PAR-28

仮想マシンとはここが違う Amazon ECS でわかるコンテナ監視実践入門

古川 雅大
株式会社はてな
Mackerel開発チーム SRE

自己紹介

- 名前: 古川 雅大 (id:masayoshi Twitter: @yoyogidesaiz)
- 所属: 株式会社はてな
- MackerelチームのSRE (Site Reliability Engineer)
- 趣味
  - ネットワーク、サーバーをいじること
  - 格闘ゲーム

Mackerelの紹介

mackerel 製品 料金 事例 資料 サポート お知らせ ログイン 無料トライアル お問い合わせ

News 2021.11.10 Terraform Provider Mackerel が AWS インテグレーションに対応しました

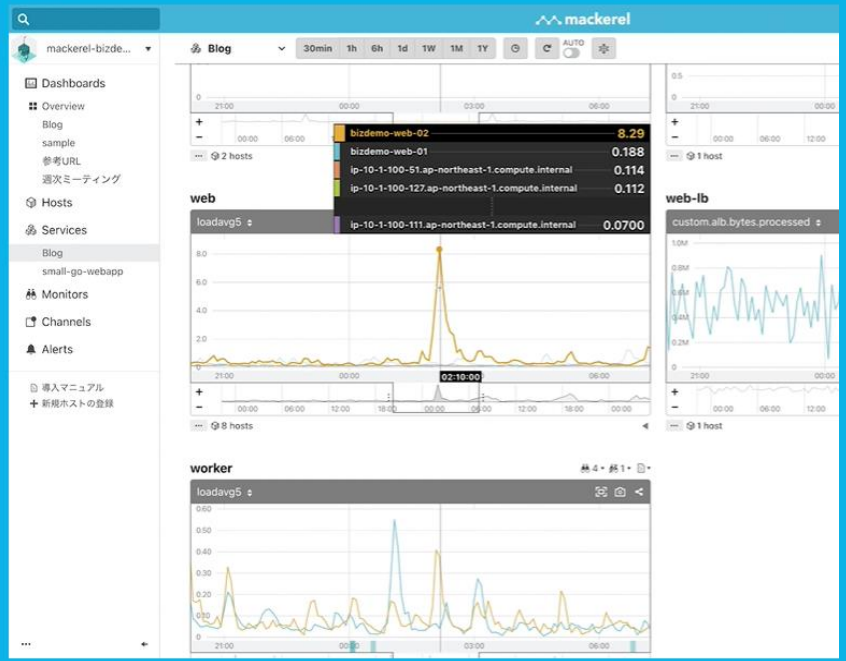
監視を育てる、Mackerel

クラウド時代に最適な監視モデルを
使いやすいUIで提供し、
システムの運用・監視に
チームで取り組む文化を作る
「クラウド運用の道標」となる
SaaS型サーバー監視サービス。

無料トライアルをはじめる

お問い合わせをする

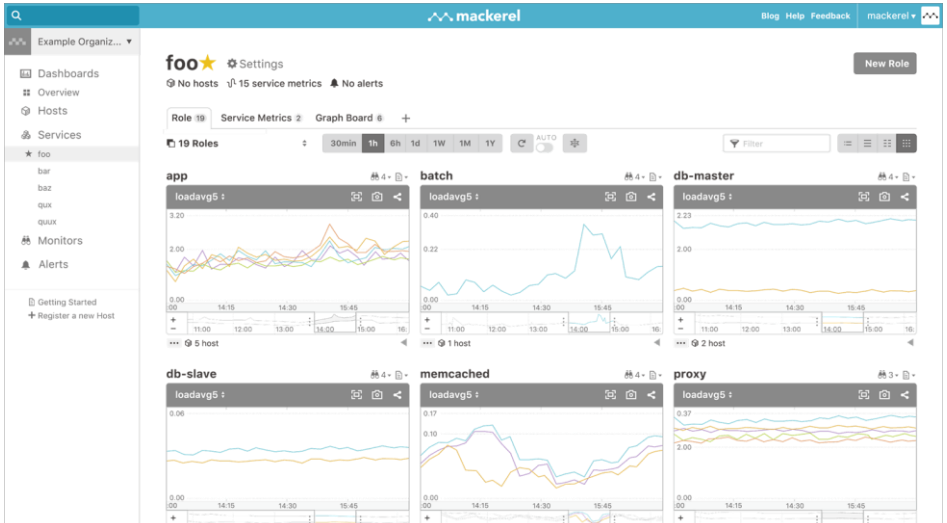
または 資料をダウンロードする >

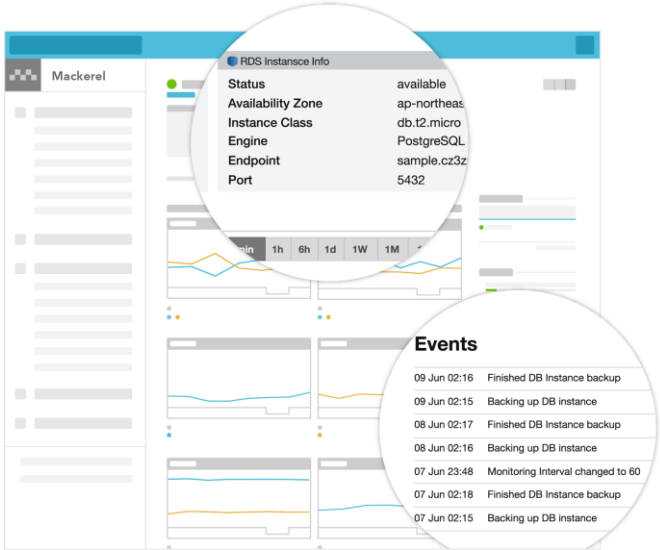


KDDI IDCFrontier CyberAgent NTT東日本 classmethod NTT SmartConnect SEGA mercari freee レコチョク

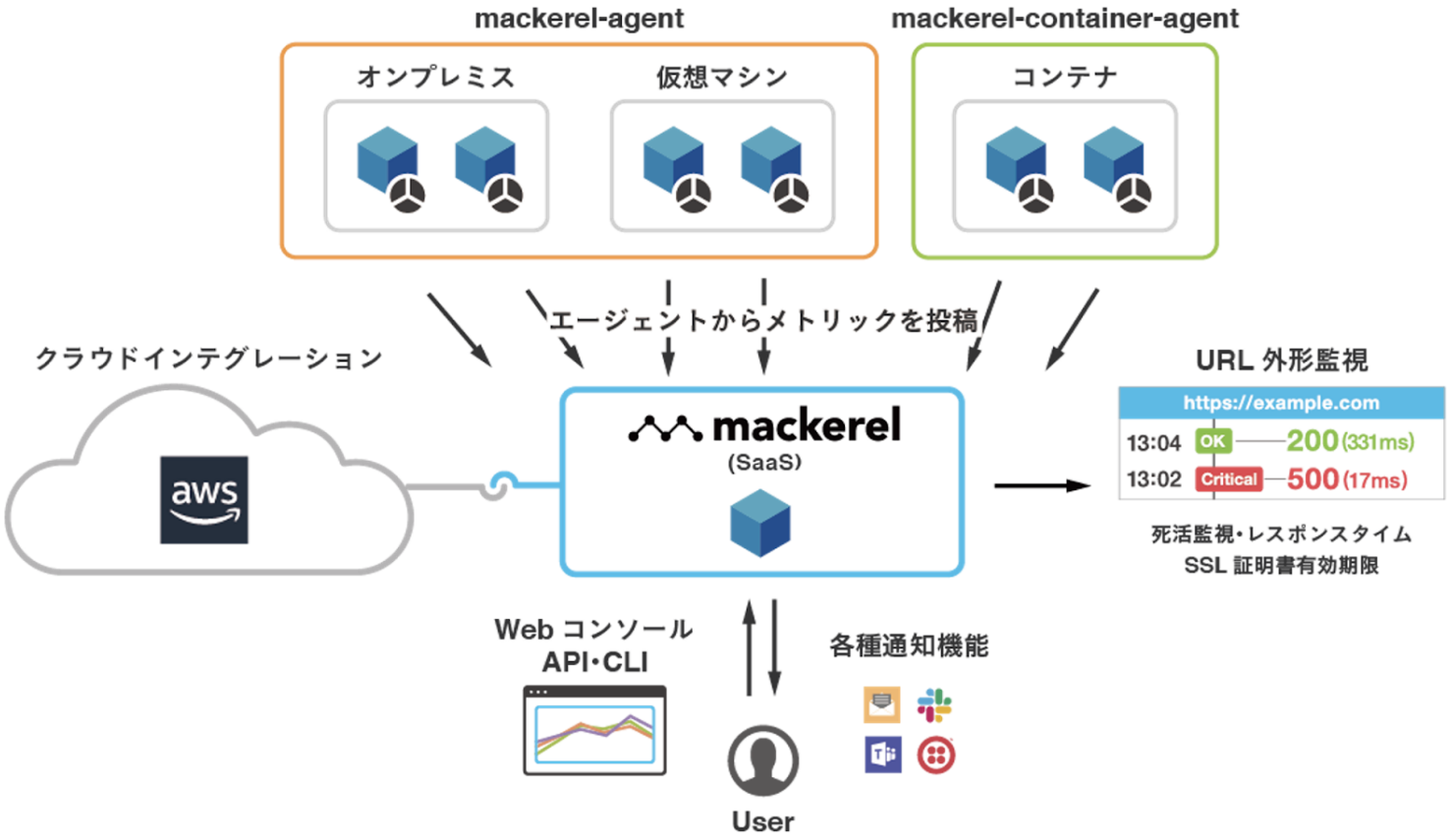
システムの運用・監視を簡単にはじめられます

- 導入はガイドにしたがってコマンドを実行するだけ
- エージェントが死活監視とメトリック取得を自動で開始
- 直感的なUIでサービスの状況を可視化
- 監視をはじめる敷居が下がり、チームで取り組めます





Mackerelのアーキテクチャ



はじめに

話すこと、話さないこと

- ターゲット
  - 仮想マシン(VM)からコンテナ化に挑戦し、コンテナの監視設計をする人
  - クラウドリフトはしたけど、クラウドシフトはまだの人
- 話すこと
  - VM とコンテナで監視がどう変わったか
  - Amazon ECS (AWS Fargate) の監視例
- 話さないこと
  - コンテナ化などの環境移行の仕方
  - コンテナ**実行基盤** (Dockerホスト側)の監視の話
    - 例えば Amazon ECS なら ECSインスタンス,
      Amazon EKS ならワーカーノード

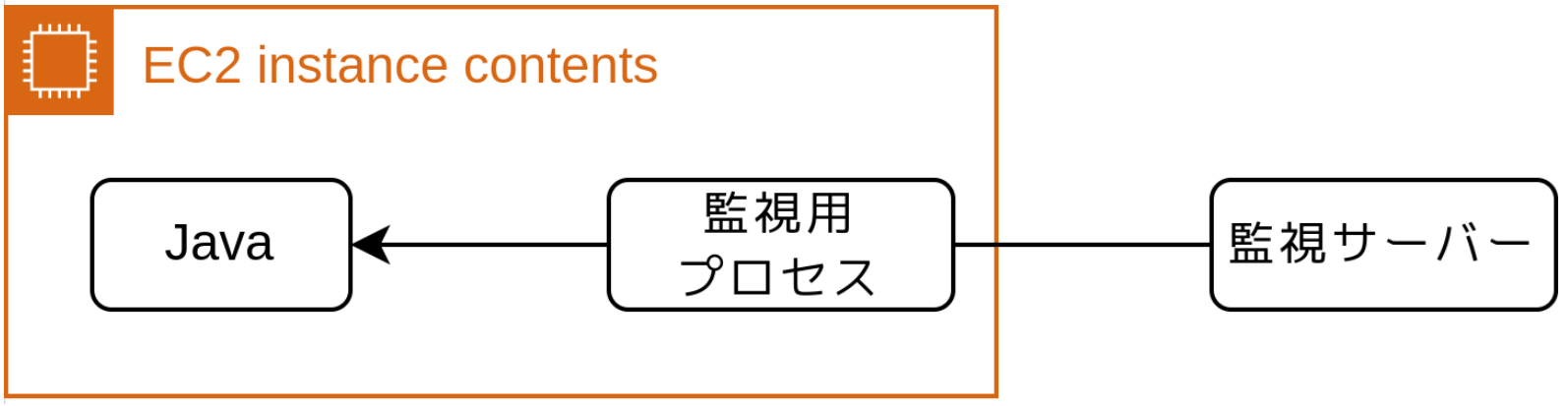
伝えたいこと

- VMとコンテナの違い
- コンテナ監視の考え方
- Amazon ECS での基本的な監視手法
- (これはPRです) 監視SaaS Mackerel の名前

VM環境の監視

今回想定するVM環境

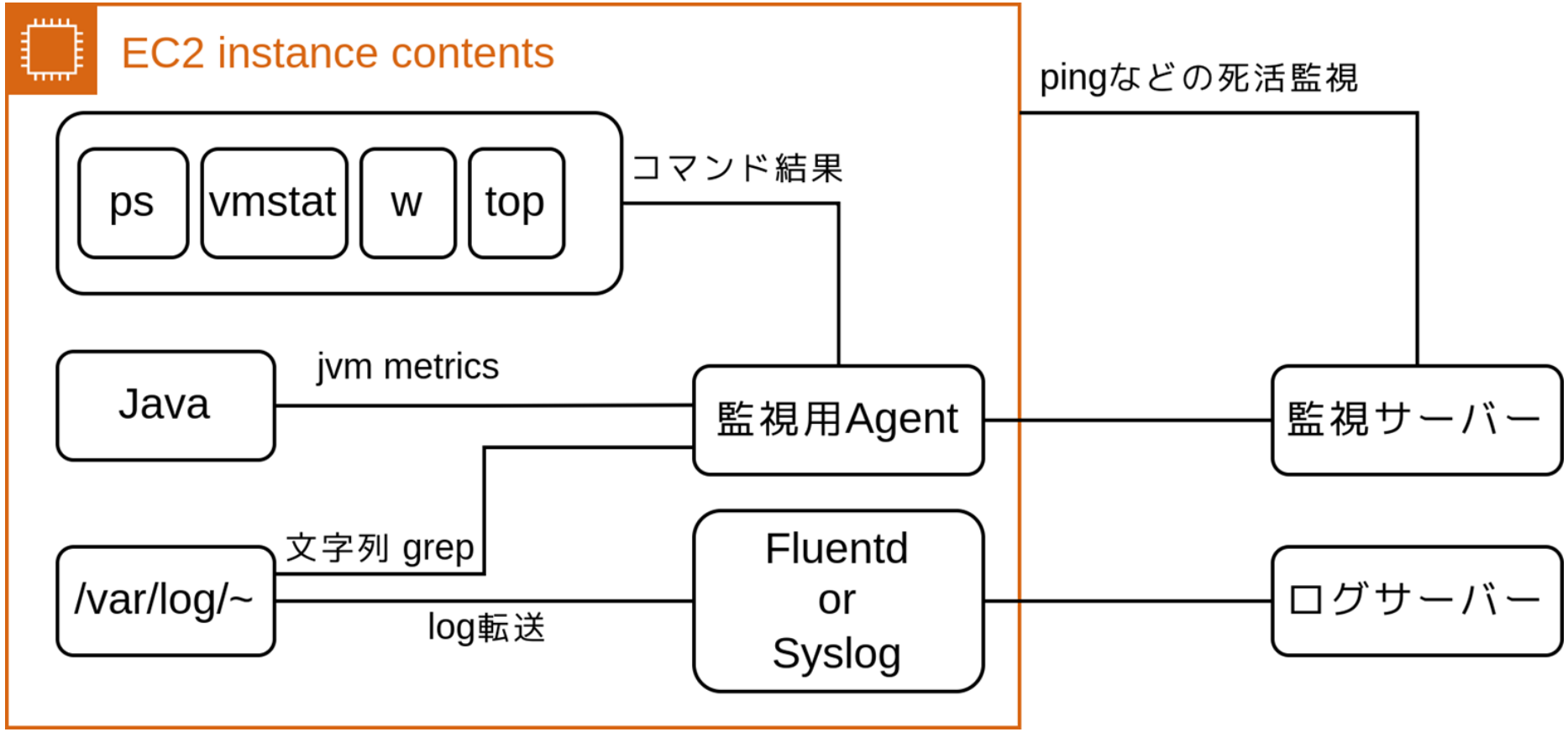
- オンプレからAWSに移行
  - Java のWebアプリケーションを Amazon EC2 上に構築
  - Amazon CloudWatch, ALB などAWSのサービスを活用せずに、オンプレ環境で利用していた監視をそのまま移行
  - クラウドリフトした状態



VM環境での監視

- サーバーリソース監視
  - cpu, memoryなどのサーバーリソース監視
- プロセス監視
  - psコマンド結果などを使ったプロセスの監視
- 死活監視
  - 監視サーバーから ping などでサーバーの死活監視
- ログ監視
  - ログでFatalなどの文字列検出を行うログチェック監視

VM環境の具体的な監視実装例



VMとコンテナの違い

想定するコンテナ環境

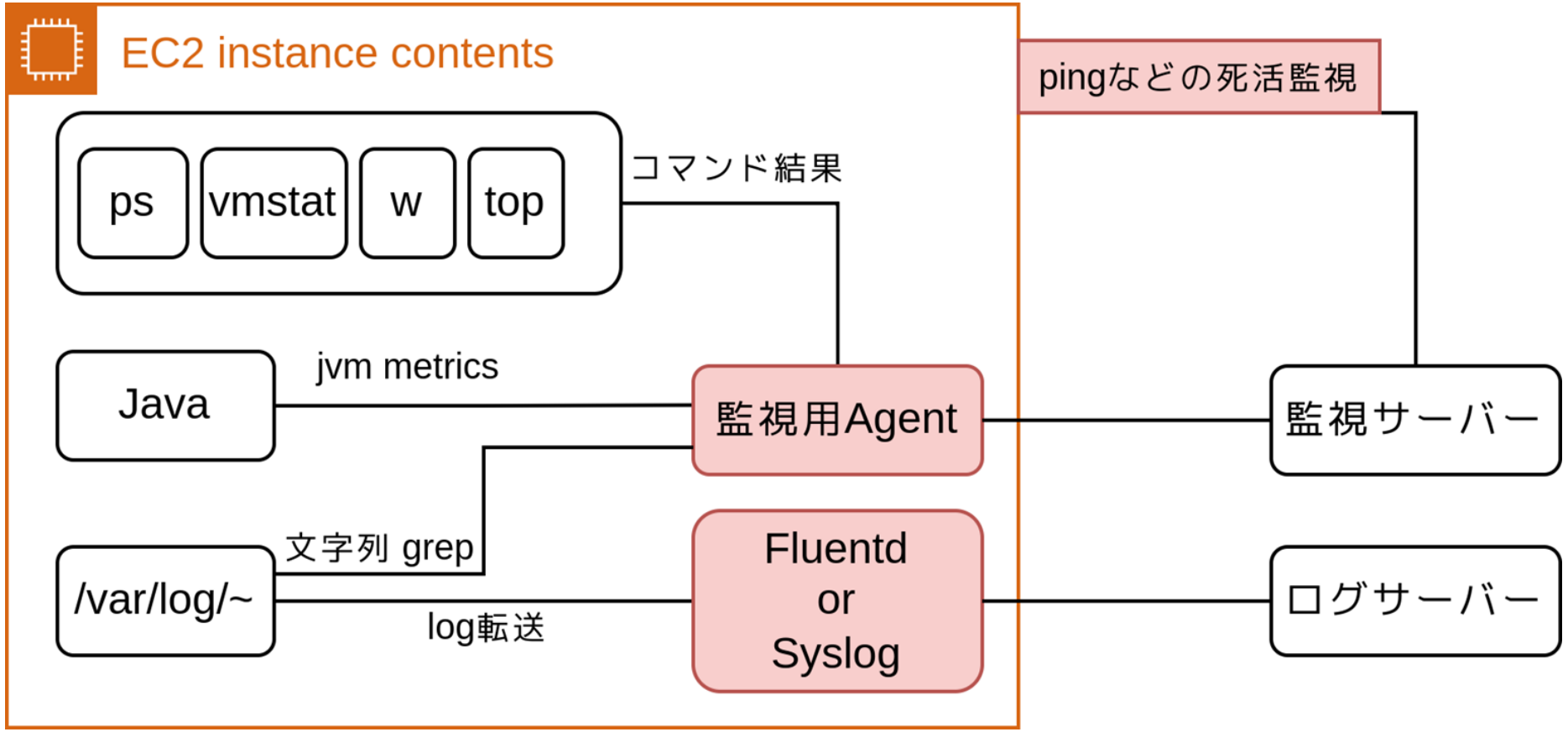
- 先程の Amazon EC2 環境を Amazon ECS (AWS Fargate) に移行
- クラウドシフトした状態を目指す
  - コンテナにおける監視の考え方を理解する
  - コンテナや AWSが提供してくれる機能を活用

VMとコンテナの違い

- 原則 **1プロセス 1コンテナ**
  - App + 監視プロセス は同じコンテナでは動かせない*
- デプロイ時に作り直される
  - 外からの死活監視も難しい (IPが変わったりする)
- コンテナ内にSSHでリモートログインが難しい
  - できるが、出来ることを前提に設計しないほうが良い
- オーケストレーターと併せて利用することが多い
  - Amazon ECS, kubernetes などの仕様,機能も考慮する必要がある

* 技術的には可能だが非推奨

コンテナでは考え直さないといけない箇所



コンテナ環境の監視

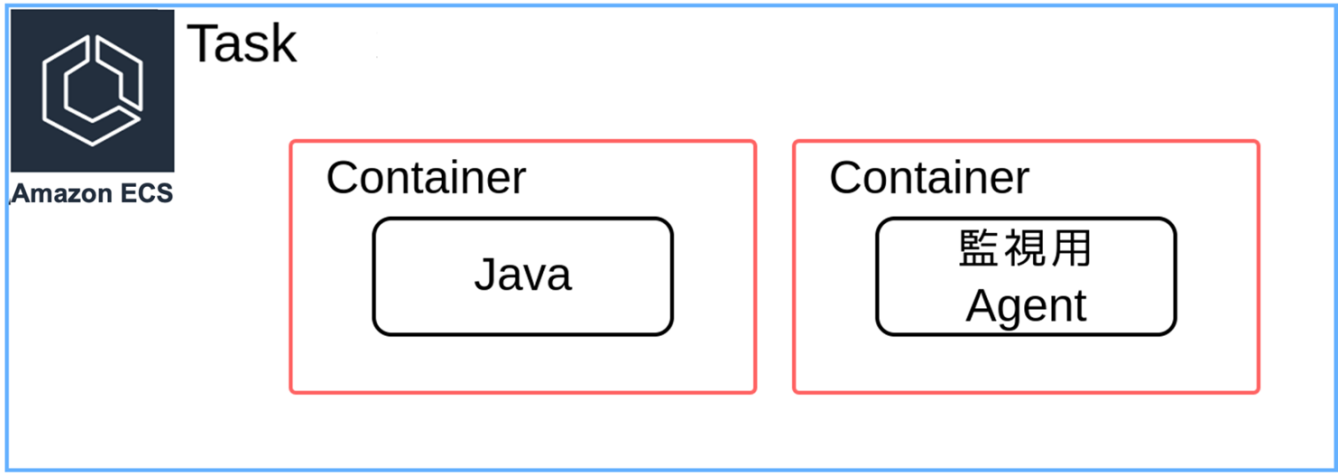
Amazon ECS でのコンテナ監視

- Side Car パターンで App のメトリックを収集
- Amazon ECS の機能を理解してプロセス監視&死活監視
- CloudWatch Metrics で AWS のリソースメトリック
- CloudWatch Logs でLog監視を実現

Side Car パターンの利用

side car*パターンの利用

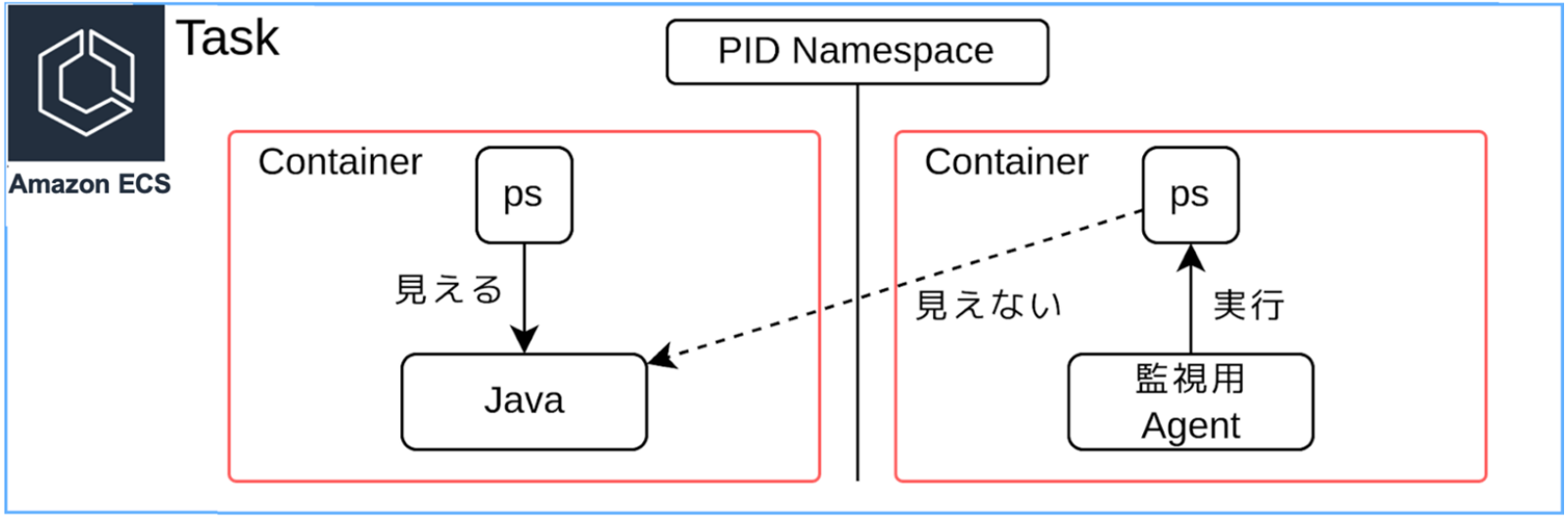
- App用コンテナ + 監視Agent用のコンテナの2つをセットで動かす
  - このセットを Amazon ECS だとタスク、k8sならpod



* 更に細かい区分にするとAdapterパターンに分類されるやり方もあるが、ここではSide Car パターンとして紹介する

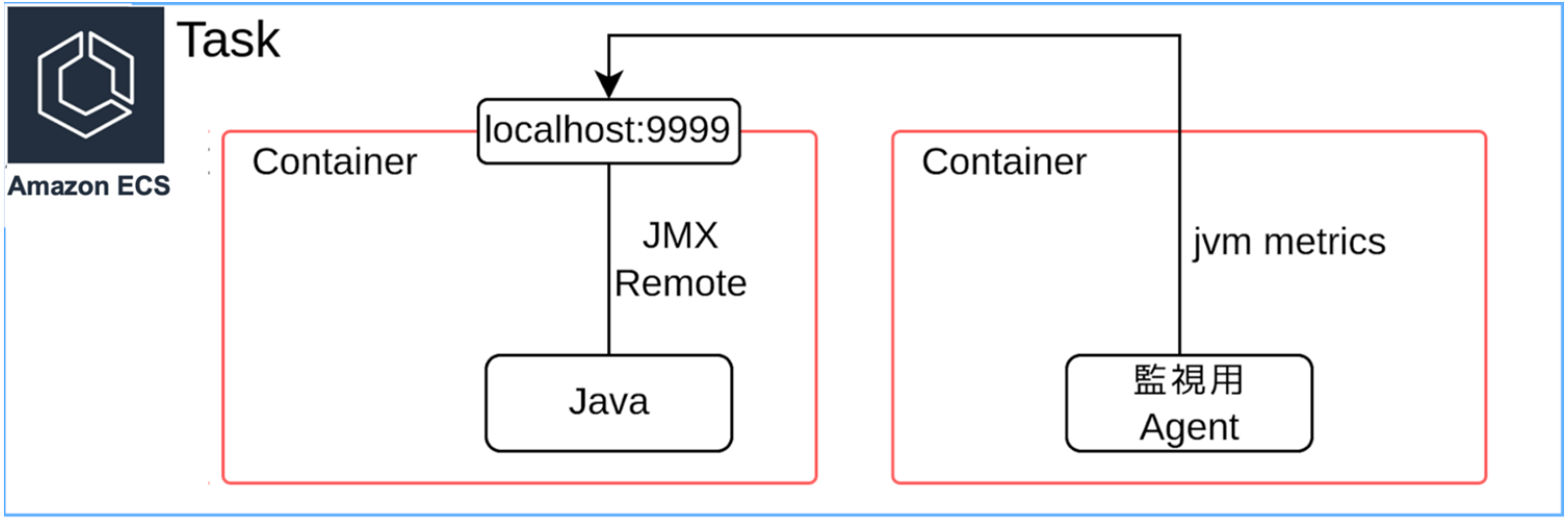
side car から見えないもの

- 例えば、psコマンドによる死活監視は出来ない
  - side car のpsコマンド結果は監視用Agentだけ
  - プロセス以外も様々な制約がある



コンテナ監視の基本はネットワーク経由

- コンテナ間はネットワーク通信が基本
- ネットワーク経由で取得できるようにしないといけない
  - 外から取れるように「メトリクスの export」 が必要



（参考）Observability（可観測性）

- 「出力から現在の状態を推定できる能力」
- コンテナやマネージドサービスを使う上では重要な能力
  - 「中に入って推定する」ことから、「外から状態を推定出来る」ようにする
  - 状態を推定できるようにメトリクスやログを出力(export) する

Amazon ECS
プロセス監視 & 死活監視

Amazon ECS の責任範囲

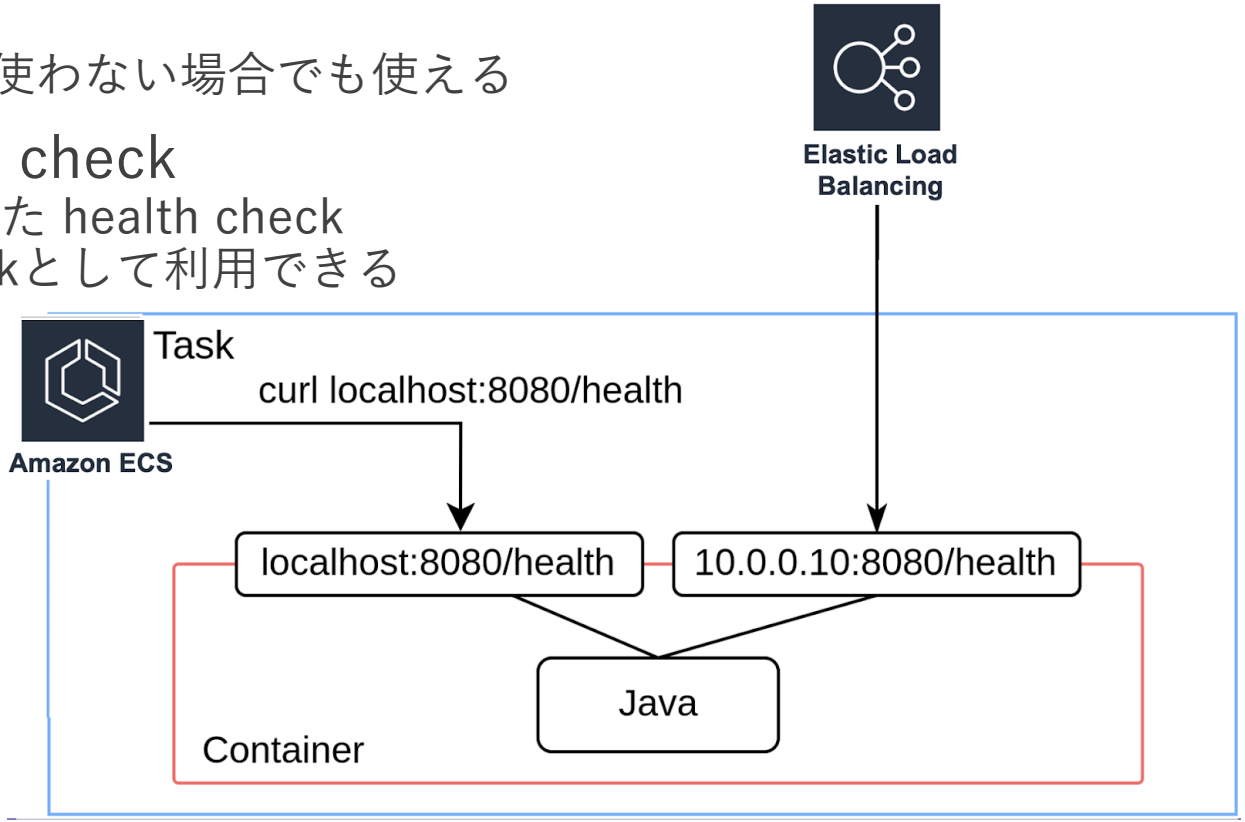
- Amazon EC2 上で動くものは、ユーザーの責任だった
  - サーバーの起動はAWSの責任範囲*
  - Appプロセスの起動や再起動はユーザーの責任範囲
    - プロセス自動起動設定ミス、デプロイ後のreloadミスなど
    - だから、プロセス監視をして異常がないか確かめたかった**
- Amazon ECS はタスク(コンテナ)の起動がAWSの責任範囲
  - 1コンテナが起動する ≒ 1プロセスが起動する
    - つまりコンテナにおける死活監視 = プロセス監視
  - コンテナが起動できない or 異常なときにタスクを終了する
  - Amazon ECS にプロセスが異常かどうか伝えるのがユーザーの責任範囲
    - プロセスの状態が外(Amazon ECS) からわかるようにする

* OSの設定で起動しない、設定ミスなどは当然ユーザー側の責任

** プロセス監視で十分かどうかという議論はある

Amazon ECS 2種類の health check

- docker run の health check
  - コマンドも使える
  - 初期起動時やALB を使わない場合でも使える
- ALBやNLB の health check
  - HTTP や TCP を使った health check
  - 外からのhealth checkとして利用できる



AWS が提供するメトリクスの活用

CPU、メモリ、ネットワーク使用量

- AWS は以下の方法で 「メトリクスを出力」 している

- タスクメタデータエンドポイント
  - コンテナ上から http リクエストするとCPU, Memoryなどの統計情報が取れる
- CloudWatch Metrics
  - Amazon ECS Container Insights を有効にすることで取得できる

https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task-metadata-endpoint-v4.html
https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-metrics-ECS.html

ALB のメトリクスの活用

- ECS サービスのタスクたちの重要なメトリクス
  - HTTPのステータスコードカウント
  - 90%ile, 99%ile のレスポンスタイム
- CloudWatch Metrics で提供される

CloudWatch Logs でLog監視

ログ配送はどうする?

- Amazon ECS (AWS Fargate) のログ配送
  - 標準出力を CloudWatch Logs に配送
    - 設定が簡単だが、CloudWatch Logs にしか送れない
  - FireLensを利用して side car の Fluent Bit などに配送
    - Fluent Bitの設定が必要だが、好きなところに配送できる

https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/using_firelens.html

ログ監視はどうする?

- CloudWatch Logs Insights で集計ができる
  - クエリを書くことで様々な集計が可能
  - 例えば、1分間で Fatal 文字列が出た回数など
  - check log 監視の変わりにも使えるし、エラー数を集計してエラー率にして監視することも出来る

https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/logs/CWL_QuerySyntax.html

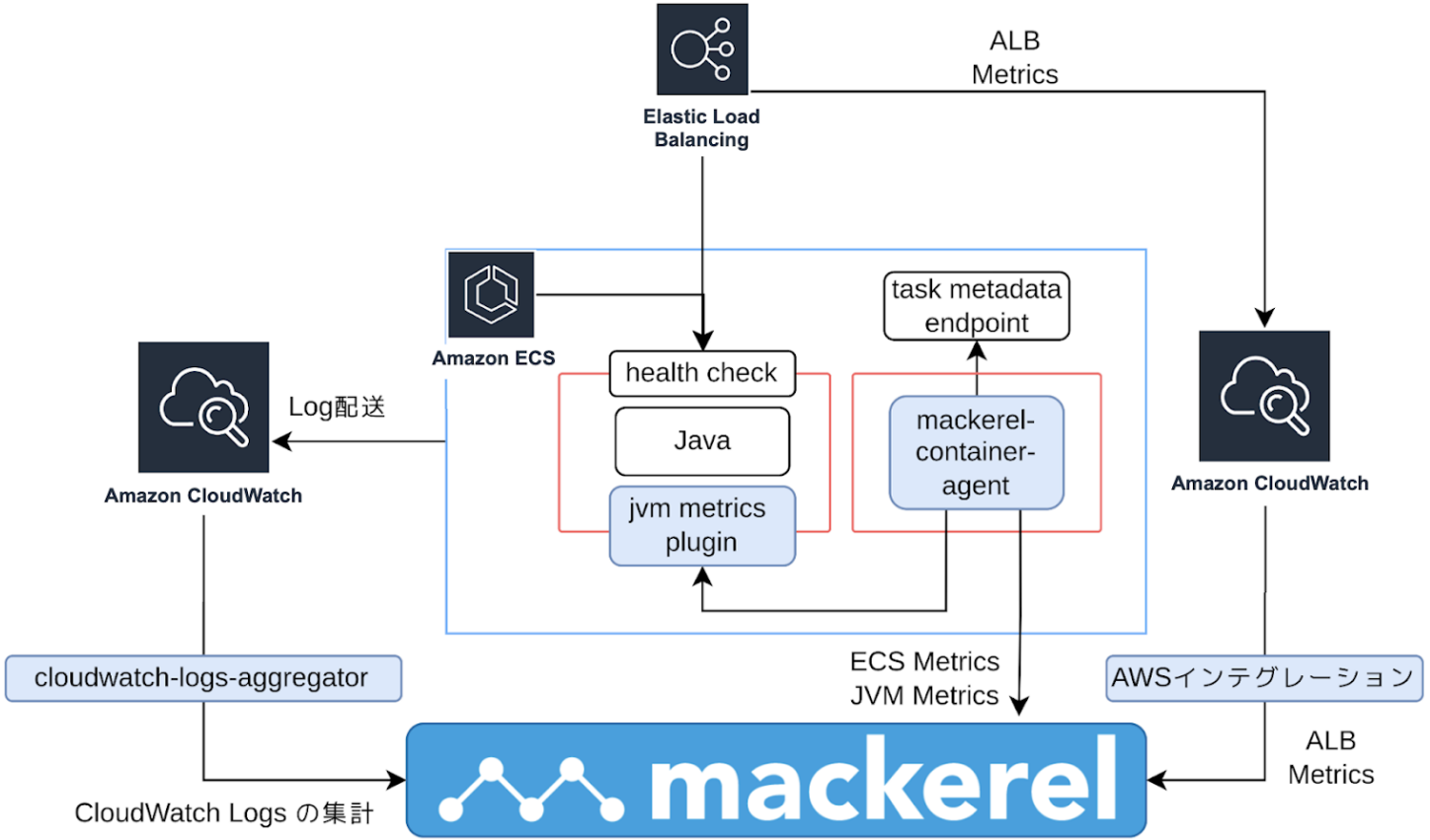
Mackerelを使った例とまとめ

Amazon ECS でのコンテナ監視

- Side Car パターンで App のメトリックを収集
- Amazon ECS の機能を理解してプロセス監視&死活監視
- CloudWatch Metrics で AWS のリソースメトリック
- CloudWatch Logs でLog監視を実現

- 複数の手法、サービスを組み合わせて作っていく
  - 最初は大変に感じるが、1回作り変えれば楽が出来る
- 組み合わせるためにツールの作成が必要だったりもする
  - 監視用Agent や その plugin, 集計や通知などなど
  - 監視SaaS を使うとそういったツールを提供してくれる
  - 監視SaaSといえば...?

Mackerelを利用した例



* 青い部分はMackerelが提供しているツール、機能になります

まとめ

- VMとコンテナの違い
- コンテナ監視の考え方
- Amazon ECS での基本的な監視手法
- (これはPRです) 監視SaaS Mackerel の名前

Thank you!

古川 雅大

株式会社はてな
Mackerel開発チーム SRE