# .Deb

銀河系唯一のDebian専門誌

2014 年 05 月 17 日

## 特集：debian と docker.io

# デビアン勉強会

## 目次

# 1 事前課題

野島 貴英

今回の事前課題は以下です:

1. 本日、何の作業をやるかを宣言ください。

この課題に対して提出いただいた内容は以下です。

## 1.1 Koji Hasebe

MAC 上で Debian 作業環境を構築します。
その環境を業務で使えるところまで行けたらと思っています。

## 1.2 dictoss(杉本　典充)

Debian GNU/kFreeBSD をいろいろ試す。

## 1.3 吉野 (yy_y_ja_jp)

- DDTSS
- manpages-ja

## 1.4 wbcchsyn

kpatch を読む
`https://github.com/dynup/kpatch` kpatch は、OS を停止せずに linux kernel にパッチを当てる為のパッチとの事。（開発中）

Linux kernel のパッチなので Gnu Linux でも使用可能なはずだが、RedHat 中心に開発されているとの事なので、Debian 向けのドキュメントや環境が整うには時間がかかりそう。なので、自分で開発中の GitHub を読んでみる。

## 1.5 regonn(Kenta Tanoue)

Debian 初心者なので Debian リファレンスを読んでいきます。（目標は半分の 6 章まで。知らなかったことをどんどんメモしていく）

## 1.6 shinyorke

**課題**　Pythonista として、Debian に慣れる。
**内容**
- 自分で作った Python アプリ (Django) を Debian 上で動かす (Python + Django + MySQL)
- nginx を入れて、Python アプリをリバースプロキシする

**環境**
- ホスト OS: Mac OS X(Mavericks)
- ゲスト OS: Debian ※ virtualbox 上で動作

今回、外向けの公開はしない。

## 1.7 野島　貴英

Debian による immutable infrastracture な環境の実験と試行錯誤。（んでもって、何か見つけたらバグレポ）

## 1.8 まえだこうへい

先月の続き。
- Golang 関係のパッケージ化の続き
- `http://qa.debian.org/developer.php?login=mkouhei@palmtb.net` のバグ潰し＆パッケージアップデート

# 2 Debian Trivia Quiz

野島 貴英

ところで、みなさん Debian 関連の話題においついていますか？Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではははりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容などからです。

問題 1. 先日とあるアーキテクチャが testing から外されました。次のうちのどれでしょう？

☐ A i386
☐ B armel
☐ C sparc

問題 2. 先月 4/26 に debian の安定版がリリースされました。バージョンはいくつでしょう？

☐ A 7.4
☐ B 7.5
☐ C 7.6

# 3 最近の Debian 関連のミーティング報告

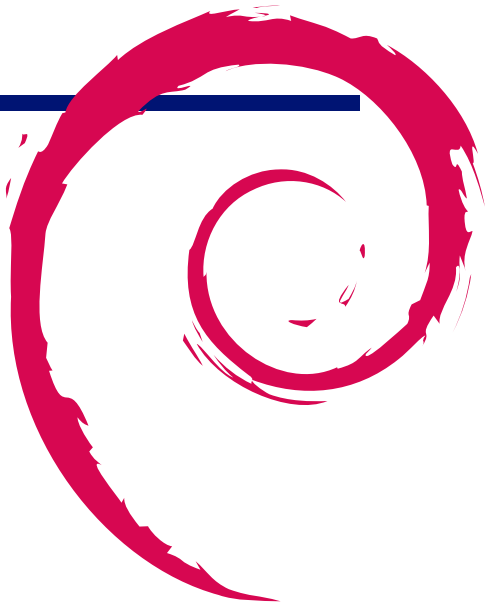
野島 貴英

## 3.1 東京エリア Debian 勉強会 112 回目報告

東京エリア Debian 勉強会 112 回目は (株) スクウェア・エニックスさんで開催されました。10 名の参加者がありました。また、まえださんにより、debian にて go 言語を元にして出来たプログラムのパッケージ化の方法/約束事/留意事項について発表がありました。

また、公式 Debian Develpoer の方がたまたま 3 名と比較的新しい参加者もいらっしゃいましたので、皆でキーサイン会も行いました。

宴会は、「浜の漁師居酒屋 こちらまる特漁業部 新宿靖国通り店」にて行いました。



# 4 debian で docker.io

野島 貴英

## 4.1 はじめに

去年あたりから、linux のコンテナ環境である docker[1] が注目されているようです。

docker は使うとわかるのですが、単に linux 上にコンテナ環境を作成するという機能の他に、aufs を利用してベースの OS のシステムに迅速に変更差分を適用するという動作により、非常に素早くカスタム化されたコンテナ環境の作成・変更・管理が出来ます。

また、変更した内容を docker リポジトリ (`https://index.docker.io`) に登録することにより、docker が動作する環境さえあれば、こちらのリポジトリから全く同じコンテナ環境を作成・動作させることができます。

今回は debian を docker ホストにして docker を使ってみた事について発表します。

## 4.2 今回利用の debian

今回発表で評価した debian は unstable(jessie/sid) となります。

なお、残念ながら安定版の debian wheezy では未だ docker はパッケージ化されていない状況です。本誌を読まれている debian 使いの方は、ぜひこの機会に debian unstable(jessie/sid) にアップグレードしてパッケージから docker をお試しください。

## 4.3 docker 仕組みの簡単なおさらい

docker を図示すると図のようになります。

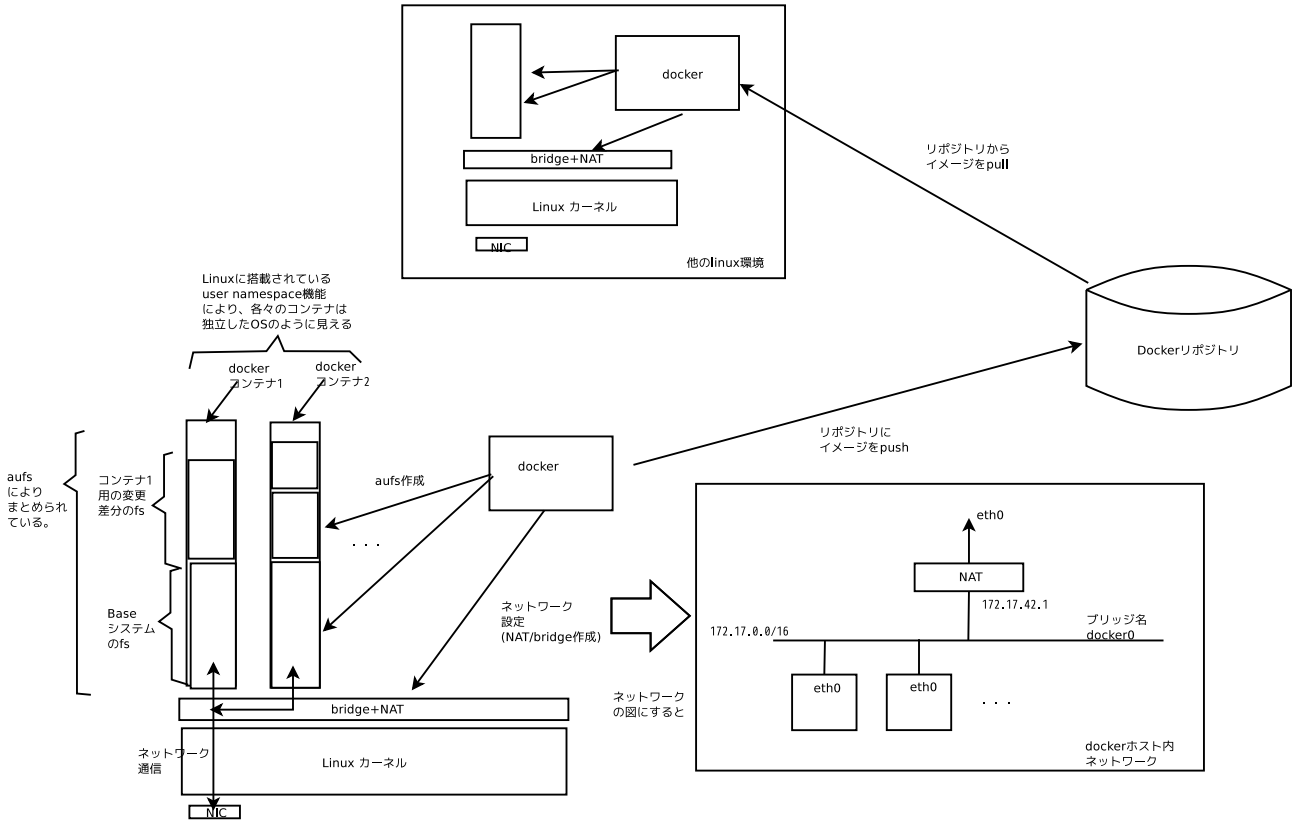



図 1　docker の構造

docker によるコンテナ環境の特徴としては、

- 非常に素早い起動、停止が可能です。
- base のファイルシステムに、aufs による差分ベースのファイルシステム内容の適用を行うため、非常に簡単にコンテナの変更・破棄が可能です。
- 構成管理を Docker リポジトリで行える。（注：一見 git のような使い方に見えますが、git を使って作られているわけではありませんので、git ほどの高機能で柔軟な変更管理はできません）
- Docker リポジトリが参照でき、docker が動く環境であれば、docker ホストの linux ディストリビューションが異なる環境でも同じ構成内容を持つ docker コンテナを動作できます。

となります。

docker ホストの内部のネットワークは、docker により、ブリッジ docker0 が作成され、docker ホストの eth0 へ NAT されて接続されます。そのため、docker ホスト外からのコンテナ側のサービスへのアクセスは、DNAT して docker ホスト側のポートへ引き出すことにより行われます。

## 4.4　手元の debian 機材で試す

以下の手順で簡単に試すことが出来ます。

Step 1.　インターネットに接続できている debian unstable 環境を用意します。

Step 2.　ip forwarding ができるようにしておきます。

```
$ sudo vi /etc/sysctl.d/ip-forward.conf
----ここから-----
net.ipv4.ip_forward=1
----ここまでを記載-----
$ sudo sysctl -p /etc/sysctl.d/ip-forward.conf
```

Step 3. docker を導入します。なお、debian パッケージの docker は、docker.io というパッケージ名であり、コマンドも docker ではなく、**docker.io** という名前になります。（以降本コマンドを docker.io コマンドと呼びます）

```
$ sudo aptitude install docker.io
$ docker.io
Usage: docker [OPTIONS] COMMAND [arg...]
 -H=[unix:///var/run/docker.sock]: tcp://host:port to bind/connect to or unix://path/to/socket to use

A self-sufficient runtime for linux containers.

Commands:
    attach    Attach to a running container
... 中略 (docker.io コマンドの help が出る)...
```

Step 3. グループ docker に操作者のログイン ID を追加し、ログインしなおします。こうすることで docker.io コマンドによる操作を一般ユーザ権限で操作できるようになります。

```
$ sudo useradd YOUR-ID docker
$ exit
login: YOUR-ID
Password: xxxxx
$
```

Step 4. 試しにコンテナとして debian-sid(jessie-sid) を docker で動かしてみます。

```
$ docker.io run -t -i -h debian-sid1 debian:sid
Unable to find image 'debian:sid' locally
Pulling repository debian
1cda8535c670: Download complete
511136ea3c5a: Download complete
0ed2a4d77969: Download complete
root@debian-sid:/#  <-- 起動した docker コンテナの debian sid。
```

Step 5. 作った docker コンテナの中で ps を導入し、process を見てみます。

```
root@debian-sid:/# apt-get install procps
root@debian-sid2:/# hash
hits        command
   2        /sbin/ip
   1        /usr/bin/apt-get
root@debian-sid2:/# ps -auxww
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  18016  1932 ?        Ss   22:17   0:00 /bin/bash
root       118  0.0  0.0  17488  1136 ?        R+   22:26   0:00 ps -auxww
root@debian-sid2:/#
```

見るとわかるとおり、init の代わりに/bin/bash が PID=1 で動作している状態です。

Step 6. Ctrl+p Ctrl+q を連続で打ち込むと、docker コンテナの shell から抜けます。なお、exit を実行すると、PID=1 の/bin/bash が終了するため、shutdown を実行したことと等価となり、コンテナが終了します。

```
root@debian-sid2:/# ... ここで Ctrl+p Ctrl+q する...
$ <- docker ホストのプロンプトが帰ってくる
$ docker.io ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
20fa6020f73b        debian:sid          /bin/bash           12 minutes ago      Up 12 minutes                           evil_euclid
(コンテナ ID: 20fa6020f73b が動作中であることを示す)
$ docker.io attach 20fa6020f73b (<--再び debian-sid に接続)
<リターンキー押す>
root@debian-sid2:/# <-再びコンテナの shell プロンプト。
root@debian-sid2:/# exit
$ docker.io ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
$
(docker.io ps をとって稼働中のコンテナを確認したが、コンテナが終了してしまっているため、動作中のコンテナ ID が表示されない↑)
```

## 4.5 docker のネットワークの癖について

docker は、docker ホストの起動時に docker がデーモンモードで動作しており、docker コマンドで指令を送ってコンテナの管理をします。ここで、docker ホストのネットワークを、デーモンモードの docker が起動時にセットアップしています。

ここで、例えば docker ホストがモバイル PC であった場合、ppp とかを後から起動するなどして、ネットワークの設定が docker デーモンがセットアップした状態と異なってしまうことがあります（特に NAT 周り。）

この場合、docker.io でコンテナを作成しようとしても、作成途中のコンテナ側からネットワークが外部へのネットワーク通信が出来ず、コンテナ作成が途中で停止する現象が起きることがあります。

この場合は、ppp などの通信をつないだ状態で、再度、docker デーモンを再起動すると、再度ネットワークがセットアップされ、問題が解決します。

```
$ pon xxxx <-- ppp を起動などして NAT が ppp の定義で書き換えられてしまう。
$ docker.io run -t -i -h debian-sid1 debian:sid
Unable to find image 'debian:sid' locally
Pulling repository debian
1cda8535c670: Download complete
... ここでハングアップしてしまう...
(Ctrl+C で停止させる)
$ sudo systemctl restart docker.io.service
(docker デーモンの再起動が行われる)
$ docker.io run -t -i -h debian-sid1 debian:sid
Unable to find image 'debian:sid' locally
Pulling repository debian
1cda8535c670: Download complete
511136ea3c5a: Download complete
0ed2a4d77969: Download complete
root@debian-sid:/#  <-今度はコンテナが無事起動する。
```

## 4.6 GCE で docker

手元の debian 機で docker を動かすだけでは物足りないかと思います。今年は immutable infrastructure[2] の年ですので、早速パブリッククラウド環境でも動かしてみることにします。

Google Compute Engine(GCE) でも docker は動くとのことですので、試してみます。

Step 1. お手元の debian sid で chromium などのブラウザを使い、google アカウントで google にログインしておきます。

Step 2. `https://developers.google.com/` の google デベロッパサイトから、Google Cloud にサインアップします。

注意：巷の blog などでは、`https://developers.google.com/compute/docs/signup` が案内されていますが、評価期間は終わっているため、こちらからサインアップする必要はありません。長い英語のアンケートに英語で答えさせられるなどの苦行が待っているため、こちらはおすすめしません。

Step 3. プロジェクトを作成するメニューが最初に現れますので、適当にプロジェクトを作成します。

Project Name: docker evalation

Project ID: docker-evaluation-test-001

Step 4. billing(支払い) メニューになるので、支払いの情報を記載します。

Step 5. 特に折り返しの電話などなく、GCE のメニューになります。

Step 6. お手元の debian unstable 機材に、google-cloud-sdk をダウンロードします。

```
$ mkdir google-sdk;cd google-sdk
$ wget https://dl.google.com/dl/cloudsdk/release/google-cloud-sdk.tar.gz
$ tar xzf google-cloud-sdk.tar.gz
$ cd google-cloud-sdk
$ ./install.sh
 ... カレントディレクトリにインストールが開始...
$ cd ..
$ zsh の場合：source google-cloud-sdk/path.zsh.inc;rehash
  bash の場合:source google-cloud-sdk/path.bash.inc;hash
```

Step 7. sdk から認証を行います。

```
$ gcloud auth login
... ここで、chromium などが開き、sdk が google のアカウントにアクセスしてよいかの
    許可を求められるので、「承諾」を押下...
$
```

Step 7. GCE の instance を作ります。ここでは、料金の最も安い f1-micro で、ネットワーク的に近いアジア地域に、debian wheezy のイメージで作ります。3 秒ぐらいで完了します。

```
$ gcutil addinstance docker-test001 --project=docker-evaluation-test-001 --image=debian-7 --machine_type=f1-micro \
--zone=asia-east1-a --wait_until_running --auto_delete_boot_disk
INFO: Resolved debian-7 to projects/debian-cloud/global/images/debian-7-wheezy-v20140415
INFO: Waiting for insert of instance docker-test001. Sleeping for 3s.
INFO: Waiting for insert of instance docker-test001. Sleeping for 3s.
INFO: Waiting for insert of instance docker-test001. Sleeping for 3s.
INFO: Waiting for insert of instance docker-test001. Sleeping for 3s.
INFO: Ensuring docker-test001 is running.  Will wait to start for: 240 seconds.
  ... 中略...
 $
```

Step 8. 作ったインスタンスにログインして、早速 debian unstable にアップグレードします。

```
$ gcutil ssh --project=docker-evaluation-test-001 docker-test001
yours@docker-test001:~$ sudo vi /etc/apt/source.list
--------------全部消して以下に置き換え--------------------
deb     http://gce_debian_mirror.storage.googleapis.com/ sid         main contrib non-free
deb-src http://gce_debian_mirror.storage.googleapis.com/ sid         main contrib non-free
deb     http://http.debian.net/debian sid         main contrib non-free
deb-src http://http.debian.net/debian sid         main contrib non-free
--------------全部消して以下に置き換えここまで--------------------
yours@docker-test001:~$ sudo apt-get update
yours@docker-test001:~$ sudo apt-get dist-upgrade
... ここで、grub はインストールせずに進めるを選択...
yours@docker-test001:~$ dpkg-reconfigure locales
...en_US.utf-8 と、ja_JP.utf-8 をメニューで選択して有効にする。
yours@docker-test001:~$ exit
$ gcutil restart --project=docker-evaluation-test-001 docker-test001
... リスタートしてしばらく待つ...
$ gcutil ssh --project=docker-evaluation-test-001 docker-test001
yours@docker-test001:~$ uname -a
Linux docker-test001 3.14-1-amd64 #1 SMP Debian 3.14.4-1 (2014-05-13) x86_64 GNU/Linux
yours@docker-test001:~$ cat /etc/debian_version
jessie/sid
```

Step 9. 早速 GCE インスタンスに docker.io パッケージを導入し、下準備。

```
yours@docker-test001:~$ sudo vi /etc/sysctl.d/ip4forward.conf
-----ここから--------
net.ipv4.ip_forward = 1
-----ここまで--------
yours@docker-test001:~$ sudo sysctl -p /etc/sysctl.d/ip4forward.conf
net.ipv4.ip_forward = 1
yours@docker-test001:~$ aptitude install docker.io
yours@docker-test001:~$ sudo useradd yours docker
yours@docker-test001:~$ exit
... 再度ログインしなおし...
$ gcutil ssh --project=docker-evaluation-test-001 docker-test001
yours@docker-test001:~$
```

Step 10. docker を動かしてみる。

```
yours@docker-test001:~$   docker.io run -t -i -h debian-sid debian:sid
Unable to find image 'debian:sid' locally
Pulling repository debian
1cda8535c670: Pulling image (sid) from debian, endpoint: https://cdn-registry-1.docker.io/v1cda8535c670: Download complete
511136ea3c5a: Download complete
0ed2a4d77969: Download complete
root@debian-sid:#   <---無事動作
```

GCE でも非常に簡単に動作しました。

### 4.7 終わりに

docker を debian で動かす事について述べました。見ての通り非常に簡単に動かすことができます。また、docker は docker コンテナの起動・停止・再作成が非常に早く、軽快に使うことができます。今は docker は勢いがあるので、皆さんもぜひ評価されてはいかがでしょうか？

## 参考文献


[1] “Docker: the Linux container engine”, https://www.docker.io/

[2] publickey,「Immutable Infrastructure はアプリケーションのアーキテクチャを変えていく」, http://www.publickey1.jp/blog/14/immutable_infrastructure_1.html

**Debian 勉強会資料**

2014 年 05 月 17 日　　初版第 1 刷発行
東京エリア Debian 勉強会 （編集・印刷・発行）

---