知ったらもっと楽しくなった！IW2018での注目セッション

Kubernetes ハンズオン
～Docker コンテナを運用のゲンバへ～

wakamonog
竹田 龍馬

このセッションの目的

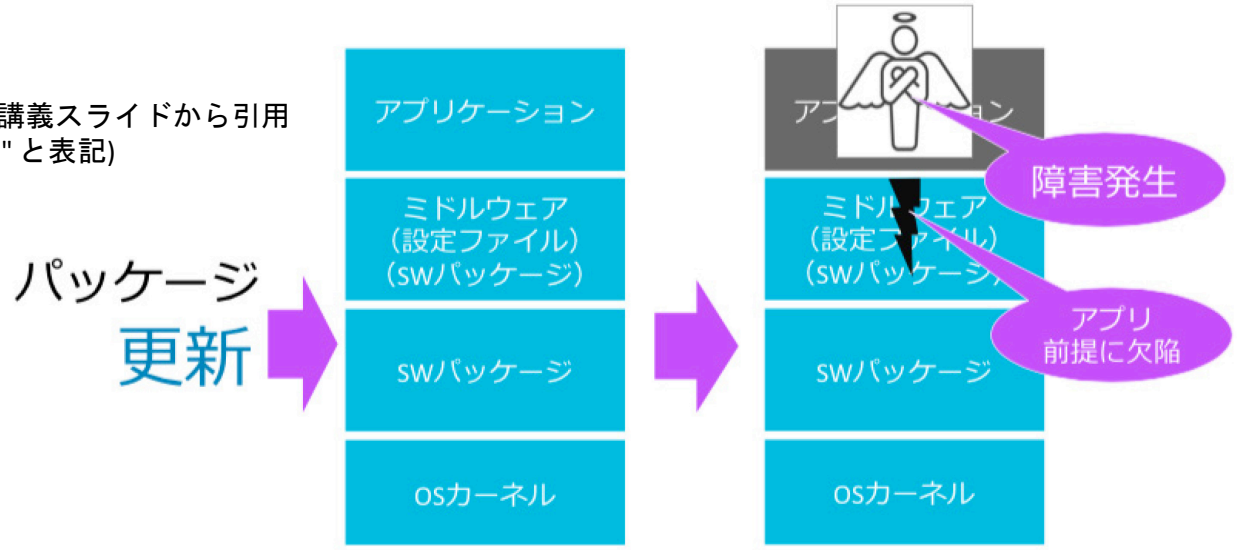
- Docker / Kubernetes (K8s) の基本的な操作方法を理解する
  - まずは基本となる Docker の使い方に慣れ
  - Docker をオーケストレーションする K8s の利便性を知り
  - 実際に Kubernetes を扱い、利便性を体感する

- ハンズオンセッションの特徴を活かす
  - 参加者の方々に「できる！」感覚を得てもらい
  - 現場への導入のきっかけとなれる事を目指す

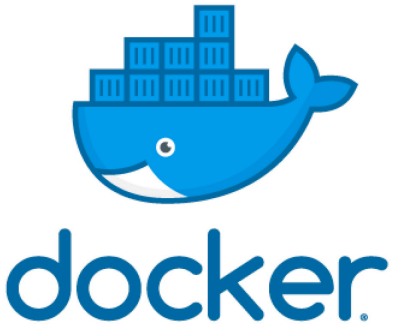
運用のゲンバが抱える課題

- 本番機と検証機のバージョンが完全に一致せず、更新作業で不具合が出た…
  - OS/ミドルウェア/パッケージ/アプリケーション などなど
- 大量のソフトウェアが導入される中、一致させた環境を作ることは容易ではない

講師の IBM 高良 真穂 氏の講義スライドから引用
(以降、"講義資料から引用" と表記)

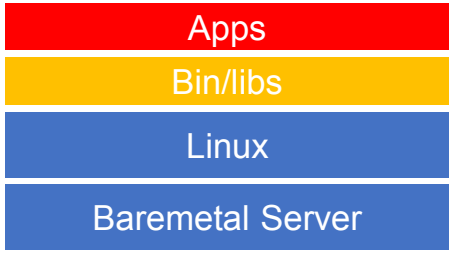


Docker とは

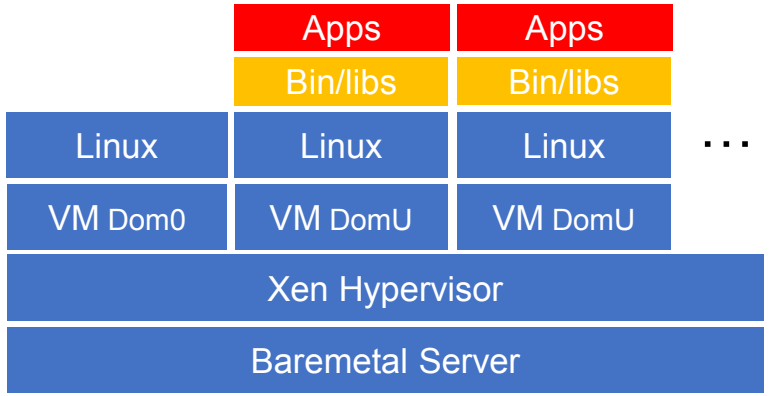


- Docker
  - コンテナ仮想化環境でアプリケーションを管理・実行するためのプラットフォーム
- Docker コンテナ
  - Docker により作られたイメージをもとに実行されるプロセス

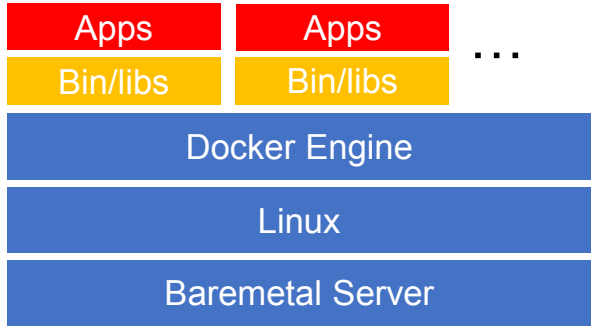
講義資料から引用



物理サーバー



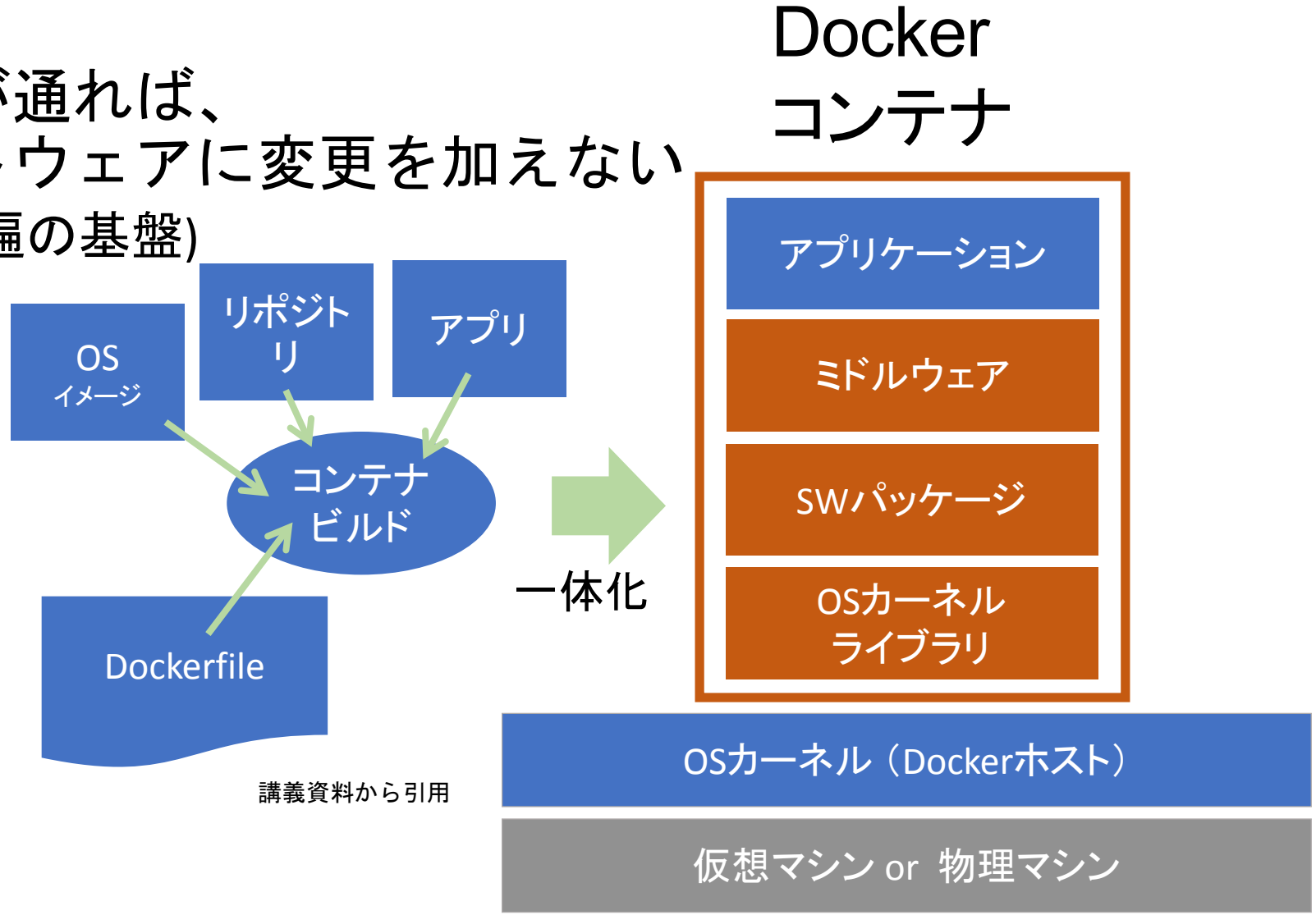
仮想サーバー



コンテナ

Docker による解決策

- アプリケーションの検証が通れば、
  その後はサーバーのソフトウェアに変更を加えない
  - Immutable Infrastructure (普遍の基盤)
- Dockerfile
  からコンテナを作成
- 変更が必要な場合
  にはコンテナを rebuild
- コンテナは
  OS kernel, SW 更新
  の影響を受けない



講義資料から引用

Docker ハンズオン

- IBM Cloud 上にてハンズオン環境の VM を作成
- Docker の環境を整備して参加者に提供
- Docker の動作理解と基礎的な操作方法について確認した
  - Docker コンテナの実行
    - `docker run hello-world`
  - コンテナの状態遷移と遷移させるコマンドについて
  - Docker Hub の見方
  - Docker Hub からのコンテナ起動
  - 複数の Docker コンテナをリンクさせる
    - ROCKET.CHAT を作成
    - DB として、mongodb を利用した
    - 次ページよりハンズオン内容を紹介

2つのコマンドで、以下の構成を作成します

演習

次の2つのコマンドを各自のIPアドレスに置き換えて、①②の順で、実行をお願いします。

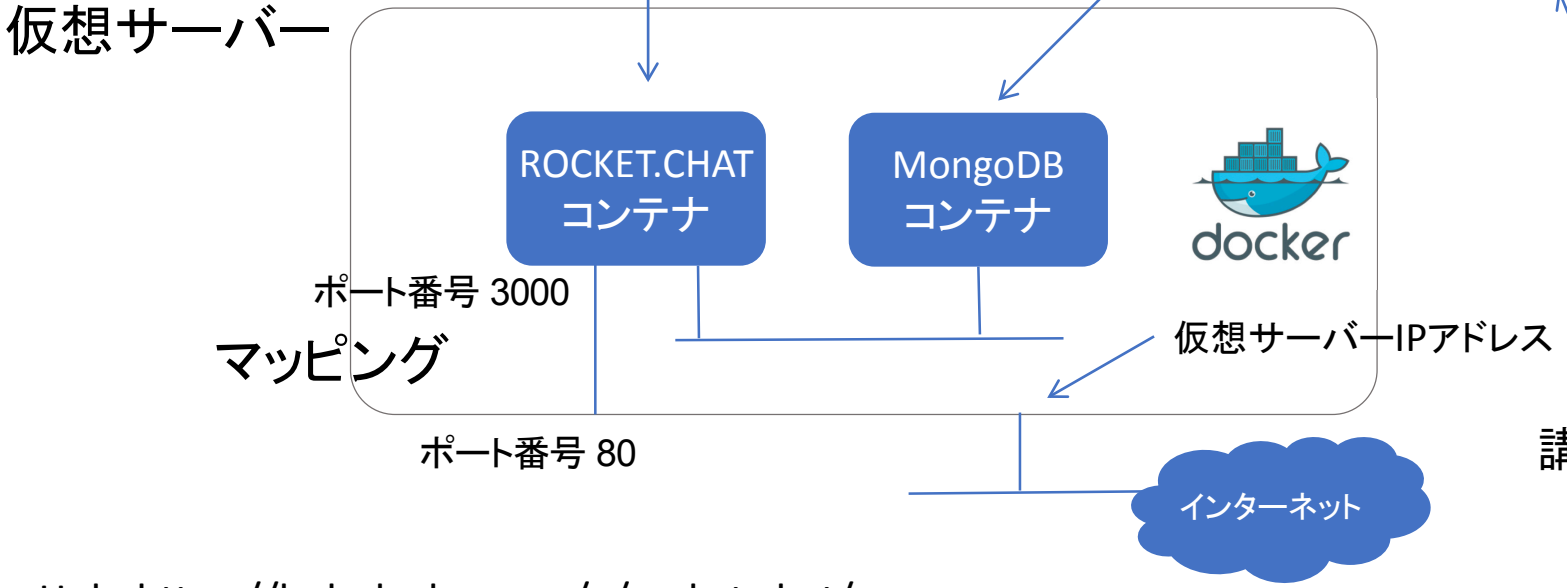
② ROCKET.CHATの起動

コンテナのプロセスへ環境変数をセットします。

```
docker run --name rocketchat -p 80:3000 --env ROOT_URL=http://仮想サーバーIPアドレス --link db -d rocket.chat
```

① MongoDBの起動

```
docker run --name db -d mongo:3.0 --smallfiles
```



--link db

- 内部ネットワークで接続二つを接続
- 名前解決を提供

講義資料から引用

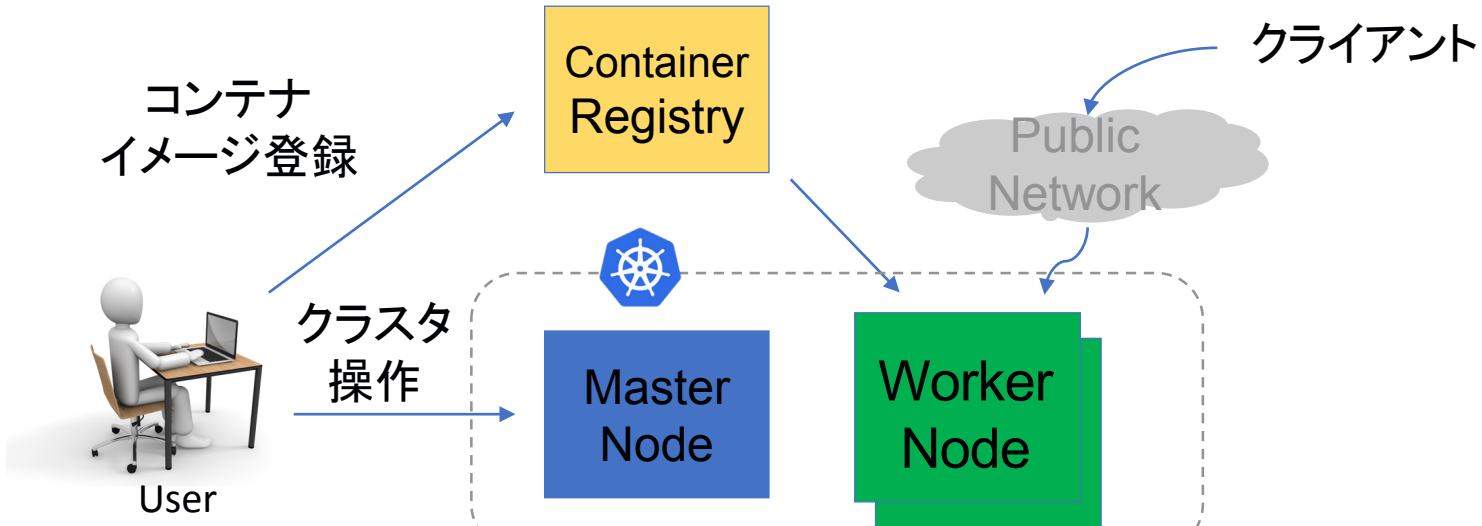
- DockerHub: https://hub.docker.com/_/rocket.chat/

Kubernetes の概要

- Docker 単体では運用のゲンバで使うための機能が不足している
- Kubernetes はコンテナのアプリケーション運用に用いる OSS
  1. コンテナの組み合わせ利用
  2. スケールアウト
  3. ロールアウト＆ロールバック
  4. 永続ストレージ利用
  5. 自己修復（可用性）
  6. クラスタの分割利用
  7. 監視＆ログ分析

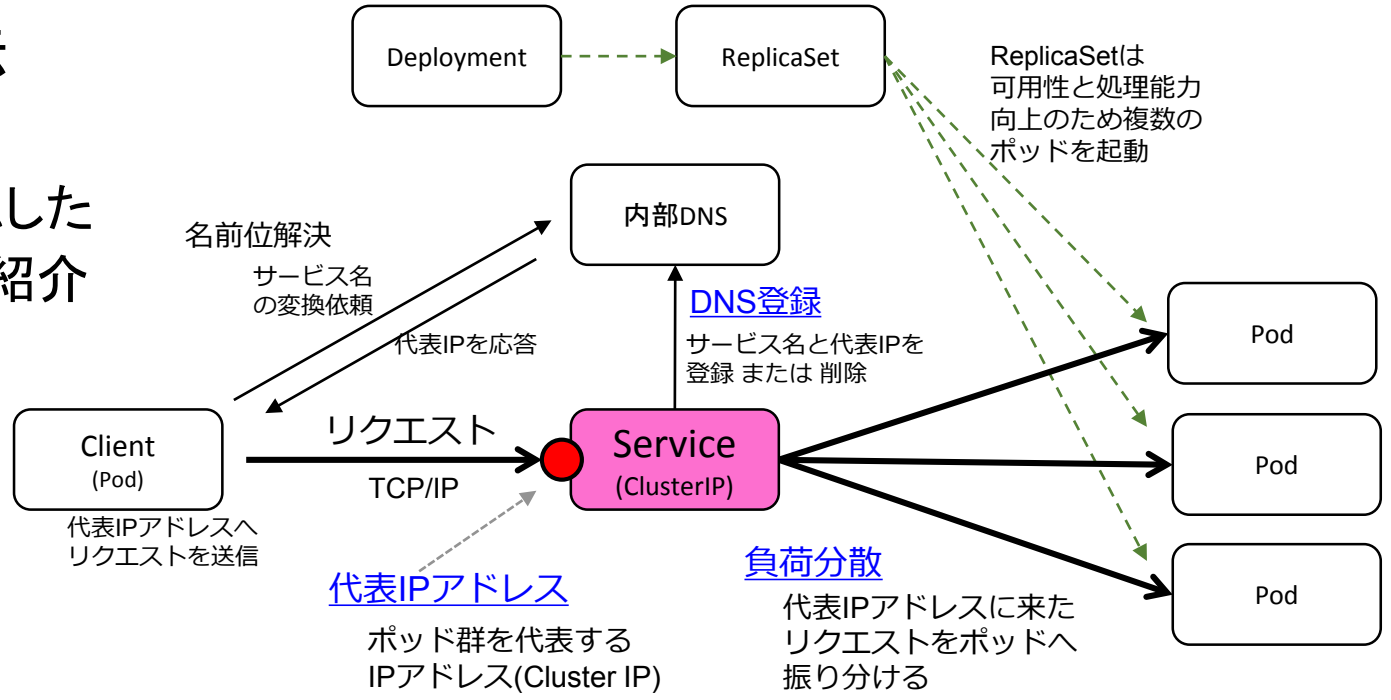
Kubernetes の三大構成要素

- マスターノード
  - K8sクラスタの制御(パブリック・クラウドの場合はマネージド)
- ワーカーノード
  - アプリのコンテナ稼働環境、ノード数可変
- コンテナ・レジストリ
  - イメージの保管場所



Kubernetes ハンズオン

- Docker ハンズオンの環境を引き続き利用
- 軽量かつローカルで K8s が動作する minikube を利用
- K8s と Docker の操作感の違いを知り、特徴を確認した
  - Pod, Controler (Deployment), Service について
  - Pod のマニフェストの記述方法
  - ROKET.CHAT をデプロイ
    - 自己回復が行われることを確認した
    - 次ページよりハンズオン内容を紹介



講義資料から引用

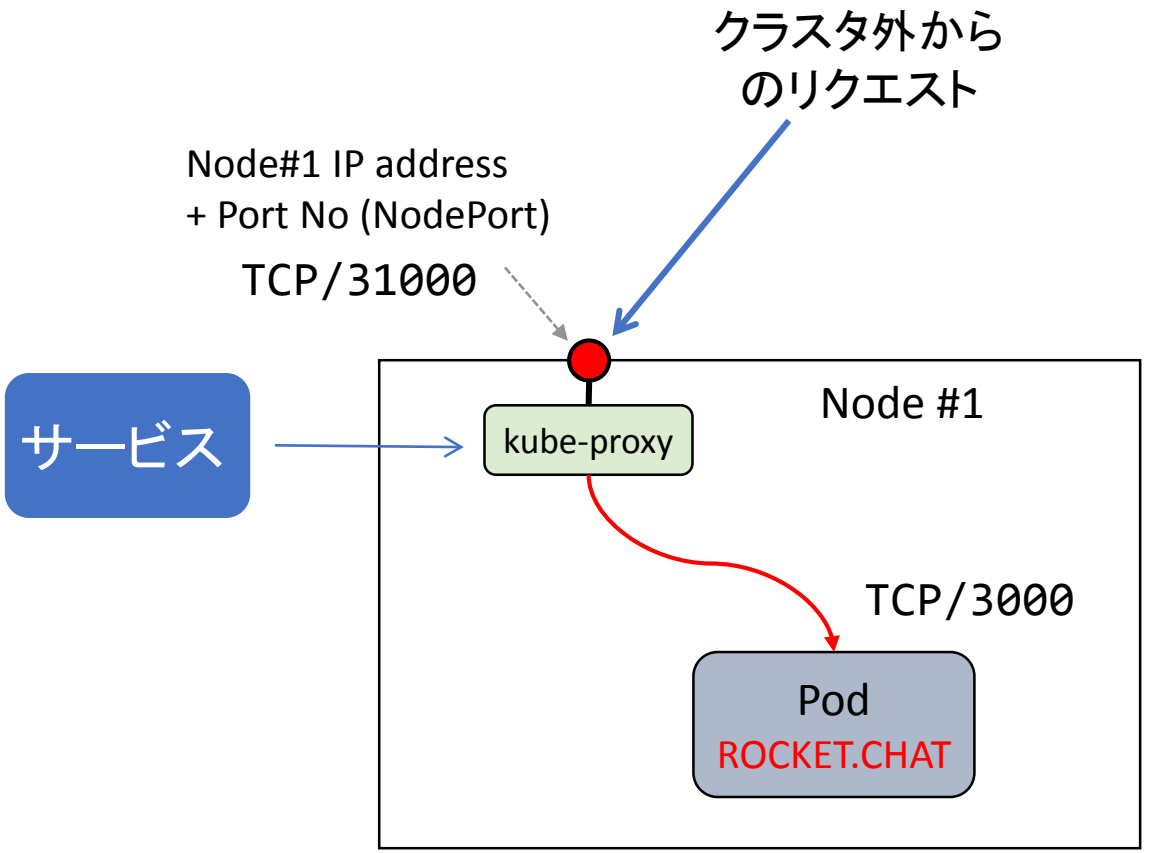
Internetからアクセスするためのサービスを作成

演習

NodePortは、ノードのIPアドレス上に公開用ポートを開きます。

サービスのマニフェスト

```
apiVersion: v1
kind: Service
metadata:
  name: rocket
spec:
  selector:
    app: rocket
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 31000
  type: NodePort
```



講義資料から引用

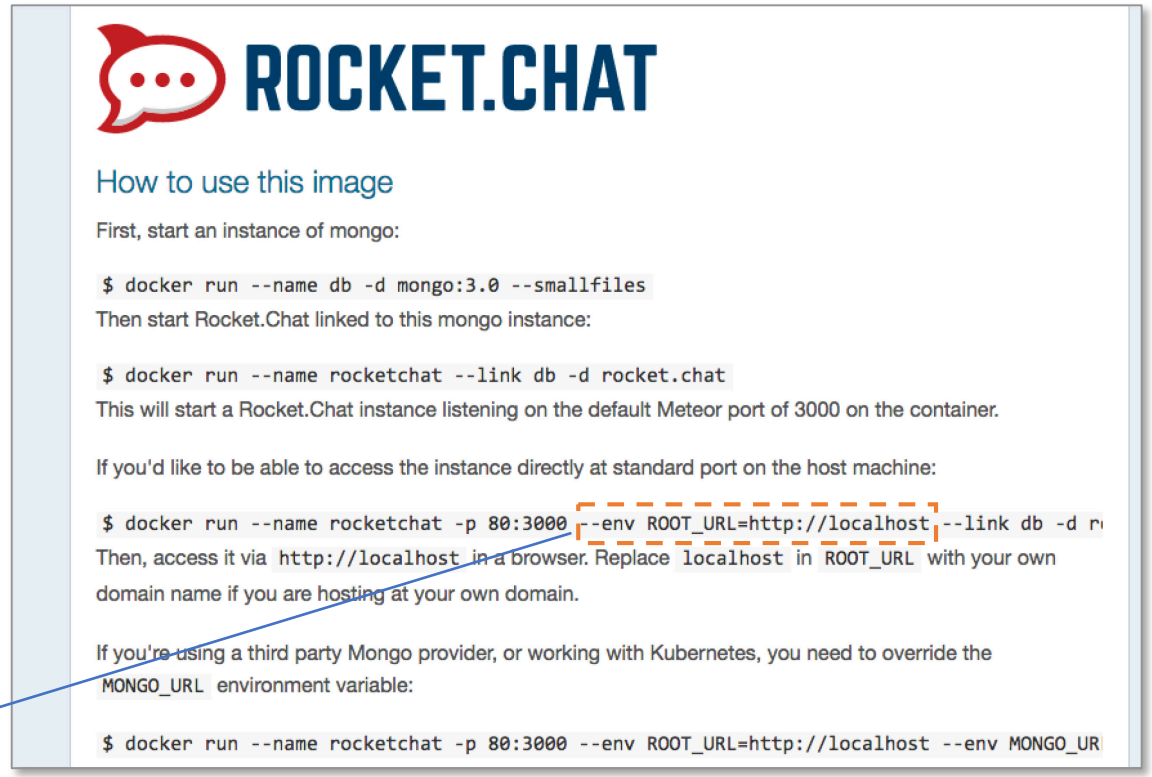
ROCKET.CHATを実行するマニフェストを作成

演習

DockerHubのROCKET.CHATのガイドに従って、YAMLを準備します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rocket
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rocket
  template:
    metadata:
      labels:
        app: rocket
    spec:
      containers:
      - name: rocket
        image: rocket.chat
        env:
        - name: ROOT_URL
          value: http://仮想サーバーのIP:31000/
```

https://hub.docker.com/r/library/rocket.chat/



講義資料から引用

ROCKET.CHATの起動

演習

ROCKET.CHATをデプロイして、起動を確認します。

ROCEKT.CHATのポッド起動	kubectl apply -f deploy-rocket.yml
起動状態の確認	kubectl get -f deploy-rocket.yml
ROCEKT.CHATのサービス作成	kubectl apply -f svc-rocket.yml
サービスの作成状態確認	kubectl get -f svc-rocket.yml

自己回復のテスト

演習

- ブラウザでROBOT.CHAT表示した状態で、下記を実施しましょう。
  - ROCKET.CHATのポッドを削除
  - MongoDBのポッドを削除
  - チャットに書き込んで、データが失われないことも確認してみましょう。

ポッドのリスト	kubectl get pod
ROCKET.CHATのポッドを指定して削除	kubectl delete pod rocket-XXXXXXXXX-YYYYY
MongoDBのポッドを指定して削除	kubectl delete pod mongodb-XXXXXXXXX-YYYYY

本セッションの総評と展望

- Docker / Kubernetes の基礎的な動作を習得する場を的供できた
- ハンズオン形式で実施するこで、
  参加者自身が体感しながら進めることができる
  - 「できる！」感覚を得てもらいやすい
- チャットベースでの質問を受付たことで、
  多くの疑問を回収できた
  - 技術的な質問は11件。その全てを 講師/TA でフォロー済み
- 参加受付開始から1ヶ月ほどで満員に
  - 来年度以降も Docker / Kubernetes に関するセッションを実施していきたい