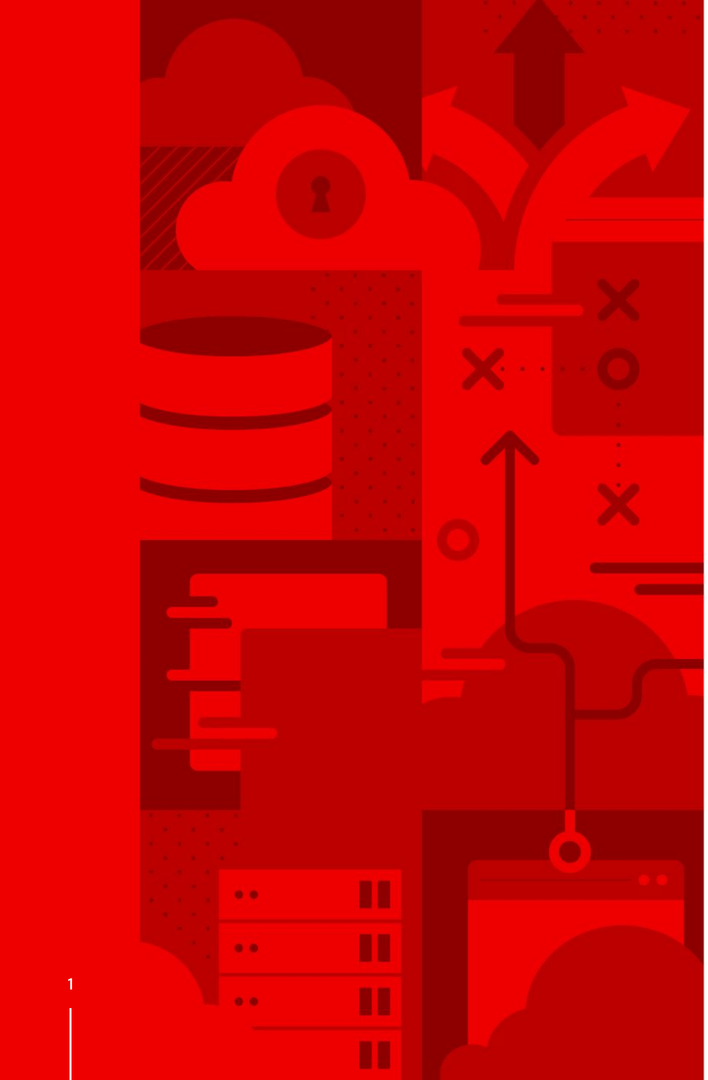
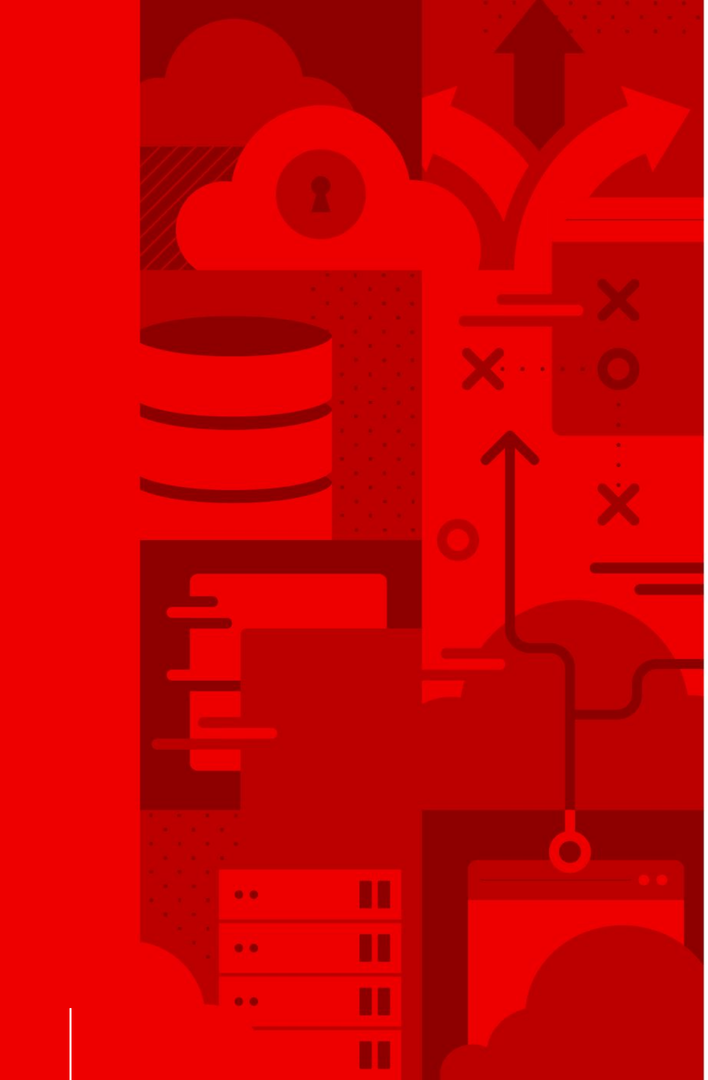


OpenShift Virtualization

Bringing Virtualization into Kubernetes

Dan Kenigsberg



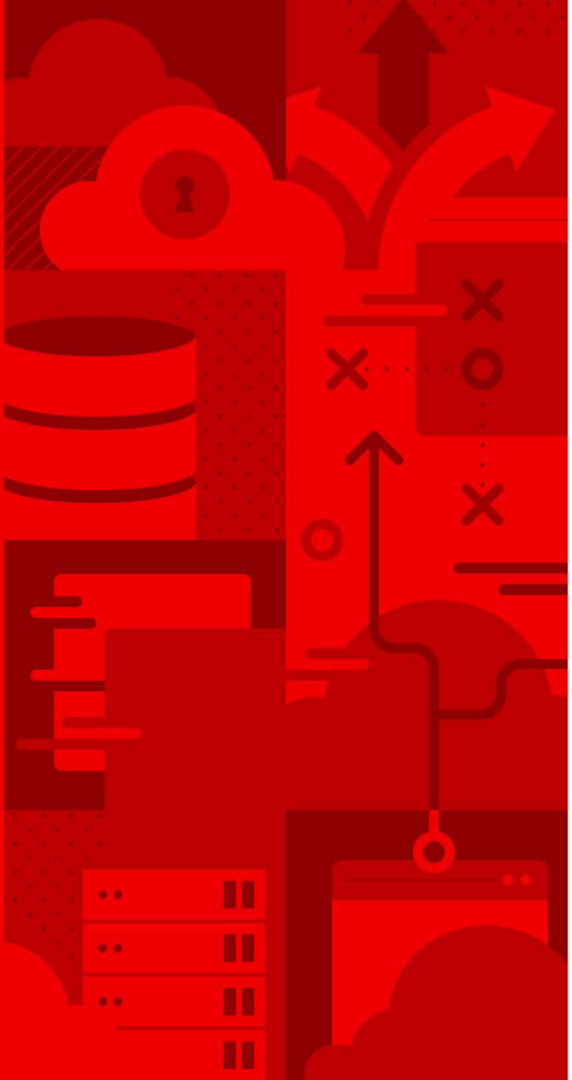
Dan Kenigsberg

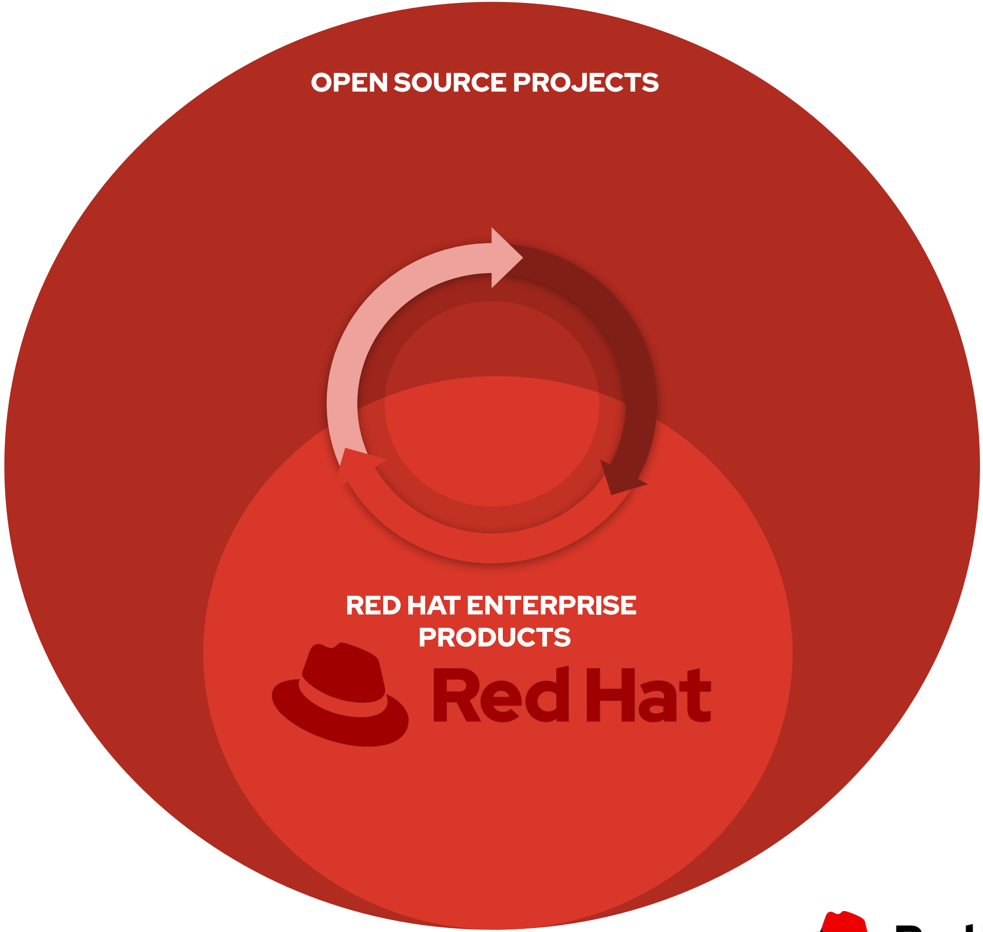
PhD, Computer Science, Technion

Director, Software Engineering, Red Hat

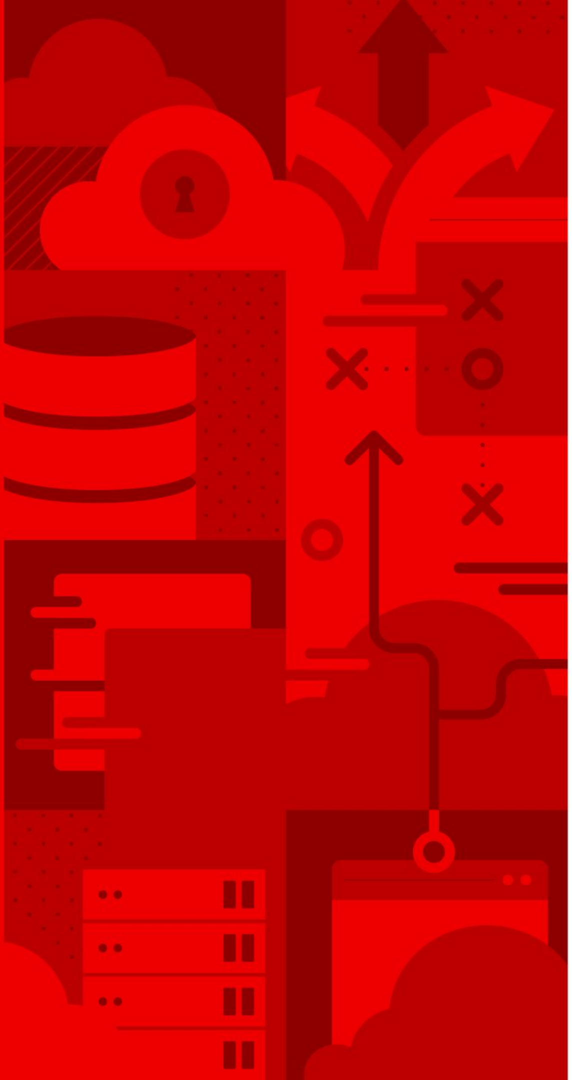
OpenShift Virtualization

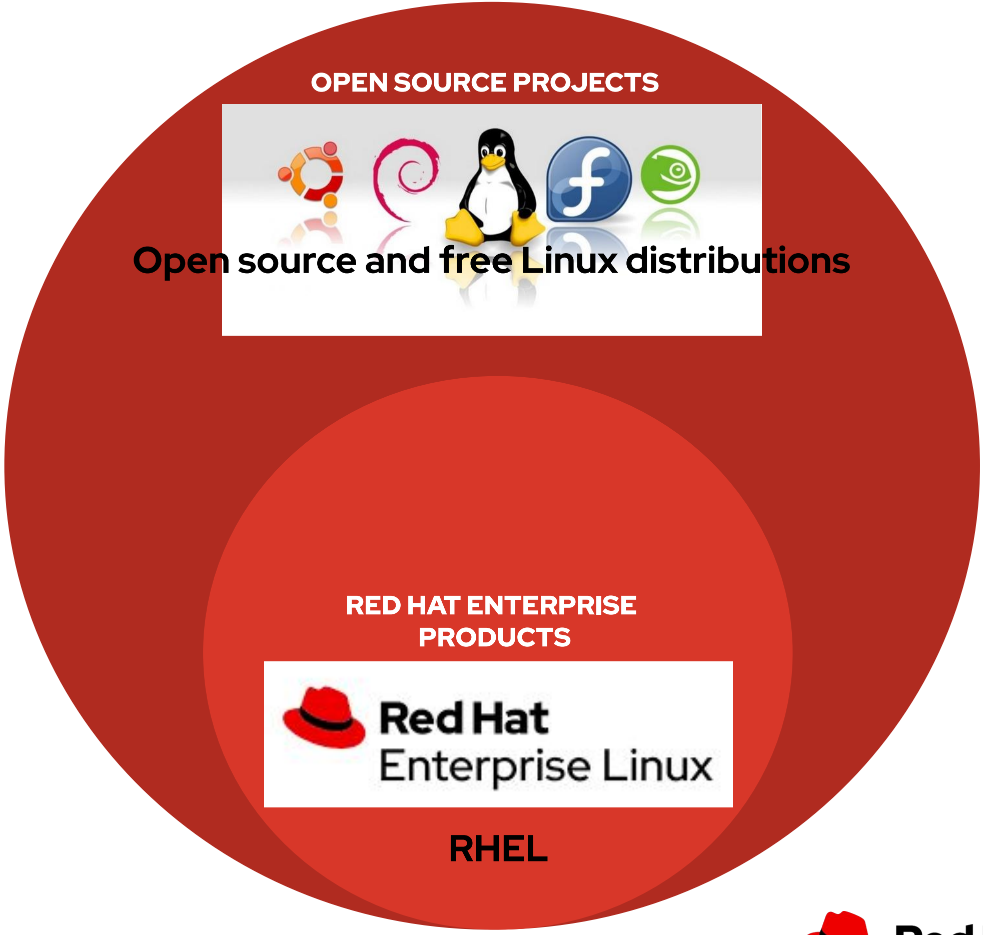
What are Red Hat products?





What are Red Hat products?





OPEN SOURCE PROJECTS

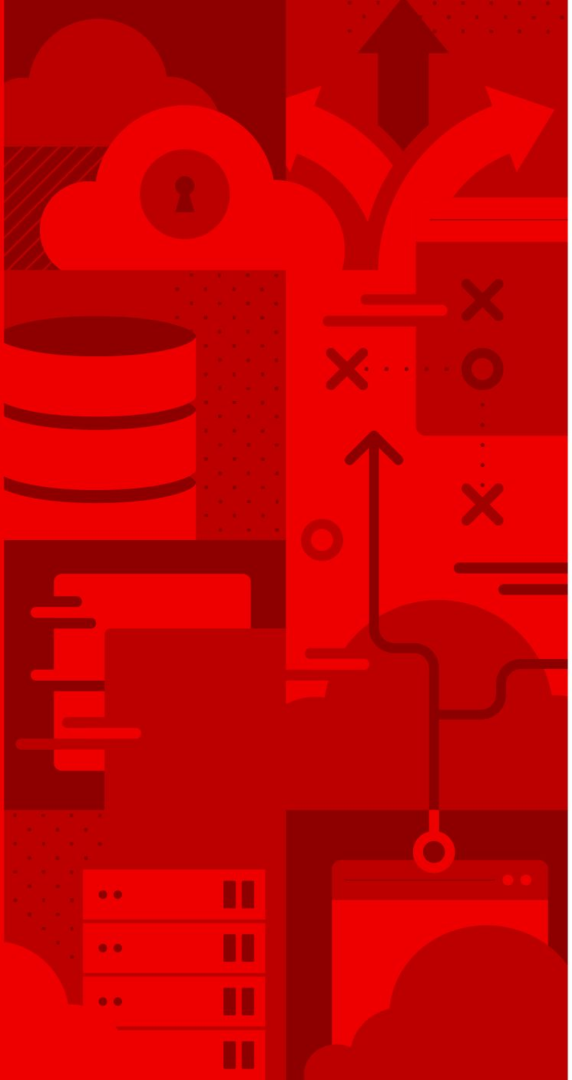
Open source and free Linux distributions

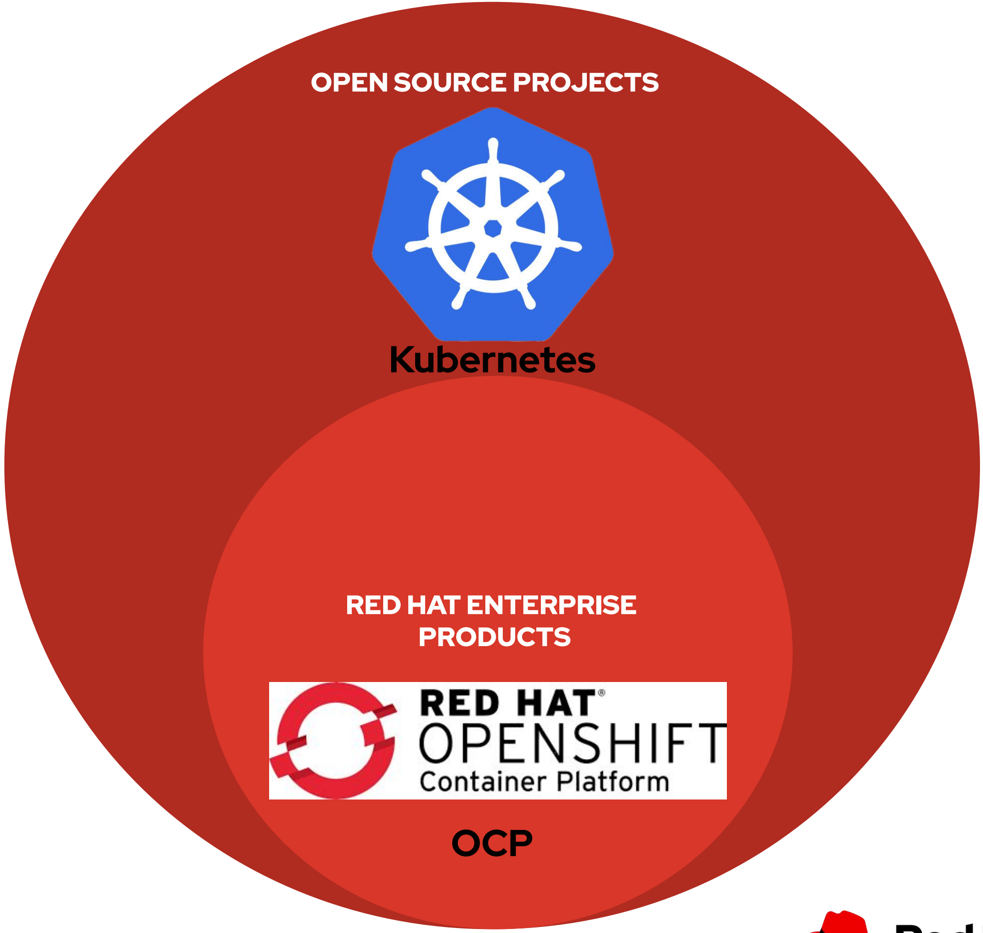
RED HAT ENTERPRISE PRODUCTS

Red Hat Enterprise Linux

RHEL

What are Red Hat products?





OPEN SOURCE PROJECTS

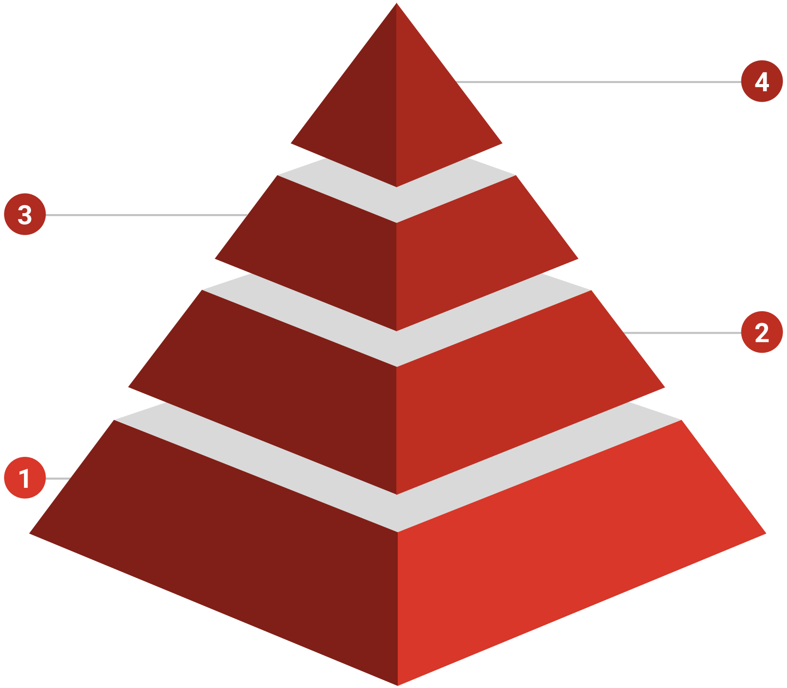
Kubernetes

RED HAT ENTERPRISE PRODUCTS

RED HAT OPENSHIFT Container Platform

OCP

Up from a single-node Linux to Orchestration



Multiple nodes running containers

Scheduling, master election, life cycle, etc.

Single Node: Linux

Processes, Scheduler, File System

Kubernetes

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services

Containers

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

Process → Container → Container Image

What is a container?

Process
Container
Container + Image

Process has an owner, an address space, an access to the file system, scheduling, priorities

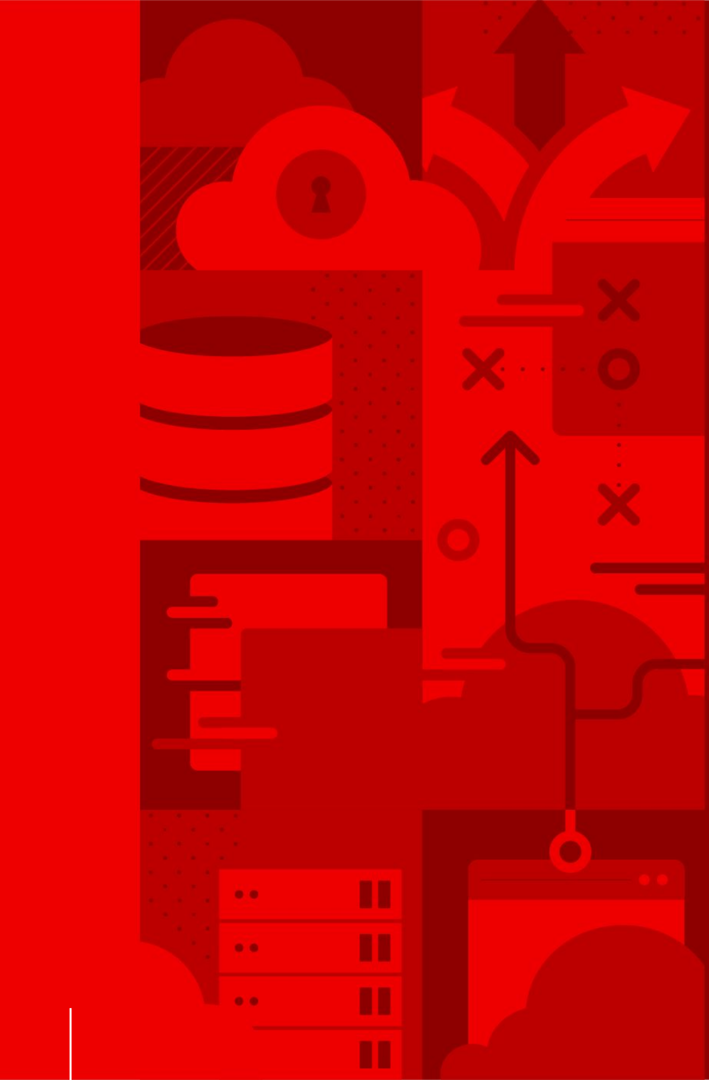
Process group (container might have multiple processes inside) with limitations:

on the file system, on the address space and memory amount, on the network namespace (only part of the network interface can be accessed/seen), on the CPU usage.

Container image is a convenient way to define what is going to be the file system of the container. That allows us to use the same code thousands of times.

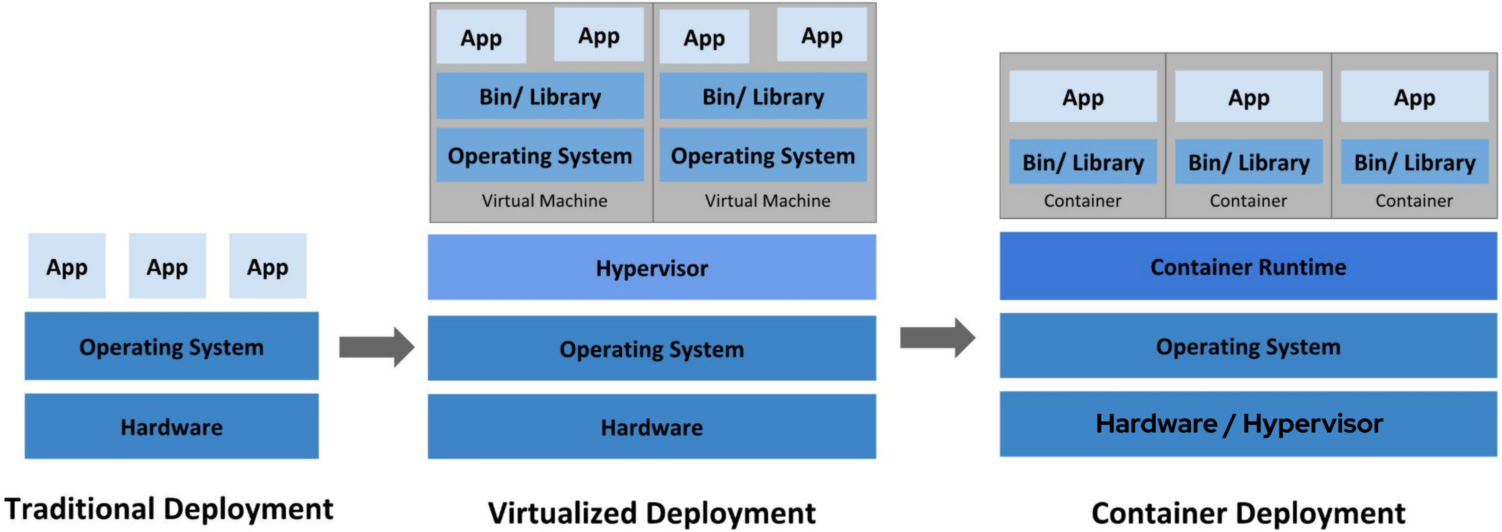
Benefits?

The main benefits is the reproducibility and scalability and agile development!



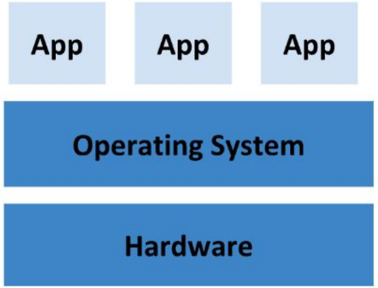
Deployment Variations

Deployment evolution



As we have seen in OS course

Traditional Deployment



Early on, organizations ran applications on physical servers.

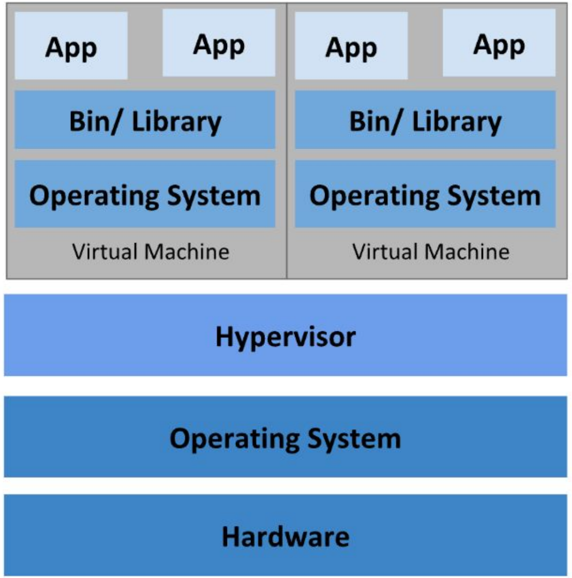
No way to define resource boundaries for applications in a physical server, and this caused resource allocation issues.

For example, if multiple applications run on a physical server, there can be instances where one application would take up most of the resources, and as a result, the other applications would underperform.

A solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

As we have seen at the last lecture in OS course

Virtualized Deployment



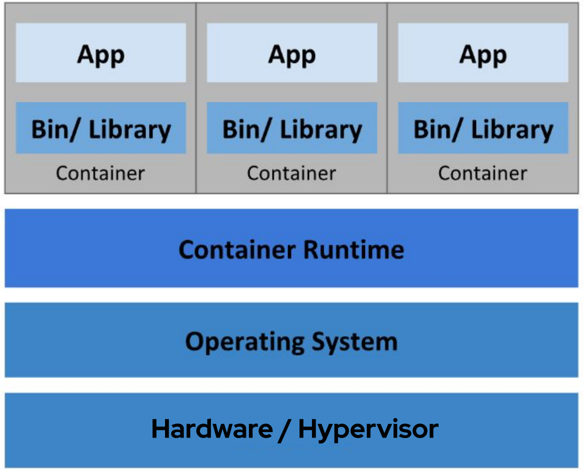
As a solution, virtualization was introduced.

It allows you to run multiple Virtual Machines (VMs) on a single physical server CPU. Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more. With virtualization you can present a set of physical resources as a cluster of disposable virtual machines.

Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Container Deployment



Containers are similar to VMs, but they have relaxed isolation properties **to share the Operating System** (OS) among the applications.

Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, share of CPU, memory, process space, and more.

As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.

Bare metal less, virtual machines more

Containers usually run on

Google cloud, AWS, Asure, etc.

And sometimes on the bare metal

Multiple nodes

Horizontal scaling

100 containers → 10 000 000 containers

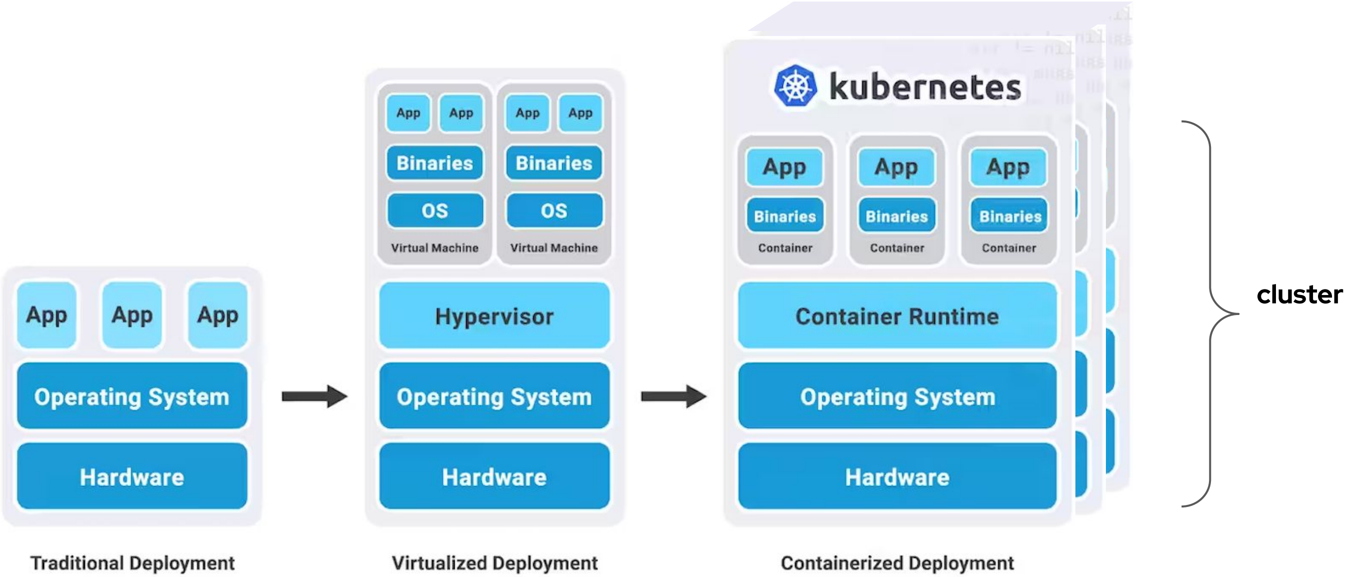
From one worker node to thousand workers

How would you manage so many resources?

Failures? Bugs? Versions control? Upgrades?

Manually... Or... Kubernetes :)

Kubernetes enters the deployments scene!

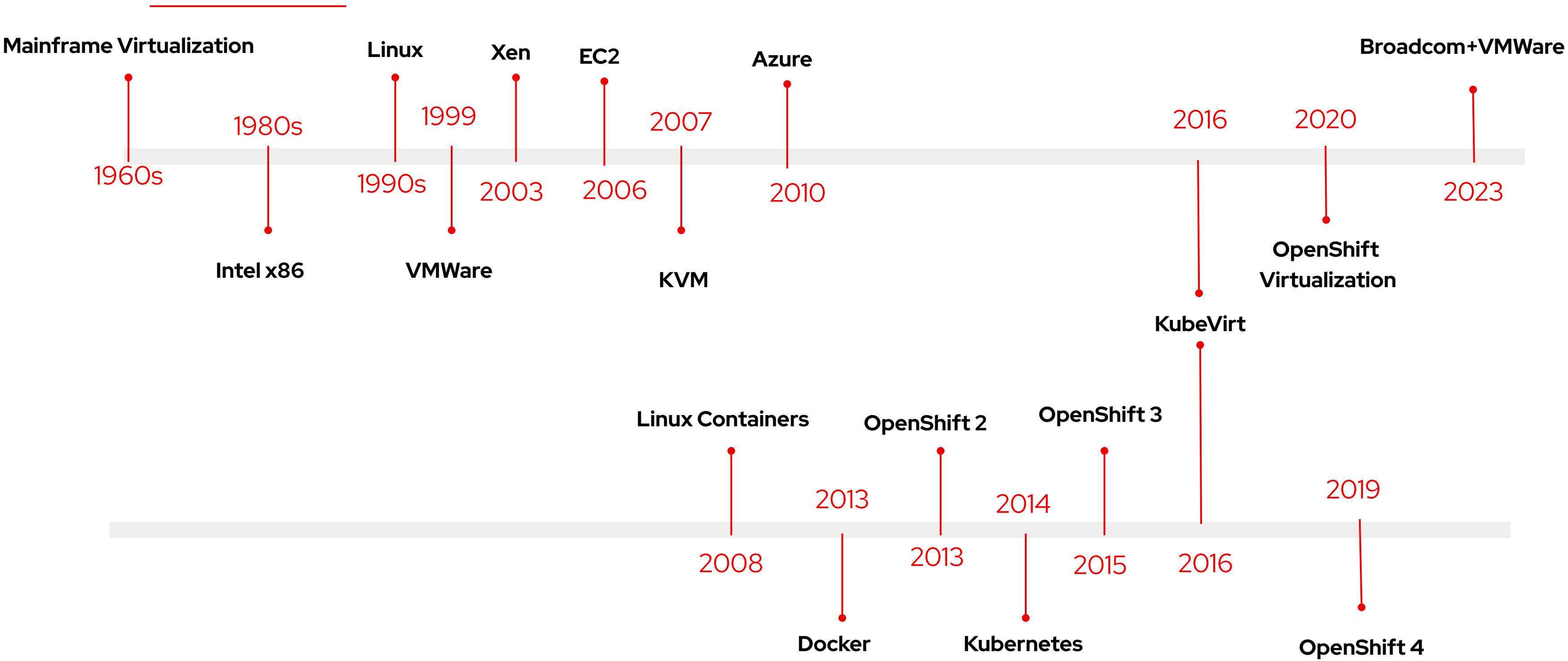


Link to the source of the image

Virtualization Timeline

Containers Timeline

A Partial History of Virtualization



1960s Mainframe Virtualization

1980s Intel x86

1990s Linux

1999 VMWare

2003 Xen

2006 EC2

2007 KVM

2010 Azure

2016 KubeVirt

2020 OpenShift Virtualization

2023 Broadcom+VMWare

2008 Linux Containers

2013 Docker

2013 OpenShift 2

2014 Kubernetes

2015 OpenShift 3

2016

2019 OpenShift 4

Kubernetes

(κυβερνήτης, קברניט)

What is Kubernetes?

- Kubernetes offers a consistent interface for both developers and administrators.
- It allows teams to focus on application development without the distraction of underlying infrastructure complexities.
- This IT tool ensures that containerized applications run reliably, effectively managing deployment and scaling while abstracting the hardware and network configurations.

Kubernetes (k8s)

- Container Orchestration System
- Open source
- Initiated by Google
- Started 2014
- Huge success
- Declarative API
- Eventual consistent

Kubernetes API

- Container
  - Containers are packages of software that contain all of the necessary elements to run in any environment. In this way, containers virtualize the operating system and run anywhere, from a private data center to the public cloud or even on a developer's personal laptop.
- Pod
  - Pods is the smallest deployable unit of computing that you can create and manage in Kubernetes.
  - Pod is one or more containers with shared network namespace, devices and resource quota.

Kubernetes API (cont.)

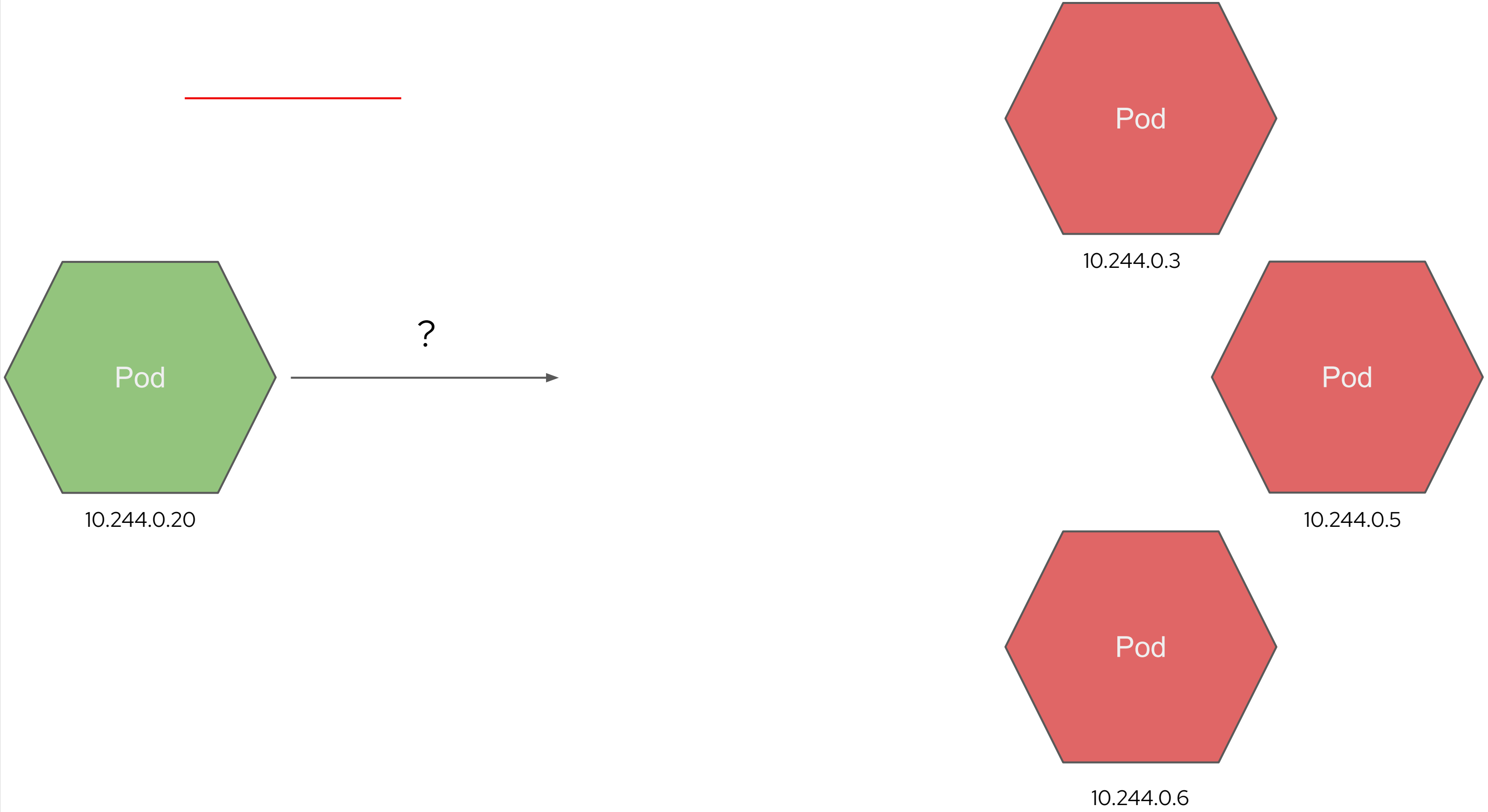
- Deployment (example next slide)
  - A Kubernetes Deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application.
  - Versioning
- Services
  - Expose Pod functionality to consumers inside the cluster and and remotely

Kubernetes Deployment Example

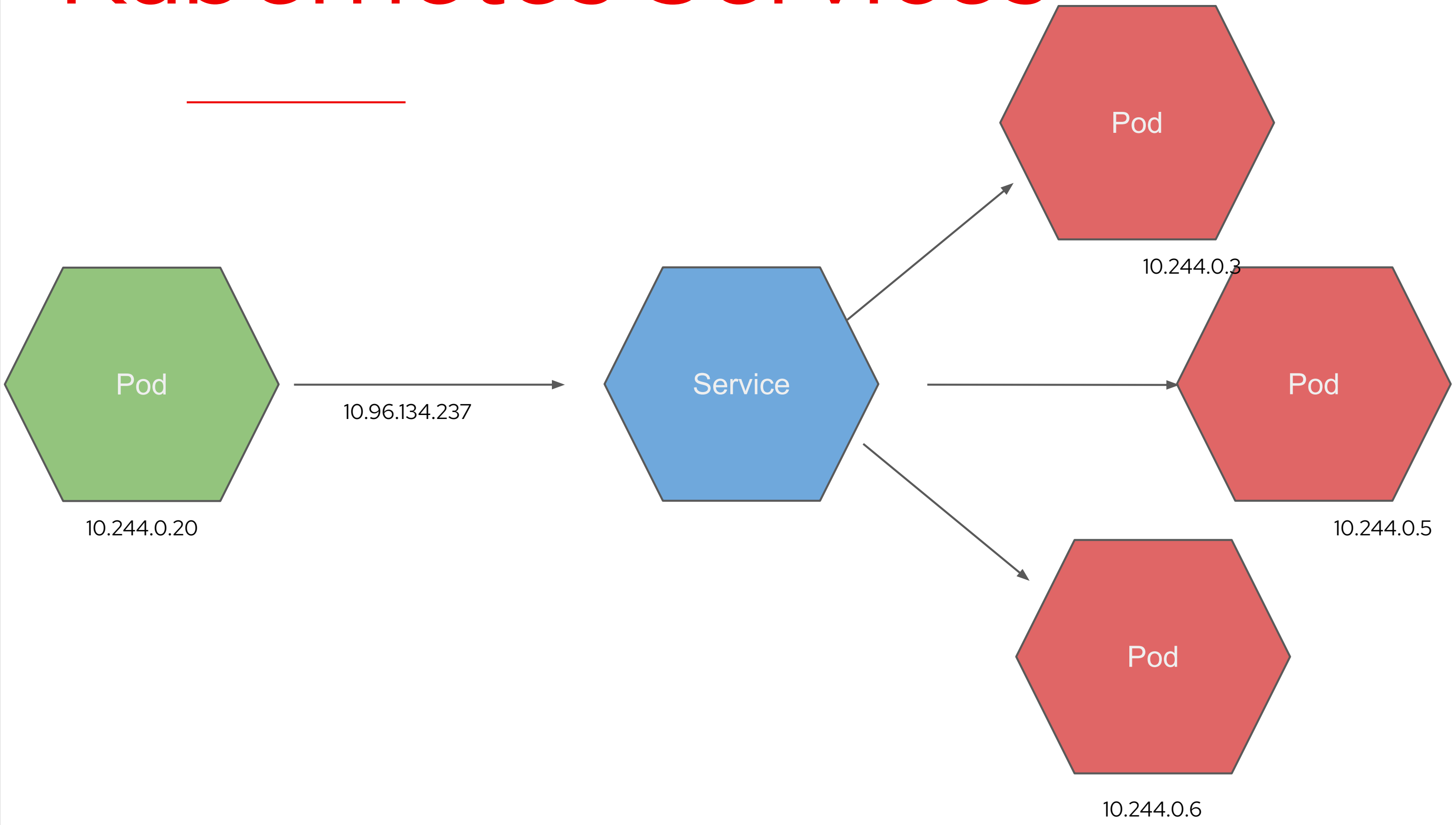
```yaml
kind: Deployment
metadata:
  name: dogood-deployment
  labels:
    app: dogood
spec:
  replicas: 3000
  selector:
    matchLabels:
      app: doogood
  template:
    metadata:
      labels:
        app: dogood
    spec:
      containers:
      - name: dogood
        image: quay.io/dogood:1.14.2
        ports:
        - containerPort: 80
```

Load Balancer Service

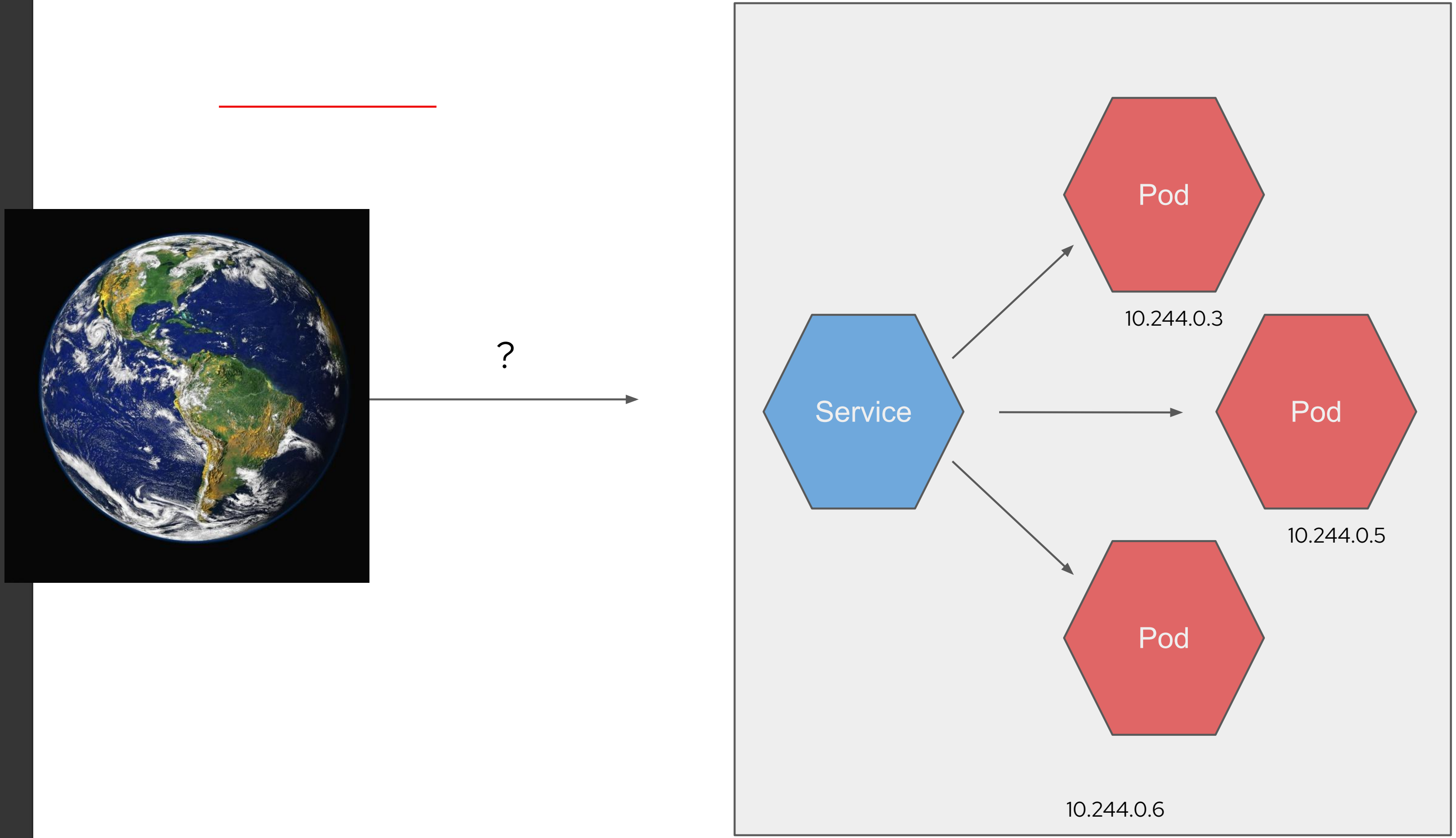
Kubernetes Services



Kubernetes Services



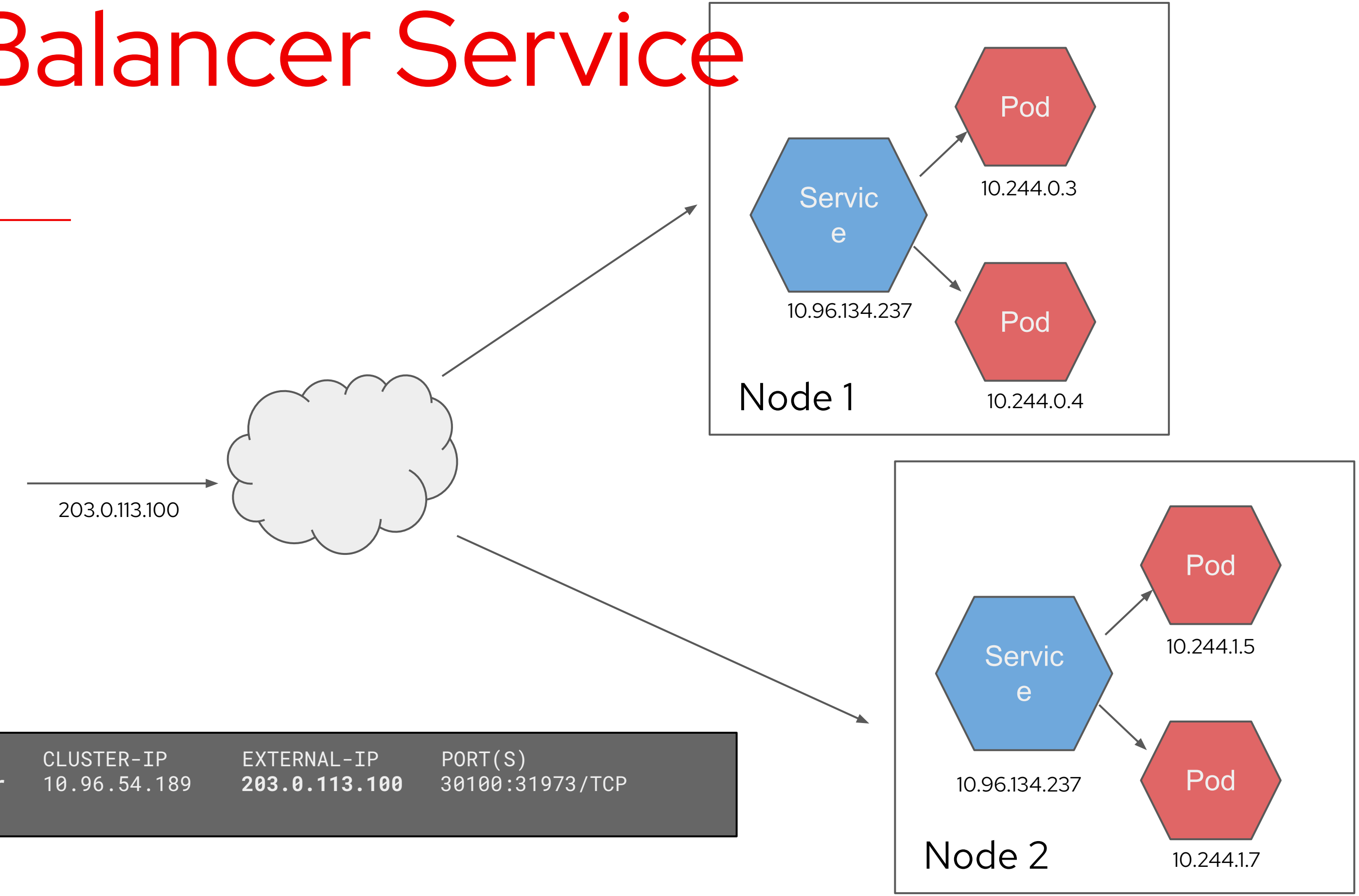
Kubernetes Services



Load Balancer Service

```
NAME         TYPE            CLUSTER-IP      EXTERNAL-IP   PORT(S)
lbservice    LoadBalancer    10.96.54.189    203.0.113.0   30100:31973/TCP
```

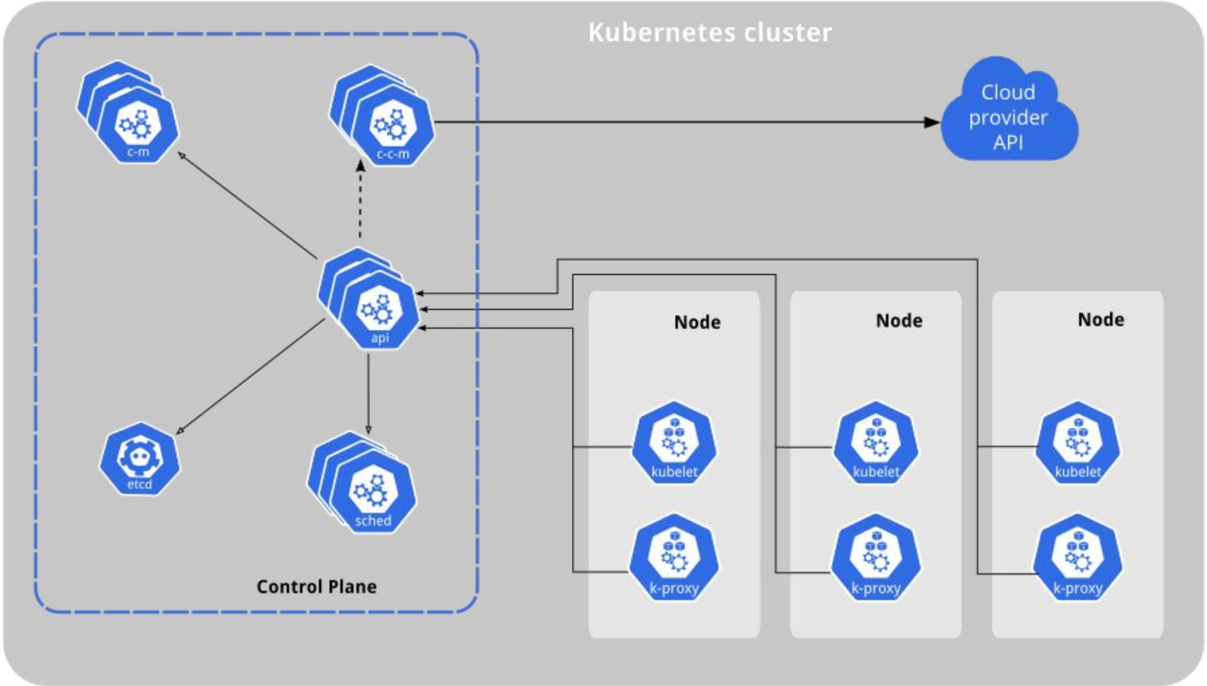
Load Balancer Service



```
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)
lbservice    LoadBalancer   10.96.54.189   203.0.113.100    30100:31973/TCP
```

Kubernetes Internals

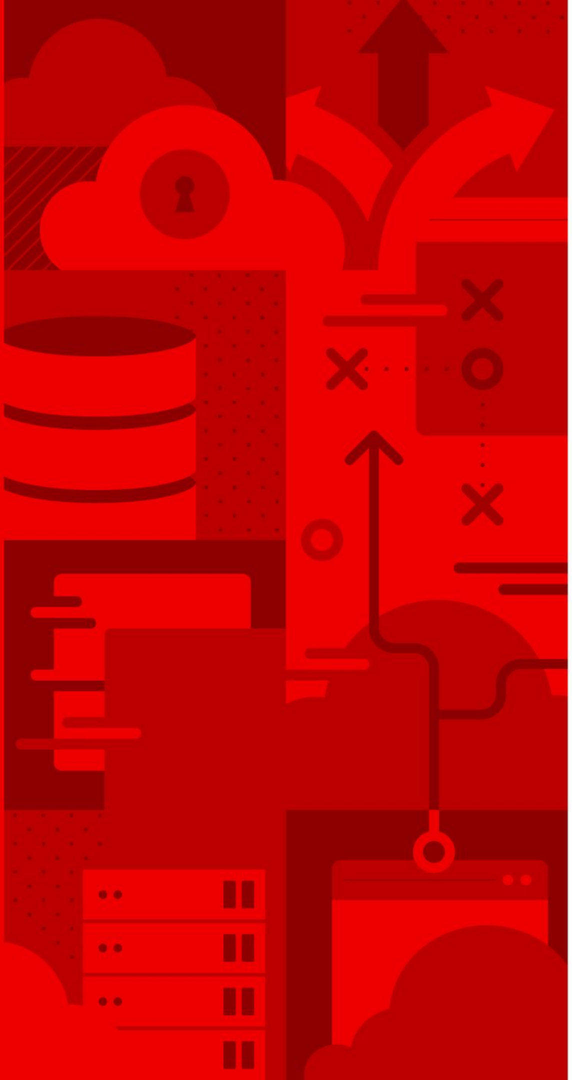
K8s Internals

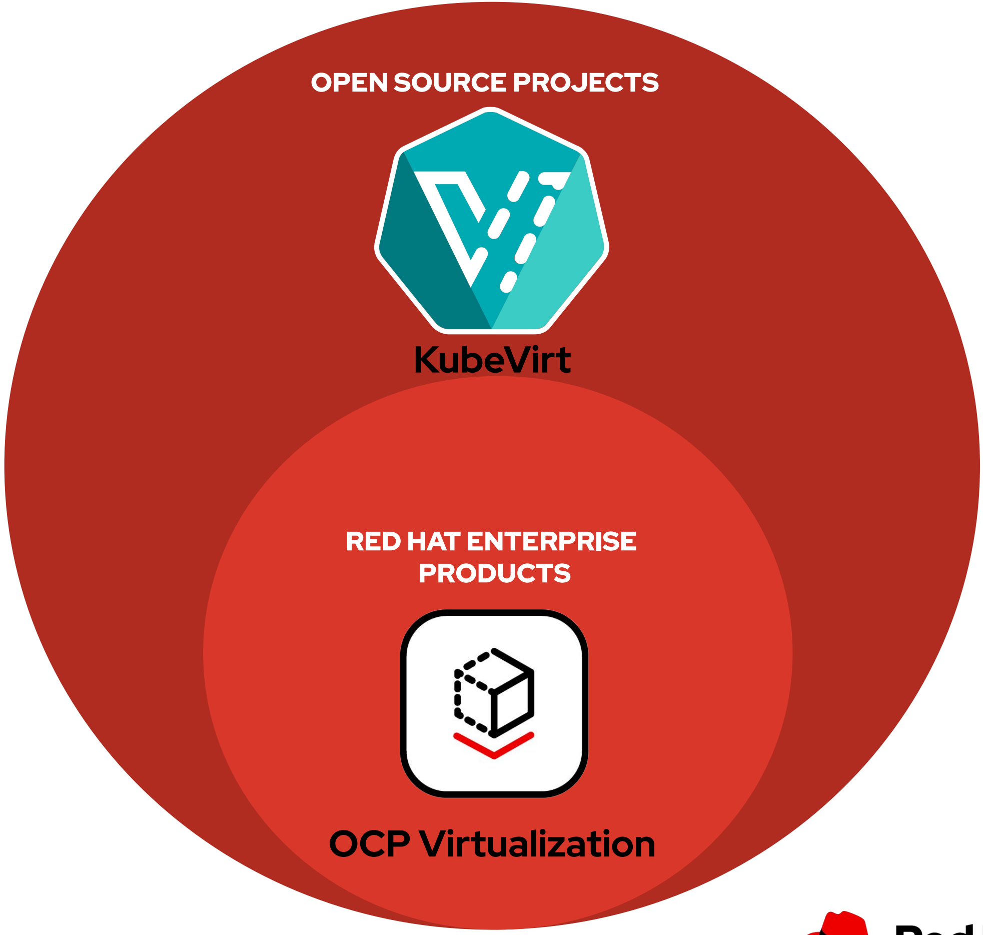


Copied from https://kubernetes.io/docs/concepts/overview/components/

Virtualization

What are Red Hat products?





OPEN SOURCE PROJECTS

KubeVirt

RED HAT ENTERPRISE PRODUCTS

OCP Virtualization

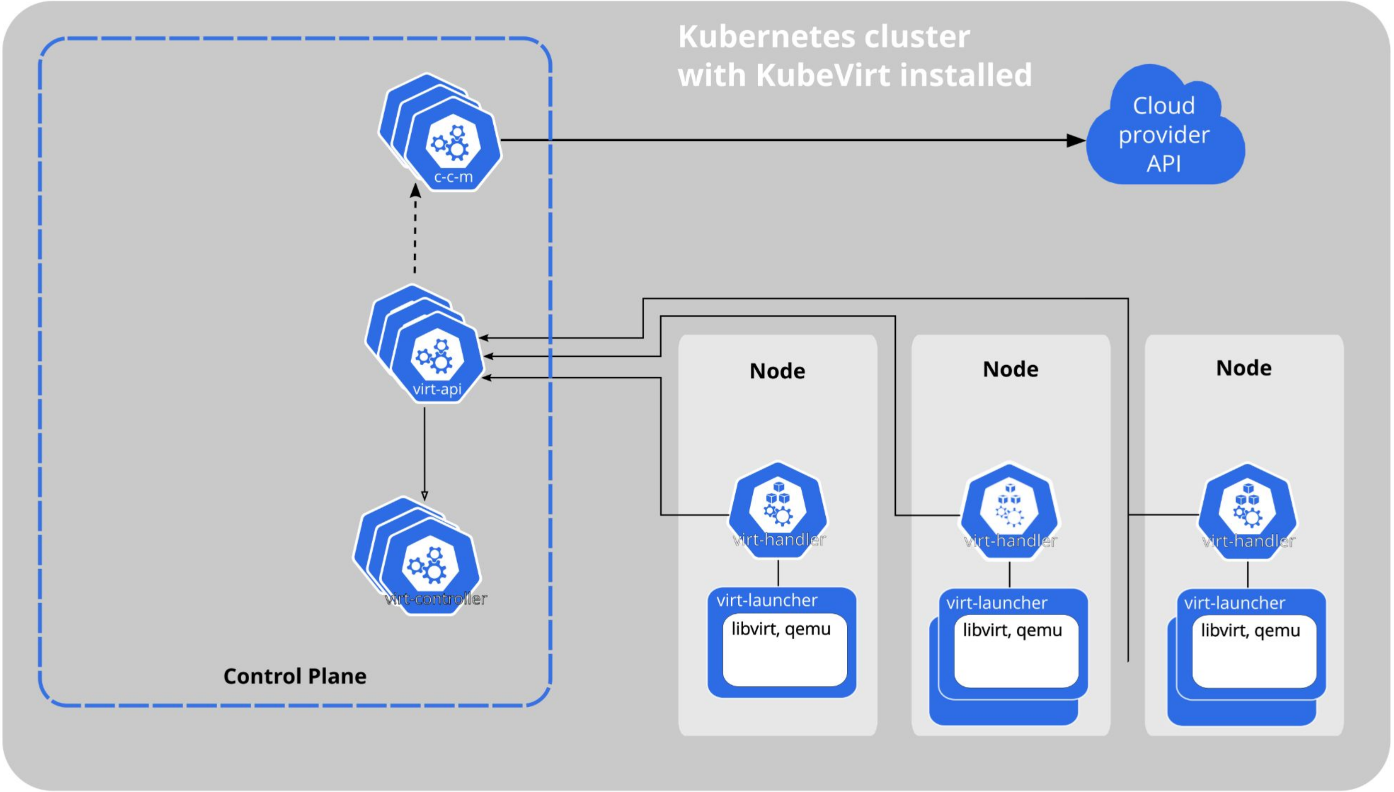
Still virtualization?

- K8s often uses virtualization to run its nodes. But why would anyone care about that? Who need to run virtual machines these days?
- ~20 years old technology in x86, running multiple operating systems on one host.
- Containers are cool, but not all workload fit into k8s deployments
  - E.g 10yo Windows monolith
- Users want to manage everything using k8s APIs
  - Wish to orchestrate a complex application in k8s
  - Find out that one piece is not ready for containers

```
kind: VirtualMachine
metadata:
  name: dogood-vm
  labels:
    app: dogood
spec:
  template:
    ...
    metadata:
      labels:
        app: dogood
    domain:
      devices:
        disks:
        - name: myboot
        interfaces:
        - masquerade: {}
    volumes:
      - name: myboot
        containerDisk:
          image: quay.io/windogood:1.14.2
...
```

Custom Resource

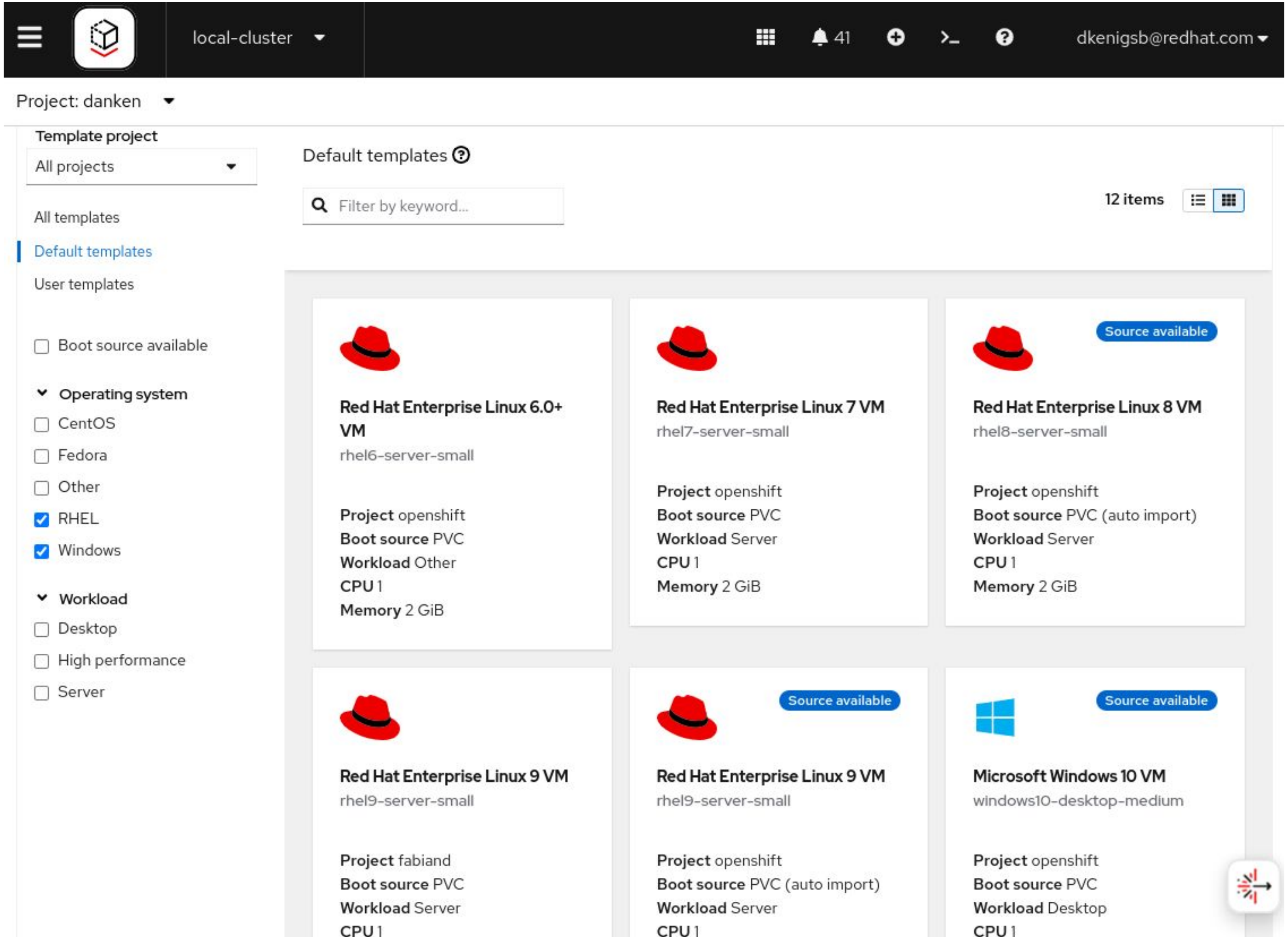
Simplified KubeVirt Architecture



Modified from https://kubernetes.io/docs/concepts/overview/components/

Presentations 101:

No live demo



Project: danken

Template project

All projects

All templates

Default templates

User templates

□ Boot source available

Operating system

□ CentOS

□ Fedora

□ Other

☑ RHEL

☑ Windows

Workload

□ Desktop

□ High performance

□ Server

Default templates

Filter by keyword...

12 items

Red Hat Enterprise Linux 6.0+ VM
rhel6-server-small

Project openshift
Boot source PVC
Workload Other
CPU 1
Memory 2 GiB

Red Hat Enterprise Linux 7 VM
rhel7-server-small

Project openshift
Boot source PVC
Workload Server
CPU 1
Memory 2 GiB

Source available

Red Hat Enterprise Linux 8 VM
rhel8-server-small

Project openshift
Boot source PVC (auto import)
Workload Server
CPU 1
Memory 2 GiB

Red Hat Enterprise Linux 9 VM
rhel9-server-small

Project fabiand
Boot source PVC
Workload Server
CPU 1

Source available

Red Hat Enterprise Linux 9 VM
rhel9-server-small

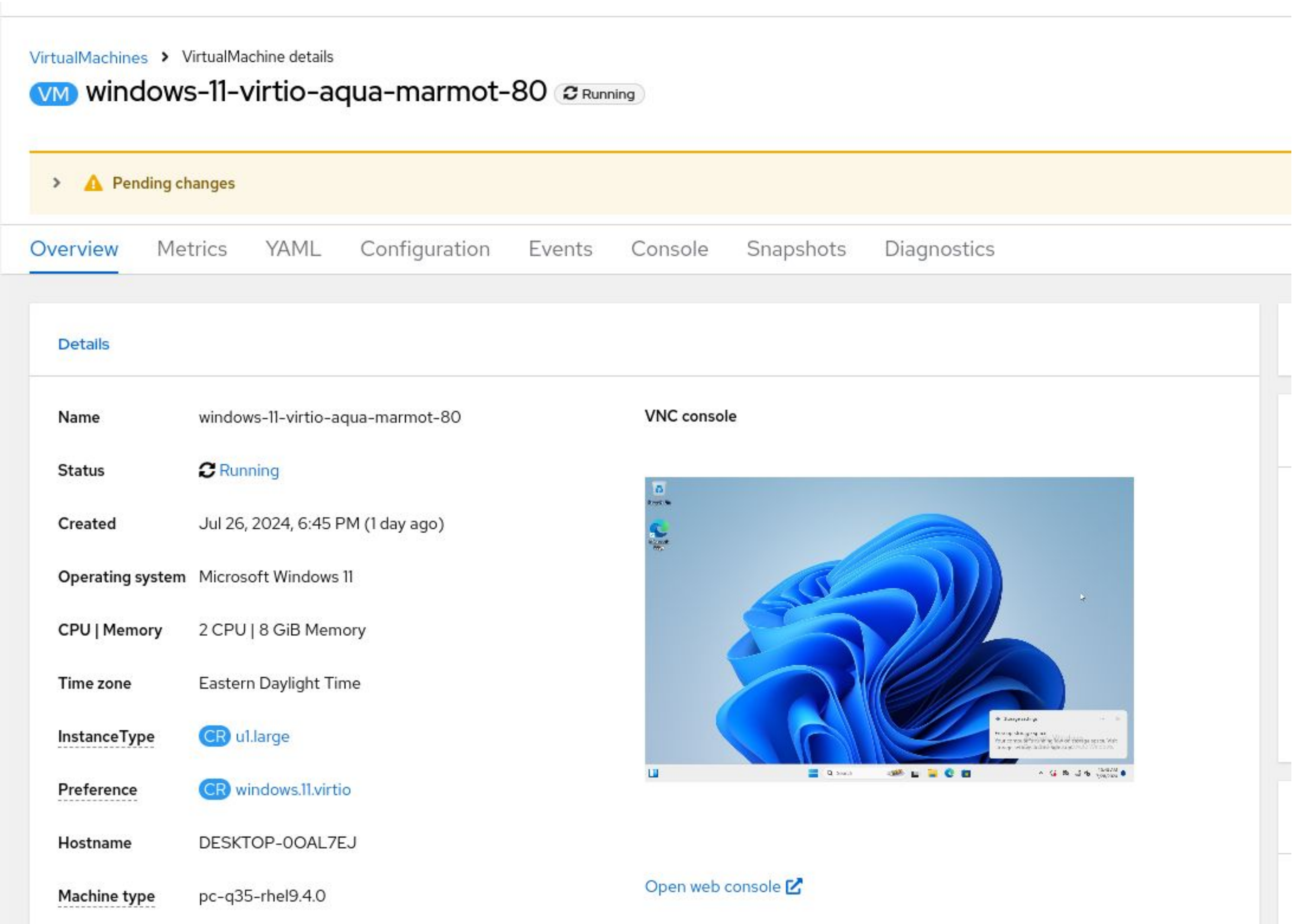
Project openshift
Boot source PVC (auto import)
Workload Server
CPU 1

Source available

Microsoft Windows 10 VM
windows10-desktop-medium

Project openshift
Boot source PVC
Workload Desktop
CPU 1

Non-Live Demo



VirtualMachines › VirtualMachine details

VM windows-11-virtio-aqua-marmot-80 Running

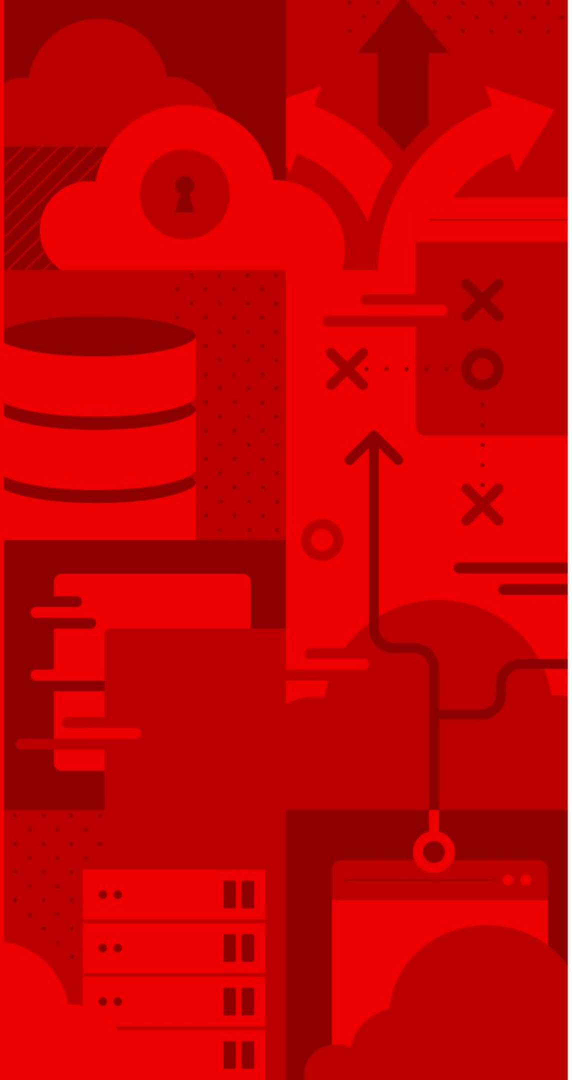
› ⚠ Pending changes

Details

Name	windows-11-virtio-aqua-marmot-80
Status	Running
Created	Jul 26, 2024, 6:45 PM (1 day ago)
Operating system	Microsoft Windows 11
CPU | Memory	2 CPU | 8 GiB Memory
Time zone	Eastern Daylight Time
InstanceType	CR u1.large
Preference	CR windows.11.virtio
Hostname	DESKTOP-0OAL7EJ
Machine type	pc-q35-rhel9.4.0

VNC console

Open web console



Thank you

Come over to https://github.com/kubevirt/kubevirt/pulls and contribute!

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat