aws DEV DAY ONLINE JAPAN

20-22.10.2020

DEV DAY

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws DEV DAY
ONLINE JAPAN

A - 5

そのコンテナでサービスインできますか？

Fumihide Nario
Solutions Architect

Amazon Web Services Japan

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

自己紹介

名前

成尾 文秀（なりお ふみひで）

所属

アマゾン ウェブ サービス ジャパン 株式会社
技術統括本部 ソリューションアーキテクト

好きなAWSサービス

Amazon ECS

Amazon EKS

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

セッションの目的とゴール

コンテナを利用した開発を進めていく中で
「本番環境でアプリケーションを動かすとはどういう事か」という視点で
予期される事態に備え次の項目について考え今後の開発に取り入れて頂く

- 環境変数
- コンテナイメージ
- ログ
- モニタリング

aws
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
In Partnership with intel

aws DEV DAY
ONLINE JAPAN

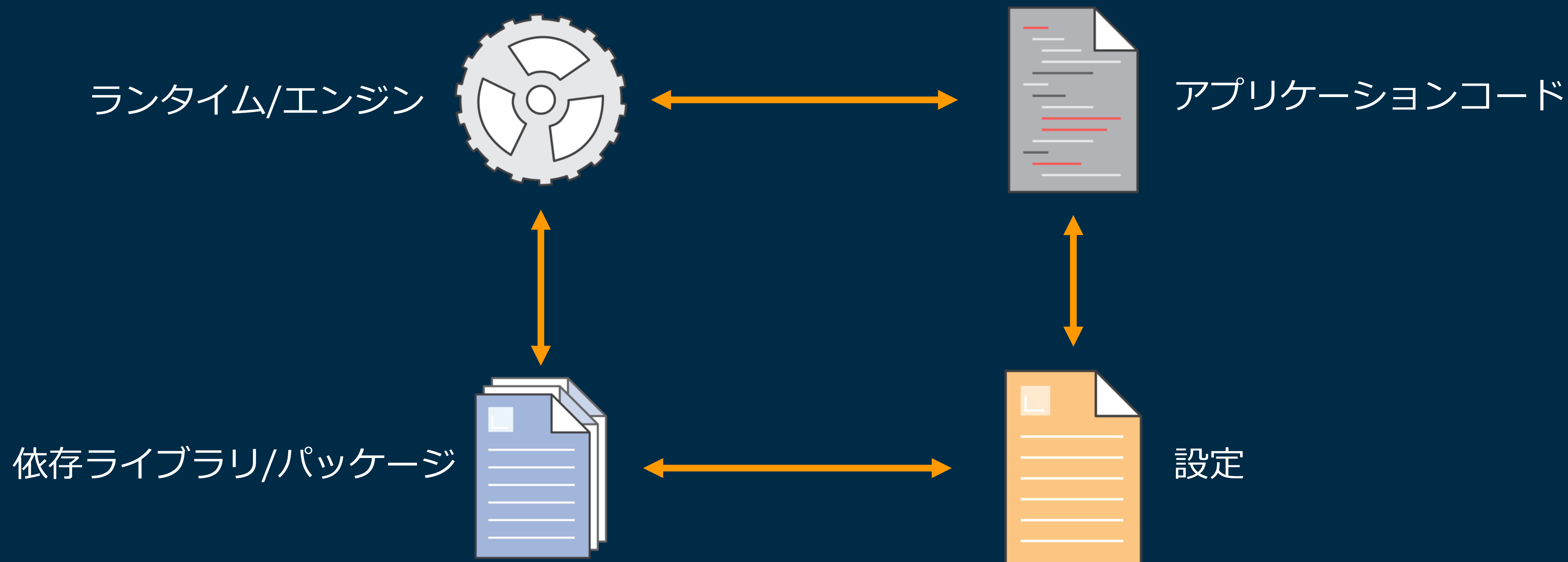
環境変数

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – アプリケーション

アプリケーションを構成するコンポーネント



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – 設定（アプリケーション）

設定

- アプリケーションが稼働する上で必要となる動作環境ごとに異なる情報
- 変更頻度が高い情報

設定例

- RDBの接続先
- 外部APIのエンドポイント
- デバッグフラグ
- 同時処理数、タイムアウト秒数

設定

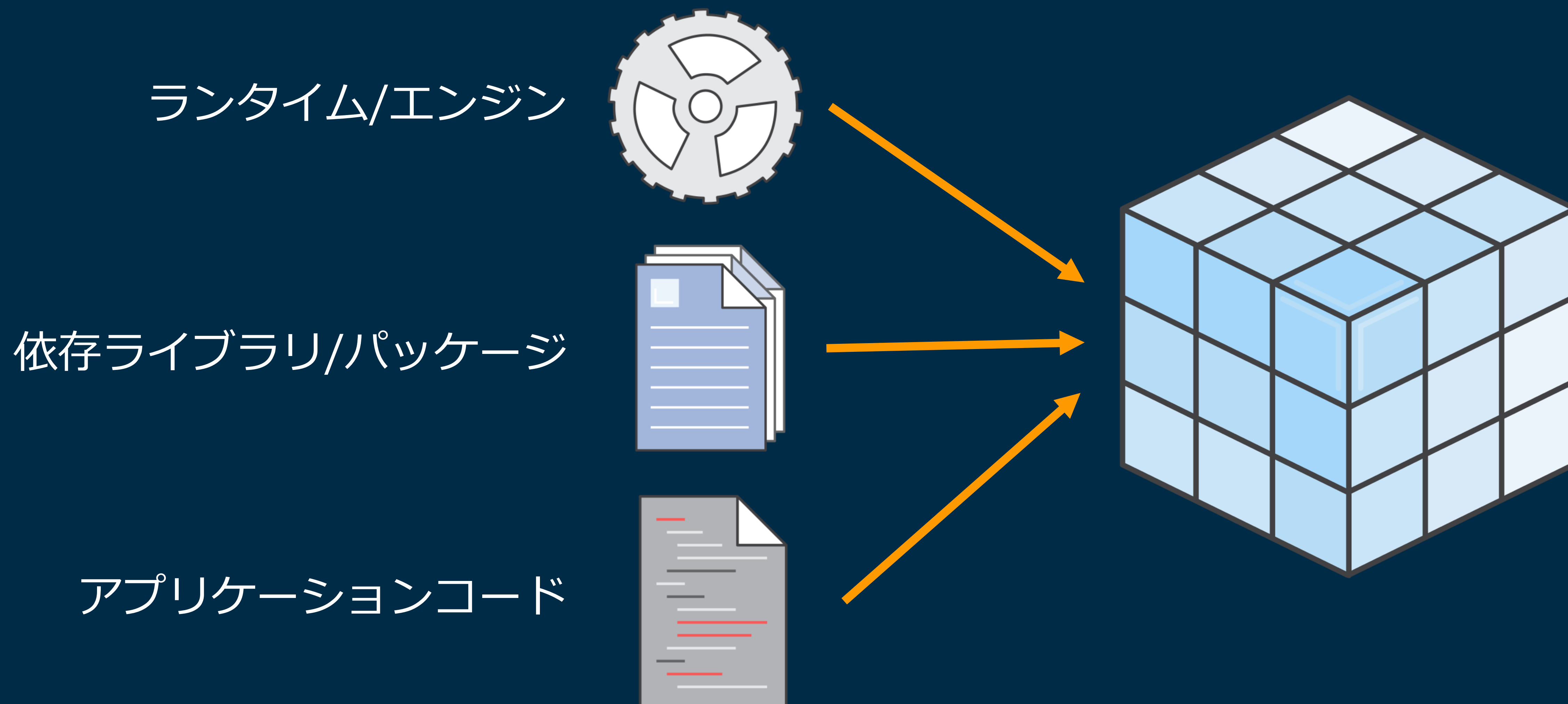
設定をソースコードに含めた場合の影響

- 変更の都度ビルド、テスト、デプロイ必要
- ソースコード、コンテナイメージの流出といったセキュリティリスク

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – コンテナ

コンテナ



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – 設定（コンテナ）

コンテナにおける設定

- 環境変数として外部から渡す

読み込み例

- 起動時にまとめて取得
- アプリケーションから必要な時に都度取得

AWSサービス

- AWS Systems Manager - Parameter Store
- AWS Secrets Manager
- Amazon S3

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – Parameter Store

Amazon ECS

Amazon ECSでのParameter Store利用方法

- タスク定義内のコンテナ定義 (containerDefinitions) の secrets で指定

```
"secrets": [{
  "name": "environment_variable_name",
  "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
}]
```

- タスク定義内のログ設定 (logConfiguration) の secretOptions で指定

```
"secretOptions": [{
  "name": "fluentd-address",
  "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter:parameter_name"
}]
```

注意点

- タスク起動時に環境変数として読み込む、更新した値を読む時は新しいタスクを起動
- 大量取得がある際はスループットに注意（デフォルト：40TPS、上限：1,000TPS）

AWS Systems Manager Parameter Store - AWS Systems Manager
https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

環境変数 – Secrets Manager

Amazon ECS

Amazon ECSでのSecrets Manager利用方法

- タスク定義内のコンテナ定義 (containerDefinitions) の secrets で指定
- タスク定義内のログ設定 (logConfiguration) の secretOptions で指定
- 完全なARNの指定以外に、EC2 起動タイプでは柔軟な指定が可能（2020/04）

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

  - json-key：指定すると特定のKeyのみ取得、指定がないと全取得
  - version-stage：AWSCURRENT（現在の値）AWSPREVIOUS （1つ前の値）を指定し取得
  - version-id：自動付与されるバージョンIDを指定して特定のデータを取得

注意点

- タスク起動時に環境変数として読み込む、更新した値を読む時は新しいタスクを起動
- 大量取得がある際は秒間取得上限に注意（GetSecretValue：3,000 per second 固定）

Specifying sensitive data using Secrets Manager - Amazon Elastic Container Service
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/specifying-sensitive-data-secrets.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

環境変数 – Amazon S3

Amazon ECS

Amazon ECSでのAmazon S3ファイルからの利用方法（EC2 起動タイプのみ）

- 従来からタスク定義内の environment で個別にパラメータ渡す事は可能
- タスク定義内の environmentFiles の secrets で S3 を指定可能（2020/05）

```
"environmentFiles": [
    {
        "value": "arn:aws:s3:::s3_bucket_name/envfile_object_name.env",
        "type": "s3"
    }
```

注意点

- タスク起動時に環境変数として読み込む、更新した値を読む時は新しいタスクを起動
- タスク定義ごとに最大 10 個まで指定可能

Specifying environment variables - Amazon Elastic Container Service
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/taskdef-envfiles.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel.

環境変数 – Secrets Manager

Amazon EKS

Amazon EKSでのSecrets Manager利用方法

- AWS Secrets Admission Controllerを利用した取得
  - K8sのmutating webhookを利用し /tmp/secret 以下に取得した secret を書き込み
- OSS の Kubernetes External Secrets を利用
  - External Secret Controller が取得した値を参照可能（Parameter Storeも可）

Kubernetes標準のSecretを利用した管理

デフォルトではbase64 Encodeしてetcdに保存

EKSではAWS KMSと連携しディスクレベルで暗号化した状態でetcdに保存可能（2020/03）

AWS Secrets Controller PoC: AWS Secrets Manager と Kubernetes の統合
https://aws.amazon.com/jp/blogs/news/aws-secrets-controller-poc/
godaddy/kubernetes-external-secrets: Integrate external secret management systems with Kubernetes
https://github.com/godaddy/kubernetes-external-secrets
高度な防御のために EKS 暗号化プロバイダーのサポートを利用する | Amazon Web Services ブログ
https://aws.amazon.com/jp/blogs/news/using-eks-encryption-provider-support-for-defense-in-depth/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws DEV DAY
ONLINE JAPAN

コンテナイメージ

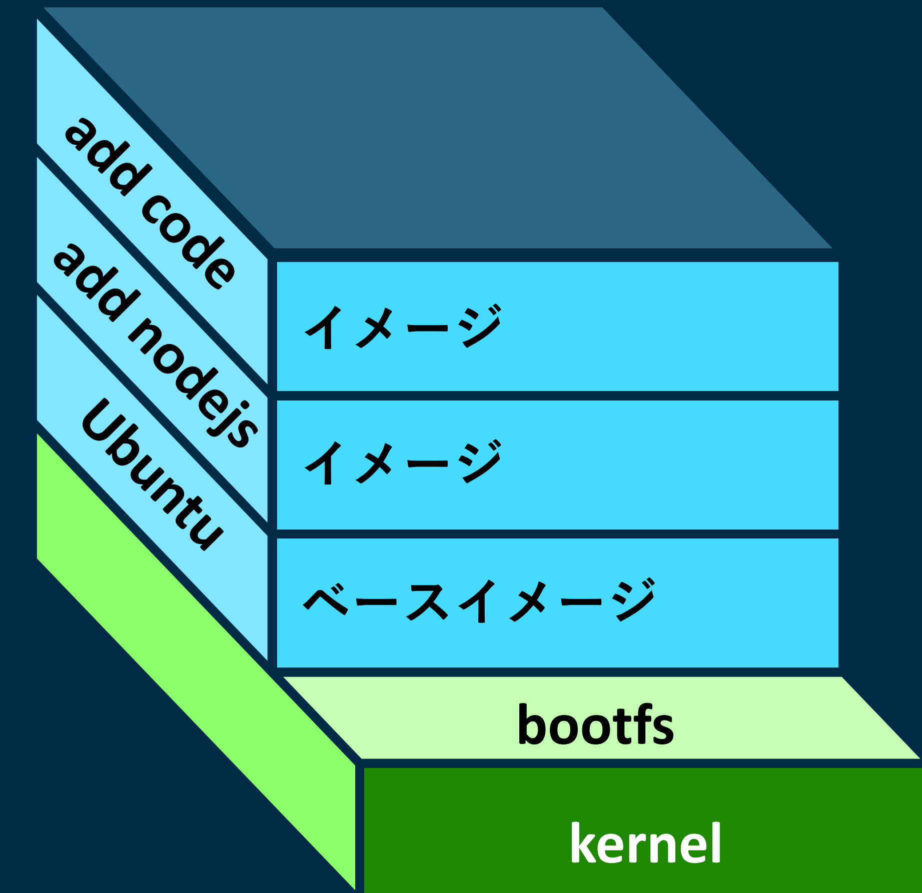
In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

コンテナイメージ – Docker Image

読み取り専用のイメージで
コンテナを起動するテンプレートとして利用

- 基本イメージ（ベースイメージ）の上に
  ライブラリやアプリケーションコードを追加
- 追加するイメージが**レイヤー（層）として積み上げられ**
  レイヤー構造を持つ1つのイメージを作成
- イメージを作成する命令は Dockerfile ※1に記述
- 出来上がったイメージはイミュータブル ※2



※1 Dockerfile：依存ライブラリのインストール、アプリケーションコードの追加を実現するコマンドを記述したテキストファイル
※2 イミュータブル：作成後に変更できないもの

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
In Partnership with intel

コンテナイメージ – デプロイ

コンテナをデプロイするとはどういう事か

- 指定したコンテナイメージが、実行環境で稼働しトラフィックを受け処理できる状態にする事

デプロイ時間の短縮は運用上で重要なポイント

- テスト時間・開発サイクルへの影響
- 新バージョン（新機能、Bug Fix）への更新
- スケールアウトへの影響

デプロイ時間の短縮に影響する部分

- コンテナイメージサイズ
- アプリケーション起動までの時間

aws

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

コンテナイメージ – イメージサイズ

コンテナイメージサイズを小さくするためのポイント

Base Image の選択

- 軽量コンテナ（ alpine, scratch, busybox, etc. ）
  サイズに拘りすぎて安定性を損なう事がないよう注意

アプリケーション開発言語

- 実行環境のためのファイル、パッケージが少ないものを選択
- シングルバイナリ（ Golang, etc. ）は相性が良い

Dockerfile の書き方に注意

- 不要なレイヤー、パッケージ、ファイルを削除
- .dockerignore を利用
- マルチステージビルドを利用

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

コンテナイメージ – その他

その他コンテナイメージで気をつけるポイント

実行ユーザー

- プロセスは root ユーザー以外で実行

パッケージング

- アップデートを起動後に実施せず、必要な全ファイルをイメージとしてパッケージ
  ダウンロード時間が影響、取得できないと起動不可

ネットワーク

- コンテナ実行環境にネットワーク的に近いイメージレジストリを利用
  例： 利用するコンテナサービスと同一リージョンの Amazon ECR

ARMアーキテクチャの利用を検討

- ARMアーキテクチャのAWS Graviton2を搭載したM6g、C6g、R6gインスタンスは
  同世代のx86ベースのインスタンスと比べてパフォーマンス向上＆20%コストダウン

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

コンテナイメージ – Dockerfile

Dockerfile の書き方次第でコンテナイメージのサイズに影響するだけでなく
ビルド時間やセキュリティにも影響

不要なコンテナレイヤーは作成しない

- コンテナイメージは複数のレイヤーにより構成され
  Dockerfile のコマンド RUN, COPY, ADD はレイヤーを作成
- まとめられる RUN はまとめる（パッケージ導入の時は可能ならミスを減らすためアルファベット順）

```
RUN apt-get update && apt-get install -y ¥
  bzr ¥
  cvs ¥
  git
```

Best practices for writing Dockerfiles
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with

コンテナイメージ – Dockerfile

不要なパッケージはインストールしない

- 全体としてのサイズだけでなく、脆弱性があった場合にはセキュリティリスクにもつながる
- apt の場合 --no-install-recommends といった推奨パッケージを入れないオプション

不要なファイルは削除

- パッケージをインストール後にリストやキャッシュが残るので削除
  - apt の場合 (apt-get clean && rm -rf /var/lib/apt/lists/*)
  - yum の場合 (yum clean all)
  - apk の場合 (--no-cache)

Best practices for writing Dockerfiles
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with

コンテナイメージ – Dockerfile

.dockerignore を利用

- 実行する Dockerfile が置かれているディレクトリ配下にある余計なファイル、ディレクトリを処理の対象から除外

ビルドキャッシュを活用

- Dockerfile に記述された順番に処理される
- ファイルのチェックサム比較で異なる場合などキャッシュが利用できない場合は、変更された行以降が再実行
- 頻繁に更新されるものほど Dockerfile の後半に記述

プロセスは root ユーザー以外で実行

- アプリケーション動作に不要な権限は渡さない

Best practices for writing Dockerfiles
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with

コンテナイメージ – Dockerfile

マルチステージビルドを利用

- アプリケーションのビルド環境とアプリケーション実行環境では、必要なパッケージが異なる
- 以前は環境別に Dockerfile を用意し、bash scriptを使いビルド環境イメージでビルドした実行ファイルをローカルを経由で本番イメージに渡すビルダーパターンが一般的だったが複雑
- マルチステージビルドでは、１つの Dockerfile 内で複数のビルドをステージという単位で実行し、ステージ間でファイルをコピー
- マルチステージビルドにより大幅に処理を簡素化
（特定ステージだけのビルド、外部イメージをステージとして指定が可能

Best practices for writing Dockerfiles
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with

コンテナイメージ – Dockerfile

ARMアーキテクチャの利用を検討

- ARMアーキテクチャのAWS Graviton2を搭載したM6g、C6g、R6gインスタンスは同世代のx86ベースのインスタンスと比べて最大 40% 高いコストパフォーマンス

- AWSコンテナサービス対応状況
  - AWS Batch 対応済
  - AWS CodeBuild ARMイメージ選択可
  - ECS 対応済
  - ECR マルチアーキテクチャイメージに対応済（2020/05）
  - EKS 対応済（2020/08）

Pushing a multi-architecture image - Amazon ECR
https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-push-multi-architecture-image.html
Leverage multi-CPU architecture support | Docker Documentation
https://docs.docker.com/docker-for-mac/multi-arch/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

コンテナイメージ – 脆弱性スキャン

コンテナ内のソフトウェア（パッケージ）の脆弱性を検出

動的スキャン (Dynamic scanning)

- ランタイム環境で実行されるスキャン
- 実行中のコンテナにある脆弱性を特定可能
- 後日発覚した脆弱性や、ゼロデイの脆弱性なども検出可能
- OSS の CNCF Falco, APNパートナーの Aqua Security, Trend Micro, Twistlock など

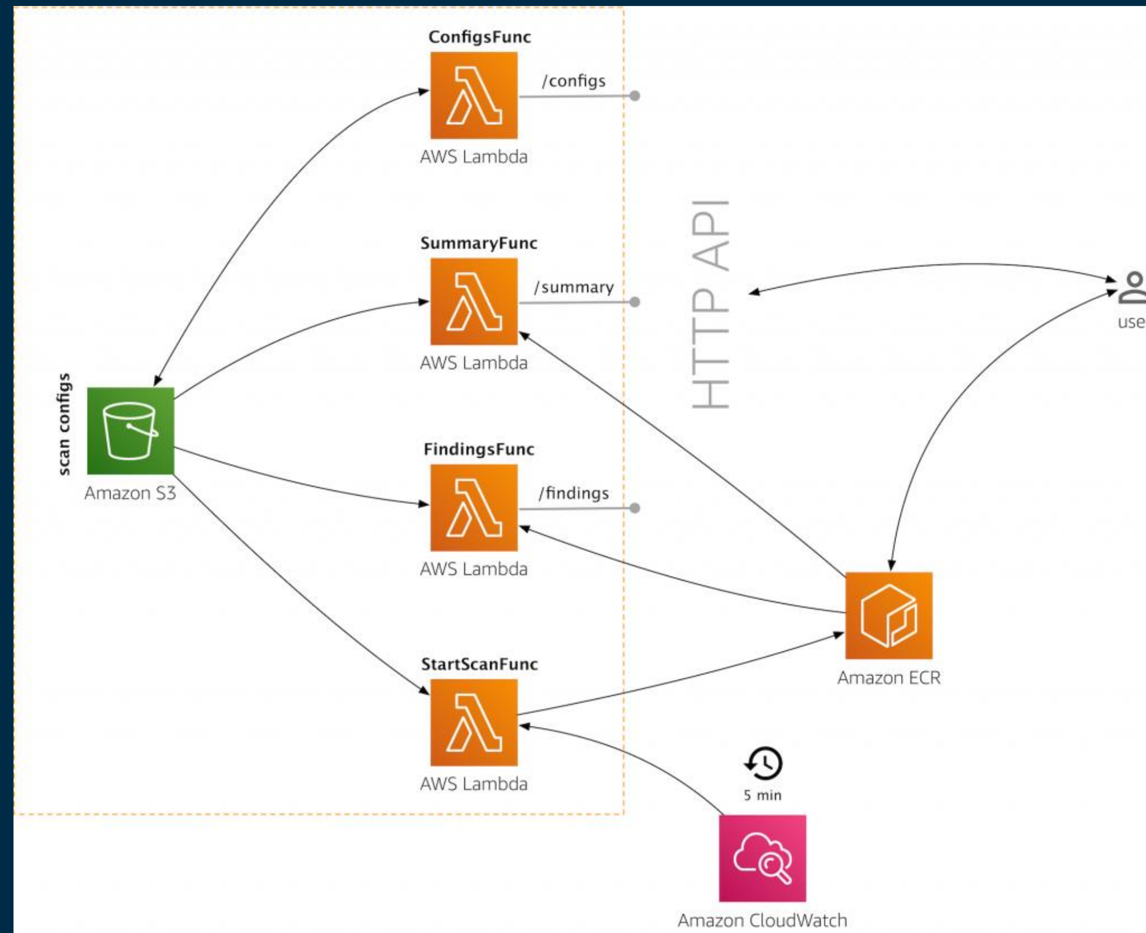
静的スキャン (Static scanning)

- デプロイ前のフェーズで実行されるスキャン
- コンテナが実行される前に脆弱性を特定可能
- コンテナイメージ内の OS パッケージをスキャンし共通脆弱性識別子 (CVE) を検出
- ECR のイメージスキャン機能はこちらに該当

aws
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
In Partnership with intel.

コンテナイメージ – 脆弱性スキャンの定期実行

Amazon ECRのイメージスキャンは定期スキャンの実施を推奨



4つの Lambdaにより構成

- ConfigFunc
  スキャン対象に関する管理（ S3 に保存 ）
- SummaryFunc
  スキャン結果のサマリを提供
- FindingsFunc
  スキャン詳細結果の Atom フィードを提供
- StartScanFunc
  スキャン実行（CloudWatch Event から5分毎に実行）

ECR Container Image Re-Scan
https://github.com/aws-samples/amazon-ecr-continuous-scan

Amazon ECRのネイティブなコンテナイメージスキャン機能について
https://aws.amazon.com/jp/blogs/news/amazon-ecr-native-container-image-scanning/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

aws DEV DAY ONLINE JAPAN

ログ

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ログ – 目的

ログから何をしたいか**目的**に合わせた仕組みを検討

目的例

- 保管目的
  - 障害やセキュリティインシデント時の調査

- 分析目的
  - ユーザーへ提供するレポート（バッチ分析）
  - カスタマーサポート（アドホック分析）

- 監視目的
  - 処理のエラーなど異常を検知してアラート
  - レスポンスタイムなどサービスの状態をダッシュボードで確認

aws

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

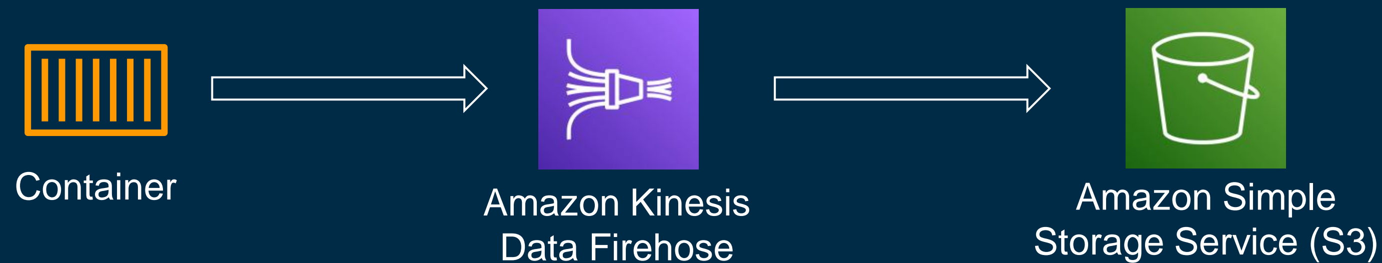
In Partnership with intel

ログ – 保管目的

保管目的

- データ量が多く保管期間も長いが利用頻度は低い
- S3 で保管
  - 99.999999999の耐久性
  - ライフサイクルによる削除や安価なストレージクラスへの変更
  - 豊富AWSサービスとのインテグレーション

AWS 構成例



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ログ – 分析目的

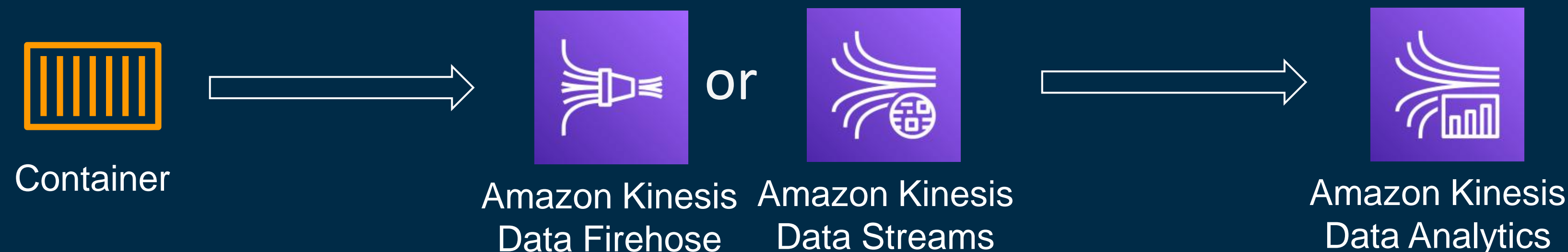
調査目的

- データ量、保管期間、利用頻度が中程度～多い
- 要件次第で分析の速度（リアルタイム、特定時間以内）が求められる
- 要件に合わせてAWSサービスとのインテグレーション

AWS 構成例

- 分析（バッチ、アドホック）

- 分析（リアルタイム）



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

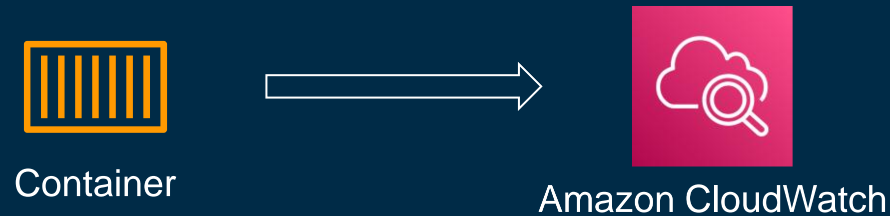
ログ – 監視目的

監視目的

- データ量少ない、保管期間短い、リアルタイム性が求められる

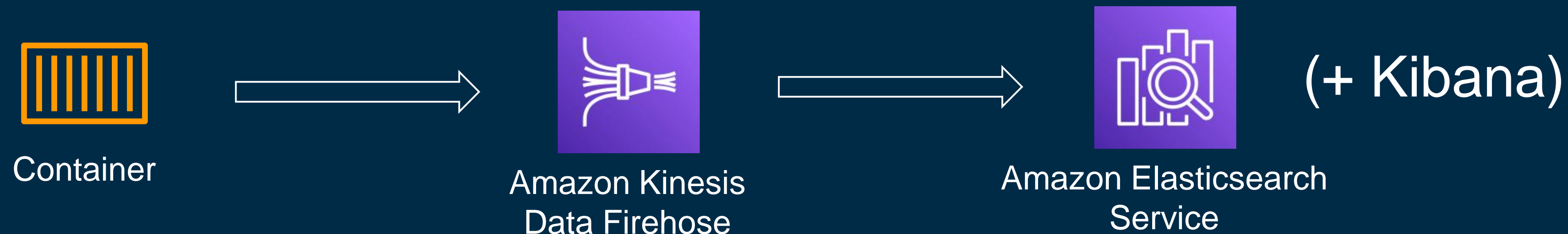
AWS 構成例

- アラート



※ エラーログやレスポンスタイム、エラーレートをカスタムメトリクスとして連携

- リアルタイム監視



(+ Kibana)

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ログ – FireLens

Amazon ECS

コンテナアプリケーションからのログ取得方法（Amazon ECS）

- FireLens という選択肢
  - Fluentd、 Fluent Bitを利用したログの収集と連携
  - Fluent Bit はAWS提供イメージあり（CloudWatch, Kinesis Data Firehose などプラグイン入り）
  - タスク定義パラメータで awsfirelens ログドライバーを指定

```
"logConfiguration": {
        "logDriver":"awsfirelens",
        "options": {
                "Name": "firehose",
                "region": "us-west-2",
                "delivery_stream": "my-stream"
        }
},
```

Custom log routing - Amazon Elastic Container Service
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/using_firelens.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

ログ – FireLens

Amazon ECS

カスタム Fluentd / Fluent Bit 設定ファイル

- S3 または File（コンテナから参照可能なPath）により設定ファイルを指定

```
{
  "containerDefinitions":[
    {
      "essential":true,
      "image":"906394416424.dkr.ecr.us-west-2.amazonaws.com/aws-for-fluent-bit:latest",
      "name":"log_router",
      "firelensConfiguration":{
        "type":"fluentbit",
        "options":{
          "config-file-type":"s3 | file",
          "config-file-value":"arn:aws:s3:::mybucket/fluent.conf | filepath"
        }
      }
    }
  ]
}
```

正規表現を使用したフィルタリング

- ログの内容に応じたログのフィルタリングをタスク定義内でサポート

Custom log routing - Amazon Elastic Container Service
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/using_firelens.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ログ – コンテナ

Amazon EKS

コンテナアプリケーションからのログ取得方法（Amazon EKS）

- Fluentd、 Fluent Bitを利用したログの収集と連携
  - EC2 Worker Nodes：リソース効率を考え各ノードに Daemonset として配置や権限分離などの目的でPodに Sidecarとして配置
- 設定ファイルは ConfigMap を利用して設定

YAML


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: fluent-bit-config
  labels:
    app.kubernetes.io/name: fluentbit
data:
  fluent-bit.conf: |
    [SERVICE]
        Parsers_File  parsers.conf
    [INPUT]
        Name              tail
        Tag               kube.*
        Path              /var/log/containers/*.log
        Parser            docker
        DB                /var/log/flb_kube.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
```


Fluent Bit による集中コンテナロギング
https://aws.amazon.com/jp/blogs/news/centralized-container-logging-fluent-bit/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

ログ – Fluentd /Fluent Bit

あくまで参考値（2019/7/31公開のBlog記事より c5.9xlargeでの検証）

CloudWatch プラグイン: Fluentd vs Fluent Bit

Log Lines Per second	Data Out	Fluentd CPU	Fluent Bit CPU	Fluentd Memory	Fluent Bit Memory
100	25 KB/s	0.013 vCPU	0.003 vCPU	146 MB	27 MB
1000	250 KB/s	0.103 vCPU	0.03 vCPU	303 MB	44 MB
10000	2.5 MB/s	1.03 vCPU	0.19 vCPU	376 MB	65 MB

Kinesis Firehose プラグイン: Fluentd vs Fluent Bit

Log Lines Per second	Data Out	Fluentd CPU	Fluent Bit CPU	Fluentd Memory	Fluent Bit Memory
100	25 KB/s	0.006 vCPU	0.003 vCPU	84 MB	27 MB
1000	250 KB/s	0.073 vCPU	0.033 vCPU	102 MB	37 MB
10000	2.5 MB/s	0.86 vCPU	0.13 vCPU	438 MB	55 MB

CloudWatchプラグイン

Fluentd は Fluent Bit と比べて
CPU4倍以上、メモリ6倍以上消費

Kinesis Firehoseプラグイン

Fluentd は Fluent Bit と比べて
CPU3倍以上、メモリ4倍以上消費

処理内容やバージョンなど様々な要素があるので検証推奨

Fluent Bit による集中コンテナロギング
https://aws.amazon.com/jp/blogs/news/centralized-container-logging-fluent-bit/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

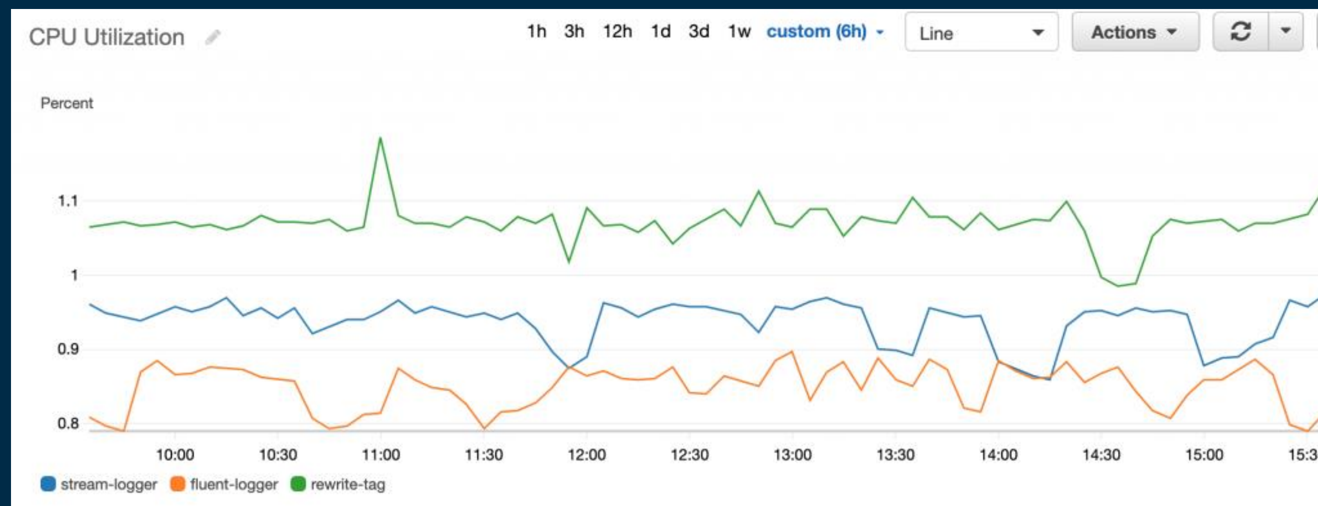
In Partnership with intel

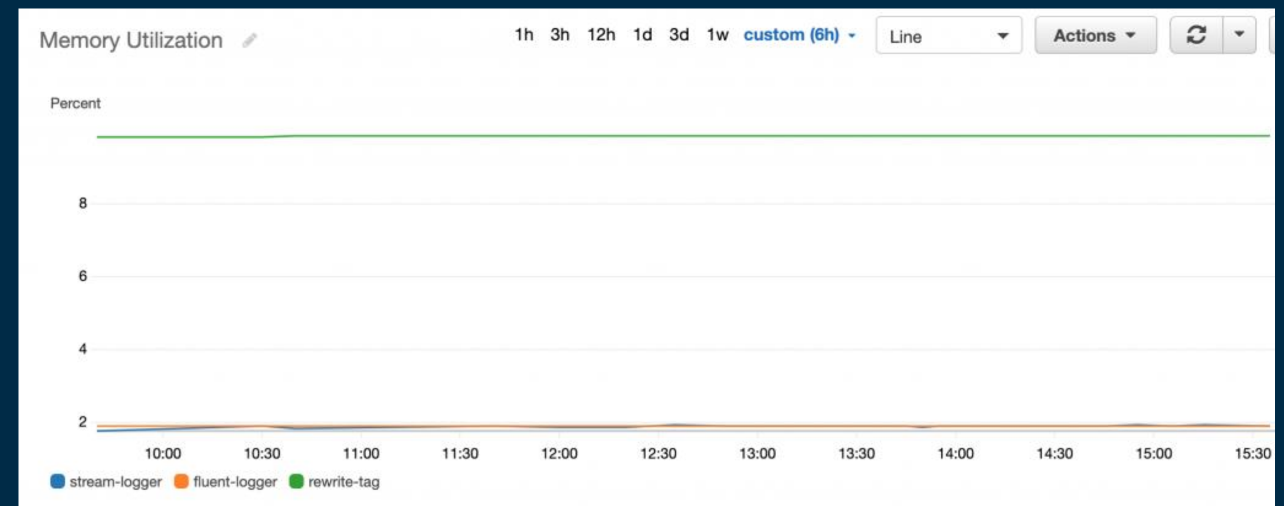
ログ – Fluent / Fluent Bit 複数ログの取り扱い

コンテナで複数のログを出力していて処理を分けたい

- 選択肢
  - Fluent Bit の Stream Processor を利用
  - Fluentd の Rewrite Tag Filter を利用
  - Fluent Logger Libraryでの Fluentd / Flunet Bit 利用（検証では fluent-logger-golang 利用）

あくまで参考値（2019/11/20公開のBlog記事より c5.9xlargeでの検証）





Splitting an application’s logs into multiple streams: a Fluent tutorial
https://aws.amazon.com/jp/blogs/opensource/splitting-application-logs-multiple-streams-fluent/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

モニタリング

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

モニタリング– メトリクス

ステータス（現在の状況）をメトリクスとして取得

取得例

- リソース（CPU・メモリ使用量）、稼働コンテナ数
- アクセス数、レスポンス時間、エラー率
- ログからのエラー検知
- デプロイ時刻・バージョン情報

利用例

- アラート
- スケーリング
- コネクティビティ
- 可視化

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel.

モニタリング– アラート

アラート

- 異常な状態を検知
- 人による「確認」や「対応」が必要

設定＆検知例

- 処理にエラーが発生した
- 特定のバージョンをデプロイしてからエラー率が上昇
- 平均レスポンスタイムがユーザー影響が出るレベルで上昇
- 他のサービスとの通信に失敗してエラーが発生
- 想定外の入力データにより処理にエラーが発生

aws

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel.

モニタリング– スケーリング

スケーリング

- コンテナに適切なリソースを割当る（スケールダウン/アップ）
- 処理、アクセスに応じてコンテナの数を調整する（スケールイン/アウト）

設定例

- CPU・メモリ使用率に基づくコンテナのリミット設定
- CPU・メモリ予約率に基づくスケーリング設定
- イベント時間に合わせたスケーリング設定
- アクセス量、キューのメッセージ数に基づくスケーリング設定

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

モニタリング– コネクティビティ

コネクティビティ

- ユーザーからサービスが利用できる状態になっているか
- 外形監視による接続性・レスポンスタイムの確認

障害例

- ネットワーク障害によりアクセスできない
- 設定変更後もDNSが古いレコードを返していてアクセスできない
- 特定のバージョンをデプロイ後からレスポンスタイムが悪化

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

モニタリング– CloudWatch

CloudWatch

メトリクス

- CloudWatch Metrics 及び Container Insights によるメトリクス収集とダッシュボード
- メトリクスデータの保存期間は最大15ヶ月（455日）

アラート

- 単一または複数のメトリクスに基づくアラーム、複合アラーム
- アラームのイベントからSNS や Chatbot を利用した通知

スケーリング

- CloudWatch Metrics と Auto Scaling Group によるスケーリング（Amazon ECS）
- Target Tracking Scaling, Step Scaling, Scheduled Scaling

コネクティビティ

- CloudWatch Synthetics による監視 (2020/04 GA)

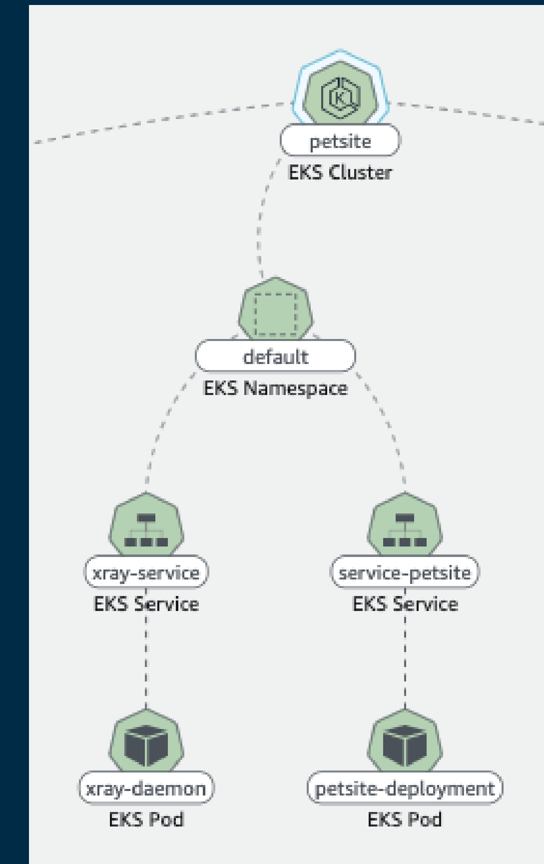
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

モニタリング– Container Insights

Container Insights

- コンテナに関するメトリクス収集
  - リソースに関する情報
  - コンテナの再起動の失敗などの診断情報
  - 埋め込みメトリクス（ログからカスタムメトリクスを生成）
    - パフォーマンスログをCloudWatch Logs Insights から確認可能
  - リソースのマップ表示

- CloudWatchダッシュボードから確認可能
- Amazon ECS、Amazon EKS、K8s on EC2に対応
- OSSモニタリングの Prometheus メトリクスの検出



ContainerInsights/Prometheus
23 Metrics

Using Container Insights - Amazon CloudWatch
https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/ContainerInsights.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

モニタリング – CloudWatch Synthetics

CloudWatch Synthetics

- Node.js スクリプトでカナリアを定義（テンプレートスクリプトあり）
- Puppeteer ライブラリを使用して Headless Chrome が指定頻度でアクセス
- レイテンシーのチェック、読み込み時間データ、UI のスクリーンショットを保存
- 結果がしきい値を超えたらCloudWatch と連携しアラーム

HTTP or HTTPS

設計図を使用する
テンプレートスクリプトから作業する

スクリプトをアップロードする
独自のスクリプトから作業する

S3 からインポート
S3 から既存のスクリプトを使用する

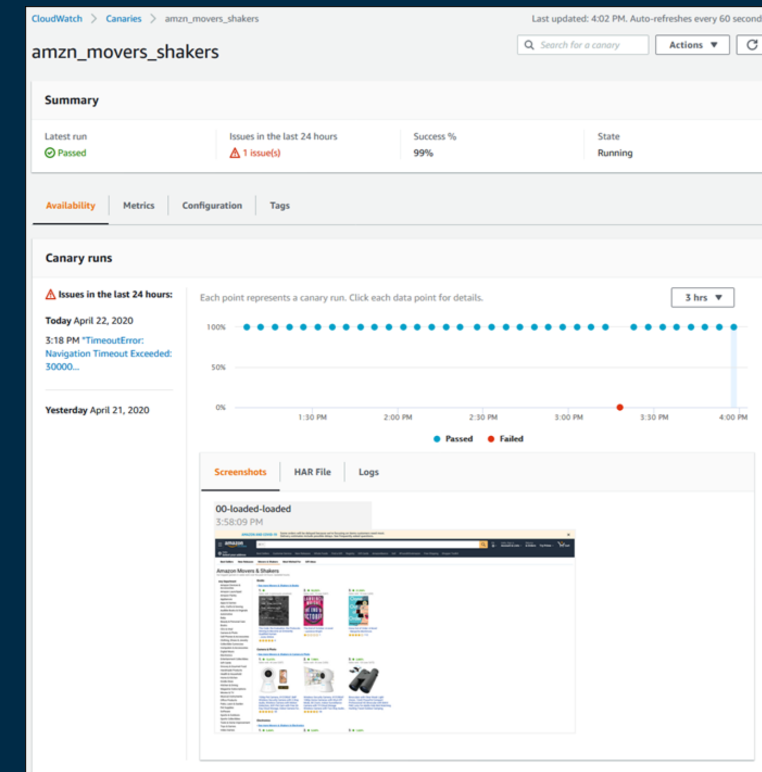
設計図

ハートビートのモニタリング
1 つの URL で基本的なページのロードを実行します。

API Canary
API をモニタリングします。

リンク切れチェッカー
指定された URL で基本的なウェブクローラーを実行し、アクセスした最初の破損したページを報告します。

GUI ワークフロービルダー
ウェブページで実行するアクションと検証を含む GUI ワークフローを作成します。



Using Synthetic Monitoring
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Synthetics_Canaries.html

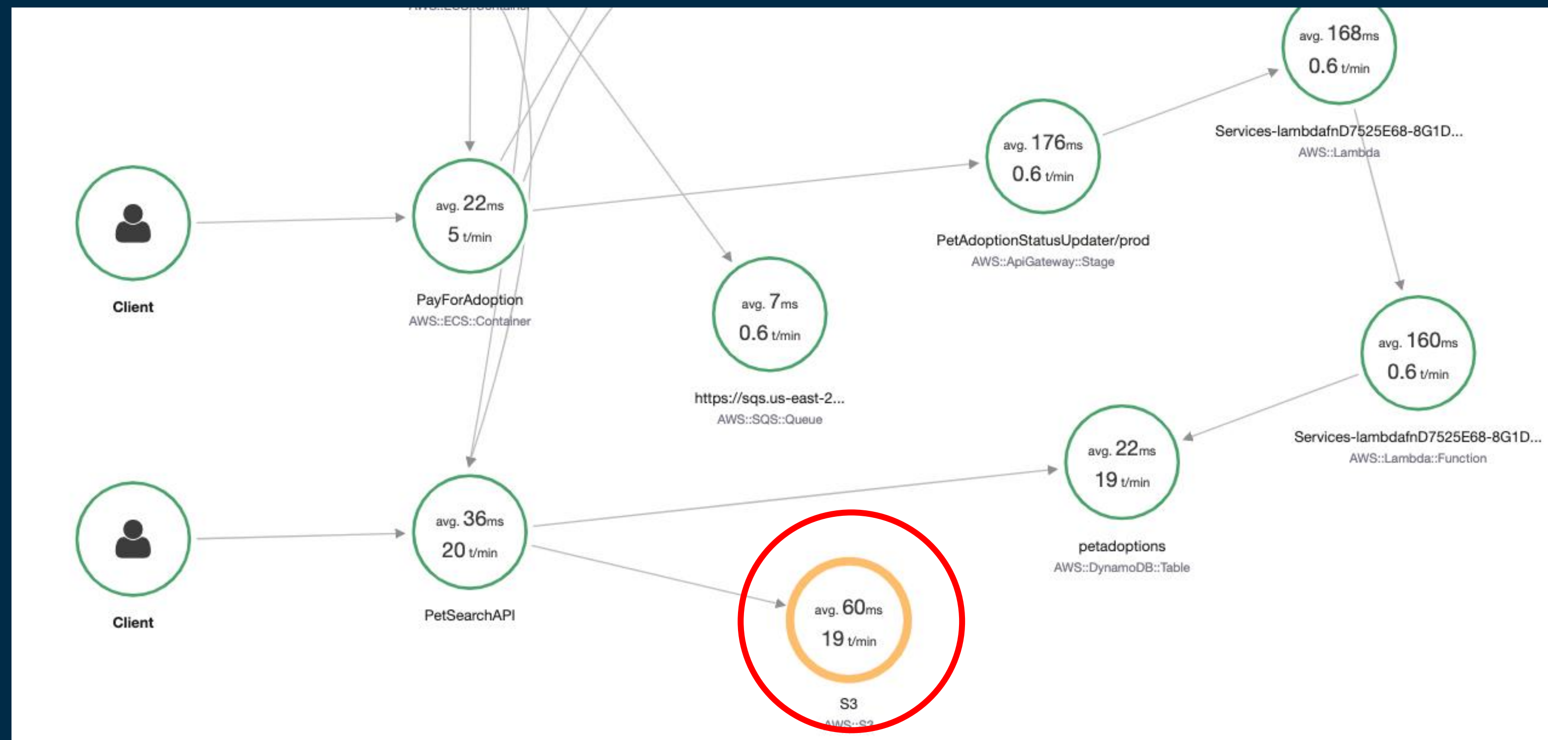
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

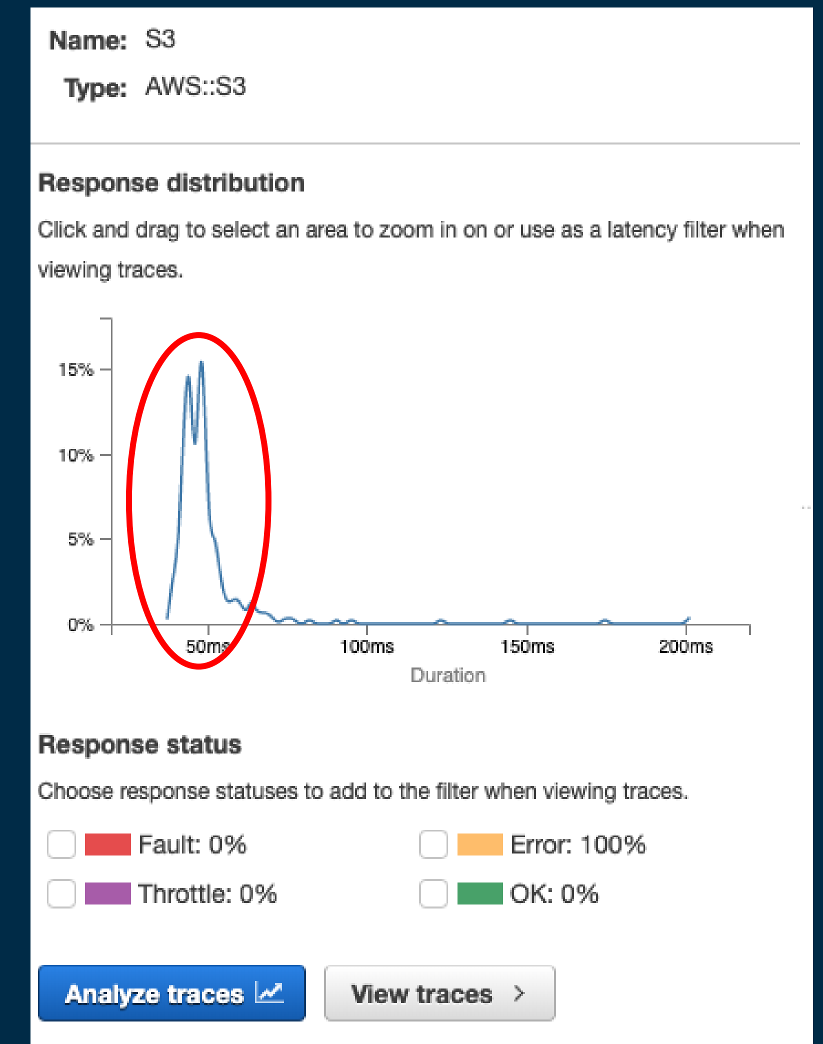
モニタリング – AWS X-Ray

アプリケーションが処理するリクエストに関するデータを収集

- 一意なトレースIDを引き継ぐことで関連付け
- パフォーマンスの問題やエラーの根本原因の特定に利用



S3拡大



What is AWS X-Ray? - AWS X-Ray
https://docs.aws.amazon.com/ja_jp/xray/latest/devguide/aws-xray.html

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws DEV DAY
ONLINE JAPAN

まとめ

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

まとめ

環境変数

- 設定としてどこまで切り出すか
- 設定をどのように管理・利用するか

コンテナイメージ

- デプロイ時間を減らすために何ができるか
- セキュリティのために何ができるか

ログ

- ログの目的・用途に合わせてAWSサービスは何を組み合わせるか
- ログの出力方法やログコレクターには何を使うか

監視

- 必要なメトリクスが収集されているか
- アラート、スケーリング、コネクティビティの観点からどのように設計・設定するか

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Appendix

Amazon ECS Workshop for AWS Fargate
https://ecsworkshop.com/

Amazon EKS Workshop
https://www.eksworkshop.com/

containers-roadmap
https://github.com/aws/containers-roadmap

One Observability デモ ワークショップ
https://observability.workshop.aws/ja/

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

Thank you!

Fumihide Nario

In Partnership with intel

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws DEV DAY
ONLINE JAPAN

Please complete the session survey in the mobile app.

In Partnership with intel.

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.