



# Not my problem!

Delegating responsibilities to  
infrastructure

Yshay Yaacobi  
@yshayy  
<https://git.io/fxh57>



As a developer, I want to **focus on building features** that deliver **business value**

# Infrastructure



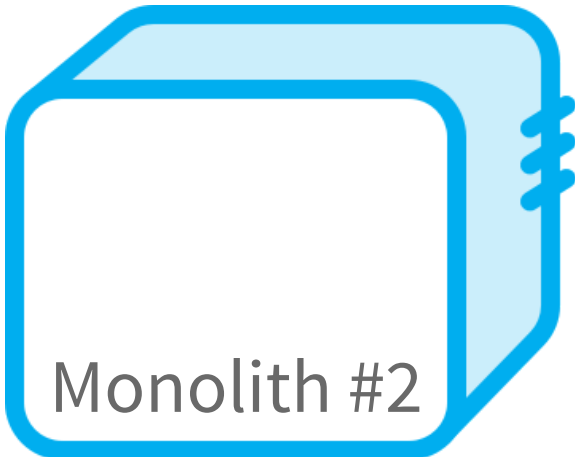
Authentication  
Secrets  
Authorization  
Monitoring  
Routing Caching  
Configuration  
Service Discovery  
Tracing  
SSL  
Logging  
Resiliency



Monolith -----> Microservices

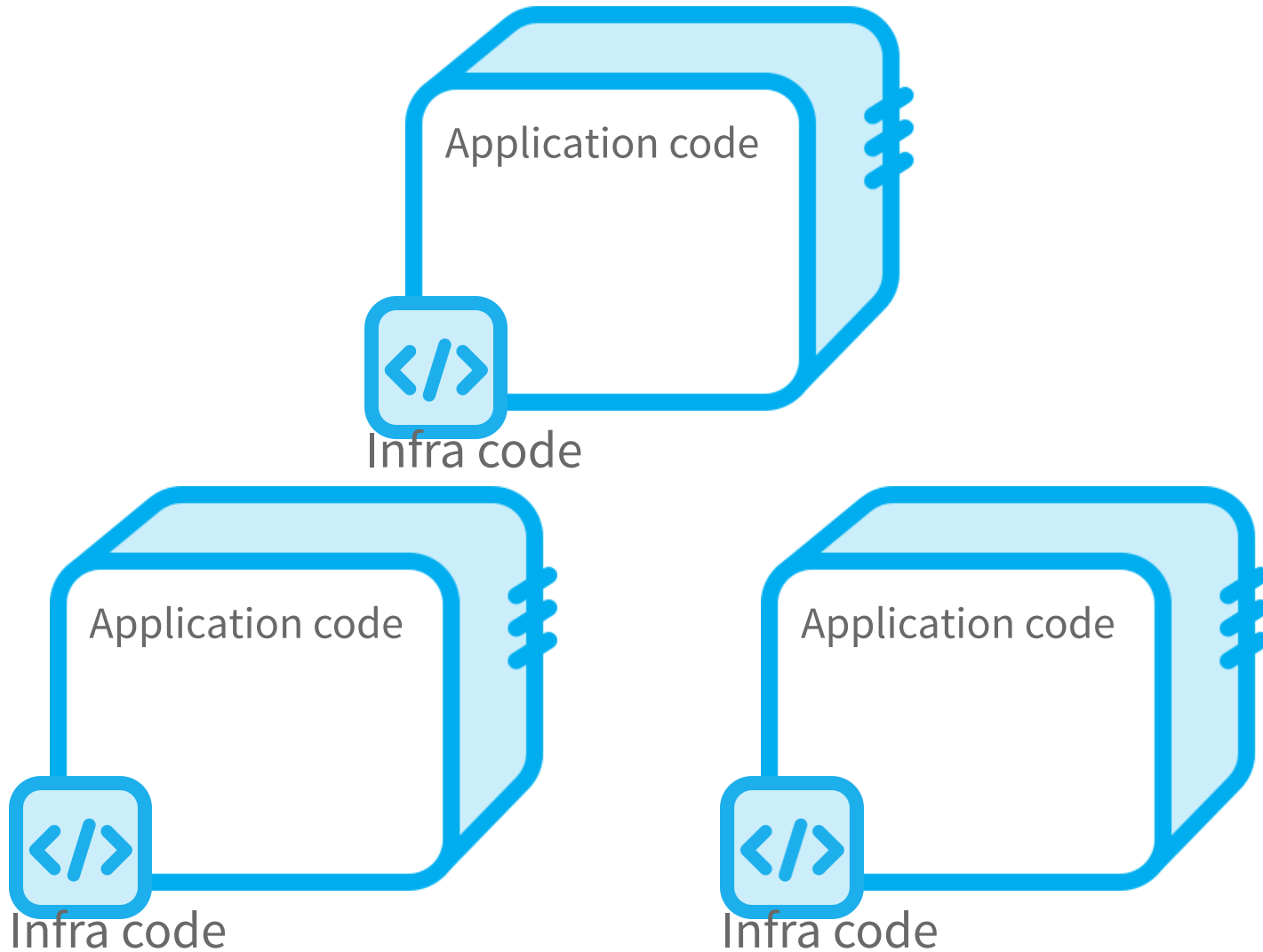


# Few Monoliths



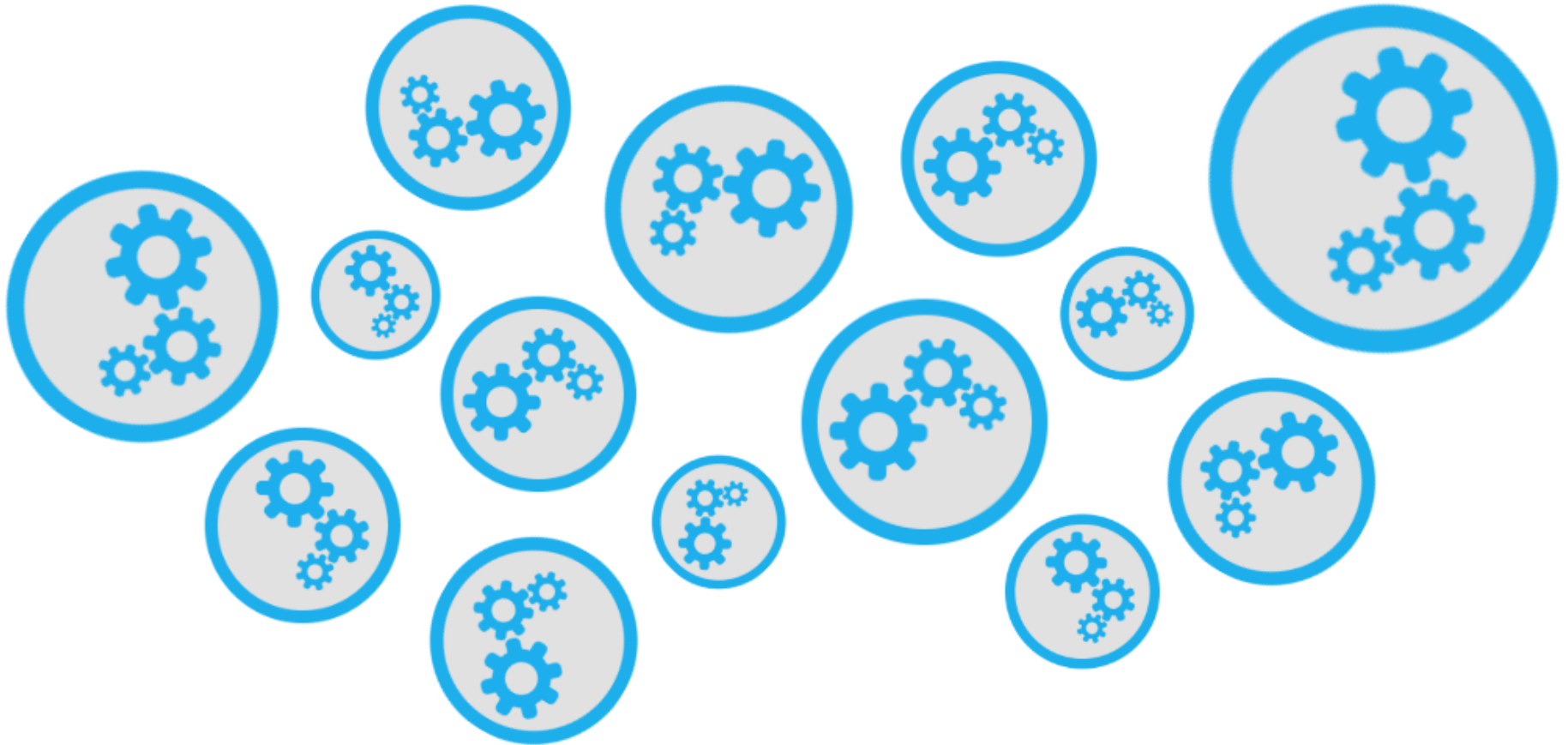


# Few Monoliths



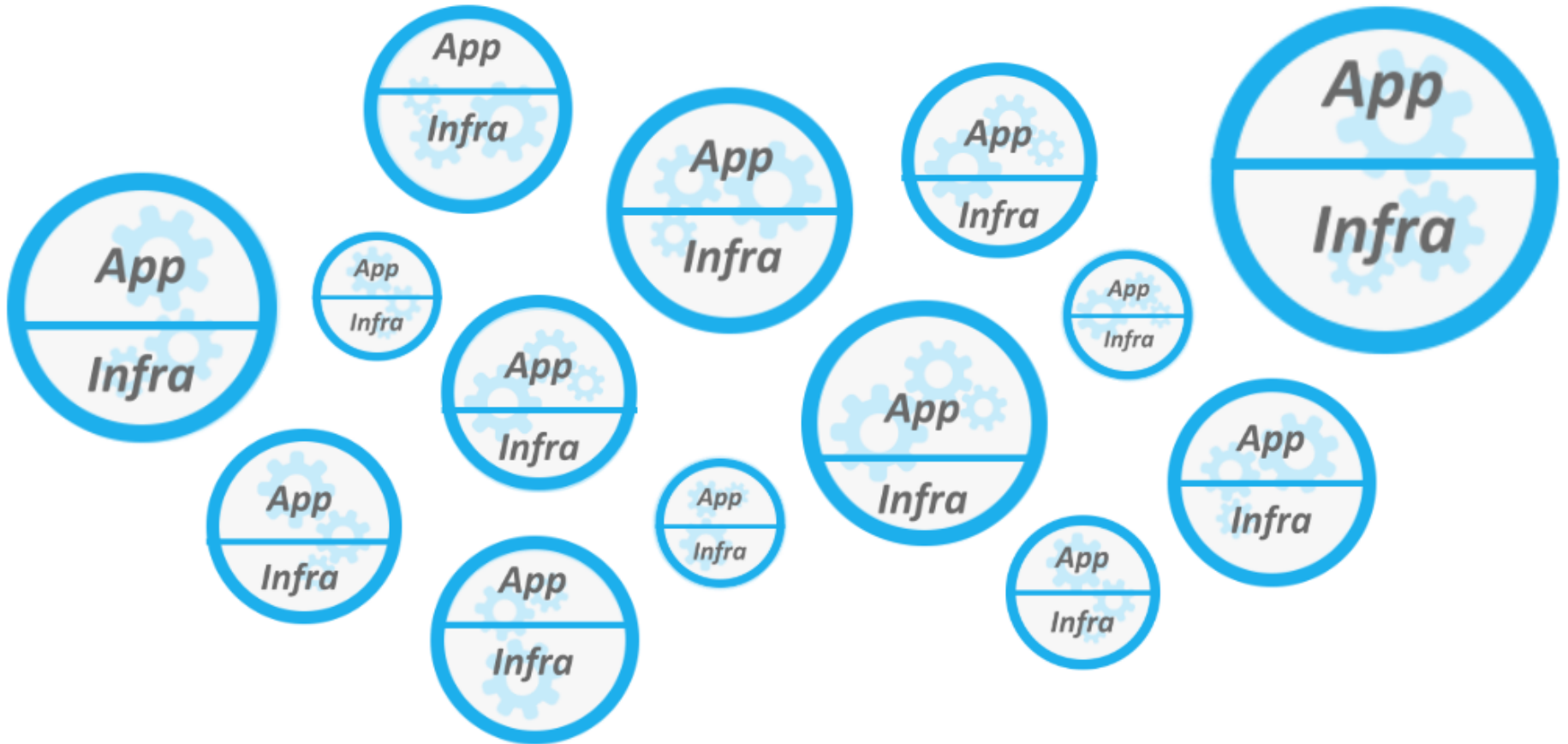


Monoliths ----> Lots of Microservices





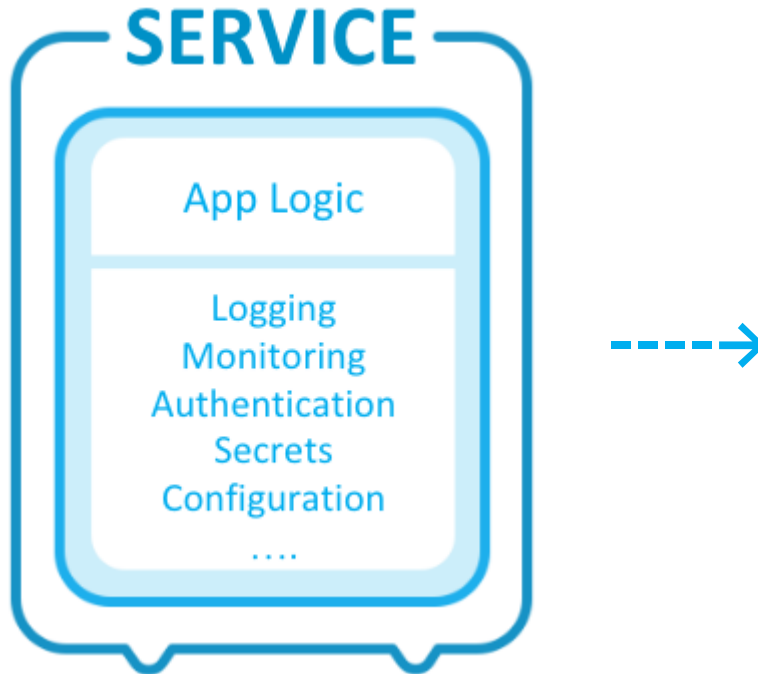
# Monoliths ----> Lots of Microservices







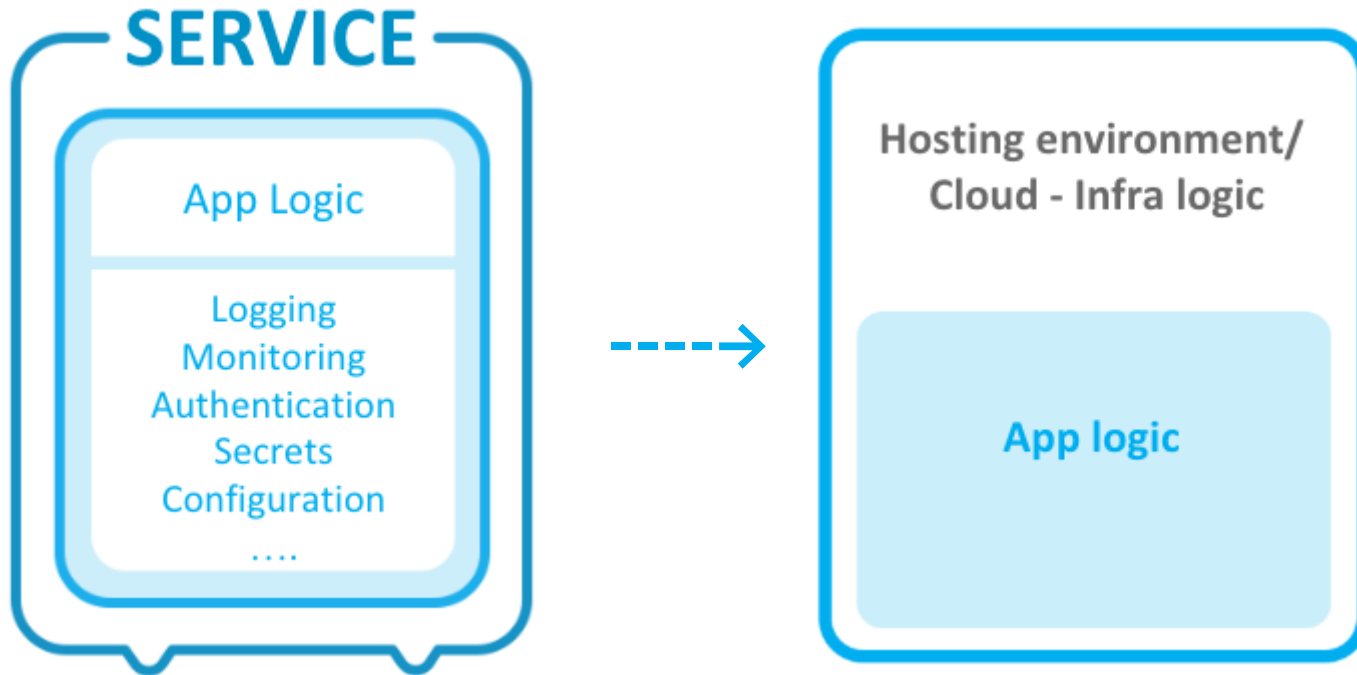
# Service structure



Infra-code and business logic code live together with every Microservice



# Service structure



How can we write only our business code and let our hosting environment take care of the rest?



# About me

- Tech lead @ Solutio
- Open source
- Cloud architecture
- Functional Programming
- Docker
- Code Quality



Yshay Yaacobi  
@yshayy



# About Solutio

- Based in Tel Aviv, acquired by Asurion at 2013
- Next generation of tech support
- 150M users worldwide
- We love open-source



*since*  
**{ 2014 }**





# Shifting to microservices

- ~5 services -> 100+ services
- Cultural change, new aspects of ownership
- CI/CD - Better tooling automation
- Self-service



# How does a service look like?





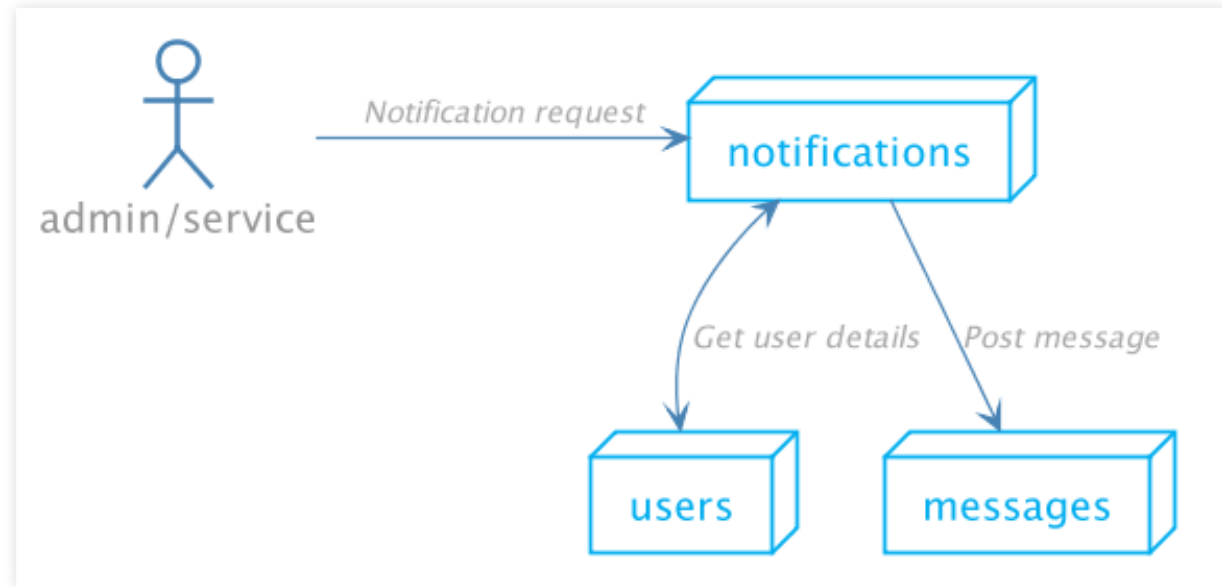
# DEMO - Notifications service V1



# Spec - Notifications service

- We want to send notifications to users
- Notifications will be based in templates
- Notifications will be personalized

# Flow - Notification service



1. Get user identifier and notification type
2. Fetch user details from users micro-service
3. Format message using a template
4. Send a message



# Let's write some code



# Demo



# What's missing?

- Visibility
- Security
- Resiliency



# Logging

- We need to add a logger
- We send logs to a 3rd party provider
- Let's get a library + a provider



- ☒ Logging
- ☐ **Error policies**
- ☐ Authentication
- ☐ Monitoring





# Error policies

- The network can fail
- We want to retry failed requests
- Let's add Polly



- ☒ Logging
- ☒ Error policies
- ☐ **Authentication**
- ☐ Monitoring



# Authentication

- We'll use JWT token
- OIDC/OAuth2



- ☒ Logging
- ☒ Error policies
- ☒ Authentication
- ☐ **Monitoring**



# Monitoring

- Performance metrics, throughput, latency
- Statsd client



- ✓ Logging
- ✓ Error policies
- ✓ Authentication
- ✓ Monitoring



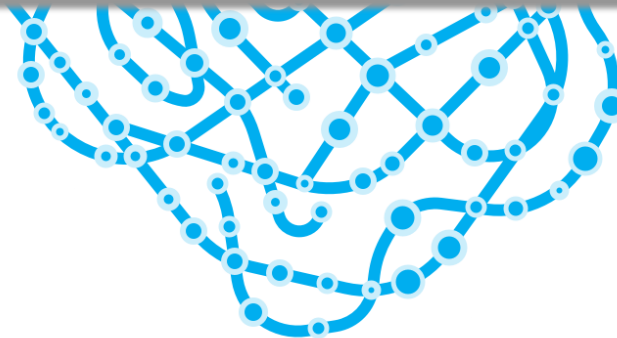
# Demo



# What just happened here?



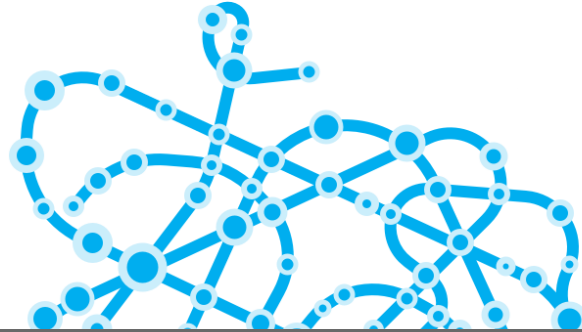
**We wanted a small micro service and got a bunch of code and dependencies**



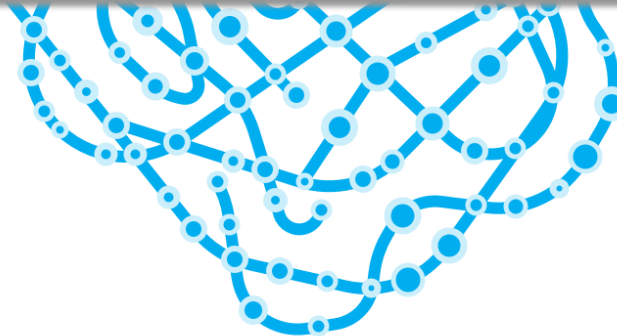




# What just happend here?



**4 X Lines of codes !!!**  
**3 X Direct dependencies !!!**





## Soluto - microservices v1

- Shared template - dependencies and code blocks
- Used by teams as a boilerplate
- Shared “common” packages/frameworks
- Lots of DI magic
- Worked well until...



# Problems

- Upgrades are **Hard**
- Dependencies can break
- Dependencies can conflict
- Dependencies requires re-build+re-deploy
- Extremely difficult to introduce global change

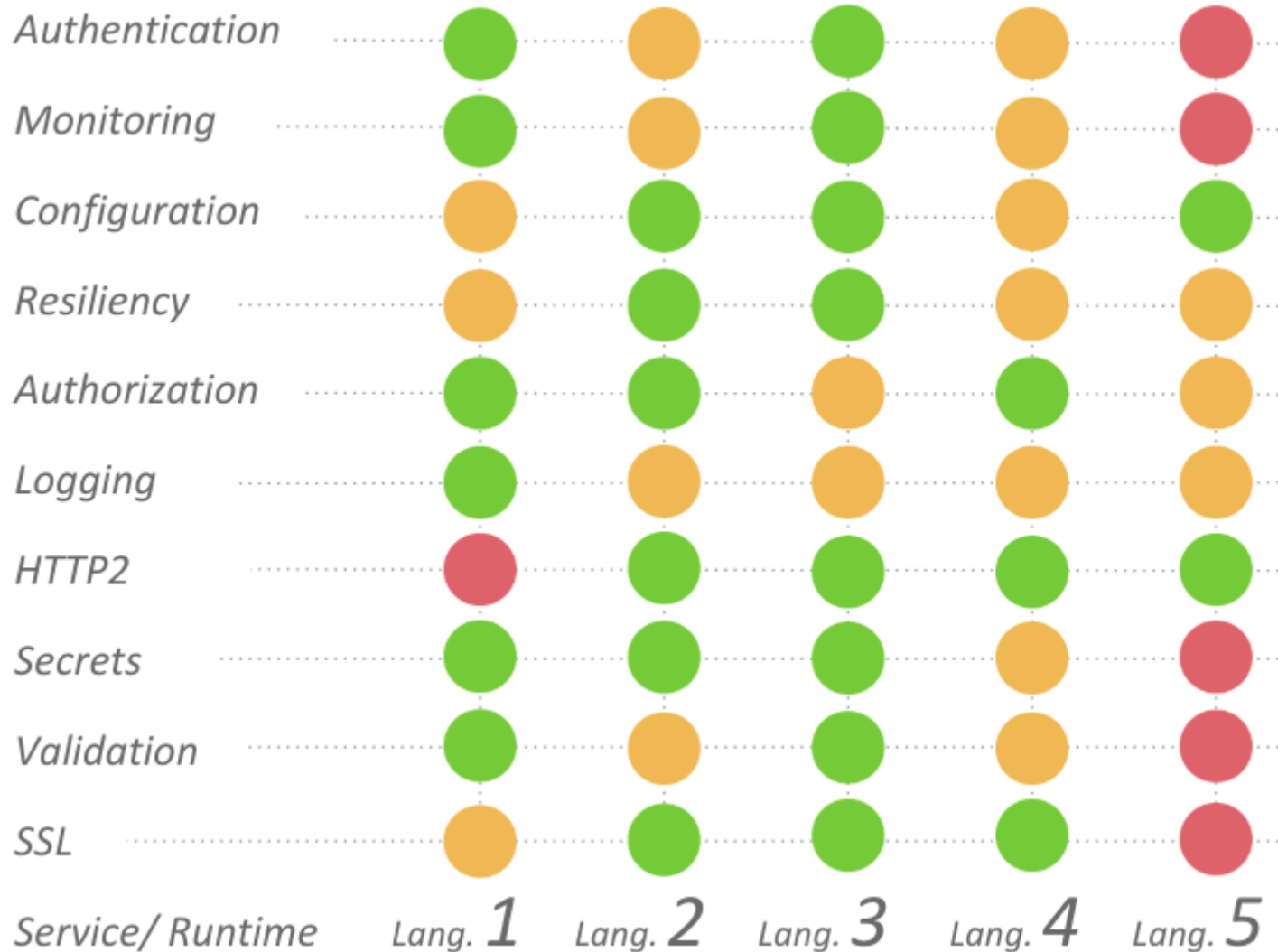


**And that's not the worse...**



**We need to solve it for each and every  
programming language**

# Not all languages are born equal







As a developer, I want to **focus on building features** that deliver **business value**

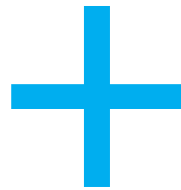




Which of these concerns can be solved at **environment level** ?



# Soluto - microservices v2





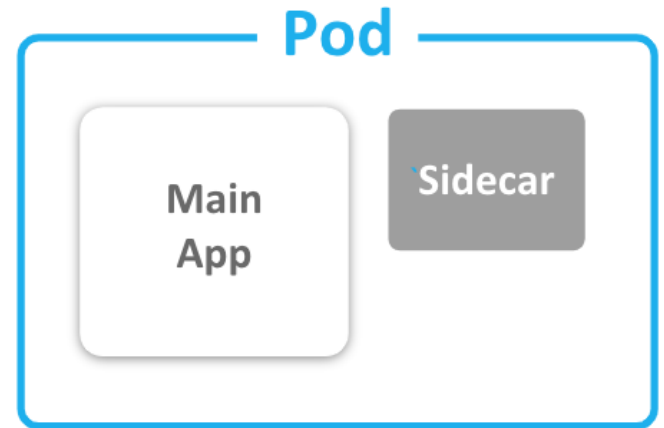
# Kubernetes

- Orchestrate our services
- Solves many cross cutting concerns



# A Sidecar Container

- Additional Container in a **pod**
- Provide **functionality** to support main app
- **Co-scheduled** together in replicas





**Sidecars can help us solve  
infrastructure concerns externally to  
our app**



# Let's get rid of stuff



# Logging Revisited



# Logging

- *Treat Logs as event streams*
- Use log forwarder
- Scrape the logs from Docker





# FluentD

- Run on every node (Daemonset)
- Declarative configuration to define log pipelines



# Logging - extra benefits

- Different policies
- Info/Debug -> low retention
- Error/Fatal -> high retention
- Plugins -> anonymization



- ☒ ~~Logging~~
- ☒ Error policies
- ☒ Authentication
- ☒ Monitoring



# Error policies Revisited

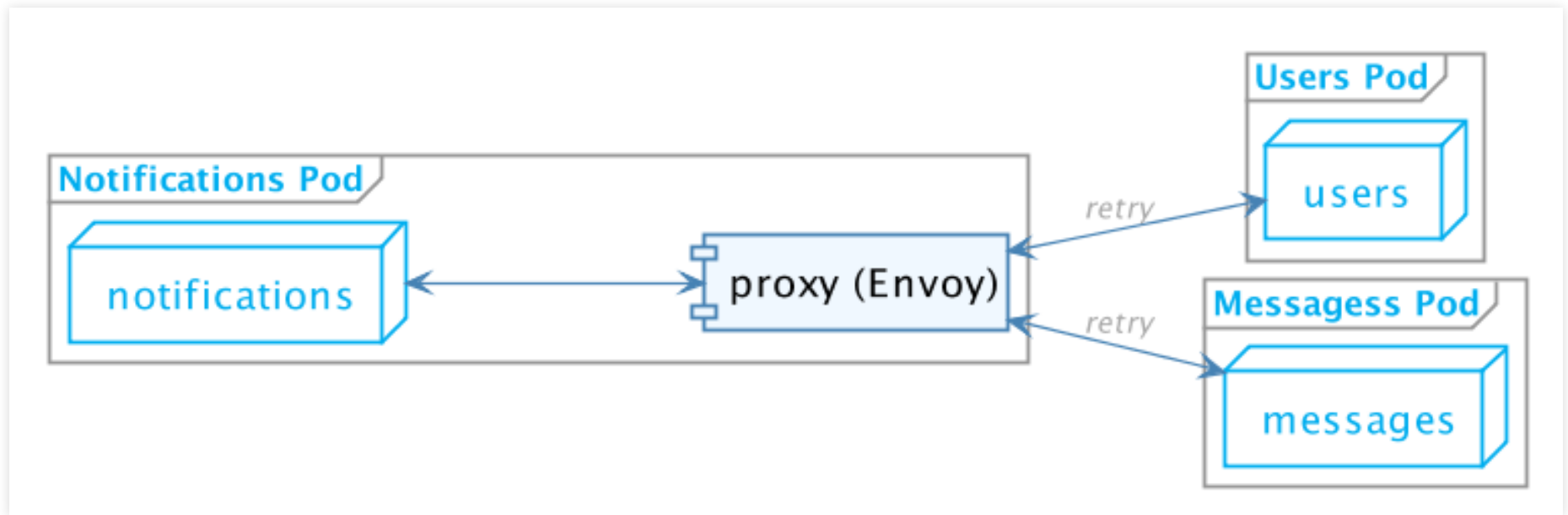


# Error policies

- Service-mesh, THE buzzword for 2018
- Istio
- Injects proxy sidecar



# Service mesh - Istio





# Service mesh - Resiliency

- Configurable failure handling



# Service mesh extra benefits

- Security
- Traffic management
- Tracing
- Too many to count...





- ☒ ~~Logging~~
- ☒ ~~Error recovery~~
- ☒ Authentication
- ☒ Monitoring



# Authentication Revisited

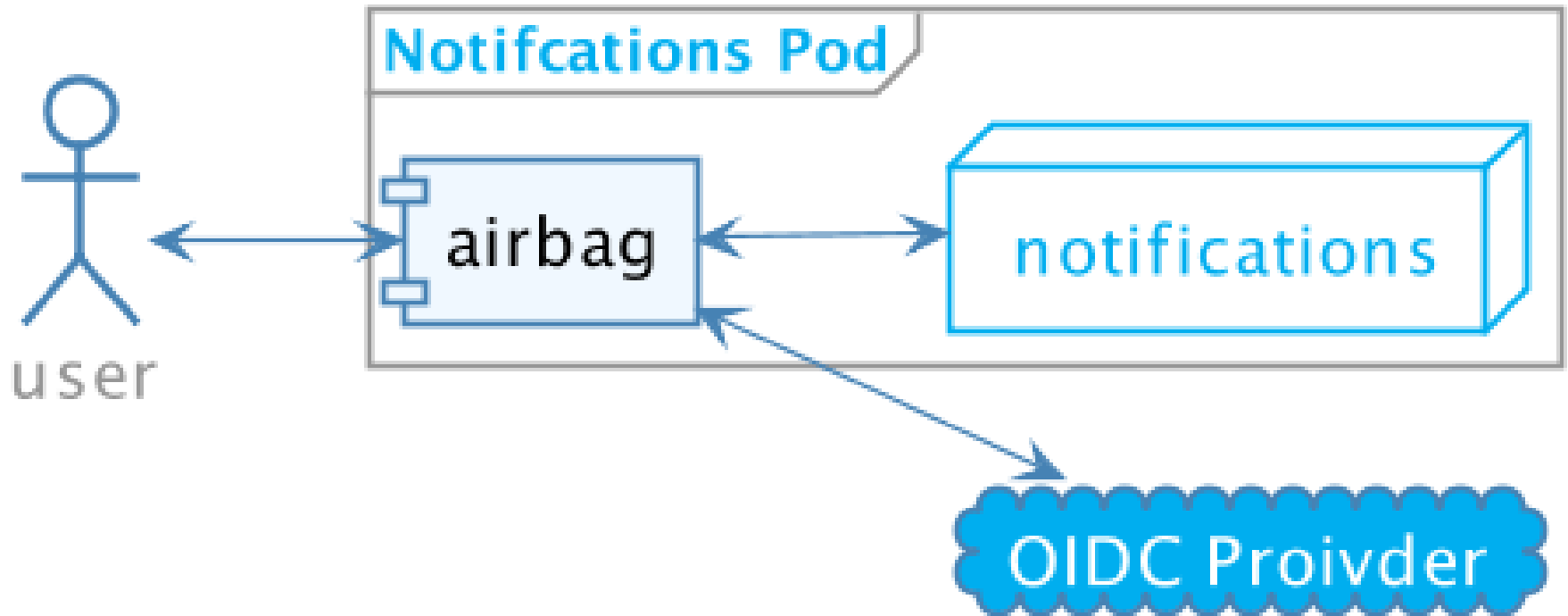


# Authentication

- Let's use a sidecar
- Solutio/Airbag



# Authentication - Airbag





# Authentication - Airbag

- Configuration yaml example
- Open source
- Probably be superseded by Istio's origin authentication



- ☒ ~~Logging~~
- ☒ ~~Resiliency~~
- ☒ ~~Authentication~~
- ☒ Monitoring



# Monitoring Revisited



# Monitoring







- ~~✓ Logging~~
- ~~✓ Resiliency~~
- ~~✓ Authentication~~
- ~~✓ Monitoring~~



# DEMO



# A new requirement

- It's **black friday!**
- We want to send **lots** of messages



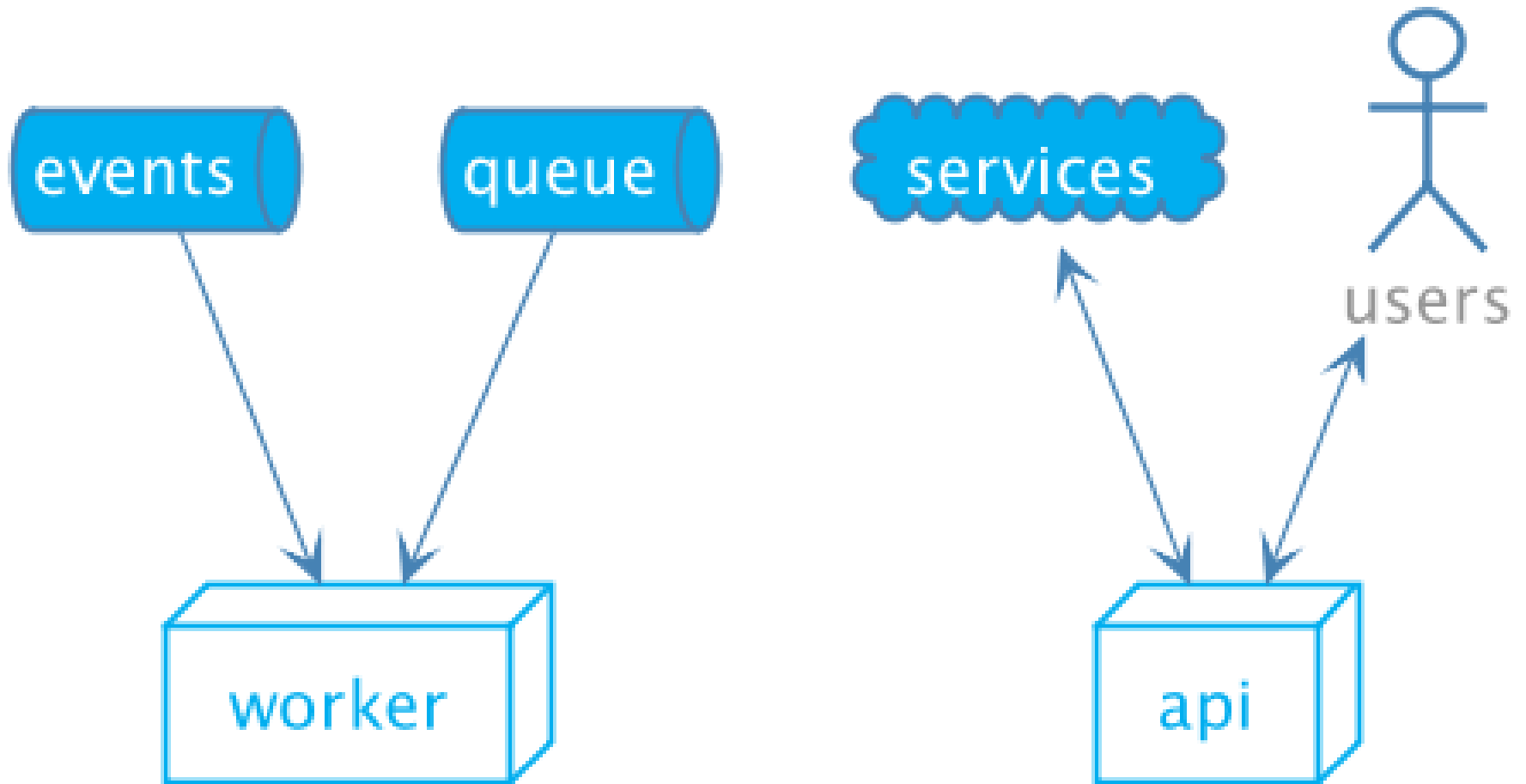
# A new requirement

- It's a batch process
- Let's use a queue



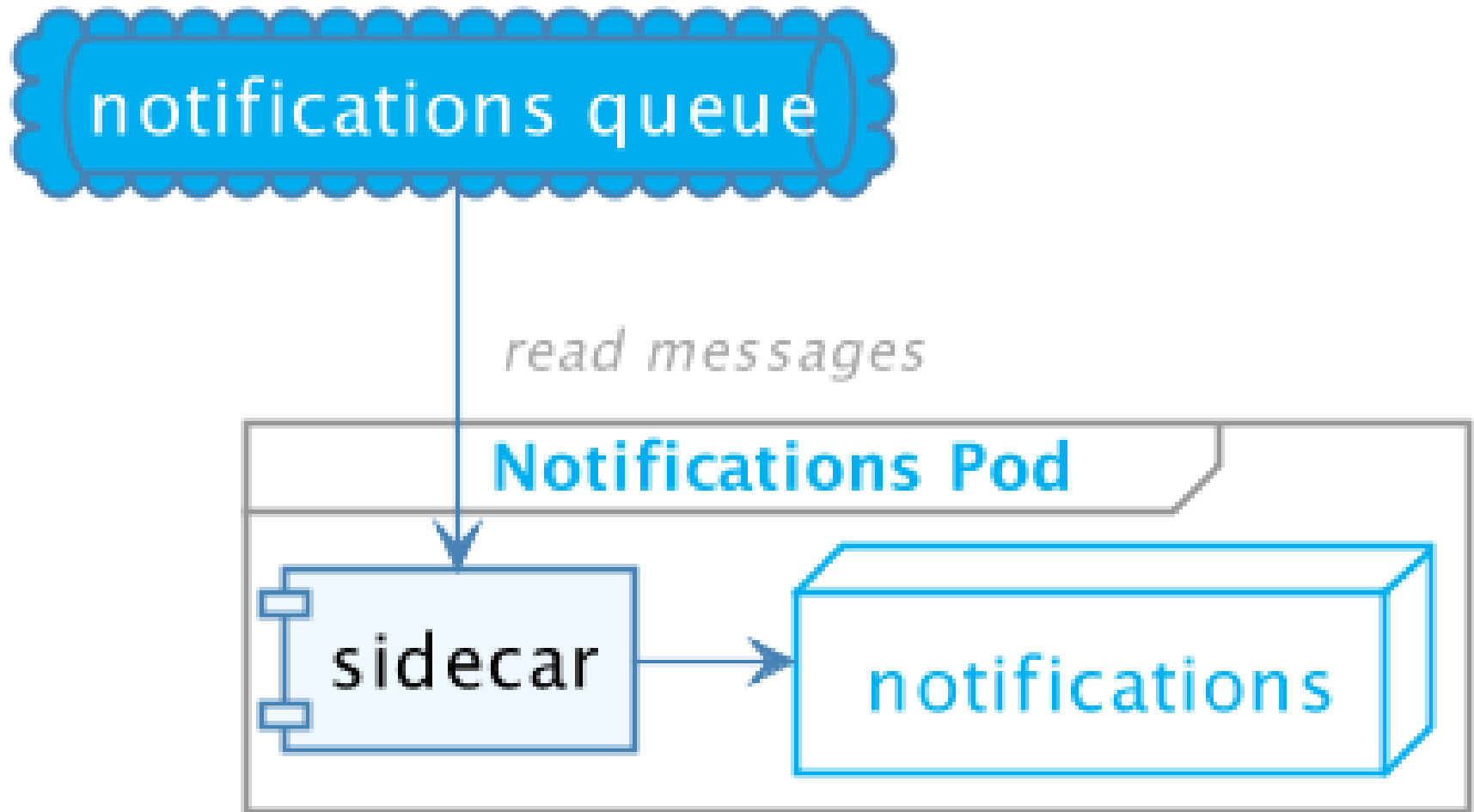
# How do we process queue items?

# Previous solutions





# Yet Another sidecar









# Deque Daemon Sidecar

- Solutio/DQD
- Read items from queue, activate service
- Back-pressure support
- Expose consumption metrics



# DQD - Demo



# DQD - benefits

- Scaled with service
- Testing
- Agnostic

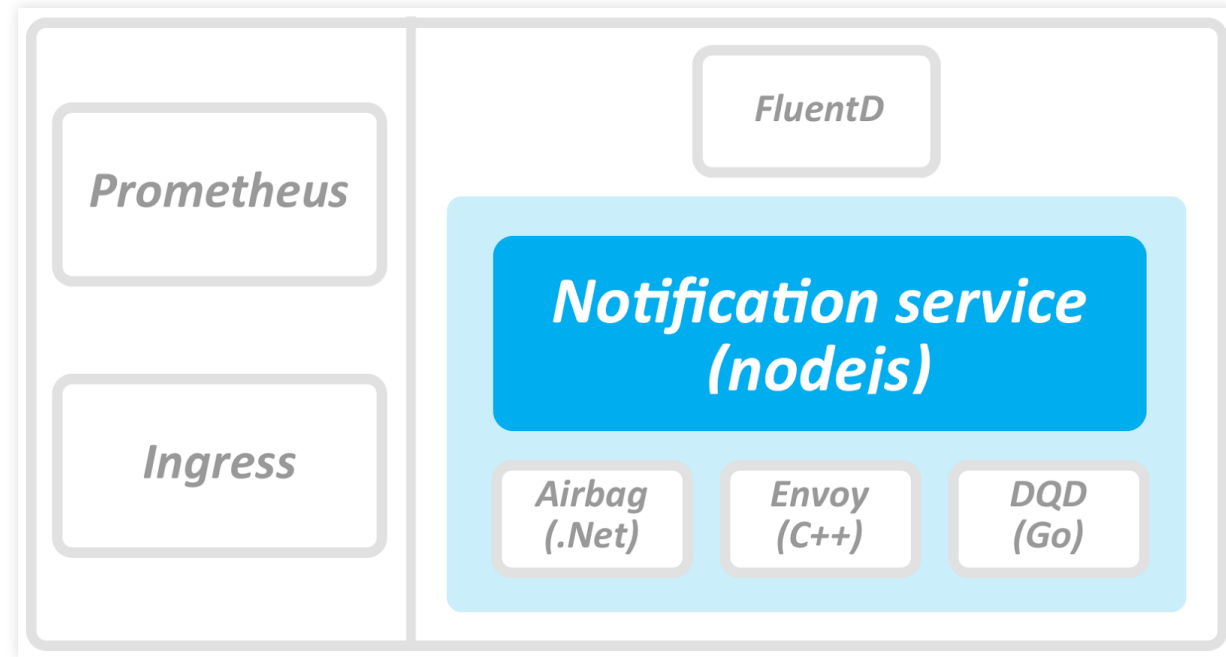


## Second iteration

- Lots of code elimination
- Dependencies removal
- Leaner, more testable service
- Declarative approach
- Better visibility



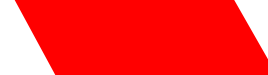
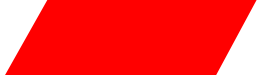
# Architecture



- External processes
- **Agents & Proxies**
- Co-scheduling

# Not all languages are built equal









## How did it affect us?

- Cleaner, leaner services
- Testing got easier
- Faster adoptions of new languages and tools
- No more Solutio.Logging/Monitoring/Auth/... packages
- Rapid adoption of k8s by teams
- We're still learning...



# How far can we go?

- **Gateways** - routing, caching, validation, rate limiting, policies
- **Configuration** - secrets, cloud resource binding
- **Tools** - remote debugging/profiling Supporting services - analytics, feature flags, etc...
- **Platform** - FAAS/Serverless/Workflows
- **Probably more...** (<https://landscape.cncf.io/>)



As a developer, I want to **focus on building features** that deliver **business value**



**What can the  
infrastructure do for  
you?**



**Thank you.**



# Questions?

(Yes, we're hiring 😊 )



# Additional resources - tools

- AirBag - [github.com/soluto/airbag](https://github.com/soluto/airbag)
- DQD - [github.com/soluto/dqd](https://github.com/soluto/dqd)
- FluentD - <https://www.fluentd.org/>
- Prometheus - <https://prometheus.io>
- Istio - <https://istio.io/>
- Skaffold - [github.com/GoogleContainerTools/skaffold](https://github.com/GoogleContainerTools/skaffold)
- Oidc-server-mock - [github.com/Soluto/oidc-server-mock](https://github.com/Soluto/oidc-server-mock)
- FluentD k8s level filter - [github.com/Soluto/fluent-plugin-kubernetes-log-level](https://github.com/Soluto/fluent-plugin-kubernetes-log-level)



## Additional resources

- CNCF landscape - <https://landscape.cncf.io/>
- Design patterns for container-based distributed systems - [https://static.googleusercontent.com/media/research.google.com/en/pubs/pubs\\_google103666726.pdf](https://static.googleusercontent.com/media/research.google.com/en/pubs/pubs_google103666726.pdf)
- Introduction to modern network load balancing and proxying - <https://blog.envoyproxy.io/introduction-to-modern-network-load-balancing-a57f6ff80236>