# Kubernetes 触ってみた 😌
# (概念編)

ykonno

2018/06/09

# 概念編

## まずKubernetesとは

- **信頼性が高く**
- **スケーラブルな**
- **分散システム**

を構築できる。

# そのためにKubernetesは
# 何を提供してくれるか

# 重要な構成要素

## 特に感銘を受けた考え方に基づく機能

- イミュータブル
- 宣言的記述
- 自己修復機能

# イミュータブル

# イミュータブル？

- 一度システム上で成果物を作成したら
- ユーザがそれを更新しても
- **システム上の成果物は更新しない**

# イミュータブルであることの利点

## 例えば従来のサーバ構築

```
$ yum install openjdk
$ yum install httpd
$ yum install postgresql
 .
 .
 .
```

更新や変更の積み重ねで表現される

# イミュータブルであることの利点

## イミュータブルは

```
$ yum install openjdk
# イメージ更新
$ yum install httpd
# イメージ更新
$ yum install postgresql
#  イメージ更新
```

上記手順一つ一つで

**システム全体のイメージが更新される**

# イミュータブルであることの利点

それによって

- イメージそれぞれがバージョン管理できる
- なのでシステムのdiffが見える
- ロールバックできる

# 宣言的記述 

# 宣言的記述

## 手続き的記述と相対する考え

## 「イミュータブル」によってもたらされる機能

# 宣言的記述

- 手続きといえば、、、

## よくある3層システムの起動手順

```
$ systemctl start postgresql
$ systemctl start tomcat
$ systemctl start nginx
```

# 宣言的記述

- 前の例を置き換えるなら

```
# release.yml
containers:
  - name: web
    image: nginx:1.15
  - name: app
    image: tomcat:8.5
  - name: db
    image: postgresql:9.6
```

**Web,AP,DB それぞれ起動されていること**

という宣言

# 宣言的記述

**こうあるべき** を書いていく

# 自己回復するシステム 

# 自己回復

## 宣言的記述によってもたらされる利点

# 自己回復

例えば、宣言的記述で記載した

```
containers:
  - name: web
    image: nginx:1.15
  - name: app
    image: tomcat:8.5
  - name: db
    image: postgresql:9.6
```

これ。

**Kubernetesは宣言された状態を保とうとします**

# 自己回復

- 例えば、app(tomcat)のコンテナが落ちたら

**Appコンテナが一つ上がっていること**

とする状態へ復元してくれる

## イミュータブル

- 何もしてないけど壊れたを無くす

## 宣言的記述

- いわゆるInflastructure as Code
- サーバ構築手順が不要！

## 自己回復

- 障害対応が不要！

# その他にも面白い考え方がある

# ラベル 🏷️

# ラベル

- 今までの自分の考え方だと
- 例：
  - いわゆる組織図
  - 階層構造をなしている
  - 親と子の関係を持つ

**しかし、この考え方は**スケールしない

# ラベル

## Googleの巨大で複雑なシステムから生まれた教訓

# ラベル

## 本番環境で、「一つしかない」を嫌う

- APサーバ1号機、APサーバ2号機 ではなく
- 「APサーバ」というラベルを付与した
  オブジェクトの集まり

# ラベル

## 強制される階層構造はユーザには不便でしかない

どうです？

ワクワクしませんか？ 

こんなわくわくする
Kubernetesが
一瞬で実行できる

それが
GoogleCloudPlatform

# 次回、 実装編

ありがとうございました