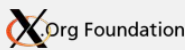
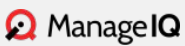




# Openshift, Microservices and more...

**Ugo Landini**  
Solution Architect  
ulandini@redhat.com

**Samuele Dell'Angelo**  
Solution Architect  
samuele@redhat.com



389 project | aeolus-configure | anaconda | attr | Audrey | Augeas | AuthHub | Autofs | Avahi | Beaker | Boxes | Btrfs | CacheFS | Cairo | candlepin | certmonger | CIFS | Cluster 3 | cobbler | colord | Completely Fair Scheduler | Conductor Control Group Configuration Library | Corosync | CRaSH | Crypto API | Cumin | Cygwin | D-Bus | Dashbuilder | Data Grids for the Java Platform | device-mapper | DeviceKit | DistributionUtilities | dim | Dogtag | DPK | Dracut | Editline Library EDS | EJB 3 | elfutils | elfutils | Embedded Jopr | Evolution | Evolution-exchange | eXo JCR | ext3 | ext4 | fence-agents | fence-virt | Flannel | fontconfig | func | Gamin | Gateln Portal | gcc | gcc | gcj | gdb | GFS | GFS2 | glib | gss-proxy GTK+ | gvfs | gzip | HTCondor | ID Utils | imagefactory | IPSec-Tools | iptraf-ng | Jandex | Java | JGroups | Jreadline | JRuby | JSFUnit | JUDDI | Katello | Kerberos | Kimchi | Koji | libguestfs | libibverbs | libminidump | libnotify | libqb libstoragemgmt | libuser | libvirt | libxml2 | libxslt | Linux Audit | Linux Infiniband Project | Linux Kernel | Linux PAM | linux-nfs | Luci | LVM2 | LVM2 | lynx | ManagerIQ | Matahari | Maven Integration for Eclipse | Maven Integration for Web Tools Platform | Mock | mod\_cluster | moVirt | nautilus | netfilter | NetworkManager | NFS-Ganesha | nmap | Open vSwitch | openais | OpenDaylight | OpenJDK | OpenSSH | OpenSSL | OPNFV | OProfile | oVirt | oVirt Engine | oVirt Node | OVMF | Oz pacemaker | PackageKit | PackageKit | pango | PAPI | Paradyne | patchutils | Performance Co-Pilot | PicketBox | PicketLink | piglit | pixman | polkit | Polkit Qt | Poppler | Portlet Bridge | PostgreSQL | PressGang psmisc | Pulp | PulseAudio | punji | pynfs | QEMU | Red Hat Update Agent | RESTEasy | rgmanager | RHQ | rpm | rpmgrill | RSYSLOG | Samba | Samba | SETroubleshoot | ShrinkWrap | slapi-ns | Smokestack | Spacewalk Spherical | Spice | Spice-gtk | Spice-protocol | Stilts | suds | SWI Prolog | SwitchYard | syslog-ng | System Security Services Daemon | systemd | SystemTap | The Foreman | Thin Crust | UberFire | uclm | udisks | udisks | UPower | util-linux | Valgrind | vc-dwim | vdagent | vdsms | vfs | Virt-clone | Virt-image | Virt-install | Virt-manager | Virt-viewer | Virtual memory manager | Wallaby | Wayland | Wise | XFS | xinput | XNIO | Xorg | XrandR | yum

# Containers and Orchestration



# Containers: standardization, automation e dependency management

User Experience: increased productivity for developers

## A Development / Deployment Time:



- Automation
- Continuous Integration / Delivery
- Configuration Management
- Service / API design
- Rigorous Testing



- Dependency management
- Design for eventual consistency



- Artifact repositories

## A Runtime:



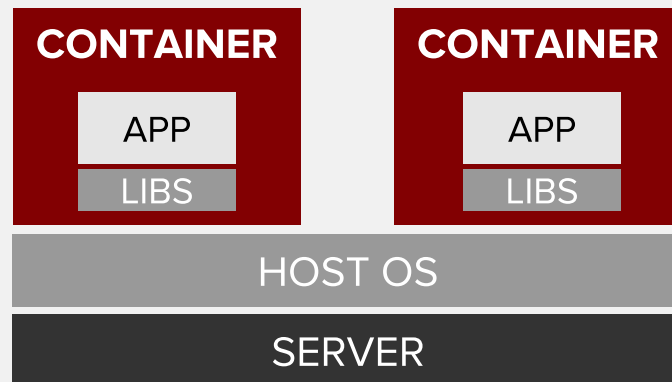
- Standardization
- Isolation
- Service Discovery
- Load Balancing
- Circuit Breaker, Fallback
- Health checks & automated recovery
- Distributed logging
- Tracing
- Infrastructure Monitoring



# Containers pack application with dependencies.

High density and efficiency

- Simple and portable deployment
- Immutable
- Isolated from host OS



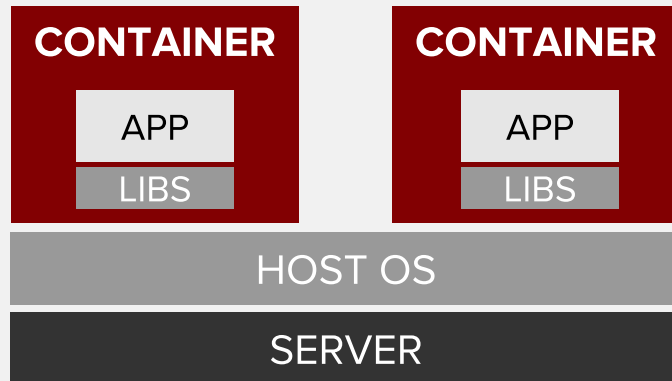
# DEMO BUILD YOUR OWN CONTAINER IN 50 lines



# Containers are Linux processes

High density and efficiency

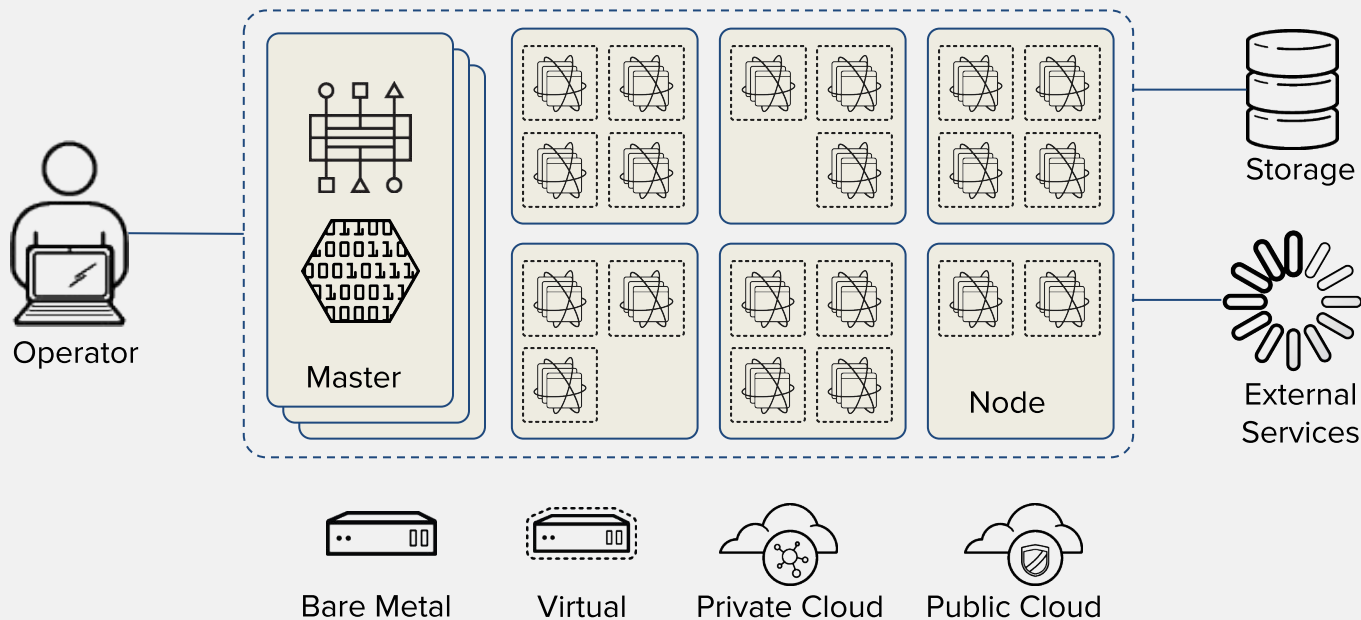
- Secure and isolated
- *Looks like* it is running in its own environment
- It's possible to run *hundreds* of container on a single machine
- Functionalities are in the **kernel**
- Docker is *only* a format





# Kubernetes is a container orchestrator for applications

From greek “*pilota*”: root for “governatore” (from latin: *gubernator*)







# Kubernetes adds fundamental functionalities for MSAs (not only!)

**Kubernetes** is the open evolution of Borg, the system internally used by Google to orchestrate and scale containers.

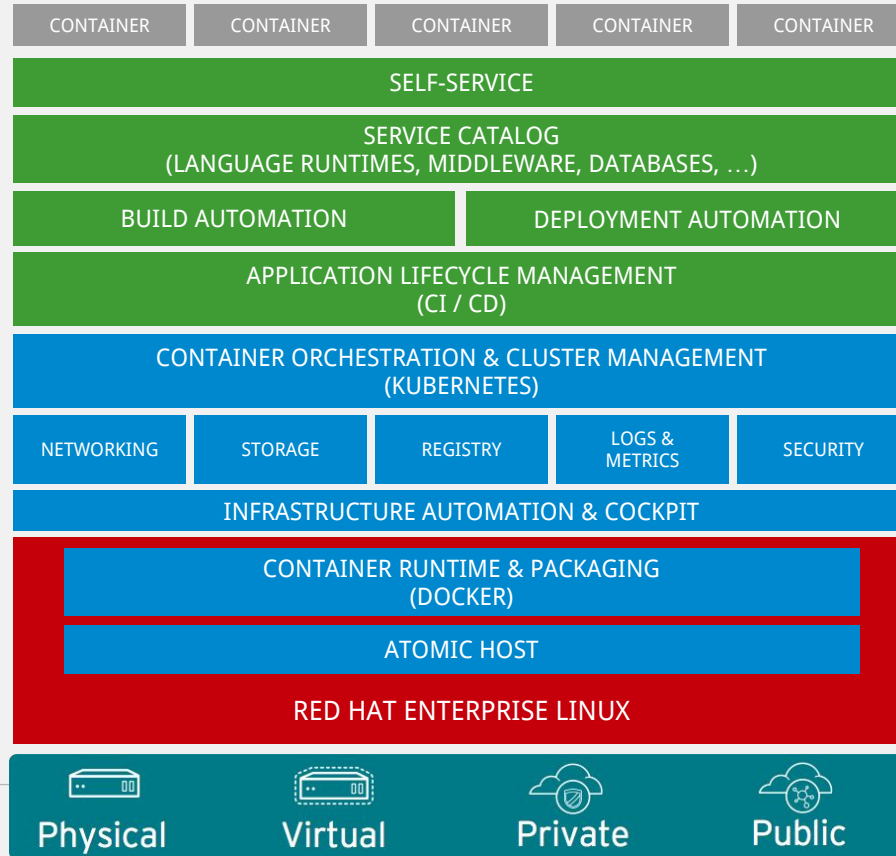
## A Development / Deployment Time:

- Automation
- Continuous Integration / Delivery
- Configuration Management
- Service / API design
- Rigorous Testing
- Dependency management
- Design for eventual consistency
- Artifact repositories

## A Runtime:

- Standardization
- Isolation
- Service Discovery
- Load Balancing
- Circuit Breaker, Fallback
- Health checks & automated recovery
- Distributed logging
- Tracing
- Infrastructure Monitoring

# OpenShift Container Platform is the **ENTERPRISE** version of Kubernetes





# OpenShift Container Platform adds services for developers

Developer has only to code and deploy.

## A Development / Deployment Time:

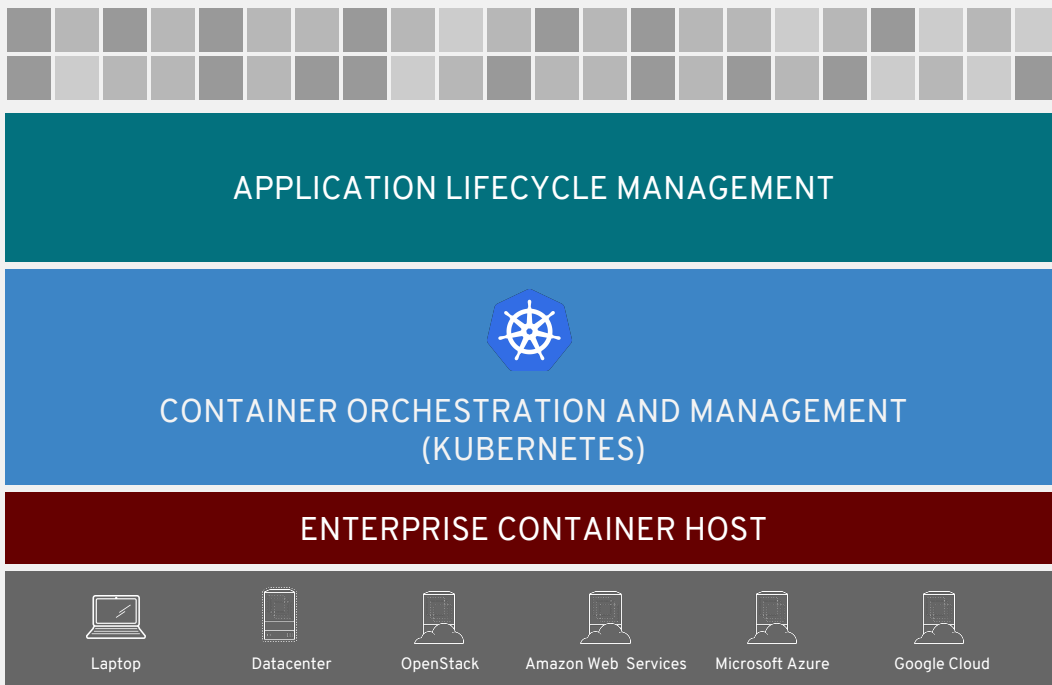
- Automation
- Continuous Integration / Delivery
- Configuration Management
- Service / API design
- Rigorous Testing
- Dependency management
- Design for eventual consistency
- Artifact repositories

## A Runtime:

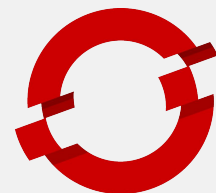
- Standardization
- Isolation
- Service Discovery
- Load Balancing
- Circuit Breaker, Fallback
- Health checks & automated recovery
- Distributed logging
- Tracing
- Infrastructure Monitoring

# Openshift Technical Introduction

# OPENSIFT CONTAINER PLATFORM



QUALSIASI  
CONTAINER



**RED HAT**  
OPENSIFT

QUALSIASI  
INFRASTRUTTURA



Laptop



Datacenter



OpenStack



Amazon Web Services



Microsoft Azure



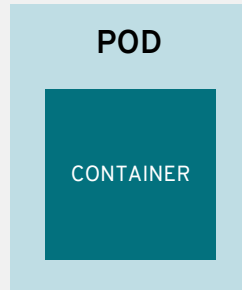
Google Cloud

A container is the smallest compute unit

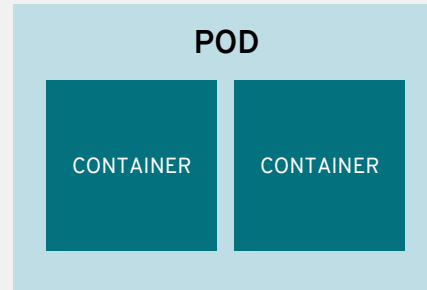


CONTAINER

containers are wrapped in pods which are units of deployment and management

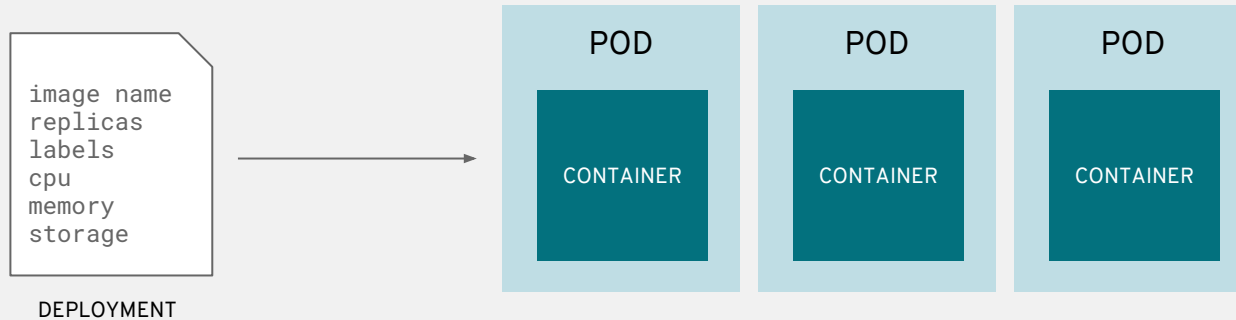


IP: 10.1.0.11



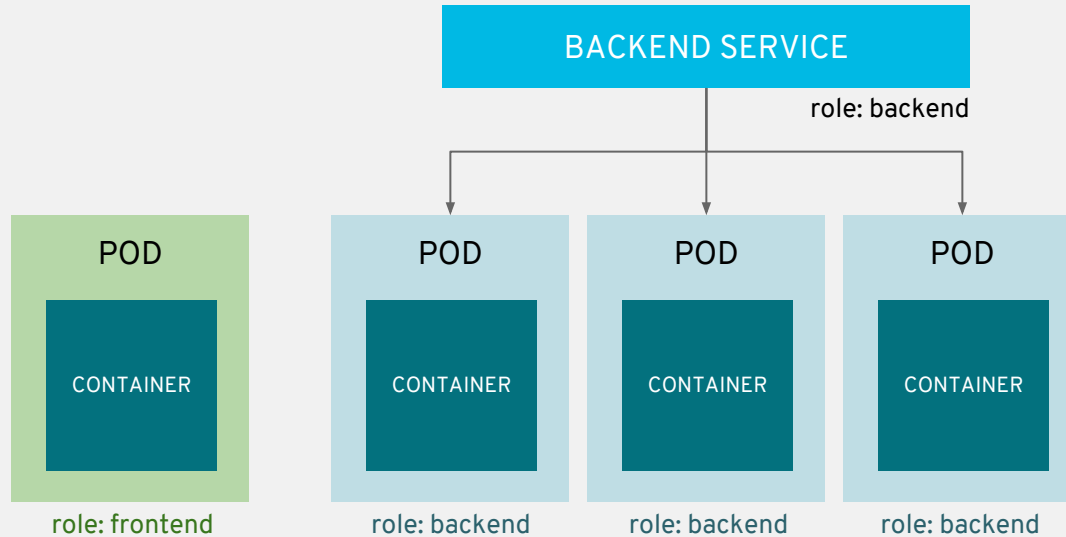
IP: 10.1.0.55

# Pods configuration is defined in a deployment

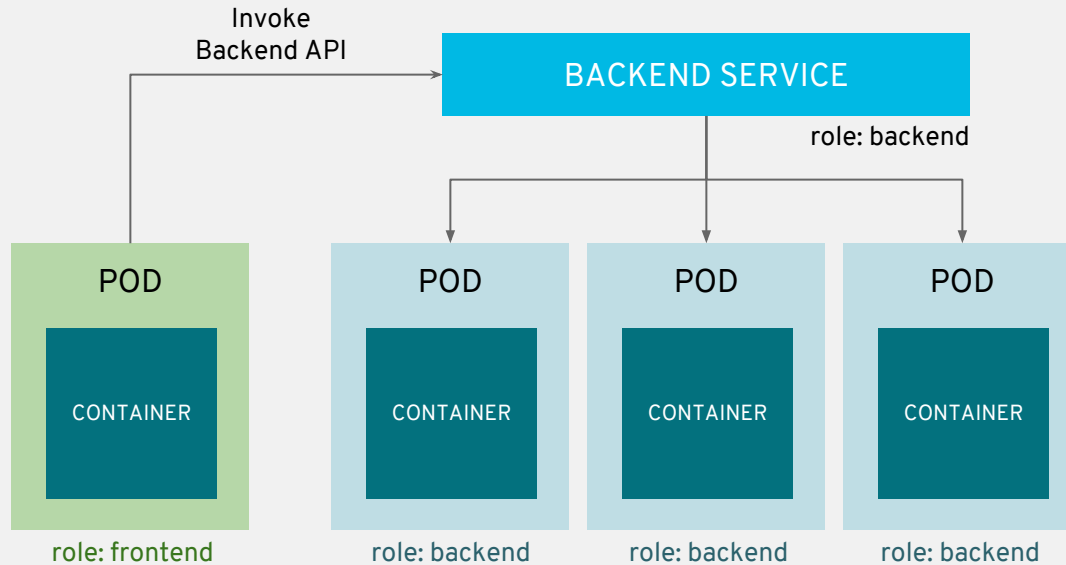




services provide internal load-balancing and service discovery across pods

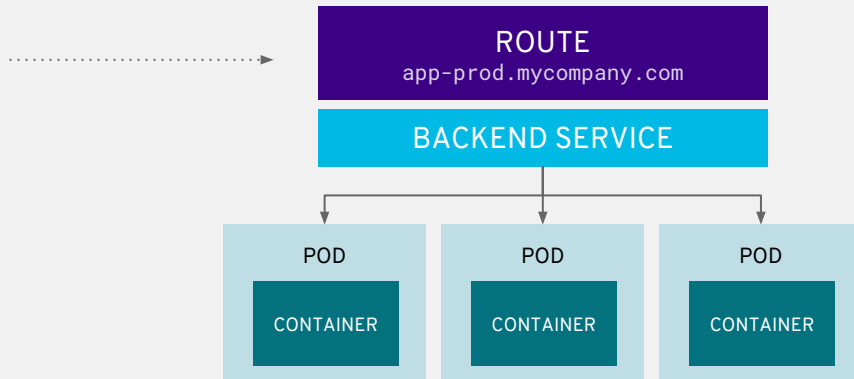


# apps can talk to each other via services

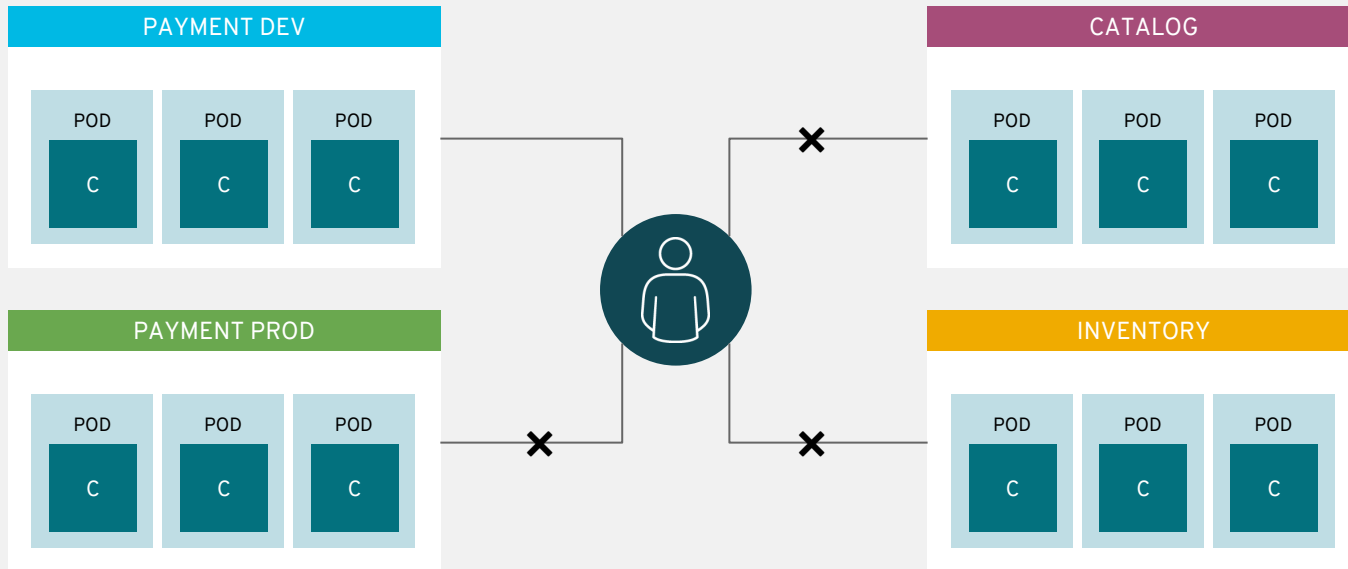


routes add services to the external load-balancer and provide readable urls for the app

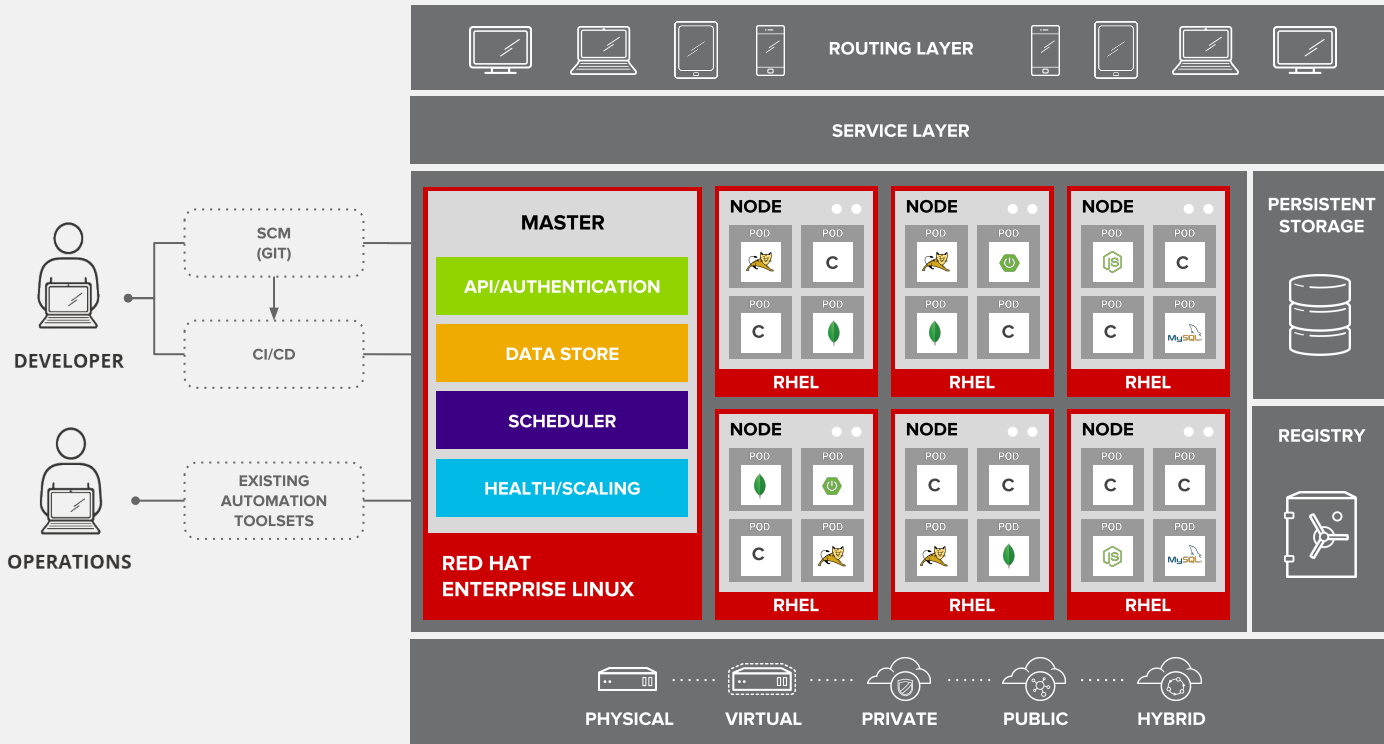
```
> curl http://app-prod.mycompany.com
```



projects isolate apps across environments,  
teams, groups and departments

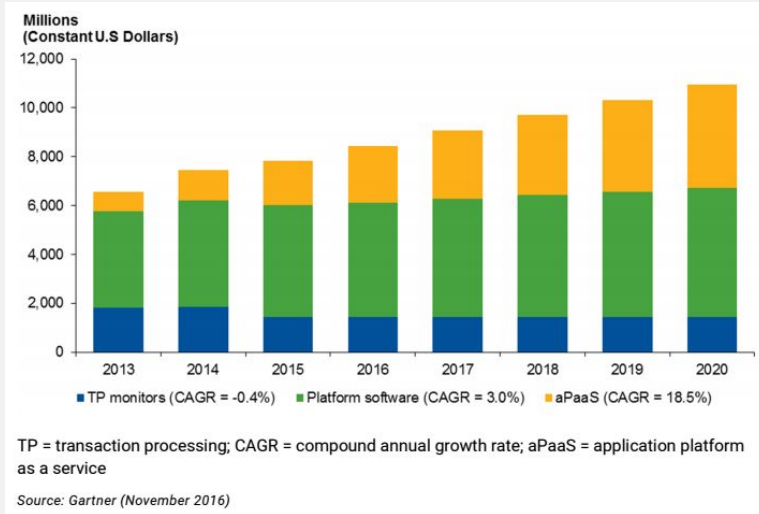


# OPENSIFT ARCHITECTURE



# Microservices Architecture 101

# State of the Market



## 2015 AP Revenue (Gartner, Nov. 2016) :

- Oracle -4.5%
- IBM -9.5%
- Red Hat +33.3%
- Amazon +50.6%
- Pivotal +22.7%

*“Resist the temptation to simply lift and shift Java EE applications from closed-source to open-source application servers for modest license savings. If you are contemplating porting an application, consider rearchitecting it to be cloud-native and moving it to aPaaS - presuming that business drivers warrant the investment.”*

Gartner (November 2016)

ThoughtWorks®

TECHNOLOGY  
RADAR

**HOLD**

45.Application Servers **new**

46.OSGi

47.SPDY **new**

*“Most teams we work with favor bundling an embedded http server within your web application. There are plenty of options available: Jetty, SimpleWeb, Webbit and Owin Self-Host amongst others. Easier automation, easier deployment and a reduction in the amount of infrastructure you have to manage lead us to **recommend embedded servers over application servers for future projects**”*

ThoughtWorks Technology Radar, May 2015

# Microservices defined

“... is an approach to developing a single application as a **suite of small services**, each running in its **own process** and communicating with **lightweight mechanisms**, often an HTTP resource API. These services are built around **business capabilities** and **independently deployable** by **fully automated deployment** machinery. There is a **bare minimum of centralized management** of these services, which may be written in **different programming languages** and use **different data storage technologies**.”

*Martin Fowler*

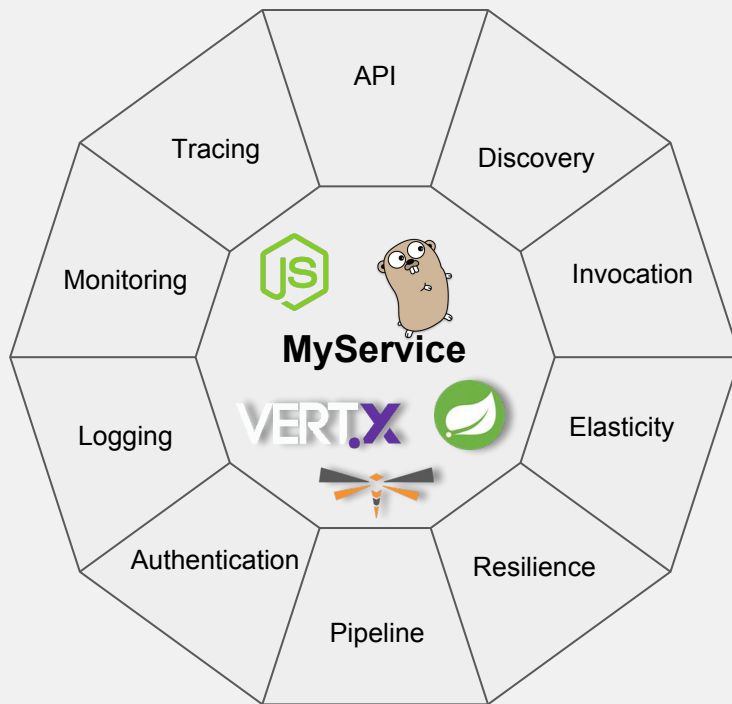
*<http://martinfowler.com/articles/microservices.html>*



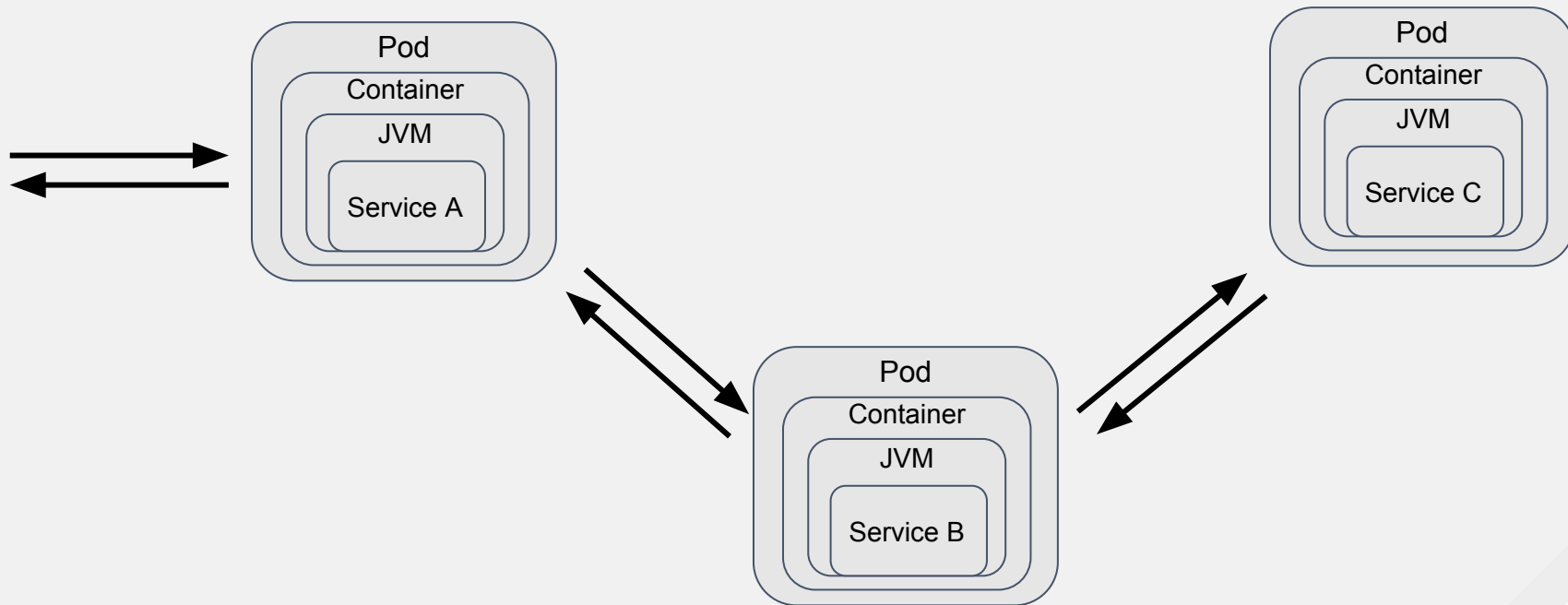
# Microservices 101

- Small single-purpose services driven from **DDD (Domain Driven Design)** or practical decomposition of an existing application or existing SOA-style **mini-services**
- **Combined** to form a system or application
- **Independently deployable** (replaceable)
- **Independently scalable**
- **Antifragile** - increased robustness and resilience under pressure
- **Fully automated** software delivery
- **Polyglot** (language and framework independence)
- **API / Contract Focused**
- Typically **event-driven**
- **Decentralized** data management

# Microservices 101



# Microservices == Distributed Computing



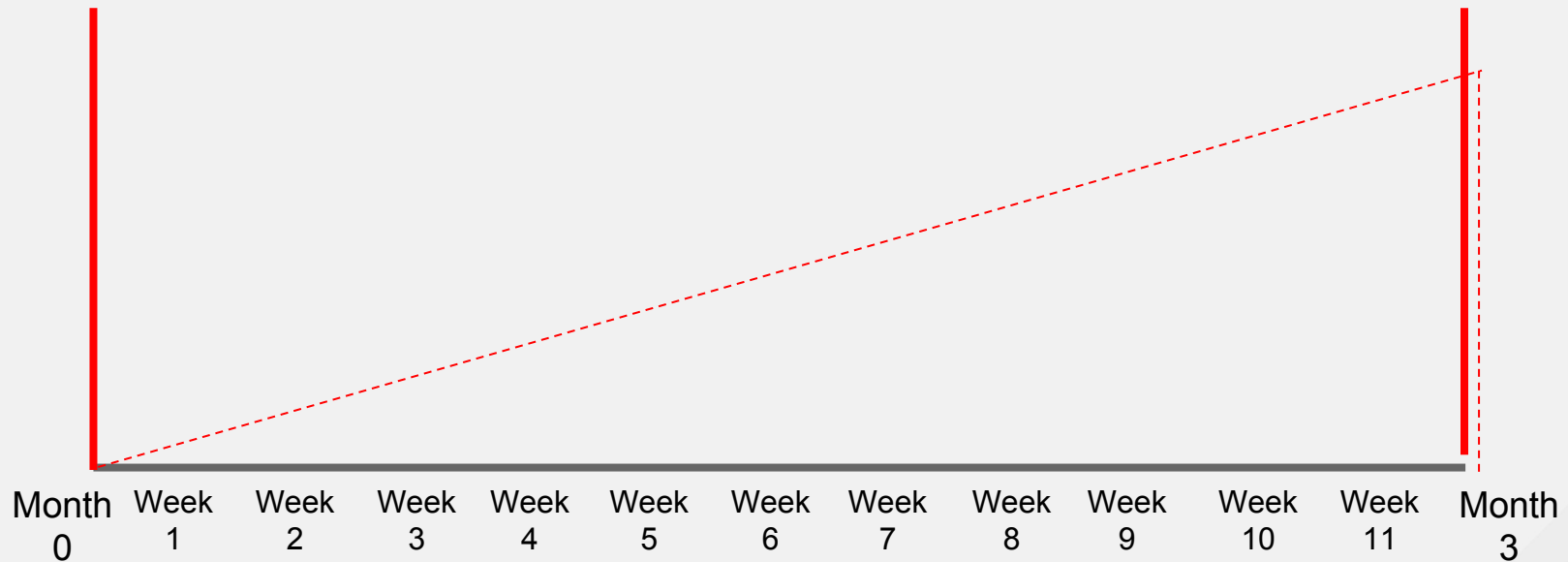
# Wait, but weren't we already doing this distributed stuff...

- ... what about **CORBA**?
- ... and **RMI**?
- ... **EJB**?
- ... **SOA**?

What's the difference?

# Maturing the Application LifeCycle

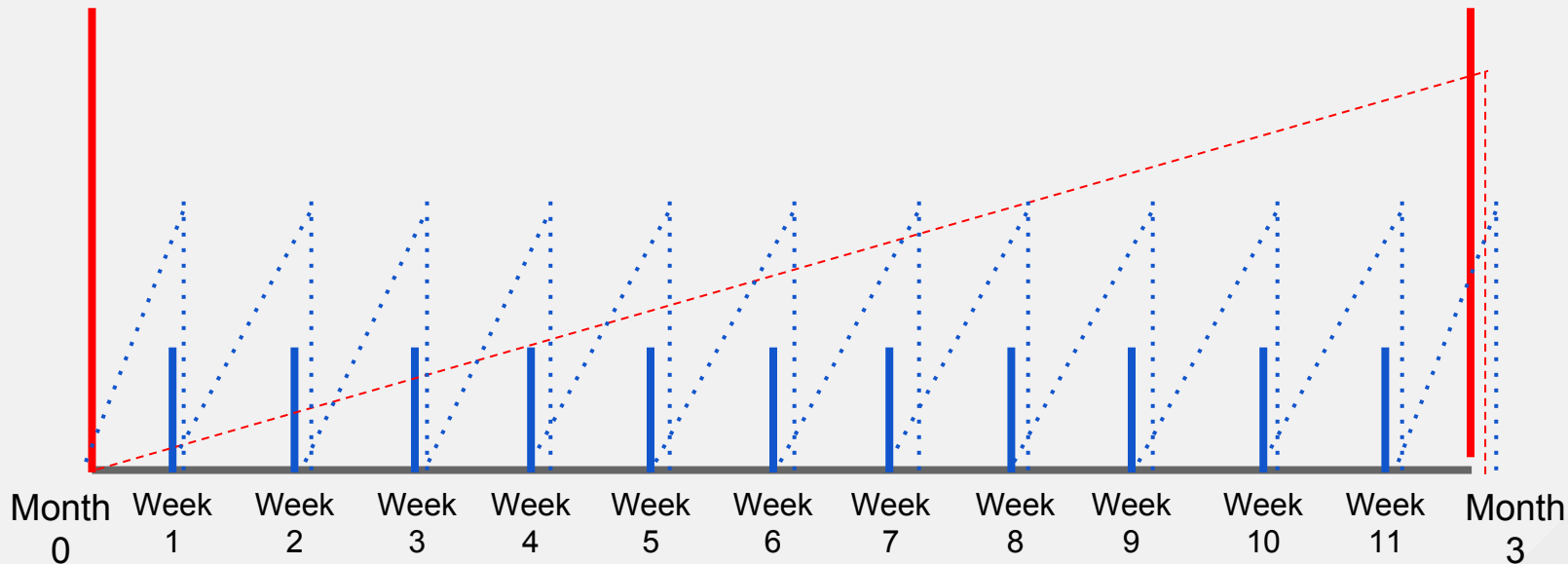
## Monolith Java EE Lifecycle



# Maturing the Application LifeCycle

Monolith Java EE Lifecycle

Fast Moving Java EE Monolith

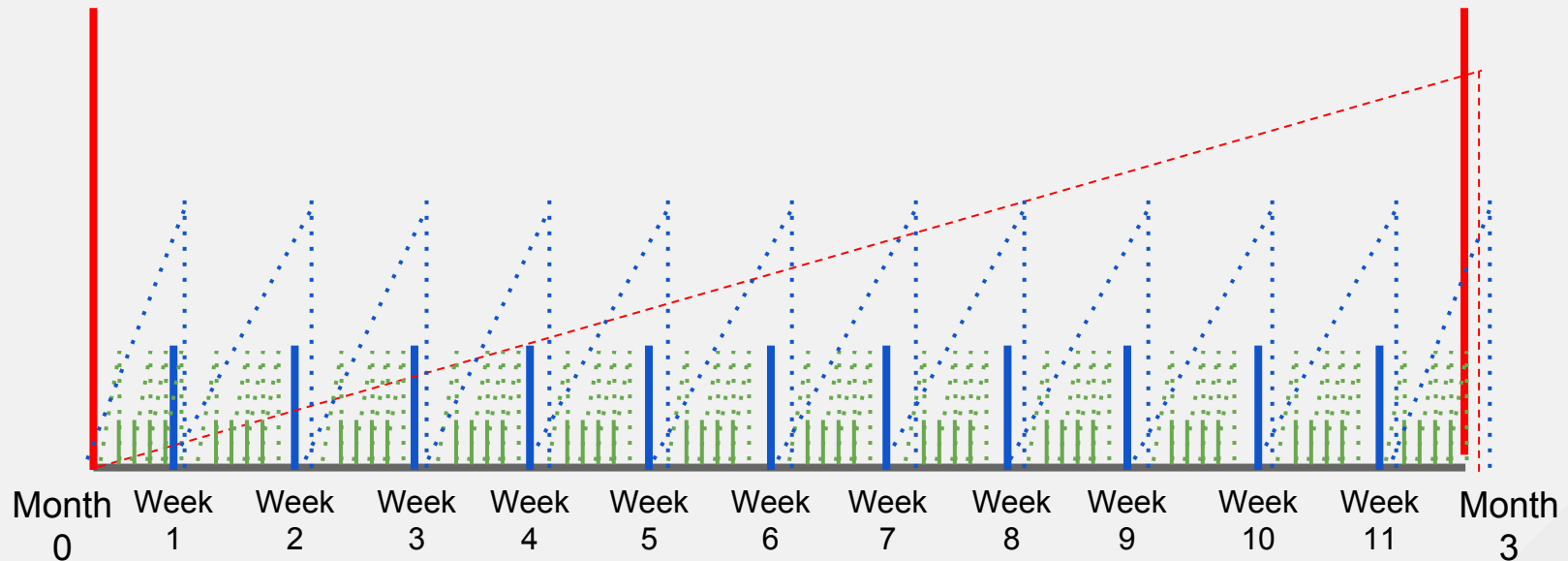


# Maturing the Application LifeCycle

Monolith Java EE Lifecycle

Fast Moving Java EE Monolith

Java EE Microservices



# What's the difference?

- Same ideas, new technologies (which will evolve in the future)
- But now, we should be able to bring a new feature in **production** in a few minutes



# Microservices: the Good, the Bad...

## The Good

- Domain-Driven Design
- Low coupling, high cohesion
- APIs and contracts
- Agile software development
- Full lifecycle automation
- Dev and Ops working together
- Common packaging / container format
- Rethinking Data



# Microservices: the Good, the Bad...

## The Bad

- Too much Dogma / CS purity
- Tradeoff between Agility & Operational Complexity
- Magnificent Monoliths and Stupendous SOA are not necessarily bad
- Microservices / Unicorn Envy
- Not all organizations can afford the **skills** and talent required to be successful
- Maintaining **data consistency** is hard in distributed systems



# Microservices: the Good, the Bad...

## The Ugly

- Building large scale distributed systems is **really** hard
- **Monitoring** / **APM** tools need to catch up
- Heterogeneity (languages, frameworks, data stores)
- Event-based, asynchronous, reactive programming is still in it's infancy and skills are **rare**
- **CAP: Consistency, Availability, Partition Tolerance**  
? – choose two



# Microservices Recommendations

- Understand and state your goals
- Understand the **tradeoffs**
- Start with People, Process and Culture
  - Agile Dev / DevOps is a prerequisite
- Invest in **automation** (provisioning, CI/CD, etc.)
- Start **small**
  - Small non-mission-critical green-field
  - Decomposition of existing monolith
- Get help - eg. **Red Hat Innovation Labs**

# Java Microservices Platform (2014)



NETFLIX Ribbon



CLOUD **FOUNDRY**

# Why these components?



**Eureka** is the Service Registry where the clients lookup for service locations a.k.a Service Discovery



**Config Server** externalized the Configuration



**Ribbon** is the client side Load Balancer



**Hystrix** is the Circuit Breaker



**Zipkin** is the Distributed Tracer



**Zuul** is the smart proxy purely based on Java

# Why these components?



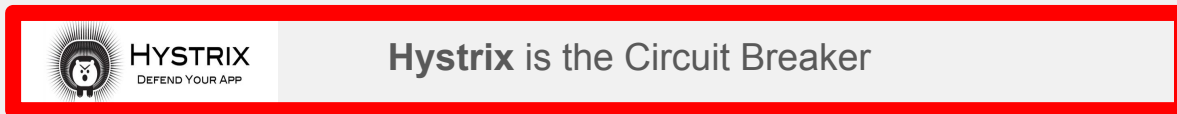
**Eureka** is the Service Registry where the clients lookup for service locations a.k.a Service Discovery



**Config Server** externalized the Configuration



**Ribbon** is the client side Load Balancer



**Hystrix** is the Circuit Breaker



**Zipkin** is the Distributed Tracer

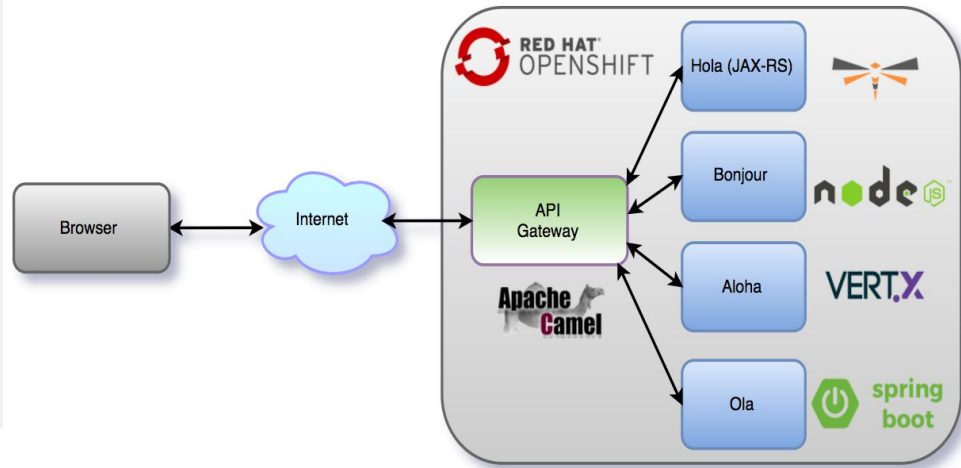
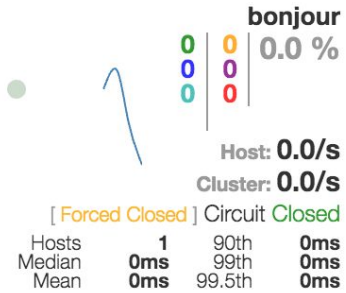


**Zuul** is the smart proxy purely based on Java

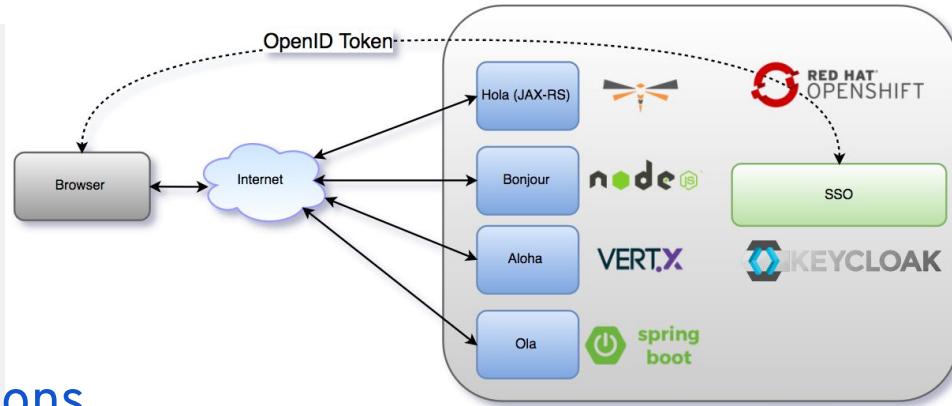




Success | Short-Circuited | Bad Request | Timeout | Rejected | Failure | Error %



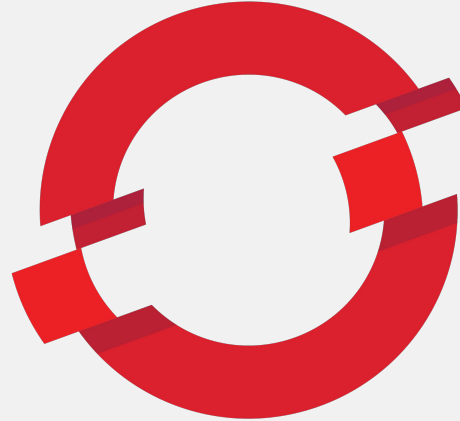
2.0



[bit.ly/msa-instructions](http://bit.ly/msa-instructions)

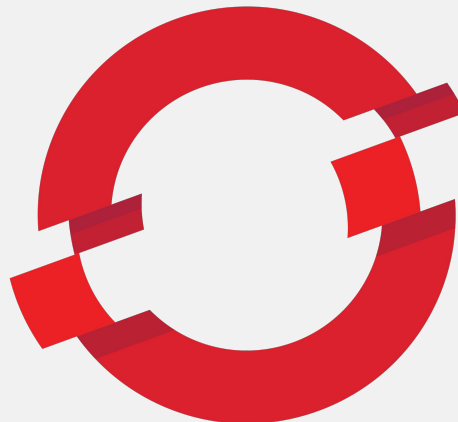


# Better Microservices Platform (2016/2017)



# OPENSHIFT

# Even Better Microservices Platform (2018)



# OPENSIFT

# Istio

# Istio



## Istio - Sail

(Kubernetes - Helmsman or ship's pilot)



# Istio

## 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.

## Resilience Across Languages and Platforms

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

## Telemetry and Reporting

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

## Policy Enforcement

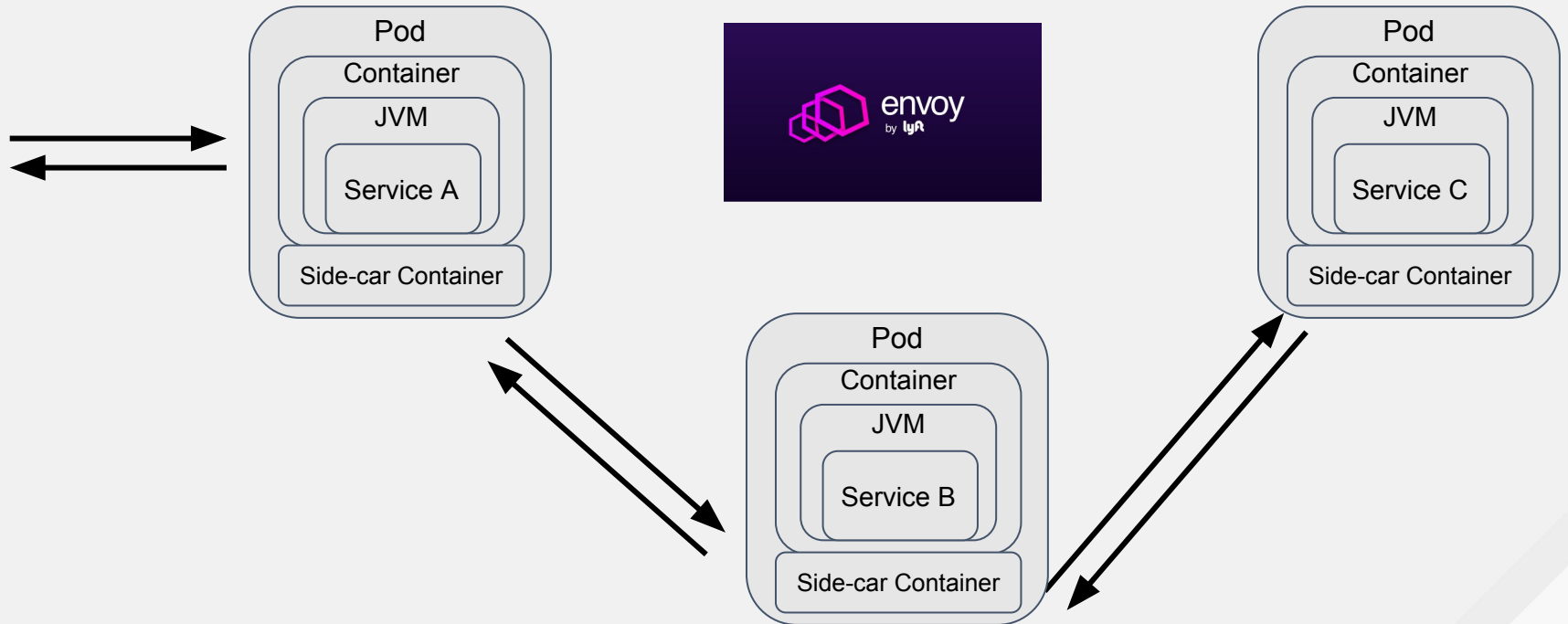
Apply organizational policy to the interaction between services, ensure access policies are enforced and resources are fairly distributed among consumers.



# Sidecar?

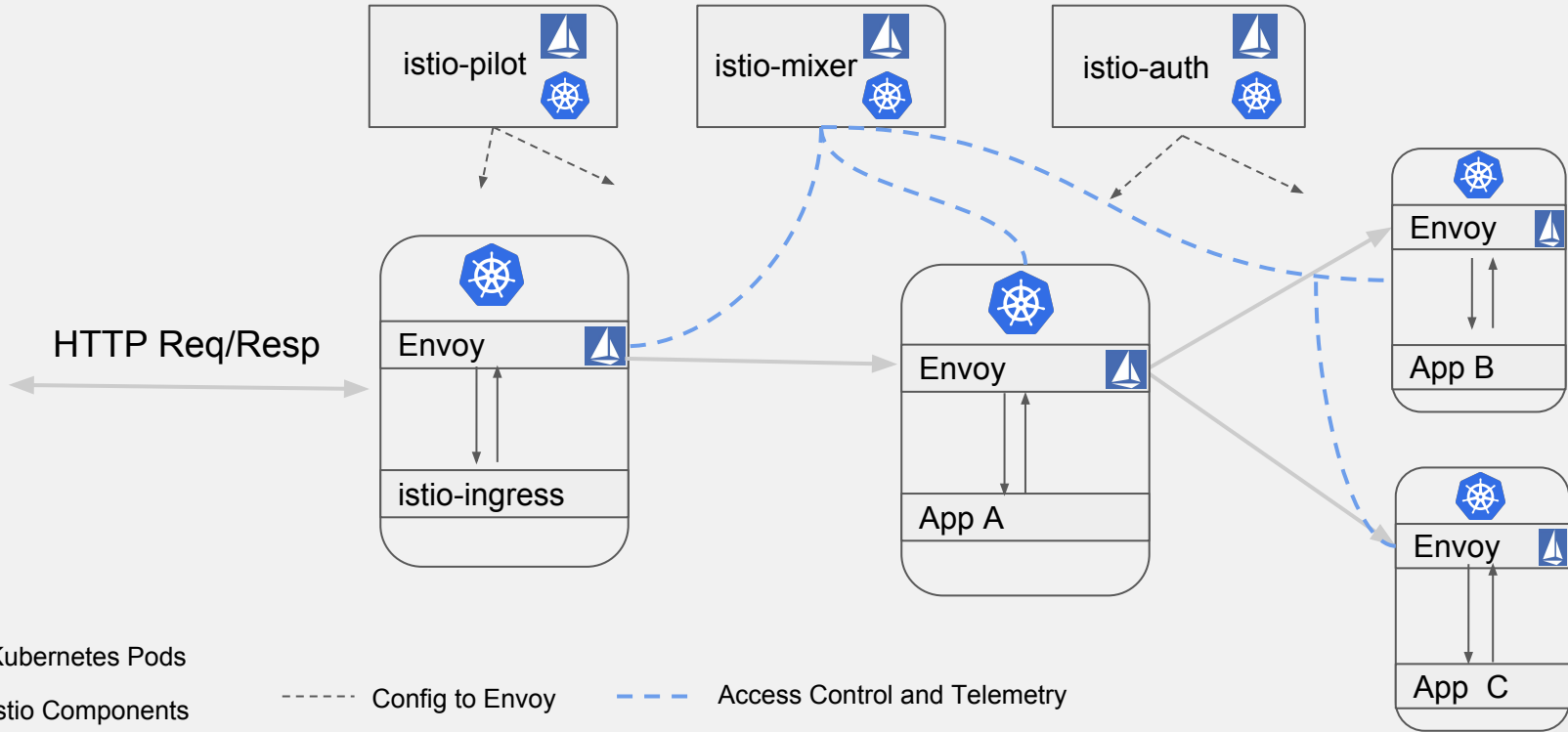


# Pods with 2 containers!



# Istio Service Mesh

*Currently upstream only*





# Istio Components

- Control Plane
  - Istio-Pilot - istioctl, API, config
  - Istio-Mixer - Quota, Telemetry, Rate Limiting, ACL
  - Istio-Auth - TLS and Certificates
- Data Plane
  - Envoy proxy deployed as “side-cars” with applications

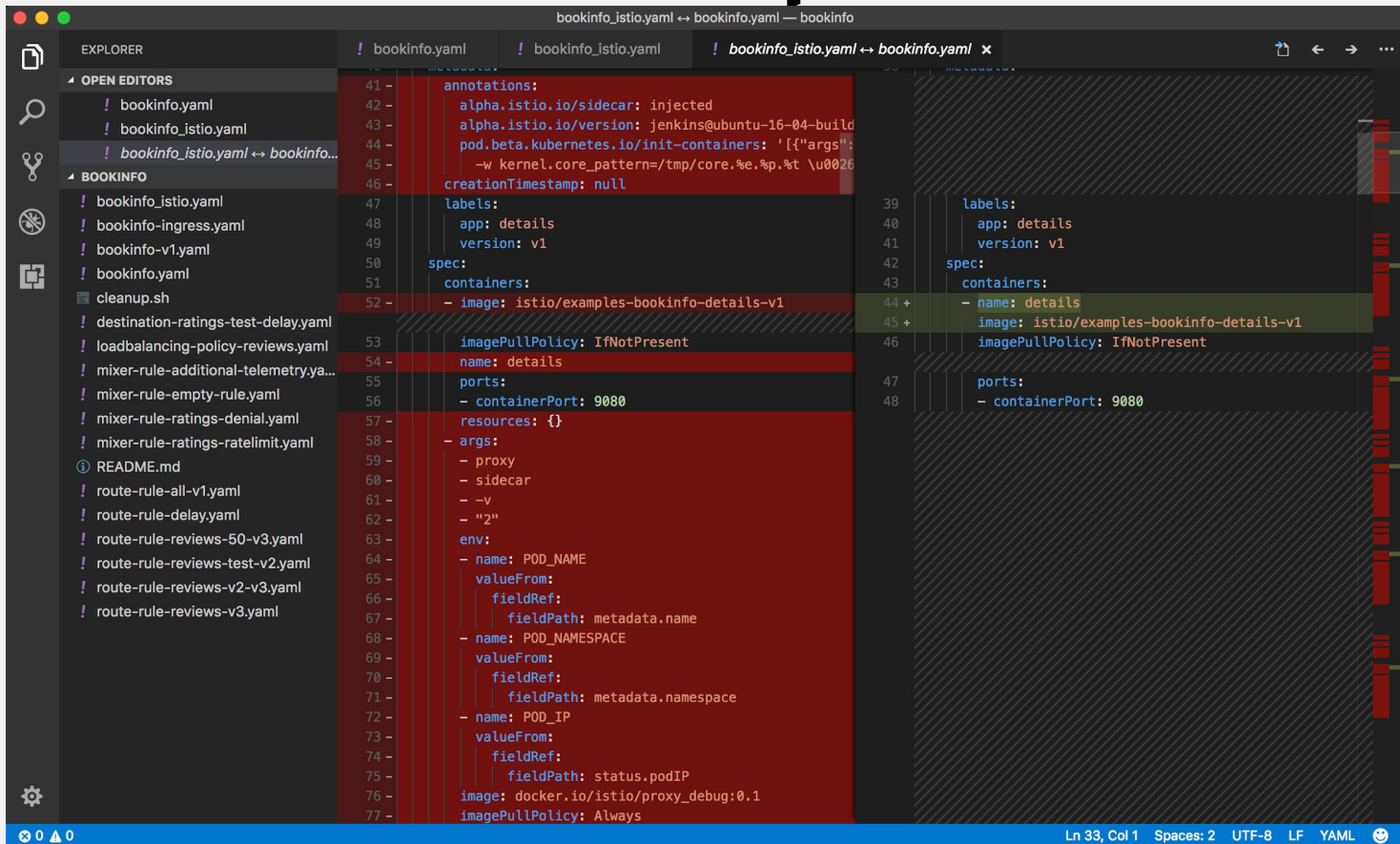
# Circuit Breakers

Before Istio	After Istio
Boiler plate code	No code related to circuit breaking mixed with business logic
Multiple libraries and dependencies e.g. Hystrix	No libraries
Separate dashboard to collect circuit breaker e.g. Hystrix Turbine	All metrics can be collected and displayed in Grafana without extra bit of code
	Define circuit breakers using Kubernetes Tags

# Tracing

Before Istio	After Istio
Boiler plate code	No code related to tracing mixed with business logic
Multiple libraries and dependencies e.g. Zipkin	No libraries

# All in one place



The screenshot shows a code editor with a diff view. The left pane shows the original file, and the right pane shows the modified file. The diff highlights changes in the container configuration, specifically the image name and the container port.

```
bookinfo_istio.yaml ↔ bookinfo.yaml — bookinfo
! bookinfo.yaml
! bookinfo_istio.yaml
! bookinfo_istio.yaml ↔ bookinfo...
! bookinfo_istio.yaml
! bookinfo-ingress.yaml
! bookinfo-v1.yaml
! bookinfo.yaml
cleanup.sh
! destination-ratings-test-delay.yaml
! loadbalancing-policy-reviews.yaml
! mixer-rule-additional-telemetry.ya...
! mixer-rule-empty-rule.yaml
! mixer-rule-ratings-denial.yaml
! mixer-rule-ratings-ratelimit.yaml
① README.md
! route-rule-all-v1.yaml
! route-rule-delay.yaml
! route-rule-reviews-50-v3.yaml
! route-rule-reviews-test-v2.yaml
! route-rule-reviews-v2-v3.yaml
! route-rule-reviews-v3.yaml
41 - annotations:
42 -   alpha.istio.io/sidecar: injected
43 -   alpha.istio.io/version: jenkins@ubuntu-16-04-build
44 -   pod.beta.kubernetes.io/init-containers: '[{"args":
45 -     -w kernel.core_pattern=/tmp/core.%e.%p.%t \u0026
46 -     creationTimestamp: null
47 -     labels:
48 -       app: details
49 -       version: v1
50 -     spec:
51 -       containers:
52 -         - image: istio/examples-bookinfo-details-v1
44 +
45 +     labels:
46 +       app: details
47 +       version: v1
48 +     spec:
49 +       containers:
50 +         - name: details
51 +           image: istio/examples-bookinfo-details-v1
52 +           imagePullPolicy: IfNotPresent
53 +           name: details
54 +           ports:
55 +             - containerPort: 9080
56 +           resources: {}
57 +           - args:
58 +             - proxy
59 +             - sidecar
60 +             - -v
61 +             - "2"
62 +           env:
63 +             - name: POD_NAME
64 +               valueFrom:
65 +                 fieldRef:
66 +                   fieldPath: metadata.name
67 +             - name: POD_NAMESPACE
68 +               valueFrom:
69 +                 fieldRef:
70 +                   fieldPath: metadata.namespace
71 +             - name: POD_IP
72 +               valueFrom:
73 +                 fieldRef:
74 +                   fieldPath: status.podIP
75 +             image: docker.io/istio/proxy_debug:0.1
76 +             imagePullPolicy: Always
```

Ln 33, Col 1 Spaces: 2 UTF-8 LF YAML

# How to use it

## Routes and commands injected via CLI or API:

```
apiVersion: config.istio.io/v1alpha2
kind: RouteRule
metadata:
  name: reviews-test-v2
spec:
  destination:
    name: reviews
  precedence: 2
  match:
    request:
      headers:
        cookie:
          regex: "^(.*?;)?(user=jason)(;.*)?$"
  route:
  - labels:
    version: v2
```

# Operator Framework

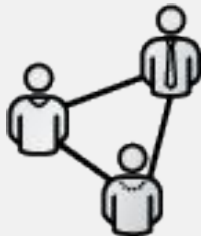
# Why Operators?



\$ oc new-app myapp

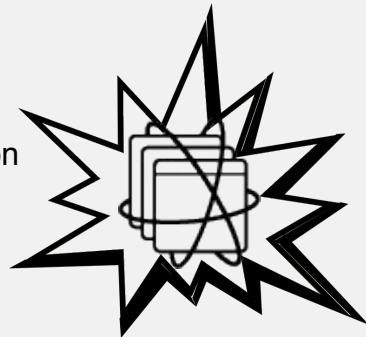


30 days later....



Platform  
Services Team

Tries to keep the application  
framework or runtime from  
exploding



Developer

Wouldn't be great if....



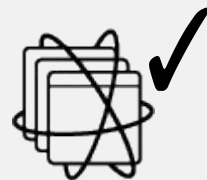
\$ oc create -f myAppsTask.yaml

Operator SDK



The Platform

- re-index
- backup
- restore
- defrag
- recycle
- ...any admin task



Developer

# Operator Lifecycle Management



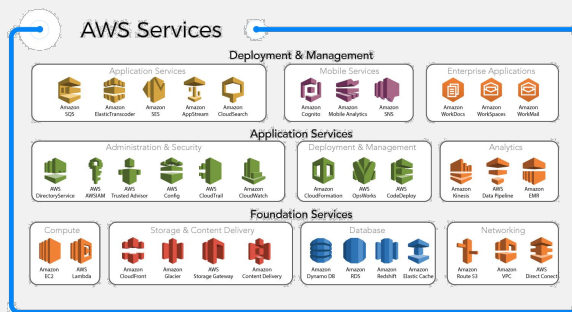


# The Result

Your App



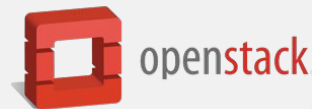
Is as **automated** as these apps



but runs....



Google Cloud Platform





# THANK YOU



[plus.google.com/+RedHat](https://plus.google.com/+RedHat)



[facebook.com/redhatinc](https://facebook.com/redhatinc)



[linkedin.com/company/red-hat](https://linkedin.com/company/red-hat)



[twitter.com/RedHatNews](https://twitter.com/RedHatNews)



[youtube.com/user/RedHatVideos](https://youtube.com/user/RedHatVideos)