このコンテンツは公開から3年以上経過しており内容が古い可能性があります
最新情報についてはサービス別資料もしくはサービスのドキュメントをご確認ください

[AWS Black Belt Online Seminar] AWS Batch

サービスカットシリーズ

Solutions Architect 宮本 大輔
2019/09/11

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

過去資料
https://amzn.to/JPArchive

自己紹介

- 名前
  宮本 大輔 （みやもと だいすけ）
- 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
  HPC Solutions Architect
- 好きな AWS サービス
  - AWS ParallelCluster
  - Amazon FSx for Lustre
  - AWS Snowball シリーズ

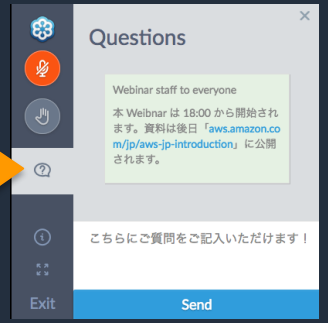
AWS Black Belt Online Seminar とは

「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

内容についての注意点

- 本資料では2019年9月11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.

本セミナーの概要

- 本セミナーで学習できること
  - バッチコンピューティングをクラウドで行う利点
  - AWS Batch とはどのようなサービスか、どういったときに使用するべきか
  - AWS Batch の便利な使い方、利用例

- 対象者
  - HPC や機械学習、メディア処理などのバッチコンピューティングにおいて AWS Batch を利用中または検討中のエンジニア、アーキテクトの方
  - 次の AWS のサービスの概要レベルの知識が前提になります

    Amazon VPC / Amazon EC2 / Amazon S3 などのAWS基礎サービス

目次

目次

- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

バッチコンピューティングとは

本セッションにおける**バッチコンピューティング**とは
シミュレーションなど負荷の高い計算を
スーパーコンピュータやクラスタ等で順次実行していくような処理

例:

- 設計製造での最適化を行うためのパラメータ探索
- モンテカルロシミュレーションによる金融商品のプライシング・リスク計算
- ゲノム分析などの大規模解析
- 3DCG レンダリングや動画エンコードなどのメディア処理

似て非なる『バッチ処理』

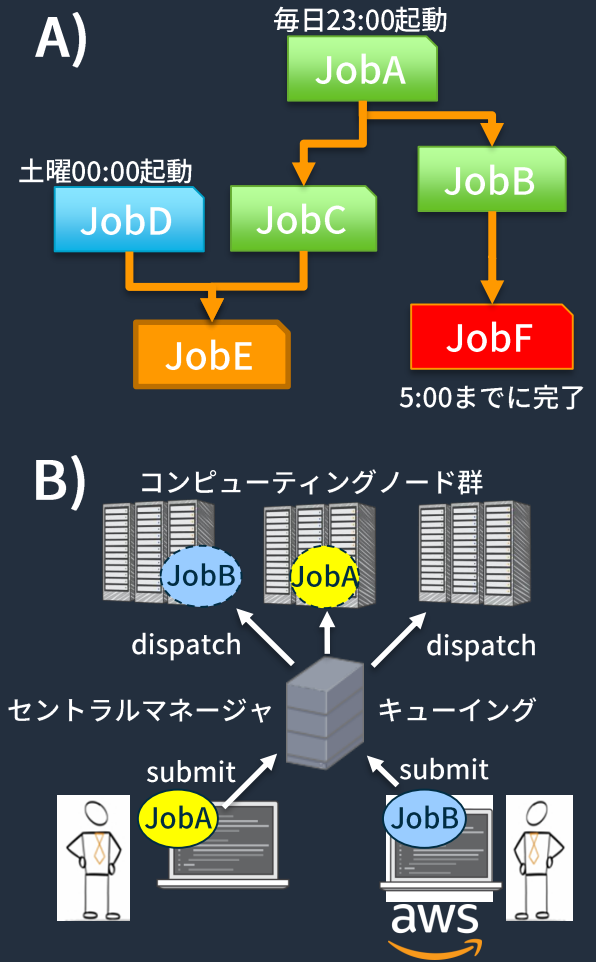
俗に言う『バッチ処理』は２種類ある

A) 定型業務における『バッチ処理』
夜間バッチのように、予めジョブの起動や順序を定義しておいて
ジョブの実行・管理をするシステム（ジョブネット等）
ミドルウェア例：Hinemos、SOS JobScheduler etc.

B) 大規模計算における『バッチ処理』
スーパーコンピュータ等で行う大規模科学計算、CGレンダリング、
機械学習における学習プロセス、ゲノム分析などのアドホックな計算
ミドルウェア例：SGE、Torque、OpenLAVA etc.

AWS Batch はこちら

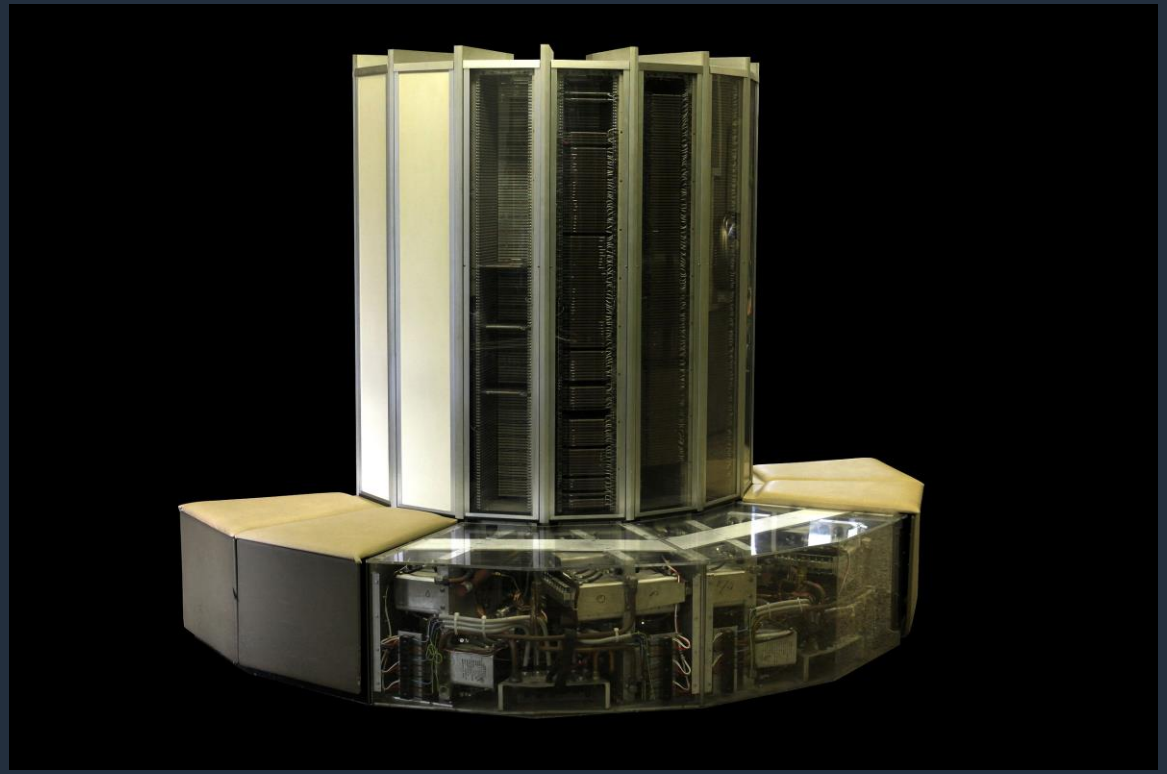
AWS Batchは負荷の高いコンピューティングを効率的に行うためのサービス



昔はコンピュータリソースは高価で貴重だった

CRAY-1:1976

- 世界で最初の商用スーパーコンピューター
- 毎秒1億6700万回の演算速度
- 当時の価格で$886万USD (内ストレージに$100万USD)

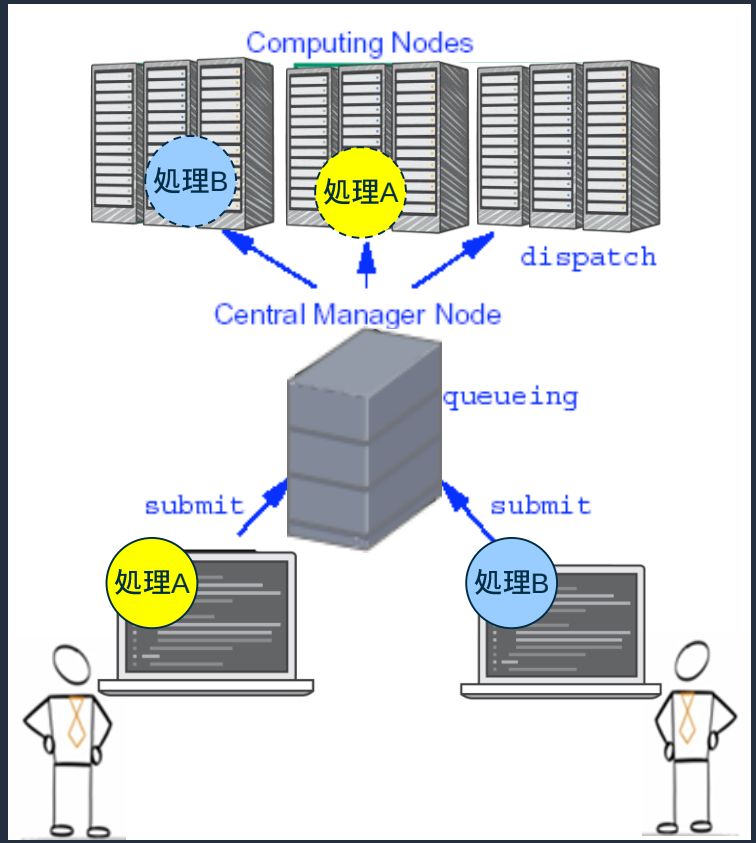


CRAY-1 on display in the hallways of the EPFL in Lausanne.
https://commons.wikimedia.org/wiki/File:Cray_1_IMG_9126.jpg

バッチジョブスケジューラの役割

高価で稀少なスーパーコンピュータを有効活用するために、「ジョブ」をキューイングして順次処理する仕組みができた

「ジョブ」が必要とするCPUやメモリ量に応じてリソースの割り当てや処理の順番を決定することでリソースの空き時間を減らし、有効活用することが可能に



<ジョブスケジューラの処理>
・ユーザが投入したジョブを待ち行列に貯める。(queueing)
・空いているコンピュータにジョブを割り当てて処理させる。(dispatch)
・空いているコンピュータが無い時は、ジョブを待ち行列で待機させ、コンピュータが空いた時点でdispatchし、処理を開始させる。

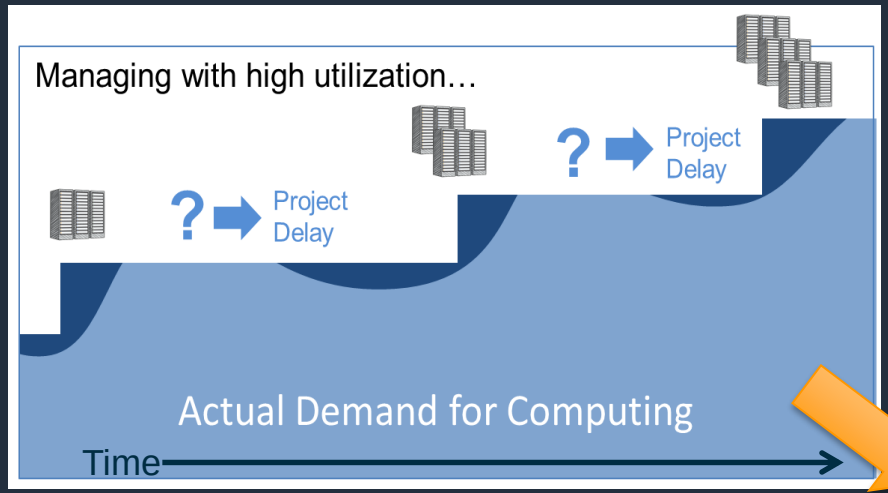
オンプレミスバッチコンピューティング環境の課題

- リソースが限られているため、ジョブが増えると待ち時間が長くなる
- 事前に利用量を想定してマシン構成を考える必要がある
- 多くのユーザが利用するため、汎用的なマシン構成になる
- 数年ごとに更新が必要であり、更新前にはアーキテクチャが古くなっている
- 台数が多いため物理的な管理コストが高い

目次

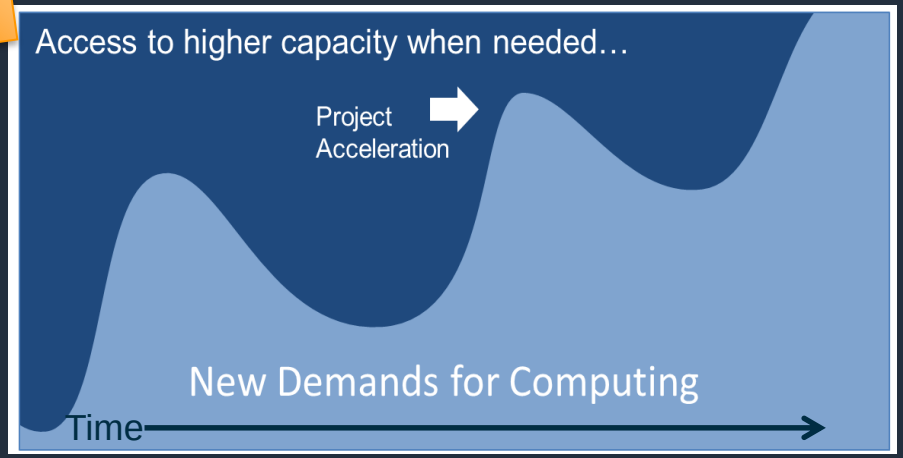
- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

クラウドによって変わる計算リソースの考え方



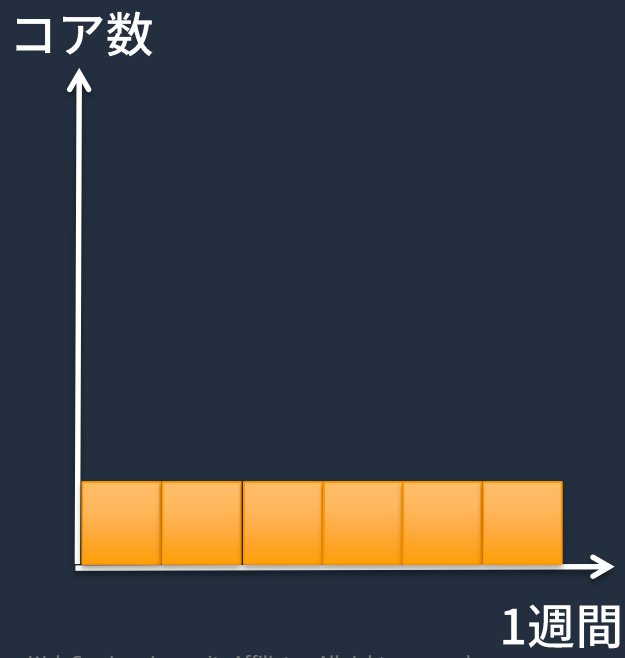
従来は処理のピークに合わせてリソースを予め用意していた。ピーク時に賄えないリソースは、ジョブスケジューラにより、処理するタイミングをずらして対応

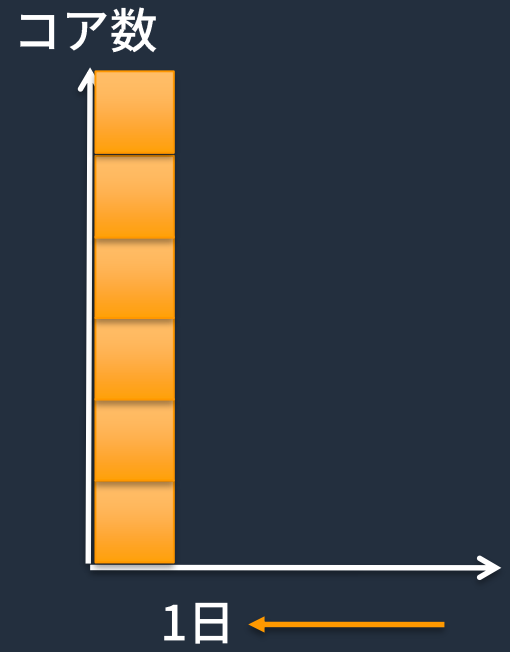
クラウドの場合はその時に必要なリソースを立ち上げて、処理を実行する



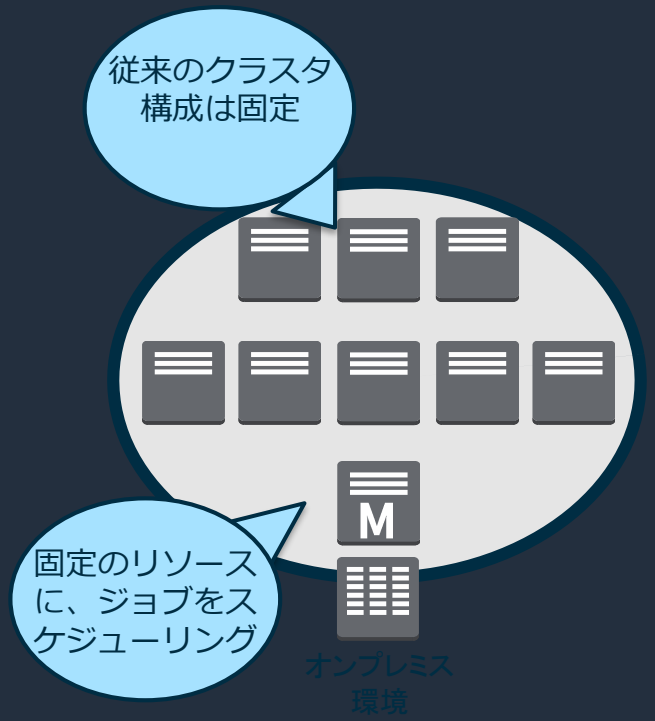
多数のジョブを同時実行して処理時間を短縮できる

従来は手持ちの限られたリソースで、逐次処理していたジョブもクラウドであれば
必要な台数、インスタンスを起動して、一斉処理が可能
AWSのインスタンス費用は原則「時間×台数」なのでどちらもほぼ同じコスト

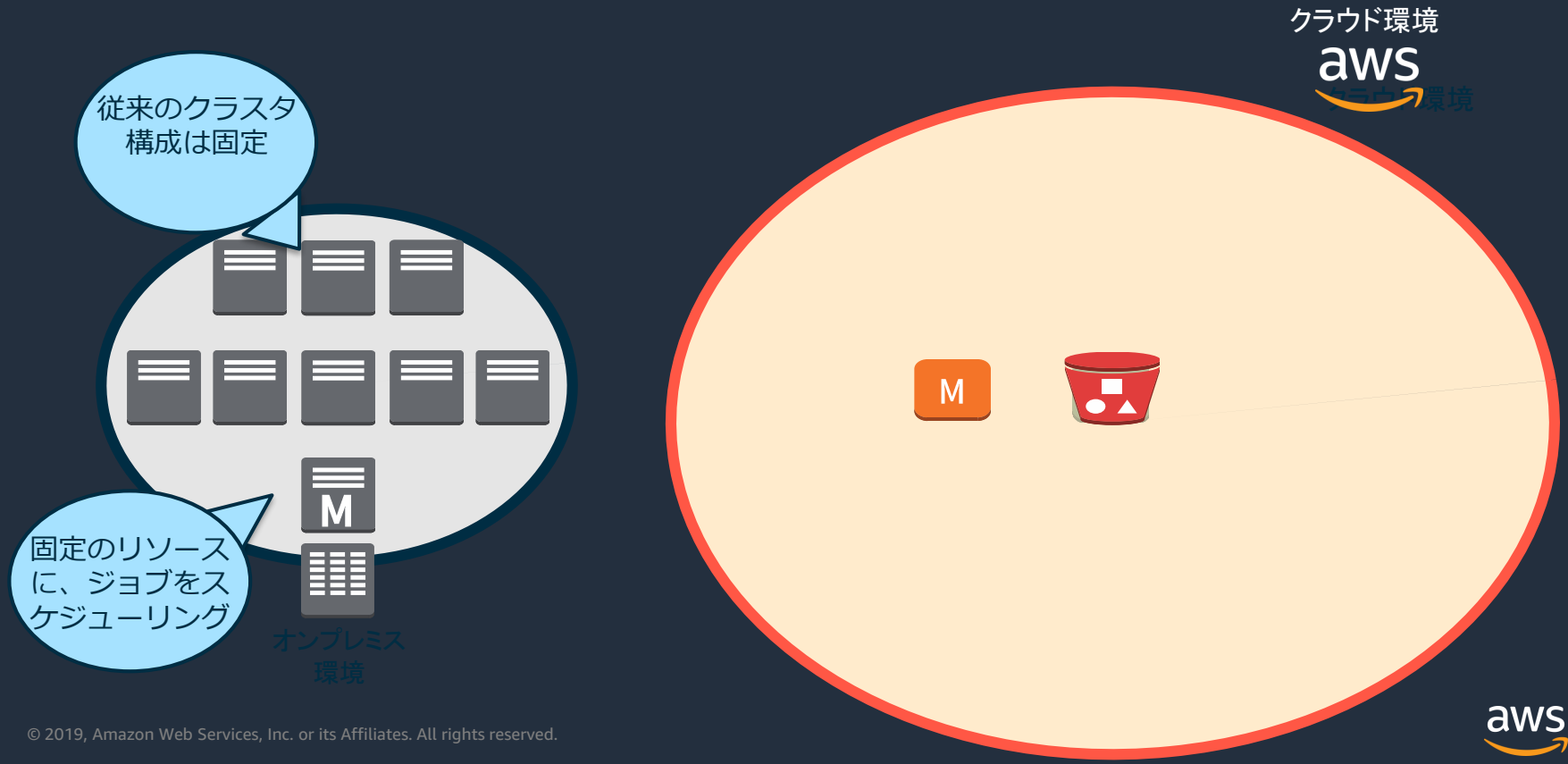




オンプレミスのバッチコンピューティング環境

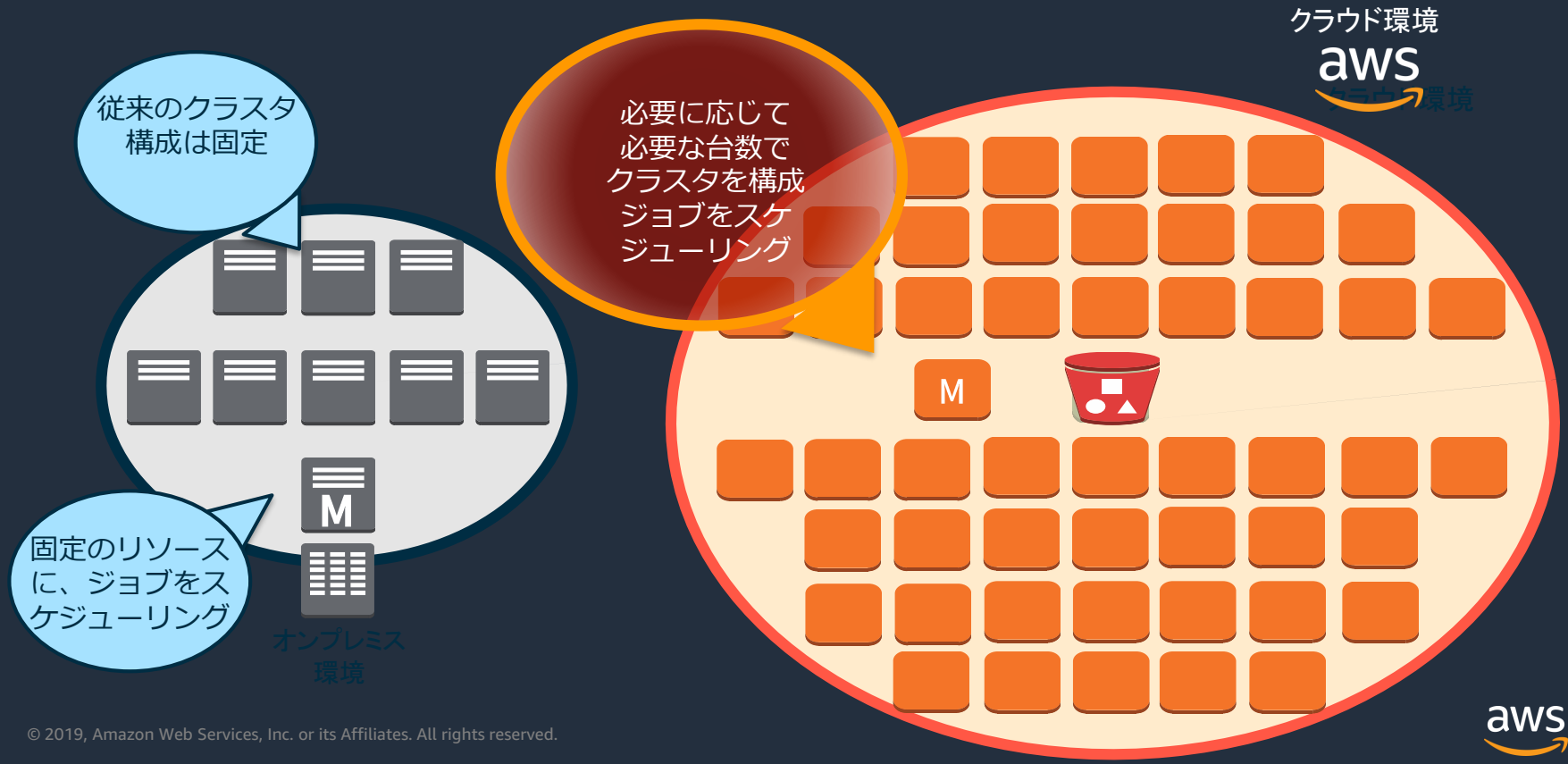


クラウドのバッチコンピューティング環境



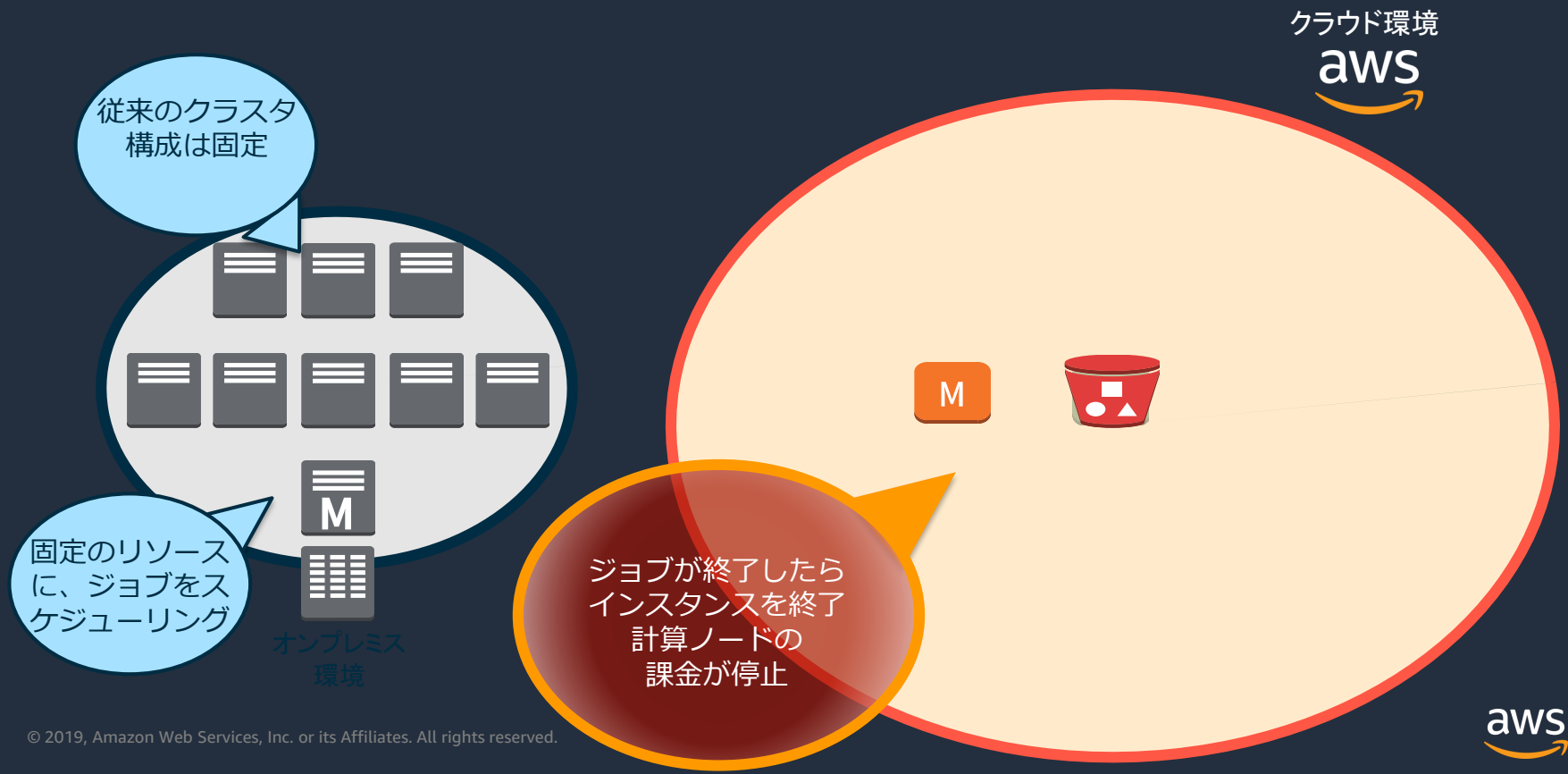
aws

クラウドのバッチコンピューティング環境



aws

クラウドのバッチコンピューティング環境



aws

AWS 上でバッチコンピューティングを行う利点

必要なときに必要な量のリソースを従量課金で利用することが可能

- 複数のジョブを同時に実行することで短時間で終わらせる
- 事前に需要を予測しなくとも、必要なときに増減可能
- タスクに適したマシン構成
  （CPUコア数、メモリ量、GPU / FPGAの有無、ストレージ構成等）
- 最新のCPUを使用することができる
- 計算機の物理的管理負担を削減

バッチコンピューティング on AWSの事例は豊富

Media & Entertainment:

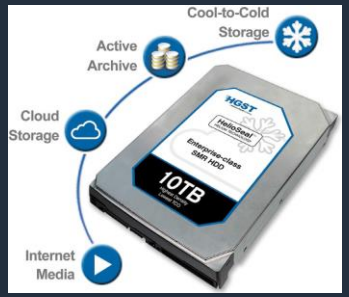
Healthcare & Life Sciences:

Internet Companies:

Scientific Research:

Engineering and Design:

バッチコンピューティング on AWSの具体例

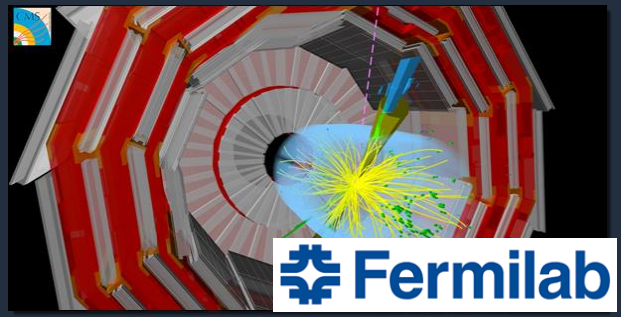


HGST

ドライブヘッド設計で、**数百万パターンに及ぶパラメータスタディ実行**。ピーク時70000以上のコアを同時利用、従来1ヶ月かかる処理を8時間で完了

Fermilab

AWS上に58000コアを伸縮自在に追加するシステムを構築。**290万ジョブを処理**し、従来6週間かかるシミュレーションを10日で完了



NOVARTIS

1000万の化合物のスクリーニング実行。ピーク時 約90000コアを同時利用し39年分の計算処理を9時間で完了

aws

目次

- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

なぜ AWS Batch を使用するか

クラウドではスケールするコンピューティング環境により
コスト効率よく大量の計算を行うことが可能

しかし、実際にそのようなシステムを構築するのは大変！

マスターノードをどう管理するか
スケジューラと計算環境をどのように接続するか
障害対応をどうするか
etc...

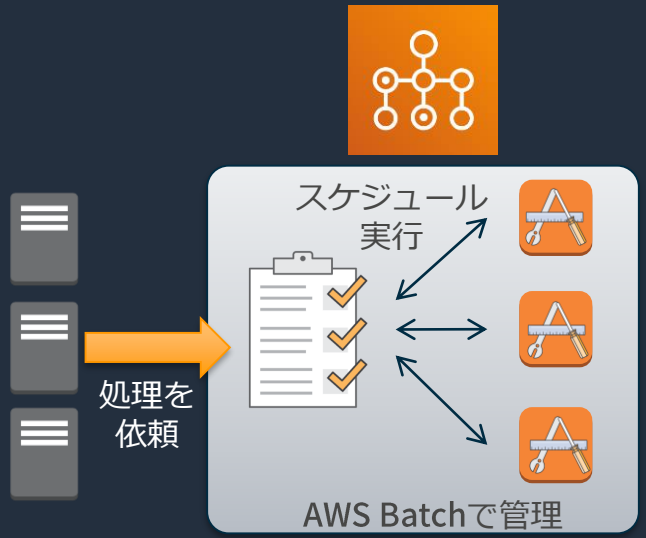
AWS Batch ではバッチコンピューティング環境を
フルマネージドで提供し、インスタンス管理が不要

aws

AWS Batch とは

バッチコンピューティングのため環境をフルマネージドで提供

- AWS Batch がインスタンスの起動や停止を行うため、スケジューラや計算ノードなどの **管理が不要**
- ジョブは **Docker コンテナイメージ** を元に作成し、自動でスケールするコンピューティング環境で実行する
- コンピューティング環境ではインスタンスタイプや vCPU 数、スポットインスタンス利用有無などを任意に指定可能
- 100 万 vCPU クラスの大規模な計算にも対応



コンテナを用意するだけでスケーラブルなバッチコンピューティング環境が得られる

aws

AWS Batch の価格と利用可能リージョン

- AWS Batch の利用自体は 無料
  - AWS Batch によって起動されるEC2インスタンスに課金される

- 利用可能リージョン（2019年9月11日時点）
  - 米国東部 (バージニア北部)、米国西部 (北カリフォルニア)、米国西部 (オレゴン)、米国東部 (オハイオ)、カナダ (中部)
  - 欧州西部 (アイルランド)、欧州 (ロンドン)、欧州 (フランクフルト)、欧州 (パリ)
  - アジアパシフィック (東京)、アジアパシフィック (ソウル)、アジアパシフィック (シドニー)、アジアパシフィック (シンガポール)、アジアパシフィック (ムンバイ)
  - 南米 (サンパウロ)

AWS Batch を利用するメリット

- バッチコンピューティングを行うための環境をフルマネージドで提供
  - ジョブキュー
  - 必要に応じて増減するコンピューティング環境
  - タイムアウト、リトライの処理
- インスタンスタイプだけではなく、**必要な vCPU 数やメモリ量、GPU 数を指定**でき、リソースの有効活用が可能
- 他の AWS サービスとの連携が可能
  - Amazon CloudWatch Events
  - AWS Step Functions
  - AWS CloudTrail etc.

AWS Batch よりも他のサービスが適しているケース

- コンテナ化が難しい場合
- 既存のジョブスケジューラから移行が難しい場合

→ AWS ParallelCluster の利用を検討
SGE や Torque に対応し、オートスケーリングが可能

- 1つのジョブが数秒など短時間で終わる場合

→ AWS Step Functions + AWS Lambda や、S3 Batch の利用を検討

- 機械学習用途でモデル設計から推論エンドポイントまで一気通貫で行いたい場合

→ Amazon SageMaker の利用を検討

目次

- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

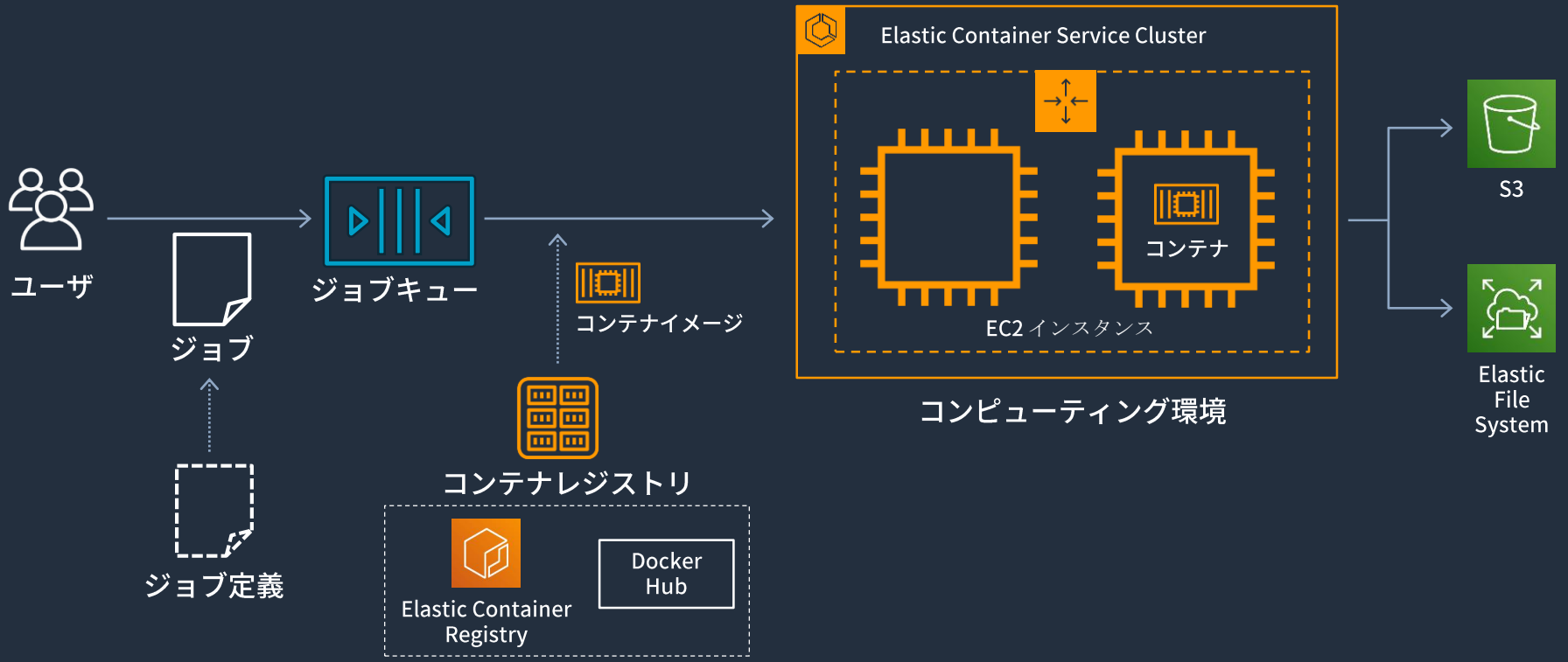
AWS Batch の重要用語・概念

- ジョブ定義 (Job Definitions)
- ジョブ (Jobs)
- ジョブキュー (Job Queues)
- コンピューティング環境 (Compute Environments)

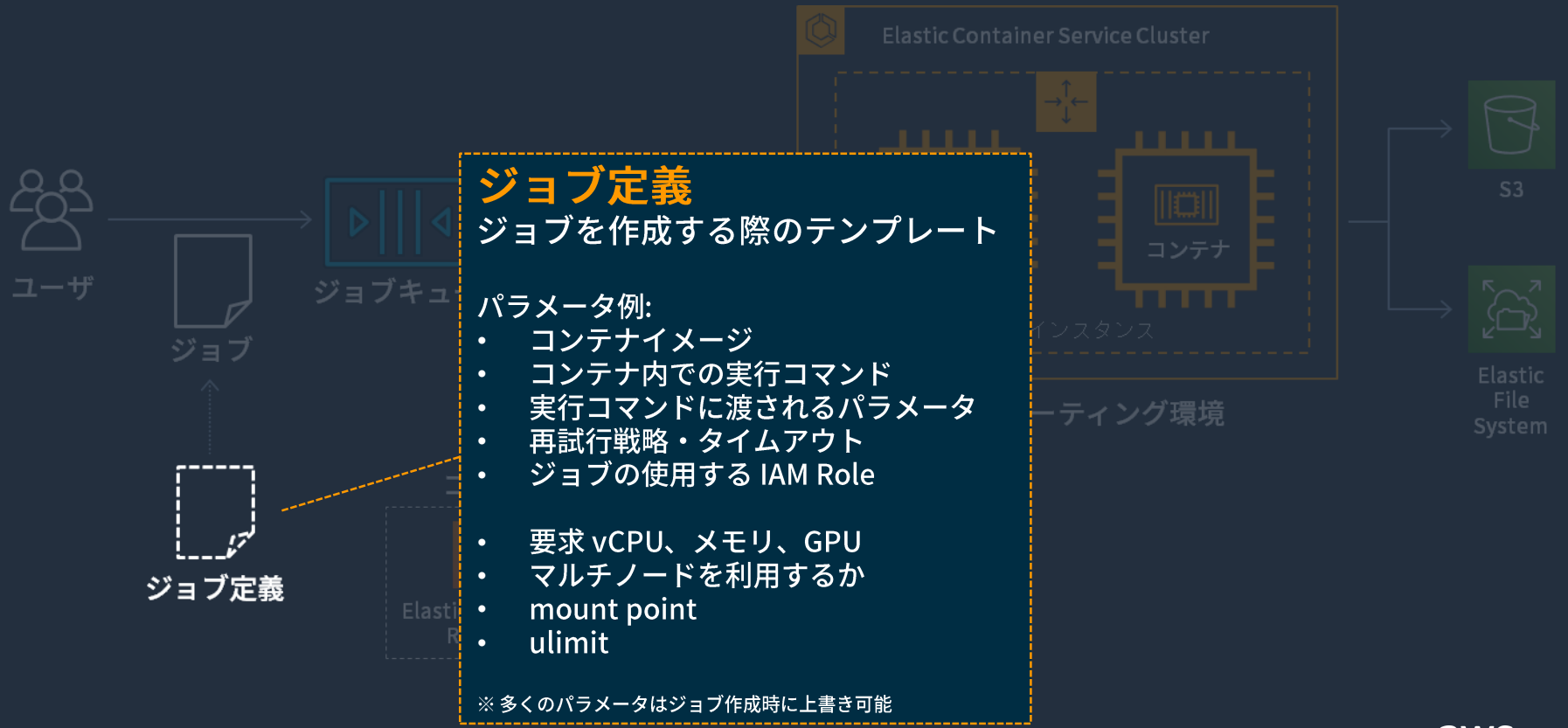
- コンテナレジストリ
- ストレージ

aws

AWS Batch のアーキテクチャ



ジョブ定義



ジョブ定義

ジョブを作成する際のテンプレート

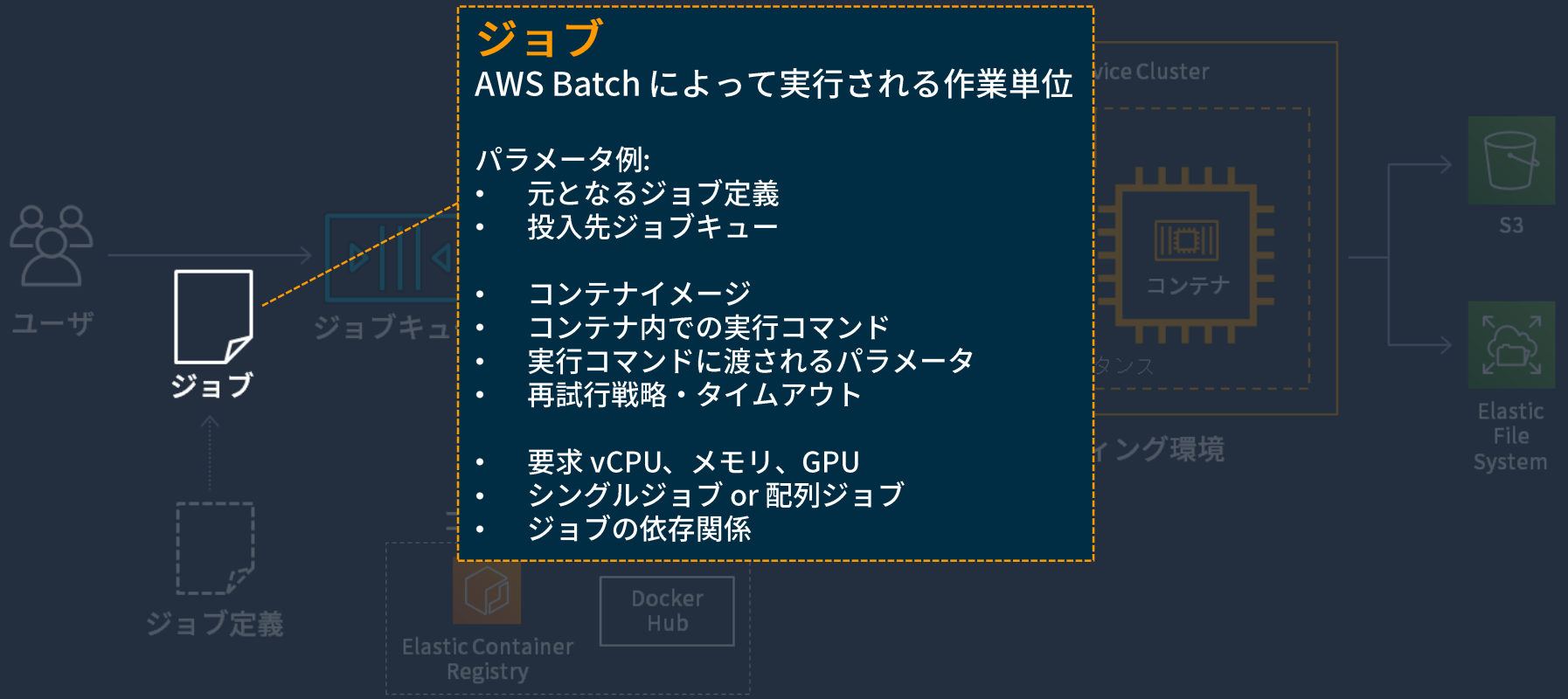
パラメータ例:

- コンテナイメージ
- コンテナ内での実行コマンド
- 実行コマンドに渡されるパラメータ
- 再試行戦略・タイムアウト
- ジョブの使用する IAM Role

- 要求 vCPU、メモリ、GPU
- マルチノードを利用するか
- mount point
- ulimit

※ 多くのパラメータはジョブ作成時に上書き可能

ジョブ



ジョブ

AWS Batch によって実行される作業単位

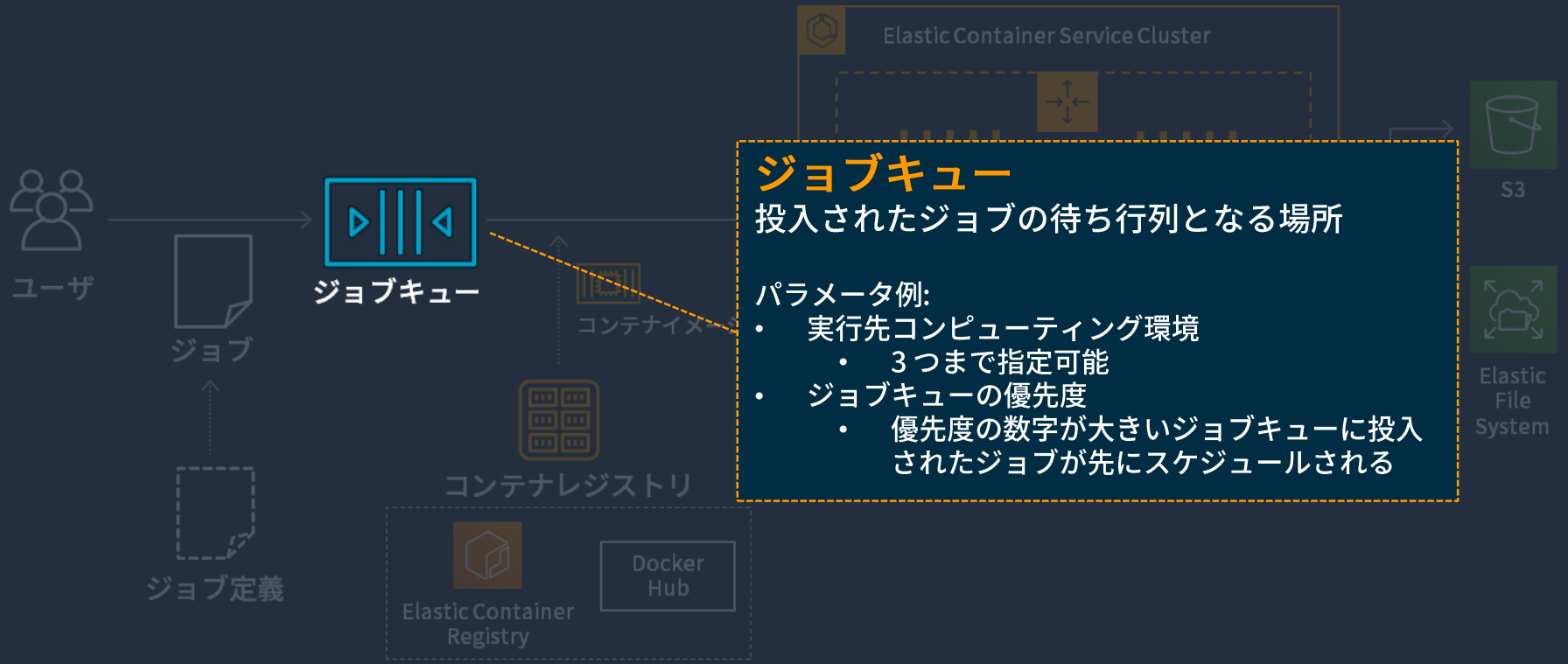
パラメータ例:

- 元となるジョブ定義
- 投入先ジョブキュー

- コンテナイメージ
- コンテナ内での実行コマンド
- 実行コマンドに渡されるパラメータ
- 再試行戦略・タイムアウト

- 要求 vCPU、メモリ、GPU
- シングルジョブ or 配列ジョブ
- ジョブの依存関係

ジョブキュー



ジョブキュー

投入されたジョブの待ち行列となる場所

パラメータ例:

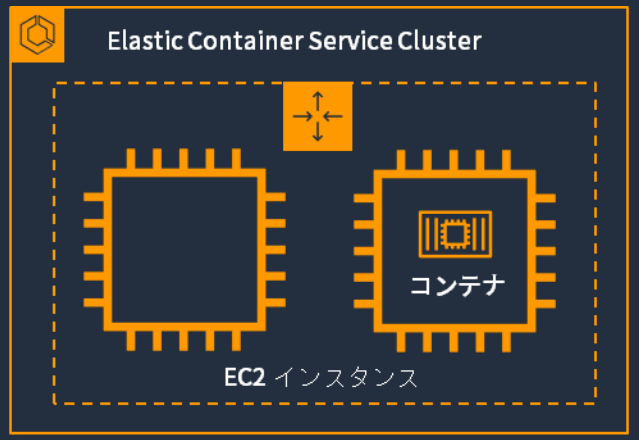
- 実行先コンピューティング環境
  - 3 つまで指定可能
- ジョブキューの優先度
  - 優先度の数字が大きいジョブキューに投入されたジョブが先にスケジュールされる

コンピューティング環境

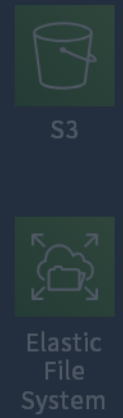
コンピューティング環境
実際に計算を行う ECS クラスター

パラメータ例:
- マネージド or アンマネージド
- （オプション）ユーザ独自の AMI
- （オプション）EC2 起動テンプレート
- オンデマンド or スポット
- 最小 vCPU 数、最大 vCPU 数
  - 最小 vCPU 数を 0 に設定することで、ジョブの無い時はインスタンスを起動しないことが可能
- 許可されたインスタンスタイプ
  - M4、C4、R3の中から合うものが選択されるOptimalや、インスタンスファミリー単位での指定も可能

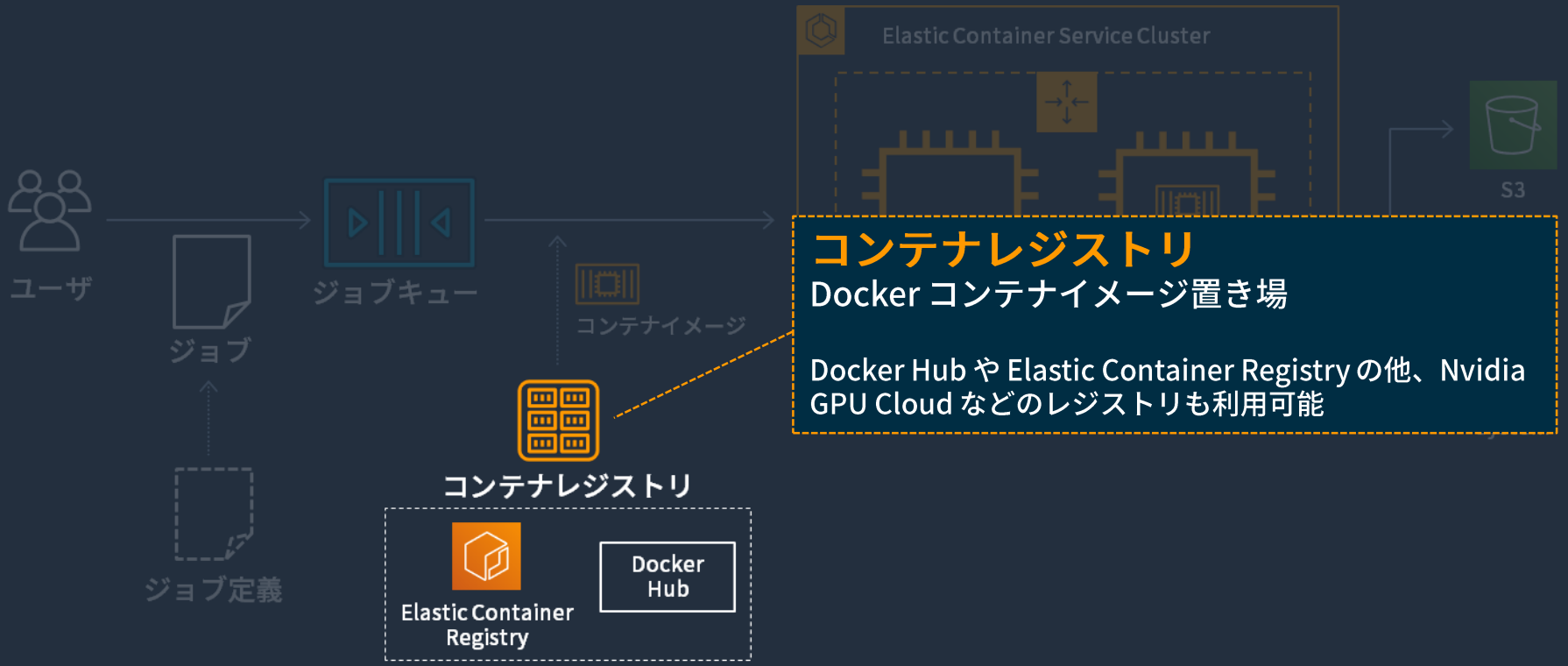
- VPC、Subnet、セキュリティグループ
- Placement Group



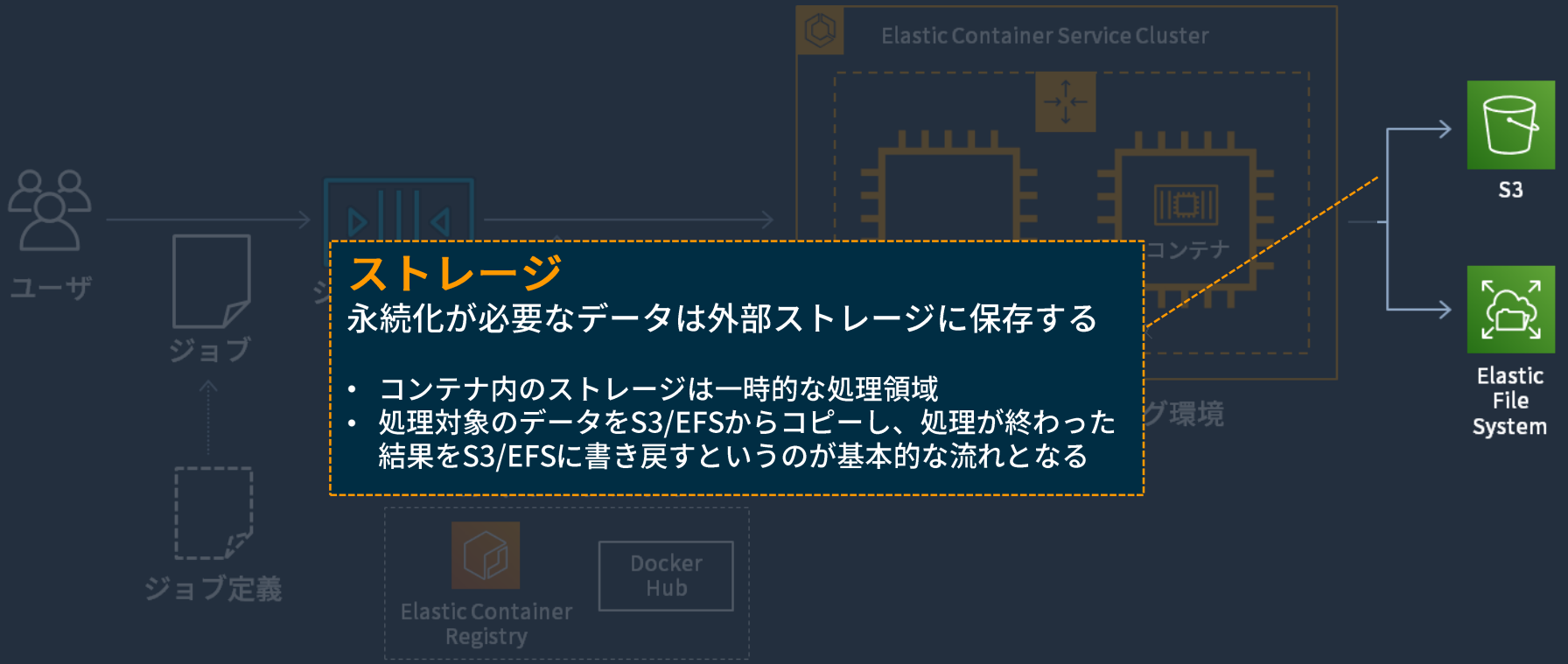
コンピューティング環境



コンテナレジストリ



ストレージ

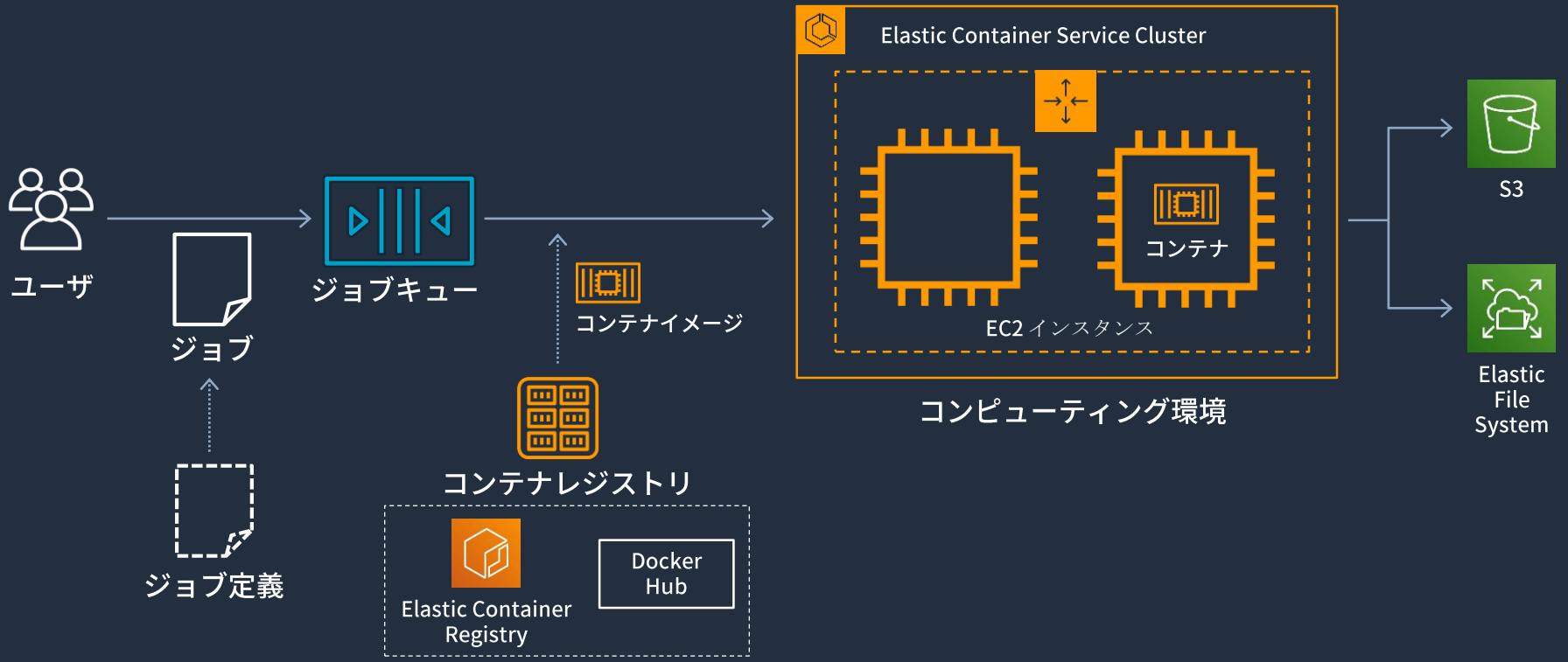


ストレージ

永続化が必要なデータは外部ストレージに保存する

- コンテナ内のストレージは一時的な処理領域
- 処理対象のデータをS3/EFSからコピーし、処理が終わった結果をS3/EFSに書き戻すというのが基本的な流れとなる

AWS Batch のアーキテクチャ



目次

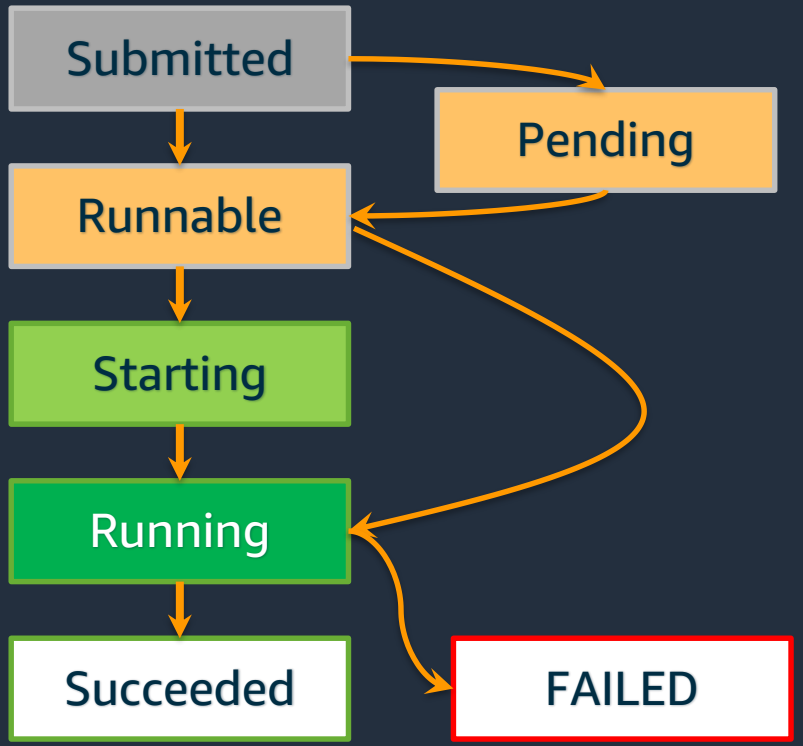
- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

AWS Batch の機能

- ジョブ状態の遷移
- 配列ジョブとジョブ依存関係
- ジョブパターン
- ジョブ・コンピューティング環境 Tips
- セキュリティ関連機能
- 応用: マルチノード並列ジョブ

ジョブ状態の遷移

- 投入されたジョブは右図のような遷移をたどる
- ジョブ状態の変化を Amazon SNS に通知することが可能
- EC2 インスタンスの起動を待っている場合は Runnable 状態となる
- ジョブの要求するリソースが、指定されたコンピューティング環境に含まれない場合は、Runnable のまま停止するため注意
  - 大きな vCPUやメモリを指定し、適合するインスタンスがコンピューティング環境に無い
  - GPUを要求したがコンピューティング環境に GPU を含んだインスタンスが無い



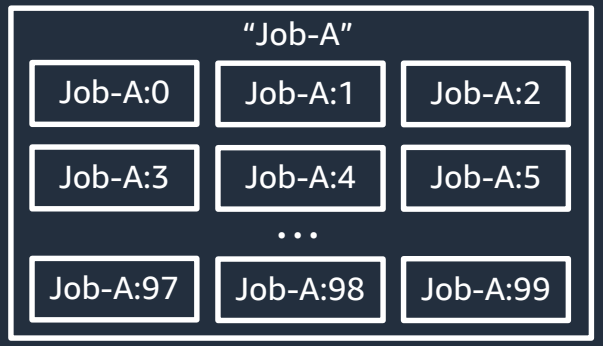
https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/batch_sns_tutorial.html
https://aws.amazon.com/jp/premiumsupport/knowledge-center/batch-job-stuck-runnable-status/

配列ジョブとジョブ依存関係

配列ジョブ

- 一回のジョブ投入で複数の子ジョブを作成
- 子ジョブでは、環境変数 $AWS_BATCH_JOB_ARRAY_INDEX により自分の 配列ID を取得可能
- 配列ID の値により処理を分ける

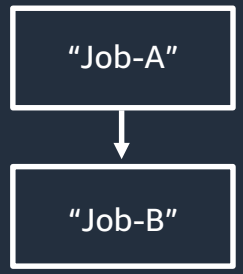
https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/array_jobs.html



ジョブ依存関係

- 依存関係に指定したジョブが完了したら実行可能となる

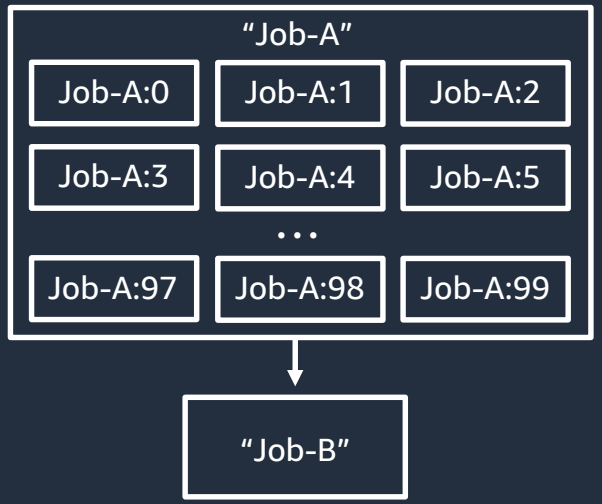
https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/job_dependencies.html



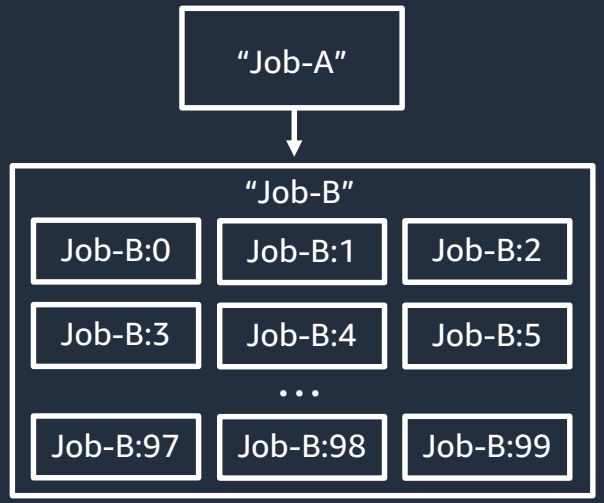
ジョブパターン

配列ジョブとジョブ依存関係を利用することで様々なジョブパターンを作成可能

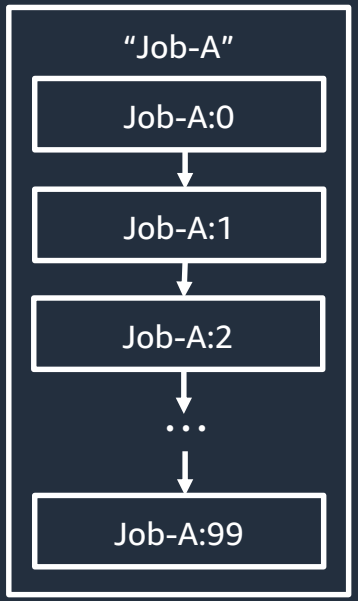
集約パターン



前処理パターン



シーケンシャルパターン



ジョブ・コンピューティング環境 Tips

- ジョブ
  - ジョブ再試行回数/タイムアウトを指定する
    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/job_timeouts.html
    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/job_retries.html
- コンピューティング環境
  - マネージド型・アンマネージド型が指定可能だが、通常はマネージド型でよい
  - インスタンスタイプにはファミリー単位で指定することで、ジョブ要求する vCPU 数やメモリ量に応じて適したものが使用される。**複数のインスタンスファミリーやOptimal（M4 or C4 or R3）を指定することで、様々なジョブに柔軟に対応可能**
  - 複数の AZ （Availability Zones）を指定することでより障害に強い構成となる
  - 最小 vCPU 数を0に指定することで使用しない時にはインスタンスが立ち上がらない設定となる
  - EC2 Launch Template を使用することでより細かいインスタンス設定（UserData、EBS 等）が可能となる
    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/launch-templates.html

セキュリティ関連機能

- IAM Policy による AWS Batch API 実行時の制約が設定可能
  - コンテナ内の POSIX ユーザの制限
  - 利用可能なコンテナイメージ・レジストリの制限

    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/ExamplePolicies_BATCH.html#iam-example-job-def
- ジョブ実行コンテナに対する IAM Role 設定が可能
  - 特定S3バケットへのアクセスなど、必要最低限の権限のみ付与することを推奨

    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/job_definition_parameters.html#containerProperties
- ジョブの投入などのAPI 呼び出しは Cloud Trail に記録される

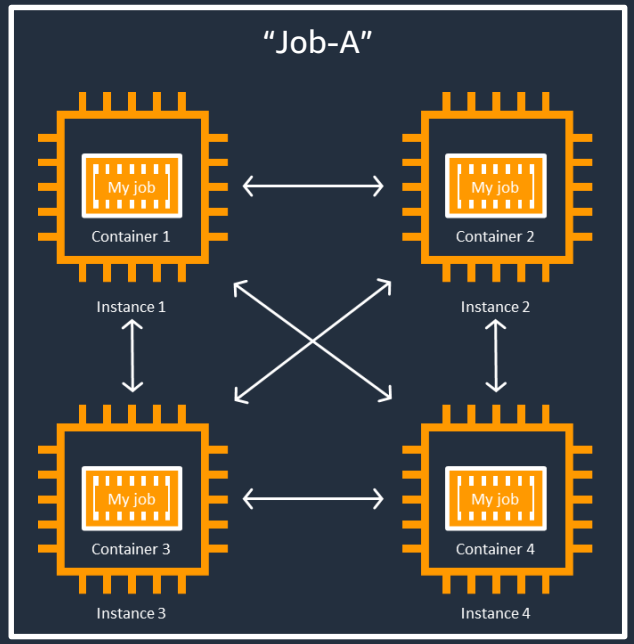
    https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/logging-using-cloudtrail.html

aws

応用: マルチノード並列ジョブ

複数コンテナ間で MPI/NCCL 通信を行い並列に処理を行うジョブも作成可能

- マルチノードの有効化はジョブ定義でのみ可能
- インスタンスタイプは固定となる
- スポットインスタンスを指定したコンピューティング環境では利用できない
- クラスタープレイスメントグループの利用も検討
- EFA (Elastic Fabric Adapter) も利用可能
    - EC2 起動テンプレートで有効化
- 1 ジョブの実行時間は短縮できるが、特に大量のジョブを実行する場合は価格性能比に注意



https://aws.amazon.com/jp/blogs/compute/building-a-tightly-coupled-molecular-dynamics-workflow-with-multi-node-parallel-jobs-in-aws-batch/
https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/multi-node-parallel-jobs.html
https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/efa.html

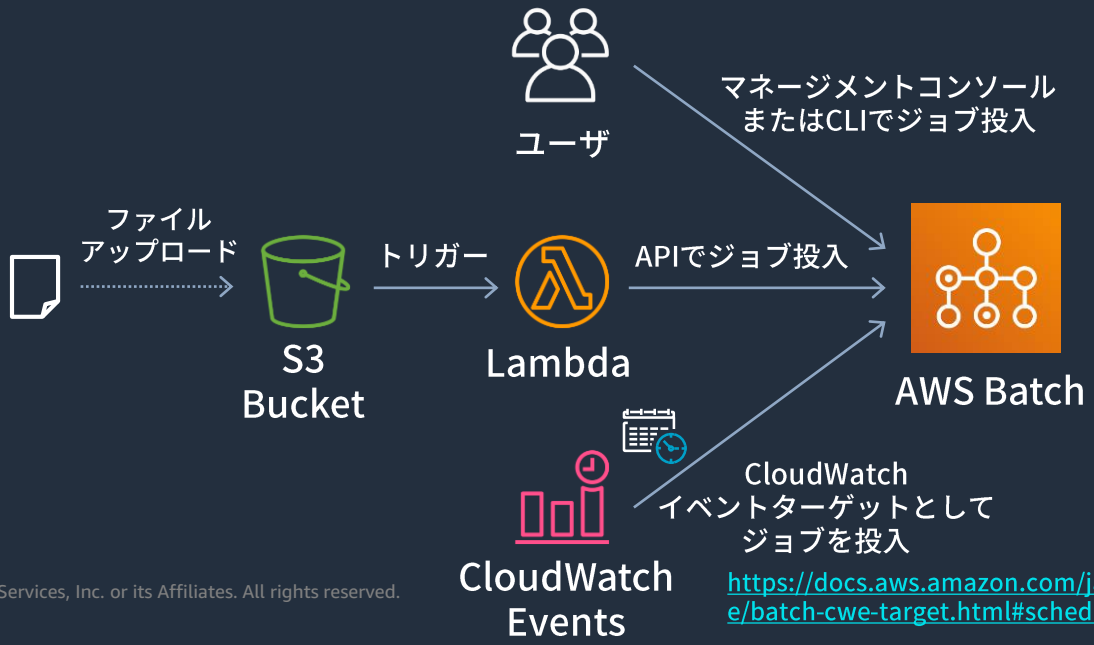
AWS Batch の活用方法

- ジョブの投入方法
- スポットインスタンス活用
- ストレージの選択
- コンテナイメージ作成の自動化
- 複雑なジョブワークフローの作成

ジョブの投入方法

他のサービスと連携し、様々なジョブ投入方法が可能

- ユーザがマネージメントコンソール/CLIからジョブを投入
- S3 へのファイルアップロードをトリガーに、Lambda からアップロードされたファイルを処理するジョブを投入
- Cloud Watch Events により決められた時刻にジョブを投入



https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/batch-cwe-target.html#scheduled-batch-job

スポットインスタンス活用 1

AWS Batch のようなバッチ処理はスポットインスタンスを利用することでコスト効率よく実行可能

- スポットインスタンスはオンデマンドインスタンスに比べ **非常に安価（最大 9 割引）で利用可能** だが、利用中に中断が発生し、**インスタンスが Terminate される事がある**

活用ポイント:

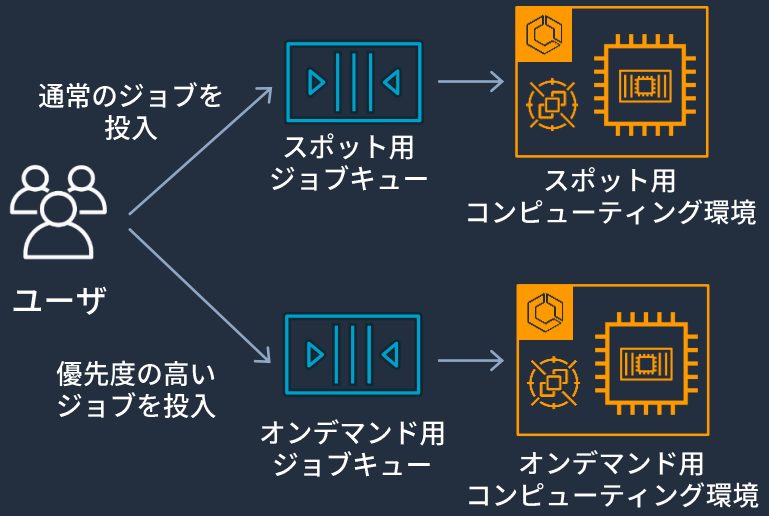
- **コンピューティング環境でインスタンスタイプ を幅広く指定する（Optimal を指定すると M4、C4、R3から適したものが選択される）**
- AWS Batch ではジョブに **再試行回数** を設定することが可能
  これにより、インスタンスが中断しジョブが失敗した場合には自動で再実行される
- 一つのジョブの処理内容を短時間で終わるように分割する
- 再実行された際に途中から処理を再開できるよう、定期的に結果をS3/EFSに出力する、チェックポイント方式とする
- 東京リージョンにこだわらず、スポット価格の安いリージョンを使用する

https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/job_retries.html
https://www.slideshare.net/AmazonWebServicesJapan/20190306-aws-black-belt-online-seminar-amazon-ec2

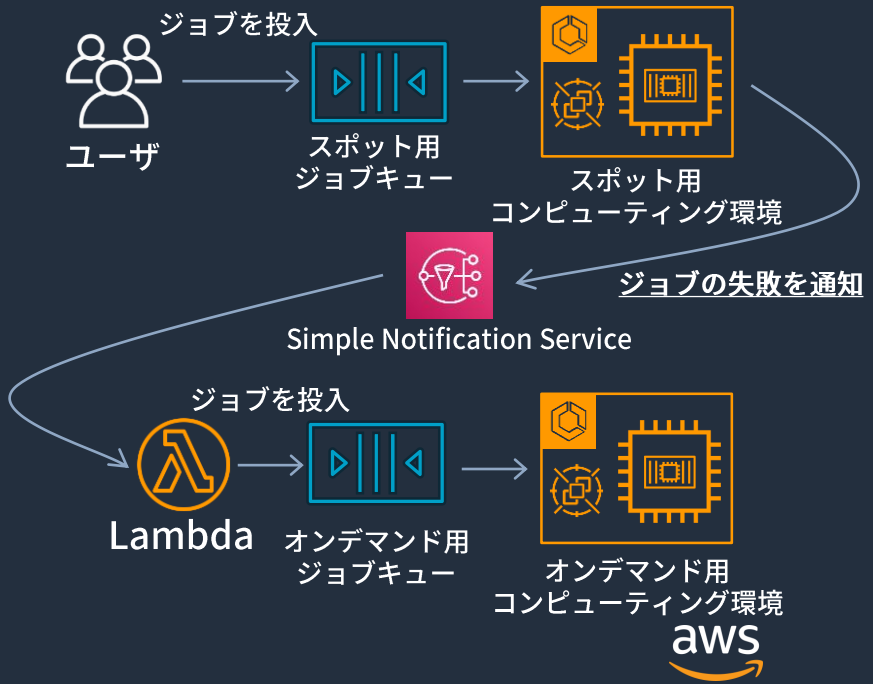
スポットインスタンス活用 2

ジョブのフローを工夫することで、オンデマンドインスタンスとスポットインスタンスを組み合わせて利用することも可能

スポット用キュー、オンデマンド用キューを使い分ける



スポットインスタンスでの実行に失敗した際にオンデマンド用キューに再投入

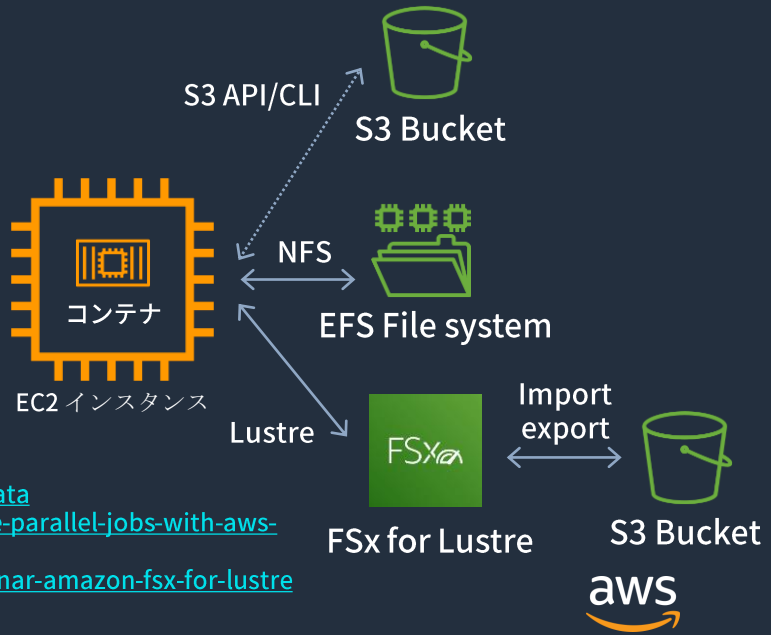


ストレージの選択

コンテナイメージ内にはデータは置かず、他のストレージサービスと連携する

同時に複数のコンテナからアクセスされることを想定し、適したパフォーマンスのストレージを選択する必要がある

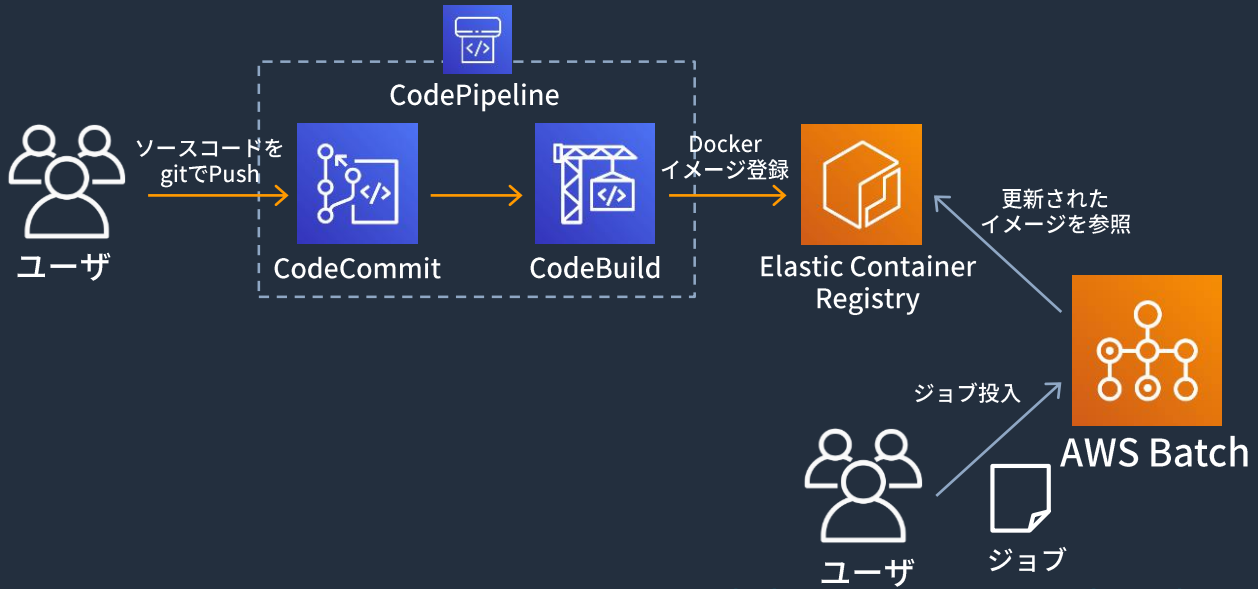
- S3: スケーラブルなパフォーマンスを持つため第一選択候補。マウントして扱うことはできないため、ファイルを一度ローカルにコピーする必要がある。小さいファイルが大量にある場合は事前に圧縮を行うと良い
- EFS: NFSによりマウント可能、パフォーマンスは利用容量に依存するため注意が必要。起動テンプレート内のユーザーデータでマウント
- FSx for Lustre: マウント可能、高いスループットに対応、S3連携 (import/export) といった利点がある。専用クライアントが必要なため独自AMIを作成し起動テンプレート内のユーザーデータでマウント



https://docs.aws.amazon.com/ja_jp/batch/latest/userguide/launch-templates.html#lt-user-data
https://aws.amazon.com/jp/blogs/compute/scalable-deep-learning-training-using-multi-node-parallel-jobs-with-aws-batch-and-amazon-fsx-for-lustre/
https://www.slideshare.net/AmazonWebServicesJapan/20190319-aws-black-belt-online-seminar-amazon-fsx-for-lustre

コンテナイメージ作成の自動化

一般的なコンテナイメージ作成の流れと同様、Code シリーズと ECR (Elastic Container Registry)、または GitHubとDocker Hub を利用することで、Dockerfile の更新からコンテナレジストリへの登録までを自動化可能

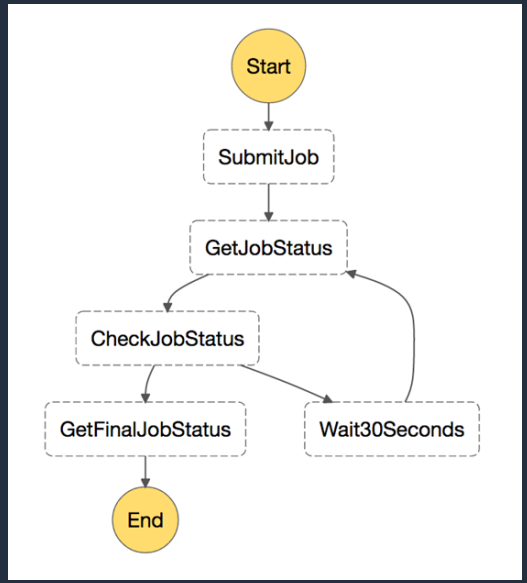


https://docs.aws.amazon.com/ja_jp/codebuild/latest/userguide/sample-docker.html

複雑なジョブワークフローの作成

ワークフローを管理できるサービスである AWS Step Functions を用いることで AWS Batch のジョブ状態を管理し、より複雑なジョブパターンや、再試行戦略が可能

- ジョブの状態をチェックし、一定時間 RUNNING に変化しなければキャンセル or 通知
- Lambda などで処理対象のファイルをリストアップしてから配列ジョブを実行
- ゲノム分析などにおける複雑な前処理のパターンも記述可能



AWS Batch ジョブのステータスチェックを行うワークフロー例

https://aws.amazon.com/jp/blogs/compute/building-simpler-genomics-workflows-on-aws-step-functions/
https://www.slideshare.net/AmazonWebServicesJapan/20190522-aws-black-belt-online-seminar-aws-step-functions

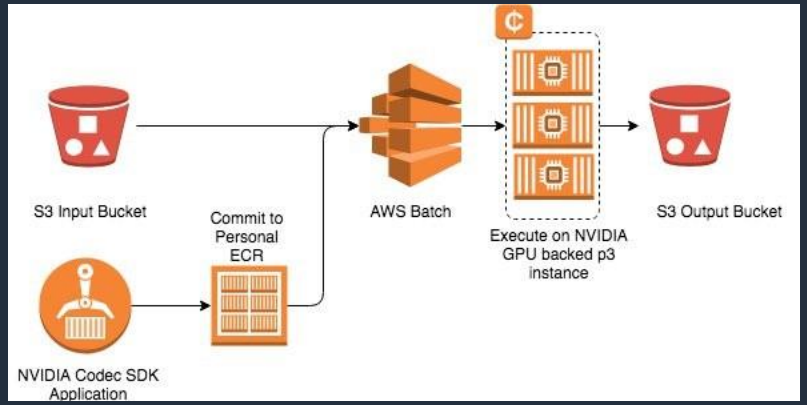
目次

- バッチコンピューティングとは
- バッチコンピューティングにおけるクラウド活用
- AWS Batch: 概要
- AWS Batch: アーキテクチャ
- AWS Batch: 機能と活用方法
- AWS Batch: 利用例

例：メディア処理

GPUを用いた動画のエンコード処理

- （現在では）GPU インスタンスを指定した場合は ECS GPU-optimized AMI が使用されるため、コンテナの中から GPU を扱う事ができる
- コンピューティング環境に、P3 family、P2 family、G3 family などを複数指定し、GPU 数リクエストでジョブを実行することが可能
- スポットインスタンスと組み合わせることでコスト効率の良い処理を実現



https://aws.amazon.com/jp/about-aws/whats-new/2019/04/AWS-Batch-supports-GPU-Scheduling/
https://aws.amazon.com/jp/blogs/compute/deploy-an-8k-hevc-pipeline-using-amazon-ec2-p3-instances-with-aws-batch/

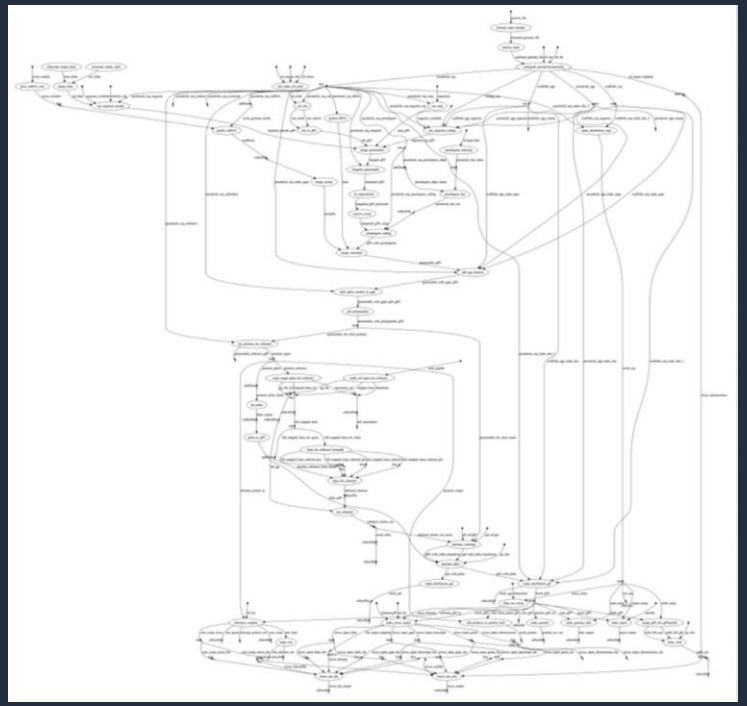
例: ゲノム分析基盤

ゲノム分析分野で利用される

- Nextflow: https://www.nextflow.io
- Cromwell: https://cromwell.readthedocs.io

といったソフトウェアが AWS Batch に対応しており、複雑なワークフロー中の一部分を AWS Batch 上で実行できる

コンピューティング環境でFPGAインスタンス（F1 family）を指定し、Illumina DRAGENを利用し、高速な処理を行うことも可能



ゲノム分析ワークフロー例

https://www.nextflow.io/blog/2017/scaling-with-aws-batch.html
https://aws.amazon.com/jp/blogs/compute/using-cromwell-with-aws-batch/
https://aws.amazon.com/jp/quickstart/architecture/illumina-dragen/

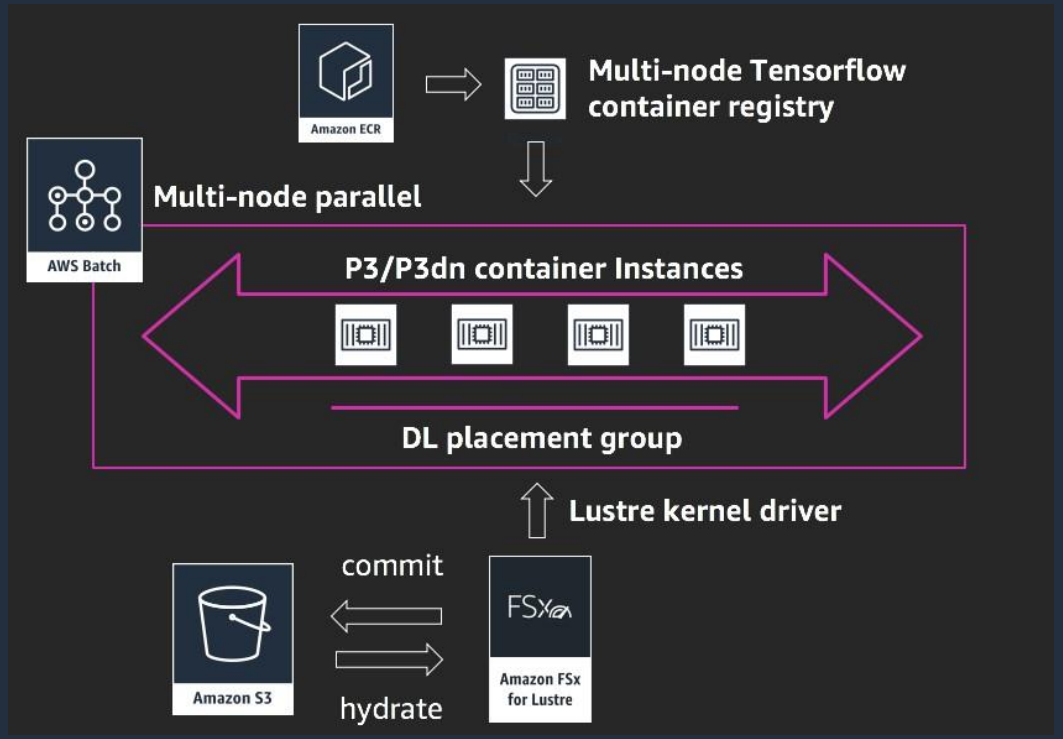
例: MPIを用いた分子動力学計算

- マルチノード並列ジョブにより GROMACS のような MPI 並列のプログラムも実行可能
- AWS Batch 上でマルチノード並列ジョブを大量に実行することで高速なパラメータ探索が可能
- Cluster Placement Group に加え、現在では EFA (Elastic Fabric Adapter) にも対応

https://aws.amazon.com/jp/blogs/compute/building-a-tightly-coupled-molecular-dynamics-workflow-with-multi-node-parallel-jobs-in-aws-batch/

例：機械学習基盤

- ResNet-50によるImageNetデータセットの学習
- EC2 p3.16xlarge × 20台（NVIDIA Tesla V100 160枚）
- Tensorflow コンテナを AWS Batch のマルチノード並列ジョブで実行
- ストレージには FSx for Lustre を利用
- AWS Batch と FSx for Lustre を使用することで**マネージドサービスによるスケーラブルな深層学習環境を構築**できた



https://aws.amazon.com/jp/blogs/compute/scalable-deep-learning-training-using-multi-node-parallel-jobs-with-aws-batch-and-amazon-fsx-for-lustre/

まとめ

- AWS Batch を利用することで、クラウド上にスケーラブルかつフルマネージドなバッチコンピューティング環境を簡単に 作成
- 配列ジョブ・依存関係の設定に加え、AWS Step Functions や 3rd party software により多種多様なワークフローを実現
- スポットインスタンスを有効活用することで、コスト効率よく計算を実行できる
- 設計におけるパラメータ探索やゲノム解析、機械学習、メディア処理など様々なワークロードで活用

参考資料

利用例

- AWS re:Invent 2018 “Set Up a Million-Core Cluster to Accelerate HPC Workloads (CMP404) ”
  - https://www.slideshare.net/AmazonWebServices/set-up-a-millioncore-cluster-to-accelerate-hpc-workloads-cmp404-aws-reinvent-2018
- AWS re:Invent 2018 “Intro to AWS Batch & How AQR Capital leverages AWS to Identify New Investment Signals (CMP372)”
  - https://www.slideshare.net/AmazonWebServices/intro-to-aws-batch-how-aqr-capital-leverages-aws-to-identify-new-investment-signals-cmp372-aws-reinvent-2018
- AWS Blog “Building a tightly coupled molecular dynamics workflow with multi-node parallel jobs in AWS Batch”
  - https://aws.amazon.com/jp/blogs/compute/building-a-tightly-coupled-molecular-dynamics-workflow-with-multi-node-parallel-jobs-in-aws-batch/
- AWS Blog “Building Simpler Genomics Workflows on AWS Step Functions”
  - https://aws.amazon.com/jp/blogs/compute/building-simpler-genomics-workflows-on-aws-step-functions/
  - https://github.com/aws-samples/aws-batch-genomics
- Quick Start: Illumina DRAGEN on AWS
  - https://aws.amazon.com/jp/quickstart/architecture/illumina-dragen/
- AWS Blog “Accelerating Precision Medicine at Scale”
  - https://aws.amazon.com/jp/blogs/compute/accelerating-precision-medicine-at-scale/

aws

利用例

- AWS Blog “Deep Learning on AWS Batch”
  - https://aws.amazon.com/pt/blogs/compute/deep-learning-on-aws-batch/
- AWS Blog “Scalable deep learning training using multi-node parallel jobs with AWS Batch and Amazon FSx for Lustre”
  - https://aws.amazon.com/jp/blogs/compute/scalable-deep-learning-training-using-multi-node-parallel-jobs-with-aws-batch-and-amazon-fsx-for-lustre/
- AWS Blog “Deploy an 8K HEVC pipeline using Amazon EC2 P3 instances with AWS Batch”
  - https://aws.amazon.com/jp/blogs/compute/deploy-an-8k-hevc-pipeline-using-amazon-ec2-p3-instances-with-aws-batch/

関連 Black Belt Online Seminar

- AWS Black Belt Online Seminar “Amazon EC2スポットインスタンス”
  - https://www.slideshare.net/AmazonWebServicesJapan/20190306-aws-black-belt-online-seminar-amazon-ec2
- AWS Black Belt Online Seminar “Amazon Container Services”
  - https://www.slideshare.net/AmazonWebServicesJapan/20180214-aws-black-belt-online-seminar-amazon-container-services
- AWS Black Belt Online Seminar “Amazon ECS Deep Dive”
  - https://www.slideshare.net/AmazonWebServicesJapan/20190731-black-belt-online-seminar-amazon-ecs-deep-dive-162160987
- AWS Black Belt Online Seminar “AWS Step Functions”
  - https://www.slideshare.net/AmazonWebServicesJapan/20190522-aws-black-belt-online-seminar-aws-step-functions
- AWS Black Belt Online Seminar “ヘルスケア・ライフサイエンス業界における AWS 活用”
  - https://www.slideshare.net/AmazonWebServicesJapan/20190423-aws-black-belt-online-seminar-aws
- AWS Black Belt Online Seminar “Amazon FSx for Lustre”
  - https://www.slideshare.net/AmazonWebServicesJapan/20190319-aws-black-belt-online-seminar-amazon-fsx-for-lustre

Q&A

お答えできなかったご質問については
AWS Japan Blog
「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

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