Linux研修【ハンズオン】
コンテナを体験してみよう！
／仮想マシン・コンテナの概念と利用

LinuC取得を目指す人を応援する情報サイト「リナスク」主宰

LinuCエバンジェリスト

水澤泰敬

LPI-JAPAN

自己紹介

水澤泰敬（ミズサワヤスタカ）

文系大学を卒業後、日経新聞関連企業、商社などへの転職を重ねたのち、フリーランスを経て有限会社を設立。10数年にわたり数々の情報まとめサイトの運営に携わる。

サイト運営時に取材したキュレーションサイトへの転職、事業譲渡によるTSUTAYA関連企業への転籍を経て再度フリーランスとなる。

2020年、日本のエンジニア育成に何らかのカタチで携わりたいと思い、エンジニアスクールで企業研修講師を務めながら約半年の準備期間を経てリナスクを立ち上げ。

趣味はギターとテレビゲーム。

リナスク

Linux技術者認定資格取得を
目指す人を応援する情報サイト

https://linuc.spa-miz.com/

本セミナーの内容

■主題・内容

ハンズオンを通じてコンテナ技術に実際に体験していただき、以下の主題の理解を深めます。
レベル１ ：1.01.2 仮想マシン・コンテナの概念と利用
レベル２　 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

■想定スキルレベル

・コンテナについて勉強したい
・Linuxについて勉強を始めたばかり（LinuCレベル１取得が目標）

■本セミナーのゴール

・コンテナの起動・再開・停止が行える（1.02.2の範囲）
・Dockerコンテナを実際に自分の手で操作してみる（2.06.2の一部）

本セミナーのサマリー

- LinuCの概要
- 仮想化とコンテナ技術の概要
- Dockerの概要
- Webブラウザで使えるクラウド環境の概要
- ハンズオン

ハンズオンの内容

- クラウド経由でLinuxの仮想環境を利用
- クラウド仮想環境にWebサーバーを構築
- Dockerコマンドでさまざまなコンテナを実行
- Dockerfileでカスタムイメージを作成
- Docker ComposeでWordpressサーバーを構築

質問については随時Q&Aで受け付けますが、時間と進行の都合上、セミナー中に全てお答えできるとは限りません。その場合は、後日公開される動画の中で回答させていただきますのであらかじめご了承ください。

■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技術者認定資格

✓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

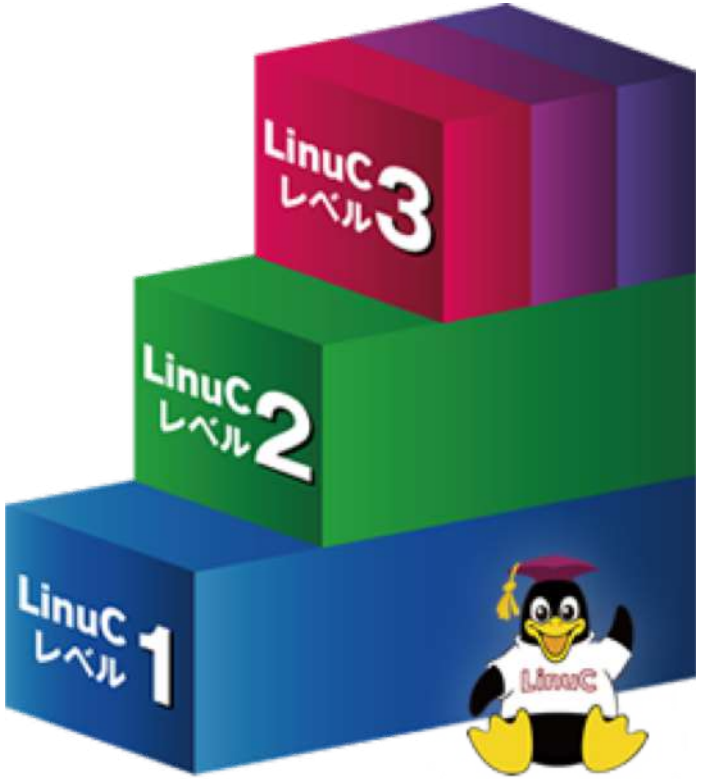
■Version10.0と従来の出題範囲の比較

	テーマ	Version 10.0	従来
LinuC-1	仮想技術	・仮想マシン／コンテナの概念 ・クラウドセキュリティの基礎	← （Version10.0で新設）
	オープンソースの文化	・オープンソースの定義や特徴 ・コミュニティやエコシステムへの貢献	← （Version10.0で新設）
	その他	→ （Version10.0で削除）	アクセシビリティ、ディスククォータ、プリンタの管理、SQLデータ管理、他
LinuC-2	仮想化技術	・仮想マシンの実行と管理（KVM） ・コンテナの仕組みとDockerの導入	← （Version10.0で新設）
	システムアーキテクチャ	・クラウドサービス上のシステム構成 ・高可用システム、スケーラビリティ、他	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	← （Version10.0で出題範囲に含む）
		→ （Version10.0で削除）	RAID、記憶装置へのアクセス方、FTPサーバーの保護、他

LinuCの3つのレベル

■順次ステップアップしていく認定構成

LinuC-1 → LinuC-2 → LinuC-3 (M S V)



■上位レベルに認定されるためには、
下位レベルの認定が必須

LinuC認定資格を取得したら

■LinuC試験に合格して認定資格の認定を取得すると、LPI-Japanから認定証と認定カードが送られてきます。認定者の証明としての認定ロゴが名刺などに使用できます。

LinuC-1

LinuC-1

LinuC-1
Certified
Professional

LinuC-1
Certified Professional

LinuCを学習することの意味

「LinuCレベル1 / レベル2」の認定を取得できるIT技術者であれば、どのような分野のIT技術者にもつながる技術を持っていることを証明できる

LinuCを学習することで、IT技術者として身に付けておきたい重要かつ基礎的なスキルを獲得できる

Linux/LinuCの学習環境

実機

- Windows PCなどにLinux OSを直接導入
- 外付HDD/SDDに導入して外付から起動すればWindowsを消さなくて済む

仮想環境

- パソコンのホストOSはそのままにLinux仮想環境を導入
- Virtual BoxやVagrantなどを利用

クラウド型環境

- AWSやGCPの仮想マシンでLinux環境を構築（無料枠もあるが基本的には有料）
- Weブラウザで使えるクラウド型学習環境サービス（無料）

LinuCを取得するなら、実機、仮想、クラウド、それぞれを可能な限り試してみよう

仮想化

■仮想化は、コンピューターをはじめとした種々の物理的なハードウェア環境を抽象化しソフトウェア化すること

- 1台のサーバーに複数の仮想サーバーを構築する
- 複数のサーバーを1台のサーバーのように見せる

■コンピューティング環境では、仮想化を行うため「仮想マシン」と「コンテナ」技術が使われている

■ストレージ、クライアント環境、ネットワークなども仮想化できる

（LinuC Version10.0 新出題範囲学習補助教材より）

仮想マシン

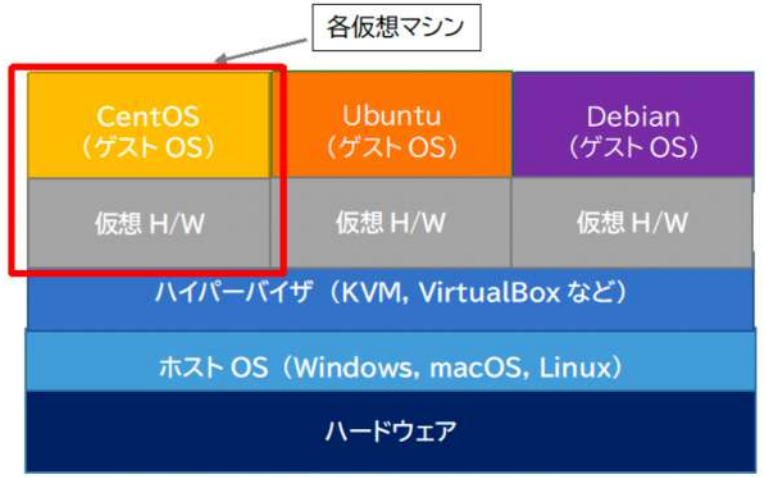
■仮想マシンとは

- 物理的なサーバーと同等の機能・動作を再現した仮想的なハードウェア環境
- ハイパーバイザーと呼ばれるソフトウェアで実現
- ハイパーバイザーはハードウェア上で直接動作する「Type1(ネイティブ型)」とハードウェアのホストOS上で稼働するアプリケーションが仮想マシンを再現する「Type2(ホスト型)」の2種類がある

Type1：KVM 、VMware ESX/ESXi、Hyper-V、Xenなど
Type2：VMware Workstation Player、VirtualBox、Microsoft Virtual Serverなど

■仮想マシンの特徴

- それぞれの仮想マシンには個別にOS(ゲストOS)、ライブラリ(実行環境)、ミドルウェア、アプリケーションをインストールして独立したコンピューティング環境が構築できる
- ハードウェアをエミュレートするためCPUやメモリのリソースを多く消費する
- 仮想マシンはホストOSとは独立しているため実行環境が変わってもそのまま動作させることができる
- ホストOSとは異なる種類のOSを仮想マシン上で実行することが可能



（LinuC Version10.0 新出題範囲学習補助教材より）

コンテナ

■コンテナの概念とその特徴

(レベル1補助教材)

- 物理的なコンピューター上で稼働する**OS(ホストOS)のリソースの一部を隔離し、仮想的に作り出された実行環境**。chrootの機能を拡張したもの。
  ↓
  ルートディレクトリを変更してコマンドを実行する
  ＝アプリケーションによるアクセスを指定したディレクトリに限定する
- ホストOSとは**隔離されたプロセスを作成**し、その上にライブラリ(実行環境)、ミドルウェア、アプリケーションをインストールして、**独立したコンピューティング環境を構築**する。
- ホストOSのプロセスとして動作するので、CPUやメモリのリソース消費は少なく、起動するまでの時間がかからない。
- コンテナ管理ソフトウェアがハードウェアやOSごとの違いを吸収するため、他の実行環境へコンテナを容易に移動・配布できる。

(レベル2補助教材)

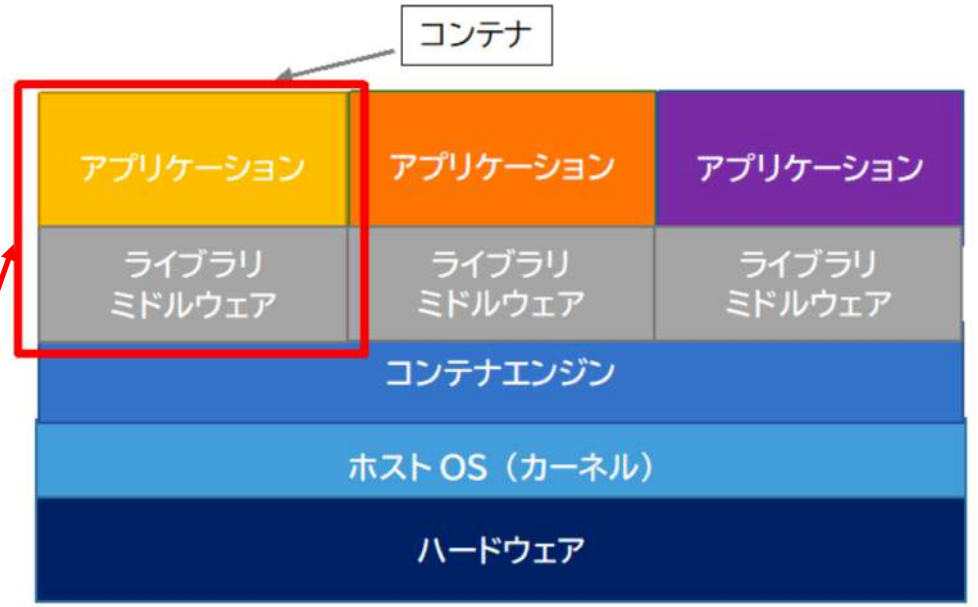
- ホスト上に区画化された空間を実現することが「コンテナ型仮想化」、実現方法が「コンテナ技術」。
- アプリケーションだけでなく動作に必要となるライブラリや設定ファイルなど実行環境をまとめたもの。
- コンテナエンジンがホストOSに常駐しコンテナの操作・管理を行う。
- 各コンテナはコンテナエンジンが動作しているホストOSのカーネルを共有している。
- 各コンテナの機能を構成するプロセスをプロセス群として扱うことができコンテナ間で互いのプロセスは参照できない。

（LinuC Version10.0 新出題範囲学習補助教材より）

コンテナ

端的に言えば

- 単一のホストOS上における
- **複数の**
- **独立した**
- **アプリケーション実行環境**



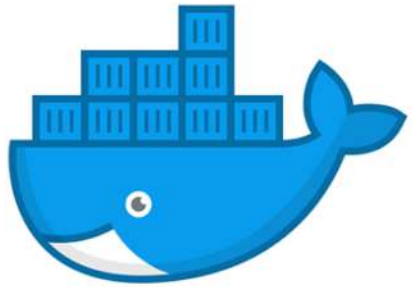
アプリケーション単位の仮想化 → **代表的なコンテナ技術がDocker**

仮想マシンはハードウェア単位の仮想化

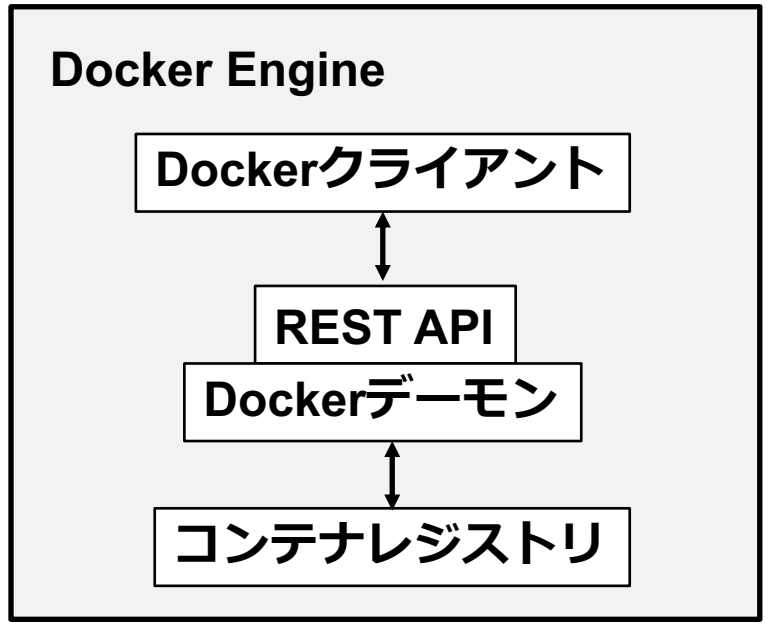
（図はLinuC Version10.0 新出題範囲学習補助教材より）

Dockerとは

各コンテナは、コンテナの動作プラットフォーム(**Docker Engine**など、右のロゴでいうと鯨が担う役割)上で動作する。



Docker Engineは、コンテナやイメージを管理するためのアプリケーション。DockerクライアントからDockerデーモンのAPIにアクセスすることでコンテナに関するさまざまな操作が行える。



Docker Engineには、有償のEnterprise Edition(EE)と無償のCommunity Edition(CE)の2系統がある。

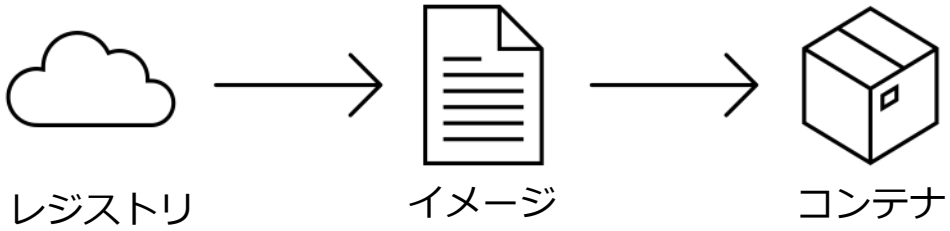
Dockerを使いこなすためにはシステム構成を考え、それに適した構成定義ファイル(**Dockerfile**)などを作成するスキルが必要。

Kubernetesなどコンテナオーケストレーション技術を用いると、複数コンテナを連動させて可用性の高いシステムを短時間に構築し稼働させることができる。

（主にLinuC Version10.0 新出題範囲学習補助教材より）

コンテナイメージとDocker Hub

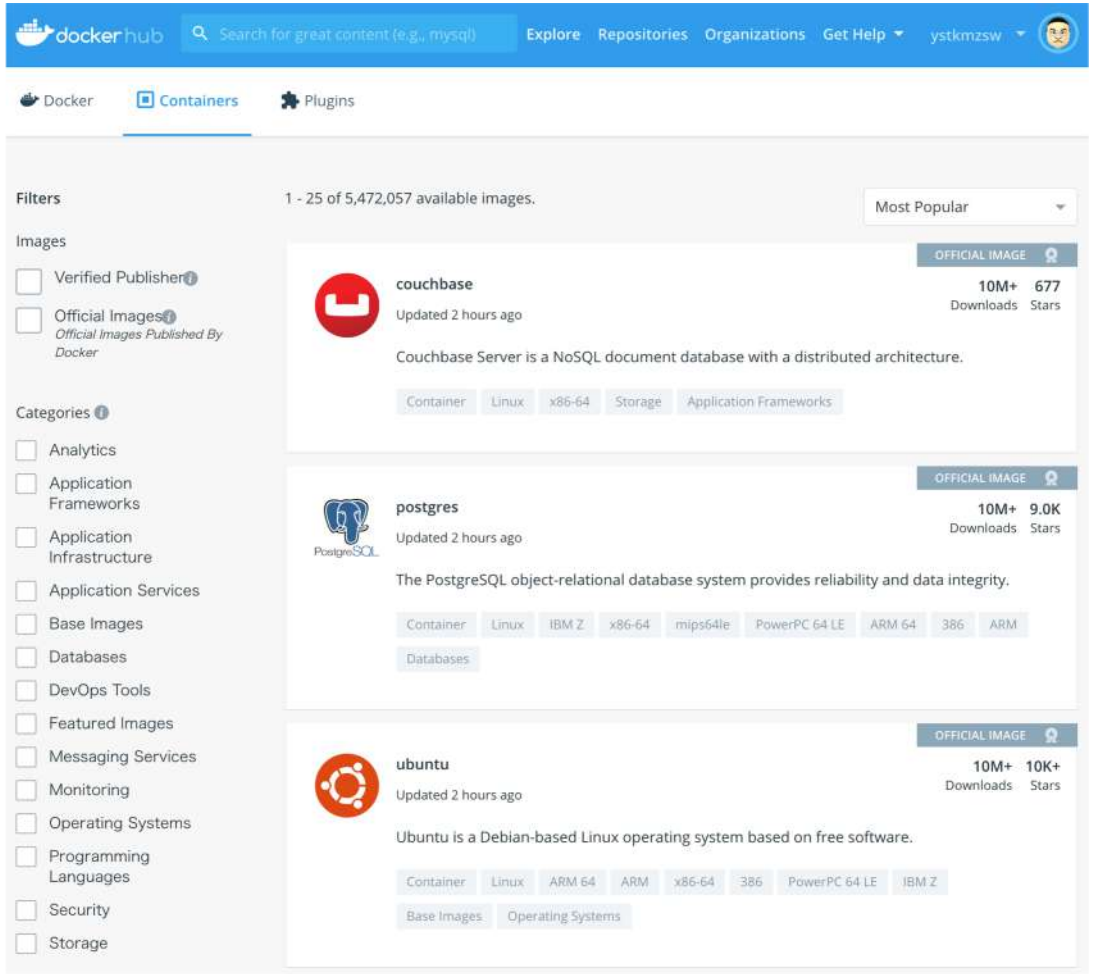
各コンテナは**イメージ**と呼ばれるコンテナのテンプレートファイルから生成される。



ユーザーは独自にコンテナに含めるコンポーネントを定義してイメージを作成することもできるし、**Docker Hub**などのレジストリなどで公開されている既成のイメージを取り込んでコンテナを生成することもできる。

Docker HubではcentosやubuntuなどのOSのほか、mysqlなどのDB, httpdなどのWebアプリといった、各種アプリケーションやミドルウェアを含むイメージなどが公開されている。

https://hub.docker.com/



（主にLinuC Version10.0 新出題範囲学習補助教材より）

コンテナのライフサイクル

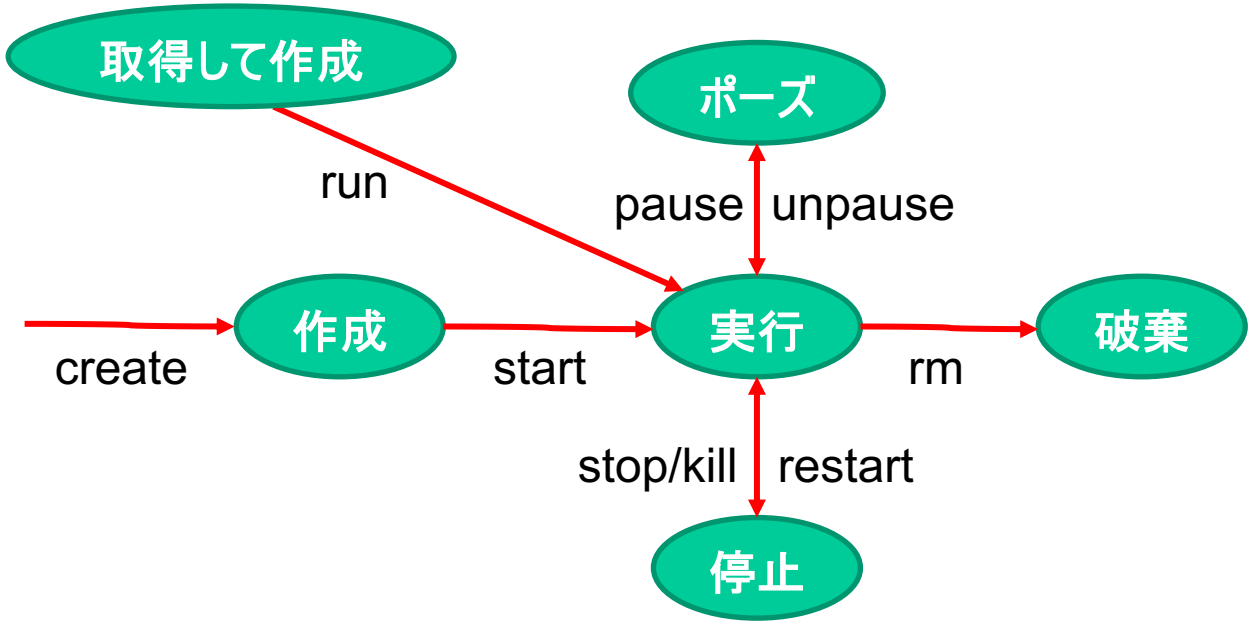
コンテナのライフサイクルは、

作成　→　実行　→　(一時)停止　→　破棄

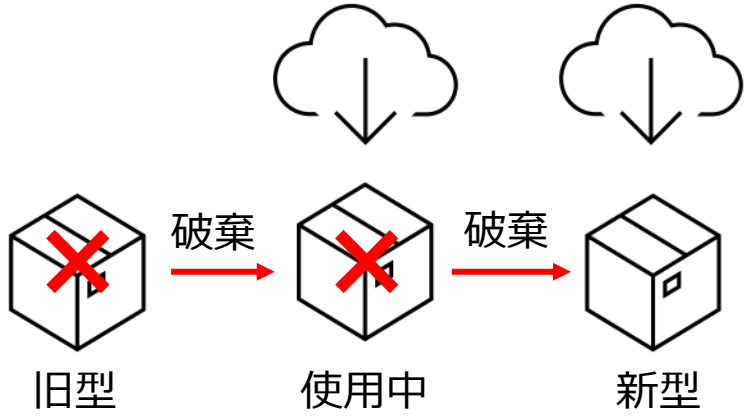
という一連の流れ。

コマンドで状態が変化する。

- 作成：docker create
- 起動：docker start
- 停止：docker stop
- 破棄：docker rm



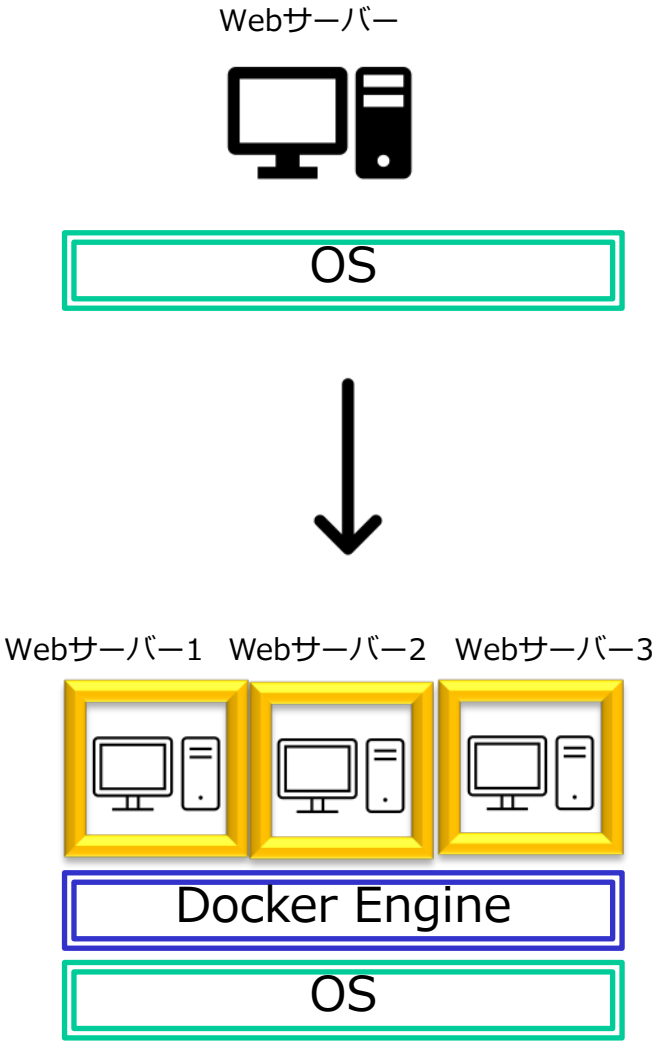
1つのコンテナをアップデートしながら使うのではなく、アップデートされた新しいコンテナを使う（＝次から次へとコンテナを乗り換える）。アップデートされたコンテナに乗り換える方が時間的にも速く間違いがない。



Dockerを使うメリットとデメリット

メリット

- 1台のホストOSに複数のサーバー(アプリケーション)を構築できる（同じ種類のサーバーの複数構築も可能）
- 動作が軽く、データ容量も軽量
- 互いに隔離されている＝安全で管理もしやすい
- 作成、複製、破棄、アップデート、入れ替えなどが容易
- カスタマイズしたコンテナをイメージ化して配布できる
- 開発環境や運用環境といった異なる環境で同じコンテナを再現できる（可搬性に優れる）



デメリット

- 物理マシンに問題が起こると構築したコンテナ全てに影響が出る
- Linuxとコンテナエンジン(Docker Engine)が必須

Dockerコマンドの基本

■基本的な記述方法

 # docker　コマンド(上位・副)　オプション (オプションの引数)　対象　引数

- 例：最新版のubuntuコンテナイメージを取得してubu1という名前で起動しbashターミナルを実行

オプション1　　対象

docker container run -it --name ubu1 ubuntu /bin/bash

上位コマンド　副コマンド　オプション2(引数あり)　引数

旧体系では省略

- **コマンド**
  新体系は上位コマンドと副コマンドの組合せ
  　上位コマンド：container, image, volume, networkなど
  　副コマンド：run, start, stop, ls, exec, commit, buildなど
  旧体系は新体系でいう副コマンドが上位コマンド
  　run, images, start, stop, ps, exec, rm, rmi, pullなど
- **オプション**
  　-d, -i, -t, -p, --nameなど
- **対象**
  　イメージやコンテナを指定
- **引数**
  　コマンドなど

主なDockerコマンド

主コマンド	上位コマンド	副コマンド	主なオプション	内容	旧体系(v.1.21以前)
docker	container	run	-d -i -t -p -v -e --name --rm	リポジトリからイメージをダウンロードしてコンテナを生成し起動する	docker run
			-d	コンソールを離してバックグラウンドで起動する（--detach）	
			-i	キーボードからの入力を標準入力としてシェルに伝える（--interactive）	
			-t	シェルのプロンプト表示を有効にする（--tty）	
			-p [ホストのポート番号:コンテナのポート番号]	ポート番号を指定する（--publish）	
			-v [ホストのディスク:コンテナのディレクトリ]	ボリュームをマウントする（--volume）	
			-e 環境変数=値	環境変数を指定する	
			--name [コンテナ名]	コンテナに名称を付与する	
			--rm	コンテナが停止されたらコンテナを即時破棄する	
		start	-i	コンテナを起動する	docker start
		restart		コンテナを再起動する	docker restart
		stop		コンテナを停止する	docker stop
		pause		コンテナを一時停止する	docker pause
		unpause		コンテナの一時停止を解除する	docker unpause
		kill		コンテナを強制終了する	docker kill
		attach		実行中のコンテナに接続する	docker attach
		create	-e -p -v --name	イメージからコンテナを作成する	docker create
		commit		コンテナをイメージに変換する	docker commit
		ls	-a	コンテナの一覧を表示する	docker ps
		rm	-f -v	停止済みのコンテナを削除する	docker rm
		prune		停止済みのコンテナをまとめて削除する	
		exec	-i -t	実行中のコンテナでプログラム（コマンド）を実行する	docker exec
		cp		コンテナとホスト間でファイルをコピーする	docker cp
		logs		コンテナのログを参照する	docker logs
		inspect		コンテナの詳細情報を表示する	docker inspect
		stats		コンテナのステータスを表示する	docker stats
docker	image	pull		リポジトリからイメージをダウンロードする	docker pull
		push		リポジトリにイメージをアップロードする	docker push
		tag		イメージにタグを付与する	docker tag
		ls		イメージの一覧を表示する	docker images
		rm		イメージを削除する	docker rmi
		build	-t	イメージを作成する	docker build
		inspect		イメージの詳細情報を表示する	

出題範囲の確認と補助教材で取りあげられているDockerコマンド

1.01.2 仮想マシン・コンテナの概念と利用（重要度：4）
- コンテナの起動と停止ができる。

出題範囲に具体的なコマンドの記述は無いが、レベル1ver.10
補助教材には下記のコマンドが掲載されている

2.06.2 Dockerコンテナとコンテナイメージの管理（重要度：3）
- 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。

■レベル1

- docker pull
- docker images
- docker run
- docker ps (–a)
- docker start
- docker stop
- docker restart
- docker pause
- docker unpause
- docker kill
- docker attach
- docker exec

■レベル2 (レベル1の内容に加えて)

- docker network create
- docker network connect
- docker stats
- docker tag
- docker push
- docker rm
- docker rmi
- docker import
- docker commit
  - 出題範囲に本ページのコマンド全てが掲載されている。

Dockerを使う方法①

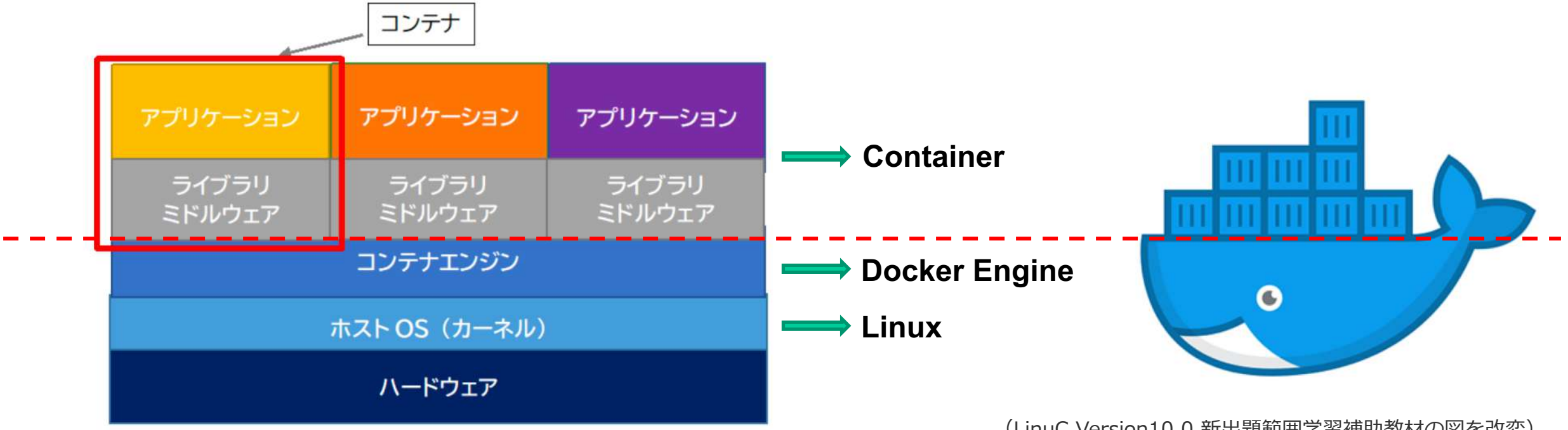
Linux実機にDockerを導入

- **ディストリビューション版のインストール**(例：CentOSの場合)

```
# yum install docker
```

- **公式CE版のインストール**(例：CentOSの場合、新出題範囲学習補助教材より)

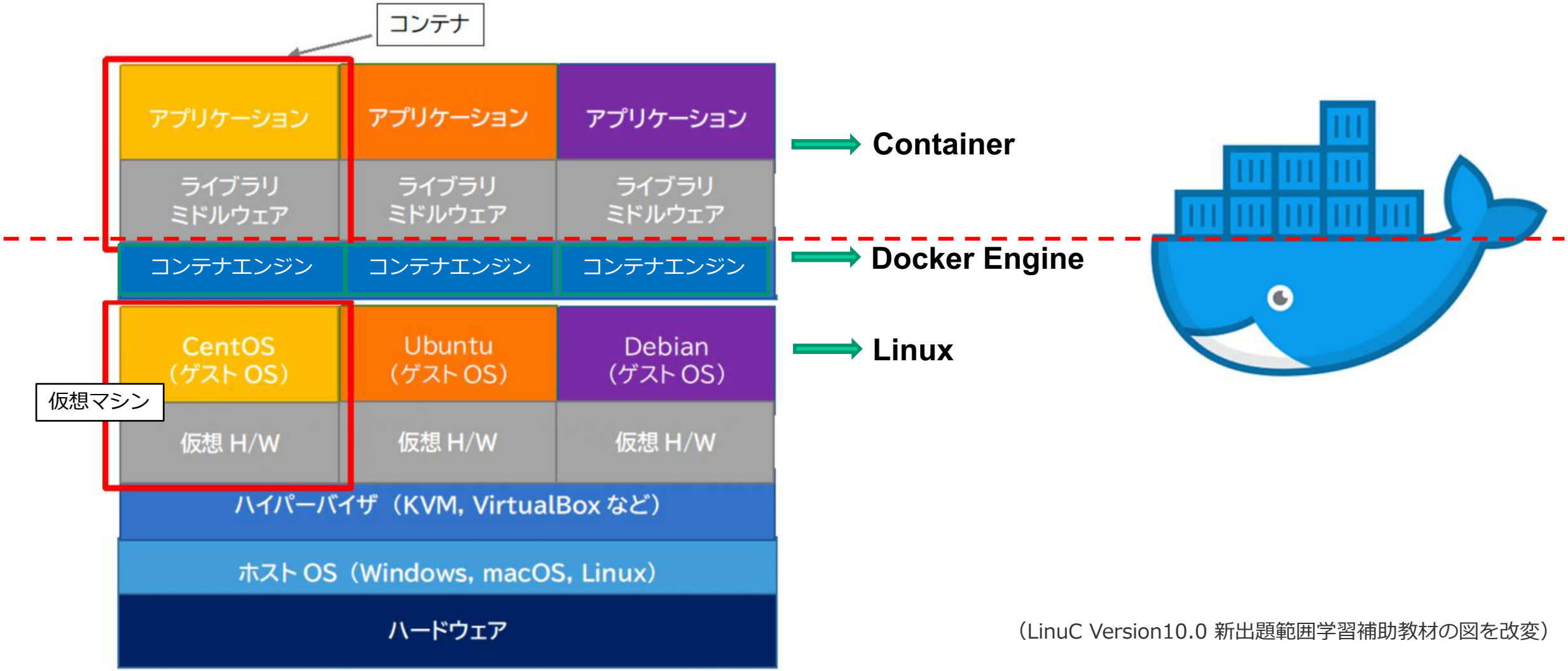
```
# yum install -y yum-utils device-mapper-persistent-data lvm2
# yum-config-manager --add-repo  https://download.docker.com/linux/centos/docker-ce.repo
# yum install docker-ce docker-ce-cli containerd.io
```



（LinuC Version10.0 新出題範囲学習補助教材の図を改変）

Dockerを使う方法②

■ ホスト型仮想マシン(VMware/VirtualBox+Linux OS)にDockerを導入

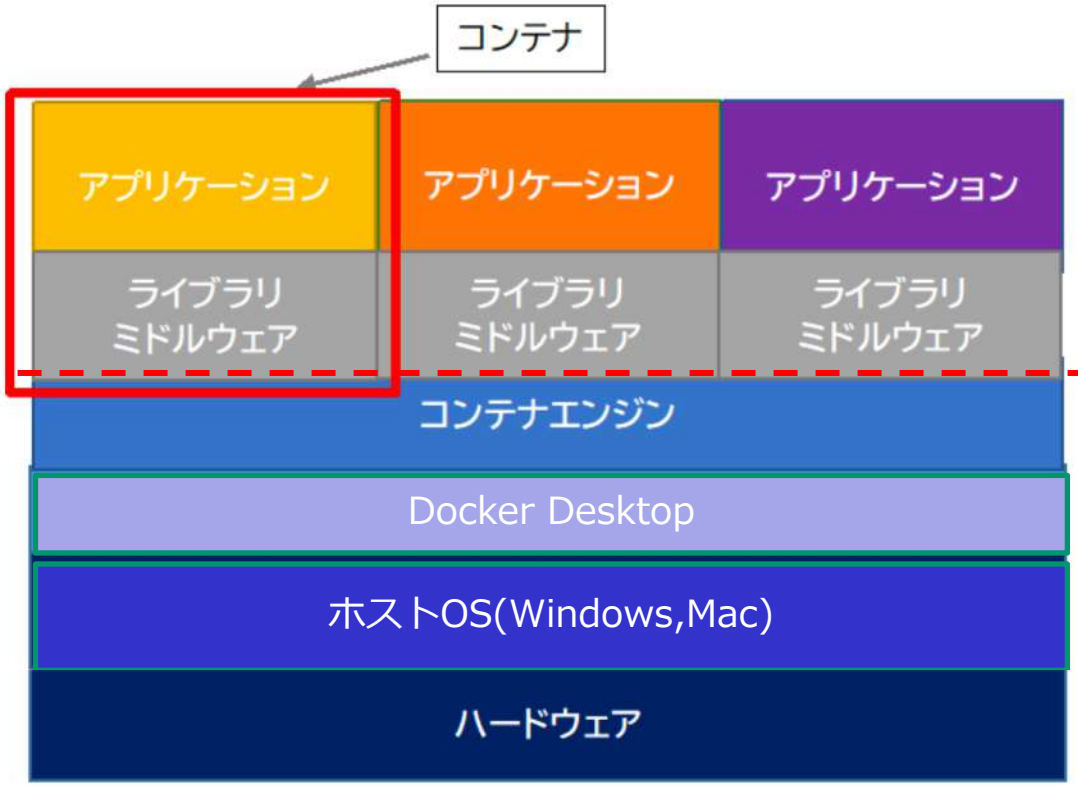


（LinuC Version10.0 新出題範囲学習補助教材の図を改変）

Dockerを使う方法③

■MacやWindowsにDocker Desktopを導入

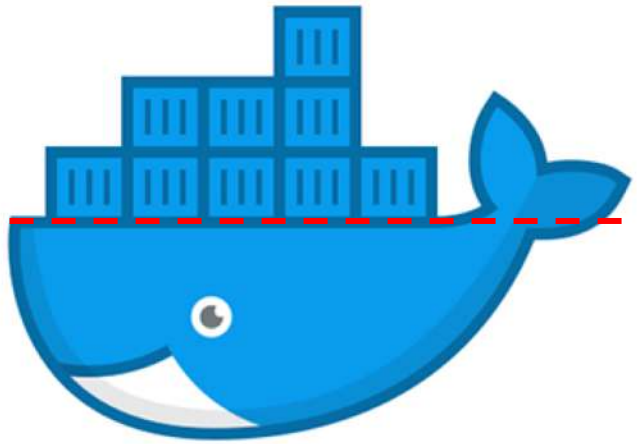
- https://www.docker.com/products/docker-desktop



Container

Docker Engine

Linux



MacはHyperKitがDocker Desktopに含まれている。
WindowsのDocker Desktopには仮想環境は付属せず、
WindowsのHyper-VやWSL2を使う。
Hyper-VはWindows Pro以上で利用できるが、
WSL2はHomeエディションでも使える。

（LinuC Version10.0 新出題範囲学習補助教材の図を改変）

Dockerを使う方法④

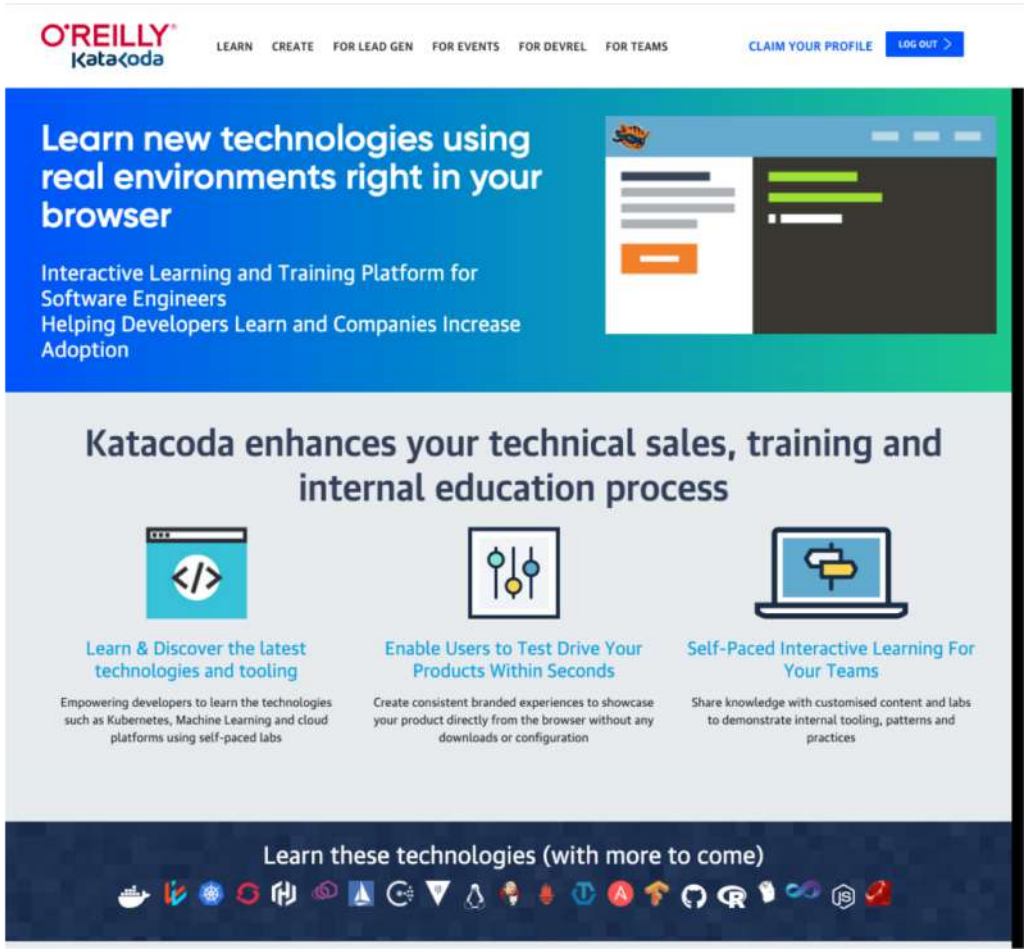
■Dockerが使えるクラウドサービスを利用

- AWSやGCPなどのクラウドサービス（一部無料枠もあるが基本的には有料）
- **Play with Docker（無料）**：https://www.docker.com/play-with-docker
  Webブラウザ経由で使えるDocker実行環境。利用にはDocker IDが必要。4時間の利用制限。サーバーの負荷からか接続できない時がある。

- **Katacoda（無料）**：https://www.katacoda.com/ ← ハンズオンで使用
  Webブラウザベースのオンライン学習サービス。Linuxインスタンスにはディストリビューション版が導入済み。比較的安定して利用できる（これまでの経験上）。

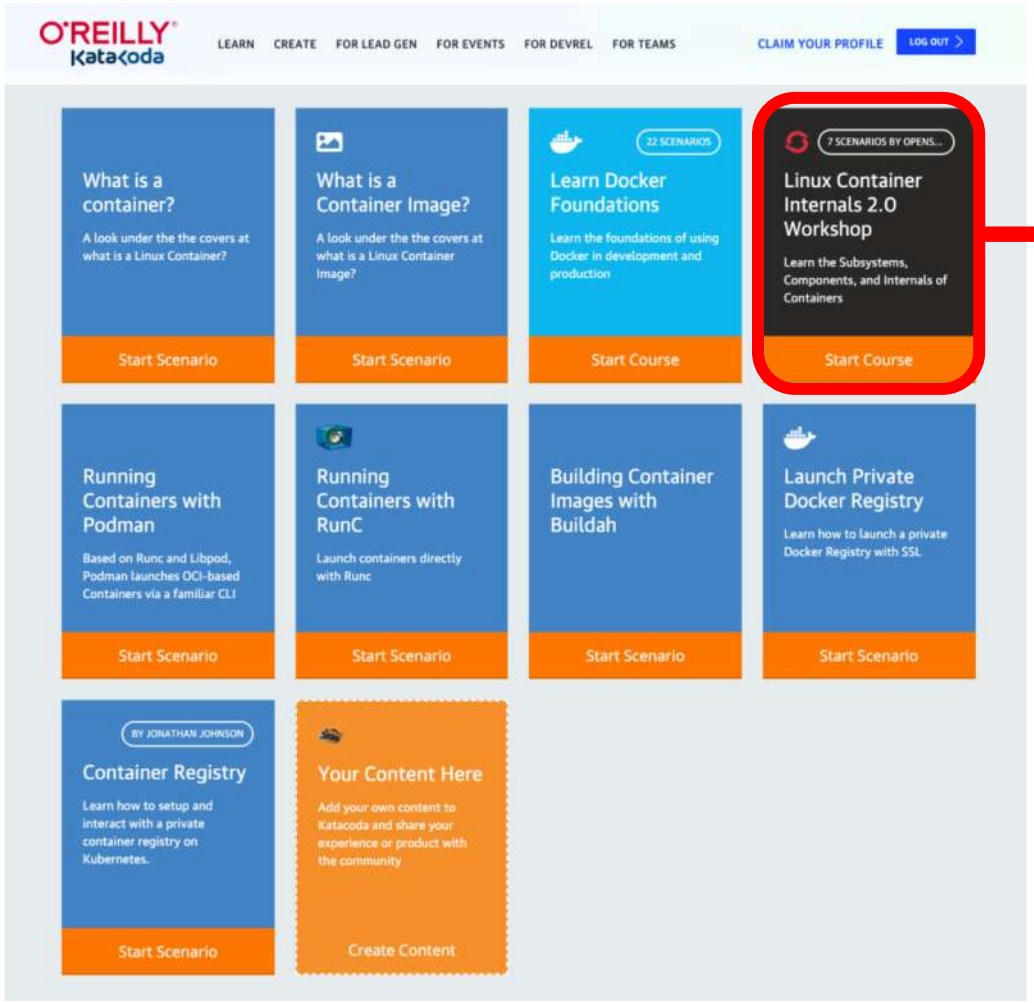
クラウド型学習環境　Katacodaとは

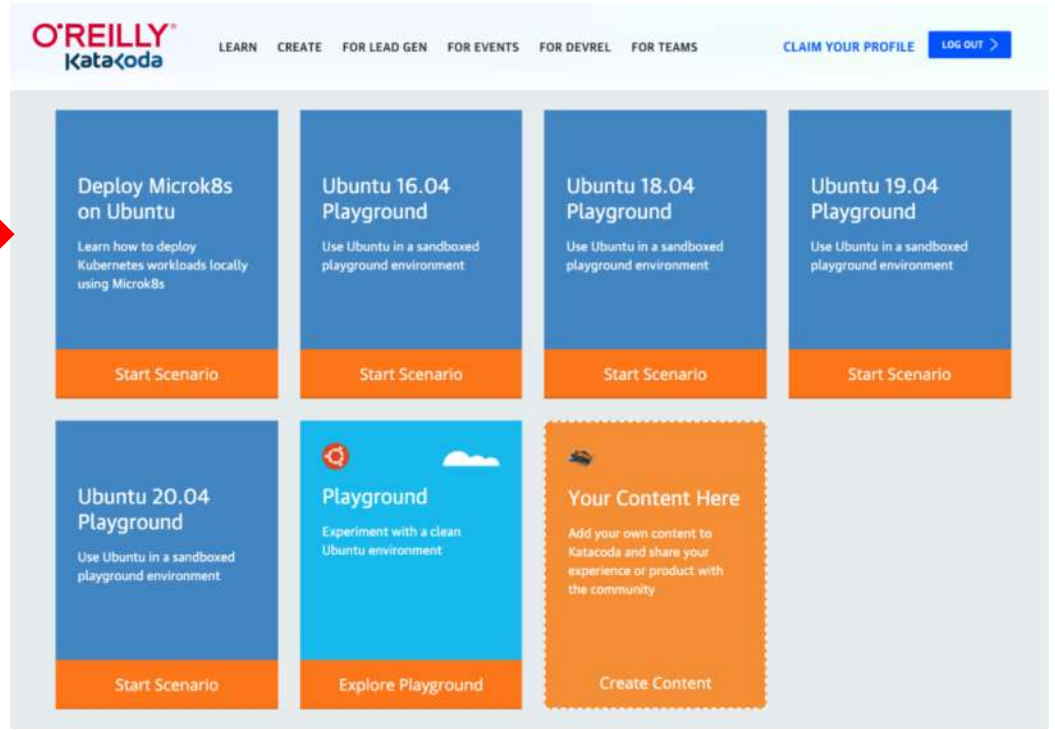
- コンピュータ関連書籍を出版している米オライリー社が運営するWebブラウザベースのオンライン学習サービス
- テーマ毎に様々なシナリオが用意されており、トータルで300を超えるシナリオを無料提供
- サービス自体が仮想化技術で構築されている



https://www.katacoda.com/

Katacodaで使えるLinux環境



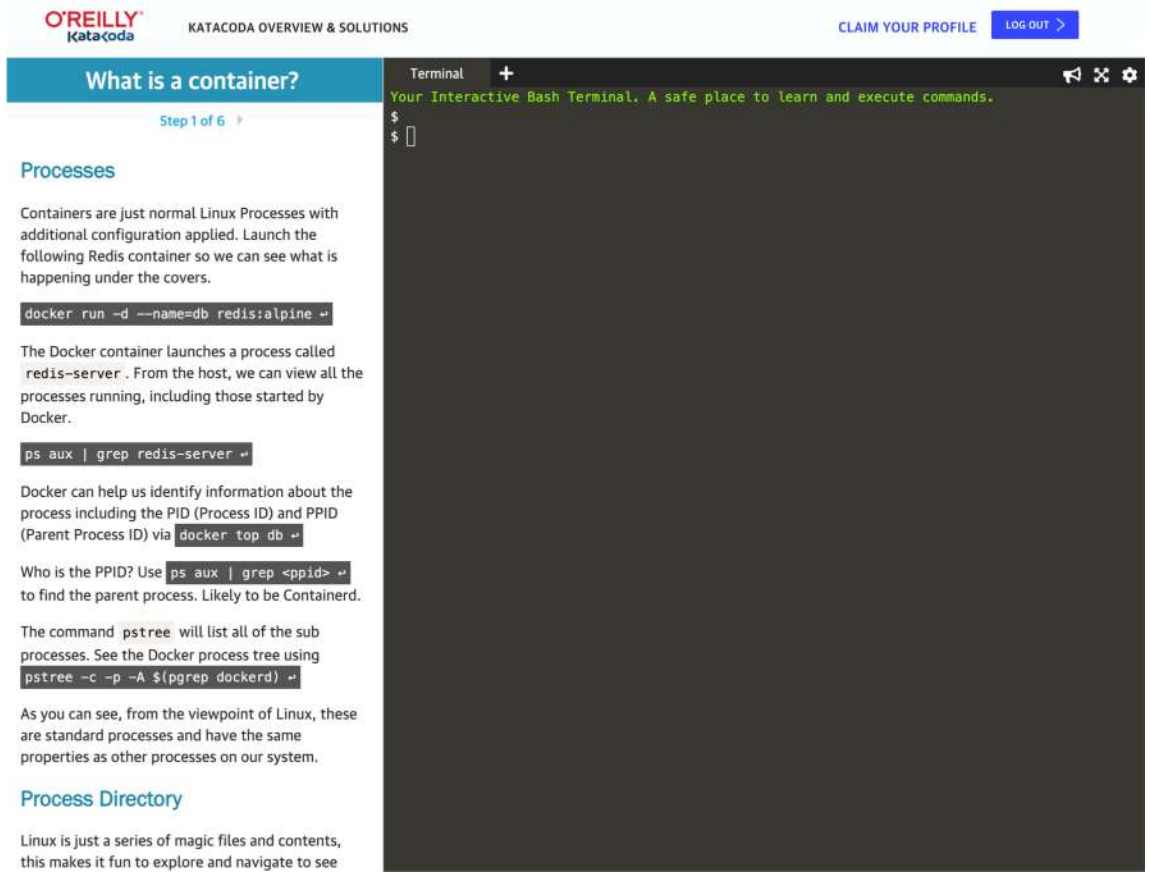


- Ubuntu：16.04, 18.04, 19.04, 20.04
- CentOS：7
- Fedora： 30

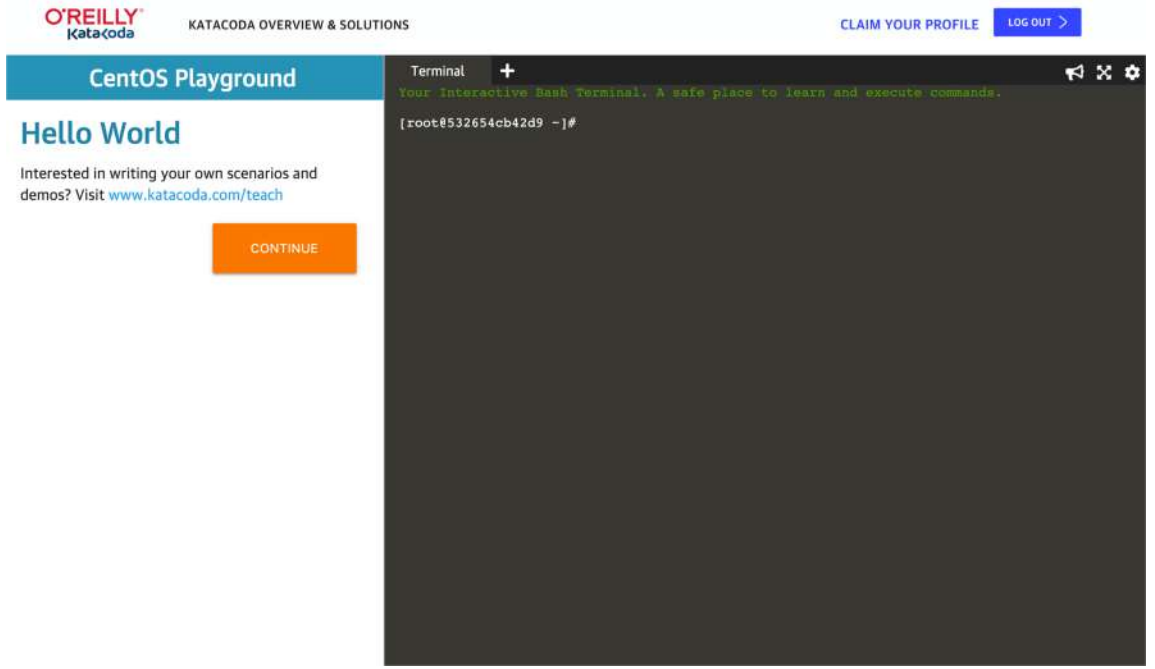
（2021/3現在）

Katacodaの基本画面

学習用インスタンスは画面左の解説を読みながら、画面右のコンソールで操作します。



Linuxプレイグラウンドは解説がありません。画面右のコンソールで自分の好きなように操作します。



Katacodaを使うメリットとデメリット

メリット

- CentOSとUbuntu両方ともに自由度の高いインスタンスが利用可能
- Linuxインスタンスにあらかじめdockerが導入済み
- コンテナ技術の学習コンテンツが豊富に用意されている
- 無料（2021/3現在）

LinuC バージョン10.0 の学習に最適

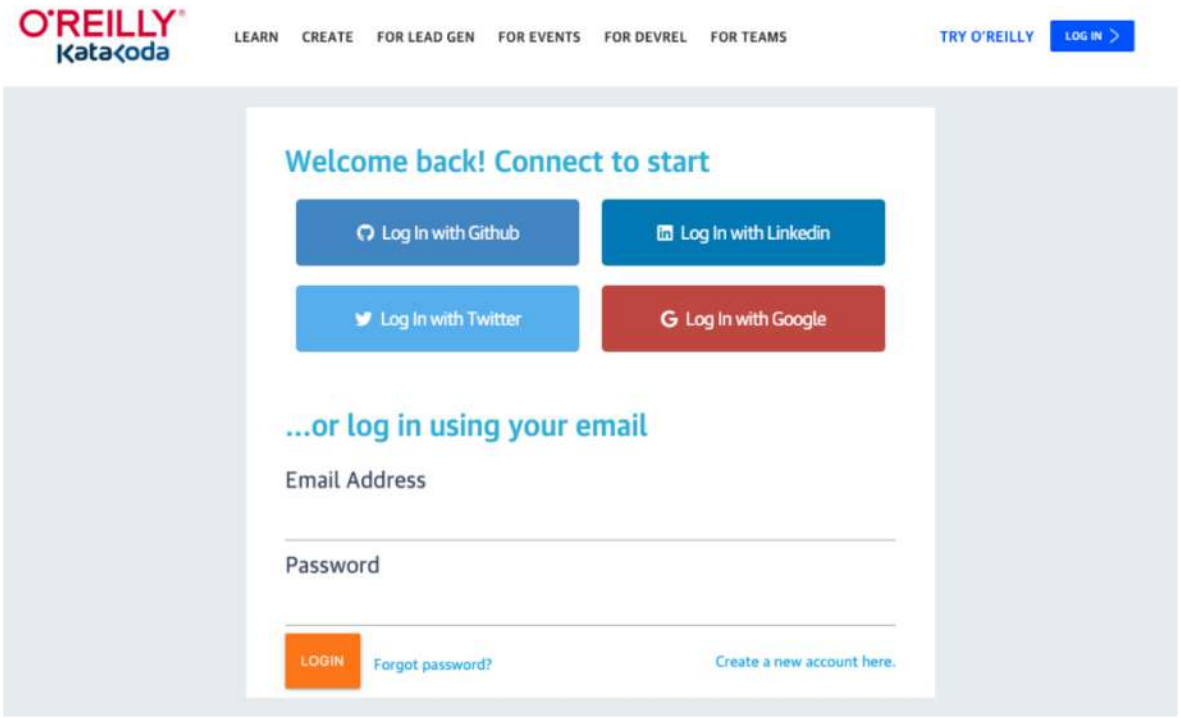
デメリット

- 時間制限があり、インスタンスを継続して利用できない
- 極めて素のLinuxの動作に近いがなんらかの制限はある
  - ターミナルでexitを入力するとインスタンスが終了してしまう
  - Dockerでデタッチ（Ctrl+PのあとにCtrl+Q）するとインスタンスが終了してしまう

など

ハンズオン

ここから先はハンズオンになります。
参加する方は「Katacoda」にログインしてください。
※Twitter、Google、GitHubいずれかのIDを持っていればソーシャルログインが可能です

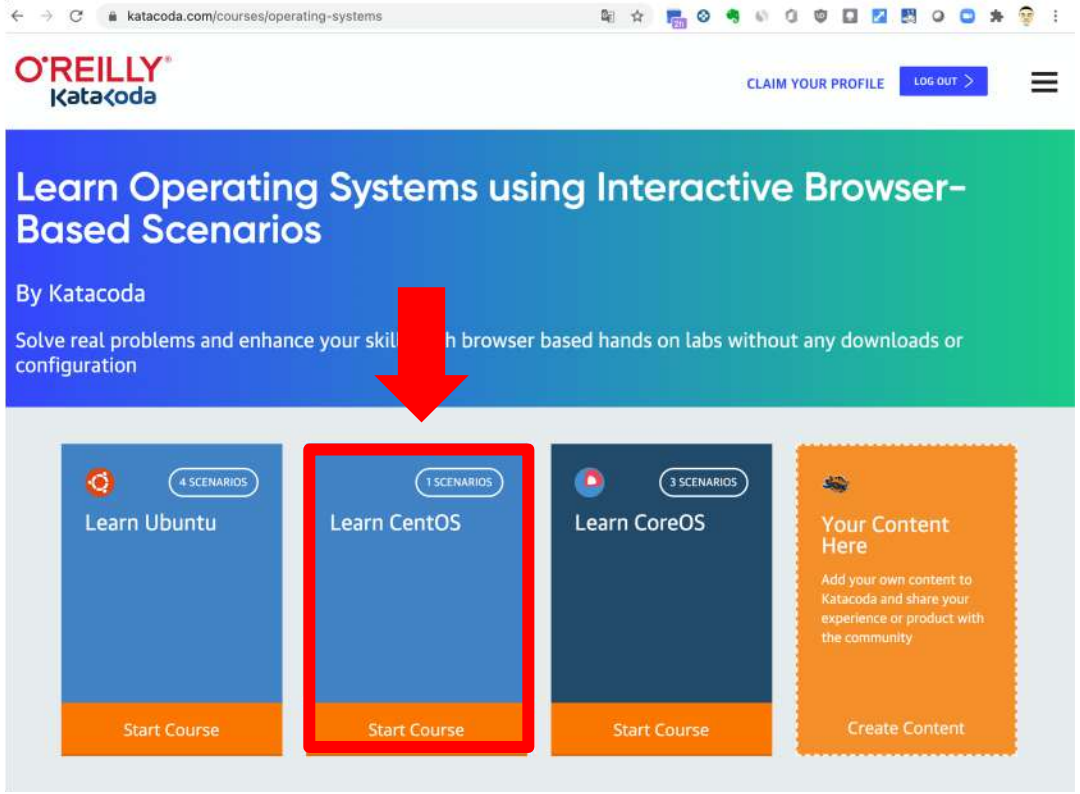


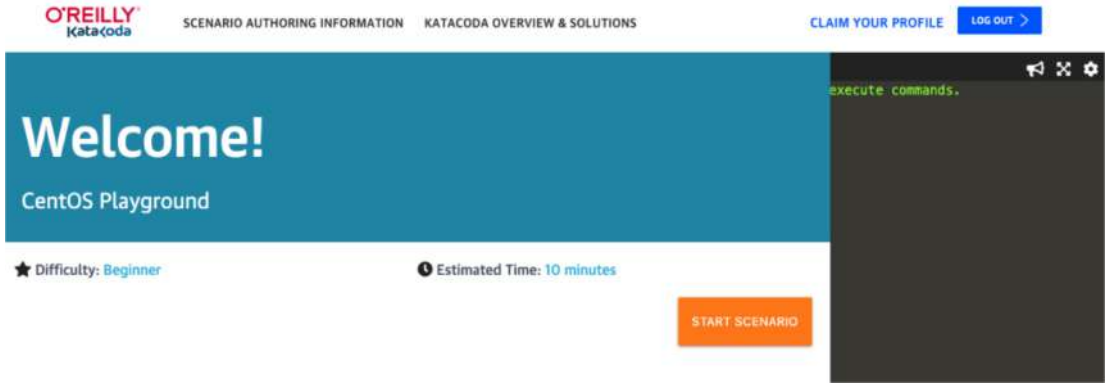
https://www.katacoda.com/login

CentOSプレイグラウンドにアクセス

Learn CentOSコースでStart Courceを選択後、PlaygroundのExplore Playgroundをクリックしてください。

https://www.katacoda.com/courses/centos/playground





START SCENARIOボタンをクリックすると、しばらくしてBashターミナルが表示されます。

左メニューに「Hello World」が表示されていればOKです。インスタンスが終了してしまうので、CONTINUEボタンはクリックしないでください。

参考資料について

- リナスクにてハンズオンの手順を解説した参考資料を公開しています。そちらを見ながら、コマンド入力してください。

https://linuc.spa-miz.com/?p=2232

講師も参考資料を共有しながらハンズオンを進めていきます。

✓長いコマンドは間違えると再入力も大変ですのでコピー＆ペーストを利用してください。

ハンズオンの中でも説明しますが、コードをペーストした際、まれにコンソールの頭に全角スペースが挿入されてしまうことがあります。そうなった場合はスペースを削除してからエンターキーで実行してください。

Ctrl+aでコンソールの先頭にカーソルを移動できます。

KatacodaのCentOS Playgroundでコンテナを動かす

■CentOSプレイグラウンドの状態確認(OS情報の確認)

- uname –a, uname –r
- cat /etc/os-release
- hostnamectl
  →Virtualizationがkvmになっているのを確認。Kataodaのサービスが仮想化技術で動作

■Apache(httpd)の導入テスト

- yum install httpd -y
- systemctl start httpd
- systemctl status httpd
- Web Previewで80番ポートにアクセスしてApacheが動作していることを確認

■htmlファイルを作って置き換えてみよう

- /var/www/htmlにindex.htmlを作成
  - echo Apache Works! > /var/www/html/index.html
- Web Previewでhttp 80番ポートにアクセス（リロードでOK）

KatacodaのCentOS Playgroundでコンテナを動かす

Dockerの導入状態確認

- Dockerのバージョン確認
  - docker –v, docker version, docker info
  - バージョンが1.2.xの場合は新体系コマンドは使えないのでアップデート
    (1.3xでもビルドバージョンが古いので全員でyum update dockerを実施)
- Dockerの再起動
  - systemctl restart docker
  - systemctl status docker
- Dockerヘルプの使い方
  - docker help
  - docker container --help, docker image --help
  - docker container run –-help

Dockerの基本動作の確認

- Hello Worldコンテナの実行
  - docker run hello-world

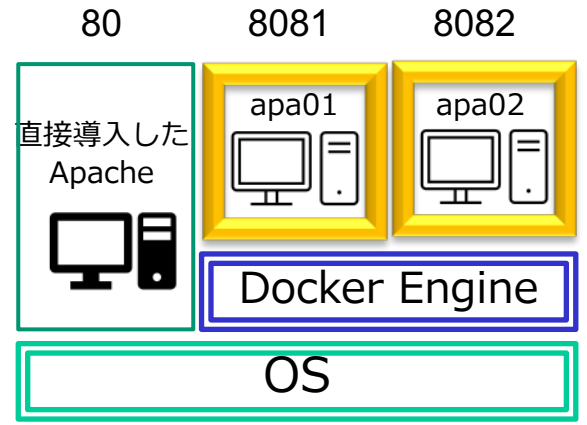
KatacodaのCentOS Playgroundでコンテナを動かす

■コンテナを実際にデプロイしてDockerの基本操作を学ぶ

- コンテナの実行（run=pull+create+start）
  - docker run -it --name ubu1 ubuntu:bionic /bin/bash
  - cat /etc/os-release　でubuntuのバージョン情報を確認
  - uname -r　でCentOSのカーネルを共有していることを確認　→ exit
- コンテナの起動と停止（start, stop）
  - docker ps → docker ps –a → docker start ubu1 → docker ps → docker stop ubu1 → docker ps
- バックグラウンドで動作中のコンテナでプログラムを実行、その後アタッチ
  - docker run -dit --name ubu2 ubuntu:bionic /bin/bash (2度目は生成が速い)
  - docker exec ubu2 cat /etc/os-release
  - docker attach ubu2　→　exit
- ログの確認
  - docker logs ubu1　→　docker logs ubu2
- コンテナとイメージの管理
  - docker ps -a, docker container ls –a
  - docker rm, docker container rm, docker container prune
  - docker images, docker image ls, docker rmi, docker image rm

KatacodaのCentOS Playgroundでコンテナを動かす

■Docker Hubの公式Apacheコンテナを複数動かす



- 名前を付け、ポートを指定してバックグランドで実行
  - docker run –d –-name apa01 –p 8081:80 httpd
  - docker run –d –-name apa02 –p 8082:80 httpd
  - Webプレビューでそれぞれのポートを指定して動作確認（It Works!）
  - docker psでコンテナの状態確認
  - ホストOSにインストールしたApacheとあわせて3つが同時に動いていることを確認
- index.htmlファイルを作成（どのような方法でも良い）
  - mkdir web1 web2
  - echo Web1 Works! > web1/index.html
  - echo Web2 Works! > web2/index.html
- コンテナ内の/usr/local/apache2/htdocsにindex.htmlを配置
  - docker cp web1/index.html apa01:/usr/local/apache2/htdocs/
  - docker cp web2/index.html apa02:/usr/local/apache2/htdocs/
  - Webプレビューでそれぞれのポートを指定して動作確認（Web1 Works!, Web2 Works!）
  - 隔離されているコンテナにホストOS側からファイルをコピーできることを確認

■デタッチ(Ctrl+p, q)の動作確認

- docker run -it --name ubu3 ubuntu:bionic /bin/bash　→　Ctrl+p, q
- The environment has expired.Please refresh to get a new environment.

KatacodaのUbuntu Playgroundでコンテナを動かす

- **Ubuntu 20.04 Playgroundを使用**
- **Ubuntuの動作確認**
  - cat /etc/os-release
  - hostnamectl
- **プロンプトの変更**（わかりにくいので）
  - export PS1="[¥u@$HOSTNAME ¥w]$"
- **Ubuntu Playgroundの特徴**（CentOS Playgroundとの違い）
  - インスタンスにDockerイメージが多数含まれる(docker psで起動中のコンテナが無いことを確認、docker images)
  - VS CodeとWeave Scopeの統合(IDEとVisualize Hostを起動)
- **Dockerの動作確認**
  - docker –v, docker version, docker info
  - docker ps –a (Visualize Host[Weave Scope]のコンテナが動いている)
- **CentOSのコンテナを動かす**
  - docker run –it --name cent7test centos:7 /bin/bash
  - cat /etc/os-release

Dockerfileでカスタムコンテナイメージを作成する

Dockerfileとは

- カスタマイズしたコンテナイメージを作成するための手順書のようなもの

index.htmlを置き換え済みのApacheコンテナイメージを作成

- index.htmlファイルを用意（Apache Web Server build by Dockerfile）
  - echo Apache Web Server build by Dockerfile > index.html
- 以下の内容をcatコマンドのヒアドキュメントでDockerfileとして作成
  - FROM httpd
  - COPY index.html /usr/local/apache2/htdocs/
  - cat > Dockerfile << EOF
    > FROM httpd
    > COPY index.html /usr/local/apache2/htdocs/
    > EOF
- docker build –t apaimg[イメージ名] .[Dockerfileがあるディレクトリ]
- docker images, docker image lsでイメージがあるか確認

生成したイメージで新たなApacheコンテナを生成

- docker run –d --name apa03 –p 80:80 apaimg
- Web Previewで動作確認

Docker ComposeでWordPressサイトを構築する

一般的に、WordPressサイトの構築には、 LAMP環境と呼ばれる、**Linux OS**、**Apache**、**MySQL(またはMariaDB)**、**PHP(またはPython)**の4つが必要。

LinuxサーバーでWordPressを使うには、 LAMP環境が連携するように設定しなくてはいけない。

Docker HubにはWordPress社公式のコンテナが公開されており、**Docker Compose**を使えば、LAMP環境の連携を設定した上でWordPressサイトが簡単に構築できる。

Docker Compose
複数のコンテナで構成されるアプリケーションの構築に必要な、コンテナイメージの情報や起動のための各種設定、さらにコンテナ同士が連携するために必要な手順などをdocker-compose.ymlファイルに記載してDocker Composeコマンドで実行する。構築に必要なコマンドは1つだけ。

Docker ComposeでWordPressサイトを構築する

■Docker Composeのインストール

- Ubuntu Playgroundでは導入済み
- CentOS Playgroundで試したい場合はyum install docker-composeでインストール

■Docker Composeのバージョン確認

- docker-compose –v

■プロジェクト用ディレクトリを作成し同ディレクトリに移動

- mkdir mywp && cd mywp

■docker-compose.ymlの作成

- Docker Hub経由でWordpress公式の設定をコピーし、viエディタを使いdocker-compose.ymlというファイル名で作成する。
  https://hub.docker.com/_/wordpress
- apacheとphpはWordPressの公式イメージで自動的に組み込まれる

Docker ComposeでWordPressサイトを構築する

Docker Composeを実行

- docker-compose up -d (-dオプションはデタッチの意味＝バックグラウンド実行)を実行すると、docker-compose.ymlに記載された内容で、WordPressとMySQLのコンテナイメージが取得され、それぞれが連携するように自動的に設定が行われる。
- 特にエラーが表示されていなければ導入は完了。完了まで多少時間がかかる場合がある。

Docker Composeの基本コマンド

- 実行
  - docker-compose up (–d)
- 停止
  - docker-compose stop
- 終了
  - docker-compose down（停止、コンテナやネットワークの削除。イメージは残る。）
  - docker-compose down --rmi all（イメージを含む全削除）

Docker ComposeでWordpressサイトを構築する

■WordPressの動作チェック

- ブラウザ左上の+ボタン（WebPreview）をクリックして「Select port to view on Host 1」を選択し、####に8080を入力してDisplay Portボタンをクリック。WordPressのロゴが入った初期設定画面が表示されたら成功。
- 必須事項(注)を記入して下のインストールボタンを押せば設定が完了。設定したアカウントでログインすればWordPressのダッシュボードが表示される。
- ダッシュボード左上のタイトルにマウスオーバーして「サイトを表示」をクリックすればブログ画面が表示される。
- WordPressの使い方についてはセミナー趣旨と異なるのでこれ以上の解説は控える。

■Dockerコマンドでコンテナの動作確認

- docker ps　→　wordpressとmysqlのコンテナが動作していることを確認
- docker image inspect wordpressでイメージの詳細が確認できる

■Visualise Host(Weave Scope)で確認

■Docker Composeを実行

- docker-compose down　→　WordPressに関わる全てのコンテナが停止して破棄される。

(注)Katacodaのインスタンス自体、短時間で消滅してしまうので必須事項は適当な物で良い。ただし、ユーザー名やパスワードをadminやdockerなど、単純なものにするとWebブラウザが当該ユーザーのパスワードが漏洩しているとして警告してくることがある。

まとめ

■セミナーゴールにたどり着けた
- コンテナの起動・再開・停止が行える（1.02.2の範囲）
- Dockerコンテナを実際に自分の手で操作してみる（2.06.2の一部）
- ハンズオンが滞りなく終わったとしたら60以上のコマンドを実行したことになる。

■クラウド型の学習環境を使うと、数ステップでLinuxを操作できる。Webブラウザさえあれば思い立った時にすぐに使える。クラウド環境だけでも、各種サーバーアプリケーションやコンテナの構築など、踏み込んだ学習が可能。Dockerも大半のコマンドが実行できるし、DockerfileやDocker Composeも利用できる。

■一方で、時間的な制約があるため、継続性が要求される内容の学習は難しい。仮想化技術がベースになっているサービスなので、利用者側も使い捨て感覚で利用する方が精神衛生上も良い。クラウド学習環境のメリットとデメリットをしっかし把握した上で、実機環境とうまく使い分けてカジュアルに活用したい。

■今回がコンテナ初体験という方は、次のステップとしてLinux実機や仮想マシンでDockerを使ってみて欲しい。デプロイしたアプリの継続利用やデータの永続化も可能になる。

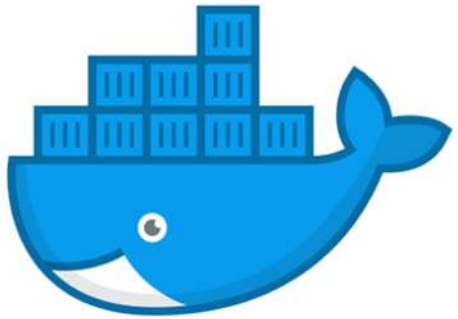
余談

■新体系と旧体系コマンドについて
- どちらで覚える？
- 試験にはどっちが出る？

■CentOS7コンテナでsystemctlを使えるようにする
- docker run -itd --privileged centos:7 /sbin/init

■Dockerの学習にオススメの書籍
- 仕組みと使い方がわかるDocker & Kubernetesのきほんのきほん
- さわって学ぶクラウドインフラ docker基礎からのコンテナ構築

ご静聴、ご参加、ありがとうございました！



今回のハンズオンで実施したこと

- クラウド経由でLinuxの仮想環境を利用
- クラウド仮想環境にWebサーバーを構築
- Dockerコマンドでさまざまなコンテナを実行
- Dockerfileでカスタムイメージを作成
- Docker ComposeでWordpressサーバーを構築

LPI-Japanとしてもハンズオン形式は初めての試みです。
実際に手を動かしてみていかがでしたか。
ご感想やご意見をお寄せください。