

```
git clone https://github.com/atomiyama/docker-iruby.git
```

はじめてのDocker

Akifumi Tomiyama

TL;DR

- Docker is なに？
- インストール
- デモ: iRubyの環境作ってみる
- 所感

Docker is なに？

調べてみました

Google

docker とは

すべて

画像

ニュース

動画

ショッピング

もっと見る

設定

ツール

約 810,000 件 (0.40 秒)

第1回 Dockerとは：超入門Docker - @IT

www.atmarkit.co.jp > Windows Server Insider >

2017/01/30 - Dockerの概要を知るための超入門連載（全4回）。Dockerとは何か、コンテナとは何か、従来のハードウェアエミュレーション型の仮想化とはどう違うのかなどについてまとめておく。

Docker入門（第一回）～Dockerとは何か、何が良いのか～ | さくらのナ...

https://knowledge.sakura.ad.jp/13265/ >

2018/02/05 - みなさん、こんにちは。Acroquest Technology (アクロクエストテクノロジー) の横山 仁 (よこやま じん) と申します。主に仕事ではインフラ関係やDevOps推進に向けた活動などに携わっています。最近では、今回の記事の内容でもあるDockerを使うこともかなり増えてきて、他にもAnsibleであったり、CIツールのJenkinsなども使って、自動化や開発環境の整備などを全行っています。この度、さくらのナレッジでDockerの入門記事の連載をさせていただくことになりました。よろしくお願いします。

【入門】Dockerとは？使い方と基本コマンドを分かりやすく解説します ...

https://www.kagoya.jp/howto/rentalserver/docker/ >

2017/08/21 - Docker(ドッカー)とはDocker Inc.(旧DotCloud社)によって開発されたコンテナ仮想化ツールです。アプリ操作をコンテナに分けて実行することや、別のOS上にコンテナを移しても動作するようになります。ここでは、Dockerの特徴や、使い方、基本コマンドについて分かりやすく解説します。

Docker - ウィキペディア

https://ja.wikipedia.org/wiki/Docker >

Docker（ドッカー）はコンテナ型の仮想化環境を提供するオープンソースソフトウェアである。VMware製品などの完全仮想化を行うハイパーバイザ型製品と比べて、ディスク使用量は少なく、仮想環境（インスタンス）作成や起動は速く、性能劣化がほとんどないという利点を持つ。目次, [非表示], 1 主な利点, 1.1 資源の効率化, 1.2 アプリ実行環境構築の容易さ, 1.3 設定間違いなどの不可逆的操作の廃棄の容易さ, 2 評価, 3 技術的な特徴, 3.1 差分管理, 3.2 Dockerfile, 4 エコシステム, 4.1 Linux以外でのサポート, 4.2 ...

Dockerとは4つの特徴と基本的な使い方 | フリエン



Docker

ドッカー

ソフトウェア

Dockerはコンテナ型の仮想化環境を提供するオープンソースソフトウェアである。VMware製品などの完全仮想化を行うハイパーバイザ型製品と比べて、ディスク使用量は少なく、仮想環境 作成や起動は速く、性能劣化がほとんどないという利点を持つ。 [ウィキペディア](#)

使用言語: [Go](#)

他の人はこちらも検索

他 15 件以上を表示


kubernetes
Kubernetes


Jenkins


Git




OpenStack


Redis

[フィードバック](#)

公式ページ




 [What is Docker?](#) [Product](#) [Community](#) [Support](#)  [Create Docker ID](#) [Sign In](#)

WHAT IS DOCKER

Docker is the world's leading software containerization platform.

[TRY DOCKER ENTERPRISE EDITION](#) [LEARN MORE](#)

 Join us in San Francisco
dockercon
2015
register now

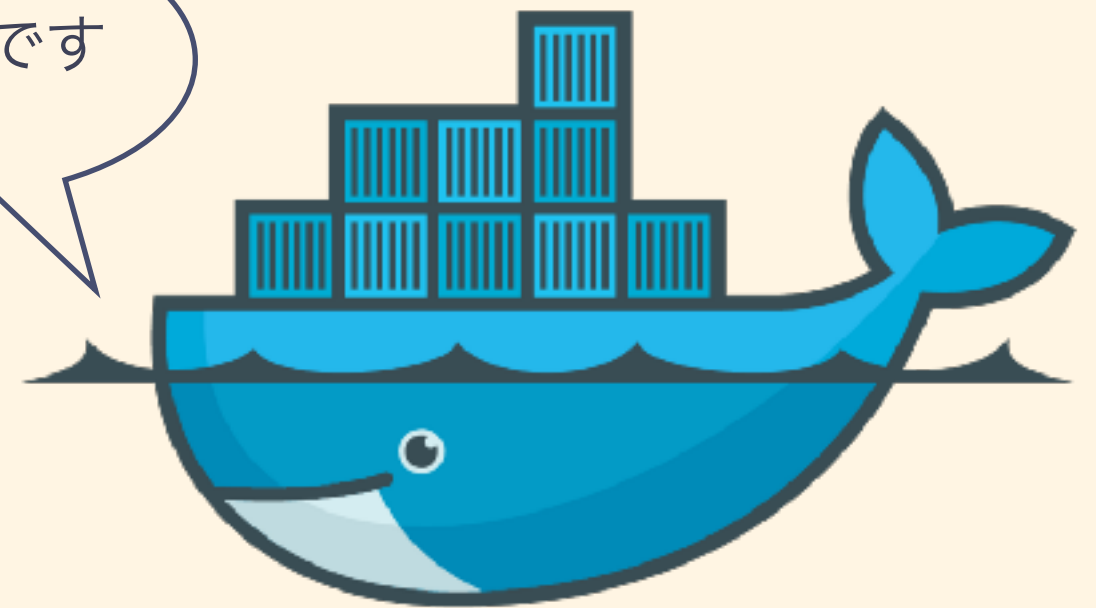
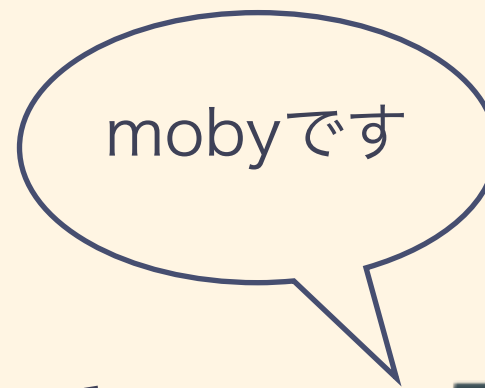
[Overview](#) [Platform](#) [Use Cases](#)

OVERVIEW

Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud. Today's businesses are under pressure to digitally transform but are constrained by existing applications and infrastructure while rationalizing an increasingly diverse portfolio of clouds, datacenters and application architectures. Docker enables true independence between applications and infrastructure and developers and IT ops to unlock their potential and creates a model for better collaboration and innovation.

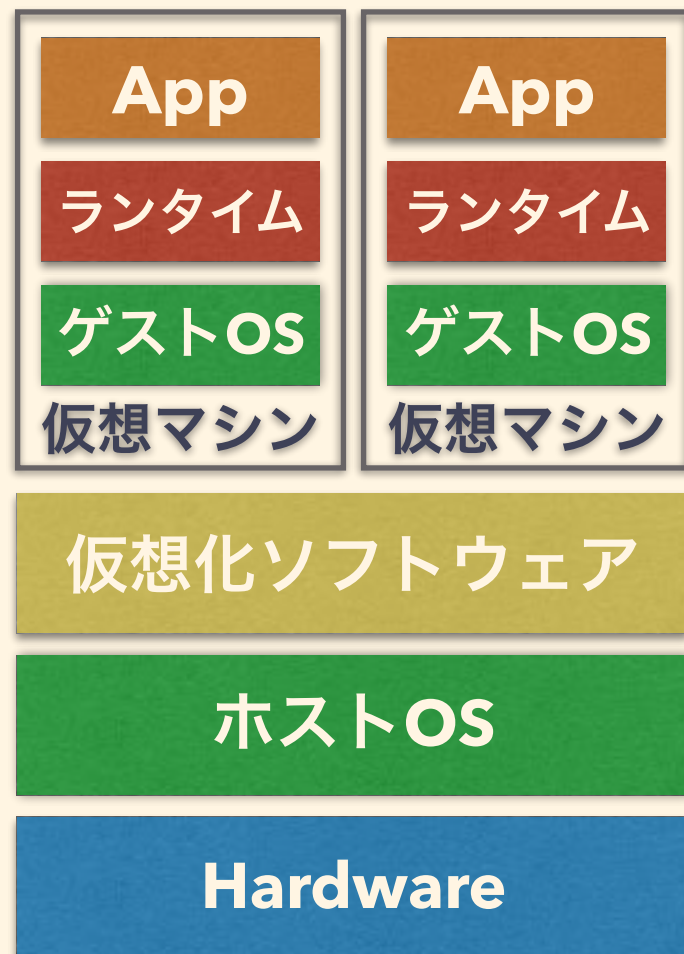
**Dockerはコンテナ型の仮想化環境を提供する
オープンソースソフトウェアである。(wikipedia)**

**VMwareとかVirtualboxと
同じようなものじゃないの？**



docker

違います！！



ホスト型

vmware[®]
WorkstationPlayer

VirtualBox



ハイパーバイザ型

vmware[®]
Infrastructure

Microsoft Hyper-V



コンテナ型

 docker

メリット

- 実行環境のファイルシステム化
ライブラリなどをDockerfileイメージ化
→レジストリで管理

- 再現性、再利用性

DockerEngineが動く環境であればどのような環境でもコンテナとして
実行可能

- 軽量

従来の仮想化技術に比べてOSを持たないため必要なリソースが少なく、
高密度化が可能



Google Container Registry



AWS Elastic Container Registry



GitLab Container Registry



DockerHub

デメリット

- ディストリビューションによる制約

Linux以外のカーネル上でLinuxコンテナは動作しない

∴ ホストOSのカーネルを使って動作する

→ Docker for windowsはHyper-V

→ Docker for macはHyperKit の仮想環境上にDocker

- 学習コストが高い

Dockerfile, docker-cli, docker-compose,
Docker Swarm, Kubernetes...etc

どんなところで使われているの？

- マイクロサービス (ex: ポケモンGO)
- CI (継続的インテグレーション)
- CD (継続的デリバリー)
- 開発基盤 (Infrastructure as Code)



Jenkins



Travis CI

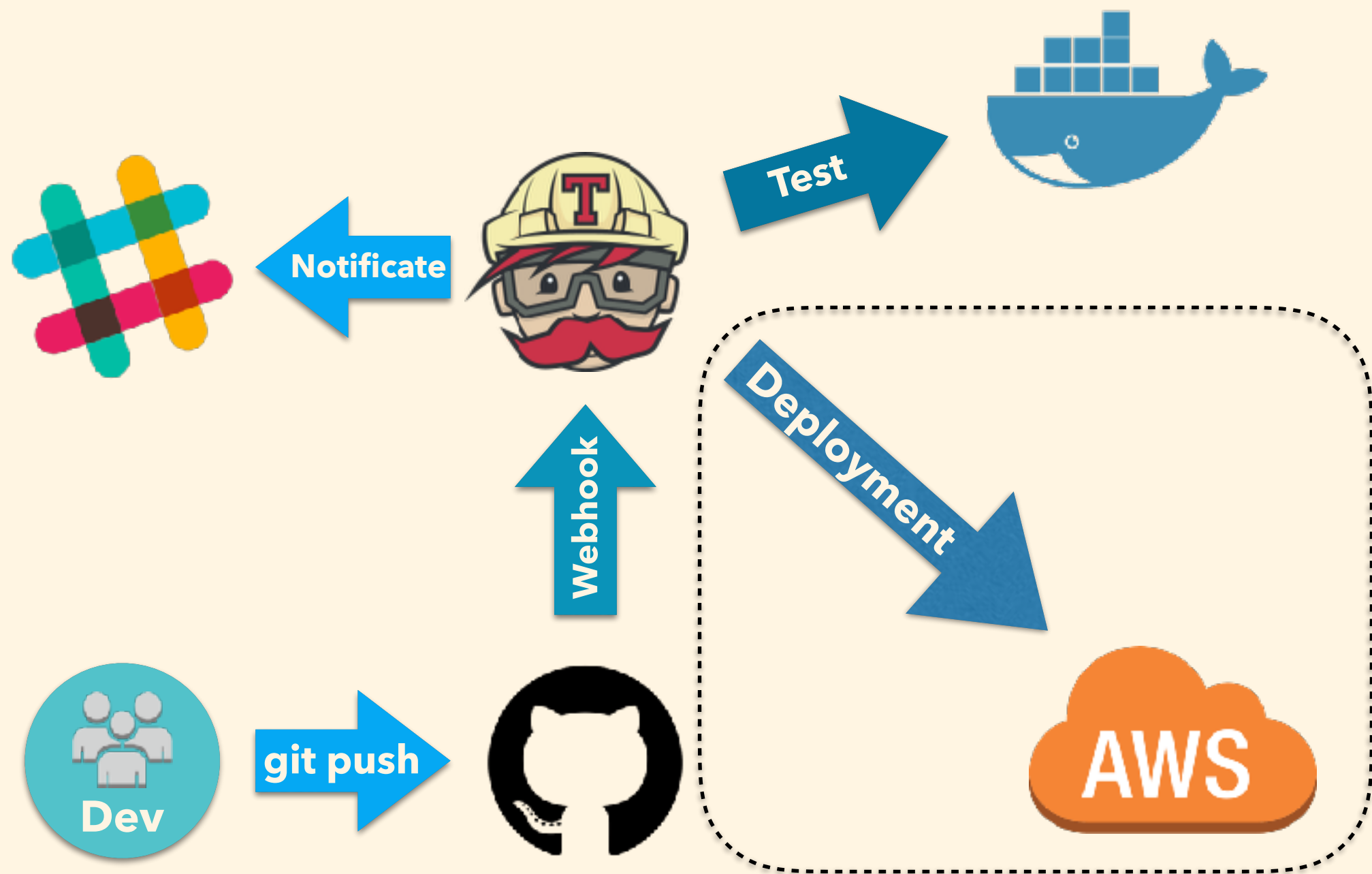


circle**ci**



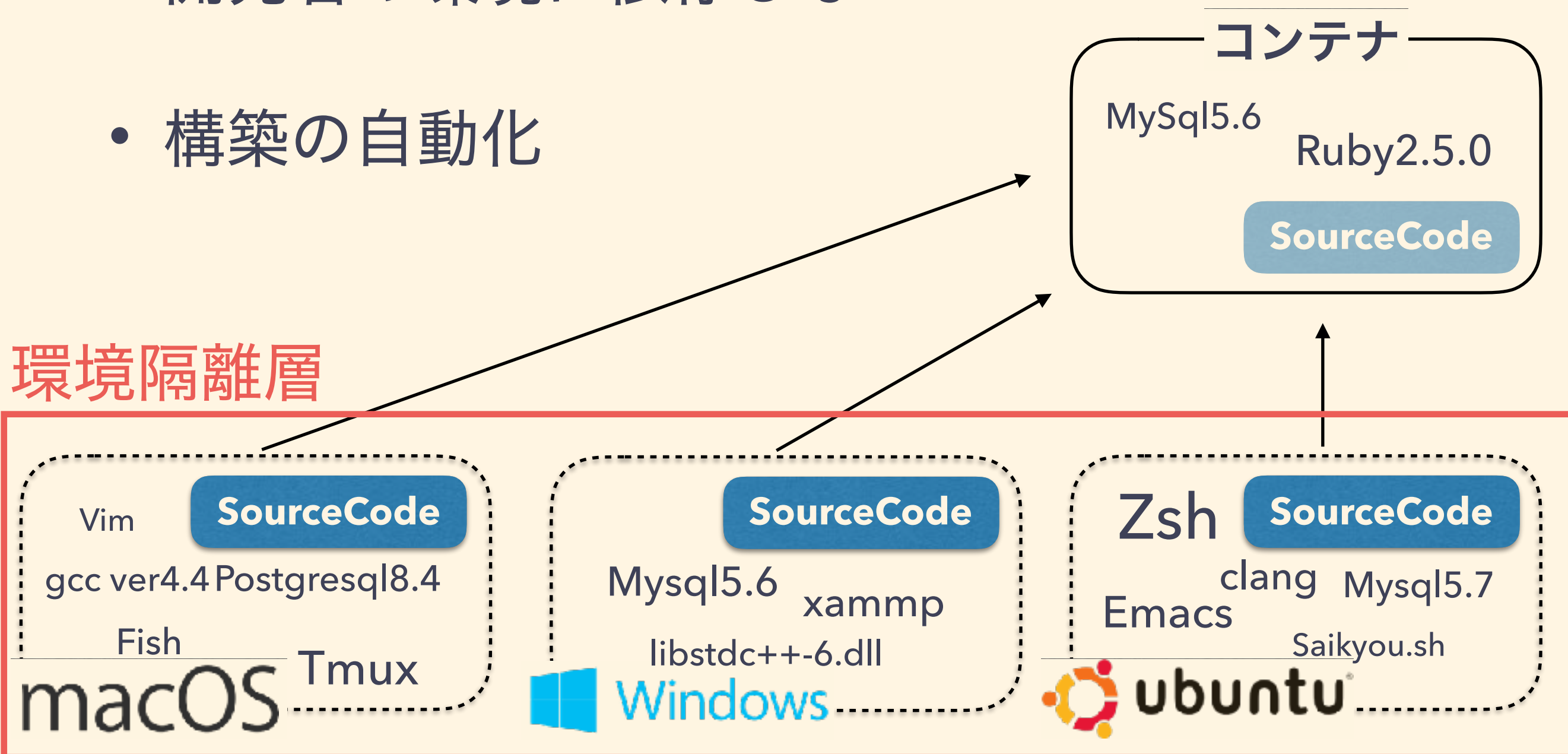
AppVeyor

CI/CDライフサイクル



開発基盤としてのDocker

- 開発者の環境に依存しない
- 構築の自動化



使ってみましょう！

まずインストール

インストール手順

```
# 今回使用するものが入ったリポジトリ
```

```
$ git clone https://github.com/atomiyama/docker-iruby.git
```

```
$ cd docker-iruby
```

<<comments

- ・ Windows

- "Docker Community Edition for Windows"で検索

- ・ macOS

- "Docker Community Edition for Mac"で検索

- ・ Linux

- リポジトリにインストール手順を記載したmdがあります.

comments

```
sudo docker run hello-world
```

Hello Docker!!



```
docker-iruby git:(master!?) 🐋 v18.03.0-ce  
→ docker run --rm hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
9bb5a5d4561a: Pull complete  
Digest: sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77  
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

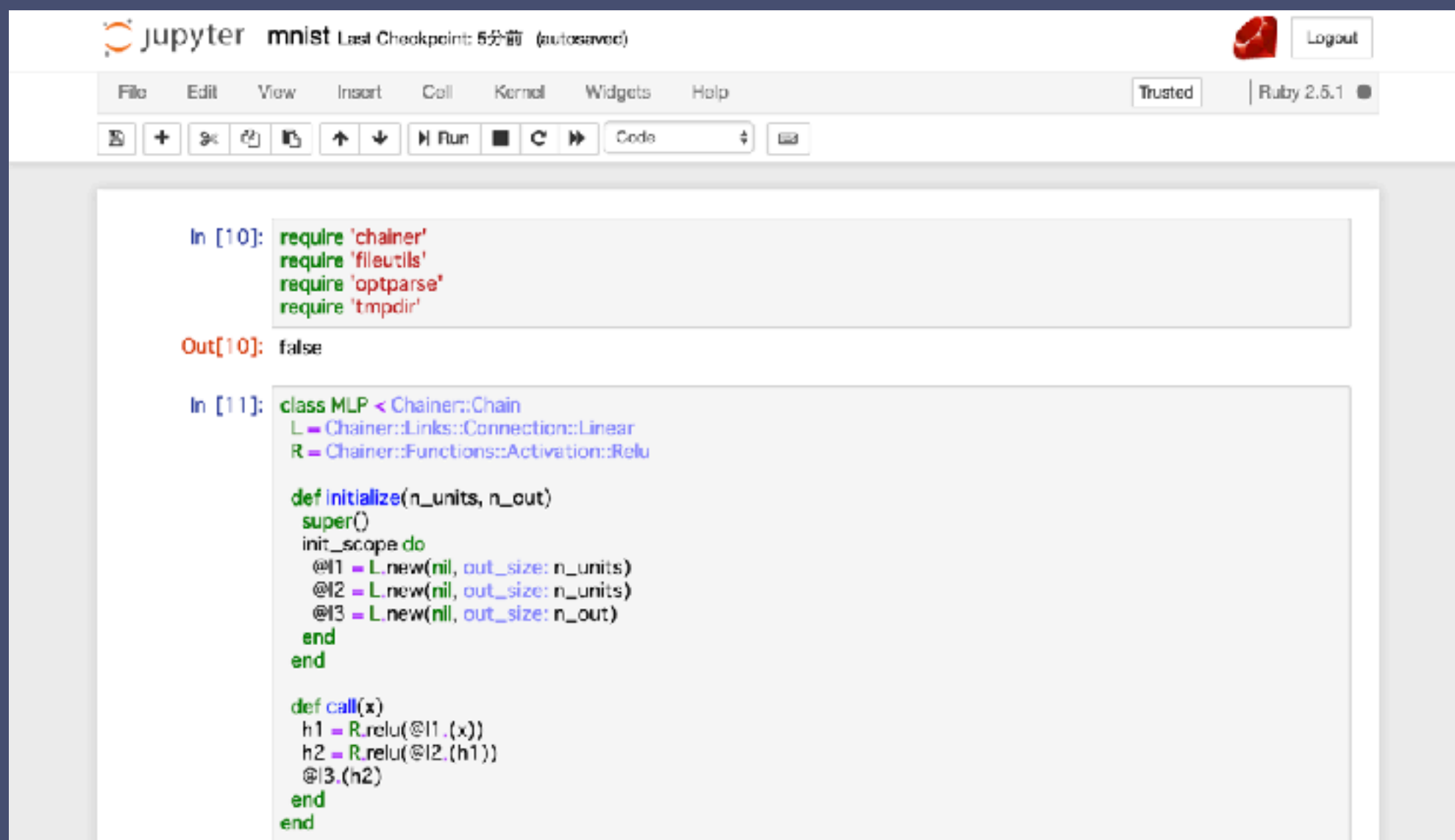
To try something more ambitious, you can run an Ubuntu container with:
\$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
<https://hub.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/engine/userguide/>

iRubyの環境を作る

コンテナ上でiRubyを使えるようにする



The screenshot shows a JupyterLab interface with a notebook titled 'mnist'. The top bar includes the Jupyter logo, the notebook name 'mnist', and a status message 'Last Checkpoint: 5分前 (autosaved)'. On the right, there is a Ruby logo and a 'Logout' button. The main menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The notebook area displays two code cells. The first cell, labeled 'In [10]:', contains Ruby code to require 'chainer', 'fileutils', 'optparse', and 'tmpdir'. The output for this cell is 'Out[10]: false'. The second cell, labeled 'In [11]:', contains Ruby code to define an 'MLP' class that inherits from 'Chainer::Chain'. The class has an 'initialize' method that sets up three linear layers and a 'call' method that applies ReLU activation to each layer's output.

```
In [10]: require 'chainer'
         require 'fileutils'
         require 'optparse'
         require 'tmpdir'

Out[10]: false

In [11]: class MLP < Chainer::Chain
         L = Chainer::Links::Connection::Linear
         R = Chainer::Functions::Activation::Relu

         def initialize(n_units, n_out)
           super()
           init_scope do
             @l1 = L.new(nil, out_size: n_units)
             @l2 = L.new(nil, out_size: n_units)
             @l3 = L.new(nil, out_size: n_out)
           end
         end

         def call(x)
           h1 = R.relu(@l1.(x))
           h2 = R.relu(@l2.(h1))
           @l3.(h2)
         end
       end
```

docker ps, rm

コンテナ一覧, 削除

OPTIONS

-a [-all] 全てのコンテナを表示(デフォルトでは非稼働コンテナは非表示)

-q [-quiet] コンテナIDのみ表示

docker-iruby git:(master?)  v18.03.0-ce

→ docker ps -a #全てのコンテナを表示

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
23902fd90107	hello-world	"/hello"	29 minutes ago	Exited (0) 29
minutes ago		nervous_shockley		

docker-iruby git:(master?)  v18.03.0-ce

→ docker rm nervous_shockley #コンテナの削除

nervous_shockley

docker-iruby git:(master?)  v18.03.0-ce

→ docker ps -a

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			



docker images, rmi

イメージファイル一覧の出力, 削除

OPTIONS

-a [-all] 全てのイメージを出力

-q [-quiet] イメージIDのみ出力

docker-iruby git:(master!?)  v18.03.0-ce

→ docker images **#イメージ一覧**

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	e38bc07ac18e	7 days ago	1.85kB

docker-iruby git:(master!?)  v18.03.0-ce

→ docker rmi hello-world **#イメージの削除**

Untagged: hello-world:latest

Untagged: hello-world@sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77

Deleted: sha256:e38bc07ac18ee64e6d59cf2eafcdddf9cec2364dfe129fe0af75f1b0194e0c96

Deleted: sha256:2b8cbd0846c5aeaa7265323e7cf085779eaf244ccbdd982c4931aef9be0d2faf

docker-iruby git:(master!?)  v18.03.0-ce

→ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

docker pull

Dockerイメージの取り込み

```
# Ruby2.5.1のイメージをダウンロード
```

```
docker-iruby git:(master)  v18.03.0-ce
```

```
→ docker pull ruby:2.5.1 #ruby2.5.1のイメージをpull
```

```
2.5.1: Pulling from library/ruby
```

```
c73ab1c6897b: Pull complete
```

```
1ab373b3deae: Pull complete
```

```
b542772b4177: Pull complete
```

```
57c8de432dbe: Pull complete
```

```
1785850988c5: Pull complete
```

```
953e617a9c21: Pull complete
```

```
09400a4d0988: Pull complete
```

```
0775b59c37c3: Pull complete
```

```
Digest: sha256:f70114f7b5901a84de89fd9ee93d41f8ed9ea5d9efee6f37e54987d331e3f9a5
```

```
docker-iruby git:(master!?)  v18.03.0-ce
```

```
→ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ruby	2.5.1	1624ebb80e3e	3 weeks ago	863MB

docker run

コンテナを実行

```
# USAGE
docker run [オプション] イメージ [コマンド] [引数...]

# OPTION
-i ,[-interactive] # コンテナのSTDINにアタッチ
-t ,[-tty]          # 疑似ターミナルを割り当て
                    [-rm]          # 終了時コンテナを削除
                    [-name]        # コンテナの名前を指定
-d ,[-detach]       # バックグラウンドで実行
-v ,[-volume]        # [ホスト側の絶対パス]:[コンテナ側の絶対パス]
-p ,[-publish]       # [外部からアクセスされるポート]:[コンテナ側のポート]
```

```
docker-iruby git:(master?) 🐳 v18.03.0-ce
→ docker run -it --rm --name tb ruby:2.5.1
irb(main):001:0>
```

```
docker-iruby git:(master?) 🐳 v18.03.0-ce
→ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

docker build

コードからイメージをビルド

```
# USAGE
docker build [オプション] パス | URL | -
```

```
# OPTIONS
    [-force-rm]    #常に中間コンテナを削除
    [-rm]          #中間コンテナを削除
    -t ,[-tag]      #[イメージ名:タグ]
```

```
docker-iruby git:(master?) 🐳 v18.03.0-ce
→cd docker
```

```
docker git:(master?) 🐳 v18.03.0-ce
→docker build . -force-rm -t docker-iruby
```

Dockerfile

イメージを生成するときに使うファイル

```
FROM ruby:2.5.1

RUN apt-get update -qq \
    && apt-get install -y git \
    libtool \
    libffi-dev \
    ruby-dev \
    make \
    libzmq3-dev \
    libczmq-dev \
    python3-pip

RUN python3 -m pip install jupyter \
    && gem install cztop iruby red-chainer \
    && iruby register --force

RUN mkdir /work
WORKDIR /work

ADD . /work
EXPOSE 8080

CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8080", "--allow-root"]
```

docker git:(master✕?)  v18.03.0-ce

→ docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker-iruby	latest	c1ce23748312	14 seconds ago	1.13GB
ruby	2.5.1	1624ebb80e3e	3 weeks ago	863MB

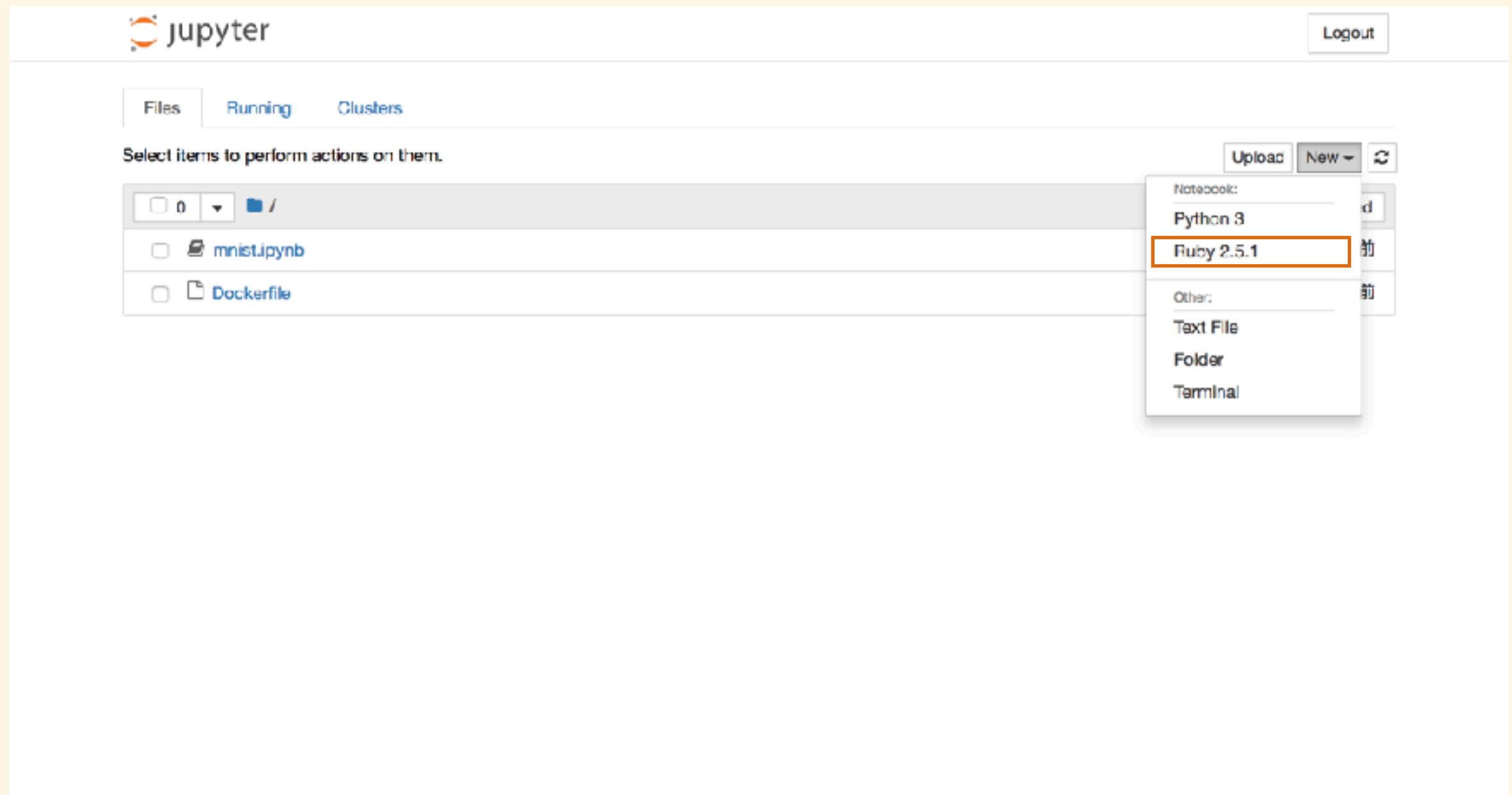
docker run!!!

```
docker git:(masterx?) 🐳 v18.03.0-ce  
→ docker run --rm -p 8080:8080 -v $PWD:/work docker-iruby:latest  
[I 08:01:35.688 NotebookApp] Copying /root/.ipython/kernels -> /root/.local/share/  
jupyter/kernels  
[I 08:01:35.703 NotebookApp] Writing notebook server cookie secret to /root/.local/share/  
jupyter/runtime/notebook_cookie_secret  
[I 08:01:35.920 NotebookApp] Serving notebooks from local directory: /work  
[I 08:01:35.921 NotebookApp] 0 active kernels  
[I 08:01:35.921 NotebookApp] The Jupyter Notebook is running at:  
[I 08:01:35.922 NotebookApp] http://0.0.0.0:8080/?  
token=7b03ce71ce6a1273c063c7d9d8d918e42d7b1c125cd96a47  
[I 08:01:35.922 NotebookApp] Use Control-C to stop this server and shut down all kernels  
(twice to skip confirmation).  
[W 08:01:35.923 NotebookApp] No web browser found: could not locate runnable browser.  
[C 08:01:35.924 NotebookApp]
```

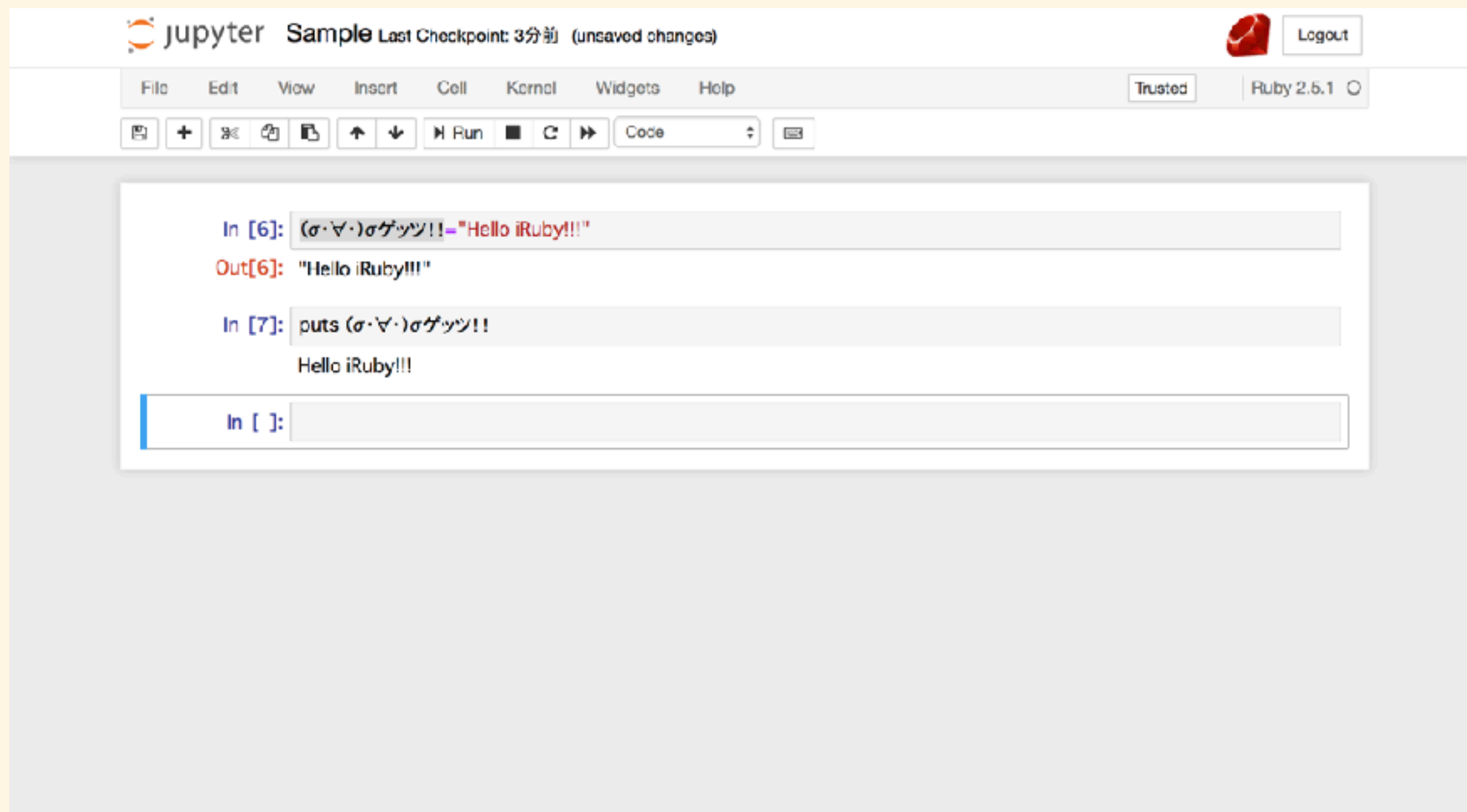
Copy/paste this URL into your browser when you connect for the first time,
to login with a token:

<http://0.0.0.0:8080/?token=7b03ce71ce6a1273c063c7d9d8d918e42d7b1c125cd96a47>

ノートブックの作成



便利！！！！

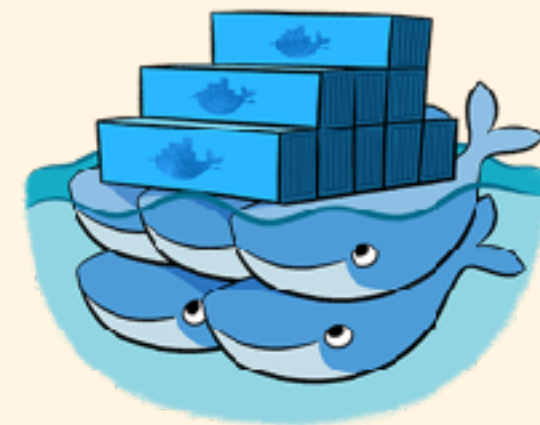


The screenshot displays the JupyterLab web interface. At the top, the header shows the Jupyter logo, the name 'Sample', and a status message 'Last Checkpoint: 3分前 (unsaved changes)'. On the right of the header is a red Ruby logo and a 'Logout' button. Below the header is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are two buttons: 'Trusted' and 'Ruby 2.5.1'. Below the menu bar is a toolbar with icons for file operations (save, new, open, close), navigation (up, down), execution (run, stop, restart), and a dropdown menu currently set to 'Code'. The main workspace contains a code cell with the following content:

```
In [6]: (σ・∀・)σゲッツ!!-"Hello iRuby!!!"  
Out[6]: "Hello iRuby!!!"  
  
In [7]: puts (σ・∀・)σゲッツ!!  
Hello iRuby!!!  
  
In [ ]:
```

所感

コンテナ化≠仮想化



- Dockerと従来の仮想化は別物
従来の仮想環境の代替にはならない
→アドホックな場面では有用
→本番環境での事例はまだ少なそう

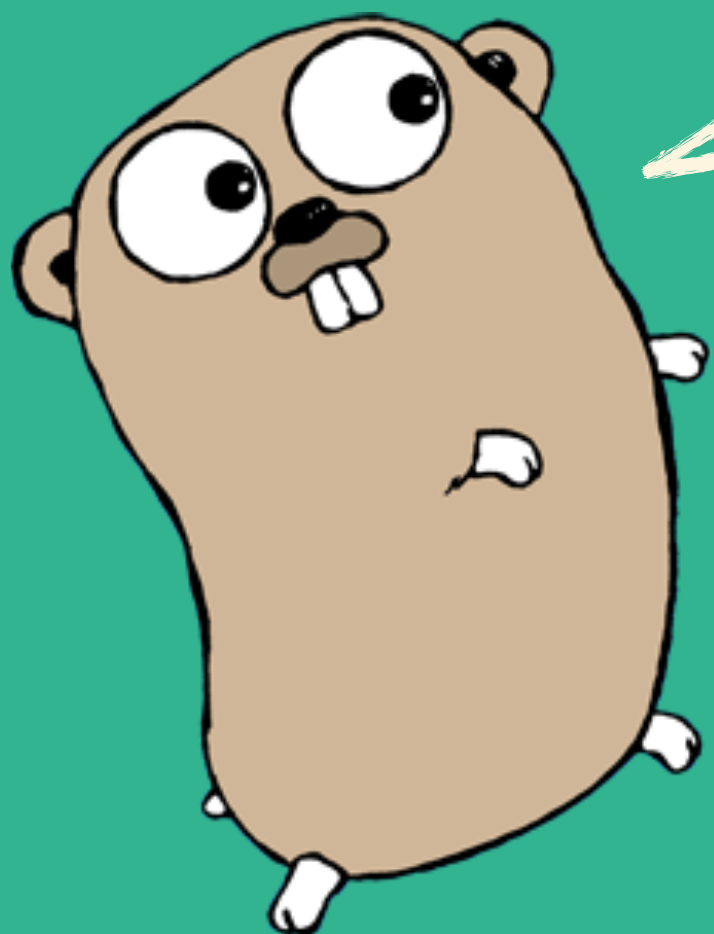


使い方さえ間違えなければ便利

- 開発実行環境のコンテナ化は優秀
 - ポータビリティ・再現性の高さ
 - 抱えるプロジェクトが増えるほど恩恵up↑↑
- テスト環境などのその場限りの環境
 - ビルドも早く使い捨て可能



ご清聴ありがとうございました



おわり