マルチアーキテクチャコンテナのビルドとデリバリー

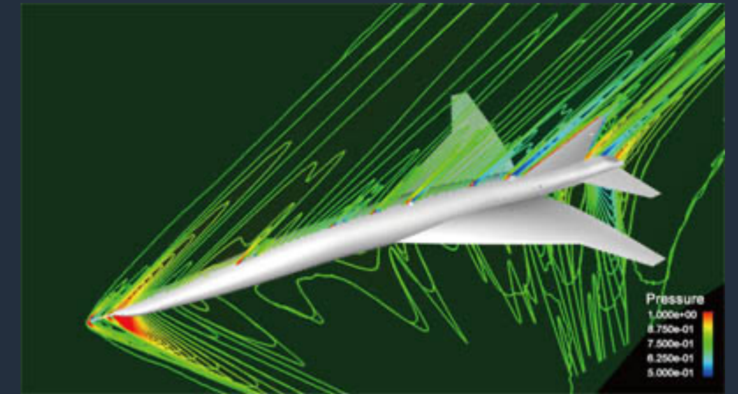
Hidekazu Karino

Amazon Web Services Japan, G.K.
Solutions Architect

© 2023, Amazon Web Services, Inc. or its affiliates.

苅野 秀和 (Hidekazu Karino)
https://twitter.com/hkford3

実物



研究対象だった飛行機

経歴

- 学生時代は飛行機 × 機械学習の研究
- クラウド楽しいと思って AWS へ

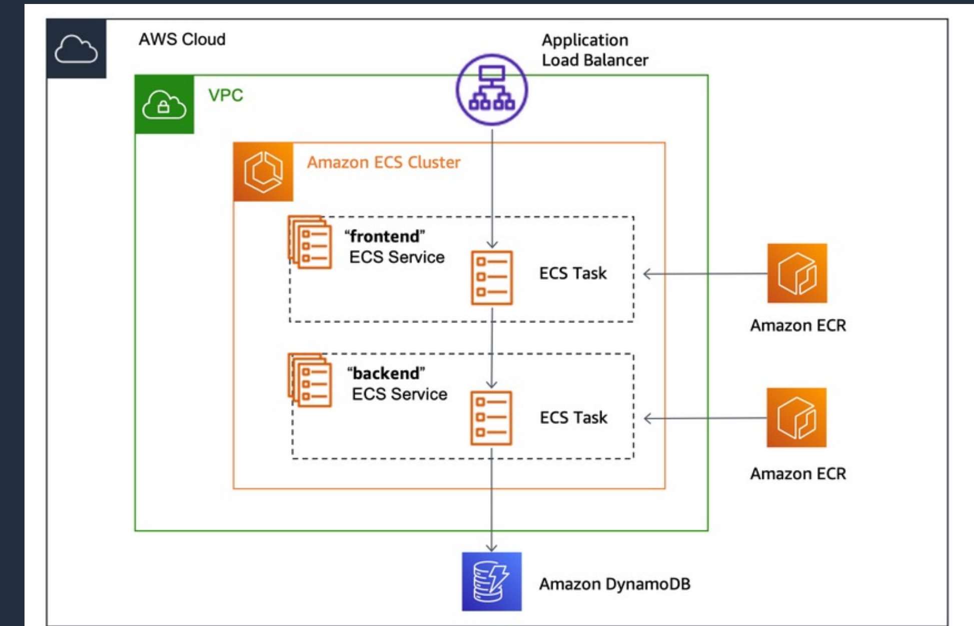
触ったことのある言語

- Python, TypeScript, Julia, Golang, Rust etc
- 学生時代は Python や Julia で機械学習のコード書いてました
- TypeScript で React アプリ書いたり Flask で API 書いたりしてます
- 最近のブームは Rust と OpenTelemetry

好きな AWS サービスやツール

- AWS CDK,AWS Copilot CLI
- ECS 環境を簡単に作れる Copilot CLI のワークショップを作りました

Zenn に記事をいくつか投稿してます: https://zenn.dev/hkdord

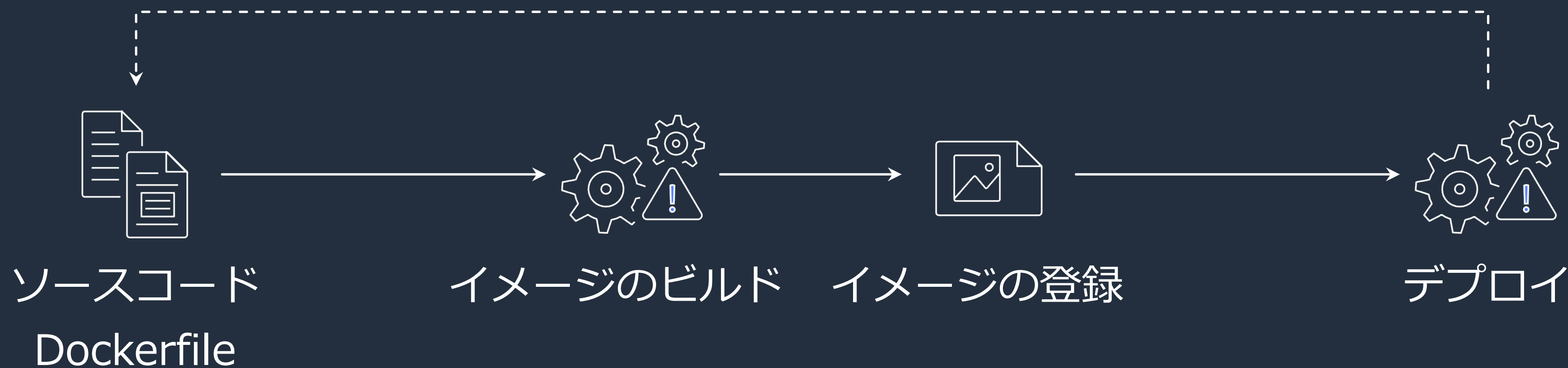


こういった構成を Copilot CLI
だと簡単に作れます

© 2021, Amazon Web Services, Inc. or its Affiliates.

aws

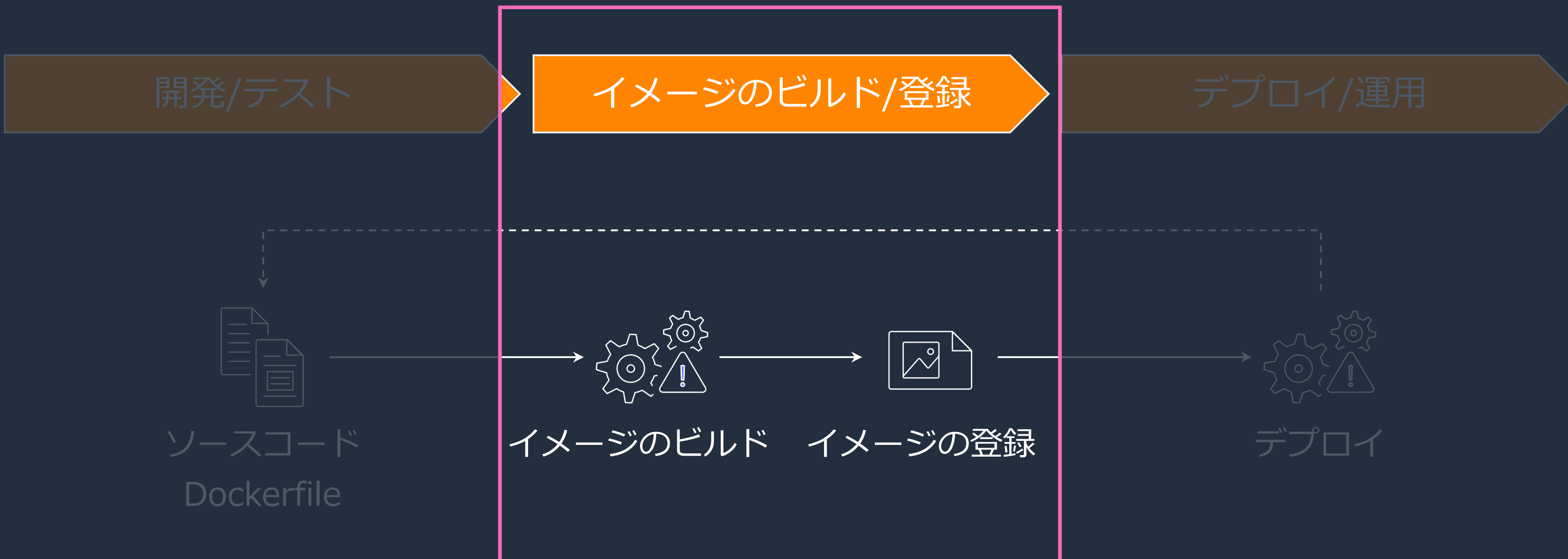
コンテナにおけるソフトウェア開発ライフサイクル



© 2023, Amazon Web Services, Inc. or its affiliates.

本セッションでお話すること

AWS Graviton と Intel 両方の CPU アーキテクチャ
に対応したコンテナイメージについてご紹介します



© 2023, Amazon Web Services, Inc. or its affiliates.

お品書き

- マルチアーキテクチャコンテナイメージの仕組み
- コンテナイメージのビルド
- デモ: ECS Fargate でマルチアーキテクチャコンテナイメージを動かす

© 2023, Amazon Web Services, Inc. or its affiliates.

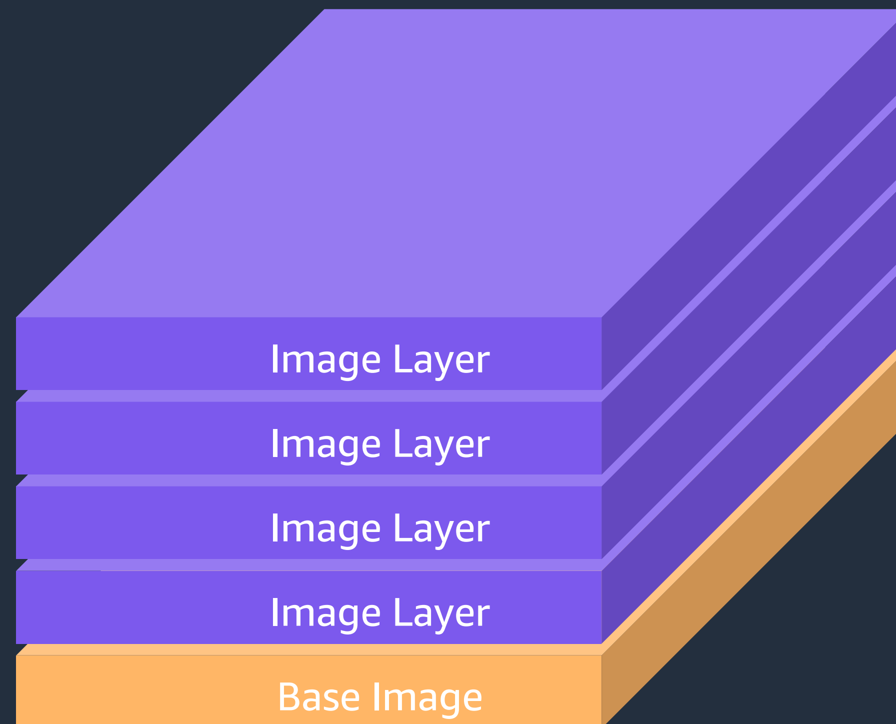
マルチアーキテクチャ
コンテナイメージの仕組み

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

その前にコンテナイメージとは

ベースイメージにイメージレイヤが積み重ねられたもの

- ベースイメージ
  - イメージの基礎
  - Ubuntu, Debian, Alpine Linux, etc
- イメージレイヤ
  - プログラムやライブラリ、メタデータを含む
  - Dockerfile の各命令が一つのレイヤに相当



© 2023, Amazon Web Services, Inc. or its Affiliates.

aws

コンテナイメージマニフェスト

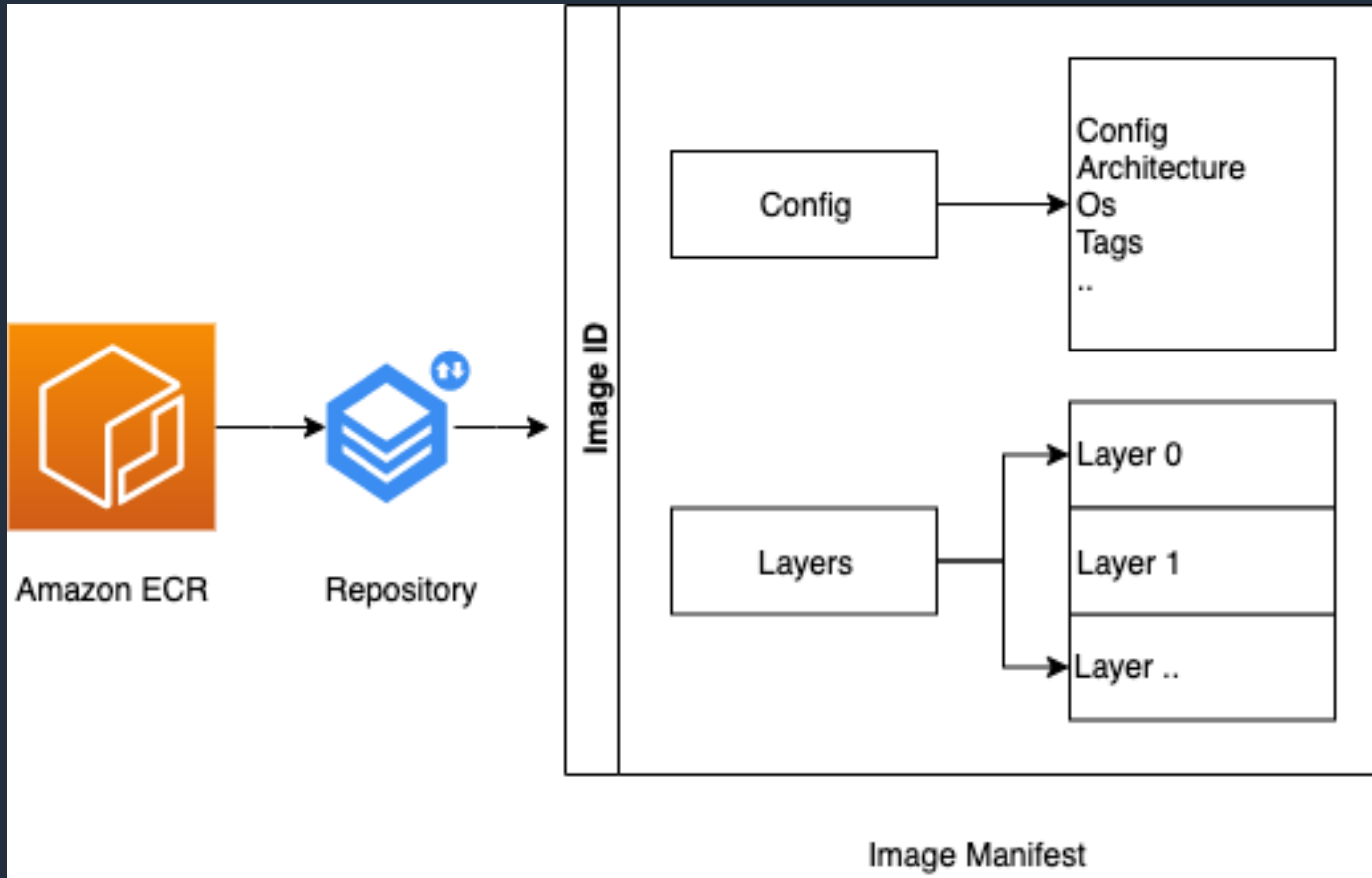

```
1  {
2    "schemaVersion": 2,
3    "mediaType": "application/vnd.docker.distribution.manifest.v2+js[illegible]
4    "config": {
5      "mediaType": "application/vnd.docker.container.image.v1+json",
6      "size": 1686,
7      "digest": "sha256:179def435e73b0b7863d1f30041ad20e5687a8f26716e07bd1d4eaee82885b8f"
8    },
9    "layers": [
10     {
11       "mediaType": "appl[illegible]ion/vnd.docker.image.rootfs.diff.tar.gzip",
12       "size": 61976108,
13       "digest": "sha256:5263c4cb36ce7acd[illegible]502b376a281d7a6075ad09beb23ac02a7668c"
14     },
15     {
16       "mediaType": "application/vnd.docker.image.rootfs.diff.tar.g[illegible]
17       "size": 170,
18       "digest": "sha256:61f77c3cbd6fb9d527236b0147b256d5299a2e1ba1a4f3583118740[illegible]
19     }
20   ]
21 }
```


コンテナイメージの configuration
CPU アーキテクチャもここに含まれる

コンテナイメージの各レイヤー

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

リポジトリにイメージを保存すると



コンテナイメージを pull するときはまずイメージマニフェストをpullしてきて、そこに記載されている各レイヤーを順番にダウンロードする。
イメージマニフェストの中の config にアーキテクチャが記載されている。

→ ひとつのコンテナイメージと CPU アーキテクチャが一対一で対応している

https://aws.amazon.com/jp/blogs/containers/introducing-multi-architecture-container-images-for-amazon-ecr/ より

© 2023, Amazon Web Services, Inc. or its affiliates.

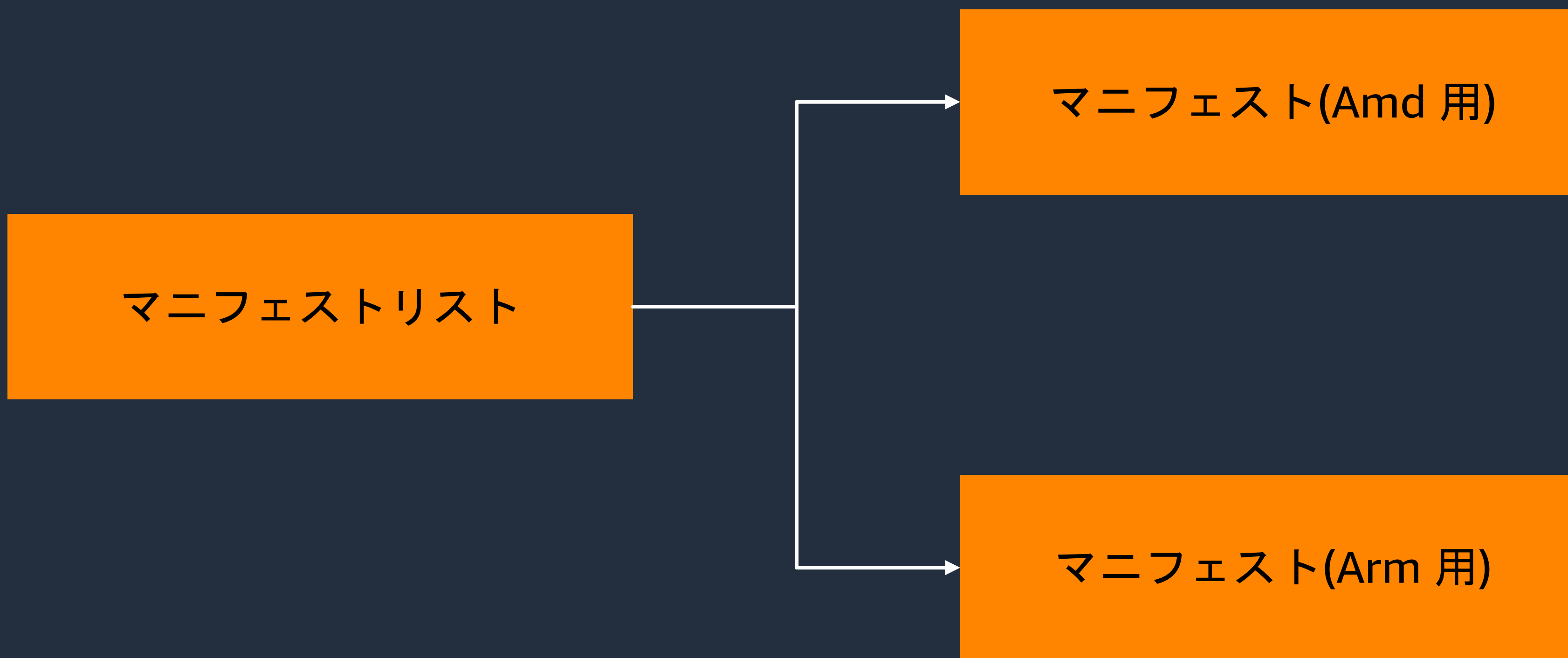
ホストとコンテナのCPUアーキテクチャ対応

- ホストマシンのアーキテクチャごとにコンテナイメージが必要

© 2023, Amazon Web Services, Inc. or its affiliates.

そこで登場するイメージマニフェストリスト

複数のコンテナイメージマニフェストを参照するもの



© 2023, Amazon Web Services, Inc. or its Affiliates.

コンテナイメージマニフェストリスト

```
dockerbuildx imagetools  inspect --raw <AWS account id>.dkr.ecr.<region>
.amazonaws.com/multi-architecture:v1

{
  "mediaType": "application/vnd.oci.image.index.v1+json",
  "schemaVersion": 2,
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:d41f23e8a9e3728c0e32f5df3a7ea92888d67a413e9210d9e079b4cfd1f3387f",
      "size": 3331,
      "platform": {
        "architecture": "arm64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:97f05d800c5e059af2662d12a912c77364e79e891d34f329d46ec1d070845167",
      "size": 3331,
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    },
    }
  ]
}
```

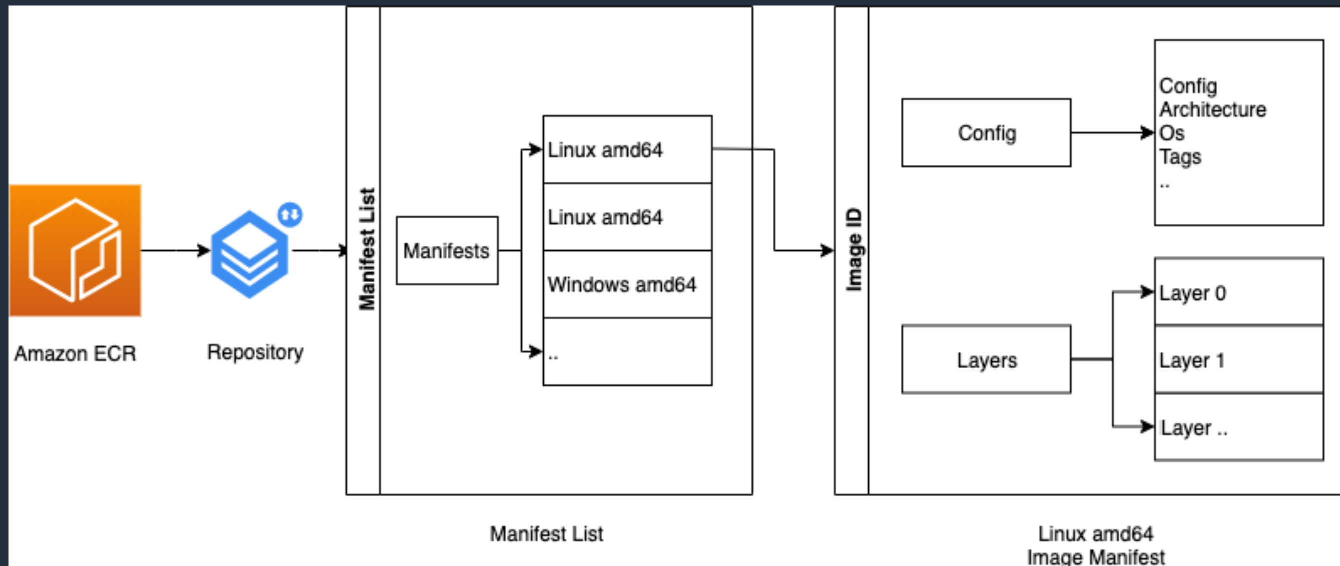
マニフェストリストの mediaType は
oci.image.index

マニフェストには従来通り
アーキテクチャが記載

各マニフェストの mediaType は
oci.image.manifest

© 2023, Amazon Web Services, Inc. or its affiliates.

リポジトリにイメージを保存すると



コンテナイメージを pull するときはまずマニフェストリストをpullしてきて、次にコンテナホストと同じアーキテクチャのマニフェストを pull する。

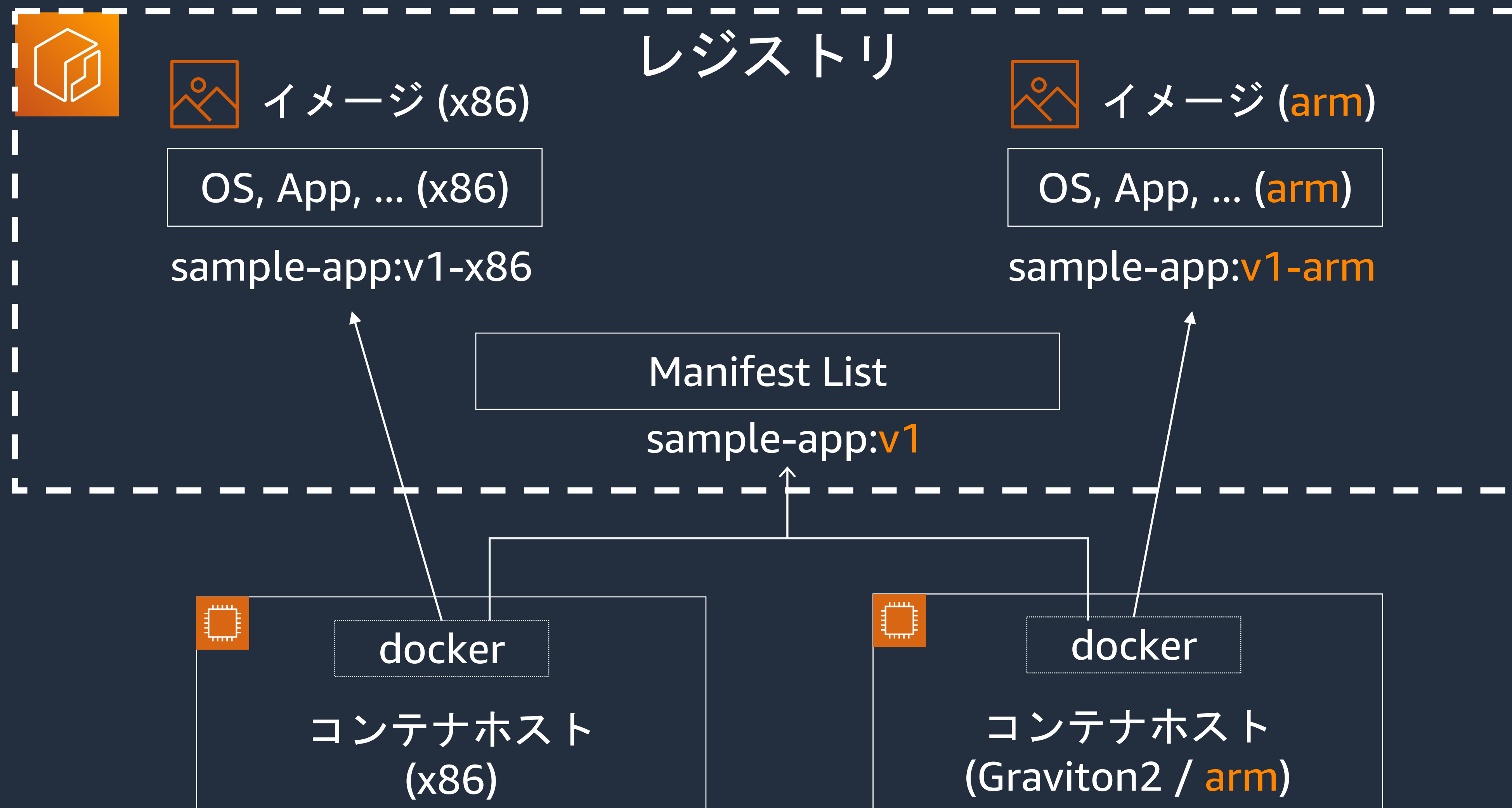
→ホストが amd64 でも arm64 でも最初に pull するのはマニフェストリストで共通

https://aws.amazon.com/jp/blogs/containers/introducing-multi-architecture-container-images-for-amazon-ecr/ より

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

イメージレジストリのマルチアーキテクチャ対応

- 同じタグで、複数のコンテナイメージをサポート



© 2023, Amazon Web Services, Inc. or its affiliates.

コンテナイメージのビルド

© 2023, Amazon Web Services, Inc. or its affiliates.

マルチアーキテクチャコンテナイメージのまとめ

- コンテナのイメージマニフェストを作成すれば特定の CPU アーキテクチャをターゲットとしたコンテナイメージを定義できる。
  これは例えば docker build で行っていたこと

- マルチアーキテクチャのコンテナイメージを作成したい場合はマニフェストリストを作成すればいい(以下は疑似コード)

```
$ docker build image-manifest -t my-app:amd64 .
$ docker build image-manifest -t my-app:arm64 .
# New!
$ docker build manifest-list -t my-app:v1 --target my-app:amd64,my-app:arm64
$ docker push manifest-list my-app:v1
```

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

docker manifest コマンド

- Docker を使っている場合 docker manfiest コマンドでマニフェストリストの作成とそのプッシュを行える。このコマンドは experimental (https://docs.docker.com/engine/reference/commandline/manifest_create)

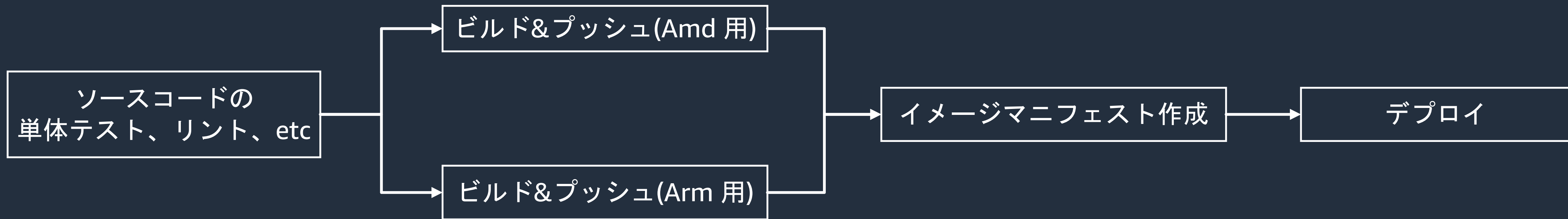
```
$ docker manifest create \
your-username/multiarch-example:manifest-latest \
--amend your-username/multiarch-example:manifest-amd64 \
--amend your-username/multiarch-example:manifest-arm32v7 \
--amend your-username/multiarch-example:manifest-arm64v8
```

```
$ docker manifest push your-username/multiarch-example:manifest-latest
```

https://www.docker.com/blog/multi-arch-build-and-images-the-simple-way/ より

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

CI/CD パイプライン (docker manifest を使う場合)




```
# AMD64
$ docker build -t your-username/multiarch-example:manifest-amd64 --build-arg ARCH=amd64/ .
$ docker push your-username/multiarch-example:manifest-amd64

# ARM32V7
$ docker build -t your-username/multiarch-example:manifest-arm32v7 --build-arg ARCH=arm32v7/ .
$ docker push your-username/multiarch-example:manifest-arm32v7

# ARM64V8
$ docker build -t your-username/multiarch-example:manifest-arm64v8 --build-arg ARCH=arm64v8/ .
$ docker push your-username/multiarch-example:manifest-arm64v8
```

© 2023, Amazon Web Services, Inc. or its affiliates.

docker buildx build コマンド

- Buildx を使っている場合 docker build と同じ引数を渡しつつ、複数のアーキテクチャ向けのイメージビルド・マニフェストリストの作成・マニフェストリストのプッシュを一度に行うことができる: https://docs.docker.com/engine/reference/commandline/buildx_build/

```
$ docker buildx build \
--push \
--platform linux/arm/v7,linux/arm64/v8,linux/amd64 \ --tag your-username/multiarch-example:buildx-latest
.
```

https://www.docker.com/blog/multi-arch-build-and-images-the-simple-way/ より

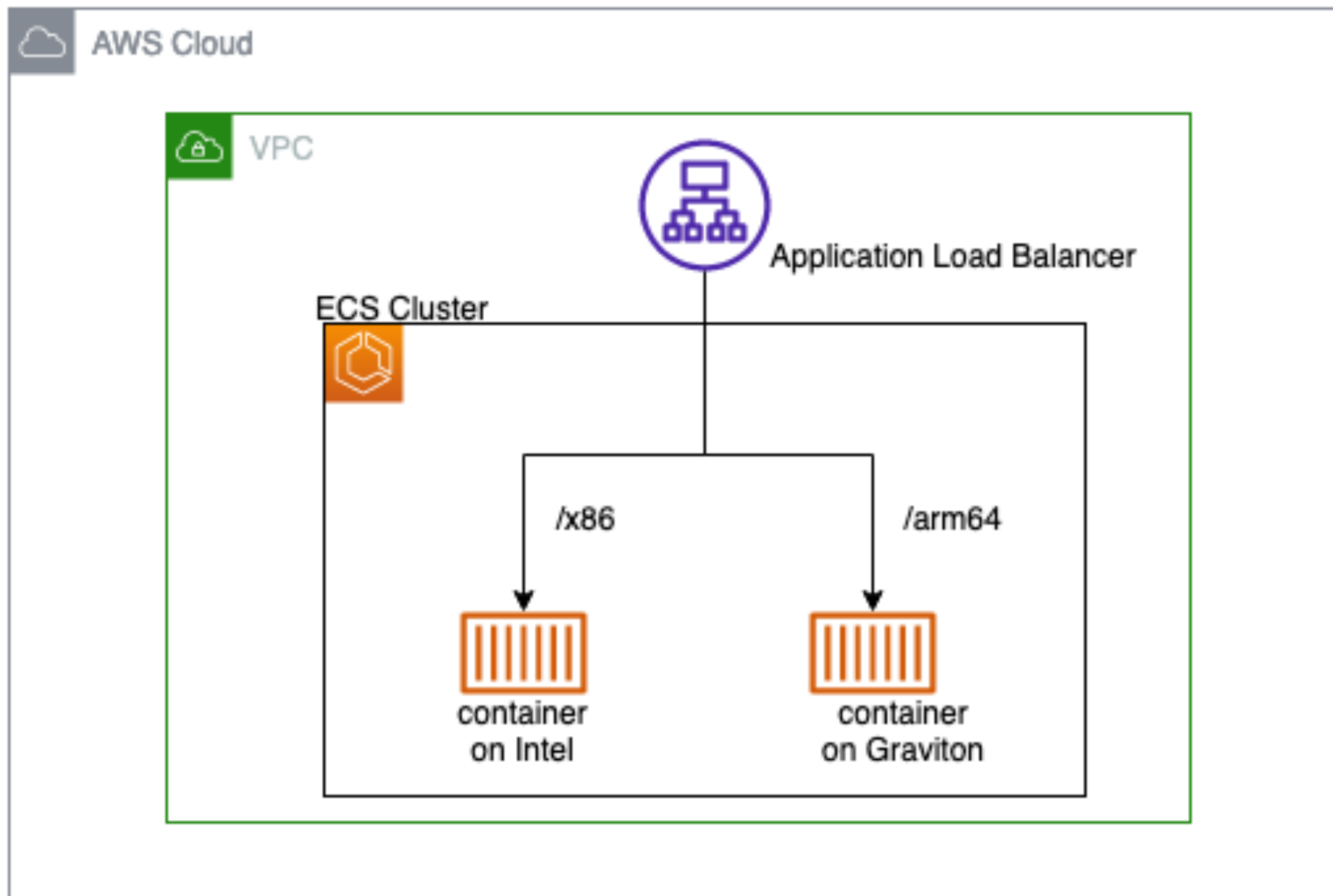
© 2023, Amazon Web Services, Inc. or its affiliates.

CI/CD パイプライン (docker buildx build を使う場合)

© 2023, Amazon Web Services, Inc. or its affiliates.

デモ: ECS Fargate で動かす

© 2023, Amazon Web Services, Inc. or its affiliates.



https://github.com/hkford/ecs-fargate-on-graviton-intel より

© 2023, Amazon Web Services, Inc. or its affiliates.

```
68            const taskDefinition = new ecs.FargateTaskDefinition(
69                this,
70                `${props.identifier}TaskDefinition`,
71                {
72                    memoryLimitMiB: 512,
73                    cpu: 256,
74                    executionRole: props.taskExecutionRole,
75                    runtimePlatform: {
76                        operatingSystemFamily: ecs.OperatingSystemFamily.LINUX
77                        cpuArchitecture: props.architecture
78                    },
79                    taskRole: props.taskRole,
80                }
81            );
82
83            const frontendImage = new ecs.AssetImage('frontend');
84
85            taskDefinition.addContainer(`${props.identifier}Container`, {
86                image: new ecs.EcrImage(props.imageRepository, "v1"),
87                logging: ecs.LogDriver.awsLogs({
88                    streamPrefix: props.identifier,
89                    logGroup: props.logGroup,
90                }),
91                portMappings: [
92                    {
93                        containerPort: 3000,
94                        protocol: ecs.Protocol.TCP,
95                    },
96                ],
97
98            });
```

タスク定義の中でCPU アーキテクチャを指定(Amd か Arm)

コンテナイメージ自体は共通

https://github.com/hkford/ecs-fargate-on-graviton-intel/blob/main/lib/ecs-services.ts#L66 より

aws
© 2023, Amazon Web Services, Inc. or its affiliates.

まとめ

© 2023, Amazon Web Services, Inc. or its affiliates.

まとめ

- マルチアーキテクチャコンテナイメージを使用することで Amd だけでなく Arm (Graviton) でも動くコンテナイメージを作成できる
- コンテナイメージのマニフェストリストによって実現
- イメージのビルドには docker manifest コマンドを使う方法と docker buildx build を使う方法などがある
- ビルドのやり方はわかったから次はデプロイや運用のことを聞きたい
  -> 次の黄さんのセッションへ Go

© 2023, Amazon Web Services, Inc. or its affiliates.