[AWS Black Belt Online Seminar]

Amazon ECS Deep Dive

サービスカットシリーズ

Solutions Architect
濵 真一
2019/07/31

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

濵 真一

Twitter/@track3jyo

- 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

- 好きなサービス
  Amazon ECS　AWS Fargete　AWS CDK

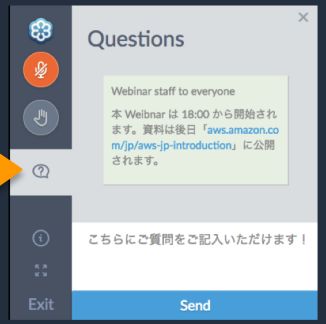
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- いただいたQ&Aをピックアップしてblogにご紹介させていただく場合がございます
- 今後のロードマップに関するご質問はお答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2019年7月31日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様が東京リージョンを使用する場合、別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

前提事項

- **前提:**
  - コンテナ、及びDockerについてご利用経験があること
  - Amazon ECS の基本的な内容を習得されていること
  - Linuxコンテナの利用に関するユースケース
-
- **対象者:**
  - Amazon ECSを使ったアプリケーションをすでに運用されている方
  - Amazon ECSを利用したワークロードの開発を進めようとされている方

本日の内容

- Amazon ECS 概要
- Amazon ECS お悩み相談室
  - Amazon ECSをご利用されているお客様からよく伺うお悩みを紹介
  - 最新のアップデートと交えて、お客様の課題や悩みがどう解決されるのかを解説
- 補足・まとめ

Amazon ECS 概要

AWSのコンテナ関連サービス

イメージレジストリ
コンテナイメージのレポジトリ

Amazon Elastic Container Registry（Amazon ECR）

オーケストレーション
デプロイ、スケジューリング、スケーリング、クラスタ管理

Amazon Elastic Container Service（Amazon ECS）

Amazon Elastic Kubernetes Service（Amazon EKS）

実行環境
コンテナ実行

AWS Fargate

Amazon Elastic Compute Cloud（Amazon EC2）

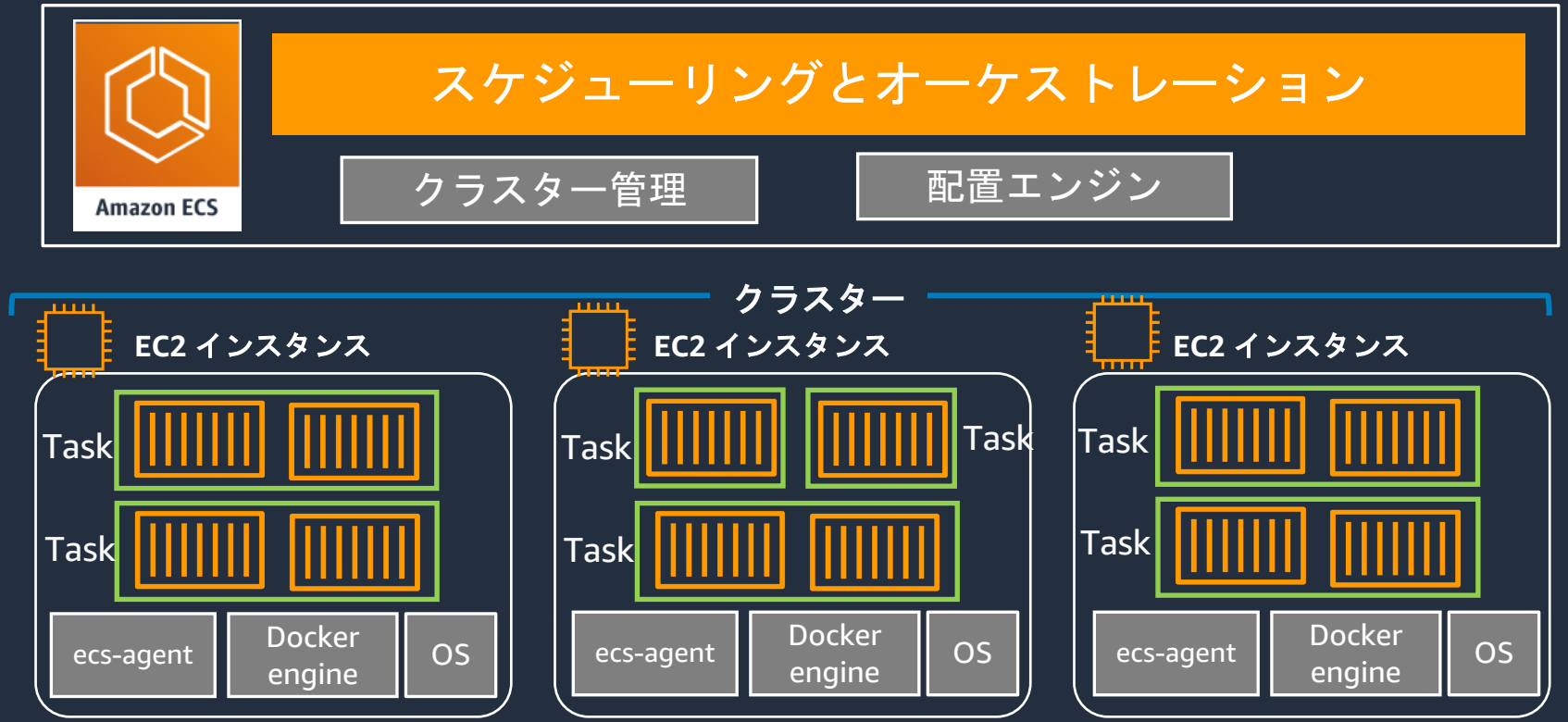
Amazon ECSとは？

■コントロールプレーンとしての基盤

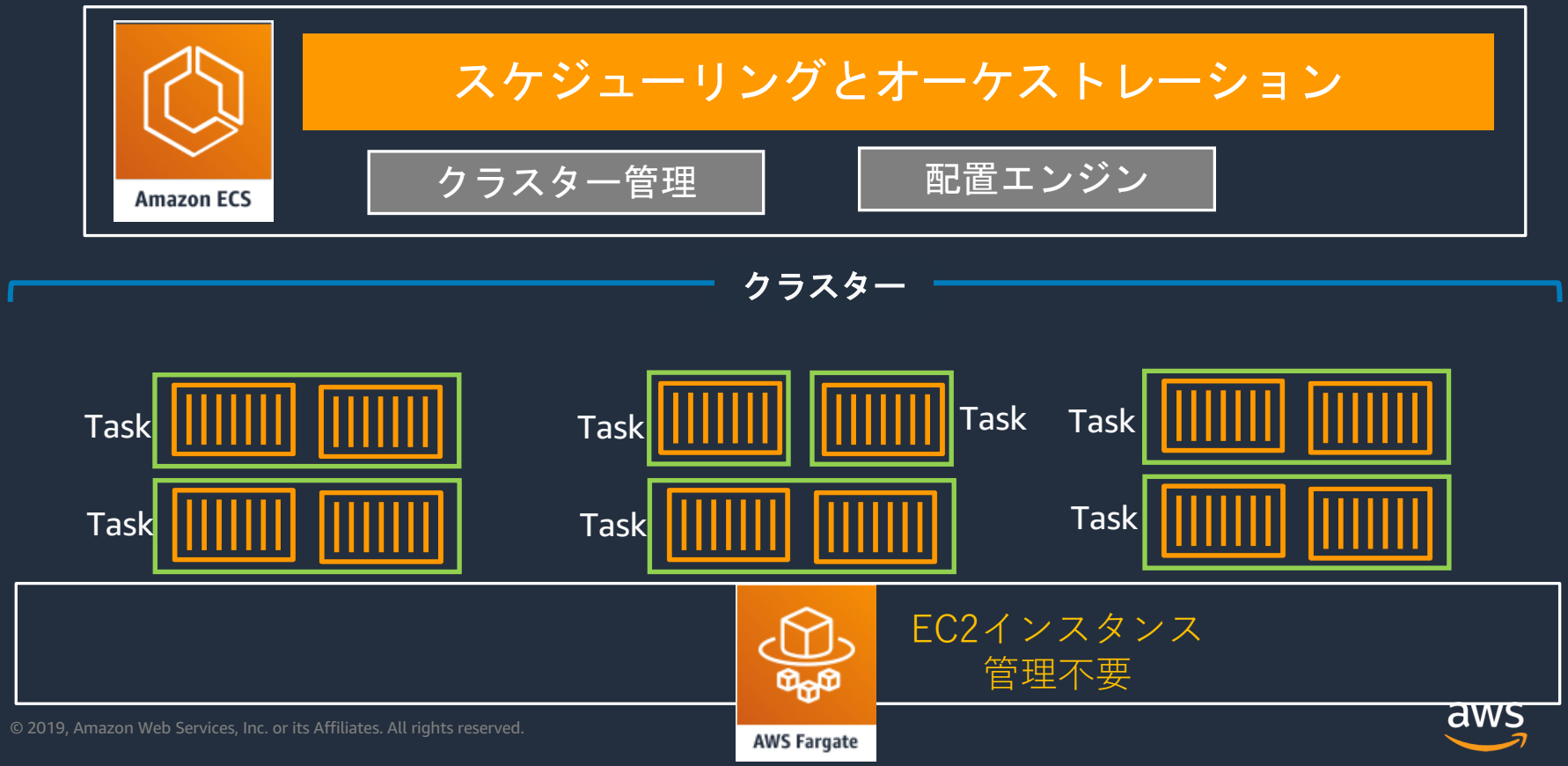
Linux & Windows

- コンテナレベルのネットワーク構成
- 高度なタスク配置戦略
- 他のAWSサービスとの連携
- ECS CLI
- グローバル展開
- 強力なスケジュールエンジン
- オートスケーリング
- CloudWatch 連携（ログ/メトリクス/イベント）
- ロードバランサー

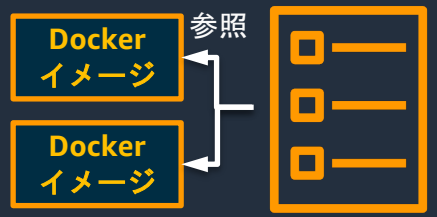
Amazon ECSの動作イメージ（on EC2）



Amazon ECSの動作イメージ（on Fargate）

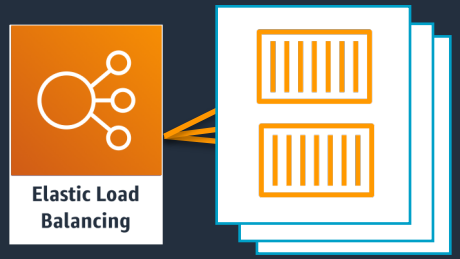


Amazon ECSの主要要素



Task Definition

参照



- Task Definitionに基づき起動されるコンテナ群
- Task内コンテナは同一ホスト上で実行される

Task

Service

- Task実行コピー数(n個)を定義
- 起動後、Task実行コピー数を維持
- ELBと連携
- 起動タイプ(EC2, Fargate)を設定

Cluster

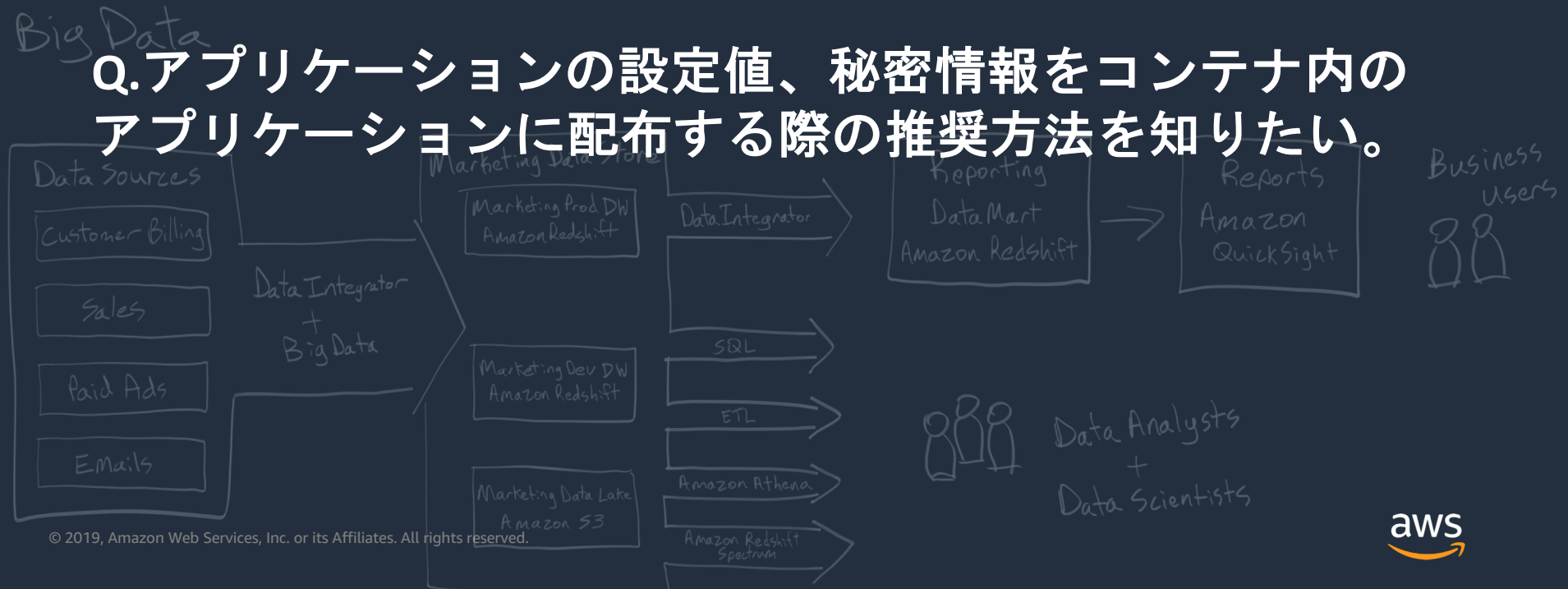
- 実行環境の境界
- IAM権限の境界(クラスターに対する操作)
- スケジュールされたタスクの実行を設定可能

Taskを構成するコンテナ群定義：

- コンテナ定義(イメージ場所等)
- 要求CPU & メモリ
- Taskに割当てるIAMロール
- ネットワークモード
- etc...

Amazon ECSお悩み相談室

Q.アプリケーションの設定値、秘密情報をコンテナ内のアプリケーションに配布する際の推奨方法を知りたい。



このお悩みの背景

- 環境ごとにイメージを作っており、デプロイ漏れ/環境差異といった課題
    - 設定を定数としてコード内に格納

- 秘密情報は暗号化して安全に保つために、 AWS Secrets Managerで管理し、ENTRYPOINTの初期化プロセスで、値を取得、復号化
    - RDSのパスワード
    - 外部APIの認証キー

現状のアプリケーション運用にかかる負担を軽減したい

Answer：

- 設定値については、ECSの環境変数を利用してコンテナ実行時に値を挿入する

- 秘密情報については、 AWS Secrets Managerを利用し、よりセキュアに秘密情報を管理した上で、ECSの環境変数との統合を利用し透過的に秘密情報をコンテナに配布する

設定は環境変数に格納する

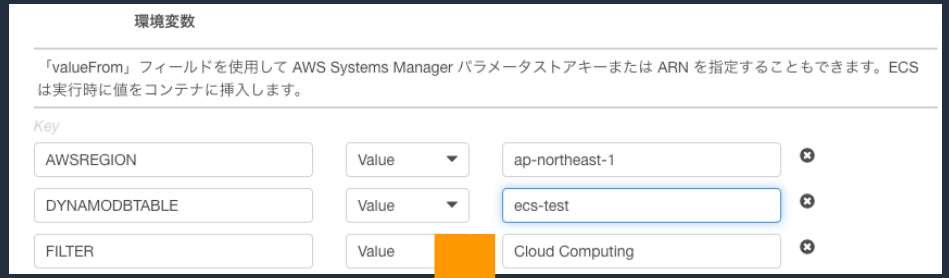
- コードを変更することなくデプロイごとに簡単に変更可能
    - 環境別にアプリケーションの挙動を切り替えることに利用

- 独自形式の設定ファイルやJava System Propertiesなど他の設定の仕組みとは異なり、言語やOSに依存しない標準

Twelve-Factor App 設定
https://12factor.net/ja/config

タスク定義の環境変数(environment)を活用する

- ECSのenvironmentにValueとしてタスク定義内で設定
  - データベースのスキーマ
  - S3のバケット名
  - 真偽値
  - 他のServiceのDNS名
- 実行時にValueがコンテナ内に挿入される




```
from __future__ import absolute_import, print_function

from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import json
import boto3
import os

# DynamoDB table name and Region
dynamoDBTable=os.environ['DYNAMODBTABLE']
region_name=os.environ['AWSREGION']

# Filter variable (the word for which to filter in your stream)
filter=os.environ['FILTER']

```

秘密情報はAWS Secrets Manager との統合機能を使う *NEW!!*

- AWS Secrets Managerとの統合機能を使って、透過的にパスワードといった秘密情報をコンテナに配布

- ECSのenvironmentの「valueFrom」フィールドを使用して、 SecretsのARNを記載

シークレット

新しいシークレットを保存する

シークレットの名前で検索

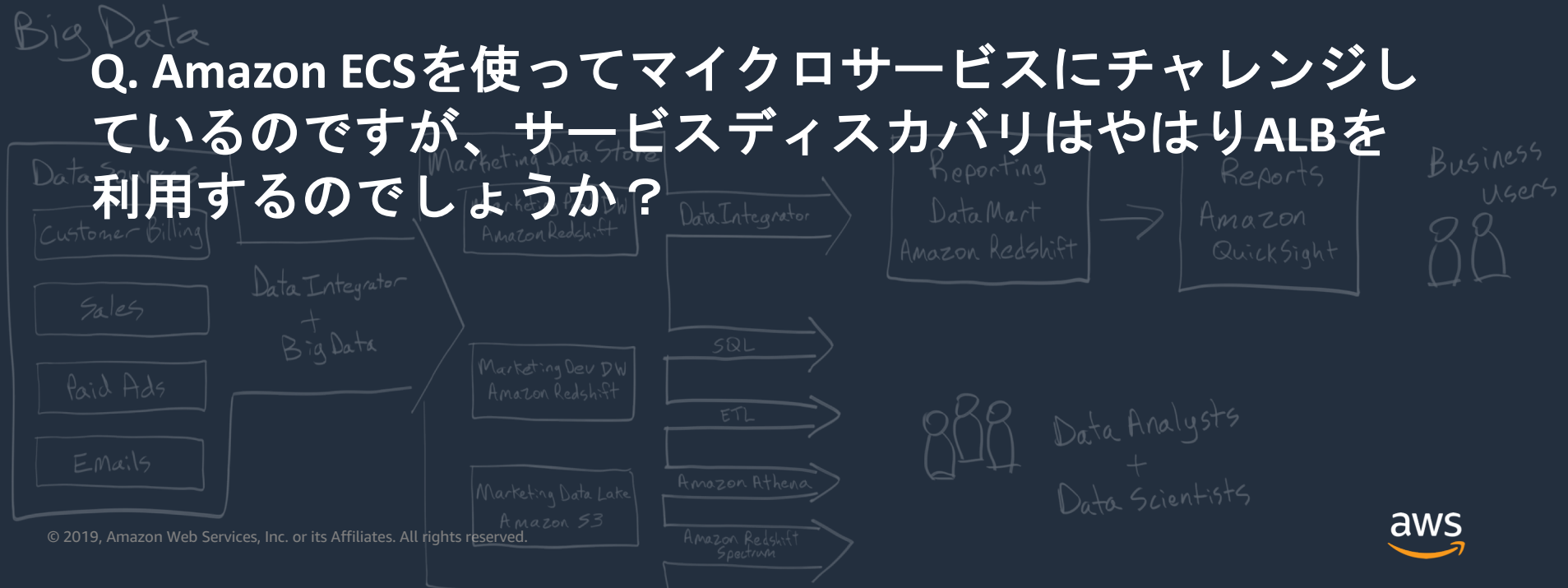
シークレットの名前	説明	最後に取得した (UTC)
CONSUMERKEY1	Twitter API Consumer Key	07/24/2019
CONSUMERSECRETKEY1	Twitter API Consumer Secret Key	07/24/2019
ACCESSTOKEN1	Twitter API Access Token	07/24/2019
ACCESSTOKENSECRET1	Twitter API Access Token Secret	07/24/2019

ACCESSTOKEN1	ValueFrom	arn:aws:secretsmanager:ap-northeast-1:
ACCESSTOKENSECRET1	ValueFrom	arn:aws:secretsmanager:ap-northeast-1:
CONSUMERKEY1	ValueFrom	arn:aws:secretsmanager:ap-northeast-1:
CONSUMERSECRETKEY1	ValueFrom	arn:aws:secretsmanager:ap-northeast-1:

AWS Secrets Manager との統合機能を使って変わること

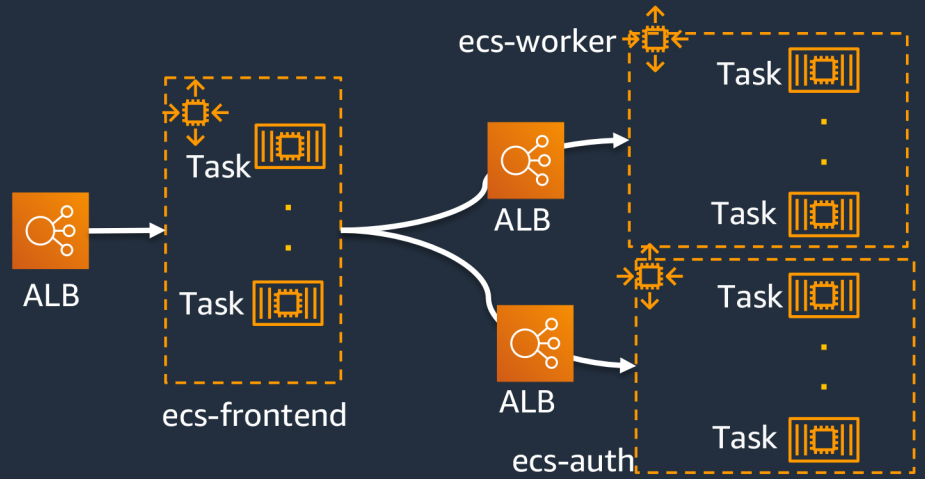
- コンテナ起動時間の短縮
- AWSマネージドで扱えるようになったことで、アプリケーションは秘密情報の管理や暗号化/復号化処理について意識しなくて良い

Q. Amazon ECSを使ってマイクロサービスにチャレンジしているのですが、サービスディスカバリはやはりALBを利用するのでしょうか？



このお悩みの背景

- ECSコンテナ環境上で、マイクロサービスなアプリケーションを展開
- 各サービス間の連携でInternal ALBを利用
- ALBは複数タスクへの負荷分散やヘルスチェック目的のみで利用



もっとシンプルな形でもいいので、ランニングコストを下げたい

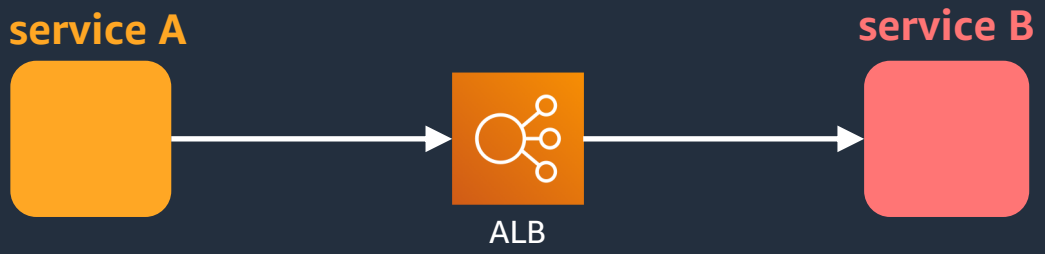
Answer：

- ECSはロードバランサベースとDNS ベースの2種類のサービスディスカバリをネイティブに利用可能
- それぞれのメリットがあるので、サービス間連携で求める要件に応じて、適切なService Discoveryを選定する
- 負荷分散/ヘルスチェック目的だけなのであれば、低コストでシンプルな設計になるECS Service Discovery（ DNS ベース）

Amazon ECSのService Discovery

ECS では ロードバランサベースとDNS ベースの
2種類のサービスディスカバリをネイティブに利用可能

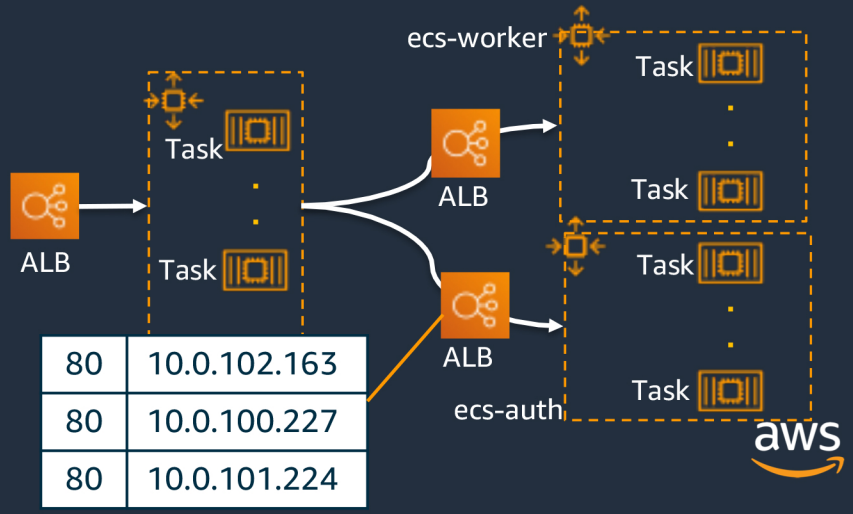
LB-based



DNS-based

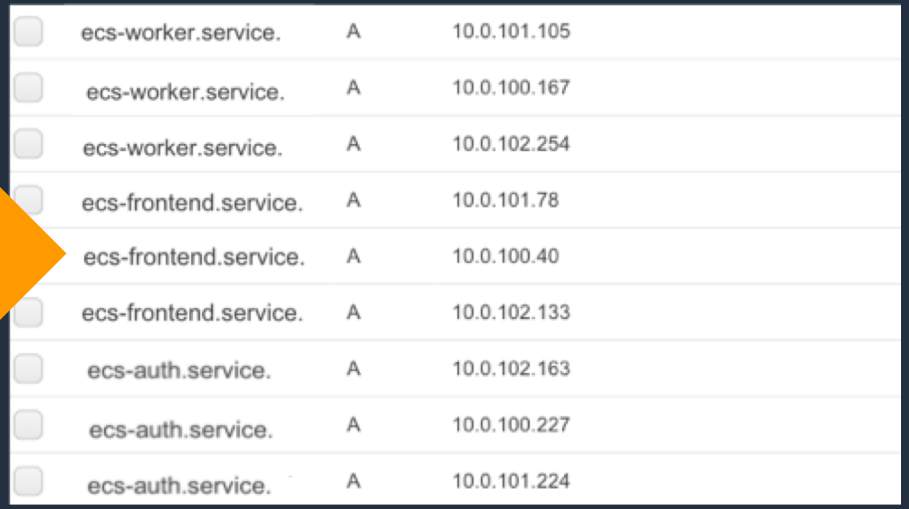
ELBを使ったService Discovery

- ロードバランサへの登録・登録解除
  - ECS サービスがタスクを起動した上で、ターゲットグループに登録
  - 停止時は登録を解除
- 呼び出し元のサービスは、ロードバランサの ドメイン名で宛先を解決
- ロードバランサの機能を利用可能
  - Connection Draining
  - L4/L7
  - ELB healthcheck



ECS Service Discovery (DNSベース)

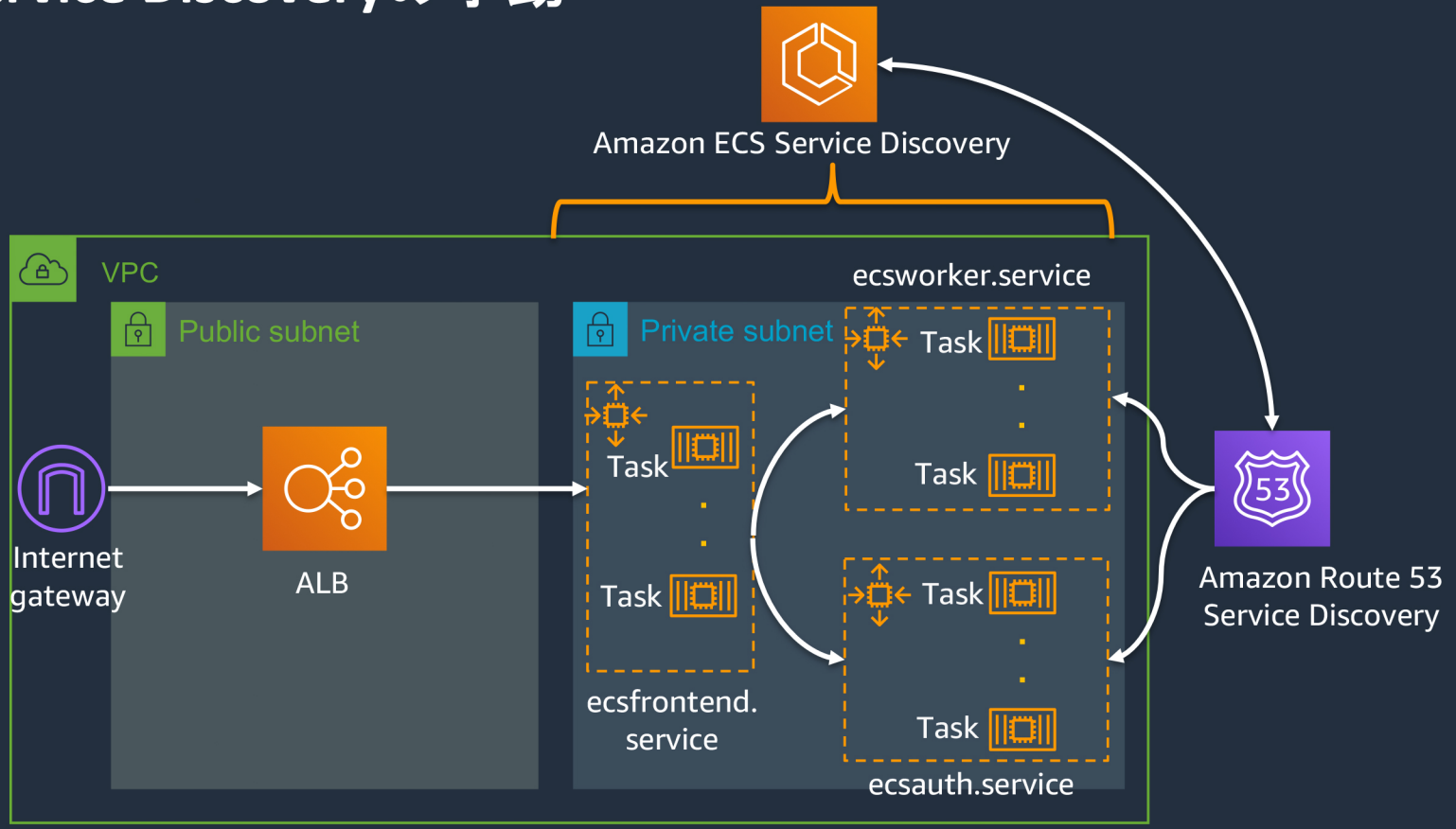
- ECS サービスが Amazon Route 53 のドメイン 名で自動登録/管理



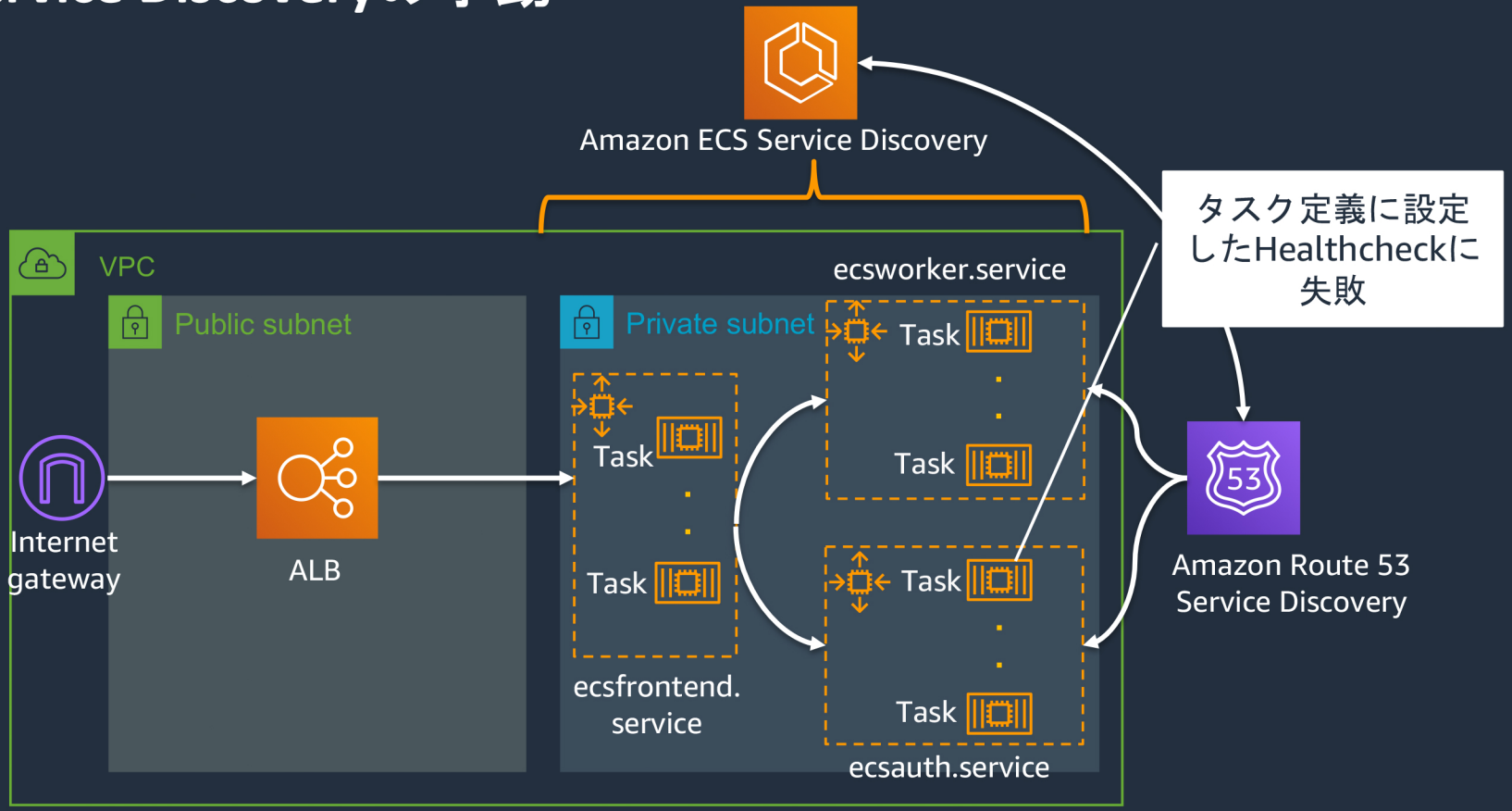
ECS Service Discovery (DNSベース)

- DNS レコードの追加・削除
  - ECS サービスがタスクを起動した上で、レコードを追加
  - 停止時はレコードを削除
- 呼び出し元のサービスは、service-name.local のような相手方の名前を使ってタスクの宛先を解決できる
- タスク定義のコンテナレベルでのヘルスチェックのみ有効

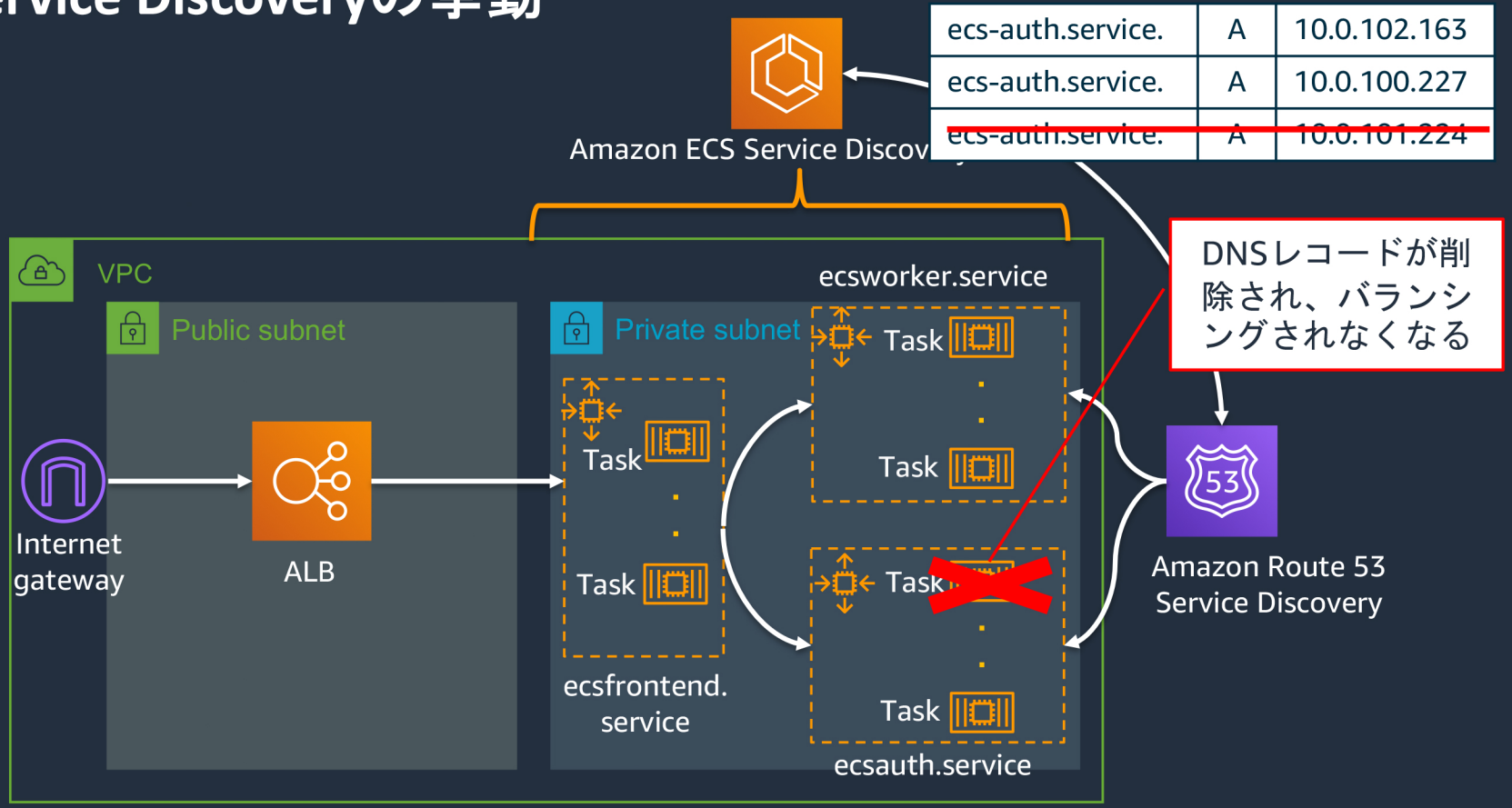
ECS Service Discoveryの挙動



ECS Service Discoveryの挙動



ECS Service Discoveryの挙動



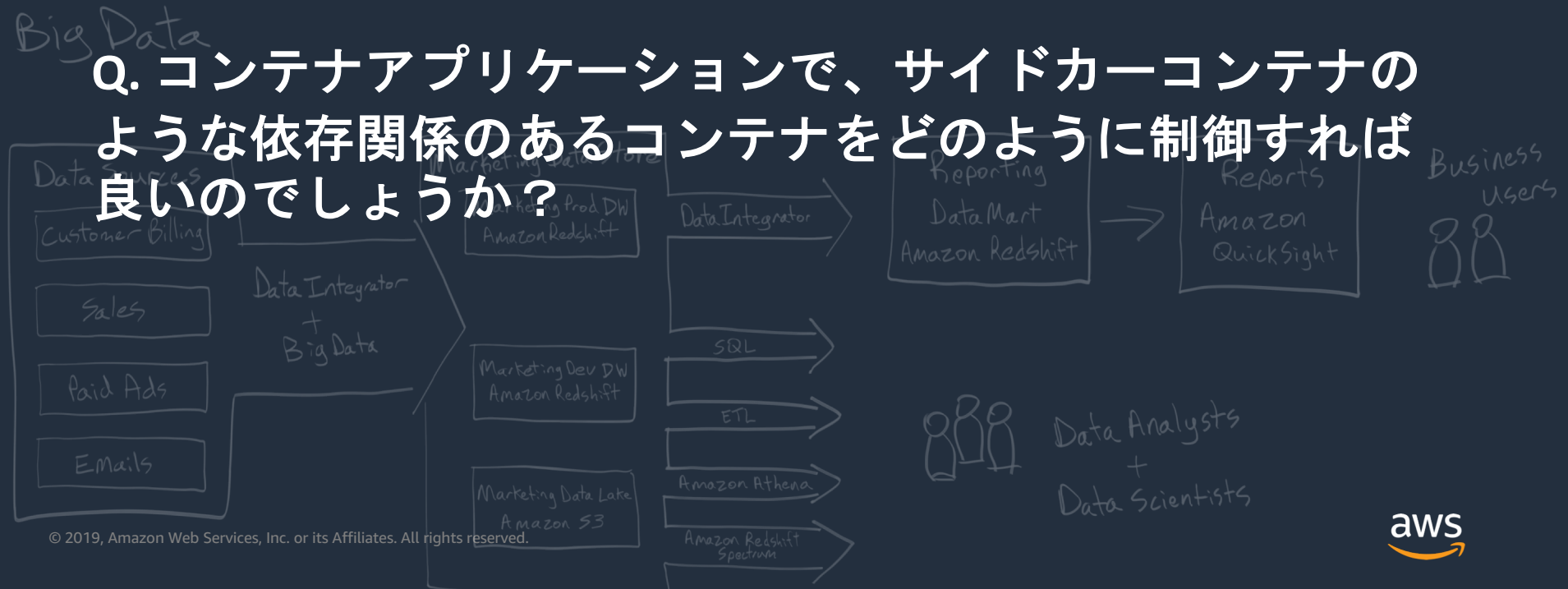
各Service Discoveryの特徴

- ELBを使ったService Discoveryのメリット
  - ロードバランサの機能を使ったルーティング制御
  - インターネットアクセスを利用可能

- ECS Service Discoveryのメリット
  - LBを使わない分低コスト
  - セキュリティグループ設計など設計がシンプルになる

サービス間連携で求める仕様に合わせて、適切なService Discoveryを選定する

Q. コンテナアプリケーションで、サイドカーコンテナのような依存関係のあるコンテナをどのように制御すれば良いのでしょうか？



このお悩みの背景

- サイドカーコンテナに依存するメインコンテナでWait処理を入れている
  - コンテナの起動順を制御したい

- コンテナアプリの初期化処理をENTRYPOINTの起動プロセスに自前で実装
  - アプリとして不要な実装を別のコンテナに分離したい

Answer：

- 依存関係のあるコンテナには、dependsOnパラメータを利用して、タスク内コンテナの起動順、停止順を制御する

- 依存関係のあるコンテナに対してタイムアウトの制御を行いたい際は、startTimeout、stopTimeoutパラメータを利用して制御する

dependsOnパラメータを利用する*NEW!!*

- タスク内のコンテナ間に依存関係を制御
- 各コンテナの起動順や停止順を明示的に設定可能
  - START
  - COMPLETE
  - SUCCESS
  - HEALTHY

```
{
  "name": "side-car",
  "image": "side-car:latest",
    ・・・
  "healthCheck": {
      "command": [
          "CMD-SHELL", "ヘルスチェックコマンド"
      ],
    ・・・
  }
},
{
  "name": "app",
  "image": "app:latest",
    ・・・
  "essential": true,
  "dependsOn": [
      {
          "containerName": "side-car",
          "condition": "HEALTHY"
      }
  ]
}
```

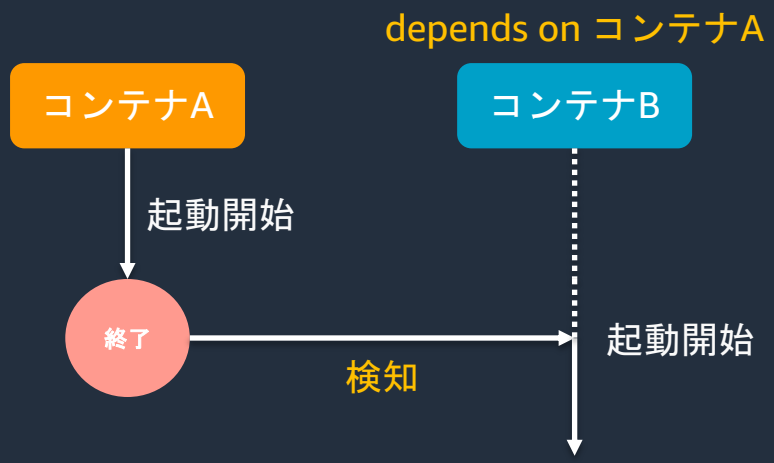
dependsOn condition詳細

- START

依存するコンテナの起動開始後実行

- COMPLETE

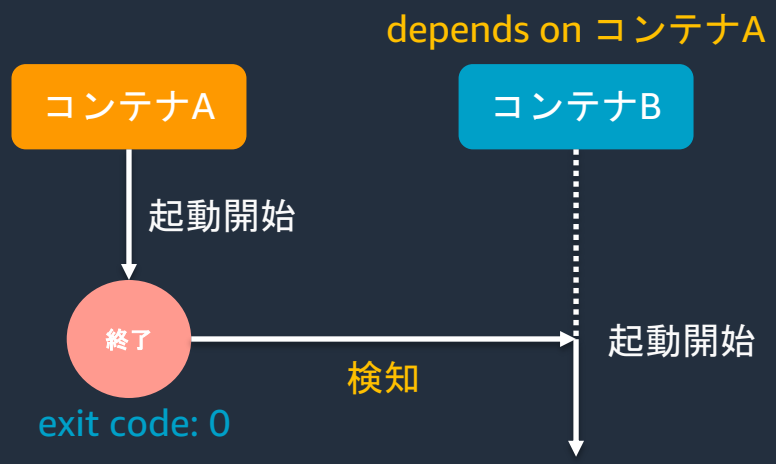
依存コンテナの実行が完了 (終了) 後実行



dependsOn condition詳細

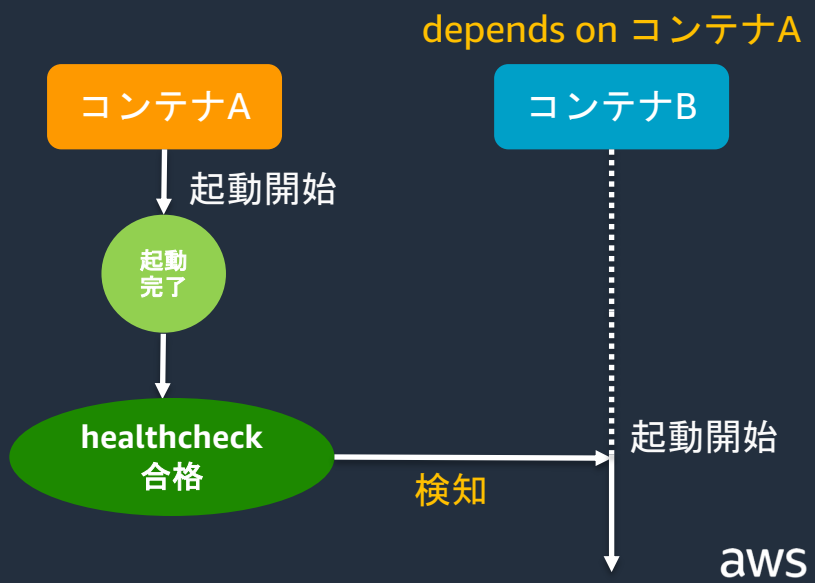
- SUCCESS

依存コンテナの実行が正常終了後に実行（ exit code: 0 ）



- HEALTHY

依存コンテナに定義したhealthcheckに合格後、実行



依存コンテナを扱う上での考慮点

コンテナの起動順が制御できたことで、今度は依存のあるコンテナのタイムアウト制御が行いたいケース

- タスク起動時に依存関係にあるコンテナがいつまでも起動してこない
  - Task Statusも変化せず、処理が進まない

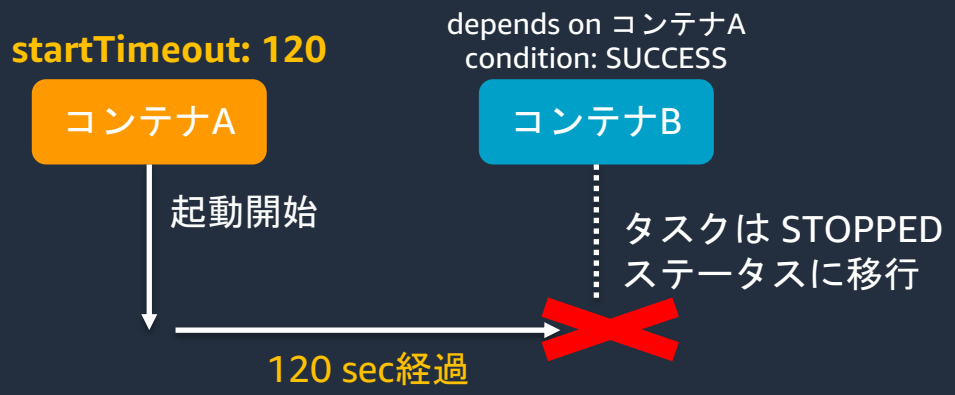
- タスク停止時のSIGTERM送信からの依存関係のあるコンテナの終了処理が終わらず、30secのタイムアウトののちに SIGKILL が送信され強制終了される
  - **処理中のデータをロストするような不具合**が発生し得る

startTimeout, stopTimeoutパラメータを利用する *NEW!!*

- コンテナの正常起動、正常終了を待つ時間をコンテナ単位で設定可能
  - 依存関係のあるコンテナでより柔軟に起動・終了を定義可能

startTimeout

コンテナの依存関係解決の再試行を止めるまでの待機時間



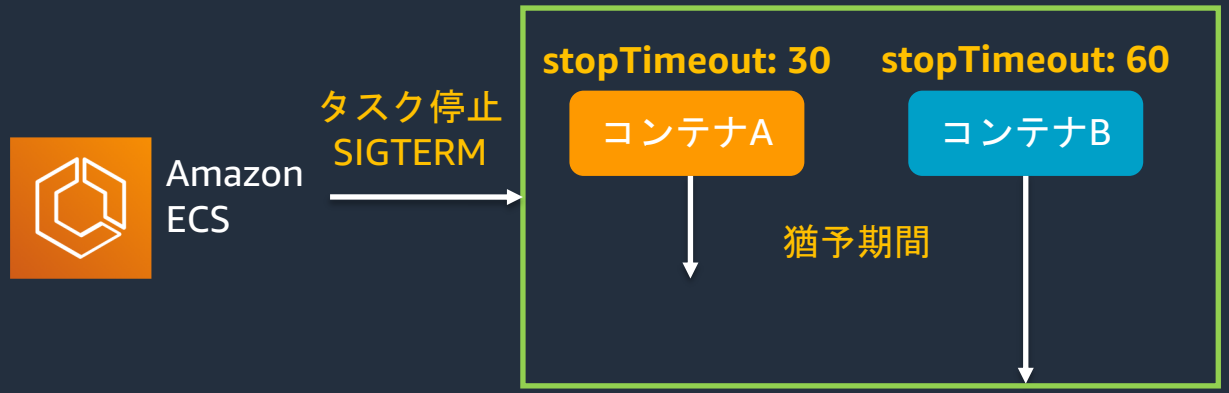
startTimeout, stopTimeoutパラメータを利用する *NEW!!*

- コンテナの正常起動、正常終了を待つ時間をコンテナ単位で設定可能
  - 依存関係のあるコンテナでより柔軟に起動・終了を定義可能

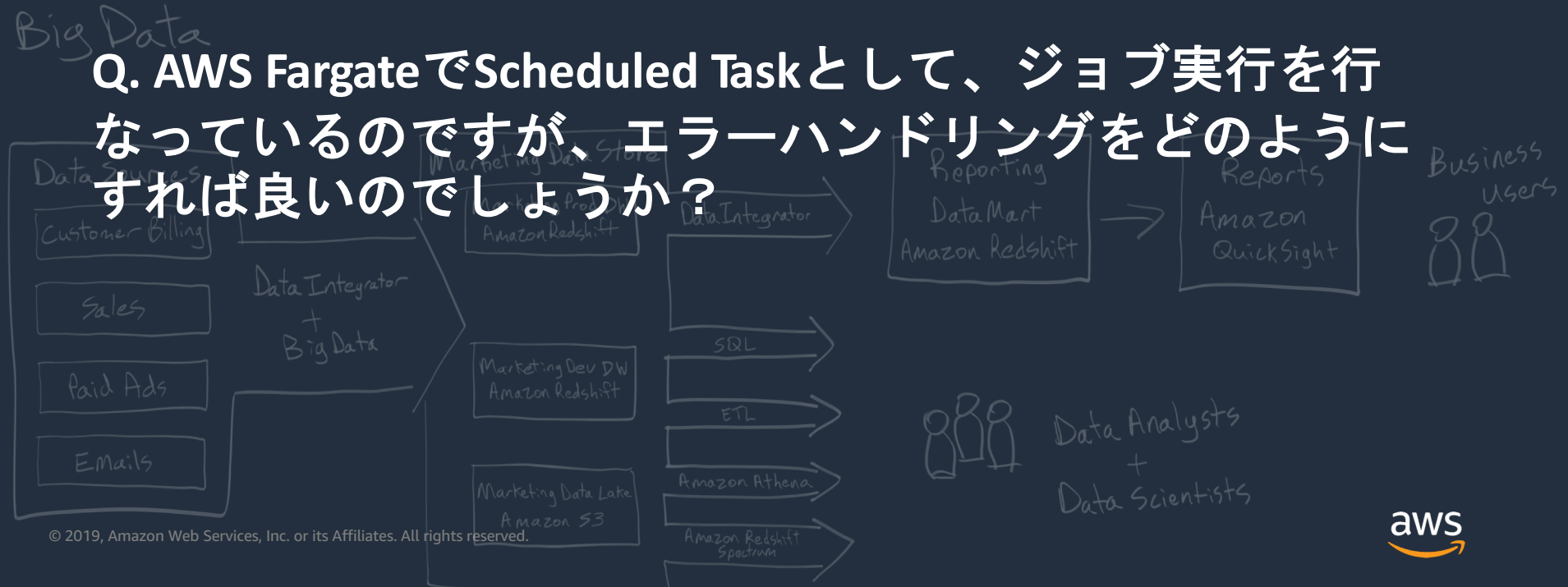
stopTimeout

コンテナが正常に終了しなかった場合に強制終了されるまでの待機時間

SIGKILLによる強制終了のあるコンテナには長めに定義するなどが可能に

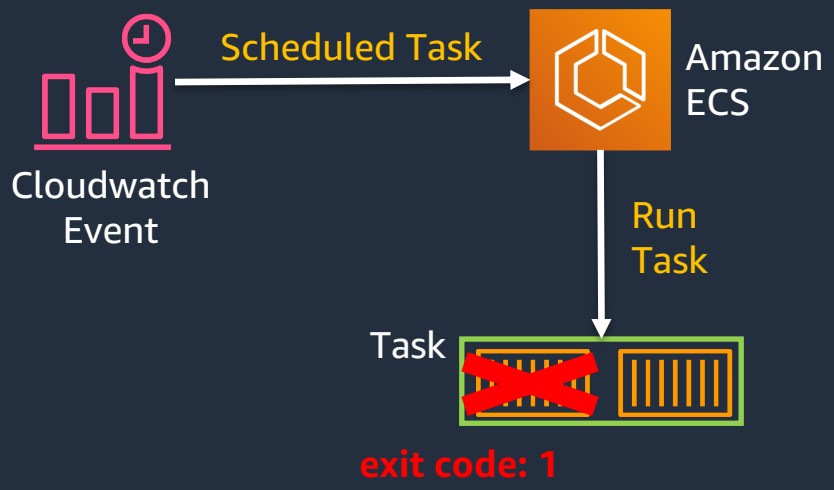


Q. AWS FargateでScheduled Taskとして、ジョブ実行を行なっているのですが、エラーハンドリングをどのようにすれば良いのでしょうか？



このお悩みの背景

- ECSのTaskを1日に1回スケジューリング実行
- Run TaskでのTask実行失敗時にエラーハンドリングはされない
- バッチジョブなので、自分たちでコンテナインスタンスを管理しなくて良いFargateを使っている



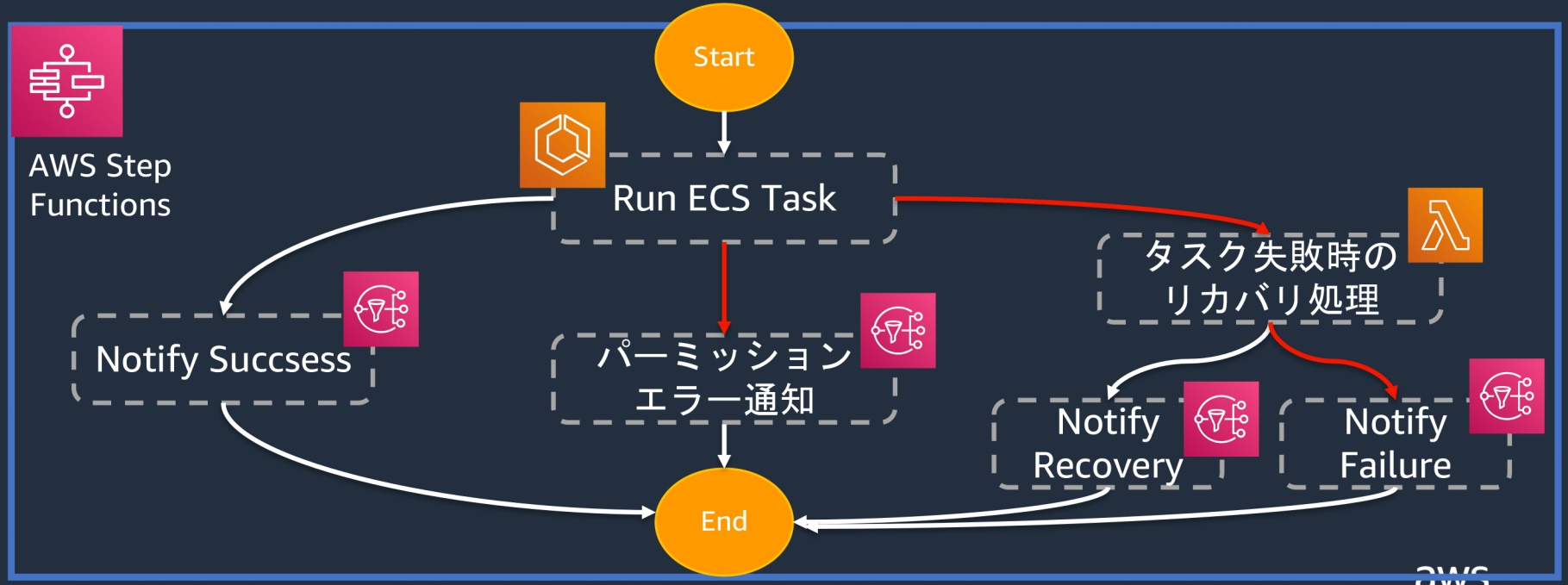
ECS Taskのジョブが失敗した時のエラーハンドリングを行いたい

Answer：

- Run Taskで実行されたTaskのエラーハンドリングは、実際にどのような制御をしたいかに応じて、他のAWSのサービスを活用
- ジョブで他のサービスとの連携など、色々なフロー管理がしたいのであれば、AWS StepFunctionsを利用する
- 単純にエラー検知をしたいのであれば、CloudWatch Events でコンテナの状態変更イベントを取得する方法も有効

Step Functions で Amazon ECSジョブタスクを管理する

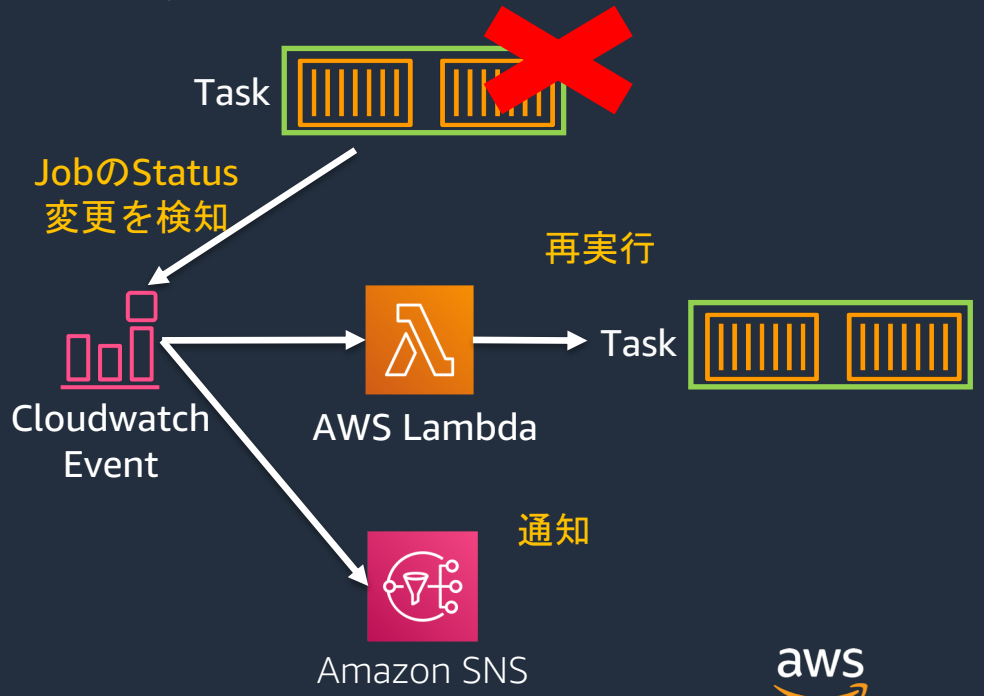
- Step FunctionsでECSタスクの状態に応じたエラーハンドリングや、再実行、通知処理などのフローを定義する



CloudWatch イベント の Amazon ECS イベントストリーム

- Faragate上のタスクの現在の状態に関するほぼリアルタイムの通知
- ECS Task State Changeのイベントタイプで、ジョブのStatusがSTOPPEDの際イベント検知

```
{
  "source": [
    "aws.ecs"
  ],
  "detail-type": [
    "ECS Task State Change"
  ],
  "detail": {
    "clusterArn": [
      "arn:aws:ecs:ap-northeast-1:xxxxxxxxxxxx:cluster/fargate-demo"
    ],
    "lastStatus": [
      "STOPPED"
    ],
    "stoppedReason" : [
      “Essential container in task exited（例）"
    ]
  }
}
```



Q. コンテナアプリケーションのデプロイで、ローリングアップデートやBlue/Greenデプロイ以外で自分たちでカスタマイズしたデプロイやロールバックを実現する方法はありますか？

このお悩みの背景

- ECSを使ったマイクロサービスアプリケーションを展開しており、1日に数回のリリースを実施している

- 用意されているDeploymentではなく、自動化されたデプロイを自分たちで構築して展開したい
    - リリースをより柔軟にコントロールしたい

Answer：

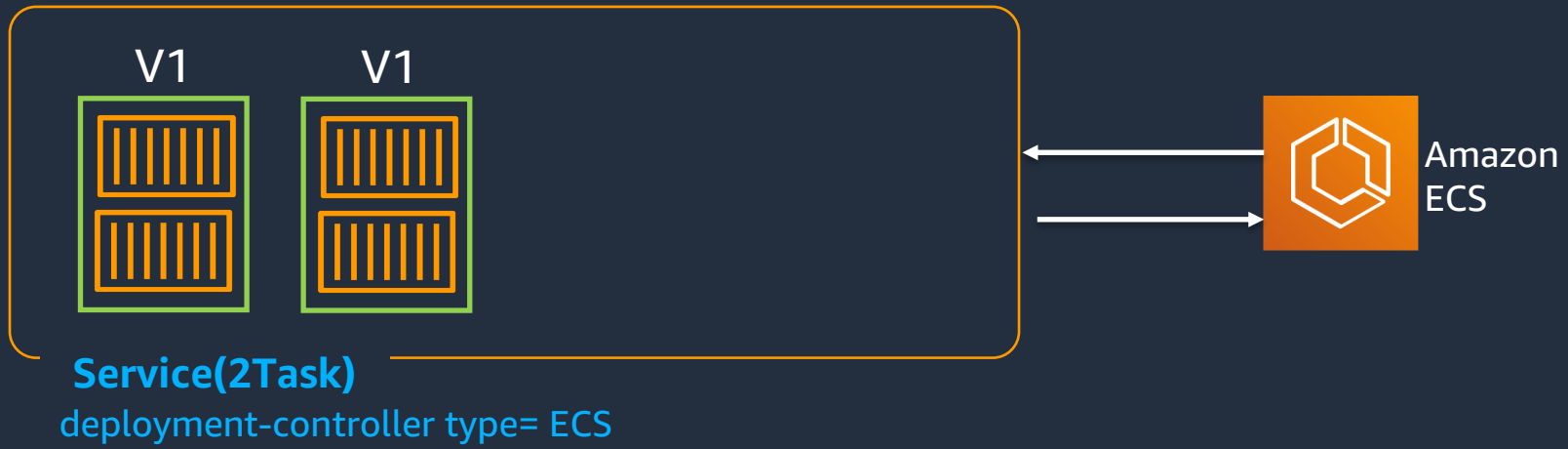
- 自分たちでカスタマイズしたデプロイやロールバックを行いたい場合は、External Deployment Controllerを利用してAPIを使って外部からデプロイをコントロールする

Amazon ECSのDeployment Controller

ローリングアップデート

- Amazon ECS でコントロール
- タスクを指定した数ずつ順番に更新していく

desiredCount: 2、min: 50%、max: 100%

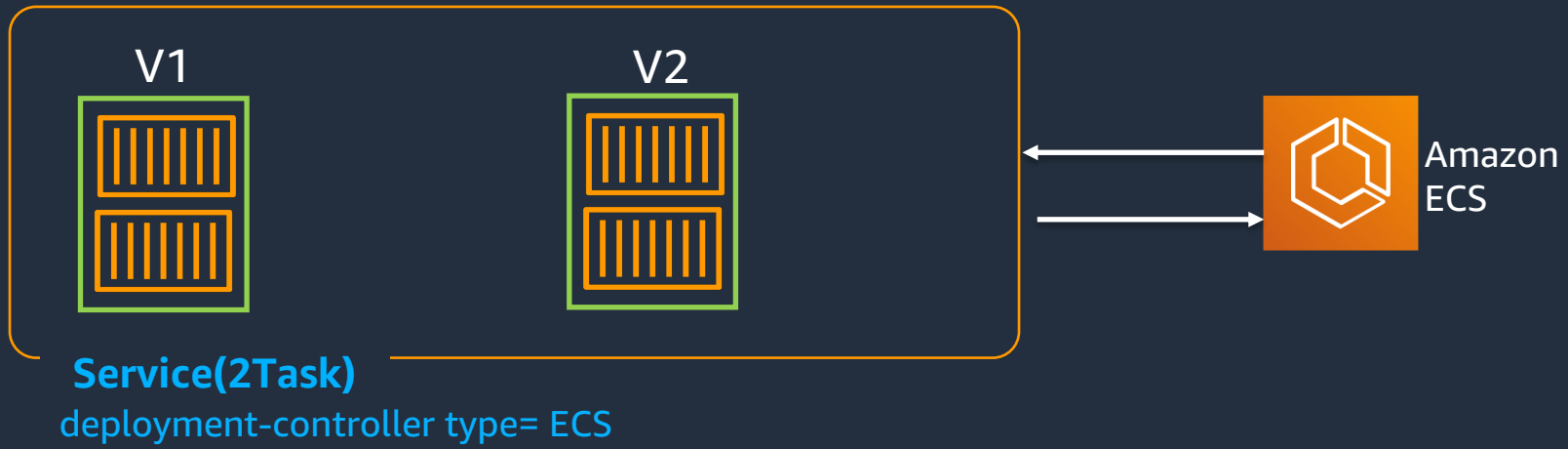


Amazon ECSのDeployment Controller

ローリングアップデート

- Amazon ECS でコントロール
- タスクを指定した数ずつ順番に更新していく

desiredCount: 2、min: 50%、max: 100%

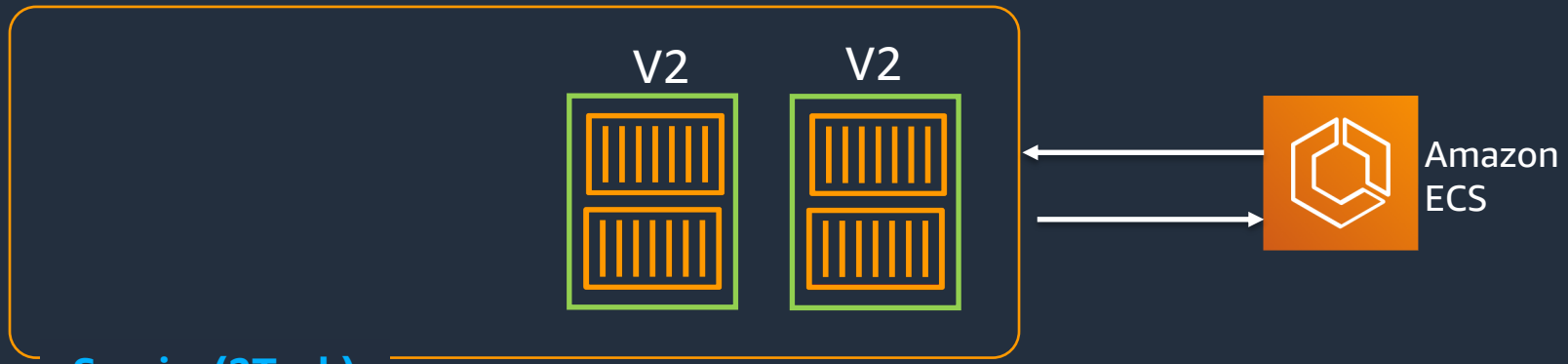


Amazon ECSのDeployment Controller

ローリングアップデート

- Amazon ECS でコントロール
- タスクを指定した数ずつ順番に更新していく

desiredCount: 2、min: 50%、max: 100%

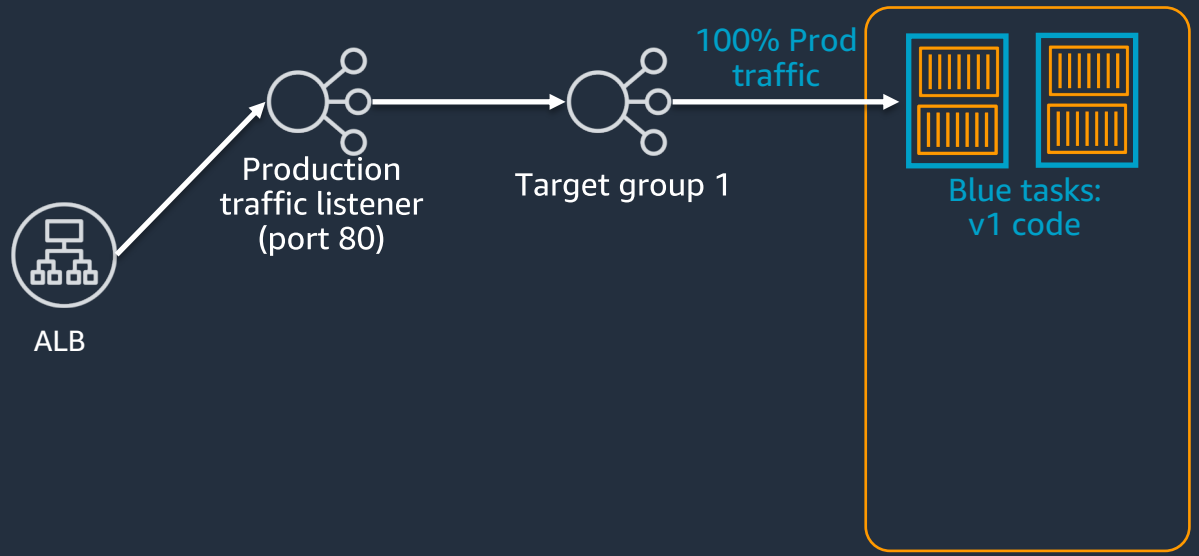


Service(2Task)
deployment-controller type= ECS

Amazon ECSのDeployment Controller

Blue/Green デプロイメント

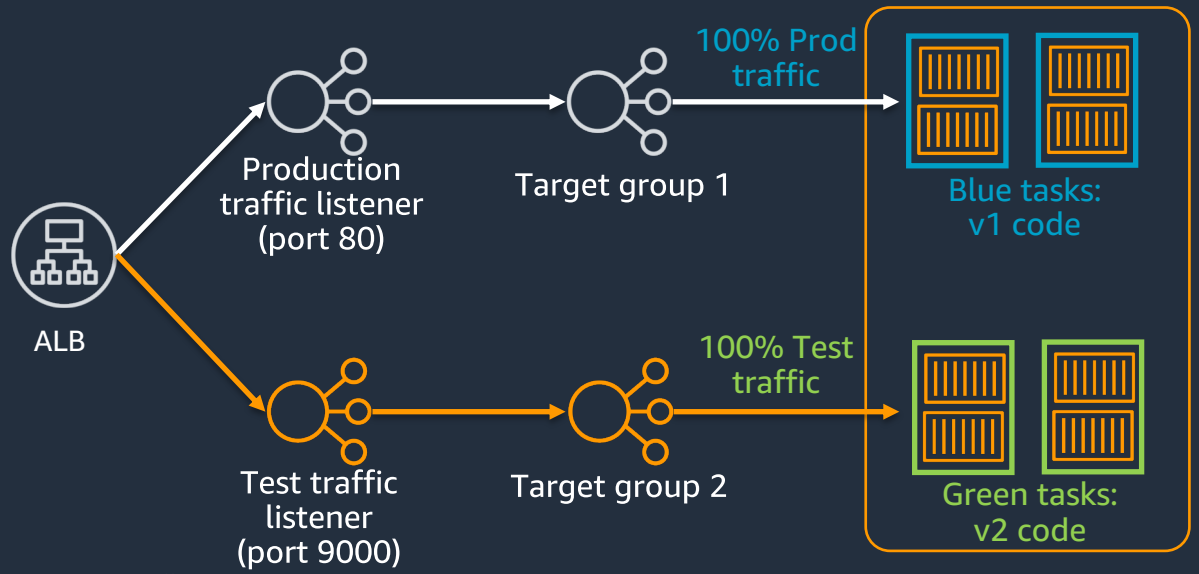
- AWS CodeDeploy,ALBでコントロール
- Green タスク群に本番トラフィックを流す前にテストを実施



Amazon ECSのDeployment Controller

Blue/Green デプロイメント

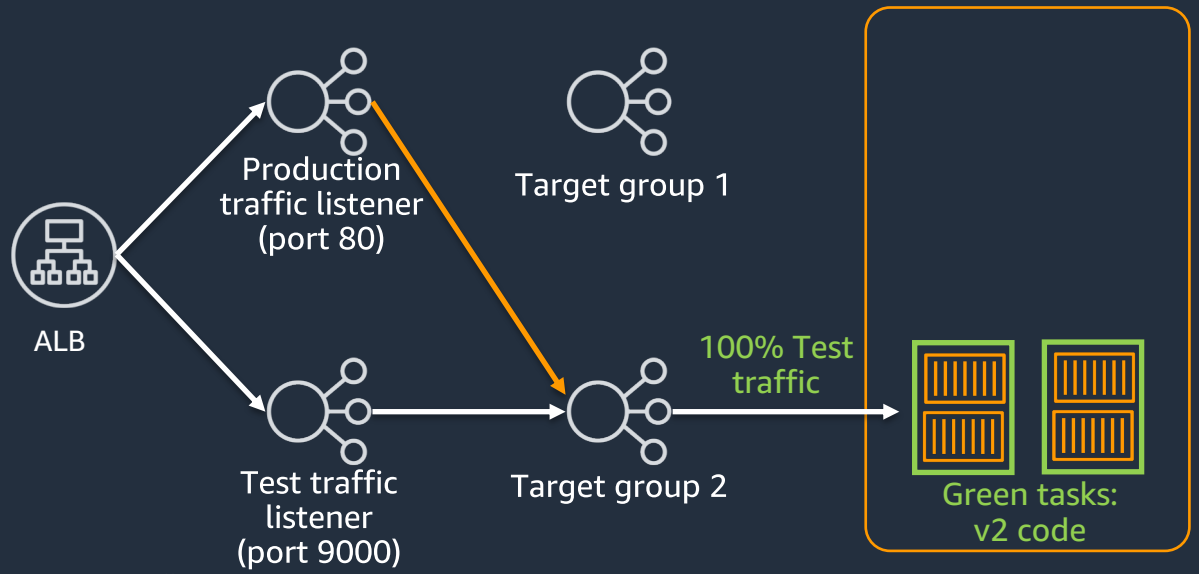
- AWS CodeDeploy,ALBでコントロール
- Green タスク群に本番トラフィックを流す前にテストを実施



Amazon ECSのDeployment Controller

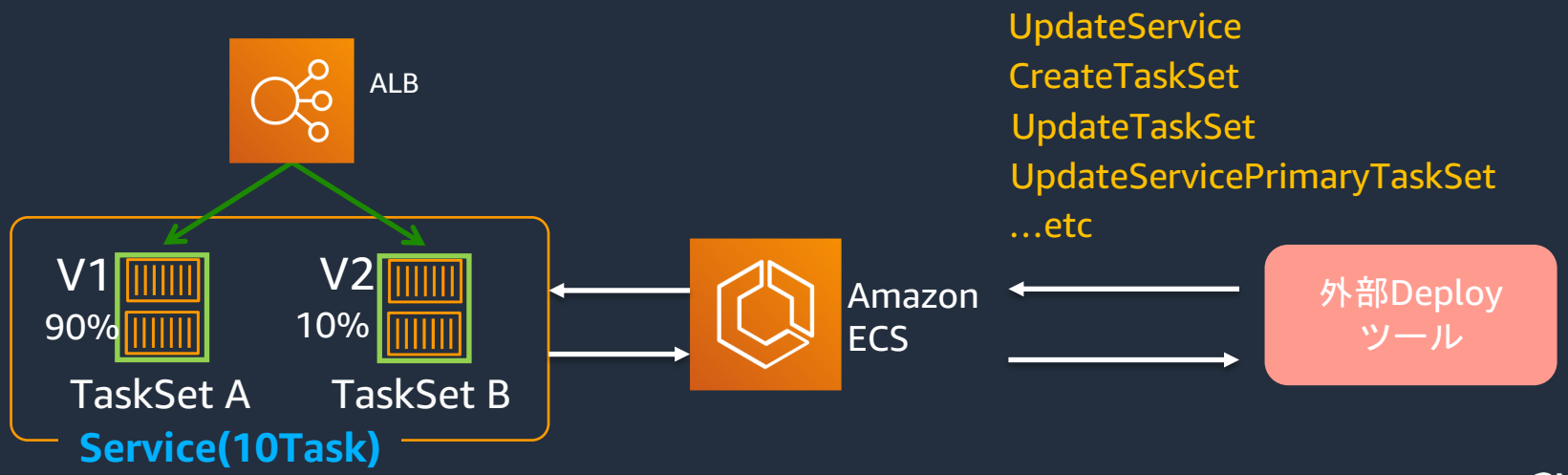
Blue/Green デプロイメント

- AWS CodeDeploy,ALBでコントロール
- Green タスク群に本番トラフィックを流す前にテストを実施



External Deployment Controller *NEW!!*

- 外のツールから API 経由でコントロール
- 自分たちでカスタマイズしたデプロイ方式をとりたい際に採用
- 一つのServiceの中に複数のTaskSetを維持しトラフィック分散可能



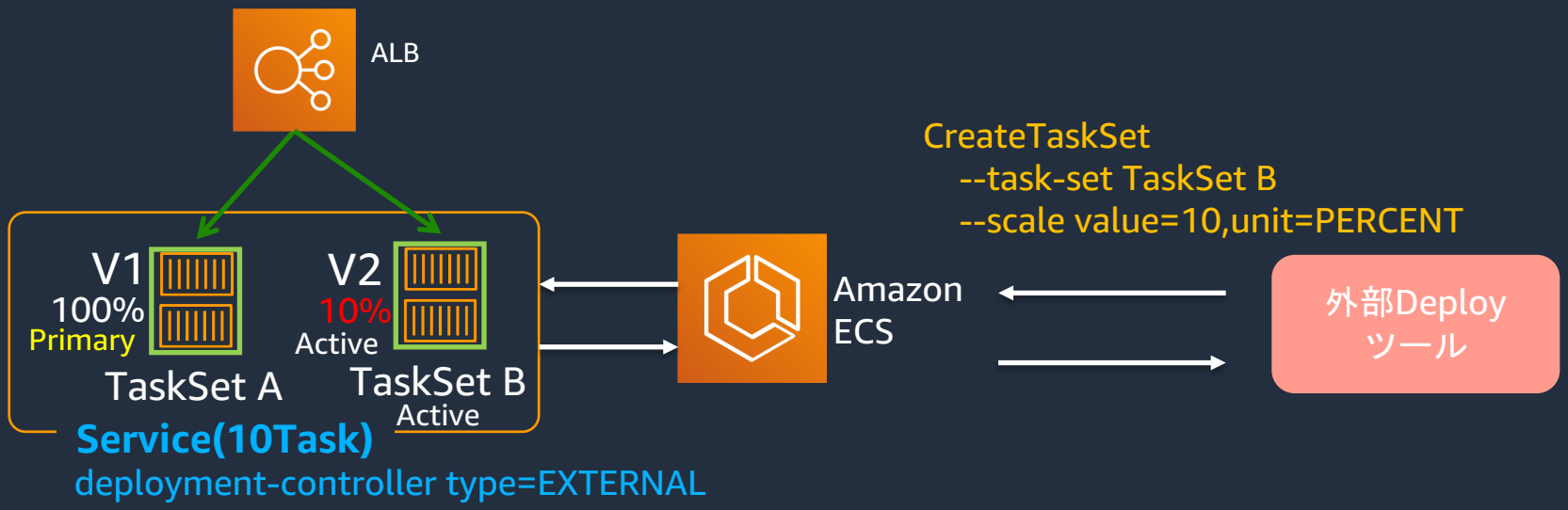
External Deployment Controller *NEW!!*

- **考慮すべき点**
  - サービスの Scale-In/Out はお客様ご自身での実装が必要
  - マネージメントコンソールからはDeployment type=externalは設定ができない
    - APIもしくはCLIでserviceを作成
- **実装例**
  - A/Bテスト
  - カナリアデプロイメント
  - …and more

External Deployment Controllerを使ったデプロイイメージ

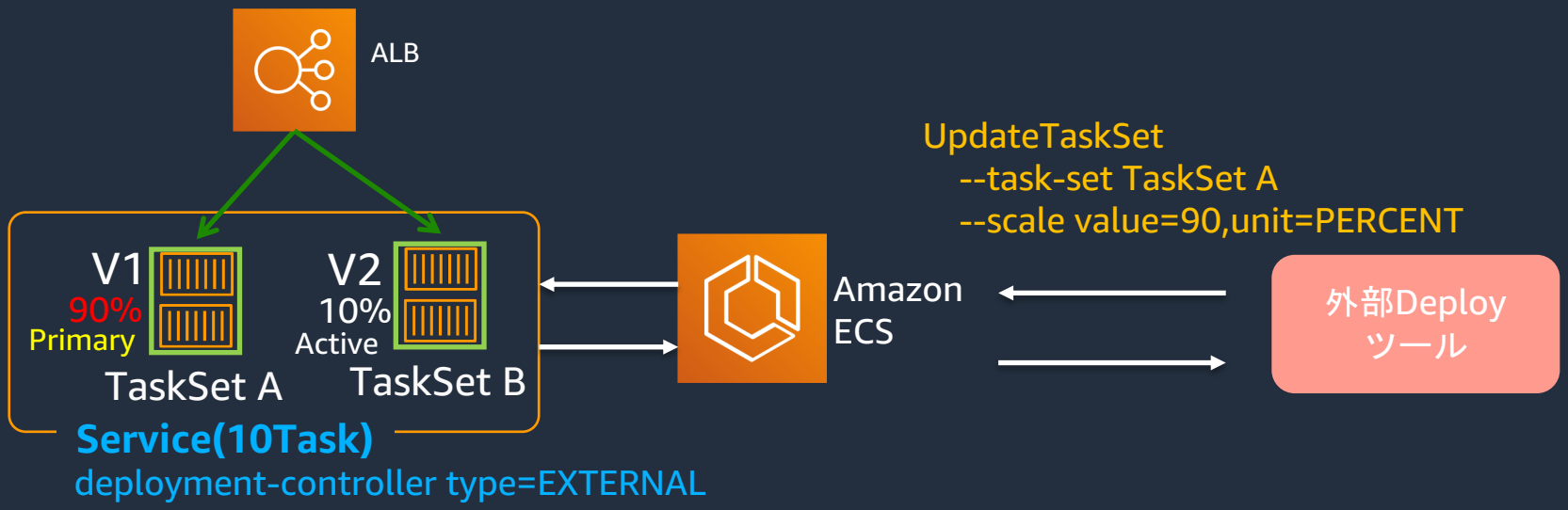
External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



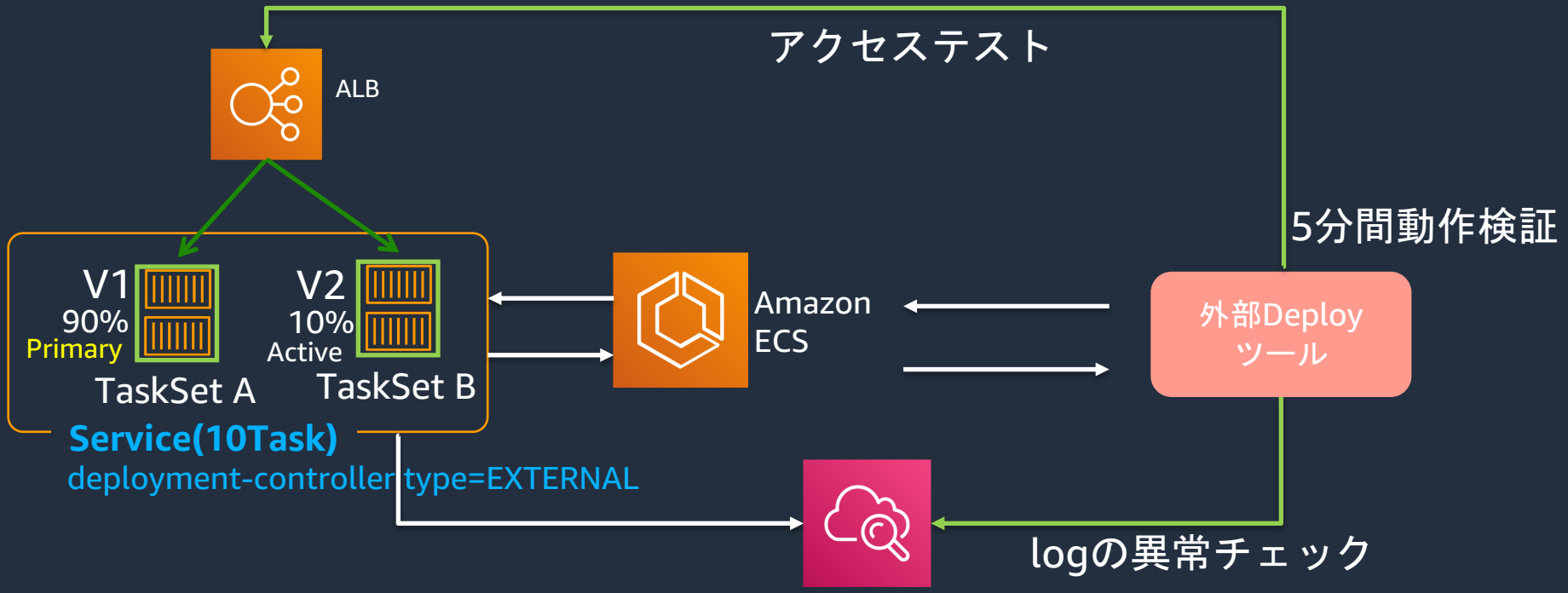
External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



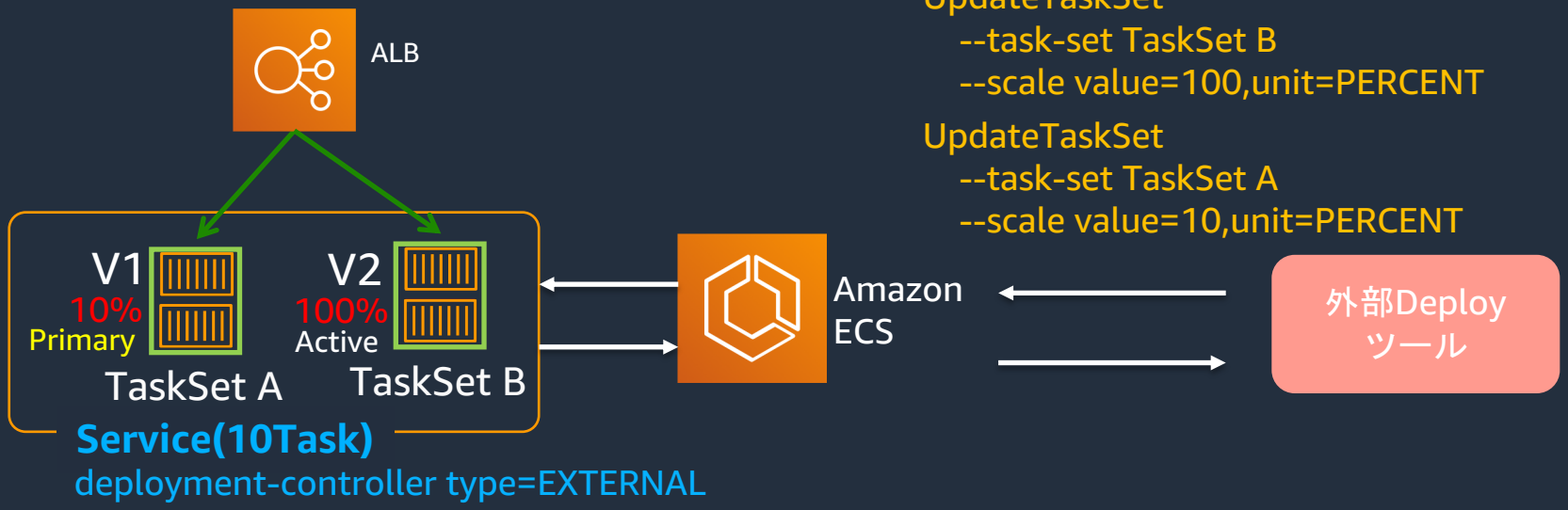
External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



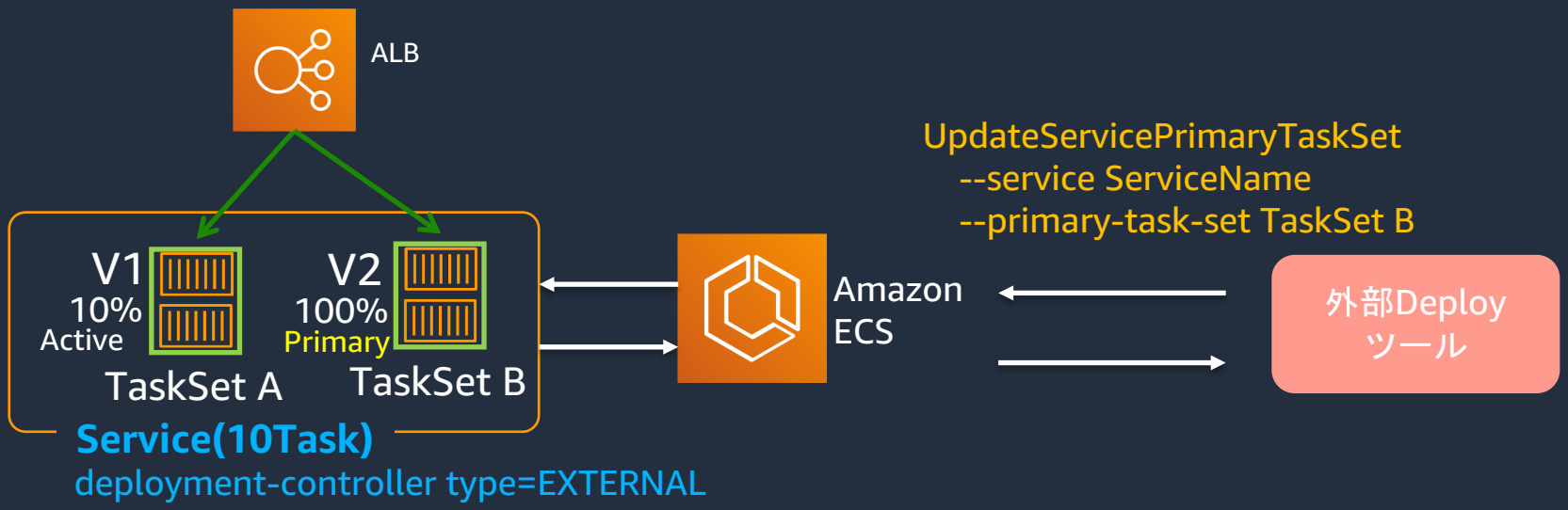
External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



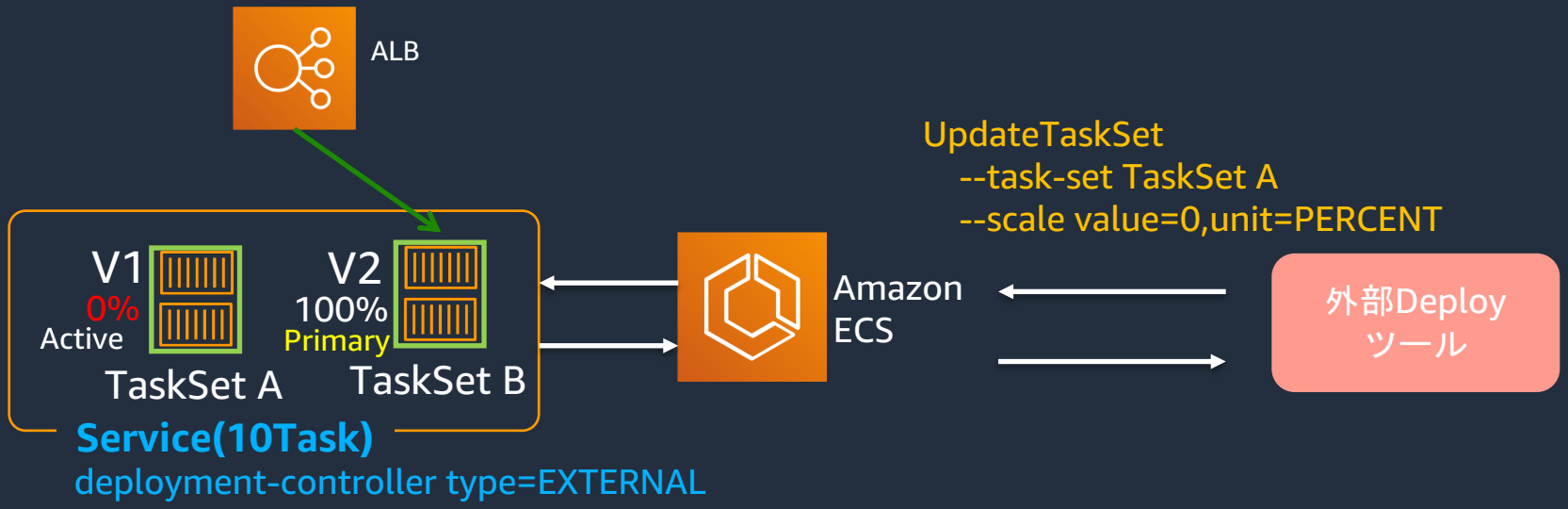
External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

External Deployment Controllerを使ったデプロイイメージ



External Deployment Controllerについて
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/userguide/deployment-type-external.html

Q. EC2起動タイプでawsvpcネットワークモードで使っていると、EC2インスタンスはリソースが余っているにも関わらず、コンテナが2、3個しか立ち上がらないんです。もっとコンテナを起動するならEC2を増やすしかないですか？

このお悩みの背景

- awsvpcネットワークモードでコンテナアプリケーションを展開
- コスト軽減の目的でSpotインスタンスを使うため、EC2起動タイプを選定
- 軽量コンテナをクラスタ上に多く配置するパターン

EC2インスタンスのCPU やメモリに空きがあっても、タスクが一定数以上配置されない

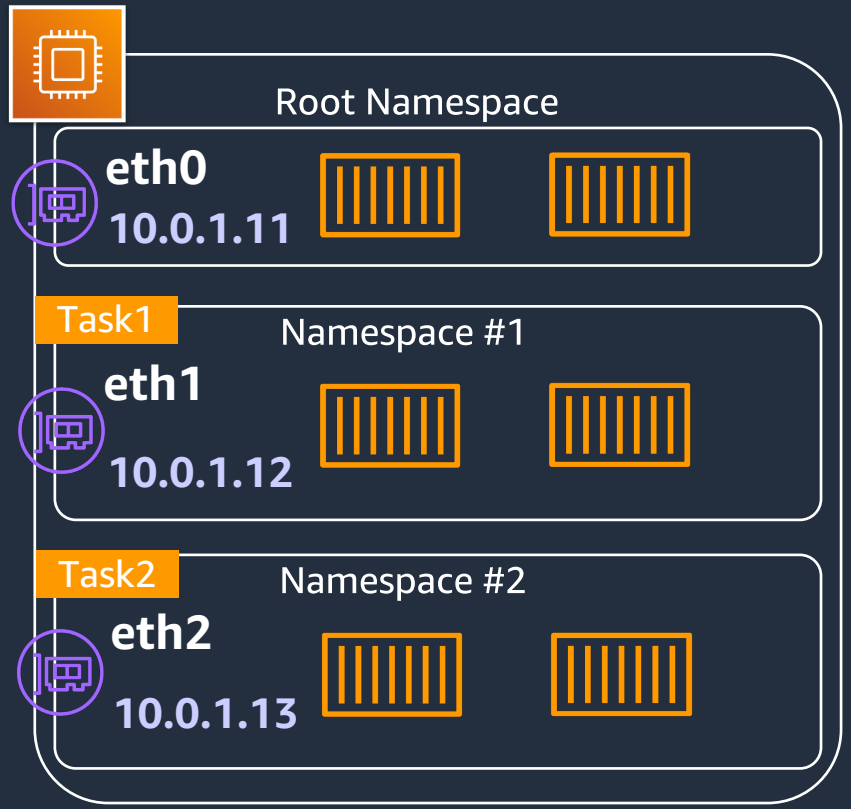
Answer：

- awsvpcネットワークモードをEC2起動タイプでご利用している場合は、ENI Trunking機能を有効にすることで、通常時のENI上限値を高めることができ、各インスタンスに、より多くのタスクを配置可能に

awsvpc ネットワークモード

- Task単位でENIを自動割り当て
  - Security Groupをタスク単位で設定可能
  - Application Load BalancerとNetwork Load BalancerにIPターゲットとして登録可能
  - VPC内の他のリソースへPrivate IPで通信可能

→コンテナワークロードに最適化されたネットワークポリシーが定義できる



awsvpcネットワークモードでのENI上限制約

- タスク毎に1ENIをEC2にアタッチすることで実現しているので、EC2に配置可能なタスク数が EC2 にアタッチ可能 ENI 数上限の影響を受ける
- CPU やメモリに空きがあっても ENI 数上限に達していると追加のタスクが配置できない
- 例: c5.large 2タスク/インスタンス

インスタンスタイプ	タスク上限
t3.micro	1
c5.large	2
c5.xlarge	3
c5.2xlarge	3
c5.4xlarge	7
c5.9xlarge	7

各インスタンスタイプのENIの上限数
https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-eni.html#AvailableIpPerENI

ENI Trunkingを有効にする *NEW!!*

- ECSのAccount SettingsでAWSVPC Trunkingを有効にすることで、ENI密度の高いコンテナインスタンスを利用できる
- awsvpc ネットワークモードを使用して、今までより多くのタスクを配置可能に
- 例: c5.large 10タスク/インスタンス

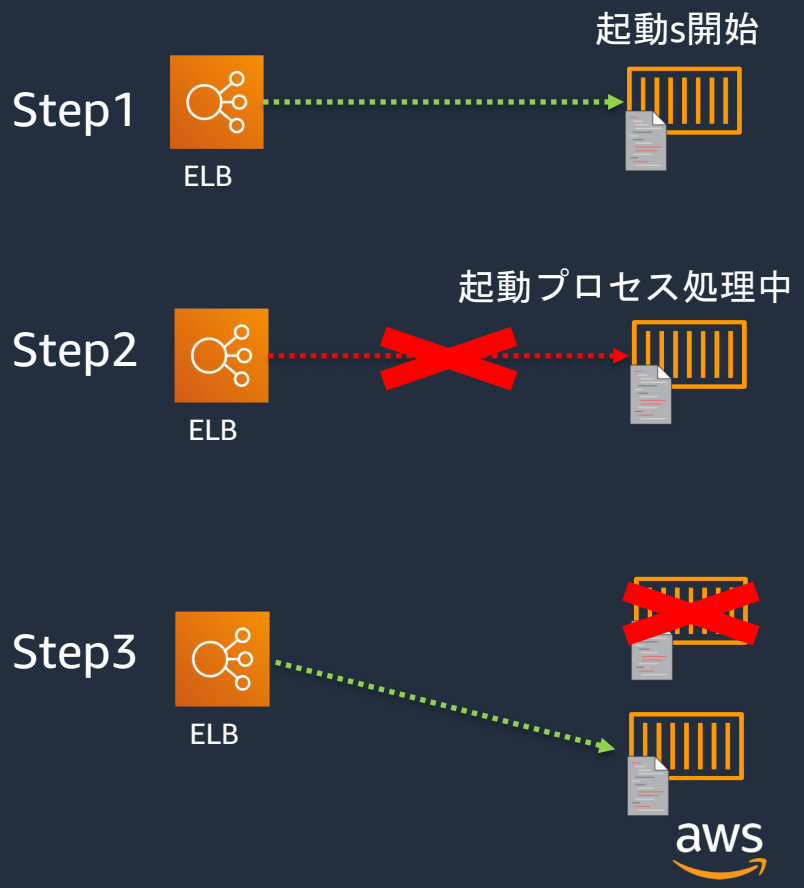
インスタンスタイプ	タスク上限
t3.micro	非対応
c5.large	10
c5.xlarge	20
c5.2xlarge	40
c5.4xlarge	60
c5.9xlarge	80

AWSVPC Trunkingがサポートされる Amazon EC2 インスタンスタイプ、及びそのENI上限
https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/using-eni.html#AvailableIpPerENI

Q. Javaアプリケーションが稼働するコンテナの起動時間が長くてALBのヘルスチェックが失敗してタスクが実行されないです。ヘルスチェックの間隔は通常運転時の長さにしておきたいです。

このお悩みの背景

- JavaのアプリケーションをコンテナでECSを利用して展開
- 起動プロセスに時間がかかり、ヘルスチェックに失敗し、タスクの再実行が起きる
- 今はヘルスチェックの間隔自体を長くしている
- 起動時のみの問題なのでできれば長くしたくない



Answer：

- ヘルスチェックの猶予期間（healthCheckGracePeriodSeconds）をアプリケーションに合わせて設定する

- ヘルスチェックの実行間隔は通常通りのまま、ALBのヘルスチェックが有効になるまでの猶予期間が設定される

ヘルスチェックの猶予期間を設定する

- Service定義パラメータにて、ヘルスチェックの猶予期間を設定（healthCheckGracePeriodSeconds）

- ECSサービススケジューラが、タスクが開始時にヘルスチェック(ELB,Route53,ECS,コンテナ)のステータスを無視する期間

パブリック IP の自動割り当て　DISABLED

ヘルスチェックの猶予期間

サービスのタスクで、ELB ヘルスチェックを開始して応答するまでに時間がかかる場合は、ヘルスチェックの猶予期間として最大 2,147,483,647 秒まで指定できます。この間は、ECS サービススケジューラは ELB ヘルスチェックのステータスを無視します。この猶予期間により、ECS サービススケジューラがタスクを異常とマークして時間より前に停止することがなくなります。これが有効であるのは、ロードバランサーを使用するようにサービスが設定されている場合のみです。

ヘルスチェックの猶予期間　120

Load balancing

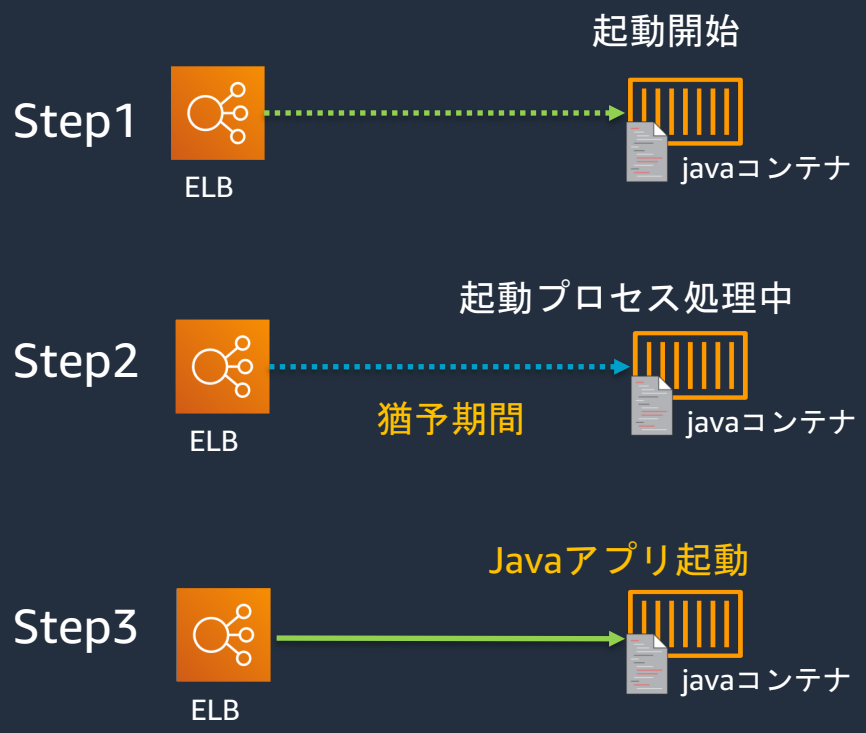
Elastic Load Balancing ロードバランサーはサービス内で実行中のタスク間に受信トラフィックを分散させます。既存のロードバランサーを選択するか、Amazon EC2 コンソールでロードバランサーを作成してください。

ロードバランサーの種類*

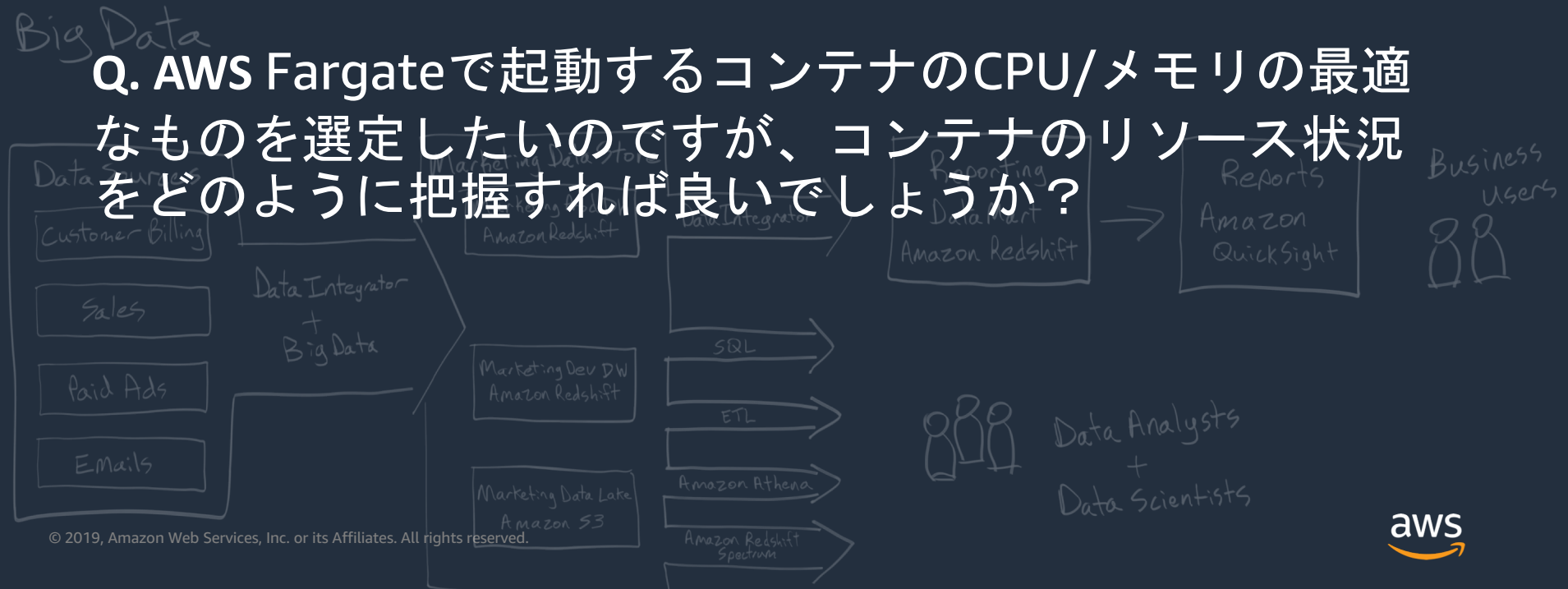
- なし
  サービスでロードバランサーは使用されません。
- Application Load Balancer
  コンテナで動的ホストポートマッピングを有効にします (コンテナインスタンスごとに複数のタスクを許可)。ルールベースのルーティングとパスを使用することで、1 つのロードバランサーの同じリスナーポートを複数のサービスで共有できます。

healthCheckGracePeriodSeconds

- ヘルスチェックの実行間隔は通常通りのまま
- Dockerfile内に時間のかかる処理を記載しなくて良い



Q. AWS Fargateで起動するコンテナのCPU/メモリの最適なものを選定したいのですが、コンテナのリソース状況をどのように把握すれば良いでしょうか？



このお悩みの背景

- Fargate上で展開しているタスク内のコンテナのリソース使用状況を元に、最適なCPU/メモリを選定してコスト最適化を測りたい
- CloudWatch にデフォルトで送信される Fargate のメトリクスはサービスレベルの CPU/メモリの使用率

タスク/コンテナ単位でのリソース状況を継続的に閲覧したい

Answer：

- プレビュー機能であるCloudwatch Container Insightsをご利用いただくことで、タスクやコンテナ単位のメトリクスを閲覧可能に

- インスタントにタスクやコンテナのメトリクスが必要な際は、ECSタスクメタデータエンドポイントで確認も可能

AWS Fargateの取得可能なメトリクス

- CloudWatch でネイティブに確認できる Fargate のメトリクスはサービスレベルの CPU/メモリ 使用率でした
- Taskやコンテナレベルのリソース状況の確認にはECSタスクメタデータエンドポイントのstatsを分析する必要があった

(詳細についてはDocker API ドキュメント参照)

169.254.170.2/stats

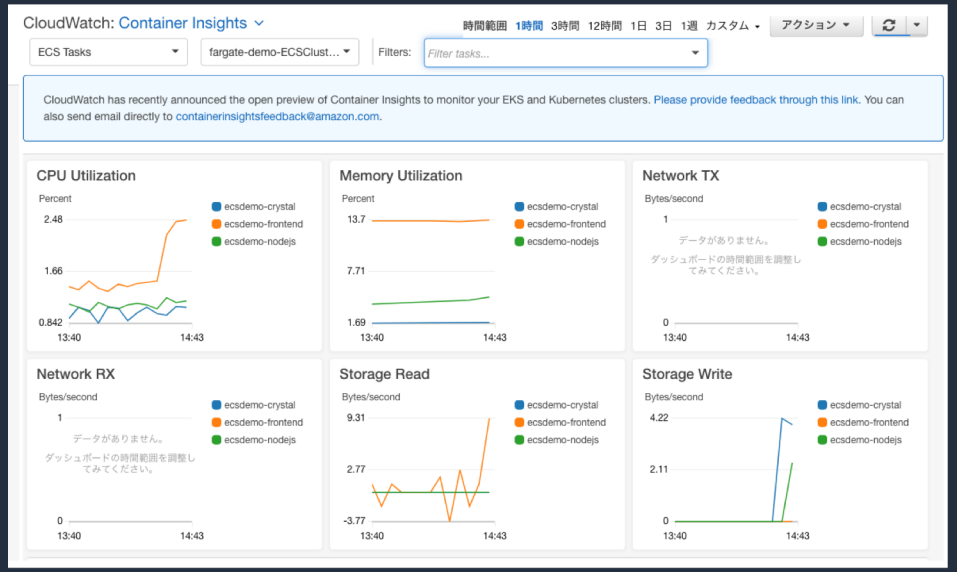
```
{
   ・・・
   "read": "2019-07-18T20:11:25.485932734Z",
   “pids_stats”: {・・・},
   "networks": {・・・},
   "memory_stats": {・・・},
   "blkio_stats": {・・・},
   "cpu_stats": {・・・},
   "precpu_stats": {・・・}
}
```

Docker API ドキュメントの「ContainerStats」
https://docs.docker.com/engine/api/v1.30/#operation/ContainerStats

Cloudwatch Container Insights

Preview

- ECSのCluster/Service/Taskレベルでメトリクスとログを収集し、リソース状況を把握できる
- Account Settingsで「CloudWatch Container Insights」を有効にすることで、以降作成されたクラスタでメトリクスが取得される



Amazon ECS CloudWatch Container Insightsの有効化手順
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/cloudwatch-container-insights.html

Cloudwatch Container Insights

Preview

- ECS メタデータAPIからのメトリクスの取得が不要に
- 以下のようなメトリクスを取得可能
    - CPU使用率
    - メモリ使用率
    - Network受信/送信
    - 現在RUNNING状態にあるタスクの数
    - 現在PENDING状態にあるタスクの数
    - サービスで実行されているタスクの数
    - …etc

Amazon ECS Container Insights Metrics一覧
https://docs.aws.amazon.com/ja_jp/AmazonCloudWatch/latest/monitoring/Container-Insights-metrics-ECS.html

Cloudwatch Container Insights

Preview

- パフォーマンスログをCloudwatch logs insightsでコンテナ単位で分析も可能

/aws/ecs/containerinsights/fargate-demo-ECSCluster-KK51UP5NJJE1/perfo…

2019-07-24 (13:00:00) - 2019-07-24 (15:00:32)

```
fields ispresent(TaskId) as hasTaskId, @message
| filter hasTaskId and TaskId like /1ac5d75d-1ad8-4a56-8f65-8c605e6d6ef1|771913Z2-99de-46c5-b67e-c24073d4dca6|7b936ece-b985
```

クエリの実行 | アクション | サンプルクエリ | フィードバックがある場合 E メールをお送りください。

ログ | 可視化

30 20 10 0
01 PM 01:15 01:30 01:45 02 PM 02:15 02:30 02:45 03 PM

6,332 レコードは次と一致しました | 10,109 レコード (3.3 秒 @ 3,060 レコード/秒 (1.7 MB /秒でスキャンされた 5.6 MB))

#	hasTaskId	@message
1	1	{"Version":"0","Type":"Container","ContainerName":"~internal~ecs~pause","TaskId":"a3b9a3e1-5907-421a-9…
2	1	{"Version":"0","Type":"Container","ContainerName":"ecsdemo-nodejs","TaskId":"a3b9a3e1-5907-421a-9c47-0…
3	1	{"Version":"0","Type":"Container","ContainerName":"~internal~ecs~pause","TaskId":"04122db6-75f7-408a-9…
4	1	{"Version":"0","Type":"Container","ContainerName":"ecsdemo-crystal","TaskId":"04122db6-75f7-408a-9a52-…
5	1	{"Version":"0","Type":"Container","ContainerName":"~internal~ecs~pause","TaskId":"1c5bfcd0-c5e6-447f-b…
6	1	{"Version":"0","Type":"Container","ContainerName":"ecsdemo-crystal","TaskId":"1c5bfcd0-c5e6-447f-b708-…
7	1	{"Version":"0","Type":"Container","ContainerName":"ecsdemo-nodejs","TaskId":"ad1289fd-1be7-4e80-97ad-e…

クエリのヘルプ | Learn m

コマンド
- fields
- filter
- stats
- sort
- limit
- parse

検出されたフィールド

フィールドの検索

- @ingestionTime
- @logStream
- @message
- @timestamp
- ClusterName
- Timestamp
- Type
- Version
- ServiceName
- CpuUtilized
- MemoryUtilized
- StorageReadBytes

（補足）本日紹介しなかった主なアップデート

- Integration with App Mesh
  - Amazon ECSのタスク定義がAWS App Meshとの統合をサポート
- Fluent bit plugin
  - Elasticsearch, Redshift, Kinesis, S3 などへのログ集約を容易にする Fluent Bit プラグインと、それらプラグインを同梱した Pre-built なコンテナイメージをリリース
- ecs-cli local option
  - ecs-cli を使用して、ローカル開発環境で ECS タスク定義を実行できるように

（補足）本日紹介しなかった主なアップデート

- ECS/ECR support for AWS PrivateLink
  - Amazon ECS および Amazon ECR が AWS PrivateLink でのアクセスが可能に
- GPU Pinning
  - ECS タスク定義で、特定のコンテナに割り当てる GPU の数を指定できるように
- Multiple target groups per ECS service
  - 1つの ECS Service に複数のターゲットグループを設定できるように

AWS コンテナサービスのロードマップの一部を公開

aws / containers-roadmap

Watch 404 | Star 1,551 | Fork 50

Code | Issues 284 | Pull requests 0 | Projects 1 | Security | Insights

containers-roadmap

Updated 5 hours ago

Filter cards | Fullscreen | Menu

24 Researching

- [ECS] Support use of swap memory
  #322 opened by mhowell-ims
  ECS Proposed
- [ECS] Increase maximum number of containers in a task definition from 10 to 30
  #368 opened by coultn
  ECS
- EKS - Cost Options on Control Plane (developer friendly)
  #45 opened by jpoley
  EKS Proposed
- [EKS]: EKS Cluster Tagging Propogation
  #374 opened by tabern
  EKS
- [EKS] [request]: AWS Secrets Manager / SSM Parameter Store
  #168 opened by jaronsummers
  EKS Proposed
- [ECS] [request]: Tasks scheduled with the DAEMON strategy should not be stopped when container instance is placed into DRAINING
  #128 opened by joshuabaird

38 We're Working On It

- [EKS]: Next Generation AWS VPC CNI Plugin
  #398 opened by tabern
  EKS Proposed
- [ECS] Mixed On-Demand/Spot tasks and services in an ECS cluster, with automatic scaling of On-Demand and Spot instances
  #391 opened by coultn
  ECS
- Fargate ephemeral volume encryption
  #314 opened by akshayram-wolverine
  Fargate
- Fargate in Paris data center
  #309 opened by daya
  Fargate Proposed
- New EKS Region: GovCloud West
  #253 opened by joemccall86
  EKS Proposed
- New EKS Region: GovCloud East
  #375 opened by tabern
  EKS
- [EKS] [request]: Add ability to set oidc options

12 Coming Soon

- [ECS] CloudFormation support for secrets configuration with logging drivers
  #393 opened by coultn
  ECS
- [ECS, Fargate] CloudFormation support for scheduled tasks with Fargate Launch Type
  #392 opened by coultn
  ECS Fargate
- [ECS] [CloudFormation]: CloudFormation Support for ipcMode and pidMode (TaskDefinition)
  #286 opened by joeinnyc
  ECS Proposed
- [ECS] [CloudFormation]: CloudFormation support for SystemControls, interactive, and pseudoTerminal
  #96 opened by talawahtech
  ECS Proposed
- ECS support for multiple target groups per service
  #12 opened by abby-fuller
  ECS Fargate
- New EKS Region : Ningxia

6 Developer Preview

- Can't see task level cpu/memory utilization in cloudwatch
  #106 opened by rutchkiwi
  ECS Proposed
- [ECS] [Metrics]: Container Insights for ECS
  #70 opened by bradenschmidt
  ECS Fargate Proposed
- [ECS, Fargate] CloudWatch metrics for number of running and pending tasks per service and cluster
  #282 opened by coultn
  ECS Fargate
- [ECS,Fargate] Multiple target groups for a service
  #104 opened by zaitsman
  ECS Fargate Proposed
- EKS Windows Nodes (preview)
  #69 opened by ofiliz
  Developer Preview EKS
- [EKS]: Support for Arm Nodes - EC2 A1 Instances
  #264 opened by tabern
  Developer Preview EKS

65 Just Shipped

- [EKS]: Kubernetes v1.10 Deprecation (July 22, 2019)
  #300 opened by tabern
  EKS
- [ECS, App Mesh] App Mesh integration in the ECS console
  #258 opened by coultn
  Console ECS Fargate
- [EKS]: Release CNI v1.5.0
  #284 opened by mogren
  EKS
- [ECR] Increase some of ECR's default service limits
  #414 opened by jtoberon
  ECR
- [ECR] [amazon-ecr-credential-helper]: provide a binary artifact as part of a tagged release
  #406 opened by jtoberon
  ECR
- [ECS] Run ECS task definitions locally
  #180 opened by clareliguori
  ECS Fargate
- [ECS][Fargate] Service and TaskSet

https://github.com/aws/containers-roadmap/projects/1

最後に

- お客様のお悩み相談と共にコンテナワークロードの運用の負担を軽減するアップデートの機能を解説
- 他のサービスとの統合機能も多くリリースされています
    - コンテナ環境を1要素として適切に組み合わせて利用
- **チームや要件に合わせた設計や機能の選定が大切です**

Amazon ECSのご利用で、発生したお悩みや課題や要望はぜひお近くのSolutions Architectまで!!

Q&A

ご質問については,
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」
にて資料公開と併せて、後日掲載します

8 月の Black Belt Online Seminar 配信予定

https://amzn.to/JPWebinar

08/06 (火) 12:00-13:00 AWS Glue
08/13 (火) 12:00-13:00 実践的 Serverless セキュリティプラクティス
08/14 (水) 18:00-19:00 AWS Serverless Application Model
08/20 (火) 12:00-13:00 Serverless モニタリング
08/21 (水) 18:00-19:00 AWS AppSync
08/28 (水) 18:00-19:00 Amazon Aurora with PostgreSQL Compatibility






AWS の日本語資料の場所「AWS 資料」で検索

aws

日本担当チームへお問い合わせ　サポート　日本語　アカウント　コンソールにサインイン

製品　ソリューション　料金　ドキュメント　学習　パートナー　AWS Marketplace　その他

AWS クラウドサービス活用資料集トップ

アマゾン ウェブ サービス (AWS) は安全なクラウドサービスプラットフォームで、ビジネスのスケールと成長をサポートする処理能力、データベースストレージ、およびその他多種多様な機能を提供します。お客様は必要なサービスを選択し、必要な分だけご利用いただけます。それらを活用するために役立つ日本語資料、動画コンテンツを多数ご提供しております。(本サイトは主に、AWS Webinar で使用した資料およびオンデマンドセミナー情報を掲載しています。)

AWS Webinar お申込 »　AWS 初心者向け »　業種・ソリューション別資料 »　サービス別資料 »

https://amzn.to/JPArchive

aws

公式Twitter/Facebook
AWSの最新情報をお届けします

@awscloud_jp

検索

もしくは
http://on.fb.me/1vR8yWm

最新技術情報、イベント情報、お役立ち情報、
お得なキャンペーン情報などを日々更新しています！

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive