สยามชำนาญกิจ
SIAM CHAMNAN KIT

Service
Develop to Deploy

Important Quality Services

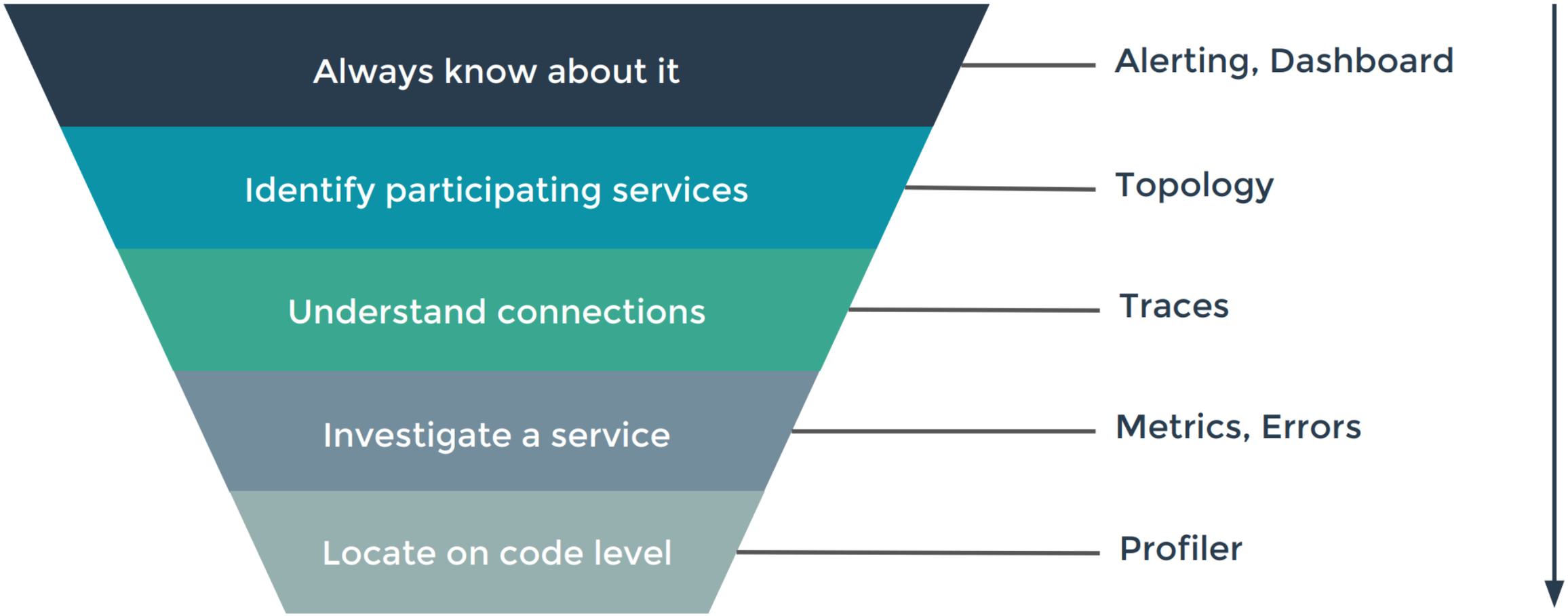
Important Quality Services

Observability
Configurability
Security

How to find an issue ?

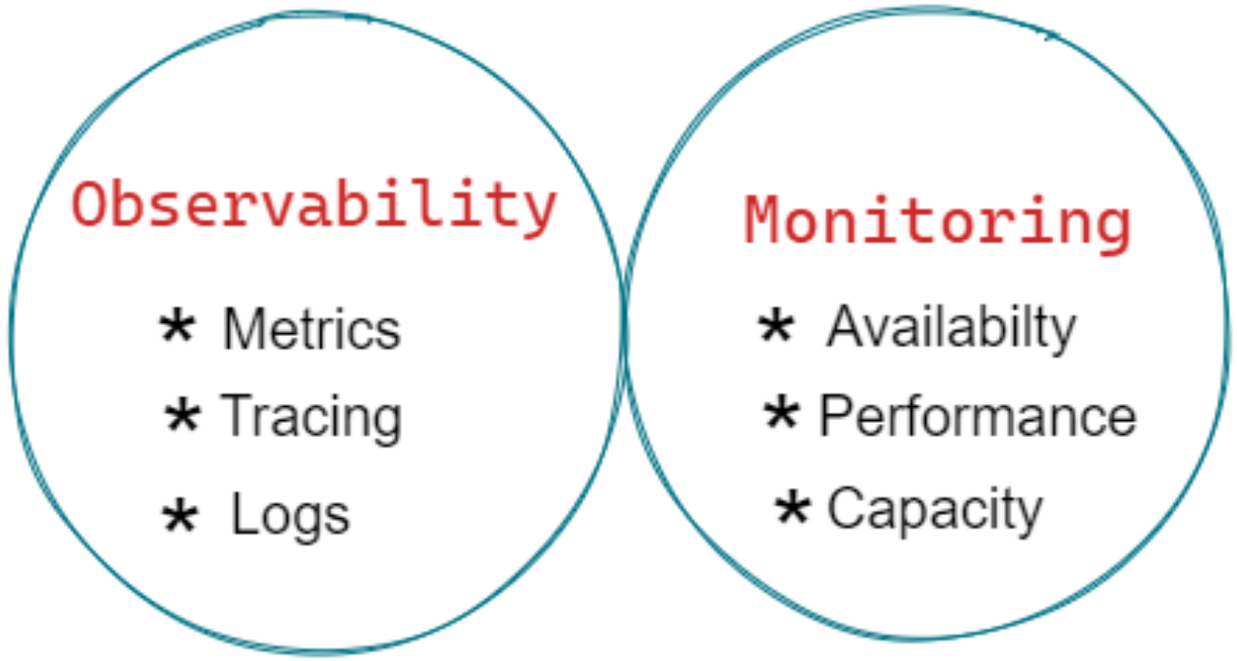
Reactive Vs. Proactive

How to find an issue ?



Observable services

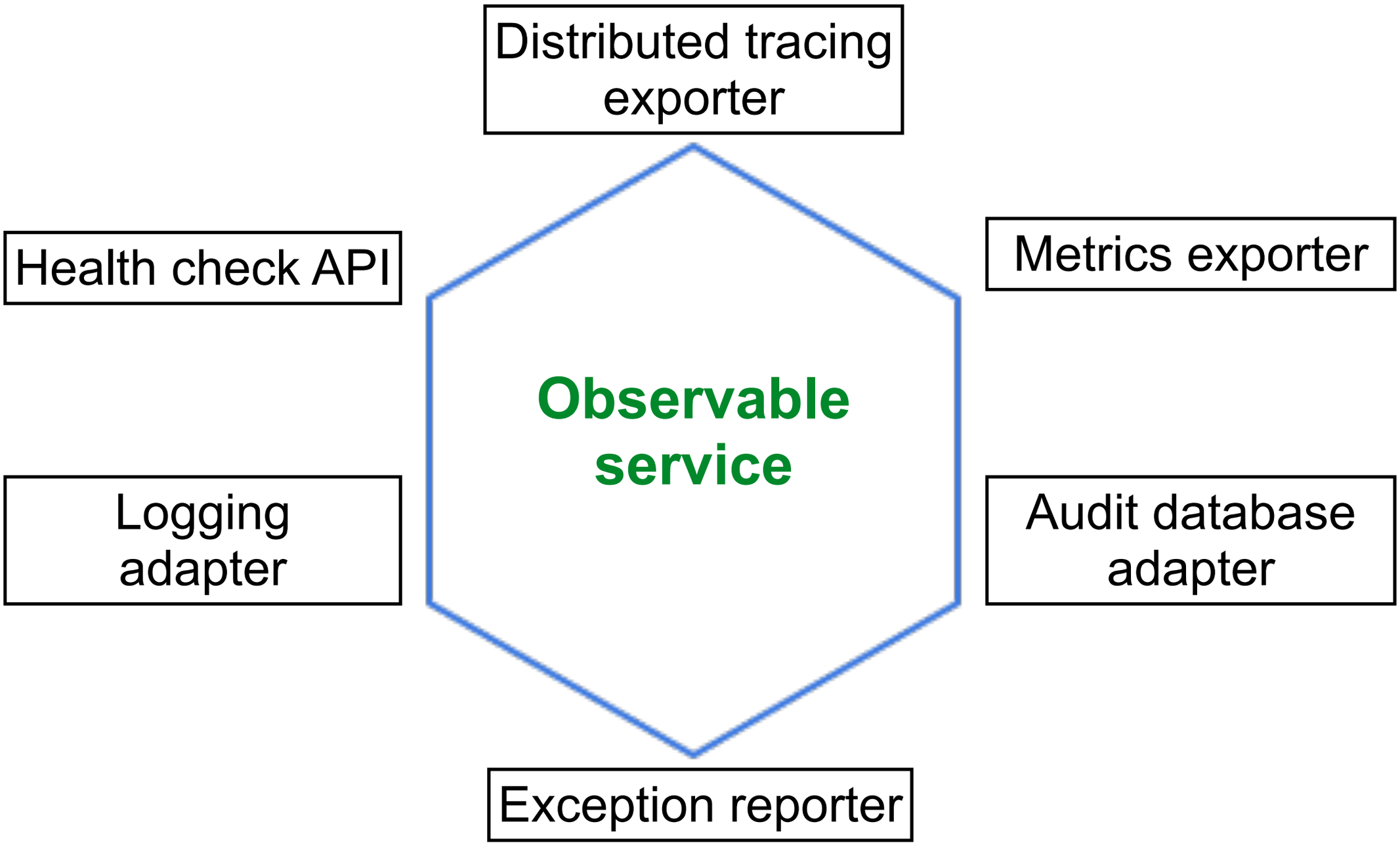
Observability vs Monitoring



Design observable services

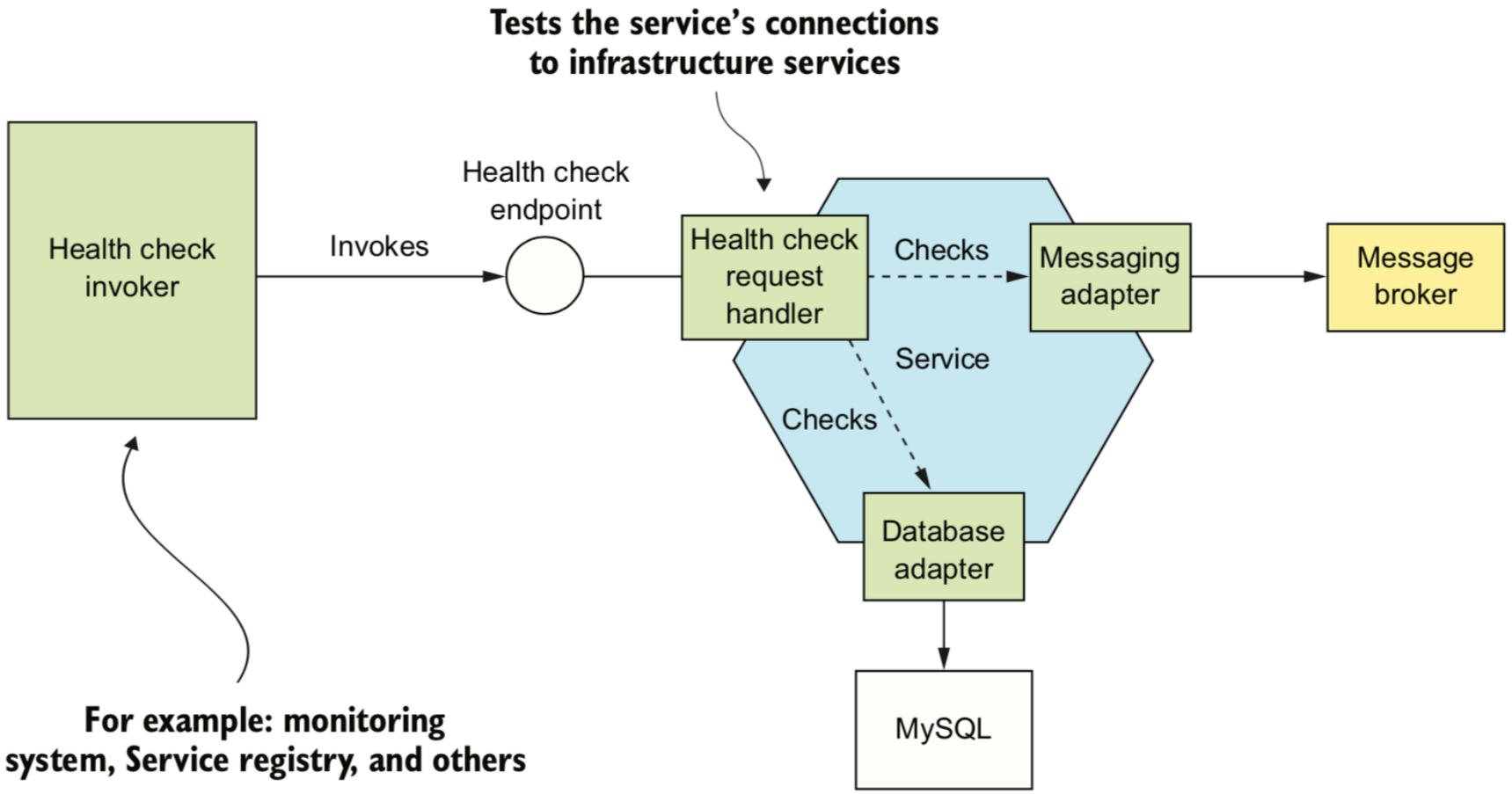
Health check API
Log aggregation
Distributed tracing
Exception tracking
Application metrics
Audit logging

Observable services



Health check API

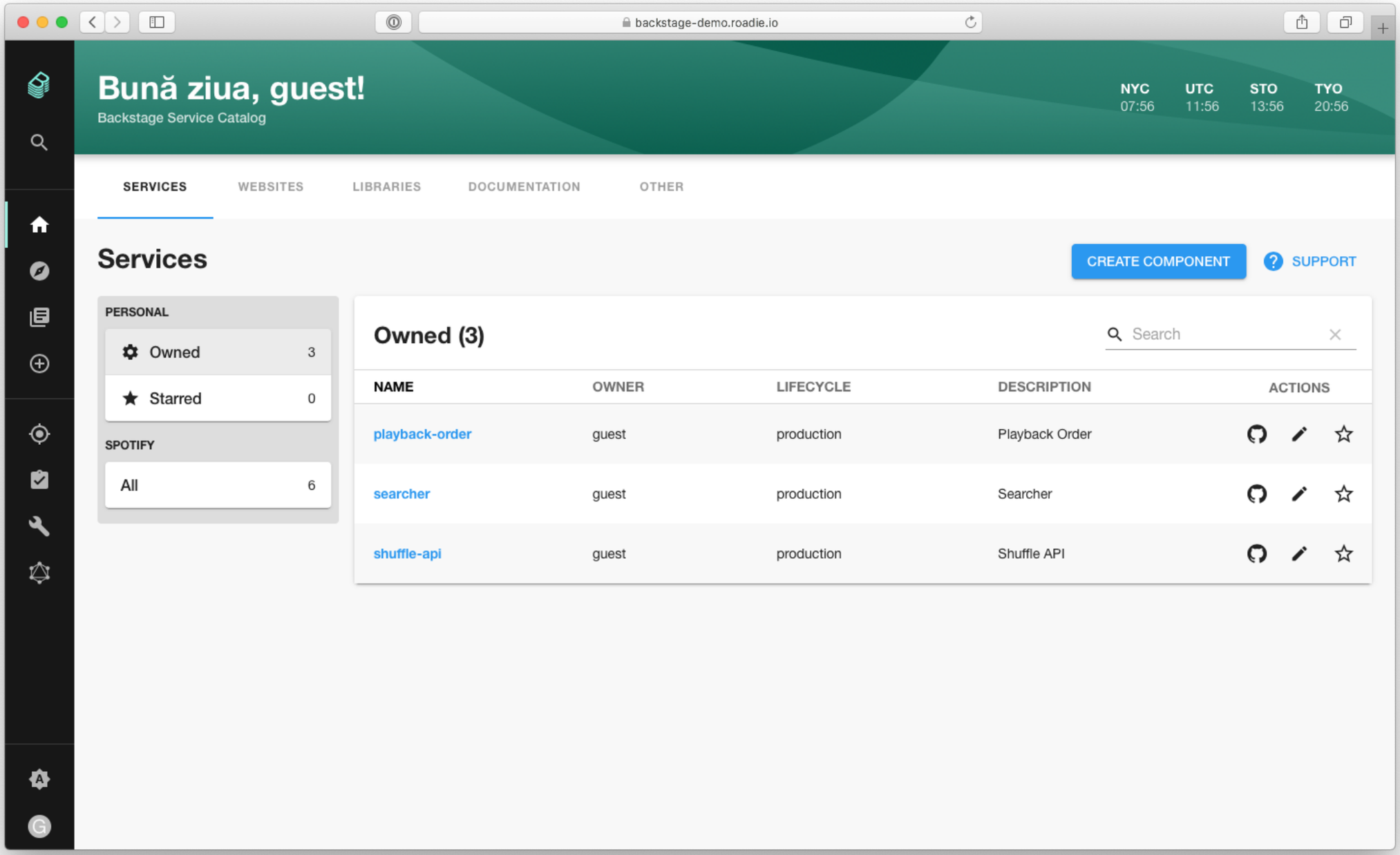
Expose an endpoint that return the health of service



Monitor your services

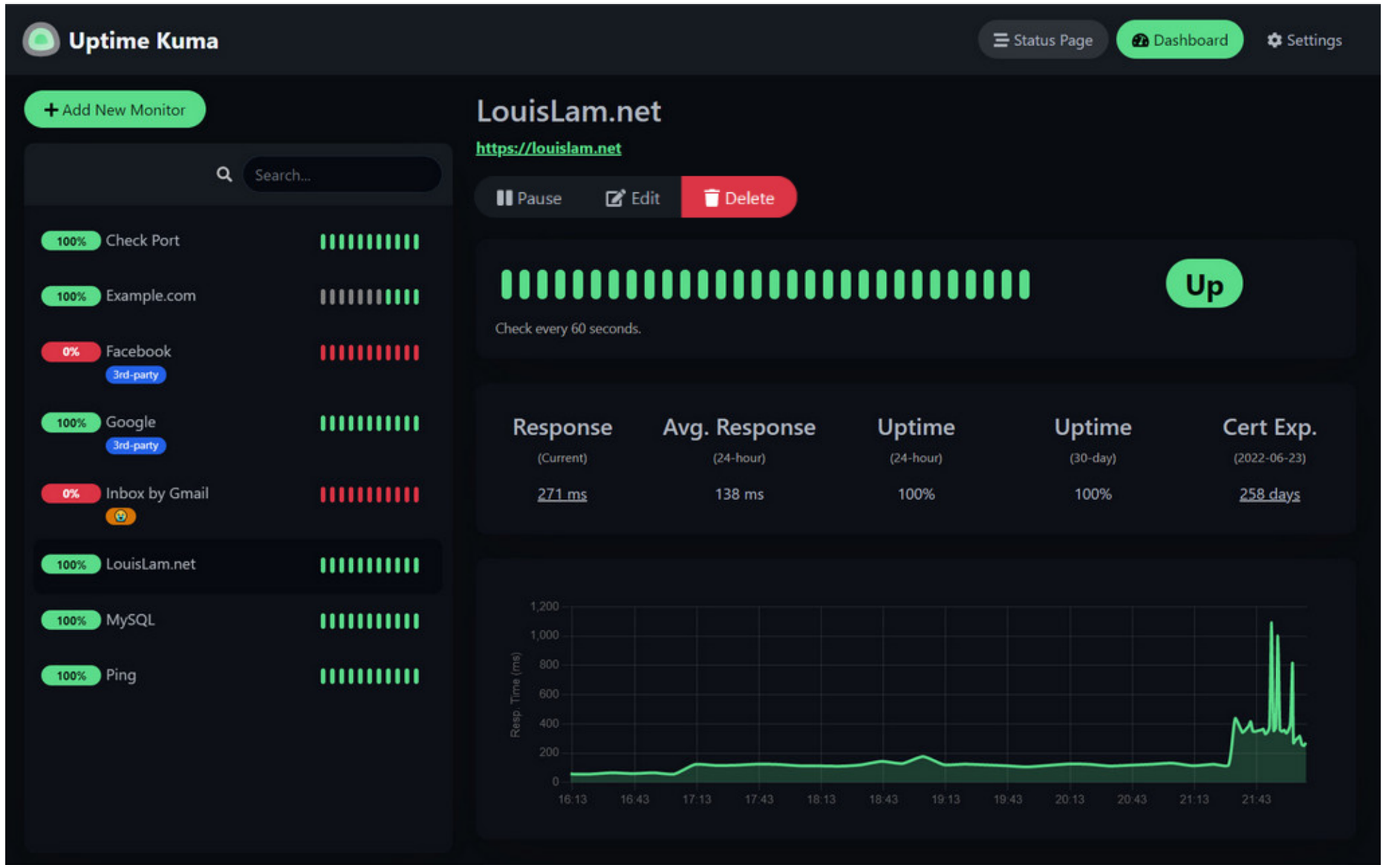
Service catalog
Uptime
Service discovery/registry

Service Catalog



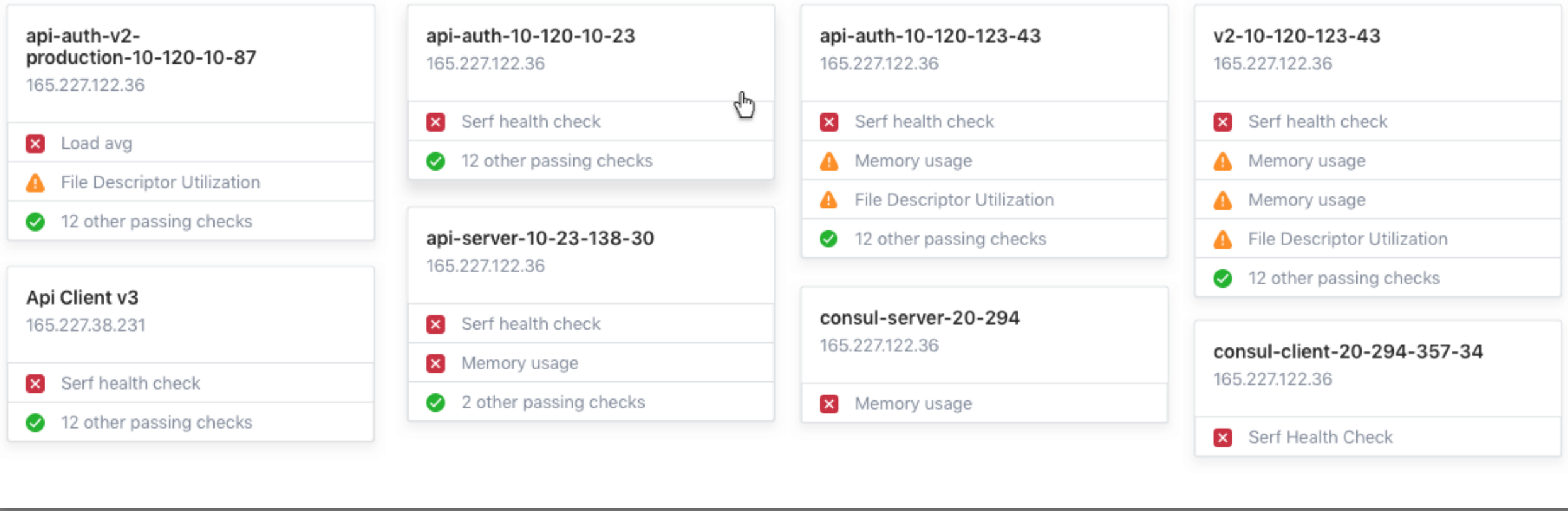
https://backstage.io/

Kuma Uptime



https://github.com/louislam/uptime-kuma

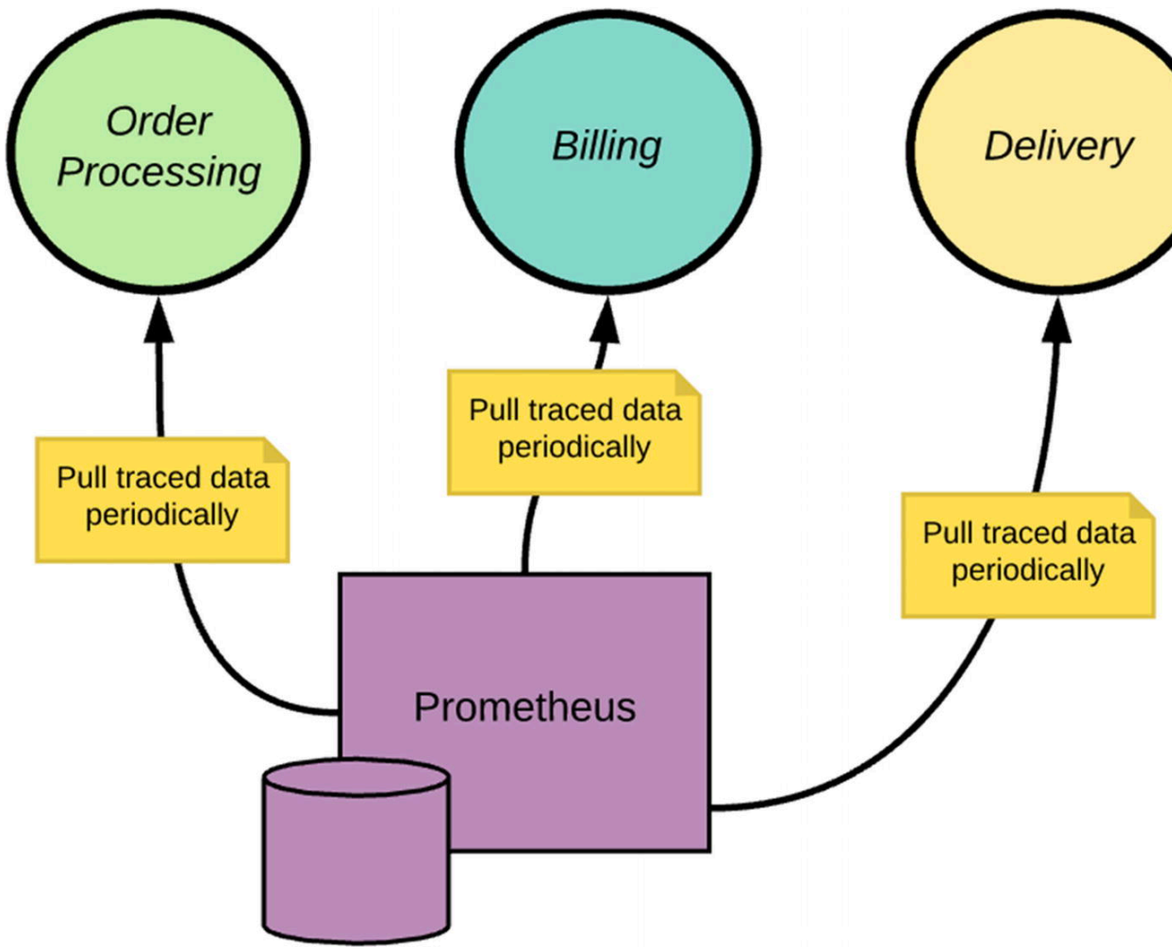
Service Discovery



https://www.consul.io/

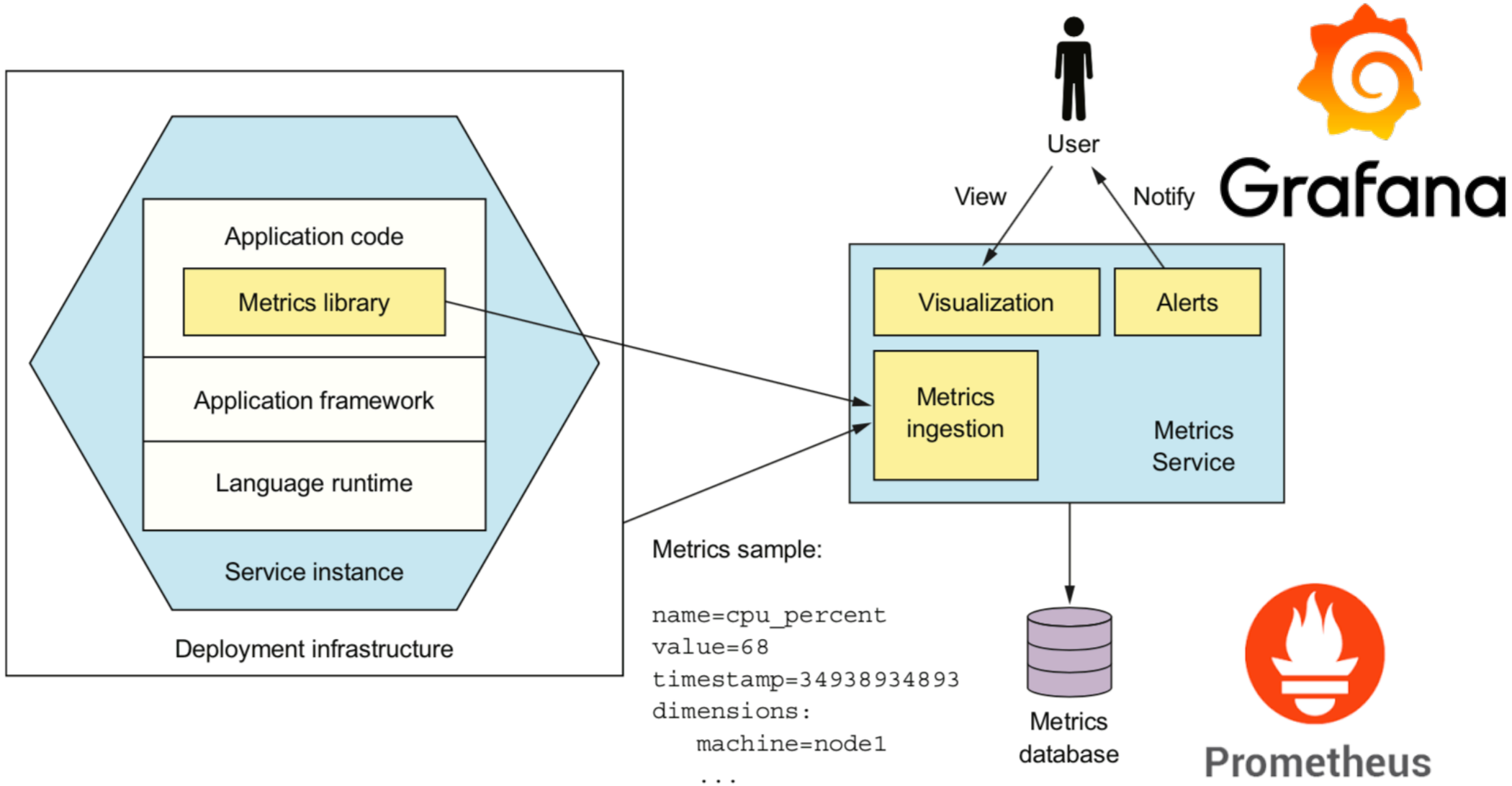
Application metrics

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



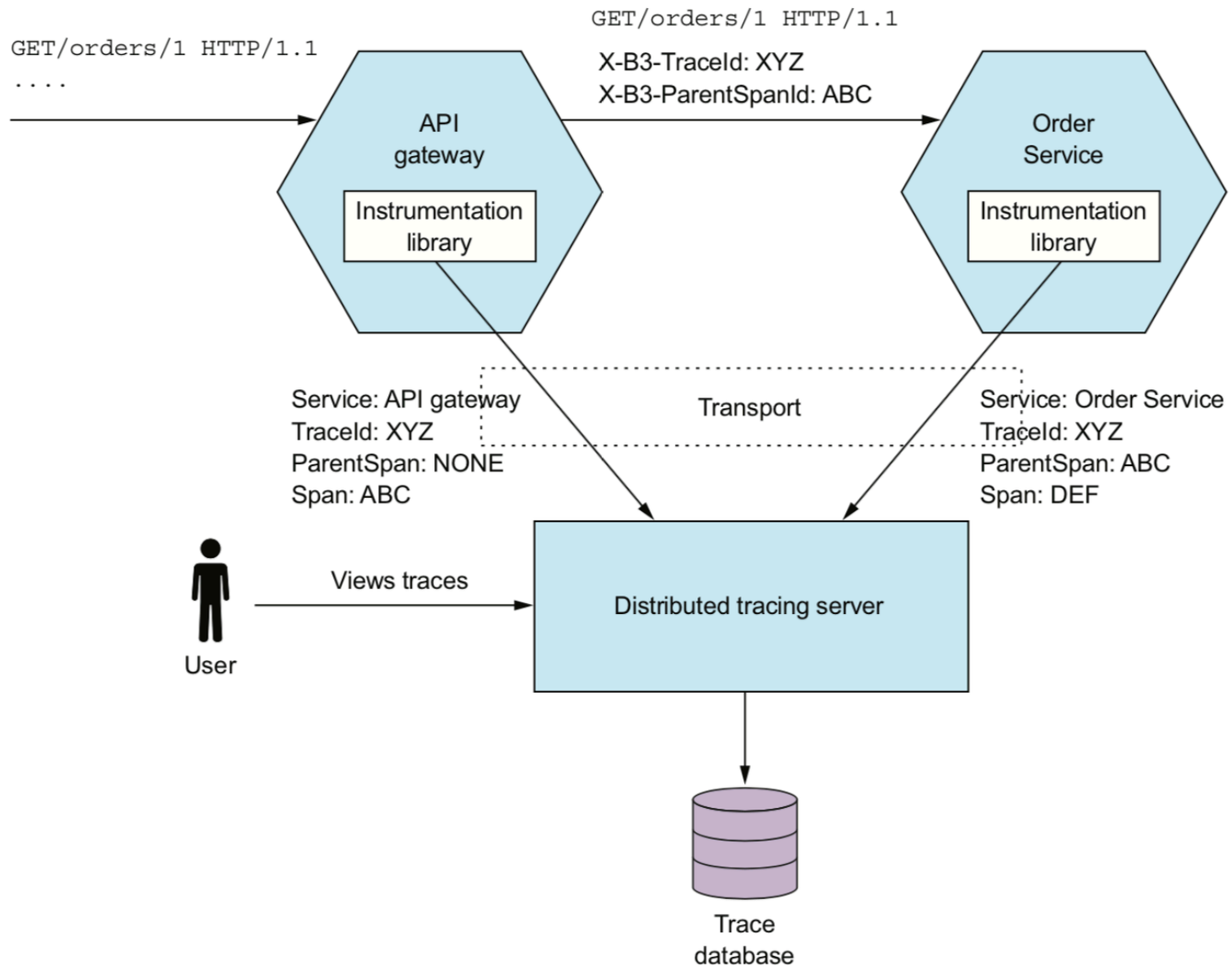
Application metrics

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

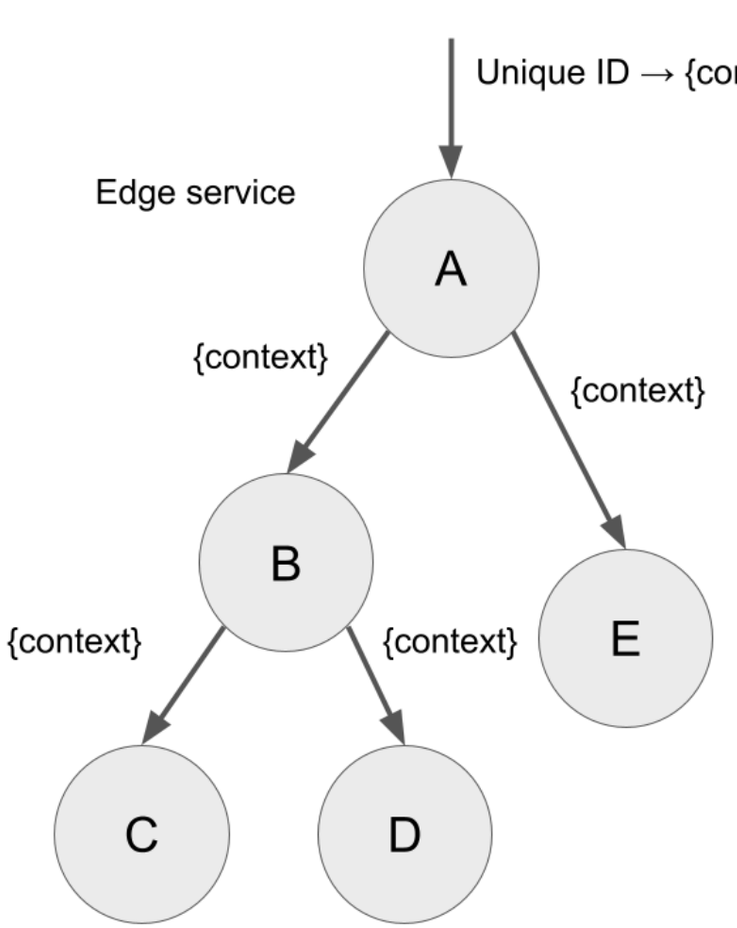


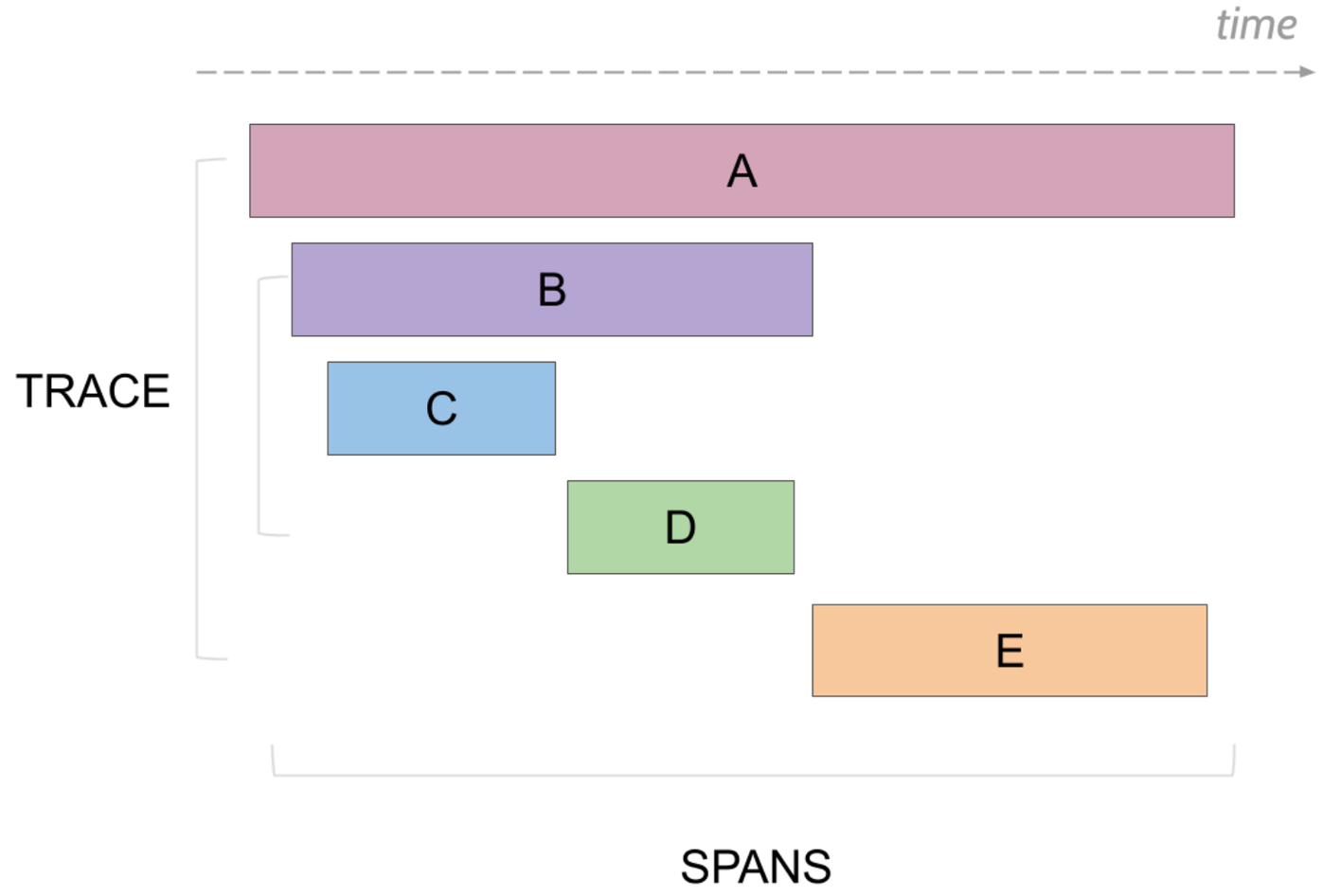
Distributed tracing

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

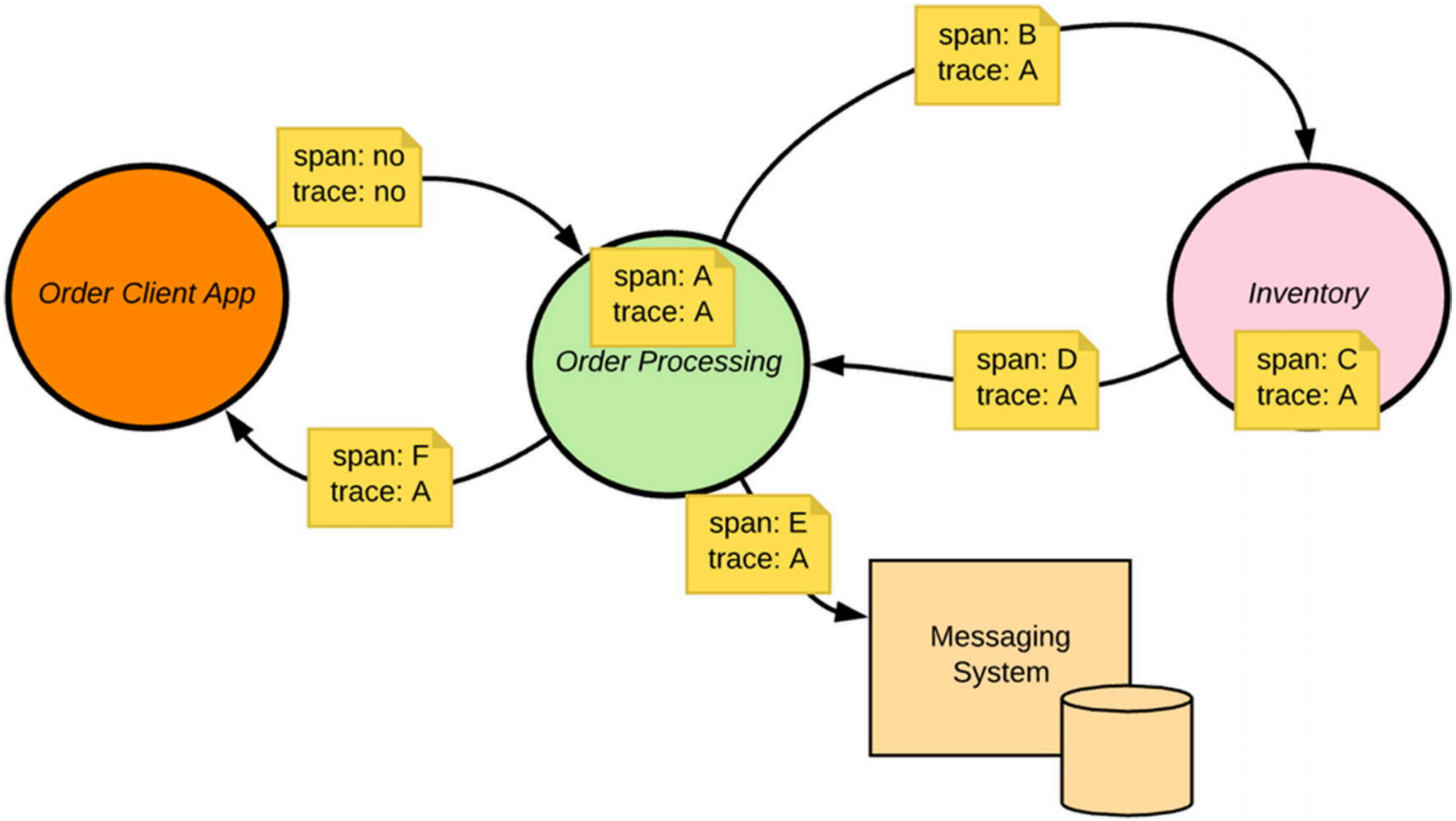


Distributed tracing

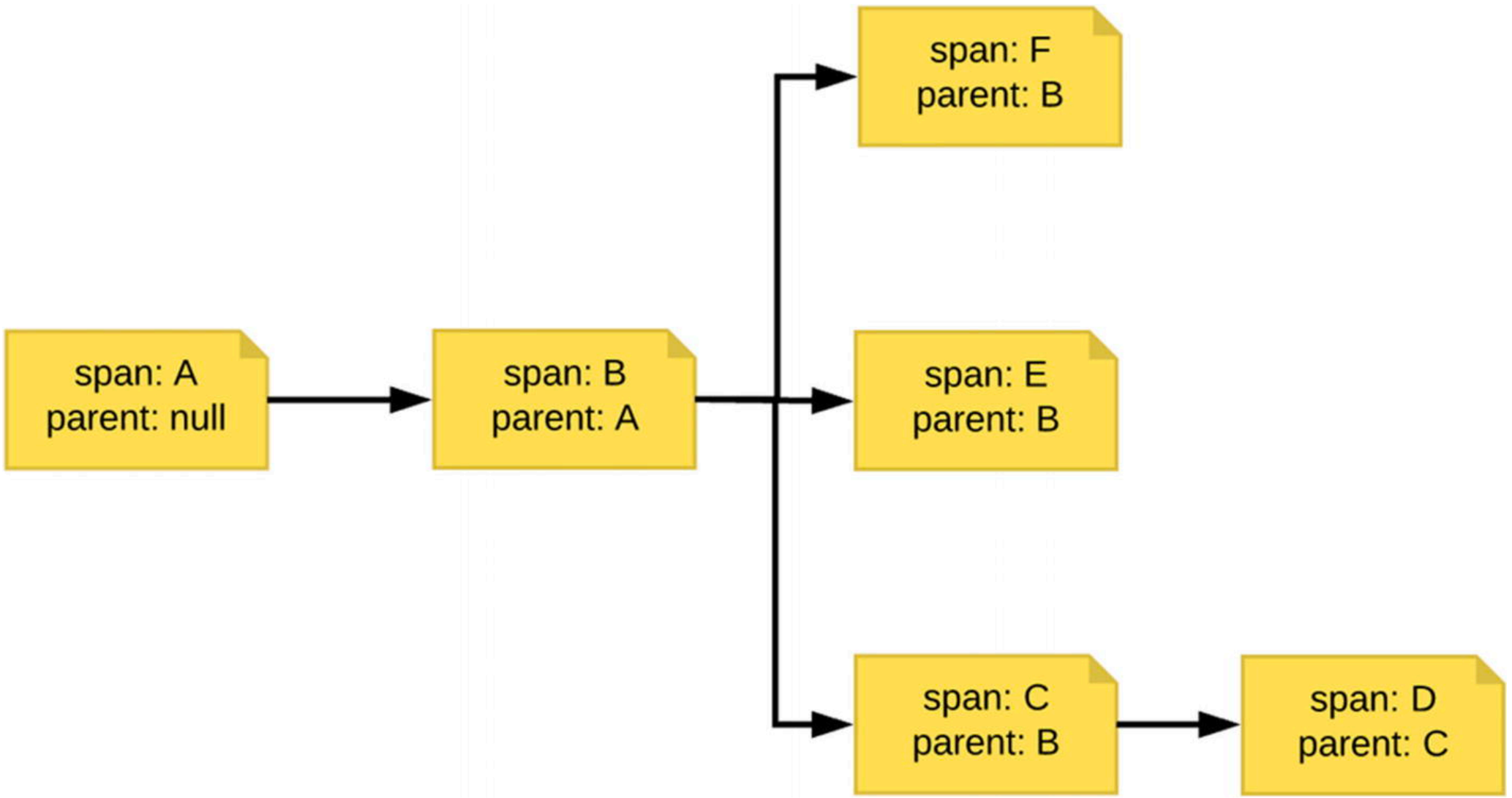




Distributed tracing



Distributed tracing



Distributed tracing tools

Format standard with OpenTelemetry
Zipkin, Jaeger, Tempo, AWS X-Ray, Elastic APM

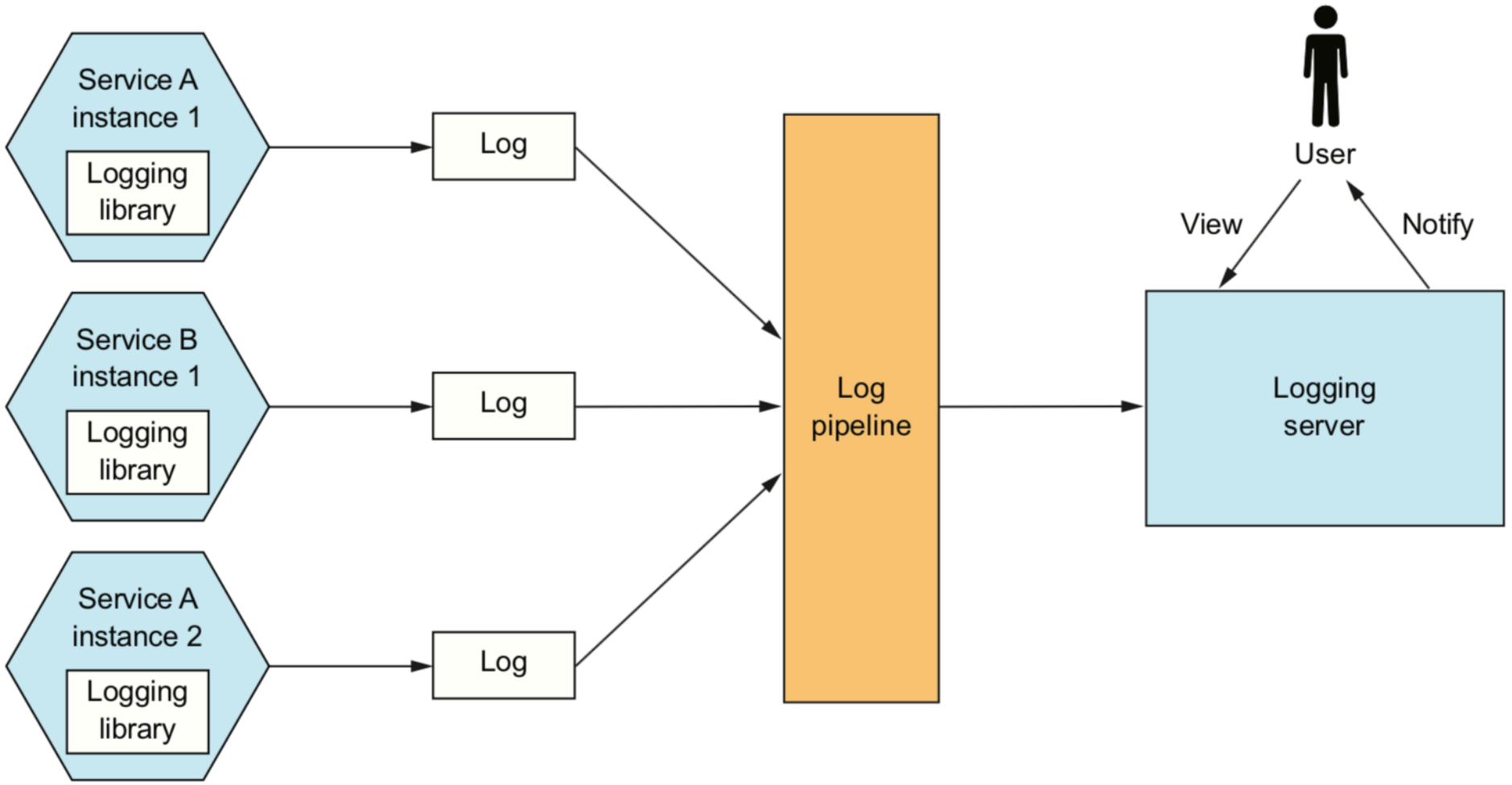
JAEGER

Grafana Tempo

OpenTelemetry

Log aggregation

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

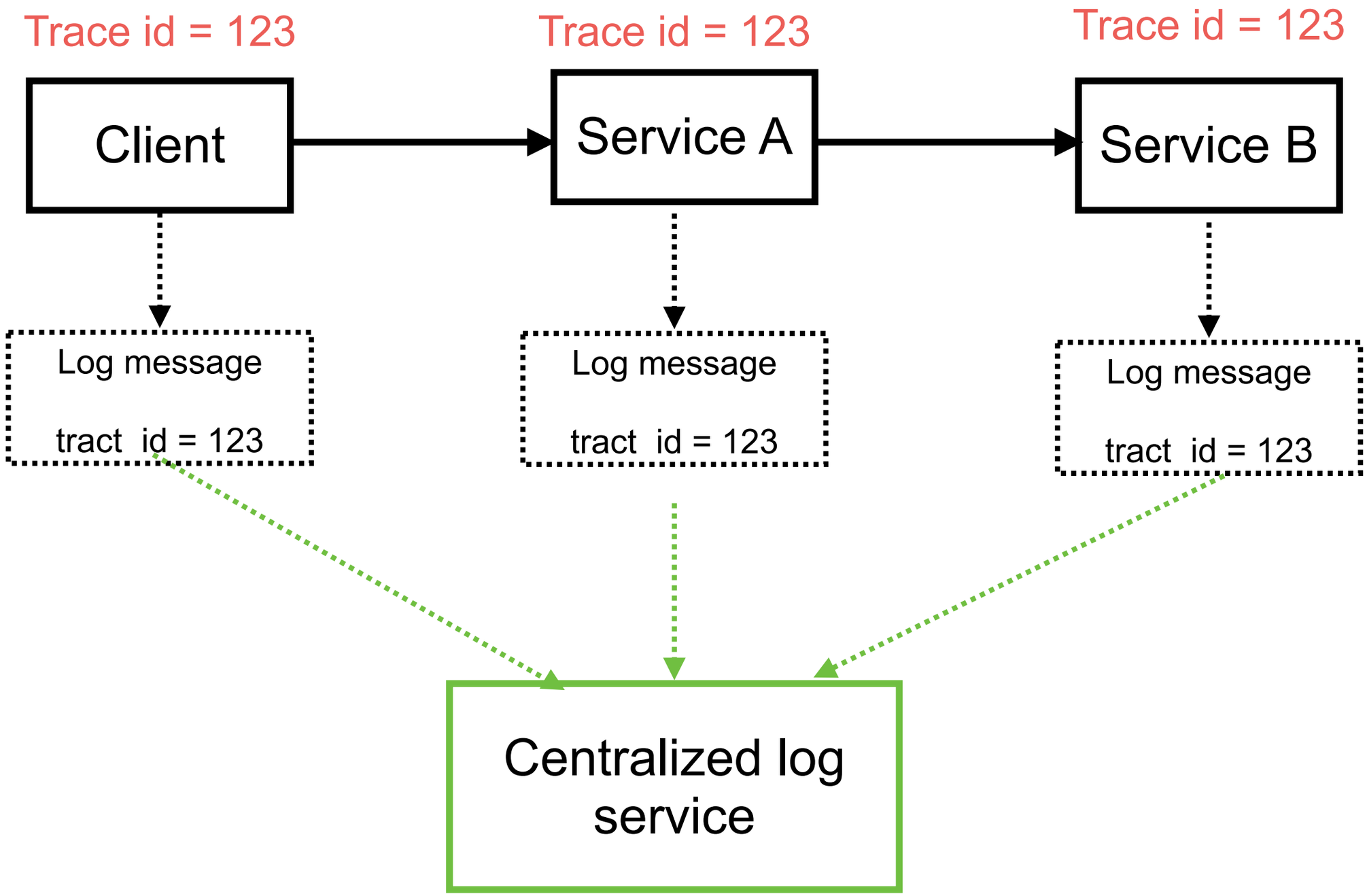


Effective Log Aggregation

Define event to log
Use structured logging
Exclude sensitive information
Log at the correct level
Be specific in your message
Don’t log large message
Make sure you keep trace Id in the log

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md

Log aggregation



Consistent Structure across all logs

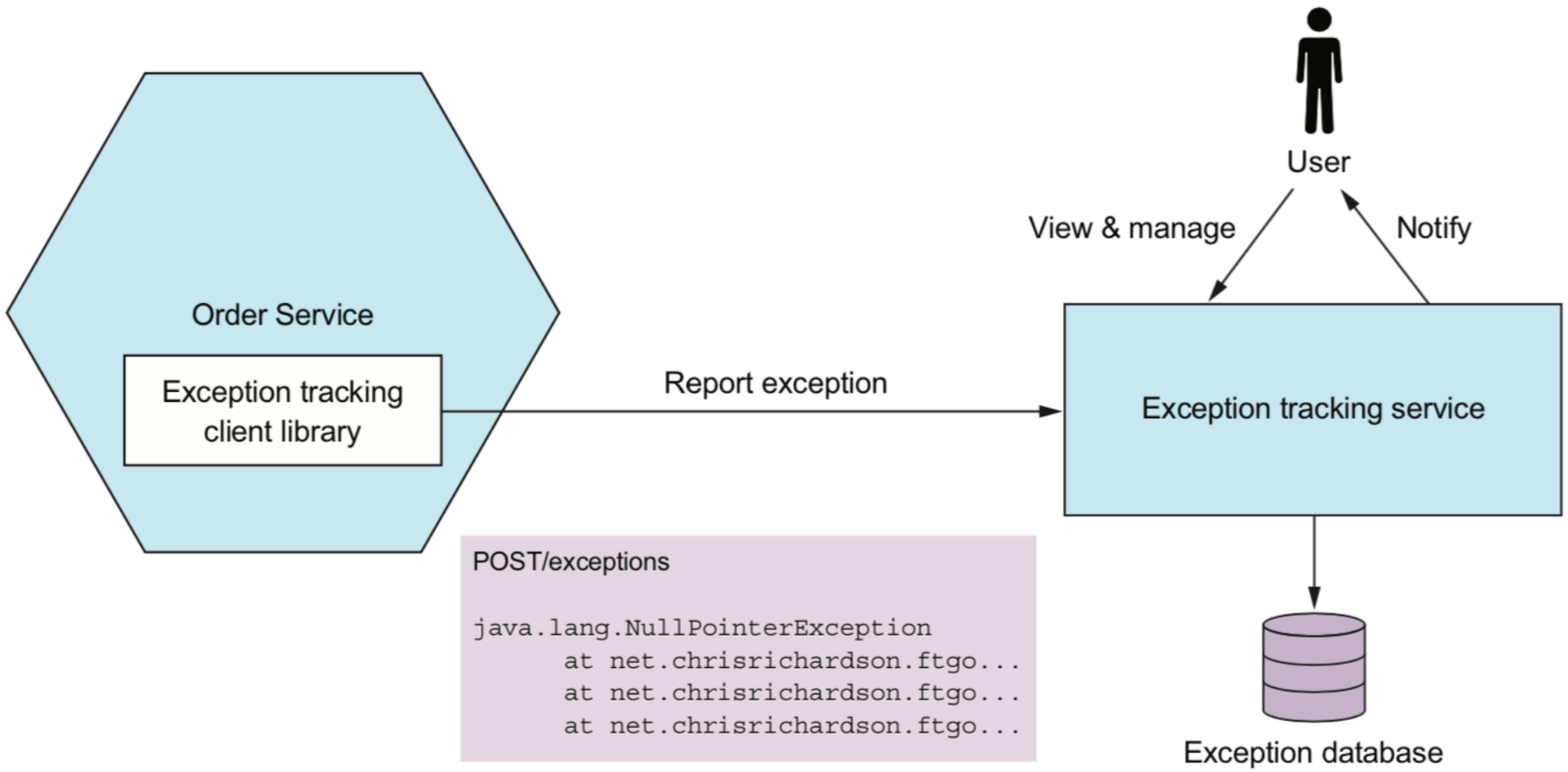
Property	Description	Example
Timestamp	Date and time of the log	2023-07-01
Log level	DEBUG, INFO, ERROR	
Trace Id or Correlation Id	Unique identifier that refer to other logs from all services	
Event/Action Name	Identify to event or action of log	Authentication fail
Service ID/ Name	Identify to service	
Request path	Path for the request	/api/products

Don’t keep !!

```
com.framework.FrameworkException: Error in web request
    at com.framework.ApplicationStarter.lambda$start$0(ApplicationStarter.java:15)
    at spark.RouteImpl$1.handle(RouteImpl.java:72)
    at spark.http.matching.Routes.execute(Routes.java:61)
    at spark.http.matching.MatcherFilter.doFilter(MatcherFilter.java:134)
    at spark.embeddedserver.jetty.JettyHandler.doHandle(JettyHandler.java:50)
    at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:1568)
    at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:144)
    at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:132)
    at org.eclipse.jetty.server.Server.handle(Server.java:503)
    at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:364)
    at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:260)
    at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:305)
    at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:103)
    at org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.java:118)
    at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:765)
    at org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedThreadPool.java:683)
    at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: com.project.module.MyProjectFooBarException: The number of FooBars cannot be zero
    at com.project.module.MyProject.anotherMethod(MyProject.java:20)
    at com.project.module.MyProject.someMethod(MyProject.java:12)
    at com.framework.ApplicationStarter.lambda$start$0(ApplicationStarter.java:13)
    ... 16 more
Caused by: java.lang.ArithmeticException: The denominator must not be zero
    at org.apache.commons.lang3.math.Fraction.getFraction(Fraction.java:143)
    at com.project.module.MyProject.anotherMethod(MyProject.java:18)
    ... 18 more
```

Exception tracking

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



Exception tracking services

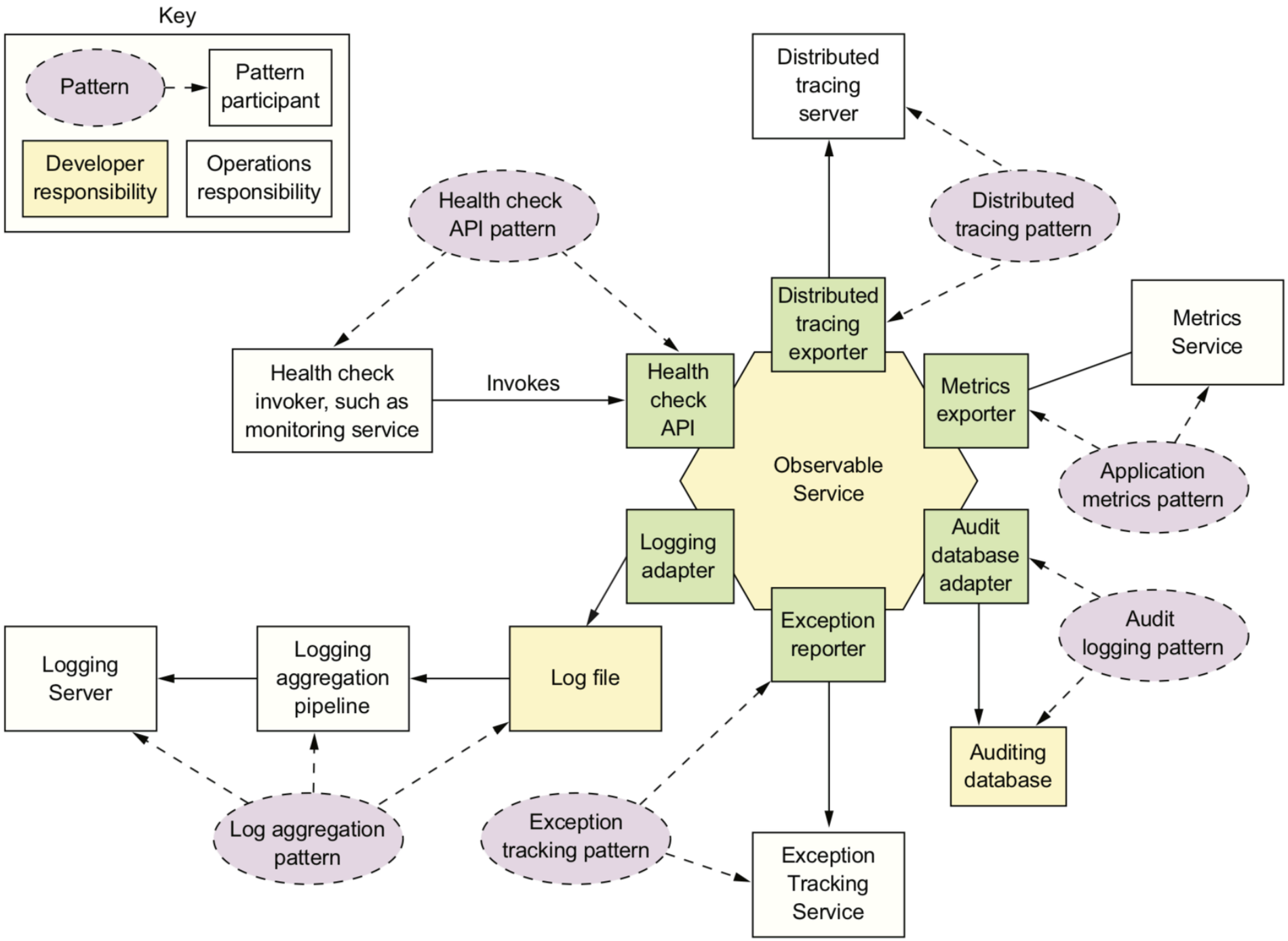
Audit logging

Log of user actions
Help customer support
Ensure compliance
Detect suspicious behavior

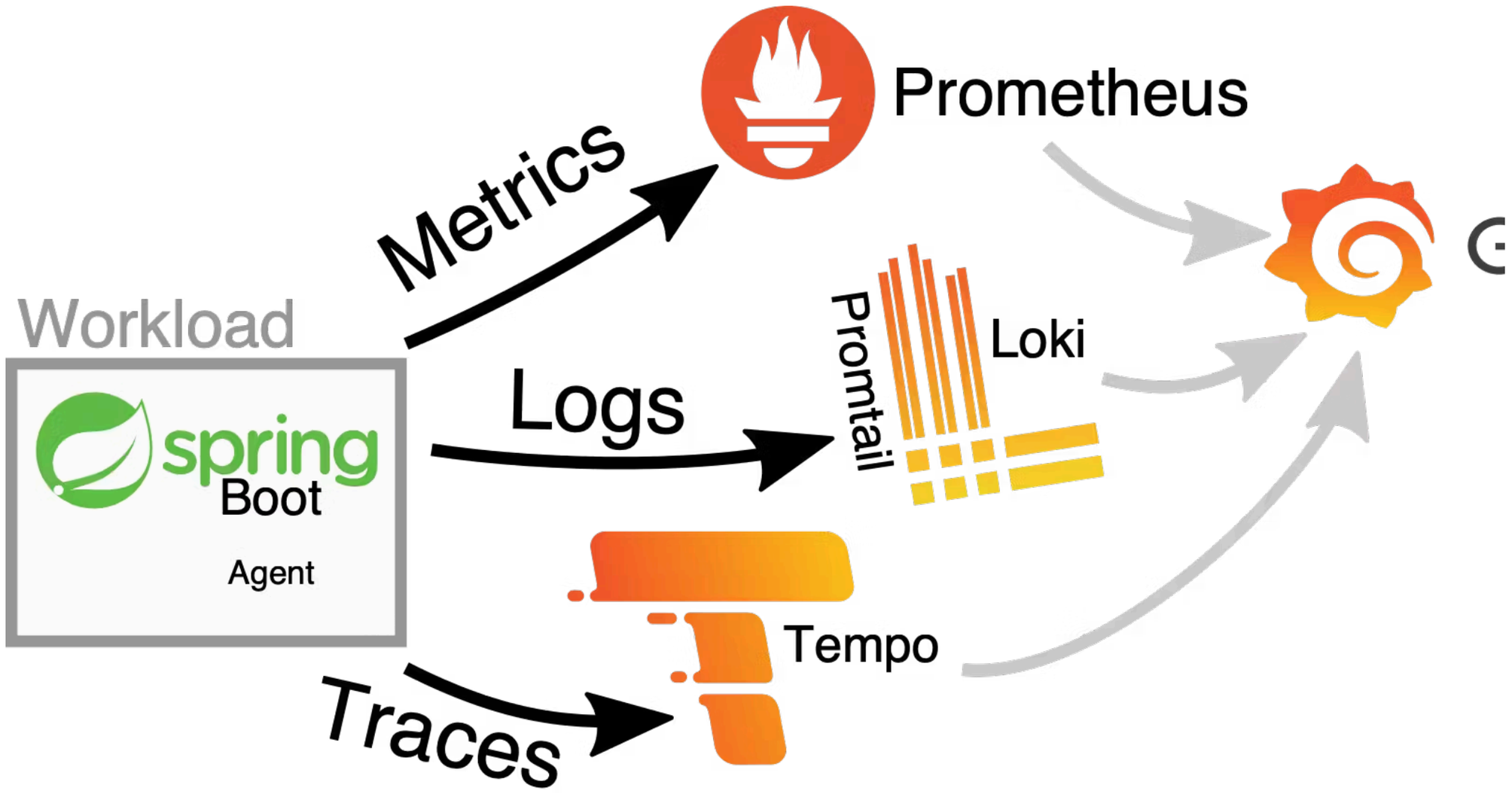
How to implement the audit logging ?

Add logging code in business logic
Use AOP (Aspect-Oriented Programming)
Use event sourcing

Observable services

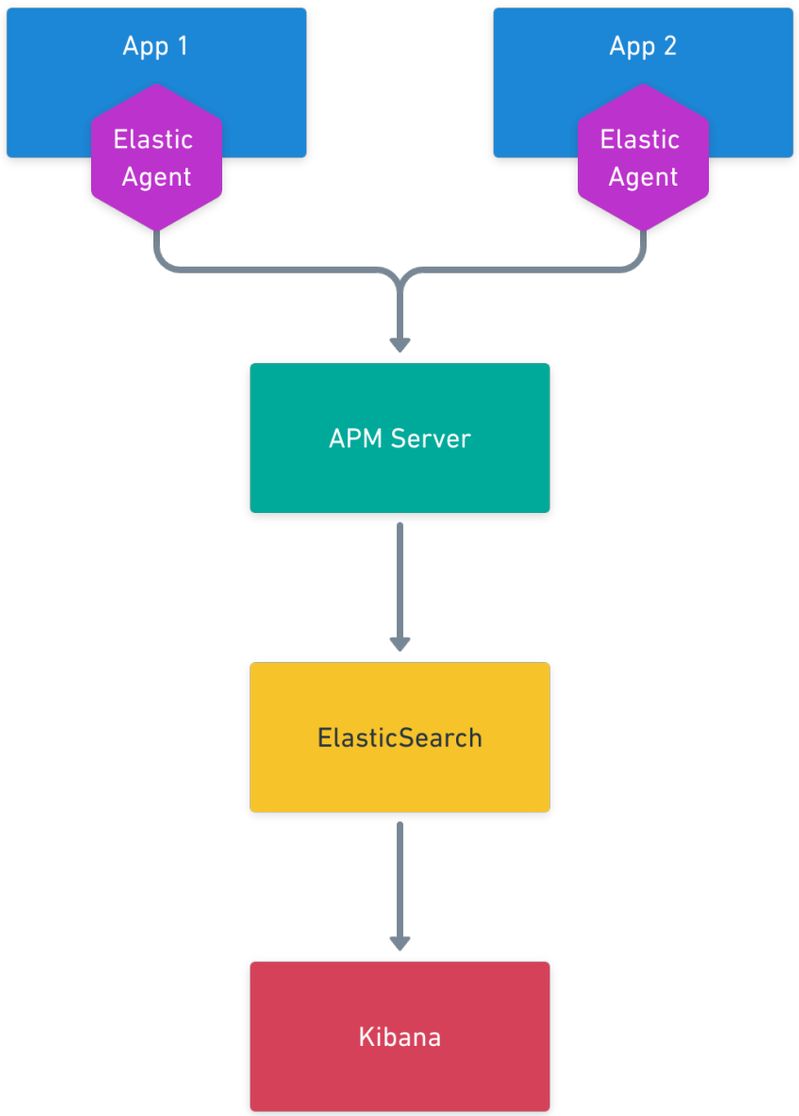


Grafana platform



https://grafana.com/

Elastic APM



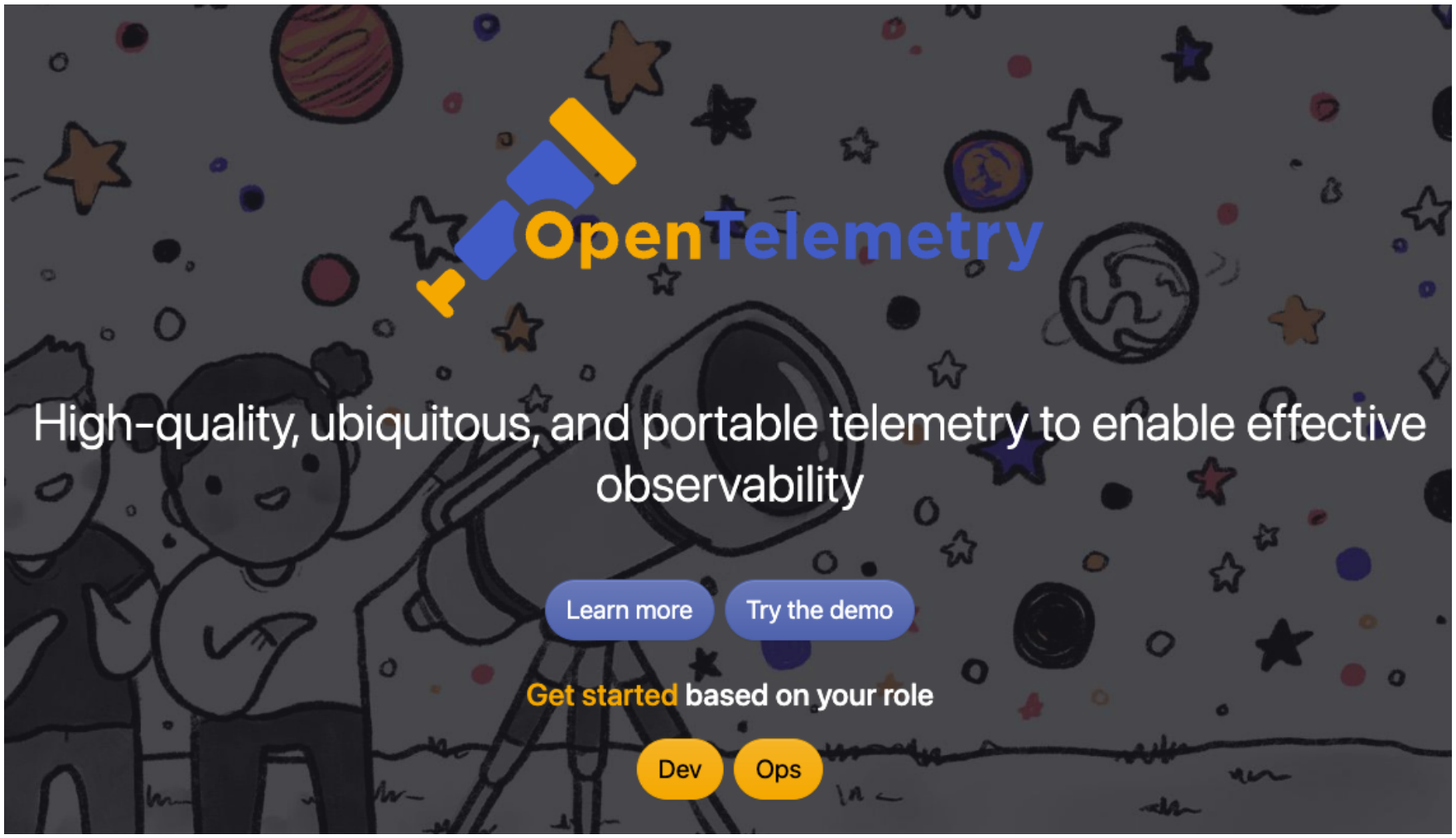
https://www.elastic.co/observability/application-performance-monitoring

Observable service workshop

https://github.com/up1/workshop-develop-microservices-2023

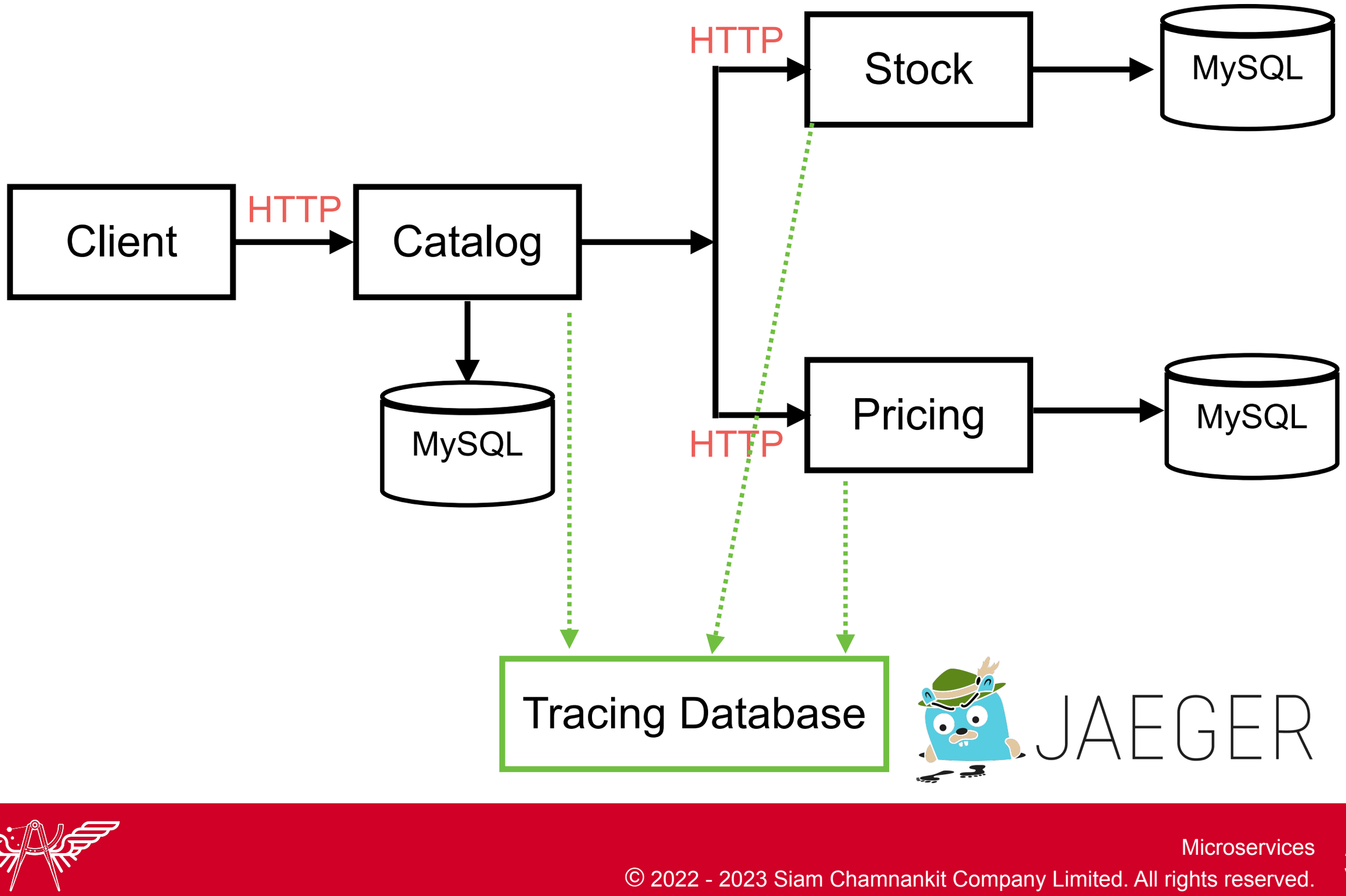
Distributed Tracing

Working with OpenTelemetry

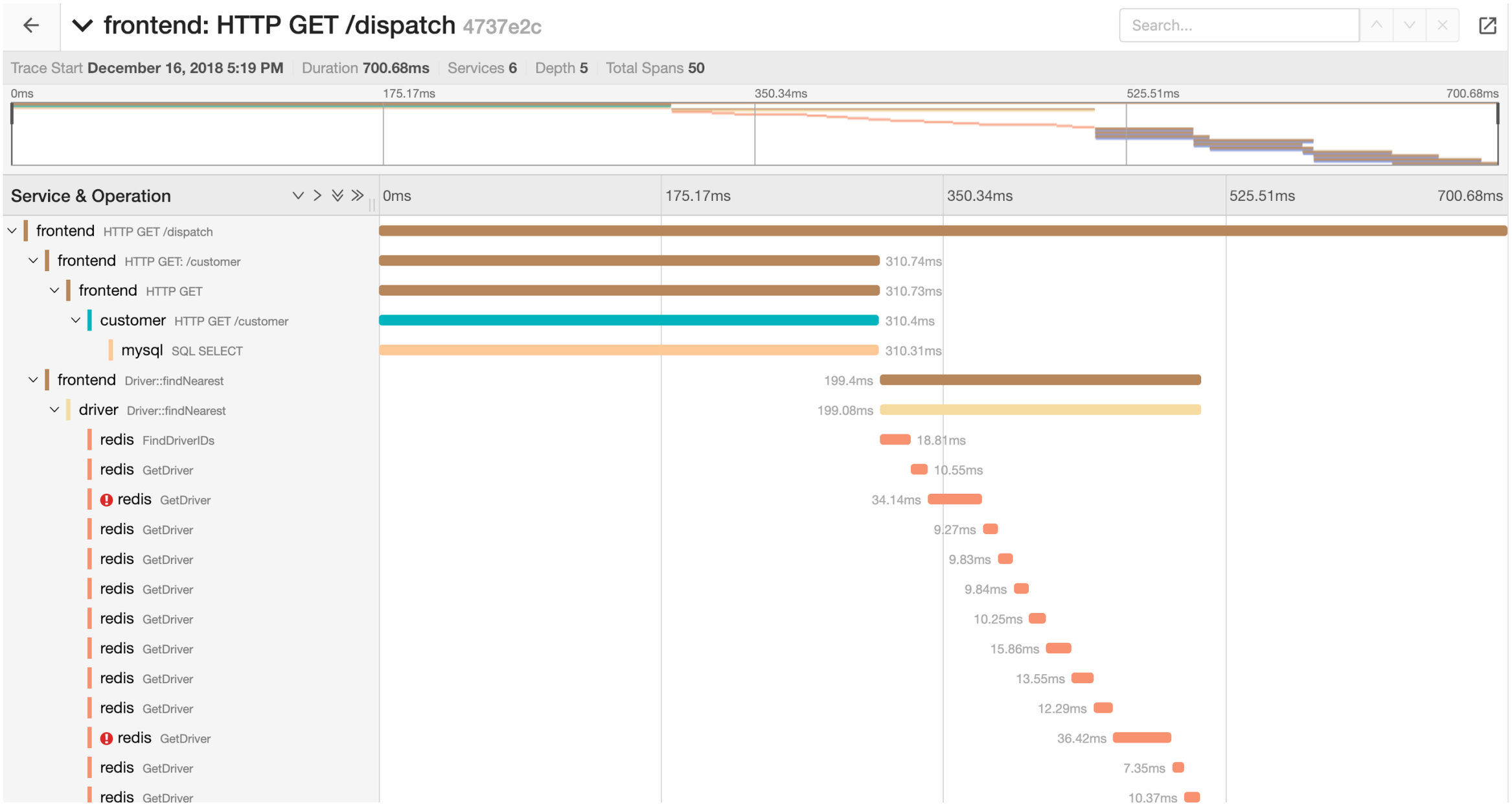


https://opentelemetry.io/

Distributed Tracing

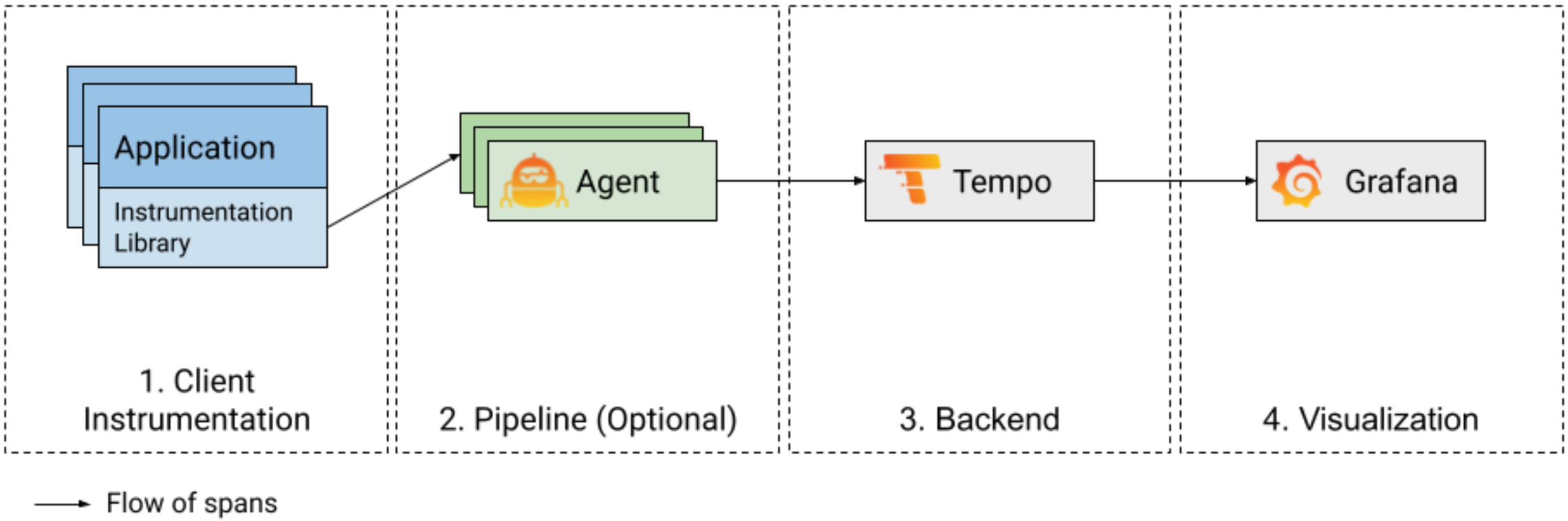


Distributed Tracing with Jaeger



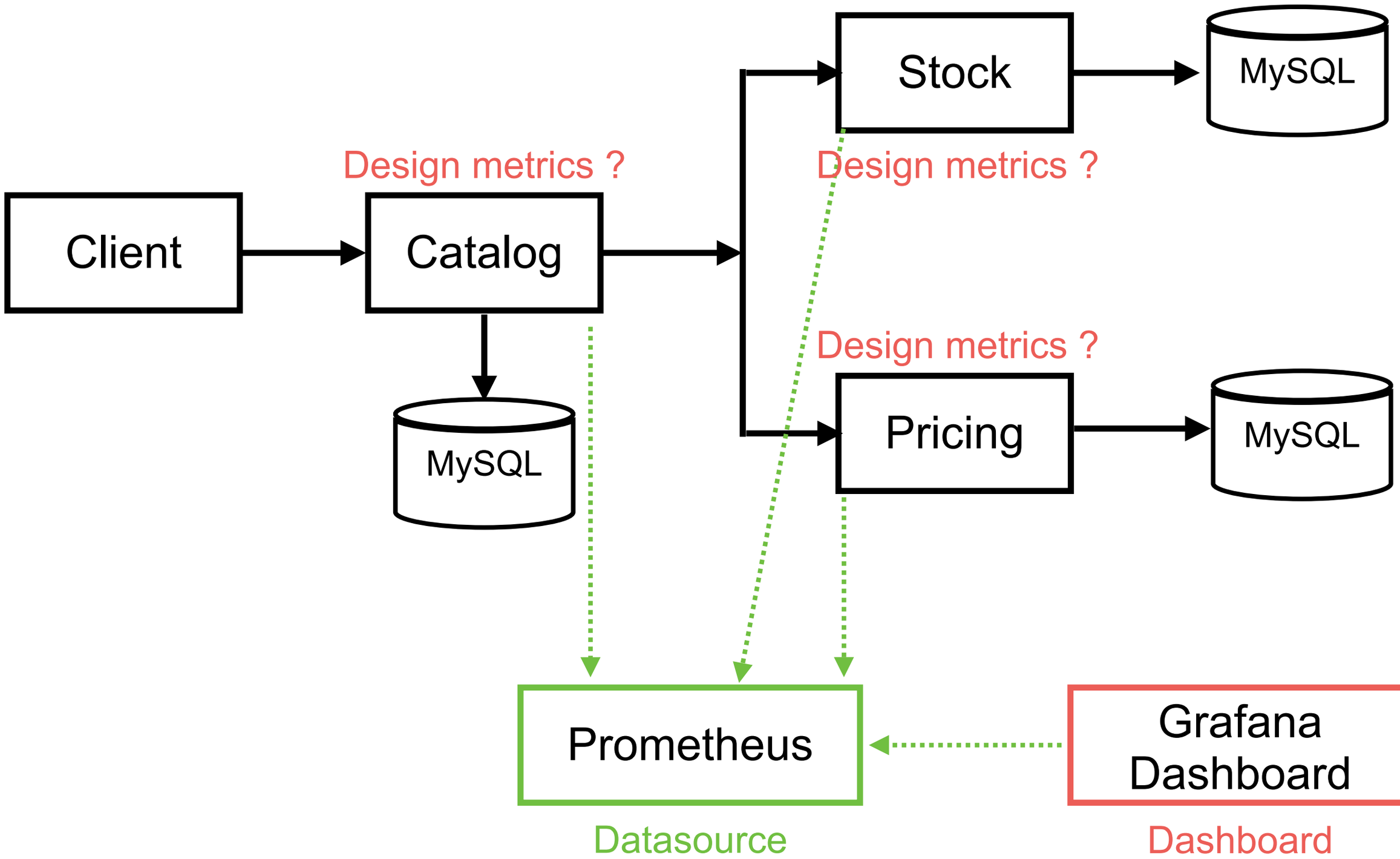
https://www.jaegertracing.io

Distributed Tracing with Grafana

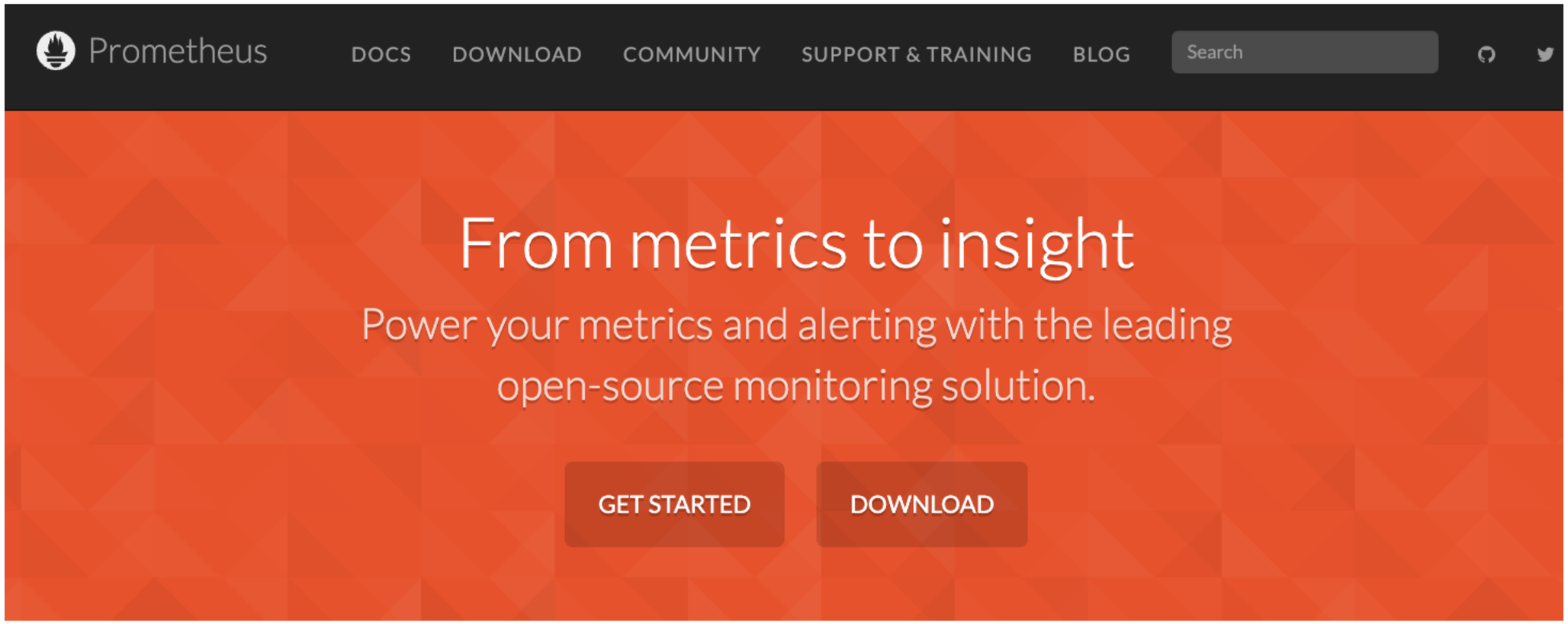


https://www.jaegertracing.io

Application metric

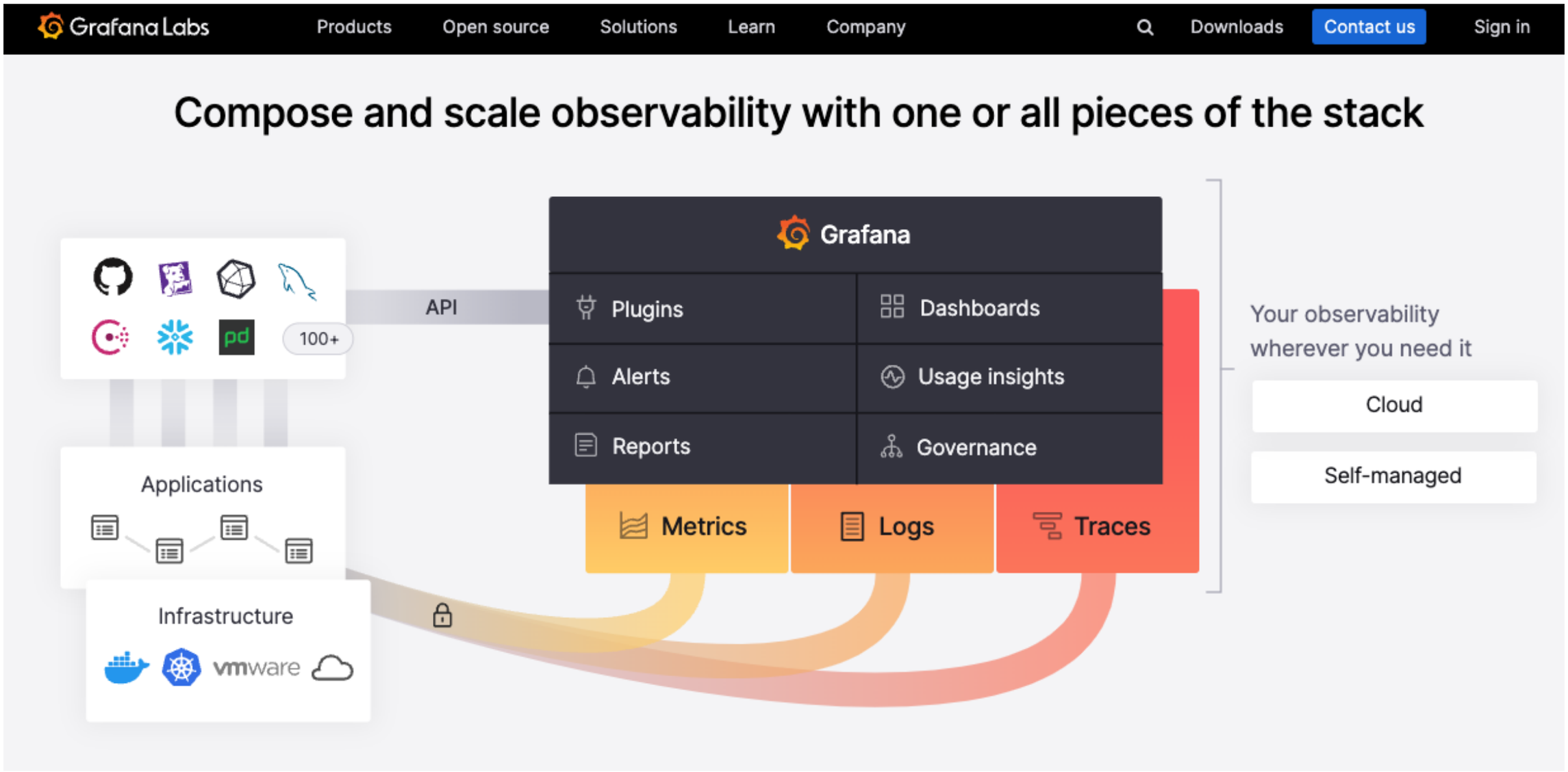


Prometheus



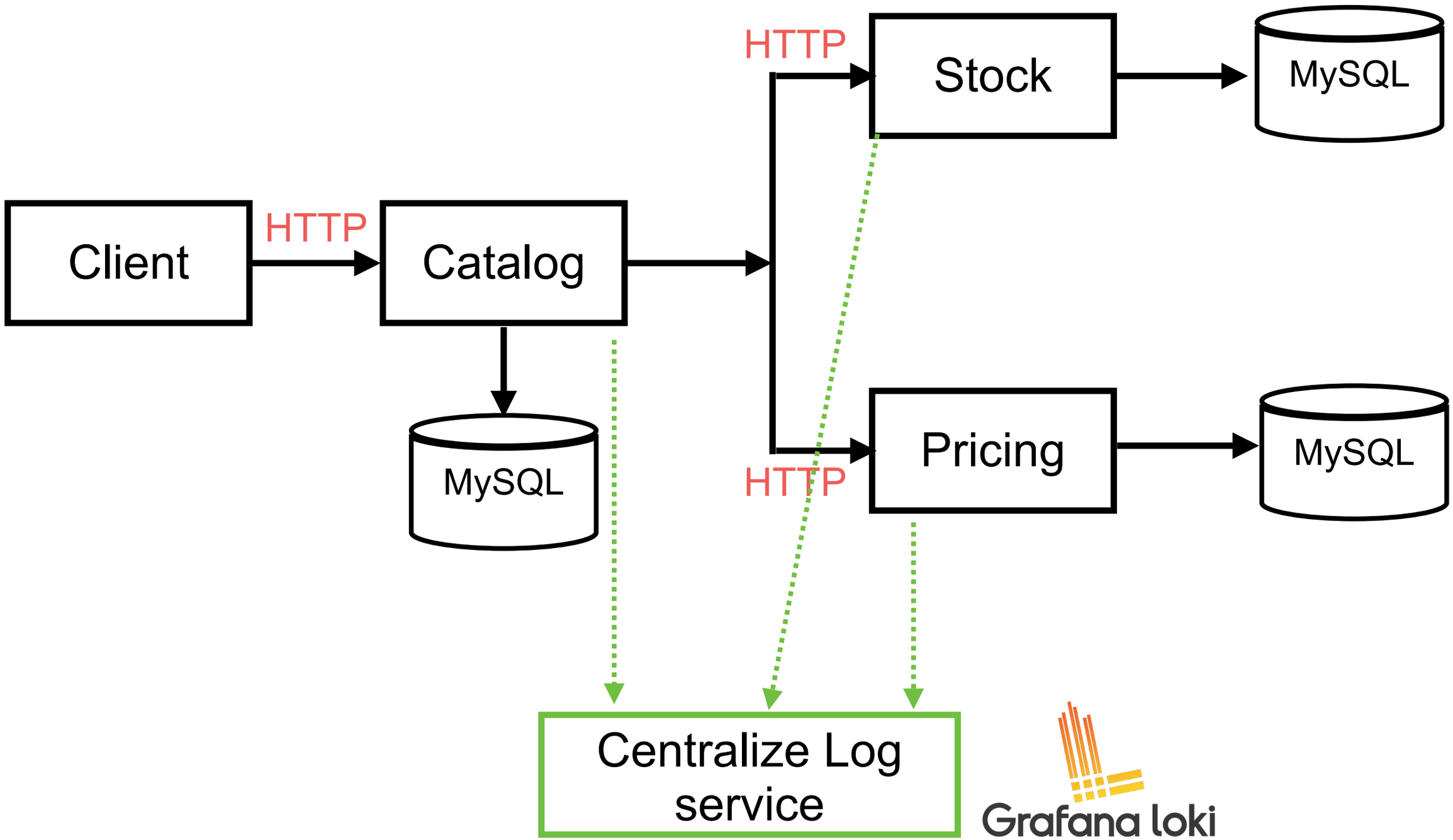
https://prometheus.io/

Grafana



https://grafana.com/

Log aggregation



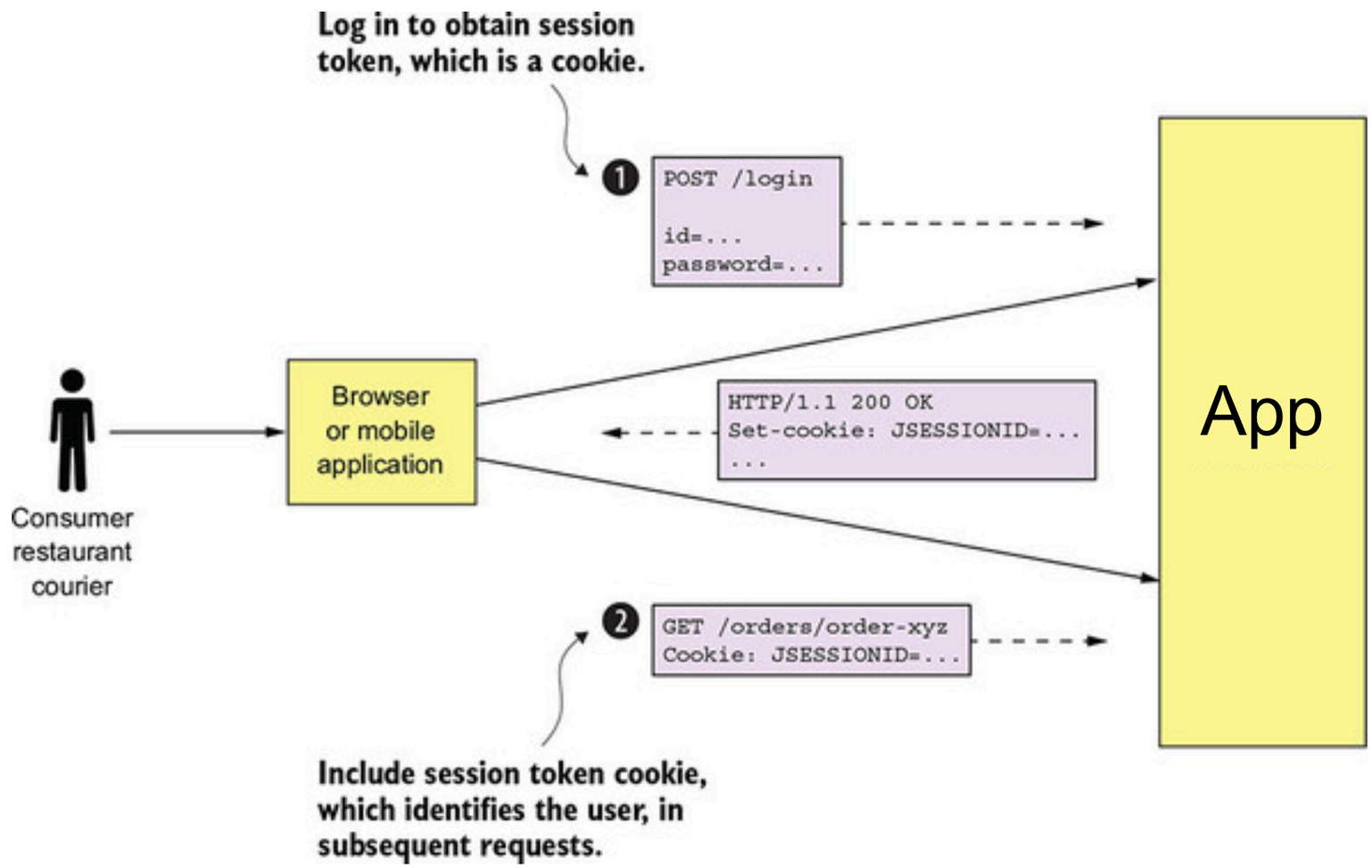
Secure services

Develop secure services

Authentication
Authorization
Auditing
Secure interprocess communication

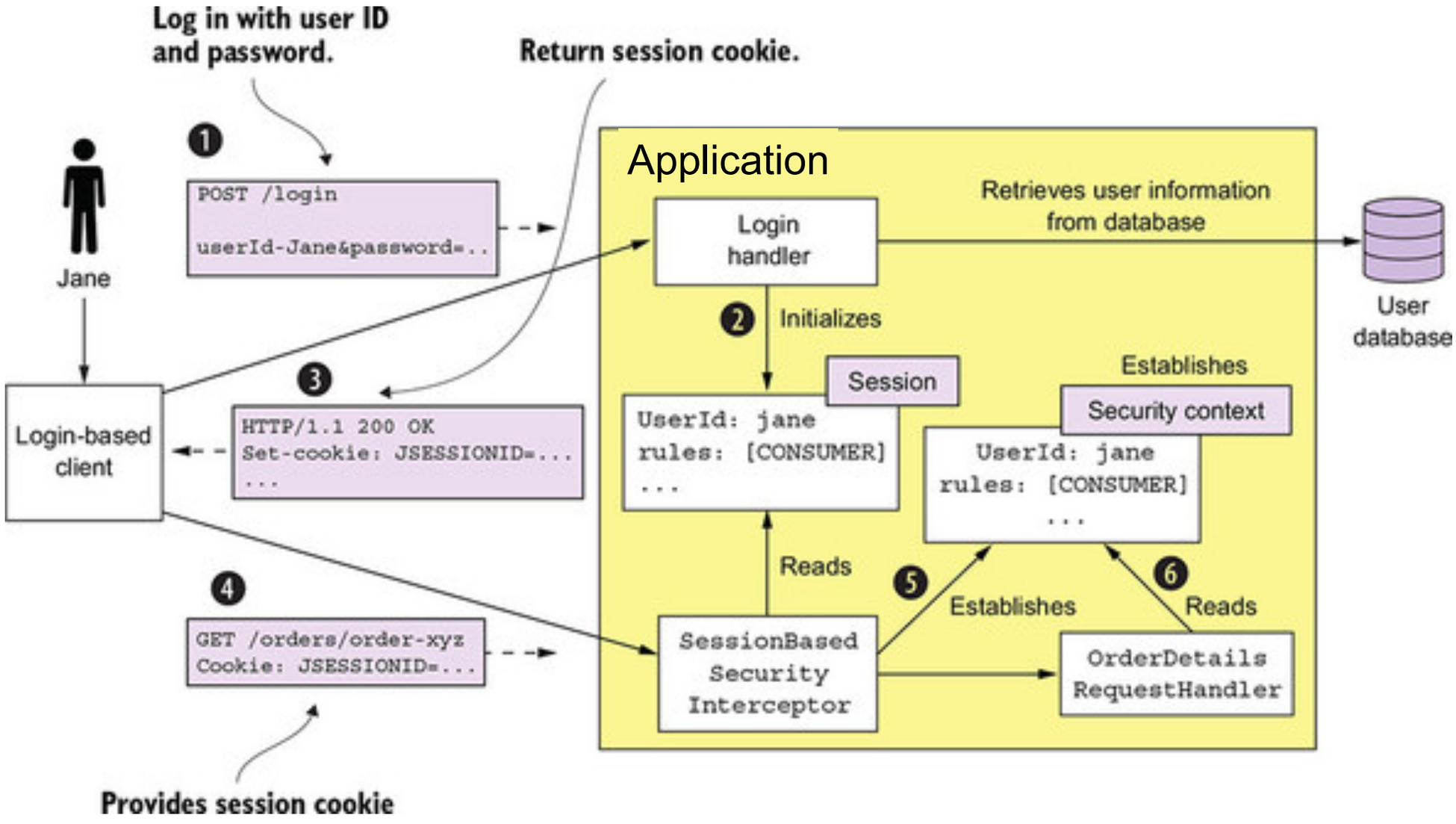
Security in traditional application

Keep security information in browser's cookie



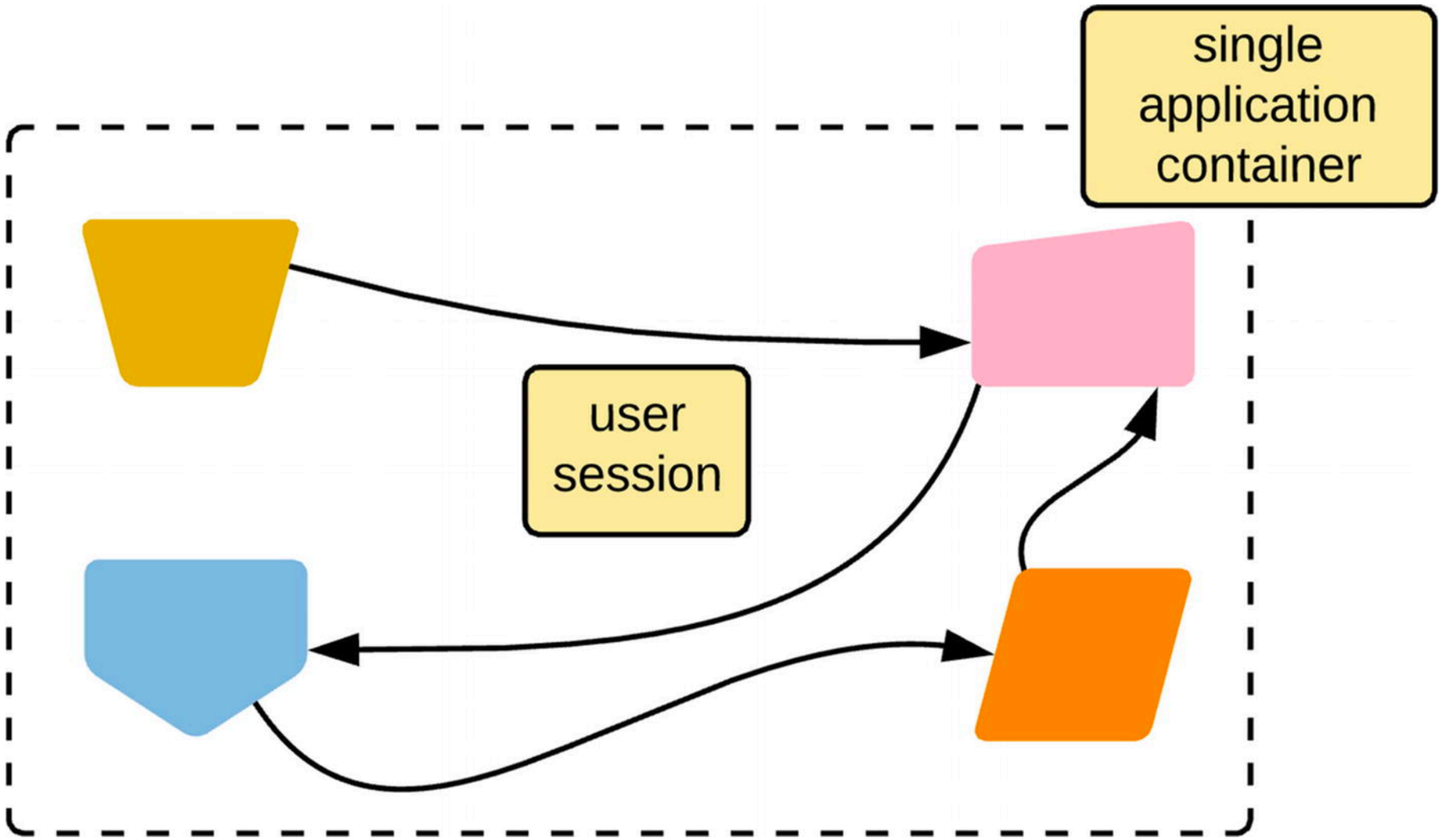
Security in traditional application

Implement security process in application



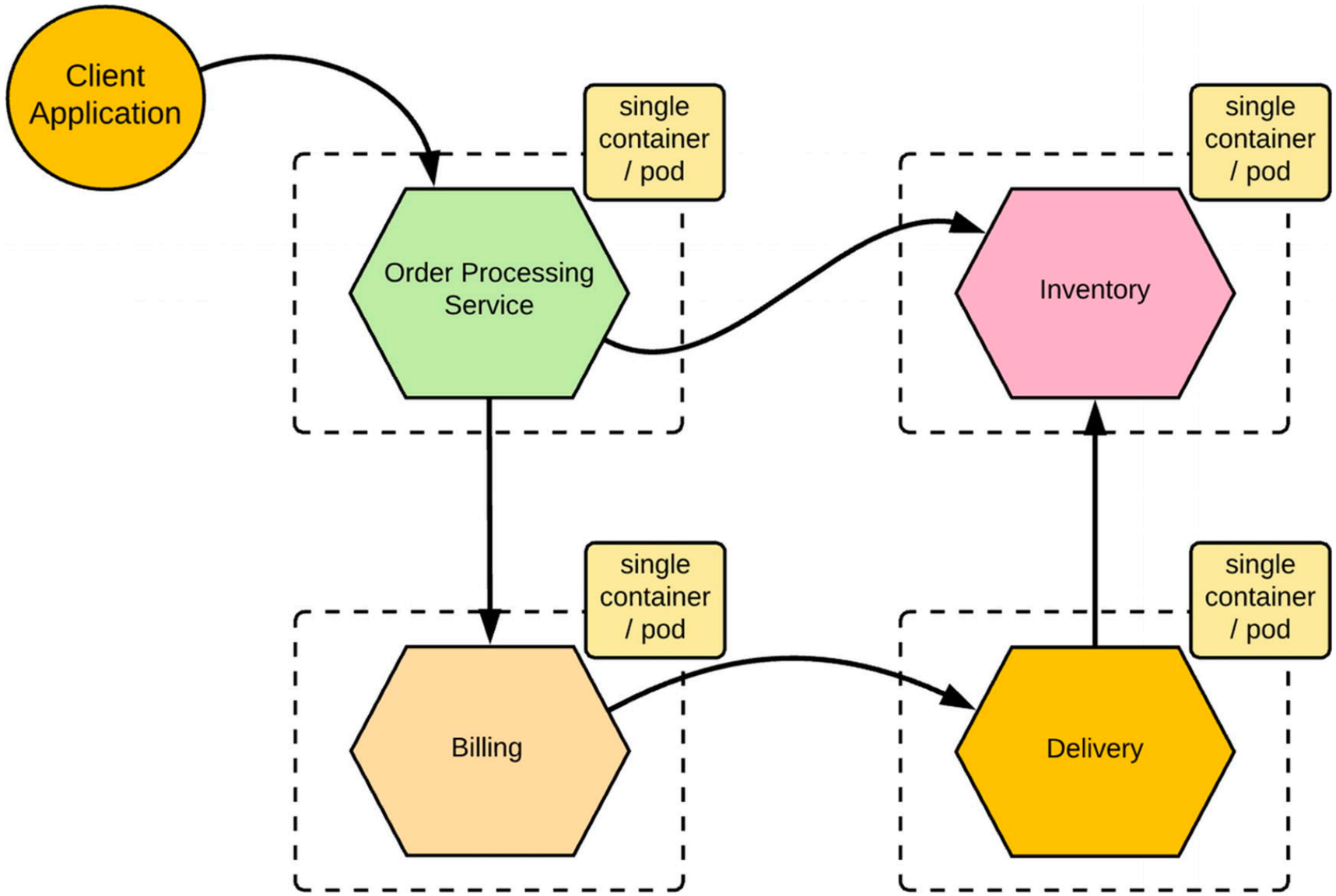
Security in traditional application

Sharing a user session in application



Security in multiple services ?

Interaction between multiple services

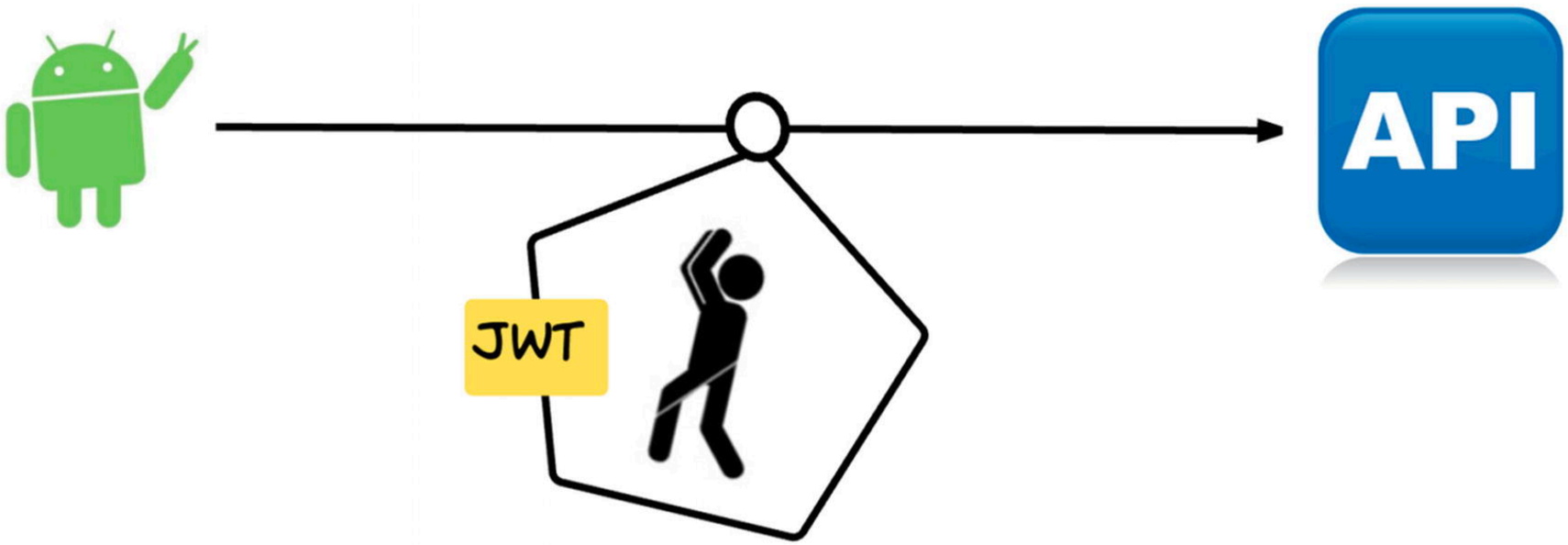


Secure service-to-service communication

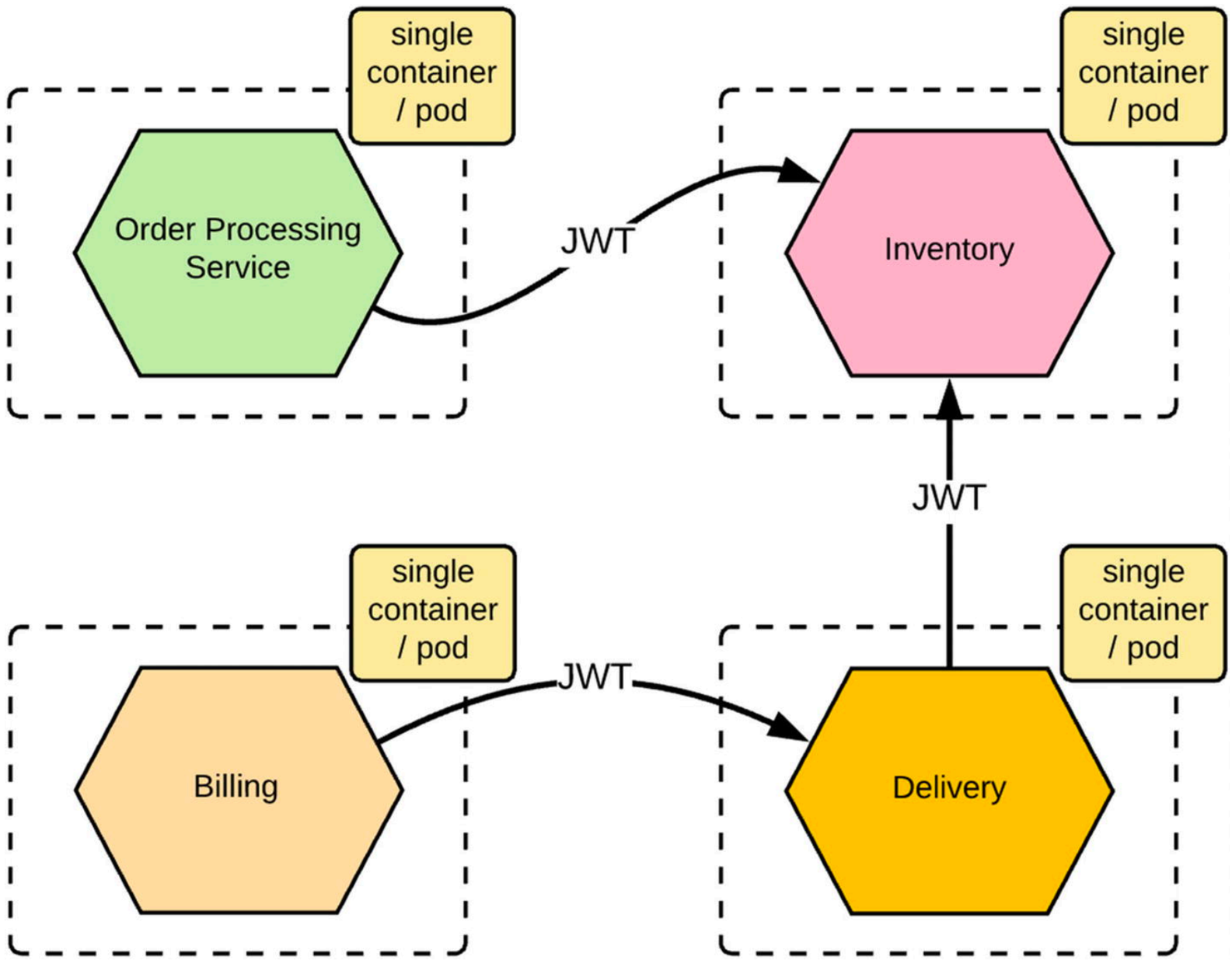
JSON Web Token (JWT) + OAuth
Transport Layer Security (TLS) mutual authentication

JSON Web Token (JWT)

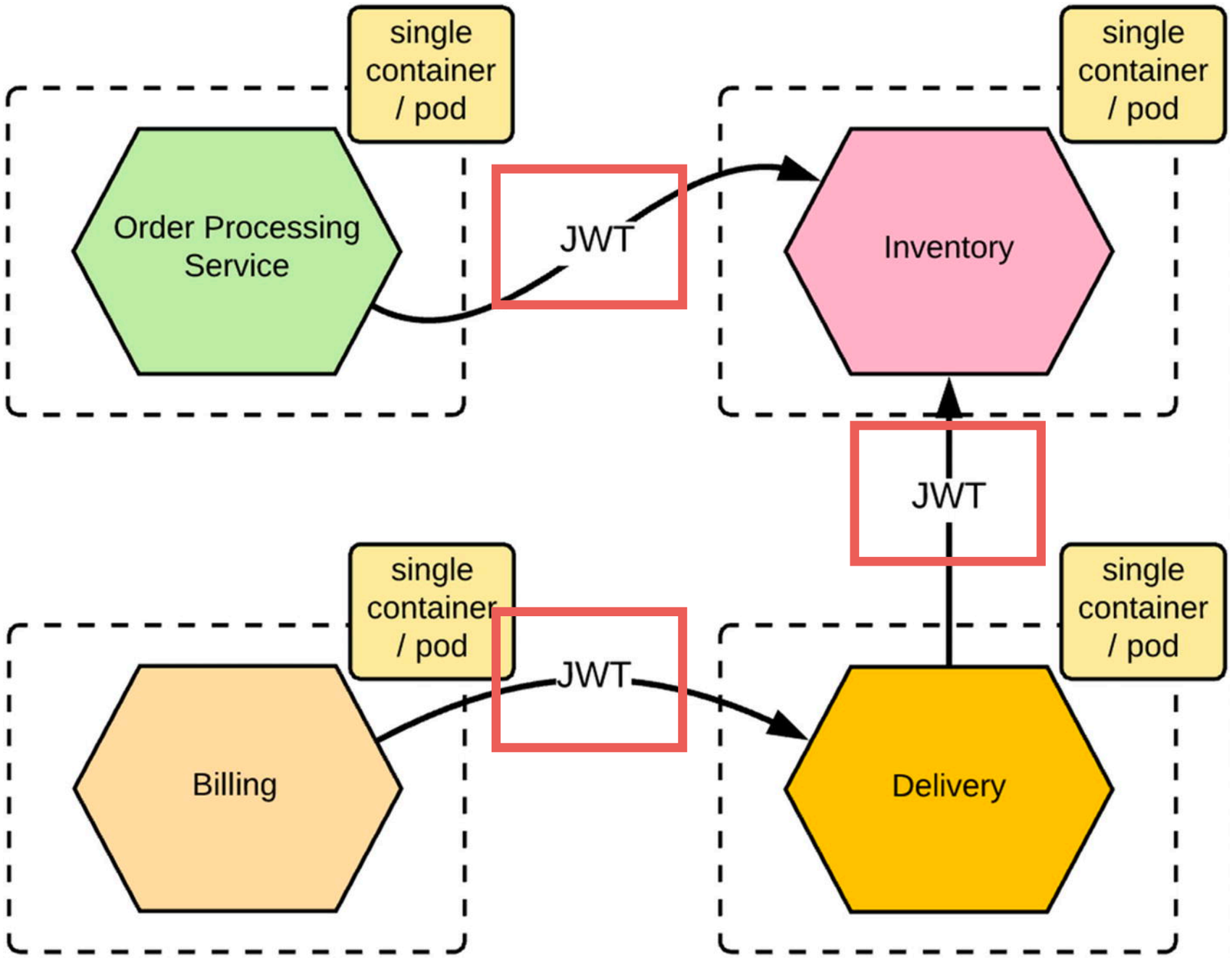
Define a container to transport data
between interested parties



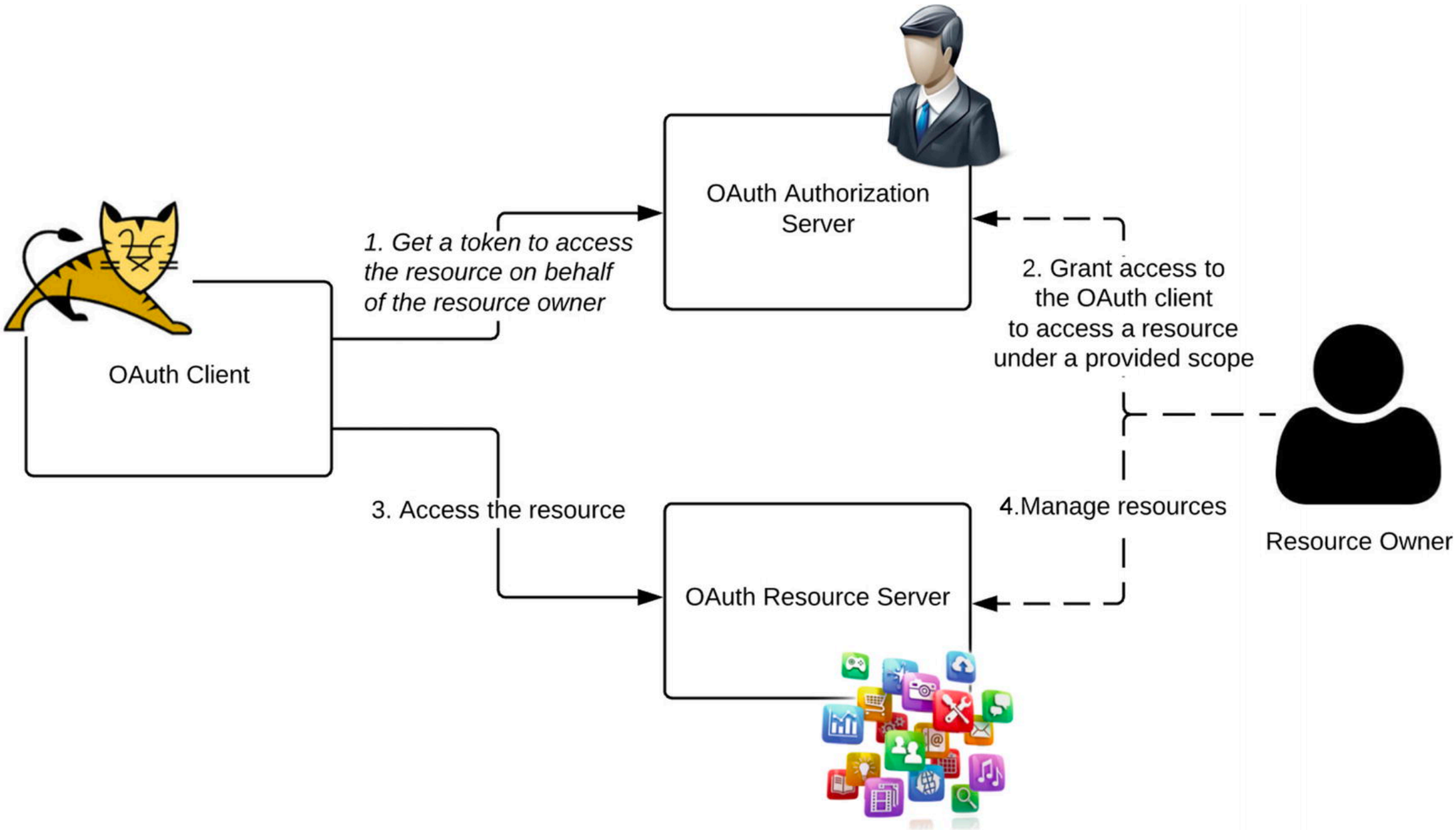
Passing user context as JWT



Problem ?

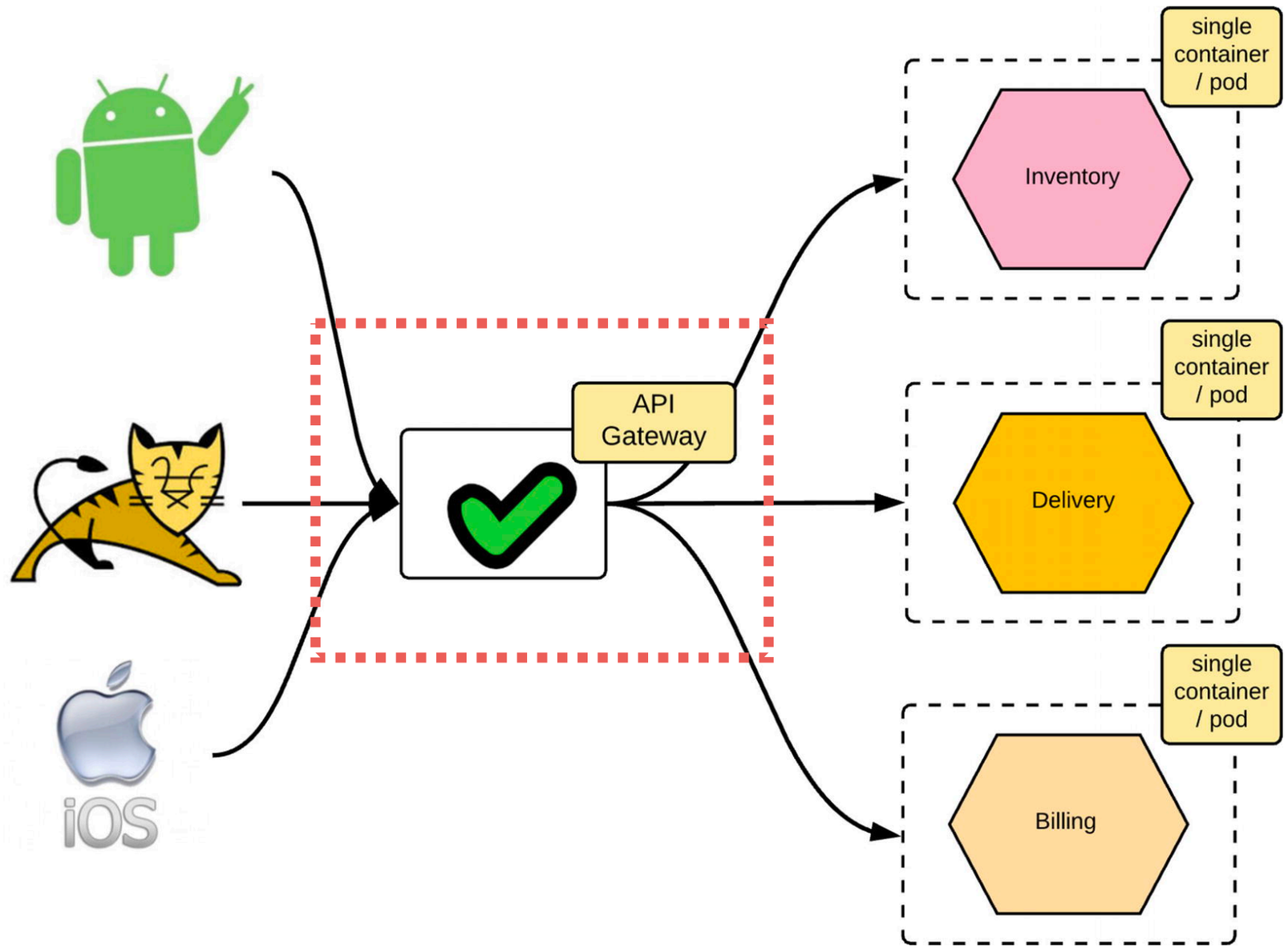


OAuth 2.0



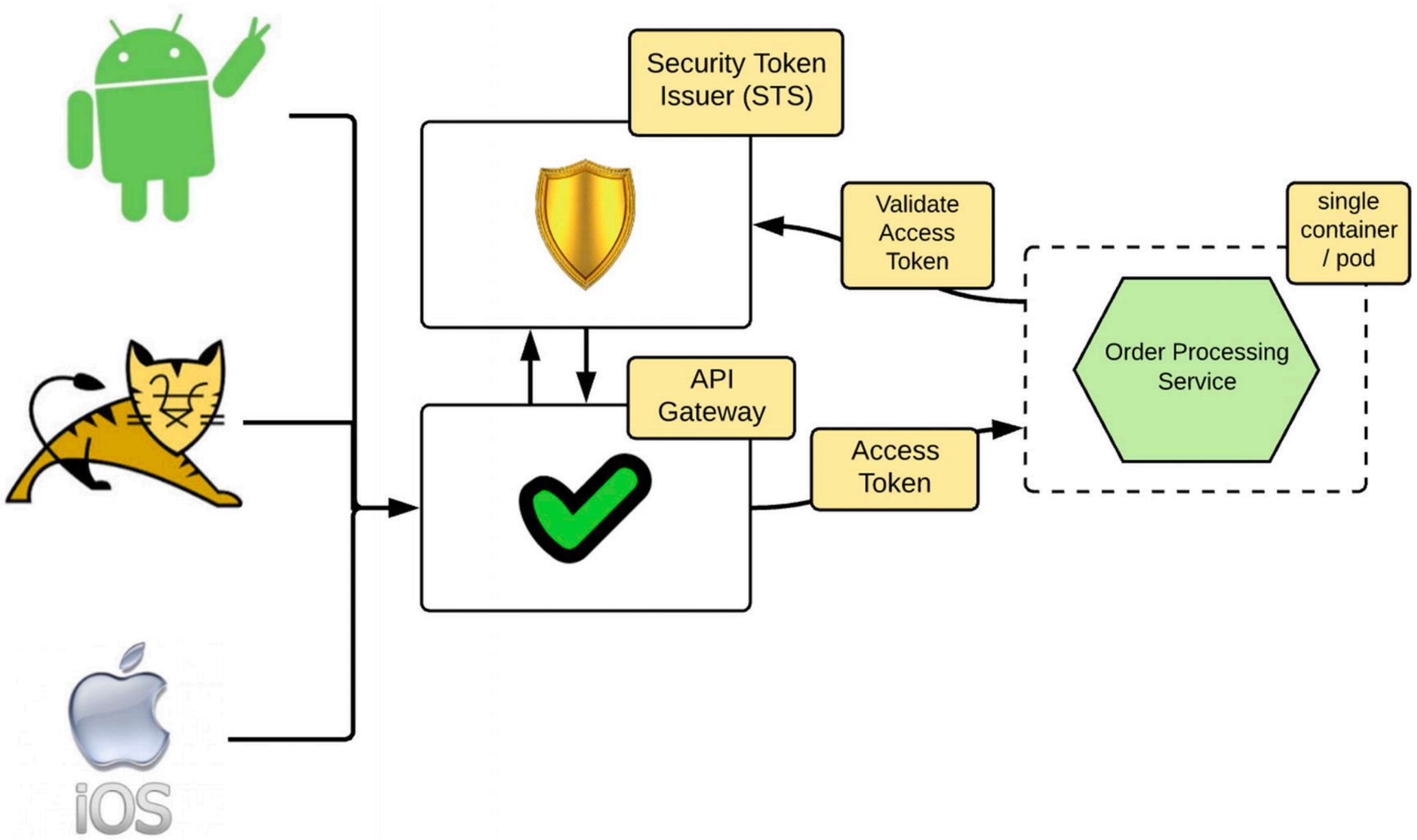
Centralize pattern

Using API gateway pattern

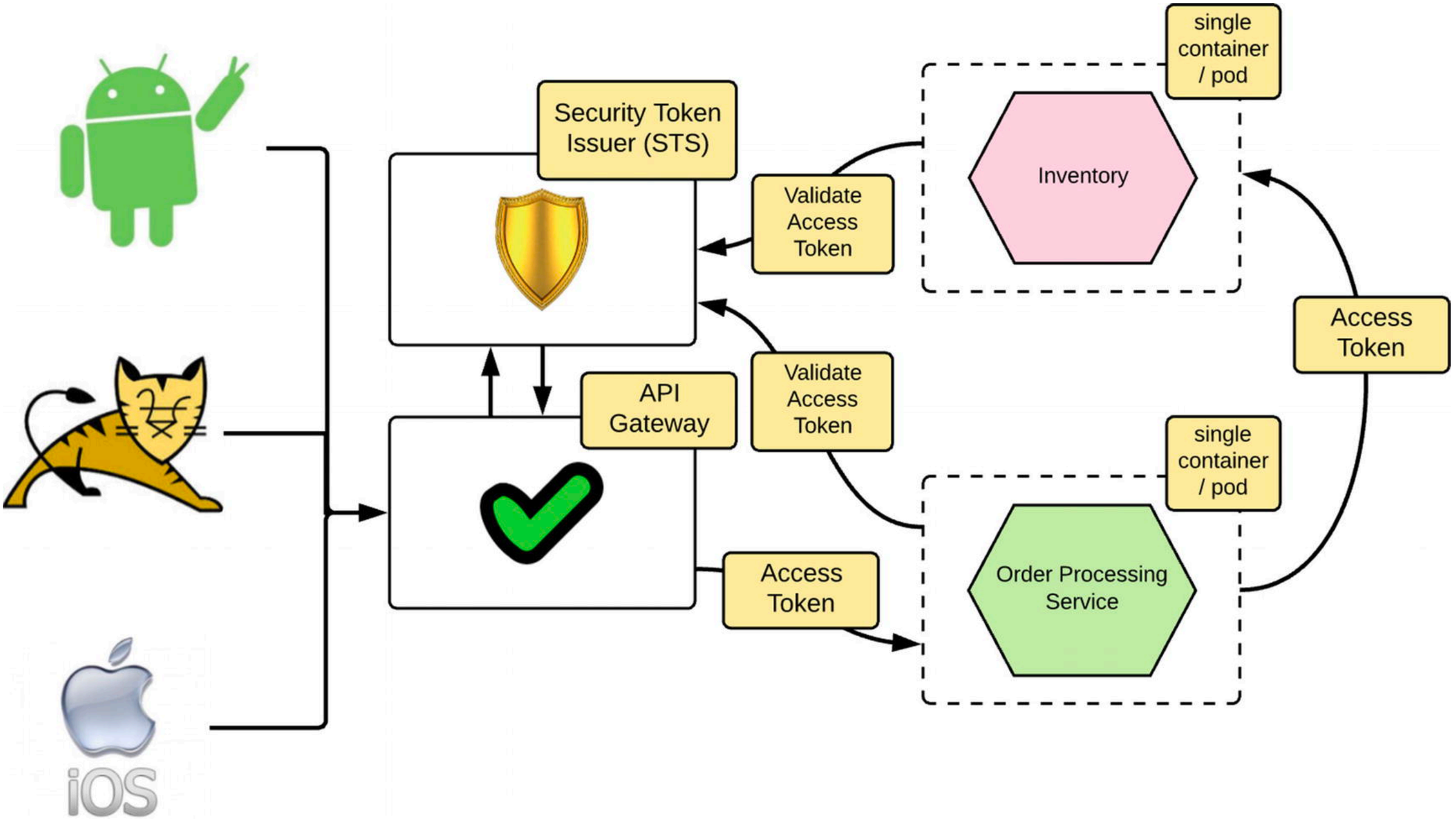


Centralize pattern

Using API gateway pattern

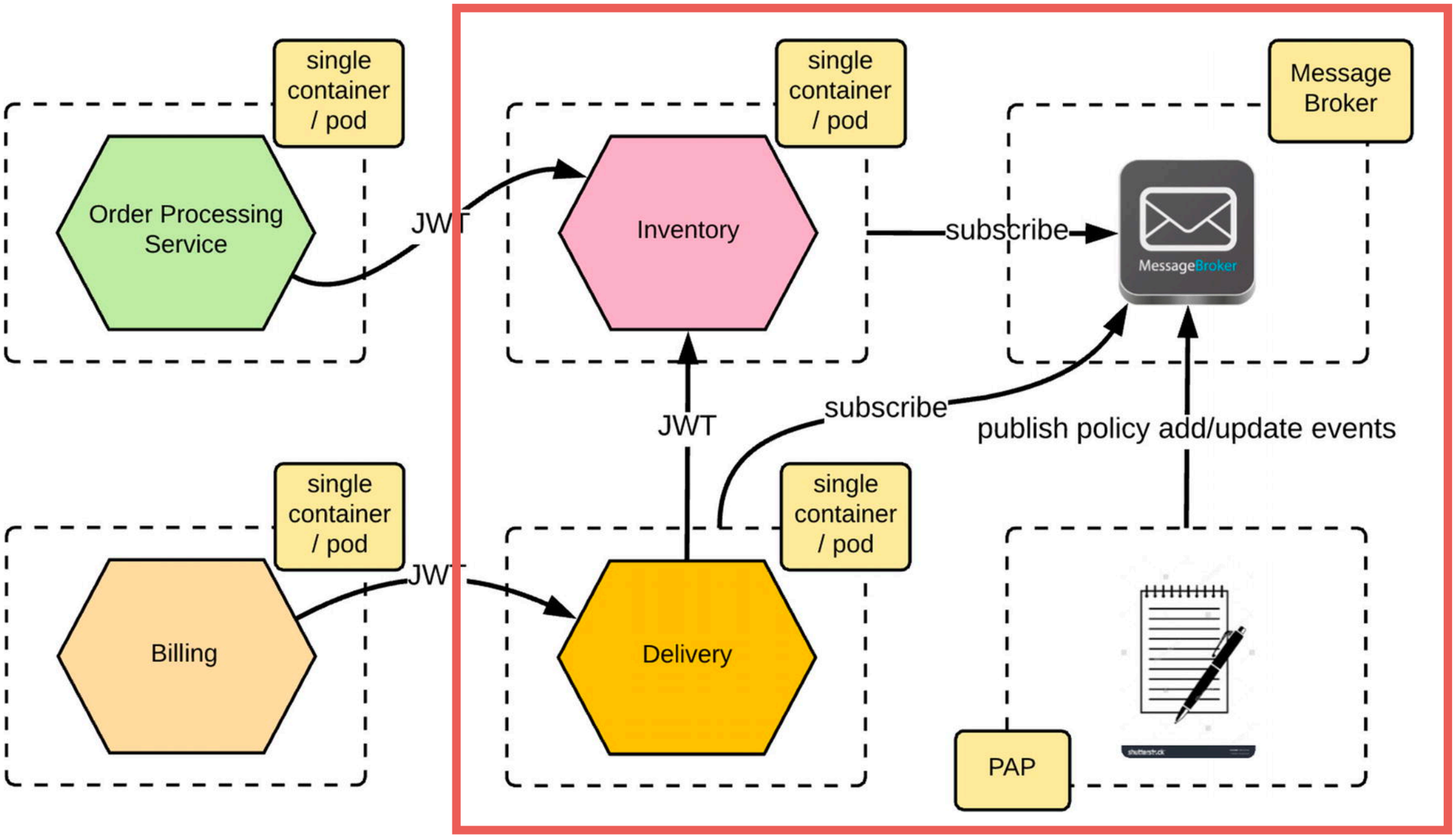


API gateway with OAuth 2.0



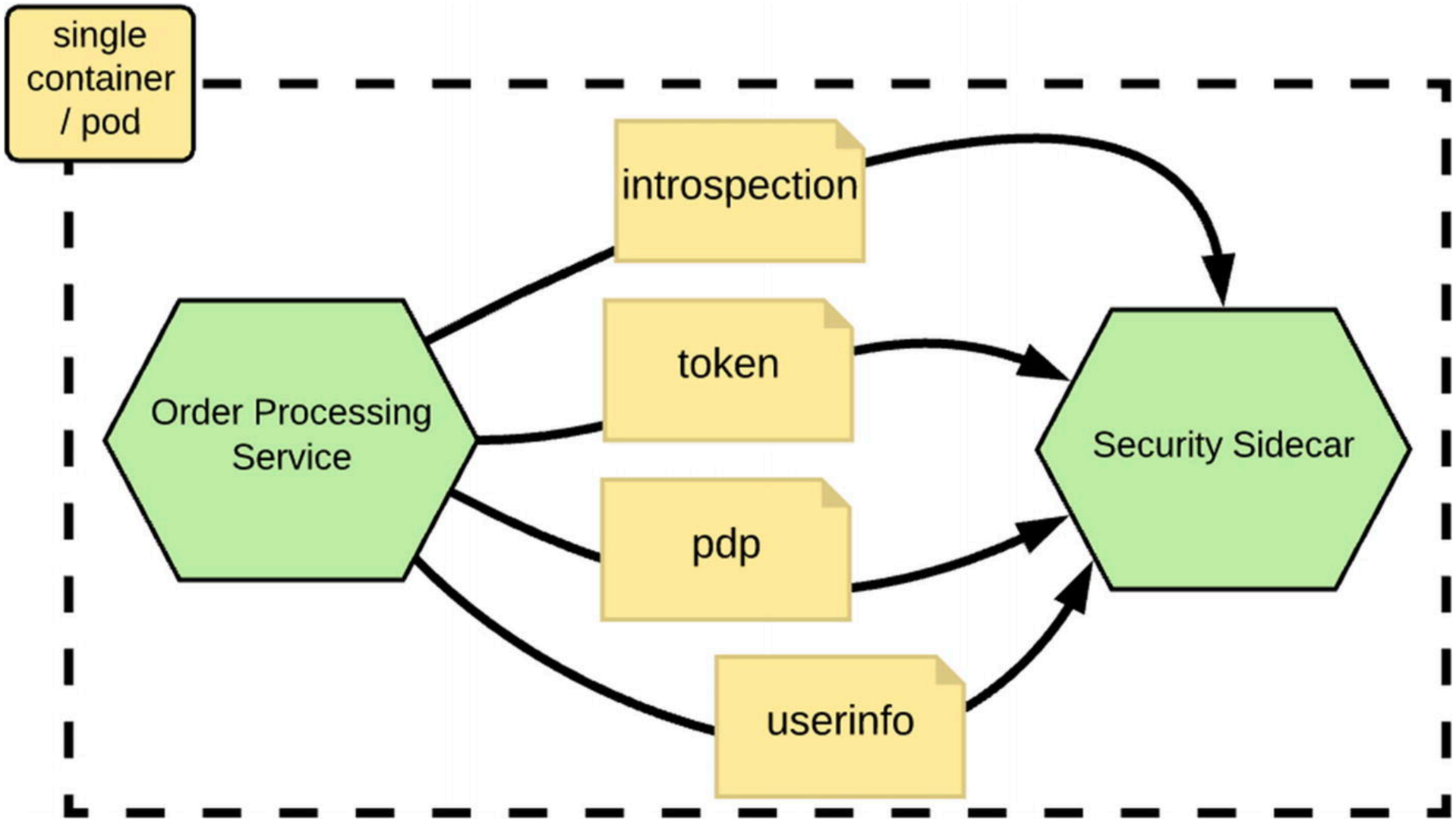
Access control of services

Policy Administration Point

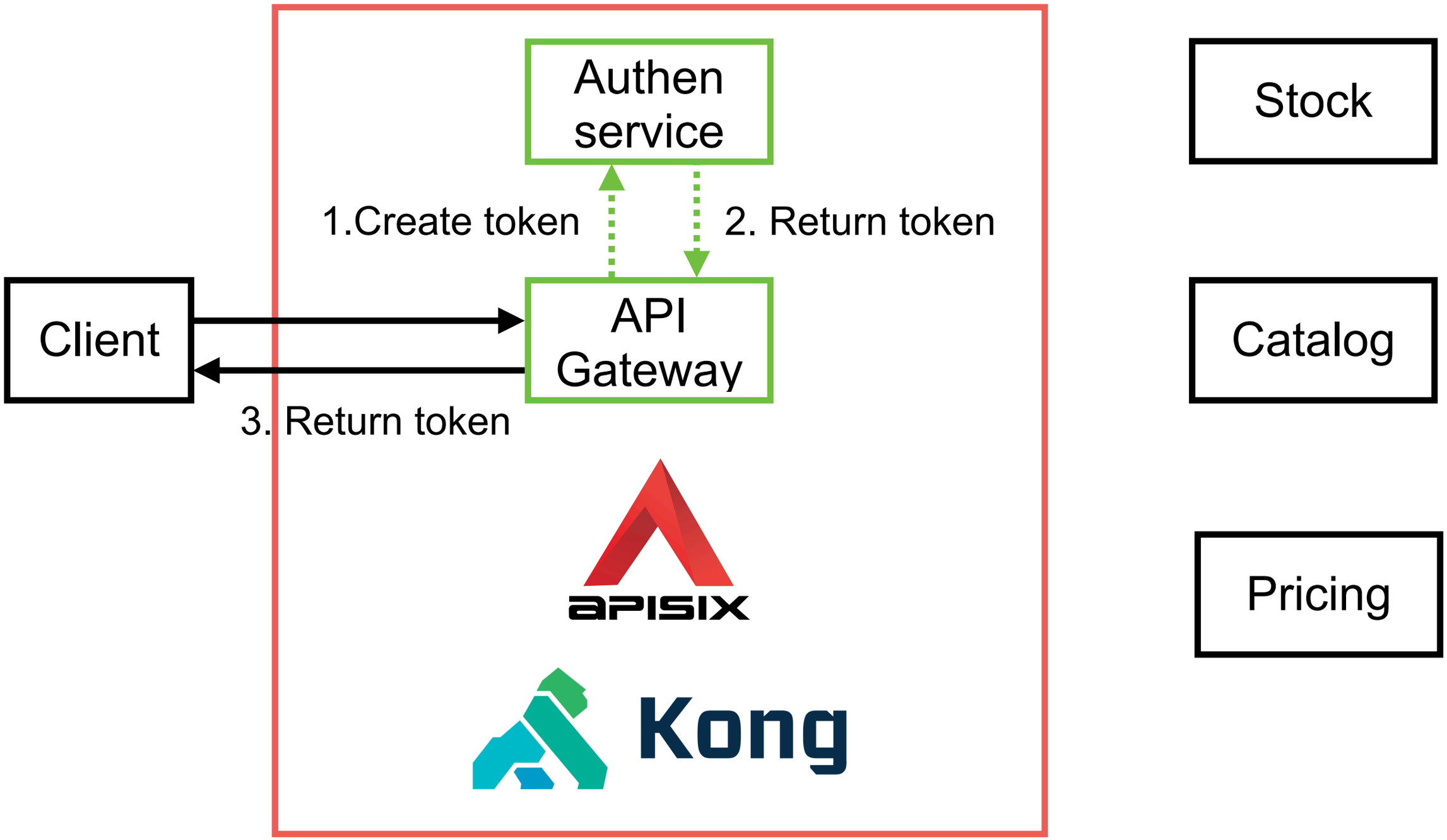


Working sidecar

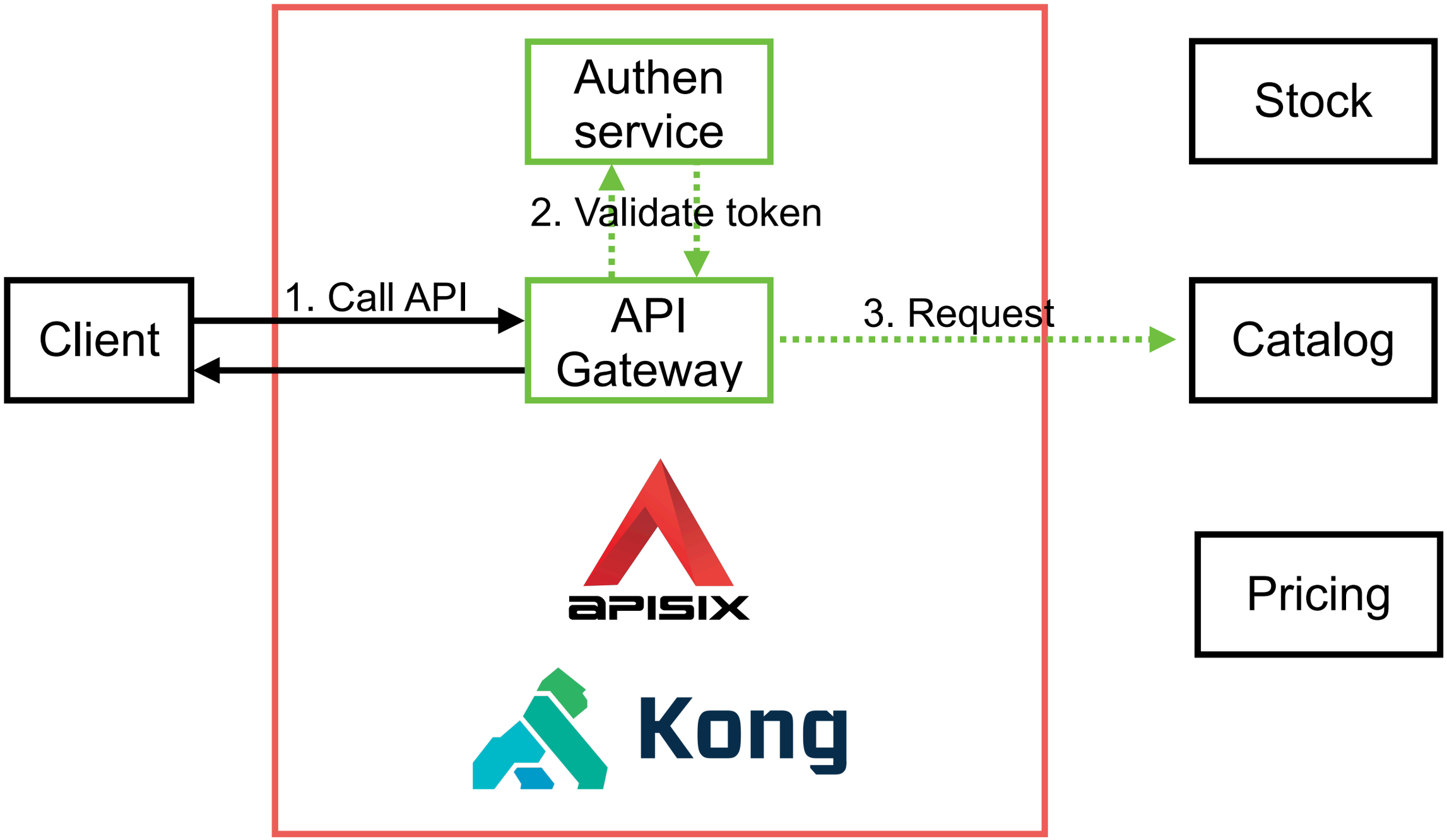
Security sidecar

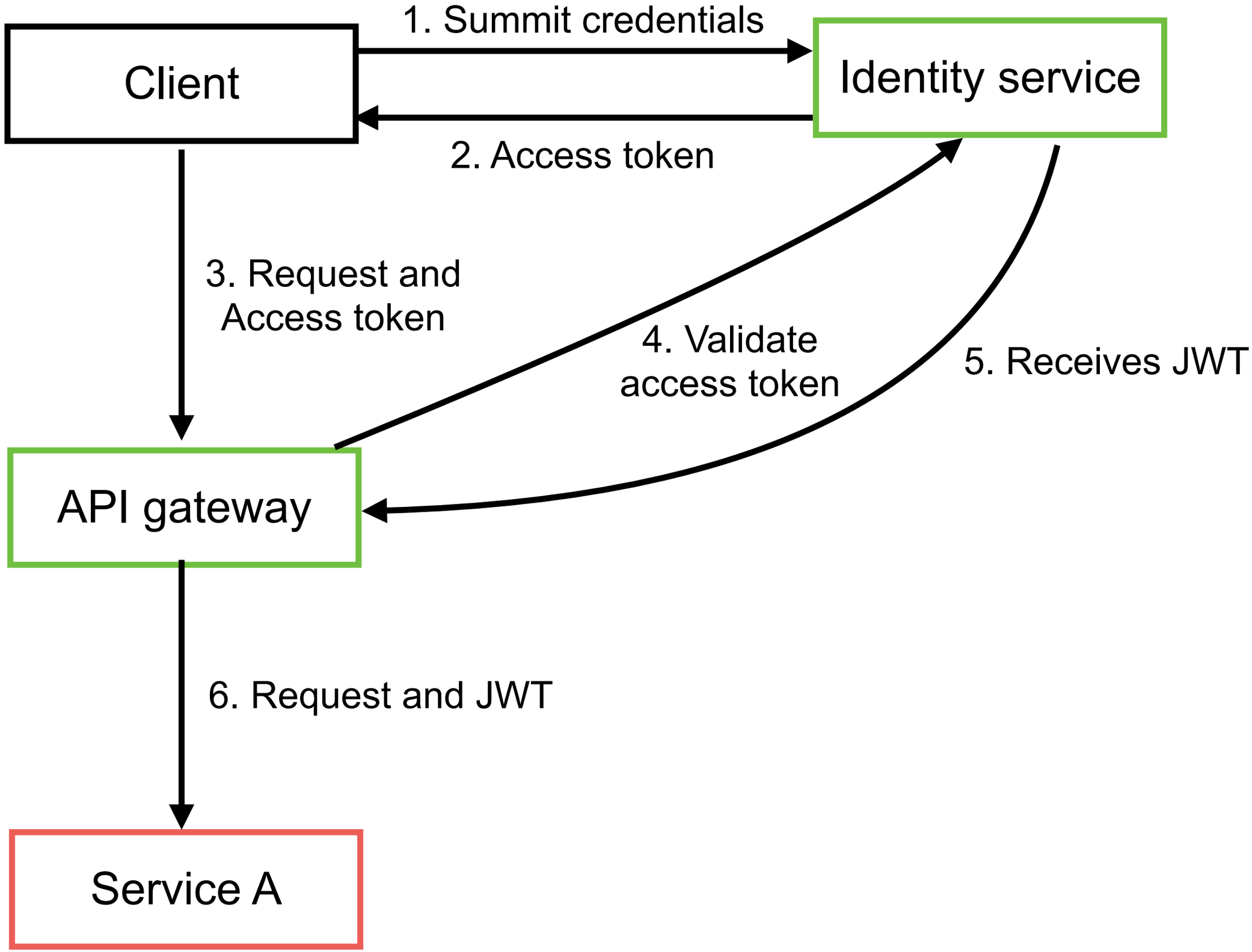


Workshop with Secure service



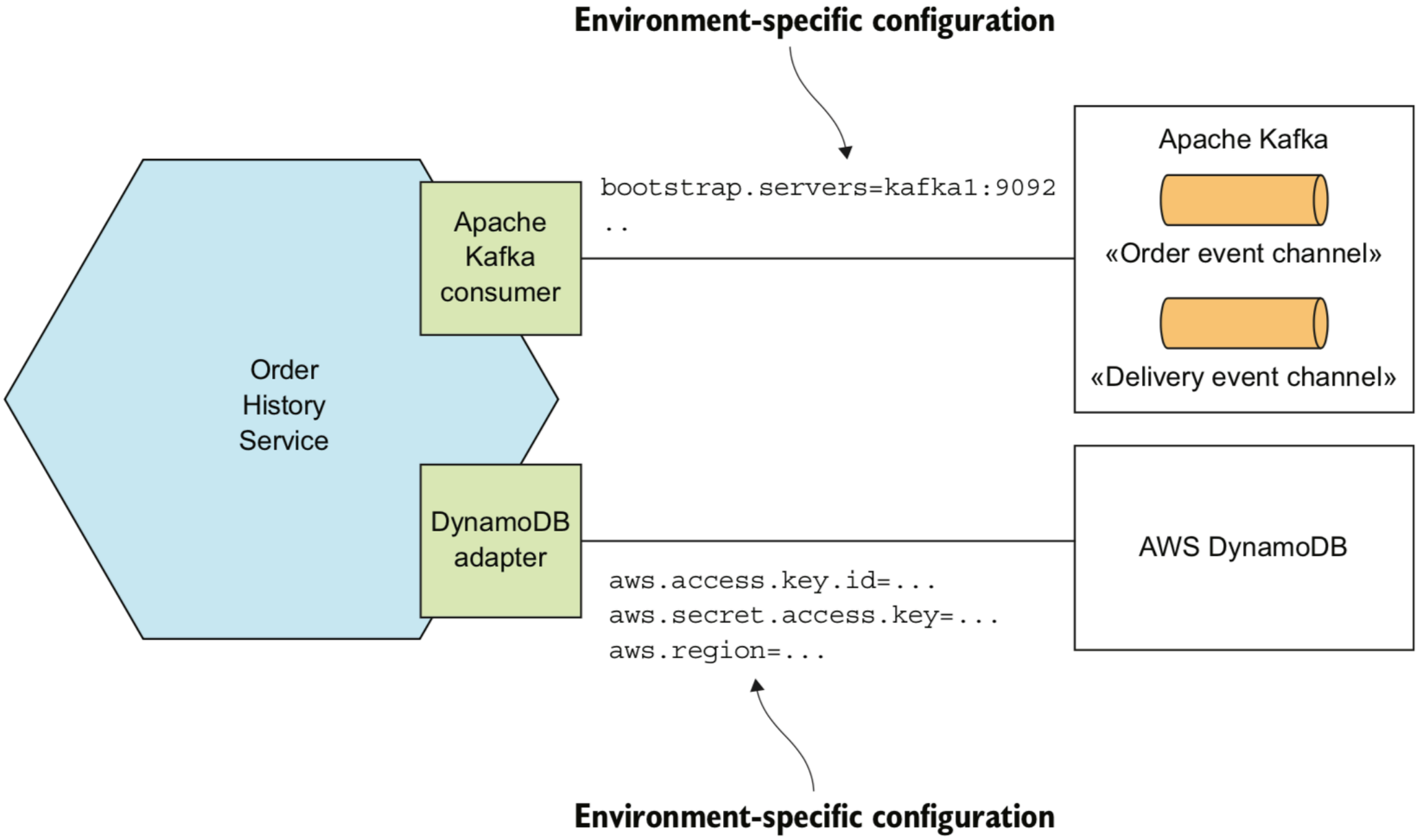
Workshop with Secure service





Configurable services

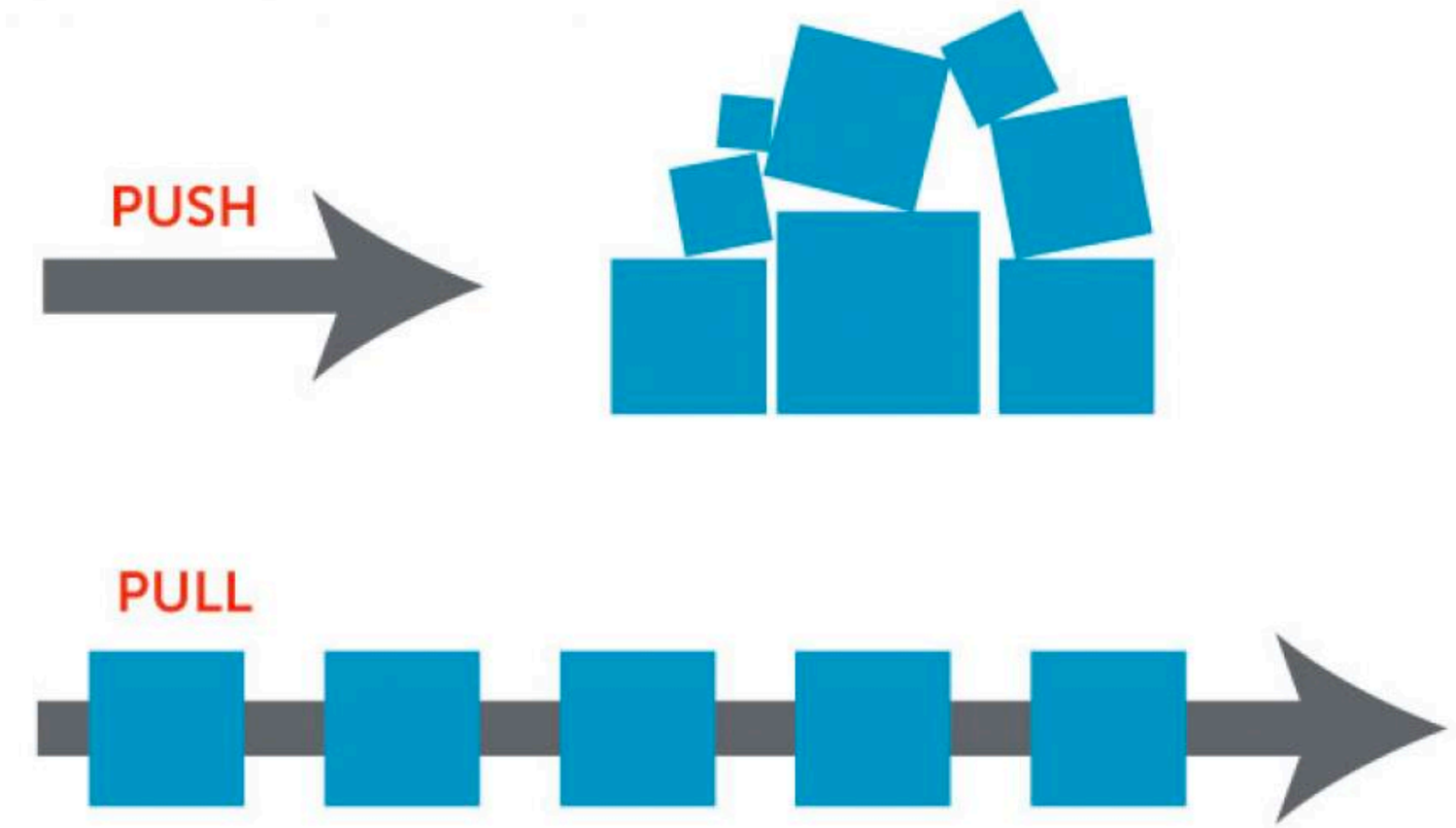
Design configurable services



External configuration models

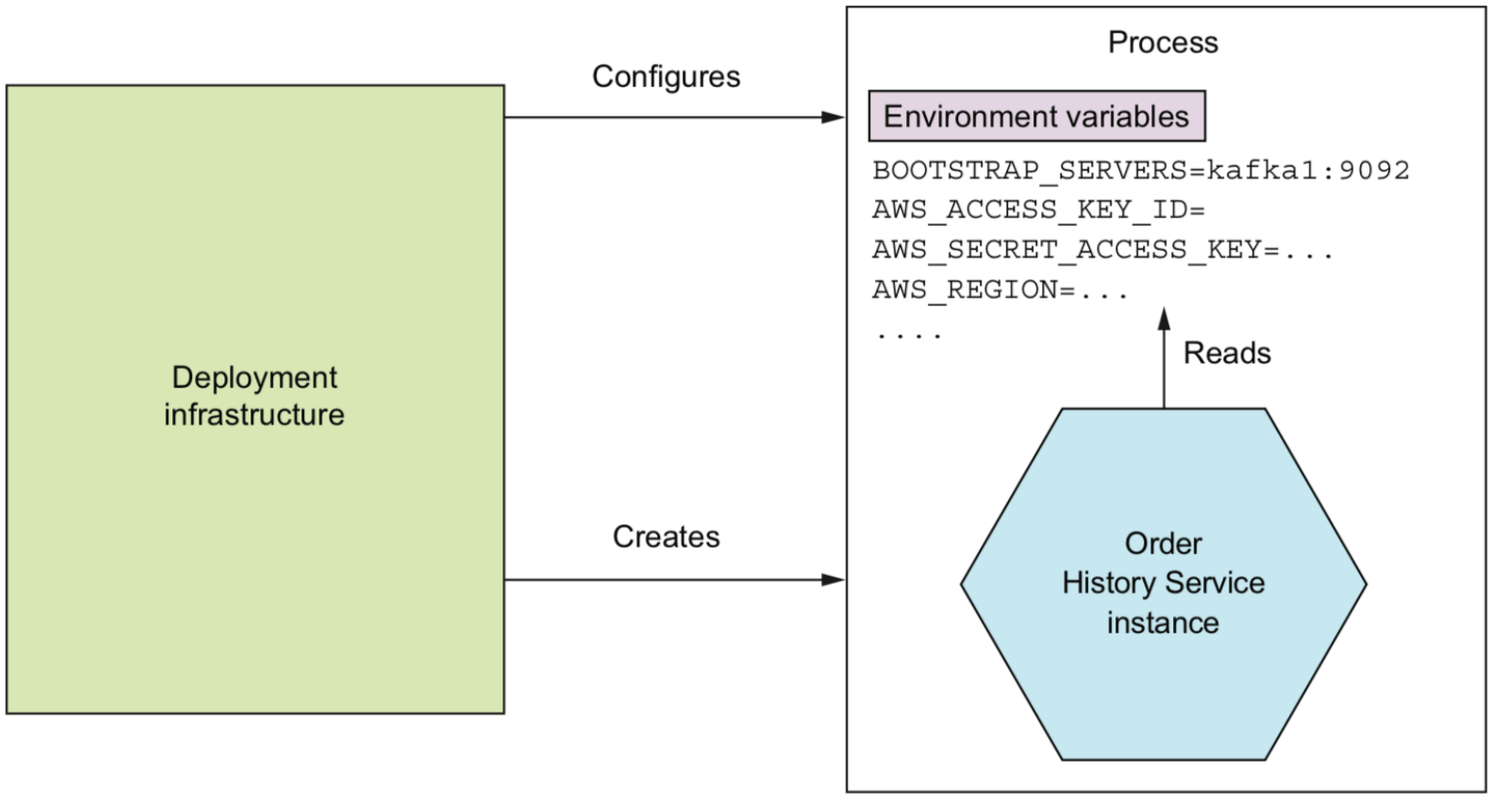
Push model

Pull model



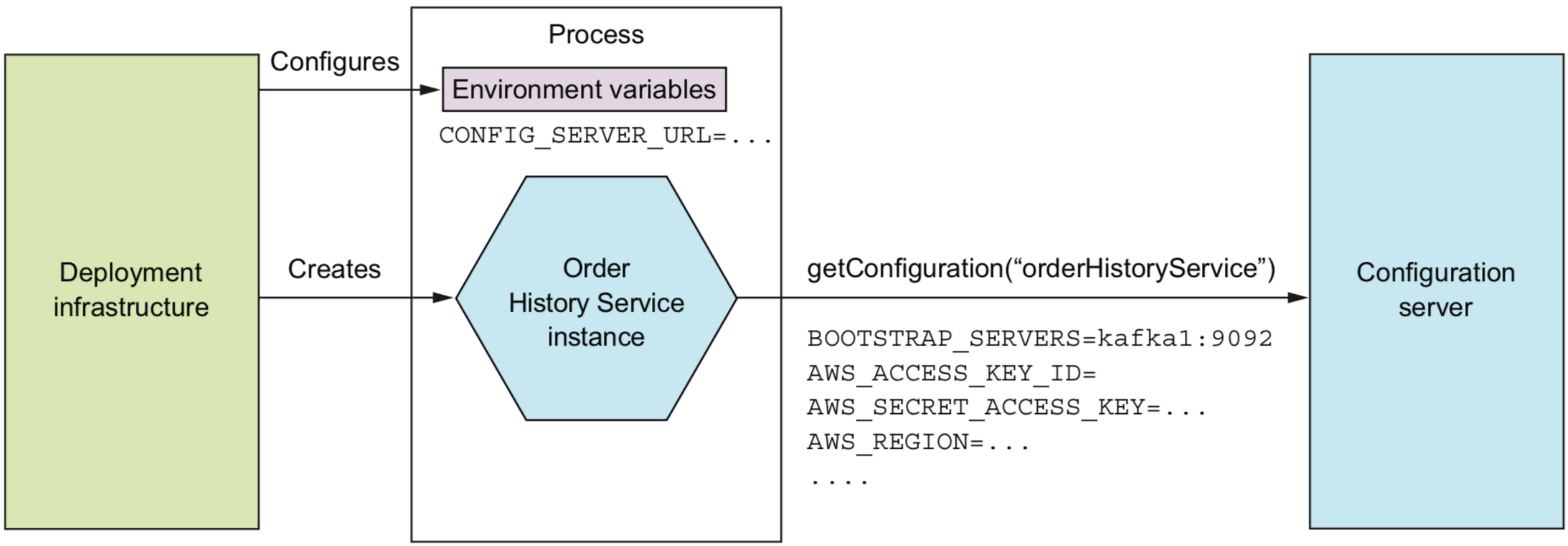
Push model

Pass the configuration to service
OS environment variables
Configuration files



Pull model

Service read configuration from configuration server

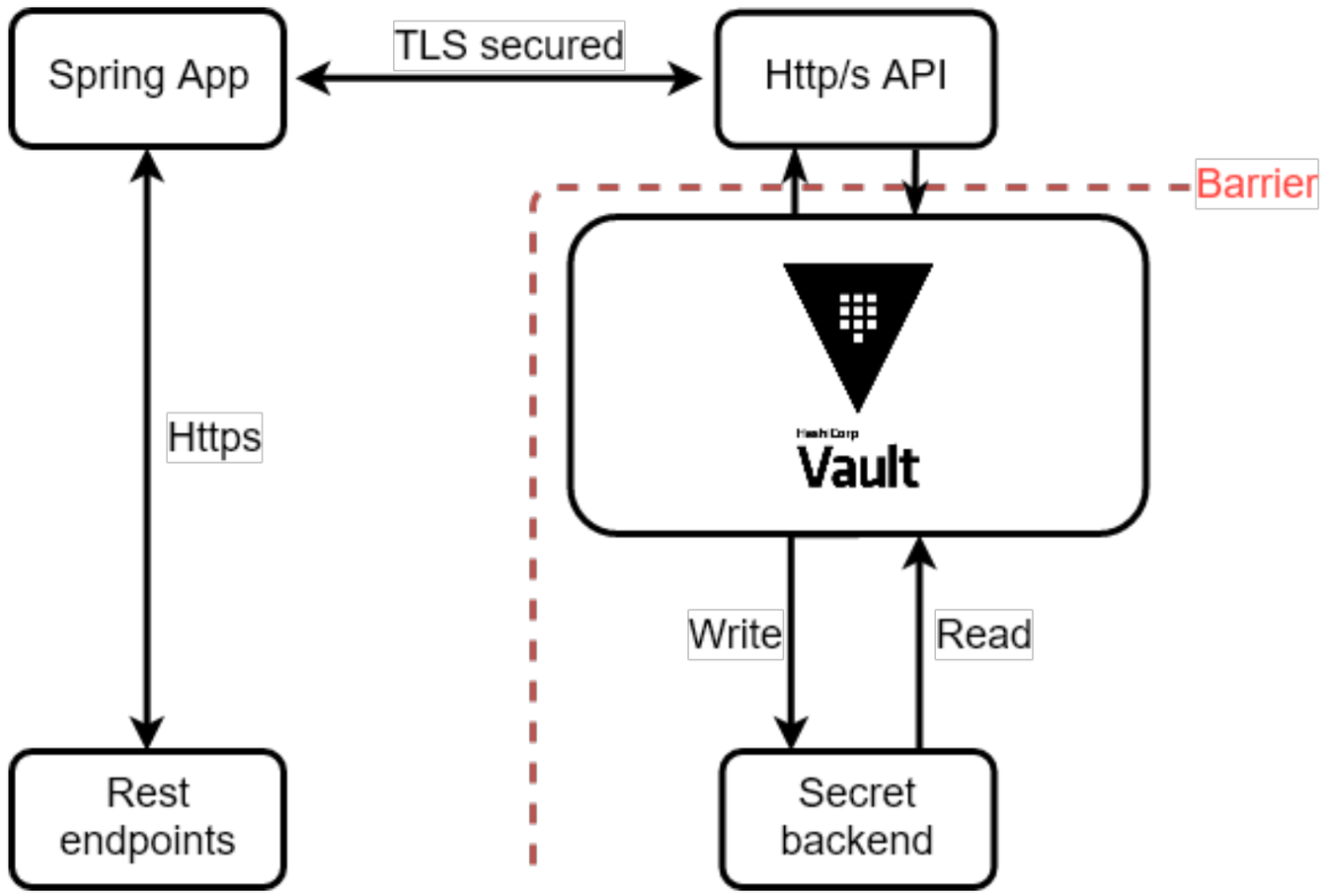


Benefits of configuration server

Centralized configuration
Transparent decryption of sensitive data
Dynamic reconfiguration

Workshop with Configuration

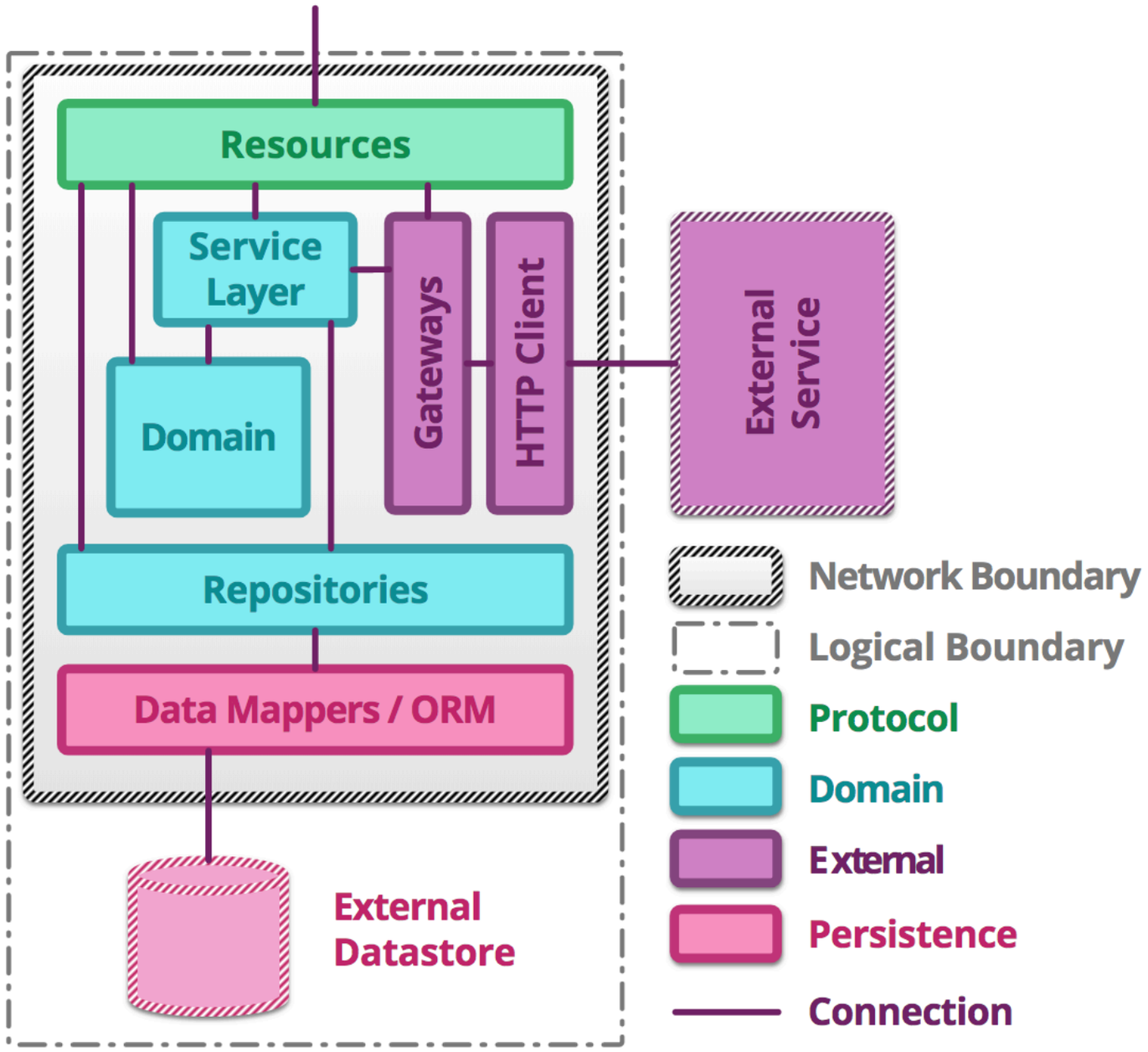
Plain text
Secure data



https://www.vaultproject.io/

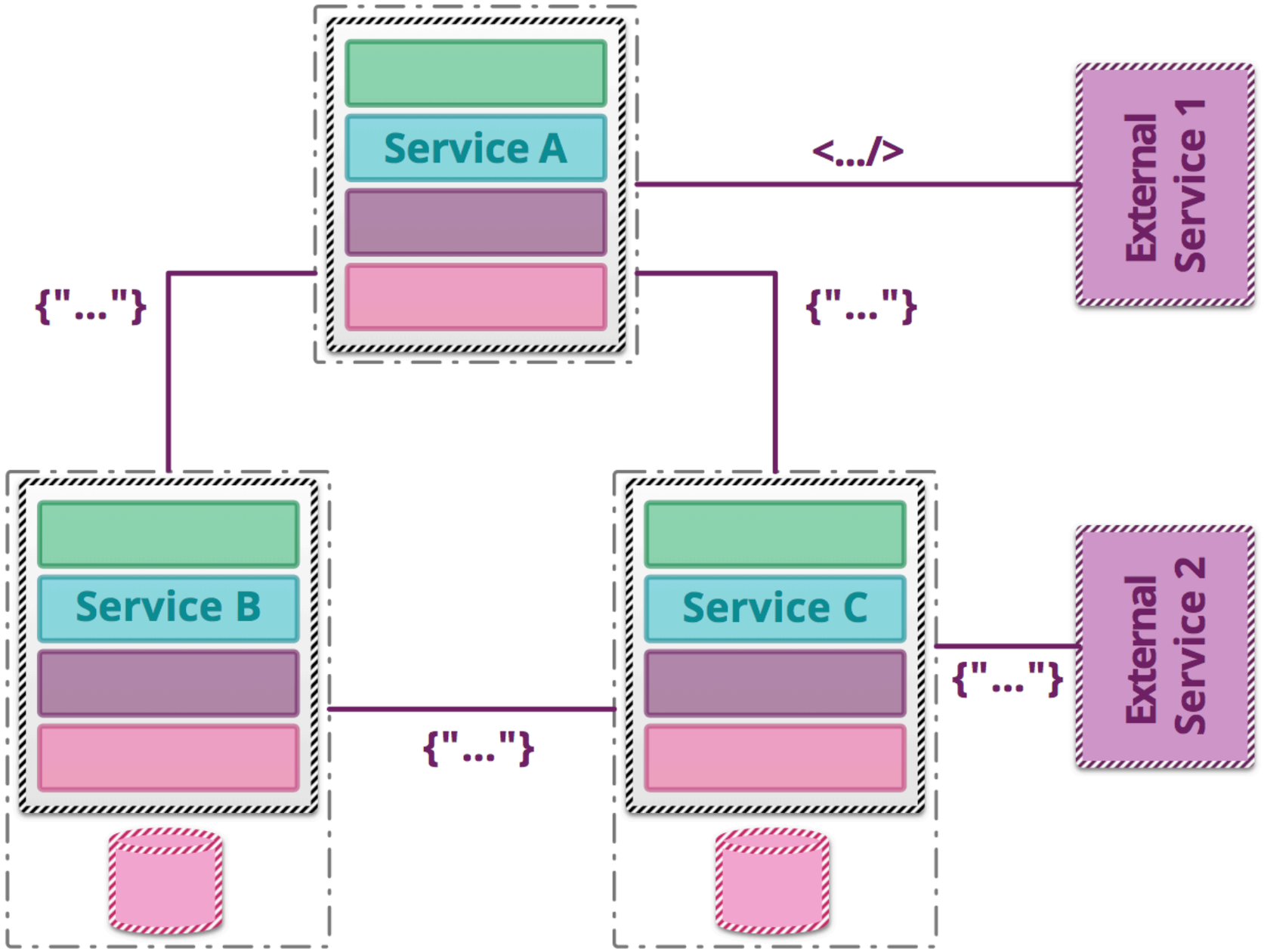
Microservice Testing ?

Service structure

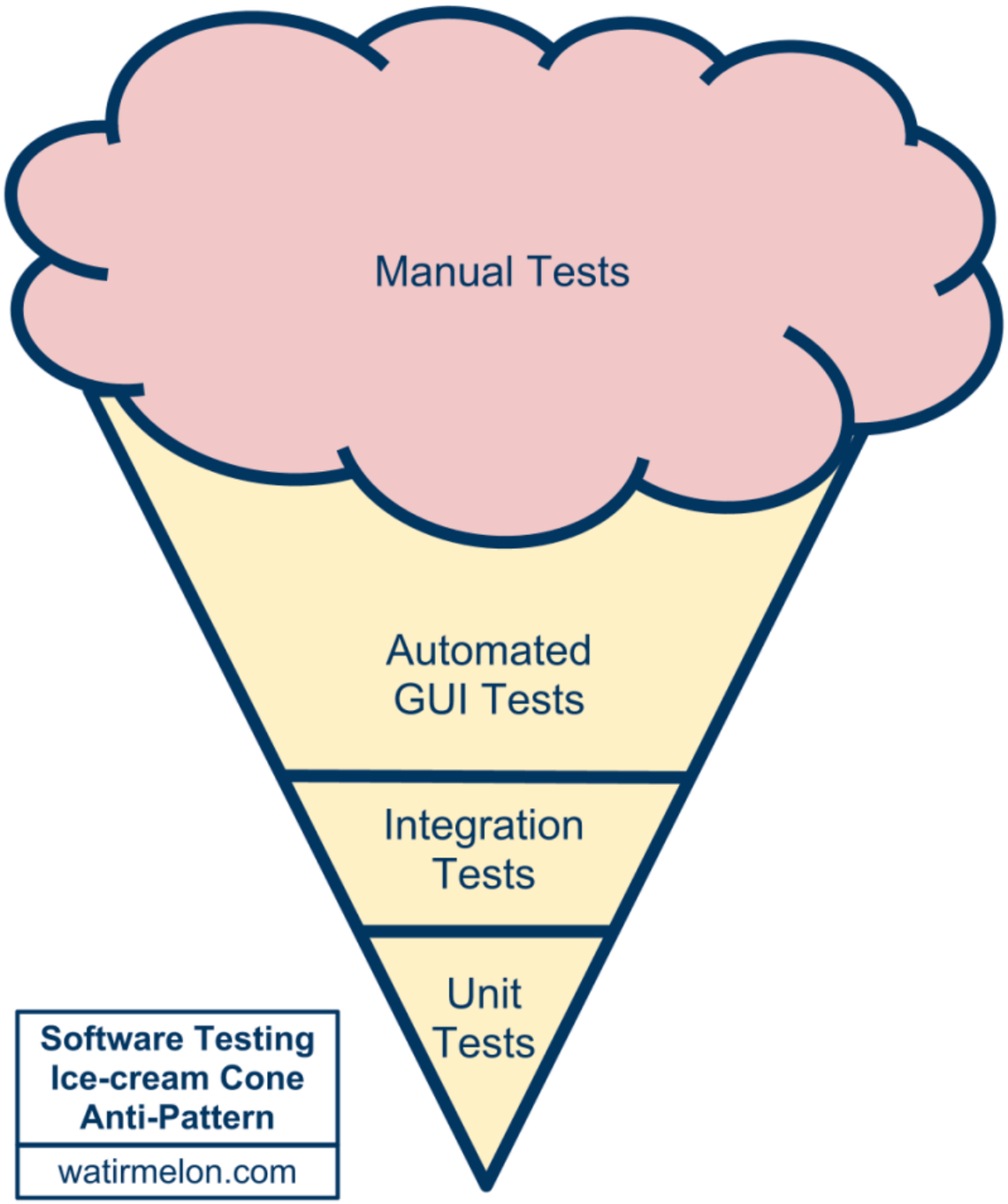


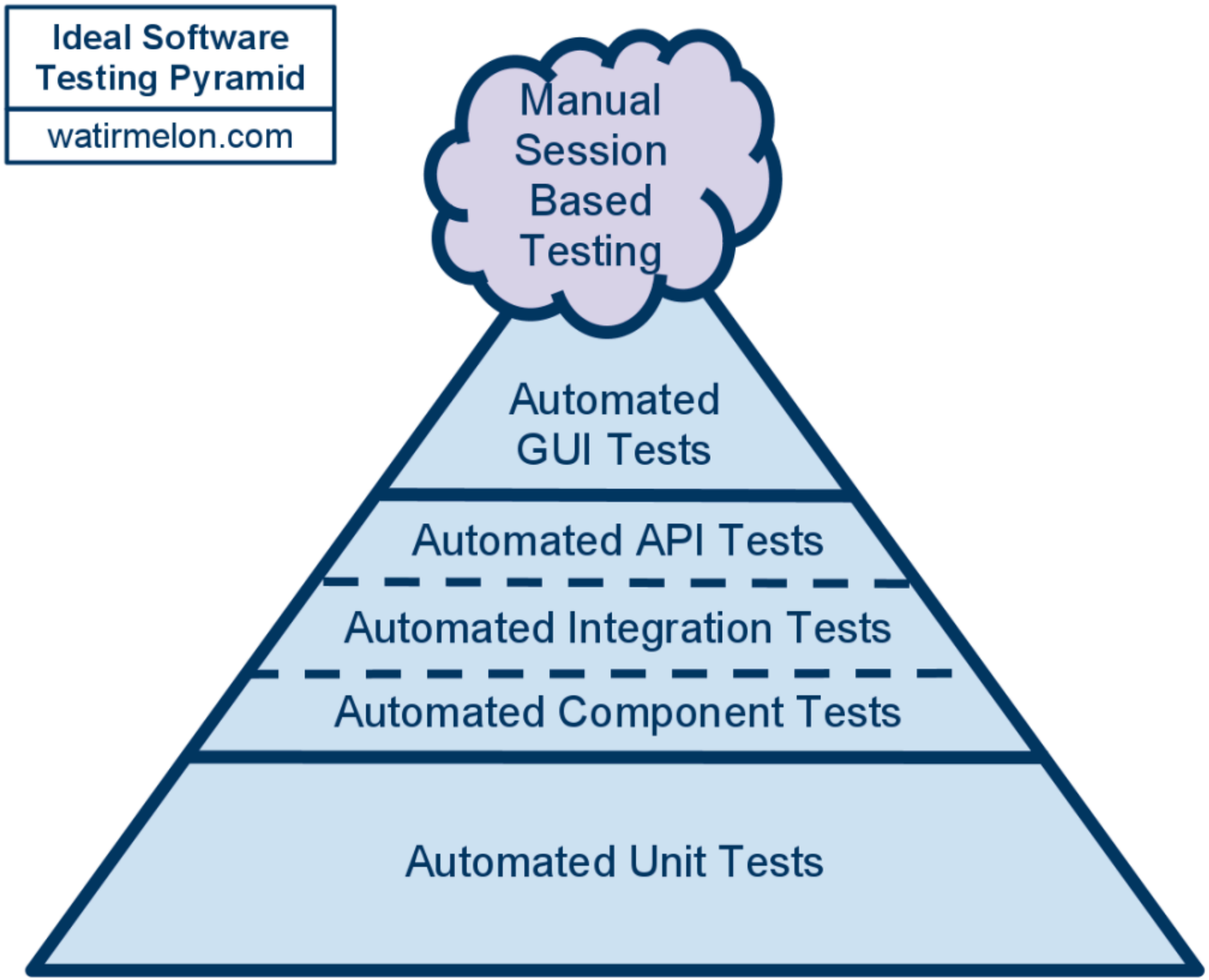
https://martinfowler.com/articles/microservice-testing

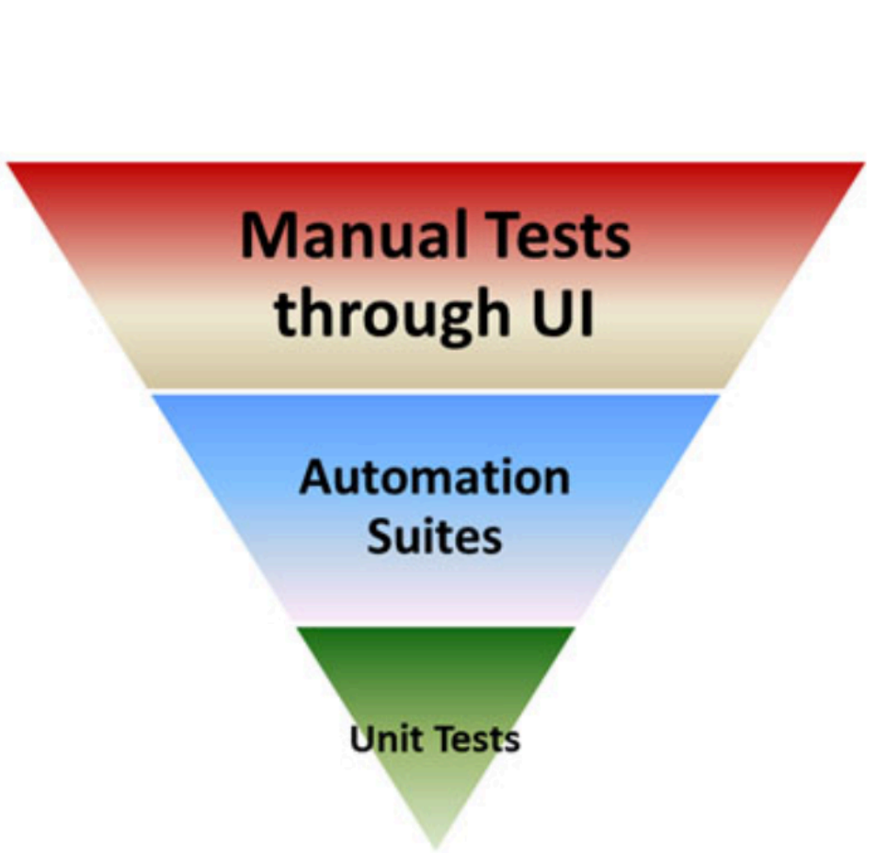
Multiple services



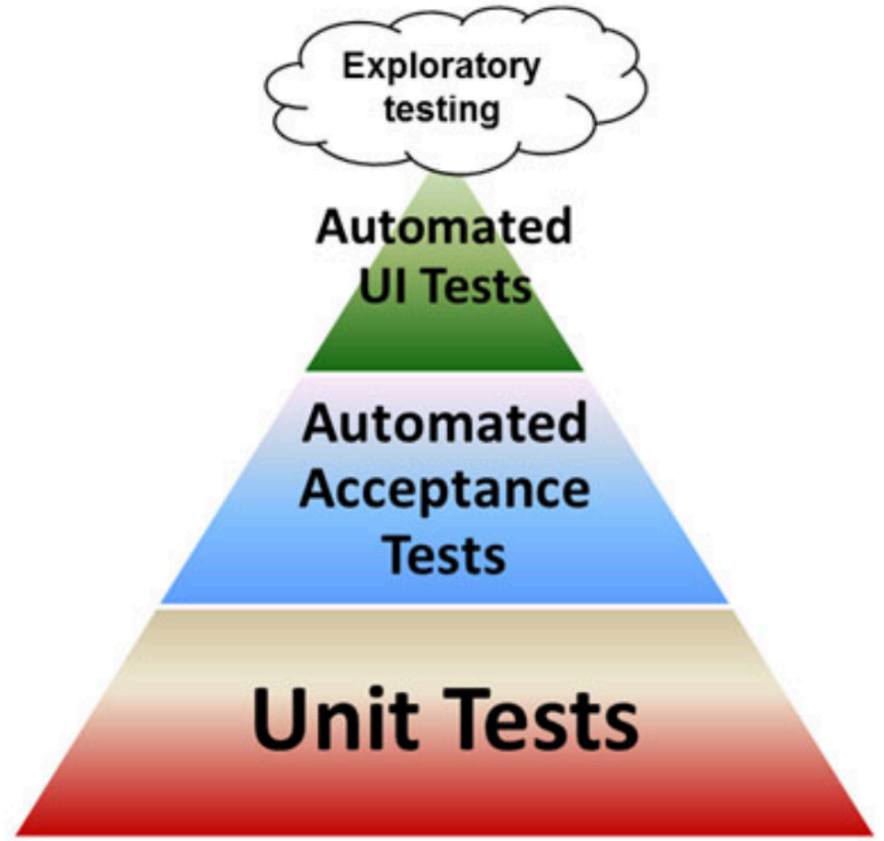
https://martinfowler.com/articles/microservice-testing





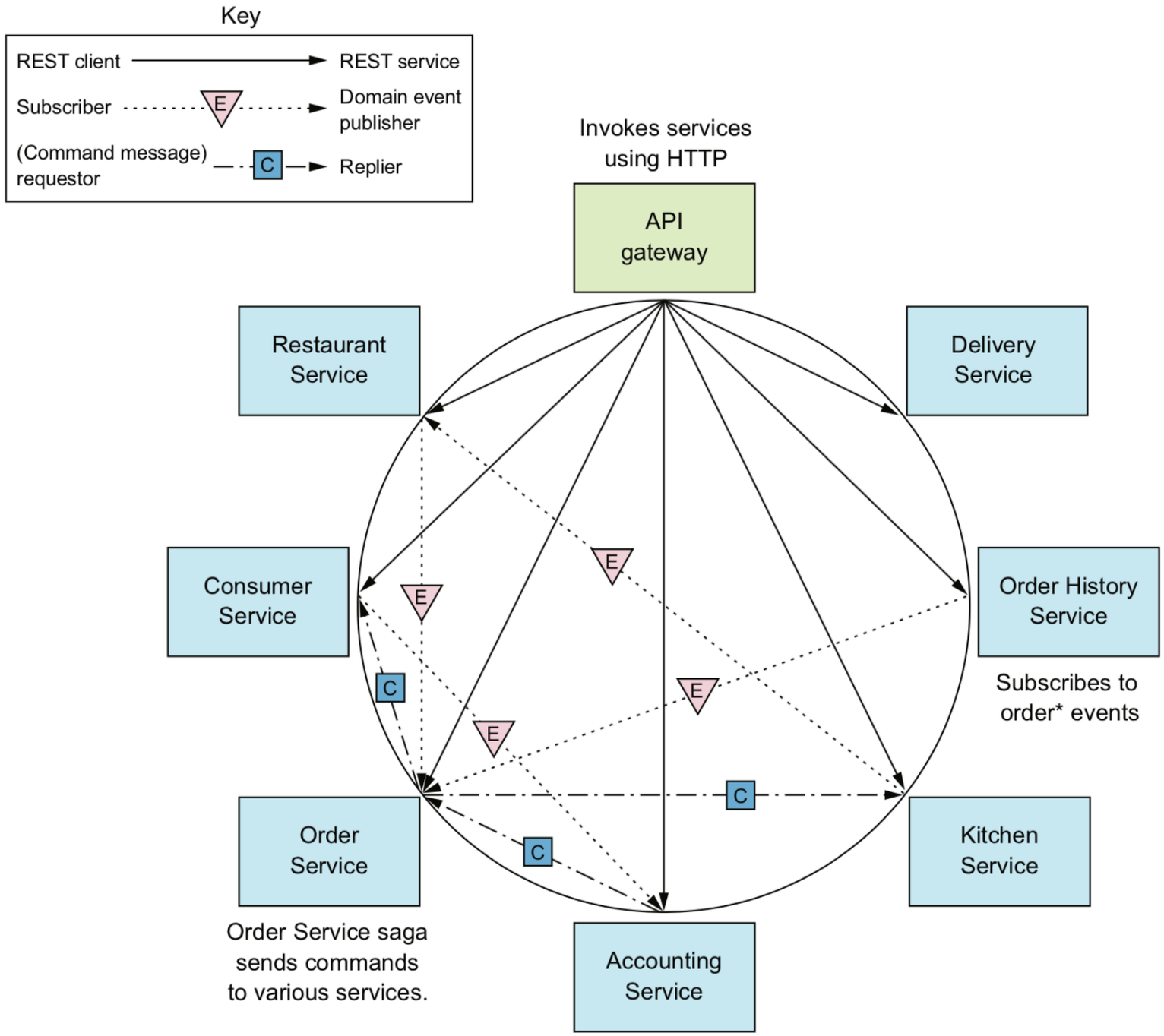


Traditional
(find bugs)

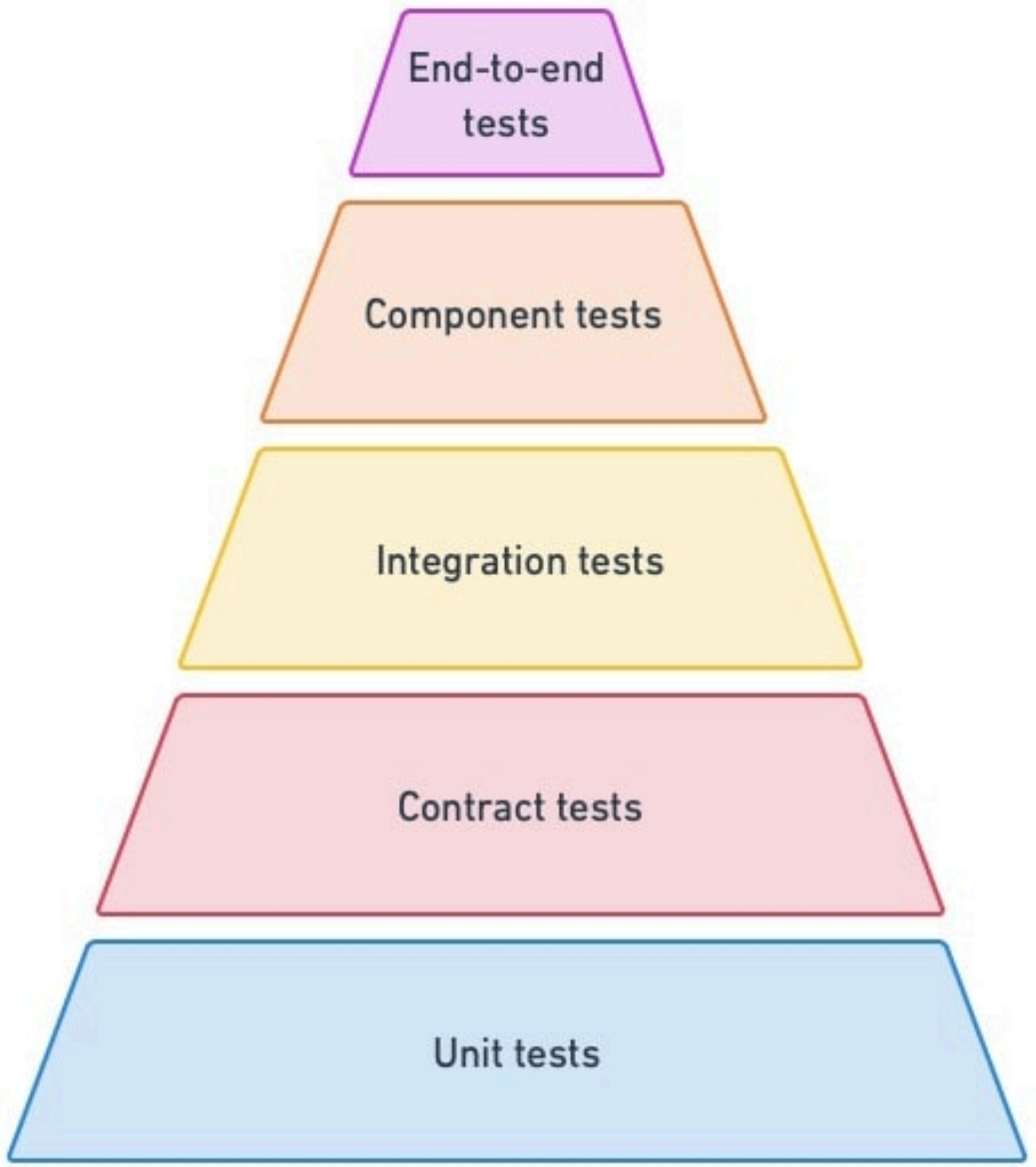


Agile
(prevent bugs)

Testing in Microservice ?

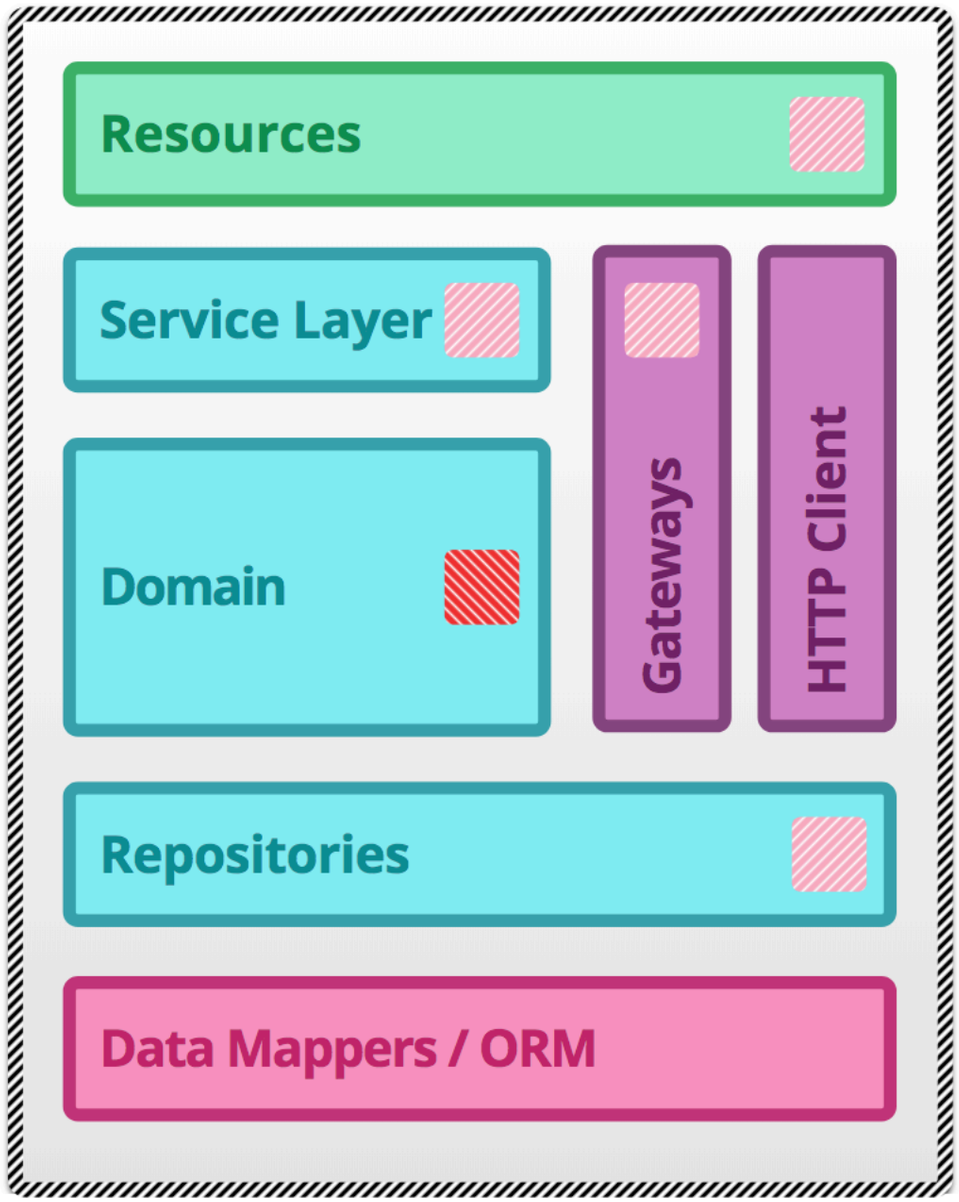


Microservice Testing



https://semaphoreci.com/blog/test-microservices

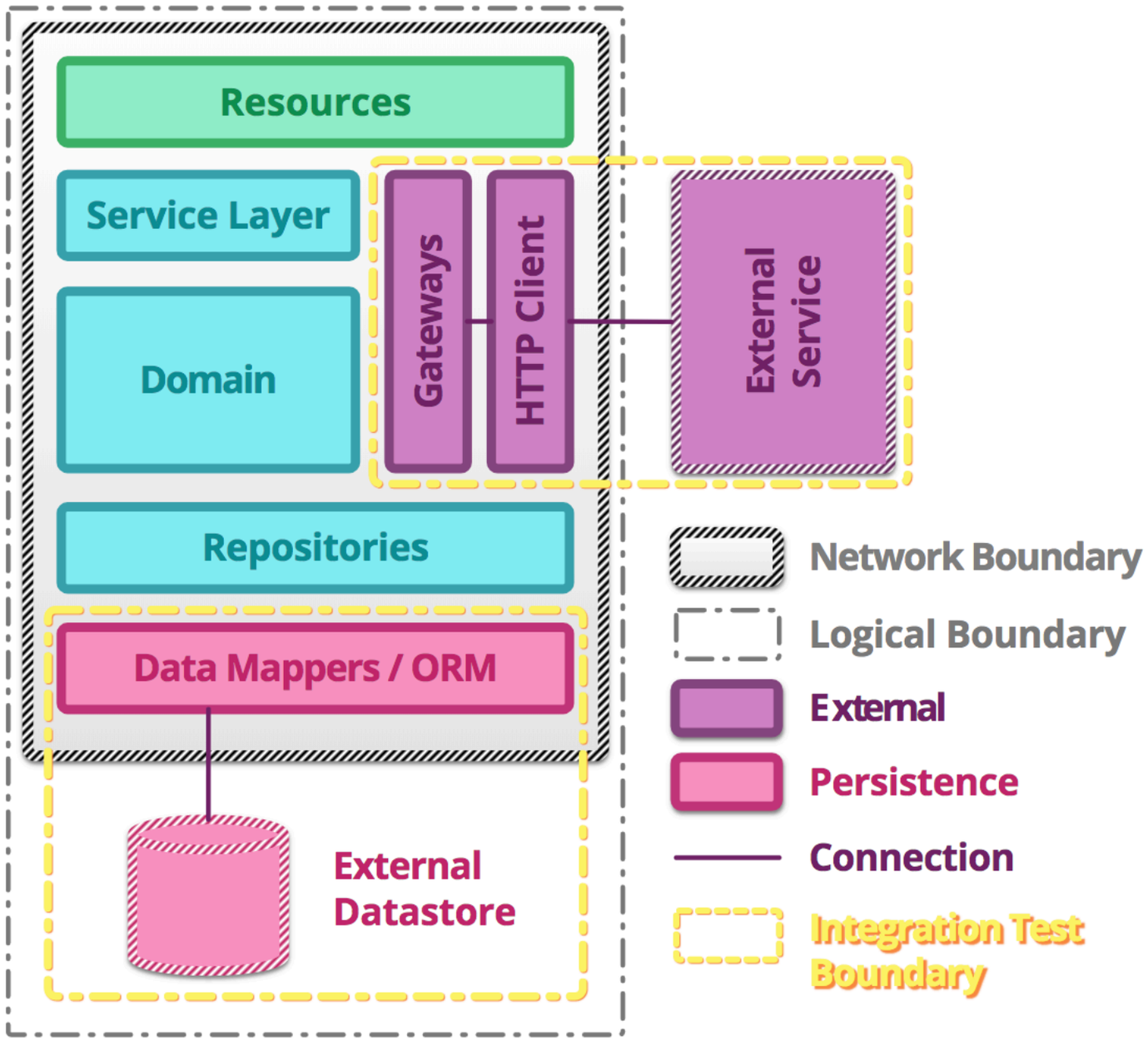
Unit testing



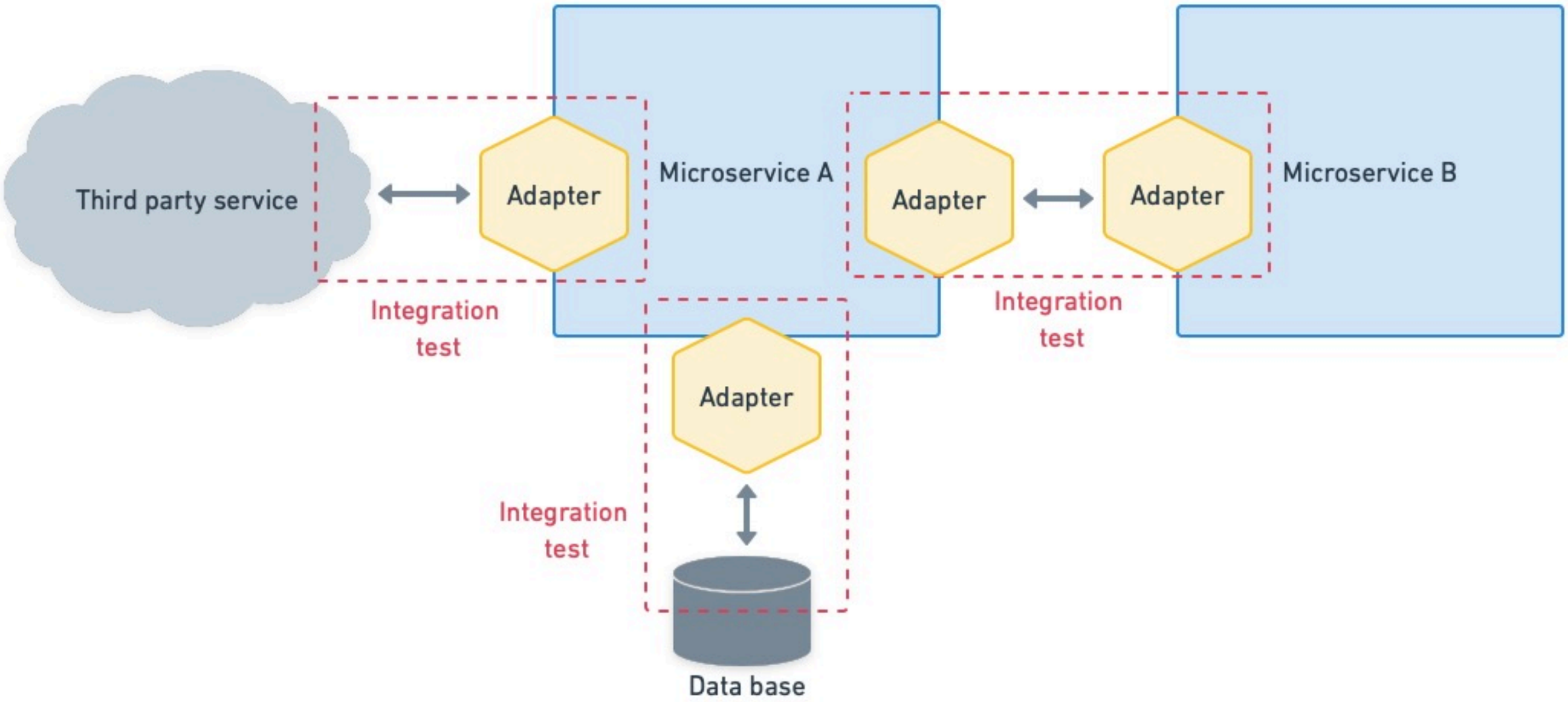
Unit - Solitary

Unit - Sociable

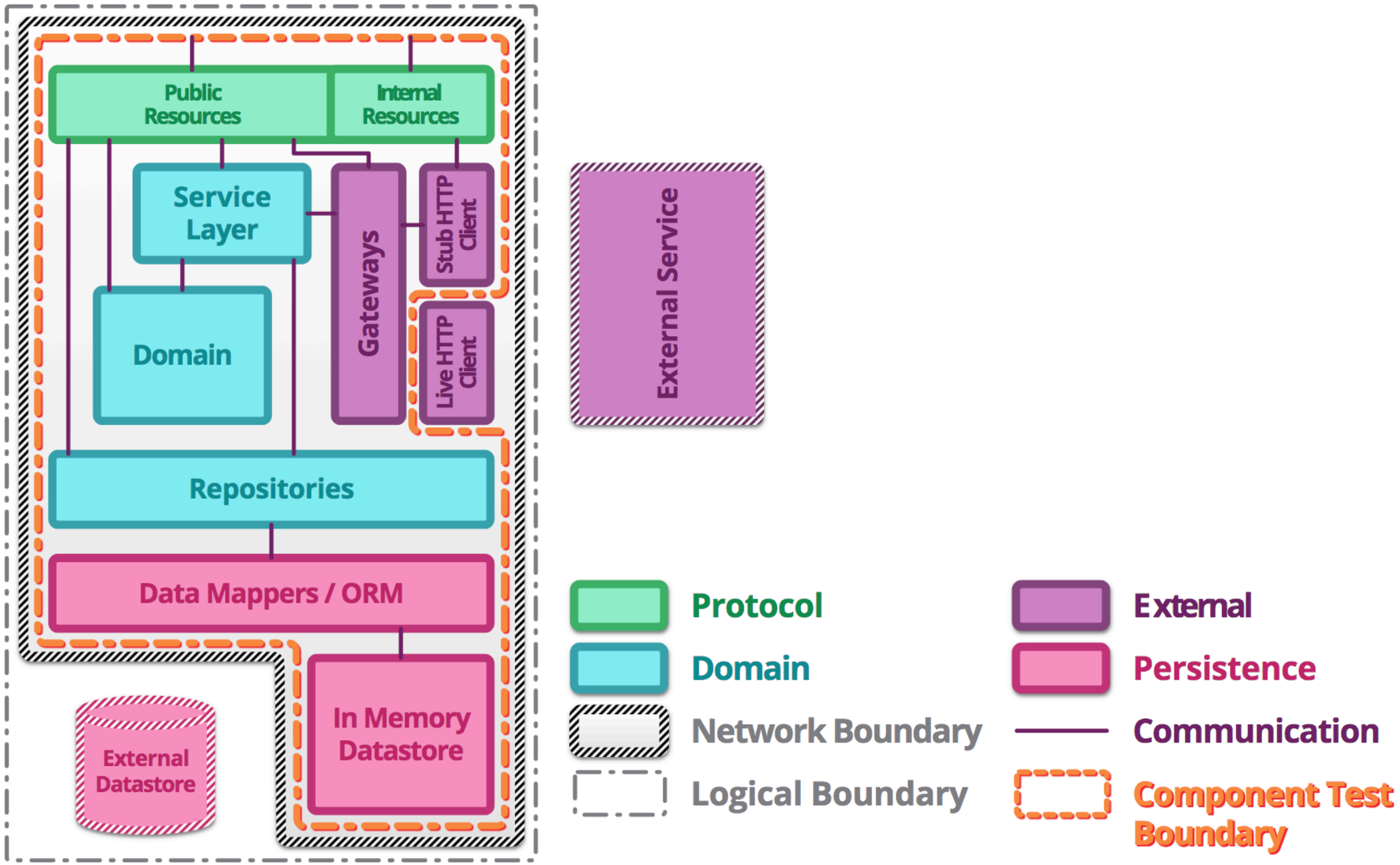
Integration testing



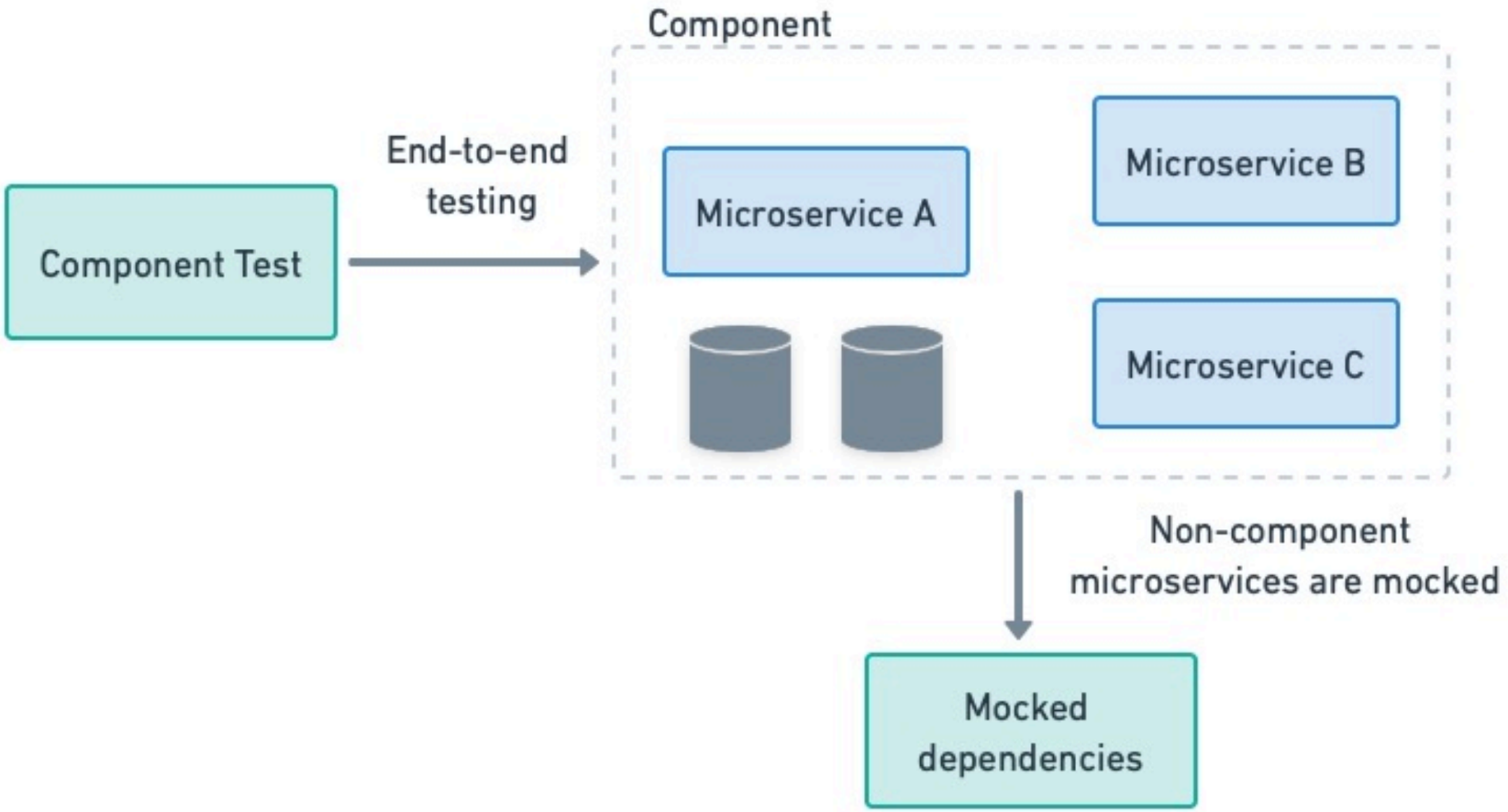
Integration testing



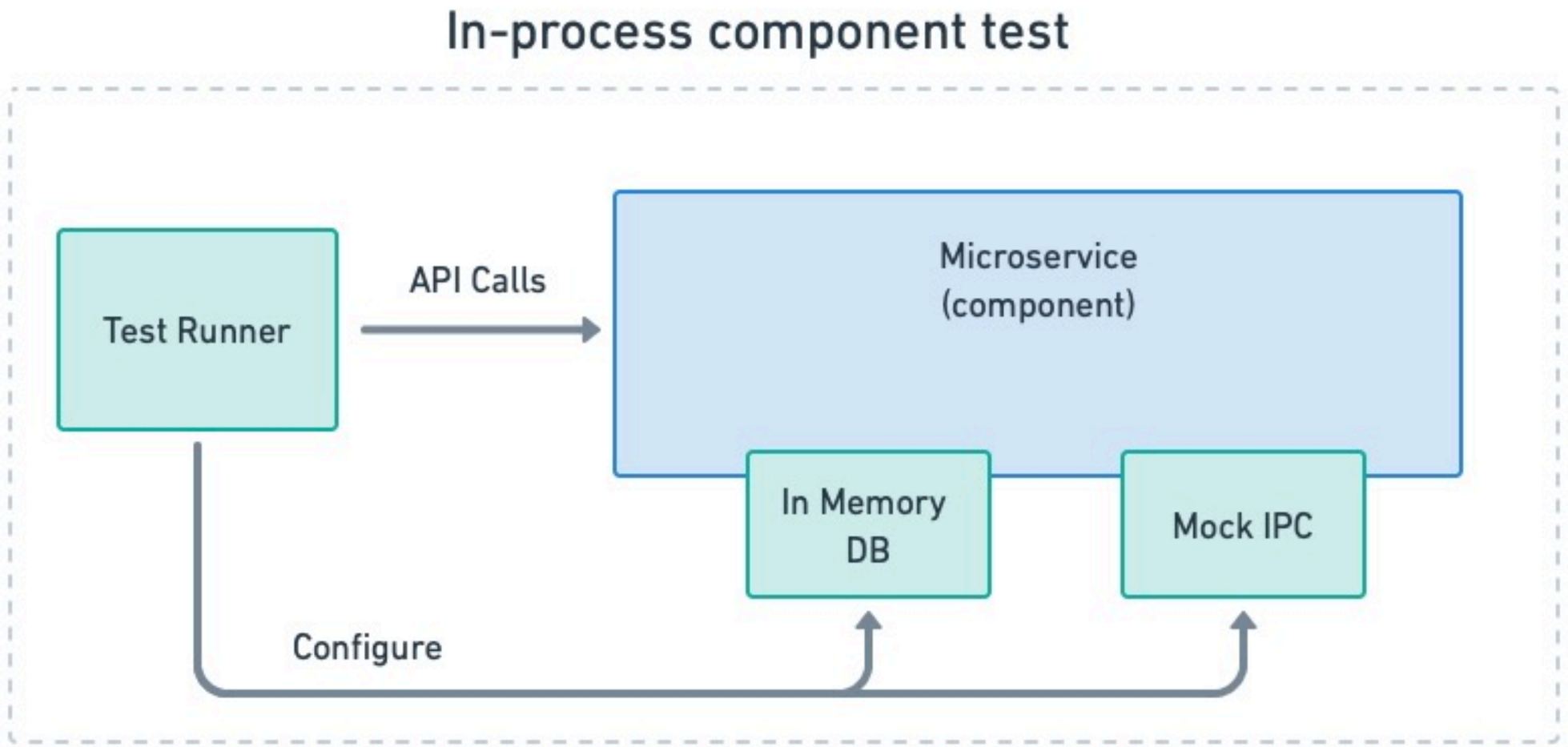
Component testing



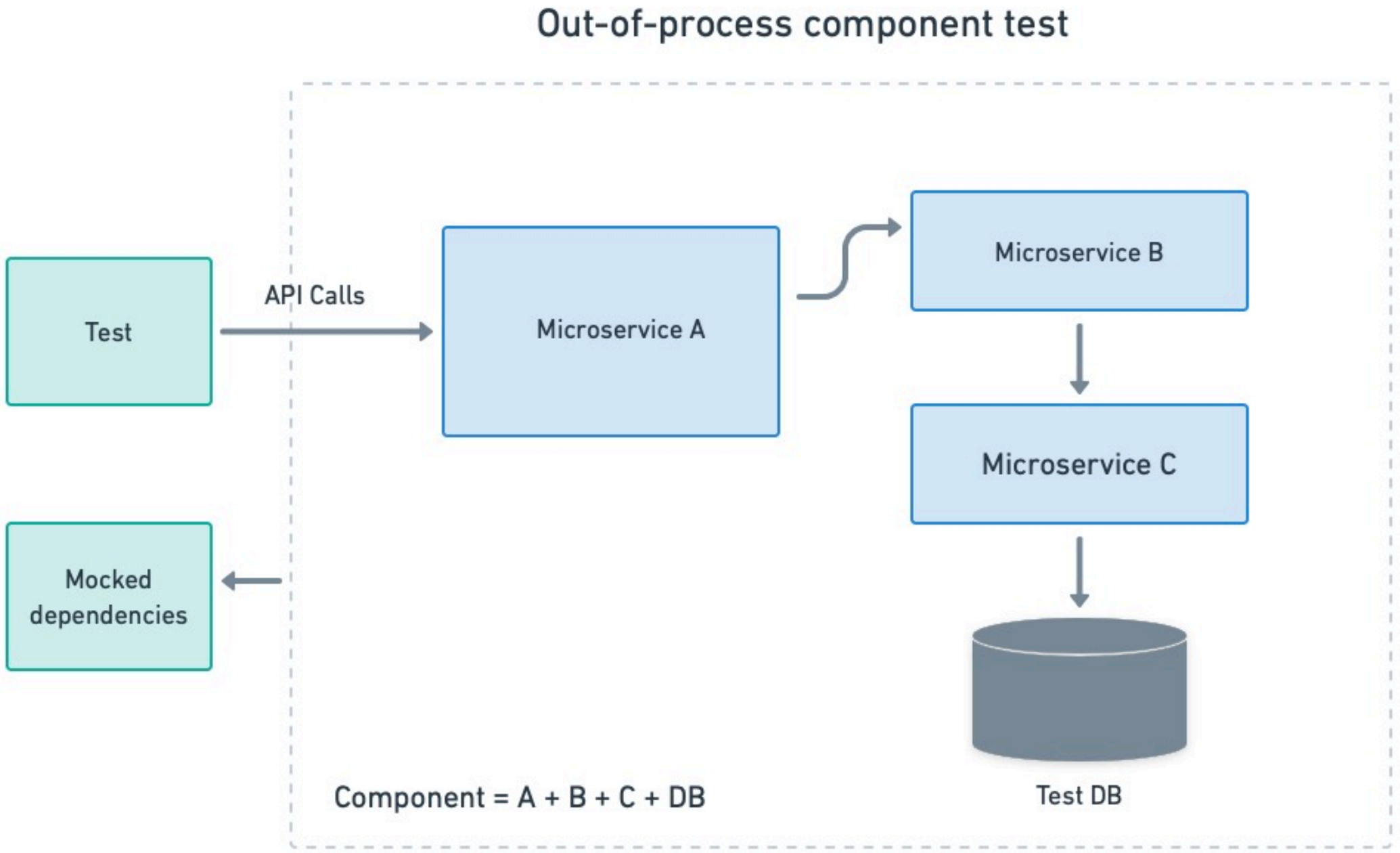
Component testing



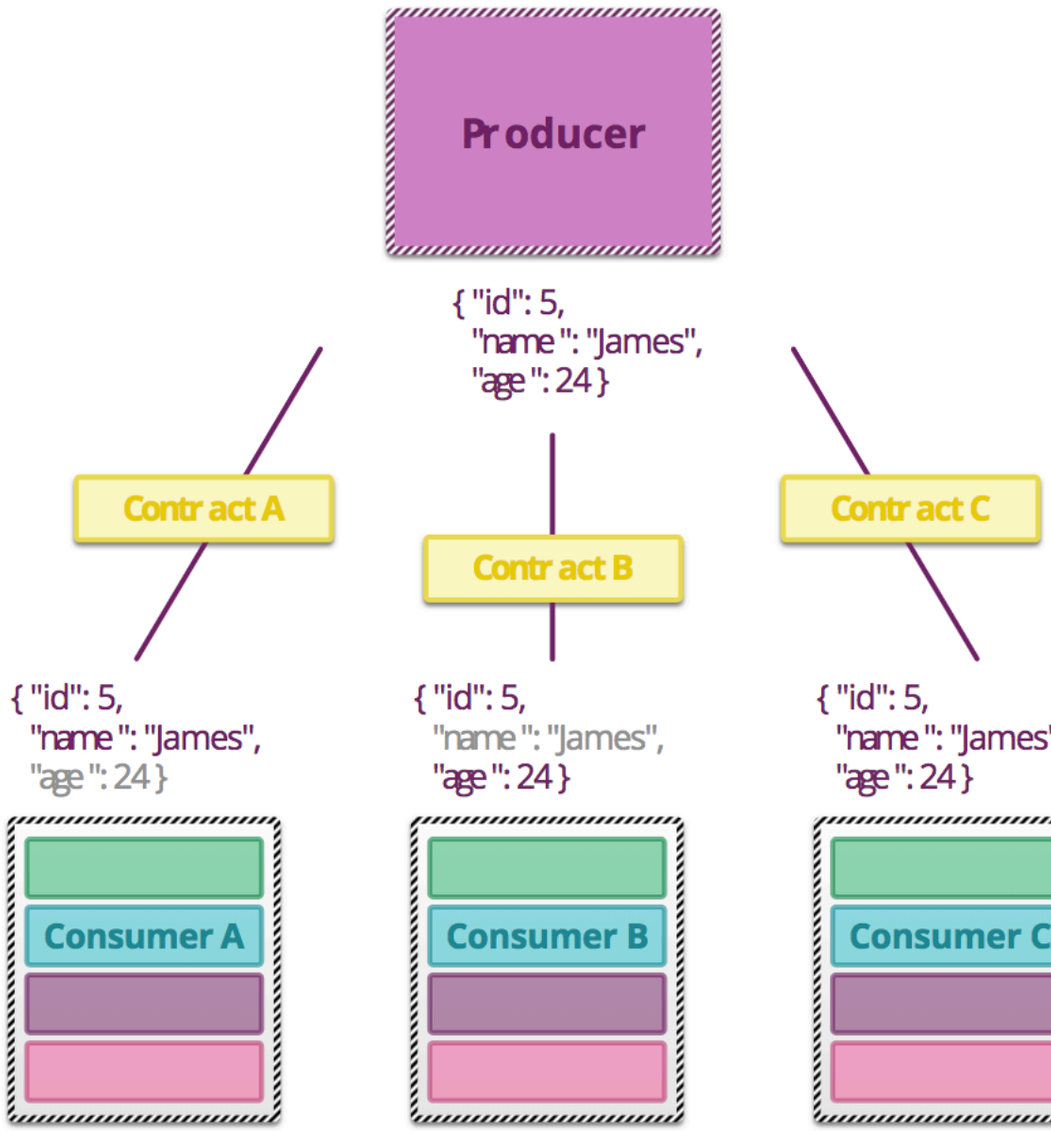
Component testing (in-process)



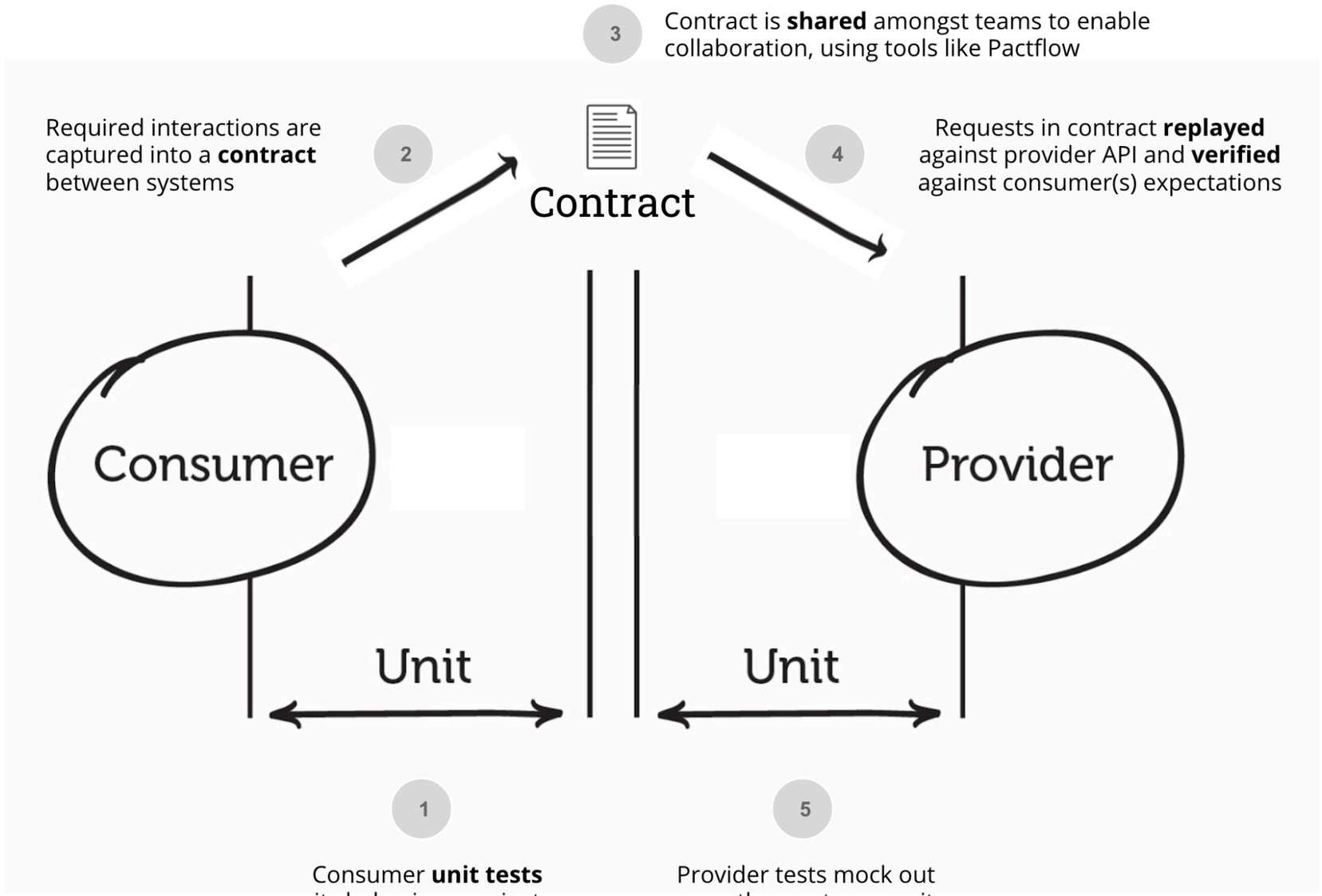
Component testing (out-process)



Contract testing



Contract testing with Pact



https://docs.pact.io/

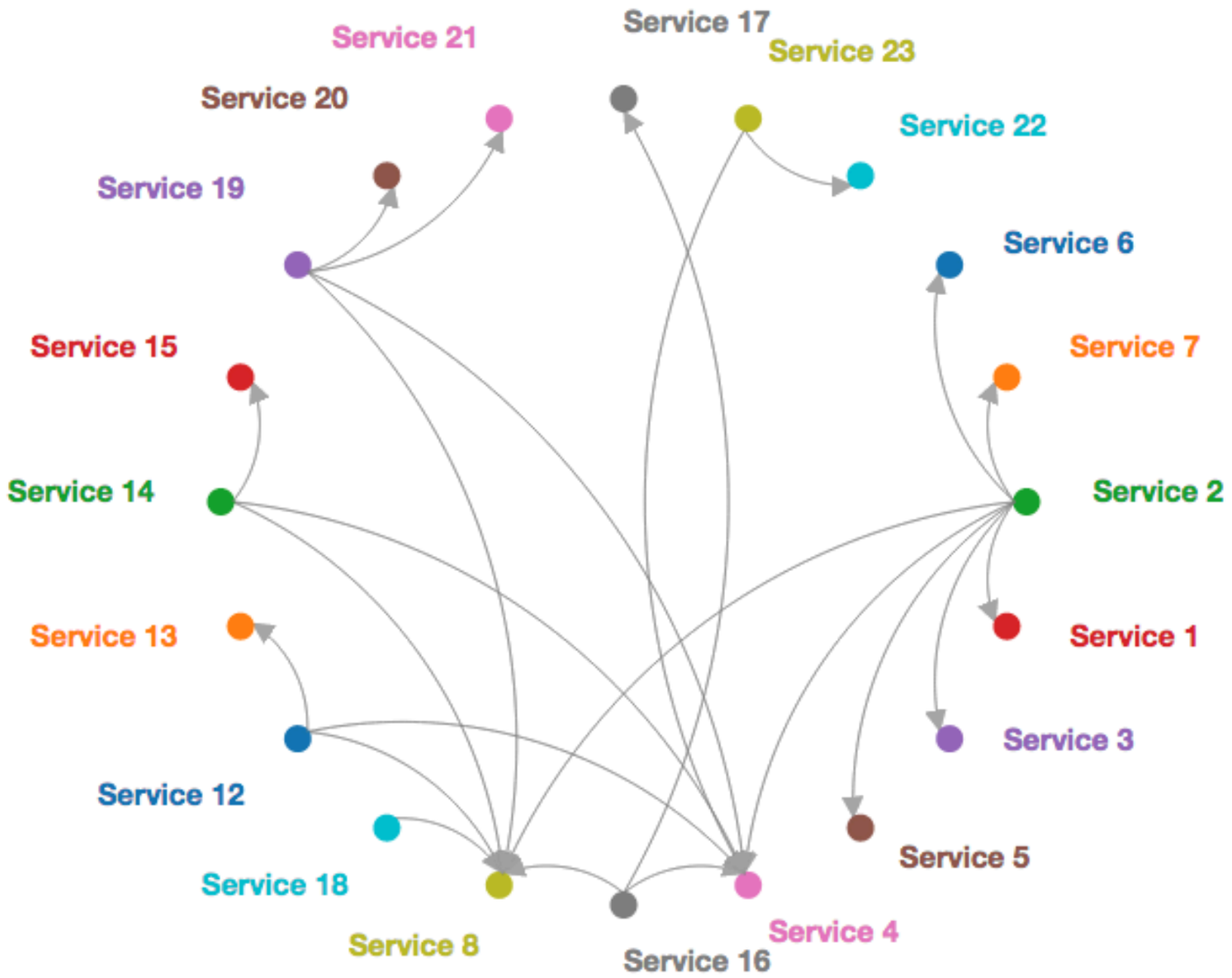
Pact broker

Store and share all contracts and verification results

Consumer	Provider	Latest pact published	Last verified
Foo	Animals	2 minutes ago	2 days ago
Foo	Bar	7 days ago	15 days ago ⚠
Foo	Hello World App	1 day ago	
Foo	Wiffles	less than a minute ago	7 days ago
Some other app	A service	26 days ago	less than a minute ago
The Android App	The back end	less than a minute ago	

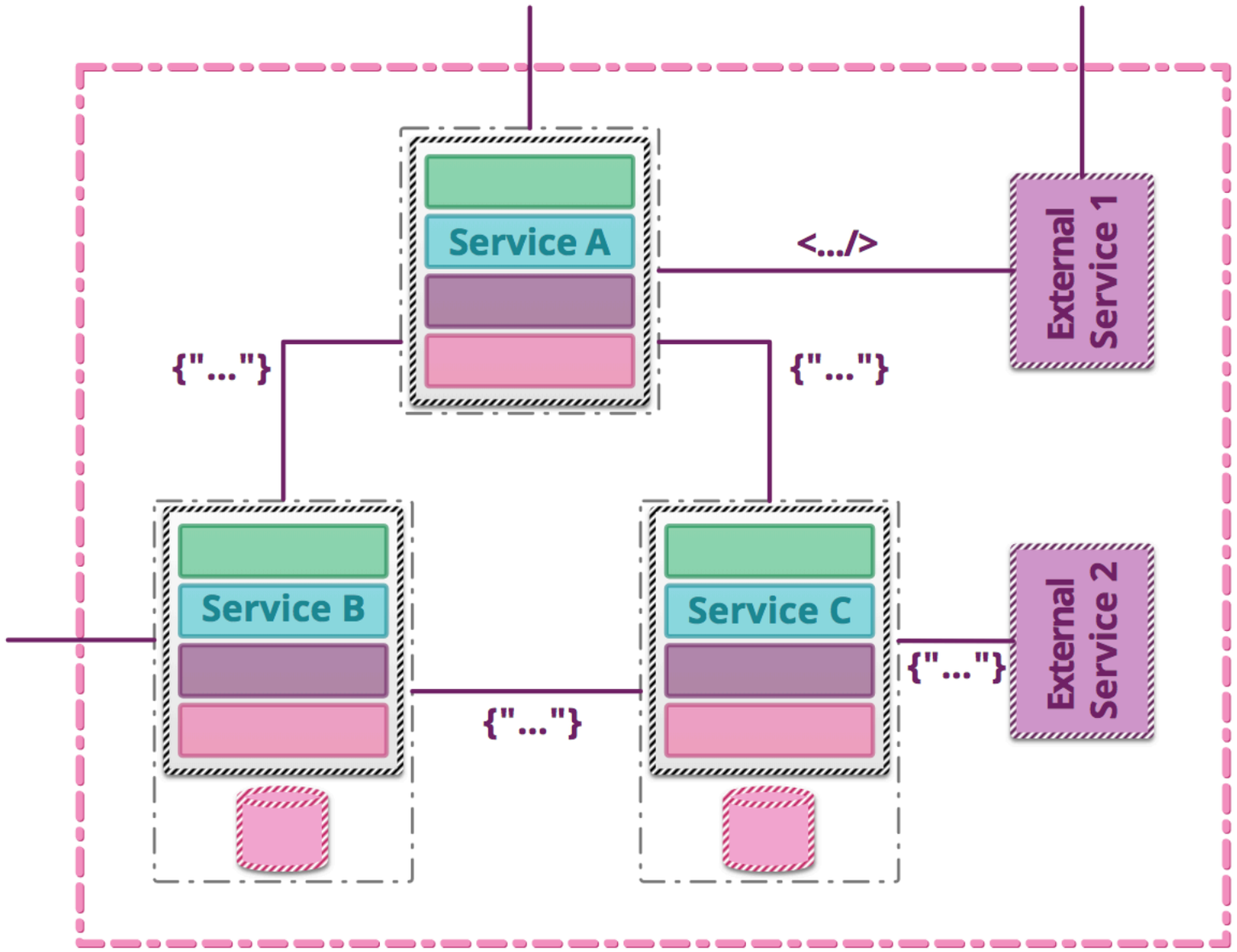
https://docs.pact.io/pact_broker

Pact broker

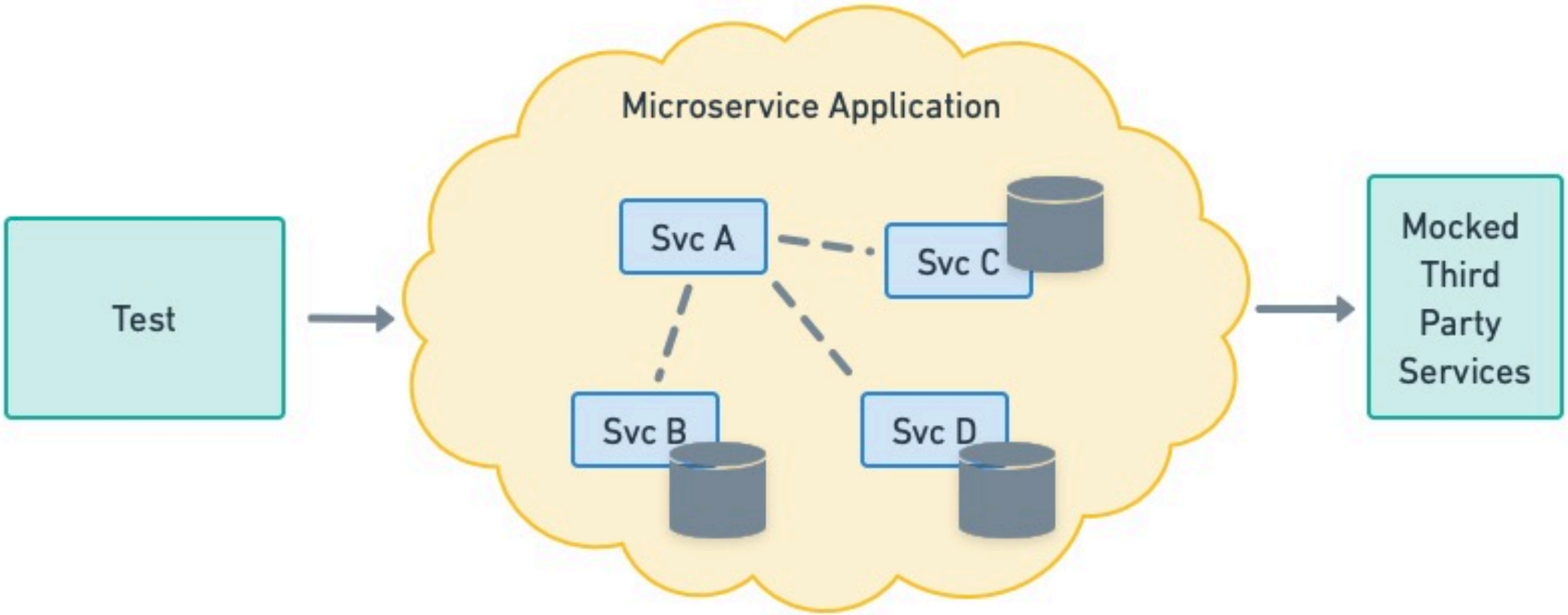


https://docs.pact.io/pact_broker

End-to-End testing

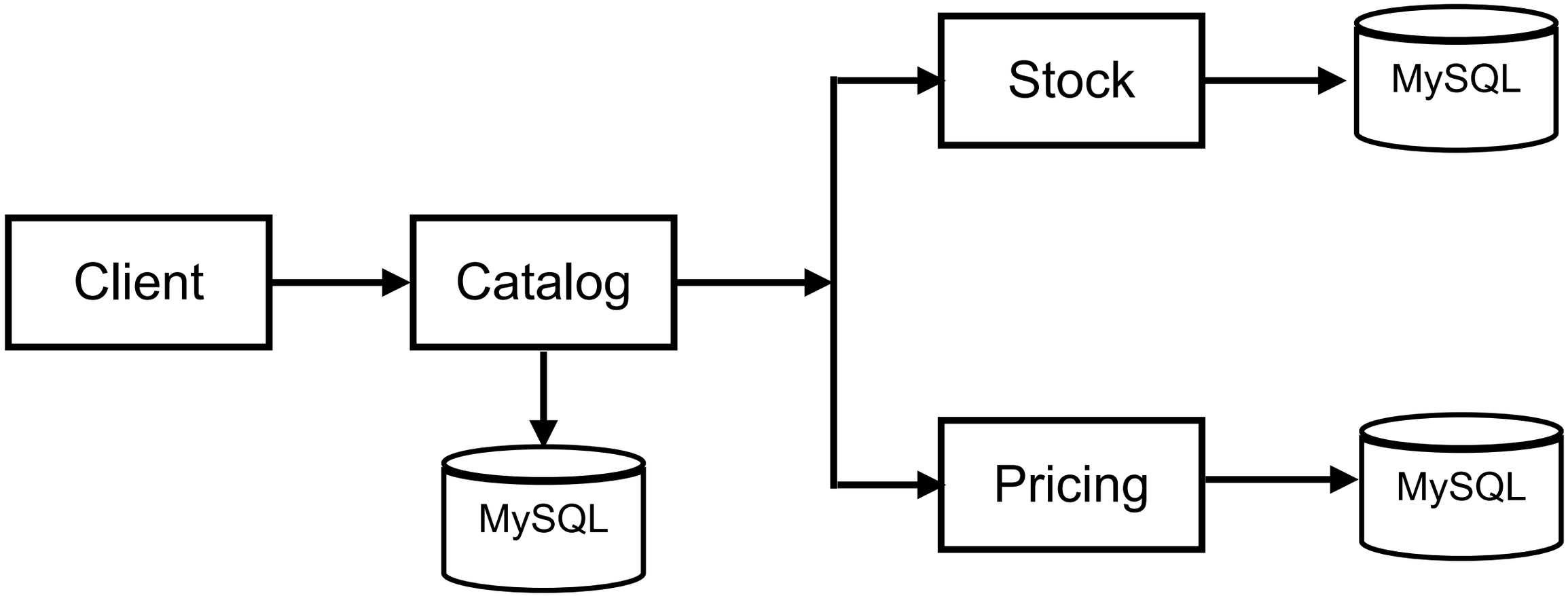


End-to-End testing



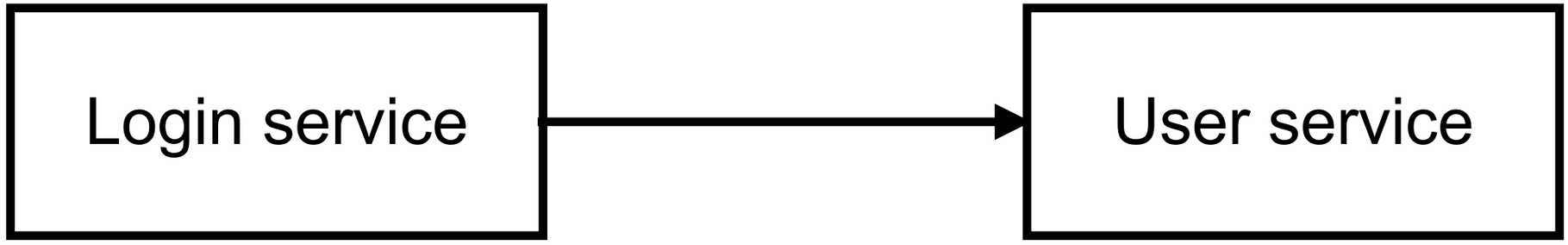
What is your testing strategy ?

How to test ?



Testing workshop

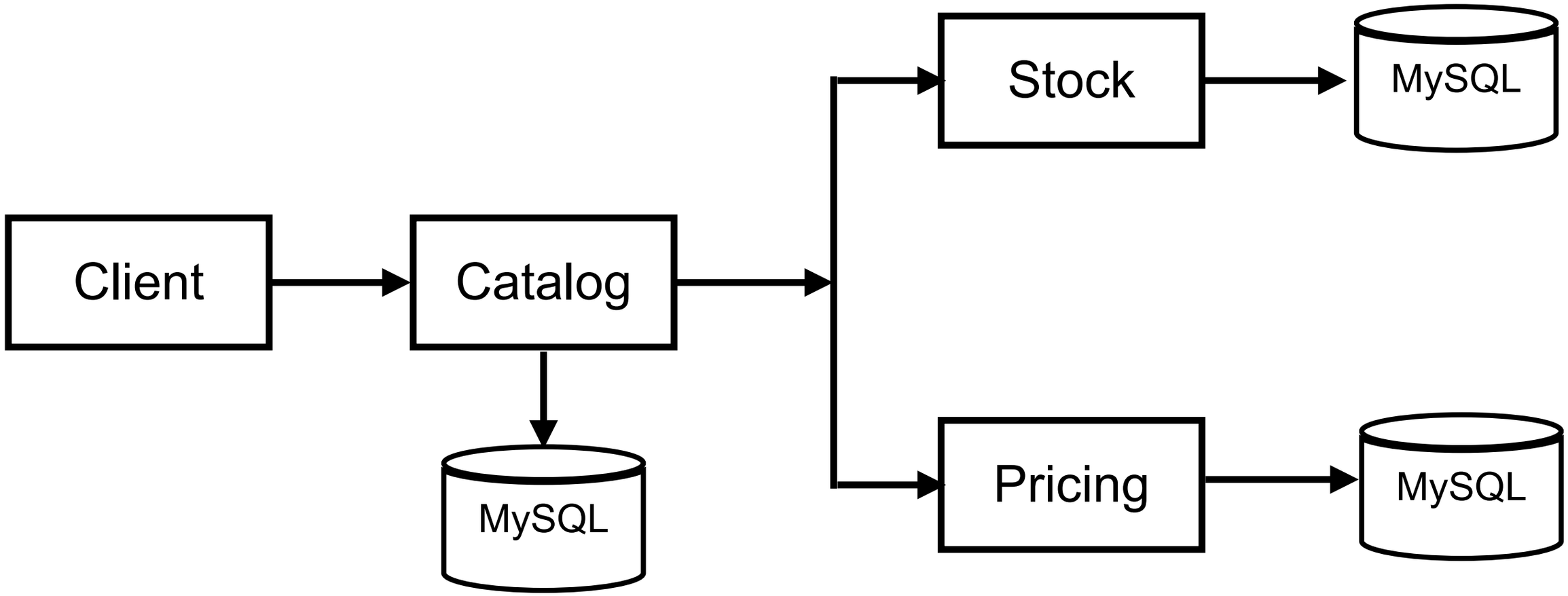
Component testing
Contract testing



https://github.com/up1/course-contract-testing

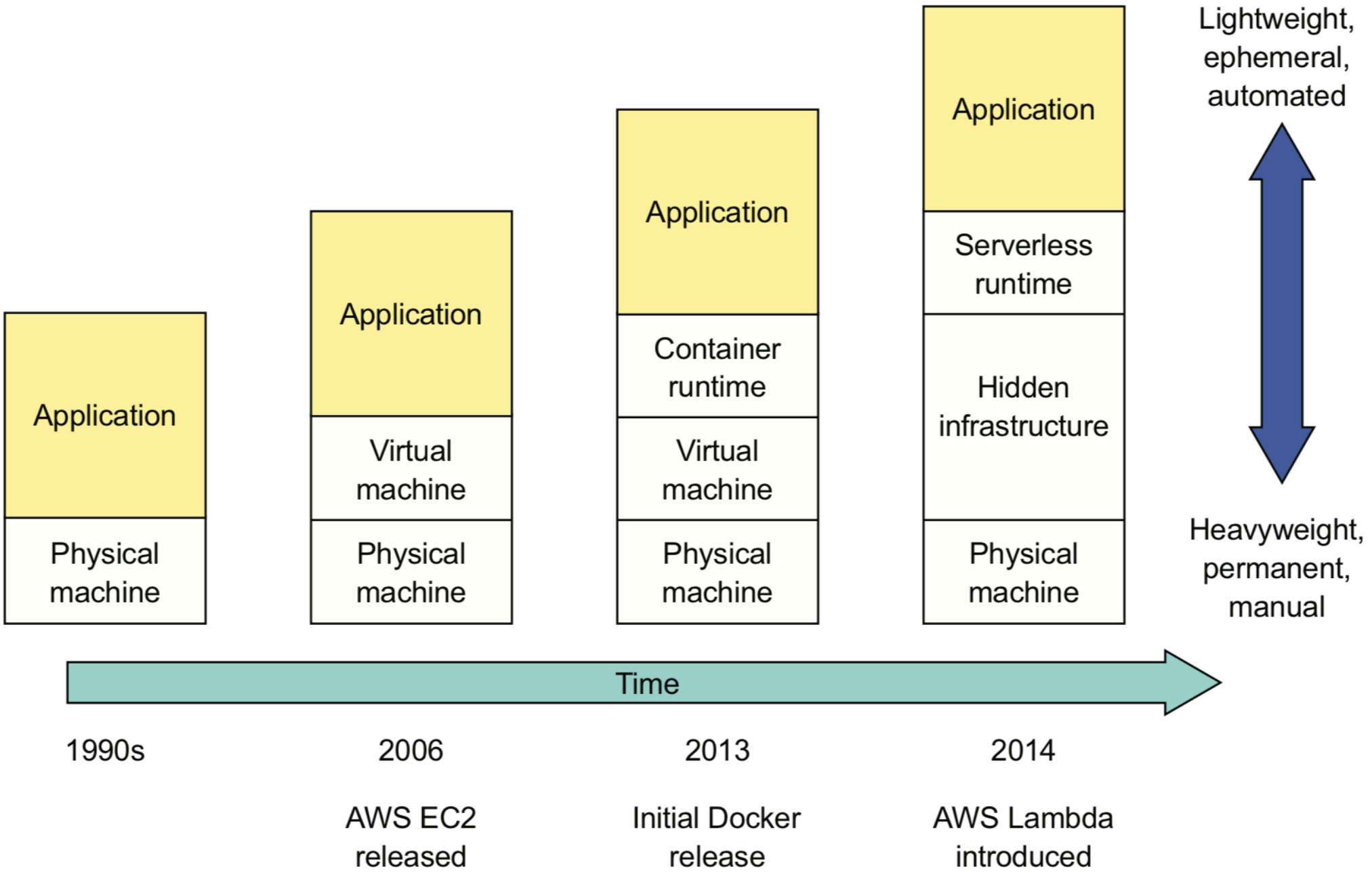
Performance testing ?
Security testing ?

How to performance test ?

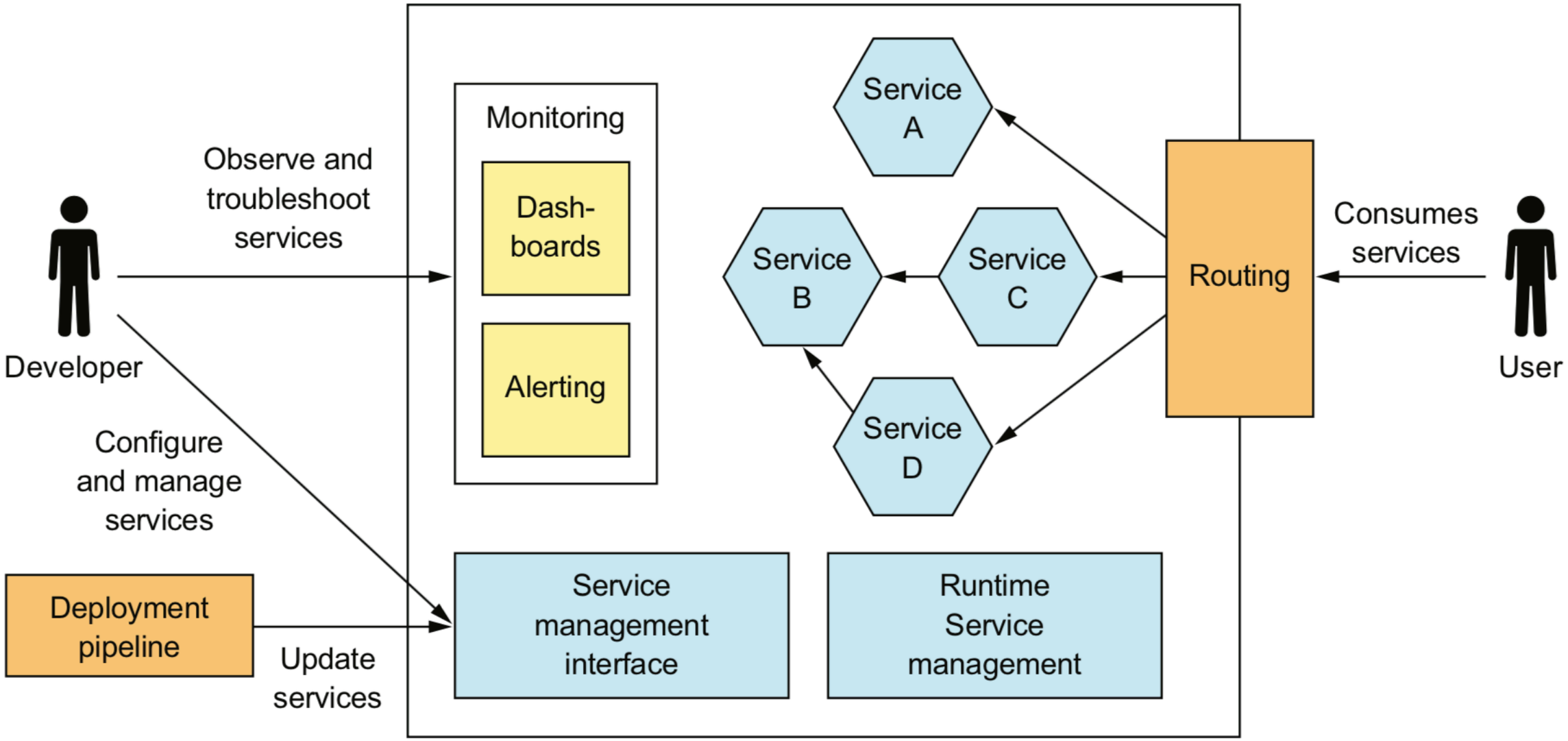


Module 3 : Deploy

Infrastructure



View of production environment



Deploy Microservice

Language-specific packaging

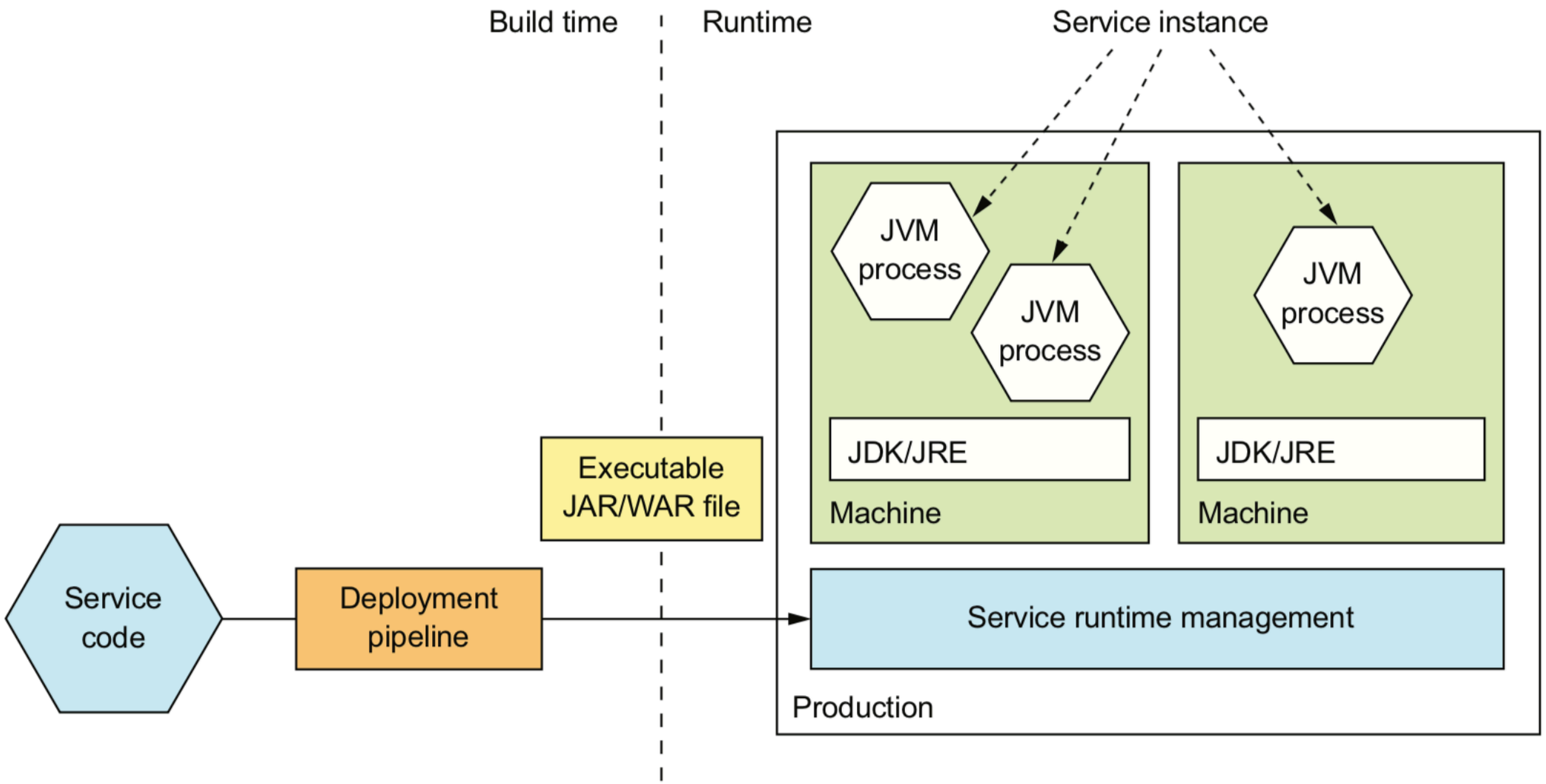
Virtual Machine (VM)

Container

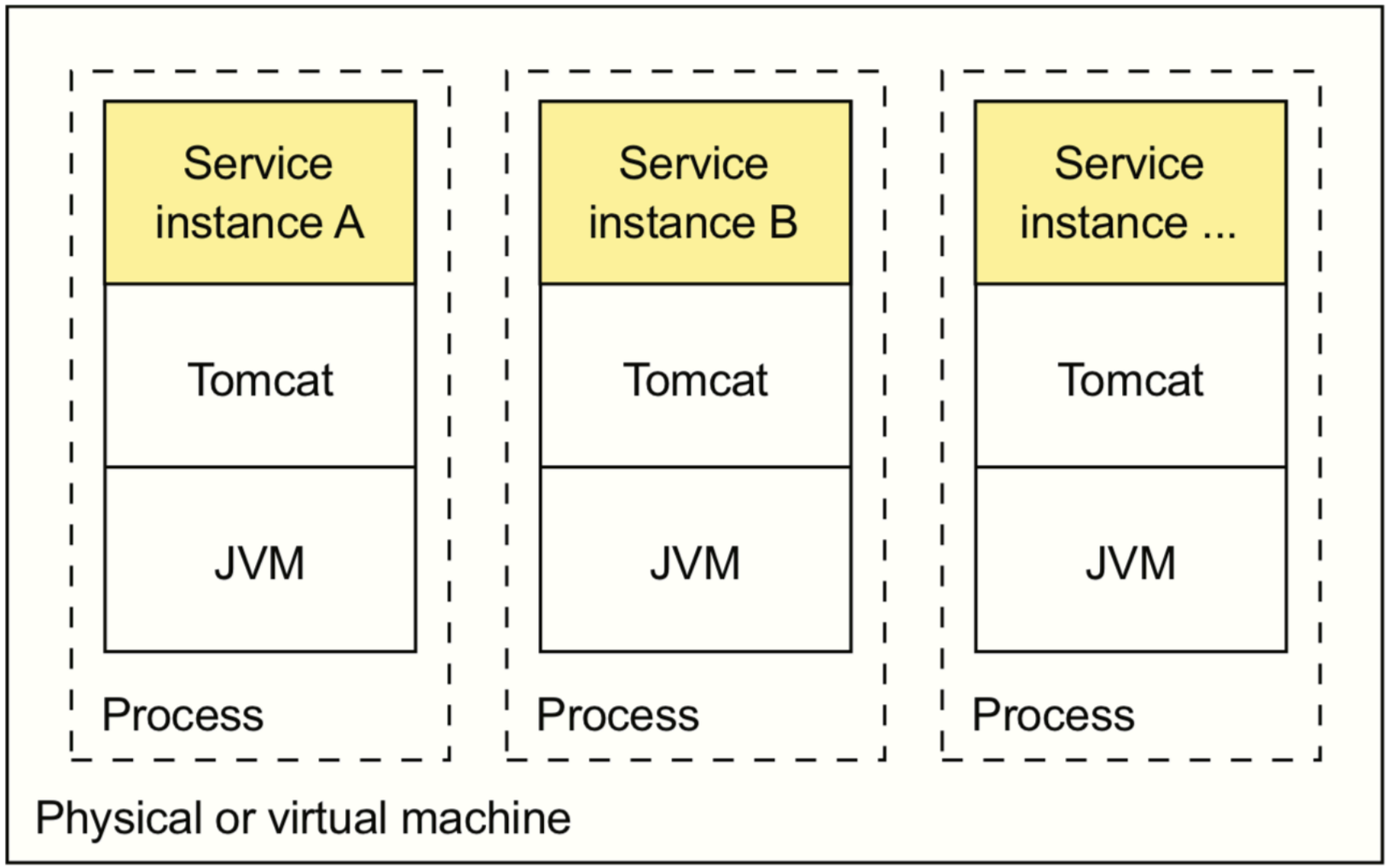
Kubernetes

Serverless/FaaS

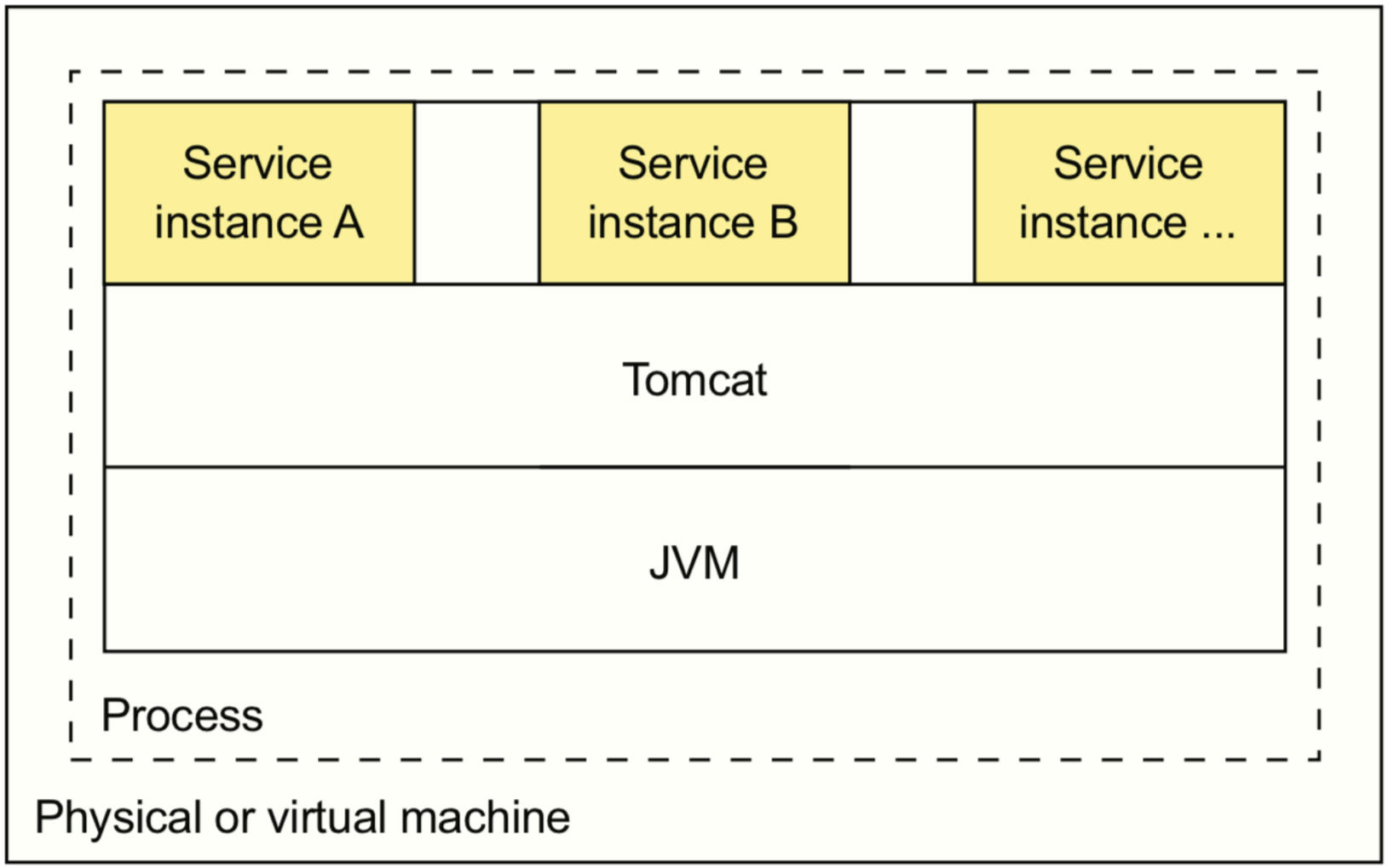
1. Language-specific packaging



Multiple services on same machine



Multiple services on same process



Benefits

Fast deployment

Efficient resource utilization

Service instances’s resources are constrained

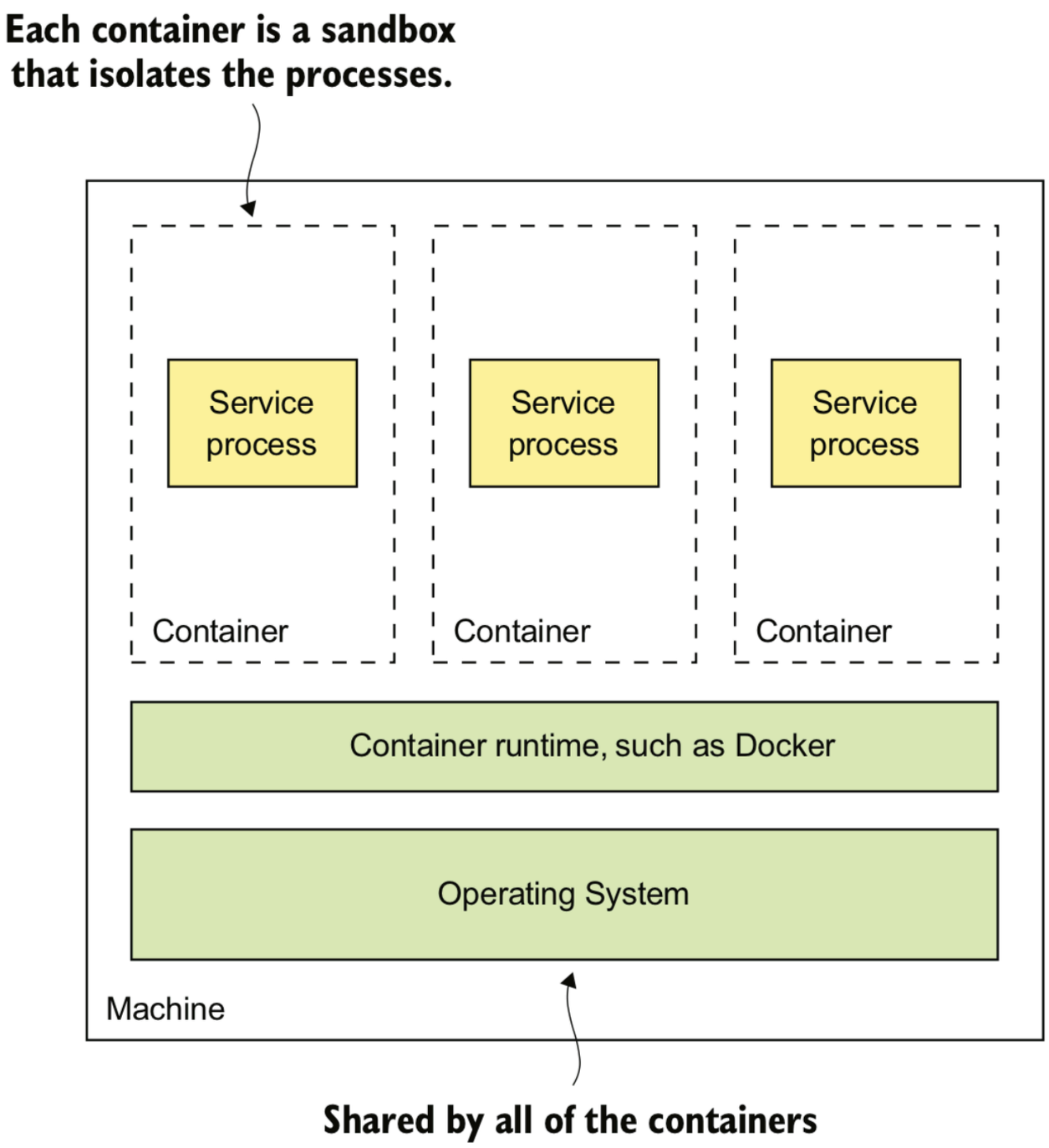
Drawbacks

Lack of encapsulation of technology stack

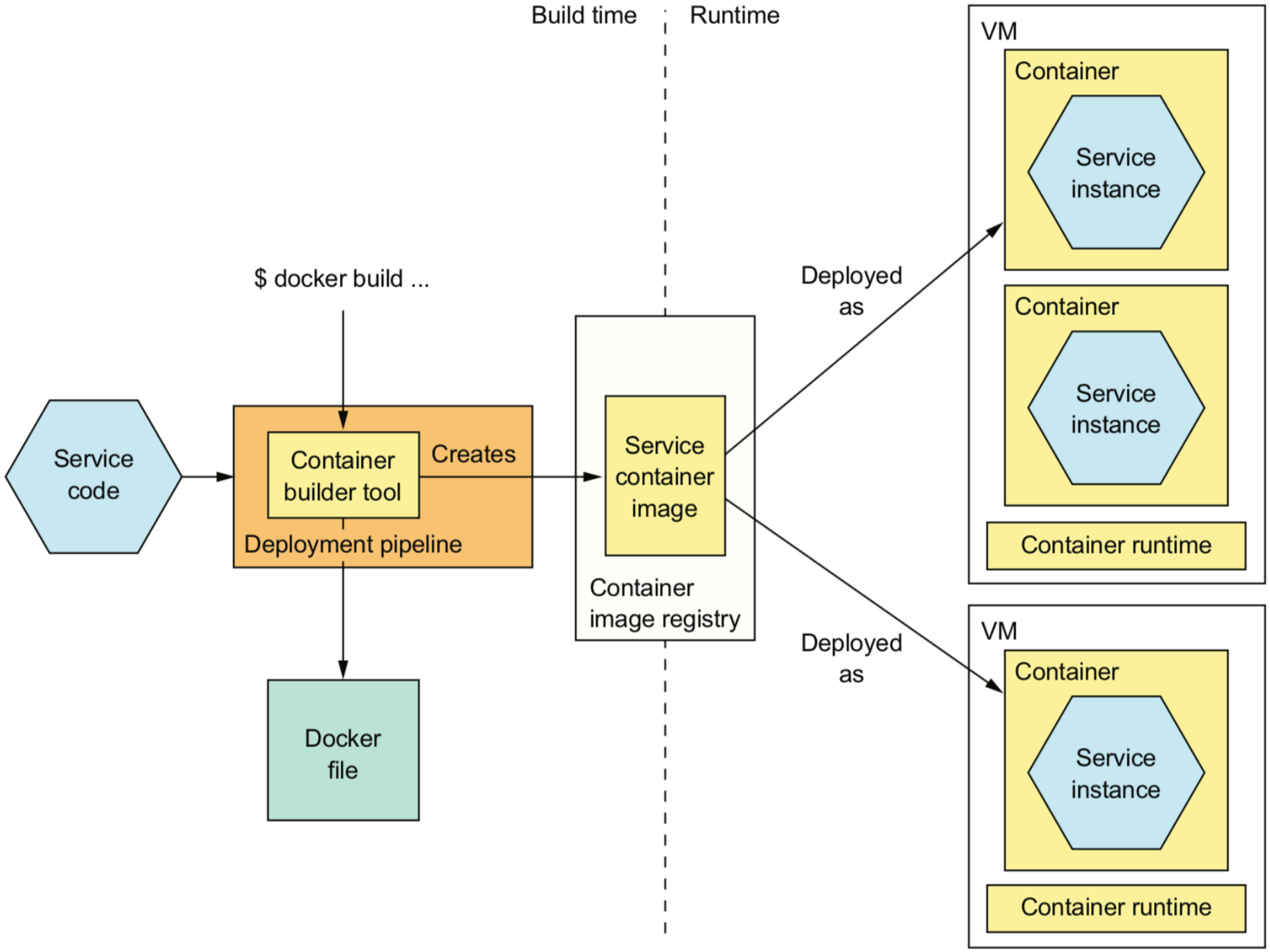
No ability to constrain resources of service

Lack of isolation

2. Working with container



Deployment with container



Deploy services with Docker

Build a docker image
Push docker image to a registry
Run docker container
Working with docker-compose

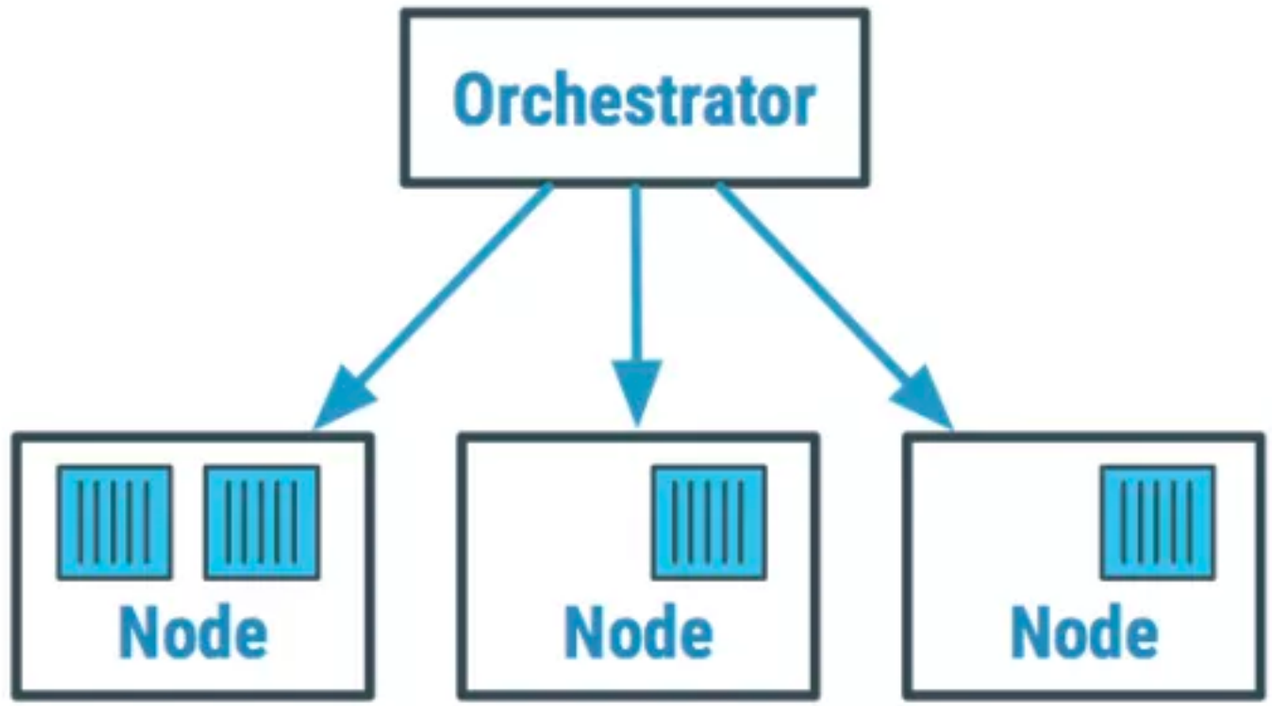
Benefits

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

Application Deployment Strategies

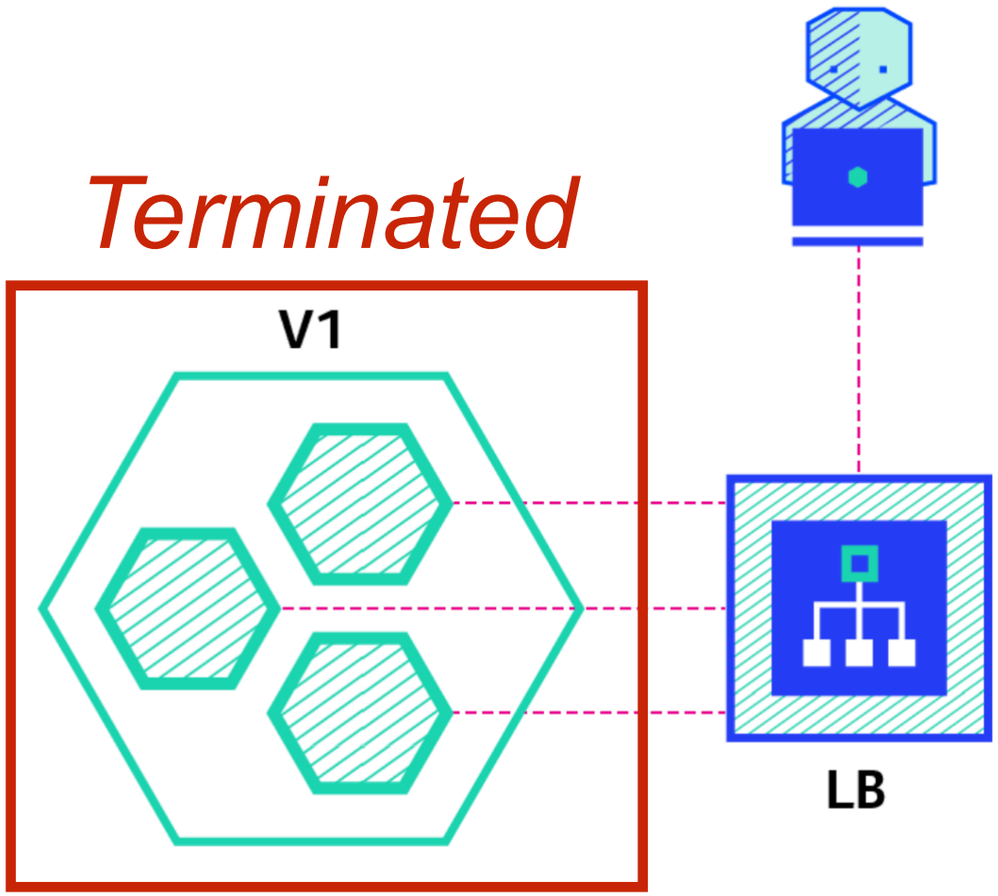
Strategies to deploy

Recreate
Ramped
Blue/Green
Canary
A/B testing
Shadow

https://thenewstack.io/deployment-strategies/

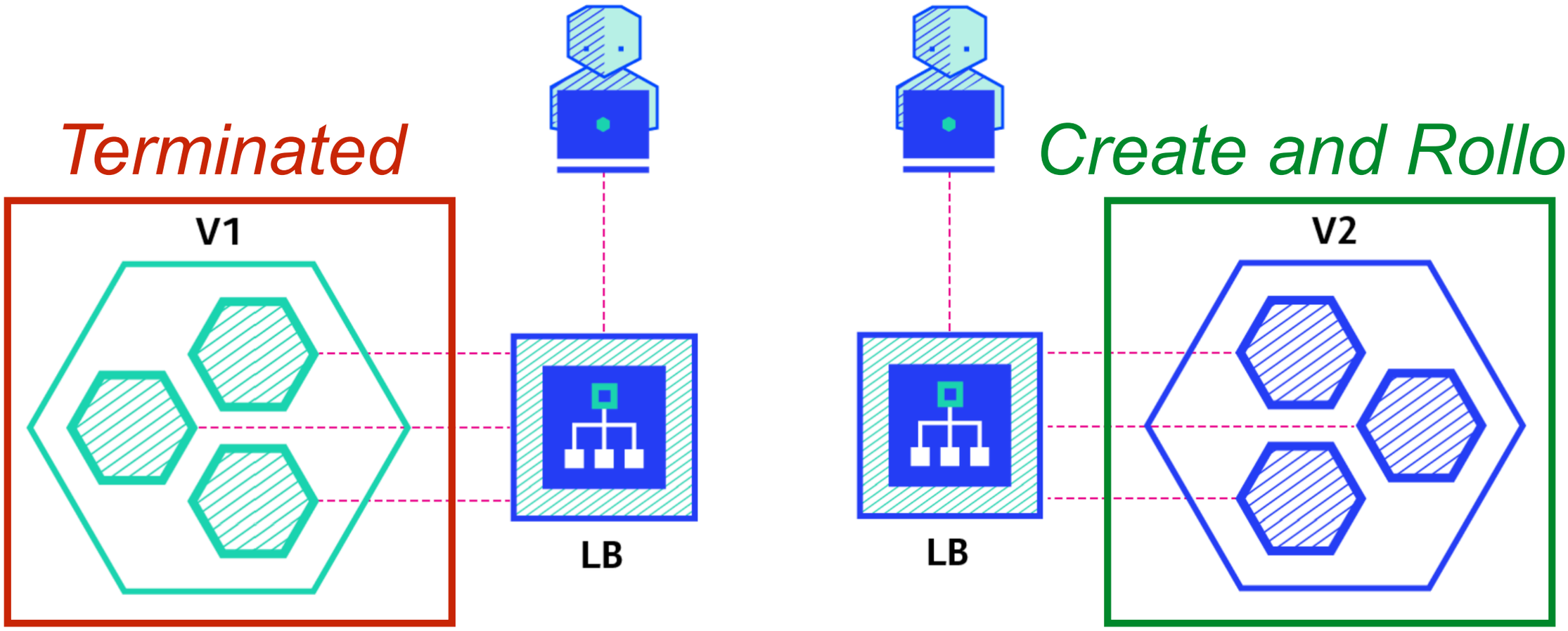
1. Recreate

Version A is terminated then version B is rollout



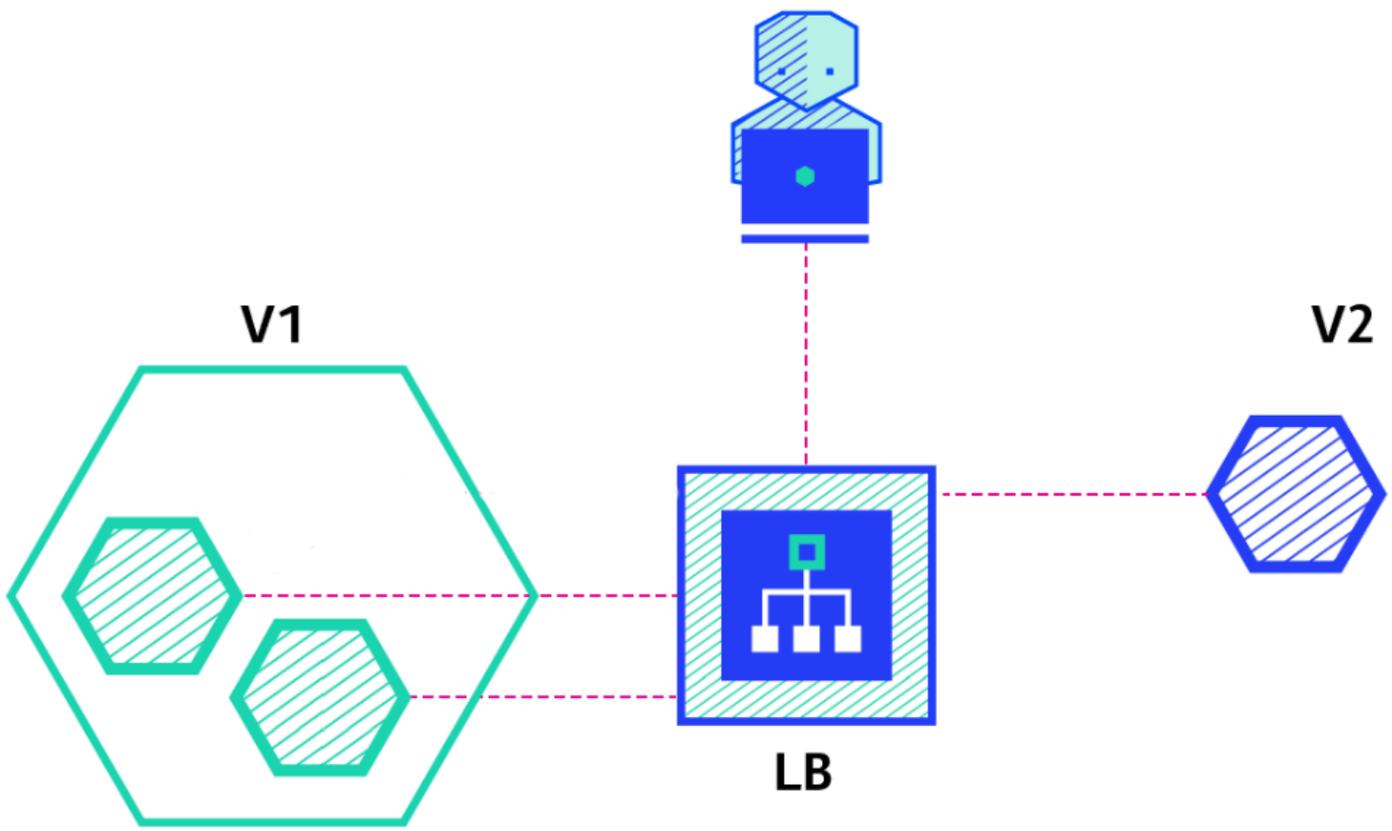
1. Recreate

Version A is terminated then version B is rollout



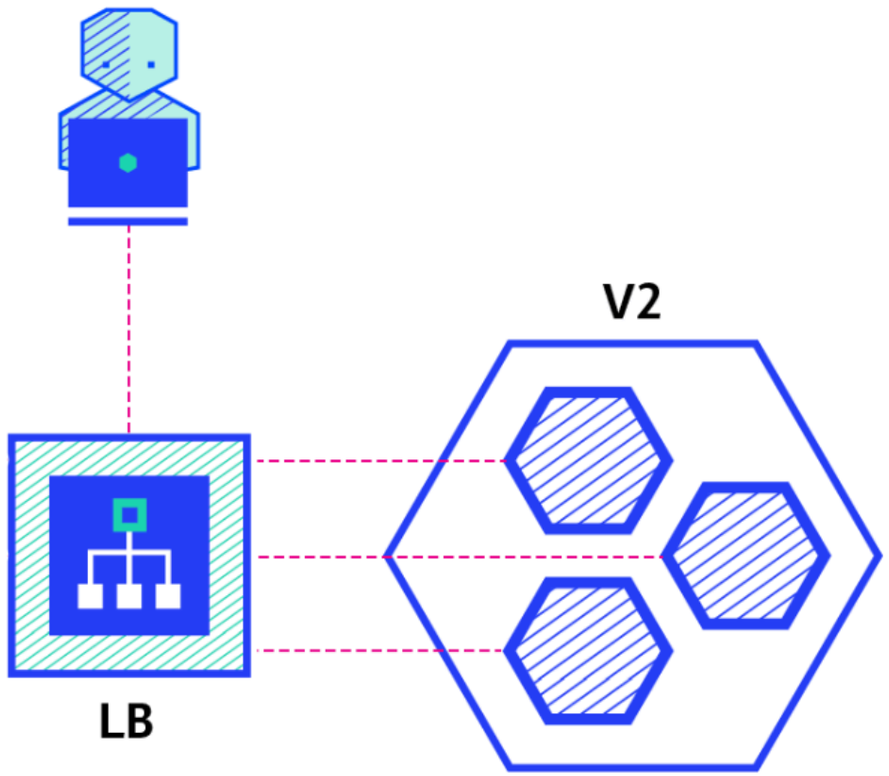
2. Ramped

Slow roll out by replace instance one-by-one



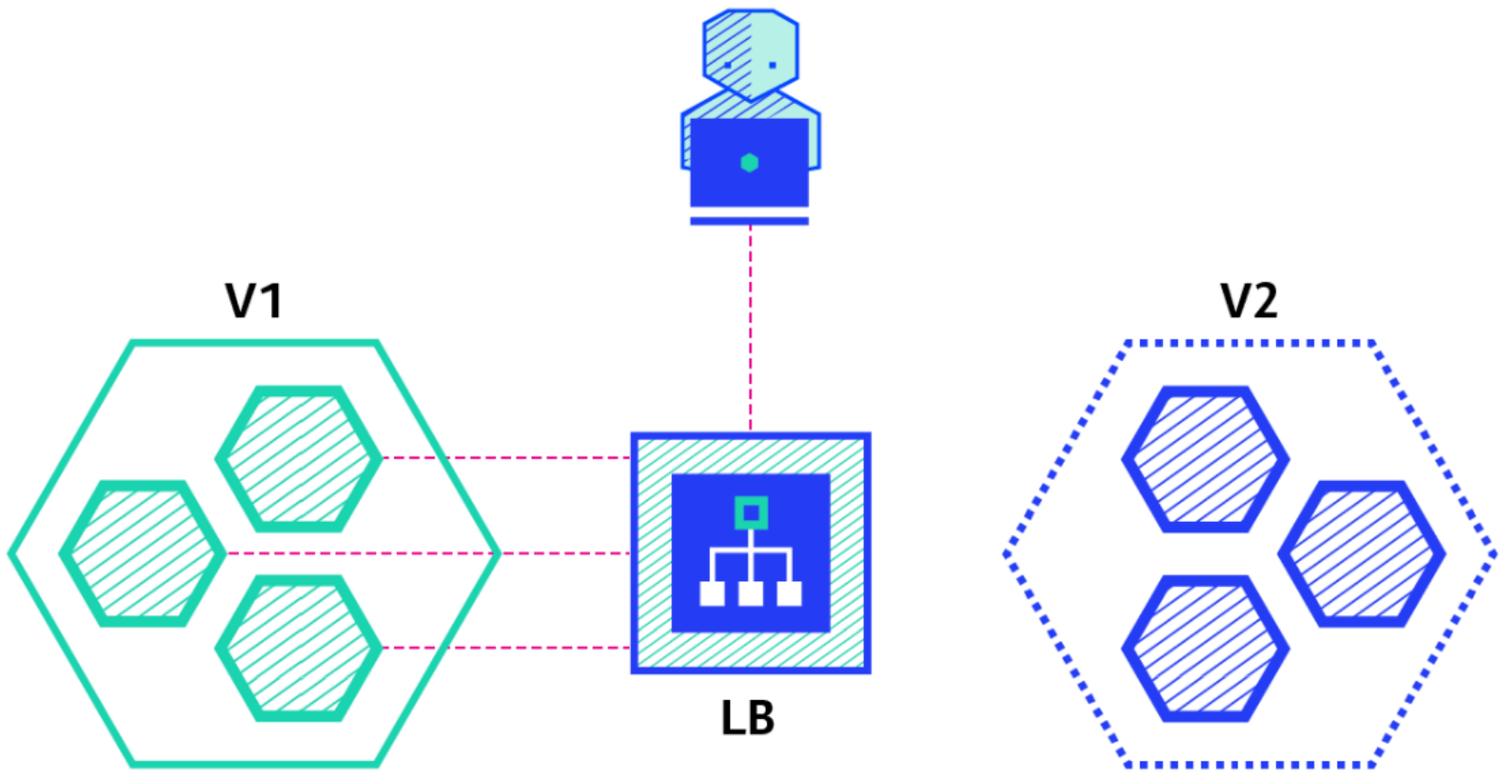
2. Ramped

Slow roll out by replace instance one-by-one

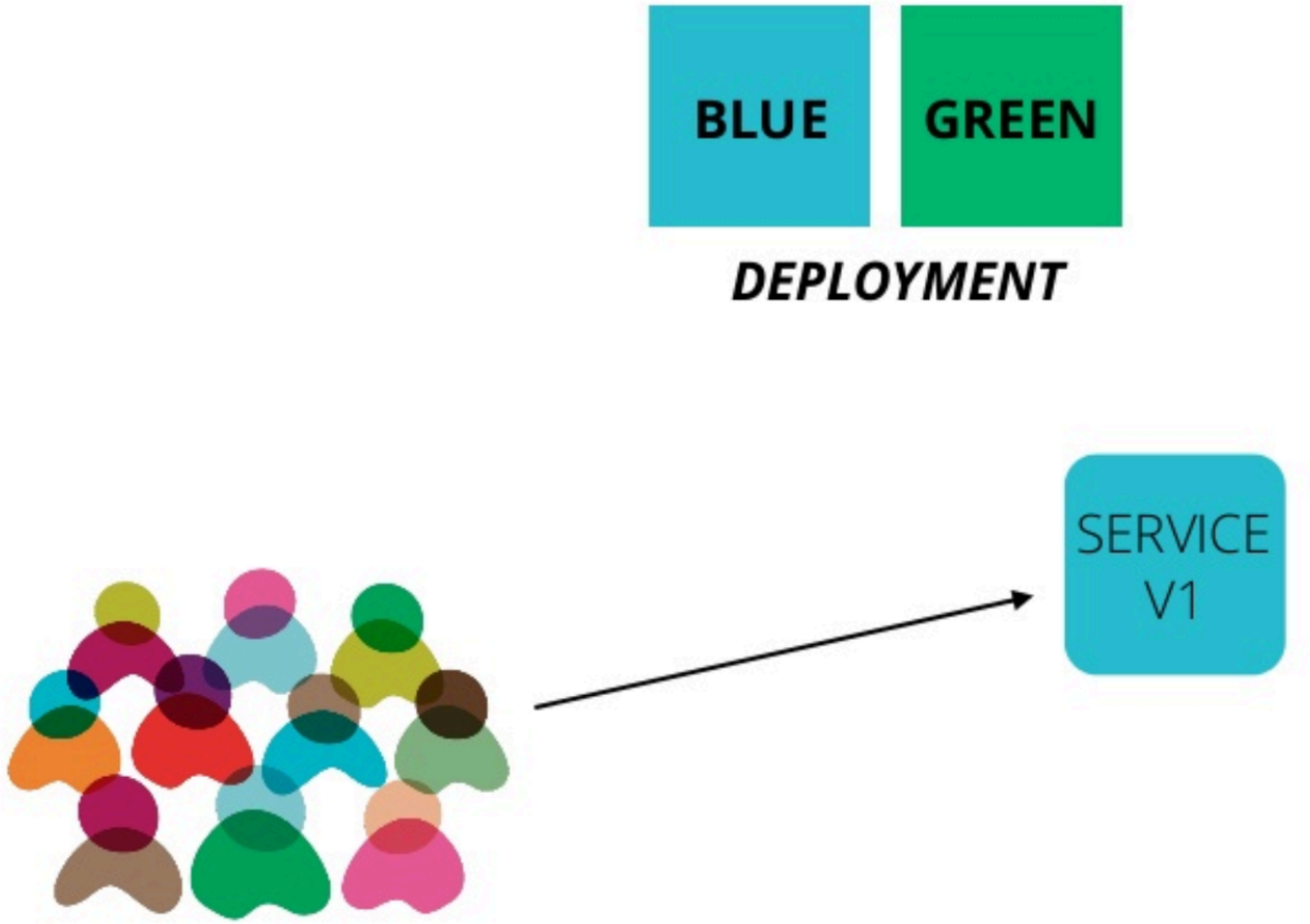


3. Blue/Green

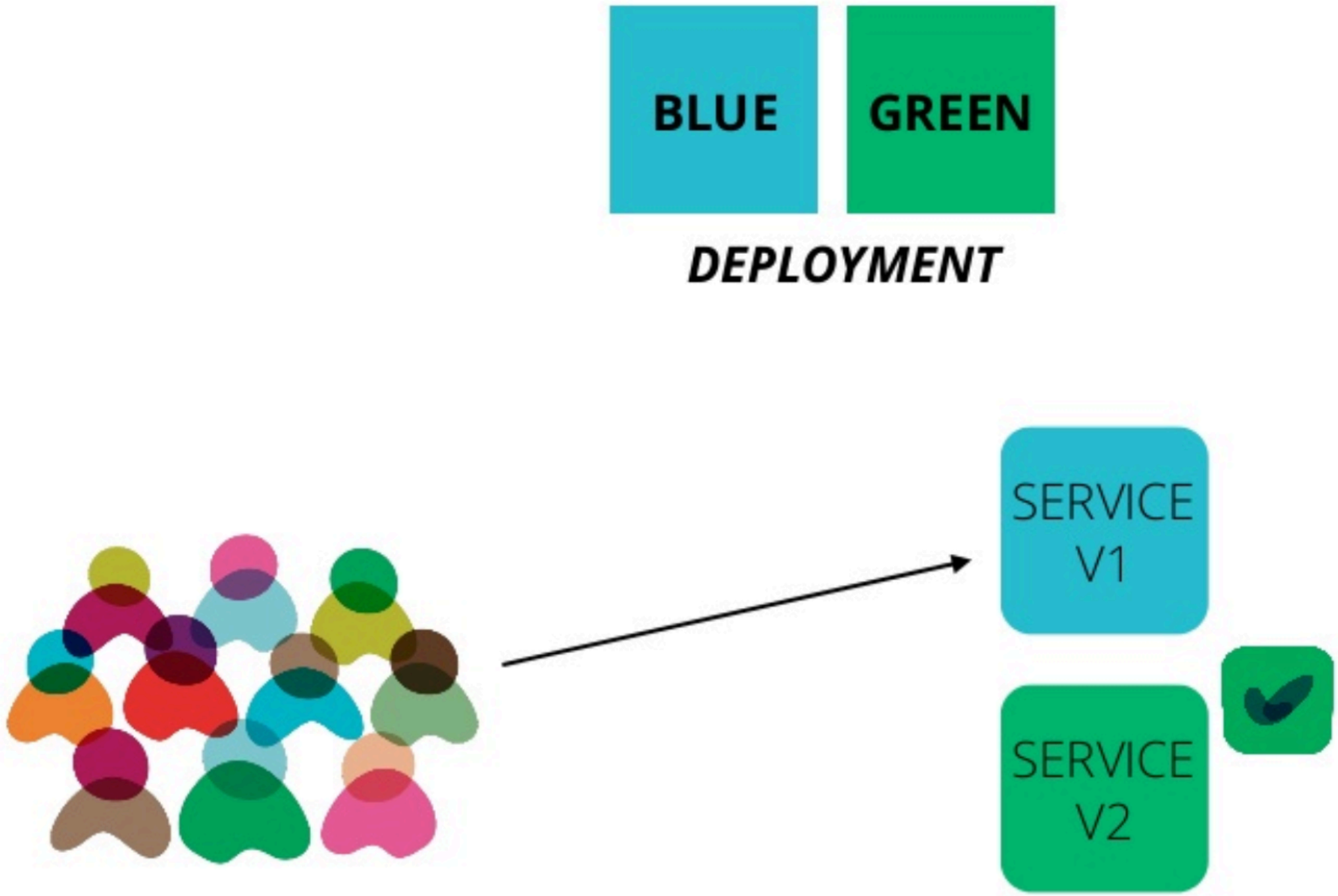
Current version is called **Blue**
New version is called **Green**



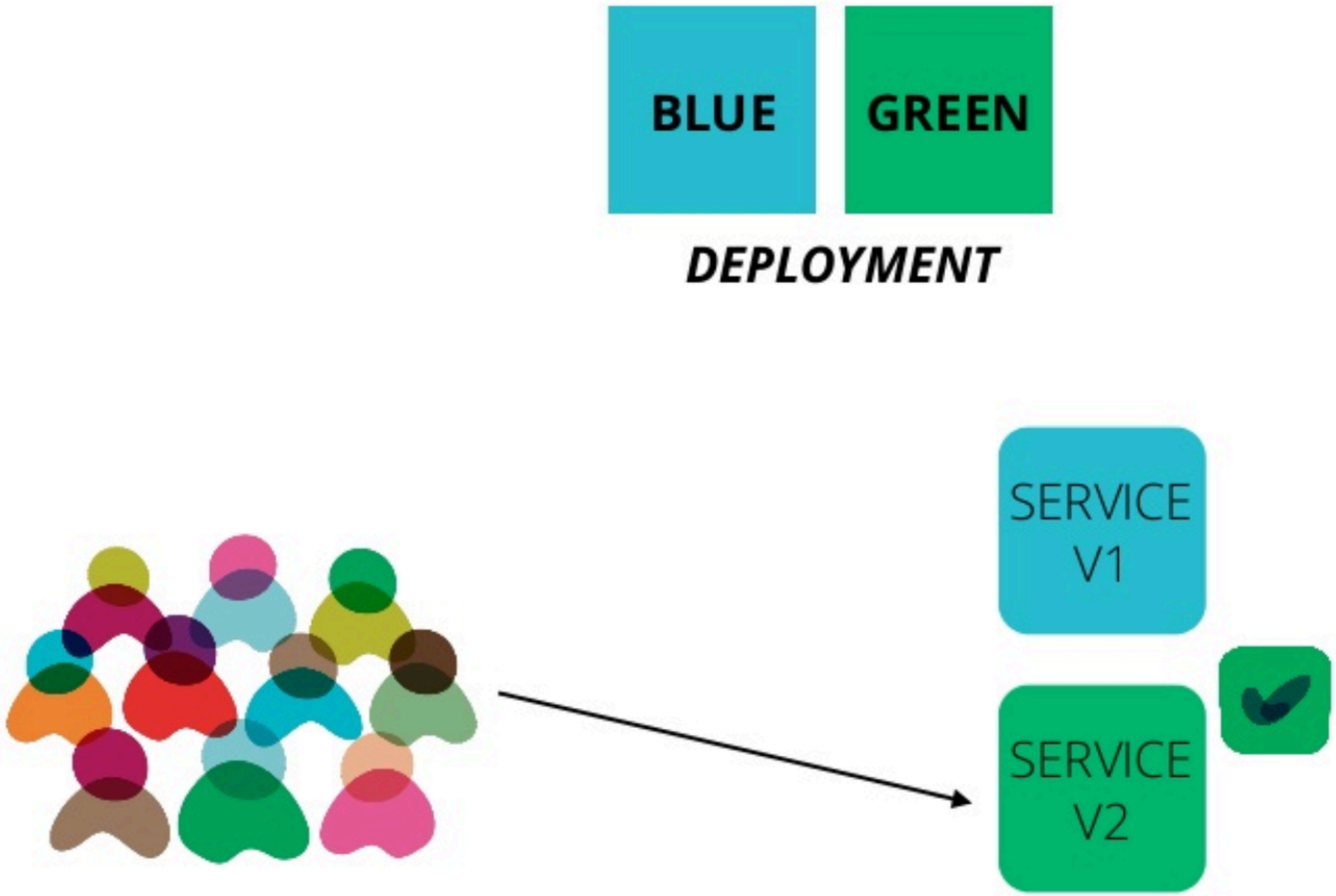
Blue Green Deployment



Blue Green Deployment



Blue Green Deployment

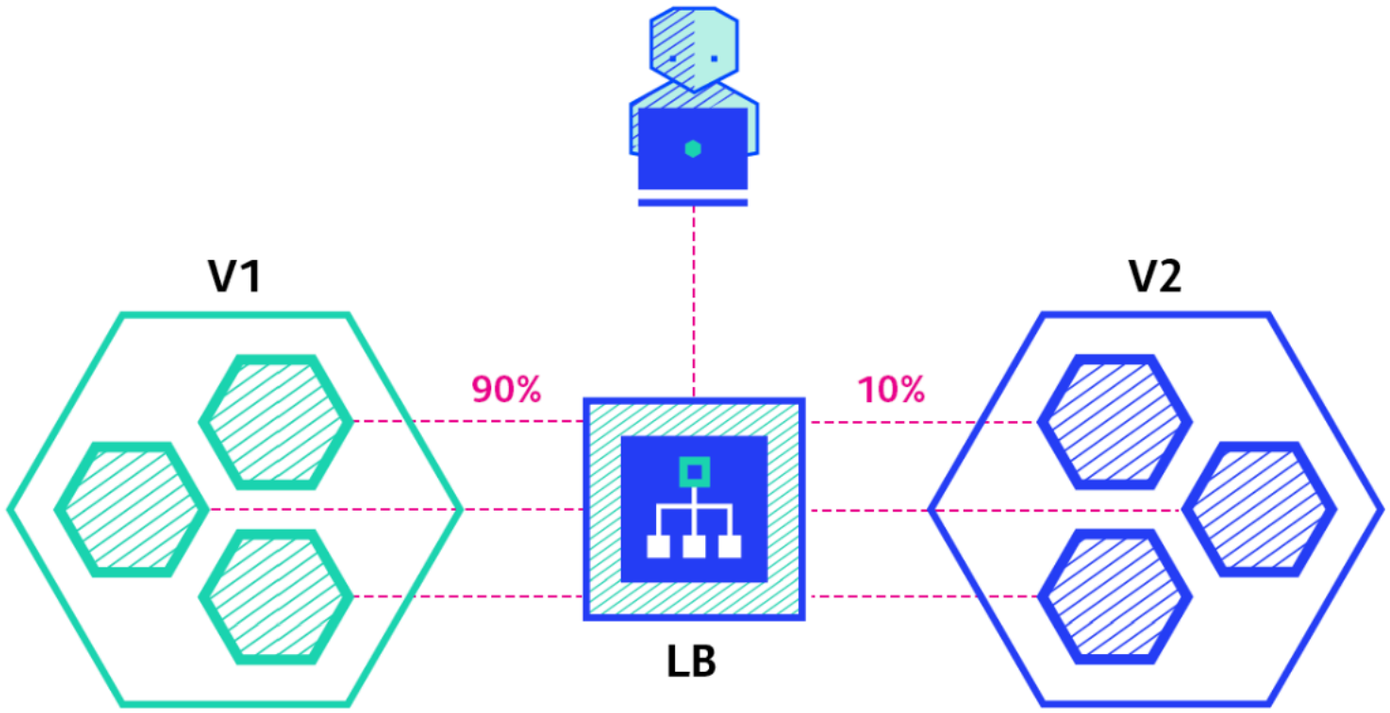


4. Canary

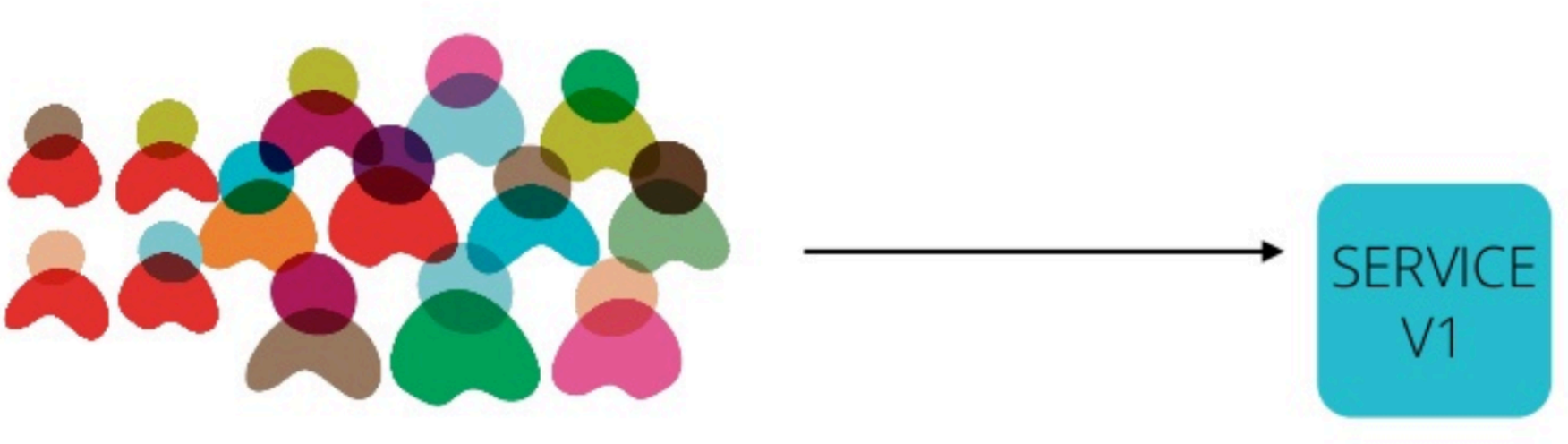
Shift production traffic from version A to B

Traffic is split based-on weight

Use when tests are lacking/not reliable and less confident in system

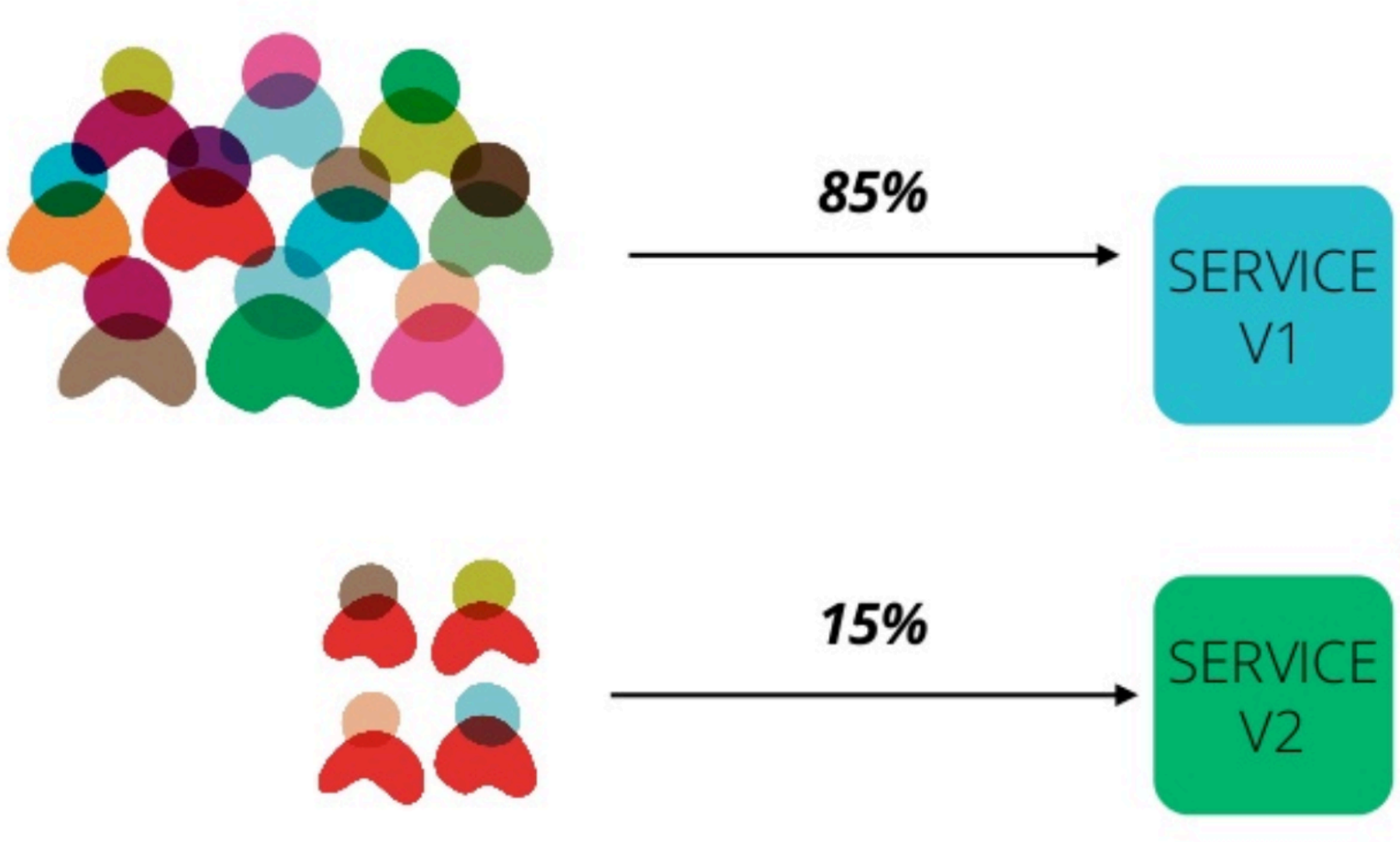


Canary Release



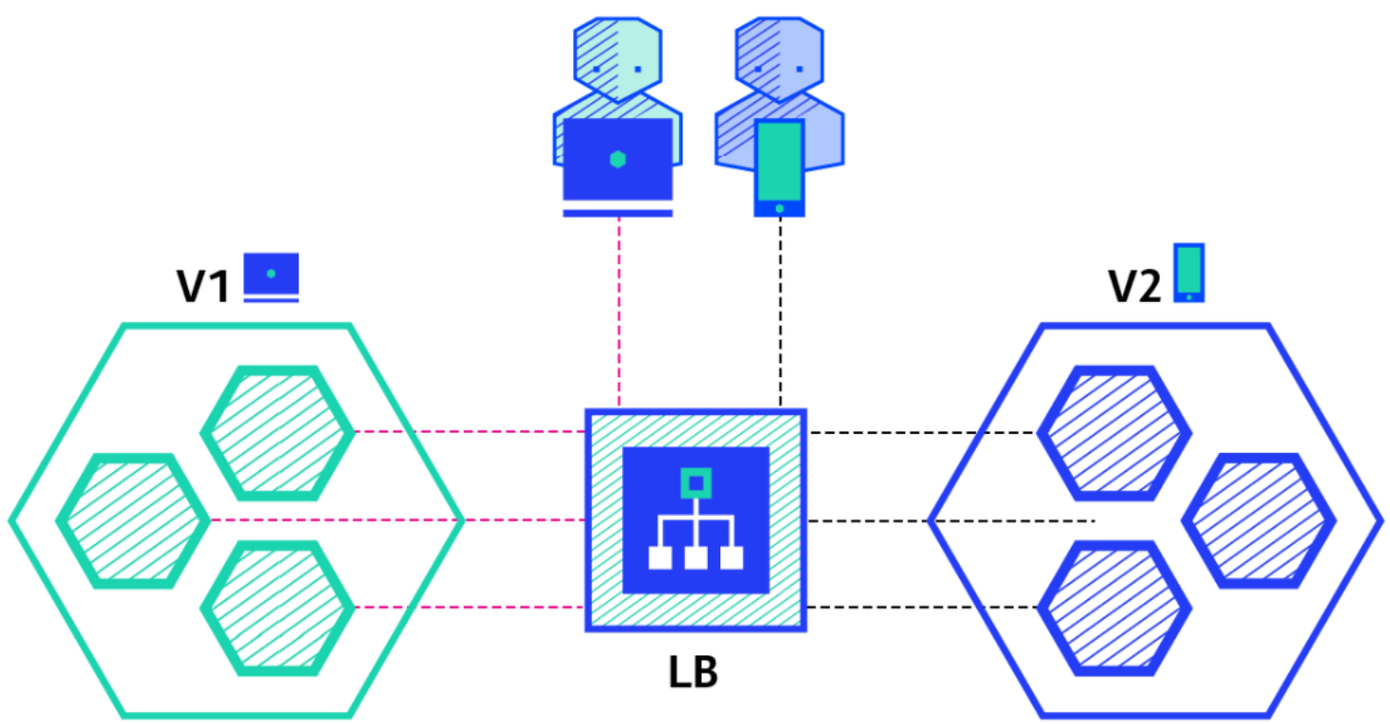
Canary Release

Canary Release



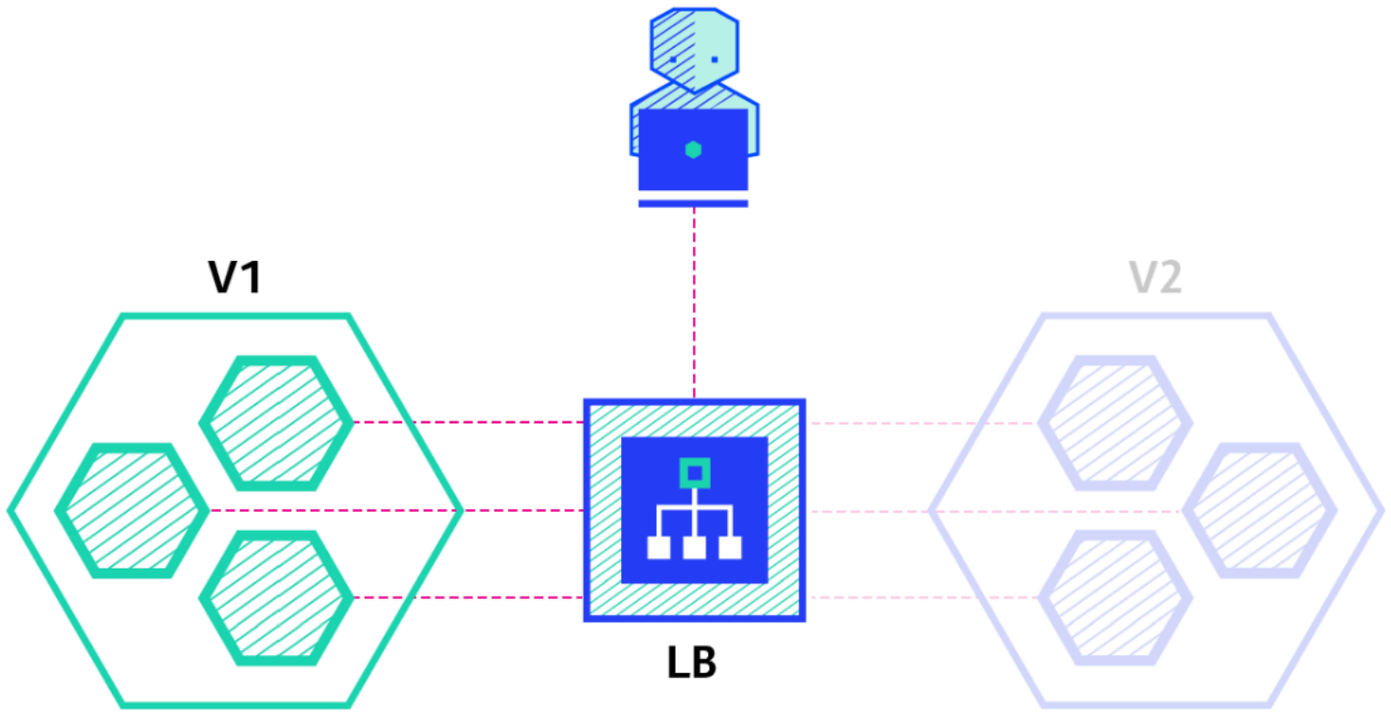
5. A/B testing

Routing the subset of users to new services under the specific condition



6. Shadow

Release version B alongside version A
Send request's A to B without production impact



DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

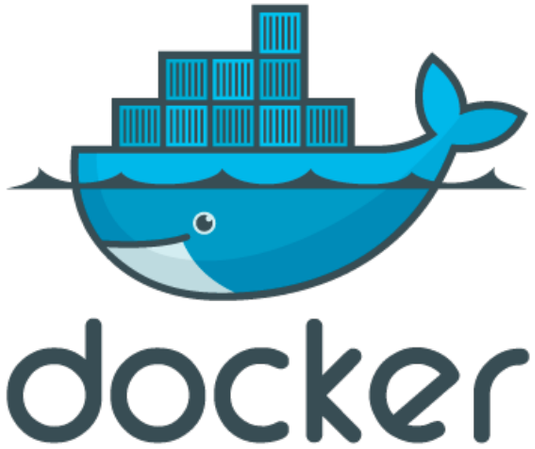
Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Container Solutions

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

Deployment workshop with



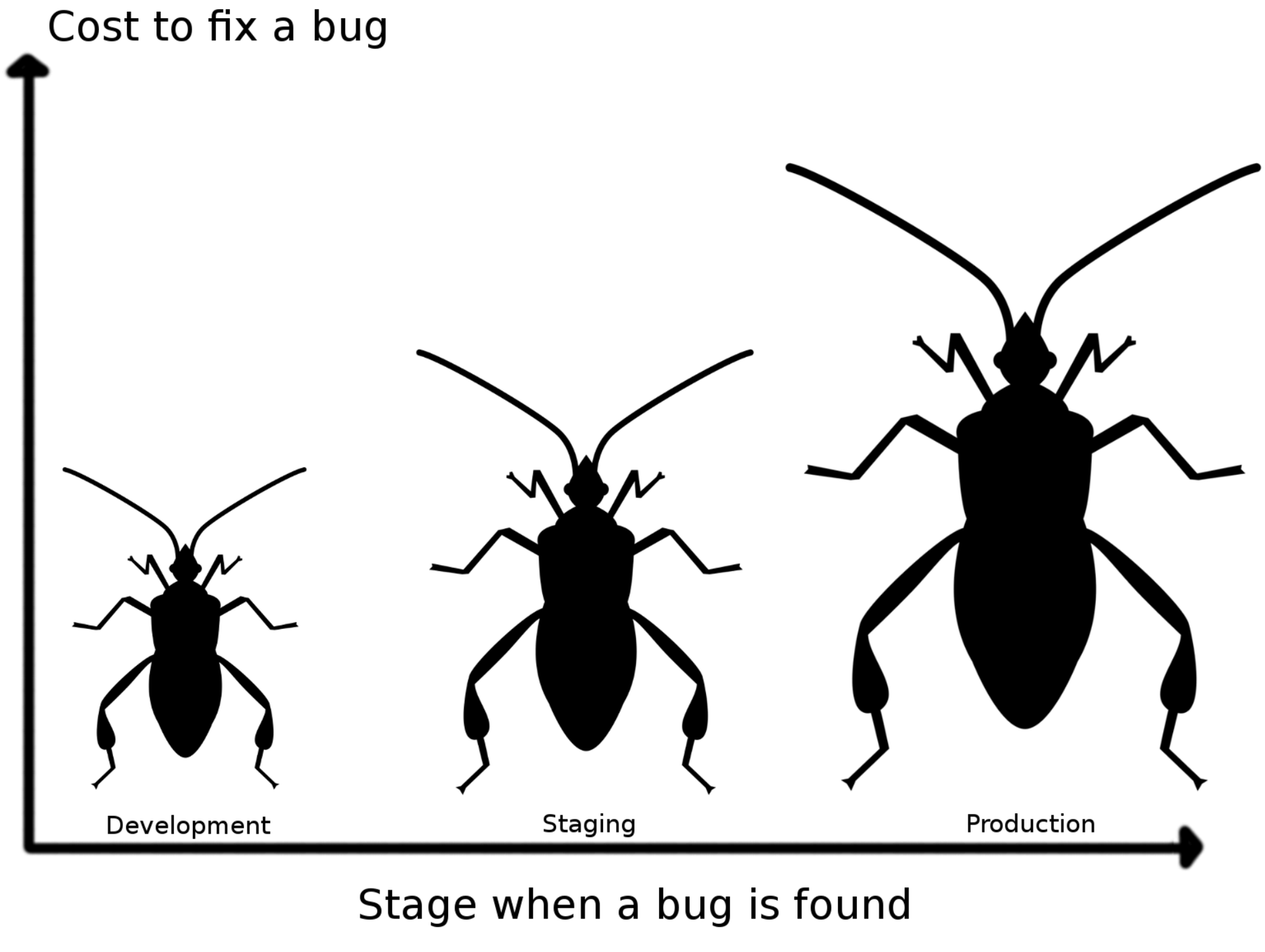
https://github.com/up1/workshop-develop-microservices-2023

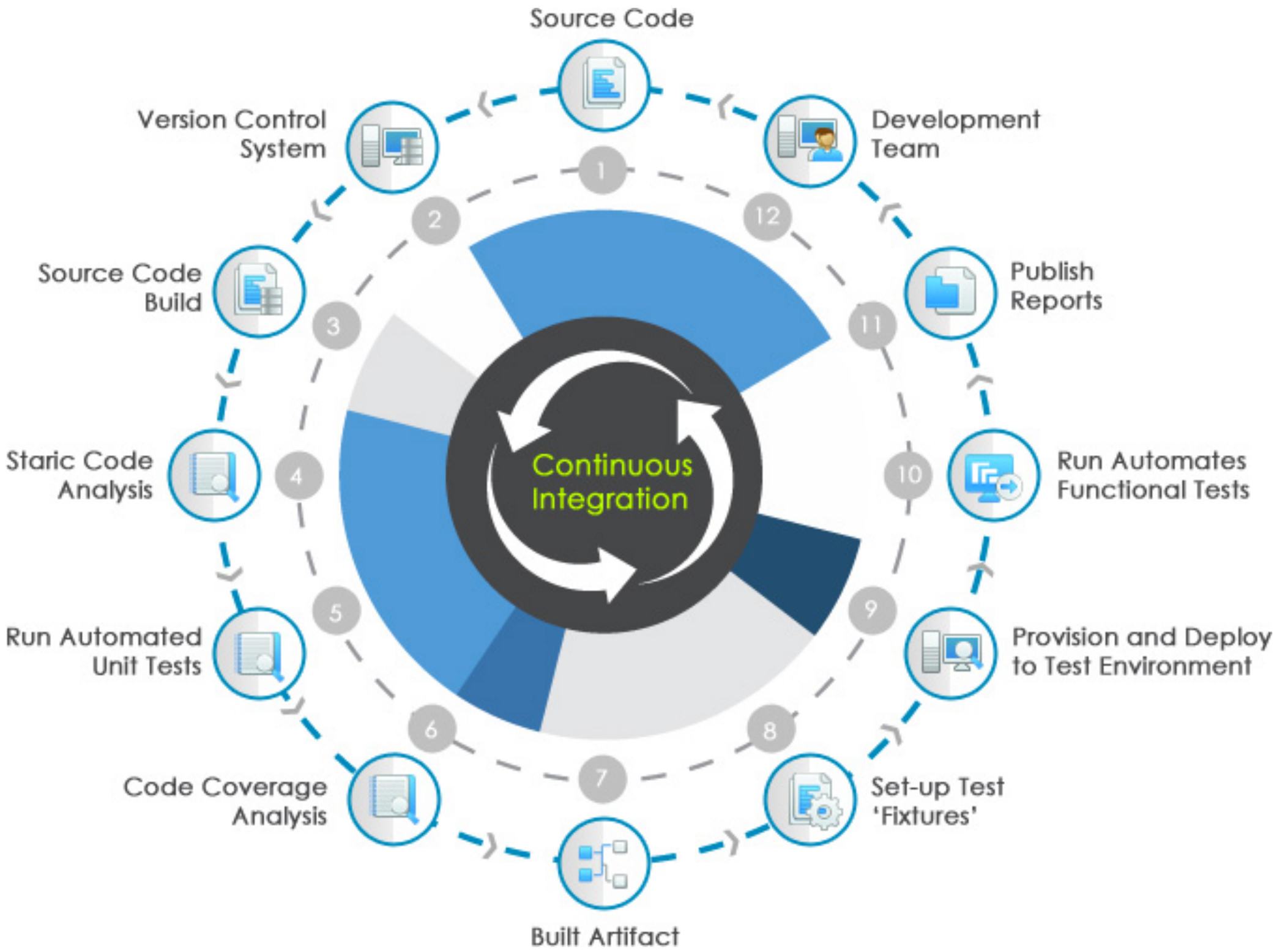
Design CI/CD process

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





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

Continuous Integration

Discipline to **integrate frequently**

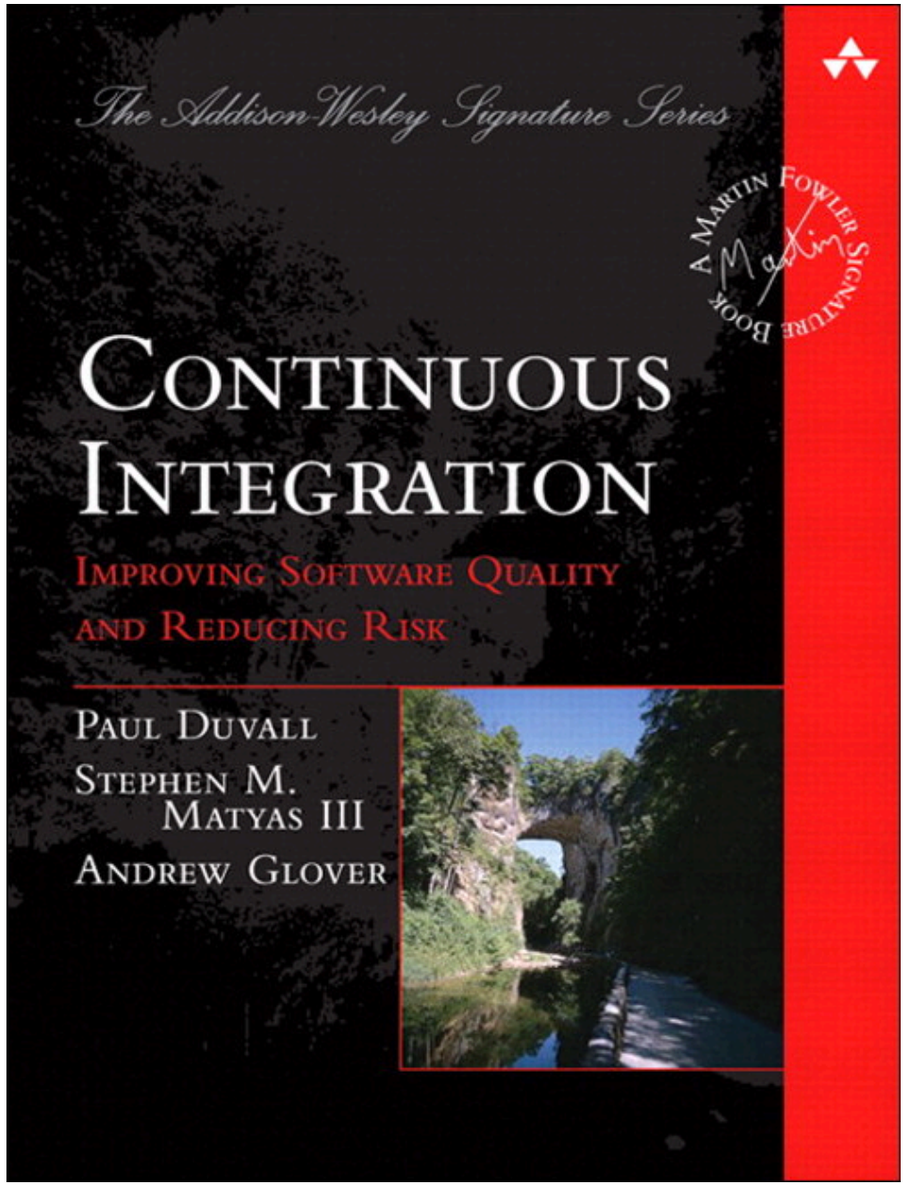
Continuous Integration

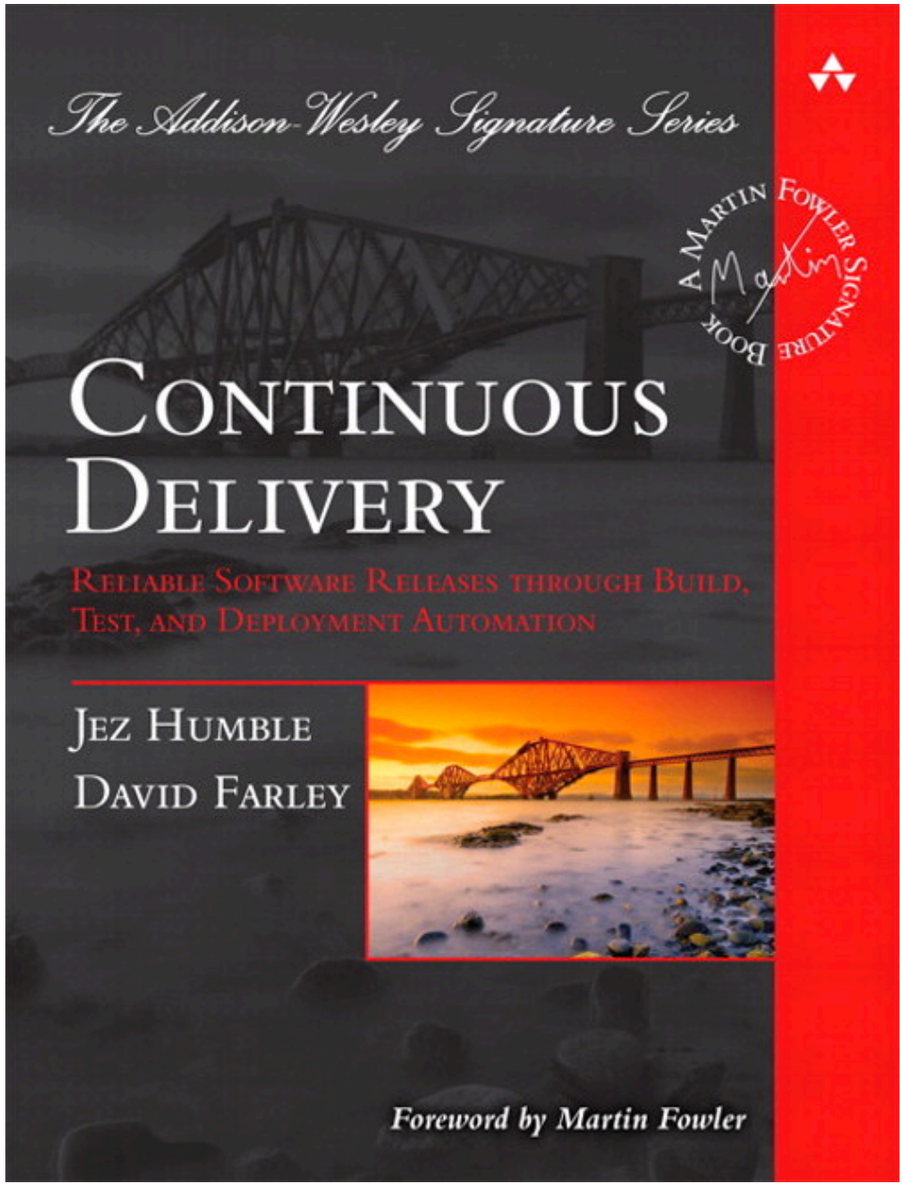
Strive to make **small change**

Continuous Integration

Strive for **fast feedback**

Improve quality and reduce risk



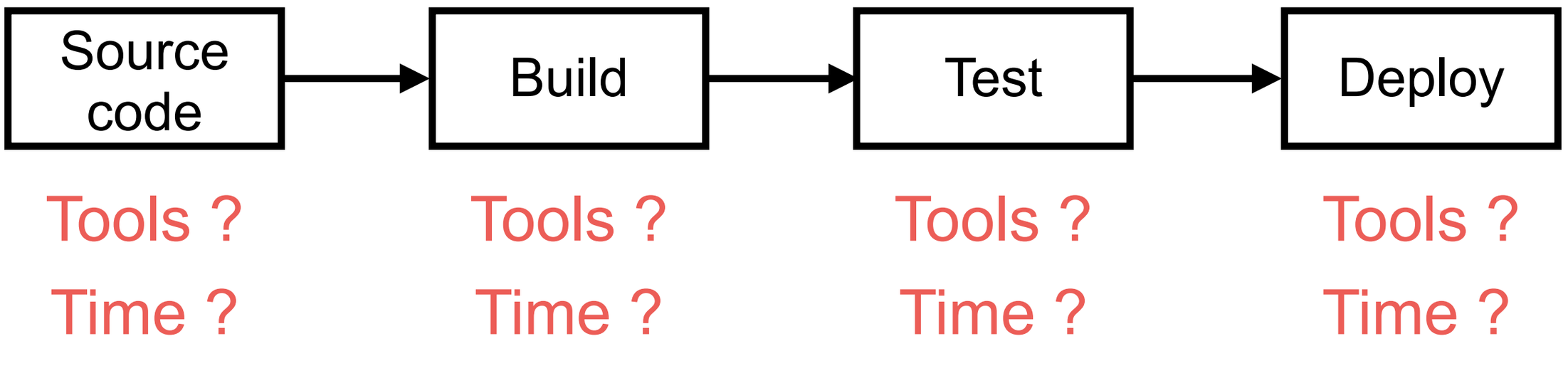


Design pipeline

Path to Production !!

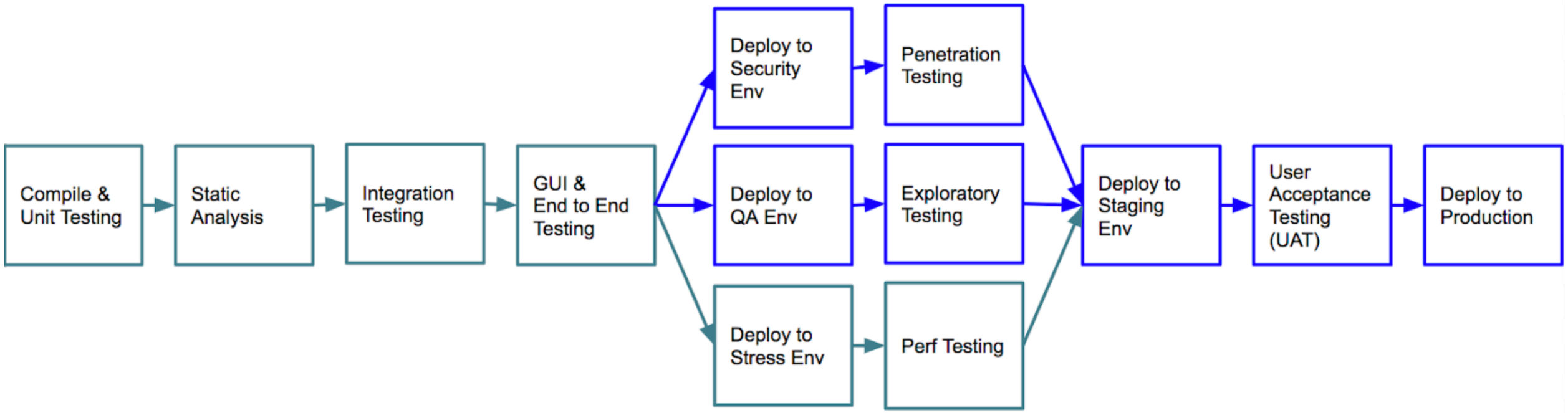
Pipeline of services

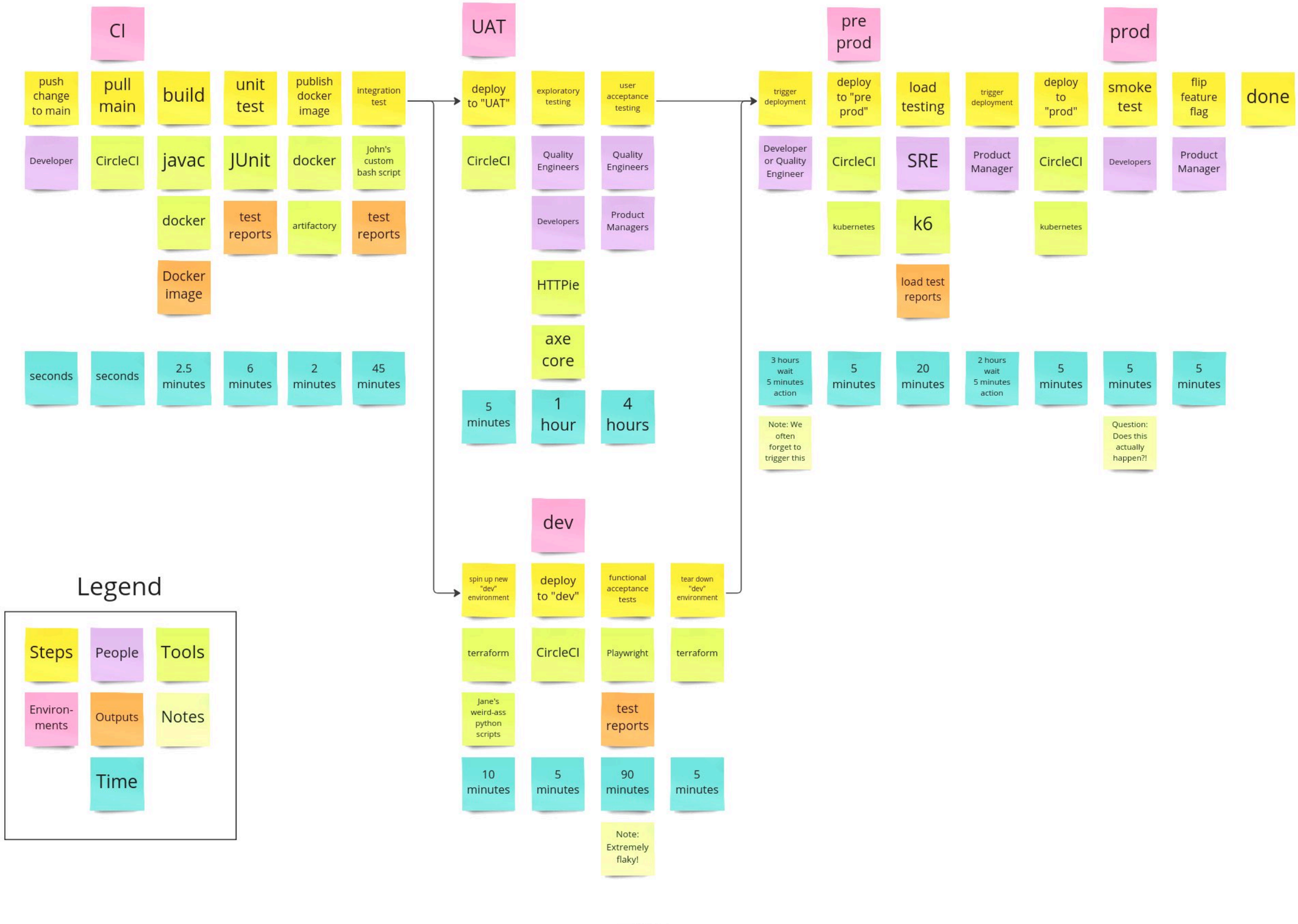
Fast feedback and high quality
Automation process



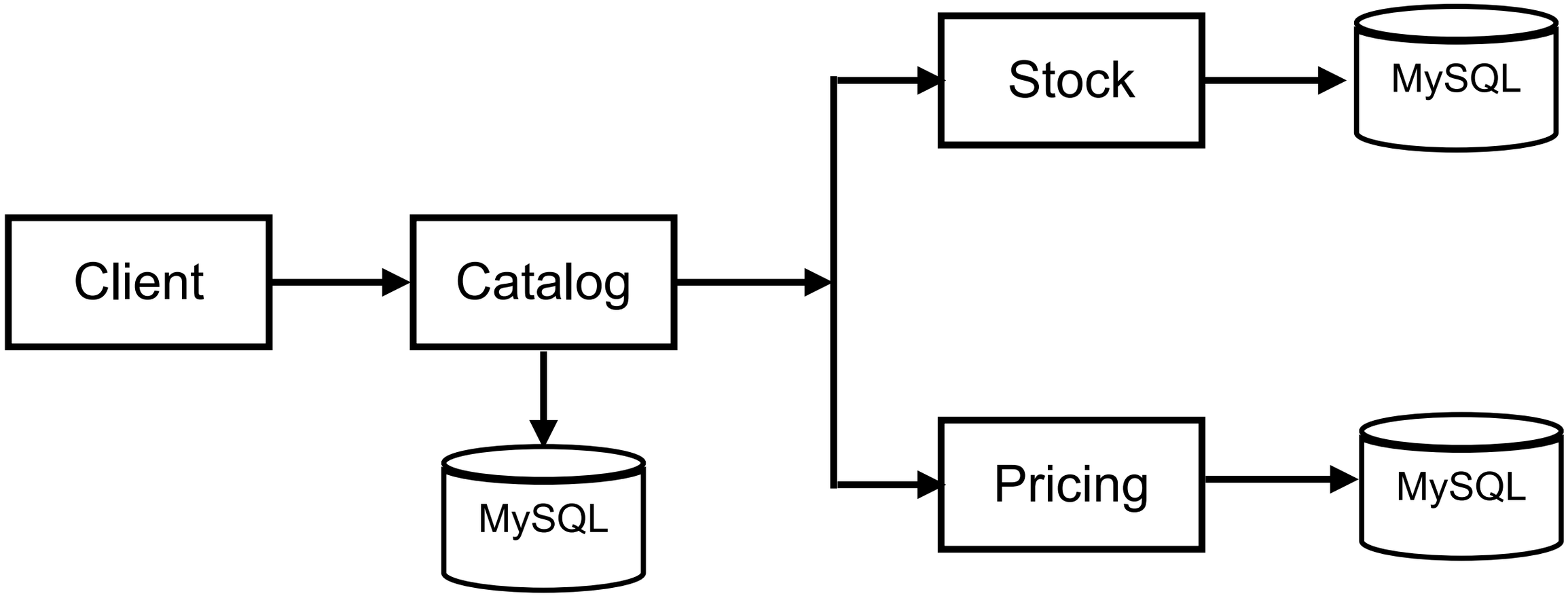
Path to Production ?

Improve your deployment/delivery pipeline





How to design pipeline ?



CI/CD pipeline

Create pipeline workshop

Jenkins

https://github.com/up1/workshop-develop-microservices-2023

Q/A