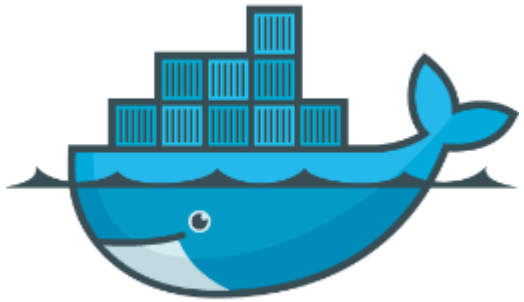
Docker コンテナ構築

ヤポドゥ ハンズオン勉強会 Vol.2



自己紹介

・田村 亮
・@launcher_test
・フリーランスエンジニア (一人親方)
・インフラ系

フリーランスになる前後(2012年くらい)からハンズオン勉強会を開催したかった。

もともとコカコー○の自販機補充員とか配管工とか佐○急便の仕分けとか警備員とか色々やってました。

ヤポドゥ勉強会

- ハンズオン形式でインフラ系技術の習得を目指す。
- 現場で使える技術を学ぶ
- とりあえず動かせるレベルが目標
- 1 ～ 3ヶ月毎に開催予定
- Github https://github.com/yapodu
- ハッシュタグ #yp_study

※ヤポドゥとはラサ語(チベット語)で「良い」の意味

いきなり Docker 運用で得た知見

- 銀の弾丸にあらず

  => 全てをコンテナ化することは現実的ではない

- 仮想サーバではない

  => カプセリングされたバイナリと思うといいかも

- 破棄が前提になるようなシステムに最適

  => Blue-Green Deployment な世界

- CentOSやUbuntoもコンテナ用カスタマイズされているので通常のOS設定とは異なる

  => カスタマイズ可能だが肥大化が伴う、せっかくのコンテナ化の意味がなくなる

- いいことのはずだがバージョンアップスピードが早くて辛い

  => 昨日参照した公式ドキュメントが古くなっている。。。

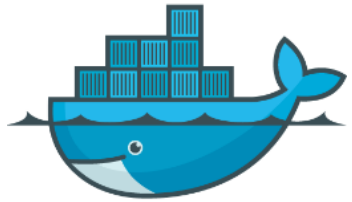
- 既存システムからの載せ替えは用ベンチマーク(特にディスク)

  => ホストと同じノリだと死ねる。ハンズオンで確認します。

Docker とは

- コンテナの管理ができる
- もとはSaaSベンダー dotCloud が開発
- SaaSではなく自社開発してたコンテナ技術＝Dockerが有名に、そして社名変更
  http://www.publickey1.jp/blog/13/dotclouddockerdocker.html

- コンテナ自体は古くからある(FreeBSD Jails,Solaris Zone等)
- Dockerは容易な導入、ポータビリティの高さから現状の人気を得る
- AWS,MS,Google,Redhat等が絡んでプラットフォームとして連結ツール開発/サービスしている



Docker 現状と今後

現状

- Dockerをコアとしたプラットフォームが出来上がった
- 構築/利用が広がっている
- エコシステムが出来つつある
- 運用知見の蓄積

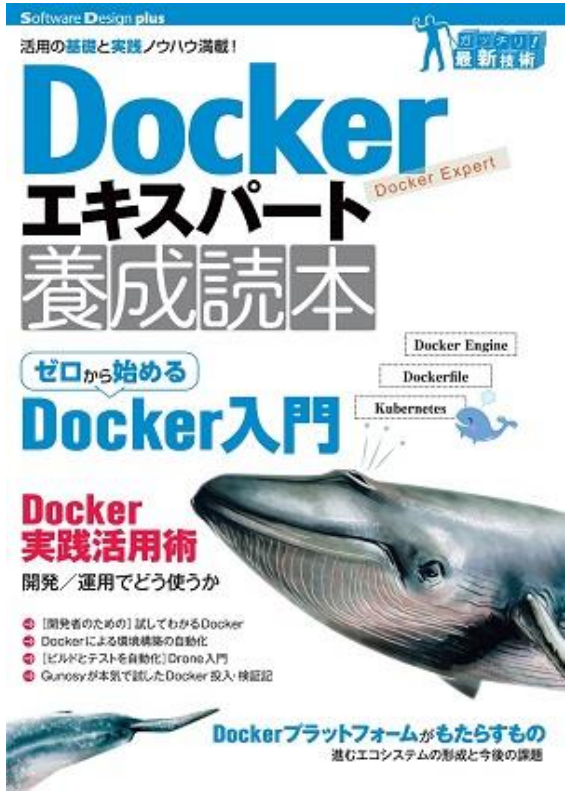
今後

- Linux以外のOSにもDockerベースのコンテナ化が可能に
- コンテナ専用OSの普及
- 既存クラウドとの連携
- 冗長、分散等のスタンダード技術の普及(現状バラバラ)
- 完全なコンテナクラウドサービスの普及

Docker その他の情報

- 本日はハンズオンでの作業をメインにしているので詳細説明は割愛
- ネットの情報も有益だが、とりあえずDocker経験値の少ない人は
  Dockerエキスパート養成読本がおすすめ。

  ※下手にいろんな情報に触れるよりこの1冊で十分
- もちろん公式ドキュメントも
  https://docs.docker.com/



作業前確認

- PDF

  https://github.com/yapodu/study-pdf/blob/master/Docker_201507.pdf

- お渡ししたインスタンス情報に沿って鍵認証パス無しログインして下さい。
- su でrootになれることを確認して下さい。
- 親機(ホスト機)のCentOS7はdockerインストールしていませんがAnsibleによる初期セットアップし実施済みです。

  playbook : https://github.com/yapodu/Ansible

Docker のインストール

・本手順では root で作業を実施しますが

ログインユーザー yp-studyにsudoを割り当ててあるのでお好きな方法で実施して下さい。

■docker yum インストール

```
$ su -
# yum install docker
```

Systemd 設定・確認

■ Systemd 設定・確認

起動

```
# systemctl start docker
# docker info
```

自動起動設定

```
# systemctl enable docker
```

確認

```
# ps -ef | grep docker
root      720     1  0 Jul23 ?
 00:03:16 /usr/bin/docker -d --selinux-enabled
# systemctl status docker
docker.service - Docker Application Container Engine
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled)
  Active: active (running) since Thu 2015-07-23 11:17:46 JST; 24h ago
# systemctl list-unit-files | grep docker
docker.service                          enabled
```

Dockerイメージのダウンロード

- DockerコンテナはDockerイメージと呼ばれるコンテナテンプレートから作成される。
- 通常はDocker公式のレポジトリサイト docker hub(hub.docker.com)からダウンロードしコンテナを作成する
- 以下はhttps://registry.hub.docker.com/_/centos/ のcentos6イメージを取得している

■docker pull と確認

書式：$ docker pull {イメージ名}:{タグ名}

以下はhttps://registry.hub.docker.com/_/centos/ のcentos6イメージを取得している

```
# docker pull centos:centos6
```

```
# docker images
REPOSITORY          TAG         IMAGE ID        CREATED         VIRTUAL SIZE
docker.io/centos    centos6     a005304e4e74    5 weeks ago     203.1 MB
```

※Docker hub指定、または自身でDockerレポジトリを作成する方法は割愛

コンテナ作成・起動

- pullしたcentos6のイメージを使用してコンテナを起動する
  ※pull指定なくてもイメージが存在しなければpullしてくる

■コンテナ作成・実行 docker run

- 書式：$ docker run [オプション] [--name 〈コンテナー名〉] 〈イメージ名〉[:〈タグ名〉] [コンテナーで実行するコマンド] [引数]
  -i 標準入力，-t tty（端末デバイス），-p ポートマッピング

・以下はcentos6のイメージをホスト機ポート10080とコンテナ80をマッピングし、yp-tets01という名前で起動しbash接続している

```
# docker run -it -p 10080:80  --name yp-test01 centos:centos6 /bin/bash
```

※loopbackデバイスの使用警告が出力されるが今回は無視

起動した bash でログインされ、プロンプトにコンテナIDホストが表示されている

```
[root@22d1e6f1b5ce /]#
```

dockerコンテナの確認

- CoreOSやAtomのような専用OSではないがdocker hubからpullしたcentosはカスタマイズされている。

■ディスクサイズ

```
[root@22d1e6f1b5ce /]# df -h
Filesystem                                                      Size  Used Avail Use% Mounted on
/dev/mapper/docker-253:0-924185-
5c8d9467f3fbe8a707e515f0c570636b6821e1fe7af943821169e496910726b2  9.8G  502M  8.8G   6% / ●←コンテナパーティション


/dev/vda3            36G  2.2G   32G   7% /etc/resolv.conf ●親パーティション
/dev/vda3            36G  2.2G   32G   7% /etc/hostname ●親パーティション
/dev/vda3            36G  2.2G   32G   7% /etc/hosts ●親パーティション
```

■プロセス

```
[root@22d1e6f1b5ce /]# ps -ef | egrep -v '(grep|ps )'
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 07:11 ?        00:00:00 /bin/bash ●docker run時に指定したbashのみ
```

■rpm

```
[root@22d1e6f1b5ce /]# rpm –qa | wc -l
132
```

コンテナ上でhttpdインストールと確認

■インストール

```
[root@22d1e6f1b5ce /]# yum install httpd -y
[root@22d1e6f1b5ce /]#  cat << EOF > /var/www/html/index.html
> Hello, World!! YP-Study
> EOF
[root@22d1e6f1b5ce /]#  /usr/sbin/httpd
[root@22d1e6f1b5ce /]#  ps –ef
```

作業PCのブラウザから自身の作業インスタンスのURLにアクセスし Hello, World!!が表示されることを確認

- http://"ip addr":10080

コンテナexit後の確認

■bashのexit

```
[root@22d1e6f1b5ce /]# exit
```

■稼働中コンテナ確認

```
# docker ps
```

■コンテナ一覧確認

```
# docker ps –a
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS                      PORTS               NAMES
4686838f4c7a        yapodu:ypst201507      "/bin/sh -c '/etc/in     5 minutes ago       Exited (137) 2 seconds ago                      yp-test01
```

※コンテナ上で動いていた唯一のプロセスbashがexit処理されたのでコンテナは終了した

別プロセスが稼働中であれば稼働し続ける

停止後のコンテナ起動と接続

■コンテナ起動

```
# docker start  yp-test01
```

■稼働中コンテナに接続と確認

```
# docker attach yp-test01 ●←起動している/bin/bashに接続
or
# docker exec -it yp-test01 bash ●←新規にbashを発行し
```

■コンテナプロセス確認

```
[root@22d1e6f1b5ce /]# ps -ef
```

- httpdが動いてない、ブラウザをリロードしエラーとなることを確認
- docker runで指定されたCOMMAND /bin/bashのみが起動時に実行される

稼働中コンテナに追加でCOMMANDを発行

■コンテナからキーバインドでdettach

```
[root@22d1e6f1b5ce /]# [Ctrl+p + Ctrl+q]
※[Ctrl+c]はexit処理
```

※[Ctrl+c]はexit処理

■稼働中コンテナにバックグランドで追加コマンド発行

```
#  docker exec -d yp-test01 /usr/sbin/httpd
```

作業PCのブラウザから自身の作業インスタンスにアクセスしapache welcomeが表示されることを確認

- http://"ip addr":10080

■コンテナプロセス確認

```
# docker attach yp-test01
[root@22d1e6f1b5ce /]# ps –ef
root       60     1  0 04:14 ?        00:00:00 /usr/sbin/httpd
[root@22d1e6f1b5ce /]# exit
```

コンテナのcommit

- 変更したコンテナはコミットすることでimageとして保存できる
- コンテナ作成時にコミットしたimageで実行できる

■コンテナイメージ作成 docker commit

- 書式：$ docker commit "NAMES or CONTAINER ID" "REPOSITORY:TAG"

■先ほどのyp-test01コンテナをレポジトリcentos:yp-commit01でコミットしてイメージ作成

```
# docker ps -a
# docker commit yp-test01 centos:yp-commit01
# docker images
REPOSITORY      TAG             IMAGE ID        CREATED         VIRTUAL SIZE
centos          yp-commit01     2d205eb58ad9    10 seconds ago  277.6 MB
```

コンテナとイメージの削除

- 作成したコンテナとイメージを削除する

■コンテナ削除 docker rm

- 書式：$ docker rm "NAMES or CONTAINER ID"

■コンテナイメージ削除 docker rmi

- 書式：$ docker rmi “IMAGE ID"

```
# docker stop yp-test01
# docker ps -a
# docker rm  yp-test01
# docker ps -a

# docker images
# docker rmi "IMAGE ID"
# docker images
```

Dockerfileの確認

- Dockerコンテナの構成内容を記述しイメージの作成を行うことができる
- Dockerfileを使えば同構成を素早く構築可能
- コンテナの作成はDockerfileを使用することが一般的

■ Dockerfile 確認

```
# cd /var/tmp
# git clone https://github.com/yapodu/hands-on_Docker
# cd hands-on_Docker/FILES
# ls -la
# vim -R Dockerfile
```

Dockerfileからイメージ作成

■Dockerfile build実行

書式：$ docker build –t “REPOSITORY:TAG ”

```
# docker build -t centos:yp-study01 .
# docker images
REPOSITORY        TAG             IMAGE ID        CREATED          VIRTUAL SIZE
centos            yp-study01      c74cc29d531a    2 minutes ago    547.2 MB
docker.io/centos  centos6          a005304e4e74    4 weeks ago      203.1 MB
```

作成されたイメージからコンテナ作成

■fluentd用ディレクトリをhost側に作成

```
# tail -7 td-agent.conf
path /var/log/td-agent/httpd/access.log用ディレクトリをホストに作成

# mkdir /home/yp-study/host-dir01
# chown 499:499 /home/yp-study/host-dir01
```

■作成成したディレクトリをマウントした状態でコンテナ作成・起動

```
# docker run ¥
-v /home/yp-study/host-dir01:/var/log/td-agent/httpd ¥
-it -p 10080:80 --name yp-file01 centos:yp-study01 /bin/bash
[root@df851460ae1a /]# ps -ef
```

ブラウザアクセス&ログ確認

作業PCのブラウザから自身の作業インスタンスのURLにアクセスし apache welcomeが表示されることを確認

- http://"ip addr":10080

■コンテナ確認

```
[root@df851460ae1a /]# tail /var/log/td-agent/httpd/access*
[Ctrl+p + Ctrl+q]
```

■ホスト確認

```
# tail /home/yp-study/host-dir01/access.*
```

ディスク速度確認

コンテナ内のディスクは一段余計なレイヤが入っているためホストと同じノリだと死ねる

■コンテナ接続

```
# docker exec -it yp-file01 bash
[root@df851460ae1a /]# cd /tmp/FIO
```

■ランダム書き込み

```
ホスト領域
[root@df851460ae1a /]# fio host-dir512m-rw.fio
コンテナ内
[root@df851460ae1a /]# fio docker512m-rw.fio
```

ディスク速度確認

■ランダム読み込み

ホスト領域

[root@df851460ae1a /]# fio host-dir512m-rr.fio

コンテナ内

[root@df851460ae1a /]# fio docker512m-rr.fio

■dd書き込み

ホスト領域

[root@df851460ae1a /]# dd if=/dev/zero of=/var/log/td-agent/httpd/dd-w-file bs=1M count=100 oflag=sync

コンテナ内

[root@df851460ae1a /]# dd if=/dev/zero of=/var/log/dd-w-file bs=1M count=100 oflag=sync

■今回構築したイメージをDocker hubにあげてますので、よかったら使用して下さい。

・URL

https://registry.hub.docker.com/u/yapodu/ypst201507/

・使用例

$ sudo docker pull yapodu/ypst201507

- 今後のハンズオン勉強会予定
  - ・ServerSpec/CircleCIで実現するContinuous Integration
    継続的インテグレーションをインフラで
  - ・Google BigQueryとBigdata
    巨大データをGoogleのインフラで処理

ありがとうございました。