LinuC レベル 1 Version10.0
技術解説無料セミナー
2020/5/23 開催

主題	1.01 Linuxのインストールと仮想マシン・コンテナの利用
副題	1.01.2 仮想マシン・コンテナの概念と利用

本日の講師

インターノウス株式会社
（LPI-Japanアカデミック認定校）

竹本 季史

LPI-JAPAN

■会社紹介：インターノウス株式会社

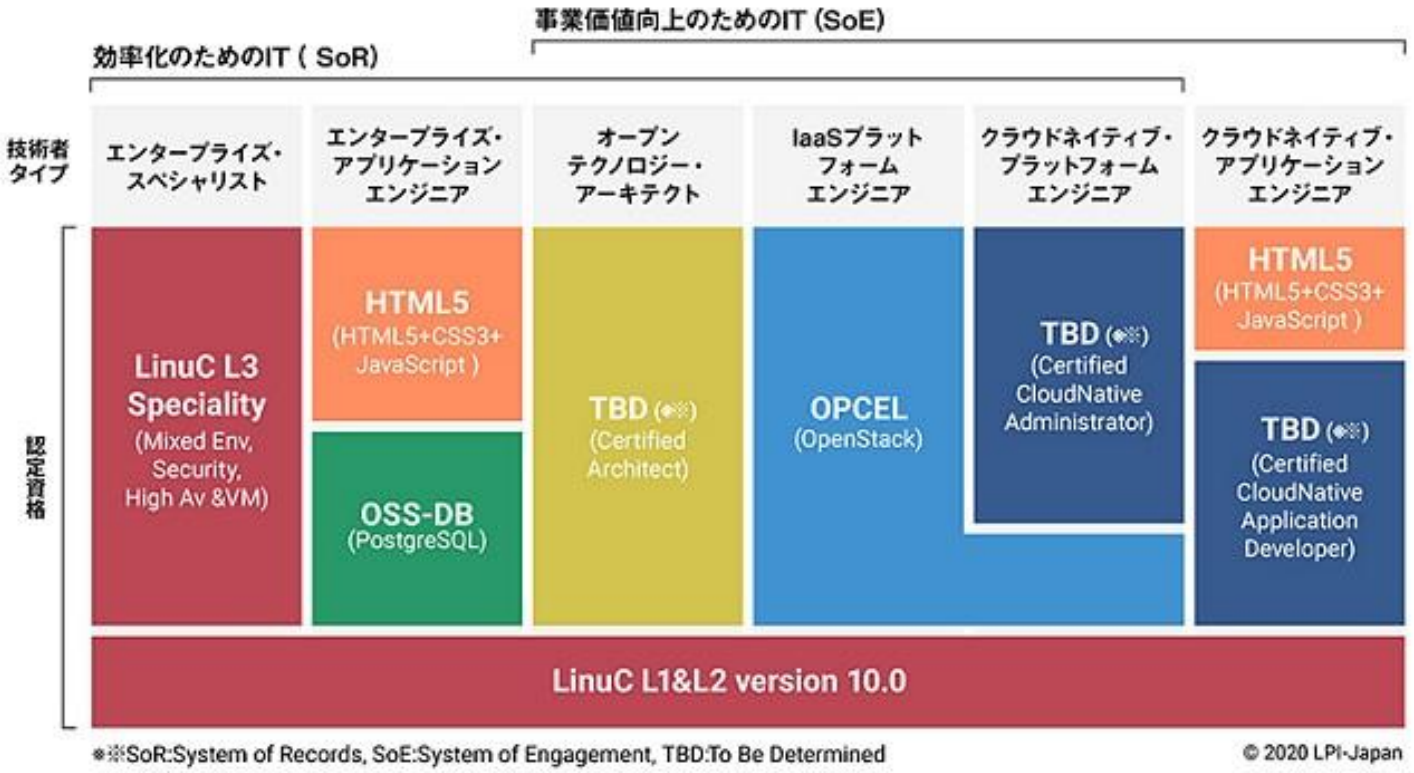
- ●人材紹介サービス、人材派遣/SESサービス、IT未経験者の教育及び就職支援サービス、法人研修サービス
- ●未経験からインフラエンジニアやプログラマーになりたい方へ、無料で研修と就職支援サービスを行っています。
  https://proengineer.internous.co.jp/lp_kiso3/

■自己紹介：竹本 季史(たけもと ときふみ)

- ●IT業界で約10年間勤務後、インターノウス株式会社エンジニアカレッジ講師。
- ●これまで約700人を未経験者からエンジニアに養成。Linuxサーバー(メール、OpenSSH、シェルスクリプト、DB、監視、演習)を担当。
- ●LinuCレベル1バージョン10.0の差分教材で「仮想マシン・コンテナの概念と利用」を執筆。

■LinuCとは

- ●Linuxを起点にしたすべてのITエンジニアを対象とした認定
- ●LinuCの認定取得に向けてLinuxを学習することは、どんな領域へのキャリアを築くにしても役立ちます（ITサービスの基盤はほぼ全てLinux！）



- ✓ LinuCでは、ITエンジニアの誰もが学ぶべきことを学べます（クラウド、セキュリティ、ネットワーク、システム管理、他）
- ✓ LinuCを学んだ上でパブリッククラウドを学ぶものいいです！

■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	・個人利用を想定した技術の削除 ・コマンドのアップデート	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	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	RAID、記憶装置へのアクセス方法、FTPサーバーの保護、他

＊レベル１は受験しやすくなりました！
→ブート周りの知識やシステム立ち上げ時のシステムチェックなどはレベル２に移設

■本セミナーについて

- ●本セミナーは試験範囲で問われる内容の理解を深めるためにポイントを解説いたします。
- ●このセミナーの内容はCentOS7を前提とした内容となっています。

■受講者の想定スキルレベル

- ●LinuCレベル1の取得を目指している方

■本セミナーのゴール

- ●仮想マシンとコンテナの違いが分かる。
- ●Dockerのインストールからコンテナの起動、接続、停止までの流れが分かる。

1.01.2 仮想マシン・コンテナの概念と利用

■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**

仮想マシンとコンテナの解説

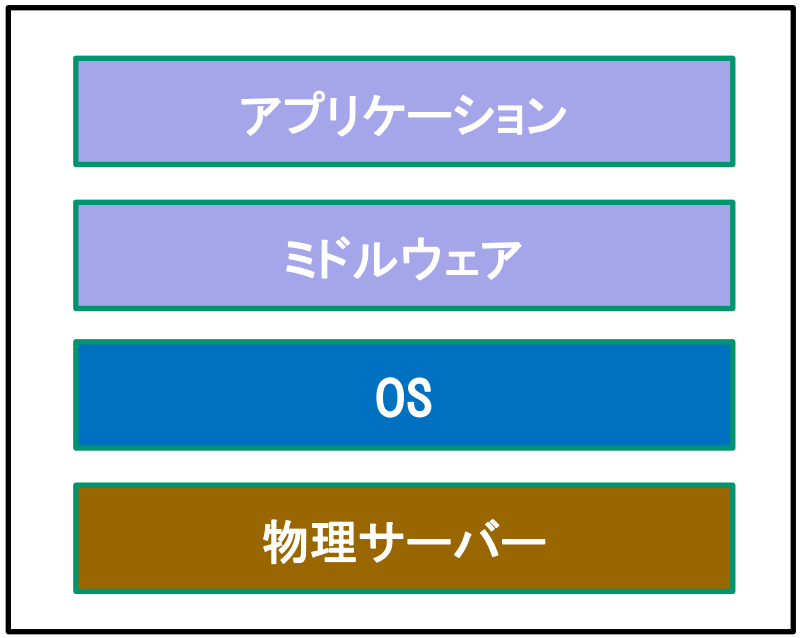
- サーバーを仮想化する必要性
- 仮想マシンの種類
- 仮想マシンの特徴
- ホスト型仮想マシンの例
- コンテナとは
- コンテナの特徴
- 仮想マシンとコンテナの違い
- 物理サーバー、仮想マシン、コンテナのイメージ

サーバーを仮想化する必要性

■基本的に1台の物理サーバーには1つのOSしか使用できません。

■仮想マシンやコンテナを使用すれば、1台の物理サーバー上で複数台のサーバーが動作しているかのように使うことができます。

■仮想化技術が使われるようになった背景には、サーバーの性能が上がり、1つのOSではサーバーの性能を十分に活かすことができないため、サーバー資源の効率的な活用のためという目的があります。



仮想化を使わない場合
(1台の物理サーバーに1つのOS)

仮想マシンの種類

■仮想マシンはハイパーバイザと呼ばれるソフトウェア上で動作します。ホストOSが必要なホスト型とホストOSが不要なハイパーバイザ型の2種類があります。

■ホスト型の製品

- VMware Workstation Player、Oracle VM VirtualBox

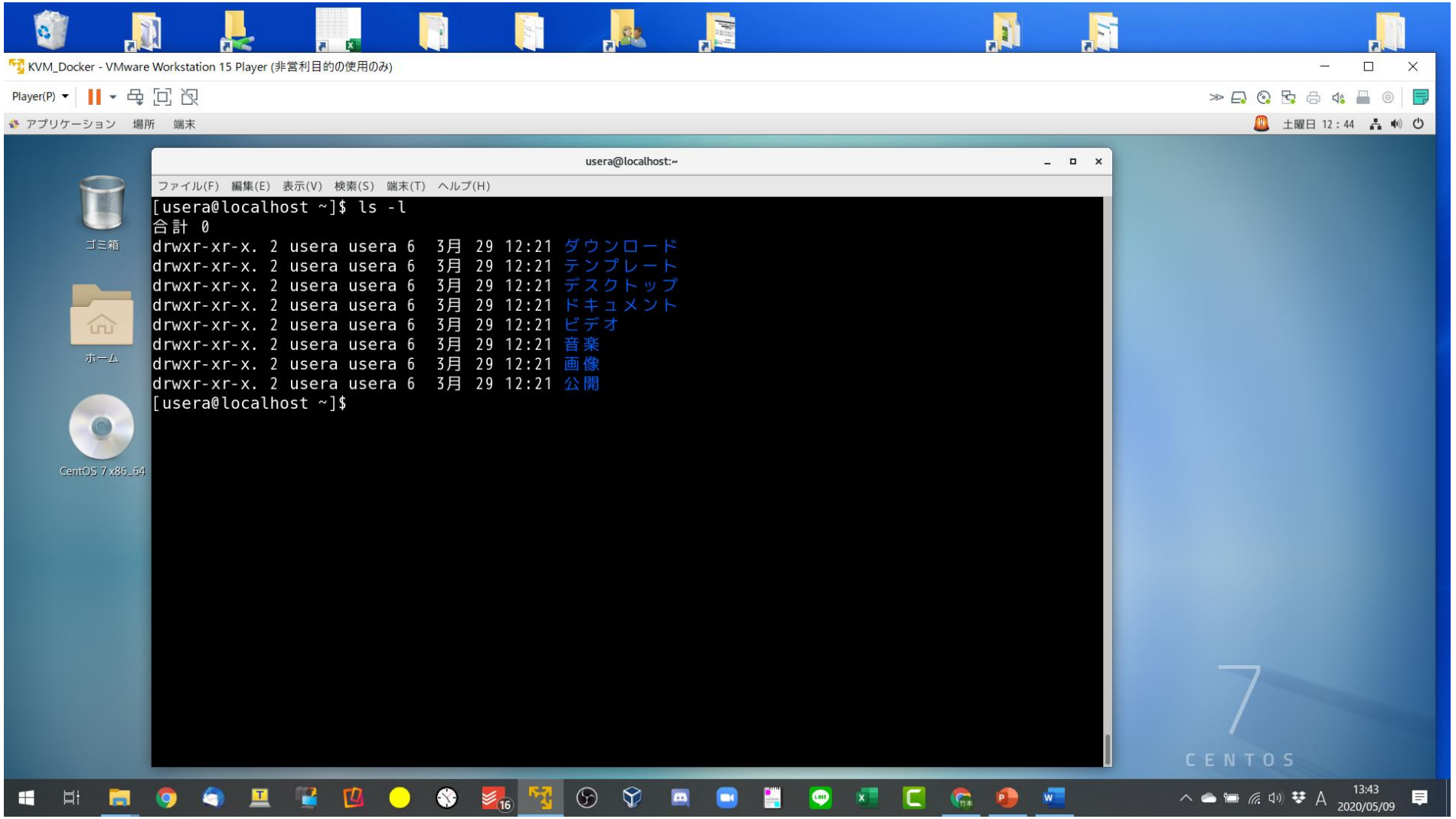
■ハイパーバイザ型の製品

- KVM、VMware ESX/ESXi、Microsoft Hyper-V、XenServer

仮想マシンの特徴

- 仮想マシンのゲストOSには独自のOSが必要
- ゲストOSはホストOSと異なるOSでも利用可能
- 複数の仮想マシン間は異なるOSを利用可能
- 起動が遅い
- CPUやメモリのリソース消費が大きい
- 仮想マシンのファイルサイズが大きい
- 別の環境に移動・配布は可能だが
  ファイルサイズが大きいため時間がかかる

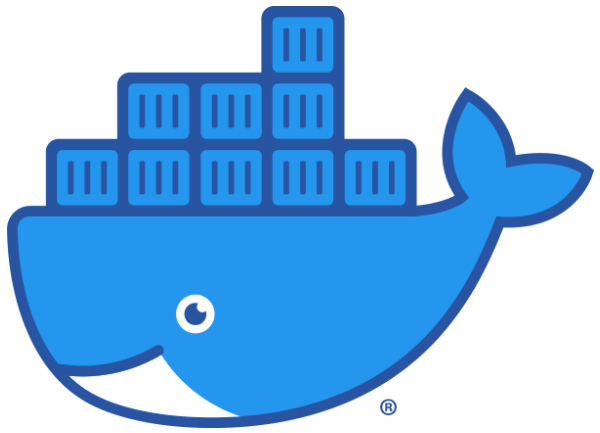
ホスト型仮想マシンの例



Windows10上のVMware Workstation PlayerでCentOS7を実行

コンテナとは

- コンテナはホストOSのカーネルを利用し、ホストOSのプロセスを分離することで、独立したOSのように動作します。
- コンテナの製品にはDocker、Podmanがあります。



Docker

Podman

コンテナの特徴

- ホストOSのカーネルを利用するためコンテナにOSがない
- ホストOSとコンテナで異なるOSは使用できない
- 起動が速い
- CPU・メモリのリソース消費が少ない
- コンテナのファイルサイズが小さい
- 別の環境に簡単に移動・配布できる

仮想マシンとコンテナの違い

	仮想マシン	コンテナ
OSの要不要	必要	不要
複数の仮想マシン/コンテナで異なるOSを使用	○可能	×不可 (ただし、ホストOSがLinuxなら他のディストリビューションを利用可能)
起動速度	△遅い	○速い
CPU・メモリのリソース消費	△大	○少
イメージのファイルサイズ	△大	○小
他の環境への移動や配布	△可能	○簡単

物理サーバー、仮想マシン、コンテナのイメージ

物理サーバー

土台(物理サーバー)

- 土台に1軒の家
- 壁なしの1LDK
- 家に1つのOS

仮想マシン

土台(物理サーバー)

- 土台に複数世帯住宅
- 壁なしの1LDK
- それぞれの家に1つのOS

コンテナ

土台(物理サーバー)

- 土台に1軒の家
- 壁で仕切って部屋を作る
- それぞれの部屋に1つのOS

Dockerの準備

- Dockerのインストール
- Dockerサービスの起動
- Dockerサービスの自動起動の設定

※ここからの操作はrootユーザーで操作することを想定しています。

Dockerのインストール

■yumコマンドでDockerをインストールします。

 # yum install docker

```
[root@localhost ~]# yum install docker
読み込んだプラグイン:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: ftp.iij.ad.jp
 * extras: ftp.iij.ad.jp
 * updates: ftp.iij.ad.jp
```

■インストールの確認をします。

 # rpm -q docker

```
[root@localhost ~]# rpm -q docker
docker-1.13.1-109.gitcccb291.el7.centos.x86_64
```

■バージョンの確認をします。

 # docker -v

```
[root@localhost ~]# docker -v
Docker version 1.13.1, build cccb291/1.13.1
```

Dockerサービスの起動

■Dockerサービスを起動します。
 # systemctl start docker

```
[root@localhost ~]# systemctl start docker
```

■Dockerサービスが起動していることを確認します。
 # systemctl status docker

```
[root@localhost ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: d
isabled)
   Active: active (running) since 火 2020-03-24 12:02:04 JST; 3min 12s ago
     Docs: http://docs.docker.com
 Main PID: 6812 (dockerd-current)
    Tasks: 17
   CGroup: /system.slice/docker.service
           ├─6812 /usr/bin/dockerd-current --add-runtime docker-runc=/usr/libexec/...
           └─6817 /usr/bin/docker-containerd-current -l unix:///var/run/docker/lib...
```

Dockerサービスの自動起動の設定

- DockerサービスがLinux起動時に自動的に起動するように設定します。

 # systemctl enable docker

```
[root@localhost ~]# systemctl enable docker
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to
/usr/lib/systemd/system/docker.service.
```

- 自動起動が設定されていることを確認します。

 # systemctl is-enabled docker

```
[root@localhost ~]# systemctl is-enabled docker
enabled
```

【実技】Dockerを使ったコンテナの操作

- コンテナ操作の流れ

①DockerHubからコンテナイメージを取得

②コンテナの作成と起動

③コンテナから切断してコンテナ状態を確認

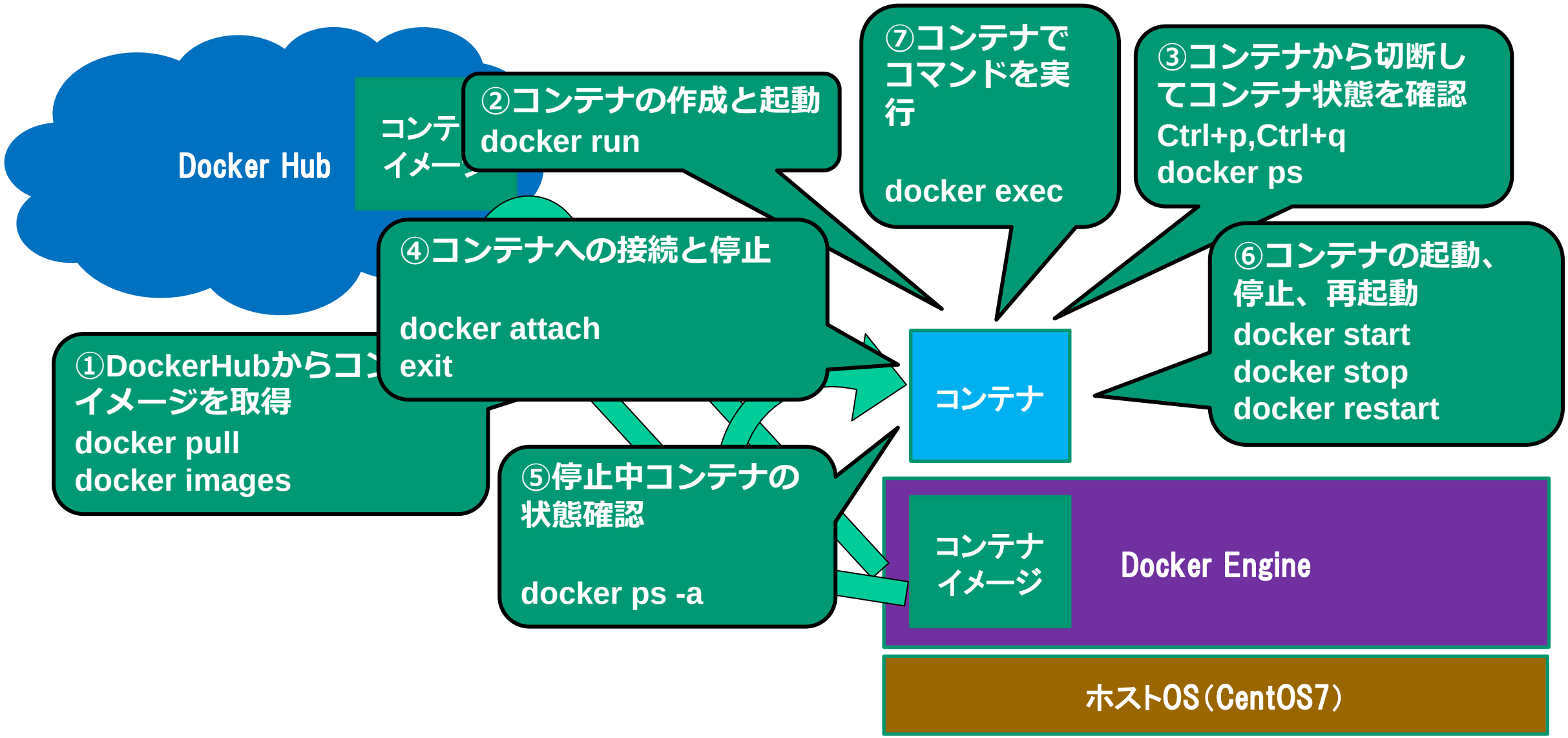
④コンテナへの接続と停止

⑤停止中コンテナの状態確認

⑥コンテナの起動、停止、再起動

⑦コンテナでコマンドを実行

コンテナ操作の流れ



①DockerHubからコンテナイメージを取得

■コンテナの元となるコンテナイメージをDockerHubから取得します。
ここではCentOSの最新版を取得します。
 # docker pull centos:latest

```
[root@localhost ~]# docker pull centos:latest
Trying to pull repository docker.io/library/centos ...
latest: Pulling from docker.io/library/centos
8a29a15cefae: Pull complete
Digest: sha256:fe8d824220415eed5477b63addf40fb06c3b049404242b31982106ac204f6700
Status: Downloaded newer image for docker.io/centos:latest
```

■コンテナイメージを取得できたことを確認します。
 # docker images

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker.io/centos    latest              470671670cac        2 months ago        237 MB
```

②コンテナの作成と起動

■docker runでコンテナイメージからコンテナを作成して、
bashシェルの起動と接続を行います。
プロンプトのホスト名がコンテナのIDとなっており、接続できたことが分かります。

 # docker run -it centos:latest /bin/bash

- -it：コンテナと標準入出力をつなげる
- centos:latest：コンテナ作成の元となるイメージ
- /bin/bash：コンテナで実行するコマンド

```
[root@localhost ~]# docker run -it centos:latest /bin/bash
[root@df219e6e8390 /]#
```

ホスト名がコンテナのIDとなっている

■コンテナのCentOSのバージョンを確認します。最新の8.1であることが分かります。

 # cat /etc/redhat-release

```
[root@df219e6e8390 /]# cat /etc/redhat-release
CentOS Linux release 8.1.1911 (Core)
```

③コンテナから切断してコンテナ状態を確認

■コンテナからデタッチ(切断)してホストOSに操作を切り替えるには、
Ctrlを押しながらp、qを連続して押します。

```
[root@df219e6e8390 /]#
[root@df219e6e8390 /]# [root@localhost ~]#
```

Ctrl+p,Ctrl+qを
押すことでホストOS
に戻りました

■ホストOSでコンテナの起動状態を確認するには、docker psを実行します。
STATUSにUpと表示されています。

 # docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
    PORTS               NAMES
df219e6e8390        centos:latest       "/bin/bash"         4 minutes ago       Up 6 seconds
                        competent_noether
```

④コンテナへの接続と停止

■デタッチ(切断)後、再びコンテナを操作するにはアタッチ(接続)します。
[コンテナID]は全コンテナで一意となればよいので
始めの2,3文字の入力でも実行可能です。
 # docker attach [コンテナID]

```
[root@localhost ~]# docker attach 242eec2998a5
[root@242eec2998a5 ~]#
```

```
[root@localhost ~]# docker attach 24
[root@242eec2998a5 ~]#
```

どちらでも接続可能

■**exit**を実行するとコンテナが停止してホストOSに操作が戻ります。
起動中のbashシェルがexitするためコンテナが停止します。

```
[root@9b3c1c0d4d83 /]# exit
exit
[root@localhost ~]#
```

ホストOSに戻りました

⑤停止中コンテナの状態確認

■停止中のコンテナを確認するときはdocker psに-aオプションを付加します。
Exited(0)はコンテナが正常に停止したことを示します。

 # docker ps –a

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
              PORTS               NAMES
df219e6e8390        centos:latest       "/bin/bash"         18 minutes ago      Exited (0) About a
 minute ago                         competent_noether
```

⑥コンテナの起動、停止、再起動

■停止中のコンテナを起動します。

 # docker start [コンテナID]

```
[root@localhost ~]# docker start 242eec2998a5
242eec2998a5
```

■起動中のコンテナを停止します。

 # docker stop [コンテナID]

```
[root@localhost ~]# docker stop 242eec2998a5
242eec2998a5
```

■コンテナを再起動します。コンテナが停止している時は起動します。

 # docker restart [コンテナID]

```
[root@localhost ~]# docker restart 242eec2998a5
242eec2998a5
```

⑦コンテナでコマンドを実行

■docker execでコンテナでコマンドを実行します。
ここではコンテナでhostnameコマンドを実行した結果をホストOSに出力します。

 # docker exec -it [コンテナID] hostname

コンテナと標準入出力をつなげる

コンテナID

コンテナで実行するコマンド

```
[root@localhost ~]# docker exec -it 242eec2998a5 hostname
242eec2998a5
```

hostnameコマンドの実行結果