

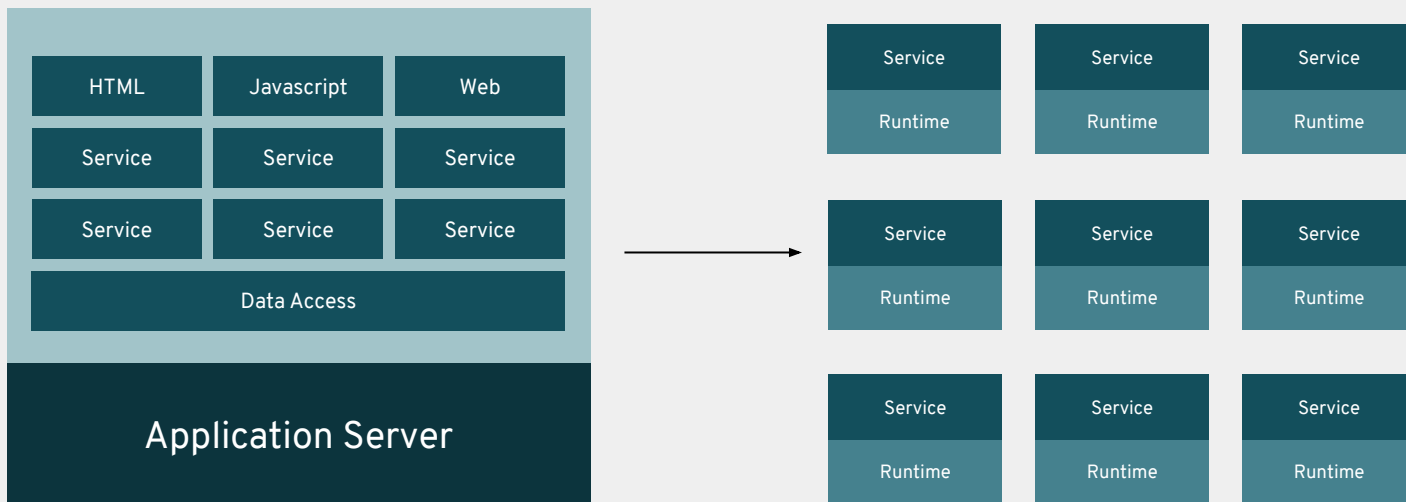
OpenShift 4.x Architecture Workshop

Istio Service Mesh

Microservices Benefits and Challenges

~~MICROSERVICES~~ ARCHITECTURE

DISTRIBUTED



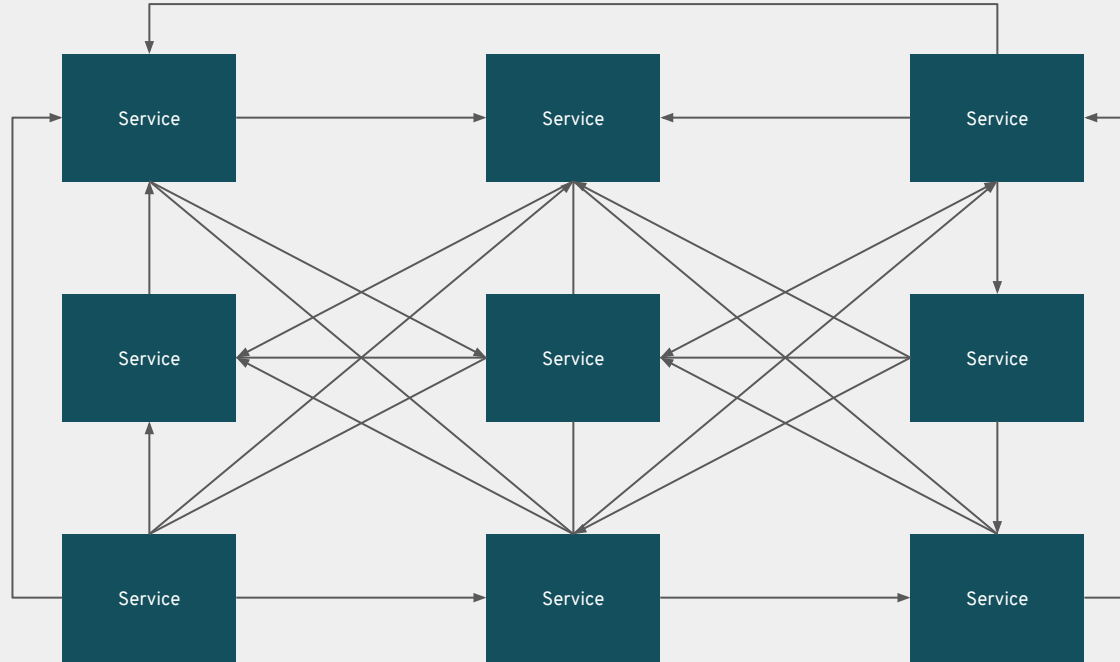
DISTRIBUTED COMPUTING CHALLENGES

Fallacies of Distributed Computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

wikipedia.org/wiki/Fallacies_of_distributed_computing

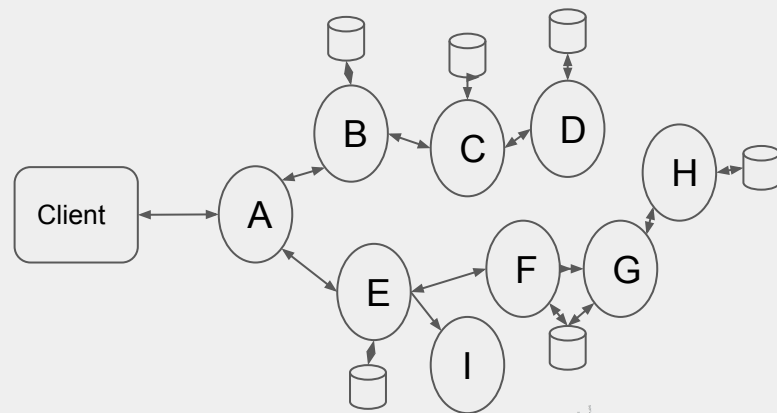
DISTRIBUTED ARCHITECTURE



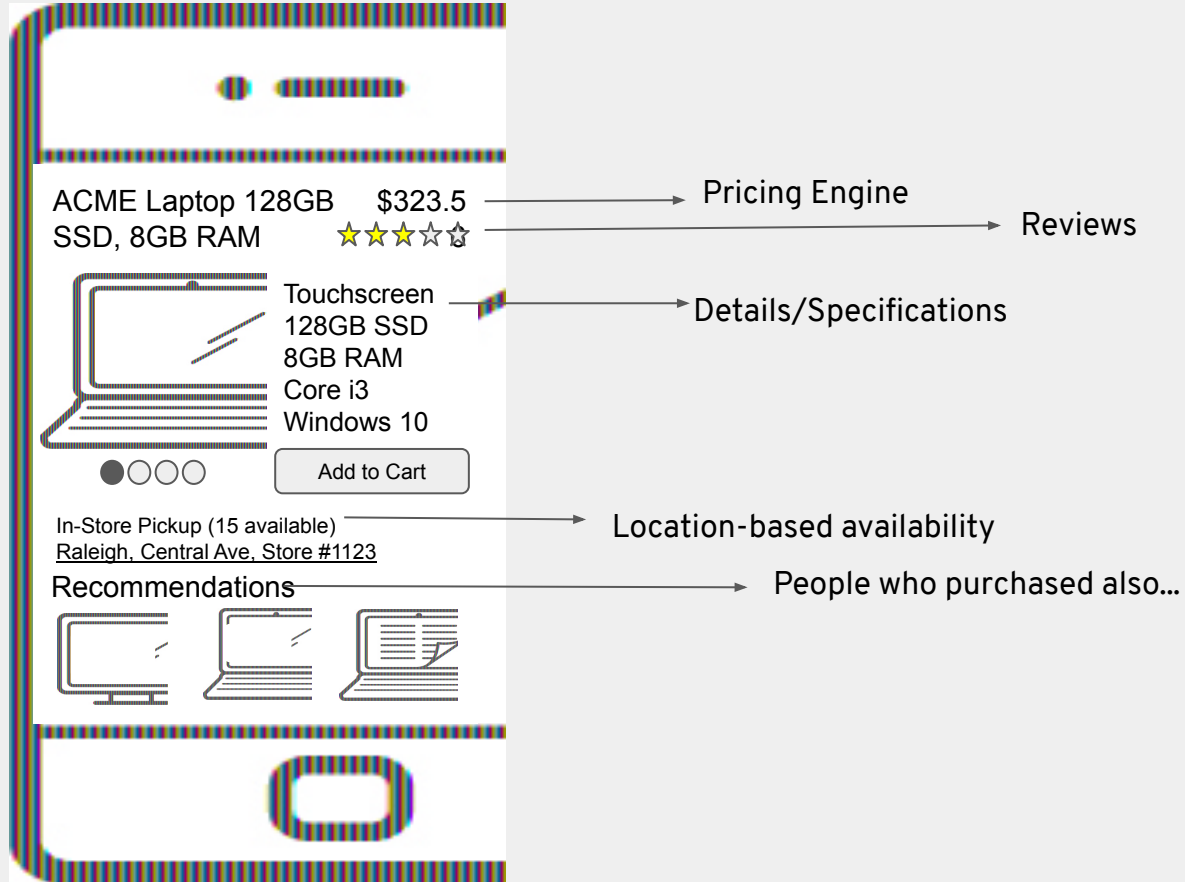
MICROSERVICES ARE HARD

Because applications must deal with

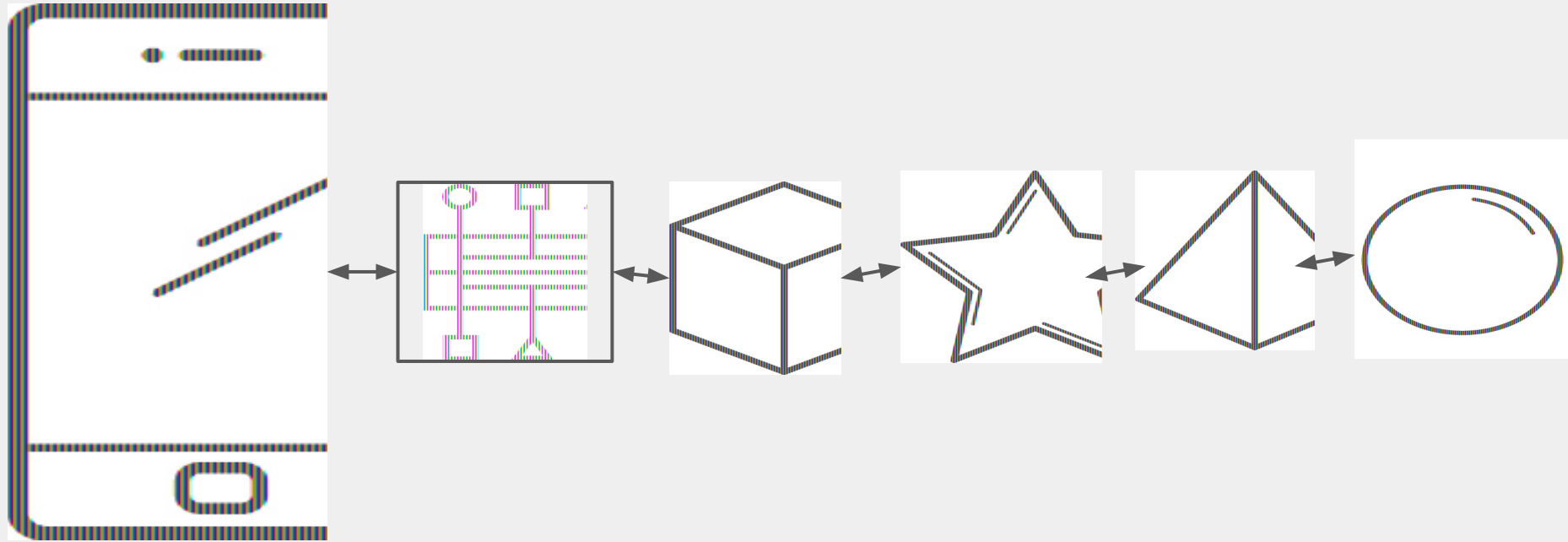
- Unpredictable failures
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources
- Distributed logs
- The fallacies of distributed computing



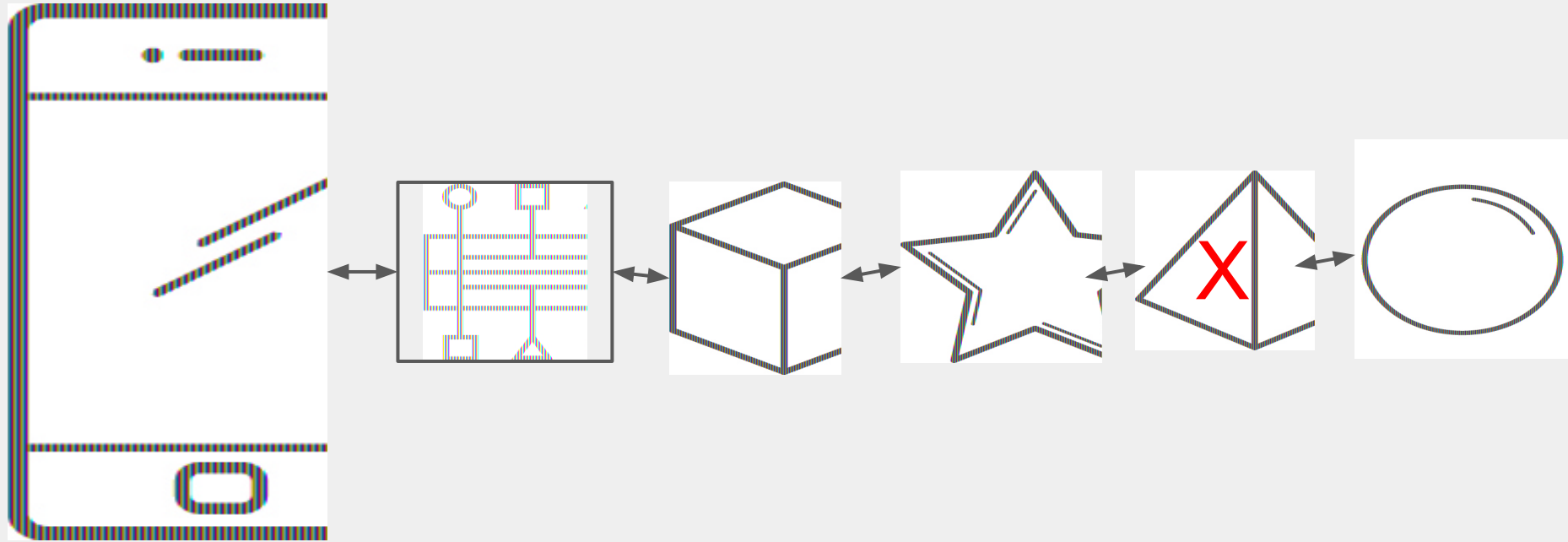
AN EXAMPLE



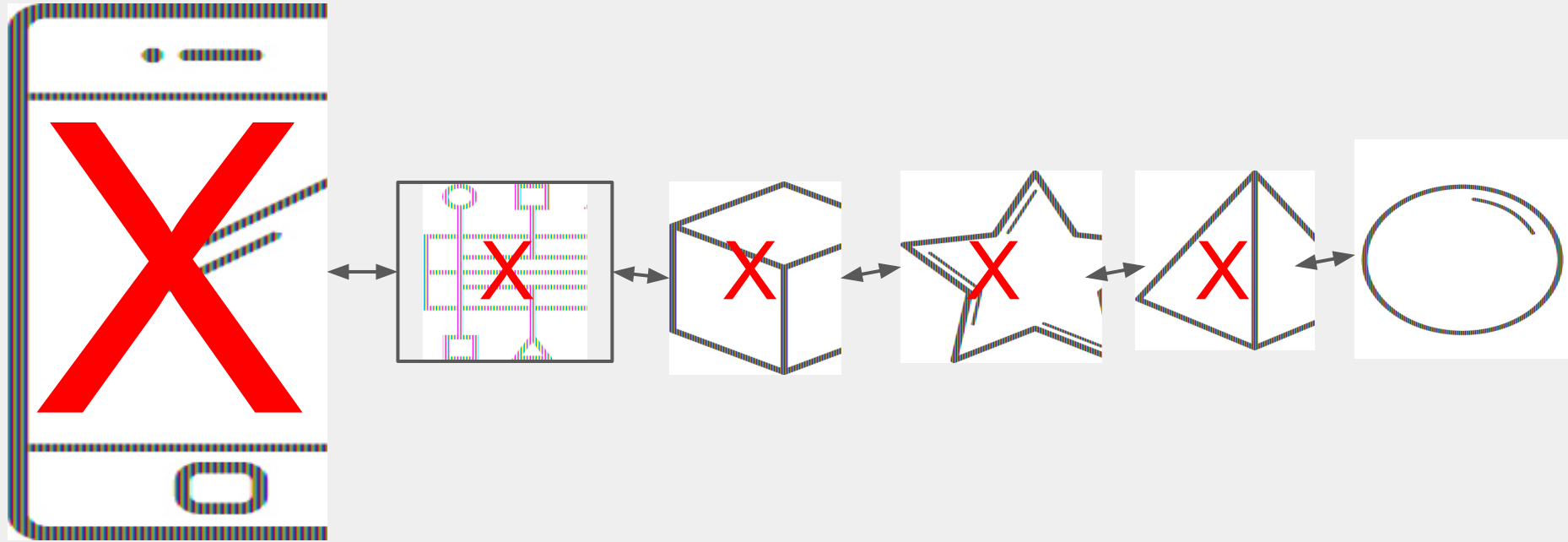
CHAINING




CHAINING (FAILURE)



CHAINING (CASCADING FAILURE)



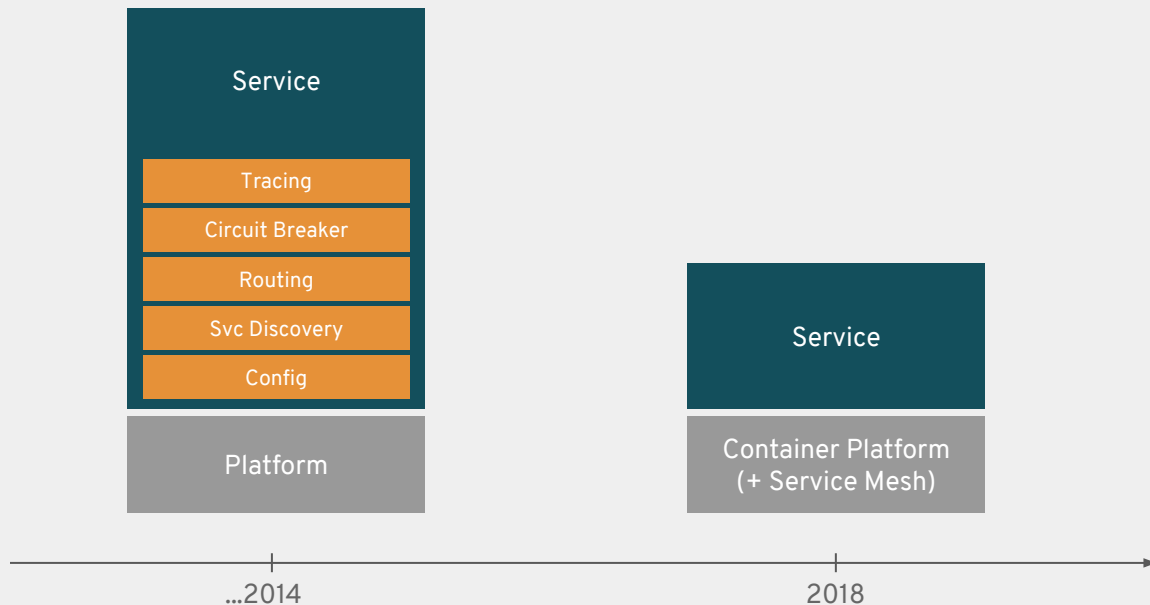
Enter the service mesh



SERVICE MESH

A dedicated network for
service-to-service communications

A better way with a service mesh

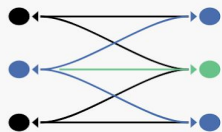


A service mesh provides a **transparent** and **language-independent** network for connecting, observing, securing and controlling the connectivity between services.



Istio

Connect, secure, control, and observe services.



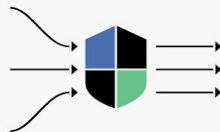
Connect

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



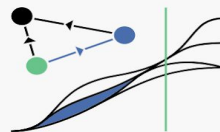
Secure

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'S CAPABILITIES AT 10,000 FEET

Traffic Management.

Rules and traffic routing lets you control the flow of traffic and API calls between services.

Service Identity and Security.

Enforce consistently across diverse protocols and runtimes with little or no application changes.

Policy Enforcement.

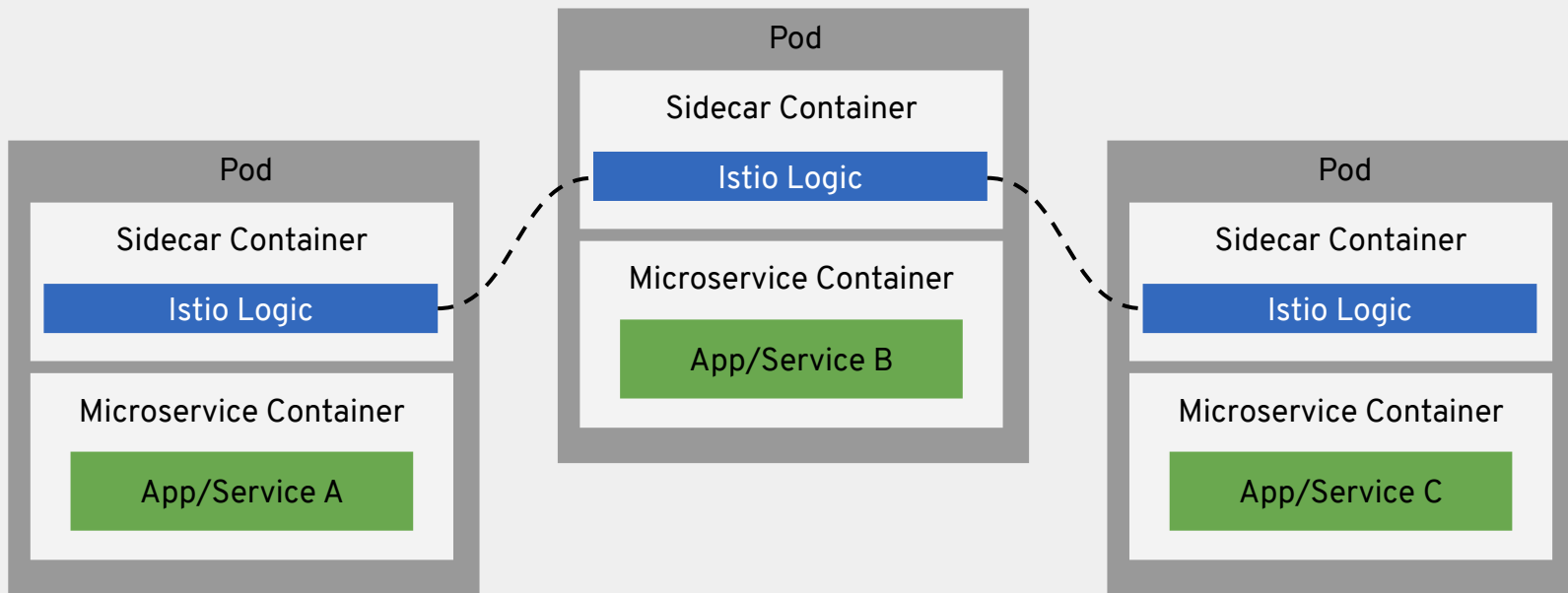
Apply to the interaction between services and ensure they are enforced. Changes are made by configuring the mesh, not by changing application code.

Observability.

Gain understanding of the dependencies between services and the nature and flow of traffic between them, providing the ability to quickly identify and fix issues.

MICROSERVICES WITH ISTIO

connect, manage, and secure microservices transparently

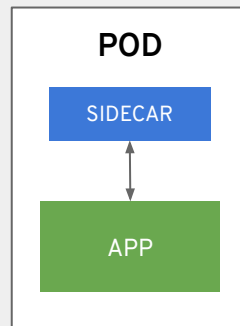


WHAT IS A SIDECAR?

A proxy instance that abstracts common logic away from individual services

SIDECAR PATTERN

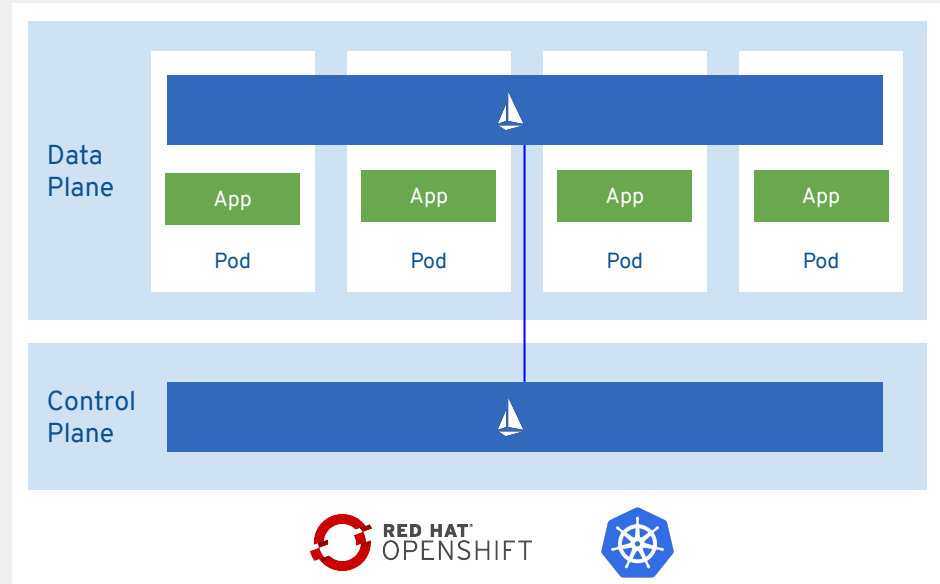
- A utility container in the same pod to enhance the main container's functionality
- Share the same network and lifecycle
- Istio uses an Istio Proxy (L7 Proxy) sidecar to proxy all network traffic between apps



ISTIO PROVIDES BOTH CONTROL AND DATA PLANES

The **data plane** is composed of a set of intelligent proxies (Envoy) deployed as sidecars that mediate and control all network communication between microservices.

The **control plane** is responsible for managing and configuring proxies to route traffic, as well as enforcing policies at runtime.



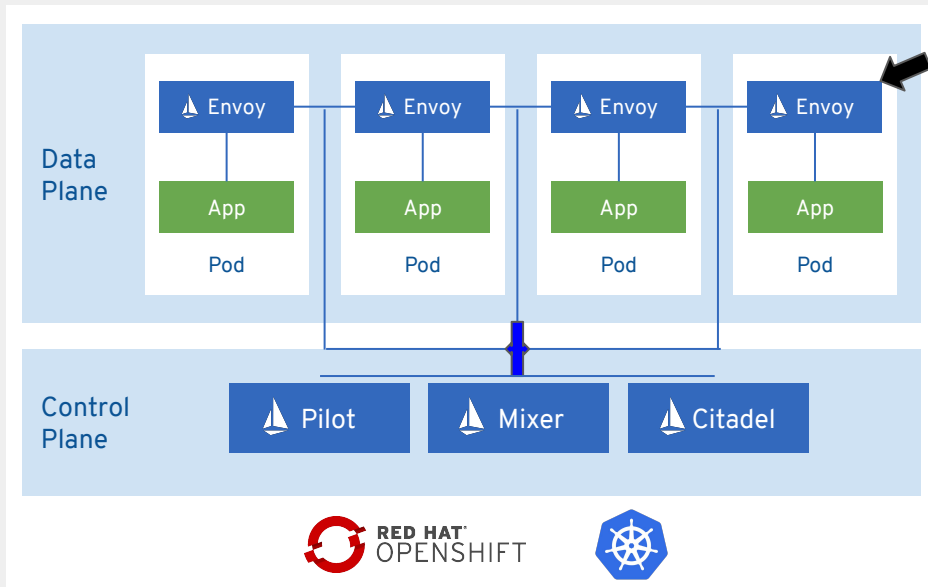
COMPONENTS OF ISTIO

Envoy, originally from Lyft - it's an intelligent proxy. Highly parallel non-blocking, network filtering, service discovery, health checking, dynamically configurable.

Pilot, the component responsible for managing a distributed deployment of Envoy proxies in the service mesh. Intelligent routing, traffic mgmt, resiliency

Mixer, which provides the policy and access control mechanisms within the service mesh. Monitoring, reporting, quotas - plugin-based.

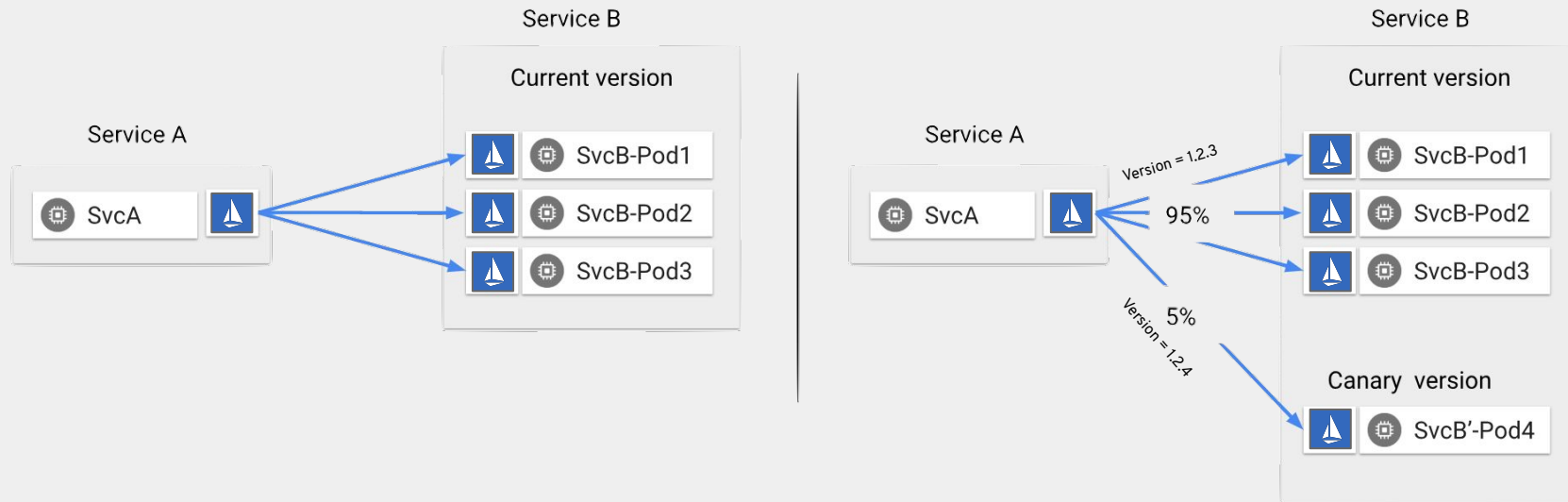
Citadel, control service-service traffic based on origin and user. Key mgmt certificate authority.



It's the sidecar



WHAT DOES CONNECT MEAN?

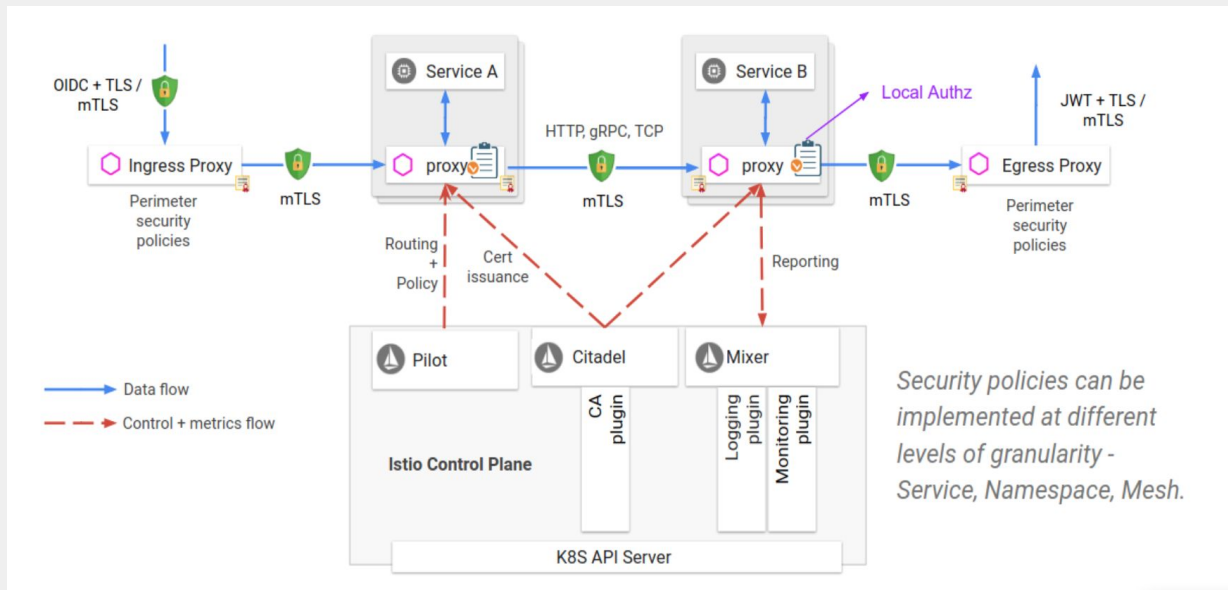


Discovery and Routing: Decoupled from infrastructure, load balancing modes, dynamic routing...

Advanced Deployments: A/B testing, gradual rollouts, canary releases, mirroring...

Failure, Health, and Testing: timeouts, retries, circuit breakers, fault injection, active health checks...

HOW DO YOU SECURE SERVICES?

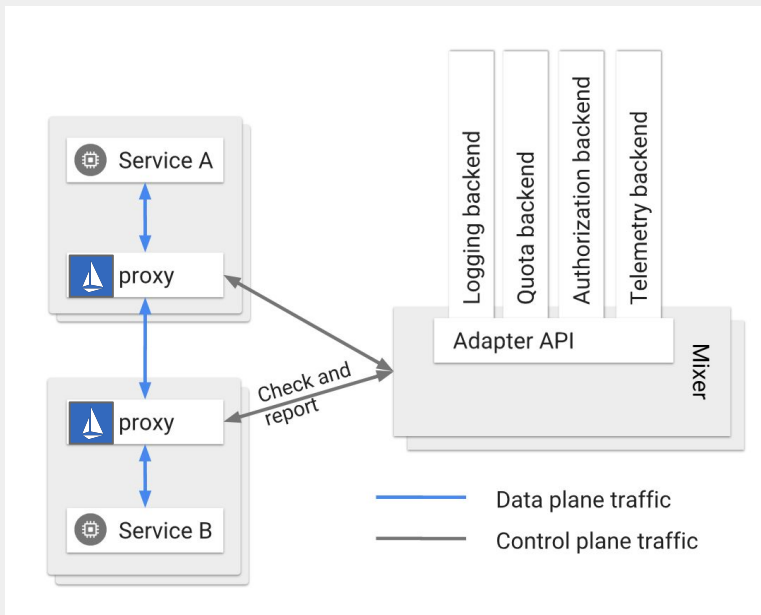


Security by default
no changes needed for
application code and
infrastructure

Defense in depth
integrate with existing security
systems to provide multiple layers
of defense

Zero-trust network
build security solutions on
untrusted networks

WHAT CAN YOU CONTROL?



Restrict to 2 requests per second per IP :

quotas:

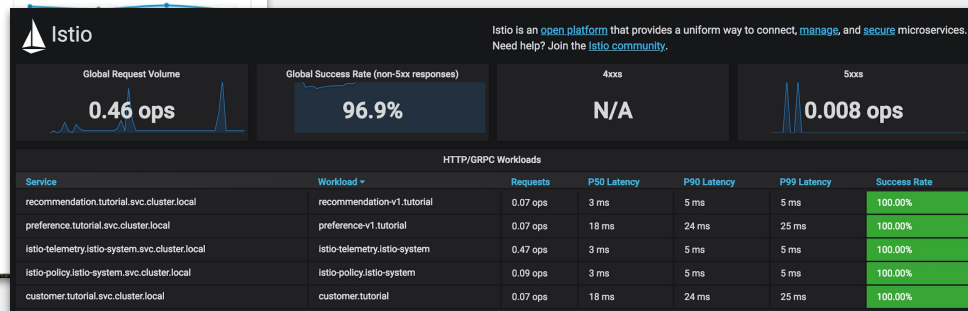
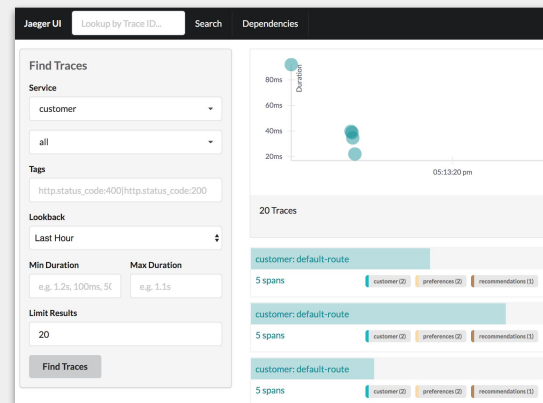
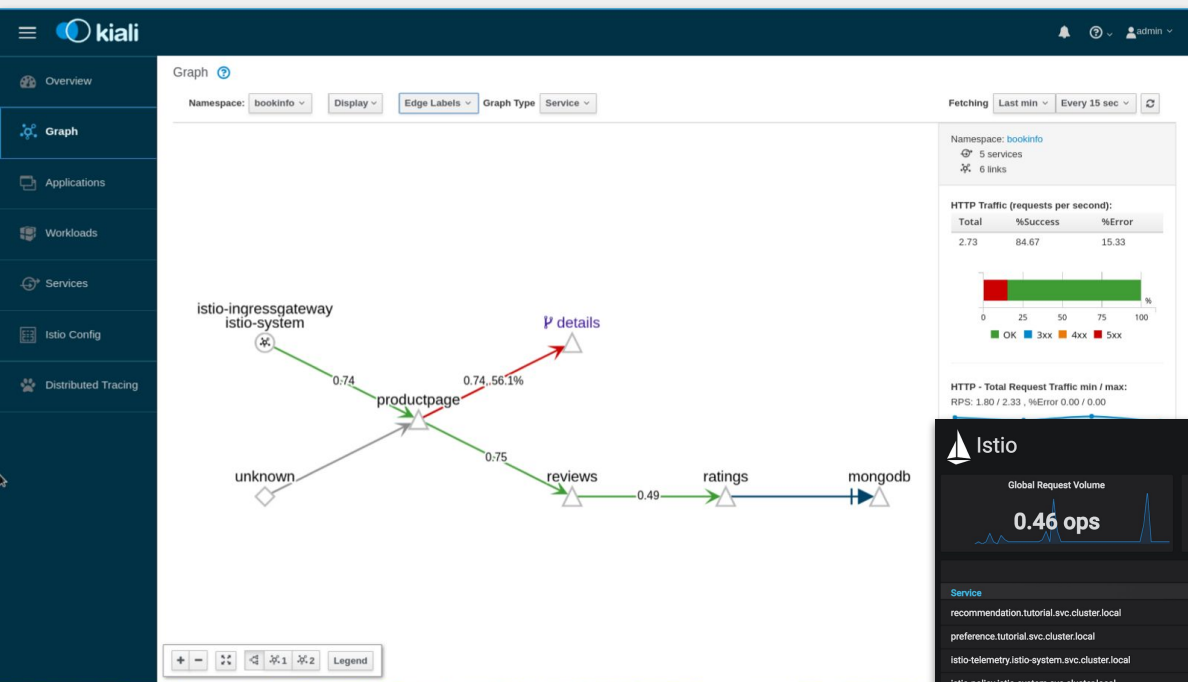
- name: requestcount.quota.istio-system
- overrides:
 - dimensions:
 - destination: someservice
 - maxAmount: 2

Exempt if:

```
match(request.headers["cookie"], "user=*) == false
```

Set and Check Policy: Open-ended, connection limits, rate limits, simple denials, lists

HOW CAN YOU OBSERVE?



Understand how your services are operating: Metrics, tracing, network visibility

ISTIO AVAILABILITY

Istio 1.0!

- After over a year of work,
- ~200 developers
- Google, IBM, VMWare, Cisco, Red Hat, others...
- Adaptors for many monitoring systems

Istio on OpenShift

- Available in Dev Preview today (3.10)
- GA coming soon (4.1)

ISTIO ON OPENSIFT

Istio on OpenShift

- Available in Dev Preview today (3.10)
- GA coming soon (4.1)
- Istio is an “operator first product” (using Operator Framework)
 - <https://github.com/Maistra/istio-operator>
 - The operator manages the install (eventually updates)
 - Istio is delivered as *containers*, **not** RPMs

TRY IT YOURSELF

<https://learn.openshift.com/servicemesh/>