Cloud Native with OpenShift on IBM Z

—

Matt Mondics
matt.mondics@ibm.com

Technical Specialist
Hybrid Cloud and AIOps with IBM Z and LinuxONE
IBM Z Washington Systems Center

What is “Cloud Native”?

- "Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds.
- Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.
- These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."

Source: Cloud Native Computing Foundation (CNCF)

Advantages to Cloud Native Adoption

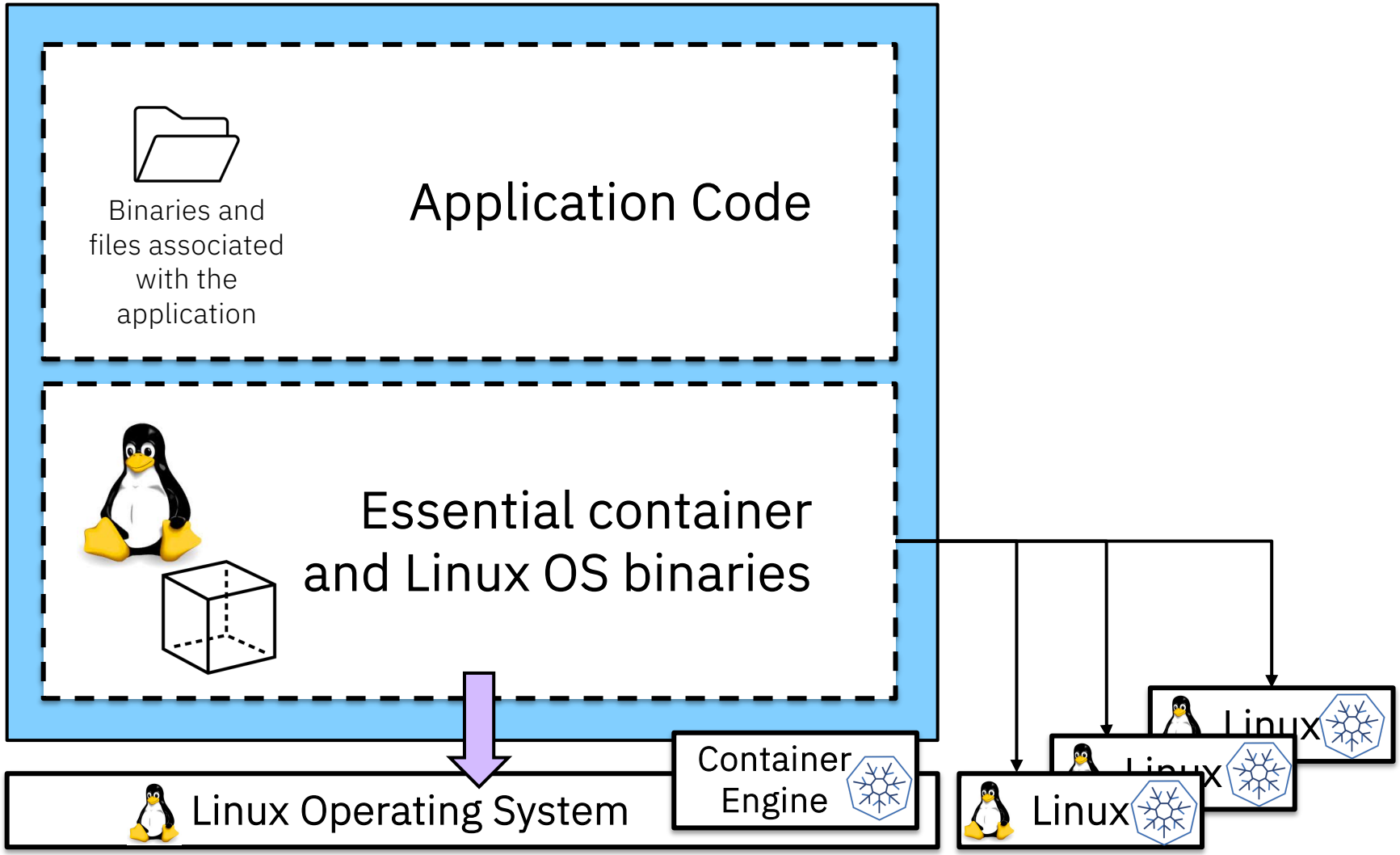
- Agility to bring applications to market quickly
- Improve Applications performance on the fly
- Avoid Changing an entire application
- Flexibility with Integration
- Speed up application development and modernization

Containers

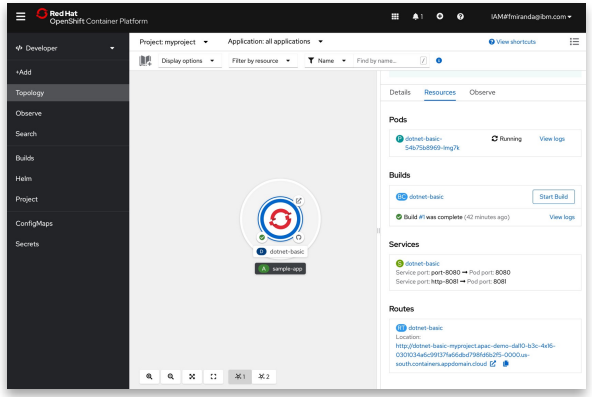
Containerization Overview

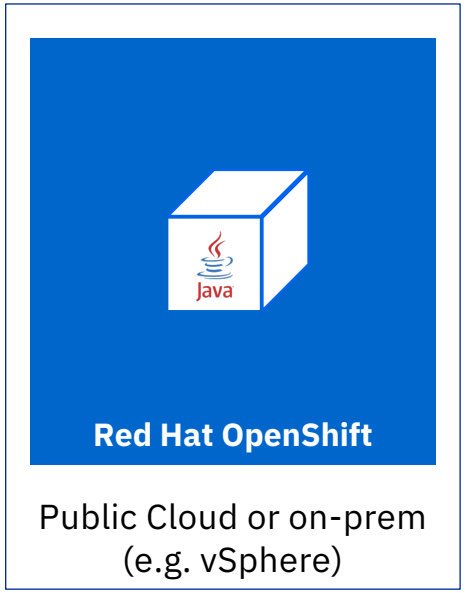
Containerization is not a mystical concept.

- Start with the application and associated files
- Add essential container and Linux files and binaries
- Build a container image with provided tooling
- Run image on systems where container runtime exists



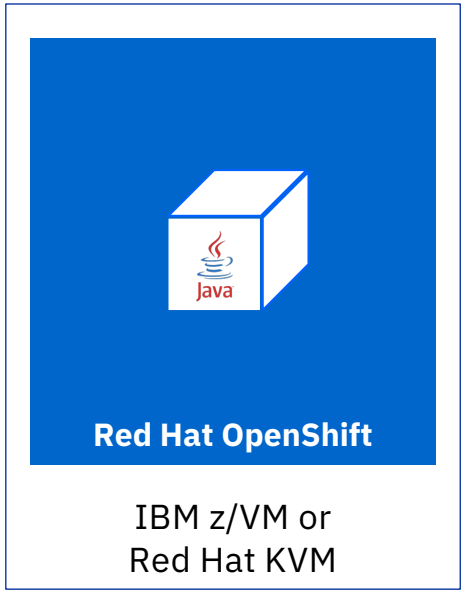
Containerization is, however, architecture-dependent





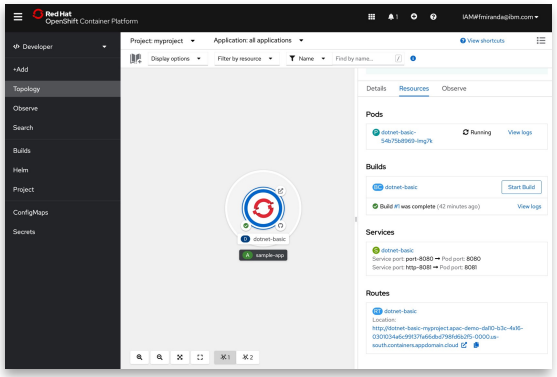
Public Cloud or on-prem
(e.g. vSphere)

x86_64

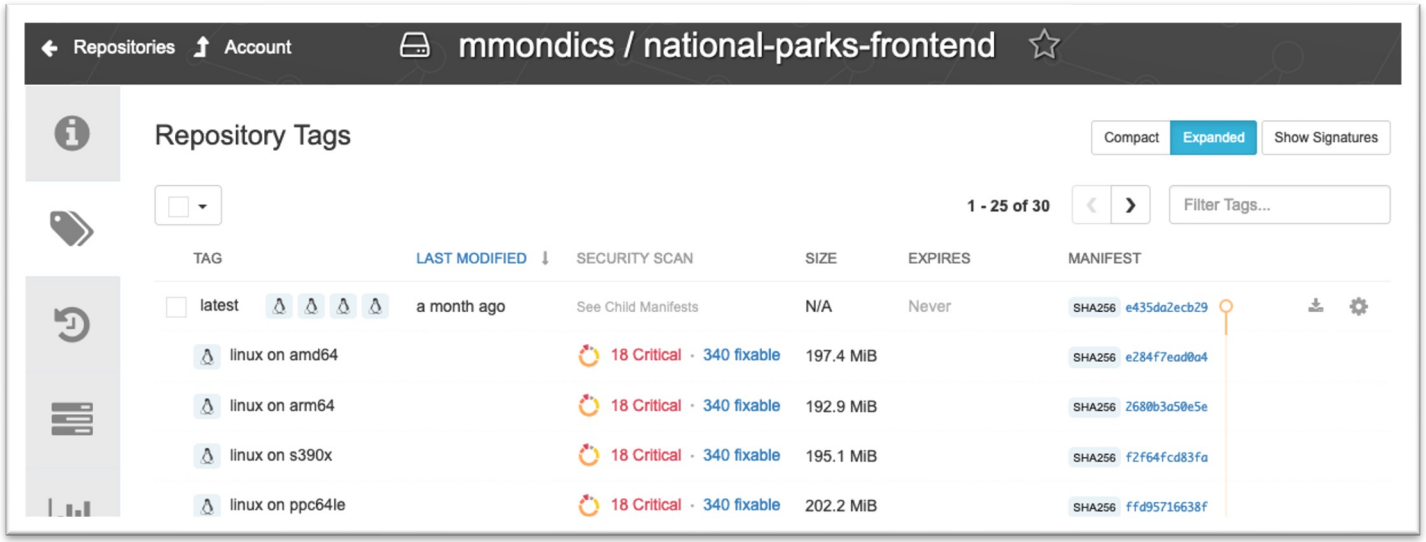


IBM z/VM or
Red Hat KVM

s390x (IBM LinuxONE)



Build multi-architecture container images!



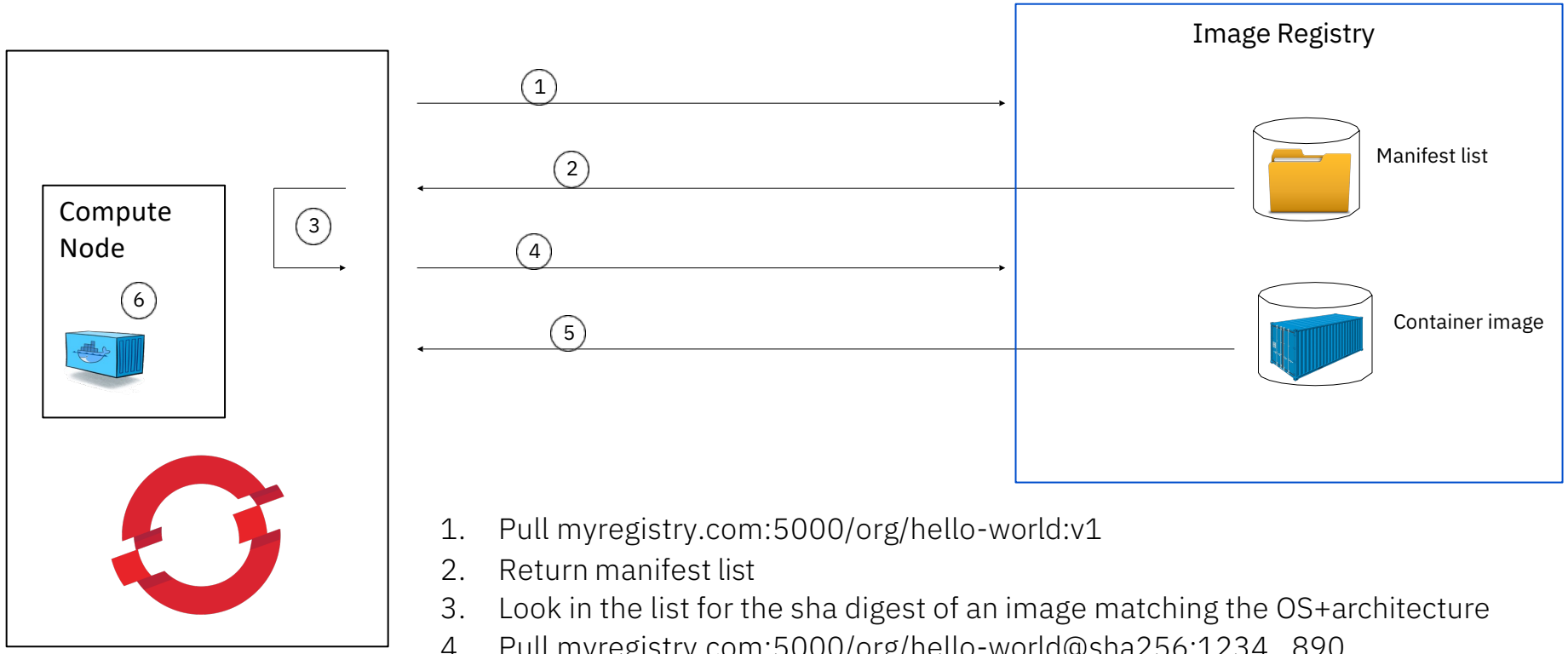
JSON metadata representation (oci)

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.oci.image.index.v1+json",
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:84485f8a085a4b93ebd4b5dfd62ecfb45a4bce57b62120ff68be8b29387a629b",
      "size": 771,
      "platform": {
        "architecture": "s390x",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:a2e28ed49b3b2bc68fdd635a1b2ccd772f47e01555532c02d131446438c87d72",
      "size": 771,
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    }
  ]
}
```


Reference: https://developer.ibm.com/tutorials/multi-architecture-cri-o-container-images-for-red-hat-openshift/

What happens when you deploy a multiarchitecture container image?

Regardless of the deployment platform (os+architecture), everyone uses the same image name when they pull an image.



1. Pull myregistry.com:5000/org/hello-world:v1
2. Return manifest list
3. Look in the list for the sha digest of an image matching the OS+architecture
4. Pull myregistry.com:5000/org/hello-world@sha256:1234...890
5. Return the container image specific to the OS+architecture
6. Container runtime starts a container instance based on the container image

Why build multiarchitecture container images?

"Build once", Deploy anywhere!

Hybrid Cloud

To take advantage of an enterprise Kubernetes layer offered by OpenShift, to develop your application code anywhere and leverage the multi-architecture DevOps to deliver new application code anywhere.

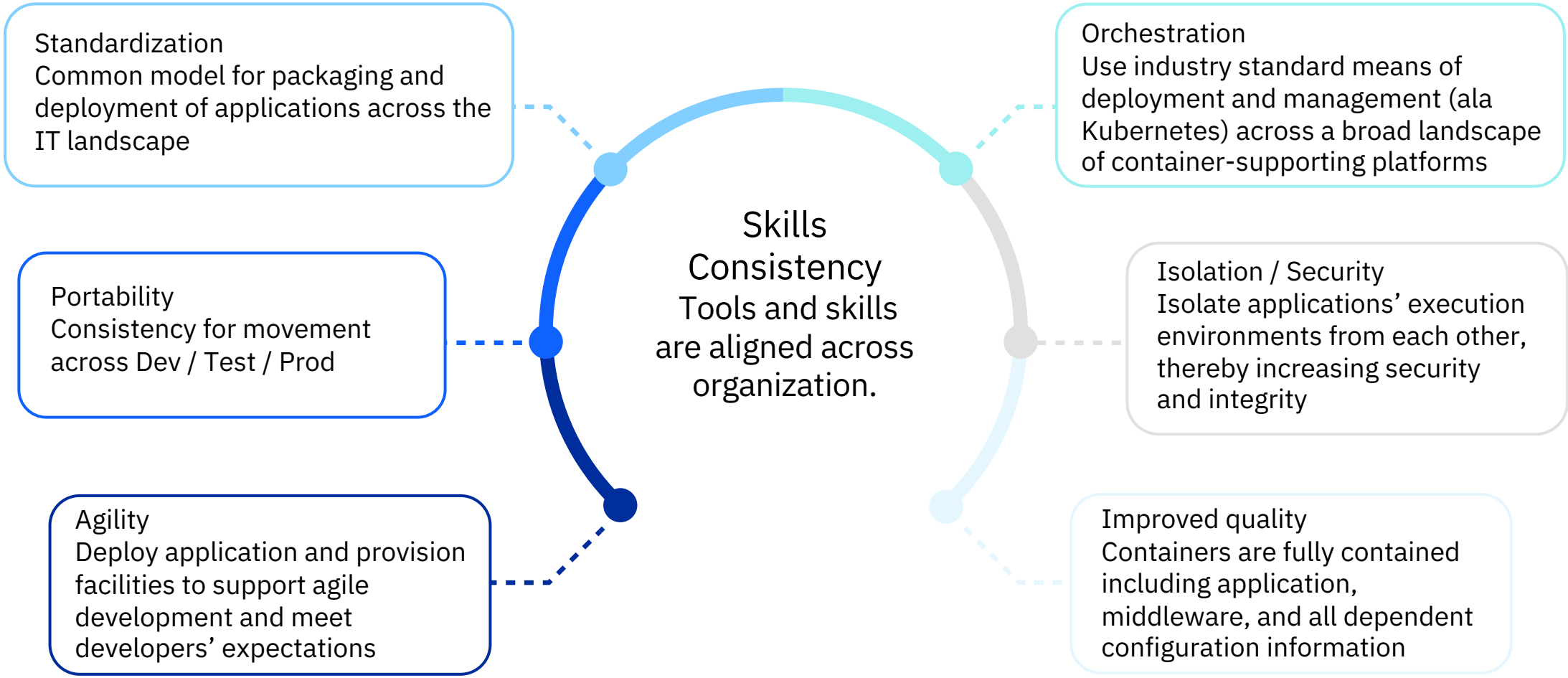
Multi-Architecture Applications

Leveraging multiple container images built for specific hardware architectures (x86, s390x, ARM64, ppcle64), all stored on a container registry and managed by a fat- manifest that will deliver the correct OS-architecture container image.

Simplify DevOps & CI/CD

Deliver maximum reduction in effort, time and speed
Using Red Hat DevSpaces (CI - Continuous Integration) and many other options of CD (continuous deployment) systems, like for example Jenkins, Tekton, or ArgoCD, combined with Red Hat OpenShift standardizes the DevOps process across hybrid Cloud.

Benefits of Containers



Kubernetes

Although container images and the containers that run from them are the primary building blocks for modern application development, to run them at scale requires a reliable and flexible distribution system. Kubernetes is the defacto standard for orchestrating containers.

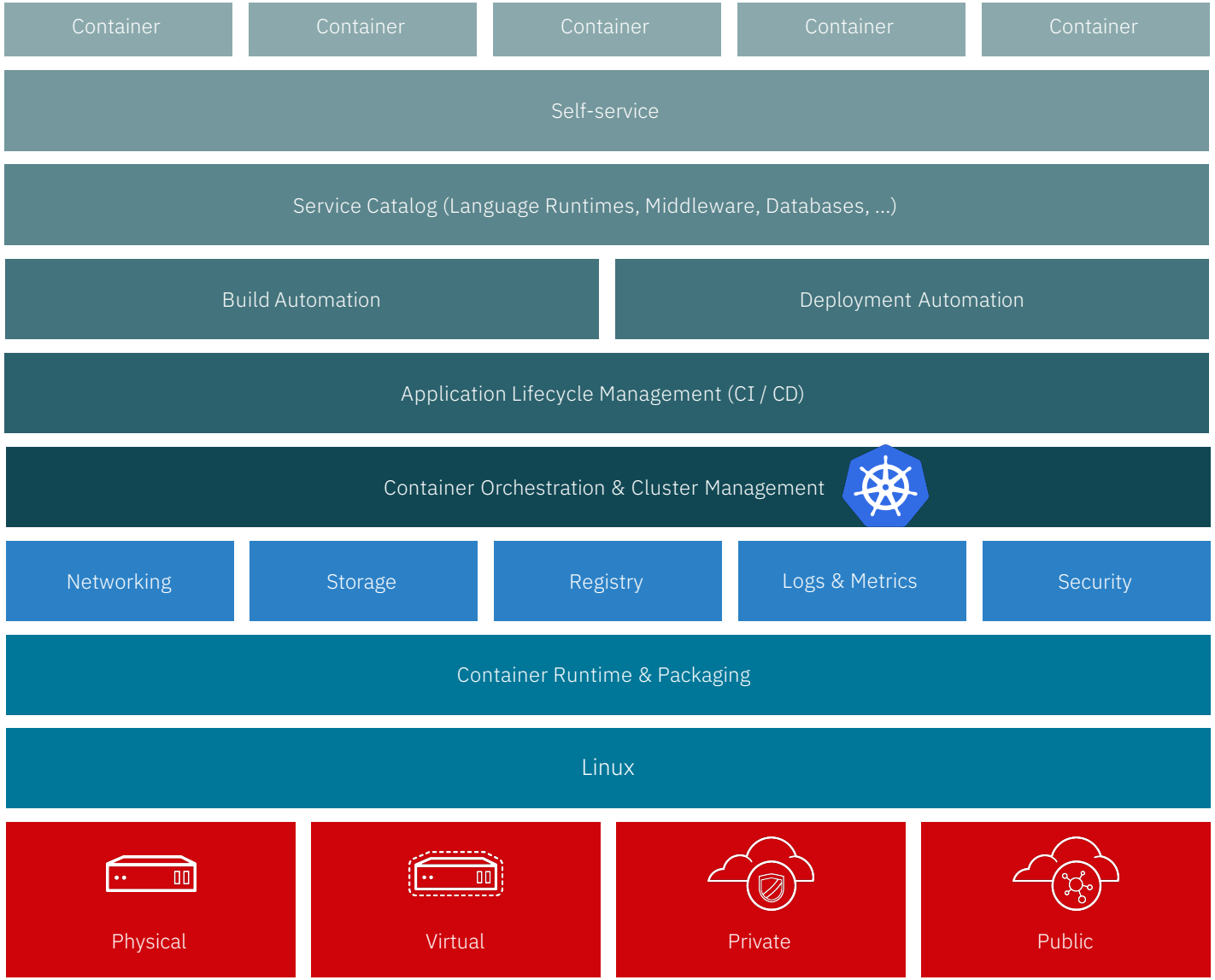
Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications.

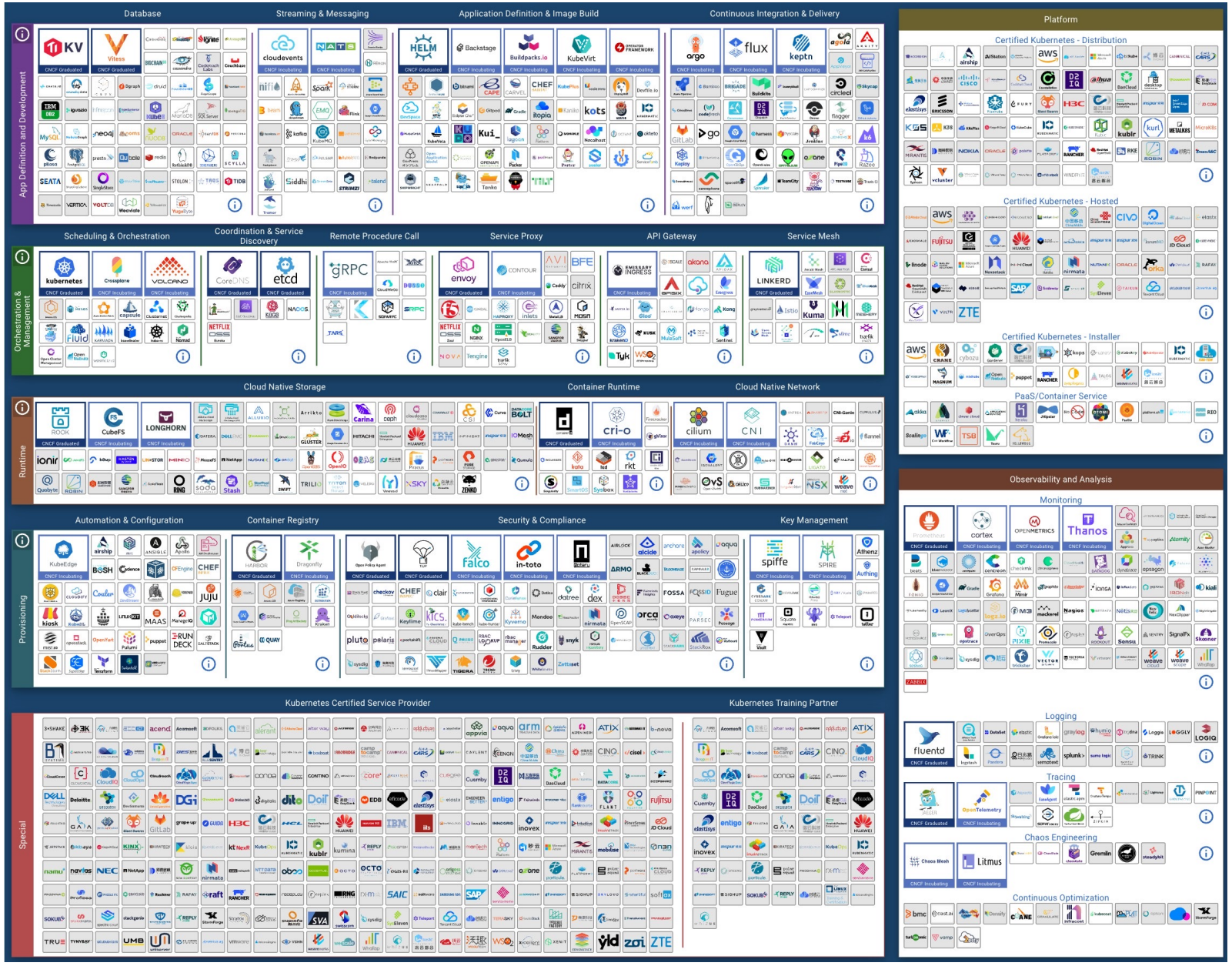
What is Kubernetes?

- A Container Orchestrator designed to automate container deployment, scaling, and management
- The basis of most cloud platforms you are probably aware of: AWS, Azure, GCP, IBM Cloud, OpenShift, and more.
- Developed by Google in 2014
- Used by Google to manage billions of containers per week running their services
- First production grade version (1.0) released July 2015
- ~Quarterly release since 1.2.0 in March 2016
- Latest 1.29 released in February 2024
- Seed technology of the Cloud Native Computing Foundation (CNCF)

CLOUD NATIVE
COMPUTING FOUNDATION

Building a Cloud Native Platform




Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Orchestration & Management
Cloud Native Storage
Container Runtime
Cloud Native Network
Runtime
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Provisioning
Kubernetes Certified Service Provider
Kubernetes Training Partner
Special
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Continuous Optimization

IBM and Red Hat – Open Source Contributors

Range: Last decade | Metric: Contributions

Kubernetes Companies statistics (Contributions, Range: Last decade), bots excluded

Rank	Company	Number
	All	3237958
1	Google LLC	1013969
2	Red Hat Inc.	392455
3	VMware Inc.	262951
4	Independent	109963
5	Microsoft Corporation	103560
6	International Business Machines Corporation	98498
7	Huawei Technologies Co. Ltd	48054
8	The Scale Factory Limited	28705
9	Intel Corporation	27623
10	CNCF	22208
11	Amazon	21806
12	NEC Corporation	21690
13	Kubermatic GmbH	21679
14	Fujitsu Limited	18758
15	SUSE LLC	17177
16		17058
17		16873
18	ZTE Corporation	16251
19	Hyper.sh	13190
20	Samsung SDS	13116

https://k8s.devstats.cncf.io/d/9/companies-table?orgId=1

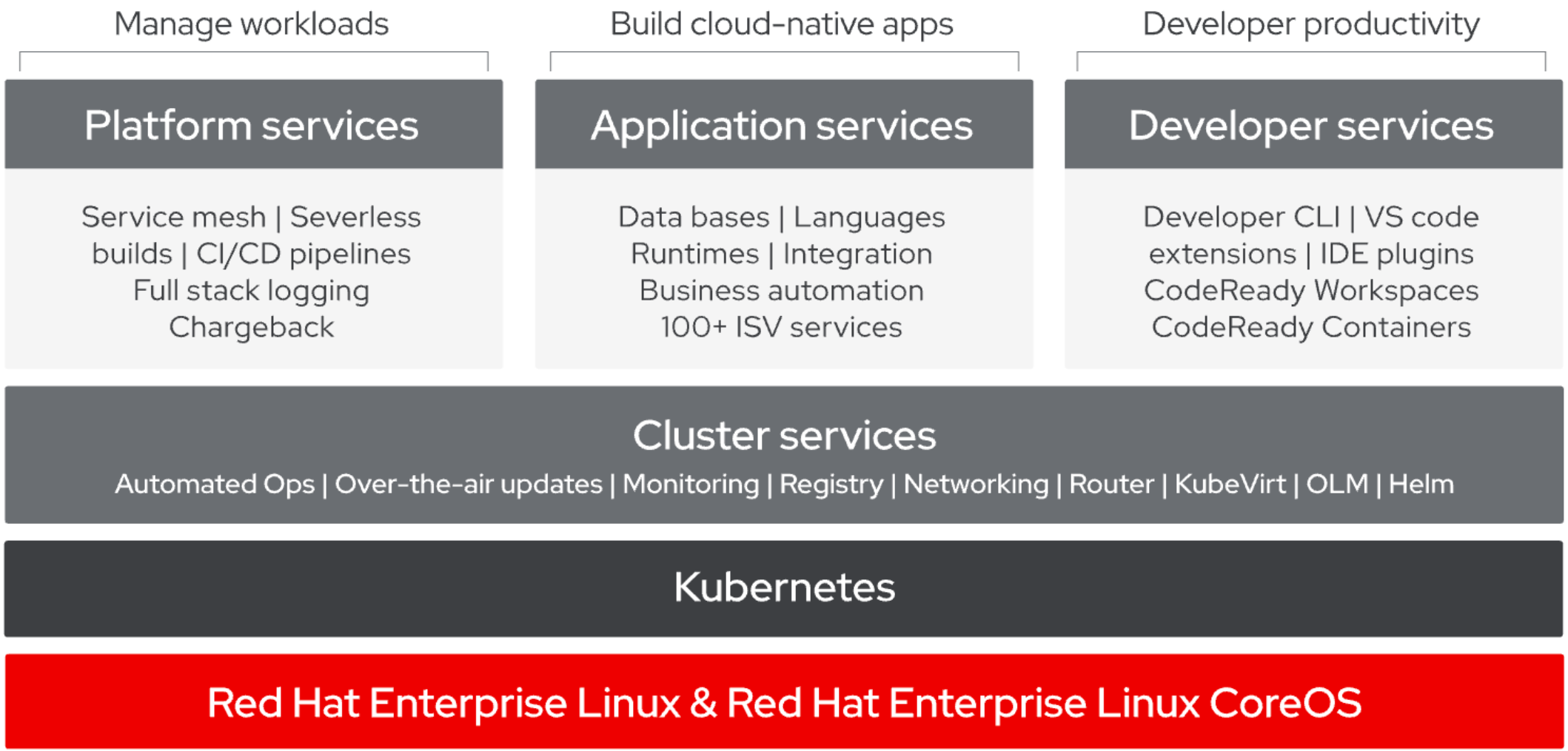
Range: Last decade | Metric: Contributions

containerd Companies statistics (Contributions, Range: Last decade), bots excluded

Rank	Company	Number
	All	87686
1	Docker Inc.	17868
2	NTT Corporation	13159
3	International Business Machines Corporation	8834
4	Google LLC	5787
5	Amazon	5626
6	Apple Inc.	4861
7	Alibaba.com	4449
8	Microsoft Corporation	2893
9	Independent	1704
10	LumApps	1475
11	Datadog	1421
12	VMware Inc.	1342
13	CNCF	1051
14	Red Hat Inc.	631
15	Tesla Inc .	540
16	LilithGames	498
17		
18	ZTE Corporation	496
19	Wargaming	460
20	SUSE LLC	451

https://containerd.devstats.cncf.io/d/5/companies-table?orgId=1

OpenShift Container Platform (OCP) Overview



IBM’s hybrid cloud and AI platform approach

IBM Consulting

Business Transformation • Technology Consulting • Application Operations

System Integrator Partners

IBM Software

IBM Cloud Paks®

Automation • Data & AI • Security • Transaction Processing

Software and SaaS Partners

Red Hat® Hybrid Cloud Platform

Development, Security and Operational Services

OpenShift® • Red Hat Enterprise Linux • Ansible® Automation Platform

IBM Infrastructure

IBM Z® / IBM LinuxONE • Distributed Infrastructure (IBM Cloud®, Power®, Storage) • Infrastructure Support

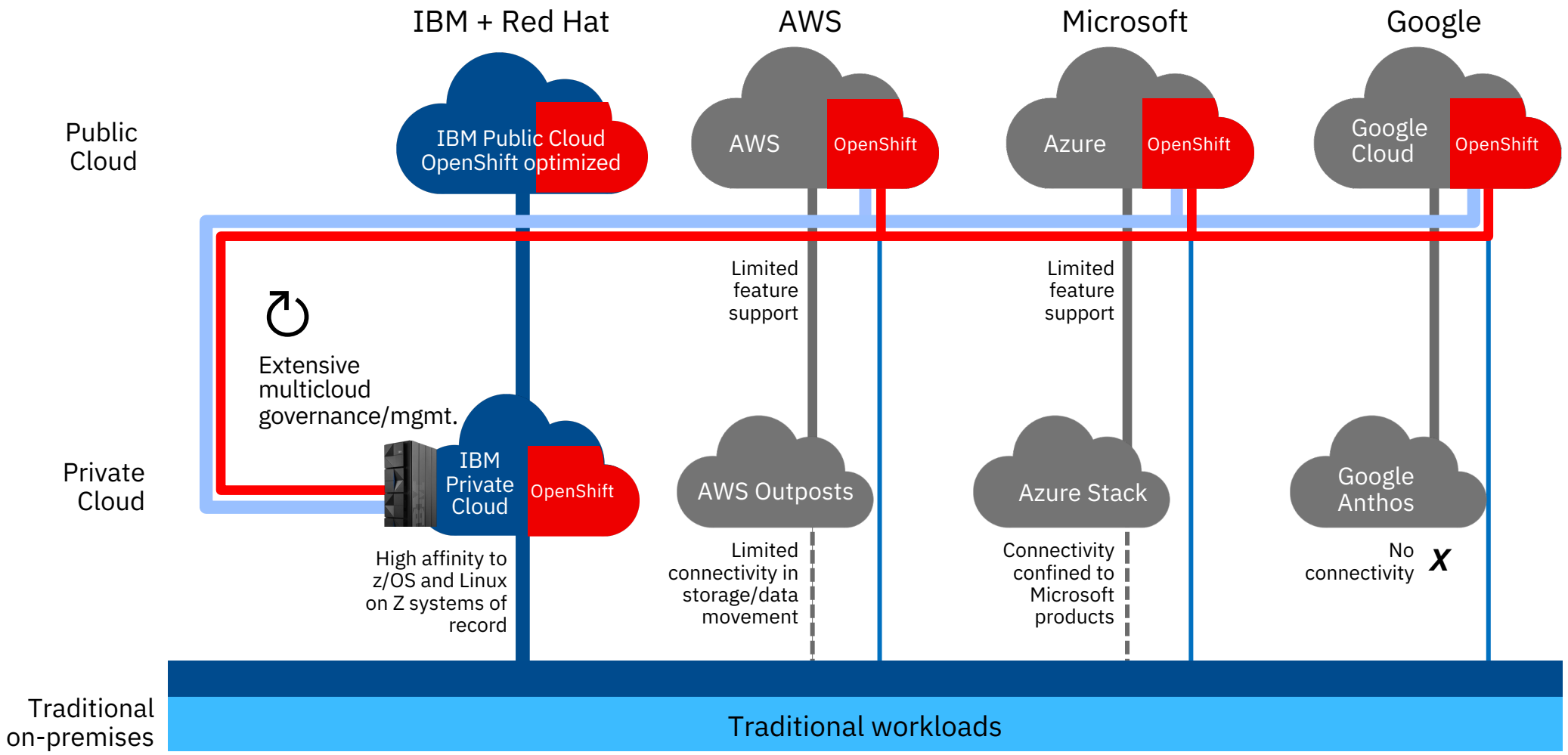
Public Clouds

AWS • Azure • Others

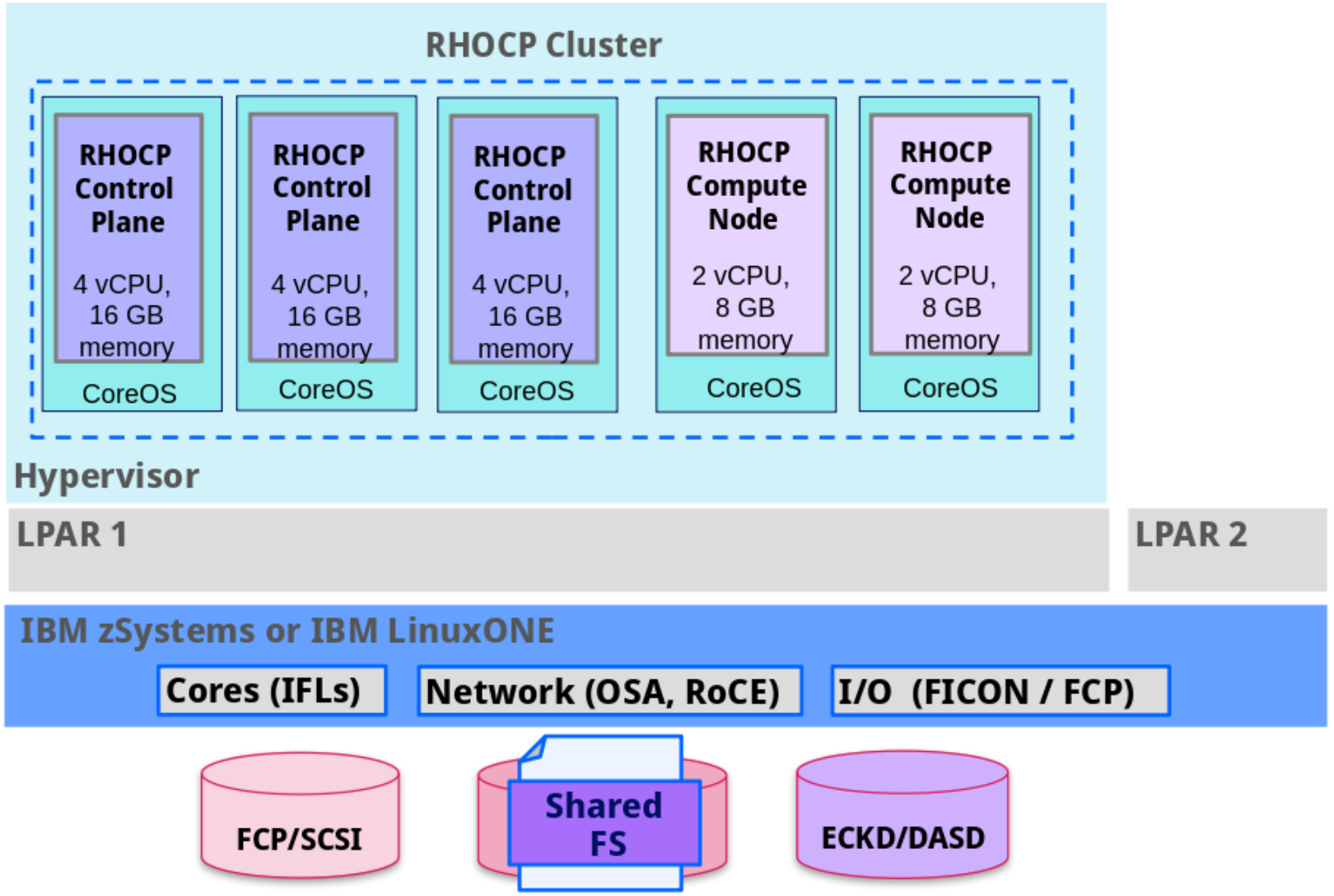
Enterprise Infrastructure

Edge

True Hybrid Multicloud

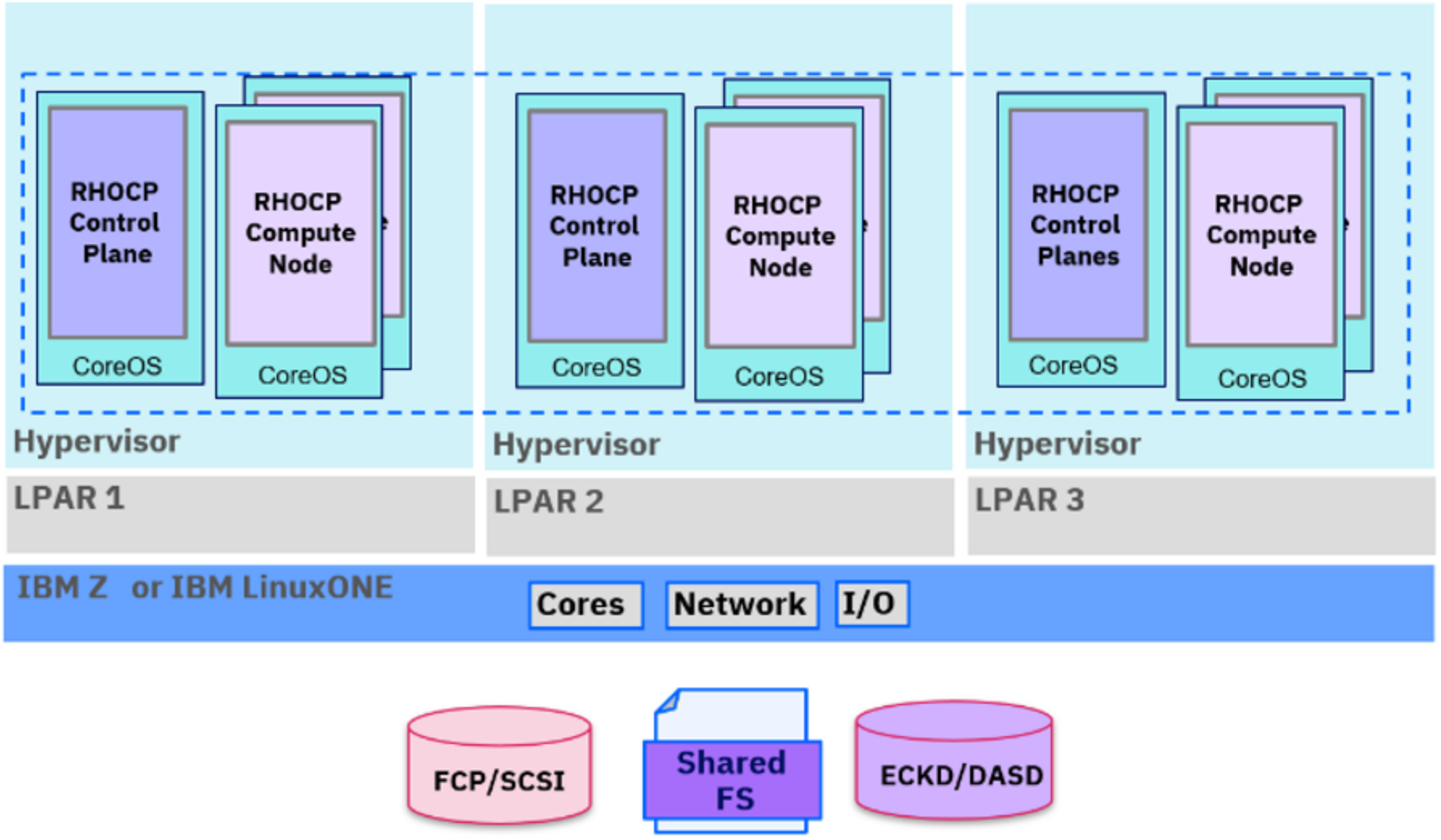


What does this look like on IBM Z?



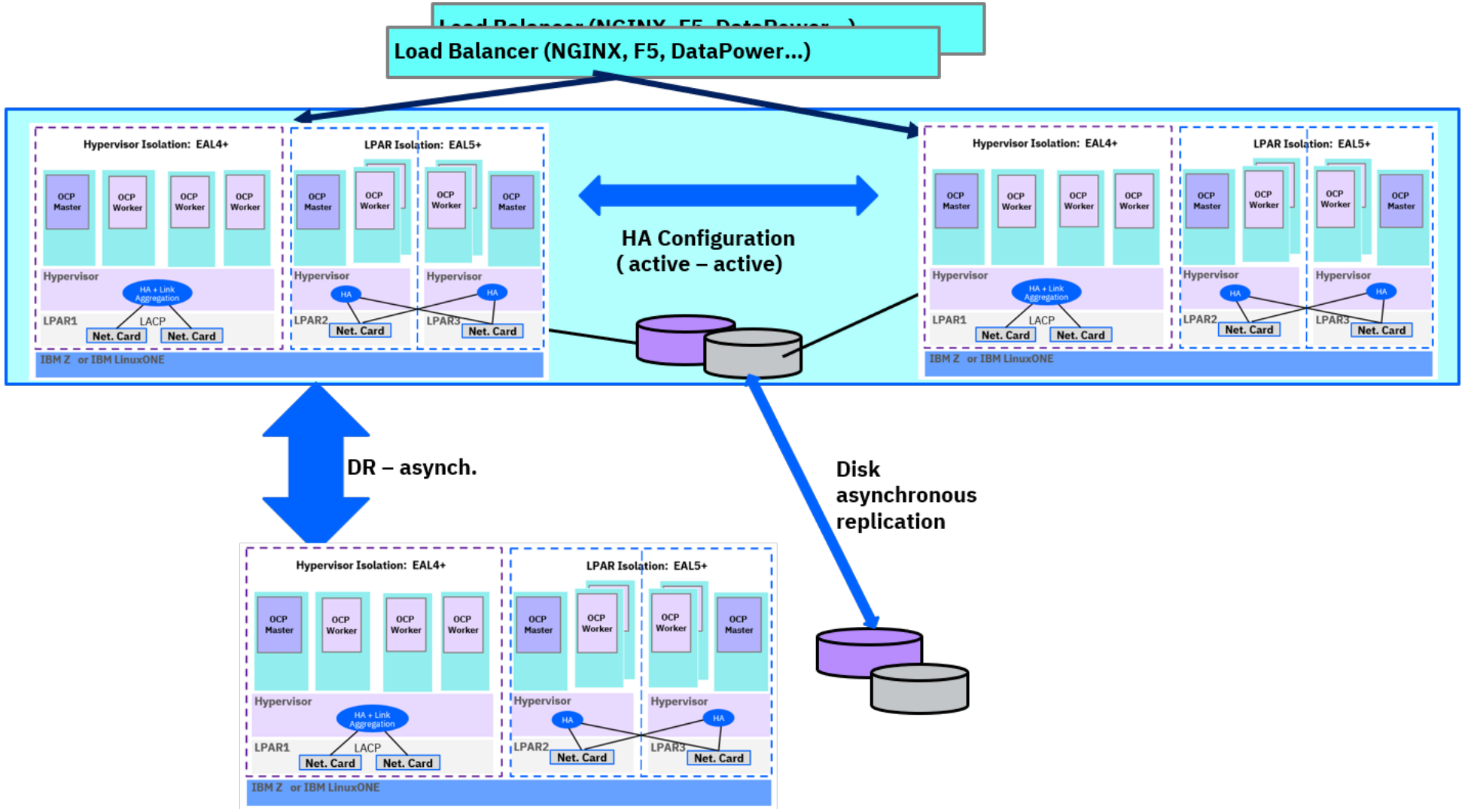
Source: [Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE - Reference Architecture]

What does this look like on IBM Z?



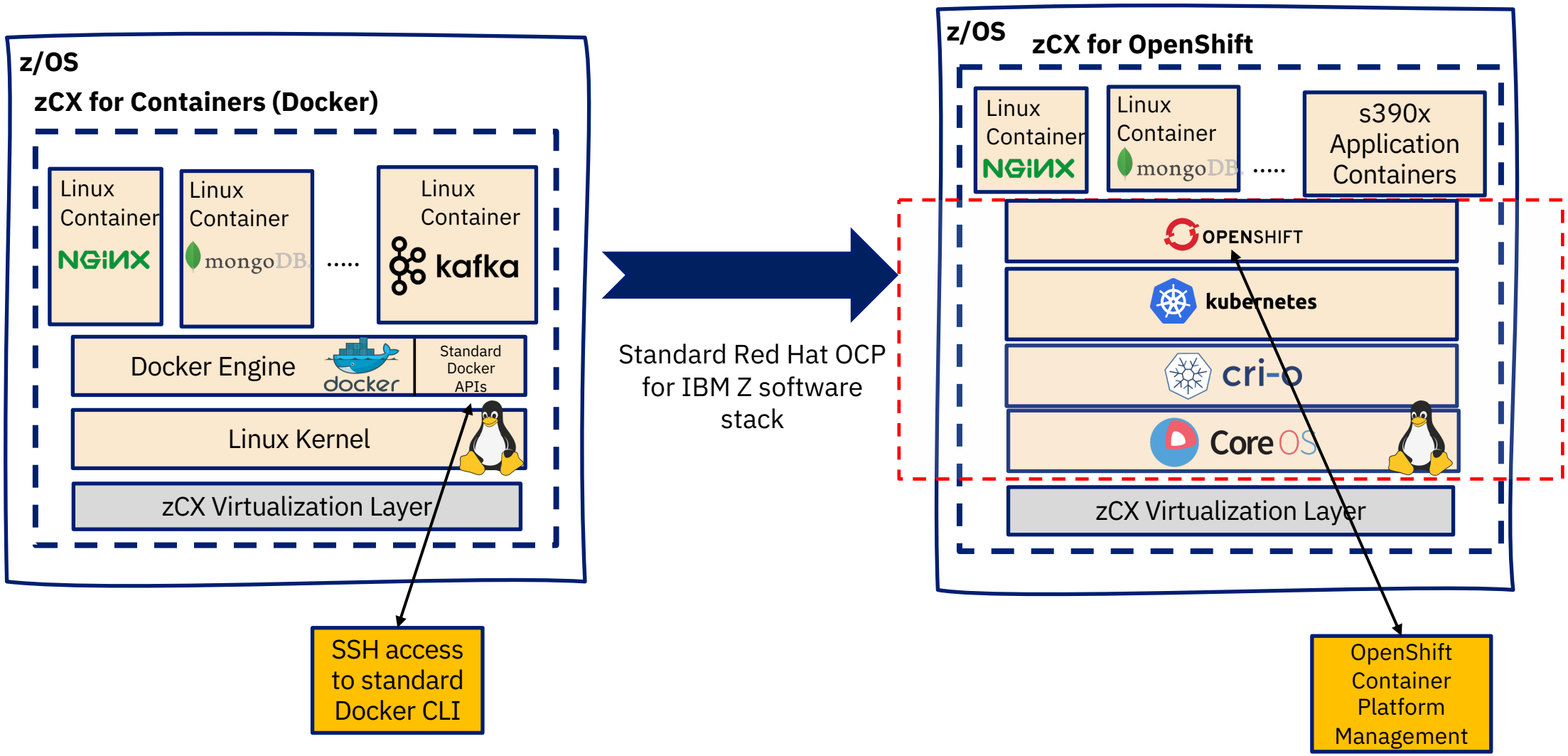
Source: [Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE - Reference Architecture]

What does this look like on IBM Z?



Source: [Red Hat OpenShift Container Platform on IBM Z and IBM LinuxONE - Reference Architecture]

zCX for Containers and zCX for OpenShift



IBM zCX Foundation for Red Hat OpenShift

Bringing Red Hat OpenShift Benefits to z/OS

- IBM zCX Foundation for Red Hat OpenShift that provides enterprise-level container orchestration and management capabilities around containerized software.
- Clients can extend and modernize their native z/OS ecosystem through an agile and flexible deployment of Linux on Z applications in a self-contained Red Hat OpenShift cluster on z/OS while exploiting z/OS Quality of Service.

z/OS Systems Programmer

A z/OS Systems Programmer will find that provisioning a Red Hat OpenShift cluster on zCX is similar to provisioning other z/OS middleware components

OpenShift Administrator

An OpenShift Administrator will find that using a Red Hat OpenShift cluster on zCX is the same as any other platforms

OpenShift Application Developer

An OpenShift Application Developer will find that developing applications for Red Hat OpenShift on zCX is the same as other platforms

OpenShift on IBM Z Use Cases

Modernization

Adopt cloud native to achieve consistency across the enterprise and grow benefit of containerized workloads

Colocation

Co-locate containerized workloads with z/OS and Linux based data to achieve lower response time and meet enterprise SLA

Platform capabilities

Benefit from high efficiency, high scalability, resiliency, out of the box availability, cryptography hardware*, low latency, and high throughput

Integration

Integration and automation of z/OS and Linux based workloads with hybrid cloud on IBM zSystems and IBM® LinuxONE

AI and Data

Leverage AI to extract insights and gain trusted, actionable results and move applications close to the data for better throughput and performance

Hyperledger fabric

Hyperledger fabric, the de facto standard for enterprise blockchain platforms, deployed on-premises on IBM zSystems and IBM® LinuxONE

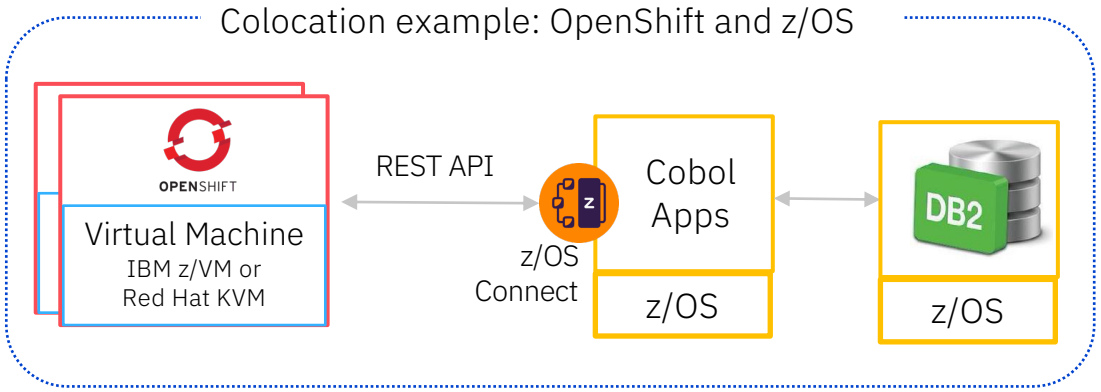
Benefits of co-locating workloads

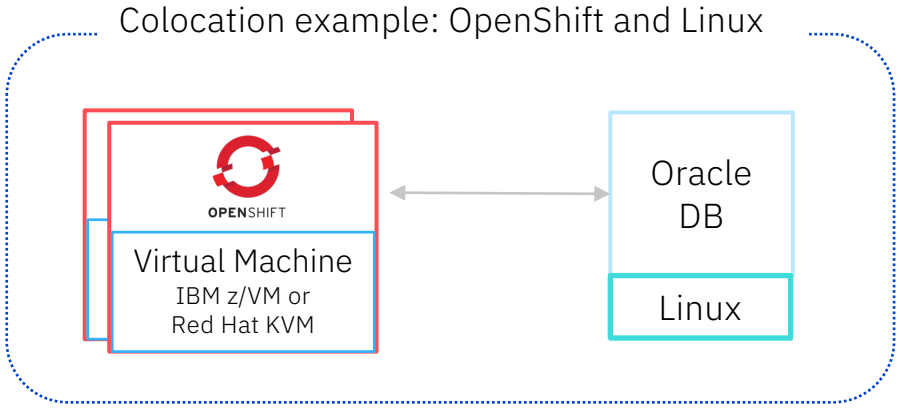
Colocation is when the presentation, business logic, and data serving layers of a **multi-tier workload onto a single physical server.**

Colocation can provide savings in

- **throughput improvements** for both interactive and streaming workloads,
- **reductions in latency** of network communication.

When accessing your database while running an OLTP workload on OpenShift Container Platform, achieve **4.2x more throughput by co-locating** the workload on IBM z16 versus running the workload on compared x86 platform connecting remotely to the IBM z16.*





* This is an IBM internal study designed to replicate banking OLTP workload usage in the marketplace deployed on OpenShift Container Platform (OCP) 4.9 on IBM z16 using z/VM versus on compared x86 platform using KVM accessing the same PostgreSQL 12 database running in an IBM z16 LPAR. IBM z16 configuration: The PostgreSQL database ran in a LPAR with 12 dedicated IFLs, 128 GB memory, 1 TB IBM FlashSystem 900 storage, RHEL 7.7 (SMT mode). The Compute nodes ran on z/VM 7.2 in a LPAR with 30 dedicated IFLs, 188 GB memory, DASD storage, and OSA connection to the PostgreSQL LPAR. LPAR with 2 IFL, 4 GB memory and RHEL 8.5 with OCP Proxy server. x86 configuration: The Compute nodes ran on KVM on RHEL 8.5 on 32 Cascade Lake Intel® Xeon® Gold CPU @ 2.30 GHz with Hyperthreading turned on, 192 GB memory, RAID5 local SSD storage, and 10Gbit Ethernet connection to the PostgreSQL LPAR. Results may vary.

Large client in NA

Co-location with low latency on IBM Z

Solution Benefits

7.3x lower transaction latency compared to the equivalent distributed systems architecture

Developers got a platform agnostic development environment

Extend the container platform all the way to IBM Z

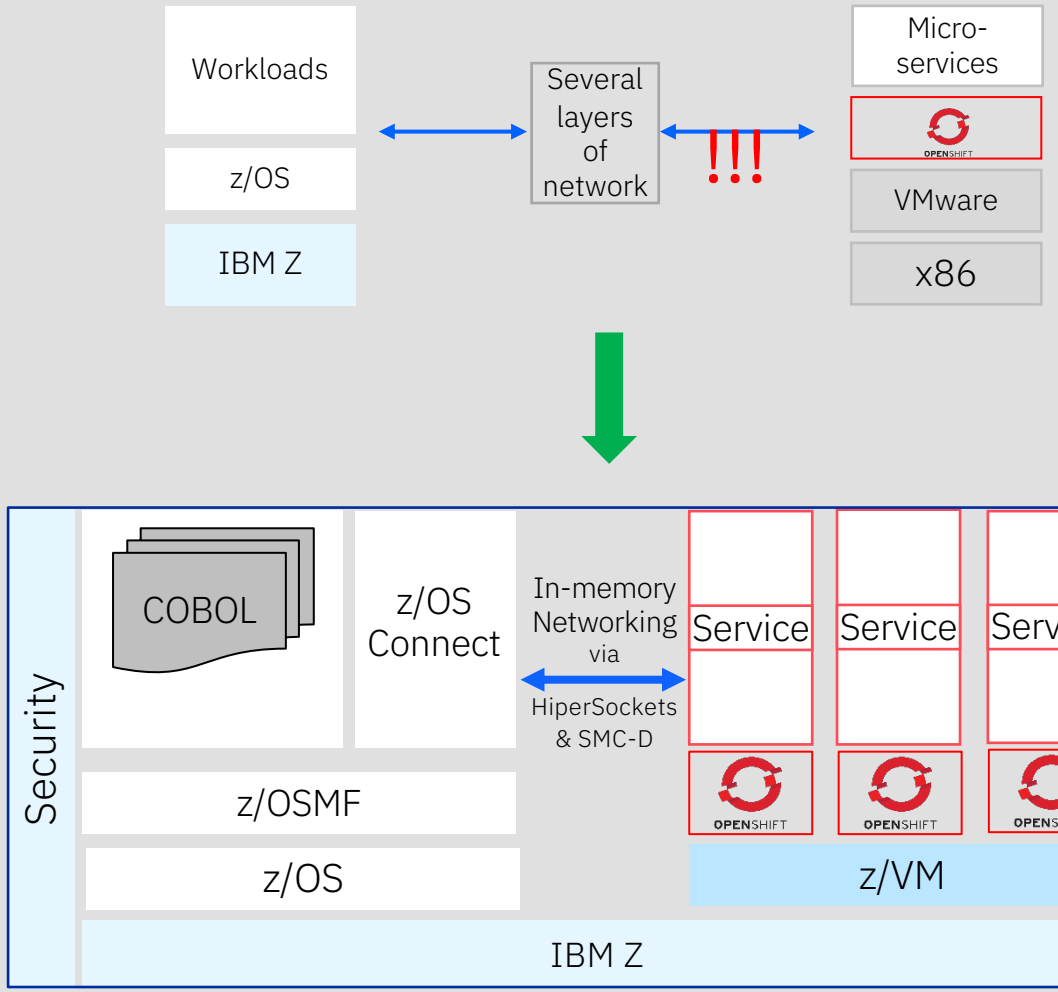
Exploiting co-location on IBM Z, the clients benefits from lower transaction latency compared to the equivalent distributed architecture, offering a more competitive and efficient service to its customers.

Business Requirements

Increase competitive business offerings by extending and modernizing the integration with existing assets while maintaining SLAs and keeping risk and cost low.

In Production

Containerized services running in OpenShift are co-located on the same IBM Z system with z/OS Db2 data and CICS for low latency, high volume transaction processing



European Bank

Modernization from large monolithic to an agile configuration

Business Requirements

Client needed to improve agility and minimize risk of large monolithic integration broker and MQ components that support critical business applications.

Client wanted to modernize to containerized microservices, still benefitting from the reliability and scalability with IBM Z.

In Production

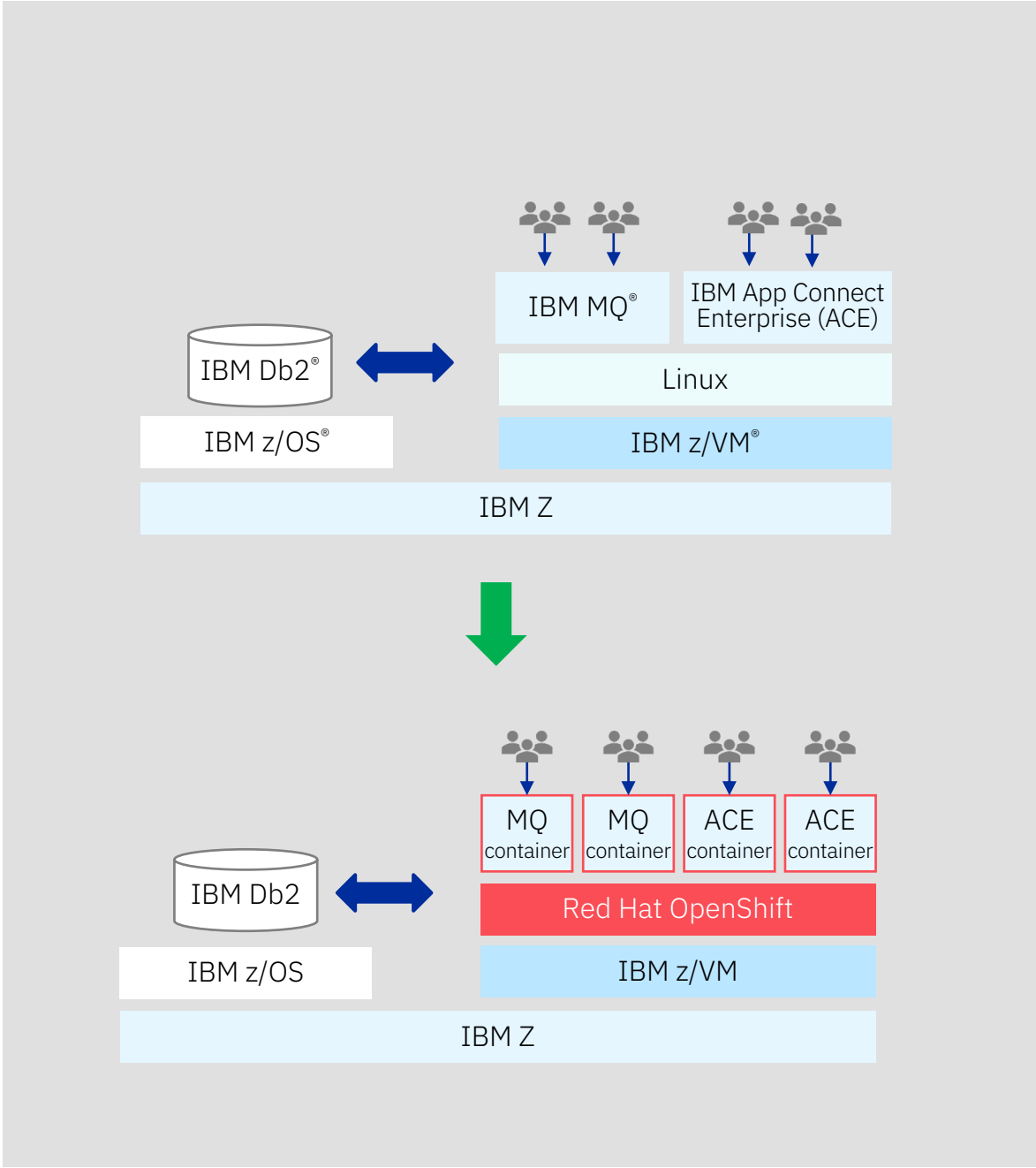
Modernization from large monolithic ‘integration broker’ to more agile configuration – still using the benefits of co-location on IBM Z.

Solution Benefits

The client decided to implement Red Hat OpenShift on IBM Z and ‘IBM Cloud Pak® for Integration’ to take advantage of the platform’s scalability, reliability, and lower TCO.

The client is taking advantage of the containerized ‘IBM App Connect Enterprise’ server and ‘IBM MQ’ instances to allow for a more agile development and production rollout of various microservices instead of changing the current large monolithic implementation.

Using Red Hat OpenShift along with pipeline technologies enables the client to be more responsive to business needs.



Large Banks in South America

Modernization from non-container workloads to an agile containerized configuration

Proposed IBM solution to all other banks (PoT)

IBM demonstrated a Proof of Technology (PoT) running a containerized solution for Kafka (Red Hat AMQ Streams) using Red Hat OpenShift Container Platform running on IBM Z leveraging the KVM hypervisor technology.

Solution Results

- High flexibility to update Kafka workloads running on Red Hat OpenShift
- Maintain strict latency requirements
- Used less 1/3 of the hardware requirements from x86
- Less software licenses
- Adhere with sustainability goals

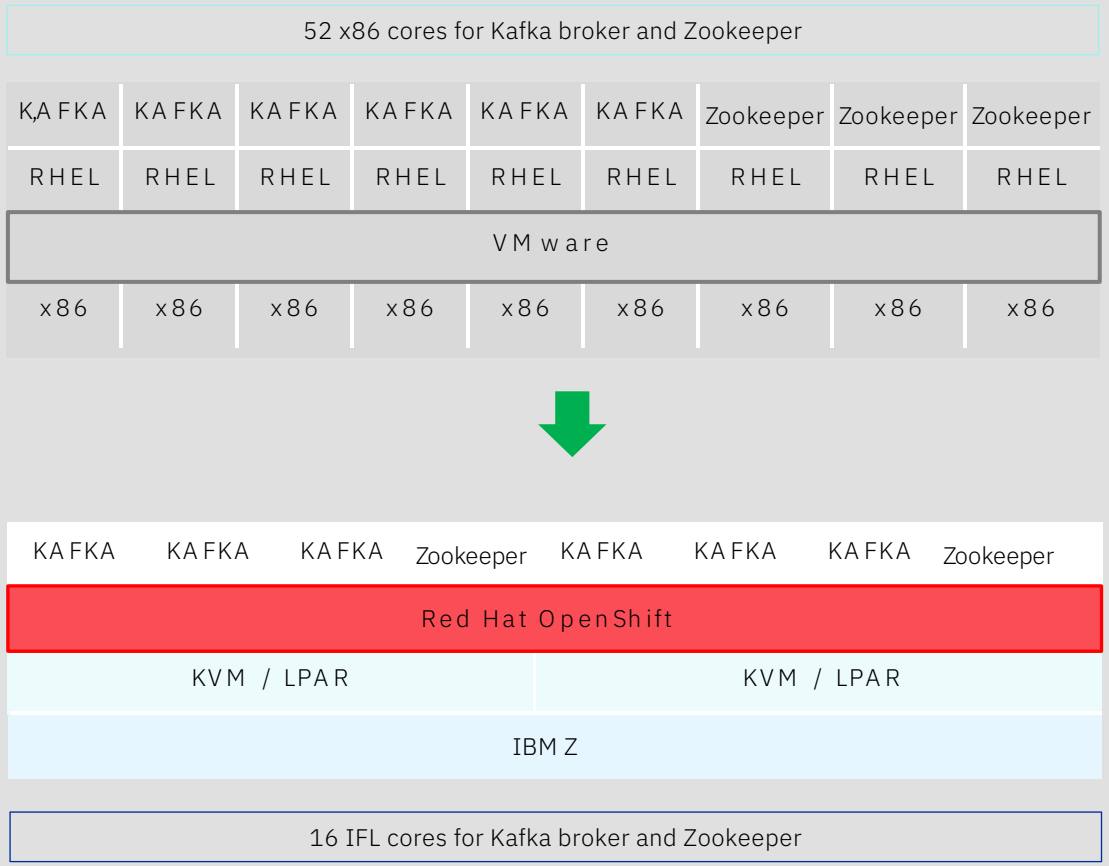
Challenge

The Central Bank developed a national peer-to-peer payment system where every transaction processed by any bank in the country would have to be validated by the central bank. Each bank can develop their own solutions, but they must comply with the strict low latency requirements to be compliant with the standards from the Central Bank.

The initial solution was deployed using VMware to host virtual machines running the Kafka workloads. Although the initial solution can meet the latency requirements, it does not offer the flexibility that a true container native solution can deliver, it and requires large amounts of hardware thus software licenses.

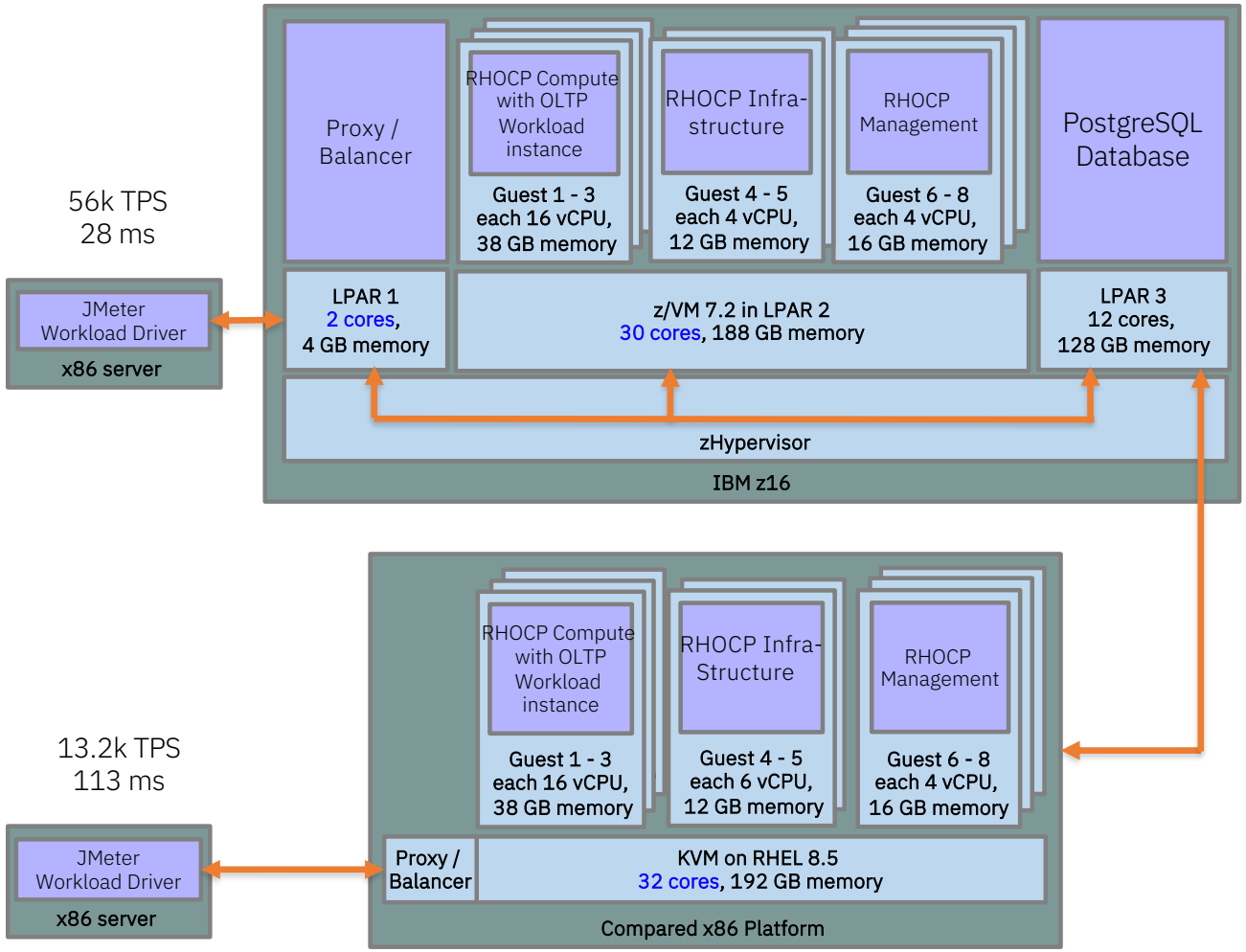
Central Bank latency requirements

- ~10 ms per message
- 600 thousand messages



Database co-location with Red Hat OpenShift on IBM z16 versus remote database access from x86

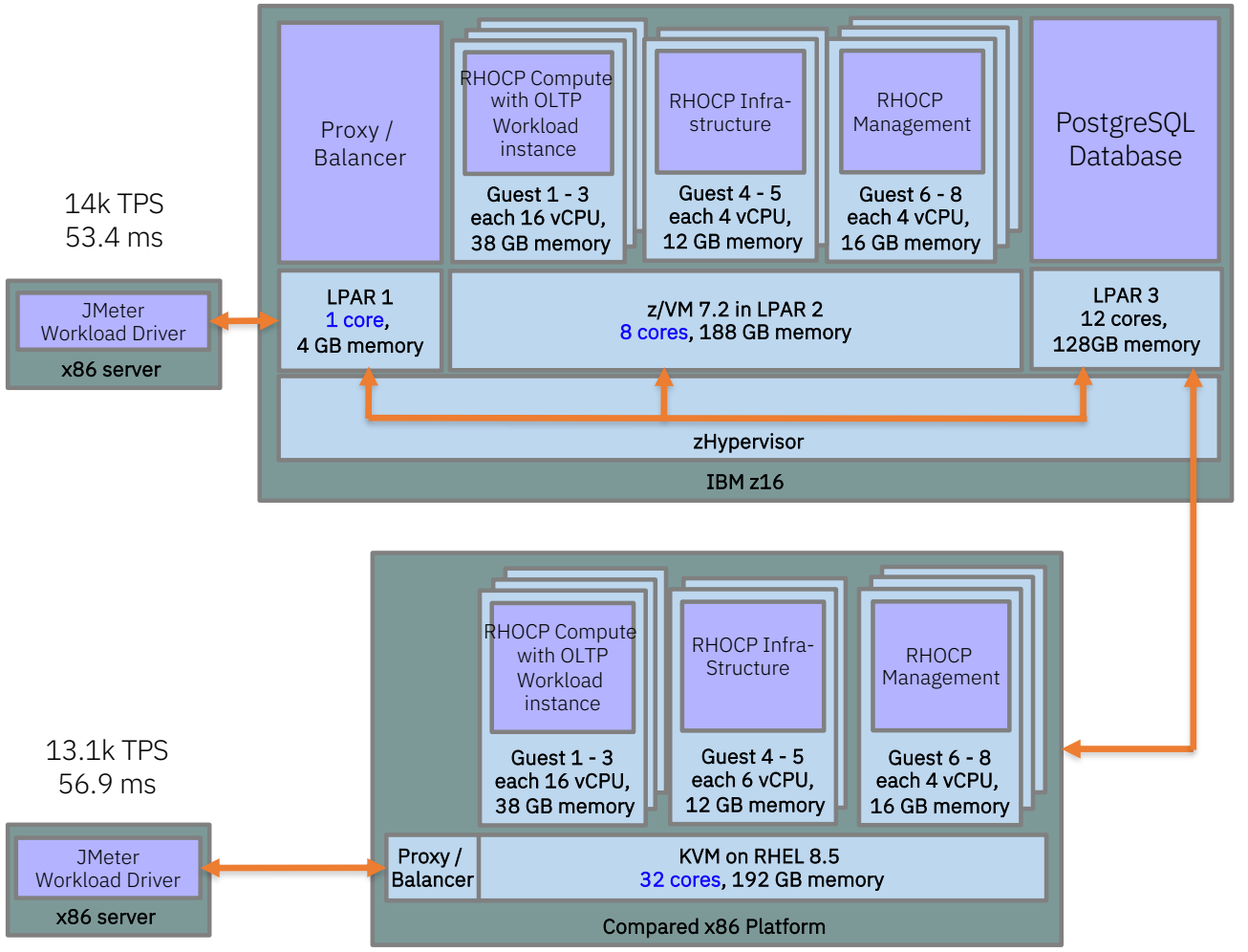
When accessing your database while running an **OLTP workload on Red Hat OpenShift Container Platform**, achieve 4.2x more throughput by co-locating the workload on IBM z16 versus running the workload on compared x86 platform connecting remotely to the IBM z16



DISCLAIMER: This is an IBM internal study designed to replicate banking OLTP workload usage in the marketplace deployed on Red Hat OpenShift Container Platform (RHOCP) 4.9 on IBM z16 using z/VM versus on compared x86 platform using KVM accessing the same PostgreSQL 12 database running in an IBM z16 LPAR. Results may vary. IBM z16 configuration: The PostgreSQL database ran in an LPAR with 12 dedicated cores, 128 GB memory, 1TB FlashSystem® 900 storage, RHEL 7.7 (SMT mode). The Compute nodes ran on z/VM 7.2 in an LPAR with 30 dedicated cores, 188 GB memory, DASD storage, and OSA connection to the PostgreSQL LPAR. LPAR with 2 cores, 4GB memory and RHEL 8.5 with RHOCP Proxy server. x86 configuration: The Compute nodes ran on KVM on RHEL 8.5 on 32 Cascade Lake Intel® Xeon® Gold 5218 CPU @ 2.30GHz with Hyperthreading turned on, 192 GB memory, RAID5 local SSD storage, and 10Gbit Ethernet connection to the PostgreSQL LPAR.

Database co-location with Red Hat OpenShift on IBM z16 database access from x86

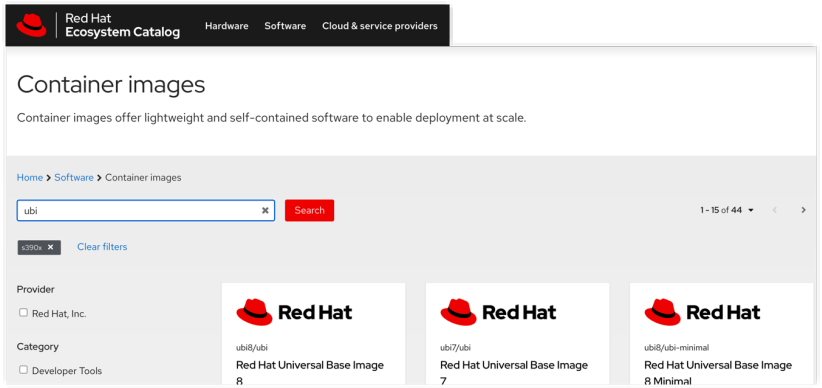
Accessing your database while running an **OLTP workload on Red Hat OpenShift Container Platform**, requires up to 3.6x fewer cores running your workload when co-located on IBM z16 versus running the workload on compared x86 platform connecting remotely to the IBM z16



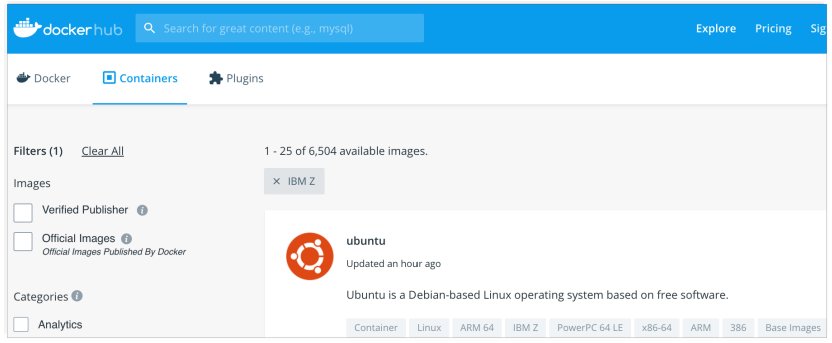
DISCLAIMER: This is an IBM internal study designed to replicate banking OLTP workload usage in the marketplace deployed on Red Hat OpenShift Container Platform (RHOCP) 4.9 on IBM z16 using z/VM versus on compared x86 platform using KVM accessing the same PostgreSQL 12 database running in an IBM z16 LPAR. IBM z16 configuration: The PostgreSQL database ran in an LPAR with 12 dedicated cores, 128 GB memory, 1TB FlashSystem 900 storage, RHEL 7.7 (SMT mode). The Compute nodes ran on z/VM 7.2 in an LPAR with 8 dedicated cores, 188 GB memory, DASD storage, and OSA connection to the PostgreSQL LPAR. The RHOCP Proxy server ran in an LPAR with 1 core, 4 GB memory and RHEL 8.5. x86 configuration: The Compute nodes ran on KVM on RHEL 8.5 on 32 Cascade Lake Intel® Xeon® Gold 5218 CPU @ 2.30GHz with Hyperthreading turned on, 192 GB memory, RAID5 local SSD storage, and 10Gbit Ethernet connection to the PostgreSQL LPAR. Both systems are delivering equal throughput. Results may vary.

Container images available for IBM Z and LinuxONE (and zCX)

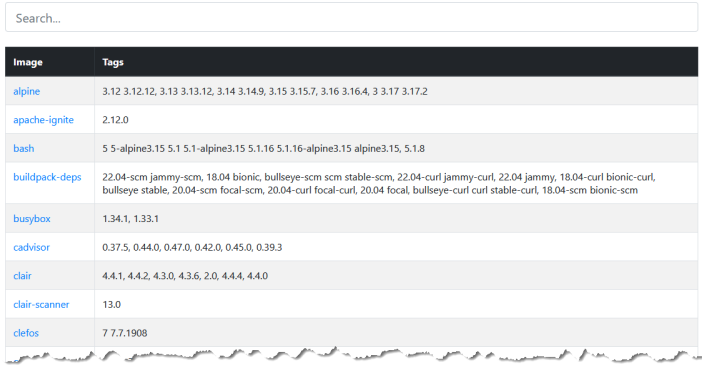
Red Hat Container Catalog provides 800+ s390x container images



Dockerhub provides more than 10 thousand+ container images for s390x, and 550+ certified, official and Verified images



IBM.registry
icr.io 1000+



https://www.linkedin.com/pulse/red-hat-container-catalog-access-270-images-s390x-from-filipe-miranda/

Large client in NA

Co-location with low latency

Multi-arch development & deployment

Business Requirements

- Increase competitive business offerings by extending and modernizing
- Maintain SLAs
- Keep risk and cost low

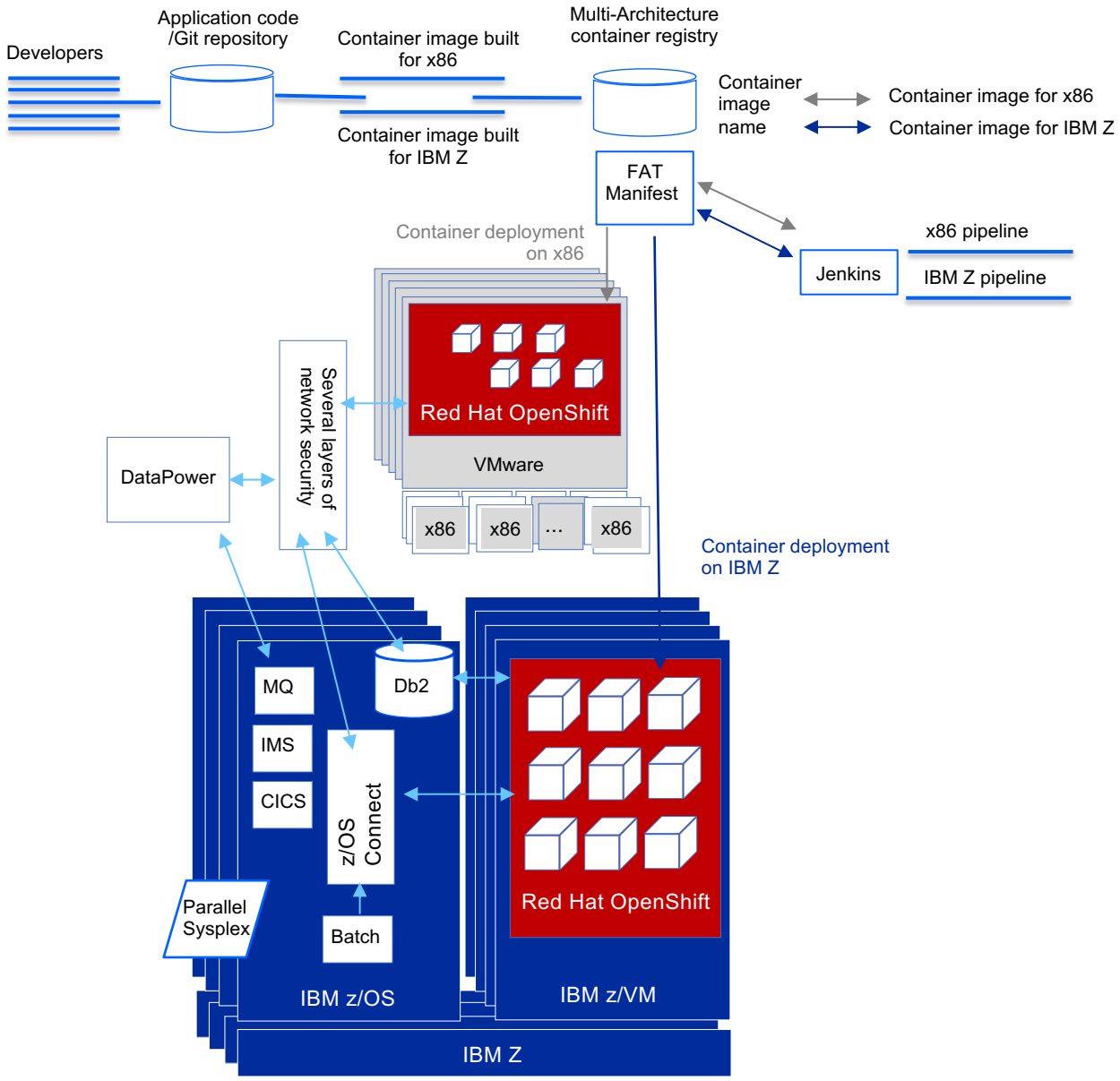
Solution

Containerized services running in Red Hat OpenShift are co-located to IBM z/OS workloads.

Creating a model where applications running on Red Hat OpenShift on IBM Z and x86 can share the same processes for development and deployment as multi-arch applications are developed once and deployed where it makes sense.

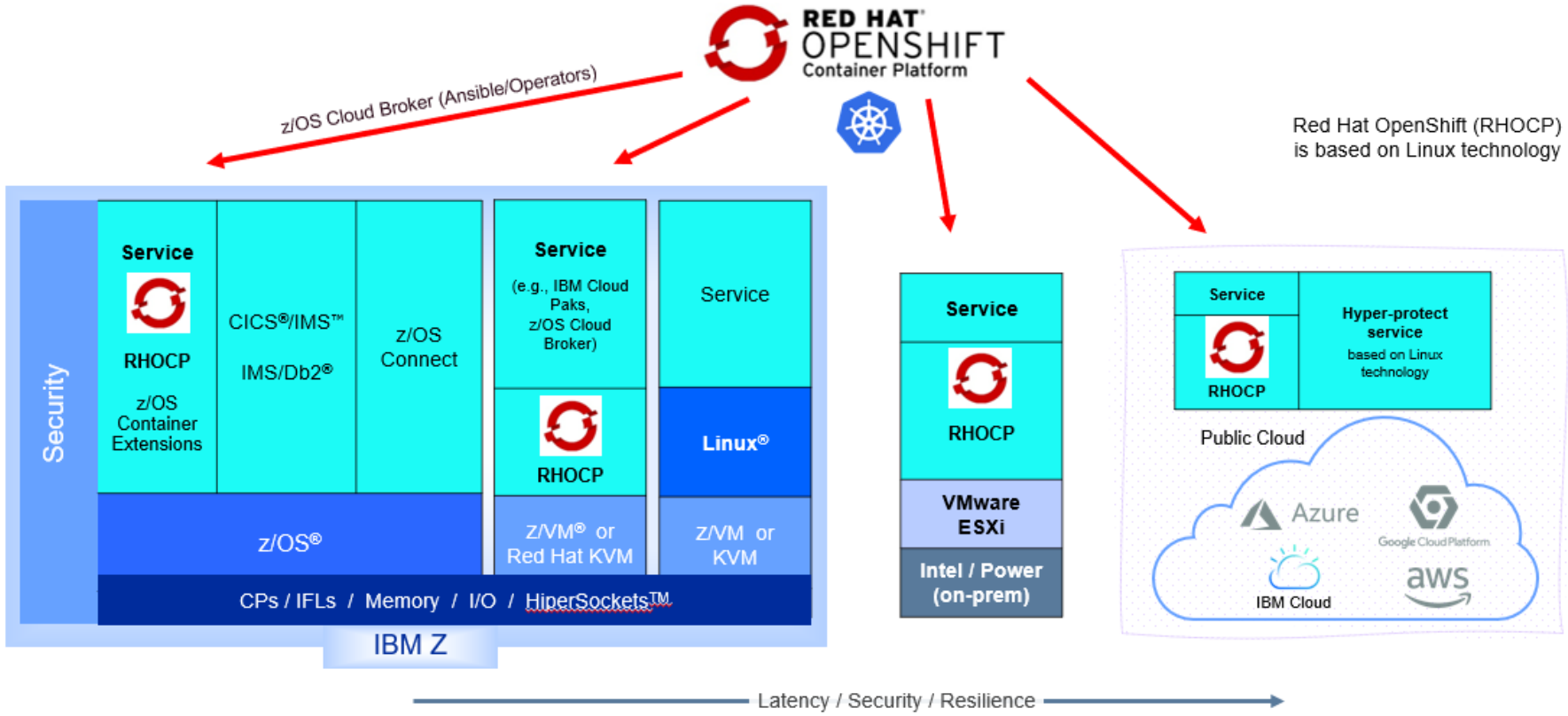
Solution Benefits

- Red Hat OpenShift deployments on IBM Z and x86 are sharing the same container registry creating a single registry for all architectures
- Developers got platform agnostic development environment
- Modernized development and deployment - all the way - through the usage of containers on IBM Z



The vision of hybrid cloud and multicloud with Red Hat OpenShift

Hybrid workloads and multiple Red Hat OpenShift clusters can run in parallel on a physical IBM Z server.



Thank you

Matt Mondics
Technical Sales Enablement Specialist - OpenShift on IBM Z
—
matt.mondics@ibm.com
IBM Washington Systems Center (WSC)

Part Two:
Red Hat OpenShift Lifecycle, Installation, Upgrades, and Options

Empowerment Promise

By the end of this segment, you will…

1) Understand the different options for deploying OpenShift on IBM zSystems.

2) Understand the requirements for a successful deployment.

3) Understand the maintenance and upgrade process and support.

On-Prem IBM Z / LinuxONE

OPENSHIFT CONTAINER PLATFORM

Installer-Provisioned Infrastructure

Simplified opinionated "Best Practices" for cluster provisioning

Fully automated installation and updates including host container OS.

Red Hat Enterprise Linux CoreOS

User-Provisioned Infrastructure

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries

Red Hat Enterprise Linux CoreOS

Red Hat Enterprise Linux

HOSTED OPENSHIFT

IBM Cloud Red Hat OpenShift

Get a powerful cluster in the IBM Cloud, fully managed by IBM engineers and support.

Azure Red Hat OpenShift

Deploy directly from the Azure console. Jointly managed by Red Hat and Microsoft Azure engineers.

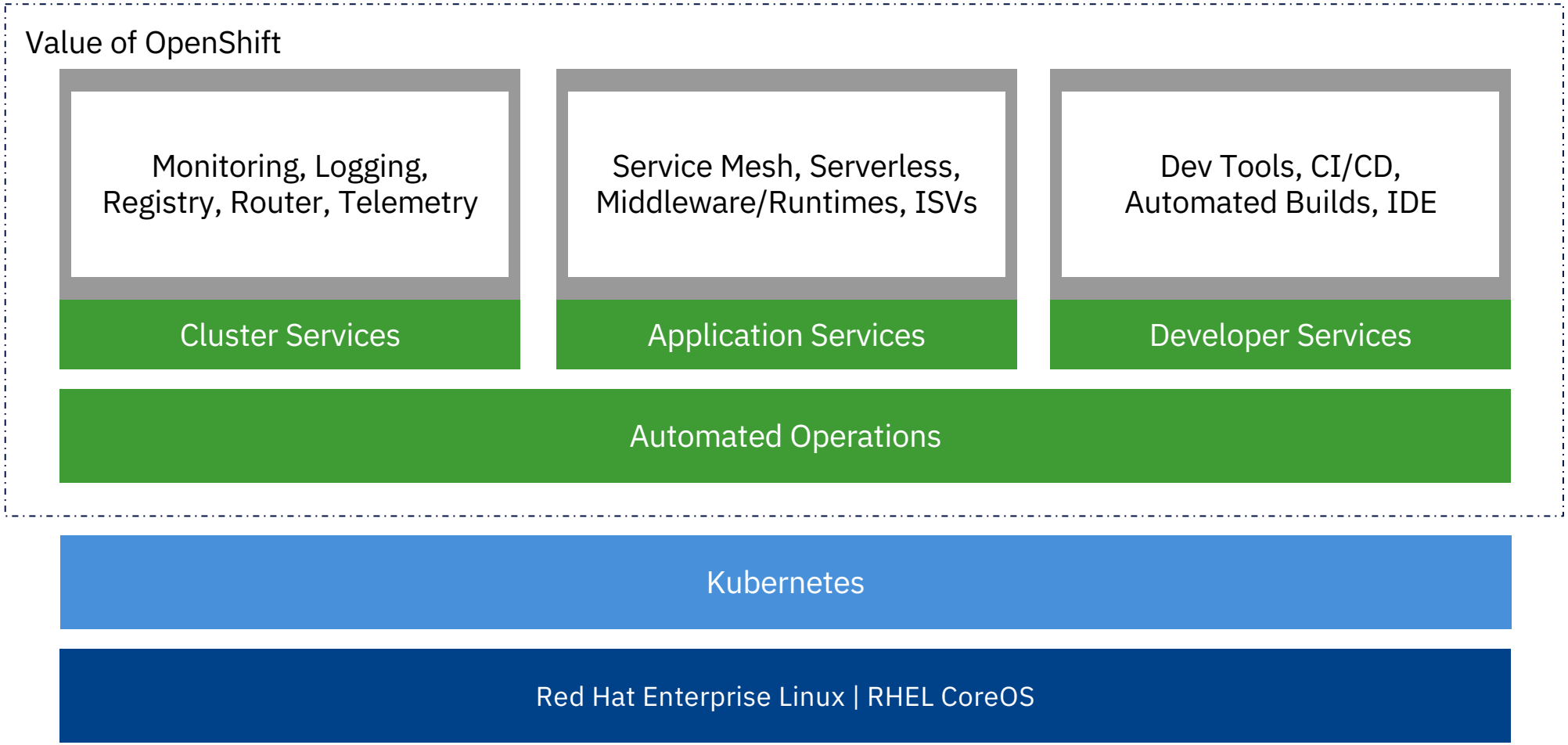
OpenShift Dedicated

Get a powerful cluster, fully managed by Red Hat engineers and support.

Installation Paradigms

	Installer-Provisioned Infrastructure	User-Provisioned Infrastructure
Build Network	Installer	User
Setup Load Balancers	Installer	User
Configure DNS	Installer	User
Hardware/VM Provisioning	Installer	User
OS Installation	Installer	User
Generate Ignition Configs	Installer	Installer
OS Support	Installer: RHEL CoreOS	User: RHEL CoreOS
Node Provisioning / Autoscaling	Yes	Only for providers with OpenShift Machine API support

Comparison of Paradigms

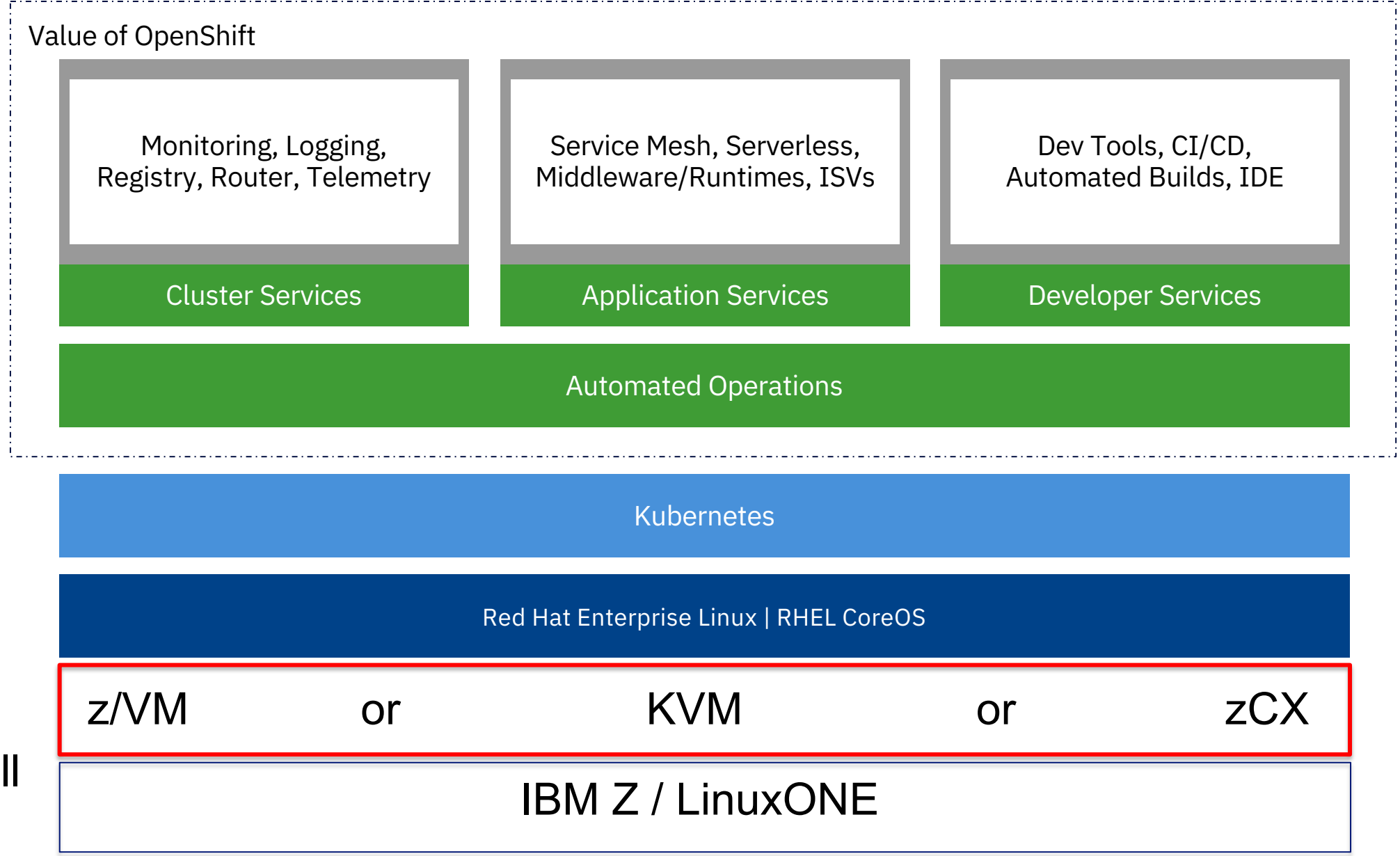


Best IT Ops Experience

CaaS ⟷ PaaS ⟷ FaaS

Best Developer Experience

Hypervisor



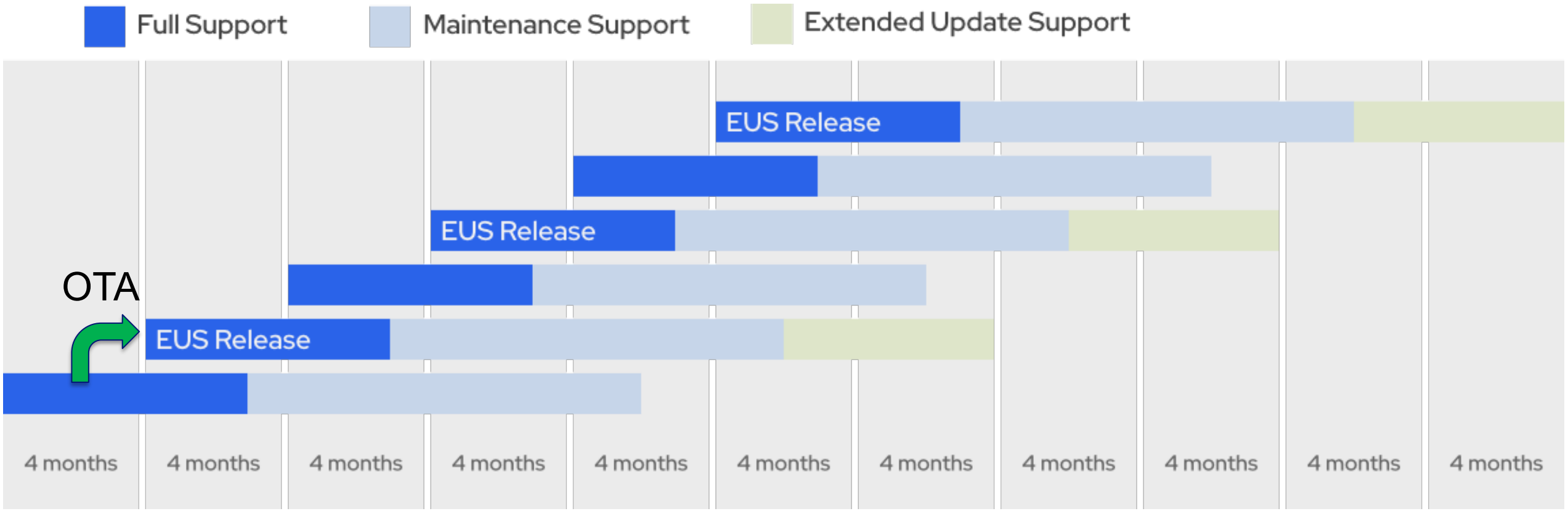
- DNS
- LB
- Firewall

Hypervisor

Method	Hypervisor	Paradigm	Automated?	Instructions
Manual z/VM	z/VM	UPI	No	Link
Manual KVM	KVM	UPI	No	Link
z/OSMF zCX	zCX	UPI	Yes	Link
ICIC	Z/VM & KVM	UPI	Yes	Link
z/VM ESI	z/VM	UPI	Yes	Link
OAAKZ	KVM	UPI	Yes	Link
RH Assisted Installer	KVM	UPI	Yes	Link

Deployment Options

Support, Upgrades and Migrations



Support Timelines

3 Key Takeaways

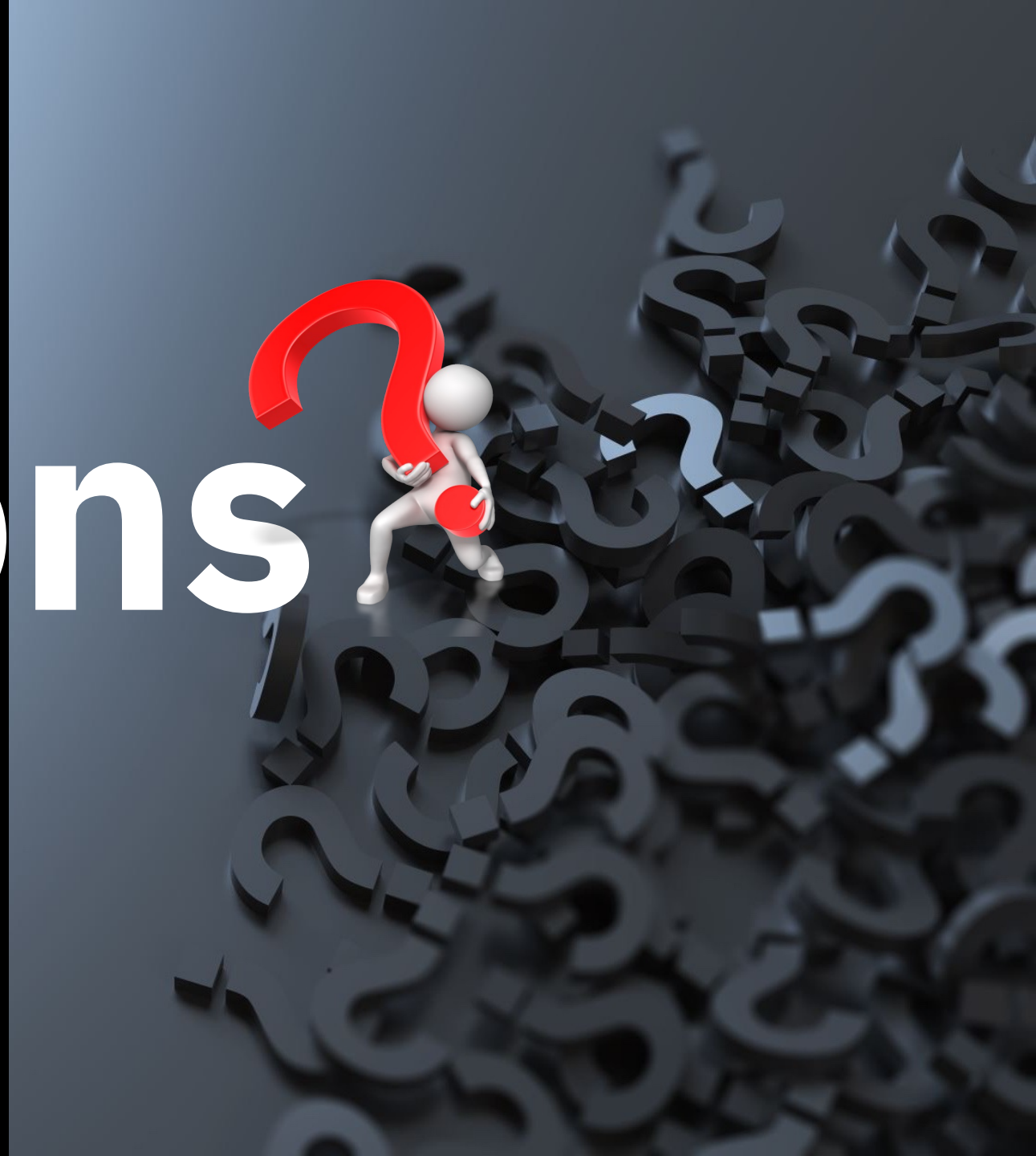
Checking back in on my empowerment promise.

1) Select the hypervisor options based on your needs and existing infrastructure.

2) Since UPI must be used, give ample time for deployment. Have a plan in place, all necessary parties on board, and an empowered deployment team.

3) Staying up-to-date with OpenShift releases is critically important for maintaining proper support, and easy to do – so don't fall behind! Make sure you have a plan in place for cluster maintenance *before* deployment.

Questions



Thank you

Jacob Emery
Technical Enablement Specialist - OpenShift on IBM zSystems and Ansible
—
jacob.emery@ibm.com
IBM Washington Systems Center (WSC)

Red Hat OpenShift Container Platform on IBM Z & LinuxONE

All workshop materials can be found here:
https://ibm.biz/ocp-z-workshop

On break until 11:15

Sche e day

Start 10:00 AM Eastern

- **Presentation**
  - Overview of OpenShift
- Short Break
- **Presentation**
  - Mini Poll
  - OpenShift on Z technical deep dive
- Short Break
- **Presentation**
  - Mini Poll
  - Installation
  - Requirements
  - Options
- Short Break
- **Lab exercises**
  - Mini Poll
  - Self-paced
  - Non-sequential

End 4:00 PM Eastern

Functional overview

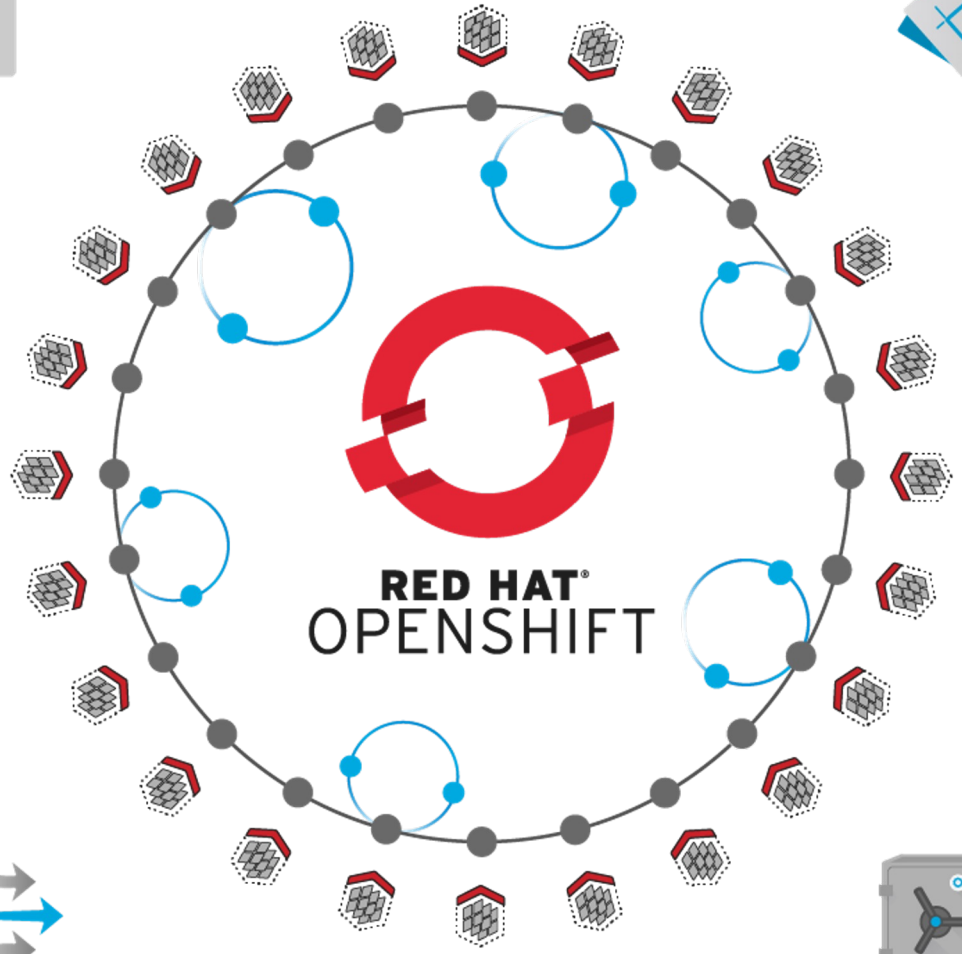
Self-Service

Multi-language

Automation

Collaboration

Multi-tenant



Standards-based

Web-scale

Open Source

Enterprise Grade

Secure

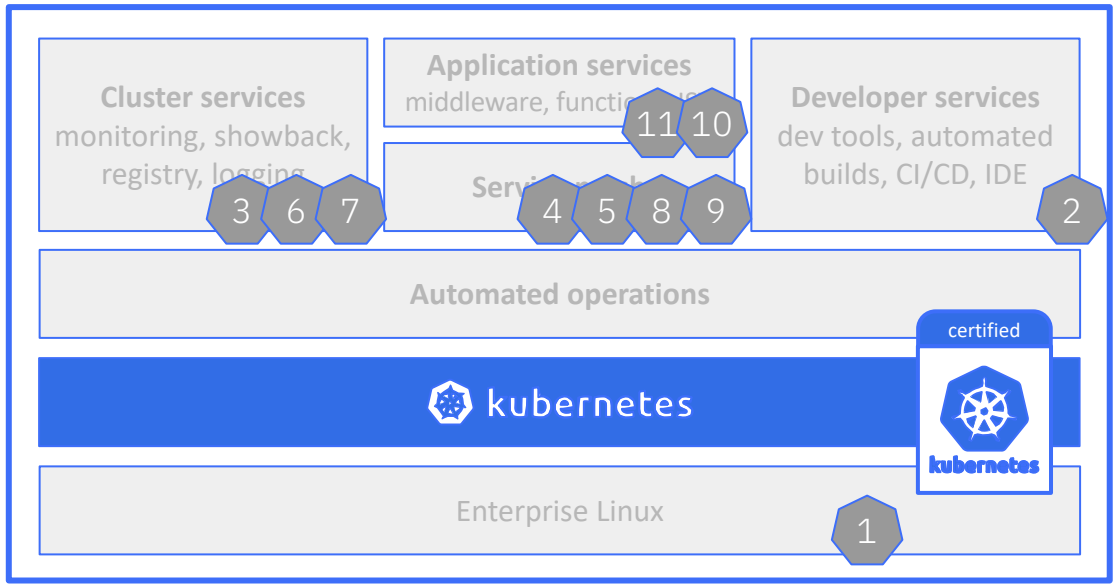
OPENSHIFT CONTAINER PLATFORM | Technical Value

kubernetes

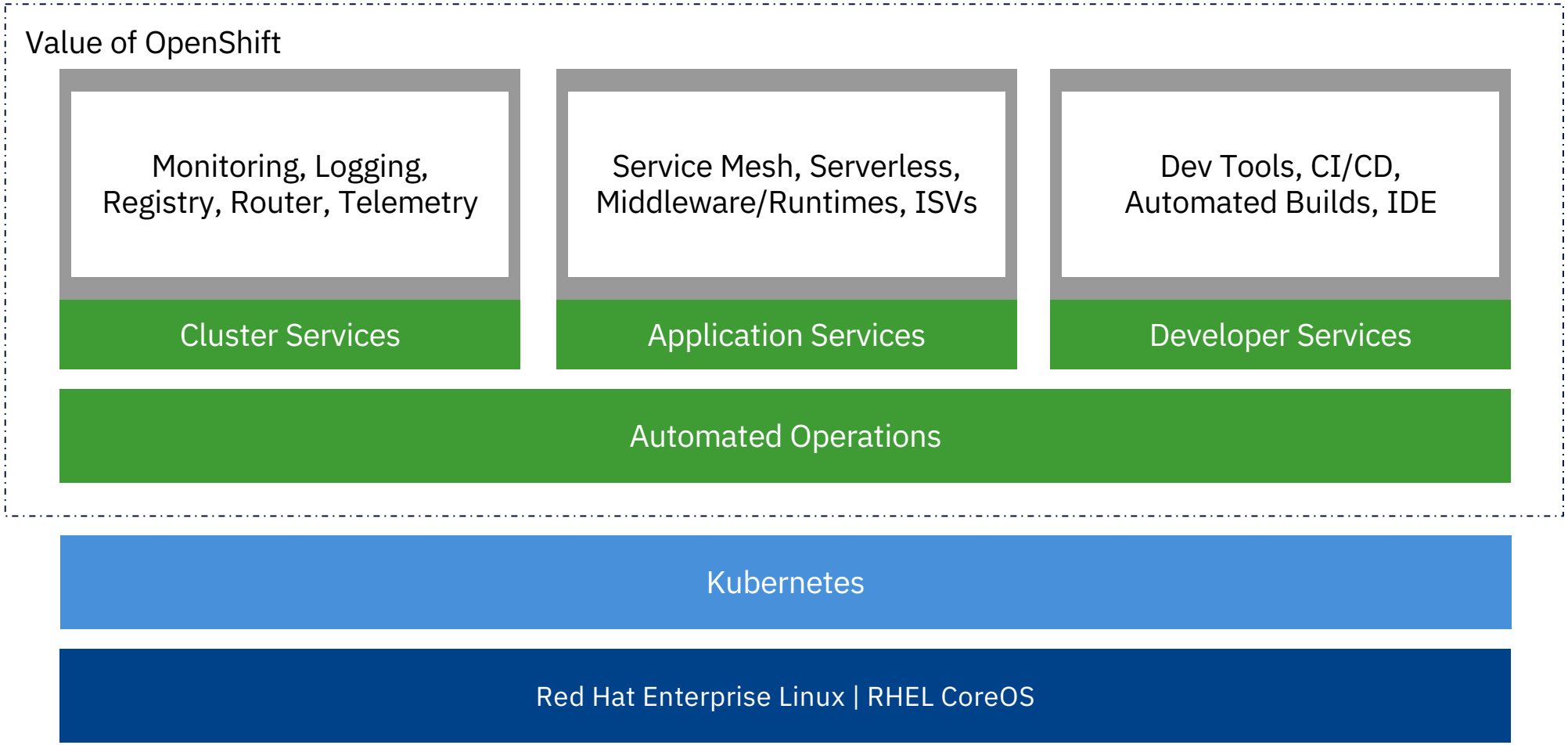
Lacks many essential components

1. Operating system
2. Container runtime (CRI-O, Containerd, Docker, etc).
3. Image registry
4. Software-defined networking
5. Load-balancer and routing
6. Log management
7. Container metrics and monitoring
8. DNS
9. Load balancing
10. Ingress
11. RBAC

The customer (or third-party) must configure, integrate, operate and support additional components to be fully operational.



What's needed to put Kubernetes into production?

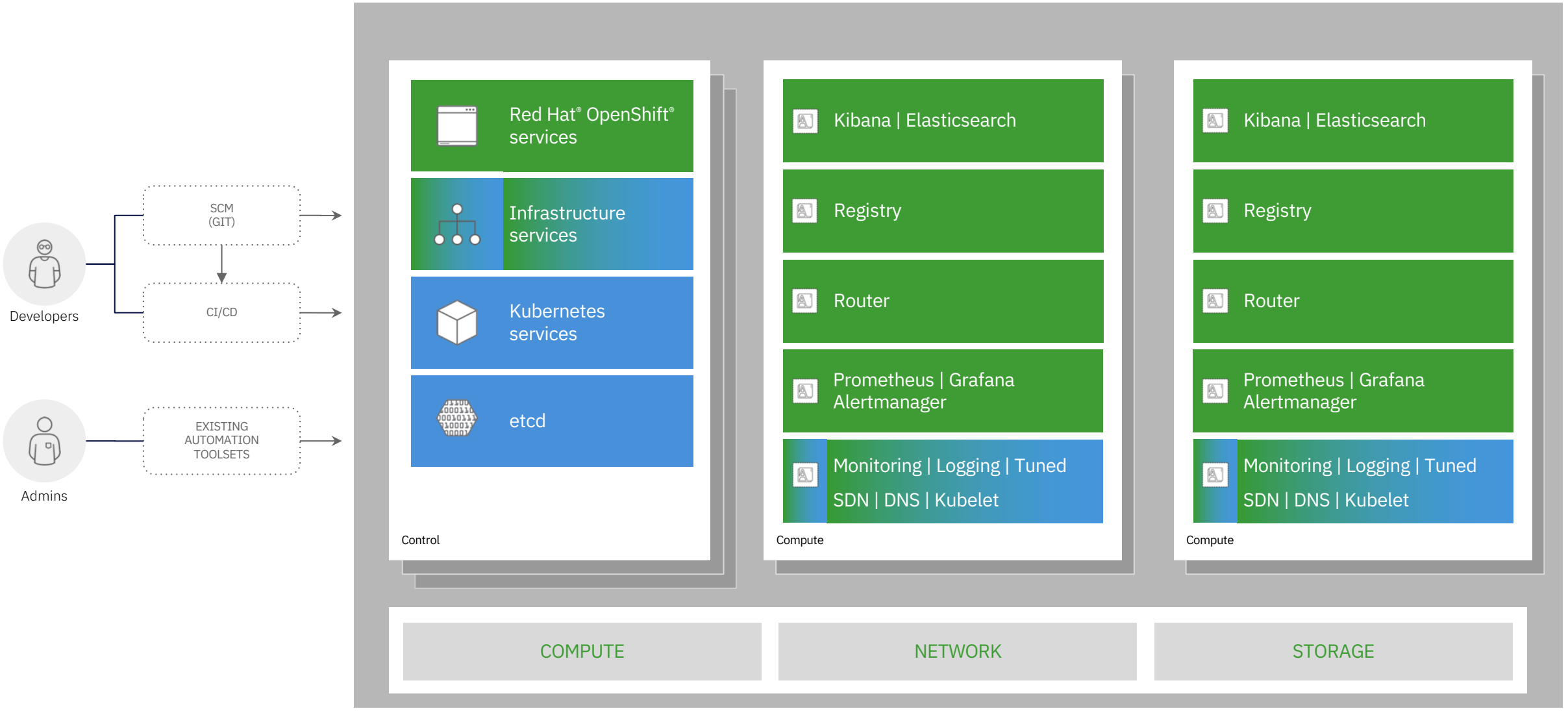


Best IT Ops Experience

CaaS ⟷ PaaS ⟷ FaaS

Best Developer Experience

A best-of-breed concept

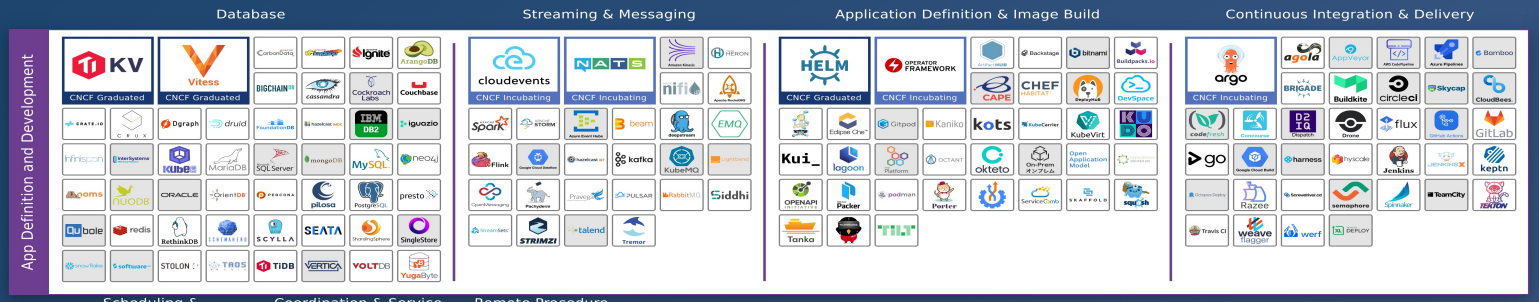


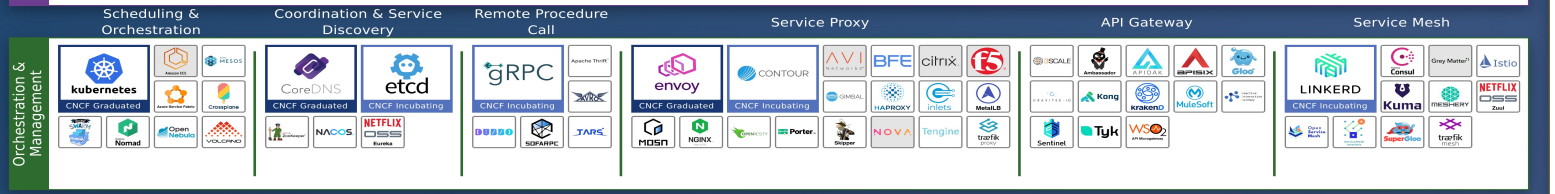
Architectural overview

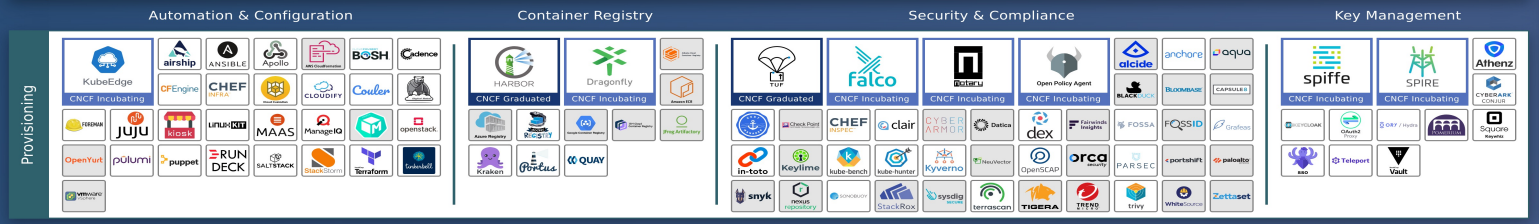
CNCF Cloud Native Landscape
2021-01-07T00:35:31Z 0480c506

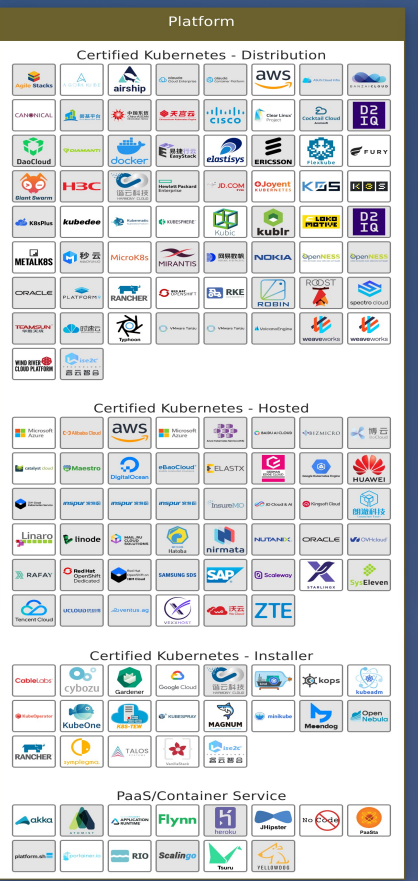
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

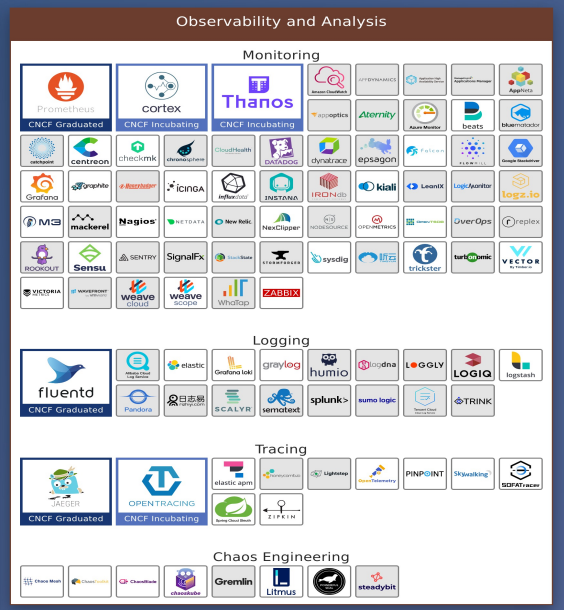
Greyed logos are not open source

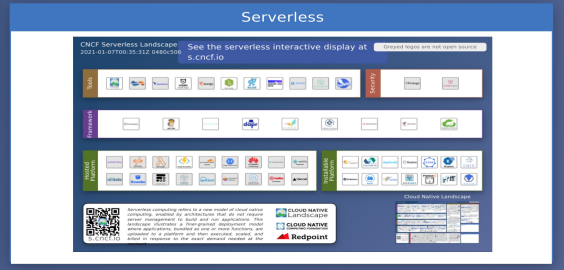


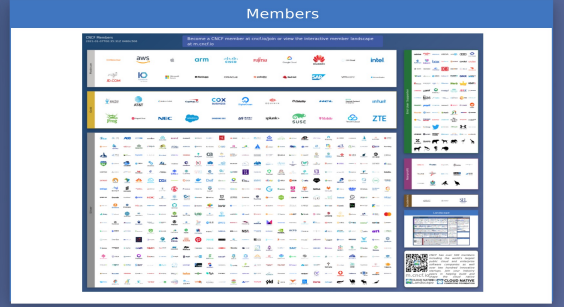










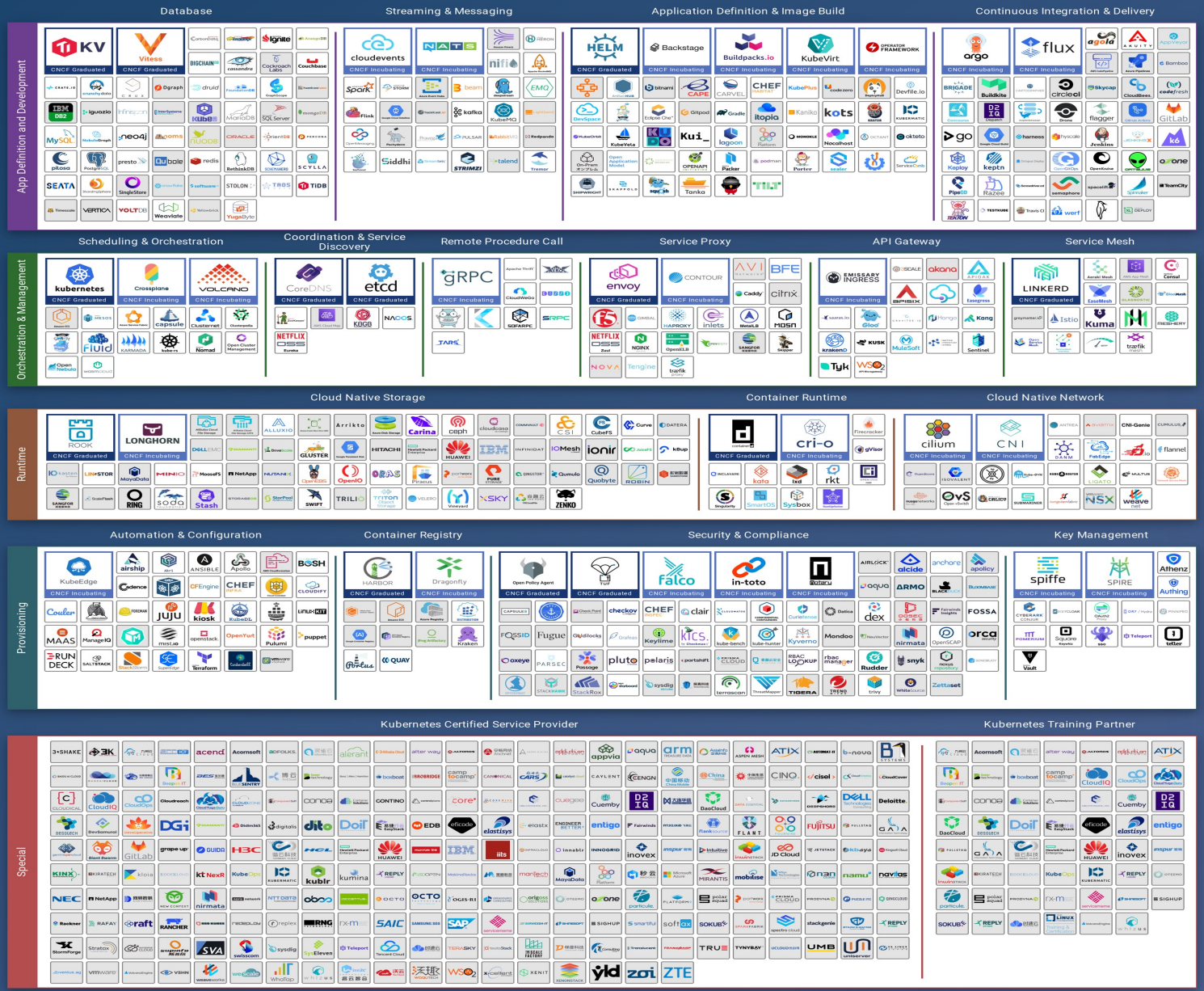


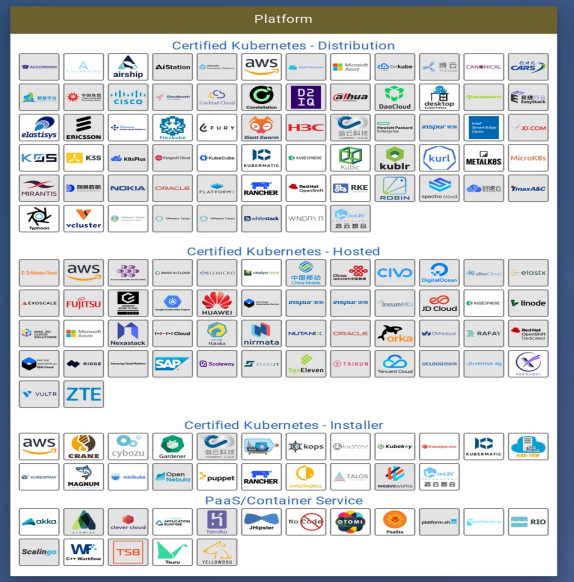
CNCF Cloud Native Landscape – January 2021

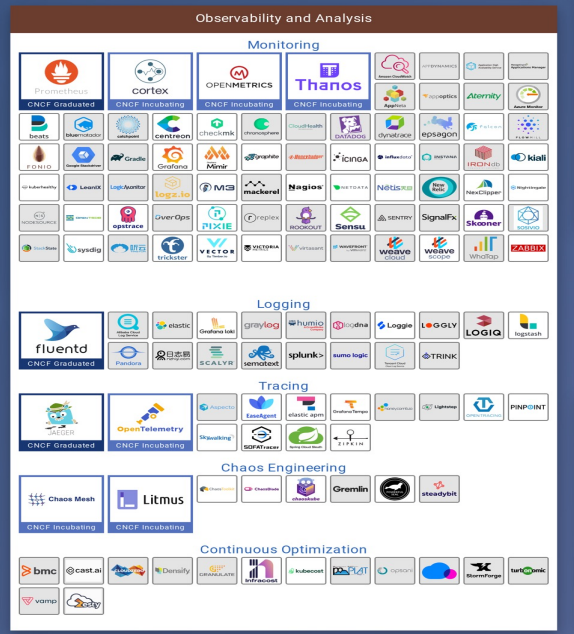
CNCF Cloud Native Landscape
2022-05-02T15:28:27Z 1b2bef25

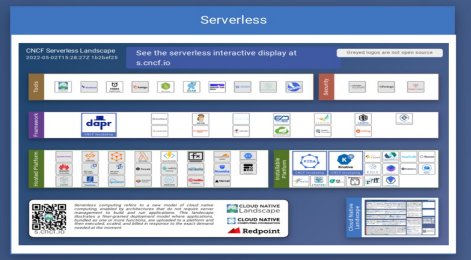
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io

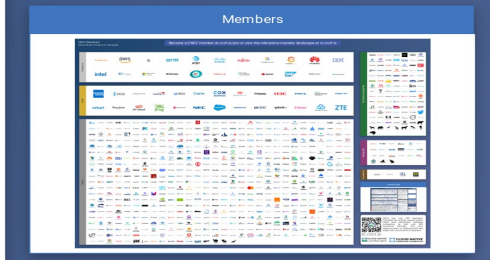
Greyed logos are not open source

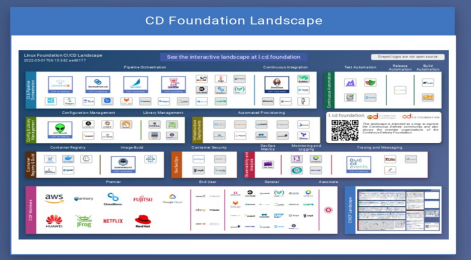












CLOUD NATIVE Landscape

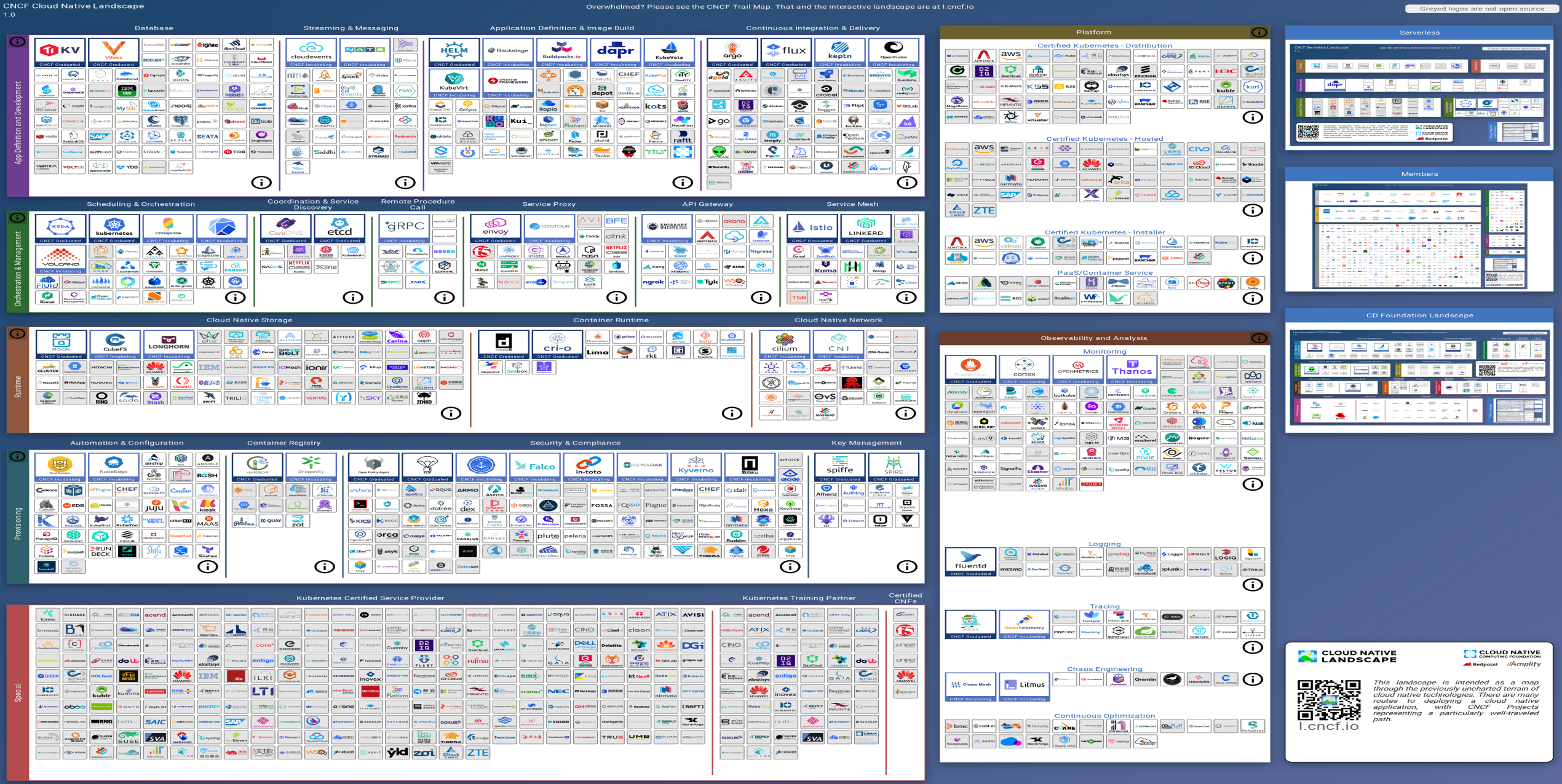
CLOUD NATIVE COMPUTING FOUNDATION

Redpoint Amplify

l.cncf.io

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.

CNCF Cloud Native Landscape – May 2022



CNCF Cloud Native Landscape – October 2023

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape *l.cncf.io* has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally

cncf.io/enduser

WHAT IS CLOUD NATIVE?

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

l.cncf.io

v20200501

1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing
- Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using GitOps paradigms such as continuous and progressive delivery and MLops

argo — CNCF Incubating

3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: cncf.io/ck
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application

kubernetes — CNCF Graduated

HELM — CNCF Graduated

4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

Prometheus — CNCF Graduated

fluentd — CNCF Graduated

OPENTRACING — CNCF Incubating

5. SERVICE PROXY, DISCOVERY, & MESH

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

envoy — CNCF Graduated

CoreDNS — CNCF Graduated

LINKERD — CNCF Incubating

6. NETWORKING, POLICY, & SECURITY

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general-purpose policy engine with uses ranging from authorization and admission control to data filtering. Falco is an anomaly detection engine for cloud native.

CNI — CNCF Incubating

Open Policy Agent — CNCF Incubating

Falco — CNCF Incubating

7. DISTRIBUTED DATABASE & STORAGE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding. Rook is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. Serving as the "brain" of Kubernetes, etcd provides a reliable way to store data across a cluster. TiKV is a high performant distributed transactional key-value store written in Rust.

Vitess — CNCF Graduated

ROOK — CNCF Incubating

etcd — CNCF Incubating

TiKV — CNCF Incubating

8. STREAMING & MESSAGING

When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues. CloudEvents is a specification for describing event data in common ways.

gRPC — CNCF Incubating

NATS — CNCF Incubating

cloudevents — CNCF Incubating

9. CONTAINER REGISTRY & RUNTIME

Harbor is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.

containerd — CNCF Graduated

HARBOR — CNCF Incubating

cri-o — CNCF Incubating

10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

TUF — CNCF Graduated

Notary — CNCF Incubating

Kubernetes and OpenShift core concepts explored in-depth

Red Hat Enterprise Linux CoreOS

Immutable Operating System

Red Hat Enterprise Linux CoreOS is versioned with OpenShift

CoreOS is tested and shipped in conjunction with the platform. Red Hat runs thousands of tests against these configurations.

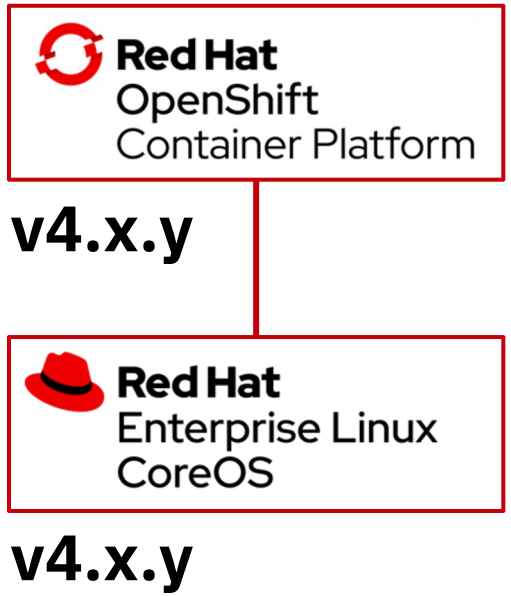
Red Hat Enterprise Linux CoreOS is managed by the cluster

The Operating system is operated as part of the cluster, with the config for components managed by Machine Config Operator:

- CRI-O config
- Kubelet config
- Authorized registries
- SSH config

RHEL CoreOS admins are responsible for:

Nothing. 😀🙌



More about CoreOS

cri-o

Lightweight Container Runtime for Kubernetes

Minimal and Secure Architecture	Designed for and optimized for Kubernetes	Runs any OCI-compliant image (including docker)

A lightweight, OCI-compliant container runtime

CRI-O tracks and versions identical to Kubernetes, simplifying support permutations

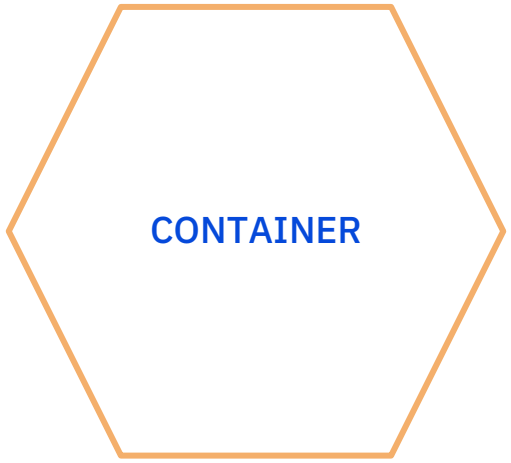
CRI-O Support in OpenShift

CRI-O 1.16  Kubernetes 1.16  OpenShift 4.6 

CRI-O 1.26  Kubernetes 1.26  OpenShift 4.13 

CRI-O 1.27  Kubernetes 1.27  OpenShift 4.14 

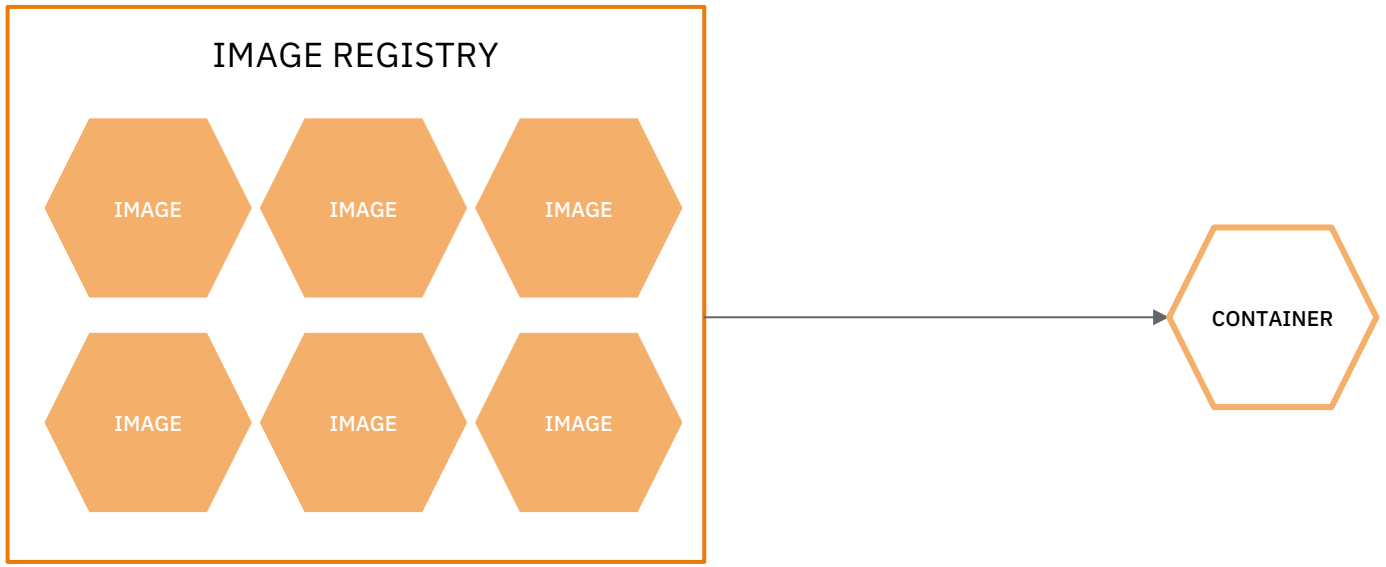
Broad ecosystem of workloads



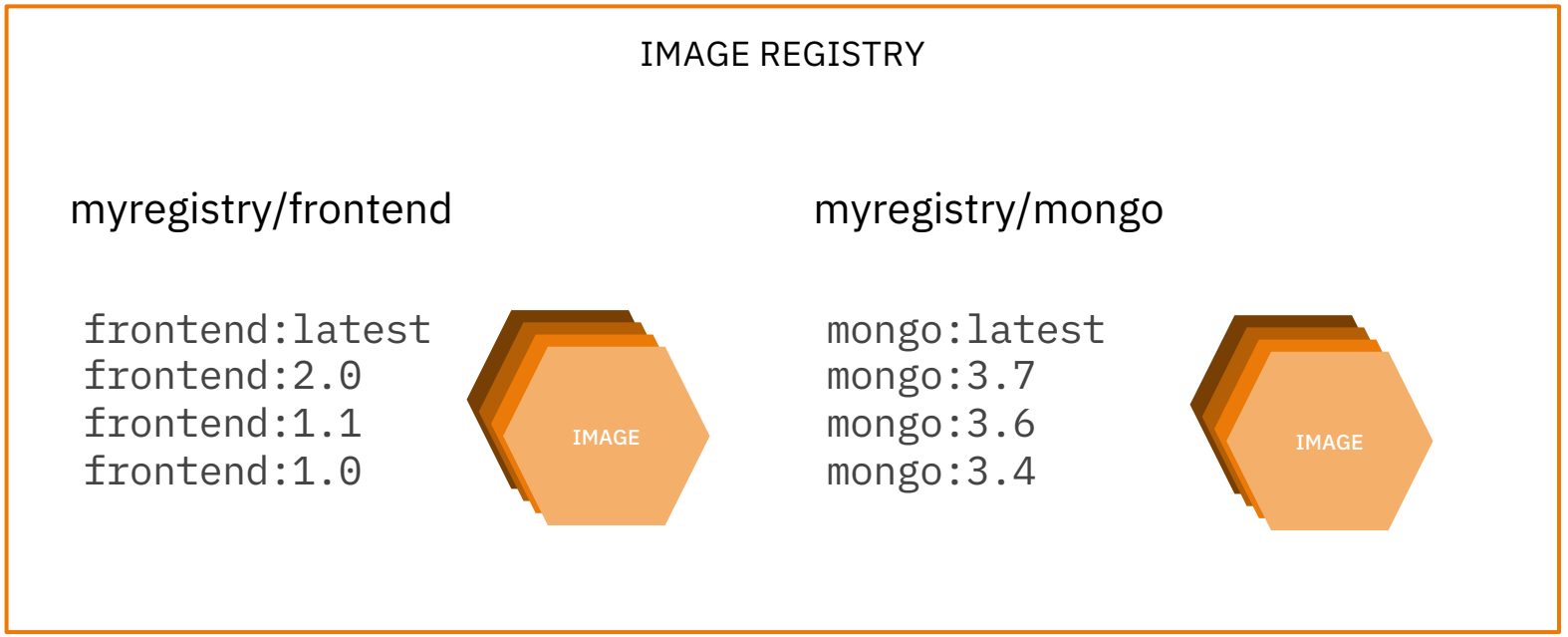
A container is the smallest compute unit

containers are created from container images

container images are stored in an image registry

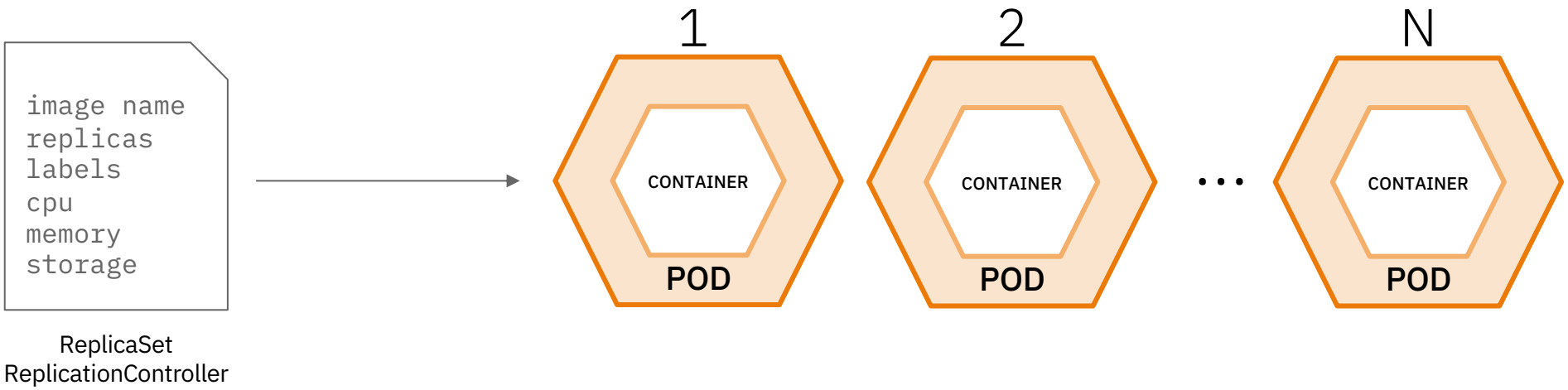


an image repository contains all versions of an image in the image registry

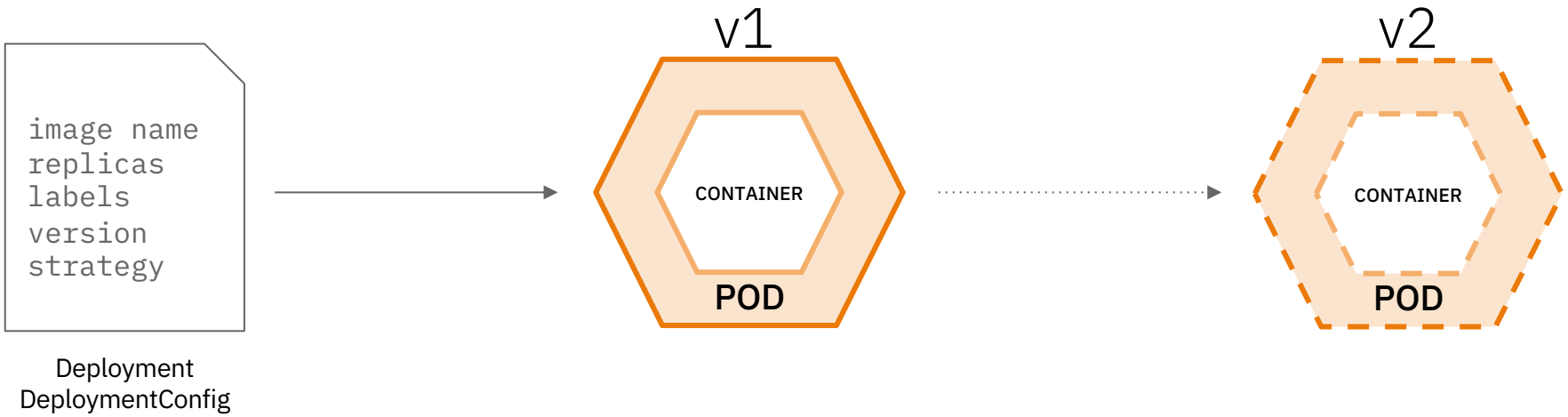


containers are wrapped in pods which are units of deployment and management

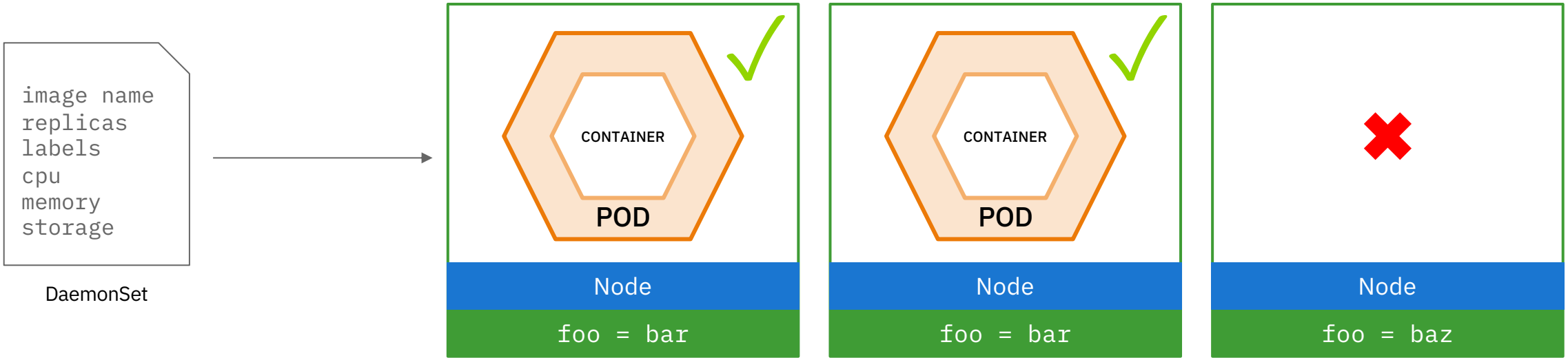
ReplicationControllers & ReplicaSets ensure a specified number of pods are running at any given time



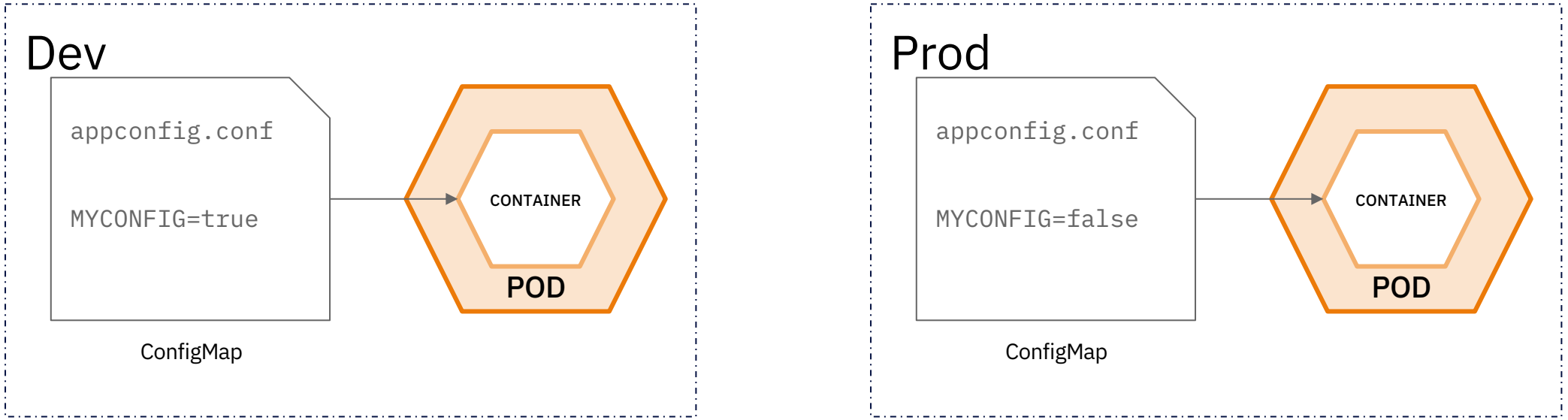
Deployments and DeploymentConfigurations define how to roll out new versions of Pods



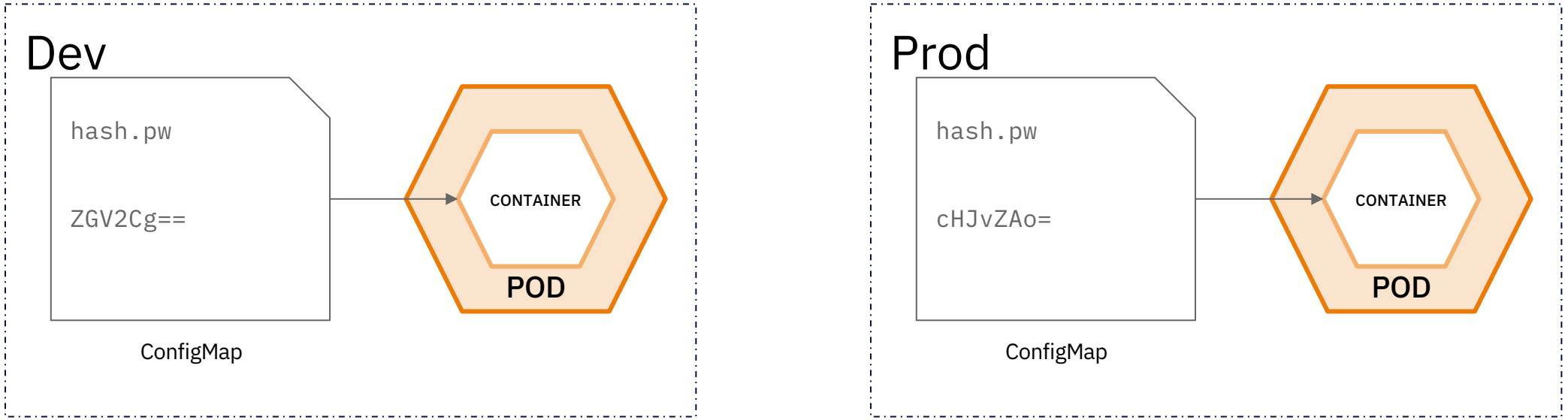
a `daemonset` ensures that all (or some) nodes run a copy of a pod

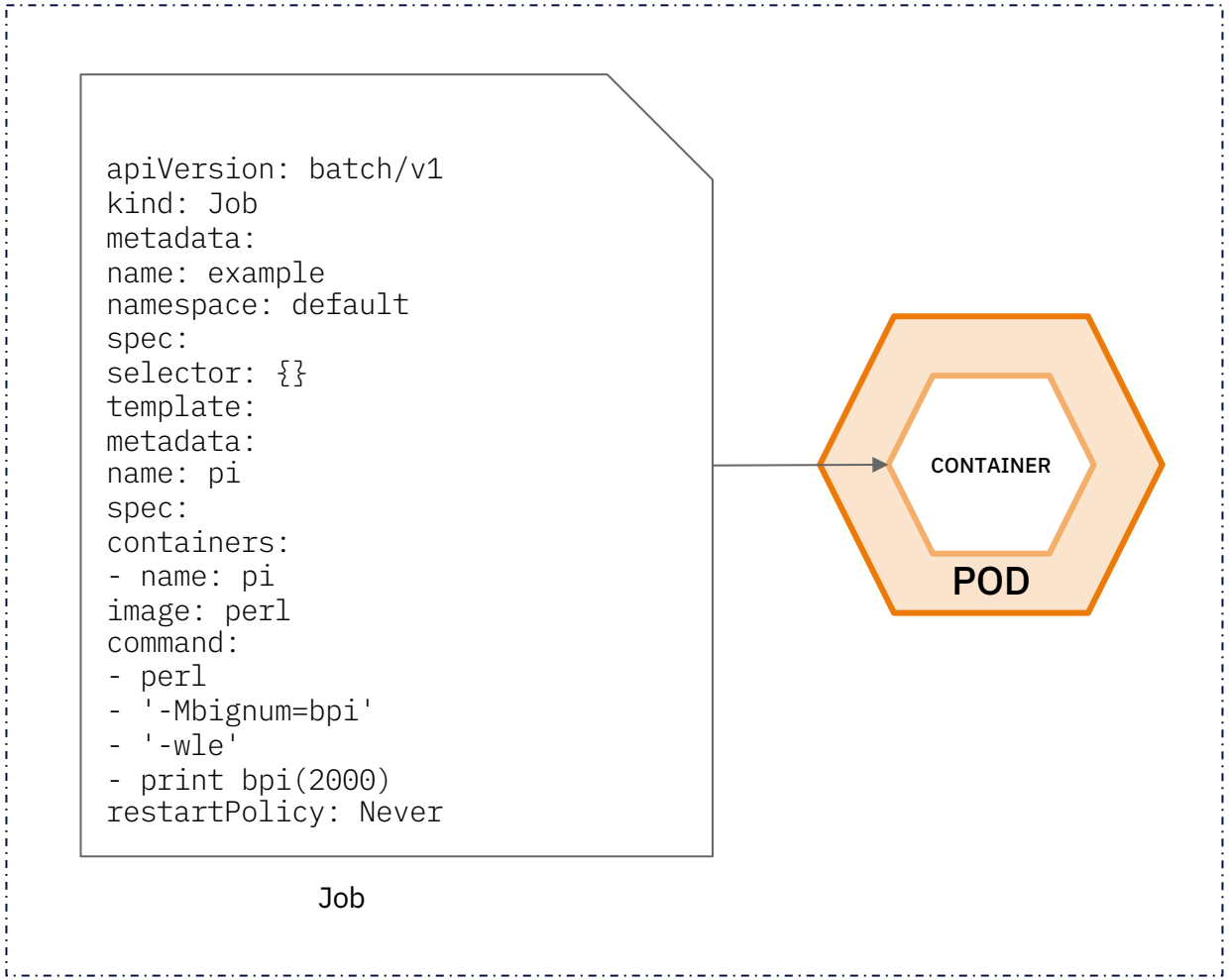


`configmaps` allow you to decouple configuration artifacts from image content

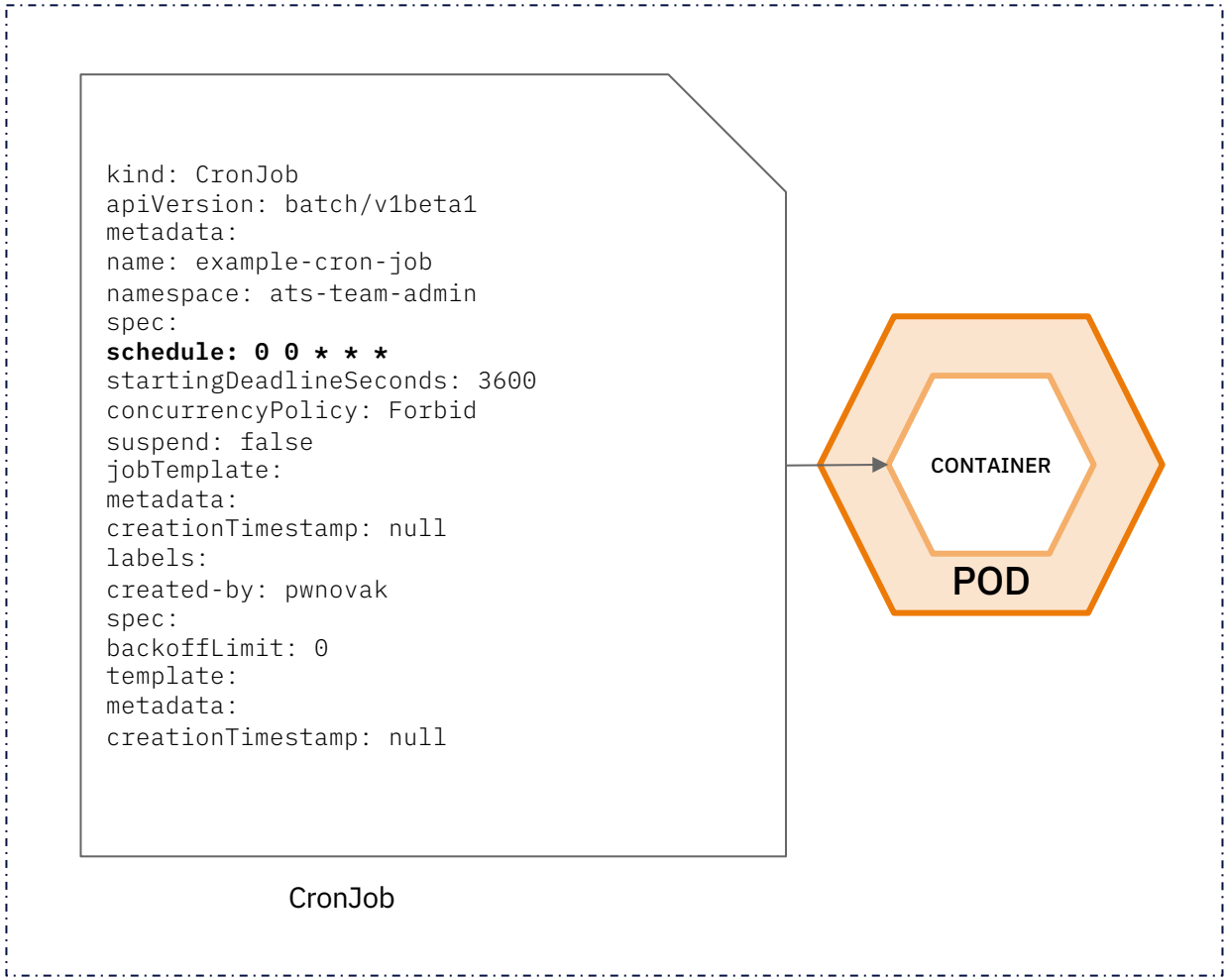


`secrets` provide a mechanism to hold sensitive information such as passwords





jobs are batch tasks that can be run either manually or via the cluster crontab.

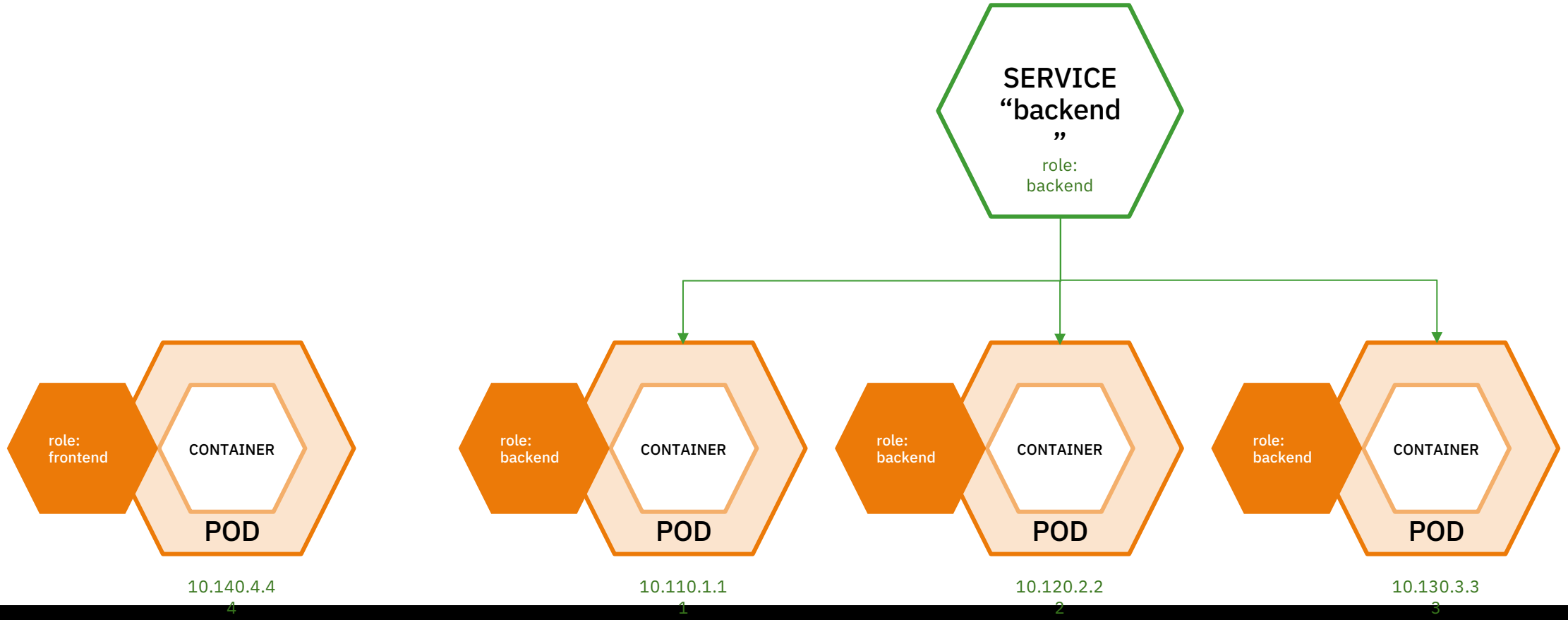


cronjobs are batch tasks run on a defined schedule via the cluster crontab.

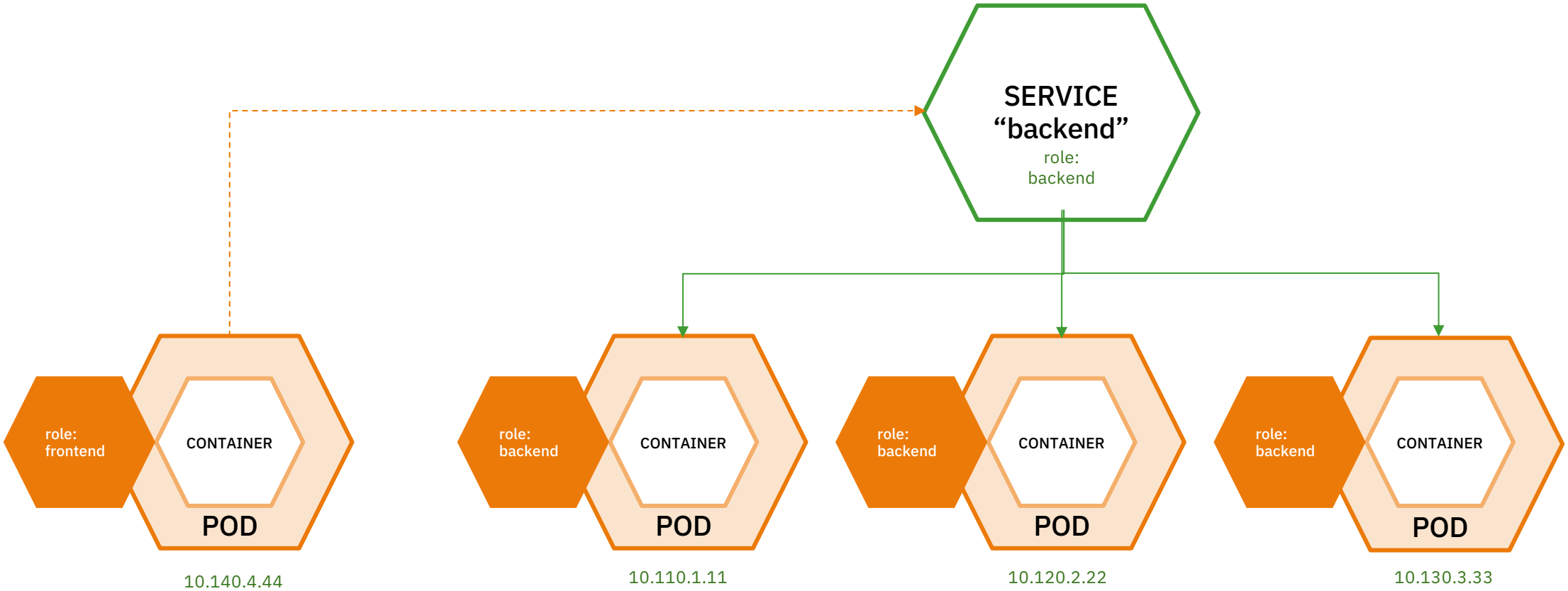
Tip: You MUST stagger your scheduling!

Cron Jobs

services provide internal load-balancing and service discovery across pods

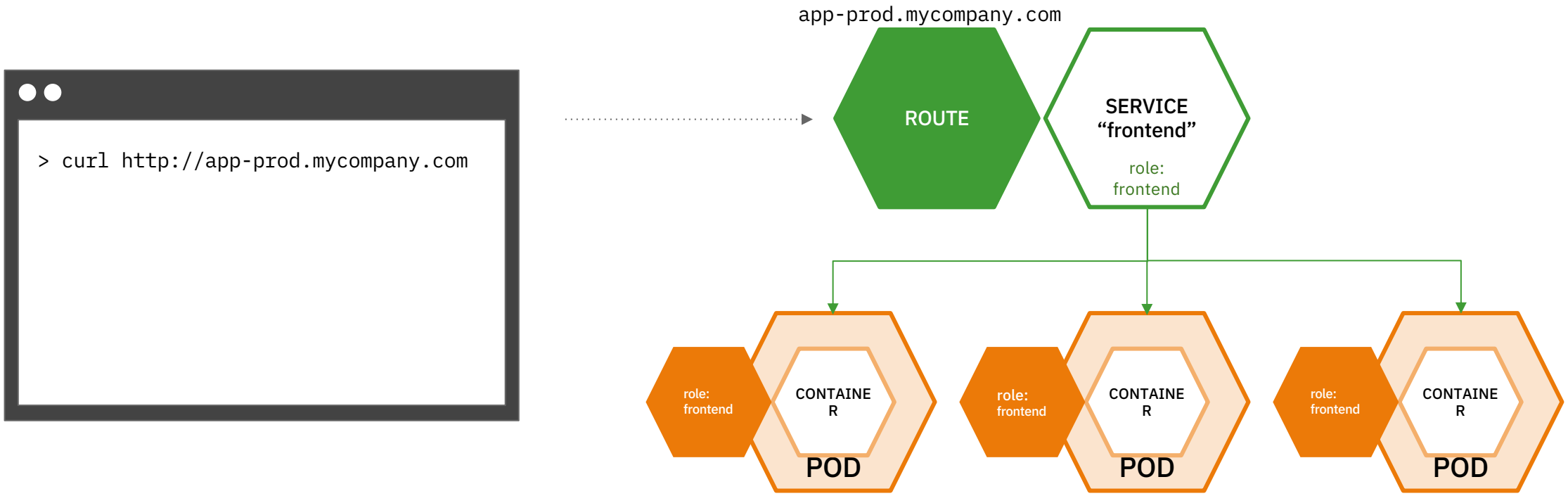


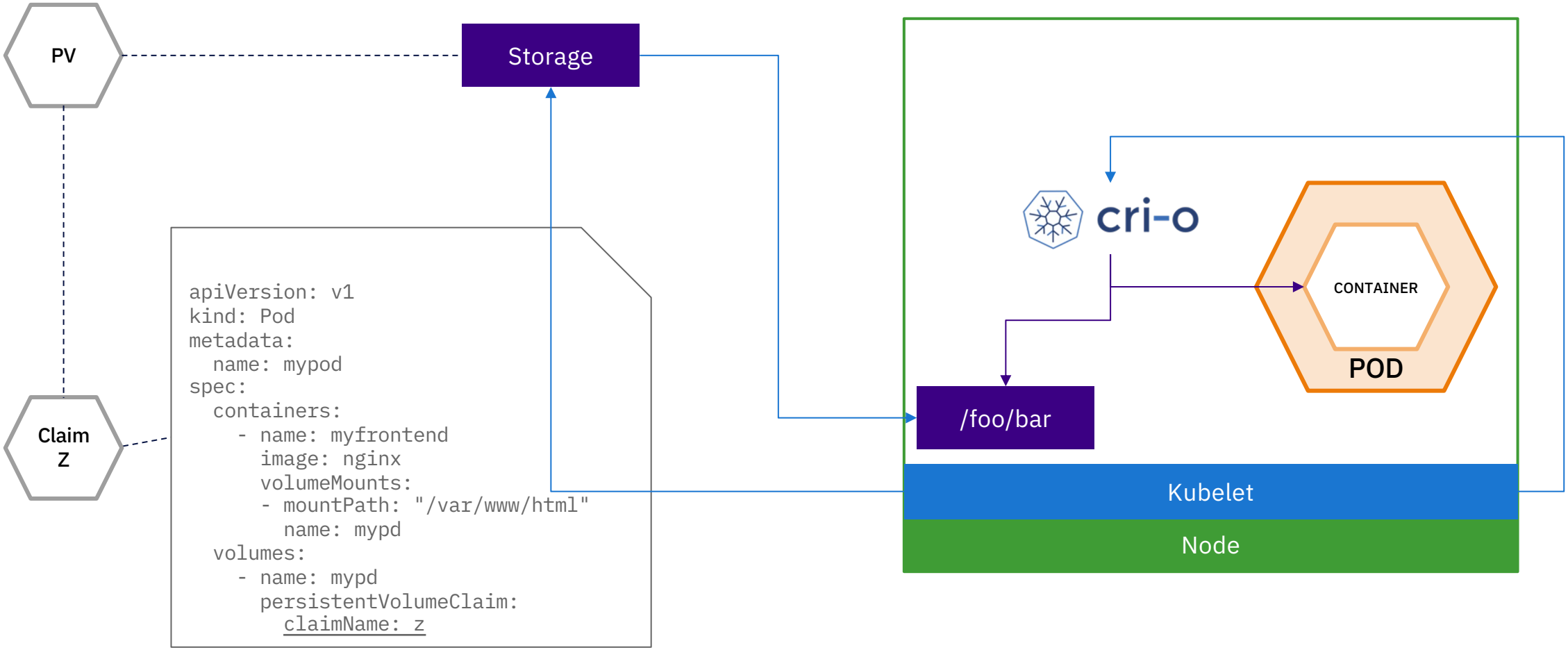
apps can talk to each other via services



Service chatter

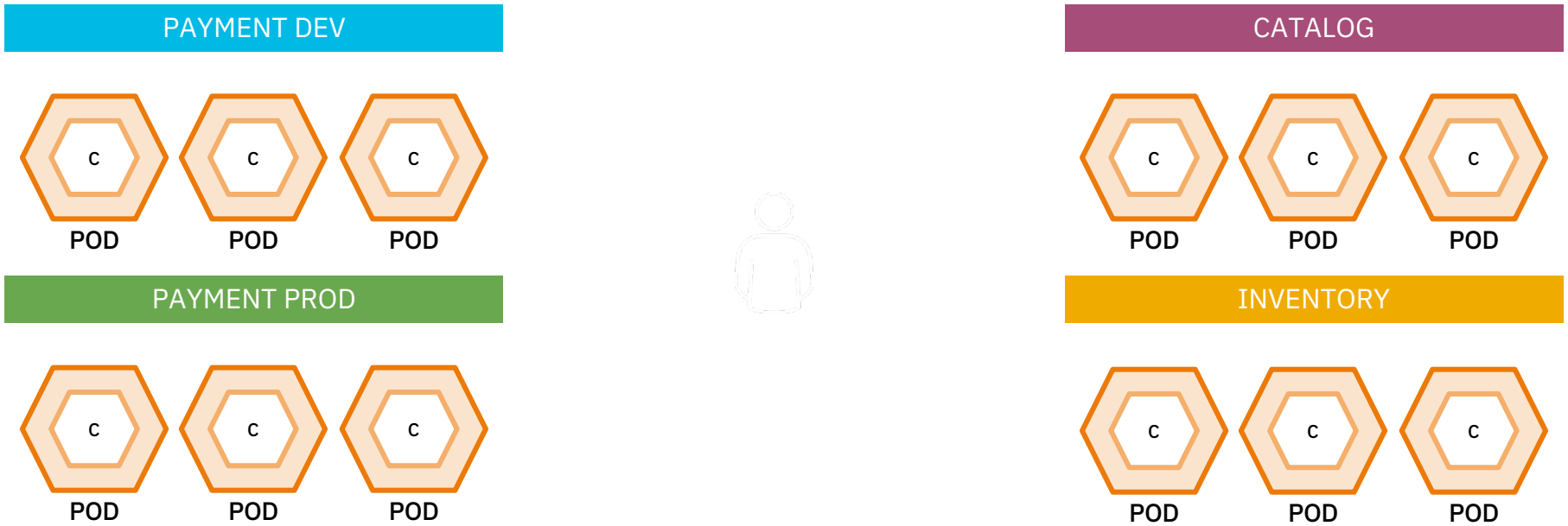
`routes` make services accessible to clients outside the environment via real-world URLs





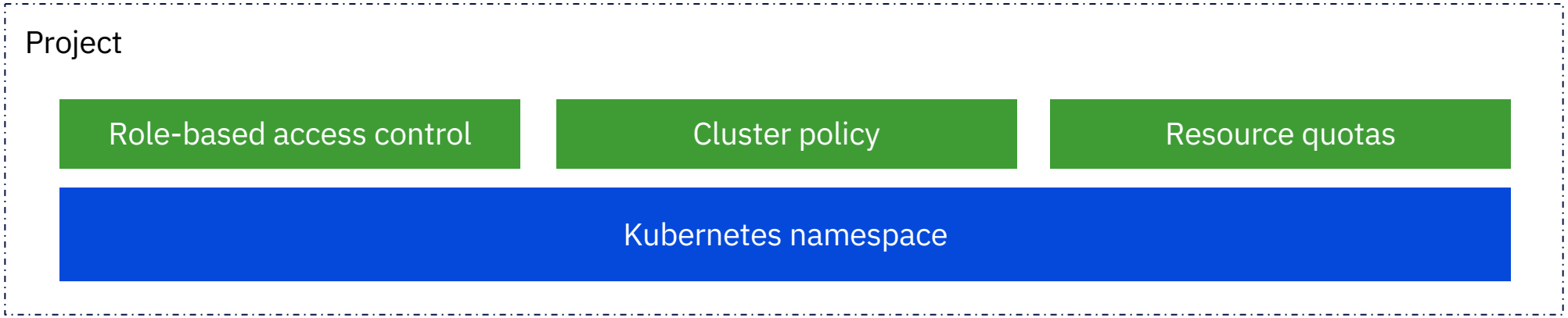
PV Consumption

Namespaces collate resources and isolate apps across environments, teams, groups and departments.

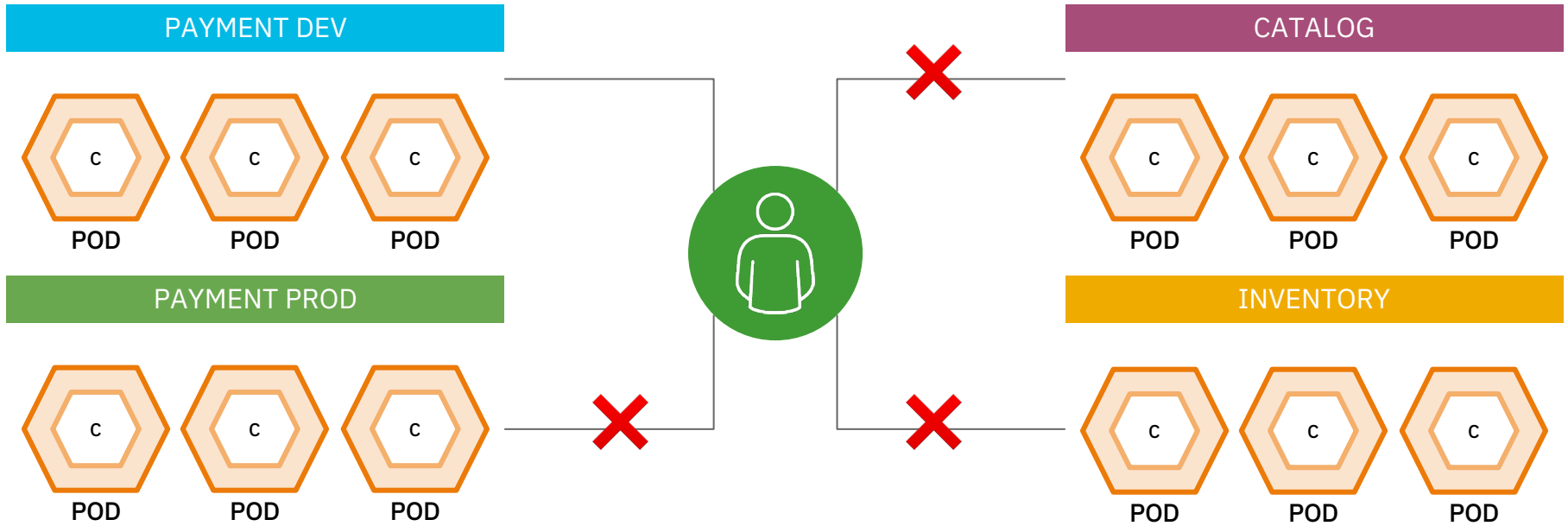


Kubernetes native namespaces

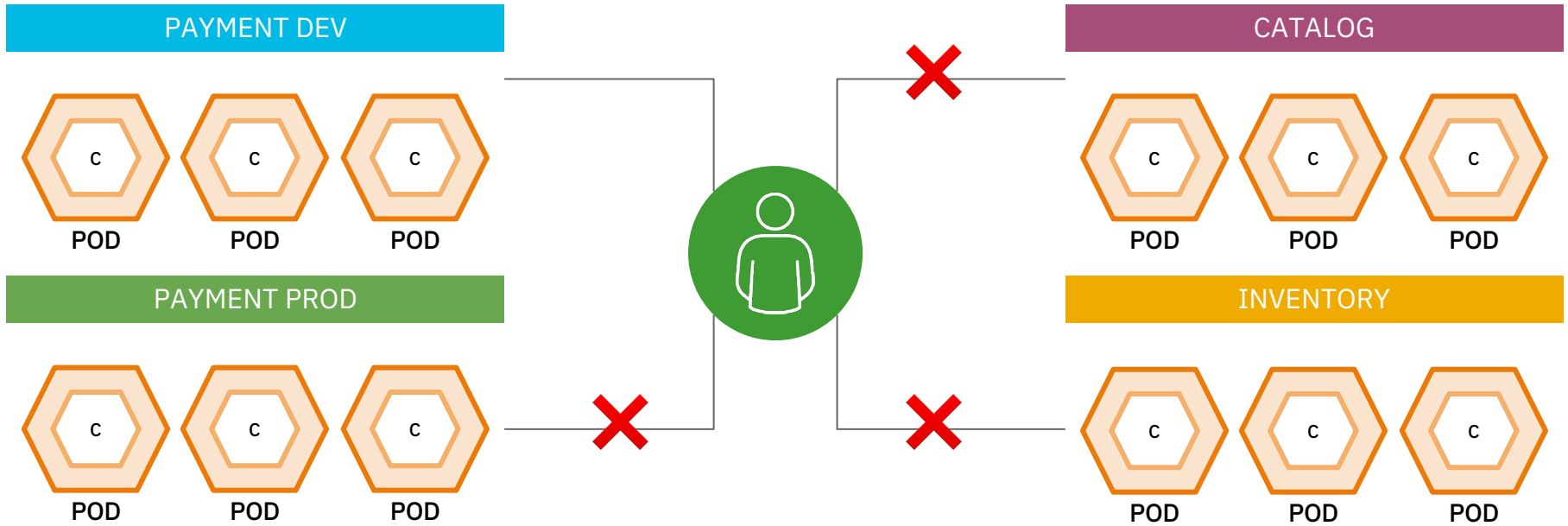
A Kubernetes native namespace plus the RBAC layer and some other OpenShift-specific enhancements is a project



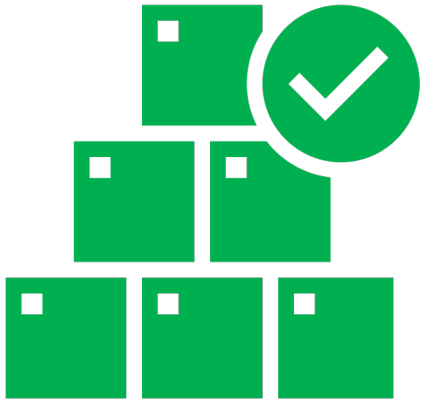
Projects provide isolation and proper security boundaries for applications across environments, teams, groups, departments, etc.

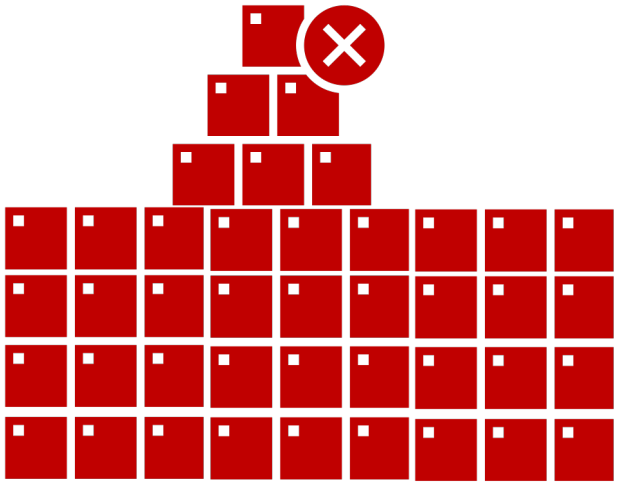


IBM Z and LinuxONE are the **only** platform where SECURE multi-tenant usage is possible



Embrace projects and use them on a sensible scale. Balance their performance enhancement against operational complexity.





OpenShift 4 Architecture

COMPUTE

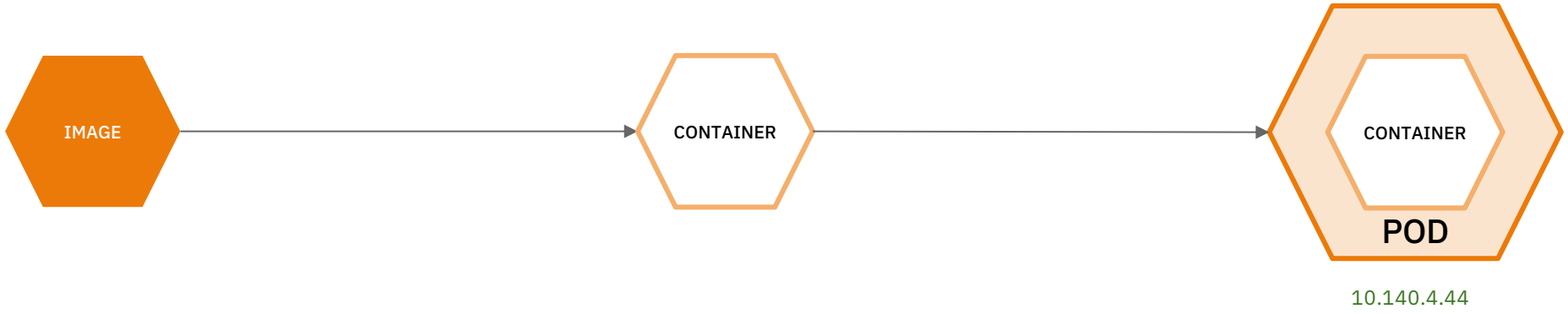
NETWORK

STORAGE

your choice of infrastructure

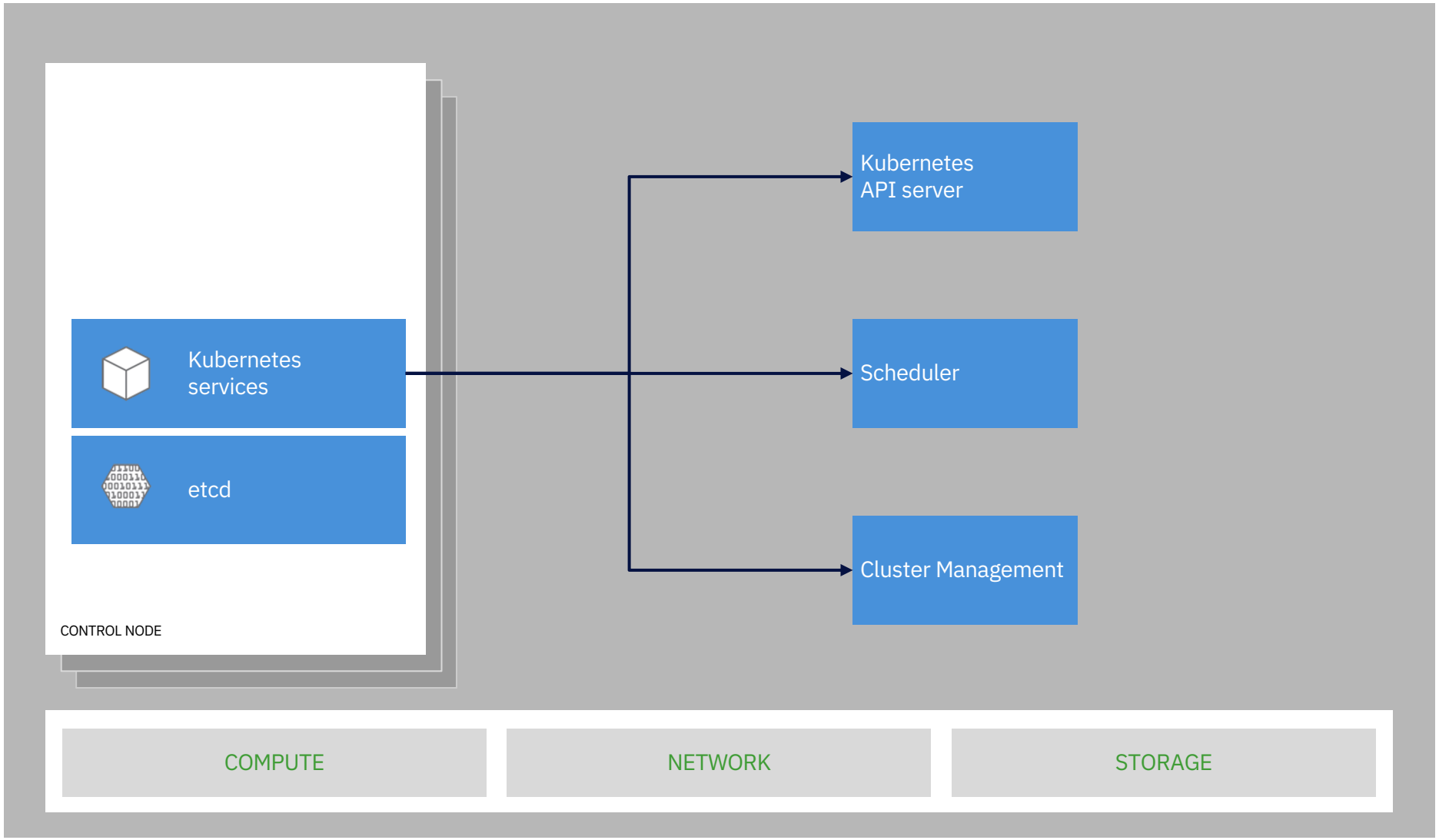
Compute nodes run workloads

Control nodes

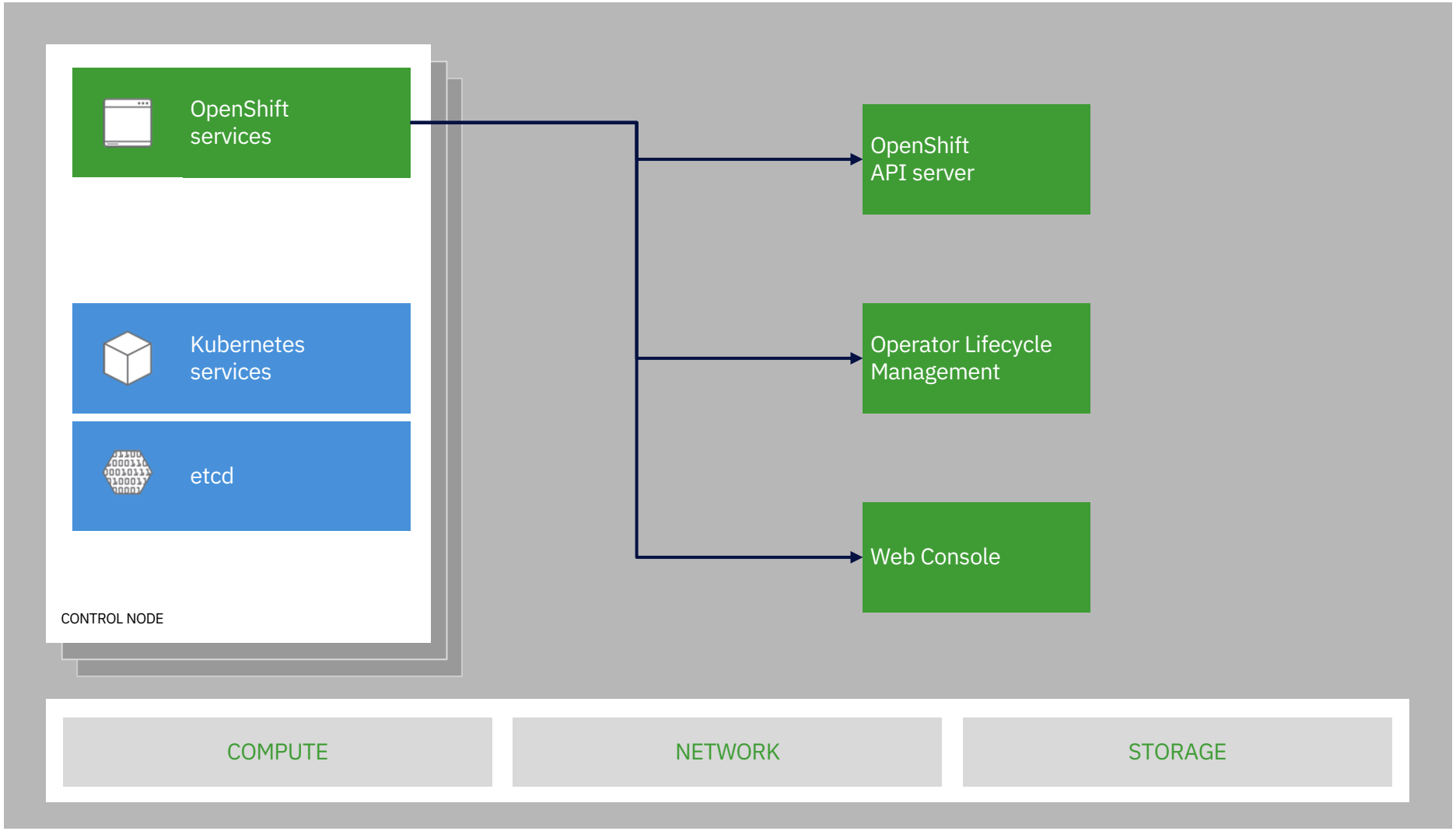


everything runs in pods

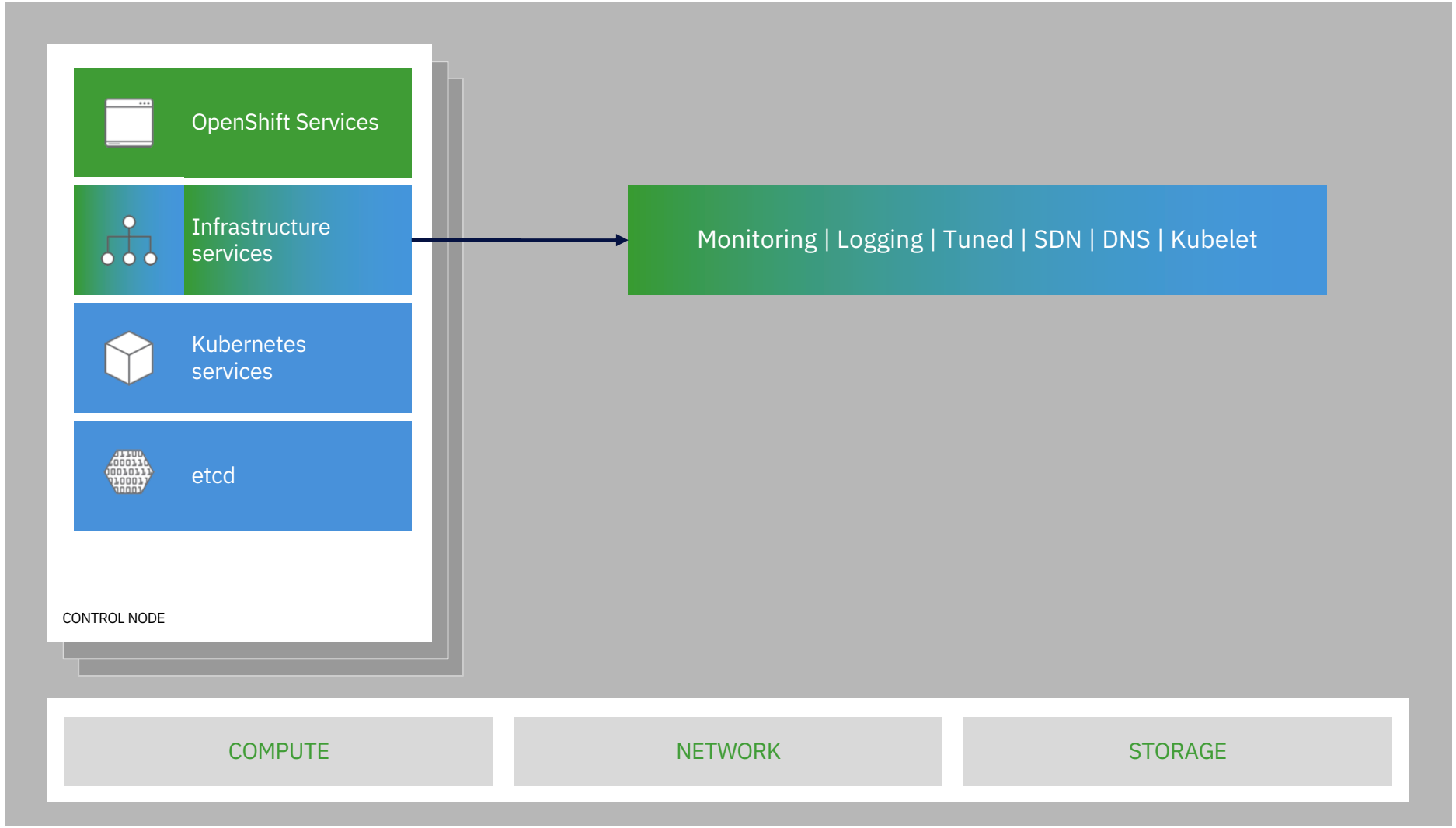

etcd
CONTROL NODE
COMPUTE
NETWORK
STORAGE



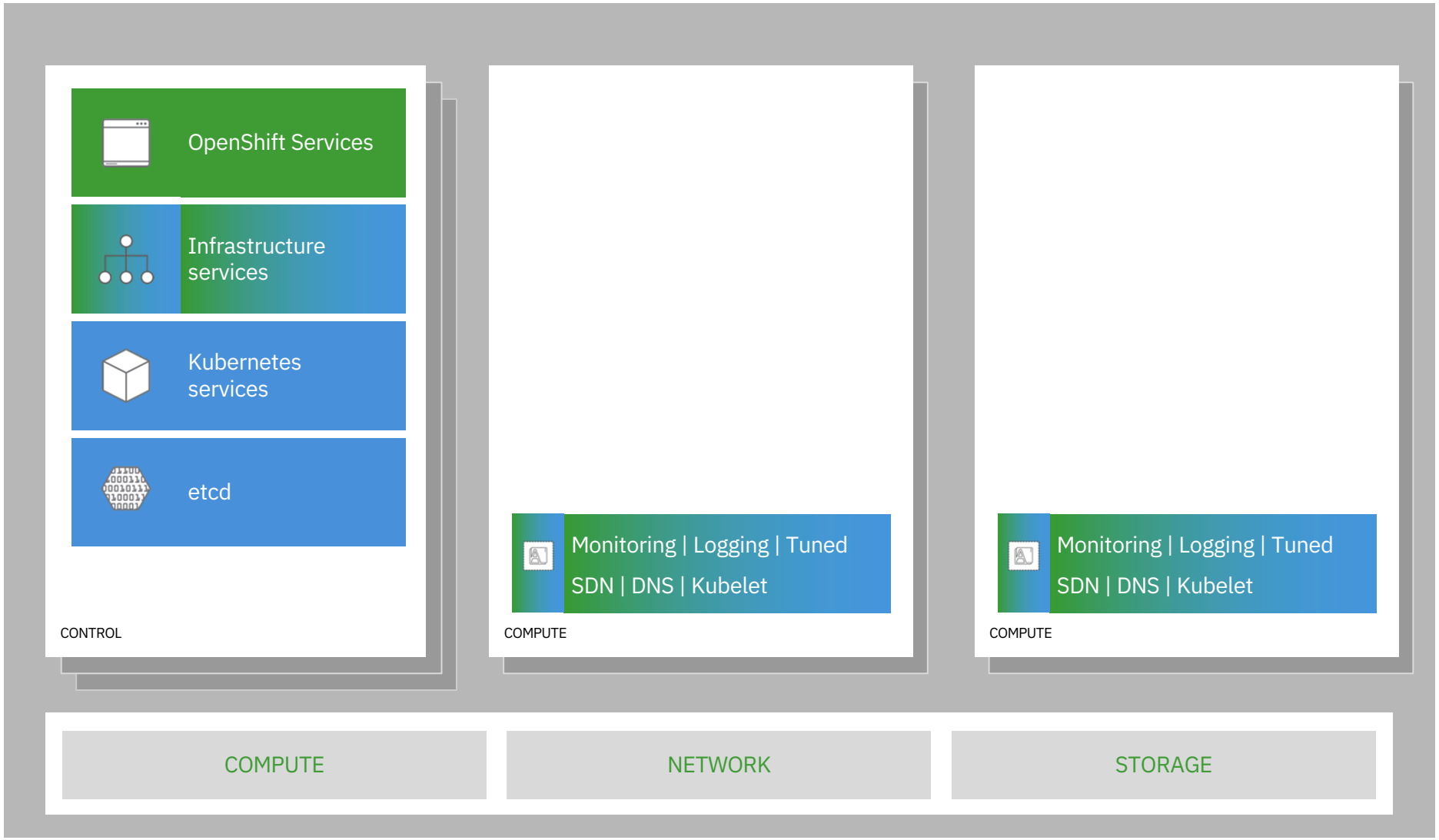
core kubernetes components



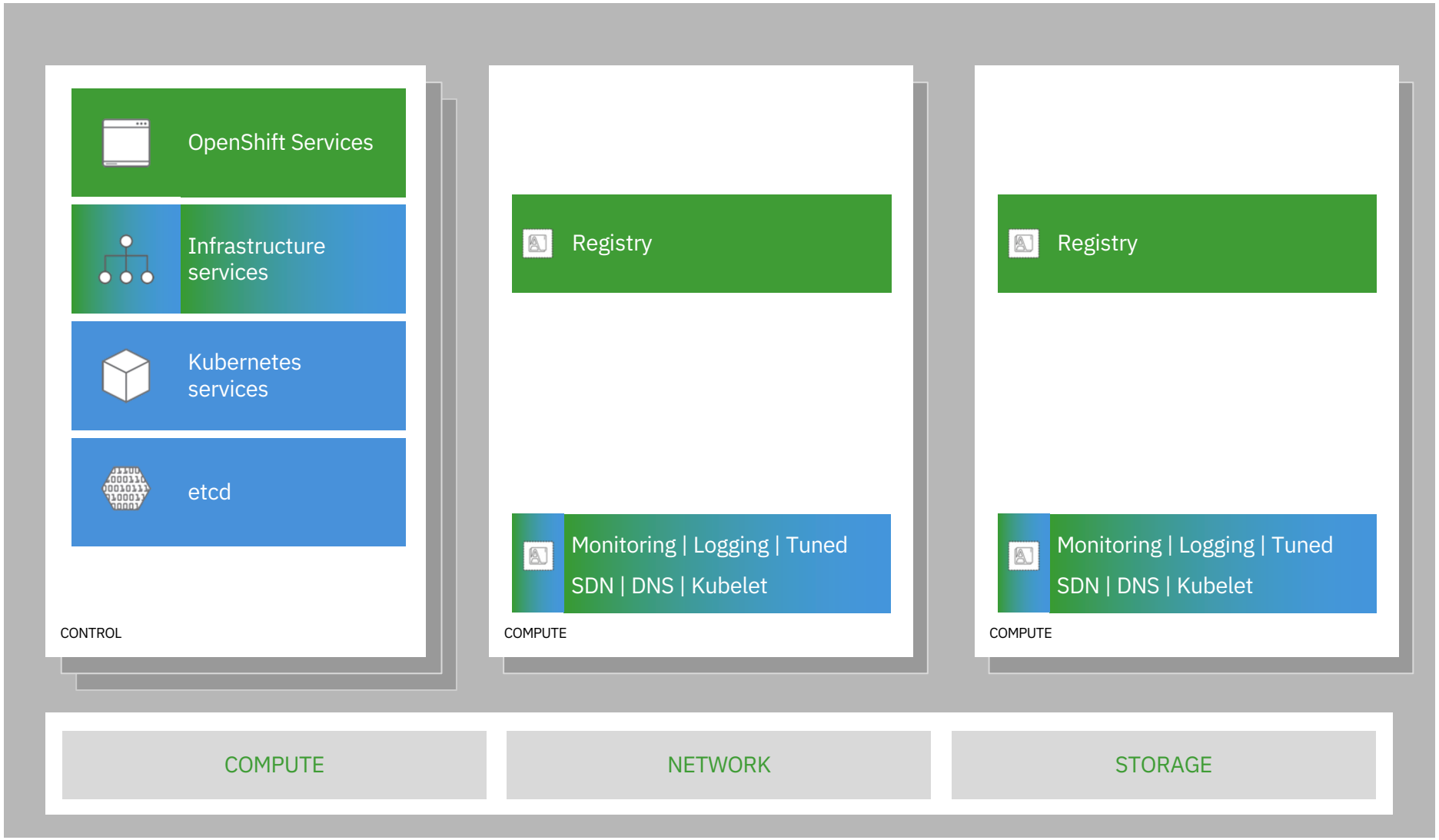
core OpenShift components



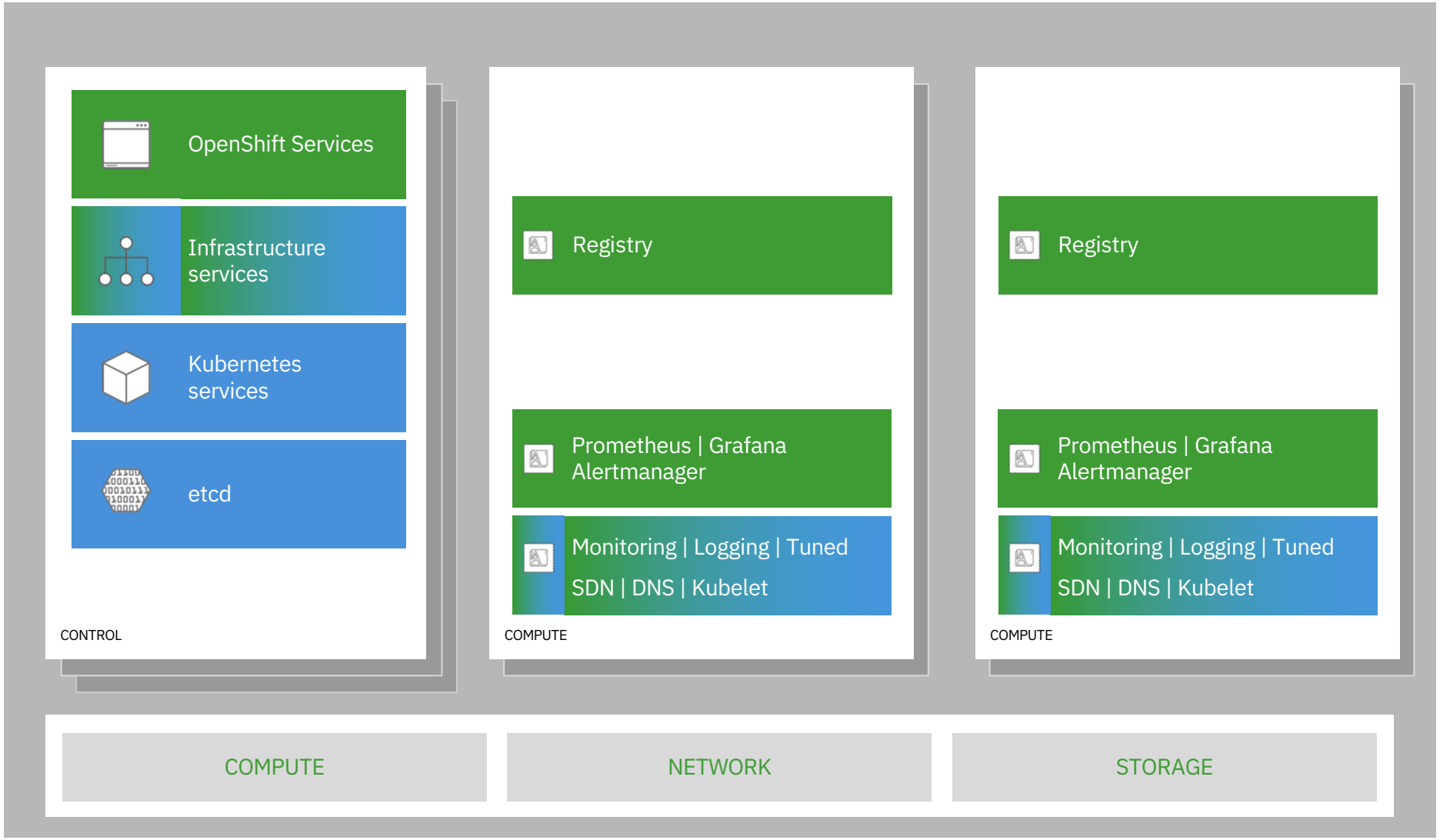
internal and support infrastructure services



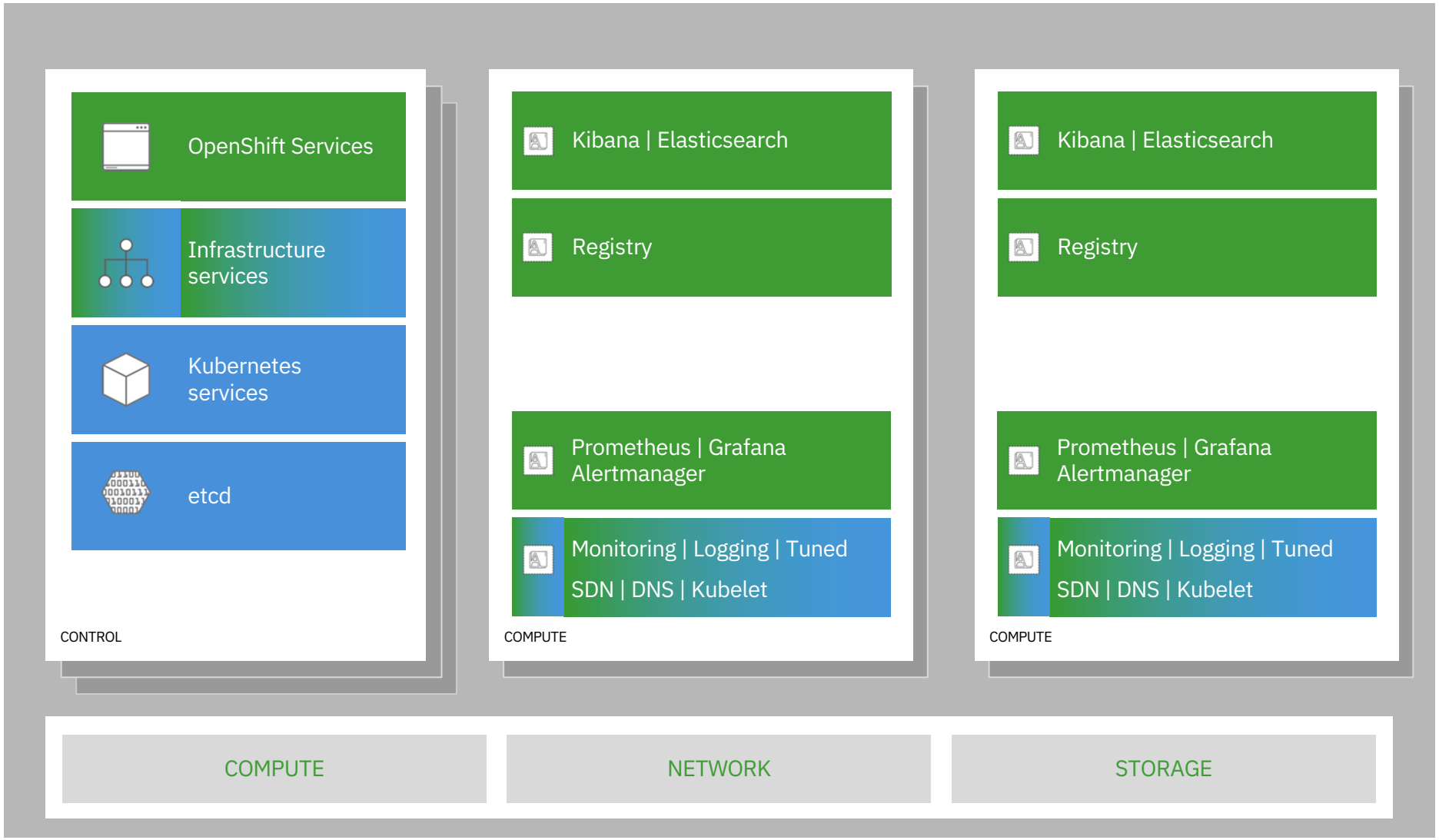
run on all hosts



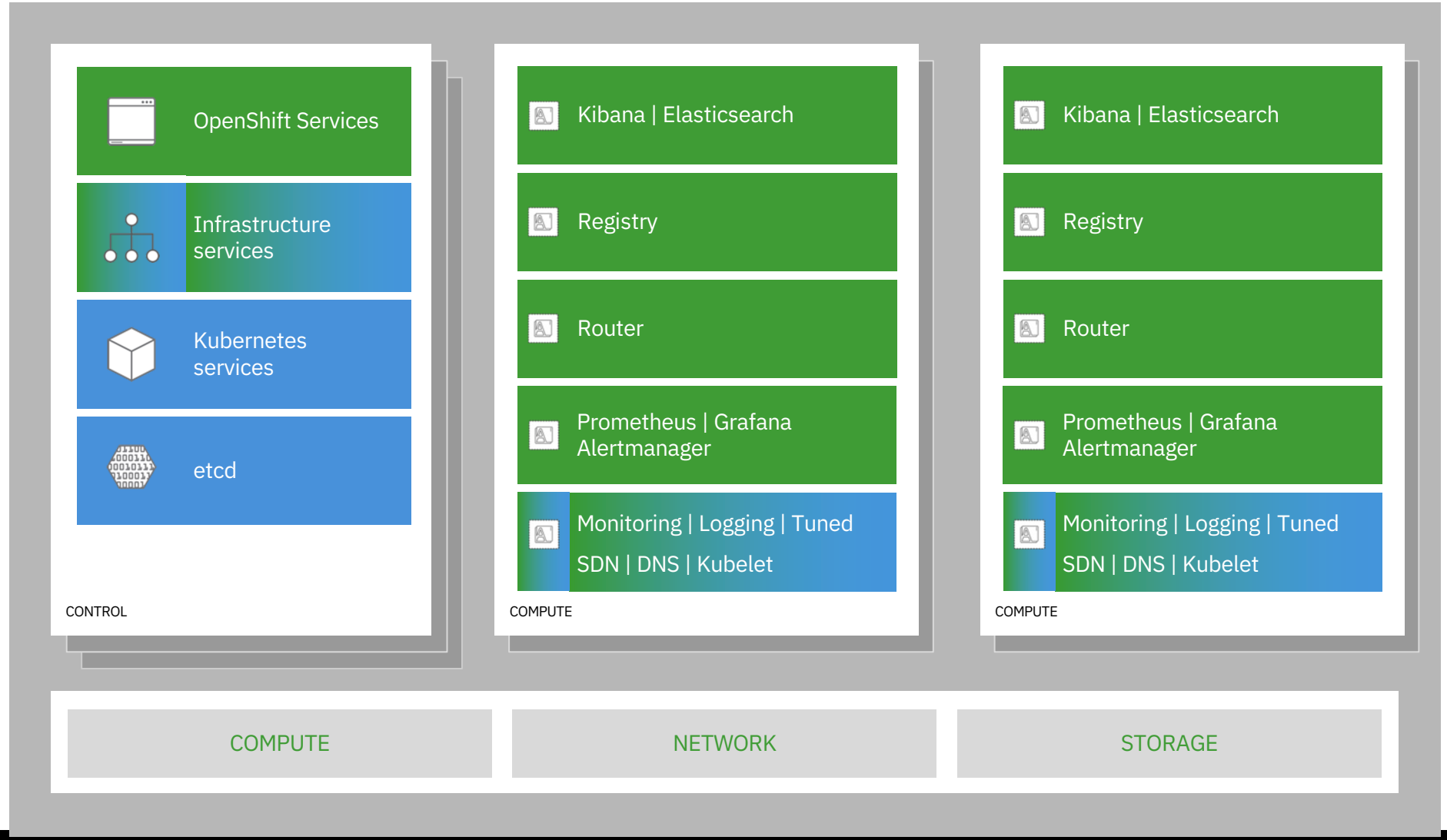
integrated image registry



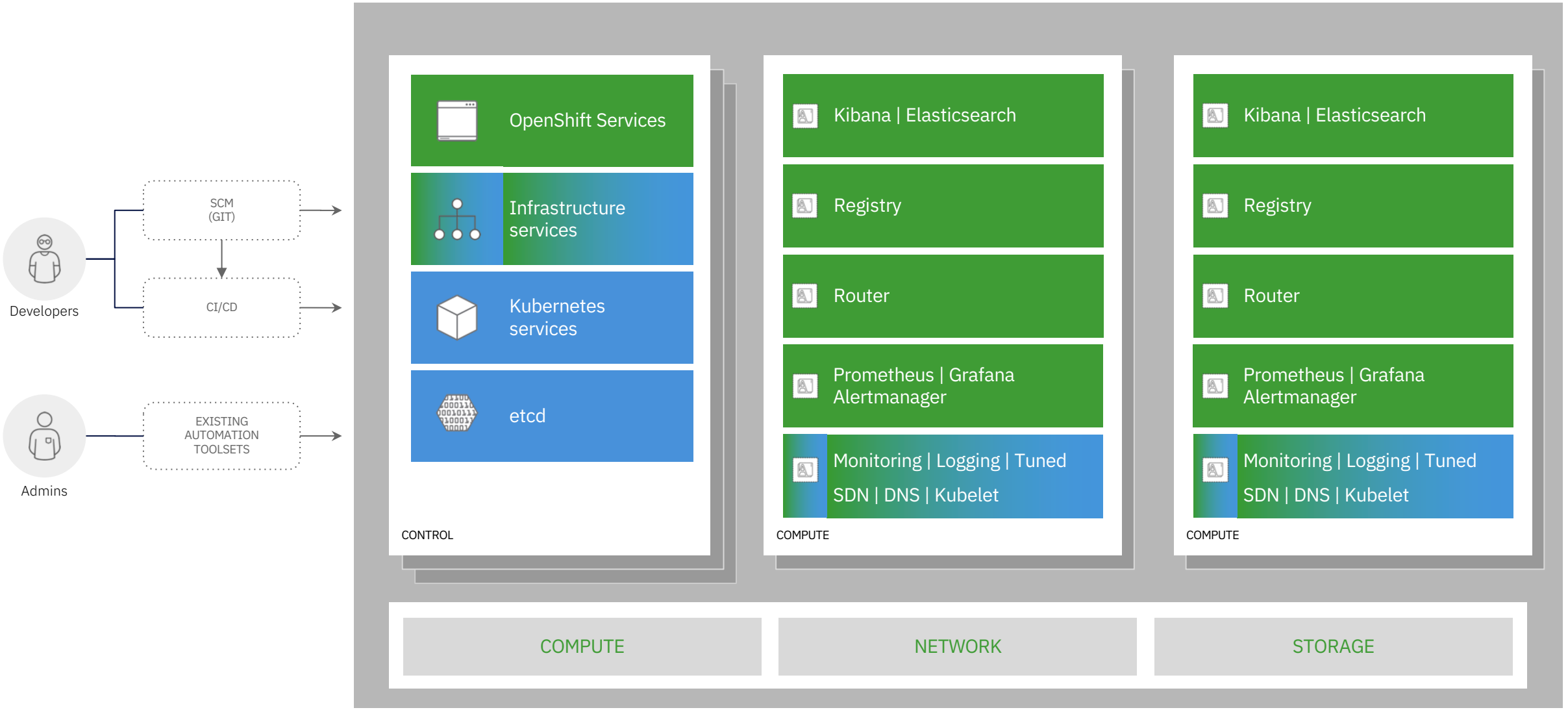
cluster monitoring



log aggregation



integrated routing



dev and ops via web, cli, API, and IDE

Creating multi-architecture deployments is good practice and should be considered mandatory for your container journey.

Fit for purpose is a fundamental criterion

What should (or shouldn’t) I think about containerizing?

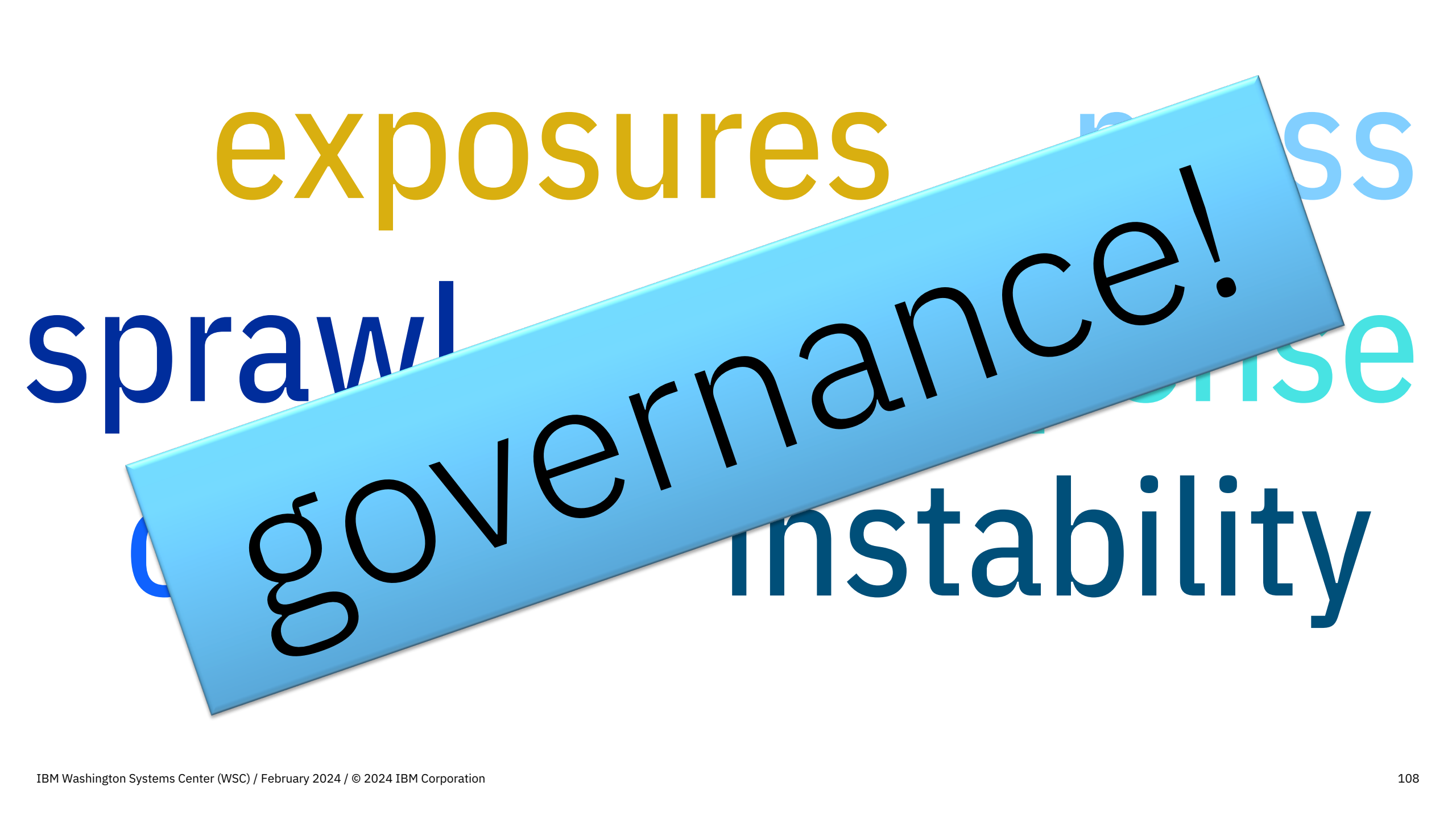
Web middleware / J2EE

Messaging and integration such as Kafka / EventStreams

HTTP content

Anything that needs to be able to rapidly scale up to handle a burst in demand, and then gracefully scale back down to a steady state after the increased demand has subsided.

- Relational databases and other types of warehouses are exceptionally **unlikely** candidates
- Putting everything into containers because all the cool kids are doing it is a terrible plan.
- Moving monolithic applications into containers and saying that you've begun a transformation into microservices and containers is as truthful as stating that you own the Brooklyn bridge.
- "Lift and shift" is and will always be a recipe for unnecessary grief and instability


exposures
sprawl
instability
governance!

Foundational governance is key to your success.

IBM has discovered that the surest path to container and microservices sprawl is to not have sound DevOps processes in place before adopting them.

As IT environments scale, thanks to the rise of containers and microservices, having mature processes in place to manage dynamic IT environments will be critical.

Most IT organizations today still don't have many mature DevOps processes!

What they do have in place was never really designed to address rapid changes to code enabled by microservices and containers.

Rise of microservices and containers is creating one of those seminal moments where organizations need to decide what role they want their internal IT operations teams to play.

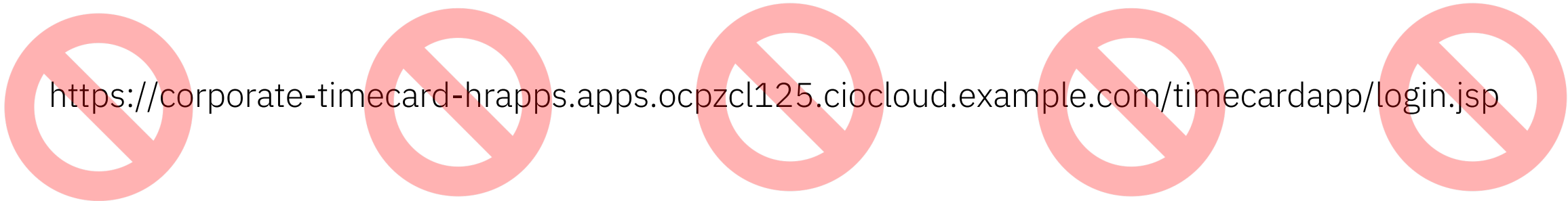
The issue facing IT organizations now is how much do they want to take care of that problem today versus waiting for an outcome that, at this point, is all but inevitable.

user experience must never be an afterthought!

If you are planning to expose any of the *https://<<application name>>.apps.<<clustername>>.<<domain>>* URLs to your end-users, you are going about this incorrectly.

If you are planning to deploy applications without governance to ensure they use a unique URI path, you are going about this incorrectly.

Especially if you are thinking about using the server root! Please don't!

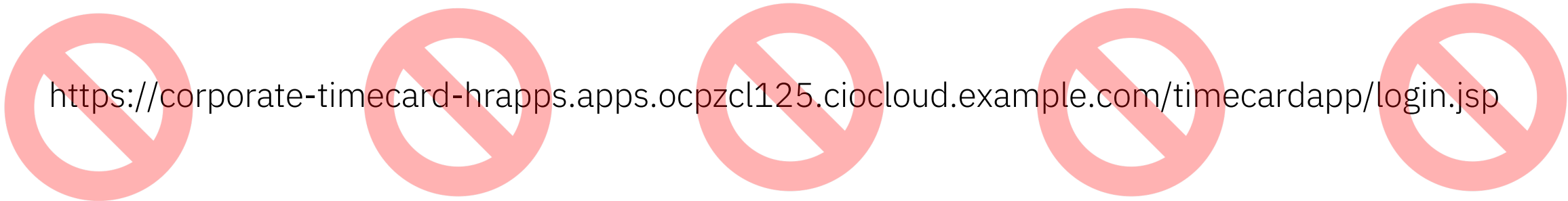


OCP for Web applications belongs behind a proxy

If you are planning to expose any of the *https://<<application name>>.apps.<<clustername>>.<<domain>>* URLs to your end-users, you are going about this incorrectly.

If you are planning to deploy applications without governance to ensure they use a unique URI path, you are going about this incorrectly.

Especially if you are thinking about using the server root! Please don't!



OCP for Web applications belongs behind a proxy

The main prerequisite: Thoughtful planning

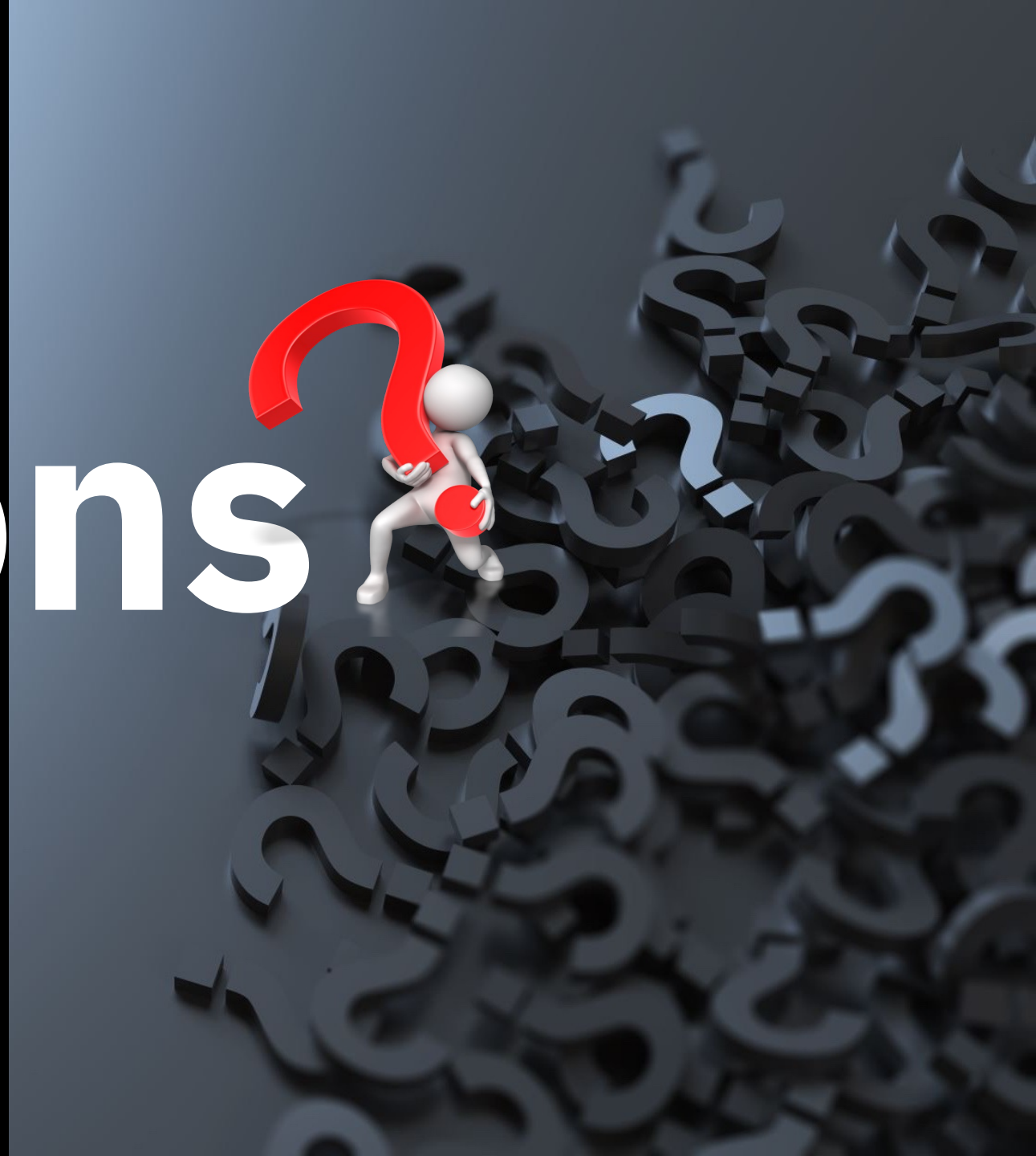
- Software architecture where a single software instance can serve multiple, distinct user groups.
- Software-as-a-service (SaaS) offerings are an example of multitenant architecture.
- In cloud computing, multitenancy can also refer to shared hosting, in which server resources are divided among different customers.
- Multitenancy is the opposite of single tenancy, when a software instance or computer system has one end-user or group of users.

When referring to a container orchestration platform such as Kubernetes, the term multitenancy usually means *a single cluster that serves multiple projects*. The cluster is configured so each project runs with some degree of isolation from the others.

- When using Kubernetes for container orchestration, it’s possible to set up multitenant environments using a single Kubernetes cluster.
- Separate each tenant into their own namespace
- Create policies that enforce tenant isolation.
- There are benefits and risks associated with this which need to be considered as part of the decision-making process.

Multitenant security is essential for enterprise-scale use of Kubernetes. Multitenancy allows you to have different teams use the same cluster while preventing unauthorized access to each other’s environments.

Questions



Operations and infrastructure deep dive

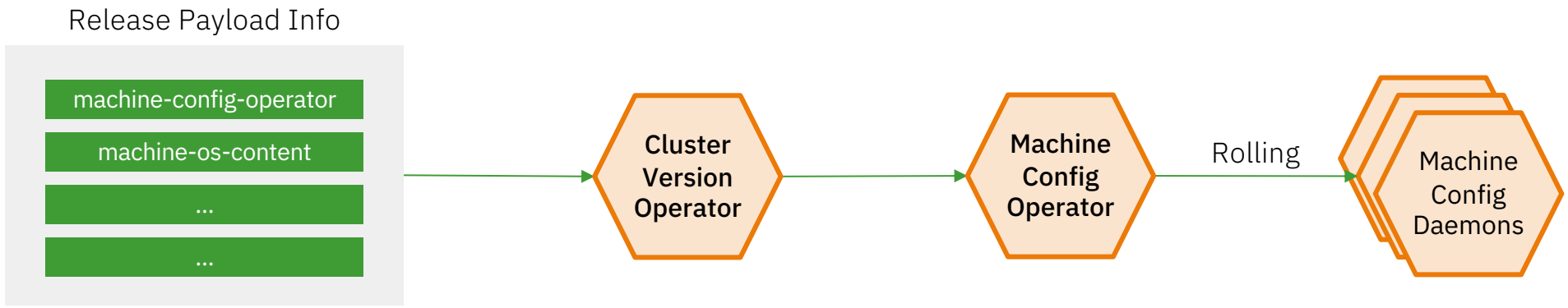
Red Hat Enterprise Linux

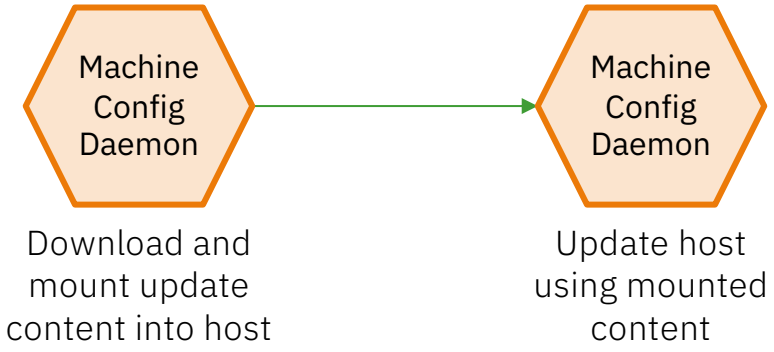
	RED HAT® ENTERPRISE LINUX®	RED HAT® ENTERPRISE LINUX CoreOS
	General Purpose OS	Immutable container host
BENEFITS	• 10+ year enterprise life cycle • Industry standard security • High performance on any infrastructure • Customizable and compatible with wide ecosystem of partner solutions	• Self-managing, over-the-air updates • Immutable and tightly integrated with OpenShift • Host isolation is enforced via Containers • Optimized performance on popular infrastructure
WHEN TO USE	When customization and integration with additional solutions is required	When cloud-native, hands-free operations are a top priority

RHEL versus RHCOS

- Remote management API via Varlink
- Image/container tagging
- Advanced namespace isolation

- Integrated into OCP build pods
- Performance improvements for knative enablement
- Image signing improvements





Over-the-air updates

CONTROL
Application Security

- Container Content
- CI/CD Pipeline
- Container Registry
- Deployment Policies

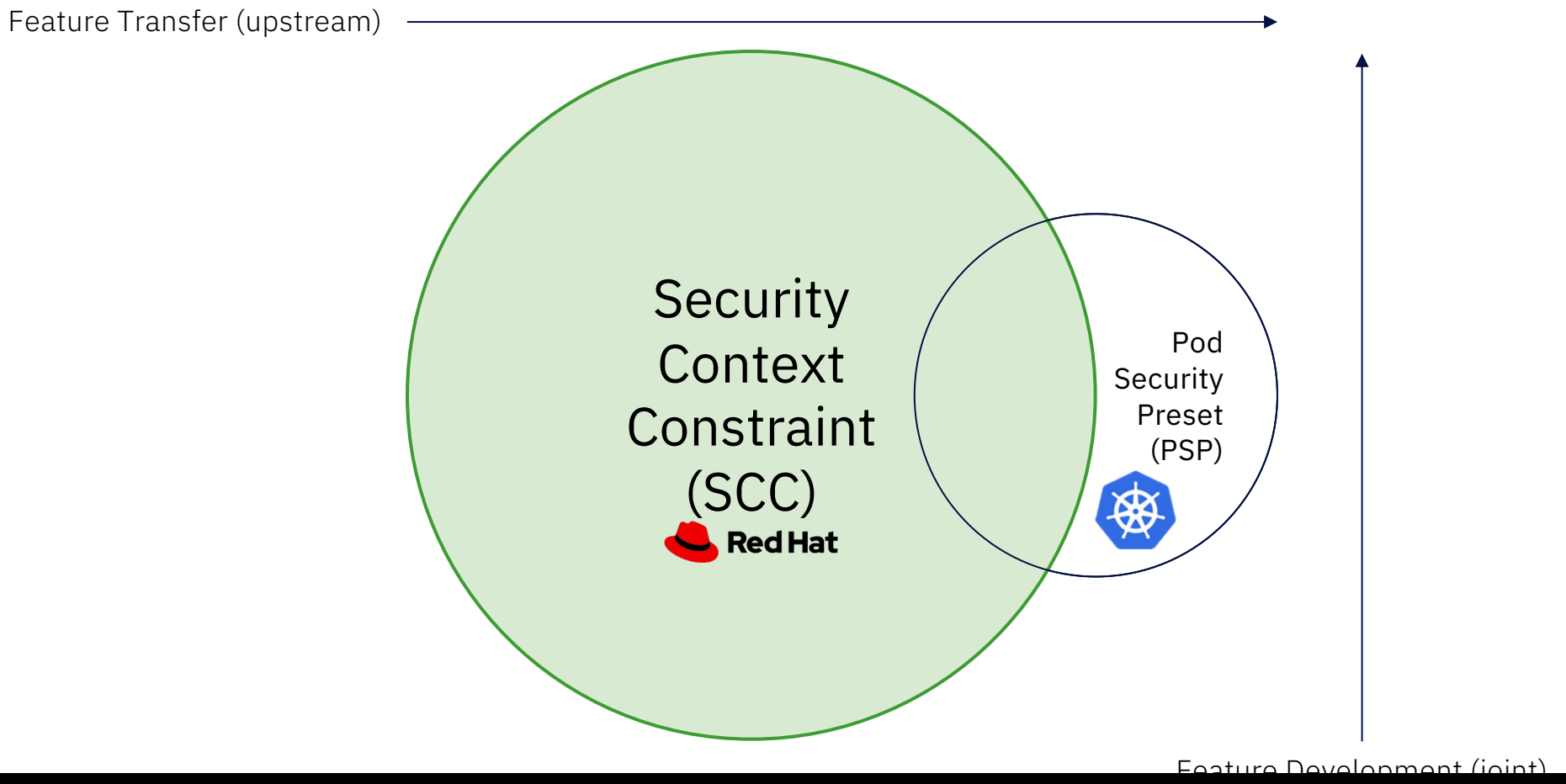
DEFEND
Infrastructure

- Container Platform
- Container Host Multi-tenancy
- Network Isolation
- Storage
- Audit & Logging
- API Management

EXTEND

- Security Ecosystem

Security is built in



Extended Depth of Protection

OpenShift Cluster Monitoring

Metrics collection and storage via Prometheus, an open-source monitoring system time series database.

Alerting/notification via Prometheus' Alertmanager, an open-source tool that handles alerts send by

Metrics visualization via Grafana, the leading metrics visualization technology.

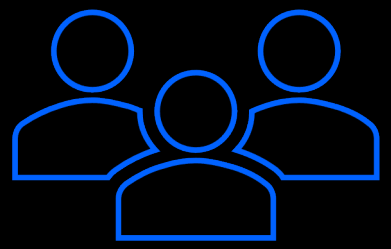
The monitoring bundle

A proven way of introducing or supplementing knowledge of Mainframe systems technologies, each workshop delivers a series of lecture and labs that demonstrate how the cutting-edge, industry-leading IBM Z and LinuxONE systems can solve business problems in many different ways.

The workshops are *no-charge* to qualified customers. Details about the workshops we offer are outlined on IBM.com at:
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1778

Enroll today by contacting your IBM representative!

Deployment and installation



User experience must be a primary consideration

OpenShift – and Kubernetes in general – were never intended to be the top of the architectural stack.

OpenShift makes it exceptionally easy to deploy applications in a rapid fashion.

– Unfortunately, it is also exceptionally easy to deploy applications in a manner which violates basic UXD (user experience design) principles.

Why does any of this matter?

– Cost, risk, and sustainability

Needless exposure of complexity:

– Is the enemy of productivity and user satisfaction.

– Creates risk by encouraging undesirable user behavior.

– Spawns additional needless workflow through supporting processes.

Masking complexity from users is an imperative!

Setup considerations: User experience

Setup considerations:

Deploy an OCP cluster named ***cluster88*** in the network subdomain ***production.ciocloud.example.com***

The cluster would operate under a cluster domain of ***cluster88.production.ciocloud.example.com***

Applications deploy under cluster application domain of ***apps.cluster88.production.ciocloud.example.com***

Deploy an application on this cluster named ***timecard***, in a project named ***hr-applications***, the OCP route would generate URLs that *start with* the following:

https://timecard-hr-applications.apps.cluster88.production.ciocloud.example.com/

You need governance – it is a must.

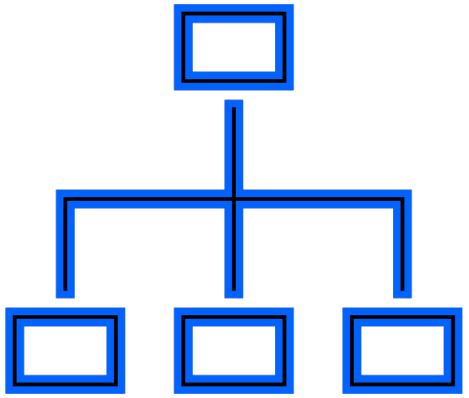
Deploy applications with governance to ensure the messy complexity of the orchestration framework is hidden from users, and that every application is assigned a unique URI path it must remain entirely inside of.

Never permit deployment URLs which use the server root. So I will deploy using the URL path /hr/timecard

Why? This will not be seen by users. It needs to be served via a reverse proxy. More on that soon.

Remember, nobody likes chaos.

Setup considerations:



Load balancing and reverse web proxy

- If you have enterprise load balancing and web proxy solutions, use them.
- If not, you will need to provide your own.
  - Load Balancer
    - Cluster address for your front-end reverse proxies.
    - Cluster addresses for your back-end OpenShift Container Platform.
  - Caching Proxy
    - Unified and consistent front-end service of Web traffic.
    - Consolidated trusted CA certificates can mean ongoing savings in the thousands of dollars.
- If you have purchased the IBM CloudPak for Applications:
  - Licensing for non-containerized IBM WebSphere Application Server Network Deployment may be included. If so, consider deploying WASND Edge Components in a High Availability model:
    - Edge Load Balancer
    - Edge Caching Proxy

Setup considerations:

BANWIS / Bastion

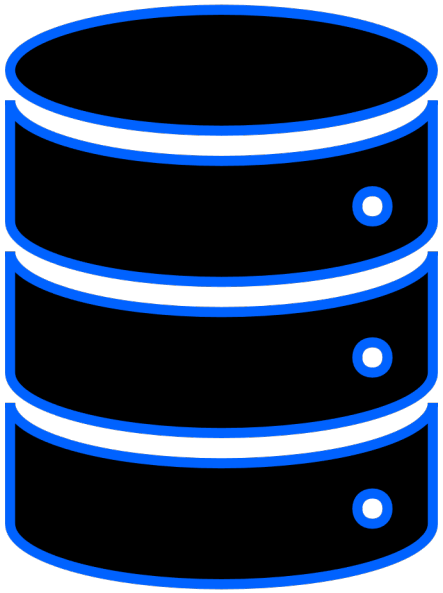
- Using this as a hosting service core during a Proof of Concept is fine, but do not go into production with a single point of failure.
- If you plan to use this for NFS:
  - Create all your exports under /srv/nfs
  - Ensure you will have ample disk space and that /srv/nfs is part of an LVM.
  - Make sure you are creating full back ups of /srv/nfs frequently, and incremental backups even more so.
- If you plan to run your own DNS, you need an HA pair for production.

Setup considerations:

Identity and Access Management (IAM)

- OpenShift Container platform requires a supporting IAM solution for the administrators and developers who will use and interact with it.
- The IBM ATS/WSC team highly recommends using LDAP because it is so prevalent and well understood.
- If you have an enterprise LDAP solution, use it. If that includes SAML integrations, even better.
- If not, you will need to provide your own.
- Your license for z/VM includes the z/VM LDAP server at **no additional charge**.
  - ATS/WSC highly recommends this solution also for the following reasons:
    - Extremely secure, scalable, and reliable.
    - If using RACF/VM for your ESM, the RACF LDAP connector means only one password to maintain for RACF, OCP, and Linux virtual servers.

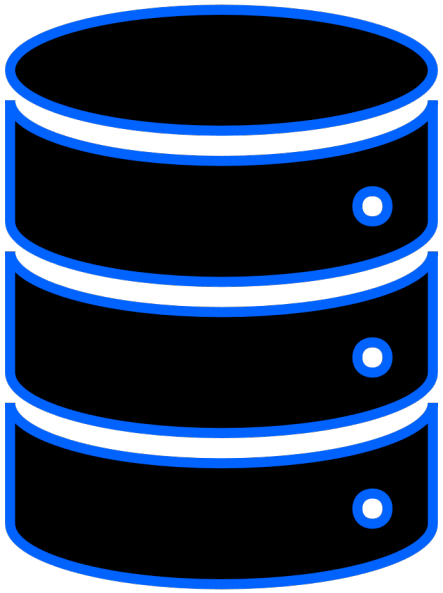
Installation considerations:



OCP Member Nodes

- Run CoreOS as their operating system.
- Each requires at least 120 GB of disk.
  - More if deploying workload requiring extra local ephemeral disk such as blockchain.
    - Consider around 200 GB for these cases as your starting point.
- CoreOS does not use LVM. Your one and only disk must be of sufficient size.
  - Not resizable after installation
  - CoreOS multipath support for FCP/SCSI LUNs planned for the near future
  - You will need 3390-A Extended Address Volumes (EAV) plus aliases available for use.

Installation considerations:



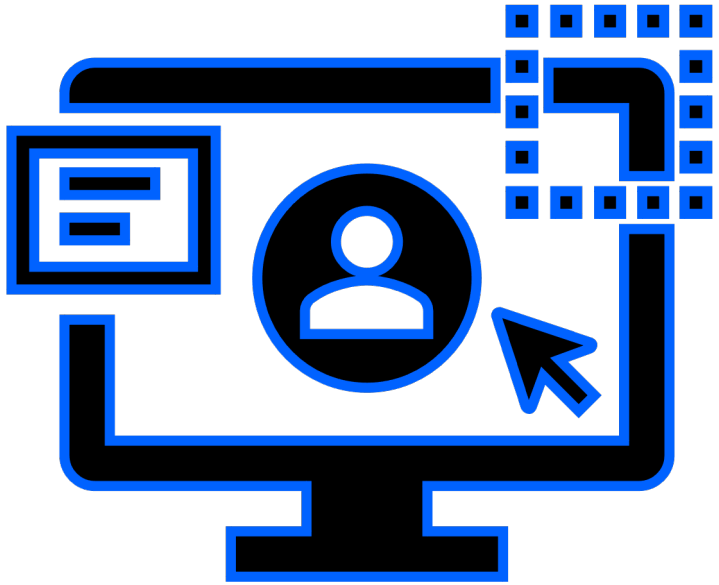
OCP Member Nodes (continued)

- 3390-A EAV plus aliases:
  - Consider number of aliases per LPAR.
  - Keep in mind that aliases do not need to be dedicated. Let z/VM virtualize the aliases.
  - Give each node 6 to 8 virtual aliases as a good starting point.
    - Ephemeral storage and disk I/O intensive workloads need 8 to 10.

BANWIS / Bastion and supporting infrastructure virtual systems

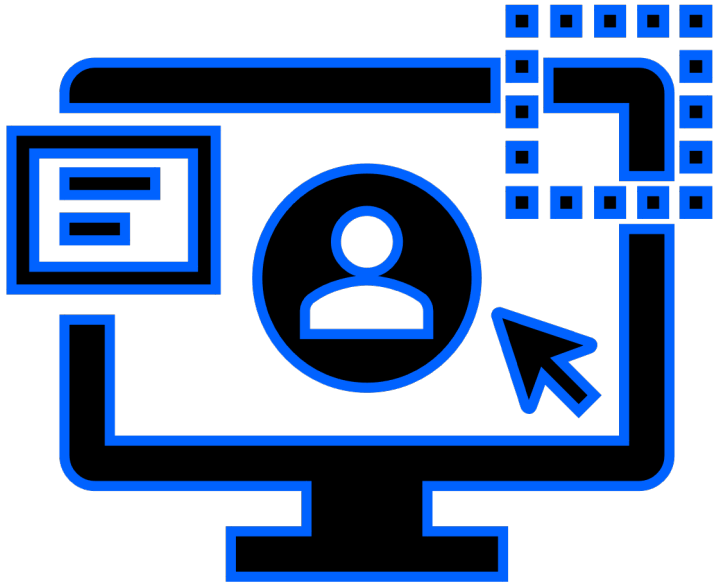
- Should follow your deployment standards for Linux virtual server
- The restrictions of CoreOS don’t apply here

Configuration considerations:



- This is a cluster – consistency is very important.
- Consider using STP if you are not already.
- Use shared profiles in the z/VM user directory:
  - Control plane nodes.
  - Compute nodes.
  - Infrastructure (offload) nodes.
    - If or when you eventually go down this route.
- Attach minidisks as DEVNO to leverage HyperPAV or use the 1-END Minidisk HyperPAV support in z/VM 7.2
- Memory (STORage, MAXSTORage, STANDBY) depends on node type. More on this coming up.

Configuration considerations:



- Again, consistency is very important.
  - Workloads are continually rebalanced.
  - In the event of a failure or maintenance, pods respawn on alternate nodes.
  - All of the nodes within the cluster should be consistent by respective type. Shared profiles are an easy way to do this.
- Examples of what shared profiles might look like are on the next two slides.

Configuration considerations:

Values here are shown as example only

```
PROFILE LPOCPCPN
***[ PROFILE: Linux | OCP ON Z | CONTROL PLANE NODE ]***
   CLASS G
   STORAGE 18G
   MAXSTORAGE 32G
   COMMAND DEFINE STORAGE STANDBY 6G RESERVED 0
   COMMAND SET RUN ON
   COMMAND TERM HOLD OFF
   COMMAND TERM MORE 001 000
   COMMAND SET PF12 RETR BACK
   COMMAND SET PF11 RETR FORW
   COMMAND SET VCONFIG MODE LINUX
   COMMAND DEFINE HYPERPAVALIAS A800 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A801 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A802 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A803 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A804 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A805 FOR BASE 0700
   COMMAND DEFINE CPU 00-05 TYPE IFL
   DATEFORMAT ISODATE
   IPL 190 PARM AUTOCR
   IUCV ALLOW
   IUCV ANY PRIORITY MSGLIMIT 2000
   LOGONBY HAYDEN PWNOVAK BADER MMONDICS SHALAWN
   MACHINE ESA 10
   OPTION APPLMON CHPIDV ONE
   XAUTOLOG LNCG4010 LNCG4020 LNCG4030 LNCG4030
   CONS 0009 3215 T OPMGRM1 OBSERVER
   NICDEF 0AD0 TYPE QDIO LAN SYSTEM VSWITCH3
   SPOOL 000C 2540 READER *
   SPOOL 000D 2540 PUNCH A
   SPOOL 000E 1403 A
   LINK OCPADMIN 0192 0192 RR
   LINK LNXMAINT 0191 0191 RR
```

Configuration considerations:

Values here are shown as example only

```
PROFILE LPOCPCON
***[ PROFILE: Linux | OCP ON Z | COMPUTE NODE ]***
   CLASS G
   STORAGE 12G
   MAXSTORAGE 64G
   COMMAND DEFINE STORAGE STANDBY 6G RESERVED 0
   COMMAND SET RUN ON
   COMMAND TERM HOLD OFF
   COMMAND TERM MORE 001 000
   COMMAND SET PF12 RETR BACK
   COMMAND SET PF11 RETR FORW
   COMMAND SET VCONFIG MODE LINUX
   COMMAND DEFINE HYPERPAVALIAS A800 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A801 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A802 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A803 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A804 FOR BASE 0700
   COMMAND DEFINE HYPERPAVALIAS A805 FOR BASE 0700
   COMMAND DEFINE CPU 00-05 TYPE IFL
   DATEFORMAT ISODATE
   IPL 190 PARM AUTOCR
   IUCV ALLOW
   IUCV ANY PRIORITY MSGLIMIT 2000
   LOGONBY HAYDEN PWNOVAK BADER MMONDICS SHALAWN
   MACHINE ESA 10
   OPTION APPLMON CHPIDV ONE
   XAUTOLOG LNCG4010 LNCG4020 LNCG4030 LNCG4030
   CONS 0009 3215 T OPMGRM1 OBSERVER
   NICDEF 0AD0 TYPE QDIO LAN SYSTEM VSWITCH3
   SPOOL 000C 2540 READER *
   SPOOL 000D 2540 PUNCH A
   SPOOL 000E 1403 A
   LINK OCPADMIN 0192 0192 RR
   LINK LNXMAINT 0191 0191 RR
```

Thank you

Paul Novak
Senior IT Specialist
SME, Virtualization & Cloud
on IBM zSystems & LinuxONE

Endicott – The Birthplace of IBM
1701 North St
Endicott, NY 13760 USA

Tel +1 607 429 6186
pwnovak@us.ibm.com

Jacob Emery
Technical Enablement
SME, Cloud on IBM zSystems
& LinuxONE

600 Anton Blvd
Costa Mesa, CA 92626 USA

Tel +1 872 772 9858
jacob.emery@ibm.com

Matt Mondics
Technical Specialist
Hybrid Cloud on IBM Z and
LinuxONE

10500 Cedar Ave
Cleveland, OH 44106 USA

Tel +1 614 551 7720
matt.mondics@ibm.com

Visit our website at http://www.ibm.com/support/techdocs