Amazon Elastic Container Service
EC2 スポットインスタンス / Fargate Spot ことはじめ

AWS Black Belt Online Seminar

アマゾン ウェブ サービスジャパン株式会社
ソリューションアーキテクト 石本 遼

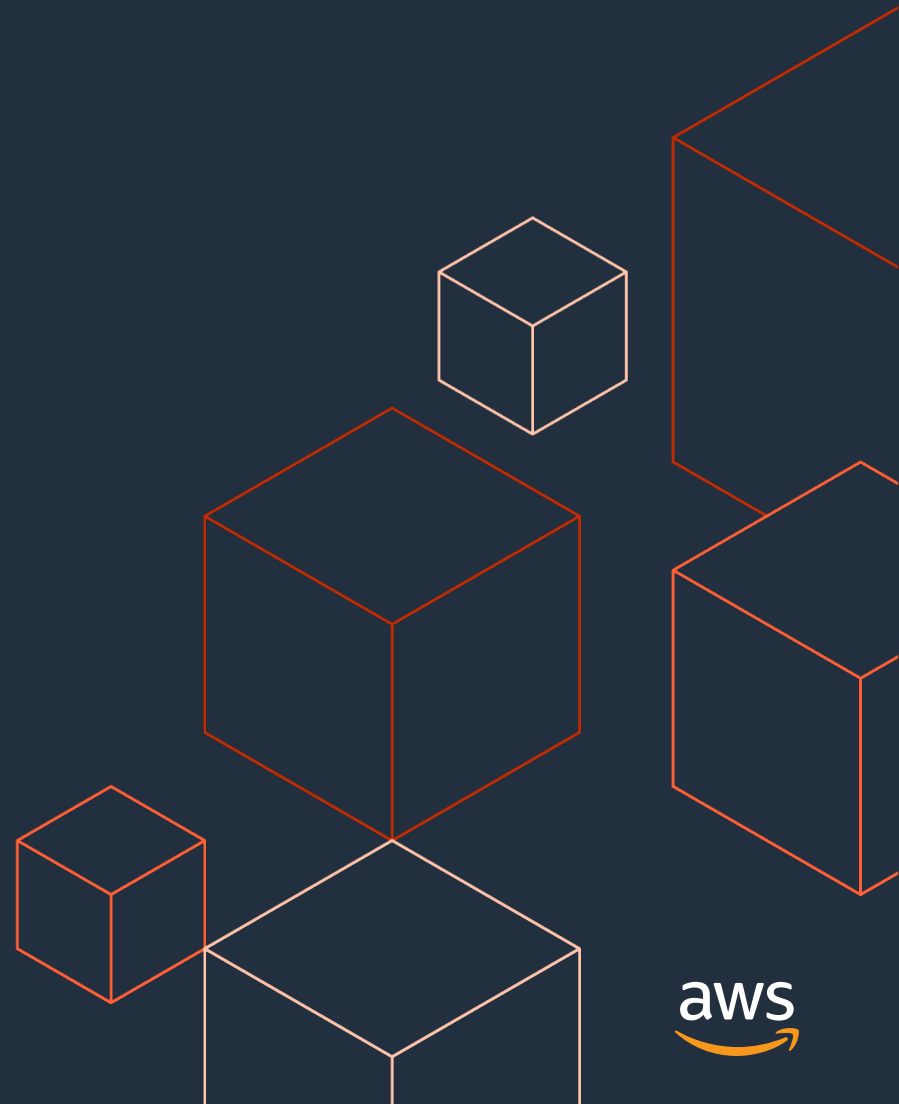
2021-Sep

本セッションの想定聴講者とゴール

- 想定聴講者
  - ECS を触ったことがある、開発 or 本番環境で利用している
  - ECS 上でのスポットインスタンス / Fargate Spot の活用を検討したいが、考慮事項の有無や、それらの詳細を理解していない
- ゴール
  - Amazon ECS Cluster Auto Scaling の構成要素、およびスポットリソースとの組み合わせ方について理解する
  - スポットリソース利用時に発生するイベントと、それらへの対策を理解する

aws

Amazon ECS における コスト最適化のポイント



AWS のコンテナサービス

オーケストレーション

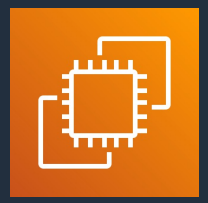
デプロイメント・スケジューリング
スケーリングとコンテナ
アプリケーションの管理

Amazon Elastic
Container Service
(Amazon ECS)

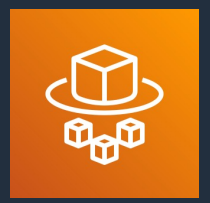
Amazon Elastic
Kubernetes Service
(Amazon EKS)

ホスティング

コンテナ実行環境



Amazon Elastic
Compute Cloud
(Amazon EC2)



AWS Fargate

イメージレジストリ

コンテナイメージリポジトリ

Amazon Elastic
Container Registry
(Amazon ECR)

アプリケーションネットワーキング

サービスディスカバリ、
サービスメッシュ

AWS App Mesh

AWS Cloud Map

AWS のコンテナサービス

オーケストレーション

デプロイメント・スケジューリング
スケーリングとコンテナ
アプリケーションの管理

コントロールプレーン
料金は発生しない

Amazon Elastic Container Service (Amazon ECS)

Amazon Elastic Kubernetes Service (Amazon EKS)

ホスティング

コンテナ実行環境

Amazon Elastic Compute Cloud (Amazon EC2)

AWS Fargate

データプレーンが
コスト最適化のポイント

イメージレジストリ

コンテナイメージリポジトリ

Amazon Elastic Container Registry (Amazon ECR)

アプリケーションネットワーキング

サービスディスカバリ、
サービスメッシュ

AWS App Mesh

AWS Cloud Map

aws

Amazon EC2 の購入オプション

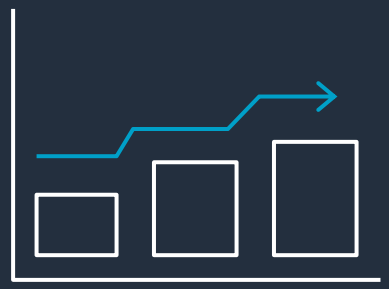
オンデマンド インスタンス

長期のコミット無し、使用分への支払い (秒単位/時間単位)。Amazon EC2 の定価

スパイクするようなワークロード

リザーブド インスタンス (RIs)

1年/3年の長期コミットに応じた大幅なディスカウント価格



一定の負荷の見通しがあるワークロード

Savings Plans

RI と同等のディスカウントに加え、さらなる柔軟性

ワークロードをまたがる柔軟性

スポット インスタンス

EC2 の空きキャパシティを活用し最大 90 % 引き。中断あり

中断に強く、様々なインスタンスタイプを活用できるワークロード

EC2 インスタンスとしての性能に違いはない

aws

AWS Fargate の購入オプション

- 柔軟な設定の選択肢 – 50 のCPU/メモリ設定から
- 通常の Fargate の価格と比較して **最大70%** の割引

* Savings Plan は最大 約 50% の割引 (3年コミット/全額一括前払い)、下限は 約 15 %

	Fargate	Savings Plan	Fargate Spot
	Price	Price	Price
1 vCPU	$0.05056	$0.03944 (22% Off)	**$0.01539 (70% Off)**
1 GB Mem	$0.00553	$0.00431 (22% Off)	**$0.00168 (70% Off)**

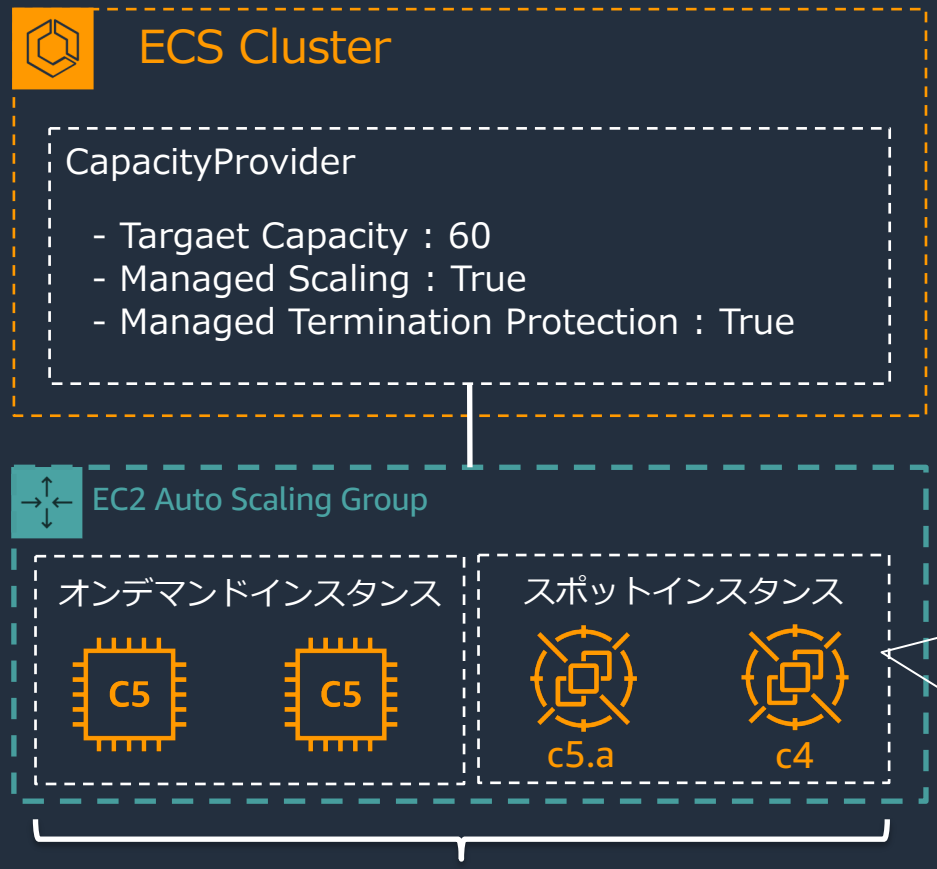
期間 1 年、全額前払い

※ 上記は東京リージョン 2021-08-03 時点での料金

AWS Fargate Pricing
https://aws.amazon.com/fargate/pricing/

Amazon ECS でのスポット利用パターン

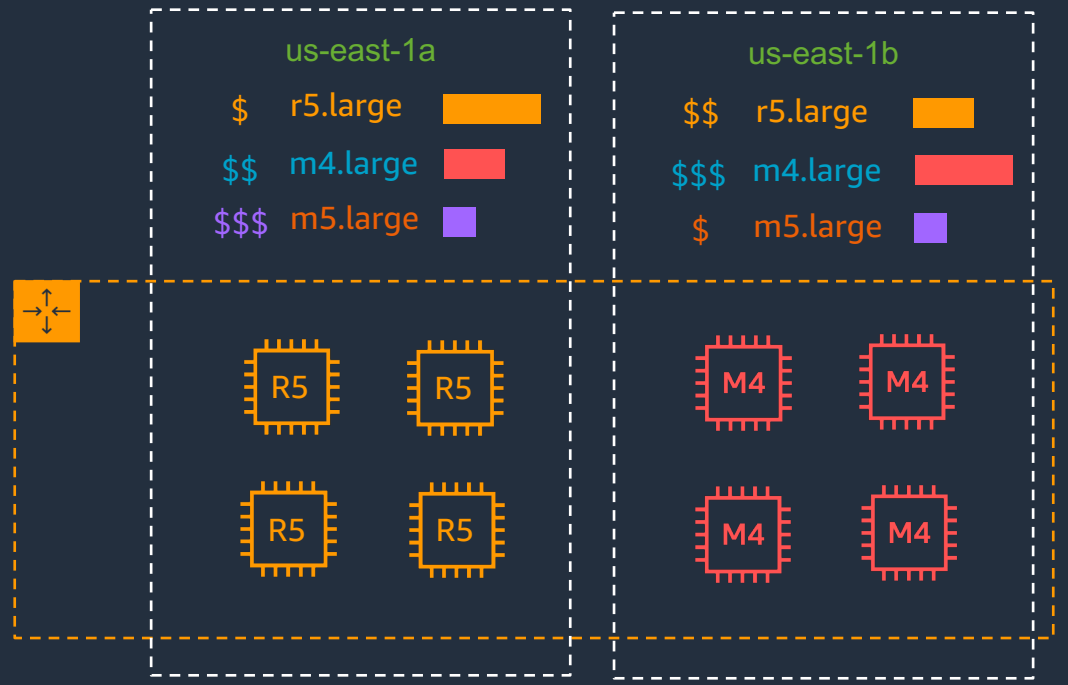
1 つの ASG 内で組み合わせるパターン



オンデマンドとスポットの
台数の割合を指定可能

スポットインスタンスの配分戦略には、
capacity-optimized が大変オススメです

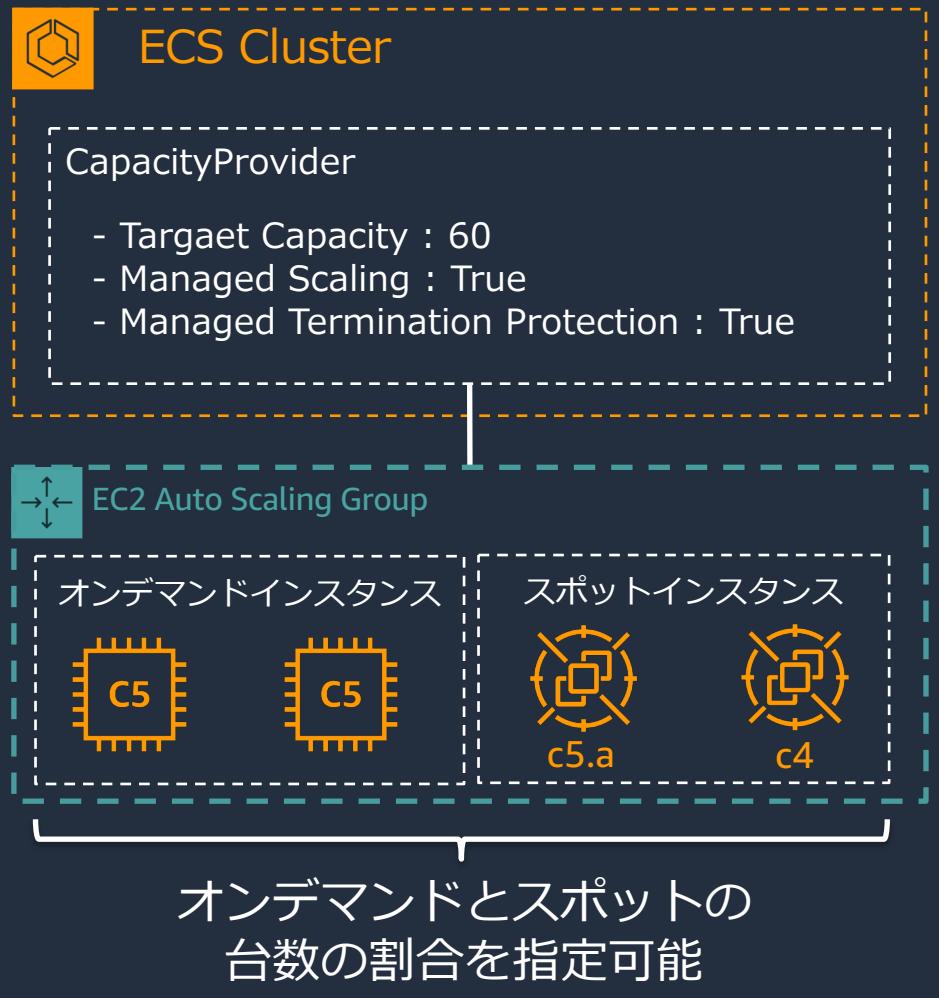
```
Overrides: ["r5.large", "m4.large", "m5.large"]
SpotAllocationStrategy: capacity-optimized
```



空きキャパシティの状況に応じて、
最適なスポットプールを自動的に選択

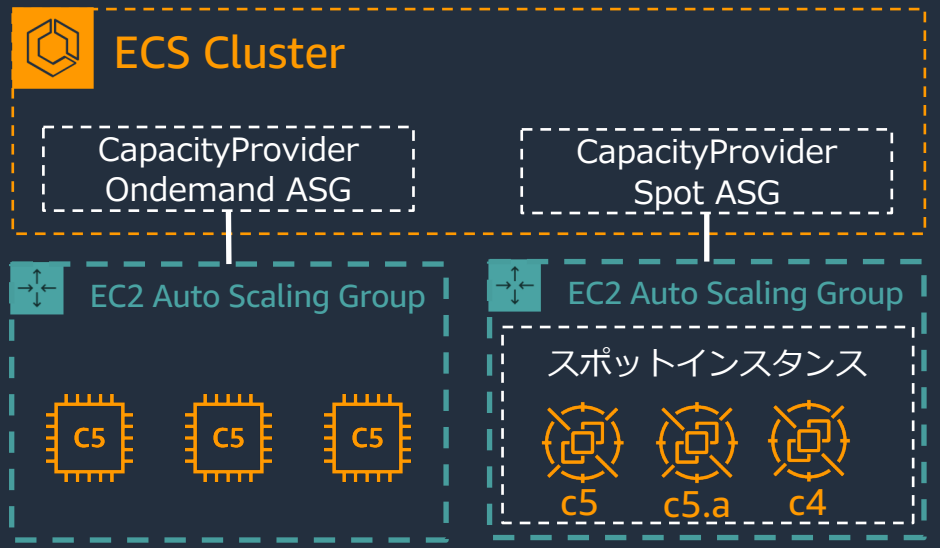
Amazon ECS でのスポット利用パターン

1 つの ASG 内で組み合わせるパターン

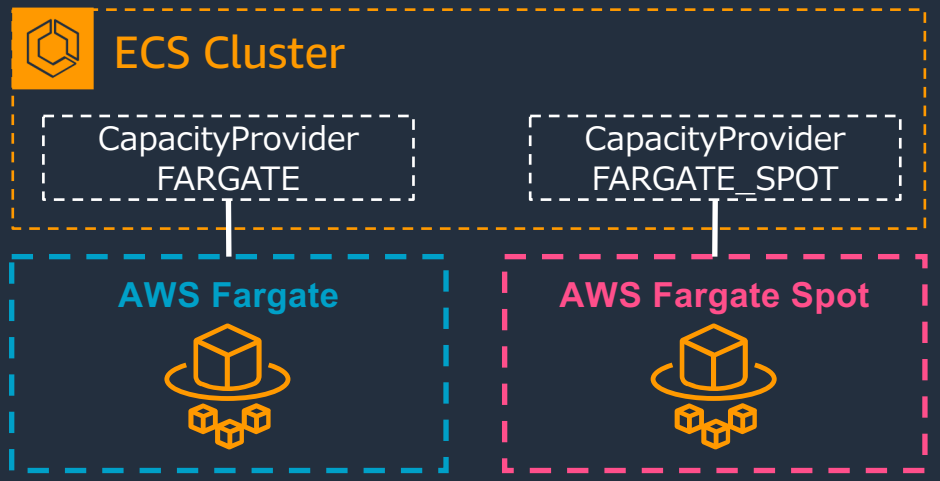


オンデマンドとスポットの
台数の割合を指定可能

オンデマンドとスポットを使い分けるパターン



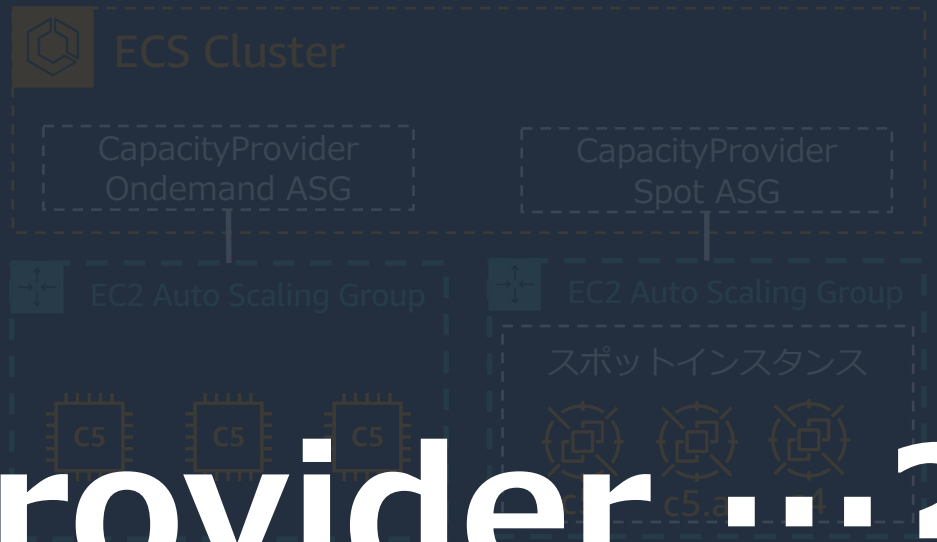
FARGATE でスポットを活用するパターン



Amazon ECS でのスポット利用パターン

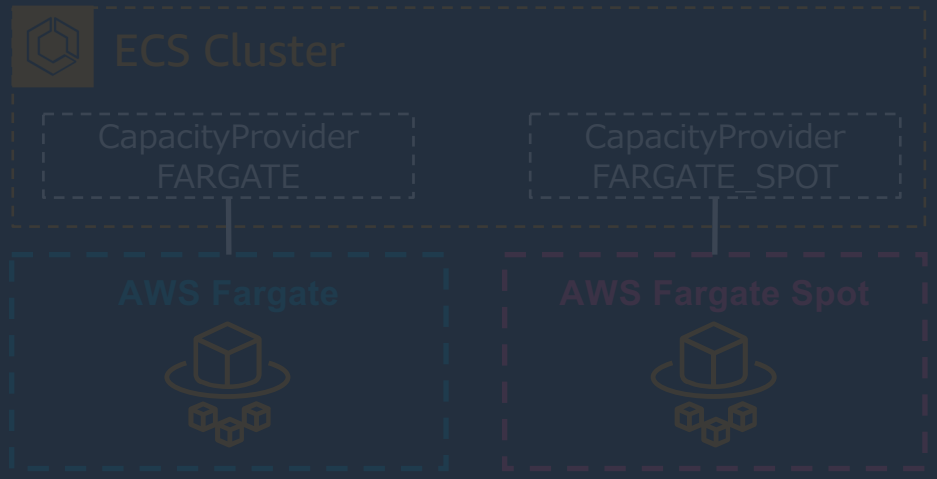
1 つの ASG 内で組み合わせるパターン

オンデマンドとスポットを使い分けるパターン



Capacity Provider …?

FARGATE でスポットを活用するパターン



スポット利用の前に知っておくべき ECS Cluster Auto Scaling

aws

ECS Cluster Auto Scaling が登場した背景

- 従来の課題
  - EC2 インスタンス, ECS タスクのスケーリングが独立している
- 上記の課題による影響
  - ECS タスクをスケールアウトする前に EC2 のスケーリングが必要
  - EC2 のスケールイン時、 ECS タスクが予期せぬ停止されることが

    AWS Lambda で “終了保護フラグ” を管理するなどの対応も…

https://aws.amazon.com/jp/blogs/news/deep-dive-on-amazon-ecs-cluster-auto-scaling/

aws

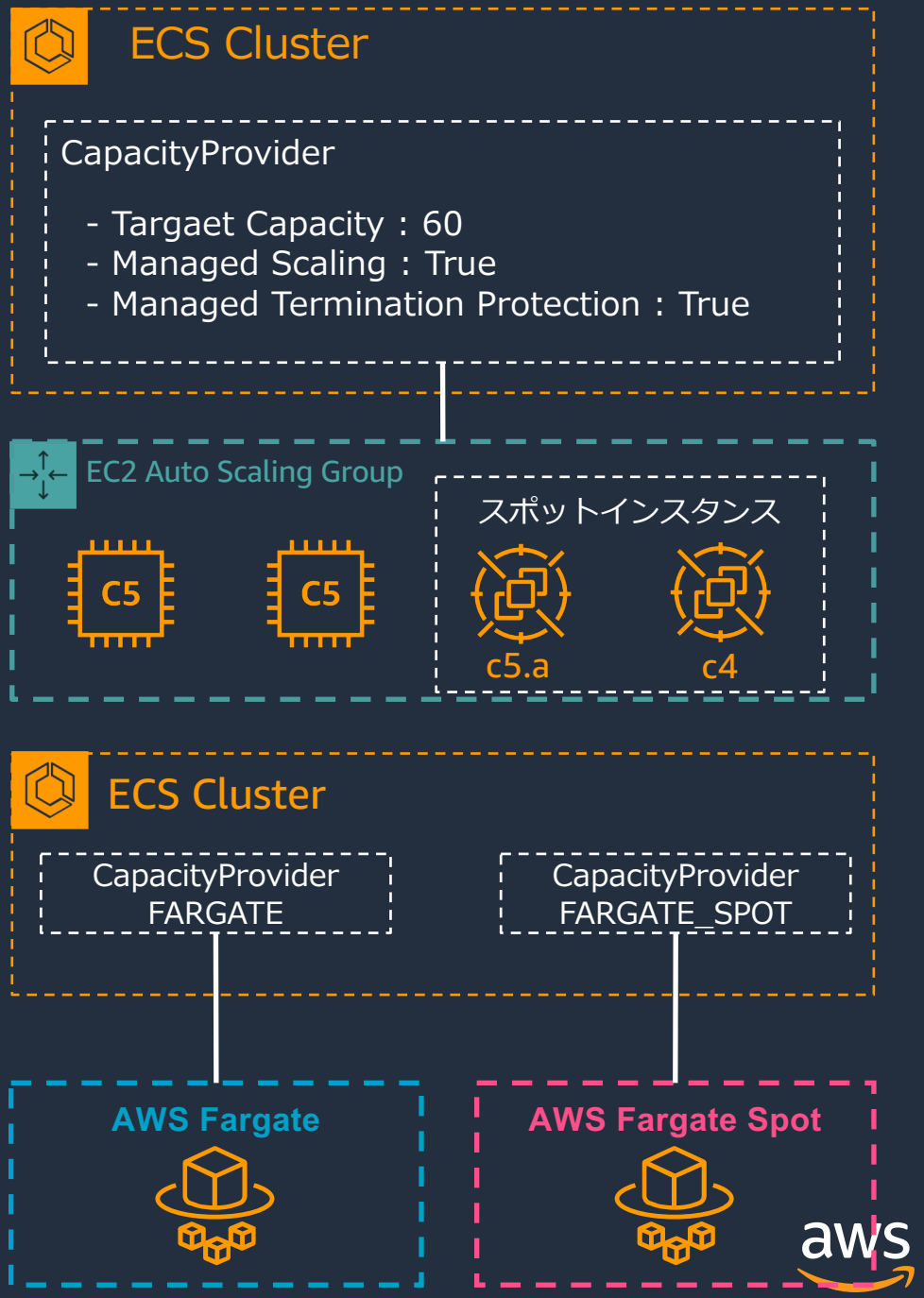
ECS Cluster Auto Scaling で登場するリソースや機能

- Capacity Provider
- Capacity Provider Strategy
- Capacity Provider Reservation
- Managed Scaling

aws

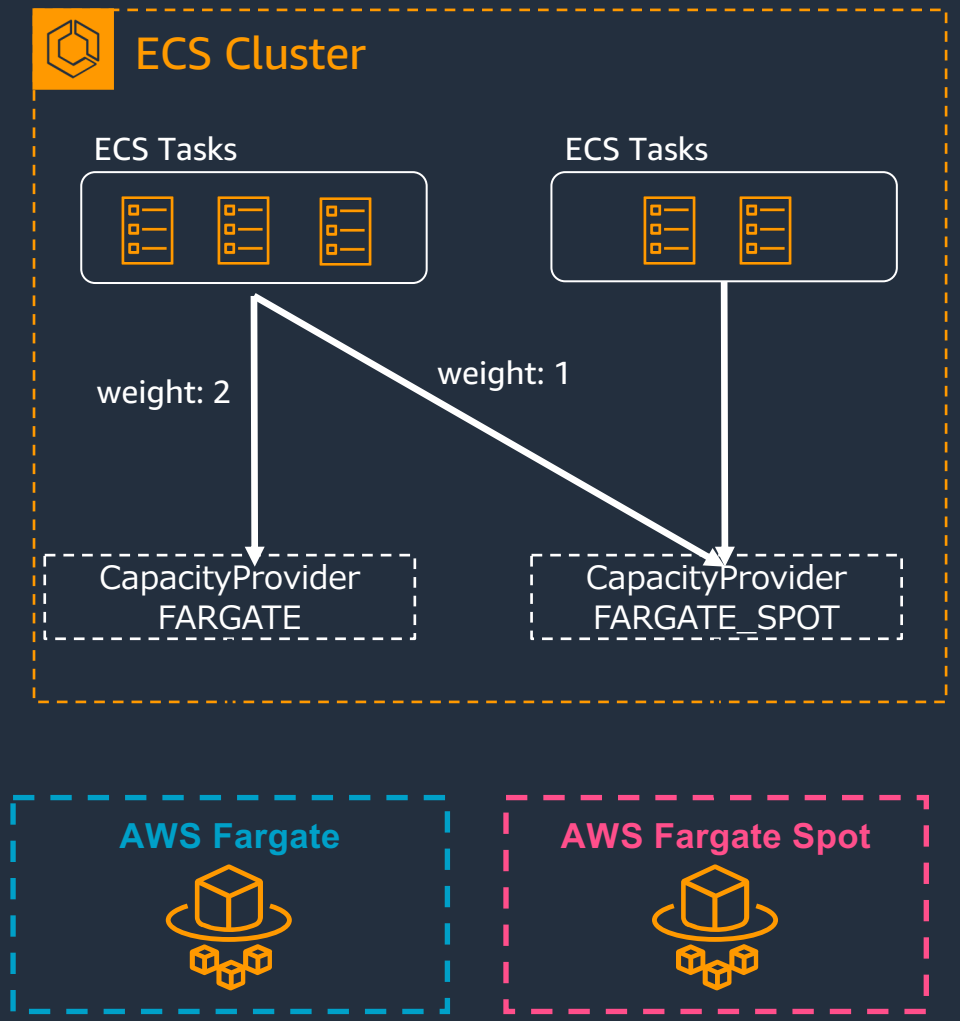
Capacity Provider

- ECS とコンテナ実行基盤（AutoScaling Group、Fargate）の間のインターフェースとしてリソースの抽象化を行う
- **AutoScaling Group** や **Fargate** を **1つの Capacity Provider と定義**して、クラスタに紐づけて利用
- Fargate においては、下記 2 つの Capacity Provider が事前に予約されており、 ECS クラスターに関連づけるだけで利用可能
  - FARGATE：いわゆるオンデマンド
  - FARGATE_SPOT：いわゆるスポット



Capacity Provider Strategy

- タスクをどの Capacity Provider に配置するかを決定する「戦略」
- Task 実行や Service を作成する際に指定する
- base と weight を使用して、複数の Capacity Provider にまたがって配置することも可能



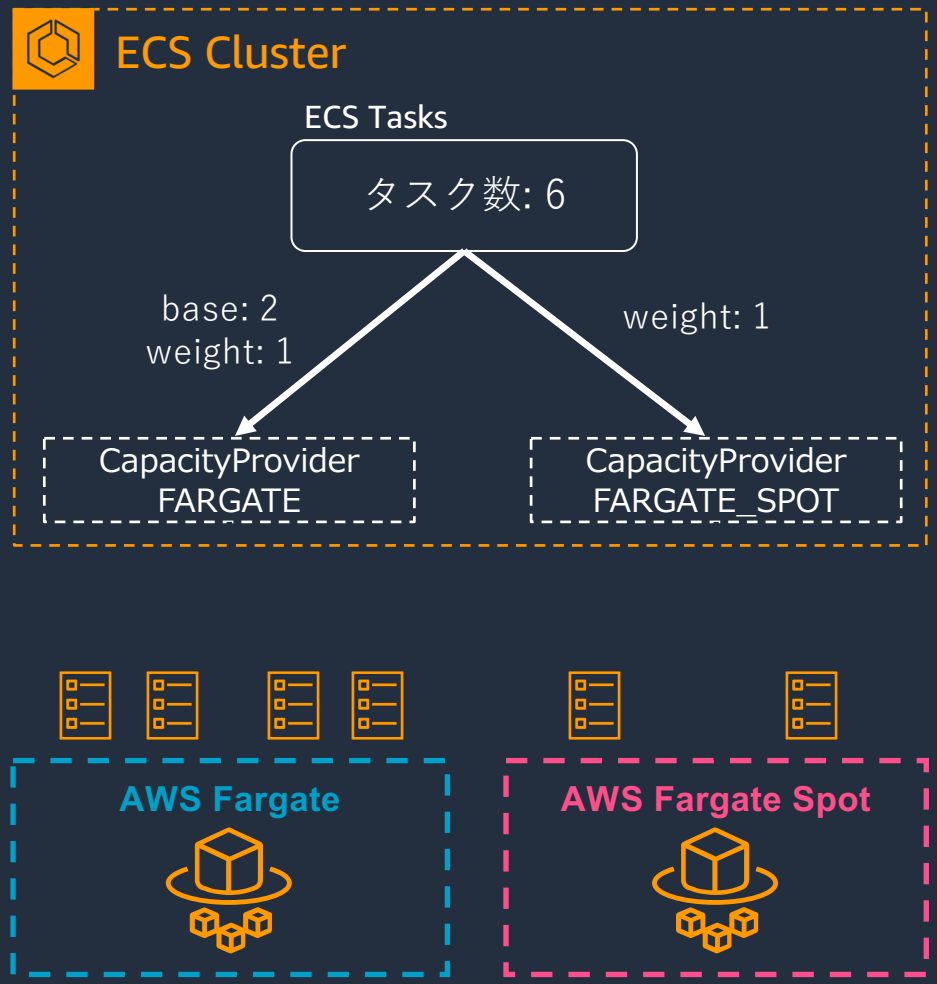
Default Capacity Provider Strategy

- ECS クラスタがデフォルトで利用する Capacity Provider Strategy
- Capacity Provider Strategy を指定せずに Task を実行した際に利用される

Capacity Provider Strategy (cont.)

- base：その Capacity Provider での最小実行タスク数
- weight：実行するタスクの総数に対する指定した Capacity Provider での相対的な割合

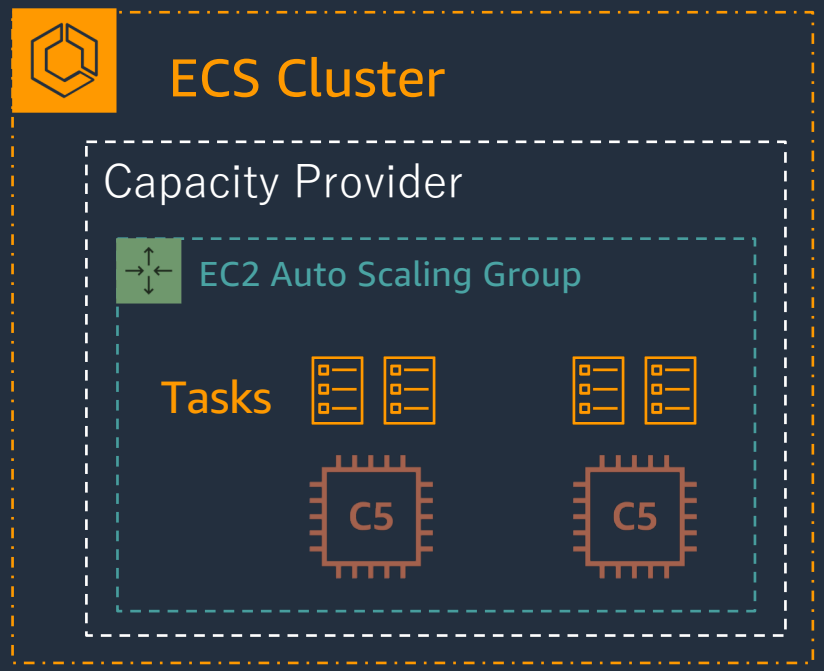
```
aws ecs create-service
  --cluster ${cluster_name}
  --service-name ${service_name}
  --task-definition ${task_def}
  --desired-count 6
 --capacity-provider-strategy
      capacityProvider=FARGATE,weight=1,base=2
      capacityProvider=FARGATE_SPOT,weight=1
```



Capacity Provider Reservation

Capacity Provider Reservation = M / N x 100 で導出される、
ECS が発行する CloudWatch Metrics

- M：**実行したいタスク** に必要な ASG 内のインスタンス数
- N：ASG 内で **既に実行されているインスタンス数**



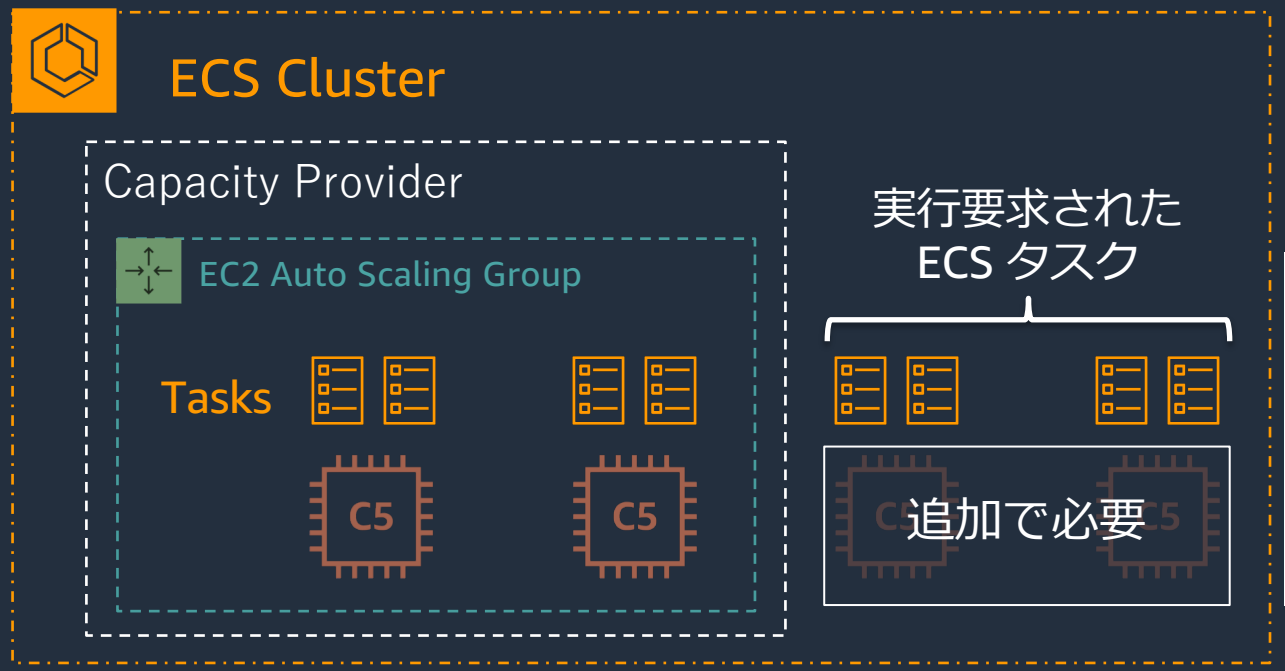
M = 4 つのタスクに必要な台数 = 2
N = 現在実行されている台数 = 2
Capacity Provider Reservation = 100 %

aws

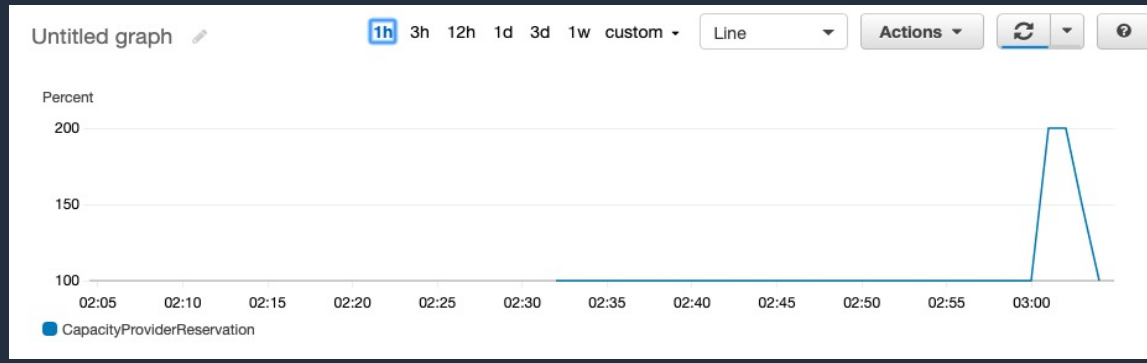
Capacity Provider Reservation (cont.)

Capacity Provider Reservation = M / N x 100 で導出される、
ECS が発行する CloudWatch Metrics

- M：**実行したいタスク** に必要な ASG 内のインスタンス数
- N：ASG 内で **既に実行されているインスタンス数**



M = 8 つのタスクに必要な台数 = 4
N = 現在実行されている台数 = 2
Capacity Provider Reservation = 200 %



キャパシティの過不足を可視化できる

aws

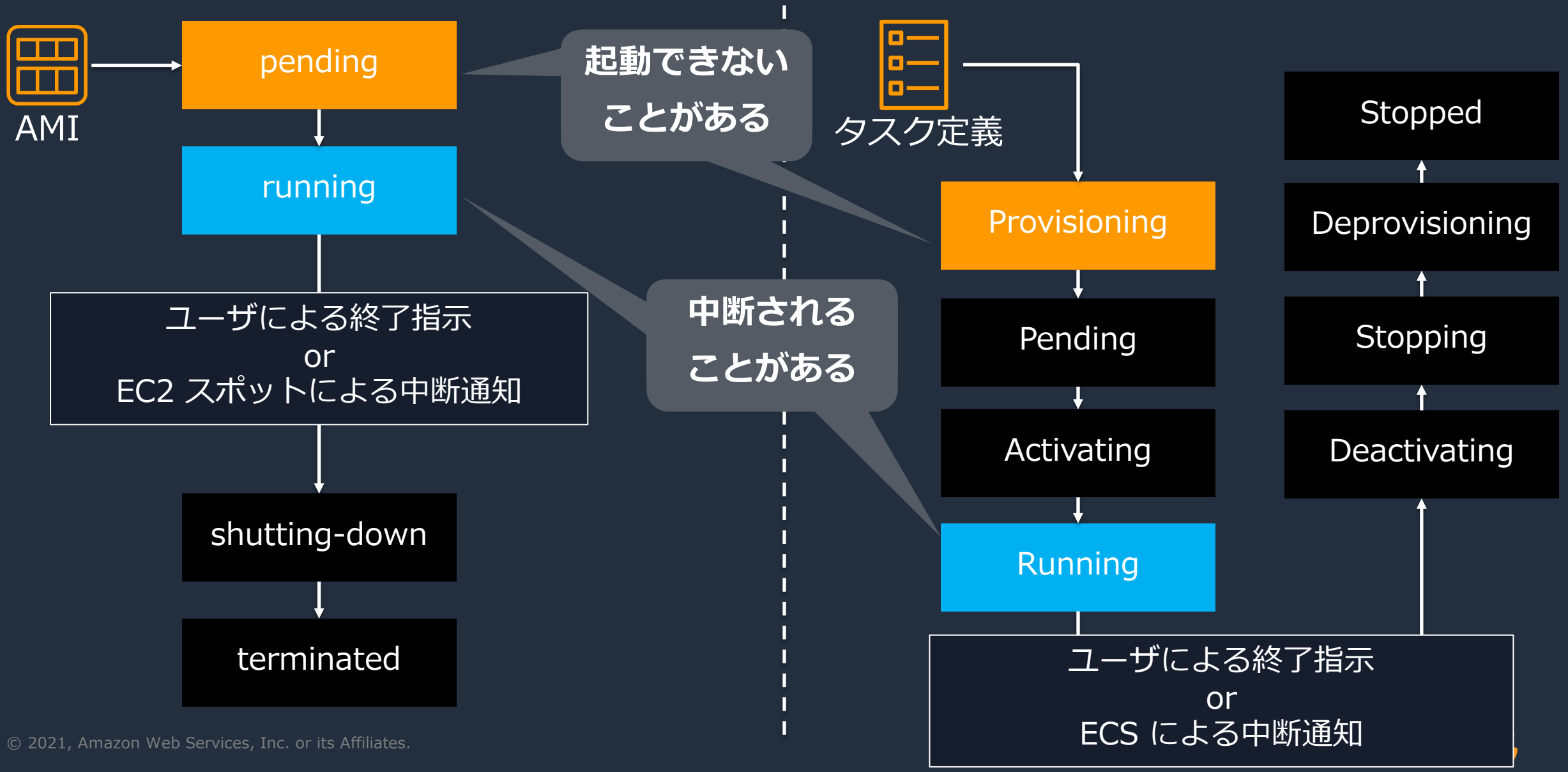
Managed Scaling

- Amazon ECS が、ユーザーに代わって Auto Scaling Group を管理
- Capacity Provider Reservation の値とターゲットキャパシティを評価し、必要に応じて ASG のスケーリングアクションを実行
- タスク実行時に配置先の EC2 が不足している場合、タスクを“PROVSIONING”ステートで待機させる

Amazon ECS
スポット利用において
注意するべき 2 つのポイント

aws

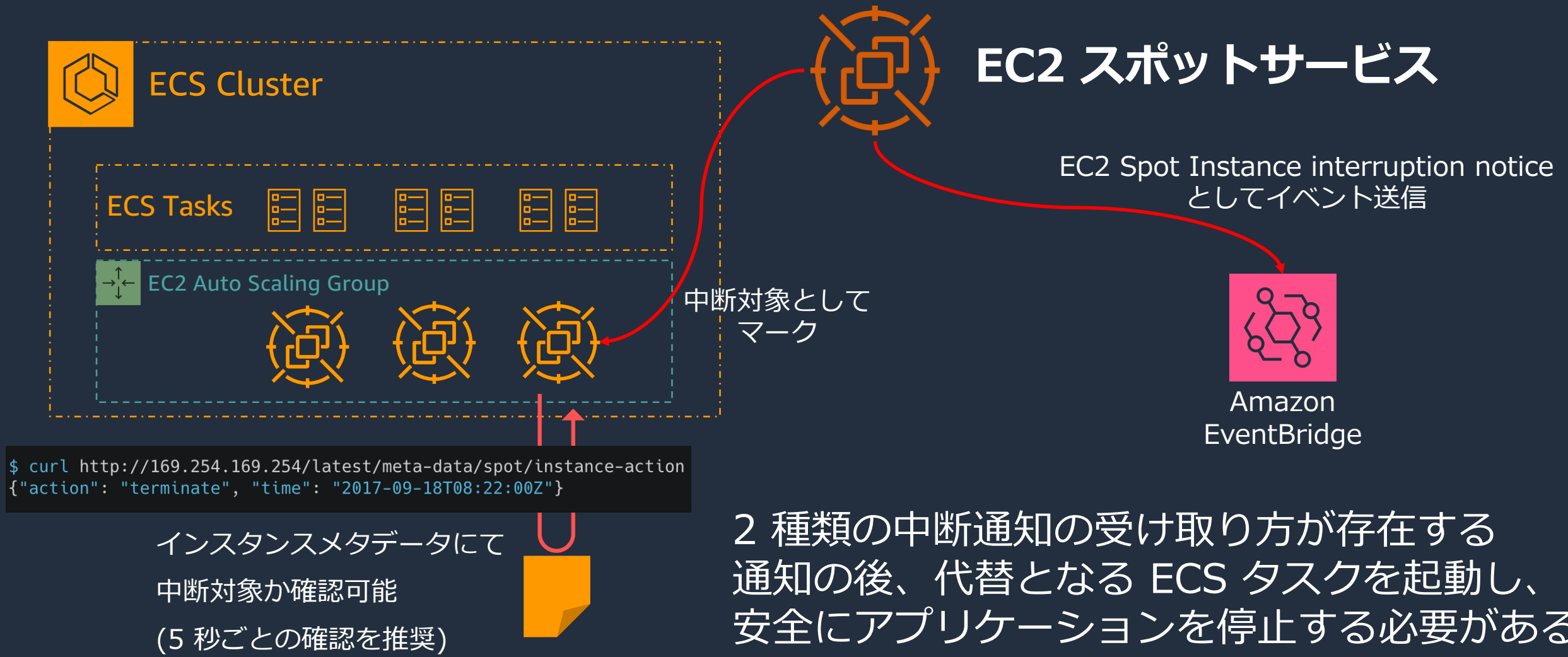
EC2 / ECS タスクのライフサイクルと スポット の関係



Amazon ECS
スポット利用において
注意するべき 2 つのポイント
- 中断への対応

aws

ECS on スポットインスタンス での中断通知のハンドリング



2 種類の中断通知の受け取り方が存在する
通知の後、代替となる ECS タスクを起動し、
安全にアプリケーションを停止する必要がある

ECS on スポットインスタンス での中断通知のハンドリング

- **中断通知発生時に対応すべきこと**
  - 新規タスクの配置停止
  - 既存タスクのドレイニング & LB切り離し & 停止 & ログ退避など
  - 別コンテナインスタンスでのサービスタスク実行

- **中断発生の対応に向けまず設定しておくべきこと**

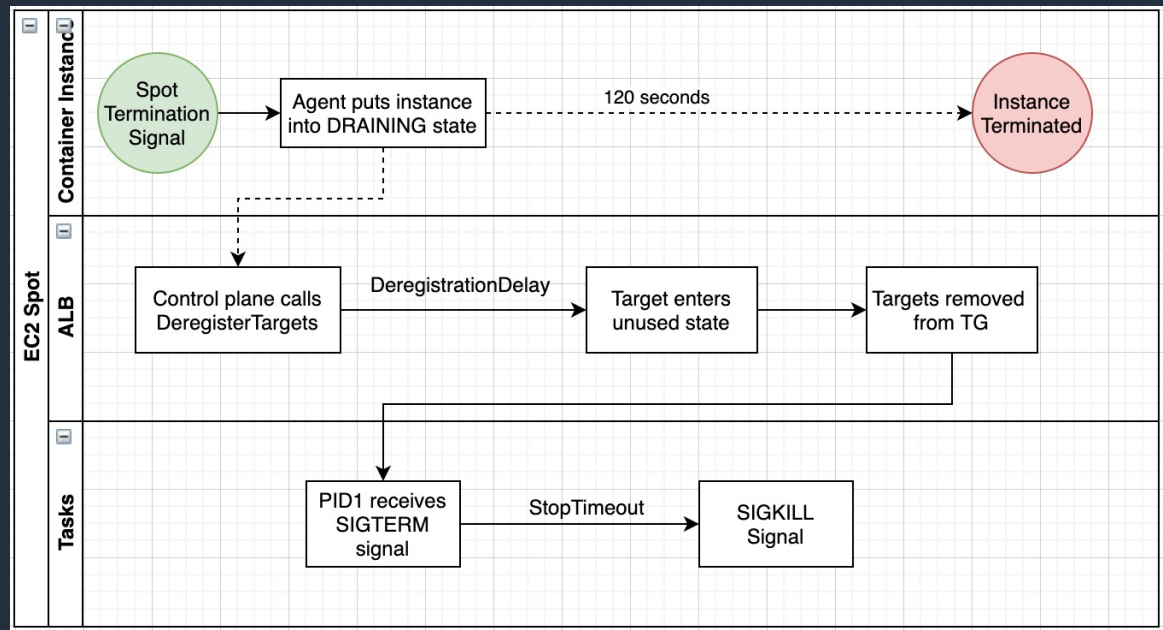
```
#!/bin/bash
cat <<'EOF' >> /etc/ecs/ecs.config
ECS_CLUSTER=MyCluster
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true
EOF
```

ECS コンテナインスタンスの Userdata などで、
ECS_ENABLE_SPOT_INSTANCE_DRAINING=true を
/etc/ecs/ecs.config に書き込んでおく

aws

ECS on スポットインスタンス での中断通知のハンドリング

- **ECS_ENABLE_SPOT_INSTANCE_DRAINING=true の挙動**
  - 中断の2分前にスポット中断通知 & ECS エージェントが検知
  - ECS エージェントが DRAINING をリクエスト、新規タスク配置停止
  - ECS が ACTIVE な別インスタンスに、代替のサービスタスクを実行
  - ECS が ALB / 該当タスクに対して DeregisterTargets API を実行 DeregistrationDelay を経て、 TargetGroup から Remove
  - 中断対象タスクに SIGTERM が送信
    - ここで必要な後処理を実行
  - stopTimeout 後に SIGKILL が送信



■参考情報
What's new：Amazon ECS が ECS サービスを実行しているスポットインスタンスの自動ドレインをサポート
AWS Blog：ECS のアプリケーションを正常にシャットダウンする方法

ECS on スポットインスタンス での中断通知のハンドリング

- **SIGTERM 発生時に ECS タスク内で対応すべきこと**
  - アプリケーションの Graceful Shutdown
    (新規受付停止/仕掛り中処理の完了)
  - バッファしているログの出力・退避
  - ロングラン処理の中断・チェックポイント書き出し

```python
import signal, time, os

def shutdown(signum, frame):
    print('Caught SIGTERM, shutting down')
    # Finish any outstanding requests, then...
    exit(0)

if __name__ == '__main__':
    # Register handler
    signal.signal(signal.SIGTERM, shutdown)
    # Main logic goes here
```

```node
process.on('SIGTERM', () => {
  console.log('The service is about to shut down!');

  // Finish any outstanding requests, then...
  process.exit(0);
});
```


aws

ECS on Fargate Spot での中断通知のハンドリング

Service

Fargate Spot

Task
Container
Container

Task
Container
Container

Task
Container
Container

Task
Container
Container

SIGTERM
シグナル

ECS コントロールプレーン

ECS Task State Change
としてイベント送信

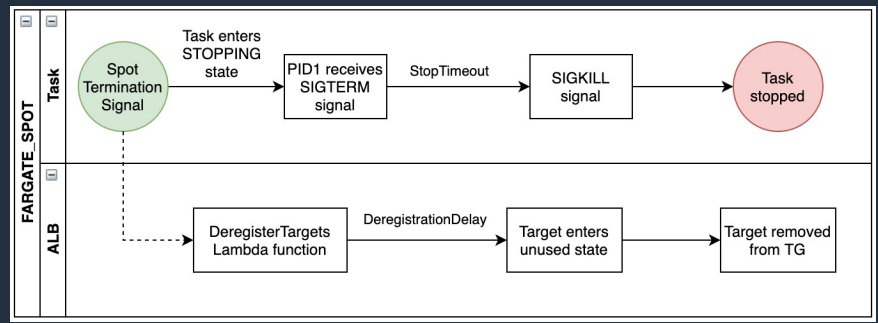
DeregisterTarget
(実行が保証されない)

Amazon
EventBridge

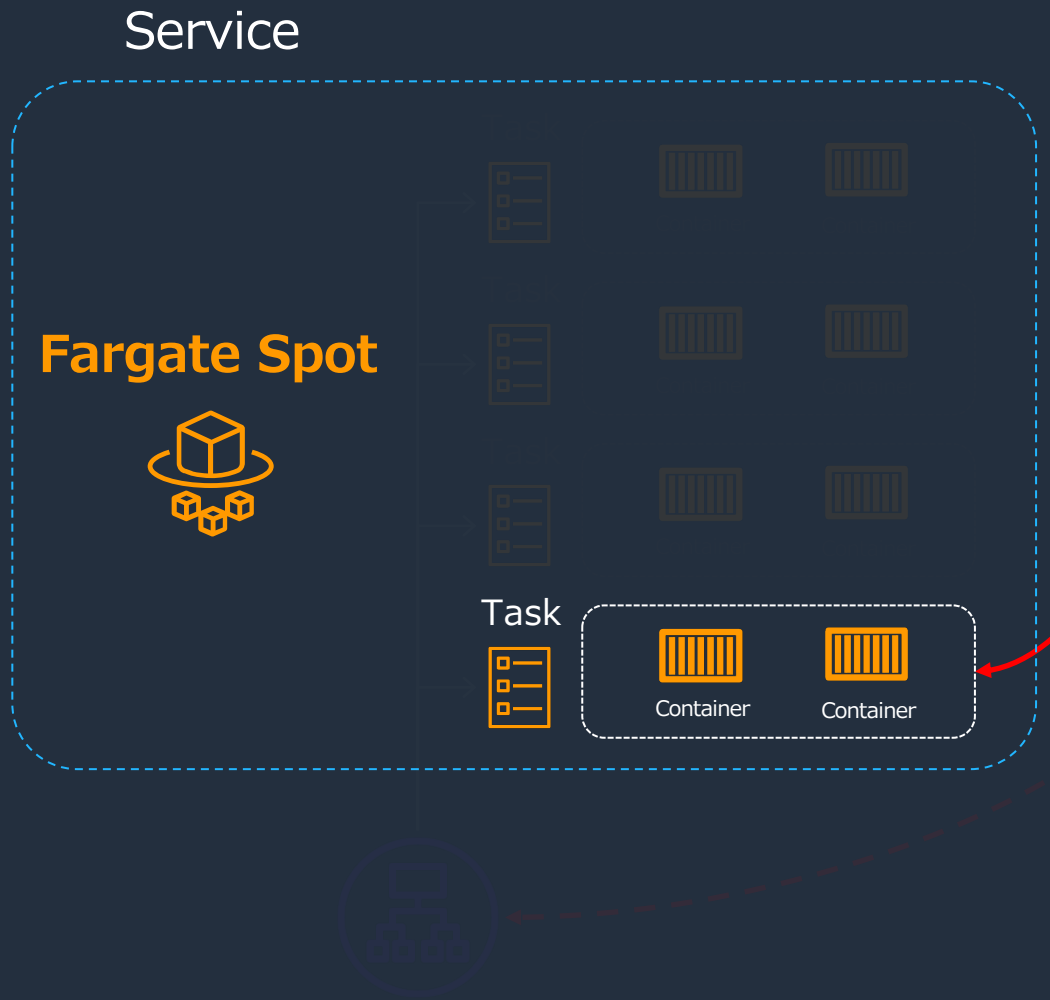
Application Load Balancer

AWS にキャパシティが必要になったとき、
Fargate Spot で稼働するタスクは
2 分前の通知とともに中断される

Graceful shutdowns with ECS
While Automated Spot Instance Draining works on EC2 Spot Instances, tasks that are run as **FARGATE_SPOT are not guaranteed to be deregistered** from a load balancer's target group until the task transitions to a STOPPED state.



ECS on Fargate Spot での中断通知のハンドリング



SIGTERM
シグナル

ECS コントロールプレーン

SIGTERM を受け取ったタスクで必要となる処理(*)

- ALB からの Deregistration

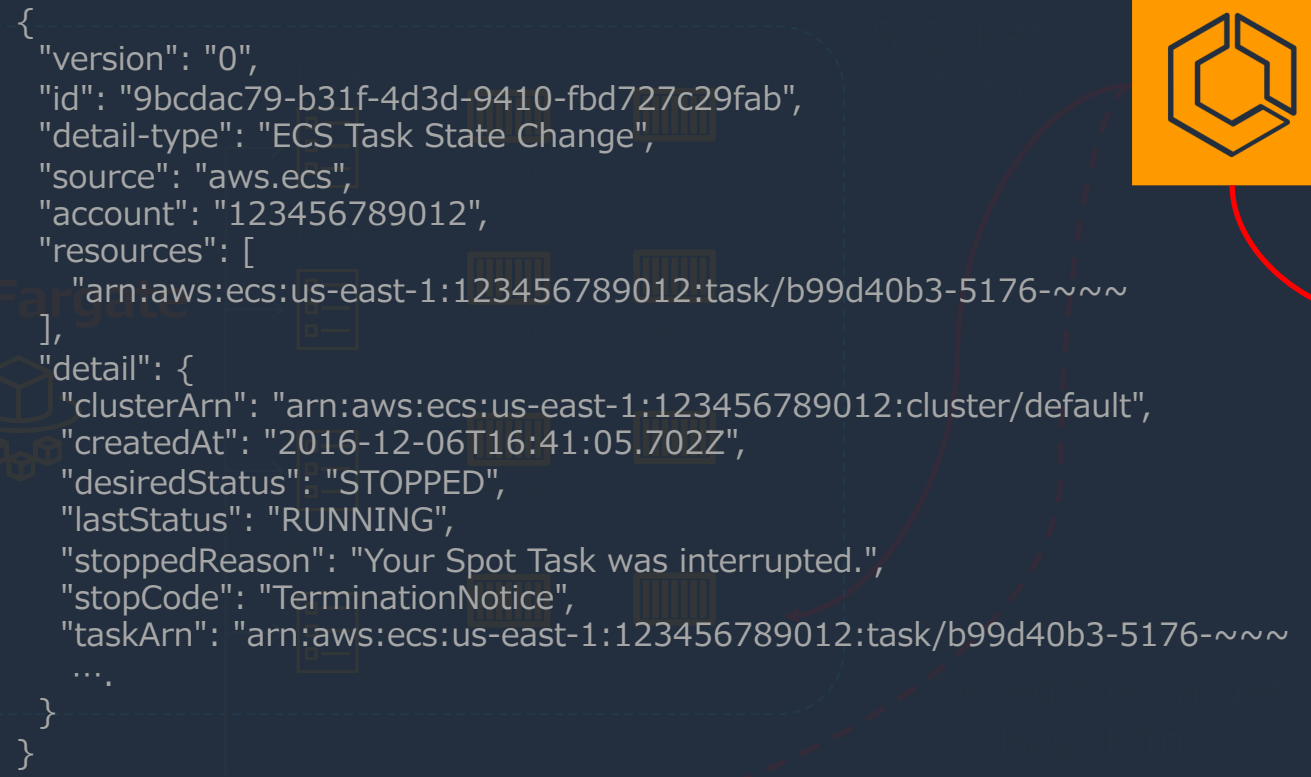
- アプリケーションの Graceful Shutdown

- ロングラン処理の中断・チェックポイント書出し

etc…

* デフォルトで SIGTERM の 30 秒後に SIGKILL が発行、
タスク定義内の stopTimeout の設定によって、
120 秒などに設定することが可能

ECS on Fargate Spot での中断通知のハンドリング



ECS コントロールプレーン

ECS Task State Change
としてイベント送信

Amazon
EventBridge

DeregisterTarget

Application Load Balancer

https://github.com/aws-samples/ecs-fargate-drain-function

Amazon ECS
スポット利用において
注意するべき 2 つのポイント
- リソースの再確保

aws

スポット中断後のリソース再確保

Amazon ECS ベストプラクティスガイド
Amazon EC2 スポットと FARGATE_SPOT の使用
https://docs.aws.amazon.com/ja_jp/AmazonECS/latest/bestpracticesguide/ec2-and-fargate-spot.html

- EC2 スポットインスタンス
  - Auto Scaling Group で desired capacity > instance count となるため
    ASG 自身が、設定されている配分戦略に基づいて不足分の再確保を試みる
- ECS タスク / Fargate Spot
  - ECS スタンドアロンタスク の場合、実行要求元にてリトライが必要
  - ECS サービス の場合、desired count > running count となるため
    ECS サービス (コントロールプレーン) がタスク起動を再試行する

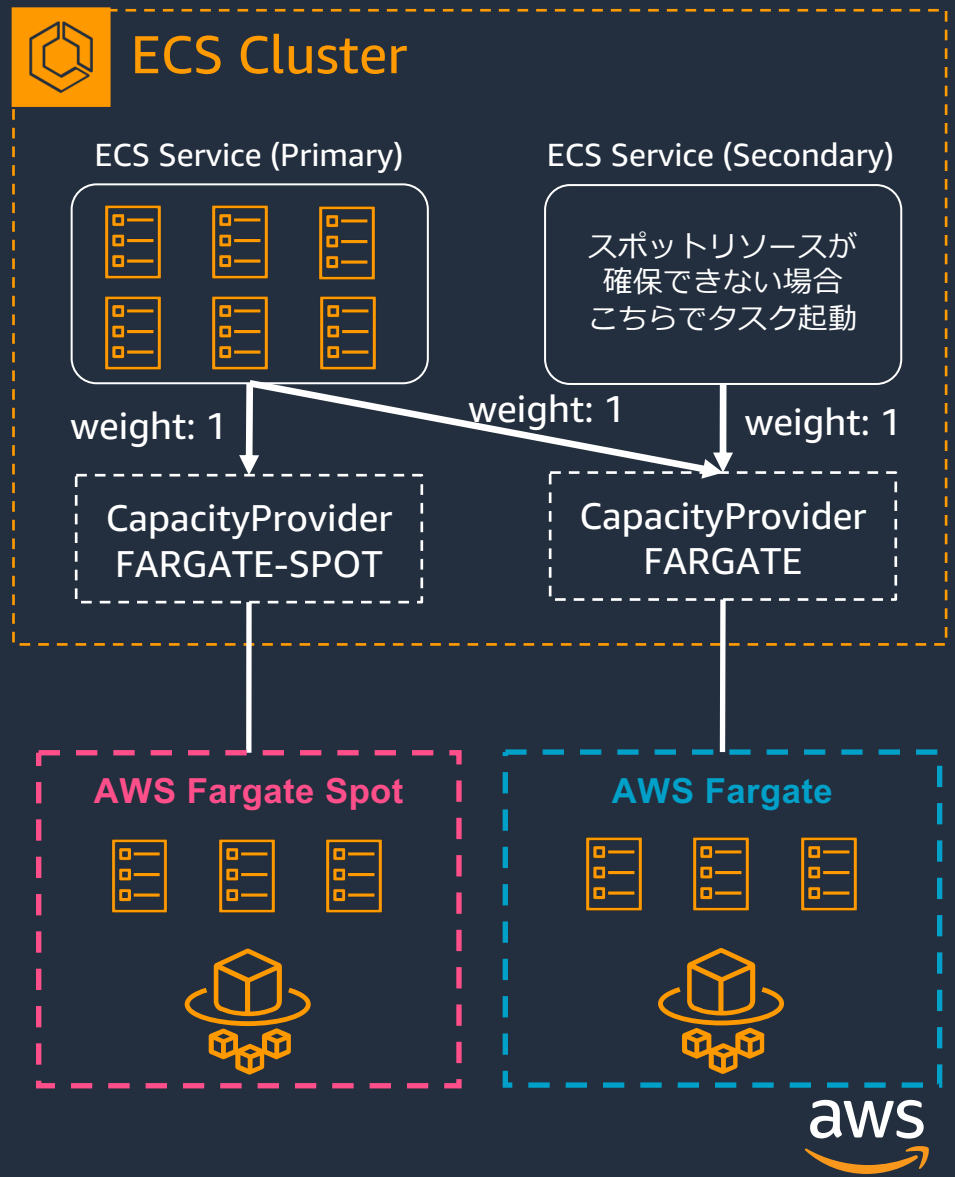
即時のスポットリソース再確保が確約されるわけではない点に注意

aws

スポットリソースが確保できない場合・・・

- Fargate Spot・EC2 ともに、**スポットリソースが再取得できない可能性がある**
- Secondary 用の ECS Service を準備し、
  1) スポットリソースが確保できない場合に Secondary でタスクを起動する
  2) Primary が先にスケールアウトするよう Auto Scaling ポリシーを調節する

  などの工夫によって、**オンデマンドリソースを確保する**アイデアも



まとめ

まとめ

- Amazon ECS のコスト最適化では、データプレーンの最適化がキモ
- スポットリソースの利用には ECS Cluster Auto Scaling の各機能が有用
- スポットリソースの利用において、注意すべきポイントは下記の 2 点
  - スポット中断への対応として、
    アプリケーションの安全な停止 と **代替リソースの確保**が必要
  - 追加のスポットリソースが確保できない場合の備えとして、
    ECS サービスの複数利用 や **スケーリングポリシーの最適化** が有効

aws

Thank You !