the History of Containers

By : Oren Oichman
Title: Senior Solution Architect
Email : ooichman@redhat.com
IRC : two_oes/ooichman

Red Hat

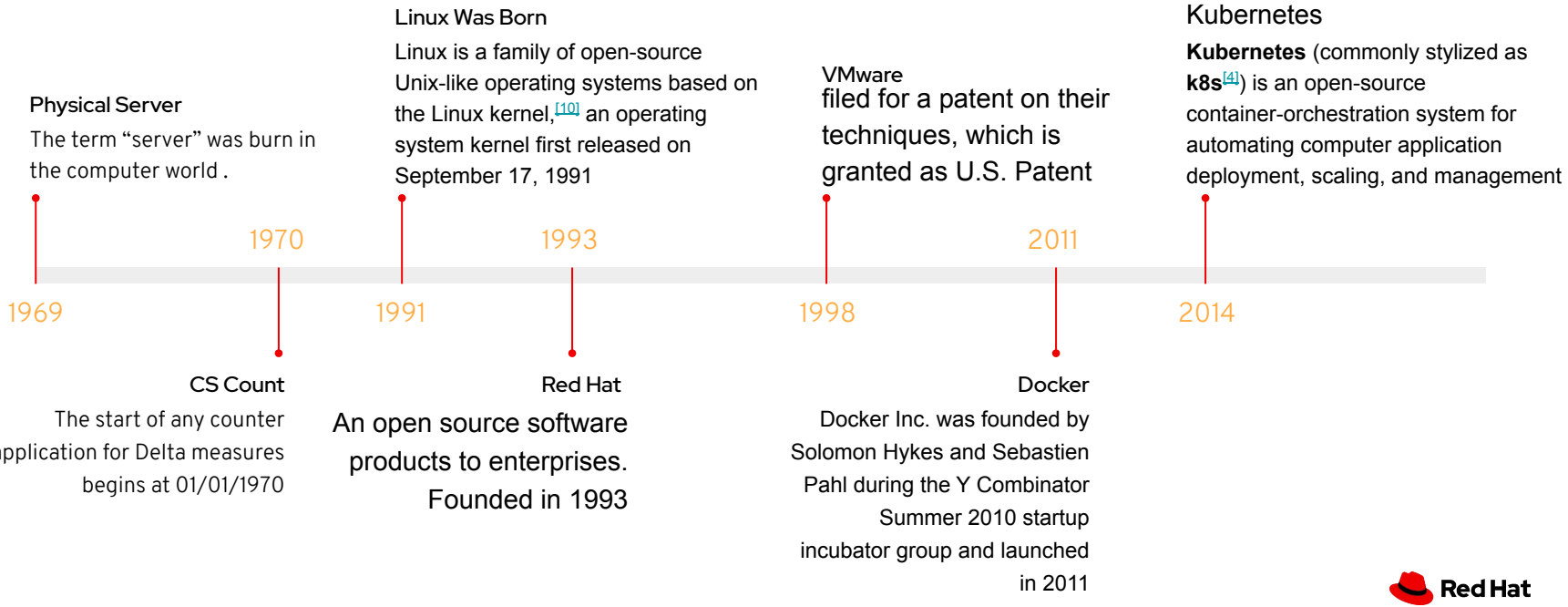
“Who controls the past controls the future. Who controls the present controls the past.”

George Orwell,
1984

Source:
Insert source data here
Insert source data here
Insert source data here

The Evolution of Platforms

From Physical to Container



Red Hat

From Physical to Containers

Optional subheading

Each Physical Server had features by the Manufacturer vendor and OS had to adopt to new hardware technologies

Virtualization possibilities :

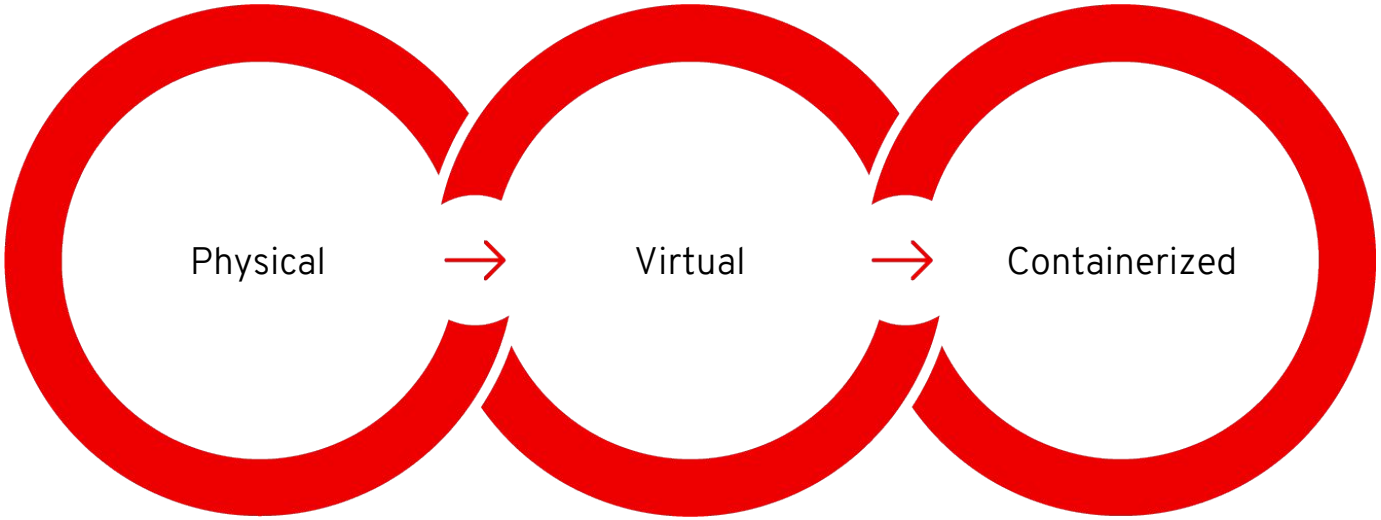
1. Running more than 1 OS on a Physical Server
2. Hardware Manufacture became Commodity

Containerize possibilities :

Completely different update process for the application and the OS

The Evolution of application

The core for application development shift



Red Hat

Development (P.O.V) on a physical Server

Long time to produce

Very long time from design to production

Taking turns

With only a few available developers had to “wait in line”

Sync with the OS

The Operating System had to have the latest drivers and firmware

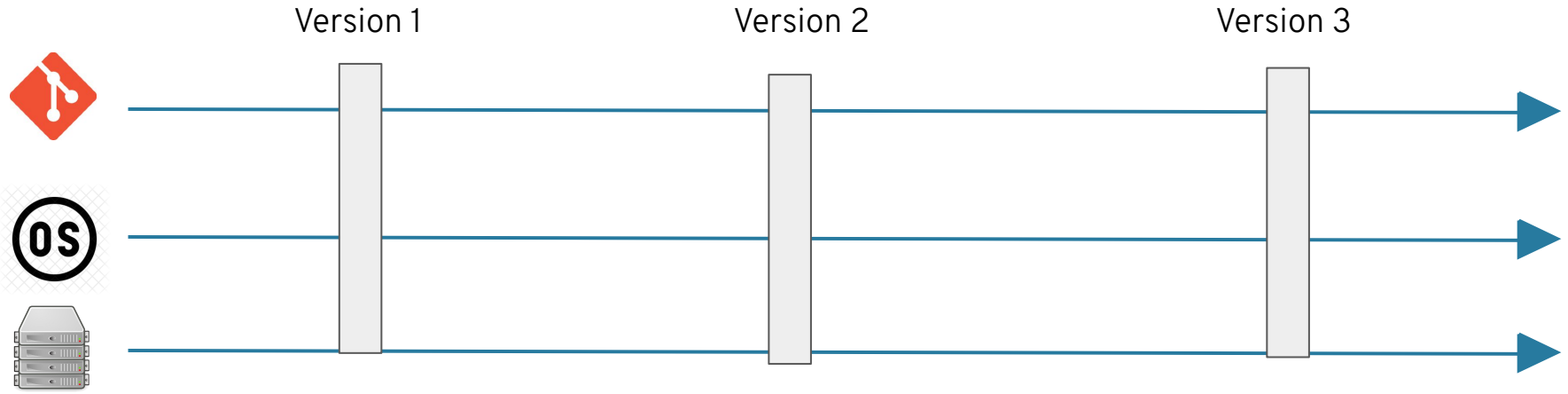
Conflicting libraries

Libraries that new application needed where not included due to Operating System conflict

Source:
Insert source data here
Insert source data here
Insert source data here

Development Timeline

Phase I	Phase II	Phase III	Phase IV	Phase V
Plan	Build	Release	deploy	Monitor
Planning on how the application should look like	Build the application with the selected Code	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning



Virtualization concept

In computing, **virtualization** (alternatively spelled *virtualisation*) refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources.

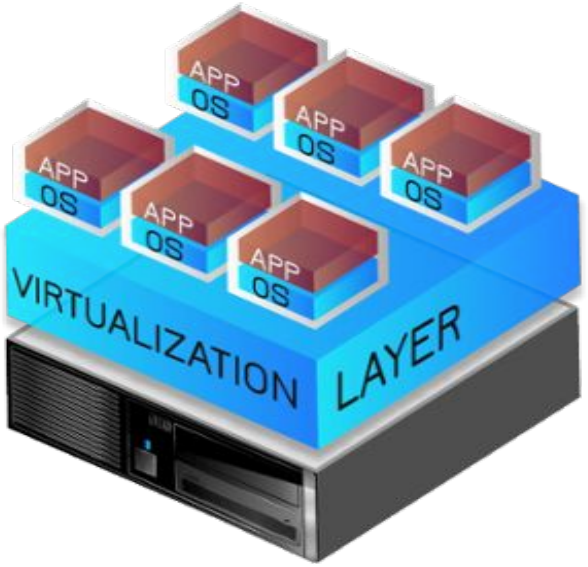
Virtualization began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different applications. Since then, the meaning of the term has broadened

Source:
Insert source data here
Insert source data here
Insert source data here

Developing on a Virtual Server (benefits)

1. More than 1 OS on a physical Server.
2. The Hardware drivers are commodity and so not relevant to the development process

Traditional Server Architecture

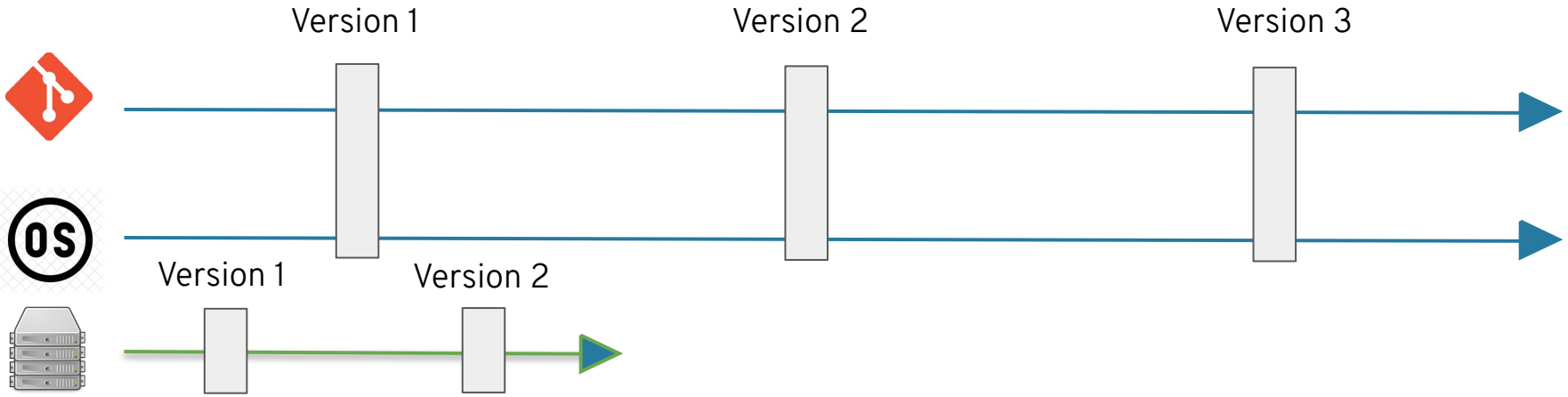


Virtualized Server Architecture

Source:
Insert source data here
Insert source data here
Insert source data here

Development Timeline

Phase I	Phase II	Phase III	Phase IV	Phase V
Plan	Build	Release	deploy	Monitor
Planning on how the application should look like	Build the application with the selected Code	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning



Container concept

A **container** is a class, a data structure or an abstract data type (ADT) whose instances are collections of other objects.

In other words, they store objects in an organized way that follows specific access rules.

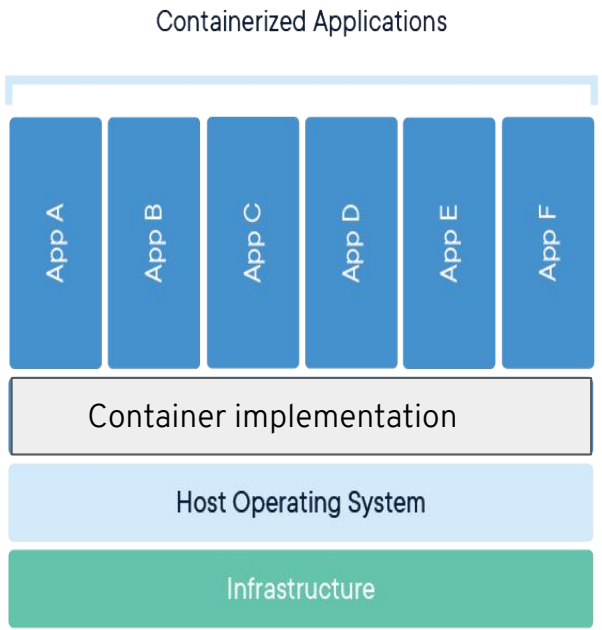
The size of the container depends on the number of objects (elements) it contains.

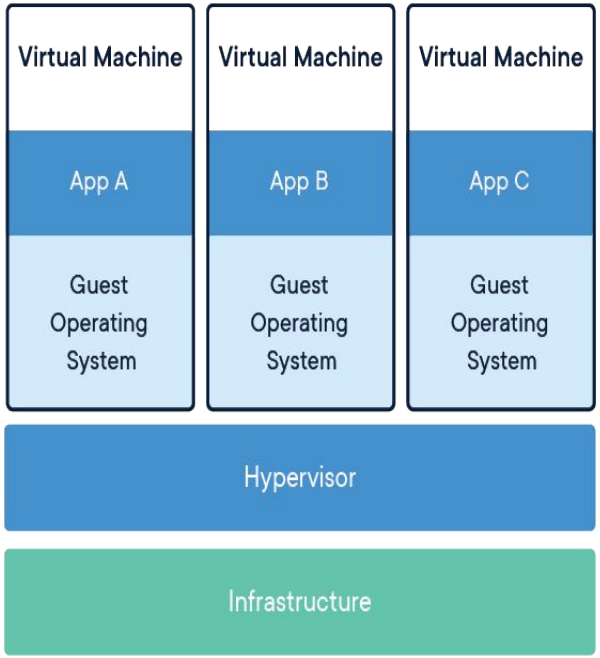
Underlying (inherited) implementations of various container types may vary in size and complexity, and provide flexibility in choosing the right implementation for any given scenario.

Source:
Insert source data here
Insert source data here
Insert source data here

Developing on a Container (benefits)

1. Each app as their own sandbox and set of libraries and binaries
2. Minimal dependency in the Operating system
3. Dedicated life cycle



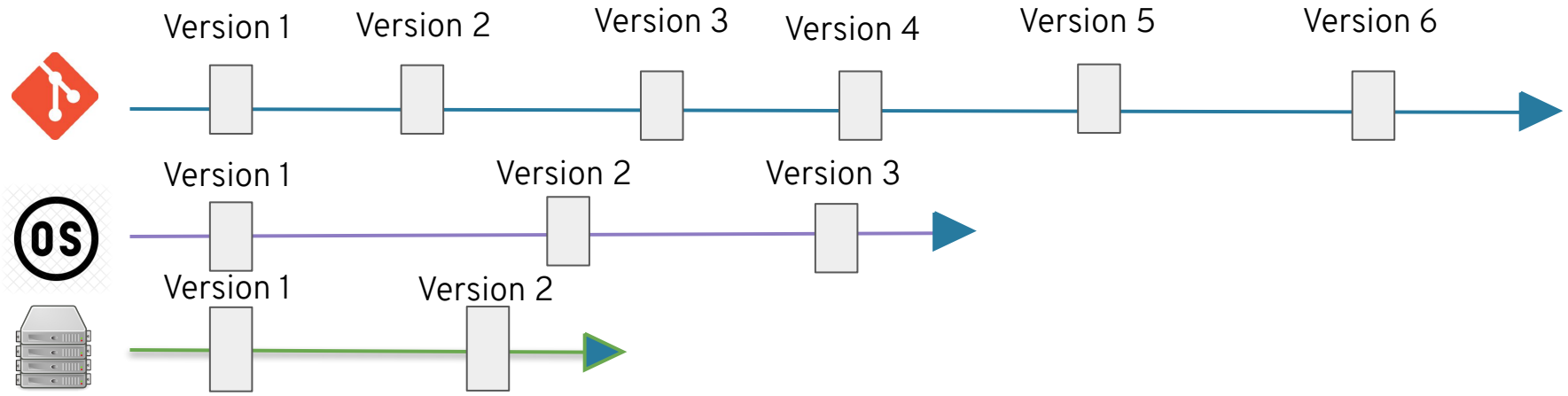


Source:
Insert source data here
Insert source data here
Insert source data here

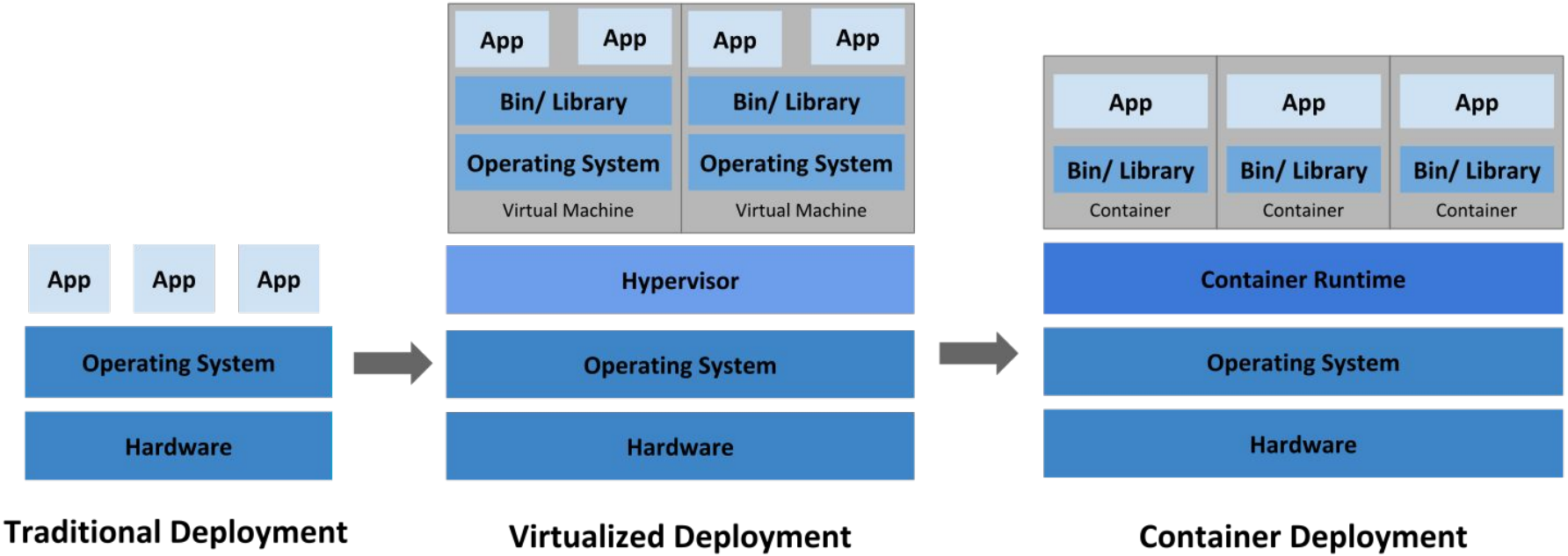
Red Hat

Development Timeline

Phase I	Phase II	Phase III	Phase IV	Phase V
Plan	Build	Release	deploy	Monitor
Planning on how the application should look like	Build the application with the selected Code	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning



The Evolution of Containers



Begin Exercise 1

Source:
Insert source data here
Insert source data here
Insert source data here

Red Hat