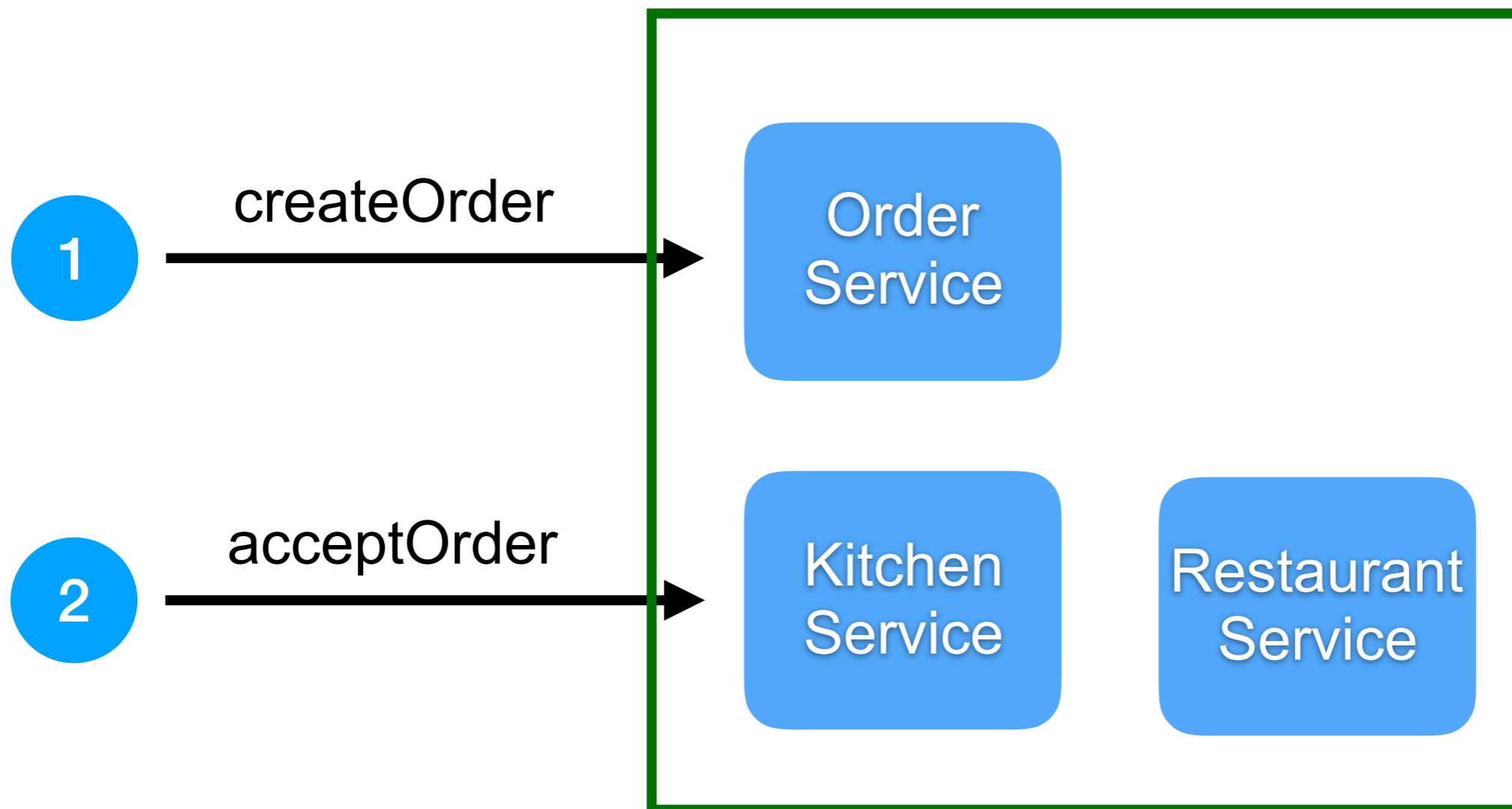
2. Identify services

Try to decomposition into services

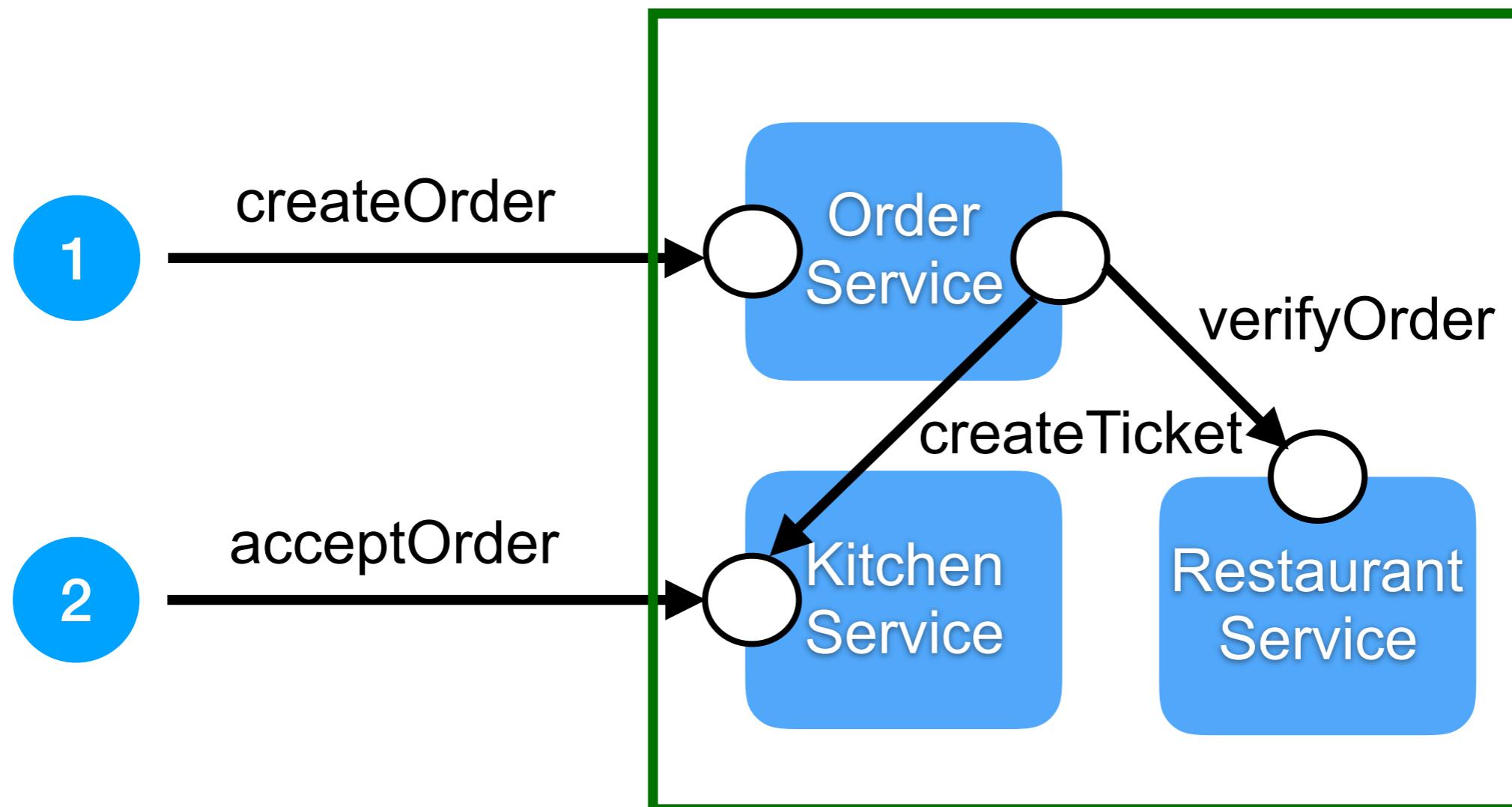
Business > Technical concept
What > How



2. Identify services



3. Define service APIs and collaborations



Problems when decompose services ?

Network latency

Reliability of communication (sync)

Maintain data consistency

God classes



Communication between services



Communication patterns

Communication style

Discovery service

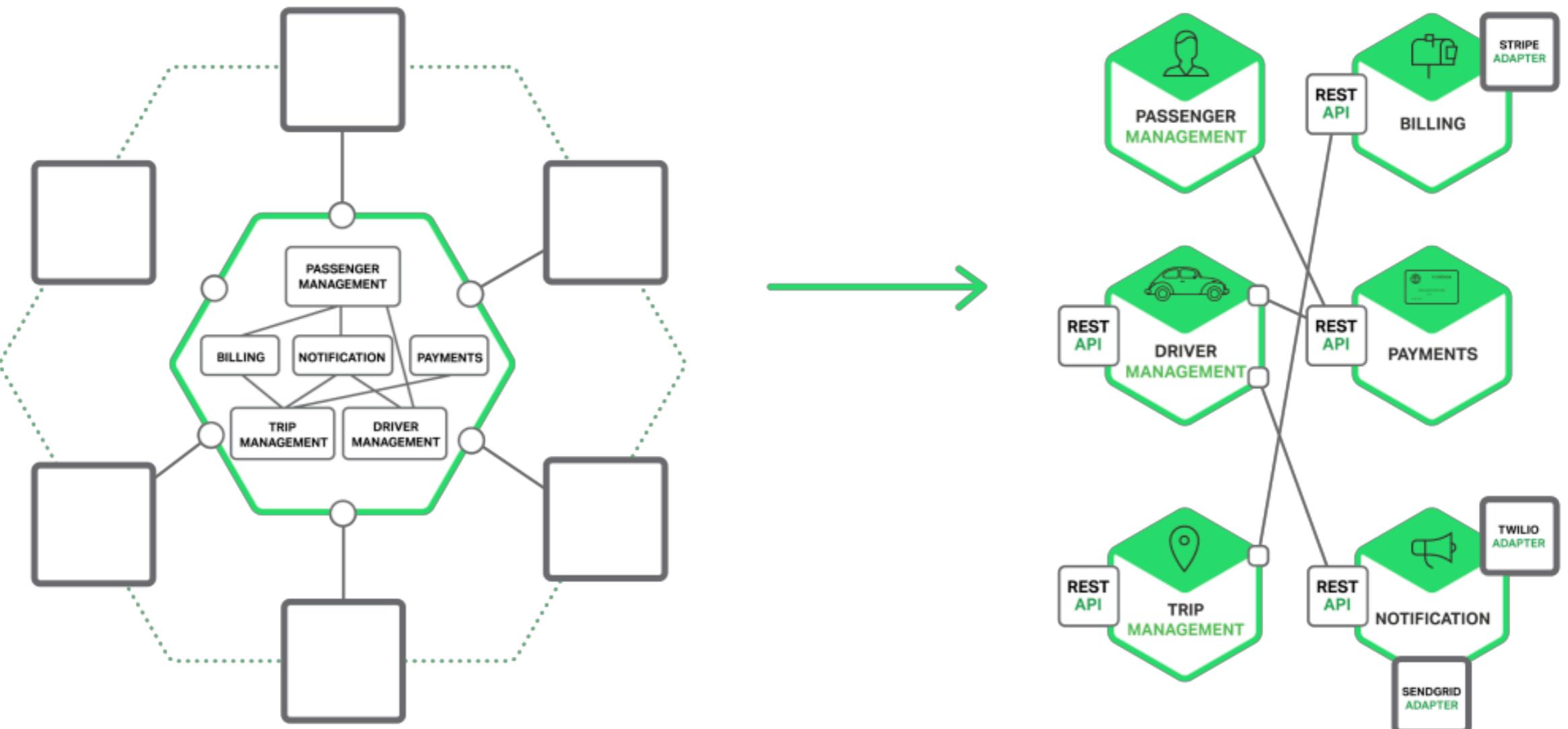
Reliability of communication

Transactional messaging

External API



Inter Process Communication (IPC)



<https://www.nginx.com>

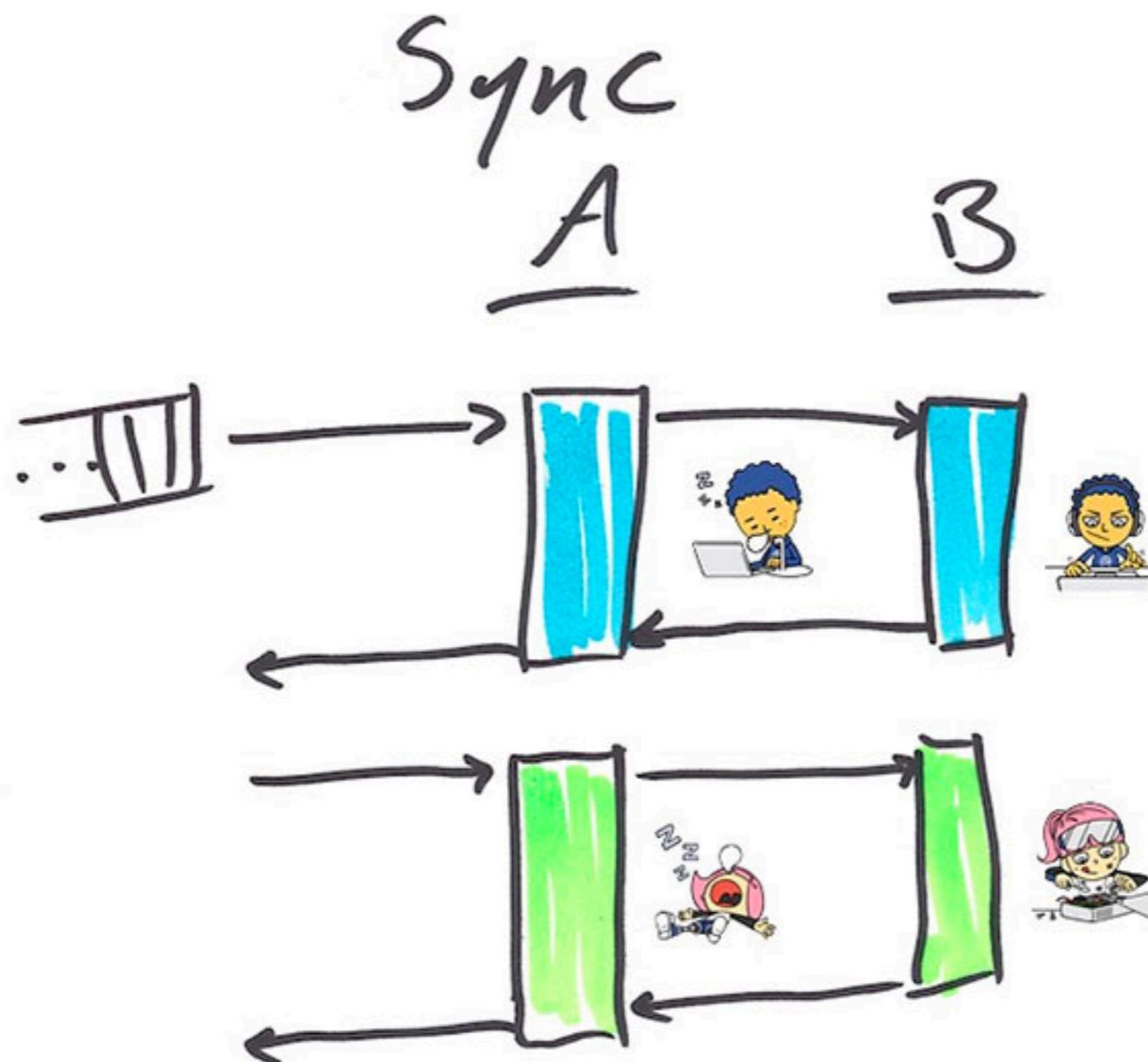


Interaction Styles

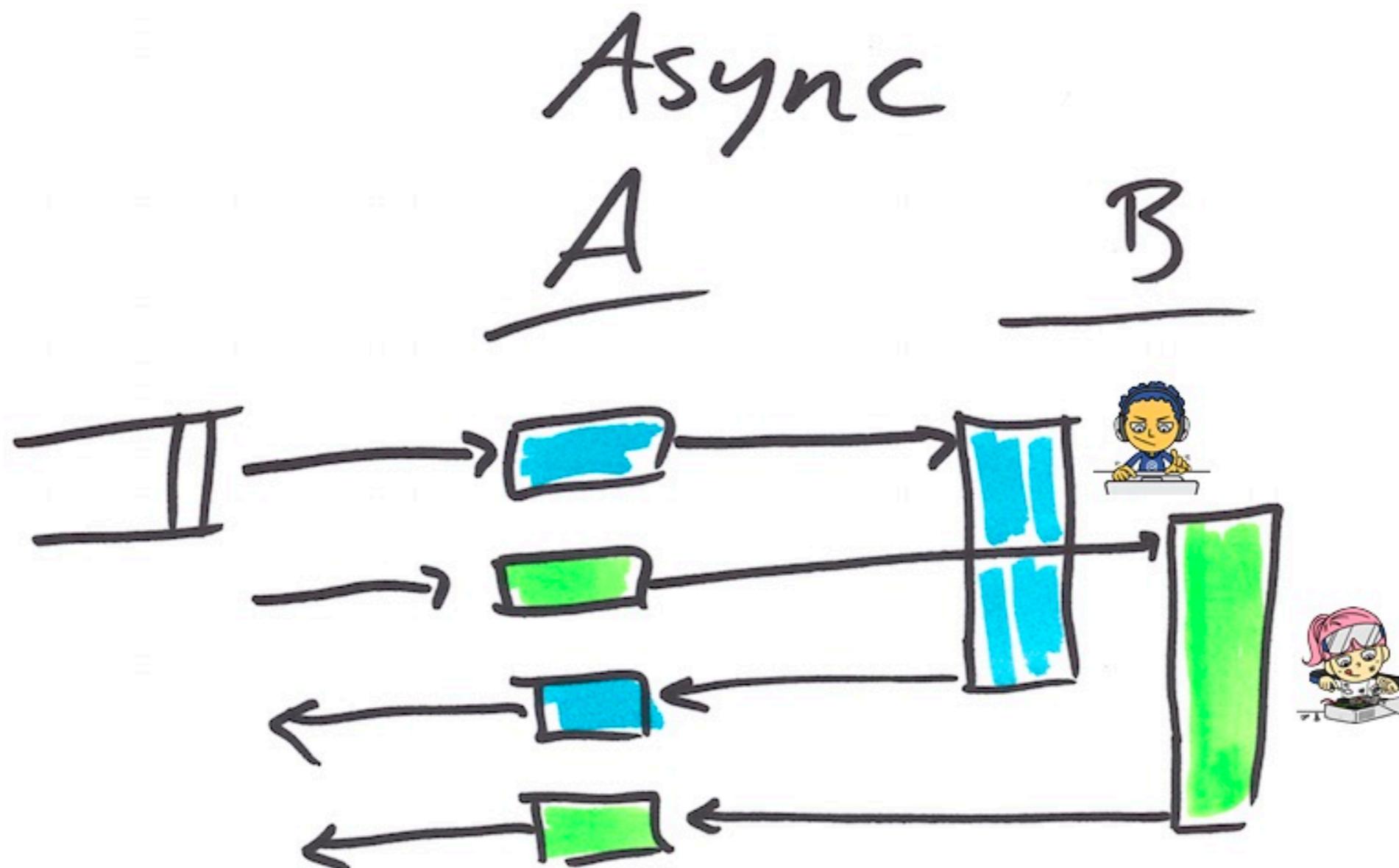
	One-to-one	One-to-many
Synchronous	Request/response	-
Asynchronous	Async request/response	Publish/subscribe
	Notification	Publish/async response



Synchronous



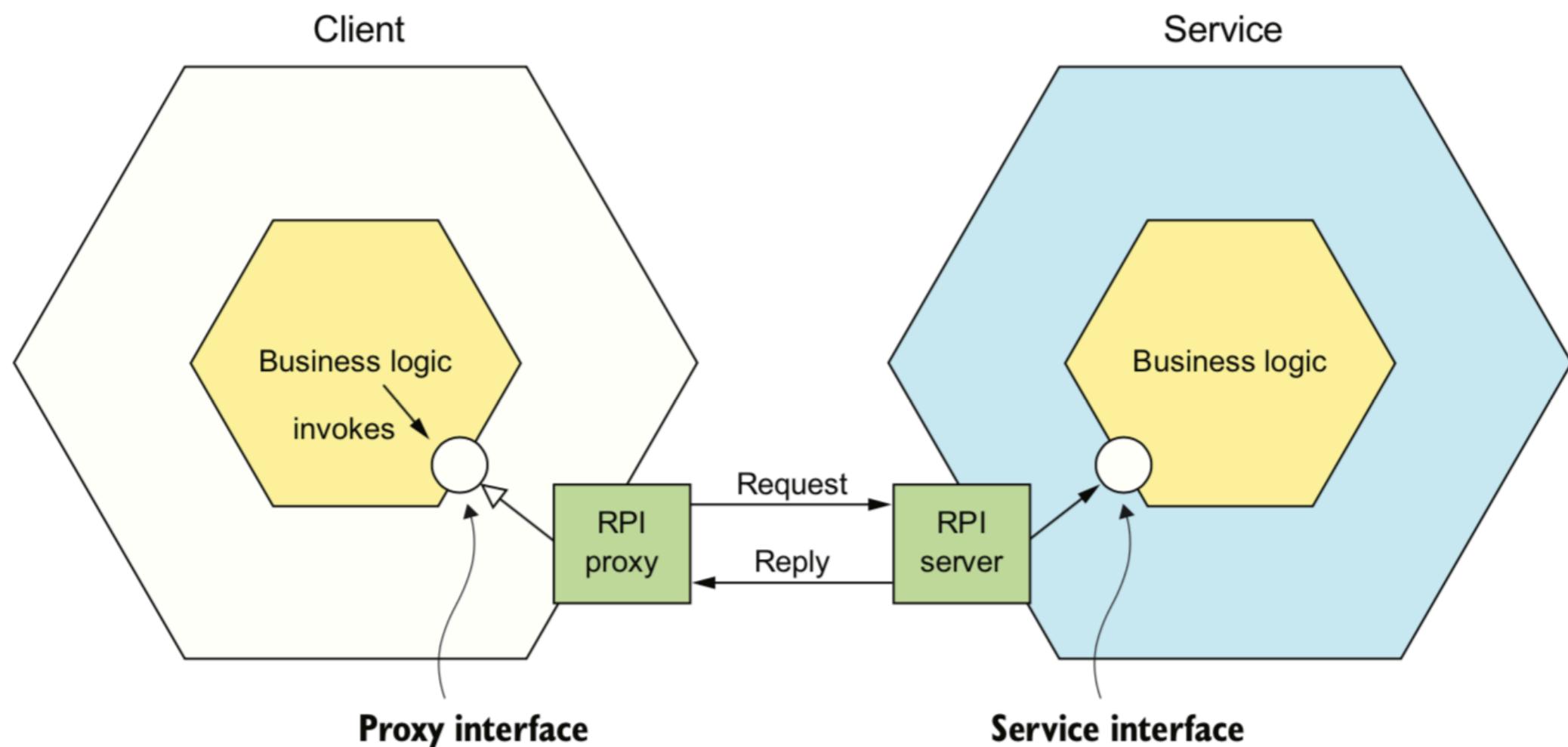
Asynchronous



Synchronous Remote Procedure Invocation



Synchronous Remote Procedure Invocation



Synchronous Remote Procedure Invocation

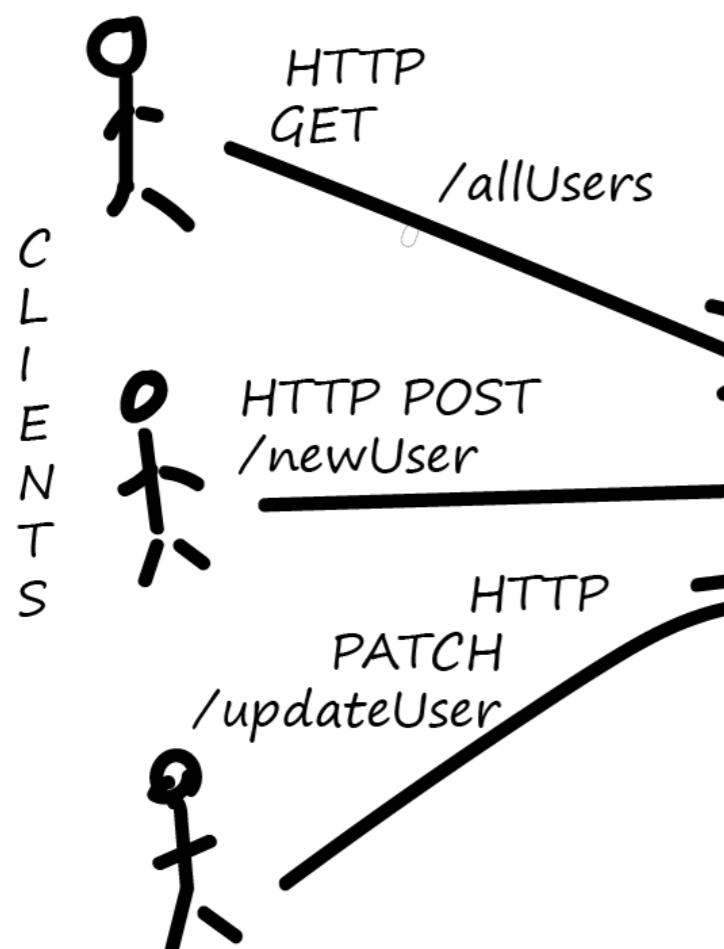
REST
gRPC

Handling failure with Circuit breaker
Service discovery



REpresentational State Transfer

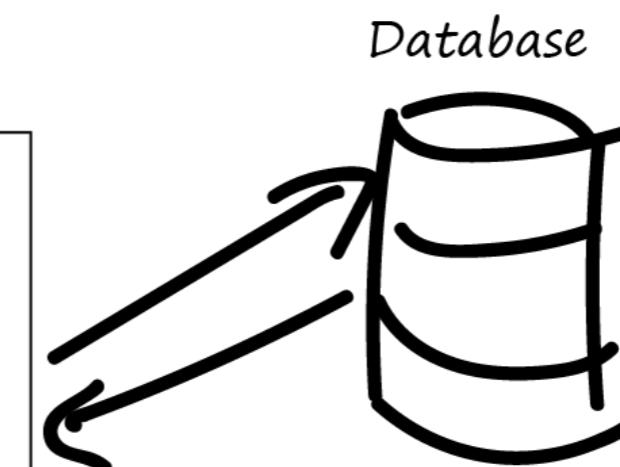
Rest API Basics



Our Clients, send HTTP Requests and wait for responses

Rest API
Receives HTTP requests from Clients and does whatever request needs. i.e create users

Typical HTTP Verbs:
GET → Read from Database
PUT → Update/Replace row in Database
PATCH → Update/Modify row in Database
POST → Create a new record in the database
DELETE → Delete from the database



Our Rest API queries the database for what it needs

Response: When the Rest API has what it needs, it sends back a response to the clients. This would typically be in JSON or XML format.

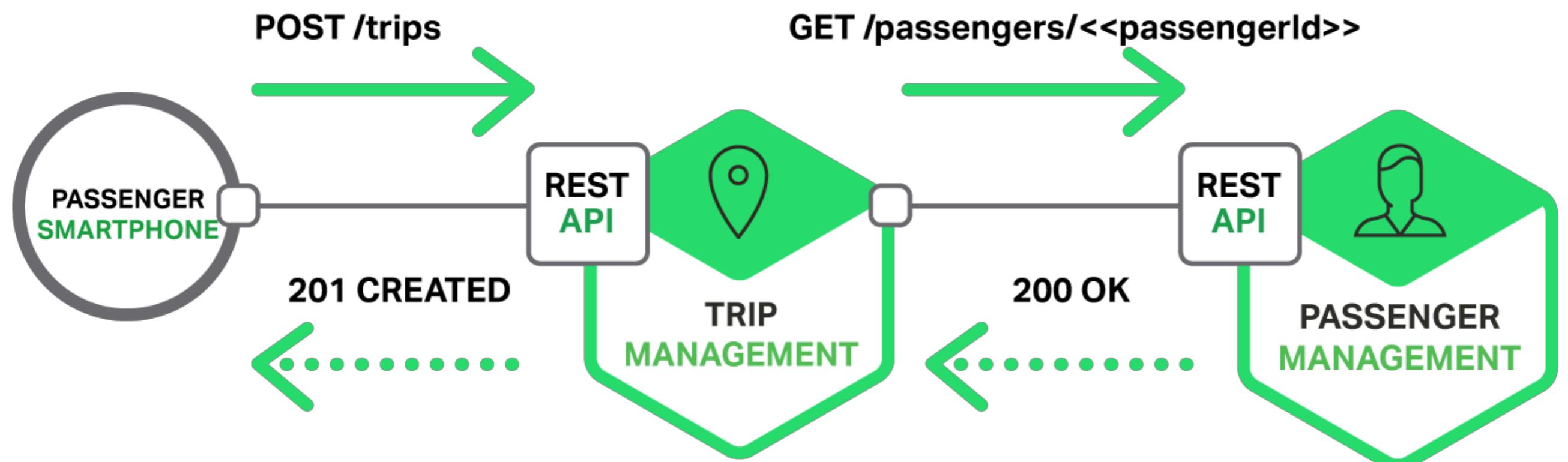


REST :: mapping with HTTP verbs

Activity	HTTP Method
Retrieve data	GET
Create new data	POST
Update data	PUT
Delete data	DELETE



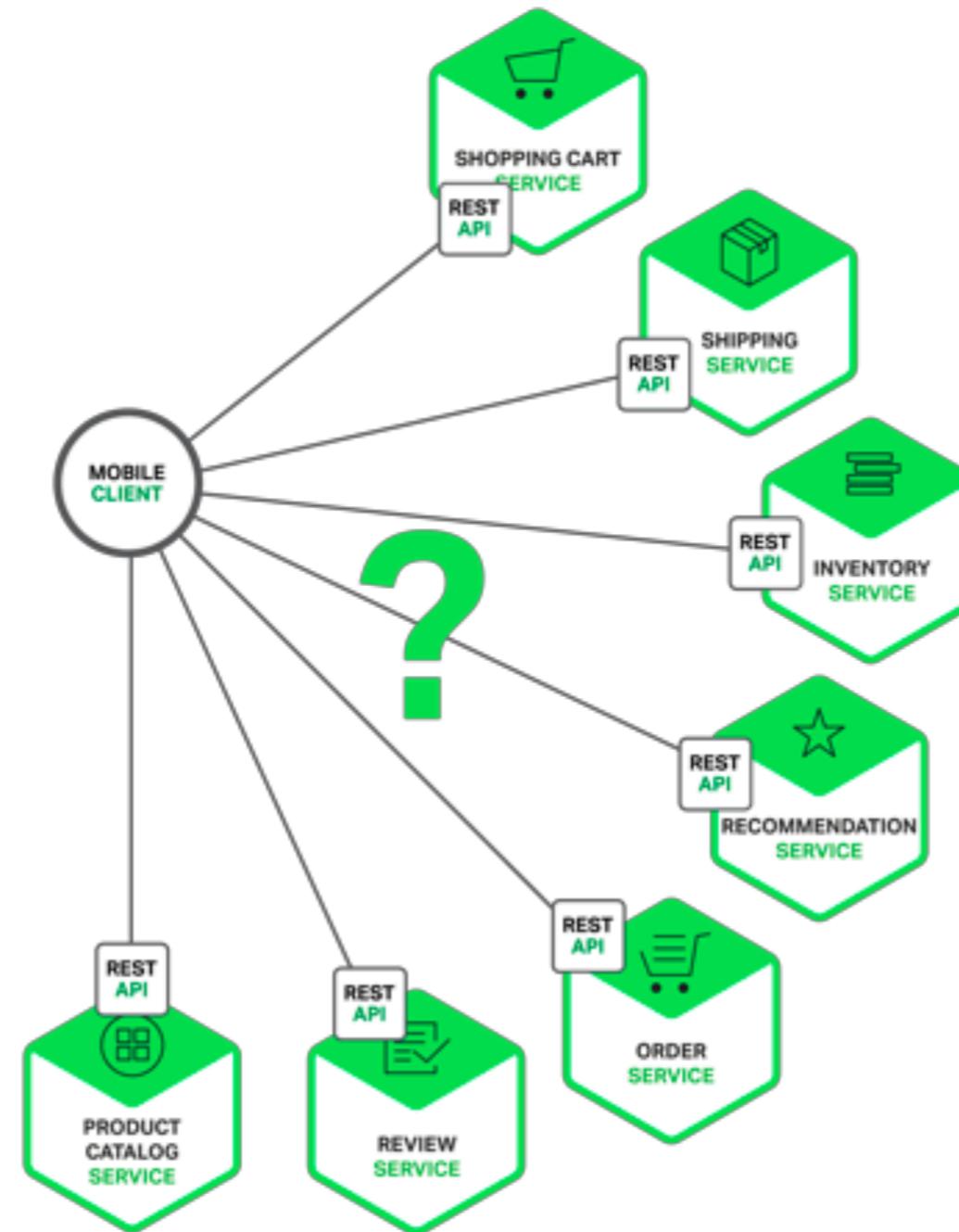
Request and response format



<https://www.nginx.com>



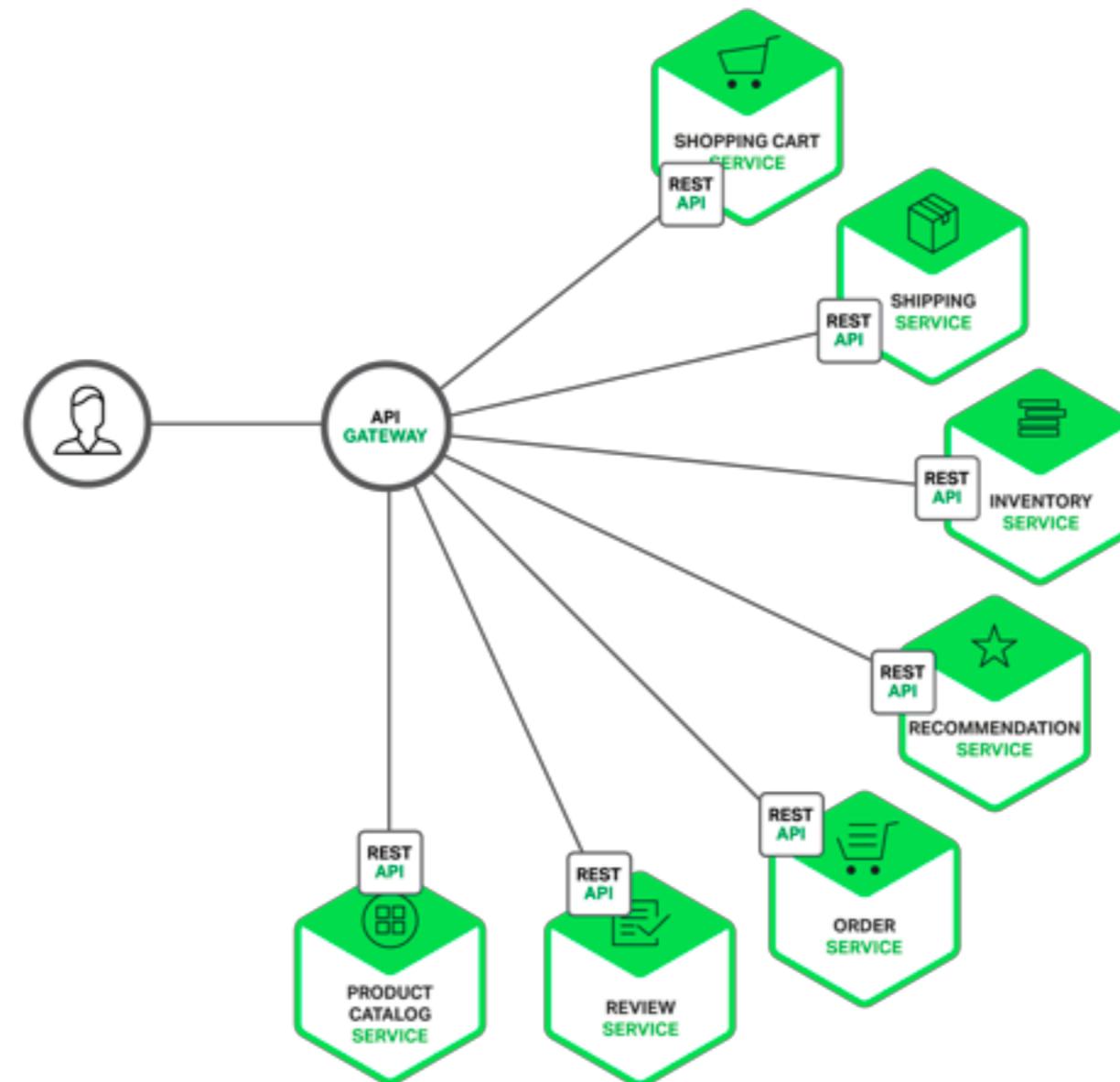
Direct communication !!



<https://www.nginx.com>



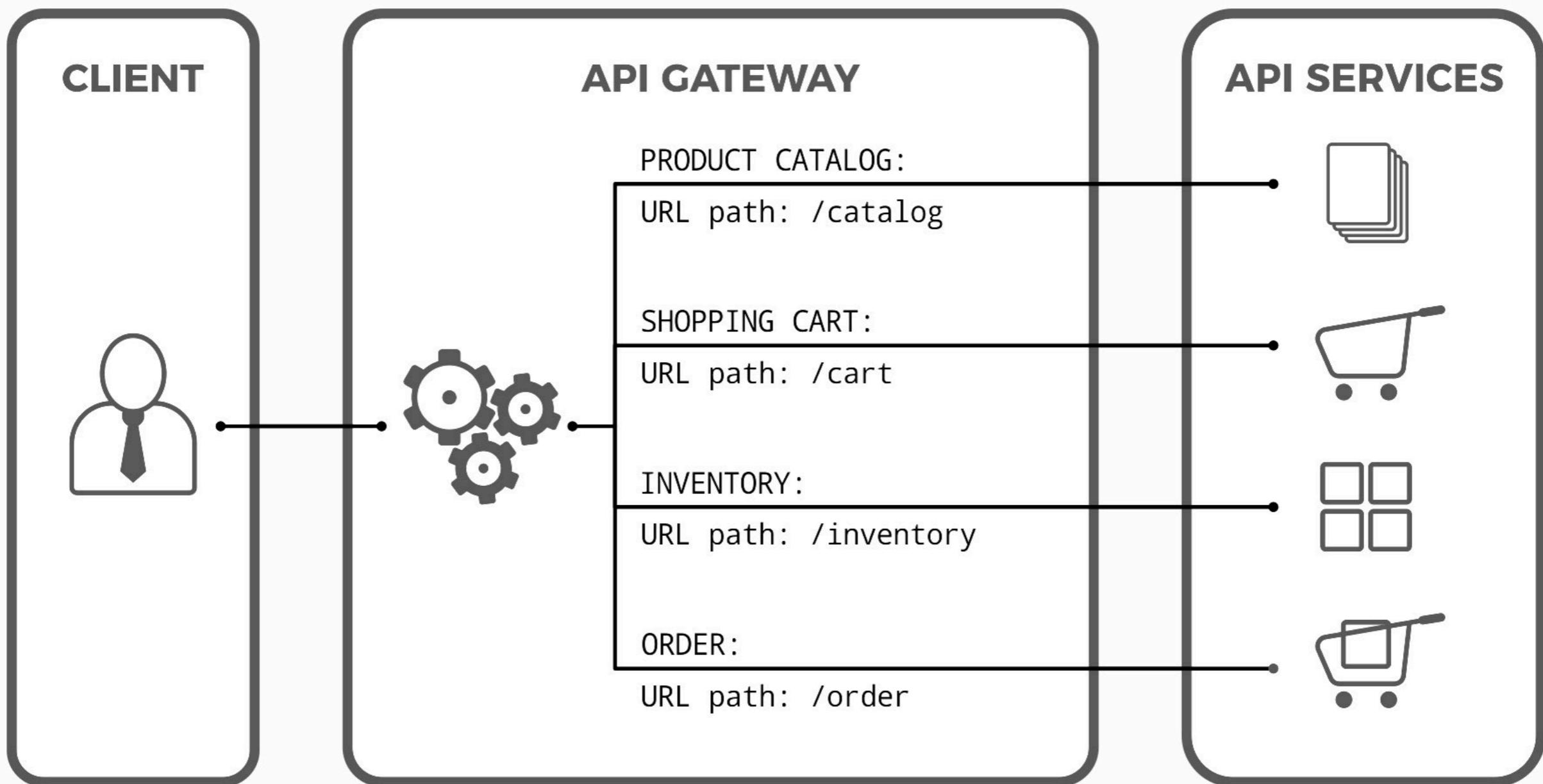
Working with API gateway



<https://www.nginx.com>



Working with API gateway



Functions of API gateway

Authentication

Authorization

Rate limiting

Caching

Metrics collection

Request logging

Load balancing

Circuit breaking



Benefits

Encapsulation interface structure
Client talk to service via gateway
Reduce round trip between service



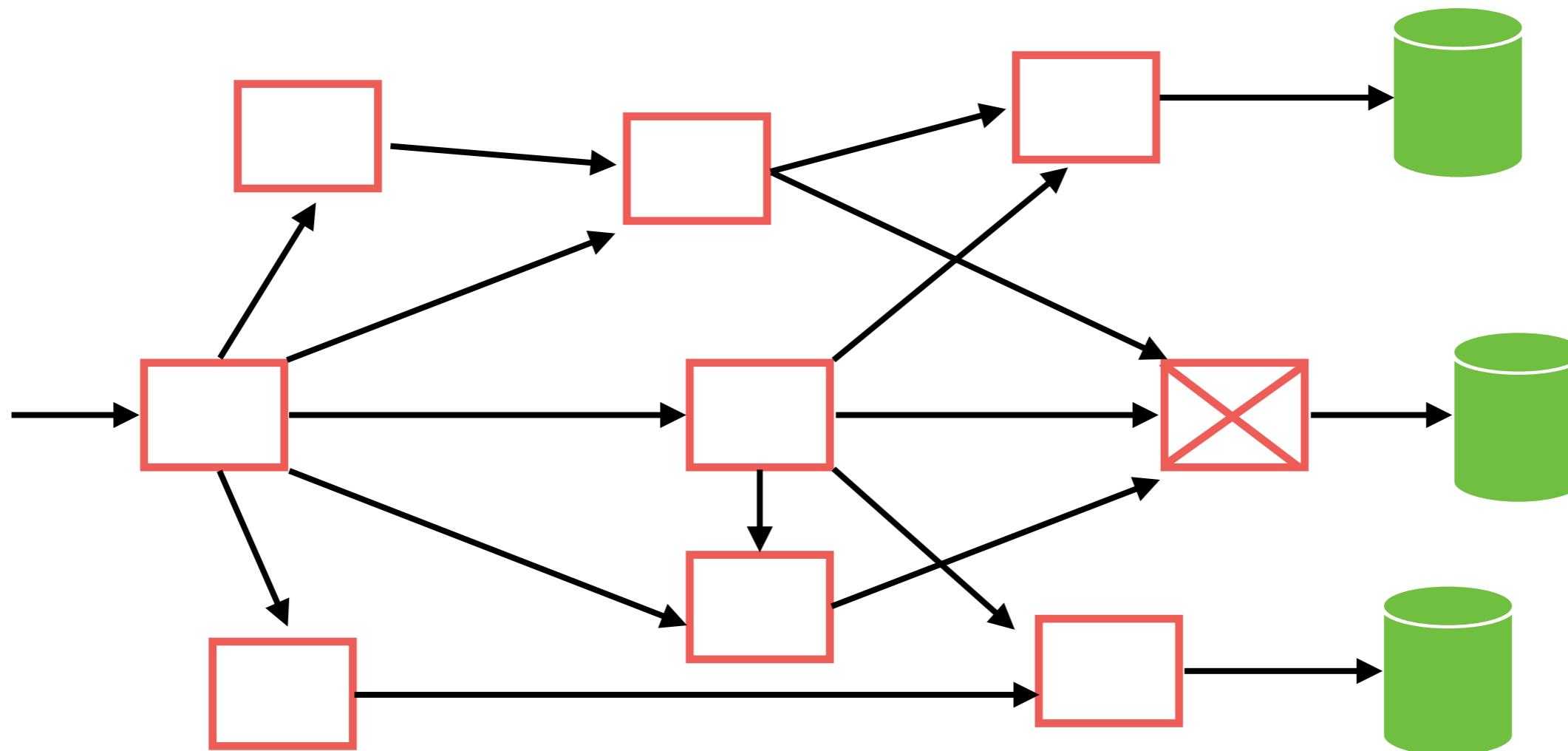
Drawbacks

Development/Performance bottleneck
Single point of failure
Waiting for update data in gateway



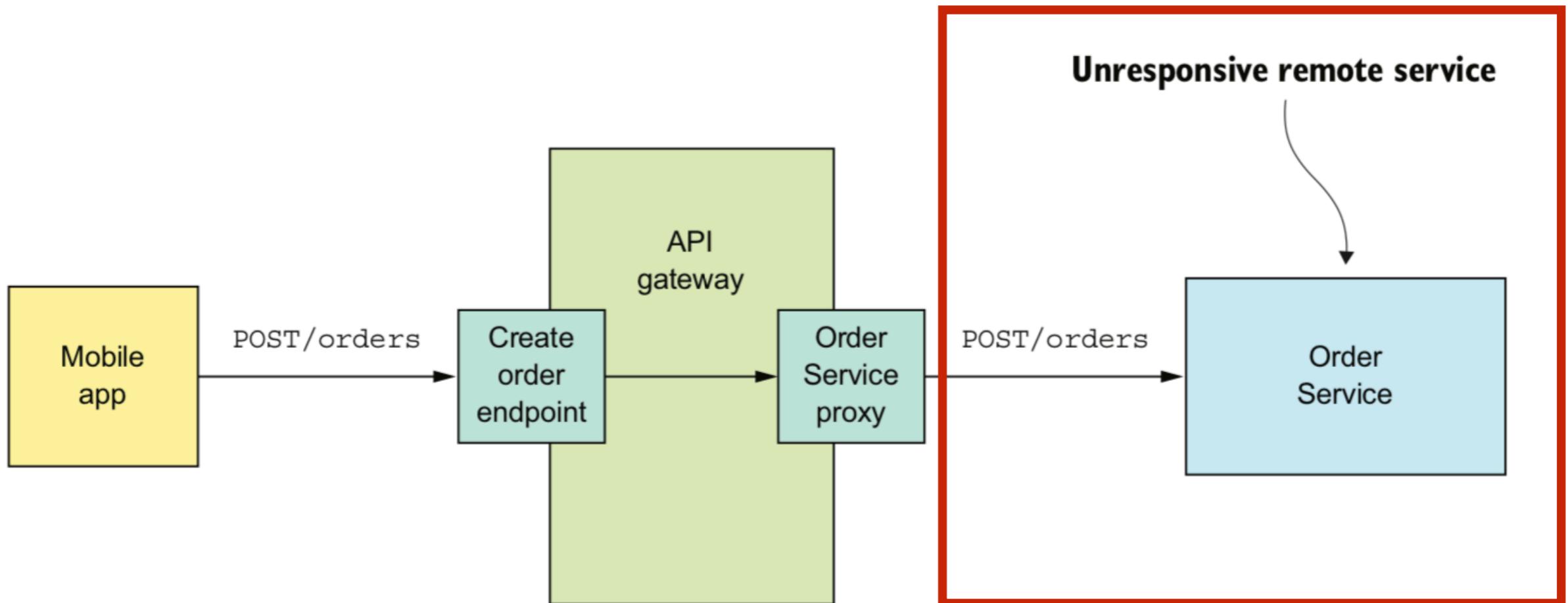
Circuit Breaker pattern

How to handling partial failure ?



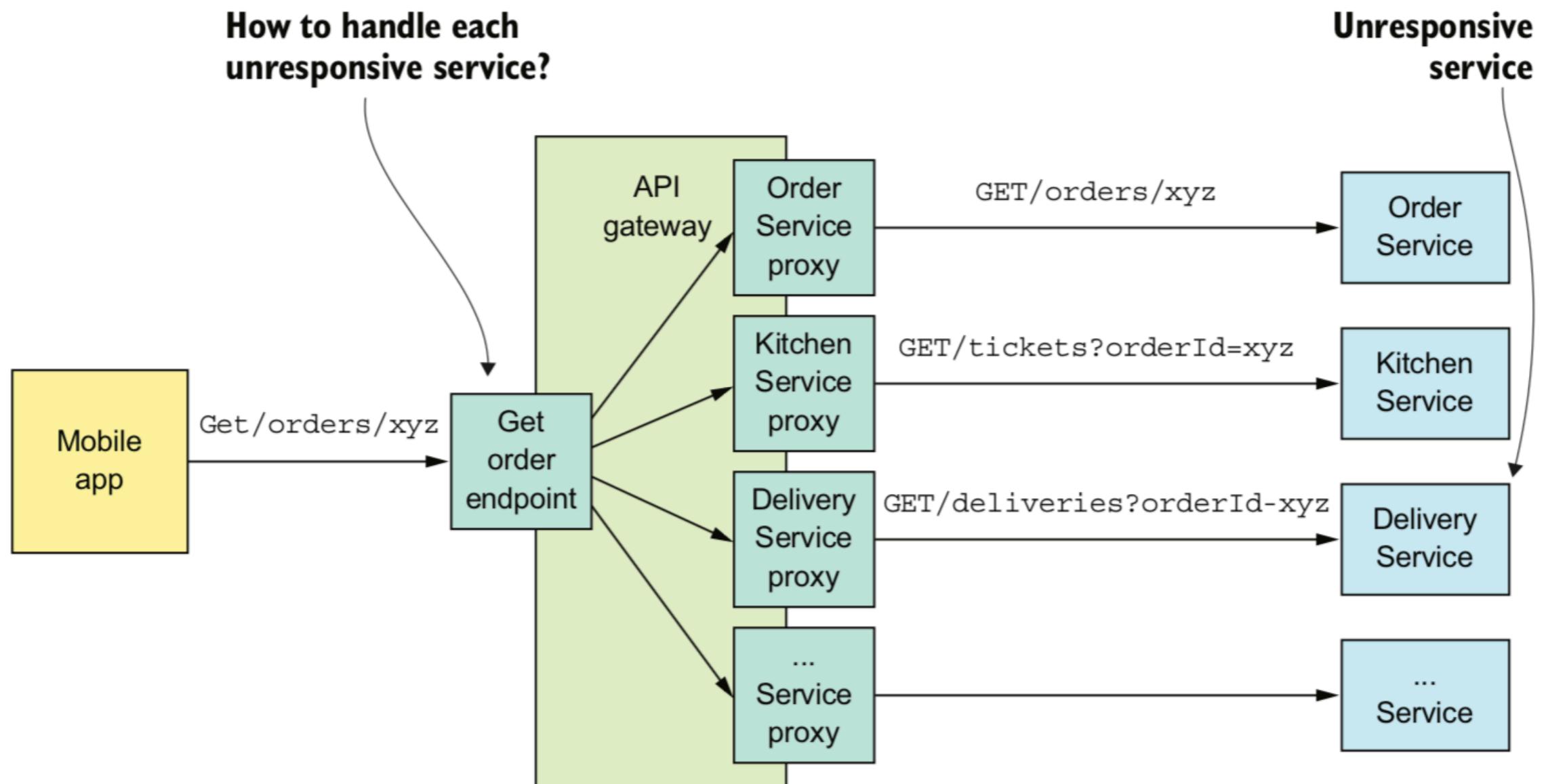
Circuit Breaker pattern

How to handling partial failure ?



Circuit Breaker pattern

How to handling partial failure ?



Develop robust patterns

Network timeout

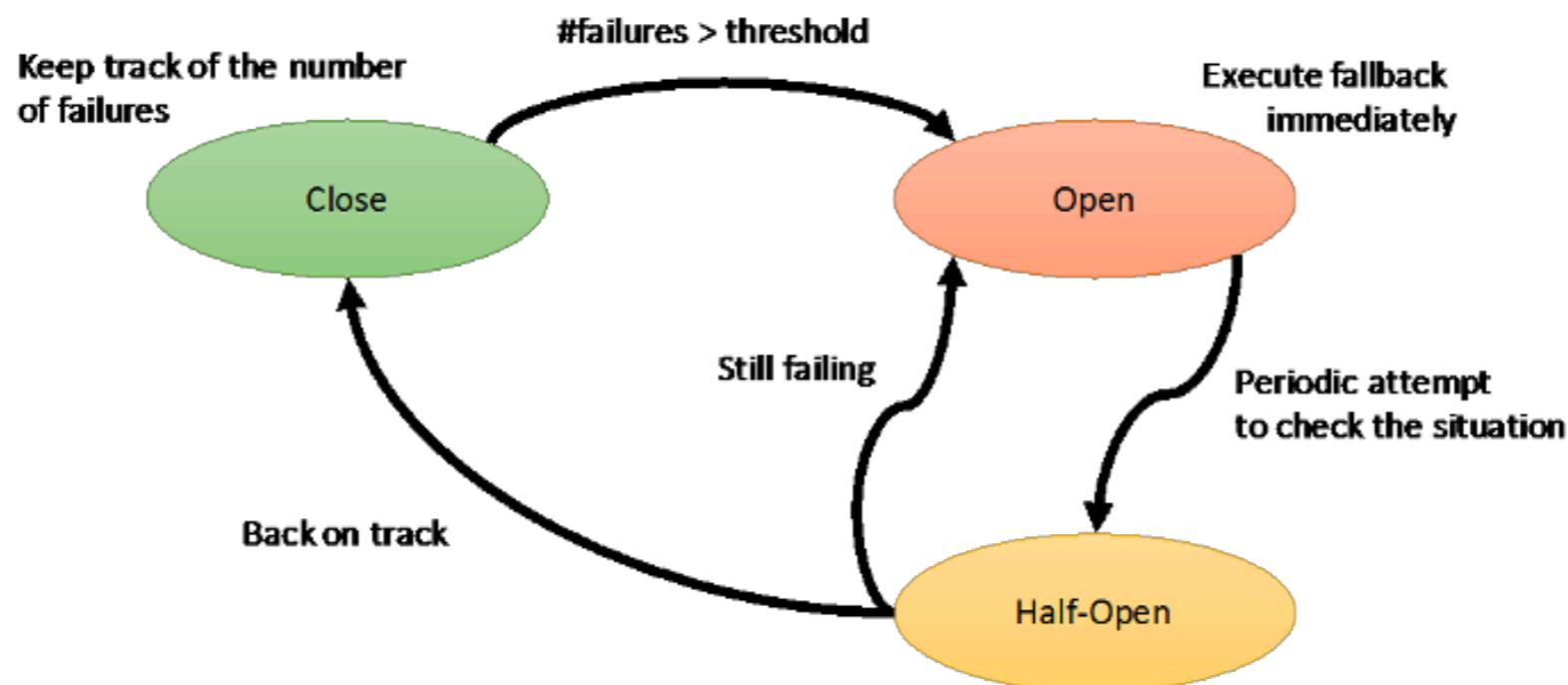
Limit request from client to service
Circuit breaker pattern



Circuit Breaker pattern

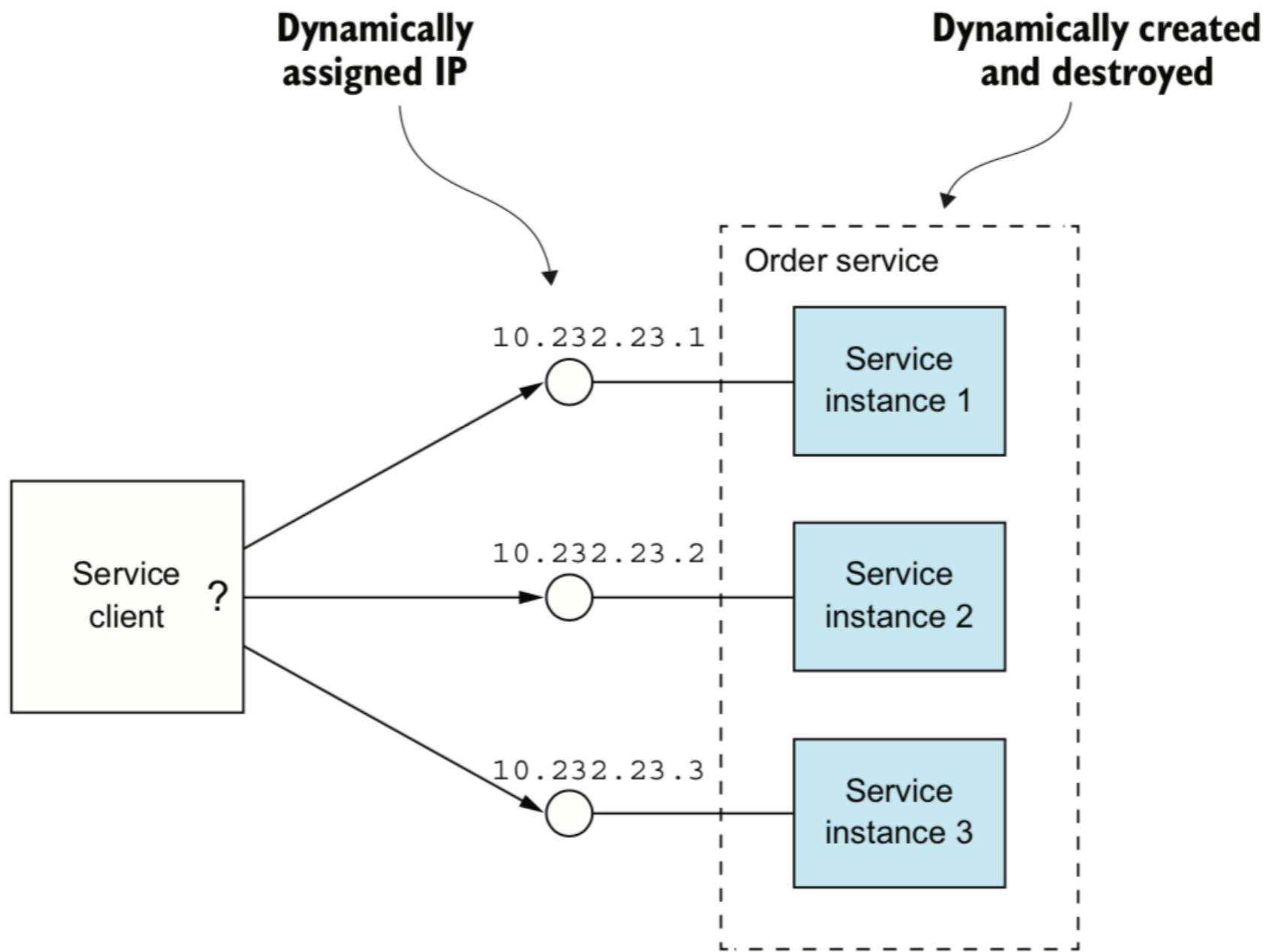
Track the number of success and failure

***If error rate exceed some threshold
then enable circuits breaker***

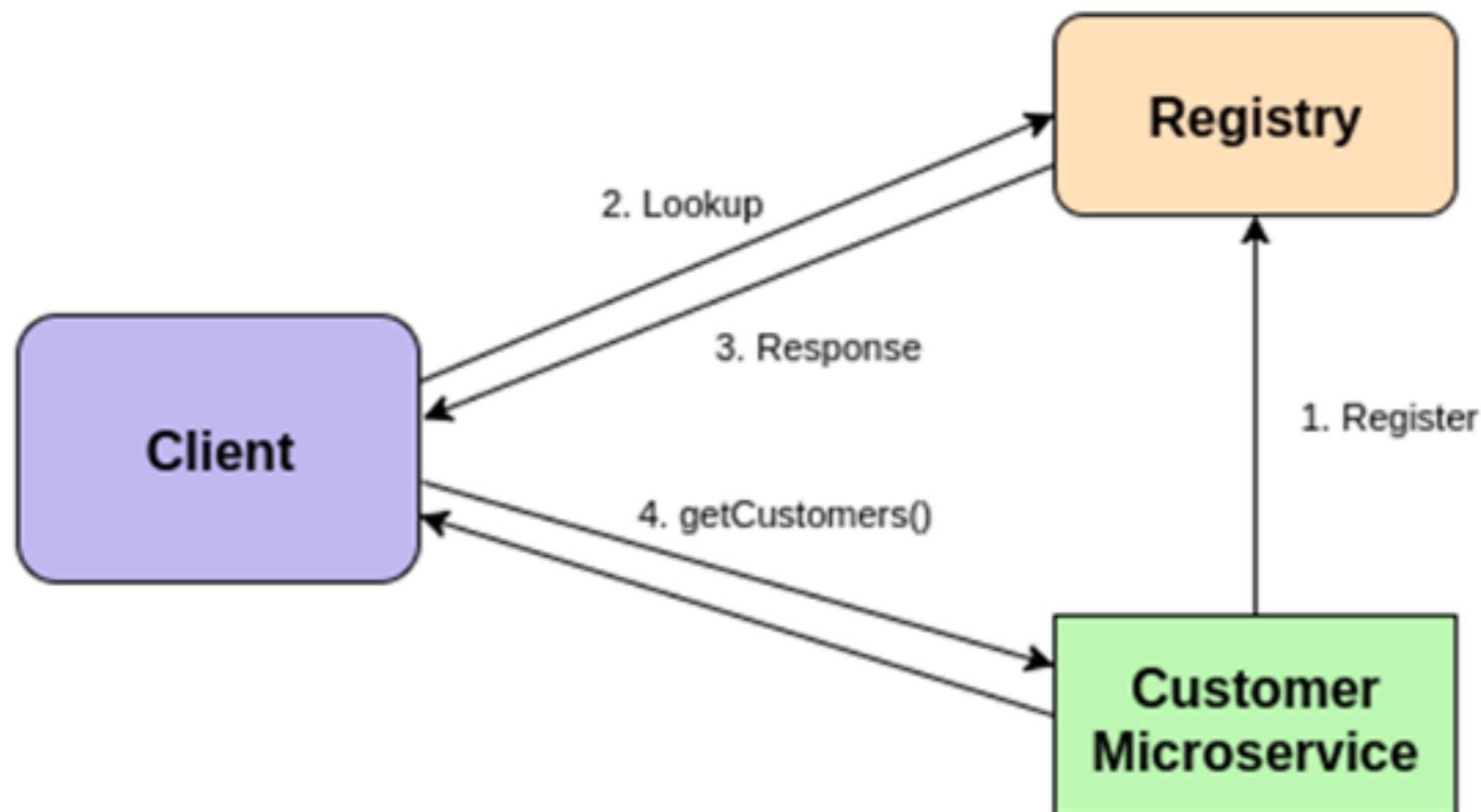


Service Discovery

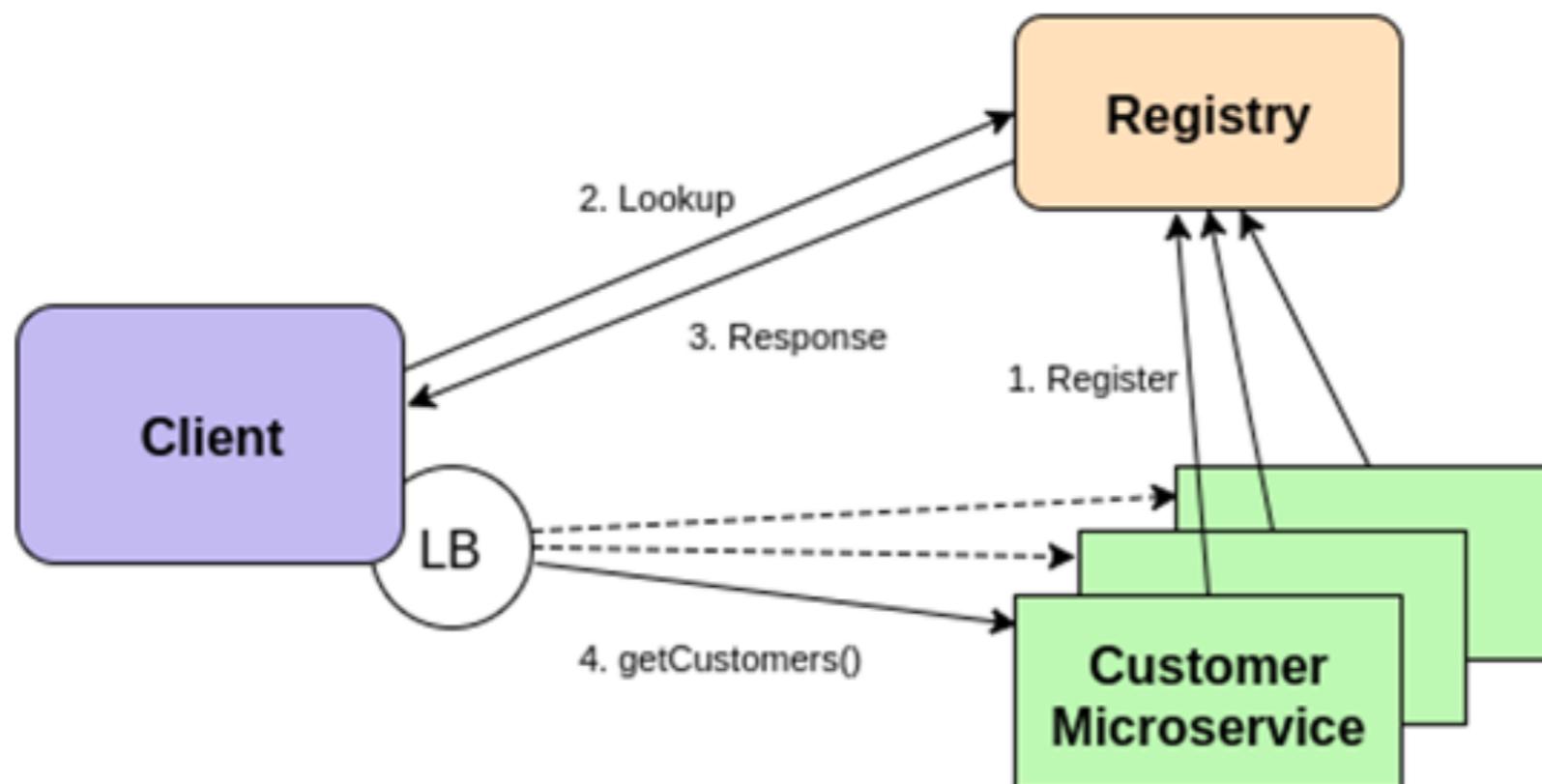
How to call another services ?



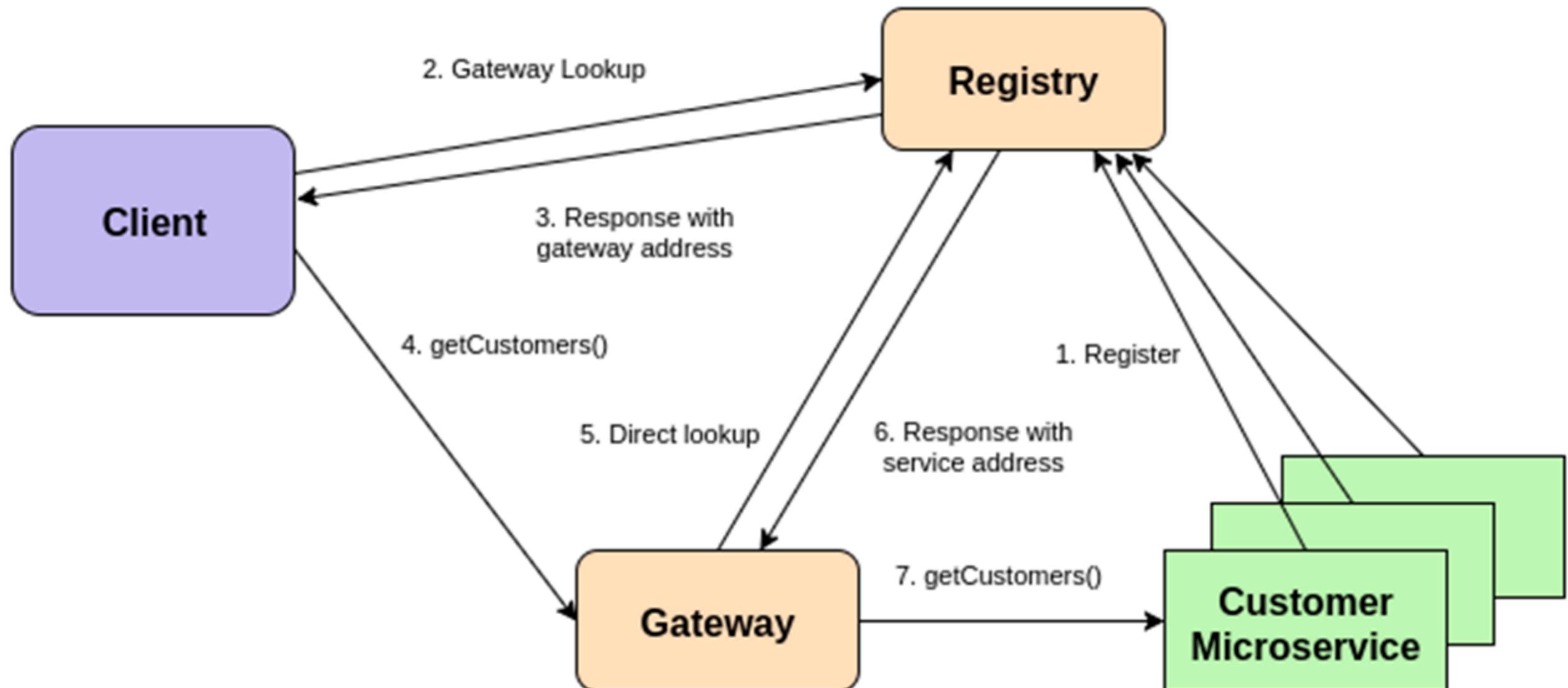
Service Discovery



Service discovery with Load balance



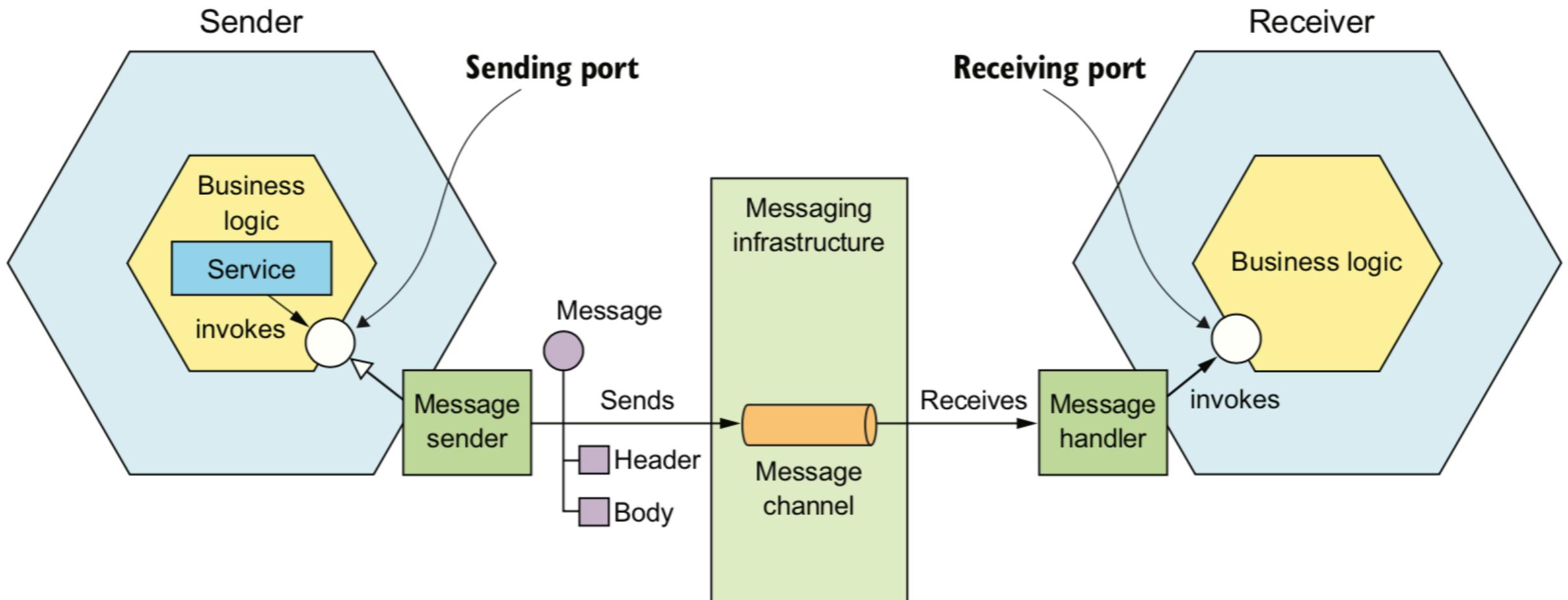
Service discovery with API gateway



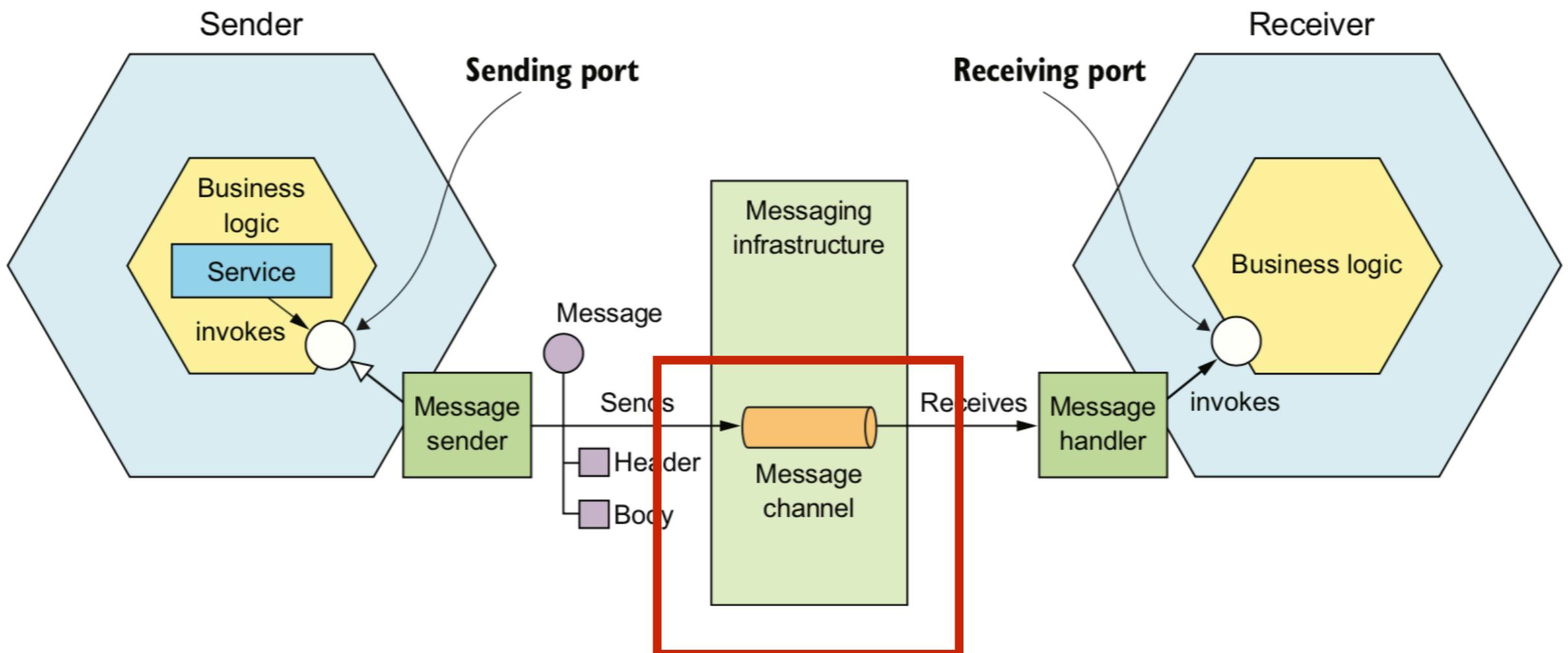
Asynchronous messaging pattern



Asynchronous messaging pattern

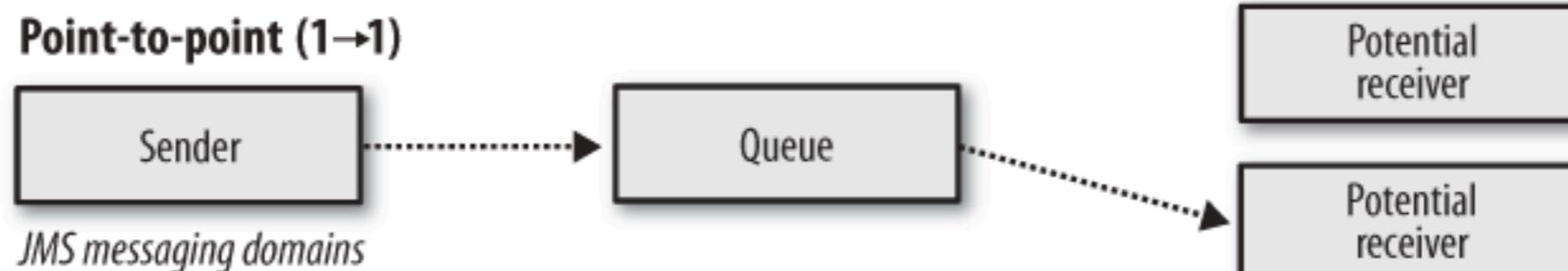


Asynchronous messaging pattern



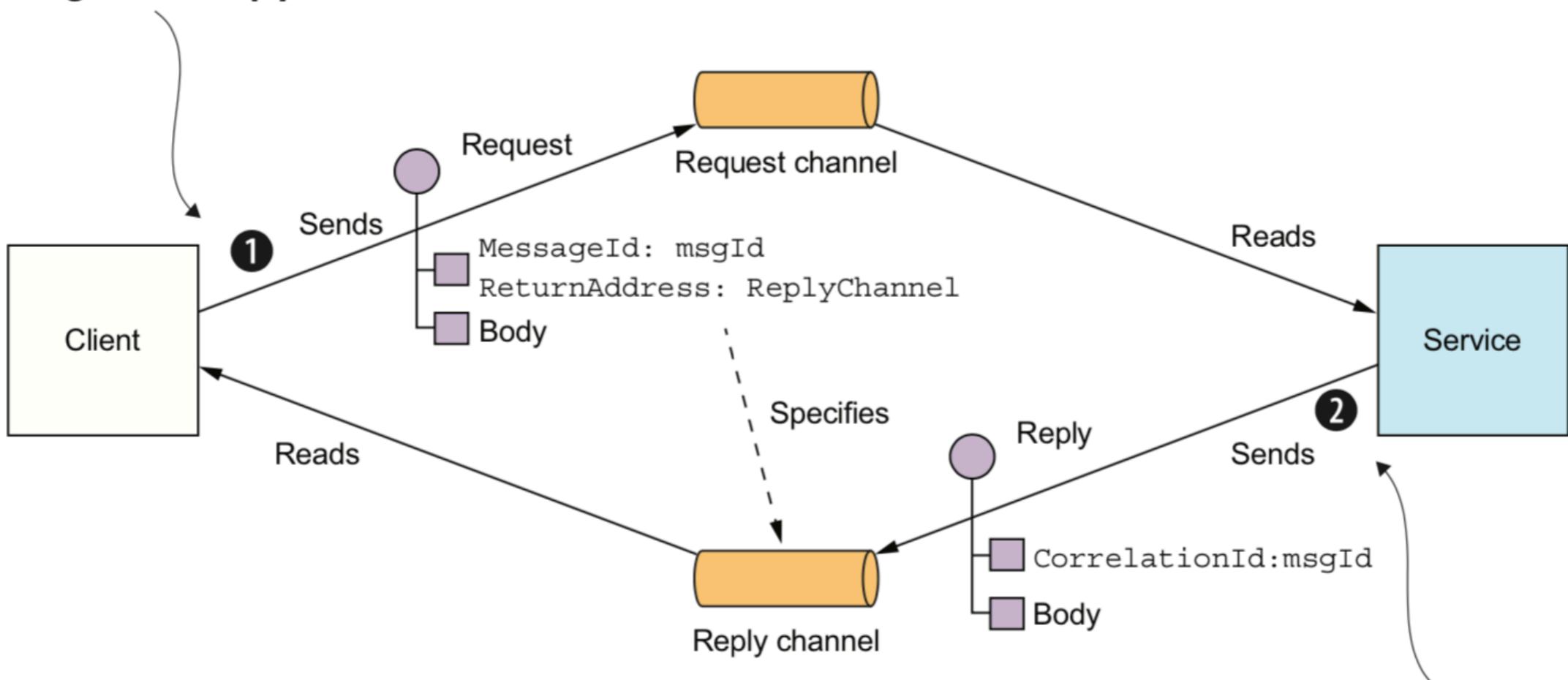
Messaging channels

Point-to-point Publish-subscribe



Async request/response (1)

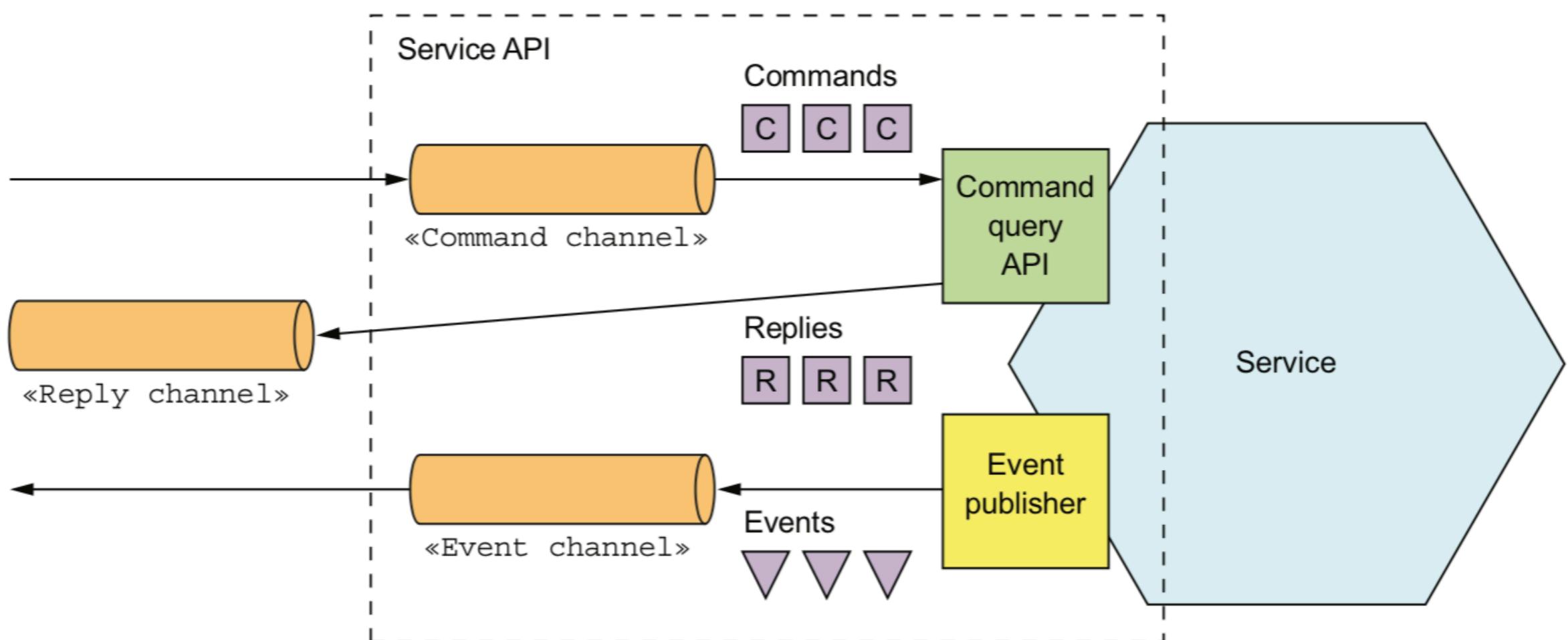
Client sends message containing msgId and a reply channel.



Service sends reply to the specified reply channel. The reply contains a correlationId, which is the request's msgId.

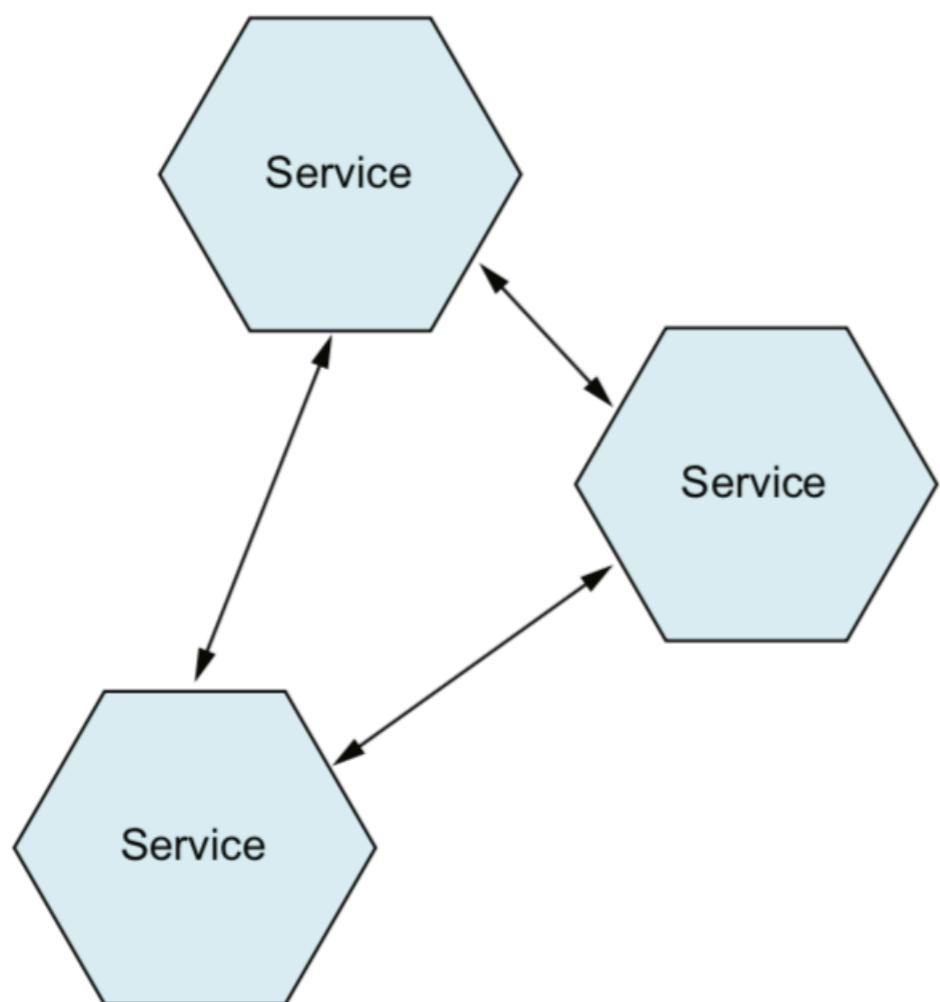


Async request/response (2)

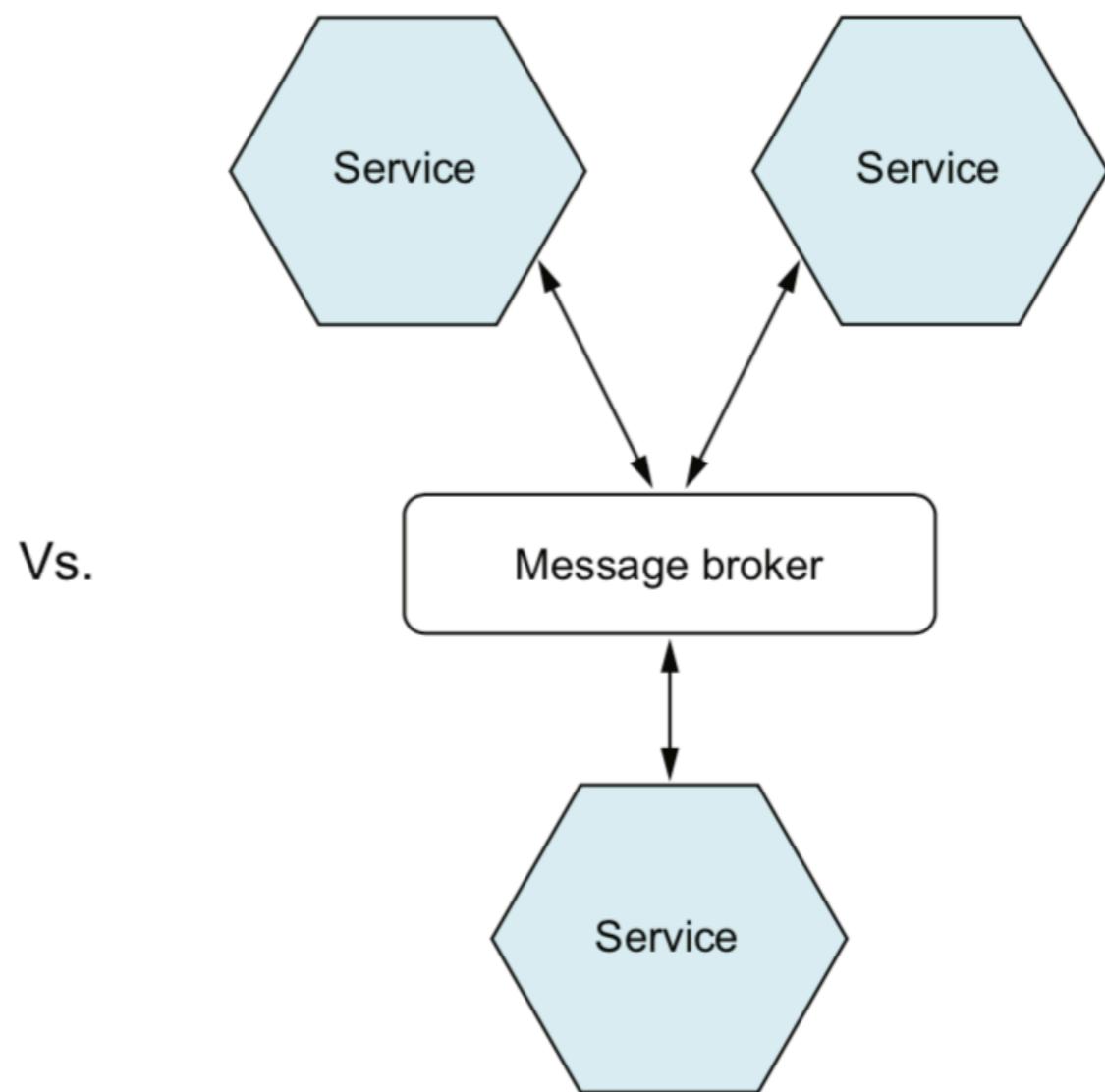


Messaging-based use message broker

Brokerless architecture



Broker-based architecture



Vs.



Message brokers



Benefits

Loose-coupling
Message buffer
Flexible communication



Drawbacks

Performance bottleneck
Single point of failure
Operational complexity

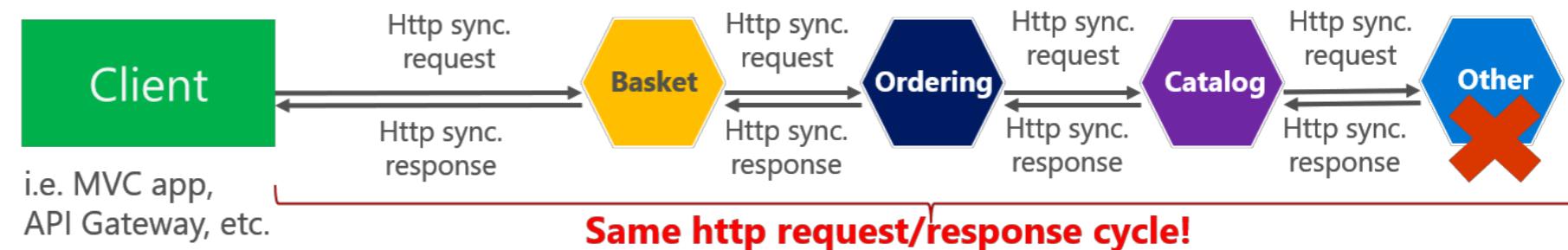


Communication

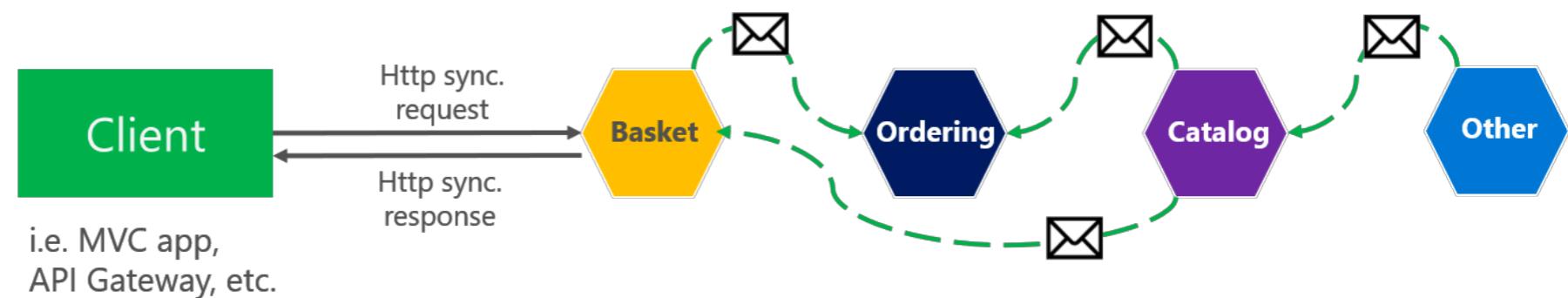
Synchronous vs. async communication across microservices

Anti-pattern

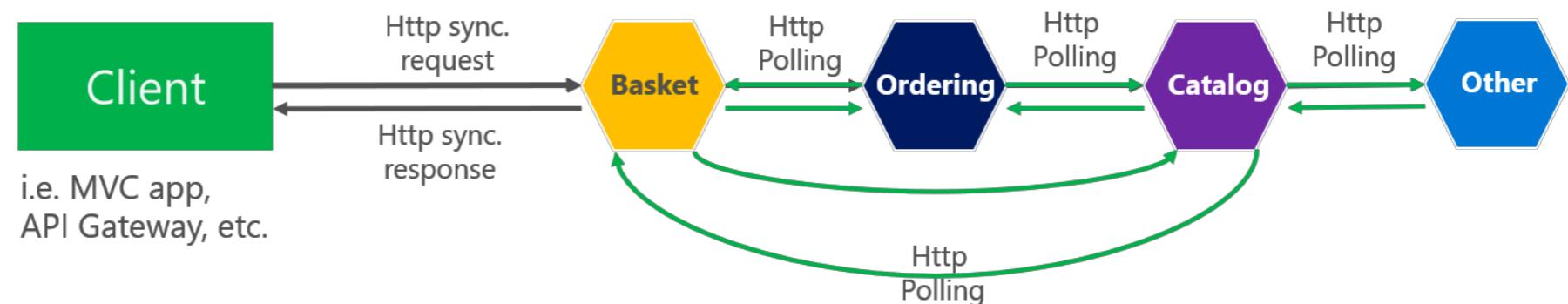
Synchronous
all req./resp. cycle



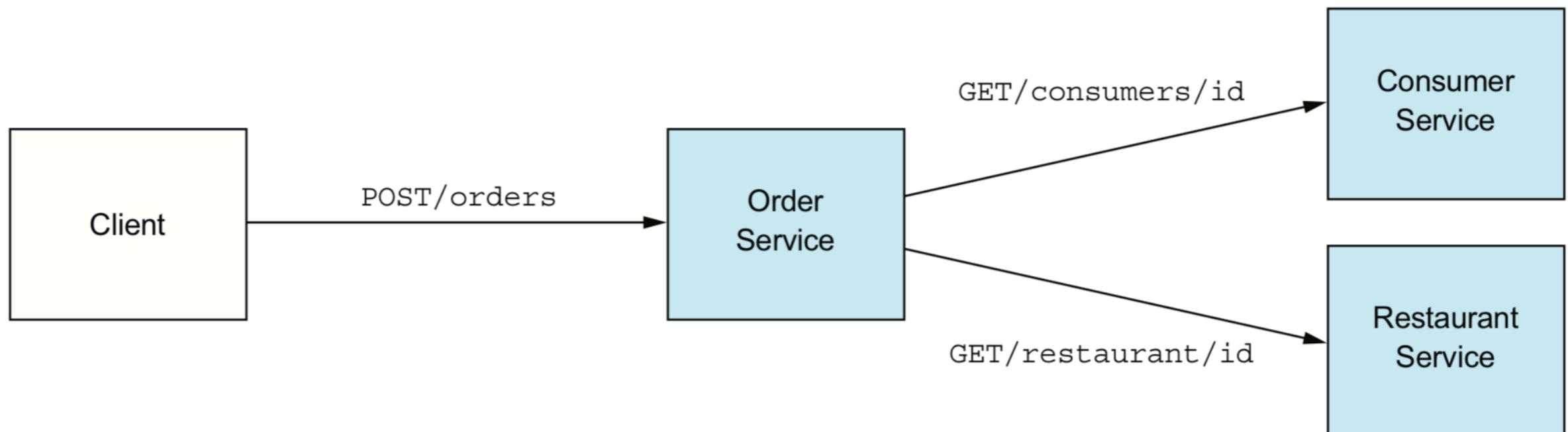
Asynchronous
Comm. across
internal microservices
(EventBus: i.e. **AMPQ**)



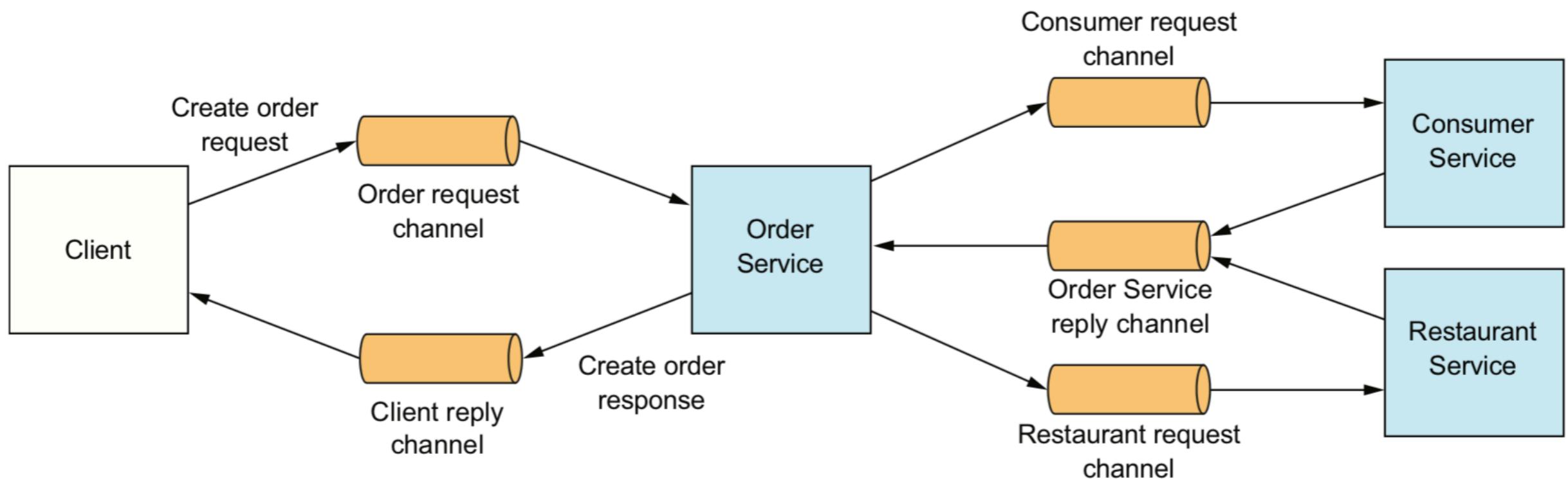
"Asynchronous"
Comm. across
internal microservices
(Polling: **Http**)



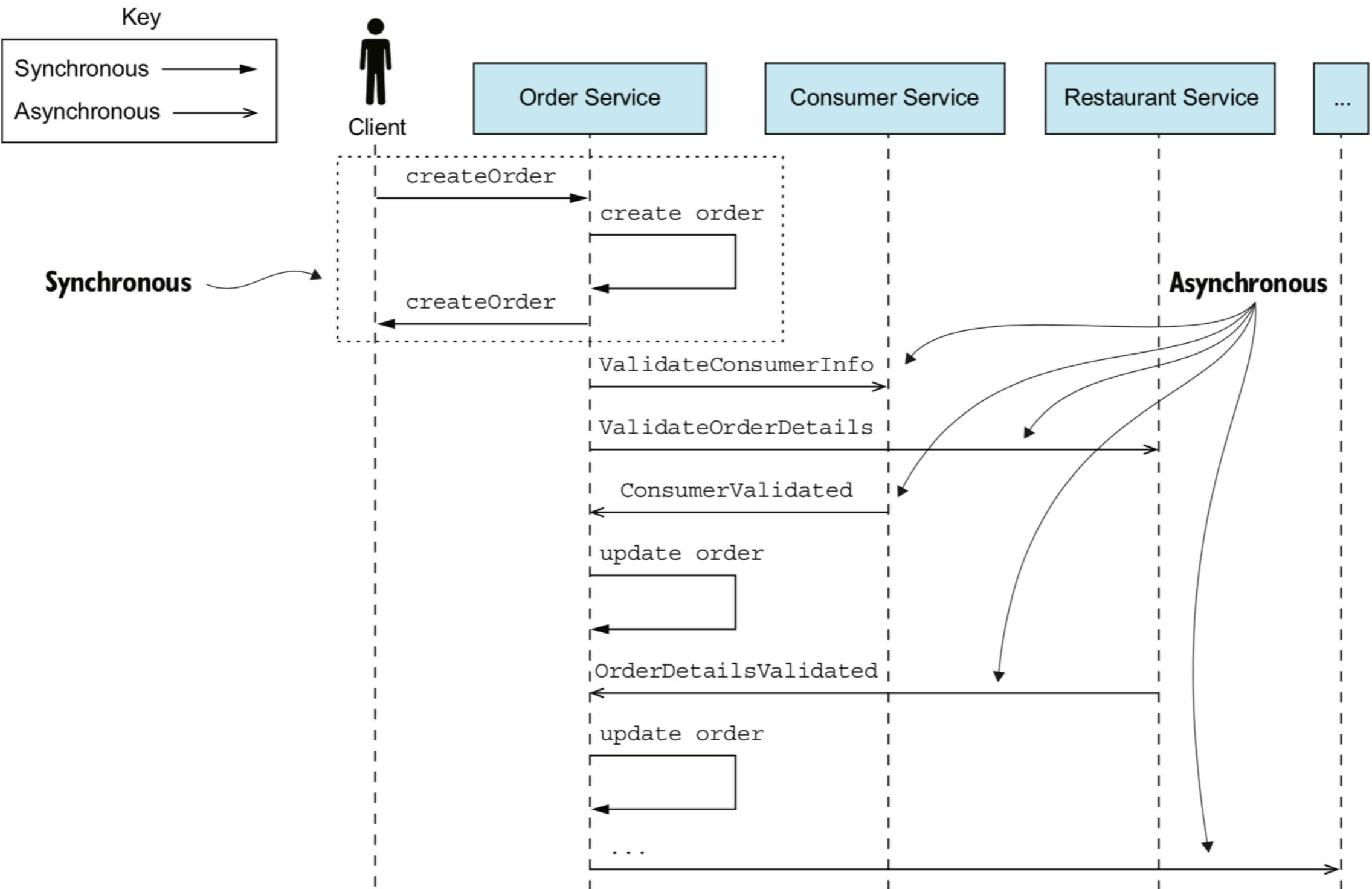
Synchronous reduce availability



Replace with asynchronous (1)



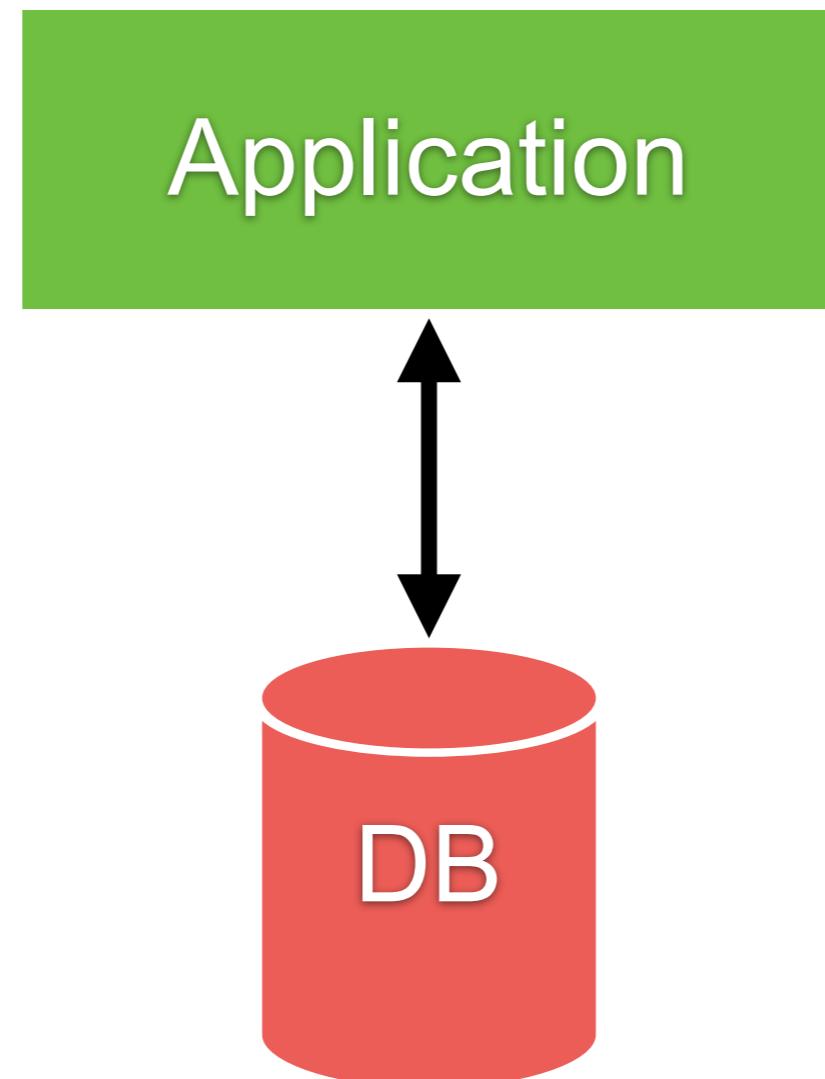
Replace with asynchronous (2)



Manage data consistency

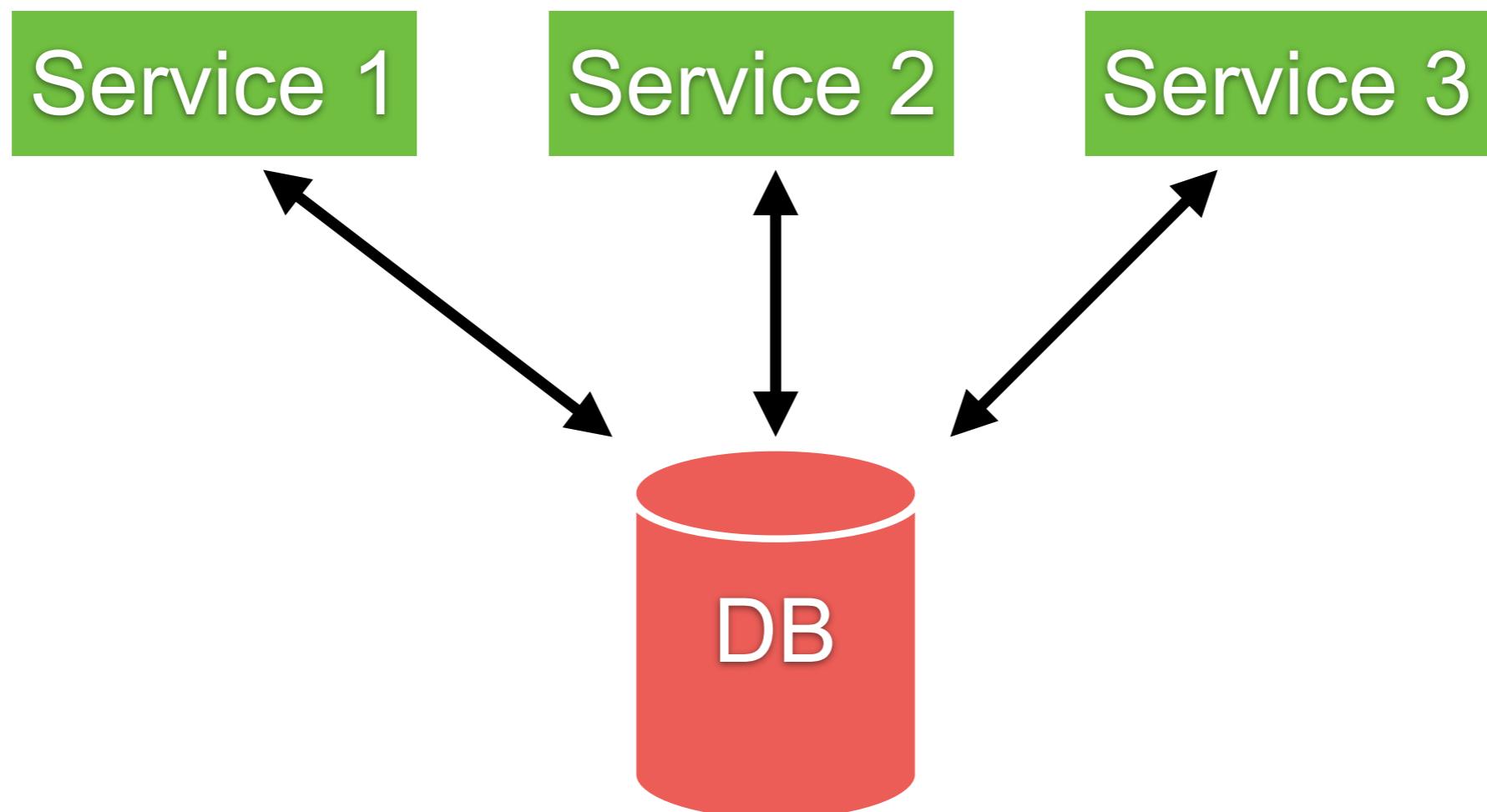


Monolithic



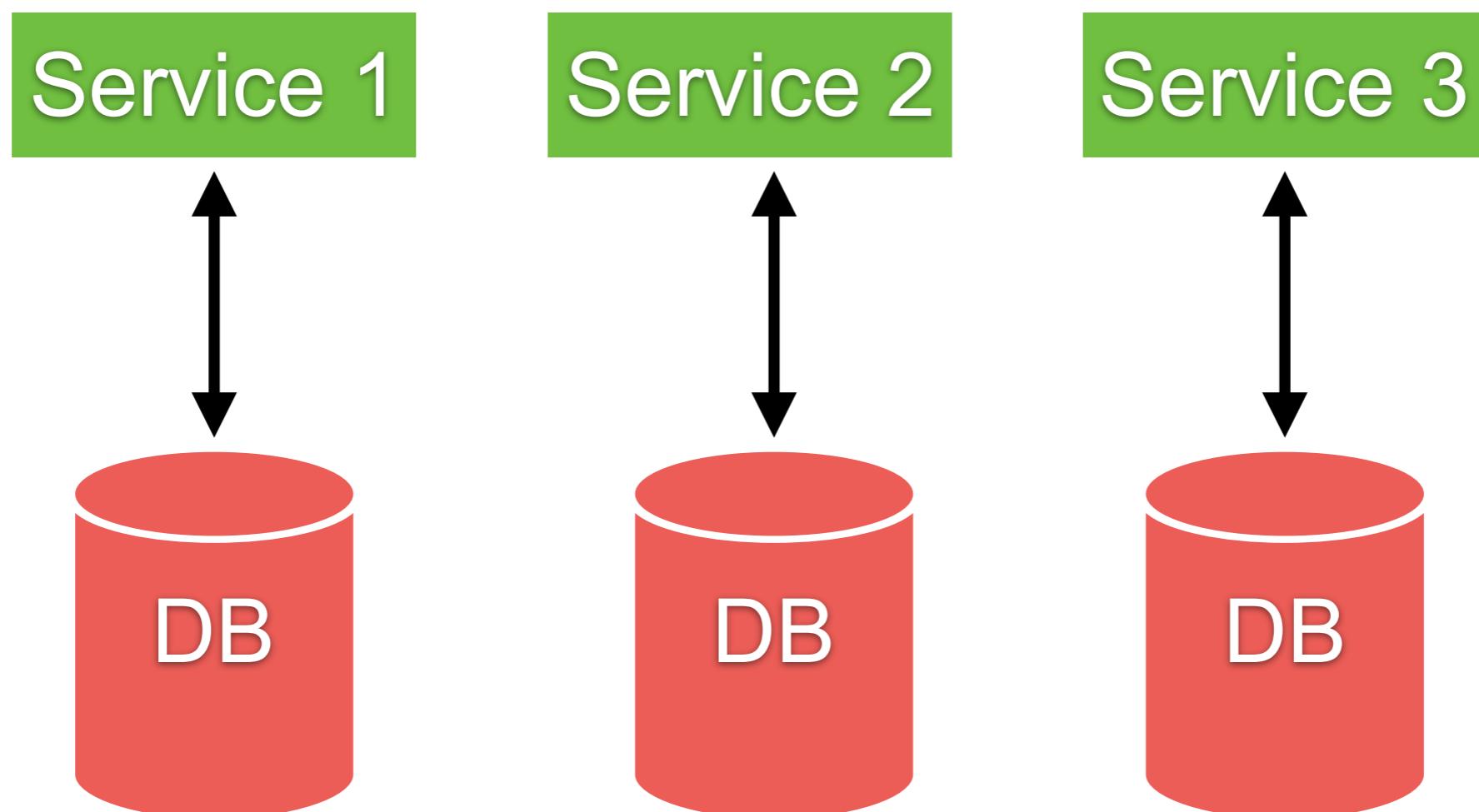
Microservices

Shared database

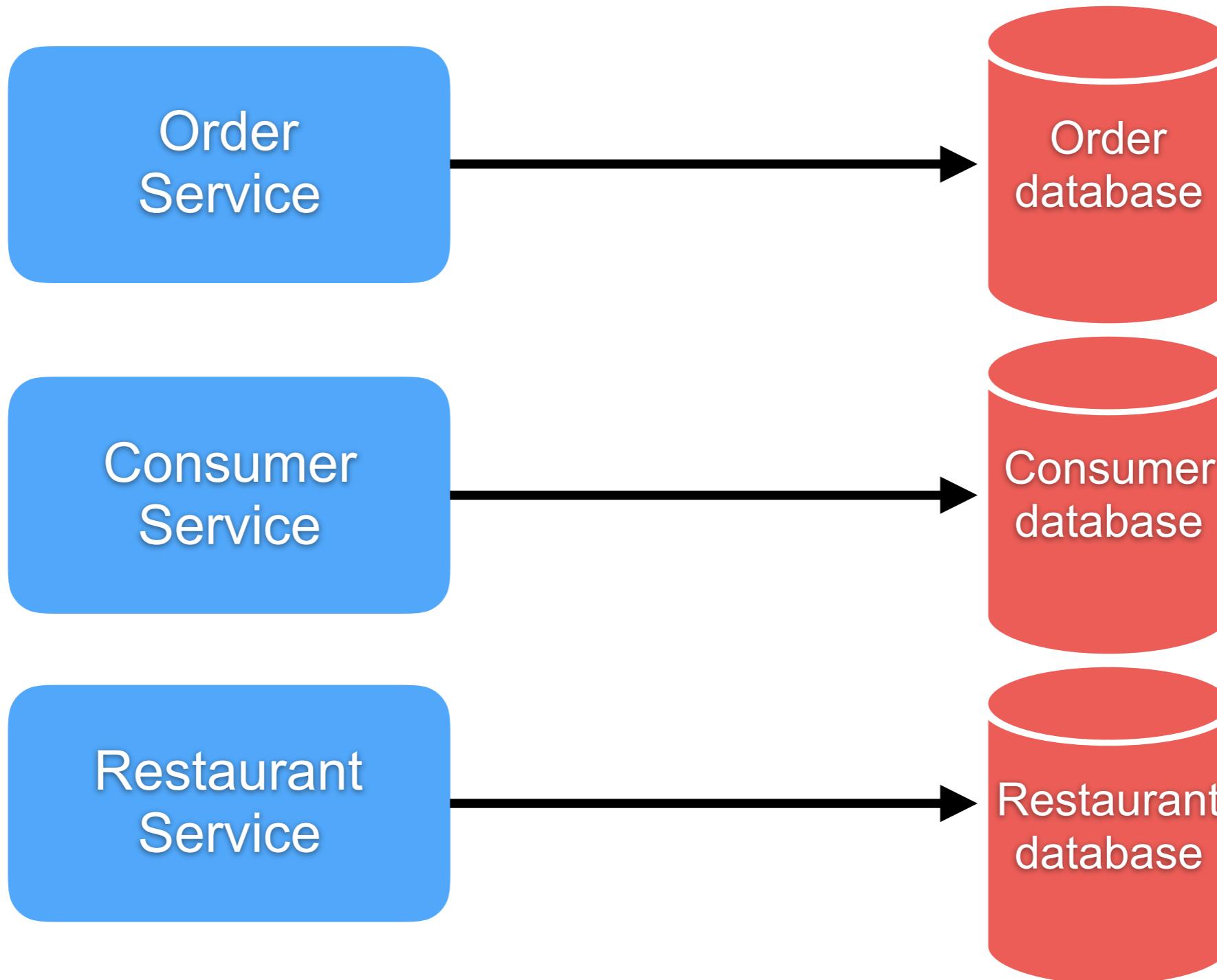


Microservices

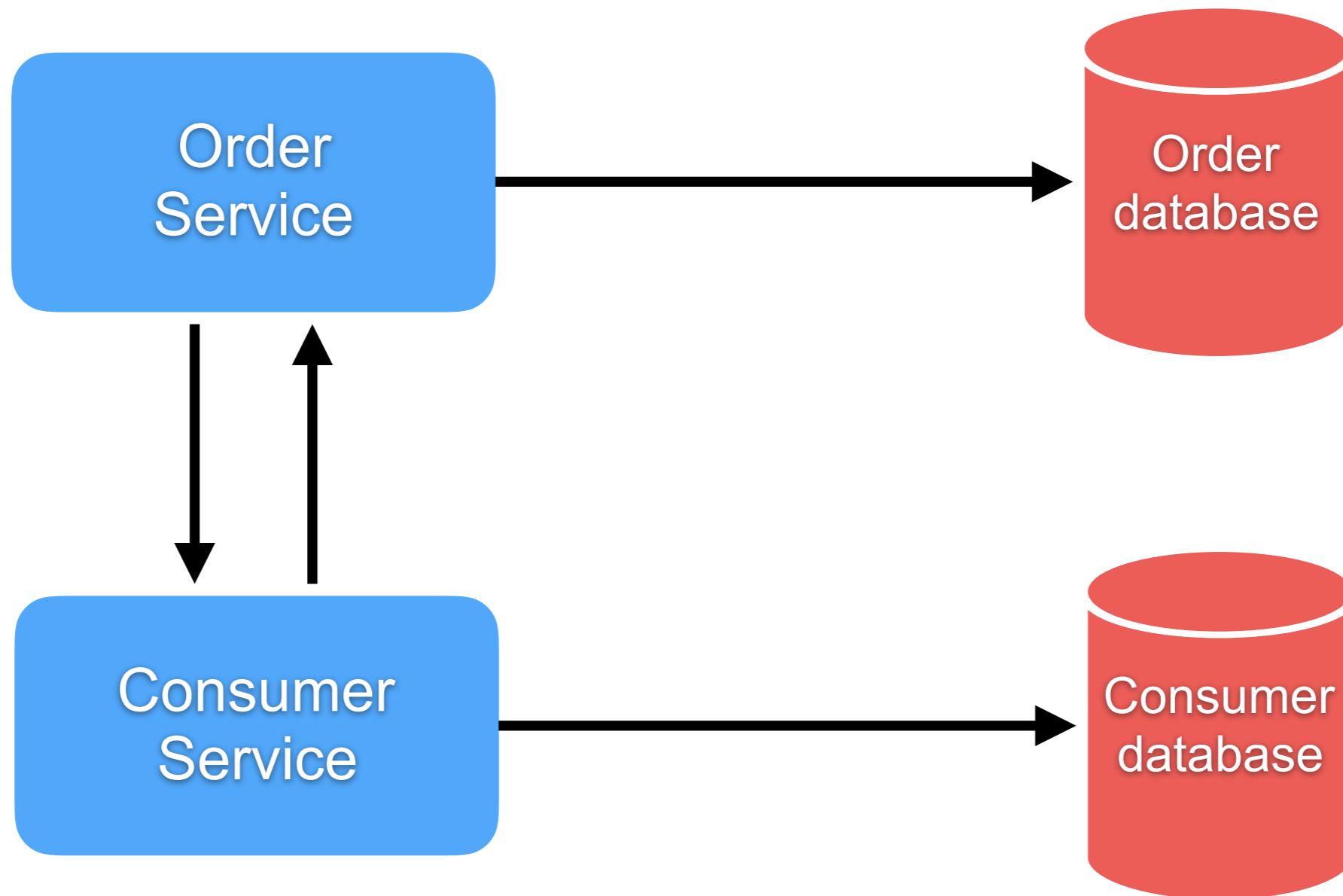
Database per service



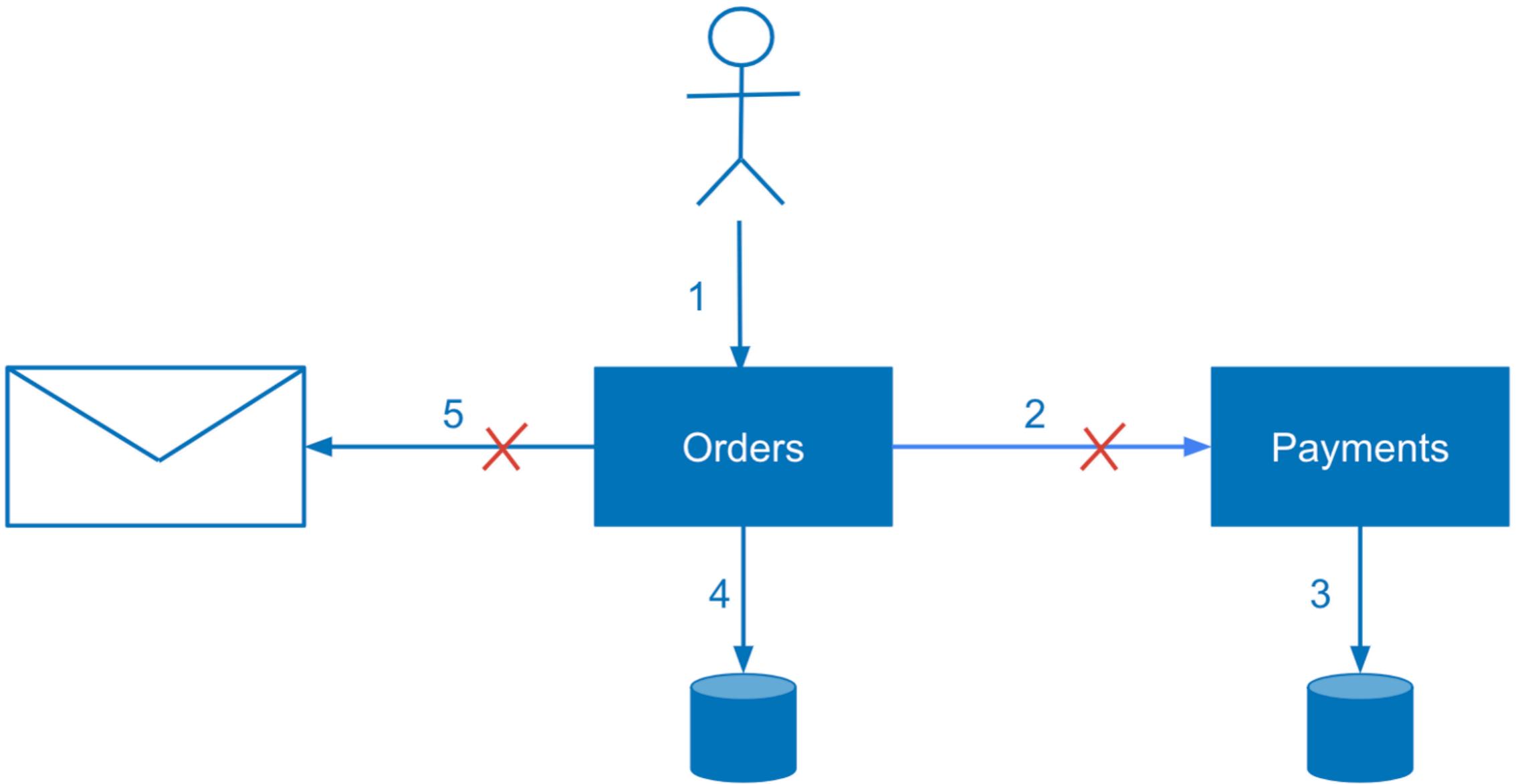
Database per service



Loose coupling = encapsulation data

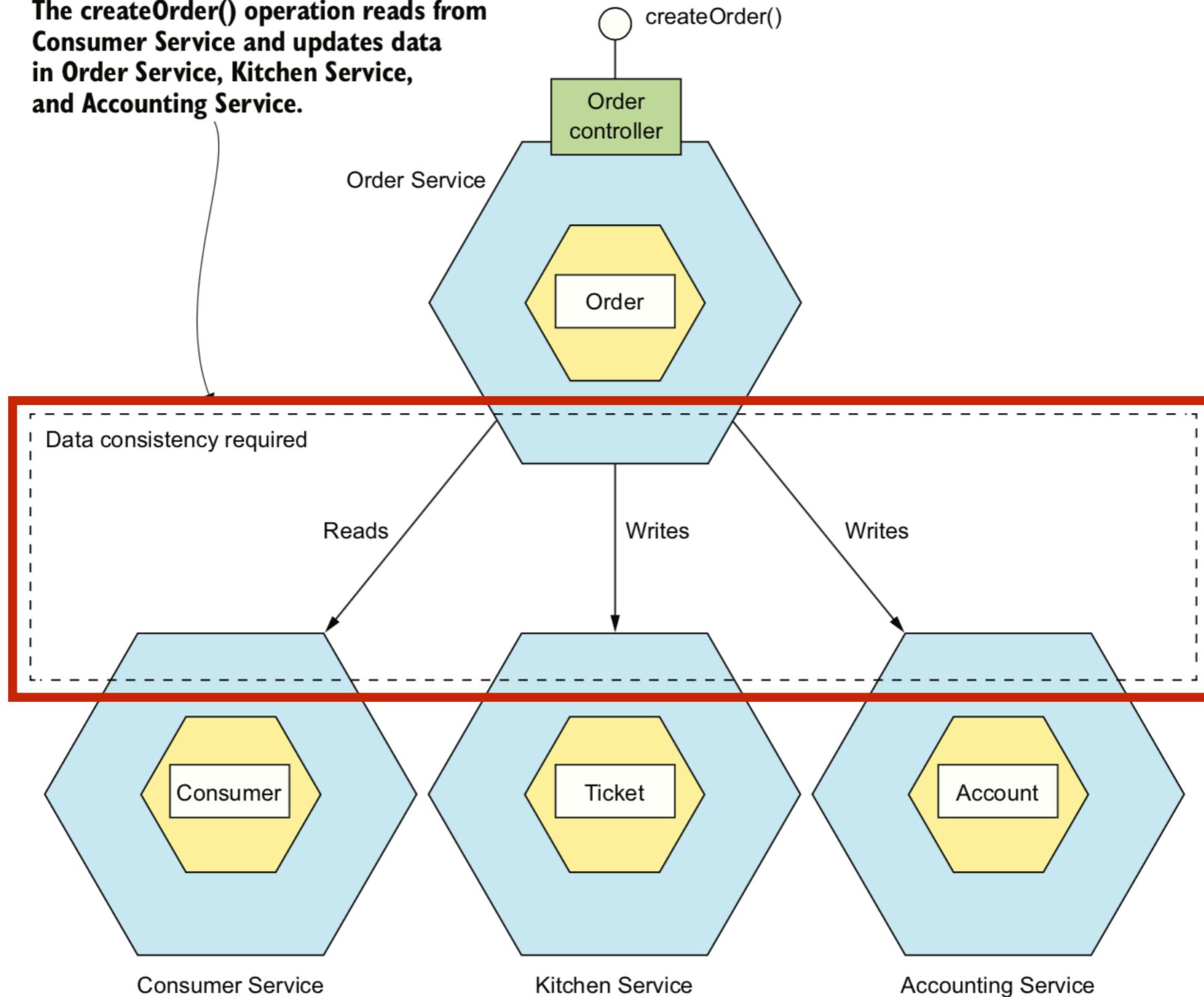


Distributed process failure !!



Problem ?

The **createOrder()** operation reads from Consumer Service and updates data in Order Service, Kitchen Service, and Accounting Service.



Can't use **ACID** transaction !!



A - Atomicity

All or Nothing Transactions

C - Consistency

Guarantees Committed Transaction State

I - Isolation

Transactions are Independent

D – Durability

Committed Data is Never Lost

(c) <http://blog.sqlauthority.com>

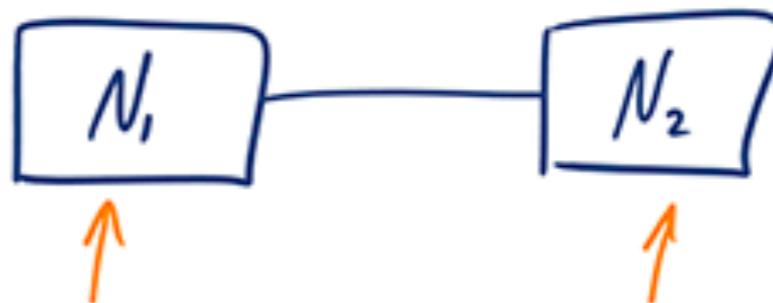


CAP Theorem

Consistency



Availability



Partition Tolerance



<http://robertgreiner.com/2014/08/cap-theorem-revisited/>



Database per service

High complexity

Query data across multiple databases

Challenge “How to join data ?”

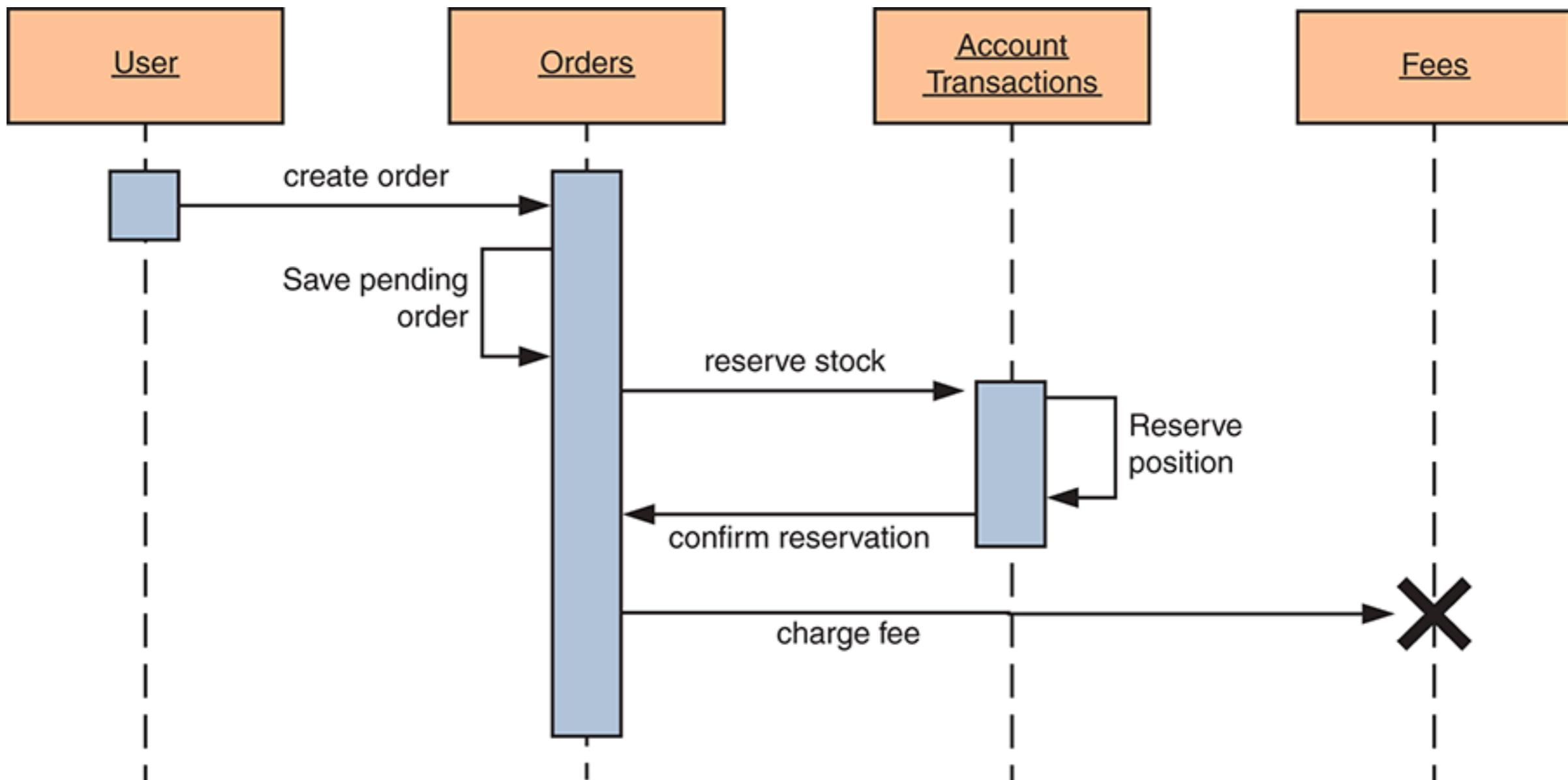
Maintain database consistency



How to maintain data consistency across services ?



Problem



Distributed transaction

Common approach is Two-phase commit(2PC)

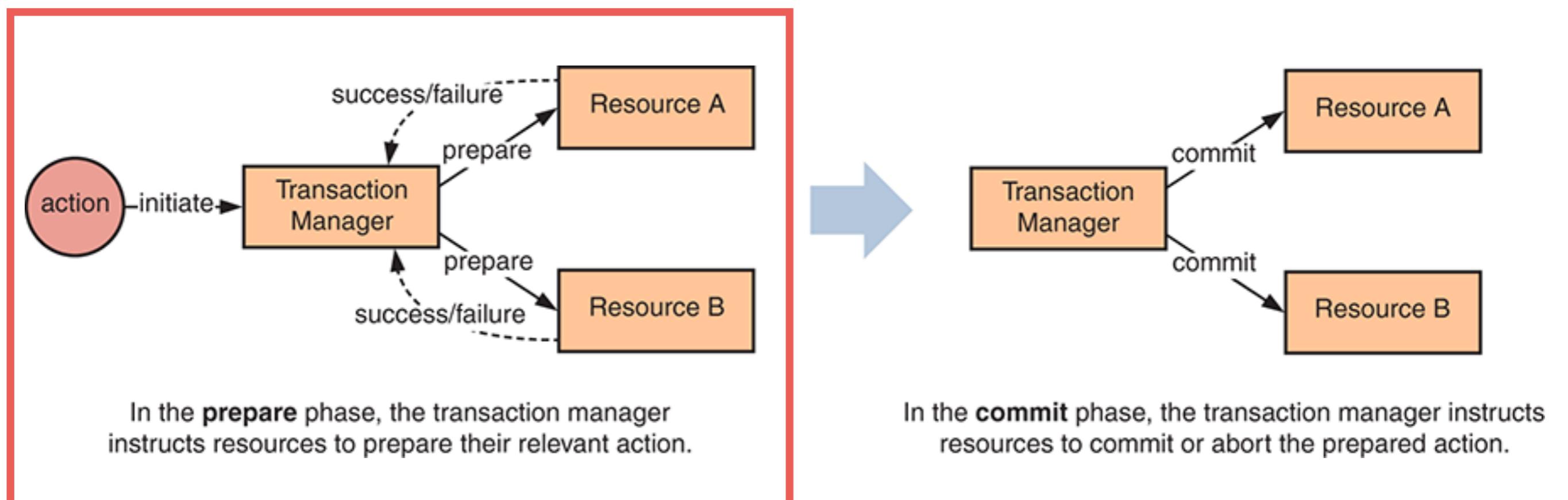
Use transaction manager to split operations across multiple resources in 2 phases

1. *Prepare*
2. *Commit*



Two-phase commit

Phase 1: prepare

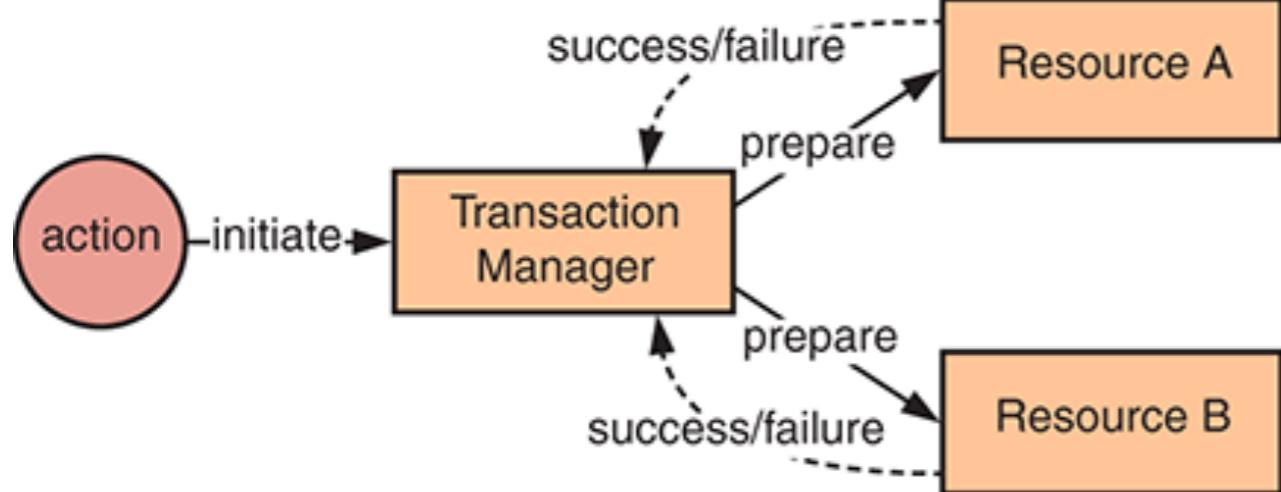


https://en.wikipedia.org/wiki/Two-phase_commit_protocol

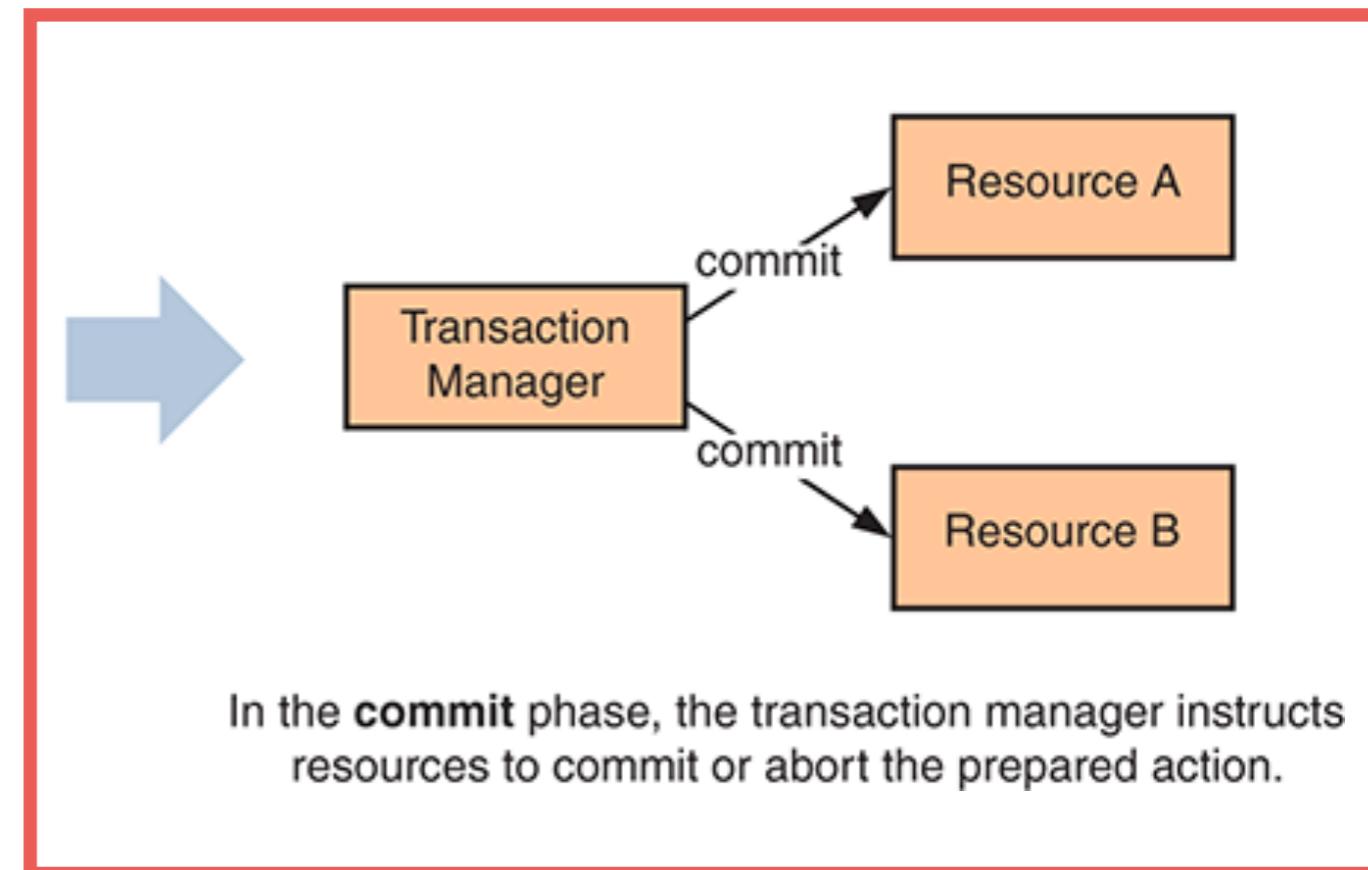


Two-phase commit

Phase 2: commit



In the **prepare** phase, the transaction manager instructs resources to prepare their relevant action.



In the **commit** phase, the transaction manager instructs resources to commit or abort the prepared action.

https://en.wikipedia.org/wiki/Two-phase_commit_protocol



**Distributed transaction places
a lock on resources under
transaction to ensure isolation**



Inappropriate for long-running operations



Increase risk of contention and deadlock

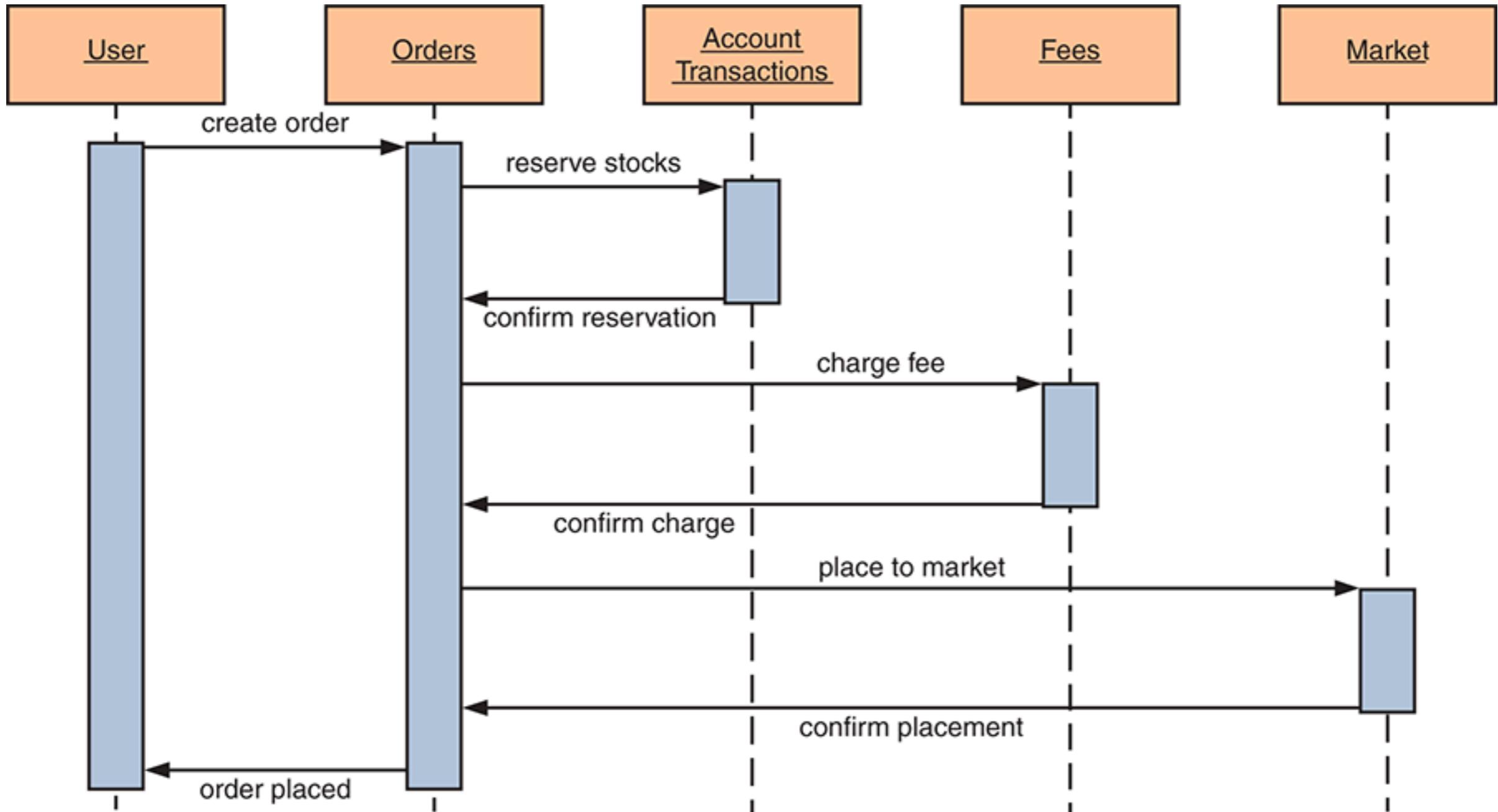


We need ...

Don't need distributed transaction
Reduce complexity of code
Reliable



Synchronous process

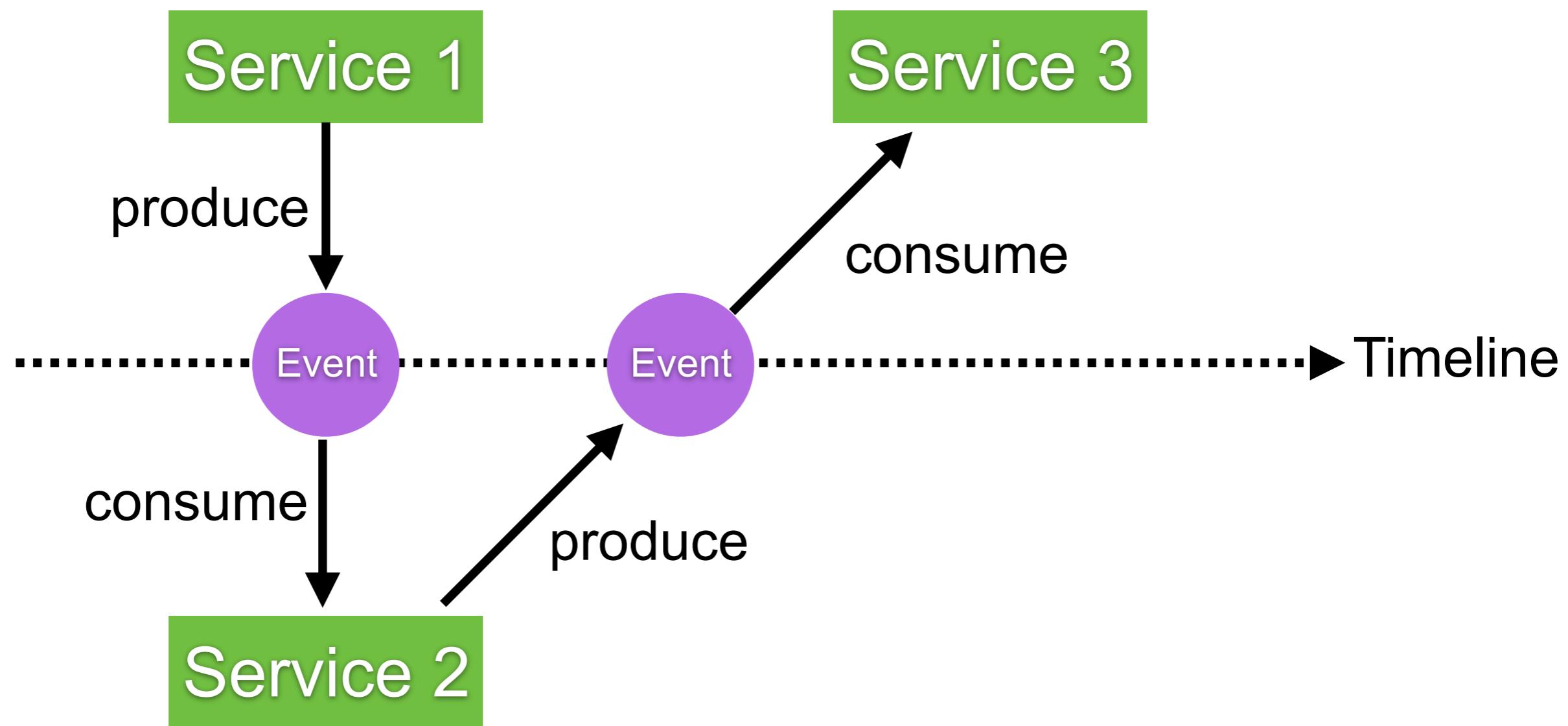


Event-based communication



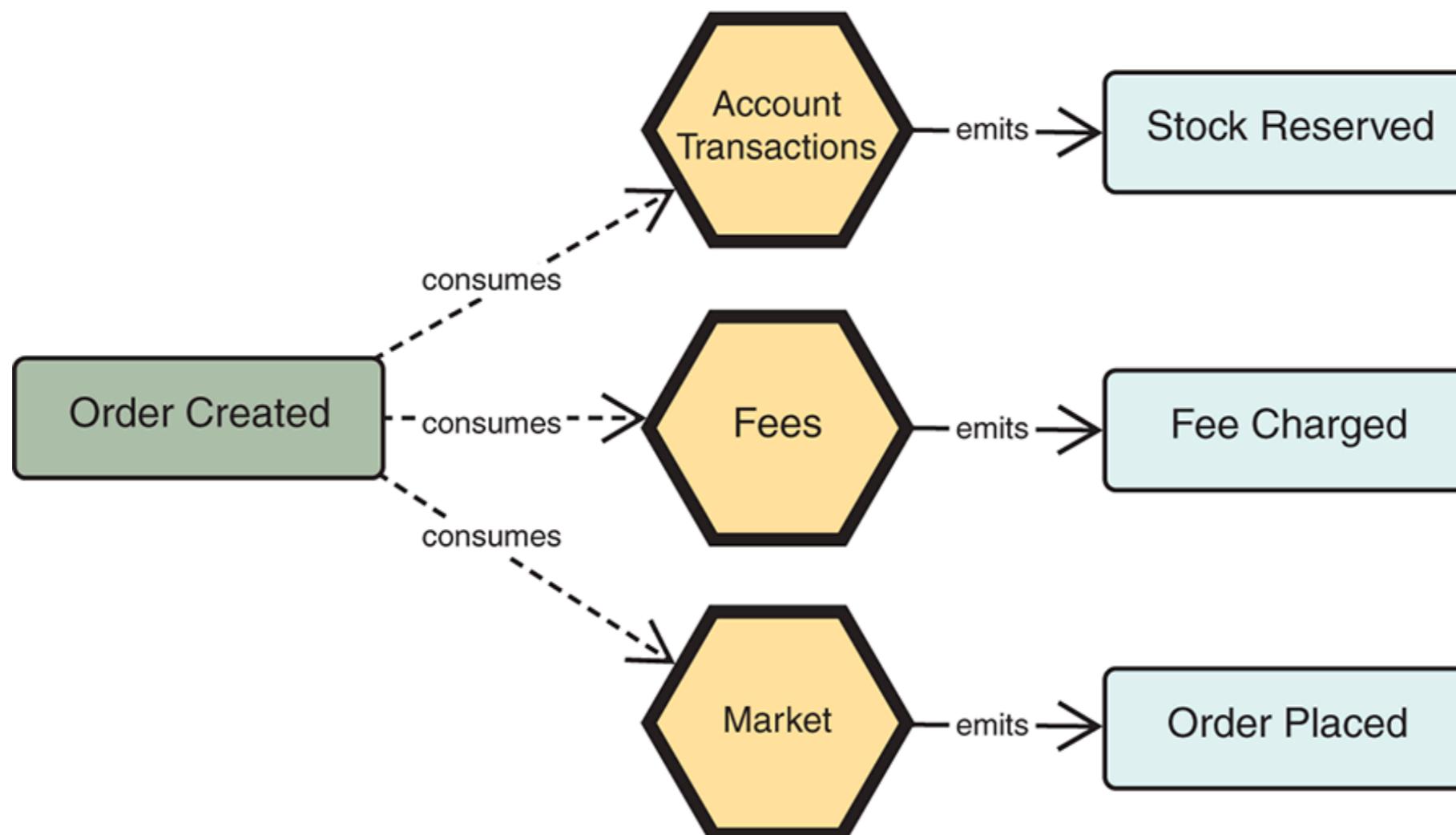
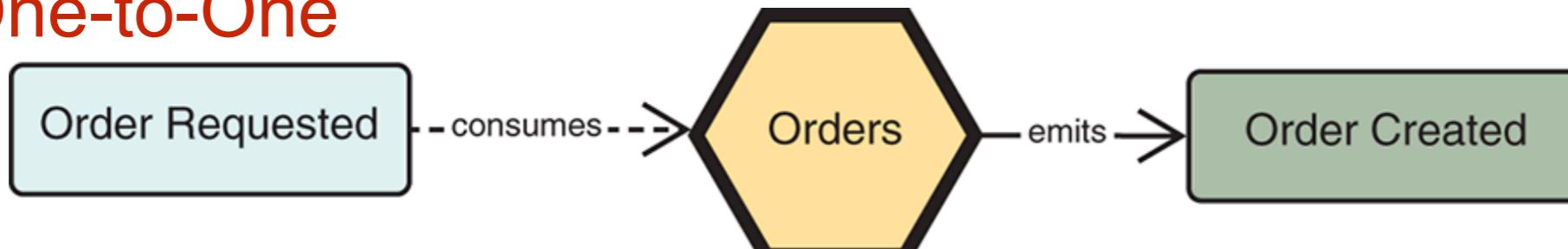
Choreography pattern

Sequence of Tx and emit **event** or **message** that trigger the next process in Tx

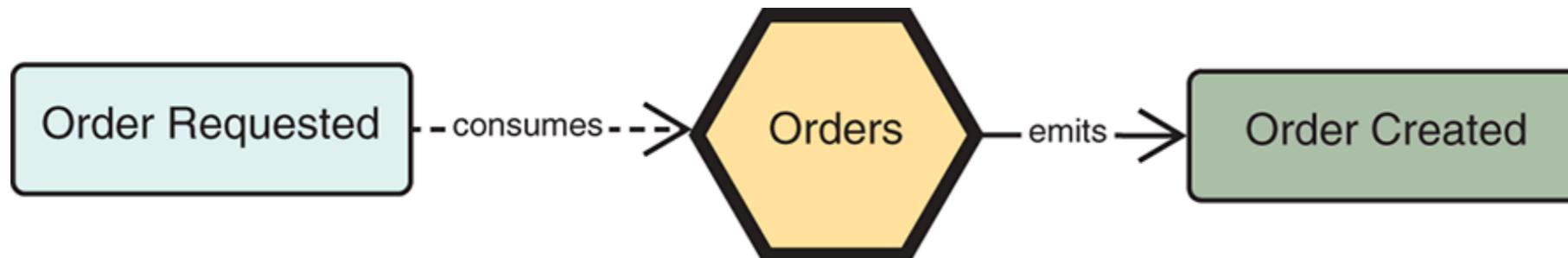


Service consume and emit event

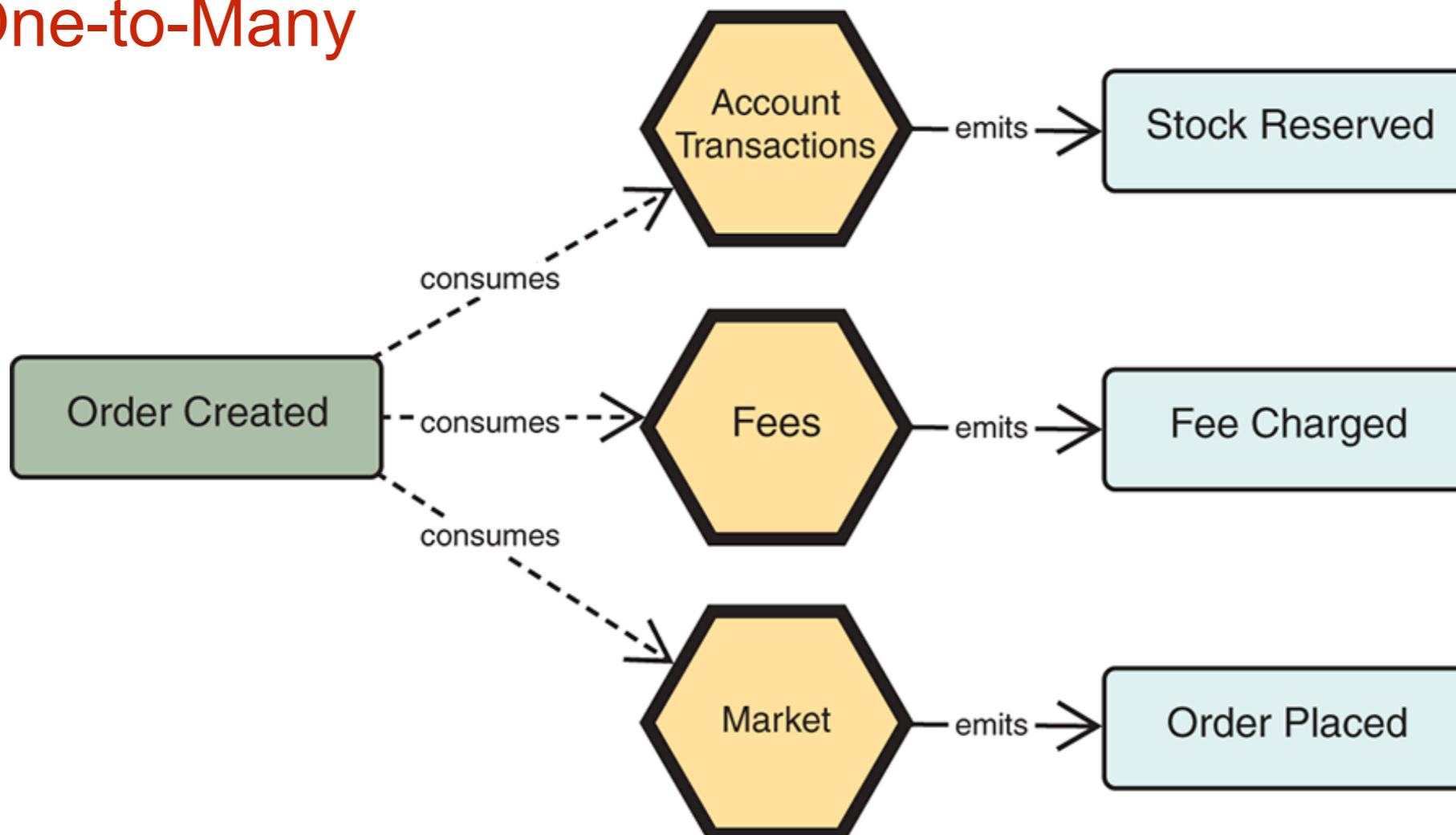
One-to-One



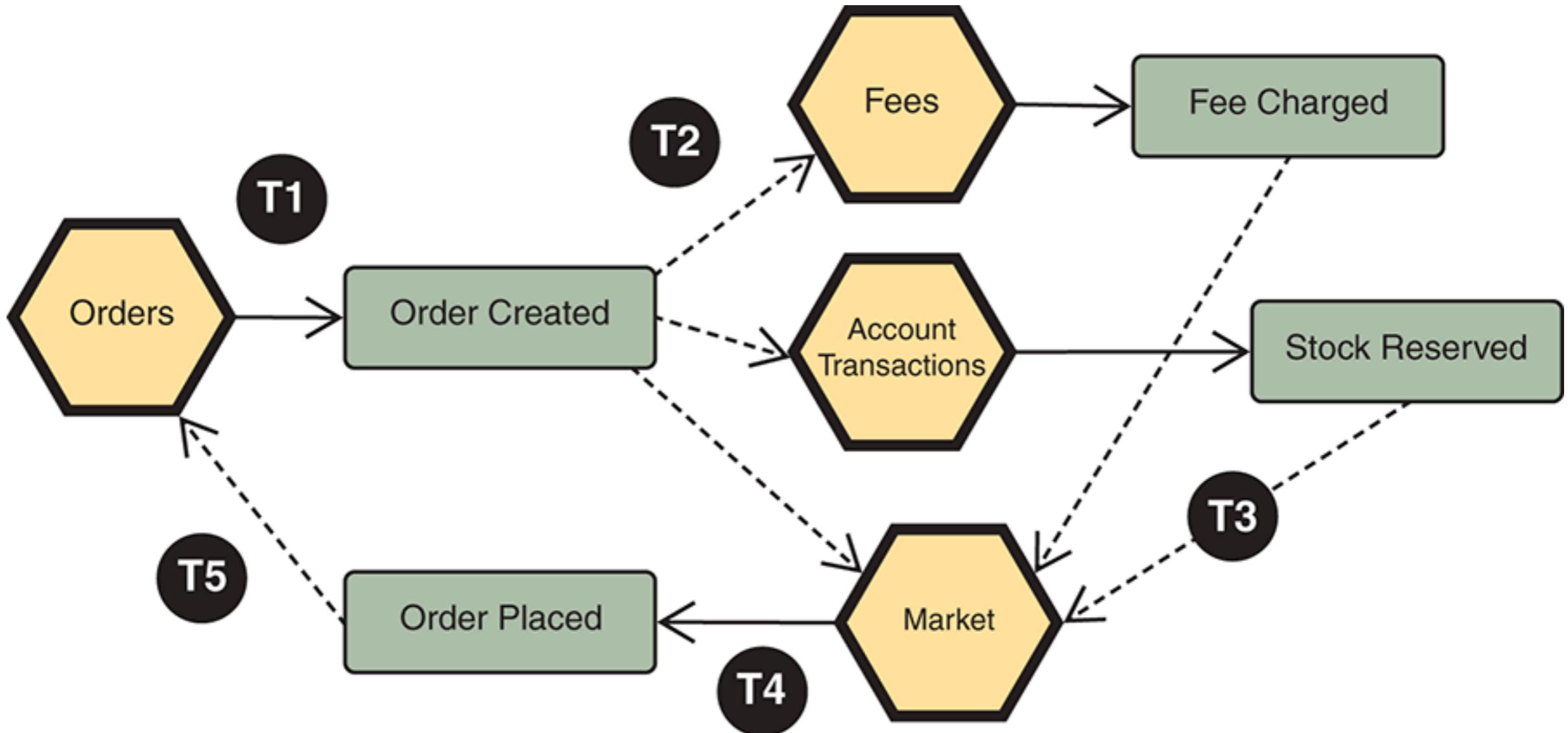
Service consume and emit event



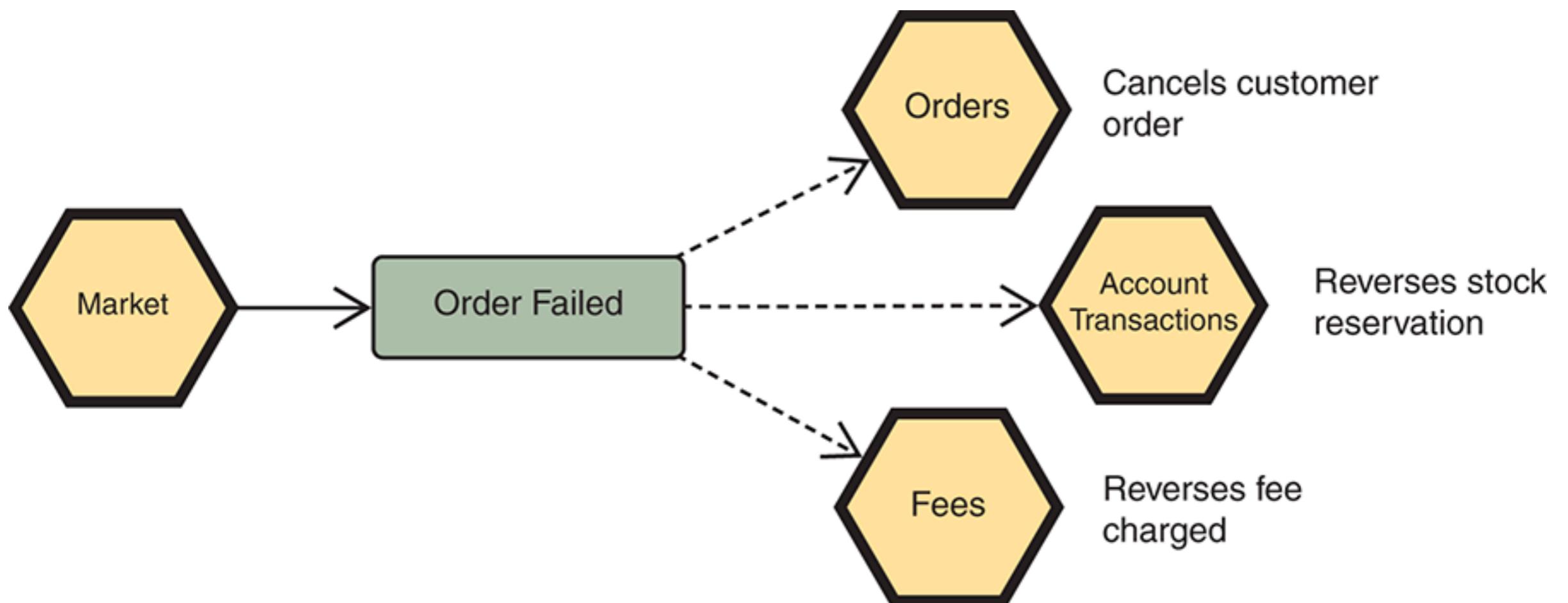
One-to-Many



Example (Success)

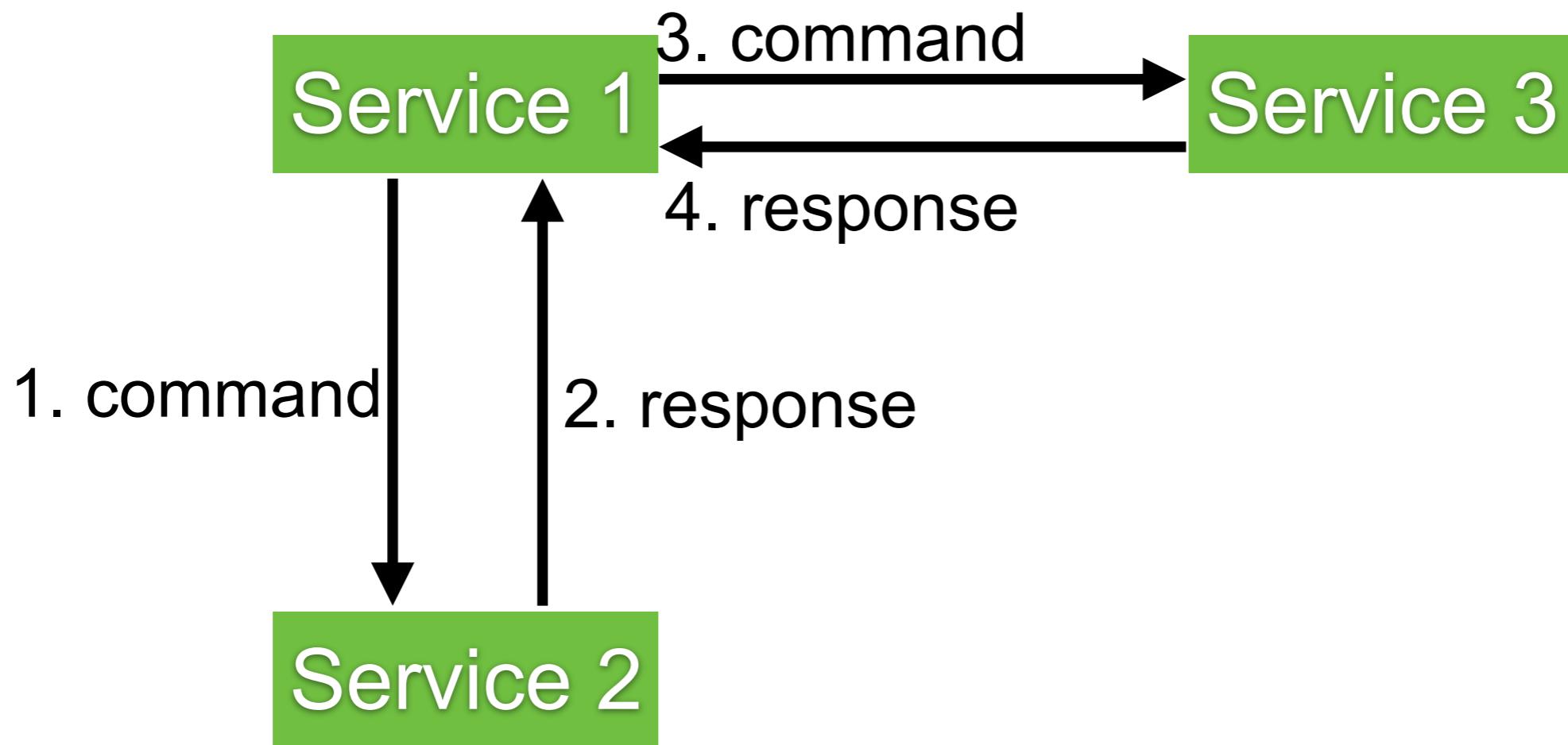


Example (Fail)

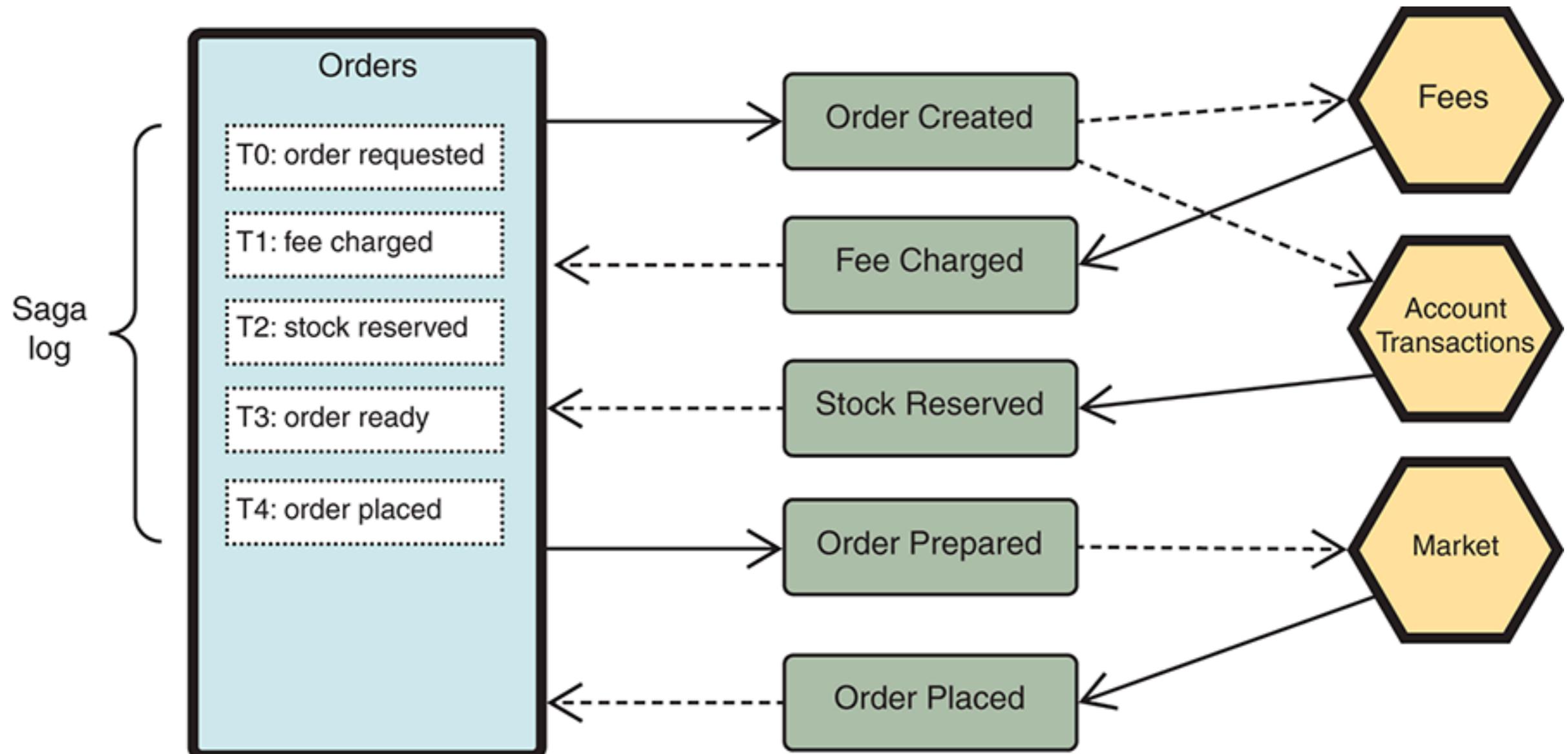


Orchestrate pattern

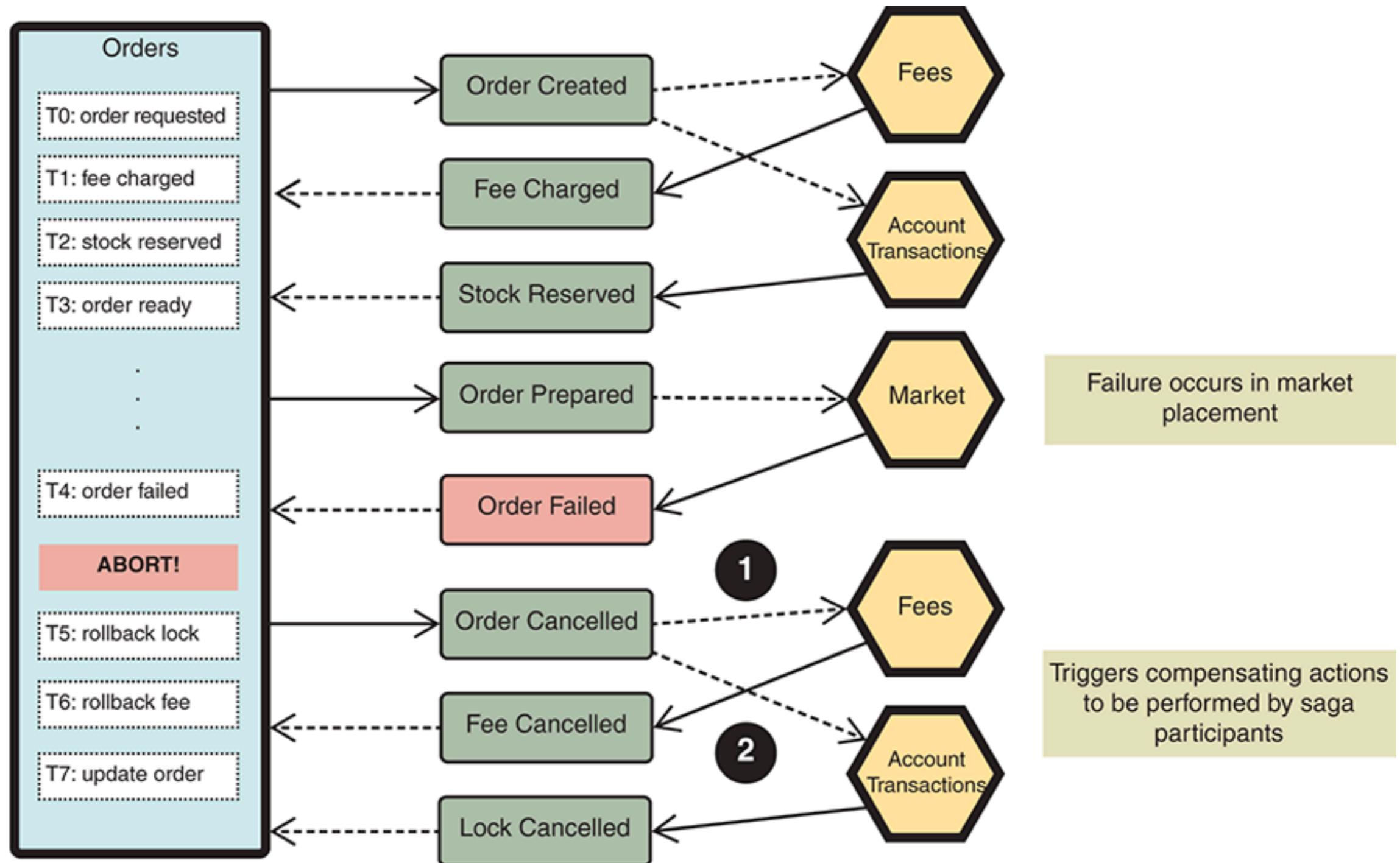
Sequence of Tx and emit **event or message** that trigger the next process in Tx



Example (Success)



Example (Fail)



Consistency patterns

Name	Strategy
Compensating action	Perform action that undo prior action(s)
Retry	Retry until success or timeout
Ignore	Do nothing in the event of errors
Restart	Reset to the original state and start again
Tentative operation	Perform a tentative operation and confirm (or cancel) later



“Saga”



Message vs Event

Message is addressed to someone

Event is something that happen and someone can react to that



Event sourcing

Maintain source of truth of business
Immutable sequence of events

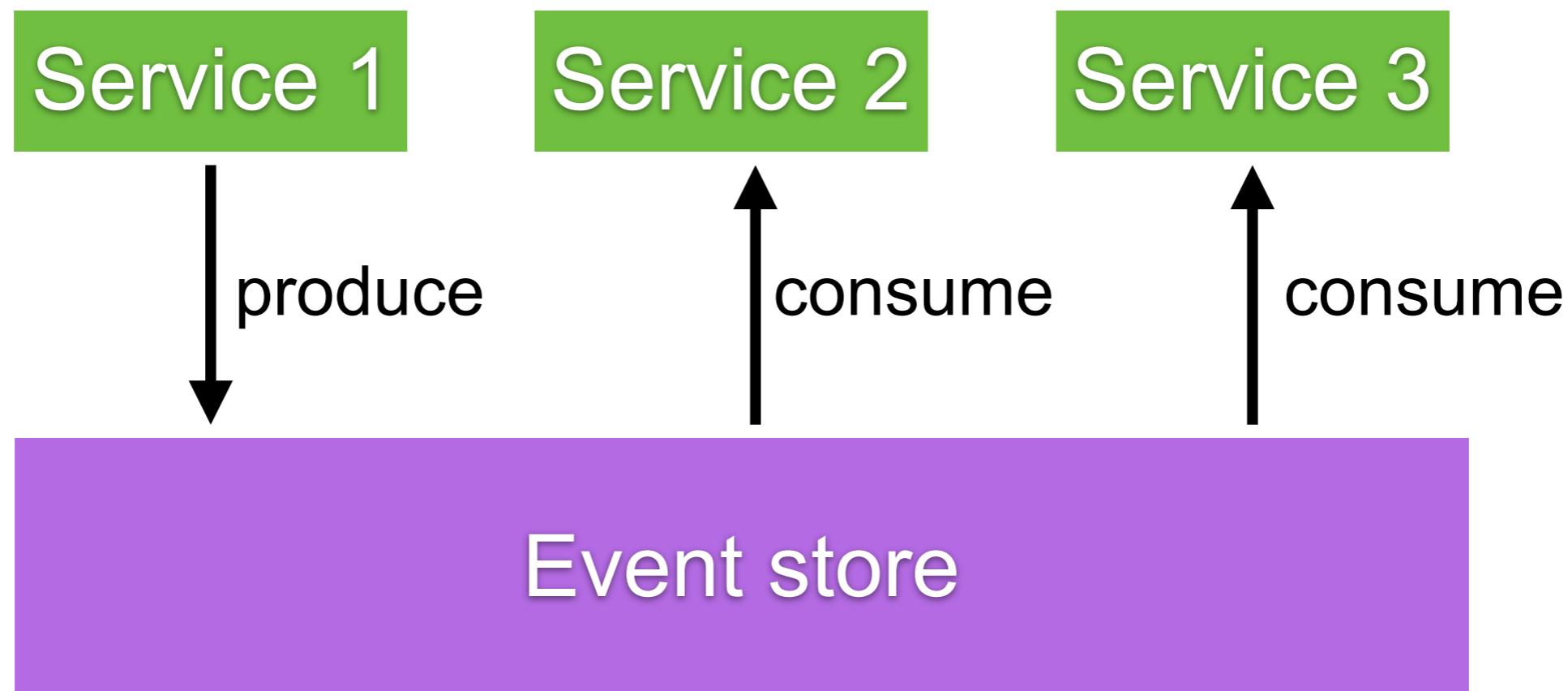


Sequence of event is keep in **Event store**



Event store

Keep events in order
Broadcast new event

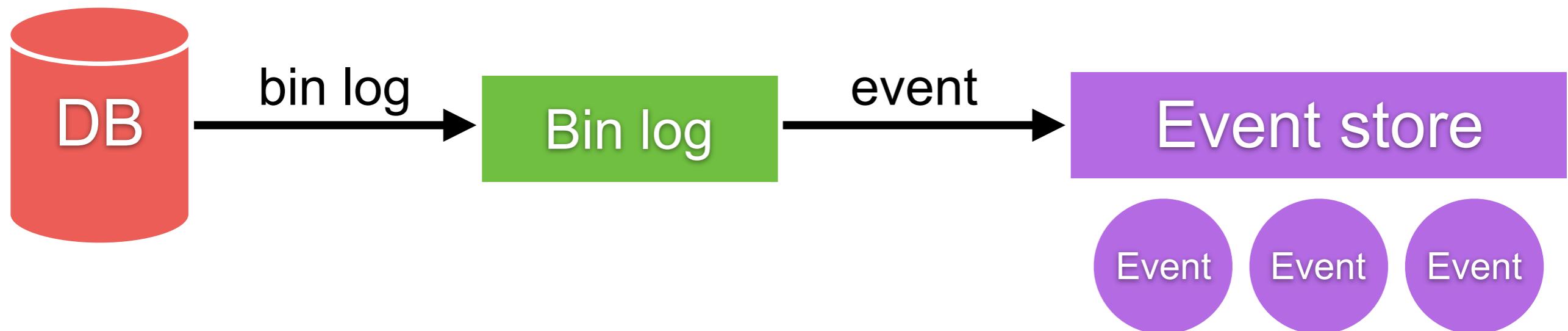


**With Event sourcing, we can
decouple services (query and
command)**



Event sourcing with database

Working with database



Binlog or Binary log event is information about data modification made to database



Event sourcing

Duplication data (denormalize database)
Complexity (separate query and command)
Difficult to maintain



Event sourcing can't solve all problems



Start with **understand** your purpose



Start with understand your
problem to resolve



How to implement queries ?



Query data from multiple services

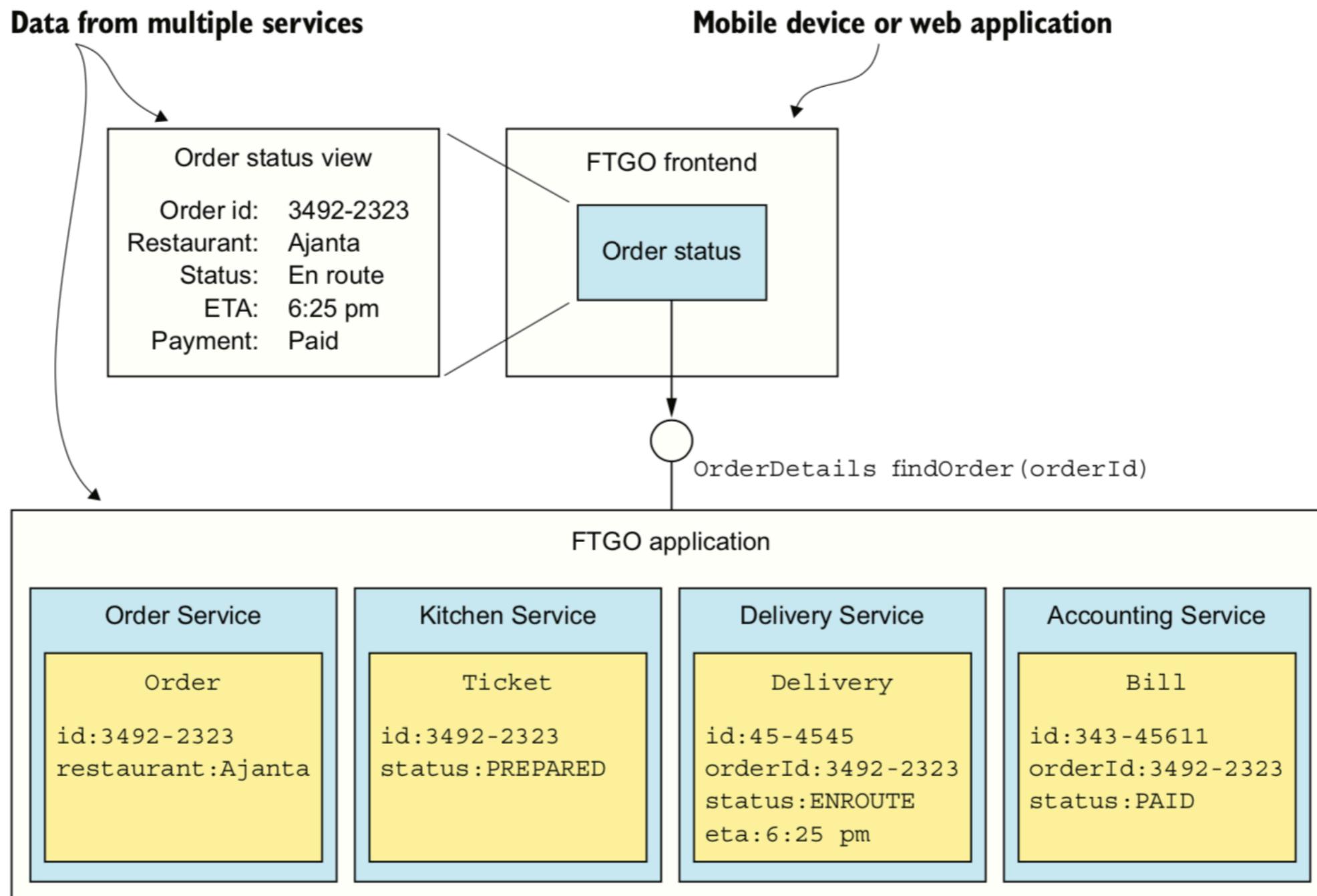
API composition

Cold data from services

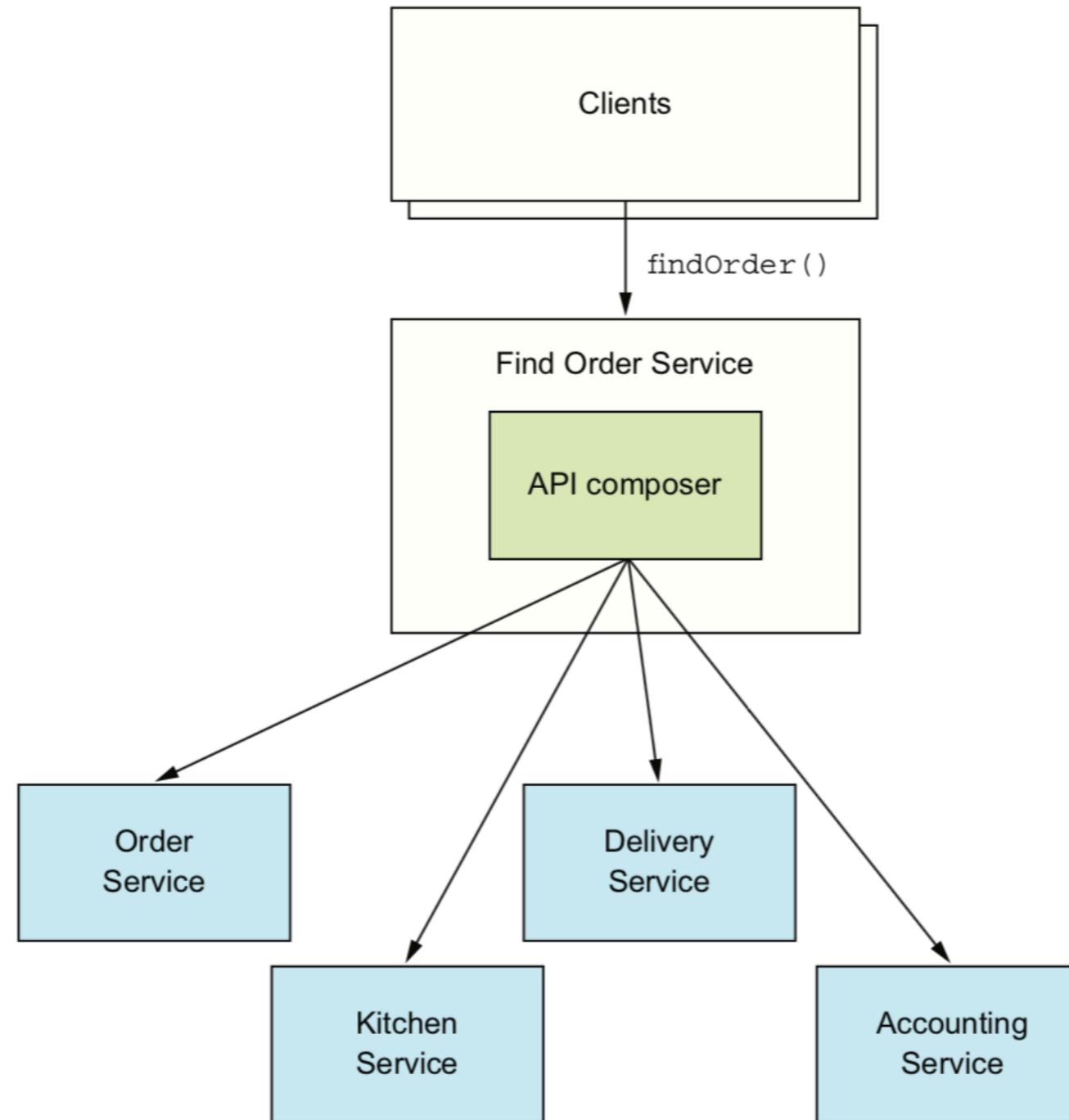
CQRS (Command Query Responsibility Segregation)



Problem ?



API composition pattern



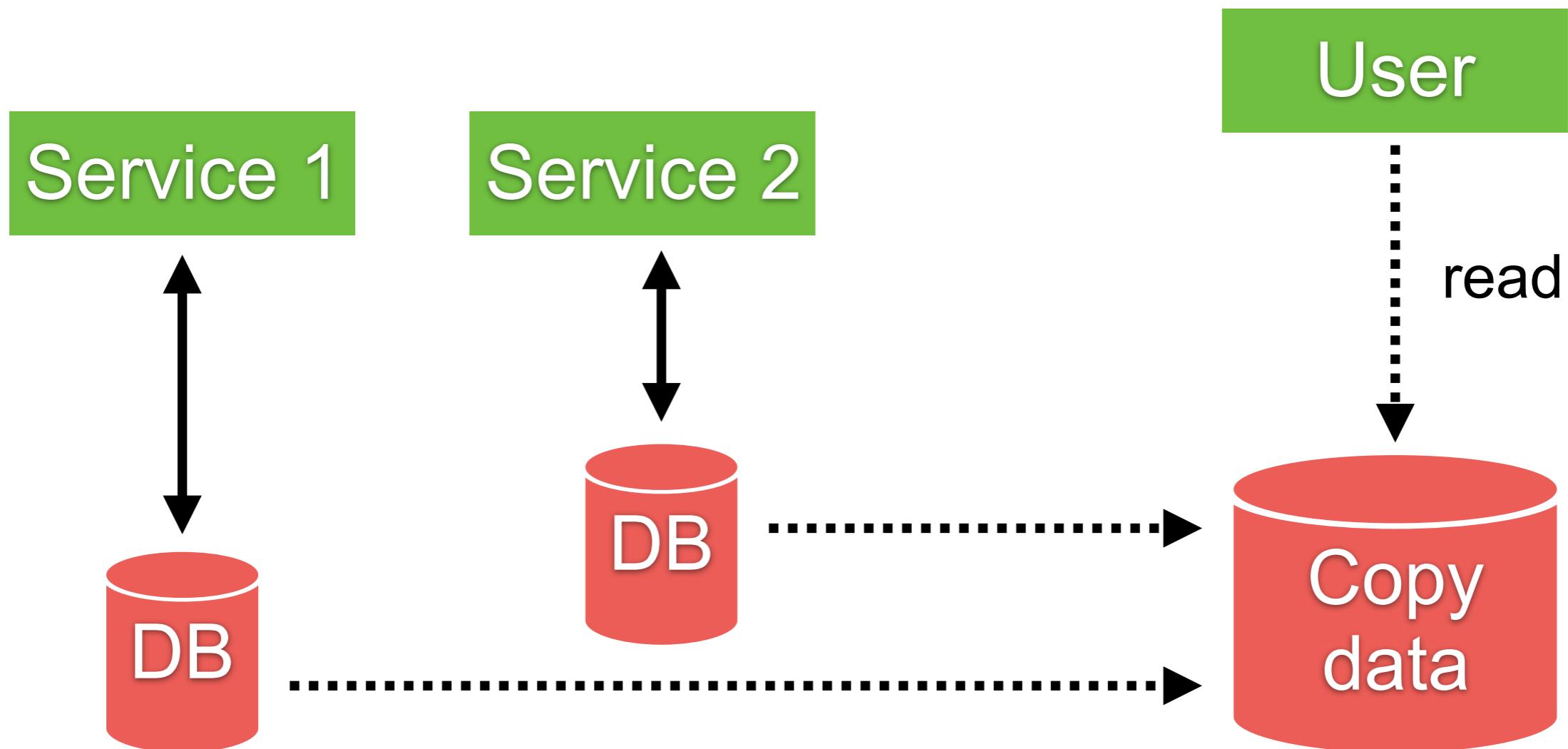
Drawbacks

Increase overhead
Lack of transactional data consistency
Reduce availability



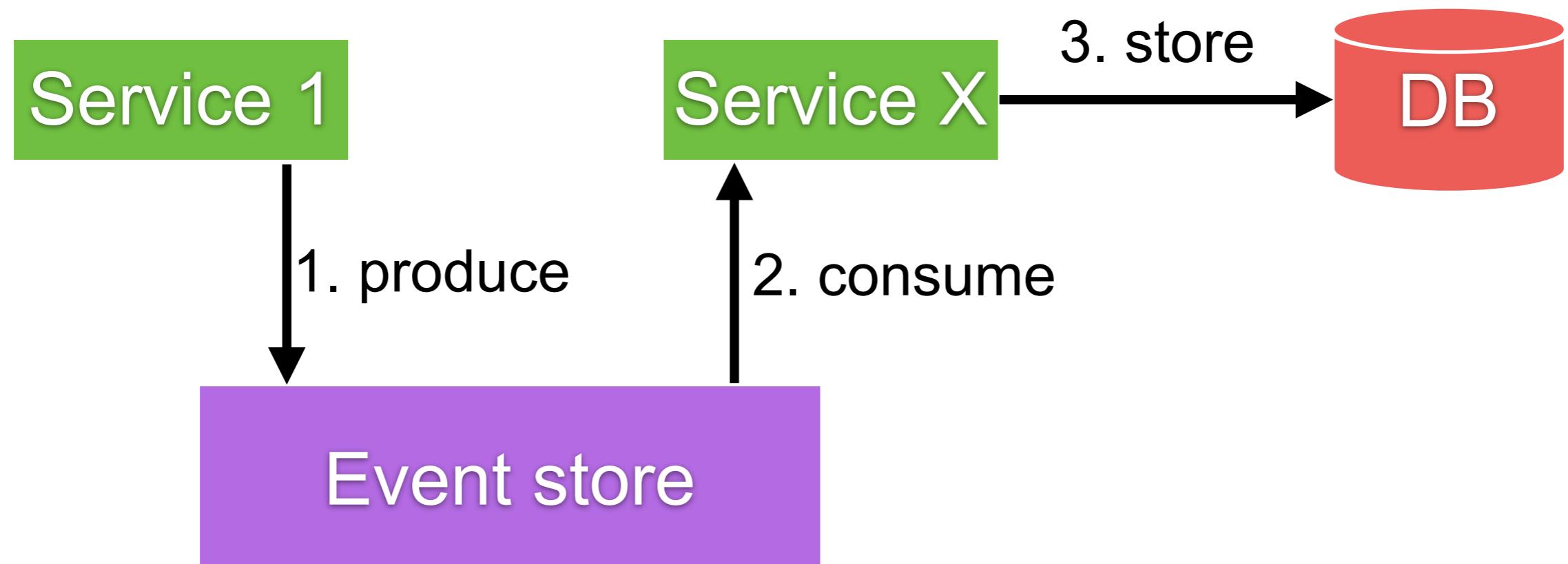
Storing copy data

Copy or caching data to other database



Working with Event-based

Publish event to store copy data

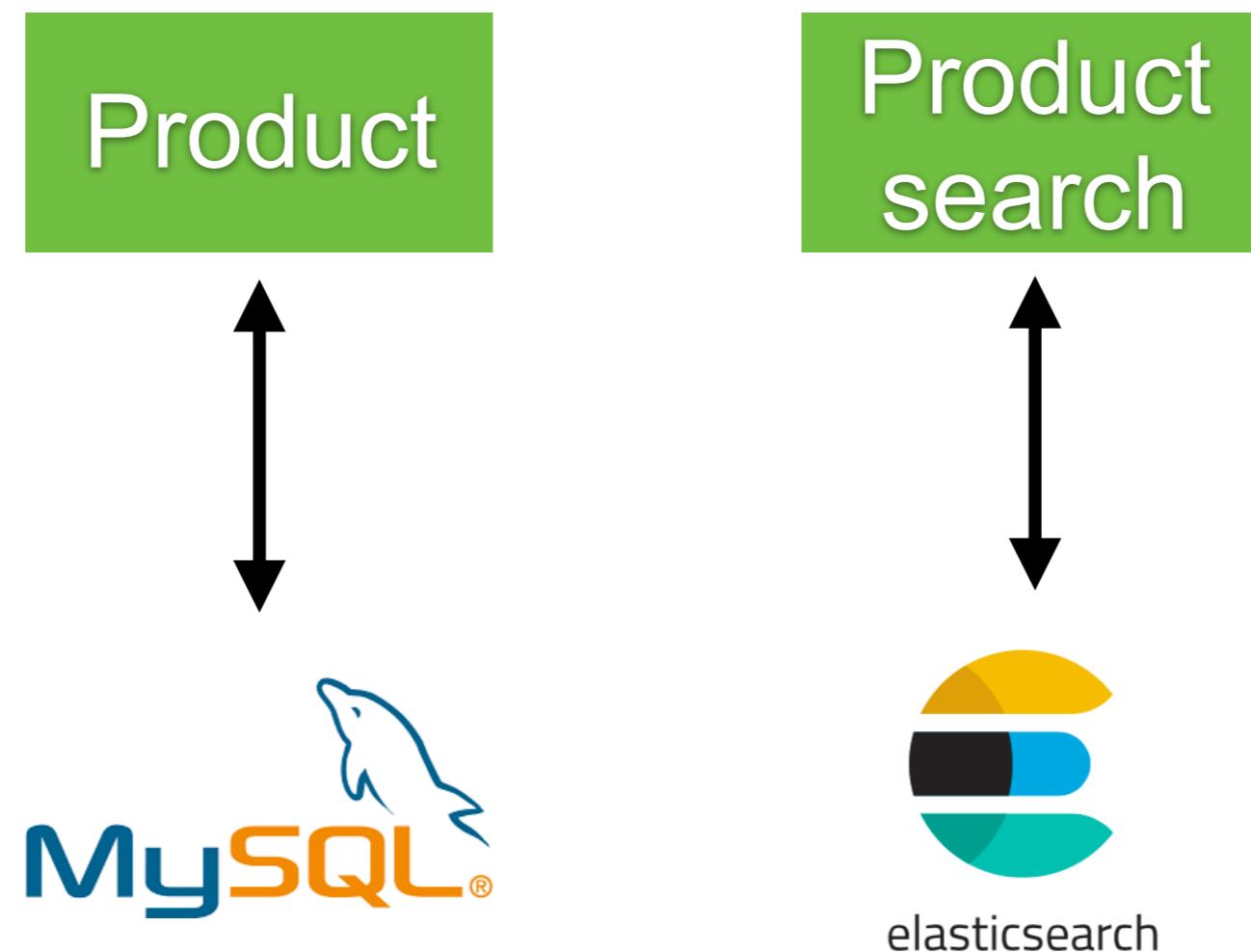


Separate query and command



Separate data for read and write

For example MySQL to write, Elasticsearch to search

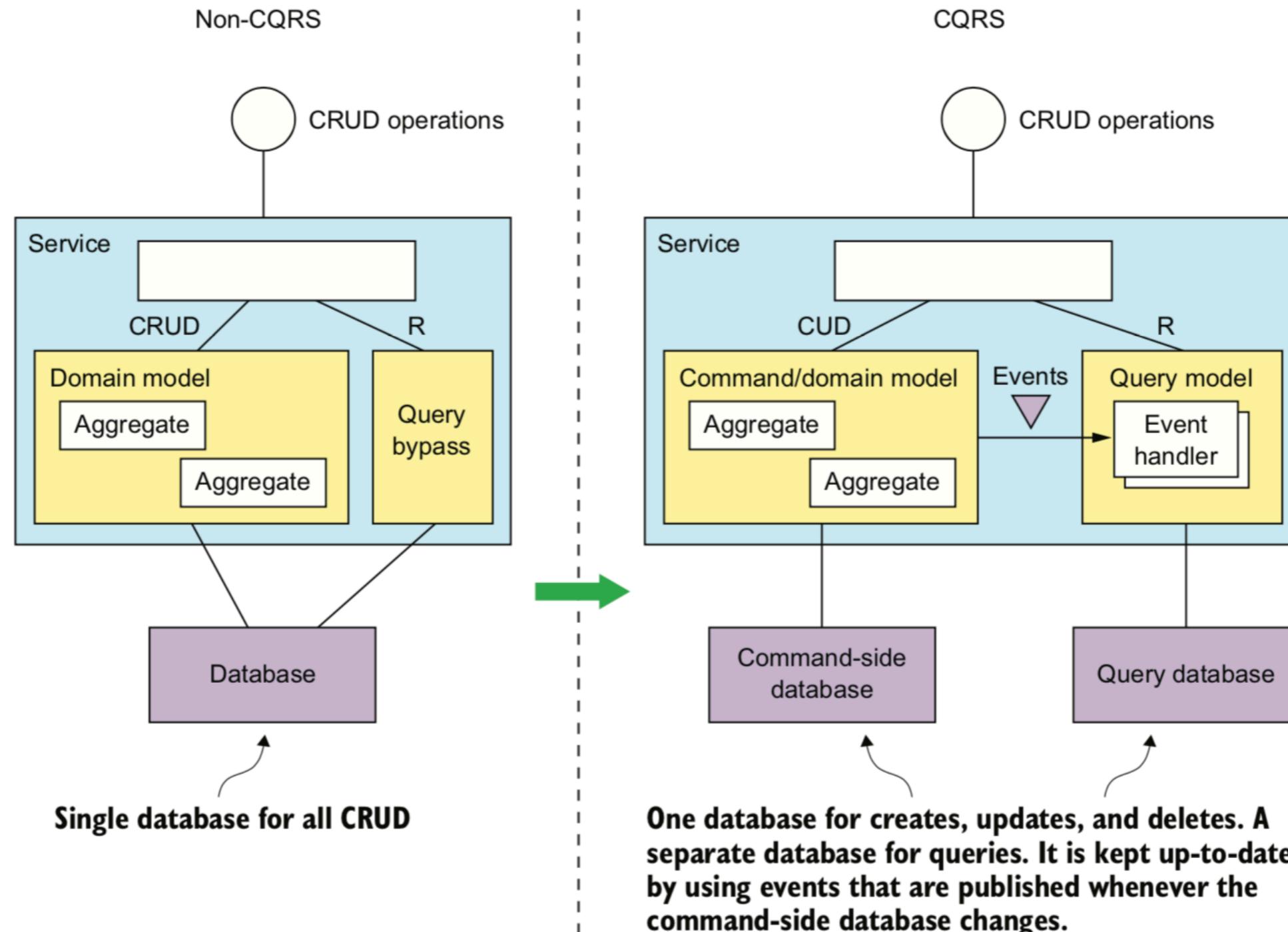


CQRS

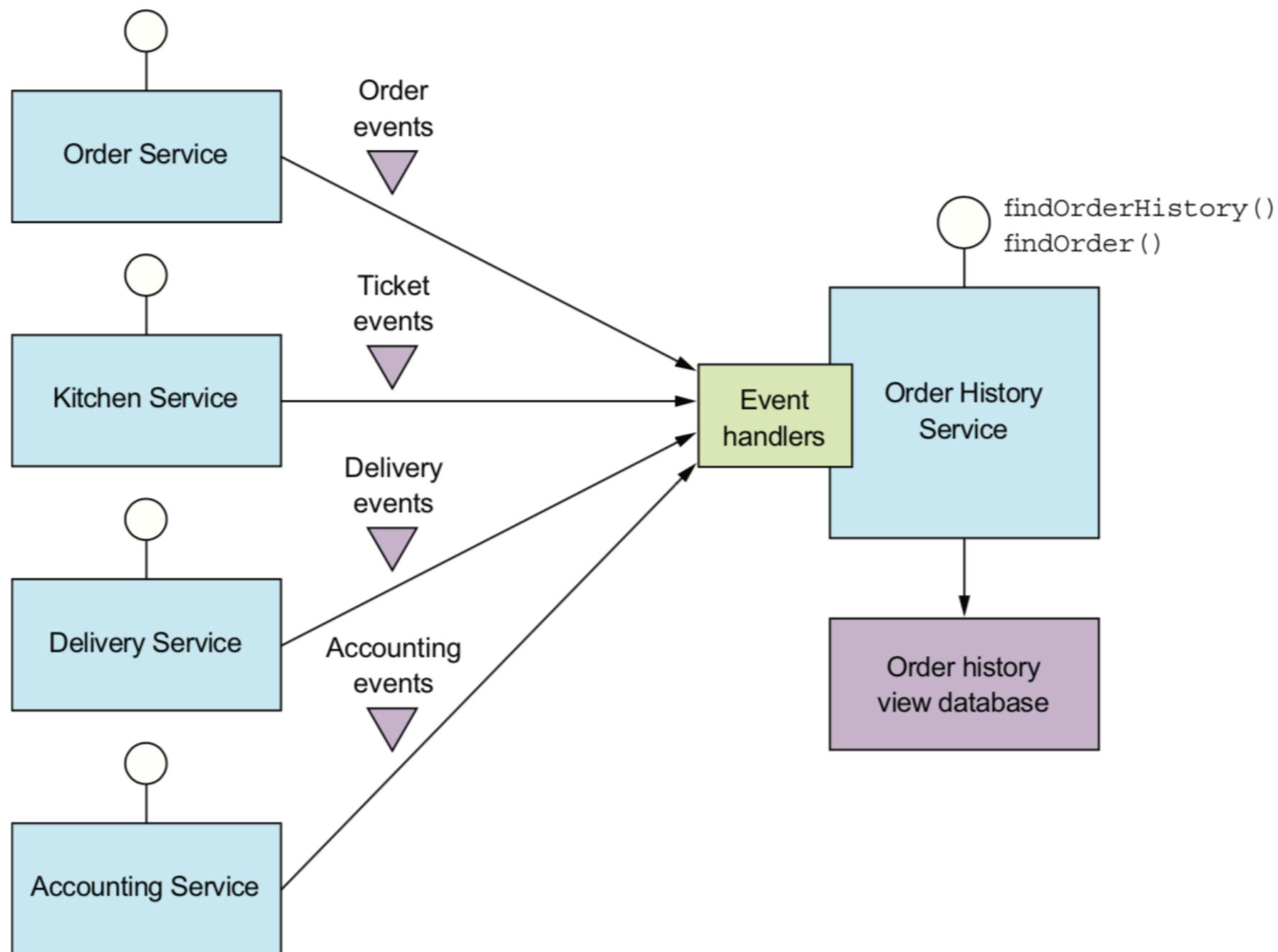
Command Query Responsibility Segregation



CQRS = Separate command from query



CQRS = Query-side service



Benefits

Improve separation of concern
Efficiency query



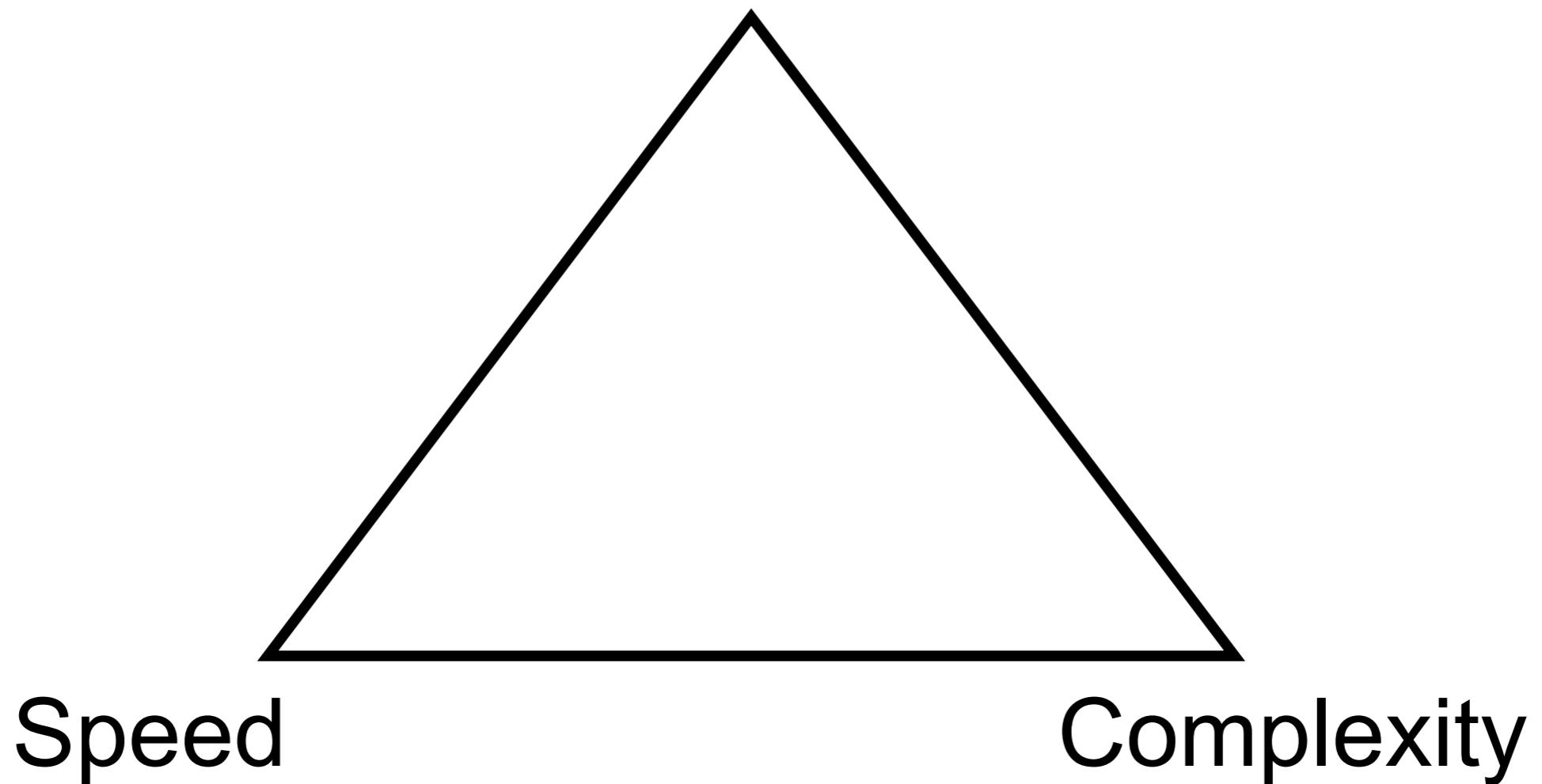
Drawbacks

More complex

Lag between command and query side



Customer value

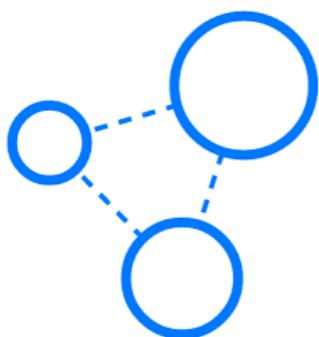


More ...



Beyond Microservice

Process and Organization



Flexible organizational structure

With clear roles and accountabilities



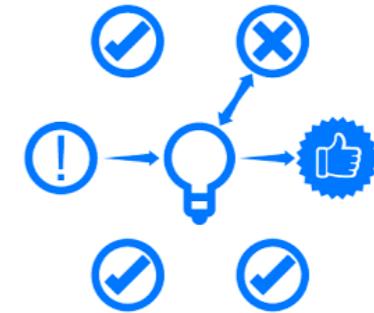
Efficient meeting formats

Geared toward action and eliminating over-analysis



More autonomy to teams and individuals

Individuals solve issues directly without bureaucracy



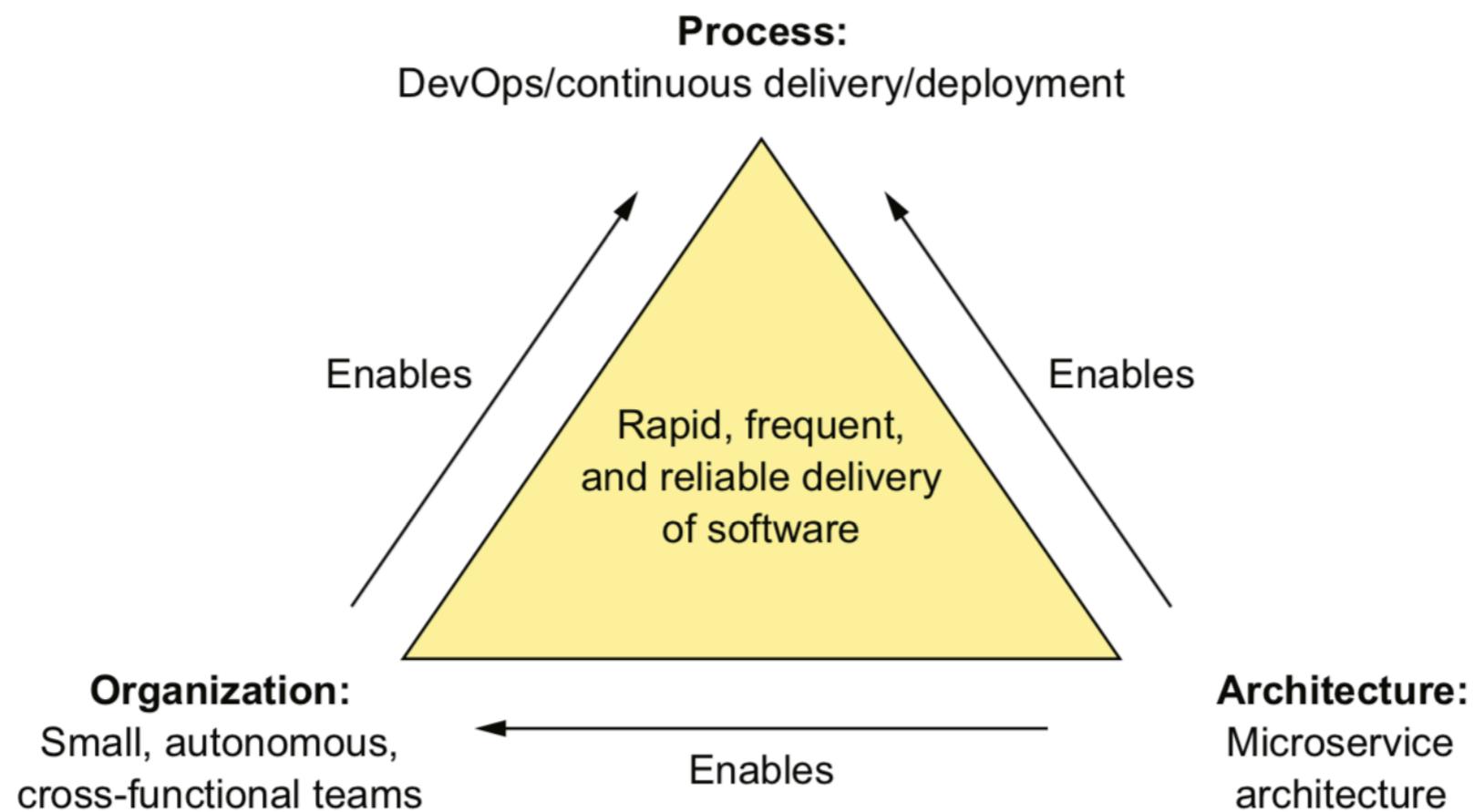
Unique decision-making process

To continuously evolve the organization's structure.



Success project needs ...

Organization process
Development process
Delivery process



Don't forget about the human side when adopt Microservice



"organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations"

- *Melvin Conwey, 1968* -



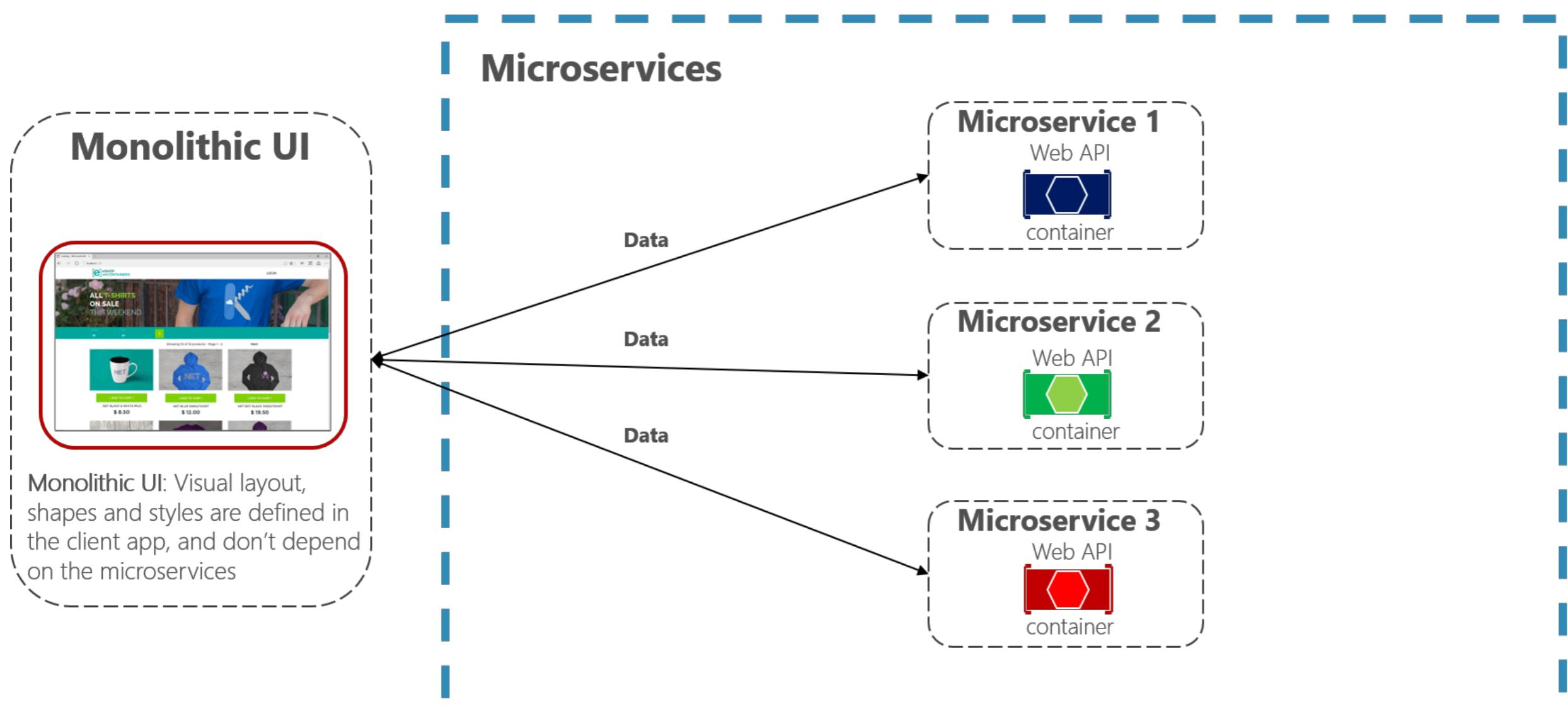
More ...



Integrate services with User Interface



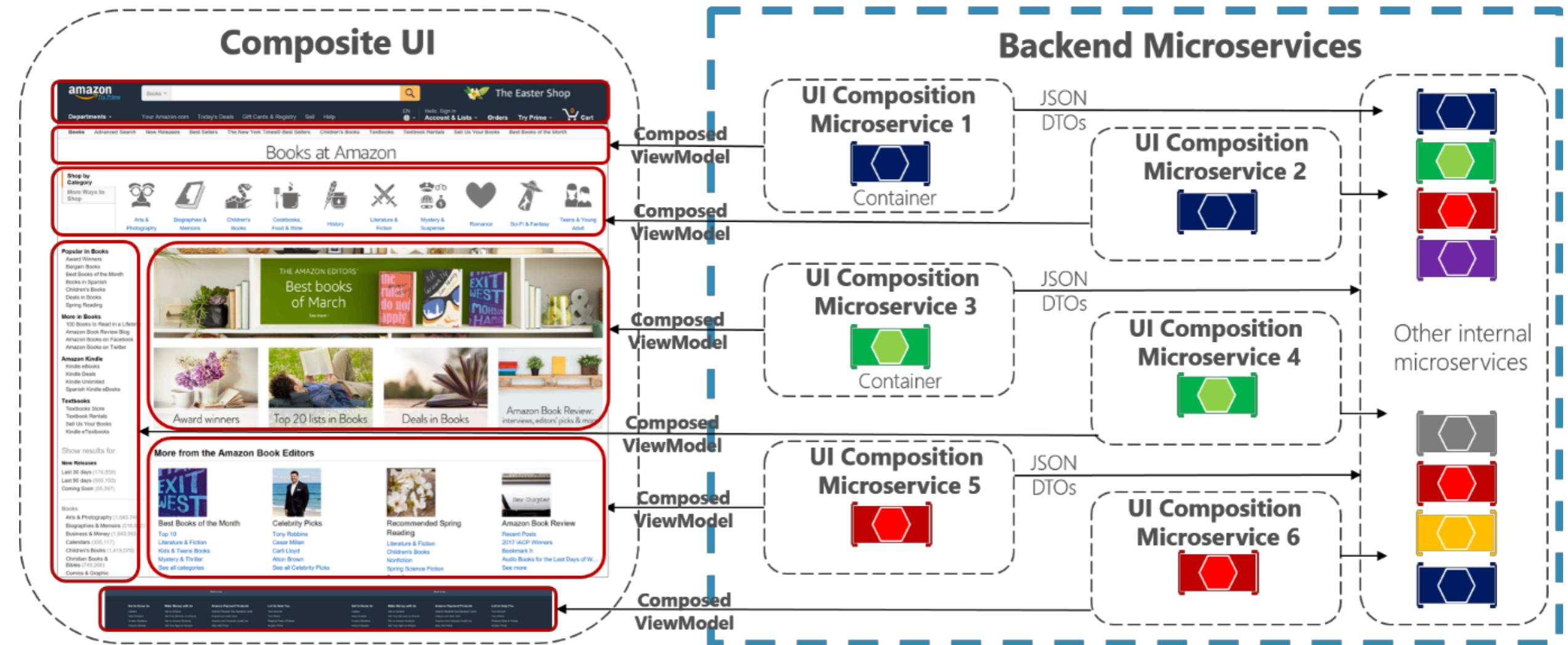
Monolithic UI consuming microservices



<https://docs.microsoft.com>



Composite UI generated by microservices



<https://docs.microsoft.com>

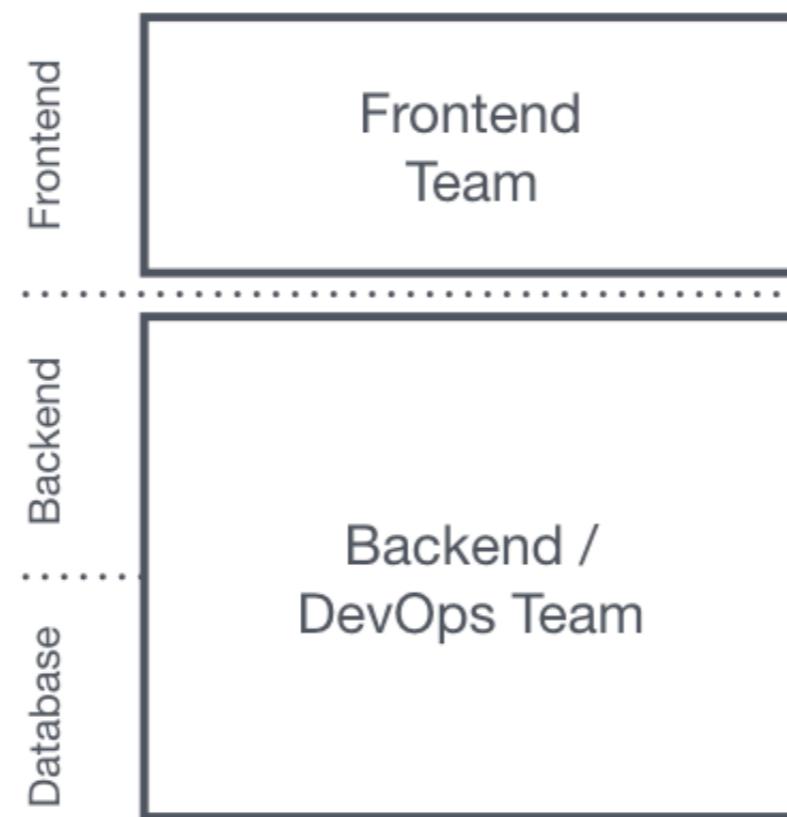


Monolith frontend

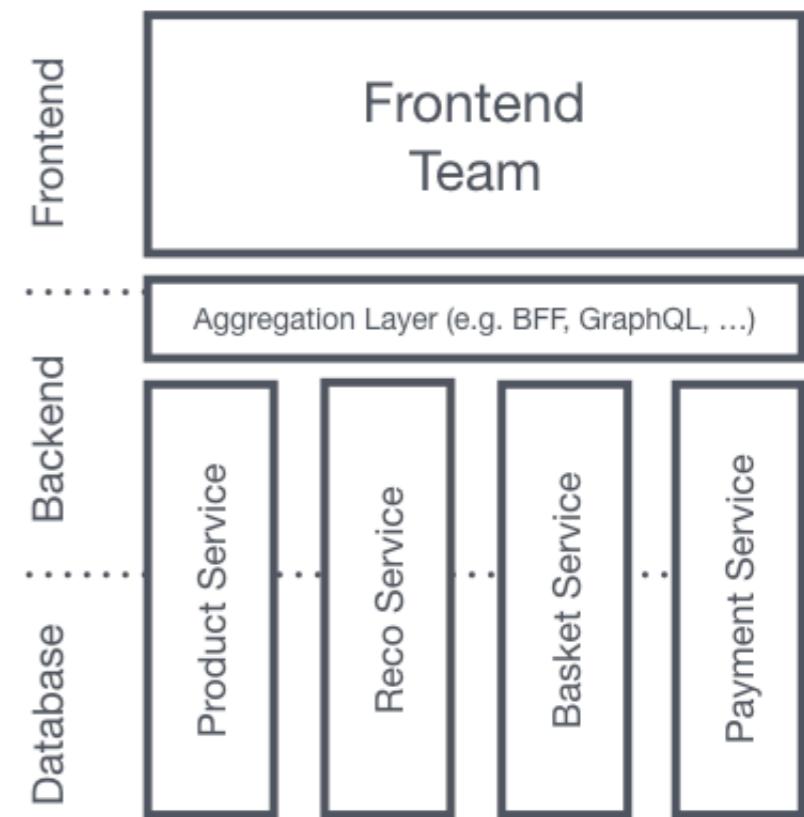
The Monolith



Front & Back



Microservices

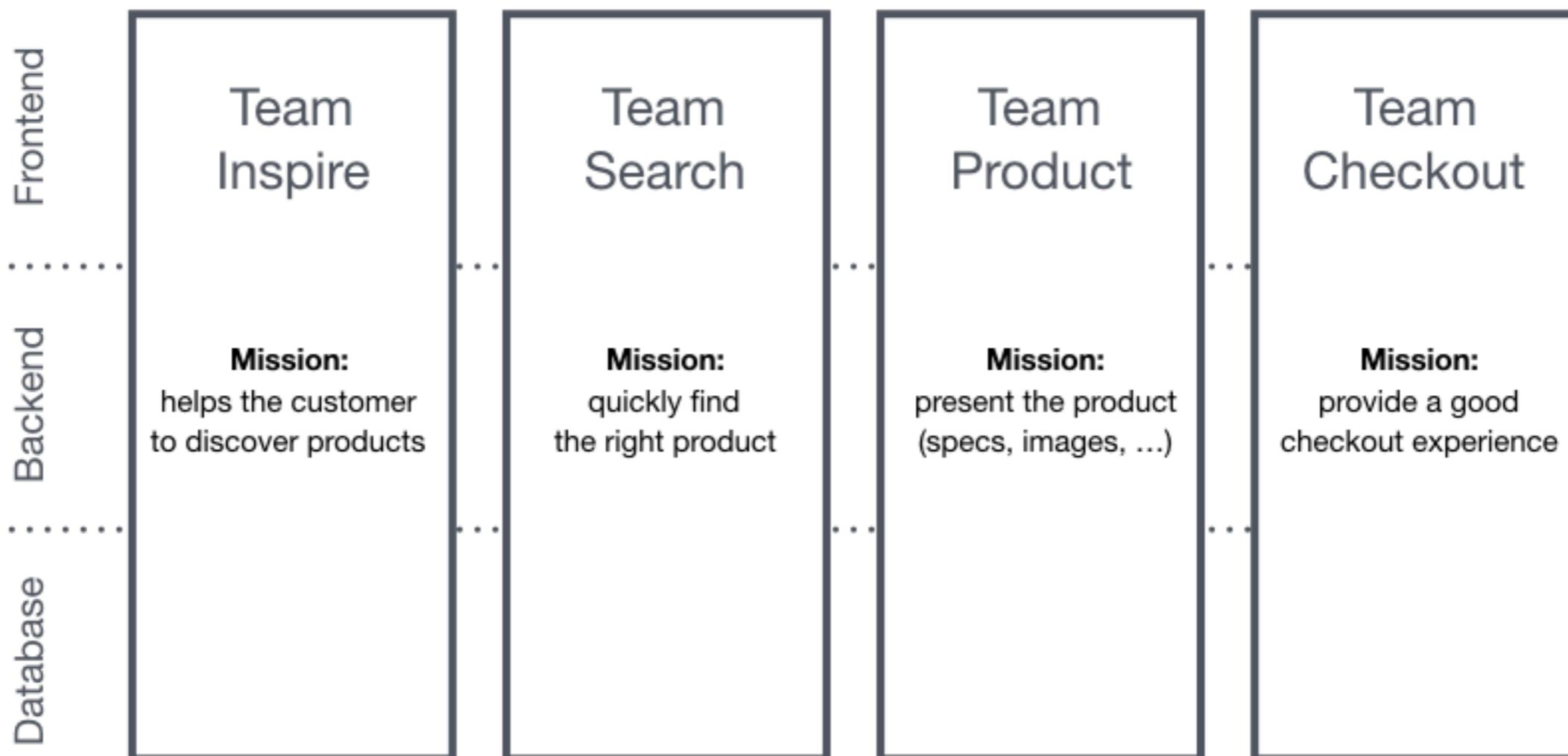


<https://micro-frontends.org/>



Micro frontend

End-to-End Teams with Micro Frontends



<https://micro-frontends.org/>



Summary



Let's start with good monolith



Module 1

Module 2

Module 3

Module 4

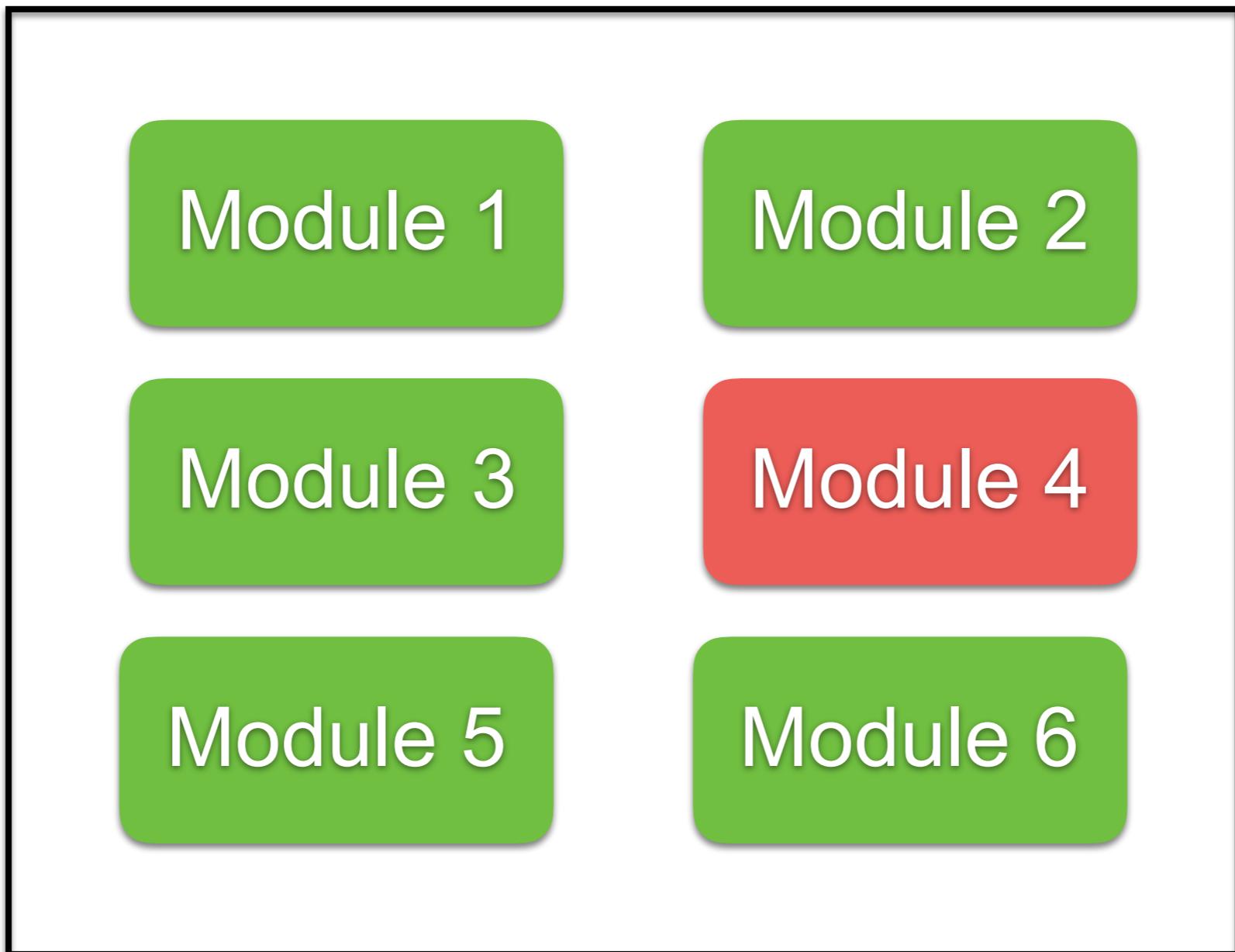
Module 5

Module 6



Find your problem





Module 1

Module 2

Module 3

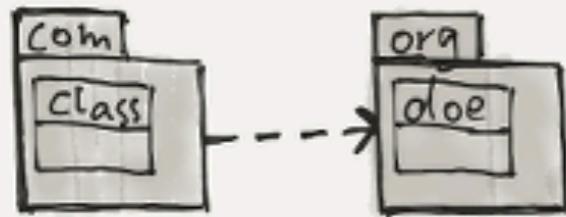
Module 5

Module 6

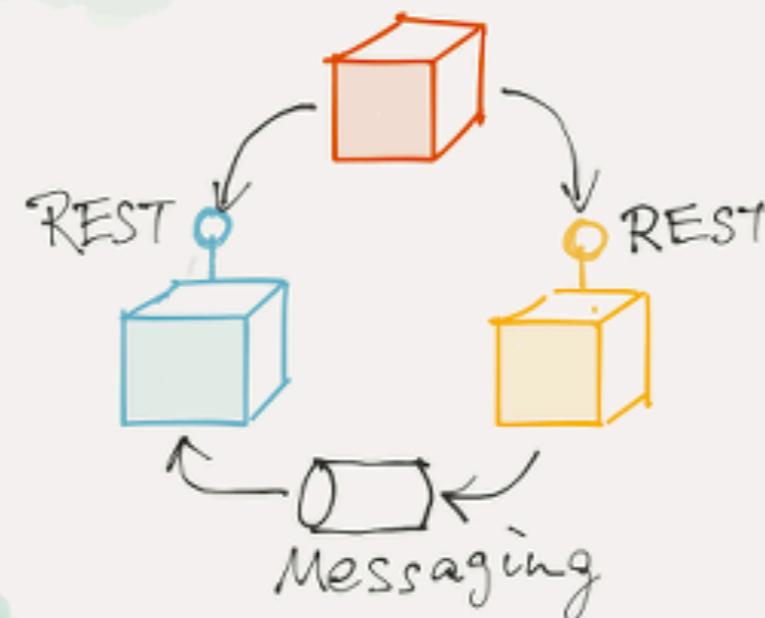
Module 4



Architecture



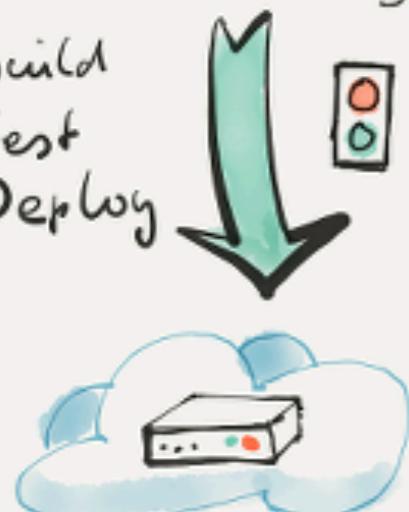
Microservices



Deployment

Continuous Delivery

`{ var i=1; }`
Build
Test
Deploy



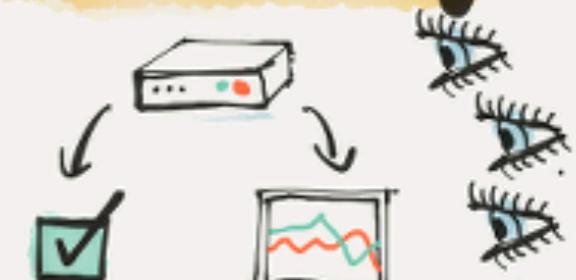
Infrastructure



People & Teams



Monitoring



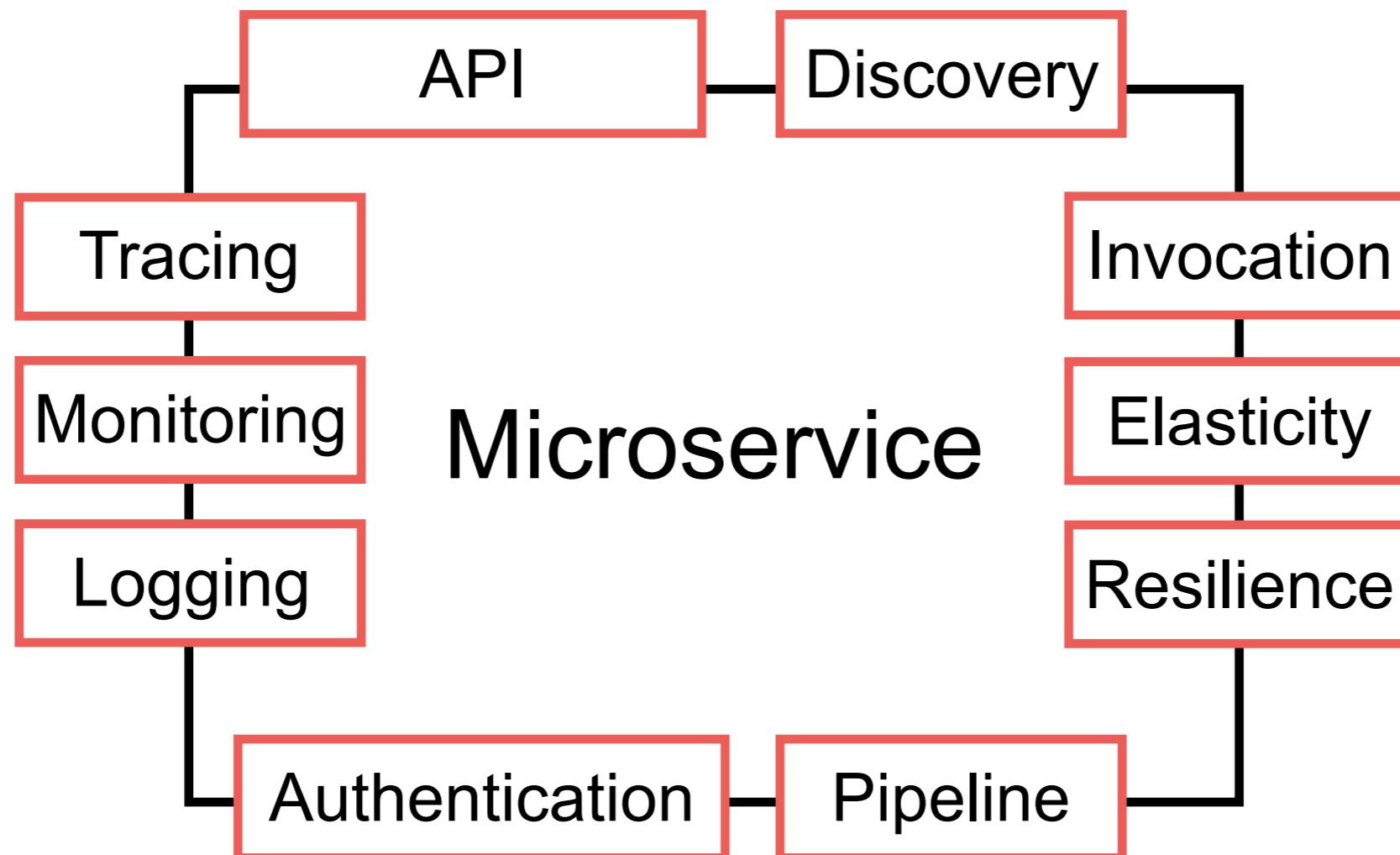
Features & Technology



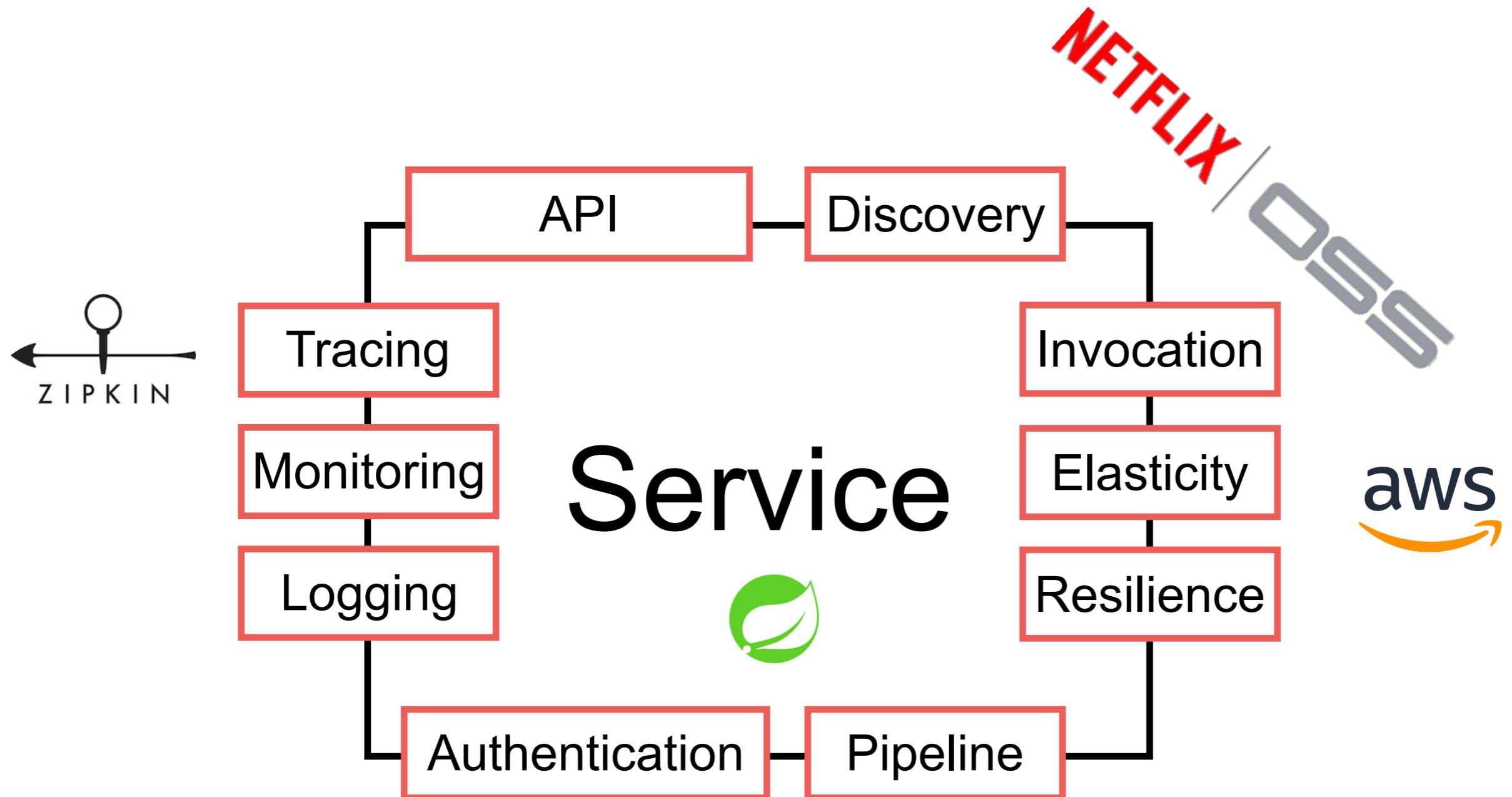
Module 2 : Develop



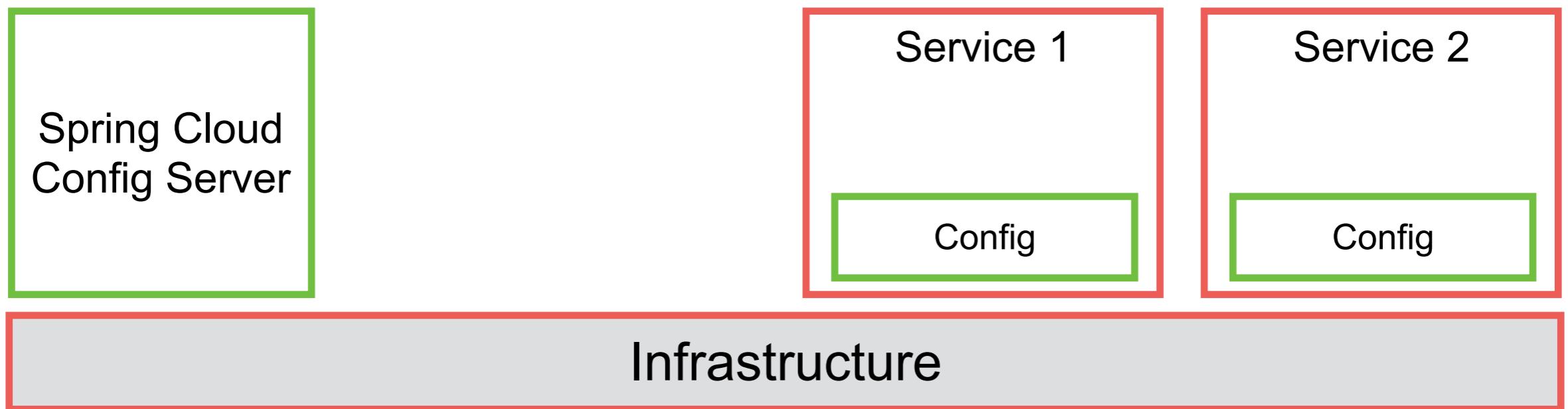
Properties of Microservice



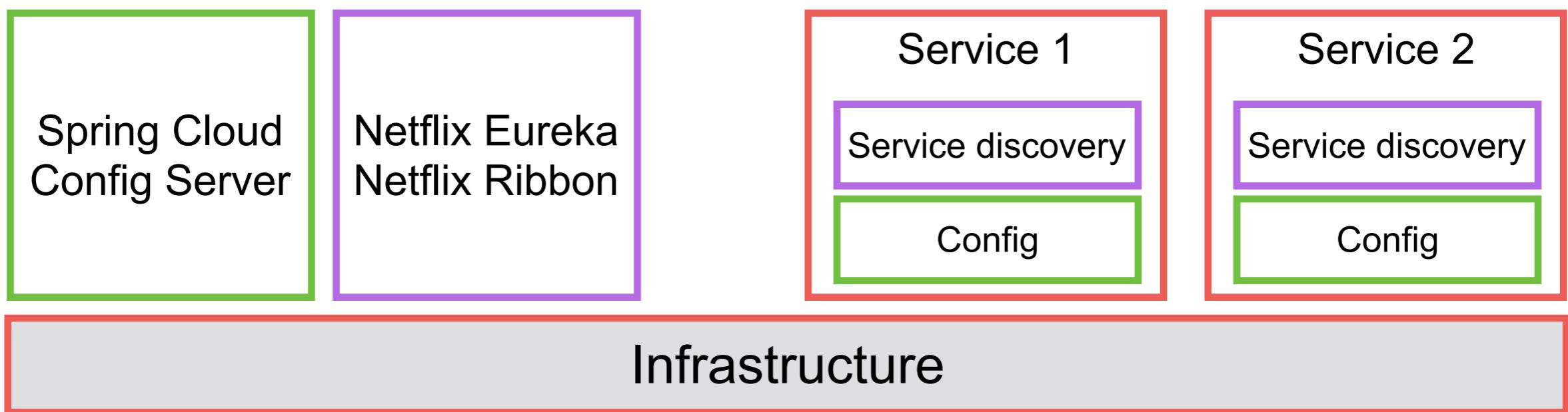
Microservice 1.0



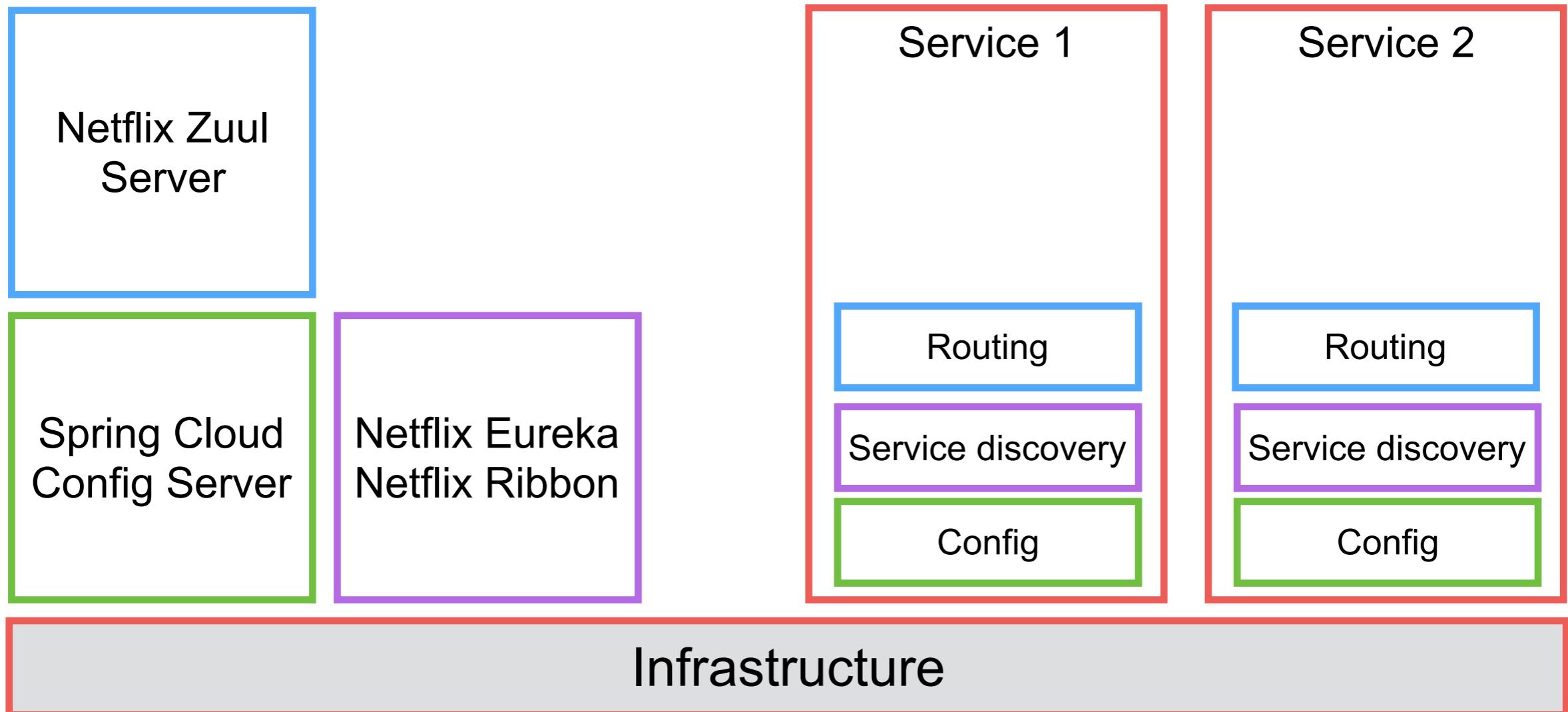
Configuration



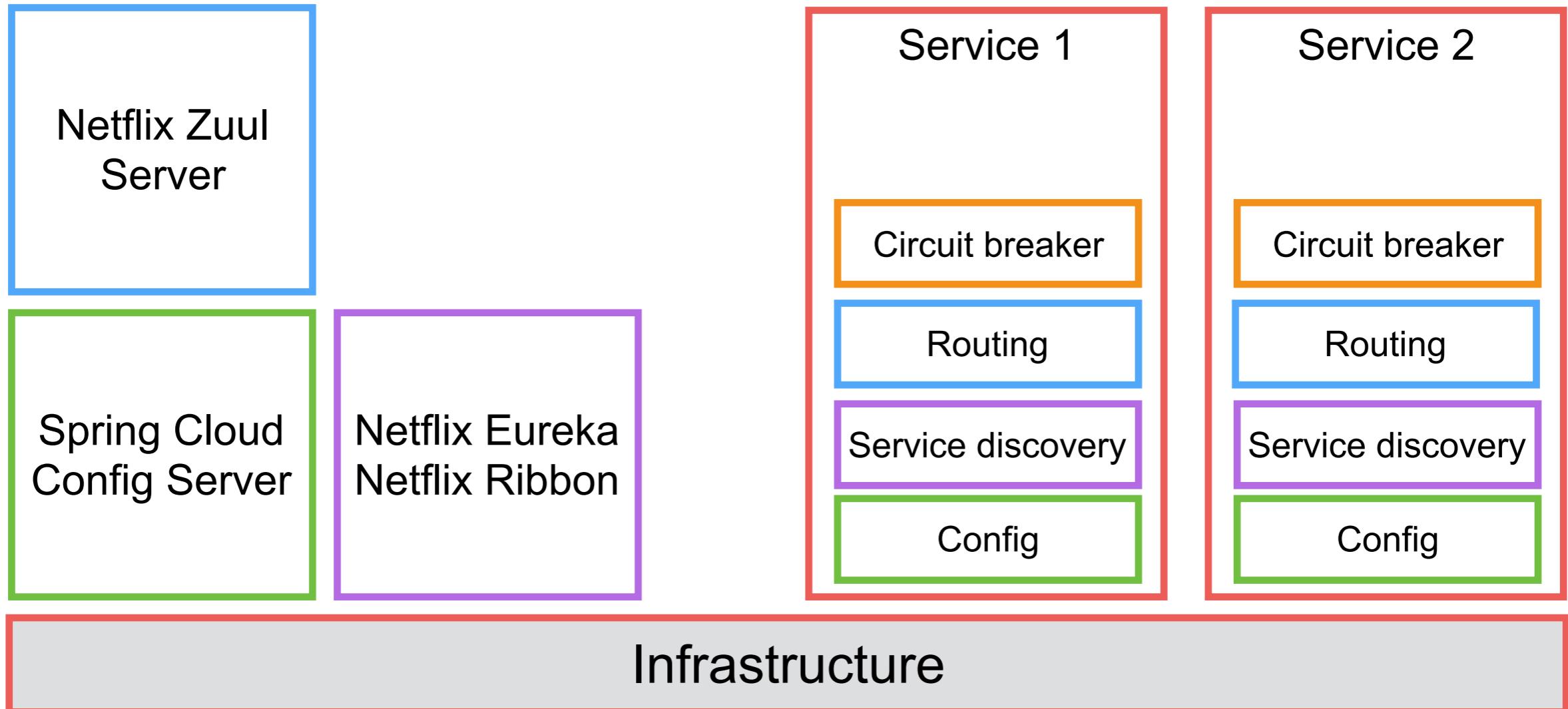
Service Discovery



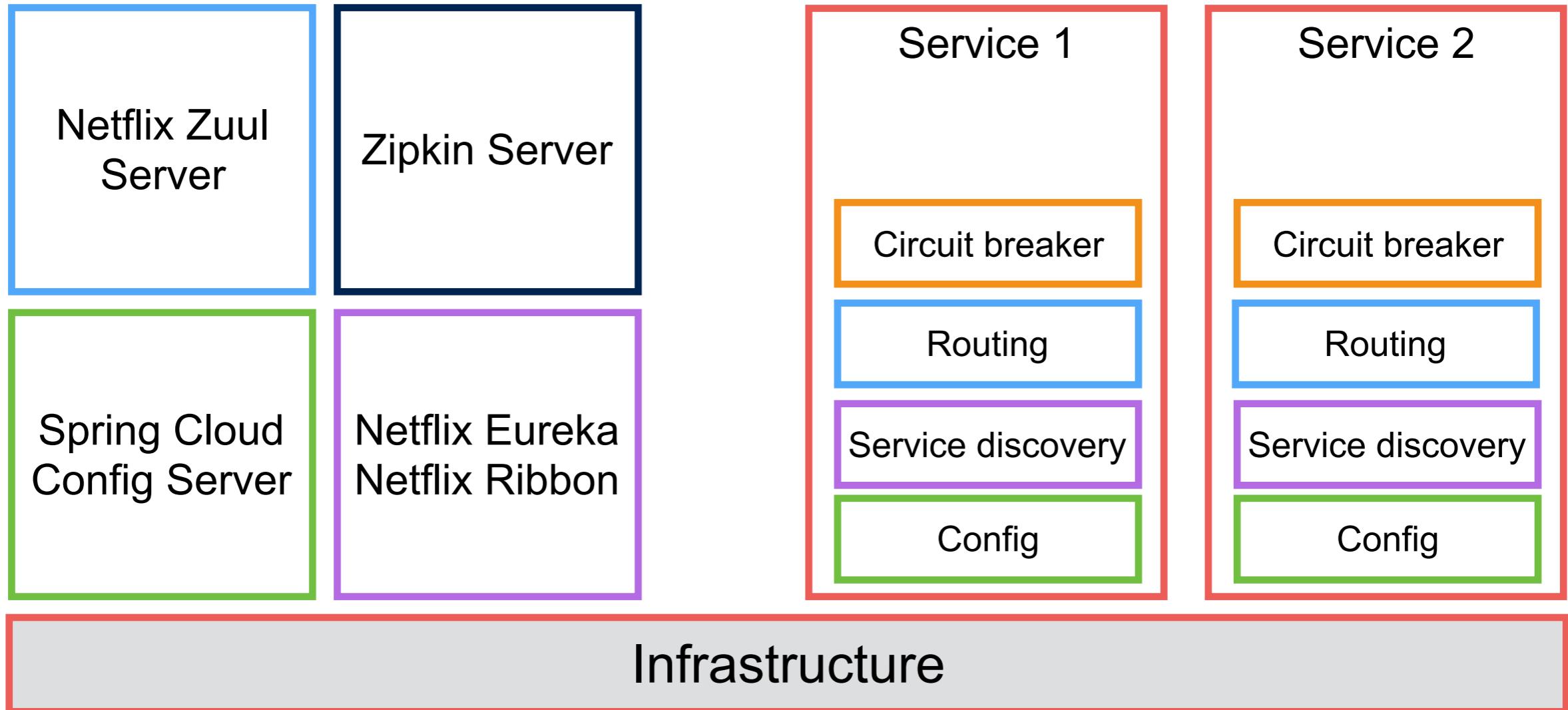
Dynamic Routing



Fault Tolerance



Tracing and Visibility



Microservice 1.0

JVM only

Add libraries to your code/service

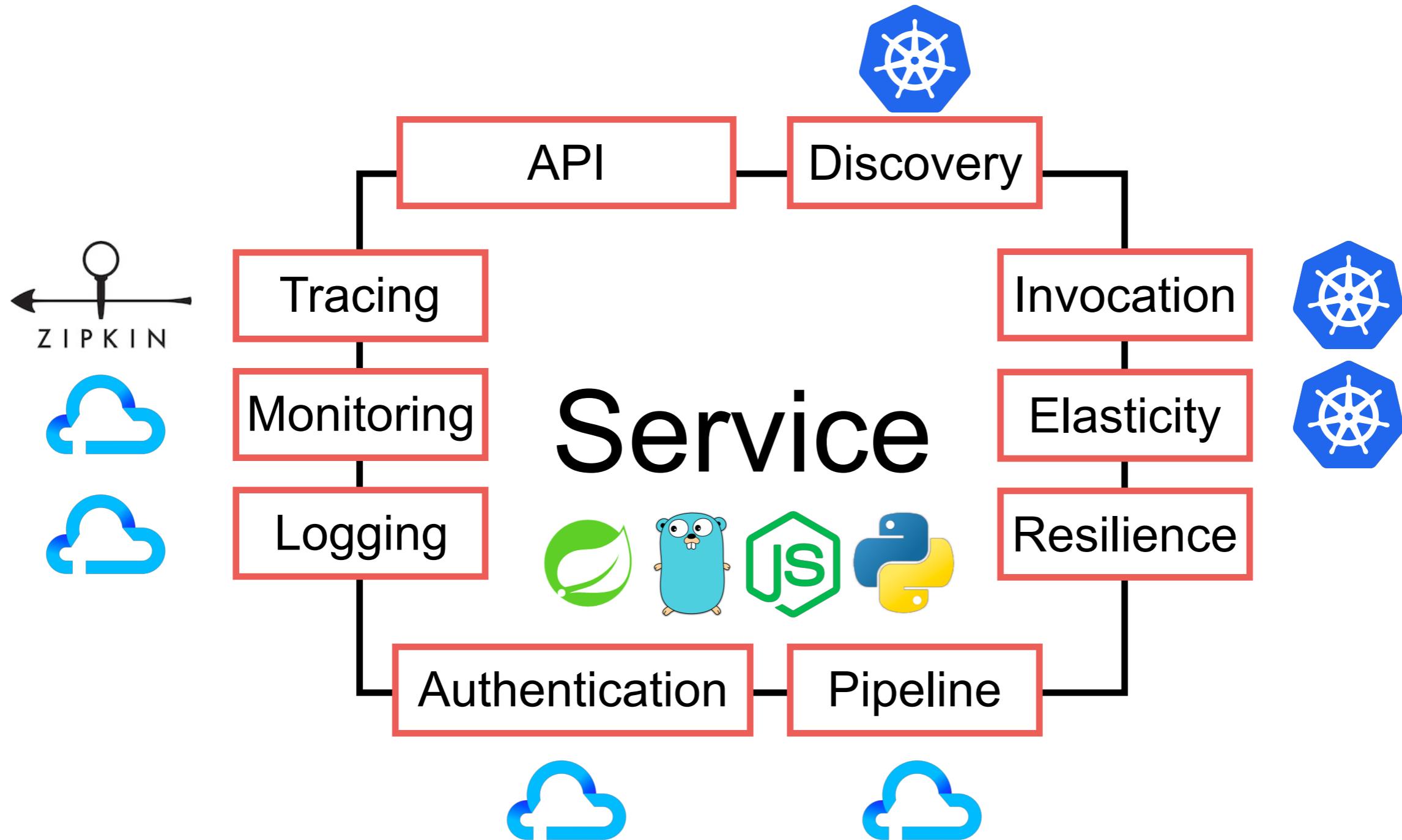


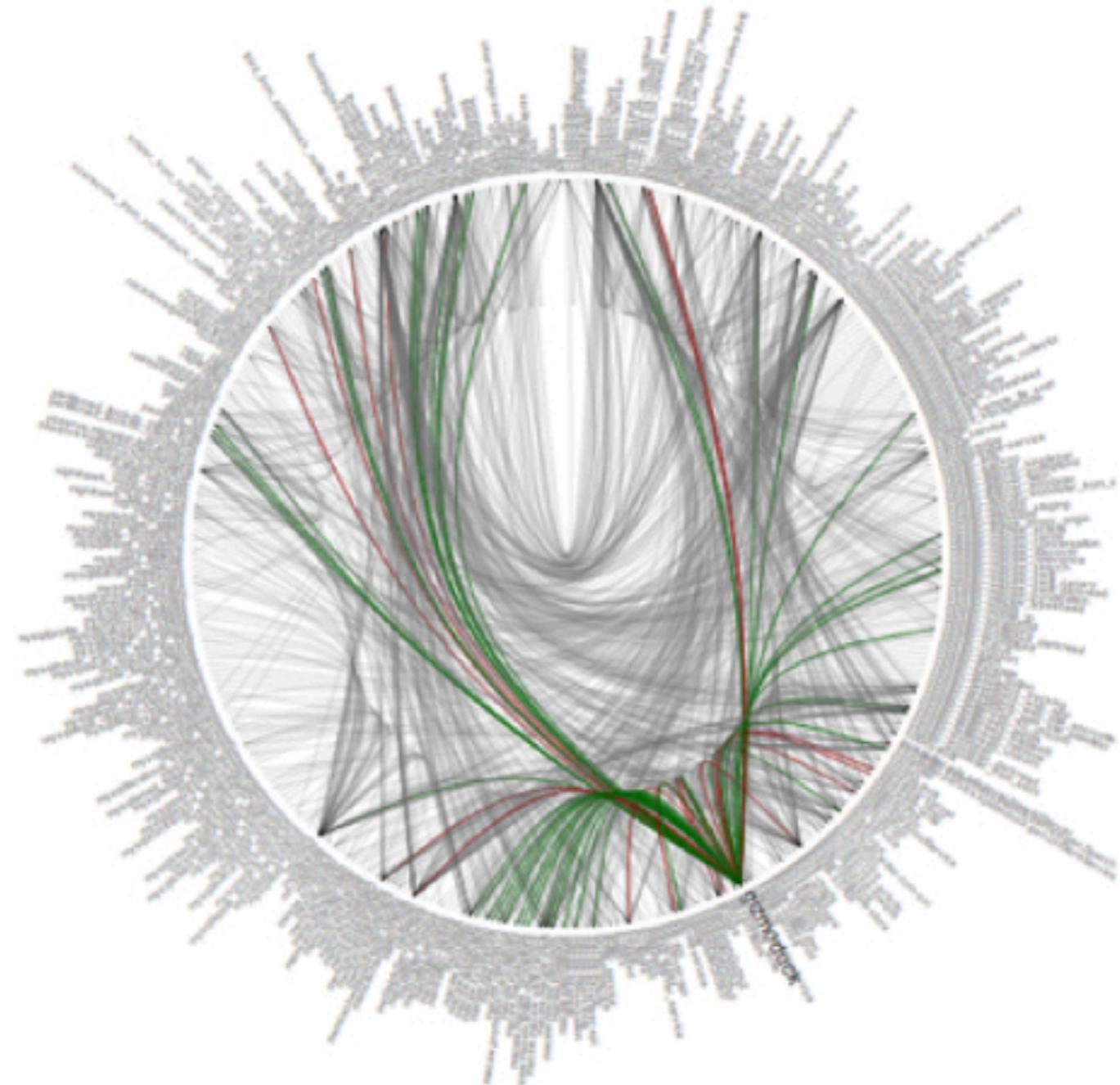


kubernetes



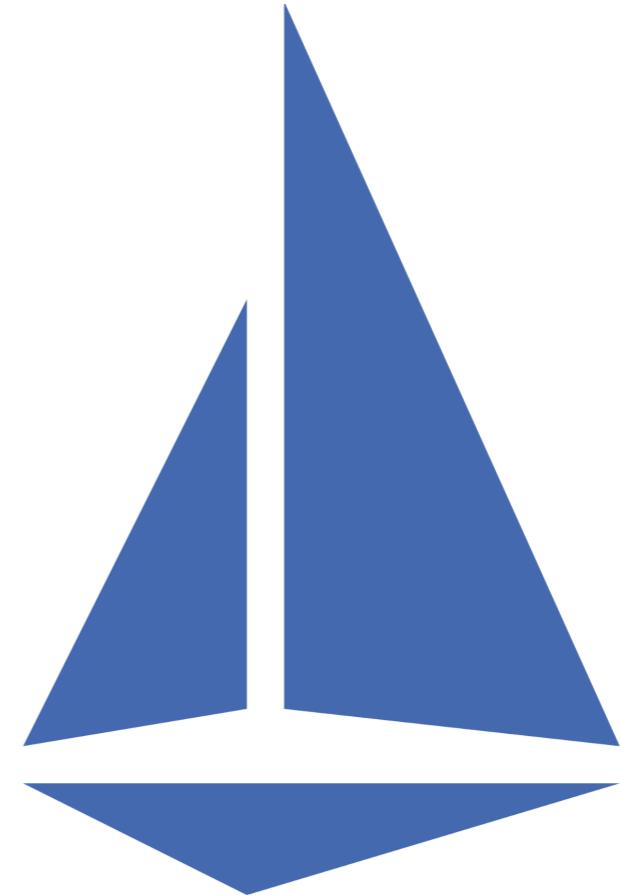
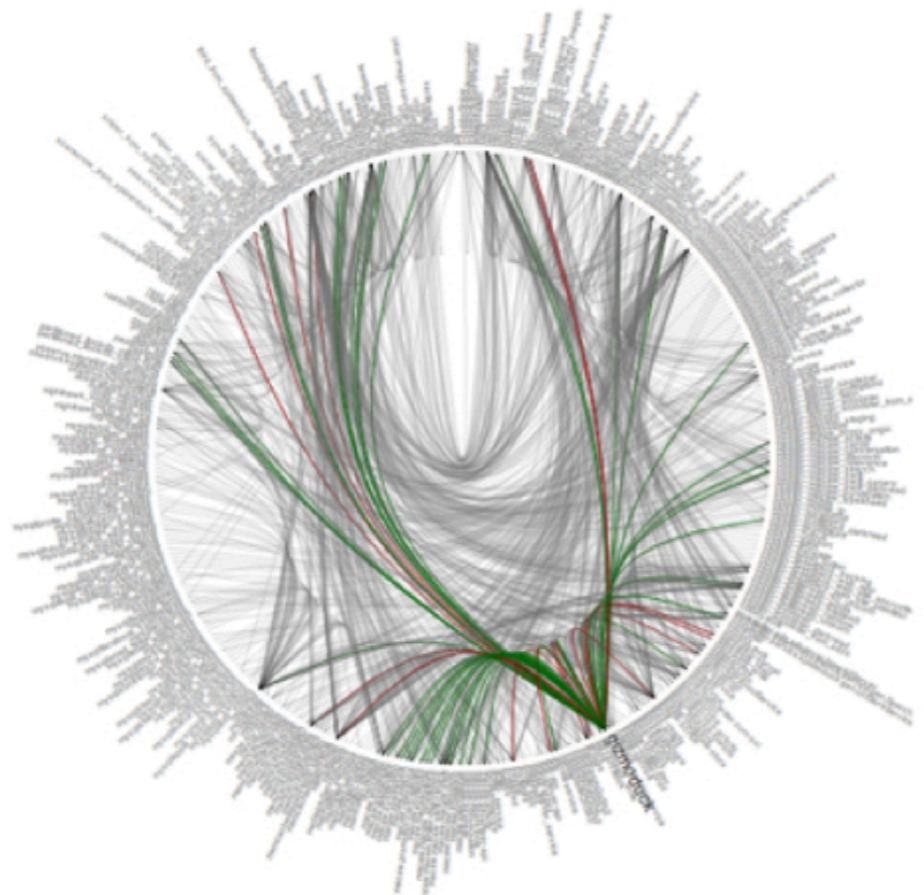
Microservice 2.0





Service Mesh

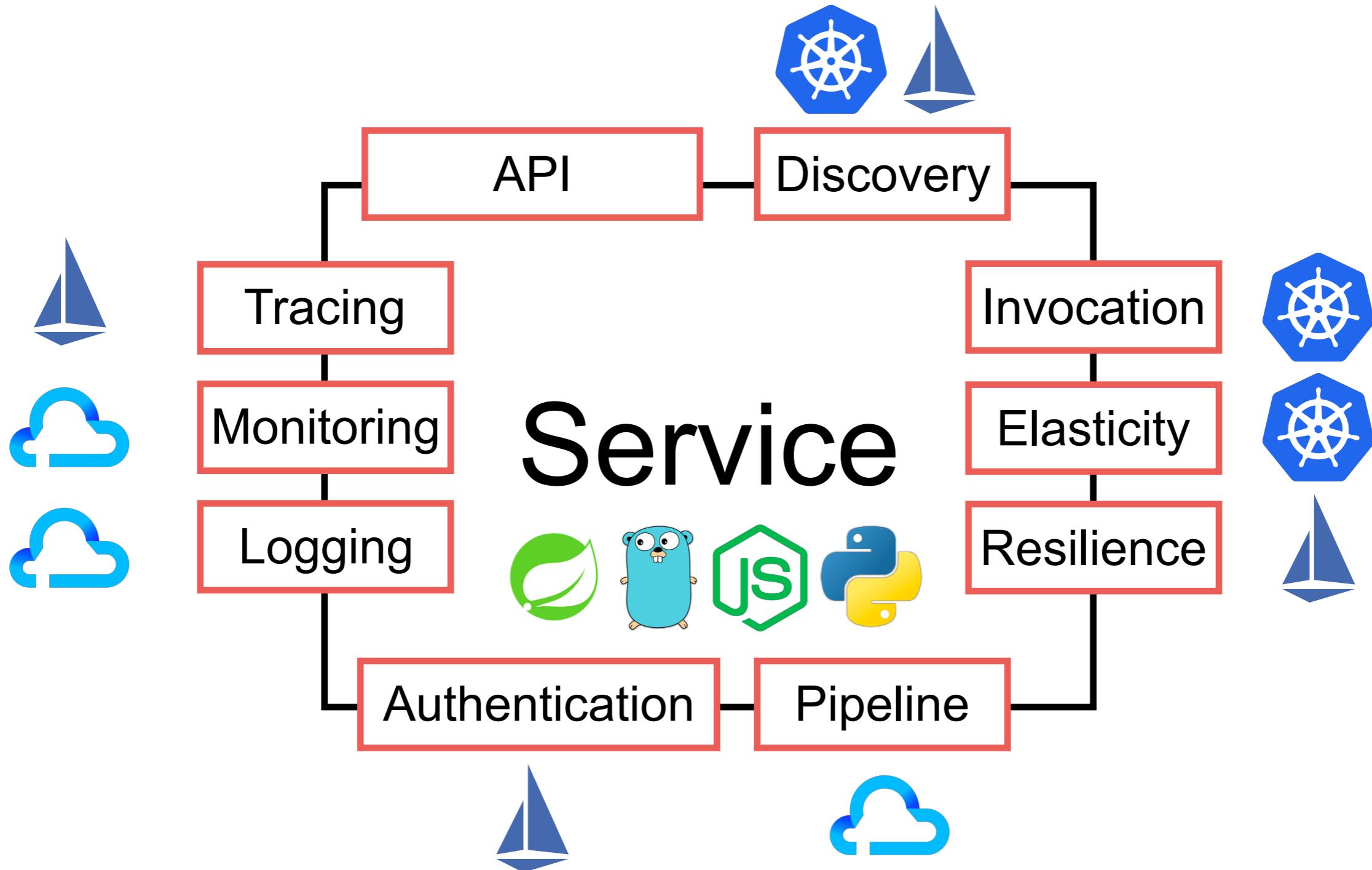




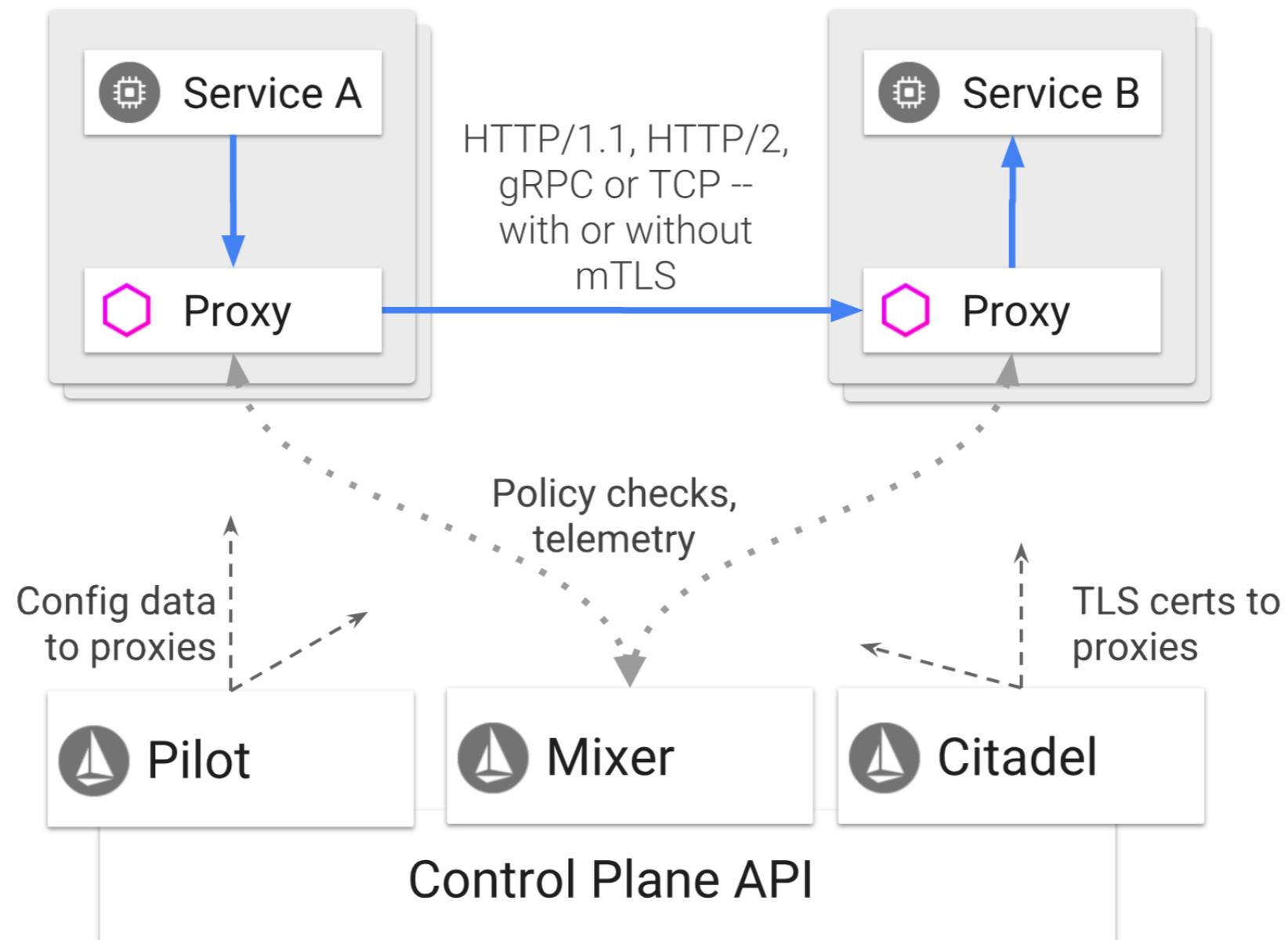
Service Mesh with Istio



Microservice 3.0



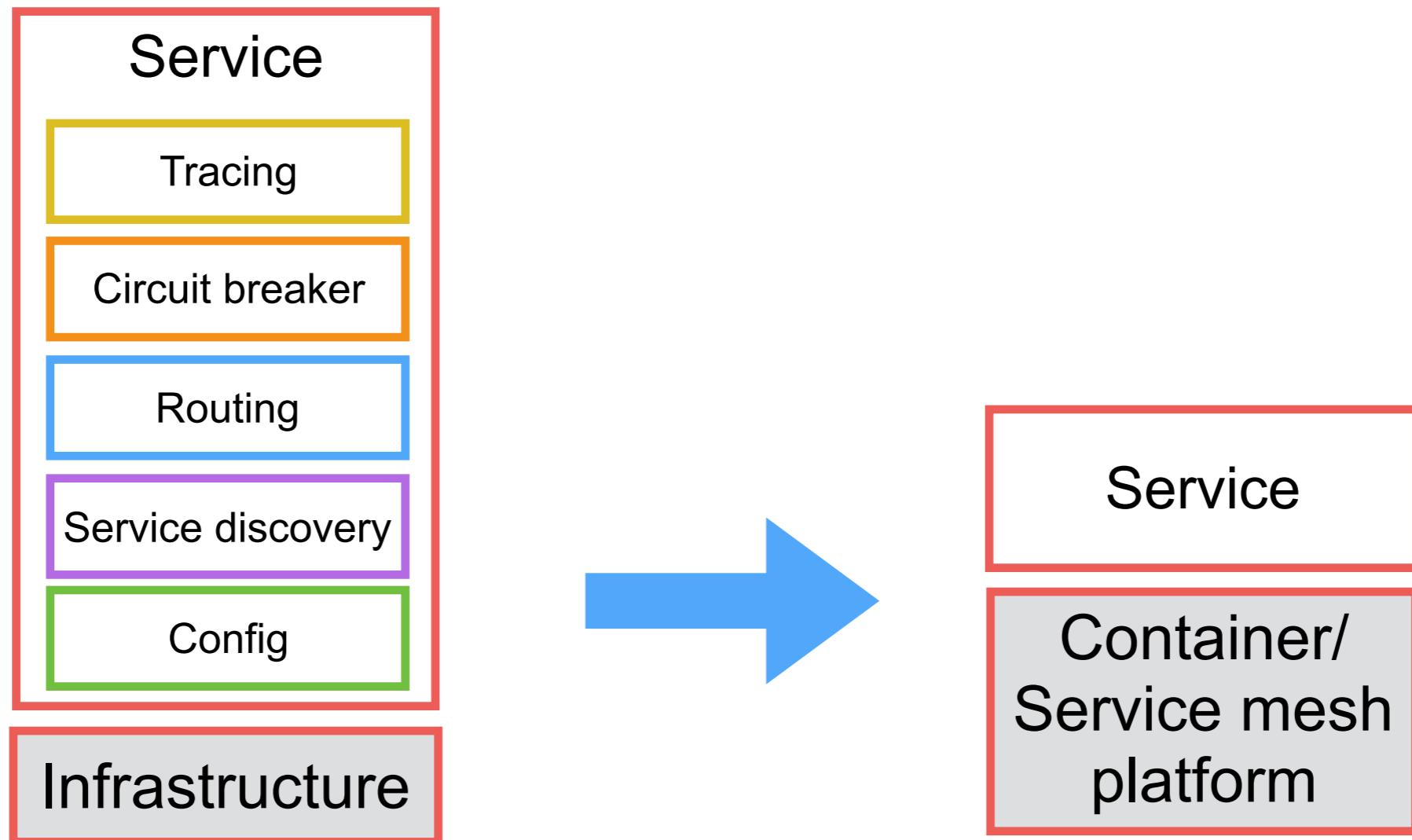
Istio



<https://istio.io/docs/concepts/what-is-istio/>



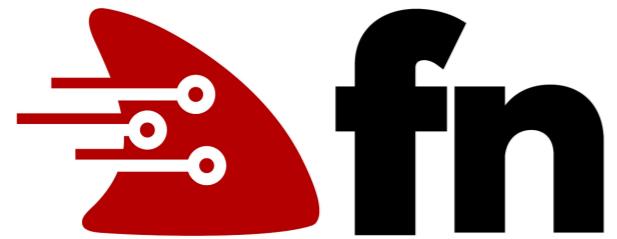
Microservice Evolution



Function-as-a-Service (FaaS)



Microservice 4.0 == FaaS



OPENFAAS



APACHE
OpenWhisk™



Workshop

Microservice 1.5



Twelve-Factor App



Twelve-Factor App

<https://12factor.net/>

Initial to build app for Heroku

Principles to cloud and container native app



1. Codebase

Codebase must be tracked in version control
and will have many deploy



2. Dependencies

Dependencies are explicitly declared and isolated

Use dependency management tools to shared
libraries



3. Configuration

Store configuration in the environment

Add configuration in environment variables or config files



4. Backing services

Treat backing services as an attached resource

Should easy to deploy and change



5. Build, Release, Run

Always have a build and deploy strategy

Build strategies for repeated builds, versioning of running system and rollback



6. Processes

Execute the application as a stateless process



7. Port Binding

Expose services via port bindings



8. Concurrency

Scale out with the process model



9. Disposability

Quick application startup and shutdown times
Graceful shutdown



10. Dev/prod parity

Application is treated the same way in dev, staging and production

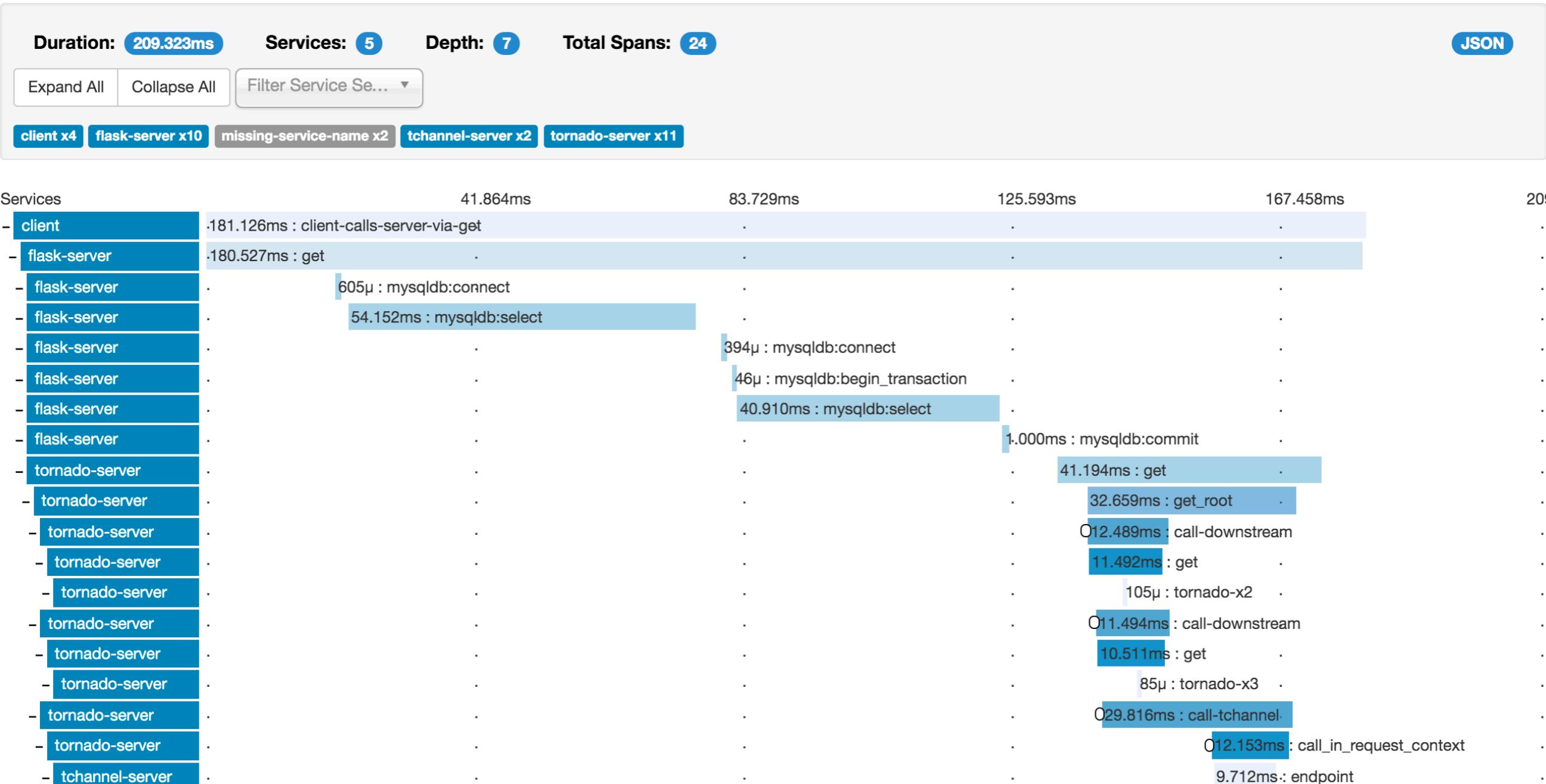


11. Log management

Treated as an event stream



Logging/Tracing



<https://zipkin.io/>



12. Admin tasks

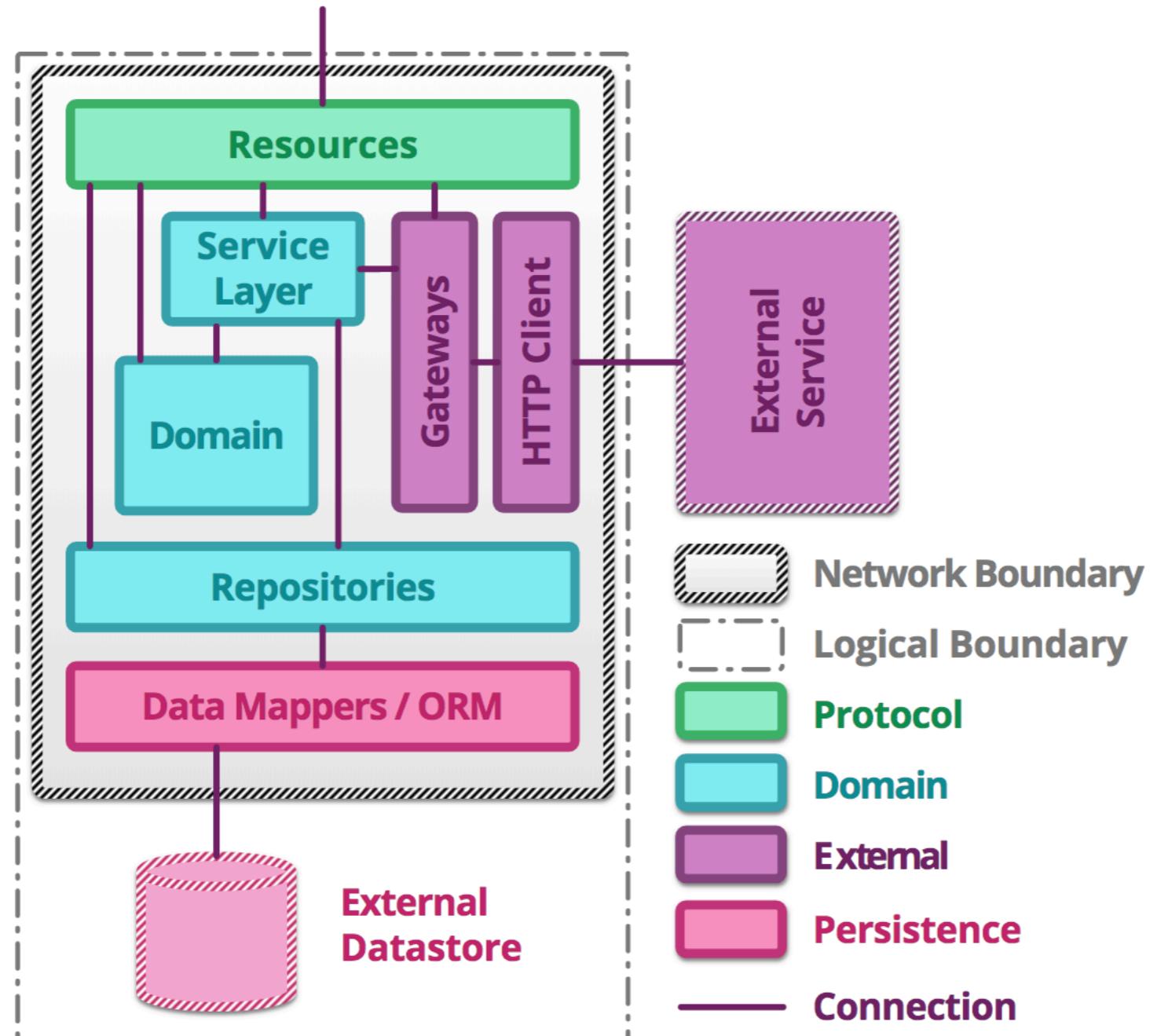
Treated the same way like the rest of the application



Microservice Testing



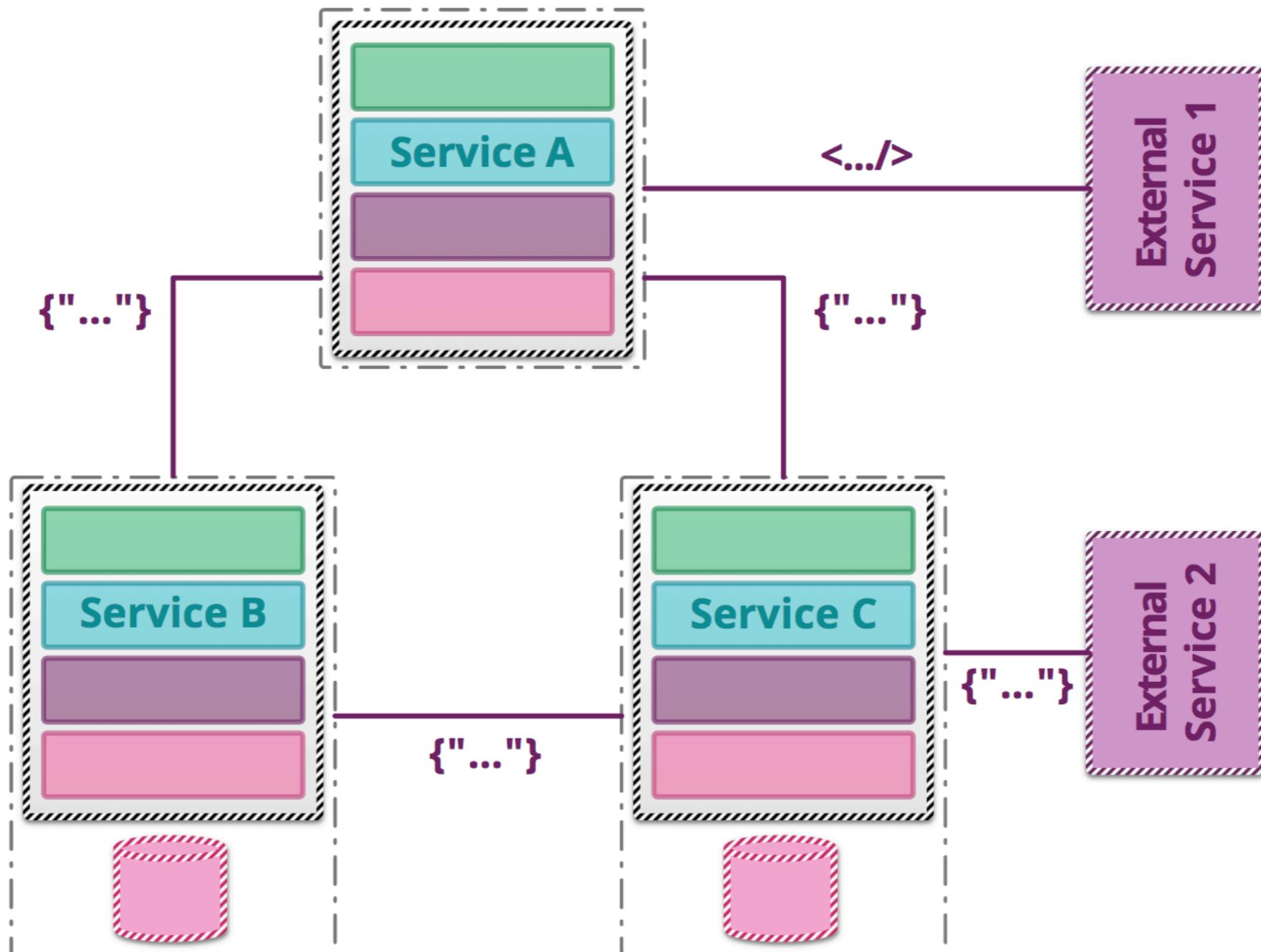
Service structure



<https://martinfowler.com/articles/microservice-testing>



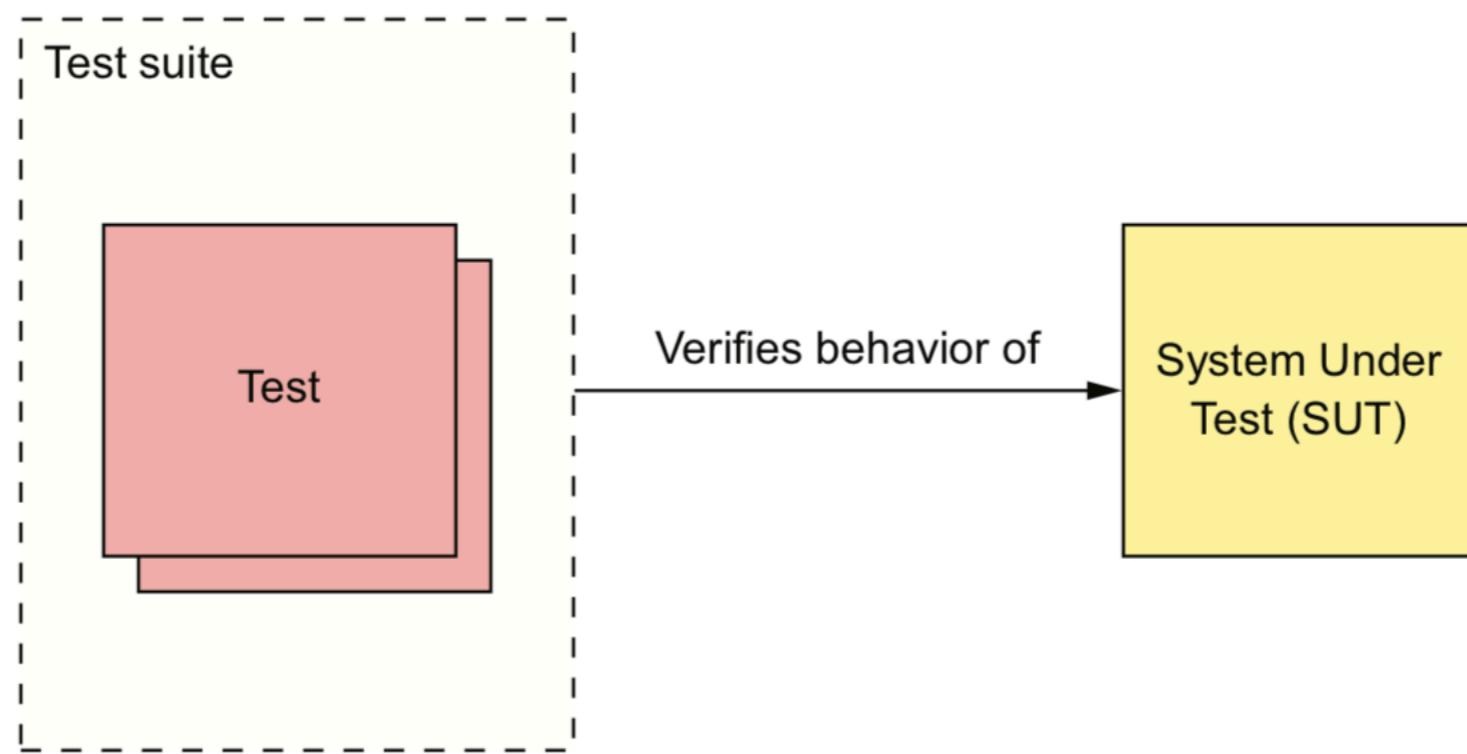
Multiple services



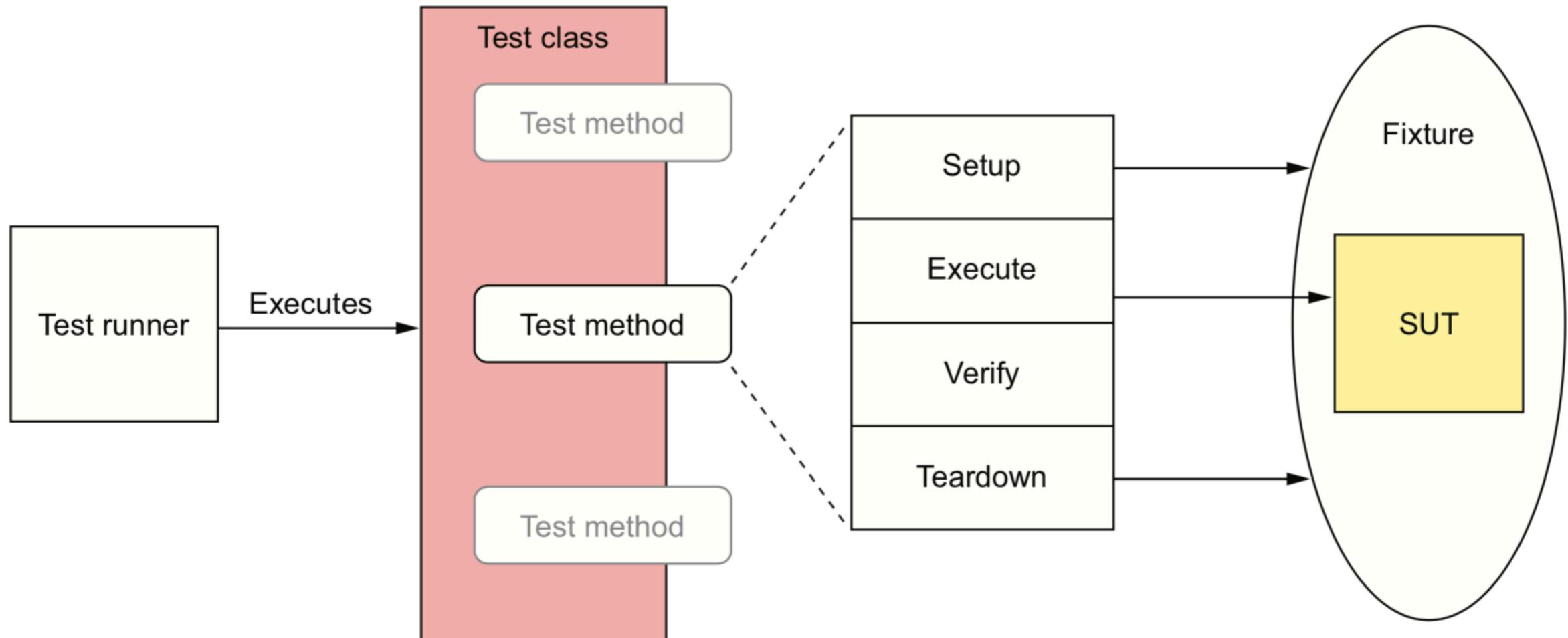
<https://martinfowler.com/articles/microservice-testing>



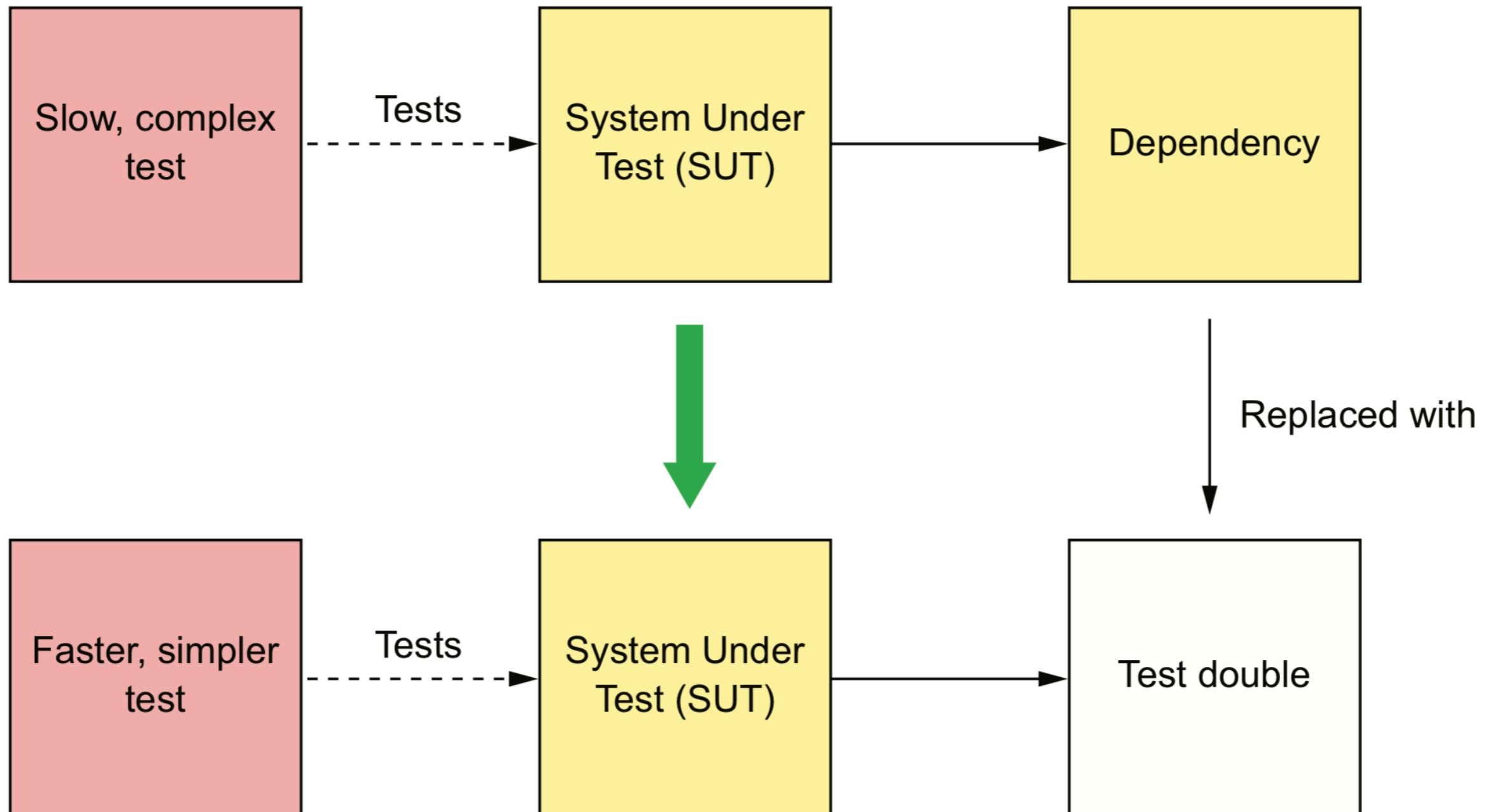
Testing

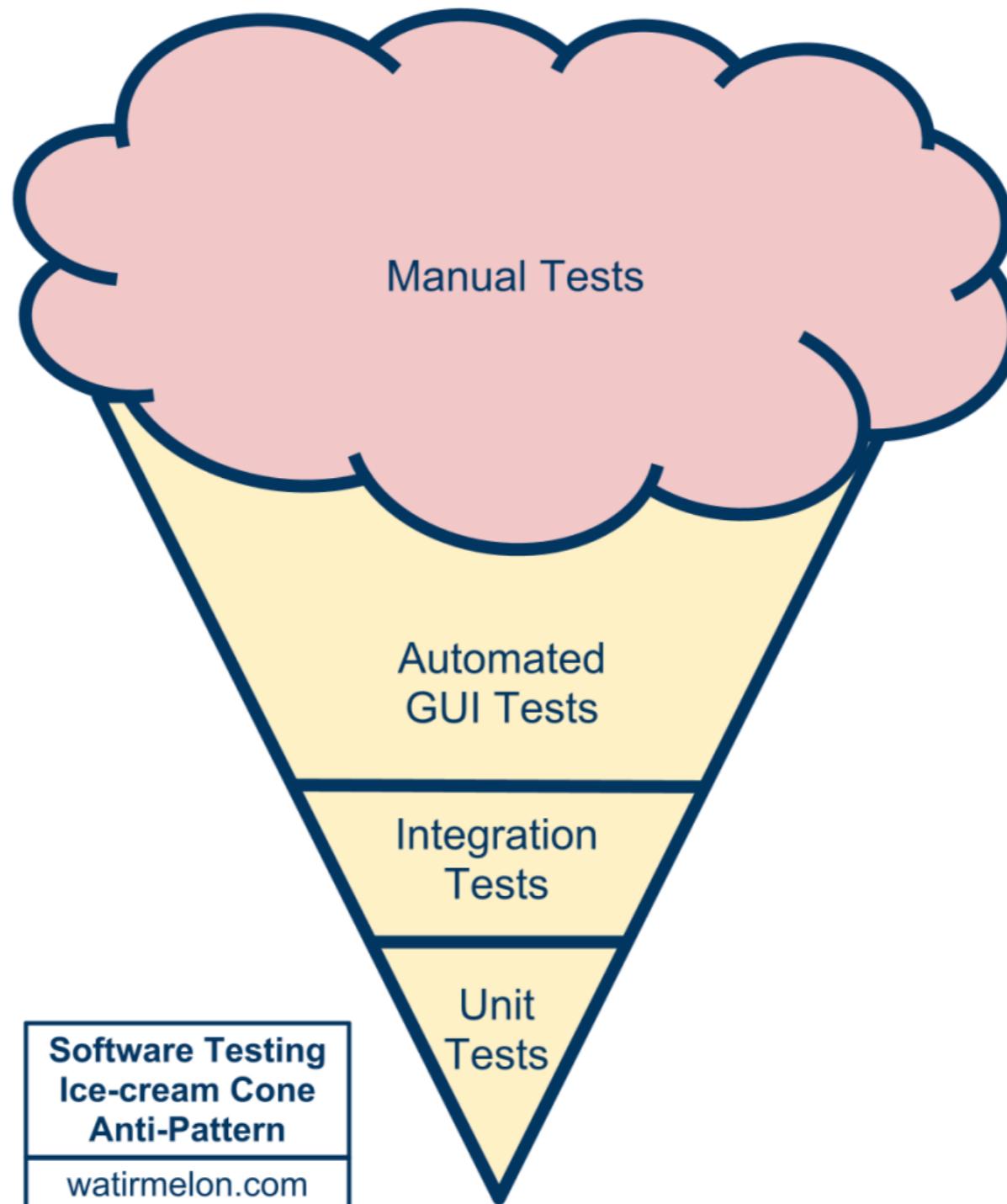


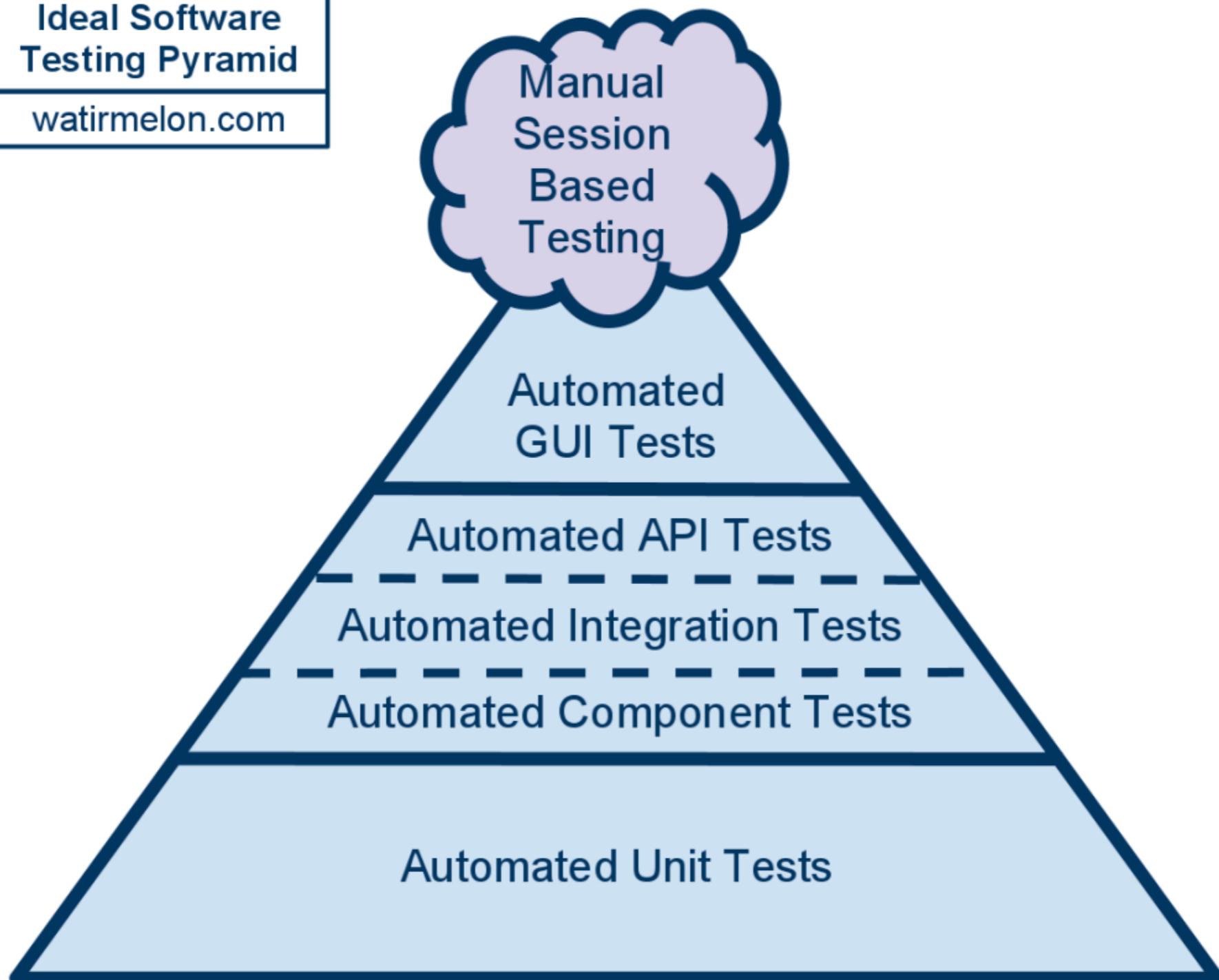
Structure of Automated tests

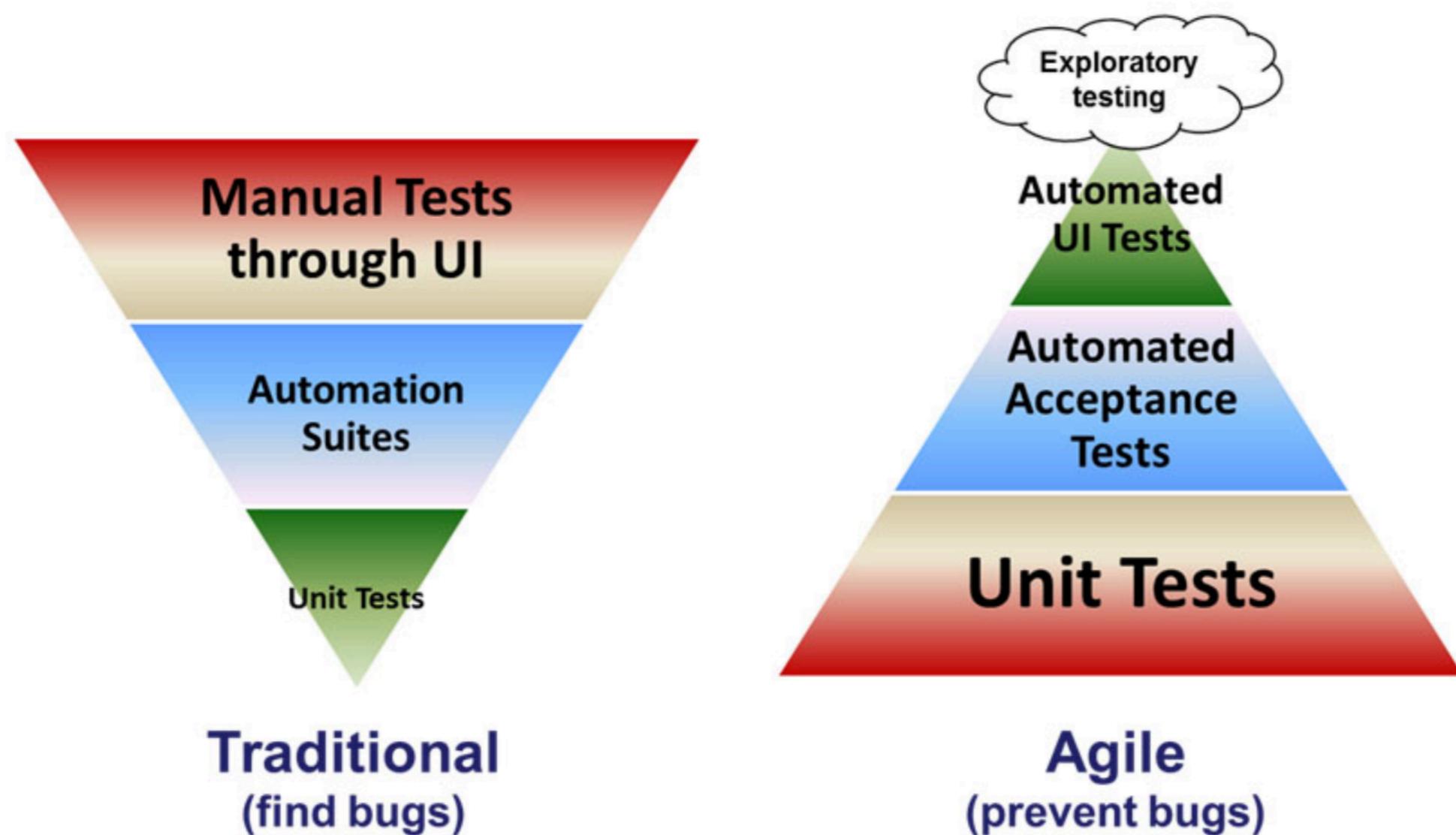


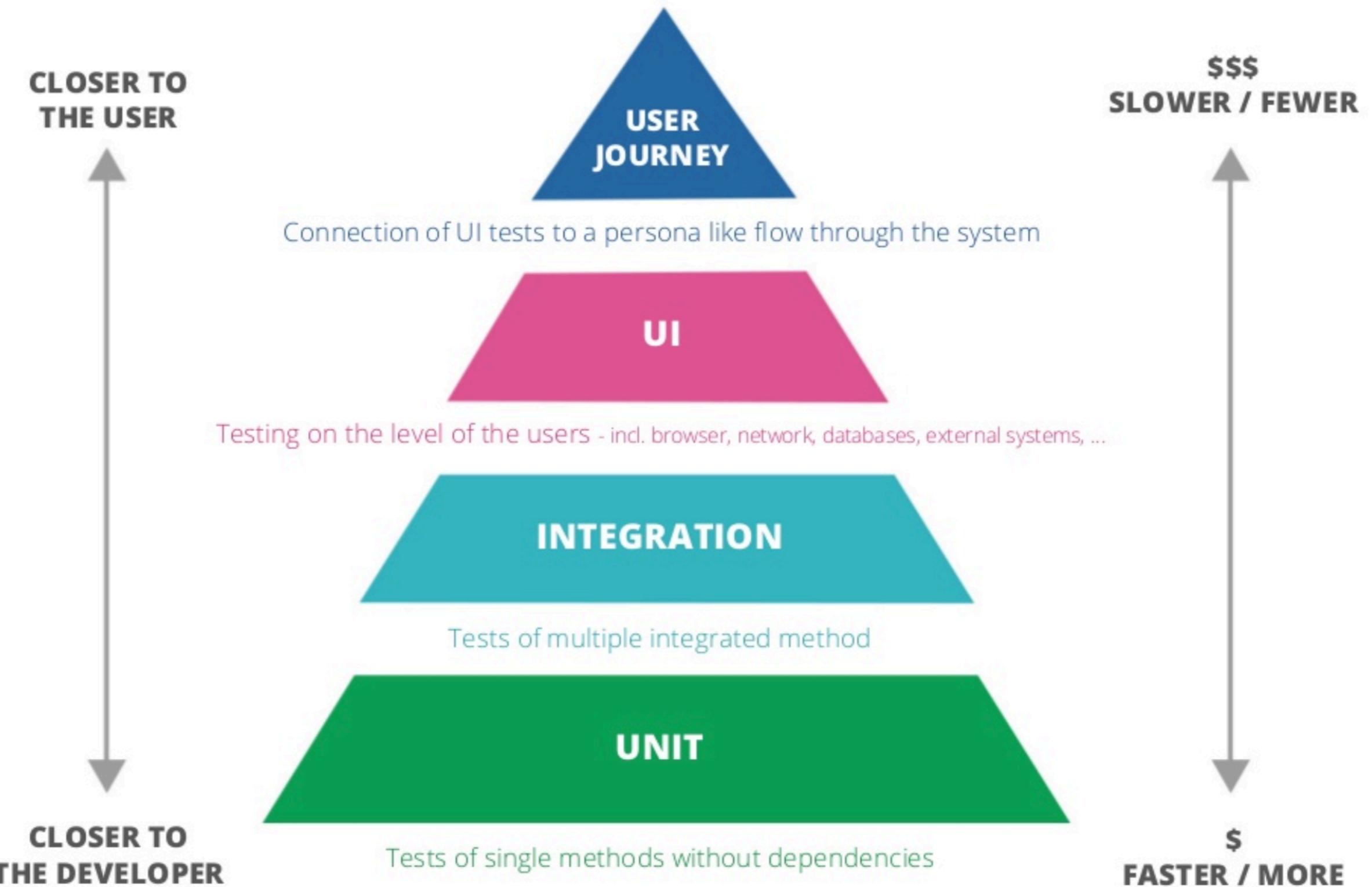
Working with Test double



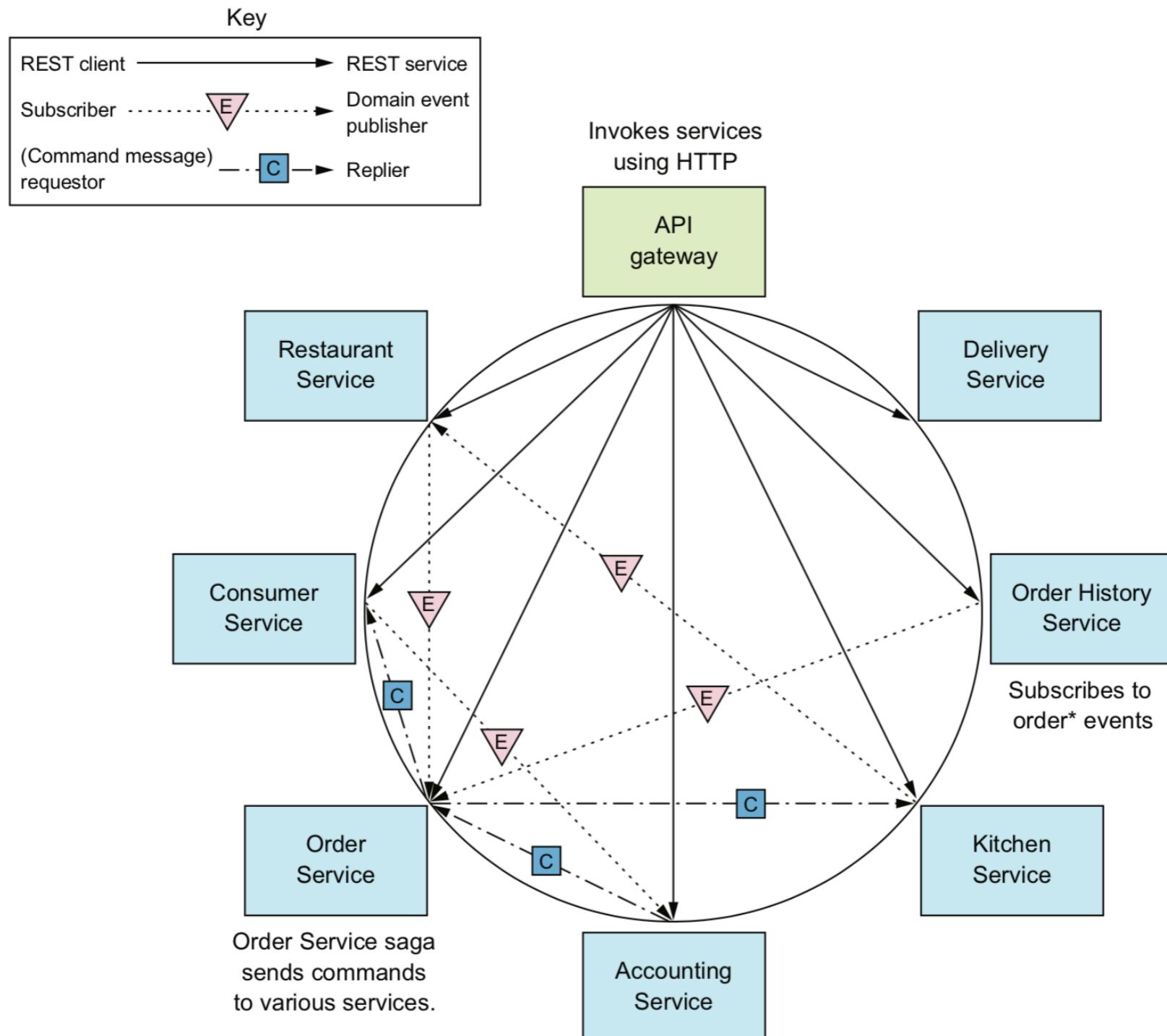






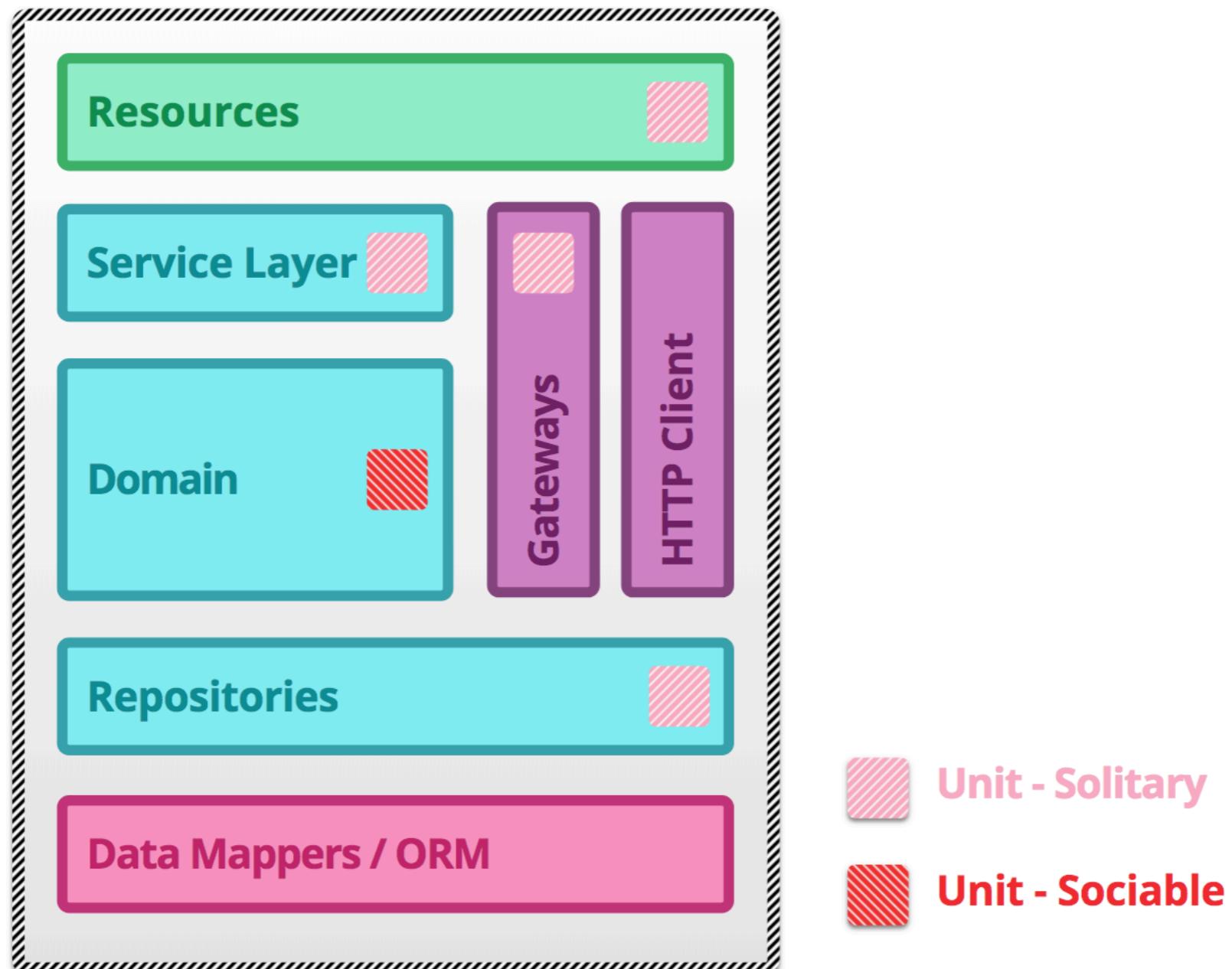


Testing Microservices ?

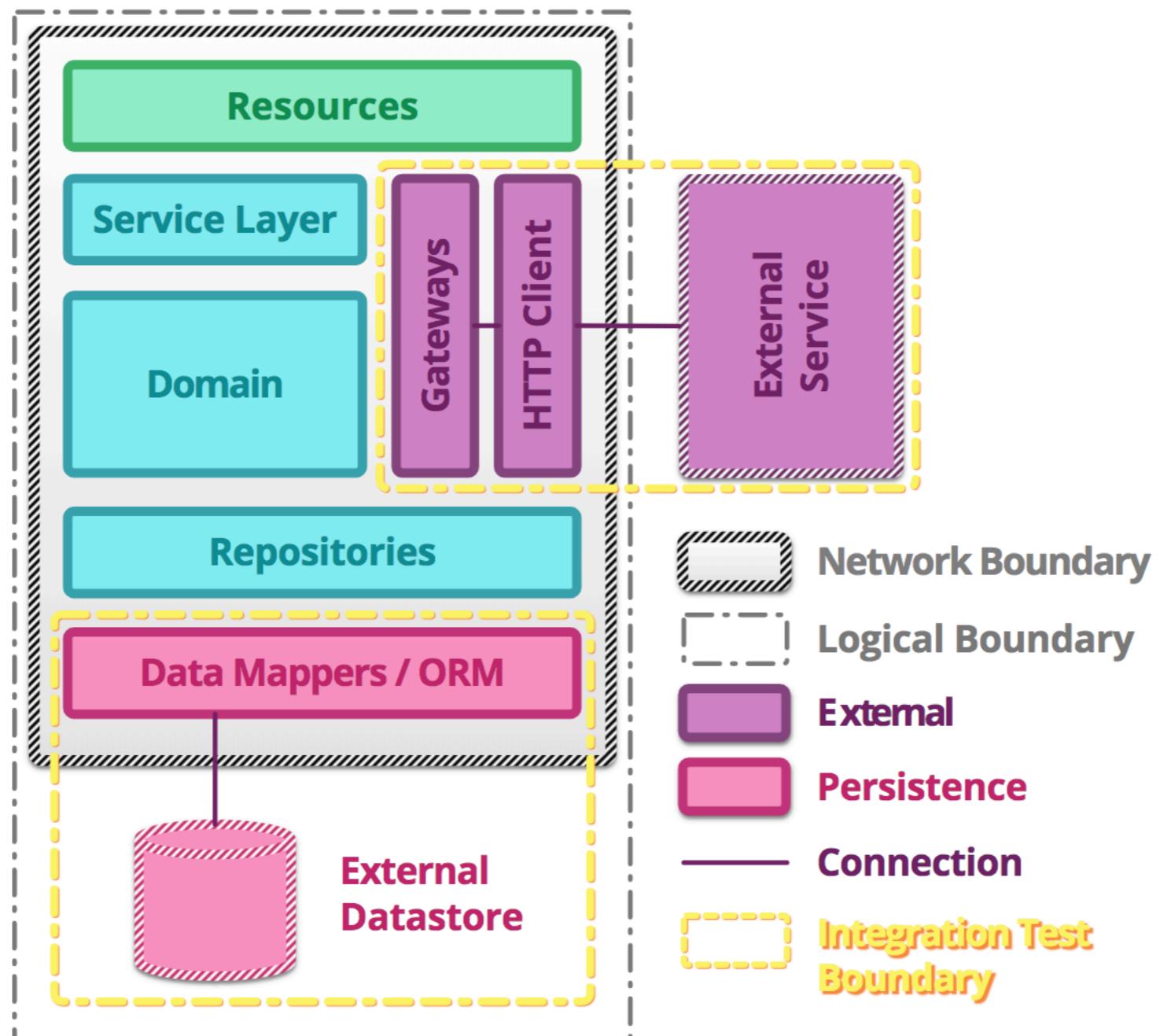




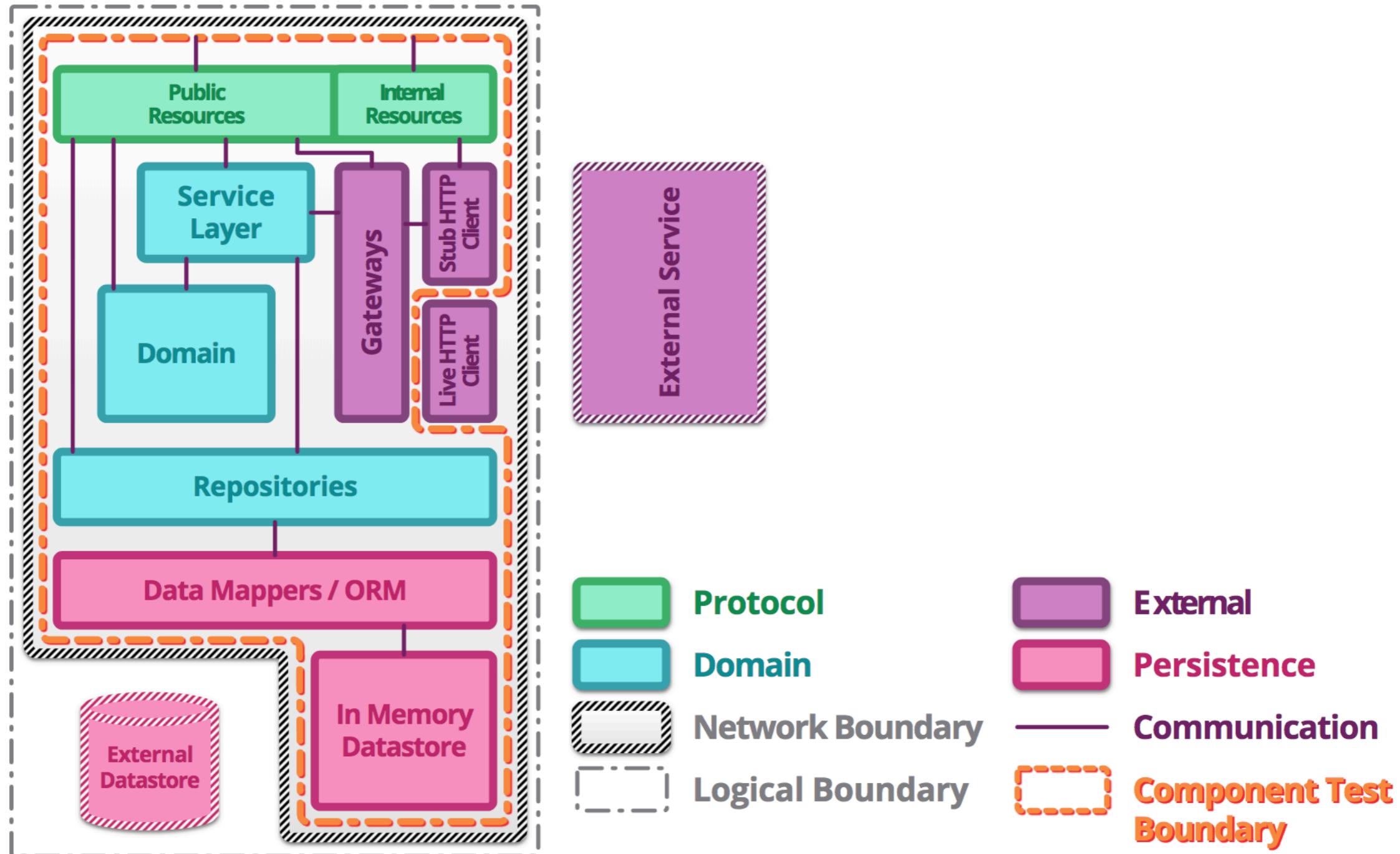
Unit testing



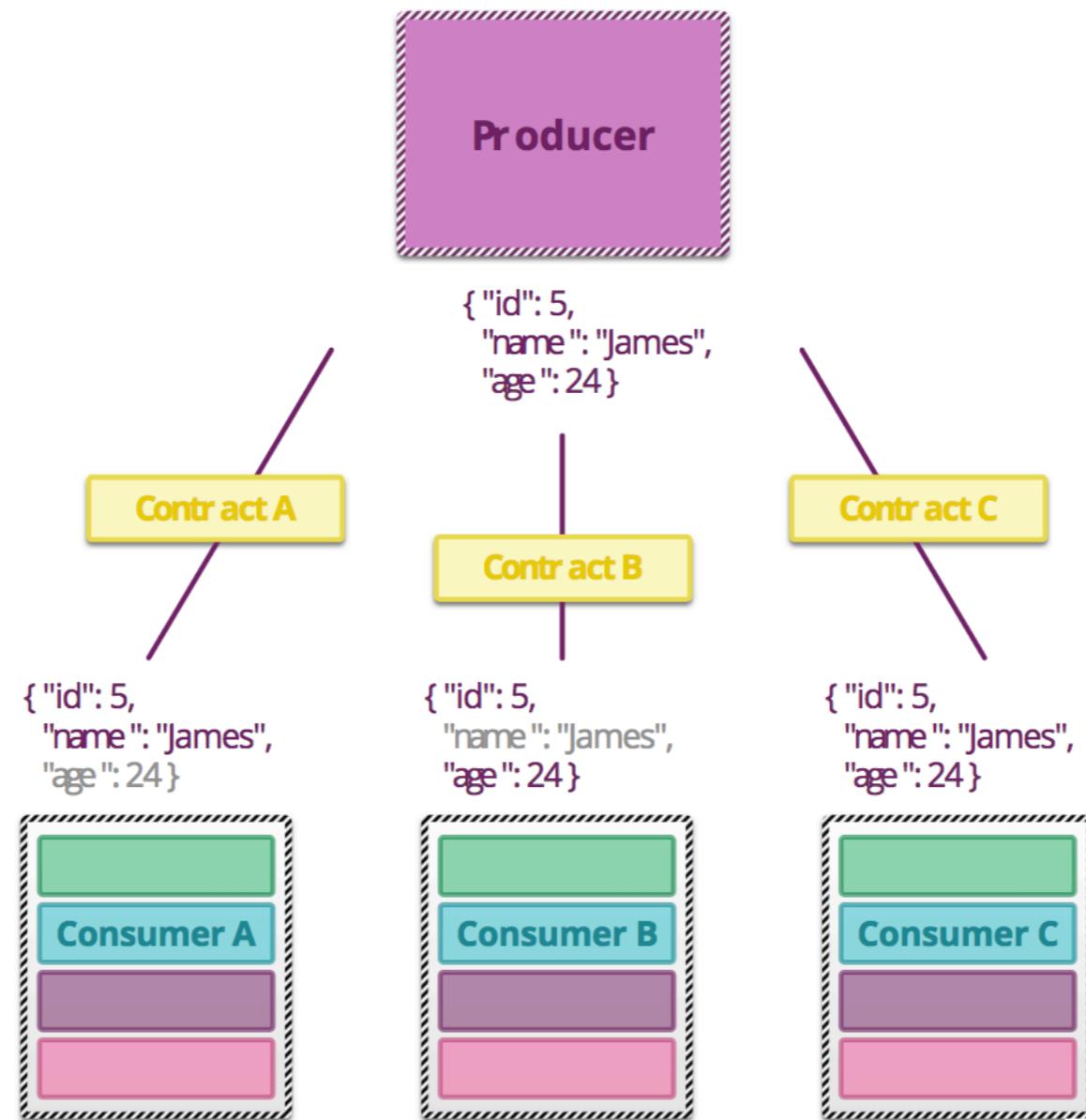
Integration testing



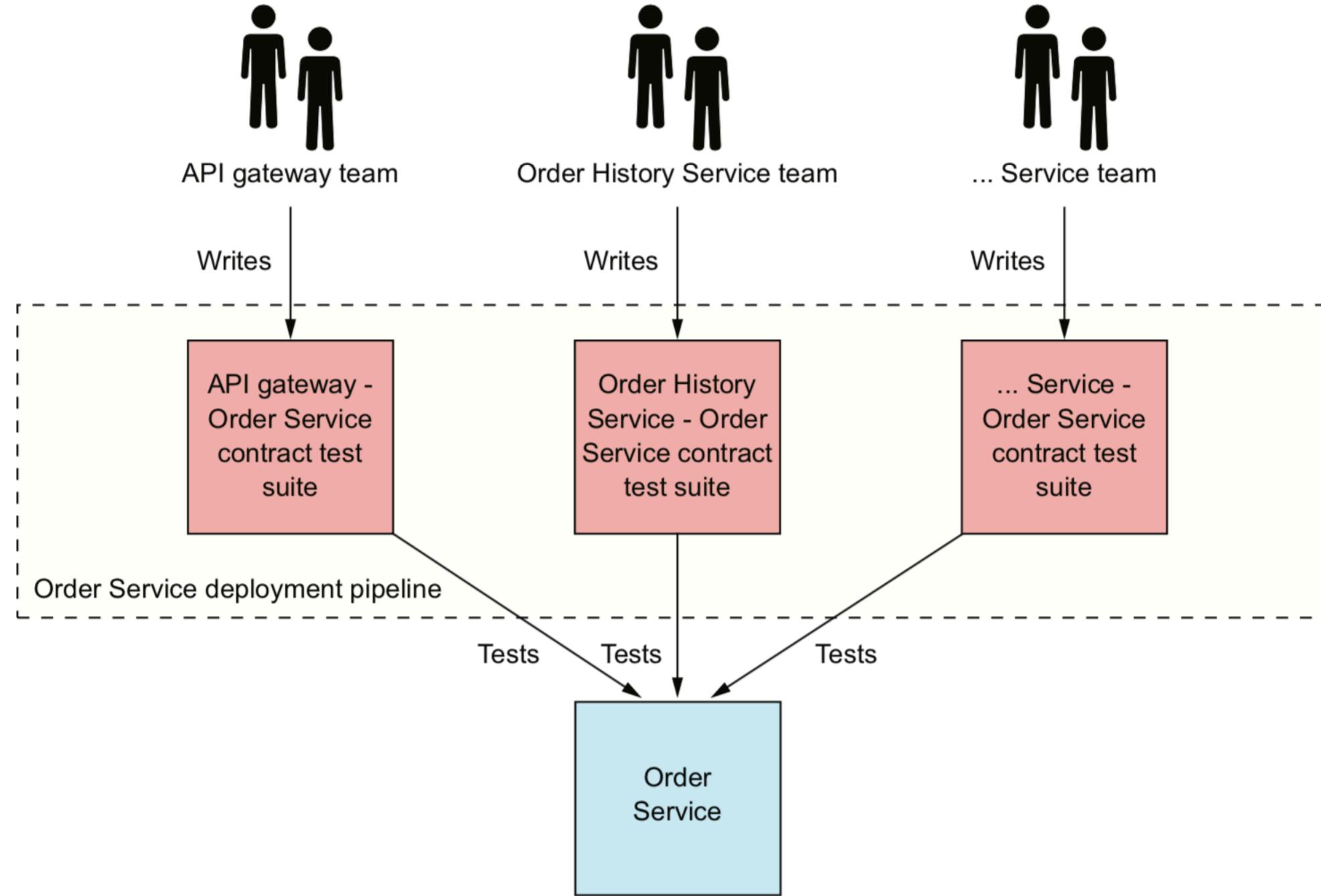
Component testing



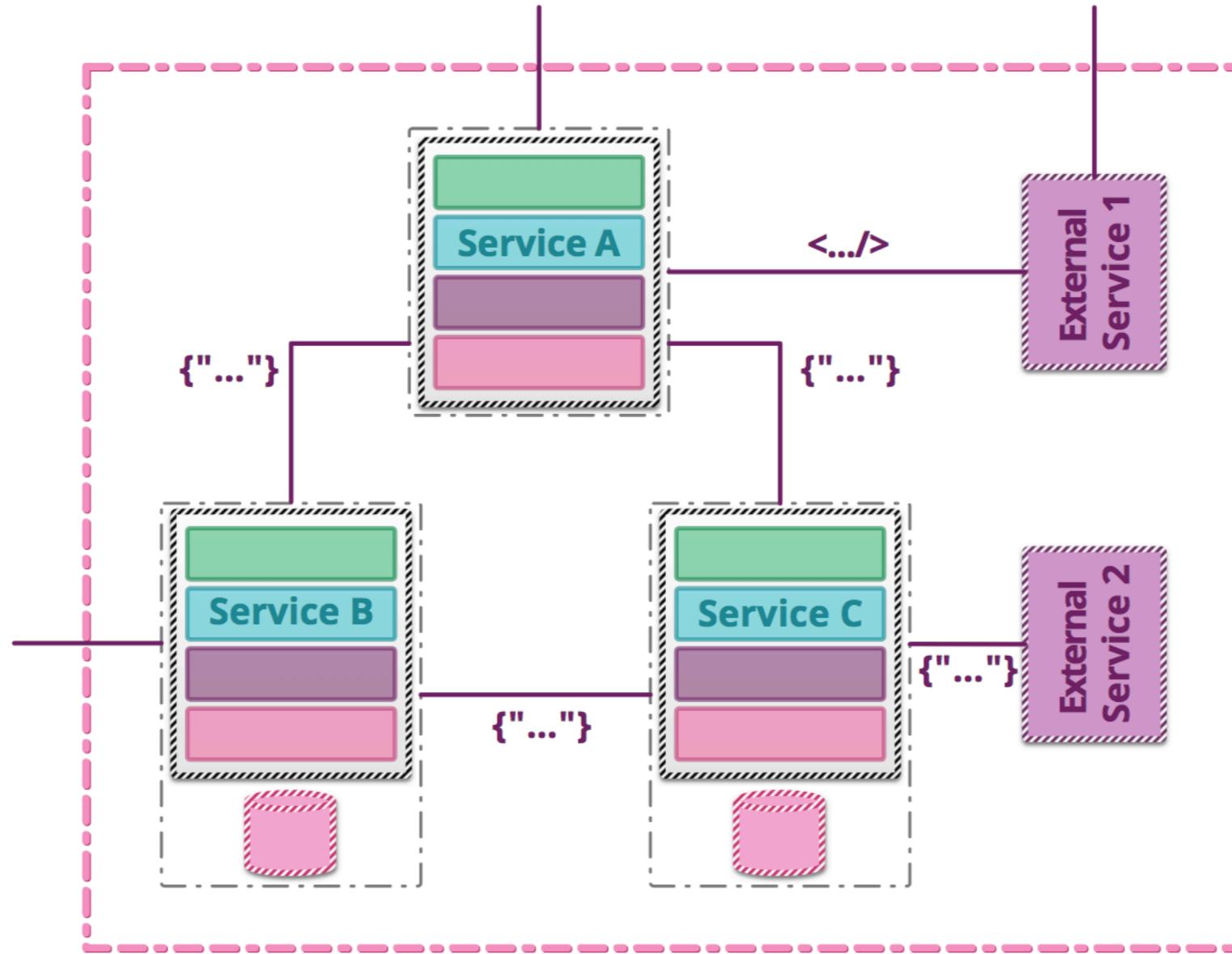
Contract testing



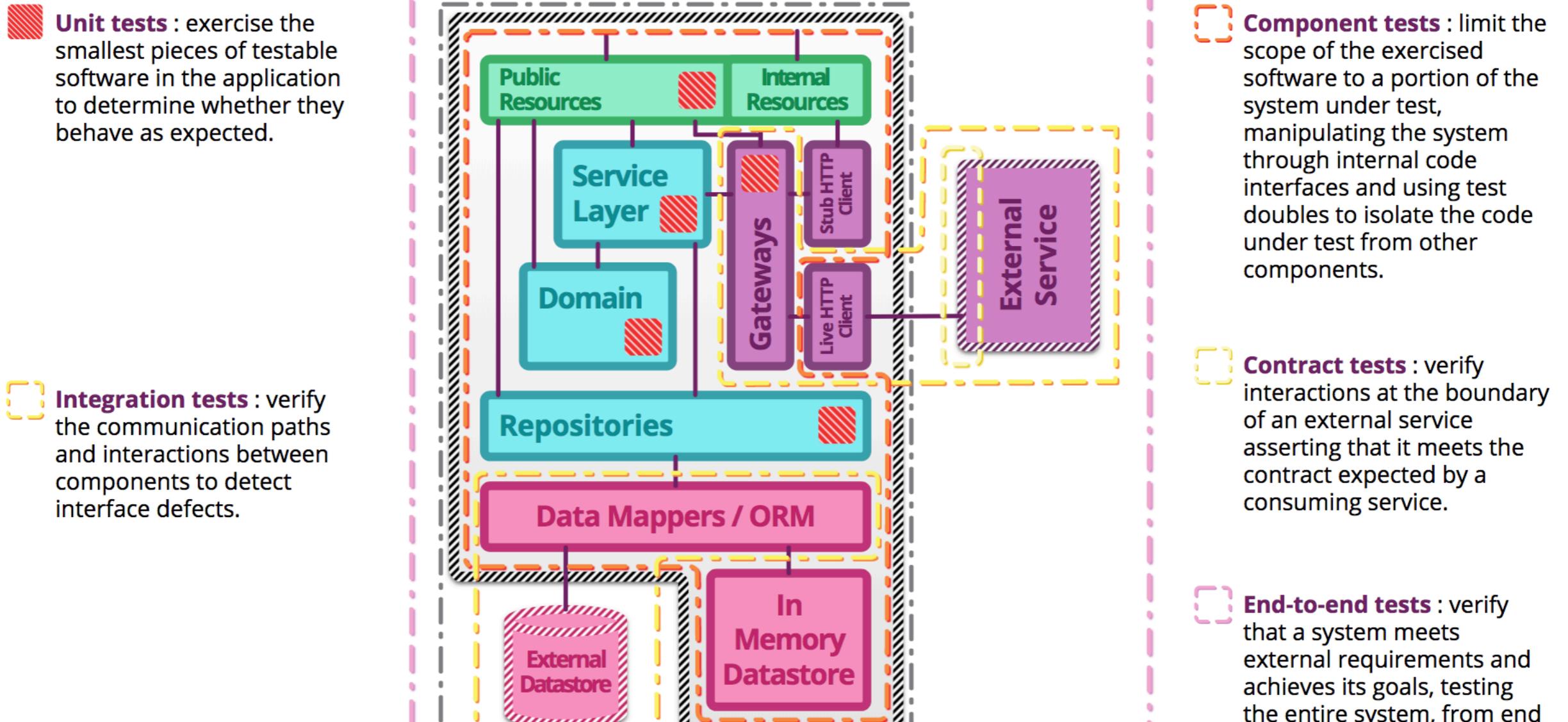
Consumer Contract testing



End-to-End testing



Summary



What is your testing strategy ?



Performance testing ?

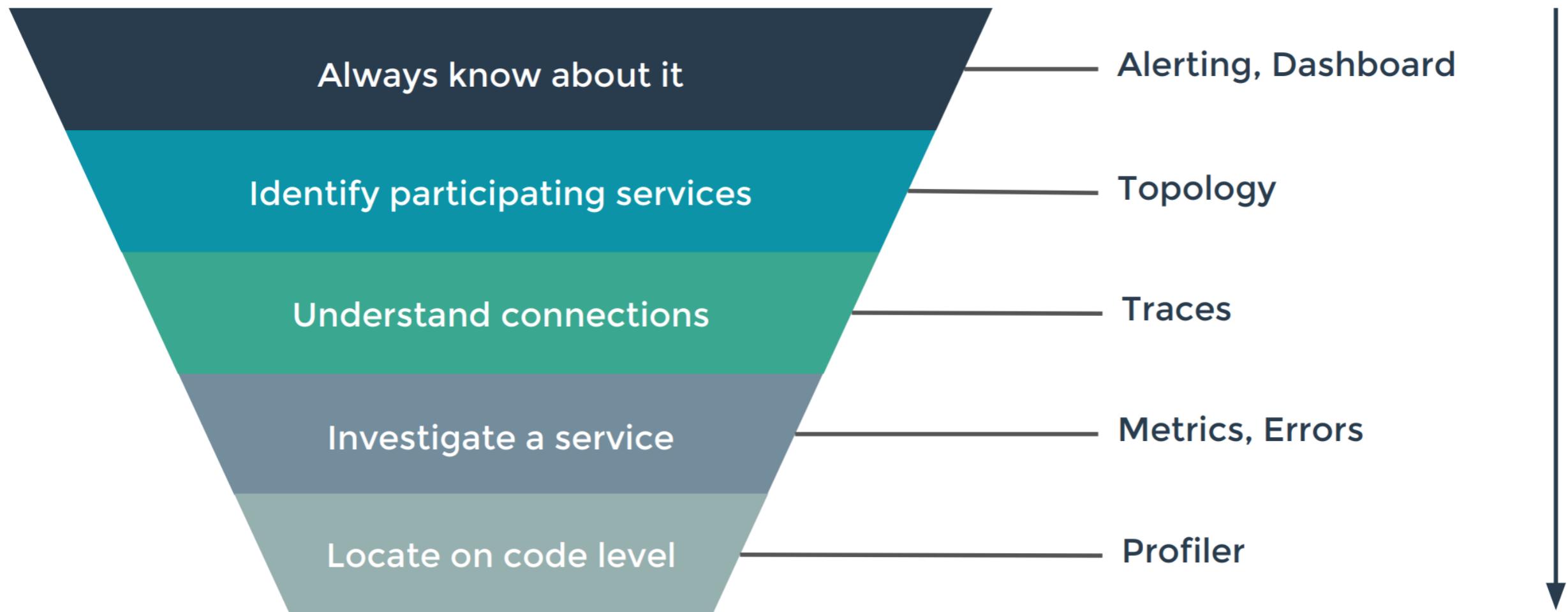
Security testing ?



How to find an issue ?



How to find an issue ?



Develop production-ready services



Important quality attributes

Security
Configurability
Observability



Secure services



Develop secure services

Authentication

Authorization

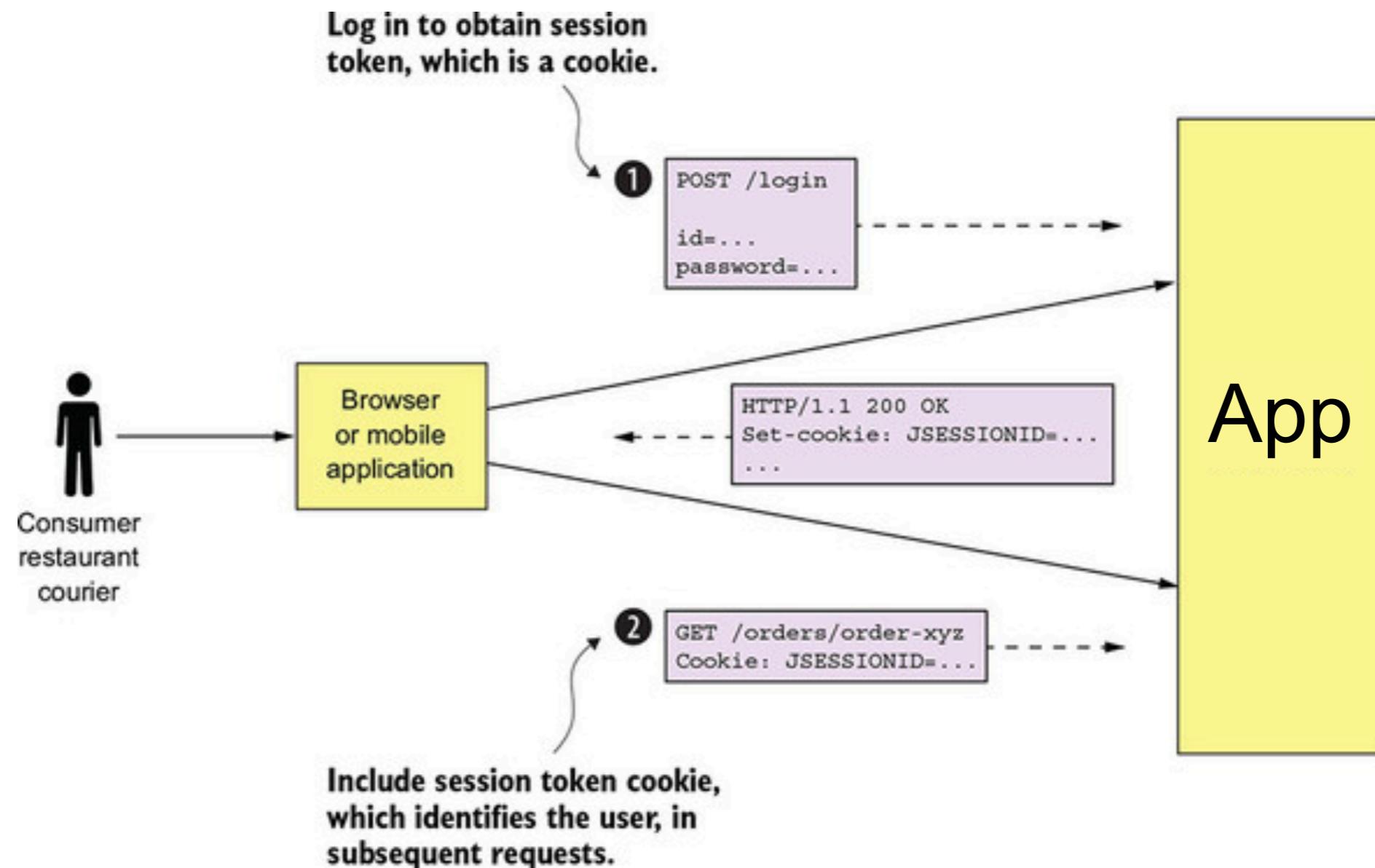
Auditing

Secure interprocess communication



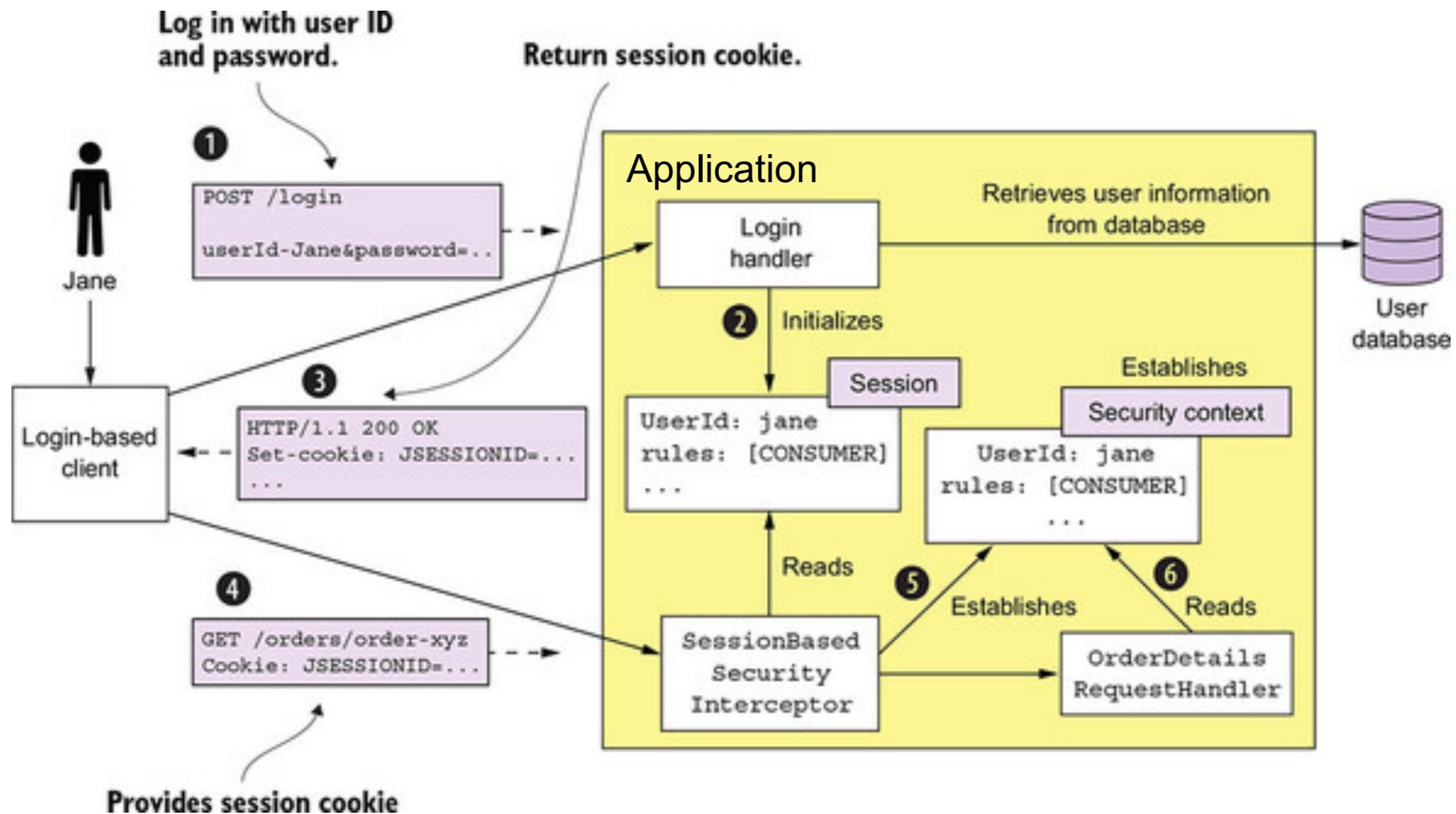
Security in traditional application

Keep security information in browser's cookie



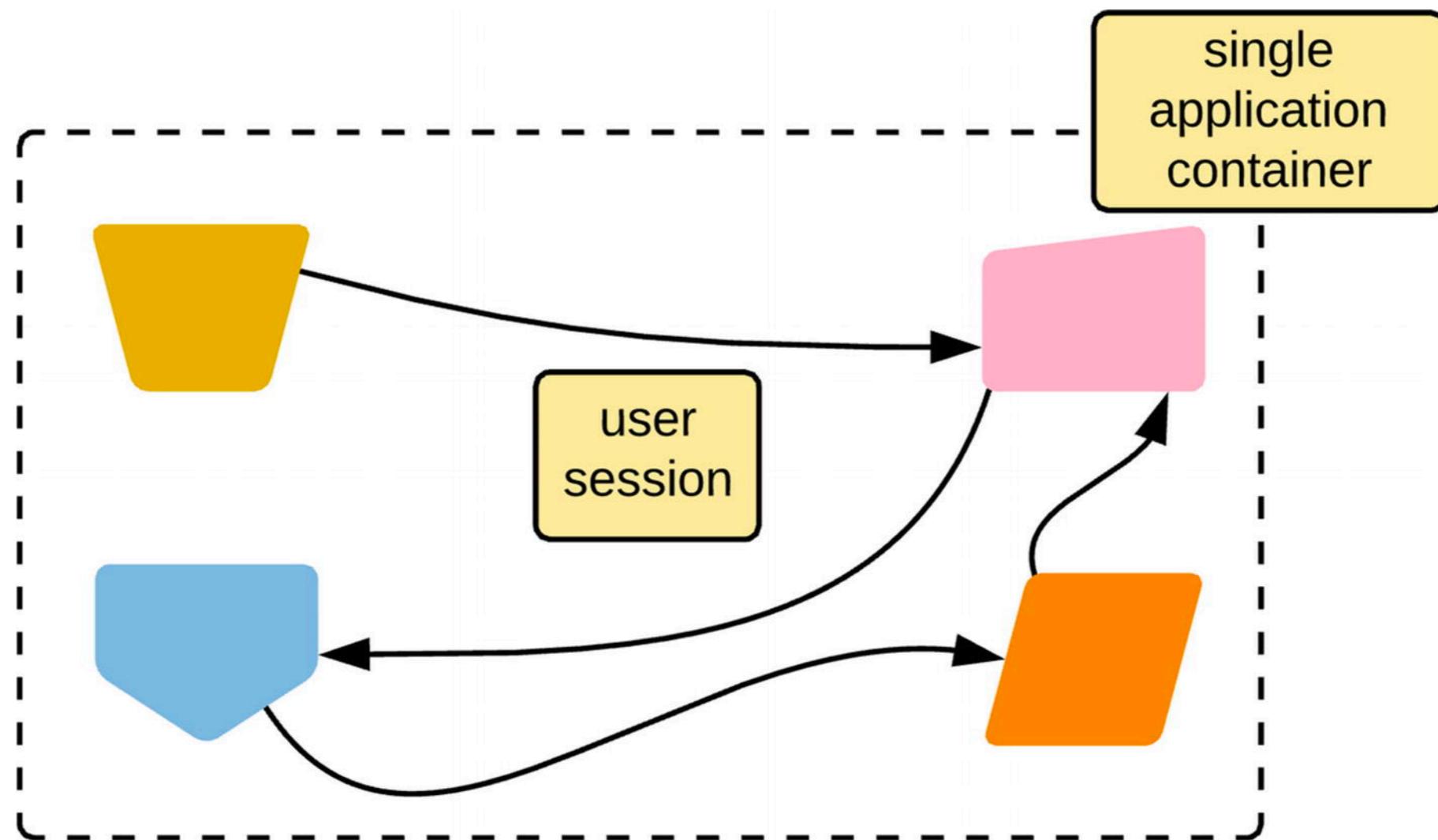
Security in traditional application

Implement security process in application



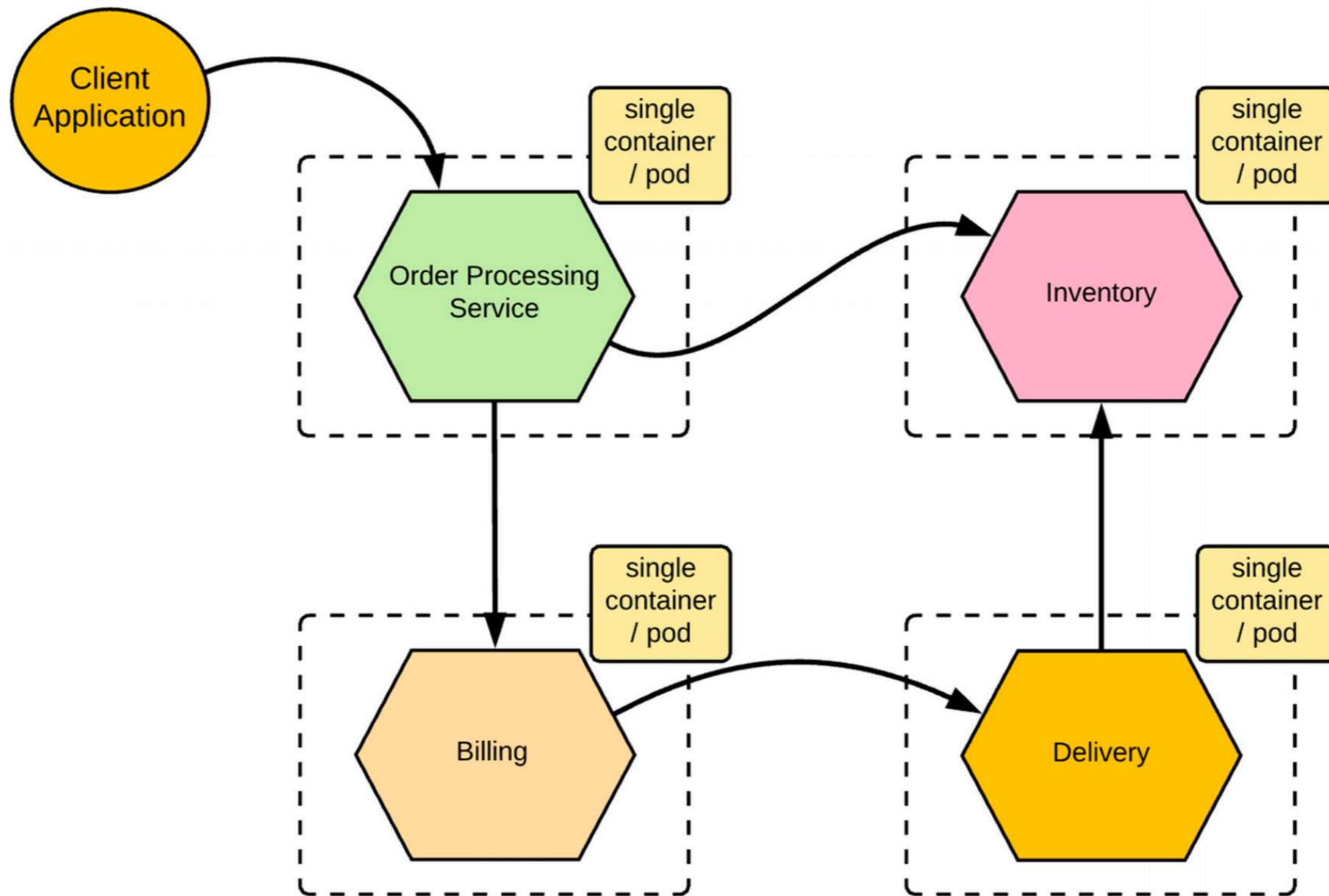
Security in traditional application

Sharing a user session in application



Security in multiple services ?

Interaction between multiple services



Secure service-to-service communication

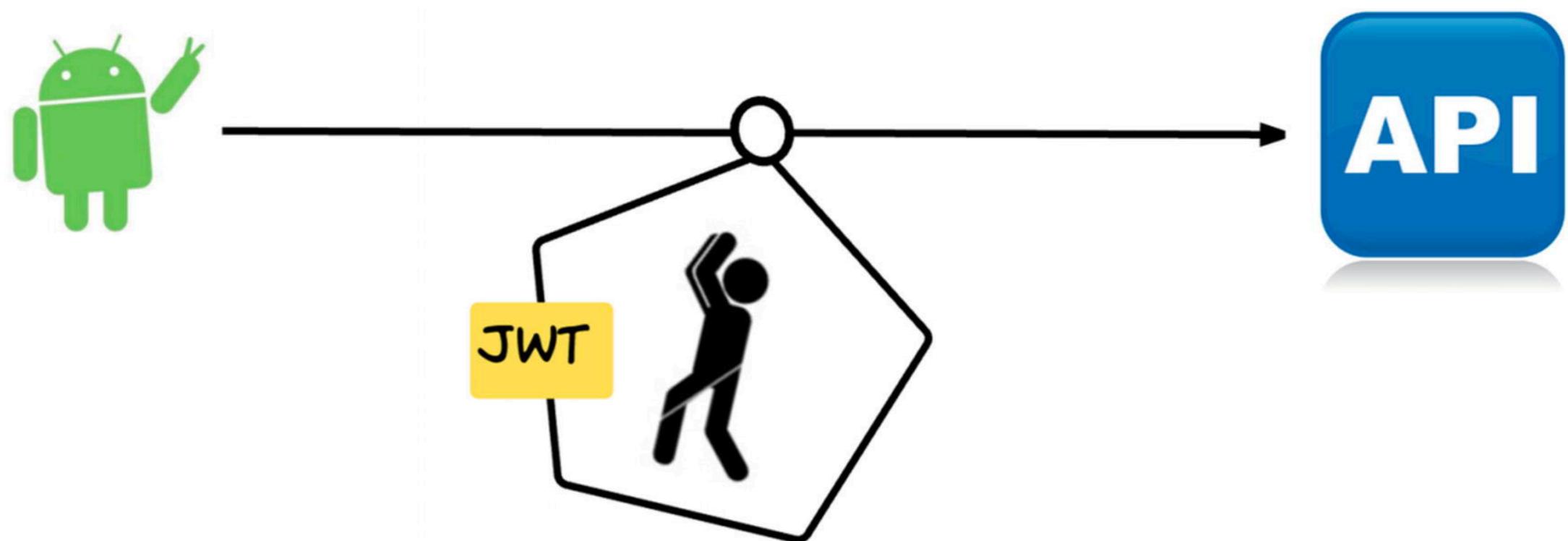
JSON Web Token (JWT)

Transport Layer Security (TLS) mutual authentication

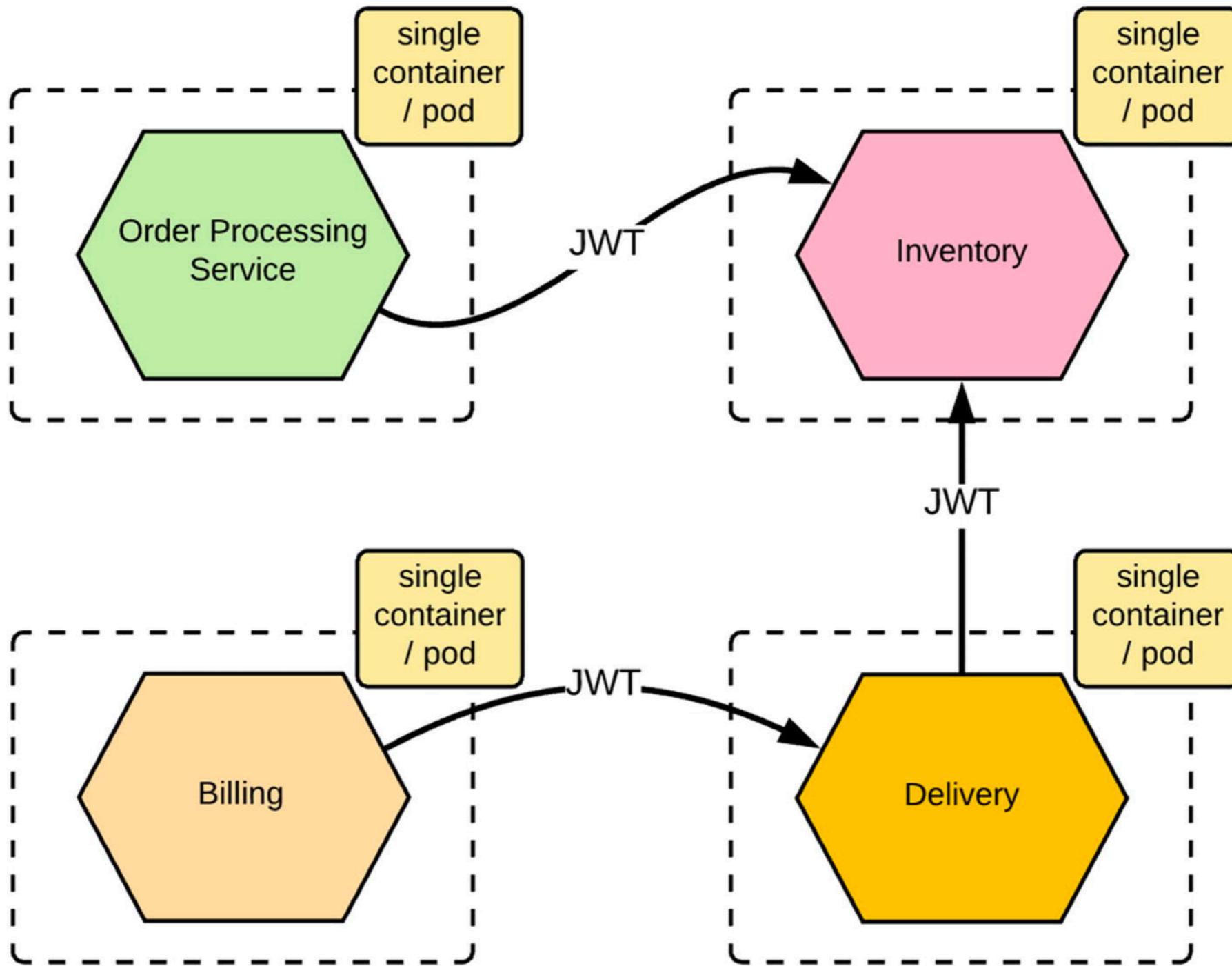


JSON Web Token (JWT)

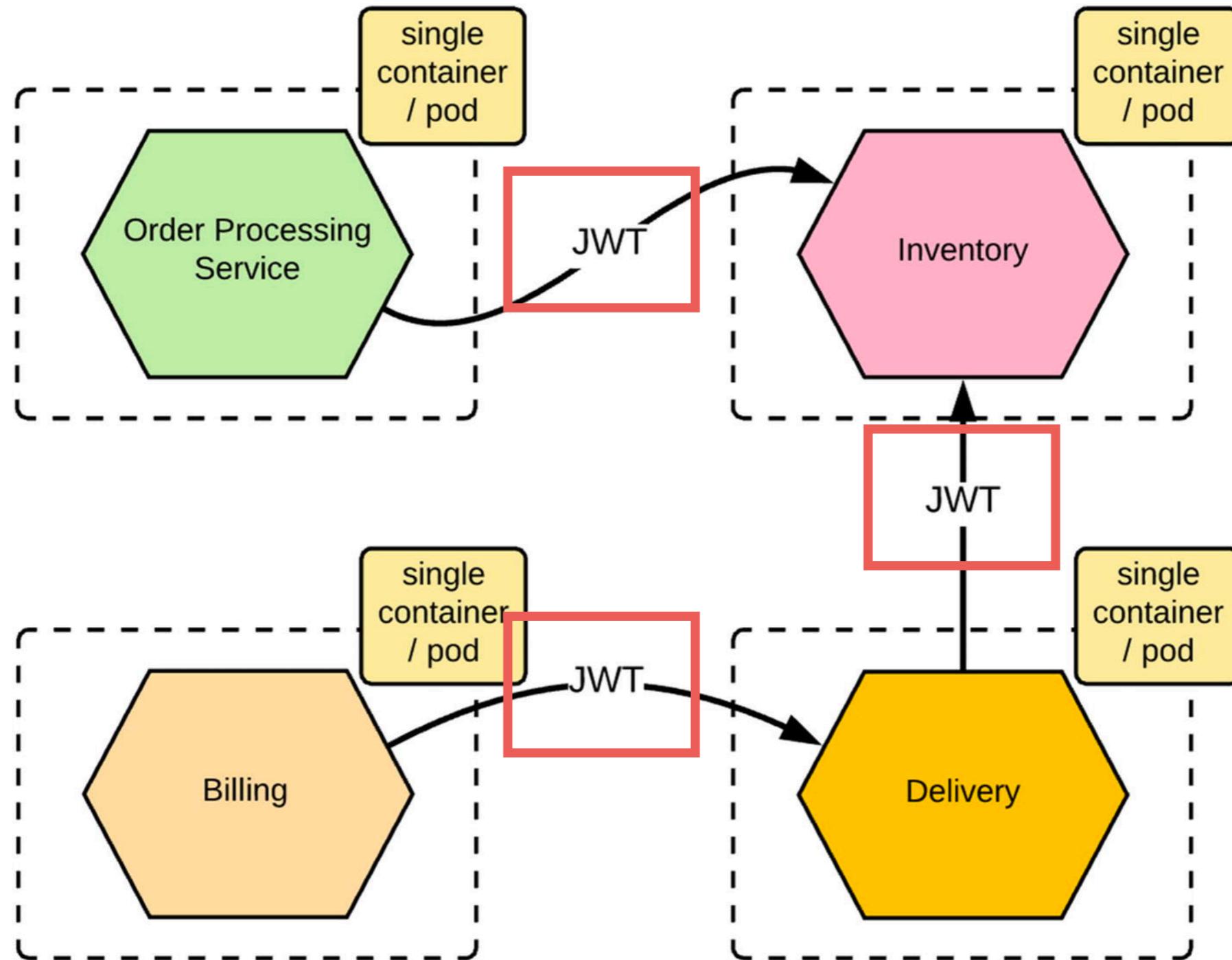
Define a container to transport data between interested parties



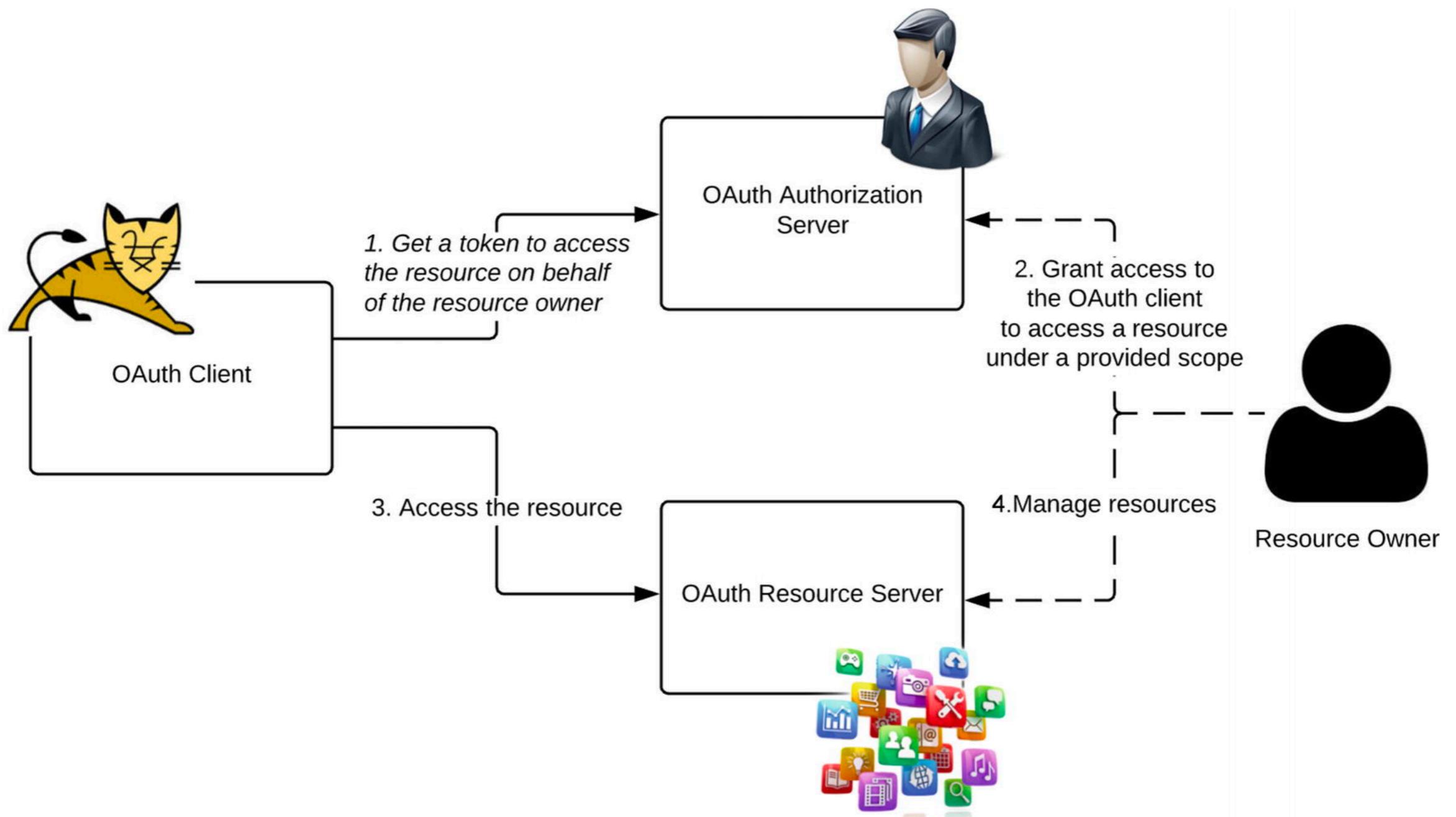
Passing user context as JWT



Problem ?

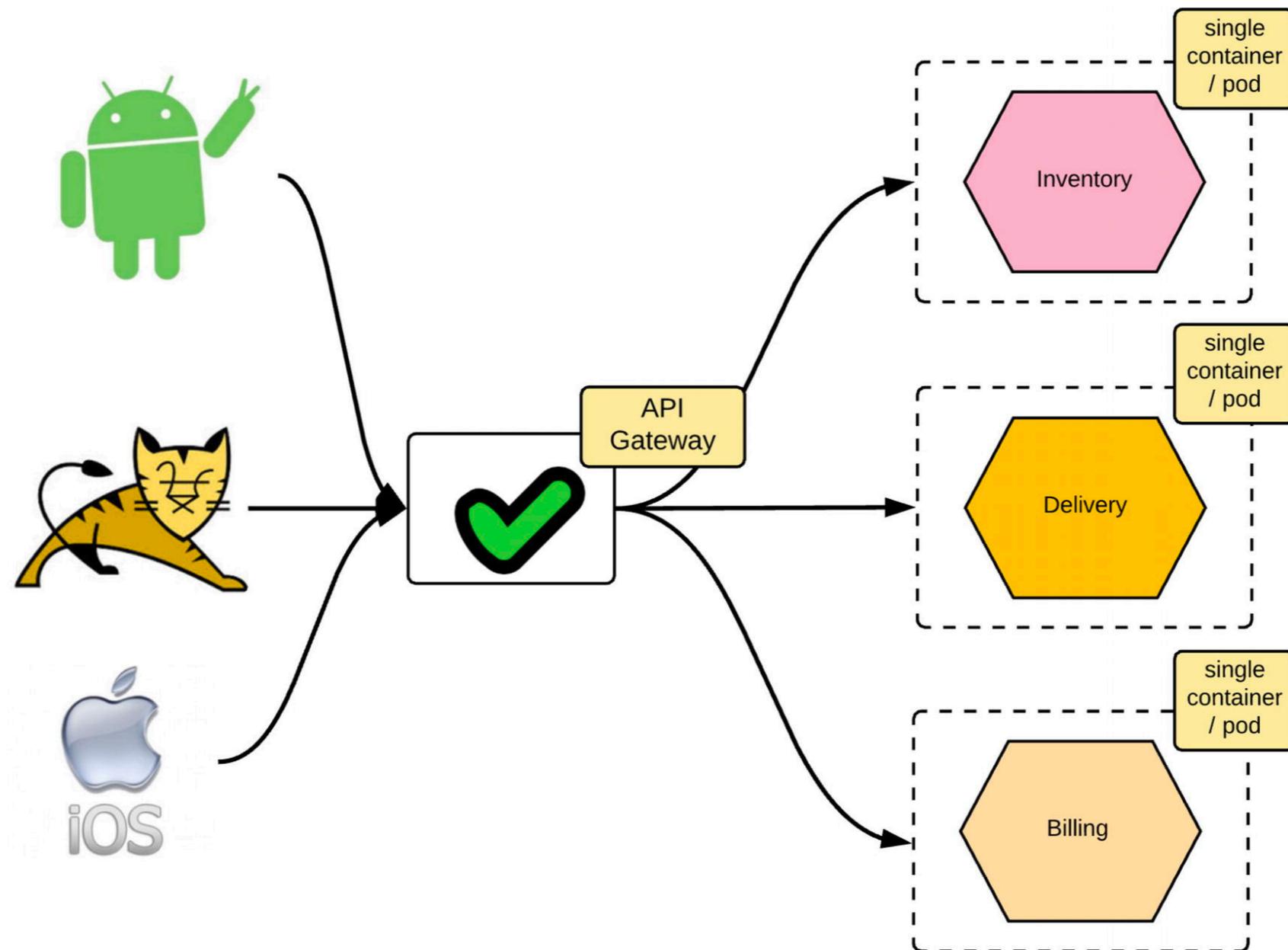


OAuth 2.0



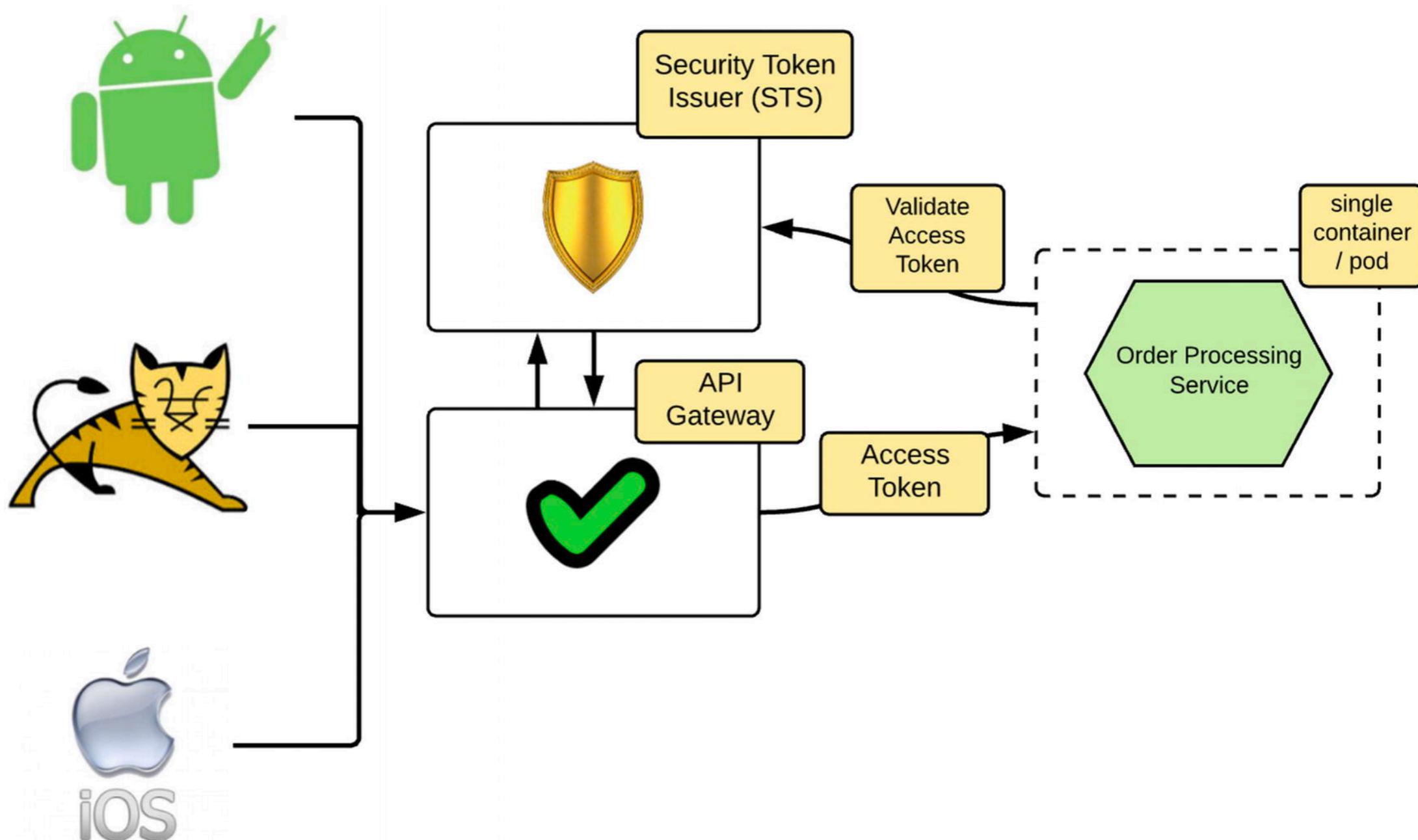
Centralize pattern

Using API gateway pattern

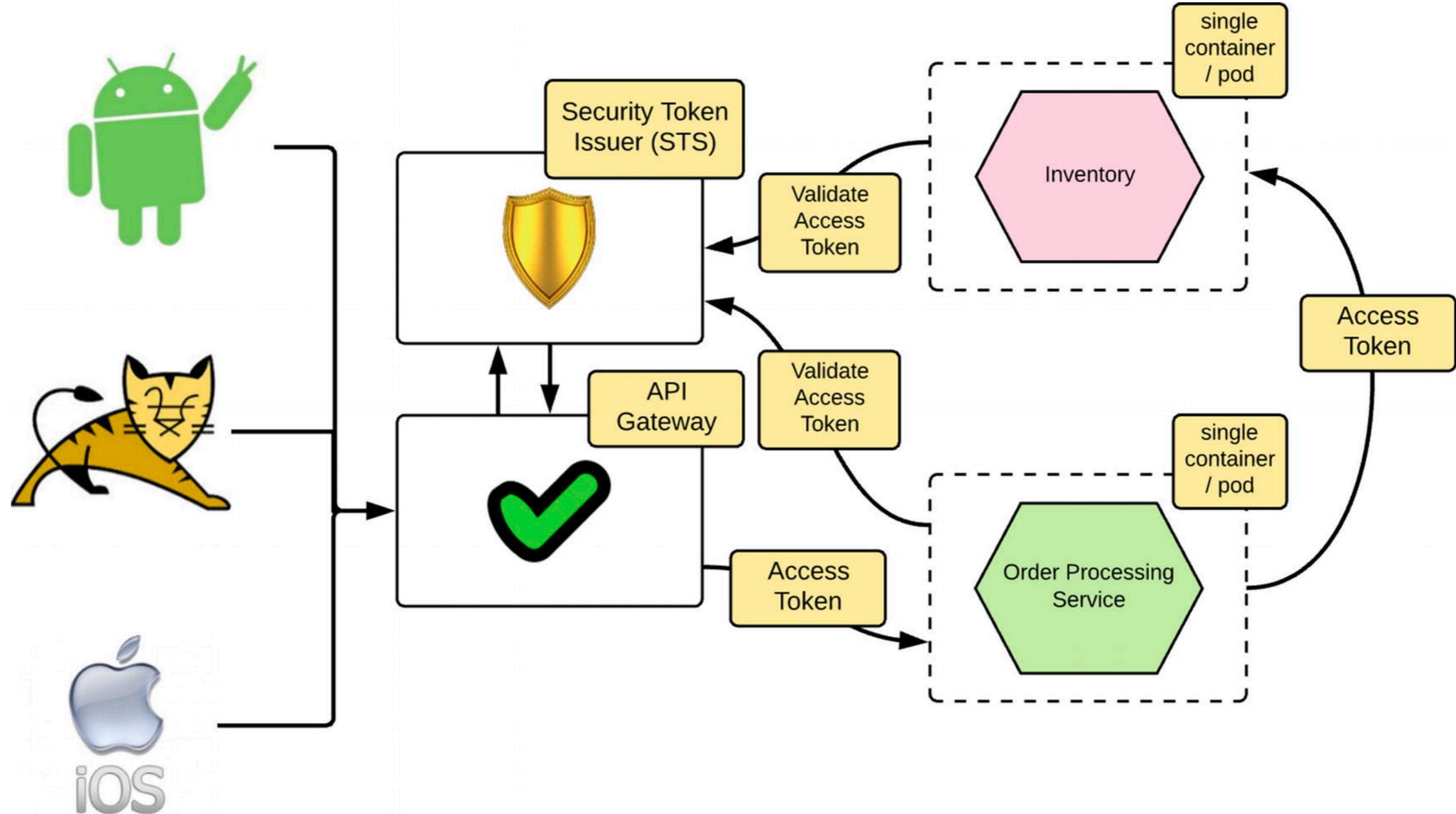


Centralize pattern

Using API gateway pattern

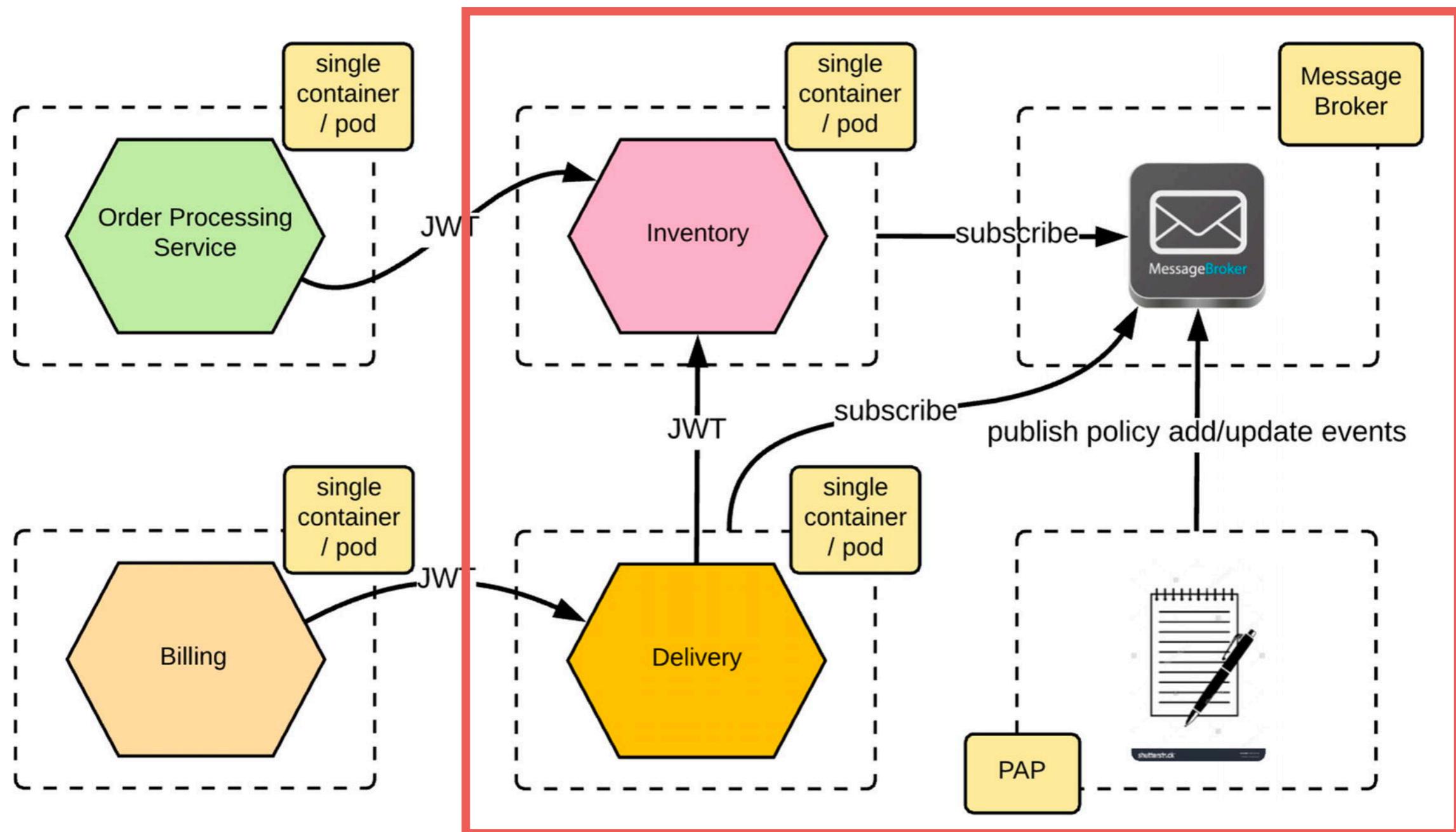


API gateway with OAuth 2.0



Access control of services

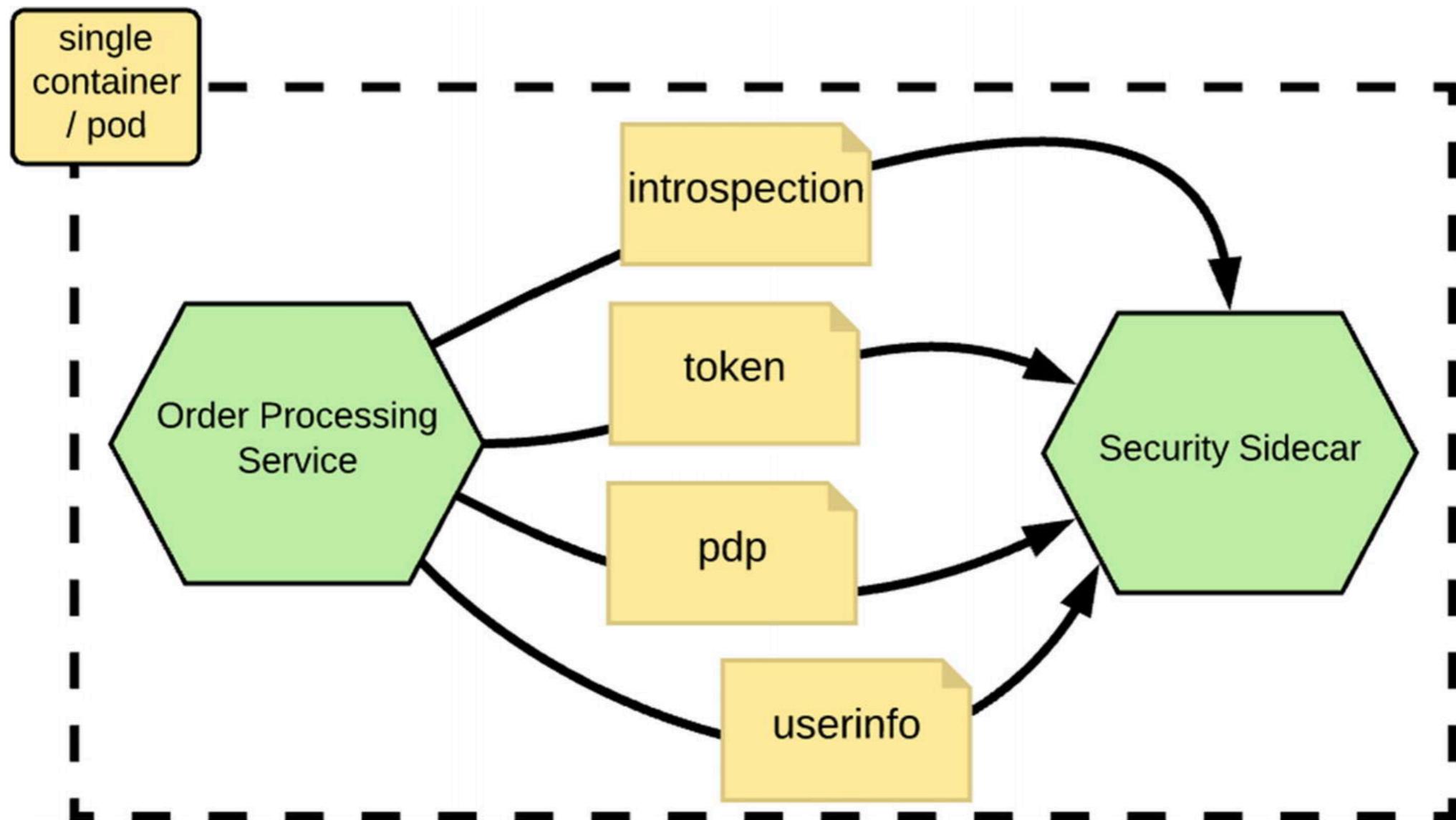
Policy Administration Point



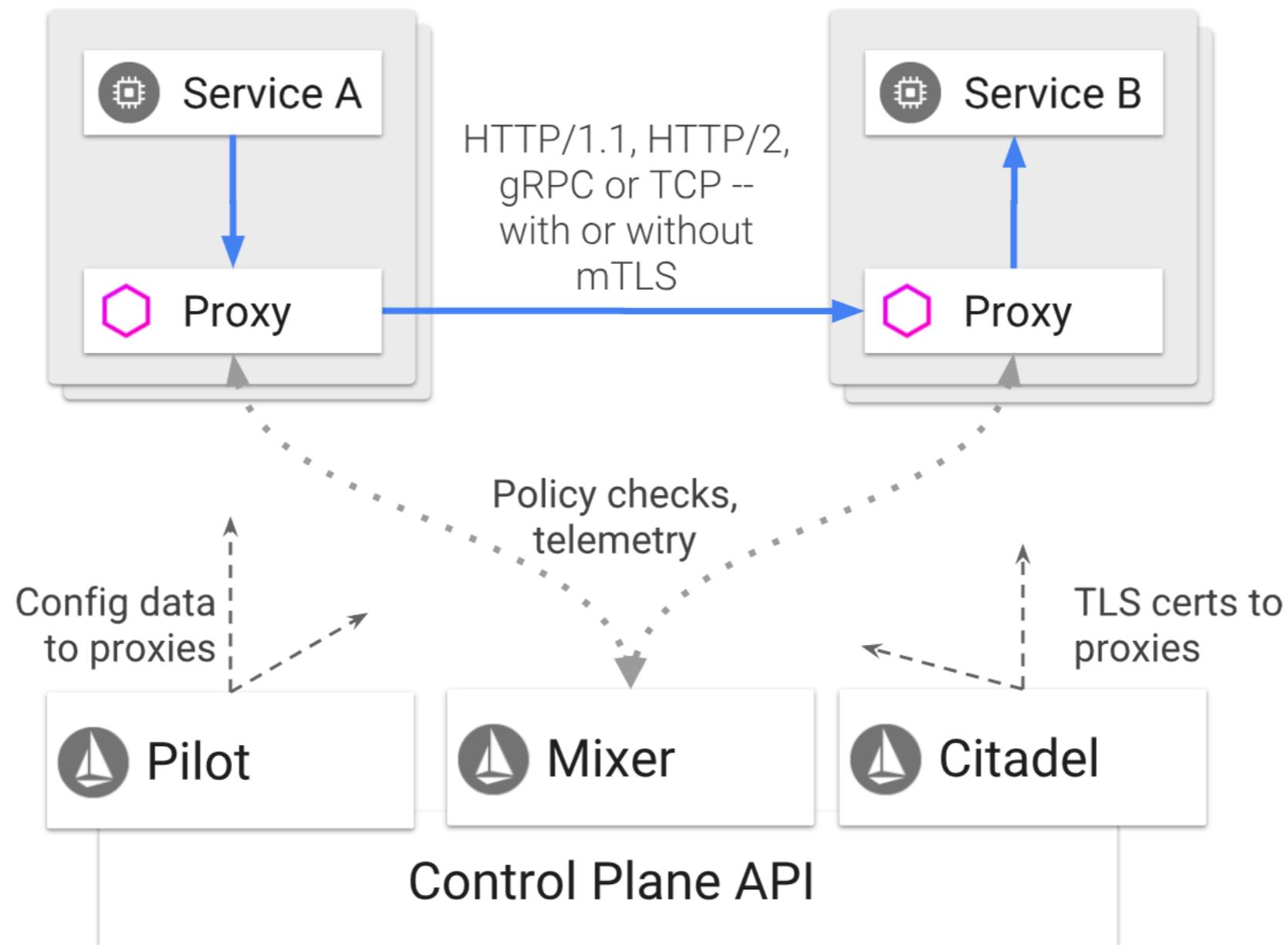
Working sidecar



Security sidecar



Security sidecar



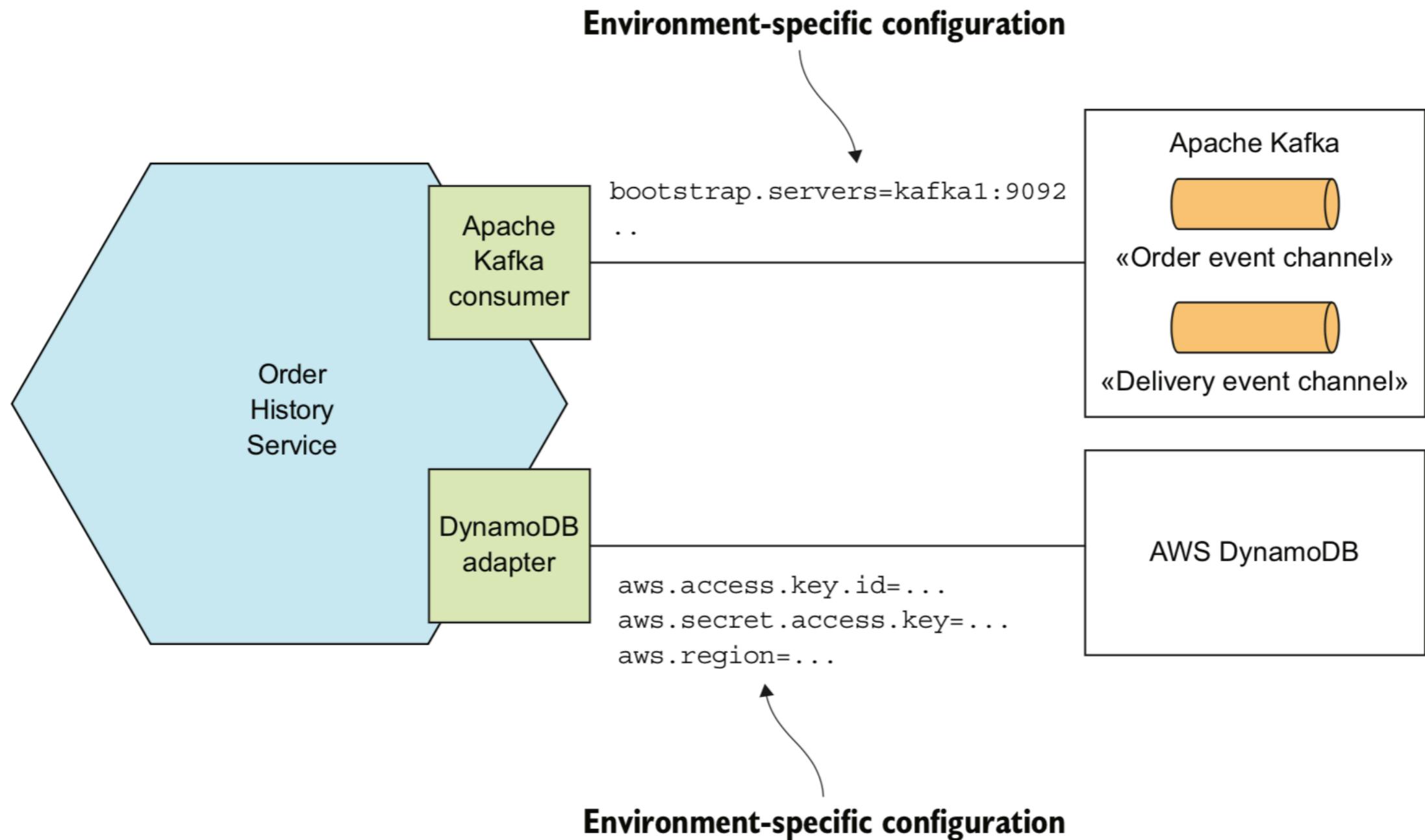
<https://istio.io/docs/concepts/what-is-istio/>



Configurable services



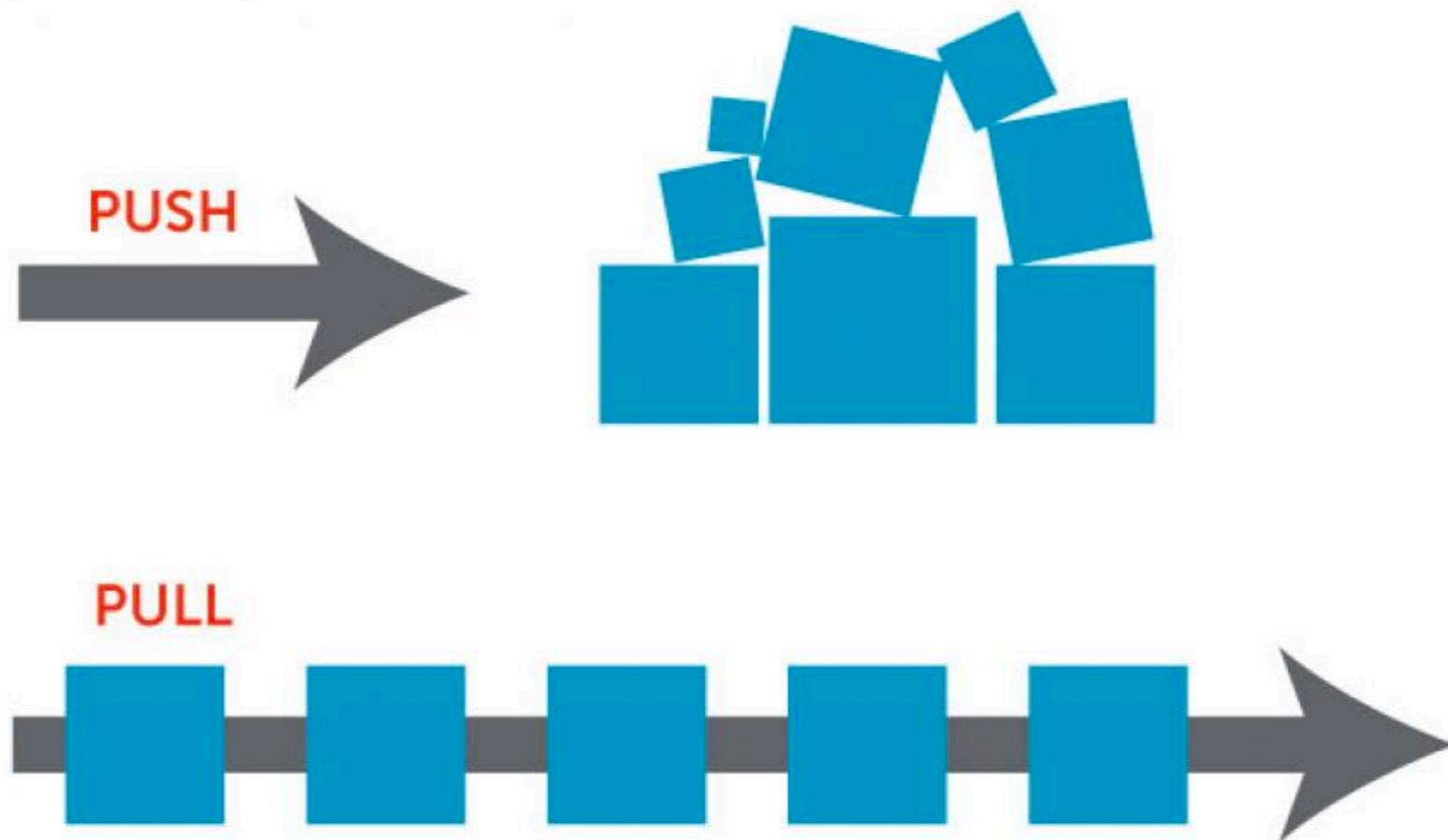
Design configurable services



External configuration models

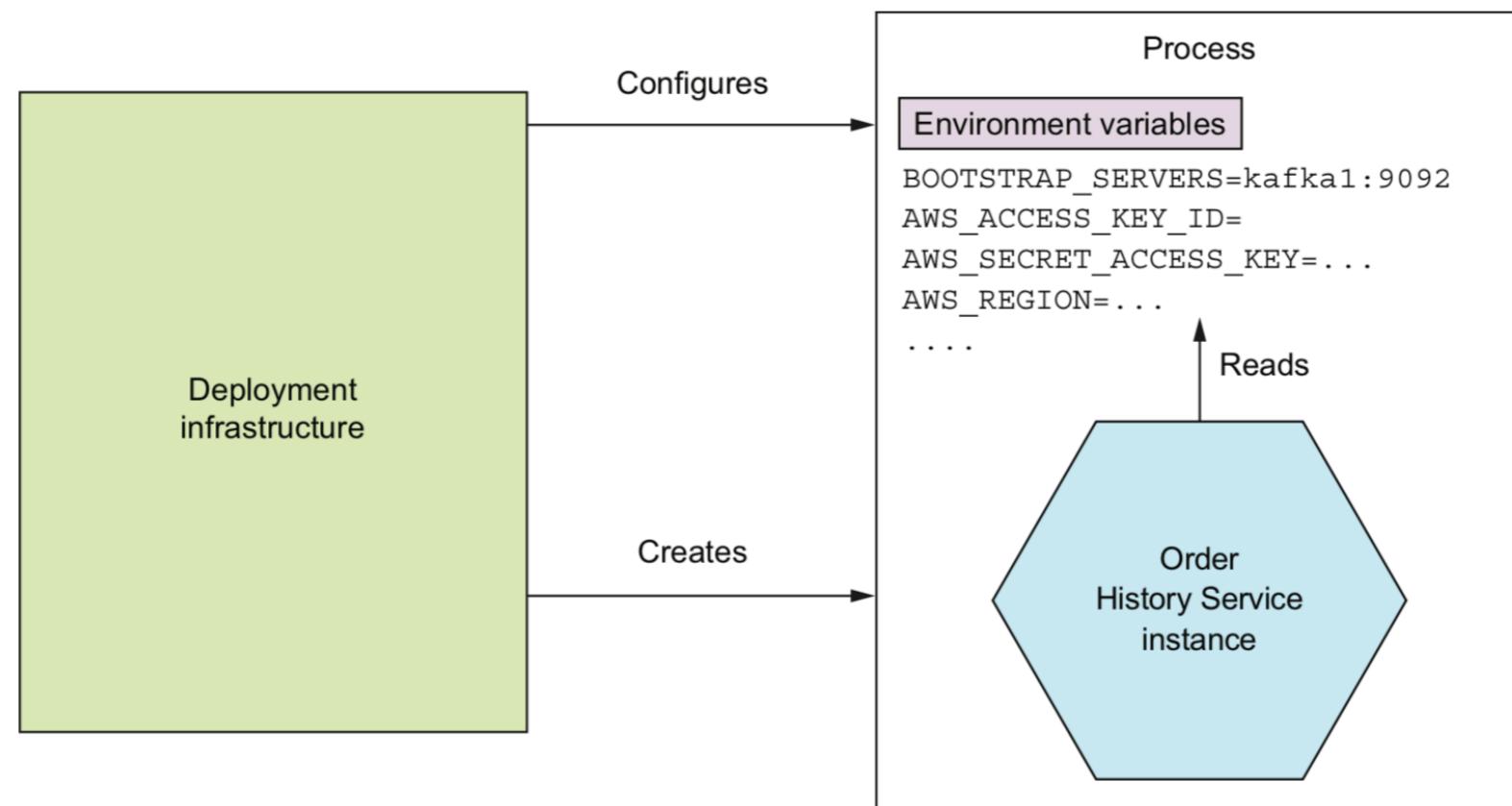
Push model

Pull model



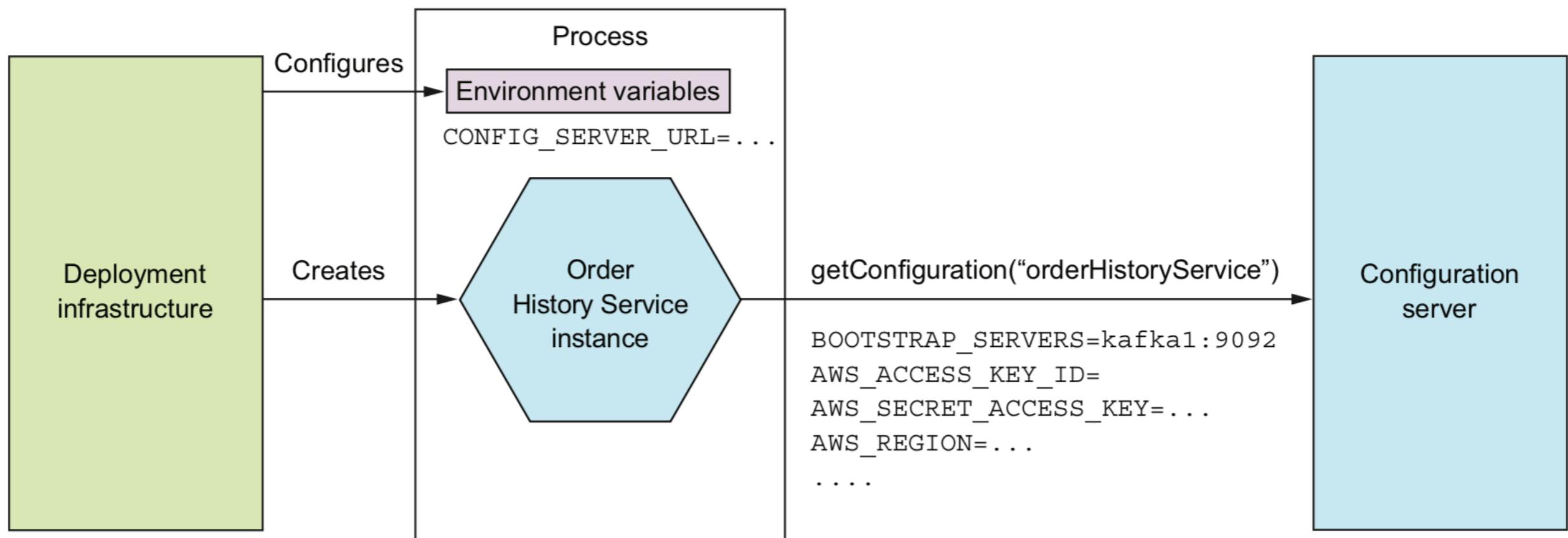
Push model

Pass the configuration to service
OS environment variables
Configuration files



Pull model

Service read configuration from configuration server



Benefits of configuration server

Centralized configuration

Transparent decryption of sensitive data

Dynamic reconfiguration



Drawbacks of configuration server

Need setup and maintain !!

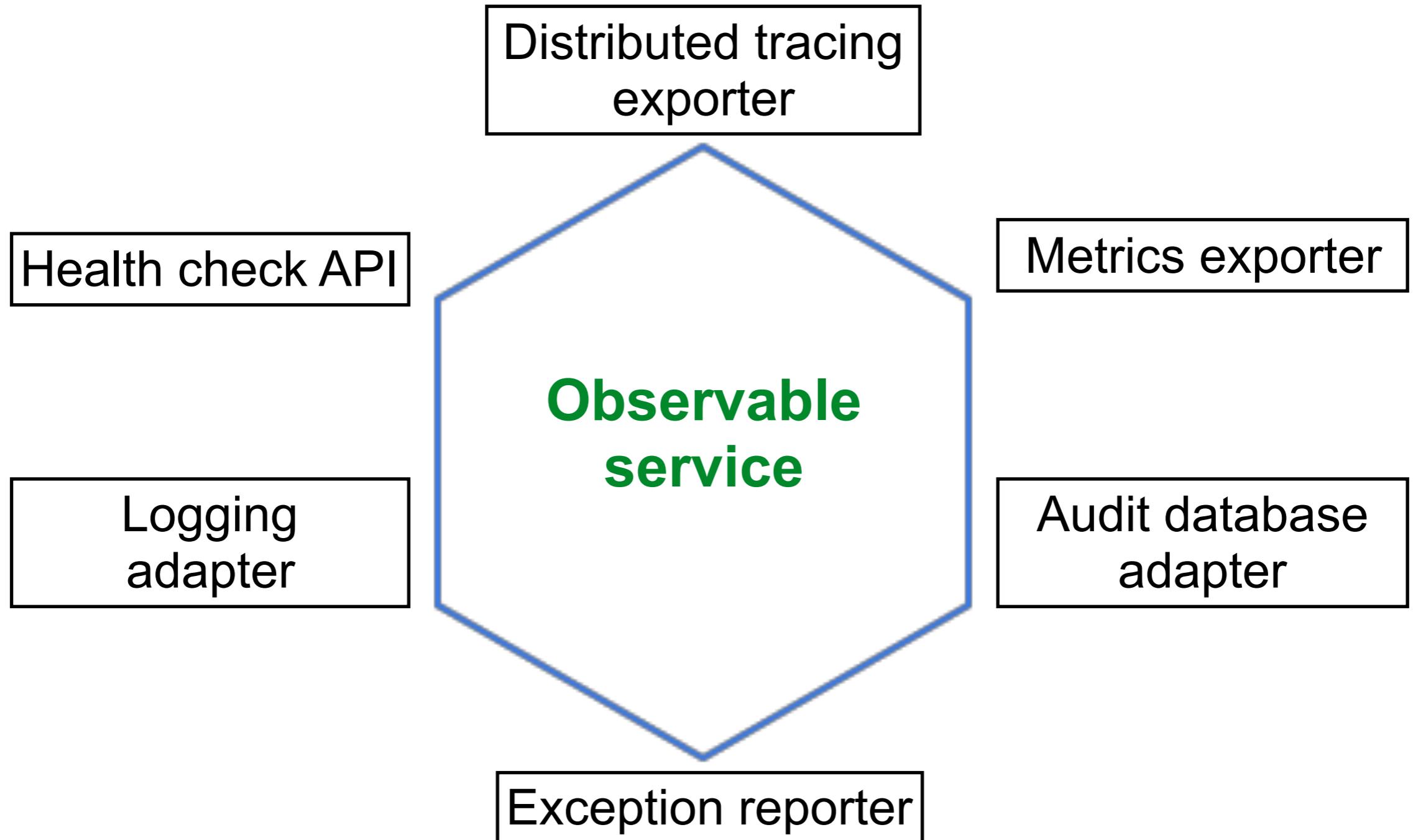


Design observable services

- Health check API
- Log aggregation
- Distributed tracing
- Exception tracking
- Application metrics
- Audit logging

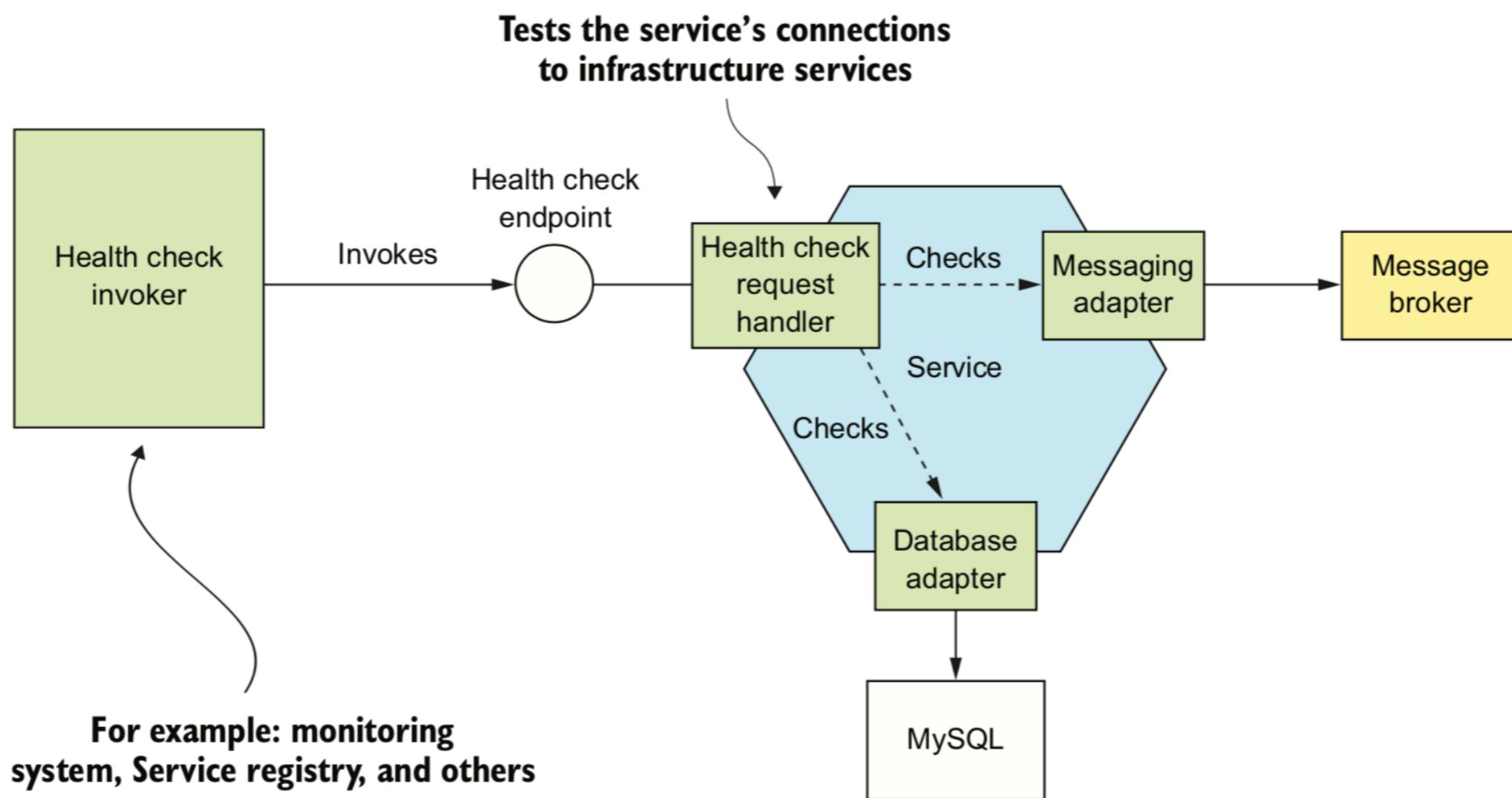


Observable services



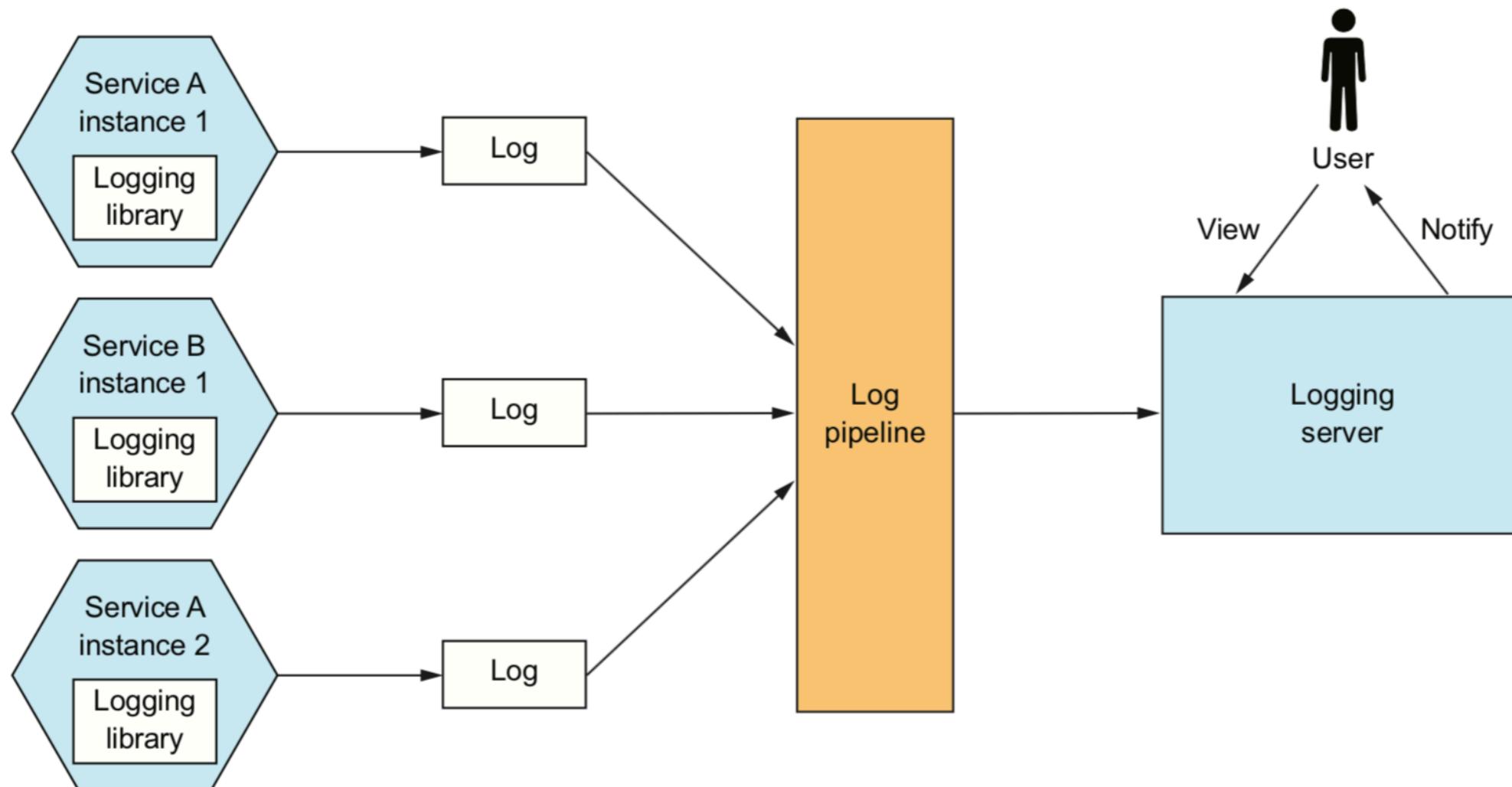
Health check API

Expose an endpoint that return the health of service



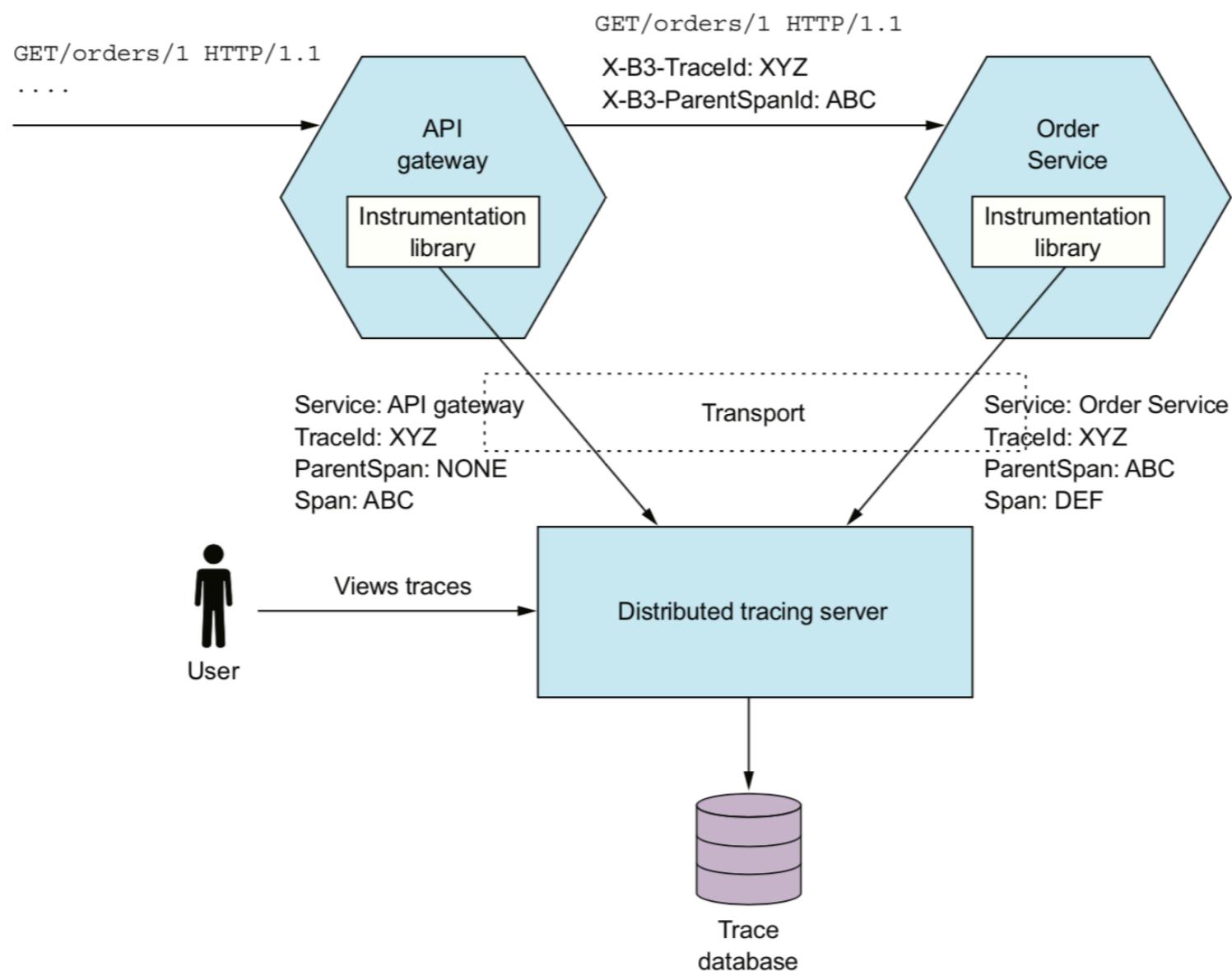
Log aggregation

Log service activity and write logs into a centralized logging server. (searching, alerting)

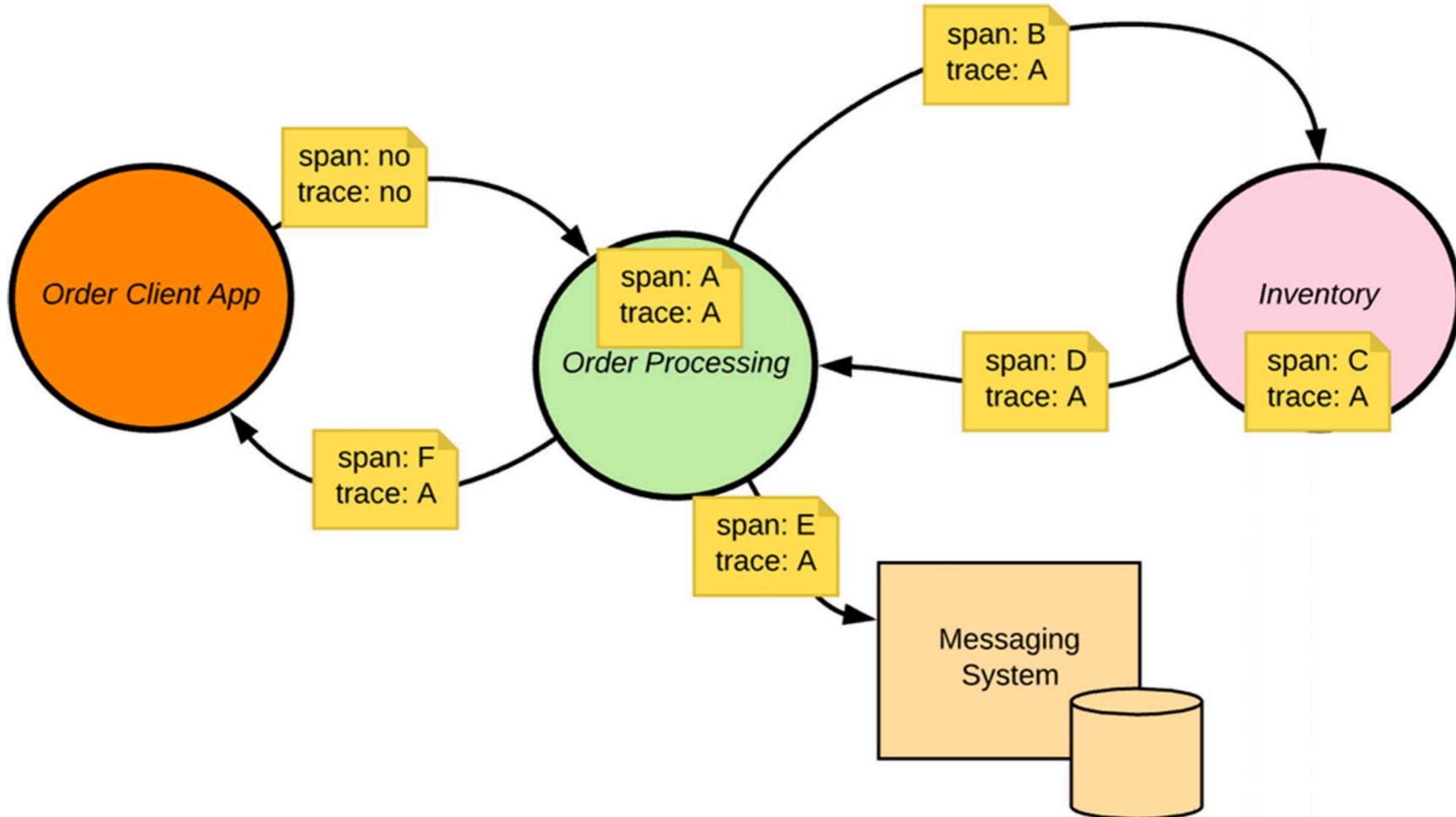


Distributed tracing

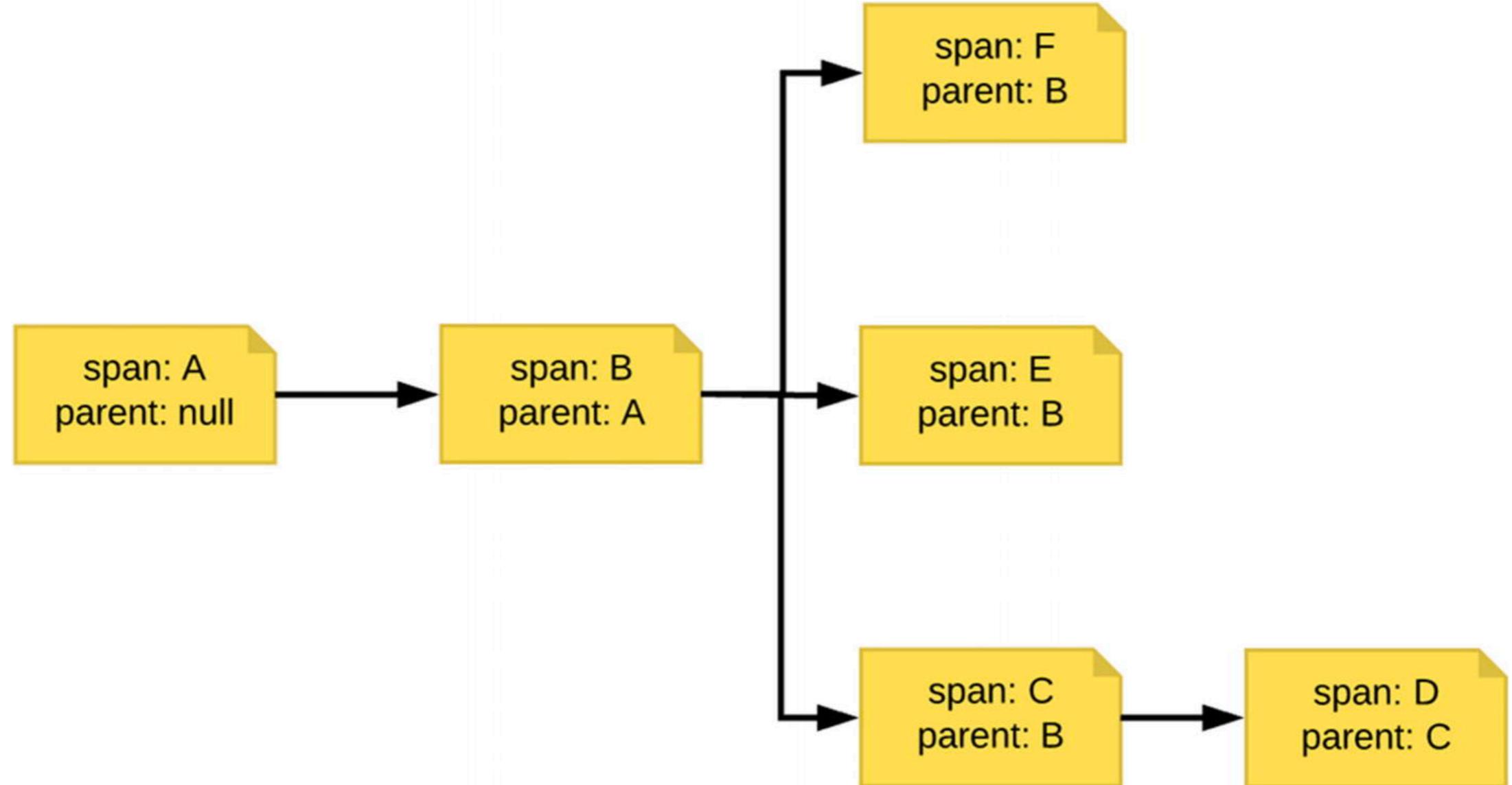
Assign each external request a unique ID and trace requests as flow between services



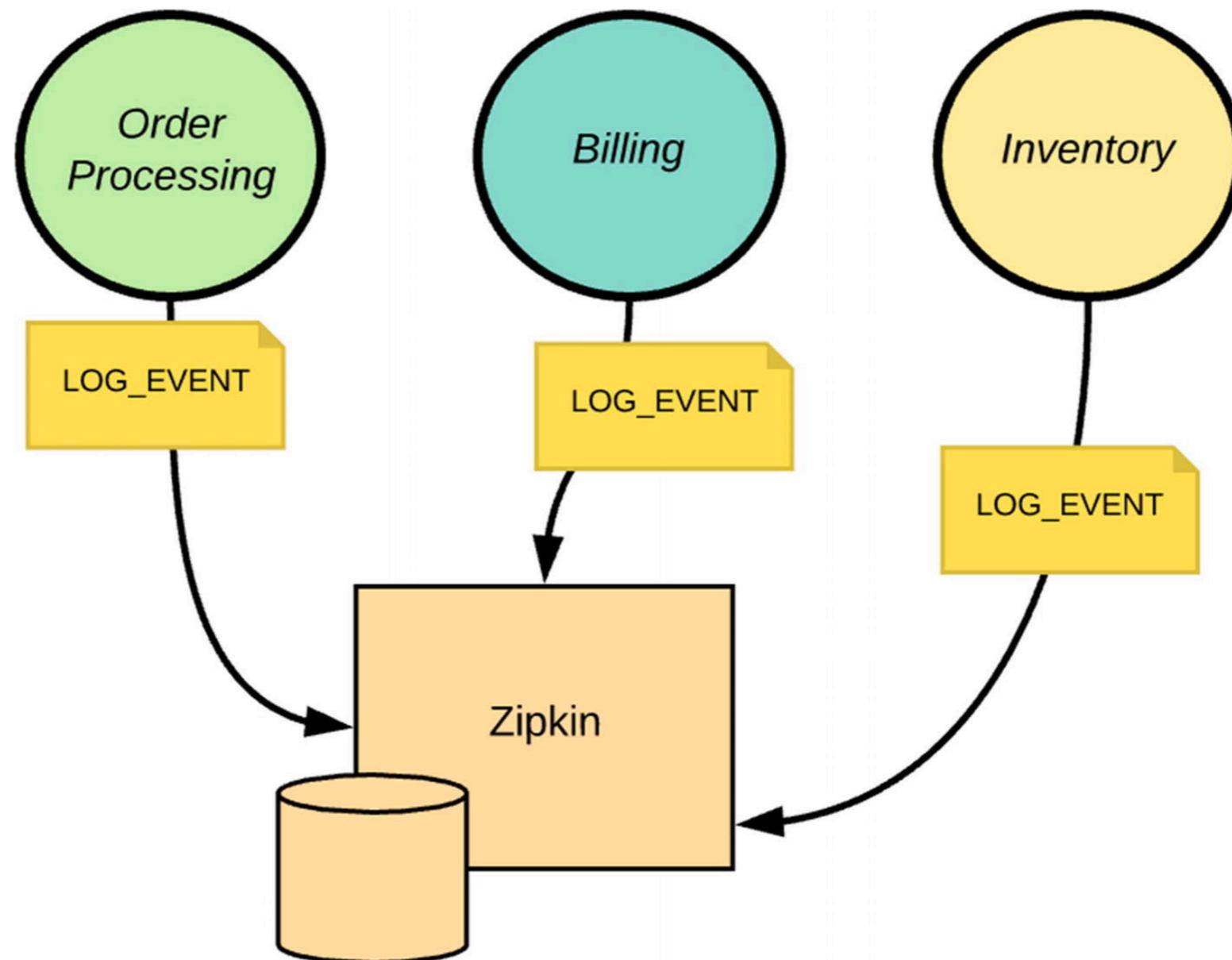
Distributed tracing



Distributed tracing



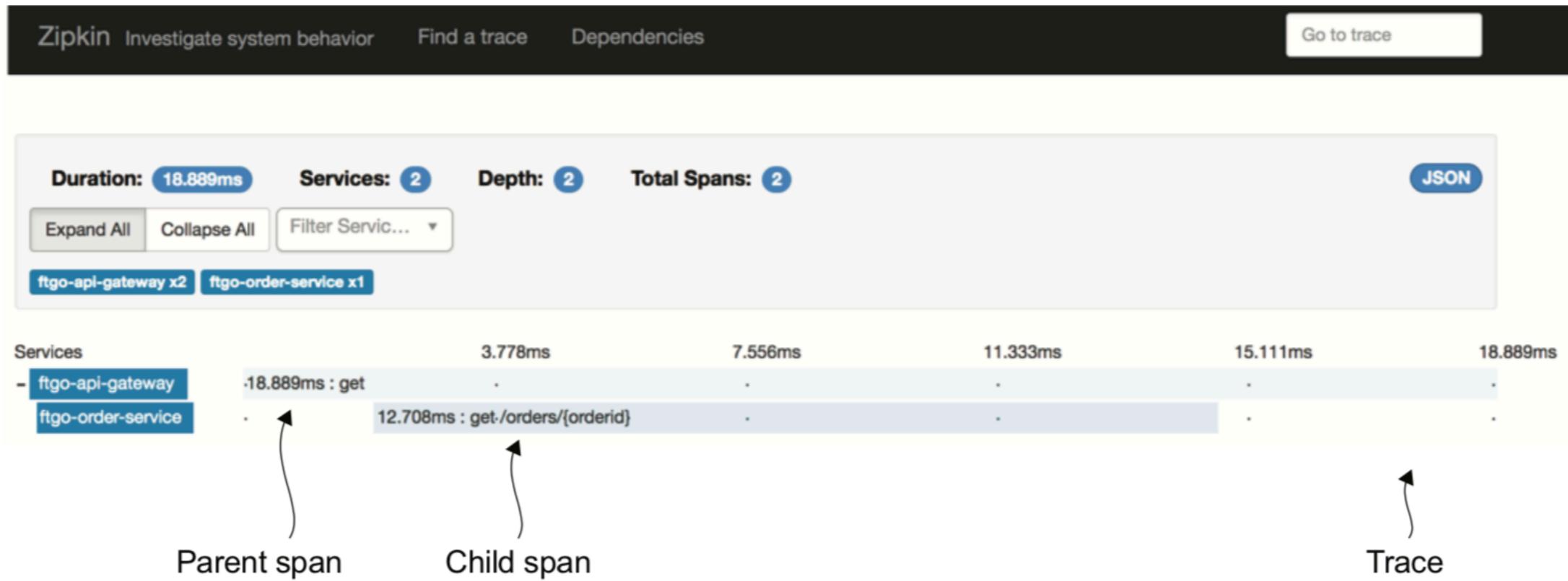
Distributed tracing with Zipkin



<https://zipkin.io/>



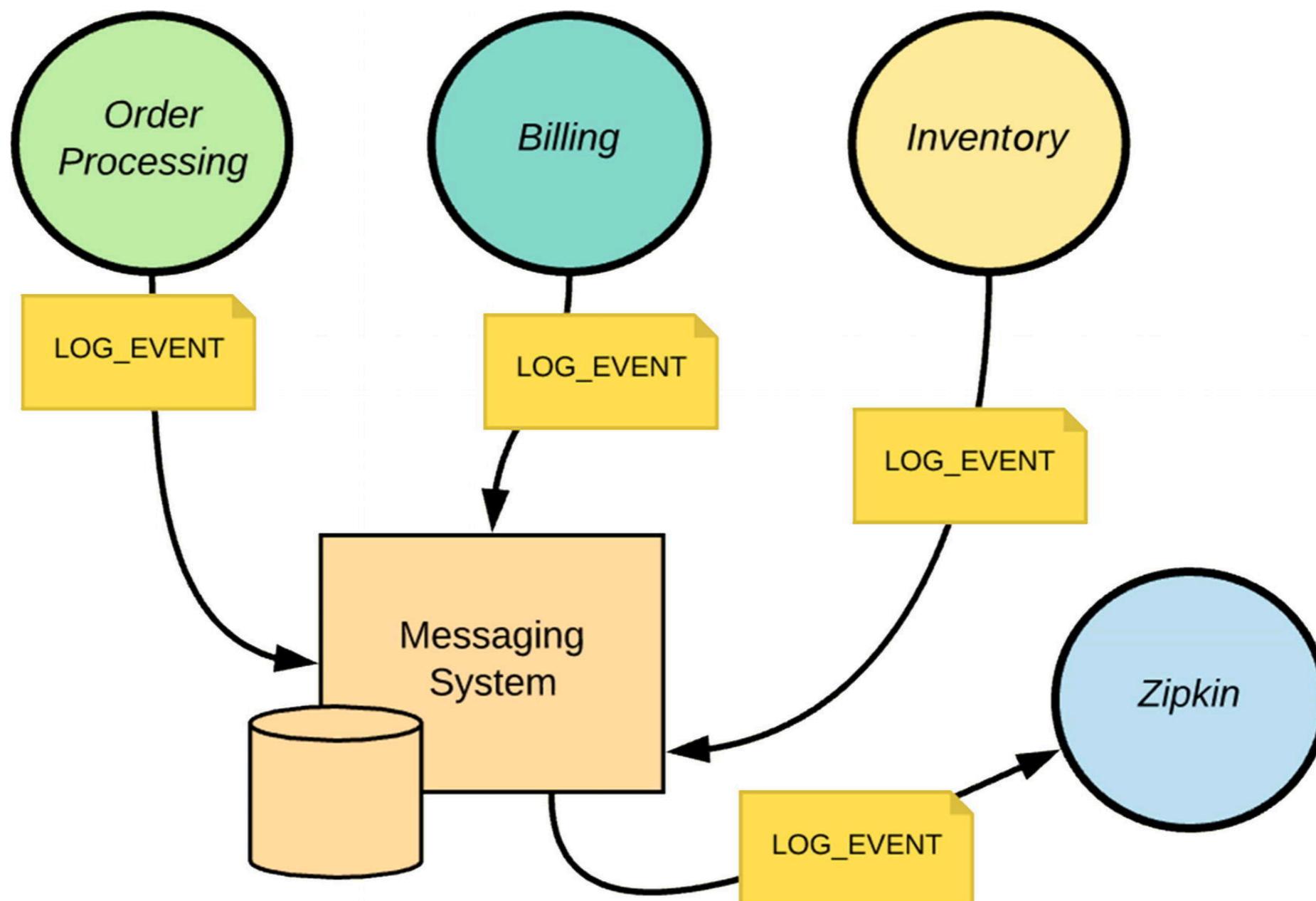
Distributed tracing with Zipkin



<https://zipkin.io/>

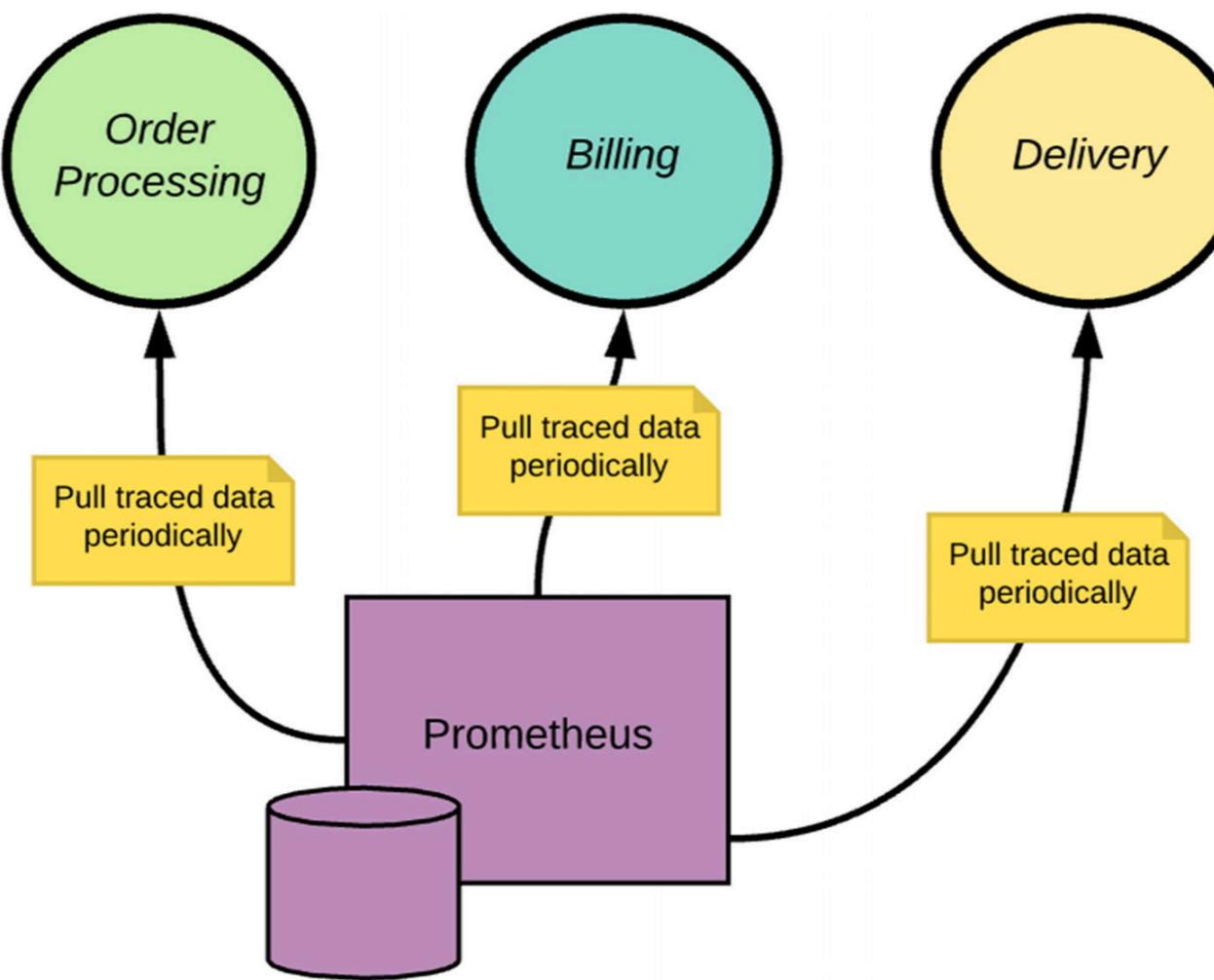


Distributed tracing with Zipkin



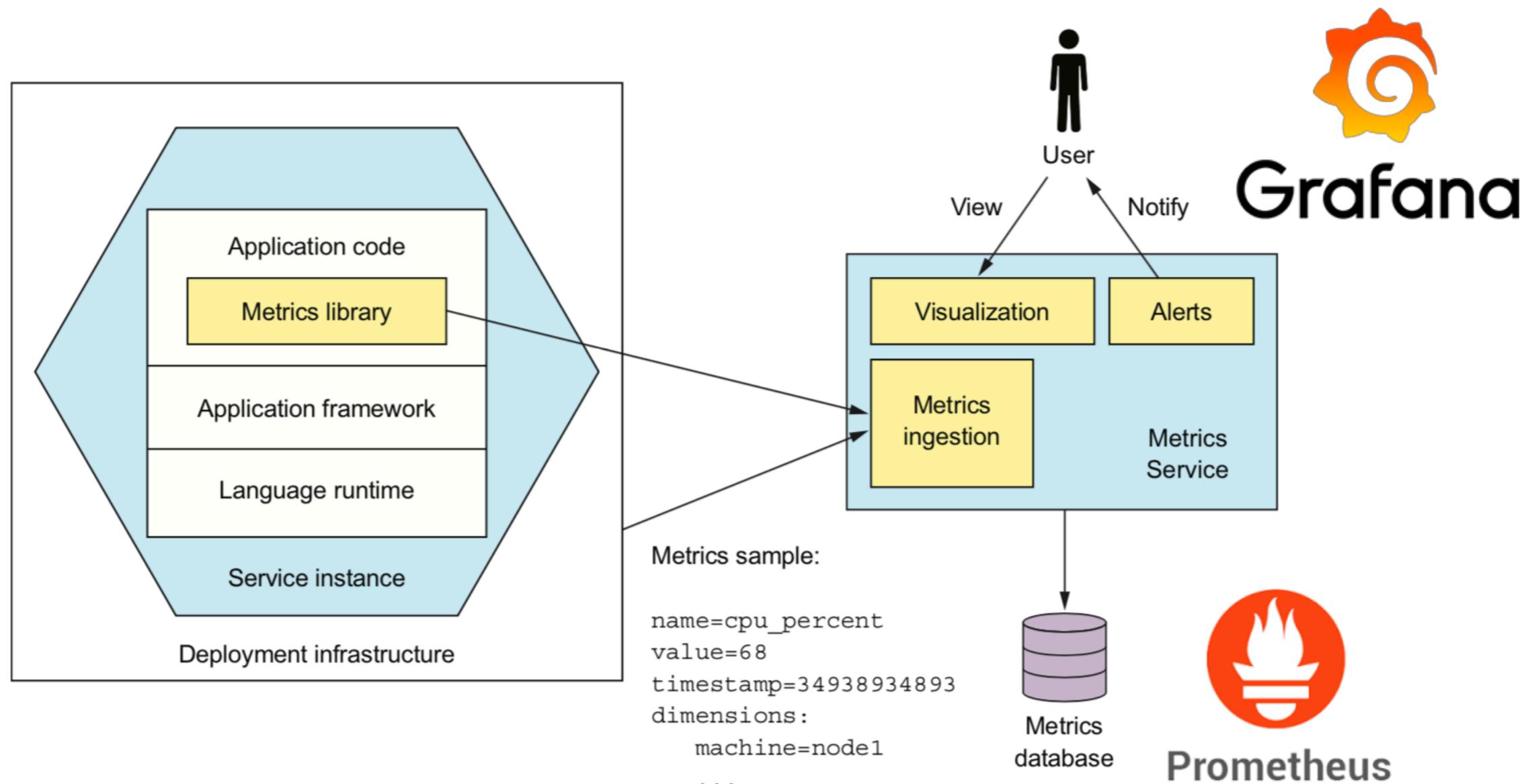
Application metrics

Services maintain metrics and expose to metric server (counters, gauges)



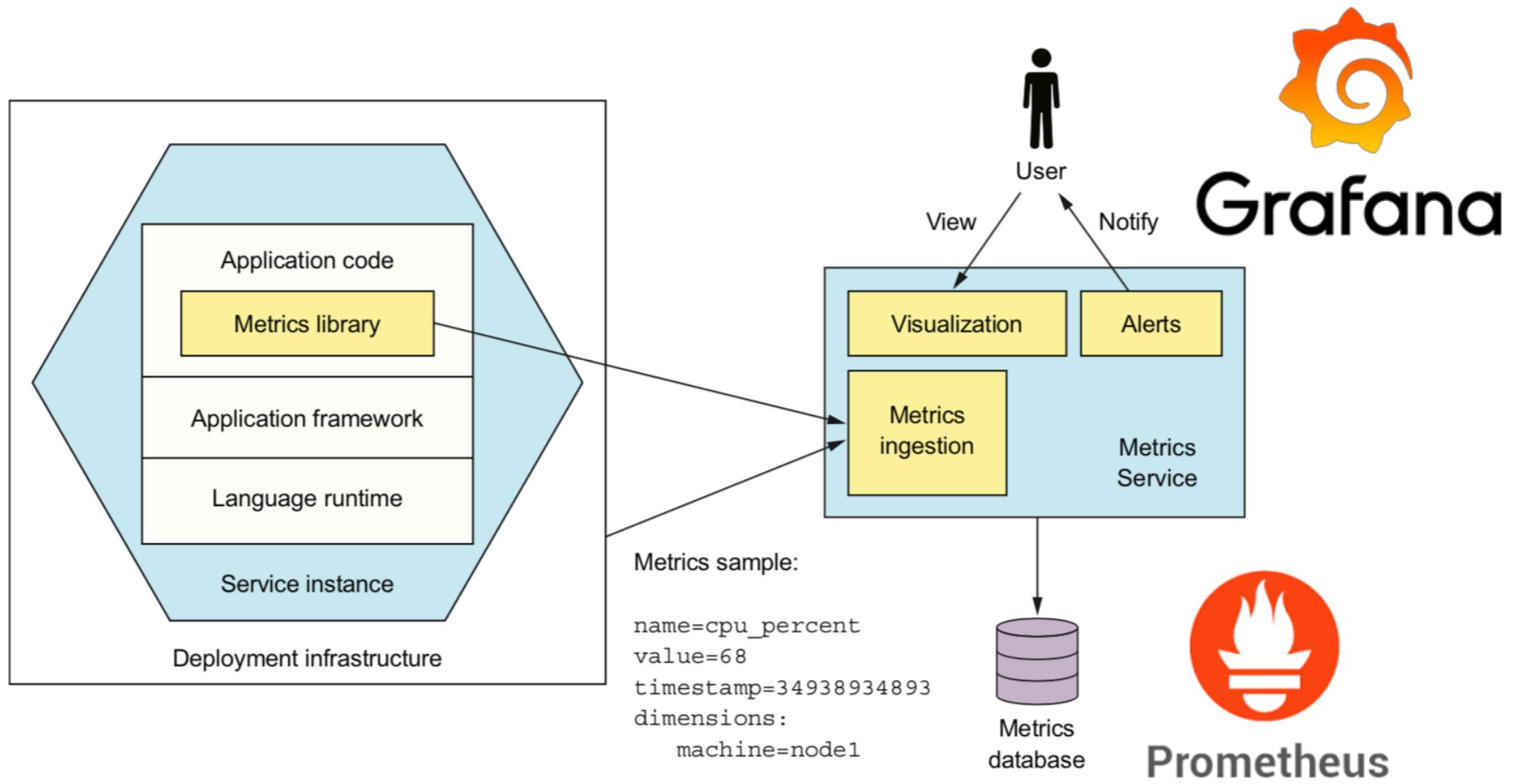
Application metrics

Services maintain metrics and expose to metric server (counters, gauges)



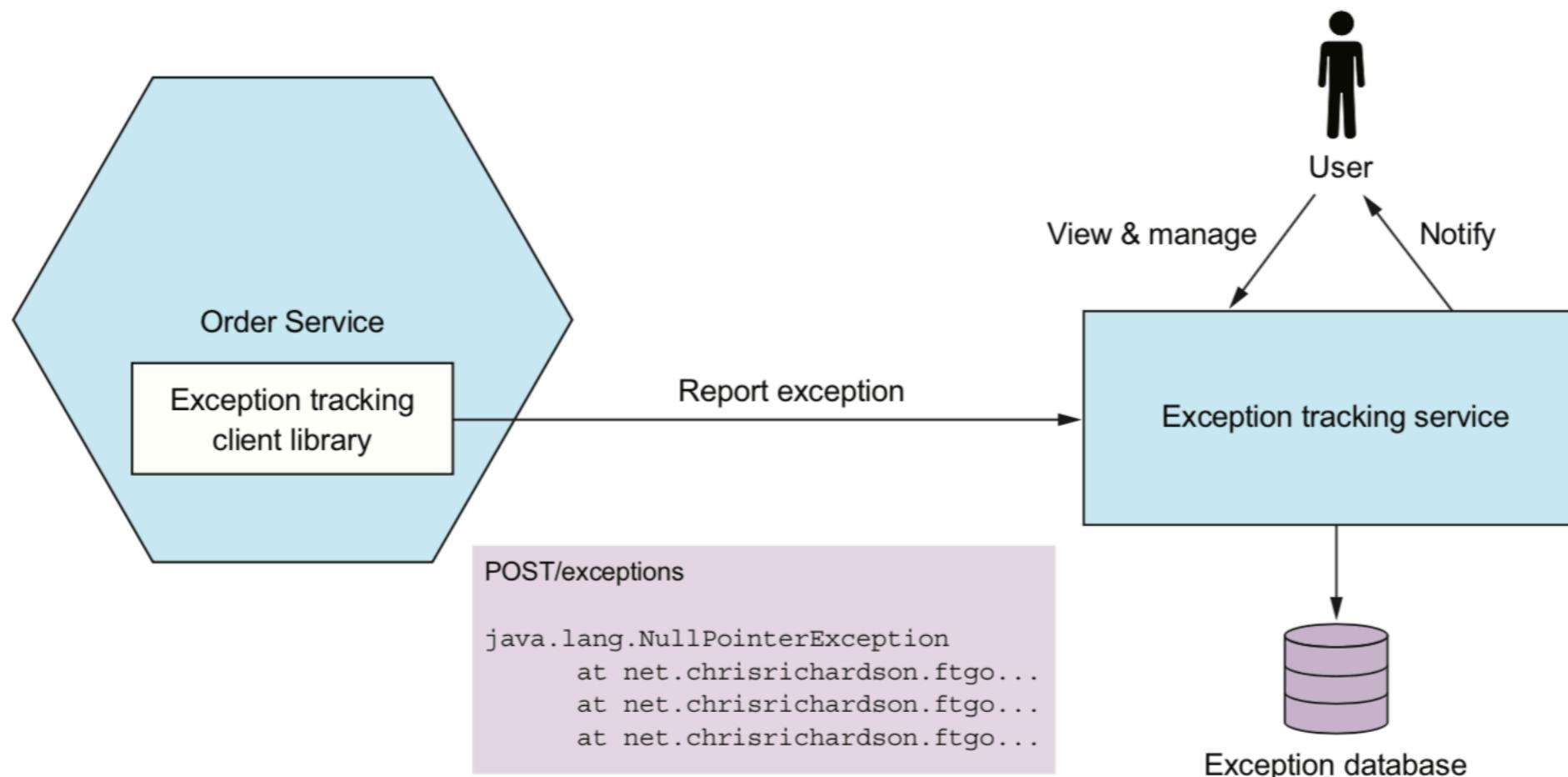
Application metrics

Metrics database => Prometheus
Visualization, Alert => Grafana



Exception tracking

Report exceptions to exception tracking service
Help to identify the root cause



Exception tracking services



Audit logging

Log of user actions

Help customer support

Ensure compliance

Detect suspicious behaviour



How to implement the audit logging ?

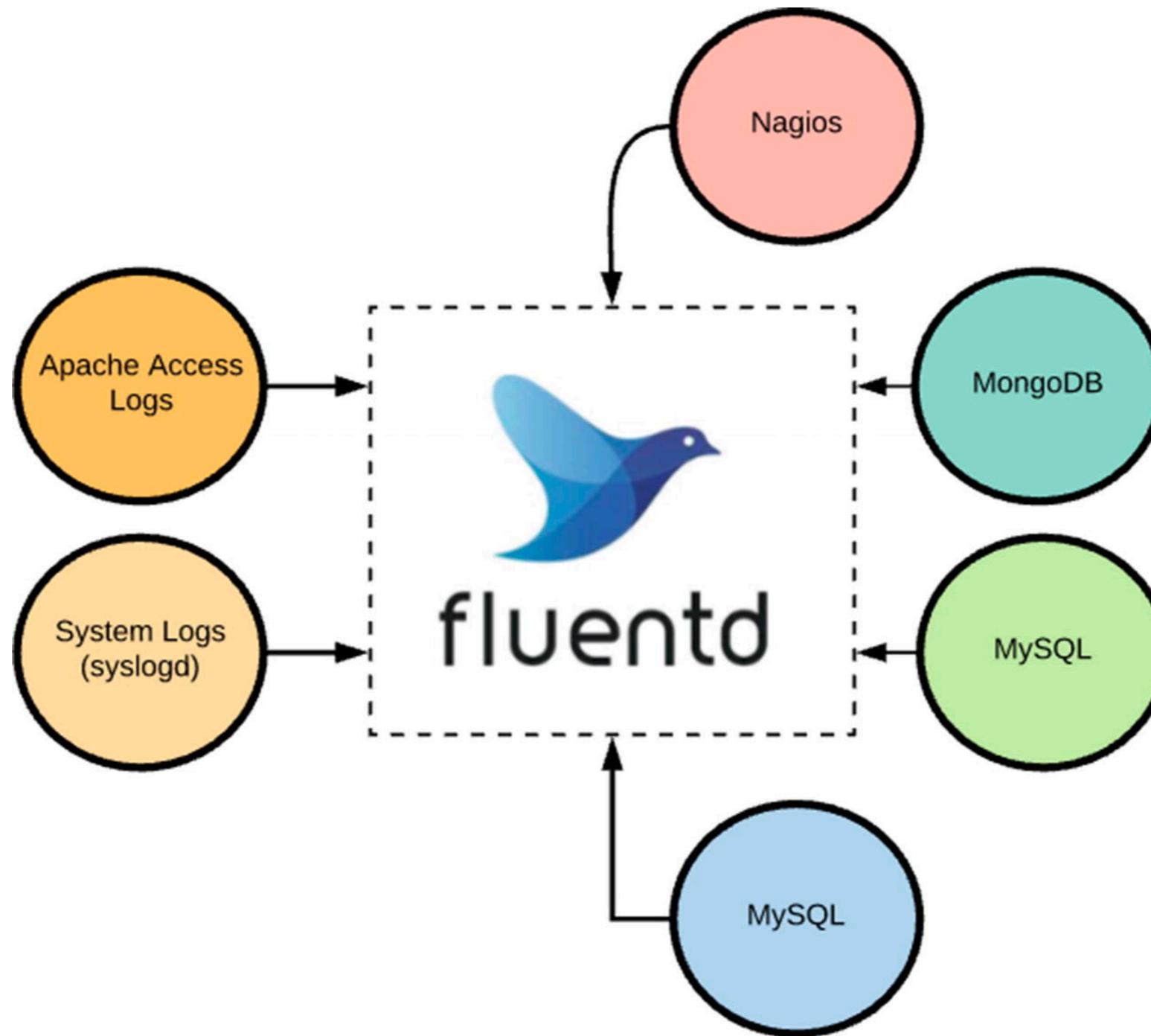
Add logging code in business logic

Use AOP (Aspect-Oriented Programming)

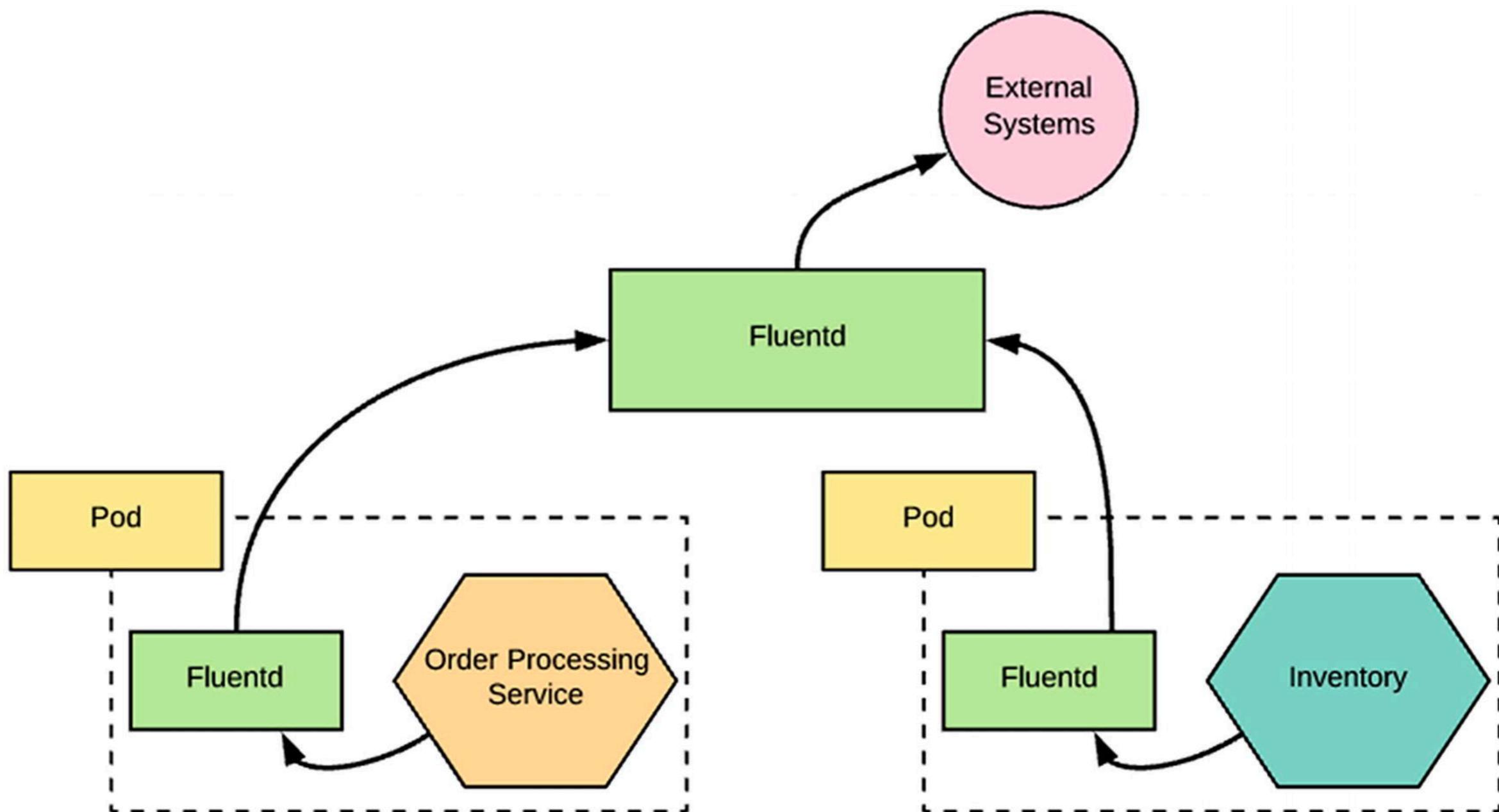
Use event sourcing



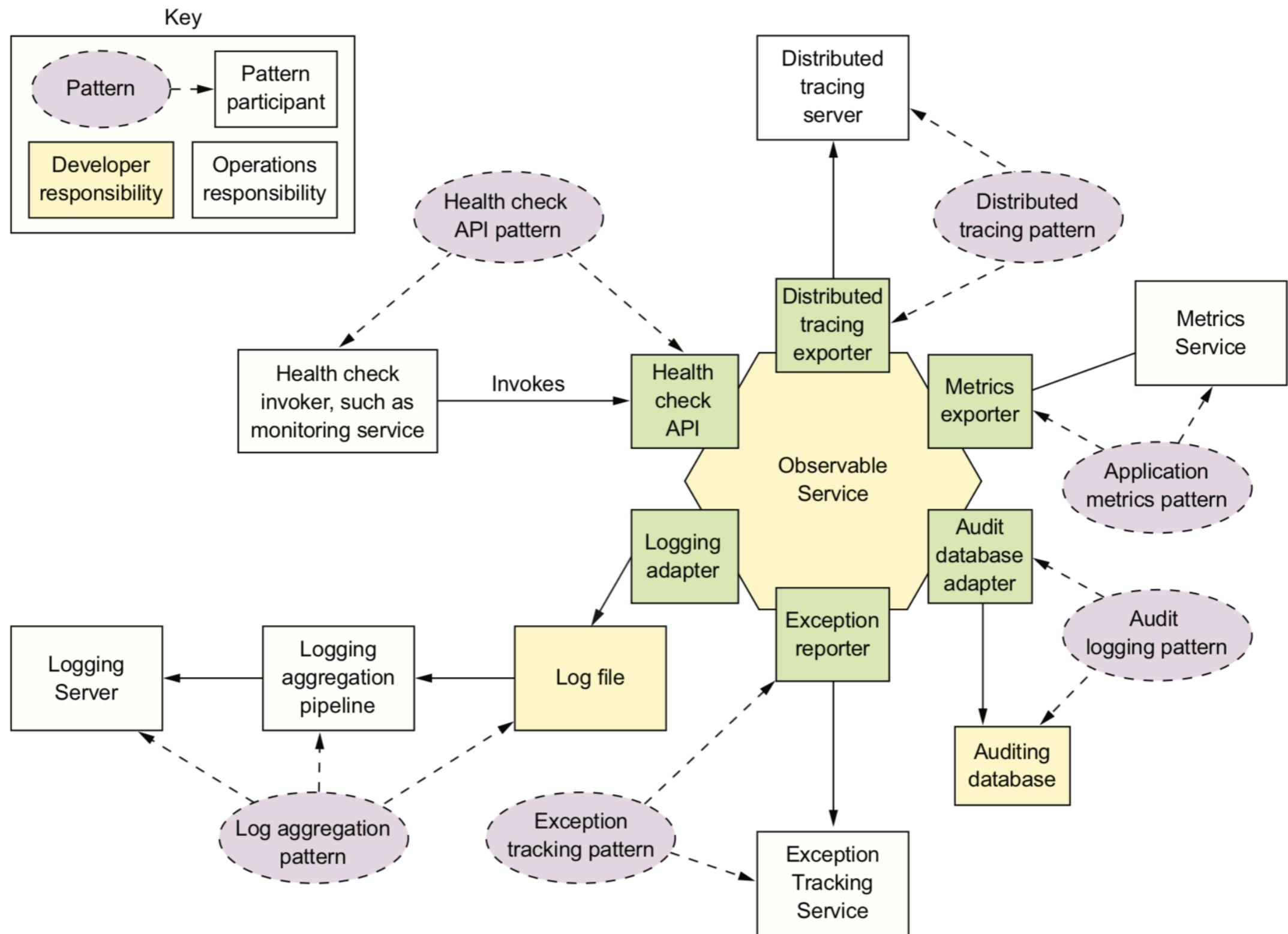
Centralize logging



Centralize logging



Observable services



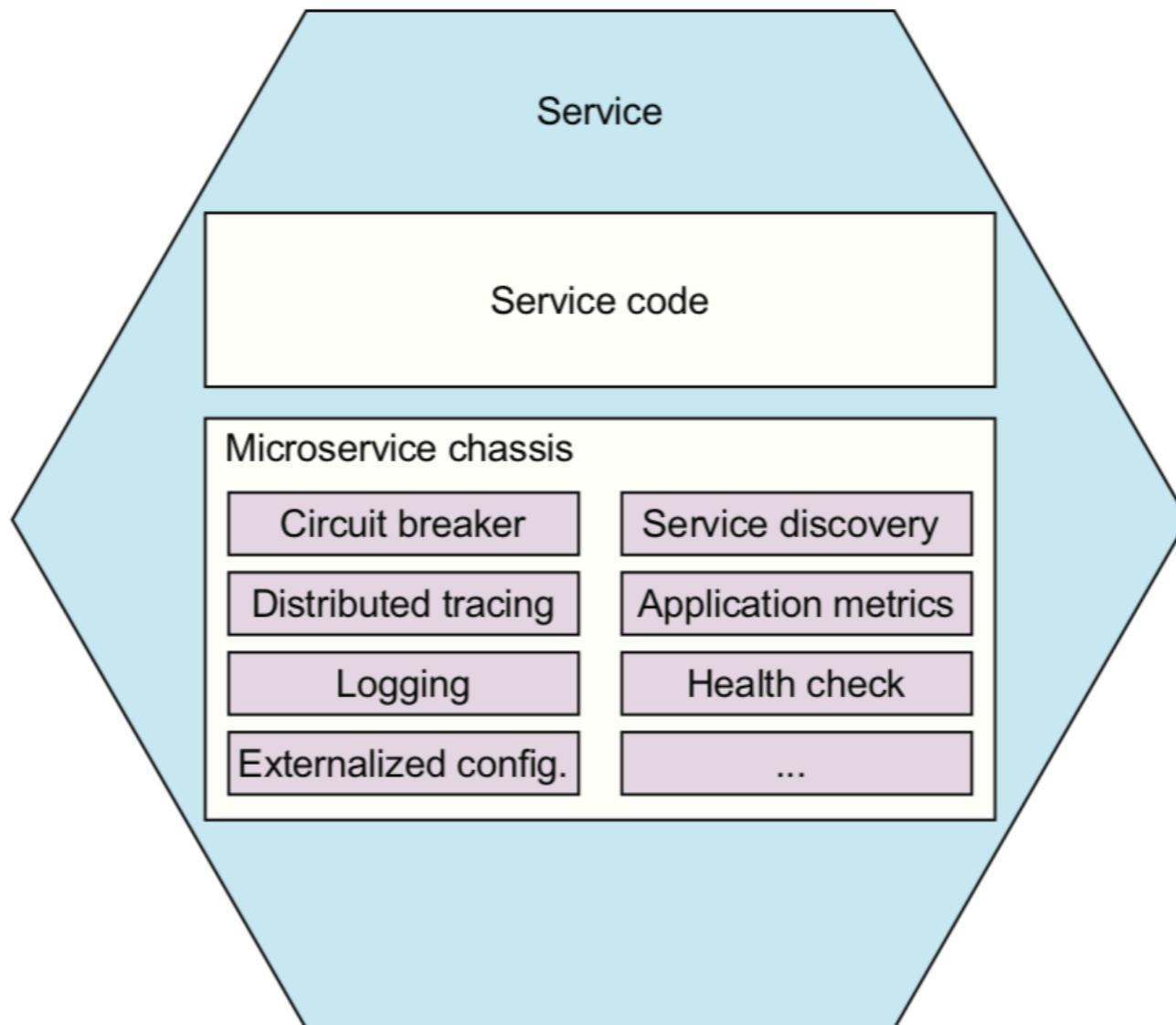
Workshop



Microservice framework



Microservice framework

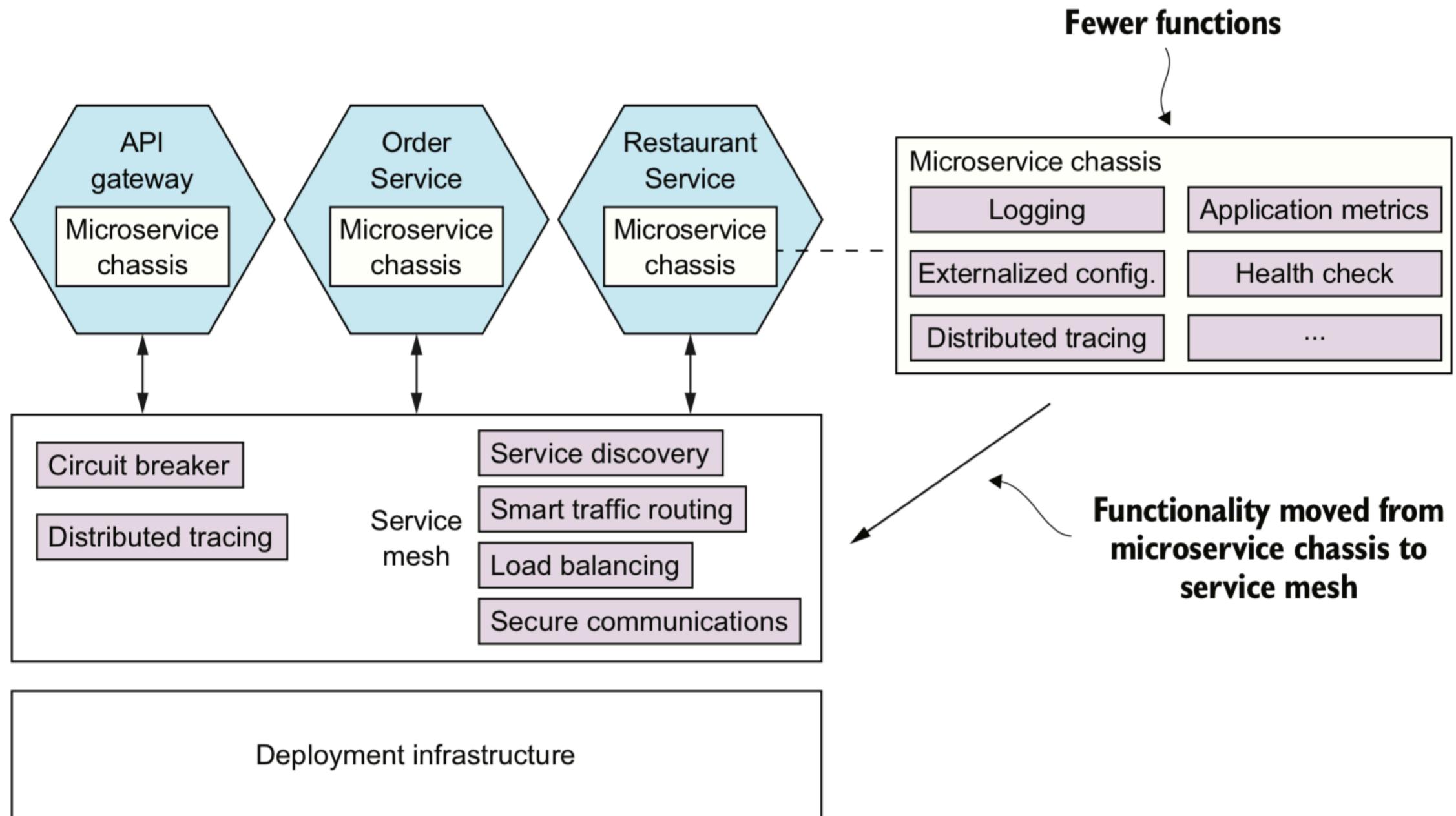


Microservice framework

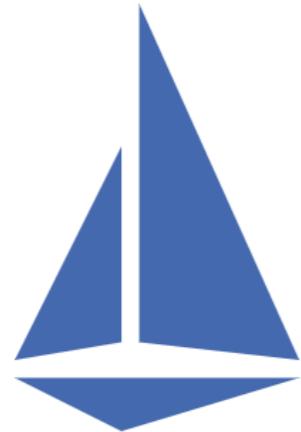
Programming language	Framework
Java	Spring boot Spring cloud
Golang	Go-kit Micro



Service Mesh



Service Mesh



Istio

Connect, secure, control, and observe services.



CONDUIT



linkerd



Module 3 : Deploy

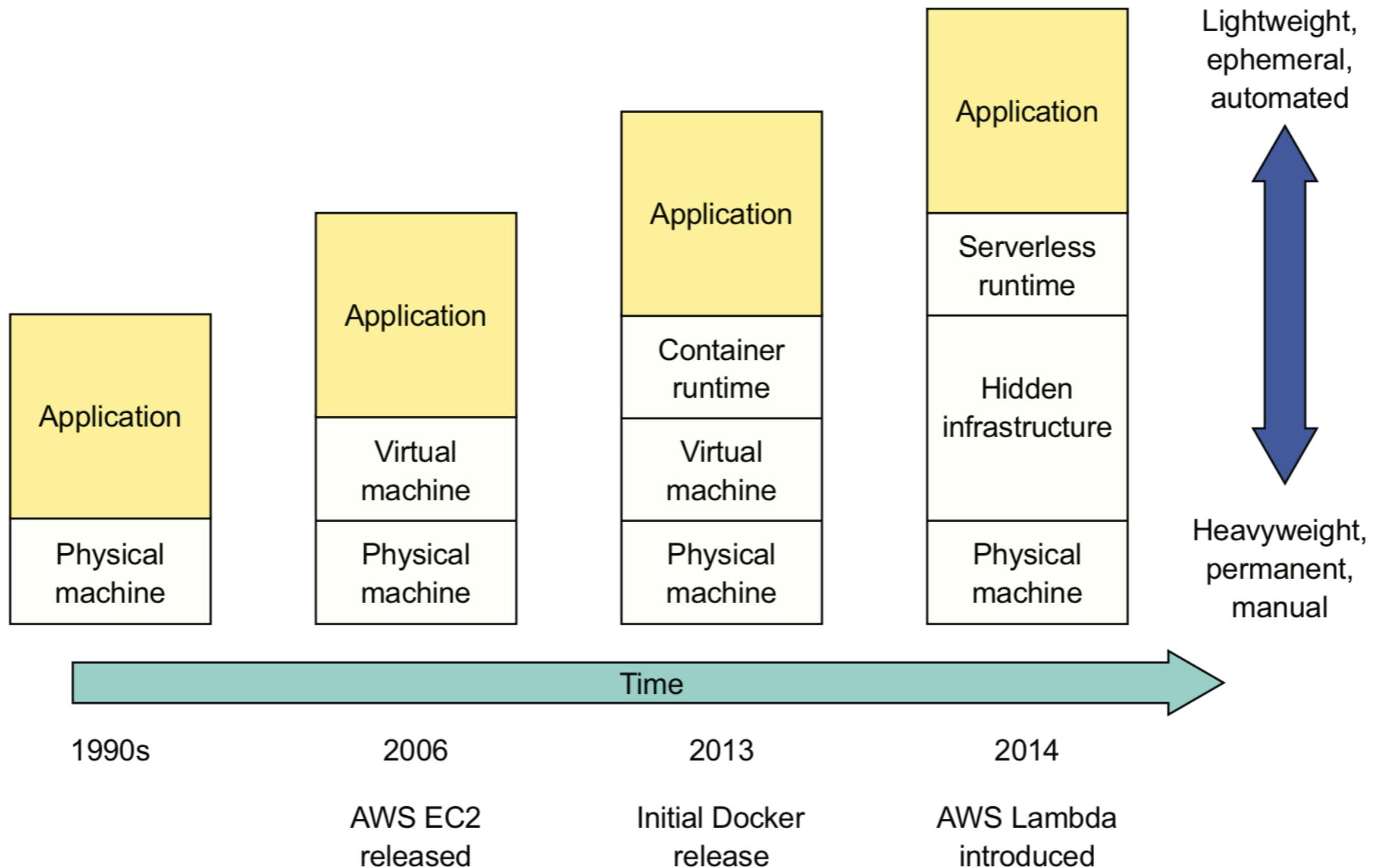




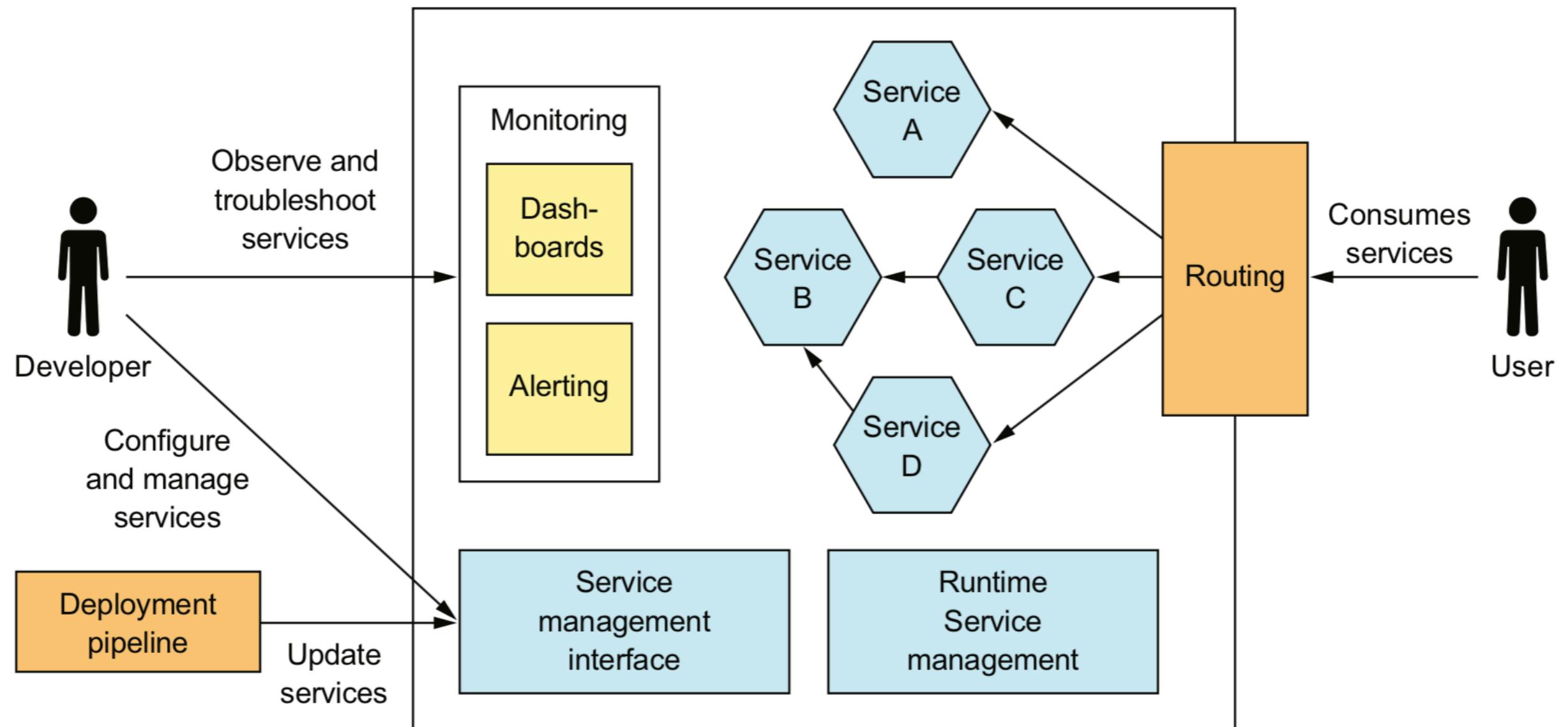
@redbadgerteam



Infrastructure



View of production environment



Deploy Microservices

Language-specific packaging

Virtual Machine (VM)

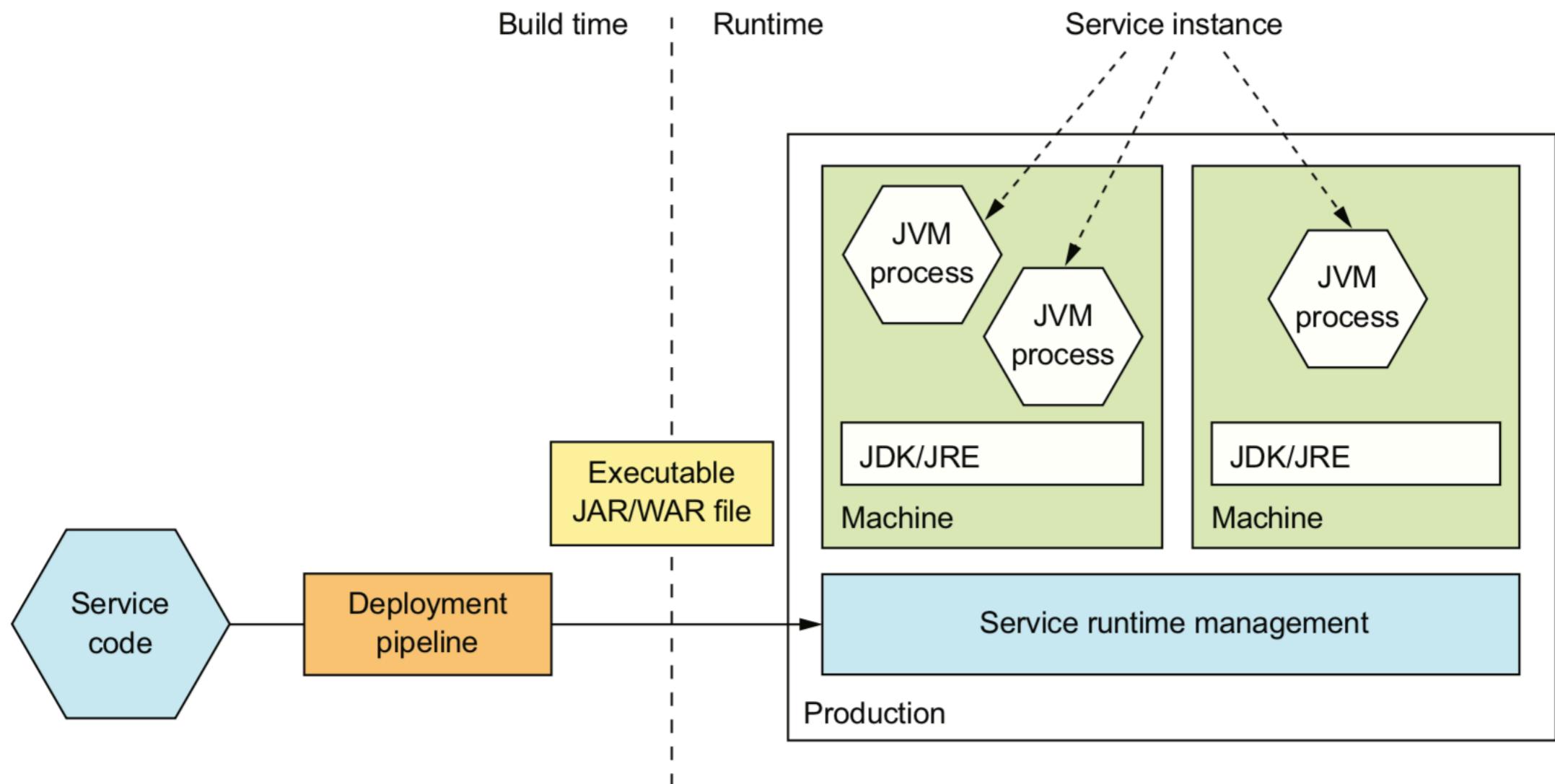
Container

Kubernetes

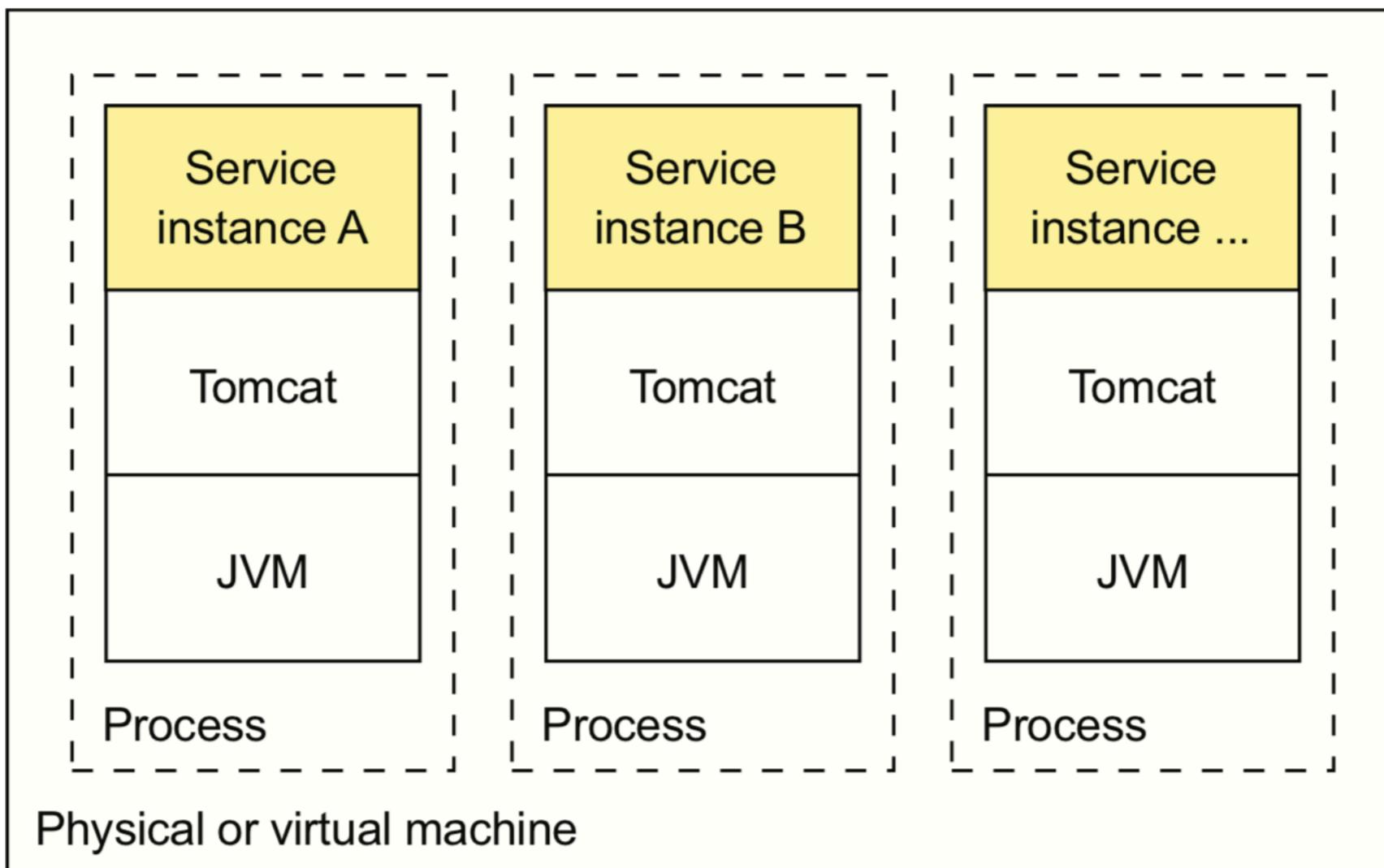
Serverless/FaaS



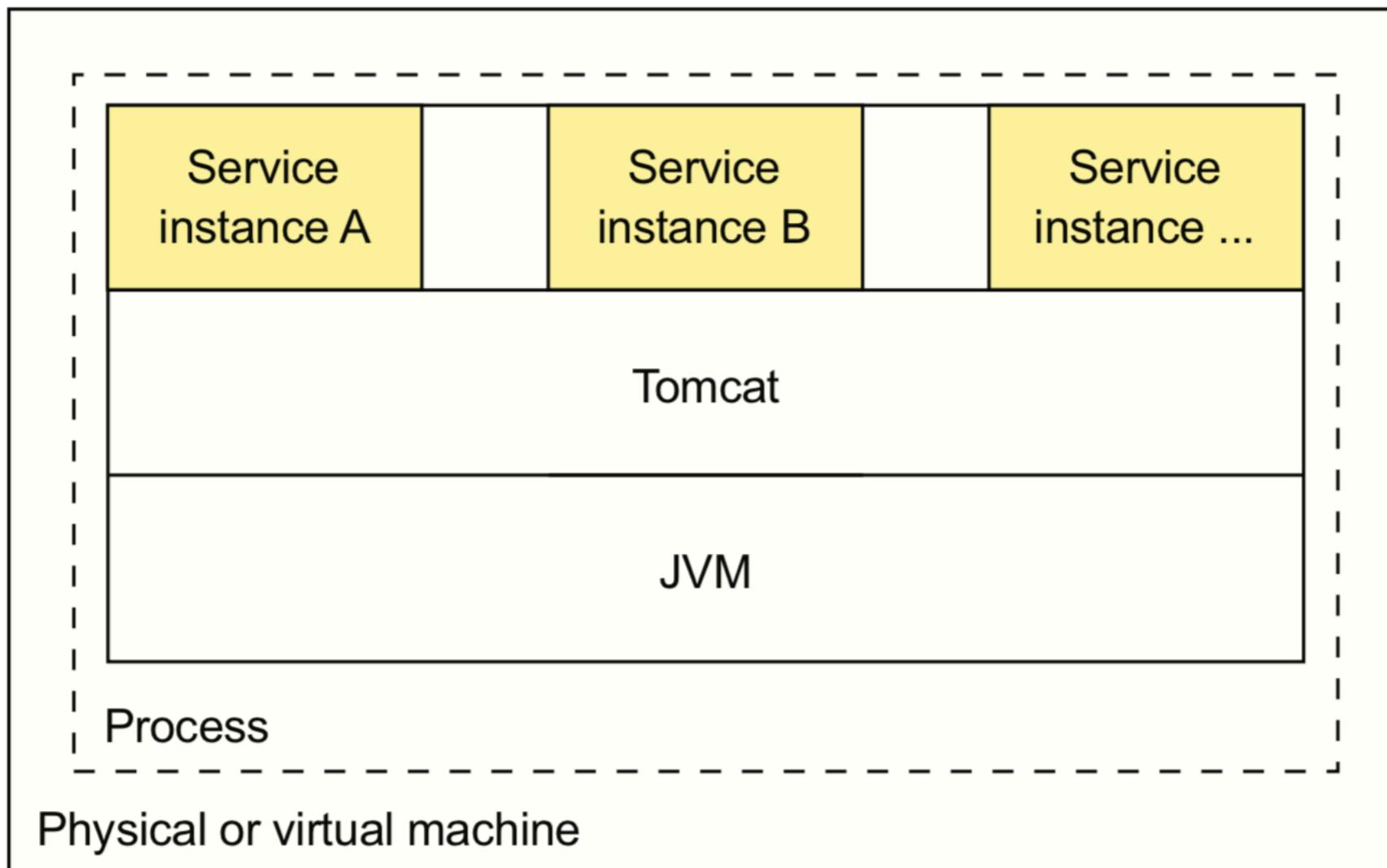
1. Language-specific packaging



Multiple services on same machine



Multiple services on same process



Benefits

Fast deployment

Efficient resource utilization

Service instances's resources are constrained



Drawbacks

Lack of encapsulation of technology stack

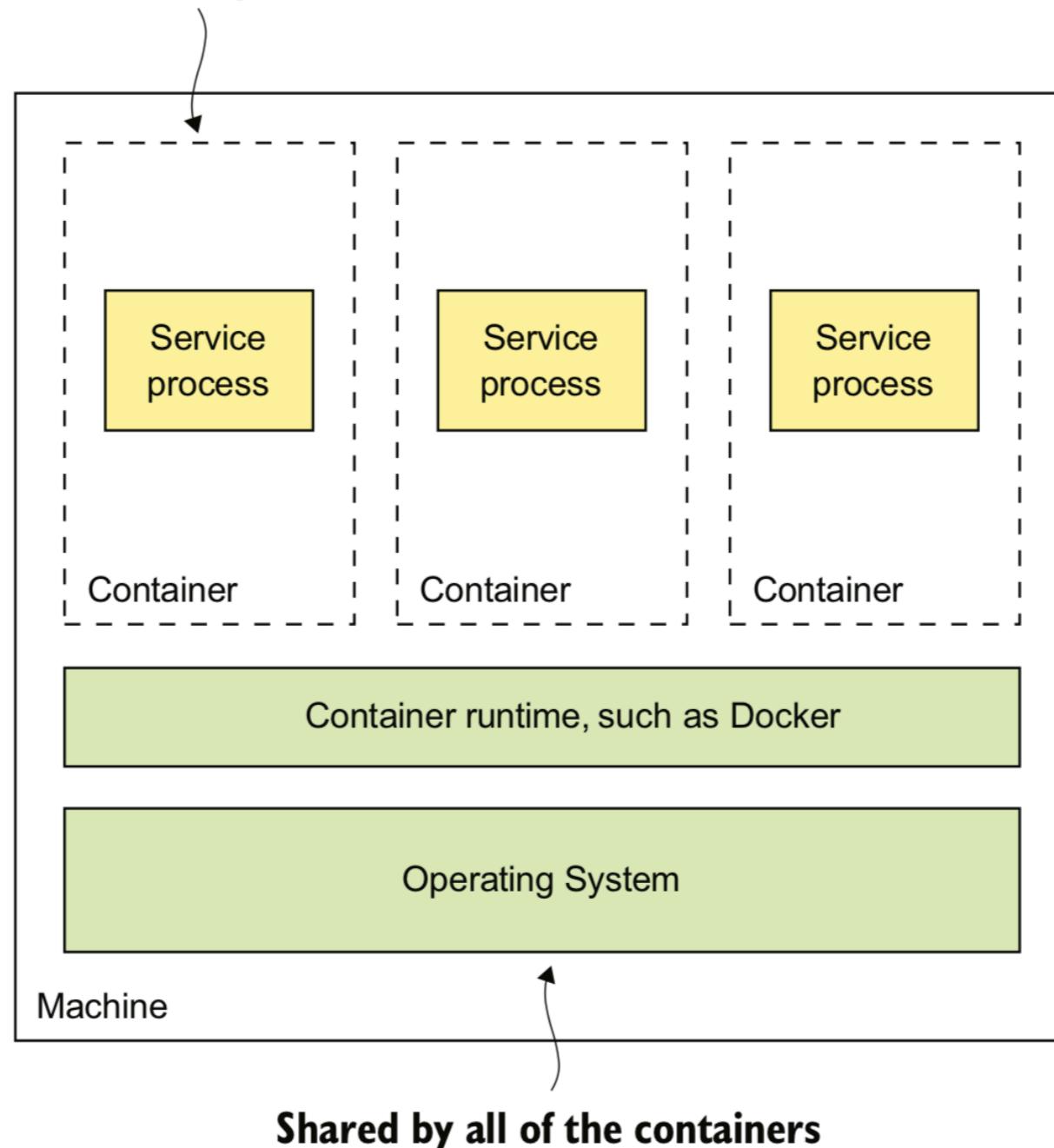
No ability to constrain resources of service

Lack of isolation

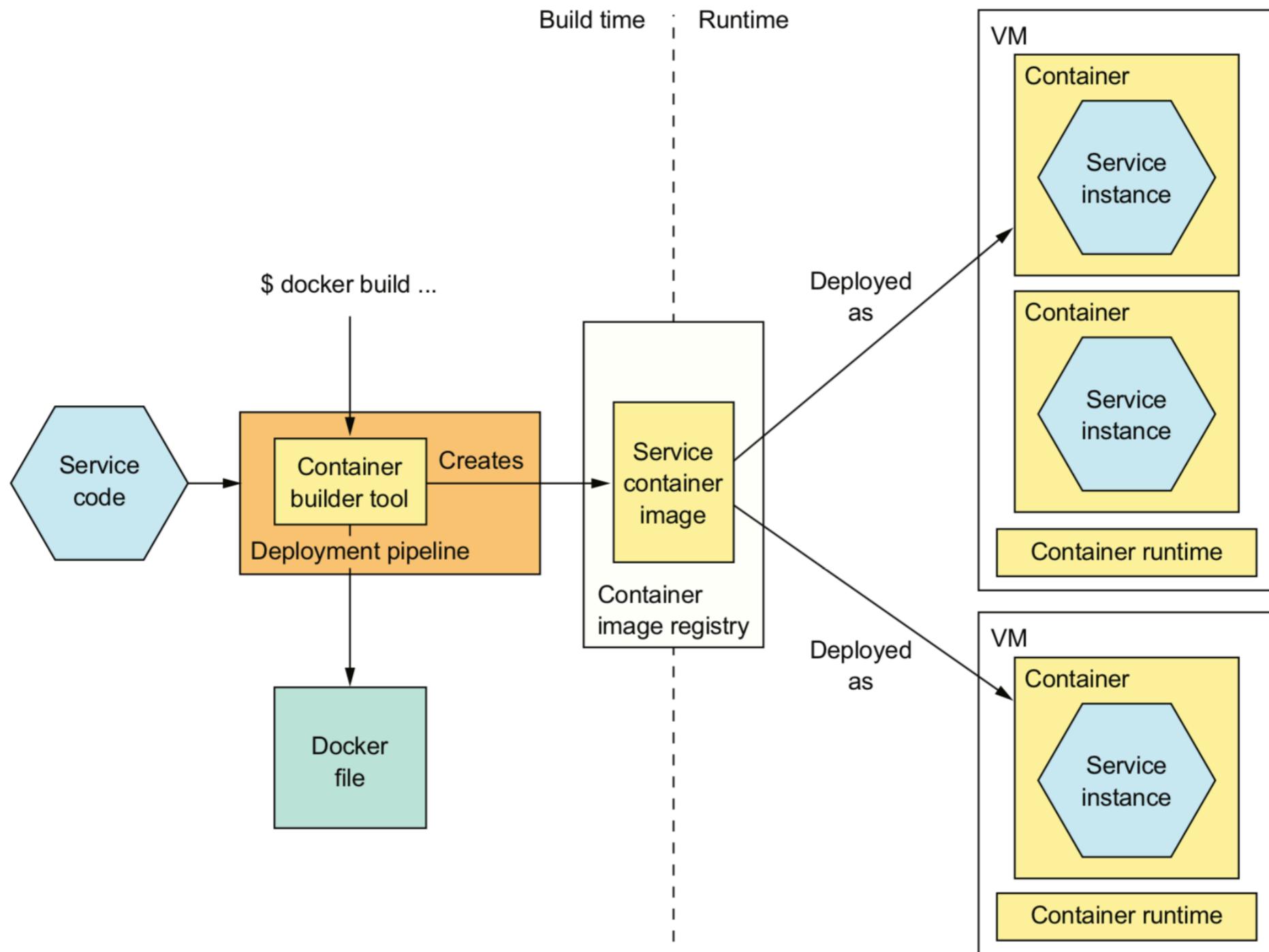


2. Working with container

**Each container is a sandbox
that isolates the processes.**



Deployment with container



Deploy services with Docker

Build a docker image

Push docker image to a registry

Run docker container

Working with docker-compose



Benefits

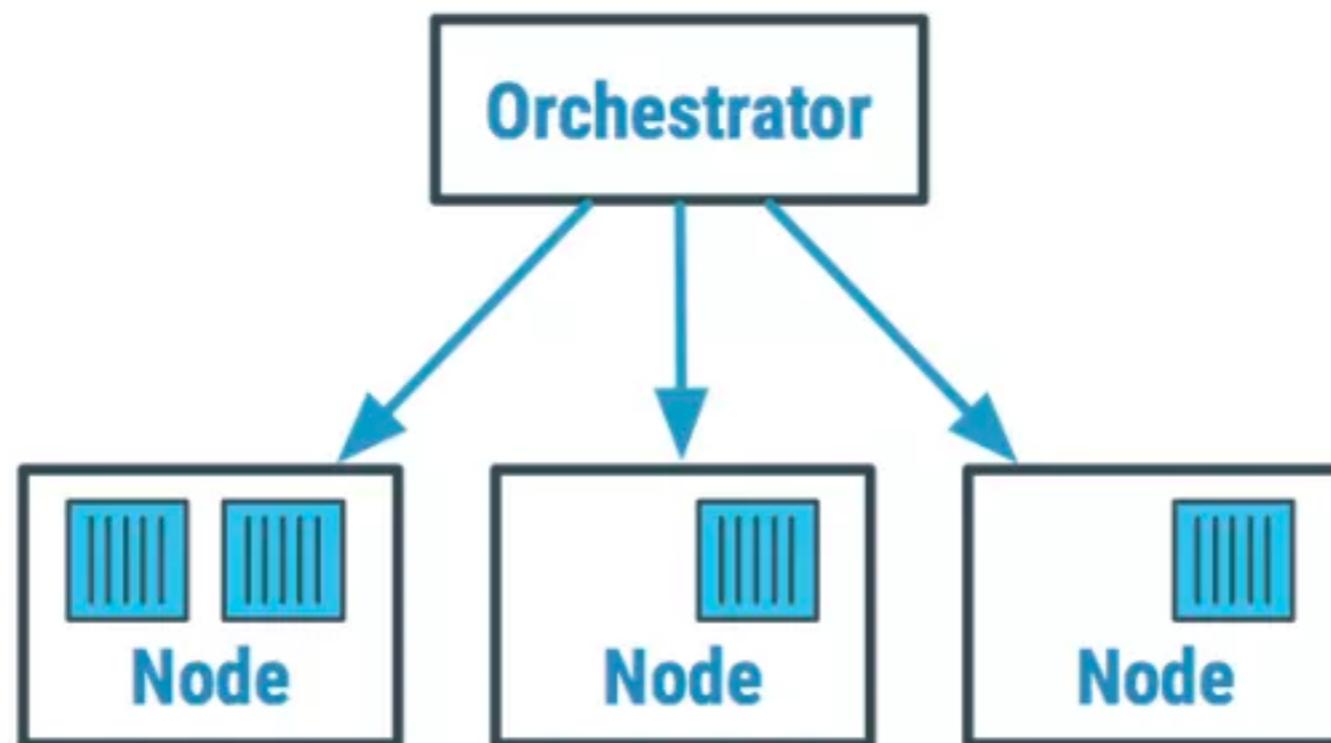
Encapsulate technology stack

Service instances are isolated

Service instances's resources are constrained



Container Orchestration ?



Orchestration tools

Configuration Management



CI/CD orchestration



Container orchestration



Cloud-specific orchestration



PaaS orchestration



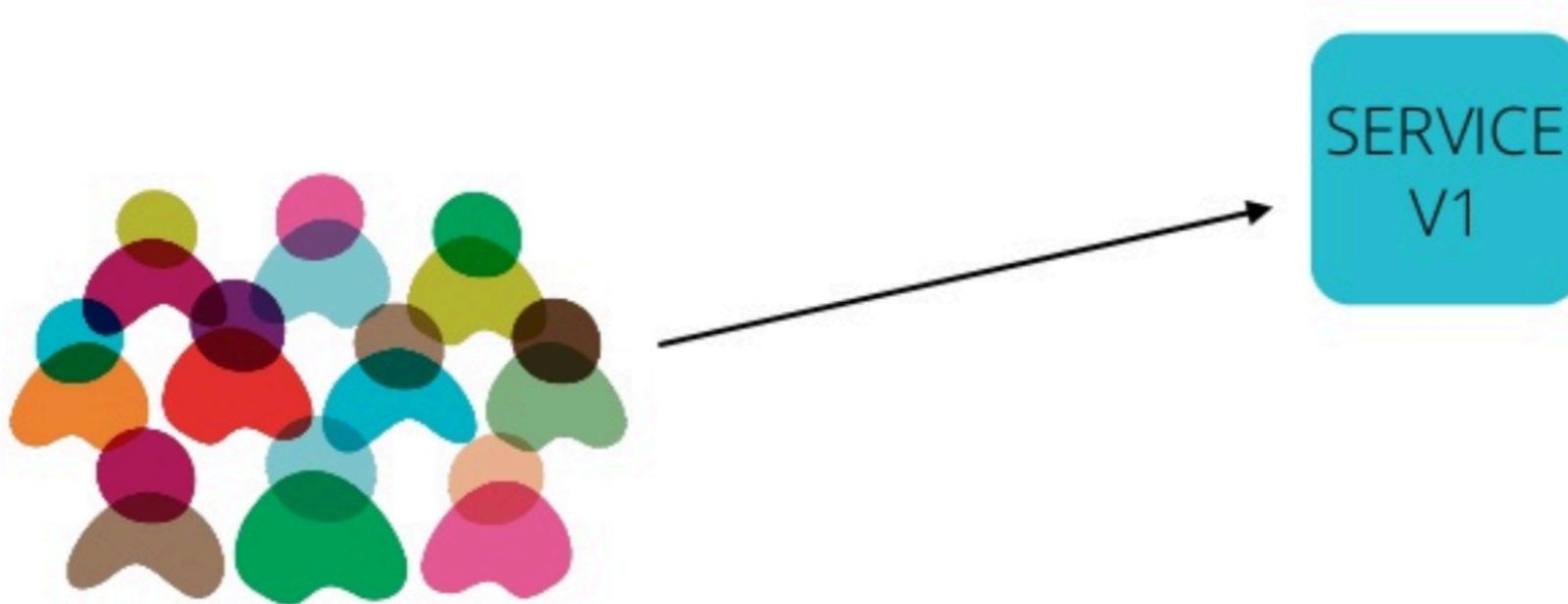
Deployment strategies



Blue Green Deployment



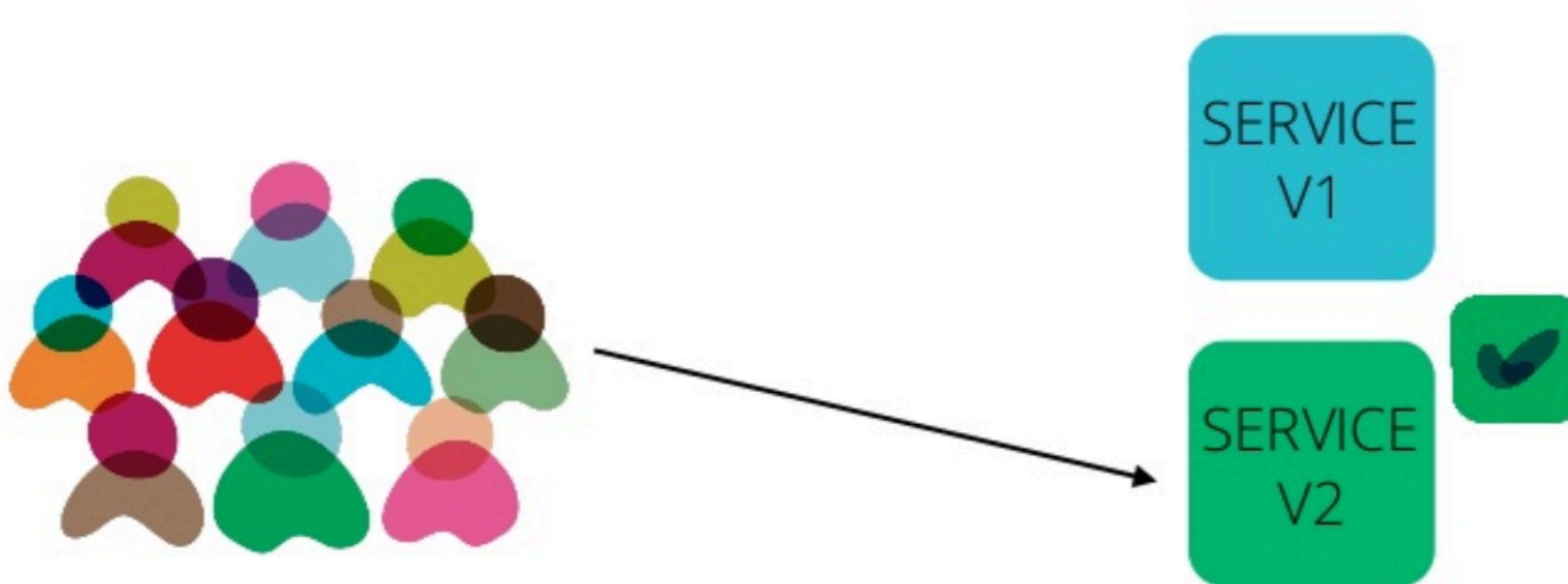
Blue Green Deployment



Blue Green Deployment



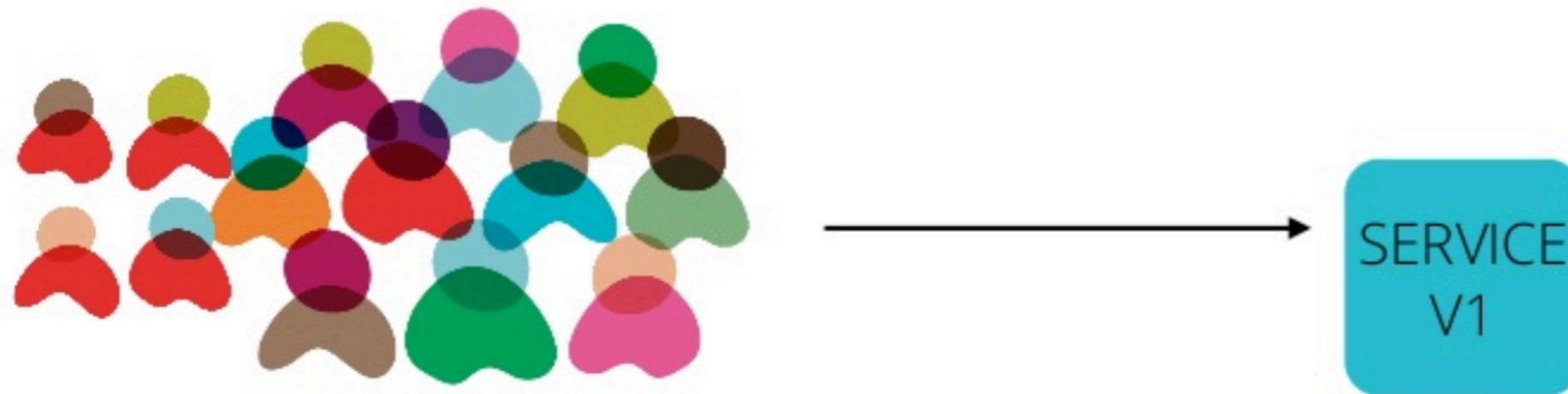
Blue Green Deployment



Canary Release



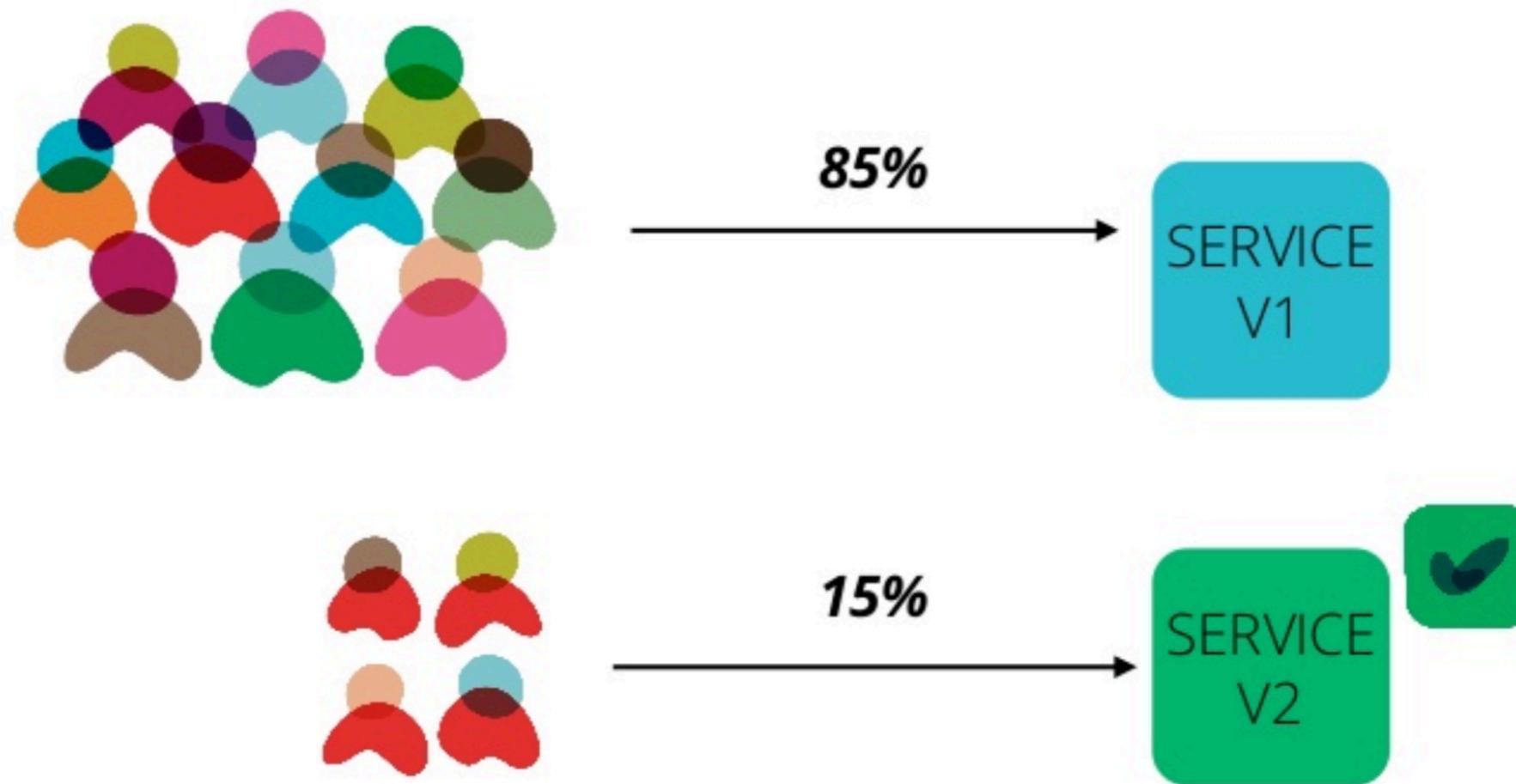
Canary Release



Canary Release



Canary Release



Mean Time to Recover (MTTR)



Mean Time to Recover (MTTR)

Tests are very important to reduce amount of defects in your systems. However, it's important to acknowledge that bugs will always happen in production.

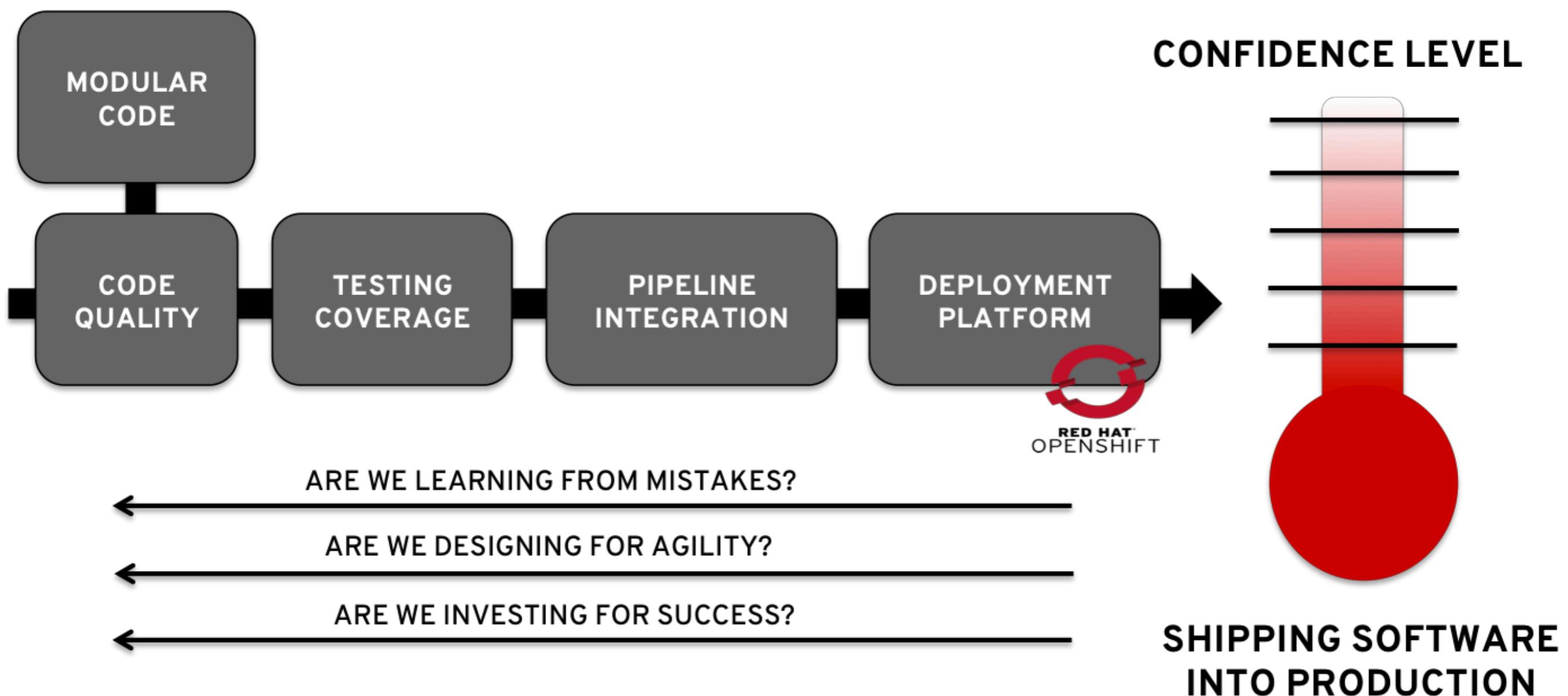


Mean Time to Recover (MTTR)

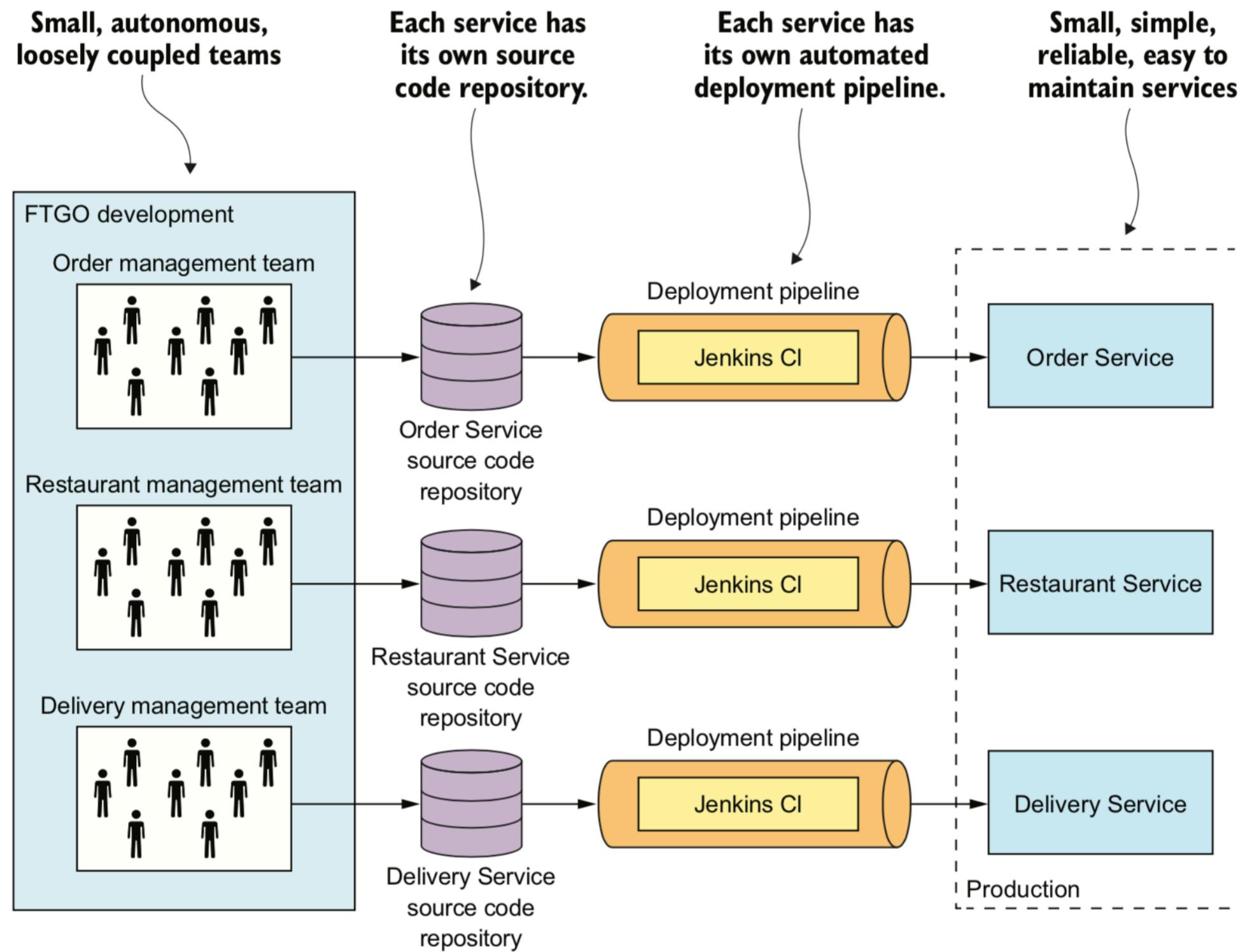
How **fast** to recover from them will help determining our success !



What can i measure ?



Deployment pipeline in each service



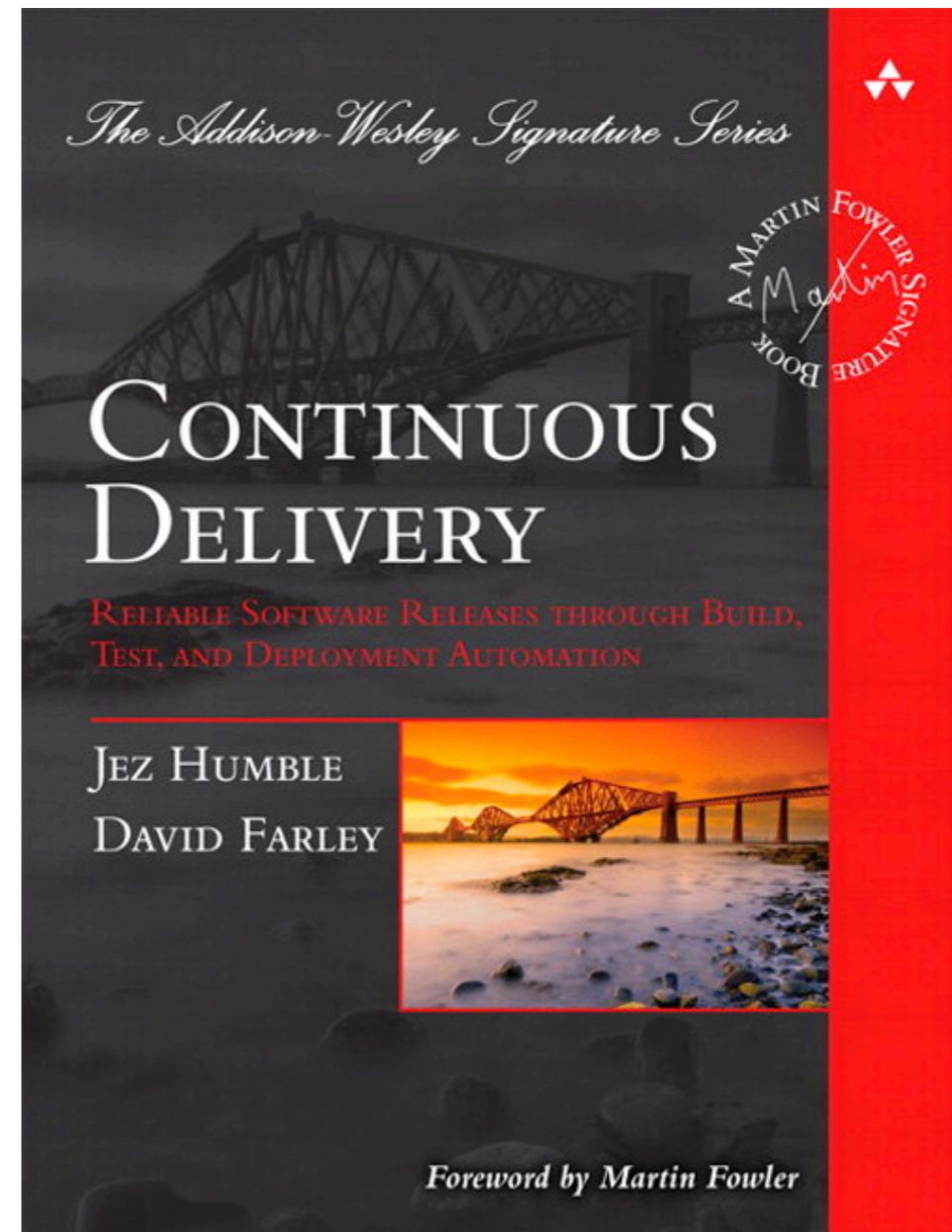
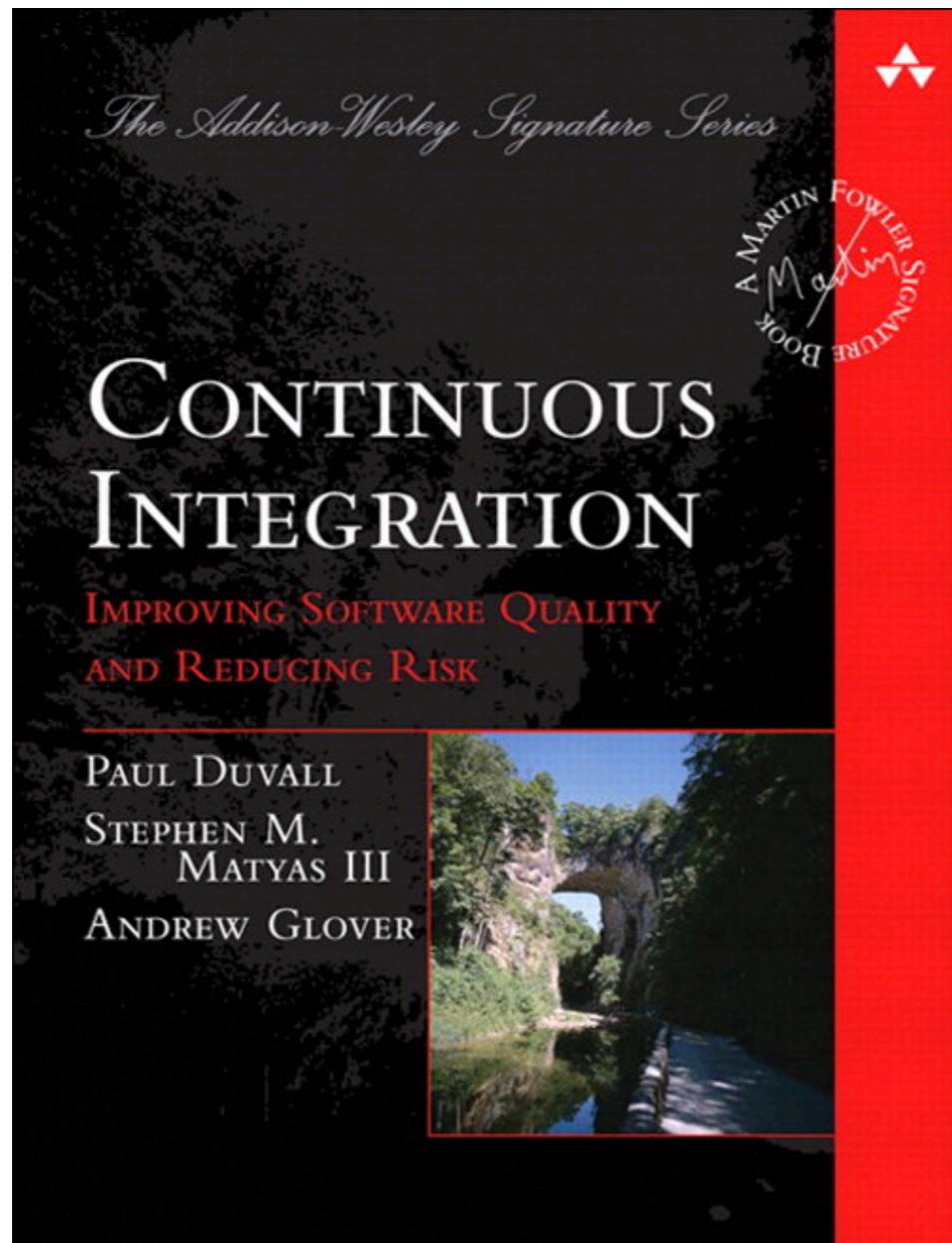
Start with Continuous Integration Continuous Delivery



**“Behind every successful agile
project, there is a
Continuous Integration System”**



Improve quality and reduce risk

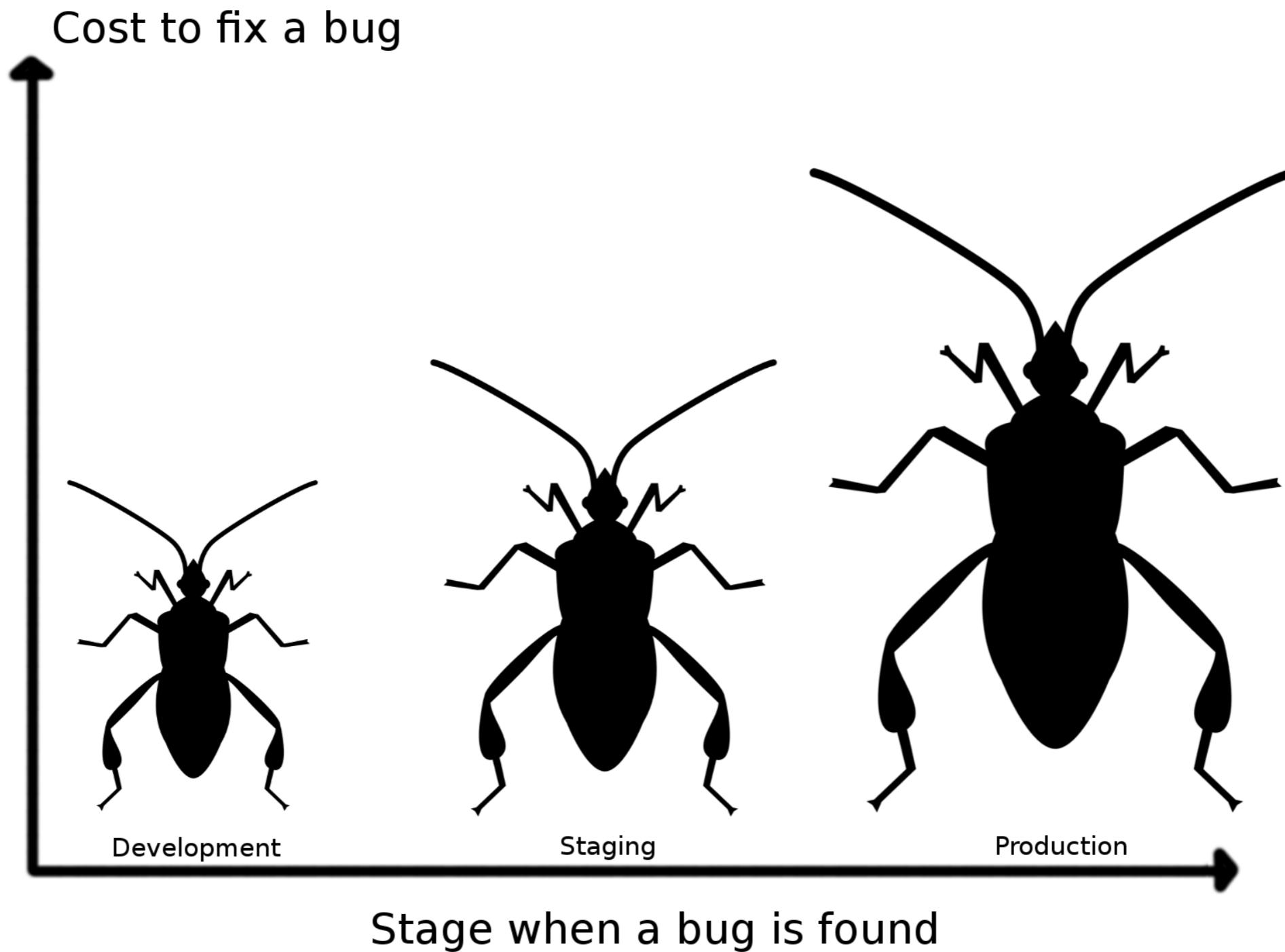


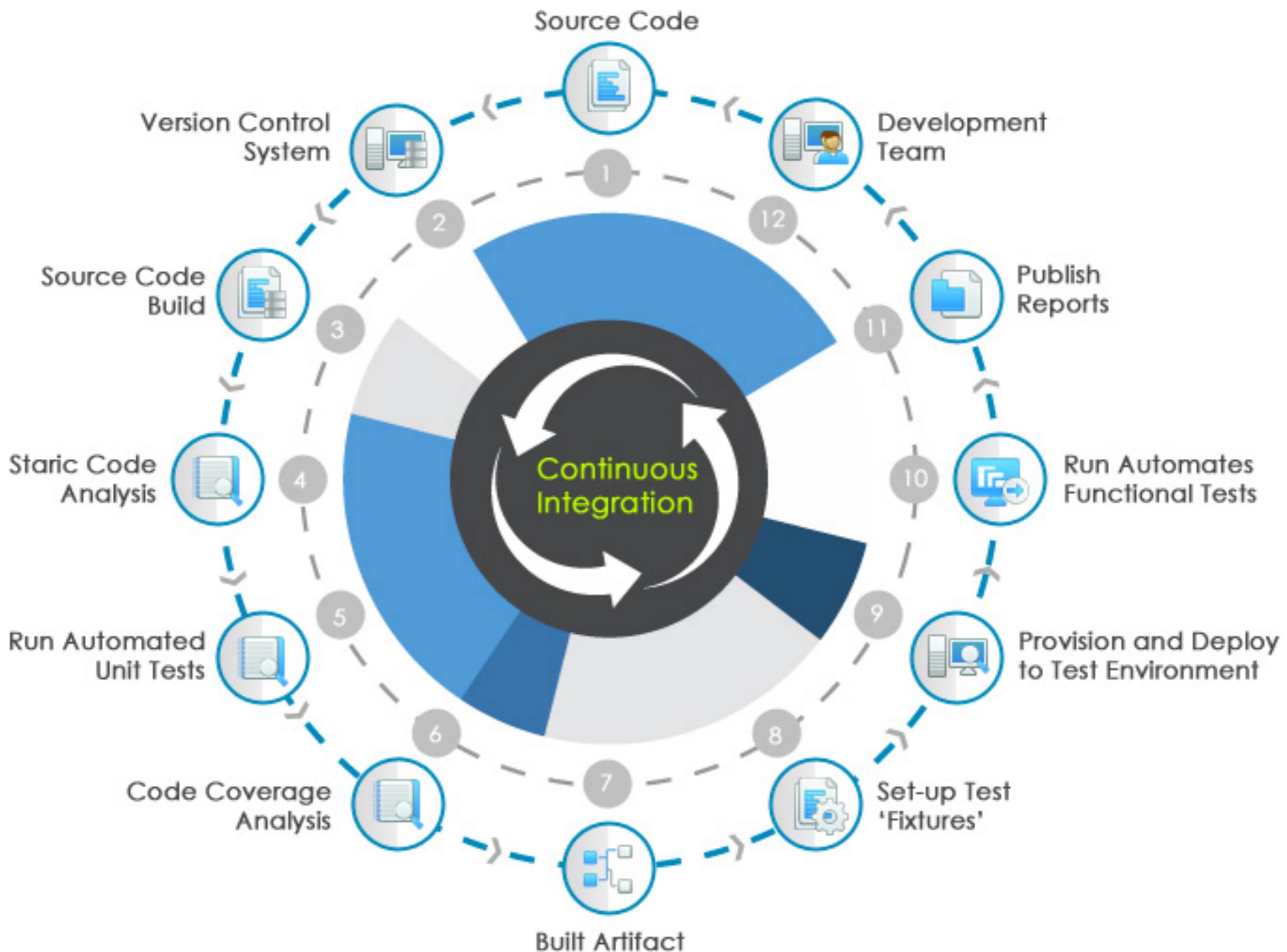
The cost of integration

1. Merging the code
2. Duplicate changes
3. Test again again !!
4. Fixing bugs
5. Impact on stability



The cost of integration







Jenkins

Bamboo



TeamCity



Hudson





Jenkins



Bamboo

CI is about what people do
not about what tools they use



Visual Studio



Team Foundation Server

Hudson



Microservices

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

Continuous Integration

Discipline to integrate frequently



Continuous Integration

Strive to make **small change**

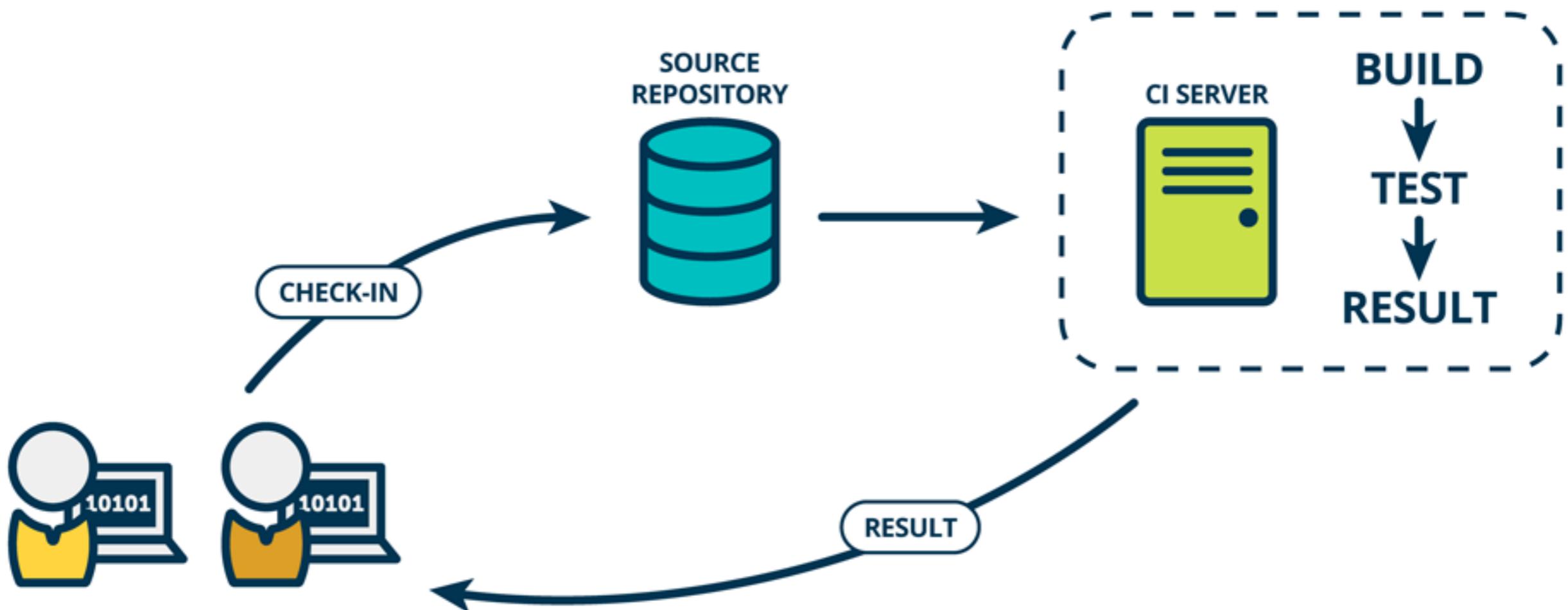


Continuous Integration

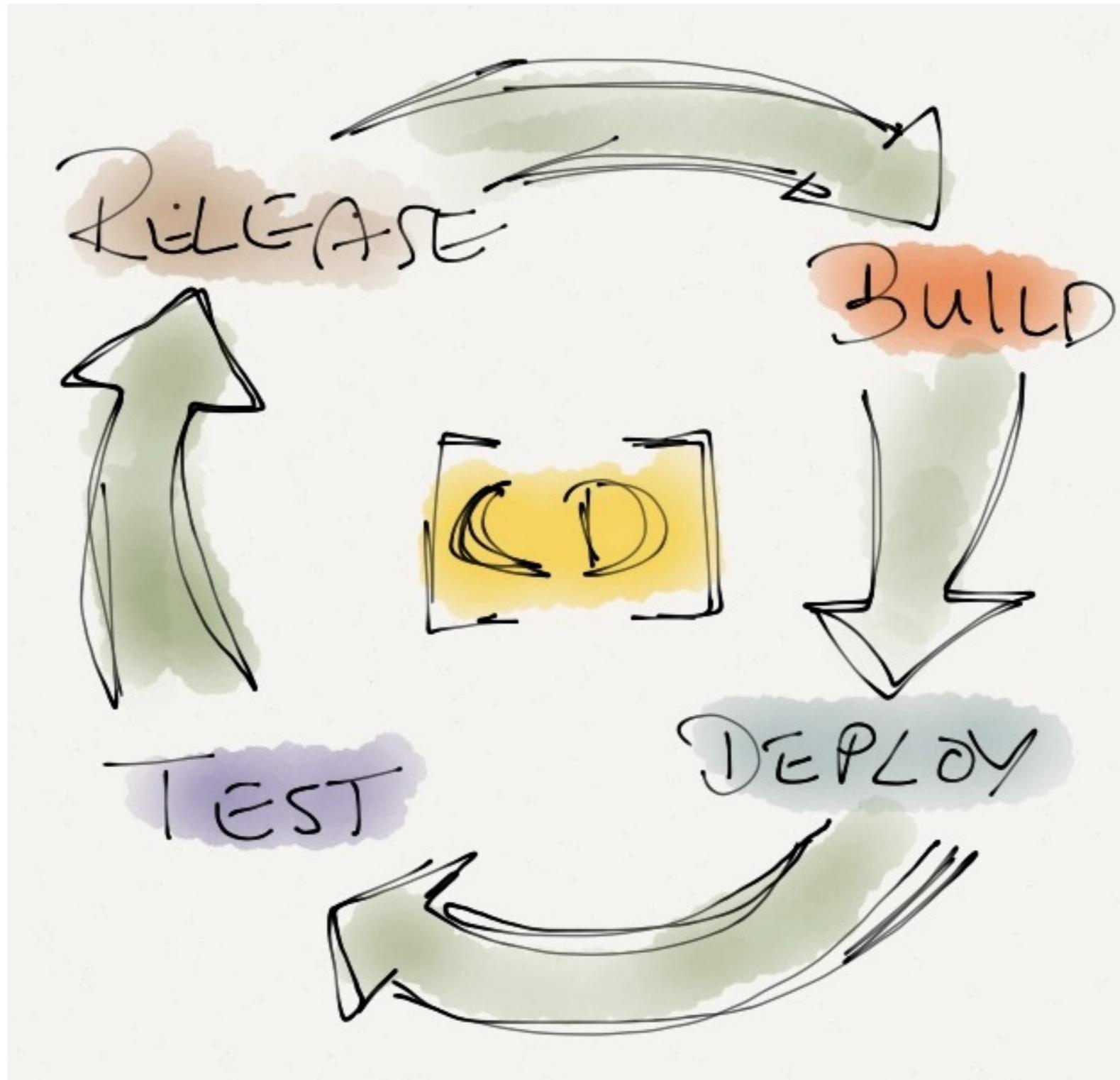
Strive for **fast feedback**



Continuous Integration



CD ?



CD ?

CONTINUOUS DELIVERY



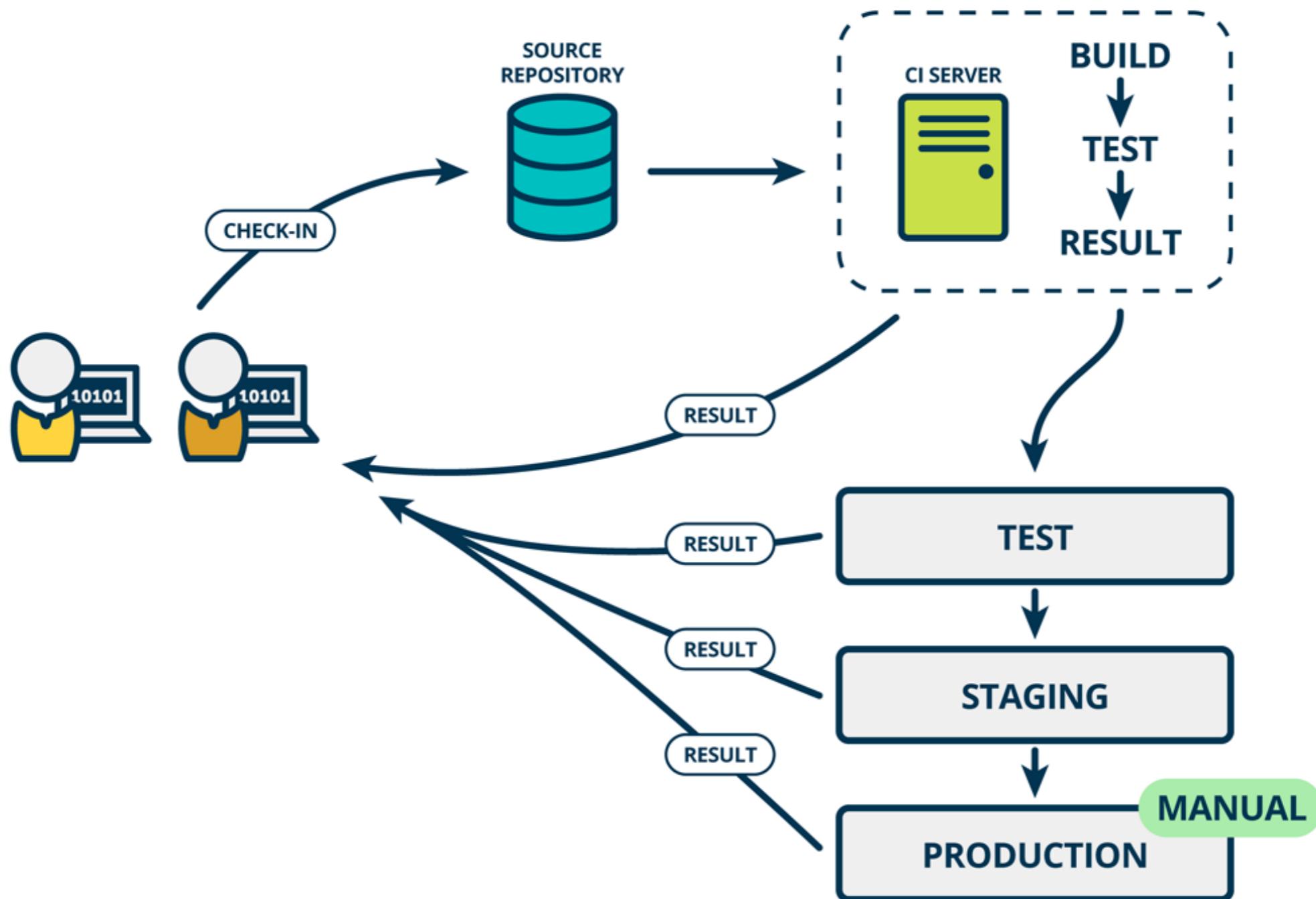
CONTINUOUS DEPLOYMENT



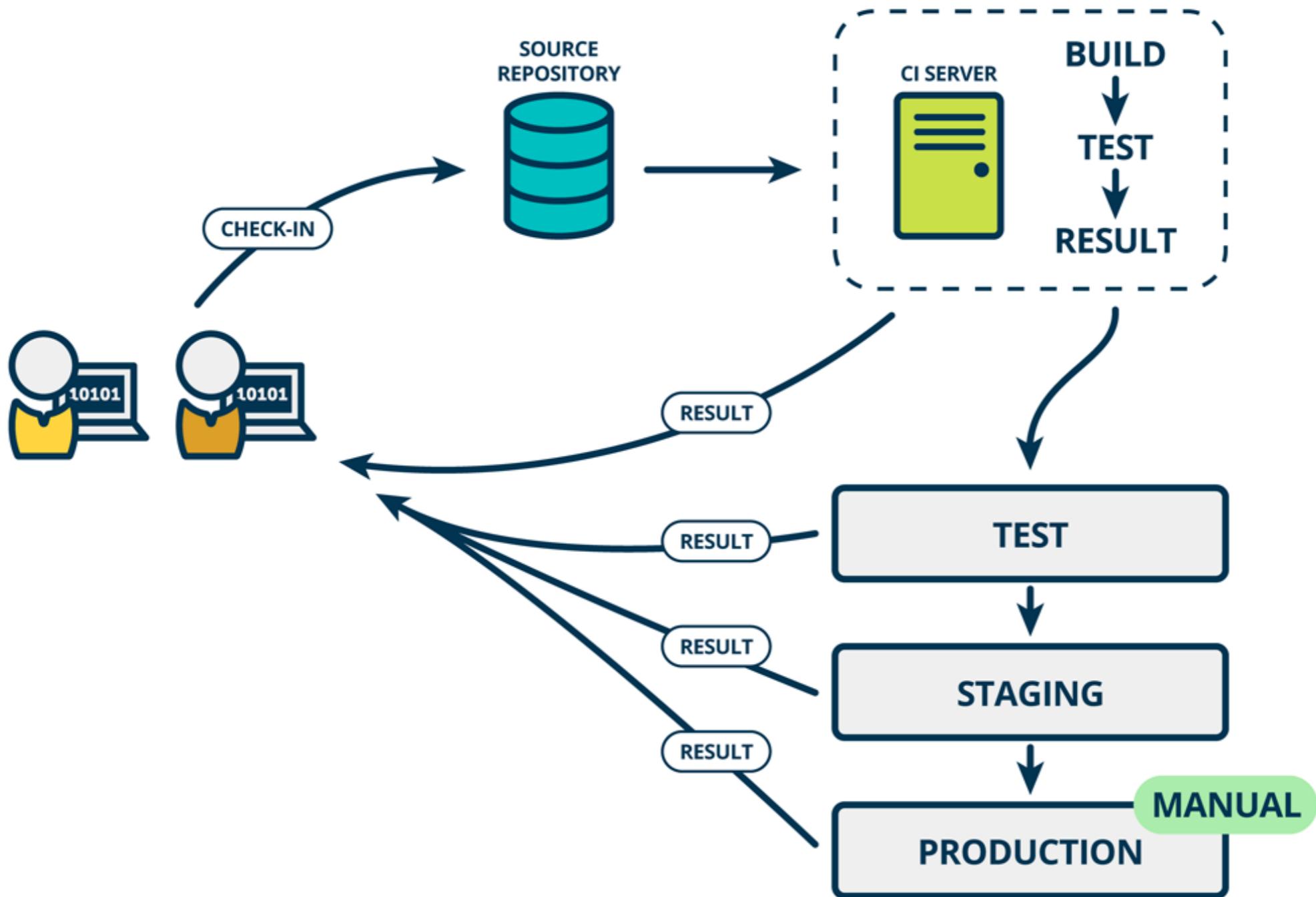
<http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>



Continuous Delivery



Rise of DevOps



Continuous Integration

is a Software development practices



Practice 1

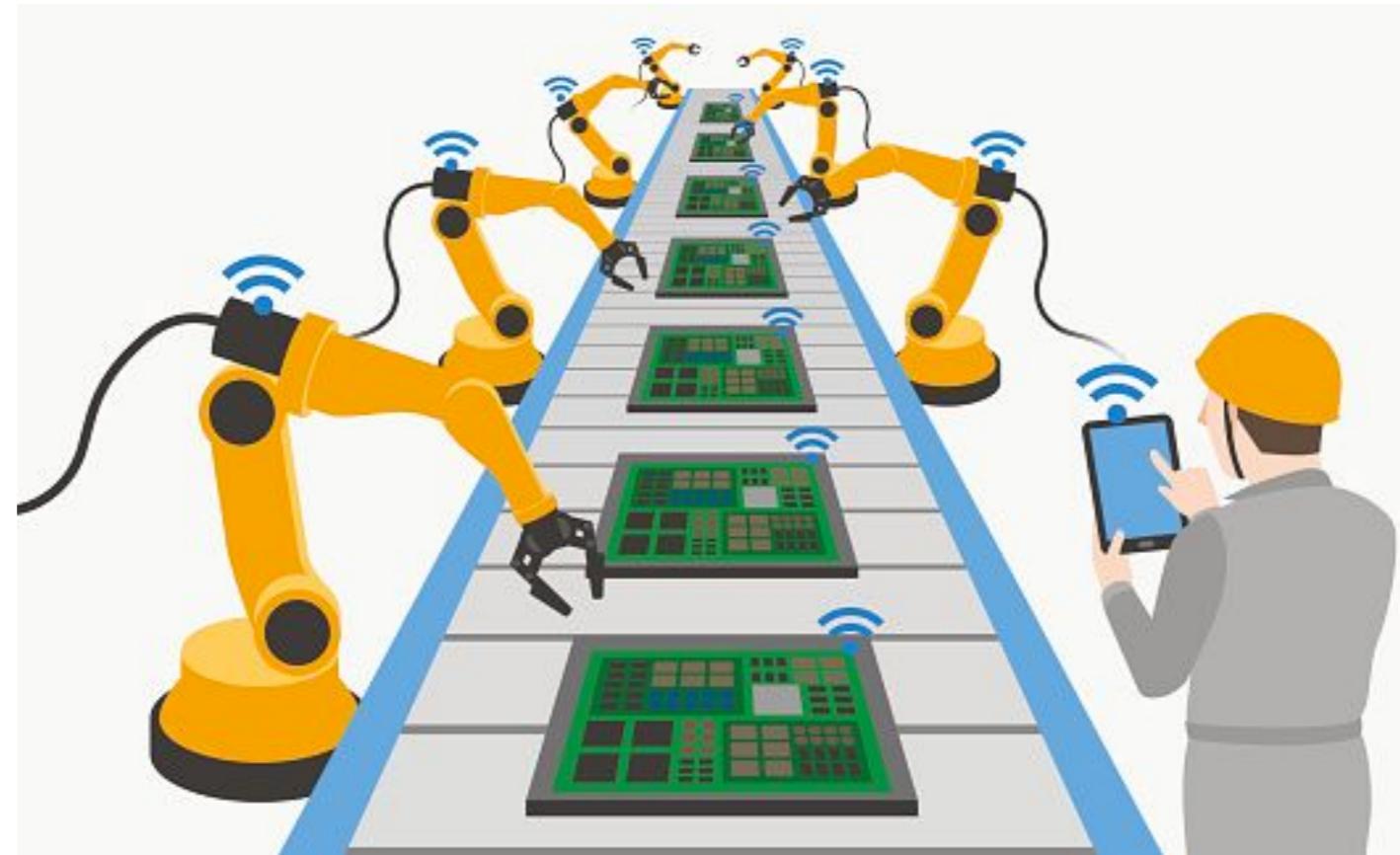
Maintain a single source repository

In general, you should store in source control
everything you need to build anything



Practice 2

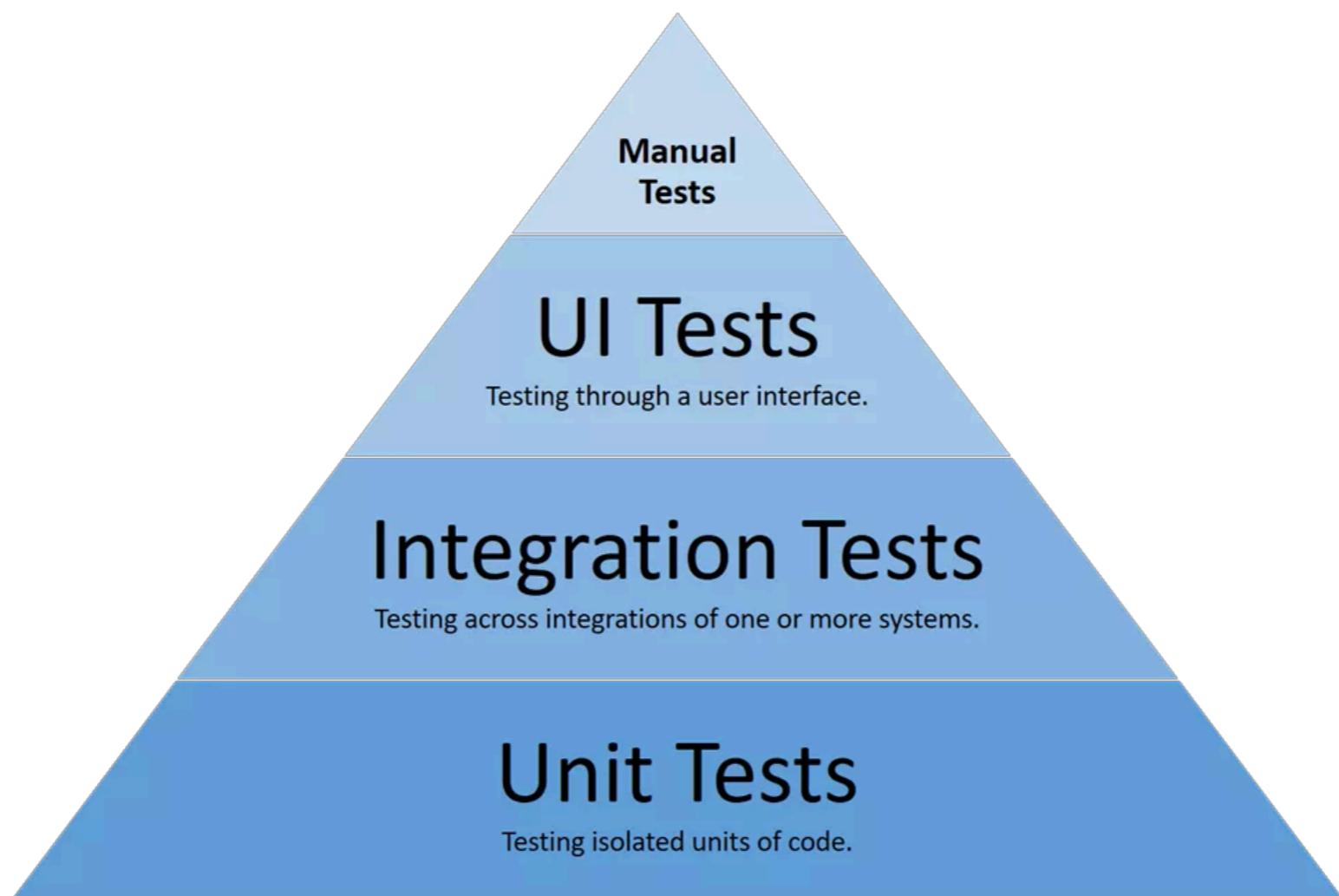
Automated the build
Automated environment for builds



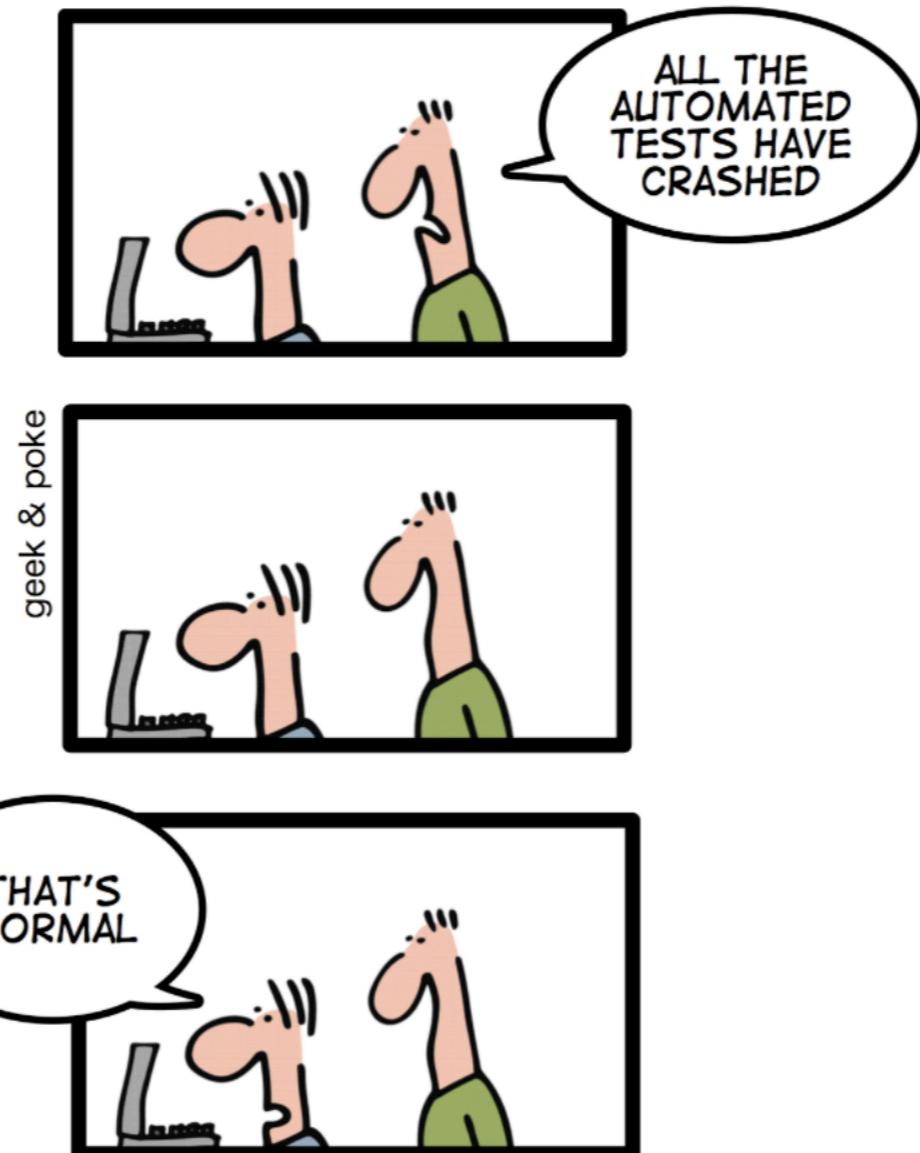
Practice 3

Make your build **self-testing**

Build process => compile, linking and **testing**



*TODAY: CONTINUOUS INTEGRATION
GIVES YOU THE COMFORTING
FEELING TO KNOW THAT
EVERYTHING IS NORMAL*

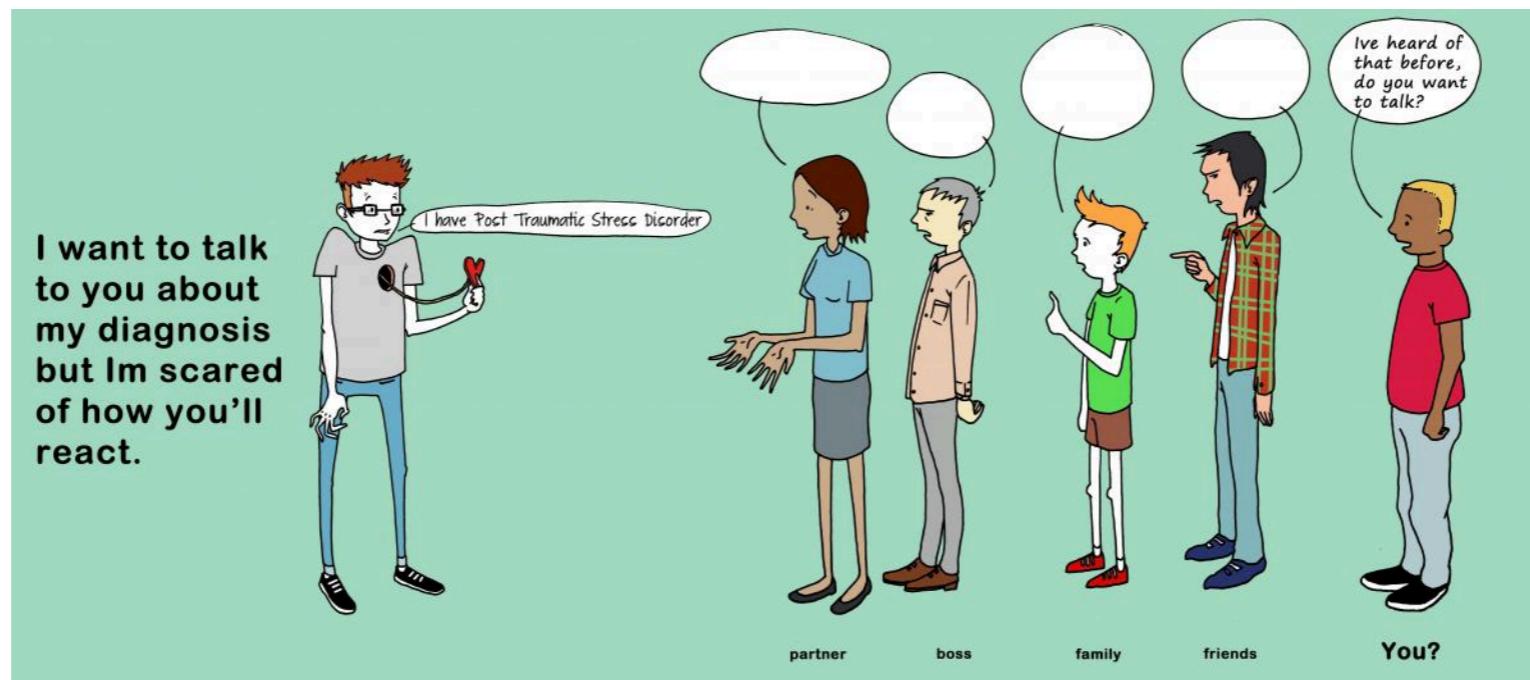


Practice 4

Everyone commits to the mainline everyday

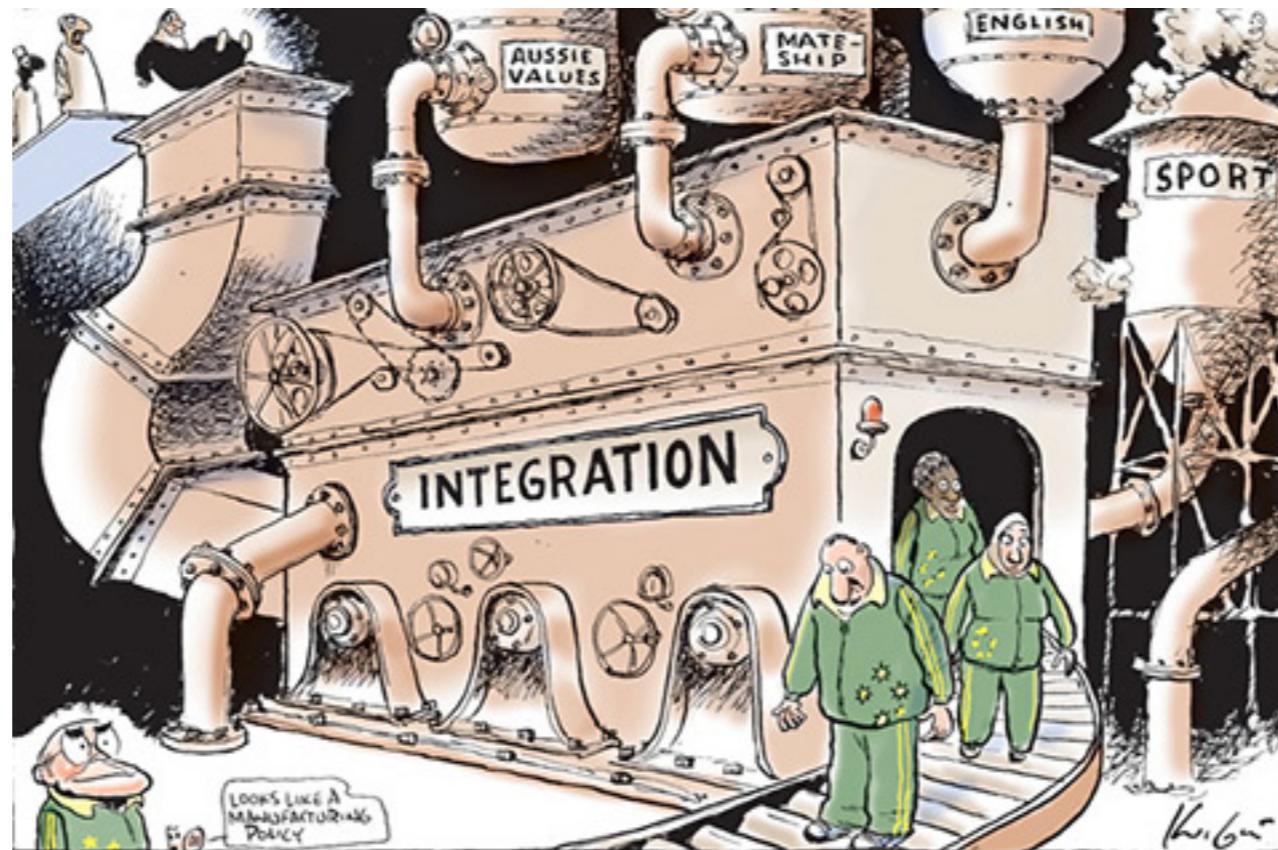
Integration is about communication

Integration allows developers to tell other developers



Practice 5

Every commits should build the mainline on an
Integration machine



Nightly build is not enough for Continuous Integration



Practice 6

Fix broken builds immediately

“Nobody has a higher priority task than fixing the build”



Practice 7

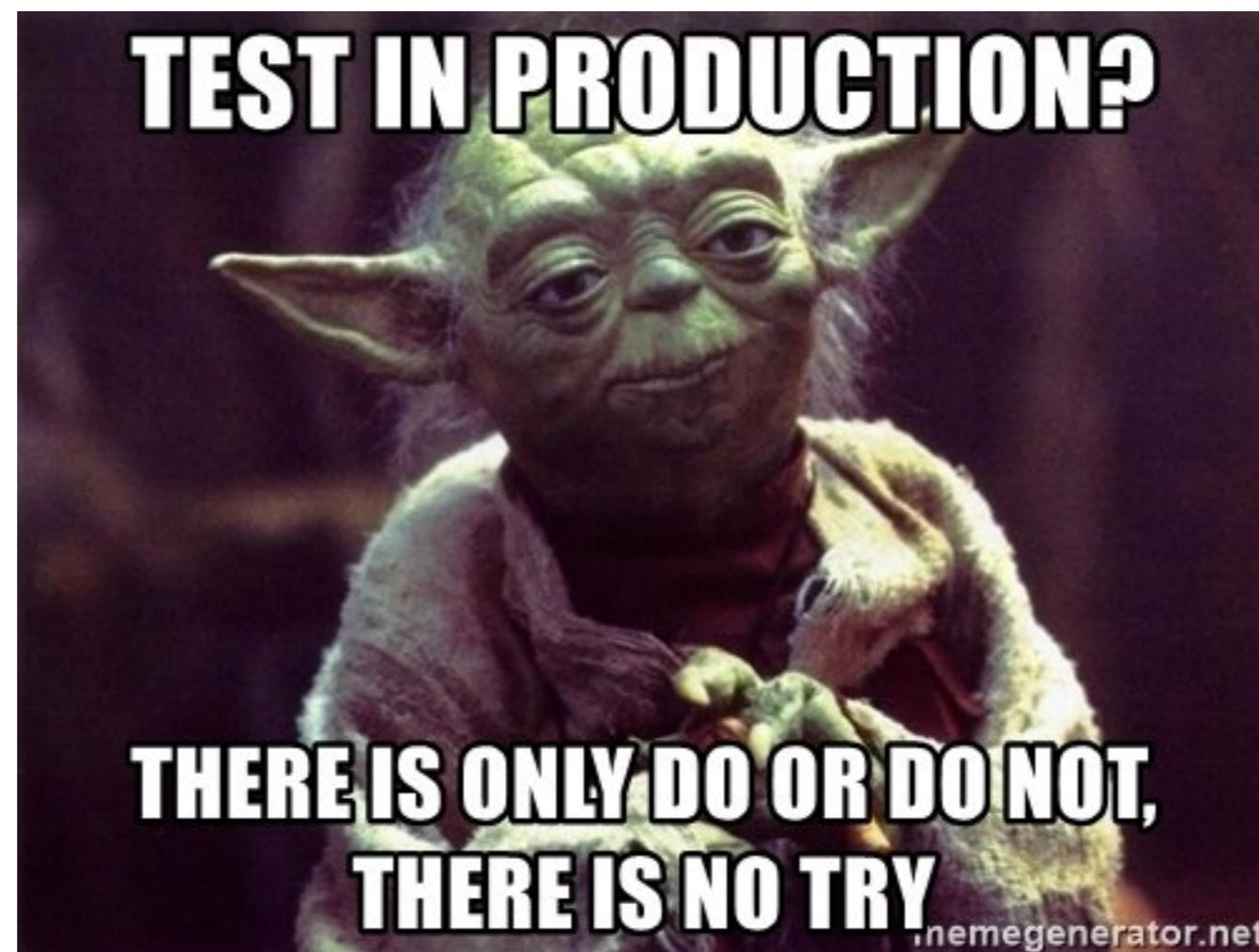
Keep the build **fast**

Continuous Integration is to provide rapid feedback



Practice 8

Test in clone of the **Production** environment



Practice 9

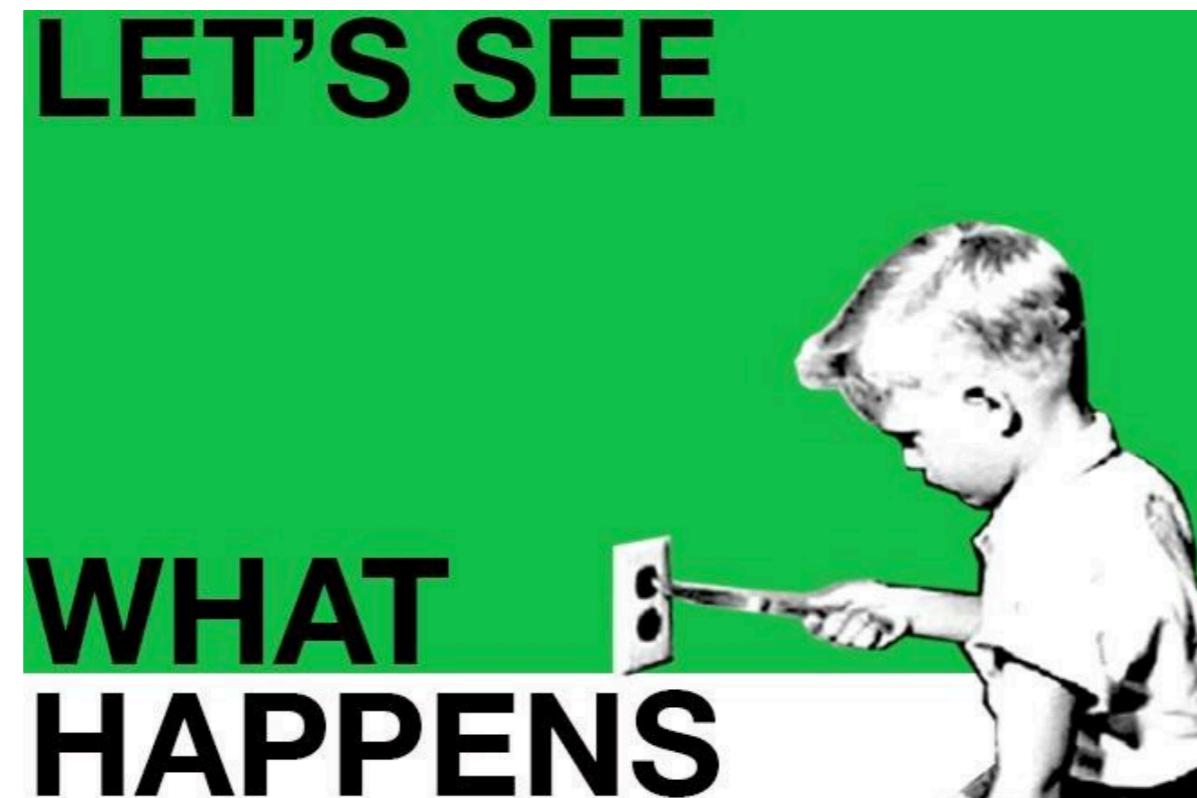
Make it easy for anyone to get
the latest executable

Make sure well known place where people can find



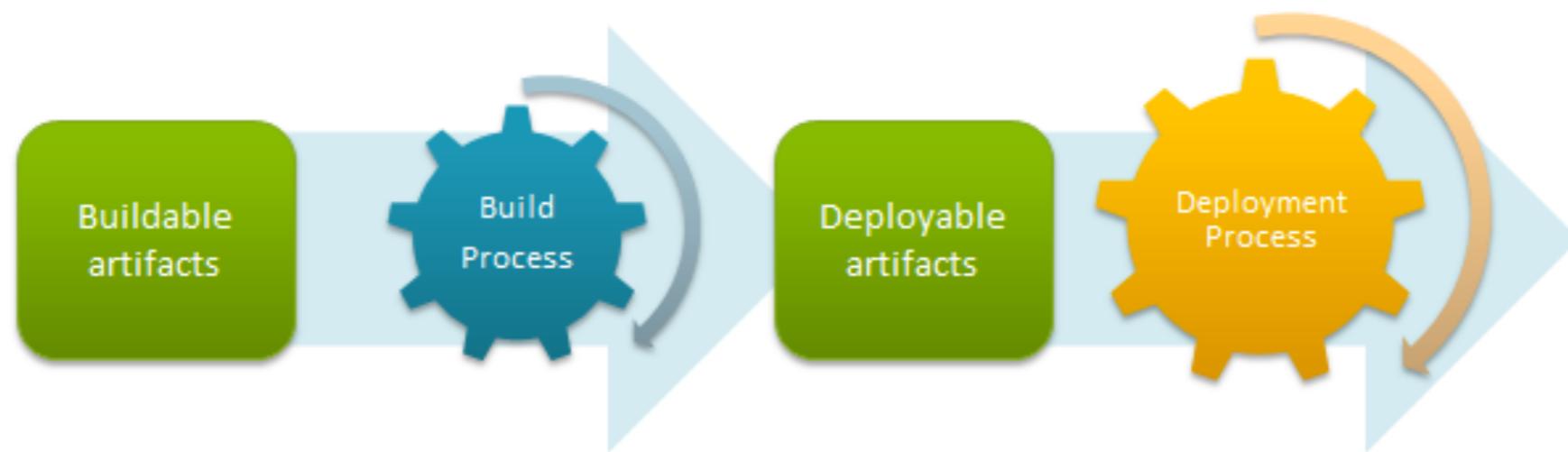
Practice 10

Everyone can see what's happening
Easier to see the state of the system and changes
Show the good information



Practice 11

Automated deployment



Continuous Delivery



Continuous Delivery

Use version control for all production artifacts

Automate your deployment process

Implement continuous integration (CI)

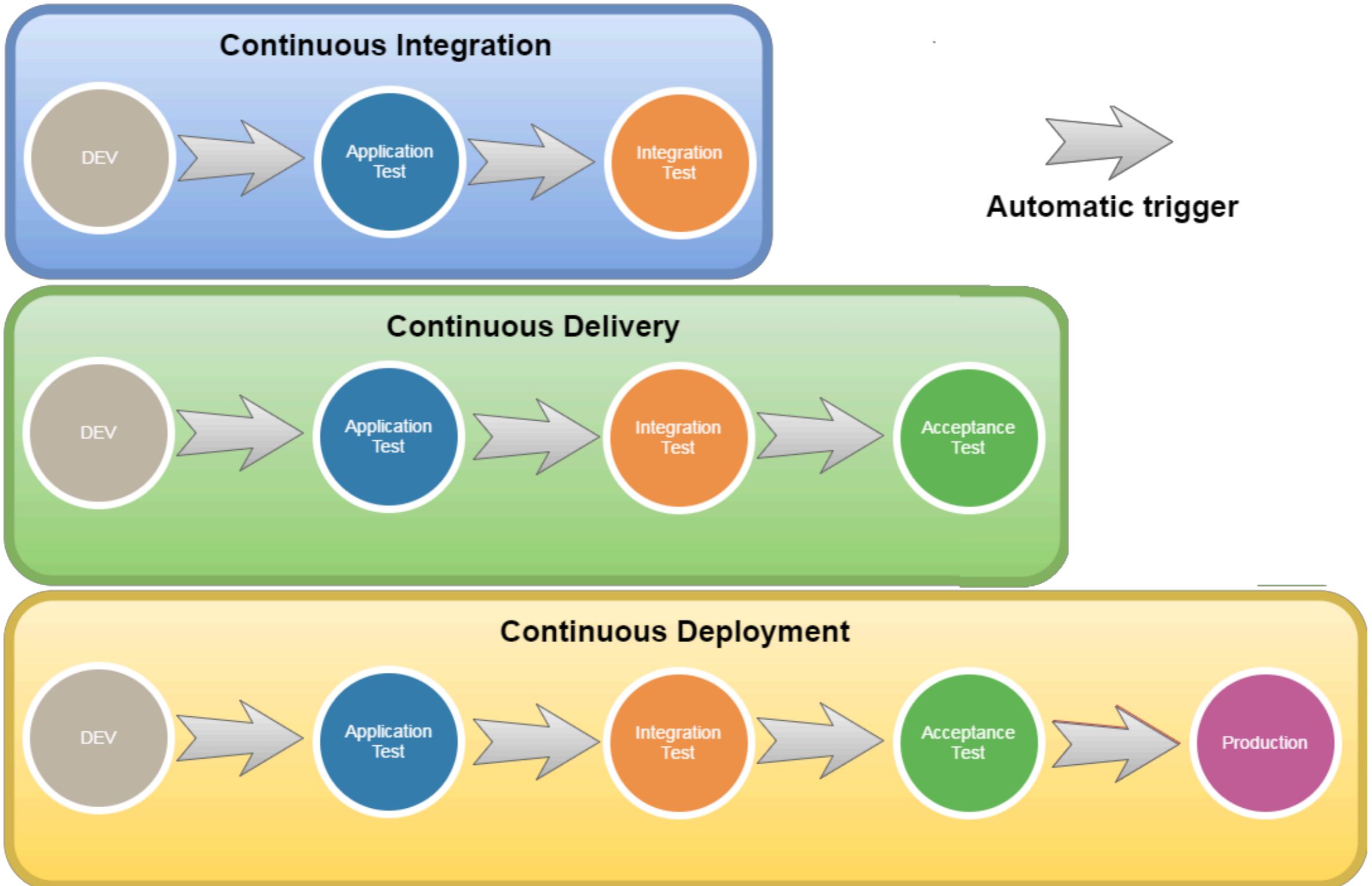
Use trunk-based development methods

Implement test automation

Support test data management

Integrate security into software development process





How to achieve the CI ?



1. Use good version control

Local
Centralize
Distributed



VSS = A brown pile of excrement with three wavy lines above it, representing VSS (Visual SourceSafe).

JUST SAY NO!



2. Choose Branch strategy

Main only

Development isolation

Feature isolation

Release isolation

Service and Release isolation

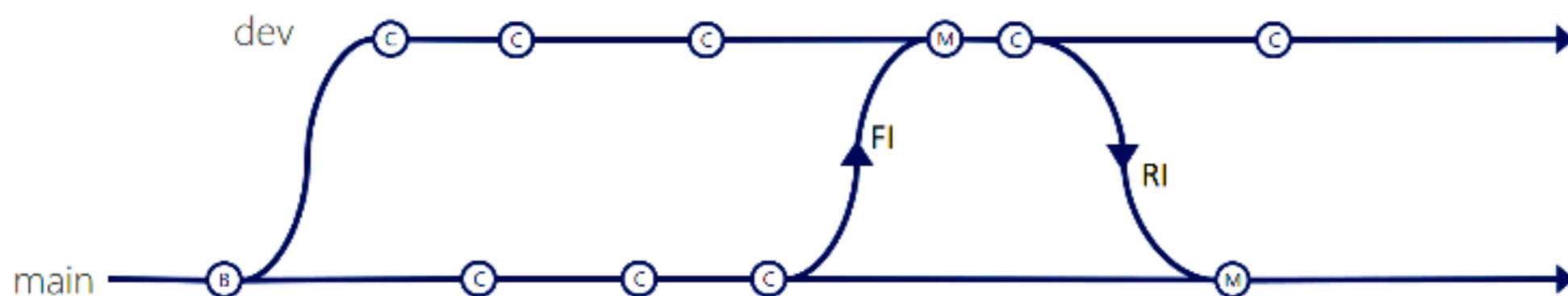
Service, Hotfix and Release isolation



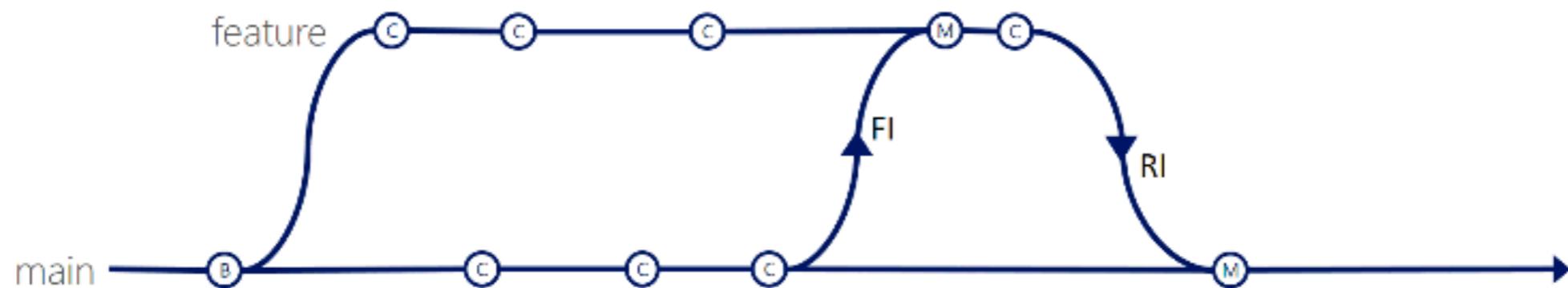
Main only



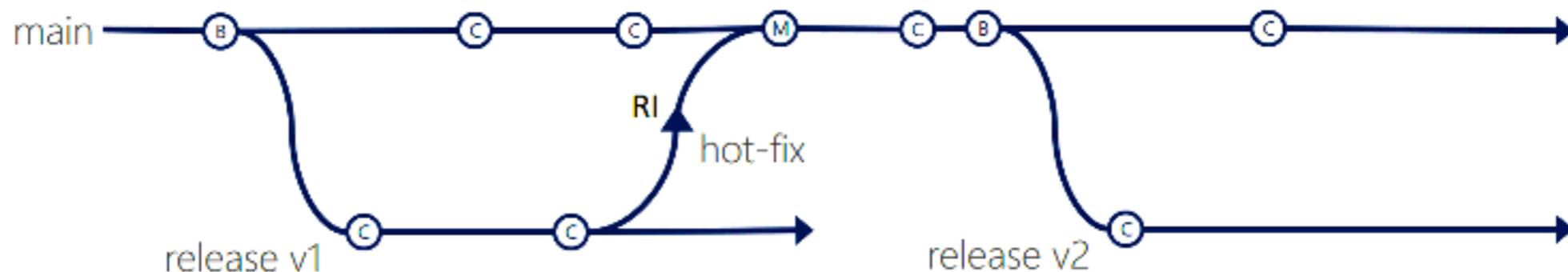
Development isolation



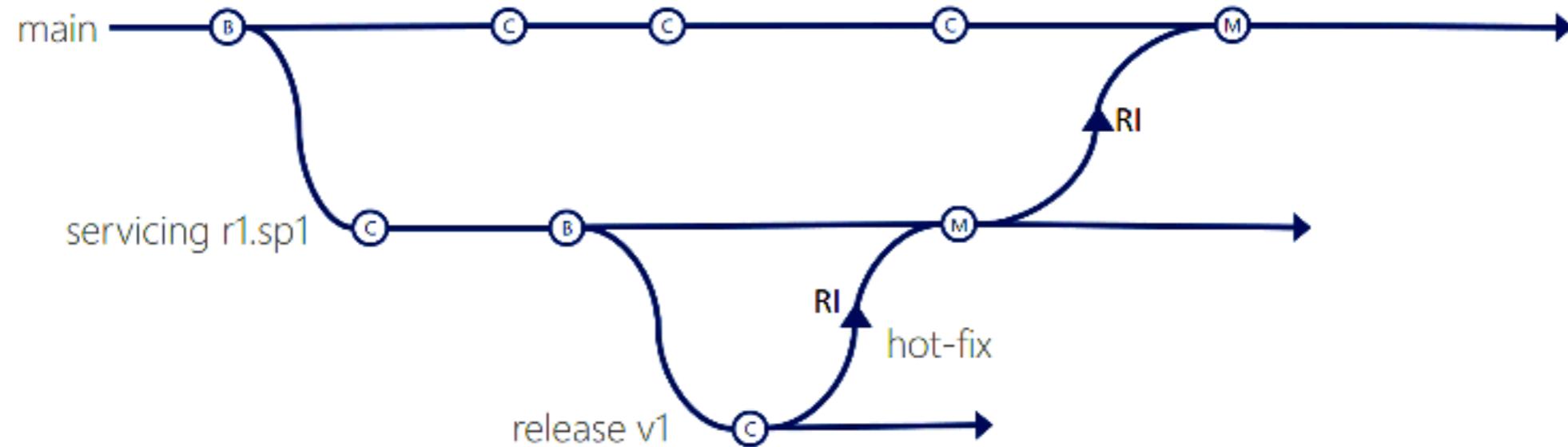
Feature isolation



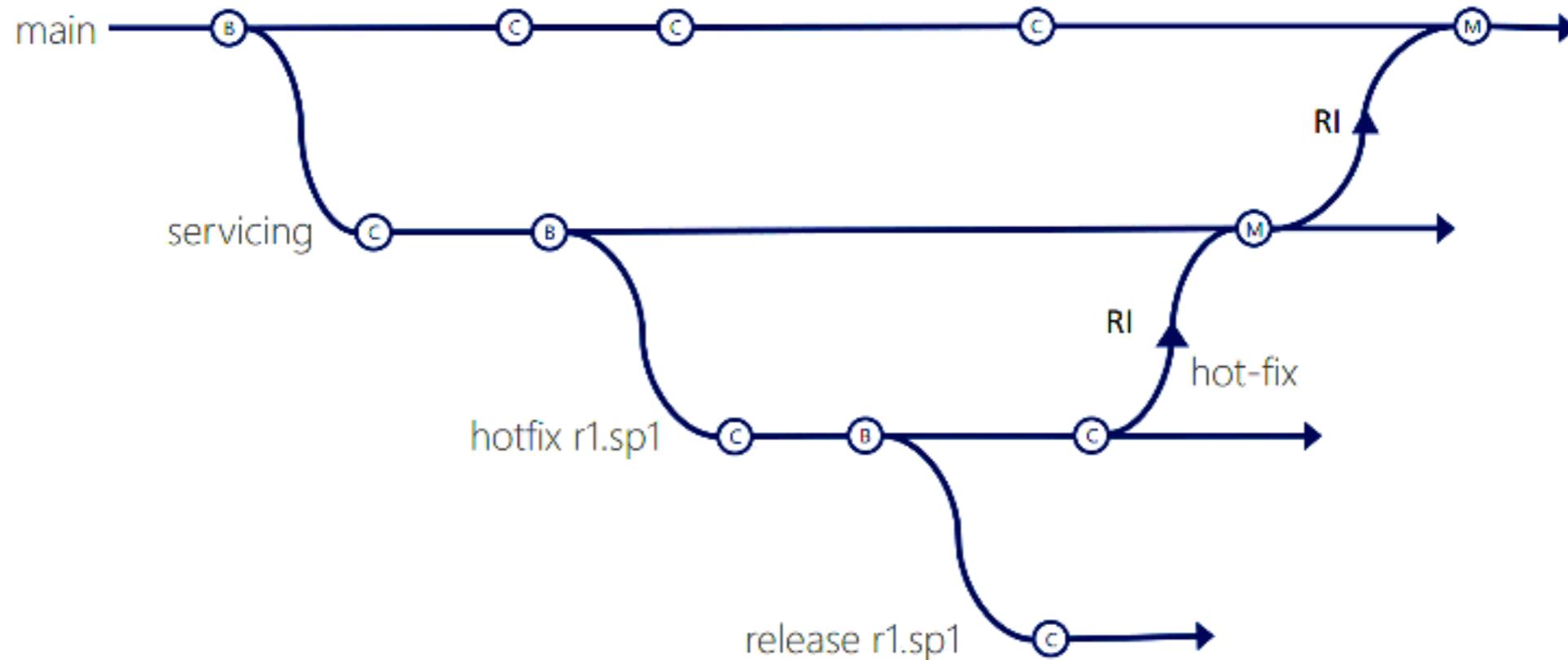
Release isolation



Service and Release isolation



Service, Hotfix, Release isolation



Validate, Validate and Validate

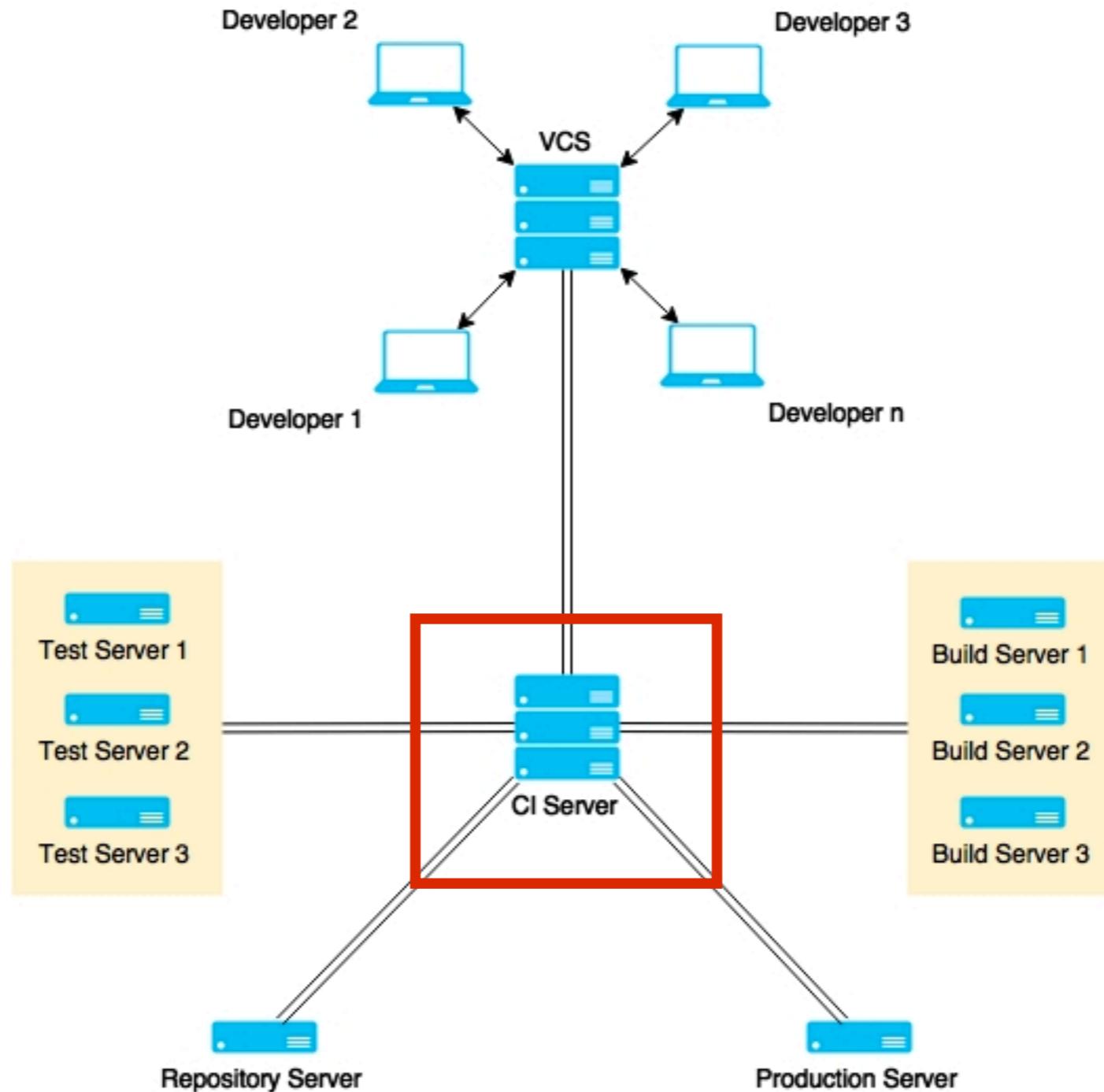


Suggestion

Keeping branches short-lived, merge conflicts are keep to as few as possible



3. Use good CI tool





Jenkins

Bamboo



TeamCity

> go™



Hudson



4. Use good build tool

- Javascript
 - Gulp, Grunt, Brocolli



- C#/.NET
 - Nant, MSBuild



- Java/JVM
 - Ant, Maven, Gradle, SBT, Leiningen



sbt gradle



More ...

Use static code analysis
Automated testing
Automated deployment
People discipline/habit



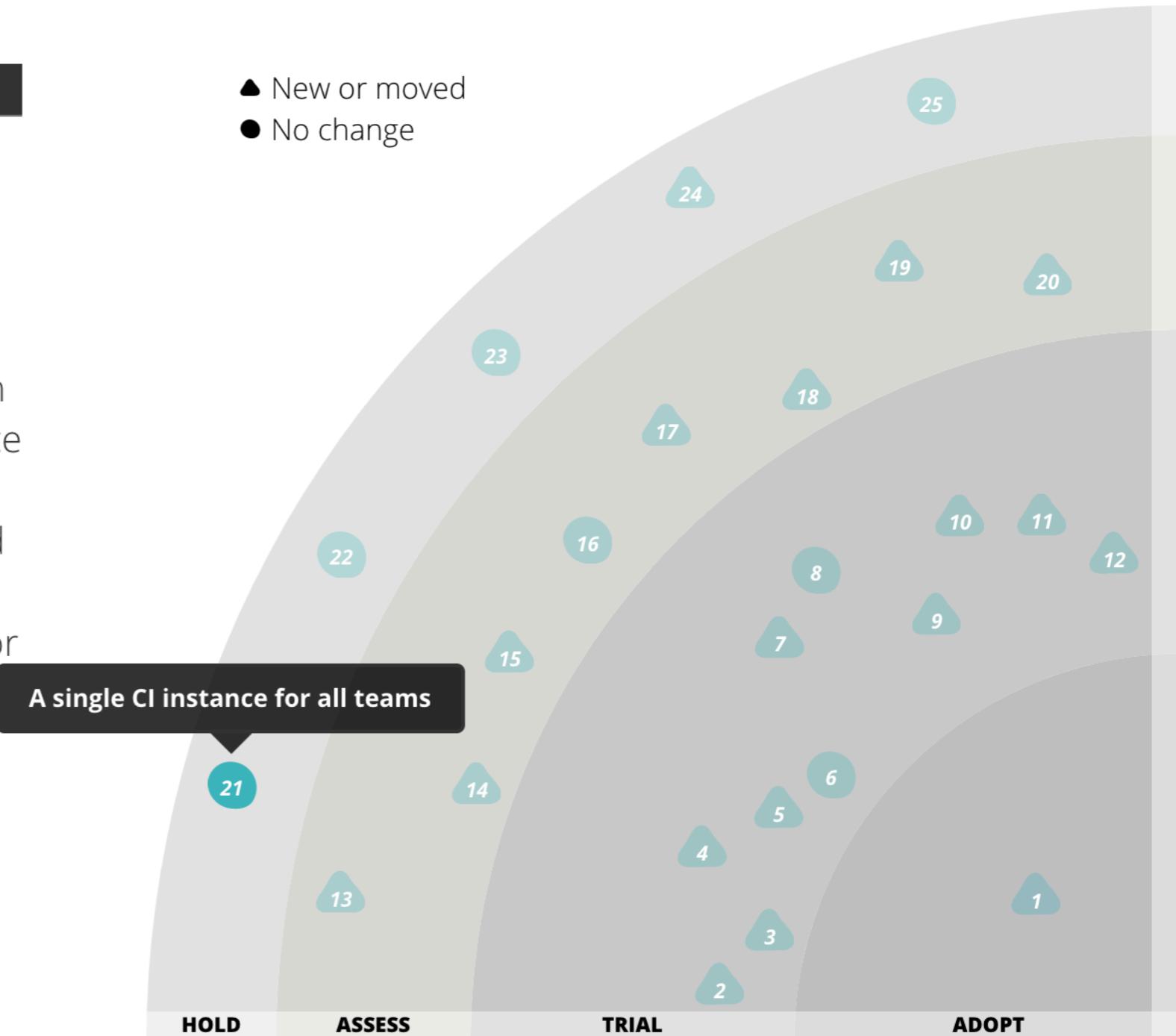
Anti-pattern for CI Server

● HOLD ?

21. A single CI instance for all teams

We're compelled to caution, again, against creating **a single CI instance for all teams**. While it's a nice idea in theory to consolidate and centralize Continuous Integration (CI) infrastructure, in reality we do not see enough maturity in the tools and products in this space to achieve the desired outcome. Software delivery teams which must use the centralized CI offering regularly have long delays depending on a central team to perform minor configuration tasks, or to troubleshoot problems in the shared infrastructure and tooling. At this stage, we continue to recommend that organizations limit their centralized investment to establishing patterns, guidelines and support for delivery teams to operate their own CI infrastructure.

- ▲ New or moved
- No change



<https://www.thoughtworks.com/radar/techniques>



Let's start with CI/CD



Application and framework to manage and monitor
the executable of **repeated tasks**



Jenkins

<https://jenkins.io/>



Centralize Continuous Integration Server



Jenkins

<https://jenkins.io/>



Why Jenkins ?

Easy !!
Extensible
Scalable
Opensource
Large community
Lot of plugins



Who use Jenkins ?

We thank the following organizations for their major commitments to support the Jenkins project.



Microsoft



redhat.

We thank the following organizations for their support of the Jenkins project through free and/or open source licensing programs.

Atlassian

Datadog

JFrog

Mac Cloud

PagerDuty

XMission

<https://jenkins.io/>



Hardware requirements

For Jenkins server

RAM +2GB

More CPU

More Disk



Installation



Jenkins in containers

Apache Tomcat
Jetty
JBoss
Websphere
WebLogic
Glassfish



Download Jenkins



The Jenkins website homepage. At the top is a dark navigation bar with links: Blog, Documentation, Plugins, Use-cases ▾, Participate, Sub-projects ▾, and Resources ▾. The main content area features the Jenkins logo on the left and the text "Jenkins" in large bold letters. Below it is the tagline "Build great things at any scale". A descriptive paragraph follows: "The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project." At the bottom of this section are two buttons: "Documentation" (white background) and "Download" (red background).

<https://jenkins.io/>



Use Long Term Support (LTS)

Getting started with Jenkins

The Jenkins project produces two release lines, LTS and weekly. Depending on your organization's needs, one may be preferred over the other.

Both release lines are distributed as `.war` files, native packages, installers, and Docker containers.

Long-term Support (LTS)

LTS (Long-Term Support) releases are chosen every 12 weeks from the stream of regular releases as the stable release for that time period. [Learn more...](#)

[Changelog](#) | [Upgrade Guide](#) | [Past Releases](#)

 [Deploy Jenkins 2.46.3](#)

 [Deploy to Azure](#)

 [Download Jenkins 2.46.3 for:](#)

Docker

FreeBSD

Weekly

A new release is produced weekly to deliver bug fixes and features to users and plugin developers.

[Changelog](#) | [Past Releases](#)

 [Download Jenkins 2.65 for:](#)

Arch Linux

Docker

FreeBSD

Gentoo



Start Jenkins

\$java -jar jenkins.war

Default port of server is **8080**

```
org.eclipse.jetty.server.AbstractConnector doStart  
ector@3e2fc448{HTTP/1.1, [http/1.1]}{0.0.0.0:8080}  
org.eclipse.jetty.server.Server doStart
```

```
winstone.Logger logInternal  
ngine v4.0 running: controlPort=disabled  
jenkins.InitReactorRunner$1 onAttained  
:ion  
jenkins.InitReactorRunner$1 onAttained  
jenkins.InitReactorRunner$1 onAttained
```



Change port of Jenkins

```
$java -jar jenkins.war --httpPort=<port>
```



Open in browser

<http://localhost:8080>

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/Users/somkiat/data/slide/ci-cd/swpark/software/keep/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue



Copy password from console

```
*****  
*****  
*****
```

Jenkins initial setup is required. An admin user has been created.

Please use the following password to proceed to installation:

a4b3a5231b8048419192d0c5afd3fce8

This may also be found at: /Users/somkiat/data/slide/ci-cd/swpi/initialAdminPassword

```
*****  
*****  
*****
```



Customize plugins

Getting Started



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.46.3



Waiting

Getting Started

Getting Started

<input type="radio"/> Folders Plugin	<input type="radio"/> OWASP Markup Formatter Plugin	<input type="radio"/> build timeout plugin	<input type="radio"/> Credentials Binding Plugin
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup Plugin	<input type="radio"/> Ant Plugin	<input type="radio"/> Gradle Plugin
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Organization Folder Plugin	<input type="radio"/> Pipeline: Stage View Plugin	<input type="radio"/> Git plugin
<input type="radio"/> Subversion Plug-in	<input type="radio"/> SSH Slaves plugin	<input type="radio"/> Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication plugin
<input type="radio"/> LDAP Plugin	<input type="radio"/> Email Extension Plugin	<input type="radio"/> Mailer Plugin	

** - required dependency

Jenkins 2.46.3



Success

Getting Started

Installation Failures

Some plugins failed to install properly, you may retry installing them or continue with

✓ Folders Plugin	✓ OWASP Markup Formatter Plugin	✓ build timeout plugin	✓ Credentials Binding Plugin
✓ Timestamper	✓ Workspace Cleanup Plugin	✓ Ant Plugin	✓ Gradle Plugin
✓ Pipeline	✓ GitHub Organization Folder Plugin	✓ Pipeline: Stage View Plugin	✓ Git plugin
✓ Subversion Plug-in	✓ SSH Slaves plugin	✓ Matrix Authorization Strategy Plugin	✓ PAM Authentication plugin
✓ LDAP Plugin	✓ Email Extension Plugin	✓ Mailer Plugin	

Jenkins 2.46.3

[Continue](#)

[Retry](#)



Create a new user

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.46.3

Continue as admin

Save and Finish



Ready to use

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

[Start using Jenkins](#)

Jenkins 2.46.3



Welcome to Jenkins

Jenkins

search [?](#) somkiat | log out

[ENABLE AUTO REFRESH](#)

New Item [add description](#)

People

Build History

Manage Jenkins

My Views

Credentials

Welcome to Jenkins!

Please [create new jobs](#) to get started.

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

Page generated: Jun 14, 2017 2:08:57 PM ICT [REST API](#) [Jenkins ver. 2.46.3](#)



About Jenkins's HOME

Default of **JENKINS_HOME** is
`<path of user>/jenkins`



About Jenkins's HOME

Data in JENKINS_HOME

```
/Users/somkiat/.jenkins
├── config.xml
├── failed-boot-attempts.txt
├── hudson.model.UpdateCenter.xml
├── jenkins.CLI.xml
├── jenkins.install.UpgradeWizard.state
├── jobs
├── logs
├── nodeMonitors.xml
├── nodes
├── plugins
├── queue.xml
├── queue.xml.bak
├── secret.key
├── secret.key.not-so-secret
├── secrets
├── updates
├── userContent
├── users
└── war
```



About Jenkins's HOME

File and Folder name	Description
config.xml	All about configuration
jobs	Keep all jobs/project
plugins	Keep all plugins
nodes	Keep all nodes
logs	Keep all logs



Change Jenkins's HOME

For Windows

```
set JENKINS_HOME=<your path>
```

For Linux/Mac

```
export JENKINS_HOME=<your path>
```

try to restart Jenkins ...



Disable Jenkins's security



Set useSecurity=false

<JENKINS HOME>/config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<hudson>
    <disabledAdministrativeMonitors/>
    <version>2.89.3</version>
    <numExecutors>2</numExecutors>
    <mode>NORMAL</mode>
    <useSecurity>false</useSecurity>
    <authorizationStrategy class="hudson.security.LoggedInAuthorizationStrategy">
        <denyAnonymousReadAccess>true</denyAnonymousReadAccess>
    </authorizationStrategy>
```



Learn to use Jenkins in the right way



Manage Jenkins

For Administrator to config anything in Jenkins

Jenkins

New Item

People

Build History

Manage Jenkins

Ivy views

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)
Configure the credential providers and types
- [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Configure System

Global setting and paths

Jenkins

New Item

People

Build History

Manage Jenkins

My Views

Credentials

New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

[Configure System](#)
Configure global settings and paths.

[Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.

[Configure Credentials](#)
Configure the credential providers and types

[Global Tool Configuration](#)
Configure tools, their locations and automatic installers.

[Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.

[Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

[System Information](#)
Displays various environmental information to assist trouble-shooting.

[System Log](#)
System log captures output from `java.util.logging` related to Jenkins.

[Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Configure System

JENKINS Home

of executors

Label name of node

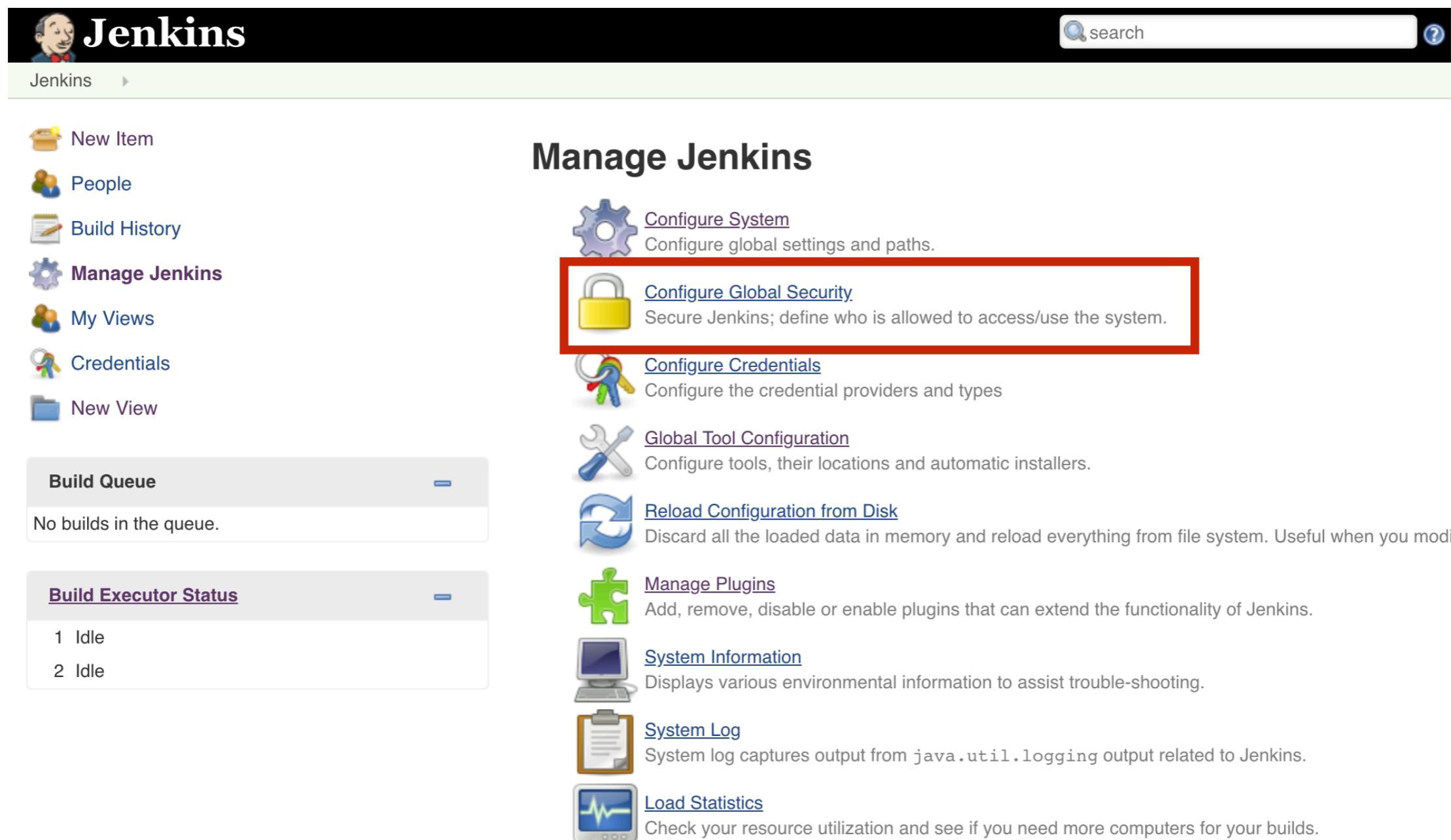
Environment variables

Email notification



Configure Global Security

Setting for secure Jenkins



The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), 'My Views', 'Credentials', and 'New View'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main content area is titled 'Manage Jenkins' and lists several configuration options: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon, highlighted with a red box), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench icon), 'Reload Configuration from Disk' (refresh icon), 'Manage Plugins' (puzzle piece icon), 'System Information' (monitor icon), 'System Log' (clipboard icon), and 'Load Statistics' (ECG icon). The 'Configure Global Security' item is described as 'Secure Jenkins; define who is allowed to access/use the system.'

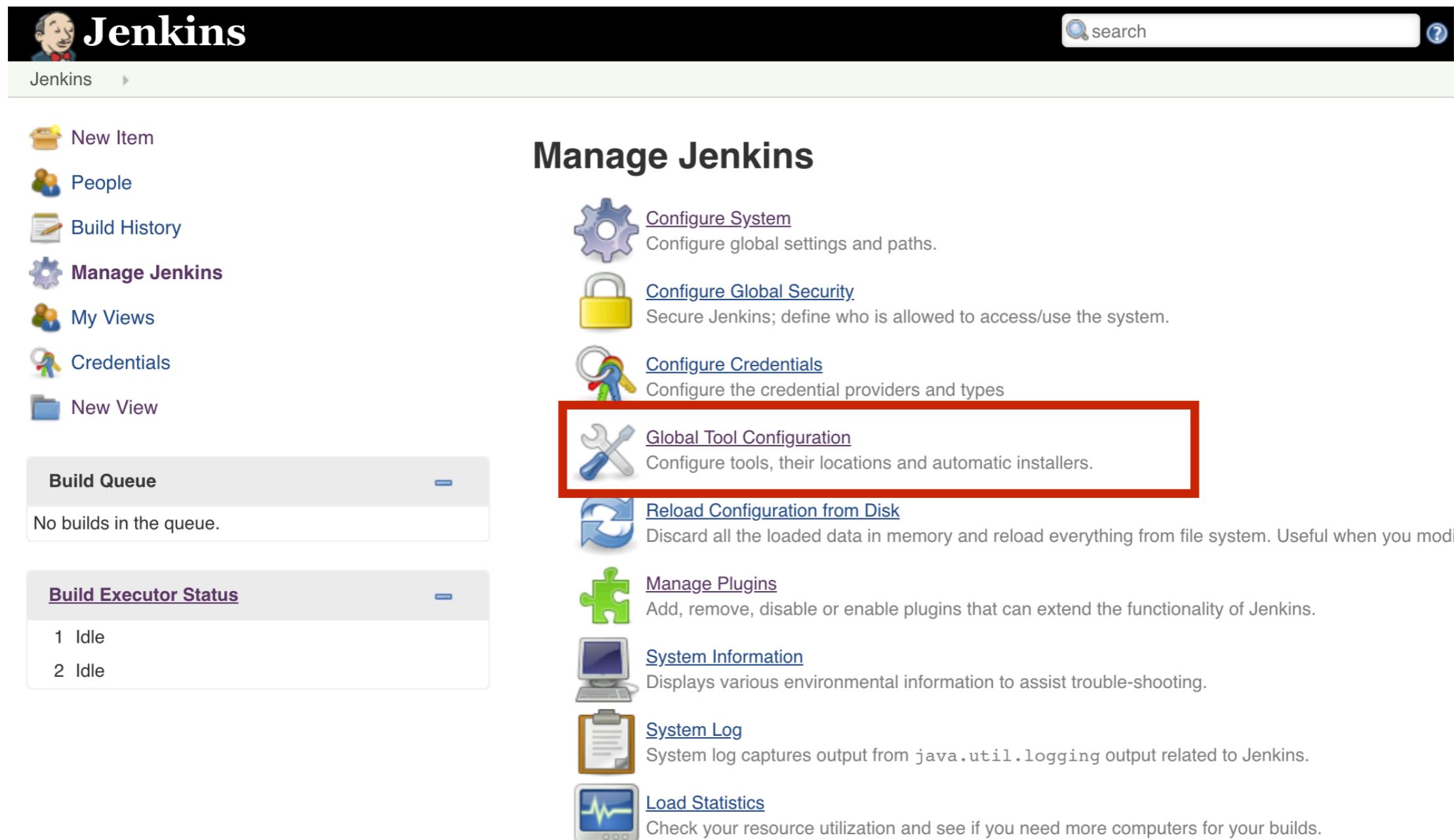
Manage Jenkins

-  [Configure System](#)
Configure global settings and paths.
-  [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
-  [Configure Credentials](#)
Configure the credential providers and types
-  [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
-  [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
-  [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  [System Information](#)
Displays various environmental information to assist trouble-shooting.
-  [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
-  [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Global Tool Configuration

Config tools, location and automatic installers



The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected and highlighted in purple), My Views, Credentials, and New View. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration (which is highlighted with a red box), Reload Configuration from Disk, Manage Plugins, System Information, System Log, and Load Statistics.

Manage Jenkins

-  [Configure System](#)
Configure global settings and paths.
-  [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
-  [Configure Credentials](#)
Configure the credential providers and types
-  [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
-  [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modify configuration files.
-  [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  [System Information](#)
Displays various environmental information to assist trouble-shooting.
-  [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
-  [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Global Tool Configuration

Apache Maven
JDK (Java Development Kit)
Git
Gradle
Docker



Manage Plugins

Add, remove, enable/disable plugins

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected and highlighted in purple), My Views, Credentials, and New View. Below that are sections for Build Queue (empty) and Build Executor Status (2 Idle). The main content area is titled "Manage Jenkins" and lists several configuration options: Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, Manage Plugins (which is highlighted with a red box), System Information, System Log, and Load Statistics.

Manage Jenkins

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Credentials](#)
Configure the credential providers and types
- [Global Tool Configuration](#)
Configure tools, their locations and automatic installers.
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modi
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from `java.util.logging` related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.



Manage Plugins

Add, remove, enable/disable plugins

				Filter: <input type="text"/>
Updates	Available	Installed	Advanced	
Install	Name ↓	Version	Installed	
No updates				

Update information obtained: 14 hr ago

[Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



Manage Plugins

Filter plugins

Filter:

Updates Available Installed Advanced

Install	Name ↓	Version	Installed
		No updates	

Update information obtained: 14 hr ago [Check now](#)

Select: [All](#), [None](#)

This page lists updates to the plugins you currently use.



Finds Jenkins's plugin

Plugins Index

Discover the 1000+ community contributed Jenkins plugins to support building, deploying and automating any project.

Browse ▶ Find plugins...

<https://plugins.jenkins.io/>



Try to install a first plugin

Choose Available tab and select a plugin

Filter:

Updates Available Installed Advanced

Install ↓

Name	Version
CCM Plug-in <input type="checkbox"/> This plug-in generates the trend report for CCM, an open source static code analysis program.	3.1
FxCop Runner plugin <input type="checkbox"/>	1.1
MSBuild Plugin <input type="checkbox"/>	1.27
MSTest plugin <input type="checkbox"/> Generates test reports for MSTest.	0.19
MSTestRunner plugin <input type="checkbox"/>	1.3.0
NAnt Plugin <input type="checkbox"/>	1.4.3
NCover plugin <input type="checkbox"/>	0.3
PowerShell plugin <input type="checkbox"/>	1.3
Violation Comments to Bitbucket Server Plugin <input type="checkbox"/> Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.	1.50
Violations plugin <input type="checkbox"/>	0.7.11

[Install without restart](#) [Download now and install after restart](#)

Update information obtained: 9 hr 37 min ago [Check now](#)



Manage Nodes

Add, remove, status of nodes

Jenkins

Nodes

ENABLE AUTO REFRESH

Back to Dashboard

Manage Jenkins

New Node

Configure

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

search

S Name ↓ Architecture Clock Difference Free Disk Space Free Swap Space Free Temp Space Response Time

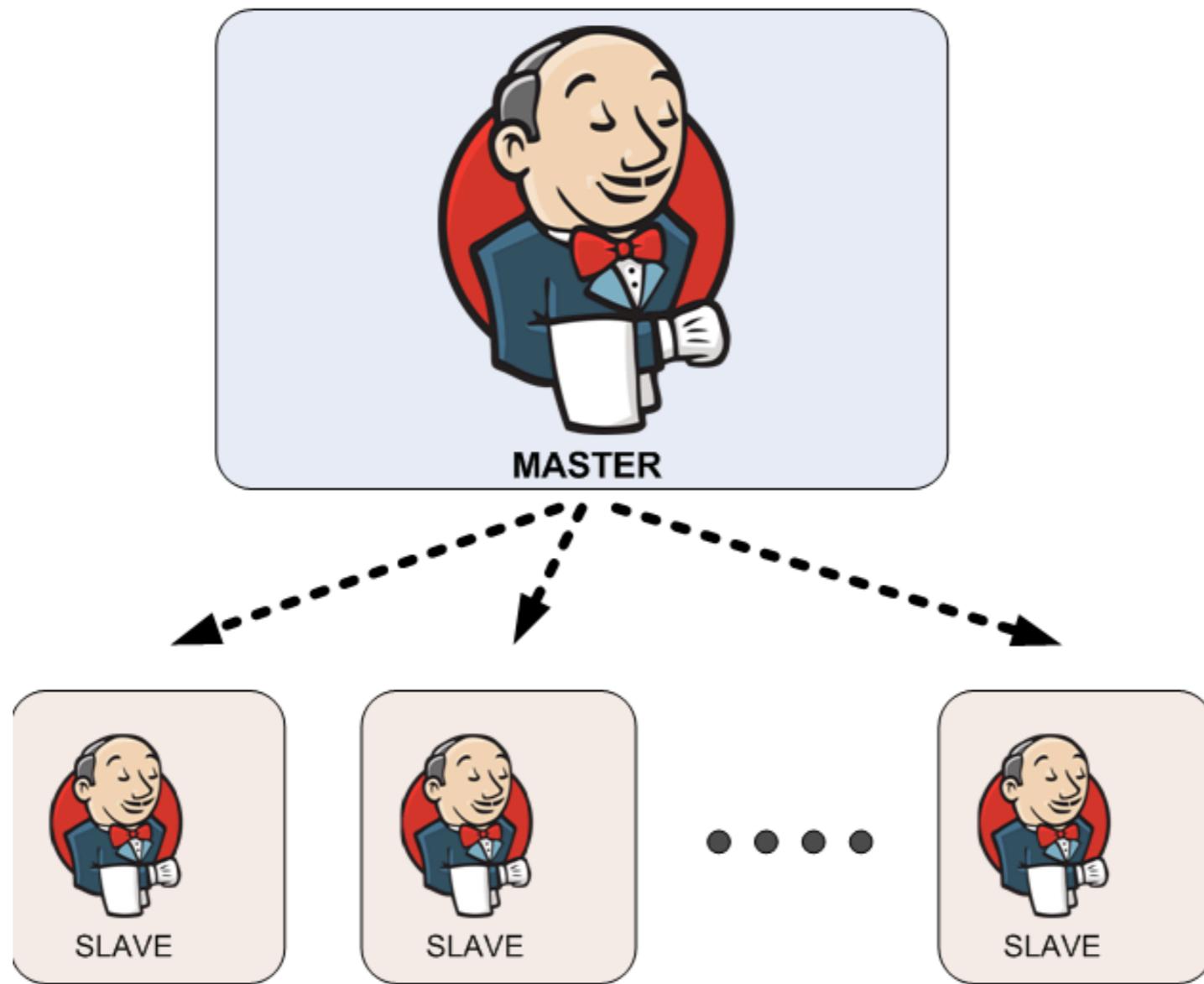
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Mac OS X (x86_64)	In sync	5.73 GB	463.00 MB	5.73 GB	0ms
	Data obtained	36 min	36 min	36 min	36 min	36 min	36 min

Refresh status



Manage Nodes

Master-slave concept to scale Jenkins



Workshop

