Hitachi & Red Hat collaborate: Container migration guide

In open source, we feel strongly that to do something well, you have to get a lot of people involved - Linus Torvalds

Yamada Tatsuya & Scott McCarty
Hitachi & Red Hat
May 10th, 2018

Why Hitachi & Red Hat Collaborate

Hitachi had a unique perspective on how to operationalize checklists as well as upstream Kubernetes contributions, and Red Hat had a lot of experience migrating applications in engineering and consulting.

- Collaborated on philosophy of how to tackle the problem of migrations
- Developed set of runbook like checklists around architecture, security & performance
- Published free e-Book: https://red.ht/2EkVdkJ

Basic Philosophy

redhat.

Purpose & Mission

To create a piece of content that would give teams easy, but crucial technical guidance

- Make the guide operational - teams can use it day to day
- Help teams leverage their existing technical knowledge
- Add additional knowledge around how to architect applications in containers
- Highlight characteristics of containerized applications
    - Architecture
    - Performance
    - Security

redhat.

Technical Guidance

redhat.

Three Pillars

Breaking the problem down

Figure 1. Application requirements

redhat.

Architecture

Code, Configuration, Data and more….

TABLE 1. TYPICAL WORKLOADS SEEN IN THE DATACENTER

	EASY	MODERATE	DIFFICULT
Code	Completely isolated (single process)	Somewhat isolated (multiple processes)	Self-modifying (e.g. actor model)
Configuration	One file	Several files	Anywhere in file system
Data	Saved in single space	Saved in several places	Anywhere in file system
Secrets	Static files	Network	Dynamic generation of certifcations
Network	HTTP, HTTPs	TCP, UDP	IPSEC, highly isolated
Installation	Packages, source	Installer and understood configuration	Installers (install.sh)
Licensing	Open source	Proprietary	Restrictive and proprietary

redhat.

Performance

Virtualization & Containers are additive technologies to bare metal

TABLE 2. WORKLOAD PLATFORM COMPARISON

	BARE METAL	+CONTAINERS	+VIRTUALIZATION
CPU intensive	Fast	**Fast**	**Fast**
Memory intensive	**Fast**	**Fast**	**Fast**
Disk I/O latency	**Fast**	**Fast**	**Medium**
Disk I/O throughput	Fast	Fast	Fast
Network latency	Fast	Fast	Medium
Network throughput	Fast	Fast	Fast
Deployment speed	Slow	Fast	Medium
Uptime (live migration)	No	No	Yes
Alternative OS	**Yes**	**Some**	**Yes**

redhat.

Security

Thinking about levels of isolation….

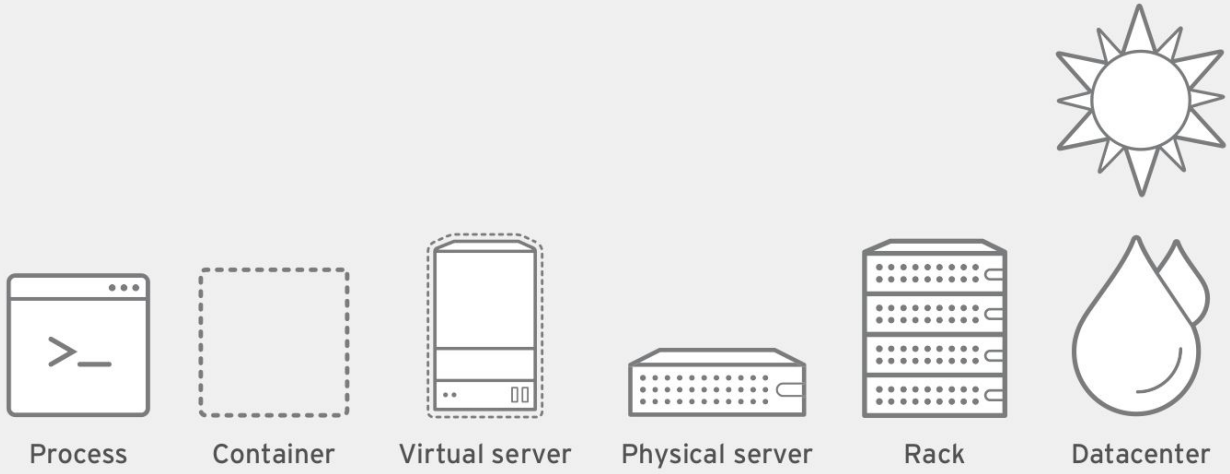


Figure 2. The tenancy scale

redhat.

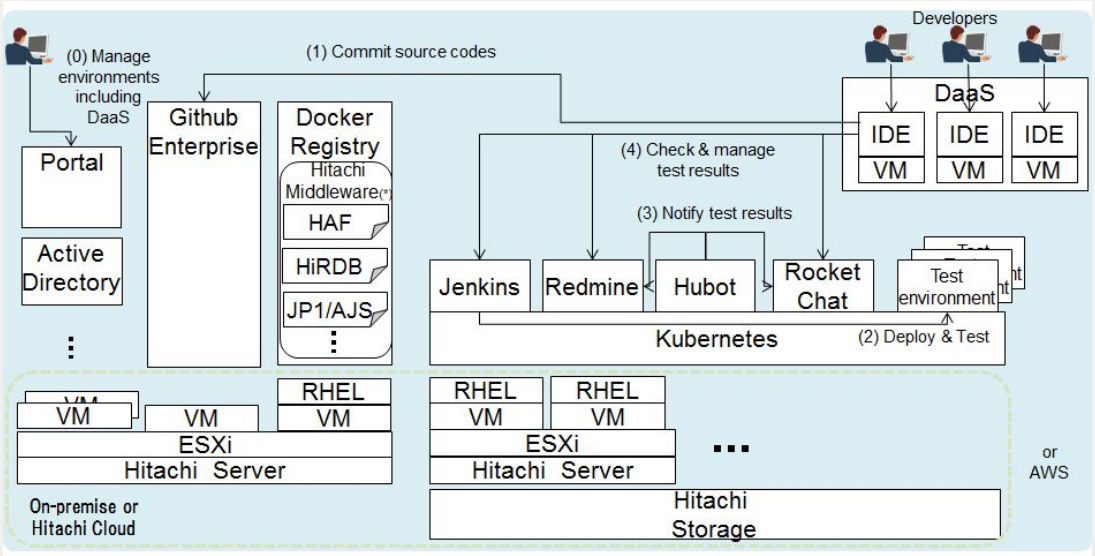
Containerizing Applications: Discovering Challenges

redhat.

Hitachi Solutions

Hitachi currently provides customers with the following kinds of DevOps services:

1. Hitachi's own DevOps Stack with Kubernetes and Docker
2. OpenShift
   a. OpenShift on Hitachi Server
   b. OpenShift on Azure + Justware



redhat.

Problems We Will Explore

We will highlight two interesting types of problems that surfaced while building container solutions at Hitachi

Issue	Detail	Category
Problem A: Middleware	A-1: Some kernel parameters cannot be configured on each container, independently.	Performance
	A-2: Some system calls cannot be executed from within a container.	Security
Problem B: Storage	B-1: Raw device cannot be mapped to container by function of kubernetes volume.	Performance

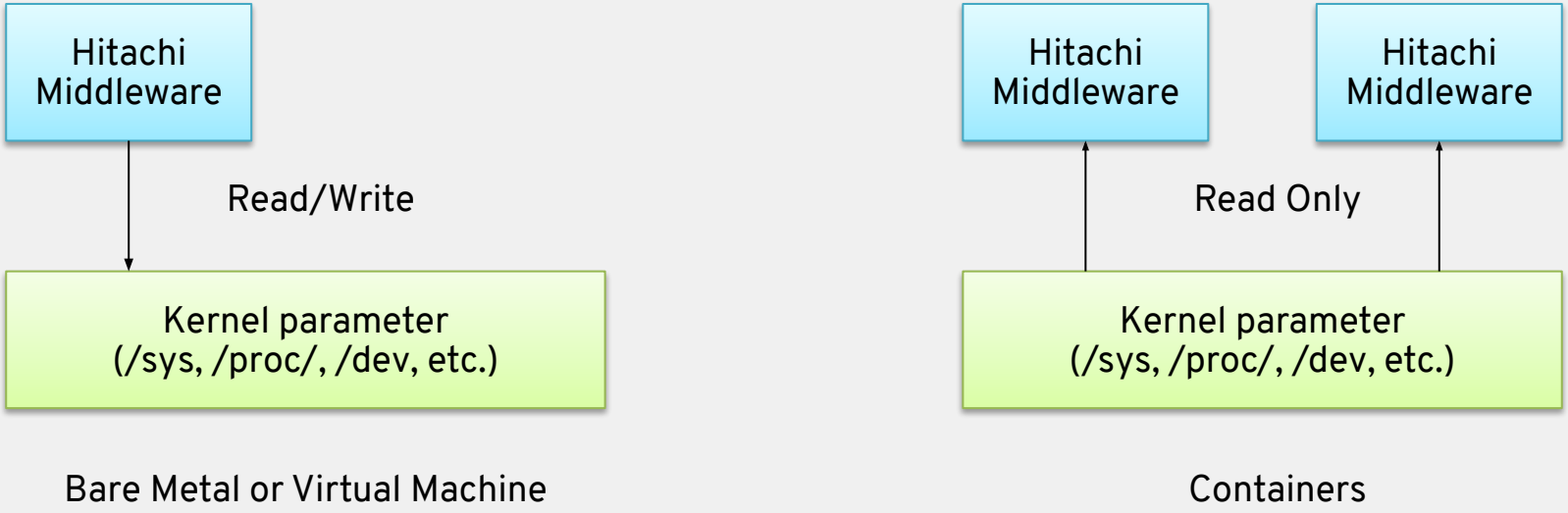
redhat.

PROBLEM A-1

redhat.

Problem A-1

Some kernel parameters cannot be configured on each container, independently



redhat.

Problem A-1

Raw device cannot be mapped to container by function of kubernetes volume

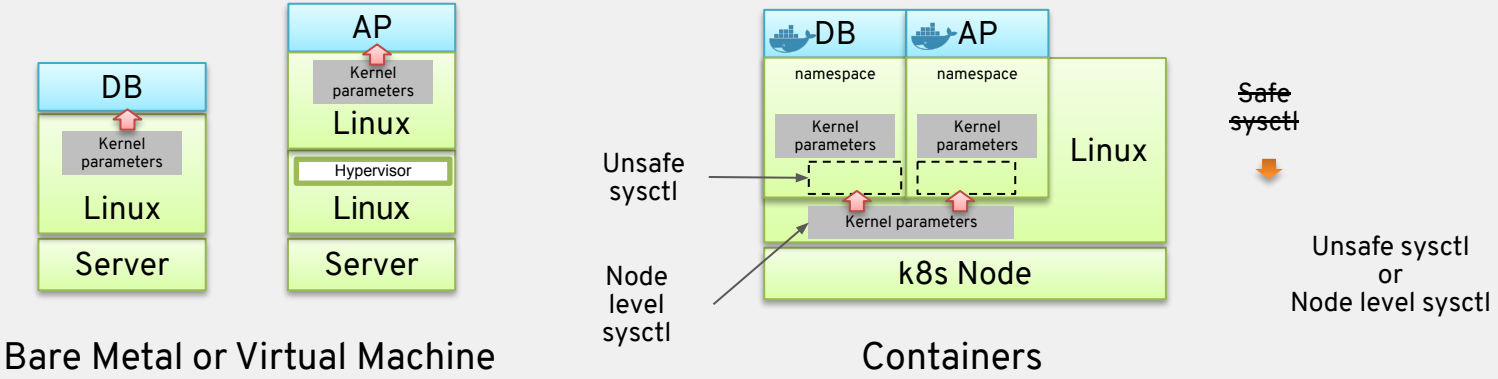
There are 3 kinds of kernel parameters in the container.

Parameters	Detail	Range
Node level sysctl	It is set for each node and can not be set for each container.	Node
unsafe.sysctl	Although it is set for each container, it may affect other containers.	Pod
safe.sysctl	It is set for each container, and it does not affect other container.	Pod

redhat.

Problem A-1

Raw device cannot be mapped to container by function of kubernetes volume

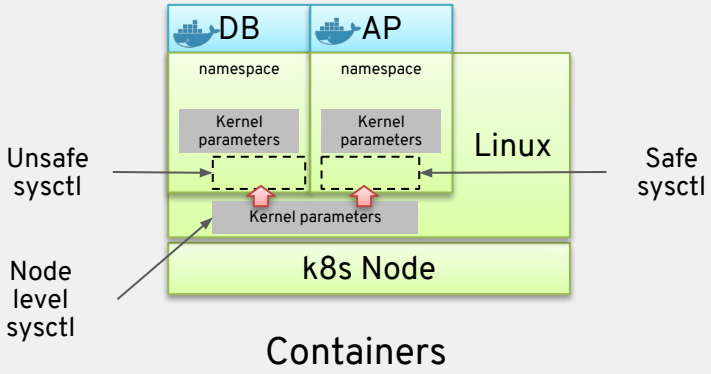


- We want to set kernel parameters to each container in order to use middleware, for example DB.
- Setting Node level sysctls or unsafe.sysctls will affect another container.

redhat.

Problem A-1

Raw device cannot be mapped to container by function of kubernetes volume



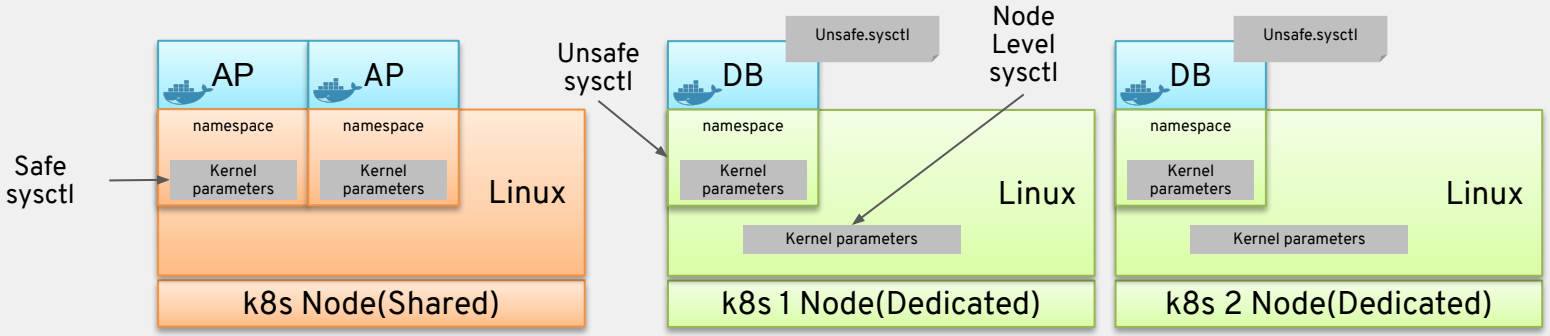
Goal

- Kernel parameters to be set without affecting other containers
- Kernel parameters classified as Node level / Unsafe should be able to also be set without affecting other containers

redhat.

Problem A-1

Solution: Pod scheduling on nodes with desired kernel parameters



- Pods with only safe.sysctl place as usual.
- Pods with unsafe.sysctl or Node level sysctl place as followings steps.
  - Before placing a Pod on dedicated node, we add a Kubernetes Taint to the node so that other Pod is not placed on the node.
  - We set sysctl settings to the node.
  - We create the Pod with Kubernetes Tolerate so that it is placed on the Taint Node.

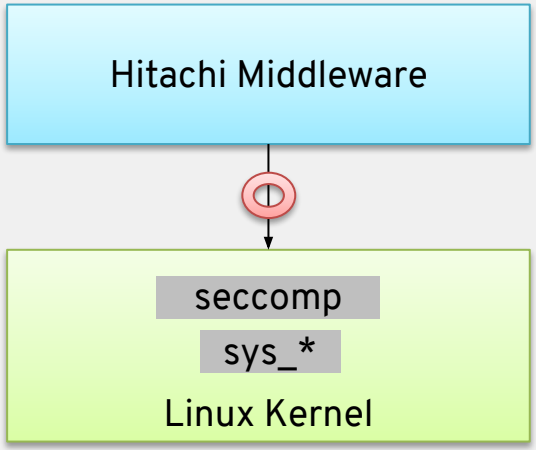
redhat.

PROBLEM A-2

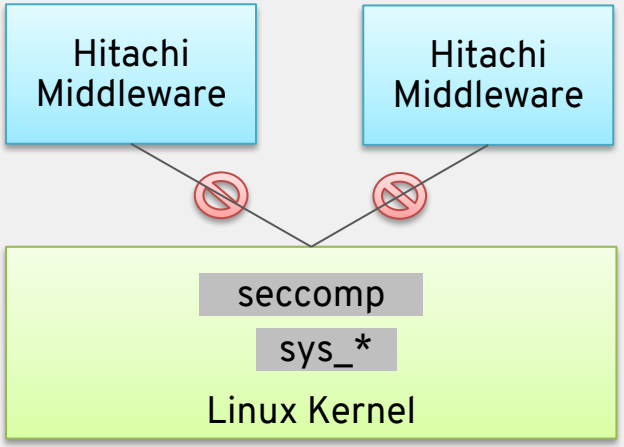
redhat.

Problem A-2

Some system calls cannot be executed from within a container



Bare Metal or Virtual Machine



Containers

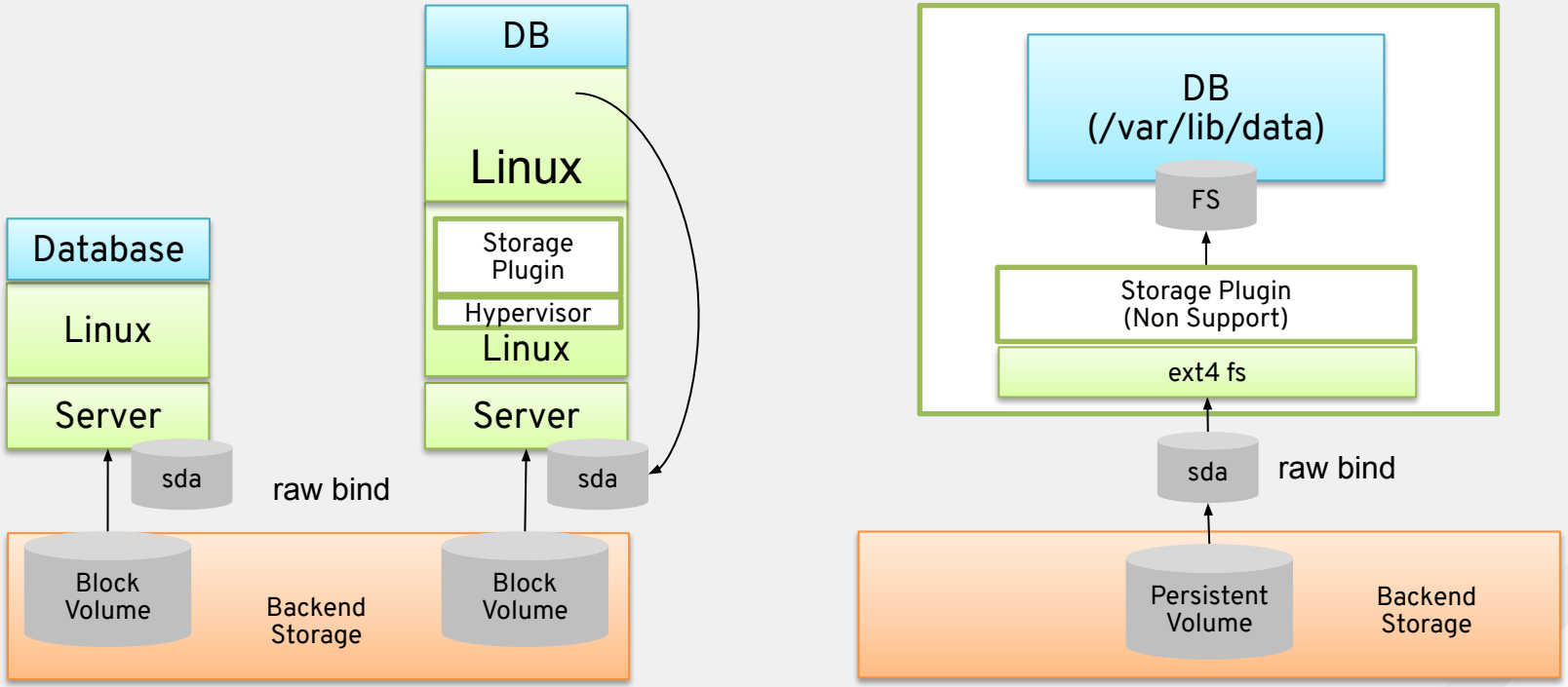
redhat.

PROBLEM B-1

redhat.

Problem B-1

Raw device cannot be mapped to container by function of kubernetes volume



Bare Metal or Virtual Machine

Containers

Learn How to Migrate

redhat.

Check Out the Guide

Download the e-Book:

https://red.ht/2EkVdkJ

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHat


youtube.com/user/RedHatVideos