

12-Factor !!



**[https://github.com/up1/
course-12-factors-app](https://github.com/up1/course-12-factors-app)**



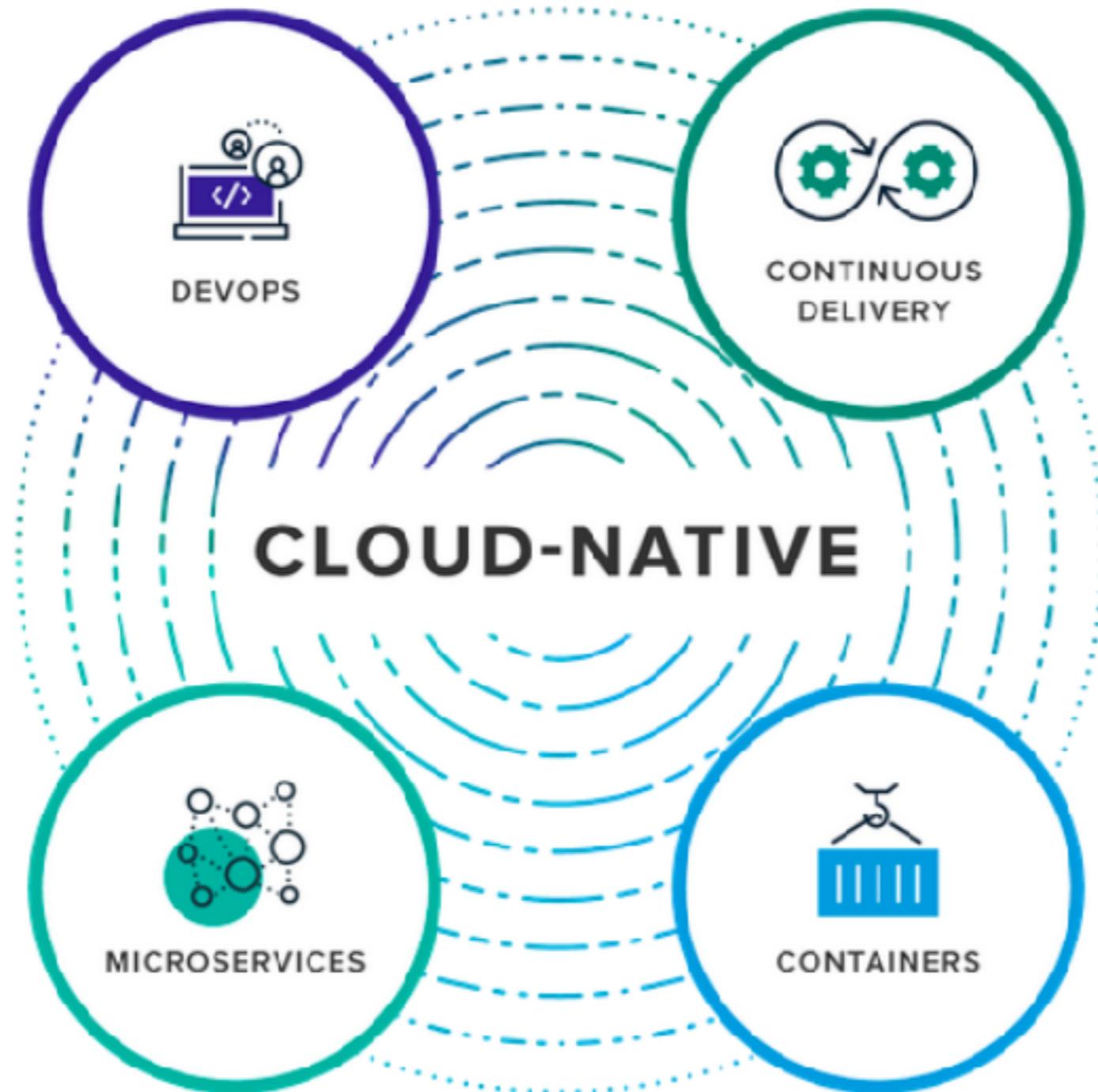
12 Factor App

<https://12factor.net/>



Foundation to build Cloud Native Application





Cloud Native Application

Increase efficiency
Reduce cost
Ensure availability

<https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/definition>



12 Factor App

Methodology for building software-as-a-service app

Declarative formats for setup automation

Portability

Deployment on modern platform

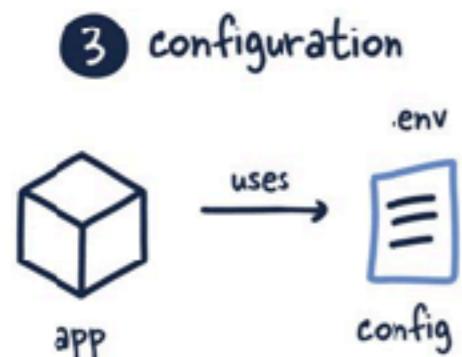
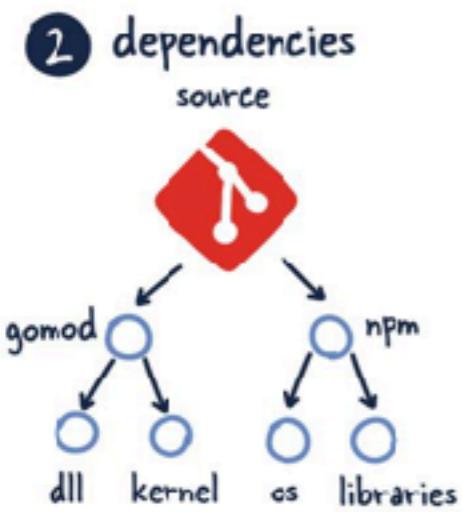
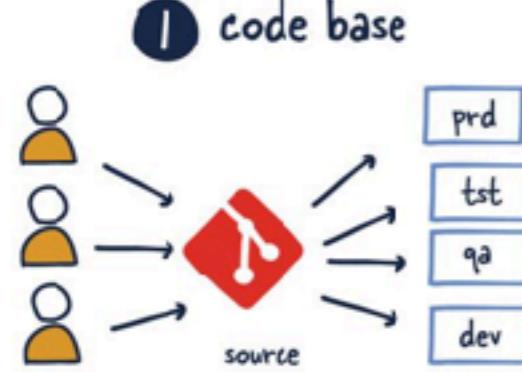
Minimize divergence (dev vs prod)

Easy to scale up/out

<https://12factor.net/>

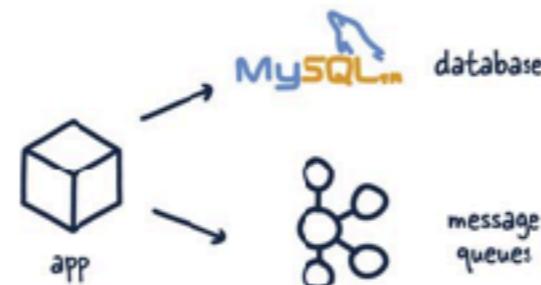


12 FACTOR APP REVISITED

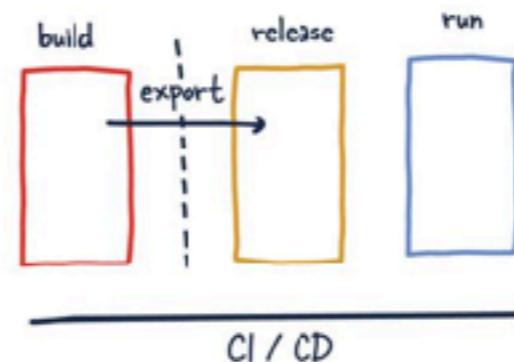


architecture notes

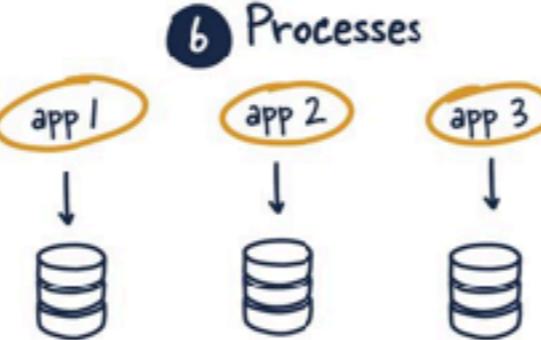
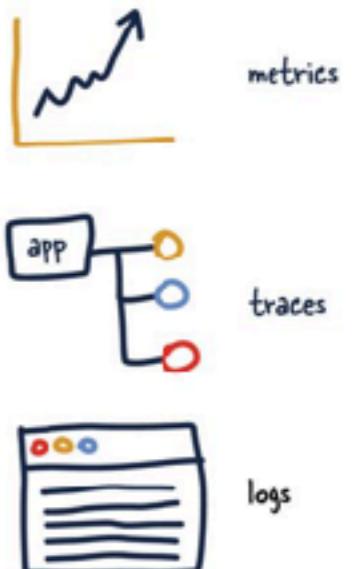
4 backing services



5 build, release, run



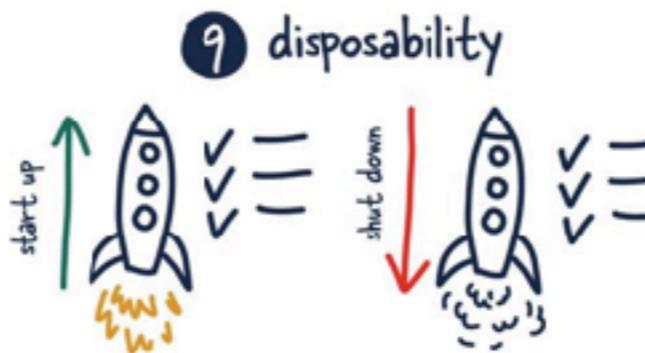
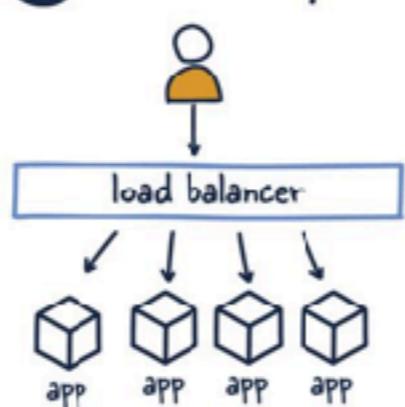
11 logs



7 port binding



12 admin processes



10 dev/prod parity

dev = prd or docker

12 + 3 Factor



<https://www.handsonarchitect.com/2022/08/cloud-native-fifteen-factor-apps.html>



12 Factor

Build

Scalable

Maintainable

Portability

Security

Minimize
divergence

Monitoring/Observability



12 Factor

Build

Scalable

Maintainable

1. Codebase
2. Dependencies
3. Config
4. Backing services
5. Build release run

API first



12 Factor

Build

Scalable

Maintainable

- 6. Processes
- 7. Port binding
- 8. Concurrency
- 9. Disposability



12 Factor

Build

Scalable

Maintainable

- 10. Dev/prod parity
- 11. Logs
- 12. Admin process

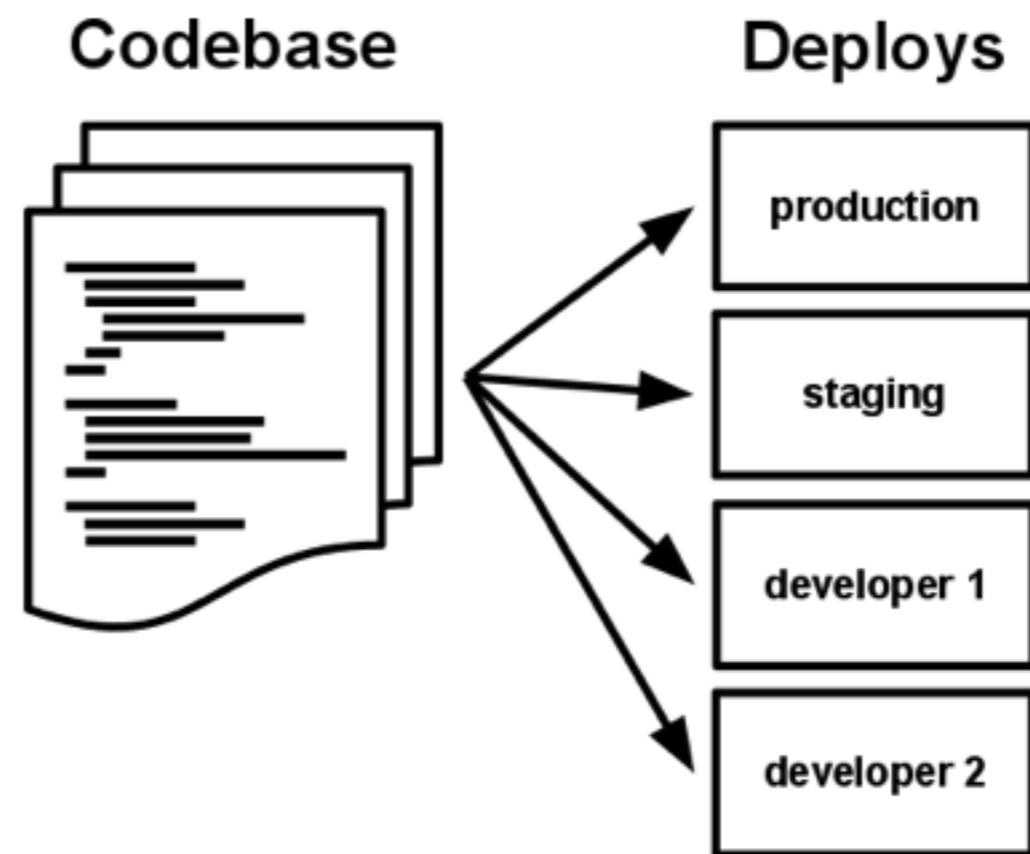


1. Codebase



1. Codebase

One codebase tracked in revision control, many deploys



Version Control System



<https://git-scm.com/>



Git Branching Strategies

by levelupcoding.co

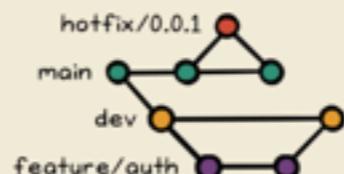
Feature branching

Each new feature has its own branch. Once the changes are complete and merged in, the branch can be deleted.



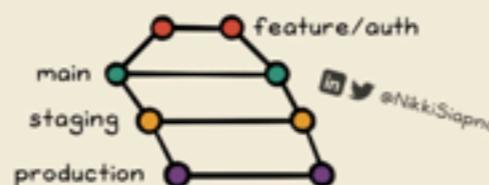
Gitflow

Has branches for features, releases, hotfixes, and a dedicated branch for production and development.



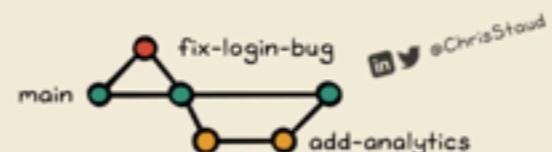
GitLab Flow

Branches are created for features, releases, and environments. The main branch is always production-ready.



GitHub Flow

Branches are created for new features, bug fixes, and experiments. The main branch is the only deployable branch, it is kept production-ready.



@NikkiSiapno

Brought to you by

LEVEL UP
CODING

Trunk-based

All branches other than the main branch are short-lived and merged in within a defined timeframe (usually a day). Large features are built incrementally and hidden behind feature flags.

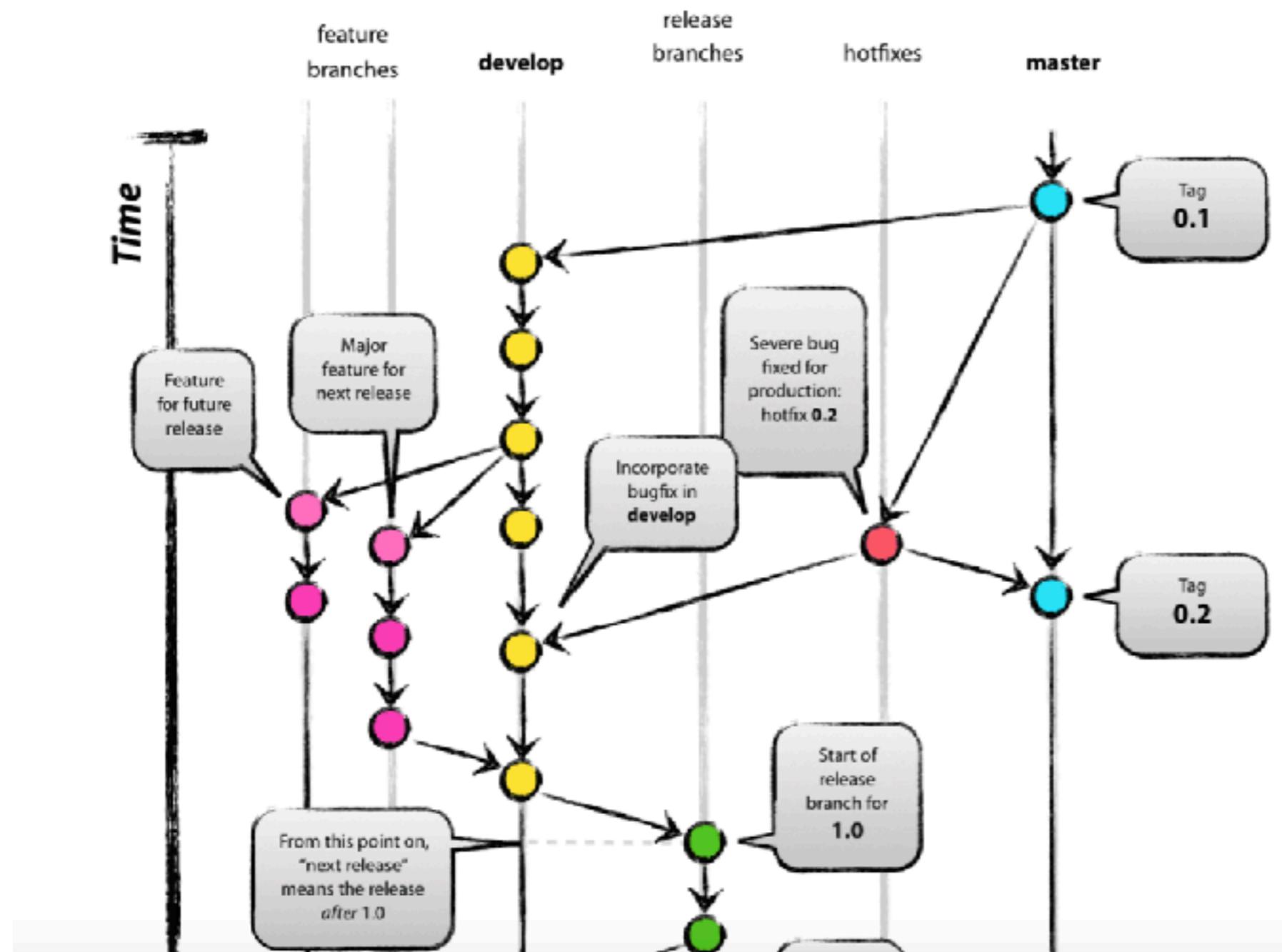


@ChrisStaud

<https://twitter.com/ChrisStaud/status/1752341883198853588>



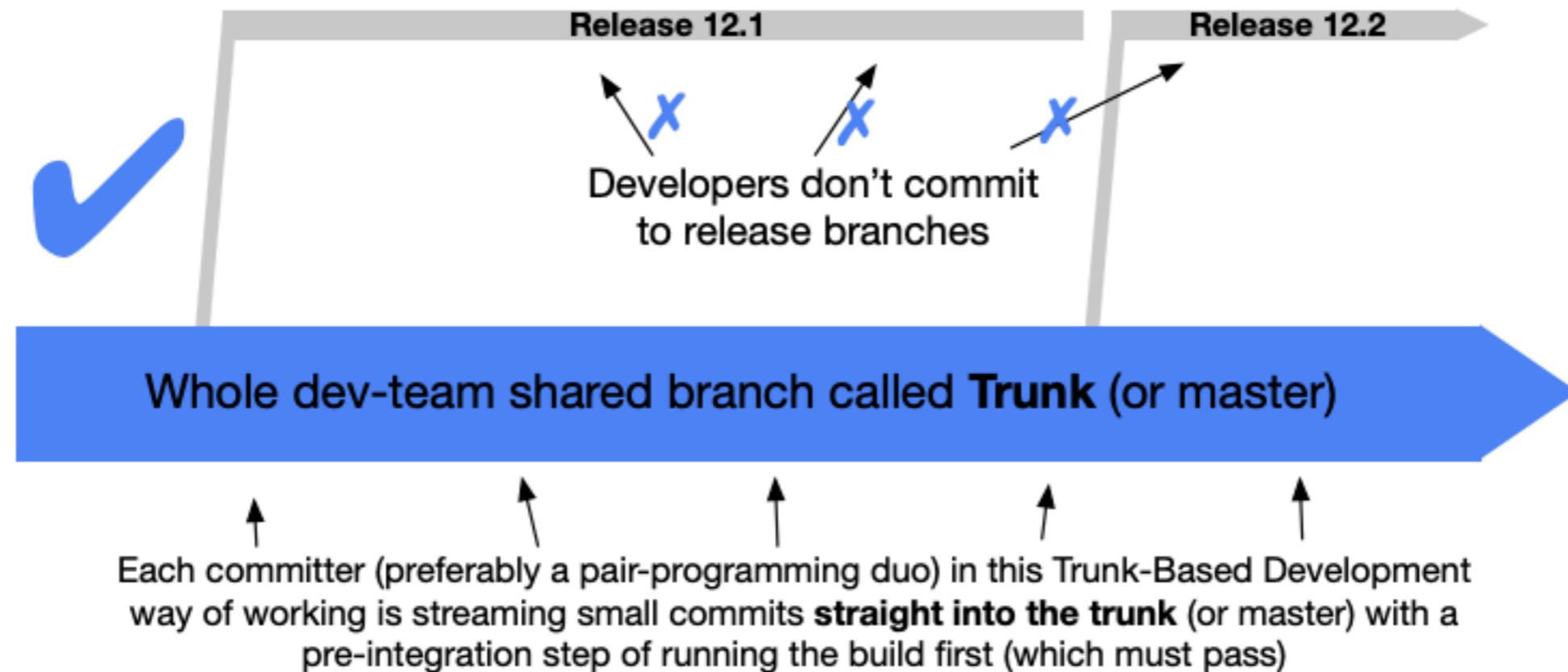
Git branching model



<https://nvie.com/posts/a-successful-git-branching-model/>



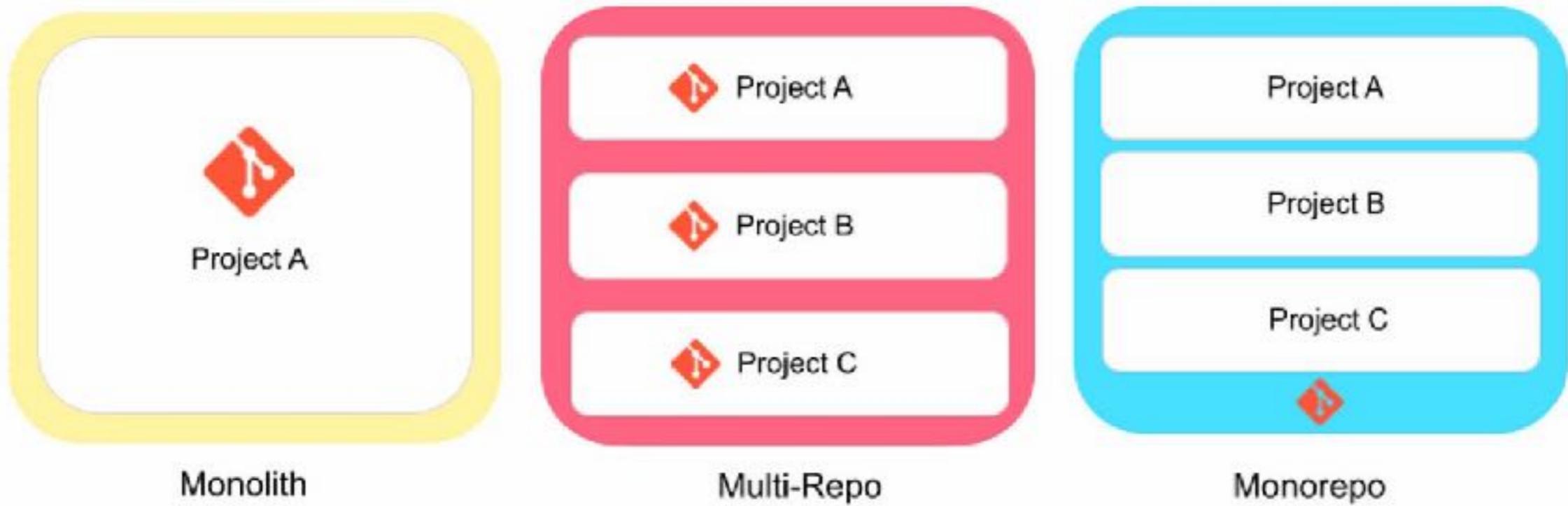
Trunk Based Development



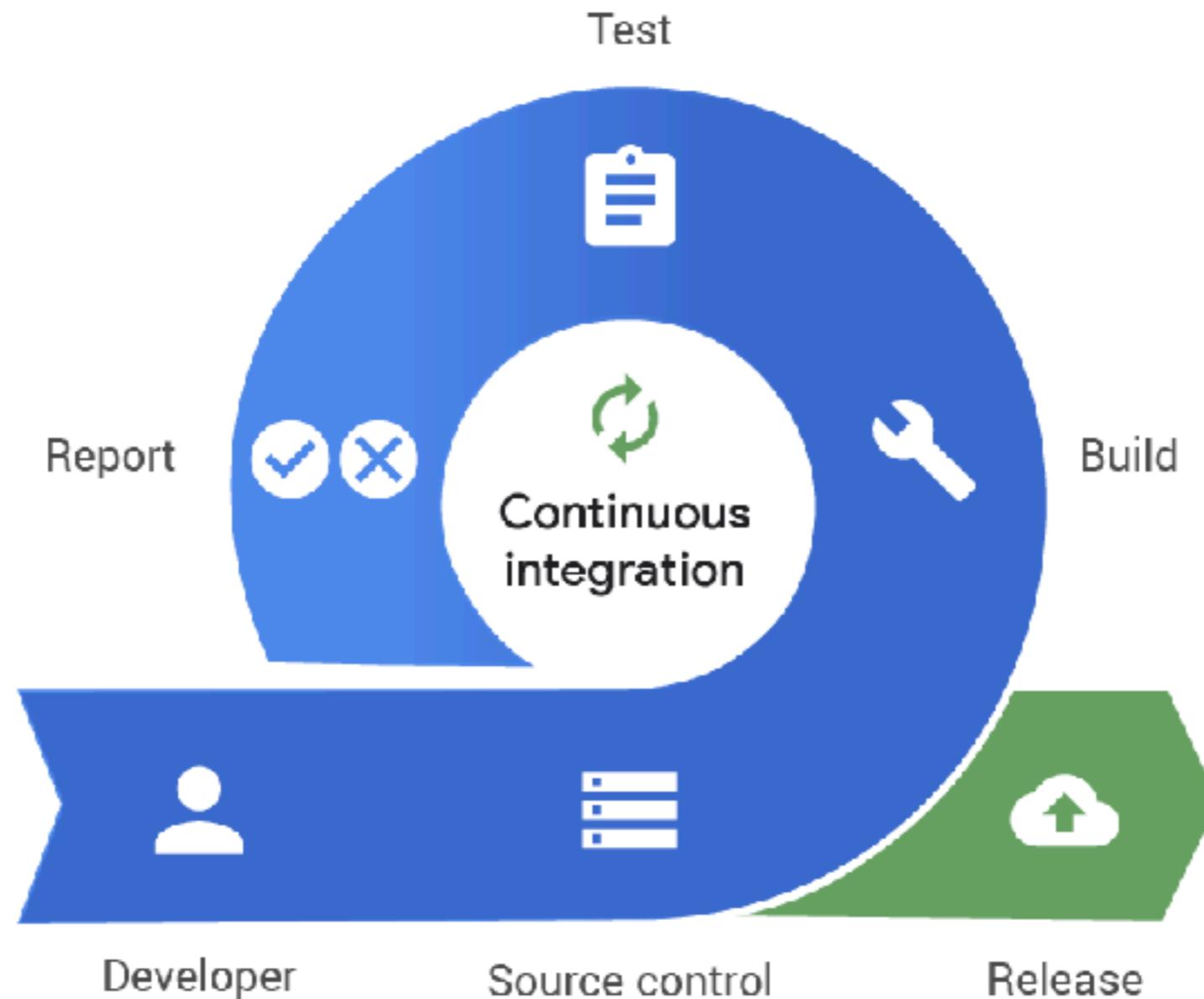
<https://trunkbaseddevelopment.com/>



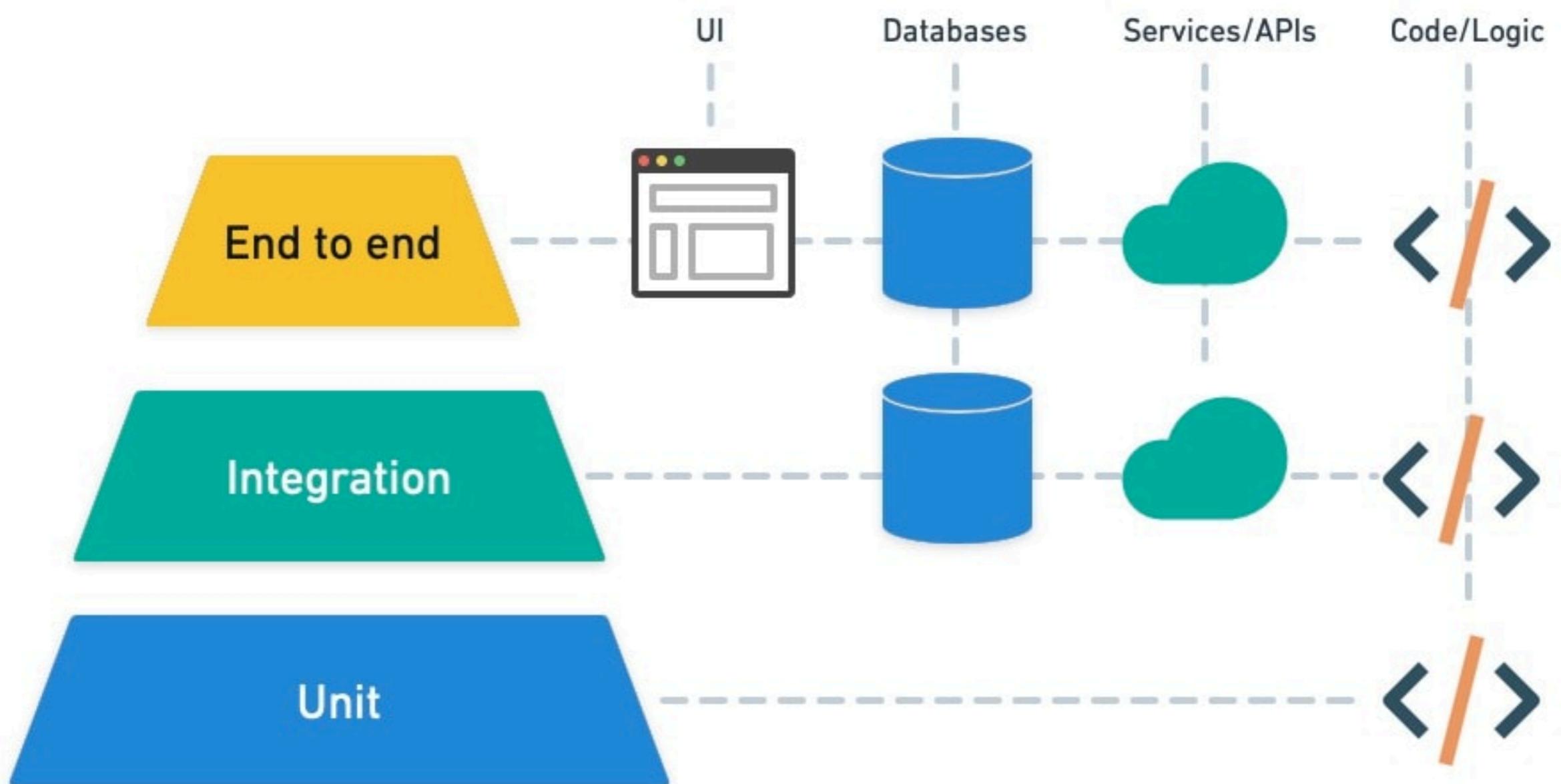
Repository ?



Continuous Integration



Automated Testing



2. Dependencies



2. Dependencies

Explicitly declare and isolate dependencies
Dependency management



3. Config



3. Config

Store config in environment

Code

Config

Credential

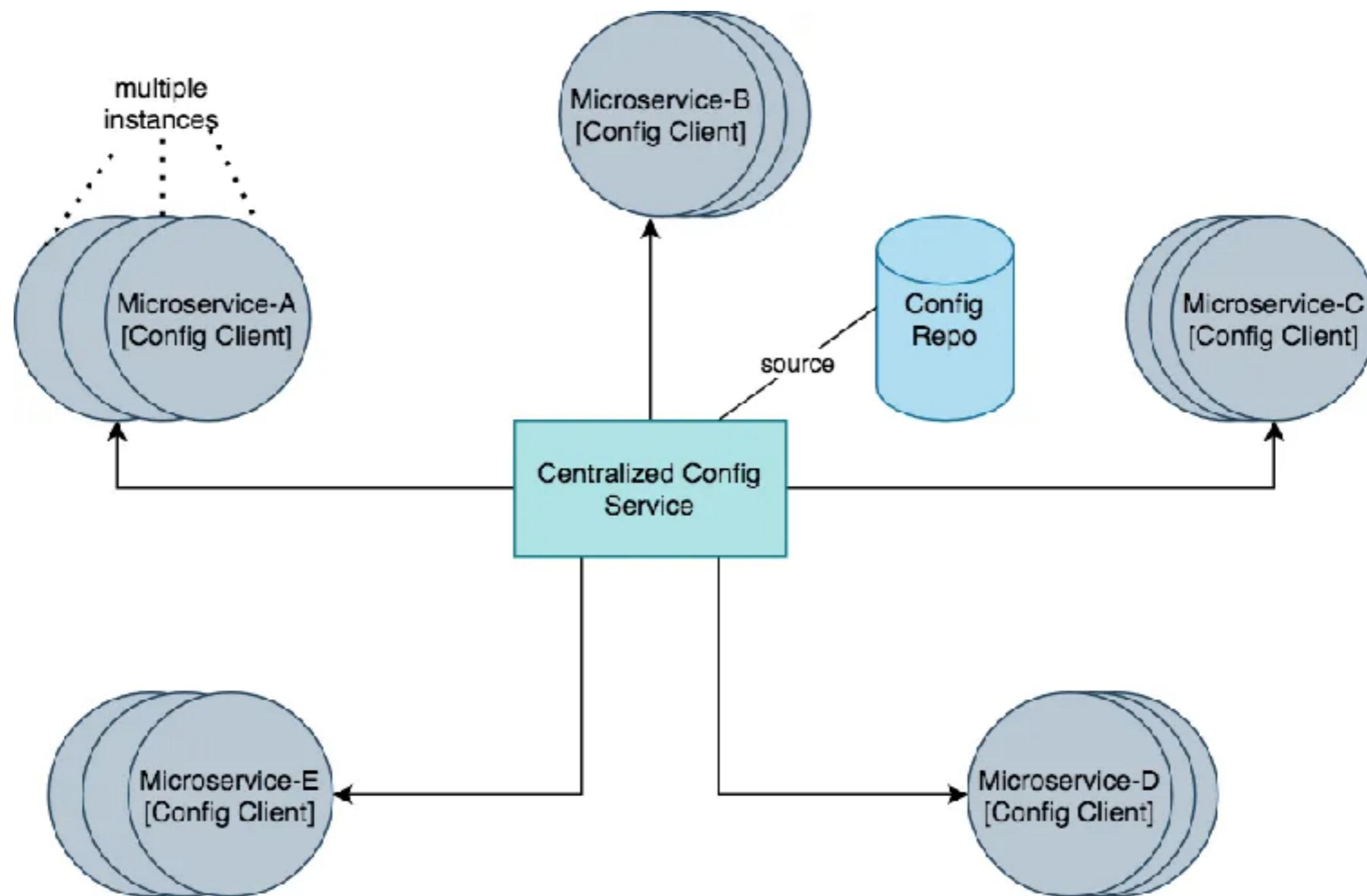


Configuration management

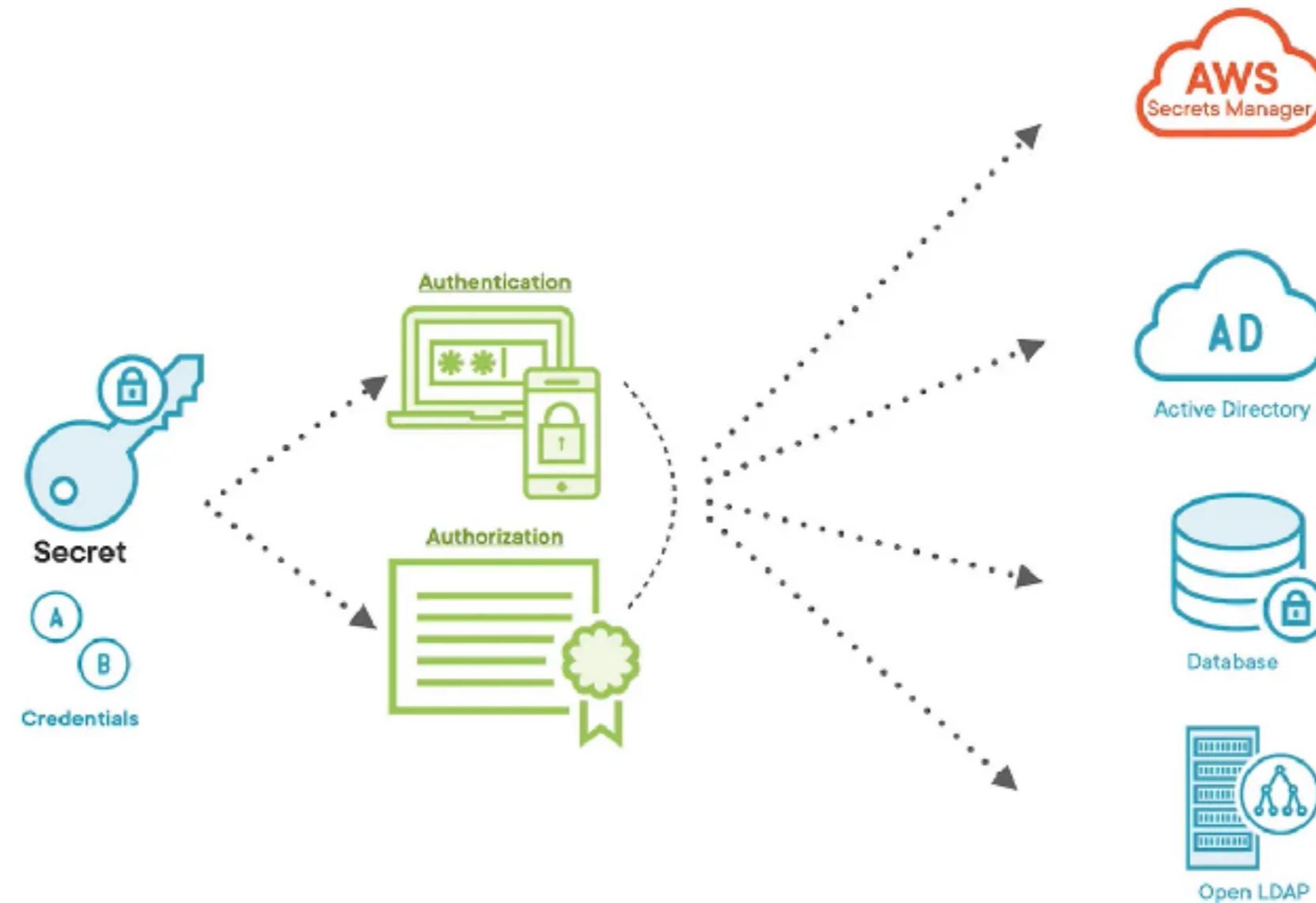
Environment variables (.env file)
Centralized management



Centralized ?



Credential/Secret management

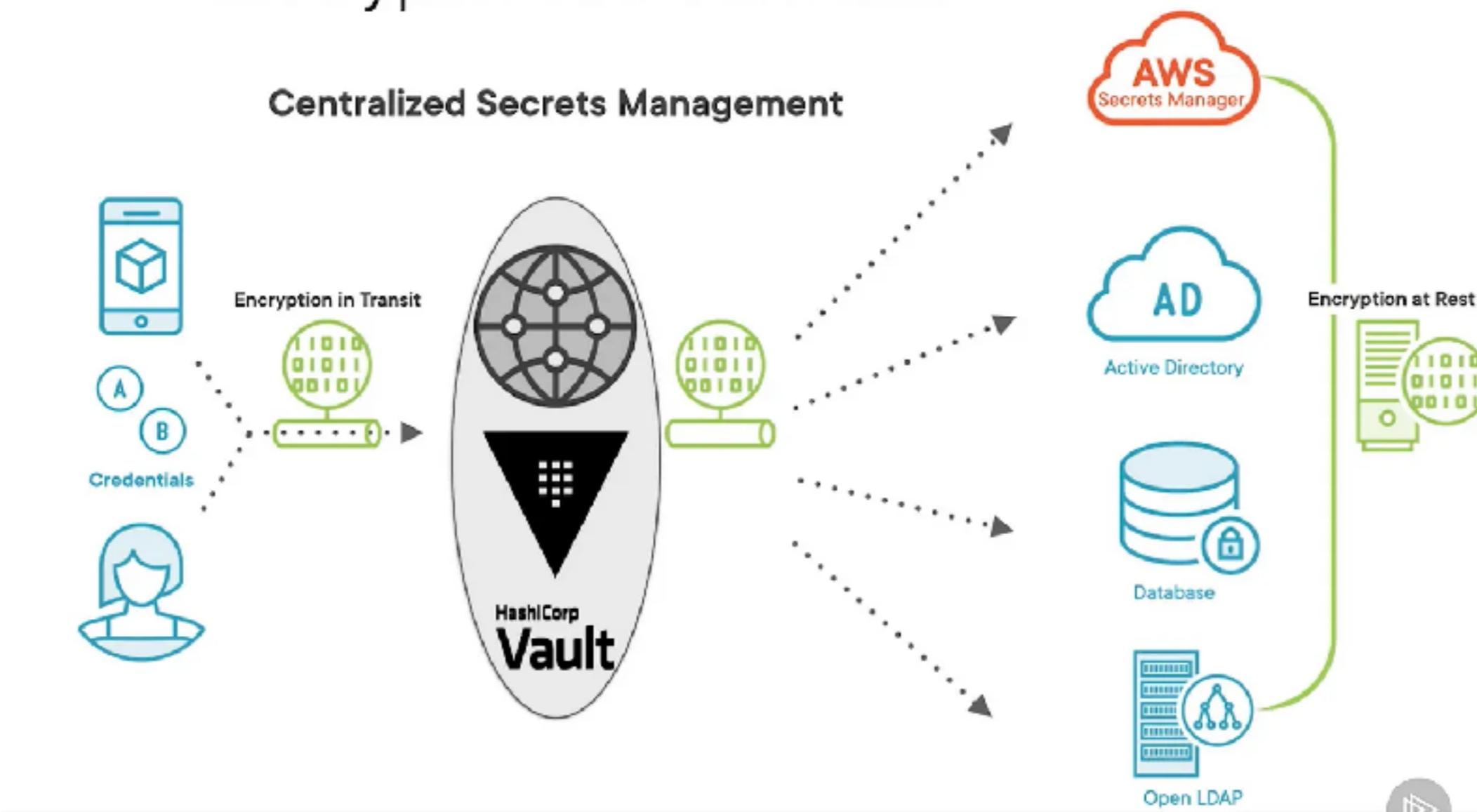


<https://medium.com/@pgpg05/secret-management-in-ci-cd-pipeline-982846596181>



HashiCorp Vault

Encryption as a Service



<https://medium.com/@pgpg05/secret-management-in-ci-cd-pipeline-982846596181>



4. Backing services



4. Backing services

Treat backing services as attached resources

Expose via addressable URL

Decouple resources from app

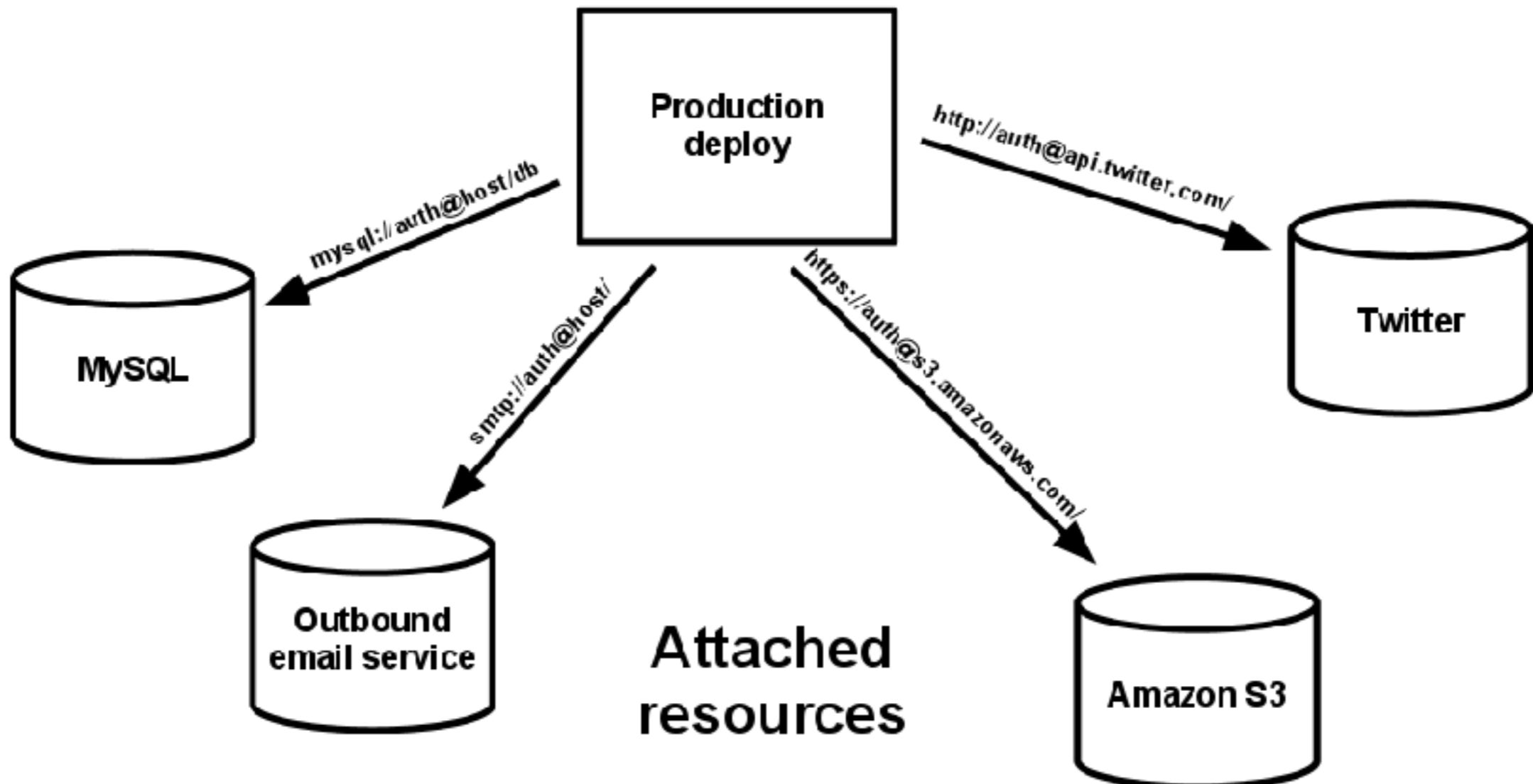
Database

Caching

Messaging



Backing services

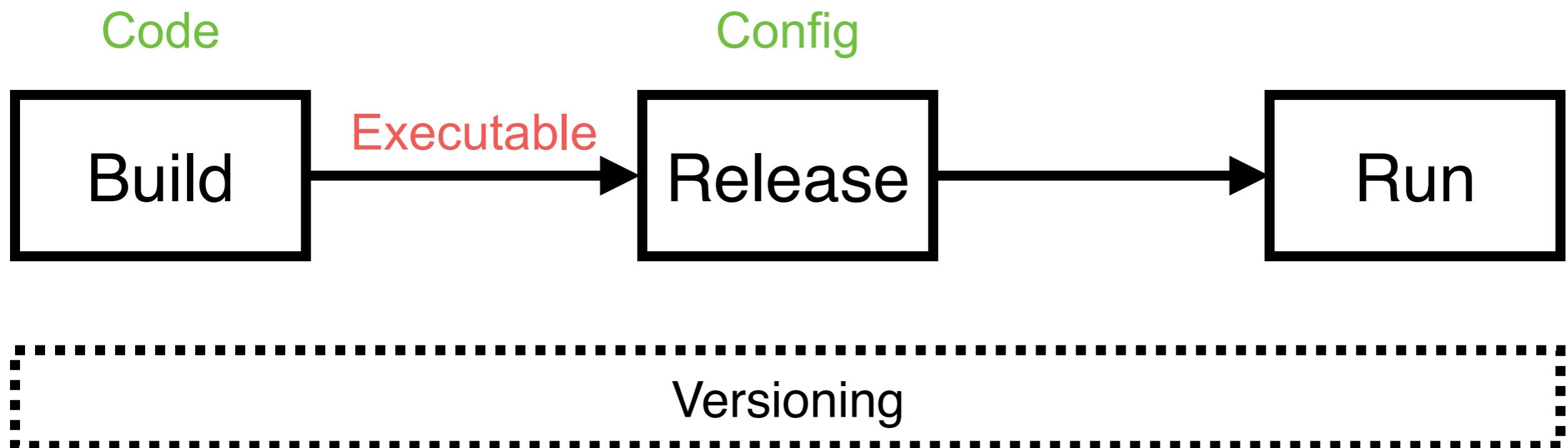


5. Build, release, run



5. Build, release, run(time)

Strictly separate build and run stages



6. Processes



6. Processes

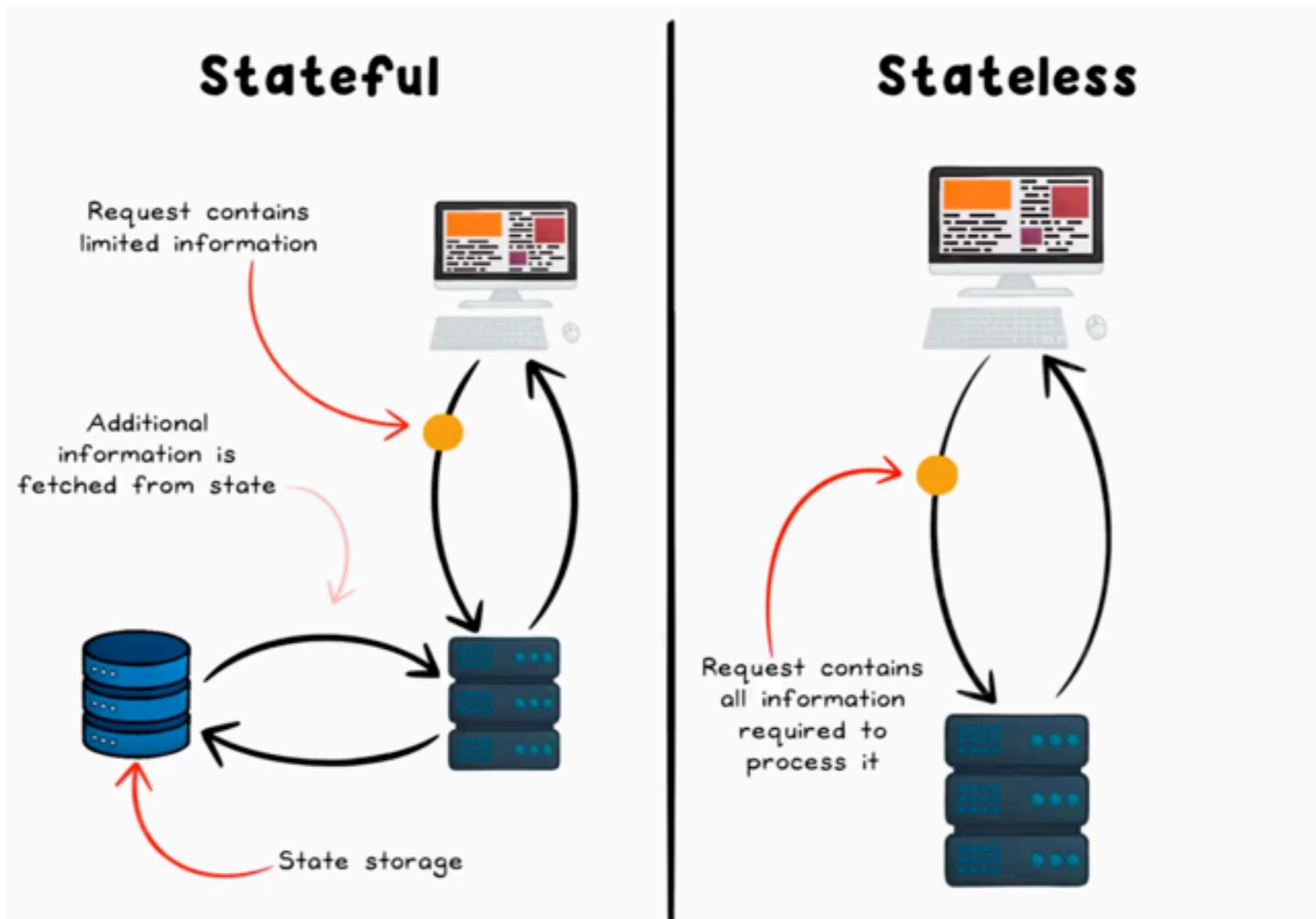
Execute the app as one or more **stateless** processes

Shared nothing

Isolated from other running services



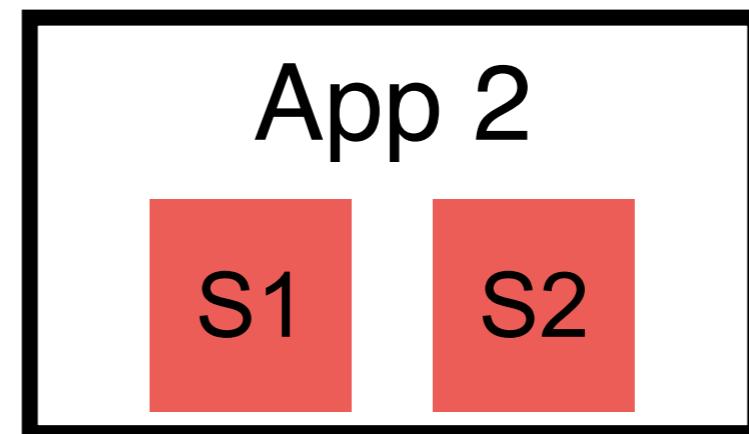
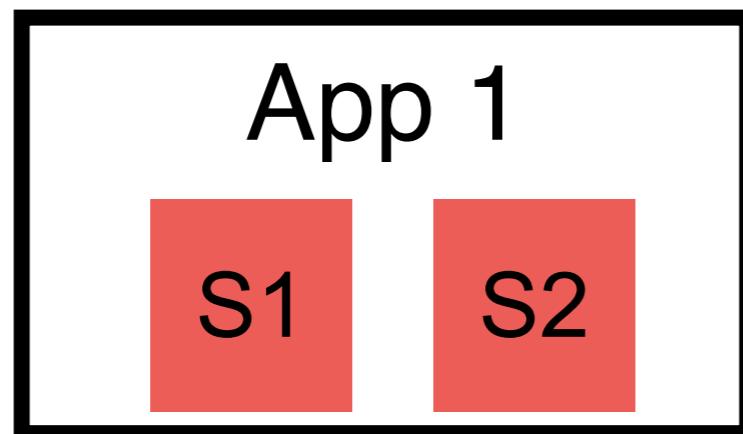
Stateful vs Stateless



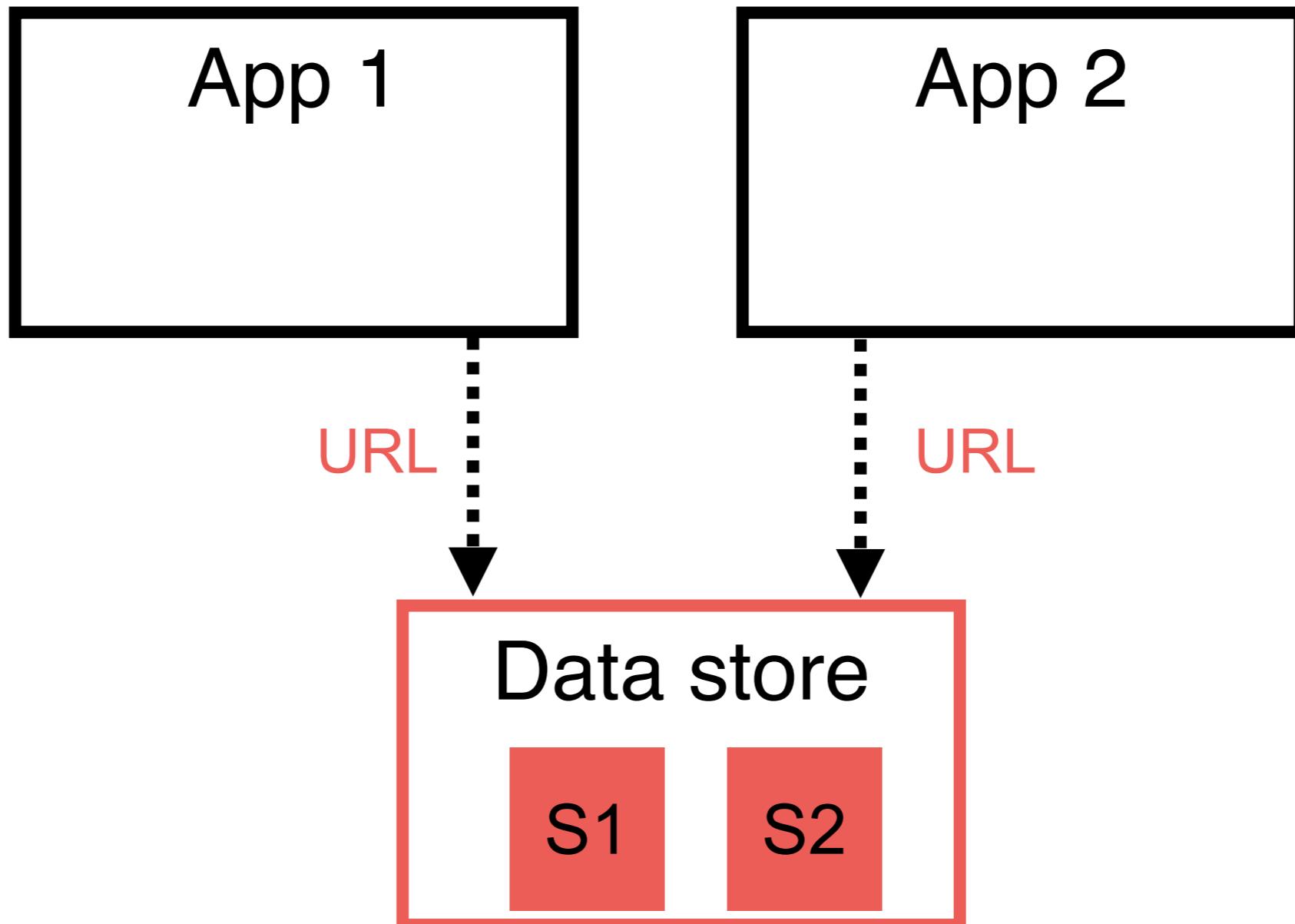
<https://x.com/ChrisStaud/status/1737481416307323316>



Isolated ?



Sharing with backing service

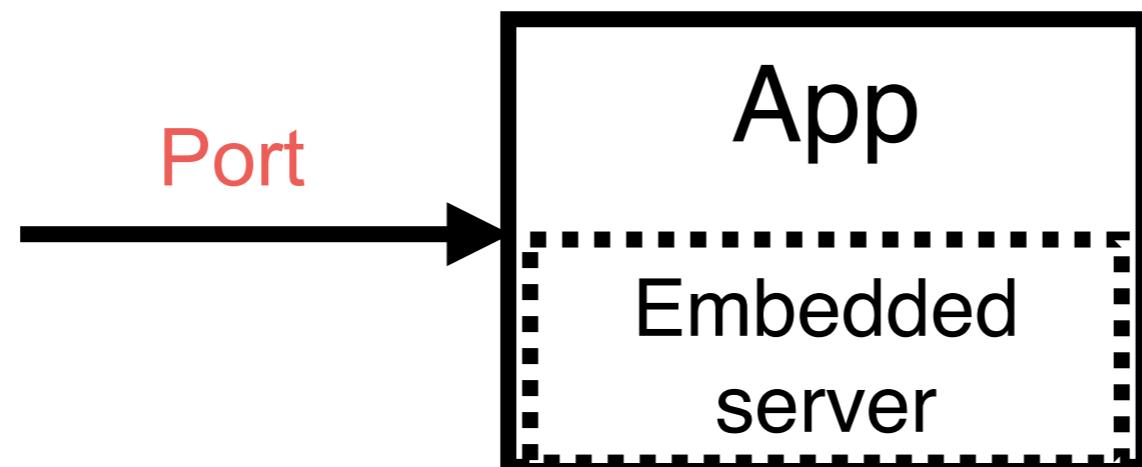


7. Port binding

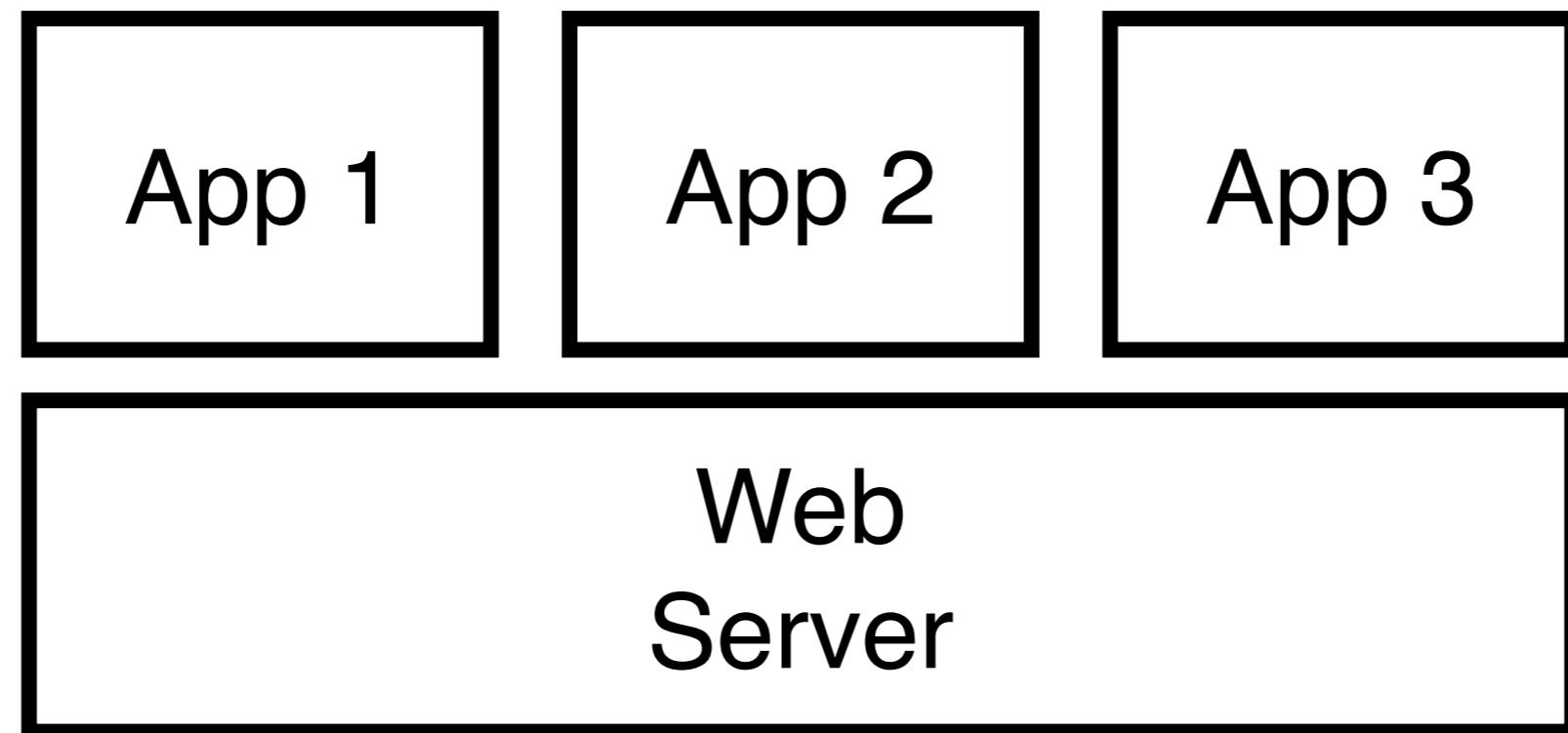


7. Port binding

Export services via port binding
Self-contained



Don't

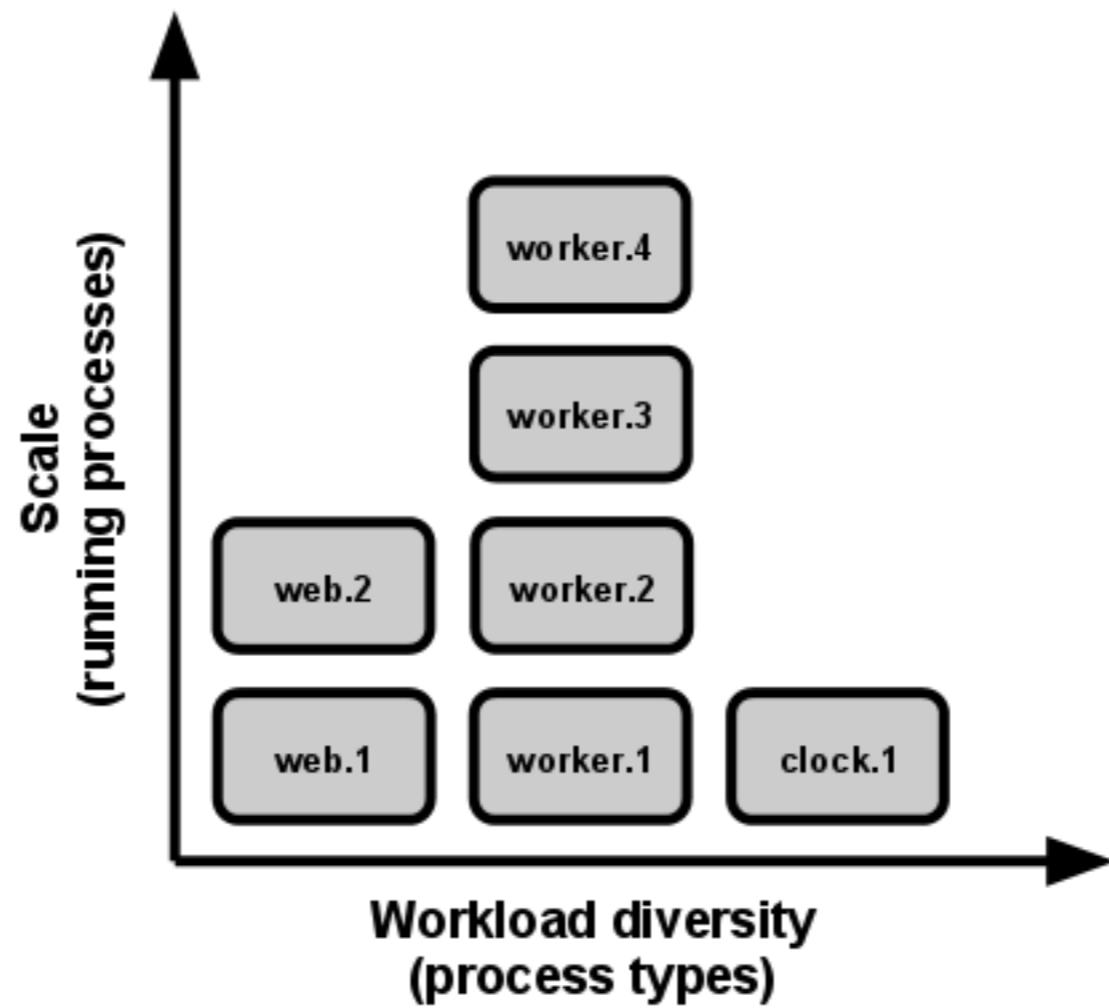


8. Concurrency



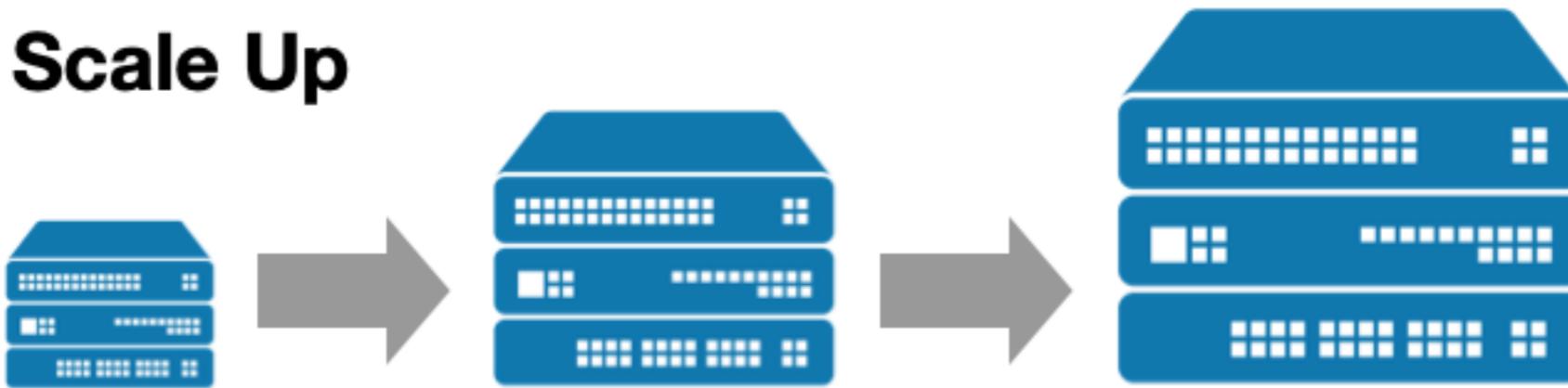
8. Concurrency

Scale out via the process model



Scaling ?

Scale Up

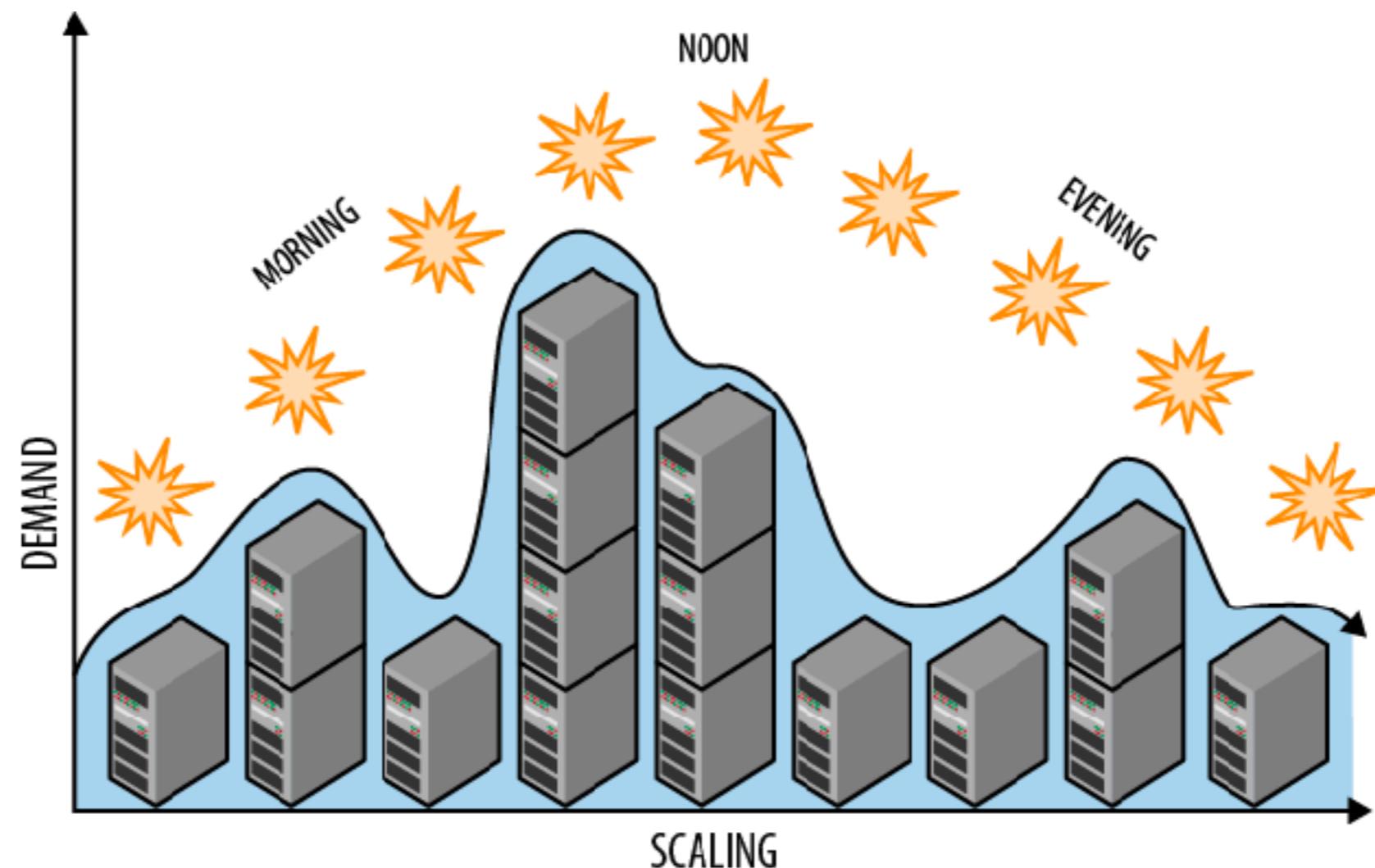


Scale Out



Elasticity Scaling

Reduce cost and improve performance by scaling automatically !!



9. Disposability



Disposable

อังกฤษ ▼ ↔ 泰语 ▼

disposable × แบบใช้แล้วทิ้ง
də'spōzəb(ə)l Bæb chī lâew thîng

คำแปลของ disposable

คุณศัพท์

ขยายถ่ายเทได้
disposable

จัดการได้
disposable, manageable



9. Disposability

Maximize robustness
Fast startup and graceful shutdown

Fast startup to increase scalability opportunity

Graceful shutdown to leave the system in a correct state



Why graceful shutdown ?

Resource cleanup
Transaction integrity

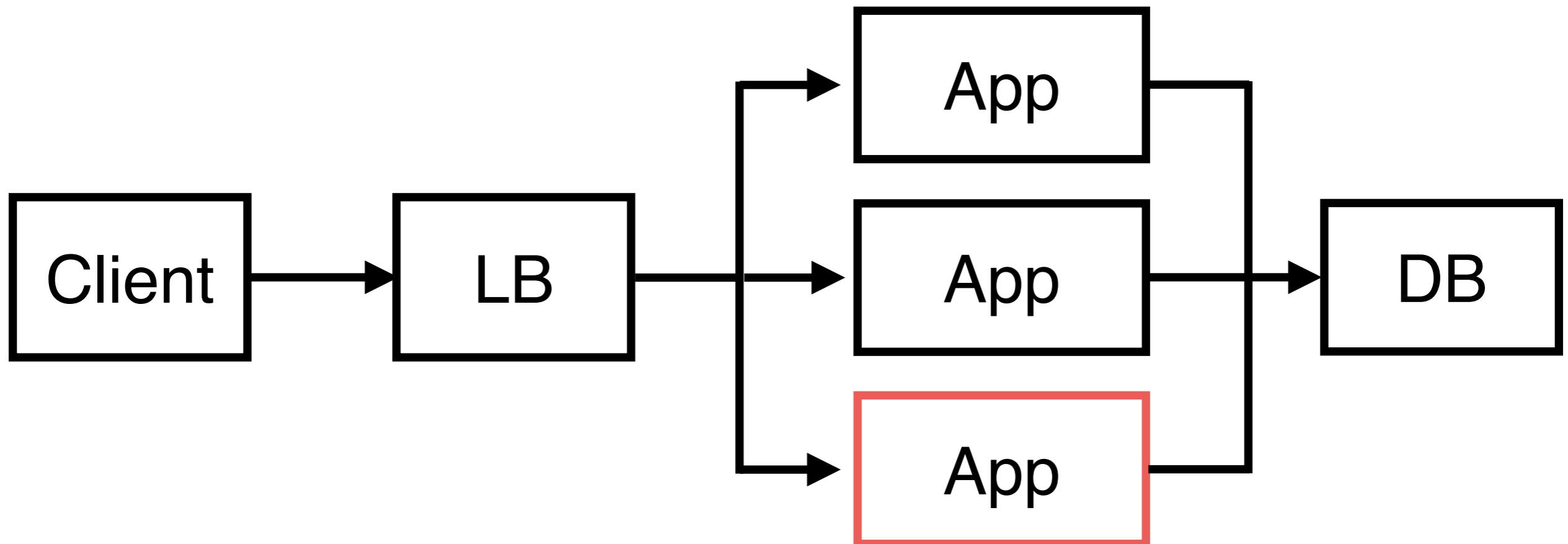
<https://levelup.gitconnected.com/maximizing-resilience-with-graceful-shutdown-in-cloud-native-golang-applications-7f0b2edef4a8>



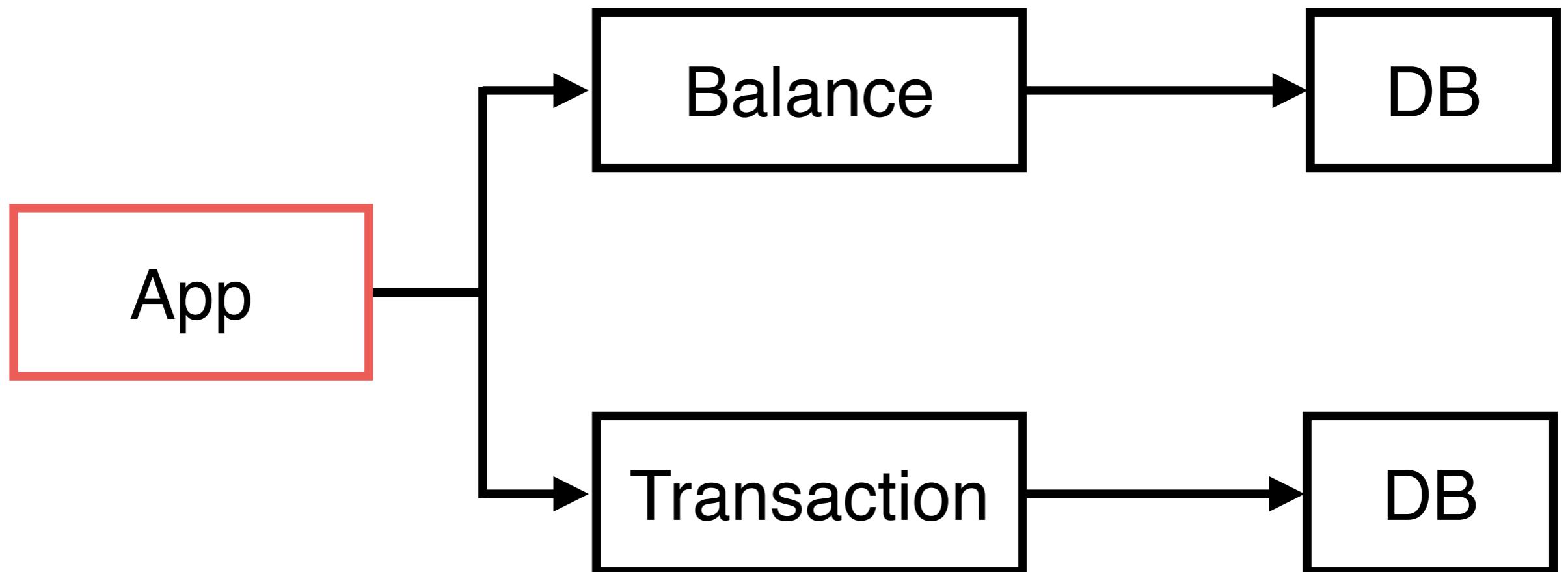
Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

Resource cleanup ?



Transaction integrity ?



Design for Failure

“Everything fails all the time”
-Werner Vogels-

<https://docs.aws.amazon.com/whitepapers/latest/running-containerized-microservices/design-for-failure.html>



Design for Failures in 12-factor

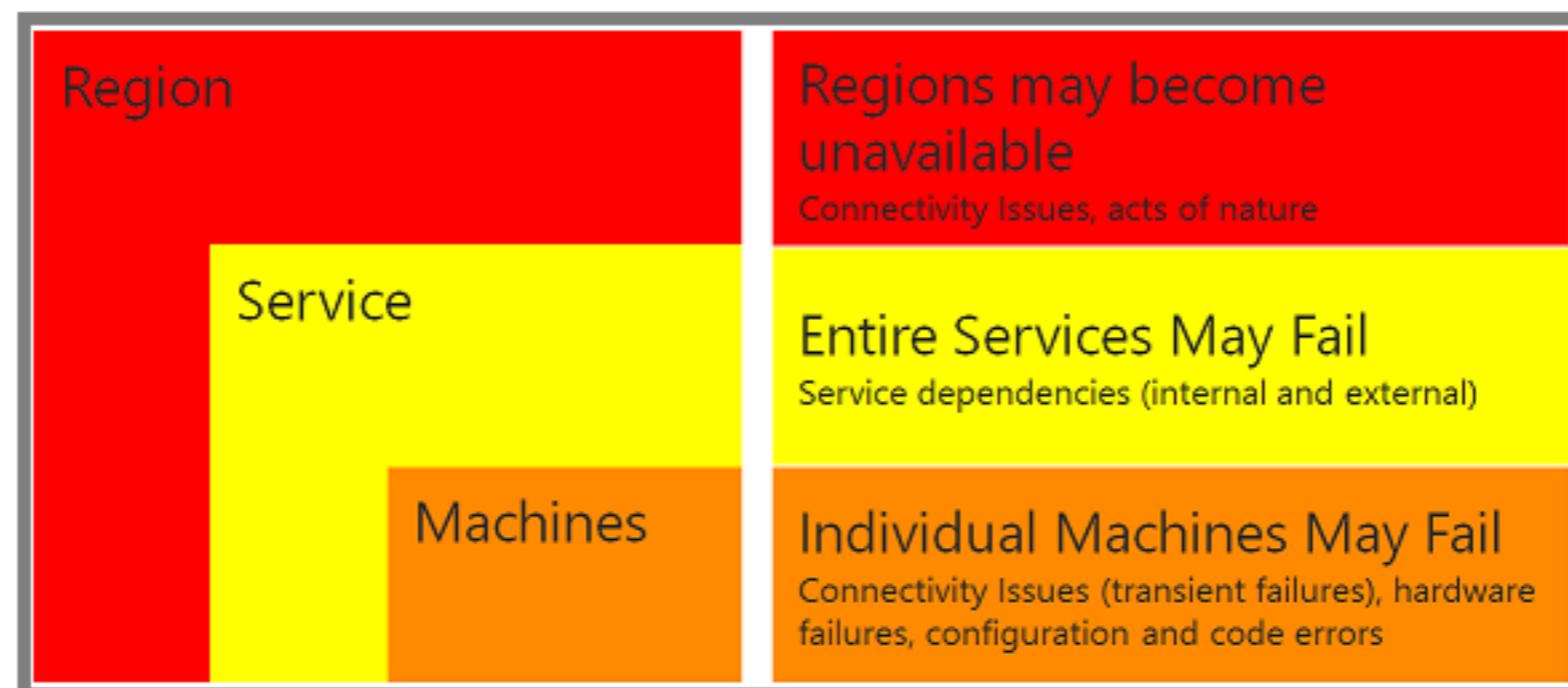
Disposability

Logs

Dev/Prod parity



Design for Failure



<https://learn.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/design-to-survive-failures>



Failures ?

Network latency

Long running sync
operation

Connection error

Server restart

Overload traffic

Rolling update

Hardware failure



Design for Failure

Defensive Design

Edge Cases

Mistake Proofing

Decoupling

Bulkhead

Redundancy

Retry

Undo

Cold Standby

Derating

Error Tolerance

Graceful Degradation

Monitoring

Fail Safe

Durability

Resilience

simplicable

https://simplicable.com/design/design-for-failure#google_vignette

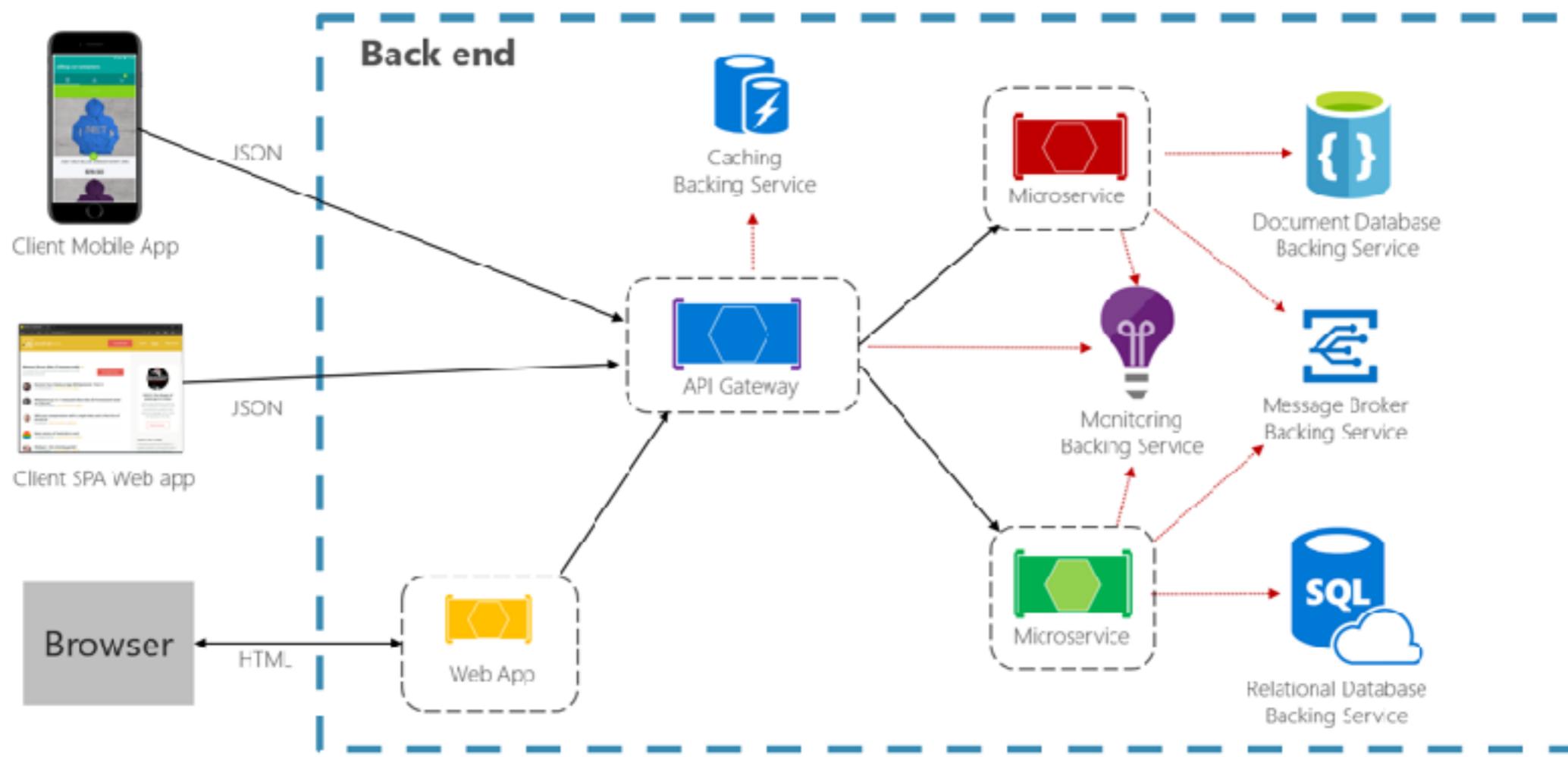


Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

Cloud Native Resilience

Ability of your system to react to failure
System still remain functional



<https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/resiliency>



How to plan/design to handle failures ?



Design for Failure patterns

Timeout
Retry
Caching
Bulkhead
Circuit breaker
Fallback
Messaging



Infrastructure design

Active-standby
Auto healing
Replication
Clustering
Multi-region

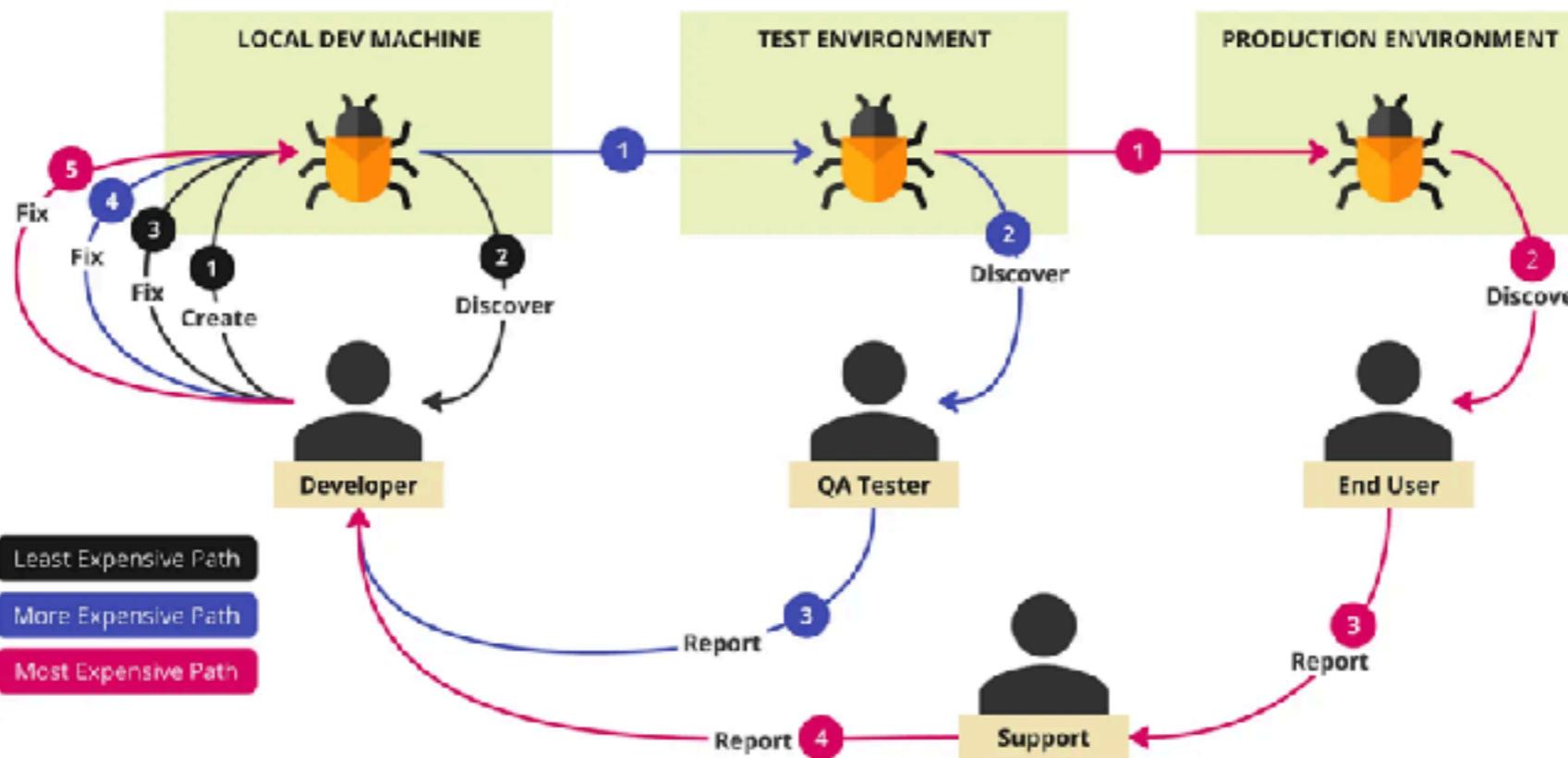


10. Dev/prod parity

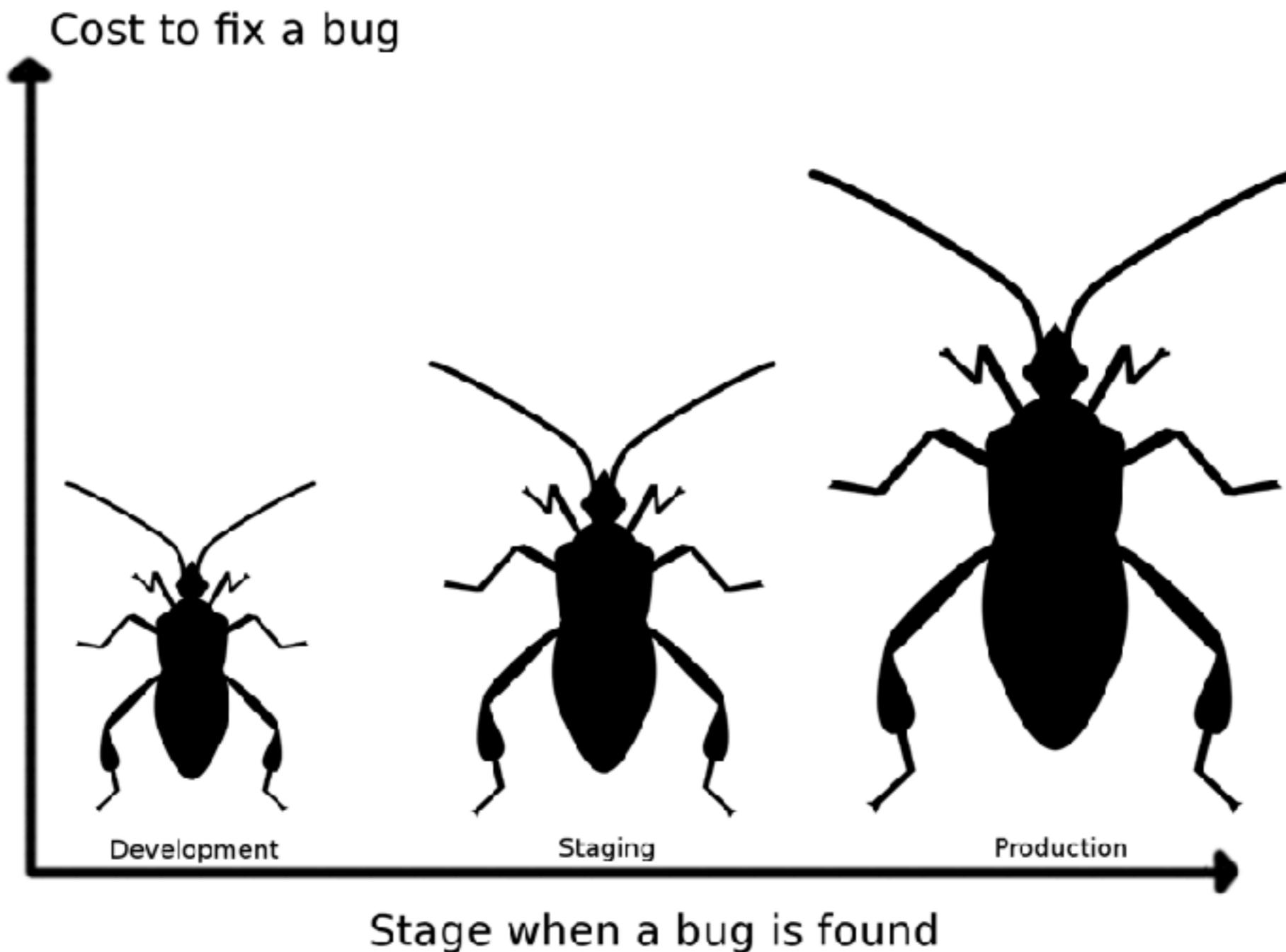


10. Dev/prod parity

Keep development, staging and production as similar as possible
Detect problems early



Cost of Bug



Gap between Dev and Prod

Time-gap
Personal-gap
Tool-gab

12-factor designed for **Continuous Deployment**
by keeping the gab between dev and prod small

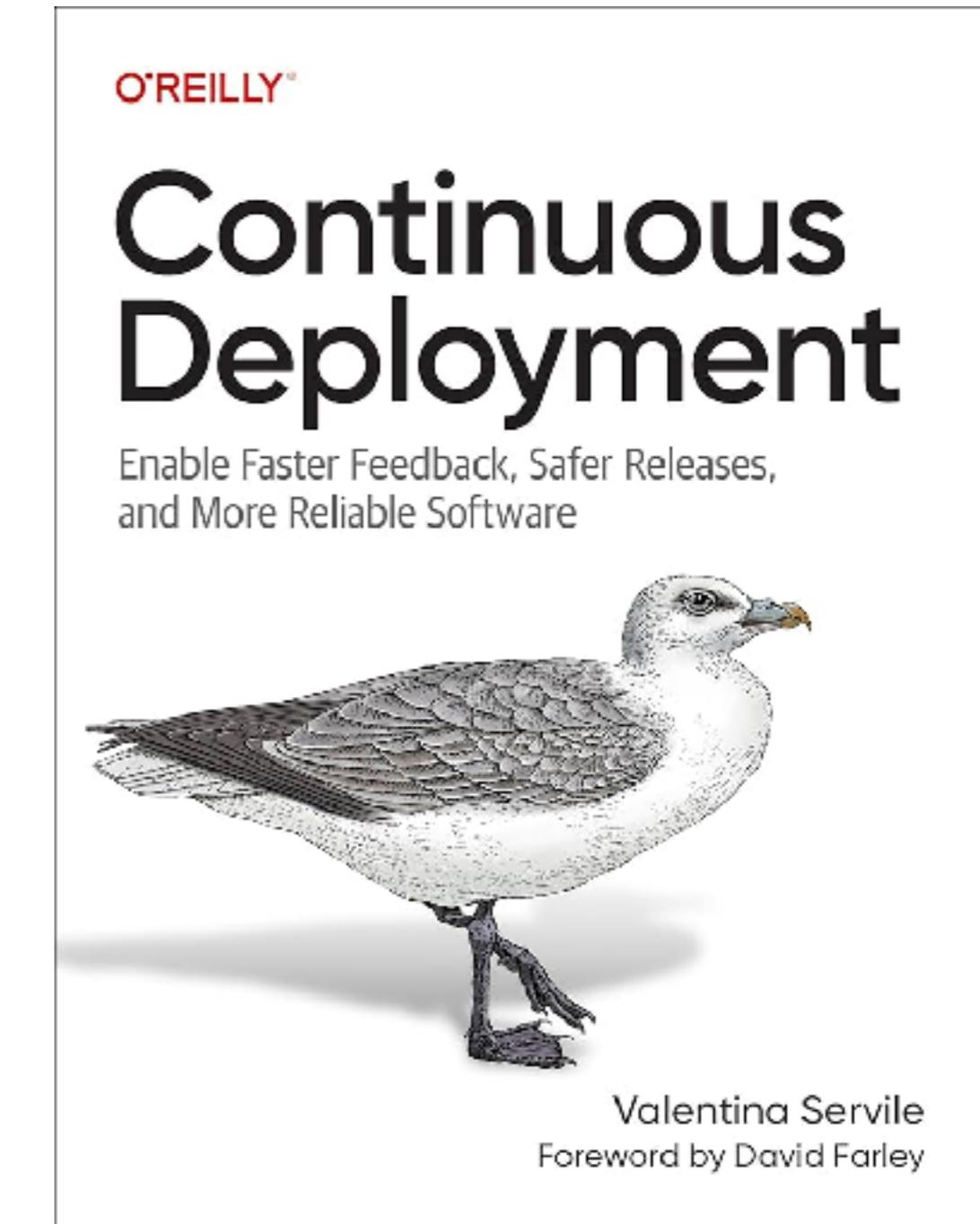
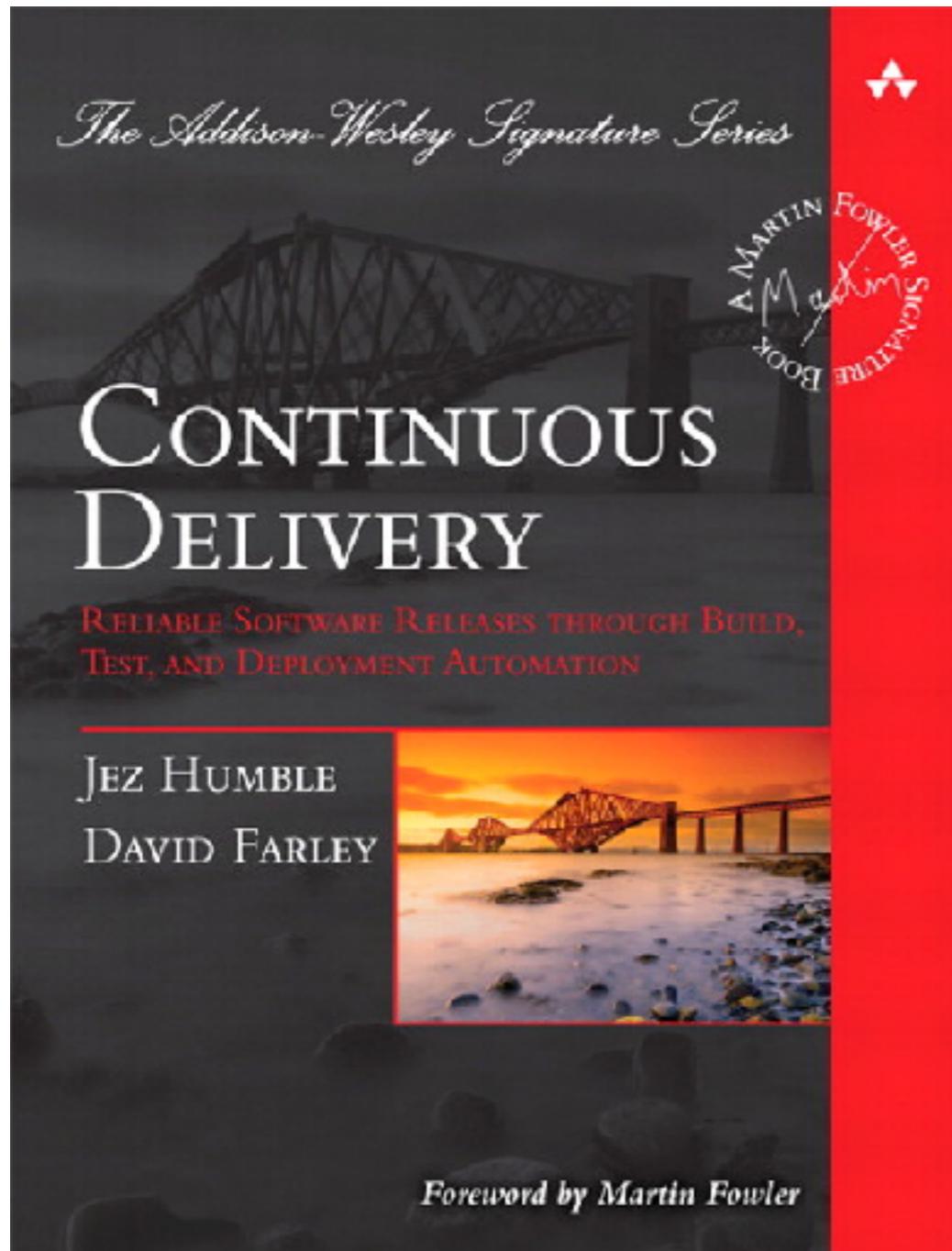


Gap between Dev and Prod

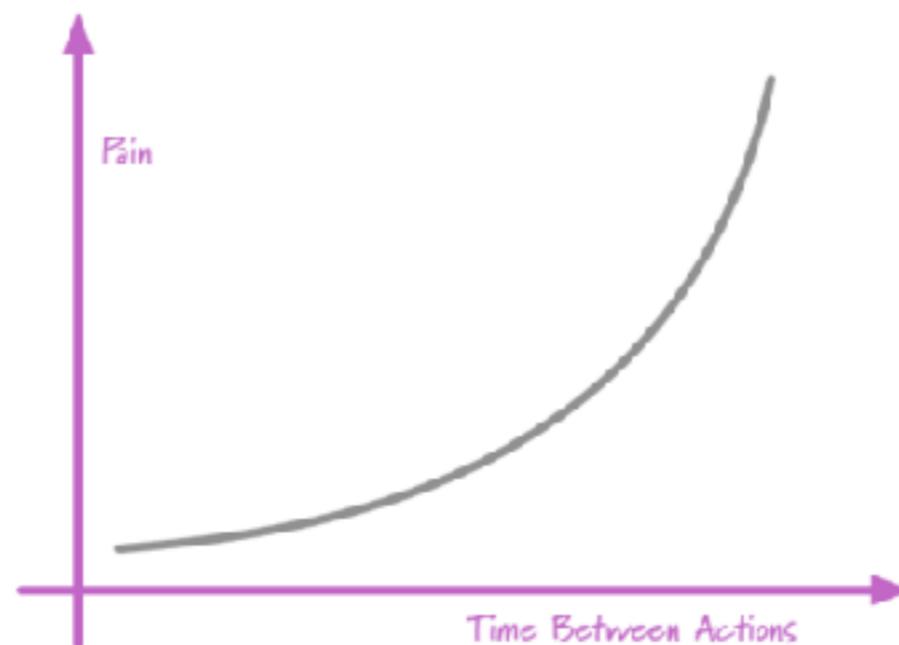
	Traditional	12-factor
Time between deploys	Weeks	Hours
Code author and code deployer	Different people	Same people
Dev vs Prod environments	Divergent	As similar as possible



Continuous Deployment



“If it hard, do it more often”



<https://martinfowler.com/bliki/FrequencyReducesDifficulty.html>



Continuous Deployment

Fast feedback

Reduce risk

Safety release

Increase reliable of software



Tools

Docker

Chef/puppet

Kubernetes

Terraform

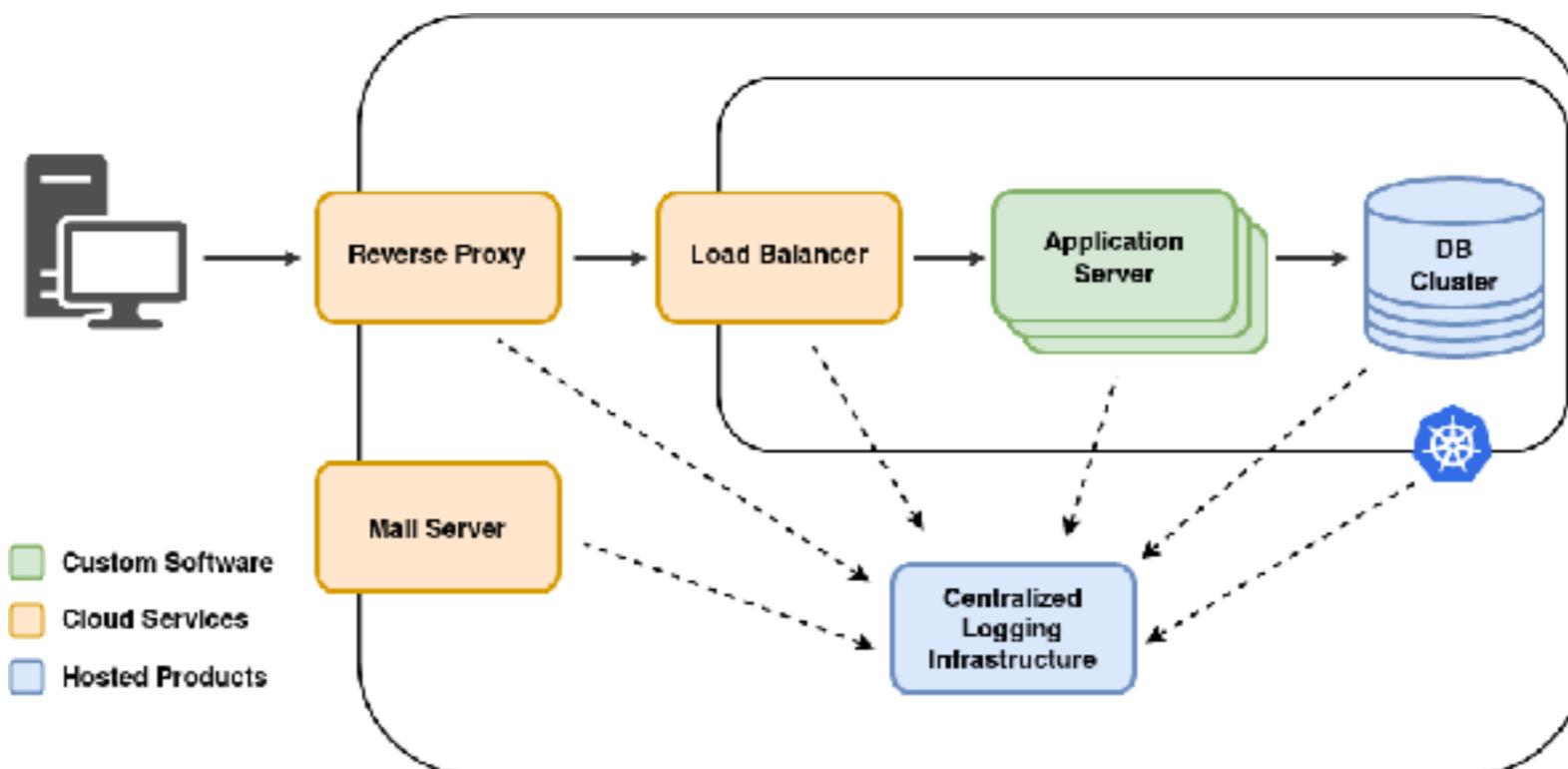


11. Logs



11. Logs

Treat log as event streams
Logs provide visibility into the **behavior** of a running app



Structured logging !!



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.ArithmetricException: 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
```



Structured logging

Log Formats Explained

Structured

- Usually JSON or LogFmt.
- Least human-readable.
- Easy to parse, search and filter.
- Consistent formatting.
- Faster troubleshooting due to added context.
- Automated reporting and analysis.



RECOMMENDED

Semi-structured

- Mix of structured and unstructured data.
- Human-readable.
- Formatting variability leads to inconsistency.
- Limited automation possibilities.
- Requires a more complex parsing logic.

Unstructured

- Free-form plain text messages.
- Human-readable.
- Challenging to parse.
- Requires manual interpretation.
- Suitable only for quick ad-hoc logging in development environments.



NOT RECOMMENDED



<https://betterstack.com/community>

<https://betterstack.com/community/guides/logging/structured-logging/>



Workshop

© 2017 - 2024 Siam Chamnankit Company Limited. All rights reserved.

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



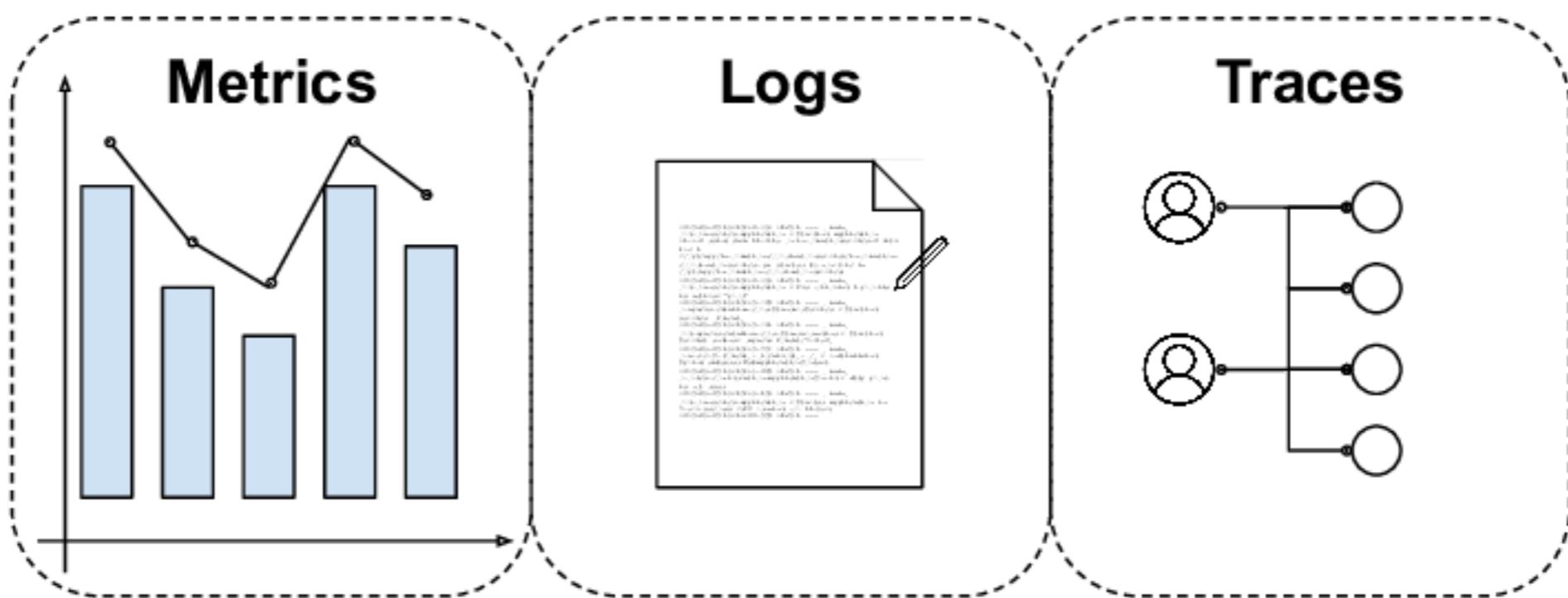
Consistent Structure across all logs

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

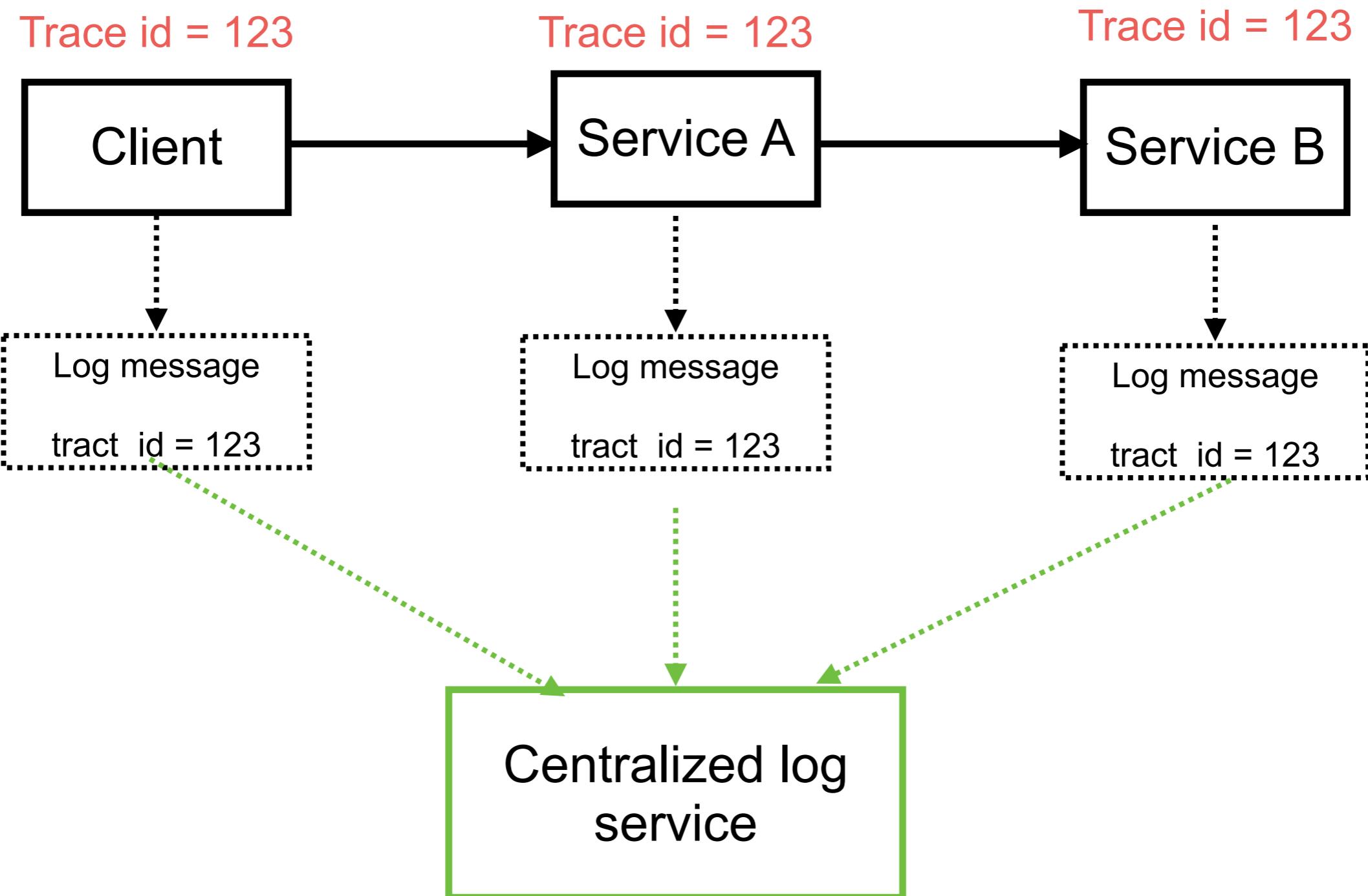


Not Only Log !!

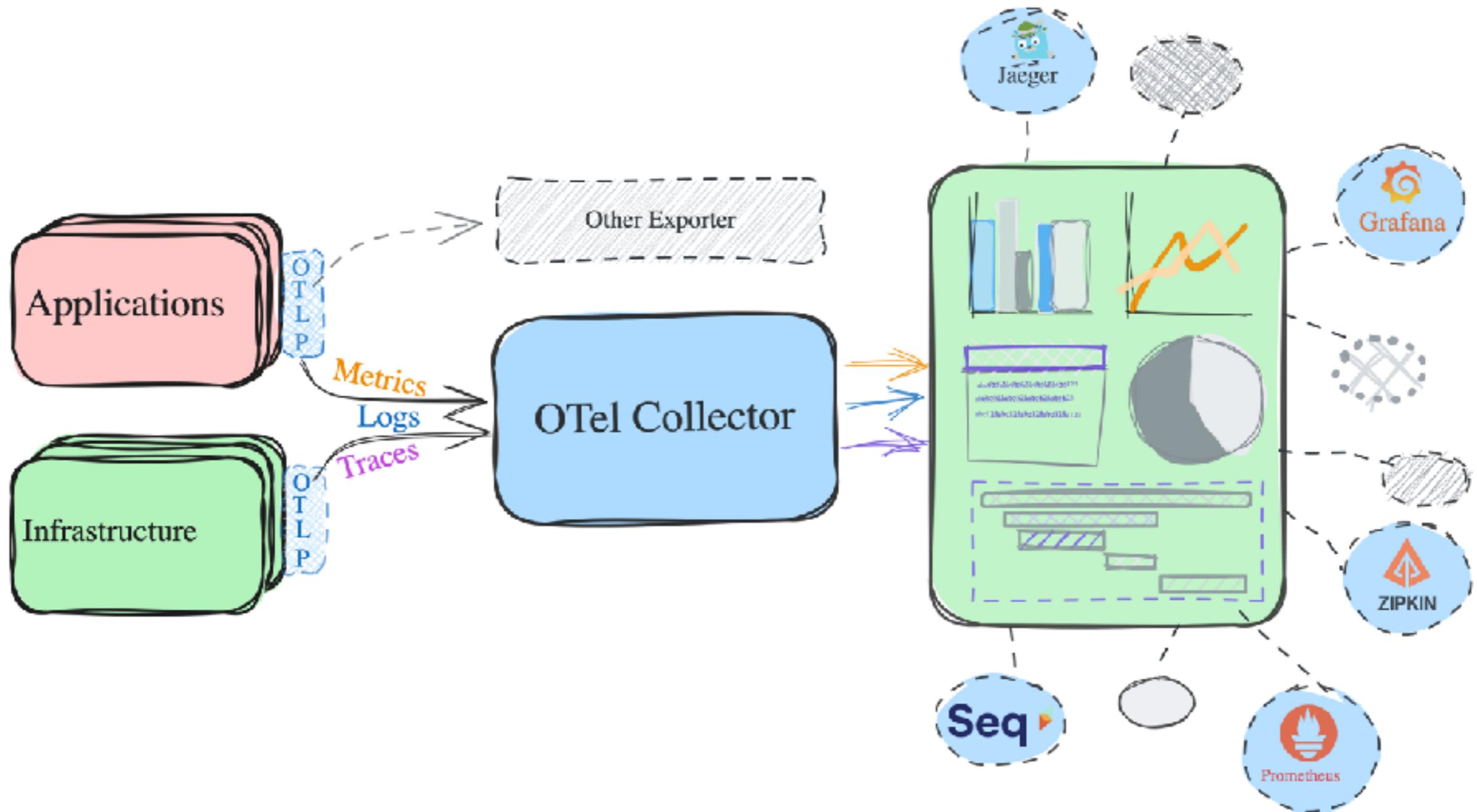




Centralized log with trace



OpenTelemetry



12. Admin processes



12. Admin processes

Run admin/management tasks as one-off processes

Reduce ad-hoc tasks
Separated from app

Data migration

Data cleanup

Computing analytics



12 + 3 Factor

O'REILLY®

Beyond the Twelve-Factor App

**Exploring the DNA of Highly Scalable,
Resilient Cloud Applications**

<https://raw.githubusercontent.com/ffisk/books/master/beyond-the-twelve-factor-app.pdf>



13. API First



13. API First

Define the service contract first

Help consumer understand request/response

How to communicate with service ?

Help to reduce bottlenecks



OpenAPI



SwaggerTM



Code First vs API First (design)

Code First



Business input

Code

API Documentation

Design First



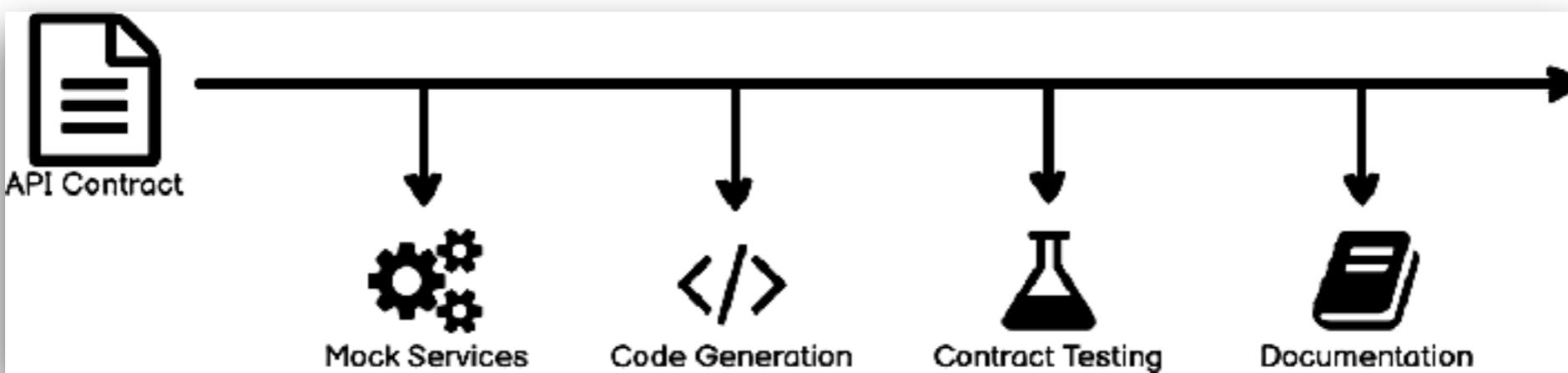
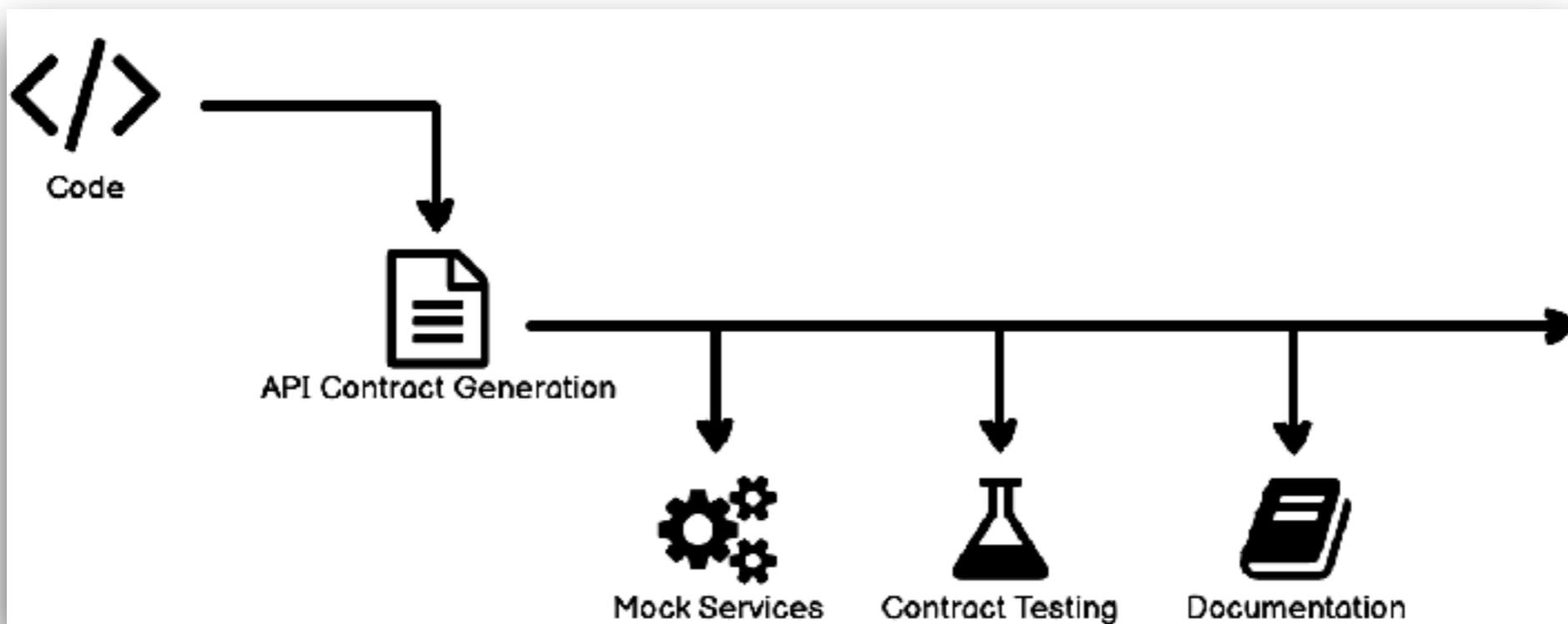
Business input

API Design

Code



Code First vs API First (design)



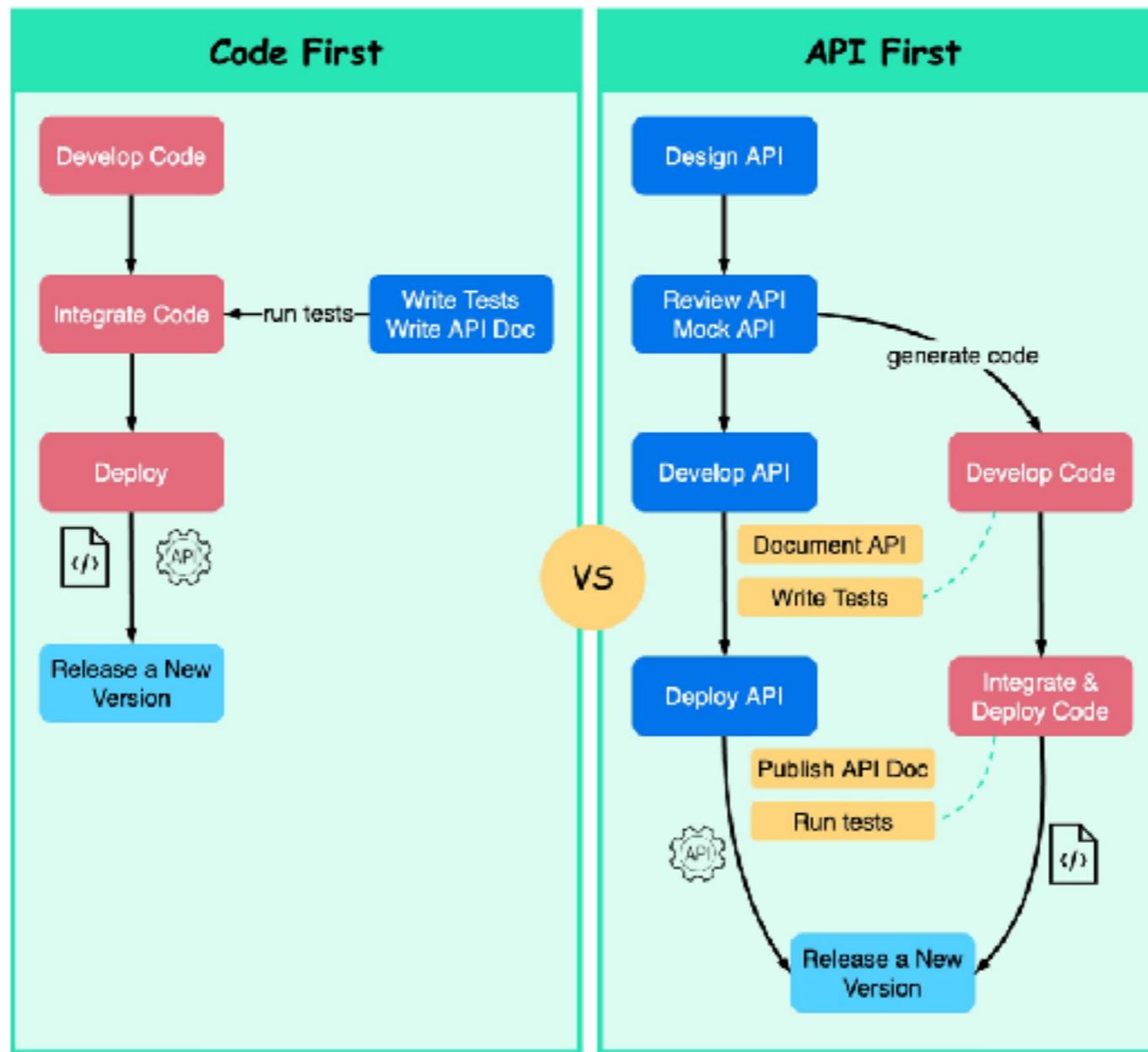
<https://sookocheff.com/post/api/the-false-dichotomy-of-design-first-and-code-first-api-development/>



Code First vs API First (design)

Code First v.s API First Development

 blog.bytebytego.com



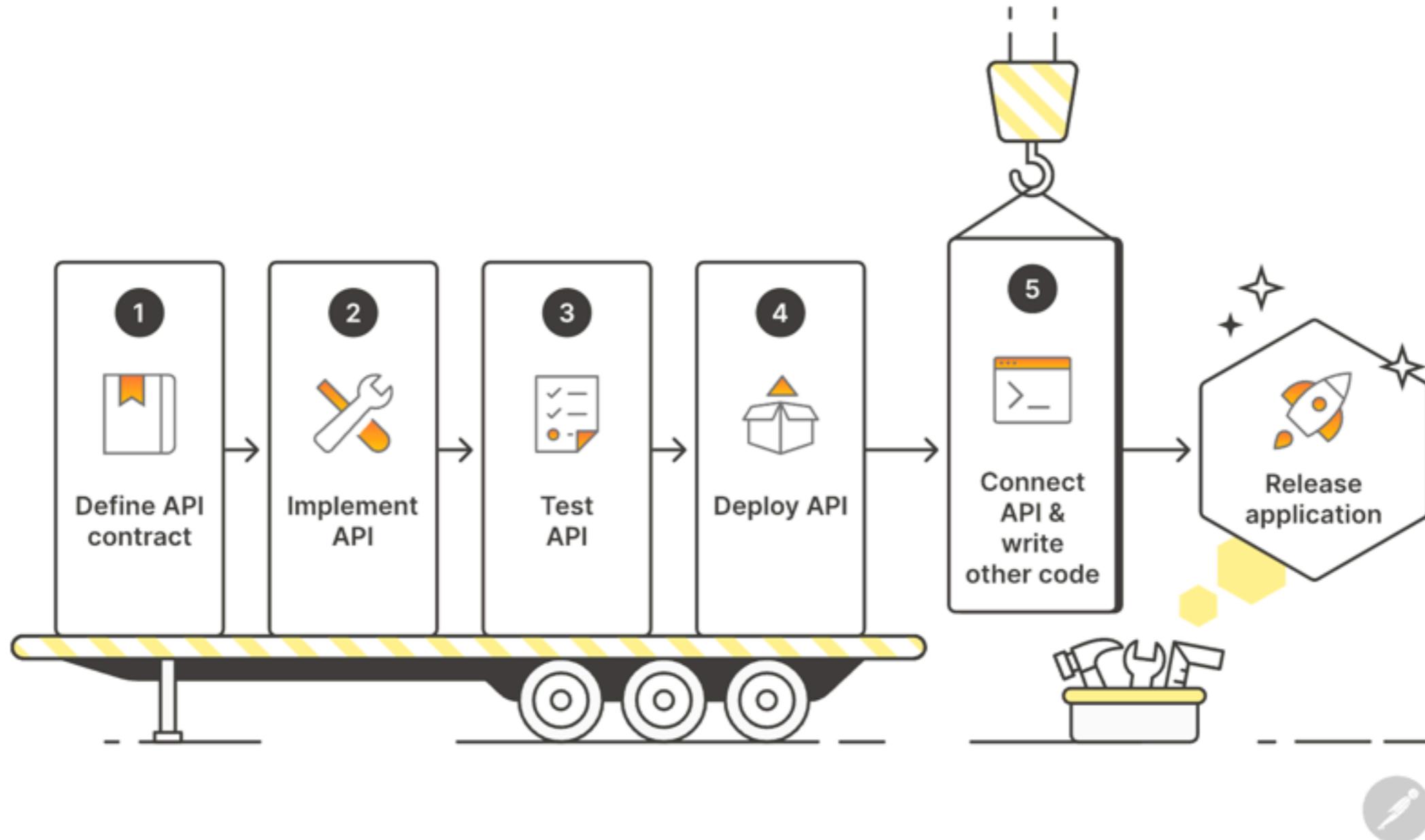
<https://twitter.com/bytebytego/status/1665965464512790528>



Workshop

© 2017 - 2024 Siam Chamnkit Company Limited. All rights reserved.

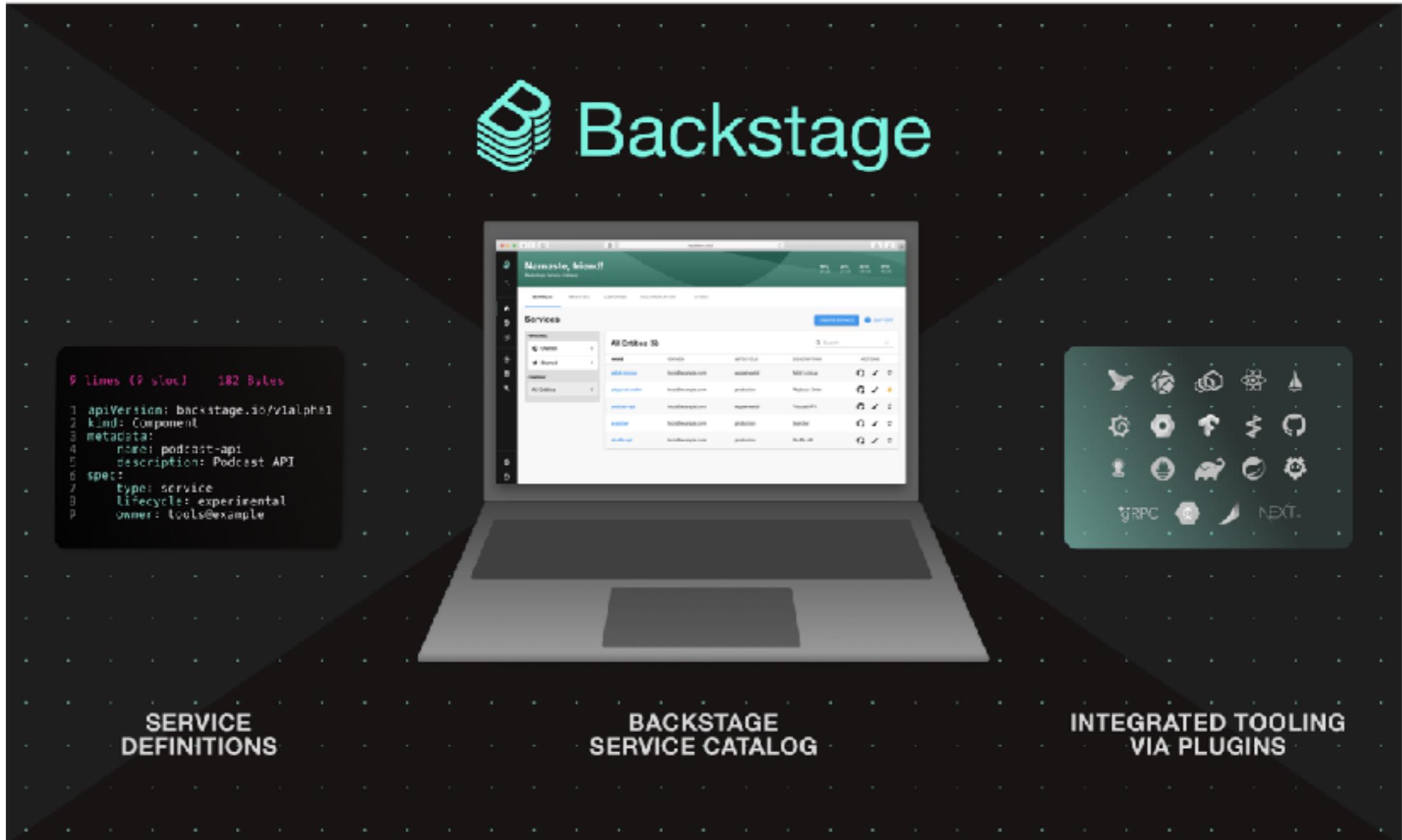
API First from Postman



<https://www.postman.com/api-first/>



Backstage Software Catalog



<https://backstage.io/>



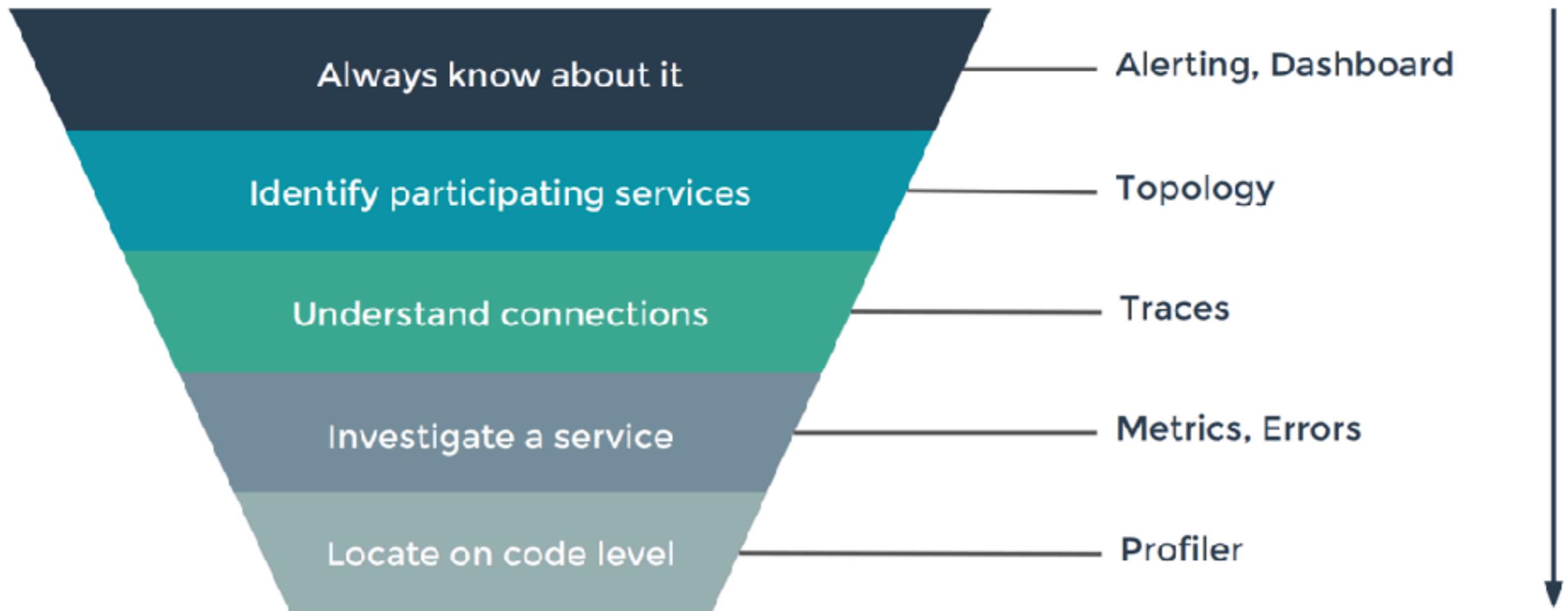
Workshop

© 2017 - 2024 Siam Chamnkit Company Limited. All rights reserved.

14. Telemetry



How to find an issue ?



Category of monitoring

APM (Application Performance Monitoring)
Domain-specific telemetry
Health and system log



Telemetry :: specific-domain

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



Domain specific domain

Stream of events and data
Data for business goals

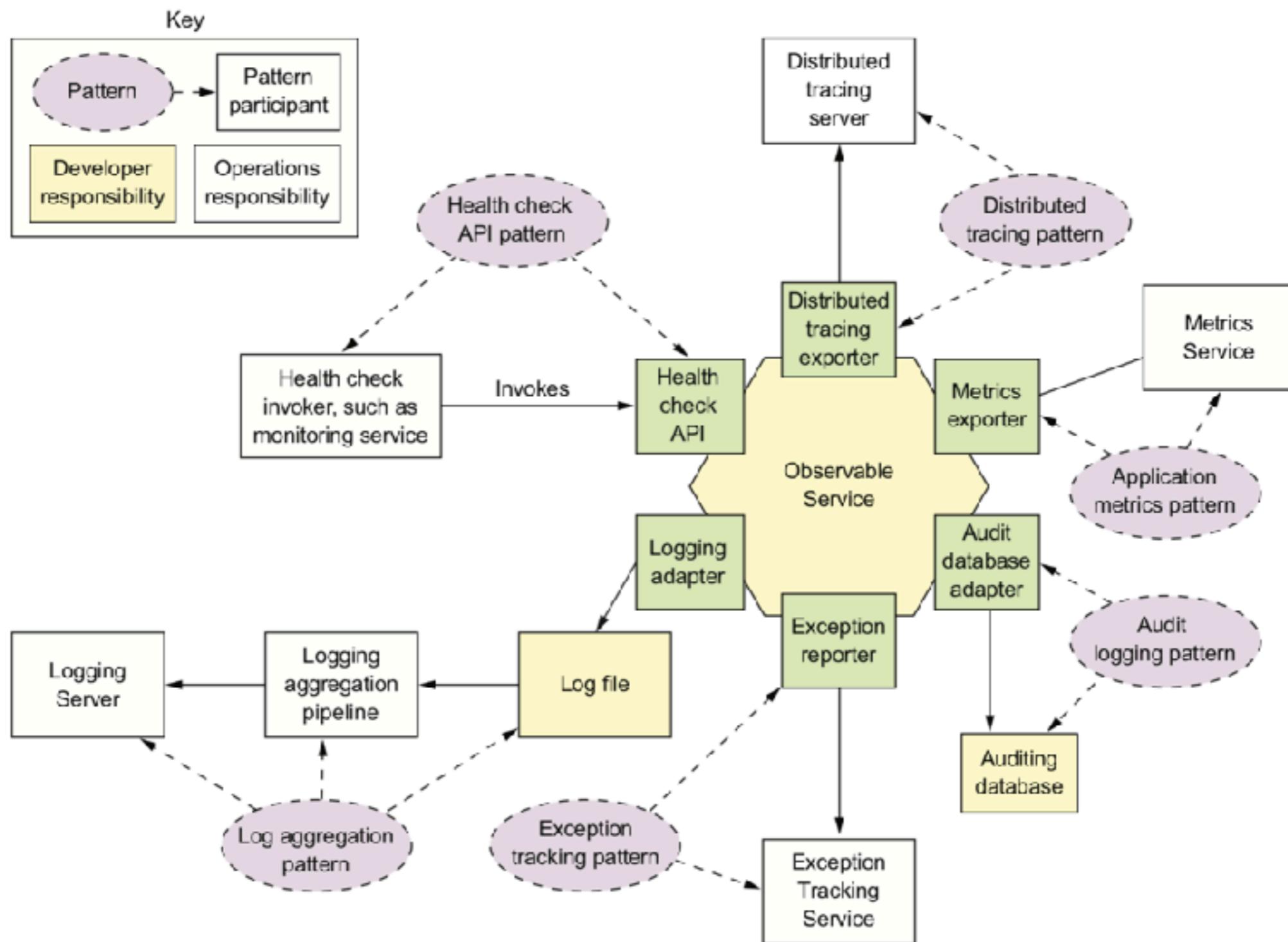
Store

Analysis

Forecast



Observable services



15. Authentication and Authorization



Security by design



Develop secure services

Authentication

Authorization

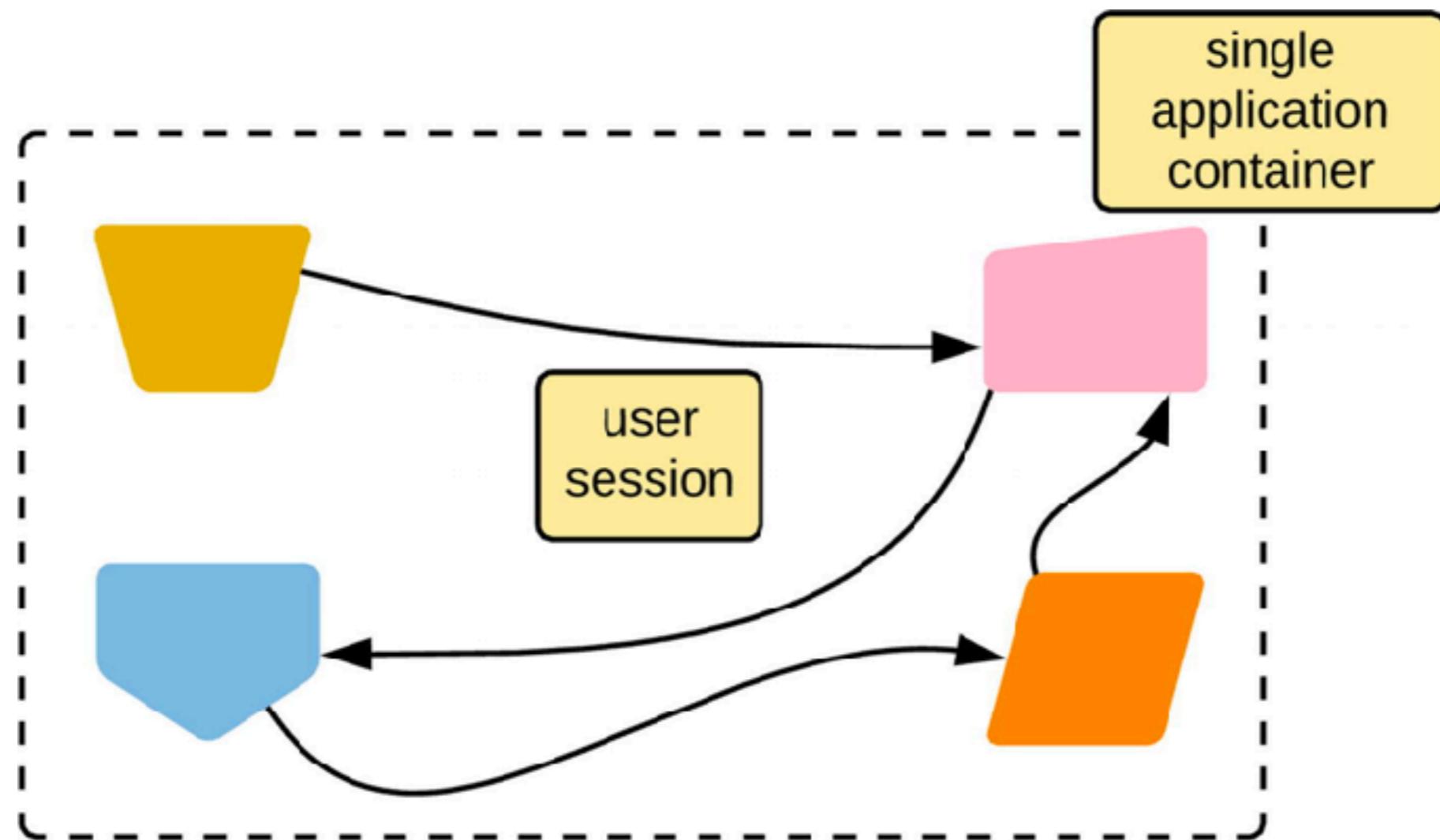
Auditing

Secure interprocess communication



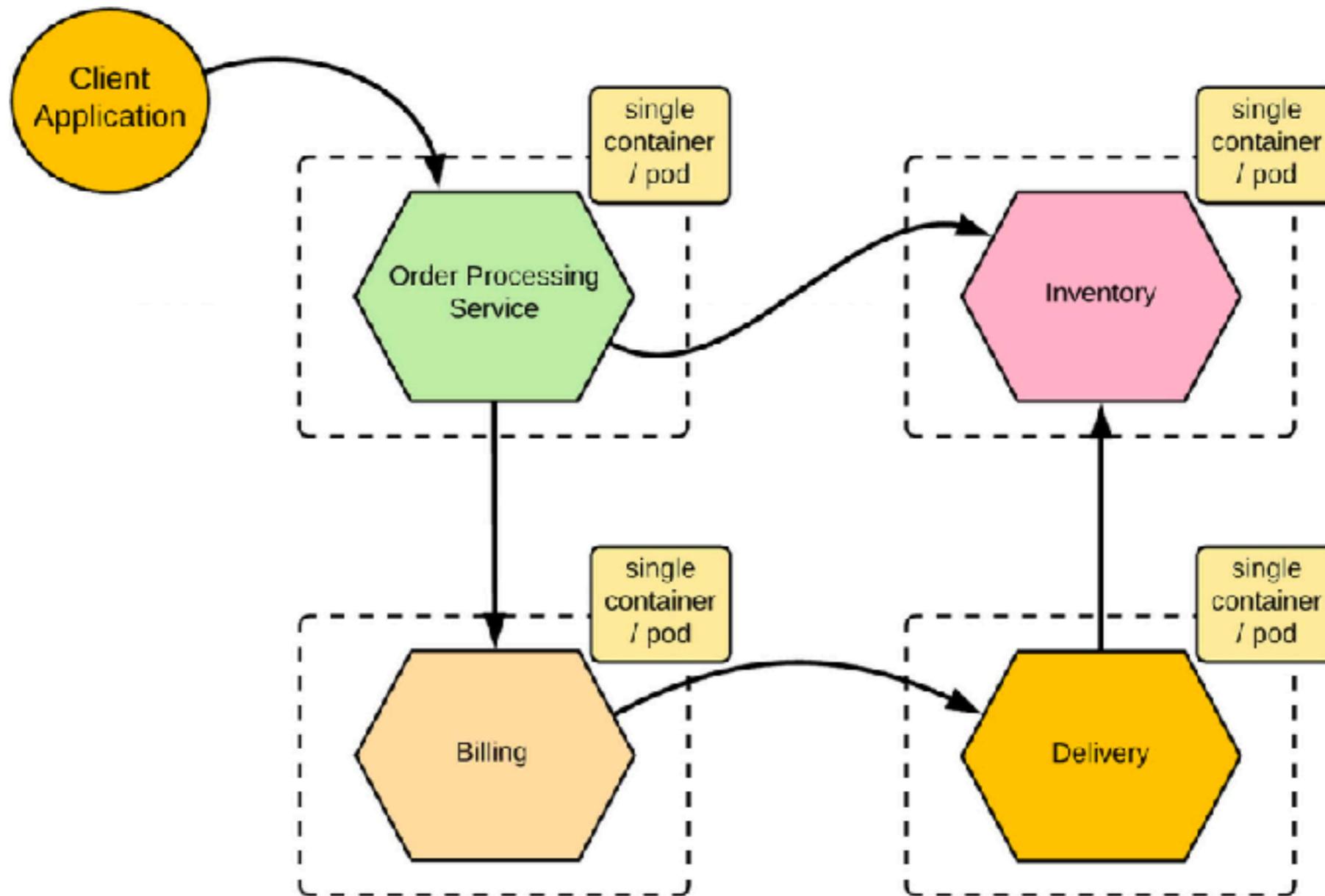
Security in traditional application

Sharing a user session in application



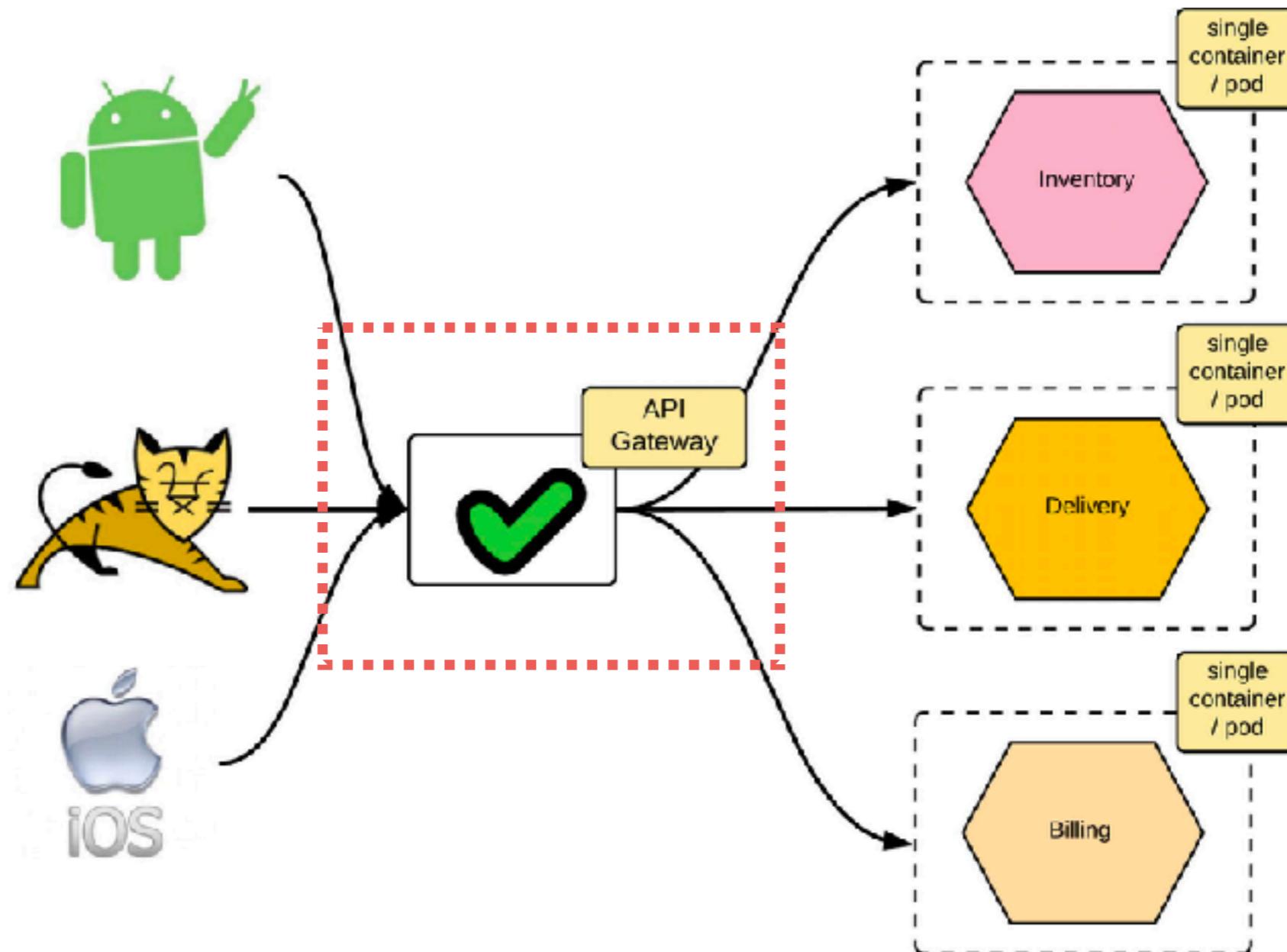
Security in multiple services ?

Interaction between multiple services



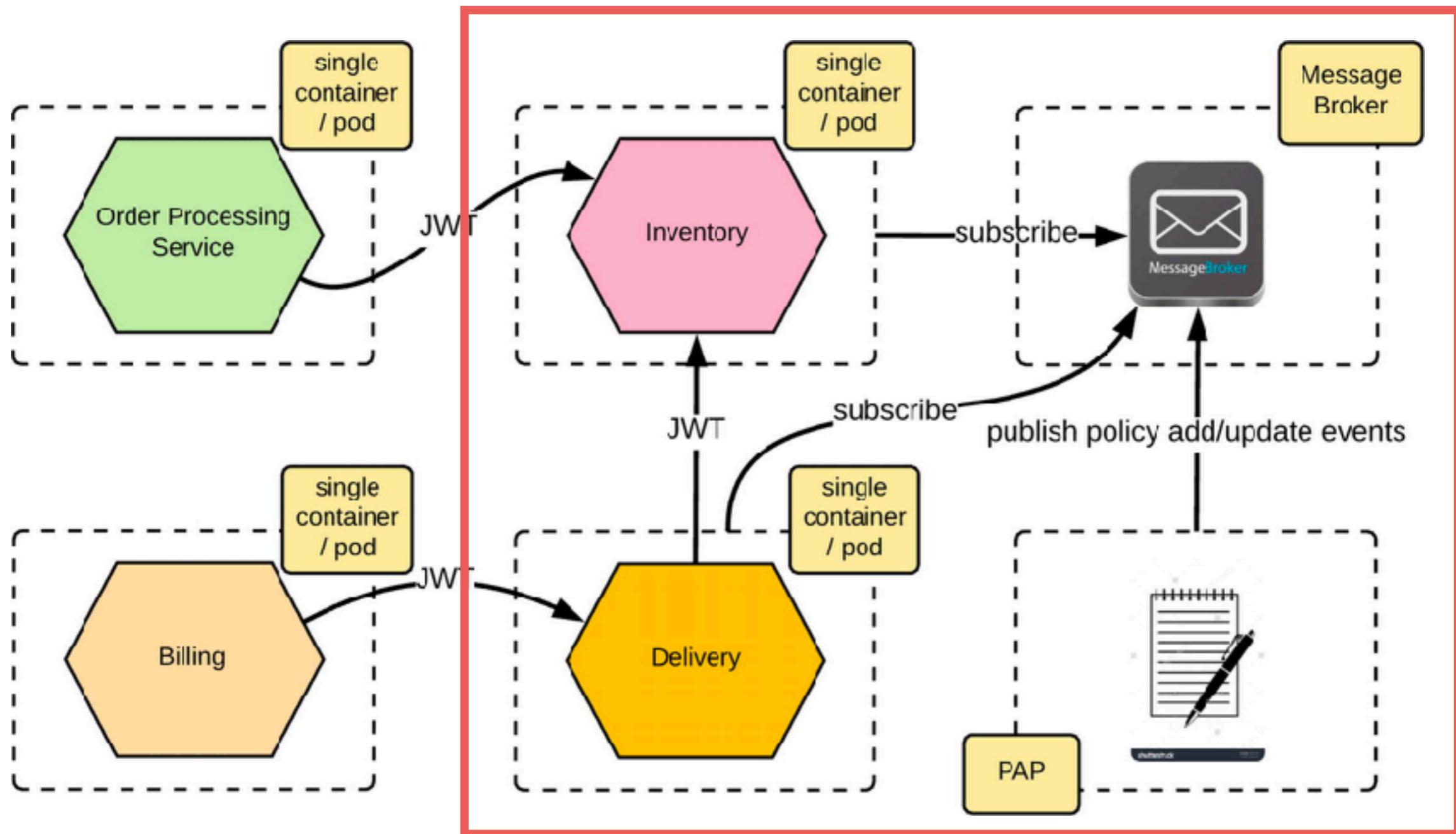
Centralize pattern

Using API gateway pattern



Access control of services

Policy Administration Point



Workshop

C#

Go

Docker



Q/A

