BY COMMUNITY • FOR COMMUNITY •

#GLOBALAZURE

2021

Latinoamérica

#GlobalAzureLatam #GlobalAzure

TV
#somosluloTV

Fundación AVANET

PointColl Community

CLOUD EXPERTS

MICROSOFT USER GROUP PERÚ
Profesionales en Tecnologías Microsoft

COMUNIDAD ai

Comunidad Xamarin en Español

Latino .NET Online

BY COMMUNITY • FOR COMMUNITY •
#GLOBALAZURE
2021
Latinoamérica

ASP.NET MVC
En Español

Student Ambassadors La Paz

COMUNIDAD SECRETOS BLAZOR

Comunidad Puebla.NET

CONO SUR TECH
Comunidad de Tecnología

Aprendiendo Azure

COMUNIDAD SECRETOS XAMARIN

#GlobalAzureLatam #GlobalAzure

Latino .NET Online

JET BRAINS

sessionize

StreamYard

ENOVEL® 10 años

Jala-soft®

Microsoft

Microsoft® Press

clarika

SOFTWARE + INNOVATION

BY COMMUNITY • FOR COMMUNITY • #GLOBALAZURE 2021 Latinoamérica

EL CAMINO DEV

POR HÉCTOR PÉREZ

DISTILLERY

ER CLOUD4U

¡Gracias a todos nuestros patrocinadores!

{Nu} NET UNIVERSITY

BY COMMUNITY • FOR COMMUNITY • #GLOBALAZURE 2021 Latinoamérica

#GlobalAzureLatam #GlobalAzure

Service Mesh en Azure Kubernetes Services

Robert Rozas Navarro
Senior Customer Engineer
Microsoft

Before microservice era

Microservice

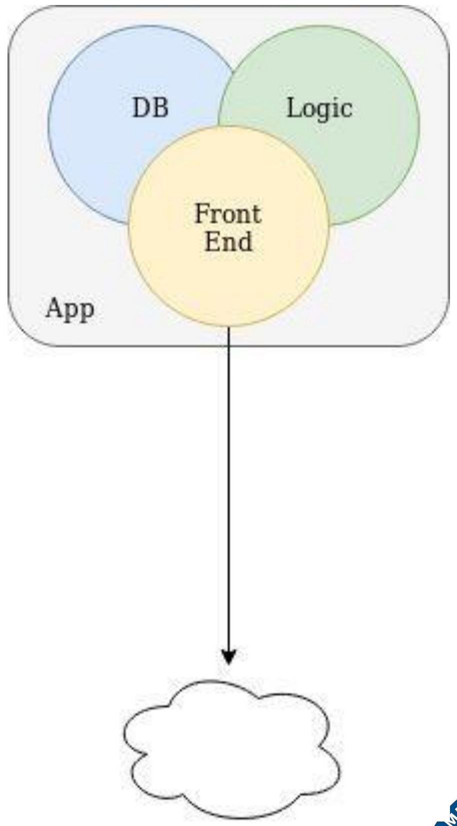
Service Mesh

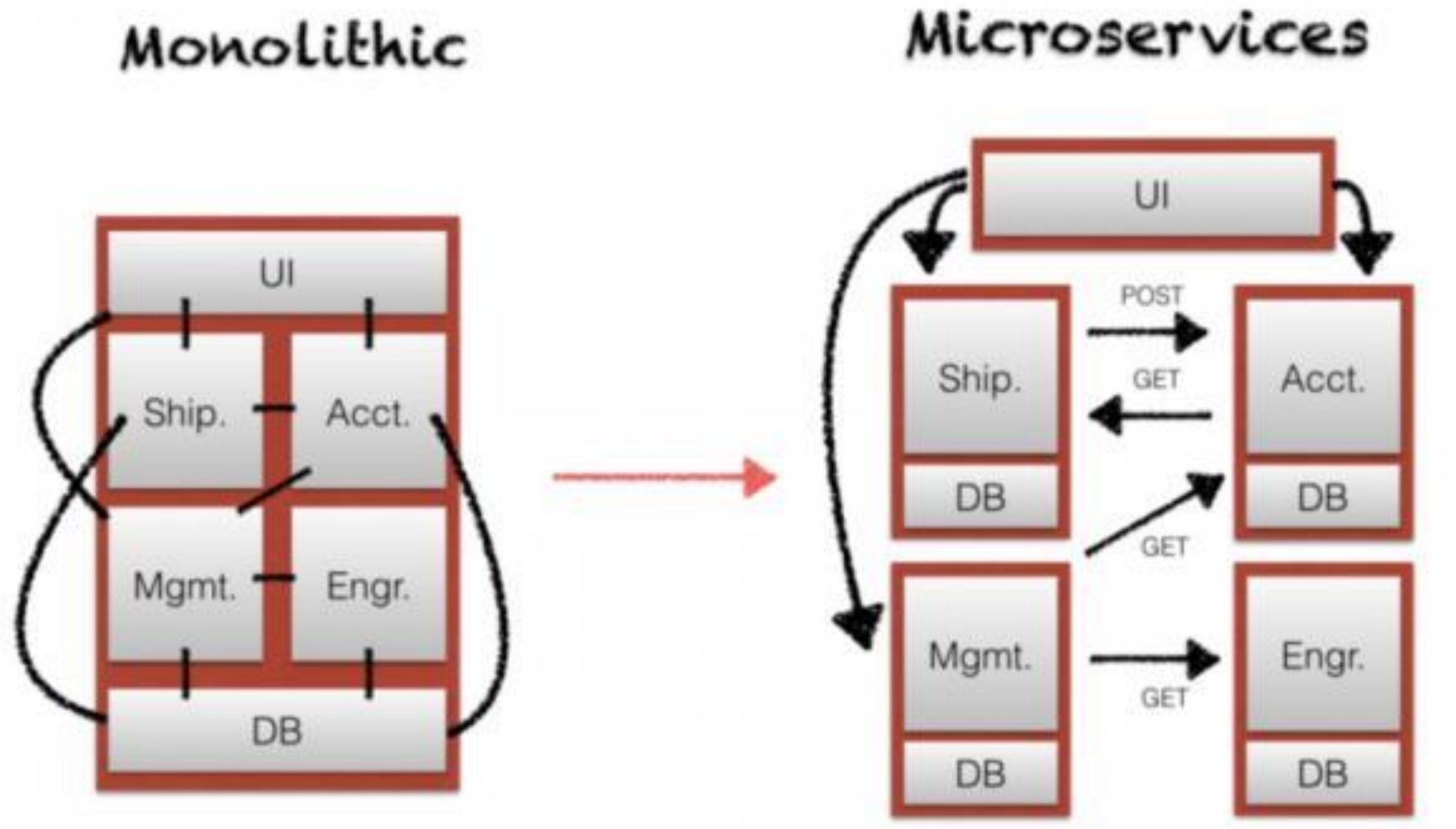
Istio

Traffic control
Secure
Polices control
Observe

MONOLITHIC ARCHITECTURE

- Strong Coupling between different modules causing anti-patterns in communicating between different modules
- Difficulties in Scaling
- Updating to new version requires complete re-install
- Problem in one module can cause the whole application to crash
- Difficult to move to a new framework or technology



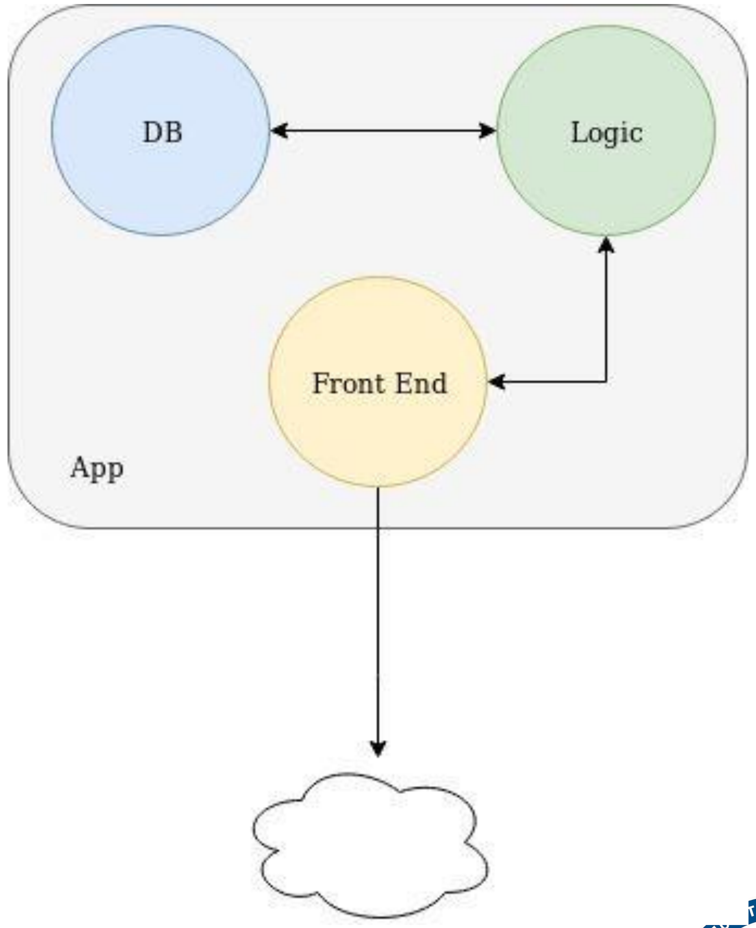

Monolithic
UI
Ship.
Acct.
Mgmt.
Engr.
DB
Microservices
UI
POST
Ship.
GET
Acct.
DB
DB
GET
Mgmt.
Engr.
GET
DB
DB

From Server to Service/Container Abstraction

Virtual Machines	⇒	Containers
Data Centers	⇒	Orchestrated Environments
Hardware Redundancy	⇒	Design for Failure
Servers	⇒	Service
IP Addresses, DNS	⇒	Service Discovery
Server Monitoring	⇒	Service Monitoring
Monolithic Applications	⇒	Microservices
TCP/IP	⇒	gRPC, REST, DataPack

MICROSERVICES ARCHITECTURE

- API contract between different modules/service ensures that each module can be developed and maintained independently
- Each service can be scaled independently
- Updating to new version requires only updates to a specific services
- Allows for easier CI/CD



AT WHAT COST?

- Replaced a reliable in-process call with an unreliable RPC.
- Secure in-process communication is replaced by insecure network.
- Access control within process was a no-op
- Latency went up
- Trivial single-stepping replaced by …?

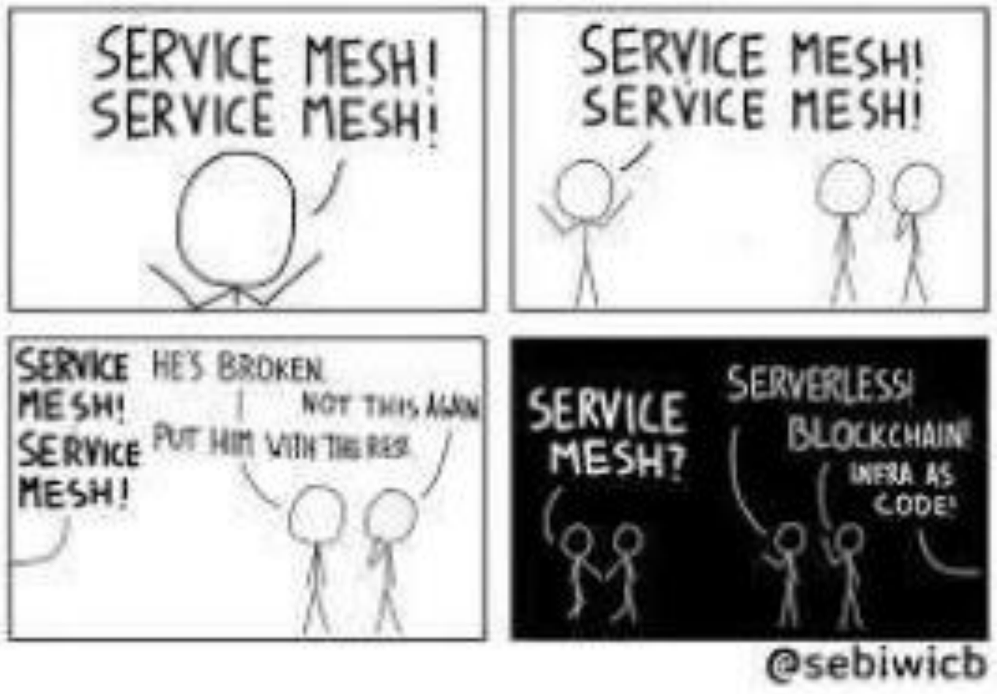
Can we fix it?

- Add retry logic to the application code
- Add entry-exit traces
- Secure inter-service connections with strong authentication

Standard Requirements for Microservice Architecture

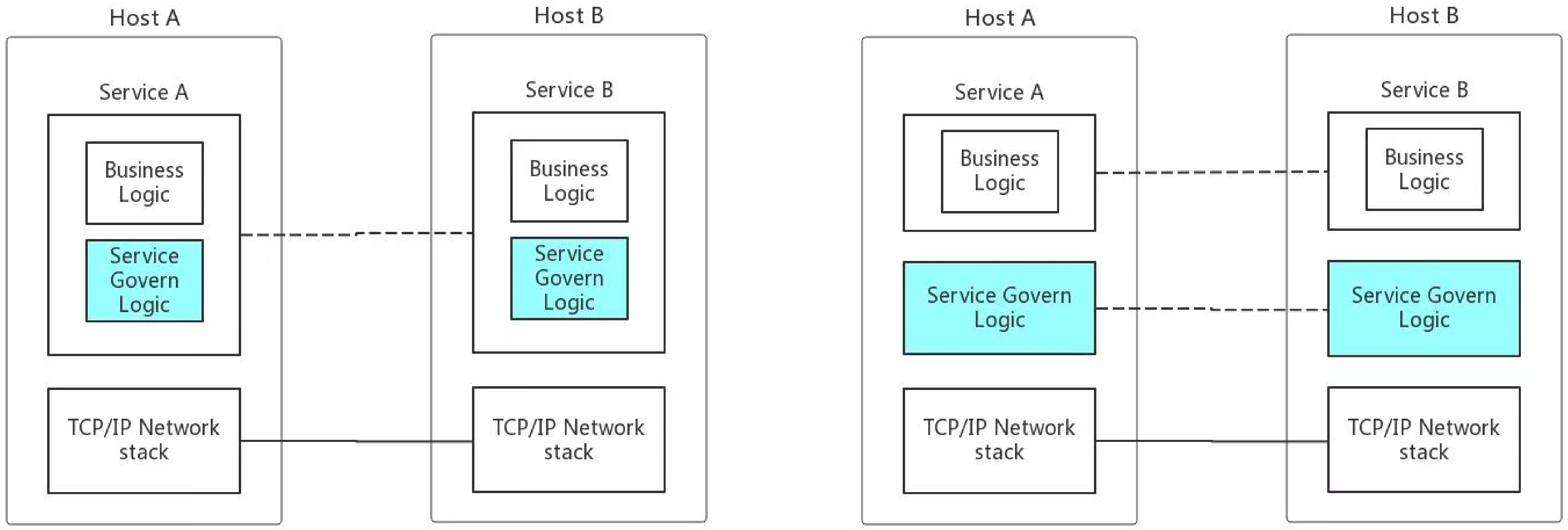
- **Load Balancing**
- **Routing**: path based routing – L7 intelligent proxy
- **Auto Service Discovery**
- **Resiliency for inter-service communications:**
  circuit-breaking
  retries and timeouts
  fault injection and handling
  rate limiting
- **Observability**:
  metrics, monitoring, distributed logging and distributed tracing
- **Security**:
  mTLS and key management
- **Multiple Inter-service communication protocols**
- **Configuration information**
- **Deployment**: native support for docker/k8s

Then what kind of software architecture can help us to sort it out?



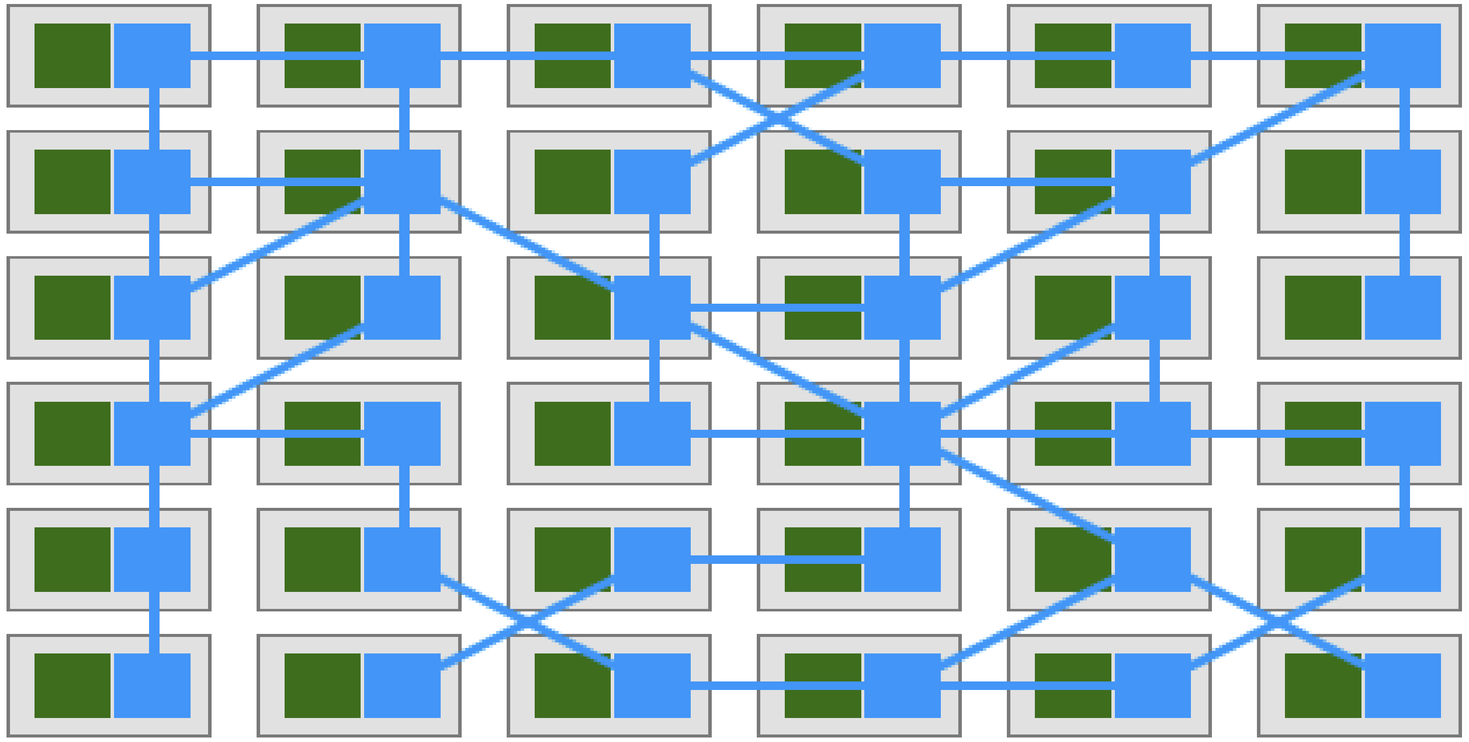
#GlobalAzureLatam #GlobalAzure

SERVICE MESH



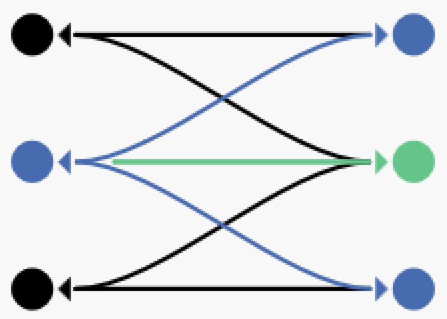
SERVICE MESH

In such model, each of your services will have a companion proxy sidecar. Given that services communicate with each other only through the sidecar proxy, we end up with a deployment similar to the diagram below:



Feature	Istio	Linkerd	Consul Connect
Traffic Redirection (Blue/Green deployment)	Yes	No	No
Traffic Splitting (Canary deployment)	Yes	No	No
Attribute based routing	Yes	No	No
Service Identification	Yes	No	Yes
Auto Proxy Injection	Yes	Yes	Yes
Non-Admin installation	No	Yes	No
Built-in Dashboard	Yes	Yes	No
Certificate Management	Yes	No	Yes
Metrics Collection	Yes	Yes	No
Built-In Dashboard	Yes	Yes	No
TLS	Yes	Yes	Yes
External Service Support	Yes	No	Yes
Rate Limiting	Yes	No	No

ISTIO, MAKE MICROSERVICE GREAT AGAIN

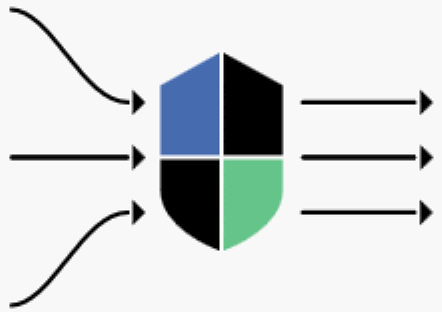


Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

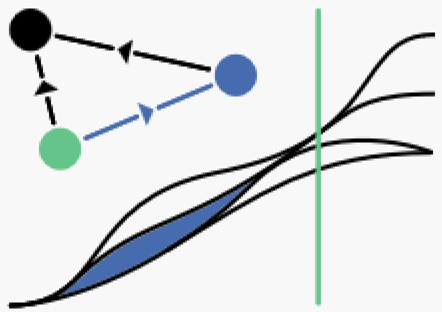
Secure

Automatically secure your services through managed authentication, authorization, and encryption of communication between services.



Control

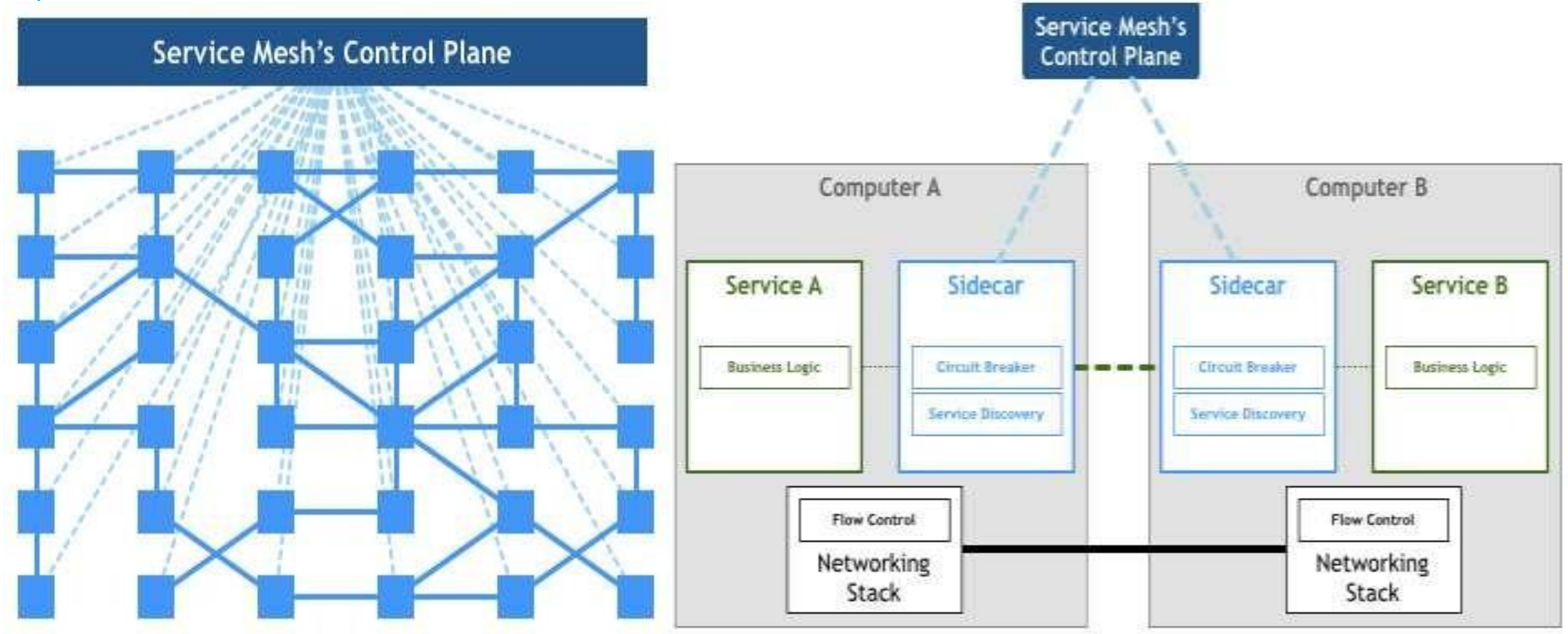
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.



Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

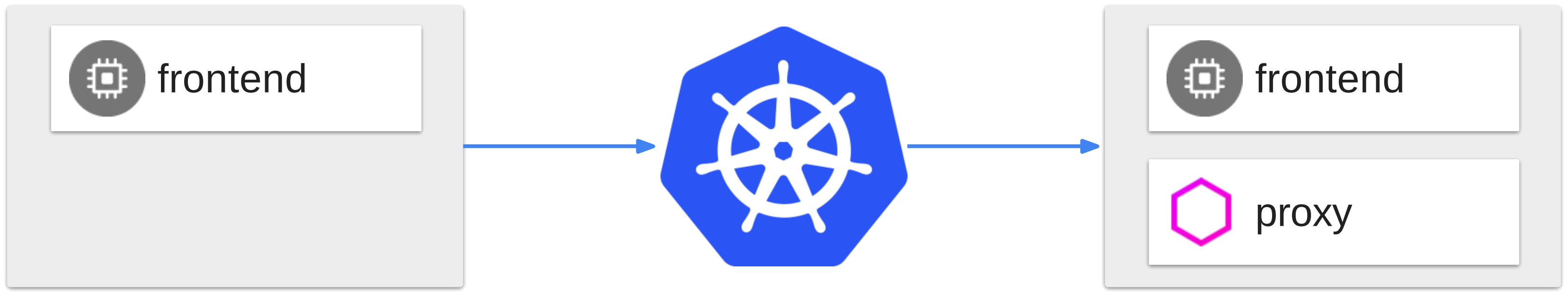
Istio


Service Mesh's Control Plane
Service Mesh's Control Plane
Computer A
Computer B
Service A
Business Logic
Sidecar
Circuit Breaker
Service Discovery
Sidecar
Circuit Breaker
Service Discovery
Service B
Business Logic
Flow Control
Networking Stack
Flow Control
Networking Stack

INJECTION

```
spec:
  containers:
  - image: frontend:latest
```

```
spec:
  containers:
  - image: frontend:latest
  - image: istio/proxy
```



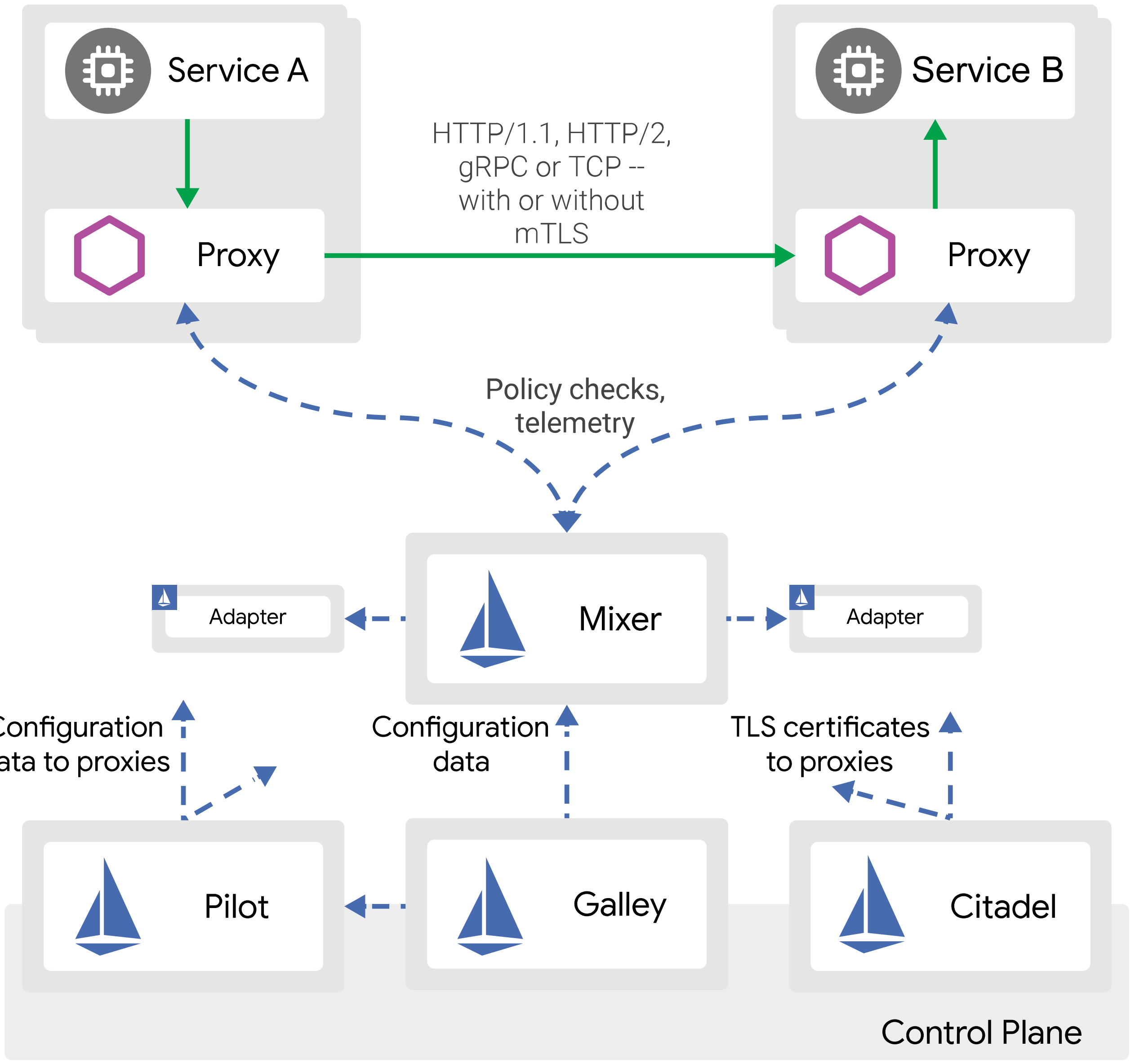
Initializer policy

```
initImage: docker.io/istio/proxy_init
 proxyImage: docker.io/istio/proxy
```

THE SIDECAR PROXY: ENVOY

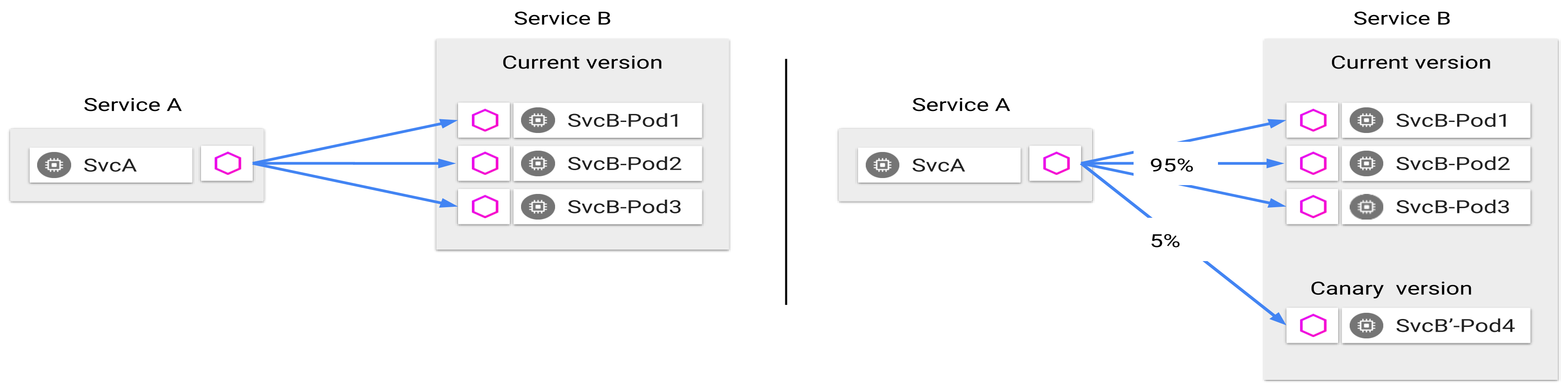
- Dynamic service discovery
- Load balancing
- TLS termination
- HTTP/2 and gRPC proxies
- Circuit breakers
- Health checks
- Staged rollouts with %-based traffic split
- Fault injection
- Rich metrics

Architecture

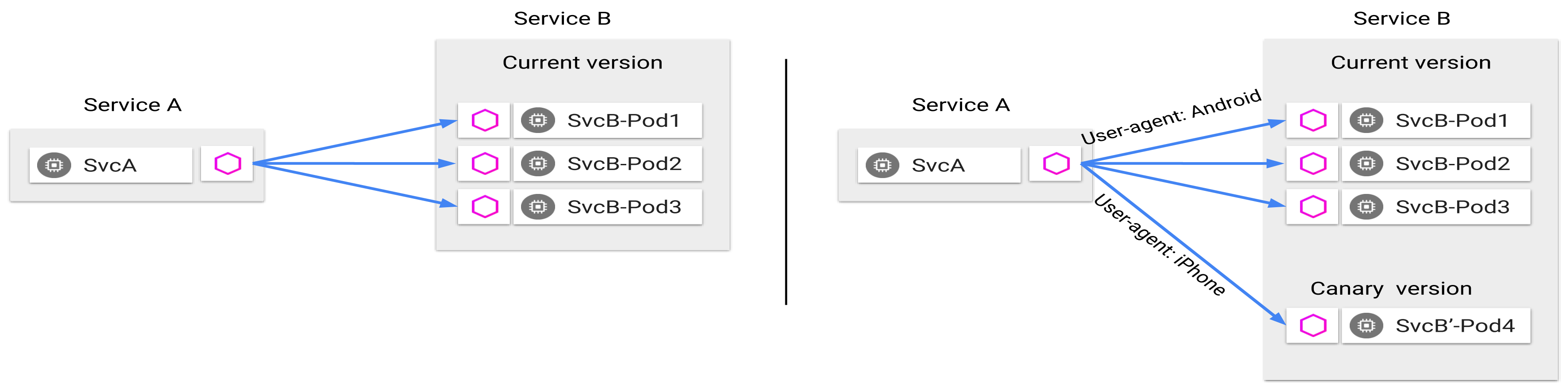


#GlobalAzureLatam #GlobalAzure

TRAFFIC MANAGEMENT



Traffic splitting decoupled from infrastructure scaling - proportion of traffic routed to a version is independent of number of instances supporting the version

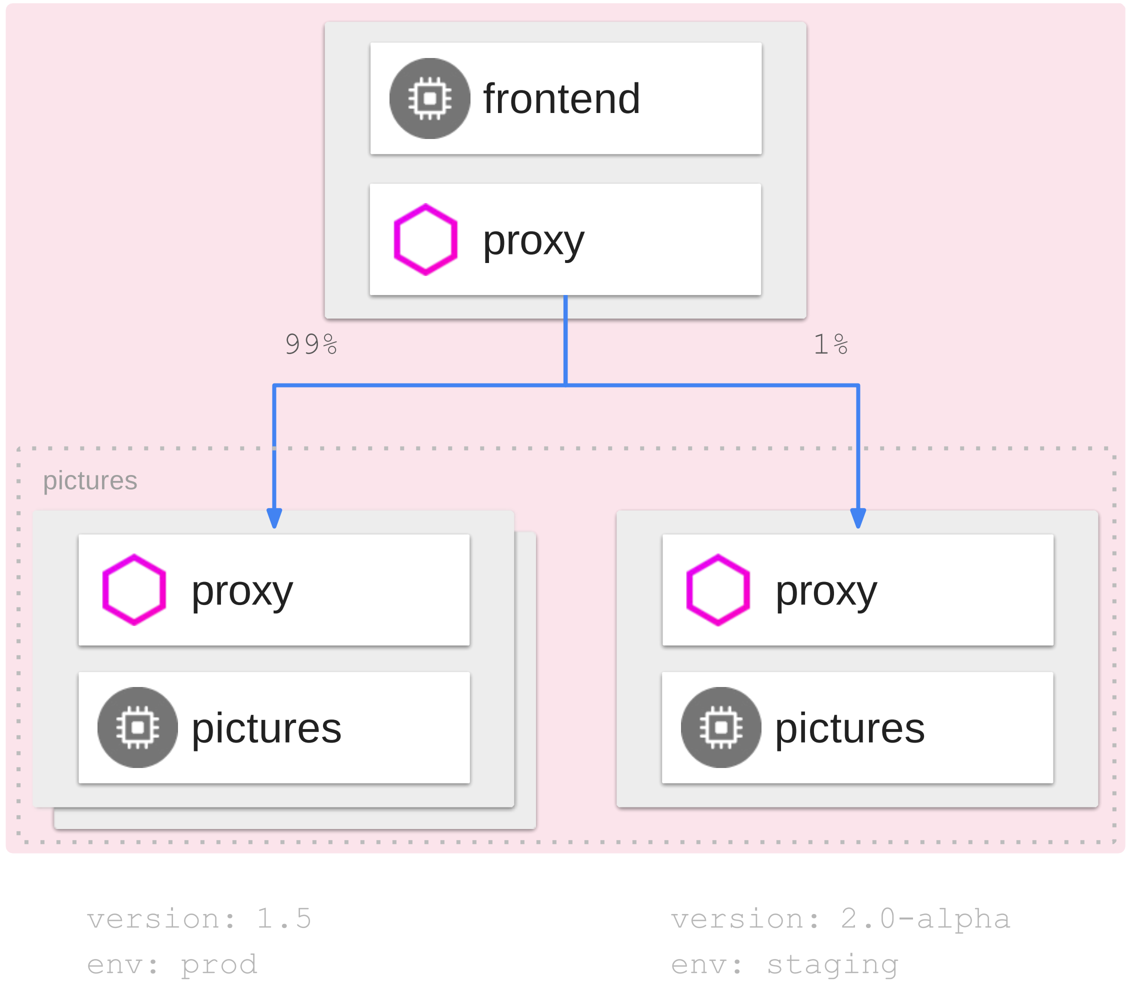


Content-based traffic steering - The content of a request can be used to determine the destination of a request

#GlobalAzureLatam #GlobalAzure

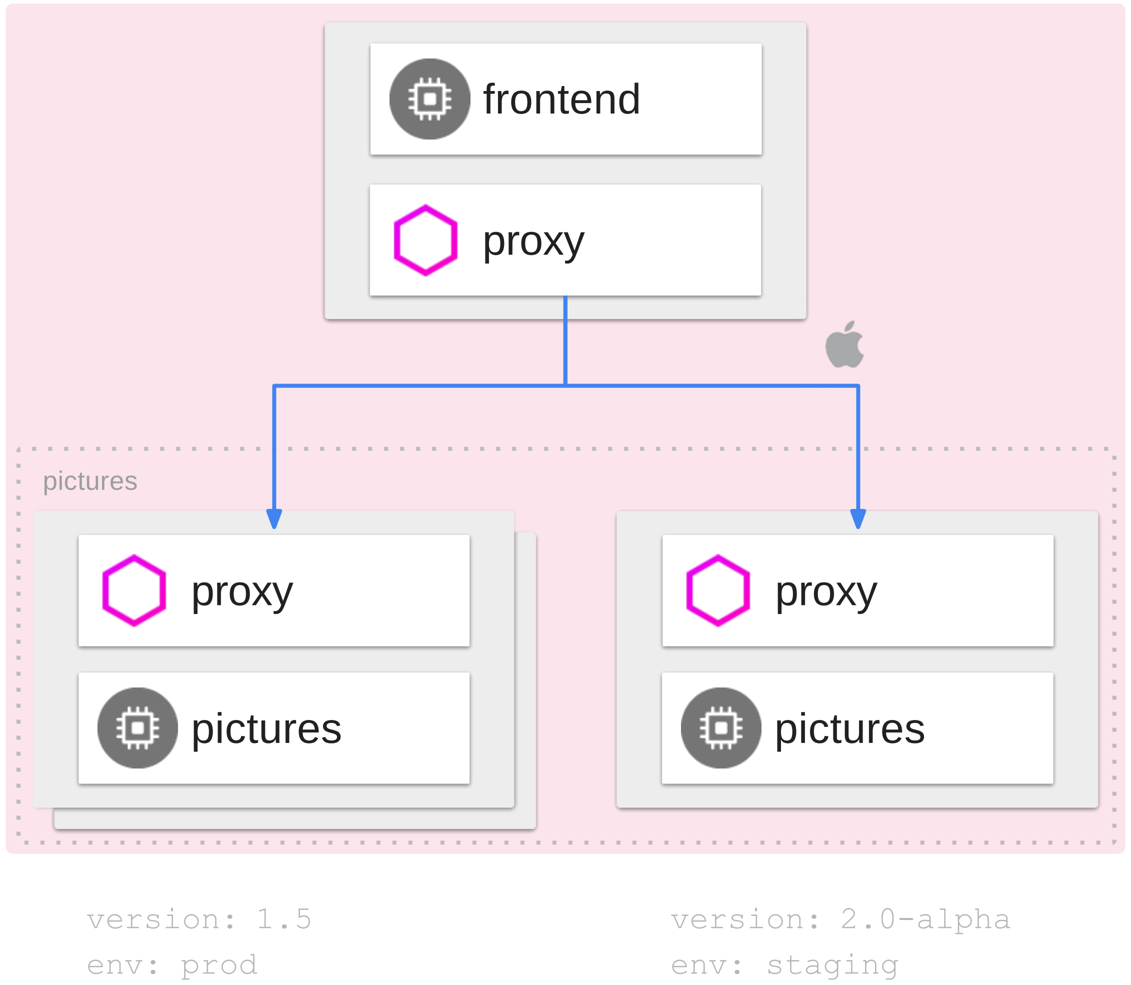
CANARY ROLLOUT

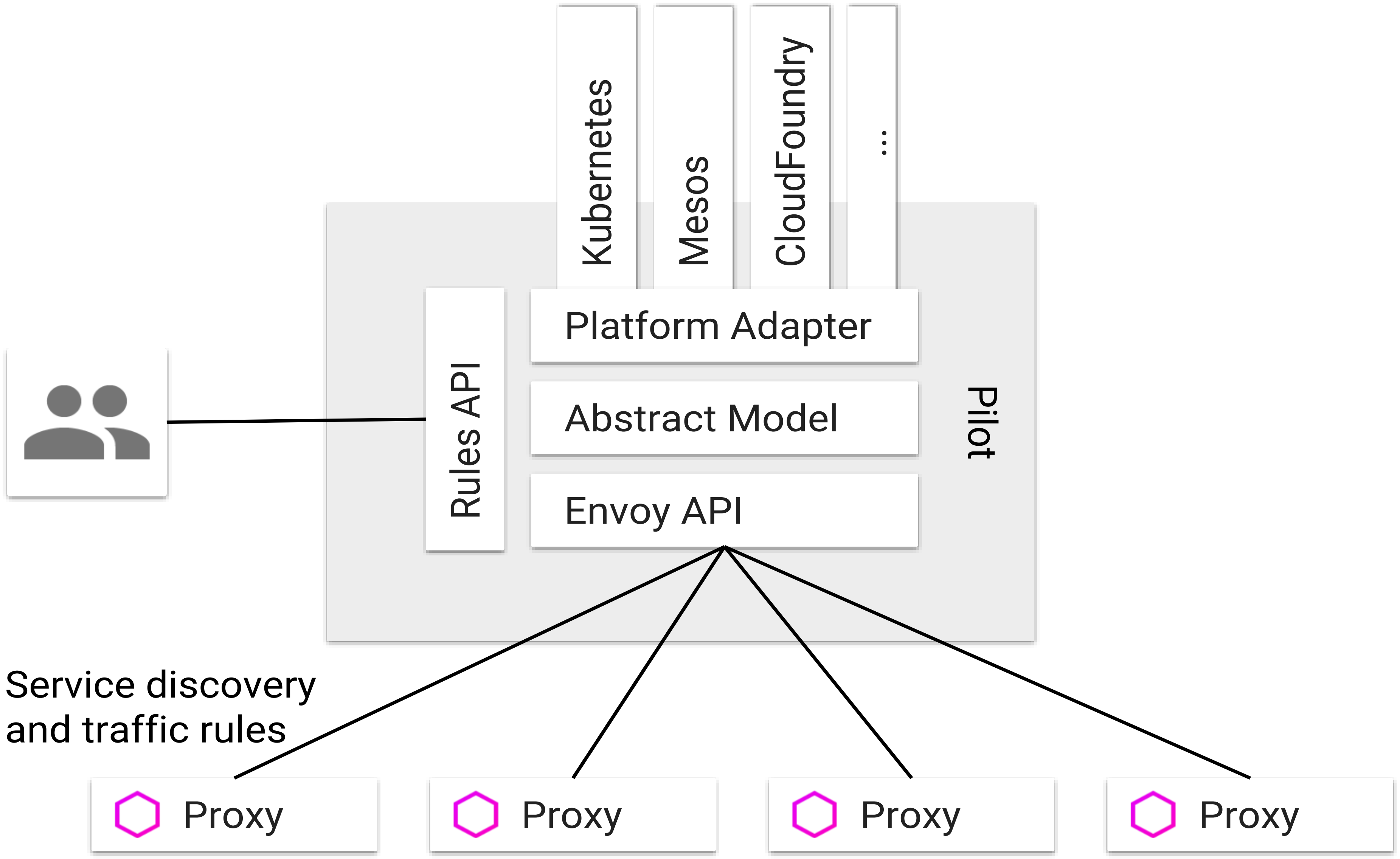
```
// A simple traffic splitting rule
destination: pictures.example.local
match:
  source: frontend.example.local
route:
- tags:
    version: v1.5
    env: prod
  weight: 99
- tags:
    version: v2.0-alpha
    env: staging
  weight: 1
```



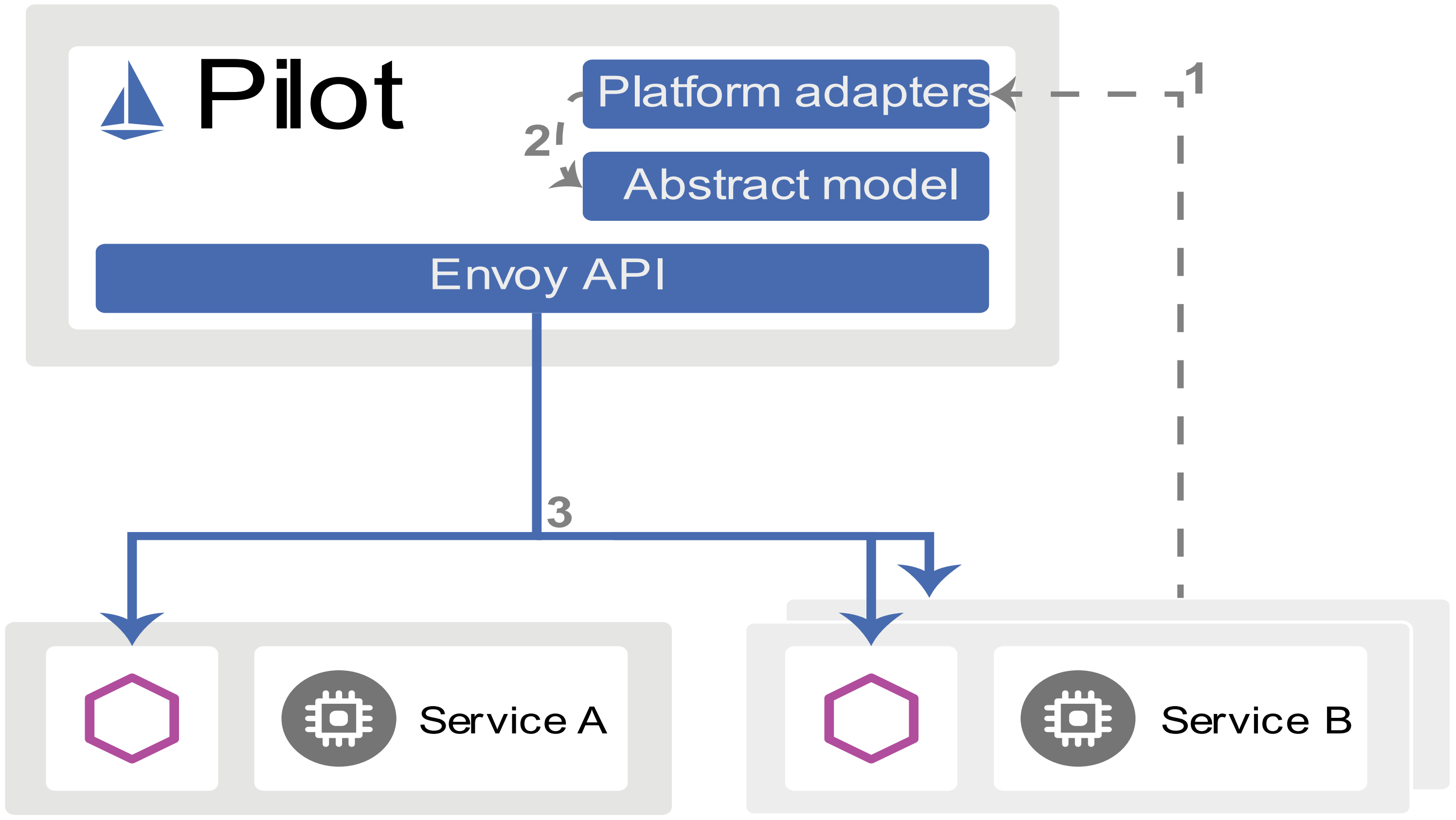
TRAFFIC STEERING

```
// Content-based traffic steering rule
destination: pictures.example.local
match:
  httpHeaders:
    user-agent:
      regex: ^(.*?;)?(iPhone)(;.*)?$
precedence: 2
route:
- tags:
    version: 2.0-alpha
    env: staging
```




Kubernetes
Mesos
CloudFoundry
...
Platform Adapter
Rules API
Abstract Model
Envoy API
Pilot
Service discovery and traffic rules
Proxy
Proxy
Proxy
Proxy

SERVICE DISCOVERY



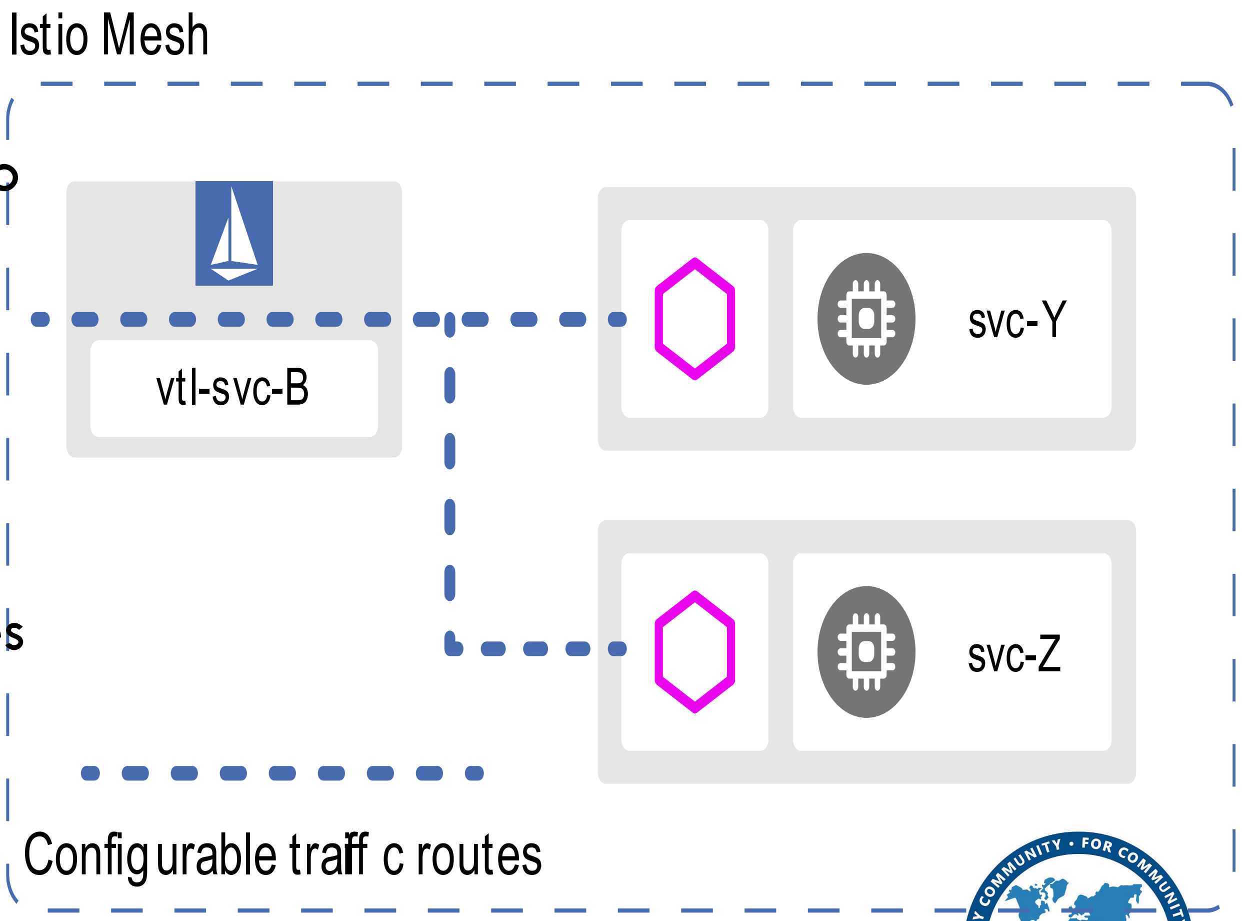
Istio service discovery leverages the service discovery features provided by platforms like Kubernetes for container-based applications.

VITUAL SERVICES

Configure each application service version as a subset and add a corresponding destination rule to determine the set of pods or VMs belonging to these subsets.

Configure traffic rules in combination with gateways to control ingress and egress traffic.

Configure traffic routes to your application services using DNS names. These DNS names support wildcard prefixes or CIDR prefixes to create a single rule for all matching services.



VITUAL SERVICE ROUTING RULE

match

headers

cookie

regex

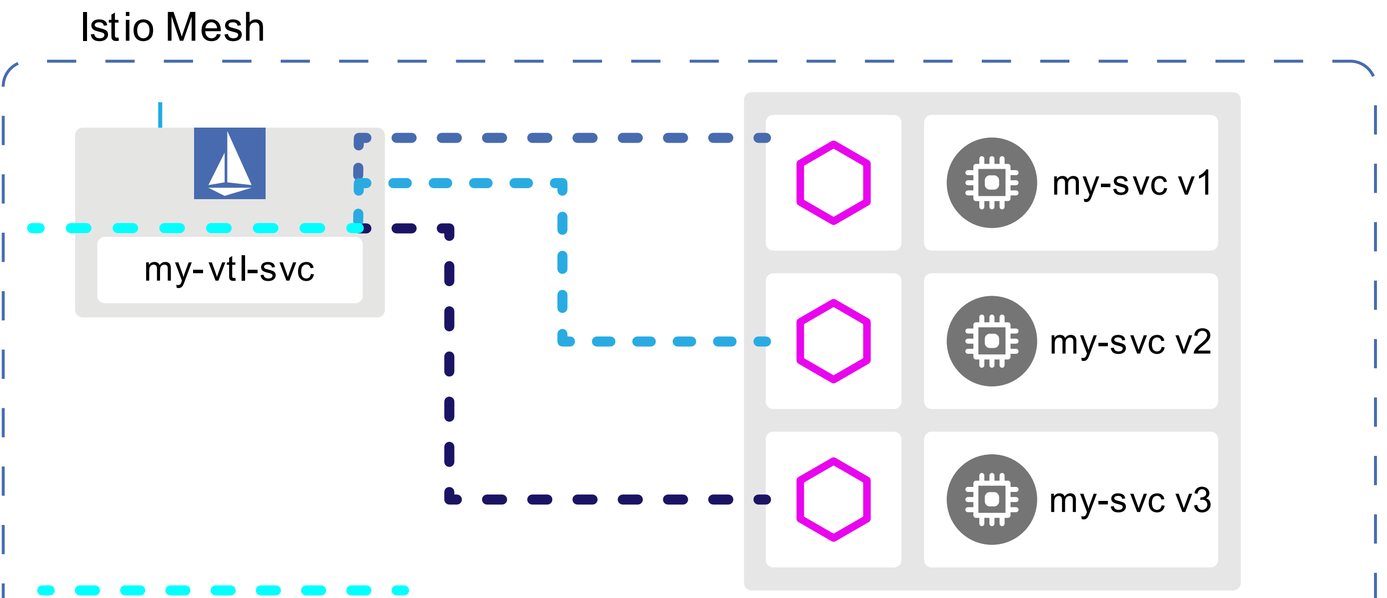
route

destination

Host

subset

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: my-vtl-svc
spec:
  hosts:
    - "*"
 http:
 - match:
   - headers:
       cookie:
         regex: "^(.*?;)?(user=jason)(;.*)?$"
   route:
     - destination:
         host: my-svc
         subset: v2
 - route:
   - destination:
       host: my-svc
       subset: v1
```



Route of incoming traffic configured
to use a simple random load balancer.

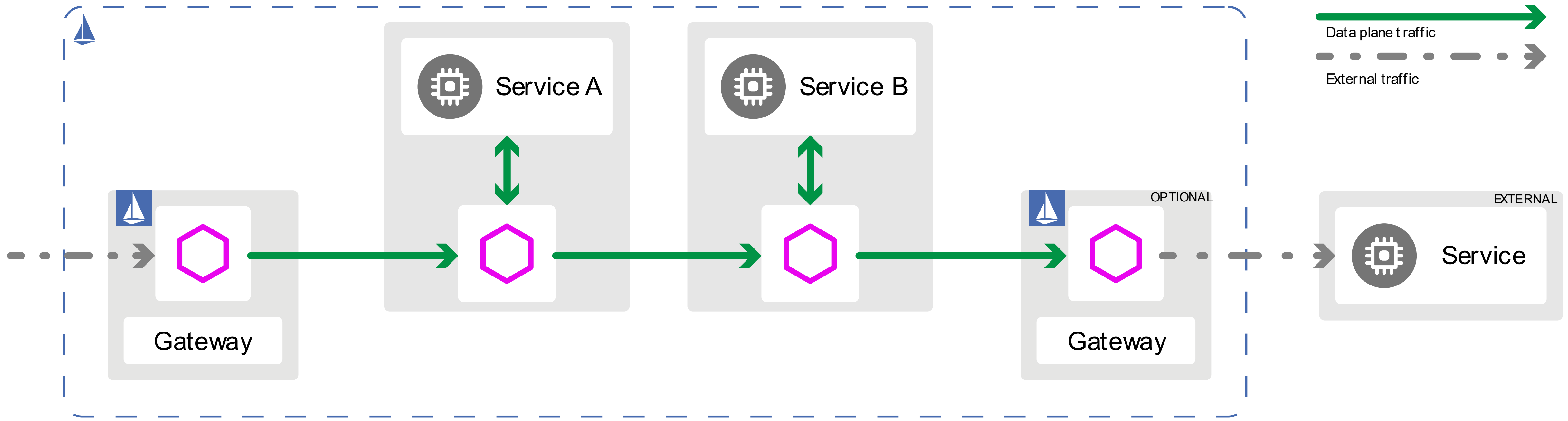
Destination rule configured to use a simple random
load balancer for the v1 subset of your service.

Destination rule configured to use a simple round- robin
load balancer for the v2 subset of your service.

Destination rule configured to use a simple random load balancer
for the v3 subset of your service.

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: my-destination-rule
spec:
  host: my-svc
  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
```

GATEWAYS



All traffic enters the mesh through an ingress gateway workload.

Use egress gateways to limit which services can or should access external networks, or to enable secure control of egress traffic to add security to your mesh

LOAD BALANCING

Round robin: Requests are forwarded to instances in the pool in turn, and the algorithm instructs the load balancer to go back to the top of the pool and repeat.

Random: Requests are forwarded at random to instances in the pool.

Weighted: Requests are forwarded to instances in the pool according to a specific percentage.

Least requests: Requests are forwarded to instances with the least number of requests. See the Envoy load balancing documentation for more information.

SECURITY

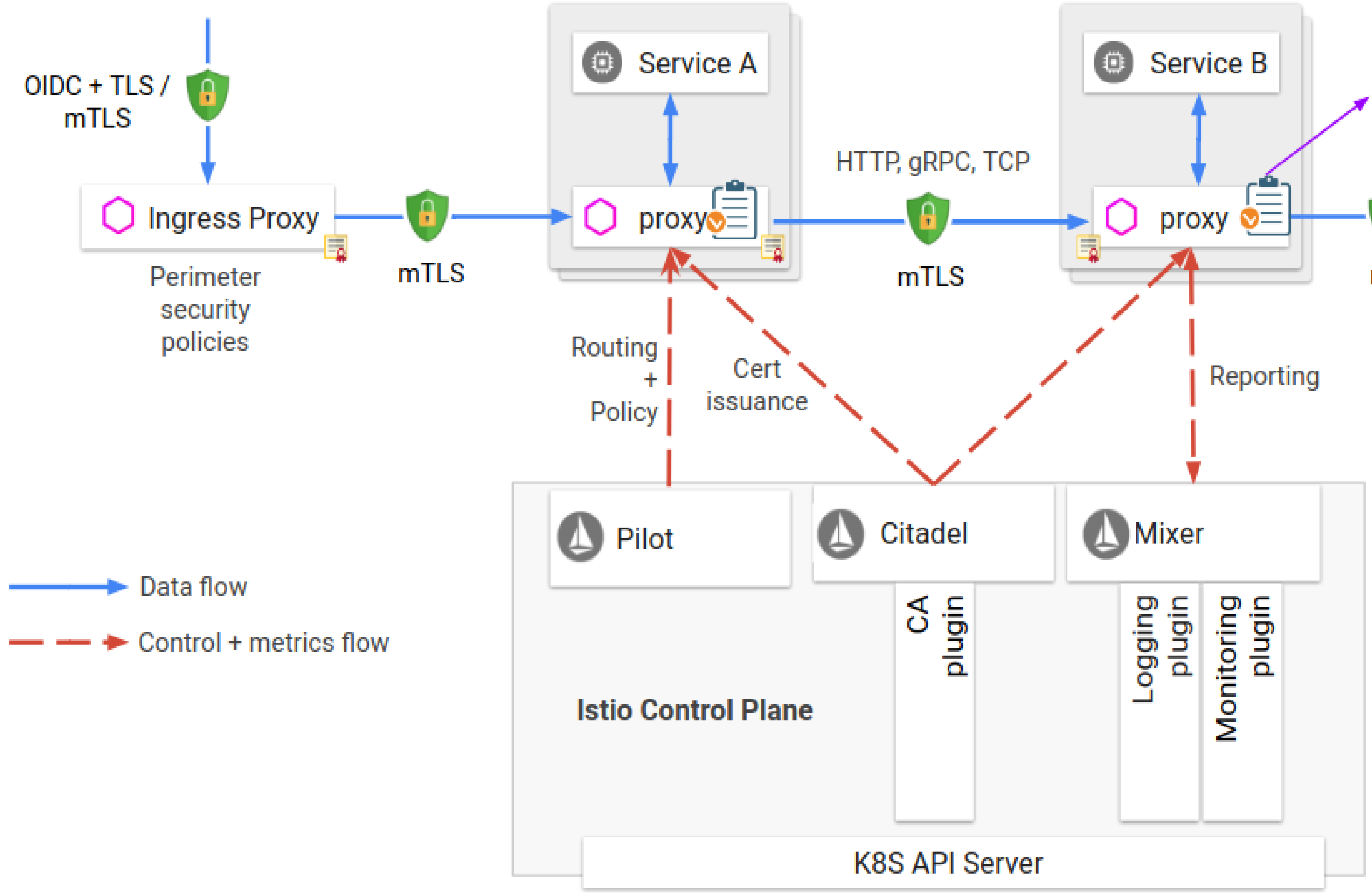
Breaking down a monolithic application into atomic services offers various benefits, including better agility, better scalability and better ability to reuse services. However, microservices also have particular security needs:

•To defend against the man-in-the-middle attack, they need traffic encryption.

•To provide flexible service access control, they need mutual TLS and fine-grained access policies.

•To audit who did what at what time, they need auditing tools.

Istio Security tries to provide a comprehensive security solution to solve all these issues.



Security policies can be implemented at different levels of granularity - Service, Namespace, Mesh.

Monitoring & tracing should not be an afterthought in the infrastructure

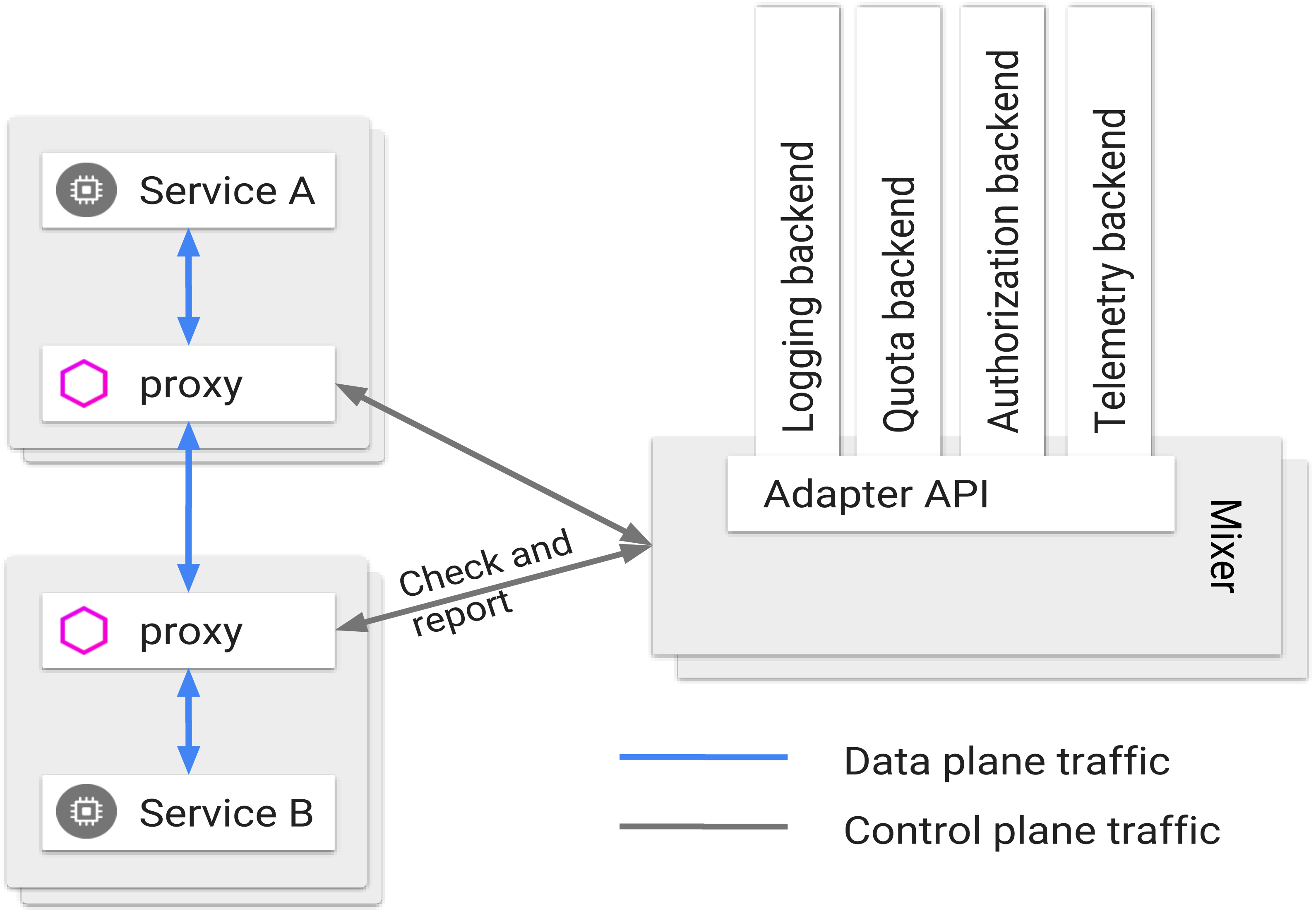
Metrics: prometheus

Logs: EFK

Distributed Tracing: Jaeger/Zipkin

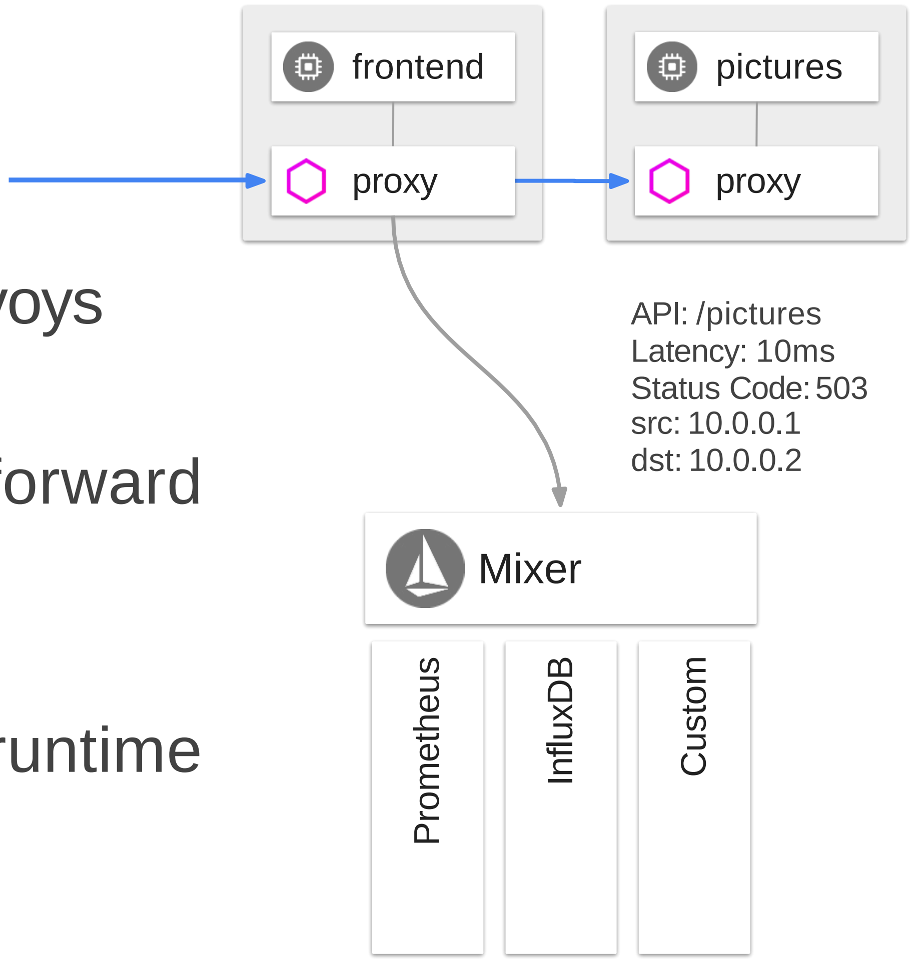
Mesh Visualizing: Kiali

Policies and Telemetry



METRICS FLOW

- Mixer collects metrics emitted by Envoys
- Adapters in the Mixer normalize and forward to monitoring backends
- Metrics backend can be swapped at runtime



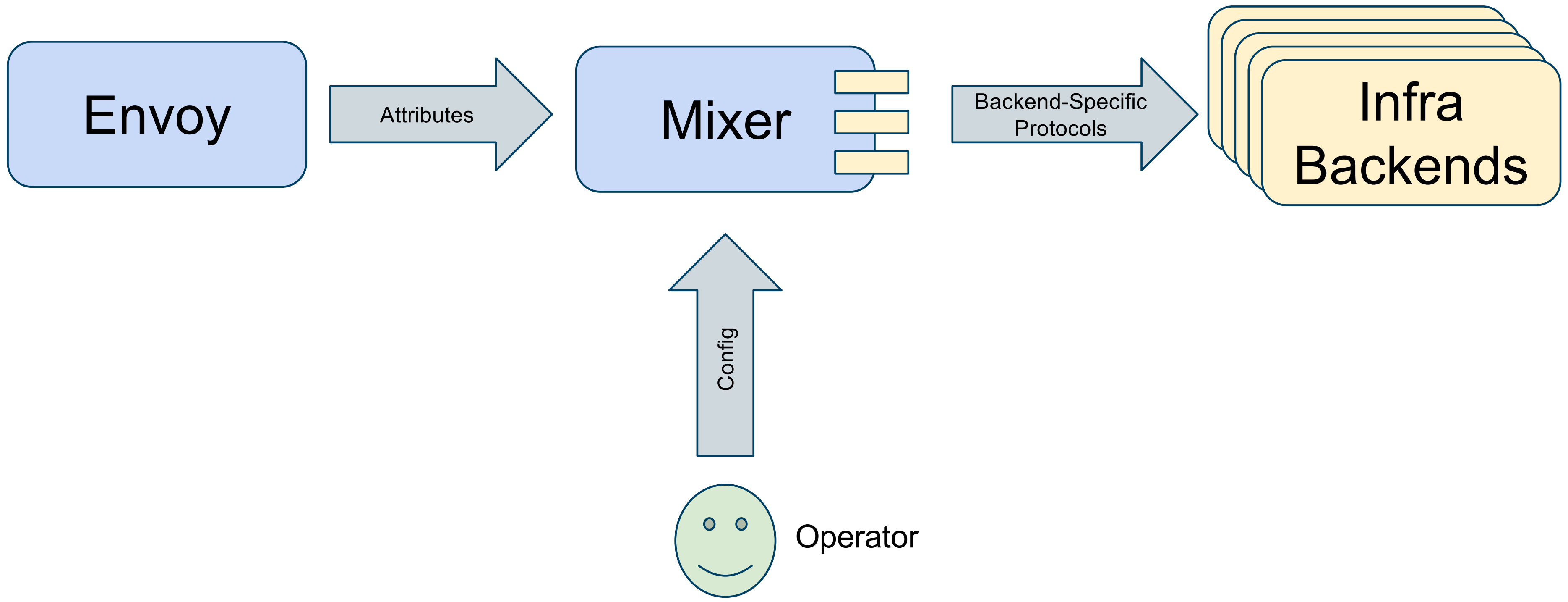
Controlling the policy and telemetry features

- Configuring a set of handlers, which determine the set of adapters that are being used and how they operate.

- Configuring a set of instances, which describe how to map request attributes into adapter inputs. Instances represent a chunk of data that one or more adapters will operate on.

- Configuring a set of rules, which describe when a particular adapter is called and which instances it is given. Rules consist of a match expression and actions

MIXER ADAPTER MODEL



MIXER ADAPTER MODEL

Handlers: configuring adapters

Templates: adapter input schema

Instances: attribute mapping

Rules: delivering data to adapters

SECCIÓN / DEMO

#GlobalAzureLatam #GlobalAzure

Q & A

#GlobalAzureLatam #GlobalAzure

Latino .NET Online

JET BRAINS

sessionize

StreamYard

ENOVEL® 10 años

Jalasoft®

Microsoft

Microsoft® Press

clarika
SOFTWARE + INNOVATION

BY COMMUNITY • FOR COMMUNITY •
#GLOBALAZURE
2021
Latinoamérica

EL CAMINO DEV
POR HÉCTOR PÉREZ

DISTILLERY

ER CLOUD4U

¡Gracias a todos nuestros patrocinadores!

{nu} NET UNIVERSITY

#GlobalAzureLatam #GlobalAzure

BY COMMUNITY • FOR COMMUNITY •
#GLOBALAZURE
2021
Latinoamérica

¡GRACIAS POR TU ATENCIÓN!

@AshWilliams

@Hackerman

#GlobalAzureLatam #GlobalAzure