Monolith to Microservices with Dynatrace & OpenShift

https://github.com/steve-caron-dynatrace/Monolith_to_Microservices

https://bit.ly/dt-rh-jun11

What we will accomplish

- Set-up TicketMonster
- Extract the UI from the monolith
- Identify a microservice
- Identify the domain model of the microservice
- Deploy the microservice
- Familiarization with cloud platform
- Deployment of microservice
- Confidence in breaking the monolith
- Persistence layer for the microservice
- (real) microservice extracted

Check Prerequisites

Make sure your laptop has Internet Access and a modern browser (Google Chrome is ideal)

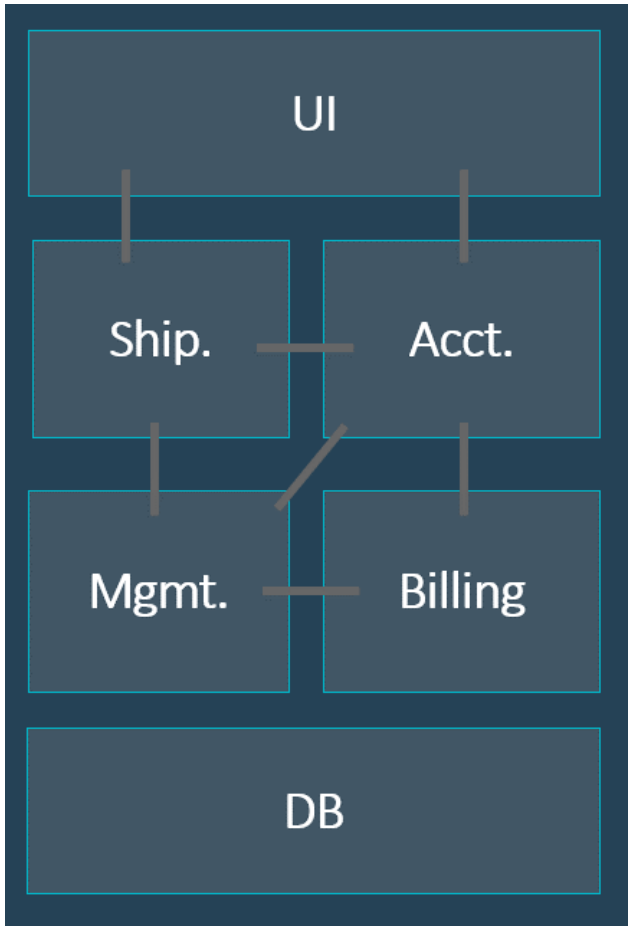
Workshop OpenShift Cluster:	
URL:	https://console-openshift-console.apps.cluster-dynatrac-edbb.dynatrac-edbb.example.opentlc.com/
User:	user## (## = assigned number)
Password:	openshift
Workshop Dynatrace tenant:	
URL:	https://wxu551.dynatrace-managed.com/
User:	user## (## = assigned number)
Password:	dynatrace

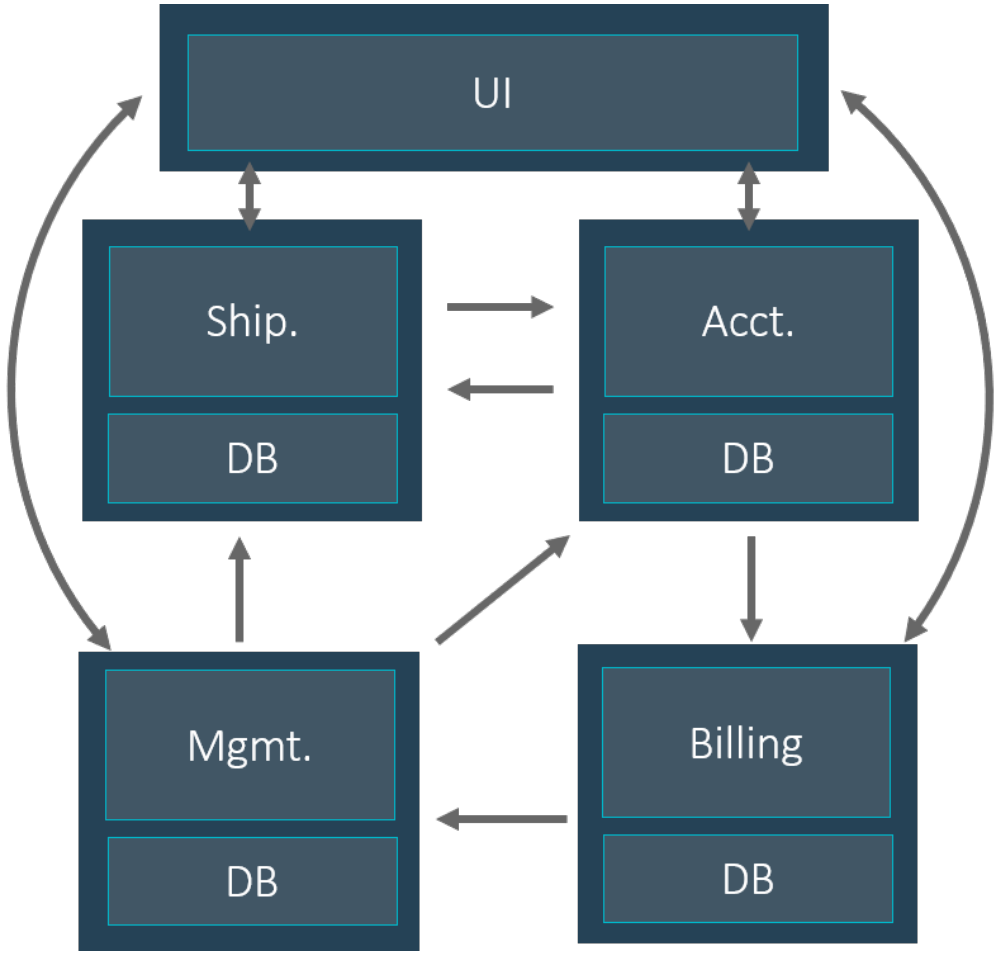
Github Repo:	
URL:	https://github.com/steve-caron-dynatrace/Monolith_to_Microservices

Introduction

Limitations of Monoliths have given rise to Microservices

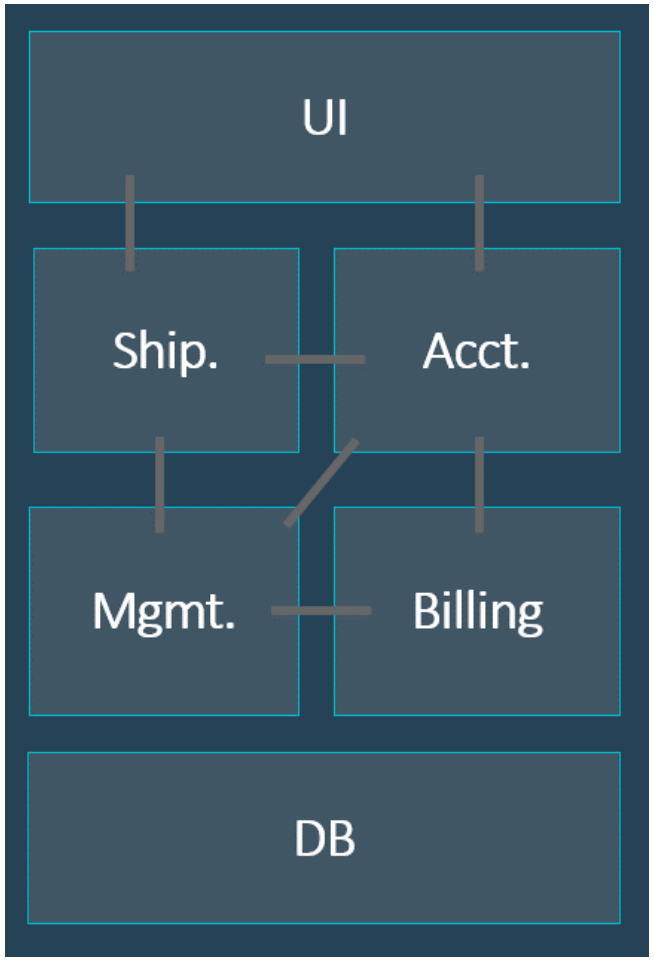
From monolith to microservices





Limitations of monolithic applications

- **Agility** –Rebuilding the whole application takes a decent amount of time
- **Scalability** – Scaling a monolith happens in both directions: vertically as well as horizontally - causing unused resources
- **DevOps Cycle** – Continuous delivery (high frequency of deployments) fails due to high build time
- Availability, fault tolerance, and resiliency

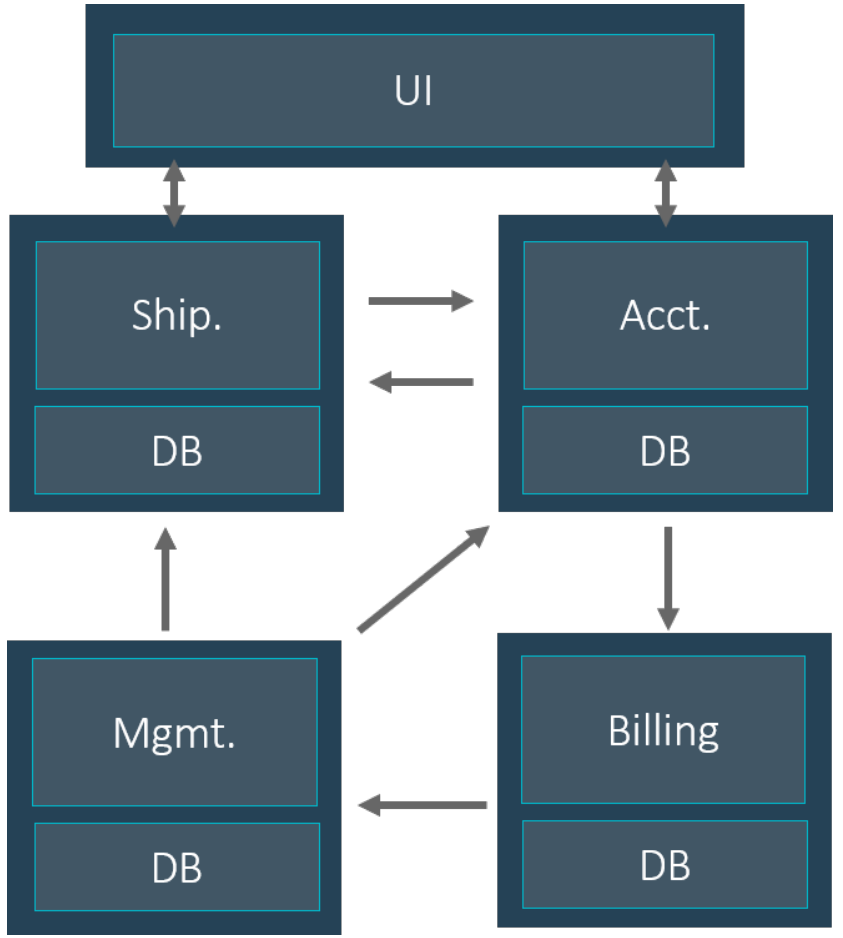


Limitation of monolithic applications

Source: https://tommcfarlin.com/design-patterns-for-refactoring-facade/

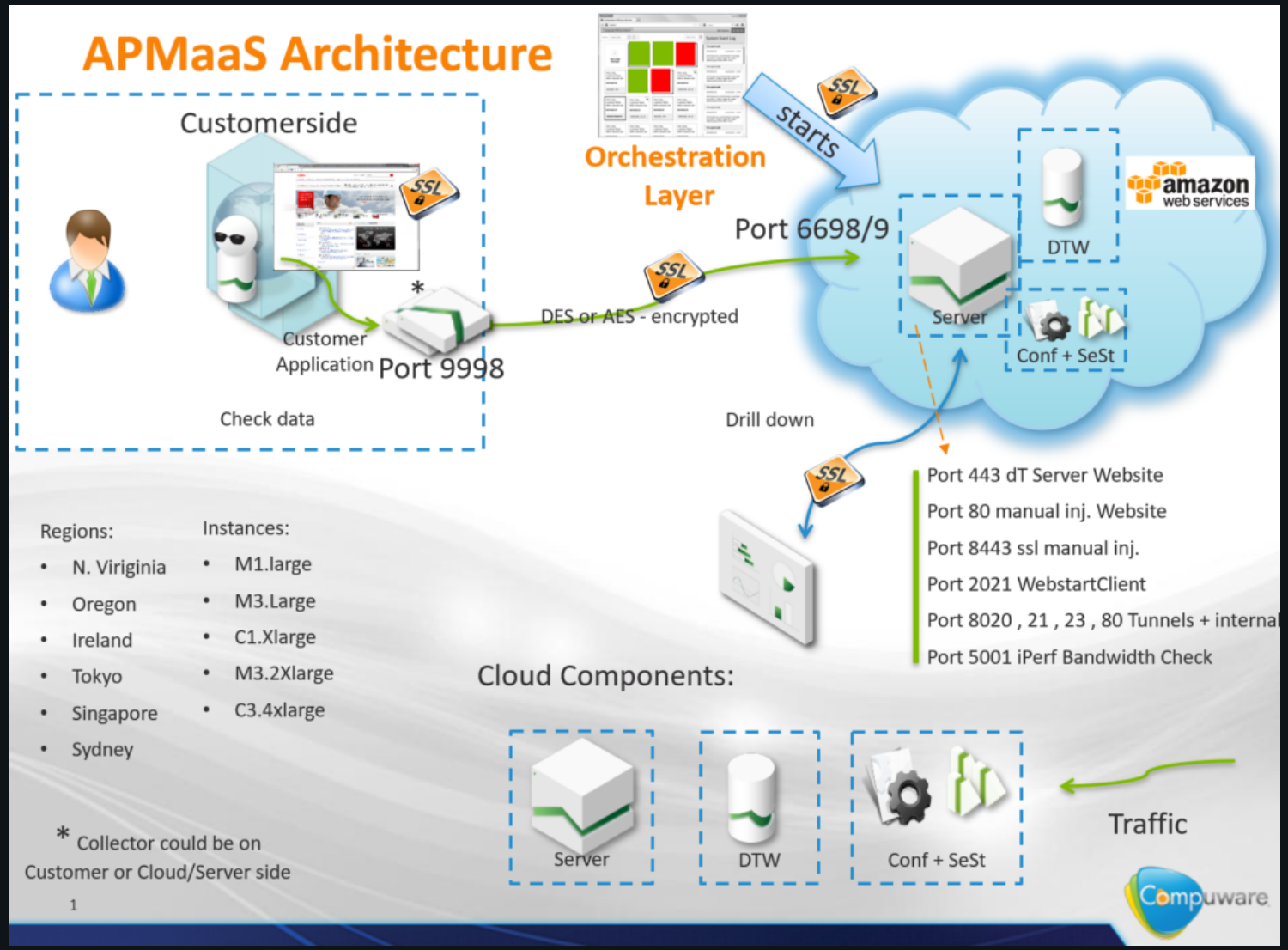
Limitations of Monoliths have given rise to Microservices

- **Agility** - Scope changes can be done in one microservice - other microservices are not impacted from these changes
- **Scalability** - Individual components can scale as needed
- **DevOps Cycle** - Since each component operates independently, continuous delivery cycle reduces



Just moving an application to the cloud won't do it – even if it feels that way

Just moving an application to the cloud won't do it – even if it feels that way



Your App / Container

Pivotal Cloud Foundry

OPENSHIFT

kubernetes

aws

Google Cloud Platform

Microsoft Azure

SAP

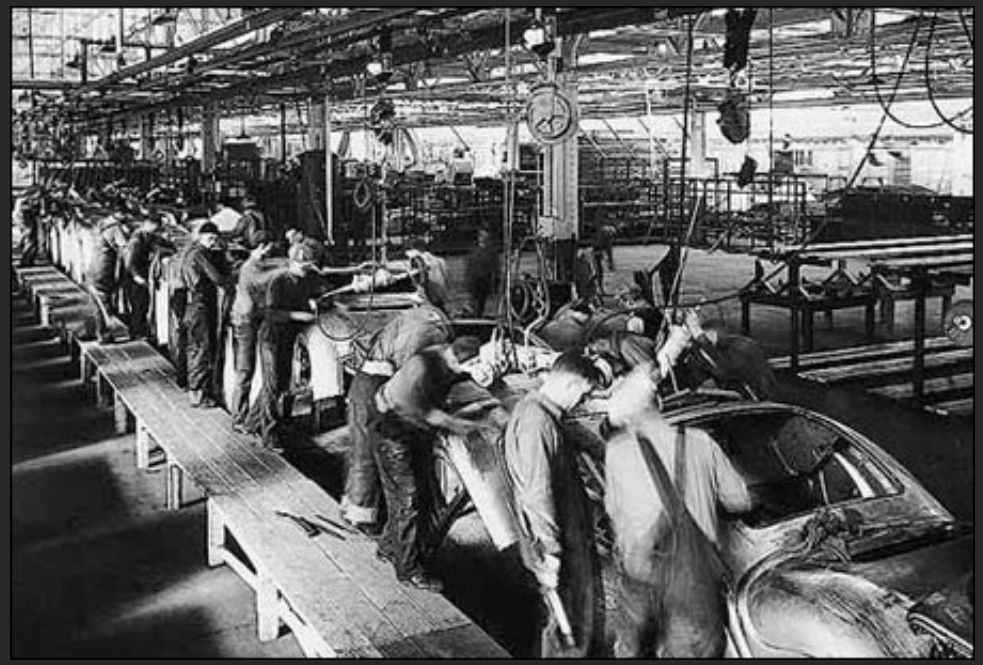
≠

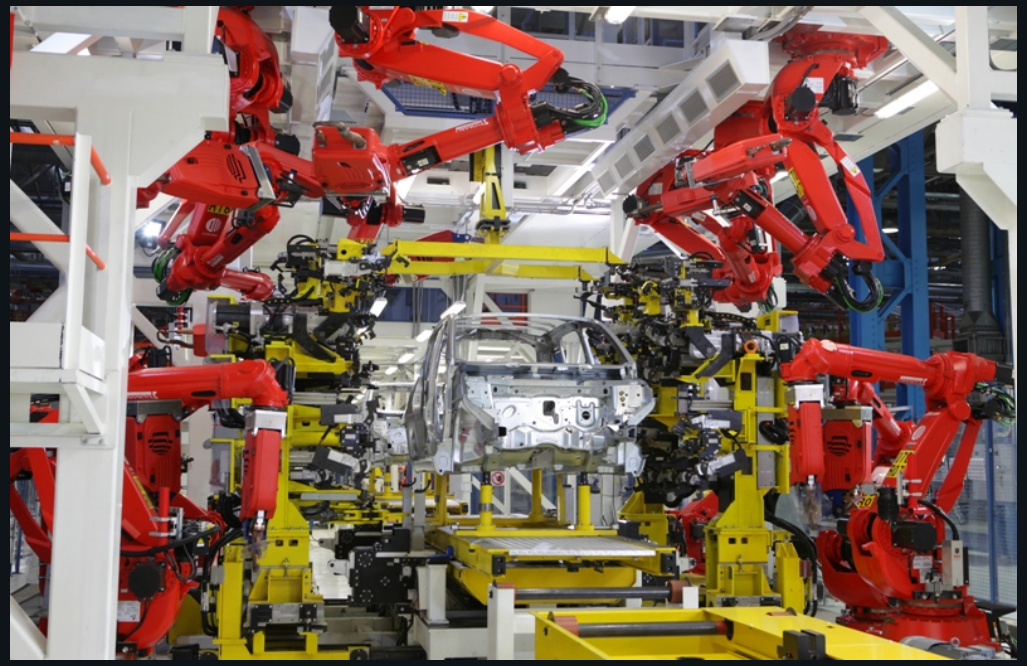
Autonomous Cloud:
Deliver Digital Business Value Faster

You need to change radically

Changing a technology stack and a modern UI is not enough

Your understanding of how you deliver needs to change



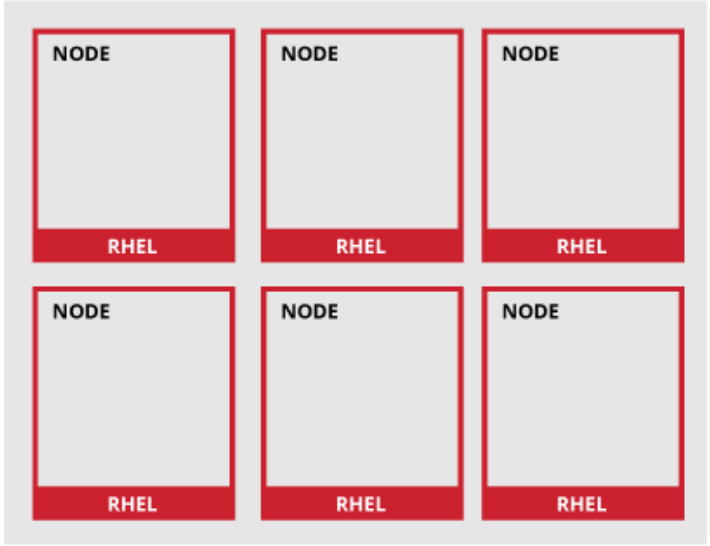


Openshift & Dynatrace primer

OpenShift Architecture

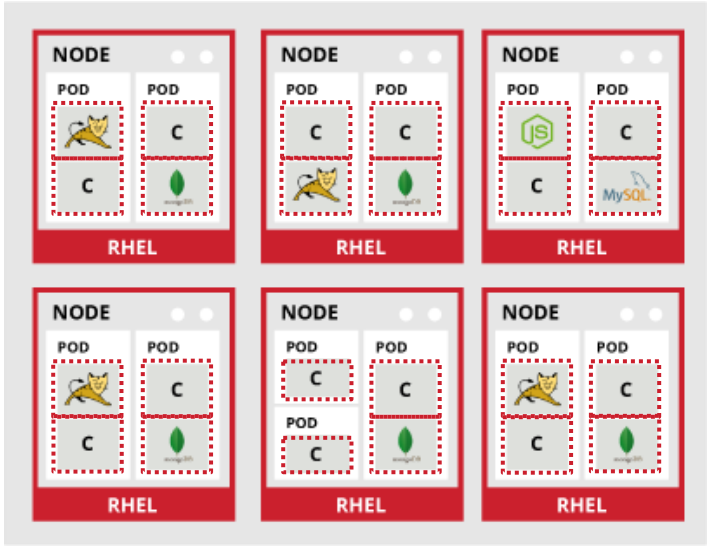
OpenShift Architecture

OpenShift nodes are instances of
Red Hat Enterprise Linux / Atomic Host



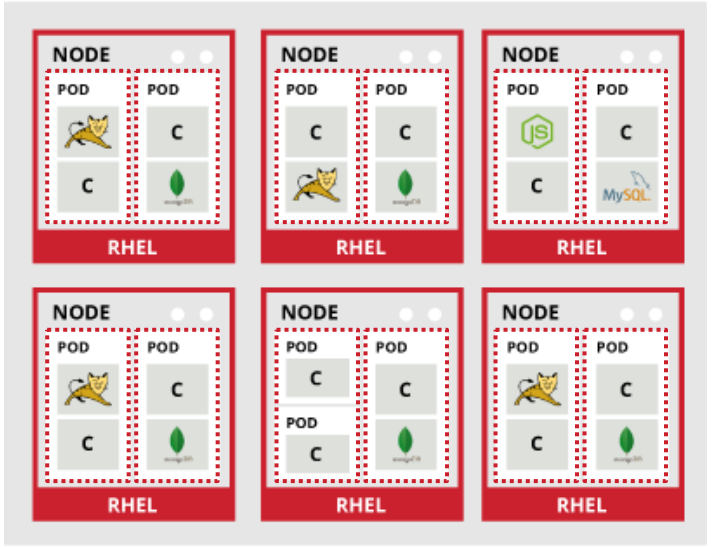
OpenShift Architecture

Application services run in containers on each node



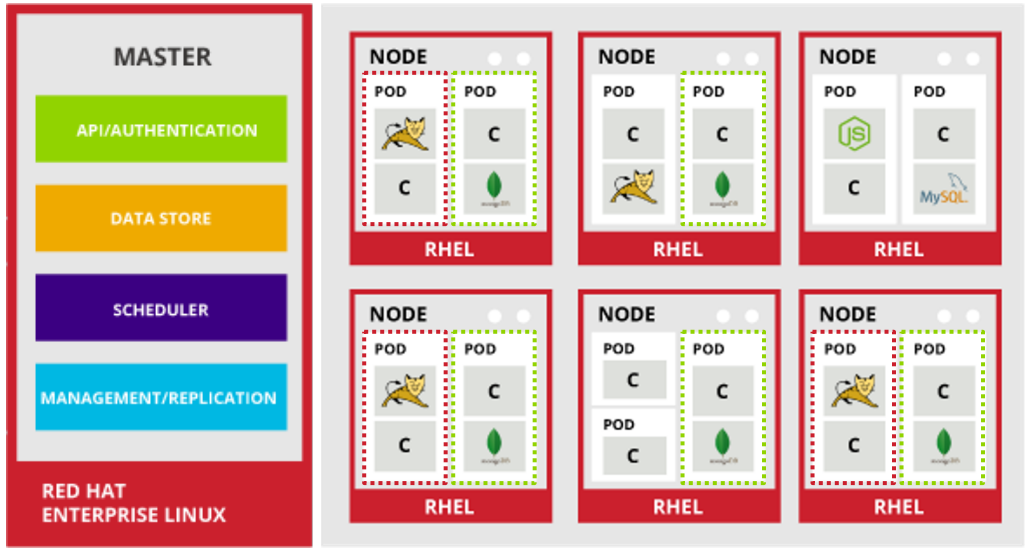
OpenShift Architecture

Pods run one to many containers as a co-located unit

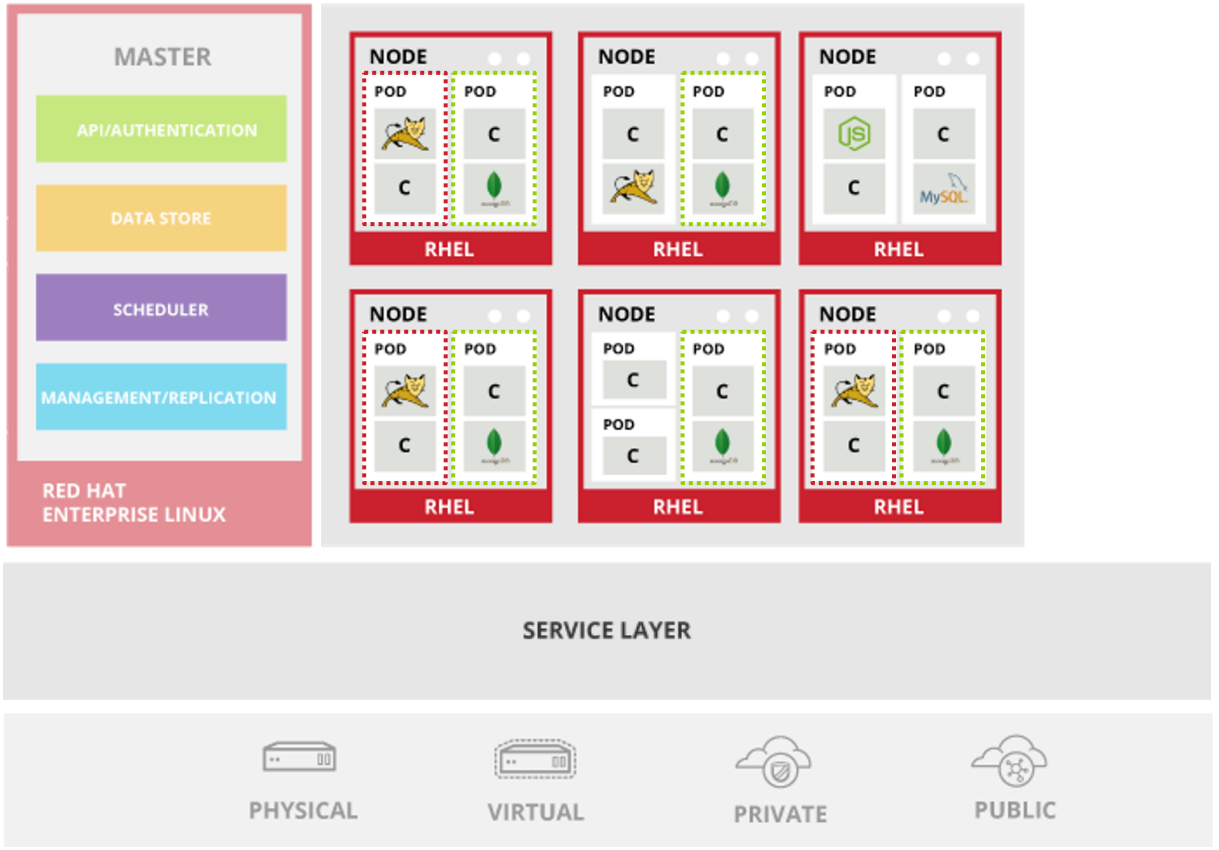


OpenShift Architecture

Masters leverage *Kubernetes*
to orchestrate nodes and applications



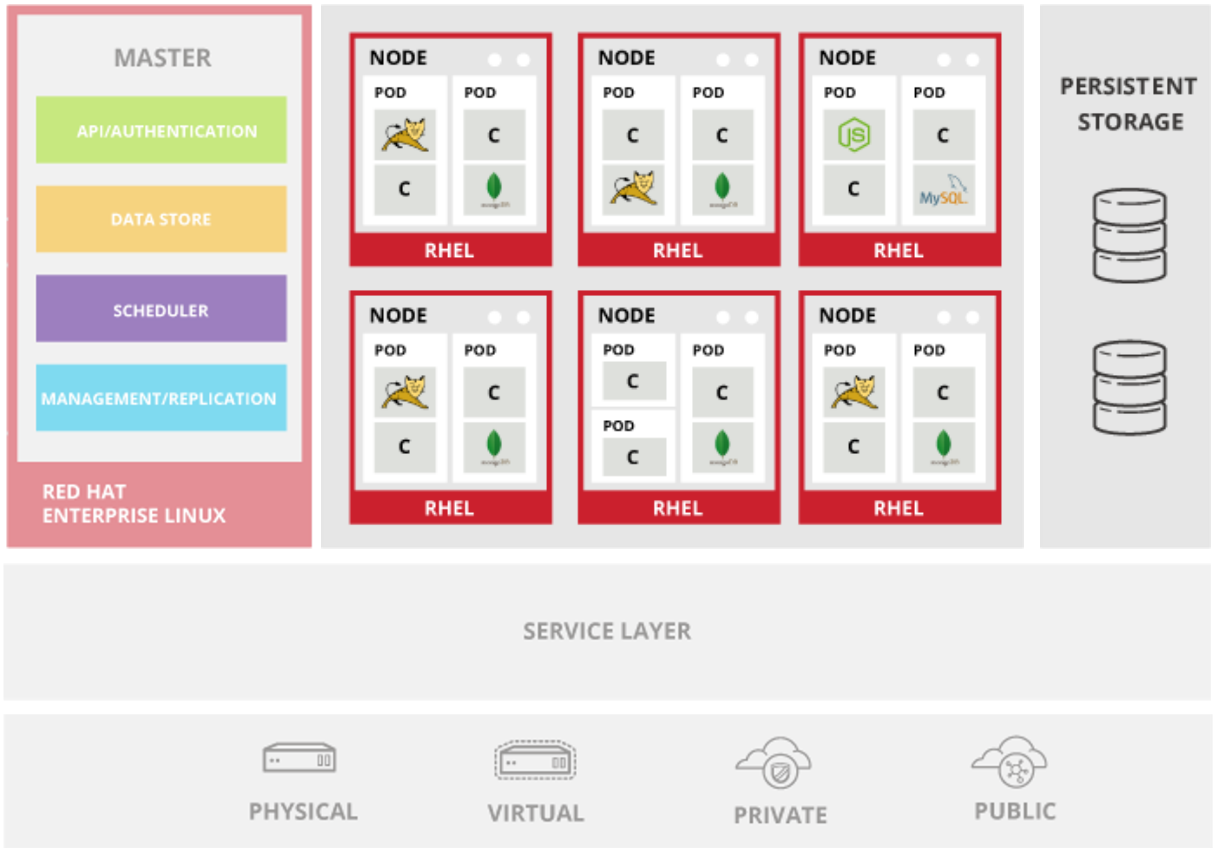
OpenShift Architecture



Services allow related pods to connect to each other

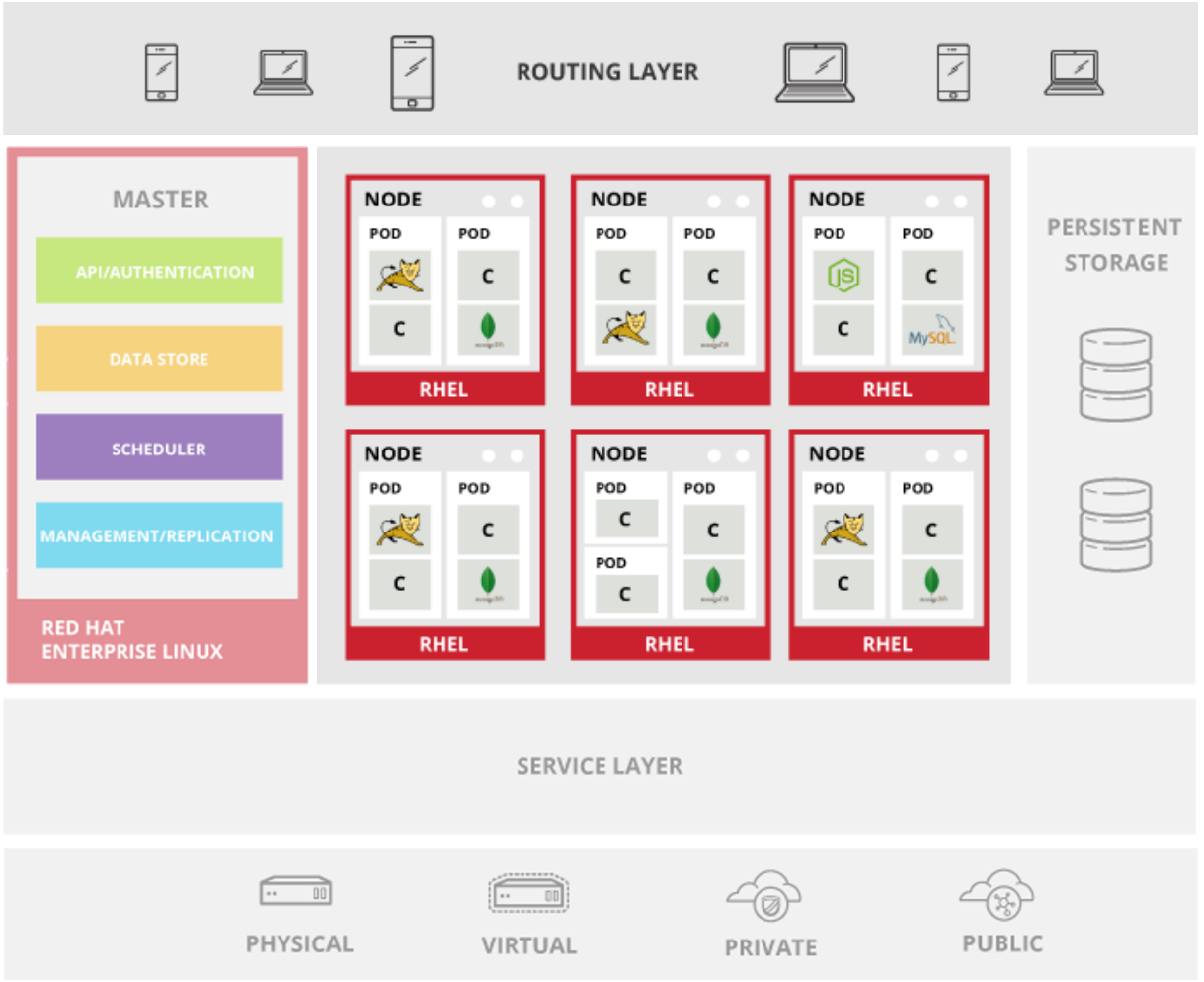
OpenShift Architecture

Pods can attach to shared persistent storage



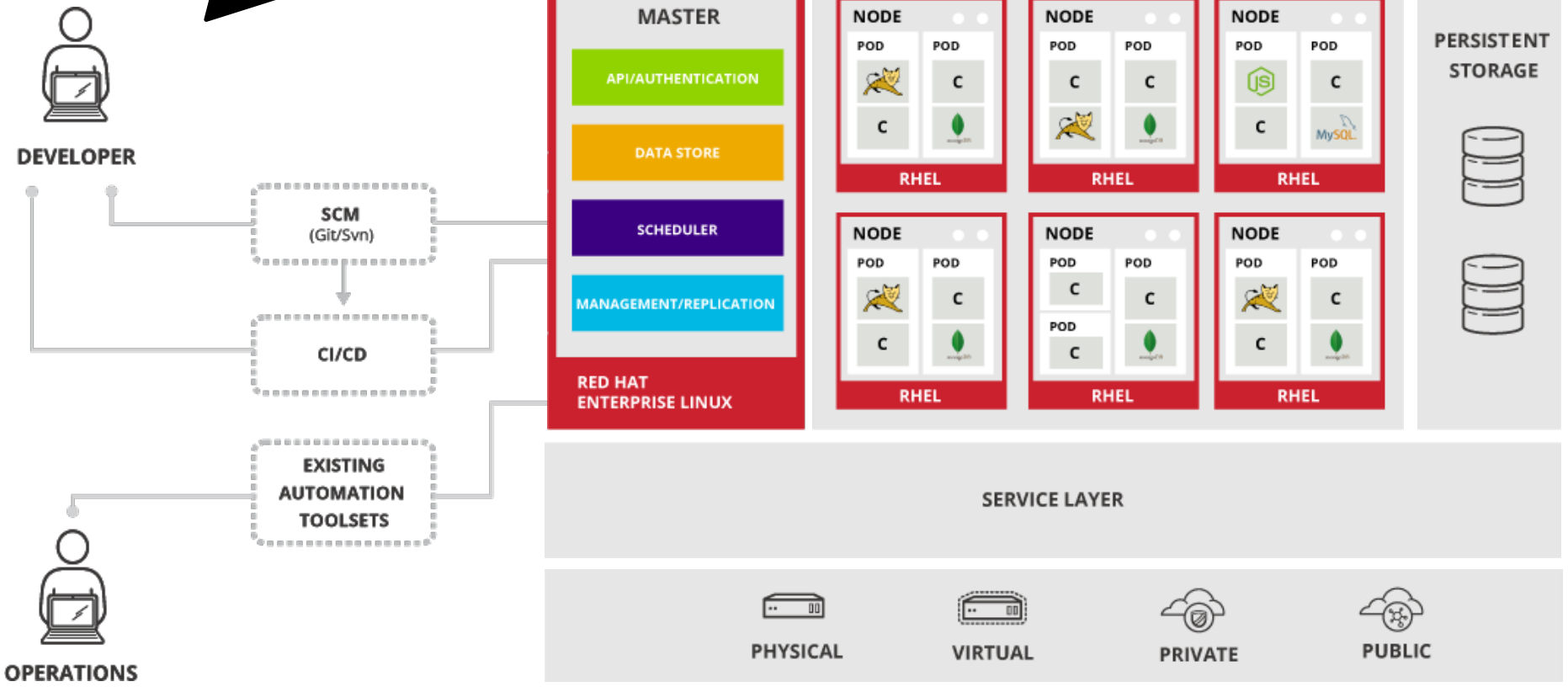
OpenShift Architecture

Routing routes external requests to pods



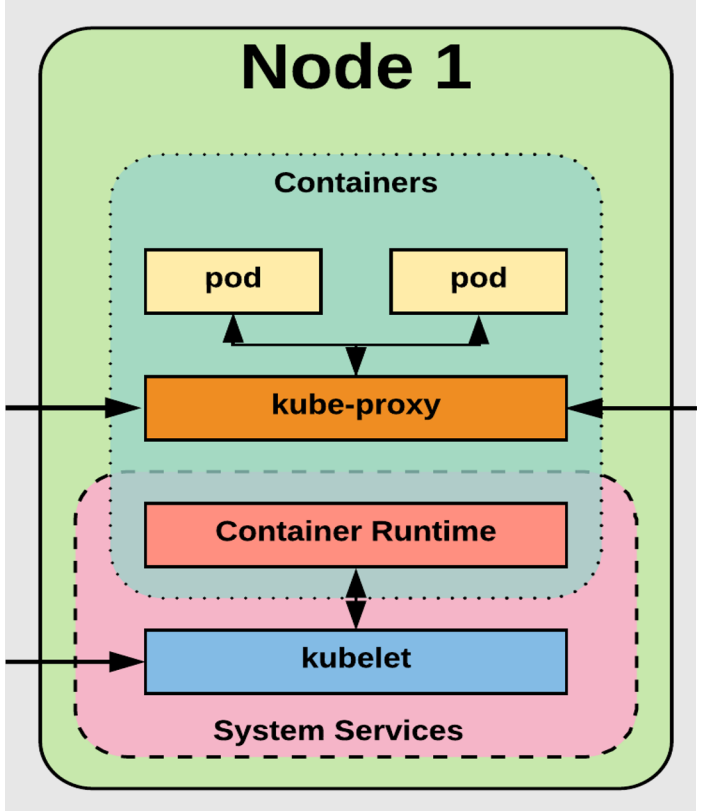
OpenShift Architecture

Devs and *Ops* access OpenShift via CLI, Web UI or IDE



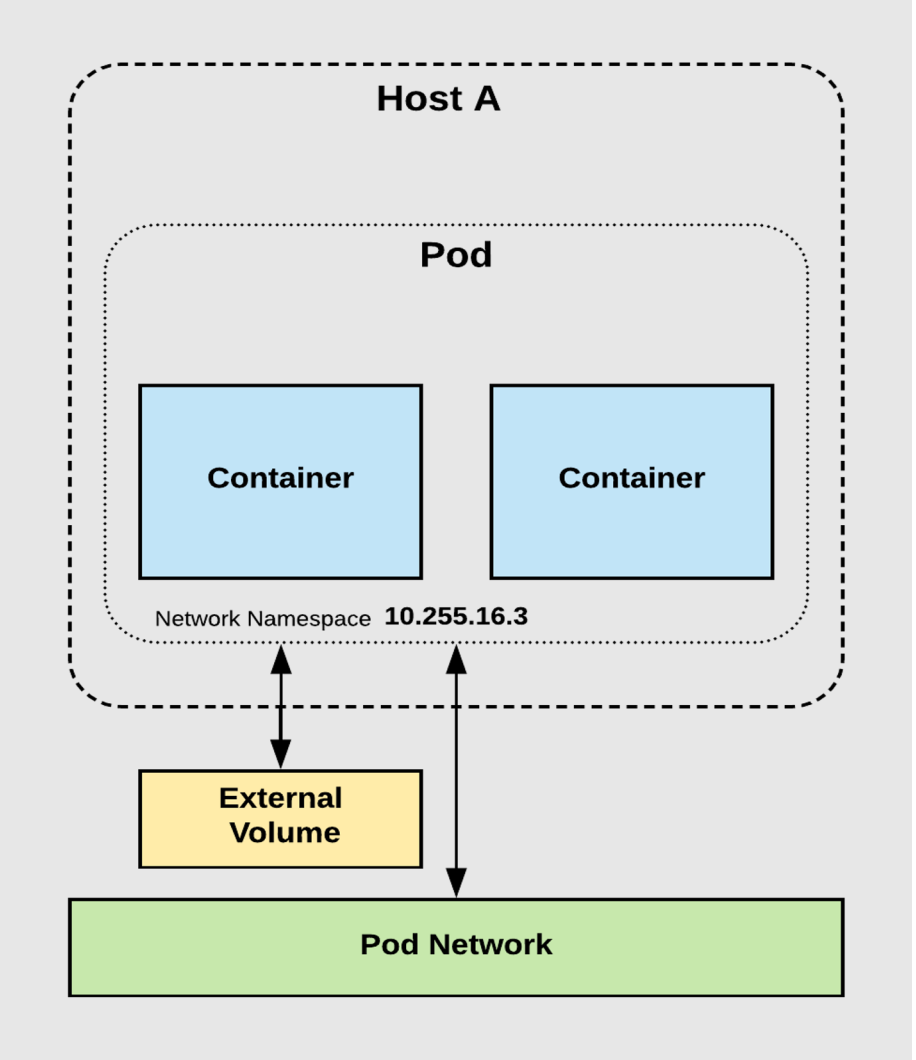
Key Concepts - Nodes

- Worker machine in Kubernetes
- VM or physical machine
- Managed by the control plane (master)
- Node components:
  - kubelet
  - kube-proxy
  - Container Runtime Engine



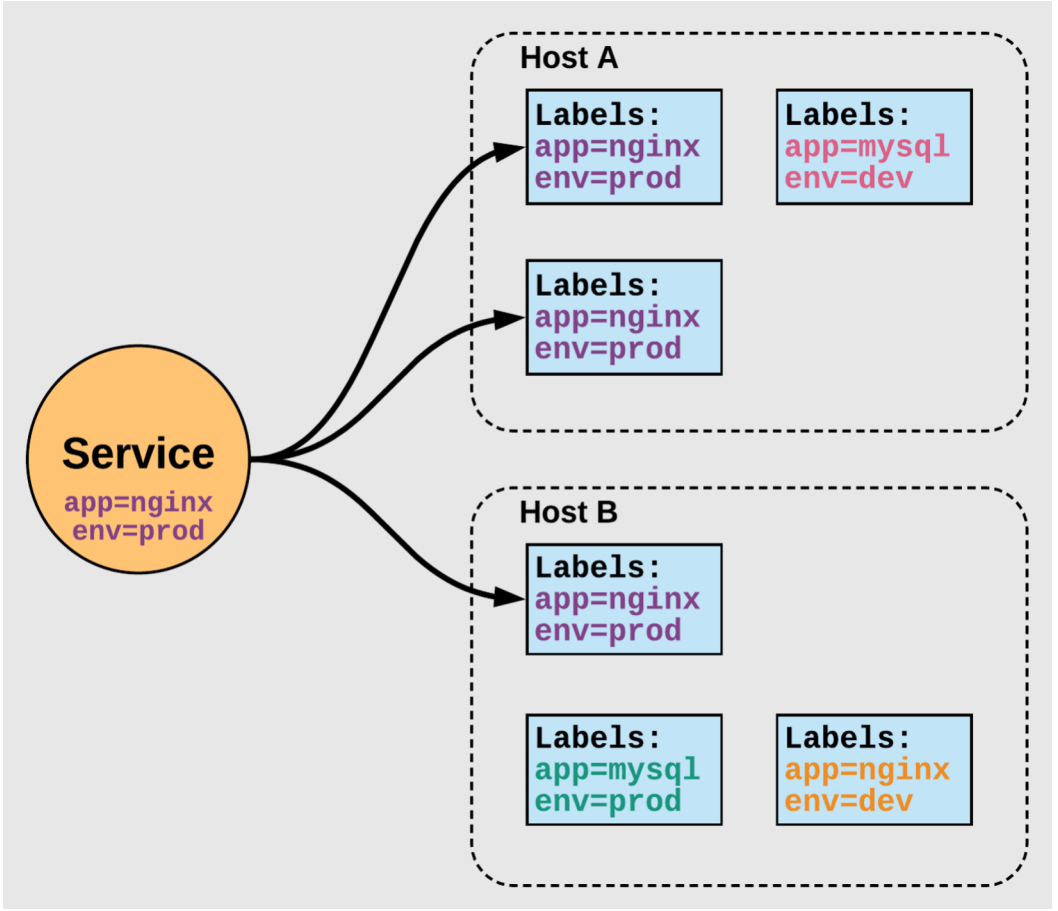
Key Concepts - Pods

- Smallest “unit of work” in Kubernetes.
- Pods are one or more containers that share volumes and namespace.
- They are also ephemeral



Key Concepts - Services

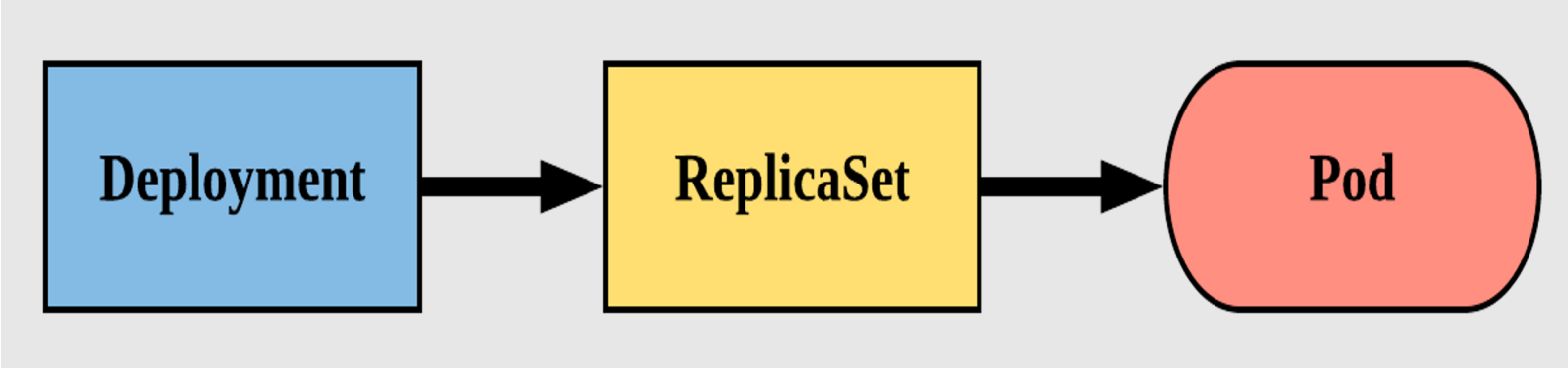
- Unified method of accessing the exposed workloads of pods.
- Think of it as an internal load balancer to your pods.
- How it is implemented depends on the cloud provider or the on-prem config
- Not ephemeral



<service name>.<namespace>.svc.cluster.local

Key Concepts – Deployments

- Way of managing Pods via ReplicaSets
- Provide rollback functionality and update control
- Each iteration creates a unique label that is assigned to both the ReplicaSet and subsequent Pods.



OpenShift concepts cheat sheet

Nodes
- are instances of underlying OS (Linux)
- virtual or physical machine
- host one to many *Pods*

Pods
- are (co-located) groups of *containers*
- a Pod's containers run *on the same host (node)*

Labels
- assign *identifying metadata* to *objects* (Pods)
- are used to organize, group or select objects

Annotations
- assign *non-identifying metadata* to objects
- are used for arbitrary data (build, debug info)

Deployments/Replicasets
- select a Pod object using *labels*
- ensure a given number of *Pod replicas*
- handle *scale-up/down* and *rolling updates*

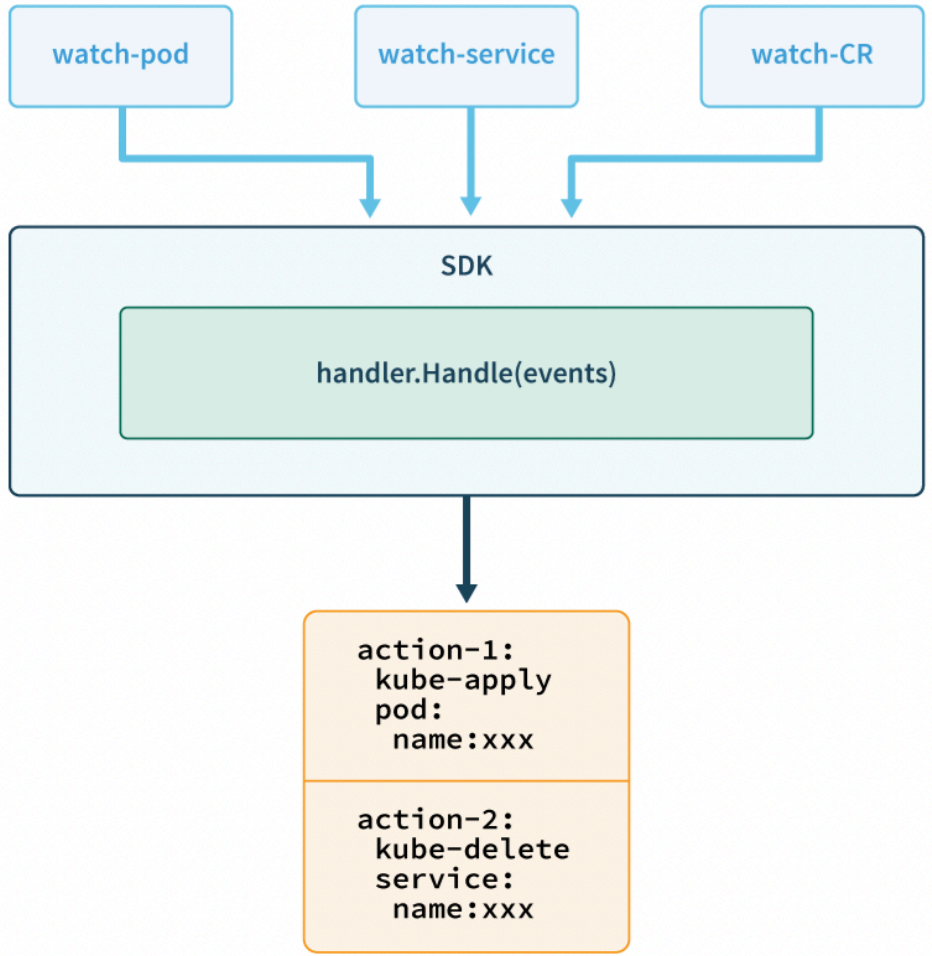
Services
- select a Pod object using *labels*
- abstract a (replicated) set of Pods by *IP, port(s)*
- provide *domain name* and simple *load balancing*

Routes
- expose services to the (cluster) outside world

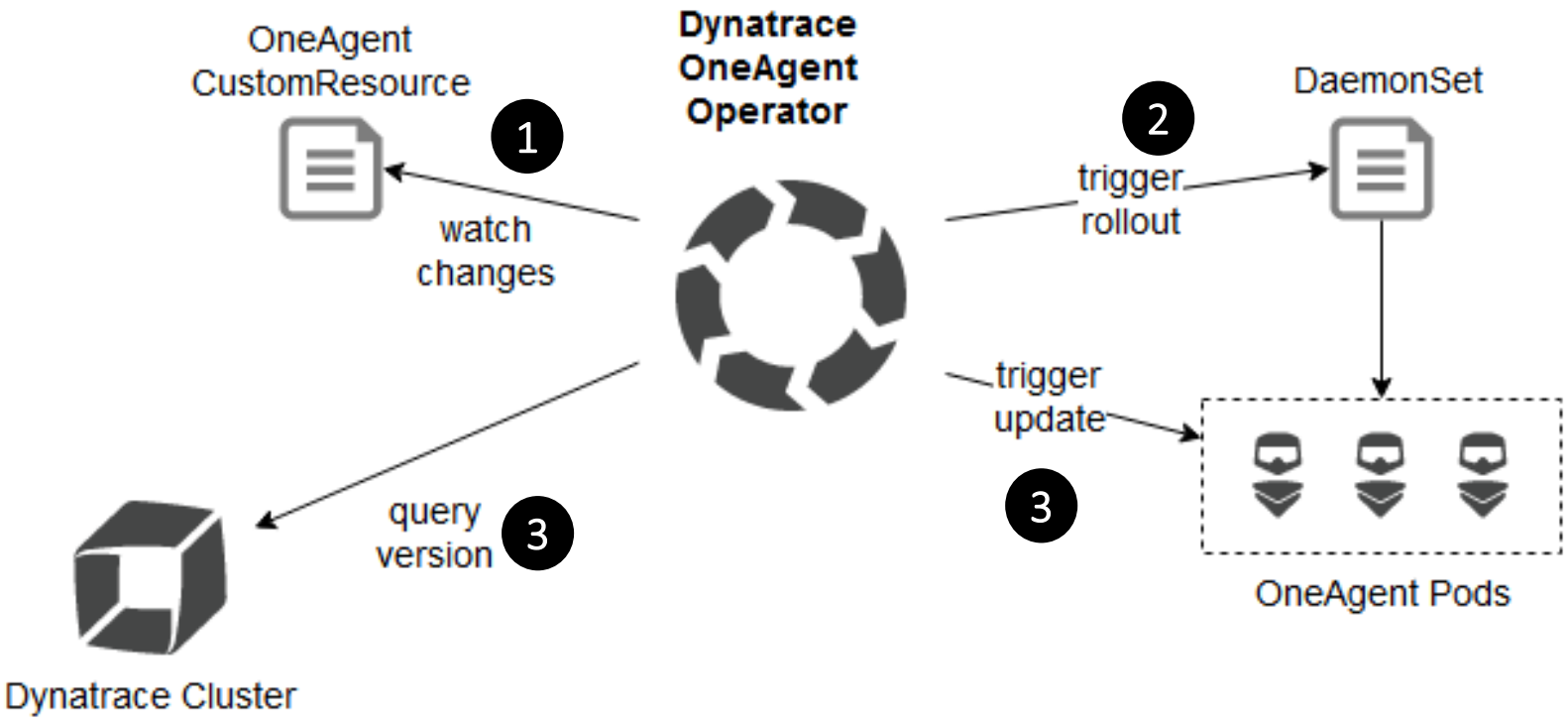
Projects
- aka namespaces: allows a community of users to organize and manage their content in isolation from other communities.

What is an Operator?

- The Operator Framework is an open source toolkit to manage Kubernetes native applications in an effective, automated, and scalable way.
- Human operational knowledge in software
- Reliably manage application lifecycles as a first-class kube native object



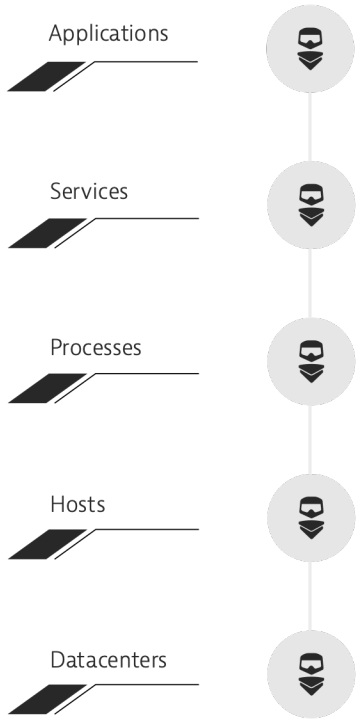
Dynatrace OneAgent Operator in action

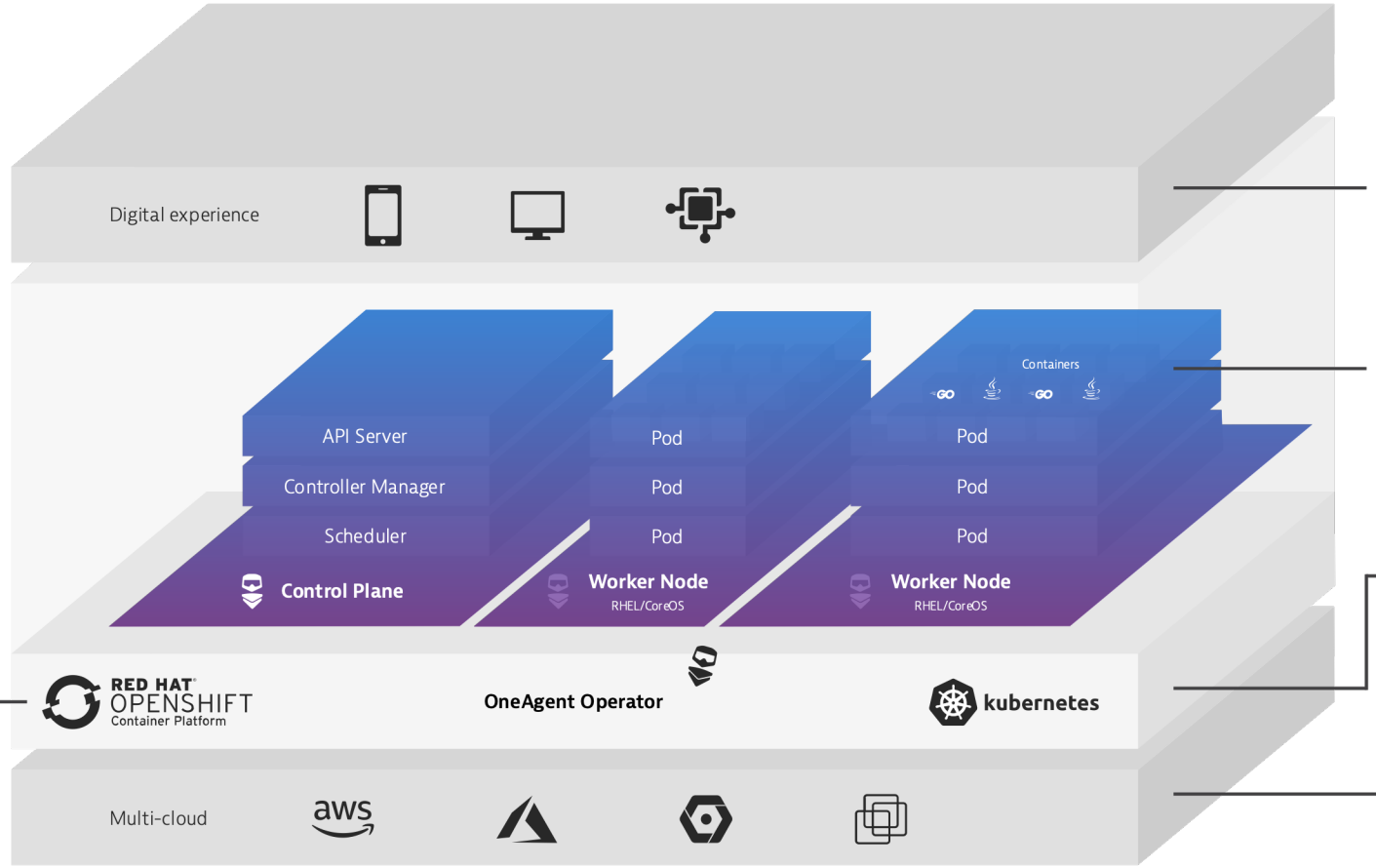


1. Watches for custom resources of type OneAgent
2. Takes care of OneAgent deployment via DaemonSet
3. Updates OneAgent to the latest version available

All-in-one monitoring of OpenShift cluster and workload

OneAgent deploys automatically via the OneAgent Operator to all layers and technologies in your environment





Monitor, analyze and optimize every digital interaction

Real-time auto discovery through OneAgent Operator injection of Docker and CRI-O containers without code or image changes

Automatic and continuous deployment of Dynatrace OneAgent Operator to all components

Full integration with all major cloud platforms