Microsoft

OpenShift
On Azure Workshop

redhat

Presenter: Veer Muchandi
Date: 02/07/2017
Title: Principal Architect - Container Solutions
Social Handle: @VeerMuchandi
Blogs: https://blog.openshift.com/author/veermuchandi/

@OpenShift

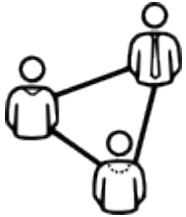
RHOpenShift

Digital Transformation

There is Evolution in...

APPLICATIONS

New ways of developing, delivering and integrating applications



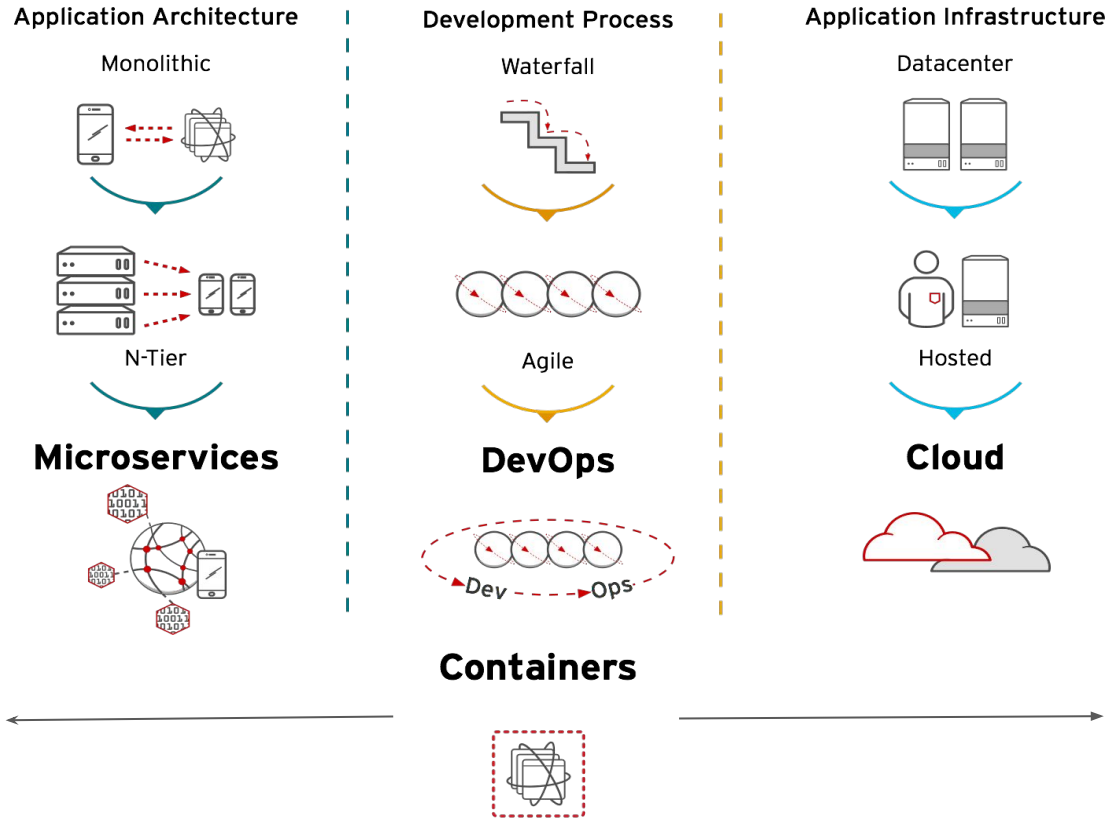
PROCESS

More agile processes across both IT and the business

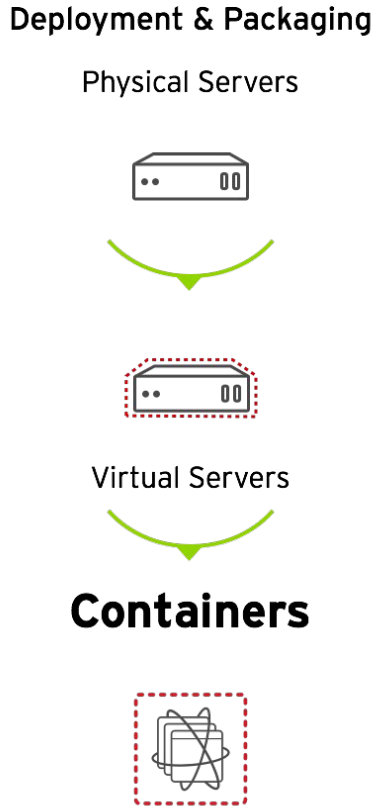
INFRASTRUCTURE

Modernize existing and build new cloud based infrastructure

Containers - Transform Apps, Infrastructure & Process



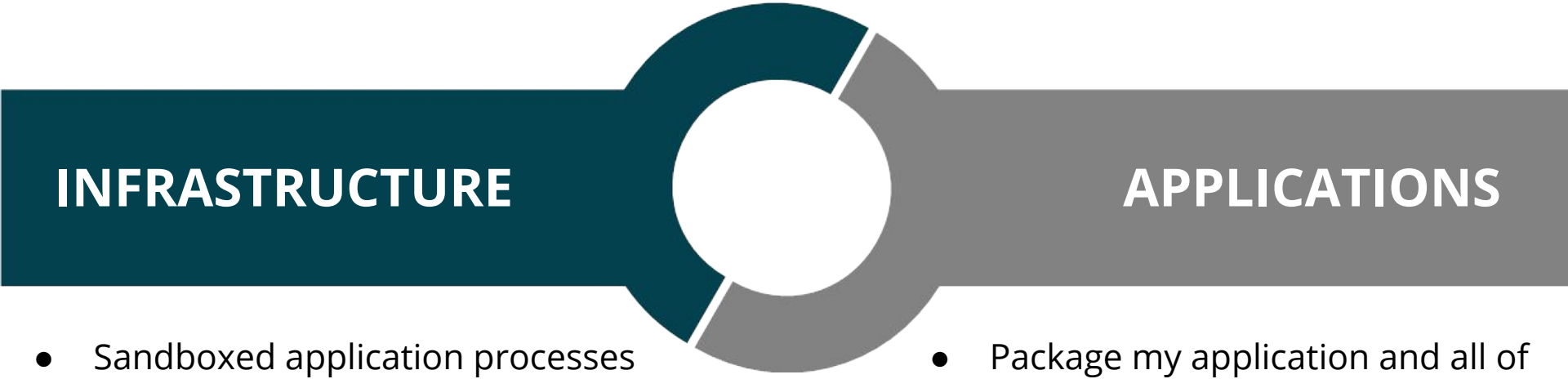
Containers - An Evolution in Application Deployment



- Enable efficiency and automation for microservices, but also support traditional applications
- Enable faster and more consistent deployments from Development to Production
- Enable application portability across 4 infrastructure footprints: Physical, Virtual, Private & Public Cloud

What Are Containers?

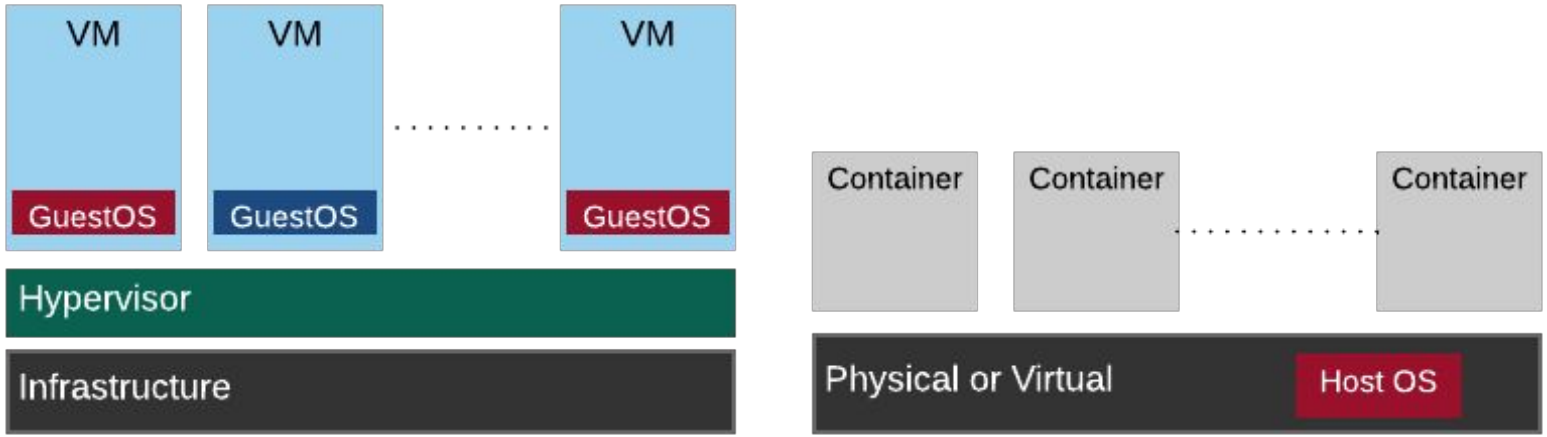
It Depends on Who You Ask



- Sandboxed application processes on a shared Linux OS kernel
- Simpler, lighter, and denser than virtual machines
- Portable across different environments

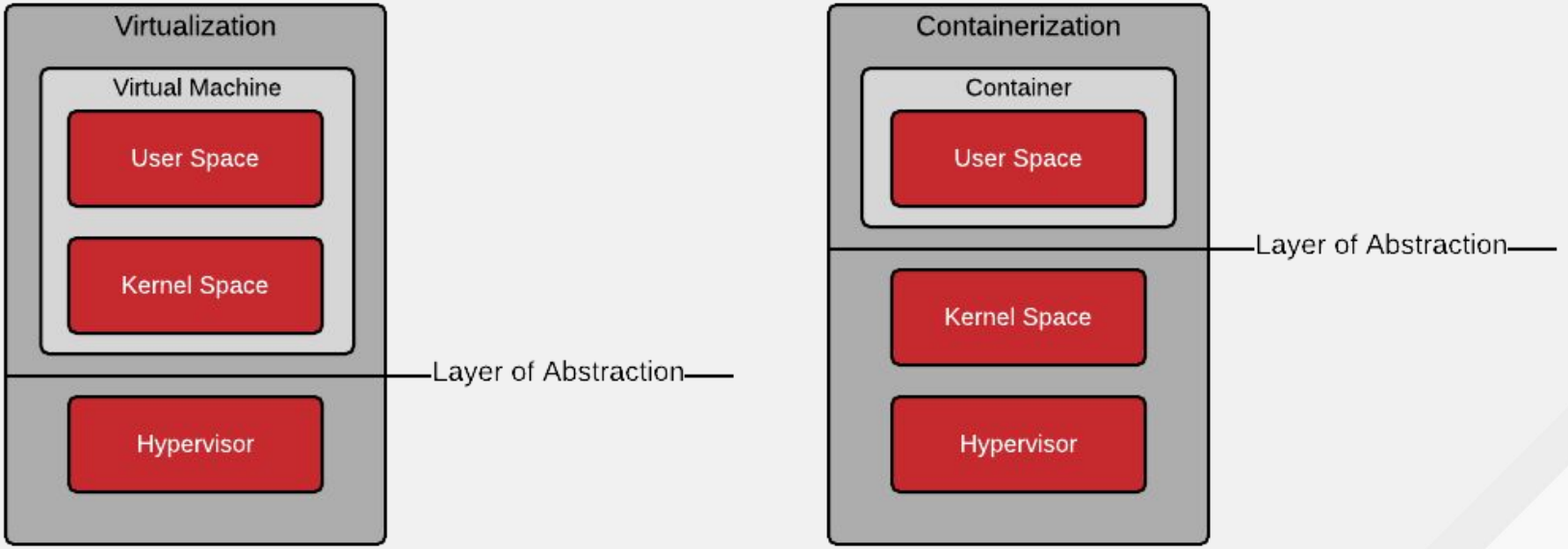
- Package my application and all of its dependencies
- Deploy to any environment in seconds and enable CI/CD
- Easily access and share containerized components

Understanding Containers

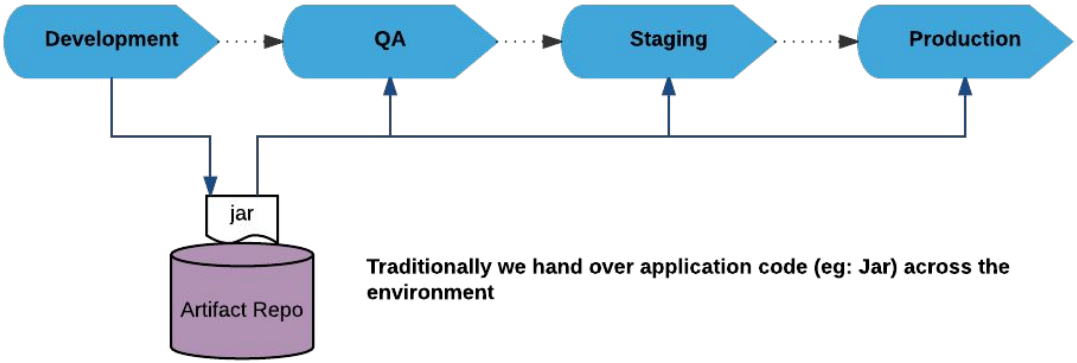


Virtual Machines vs Containers

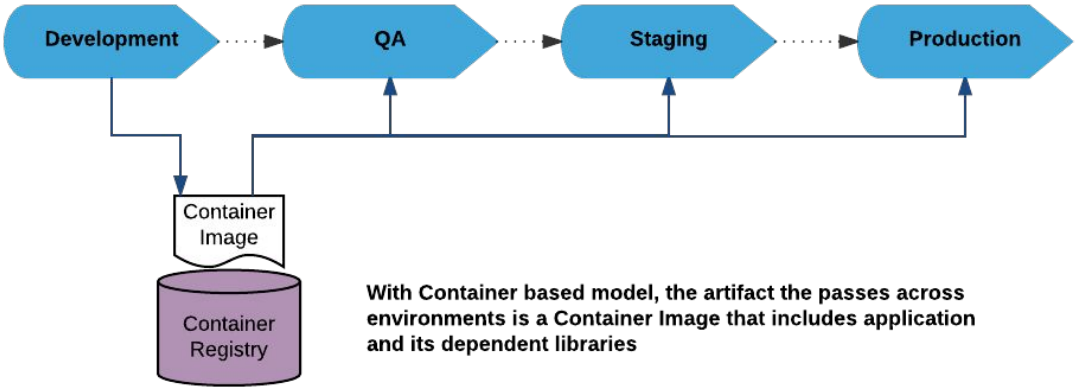
Virtualization vs Containerization



Infrastructure as Code in Containers

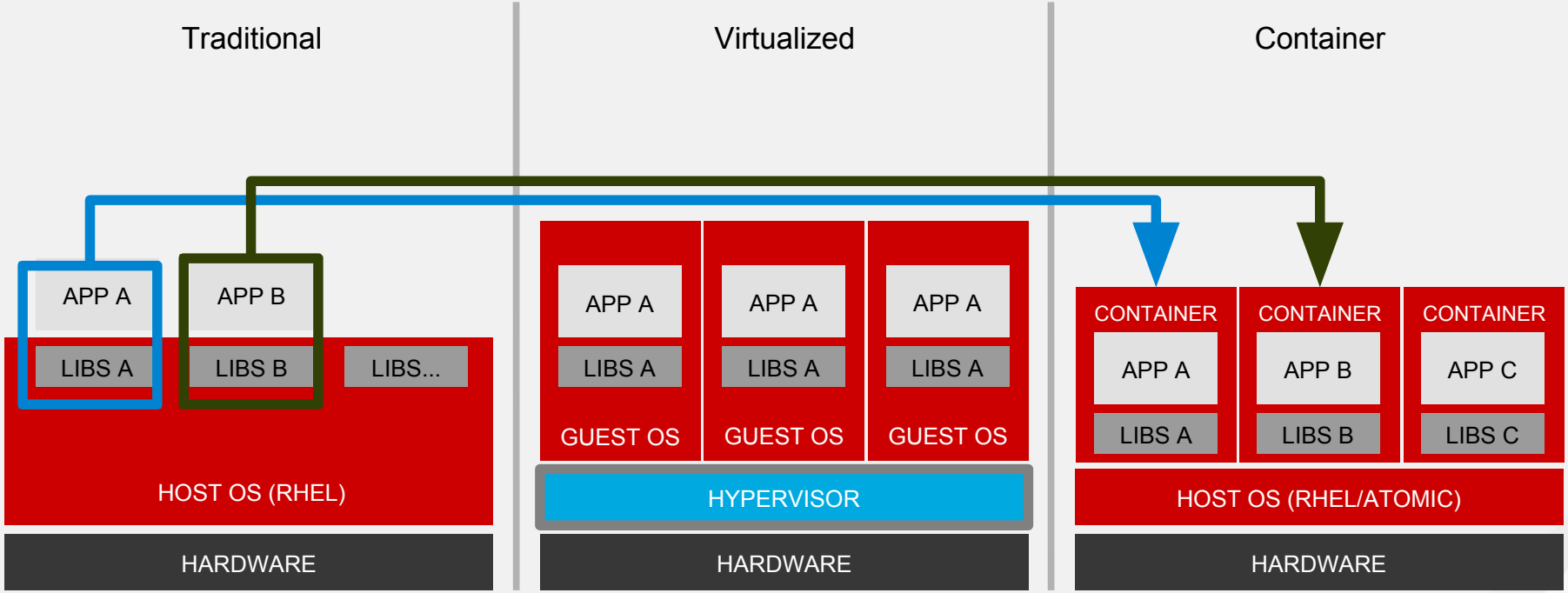


Traditionally we hand over application code (eg: Jar) across the environment



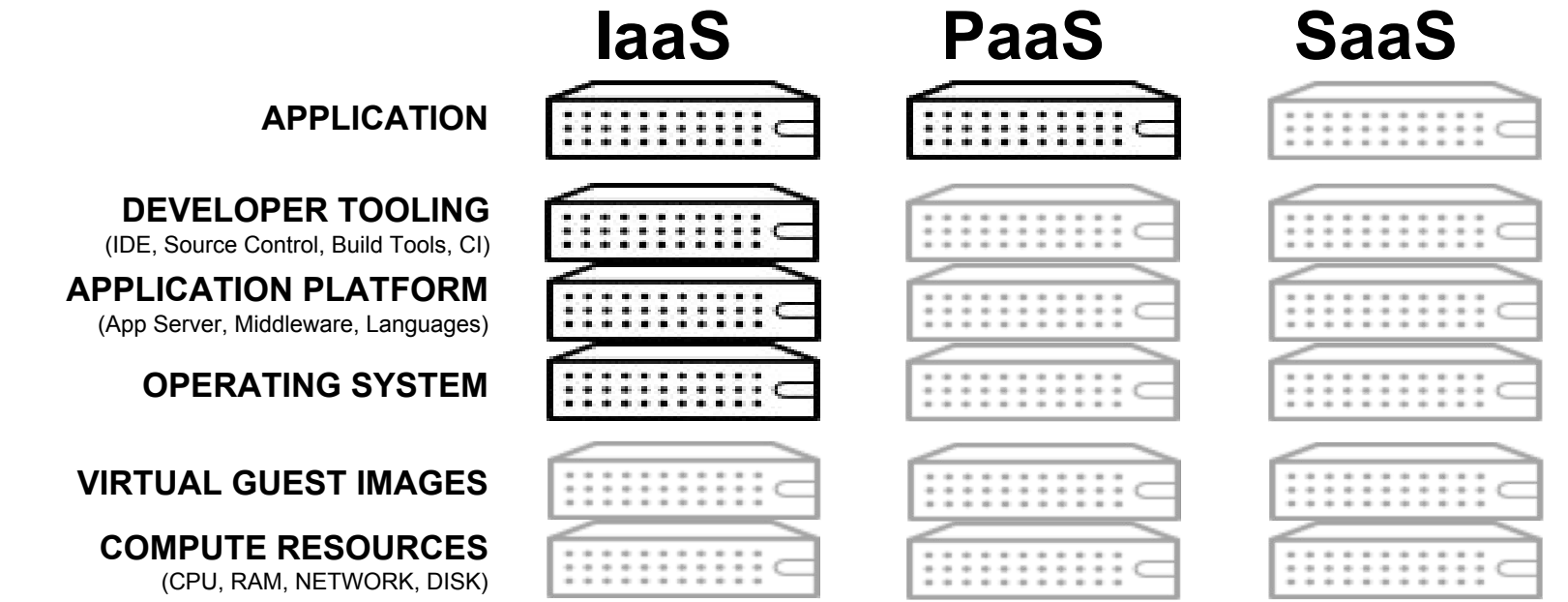
With Container based model, the artifact the passes across environments is a Container Image that includes application and its dependent libraries

TRADITIONAL VS. VIRTUALIZED VS. CONTAINERS

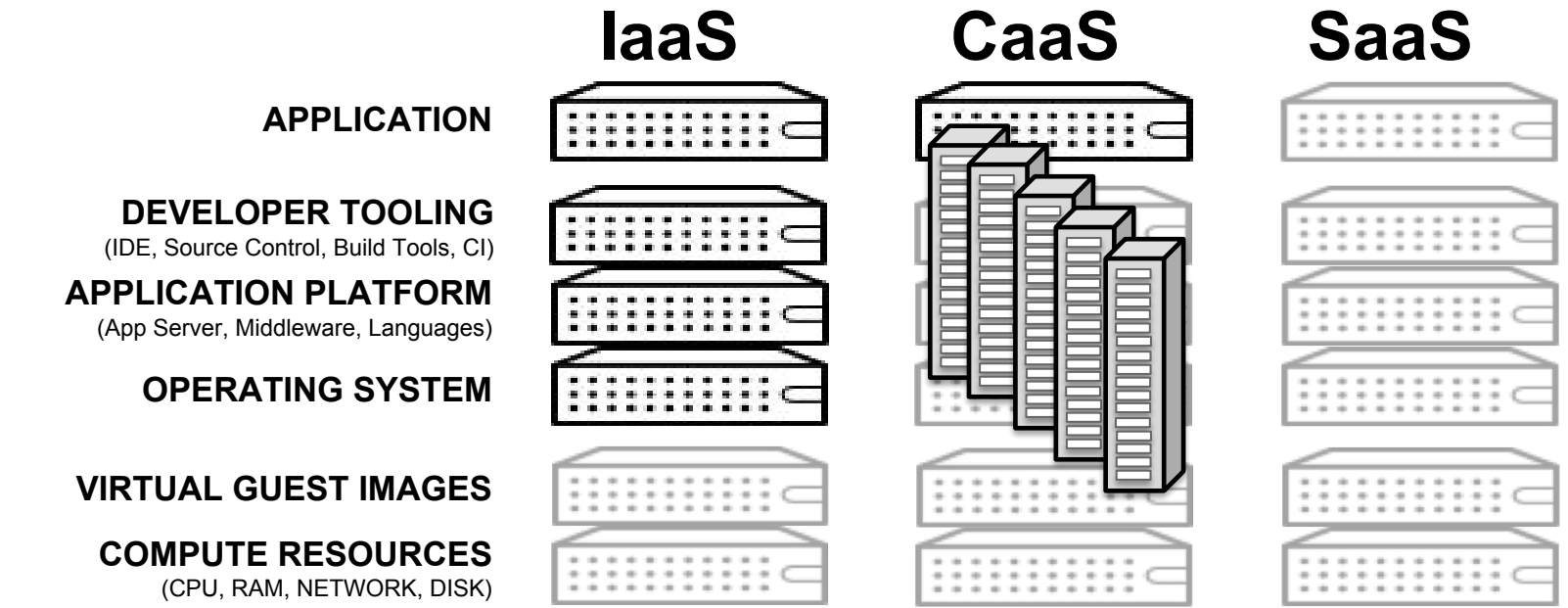


Making Containers Enterprise Consumable

Cloud Service Models

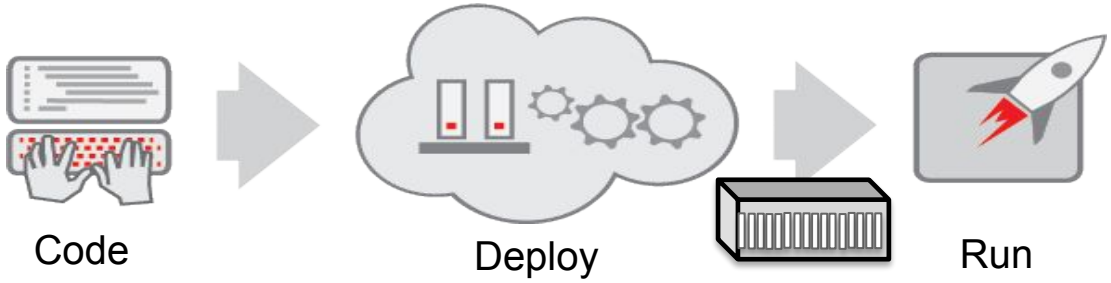


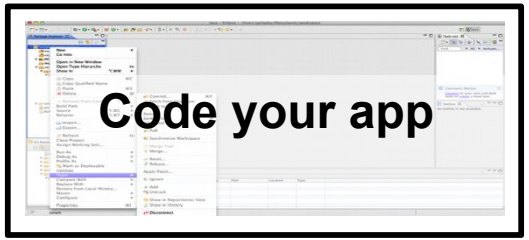
Cloud Service Models

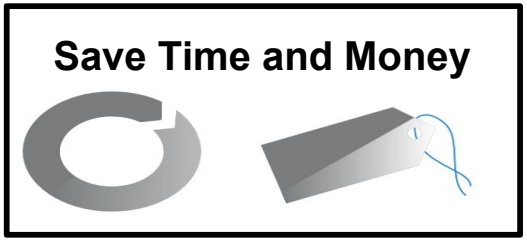


PaaS + CaaS

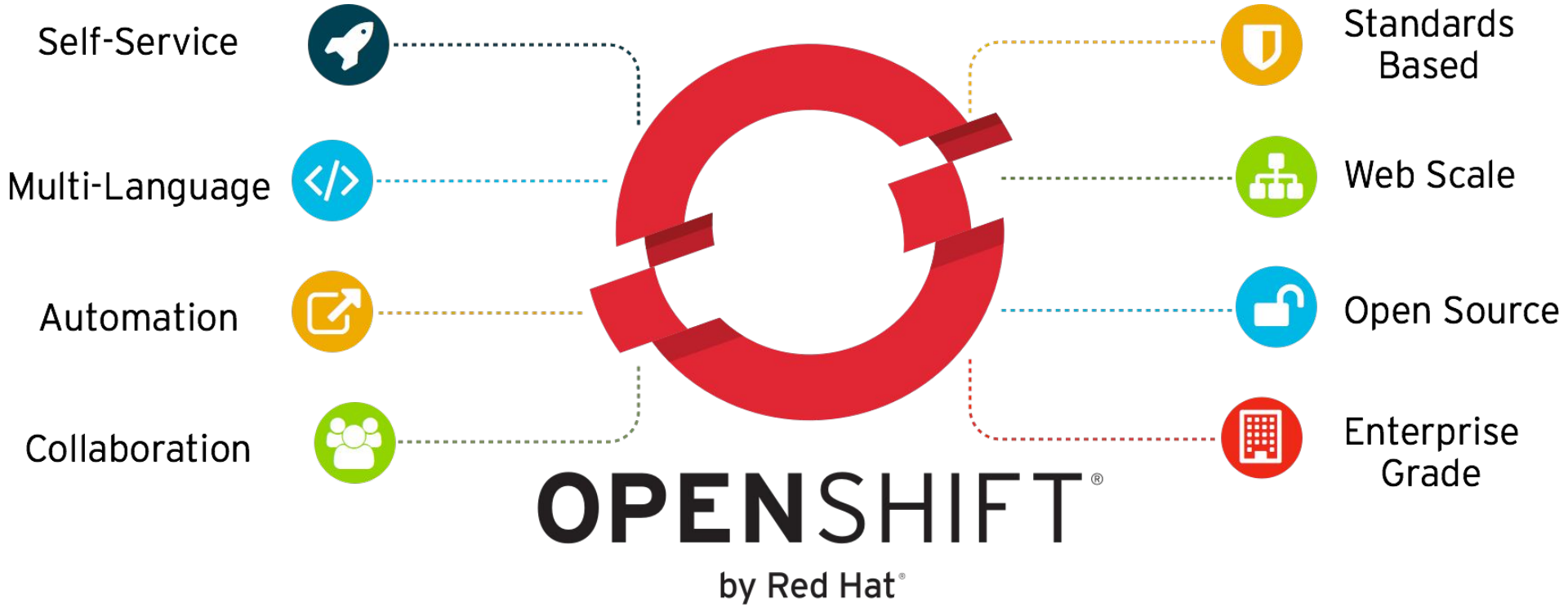
A Distributed Cloud Application Platform



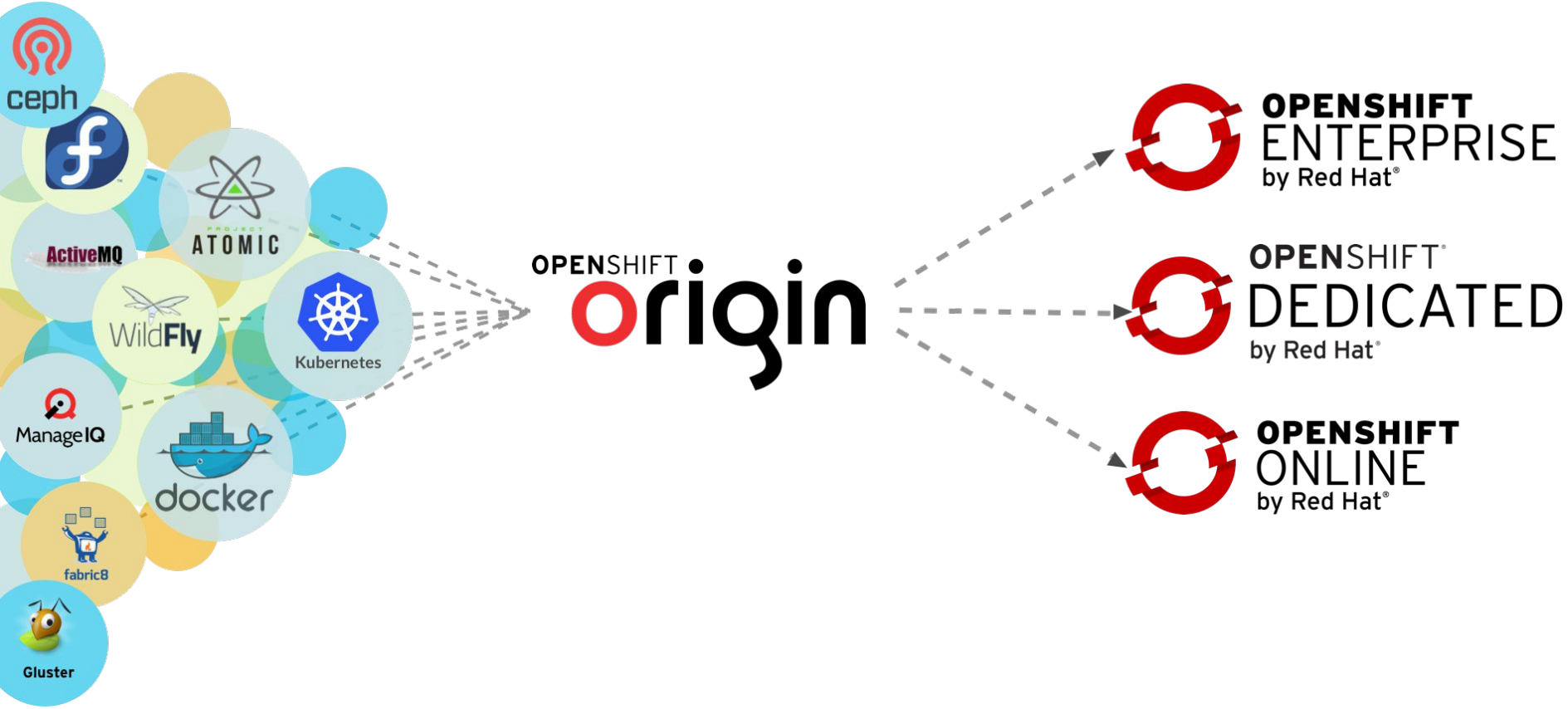




Born in 2011

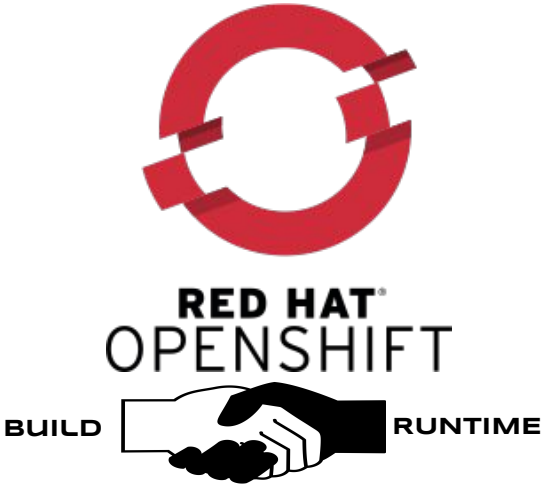

Self-Service
Multi-Language
Automation
Collaboration
Standards Based
Web Scale
Open Source
Enterprise Grade
OPENSHIFT®
by Red Hat®

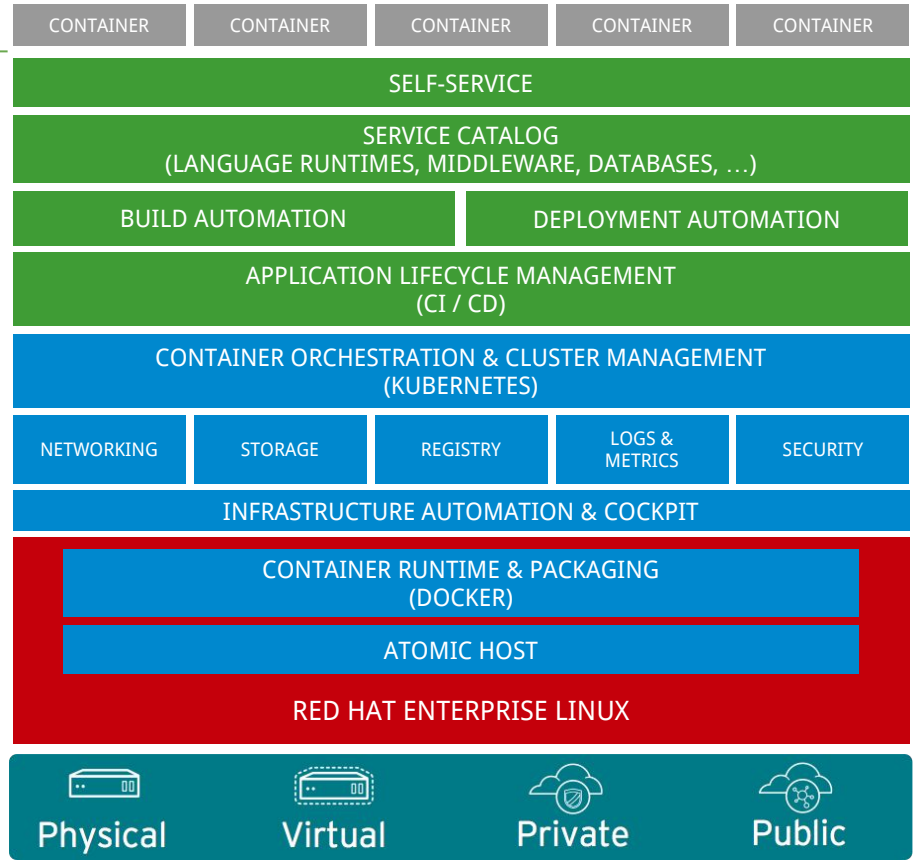
Community Powered Innovation



OpenShift Container Platform - Stack

Build, Deploy and Manage Containerized Apps





Application Services

Still needs work

NGINX
Juniper Networks
Crunchy DB
& More

OpenShift
Primed

FROM ISVs

JBoss Fuse
JBoss AM-Q
JBoss Data Virt
3Scale

JBoss EAP
JBoss Web Server
JBoss Data Grid
JBoss Dev Studio
JBoss Ops Network
Red Hat Mobile

JBoss BRMS
JBoss BPM Suite

Red Hat
Storage

Red Hat Software
Collections
Red Hat Container
Catalog
Red Hat
SSO

FROM RED HAT

Your
Private
Registries

Docker Hub

FROM THE COMMUNITY

redhat.

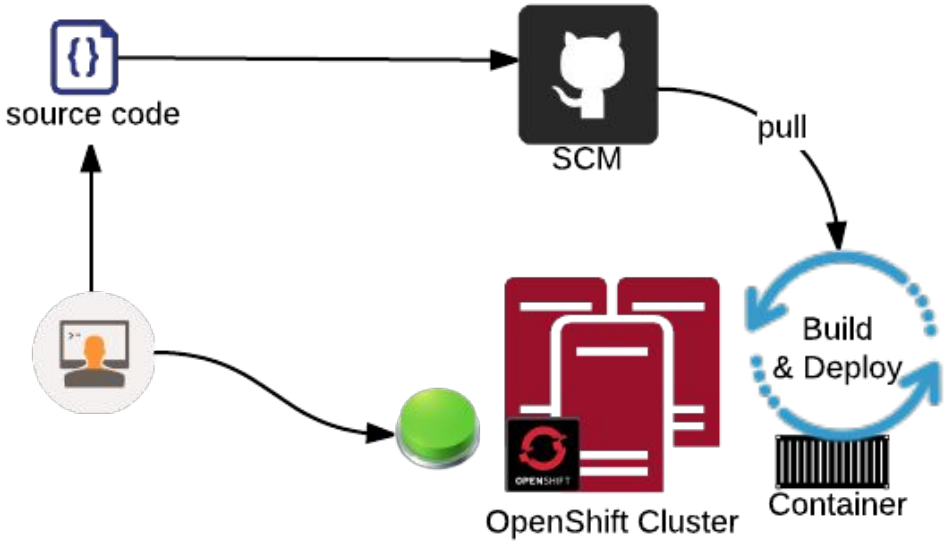
Facilitating a rich container ecosystem

Represented by a broad coalition of industry leaders focused on common standards for software containers

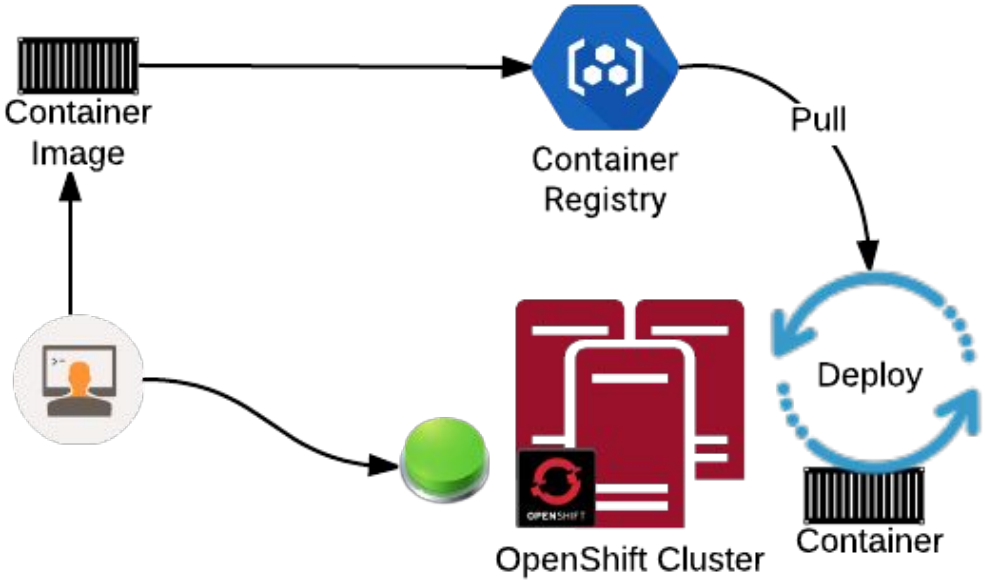
Create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems


OPENSHIFT COMMONS
An interactive community for all OpenShift Users, Customers, Contributors, Partners, Service Providers and Developers to share ideas, code, best practices, and experiences.
More at http://commons.openshift.org

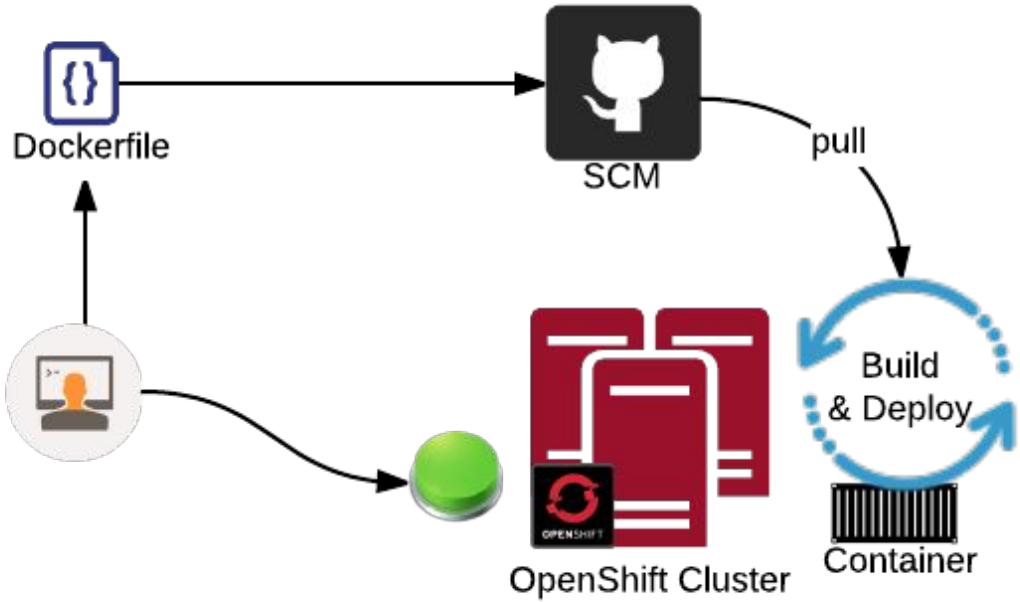
Start with Source Code - S2I



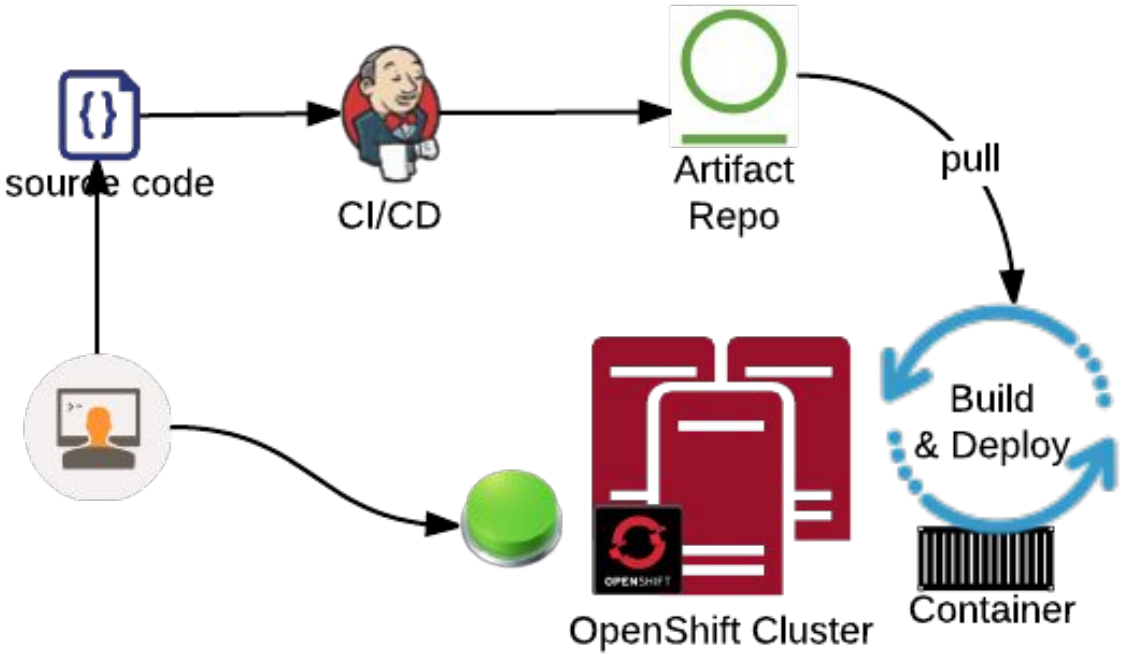
Deploy a Container Image



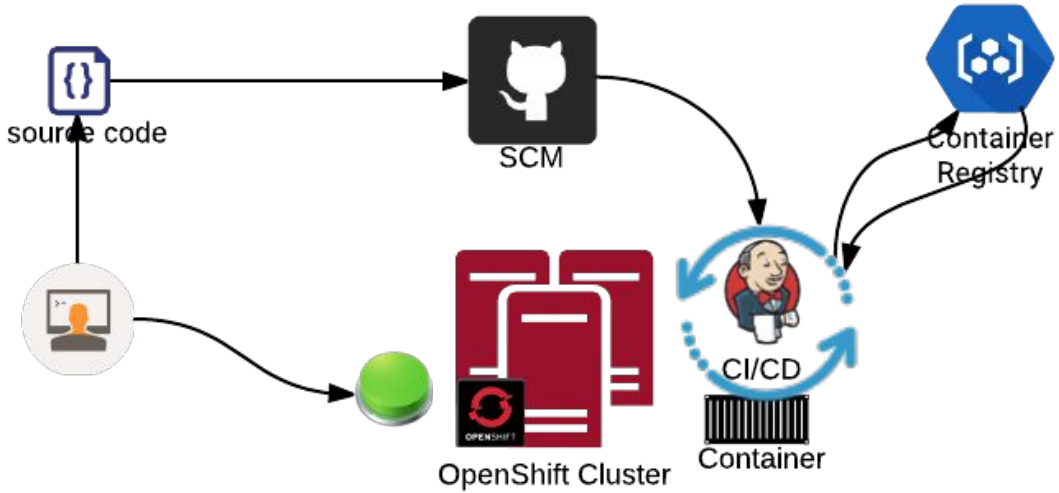
Deploy a Dockerfile



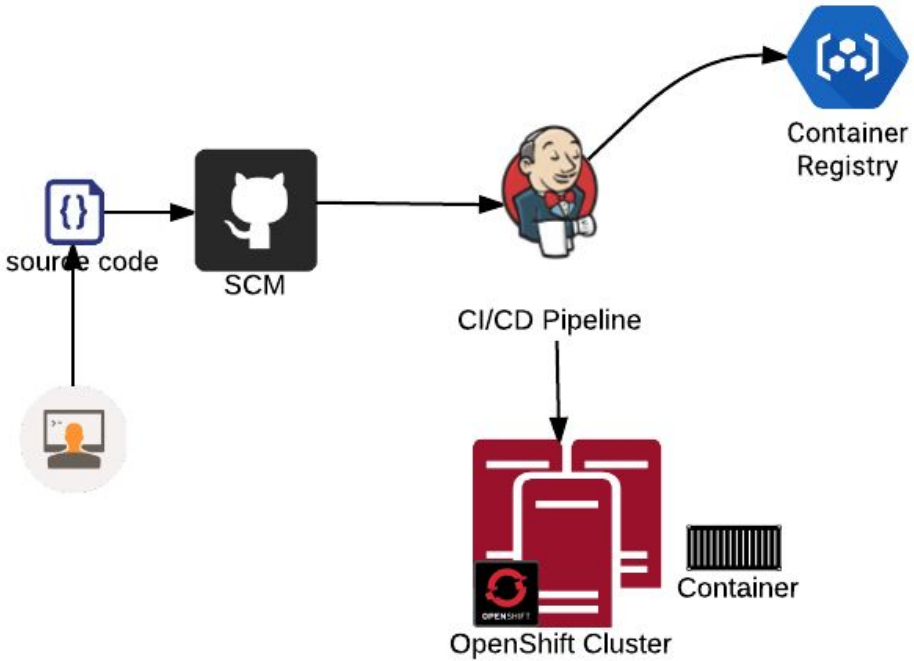
Deploy a Binary Artifact



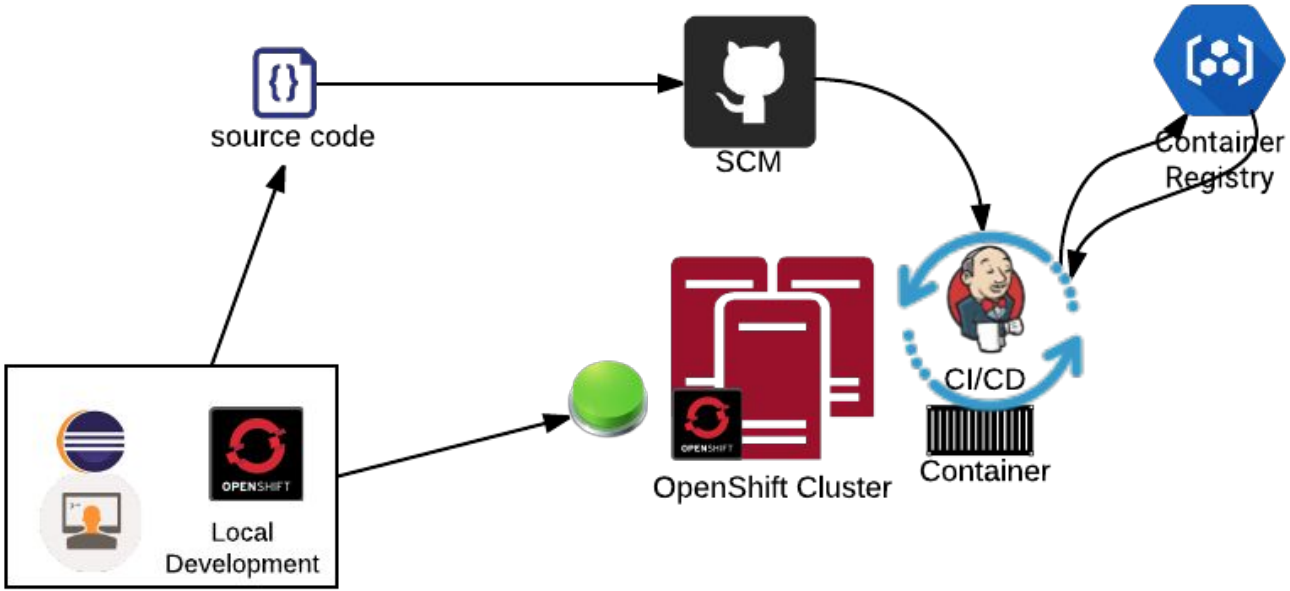
Run a CICD Pipeline on OpenShift Cluster



Use your existing CICD Pipeline to Deploy to OpenShift

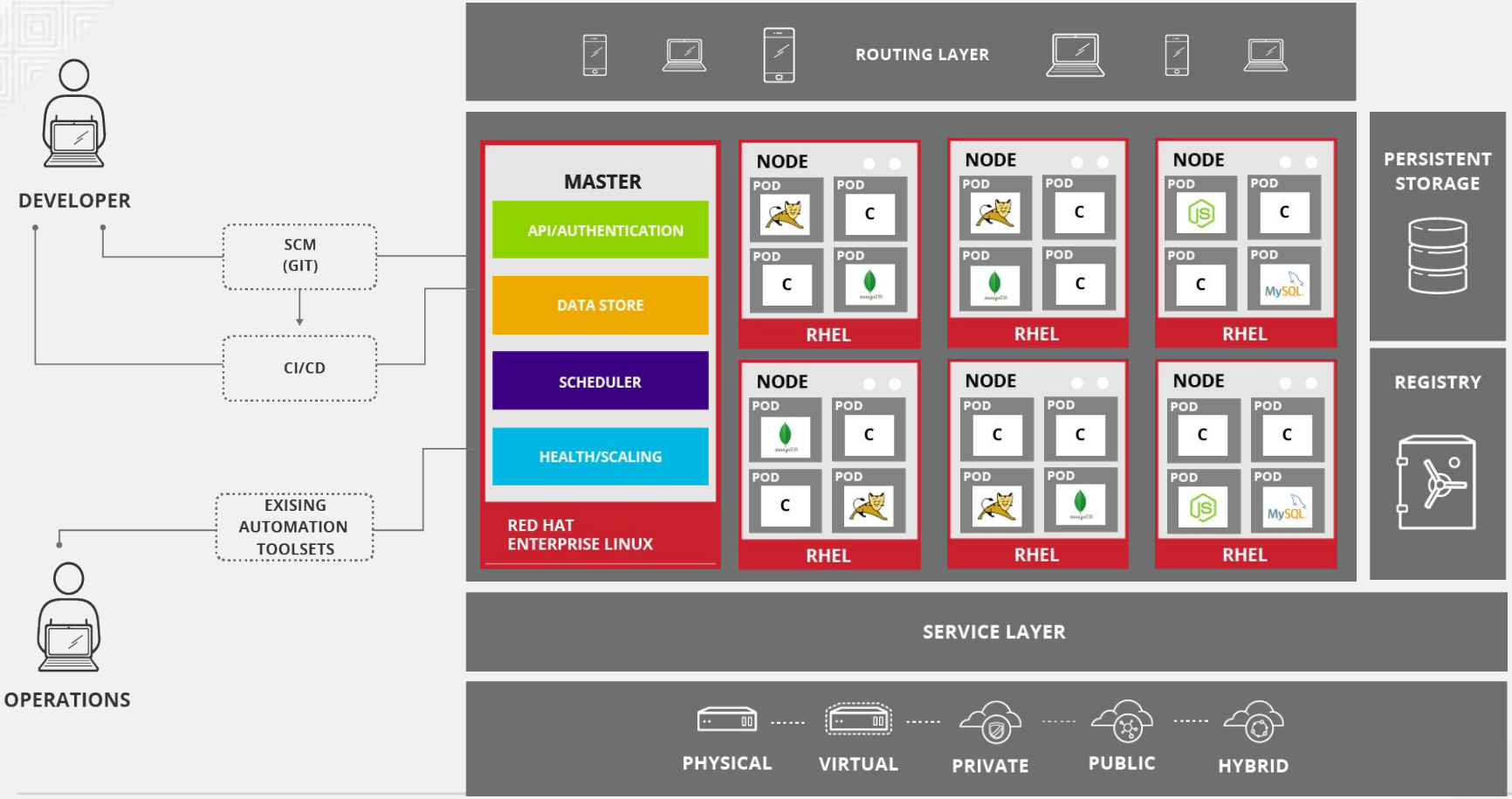


Build locally with an all-in-one cluster on your Workstation



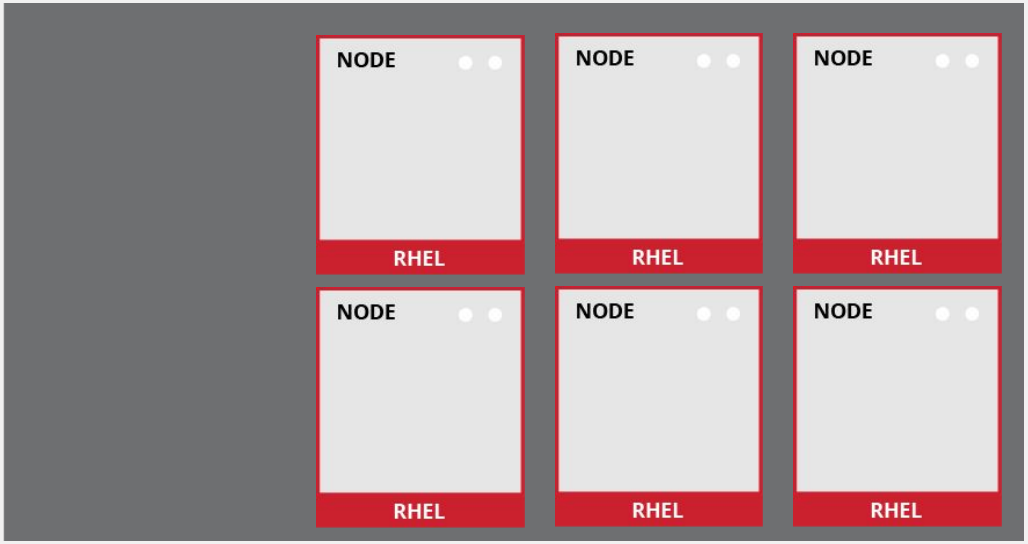

OpenShift
Architecture Overview
redhat.

10,000 foot overview



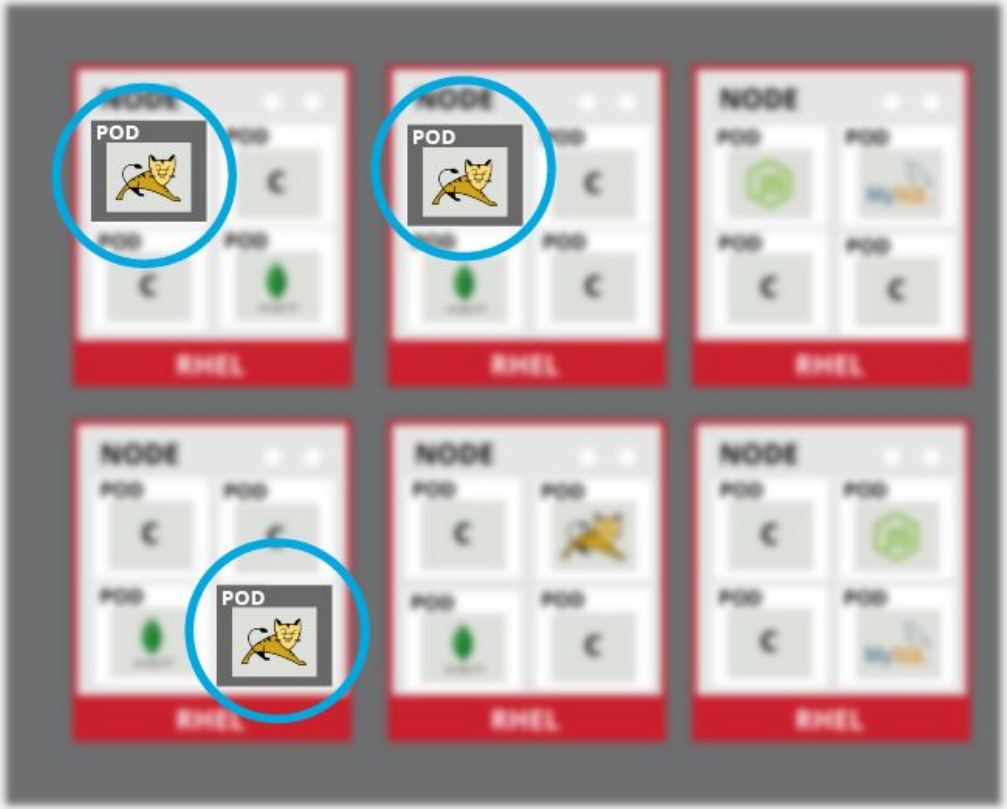
OpenShift runs on your choice of infrastructure

Nodes are instances of RHEL where apps will run

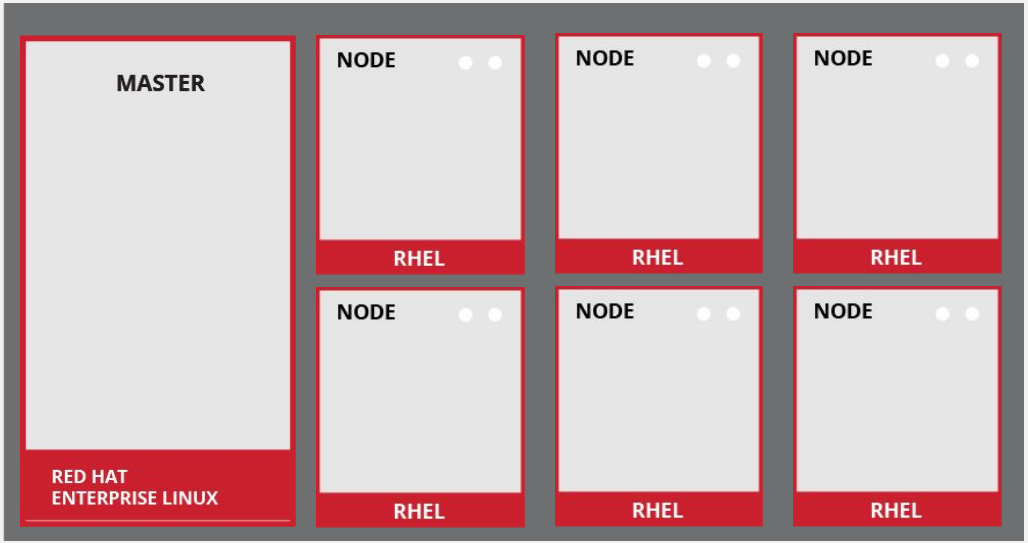


Apps and components run in containers

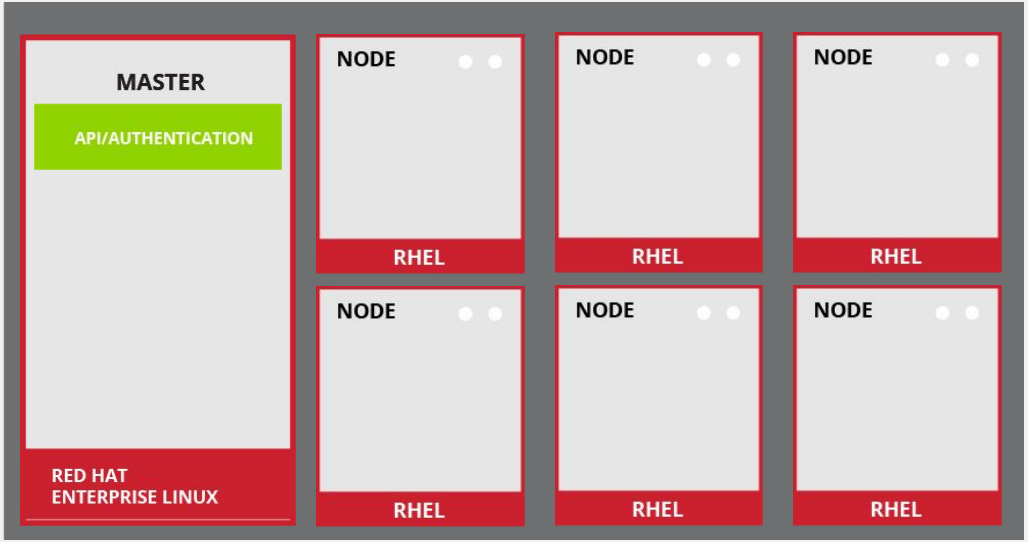
Pods are the orchestrated unit in OpenShift



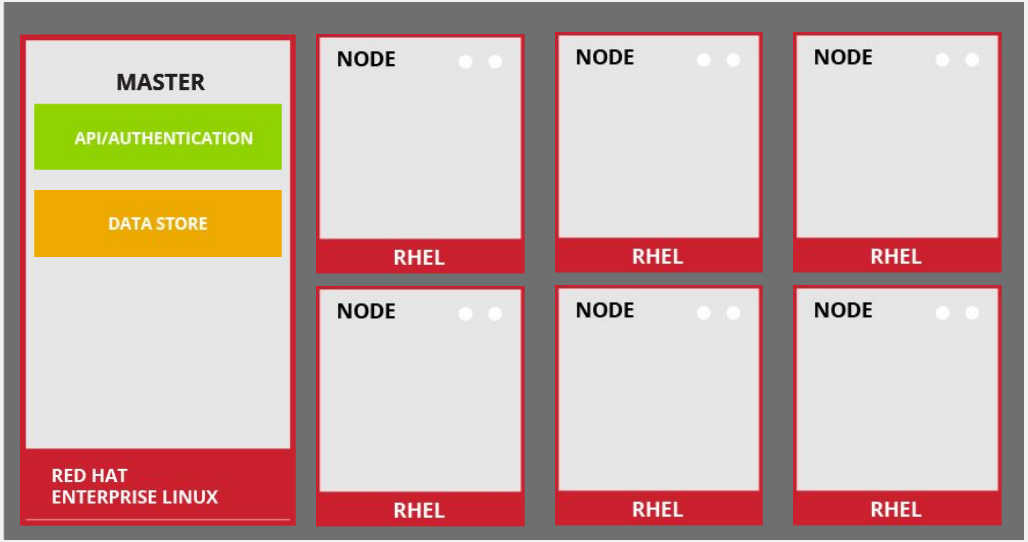
Masters are the Control Plane



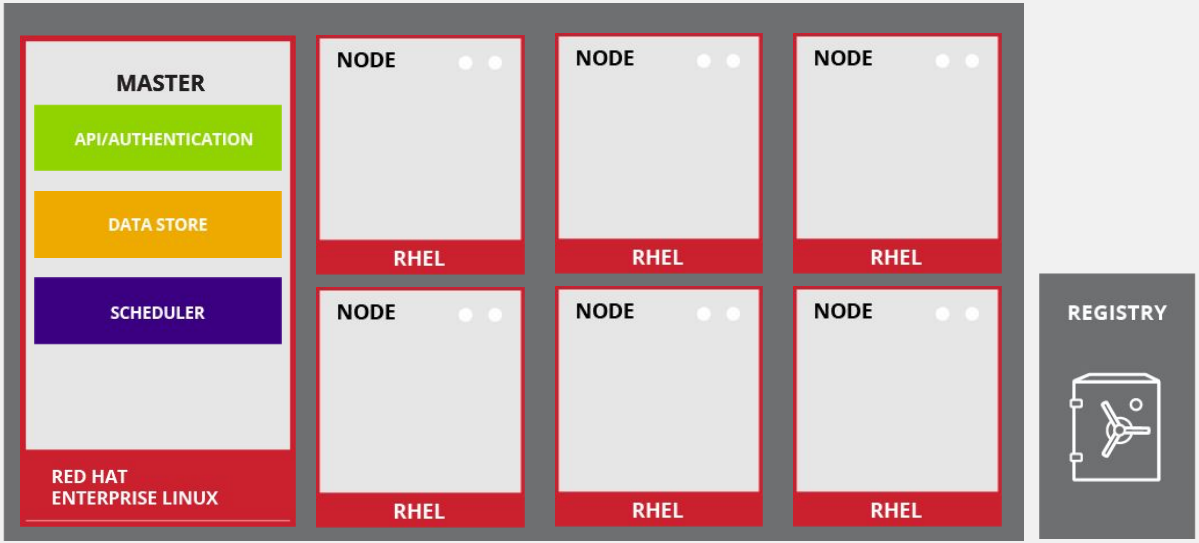
API and Authentication



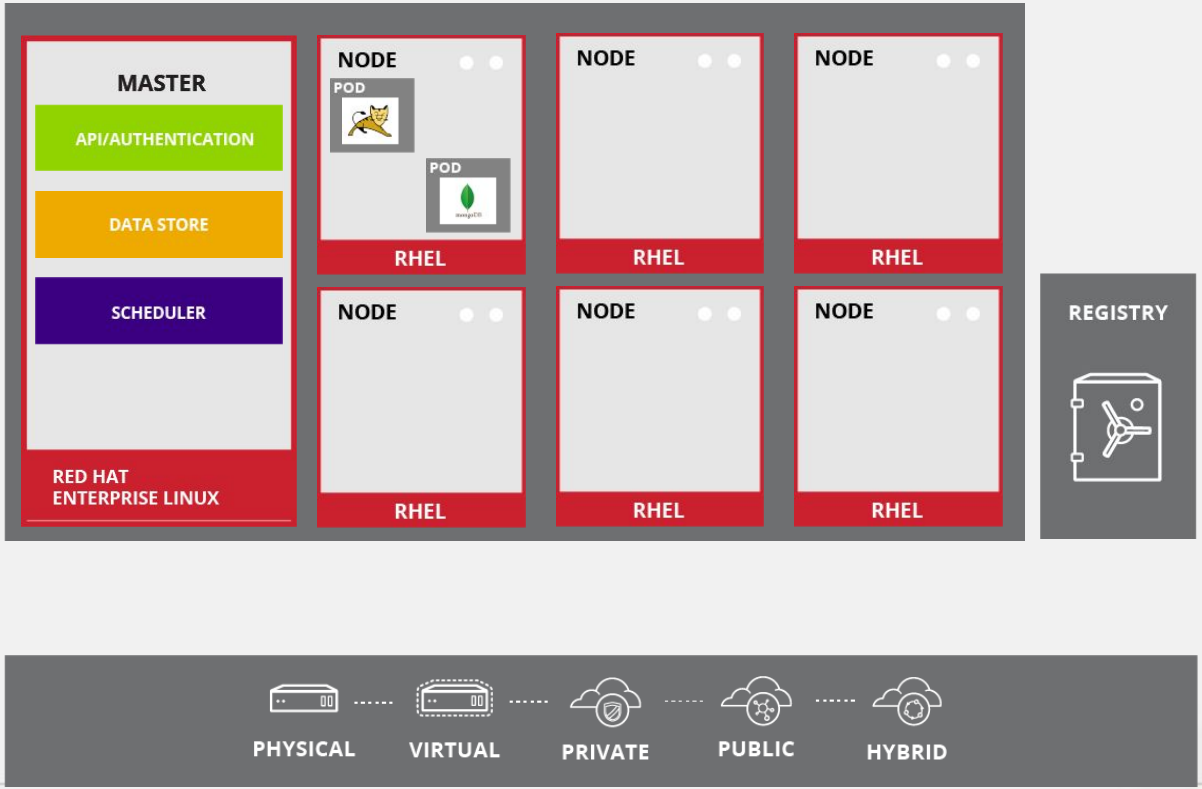
Desired and Current State



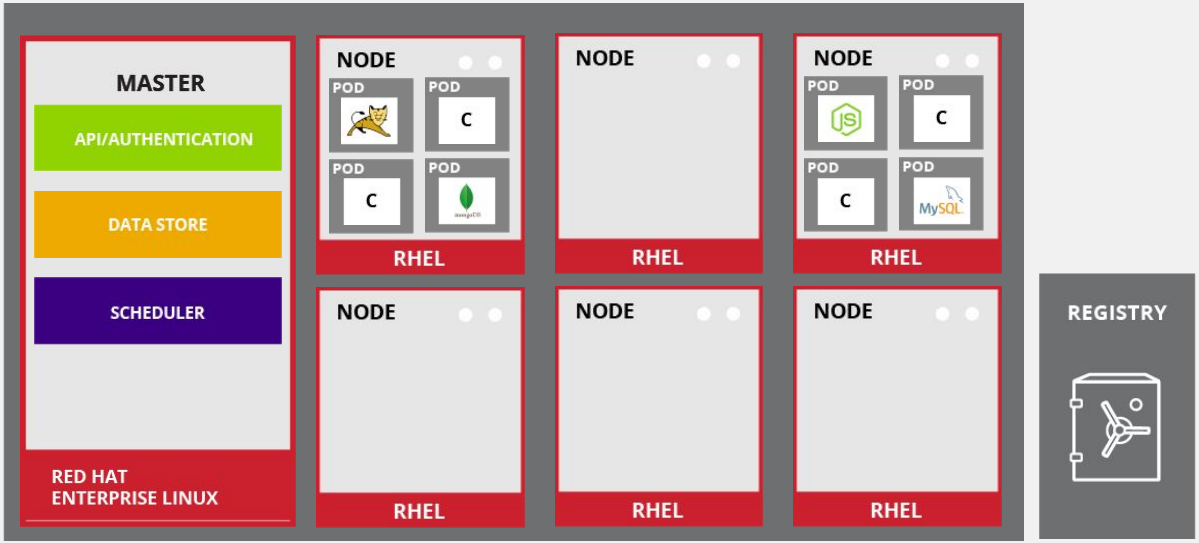
Scheduler Pulls From The Registry



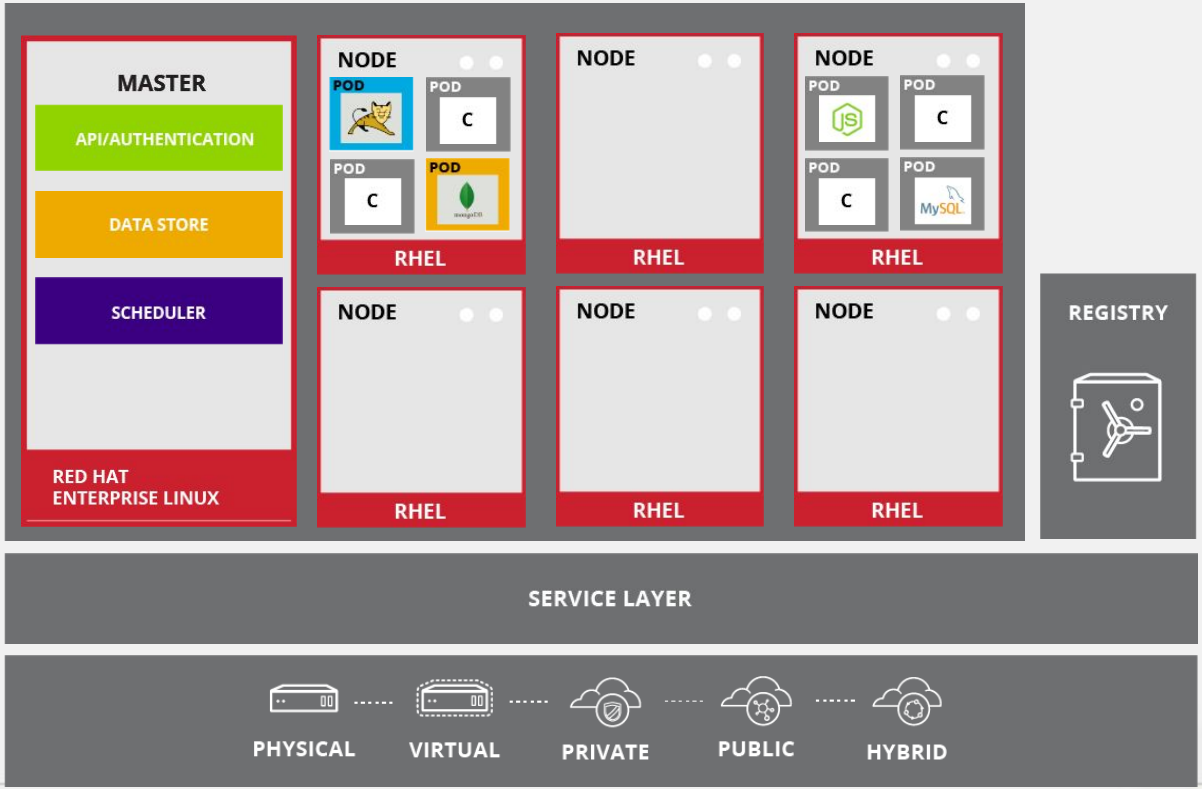
Orchestration and Scheduling



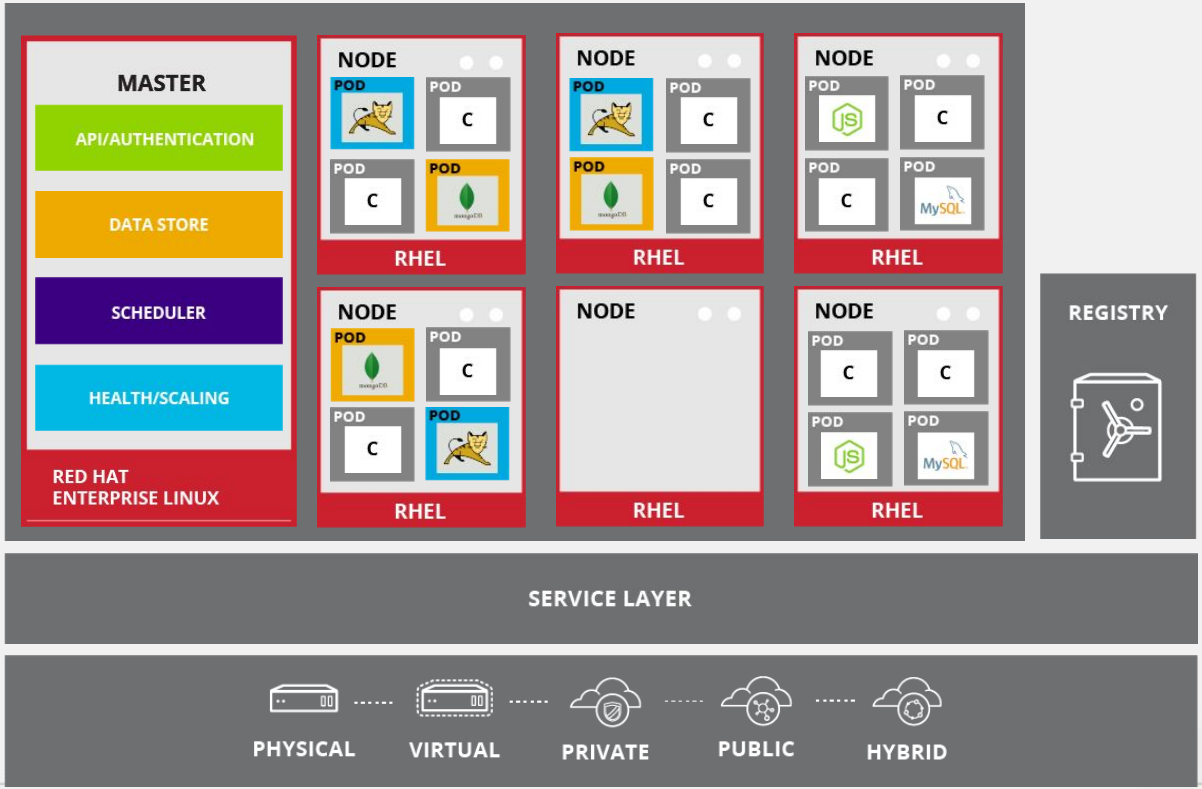
Placement by Policy



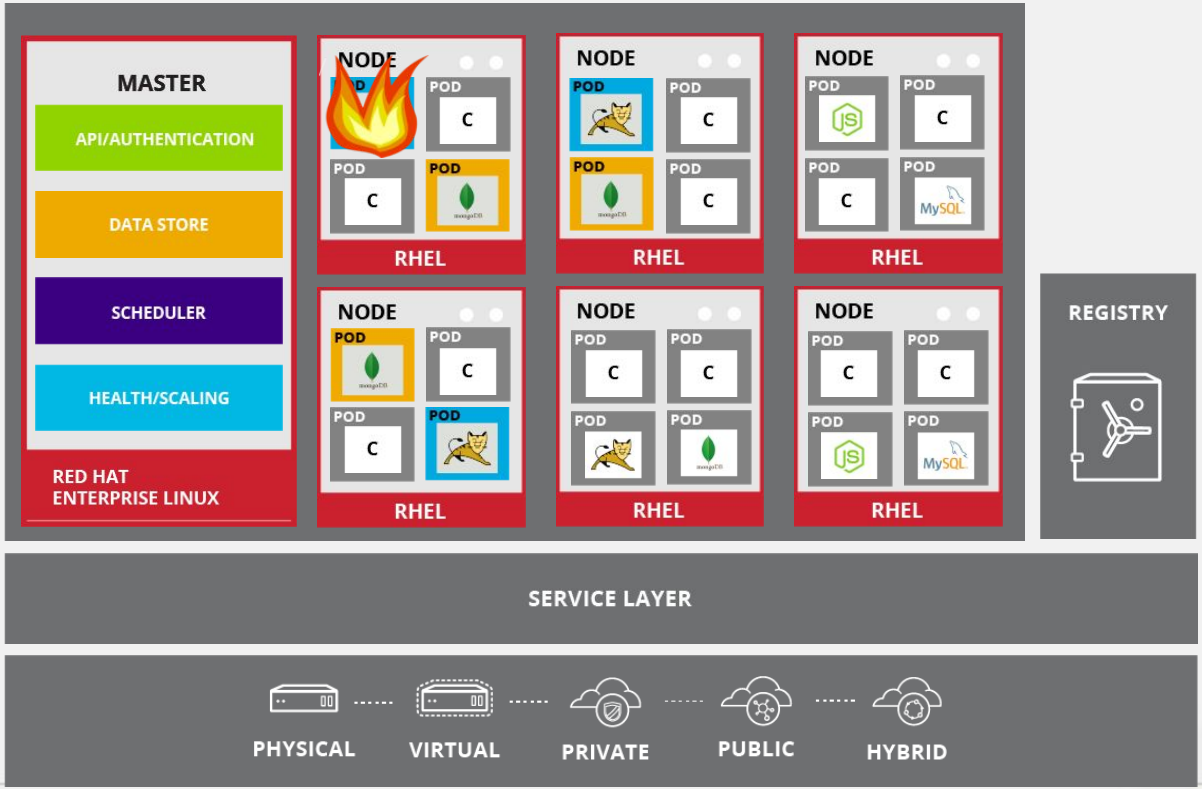
Services connect application components



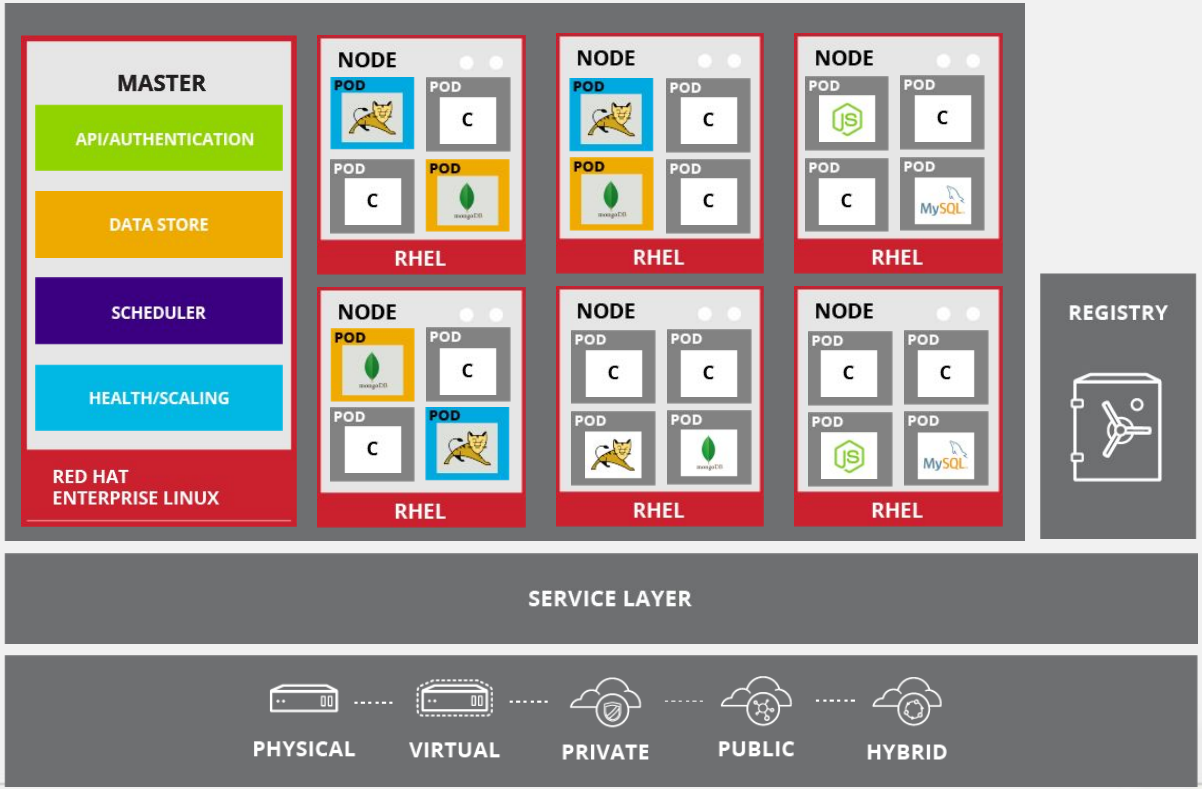
Health and Scaling



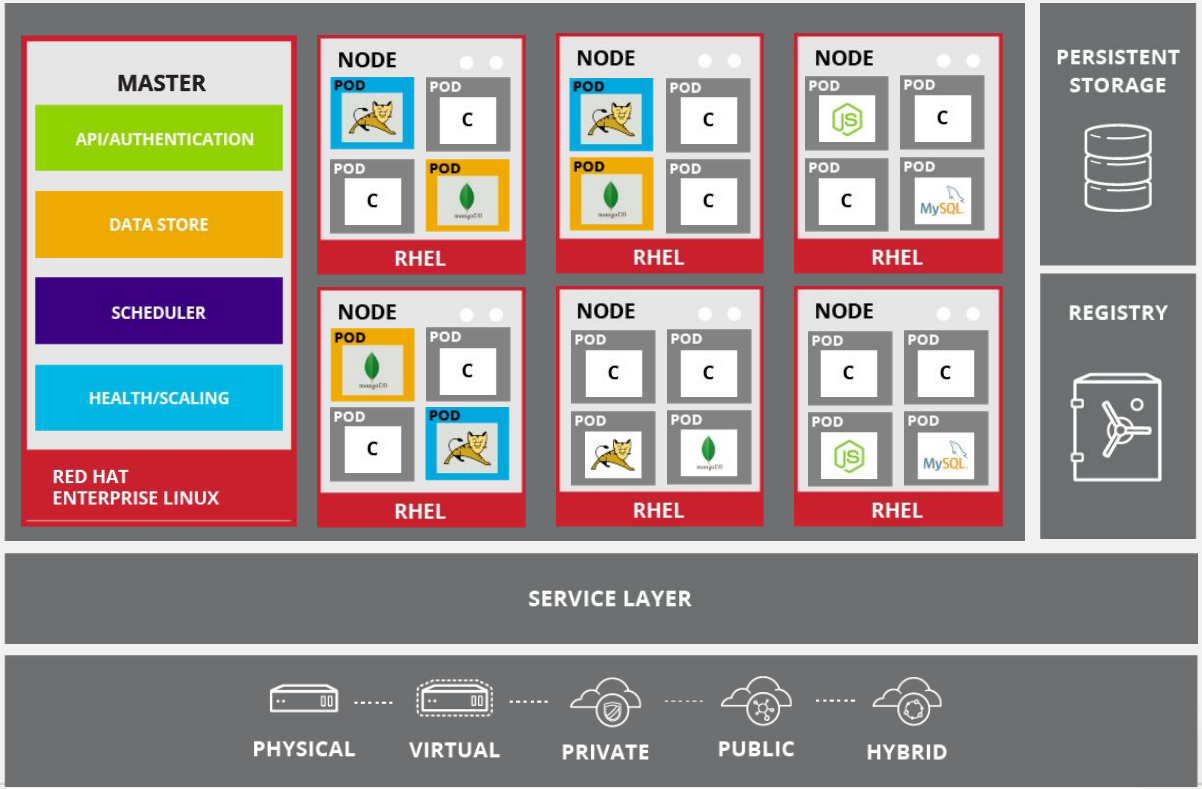
What about unhealthy Pods?



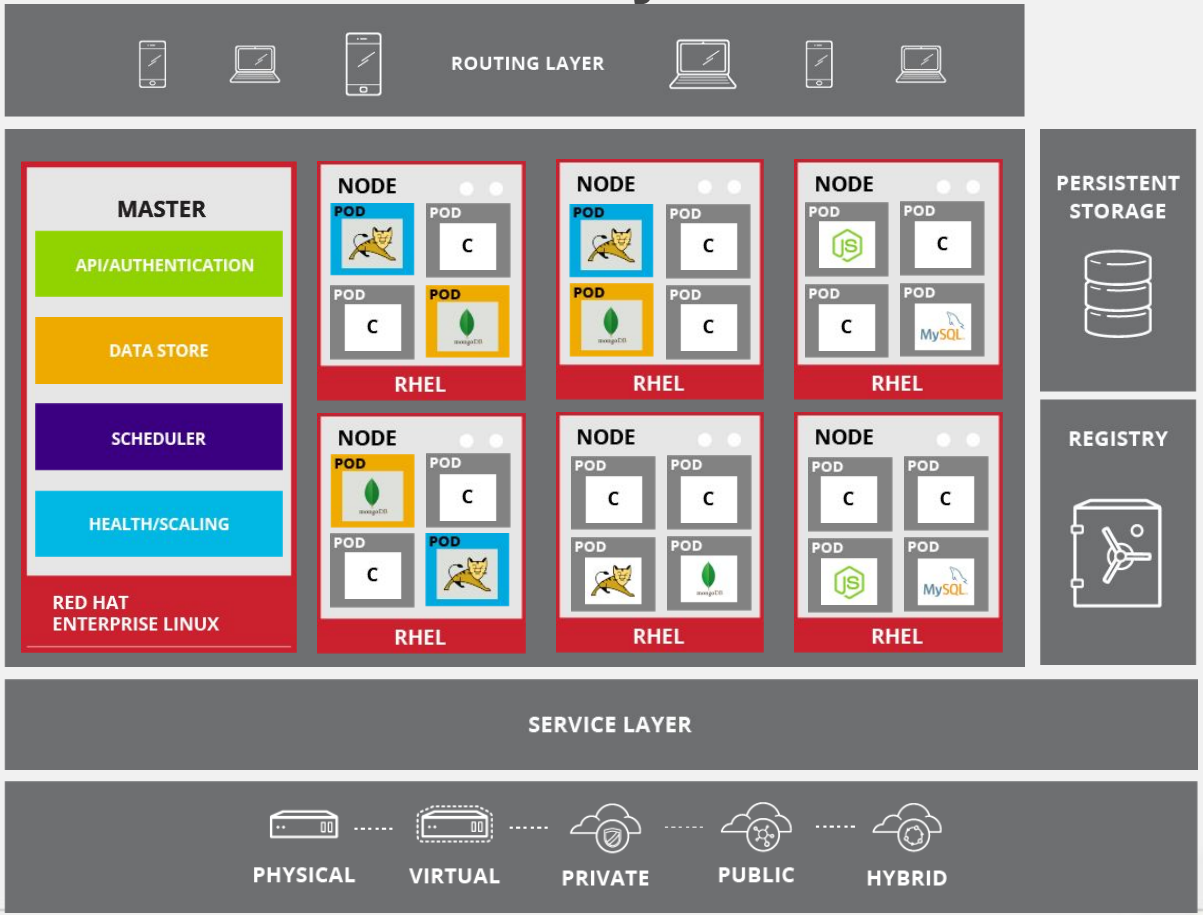
The Master remediates Pod failures



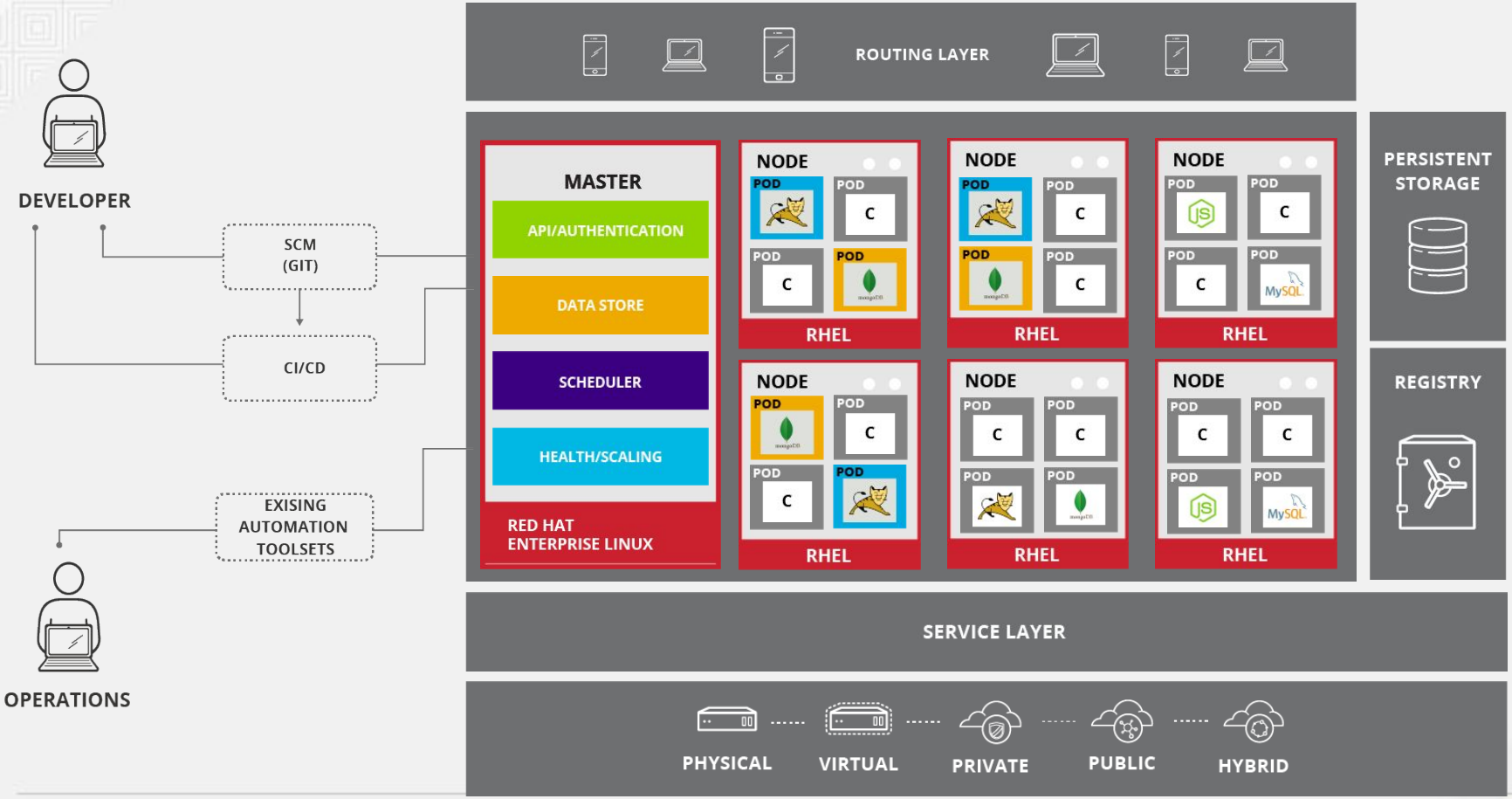
What about app data?



Routing layer for external accessibility

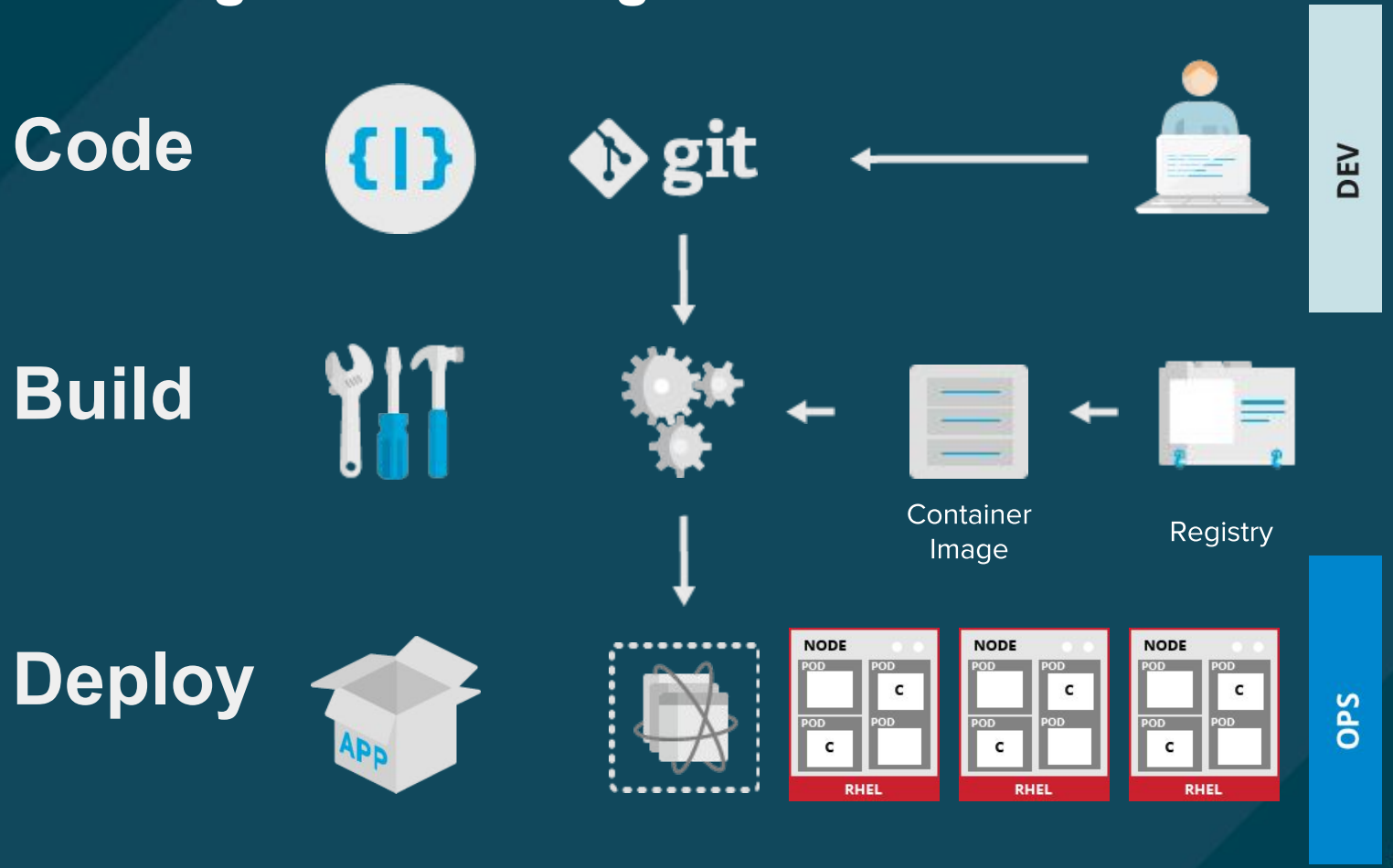


Access via Web UI, CLI, IDE, API




OpenShift
Source-to-Image
redhat.

Source 2 Image Walk Through



Source 2 Image Walk Through

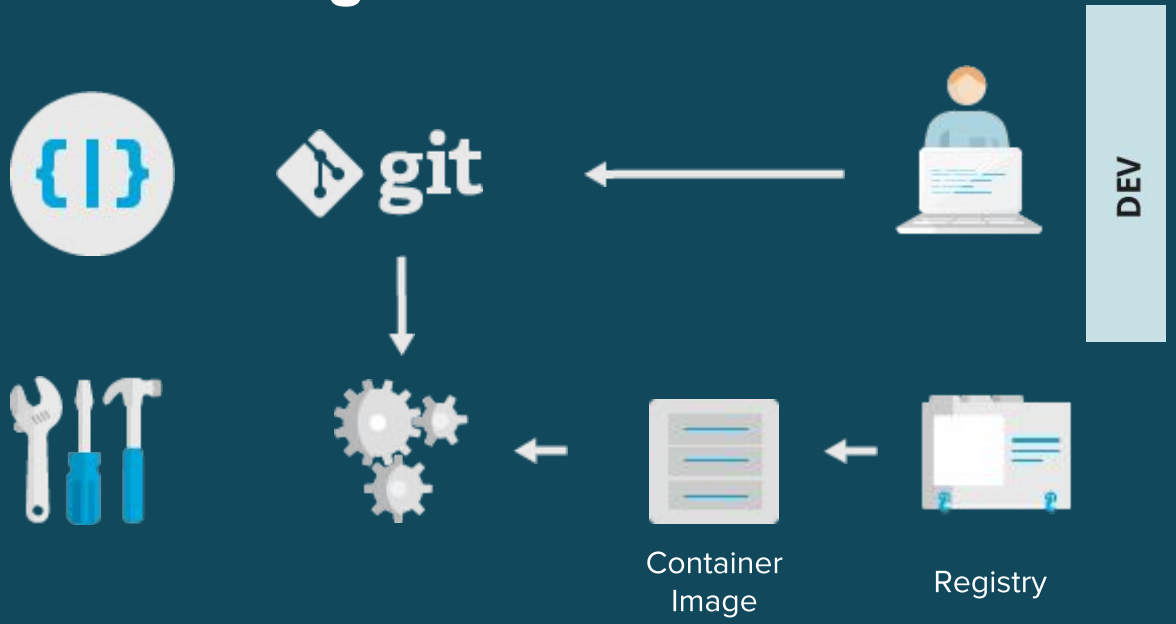
Code

Developers can leverage existing development tools and then access the OpenShift Web, CLI or IDE interfaces to create new application services and push source code via GIT. OpenShift can also accept binary deployments or be fully integrated with a customer's existing CI/CD environment.

Source 2 Image Walk Through

Build

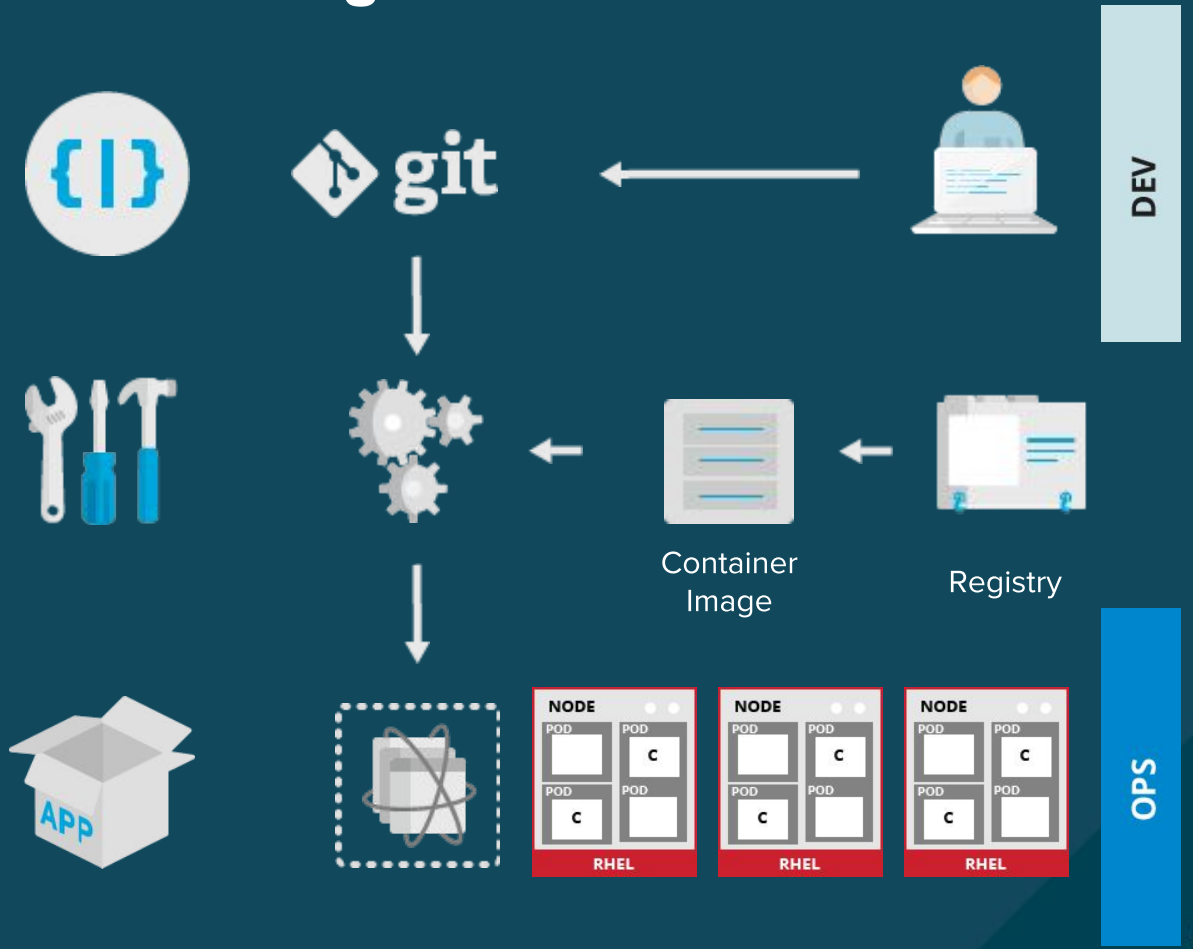
OpenShift automates the Docker image build process with Source-to-Image (S2I). S2I combines source code with a corresponding Builder image from the integrated Docker registry. Builds can also be triggered manually or automatically by setting a Git webhook. Add in Build pipelines



Source 2 Image Walk Through

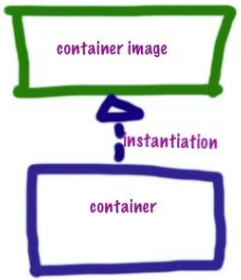
Deploy

OpenShift automates the deployment of application containers across multiple Node hosts via the Kubernetes scheduler. Users can automatically trigger deployments on application changes and do rollbacks, configure A/B deployments & other custom deployment types.



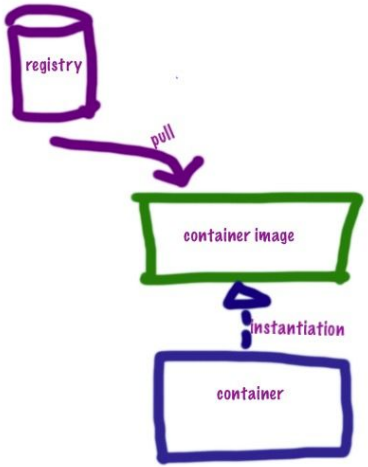
redhat

Backup
K8S concepts



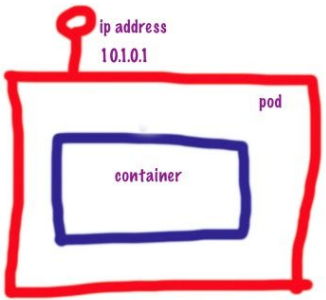
You instantiate a container image to get a container..

Just like object instantiated from a Class



So where does the container image come from?

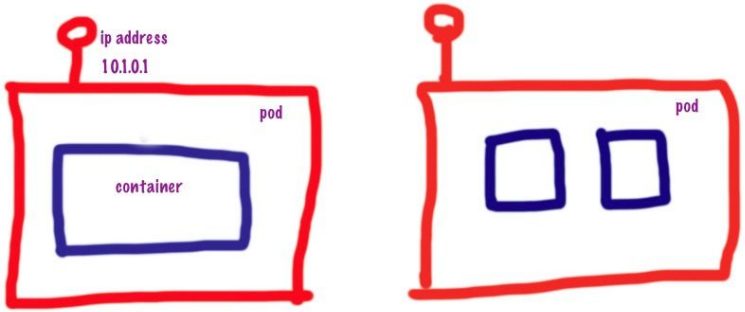
Images are stored in a Container Registry



OpenShift/K8s puts your container inside a Pod.

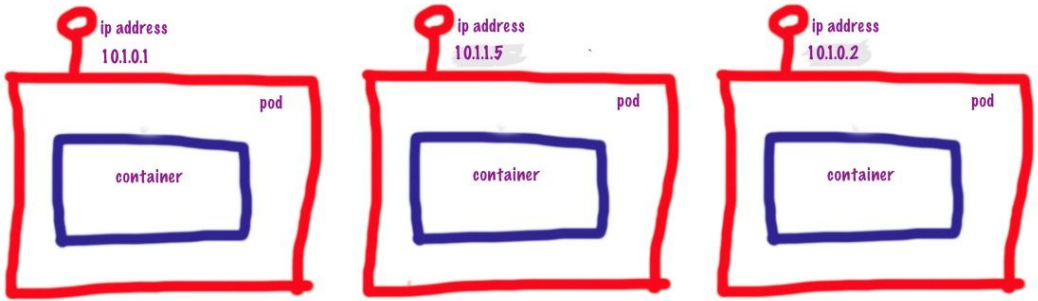
Pod is a wrapper. Gets an ip address.

Container adopts Pod's ip

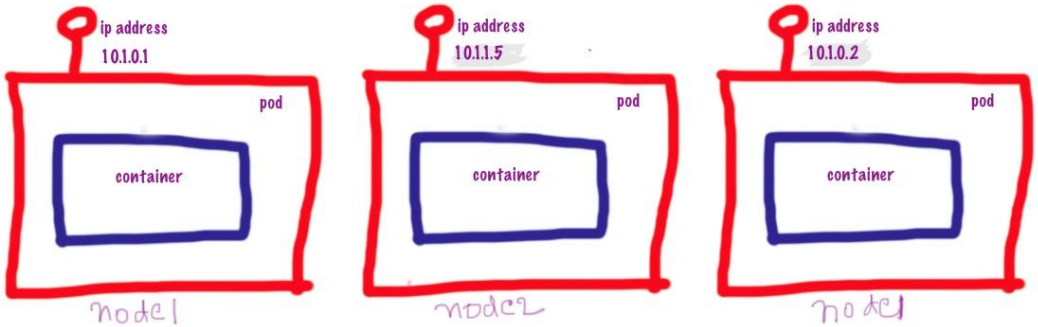


Some pods may have more than one container.. that's a special case though!!

All the containers in a pod die along with a pod.

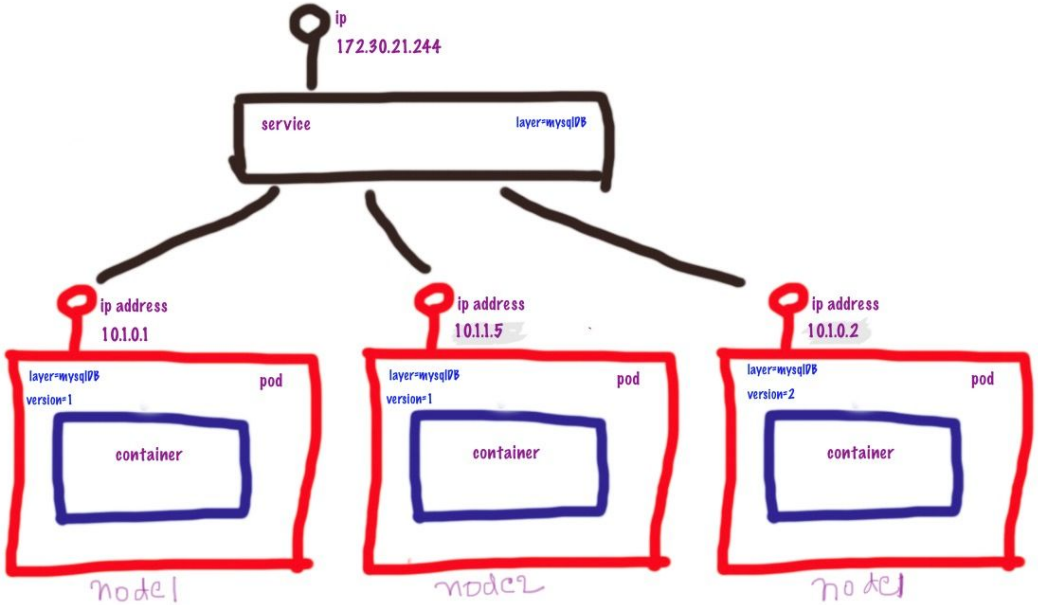


when you scale up your applicatioin component, you are scaling up pods..



Pods get distributed on your cluster.. They may land on different nodes

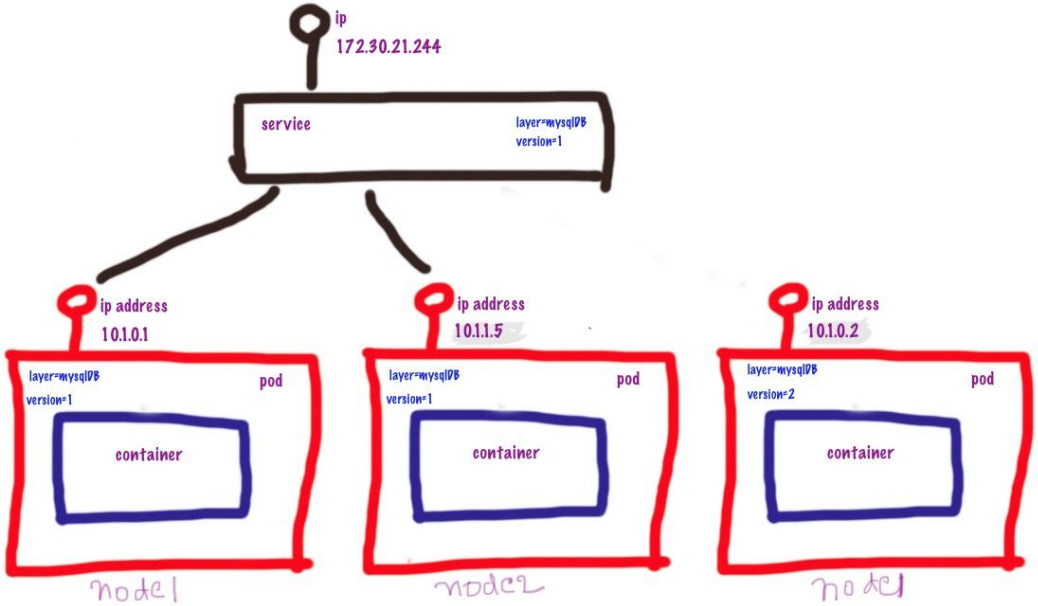
So as a client do I need to deal with all these pod ips? .. wait



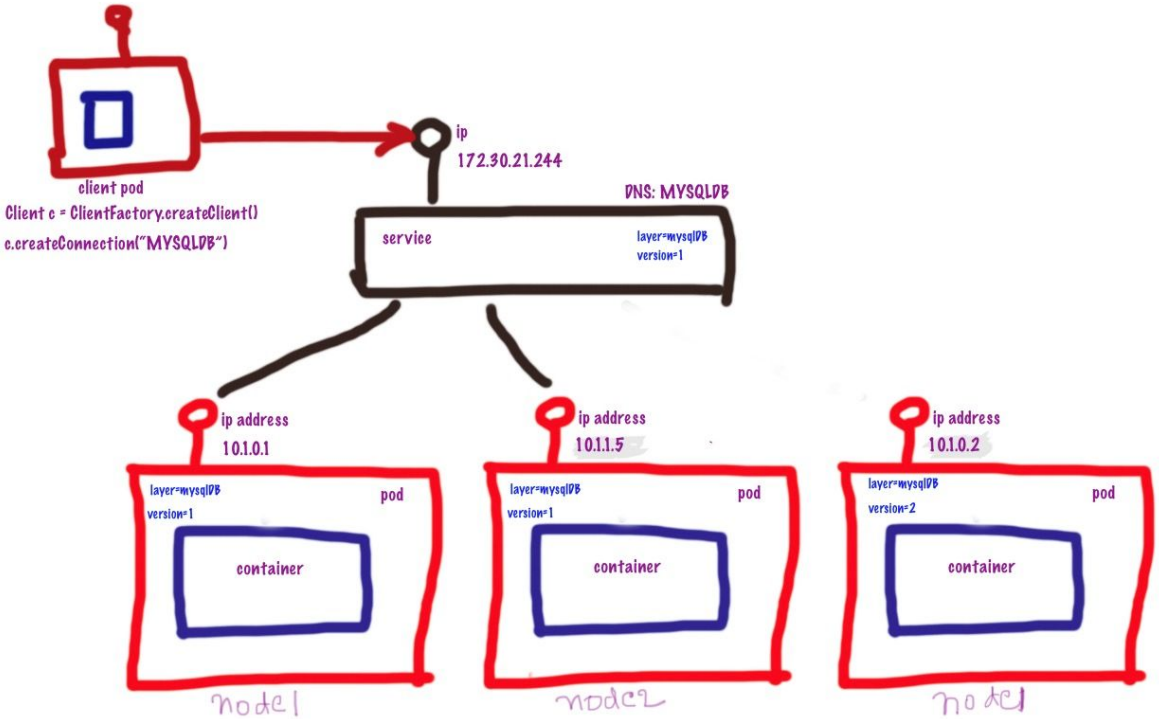
Pods can be frontended by a Service.
Service is a proxy.. Every node knows about it.

Service gets an ip

Service knows which pods to frontend based on the labels



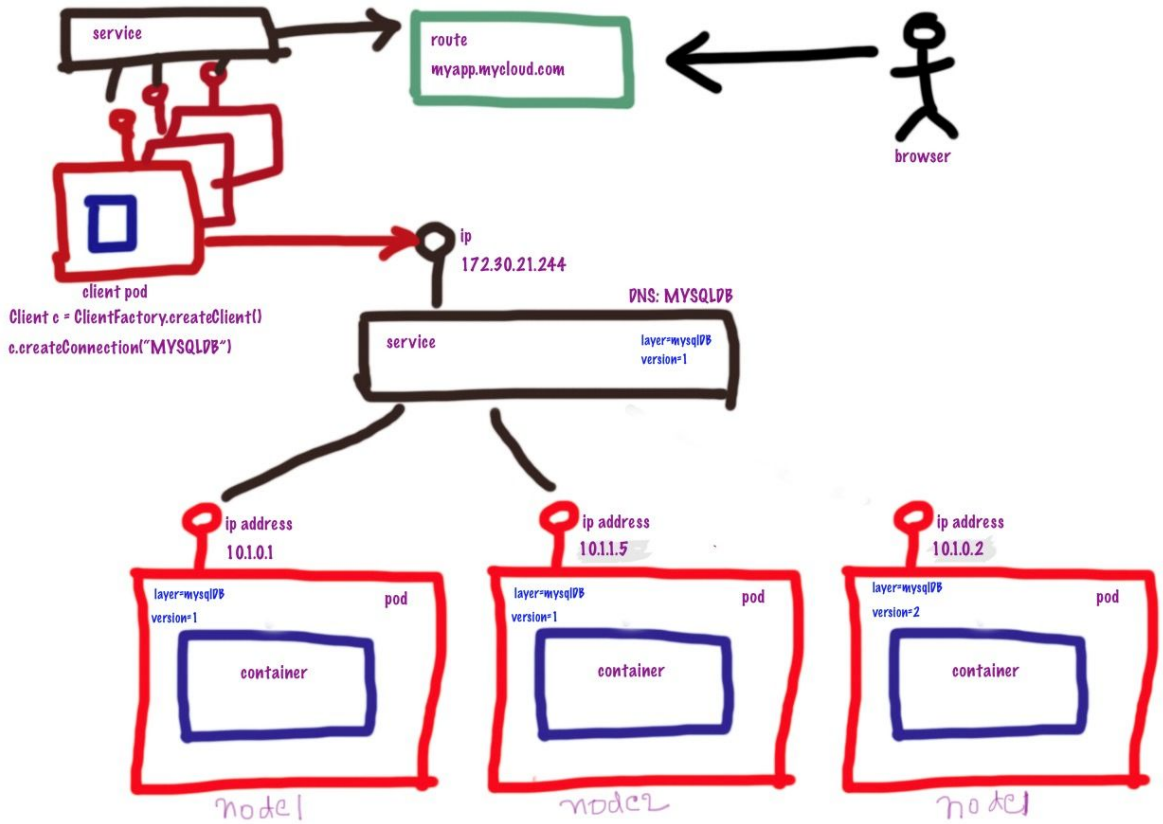
By changing the labels end points of service change



Clients can talk to the service. Service redirects the requests to the pods.

Service also gets a DNS name

Client can discover service... built in service discovery!!



when you want to expose a service externally eg: access via browser using a URL, you create a "Route"

Route gets added to a HAProxy LB.

You can configure your F5 as well as LB.