

20180419 Japan Container Days v18.04

開催情報

- 名称：[Japan Container Days v18.04](#)
- 日時：2018年4月19日(木)
- 会場：ベルサール神田
- 定員：540名
- 後援：Docker Tokyo、Kubernetes Meetup Tokyo、Rancher JP、Japan OpenStack User Group、Container SIG、Mesos User Group Tokyo、PaaS JP、Bluemix Users Group、Serverless Community、GitLab Tokyo
- 協賛：Amazon ウェブ サービス
ジャパン株式会社、グーグル・クラウド・ジャパン合同会社、日本マイクロソフト株式会社、レッドハット株式会社、株式会社エーピーコミュニケーションズ、株式会社サイバーエージェント、日本アイ・ビー・エム株式会社、Pivotalジャパン株式会社、Rancher Labs、クリエーションライン株式会社
- 報告者：ボム

全体サマリと印象

- 技術コミュニティが主導する開発者のためのベンダーニュートラルなイベント
- 2017年度はdockerの話ばかりでしたが、今年はdockerの話がなくなってきた。 (docker + Kubernetesはコンテナ運用時のスタンダードになった)
 - Rancher社は自分たちが作ってたコンテナオケストレーションシステムCattleをKubernetesに切り替えた
 - Mesosも、Kubernetesと同時運用できますよなど他のContainer Orchestration Systemもkubernetesの話をしている。
- インフラ全般が"Cloud native"に変わってきてる
 - Cloud Native System
 - Container packaged (=> containerization)
 - Dynamically managed (=> orchestration)
 - Micro-services oriented (=> service mesh)
- 一番聞きたかったセッションは「Kubernetesの運用設計ガイド」だったが会場に人があふれて入られなかった
 - Kubernetesを導入することによって変わるArchitecture designと運用はみんなすごい興味を持っている
 - <https://speakerdeck.com/spesnova/a-design-guide-for-kubernetes-in-production-japanese>

詳細

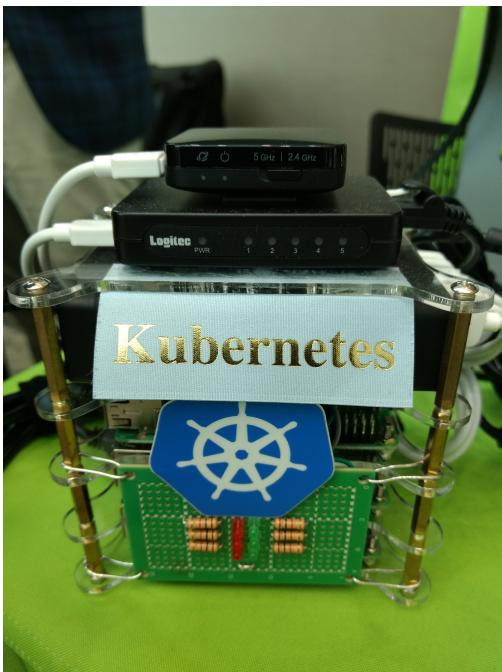
サイバーエージェントにおけるプライベートコンテナ基盤AKEを支える技術 [Masaya Aoyama(CyberAgent)]

スライド

<https://speakerdeck.com/masayaaoyma/saibaezientoniokerupuraibetokontenaji-pan-akewozhi-eruji-shu>

メモ

- OpenStack Active Technical Contributor
- CAではInfrastructure Engineerで趣味はKubernetes。
 - Raspberry Pi 3台でkubernetes clusterを作った



- AKEの紹介
- Ad Tech meets Kubernetes
 - Ad Tech needs...
 - low latency : 50ms以下 (10ms~100ms)
 - high traffic : 小さなデータをいっぱい送る 300,000 req/sec ~
 - high computing : PBクラスのデータ分析、ML
 - History Of AKE (adtech kubernetes engine)
 - 2016.06 部署でkubernetesを導入することを決定
 - 2017.04 Openstackベースのkubernetes engineをリリース (AKE)
 - 2017.07 persistent volumeがサポートできるようになった
 - 2017.09 type LoadBalancerが使えなかった 独自で実装
 - 2017.11 GKE Like Ingress → 独自実装
 - やっと使えるようになった
 - AKEだけでは15clusterを運用している

ポイント

- CAは独自実装が多い
 - Kubernetes自体がGoogleのcloud基盤を支えたものであってそれをOSS化してたらオンプレだと足りないものが多かったので独自実装して運用。
- CAでは2016年からサービス・開発基盤をコンテナ基盤に移行している (2015年末から検証)

マイクロサービスアプリケーションとしての機械学習 [Takuma Yamaguchi(Mercari)]

スライド

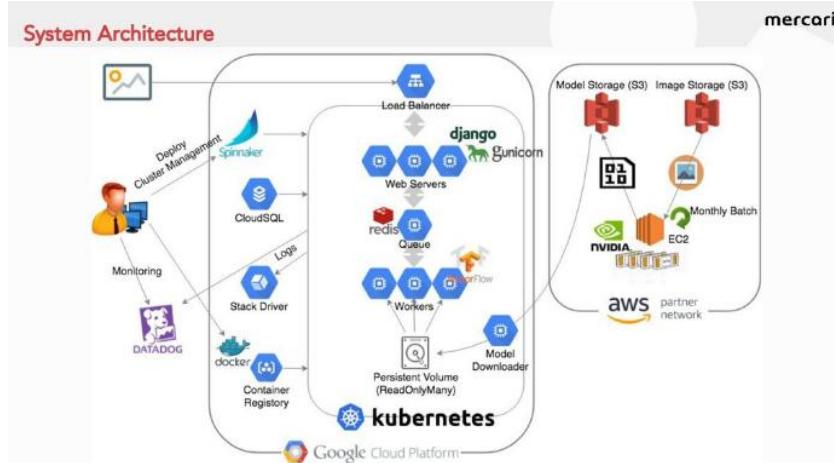
<https://speakerdeck.com/kumon/maikurosabisuapurikesiyontositefalseji-jie-xue-xi>

メモ

- Machine Learning Engineer
- 画像認識を中心とした機械学習に基づく機能開発
 - 出品時の画像認識基盤がGKE (Google Kubernetes Engine)



- 画像認識基盤を支えているGKE



- MLエンジニアが担当するところはウェブサーバとQueueとWorkNodesのみ。
- AWSでは画像保存でS3を利用しているのとGPUの利用のためにバッチサーバはAWSを利用している
- 通常の運用ではDatadogでモニタリング、Spinnakerでデプロイ・リソース管理をしている

- 新機能の画像認識基盤は一週間で作れた
 - 2017年度からMonolithic ArchitectureからMicroservicesへ移行していた
 - すべて作りなおす必要がなく、新機能はMicroserviceとして実装
- 機械学習におけるMicroservicesのメリット
 - 機械学習モデルのサービスへの組み込みが早い
 - 影響範囲が明確
 - 機械学習モデルだけ更新することも簡単
 - Blue Green Deployment

ポイント

- 機械学習エンジニアの立場から機械学習アプリケーションのシステム構成や運用を紹介

"Yahoo! JAPANのKubernetes-as-a-Service"で加速するアプリケーション開発[Masaya Ozawa(Yahoo! Japan)&Kazuki Suda(Z Lab)]

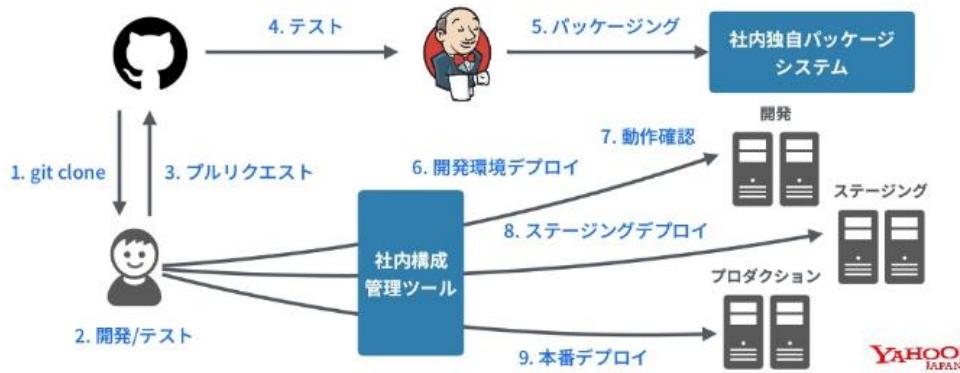
スライド

<https://www.slideshare.net/techblogyahoo/yahoo-japan-kubernetesasaservice>

メモ

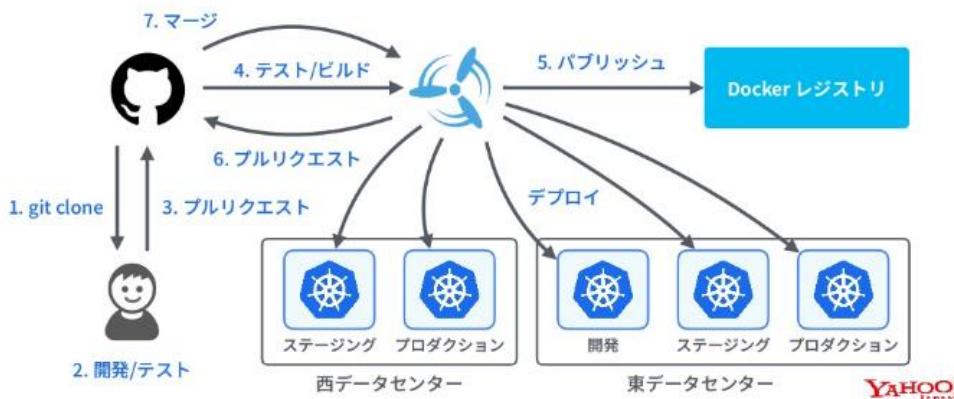
- Yahoo!ズバトクでのKubernetes導入事例
 - 2017年KAAS (Kubernetes As A Service)に移行した
 - 既存開発フロー

今までの開発フロー



○ Kubernetes導入後の開発フロー

Kubernetes導入後の開発フロー



○ 開発フローの変化

- Github上の操作だけで運用するフローを導入
 - 開発着手からリリースまでにかかる時間が短く。
- Dockerコンテナ/Kubernetesの特性ありきのフロー
 - Masterへのマージからリリースまで同じDockerイメージを利用 (portability)
 - Kubernetesを利用することでデプロイも簡単になった
- Kubernetes以降にあたり大変だったこと
 - 利用技術のスイッチ
 - packagingやci/cd pipelineの変更
 - 経験なしで学習が大変 (勉強会などで共有)
 - 考え方/設計方針のスイッチ
 - 既存の考え方や設計方針のままでは、Kubernetesと思想にあわない
 - いわゆるCloud Nativeな考え方へ変えていくためにチームメンバー間で勉強会を開きながらプロジェクトを進めた。
 - 既存システムをそのまま乗せるのは危険、機能ごとにシステムを分離してKubernetesに構築していく

• Kubernetes as a Service

- Kubernetesクラスタの作成・削除・アップグレードなどの運用を簡単に行えるようにするための管理サービス
- GKE, AKE, EKS 対応
- 2016年の7月から開発を始めて、2017年10月からヤフーの一部サービスで本番運用
- Kubernetesの管理をKubernetesで行っている
 - 各サービスチームに提供するKubernetesクラスタはインフラ全体を管理するKubernetes上で動いている

• Z Lab Corporation ソフトウェアエンジニア @superbrothers

- ヤフーにおけるコンテナインフラ基盤の開発や技術研究
- kubernetesでインフラを管理
- 独自実装でloadbalancer・ingressを作った

ポイント

- Yahoo!ズバトクはKubernetes as a Serviceを導入することによって開発フローや障害対応の自動化が進んだ
- Kubernetes as a Serviceは運用のすべてを自動化することを目指している (NoOpsにしたい)

Kubernetesセキュリティベストプラクティス - Ian Lewis(Google)

スライド

<https://speakerdeck.com/ianlewis/kubernetesfalsesekiyuriteifalsebesutopurakuteisu>

メモ

- kubernetes clusterからアプリケーションを管理する際のセキュリティベストプラクティス
 - RBAC
 - Role Based Access Control
 - ユーザーやサービスアカウントに利用するロールのみを付与
 - RBAC設定はネームスペース範囲
 - API Server Firewall
 - APIサーバへのアクセスをIPアドレスに制限
 - Network Policy
 - データベースをアクセスするPodを制限する
 - How: label selectorでPodを選択
 - network plug-inを利用する: calico, weave, etc
 - Non-root ユーザーで実行
 - コンテナが突破されたもHostを安全にする
 - 読み込み専用ファイルシステムを利用する
 - readOnlyRootFileSystem: true

ポイント

- manifestを書く際のセキュリティポイントを共有

Custom MetricsとMultiple Metricsを使ってKubernetesのクラスタの強みを最大限活用する - Kodai Sakabe(Wantedly)

スライド

<https://speakerdeck.com/koudaiii/number-containerdaysjp>

メモ

- Horizontal Pod Autoscalers - Custom Metrics and Multiple Metrics
- Wantedly
 - Microservices : 124 (QA+Prod含め)
 - Kubernetes Node : 80+ (GKE+AWS)
 - Containers : 3K+
- Kubernetesを導入するには人のパラダイムシフトが必要
 - リソースの管理はアプリケーションが何個立ち上がっているんだっけではなく、nodeごとに配置されているpodsの(metricsから)percentを見る
- HPA (Horizontal Pod Autoscalers)
 - 特定条件によってscale out, scale inする機能
 - custom Metrics
 - Multiple Metrics
- Kubernetes 公式ドキュメントだけではカスタムメトリックは利用できない
 - kubernetesは三ヶ月に一回大きくマイナーバージョンが変わる
 - ドキュメントはそれについていく
- HPAを導入するにはまずPodの状況を可視化する
 - prometheus
- cpu利用率をベースにして作成する
 - 基本CPU利用率50%を基準としてscale out/inする

ポイント

- Custom Metricsを利用したPod Autoscalersの実装に関しての簡単な説明

Container Networking Deep Dive - Hirofumi Ichihara(NTT)

スライド

<https://www.slideshare.net/hichihara/container-networking-deep-dive-94307233>

メモ

- Openstack core reviewer
- DockerはCNI、KubernetesはCNMを利用
- CNI (Container Network Interface)
 - CNI Plugin
- CNM (Container Network Model)
 - CNM IPAM Plugin
 - CNM Network Plugin
- flannel
 - シンプルで簡単な設定でL3ネットワークの構築ができる
- Project Calico
 - L3ネットワーク構築
 - IPIPによるtunnnlingを支援
- パフォーマンス改善
 - オフロード機能を適切に設定する
 - VXLANハードウェアオフロード
 - ネットワークと向き合う
 - tcpdumpしながらiptablesルールとlinux network設定を読み続く

ポイント

- Kubernetesにおけるネットワークプラグインの選考に役に立つ。

Helmを利用したKubernetes as a Serviceの実装 - Motohiro Otsuka(NEC Solution Innovators)

スライド

<https://speakerdeck.com/yuanying/helm-woli-yong-sita-kubernetes-as-a-service-falseshi-xian>

メモ

- NEC OSS Community部署
- Motivation
 - 社内のプライベットクラウドで自由にk8sクラスタが欲しい、けれども管理したくない
- What Is k8s
 - コンテナオーケストレーションツール
 - クラウドアプリケーションを構築するための一番最速な方法 (cloud-native application)
- Helm
 - k8s package manager
 - 一般的なサービス(redmine,nginxなど)を簡単に構築するためのもの
 - Helmでpackage化することでk8sで簡単にデプロイできる
- KaaS (Kubernetes as a Service)
 - K8s clusterをクラウドのユーザに提供するサービス
 - control plane (k8s master)の部分をクラウドで用意する、worker nodeはユーザ側で用意する
- kubernetes(helm)でkubernetesを管理する
- Live Demo

ポイント

- kubernetesのpackage管理ツールの紹介

Google Kubernetes Engineにおけるバッチ処理のパターン - Shota Yoshikai(Kabuku)

スライド

<https://speakerdeck.com/tinjyuu/google-kubernetes-engine-niokerubatutichu-li-falsepatan>

メモ

- 3D解析とイメージ解析の基盤としてGKEを利用している
- バッチ処理の定義
 - 非同期
 - 数秒から数時間以上かかる処理
- バッチ処理のパターン
 - kubernetes job
 - message queuing

- Cloud Pub/Sub
- Rabbit MQ
- Kubernetes job
 - 一度限りの処理を実行させるためのコンテナ
 - 処理が終わるとコンテナが終了する
 - kubernetesの標準機能
 - リトライ機能
 - 数が多くなるほど大変になる
- (ondemand) message queingを組み合わせる
 - 管理コストが高まる
- (cloud) message queingを組み合わせる
 - コストがかかるけど安心・安定感ある
- バッチ処理のパターンを検討する上で
 - 構成をシンプルに
 - Jobで要件を満たすか
 - 構成するコンポーネント数を少なくする
 - 導入コスト、運用コスト
 - クラウドを使うのがミドルウェアをブチ込むのか
- Jobでのバッチ処理を検討して難しいのであればMessage queingを検討

ポイント

- 大きくリソースが必要なところ(イメージ認識・解析)でのシステム運用例

~ベンダーロックインを回避せよ~ Kubernetesコンテナー管理OSS、Rancher 2.0の全貌 - Yosuke Shindo(Rancher Labs)

スライド

共有なし

メモ

- cloudstackを作った人たちが作った会社
- VPがCNCFのボーダーメンバーの一人
- Rancher1.6と2.0との大きい違い
 - マルチオーケストレータ(cattle,k8s,mesos)からKubernetesベースに。
 - シングルテナントからマルチテナントに。
 - 1.6は今年いっぱいサポート
- rancherはgitlabとの組み合わせが非常にいい(牛とたぬき)
- Rancher2.0
 - GKE, AKS, EKS対応
 - オーケストレータはKubernetesをベースに
 - k8s clusterがimportできる
 - on-premiseで便利な機能を揃えている
 - 1.6とのマイグレーション機能は2.1から
 - 5月 GA リリース予定 (現状は2.0 beta)

ポイント

- rancherがcattleを捨てて2.0からはkubernetesを採用する
- 2018年5月 2.0 GAリリース予定

ファイル

IMG_20180419_130047820.jpg	4.81 MB	2018/04/26	李 大範	
mercari_image_reco.PNG	183 KB	2018/04/26	李 大範	
mercari_architecture.PNG	188 KB	2018/04/26	李 大範	
yahoo_kube_devlop.PNG	125 KB	2018/04/26	李 大範	
yahoo_kube_devlop_new.PNG	156 KB	2018/04/26	李 大範	