PART 5: RESILIENT DISTRIBUTED APPS

DISTRIBUTED SERVICES ARCHITECTURES

Benefits (when implemented correctly):

- Performance
- Reliability
- Resiliency
- Extensibility
- Availability
- Robustness

DISTRIBUTED SERVICES ARCHITECTURES

Fallacies of Distributed Computing

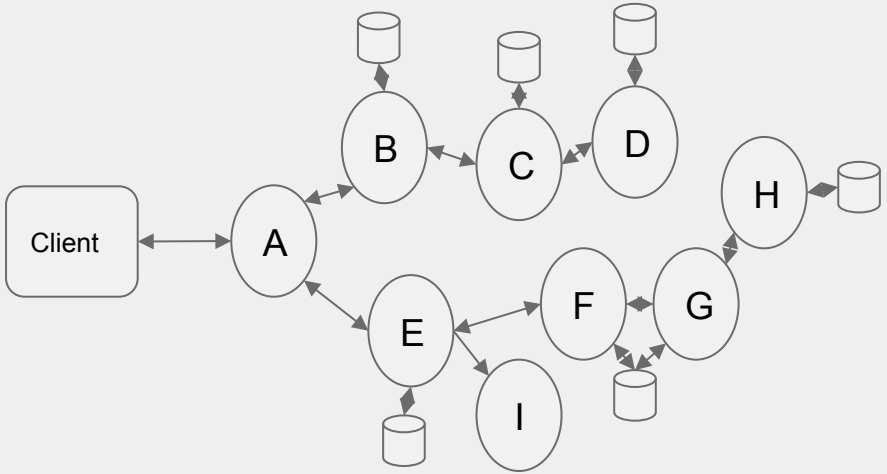
- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

wikipedia.org/wiki/Fallacies_of_distributed_computing

DISTRIBUTED SERVICES ARCHITECTURES

Applications must deal with

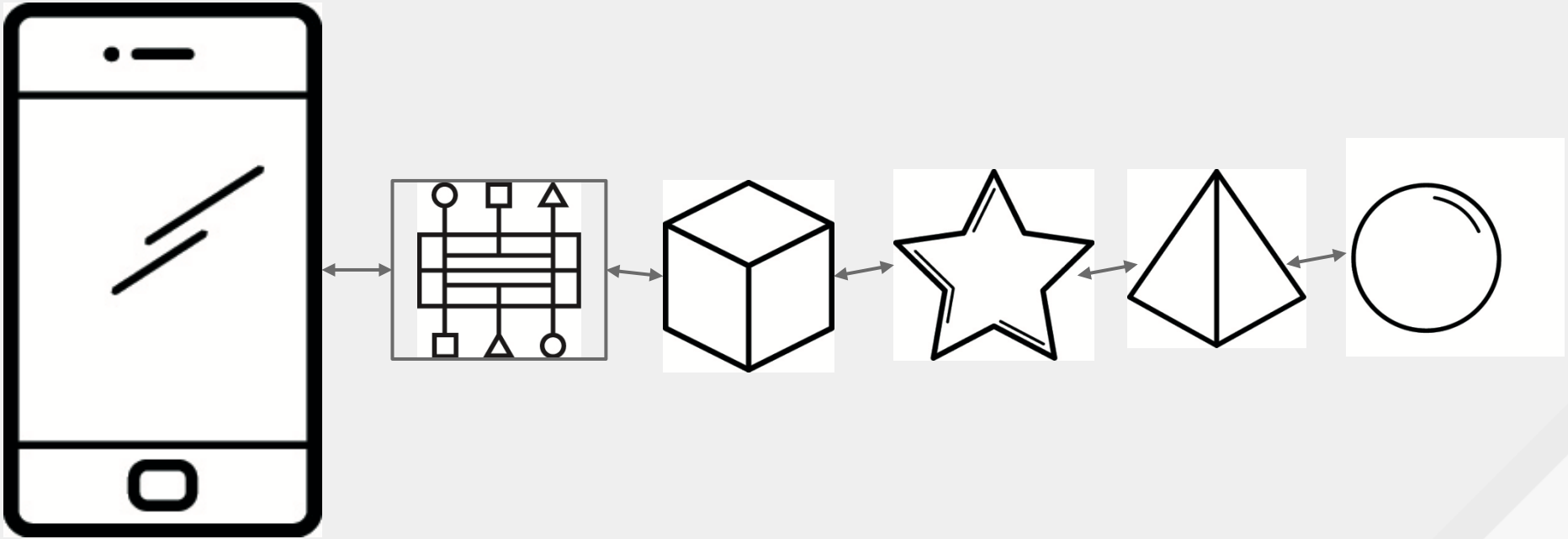
- Unpredictable failure modes
- End-to-end application correctness
- System degradation
- Topology changes
- Elastic/ephemeral/transient resources



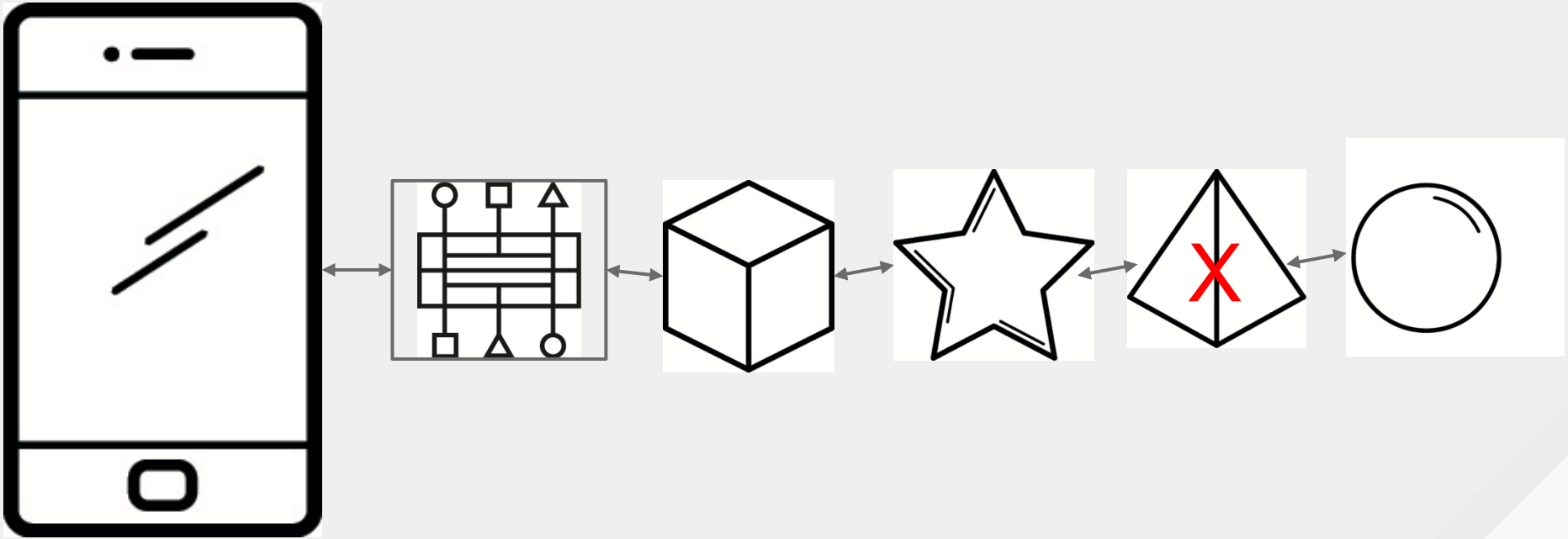
MICROSERVICES == DISTRIBUTED COMPUTING

Example

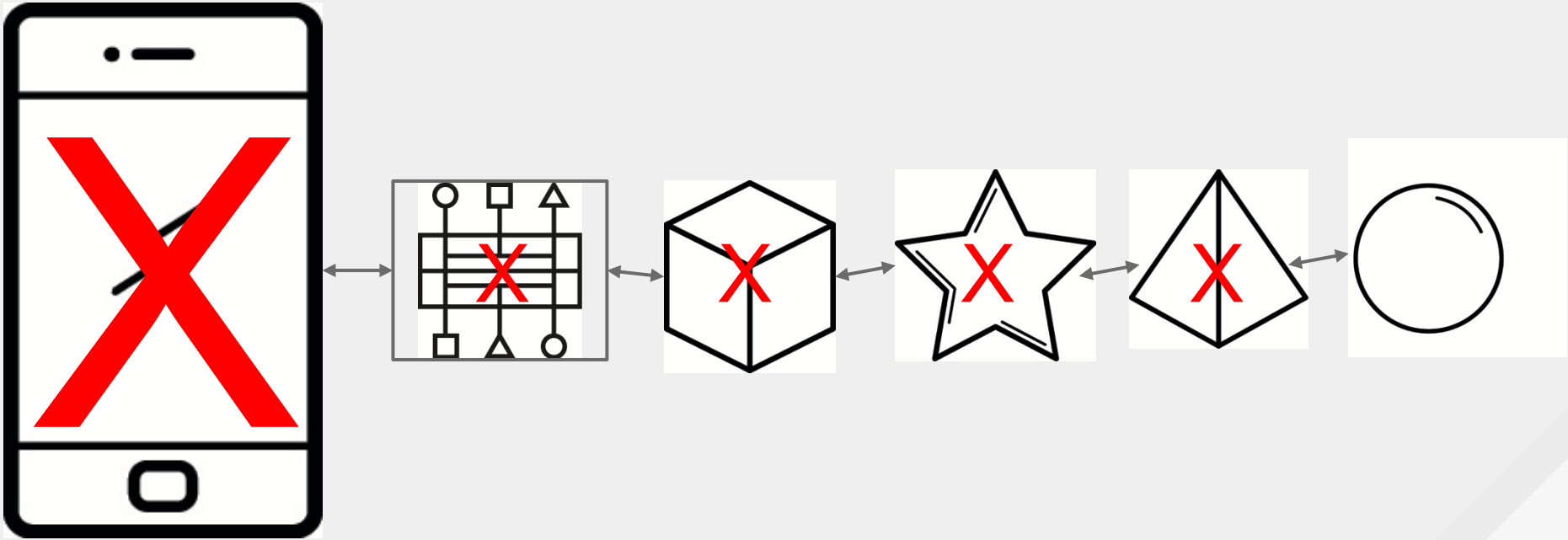
Chaining



Chaining (Fail)



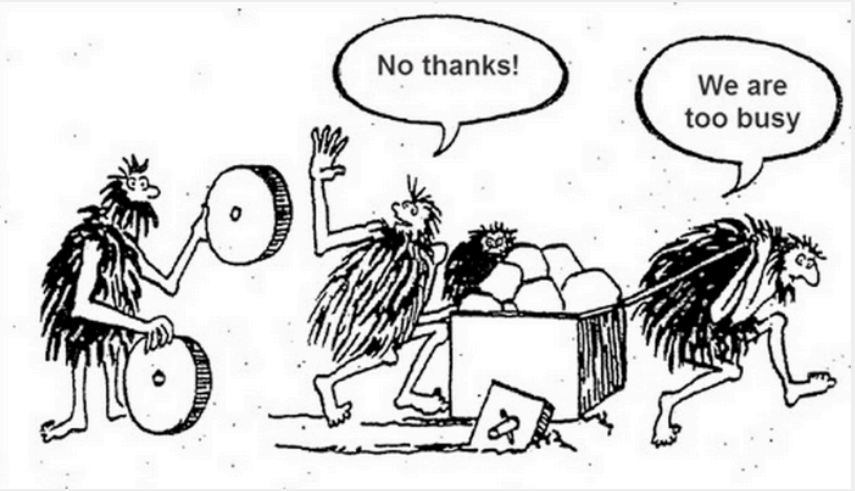
Chaining (Cascading Fail)



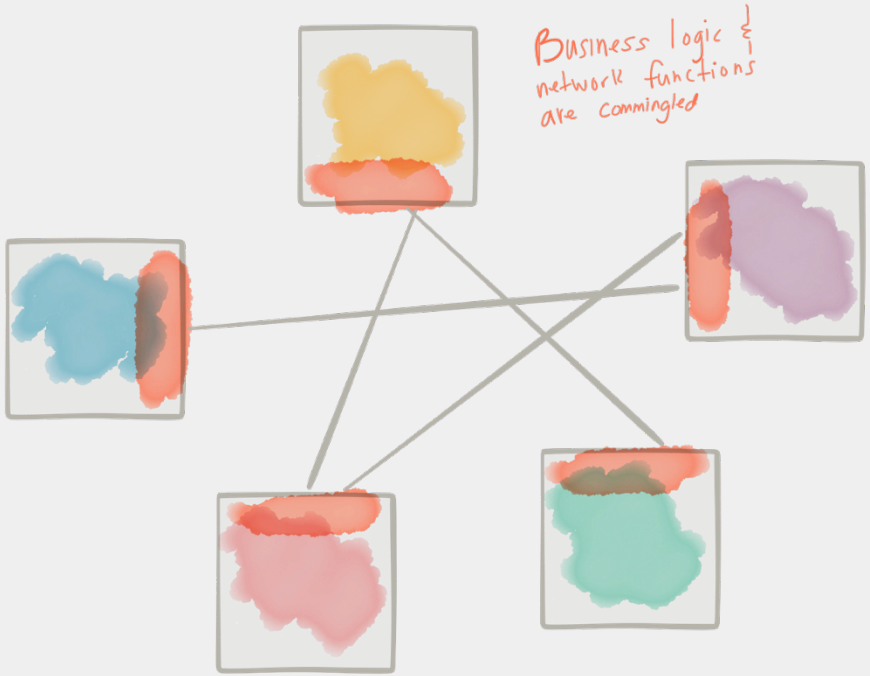
POSSIBLE SOLUTIONS

Today, Developers do this:

- Circuit Breaking
- Bulkheading
- Timeouts/Retries
- Service Discovery
- Client-side Load Balancing

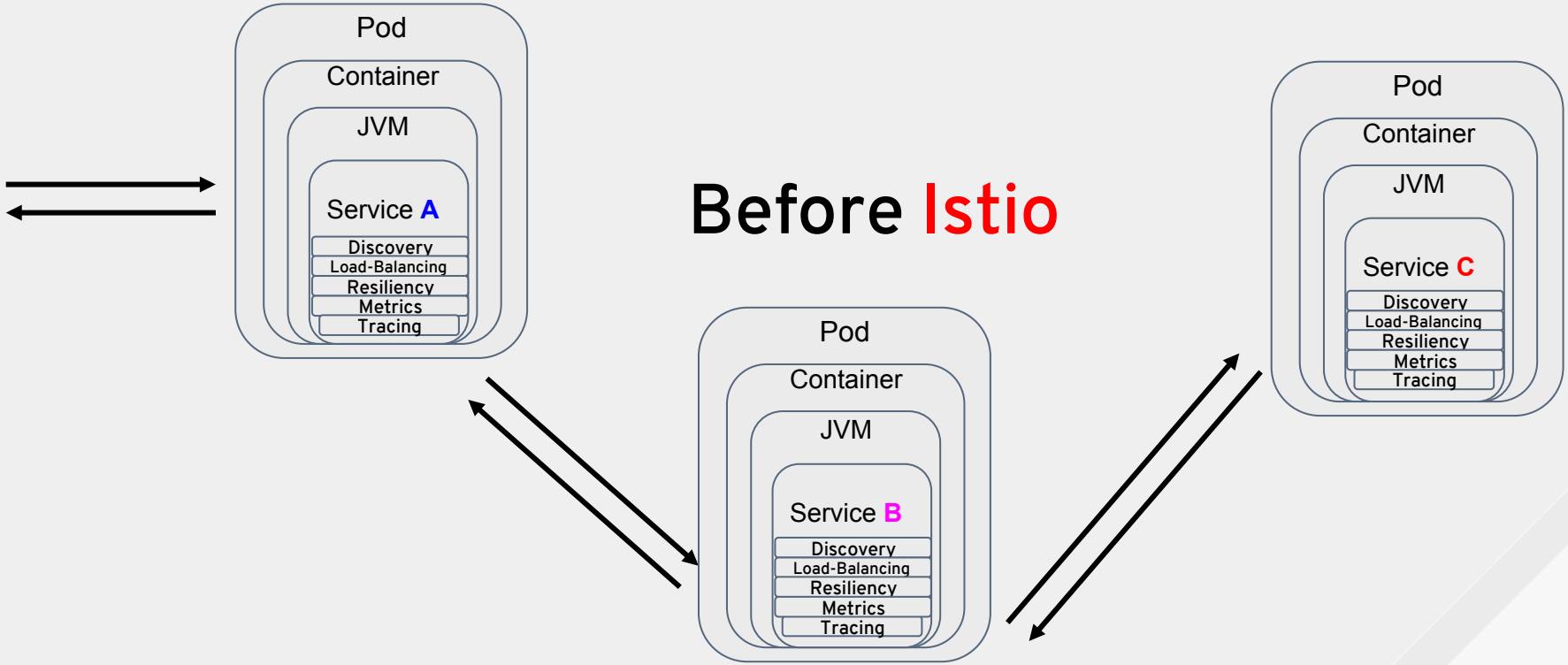


TOO MUCH INFRASTRUCTURE IN BUSINESS LOGIC



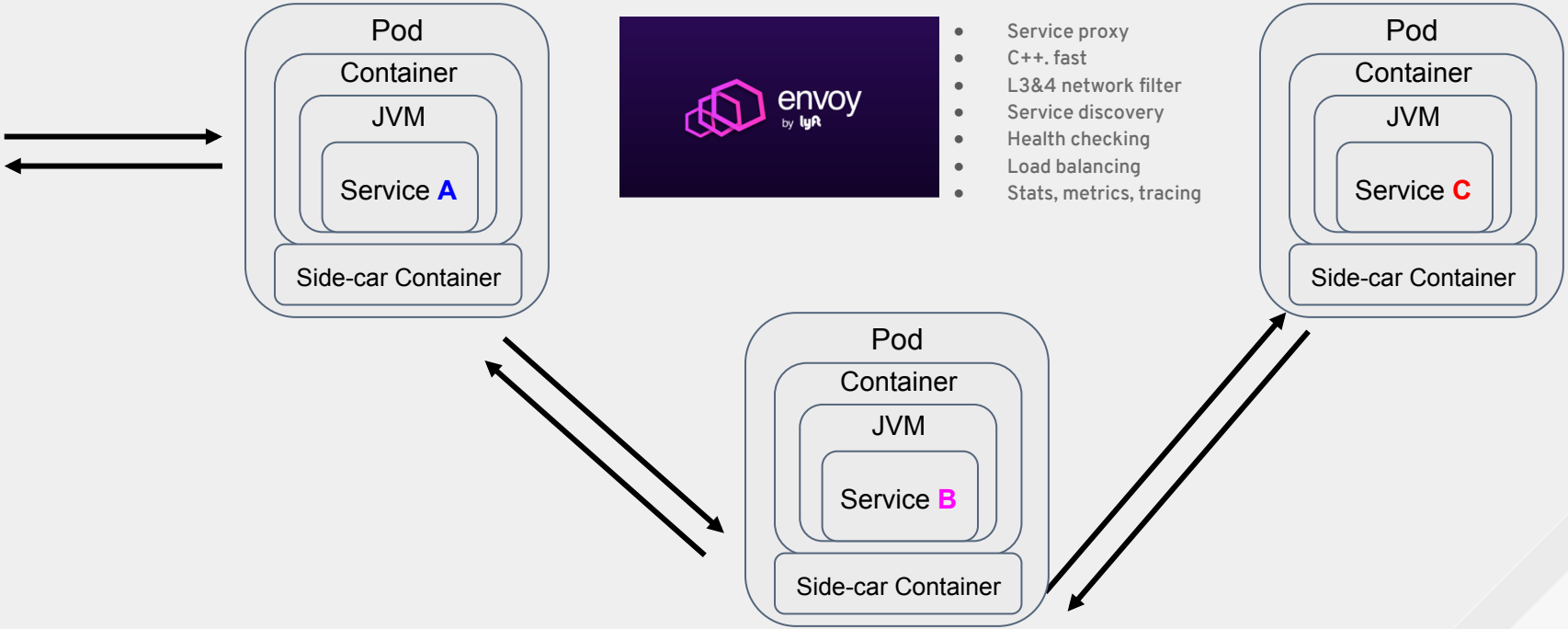
LAB: DETECTING AND PREVENTING ISSUES IN DISTRIBUTED APPS WITH ISTIO

PODS WITHOUT SERVICE MESH



SIDECARS

PODS WITH TWO CONTAINERS



Istio - Sail

(Kubernetes - Helmsman or ship's pilot)

ISTIO - A ROBUST SERVICE MESH FOR MICROSERVICES

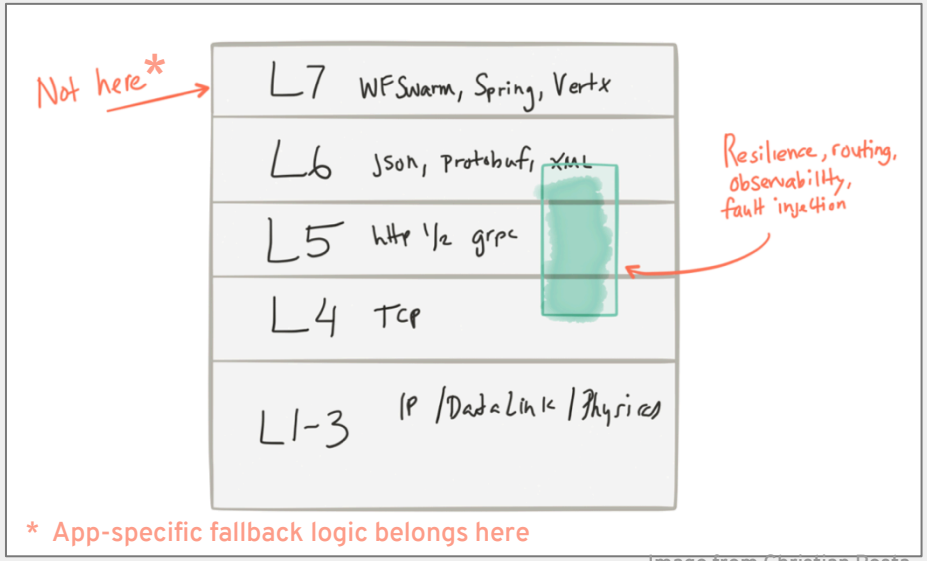


Image from Christian Posta

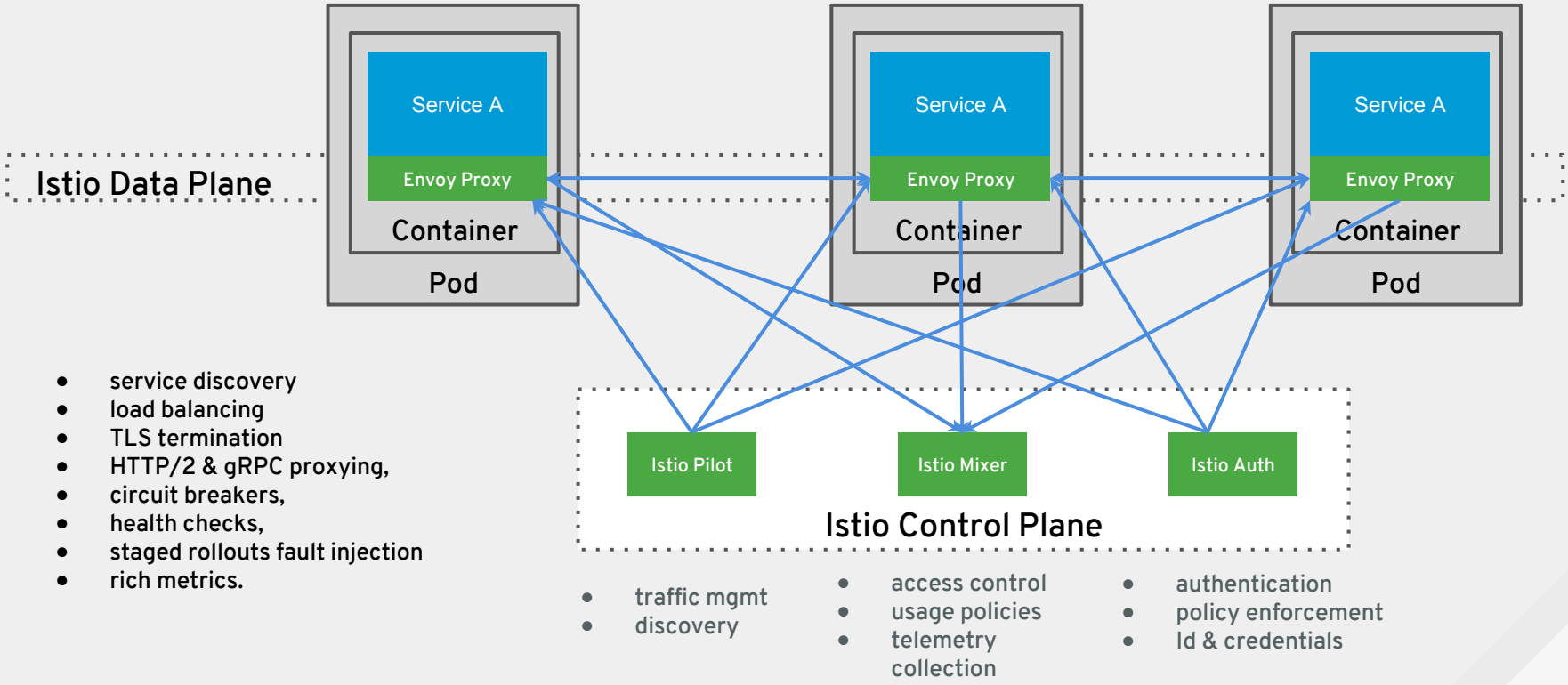
Key Features

- Intelligent routing and load balancing
- Fleet-wide, in-depth observability
- Resiliency across languages and platforms
- Fault injection
- Developer productivity
- Policy driven ops
- Circuit breaking, outlier detection
- Timeouts/retries
- Rate limiting
- Secure by default
- Incremental, unobtrusive adoption

Further Reading :

https://blog.openshift.com/red-hat-istio-launch/
https://istio.io/blog/istio-service-mesh-for-microservices.html
http://blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/

ISTIO - A ROBUST SERVICE MESH FOR MICROSERVICES



- service discovery
- load balancing
- TLS termination
- HTTP/2 & gRPC proxying,
- circuit breakers,
- health checks,
- staged rollouts fault injection
- rich metrics.

- traffic mgmt
- discovery

- access control
- usage policies
- telemetry collection

- authentication
- policy enforcement
- Id & credentials

MICROSERVICES 3.0 - SERVICE MESH

Code Independent:

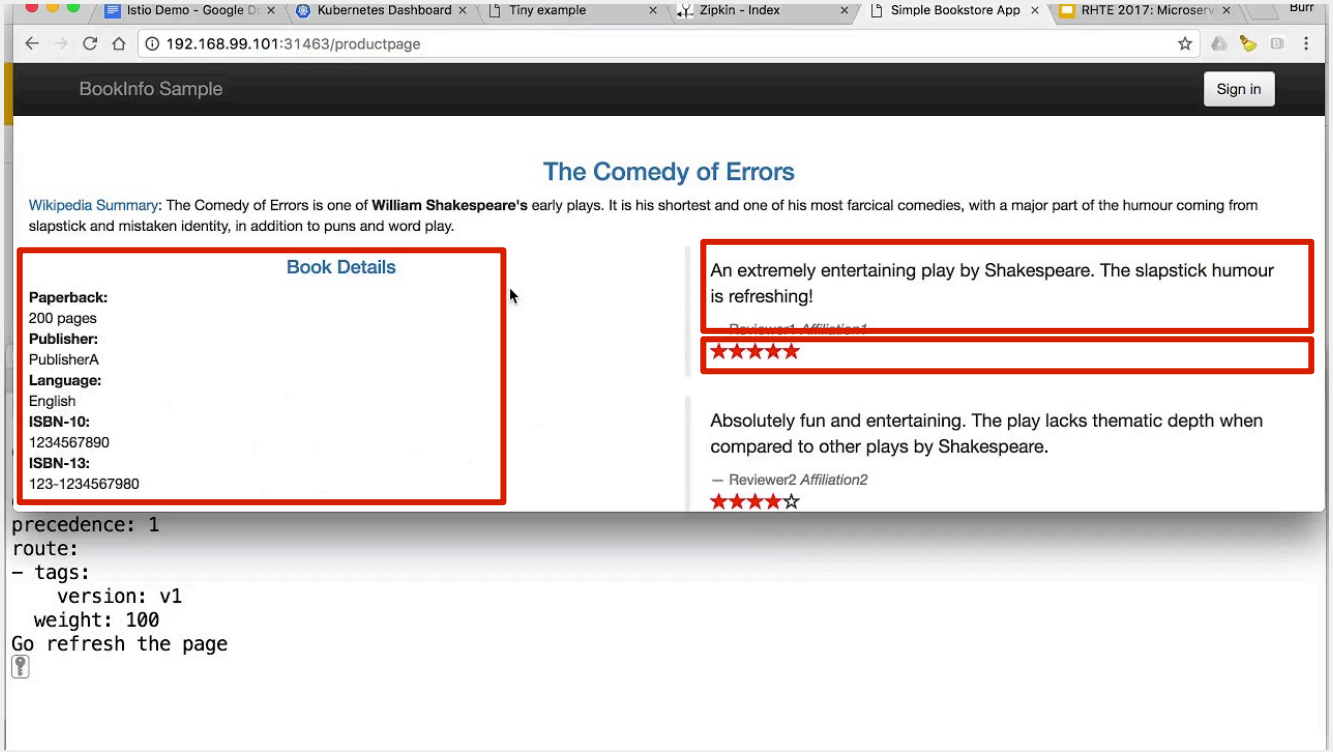
- Intelligent Routing and Load-Balancing
  - A/B Tests
  - Canary Releases
  - Dark Launches
- Distributed Tracing
- Circuit Breakers
- Fine grained Access Control
- Telemetry, metrics and Logs
- Fleet wide policy enforcement

GOAL FOR LAB

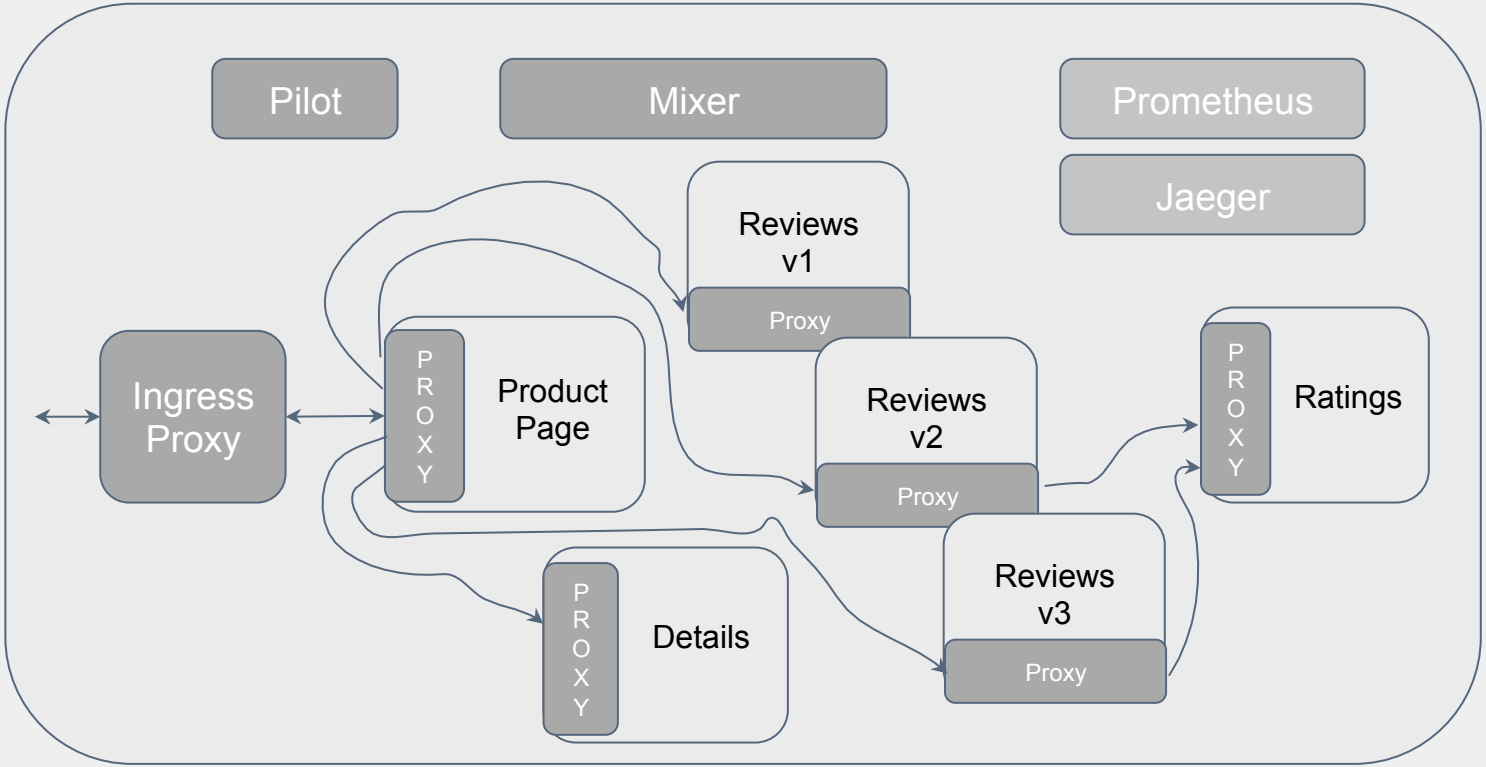
In this lab you will learn:

- How to **install Istio** onto **OpenShift Container Platform**
- How to deploy apps with **sidecar proxies**
- How to generate and visualize **deep metrics** for apps
- How to **alter routing** dynamically
- How to **inject faults** for testing
- How to do **rate limiting**
- How Istio implements **circuit breaking** and **distributed tracing**

SAMPLE APP: “BookInfo”



SAMPLE APP: “BookInfo”



LAB: DETECTING AND PREVENTING ISSUES IN DISTRIBUTED APPS WITH ISTIO

SCENARIO 7

PREVENT AND DETECT ISSUES IN A DISTRIBUTED SYSTEM

WRAP-UP AND DISCUSSION

RESULT OF LAB

In this lab you learned:

- How to install Istio onto OpenShift Container Platform
- How to deploy apps with sidecar proxies
- How to generate and visualize deep metrics for apps
- How to alter routing dynamically
- How to inject faults for testing
- How to do rate limiting
- How Istio implements circuit breaking and distributed tracing
- Use cases for service mesh

MICROSERVICES 4.0?

Service

> Autonomous
> Loosely-coupled

Microservice

> Single Purpose
> Stateless
> Independently Scalable
> Automated

Function

f( )

> Single Action
> Event-sourced
> Ephemeral

SERVERLESS PROJECTS / SERVICES

syncano

webtask

APACHE OpenWhisk™

Iron.io

APEX | SERVERLESS INFRASTRUCTURE

Microsoft Azure

Back&

fission

SERVERLESS

http://funcatron.org

CLOUD FUNCTIONS
BETA

serverless-docker

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos