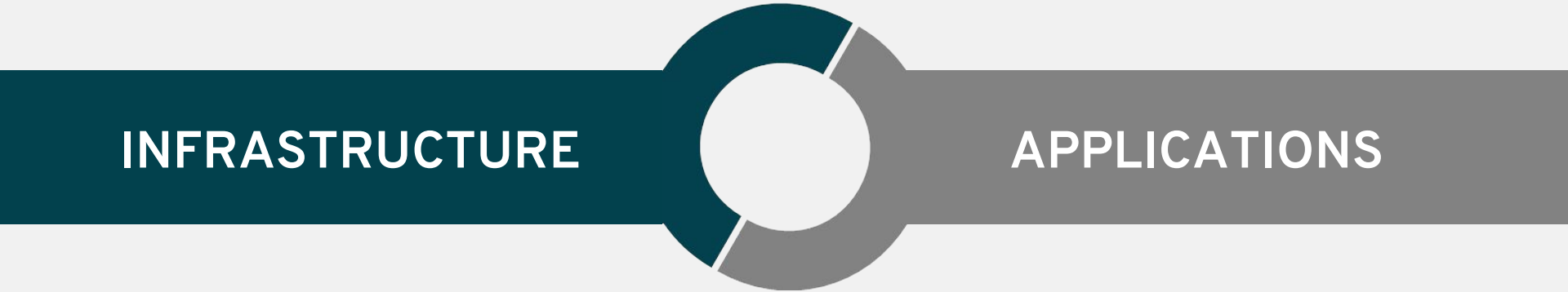

OPENSHIFT 3.7 and beyond
redhat.

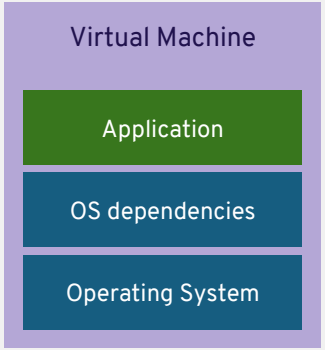
Qu’est qu’un conteneur ?

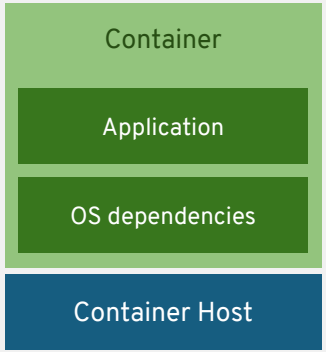


- Processus sur un système d’exploitation
- Plus simple, léger et dense des VMs
- Portable entre environnements

- Applications et toutes ses dépendances
- Déploiement dans un environnement en quelques secondes
- Facile d’accès et facile à partager

Machine Virtuelle vs Conteneur

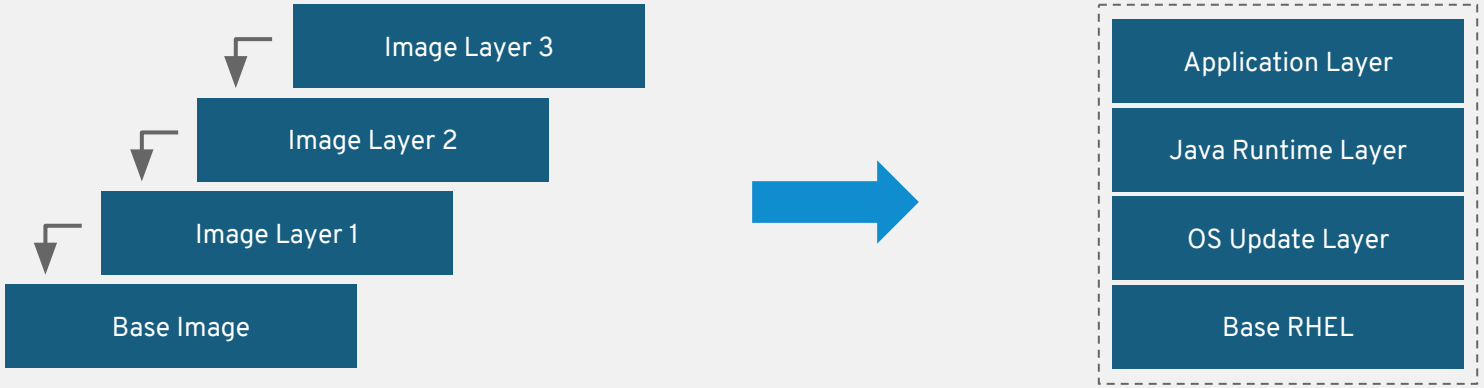




- ✚ Isolation par VM
- ▬ OS complet
- ▬ Compute fixe
- ▬ Mémoire Fixe
- ▬ Utilisation de ressources élevée

- ✚ Isolation par conteneur
- ✚ Noyau de l'OS partagé
- ✚ Compute à la demande
- ✚ Mémoire à la demande
- ✚ Faible utilisation des ressources

Conteneurs – plusieurs couches indépendantes



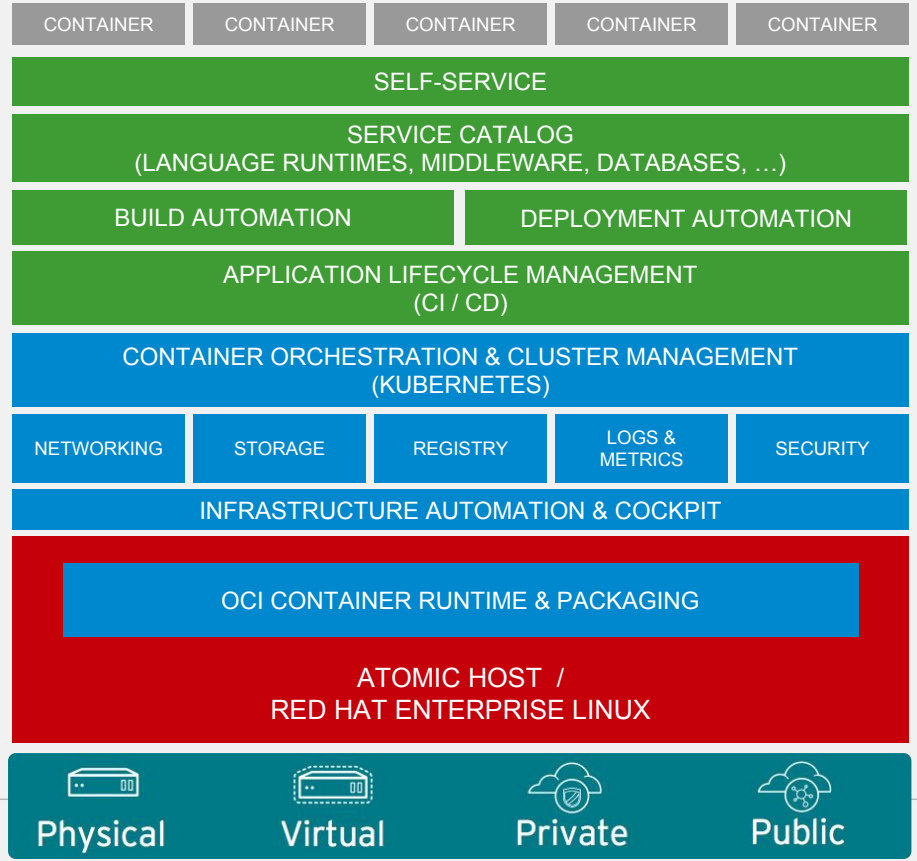
Container Image Layers

Example Container Image


OPENSHIFT
redhat.

OpenShift = Enterprise Kubernetes+

Bâtir, déployer et gérer des applications en conteneurs

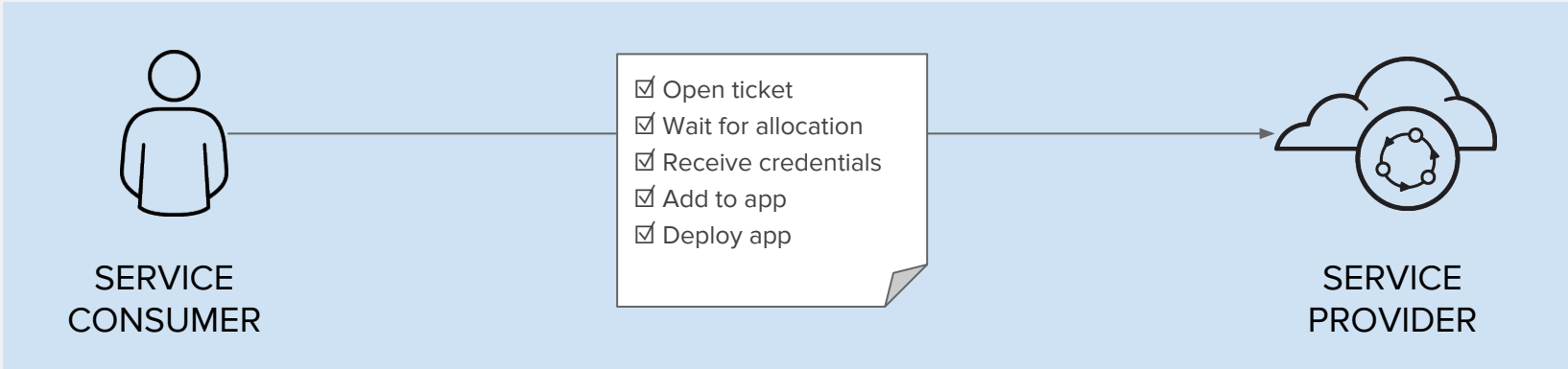


OCP 3.7 - Thèmes

- Services Multi-Cloud
  - Service Catalog/Broker
- Partenariat Amazon Web Services
  - Services AWS – Livrés et maintenus par AWS.
- Déploiement d’application sur plusieurs IAAS géré par Ansible
  - Déploiement par phases
  - Ansible Playbook Bundles « APB »
- API de gestion et monitoring par événement
  - Prometheus et CloudForms
- Sécurité
  - NetworkPolicy, Auditing, Node API Restrictions, RBAC

SERVICE BROKER

WHY A SERVICE BROKER?



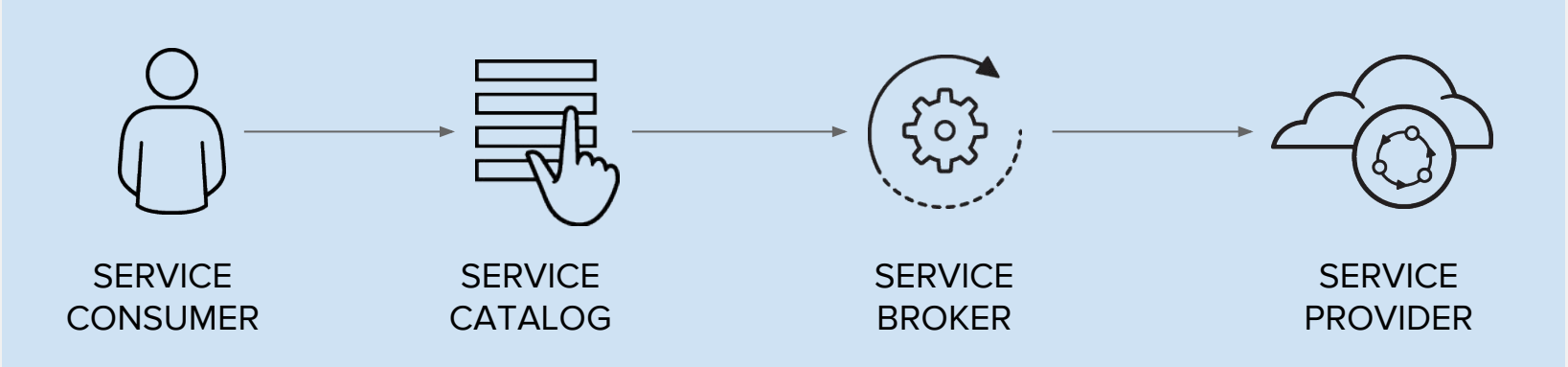
Manual, Time-consuming and Inconsistent

OPEN SERVICE BROKER API™

A multi-vendor project to standardize how services are consumed on cloud-native platforms across service providers

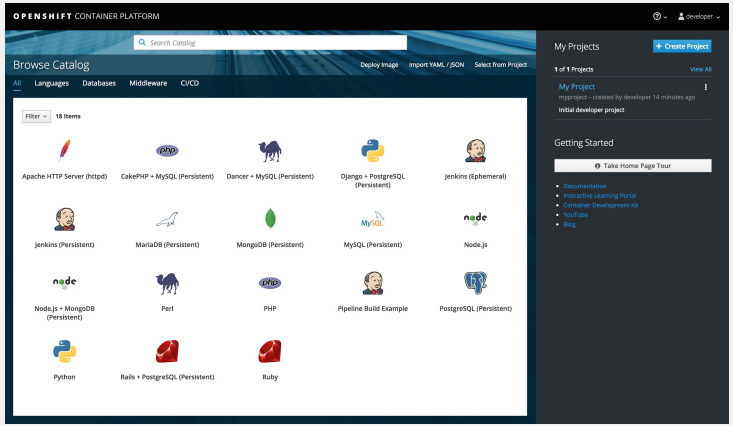
FUJITSU Pivotal IBM redhat. Google SAP

WHAT IS A SERVICE BROKER?

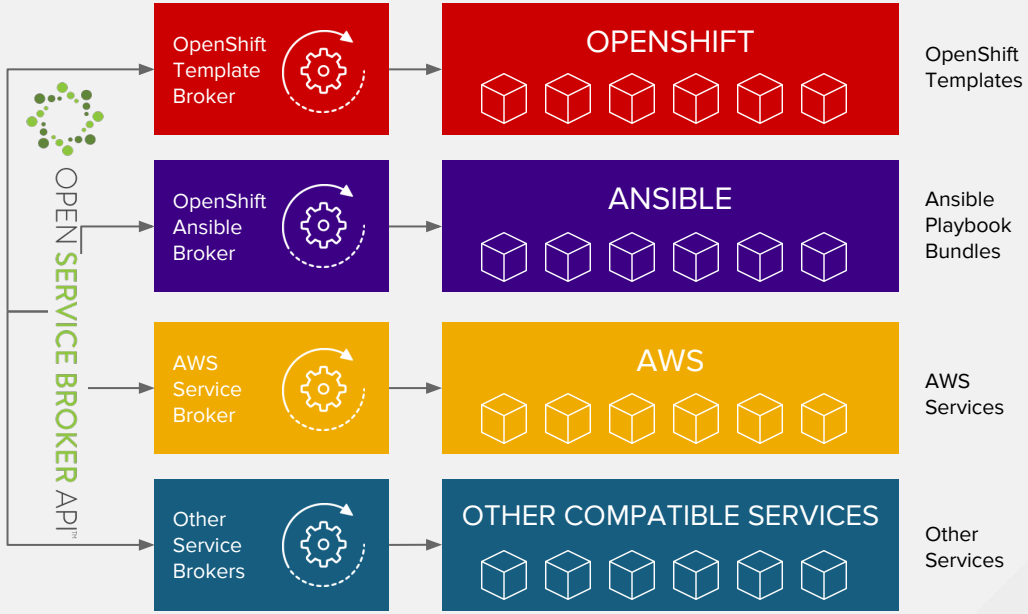


Automated, Standard and Consistent

OPENSHIFT SERVICE CATALOG



OPENSHIFT SERVICE CATALOG



redhat.

DÉMO #1

. Service Broker

Installation

Feature(s): [Make the installer modular to allow playbooks to run independently.]

Description: The installer has been enhanced to allow admins to install specific components.

How it Works: By breaking up the roles and playbooks, we allow for a better targeting of ad hoc administration tasks.

Sample ordering of playbooks:

playbooks/byo/openshift-checks/pre-install.yml
playbooks/byo/openshift-etcd/config.yml
playbooks/byo/openshift-nfs/config.yml (if required)
playbooks/byo/openshift-loadbalancer/config.yml (if required)
playbooks/byo/openshift-master/config.yml
playbooks/byo/openshift-master/additional_config.yml
playbooks/byo/openshift-node/config.yml
playbooks/byo/openshift-glusterfs/config.yml (if required)
playbooks/byo/openshift-cluster/openshift-hosted.yml
playbooks/byo/openshift-cluster/openshift-metrics.yml (if required)
playbooks/byo/openshift-cluster/openshift-logging.yml (if required)
playbooks/byo/openshift-cluster/service-catalog.yml (if required)
playbooks/byo/openshift-management/config.yml (if required)

Installation

Feature(s): Both Install AND Configure CFME 4.6 from the OpenShift installer

Description: CloudForms 4.6 will be fully supported running on OCP 3.7 as a set of containers.

How it Works: We have automated the installation experience to the level we have only done for metrics and logging in the past. Now management (CloudForms) is an available API endpoint on all OpenShift clusters that choose to use it. More cluster admins will be able to leverage CloudForms and begin experiencing the insight and automations available to them in the full OpenShift container platform.

- To install CFME 4.6:

 # ansible-playbook -v -i <YOUR_INVENTORY> playbooks/byo/openshift-management/config.yml

- To configure CFME 4.6 to consume the OpenShift installation it is running on:

 # ansible-playbook -v -i <YOUR_INVENTORY> playbooks/byo/openshift-management/add_container_provider.yml

- You can also automate the configuration of the provider to point to multiple OpenShift clusters:

 # ansible-playbook -v -e container_providers_config=/tmp/cp.yml playbooks/byo/openshift-management/add_many_container_providers.yml

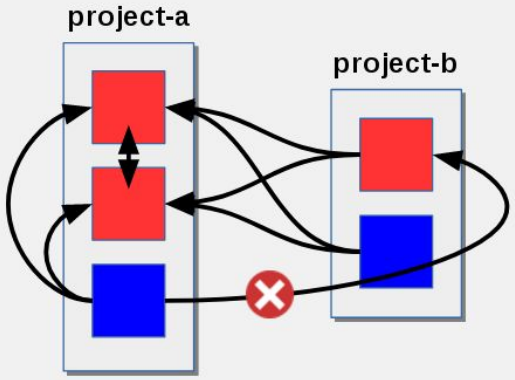
Networking

Feature(s): [Network Policy]

Description: Optional plugin specification of how selections of pods are allowed to communicate with each other and other network endpoints.

How it Works: Fine-grained network namespace isolation using labels and port specifications

- what ingress traffic is allowed to any pod, from any other pod
- on specific ports
- including traffic from pods located in other projects



Policy applied to namespace: **project-a**

```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-to-red
spec:
  podSelector:
    matchLabels:
      type: red
  ingress:
  - {}
```

DÉMO #2

. Network Policy

Metrics

Feature(s):

- **Introducing Prometheus (Tech Preview)**
- **PS: Hawkular is still the supported Metrics stack**

Description:

OpenShift Operators deploy Prometheus on an OCP cluster, collect Kubernetes and Infrastructure metrics, get alerts. Operators can see and query metrics and alerts on Prometheus web dashboard. Or They can bring their own Grafana and hook it up to Prometheus.

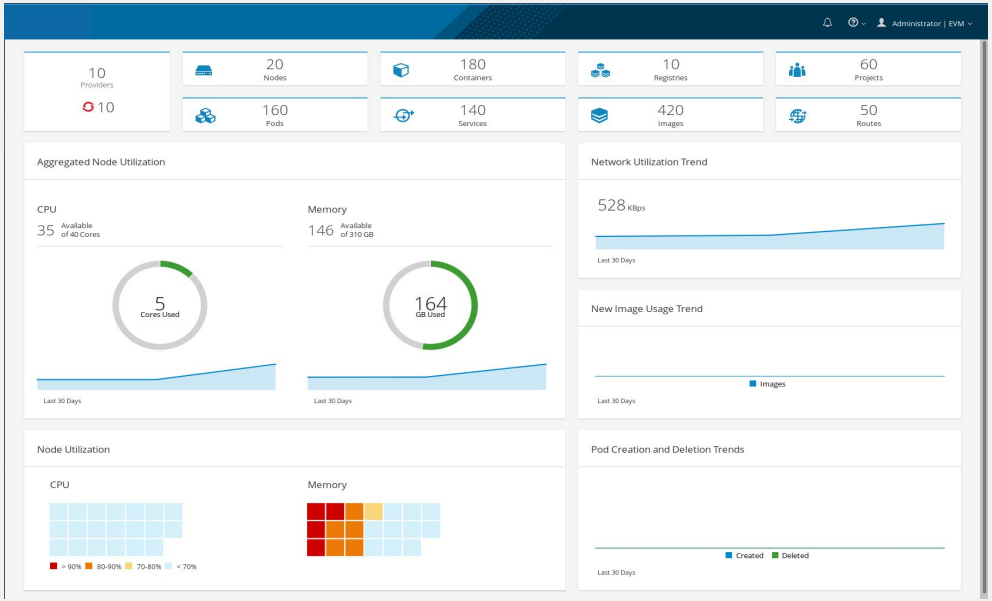
How it Works:

- `New OpenShift installer playbook for installing Prometheus (2.0) server, alert manager and oAuth-proxy`
- `Deploys Statefulset comprising server, alert-manager, buffer and oAuthProxy in front and a PVC one for server and one for alert manager`
- `Alerts can be created in a rule file and selected via inventory file`

DÉMO #3

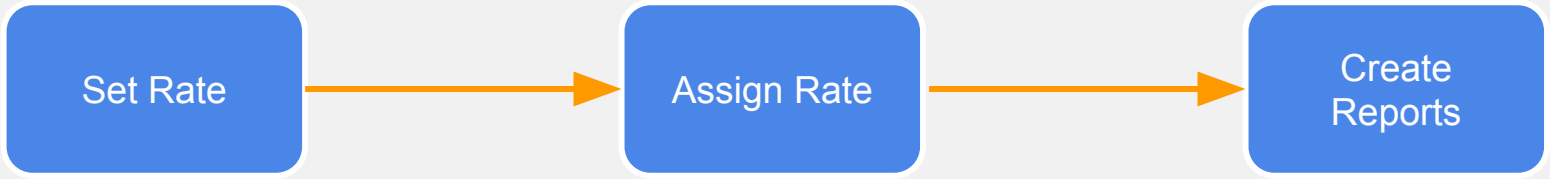
. Prometheus

CloudForms OpenShift Provider - CF 4.6



- Containerized (aka podified) CF
  - Installed on OpenShift with Installer or Template
- OpenShift Provider for Prometheus
  - Metrics and Alerts in container dashboards
  - Alerts management
- Chargeback
  - By Allocation (vs. Usage)

CF 4.6 and Chargeback



- By Usage
- By Allocation NEW

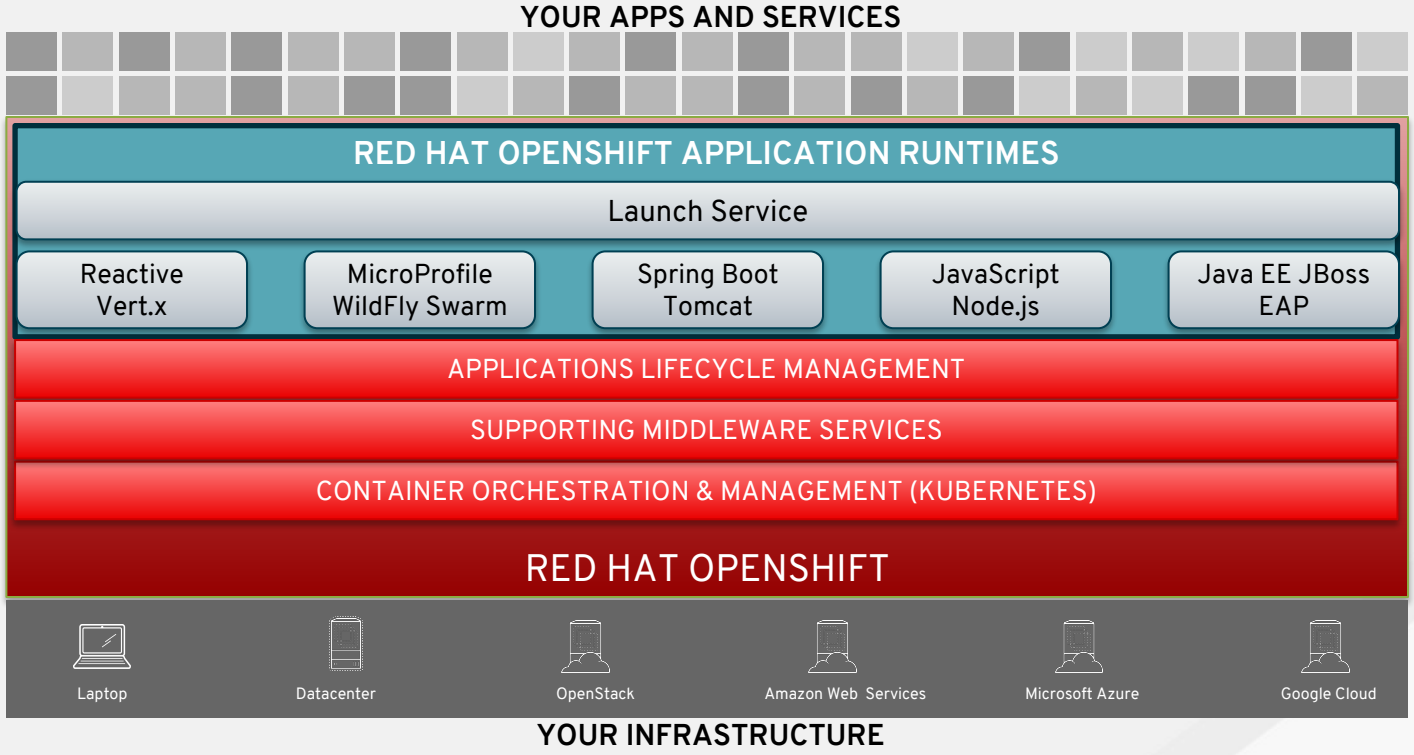
- The Enterprise
- By Container Provider
- By Image Label
- By Image Tag

- By Project
- By Image/Project

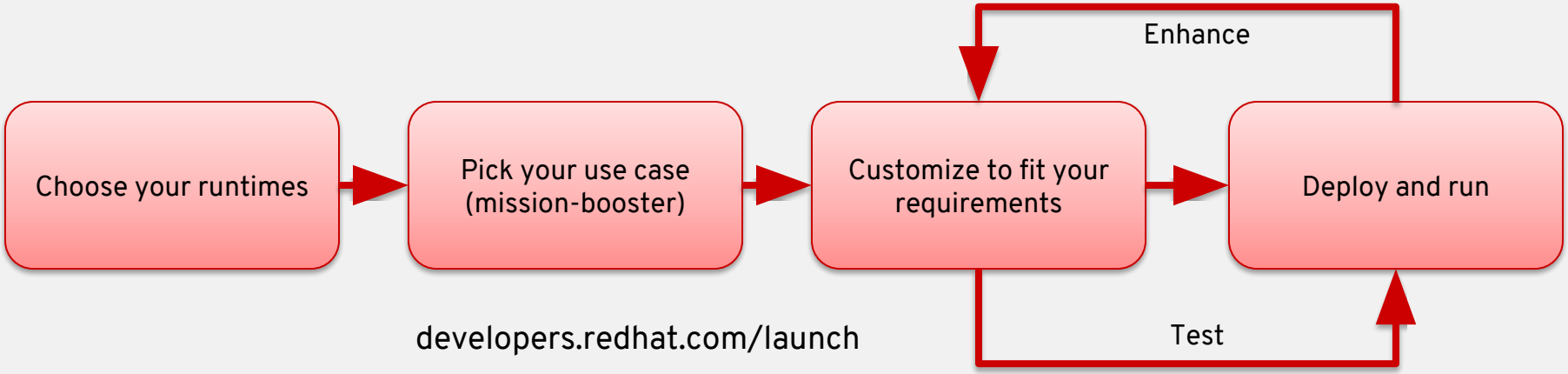
RED HAT OPENSHIFT APPLICATION RUNTIMES

Providing curated set of integrated runtimes and frameworks *that standardizes Cloud Native App Dev*

- ✓ Simplified development
- ✓ Strategic flexibility
- ✓ DevOps automation



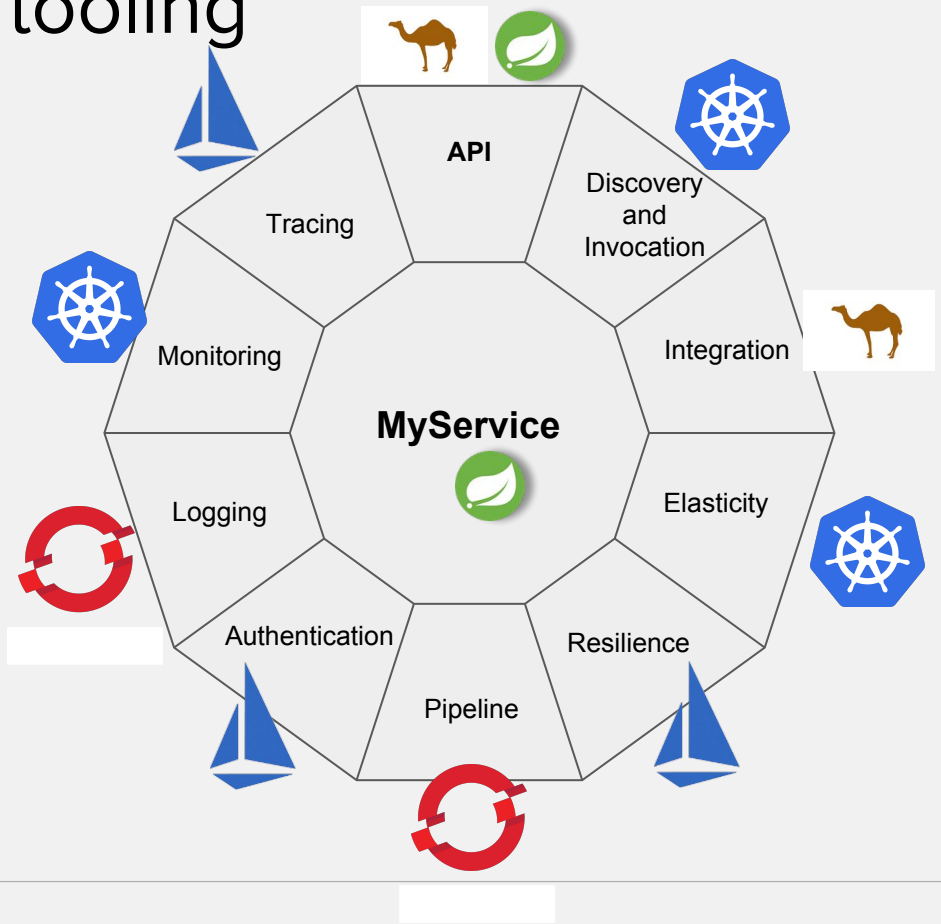
RHOAR GETTING STARTED EXPERIENCE



MISSION-BOOSTER: A working application-implementation showcasing different pieces of cloud native application.

MICROSERVICES INFRASTRUCTURE: ISTIO SERVICE MESH

Microservices tooling



Istio - Sail

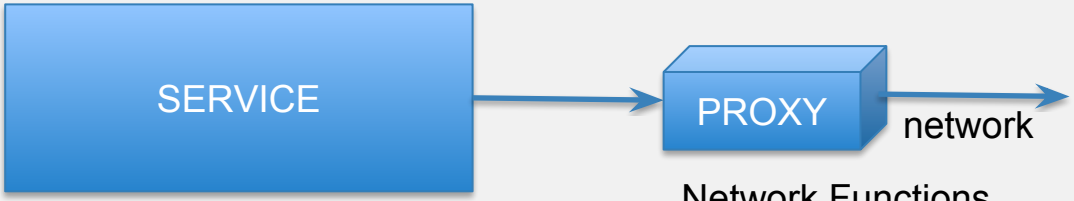
(Kubernetes - Helmsman or ship's pilot)

What if ?

Network Functions

- Authentication
- Routing
- Telemetry
- …

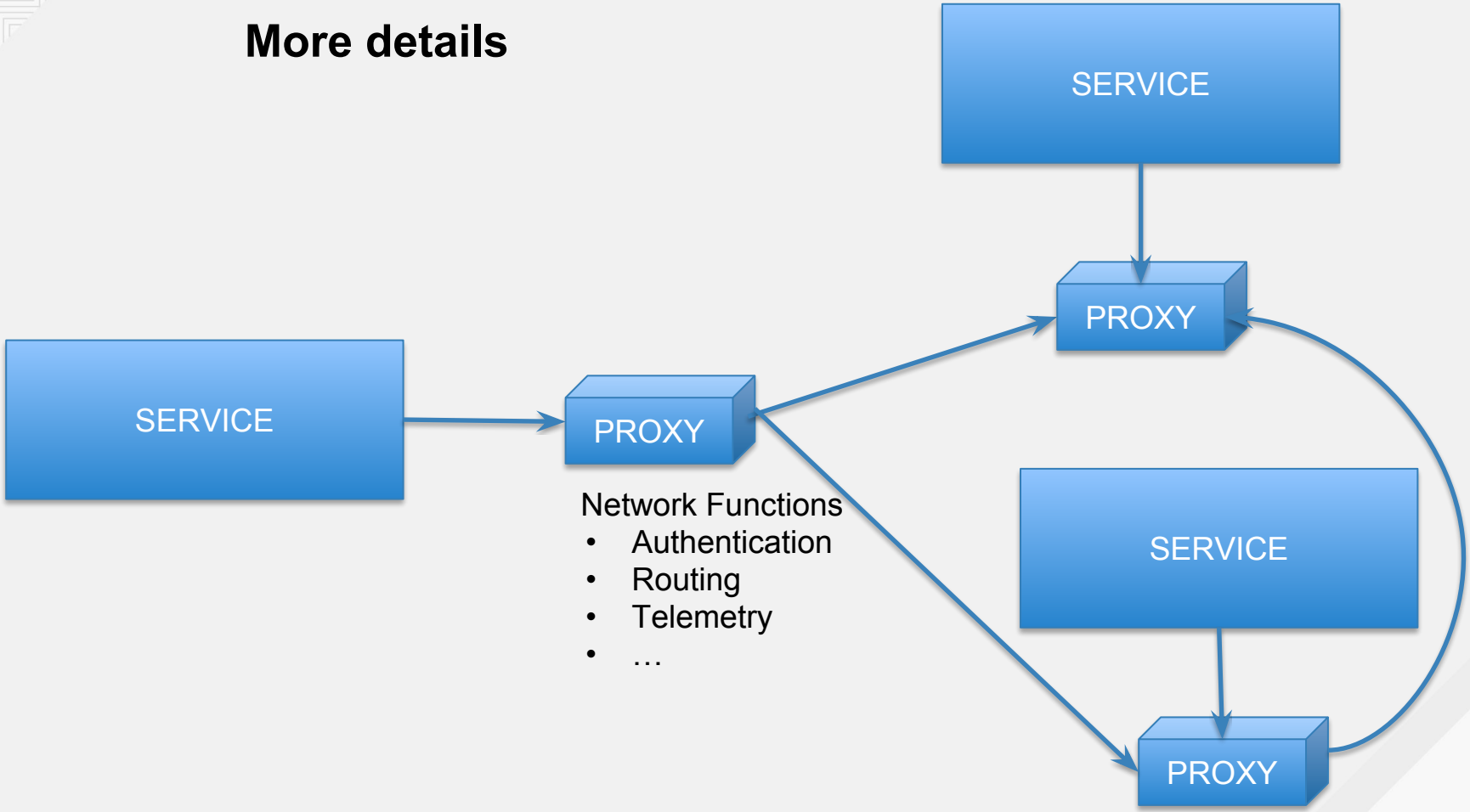
What if ?



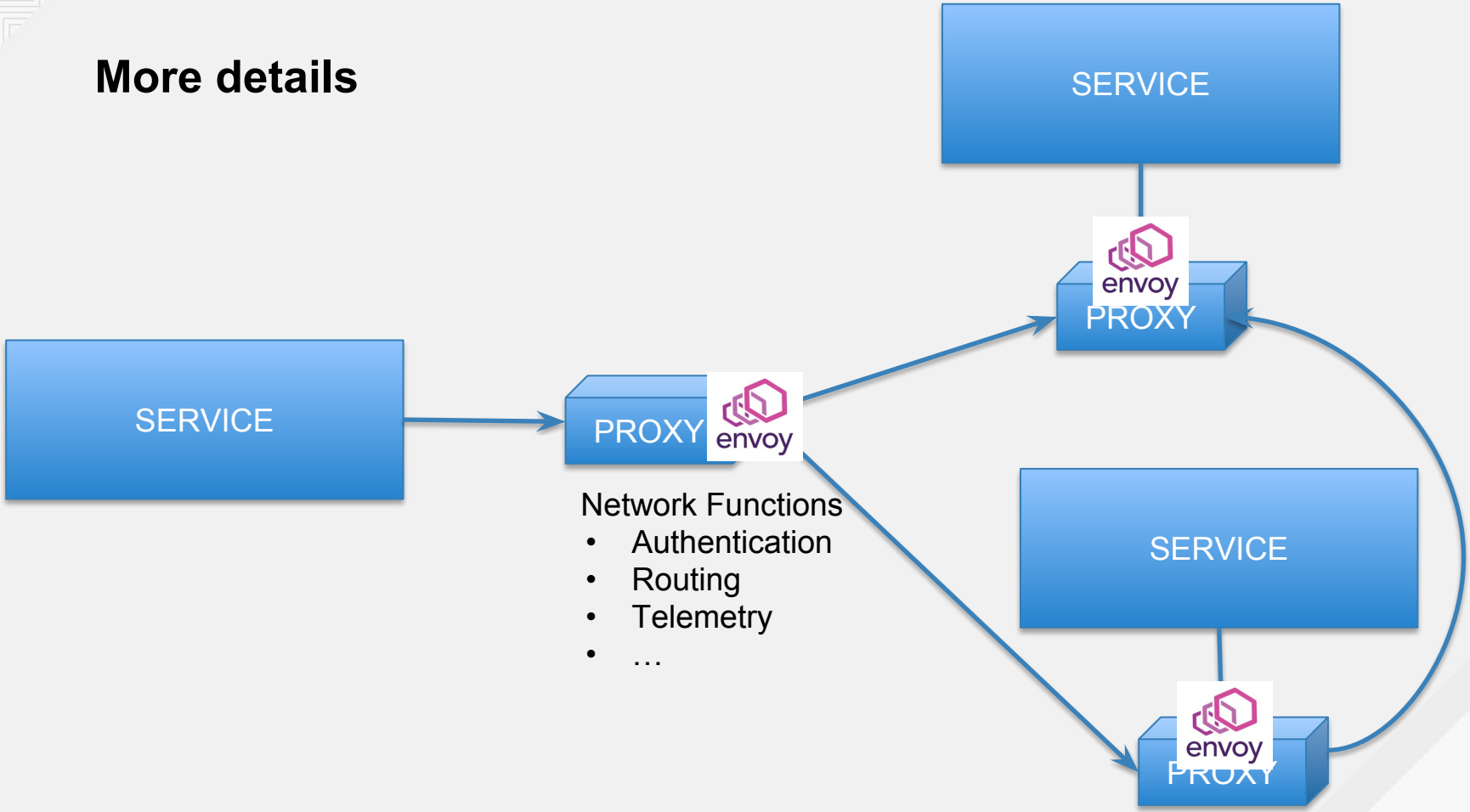
Network Functions

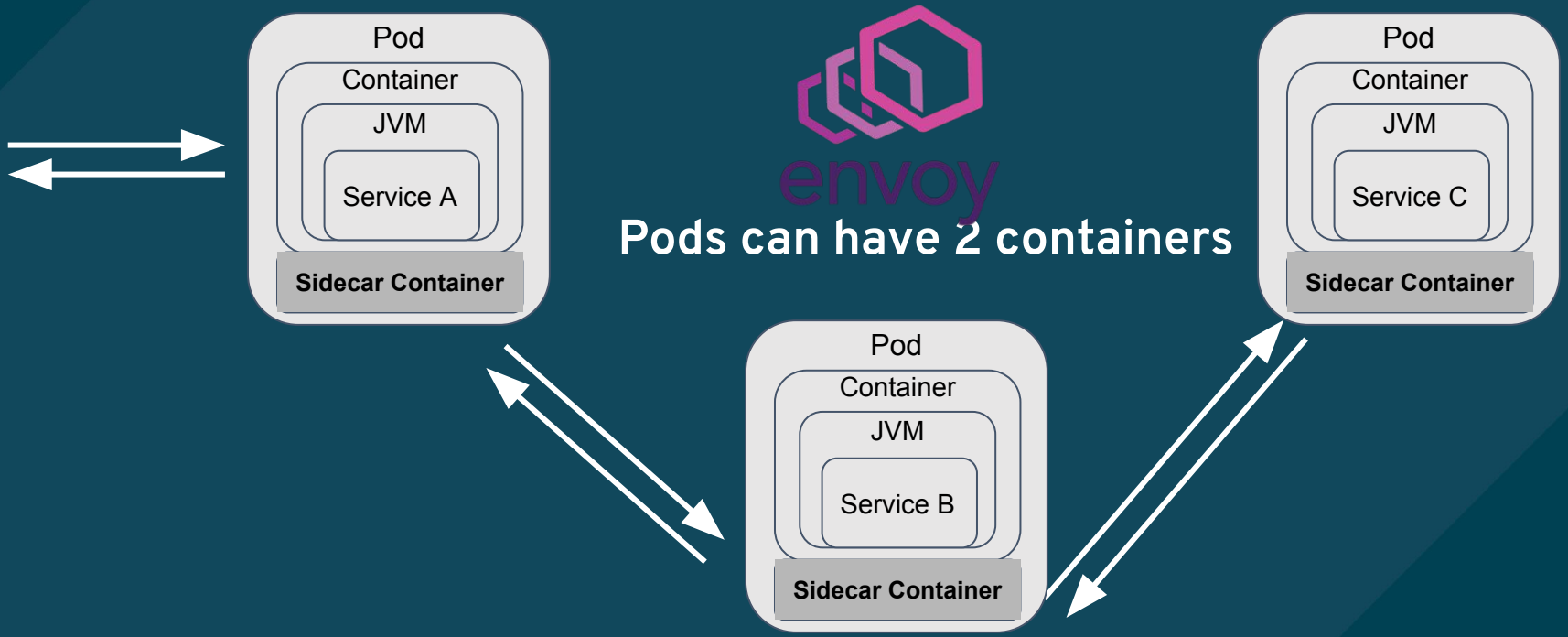
- Authentication
- Routing
- Telemetry
- …

More details



More details




Pod
Container
JVM
Service A
Sidecar Container
envoy
Pods can have 2 containers
Pod
Container
JVM
Service C
Sidecar Container
Pod
Container
JVM
Service B
Sidecar Container

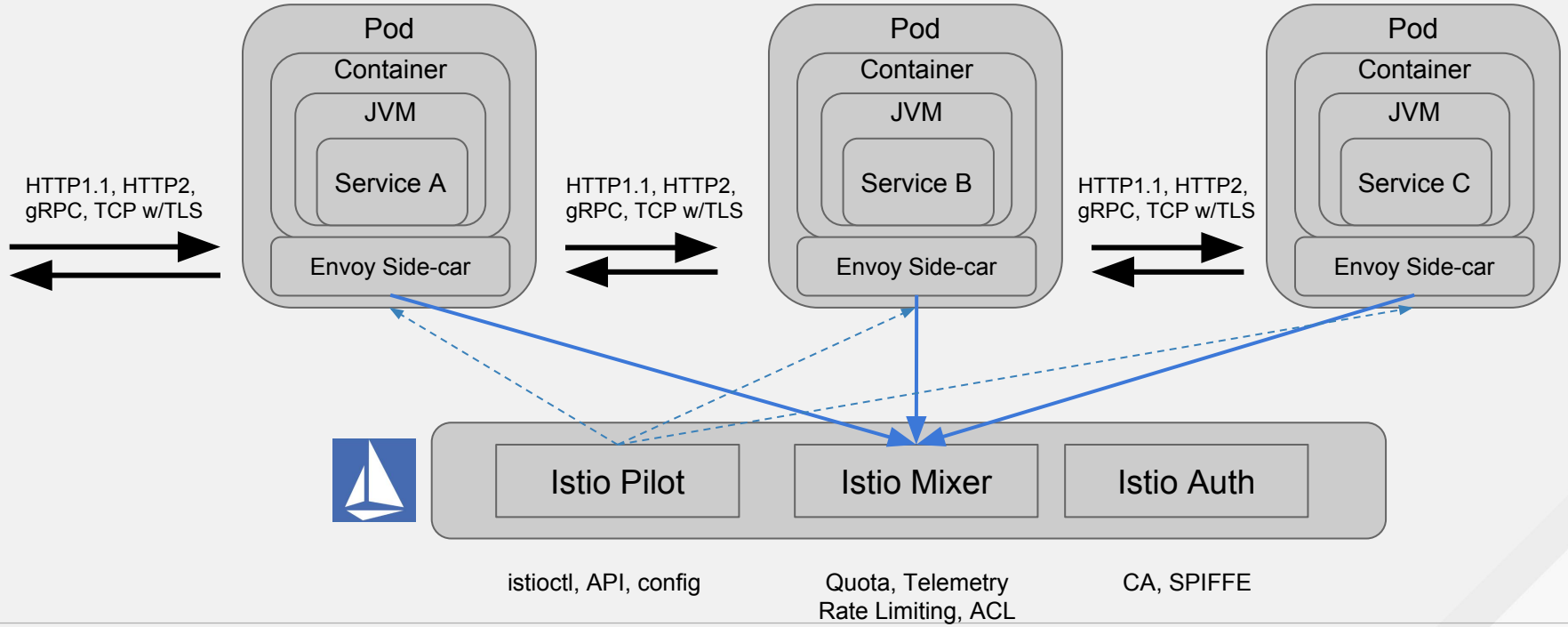
- Service Proxy
- Highly parallel, non-blocking
- L3/4 network filter
- Out of the box L7 filters
- HTTP 2
- Baked in service discovery/health checking
- Metadata based routing and load balancing
- Stats, metrics, tracing
- Dynamic configuration

Next Generation Microservices - Service Mesh

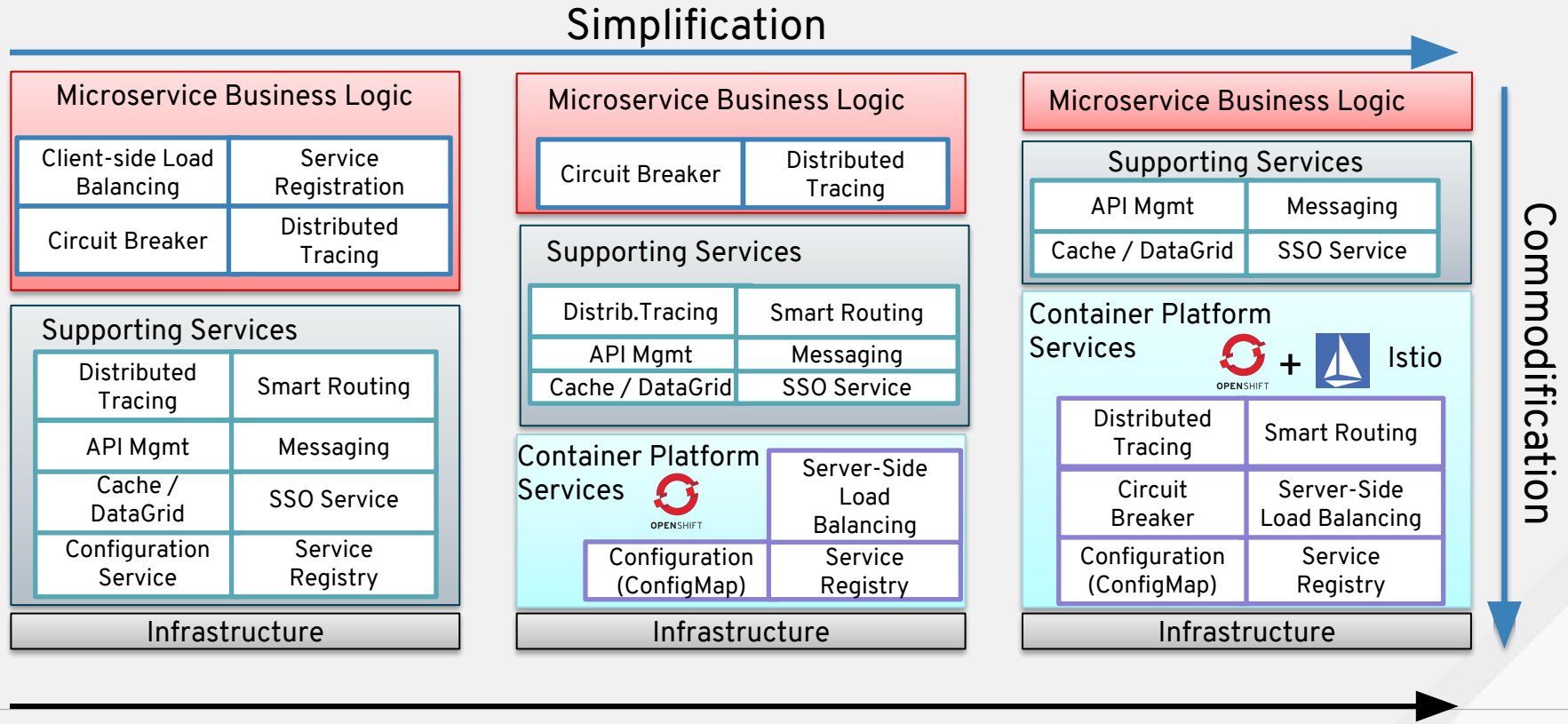
Code Independent (Polyglot)

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

Istio Control Plane



EVOLUTION OF MICROSERVICES



DÉMO #4

. ISTIO