[AWS Black Belt Online Seminar]
Amazon Elastic Container Service

サービスカットシリーズ

Partner Solutions Architect
金森政雄
2020/04/22

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

過去資料
https://amzn.to/JPArchive

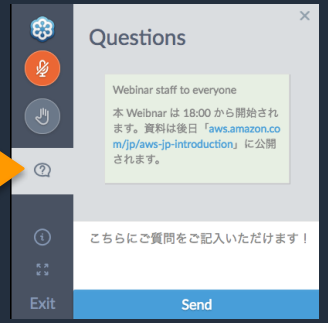
AWS Black Belt Online Seminar とは

「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
- 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

自己紹介

金森 政雄

- 所属/役職：
  パートナー技術本部/
  パートナーソリューションアーキテクト
- 好きなサービス

Amazon Elastic Container Service

AWS Step Functions

- リモートワークの過ごし方
  - 午後に20 分間の仮眠
  - 定期的な小休憩

内容についての注意点

- 本資料では2020年04月22日現在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には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.

本セミナーの概要

- 本セミナーで学習できること
  - AWS が提供するコンテナオーケストレーションツールである、Amazon Elastic Container Service (ECS)の基本と機能概要

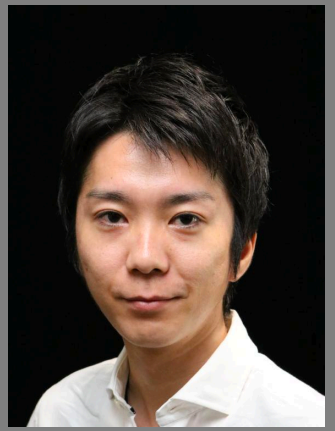
- 対象者
  - コンテナ(docker)をクラウドでの本番環境で利用することを検討されている方
  - コンテナのオーケストレーションツールに興味のある方

前置き

※本セッションでは、「コンテナとは何か」や、コンテナを活用するための基本的な開発/運用の考え方については取り扱いません。そのような情報が必要の方は、下記のAWS Summit の講演資料などをご参照ください。

AWS Summit 2019
【初級】AWS コンテナサービス入門

動画:
https://youtu.be/L4bLDNRSYC8
資料:
https://pages.awscloud.com/rs/112-TZM-766/images/C3-01.pdf



Speaker Hara Tori

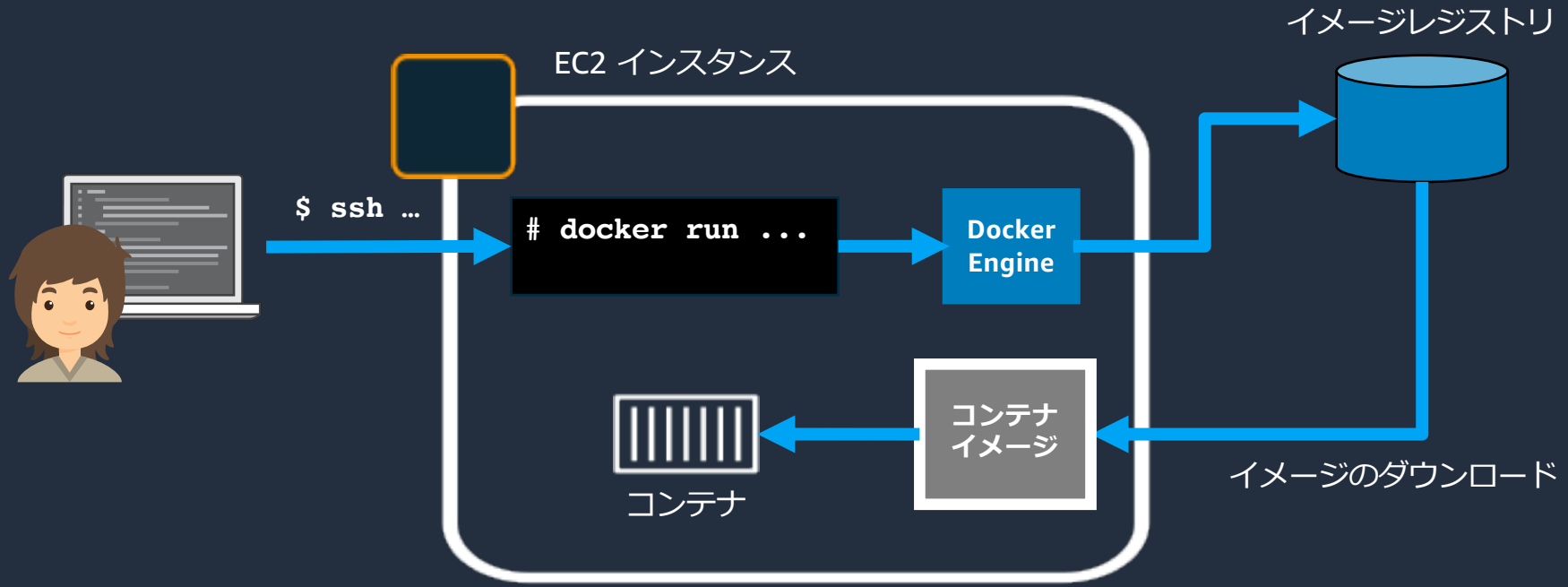
本日のアジェンダ

- AWS のコンテナサービス概要
- Amazon ECS の基本
    - Amazon ECS の主要要素
    - コンテナの実行環境
- Amazon ECS の機能紹介
    - タスク定義詳細
    - コンテナ実行

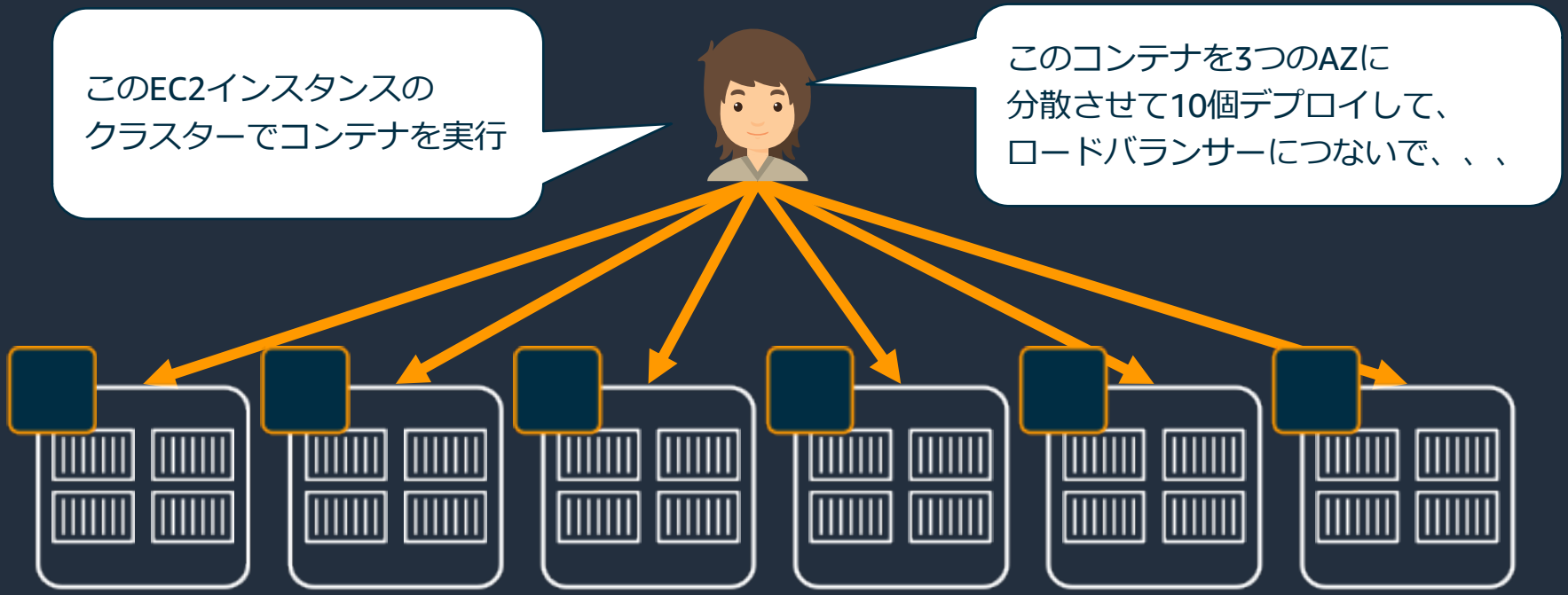
本日のアジェンダ

- AWS のコンテナサービス概要
- Amazon ECS の基本
    - Amazon ECS の主要要素
    - コンテナの実行環境
- Amazon ECS の機能紹介
    - タスク定義詳細
    - コンテナ実行

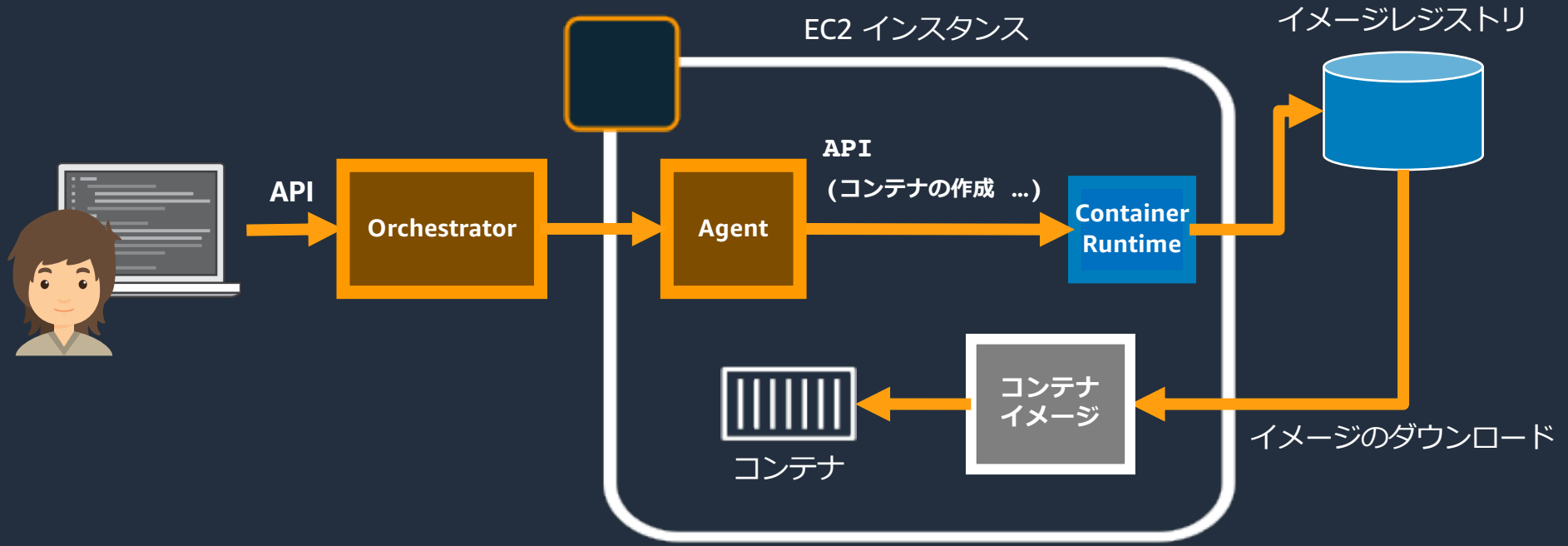
サーバ上でのコンテナ実行(docker の場合)



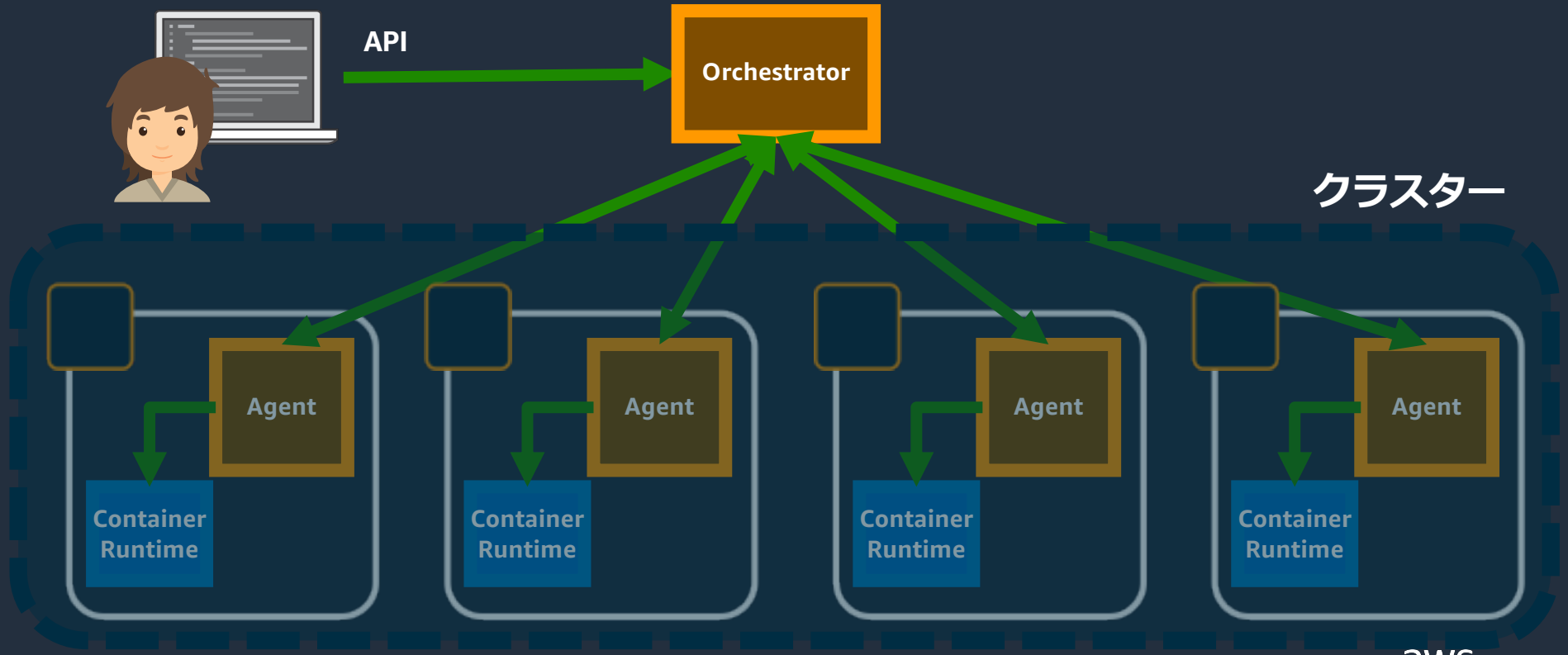
手作業でのコンテナイメージダウンロードと実行は非効率かつミスオペレーションを招く



コンテナオーケストレーションの仕組み



コンテナオーケストレーションによるクラスター管理



AWSのコンテナ関連サービス

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、クラスター管理

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）

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）

Amazon Elastic Container Service

aws

Amazon Elastic Container Service (ECS)

- クラウドでコンテナを本番環境利用するためのオーケストレーター
- 各種 AWS サービスとの高度な連携
- 多様なワークロードをサポートする「タスク」「サービス」というシンプルなリソース表現
- Linux/Windows サポート

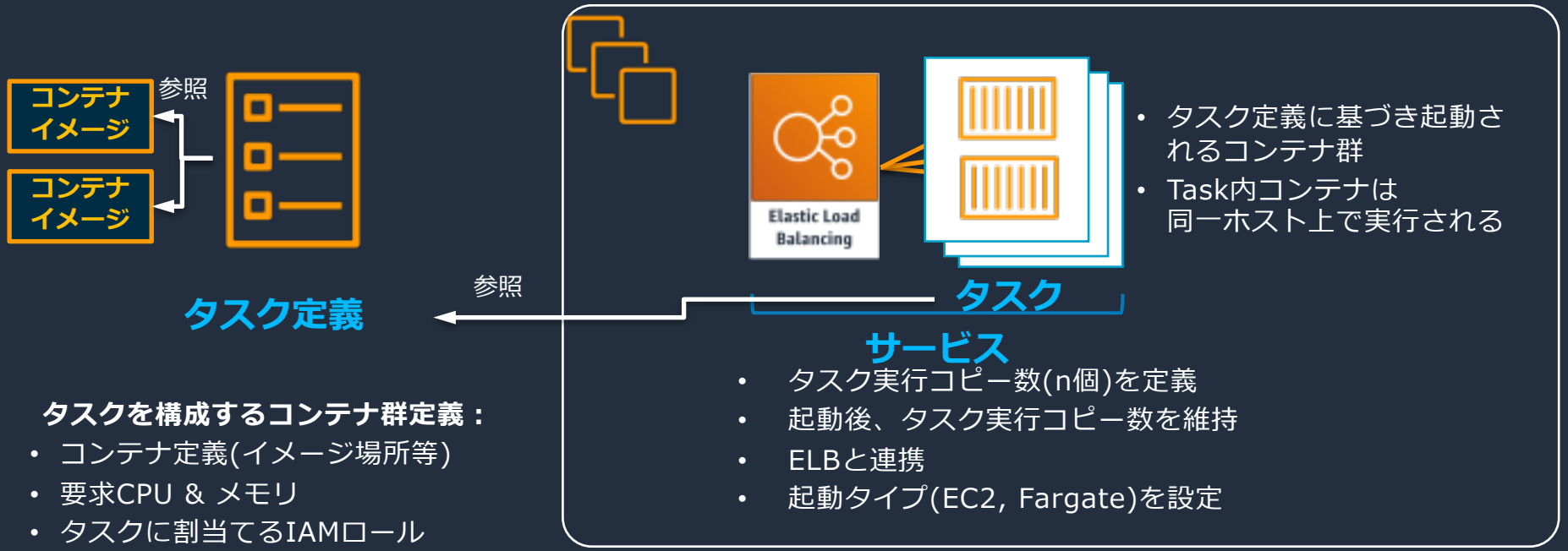
サービス紹介ページ

https://aws.amazon.com/jp/ecs/

本日のアジェンダ

- AWS のコンテナサービス概要
- Amazon ECS の基本
  - Amazon ECS の主要要素
  - コンテナの実行環境
- Amazon ECS の機能紹介
  - タスク定義詳細
  - コンテナ実行

Amazon ECSの主要要素



- タスク定義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

タスク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
- タスクに割当てるIAMロール
- ネットワークモード
- etc...

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

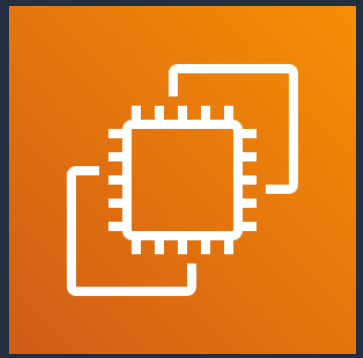
- 実行環境の境界
- IAM権限の境界(クラスタに対する操作)
- スケジュールされたタスクの実行を設定可能

本日のアジェンダ

- AWS のコンテナサービス概要
- Amazon ECS の基本
  - Amazon ECS の主要要素
  - コンテナの実行環境
- Amazon ECS の機能紹介
  - タスク定義詳細
  - コンテナ定義詳細
  - コンテナ実行

コンテナの実行環境

ECS で管理されるコンテナはどこで動く?



Amazon EC2

AWS Fargate

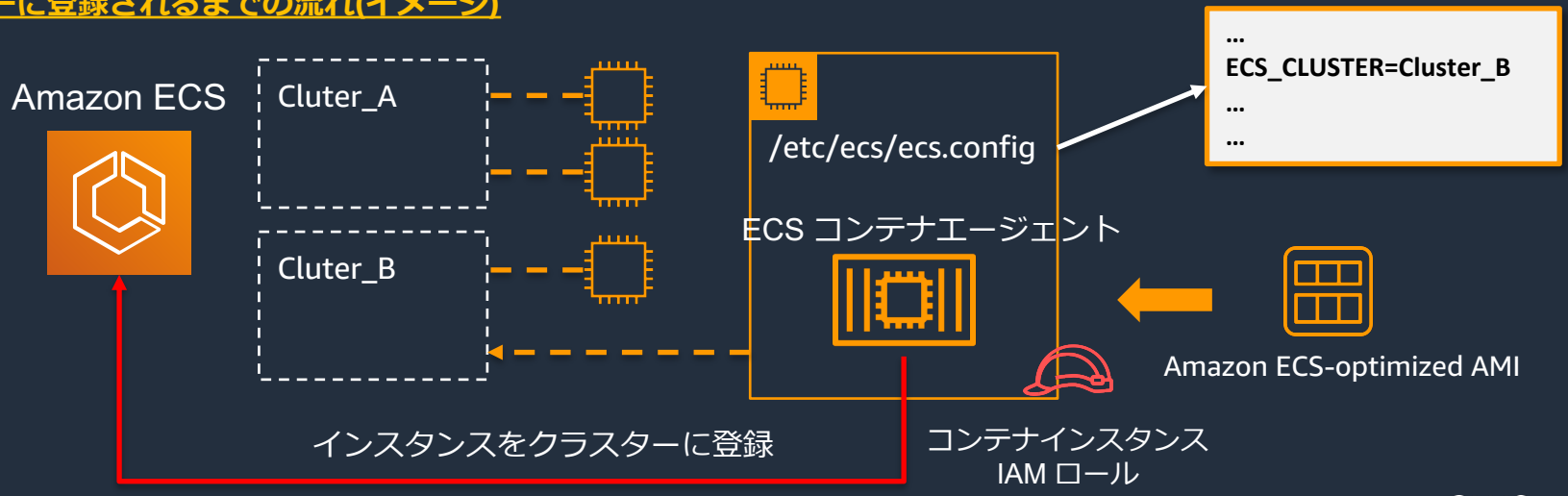
→選択肢はEC2 起動タイプとFargate 起動タイプの2つ

EC2 起動タイプ – ECS コンテナインスタンス

https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/ECS_instances.html

- ECS コンテナエージェントを実行しクラスターに登録されているEC2 インスタンス
- EC2 起動タイプを使用して実行されたタスクはアクティブなコンテナインスタンスに配置される
- Amazon ECS コンテナインスタンス IAM ロールが設定されている必要がある

クラスターに登録されるまでの流れ(イメージ)



Amazon ECS-optimized AMI と ECS コンテナエージェント

Amazon ECS-optimized AMI

- コンテナインスタンスに求められる要件/推奨事項にしたがって事前構成されたAMI。下記のようなものがインストール済み
  - Docker デーモン
  - ECS コンテナエージェント etc …
- Amazon Linux 1/Amazon Linux 2/Windows / GPU などのAMI を用意

参考 : https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/ecs-optimized_AMI.html

ECS コンテナエージェント

- ECS コントロールプレーンと通信し、コンテナインスタンスの管理やタスクの実行/停止を行う
- Amazon ECS-optimized AMI を元に起動した場合、事前にインストールされている。そうではない場合、自身でインストールする必要がある
- このエージェント自体もコンテナで実行される

参考 : https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/ECS_agent.html

コンテナインスタンスのドレイン

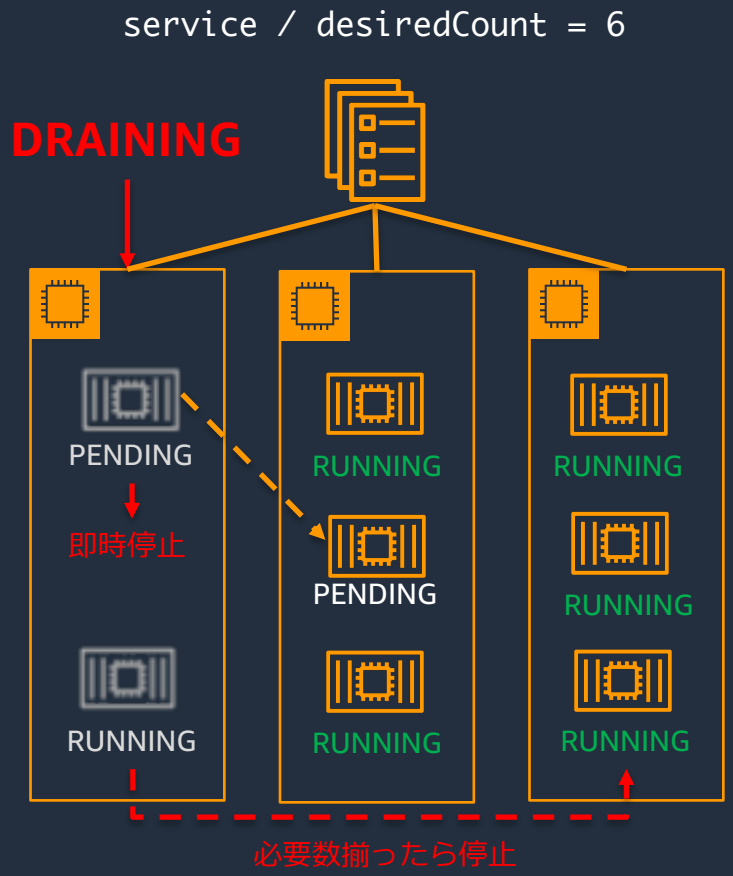
クラスターからコンテナインスタンスを削除する必要がある場合

- システム更新
- エージェント/Docker デーモンの更新
- AutoScaling のスケールイン etc

コンテナインスタンスをDRAINING に設定

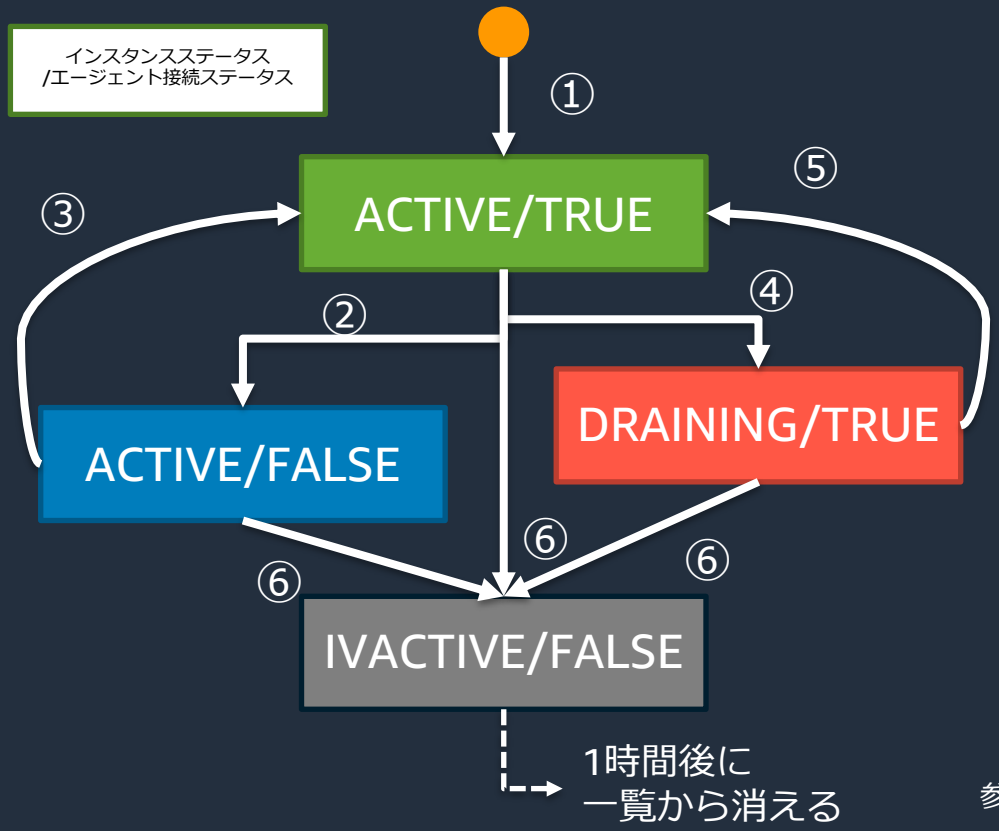
- 新規タスクは配置されない
- PENDING 状態のサービスタスクは即時停止
- RUNNING 状態のサービスタスクはサービスのデプロイ設定に従って停止し、代替されていく

参考:
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/container-instance-draining.html



コンテナインスタンスのライフサイクル

凡例



①コンテナエージェントがインスタンスをクラスターに登録

②インスタンスを停止(終了ではない)

③再起動で再接続

④インスタンスのドレインを実施

⑤ACTIVE に戻すとタスクが再度スケジュールされるように

⑥コンテナインスタンスを登録解除または終了

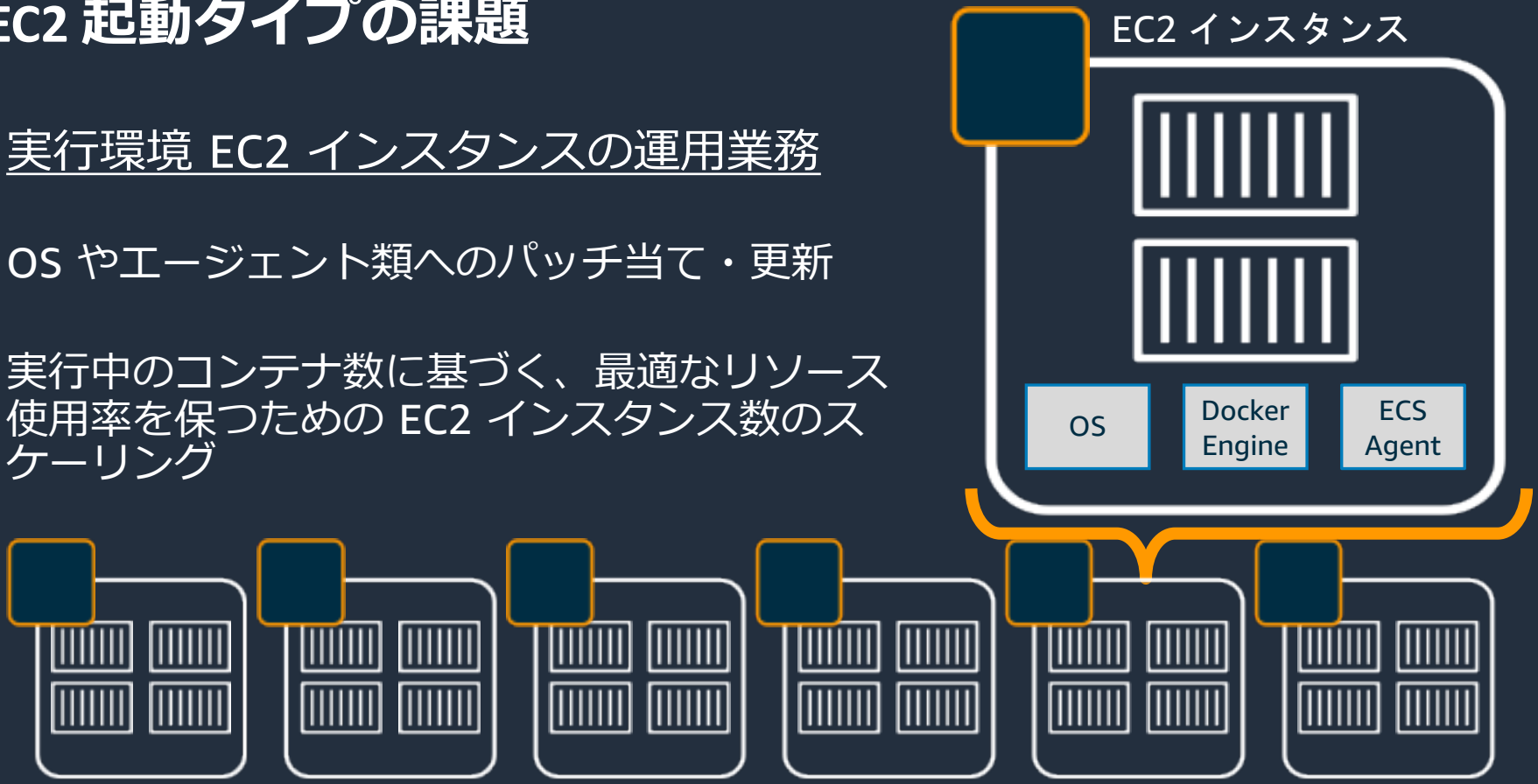
参考 https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/ECS_instances.html#container_instance_life_cycle

EC2 起動タイプの課題

実行環境 EC2 インスタンスの運用業務

OS やエージェント類へのパッチ当て・更新

実行中のコンテナ数に基づく、最適なリソース使用率を保つための EC2 インスタンス数のスケーリング



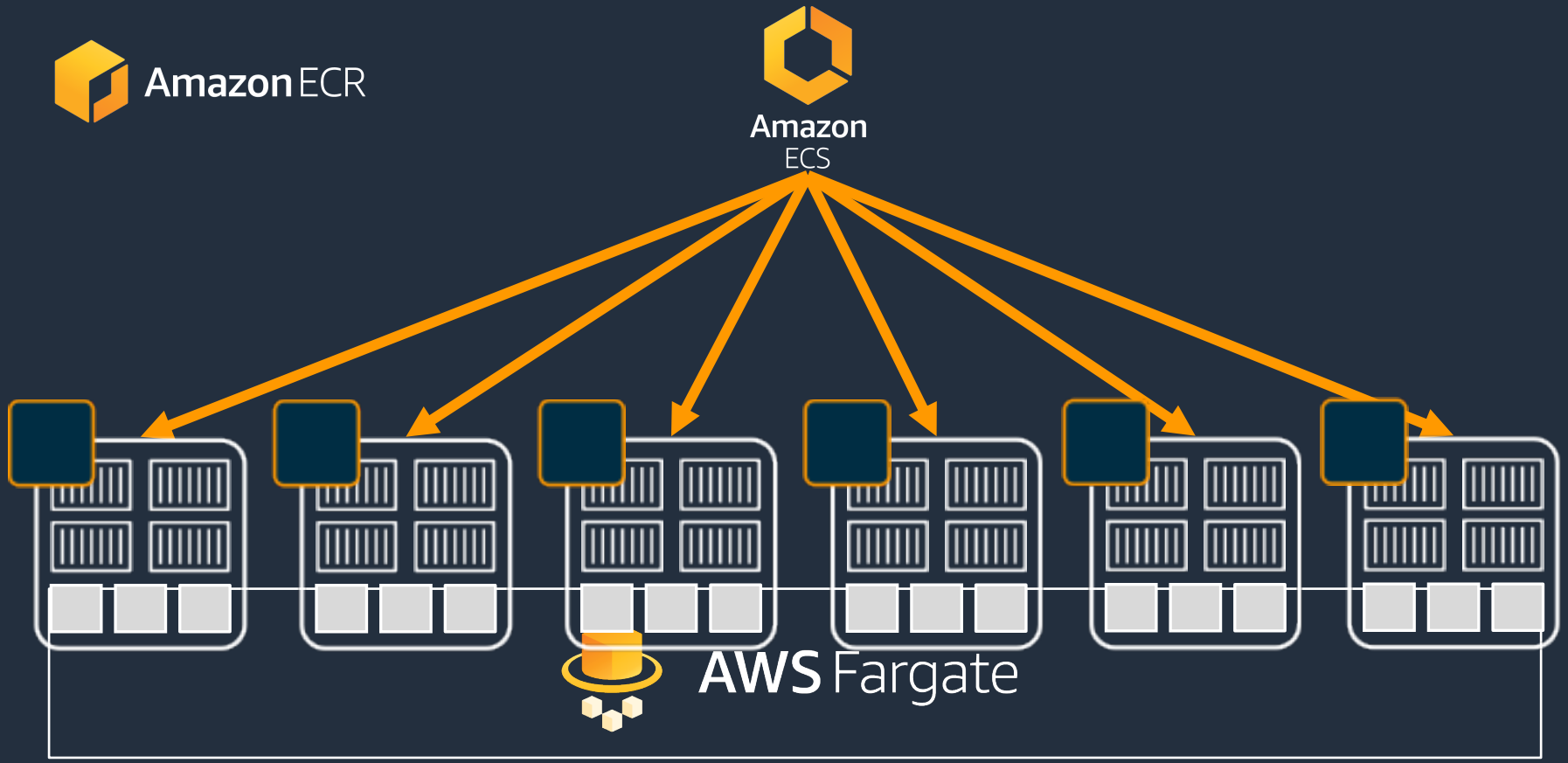
AWS Fargate

AWS マネージド
EC2 インスタンスのプロビジョン、スケール、管理不要

コンテナネイティブ
仮想マシンを意識しないシームレスなスケーリング
コンテナの起動時間・使用リソースに応じた料金設定

AWS サービスとの連携
VPC ネットワーキング、Elastic Load Balancing、IAM、CloudWatch、etc.

サービス紹介ページ
https://aws.amazon.com/jp/fargate/



Fargate : タスク割り当てCPUとメモリ設定

柔軟な設定の選択肢

50 パターンのCPU/メモリの組み合わせから選択

1 vCPU/時　＝ $0.05056

1 GB Mem/時　＝ $0.00553

(※東京リージョン、2020/04/22現在)

CPU	Memory	
256 (.25 vCPU)	0.5GB, 1GB, 2GB	→ 3種類
512 (.5 vCPU)	1GB to 4GB (1GB 刻み)	→ 4種類
1024 (1 vCPU)	2GB to 8GB (1GB 刻み)	→ 7種類
2048 (2 vCPU)	4GB to 16GB (1GB 刻み)	→ 13種類
4096 (4 vCPU)	8GB to 30GB (1GB 刻み)	→ 23種類

https://aws.amazon.com/jp/fargate/pricing/

AWS Fargate についてもっと詳しく知りたい方は

AWS Fargate についてもっと詳しく知りたい方は、下記のBlack Beltにて、詳しく説明されておりますので、そちらをご覧ください。

[AWS Black Belt Online Seminar]
AWS Fargate (2019/09/25)

動画:
https://youtu.be/rwwOoFBq2AU
資料:
https://d1.awsstatic.com/webinars/jp/pdf/services/20190925_AWS-BlackBelt_AWSFargate.pdf

Speaker 川崎 一青

本日のアジェンダ

- AWS のコンテナサービス概要
- Amazon ECS の基本
  - Amazon ECS の主要要素
  - コンテナの実行環境
- Amazon ECS の機能紹介
  - タスク定義詳細
  - コンテナ実行

タスク定義

- アプリケーションを構成する1つ以上(最大10個) のコンテナを定義するJSON 形式のテキストファイル
- 使用するコンテナ、起動タイプ、コンテナの実行方法など様々なパラメータを定義できる
- アプリケーションを実行するときは、タスク定義をインスタンス化したタスクを実行する

参考:
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task_definitions.html

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```


aws

タスク定義の詳細 – 代表的なパラメータ

必須

- ファミリー (family)
- コンテナ定義 (containerDefinitions)

オプション

- タスクロール(taskRoleArn)
- タスク実行ロール(executionRoleArn)
- ネットワークモード (networkMord)
- ボリューム (volumes)
- 起動タイプ(requiresCompatibilities)
- タスクサイズ（ cpu / memory)

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```


aws

タスク定義の詳細 – 代表的なパラメータ

必須

- ファミリー (family)
- コンテナ定義 (containerDefinitions)

オプション

- タスクロール (taskRoleArn)
- 実行ロール (executionRoleArn)
- ネットワークモード (networkMode)
- 起動タイプ(requiresCompatibilities)
- タスクサイズ（ cpu / memory)

タスク定義の名前のようなもの。タスク定義を登録する際に必ず指定する。
ファミリーとリビジョン番号(最初は1)で1つのタスク定義が特定される。

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```

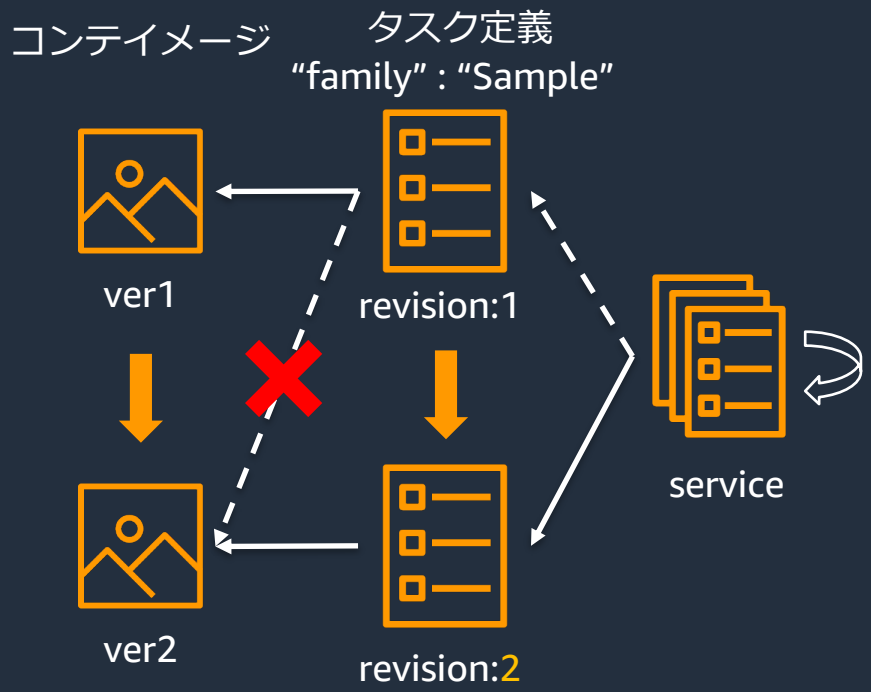
タスク定義の更新方法

タスク定義の更新

- タスク定義はイミュータブル(変更できない)
- 変更する場合はファミリーの新しいリビジョンを作成する

実行中のタスク/サービスへの影響

- 実行中のタスクは更新されない
- サービスから利用している場合、サービスを更新する必要がある



参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/update-task-definition.html

タスク定義の詳細 – 代表的なパラメータ

必須

- ファミリー (family)
- **コンテナ定義 (containerDefinitions)**

オプション

- タスクロール(taskRoleArn)
- タスク実行ロール (executionRoleArn)
- ネットワークモード (networkMode)
- 起動タイプ(requiresCompatibilities)
- タスクサイズ（ cpu / memory)

タスク実行時コンテナランタイムに渡されるコンテナ定義。コンテナのイメージやポートマッピング、メモリ制限などを指定する

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```


aws

コンテナ定義の代表的なパラメータ

名前 (name)

コンテナの名前。タスク内で複数のコンテナをリンクする場合、この名前を利用(※)。

(※ ネットワークモードがbridge の時のみ)

イメージ(image)

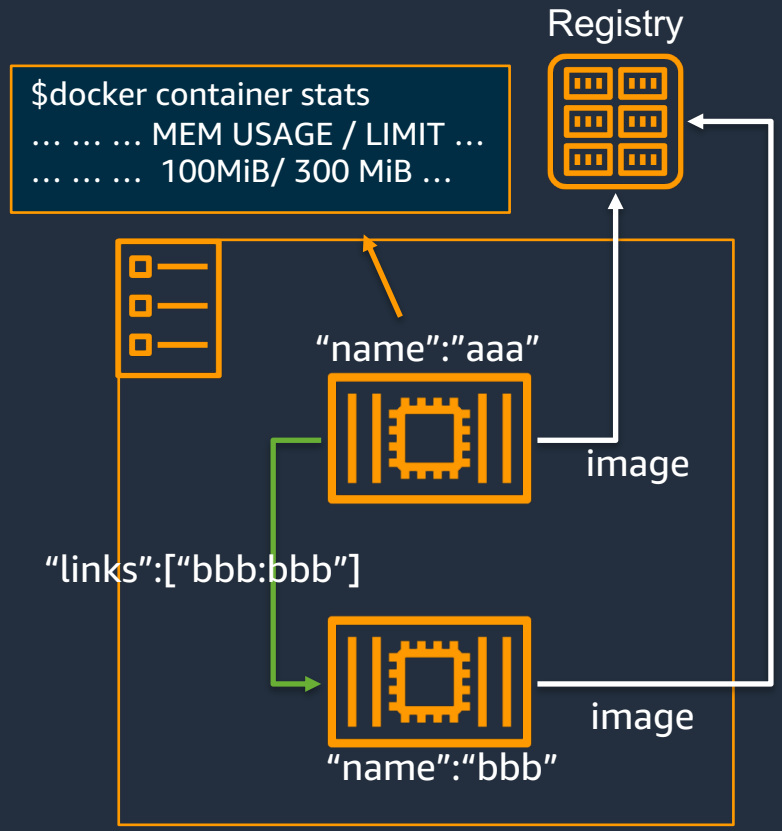
使用するコンテナのイメージを指定する

メモリ (memory / memoryReservation)

memory – コンテナに適用されるメモリのハードリミット

memoryReservation – コンテナ用に予約されるメモリのソフト制限

etc....



コンテナ定義: logConfiguration パラメータ

コンテナのログ設定を指定する

- ログドライバーの指定 (logDriver)
- ログドライバーに送信する設定(options, secretOptions)

参考 : https://docs.aws.amazon.com/AmazonECS/latest/APIReference/API_LogConfiguration.html

ログドライバーの例: awslogs ログドライバー

- ✓ コンテナアプリのログはSTDOUT/STDERRを利用
  - ✓ ローカルの開発環境でもデバッグが容易
  - ✓ ログの出力ストリームの送り先は実行環境側でコントロールすべき
- ✓ awslogs ログドライバを利用することでSTDOUT /STDERR がCloudWatch Logsに出力される

参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/using_awslogs.html

コンテナ定義: environment 、 secrets パラメータ

コンテナに環境変数を渡す方法は2つ

environment

- コンテナに環境変数として設定する
- 定義ファイルに平文で記述されるので機密情報には向かない

environmentの例

```
"environment" : [
{ "name" : "string", "value" : "string" },
{ "name" : ”hoge", "value" : ”huga" }
]
```

secrets

- 機密データを環境変数に設定する場合に利用
- AWS Secrets Manager シークレット、AWS Systems Manager パラメータストアのパラメータを参照可能
- それぞれタスク実行ロールに適切なIAM アクセス許可の設定が必要

secretsの例

```
"secrets": [
{ “name”: “environment_variable_name”,
"valueFrom":"arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-AbCdEf"
}]
```

参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/specifying-sensitive-data.html

コンテナでの機密情報の取り扱い

コンテナの環境変数(secrets)、ログ設定(logConfiguration のsecretOptions)では下記のいずれかに格納されたデータを利用可能

AWS Secrets Manager シークレット

- EC2 起動タイプではJSON キーやバージョンを指定可能(Fargate 起動タイプは未サポート)

シークレットを指定するAmazon リソースネーム (ARN)の構文

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

シークレット名　JSONキー　バージョンラベルかバージョンIDはどちらか片方のみ指定可能

AWS Systems Manager パラメータストア パラメータ

- 同じリージョンの場合は名前のみでも可だが別リージョンの場合、完全なARN が必要

パラメータを指定するAmazon リソースネーム (ARN)の構文

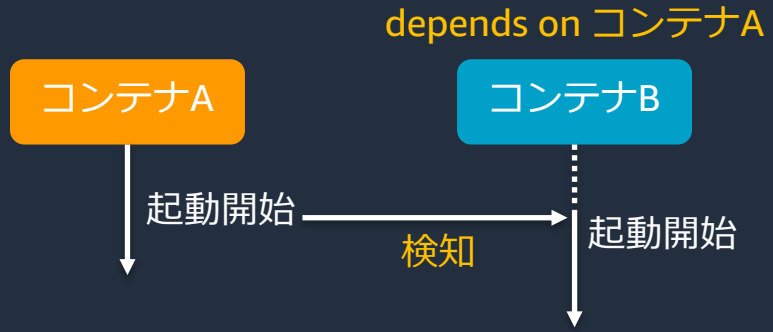
```
arn:aws:ssm:region:aws_account_id:parameter/parameter_name
```

パラメータ名

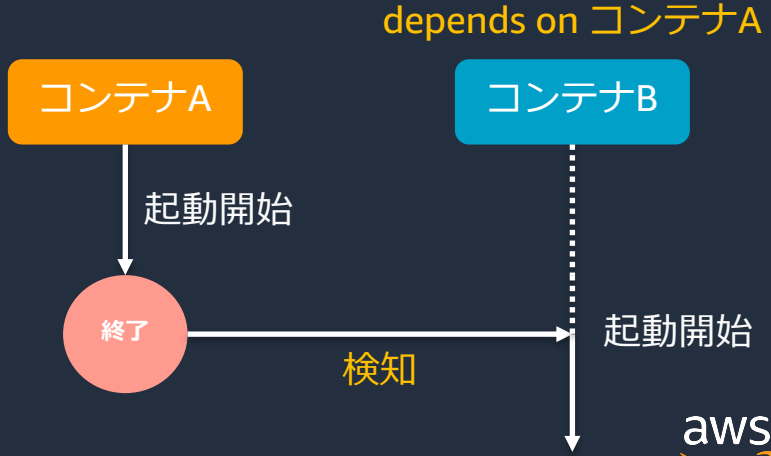
コンテナ定義: dependsOnパラメータ

- 事前処理を完了してからコンテナを起動したかったり、監視用のコンテナが正常に動いてから起動したいなどの実行順序の制御を行いたい時に利用
- タスク内のコンテナ間の依存関係を制御し、起動順や停止順を明示的に設定できる
- “condition ” に設定できる値→ START | COMPLETE | SUCCESS | HEALTHY

“condition” : “START” の例



“condition” : “COMPLETE” の例



タスク定義の詳細 – 代表的なパラメータ

コンテナが利用できる IAM ロールを指定する。
アプリケーションはこのIAMロールで許可されたAWS サービスのAPIを実行できる。

- コンテナ定義 (containerDefinitions)

オプション

- **タスクロール(taskRoleArn)**
- **タスク実行ロール(executionRoleArn)**
- ネットワークモード (networkMord)
- ボリューム (volumes)

ECS コンテナエージェントが利用する IAM ロールを指定する。
この権限を使ってコンテナのイメージをpull したりCloudWatch Logs に書き込んだりする 。

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```

Amazon ECS とIAM の連携整理

クラスターパーミッション

誰がクラスター内でタスクを起動/参照できるのかを制御

コンテナインスタンスロール

コンテナインスタンスの ECS のAPI へのアクセスを許可する

アプリケーション: タスクロール

アプリケーションのコンテナがAWSリソースに安全にアクセスすることを許可する

ハウスキーピング: タスク周りの下働きの実行を許可

タスク実行ロール

- プライベートレジストリのイメージを取得
- CloudWatch Logs への書き込み

ECS のサービスにリンクされたロール

- Elastic network interfaceの作成
- ELBへのターゲットの登録/解除

タスク定義の詳細 – 代表的なパラメータ

必須

- ファミリー (family)
- コンテナ定義 (containerDefinitions)

オプション

- タスクロール(taskRoleArn)
- タスク実行ロール(executionRoleArn)
- **ネットワークモード (networkMord)**
- ボリューム (volumes)
- 起動タイプ(requiresCompatibilities)

タスクのコンテナが使用するDocker ネットワークモードを指定する。
none | bridge | host | awsvpc のいずれかを設定。
(Windows コンテナの場合、指定しない)

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```

ネットワークモード

コンテナネットワーク動作を設定

noneモード

- 外部と接続しない

bridgeモード (EC2起動タイプのデフォルト)

- Dockerの組み込み仮想ネットワークを使用して外部ネットワークと通信(コンテナリンク利用可)

hostモード

- Dockerの組み込み仮想ネットワークをバイパスし、コンテナポートがホストEC2インスタンスのNICに直接マッピング

awsvpcモード

- ECS管理下のENIがタスクにアタッチされる
- セキュリティグループをENIに設定できる
- Fargate起動タイプでタスクを起動する場合の前提

awsvpc ネットワークモード

タスク 毎に ENI を自動割り当て

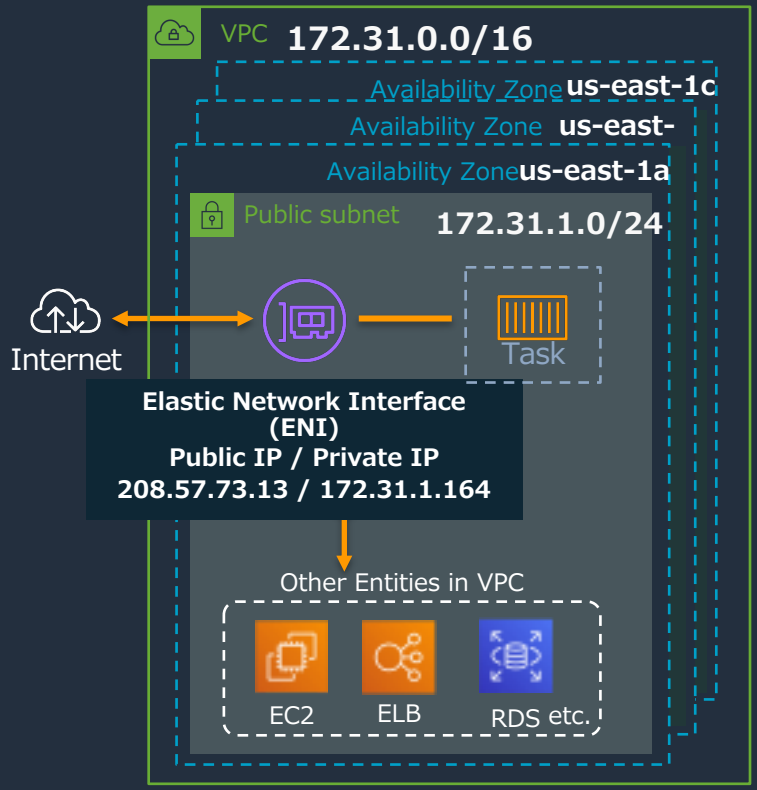
- セキュリティグループ を タスク 毎に設定可
- VPC Flow Logs でのモニタリング

タスク 内のコンテナは localhost インターフェースを共有

- Link 不要で互いにアクセス可能

VPC 内の他のリソースへ Private IP で通信が可能

- Fargate では Public IPの割当も可能



参考 : https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task-networking.html

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 トランキング

- ECSのアカウント設定でAWS VPC トランキング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

タスク定義の詳細 – 代表的なパラメータ

必須

- タスクロール(taskRoleArn)
- コンテナ定義(containerDefinitions)
- タスク実行ロール(executionRoleArn)
- ネットワークモード (networkMord)
- **ボリューム (volumes)**
- 起動タイプ(requiresCompatibilities)
- タスクサイズ（ cpu / memory)

データを共有したり、永続化するためのボリュームのリスト。
Docker ボリューム、バインドマウント、EFS ボリュームの3種類が利用可能。

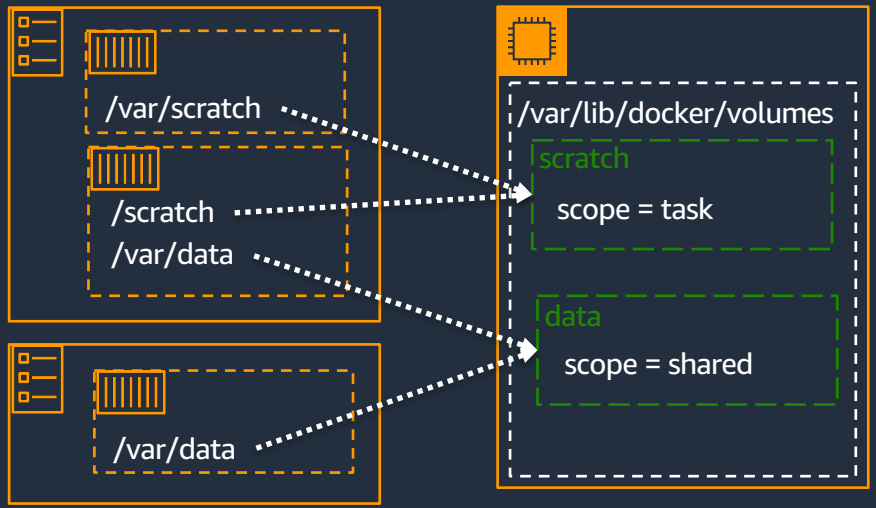
```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```

データボリュームの使用

https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/using_data_volumes.html

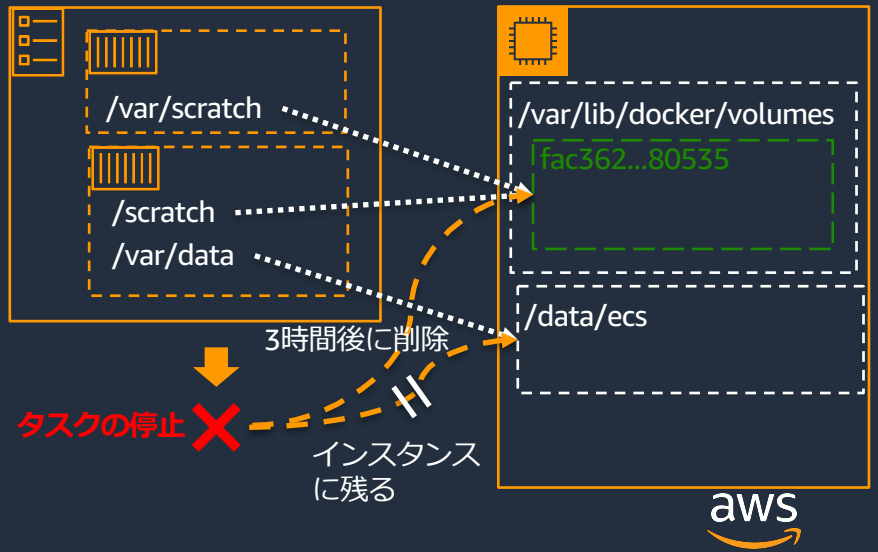
Docker ボリューム

- タスク間での共有や明示的なライフサイクル管理
- 3rd Party のボリュームドライバーの利用
- EC2 起動タイプのみ



バインドマウント

- ホストマシン上のファイル/ディレクトリをコンテナにマウント
- EC2 or Fargate 起動タイプ



Amazon Elastic File System (Amazon EFS) との連携

New!

Amazon EFS ボリューム

- Amazon EFS はスケーラブルな完全マネージド型のNFS ファイルシステム
- タスクはそれが配置されているコンテナインスタンスに関わらず、同じ永続的ストレージにアクセスできるようになる
- EC2 起動タイプの場合、ECSコンテナ エージェントバージョン 1.35.0 以降(EFS アクセスポイント/IAM 認証機能を使うには1.38.0 以降)、 Fargate 起動タイプの場合、プラットフォームバージョン 1.4 以降である必要がある

参考: https://aws.amazon.com/jp/blogs/news/amazon-ecs-supports-efs/

タスク定義の詳細 – 代表的なパラメータ

必須

- ファミリー (family)
- コンテナ定義 (containerDefinitions)

オプション

- タスクロール (taskRoleArn)
- 実行ロール (executionRoleArn)
- ネットワークモード (networkMord)
- ボリューム (volumes)
- 起動タイプ(requiresCompatibilities)
- **タスクサイズ（ cpu / memory)**

タスクに適用する、CPUとメモリのハード制限を設定する。Fargate 起動タイプでは必須かつ、決められた値を設定する必要がある。

```
{
"family": "",
"taskRoleArn": "",
"executionRoleArn": "",
"networkMode": "none",
"containerDefinitions": [...],
"volumes": [...],
"placementConstraints": [...],
"requiresCompatibilities": [...],
"cpu": "",
"memory": "",
"tags": [...],
"pidMode": "host",
"ipcMode": "host",
"proxyConfiguration": {..}
}
```

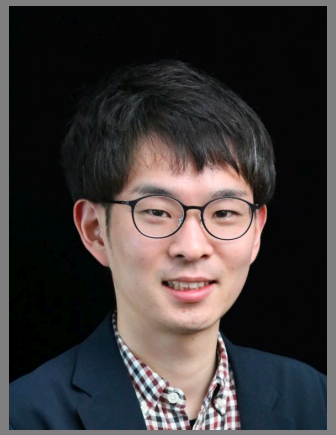

aws

より詳細な挙動や、ユースケースを知りたい方は

下記の「Amazon ECS Dive Deep」 のBlack Beltにてユースケースと詳細な挙動、設定について解説しておりますので、ご参照ください。

[AWS Black Belt Online Seminar]
Amazon ECS Deep Dive(2019/07/31)

動画:
https://youtu.be/3bohQetK2OE
資料:
https://d1.awsstatic.com/webinars/jp/pdf/services/20190731_AWS-BlackBelt_AmazonECS_DeepDive_Rev.pdf



Speaker 濵 真一

本日のアジェンダ

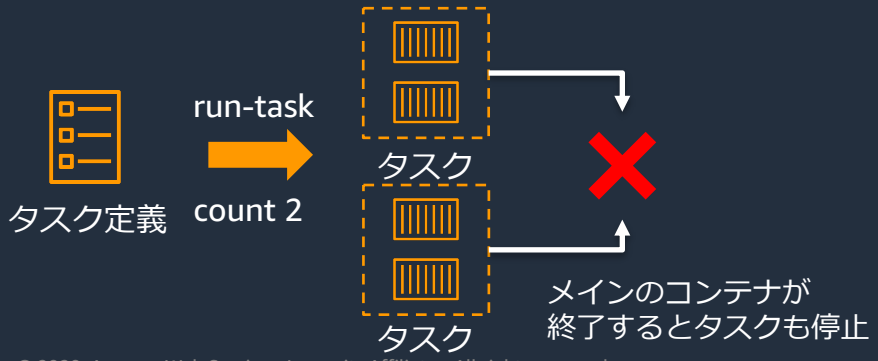
- **AWS のコンテナサービス概要**
- Amazon ECS の基本
  - Amazon ECS の主要要素
  - コンテナの実行環境
- Amazon ECS の機能紹介
  - タスク定義詳細
  - コンテナ実行

コンテナの実行方法

タスク定義からコンテナを実行する方法は2つ

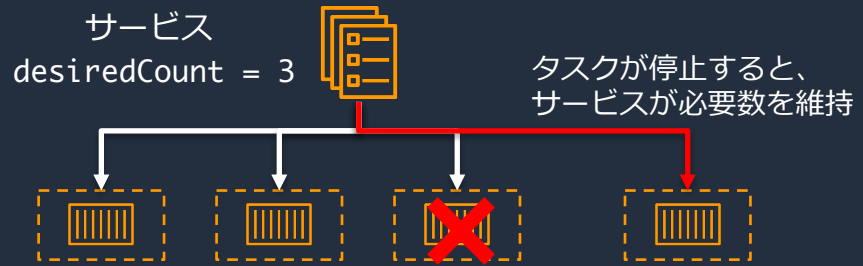
タスク

- タスク定義に従って実行されるアプリケーションの実行単位
- タスク定義の一部のパラメータはタスク実行時に上書き可能
- バッチジョブなど処理が終わると停止するワークロードなど



サービス

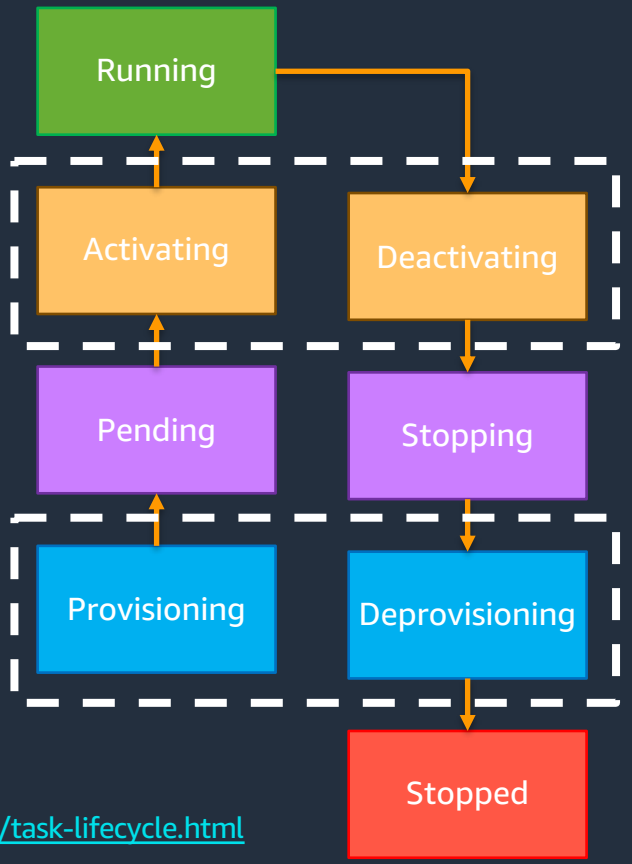
- 指定した数のタスクを維持する
- タスクが失敗/停止した場合は新しいタスクを起動して置き換え
- Web アプリケーションなど長時間実行するアプリケーションで利用
- ELB との連携や、Auto Scaling



タスクのライフサイクル

タスク定義を元にタスクが実行されると、
複数の状態を経由していく

Amazon ECS コンテナエージェントはタスクの変更のリクエストを目的のステータス(desiredStatus)と最後の既知のステータス(lastStatus)を使って追跡する



各ステータスの作業の例

Provisioning / Deprovisioning

- ENI のプロビジョン/デタッチなど

Activating / Deactivating

- ELB のターゲットグループの登録/解除など
  (サービスの一部の場合)

参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task-lifecycle.html

タスクの配置

タスク配置の制約事項

- タスク配置中に考慮されるルール
- AZ やインスタンスタイプに基づいて制約をかけられる

t3 インスタンスに配置する例

```
"placementConstraints": [ {
   "expression": "attribute:ecs.instance-type =~ t3.*",
   "type": "memberOf"
} ]
```

タスクの配置戦略

- タスク配置 / 終了時にインスタンスを選択するアルゴリズム
- binpack | random | spread の3つの配置戦略をサポート

※これらはEC2 起動タイプでのみ利用可能

タスク配置のプロセス

1. タスク定義で要求される要件を満たすインスタンスを識別
2. タスク配置の制約事項を満たすインスタンスを識別
3. タスク配置戦略を満たすインスタンスを識別
4. タスクを配置するインスタンスを選択

参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/task-placement.html

aws

サービススケジューラ戦略

サービスがタスクをスケジューリングする際の戦略は以下の2つ

レプリカ(REPLICA)

- クラスタ全体で必要数のタスクを維持
- デフォルトではAZ 間で分散される
- タスク配置戦略と制約によって配置をカスタマイズ可能
- 通常のアプリケーションではこちらを利用

デーモン(DAEMON)

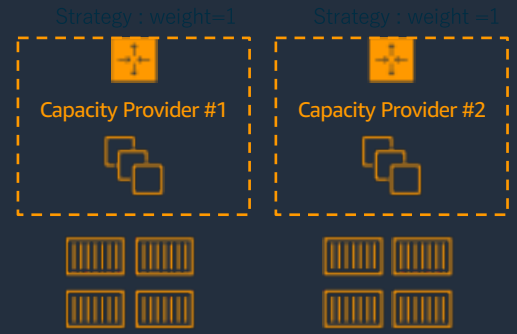
- (タスク配置の制約事項を満たす)コンテナインスタンス毎に1つのタスクをデプロイする
- ロギングやモニタリングなど各コンテナインスタンスで実行したい共通機能を実行する際に利用
- Fargate 起動タイプでは利用不可

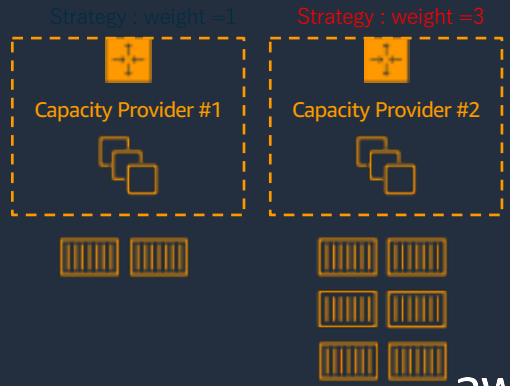
参考: https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/developerguide/ecs_services.html#service_scheduler

Amazon ECS クラスターキャパシティープロバイダー

New!

- タスク(コンテナ)の配置先を決定するための新しい方法
- タスク配置先の柔軟なコントロールが可能に
  - 例えば60%はオンデマンドで残りはスポット、等
- EC2 と Fargate の双方で利用可能
  - EC2 の場合は作成済みの EC2 ASG にキャパシティープロバイダー を紐付ける
  - Fargate 用 キャパシティープロバイダー は自動的に用意される (既存クラスターへは CLI で追加可能)
- **東京**を含む、ECS を利用可能な全てのリージョンで利用可能





https://aws.amazon.com/about-aws/whats-new/2019/12/amazon-ecs-capacity-providers-now-available/

サービスのAuto Scaling

Application Auto Scaling サービスを活用してサービスの必要タスク数(desiredCount)を自動的に増減させる

ターゲット追跡スケーリングポリシー

- 指定したメトリクスがターゲットの値に近づくように自動的に調整

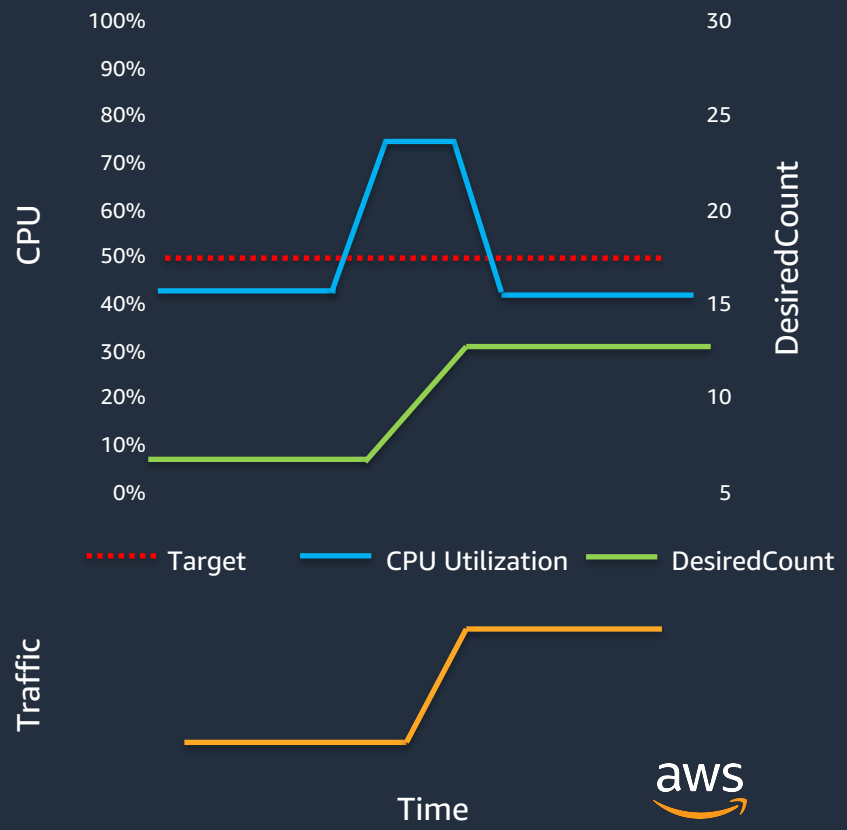
ステップスケーリングポリシー

- アラームをトリガーに調整値に基づいて増減
- ターゲット追跡スケーリングポリシーと組み合わせて高度なスケーリングも

スケジュールに基づくスケーリング

- 日付と時刻に基づいてタスク数を増減

ターゲット追跡スケーリングポリシーの動作イメージ

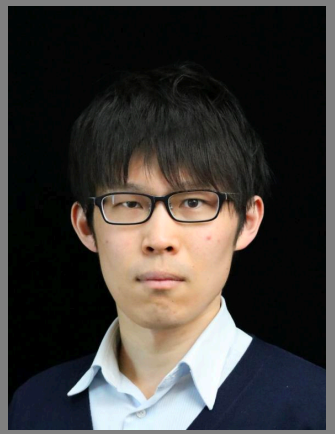


コンテナアプリケーションの監視について知りたい方は

コンテナで稼働しているアプリケーションやシステムの監視、モニタリング方法について学びたい方は下記もご参照ください

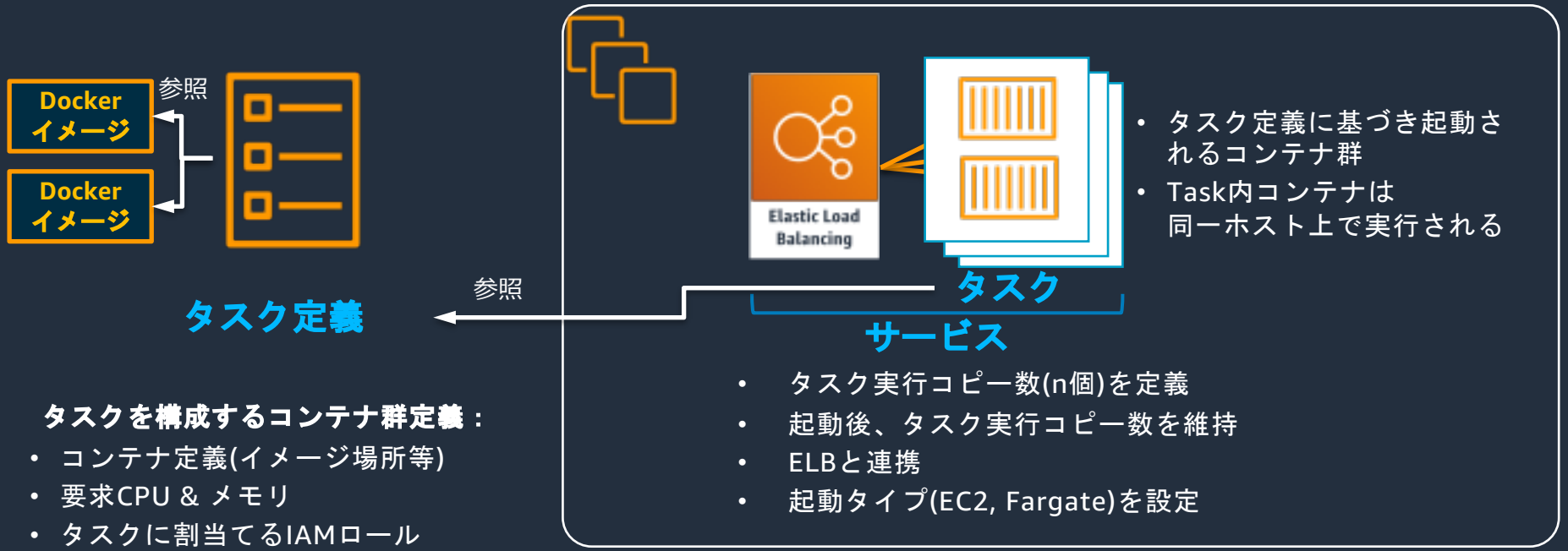
[AWS Black Belt Online Seminar]
Amazon CloudWatch Container Insights で始めるコンテナモニタリング入門(2019/11/27)

動画:
https://youtu.be/-w1nb99hxz8
資料:
https://d1.awsstatic.com/webinars/jp/pdf/services/20191127_AWS-BlackBelt_Container_Insights.pdf



Speaker 水馬 拓也

まとめ – Amazon ECSの主要要素 振り返り



- タスク定義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

タスクを構成するコンテナ群定義：

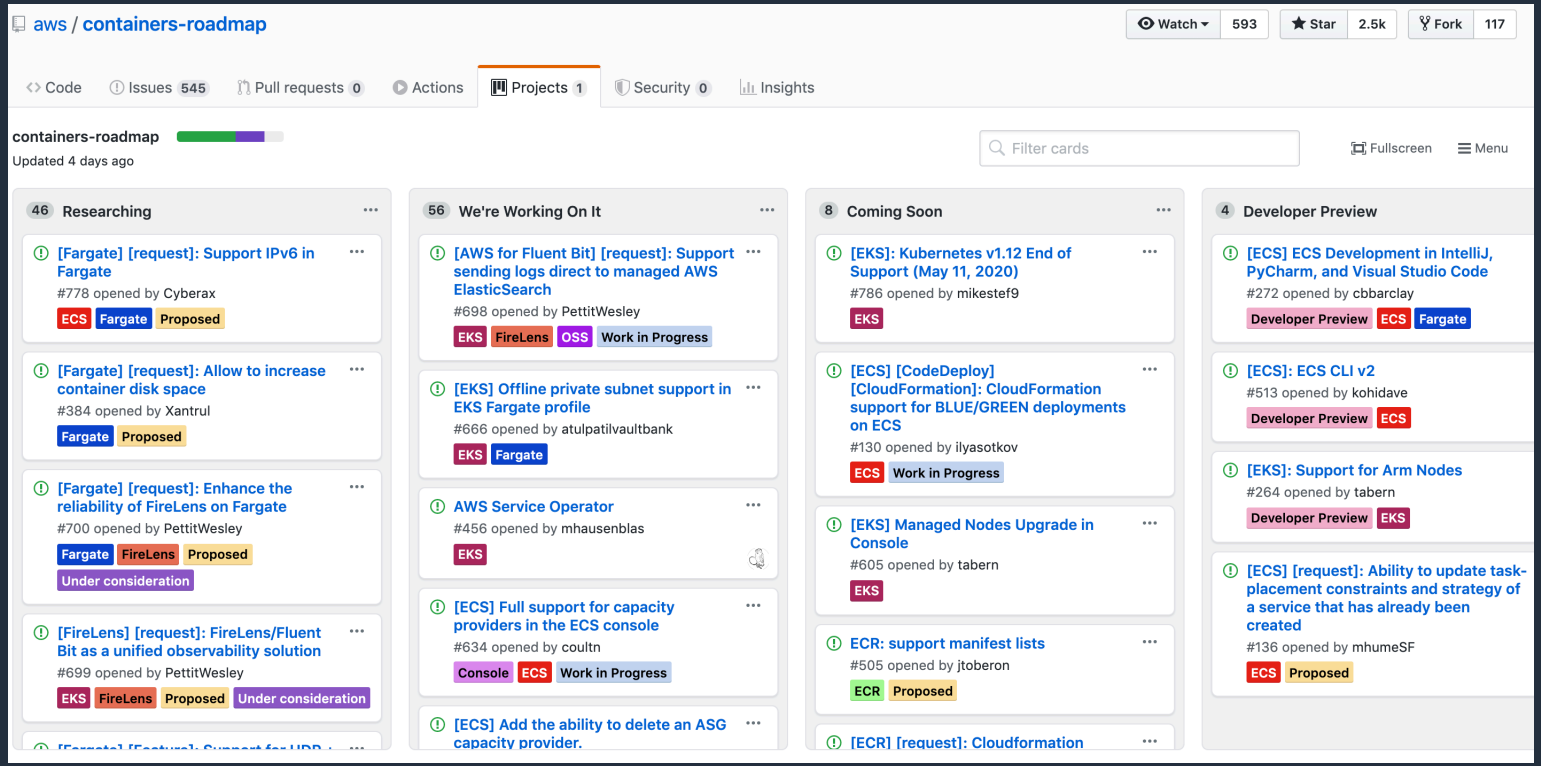
- コンテナ定義(イメージ場所等)
- 要求CPU & メモリ
- タスクに割当てるIAMロール
- ネットワークモード
- etc...

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

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

参考:AWS コンテナサービスの公開ロードマップ

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



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