

# Dockerと やわらかい仮想化

SECKUN 2020 version

近藤宇智朗 / GMO Pepabo, Inc.

# 近藤宇智朗 / Uchio Kondo

---



- ・GMOペパボ **技術基盤**チーム
- ・関心範囲: Ruby言語、Linux、Containers、システムプログラミング
- ・RubyKaigi 2016, 2018, 2019 登壇
- ・eBPF記事 @ Software Design(11・12月号)
- ・「mrubyシステムプログラミング」  
11月25日 刊行



# 講義前半の予定

---

- ・動画の内容:
  - ・ Dockerとコンテナ仮想化に関する概要
  - ・ オンライン演習（事前予習OK）
  - ・ Docker/Docker Compose を実際に触れる演習
- ・ 後半について
  - ・ コンテナの要素技術とセキュリティ（予定）



コンテナ座学

# そもそも、コンテナとは

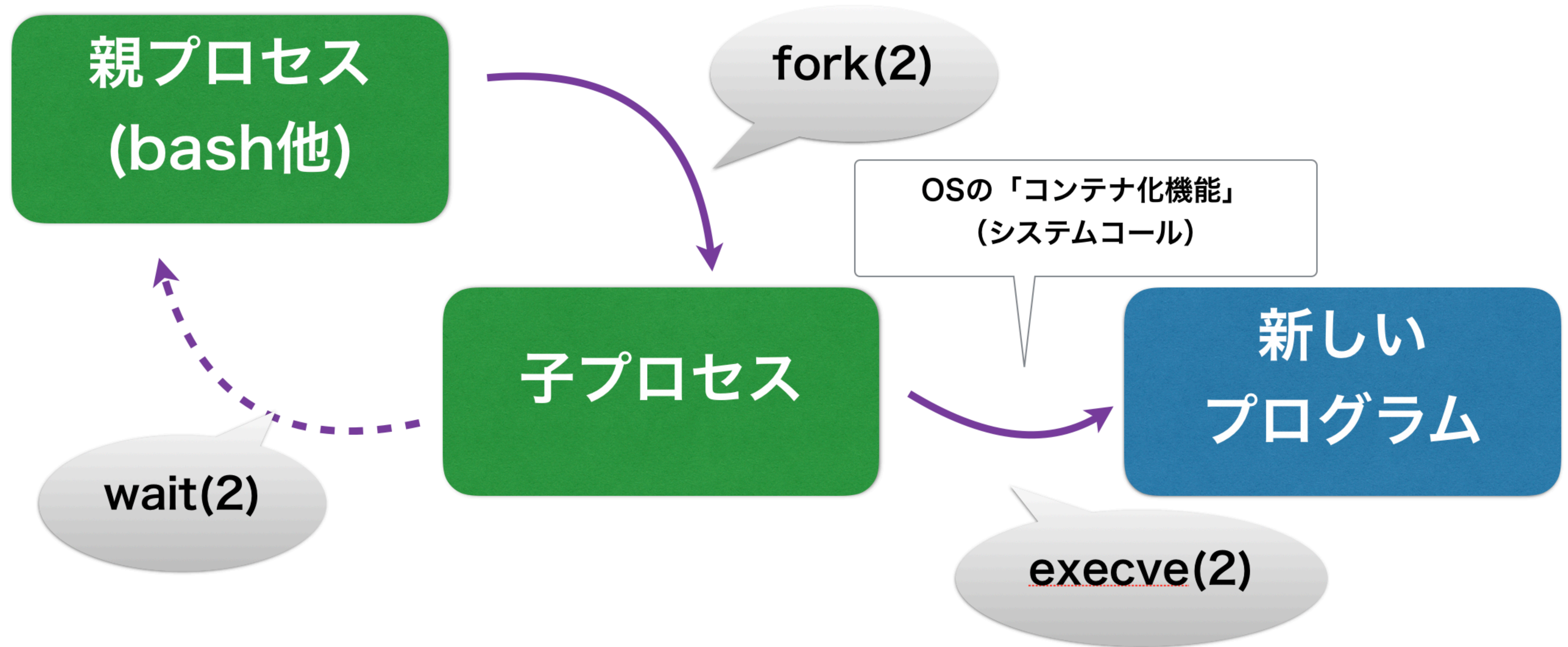
---

- ・Linuxの場合、プロセスに対して隔離、リソース制限、権限の制限などを行い、独立した環境をなるべく安全に提供したもの（とする）
- ・どのような経緯でこういった「仮想化」が出現したのか？





# fork/execve の間に



# beforeコンテナの「計算機の共有」

---

- ・人類はコンピュータを効率よく共有したかった
- ・UTS、マルチテナンシー
- ・UNIXユーザーによる計算機の共有
- ・Apache HTTP ServerなどによるVirtualHost機能
- ・chrootをはじめとした可視範囲の制限
- ・仮想マシンの作成 (kvm、QEMU)





# Apacheのマルチテナンシー

- ・ 松本 「Webサーバの高集積マルチテナントアーキテクチャに関する研究」 より
- ・ VirtualHost
- ・ chroot
- ・ スレッド単位の権限管理
- ・ suEXEC ...

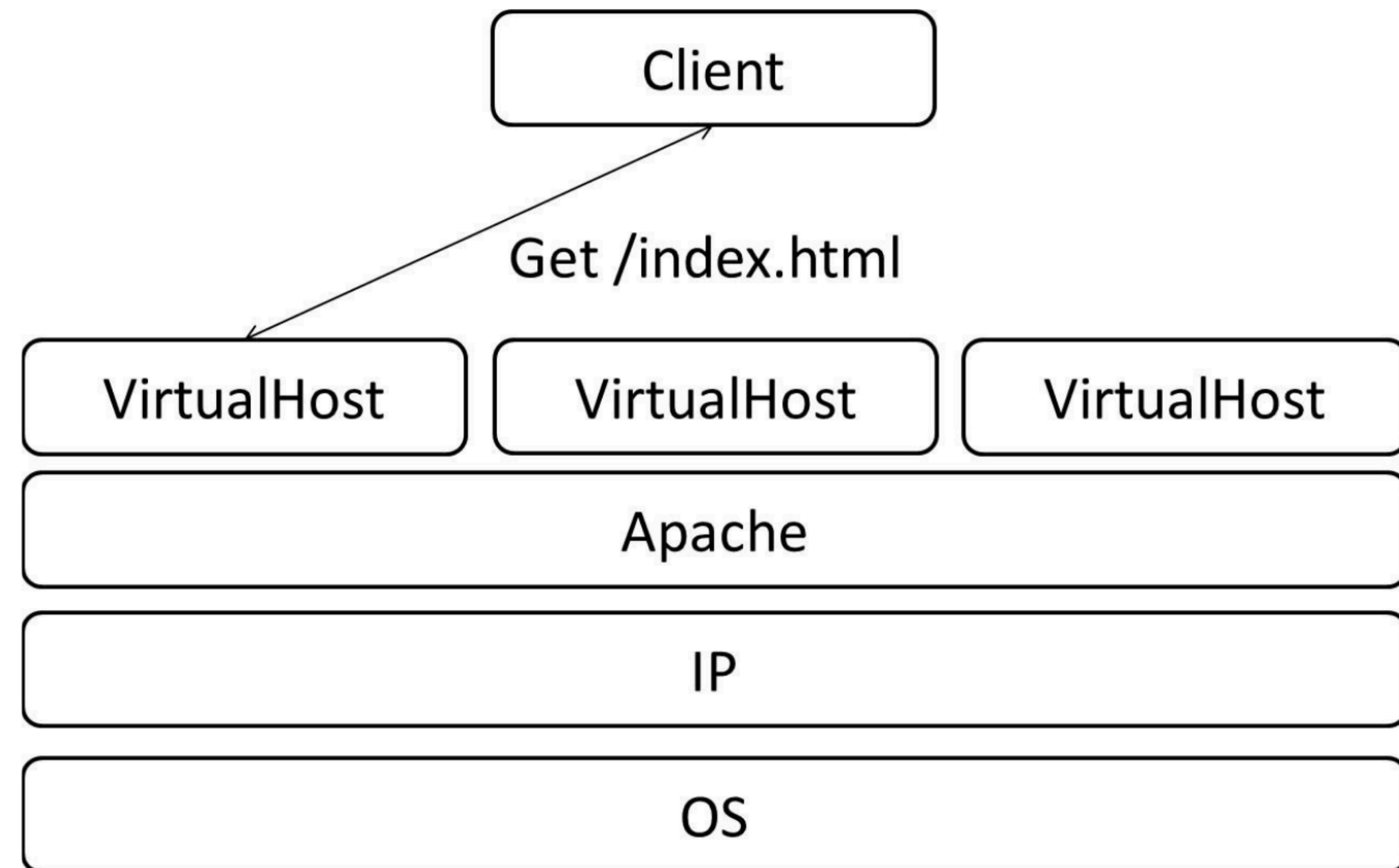


図 2.3 単一プロセスで複数ホストを扱う構成



# 簡単な仮想化の系図

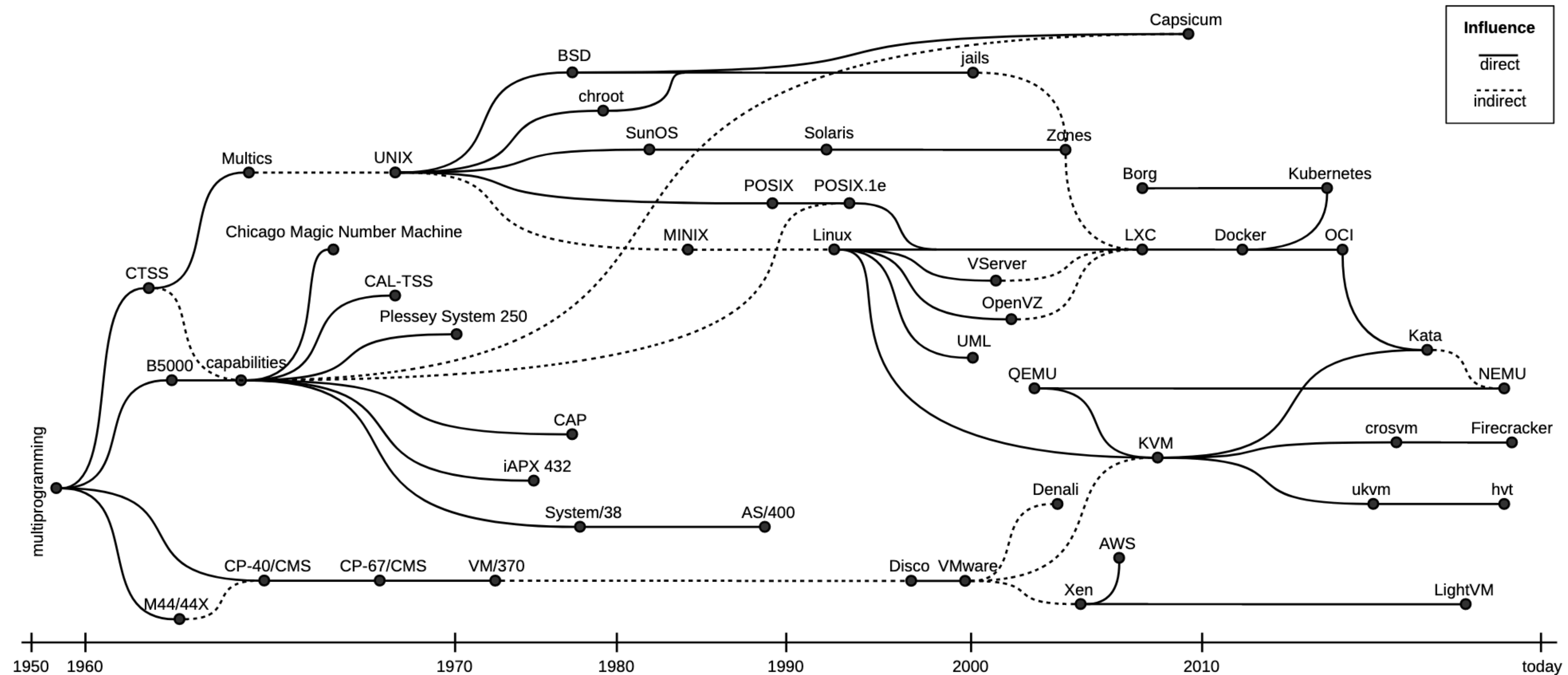


Figure 1: The evolution of virtual machines and containers.

- 「The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers」より

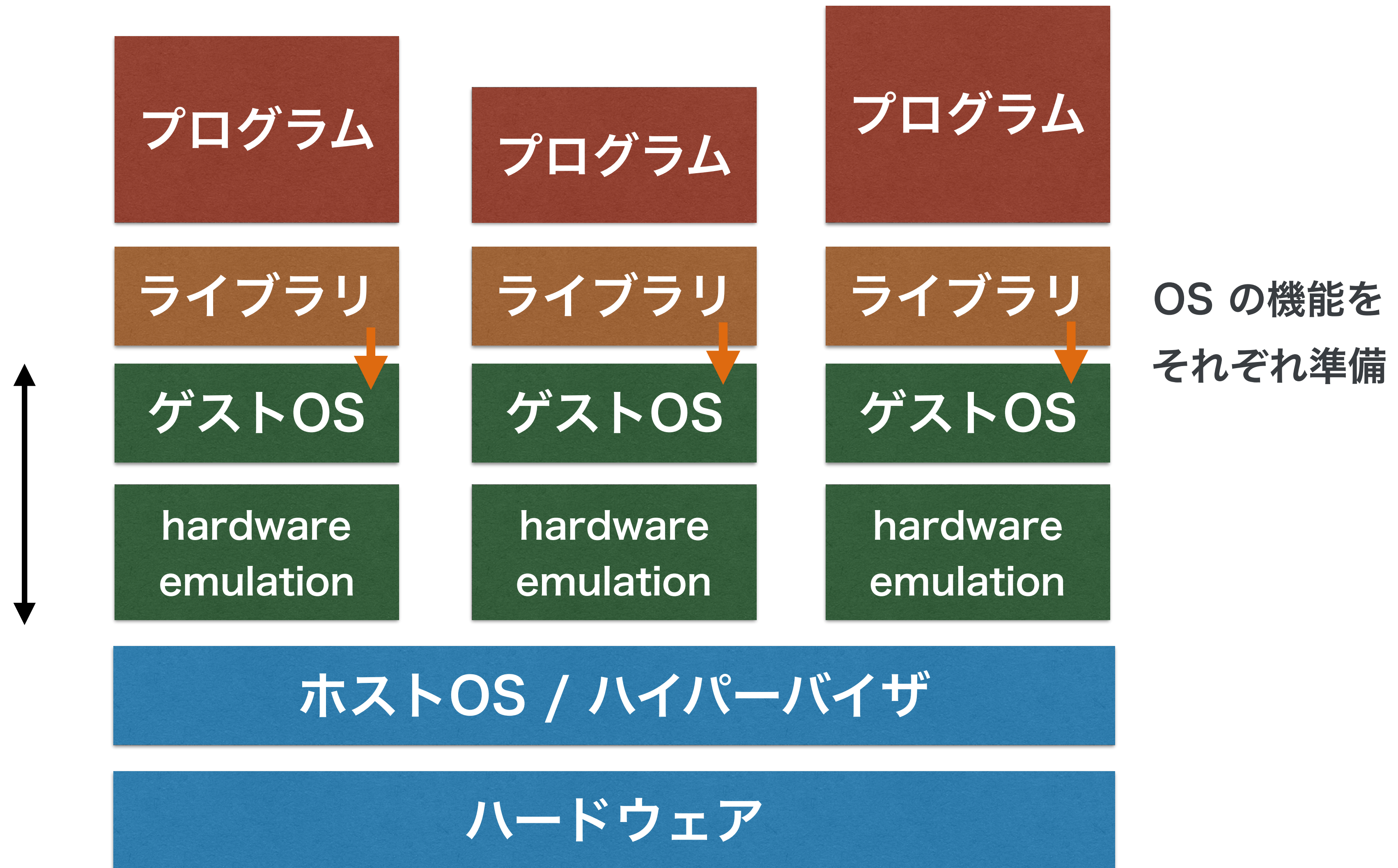
# コンテナの登場

---

- ・KVMなどのように、ハードウェアを含んだ仮想化は安全で自由度が高く、普及した。
- ・一方、オーバーヘッドの低減や集積率の向上、起動時間などの高速化、環境のポータビリティ向上などの需要が出てくる
- ・少しずつ進化を続けていたコンテナに注目が集まる

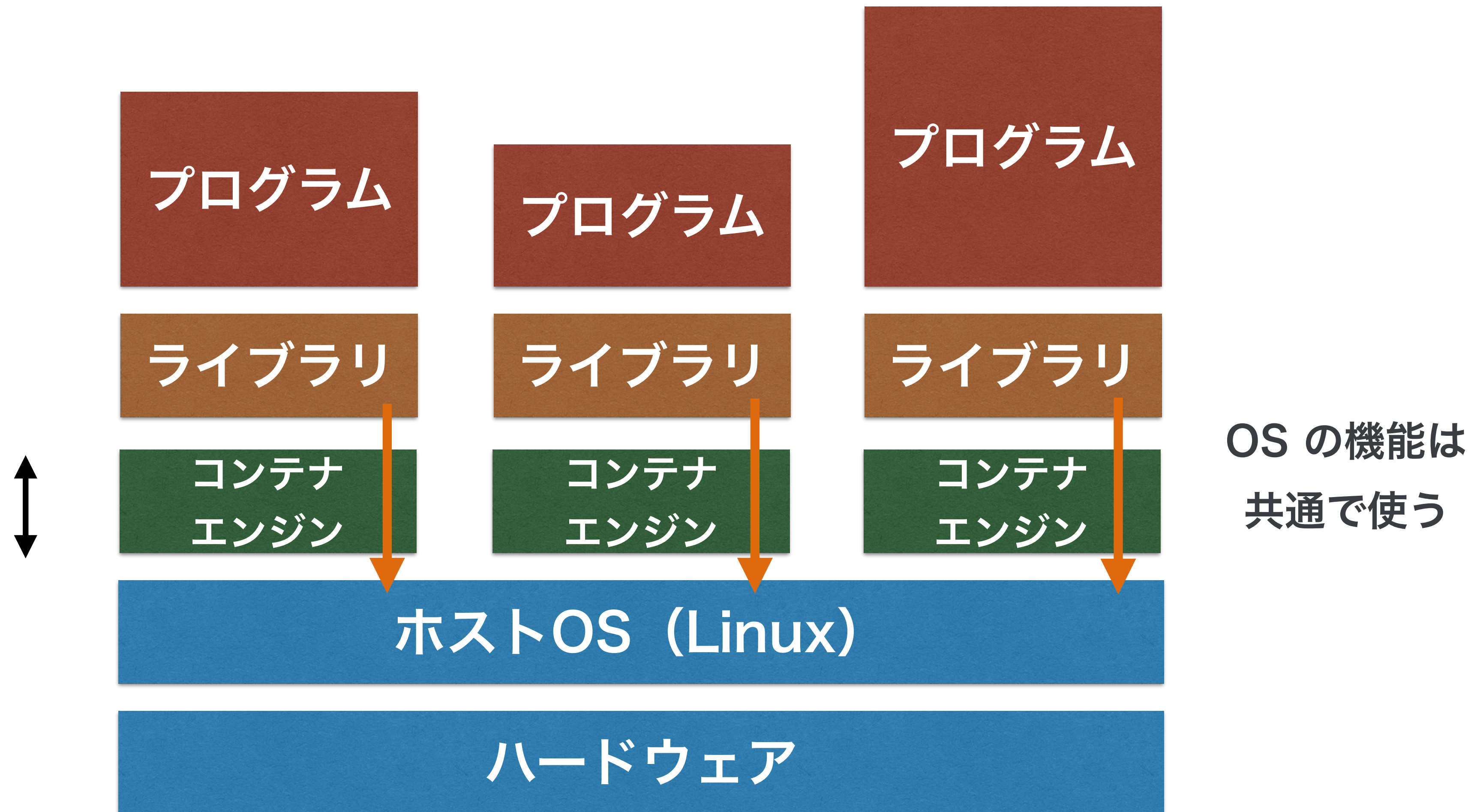


# いわゆる仮想化（例）





# コンテナ型「仮想化」





# コンテナによる「仮想化」技術要素

What Can  
I See?

namespaces

pid

mnt

net

utc

usr

chroot

What Can  
I Do?

capabilities

net\_bind\_service



sys\_reboot

seccomp

open()



init\_module()

LSMs

apparmor

selinux

Resource  
Isolation

cgroups

cpu

memory

net\_cls

blkio

devices

# プロセスからアクセスできる範囲の分離

---

- ・ファイルシステムの隔離: chroot/pivot\_root
- ・OSリソースの隔離: Linux Namespace
- ・PIDの隔離、マウントポイントの隔離、User ID mappingほか



# プロセスの権限の制限

---

- ・特権を分割し、一部のみの剥奪や付与: Linux Capability
- ・システムコール単位での呼び出し制限、トレース: seccomp
- ・Linux Security Module(LSM): AppArmor/SELinux



# プロセスのハードウェアリソースの制限

---

- ・古典的resource limit: ulimit(1), rlimit/prlimit
- ・cgroup (Control Groups)
  - ・CPU、メモリ、ブロックI/O、PID数ほか
  - ・単にプロセスのグループ単位での操作も行える





# Dockerの登場

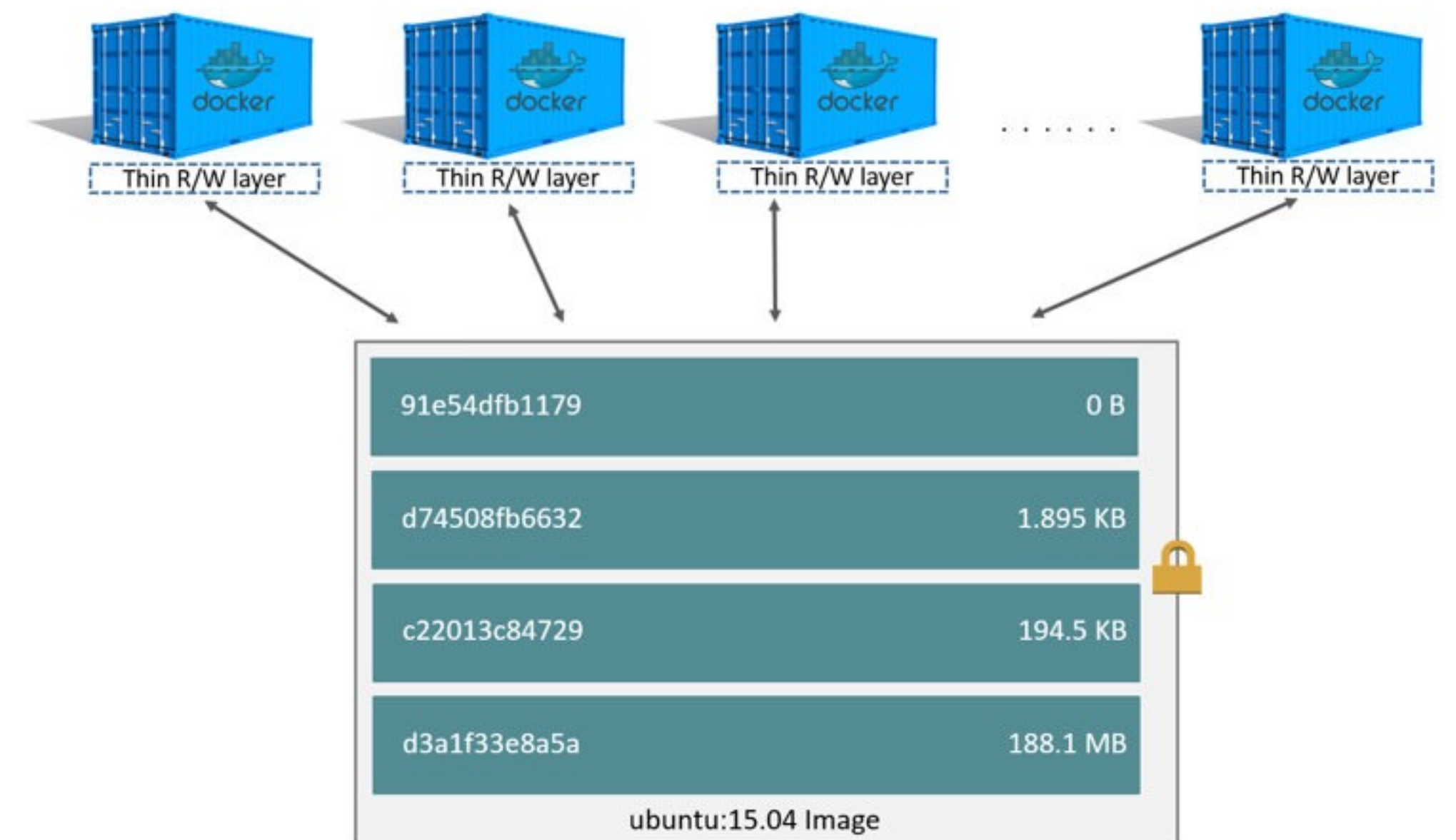
---

- ・PaaSであったDotCloud社の社内コンテナ基盤をOSS化したものがDockerの始まり。そこから一気に広まった
- ・Dockerは「イメージ」の概念をコモディティ化した
- ・コンテナの価値に、「ポータブルな環境の再現」が加わる
- ・コンテナの運用の留意点において「イメージの管理」が大きく占められるようになる



# Dockerにおけるイメージ

- ・差分ファイルシステム（Union Filesystem）を用いて変更箇所だけを追記して管理可能にした
- ・クラウド上にイメージをプッシュし、誰でもプルできるよう



<https://docs.docker.com/storage/storagedriver/>

# Docker/Kubernetes/OCI

---

- ・ランタイムの開発、乱立
- ・Open Container Initiativeの成立、OCI specの公開
- ・コンテナが満たすべき仕様の整理（イメージ、セキュリティ他）
- ・Kubernetesの登場
- ・Google社内の基盤「Borg」をベースに公開されたもの
- ・高レベルランタイム、低レベルランタイムのレイヤ分けが進む



# OCI spec

- <https://github.com/opencontainers/runtime-spec>

master runtime-spec / config-linux.mdGo to file...

saschagrunert

Add support for SCMP\_ACT\_KILL\_THREAD ... ✓

Latest commit 2fe0475 on Aug 25History

46 contributors

798 lines (627 sloc)32.3 KBRawBlame

### Linux Container Configuration

This document describes the schema for the [Linux-specific section](#) of the [container configuration](#). The Linux container specification uses various kernel features like namespaces, cgroups, capabilities, LSM, and filesystem jails to fulfill the spec.

### Default Filesystems

The Linux ABI includes both syscalls and several special file paths. Applications expecting a Linux environment will very likely expect these file paths to be set up correctly.

The following filesystems SHOULD be made available in each container's filesystem:

| Path     | Type                   |
|----------|------------------------|
| /proc    | <a href="#">proc</a>   |
| /sys     | <a href="#">sysfs</a>  |
| /dev/pts | <a href="#">devpts</a> |
| /dev/shm | <a href="#">tmpfs</a>  |

### Namespaces

A namespace wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the

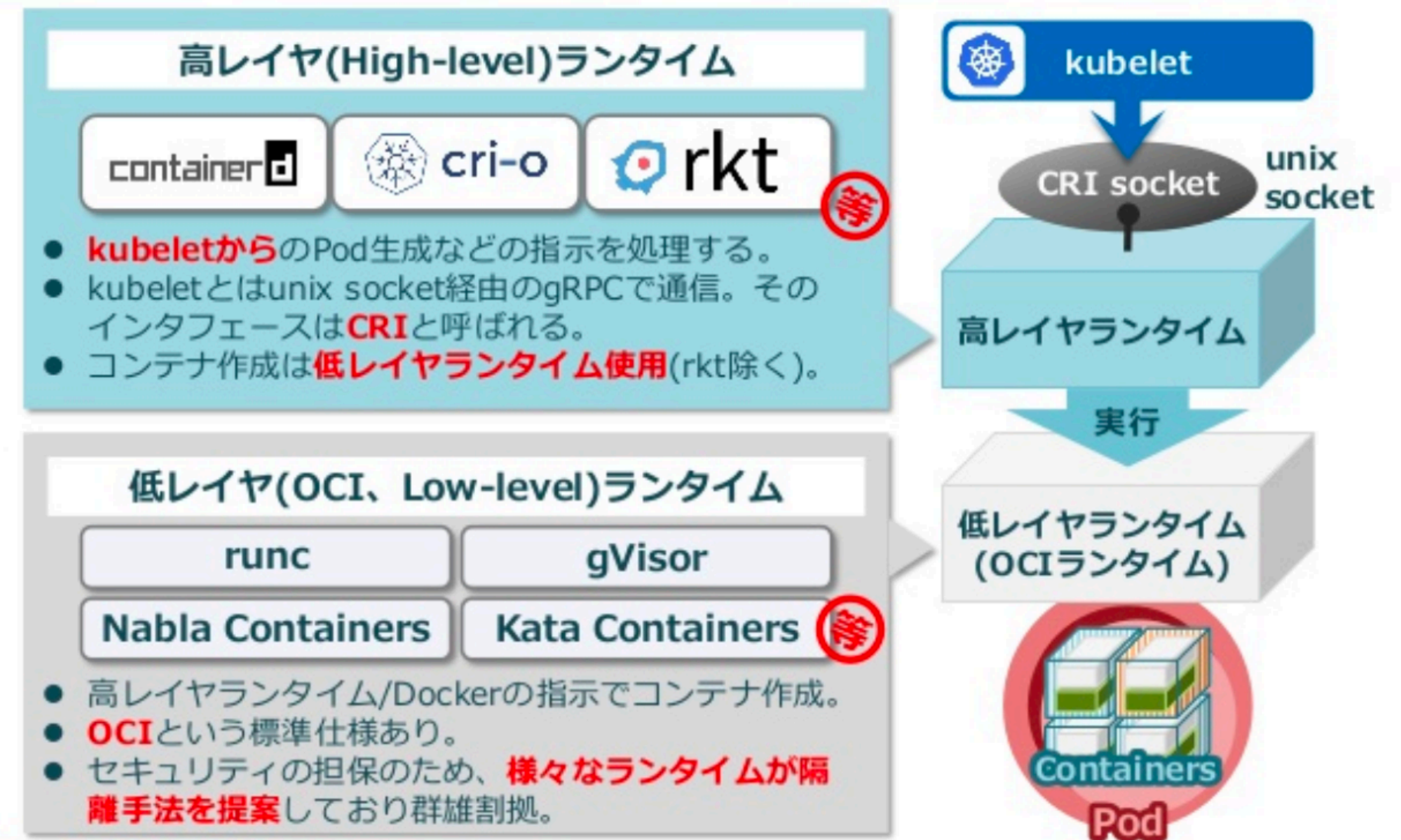




# 昨今のコンテナランタイム

- ・高レベルランタイム/低レベルランタイムを区別して理解する

## 本資料におけるkubelet以下のレイヤ区別



# 昨今のコンテナランタイム

---

- ・低レベルランタイムにおける実装戦略ごとの分類
- ・プロセスベースのリソース・権限分離を用いるもの: **runc**
- ・VMMをベースにできるだけ軽量にしたもの: **Firecracker, Kata**
- ・ユーザランドでサンドボックスを構成したもの: **gVisor**
- ・ユニカーネルをコンテナに適用したもの: **Nabla Container**



# 参考文献（論文）

---

- ・松本亮介, 「Webサーバの高集積マルチテナントアーキテクチャに関する研究」, 2017年度 京都大学大学院情報学研究科 博士論文学位審査公聴会, May 2017 (\*)
- ・Allison Randal, 「The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers」, ACM Computing Surveys, Volume 53, Issue 1, Article 5 (2020), Apr 2019 (\*\*)

(\*) <https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/225954>

(\*\*) <https://arxiv.org/abs/1904.12226>

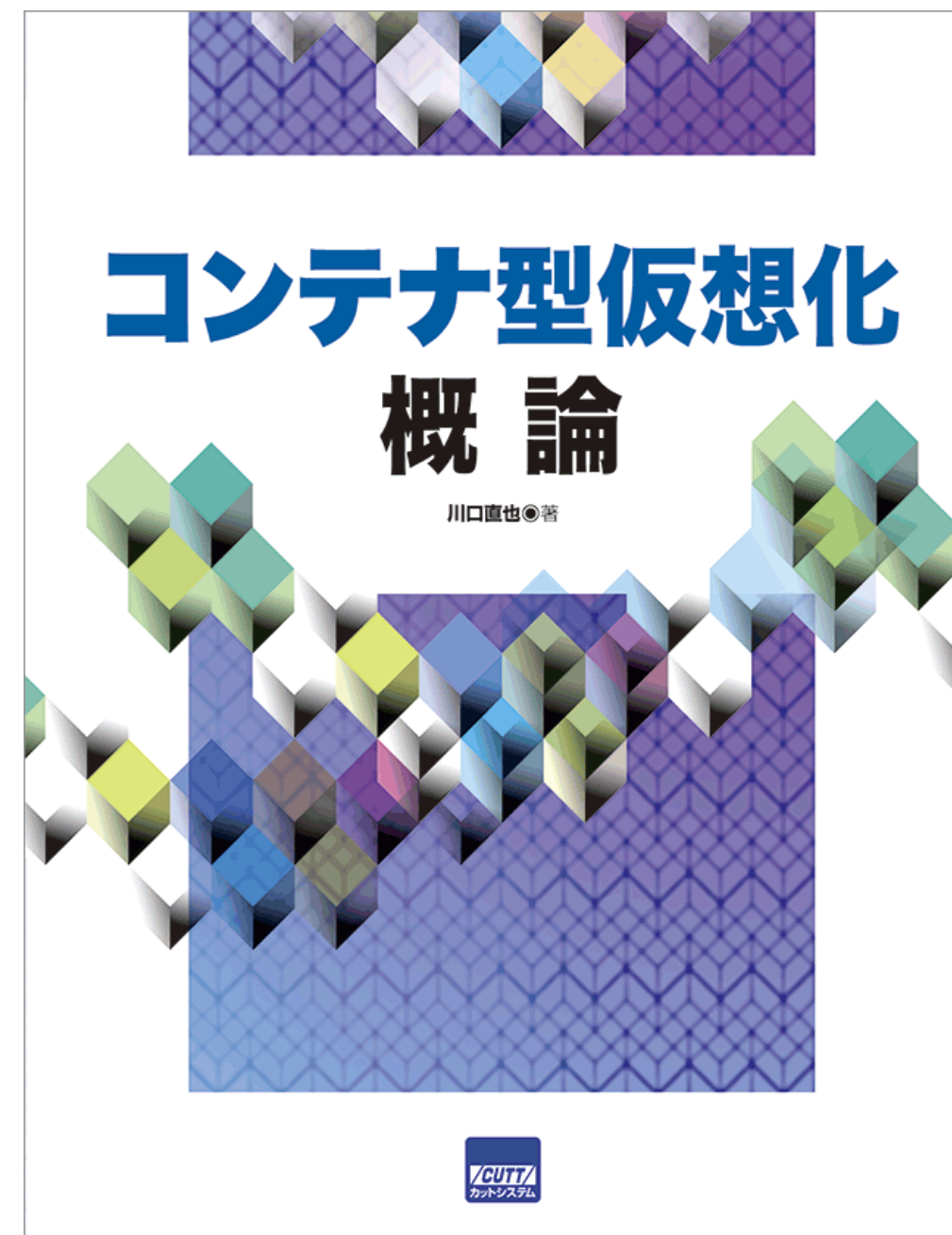




# 参考文献（書籍）

---

- ・ コンテナ型仮想化概論（川口直也, 2020, カットシステム）
- ・ <http://www.cutt.co.jp/book/978-4-87783-478-4.html>





# 参考資料（その他）

---

- ・近藤宇智朗, 森田浩平 「コンテナのセキュリティを中身から理解しよう」 <https://speakerdeck.com/udzura/inside-out-container-and-its-security>
- ・松本亮介 「ユビキタスデータセンターOSの文脈におけるコンテナ実行環境の分類」 <https://hb.matsumoto-r.jp/entry/2019/02/08/135354>



# 参考資料（その他: 2）

---

- ・徳永航平 「今話題のいろいろなコンテナランタイムを比較してみた」  
<https://www.slideshare.net/KoheiTokunaga/ss-123664087>
- ・宮下剛輔 「Compare OCI Runtimes」 <https://speakerdeck.com/mizzy/compare-oci-runtimes>
- ・加藤泰文 「LXCで学ぶコンテナ入門 - 軽量仮想化環境を実現する技術」 [https://gihyo.jp/admin/serial/01/linux\\_containers/0001](https://gihyo.jp/admin/serial/01/linux_containers/0001)



# コンテナの利用と オーケストレーション

# コンテナの重要なユースケース

---

- ・環境のポータブル化
- ・環境構築の自動化（オーケストレーション）
- ・環境のコード化（Infrastructure as Code）
- ・集約された環境提供
- ・（サンドボックス？）



# コンテナ以前のIaC

---

- ・ **Chef/Puppet...** = インスタンスレベルの構成管理
- ・ **Terraform...** = アーキテクチャレベルの構成管理、IaaSの管理
  - ・ システム管理を自動化したいというモチベーションが最初にあった
  - ・ Agile Infra/DevOps - ソフトウェア開発のプラクティスをインフラ管理に応用するために使われ広まる
  - ・ インスタンスレベル→APIによるIaaSの操作と管理 に広がる





# コンテナのオーケストレーション

---

- ・「イメージ」の発達 = コンテナイメージをオーケストレーションのパーツとして捉えられるようになる。組み合わせベースのインフラ
- ・BorgからのKubernetes = コンテナを管理する分散アーキテクチャの一つのパターンができ、それがオープンになり再実装される
- ・Borg由来の様々な概念 - Reconciliation Loop、分散ストレージへの設定保存、MasterとKubelet、...



# コンテナベースを前提としたエコシステム

---

- ・「コンテナベースのインフラ」は今までに存在しなかったものなので、既存のミドルウェアの切り替わりが起こる
- ・CNCF (The Cloud Native Computing Foundation) の設立
- ・Cloud Nativeの定義 <https://github.com/cncf/toc/blob/master/DEFINITION.md>
- ・新世代の、あるいは位置づけを新しくしたミドルウェアたち:
  - ・Prometheus、Fluentd、CoreDNS、Envoy、Helm、Rook、Falco、OpenTracing、gRPC.....



# 参考文献・資料

---

- ・「Large-scale cluster management at Google with Borg」  
<https://pdos.csail.mit.edu/6.824/papers/borg.pdf>
- ・宮下剛輔 「Infrastructure as Codeのこれまでとこれから」  
<https://speakerdeck.com/mizzy/infra-study-meetup-number-1>



演習

# ここから先は手を動かす

---

- Gistはこちら: <https://gist.github.com/udzura/fa82a8cb49a611c5f6400e561cf38a1d>
- Gistを開きながら実際に入力していきましょう





# 演習・問題について

---

- ・オンラインで進捗状況を把握するため、スプレッドシートを共有しますので、お名前を書いて丸つけをして行ってください。
- ・問題の回答については後日 goocus にご回答お願いします。手元に控えておいてください。
- ・質問があればSlackにまずは投げただけだと。



前半終了