QAWARE
SOFTWARE ENGINEERING

O'REILLY
Software Architecture
CONFERENCE

Putting microservices on a diet with Istio

Mario-Leander Reimer, QAware GmbH

mario-leander.reimer@qaware.de

London, 29th October 2018

Mario-Leander Reimer

Principal Software Architect, QAware GmbH

- Developer && Architect
- 20+ years of experience
- #CloudNativeNerd
- Open Source Enthusiast
- Speaker && Author

Mail: mario-leander.reimer@qaware.de

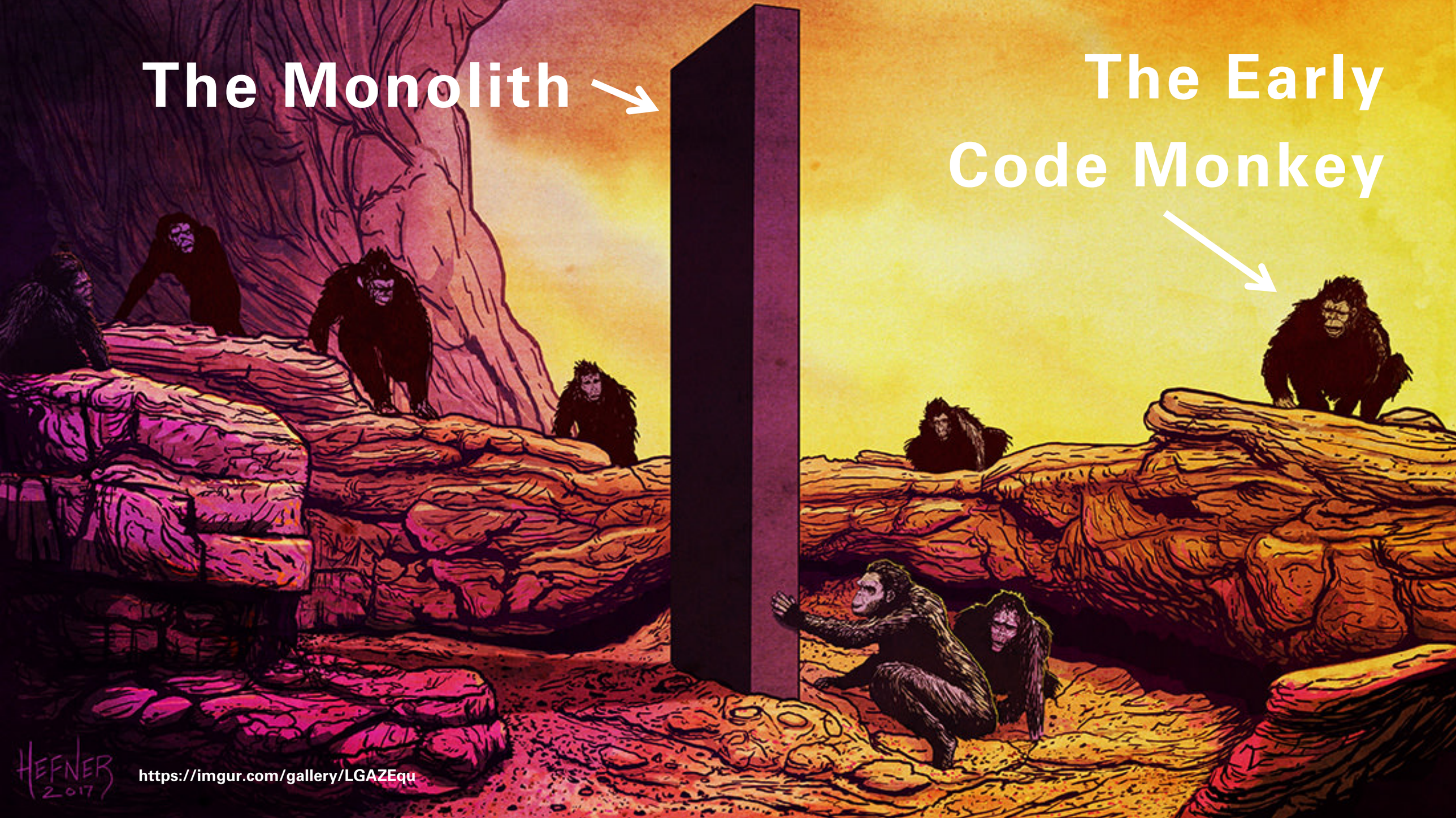
Twitter: @LeanderReimer

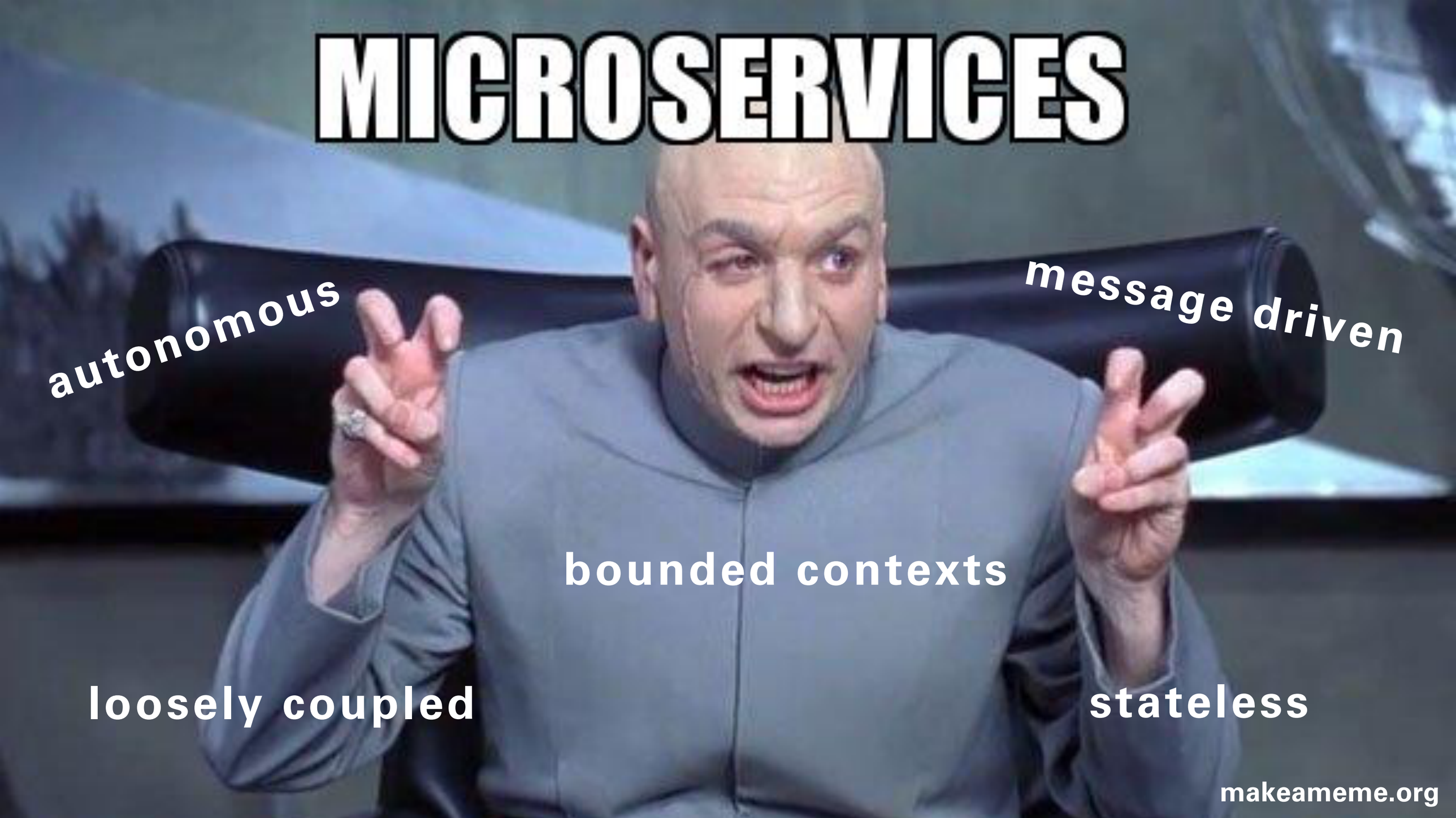
Github: https://github.com/lreimer/

Slides: https://speakerdeck.com/lreimer/

Fork me on Github.

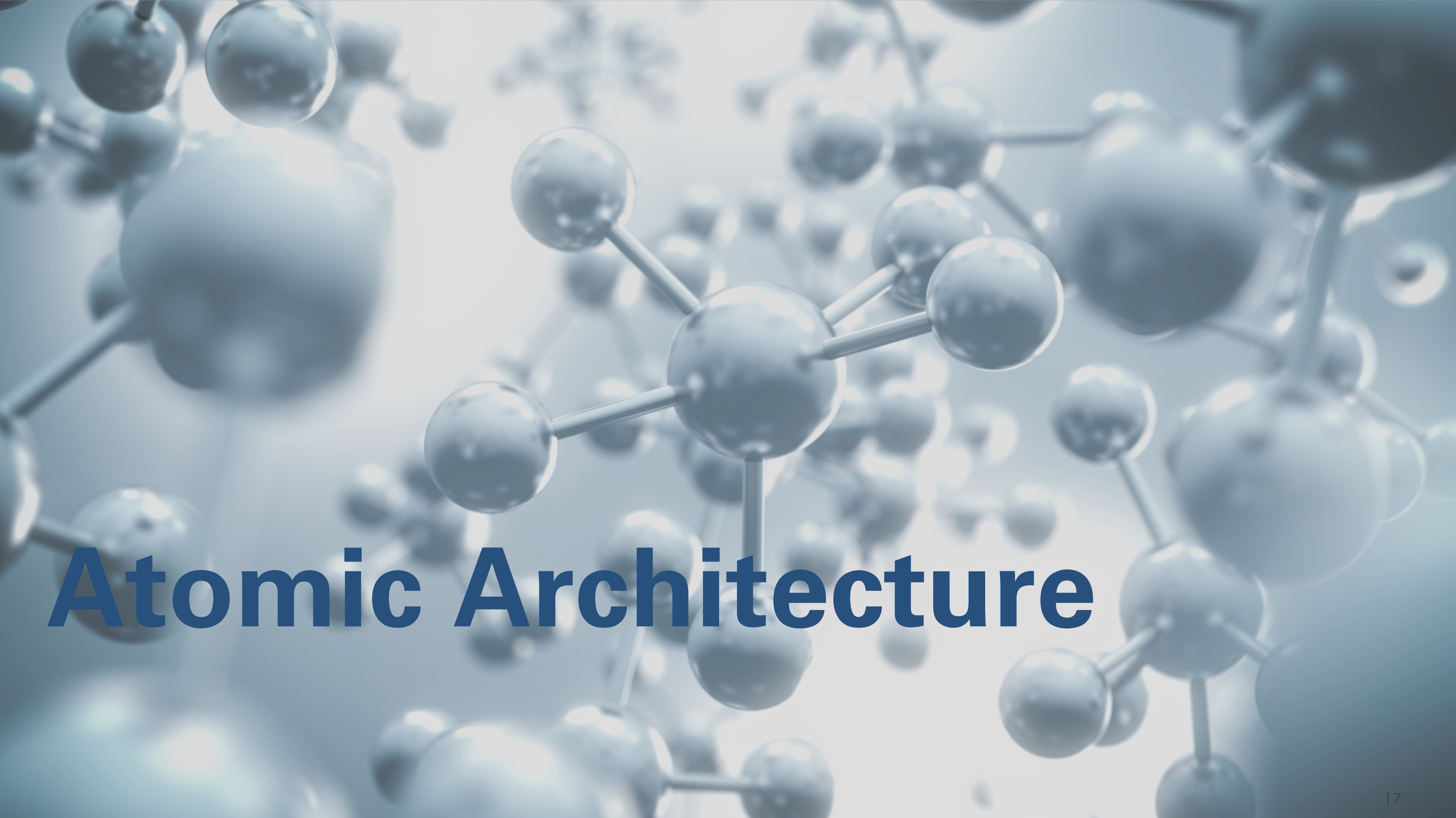
https://github.com/lreimer/microservice-diet-with-istio


The Monolith
The Early
Code Monkey
HEFNER
2017
https://imgur.com/gallery/LGAZEqu


MICROSERVICES
autonomous
message driven
bounded contexts
loosely coupled
stateless
makeameme.org

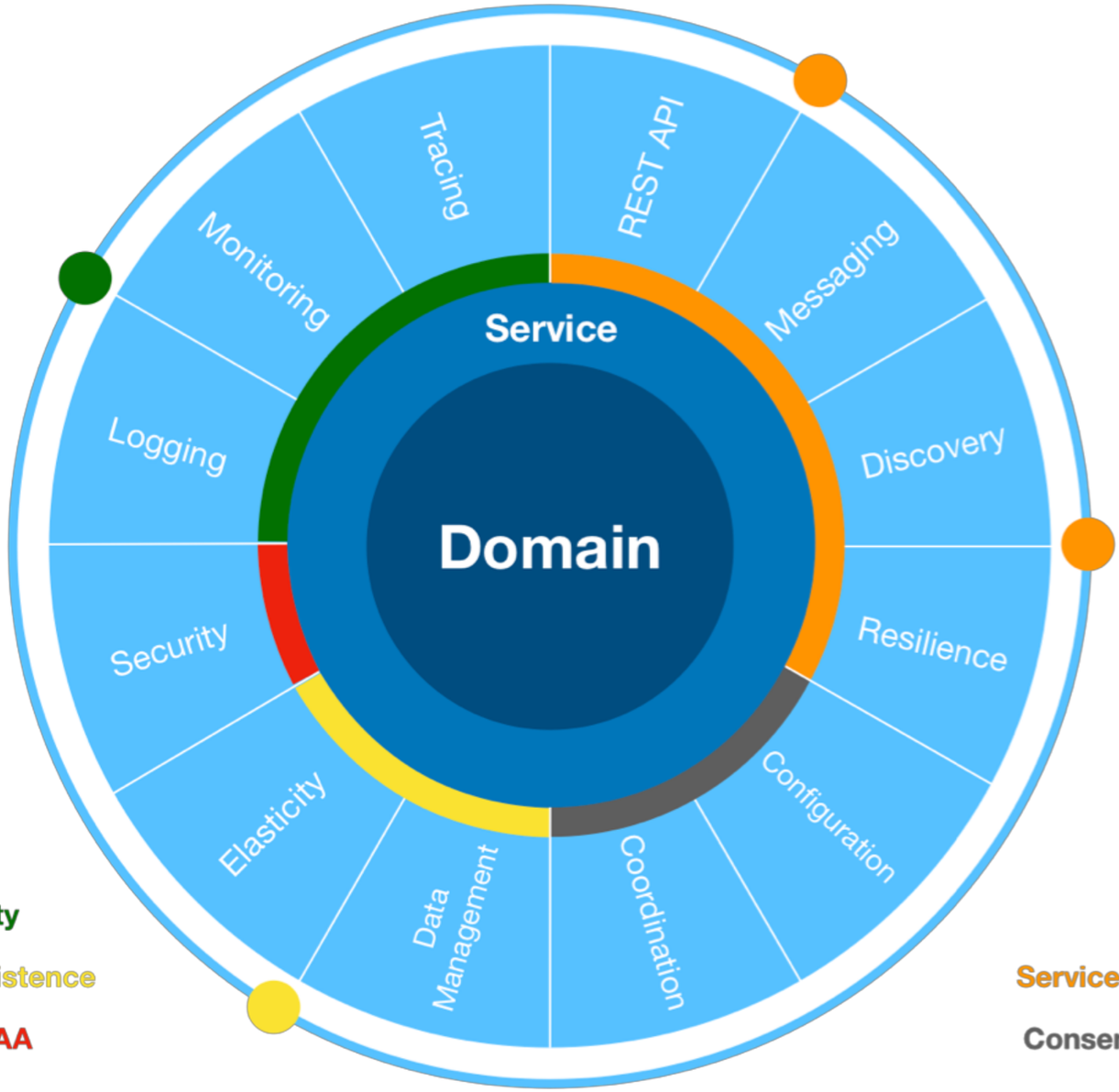
Essential Cloud-native Design Principles.

- **Design for Distribution**: Containers; microservices; API driven development.
- **Design for Configuration**: One image, multiple environments.
- **Design for Resiliency**: Fault-tolerant and self-healing.
- **Design for Elasticity**: Scales dynamically and reacts to stimuli.
- **Design for Delivery**: Short roundtrips and automated provisioning.
- **Design for Performance**: Responsive; concurrent; resource efficient.
- **Design for Automation**: Automated Dev & Ops tasks.
- **Design for Diagnosability**: Cluster-wide logs, metrics and traces.
- **Design for Security**: Secure Endpoints, API-Gateways, E2E-Encryption



Atomic Architecture

Atomic Microservice Blueprint.



Diagnosability

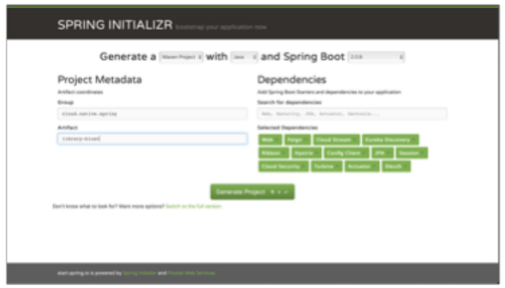
State & Persistence

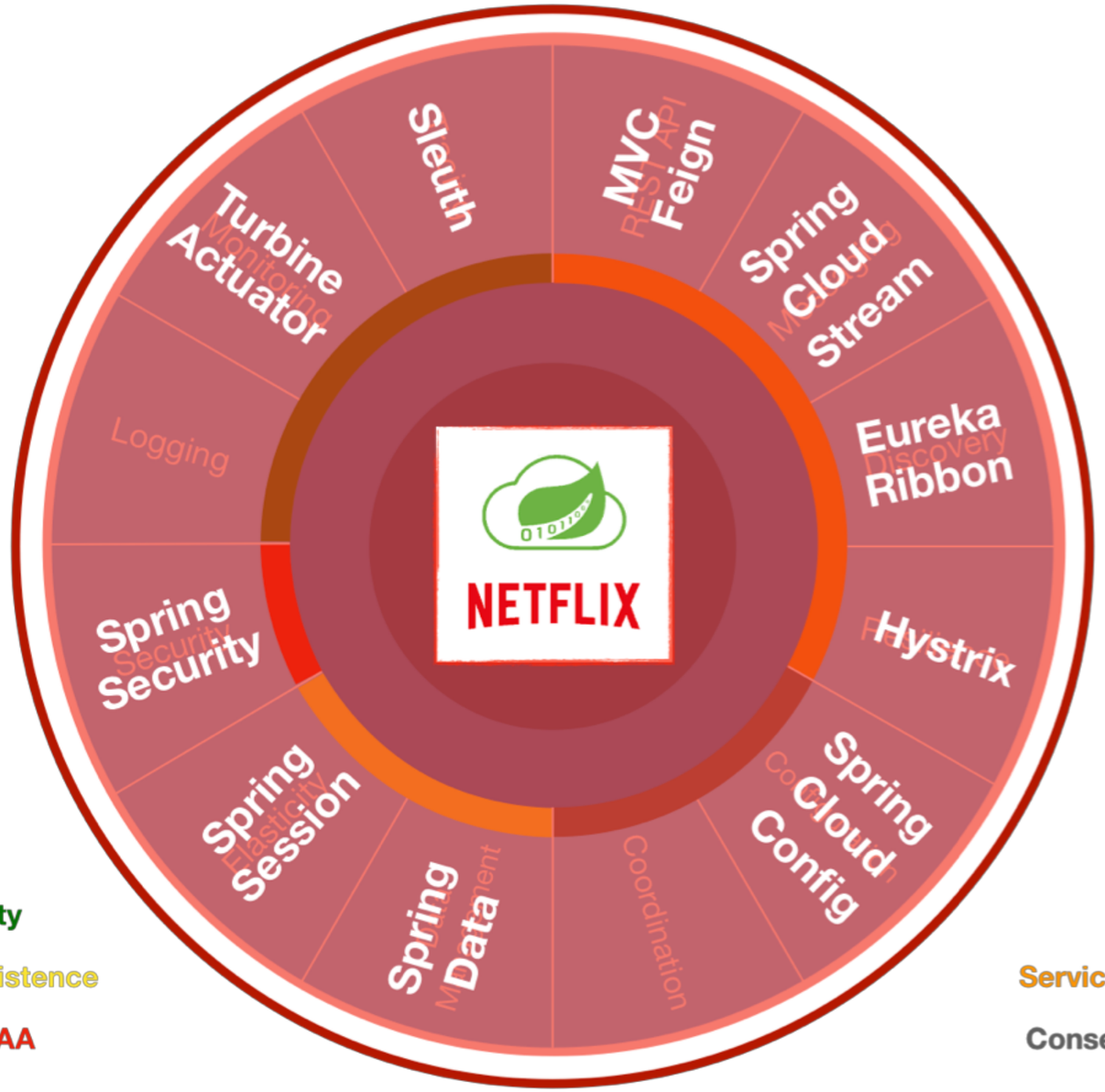
Security & AAA

Service Exposition

Consens & Config

Concrete Blueprint Incarnation with Spring Cloud Netflix.





Some Facts:

- 58 MB Uberjar
- 192 Dependencies
- 3 KB Classes

Diagnosability

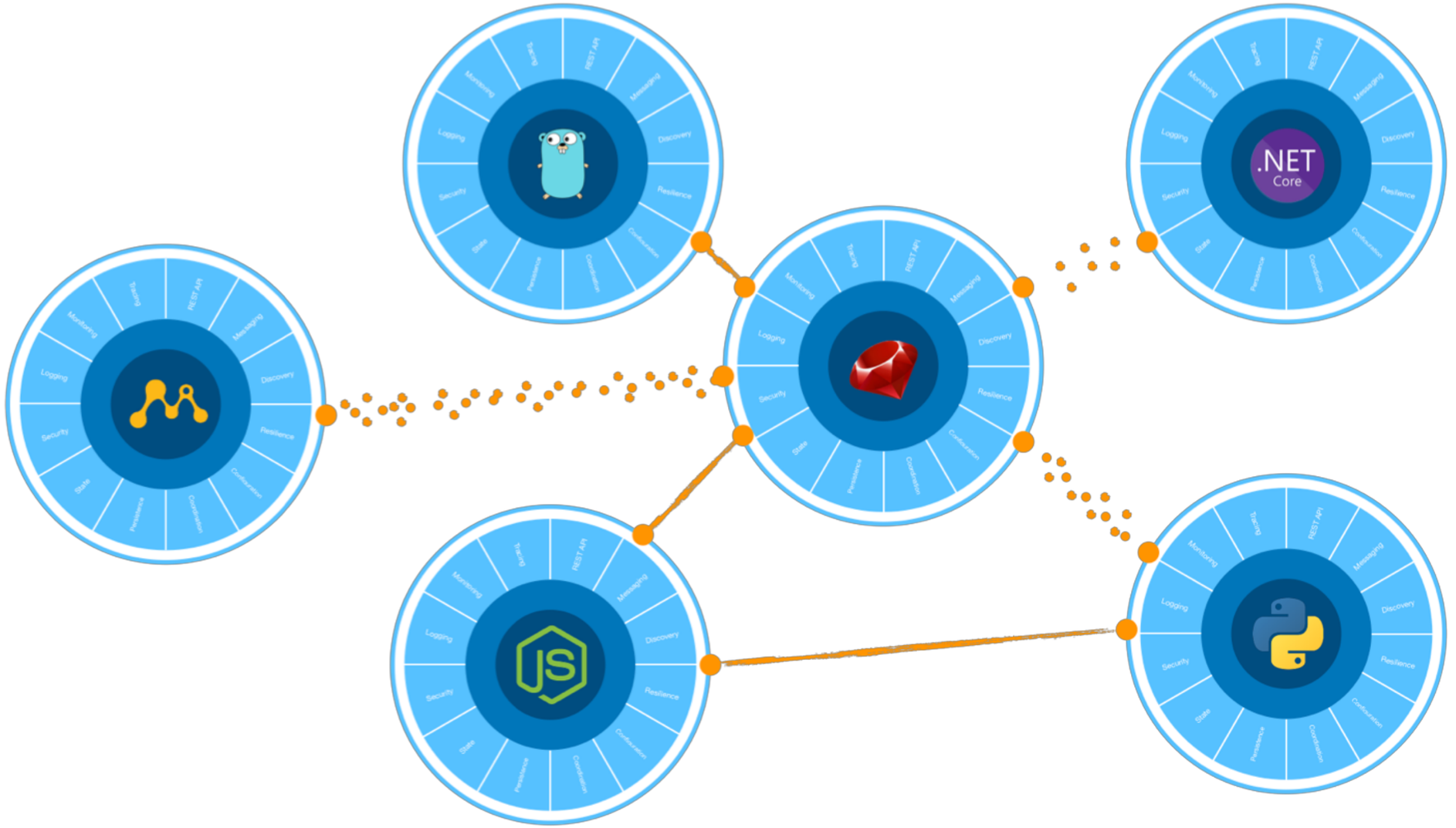
State & Persistence

Security & AAA

Service Exposition

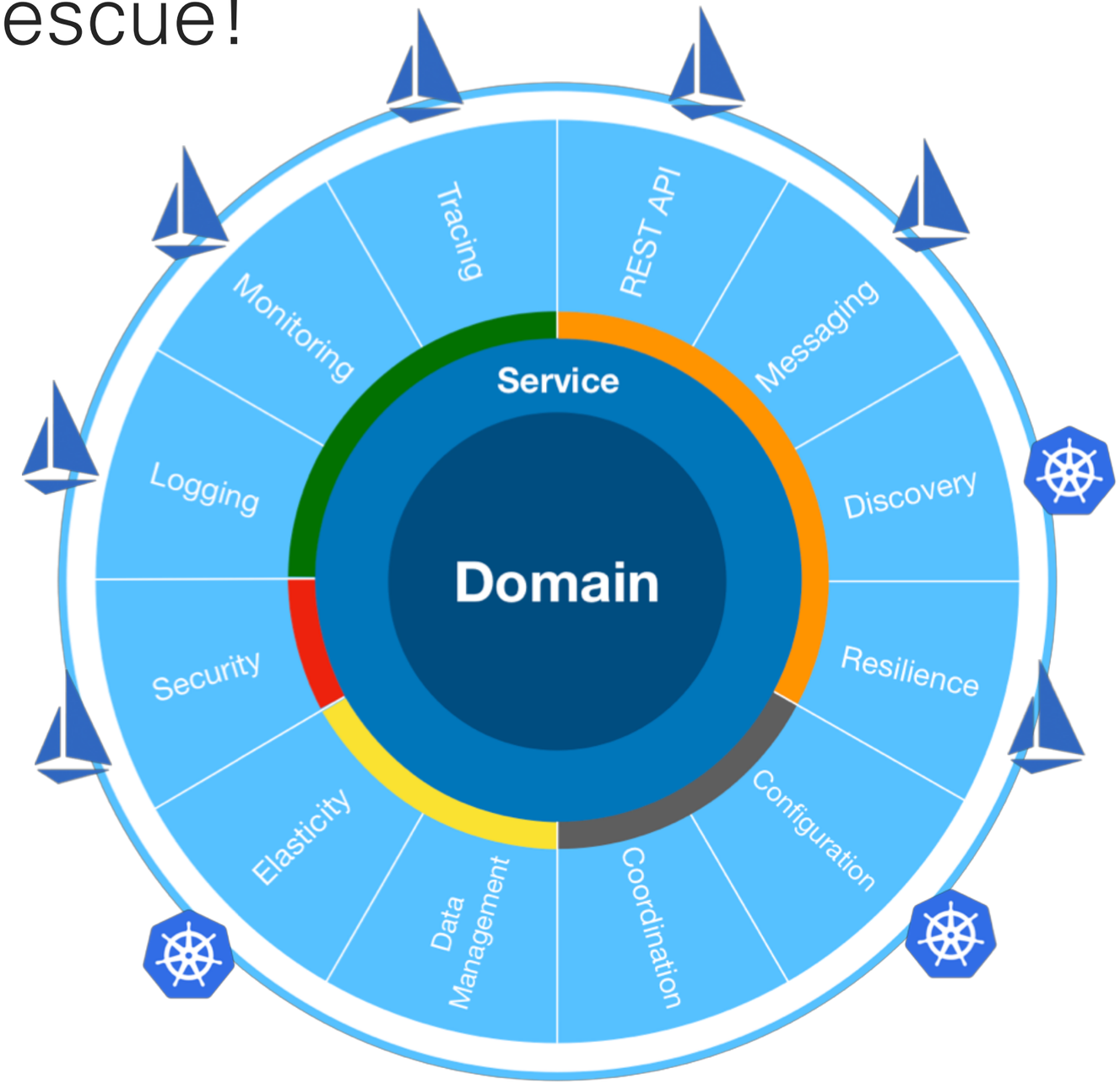
Consens & Config

A polyglot microservice architecture suffers from severe library bloat and bad maintainability in the long run.

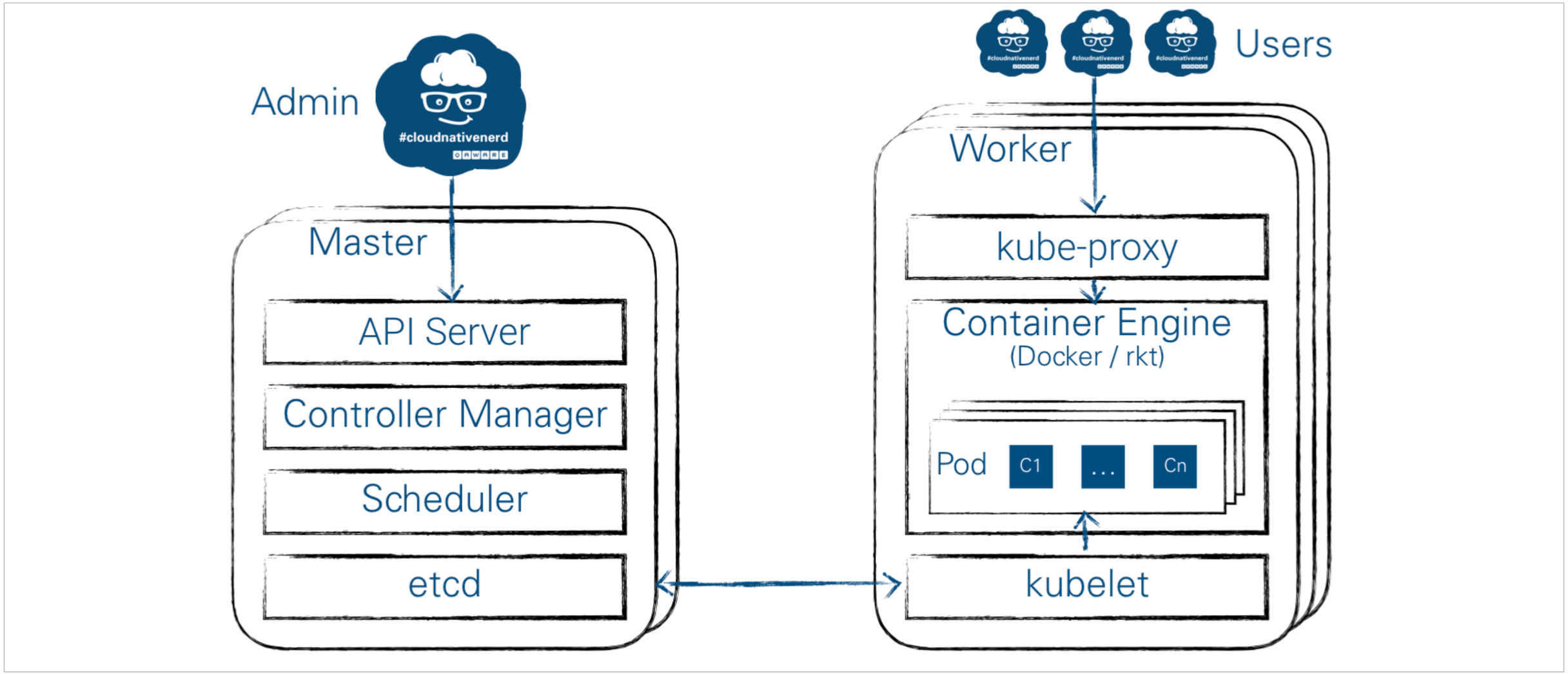


Istio is like AOP, but for microservice communication.

Istio to the Rescue!



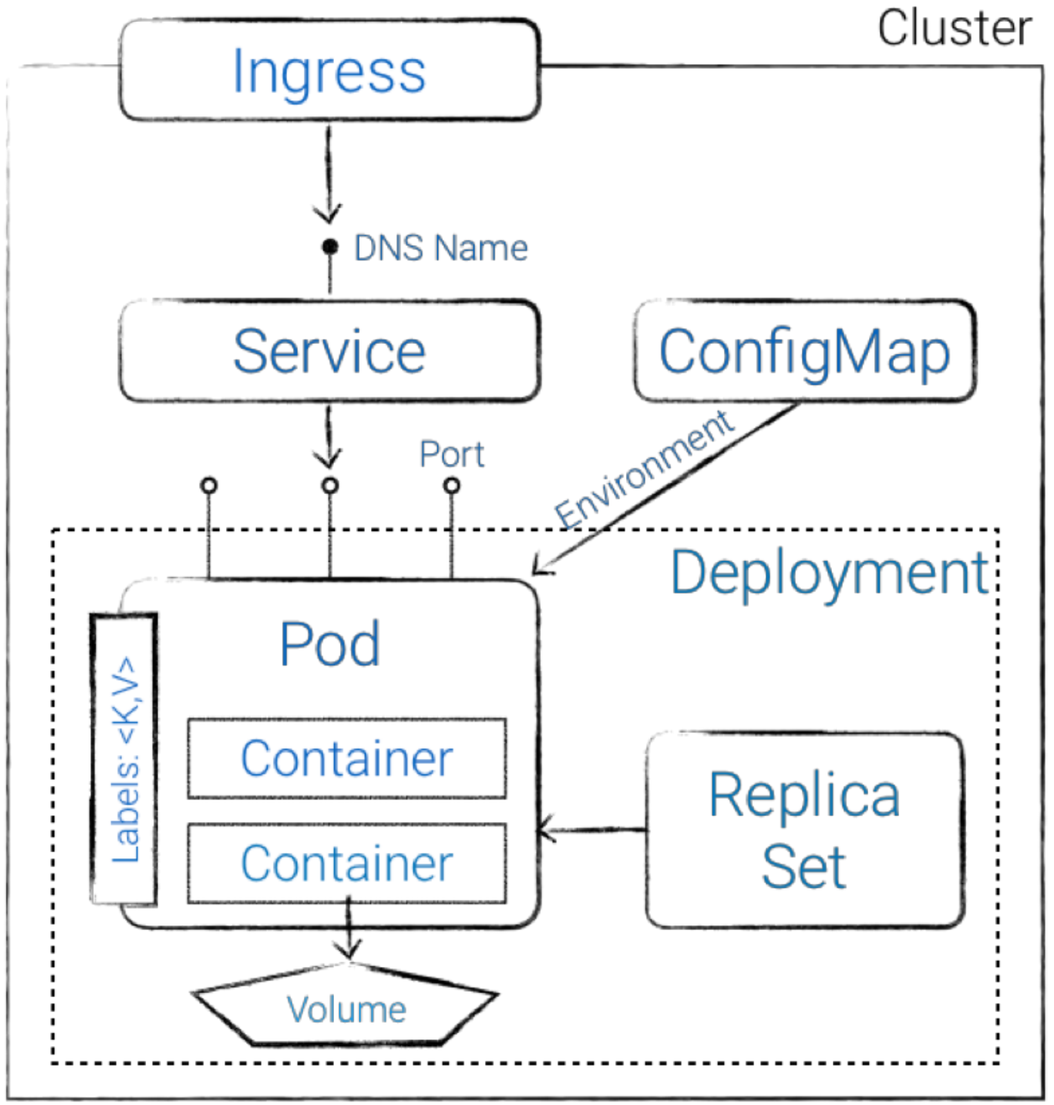
Conceptual View on a Kubernetes Cluster.



Demo

Kubernetes Glossary.

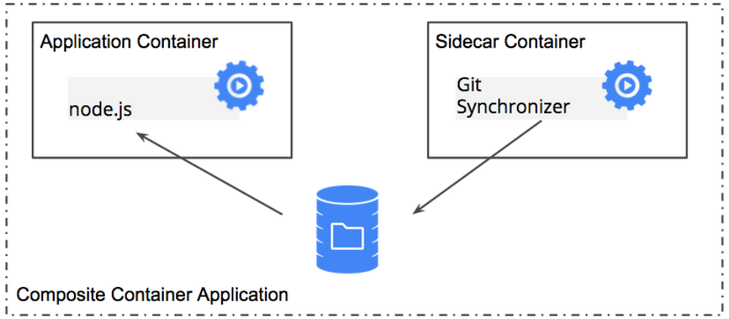
- **Pods** are the smallest unit of compute in Kubernetes
- **Labels** are key/value pairs used to identify Kubernetes resources
- **Replica Sets** ensure that the desired number of pod replicas are running
- **Deployments** are an abstraction used to declare and update pods, RCs, …
- **Services** are an abstraction for a logical collection of pods providing DNS name
- **Ingress** routes traffic from outside the cluster to services and ports based on URL patterns and host



GoF in the Cloud: Container Orchestration Patterns.

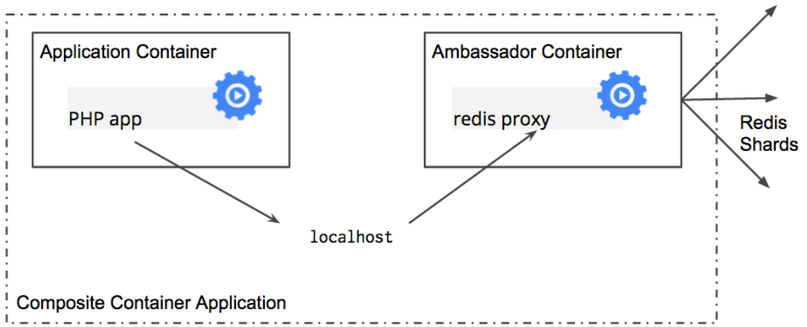
1. Sidecar Container:
Extend container behaviour

- Log Extraction / Reformating (fluentd, logstash)
- Scheduling (cron, quartz)



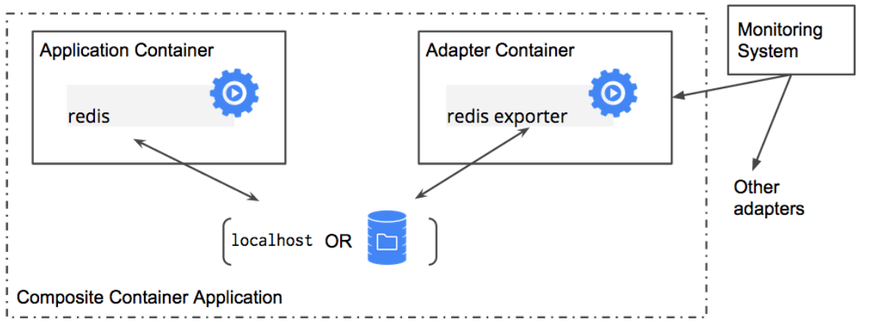
2. Ambassador Container:
Proxy communication

- TLS Tunnel (Stunnel, ghostunnel, Istio)
- Circuit Breaking (linkerd, Istio)
- Request Monitoring (linkerd, Istio)



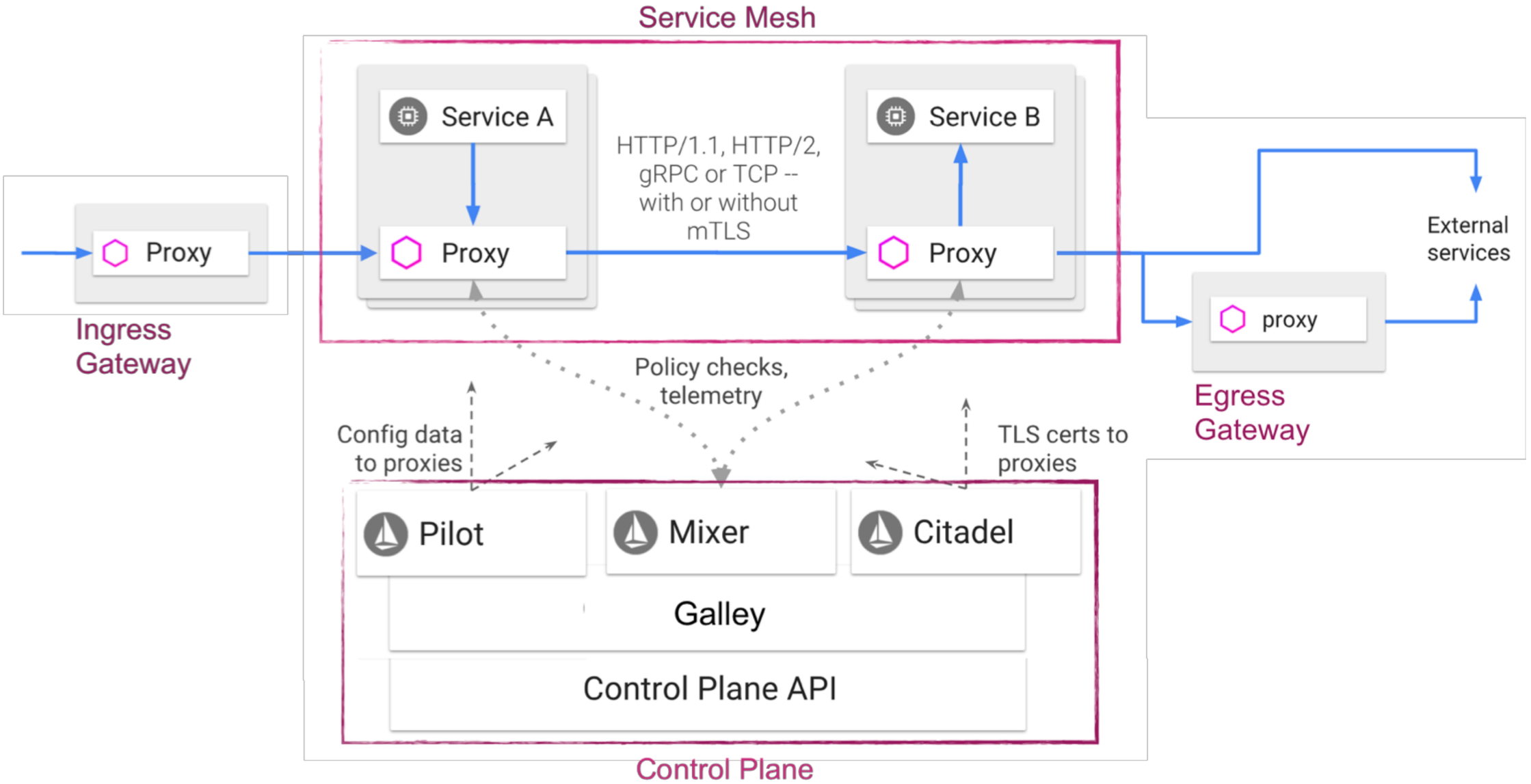
3. Adapter Container:
Provide a standardized interface

- Monitoring (Prometheus)
- Configuration (ConfigMaps, Secrets, …)



http://blog.kubernetes.io/2015/06/the-distributed-system-toolkit-patterns.html

Conceptual Istio Architecture and Components.

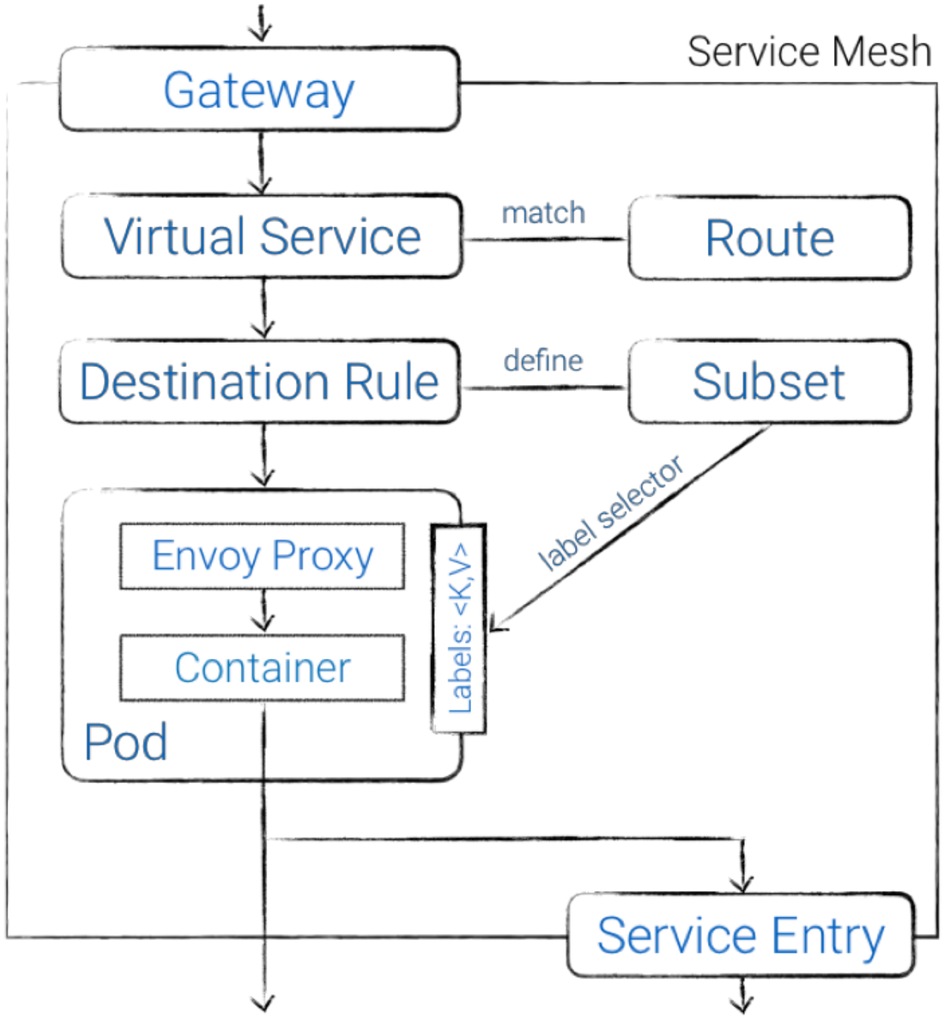


- **Envoy**: Sidecar proxy per microservice that handles inbound/outbound traffic within each Pod. Extended version of Envoy project.
- **Gateway**: Inbound gateway / ingress. Nothing more than a managed Envoy.
- **Mixer**: Policy / precondition checks and telemetry. Highly scalable.
  - Envoy caches policy checks within the sidecar (level 1) and within envoy instances (level 2), buffers telemetry data locally and centrally, and can be run in multiple instances.
  - Mixer includes a flexible plugin model.
  - https://istio.io/blog/2017/mixer-spof-myth.html
- **Pilot**: Pilot converts high level routing rules that control traffic behavior into Envoy-specific configurations, and propagates them to the sidecars at runtime.
  - Watches services and transforms this information in a canonical platform-agnostic model (abstracting away from k8s, Nomad, Consul etc).
  - The envoy configuration is then derived from this canonical model. Exposes the Rules API to add traffic management rules.
- **Citadel**: CA for service-to-service authx and encryption.
  - Certs are delivered as a secret volume mount. Workload identity is provided in SPIFFE format.
  - https://istio.io/docs/concepts/security/mutual-tls.html

Demo

Istio Glossary.

- **Gateway** configures a load balancer for HTTP/TCP traffic, enables ingress traffic into the service mesh
- **Virtual Service** defines the rules that control how requests for a service are routed within the service mesh
- **Destination Rule** configures the set of policies to be applied to a request after VirtualService routing has occurred
- **Service Version** aka **Subset** allows to select a subset of pods based on labels
- **Service Entry** enables requests to services outside of the service mesh



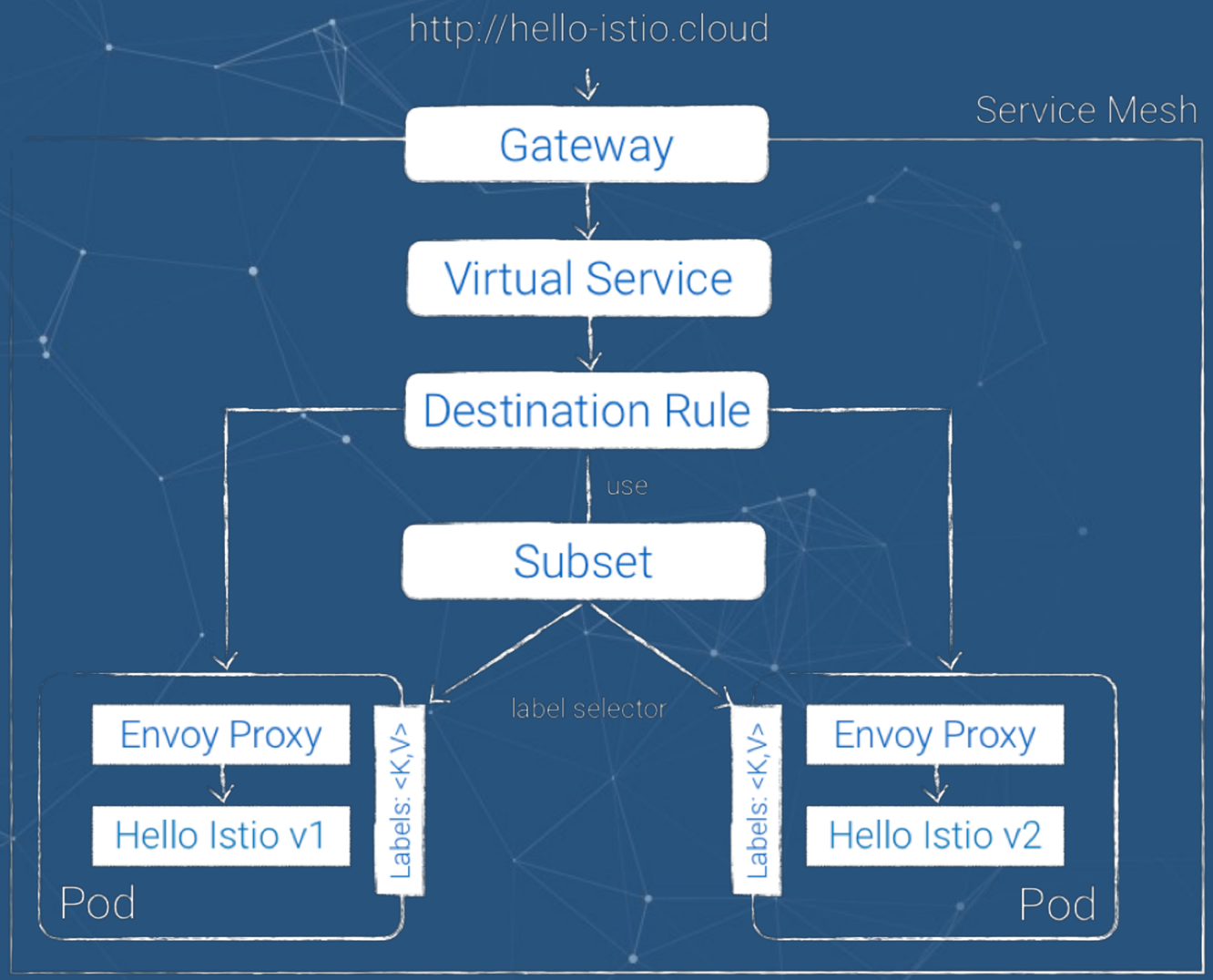
Example Istio Gateway and VirtualService Definitions.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: hello-istio-gateway
spec:
  selector:
    # use istio default controller
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "hello-istio.cloud"
```

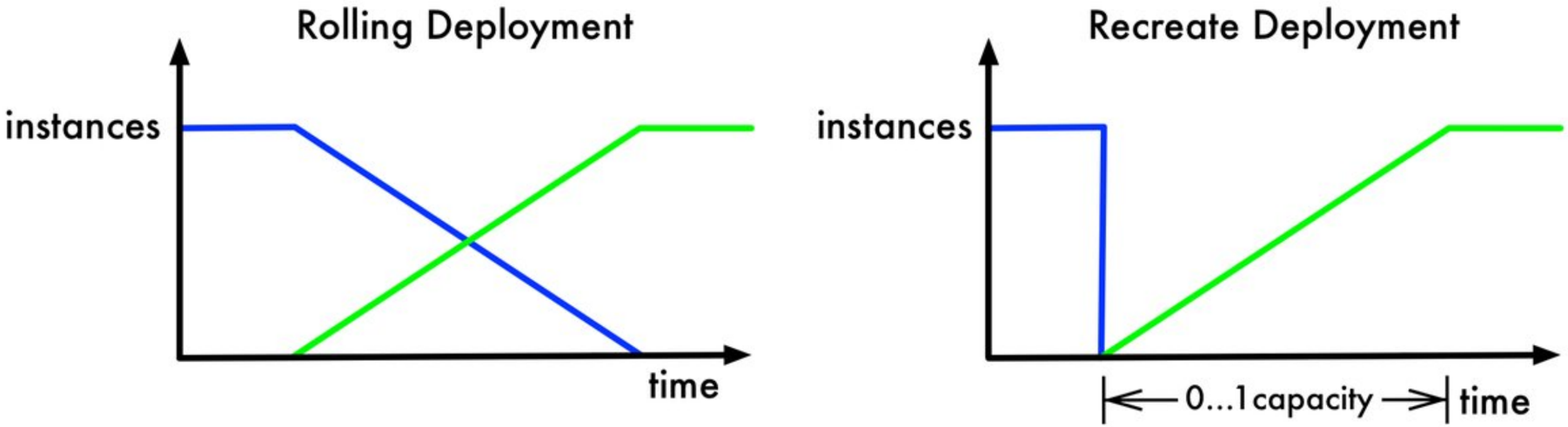
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: hello-istio
spec:
  hosts:
  - "hello-istio.cloud"
  gateways:
  - hello-istio-gateway
  http:
  - match:
    - uri:
        exact: /api/hello
    route:
    - destination:
        host: hello-istio
        port:
          number: 8080
```

Exact URI Routing

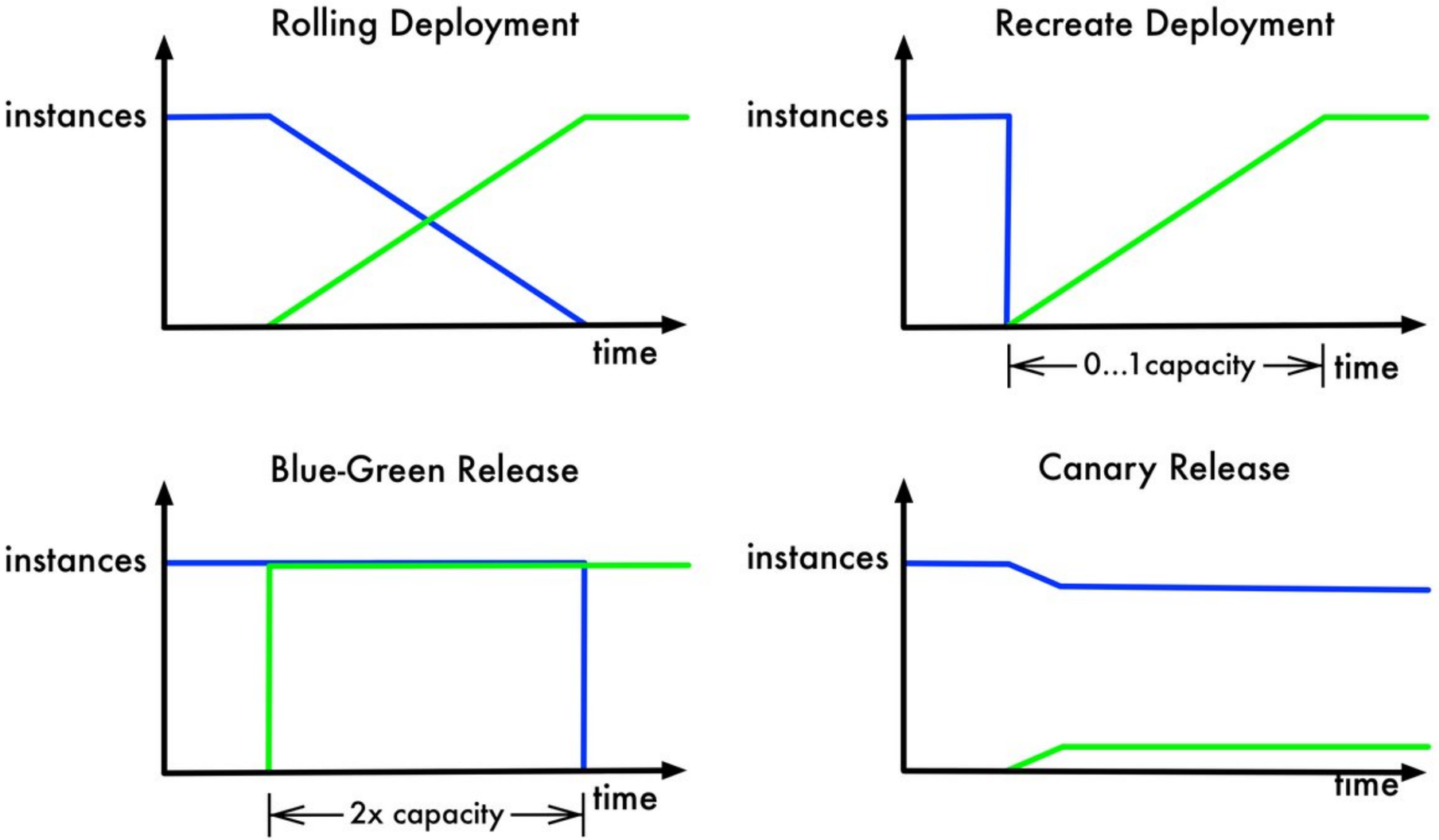
Hello Istio Demo



Different release patterns can easily be applied.



Different release patterns can easily be applied.



Examples for different routing configurations.

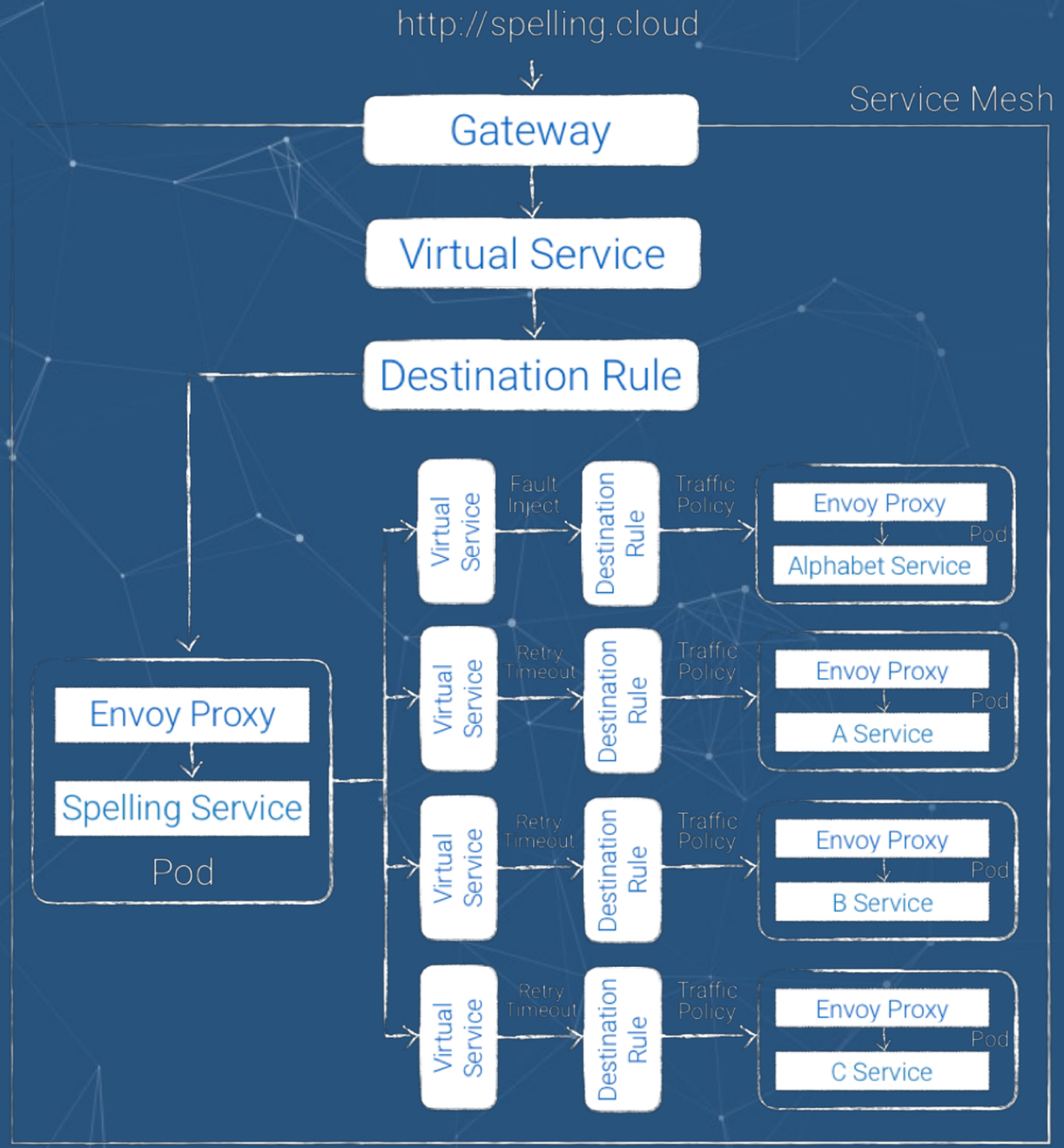
```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: hello-istio
spec:
  hosts:
  - "hello-istio.cloud"
  gateways:
  - hello-istio-gateway
  http:
  - route:
    - destination:
        host: hello-istio
        subset: v1
      weight: 70
    - destination:
        host: hello-istio
        subset: v2
      weight: 30
```

Weighted Traffic Routing

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: hello-istio
spec:
  hosts:
  - "hello-istio.cloud"
  gateways:
  - hello-istio-gateway
  http:
  - match:
    - headers:
        user-agent:
          regex: ".*Chrome.*"
    route:
      - destination:
          host: hello-istio
          subset: v2
  - route:
    - destination:
        host: hello-istio
        subset: v1
```

Header based Traffic Routing

Alphabet Demo



Examples for fault injection and circuit breaker policy.

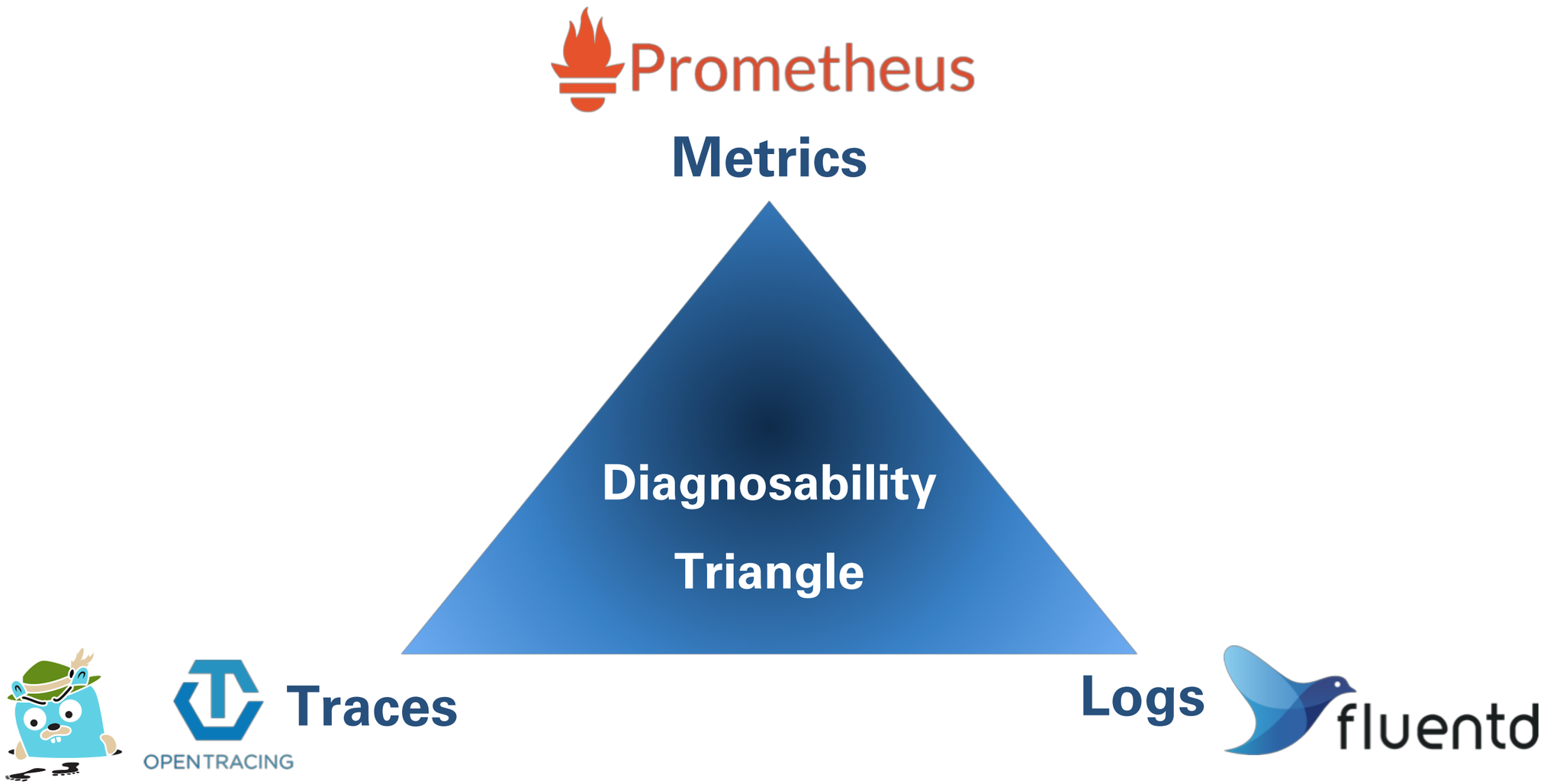
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: alphabet-service
spec:
  hosts:
  - alphabet-service
  http:
  - fault:
      delay:
        fixedDelay: 2s
        percent: 50
      abort:
        httpStatus: 500
        percent: 50
    route:
    - destination:
        host: alphabet-service
        subset: v1
```

Fault Injection

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: alphabet-service
spec:
  host: alphabet-service
  trafficPolicy:
    connectionPool:
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
      tcp:
        maxConnections: 1
    outlierDetection:
      baseEjectionTime: 5.000s
      consecutiveErrors: 1
      interval: 1.000s
      maxEjectionPercent: 100
  subsets:
  - name: v1
    labels:
      version: v1
```

Circuit Breaker Policy

Istio has built-in support for service mesh diagnosability.



Not all Istio features are marked Stable yet, but Beta can already be used in Production.

- Istio v1.0.3 is deemed production ready.
  - *Core*: 4 Stable, 3 Beta, 6 Alpha
  - *Traffic Management*: 1 Stable, 5 Beta, 1 Alpha
  - *Security and Policy Enforcement*: 5 Stable, 2 Beta, 4 Alpha
  - *Telemetry*: 4 Stable, 2 Beta, 7 Alpha
- Other Service Mesh technologies are emerging.
  - Linkerd and Conduit
  - Consul Connect

See https://istio.io/about/feature-stages/

Q&A

Mario-Leander Reimer

mario-leander.reimer@qaware.de

@LeanderReimer


twitter.com/qaware


linkedin.com/company/qaware-gmbh


xing.com/companies/qawaregmbh


github.com/qaware


slideshare.net/qaware


youtube.com/qawaregmbh