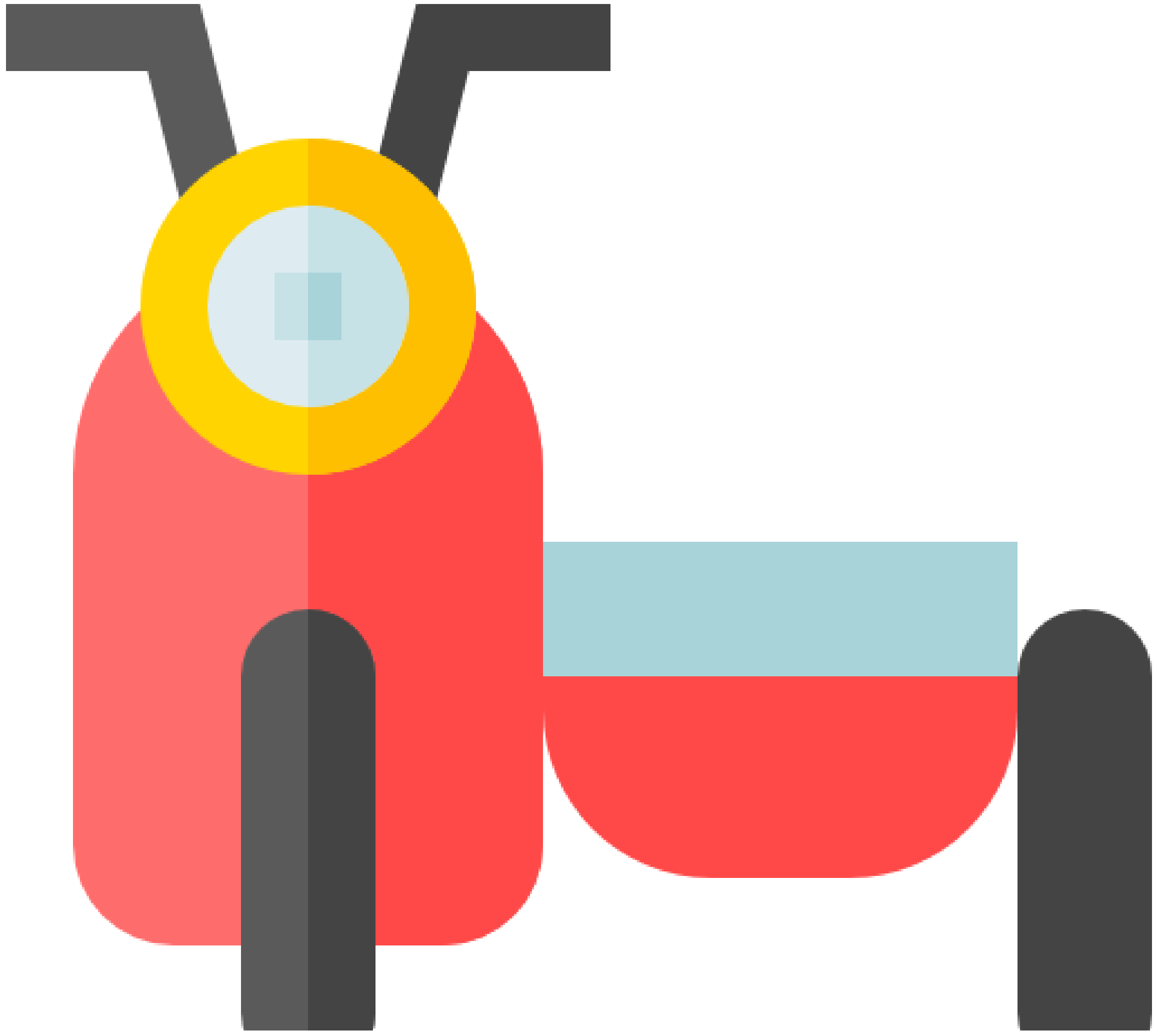


Sidecars

Diego Pacheco

About me...

- ❑ Cats Father
- ❑ Software Architect
- ❑ Agile Coach
- ❑ SOA/Microservices Expert
- ❑ DevOps Practitioner
- ❑ Author
- ❑ Speaker

diegopacheco

@diego_pacheco

http://diego-pacheco.blogspot.com.br/

tinyurl.com/diegopacheco

Diego Pacheco
Building Applications with Scala
Write modern, scalable, and reactive applications with the power of Scala
Packt

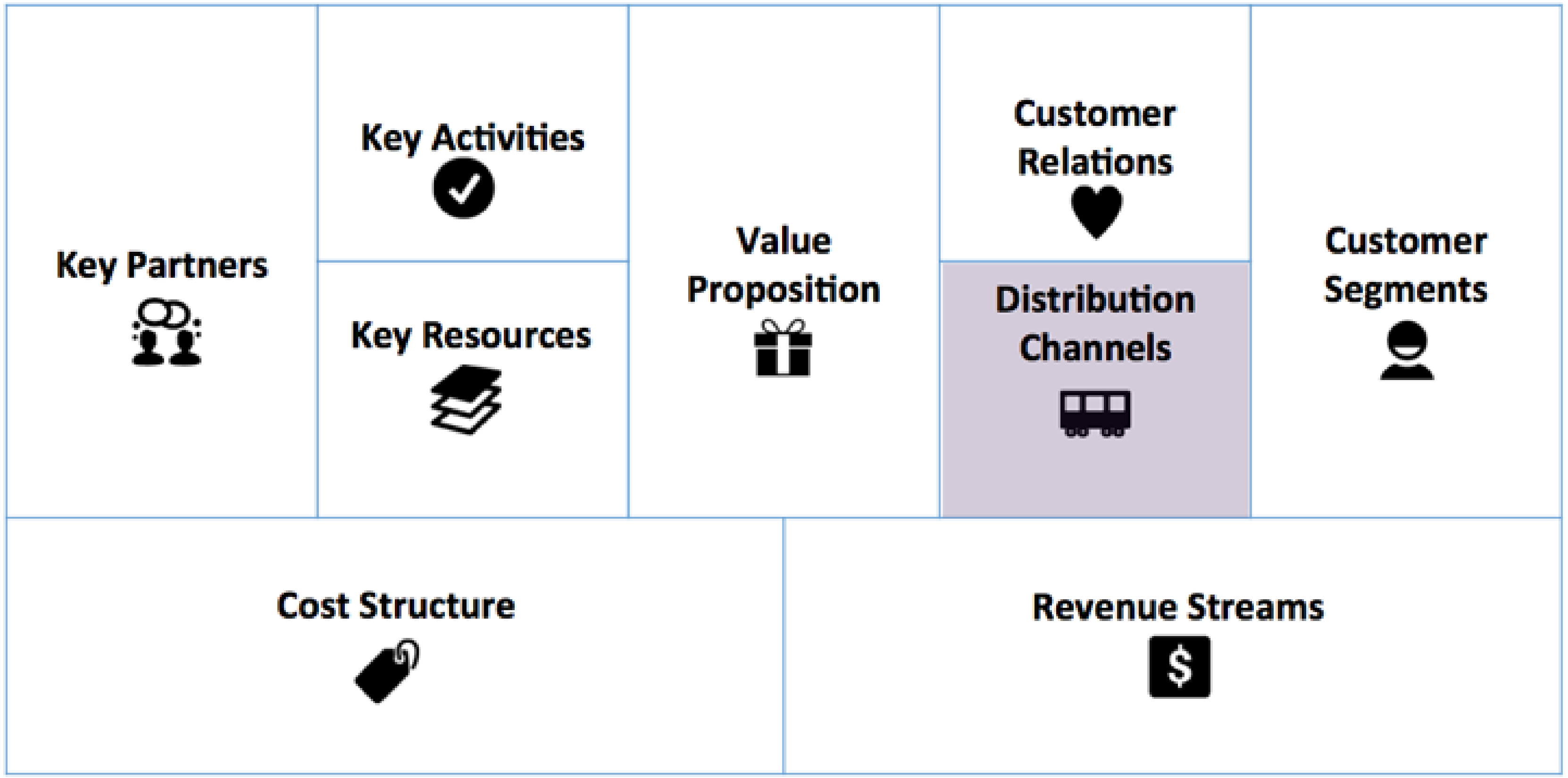
Diego Pacheco
Building Effective Microservices
Explore microservices and their implementation hands-on
Packt

https://diegopacheco.github.io/

Disclaimer

Not a motorbike presentation.

How can we distribute Software?



How can we distribute Software?

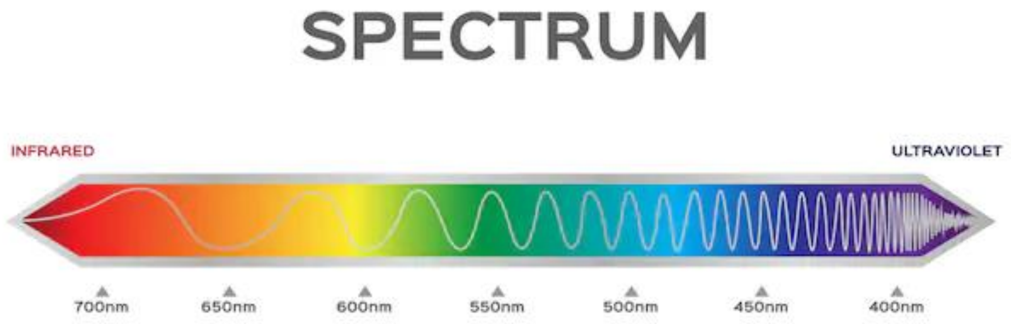
- Tooling
- Libraries (Shared Jars)
- SOA Services (Remote APIs)
- Internal Managed Services / Self Service Platform (Generic UI, Jenkins, etc...)
- Platforms (Kubernetes / Istio)
- Sidecars

How can we distribute Software?

- Tooling
- Libraries (Shared Jars)
- SOA Services (Remote APIs)
- Internal Managed Services / Self Service Platform (Generic UI, Jenkins, etc...)
- Platforms (Kubernetes / Istio)
- Sidecars

Sidecar All The Things?

MIN
MAX

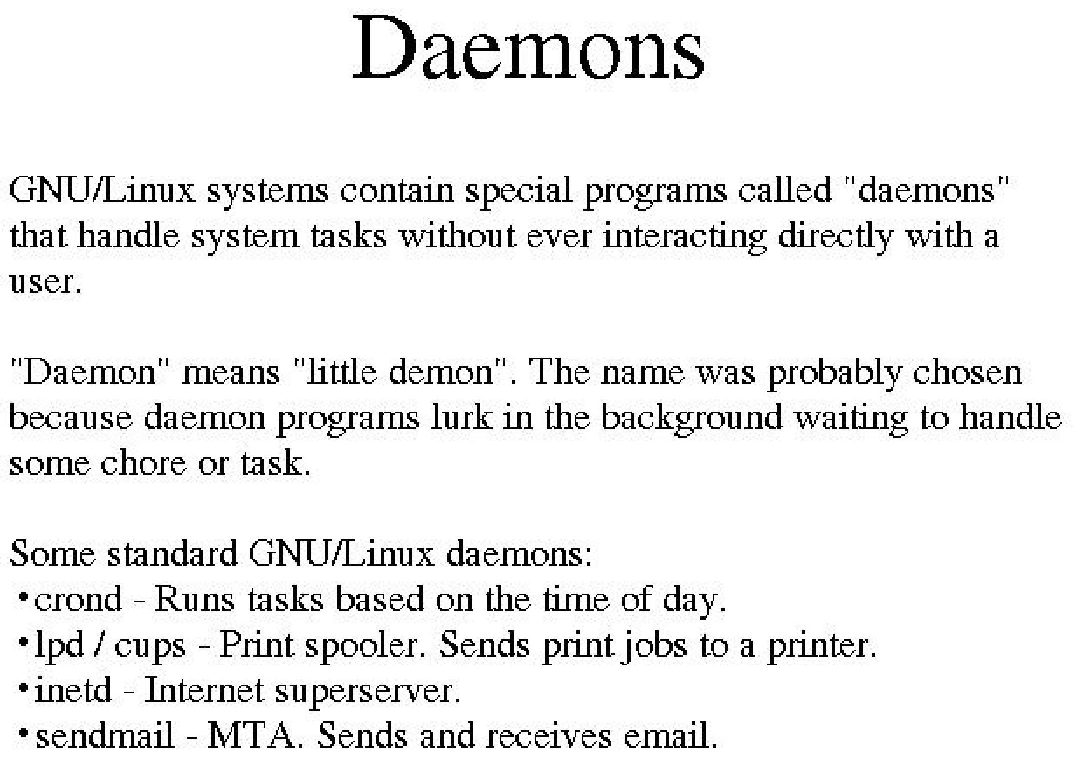

SPECTRUM
INFRARED
ULTRAVIOLET
700nm
650nm
600nm
550nm
500nm
450nm
400nm

Disclaimer: Not only for Containers.

How to Build a Sidecar?

- YOU DON'T NEED:
  - Specific Language
  - Specific Library
  - Specific Framework
  - Specific Technology
  - Run software on The Cloud
  - Run software on K8s
  - Run software on Istio

Sidecars are not "completely new" idea



Daemons

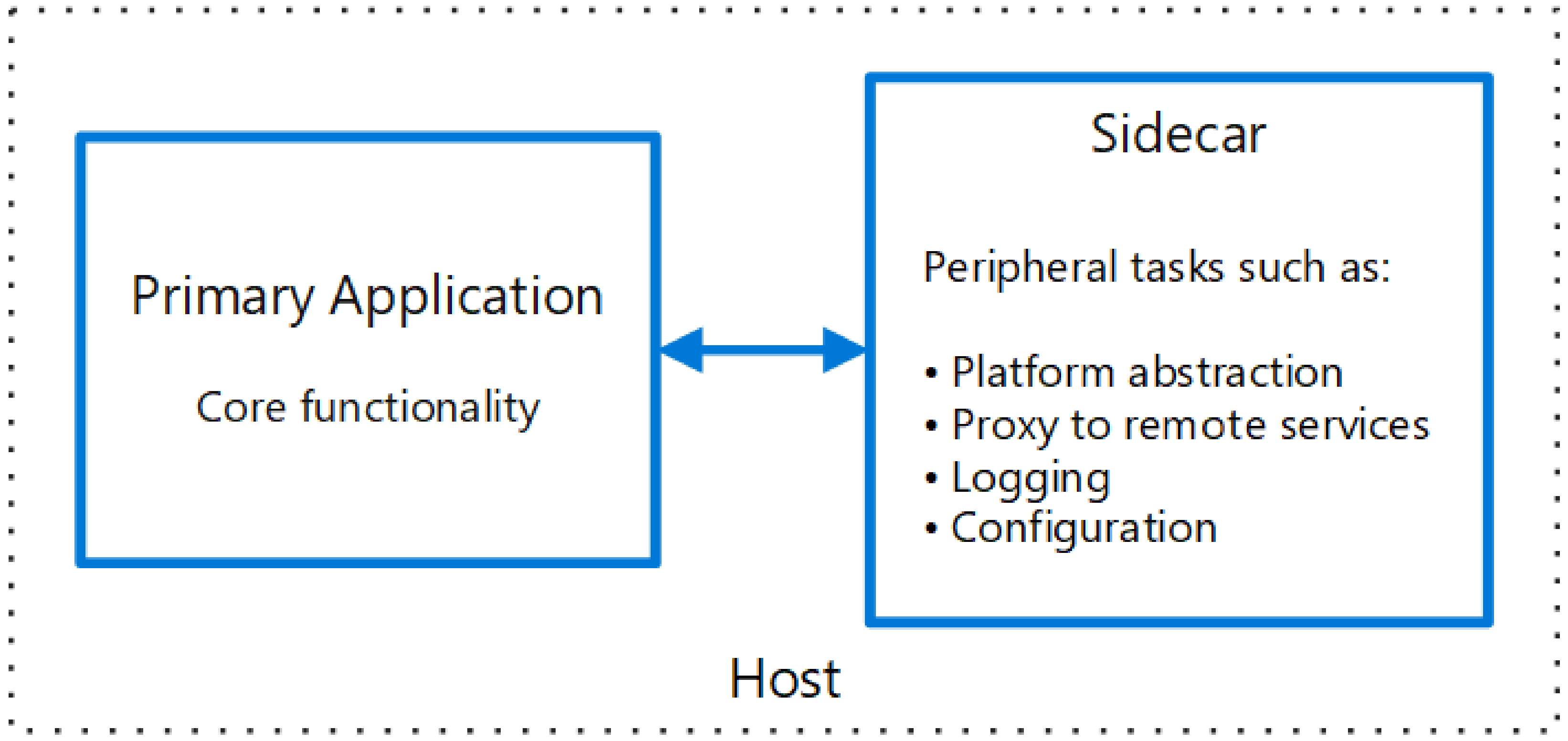
GNU/Linux systems contain special programs called "daemons" that handle system tasks without ever interacting directly with a user.

"Daemon" means "little demon". The name was probably chosen because daemon programs lurk in the background waiting to handle some chore or task.

Some standard GNU/Linux daemons:

- crond - Runs tasks based on the time of day.
- lpd / cups - Print spooler. Sends print jobs to a printer.
- inetd - Internet superserver.
- sendmail - MTA. Sends and receives email.

Sidecar Pattern



Sidecar Benefits

- Provides Safe Re-use
- Decoupling
- Isolation
- Encapsulation
- Avoid Binary Coupling
- Fredon do use any language / library without affecting the "main application"
- Upgrades / Deploys independent from the "main application"
- NO SPOF & Scalability
- Avoid Massive Migrations (Libraries Issue)
- Works Perfectly with Containers / K8s

Sidecar Drawbacks

- Becomes part of the Reliability Path
- Requires Great Observability or becomes an awful Blackbox
- More Deployment Complexity (EC2)
- Might be hard to debug for Application Tier

Performance Tradeoffs

- Different languages / libs might allow better performance but might create debuting issues for other languages. I.e. Sidecar in Java and App in JavaScript.
- HTTP (using Netty) will give you
  - ~1..4 ms overhead
- There faster communication mechanism like IPC (0 ms)
  - Aeron (Java)
  - OpenHFT/Chronicle-Queue (Java)
  - IPC Drawbacks
    - Much more code
    - Much more complex (low level programing)
    - Much more obscure to debug / understand

It's Perfectly Possible do Sidecars on EC2

NETFLIX

OSS

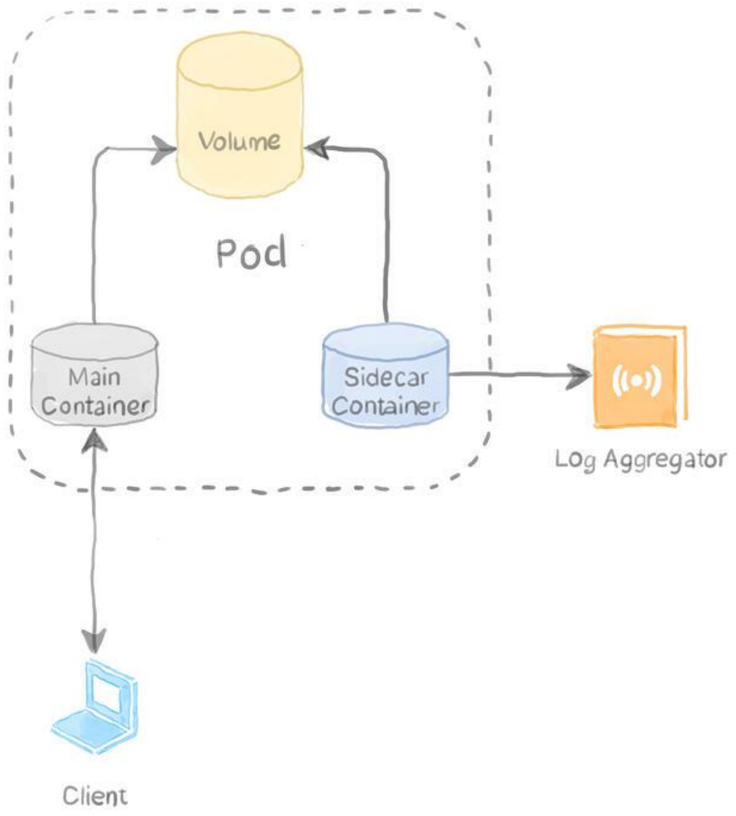
Netflix Sidecars on EC2

Manager

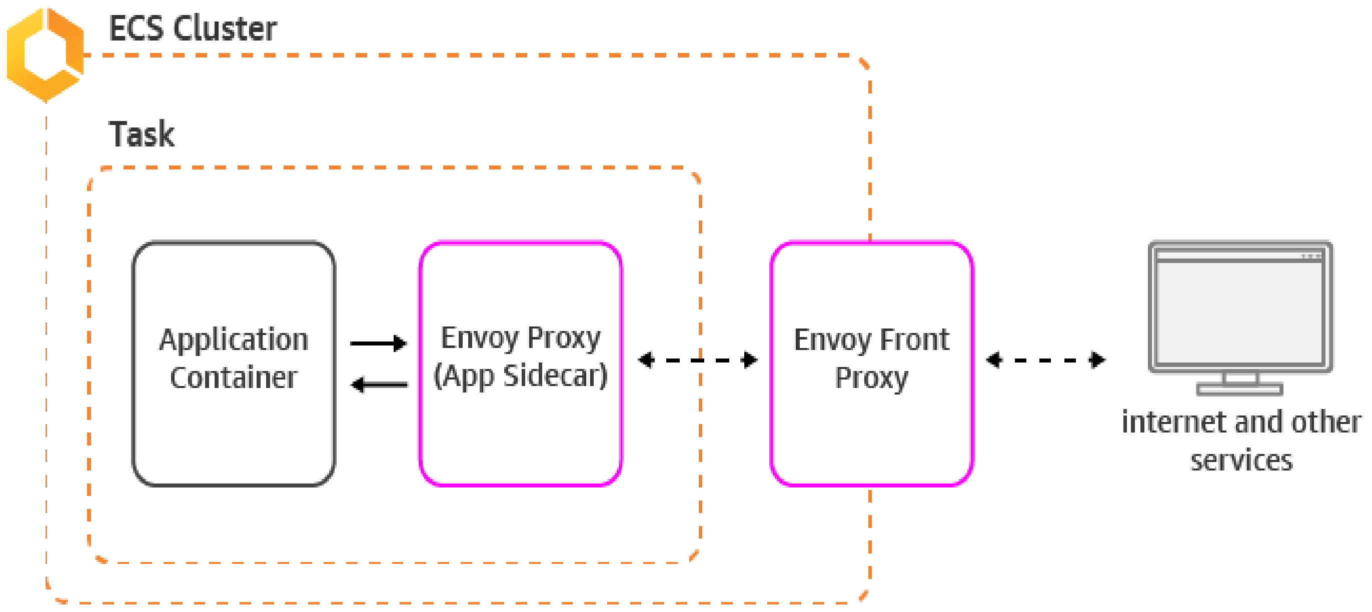
Standard Practice in K8s

```
apiVersion: v1
kind: Pod
metadata:
  name: webserver
spec:
  volumes:
    - name: shared-logs
      emptyDir: {}
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - name: shared-logs
          mountPath: /var/log/nginx

    - name: sidecar-container
      image: busybox
      command: ["sh","-c","while true; do cat /var/log/nginx/access.log /var/log/nginx/
      volumeMounts:
        - name: shared-logs
          mountPath: /var/log/ngin
```



Envoy



- Advanced Load Balancer
  - Retries
  - Circuit Breaker
  - Global Rate Limiting
  - Request Shadowing
- HTTP/2 & gRPC Support
- Observability (Deep L7)
- Declarative
- Sidecar :D
- Used in ISTIO (Service Mesh)

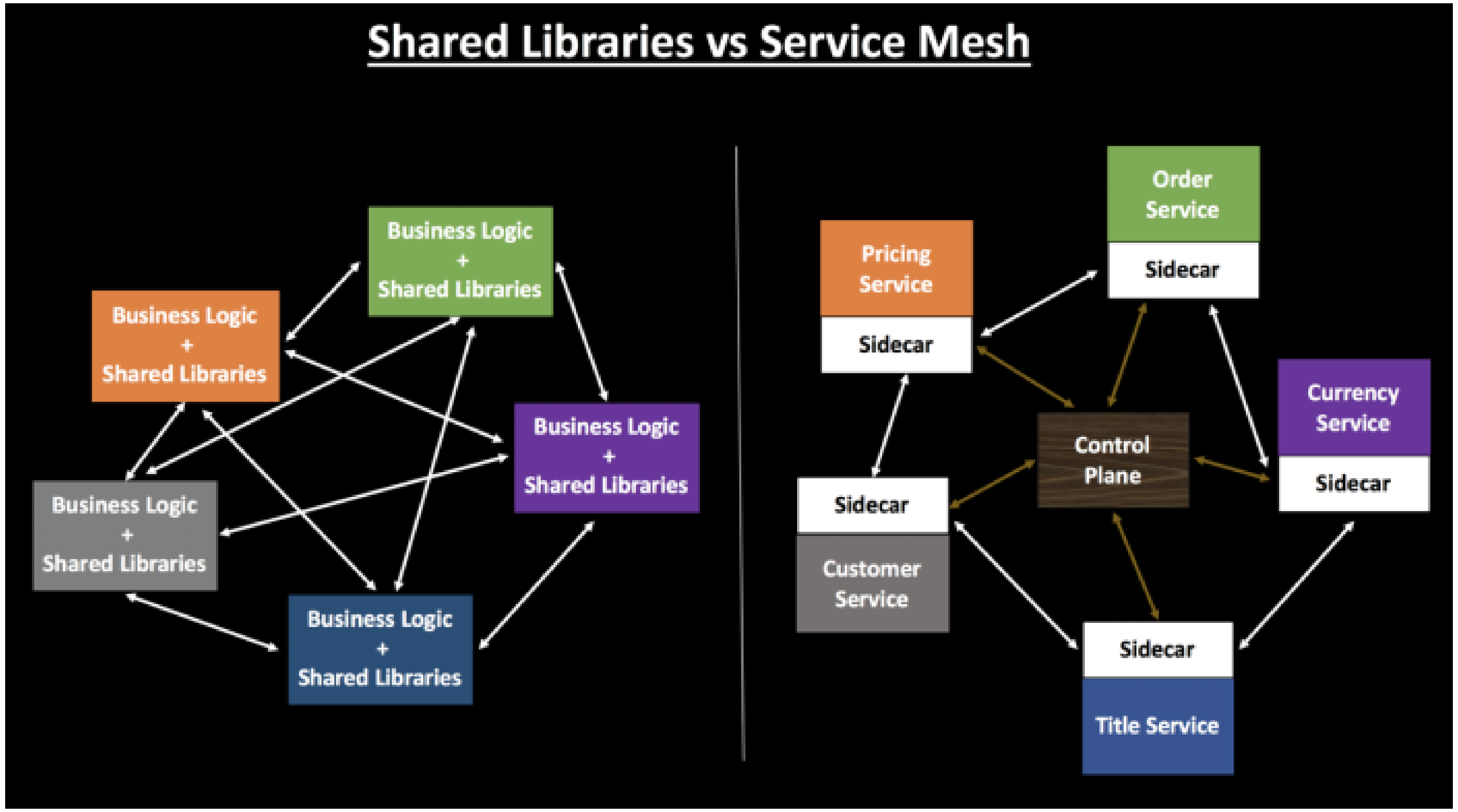
Envoy

```yaml
listeners:
- name: listener_0
  address:
    socket_address: { address: 0.0.0.0, port_value: 10000 }
  filter_chains:
  - filters:
    - name: envoy.filters.network.http_connection_manager
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
        stat_prefix: ingress_http
        codec_type: AUTO
        route_config:
          name: local_route
          virtual_hosts:
          - name: local_service
            domains: ["*"]
            routes:
            - match: { prefix: "/" }
              route: { host_rewrite_literal: www.google.com, cluster: service_google }
        http_filters:
        - name: envoy.filters.http.router
```

Smart endpoints and dumb pipes

- Dumb Pipes
  - HTTP or Lightweight messaging
  - "Be" the web no "behind the web"
- Smart Endpoints
  - As Decoupled and Cohesive service as possible
  - Doing Routing and Choreography decisions

Smart Sidecars and dumb pipes


Shared Libraries vs Service Mesh
Business Logic + Shared Libraries
Business Logic + Shared Libraries
Business Logic + Shared Libraries
Business Logic + Shared Libraries
Business Logic + Shared Libraries
Pricing Service
Sidecar
Order Service
Sidecar
Currency Service
Sidecar
Control Plane
Sidecar
Customer Service
Sidecar
Title Service

How we should avoid?

- When you need Extreme Performance and Extreme Low Latency
- The solution is not that complicated
- The complexity does not pay off.
- The code does not change much and is super simple
  - (Let's not turn Date Utils on sidecars :-))
- You want scale the sidecar apart for the application (***)

How we can consider use it?

- When we want avoid binary coupling
- When we want make the solution completely transparent for the application (reduce coupling).
- When we really need use different languages / libs for performance of some design reason.
- When you are using containers / Service Mesh.

Remember There are other options...

- Tooling
- Libraries (Shared Jars)
- SOA Services (Remote APIs)
- Internal Managed Services / Self Service Platform (Generic UI, Jenkins, etc...)
- Platforms (Kubernetes / Istio)

Interesting Reading

- https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar
- https://www.magalix.com/blog/the-sidecar-pattern
- https://github.com/Netflix/Prana
- https://github.com/Netflix/Priam
- https://github.com/Netflix/Raigad
- https://github.com/Netflix/dynomite-manager
- https://github.com/envoyproxy/envoy
- https://github.com/istio

Thank You!

Sidecars

Diego Pacheco