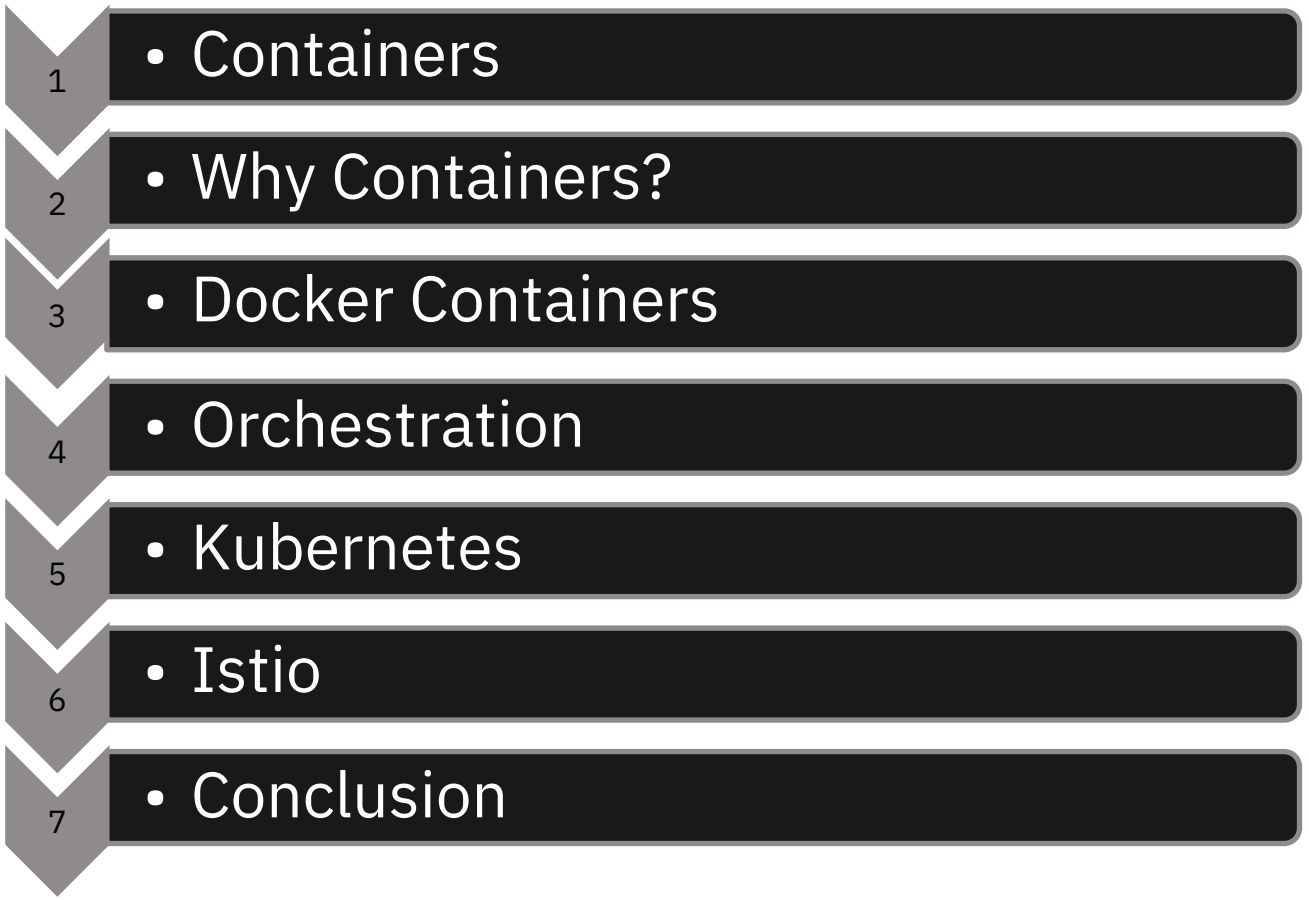
Serverless and Containers in Cloud Age – part two

IBM **Code**

Marek Sadowski
Developer Advocate | IBM San Francisco

@blumareks

Agenda


1
• Containers
2
• Why Containers?
3
• Docker Containers
4
• Orchestration
5
• Kubernetes
6
• Istio
7
• Conclusion

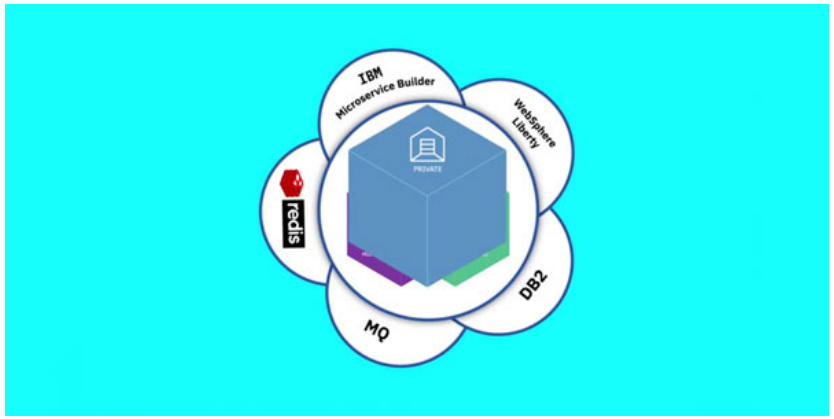
Containers

A standard way to package an application and all its dependencies so that it can be moved between environments and run without changes.

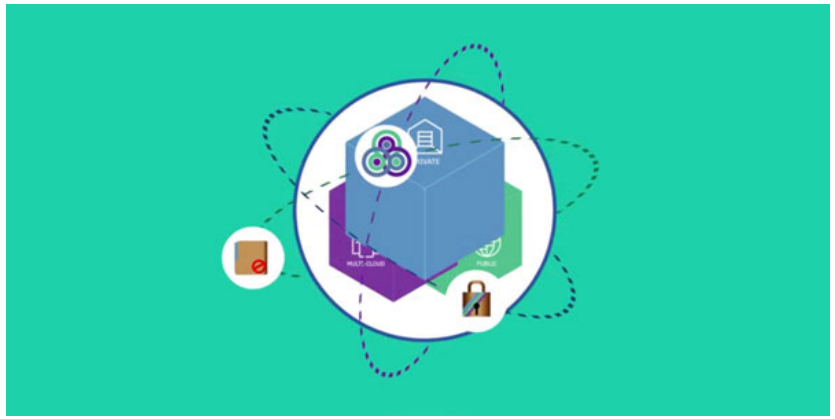
Containers work by isolating the differences between applications inside the container so that everything outside the container can be standardized.

Why Containers?

Use Cases for Containers



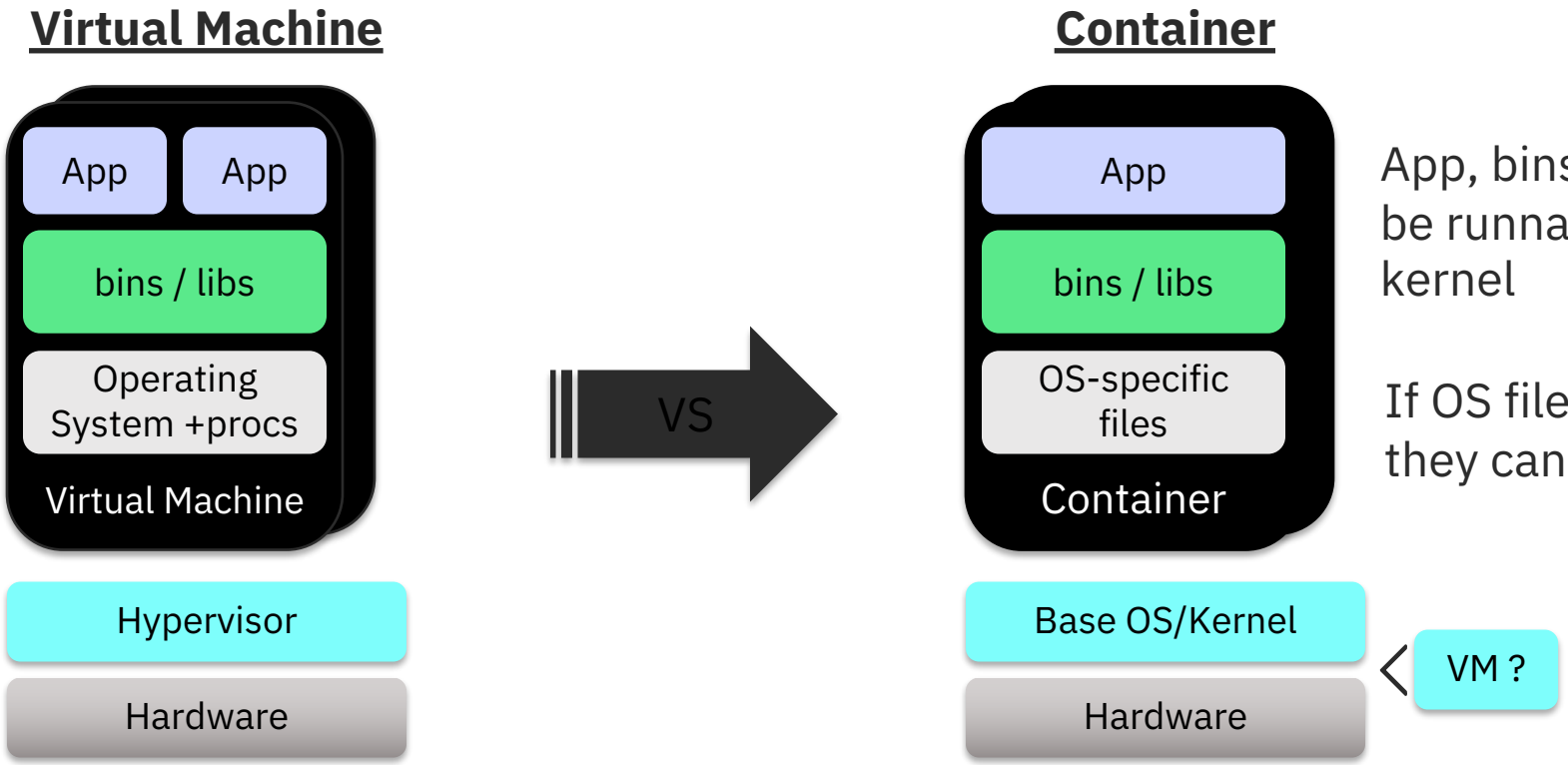
Create new cloud-native apps
Streamline development
with built-in microservices, runtimes, containers and Kubernetes orchestration plus integrated management.



Modernize your heritage apps on cloud
Move your apps as-is to the cloud or re-factor an app and use it in new development and application workload models.

Open your data center to work with cloud services
Protect and leverage your in-house data and pull in external

VM vs Container



Each VM has its own OS

Containers share the same base Kernel

App, bins/libs/OS must all be runnable on the shared kernel

If OS files aren't needed they can be excluded.

Dev vs. Ops

Dev	Ops

- Code
- Libraries
- Configuration
- Server runtime
- OS

- Logging
- Remote access
- Network configuration
- Monitoring

Separation of concerns

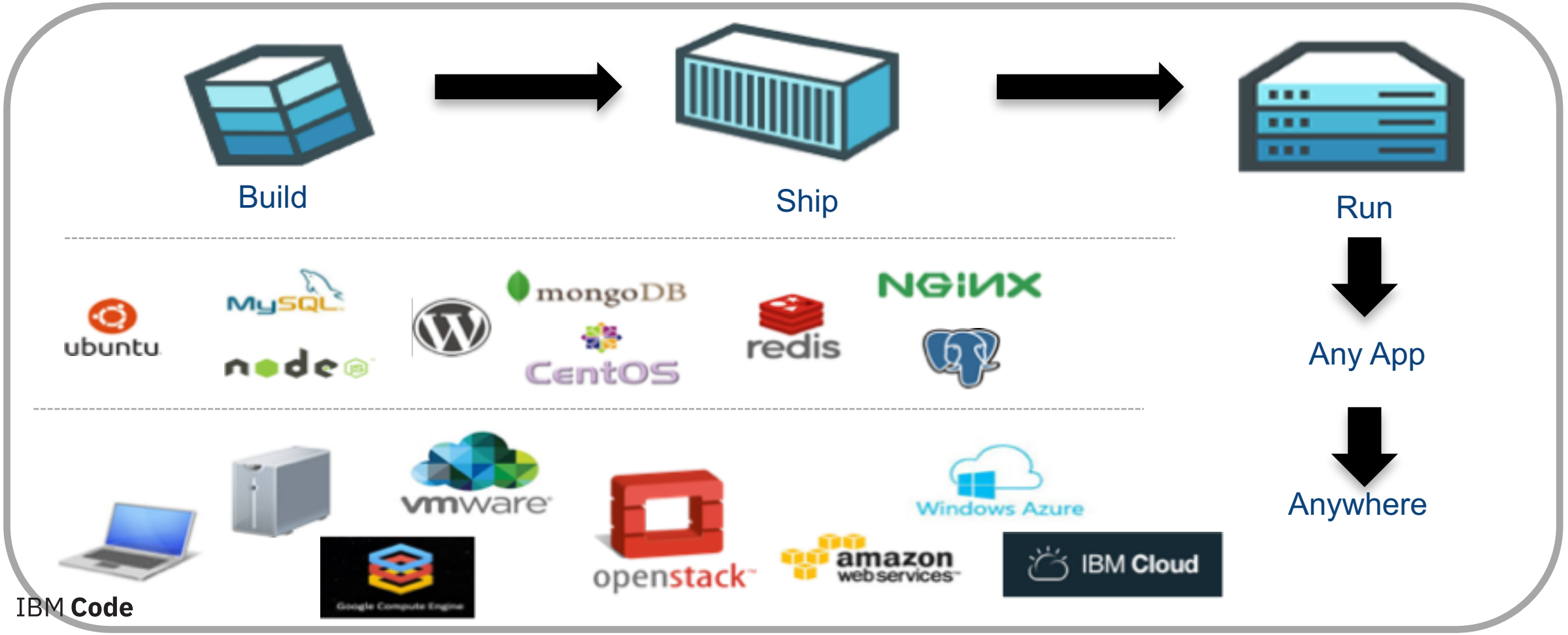
A container separates and bridges the Dev and Ops in DevOps

- Dev focuses on the application environment
- Ops focuses on the deployment environment

Docker Containers

Docker Mission

Docker is an **open platform** for building distributed applications for **developers** and **system administrators**



What is Docker?

Tooling to manage containers

- Containers are not new
- Docker just made them easy to use

Docker creates and manages the lifecycle of containers

- Setup filesystem
- CRUD container
  - Setup networks
  - Setup volumes / mounts
  - Create: start new process telling OS to run it in isolation

Our First Container

```
$ docker run ubuntu echo Hello World
Hello World
```

Our First Container

```
$ docker run ubuntu echo Hello World
Hello World
```

What happened?

- Docker created a directory with a "ubuntu" filesystem (image)
- Docker created a new set of namespaces
- Ran a new process: `echo Hello World`
  - Using those namespaces to isolate it from other processes
  - Using that new directory as the "root" of the filesystem (`chroot`)
- That's it!
  - Notice as a user I never installed "ubuntu"
- Run it again - notice how quickly it ran

```
Mareks-MBP:~ mareksadowski$ docker run ubuntu echo Hello World
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
6b98dfc16071: Pull complete
4001a1209541: Pull complete
6319fc68c576: Pull complete
b24603670dc3: Pull complete
97f170c87c6f: Pull complete
Digest: sha256:5f4bdc3467537cbbe563e80db2c3ec95d548a9145d64453b06
Status: Downloaded newer image for ubuntu:latest
Hello World
Mareks-MBP:~ mareksadowski$ docker run ubuntu echo Hello World
Hello World
Mareks-MBP:~ mareksadowski$
```

ssh-ing into a container - fake it...

```
$ docker run -ti ubuntu bash
```

```
Mareks-MBP:~ mareksadowski$ docker run -ti ubuntu bash
root@c6d6a09b9db9:/# pwd
/
root@c6d6a09b9db9:/# time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
root@c6d6a09b9db9:/# du
12      ./root
3836    ./sbin
4       ./run/lock
4       ./run/mount
8       ./run/systemd
```

Now the process is "bash" instead of "echo"

But its still just a process

Look around, mess around, its totally isolated

– MAKE SURE YOU'RE IN A CONTAINER!

A look under the covers

```
$ docker run ubuntu ps -ef
UID          PID    PPID  C
STIME TTY          TIME CMD
root           1       0  0
14:33 ?        00:00:00 ps -ef
```

Things to notice with these examples

- Each container only sees its own process(es)
- Running as "root"
- Running as PID 1

Docker Component Overview

Docker Engine

– Manages containers on a host

– Accepts requests from clients

  • REST API

– Maps container ports to host ports

  • E.g. 80 → 3582

Images

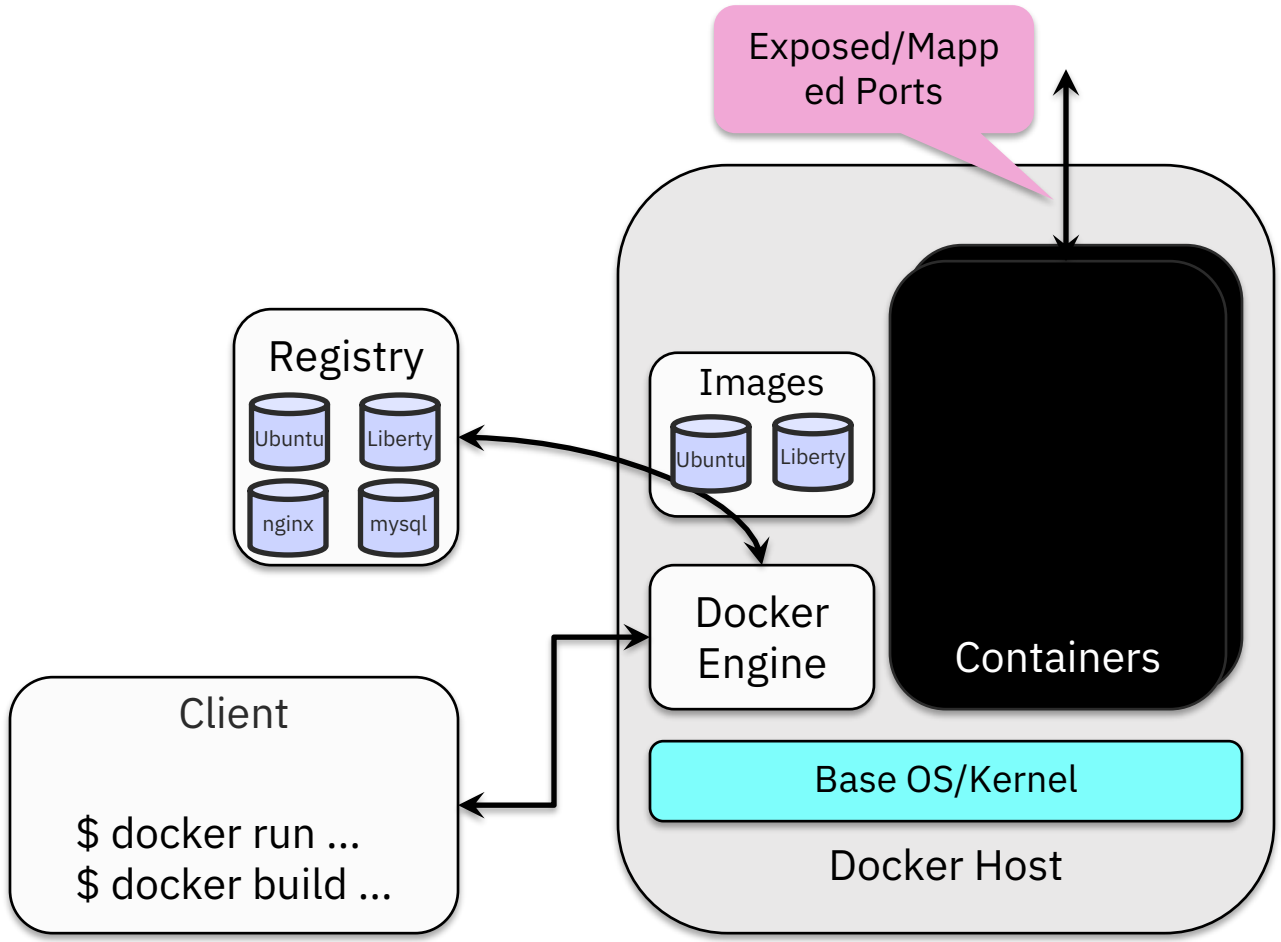
Docker Client

– Drives engine

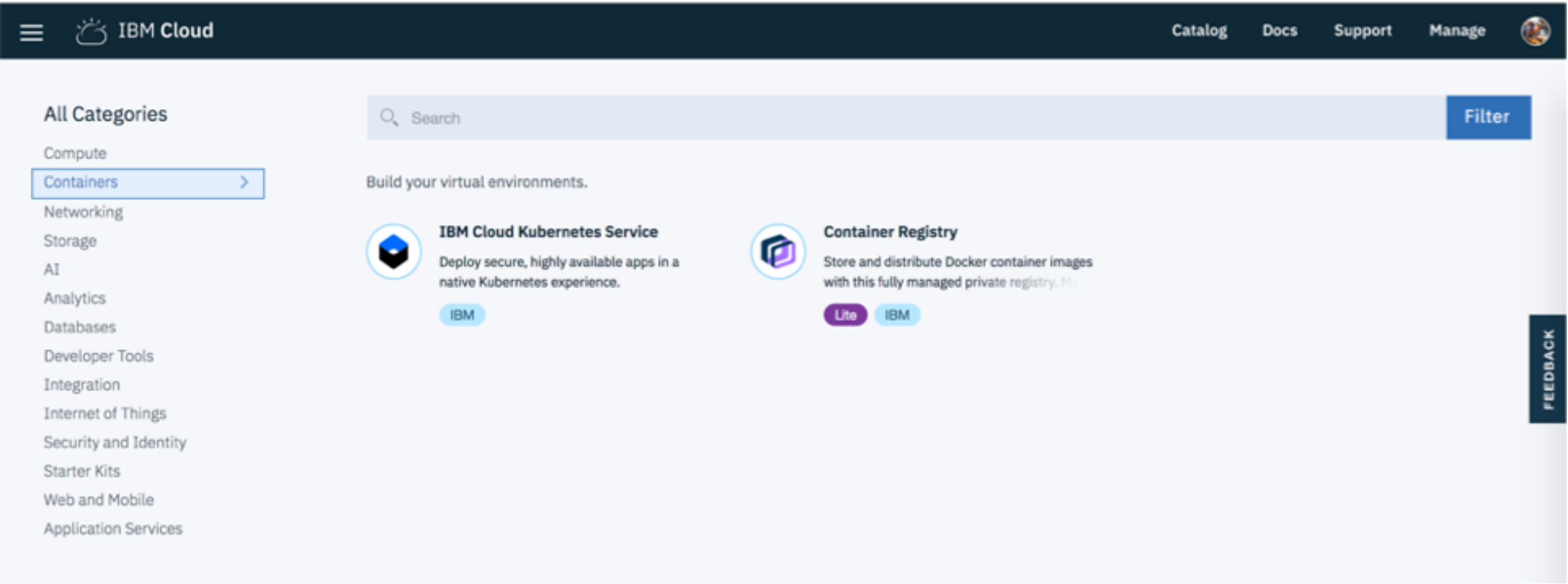
– Drives "builder" of Images

Docker Registry

– Image DB



IBM Cloud – Enterprise grade registry



Containers

Everyone’s container journey starts with one container....

Containers

At first the growth is easy to handle....

Containers

But soon it is overwhelming... chaos reigns

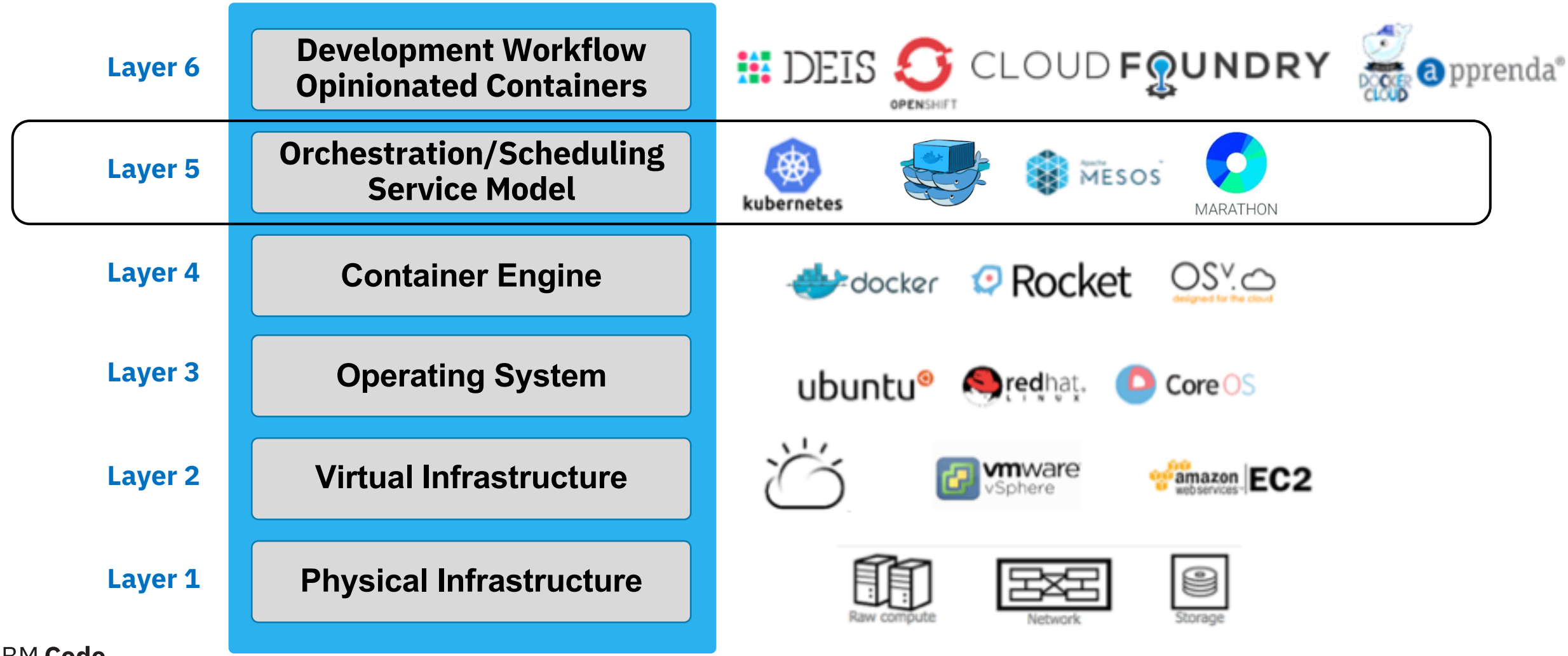
Orchestration

Container Orchestration

Allows users to define how to coordinate the containers in the cloud when the multi-container packaged application is deployed.

- **Scheduling**
- **Cluster management**
- **Service discovery**
- **Provisioning**
- **Monitoring**
- **Configuration management**

Container Ecosystem Layers



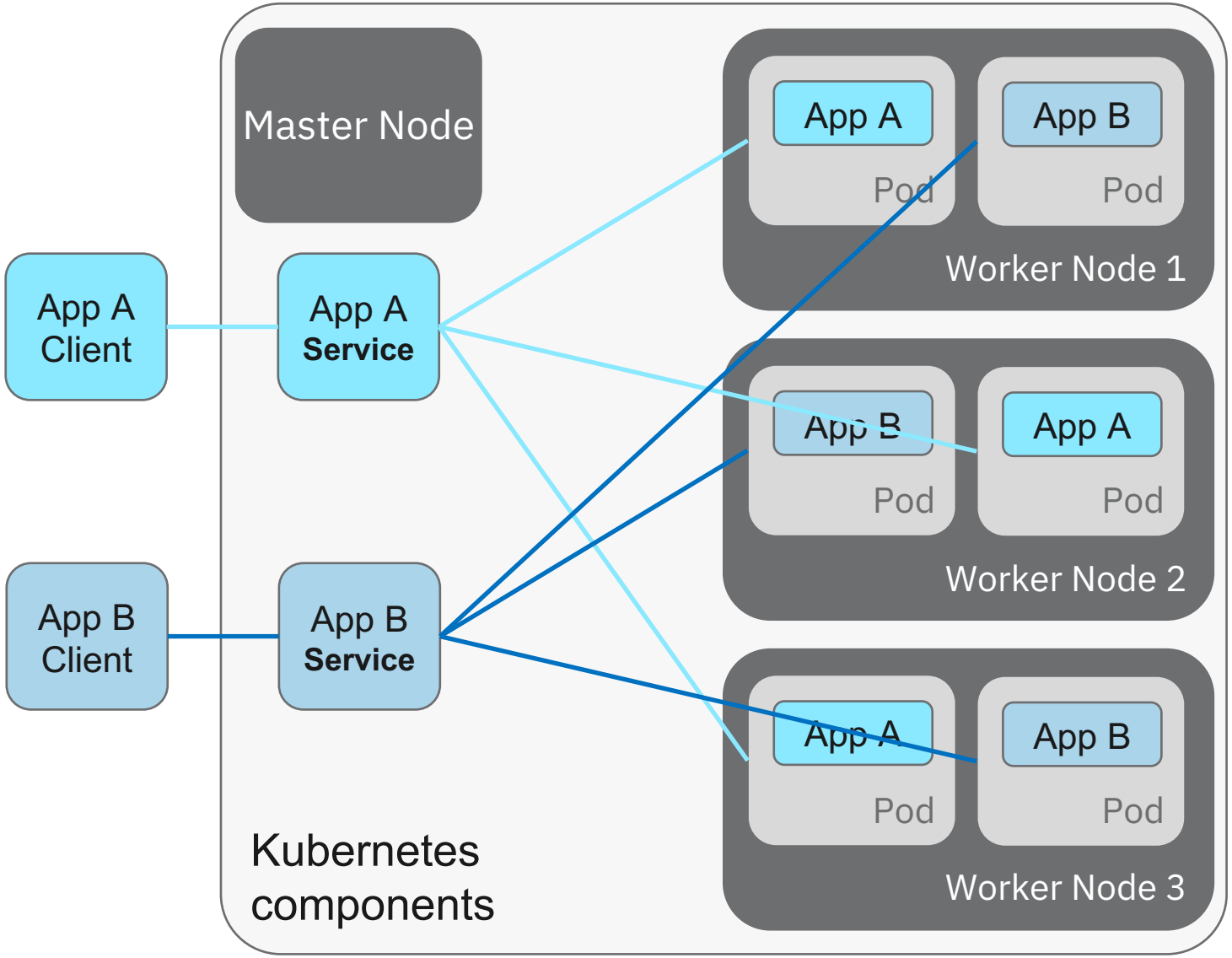
Kubernetes

What is Kubernetes?

- Container orchestrator
- Manage applications, **not machines**
- Designed for extensibility
- Open source project managed by the Linux Foundation

Kubernetes Architecture: Workloads

- Container
  - Packaging of an app
- Pod
  - Unit of deployment
- Service
  - Fixed endpoint for 1+ pods



Kubernetes Capabilities

Intelligent Scheduling

Self-healing

Horizontal scaling

Service discovery & load balancing

Automated rollouts and rollbacks

Secret and configuration management

Cluster Management Capabilities

Simplified cluster management

Design your own cluster

Container security & isolation

Extend with IBM Cloud & Watson

Native open-source experience

Integrated operational tools

IBM Cloud
Kubernetes Service

Design Your Own Cluster

- Tunable capacity
- Select between shared and dedicated compute using virtual server instances
- Bare metal worker nodes
- Edge nodes
- Configurable networking and storage
- Integrated VPN in-cluster providing IPSec tunnels

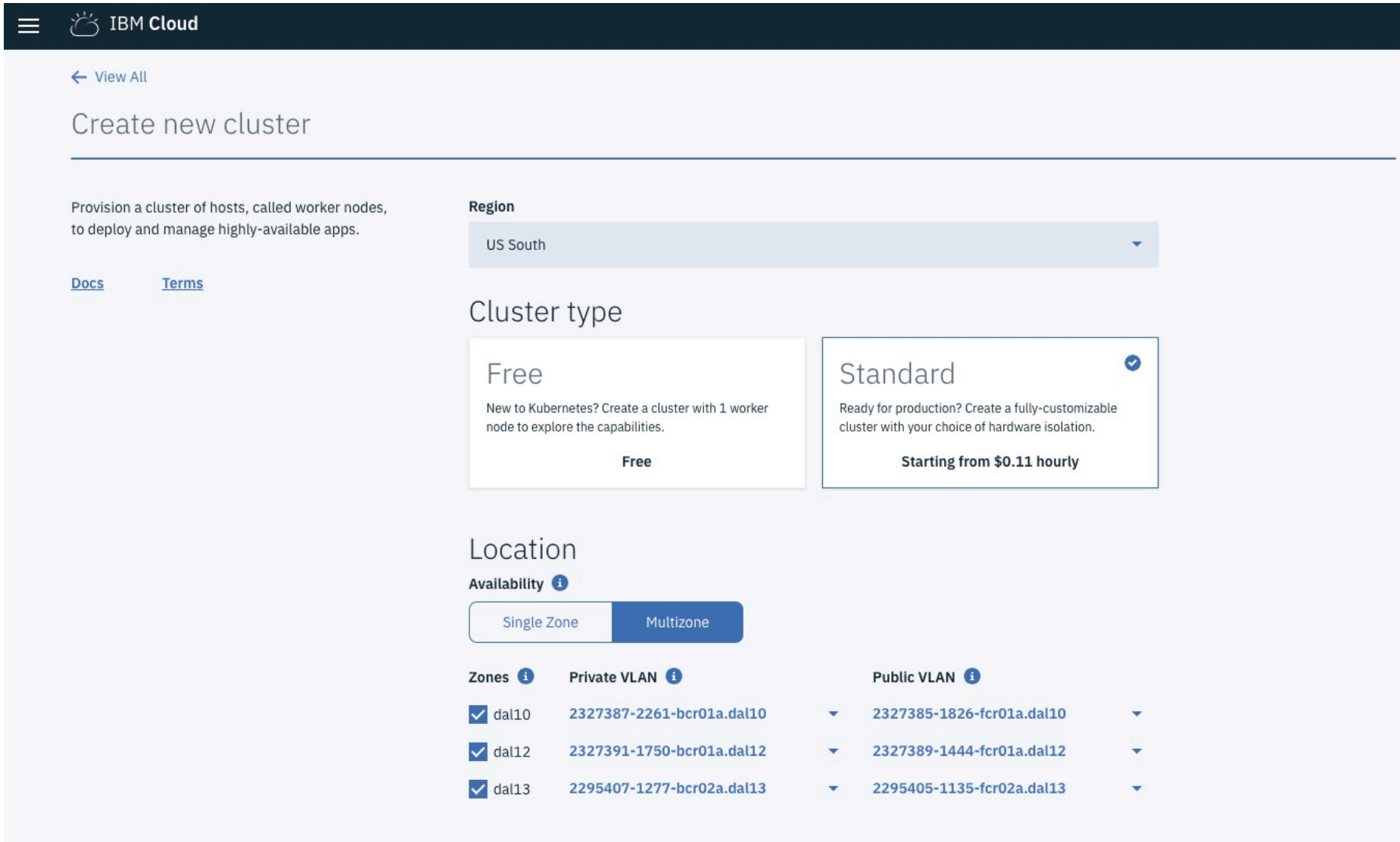
IBM Cloud
Kubernetes Service

Container Security & Isolation

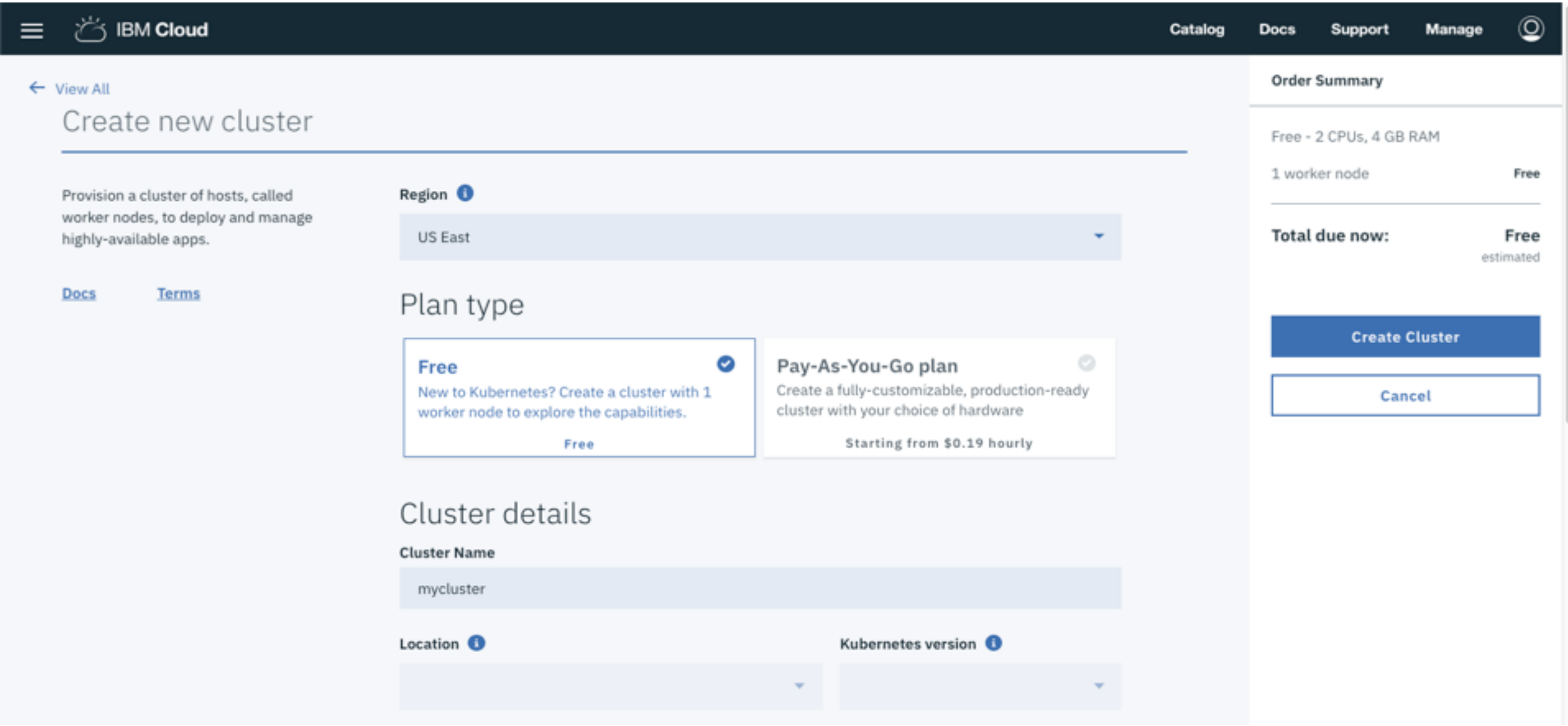
- Provides Docker image and running container scanning to detect vulnerabilities and configuration weaknesses with Vulnerability Advisor
- Image security enforcement
- Store your images securely in your hosted private registry
- Docker image signing
- Automatic encryption of secrets and volumes
- Isolated networking and storage
- Default LUKS encryption of /var/lib/docker
  - Every worker node in each cluster has a unique encryption key

Multizone Kubernetes

In June 2018 IBM announced the availability of multizone clusters. Now within the same IBM Cloud region, you can deploy a Kubernetes cluster with worker nodes distributed across different zones (i.e., physical datacenters) ensuring your apps and clusters remain highly available.



IBM Cloud Kubernetes – try Kubernetes for free!



Istio

An open platform to connect, manage, and secure microservices

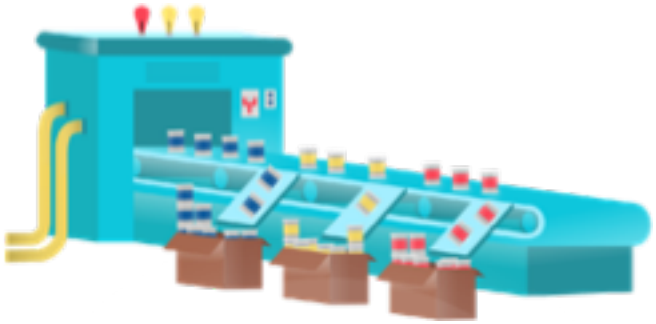
IBM **Code**

http://istio.io

Istio

WHERE ISTIO FINDS ITS PLACE

Intelligent Routing and Load Balancing



Fleet Wide Policy Enforcement

Resiliency across Languages and Platforms

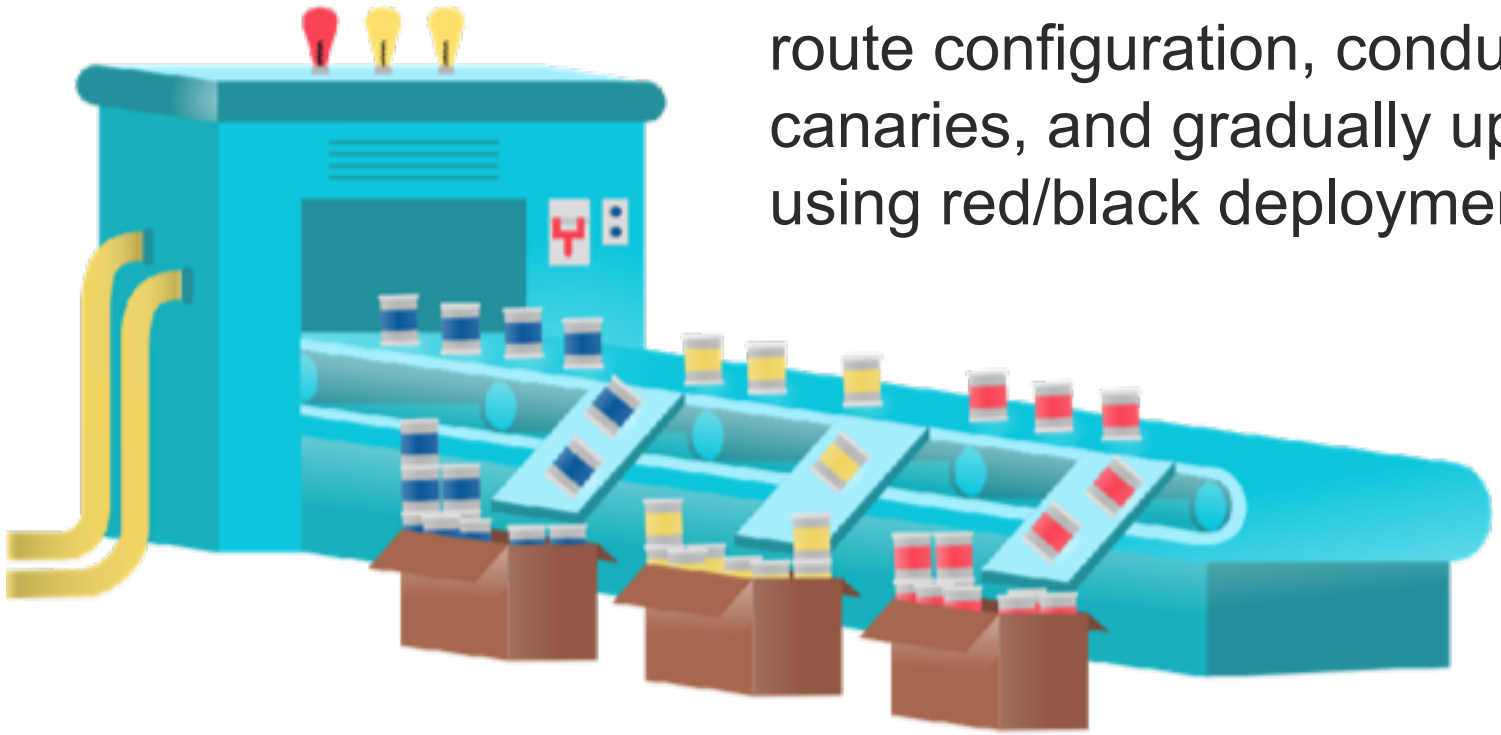
In-Depth Telemetry and Reporting

IBM **Code**

Istio

WHERE ISTIO FINDS ITS PLACE

Intelligent Routing and Load Balancing

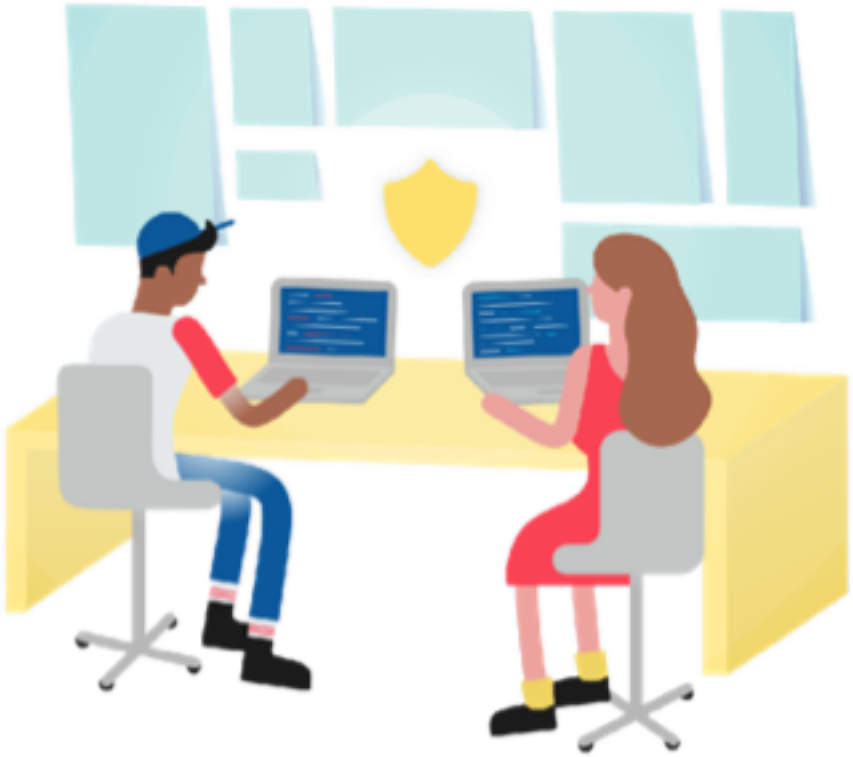


Control traffic between services with dynamic route configuration, conduct A/B tests, release canaries, and gradually upgrade versions using red/black deployments.

Istio

WHERE ISTIO FINDS ITS PLACE

Resiliency across Languages and Platforms



Increase reliability by shielding applications from flaky networks and cascading failures in adverse conditions.

IBM **Code**

Istio

WHERE ISTIO FINDS ITS PLACE

Fleet Wide Policy Enforcement

Apply organizational policy to the interaction between services, ensure access policies are enforced and resources are fairly distributed among consumers.

Istio

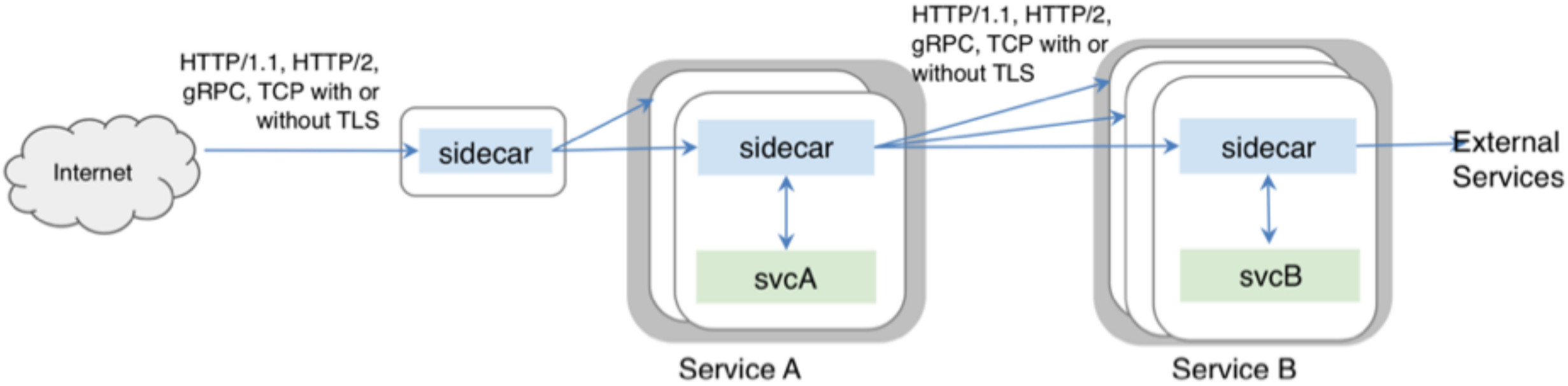
WHERE ISTIO FINDS ITS PLACE

In-Depth Telemetry and Reporting

Understand the dependencies between services, the nature and flow of traffic between them and quickly identify issues with distributed tracing.

IBM **Code**

Istio – original concept - Sidecar



Outbound features:

- Service authentication
- Load balancing
- Retry and circuit breaker
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection

Inbound features:

- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

IBM **Code**

Envoy by Lyft – sidecar of choice

- A C++ based L4/L7 proxy
- Low memory footprint
- Battle-tested @ Lyft
  - 100+ services
  - 10,000+ VMs
  - 2M req/s

Goodies:

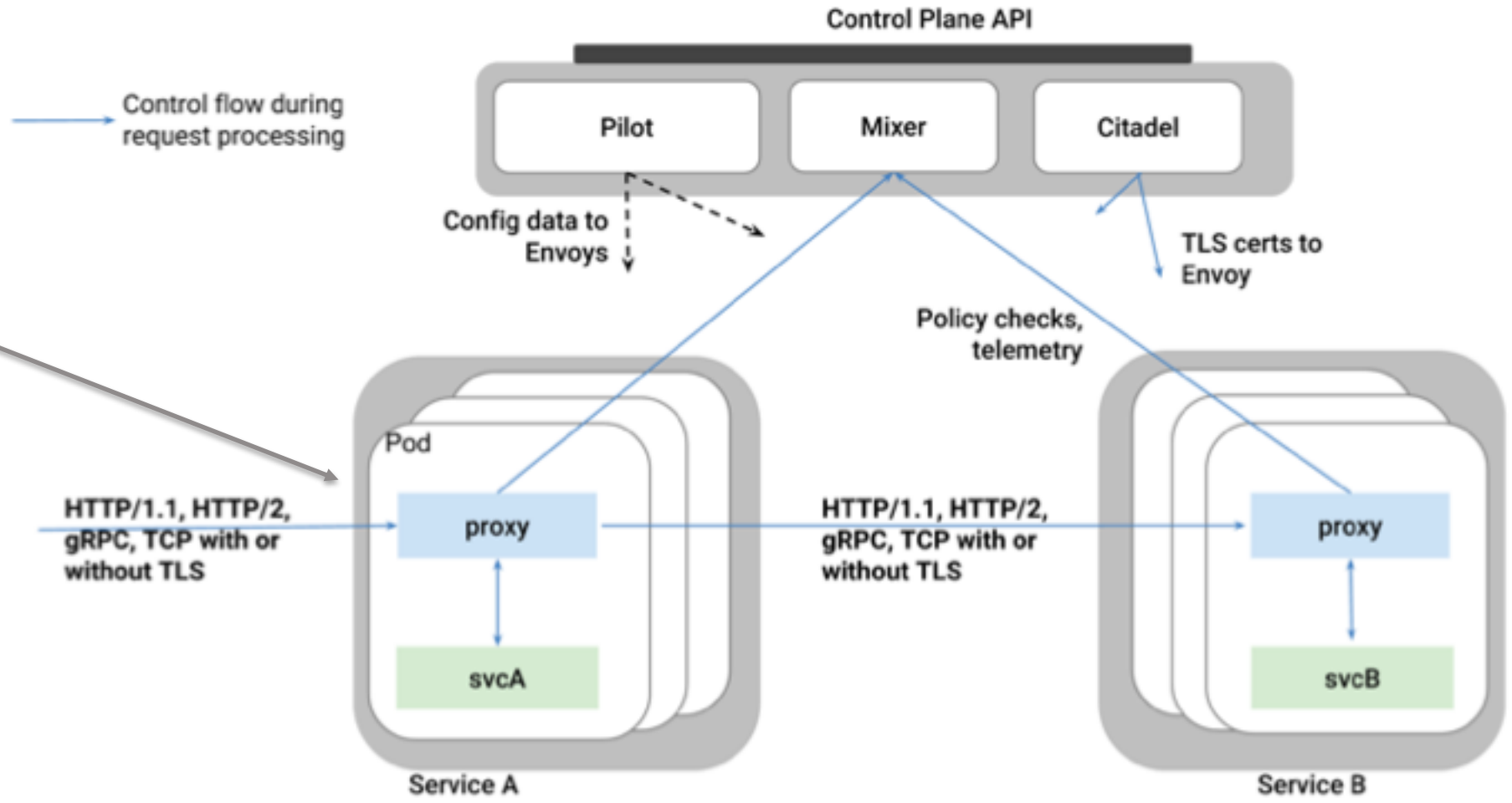
- HTTP/2 & gRPC
- Zone-aware load balancing w/ failover
- Health checks, circuit breakers, timeouts, retry
- budgets
- No hot reloads - API driven config updates

Istio's contributions:

- Transparent proxying w/ SO_ORIGINAL_DST
- Traffic routing and splitting
- Request tracing using Zipkin
- Fault injection

Istio Architecture

Traffic is transparently intercepted and proxied. An App is unaware of Envoy's presence

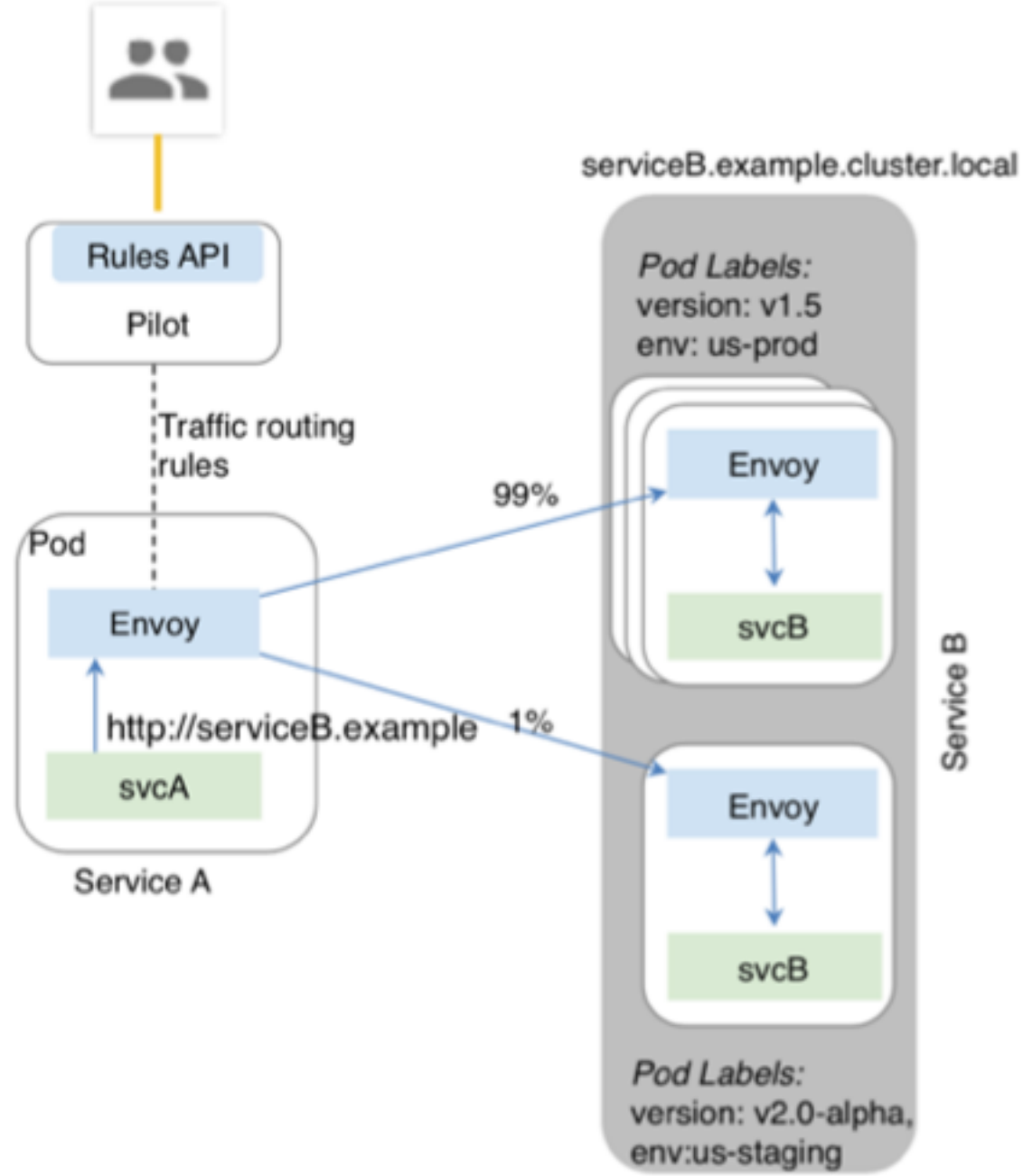


Traffic splitting

```
// A simple traffic splitting rule

destination: serviceB.example.cluster.local
match:
  source: serviceA.example.cluster.local
route:
- tags:
    version: v1.5
    env: us-prod
  weight: 99
- tags:
    version: v2.0-alpha
    env: us-staging
  weight: 1
```

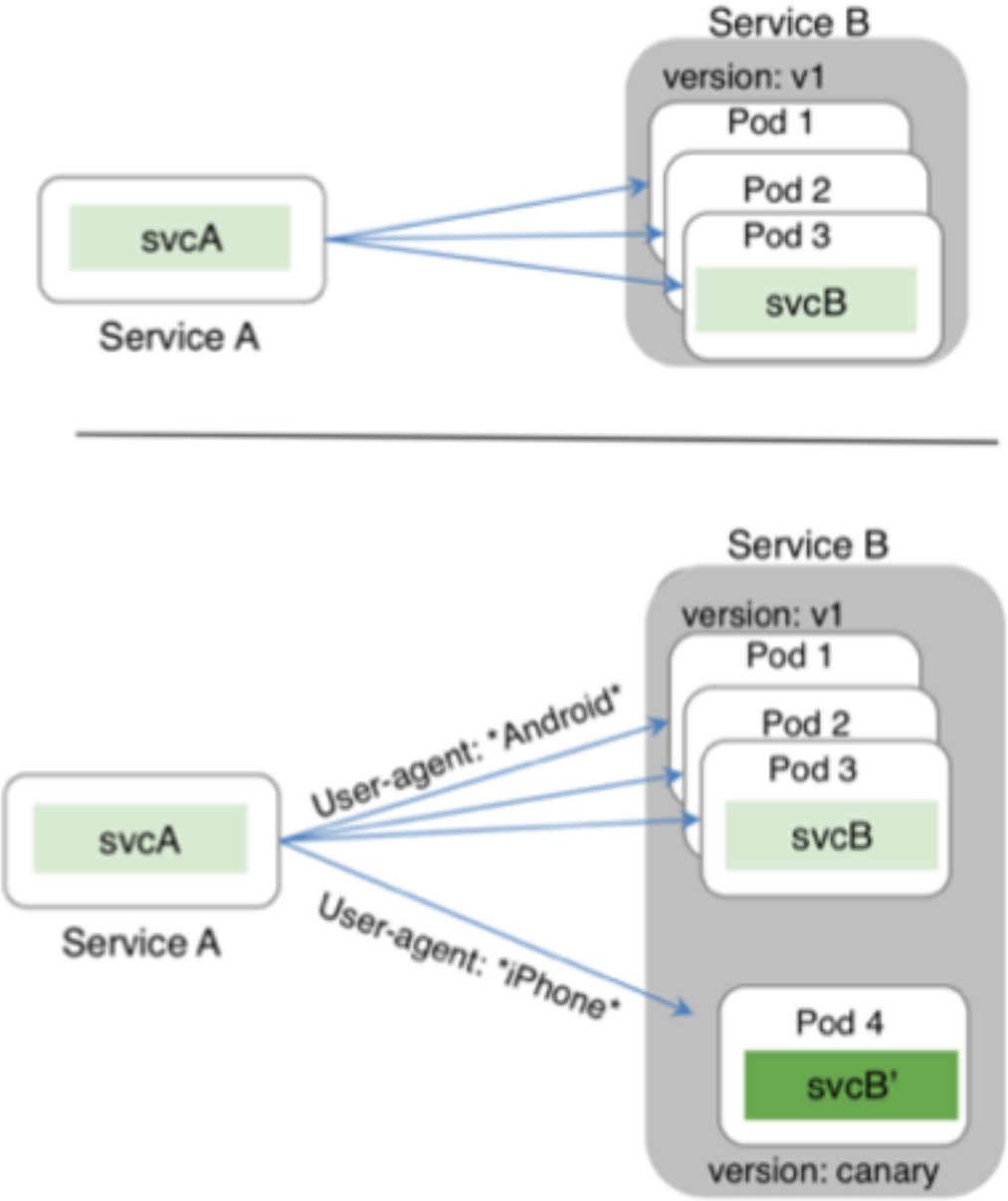
Traffic control is decoupled from infrastructure scaling



Traffic steering

```
// Content-based traffic steering rule

destination: serviceB.example.cluster.local
match:
  httpHeaders:
    user-agent:
      regex: ^(.*?;)?(iPhone)(;.*)?$
precedence: 2
route:
- tags:
    version: canary
```



Content-based traffic steering

Resiliency

Istio adds fault tolerance to your application without any changes to code

```
// Circuit breakers

destination: serviceB.example.cluster.local
policy:
- tags:
    version: v1
  circuitBreaker:
    simpleCb:
      maxConnections: 100
      httpMaxRequests: 1000
      httpMaxRequestsPerConnection: 10
      httpConsecutiveErrors: 7
      sleepWindow: 15m
      httpDetectionInterval: 5m
```

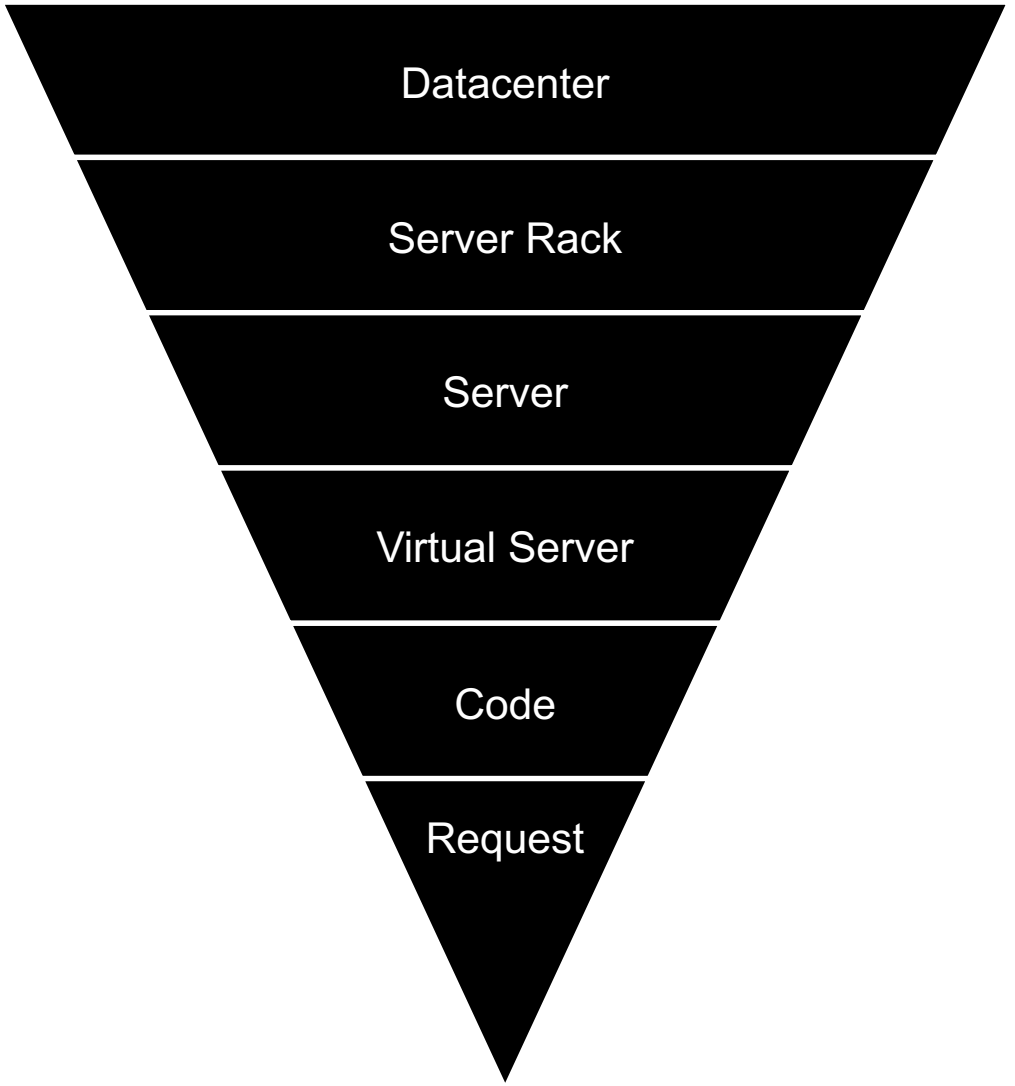
Resilience features

- Timeouts
- Retries with timeout budget
- Circuit breakers
- Health checks
- AZ-aware load balancing w/ automatic failover
- Control connection pool size and request load
- Systematic fault injection

Sometimes things don’t go as planned…

Many possibilities for failures

- Failures WILL happen
- Their root causes can be complex and affect different number of users
- Reduce the likelihood of correlated failures
- Recover quickly
- Proactive detection of problems before they happen

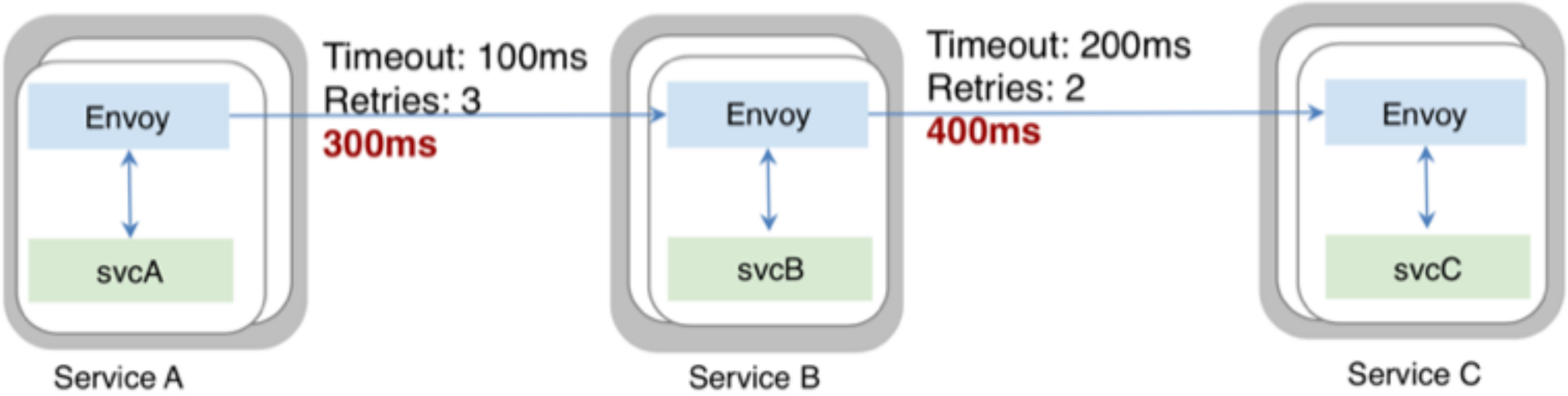


Resiliency testing

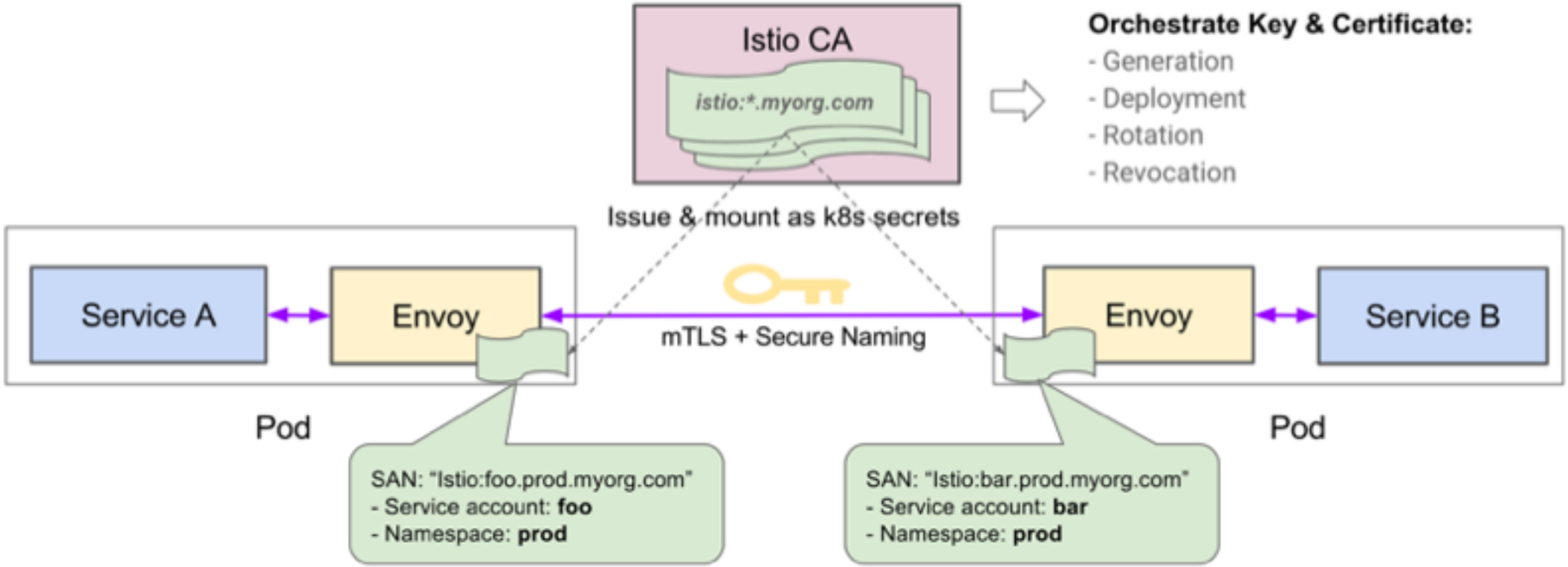
Systematic fault injection to identify weaknesses in failure recovery policies:

HTTP/gRPC error codes

Delay injection



Istio-Auth


Istio CA
istio:*.myorg.com
Orchestrate Key & Certificate:
- Generation
- Deployment
- Rotation
- Revocation
Issue & mount as k8s secrets
Service A
Envoy
mTLS + Secure Naming
Envoy
Service B
Pod
Pod
SAN: "Istio:foo.prod.myorg.com"
- Service account: foo
- Namespace: prod
SAN: "Istio:bar.prod.myorg.com"
- Service account: bar
- Namespace: prod

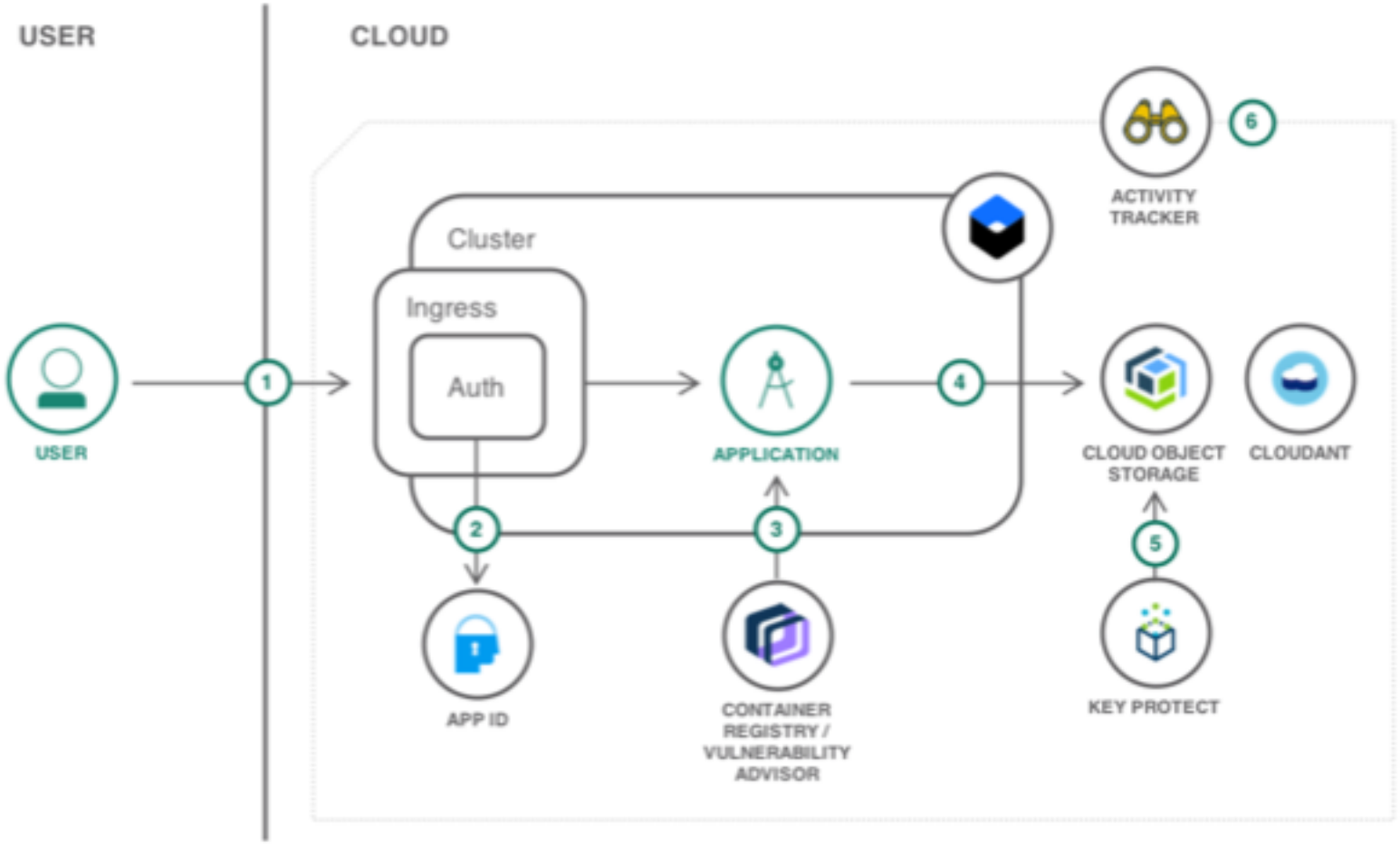
Conclusion

Conclusion

- Why containers?
- Docker containers
- Container orchestration
- Istio – beyond smart routing and resiliency

- Get started with Istio and IBM Cloud Kubernetes Service
  - https://developer.ibm.com/courses/all/get-started-istio-ibm-cloud-container-service

The architecture

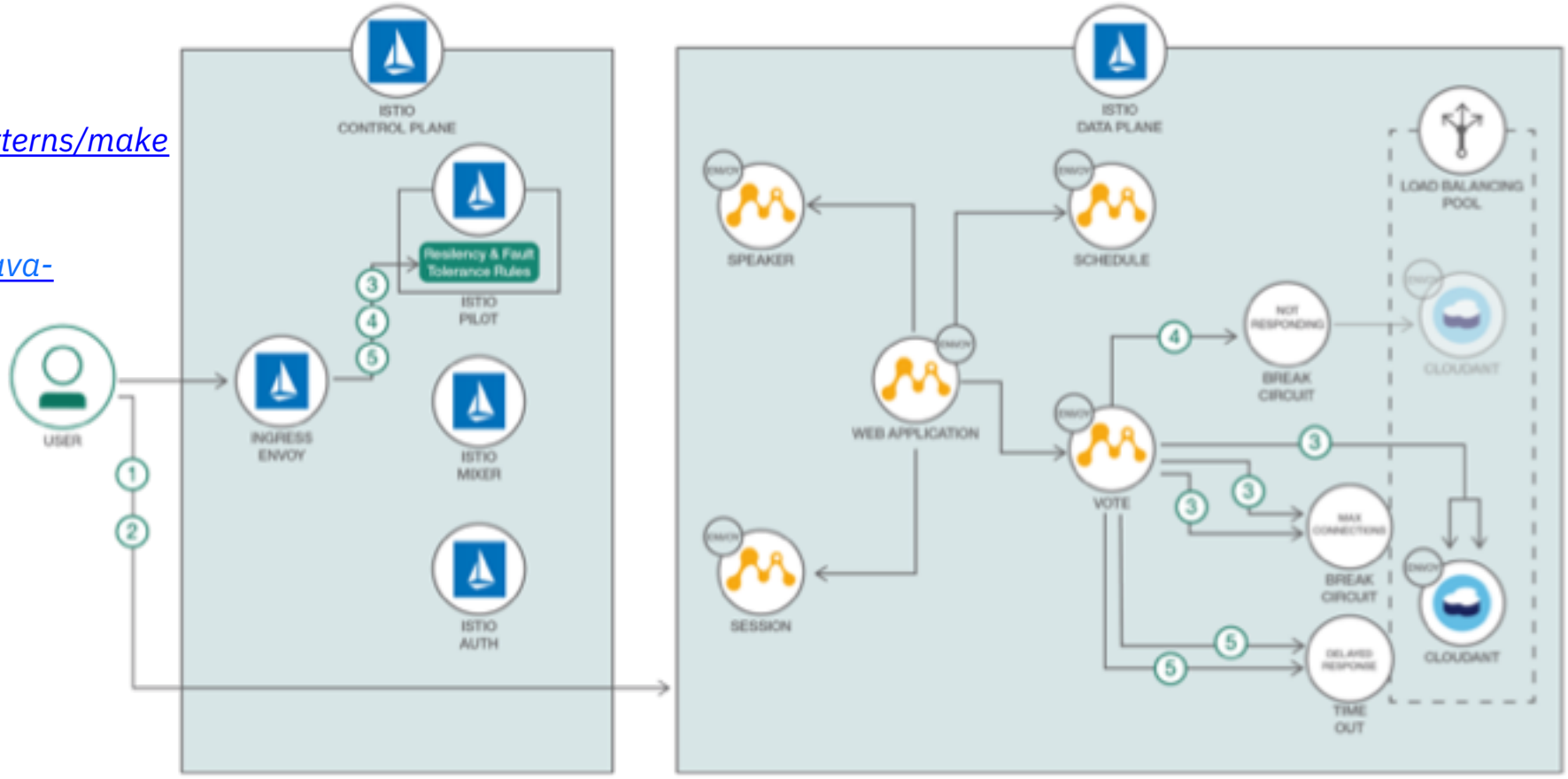


Java Based Code Example – aka pattern

Enable your Java microservices with advanced resiliency features leveraging Istio

Developer Works
Code: https://developer.ibm.com/code/patterns/make-java-microservices-resilient-with-istio/

Github: https://github.com/IBM/resilient-java-microservices-with-istio



Resources

- Docker tutorial - https://docs.docker.com/get-started/
- Kubernetes tutorial - https://kubernetes.io/docs/tutorials/kubernetes-basics/
- The Evolution of Linux Containers and Their Future
  - https://dzone.com/articles/evolution-of-linux-containers-future
- Introduction to container orchestration
  - https://www.exoscale.ch/syslog/2016/07/26/container-orch/
- TNS Research: The Present State of Container Orchestration
  - https://thenewstack.io/tns-research-present-state-container-orchestration/
- Large-scale cluster management at Google with Borg
  - https://research.google.com/pubs/pub43438.html
- Call for code: http://developer.ibm.com/callforcode
- The other SF City events: https://developer.ibm.com/code/community/cities/san-francisco/

Thank you

twitter.com/blumareks

github.com/blumareks

developer.ibm.com/code