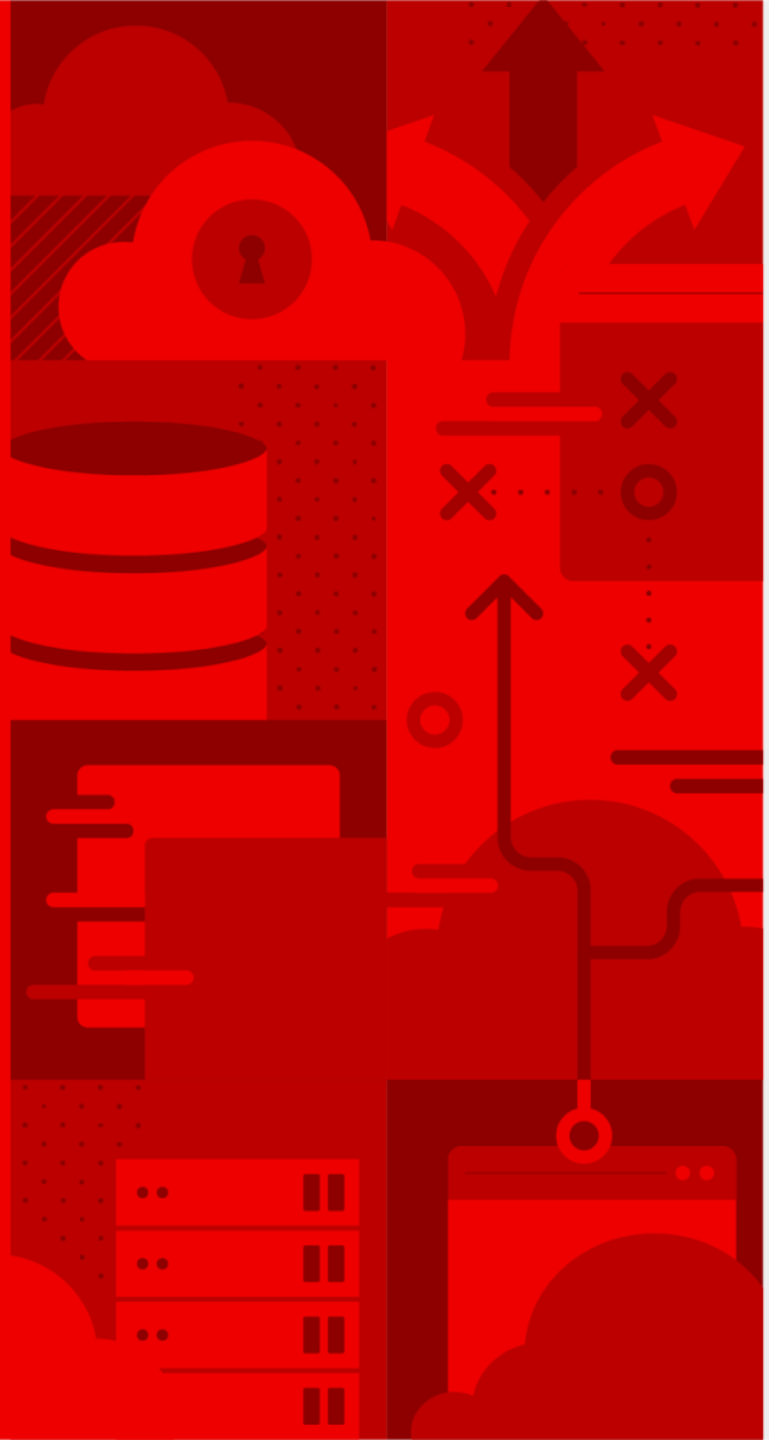


OPTIONAL SECTION MARKER OR TITLE

OpenShift Service Mesh Workshop

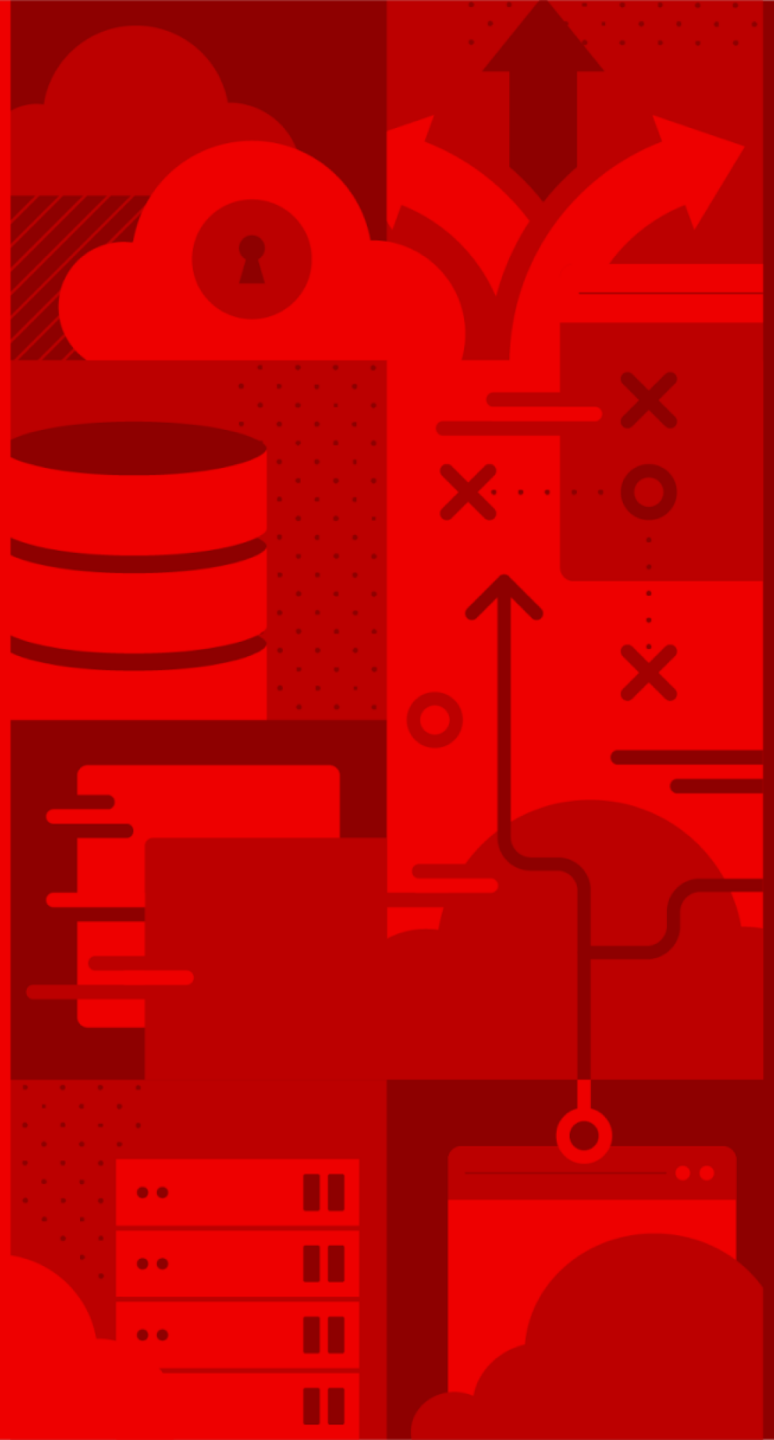
Dec 20, 2019

Voravit L
Senior Solution Architect

Red Hat

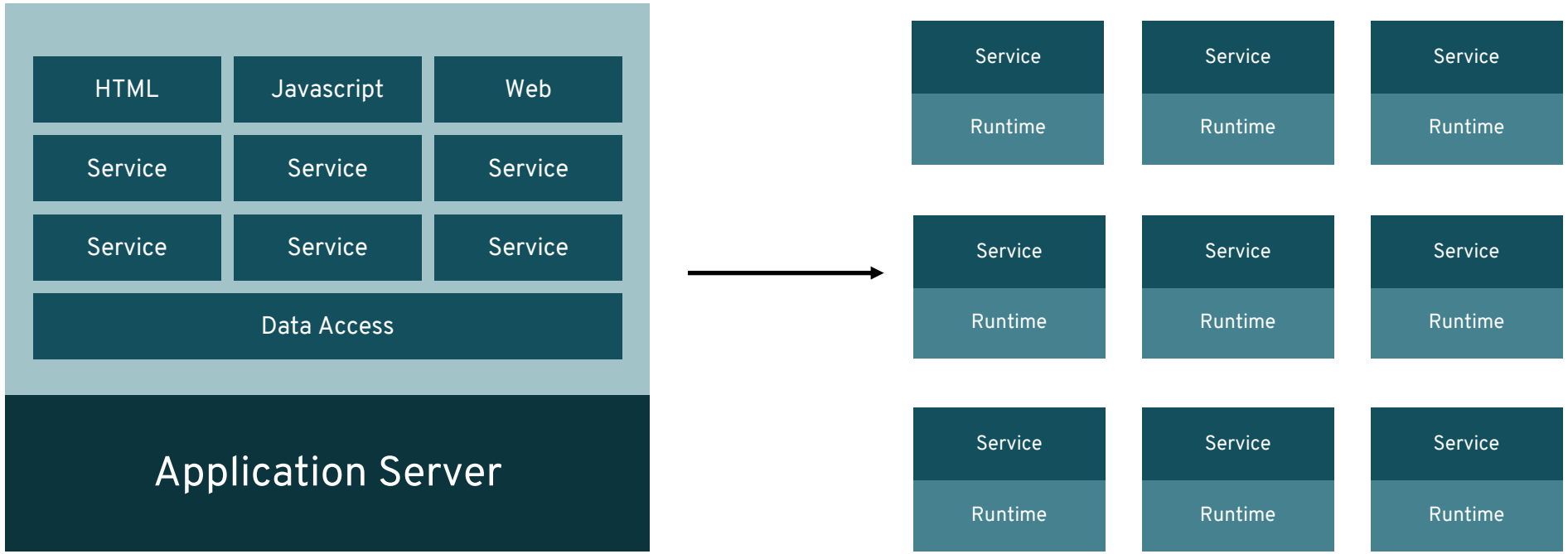
Agenda

- Describe the architecture of the Service Mesh and its components
- Labs
  - Lab 0:Describe and demonstrate how to Install the OpenShift Service Mesh using Operators
  - Lab 1: Configure Control Pane
  - Lab 2: Deploy Microservices
  - Lab 3: Observability
  - Lab 4: Traffic Management
  - Lab 5: Istio Gateway
  - Lab 6: Timeout
  - Lab 7: Circuit Breaker
  - Lab 8: Secure with mTLS



Microservices Benefits and Challenges

DISTRIBUTED
~~MICROSERVICES~~ ARCHITECTURE



Red Hat

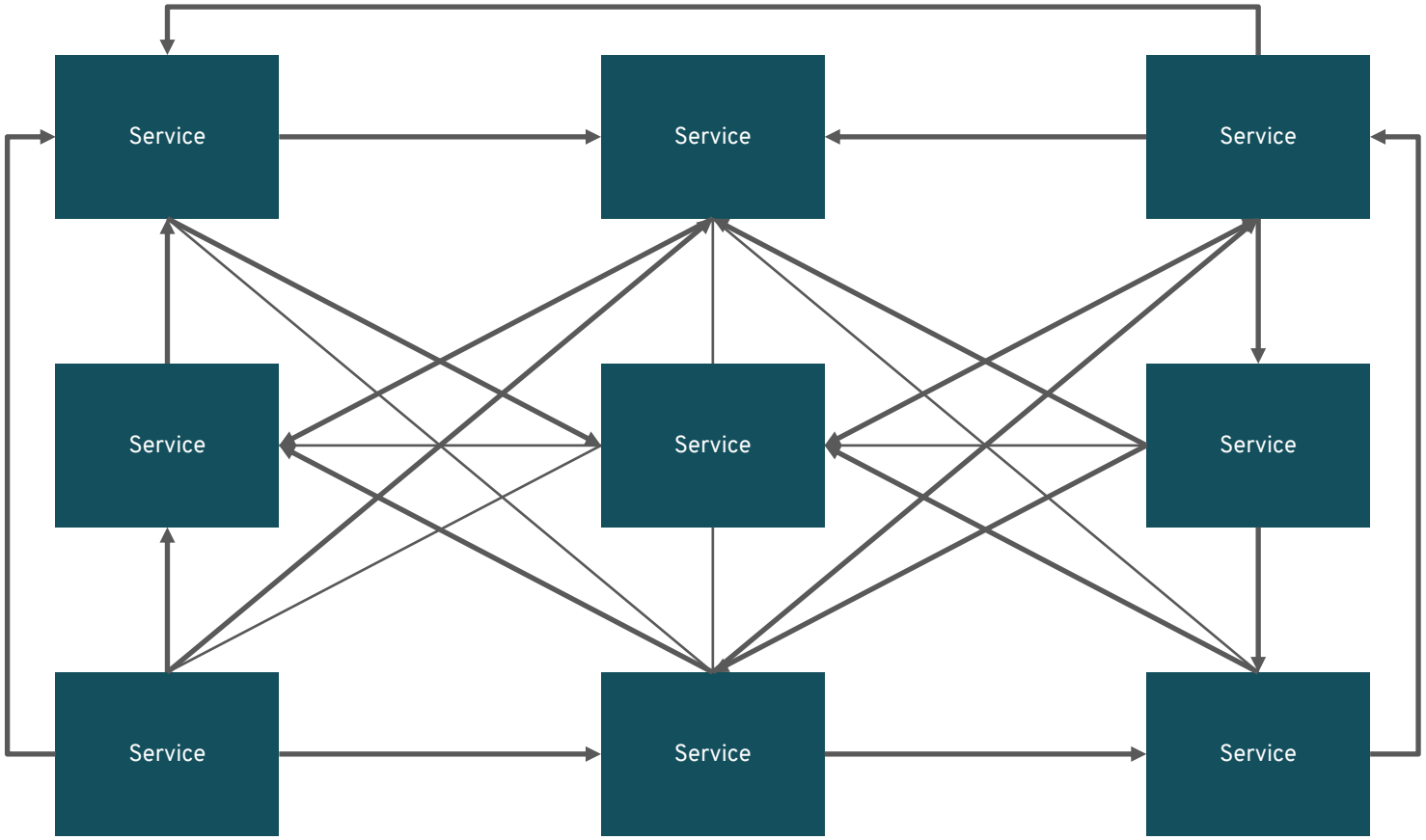
DISTRIBUTED COMPUTING CHALLENGES

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.

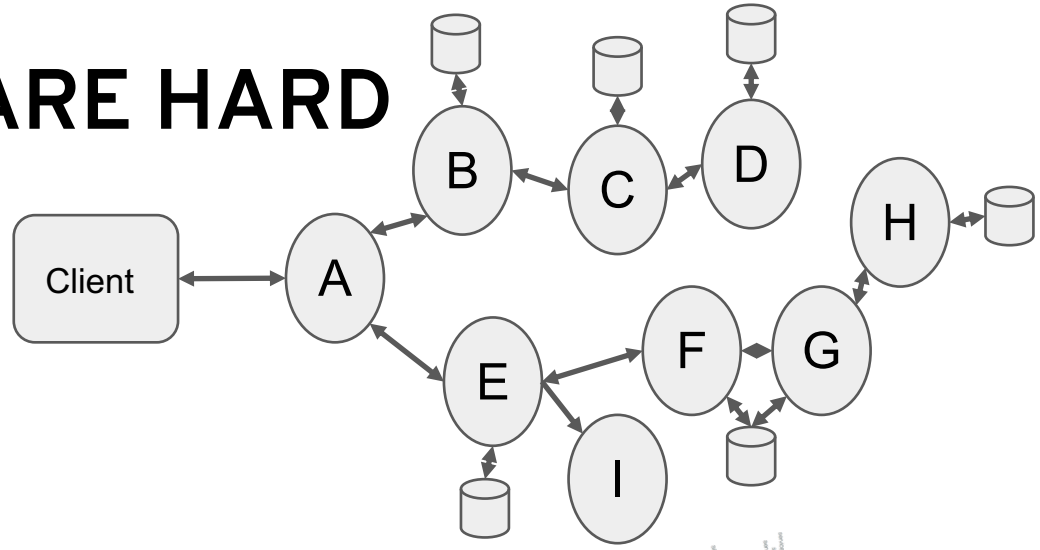
wikipedia.org/wiki/Fallacies_of_distributed_computing

DISTRIBUTED ARCHITECTURE



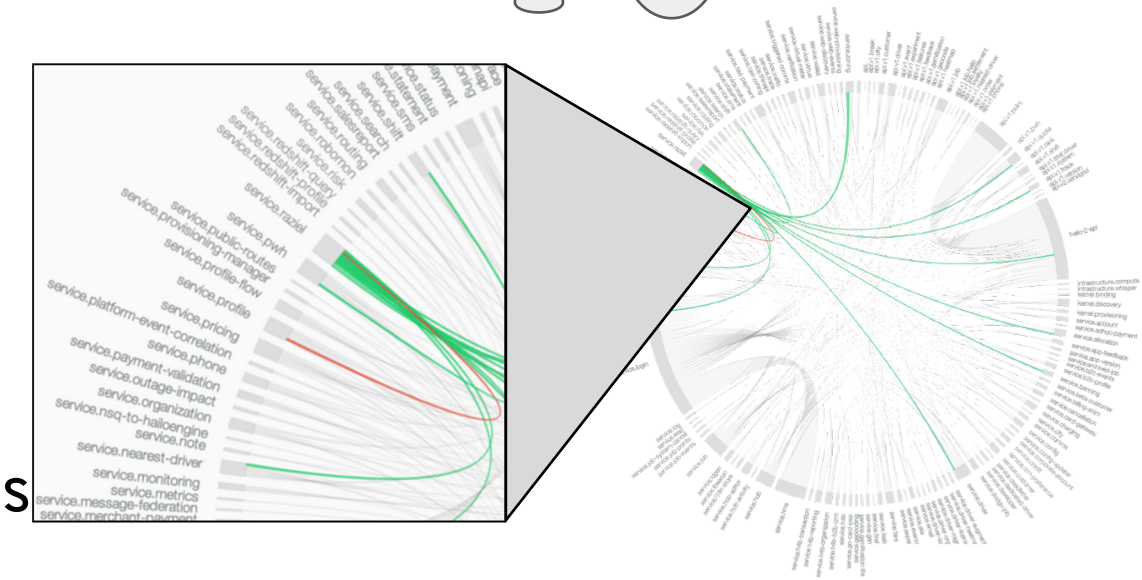
Red Hat

MICROSERVICES ARE HARD



Because applications must deal with

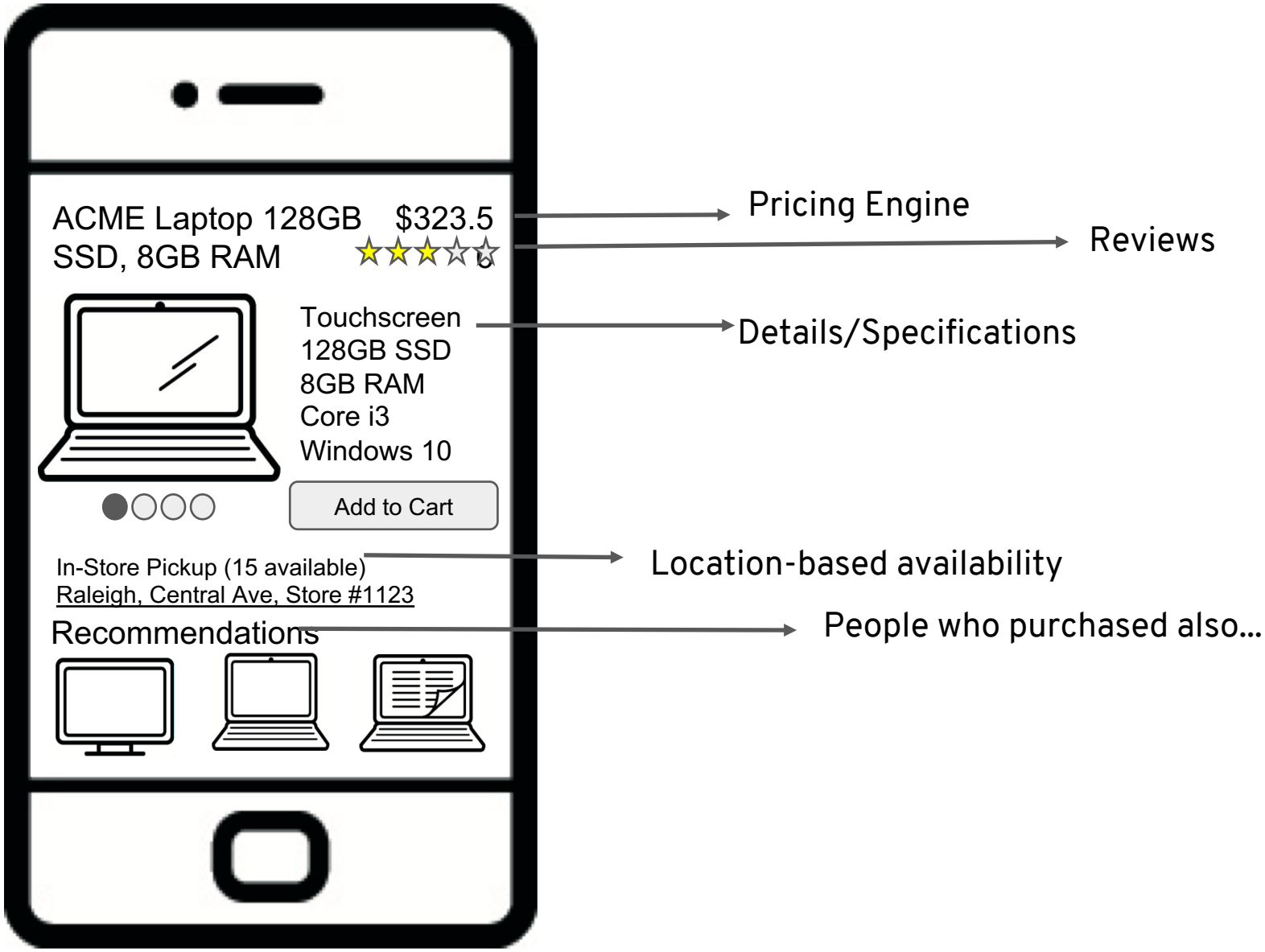
- Unpredictable failures
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources
- Distributed logs



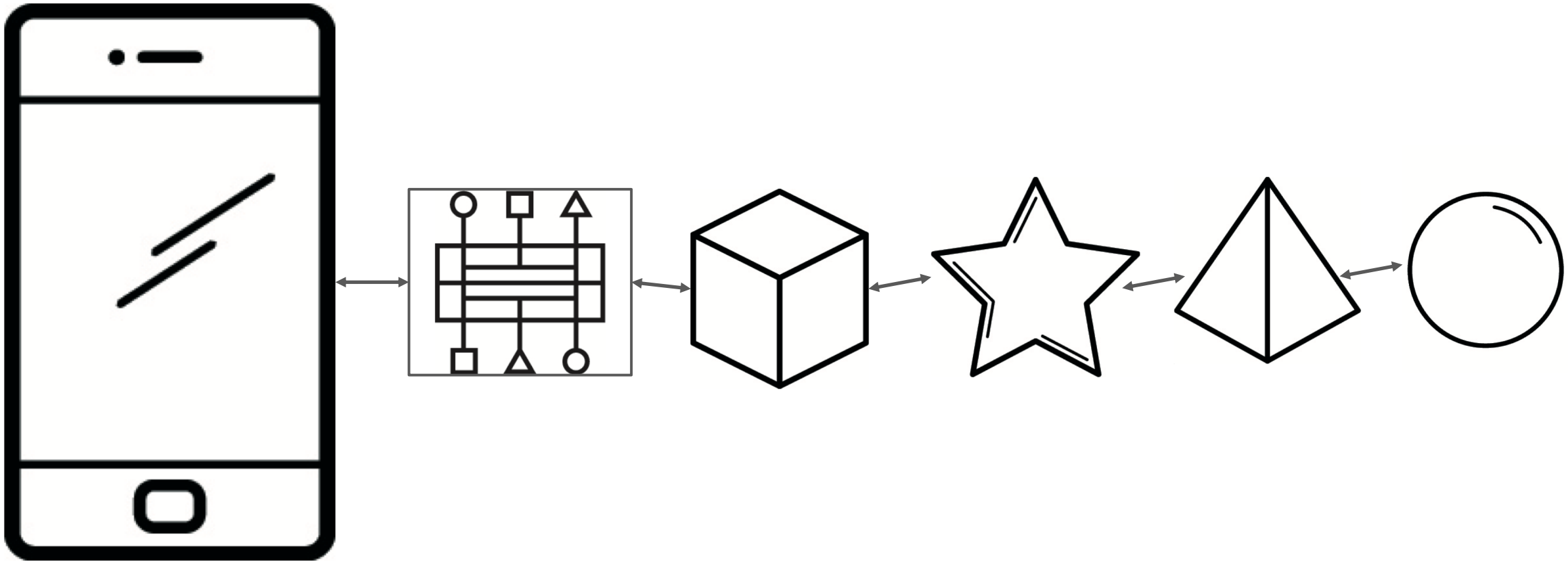
The fallacies of distributed computing

Red Hat

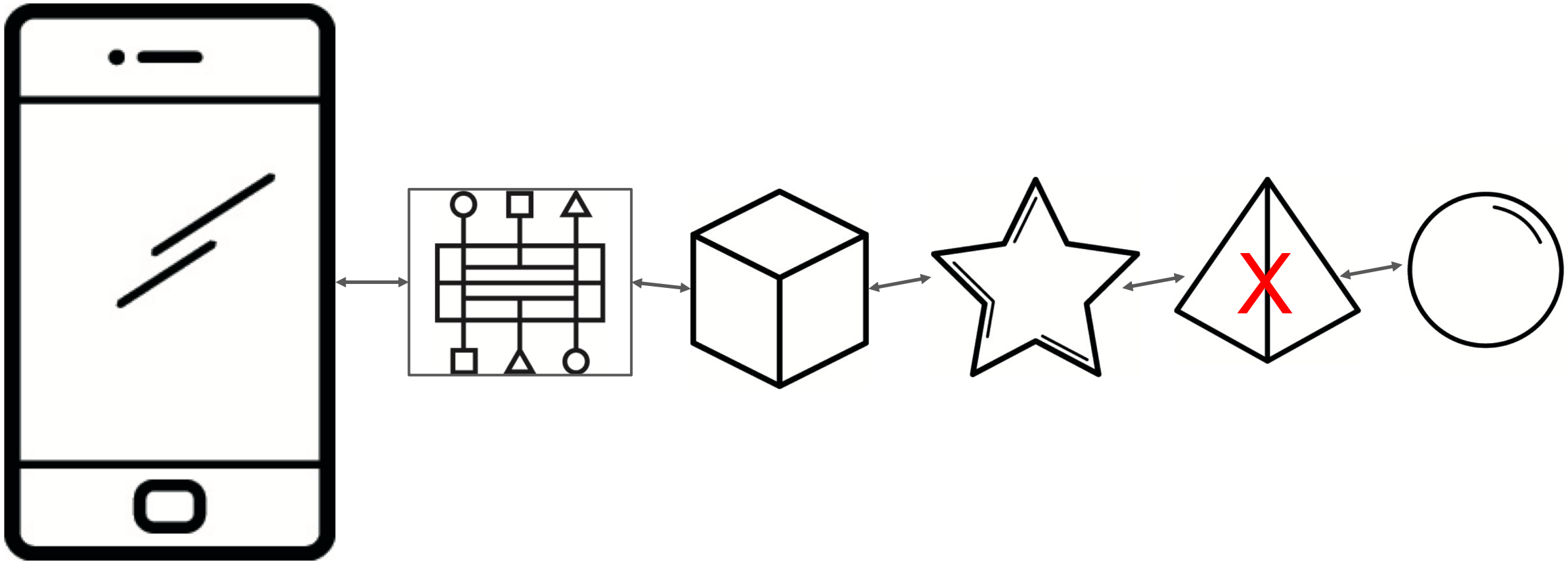
An Example



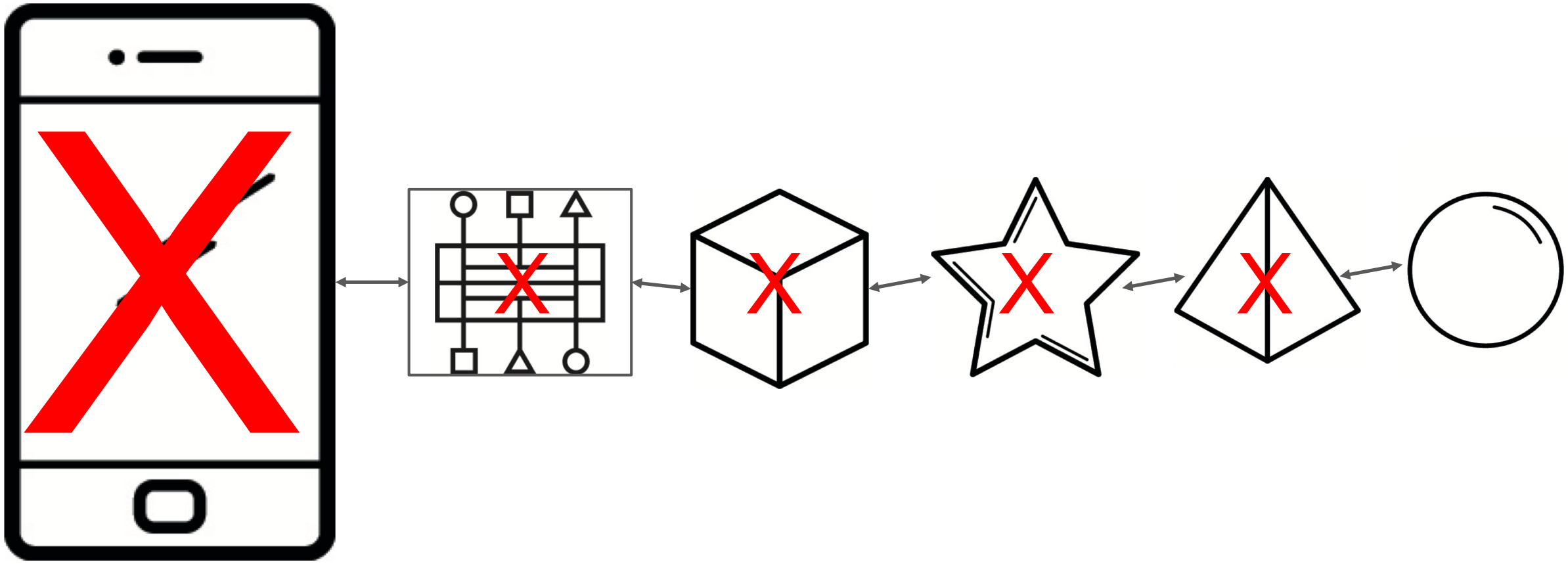
Chaining

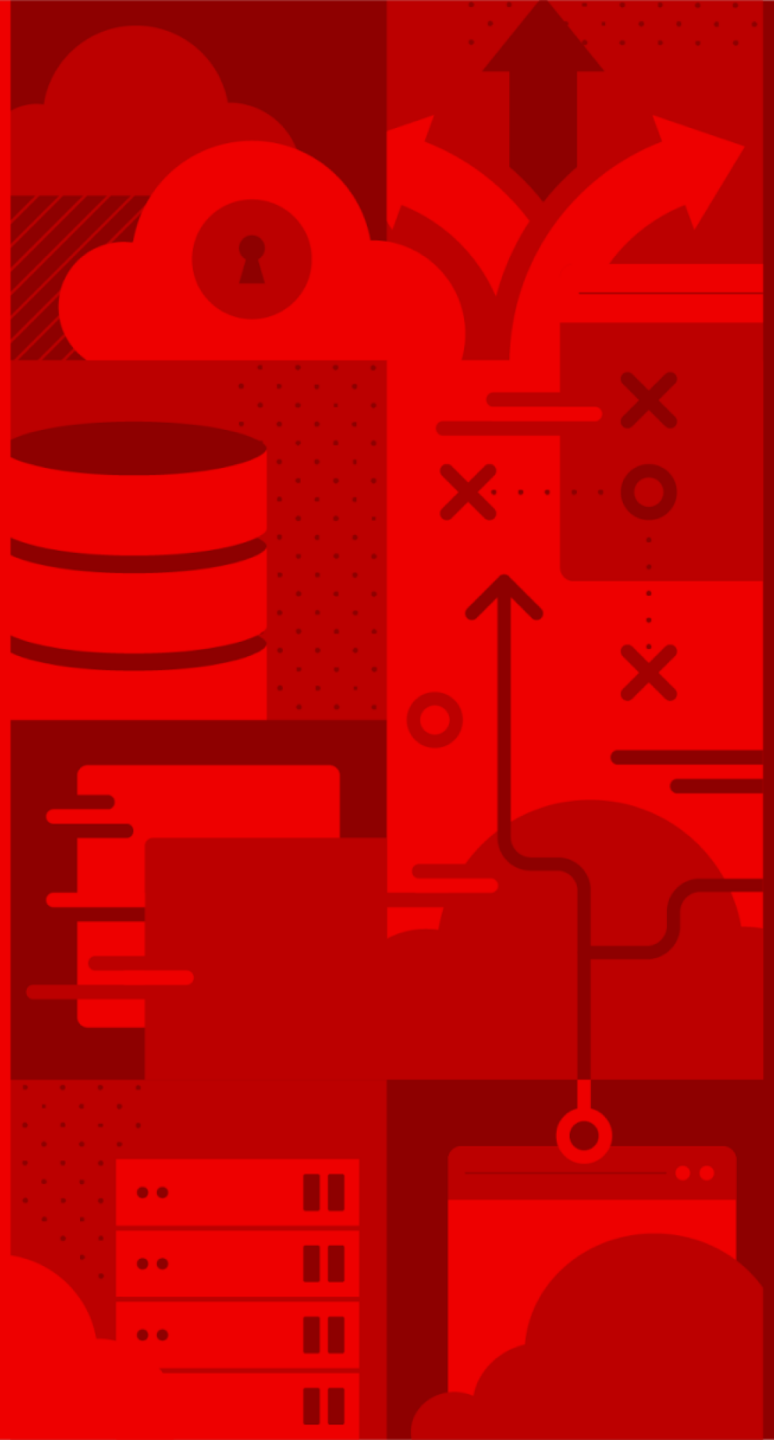


Chaining (Failure)



Chaining (Cascading Failure)



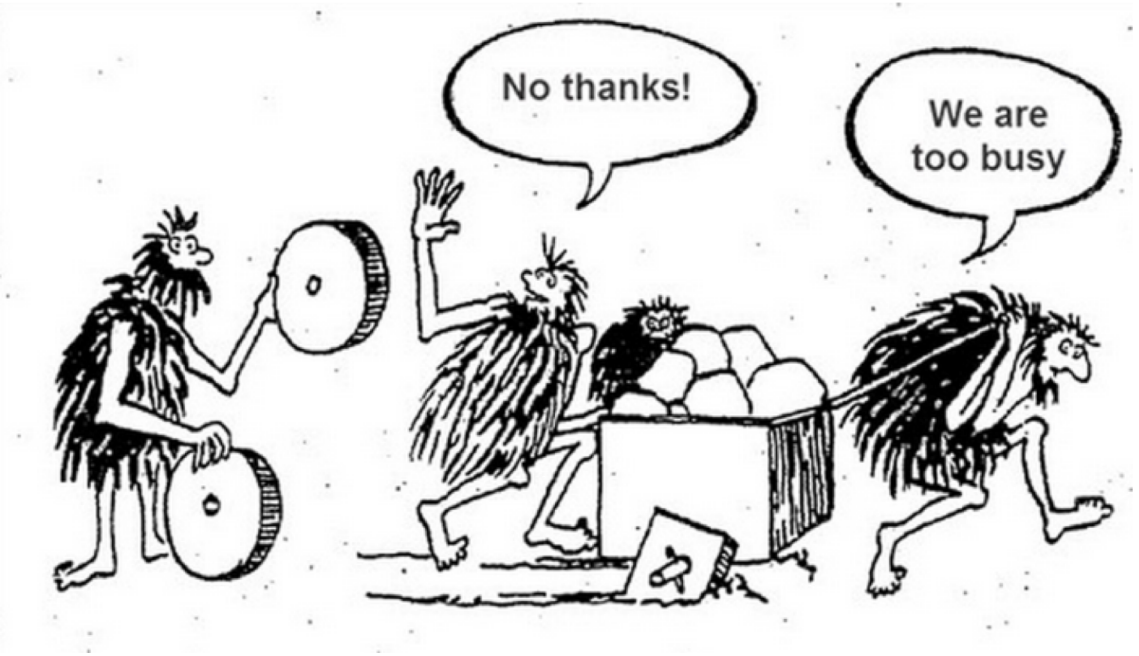


Traditional Approaches

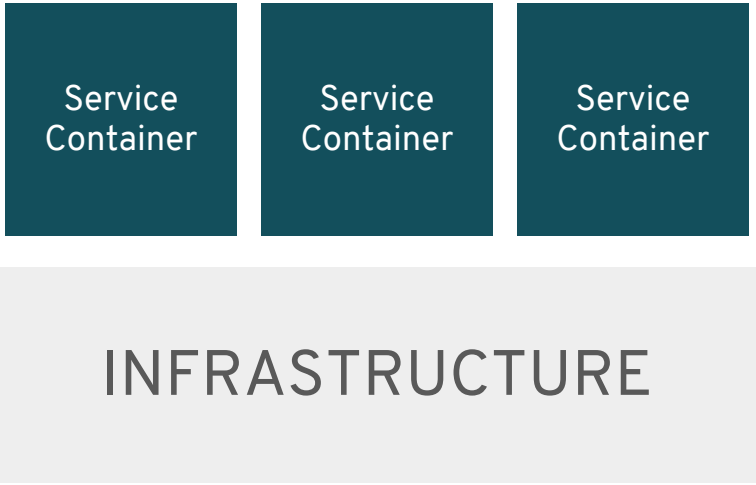
POSSIBLE SOLUTIONS

Have your developers do this:

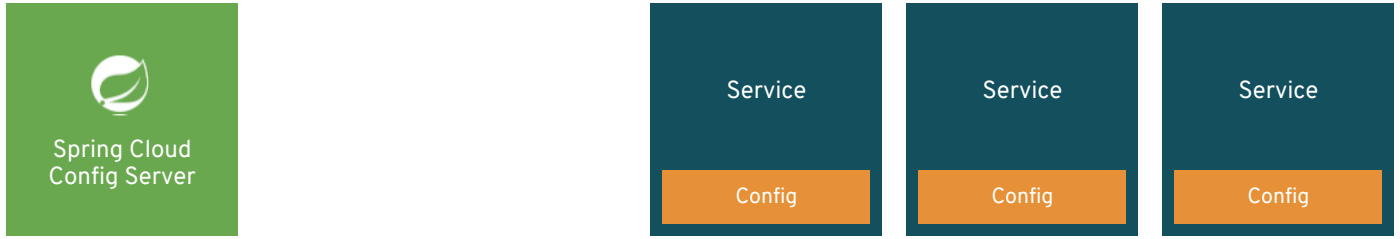
- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Load Balancing
- Traffic Control



DEPLOYMENT



CONFIGURATION



INFRASTRUCTURE

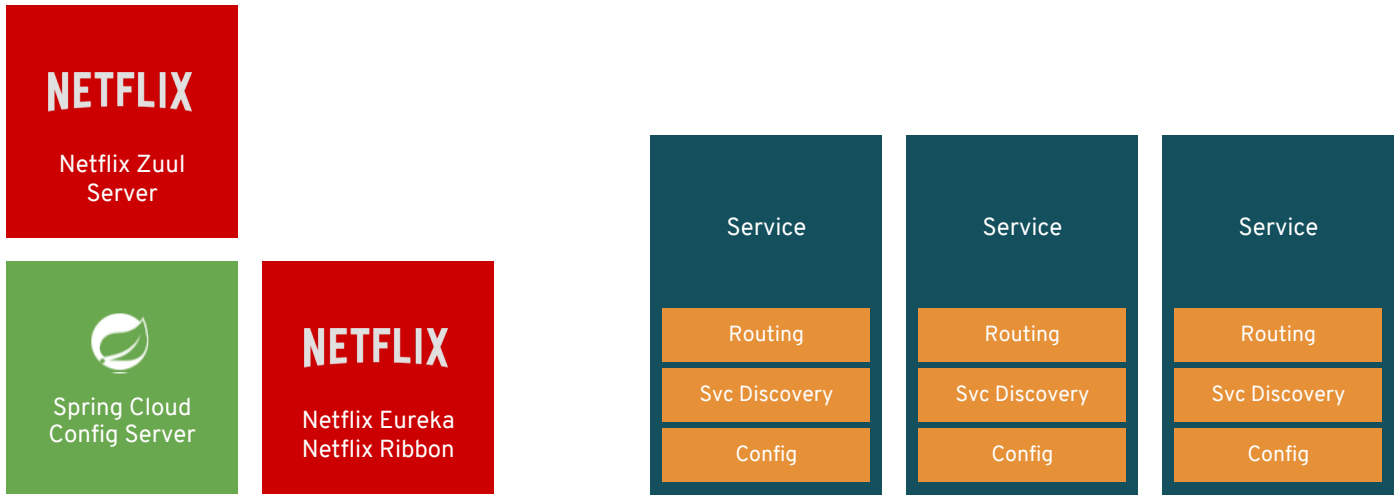
Red Hat

SERVICE DISCOVERY

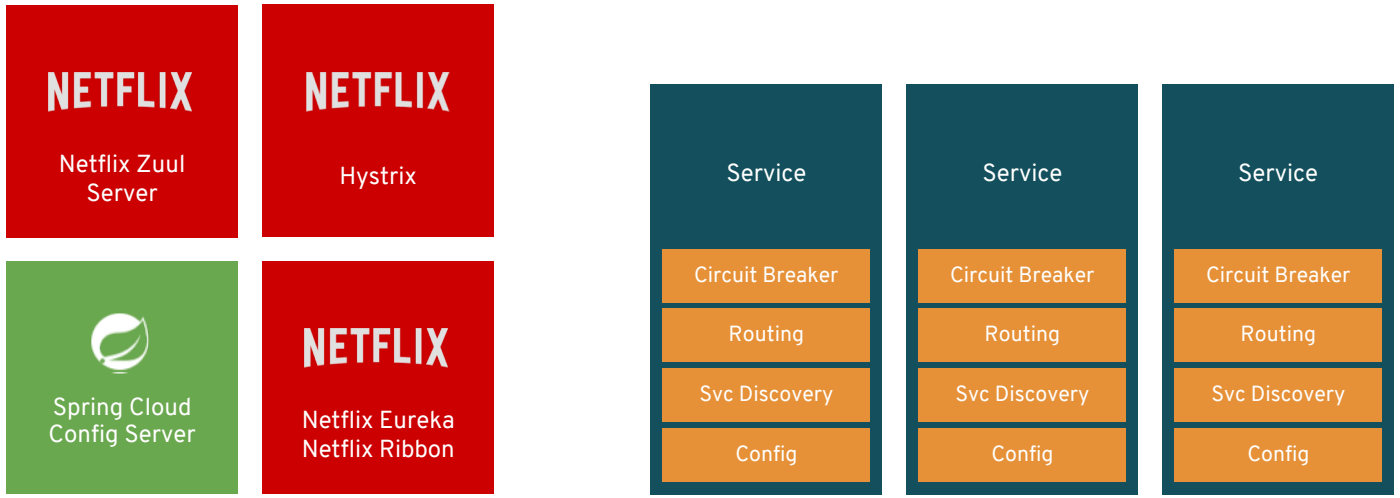
INFRASTRUCTURE

Red Hat

DYNAMIC ROUTING



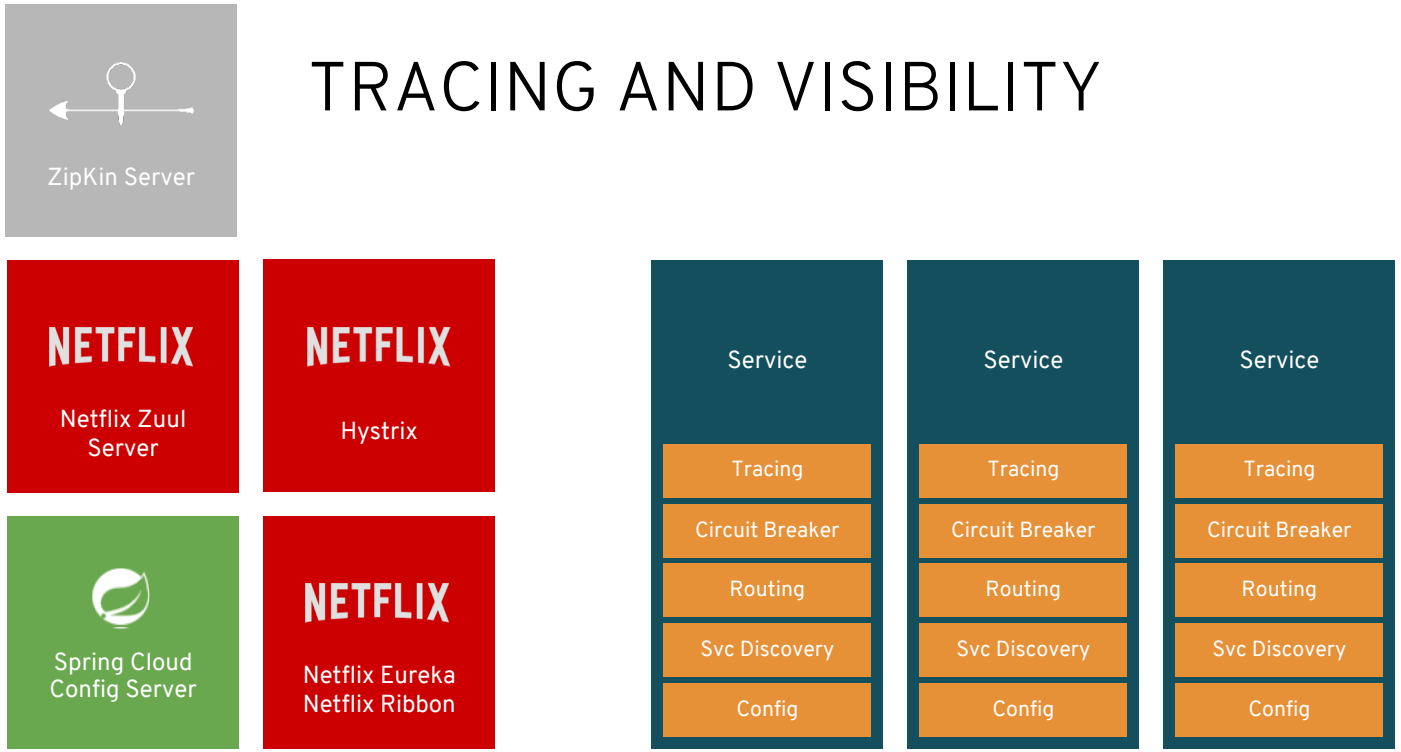
FAULT TOLERANCE



INFRASTRUCTURE

Red Hat

TRACING AND VISIBILITY

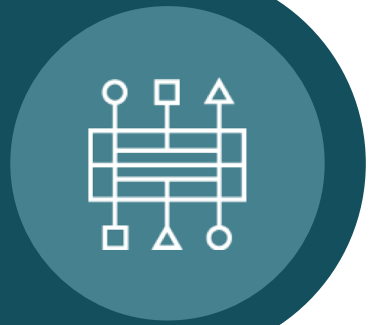


INFRASTRUCTURE

Red Hat

WHAT ABOUT...?

POLYGLOT APPS



EXISTING APPS

THERE SHOULD BE A BETTER WAY

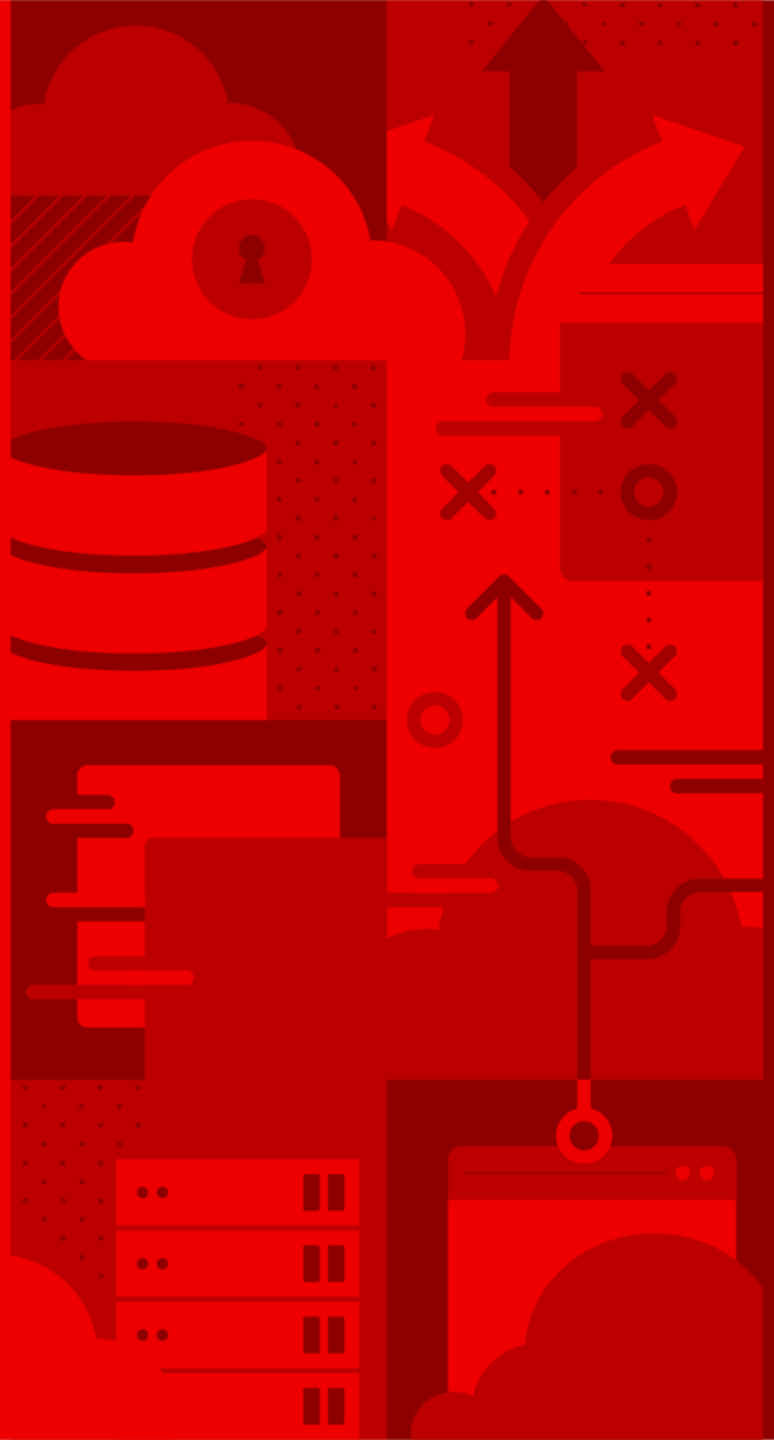
ADDRESS THE COMPLEXITY IN THE INFRASTRUCTURE

Kubernetes exacerbates the problem

The trends of containerization, microservices and hybrid/multi-cloud deployments have created more distributed applications than ever.

This has left enterprises unable to **connect, observe or secure or control** their services in a consistent way.

Red Hat

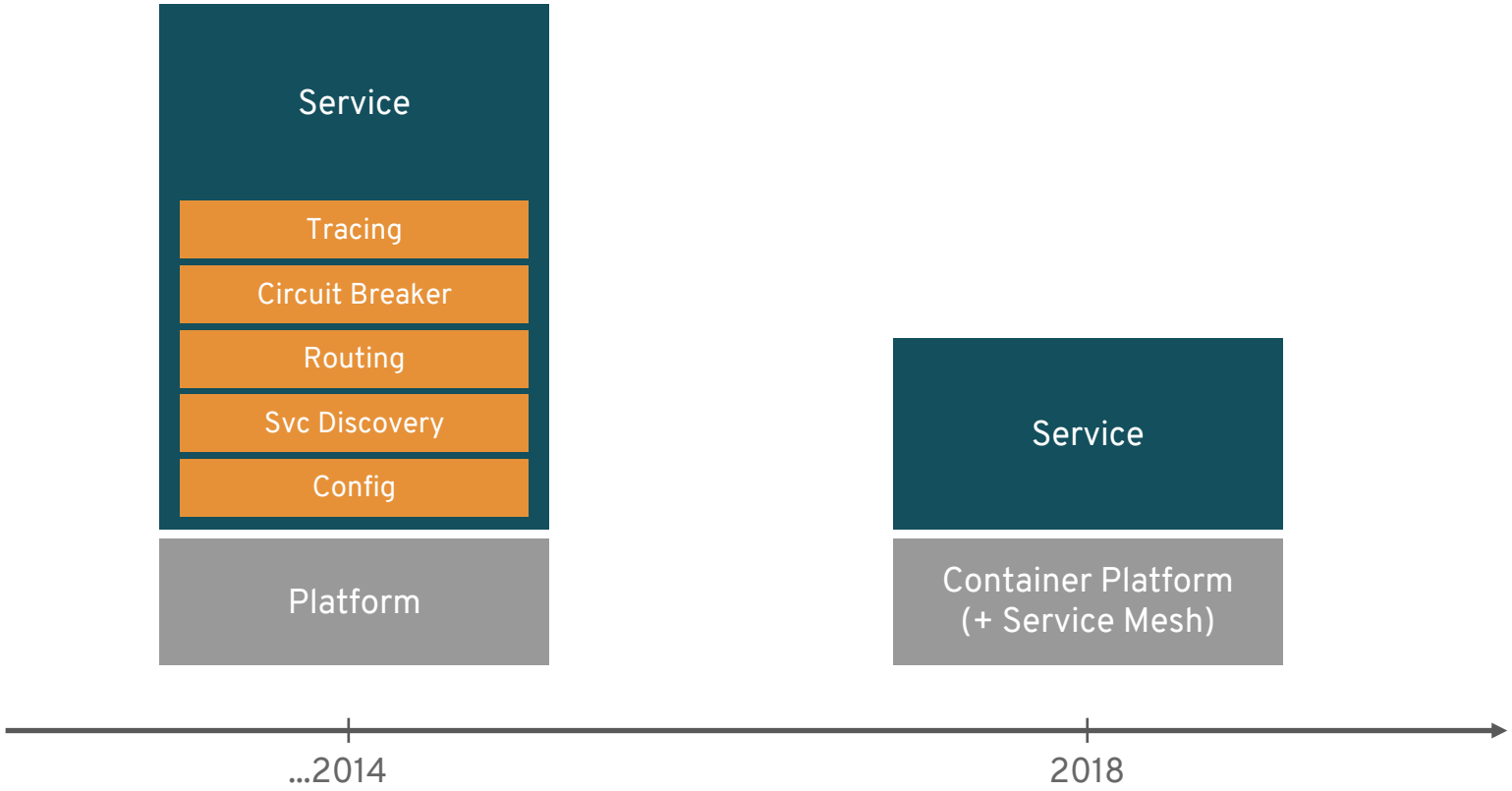


Enter the Service Mesh

SERVICE MESH

A dedicated network for service-to-service communications

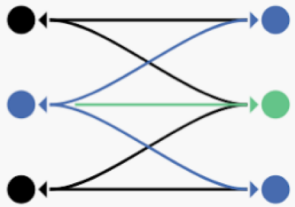
MICROSERVICES EVOLUTION



Red Hat

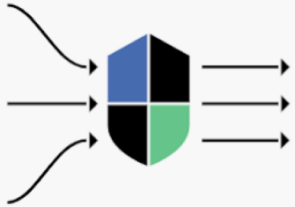
Istio

Connect, secure, control, and observe services.

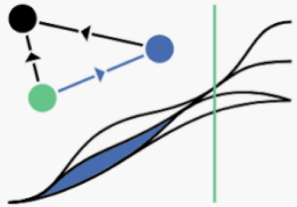


Connect

Secure



Control



Observe

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.

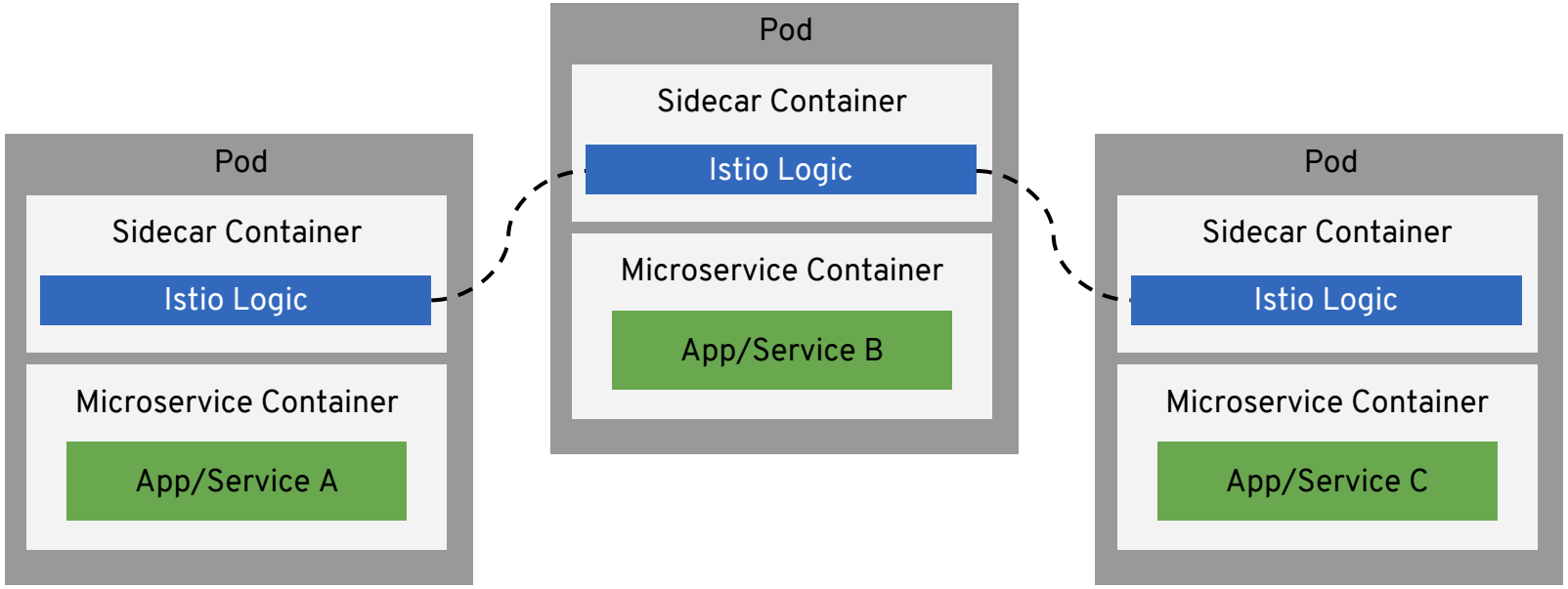
Service Identity and Security.
Enforce consistently across diverse protocols and runtimes with little or no application changes.

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*

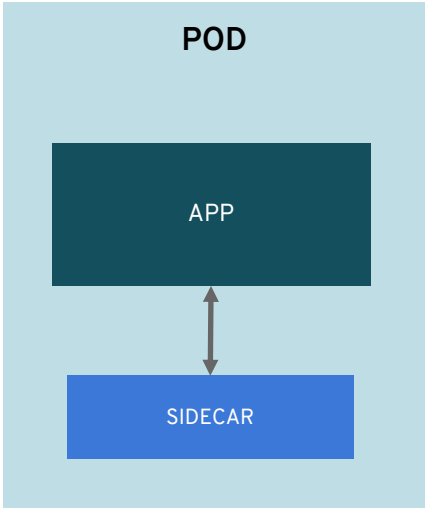


WHAT IS A SIDECAR?

A proxy instance that abstracts common logic away from individual services

SIDECAR PATTERN

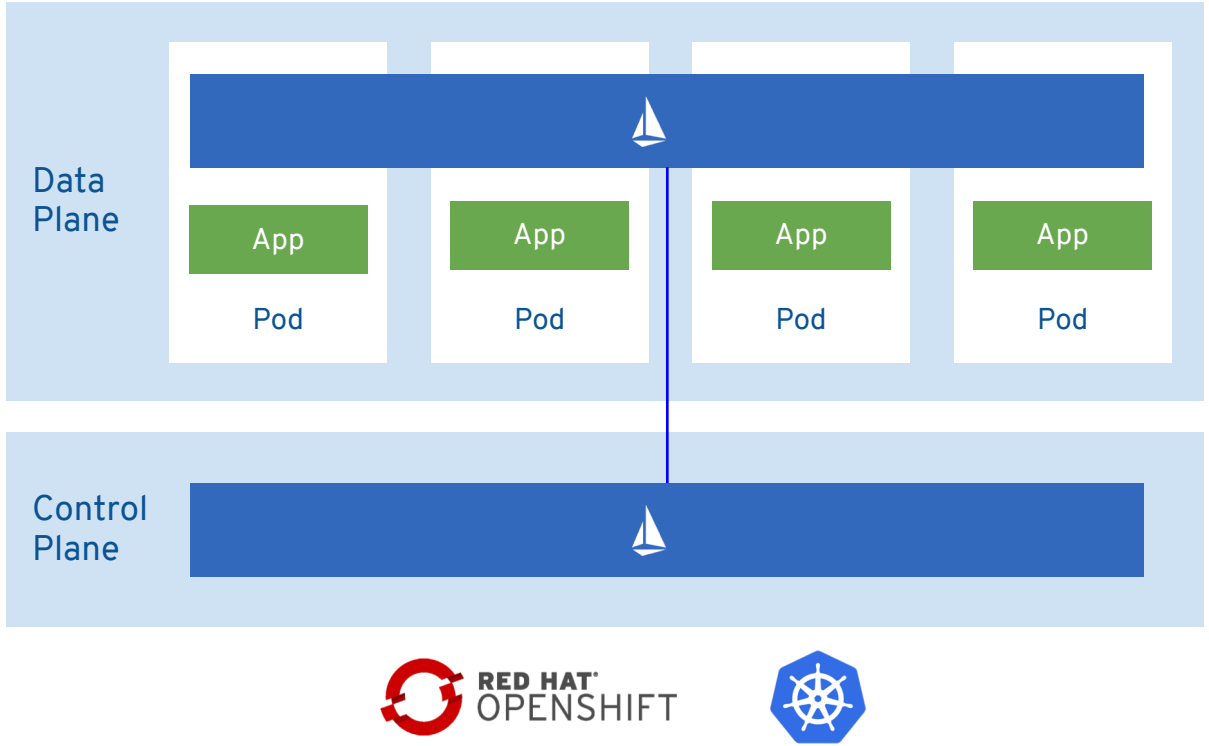
- A utility container in the same pod to enhance the main container's functionality
- Share the same network and lifecycle
- Istio uses an Istio Proxy (L7 Proxy) sidecar to proxy all network traffic between apps



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.



Red Hat

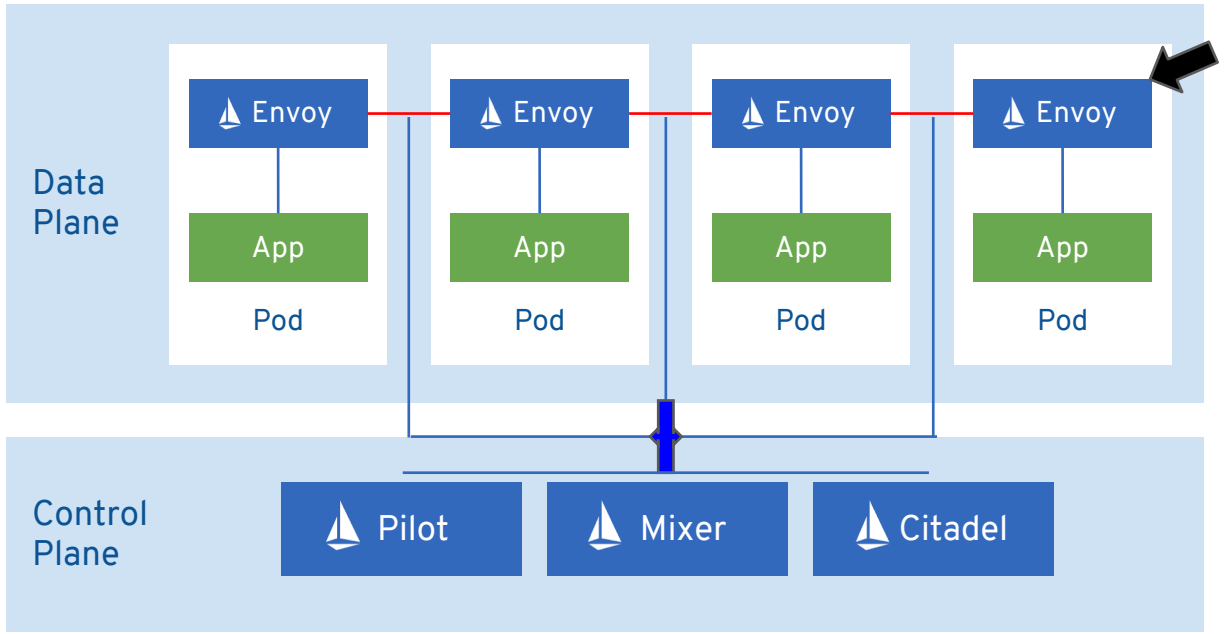
COMPONENTS OF ISTIO

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.

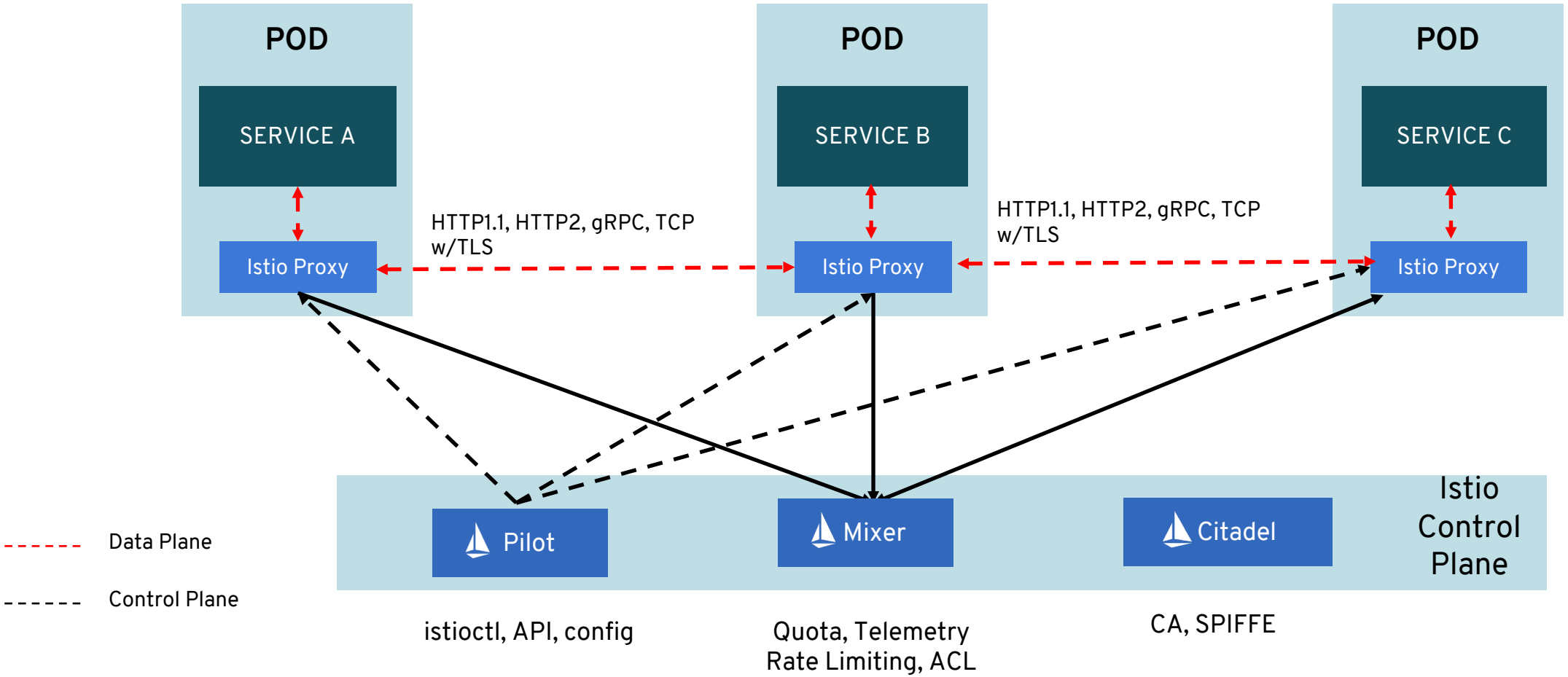


It's the sidecar

RED HAT OPENSHIFT

Red Hat

Istio Service Mesh - Control Plane & Data Plane



SERVICE MESH ECOSYSTEM

Jaeger

Secure

Observe

Istio

Connect

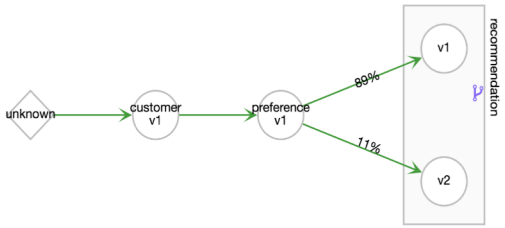
Control

kiali

Kiali

Prometheus

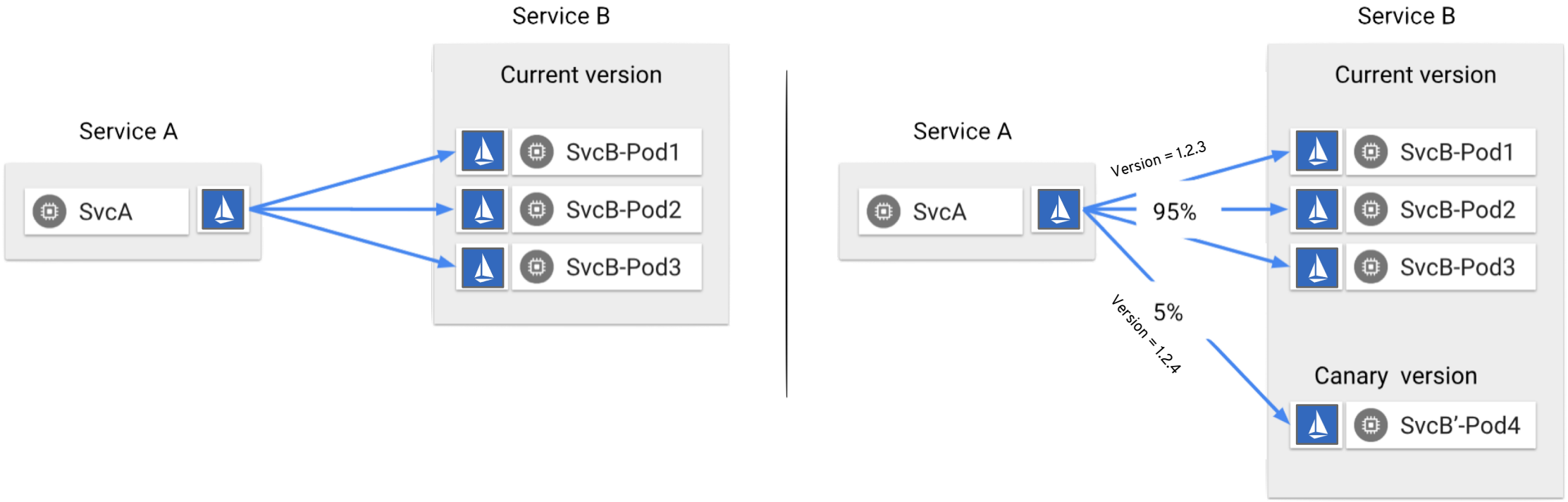
Observe



Grafana

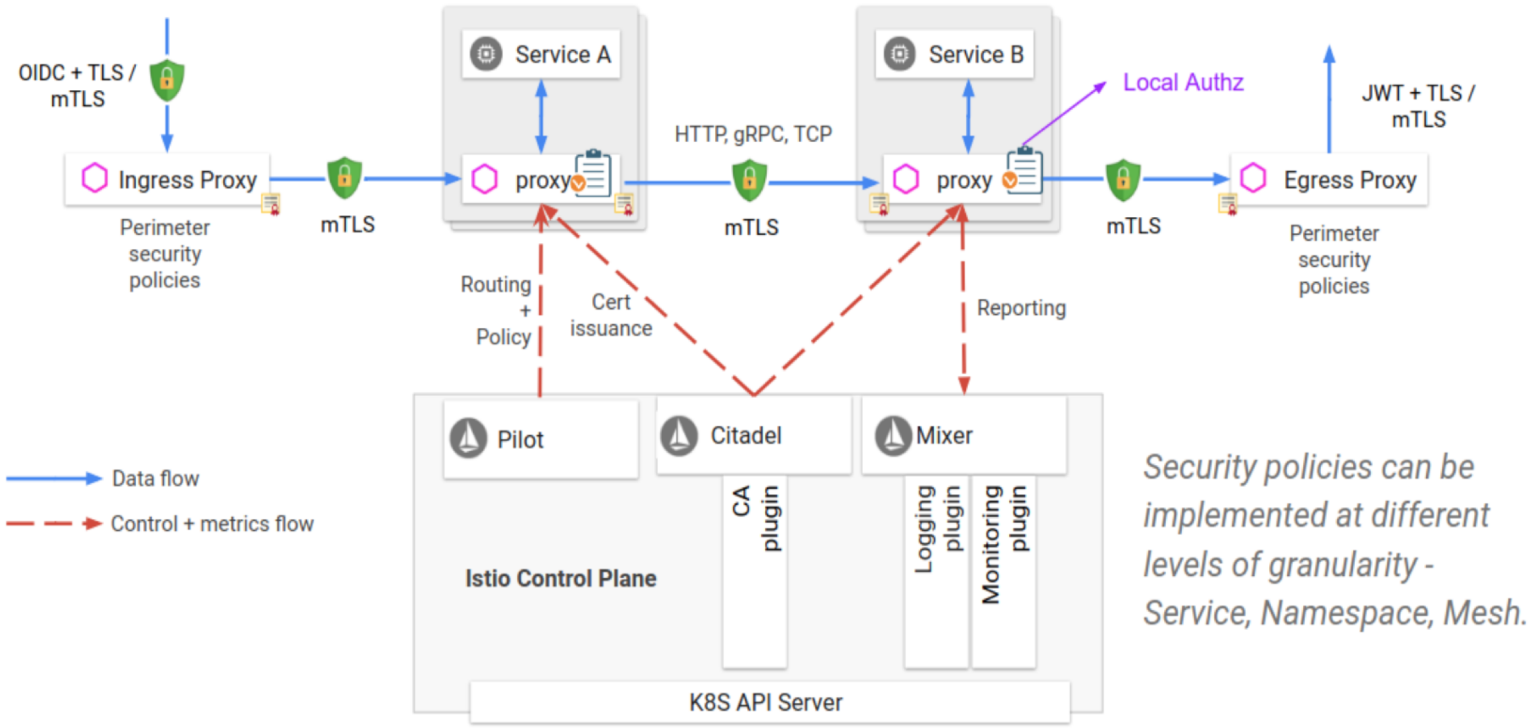
Red Hat

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...

WHAT DOES SECURE MEAN?

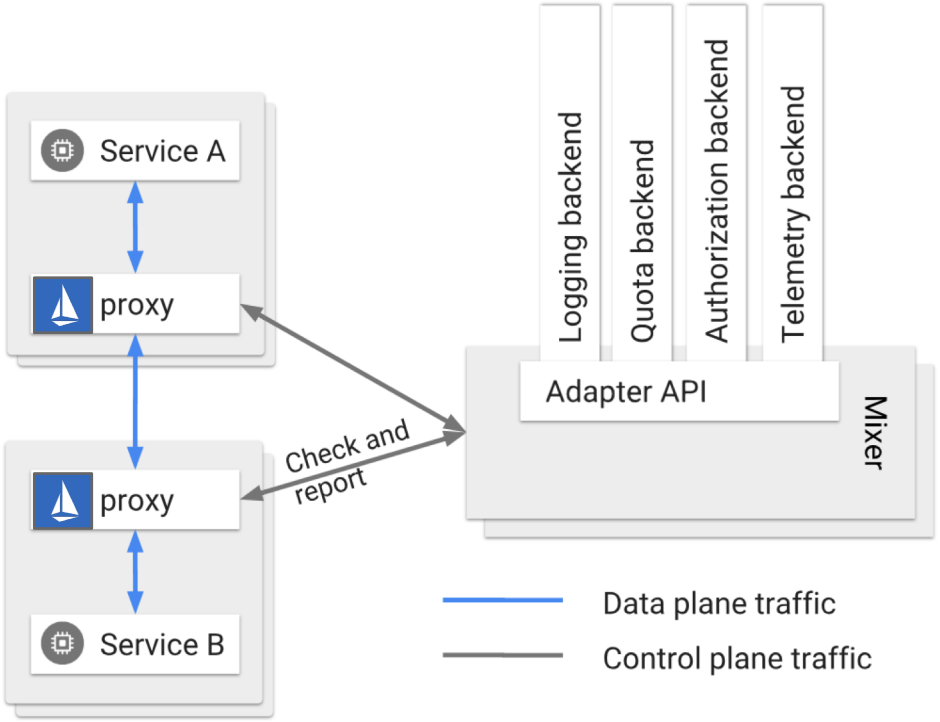


Security by default
no changes needed for application code and infrastructure

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

Zero-trust network
build security solutions on untrusted networks

WHAT DOES CONTROL MEAN?



Restrict to 2 requests per second per IP :

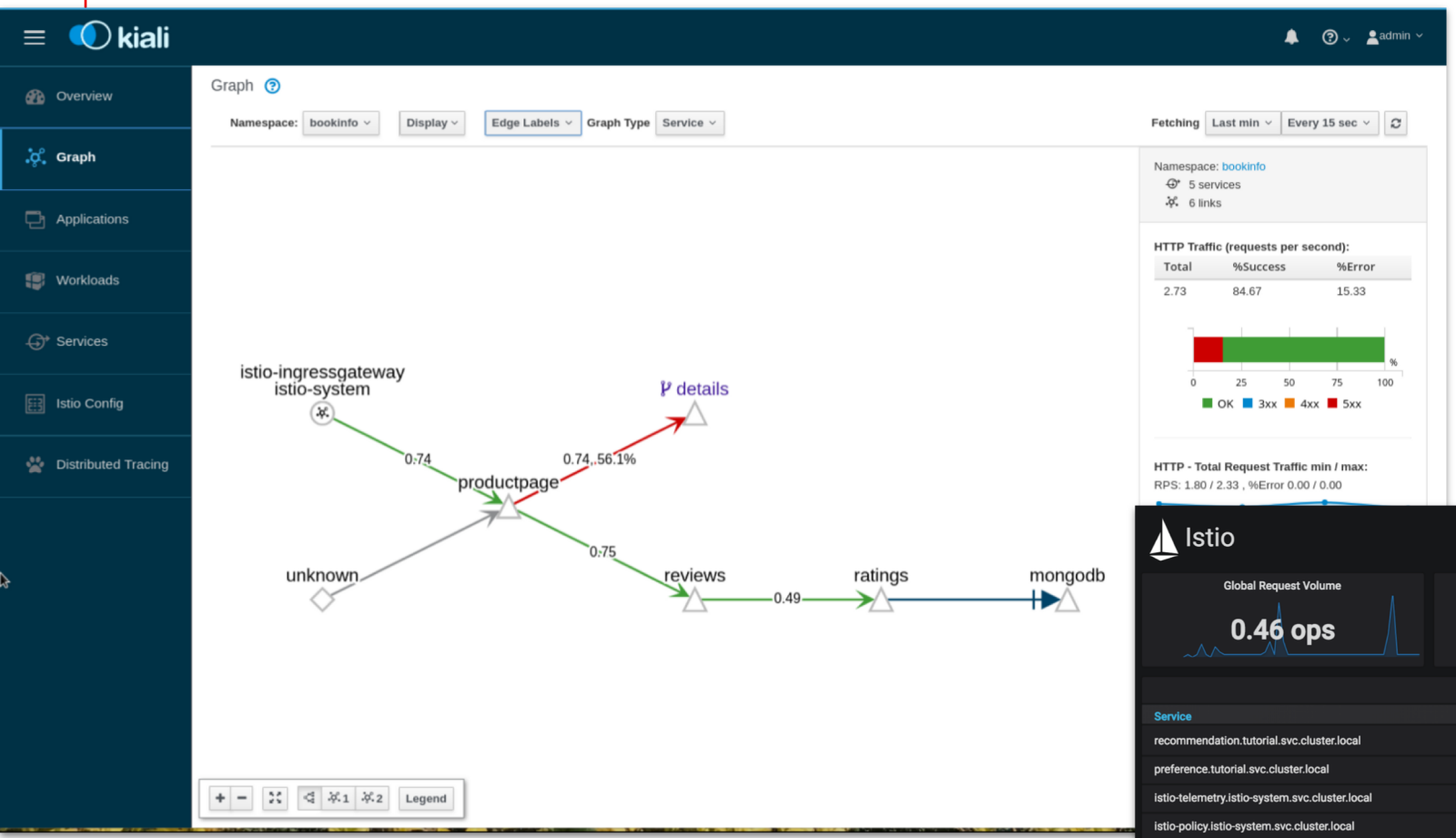
```
quotas:
  - name: requestcount.quota.istio-system
      overrides:
      - dimensions:
              destination: someservice
                    maxAmount: 2
```

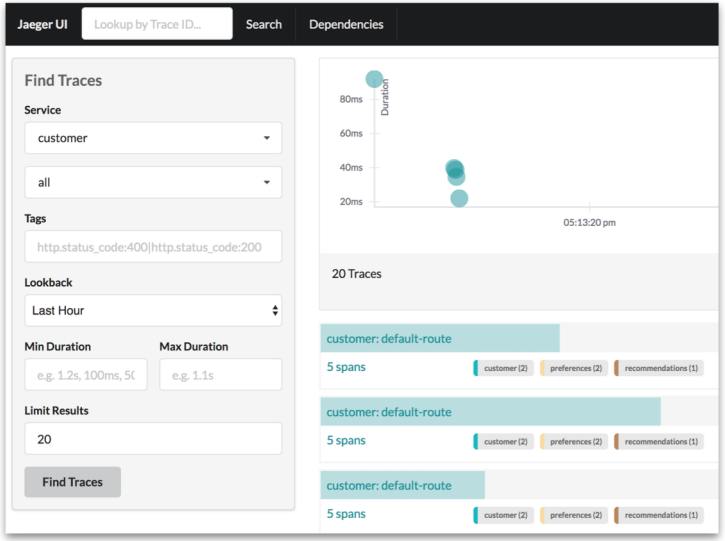
Exempt if:

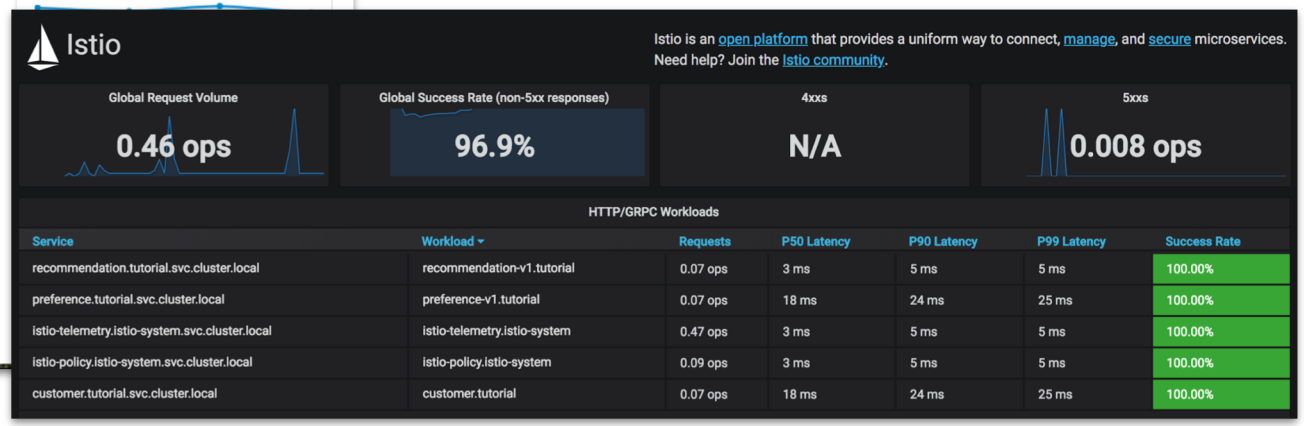
```
match(request.headers["cookie"], "user=*") == false
```

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

HOW CAN YOU OBSERVE?

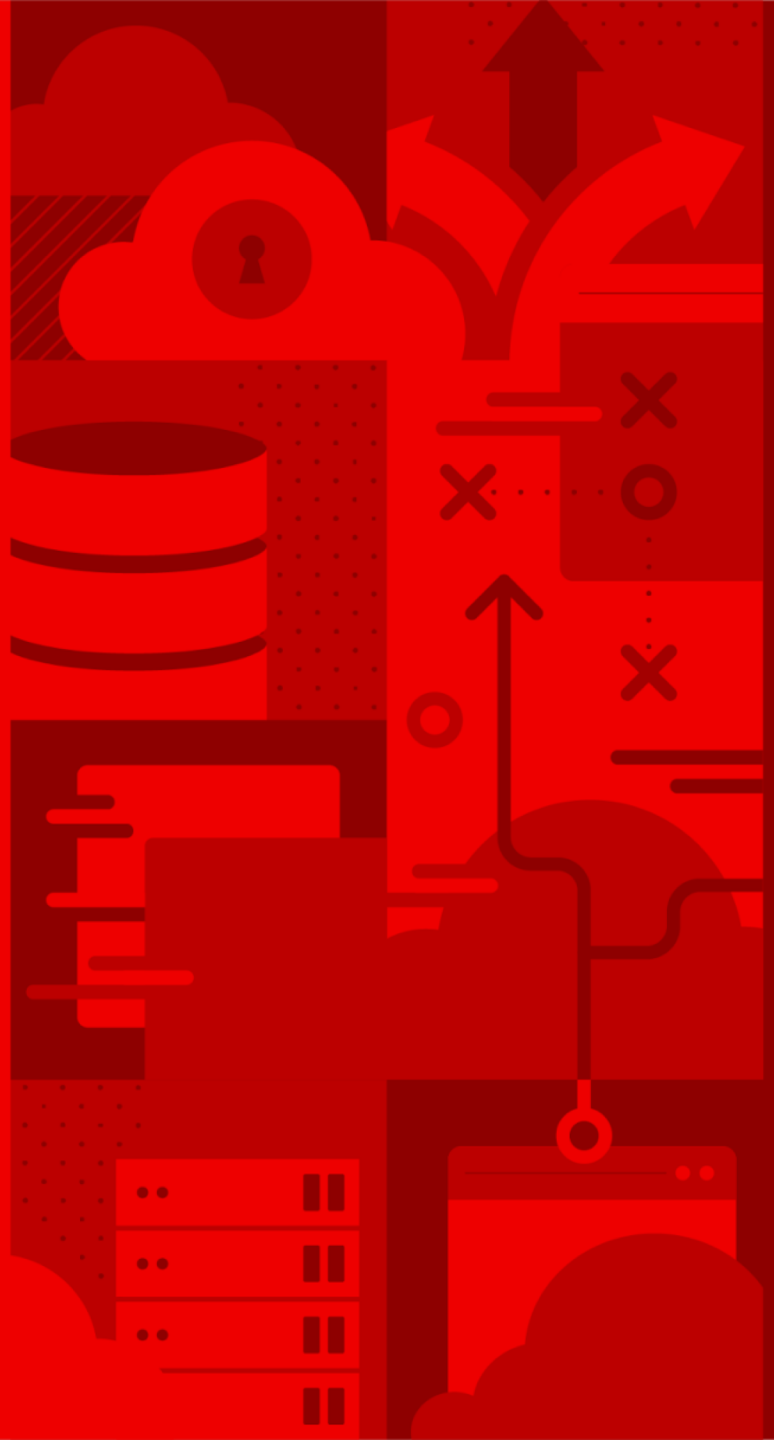






Understand how your services are operating: Metrics, tracing, network visibility

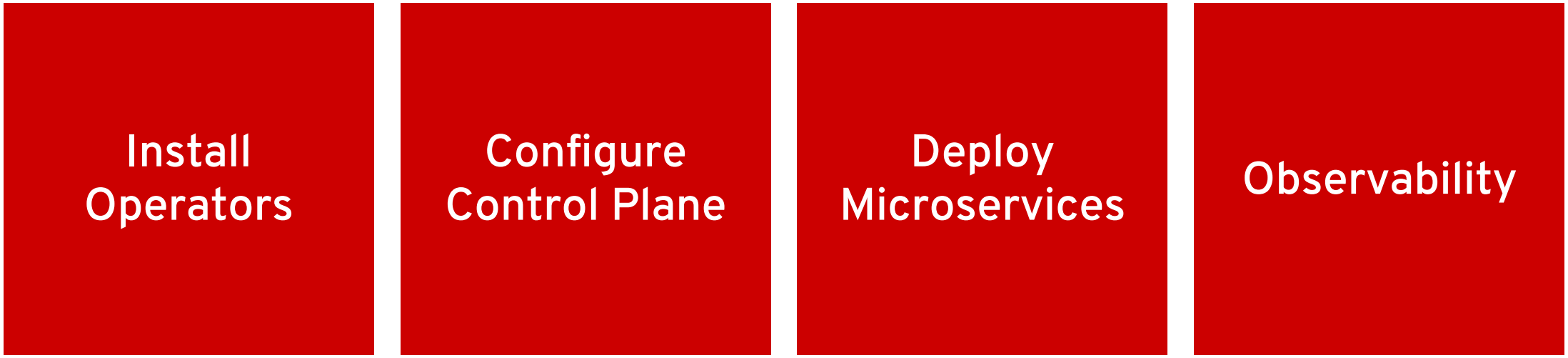
Red Hat



LAB

Red Hat

PART1: Install, Configure and Observability



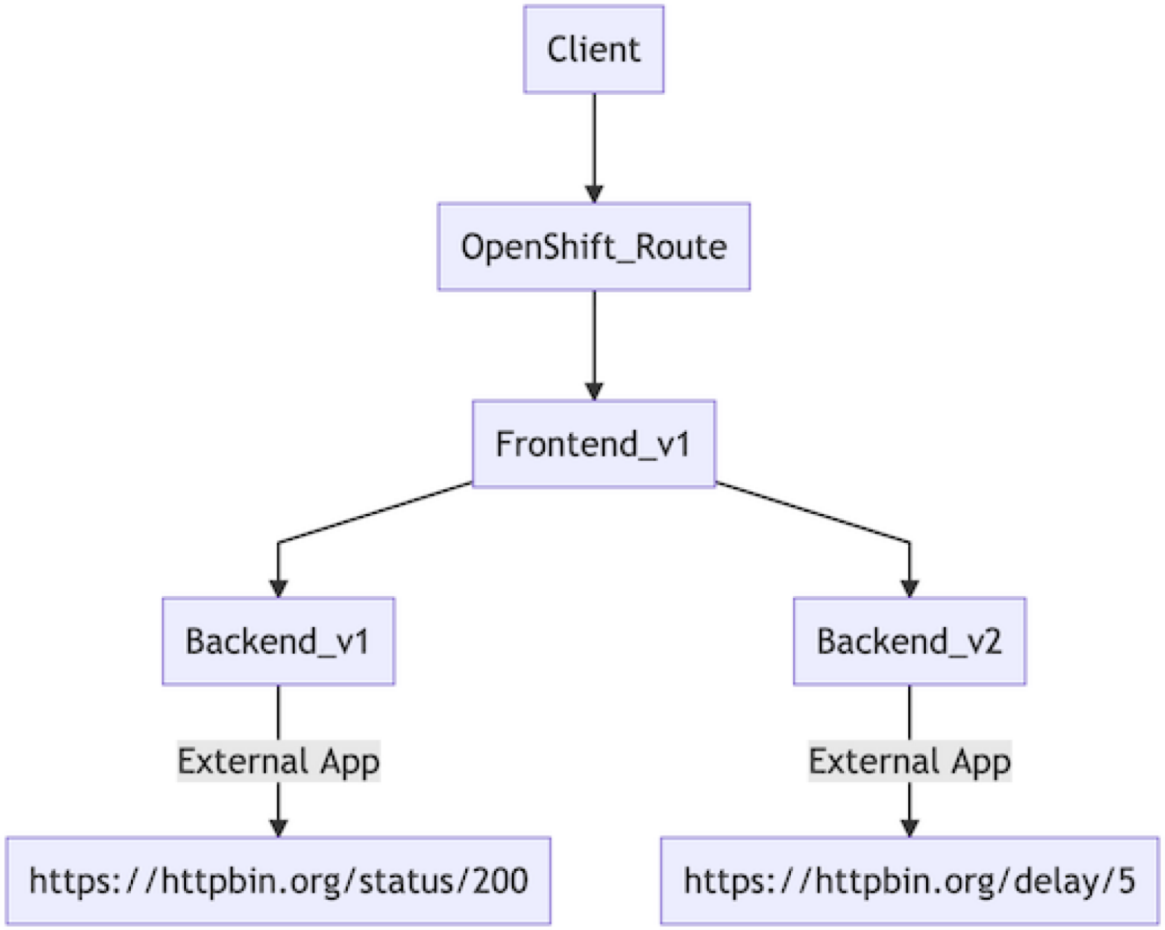
Red Hat

Installation & Configure Control Plane

OpenShift Service Mesh Installation steps including

- The Service Mesh installation process uses the OperatorHub to install the ServiceMeshControlPlane custom resource definition (CRD) within the openshift-operators project.
  - ElasticSearch
  - Jaeger
  - Kiali
  - OpenShift Service Mesh
- Create CRD in your Control Plane project.
  - ServiceMeshControlPlane
- Add project(s) into Service Mesh
  - ServiceMeshMemberRoll.

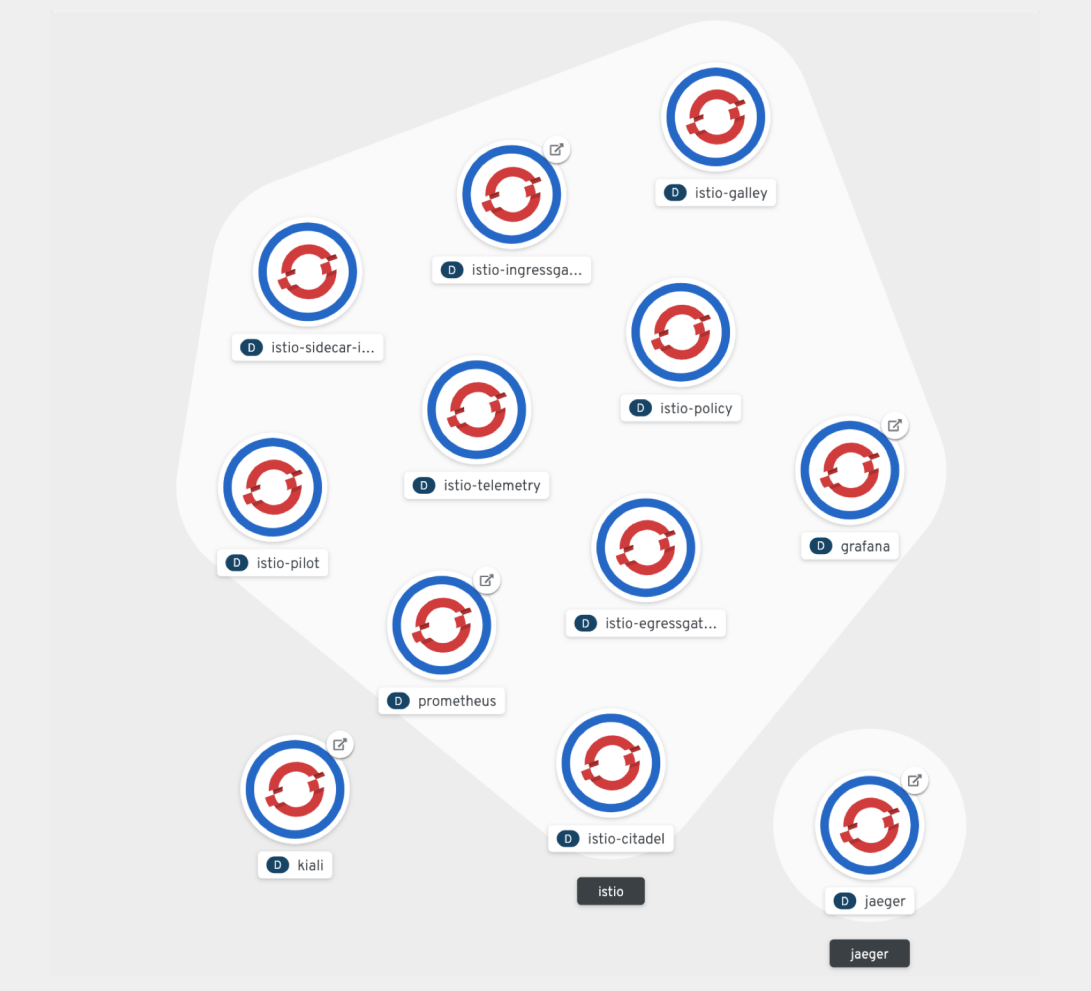
Deploy MicroServices



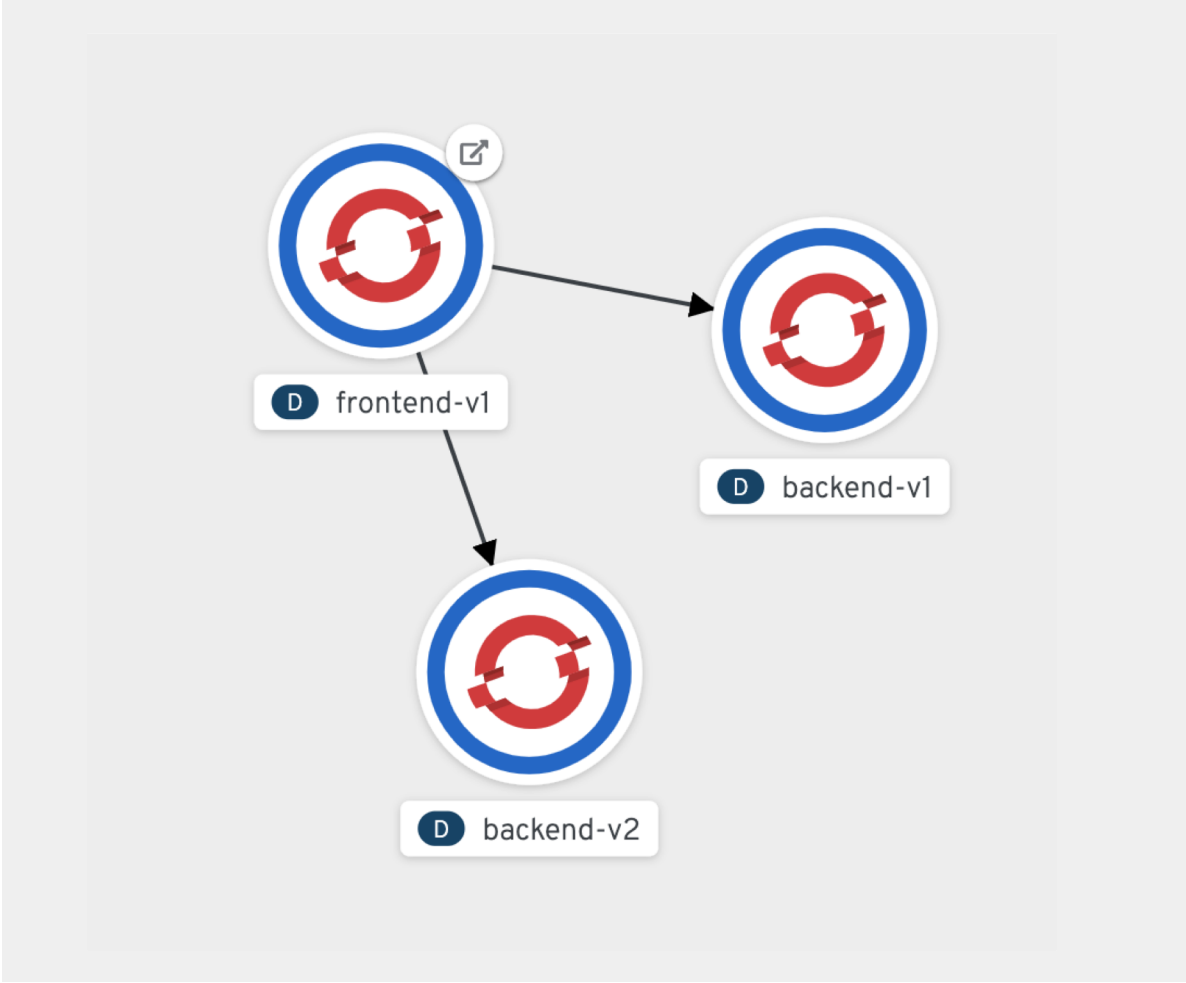
- 2 Applications frontend and backend
- Deploy everything with YAML
- Frontend and Backend app consumes environment variables for their configuration. (Except version)
- Backend use external website for generate response code and delay.
  - https://httpbin.org

Lab Environment

userX-istio-system



userX



Red Hat

PART2: Connect

Traffic Management

Istio Gateway

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

Destination Rule

- Define policies that applied to traffic after routing occurs
  - Load Balancing
  - Connection Pool Size
  - Outlier Detection
- Explanation:
  - **hosts**: service
  - Segment **backend service** with label app and version. This segmentation is called subset.
  - Use ROUND ROBIN for load balancing each subset.

Red Hat

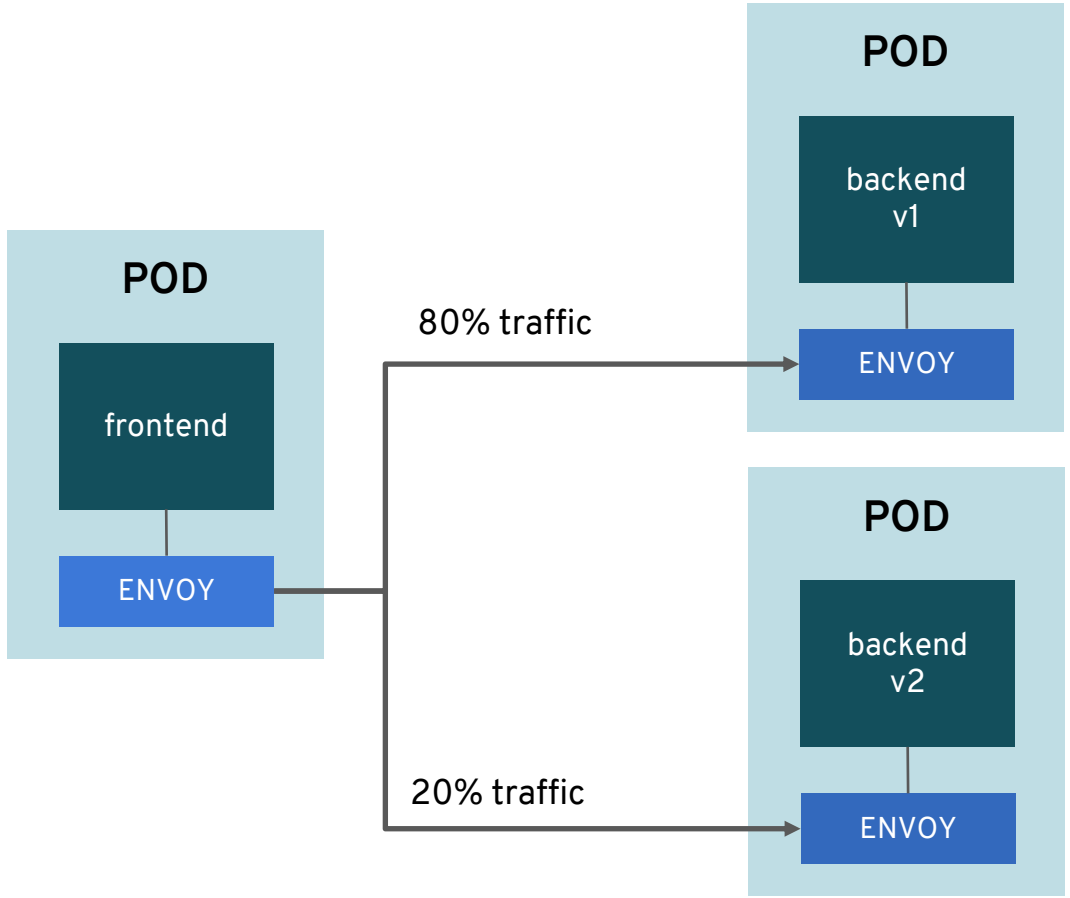
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: backend-virtual-service
spec:
  hosts:
  - backend
  http:
  - route:
    - destination:
        host: backend
        subset: v1
      weight: 80
    - destination:
        host: backend
        subset: v2
      weight: 20
```

Virtual Service

- Define rules that control how requests for service are routed within the service mesh
- Configuration affecting traffic routing. Here are a few terms useful to define in the context of traffic routing.
- Explanation:
  - **hosts**: address that client use to connect.
  - Routing traffic to destination based on subset by weight.
  - Also possible to route by using URI matching and URI rewriting

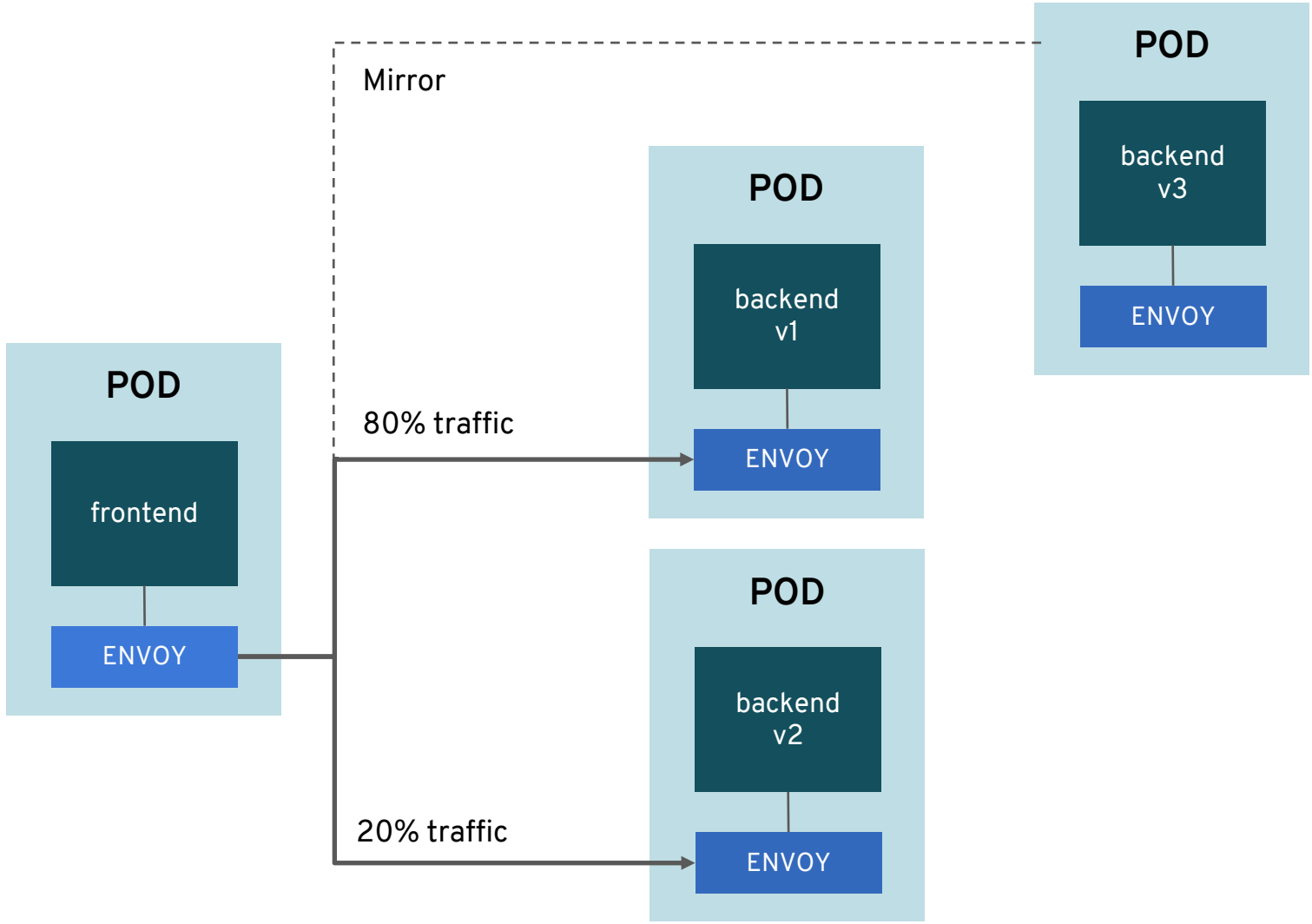
Red Hat

A/B DEPLOYMENT WITH ISTIO



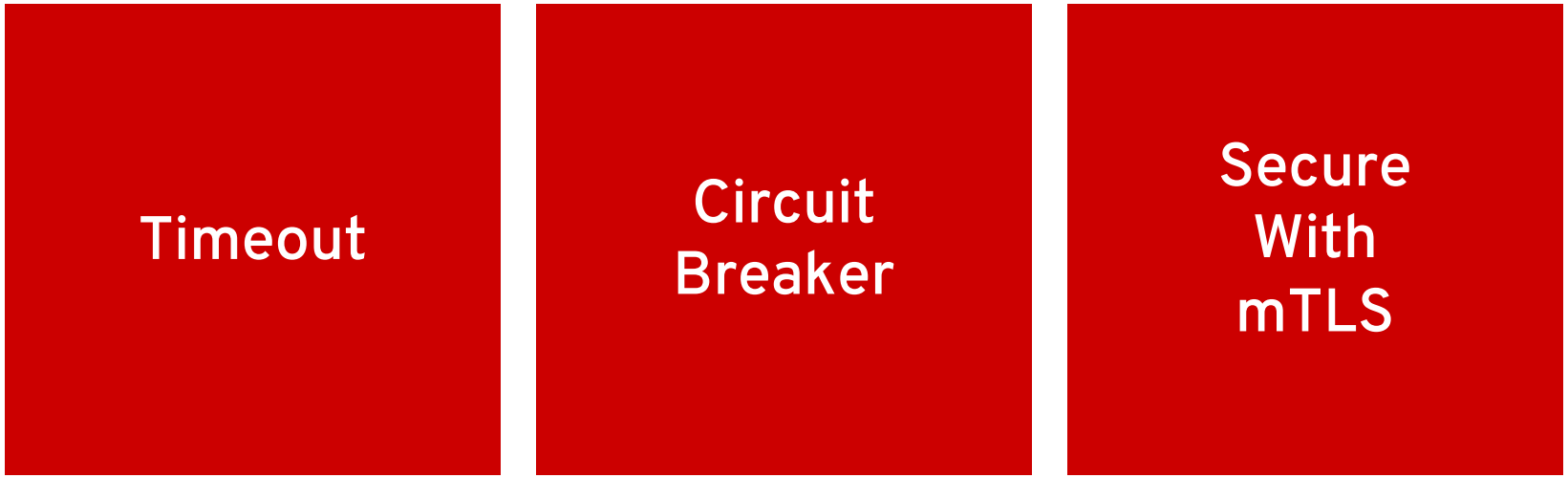
Red Hat

DARK LAUNCHES WITH ISTIO

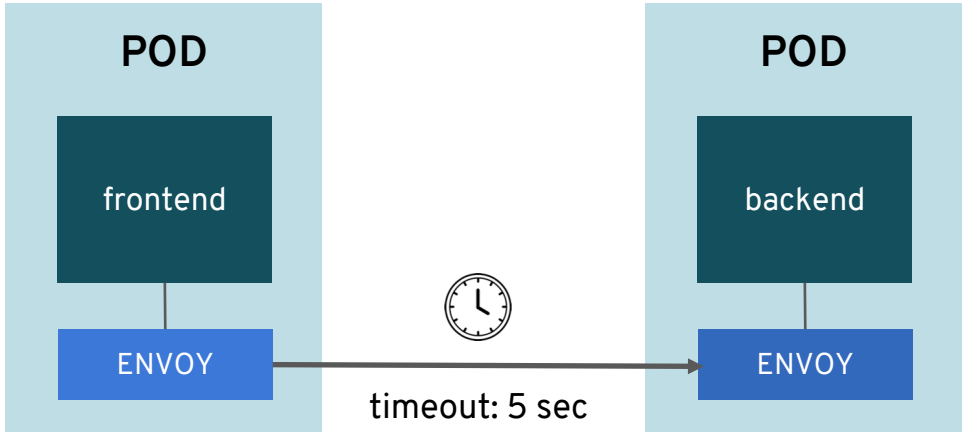


Red Hat

PART3: Control & Secure



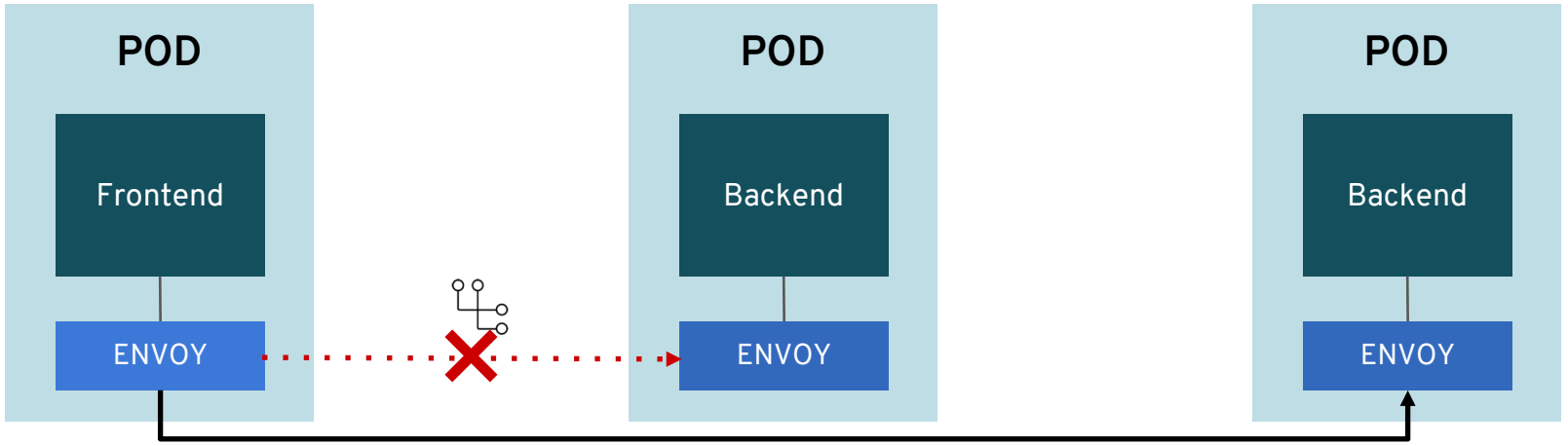
TIMEOUTS AND RETRIES WITH ISTIO



configure timeouts and retries, transparent to the services

Red Hat

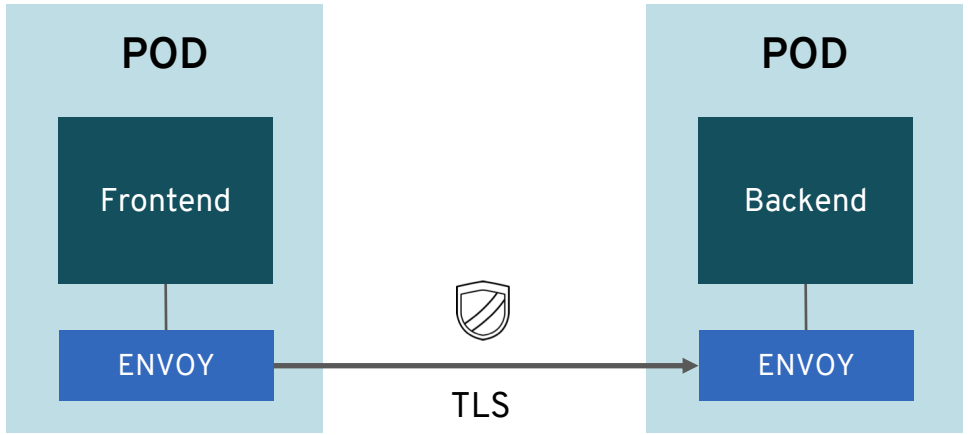
CIRCUIT BREAKERS WITH ISTIO



Once failed rate is exceed limit. The circuit breaker “trips” and stops further connections to that host.

Red Hat

SECURE COMMUNICATION WITH ISTIO



mutual TLS authentication, transparent to the services

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat