HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

Introduction to Service Meshes

Marvin Bertram, Richard Bischof, Kevin Schulze

Agenda

Chapter 1: Introduction

Cloud Computing

- 5 characteristica - 4 deployment models - 3 service models
- enabling everyone to provide high quality services

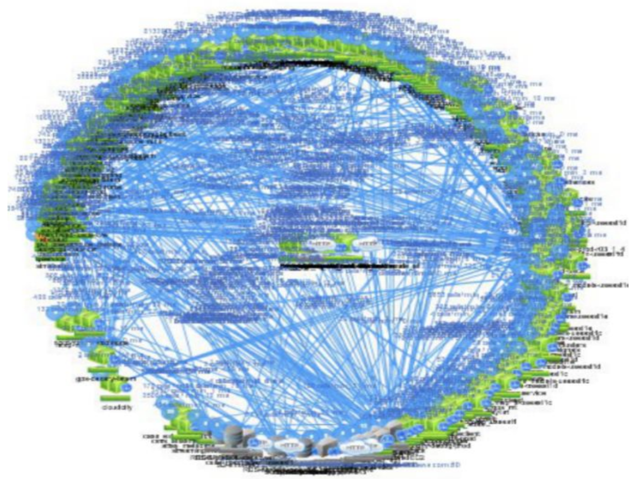
Agile Manifesto

- 4 principles of values in agile work
- shift of mind from *“never change a running system”* to *“fail fast, fail often”*

Transition from monoliths to microservices

- “big ball of mud” **[Foote,1999]**
- systems should consist of loosely coupled services
- Domain-Driven-Design

Chapter 1: Motivation
"Death-Star of Microservices"



Netflix

Twitter

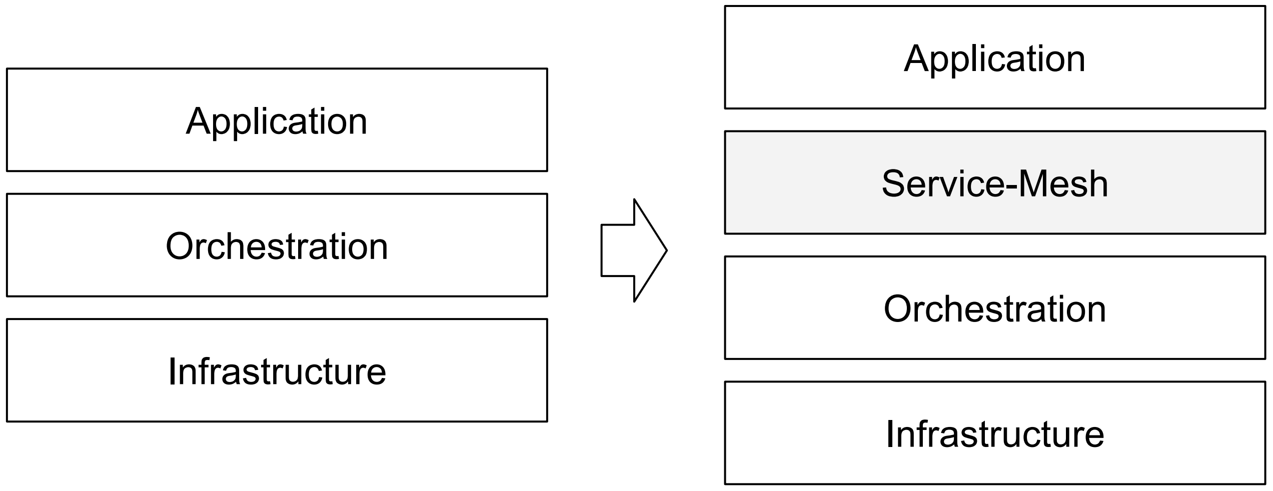
Why should the infrastructure be extracted from applications?

- collaborative development and reusability of artefacts
- save and reliable deployments
- change of infrastructure without change of application

Chapter 2: Fundamentals: Services-Meshes

Service Meshes: dedicated infrastructure component

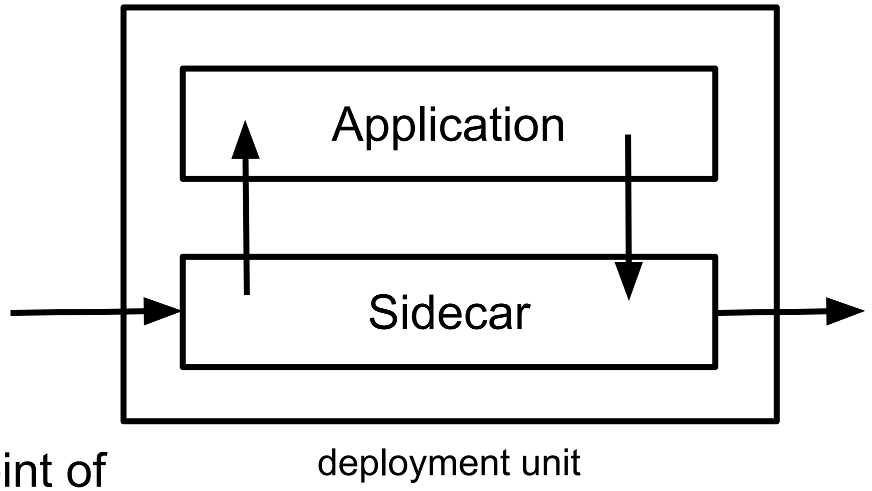
- *observing*, *controlling* and *securing* communication between applications
- earlier approaches: ESB, API Gateways
- only focuses on networking rather than business concerns
- Service Mesh Interface (SMI): standard developed by Microsoft, HashiCorp, Buoyant, Solo.io



Chapter 2: Fundamentals: Sidecar-Pattern

One Pattern of distributed systems

1. **Application**
   a. contains application
   b. focuses on domain
2. **Sidecar**
   a. extends functionality by establishing single point of communication to application



Chapter 2: Fundamentals: Kubernetes

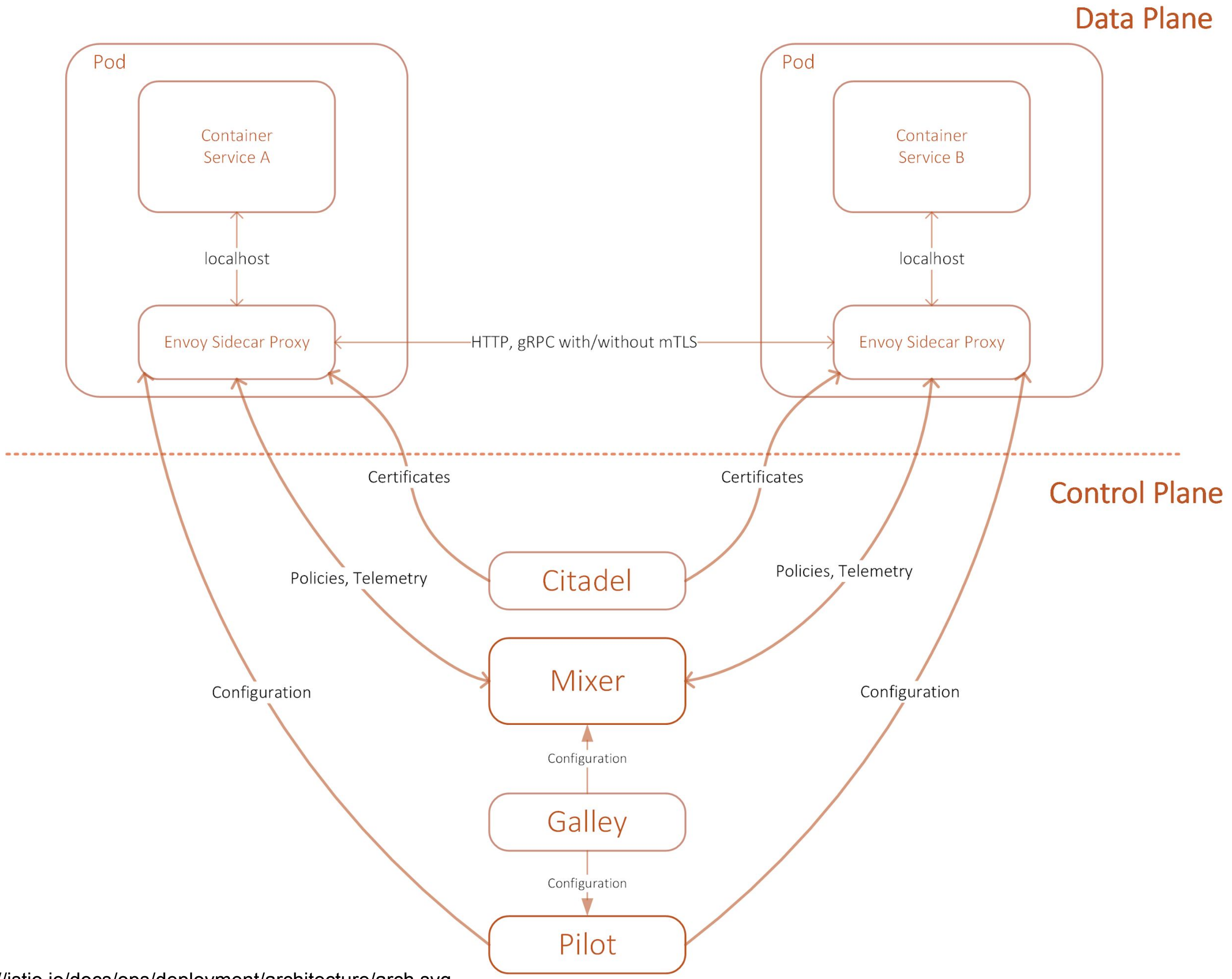
Runtime: Docker-Container

- immutable application artefacts
- containers

Orchestration: Kubernetes

- Pods
  - one or more containers
- Deployments
  - controller for pods and replica-sets
- Services
  - exposes pod-services through the cluster (DNS, NodePort, LoadBalancer)
- Custom Resource Definitions (CRD) z. B. ***VirtualService***, ***DestinationRule***

Chapter 3: Istio Architecture



After: https://istio.io/docs/ops/deployment/architecture/arch.svg

Chapter 3: Use Case: Security

[Istio, 2019]

- mTLS encryption between the services

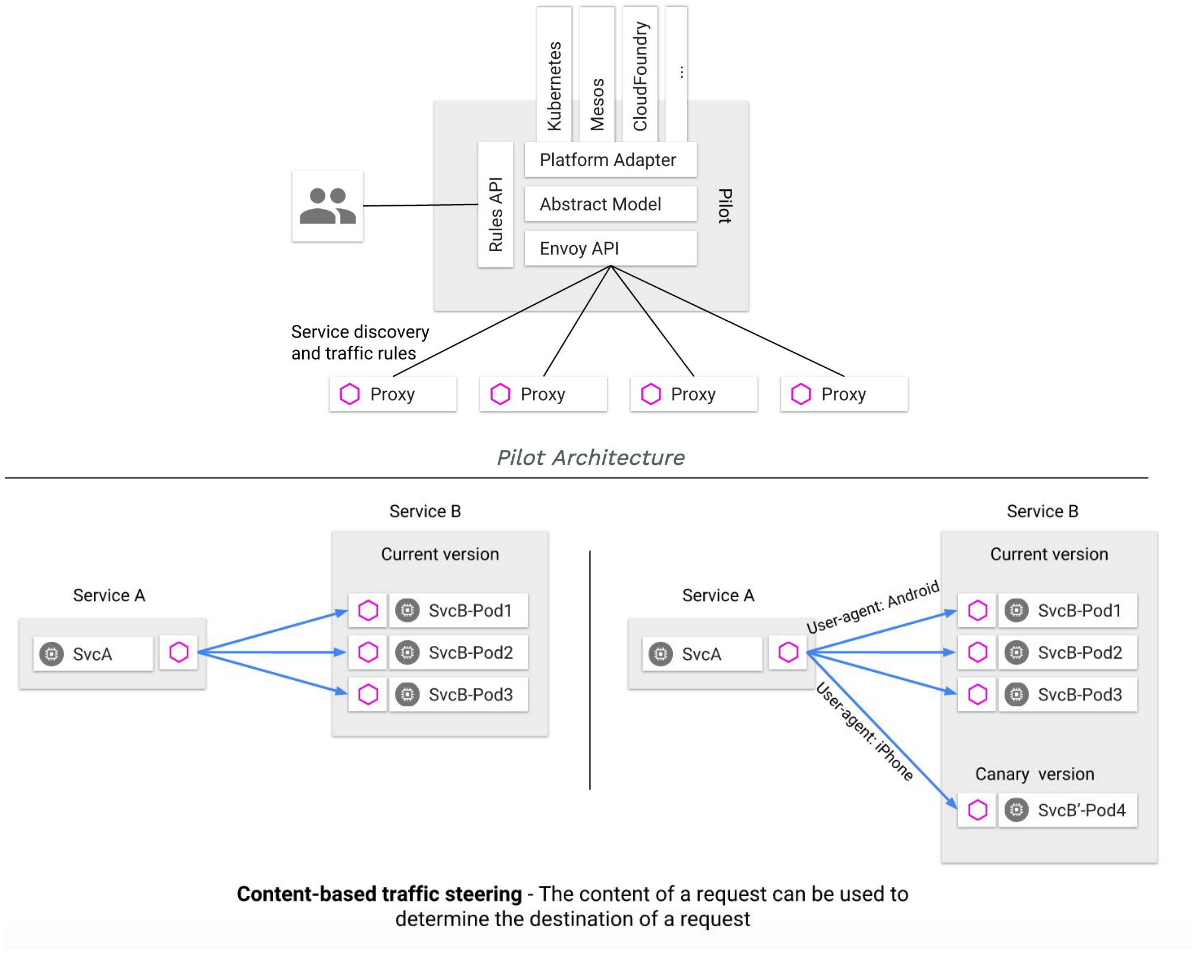
- Authentication
  - Service-to-service
  - End-user to service
    - Different identity providers

- Authorization
  - Policies can be applied to namespaces, services, mesh
  - Define invocation path: service A → service B → service C
  - Role-based access control

- Audit
  - Who called which service at what time?

Source: https://archive.istio.io/v1.0/docs/concepts/security/

Chapter 3: Use Case: Traffic Management



Content-based traffic steering - The content of a request can be used to determine the destination of a request

Source: https://archive.istio.io/v1.0/docs/concepts/traffic-management/

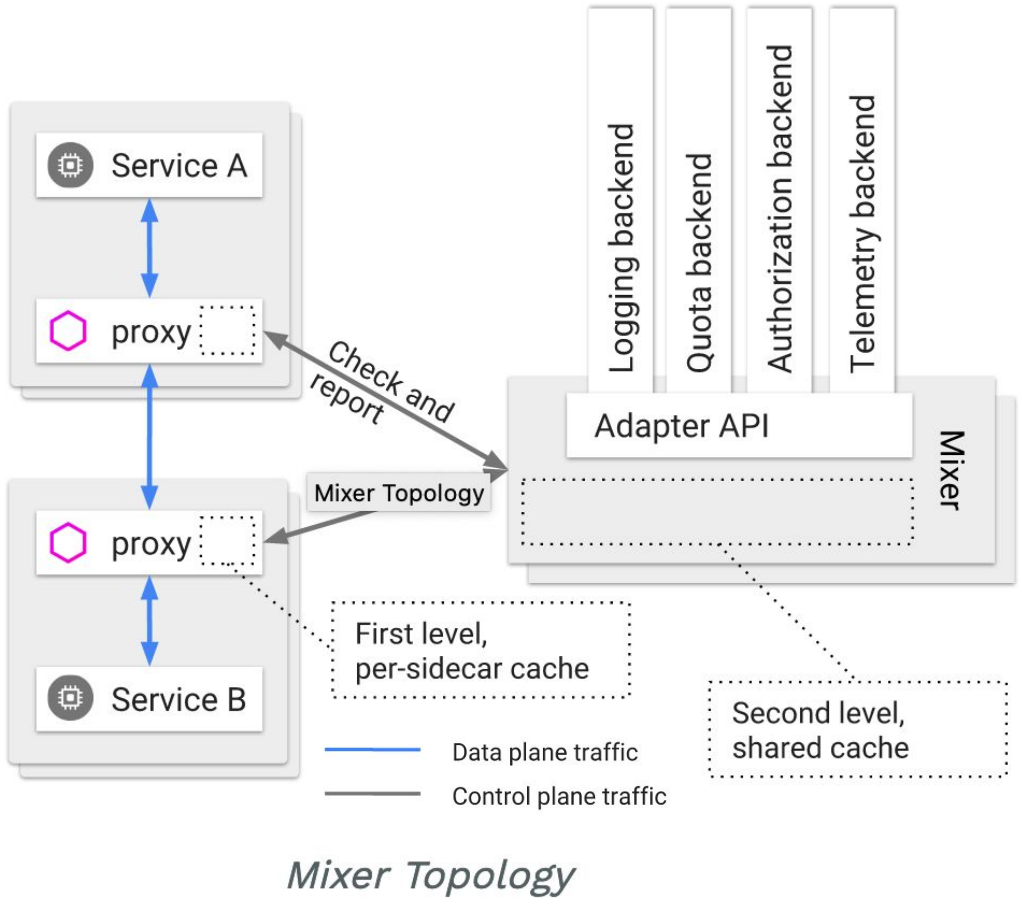
- Traffic rules specified via Pilot
- Traffic management features:
  - A/B testing
  - Gradual rollouts

- handles failure recovery using:
  - timeouts
  - retries
  - circuit breakers
  - fault injection

Realized through the sidecars deployed across the service mesh

Chapter 3: Use Case: Policies & Telemetry

[Istio, 2019]



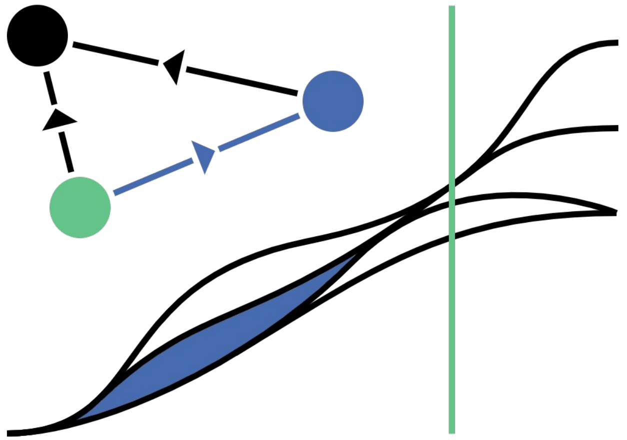
Source: https://archive.istio.io/v1.0/docs/concepts/policies-and-telemetry/

- **Mixer** is responsible for providing **policy controls** and **telemetry collection**
- The sidecar calls Mixer before each request to perform precondition checks, and after each request to report telemetry
- Sidecar has local caching and buffers telemetry

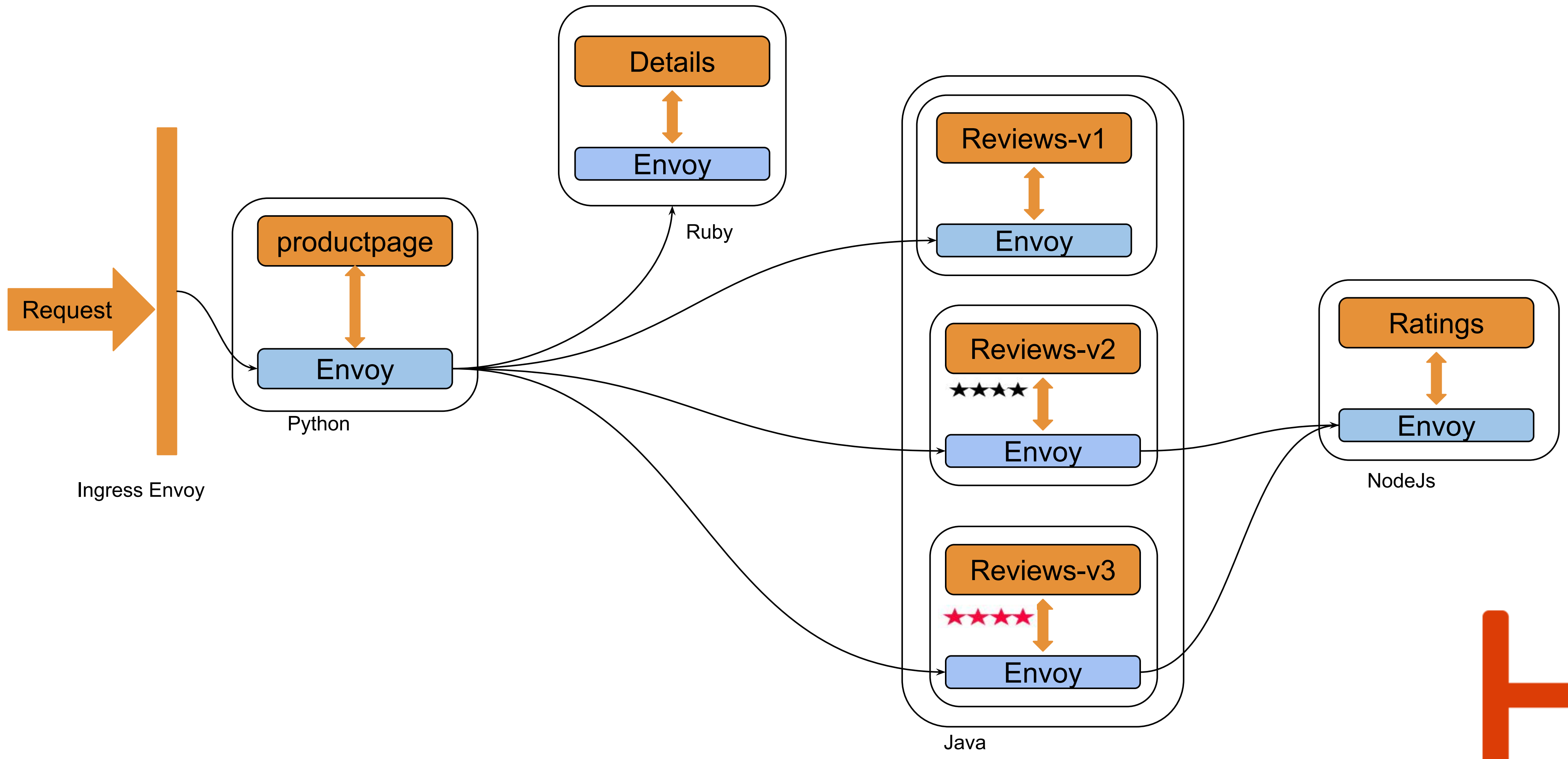
Chapter 3: Use Case: Observability

- Tracing
  - Determine which microservices are used in a request and their dependencies
  - Flow through the mesh
  - Different tracing backends supported:
    - Jaeger, LightStep, Zipkin
    - Default is Envoy
- Metrics
  - Inbound, outbound and inside the mesh
  - Error rates, response times, traffic volume, ...
  - Proxy-level, service-level, control-plane metrics

- Service Graph
  - Visualizing and monitoring of the service mesh


[Istio, 2019]

Chapter 4: Istio: Bookinfo-Application



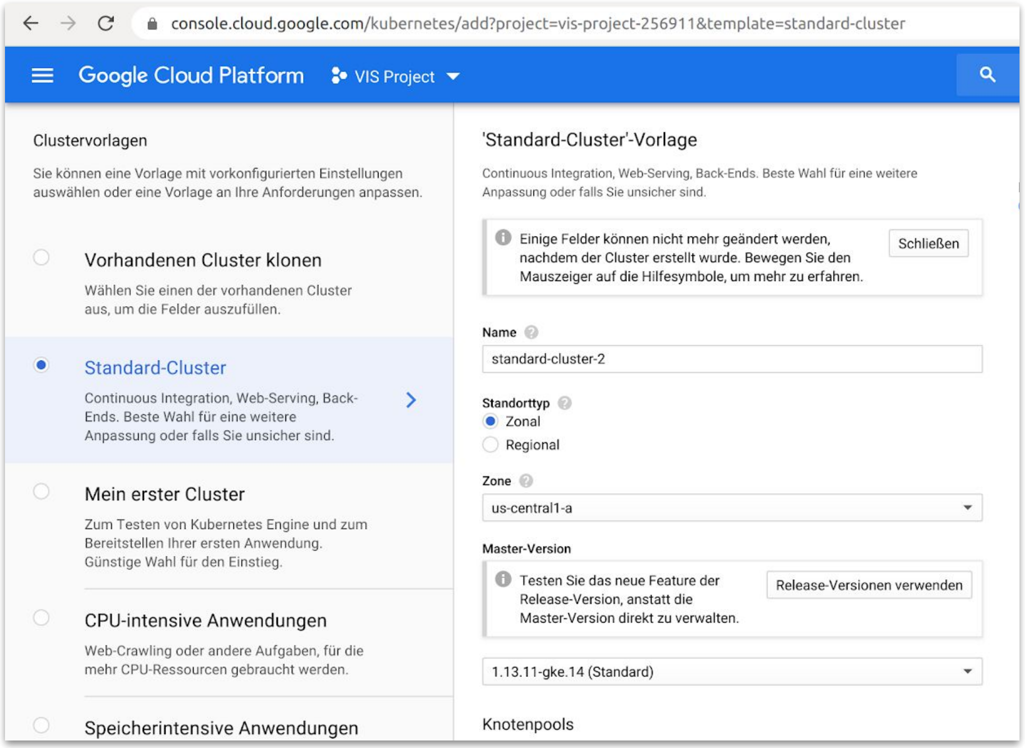
Source: https://istio.io/docs/examples/bookinfo/noistio.svg

Chapter 4: Istio: Bookinfo-Application Deployment

Google Cloud

- Google Kubernetes Engine
- Kubernetes Version 1.13.11
- 4 Node Cluster
- Automatic Istio Integration

```
$ kubectl get nodes
$ kubectl get namespaces
$ kubectl get all -n istio-system
$ kubectl get all -n default
```


console.cloud.google.com/kubernetes/add?project=vis-project-256911&template=standard-cluster

Google Cloud Platform — VIS Project

Clustervorlagen

Sie können eine Vorlage mit vorkonfigurierten Einstellungen auswählen oder eine Vorlage an Ihre Anforderungen anpassen.

- Vorhandenen Cluster klonen — Wählen Sie einen der vorhandenen Cluster aus, um die Felder auszufüllen.
- [✓ Standard-Cluster] — Continuous Integration, Web-Serving, Back-Ends. Beste Wahl für eine weitere Anpassung oder falls Sie unsicher sind.
- Mein erster Cluster — Zum Testen von Kubernetes Engine und zum Bereitstellen Ihrer ersten Anwendung. Günstige Wahl für den Einstieg.
- CPU-intensive Anwendungen — Web-Crawling oder andere Aufgaben, für die mehr CPU-Ressourcen gebraucht werden.
- Speicherintensive Anwendungen

'Standard-Cluster'-Vorlage

Continuous Integration, Web-Serving, Back-Ends. Beste Wahl für eine weitere Anpassung oder falls Sie unsicher sind.

Einige Felder können nicht mehr geändert werden, nachdem der Cluster erstellt wurde. Bewegen Sie den Mauszeiger auf die Hilfesymbole, um mehr zu erfahren. [Schließen]

Name: standard-cluster-2

Standorttyp: [✓ Zonal] Regional

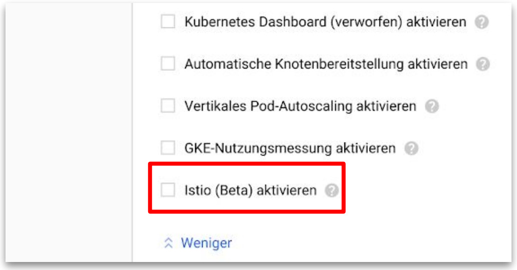
Zone: us-central1-a

Master-Version

Testen Sie das neue Feature der Release-Version, anstatt die Master-Version direkt zu verwalten. [Release-Versionen verwenden]

1.13.11-gke.14 (Standard)

Knotenpools


- ☐ Kubernetes Dashboard (verworfen) aktivieren
- ☐ Automatische Knotenbereitstellung aktivieren
- ☐ Vertikales Pod-Autoscaling aktivieren
- ☐ GKE-Nutzungsmessung aktivieren
- ☐ Istio (Beta) aktivieren

Weniger

Chapter 4: Istio: Bookinfo-Application Observability with Kiali Demo

- Topology visualization of the mesh
- Different views of metrics and services
- Monitoring of the service mesh
- Limited configuration of Istio
- Validation
- Jaeger[1] tracing included

```
$ istioctl dashboard kiali
```

[1] https://www.jaegertracing.io/docs/1.9/

Chapter 4: Istio: Bookinfo-Application Demo

http://34.69.155.52/productpage

Chapter 4: Istio: Bookinfo-Application Traffic Routing Demo

- Route traffic from iOS users to reviews v1 page
- Route traffic from Android users to reviews v2 page
- Route traffic from Windows users to reviews v3 page
- All other traffic goes to version 1 of the reviews page

- Destination Rule
  - Subsets
- Virtual Service
  - Traffic Routing

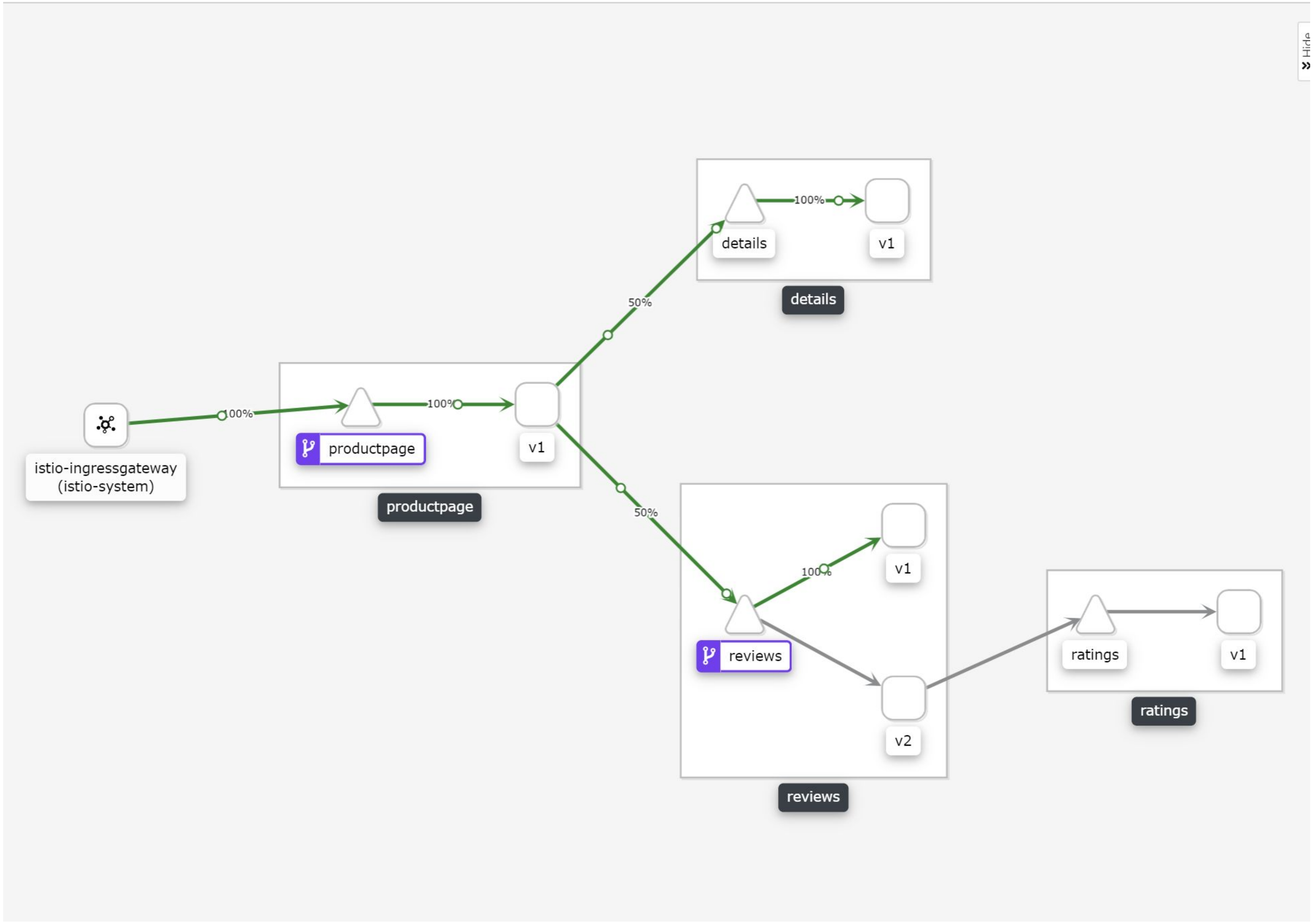
Chapter 4: Istio: Bookinfo-Application Traffic Routing Demo

Destination Rule

```
kind: DestinationRule
metadata:
 name: dr-reviews
spec:
 host: reviews
 subsets:
 - labels:
     version: v1
   name: version-v1
 - labels:
     version: v2
   name: version-v2
 - labels:
     version: v3
   name: version-v3
```

Virtual Service

```
kind: VirtualService
metadata:
 name: reviews-virtualservice
spec:
 hosts:
 - reviews
 http:
 - match:
   - headers:
       user-agent:
         regex: .*iPhone.*
   route:
   - destination:
       host: reviews
       subset: version-v1
 - match:
   - headers:
       user-agent:
         regex: .*Android.*
```


Hide
istio-ingressgateway (istio-system)
100%
productpage
100%
v1
productpage
50%
details
100%
v1
details
50%
reviews
100%
v1
v2
reviews
ratings
v1
ratings

Chapter 5: Conclusion

Pros	Cons
• simplifies application architecture	• higher overall complexity
• decoupling application from infrastructure code	• Adding extra hop for communication
• development focuses on domain weather deployment environment	• useful only in high automated environments
• consistency across all microservices	

→ The concept of services meshes can be beneficial for microservice architectures

→ Nevertheless requirements and use cases should also taken into account

Thank you for your attention!

References

[Burns, 2018] Brendan Burns: “Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services” O’Reilly 2018

[Wolff, 2018] https://www.heise.de/developer/artikel/Microservices-Oder-lieber-Monolithen-3944829.html

[Intro to Istio] Sutter, B. and Posta, C., “Introducing Istio Service Mesh for Microservices”, O'Reilly Media 2019

[Foote et. al. 1999] Big Ball of Mud, Department of Computer Science, University of Illinois

[Istio, 2019] https://istio.io/

[Kiali] https://kiali.io/

[Cockroft,2014]
https://gotocon.com/dl/goto-berlin-2014/slides/AdrianCockcroft_MigratingToCloudNativeWithMicroservices.pdf