LinuC レベル2
技術解説セミナー

コンテナ技術・Docker/Kubernetesについて

2021/10/10（Sun） 13:00-14:15

LPI-Japanプラチナスポンサー 株式会社ゼウス・エンタープライズ
鯨井 貴博（LinuCエヴァンジェリスト）

Who are you?（講師紹介）

鯨井貴博
LPI-Japan プラチナスポンサー　株式会社ゼウス・エンタープライズ
LinuCエバンジェリスト

Linuxとの出会い

大学時代 Unixの存在を知り、日経Linuxを読み始める。
2000年にVine Linux 2.0で一度挫折を経験。
その悔しさを忘れきれず、2007年 他業種からIT業界に転職しLinuxに再チャレンジ。

SE・商用製品サポート・インストラクター・プロジェクト管理などを経験し、現在に至る。
自分自身が学習で苦労した経験から、初心者を含む受講者に分りやすい講義を行うように心がけている。

また、興味の向くIT技術・オープンソースソフトウェアなどについて、
Opensourcetehブログ（https://www.opensourcetech.tokyo/）で執筆中。
実際に自分でやってみる/使ってみる・開発者本人から話を聞いてみることを大切にしています。


Linus Torvaldsさん(Linux開発者)


Igor Sysoevさん(nginx開発者)


Alexei Vladishevさん(Zabbix開発者)

ゼウス・エンタープライズからのお知らせ

提供するITサービス

リナックス・ネットワークグループ

リナックスに特化したエンジニアとヘルプデスクの派遣型支援サービスを行っています。研修は、エンジニアの育成に特化したLPI-Japanのアカデミック認定校「Zeus Linux Training Center / Zeus Network Training Center」で行います。資格取得の支援、新しい知識の習得に努め、つねに高水準のサービスの提供をめざしています。

コンピュータシステムの企画・開発・保守
請負業務
特定労働者派遣
・ヘルプデスク
・エンジニア

アプリケーショングループ

Webアプリケーションやお客様のシステムなど、プログラミング言語を用いた開発支援サービスを行っています。
また、「美味しいお店選び」をモットーとしたグルメサイト「Gourmet True」を2020年1月にリリース。これを皮切りに様々なインターネットサービスも開発しています。

アプリケーション開発
システム開発

クロスボーダーSE&ITソリューショングループ

語学力堪能なエンジニア(クロスボーダーSE)とヘルプデスク(バイリンガルコーディネータ)による支援サービスを行っています。
技術用語を多く含む会話の通訳、ドキュメントの翻訳という専門性を活かし、オフショア開発のプロジェクト、外国人技術者を多数かかえる現場などにありがちな意思疎通でのデメリット回避などにカを発揮します。

コンピュータシステムの企画・開発・保守
翻訳業務　通訳業務
給与代行　与信代行サービス　特定労働者派遣
・バイリンガルコーディネータ
・クロスボーダー採

Zeus Linux Training Center / Zeus Network Training Center

当社で培われた技術者向け研修カリキュラムを、一般のお客様や企業様向けに提供することを目的としたITキャリアスクール「Zeus Linux Training Center / Zeus Network Training Center」を運営しています。
当スクールはLPI-Japanのアカデミック認定校です。未経験者からプロの技術者養成まで、「現場ですぐに活躍できる人材」の育成をめざした研修は、技術力を求める企業からも高い評価をいただいています。

教育事業（BtoB、BtoC）
IT・英会話・マナー
社員教育
就職会場

https://www.zeus-enterprise.co.jp/solution/service

提供するITサービス

銀座ほこてん
子供プログラミング教室

COURSE
～コース紹介～

Minecraftコース

大人気ゲーム！Minecraftを使って楽しくプログラミング！

「Minecraft MakeCode」では、通常のMinecraftとは違い、エージェントという小さなロボットをプログラムによって操ることで、プレイヤーの代わりに様々な作業をとても短い時間で行わせることができます！プログラミング的手法を使い、Minecraftの世界を自由に作り上げましょう！
→各コマ、集合型レッスン定員12名・オンラインレッスン定員3名



※ 保護者同伴可能

会場	銀座ほこてん子供プログラミング教室 〒104-0061 東京都中央区銀座５丁目８－２０ 銀座コア8階
対象年齢	小学校3年～6年
講習時間	10:30~11:30/12:00~13:00/13:30~14:30/ 15:00~16:00/16:30~17:30
講習曜日	毎日
持ち物	筆記用具
入学金	ありません
月謝	6,000円～（税込）/ 月２回～（1コマ60分）
教材費	Minecraftのライセンス代：3300円 （税込） / テキスト代：2530円（税込）
無料体験	好評受付中！必要機材は全てお貸しします！

オンラインレッスン対応!!

当スクールでは通常のレッスンをオンラインでもご受講いただけます!
インターネット環境とPCをお持ちでしたら、Zoomを使用し
オンラインコースでもプログラミングを学べます!

キャンペーン
兄弟・姉妹一緒のお申し込みで 10 %OFF! 会員月謝がずっと

https://www.it-training.tokyo/kids/index.html

提供するITサービス

8月2日(月)オープン

ゼウス・エンタープライズ貸会議室 池袋は、
池袋駅および東池袋駅からアクセス良好。
100名収容可能な会議室です。

音響・映像設備、無料Wi-Fi、高速ネット回線完備で快適にご利用頂けます。
スクール、会議、発表会、説明会、オンライン配信(WEB会議・ウェビナー)等の
シーンでご活用ください。

スクール　会議　発表会　説明会　オンライン配信（WEB会議・ウェビナーなど）

https://www.zeus-enterprise.co.jp/ikebukuro_office/

アジェンダ

ZEUS enterprise

1. LinuCについて
   - 試験概要と特徴

2. 技術解説
   コンテナ技術
   Docker/Kubernetesについて
   - 1.01.2 仮想マシン・コンテナの概念と利用
   - 2.06.1 コンテナの仕組み
   - 2.06.2 Dockerコンテナとコンテナイメージの管理

   実機におけるコンテナ操作 by Katacoda

3. Appendix
   Podmanについて

4. Q&A

LinuC

本日のゴール

- オンプレ/仮想マシン/コンテナの違いを理解する
- コンテナ技術の概要を理解する
- Dockerの操作を理解する
- Kubernetesについて知る

LinuC について

LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定

✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化・コンテナを問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した「今」身につけておくべき技術範囲を網羅

今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

✓Linuxの範疇だけにとどまらない領域までカバー

セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

クラウドを活用できるITエンジニアに必須の技術がまとまっている

ZEUS enterprise

AWSなどのパブリッククラウドを活用するための技術

間が欠けている状態

オンプレミスのサーバーサイドLinux技術

【今まで／その他】

AWSなどのパブリッククラウドを活用するための技術

仮想マシン／コンテナ技術、クラウドセキュリティ、アーキテクチャ、ほか

オンプレミスのサーバーサイドLinux技術

LinuC Version10.0

LinuC

LinuC Level1試験

ZEUS enterprise

101試験

1.01：Linuxのインストールと仮想マシン・コンテナの利用
- 1.01.1Linuxのインストール、起動、接続、切断と停止
- 1.01.2仮想マシン・コンテナの概念と利用
- 1.01.3ブートプロセスとsystemd
- 1.01.4プロセスの生成、監視、終了
- 1.01.5デスクトップ環境の利用

1.02：ファイル・ディレクトリの操作と管理
- 1.02.1ファイルの所有者とパーミッション
- 1.02.2基本的なファイル管理の実行
- 1.02.3ハードリンクとシンボリックリンク
- 1.02.4ファイルの配置と検索

1.03：GNUとUnixのコマンド
- 1.03.1コマンドラインの操作
- 1.03.2フィルタを使ったテキストストリームの処理
- 1.03.3ストリーム、パイプ、リダイレクトの使用
- 1.03.4正規表現を使用したテキストファイルの検索
- 1.03.5エディタを使った基本的なファイル編集の実行

1.04：リポジトリとパッケージ管理
- 1.04.1apt コマンドによるパッケージ管理
- 1.04.2Debianパッケージ管理
- 1.04.3yumコマンドによるパッケージ管理
- 1.04.4RPMパッケージ管理

1.05：ハードウェア、ディスク、パーティション、ファイルシステム
- 1.05.1ハードウェアの基礎知識と設定
- 1.05.2ハードディスクのレイアウトとパーティション
- 1.05.3ファイルシステムの作成と管理、マウント

102試験

1.06：シェルおよびスクリプト
- 1.06.1シェル環境のカスタマイズ
- 1.06.2シェルスクリプト

1.07：ネットワークの基礎
- 1.07.1インターネットプロトコルの基礎
- 1.07.2基本的なネットワーク構成
- 1.07.3基本的なネットワークの問題解決
- 1.07.4クライアント側のDNS設定

1.08：システム管理
- 1.08.1アカウント管理
- 1.08.2ジョブスケジューリング
- 1.08.3ローカライゼーションと国際化

1.09：重要なシステムサービス
- 1.09.1システム時刻の管理
- 1.09.2システムのログ
- 1.09.3メール配送エージェント(MTA)の基本

1.10：セキュリティ
- 1.10.1セキュリティ管理業務の実施
- 1.10.2ホストのセキュリティ設定
- 1.10.3暗号化によるデータの保護
- 1.10.4クラウドセキュリティの基礎

1.11：オープンソースの文化
- 1.11.1オープンソースの概念とライセンス
- 1.11.2オープンソースのコミュニティとエコシステム

https://linuc.org/linuc1/range/101.html
https://linuc.org/linuc1/range/102.html

LinuC

LinuC Level2試験

ZEUS enterprise

201試験

- 2.01：システムの起動とLinuxカーネル
  - 2.01.1 ブートプロセスとGRUB
  - 2.01.2 システム起動のカスタマイズ
  - 2.01.3 Linux カーネルの構成要素
  - 2.01.4 Linuxカーネルのコンパイル
  - 2.01.5 カーネル実行時における管理とトラブルシューティング
- 2.02：ファイルシステムとストレージ管理
  - 2.02.1 ファイルシステムの設定とマウント
  - 2.02.2 ファイルシステムの管理
  - 2.02.3 論理ボリュームマネージャの設定と管理
- 2.03：ネットワーク構成
  - 2.03.1 基本的なネットワーク構成
  - 2.03.2 高度なネットワーク構成
  - 2.03.3 ネットワークの問題解決
- 2.04：システムの保守と運用管理
  - 2.04.1 makeによるソースコードからのビルドとインストール
  - 2.04.2 バックアップとリストア
  - 2.04.3 ユーザへの通知
  - 2.04.4 リソース使用状況の把握
  - 2.04.5 死活監視、リソース監視、運用監視ツール
  - 2.04.6 システム構成ツール
- 2.05：仮想化サーバー
  - 2.05.1 仮想マシンの仕組みとKVM
  - 2.05.2 仮想マシンの作成と管理
- 2.06：コンテナ
  - 2.06.1 コンテナの仕組み
  - 2.06.2 Dockerコンテナとコンテナイメージの管理

202試験

- 2.07：ネットワーククライアントの管理
  - 2.07.1 DHCPサーバーの設定と管理
  - 2.07.2 PAM認証
  - 2.07.3 LDAPクライアントの利用方法
  - 2.07.4 OpenLDAPサーバーの設定
- 2.08：ドメインネームサーバー
  - 2.08.1 BINDの設定と管理
  - 2.08.2 ゾーン情報の管理
  - 2.08.3 セキュアなDNSサーバーの実現
- 2.09：HTTPサーバーとプロキシサーバー
  - 2.09.1 Apache HTTPサーバーの設定と管理
  - 2.09.2 OpenSSLとHTTPSの設定
  - 2.09.3 nginxの設定と管理
  - 2.09.4 Squidの設定と管理
- 2.10：電子メールサービス
  - 2.10.1 Postfixの設定と管理
  - 2.10.2 Dovecotの設定と管理
- 2.11：ファイル共有サービス
  - 2.11.1 Sambaの設定と管理
  - 2.11.2 NFSサーバーの設定と管理
- 2.12：システムのセキュリティ
  - 2.12.1 iptables や firewalld によるパケットフィルタリング
  - 2.12.2 OpenSSH サーバーの設定と管理
  - 2.12.3 OpenVPNの設定と管理
  - 2.12.4 セキュリティ業務
- 2.13：システムアーキテクチャ
  - 2.13.1 高可用システムの実現方式
  - 2.13.2 キャパシティプランニングとスケーラビリティの確保
  - 2.13.3 クラウドサービス上のシステム構成
  - 2.13.4 典型的なシステムアーキテクチャ

https://linuc.org/linuc2/range/201.html
https://linuc.org/linuc2/range/202.html

LC LinuC

LinuC 学習のコツ

①出題範囲の内容について調べてみる
　公式ドキュメント・技術書など
②実際に操作してみる
　これが大事！
③学習の補助教材などを利用する
・メールマガジン
・標準教科書
・過去のセミナー資料
　詳細は、https://lpi.or.jp/learning/

LPI-Japan LinuC通信「レベル2・レベル3を受けてみよう！」でサンプル問題作ってるので、よかったら登録してください！

LinuC 学習のコツ

④過去セミナーの動画
https://www.youtube.com/user/LPIJapan

LinuC 学習のコツ

ZEUS enterprise

学習の具体的な進め方(2~3か月程度)

試験範囲の確認(LinuC HP)

●月頃、受験しようという目標を立てる

↓

LinuC認定教材の購入・1週目読込 https://lpi.or.jp/linuc1/book.shtml https://lpi.or.jp/linuc2/book.shtml

↓

LinuC認定教材・Webサイトを参考に、実機操作(サーバ構築やコマンド操作)を試す

※操作やトラブルシュートで力が身に付く！

↓

LinuC認定教材 2週目読込

●月●日頃、受験しようと決める

↓

問題集やメルマガサンプル問題で理解力確認 https://linuc.org/study/samples/

※理解不足箇所の洗い出し

↓

LinuC認定教材 3週目

※弱点の補強

↓

受験申込

受験日の変更も可能なので安心

↓

問題を8~9割以上、正解となるまで繰り返し解く

苦手な部分を重点的に復習

↓

受験

・受験まで継続して学習すること
・繰り返し学習し、理解度/問題正解率を高めた状態で受験すること

LC LinuC

コンテナ技術

1.01.2 仮想マシン・コンテナの概念と利用

重要度　　4

概要

仮想マシンとコンテナを実現する基本技術を理解している。
ハイパーバイザー、コンテナのそれぞれの特徴と違いを理解する。
仮想マシン、コンテナの起動、停止ができる。

詳細

カーネルとハイパーバイザーと仮想マシンの関係を理解している。
　ハイパーバイザー、仮想化支援機能
　仮想マシンとコンテナの特徴を理解している。
ホストOSとコンテナの関係を理解している。
仮想マシンの起動と停止ができる。
　virsh
仮想マシンにログインできる。
コンテナの起動と停止ができる。
　docker

2.06.1 コンテナの仕組み

重要度　　2

概要　　基本的なコンテナの仕組みについて理解している。
詳細

物理マシン、仮想マシン、コンテナの特徴と違いを理解している。
コンテナのファイルシステムとイメージの関係を知っている。
コンテナを実現する技術の概念を知っている。
　名前空間, cgroups

2.06.2 Dockerコンテナとコンテナイメージの管理

重要度　　3

概要

Dockerを導入してコンテナ実行環境を構築できる。
Dockerコンテナを実行できる。
コンテナイメージを管理できる。

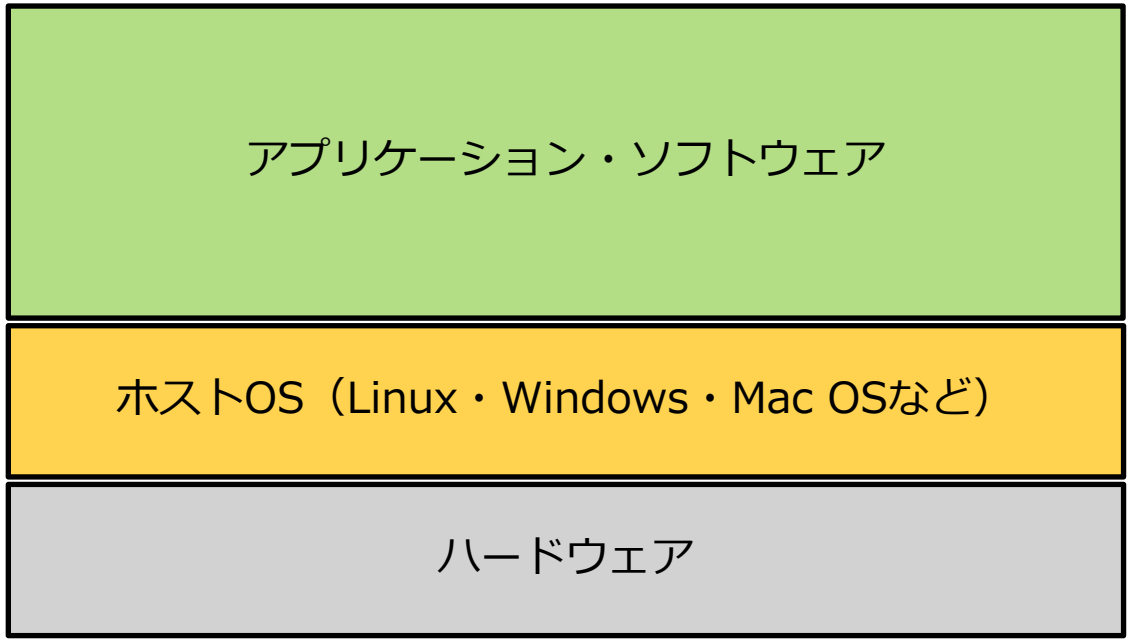
詳細

Dockerを導入して、ネットワークを構成する。
　ポート変換, フラットL2ネットワーク
Dockerコンテナを実行して、停止する。
　docker ps/stats, docker run/create/restart, docker pause/unpause, docker stop/kill, docker rm
Dockerコンテナに接続してプロセスを実行する。
　docker attach, docker exec
コンテナイメージを管理する。
　Dockerレジストリ: docker images, docker pull, docker rmi, docker import
　Dockerfile: docker build, docker commit

物理マシン・仮想マシン・コンテナの違い

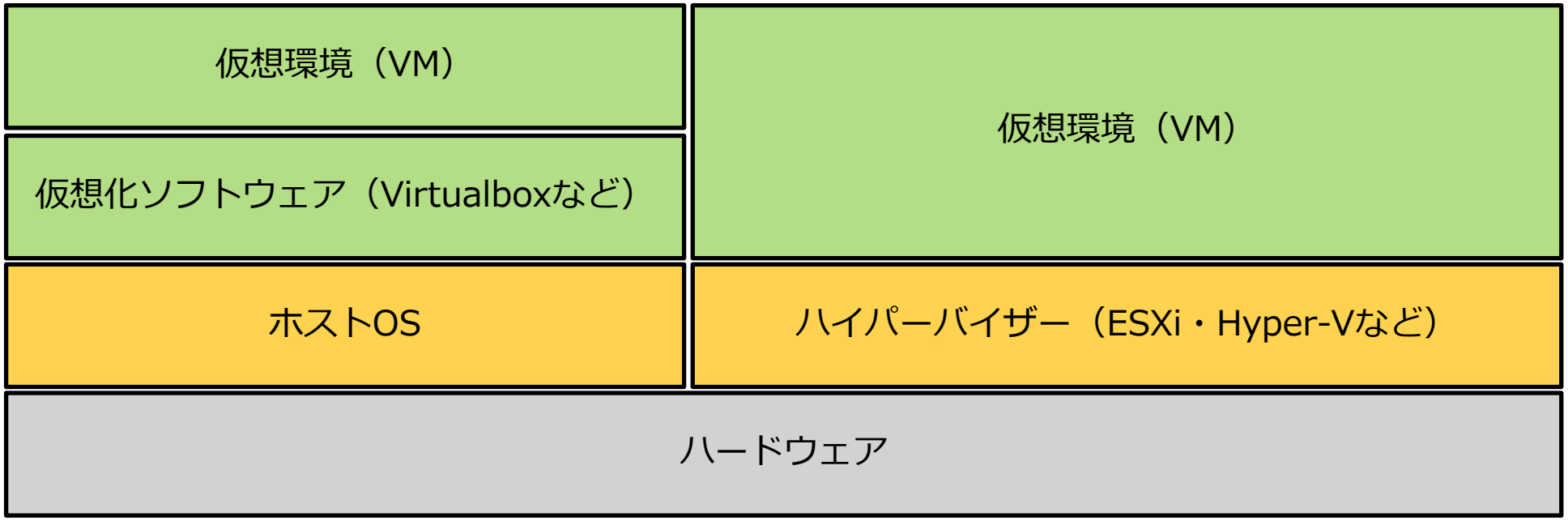
物理マシンの特徴

1HWを1OSで占有できる　⇔　環境構築まで長時間要する（HW調達・インストールの手間など）
柔軟なリソース調整が困難（過剰になったり、不足したり）



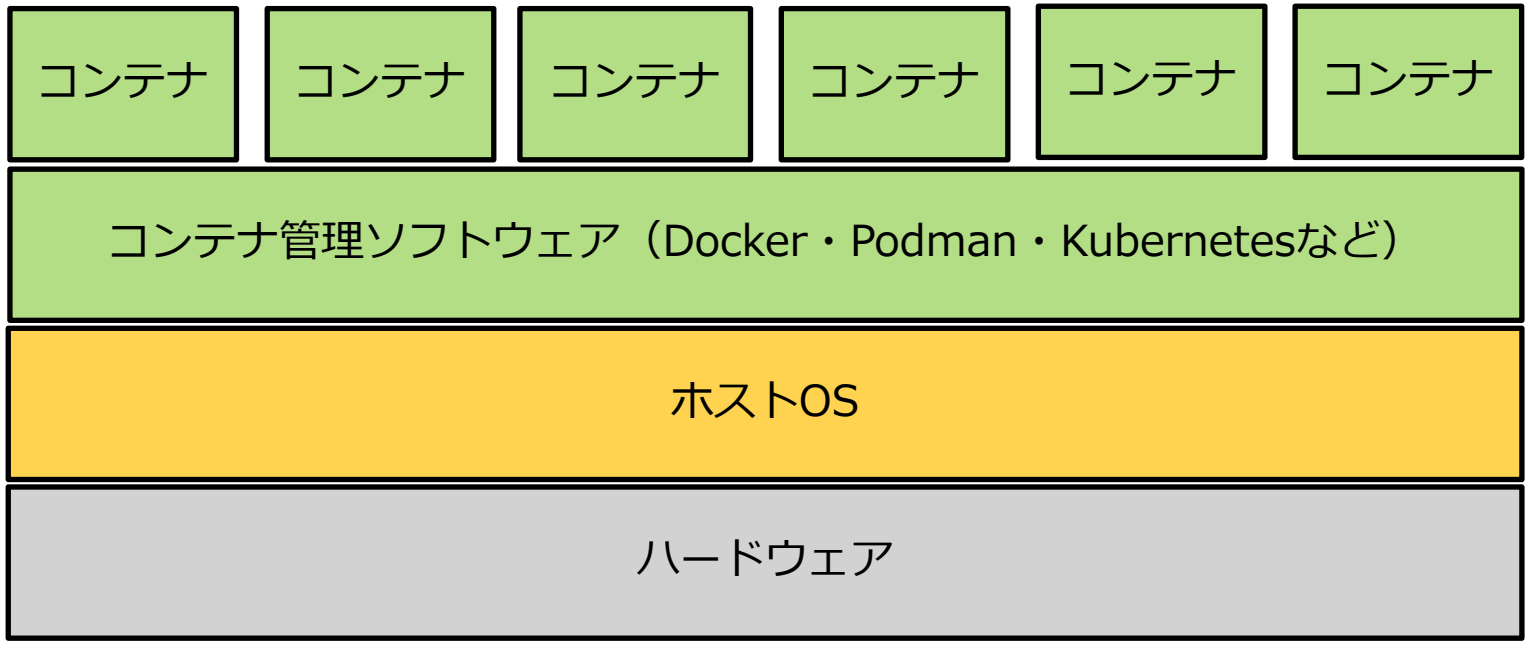
仮想マシンの特徴

1HWを複数OSで共有　　⇔　柔軟なリソース調整（増減）が可能
環境構築まで長時間要する（インストールの手間など）

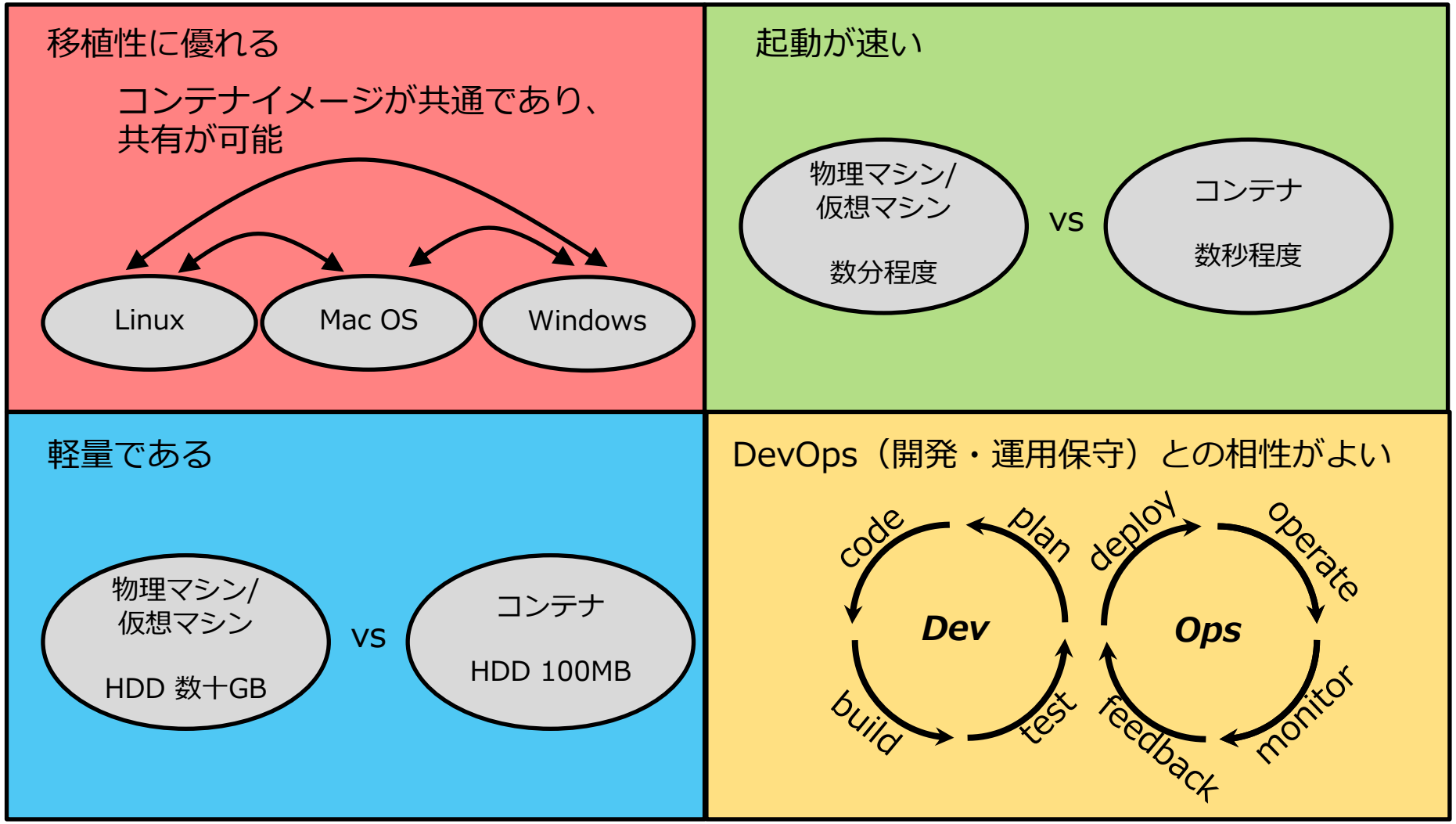


コンテナの特徴

1HWを複数コンテナで共有 ⇔　柔軟なリソース調整（増減）が可能
環境構築が容易（インストールの手間など）
軽量



コンテナの特徴



コンテナを支える技術 名前空間（namespaces）

ZEUS enterprise

名前空間は、 グローバルシステムリソースを抽象化層で覆うことで、 名前空間内のプロセスに対して、 自分たちが専用の分離されたグローバルリソースを持っているかのように見せる仕組みである。 グローバルリソースへの変更は、 名前空間のメンバーである他のプロセスには見えるが、 それ以外のプロセスには見えない。
名前空間の一つの利用方法はコンテナーの実装である。

名前空間	定数	分離対象
Cgroup	CLONE_NEWCGROUP	Cgroupルートディレクトリ
IPC	CLONE_NEWIPC	System V IPC/POSIX メッセージキュー
Network	CLONE_NEWNET	ネットワークデバイス/スタック/ポート など
Mount	CLONE_NEWNS	マウントポイント
PID	CLONE_NEWPID	プロセスID
Time	CLONE_NEWTIME	起動時刻/モノトニック時刻
User	CLONE_NEWUSER	ユーザID/グループID
UTS	CLONE_NEWUTS	ホスト名/NISドメイン名

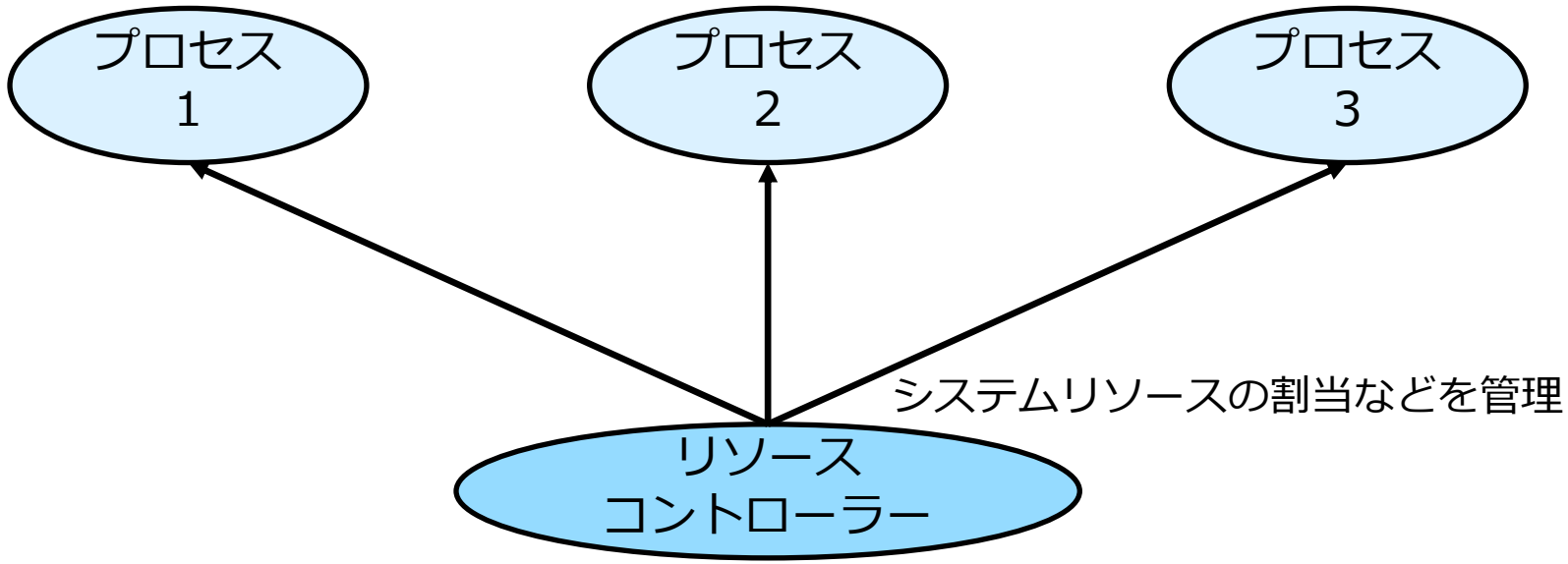
これらを使い
各コンテナを仮想に
隔離する
Linux kernelの機能

http://man7.org/linux/man-pages/man7/namespaces.7.html
http://manpages.ubuntu.com/manpages/focal/ja/man7/namespaces.7.html
https://linuxjm.osdn.jp/html/LDP_man-pages/man7/namespaces.7.html

LinuC

コンテナを支える技術 cgroups

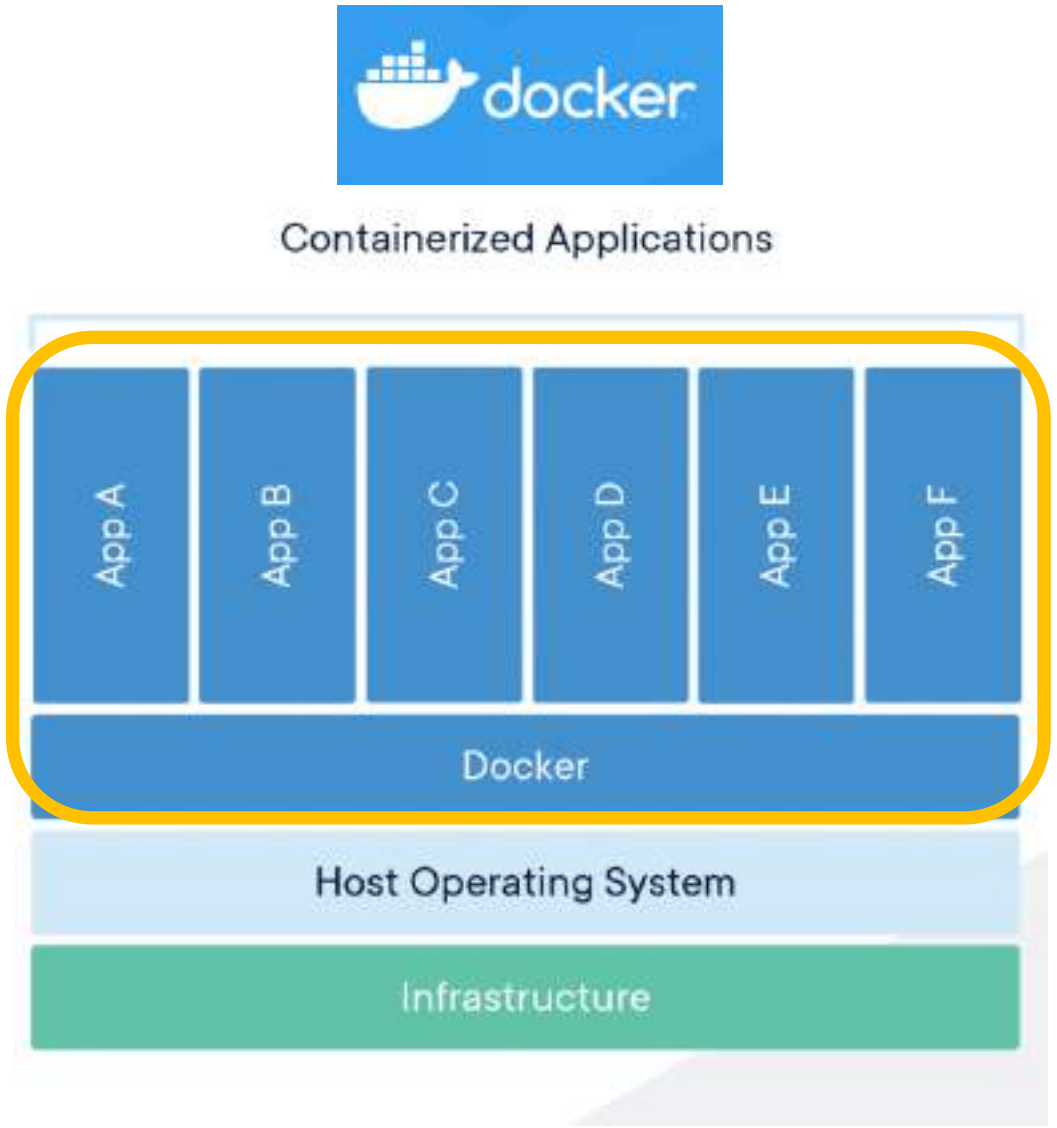
cgroups（Control Groups）は、Linuxカーネルの機能の1つで、プロセスを階層的なグループに編成し、さまざまなタイプのリソースの使用（メモリ、CPU,ネットワーク帯域など）を制限および監視できるようにします。
カーネルのcgroupインターフェースは、cgroupfsと呼ばれる疑似ファイルシステムを介して提供されます。



http://man7.org/linux/man-pages/man7/cgroups.7.html
https://wiki.archlinux.jp/index.php/Cgroups
https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/8/html/managing_monitoring_and_updating_the_kernel/setting-limits-for-applications_managing-monitoring-and-updating-the-kernel

Docker

Dockerとは



コンテナ

コンテナ管理ソフトウェア

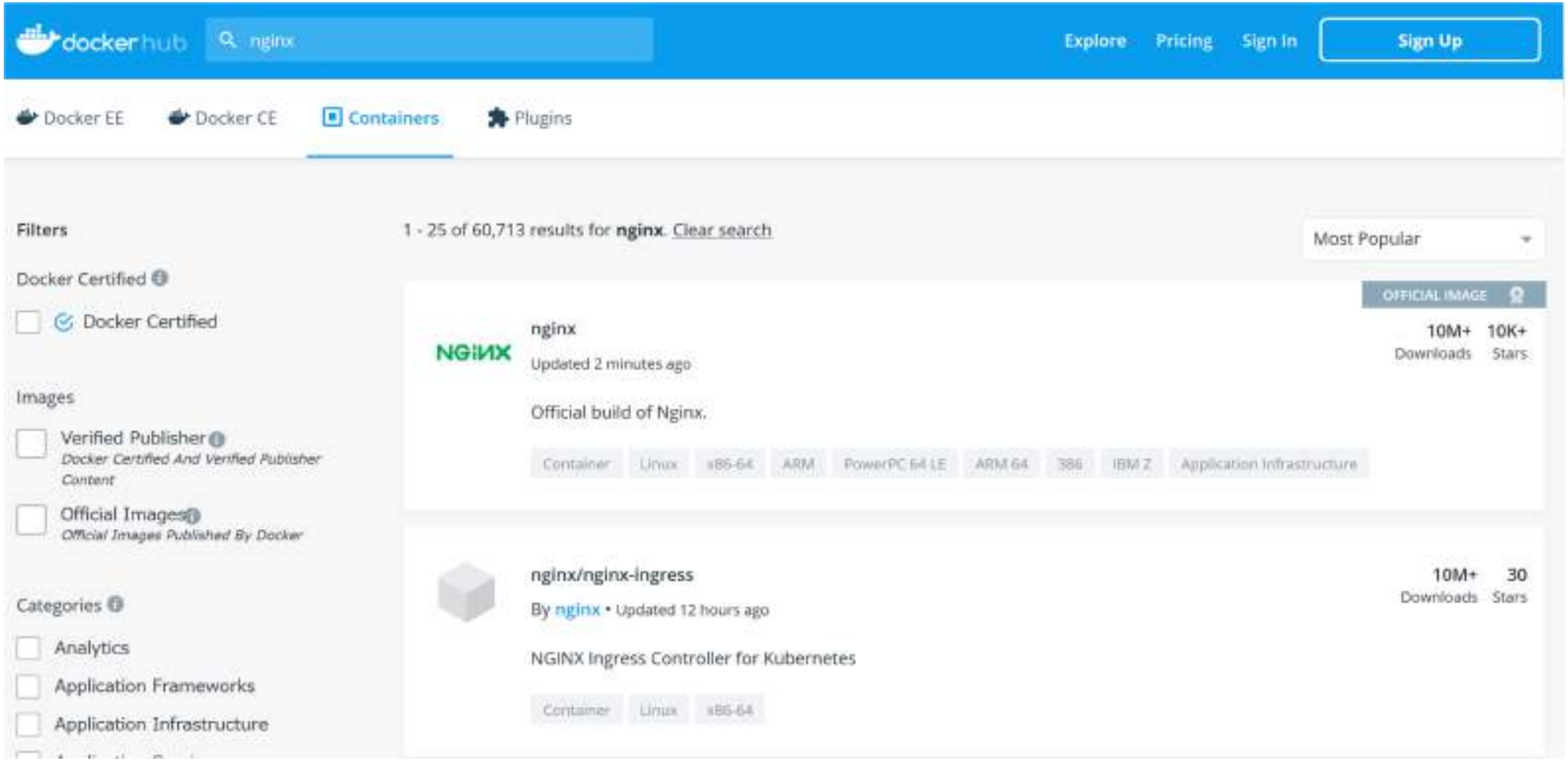
ホストOS

ハードウェア

https://www.docker.com/resources/what-container

コンテナイメージの配布場所

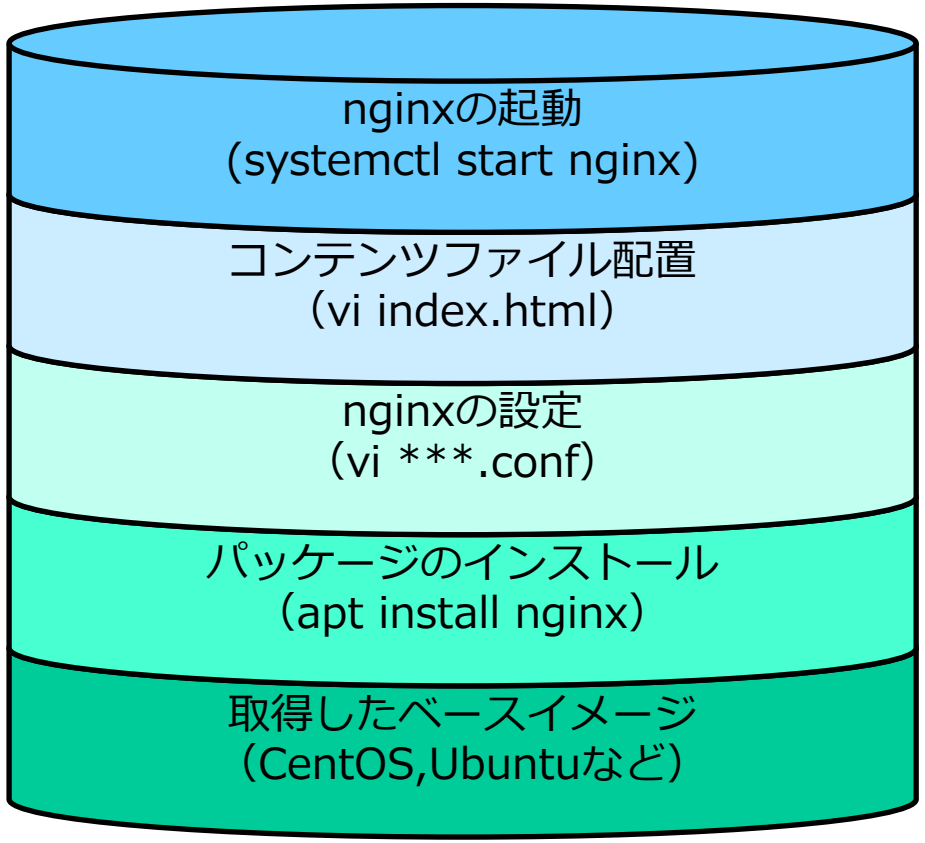
DockerHubなどにあるコンテナイメージを利用できる



https://hub.docker.com/search?q=&type=image

コンテナイメージの構成

ベースイメージを取得し、そこにイメージの層（レイヤー）を重ねて利用する

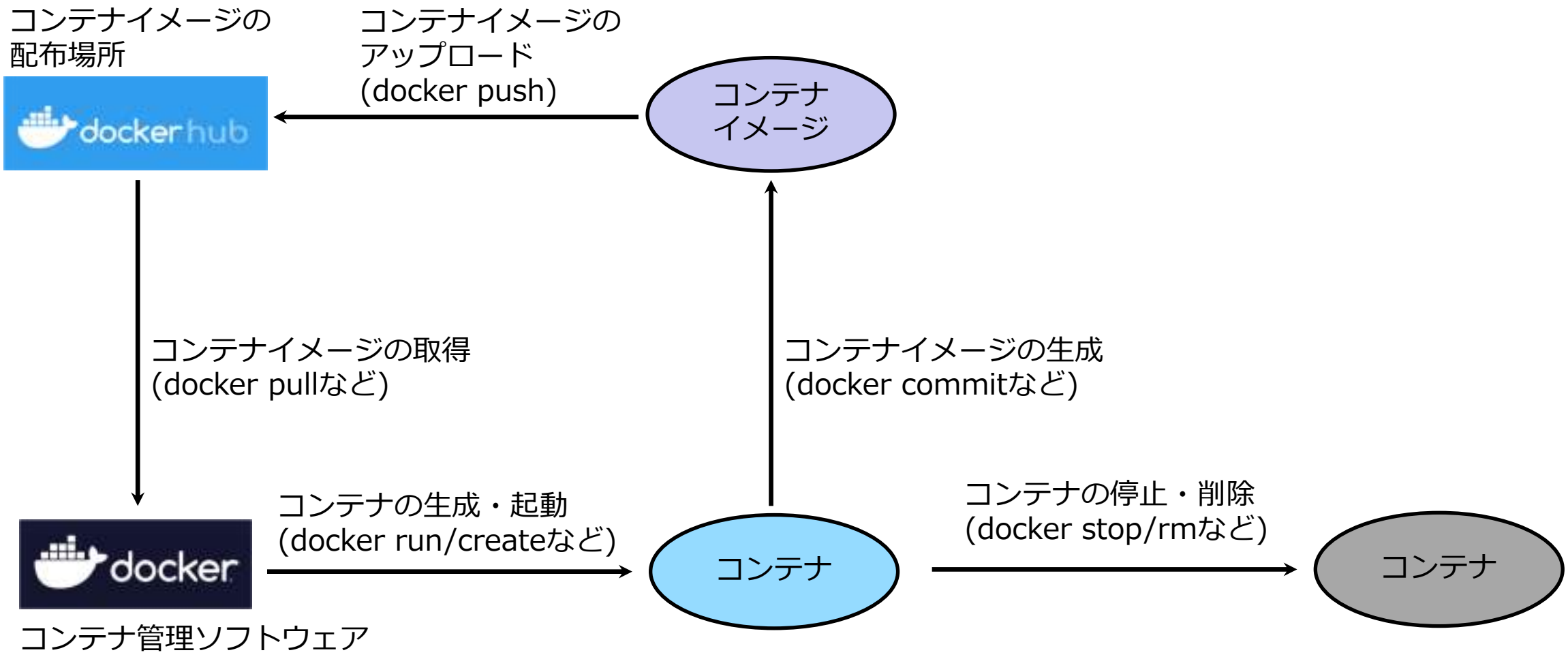


ベースイメージに対して、
コマンドを実施しサービスを構築

Overlay Filesystemという仕組みを使っている

https://www.kernel.org/doc/html/latest/filesystems/overlayfs.html
https://docs.docker.jp/v17.06/engine/userguide/storagedriver/overlayfs-driver.html

コンテナ・コンテナイメージのライフサイクル



Dockerによるコンテナの操作

ZEUS enterprise

docker images（取得したコンテナイメージの一覧表示）

使い方：*docker images*

docker pull（コンテナイメージの取得）

使い方：*docker pull URL/*取得するコンテナイメージ*[:タグ名]*
※デフォルトでは、*https://hub.docker.com/*からコンテナイメージ取得をする
※タブ名を未指定の場合、*:latest* が適用される

```
[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@cc6516dd95d9 ~]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
07aded7c29c6: Pull complete
bbe0b7acc89c: Pull complete
44ac32b0bba8: Pull complete
91d6e3e593db: Pull complete
8700267f2376: Pull complete
4ce73aa6e9b0: Pull complete
Digest: sha256:969419c0b7b0a5f40a4d666ad227360de5874930a2b228a7c11e15dedbc6e092
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f8f4ffc8092c        2 days ago          133MB
```

コンテナイメージ名とタグ

https://docs.docker.com/engine/reference/commandline/images/
https://docs.docker.com/engine/reference/commandline/pull/

LinuC

Dockerによるコンテナの操作

docker create（コンテナ生成）

使い方：*docker create* コンテナイメージ

docker ps（稼働コンテナの一覧表示）

使い方：*docker ps [-a]*
※*-a*をつけると、停止中などのコンテナを含めて表示する

```
[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f8f4ffc8092c        2 days ago          133MB

[root@cc6516dd95d9 ~]# docker create nginx
29cf25137989beb7d527f02f2e66a409f65b7aa90fcbecea7652f2847fa9cae3

[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[root@cc6516dd95d9 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
29cf25137989        nginx               "/docker-entrypoint.…"   14 seconds ago      Created                                 sharp_lamarr
```

コンテナID ※コンテナを制御するために使用

https://docs.docker.com/engine/reference/commandline/create/
https://docs.docker.com/engine/reference/commandline/ps/

Dockerによるコンテナの操作

ZEUS enterprise

docker run（コンテナ生成 & コンテナ起動）

使い方：*docker run オプション コンテナイメージ[:タグ名]*

-d：バックグラウンドでコンテナを実施
-i：コンテナの標準入力を開く
-t：端末（*TTY*）を割り当てる
-p：公開ポートを指定（ポート変換）
　　-p 8080:80 とした場合、ホスト*OS*の*TCP8080*番ポート経由で、コンテナの*TCP80*番ポートへアクセスする
--priviledge：*systemctl*を使える権限などを付与

```
[root@cc6516dd95d9 ~]# docker run --name nginx01 -d -p 8080:80 nginx
73760180eb30b9a93a1c57873a1406fe8e99d8060ae312254bcadebb3cb8913b
[
root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   4 seconds ago       Up 2 seconds        0.0.0.0:8080->80/tcp   nginx01
```

LinuC

https://docs.docker.com/engine/reference/commandline/run/

Dockerによるコンテナの操作

ZEUS enterprise

docker stats（コンテナの稼働状況確認）

使い方：*docker stats [コンテナID]*
※*ctrl* + *c* で終了させる

```
[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   2 minutes ago       Up 2 minutes        0.0.0.0:8080->80/tcp     nginx01

[root@cc6516dd95d9 ~]# docker stats

CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O           PIDS
73760180eb30        nginx01             0.00%               1.98MiB / 1.46GiB     0.13%               2.63kB / 0B         0B / 4.1kB          3
```

LinuC

https://docs.docker.com/engine/reference/commandline/stats/

Dockerによるコンテナの操作

docker pause（コンテナプロセスの中断）

使い方：*docker pause* コンテナ*ID*

docker unpause（コンテナプロセスの再開）

使い方：*docker unpause* コンテナ*ID*

```
[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   3 minutes ago       Up 2 minutes        0.0.0.0:8080->80/tcp     nginx01
[
root@cc6516dd95d9 ~]# docker pause 73760180eb30
73760180eb30

[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   3 minutes ago       Up 3 minutes (Paused)    0.0.0.0:8080->80/tcp     nginx01

[root@cc6516dd95d9 ~]# docker unpause 73760180eb30
73760180eb30

[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   3 minutes ago       Up 3 minutes        0.0.0.0:8080->80/tcp     nginx01
```

https://docs.docker.com/engine/reference/commandline/pause/
https://docs.docker.com/engine/reference/commandline/unpause/

Dockerによるコンテナの操作

docker stop（コンテナの停止）

使い方：*docker stop* コンテナ*ID*

-t：コンテナ停止の待ち時間を指定（デフォルト*10*秒）

docker kill（コンテナの強制停止）

使い方：*docker kill* コンテナ*ID*

docker start（停止しているコンテナの起動）

使い方：*docker start* コンテナ*ID*

docker restart（コンテナの再起動）

使い方：*docker restart* コンテナ*ID*

https://docs.docker.com/engine/reference/commandline/stop/
https://docs.docker.com/engine/reference/commandline/kill/
https://docs.docker.com/engine/reference/commandline/start/
https://docs.docker.com/engine/reference/commandline/restart/

Dockerによるコンテナの操作

ZEUS enterprise

docker attach（稼働コンテナへの接続）

使い方：*docker attach* コンテナ*ID*

※接続し作業完了後、*ctrl* + *c* でコンテナごと終了、*ctrl* + *p*・*ctrl* + *q* でコンテナ内のプロセスのみ終了となる

docker exec（稼働コンテナでのコマンド実行）

使い方：*docker exec* コンテナ*ID* 実行コマンド

-i：コンテナの標準入力を開く
-t：端末（*TTY*）を割り当てる

```
[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   5 minutes ago       Up 4 minutes        0.0.0.0:8080->80/tcp   nginx01

[root@cc6516dd95d9 ~]# docker exec -it 73760180eb30 /bin/bash

root@73760180eb30:/# uname -a
Linux 73760180eb30 3.10.0-1062.12.1.el7.x86_64 #1 SMP Tue Feb 4 23:02:59 UTC 2020 x86_64 GNU/Linux
root@73760180eb30:/# exit
exit
```

LinuC

https://docs.docker.com/engine/reference/commandline/attach/
https://docs.docker.com/engine/reference/commandline/exec/

Dockerによるコンテナの操作

ZEUS enterprise

docker rm（コンテナの削除）

使い方：*docker rm* コンテナ*ID*
※停止しているコンテナを削除

-f：稼働コンテナを強制削除

```
[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   7 minutes ago       Up 7 minutes        0.0.0.0:8080->80/tcp    nginx01

[root@cc6516dd95d9 ~]# docker stop 73760180eb30
73760180eb30

[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES

[root@cc6516dd95d9 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
73760180eb30        nginx               "/docker-entrypoint.…"   7 minutes ago       Exited (0) 7 seconds ago                        nginx01

[root@cc6516dd95d9 ~]# docker rm 73760180eb30
73760180eb30

[root@cc6516dd95d9 ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

LinuC

https://docs.docker.com/engine/reference/commandline/rm/

Dockerによるコンテナの操作

ZEUS enterprise

docker rmi（コンテナイメージの削除）

使い方：*docker rmi コンテナイメージ[:タグ名]*

```
[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f8f4ffc8092c        2 days ago          133MB

[root@cc6516dd95d9 ~]# docker rmi nginx
Untagged: nginx:latest
Untagged: nginx@sha256:969419c0b7b0a5f40a4d666ad227360de5874930a2b228a7c11e15dedbc6e092
Deleted: sha256:f8f4ffc8092c956ddd6a3a64814f36882798065799b8aedeebedf2855af3395b
Deleted: sha256:f208904eecb00a0769d263e81b8234f741519fefa262482d106c321ddc9773df
Deleted: sha256:ed6dd2b44338215d30a589d7d36cb4ffd05eb28d2e663a23108d03b3ac273d27
Deleted: sha256:c9958d4f33715556566095ccc716e49175a1fded2fa759dbd747750a89453490
Deleted: sha256:c47815d475f74f82afb68ef7347b036957e7e1a1b0d71c300bdb4f5975163d6a
Deleted: sha256:3b06b30cf952c2f24b6eabdff61b633aa03e1367f1ace996260fc3e236991eec
Deleted: sha256:476baebdfbf7a68c50e979971fcd47d799d1b194bcf1f03c1c979e9262bcd364

[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

LinuC

https://docs.docker.com/engine/reference/commandline/rmi/

Dockerによるコンテナの操作

ZEUS enterprise

docker export（コンテナからtarball作成）

使い方：*docker export コンテナID > tarballファイル名*

```
[root@cc6516dd95d9 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
d6bd1c3a6c3c        nginx:1.21.3        "/docker-entrypoint.…"   5 seconds ago       Up 4 seconds        0.0.0.0:8080->80/tcp     nginx02

[root@cc6516dd95d9 ~]# docker export d6bd1c3a6c3c > nginx02.tar

[root@cc6516dd95d9 ~]# ls
nginx02.tar  original-ks.cfg
```

LinuC

https://docs.docker.com/engine/reference/commandline/export/

docker import（tarballからコンテナイメージ作成）

使い方：*cat tarballファイル名 | docker import – コンテナイメージ名:タグ名*

```
[root@cc6516dd95d9 ~]# ls
nginx02.tar  original-ks.cfg

[root@cc6516dd95d9 ~]# cat nginx02.tar | docker import - nginx:import_test
sha256:a3b9f3b60499f28caa95435b1f4e5293baf704fd1e88b66334ee70026bdeb31d

[root@cc6516dd95d9 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               import_test         a3b9f3b60499        6 seconds ago       132MB
nginx               1.21.3              f8f4ffc8092c        2 days ago          133MB
```

https://docs.docker.com/engine/reference/commandline/import/

Dockerによるコンテナの操作

ZEUS enterprise

Dockerfileを使ったコンテナの生成

必要なもの（以下を同一ディレクトリに保存する）

- *Dockerfile*
- 設定ファイル*(nginx.repo)*や*Web*サーバのコンテンツファイル*(index.html)*など

Dockerfileのサンプル

FROM centos:latest・・・利用するベースイメージの指定
MAINTAINER SubMattNesk(kujiraitakahiro) <SMN@test.local>・・・メンテナーの情報記載
COPY ./nginx.repo /etc/yum.repos.d/・・・コンテナビルド時にコピーするファイルとその配置場所
RUN [“/bin/bash”,“-c”,“yum -y install nginx”]・・・コンテナビルド後、実行するコマンド
EXPOSE 80・・・コンテナで公開するポート
LABEL “version” = “0.02” ¥・・・コンテナに付与するラベル（複数行の場合、バックスラッシュを使う）
"description" = "presented by SubMattNesk"
ADD ./index.html /usr/share/nginx/html/・・・コンテナにコピーするファイルとその配置場所
ENTRYPOINT [“/usr/sbin/nginx”,“-g”,“daemon off;”]・・・コンテナ構築時に実行するコマンド
CMD [“-c”,“/etc/nginx/nginx.conf”]・・・*ENTRYPOINT*で使うコマンドオプション

LinuC

https://docs.docker.com/engine/reference/builder/

nginx.repo

```
[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
```

index.html

```
<!DOCTYPE html>
<link rel="stylesheet" href="main.css">
<html>
<head>
 <title>Sample HTML5 Page</title><!-- ここを編集すると、HTML文書のタイトルを変更できます -->
</head>
<body>
<p>Test Message</p><!-- ここを編集すると、表示される文字情報を変更できます -->
</body>
</html>
```

https://docs.docker.com/engine/reference/builder/

Dockerによるコンテナの操作

ZEUS enterprise

docker build（Dockerfileからのコンテナイメージ生成）

使い方：*docker build Dockerfile*があるディレクトリパス

-t：タグ名を指定する

```
[root@cc6516dd95d9 ~]# cat Dockerfile
FROM centos:latest
MAINTAINER SubMattNesk(kujiraitakahiro) <SMN@test.local>
COPY ./nginx.repo /etc/yum.repos.d/
RUN [“/bin/bash”,“-c”,“yum -y install nginx”]
EXPOSE 80
LABEL “version” = “0.02” ¥
"description" = "presented by SubMattNesk"
ADD ./index.html /usr/share/nginx/html/
ENTRYPOINT [“/usr/sbin/nginx”,“-g”,“daemon off;”]
CMD [“-c”,“/etc/nginx/nginx.conf”]

[root@54b9b9864066 ~]# ls
Dockerfile  index.html  nginx.repo  original-ks.cfg
```

LinuC

Dockerによるコンテナの操作

ZEUS enterprise

```
[ [root@54b9b9864066 ~]# docker build -t sample/nginx ./
Sending build context to Docker daemon  316.5MB
Step 1/9 : FROM centos:latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
 ---> 5d0da3dc9764
Step 2/9 : MAINTAINER SubMattNesk(kujiraitakahiro) <SMN@test.local>
 ---> Running in 86d8fccafb27
Removing intermediate container 86d8fccafb27
 ---> 328876b7c4d0
.
省略
.
Step 8/9 : ENTRYPOINT [“/usr/sbin/nginx”,“-g”,“daemon off;”]
 ---> Running in edc416928338
Removing intermediate container edc416928338
 ---> f682e9a6f5df
Step 9/9 : CMD [“-c”,“/etc/nginx/nginx.conf”]
 ---> Running in 7951c25c67e4
Removing intermediate container 7951c25c67e4
 ---> 2307099dcce3
Successfully built 2307099dcce3
Successfully tagged sample/nginx:latest

[root@54b9b9864066 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
sample/nginx        latest              2307099dcce3        6 seconds ago       343MB
centos              latest              5d0da3dc9764        2 weeks ago         231MB
```

生成されたコンテナイメージ

LinuC

Dockerによるコンテナの操作

docker commit（コンテナからコンテナイメージを生成）

使い方：*docker commit* コンテナ*ID* コンテナイメージ名*:*タグ名

```
[root@2414cef14360 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f8f4ffc8092c        2 days ago          133MB
[root@2414cef14360 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f8f4ffc8092c        2 days ago          133MB
[root@2414cef14360 ~]# docker run --name nginx_production -d -p 8080:80 nginx
97e02fefac1dc4d8a76178fd5ab863b1aabcbbea63bc53fa3252c1a227bf7abb
[root@2414cef14360 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
97e02fefac1d        nginx               "/docker-entrypoint.…"   4 seconds ago       Up 3 seconds        0.0.0.0:8080->80/tcp   nginx_production
[root@2414cef14360 ~]# docker commit nginx_production dev_nginx:version1.0
sha256:1937bcc85c93ac4b4ccf6374466114a049c787459fcdc5e5400258b6e68b83d3
[root@2414cef14360 ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
dev_nginx           version1.0          1937bcc85c93        6 seconds ago       133MB
nginx               latest              f8f4ffc8092c        2 days ago          133MB
```

生成されたコンテナイメージ

※生成したコンテナイメージは、*docker login*（*Dockerhub*へのログイン）・
docker push（*Dockerhub*へのアップロード）で共有・公開が可能

https://docs.docker.com/engine/reference/commandline/commit/

コンテナ操作デモ

コンテナ操作デモ

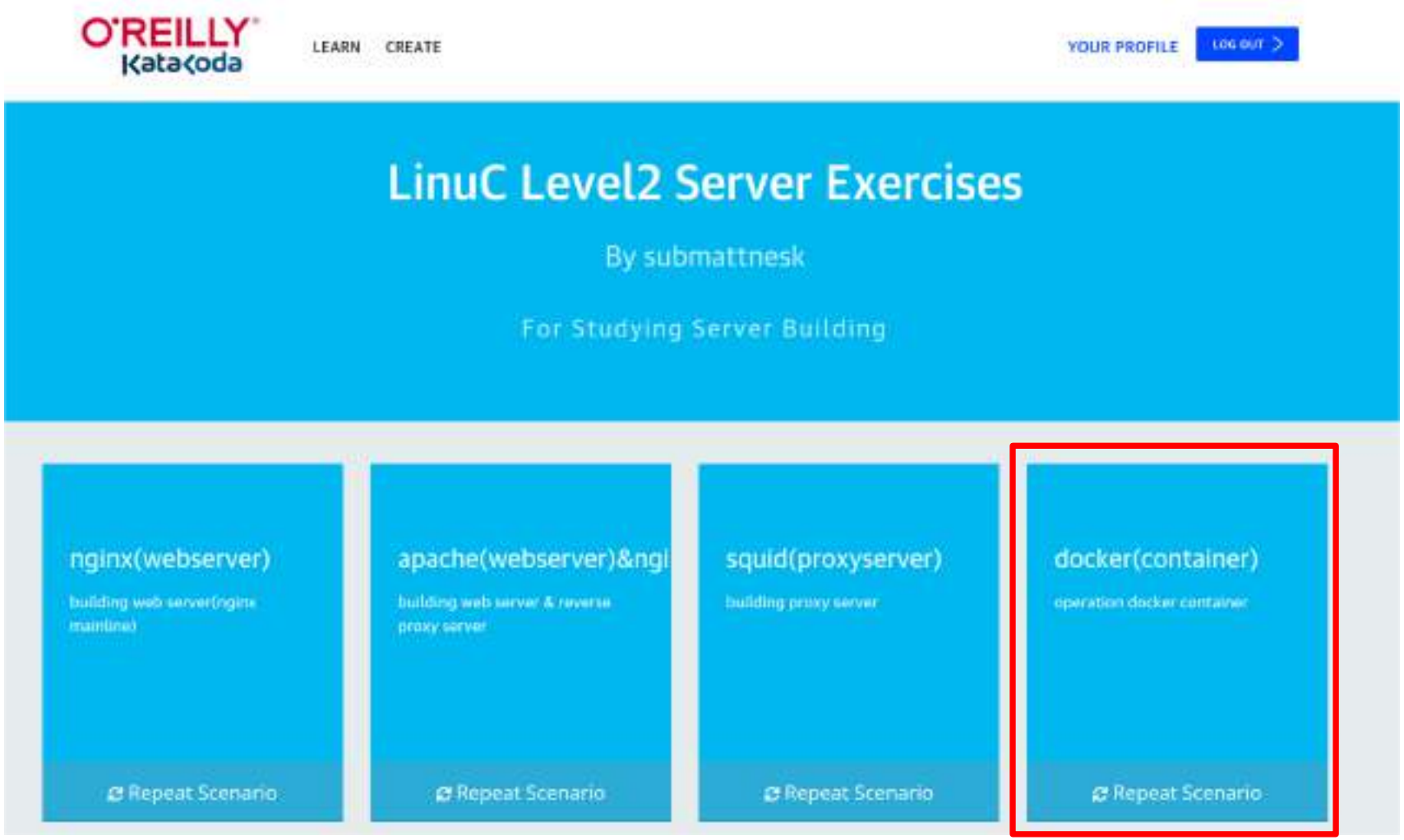
- PC(WindowsまたはMacなど)でWebブラウザから利用できます。

- ユーザ登録すれば利用可能です。
  Twitter、LinkedIn、Google、Githubのソーシャルアカウントを使えます。
  https://www.katacoda.com/submattnesk

コンテナ操作デモ

実際に触ってみてコンテナを学ぼう！

https://www.katacoda.com/submattnesk/courses/courses



Kubernetesとは

Kubernetesとは

Kubernetes (K8s)は、デプロイやスケーリングを自動化したり、コンテナ化されたアプリケーションを管理したりするための、オープンソースのシステムです。

管理や検出を容易にするため、アプリケーションを論理的な単位に分割し、コンテナをグルーピングします。KubernetesはGoogleでの15年にわたる経験を基に構築されており、コミュニティのアイディアや慣習との最善の組み合わせを取っています。

https://kubernetes.io/ja/

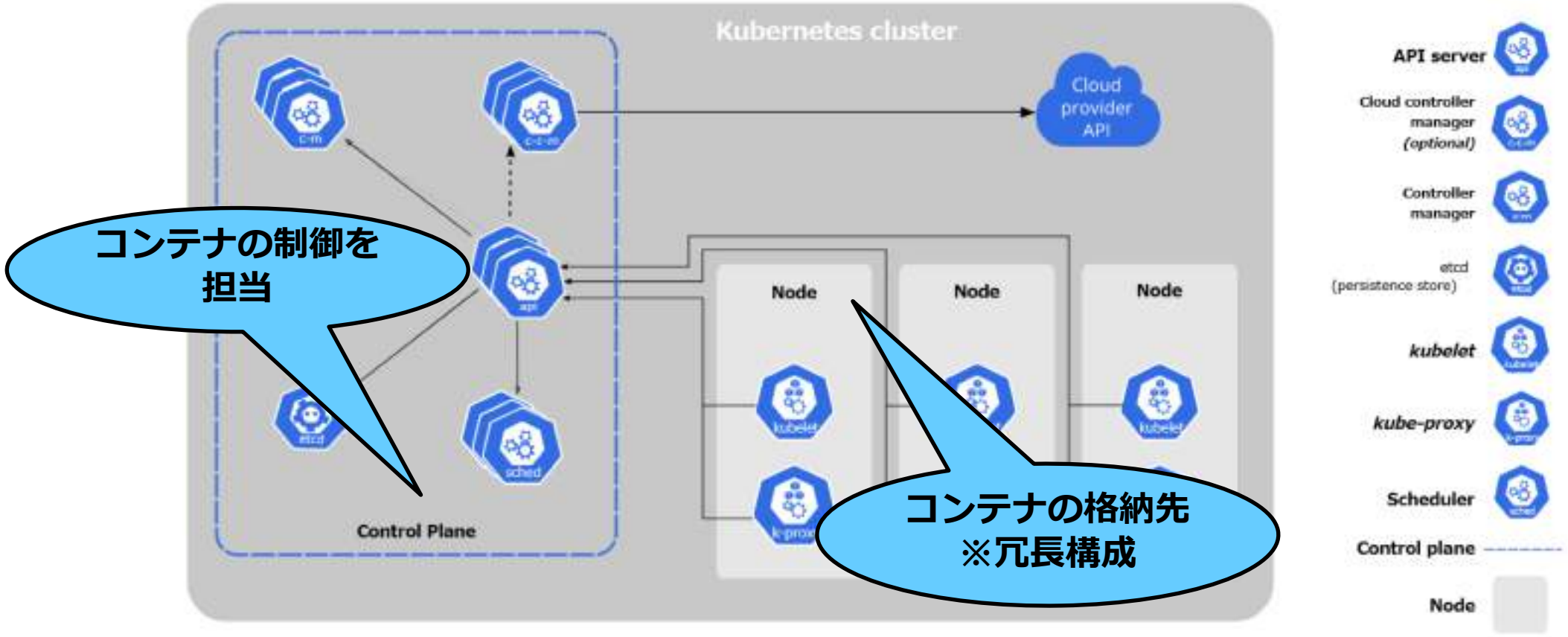
Kubernetesとは

Kubernetesが提供してくれるもの

- **サービスディスカバリーと負荷分散** Kubernetesは、DNS名または独自のIPアドレスを使ってコンテナを公開することができます。コンテナへのトラフィックが多い場合は、Kubernetesは負荷分散し、ネットワークトラフィックを振り分けることができるため、デプロイが安定します。
- **ストレージ オーケストレーション** Kubernetesは、ローカルストレージやパブリッククラウドプロバイダーなど、選択したストレージシステムを自動でマウントすることができます。
- **自動化されたロールアウトとロールバック** Kubernetesを使うとデプロイしたコンテナのあるべき状態を記述することができ、制御されたスピードで実際の状態をあるべき状態に変更することができます。例えば、アプリケーションのデプロイのために、新しいコンテナの作成や既存コンテナの削除、新しいコンテナにあらゆるリソースを適用する作業を、Kubernetesで自動化できます。
- **自動ビンパッキング** コンテナ化されたタスクを実行するノードのクラスターをKubernetesへ提供します。各コンテナがどれくらいCPUやメモリー(RAM)を必要とするのかをKubernetesに宣言することができます。Kubernetesはコンテナをノードにあわせて調整することができ、リソースを最大限に活用してくれます。
- **自己修復** Kubernetesは、処理が失敗したコンテナを再起動し、コンテナを入れ替え、定義したヘルスチェックに応答しないコンテナを強制終了します。処理の準備ができるまでは、クライアントに通知しません。
- **機密情報と構成管理** Kubernetesは、パスワードやOAuthトークン、SSHキーのよう機密の情報を保持し、管理することができます。機密情報をデプロイし、コンテナイメージを再作成することなくアプリケーションの構成情報を更新することができます。スタック構成の中で機密情報を晒してしまうこともありません。

https://kubernetes.io/ja/docs/concepts/overview/what-is-kubernetes/

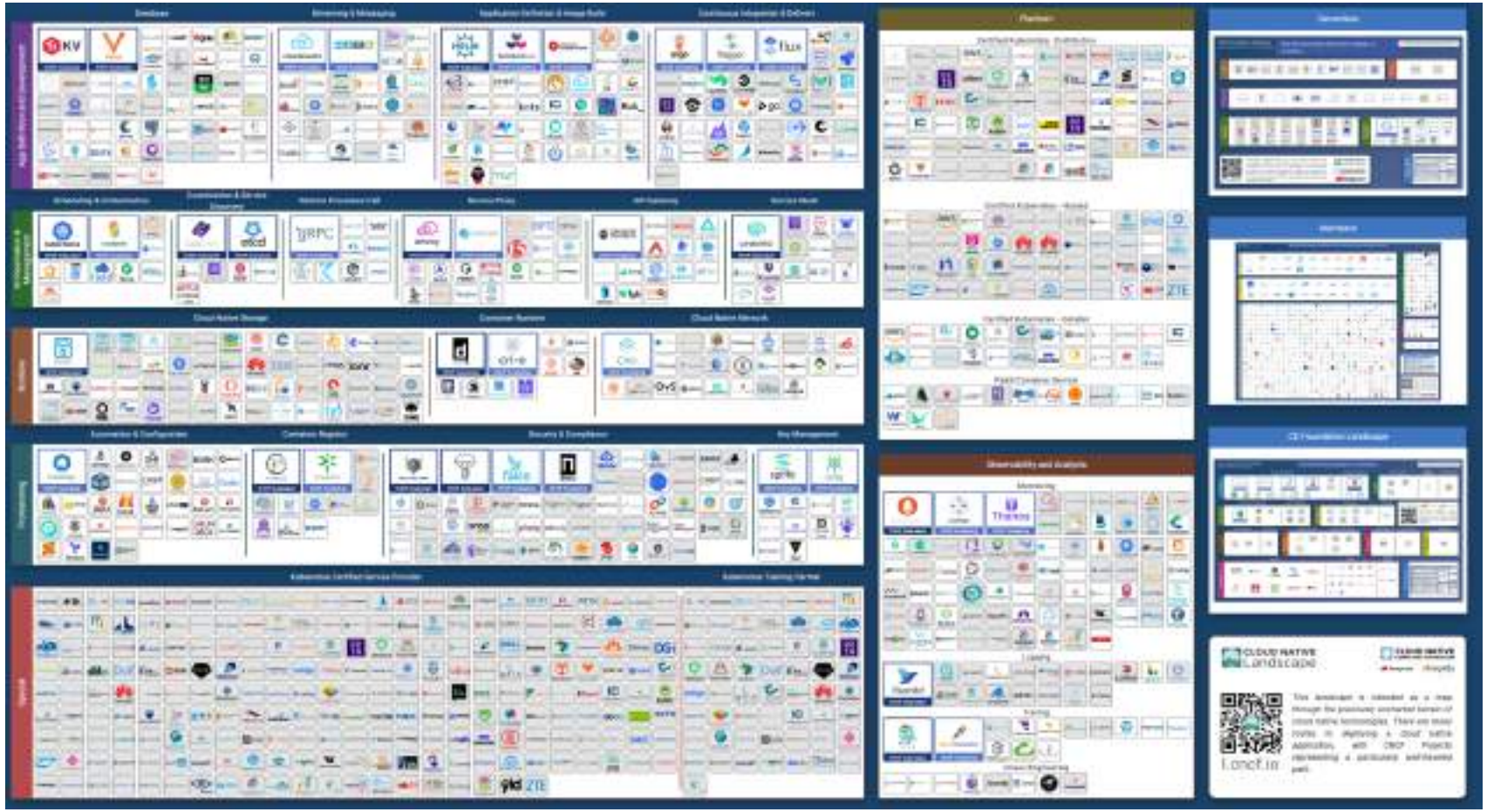
Kubernetesの動作環境



https://kubernetes.io/ja/docs/concepts/overview/components/

Kubernetesとは

CNCF(Cloud Native Computing Foundation)のlandscapeで
Kubernetesに関するプロジェクトを確認できる。



https://landscape.cncf.io/
https://www.cncf.io/

コンテナを起動してみる

```
ubuntu@kubeonubu:~$ kubectl run nginx001 --image=nginx:latest --port=80
pod/nginx001 created

ubuntu@kubeonubu:~$ kubectl get pods
NAME       READY   STATUS              RESTARTS   AGE
nginx001   0/1     ContainerCreating   0          5s

ubuntu@kubeonubu:~$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
nginx001   1/1     Running   0          7s
```

dockerでのコンテナ起動と同じような操作

コンテナを起動してみる Part2

```
ubuntu@kubeonubu:~$ cat nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21.3
        ports:
        - containerPort: 80
```

yamlファイルを使ったコンテナ起動
※3台のnginxを起動し、
自動復旧などしてくれる

```
ubuntu@kubeonubu:~$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created

ubuntu@kubeonubu:~$ kubectl get pods,deployments
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx001                        1/1     Running   0          15m
pod/nginx-deployment-db9778f94-dj4q9   1/1     Running   0          12m
pod/nginx-deployment-db9778f94-s5hbl   1/1     Running   0          12m
pod/nginx-deployment-db9778f94-7mlc5   1/1     Running   0          12m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           12m
```

Appendix

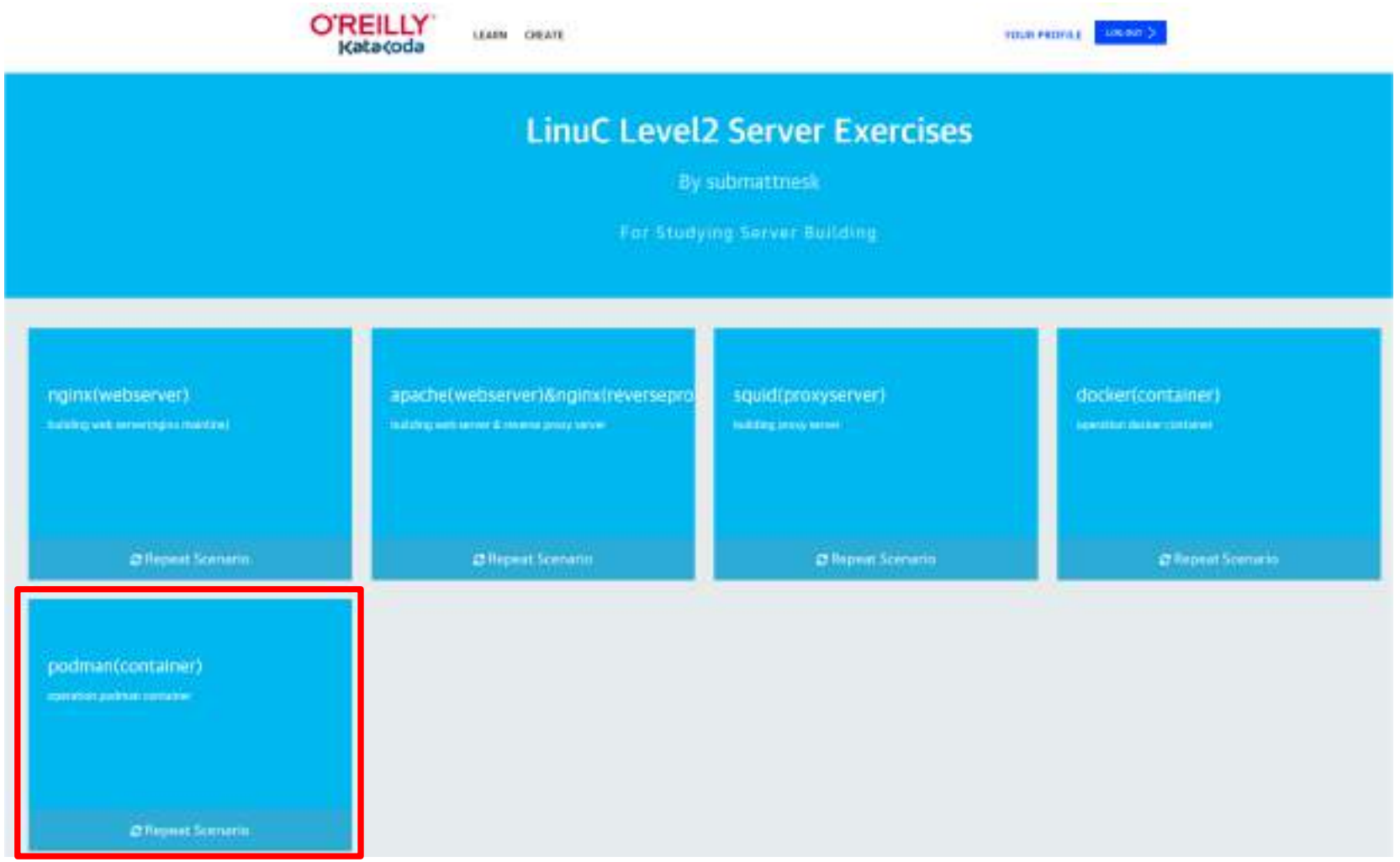
Podmanとは

Redhat社を中心としてコミニティで開発されているコンテナ管理ソフトウェア。
podmanコマンドを使う・内部アーキテクチャー・リリースサイクルなどに違いがある。

https://podman.io/

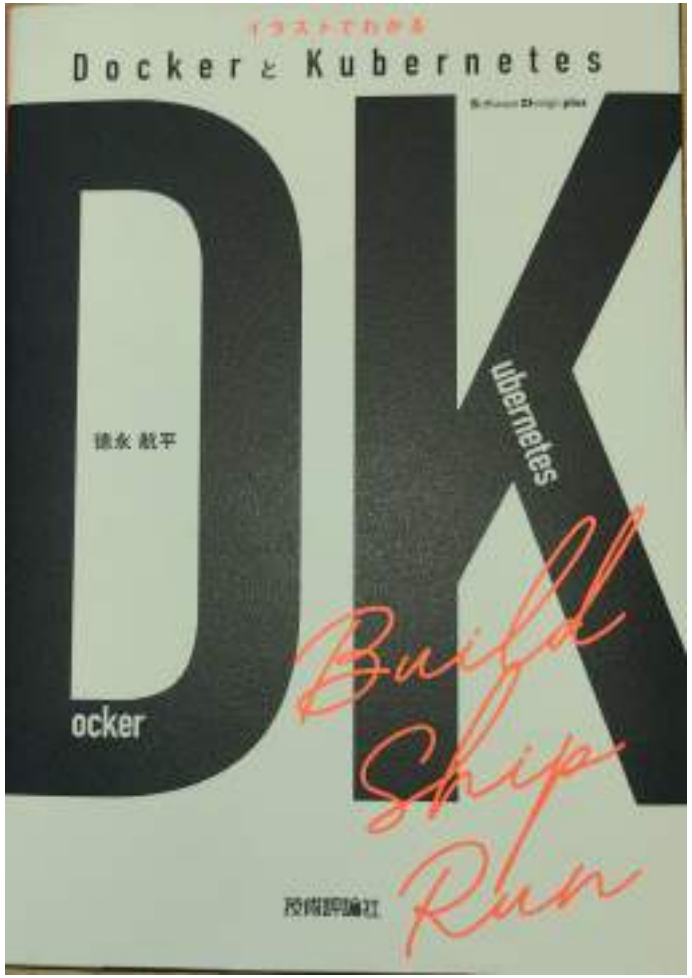
Podmanを使ってみよう！

https://www.katacoda.com/submattnesk/courses/courses



コンテナ関連でよかった書籍

イラストでわかるDockerとKubernetes
https://gihyo.jp/book/2020/978-4-297-11837-2



本日のまとめ

本日のまとめ

ZEUS enterprise

- オンプレ/仮想マシン/コンテナの違いを理解する
- コンテナ技術の概要を理解する
- Dockerの操作を理解する
- Kubernetesについて知る

“コンテナの良さ”を中心に話をしてきましたが、

コンテナを理解し、
物理マシン・仮想マシン・コンテナのどれが一番適切か
判断できることが大事！

LinuC

ZEUS enterprise

Q & A

LinuC

ZEUS enterprise

Thank you for join today's seminar!

https://www.opensourcetech.tokyo/

https://twitter.com/matt_zeus

https://www.facebook.com/takahiro.kujirai.1

LC LinuC