IBM Systems Lab Services

Proven IT infrastructure expertise

Containerisation/Kubernetes and OpenShift on IBM Power - 101

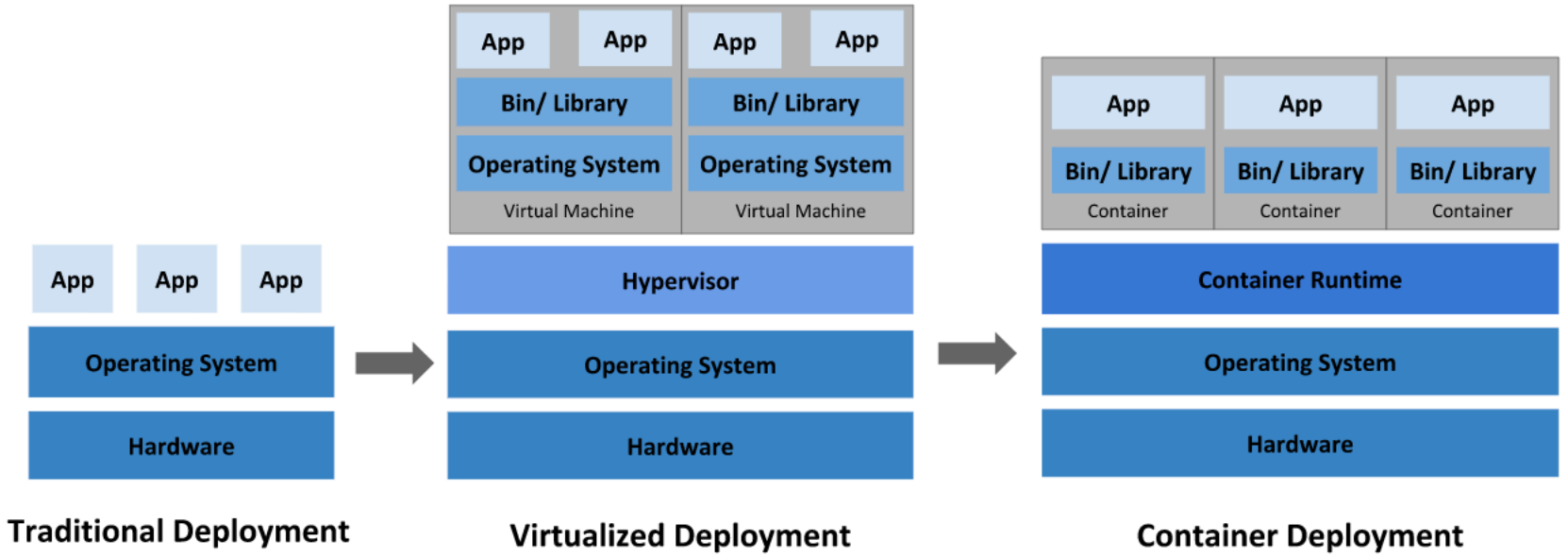
Stu Cunliffe
IBM System Lab Services
@StuCunliffe
s_cunliffe@uk.ibm.com

Session Objectives

- Overview of Containers, Docker and GitHub
  - Building and deploying images
- Overview of Kubernetes
  - Pods
  - Deployments & Upgrades
  - Services & Routes
  - Storage
- OpenShift
  - Architecture
  - Requirements
  - Installation
  - Building apps on OpenShift – demos
- Cloud Paks
  - Multicloud Manager Cloud Pak
  - Cloud Automation Manager – integrating with multi-hybrid clouds including on-premise PowerVM

Overview of Docker Containers

- Docker performs operating system level virtualization also known as containerization.
- Containers allow us to package an application along with all its dependencies such as libraries.
- Docker is the tool to allow us to create, deploy and run applications within containers.
- A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.



What are the benefits of containers?

Pros

- Lightweight, fast, isolated
- Contains all the dependencies, libraries, binaries and config needed, easy to migrate
- Small – most containers are <100MB
- Consolidation – more workloads than VMs
- Upgrades – single container can be upgraded as opposed to a whole VM
- Allows developers to work on their microservice apps, build then and share them.
- The decoupling of applications from the environment they would normally run.

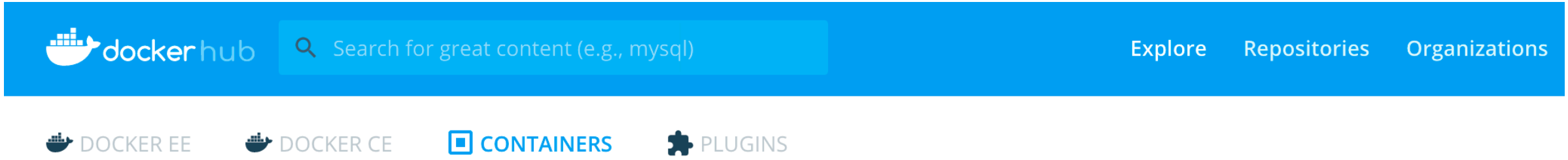
Cons

- Linux only
- Shares the OS kernel - potential vulnerabilities
- More items to monitor and manage

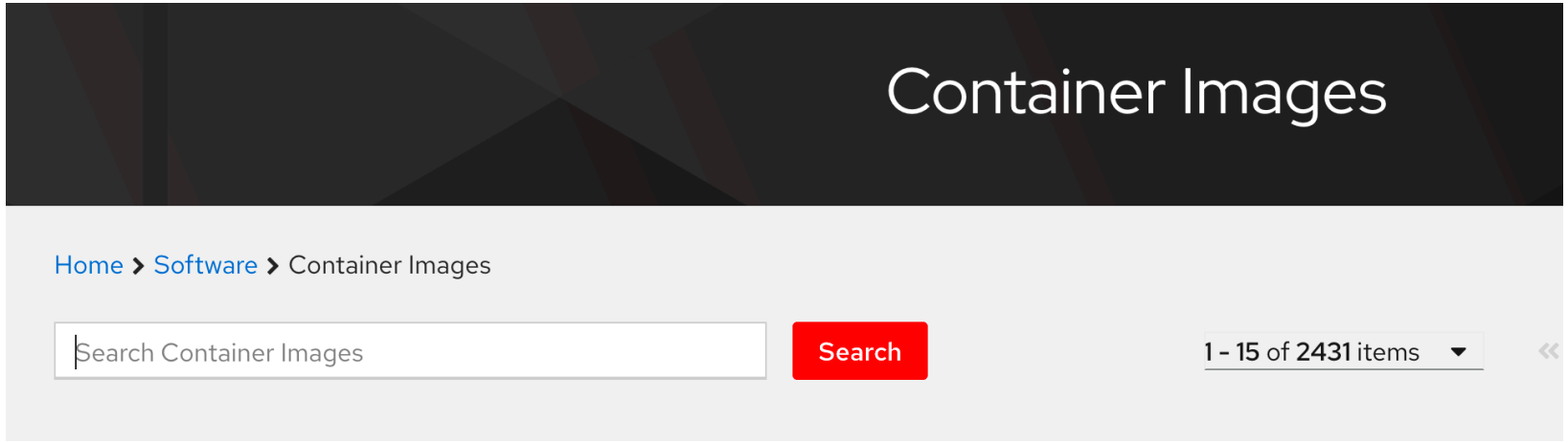
Container images – where to find them.

Images are stored in a number of locations depending on what they contain, licensing etc.:

- DockerHub (https://hub.docker.com/) – contains thousands of images for all architectures.



- Red Hat (https://catalog.redhat.com/software/containers/search) – supported and certified images.



Building a new docker image: Option 1 manually (not recommended)

Process to manually update/create a new image:

1. Deploy a ‘base’ image to create a new container e.g. registry.redhat.io/rhel7
2. Login to that container and configure it as required:
   a. Install new rpms
   b. Add users
   c. Copy files into the container
   d. Create start script to fire up when container boots
   e. etc….
3. Capture that container as a new image

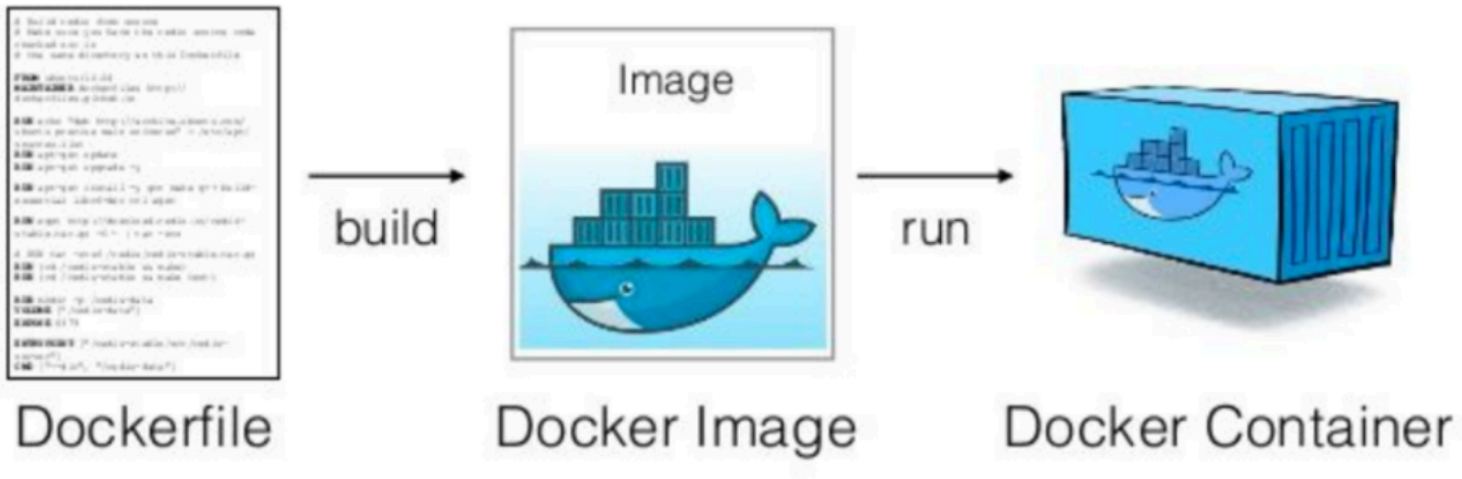
Issues with this method:

- Each time you want to make a change to the image, you have to repeat all these steps
- Prone to error if you get one of the steps wrong
- Not very agile

Building a new docker image: Option 2 Dockerfile

Process to update/create a new image using Dockerfile:

1. Create a Dockerfile that uses the ‘base’ image.
2. Add the desired configuration to that Dockerfile:
   a. Install new rpms
   b. Add users
   c. Copy files into the container
   d. State startup command to run when container is built
   e. etc....
3. Build a new image using that Dockerfile.



Dockerfile

Example Dockerfile:

```
FROM ubuntu
CMD /bin/bash
MAINTAINER Stu Cunliffe,UK s_cunliffe@uk.ibm.com
RUN apt-get update
RUN apt-get install -y npm
RUN mkdir -p /usr/src/node-red
WORKDIR /usr/src/node-red
RUN groupadd --force node-red
RUN useradd --home /usr/src/node-red --gid node-red node-red
RUN chown -R node-red:node-red /usr/src/node-red
USER node-red
RUN npm install node-red
EXPOSE 1880/tcp
COPY package.json /usr/src/node-red/package.json
COPY flow-file.json /usr/src/node-red/.node-red/node-red
CMD npm start node-red
```

We can then build the image by simply calling:

 # docker build . –t docker.io/cunlifs/node-red-iss:v0.8

Dockerfile combined with GitHub

Example Dockerfile in GitHub:

```
FROM ubuntu
CMD /bin/bash
MAINTAINER Stu Cunliffe,UK s_cunliffe@uk.ibm.com
RUN apt-get update
RUN apt-get install -y npm
RUN mkdir -p /usr/src/node-red
WORKDIR /usr/src/node-red
RUN groupadd --force node-red
RUN useradd --home /usr/src/node-red --gid node-red node-red
RUN chown -R node-red:node-red /usr/src/node-red
USER node-red
RUN npm install node-red
EXPOSE 1880/tcp
COPY package.json /usr/src/node-red/package.json
COPY flow-file.json /usr/src/node-red/.node-red/node-red
CMD npm start node-red
```

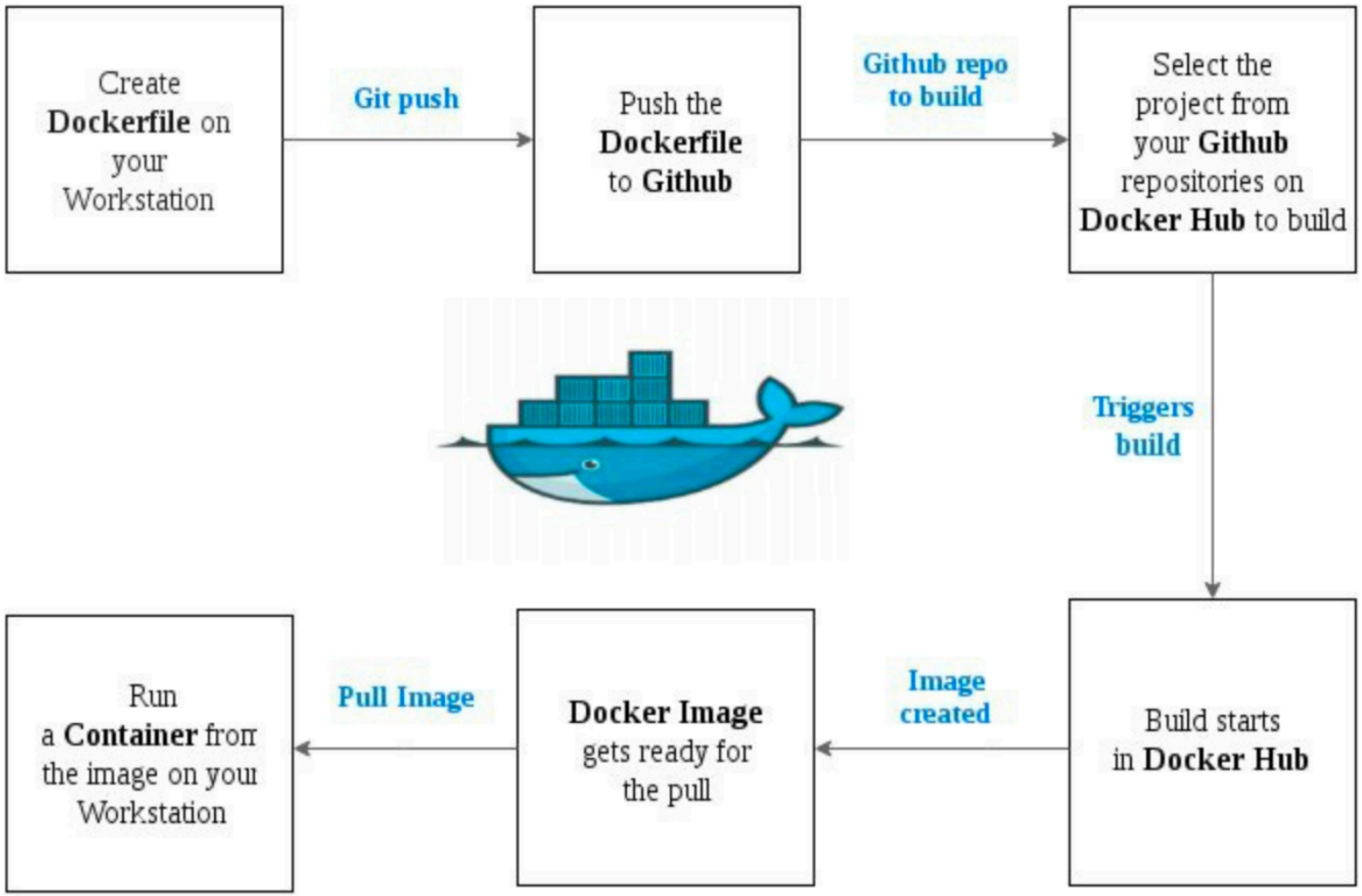
We can then build the image by simply calling the github source:

 # docker build github.com/cunlifs/node-red-iss -t docker.io/cunlifs/node-red-iss:v0.8

Pushing the new image back to Docker Hub

For the new image to be available we need to push it back to Docker Hub:

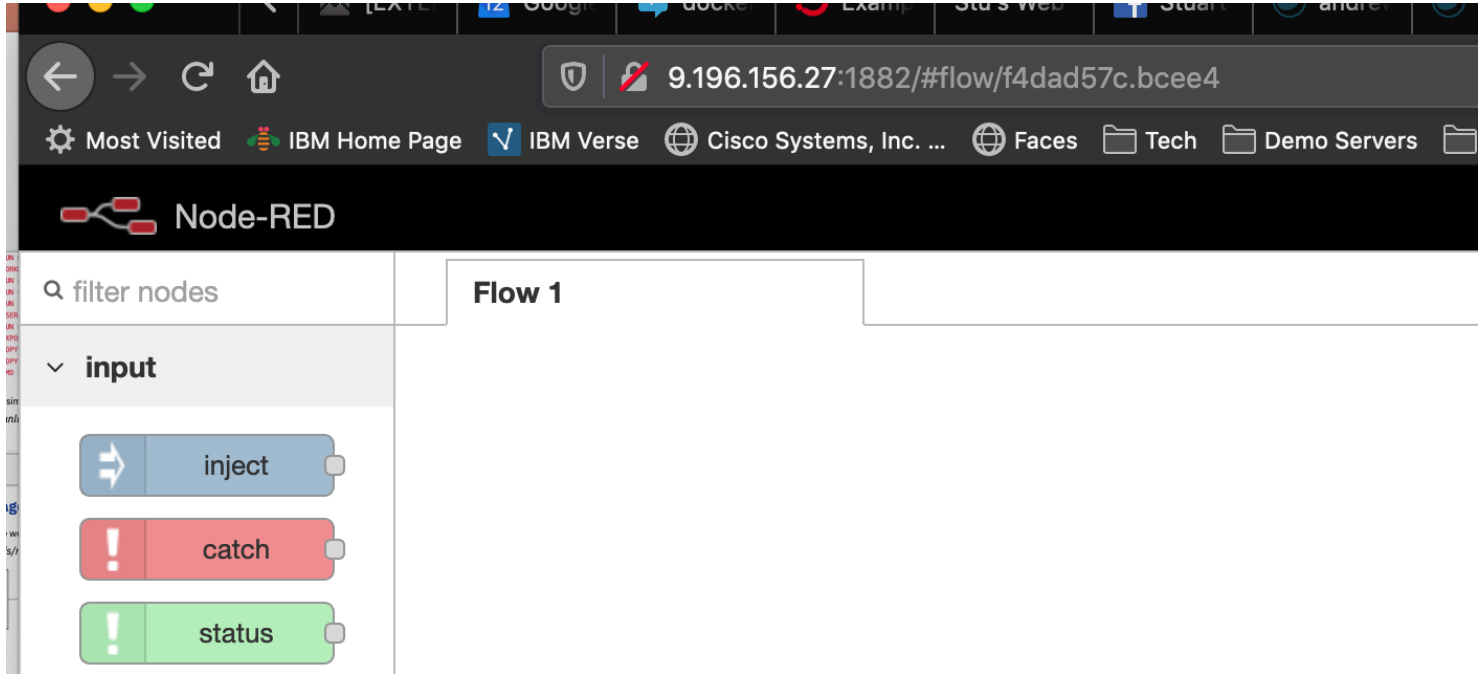
 # docker push docker.io/cunlifs/node-red-iss:v0.8



Creating and running a container from an image

We could just run it from the command line of our docker host:

 # docker run -d --name mynodered -p 1882:1880 cunlifs/node-red-iss:v0.8



Issues with this method:

- What it if fails, how would monitor it and restart it?
- What if we want to scale and add more instances?
- How do we upgrade it with little or no impact to the users?

Overview of Kubernetes (K8s)

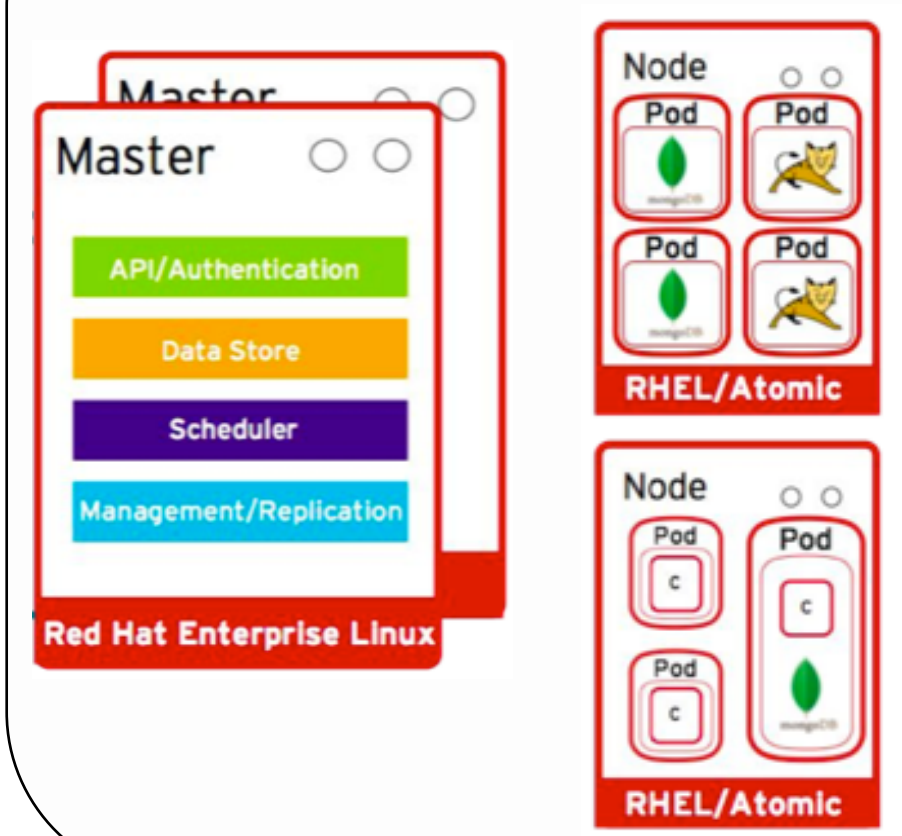
- Kubernetes is the orchestration layer that manages containers across a group of physical servers or VMs.
- Kubernetes is specifically designed to manage the ephemeral nature of thousands of containers by:
  - deploying containerised applications
  - scaling up due to demand
  - scaling down or terminating
  - version control
  - internal and external container communications
  - storage creation and attachment
  - monitoring – what to do when a host server/VM fails

- Kubernetes runs just as well on traditional on-premises infrastructure stacks as it does for third-party service providers and public cloud environments.
- Offerings include: IBM Kubernetes Service (IKS), Azure Kubernetes Service (AKS), Amazon Elastic Kubernetes Service (EKS) and Red Hat OpenShift.

Key features of Kubernetes

At a high-level, Kubernetes is two things

Cluster



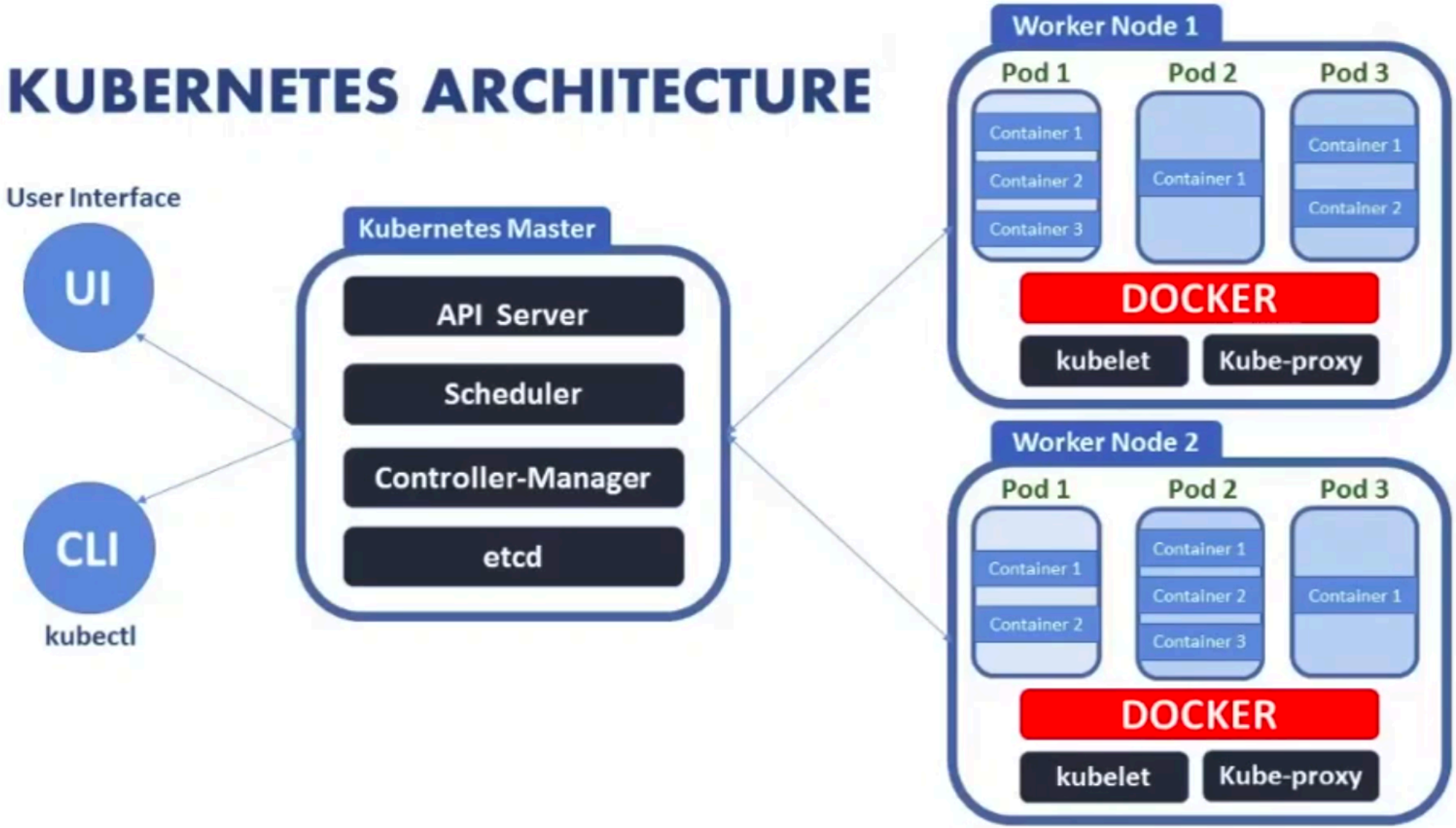
Orchestrator

What?
Where?
When?
How many?
Monitor?
What if…?

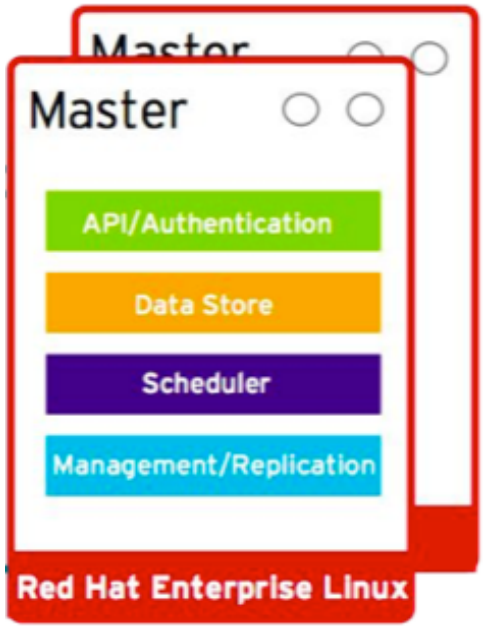
Kubernetes Terminology

1. Nodes – VM/Server that runs a workload
2. Pods – Collection of containers, normally just a single container per pod
3. Projects – Namespaces/Reserved space
4. Services – Collection of pods, exposed as an endpoint
5. Routes – exposes services for external comms
6. Deployments - creates the pods and replica sets in single configuration

KUBERNETES ARCHITECTURE



Master(s) – the ‘control plane’



At a high-level they have 4 main functions/components:

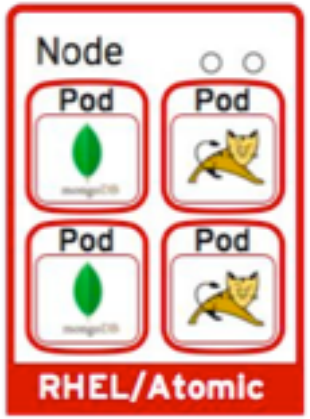
1. API Server – validates and configures the data for pods, services and replication controllers. It also assigns pod to node and synchronises pod information with service configuration. Entry point of the cluster.

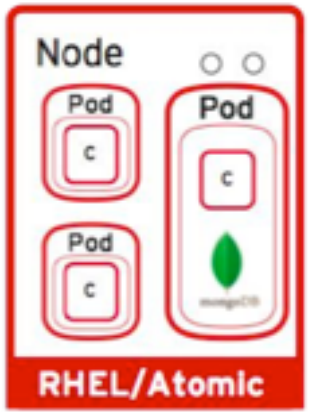
2. Etcd (cluster or data store) - the only stateful part of the control plane, stores the entire configuration and state of the cluster. Typically deployed with 2n+1 peer services. Optionally can be created outside of K8s cluster.

3. Scheduler – watches for new work tasks and assigns them to the appropriate healthy node.

4. Controller Management Server – watches etcd for changes to replication controller objects and then uses the API to enforce the desired state. ’The controller of controllers’.

Worker nodes





At a high-level they only have 3 roles:

1) Watch the API server for new work assignments

2) Execute new work assignments

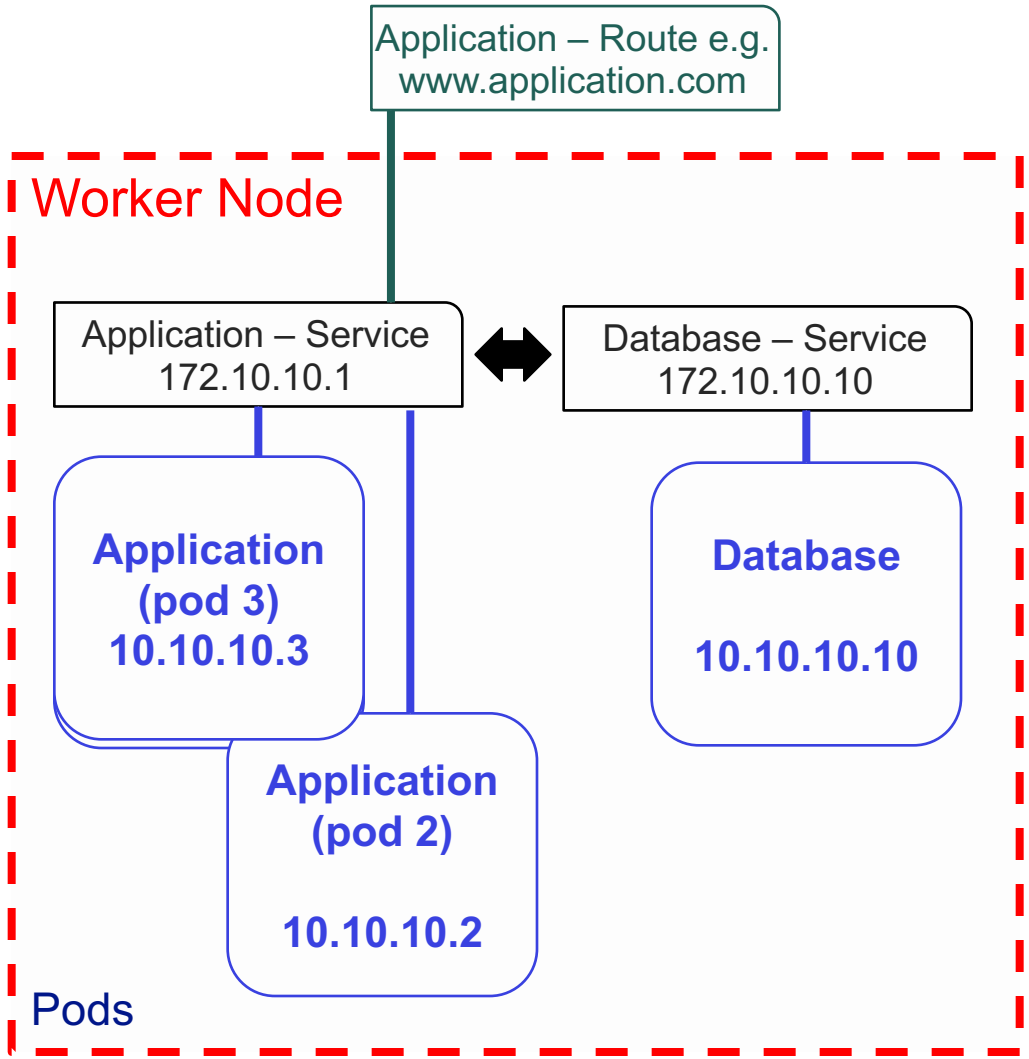
3) Report back to the control plane (via the API server)

They run the application containers.

Worker nodes:

- Can be bare-metal or VMs
- Communicate with the master via authorization certificates
- Consists of kubelet, kube-proxy and container runtime (e.g docker)

Kubernetes Terminology – Services and Routes



Service

1) For each application a ‘service’ is created.

2) The service knows about the pods/endpoints running that application.

3) Should a pod restart or a new replica get created the service adds it as a new ‘endpoint’.

Route

1) Each service can be exposed as an external route.

2) As the service add/removes pods the route is unaffected.

Kubernetes Terminology – Deployment

A deployment consists of 3 separate API objects:

1. A **deployment configuration**, which describes the **desired state** of a particular component of the application.

2. One or more replication controllers, which contain a point-in-time record of the state of a deployment configuration.

3. One or more pods, which represent an instance of a particular version of an application.

Note: We do not need to manipulate replication controllers or pods owned by deployment configurations. The deployment system ensures changes to deployment configurations are propagated appropriately.

Kubernetes Terminology – Deployment Configs

```
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  annotations:
    description: Defines how to deploy node-red server
labels:
  app: node-red
  name: node-red-smc
  namespace: lab-services-uk
spec:
  replicas: 1
template:
   spec:
      containers:
      - image: docker.io/cunlifs/node-red:v1.3
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
            path: /
            port: 1880
            scheme: HTTP
      ports:
        - containerPort: 1880
          protocol: TCP
        readinessProbe:
          failureThreshold: 3
          httpGet:
            path: /
            port: 1880
            scheme: HTTP
          initialDelaySeconds: 3
          periodSeconds: 10
          successThreshold: 1
          timeoutSeconds: 3
        resources:
          limits:
            memory: 512Mi
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      restartPolicy: Always
```

Kubernetes and Docker: complementary technologies

Develop application in your favourite language

Package application in Docker

Test application in Docker

Ship application in Docker

Development

Production

Kubernetes Cluster

Node 1

Runtime:

Node 2

Runtime:

Node 3

Runtime:

containerd

Node 4

Runtime:

kata

Deploy and manage application with Kubernetes

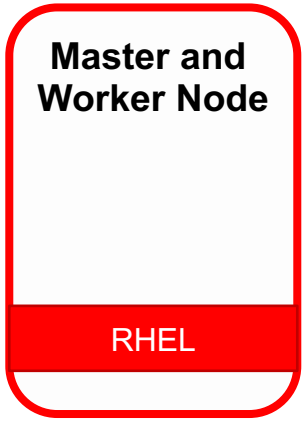
Overview of Red Hat OpenShift Container Platform (OCP)

- Kubernetes (K8s) is an open source project, whereas OpenShift is a Red Hat product. K8s has around 4 releases per year.
- OpenShift v3 runs on Red Hat Enterprise Linux (RHEL) or Red Hat Atomic
- OpenShift v4 runs Red Hat CoresOS – optionally RHEL for compute nodes
- OpenShift is a Platform-as-a-Service (PaaS)
- There is an opensource version of OpenShift called OKD (previously OpenShift Origin) and can be run on RHEL or CentOS. Contains most features but no support or Red Hat based official images.
- OpenShift Container Platform (OCP) is a product you can install on your infrastructure that includes paid support with subscription
- Pod one or more containers deployed together on a host, smallest entity.

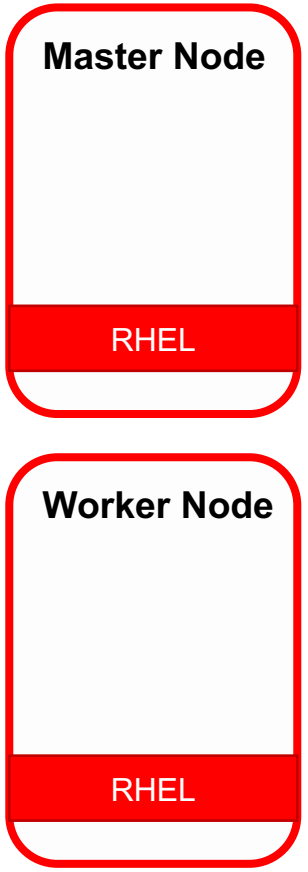
Red Hat OpenShift Container Platform - solutions

RED HAT® OPENSHIFT Container Platform

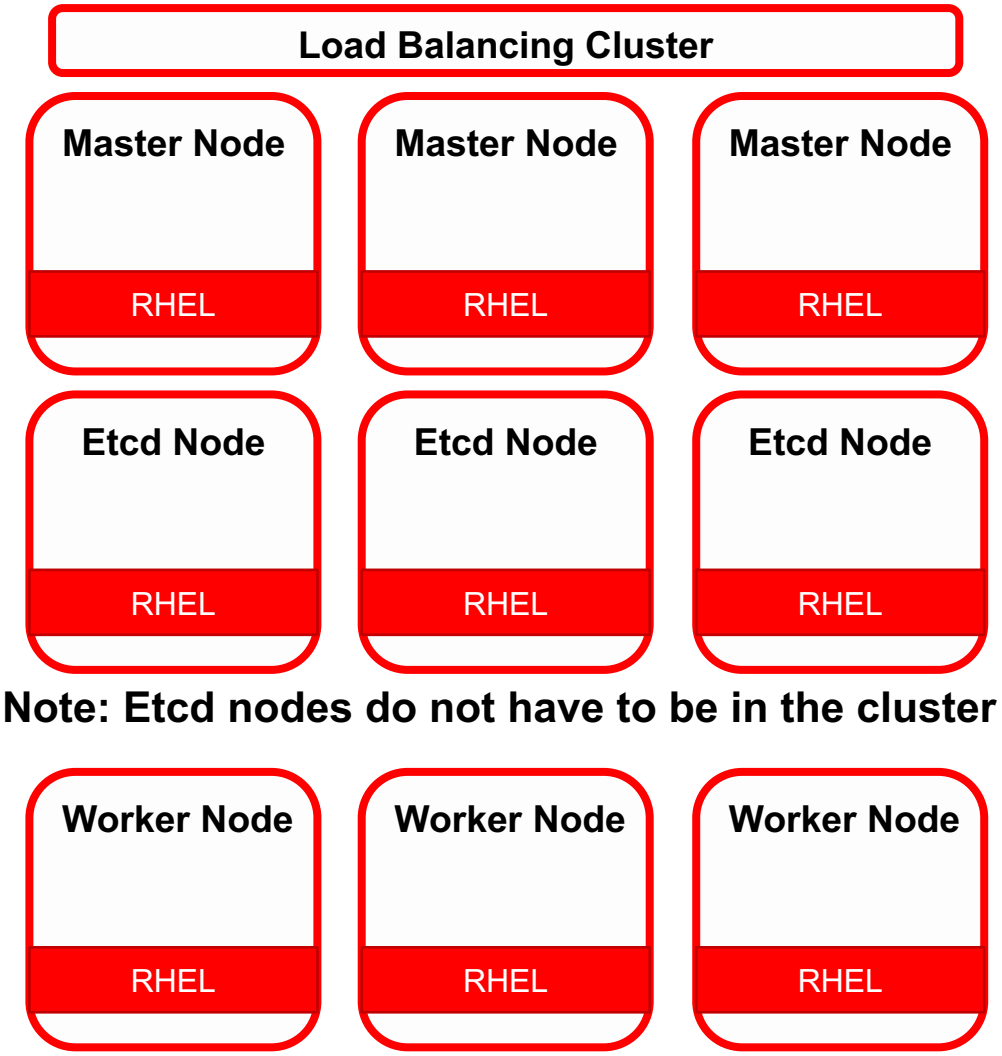
"All-in-one"
Basic test environment



"Separate Master/Worker"
PoC environment



"Highly Available"
Production environment



Installing OCP on Power - hardware

- Masters:
  - IBM Power9 or IBM Power8
  - Min 4 vCPU (more recommended) – recommend min of 2 physical cores + #pods
  - Min 16GB RAM (more if etcd is installed) – recommend min of 32GB
  - If running Cloud Pak on master – size to that requirement

#pod notes: A master host needs to meet the minimum requirements and have 1 CPU core and 1.5 GB of memory for each 1000 pods. Therefore, the recommended size of a master host in an OpenShift Container Platform cluster of 2000 pods is the minimum requirements of 2 CPU cores and 16 GB of RAM, plus 2 CPU cores and 3 GB of RAM, totalling 4 CPU cores and 35 GB of RAM.

- Nodes:
  - IBM Power9 or IBM Power8
  - Min 1 vCPU – recommend at least 1 core + overhead/HA
  - Min 8GB RAM – recommend min of 16GB

Sizing is heavily dependent on the number and type of applications being run on the nodes.
Overhead and HA notes: Leave at least 10% overhead, plus if in a production environment enough capacity to host the pods should a node be lost.

HW Pre reqs: https://docs.openshift.com/container-platform/3.11/install/host_preparation.html#hardware

Installing OCP on Power - software

- Build the appropriate number of RHEL VMs/BareMetal Servers
- Install is done via a number of Ansible playbooks.
- Pre-reqs:
  - Ensure VMs have access to RHEL repository – usually via subscription manager
  - DNS working across all nodes (Dnsmasq installed)
  - SELinux set to enforcing
  - Proxies set for remote access, and ’no proxy’ for local access
  - Install openshift-ansible
  - Install docker
  - Exchange public ssh keys from the Ansible install server to hosts

SW Pre reqs: https://docs.openshift.com/container-platform/3.11/install/host_preparation.html#software-prerequisites

Installing OCP on Power via Ansible Playbooks – Simple Cluster

Once openshift-ansible is installed we have to build our appropriate Ansible inventory file, depending on the number of master, infrastructure, etcd and worker nodes etc.

```
[OSEv3:children]
masters
nodes
etcd

[OSEv3:vars]
ansible_user=root
openshift_deployment_type=openshift-enterprise
oreg_url=registry.redhat.io/openshift3/ose-${component}:${version}
oreg_auth_user=StuCunliffe
oreg_auth_password=xxxxxxx

openshift_router_selector='node-role.kubernetes.io/infra=true'
openshift_registry_selector='node-role.kubernetes.io/infra=true'

[masters]
lab-ocp-1.labs.uk.ibm.com

[etcd]
lab-ocp-1.labs.uk.ibm.com

[nodes]
lab-ocp-1.labs.uk.ibm.com openshift_node_group_name='node-config-master-infra'
lab-ocp-2.labs.uk.ibm.com openshift_node_group_name="node-config-compute"
```

Simple Cluster:
1 x Master
1 x Worker

Installing OCP on Power via Ansible Playbooks – Highly Available Cluster

HA Cluster:
3 x Masters
2 x Workers

```
[OSEv3:children]
…
lb

[OSEv3:vars]
openshift_master_cluster_method=native
openshift_master_cluster_hostname=lab-ocp1-lb1.labs.uk.ibm.com
openshift_master_cluster_public_hostname=lab-ocp1-lb1.labs.uk.ibm.com
…
[masters]
lab-ocp1-m1.labs.uk.ibm.com
lab-ocp1-m2.labs.uk.ibm.com
lab-ocp1-m3.labs.uk.ibm.com

[etcd]
lab-ocp1-m1.labs.uk.ibm.com
lab-ocp1-m2.labs.uk.ibm.com
lab-ocp1-m3.labs.uk.ibm.com

[lb]
lab-ocp1-lb1.labs.uk.ibm.com

[nodes]
lab-ocp1-m1.labs.uk.ibm.com openshift_node_group_name='node-config-master-infra'
lab-ocp1-m2.labs.uk.ibm.com openshift_node_group_name='node-config-master-infra'
lab-ocp1-m3.labs.uk.ibm.com openshift_node_group_name='node-config-master-infra'
lab-ocp1-w1.labs.uk.ibm.com openshift_node_group_name="node-config-compute"
lab-ocp1-w2.labs.uk.ibm.com openshift_node_group_name="node-config-compute”
```

Installing OCP on Power: pre requisities setup

ansible-playbook playbooks/prerequisites.yml

This runs 23 ‘plays’ which consist of 211 ’tasks’:

```
PLAY RECAP *********************************************************************
lab-ocp1-lb1.labs.uk.ibm.com : ok=25   changed=4    unreachable=0    failed=0
lab-ocp1-m1.labs.uk.ibm.com : ok=69   changed=6    unreachable=0    failed=0
lab-ocp1-m2.labs.uk.ibm.com : ok=54   changed=6    unreachable=0    failed=0
lab-ocp1-m3.labs.uk.ibm.com : ok=54   changed=6    unreachable=0    failed=0
lab-ocp1-w1.labs.uk.ibm.com : ok=51   changed=6    unreachable=0    failed=0
lab-ocp1-w2.labs.uk.ibm.com : ok=51   changed=6    unreachable=0    failed=0
localhost                  : ok=12   changed=0    unreachable=0    failed=0


INSTALLER STATUS ***************************************************************
Initialization  : Complete (0:02:01)
Thursday 02 January 2020  14:24:59 +0000 (0:00:00.420)       0:05:23.121 ******
===============================================================================
```

Installing OCP on Power: deploy cluster

ansible-playbook playbooks/deploy_cluster.yml

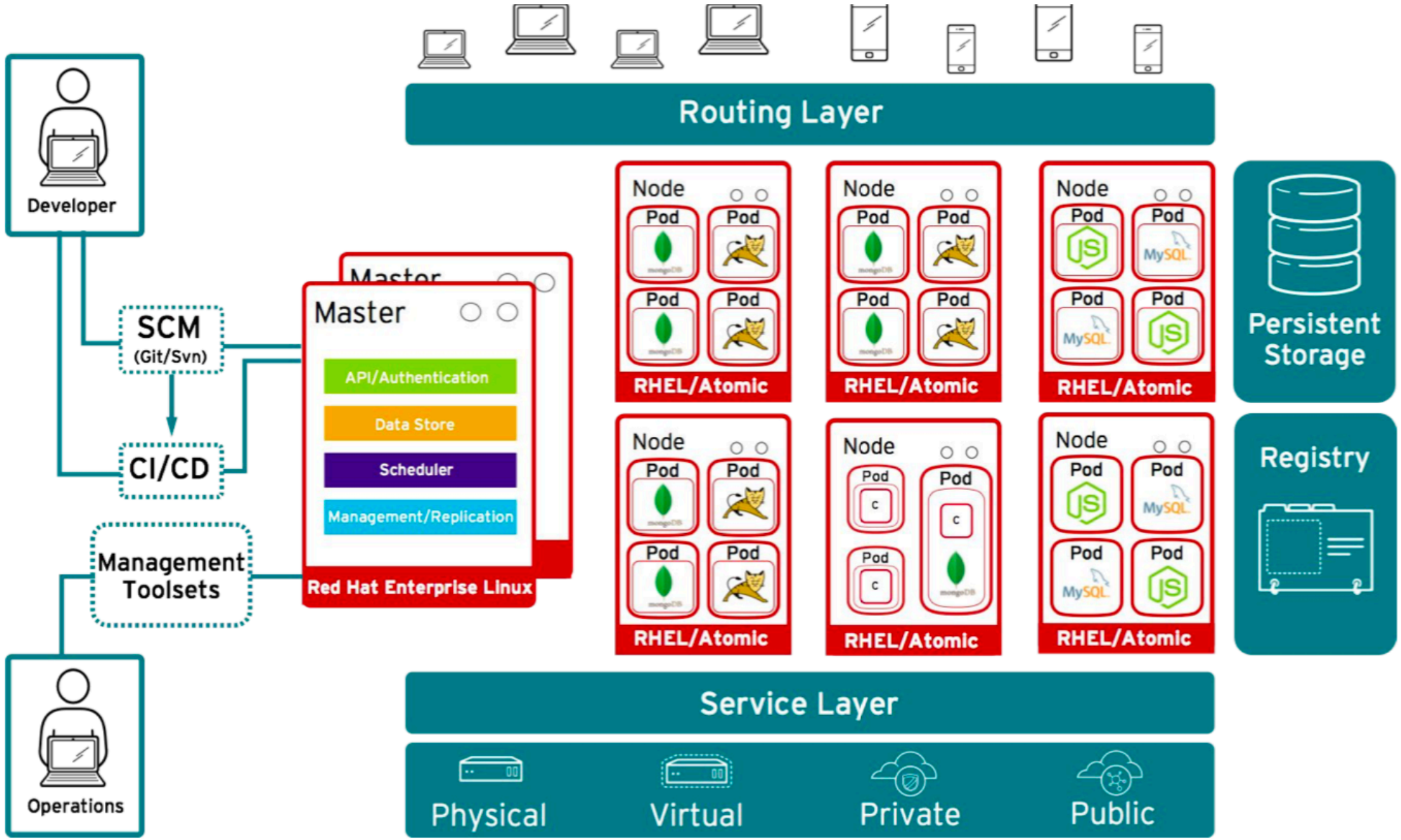
This runs 146 ‘plays’ which consist of 1,796 ’tasks’:

```
PLAY RECAP *********************************************************************
lab-ocp1-lb1.labs.uk.ibm.com : ok=26   changed=7    unreachable=0    failed=0
lab-ocp1-m1.labs.uk.ibm.com : ok=657  changed=281  unreachable=0    failed=0
lab-ocp1-m2.labs.uk.ibm.com : ok=281  changed=121  unreachable=0    failed=0
lab-ocp1-m3.labs.uk.ibm.com : ok=281  changed=123  unreachable=0    failed=0
lab-ocp1-w1.labs.uk.ibm.com : ok=118  changed=41   unreachable=0    failed=0
lab-ocp1-w2.labs.uk.ibm.com : ok=118  changed=41   unreachable=0    failed=0
localhost                  : ok=12   changed=0    unreachable=0    failed=0


INSTALLER STATUS ***************************************************************
Initialization             : Complete (0:00:23)
.......
Tuesday 31 December 2019  17:02:44 +0000 (0:00:00.026)       0:50:49.339 ******
===============================================================================
```

Overview of Red Hat OpenShift

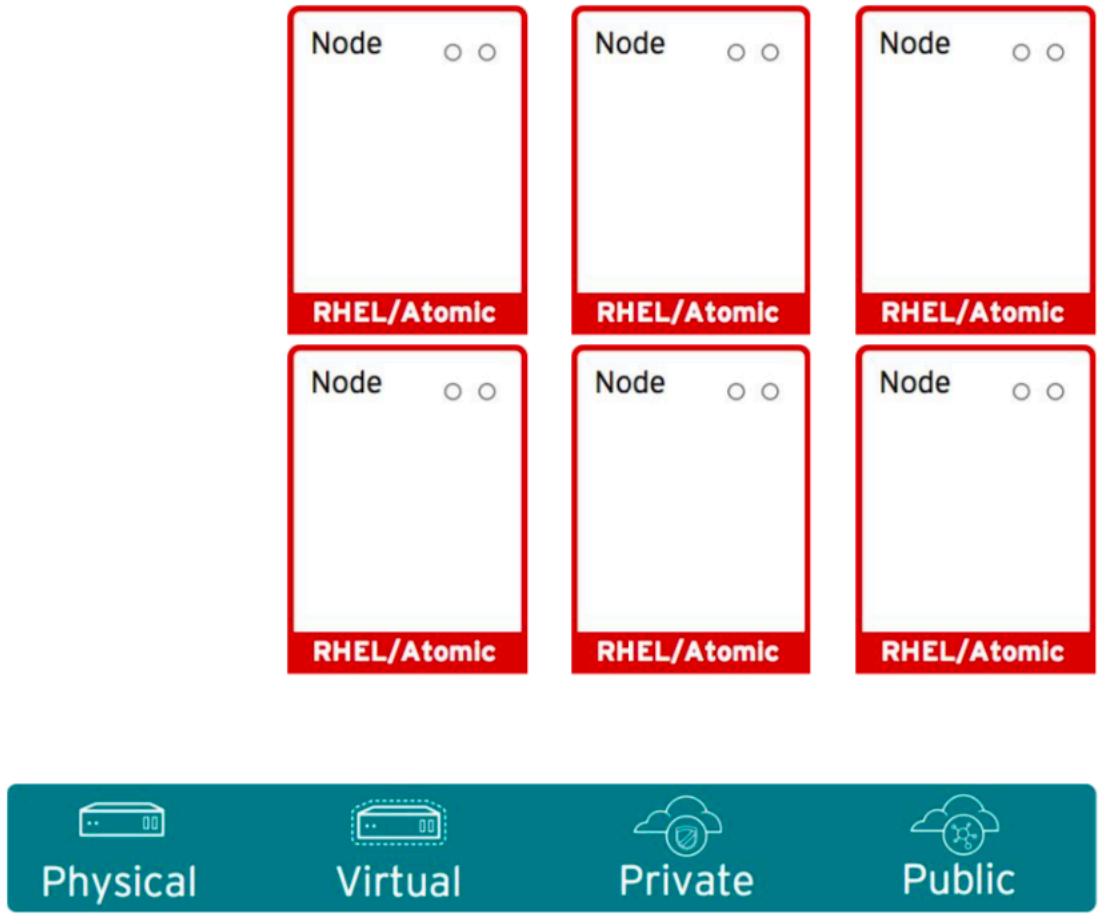
OpenShift: Cloud Architecture



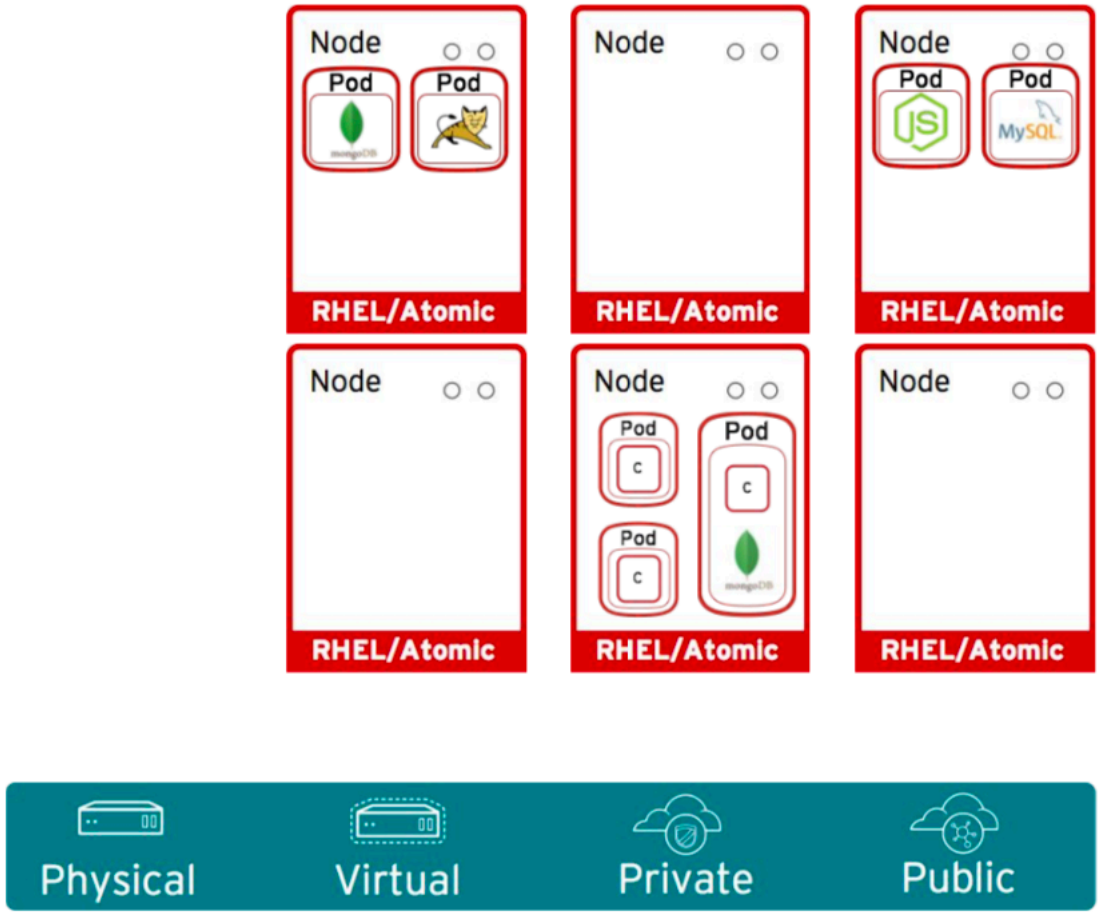
Source: Stephen Bylo – Snr Solution Architect Red Hat

Runs on your choice of infrastructure

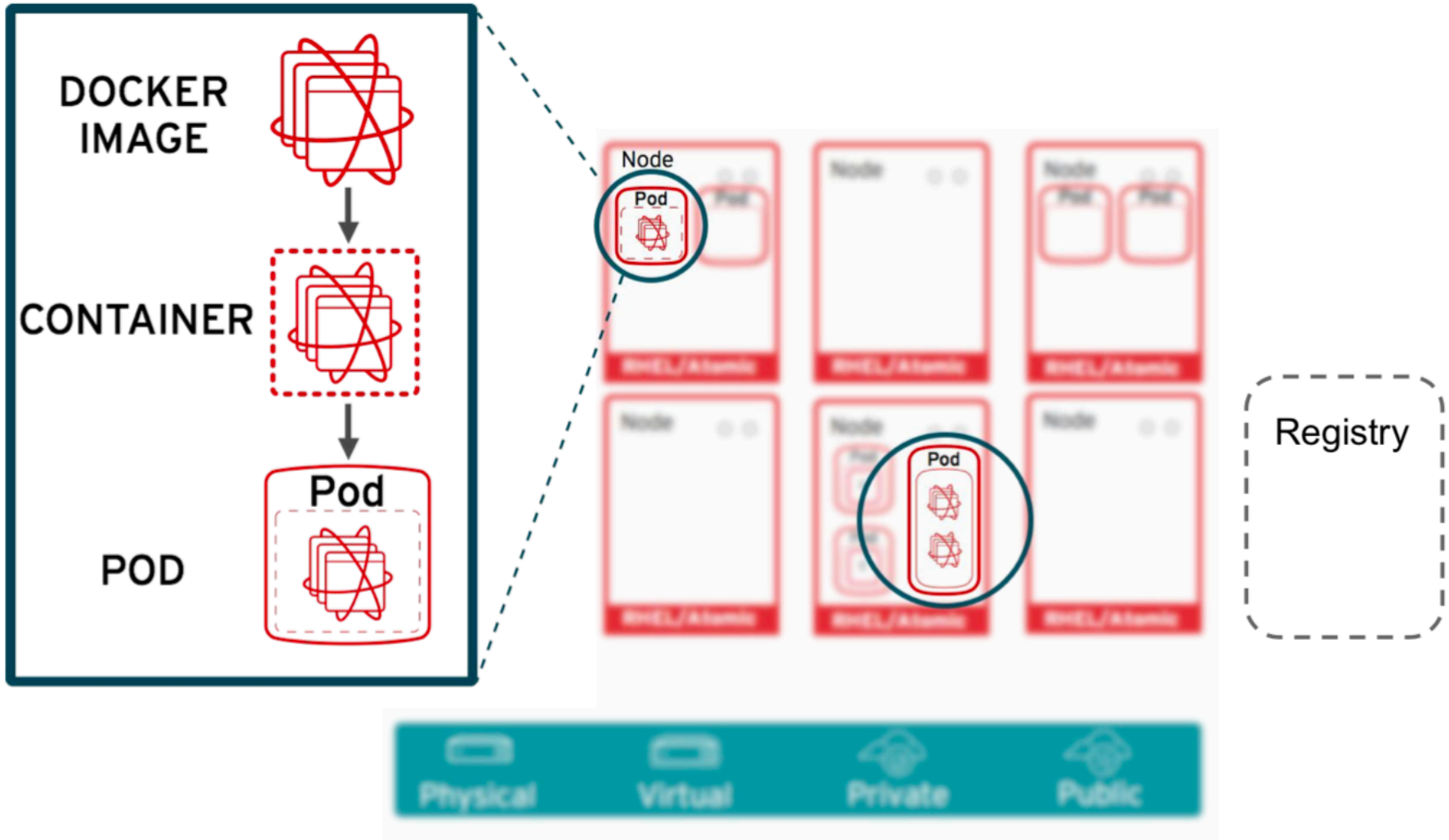
Nodes are instances of RHEL where the app will run



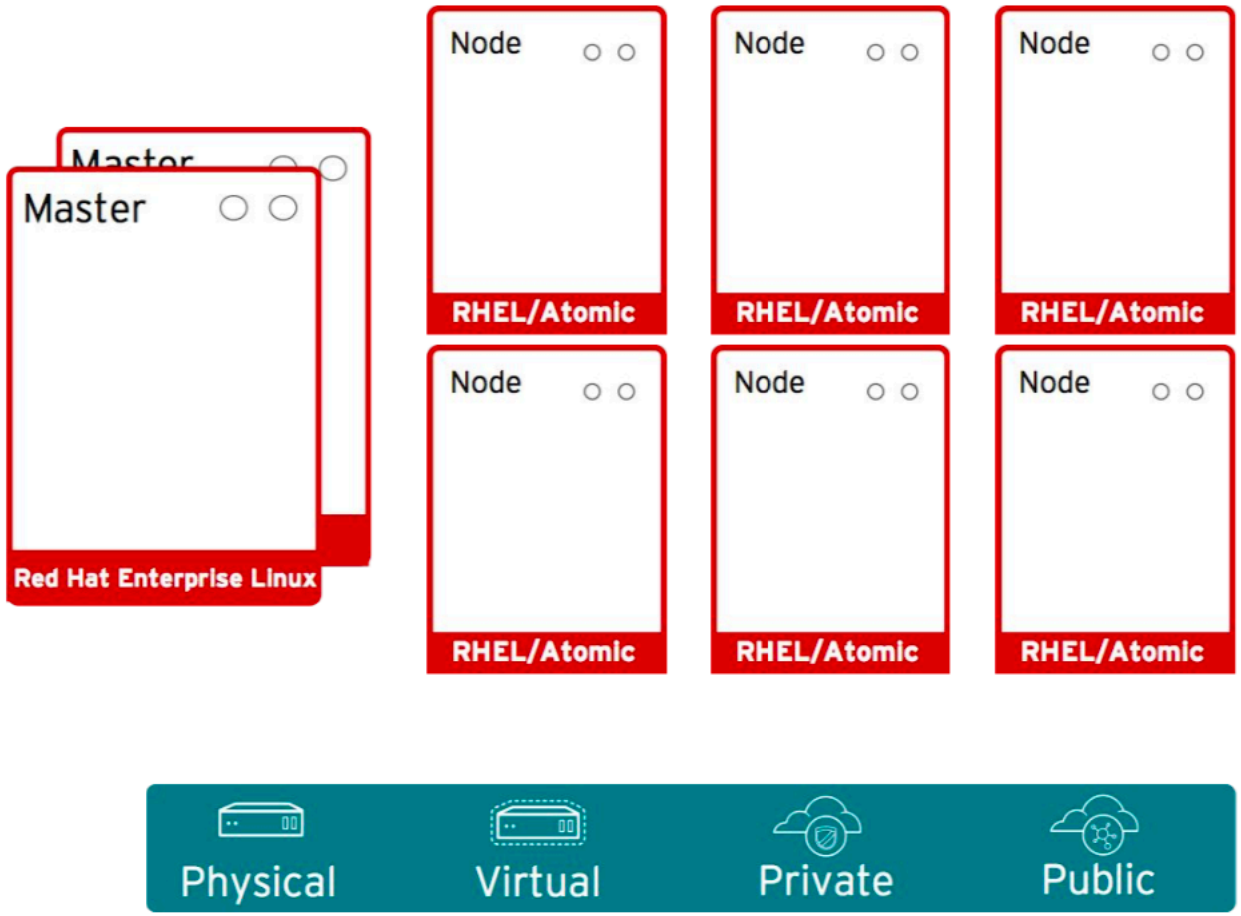
Applications run in pods/containers on each node



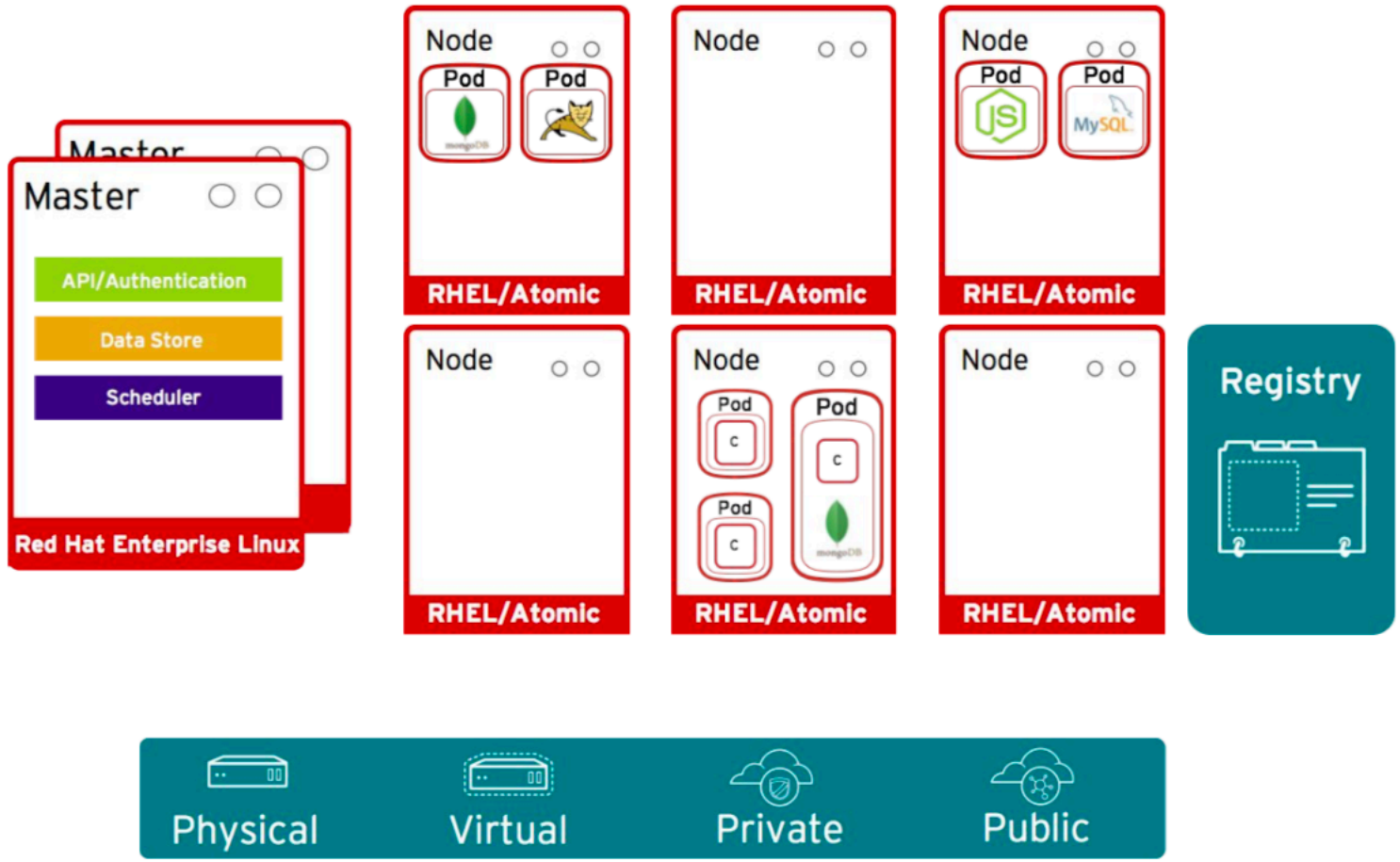
Pods run one or more containers as a unit



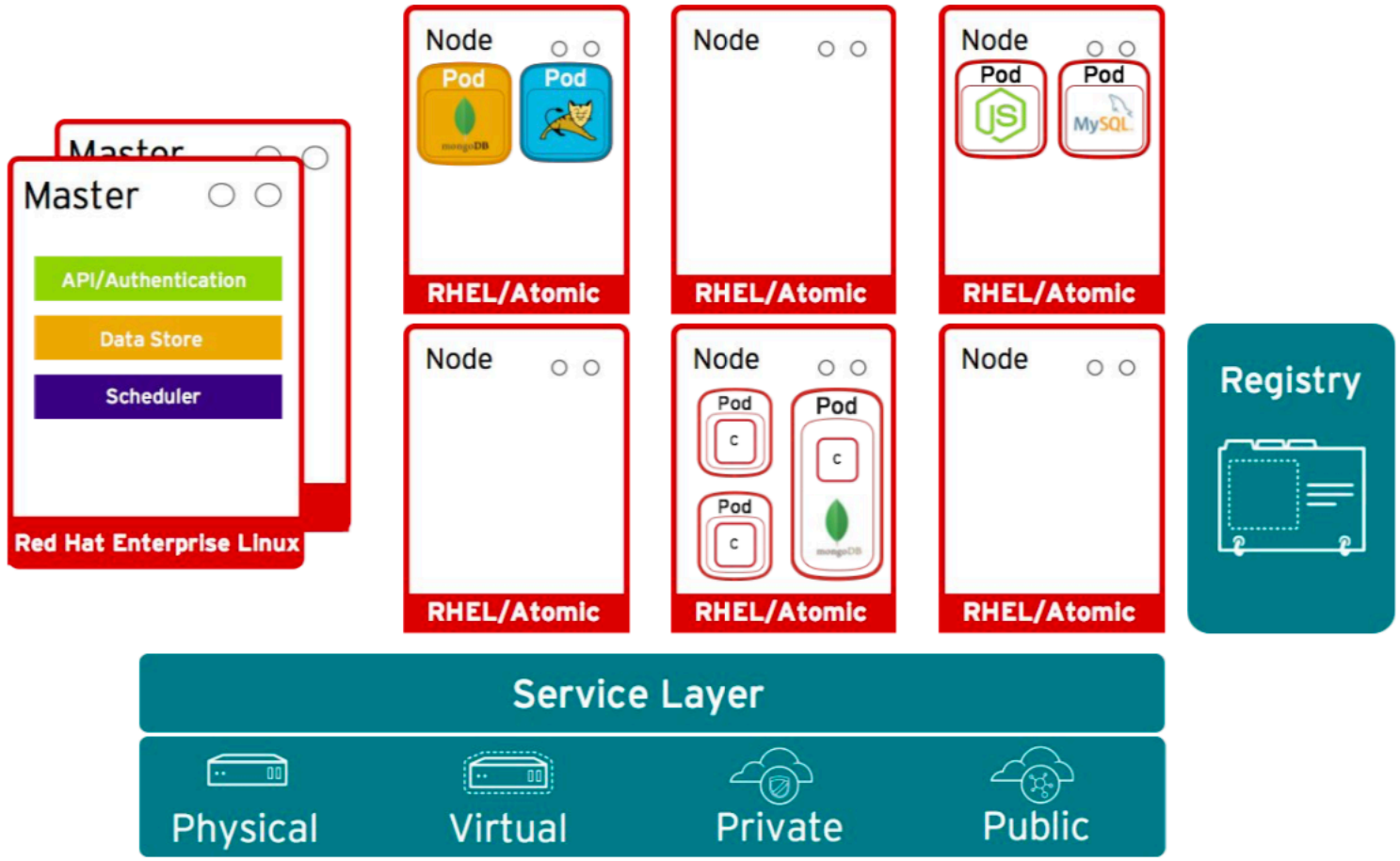
Masters leverage Kubernetes to orchestrate the nodes and apps



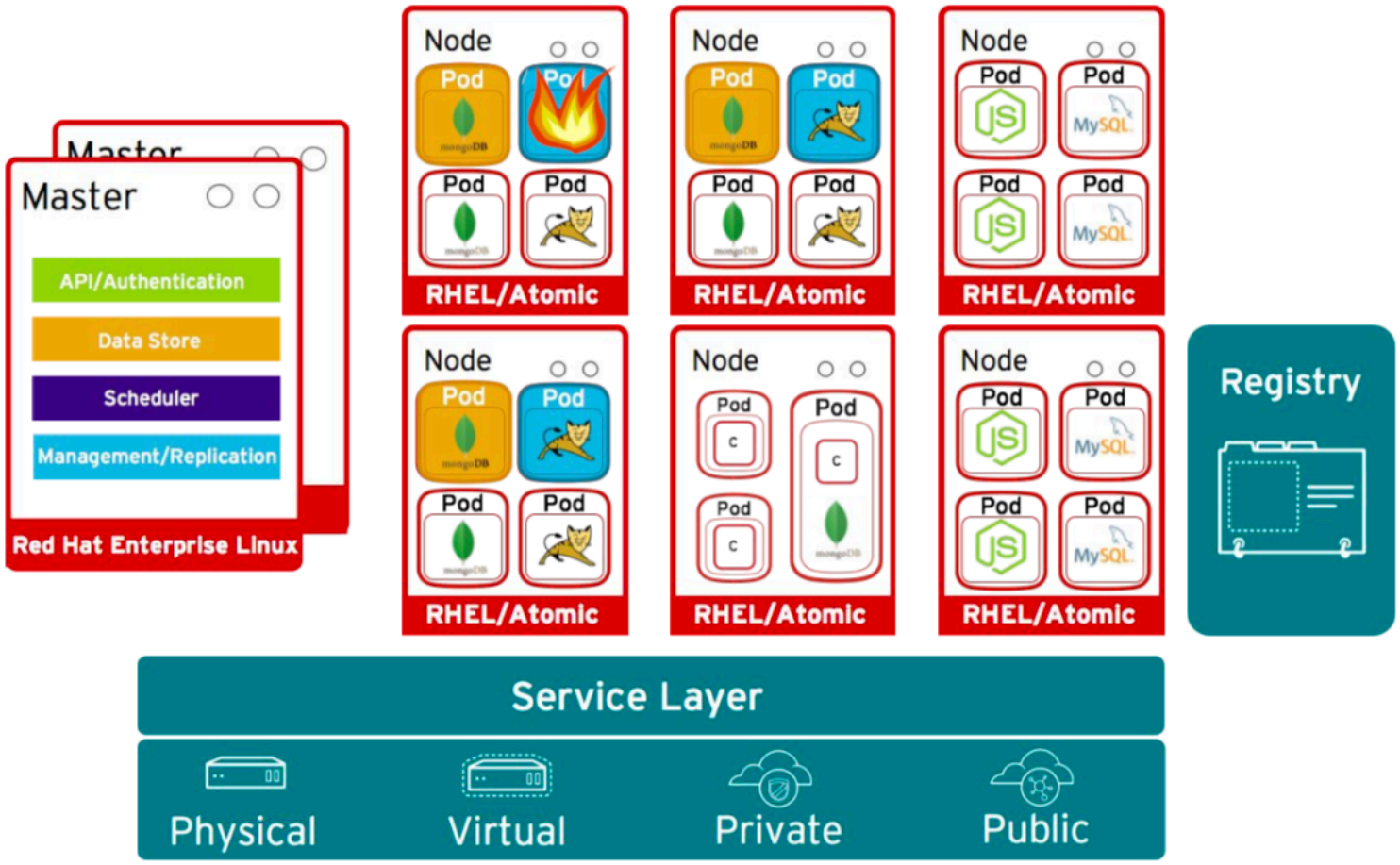
Master provide: auth APIs, key-value stores and scheduler for pods



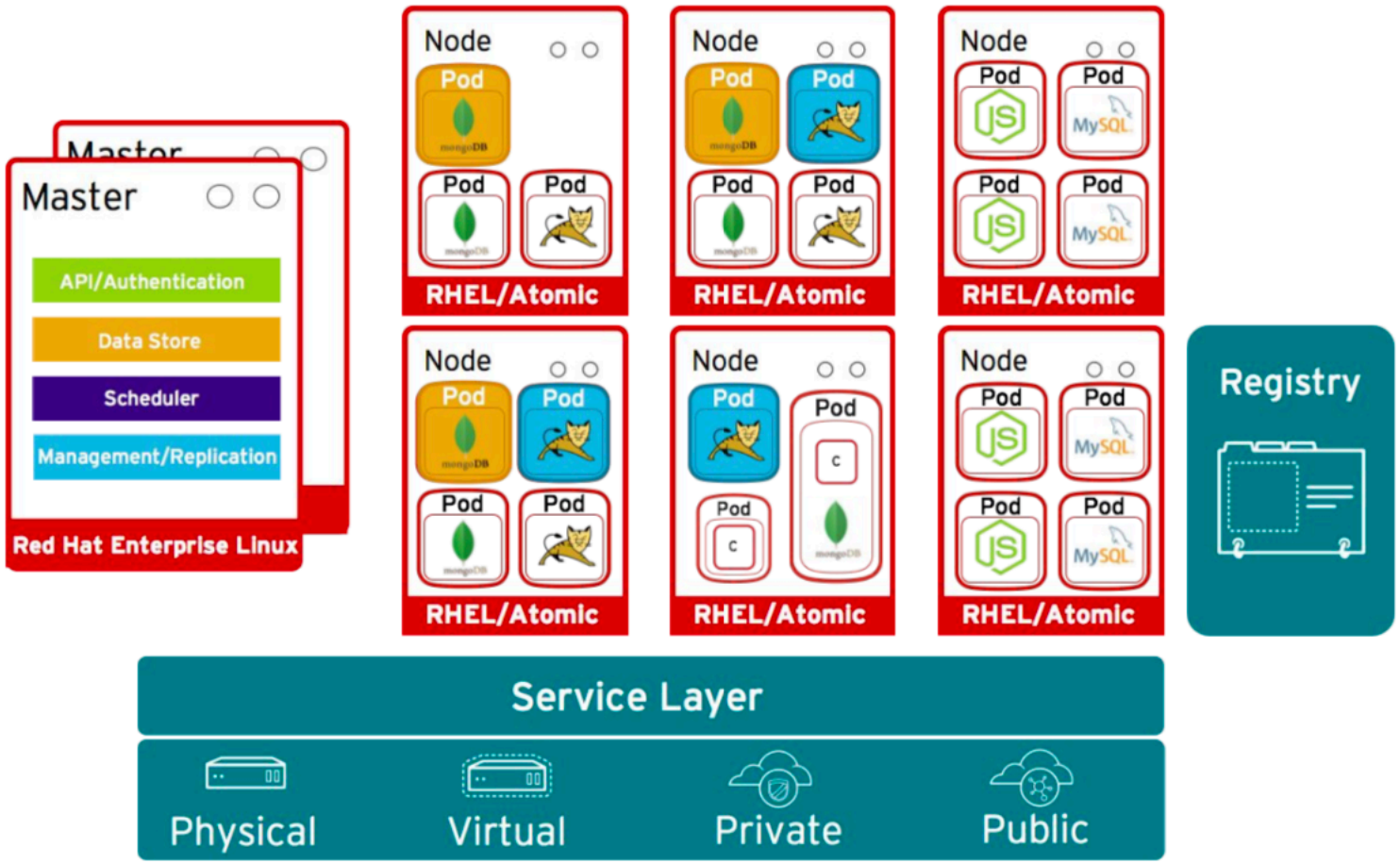
Services allow related pods to communicate with each other



Mgmt/Replication controls the pod lifecycle

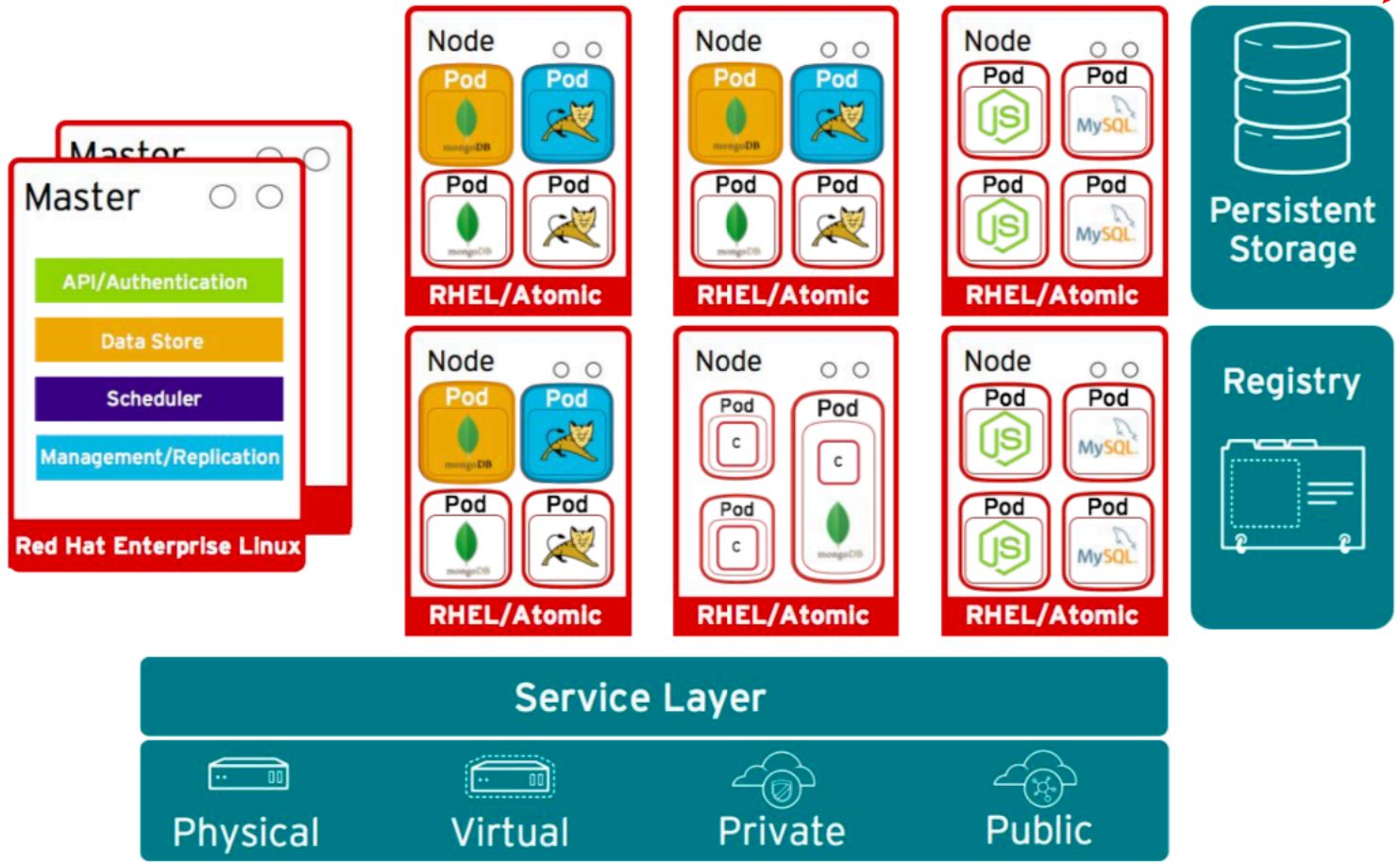


Responds to issues within the cluster, recovers failed pods



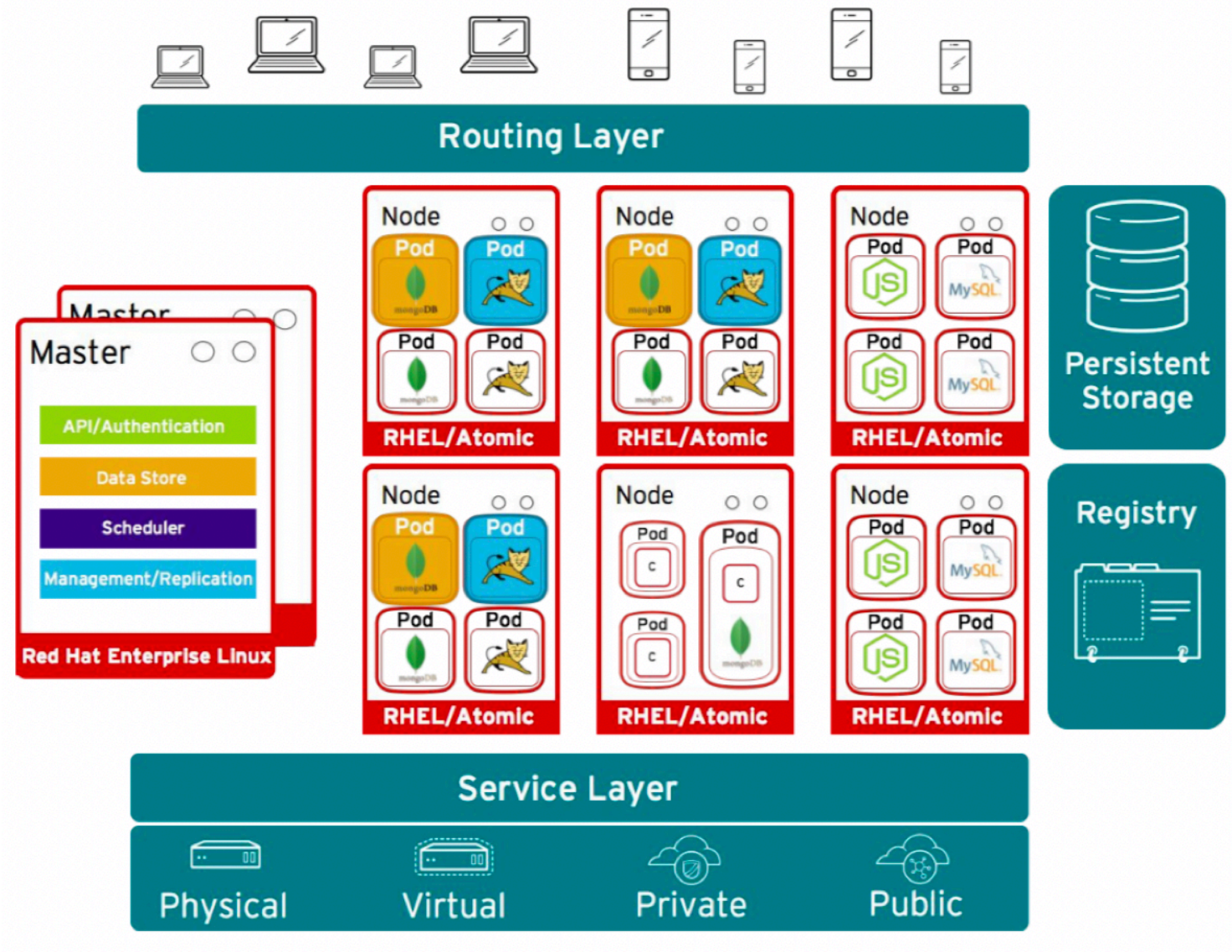
Pods can attach to shared storage for stateful services

- NFS
- FC
- OpenStack
- GlusterFS
- iSCSI, etc

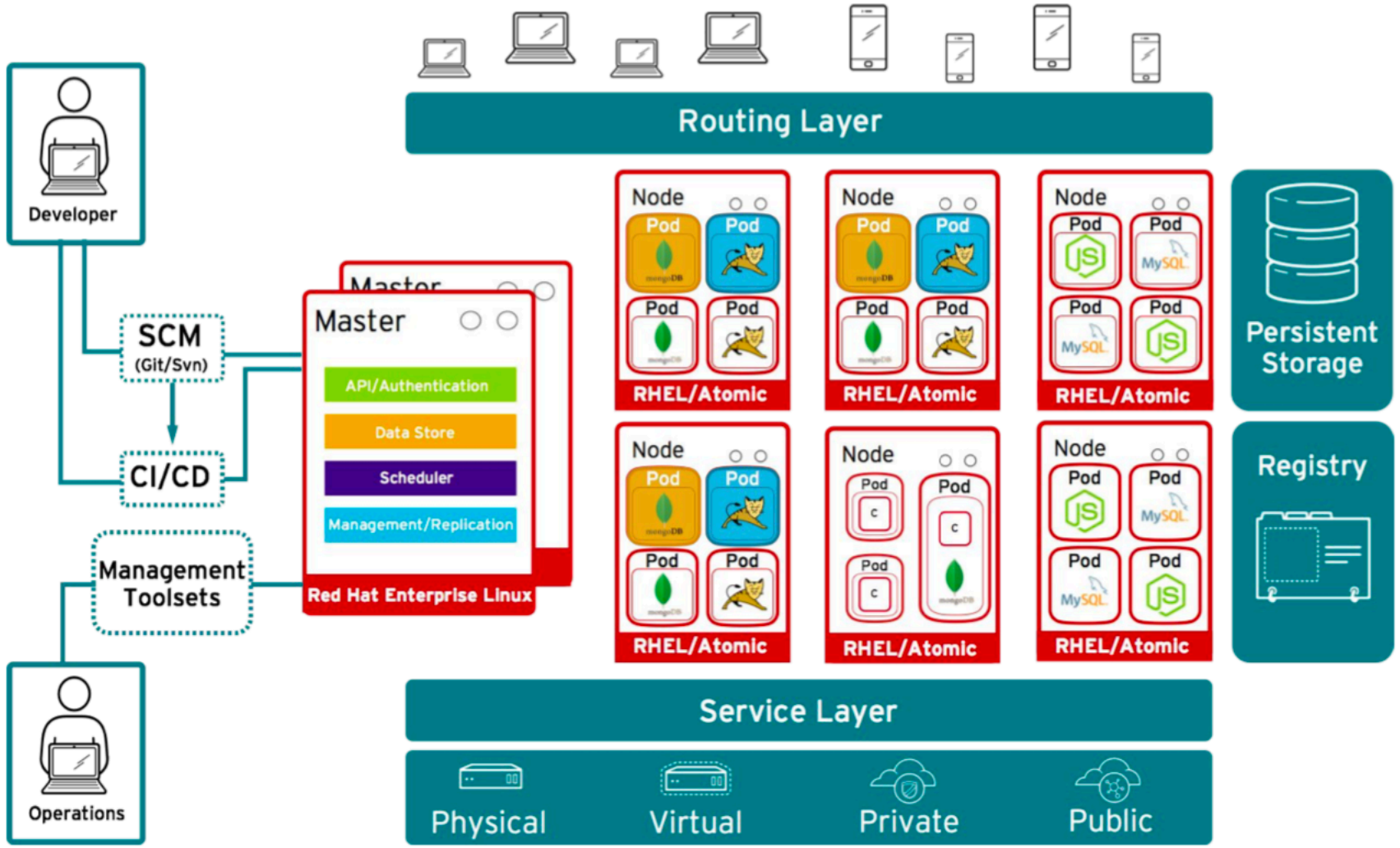


https://developer.ibm.com/storage/2019/12/26/ibm-spectrum-scale-csi-driver-video-blogs/

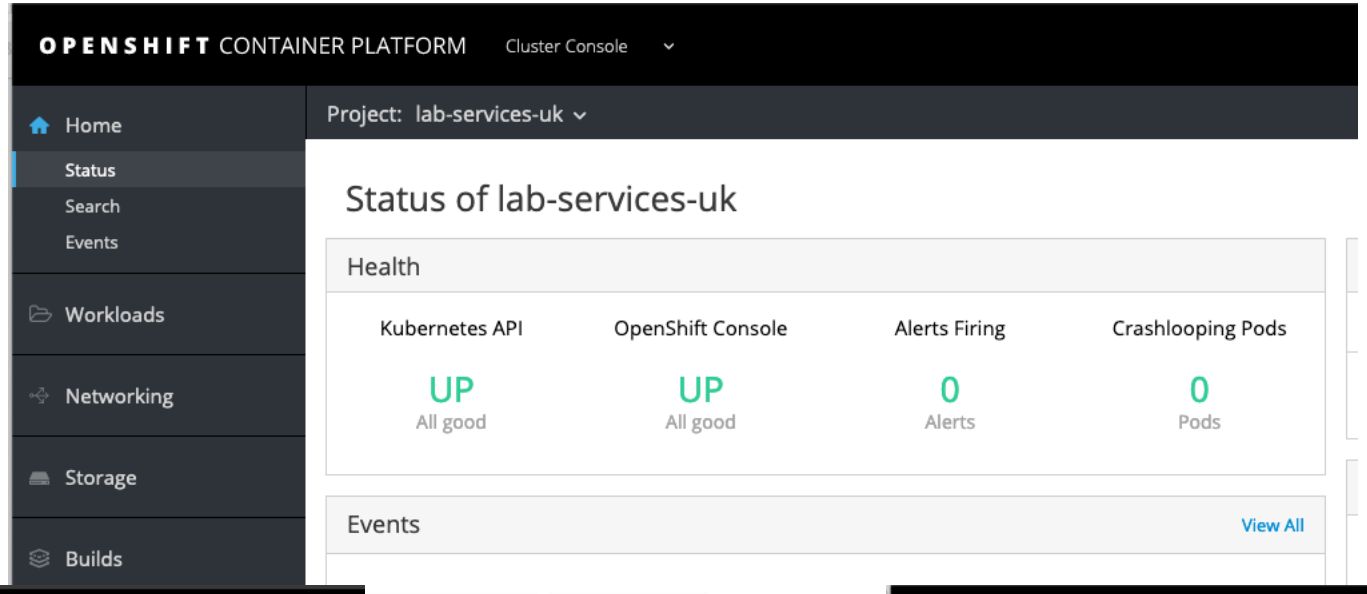
Router layer exposes pods to external users

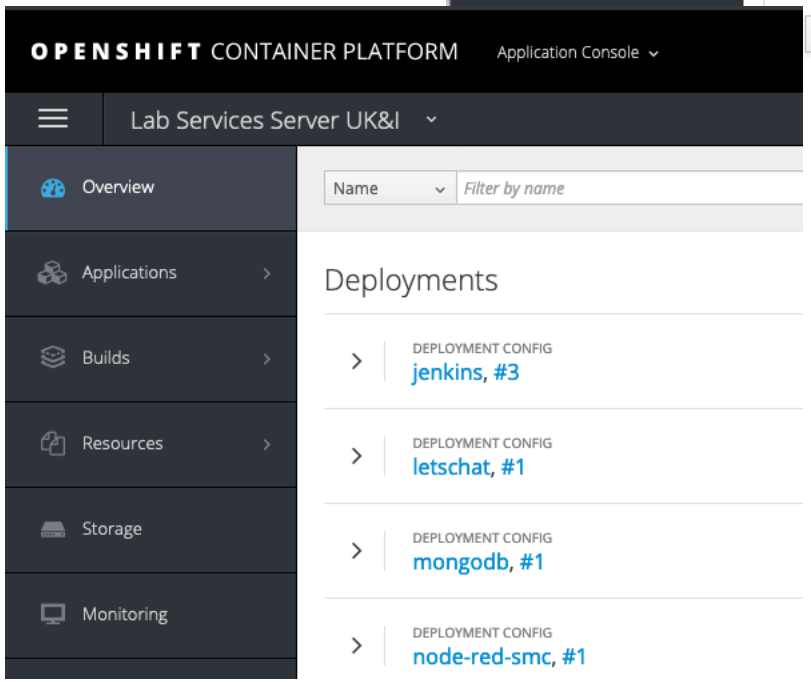


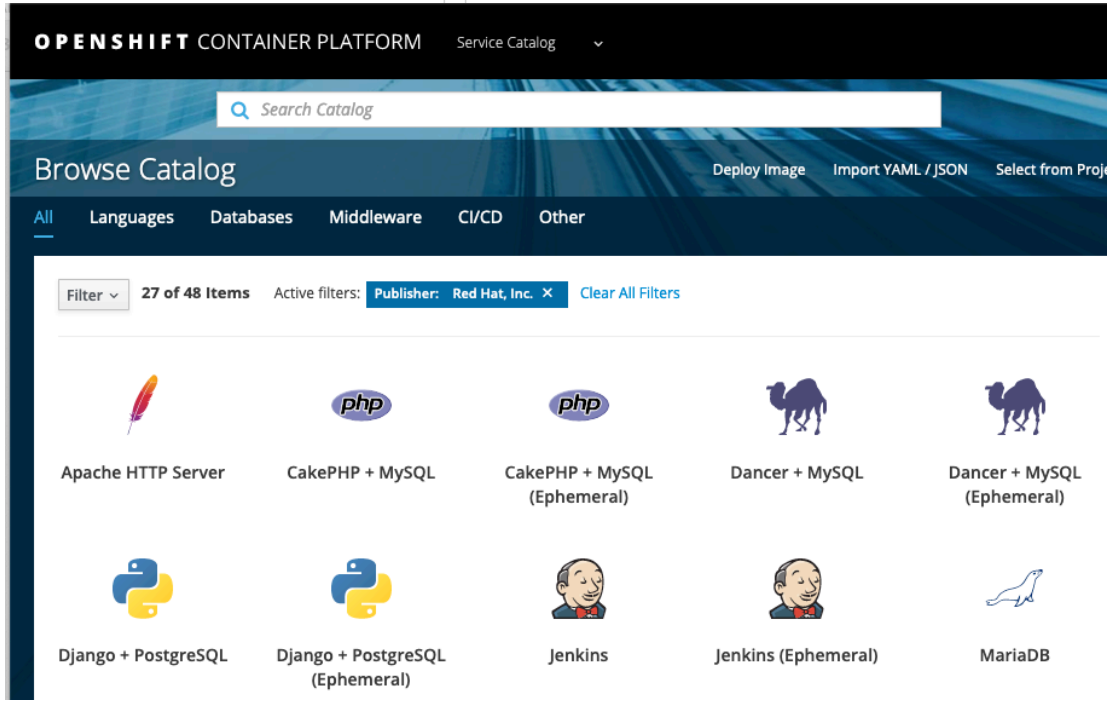
Developers and operators access OpenShift via web/CLI



A look around OpenShift







OpenShift demo

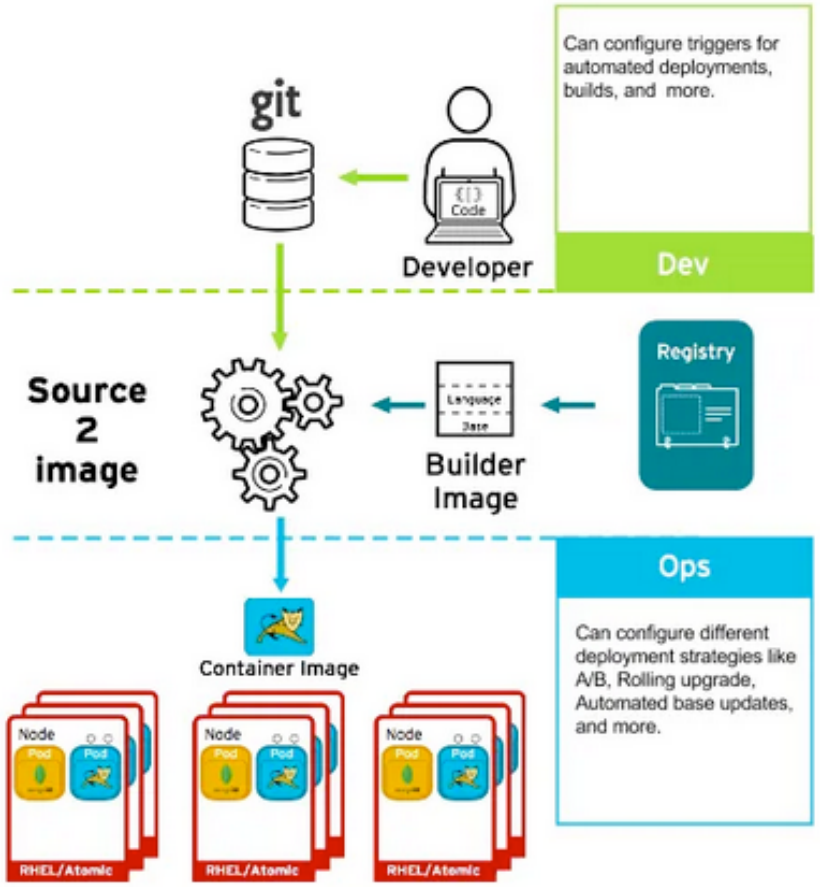
1. Create a template for a new deployment config
   - Uses existing docker image (built using docker build)

2. Use Source2Image inject application configuration into a new deployment
   - Builds a new image within our OCP cluster (S2I)

3. Use IBM Multicloud Manager Cloud Pak and Cloud Automation Manager to deploy VM/LPARs across hybrid multi cloud environment (e.g. public cloud and on-premise IBM PowerVM)
   - Uses prebuilt VM images provided by our cloud environment

OpenShift: Source to Image (S2I)

Combines a runtime (ruby, python, httpd etc) with source code to produce an image to run that code.



Code

Build

Deploy

https://lab-ocp-1.labs.uk.ibm.com:8443/console/project/openshift/browse/images

Use Out-of-Box OpenShift Content for Cloud Native Apps

Databases

- MongoDB, PostgreSQL, MariaDB, MySQL, Redis

Application Servers

- Apache, NGINX

Development Tools

- Node.js, OpenJDK

Programming Languages

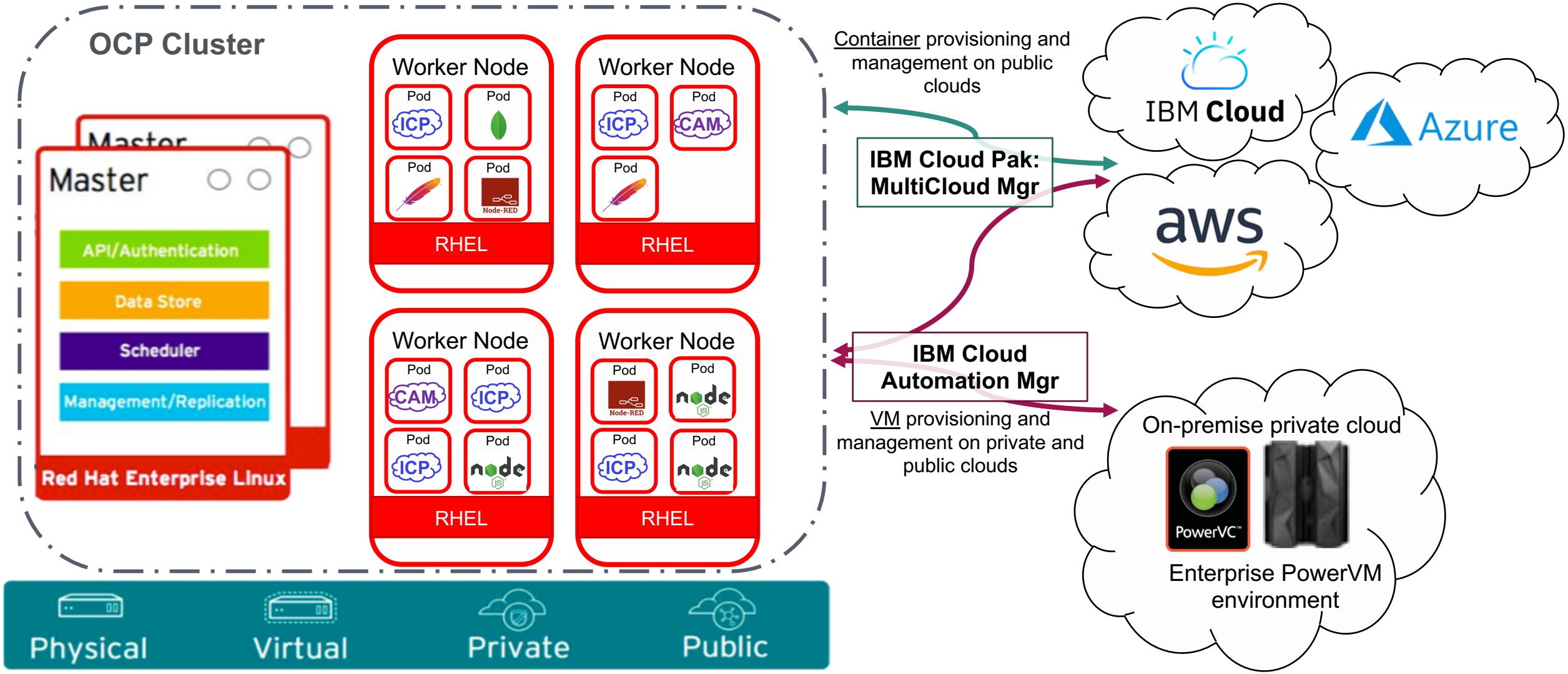
- Python, Go, Perl, Ruby

Automation Tools

- Ansible

…extendable to include your own content too!

High-level view of OpenShift, Hybrid Multicloud and DevOps



Example Apps

Apache: Pod

MongoDB: Pod

NodeJS: Pod

Node-Red: Pod

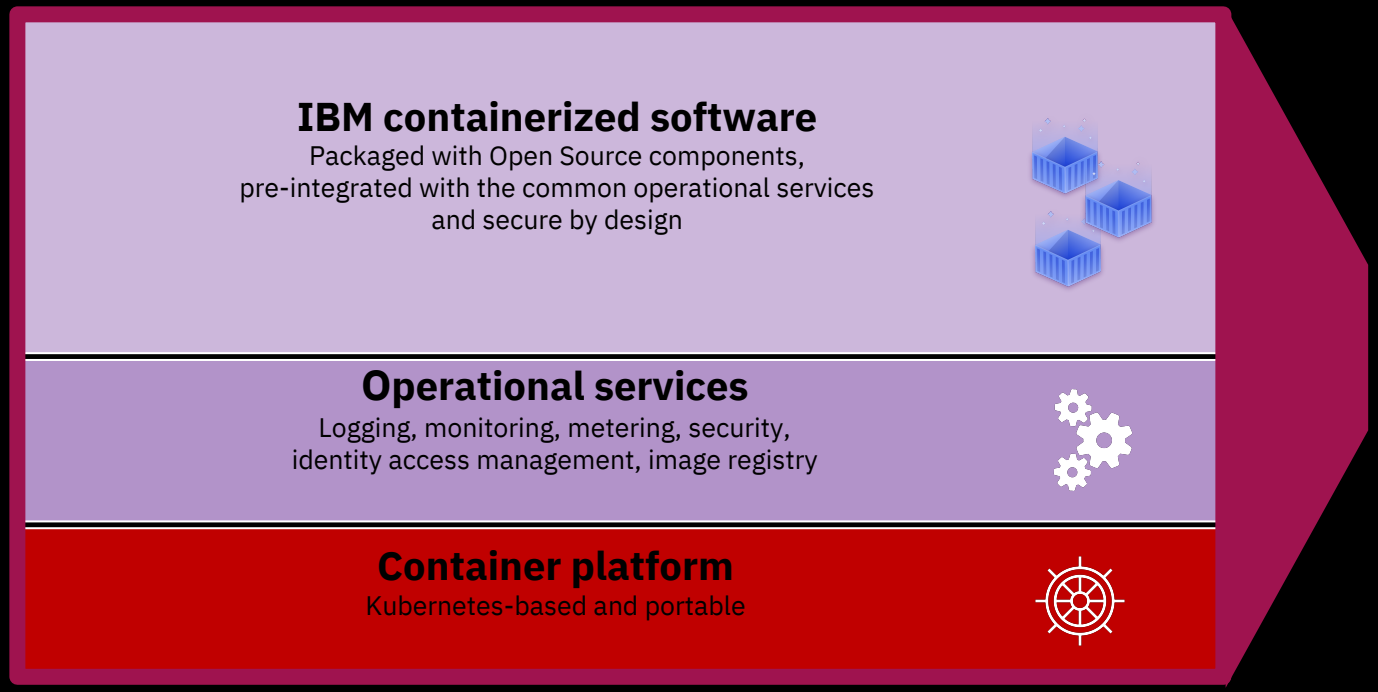
IBM Cloud Pak: (MultiCloud Mgr) Pod ICP

IBM Cloud Automation Manager: Pod CAM

IBM Cloud Paks

IBM Cloud Paks – Middleware Anywhere

Enterprise-ready, containerized software solutions that give you an open, faster, more secure way to move core business applications to any cloud



Complete yet simple
Application, data and AI services, fully modular and easy to consume

IBM certified
Full software stack support and ongoing security, compliance and version compatibility

Run anywhere
On-premises, on private and public clouds and in pre-integrated systems

Pre-integrated Systems · IBM Cloud · RED HAT OPENSHIFT · aws · Azure · openstack · Google Cloud · POWER9

IBM Cloud Paks and Red Hat OpenShift on Power Systems

IBM Cloud Pak for Applications

IBM Application Navigator
IBM Cloud Transformation Advisor
IBM WebSphere Liberty
IBM WebSphere Application Server
Microclimate
Node.js
Open Liberty
Spring
Swift

IBM containerized software

Operational services
ICP 4.1

Container platform
RH OpenShift 3.11

IBM Cloud Pak for Data

IBM Data Virtualization
IBM Cognos Dashboard
IBM Db2 Warehouse
IBM Streams
IBM Unified Governance
IBM Watson Machine Learning
IBM Watson Studio
Python
RStudio
Spark

IBM containerized software

Operational services

Container platform
RH OpenShift 4.x

IBM Cloud Pak for Integration

IBM API Connect
IBM App Connect Enterprise
IBM Aspera
IBM DataPower
IBM Event Streams
IBM MQ

IBM containerized software

Operational services
ICP 4.1

Container platform
RH OpenShift 3.11

IBM Cloud Pak for Automation

IBM Business Automation Content Analyzer
IBM Business Automation Insights
IBM Business Automation Navigator
IBM FileNet Content Manager
IBM Operational Decision Manager

IBM containerized software

Operational services

Container platform
RH OpenShift 4.x

IBM Cloud Pak for Multicloud Management

IBM Cloud App Management (tech preview)
IBM Cloud Automation Manager
IBM Multicloud Manager
IBM Cloud Event Management

IBM containerized software

Operational services
ICP 4.1

Container platform
RH OpenShift 3.11

Runs on choice of IBM Power Systems Infrastructure-as-a-Service (IaaS)

PowerVC
PowerVM

Bare-metal

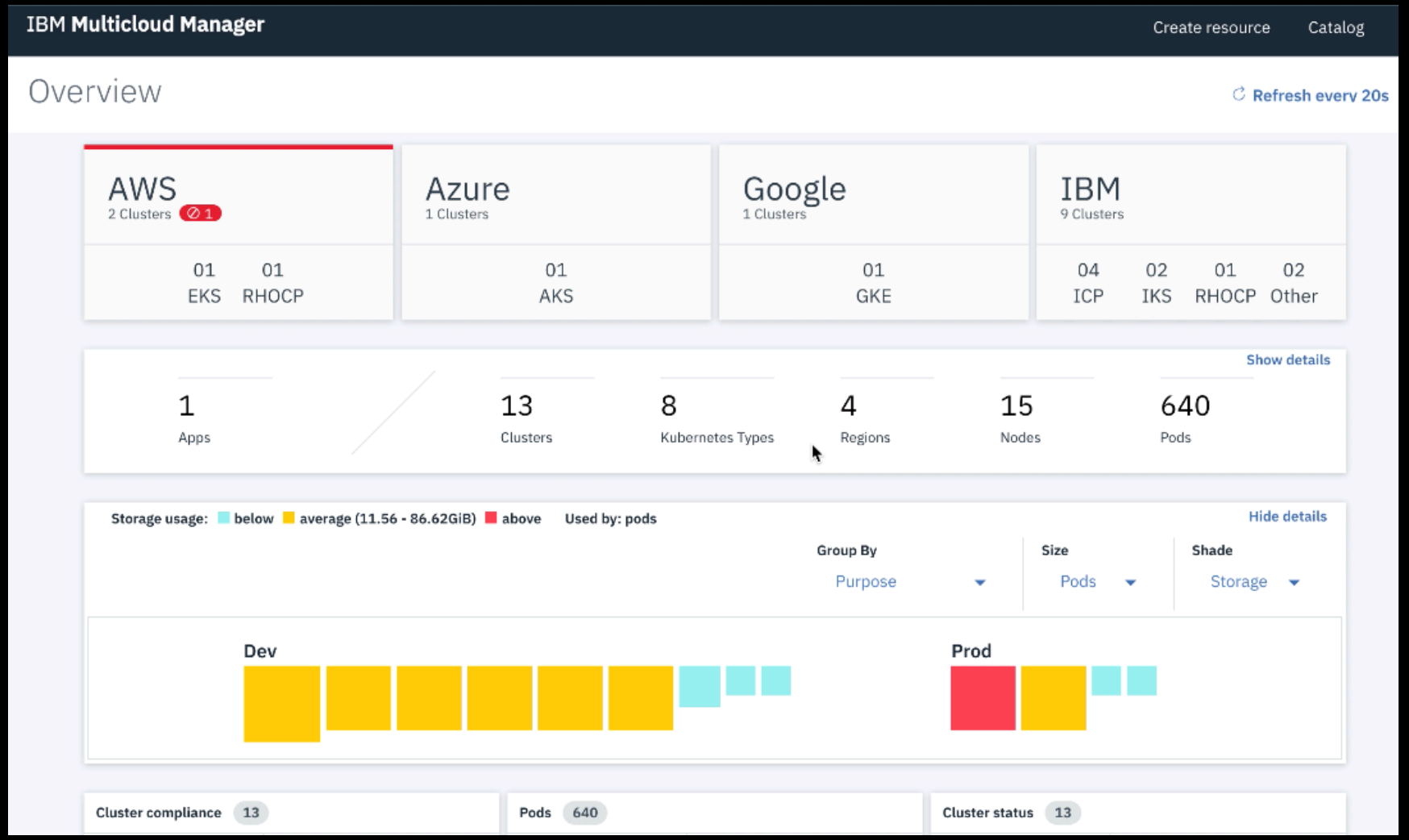
IBM Cloud Pak for Multicloud Management: IBM Multicloud Manager

IBM Multicloud Manager

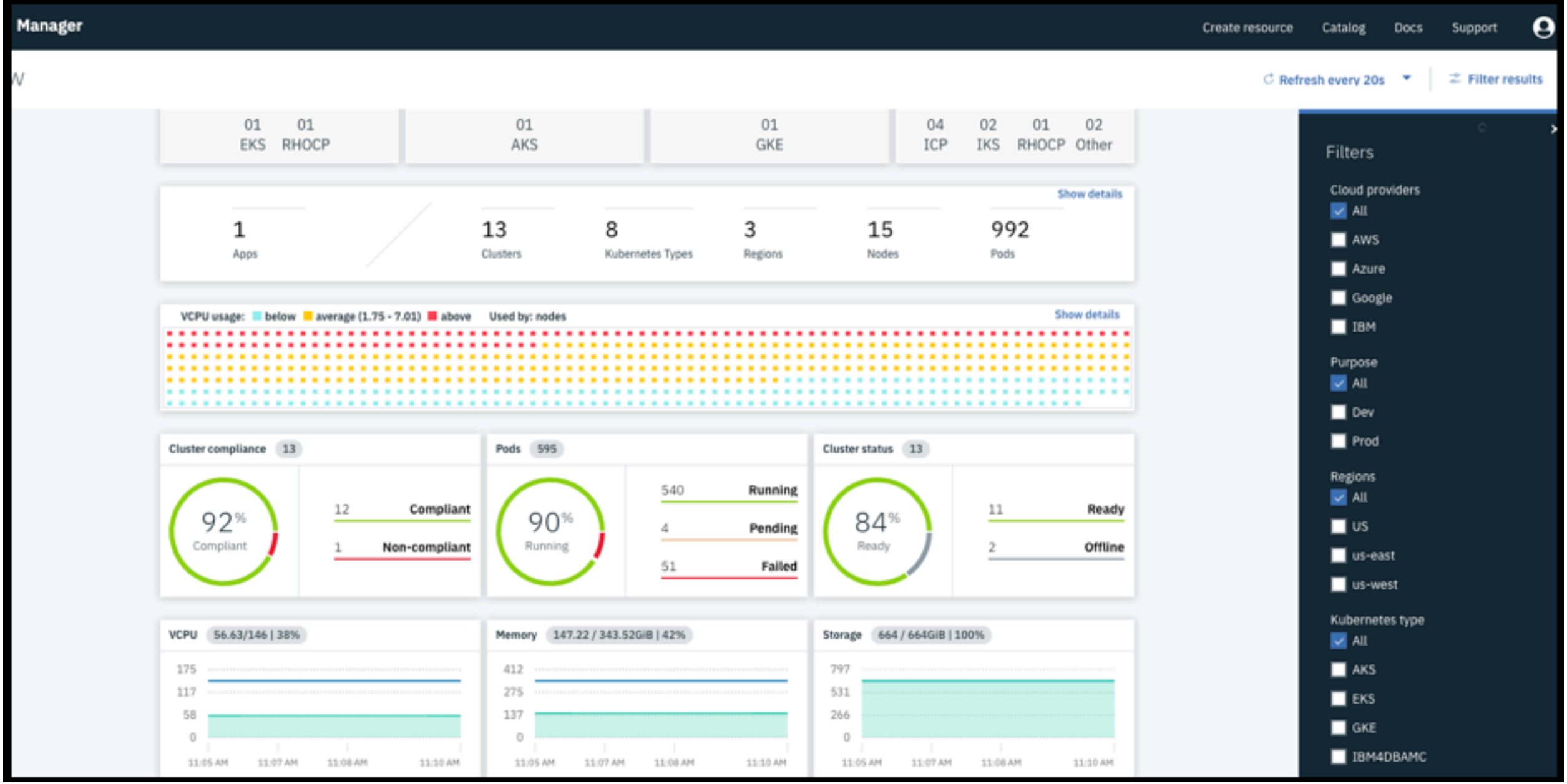
Visibility
- Development teams can see all the deployed pods
- IT Operations can see clusters and nodes

Security and Governance
- Consistent configuration and security policies across cluster

Automation
- Automatically provision, configure and deliver additional Kubernetes clusters in any cloud environment supported by Cloud Automation Manager

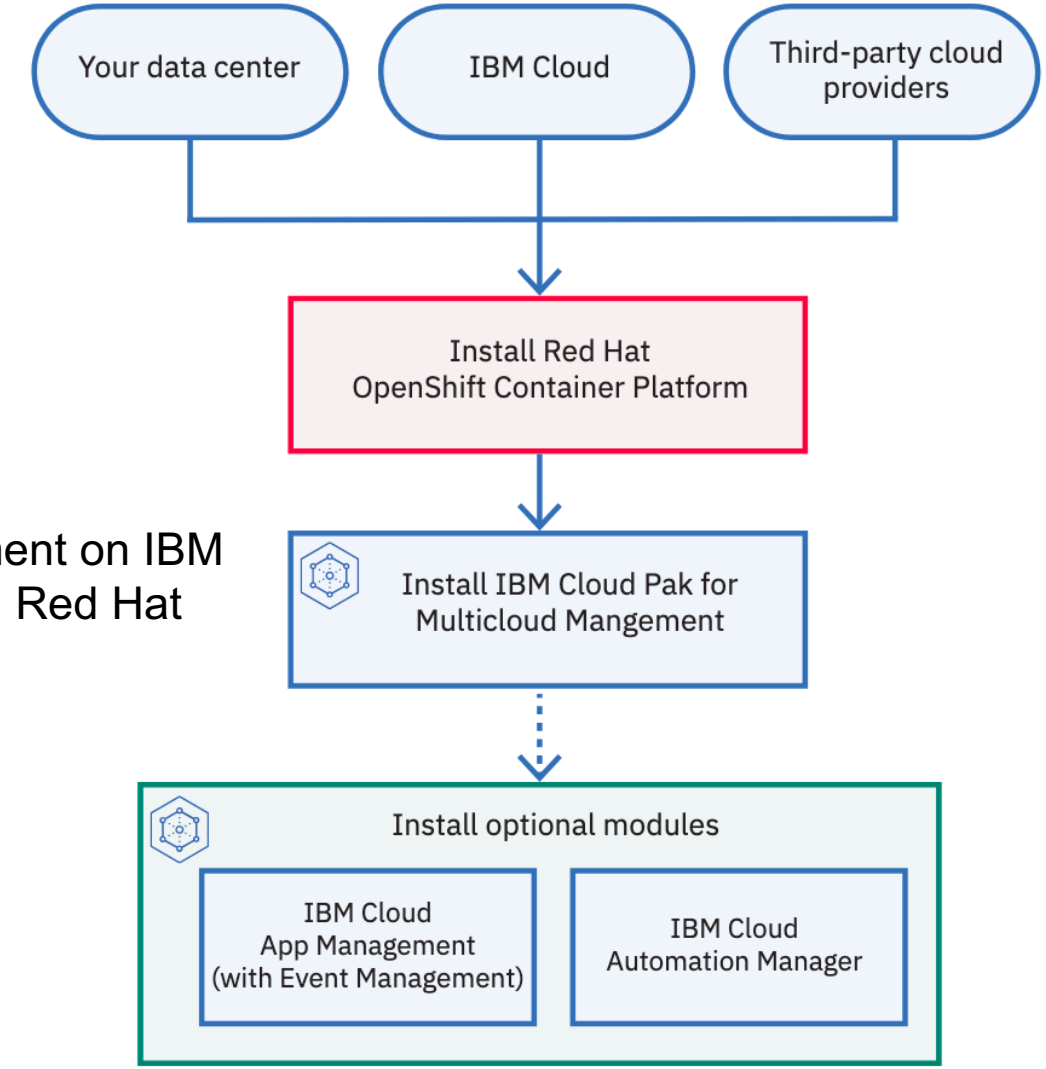


Overview Page: Check compliance & health status of all your clusters



Installing IBM Cloud Pak for Multicloud Manager

You can install the IBM Cloud Pak for Multicloud Management on IBM Cloud, Linux x86_64, and Linux on Power (ppc64le) with Red Hat OpenShift Container Platform version 3.11.



Installing IBM Cloud Pak for Multicloud Manager

https://www.ibm.com/support/knowledgecenter/SSFC4F_1.1.0/cp/installing/cloud_pak_foundation.html

- Prepare OpenShift cluster – numerous pre-reqs (listed above)
- Untar the MCM package and load images into local docker repositories on OCP cluster
- Extract the main cluster directory (docker run)
- Update the config file to point at the appropriate OCP node:

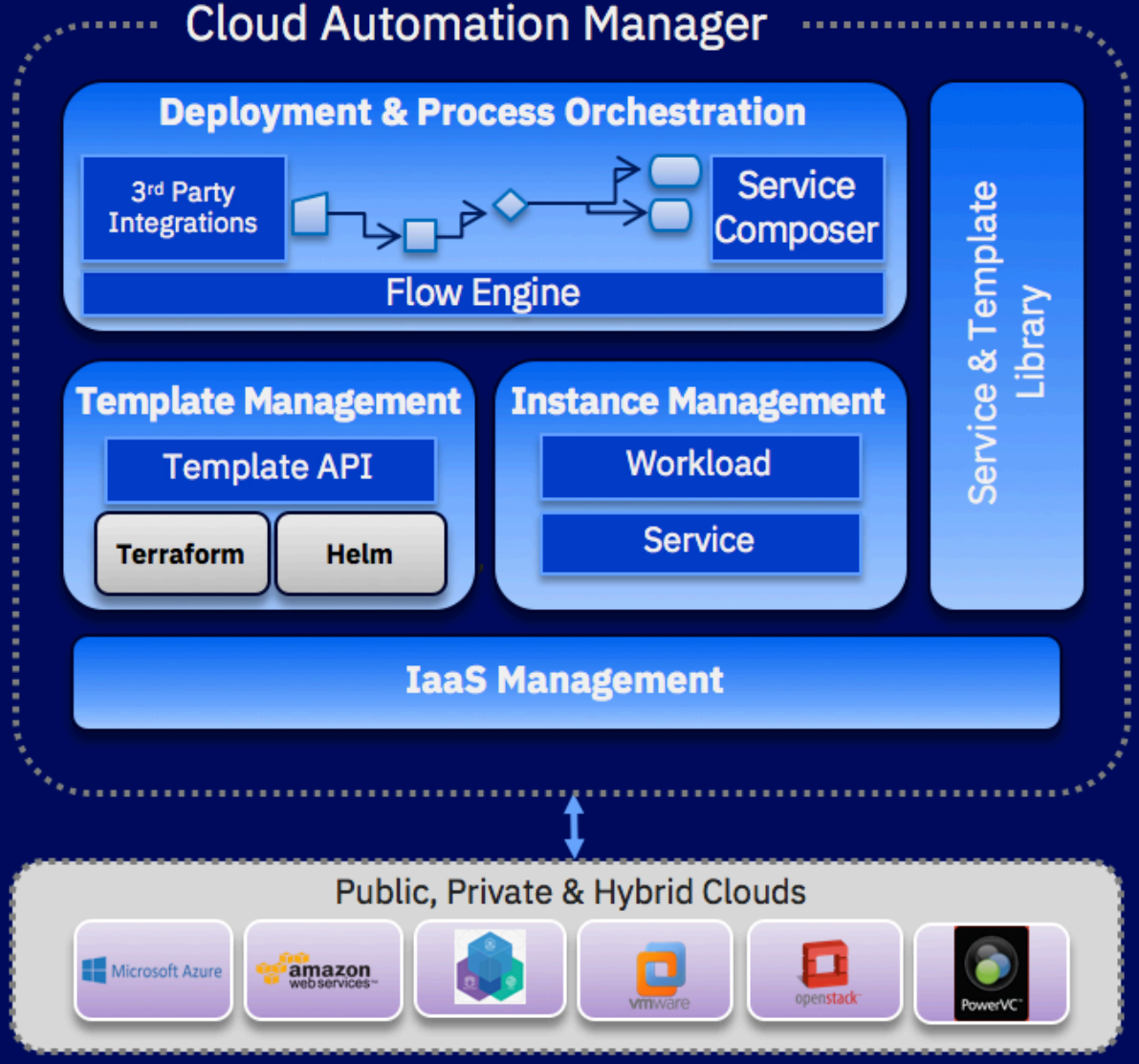
```
---
# A list of OpenShift nodes that used to run ICP components
cluster_nodes:
  master:
    - lab-ocp-1.labs.uk.ibm.com
  proxy:
    - lab-ocp-1.labs.uk.ibm.com
  management:
    - lab-ocp-1.labs.uk.ibm.com
  va:
    - lab-ocp-1.labs.uk.ibm.com
storage_class: ibm-powervc-k8s-volume-default
```

- Use docker run to install MCM Cloud Pak

IBM Cloud Pak for Multicloud Management: Cloud Automation Manager

IBM Cloud Automation Manager
Full stack automation and service orchestration

- ***Automated provisioning*** *– Automated provisioning of infrastructure and applications with workflow orchestration*
- ***Self-service*** **-** Self-service access to cloud infrastructure and application services
- ***Manage and govern*** *– Manage and govern workloads across multiple and hybrid clouds*
- ***Built with open technology*** *to avoid vendor lock-in*



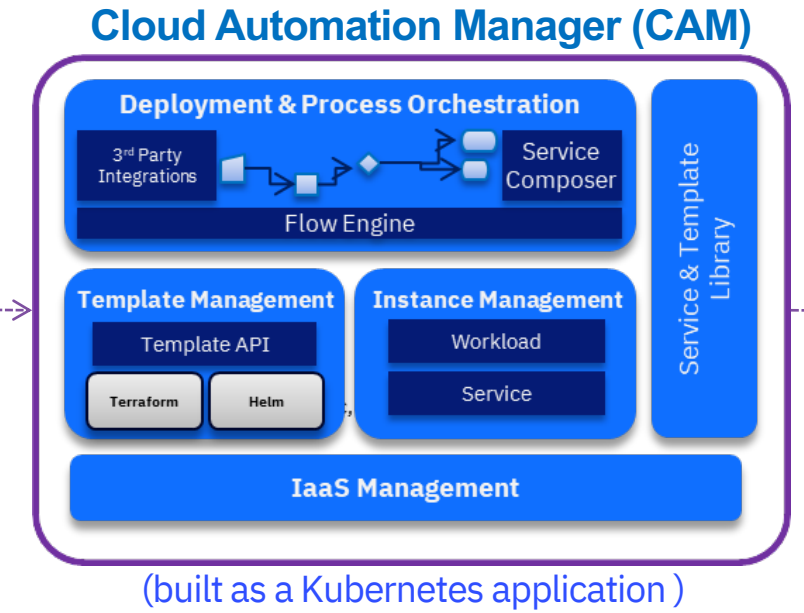
Cloud-enable your Mission Critical Applications – including existing VM apps running on AIX, IBM i, Linux, VMware and more

Traditional Enterprise Apps

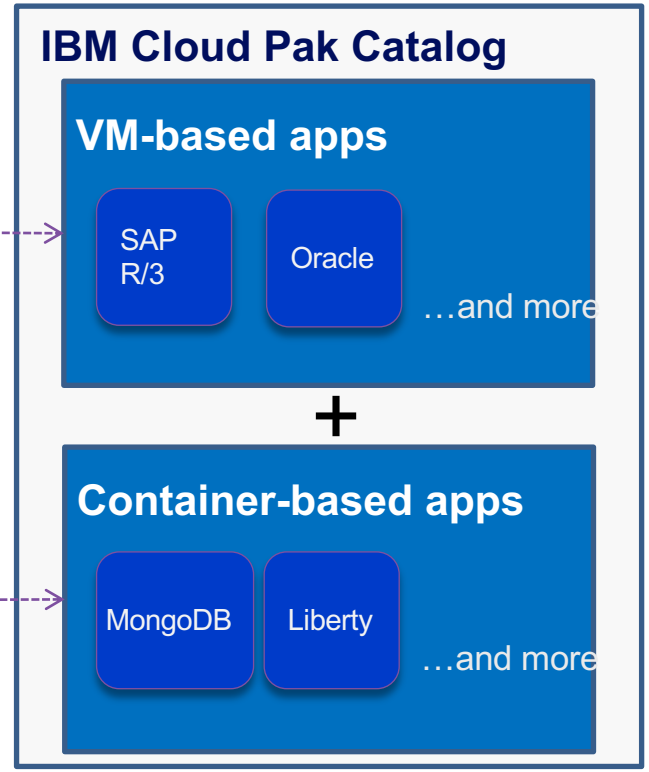
WebSphere App Srv
WebSphere MQ
SAP R/3
Oracle
Db2

Container Apps

WebSphere Liberty
MongoDB
Redis
Jenkins



- Create app templates
- Define service compositions
- Uses standard Terraform to deploy to any cloud
- Standard Helm charts for consistent operations



Cloud Automation Manager

- cam-bpd-cds
- cam-bpd-mariadb
- cam-bpd-ui
- cam-broker
- cam-iaas
- cam-mongo
- cam-portal-ui
- cam-provider-helm
- cam-provider-terraform
- cam-proxy
- cam-service-composer-api
- cam-service-composer-ui
- redis
- cam-ui-basic
- cam-ui-connections
- cam-ui-instances
- cam-ui-templates
- cam-orchestration
- cam-tenant-api

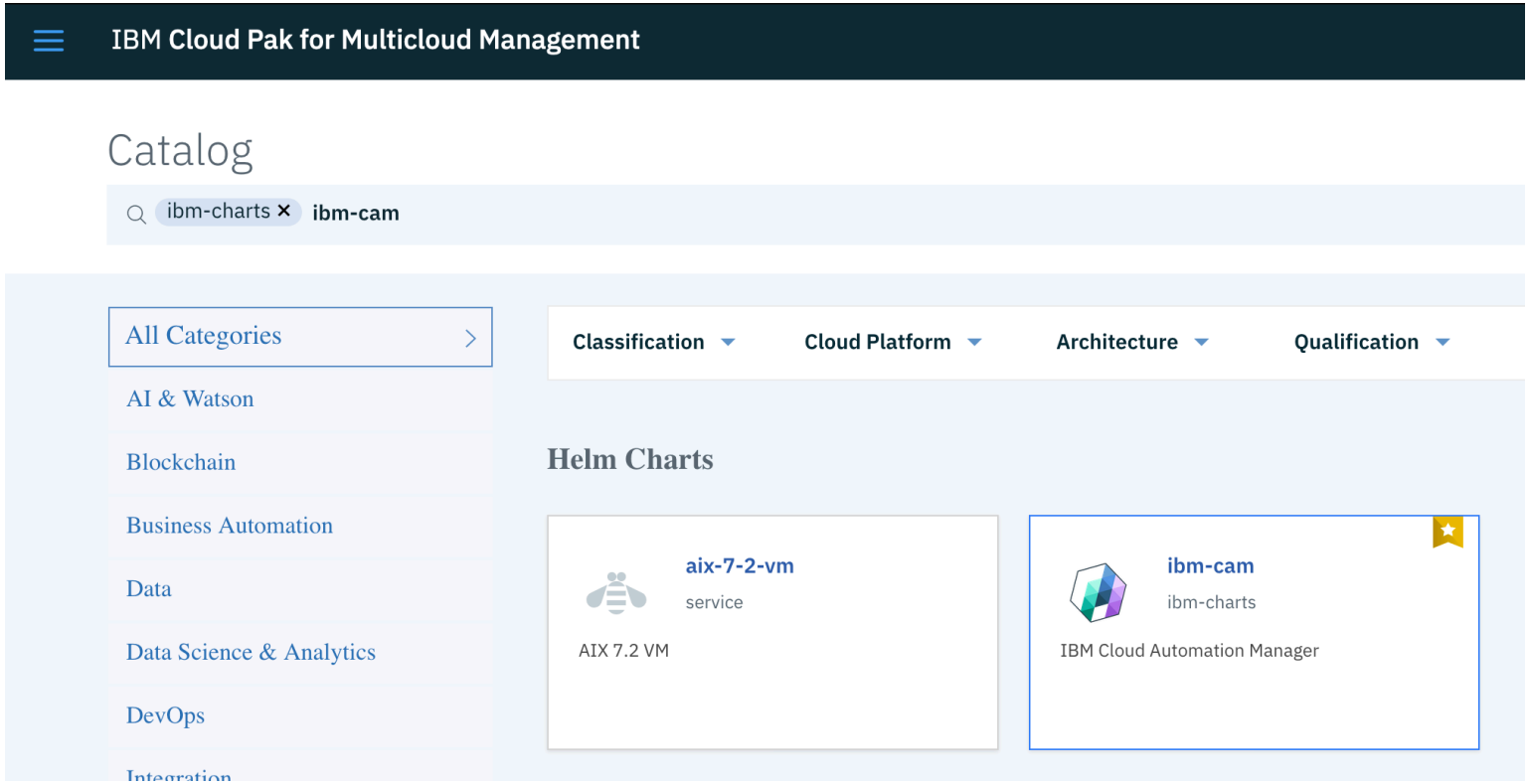
Delivered and managed as a single application with IBM Cloud Pak

Installing Cloud Automation Manager

IBM Cloud
Automation Manager

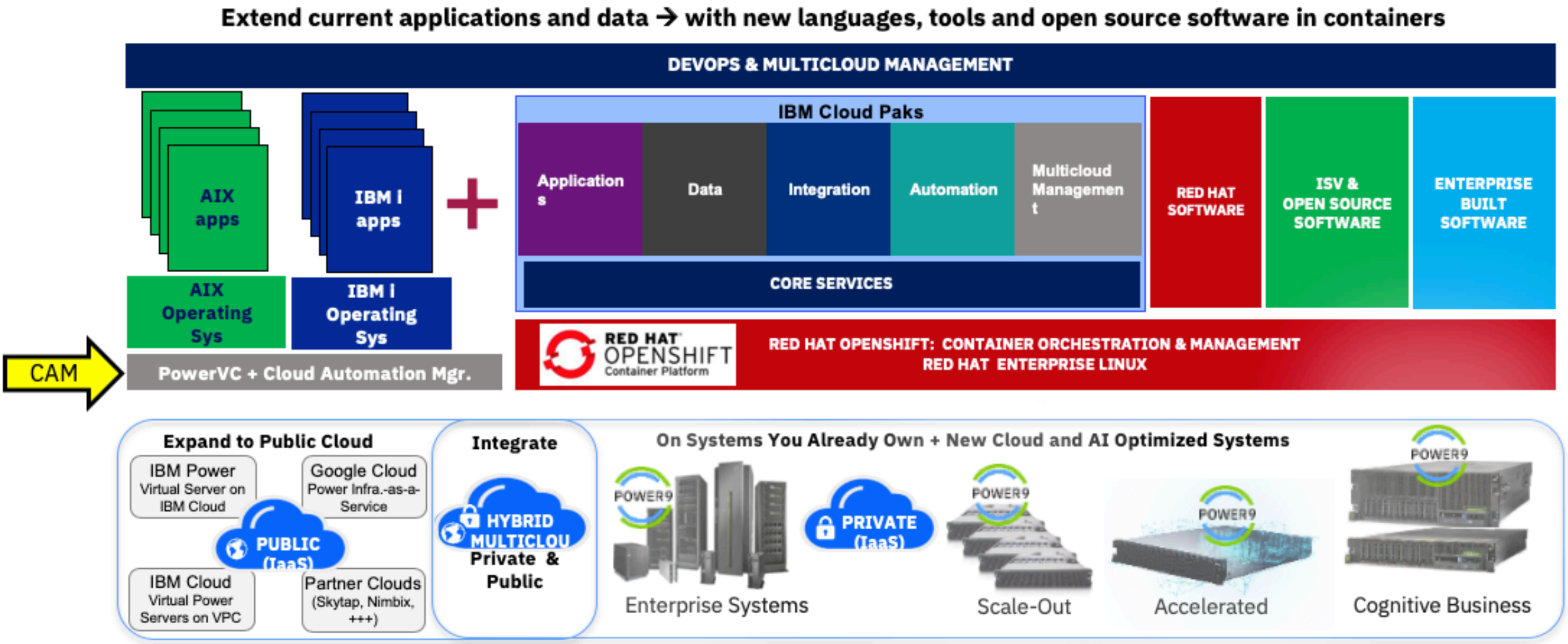
https://www.ibm.com/support/knowledgecenter/SS2L37_3.2.1.0/cam_install_offline_icpos.html?cp=SSFC4F_1.1.0

- Download the CAM tar file from PPA
- Untar the MCM package and load images into local docker repositories on OCP cluster
- Login to the ICP interface on the cluster and select the new helm chart:



IBM Hybrid Cloud on POWER – Enterprise IT + Cloud

Modernise AIX and IBM i apps on POWER



Modernization and Cloud Journey for POWER

- **Seamless management** of IT infrastructure, multiple clouds and applications

Traditional IT Infrastructure

+

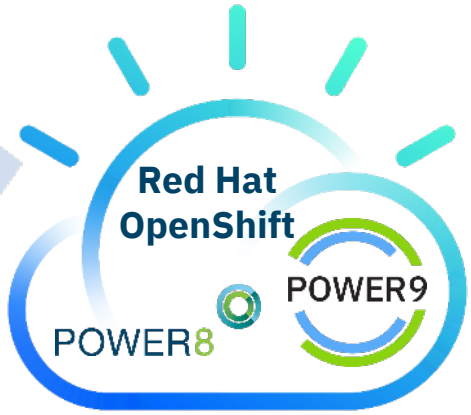
Manage Hardware and Ops
HMC

+

Manage Cloud Infrastructure
PowerVC Cloud

+

Create and Manage Cloud Native Apps
Red Hat OpenShift
+ Cloud Automation Mgr.



- ✓ Build on existing PowerVM and KVM hypervisors
- ✓ Use PowerVC to manage Power Cloud infrastructure
- ✓ Modernize existing, build new apps on POWER and x86
- ✓ Securely connect to business critical data and apps
- ✓ Provide cloud automation and single self-service portal for AIX and IBM i VMs plus Linux containerized apps

Thank you

email: s_cunliffe@uk.ibm.com
Twitter: @StuCunliffe

Notices and disclaimers

Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml