SERVICE MESH ❤️ LEGACY

Benefits of a service mesh when integrating Kubernetes with legacy services

Dr. David Meder-Marouelli, Stephan Fudeus

1&1

Stephan Fudeus

- Expert for Container Platforms
- 16 years of distributed systems
- Since 2017 with 1&1 Mail & Media

Dr. David Meder-Marouelli

- Architect for Automation
- 15 years of IT experience
- Since 2015 with 1&1 Mail & Media

United Internet / 1&1 Mail & Media

United Internet

- Leading European internet specialist
- > 9000 employees
- > 90.000 servers in 10 data centers worldwide
- Access and Applications

1&1 Mail & Media

- Several free basic services and professional fee-based e-mail solutions
- One of the most powerful online marketing platforms
- > 33 million active users / month
- Multiple brands

Product Overview

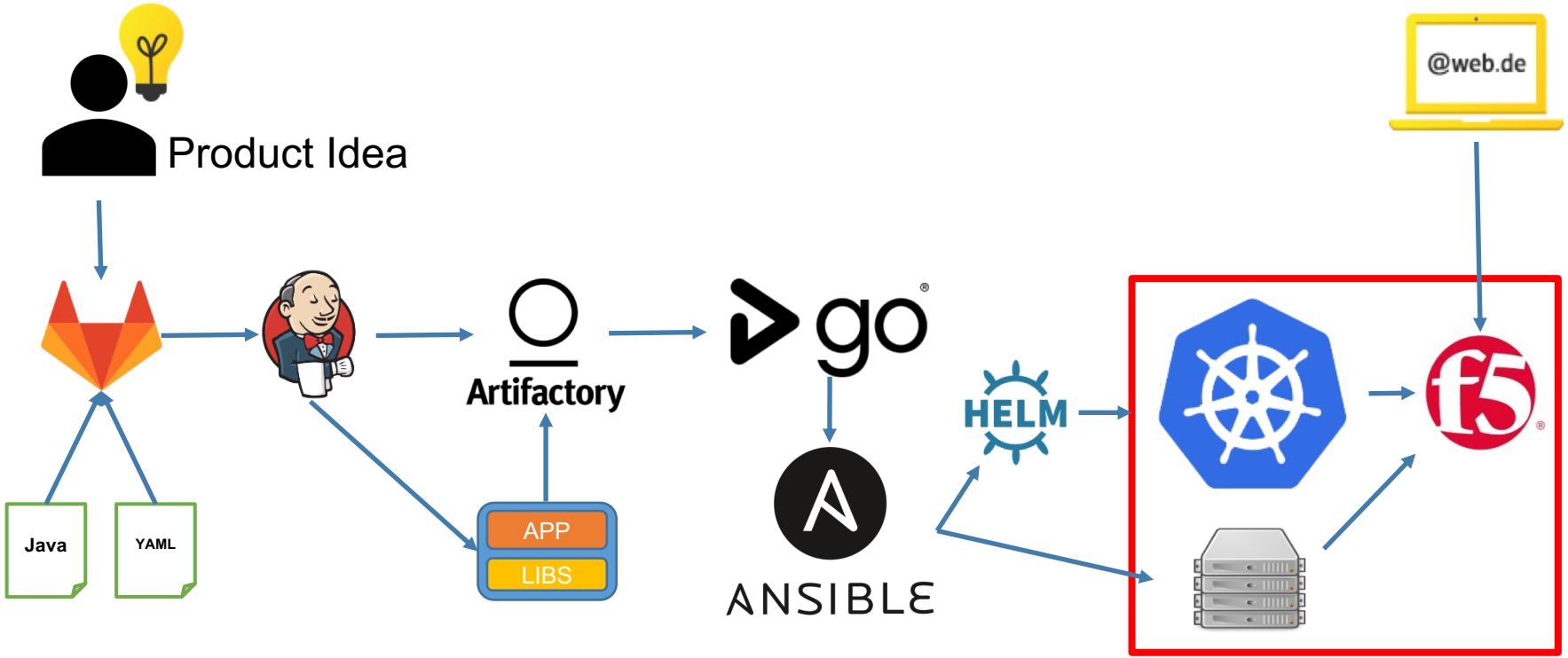
Communication and Organisation
E-Mail, Calendar, Contacts, SMS, Fax

De-Mail
Legally compliant communication and identity management

Online Office
documents, spreadsheets, presentations

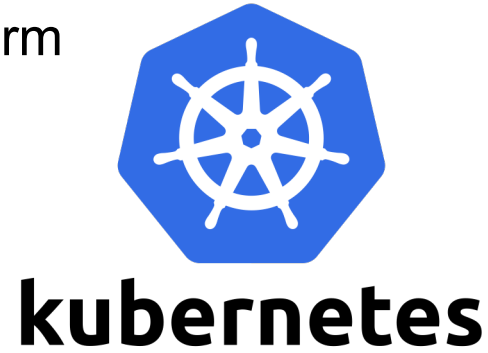
Cloud-Storage
for photos, videos, music and documents

Ideal Software Lifecycle



Container Strategy

- Kubernetes as centrally provided orchestration platform
  - Fast deployment cycles
  - Focus on soft multi-tenancy
    - Friendly users, but with security in mind
  - Focus on microservices



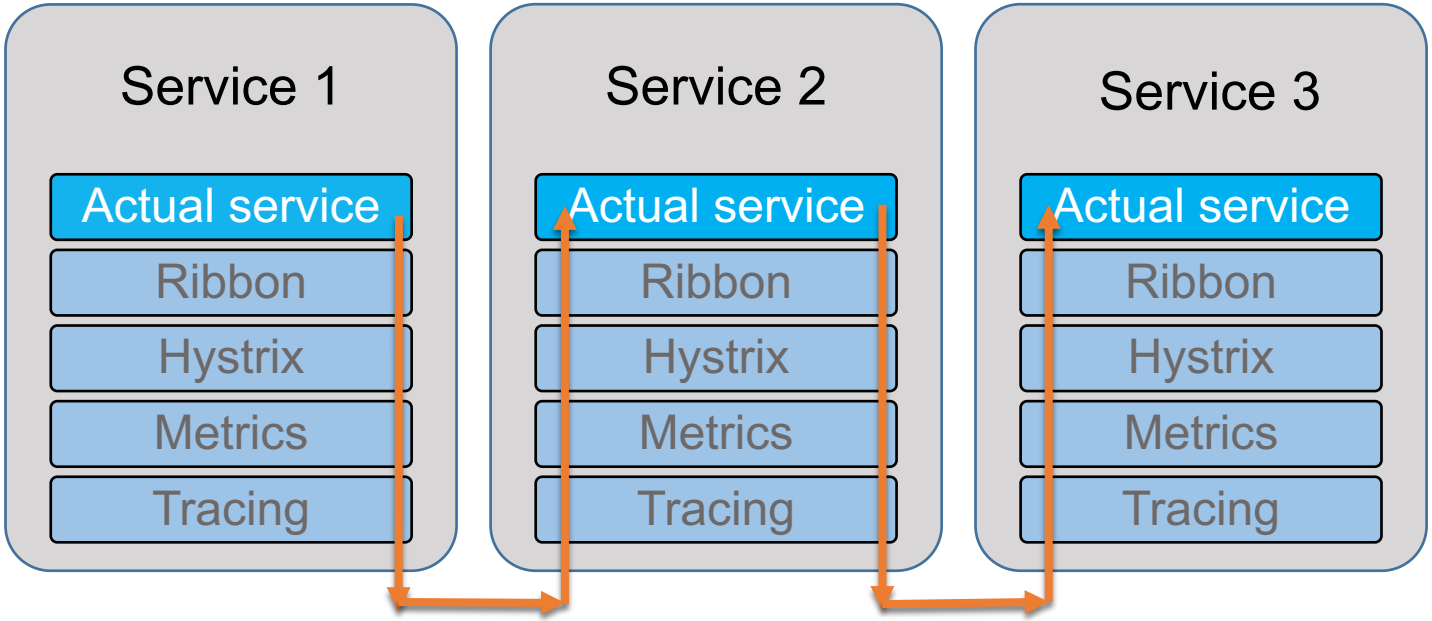
- Multiple clusters decoupled on network dimensions
  - fe/be/infrastructure, data center, live/non-live
  - bare-metal on-premise
  - non-routable podCIDR and serviceCIDR (RFC 6598 / CGNAT / 100.64.0.0/10)

Legacy Gap

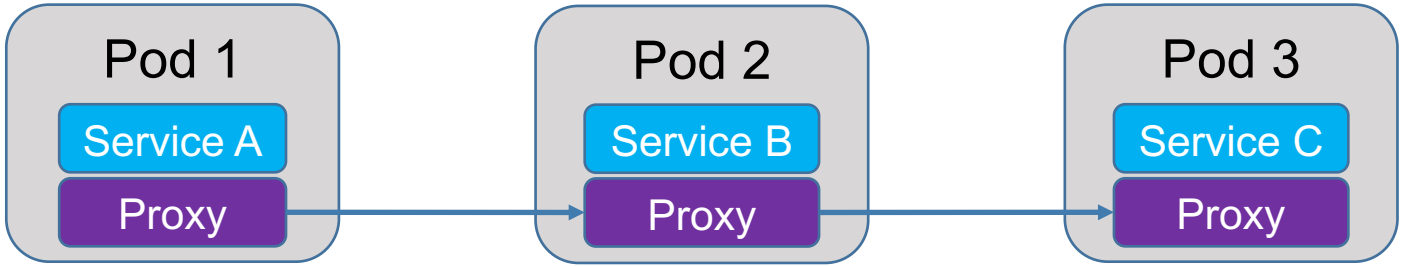
- Ca. 1000 existing services
  - 90% on VMs and bare metal
  - 10% on Kubernetes
- Complex dependencies
- Not cloud (native) ready
  - Not stateless
  - IP based ACLs
- … some never can/will be
- Both worlds need to interact

Typical Application Stack

- Mostly Java SpringBoot
- Several PHP
- Few Node.js SPA



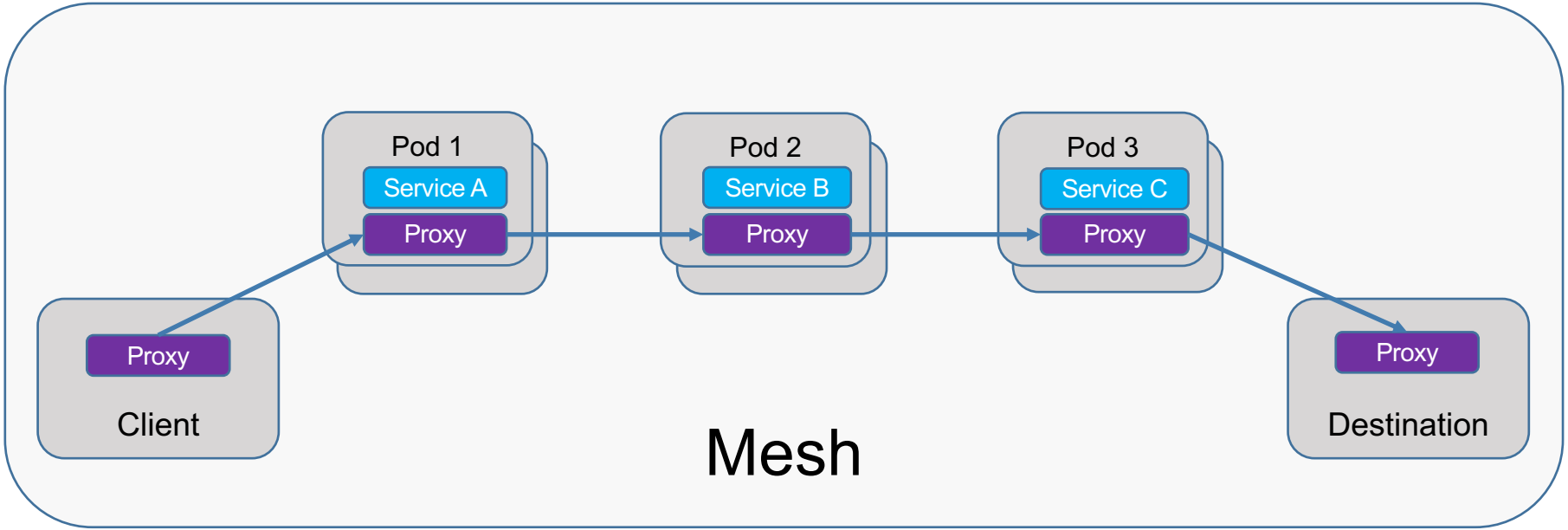
Cross cutting concerns @ infrastructure level



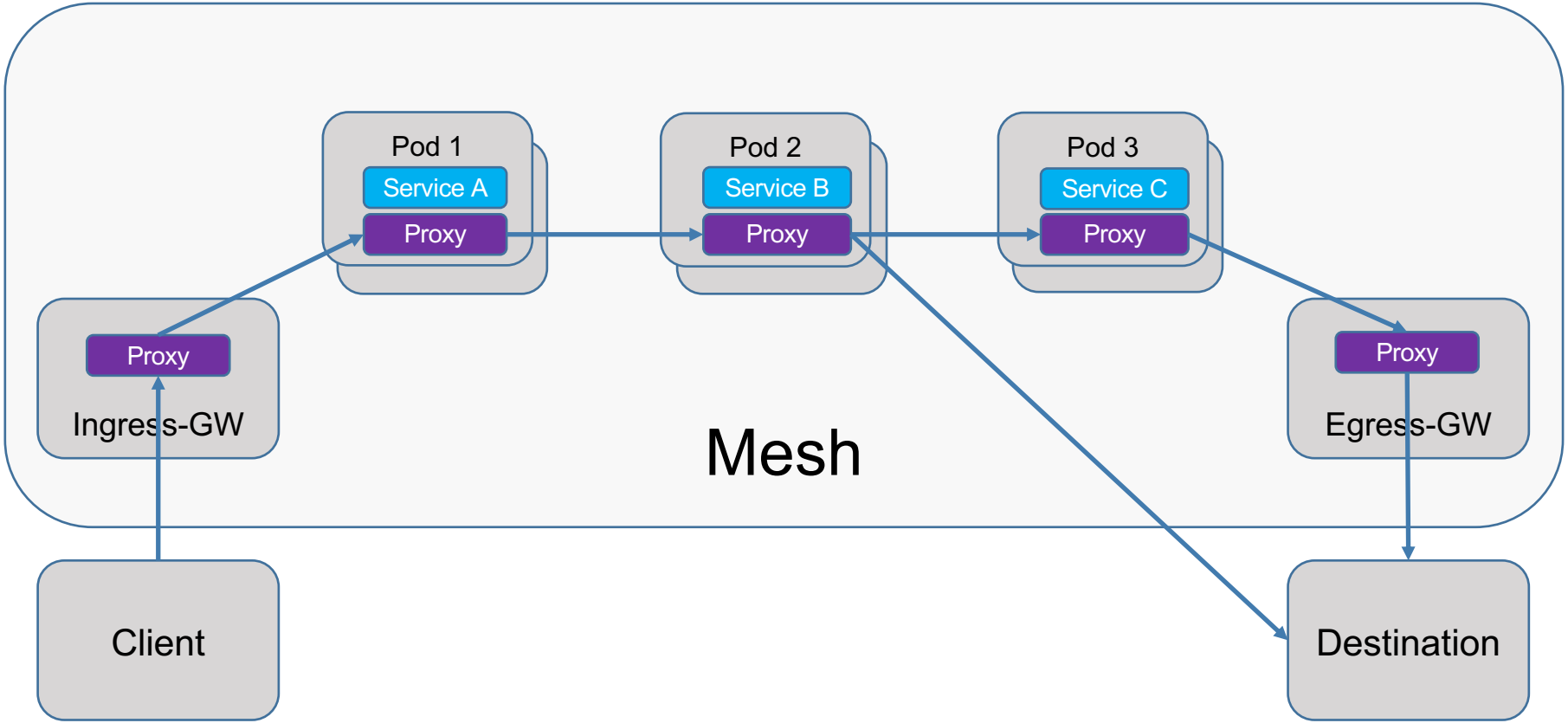
Benefits of using a Service Mesh

- Deal with the service mess of microservices
- Externalize required standard functionality
  - Telemetry
  - Request routing
  - Circuit breaking
  - Rate limiting
  - TLS/Authn/Authz
  - …
- Declaratively configured
- Centrally maintained
- Language agnostic

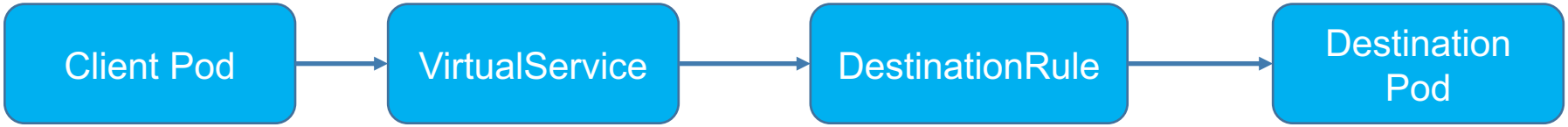
General connectivity – in-mesh



General connectivity – out-of-mesh



Configuration Objects




```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 75
    - destination:
        host: reviews
        subset: v2
      weight: 25
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  trafficPolicy:
    loadBalancer:
      simple: RANDOM
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
    trafficPolicy:
      loadBalancer:
        simple: ROUND_ROBIN
  - name: v3
    labels:
      version: v3
```

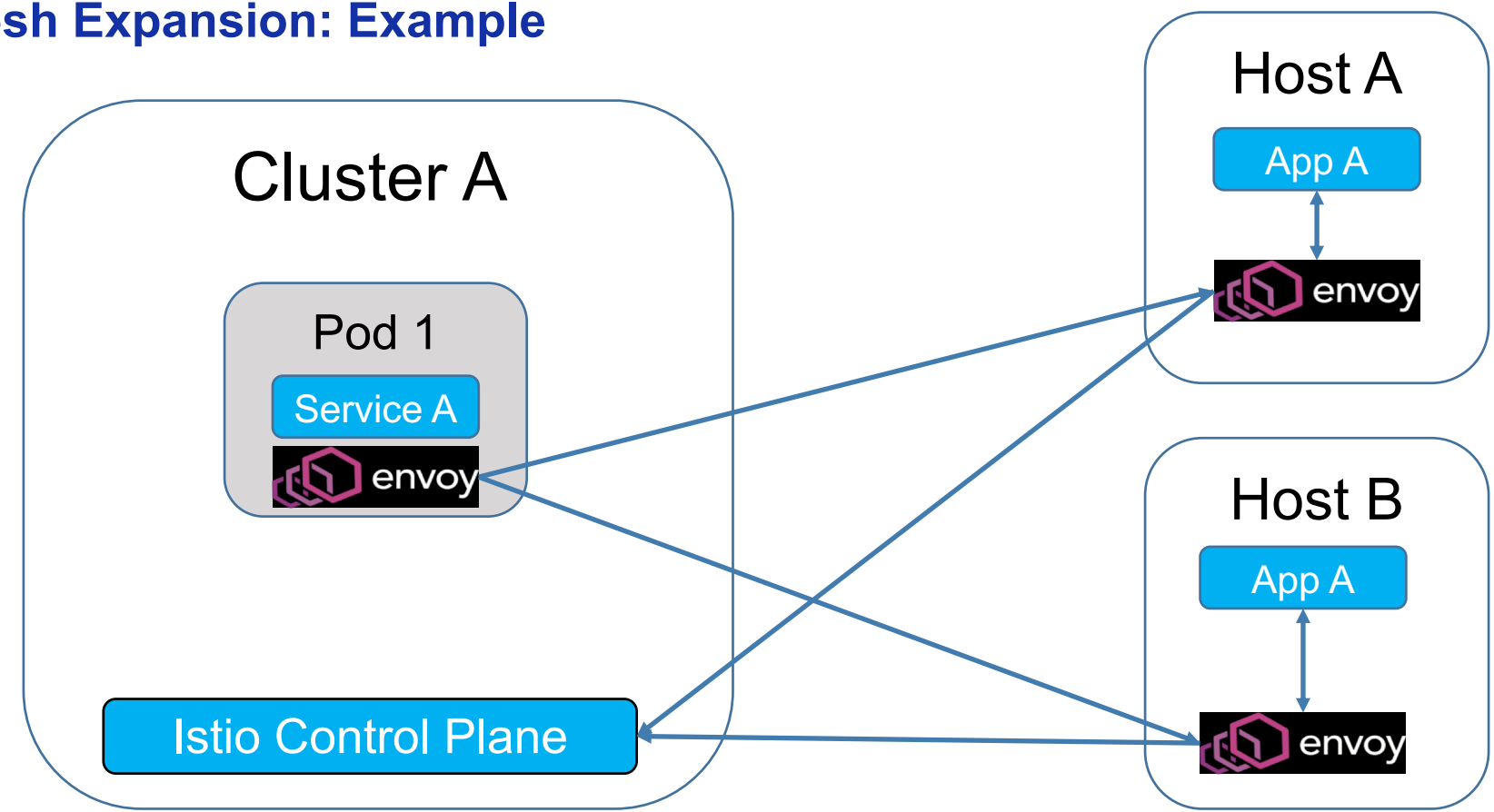
Connecting to the „Old World“

- „Mesh expansion“ (based on Istio 1.0.x)
- Install Envoy on hosts and connect to Istio control plane

- mTLS: Universal transport encryption and authentication
  - Uniformly configured with Kubernetes resources

- Central observability by Istio telemetry

Mesh Expansion: Example



Problems with Operating Istio

- Security concerns
  - High privileges for control plane components
    - run as root
    - writable root filesystem
  - High privileges for admins and serviceaccount
    - net_admin capabilities
    - run as root
  - Same problem with bookinfo sample application
- Sidecar injection
  - Problematic order of automatic sidecar injection vs. PSP evaluation

Problems with Mesh Expansion

- Connectivity to control plane
  - Manual DNS tweaks via dnsmasq
  - Exposure of K8s-DNS required
- Telemetry / Policy
  - Mixer connections only to pod IP address
  - Additional tweaks to iptables required for PoC
- Inbound (in-mesh) calls
  - Would proxy to pod IP addresses, which are not routable
- mTLS setup

Result for Istio 1.0

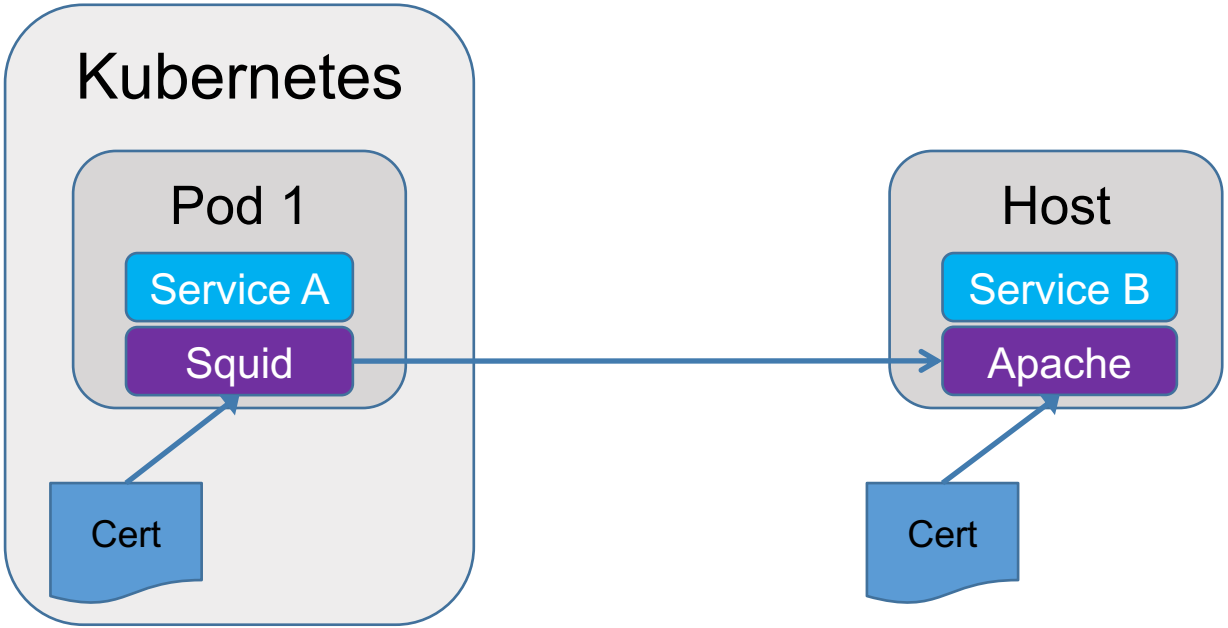
- Not suitable for production in our setup
  - Too many unstable tweaks necessary
  - Too much interference on expansion nodes

- Blocked migration of services to Kubernetes
  - Need for intermediate short term solution

Istio v1.0

„Service Mesh Light“

- Mimic mTLS with custom proxy setup and static certificates
- Automatic sidecar injection as part of deployment pipeline



Istio 1.1 to the Rescue?

- Control plane connection ✓
- Outbound expansion ✓
- mTLS setup ✓
- Istio-CNI / Security concerns ✓

But:

- Inbound expansion ✗
- Automatic sidecar injection ✗
- Documentation complex
- Documentation partially inconsistent
- Multi-tenancy unclear

What‘s up next?

- Solving the remaining issues
  - Inbound mesh expansion
  - Large scale evaluation
  - Production readiness
    - Redundancy
    - Permissions
    - Scalability
  - Multi-tenancy questions
    - Responsibilities
  - Performance and overhead
    - Costs
    - Runtime Overhead

What‘s up next?

- More benefits to be leveraged
  - Multi-cluster connectivity
  - SPIFFE interoperability
  - Locality based routing
  - Error scenarios
  - Authorization (request-based with JWT)
- Evaluate alternatives (e.g. Linkerd2)
- Provide upstream patches

Summary

- Istio (>= 1.1) fits our needs
  - Feature set and K8s integration look fine
  - Not yet fully covering our requirements

- Hard to set up the right way
- Organisational questions need to be solved

- Direction of development is promising


GER 72
GMX
WEB.DE
1&1
UNITED INTERNET TEAM GERMANY
1&1
GMX