「プライベート富岳」構築マニュアル

2025年4月1日
ver0.01

国立研究開発法人 理化学研究所 計算科学研究センター／
株式会社 JSOL／株式会社 理研数理

目次

１．「バーチャル富岳」とは

（１）「バーチャル富岳」概要

「バーチャル富岳」の目的

- 「富岳」のソフトウェアの成果を広く普及する。

  スーパーコンピュータ「富岳」向けに高度に準備、チューニングされたソフトウェアスタックをパッケージ化して世界中に提供することで、スーパーコンピュータ用ソフトウェアスタックのデファクトスタンダードを目指す。

- 共通環境でアプリを作成・活用できる仕組みを提供する。

  従来のスーパーコンピュータの利用者は、個別の環境に習熟する必要があり、別のスーパーコンピュータに移行する際に再学習する必要があった。またスーパーコンピュータ環境の提供側もシステム更新ごとにソフトウェアをゼロから整備する必要があり大変な手間や投資が必要であった。「バーチャル富岳」は、共通環境でアプリケーションを作成・活用できる仕組みを提供することで、これらの課題の解決を目指す。

「バーチャル富岳」の特徴

- 最先端の研究プラットフォームに必要なソフトウェアを厳選している。
- 「富岳」は現在アクティブな主力システムであるため、すべてのソフトウェアが継続的に更新される。
- 標準仕様はアプリケーション中心である。（基本的にミドルウェアは含まれない）
- Spackやコンテナ仮想化技術などの汎用技術に基づいて標準仕様を定義している。

（２）「サテライト富岳」と「プライベート富岳」

「バーチャル富岳」には「サテライト富岳」、「プライベート富岳」の２種類の環境がある。
本マニュアルでは、２種類の環境のうち「プライベート富岳」についての環境構築手順について解説する。

- 「サテライト富岳」

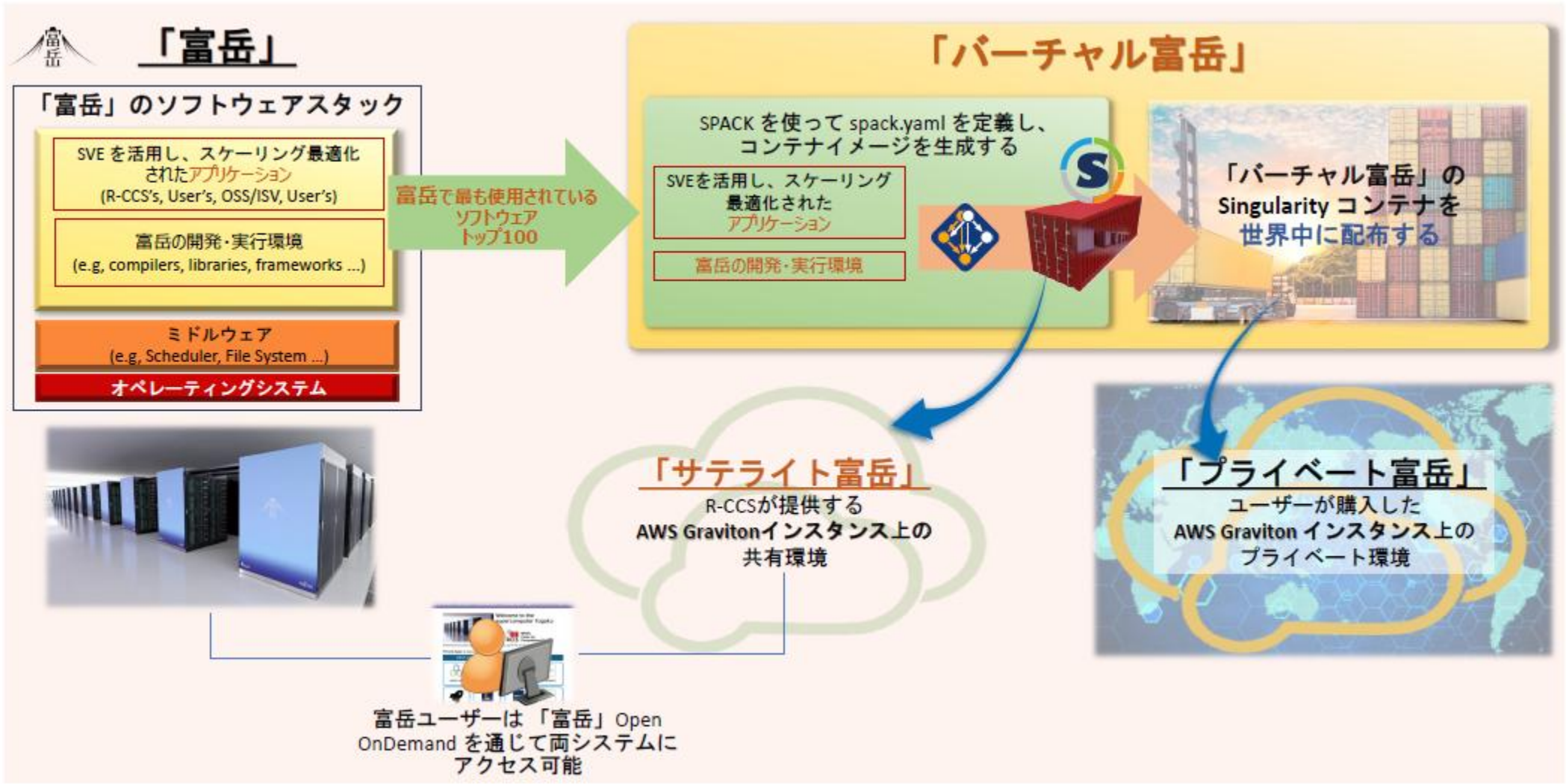
  R-CCSが提唱する「バーチャル富岳」の実証・開発環境として整備された計算機システムであり、商用クラウド上に「富岳」と同等のソフトウェアスタックを構築している。富岳ユーザ向けの試行用共用環境として提供される。

- 「プライベート富岳」

  利用者が自ら購入したAWSクラウドインスタンスに、事前構築済みのソフトウェアスタックを含むコンテナイメージをインストールすることで、自由に構築できるプライベートな「富岳」環境である。

（２）「サテライト富岳」と「プライベート富岳」

「サテライト富岳」・「プライベート富岳」概要



（２）「サテライト富岳」と「プライベート富岳」

「サテライト富岳」

「サテライト富岳」とは、R-CCSが提唱する「バーチャル富岳」の実証・開発環境として整備された、商用クラウド上に「富岳」と同等のソフトウェアスタックを構築した計算機システムである。

- 特徴
  - 「富岳」アカウント所持者のみ利用可能な共用環境である。
  - 「富岳」のログインノードからのみ接続可能である。
  - 検証用のため、小規模な環境である。
- 用途
  - 「富岳」利用者へ提供する「バーチャル富岳」の試験環境

    「バーチャル富岳」を試してみたい「富岳」利用者の利用や、将来の「バーチャル富岳」へソフトウェアの提供を検討しているソフトウエア開発者の開発環境としての利用を想定している。また、「富岳」利用者に試験的に「バーチャル富岳」を使ってもらうことで、ソフトウェアの構成内容を見直し「バーチャル富岳」のアップデートを継続的に行う。

（２）「サテライト富岳」と「プライベート富岳」

「プライベート富岳」

利用者が自ら購入したクラウドインスタンスやコンピュータシステムに、「バーチャル富岳」を導入することで構築するプライベートな「富岳」環境である。だれでも「富岳」と同じ環境を自身のプライベートな環境に構築できる。また、利用にあたって成果公開の義務はなく、高い秘匿性が担保された環境を構築可能である。

- 特徴
  - 利用にあたって審査不要で、利用者が自身で準備した環境に利用者が自由に構築する。
  - 成果公開の義務はなく、秘匿性が担保された環境である。
- 用途
  - 秘匿性が高いシミュレーション環境

    最先端の研究では「富岳」を使用してシミュレーションを行い、その成果を活用した製品開発時には、秘匿性は担保された「プライベート富岳」を使用してシミュレーションを実施する。
  - 「富岳」での大規模計算の前の検証用環境

    「プライベート富岳」上の小規模な環境で基本となるプログラムの開発や動作検証を行い、その検証結果をもとに「富岳」を用いて大規模なシミュレーションを実施する。

（３）「バーチャル富岳」についての参考資料

参考資料

「バーチャル富岳」について詳しい情報は理化学研究所の下記HP、プレスリリースを参照方。

- 「バーチャル富岳」(R-CCS公式HP)
  URL : https://www.r-ccs.riken.jp/fugaku/virtual-fugaku/

- 「バーチャル富岳」初版の提供を開始 – 次世代計算基盤にもつながるエコシステム構築へ大きな一歩 –
  （2024年8月5日理化学研究所　プレスリリース）
  URL : https://www.riken.jp/pr/news/2024/20240805_1/index.html

問合先

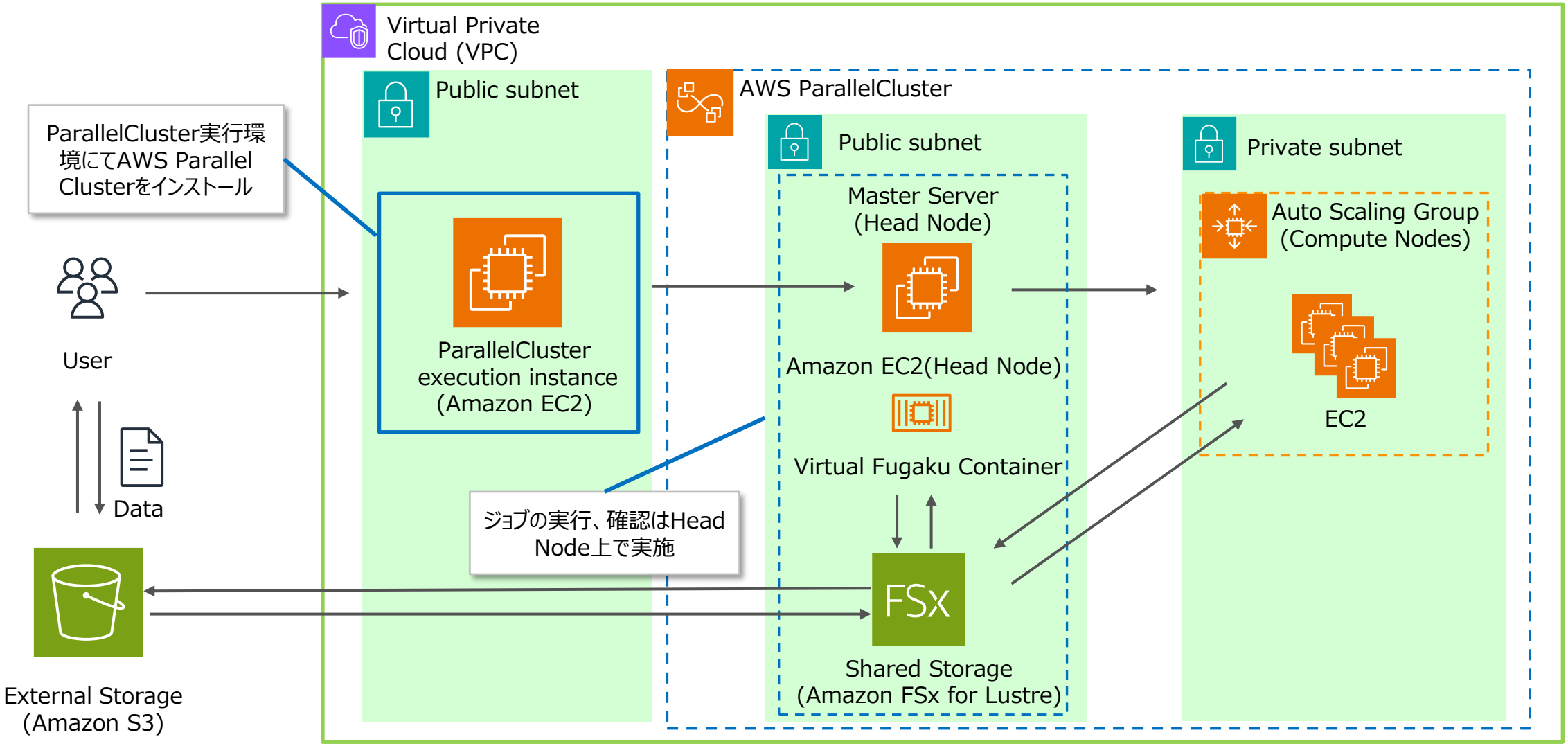
virtual-fugaku@ml.riken.jp

2.「プライベート富岳」の構築

（１）「プライベート富岳」の構成

「プライベート富岳」の基本構成例

現在、「バーチャル富岳」は、AWS が提供する Graviton3/3E ベースのインスタンス上で動作検証が行われている。本マニュアルでは、AWS クラウド環境における「プライベート富岳」の構築方法について、以下に示す構成を例に解説する。環境は任意のリージョンで構築可能である。本マニュアルでは東京リージョンを選択している。



「プライベート富岳」の構成要素

「プライベート富岳」は下記要素で構成される。

Head Node

膨大な計算リソースを備えた計算ノードにアクセスするための入り口であり、ユーザーがファイルの参照、ジョブの投入・状況確認、コードのコンパイルなどを実行する環境である。

計算ノード

長時間かかる計算や、大量の CPU やメモリを使用する計算を行うためのノードである。直接ログインすることはできず、ユーザーはログインノードからジョブスクリプトを経由してアプリケーションやプログラムを実行する。

「バーチャル富岳」コンテナ

「富岳」向けに整備されたソフトウェア群のうち、利用頻度の高いものを抽出し、それらのバイナリを一体化して Singularity コンテナイメージとして配布したもの。ユーザーは、自身が購入したクラウドインスタンスに「バーチャル富岳」コンテナイメージをダウンロードすることで、「富岳」と同等の環境を構築できる。

共有ストレージ

Head Node と計算ノードで共有されるストレージである。ジョブの入出力ファイルの格納場所として利用する。

外部ストレージ

データを長期的に保存するためのストレージである。AWS ParallelCluster で構築した環境は、削除すると環境内に保存されているデータも消去される。そのため、クラスタ内のファイルシステムに保存したデータを外部ストレージにエクスポートすることで、クラスタ削除後もデータを保持できる。

使用するAWSサービス

「プライベート富岳」では主に下記のAWSサービスを利用する。

Amazon VPC

AWS上に仮想ネットワークを構築するサービスである。「プライベート富岳」は、本サービスを利用して構築された仮想ネットワーク環境に構築される。

Amazon EC2

AWSが提供する仮想サーバーを構築できるサービスである。本サービスを利用して構築された仮想サーバ上にAWS ParallelClusterをインストールし、クラスターを構築する。

AWS ParallelCluster

オープンソースのクラスター管理ツールであり、簡単にハイパフォーマンスコンピューティング (HPC) クラスターをデプロイおよび管理できる。本ツールは「プライベート富岳」においてクラスタの構築に使用される。

使用するAWSサービス

「プライベート富岳」のストレージでは主に下記のAWSが提供するサービスを利用する。

AWS FSx for Lustre

AWSが提供するフルマネージド型の共有ストレージサービスである。ヘッドノードおよび計算ノードからデータの保存や削除が可能な共有ストレージとして使用する。

Amazon　S3

AWSが提供するオブジェクトストレージサービスの一種であり、外部ストレージとして利用する。クラスタ内の共有ストレージに保存されたデータを、Amazon S3で構築した外部ストレージにエクスポートすることで、クラスタ削除後もデータを保持できる。

使用するAWSサービス

各AWSのサービスについて詳細は下記URLを参照方。

- Amazon VPC

  URL : https://aws.amazon.com/jp/vpc/?nc2=h_ql_prod_fs_vpc

- Amazon EC2

  URL : https://aws.amazon.com/jp/ec2/?nc2=h_ql_prod_fs_ec2

- AWS ParallelCluster

  URL : https://aws.amazon.com/jp/hpc/parallelcluster/

- AWS FSx for Lustre

  URL : https://aws.amazon.com/jp/fsx/lustre/

- Amazon S3

  URL : https://aws.amazon.com/jp/pm/serv-s3/

2.「プライベート富岳」の構築
（２）クラスタ環境の構築

ネットワーク構築

ネットワーク構築の流れ

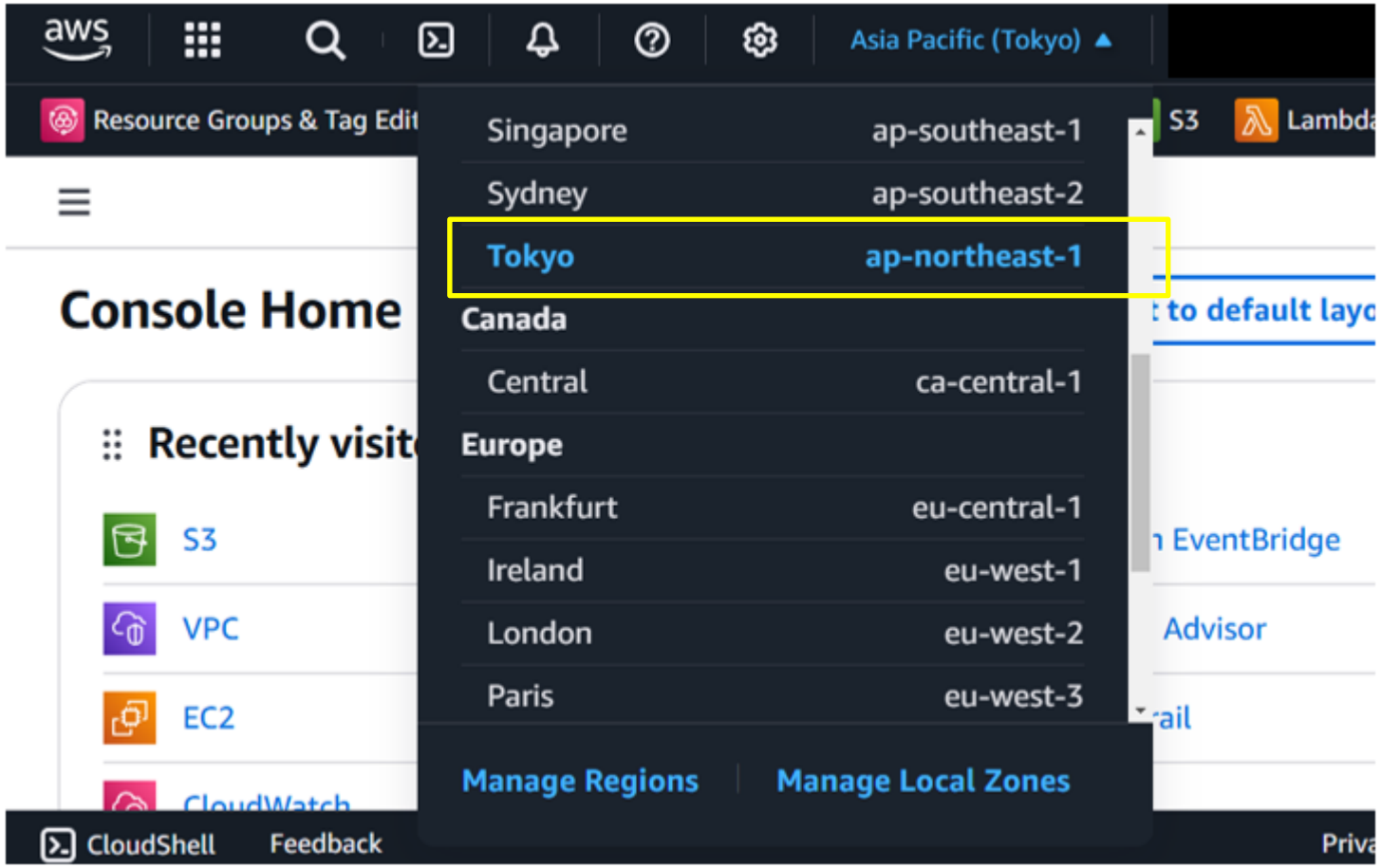
AWS ParallelCluster を用いてクラスタ環境を構築する際は、まずネットワークを設定する必要がある。本項では、クラスタ構築時のネットワーク設定方法について解説する。クラスタ構築に当たって必要なネットワーク環境は、以下の手順で構築する。次ページ以降では、それぞれの手順を詳しく説明する。

1．リージョンの選択
2．VPCの作成
3．DNSホスト名の有効化
4．パブリックサブネットの作成
5．インターネットゲートウェイの作成
6．インターネットゲートウェイのVPCへアタッチ
7．ルートテーブルの作成
8．ルートテーブルの関連付け
9．ルート設定の追加
10．キーペアの作成
11．セキュリティグループの作成

ネットワーク構築

1. リージョンの選択

1.1 AWSマネジメントコンソールの右上にあるプルダウンから「Tokyo ap-northeast-1」を選択する。



ネットワーク構築

2. VPCの作成(1/3)

2.1 VPC一覧ページへアクセスする。

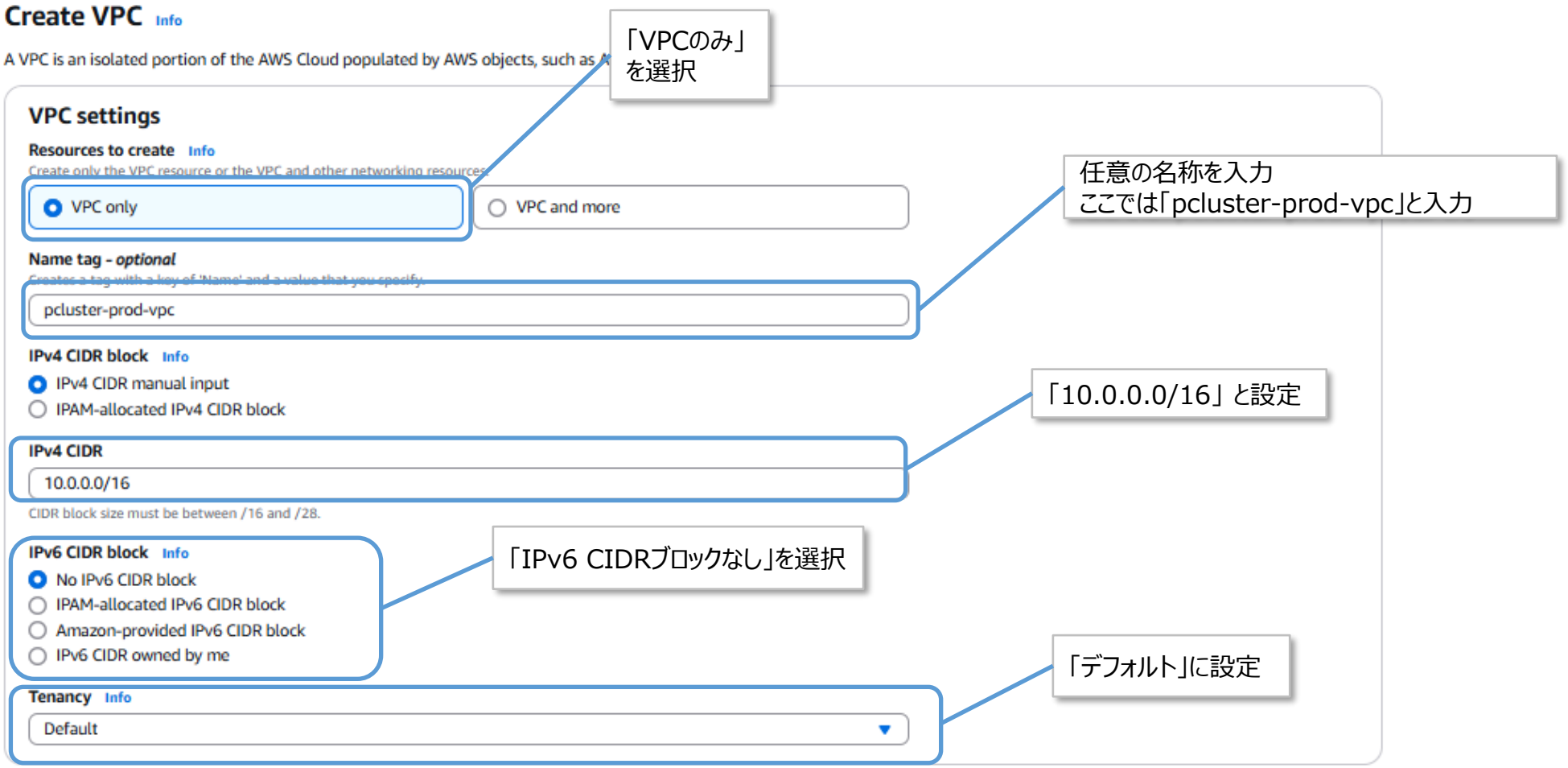
https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#vpcs

2.2 右上の「VPCを作成」をクリックする。

ネットワーク構築

2. VPCの作成(2/3)

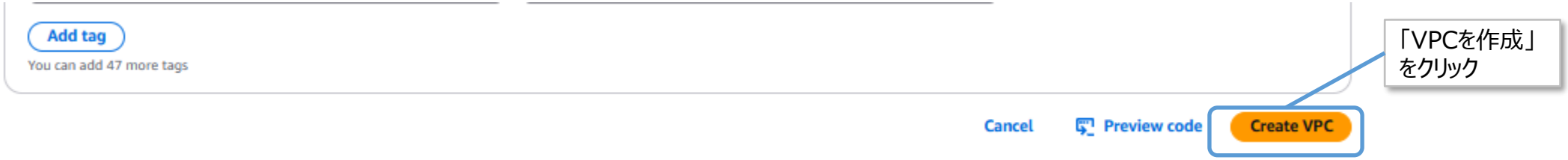
2.3 下記のようにパラメータを設定する。



ネットワーク構築

2. VPCの作成(3/3)

2.4 「VPCを作成」をクリックする。



2.5 下記の画面が表示されるとVPCの作成が完了する。

ネットワーク構築

3. DNSホスト名の有効化(1/2)

3.1 VPC一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#vpcs

3.2 作成したVPCを選択する。

3.3 右上の「アクション」=>「VPCの設定を編集」をクリックする。

ネットワーク構築

3. DNSホスト名の有効化(2/2)

3.4 「DNS解決を有効化」と「DNSホスト名を有効化」にチェックを入れ「保存」をクリックする。

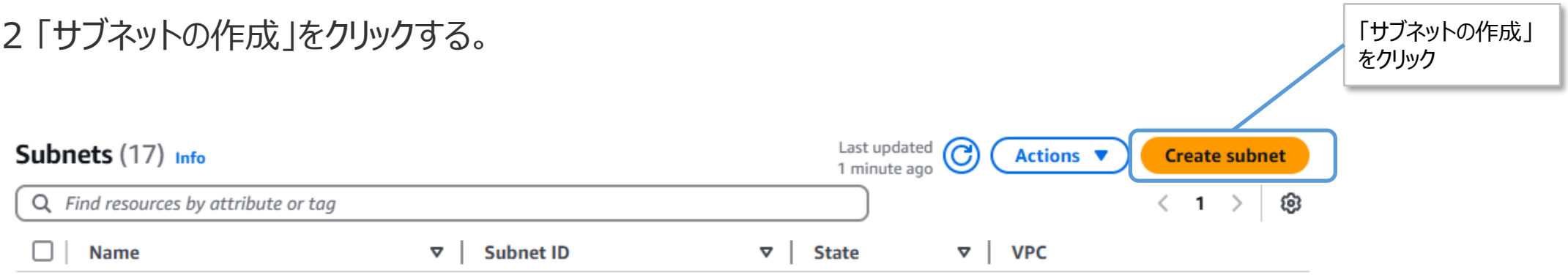
ネットワーク構築

4. パブリックサブネットの作成(1/3)

4.1 サブネット一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#subnets

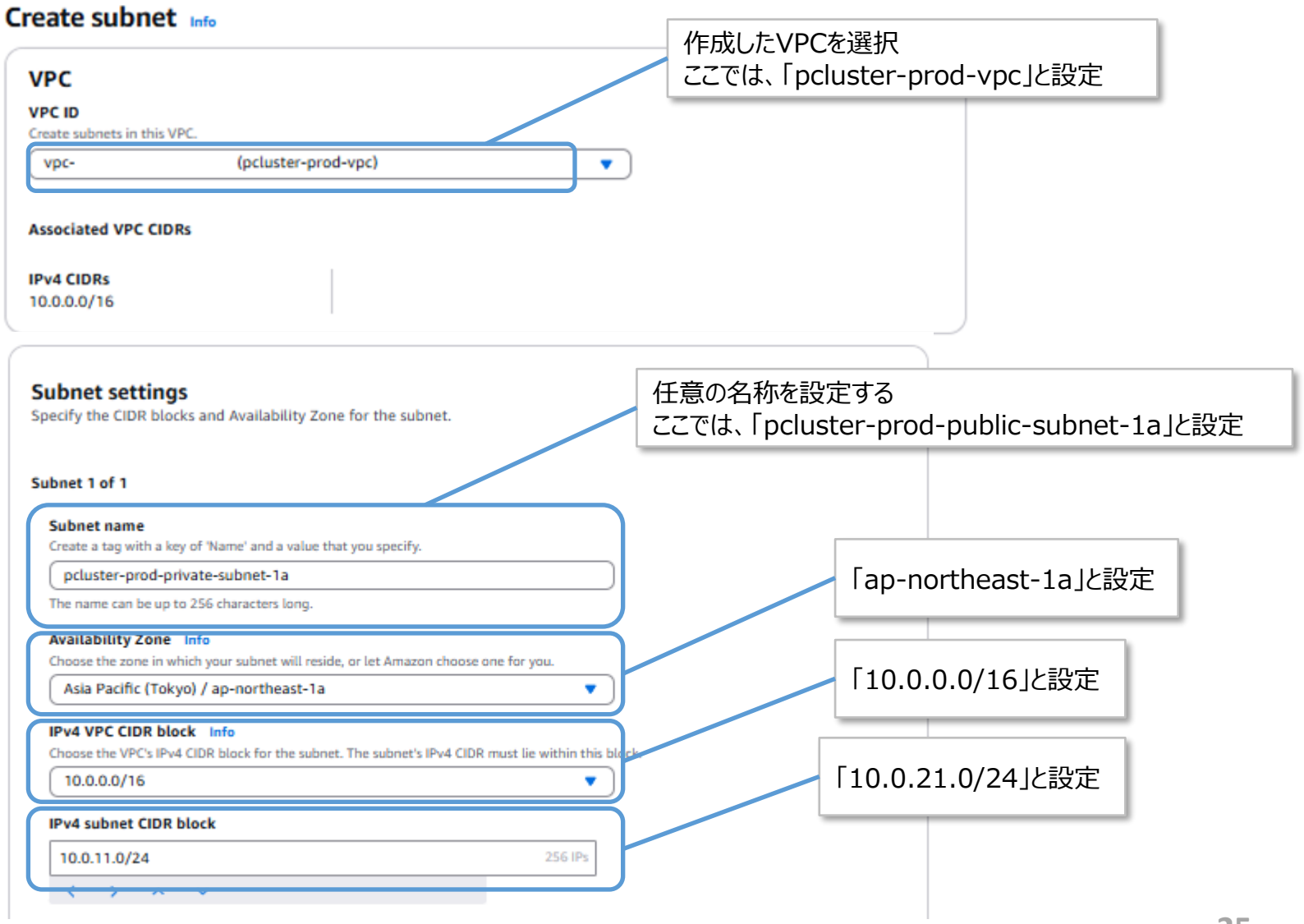
4.2 「サブネットの作成」をクリックする。



ネットワーク構築

4. パブリックサブネットの作成(2/3)

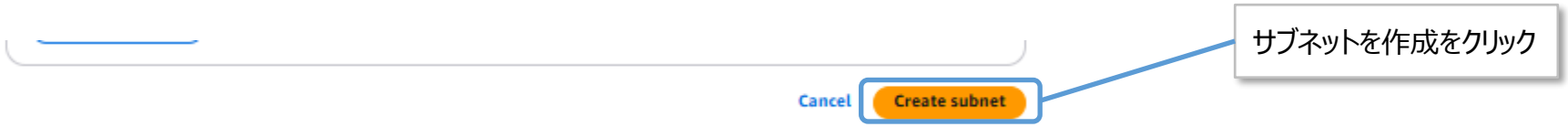
4.3 下記のようにパラメータを設定する。



ネットワーク構築

4. パブリックサブネットの作成(3/3)

4.4 「サブネットを作成」をクリックする。



4.5 下記の画面が表示されるとサブネットの作成が完了する。

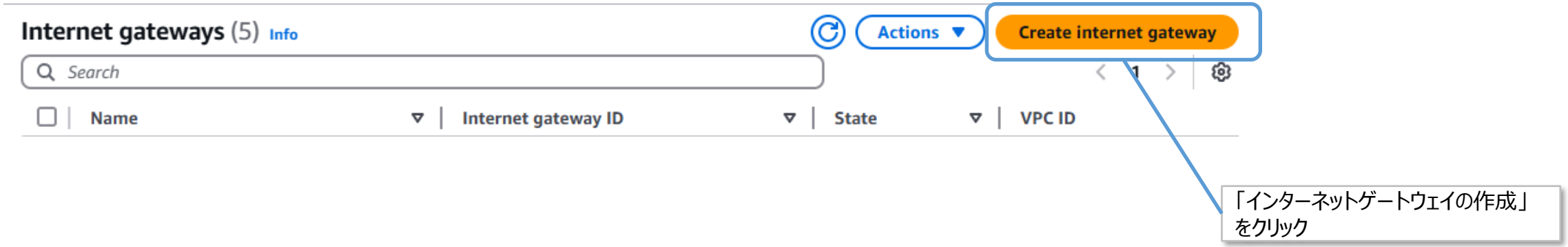
ネットワーク構築

5. インターネットゲートウェイの作成(1/2)

5.1 インターネットゲートウェイ一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#igws

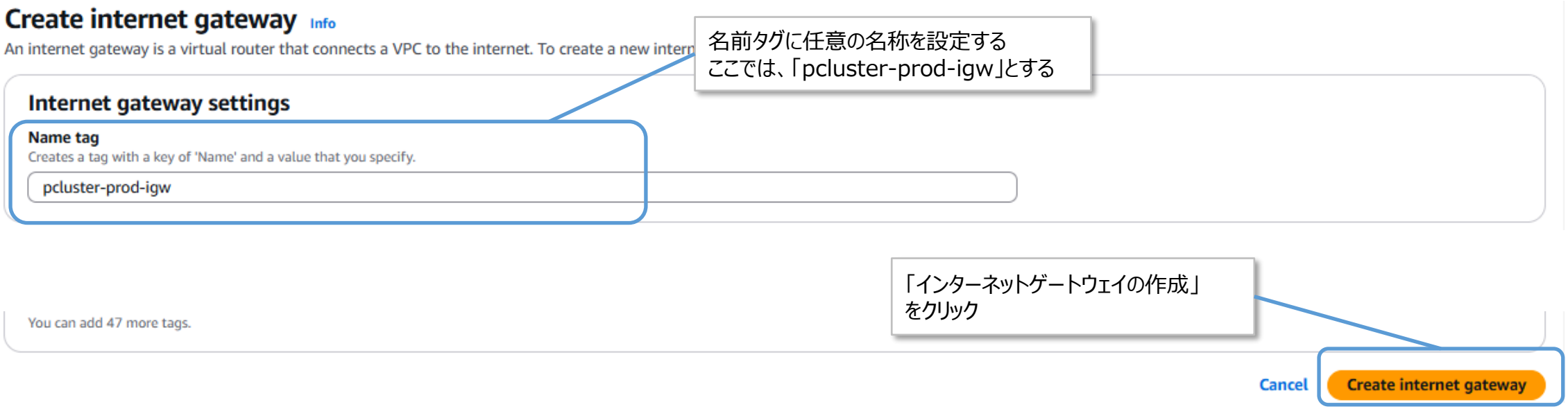
5.2「インターネットゲートウェイの作成」をクリックする。



ネットワーク構築

5. インターネットゲートウェイの作成(2/2)

5.3 下記のように名前タグに名称を設定し、「インターネットゲートウェイの作成」をクリックする。



5.4 画面に下記メッセージが表示されるとインターネットゲートウェイの作成が完了する。

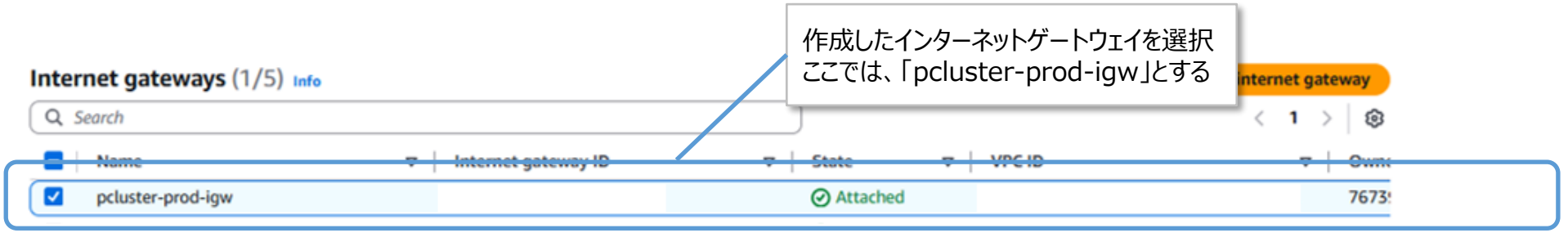
ネットワーク構築

6. インターネットゲートウェイのVPCへのアタッチ(1/2)

6.1 インターネットゲートウェイ一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#igws

6.2 作成したインターネットゲートウェイを選択する。

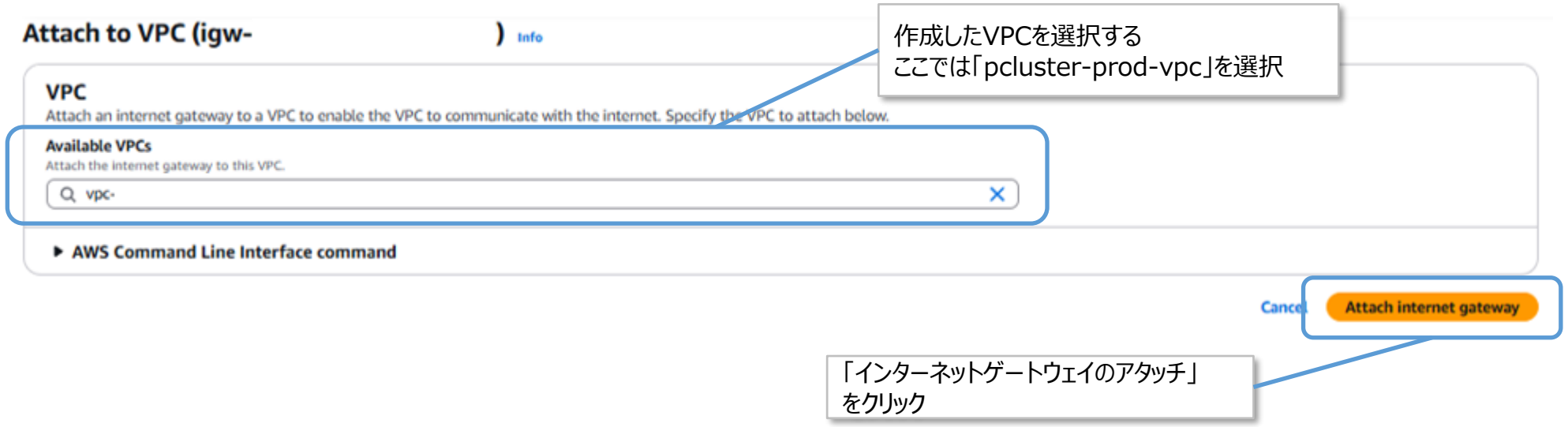


6.3 右上の「アクション」=> 「VPCにアタッチ」をクリックする。

ネットワーク構築

6. インターネットゲートウェイのVPCへのアタッチ(2/2)

6.4 「使用可能なVPC」のプルダウンから作成したVPCを選択し、「インターネットゲートウェイのアタッチ」をクリックする。



6.5 画面に下記メッセージが表示されるとインターネットゲートウェイのVPCへのアタッチが完了する。

ネットワーク構築

7. ルートテーブルの作成(1/3)

7.1 ルートテーブル一覧ページへアクセスする。

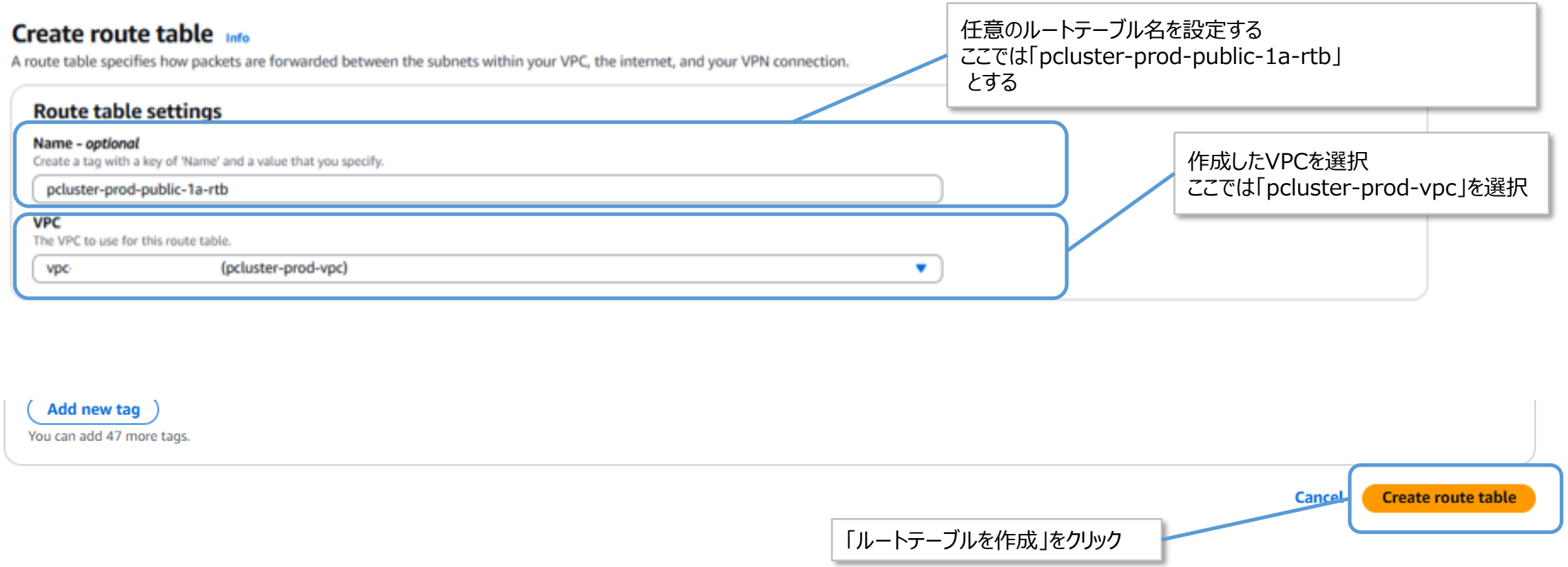
https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#RouteTables

7.2 「ルートテーブルを作成」をクリックする。

ネットワーク構築

7. ルートテーブルの作成(2/3)

7.3 下記のようにパラメータを設定し、「ルートテーブルを作成」をクリックする。



ネットワーク構築

7. ルートテーブルの作成(3/3)

7.4 画面に下記メッセージが表示されるとルートテーブルの作成が完了する。

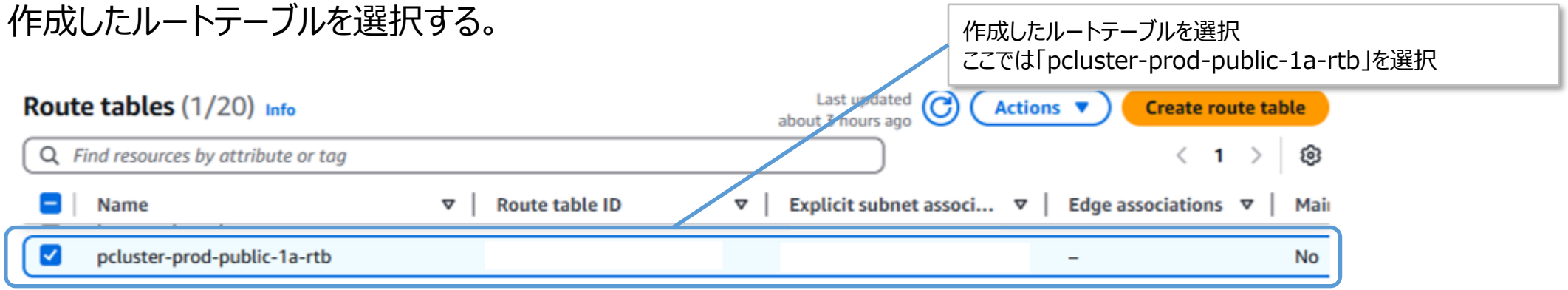
ネットワーク構築

8. ルートテーブルの関連付け(1/2)

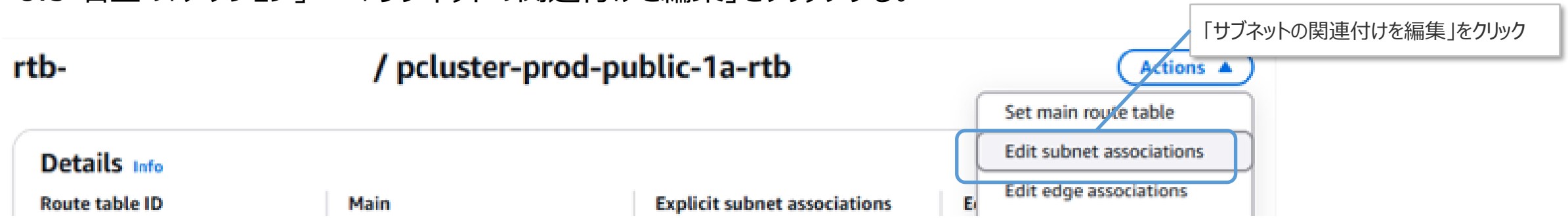
8.1 ルートテーブル一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#RouteTables

8.2 作成したルートテーブルを選択する。



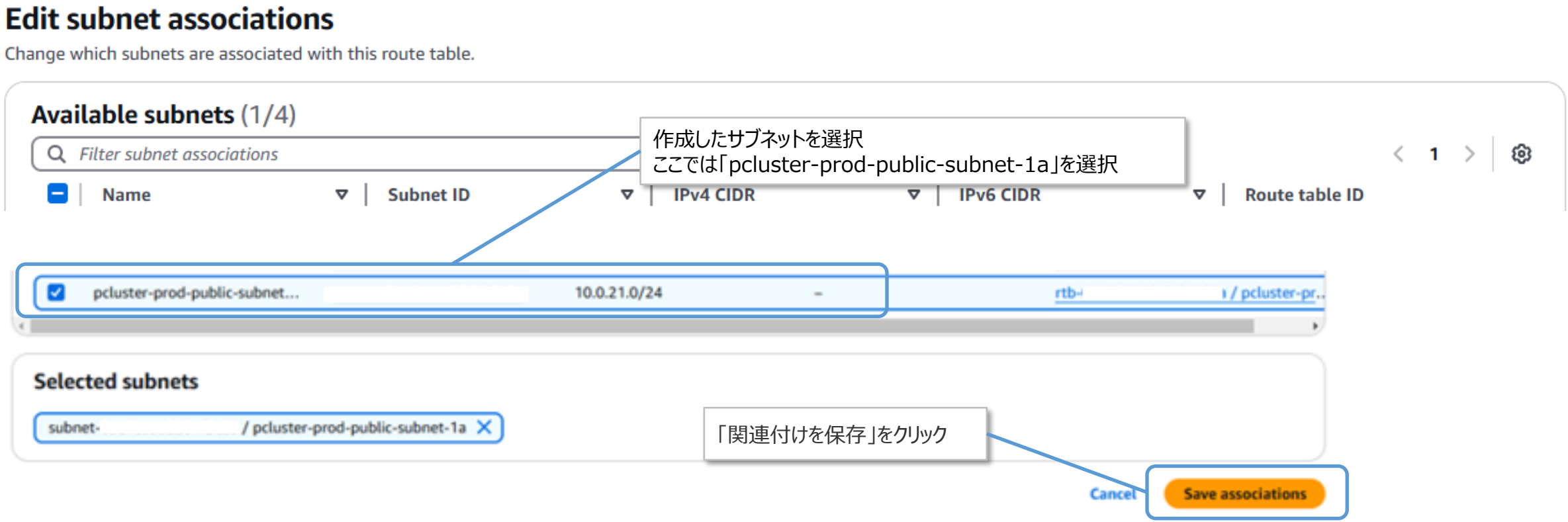
8.3 右上の「アクション」=>「サブネットの関連付けを編集」をクリックする。



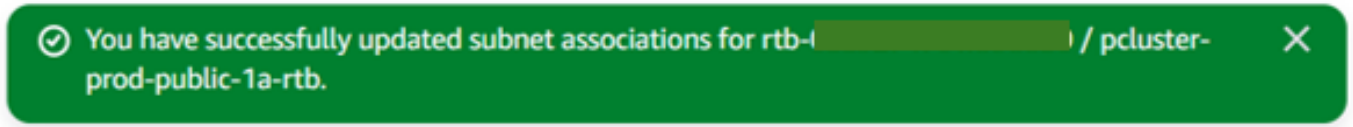
ネットワーク構築

8. ルートテーブルの関連付け(2/2)

8.4 作成したサブネットを選択し、「関連付けを保存」をクリックする。



8.5 画面に下記メッセージが表示されるとルートテーブルの関連付けが完了する。



ネットワーク構築

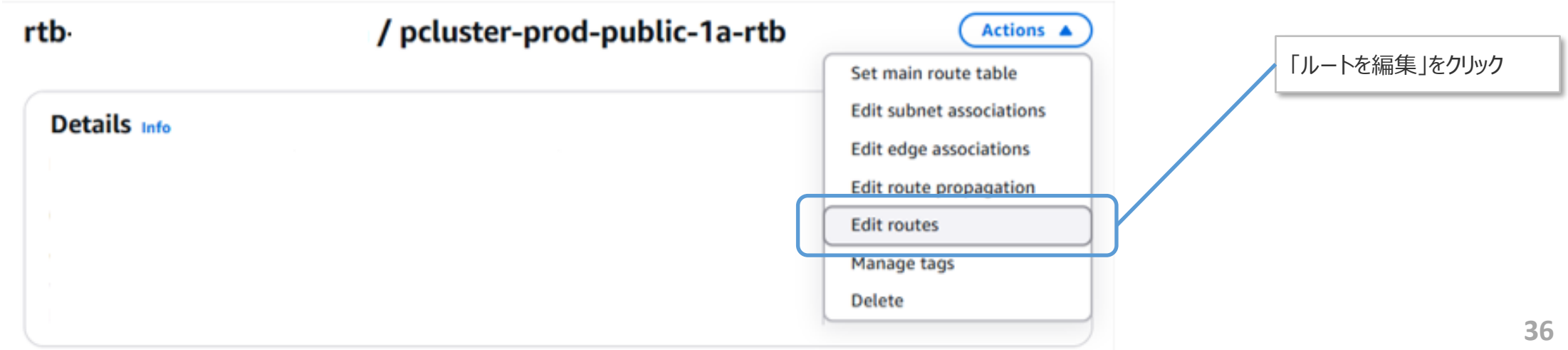
9. ルート設定の追加(1/3)

9.1 ルートテーブル一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#RouteTables

9.2 作成したルートテーブルを選択する。

9.3 右上の「アクション」=>「ルートを編集」をクリックする。

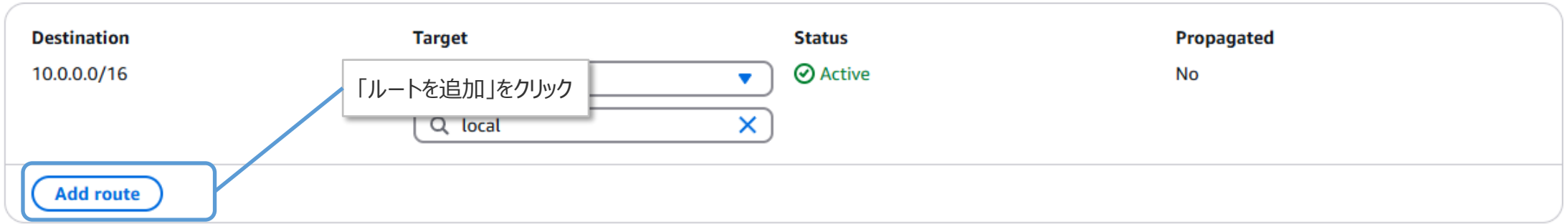


ネットワーク構築

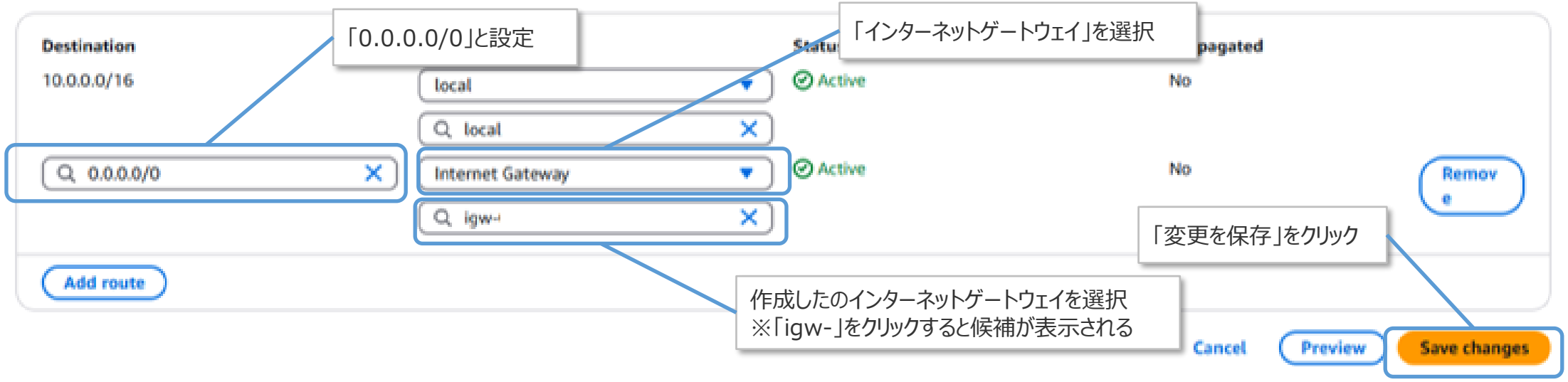
9. ルート設定の追加(2/3)

9.4 「ルートを追加」をクリックする。

Edit routes



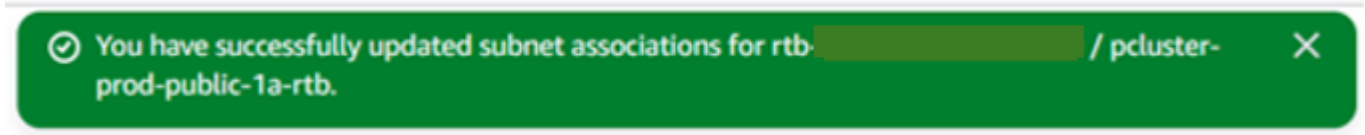
9.5 下記のようにパラメータを設定し、「変更を保存」をクリックする。



ネットワーク構築

9. ルート設定の追加(3/3)

9.6 画面に下記メッセージが表示されるとルート設定の追加が完了する。



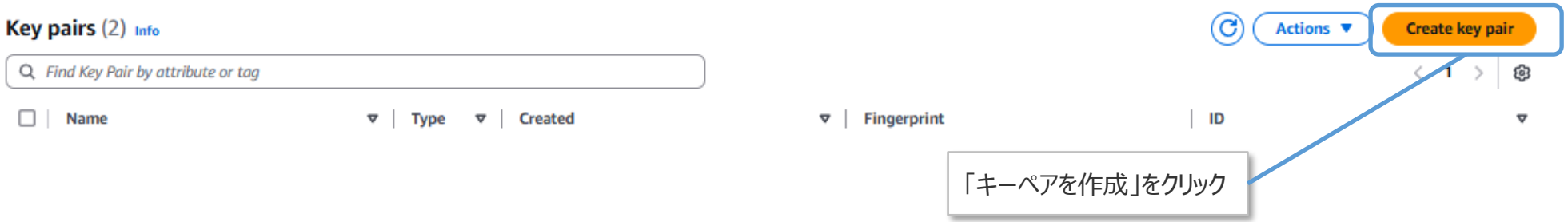
ネットワーク構築

10. キーペアの作成(1/3)

10.1 キーペア一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#KeyPairs

10.2 「キーペアを作成」をクリックする。



ネットワーク構築

10. キーペアの作成(2/3)

10.3 下記のようにパラメータを設定する。

ネットワーク構築

10. キーペアの作成(3/3)

10.4 「キーペアを生成」をクリックする。

「キーペアを生成」をクリック後、生成された秘密鍵が自動的にダウンロードされる。

10.5 画面に下記メッセージが表示されるとキーペアを生成が完了する。

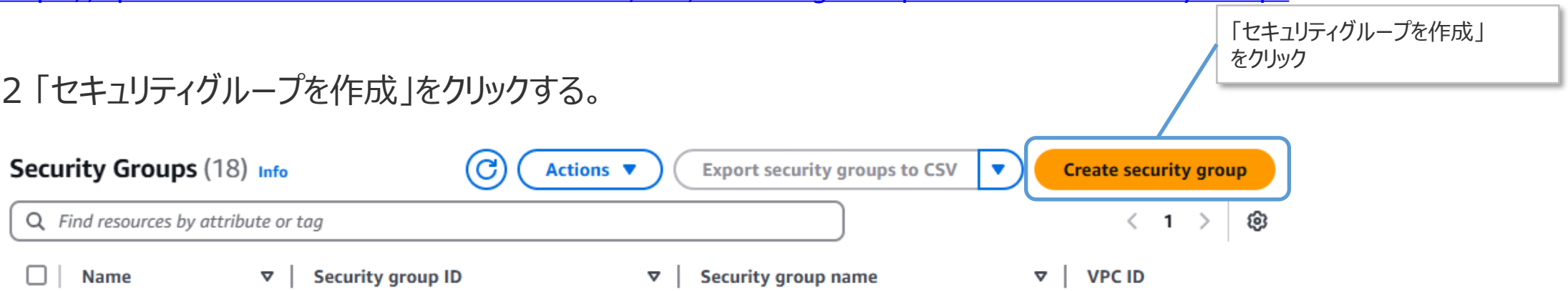
ネットワーク構築

11. セキュリティグループの作成(1/4)

11.1 セキュリティグループ一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#SecurityGroups

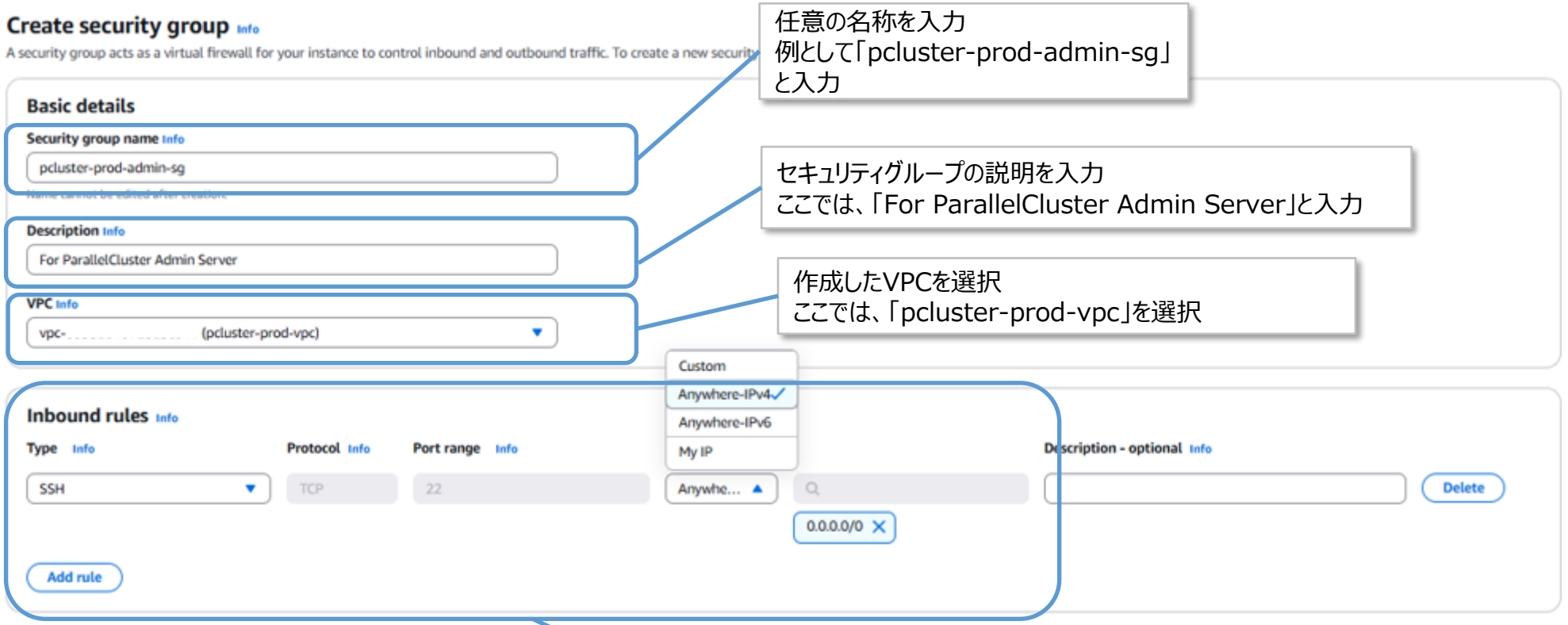
11.2 「セキュリティグループを作成」をクリックする。



ネットワーク構築

11. セキュリティグループの作成(2/4)

11.3 下記のようにパラメータを設定する



任意の名称を入力
例として「pcluster-prod-admin-sg」と入力

セキュリティグループの説明を入力
ここでは、「For ParallelCluster Admin Server」と入力

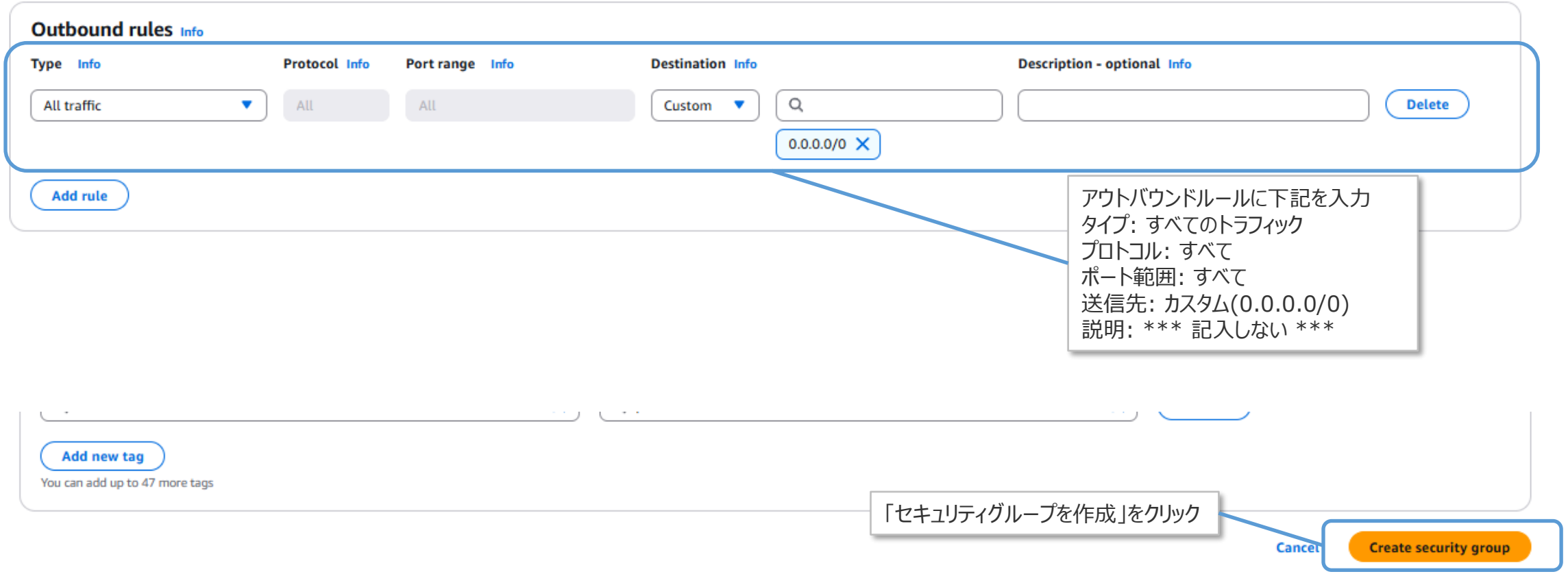
作成したVPCを選択
ここでは、「pcluster-prod-vpc」を選択

インバウンドルールに下記を入力
タイプ: SSH
プロトコル: TCP
ポート範囲: 22
ソース: Anywhere-IPv4(0.0.0.0/0)
説明: *** 記入しない ***

ネットワーク構築

11. セキュリティグループの作成(3/4)

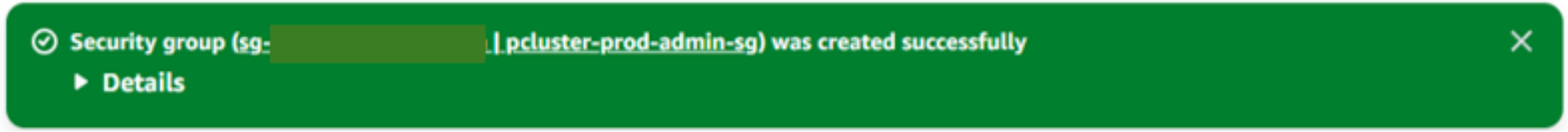
11.4 下記のようにパラメータを設定し「セキュリティグループを作成」をクリックする。



ネットワーク構築

11. セキュリティグループの作成(4/4)

11.5 画面に下記メッセージが表示されるとセキュリティグループの作成が完了する。



ParallelCluster実行用インスタンスの作成

ParallelCluster実行用インスタンスの作成の流れ

本項では、ParallelCluster 実行用インスタンスの作成手順について解説する。本マニュアルでは、ParallelCluster 実行用インスタンス上で、AWS ParallelCluster のインストール、セットアップ、クラスタの Head Node へのログインなどの作業を行う。 ParallelCluster実行用インスタンスは、以下の手順で構築する。次ページ以降では、それぞれの項目について詳しく解説する。

1．EC2インスタンスの作成
2．固定IPアドレスの発行
3．固定IPアドレスの関連付け
4．アクセスキーの作成
5．ParallelCluster実行用インスタンスへのログイン

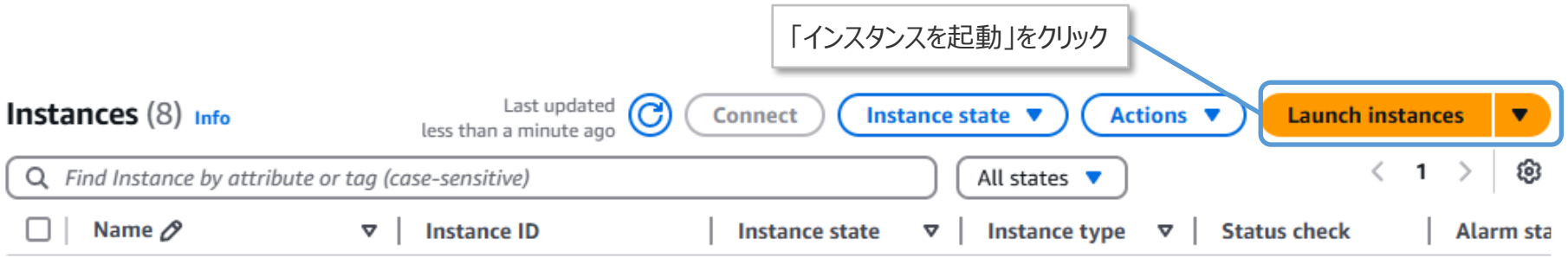
ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(1/8)

1.1 セキュリティグループ一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#Instances

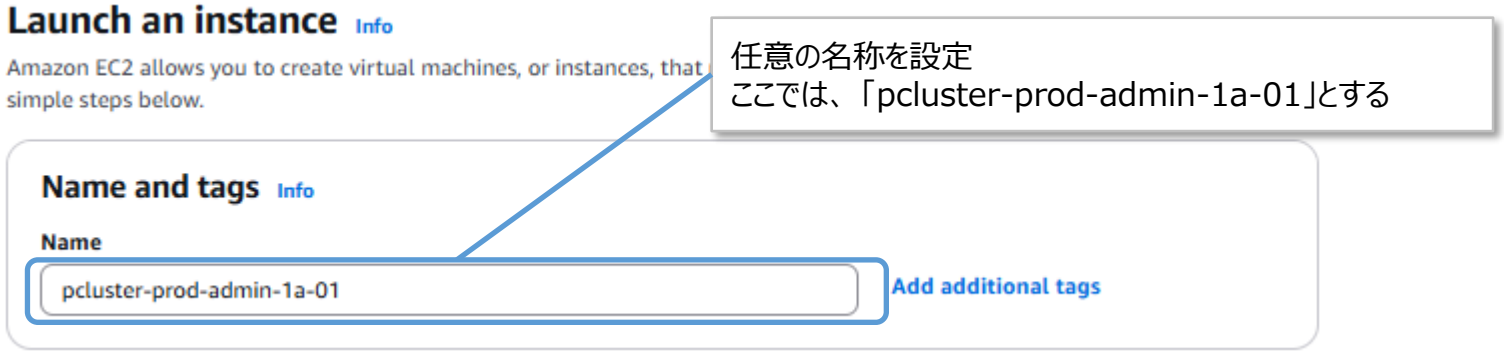
1.2 右上の「インスタンスを起動」をクリックする。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(2/8)

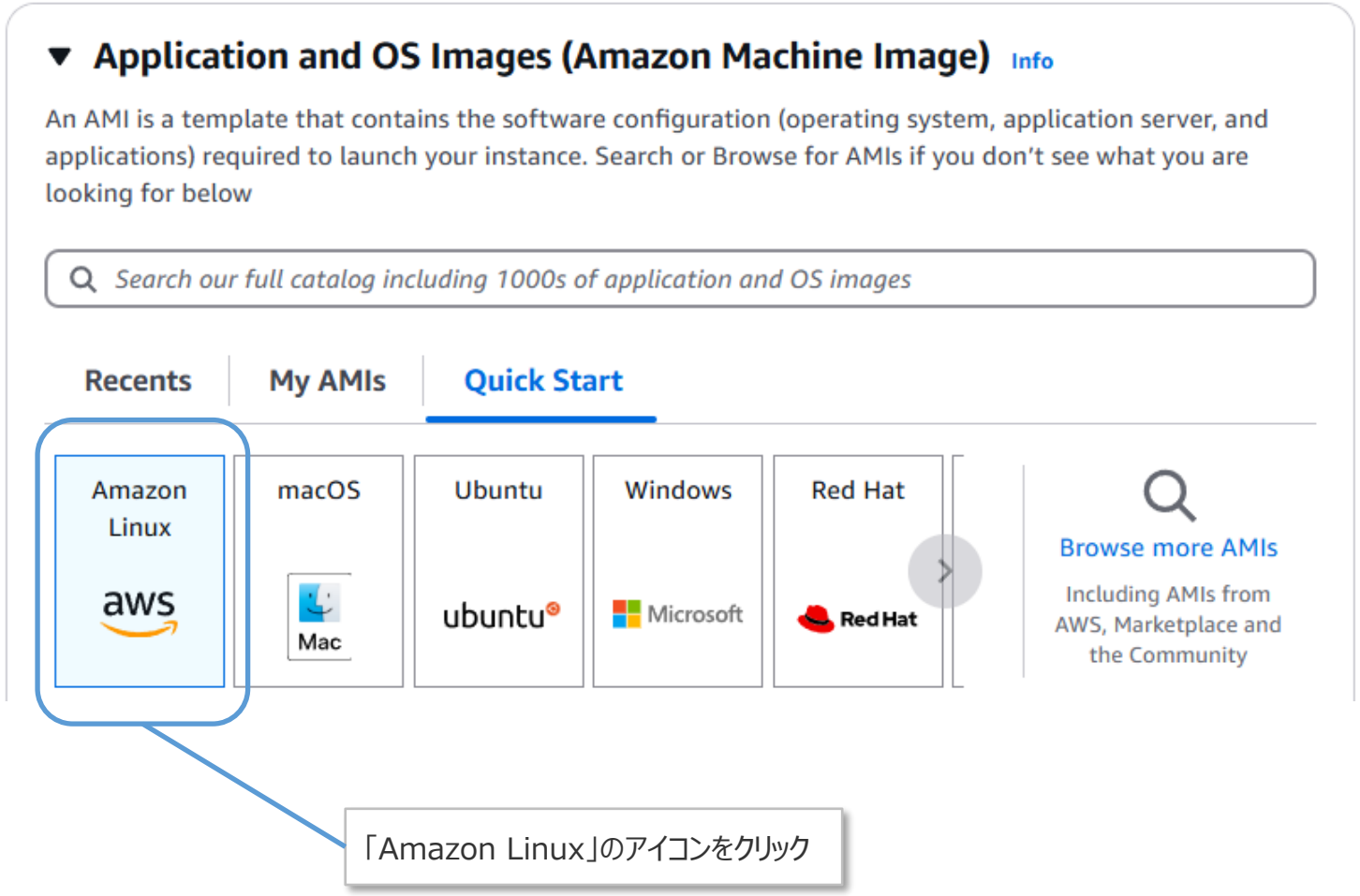
1.3 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(3/8)

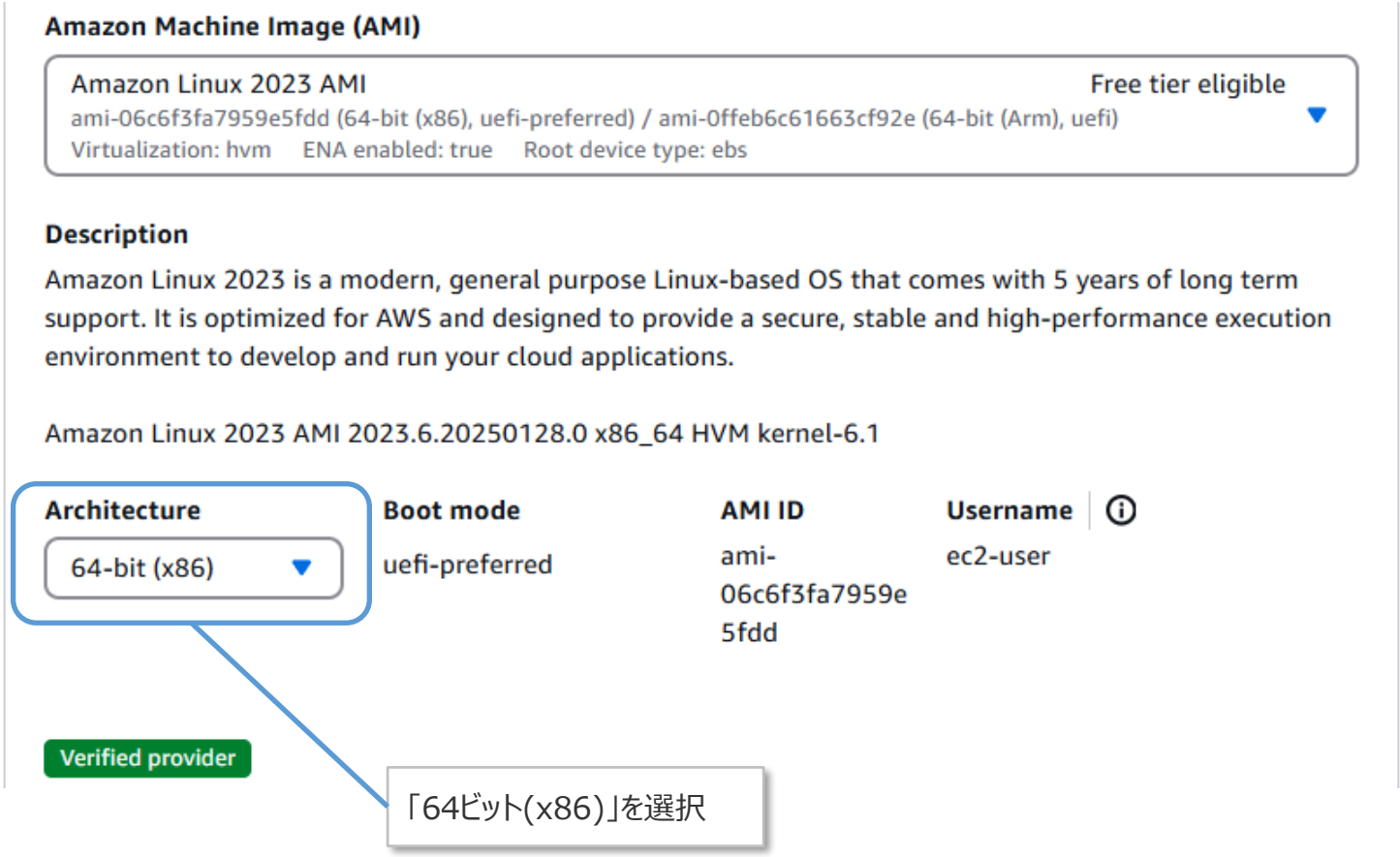
1.4 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(4/8)

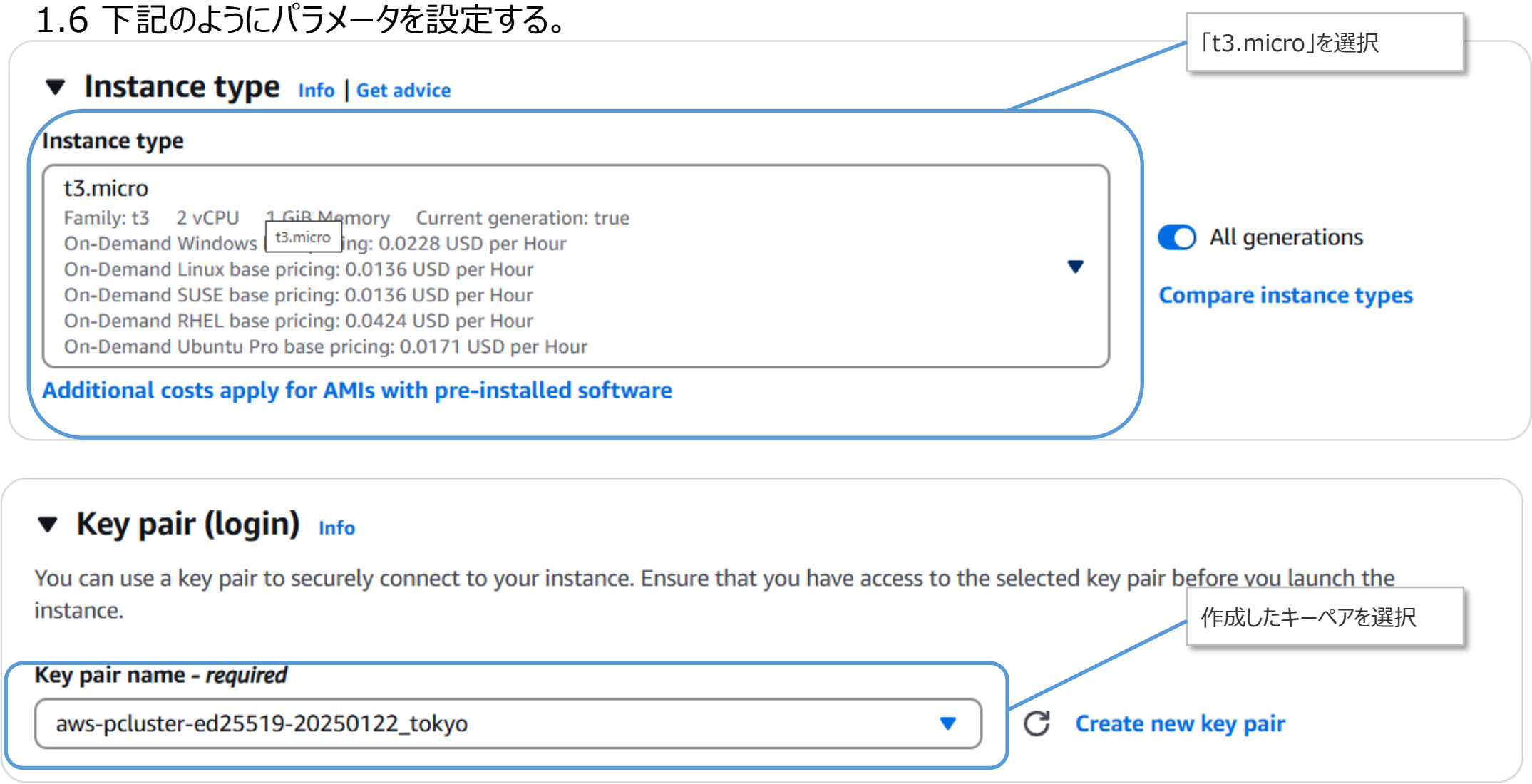
1.5 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(5/8)

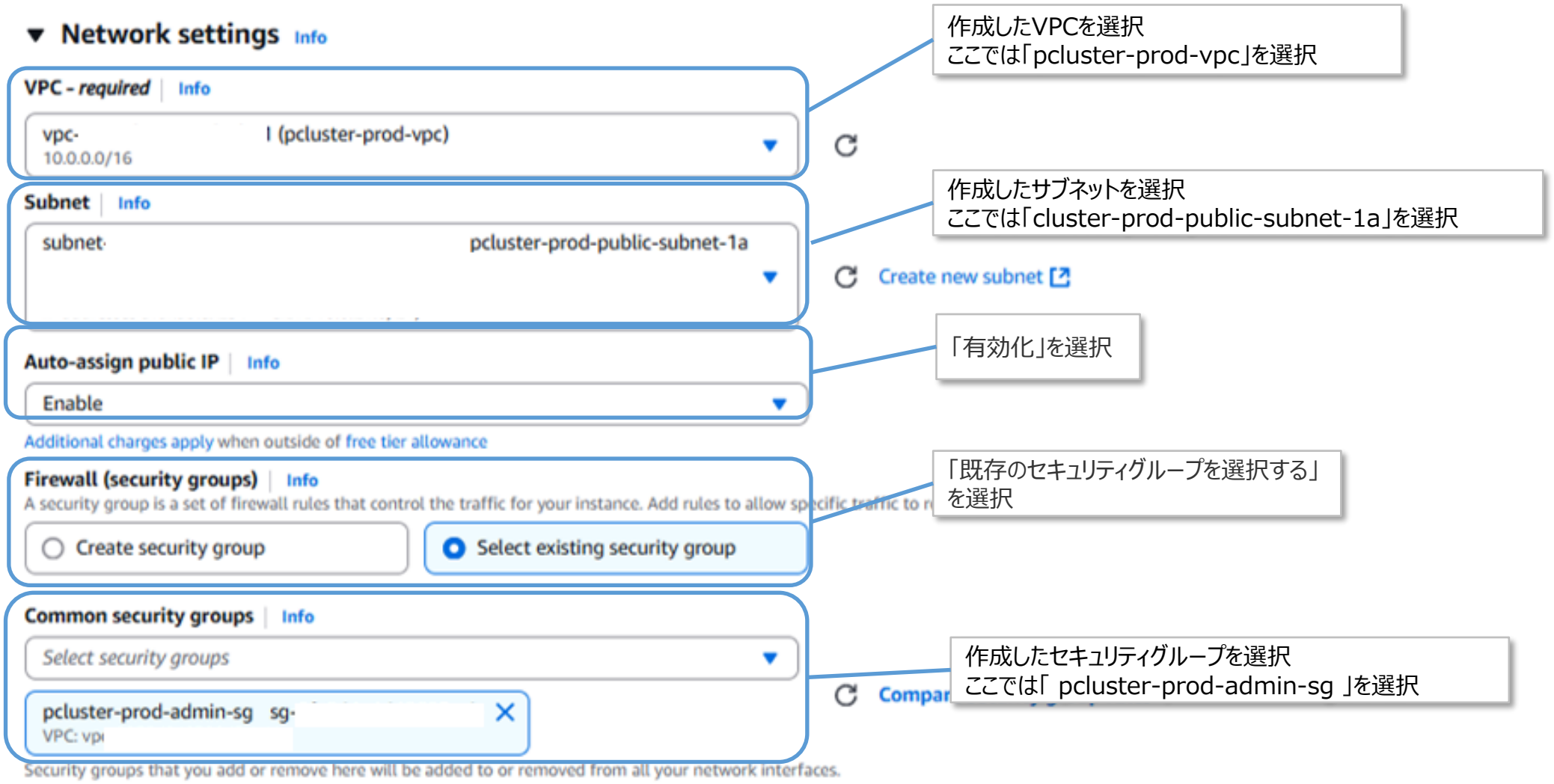
1.6 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(6/8)

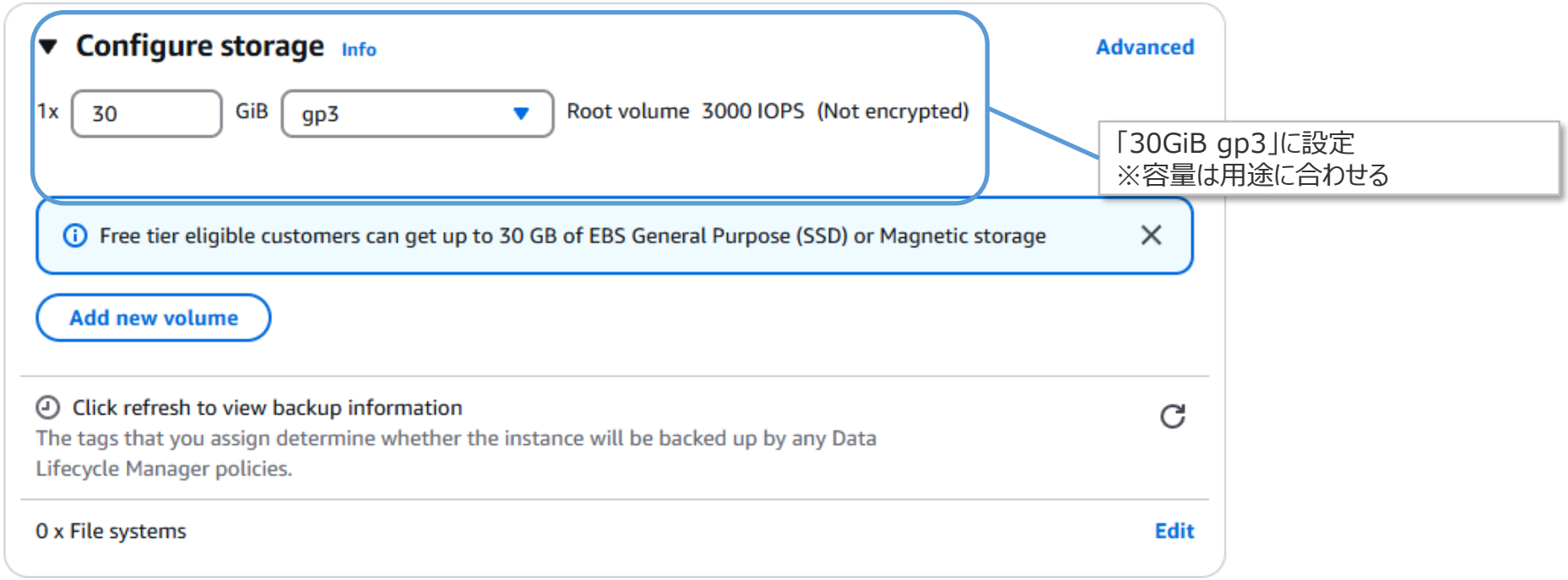
1.7 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(7/8)

1.8 下記のようにパラメータを設定する。



1.9 右側の「インスタンスを起動」をクリックする。

ParallelCluster実行用インスタンスの作成

1. EC2インスタンスの作成(8/8)

1.10 EC2インスタンス一覧ページへ戻り、数分経過後に作成中のEC2インスタンスの「ステータスチェック」が「3/3のチェックに合格しました」になることを確認する。

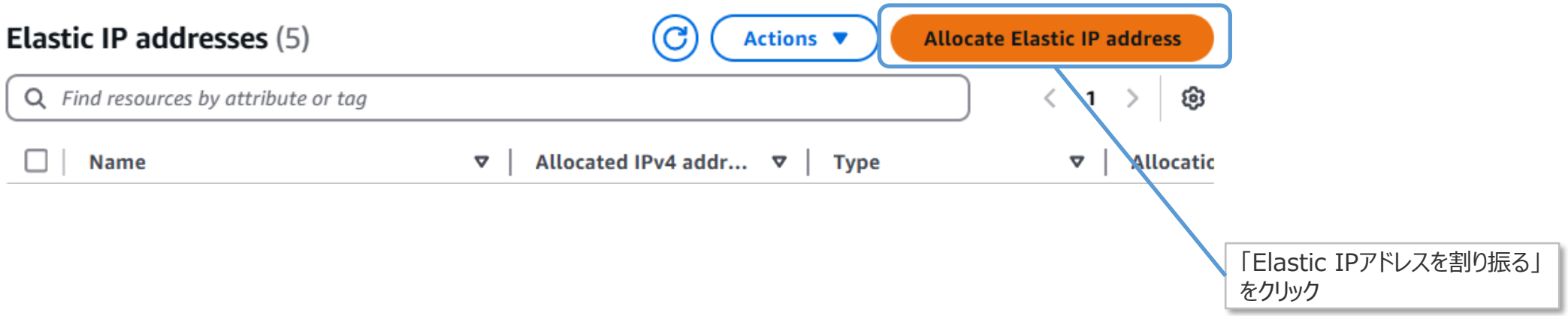
ParallelCluster実行用インスタンスの作成

2. 固定IPアドレスの発行(1/3)

2.1 Elastic IPアドレスを発行するため、EIP一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#Addresses

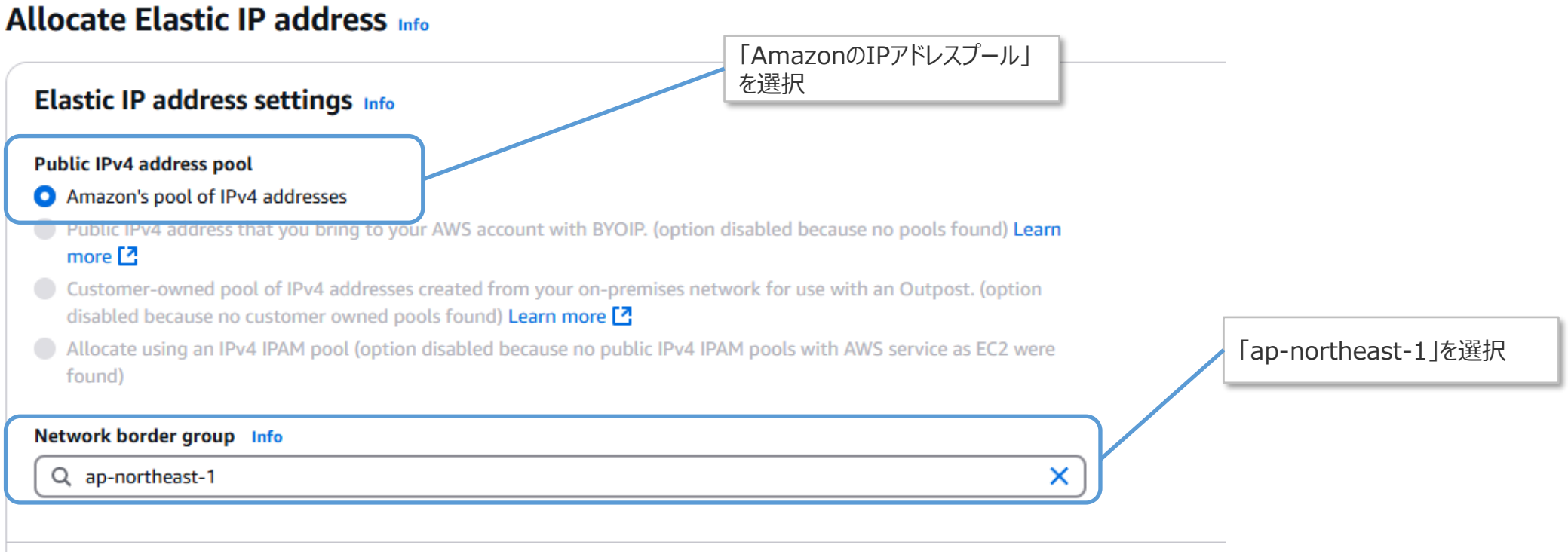
2.2 右上の「Elastic IPアドレスを割り振る」をクリックする。



ParallelCluster実行用インスタンスの作成

2. 固定IPアドレスの発行(2/3)

2.3 下記のようにパラメータを設定する。



ParallelCluster実行用インスタンスの作成

2. 固定IPアドレスの発行(3/3)

2.4 右下の「割り振る」をクリック。

2.5 画面に下記メッセージが表示されるとElastic IPアドレスを発行が完了する。

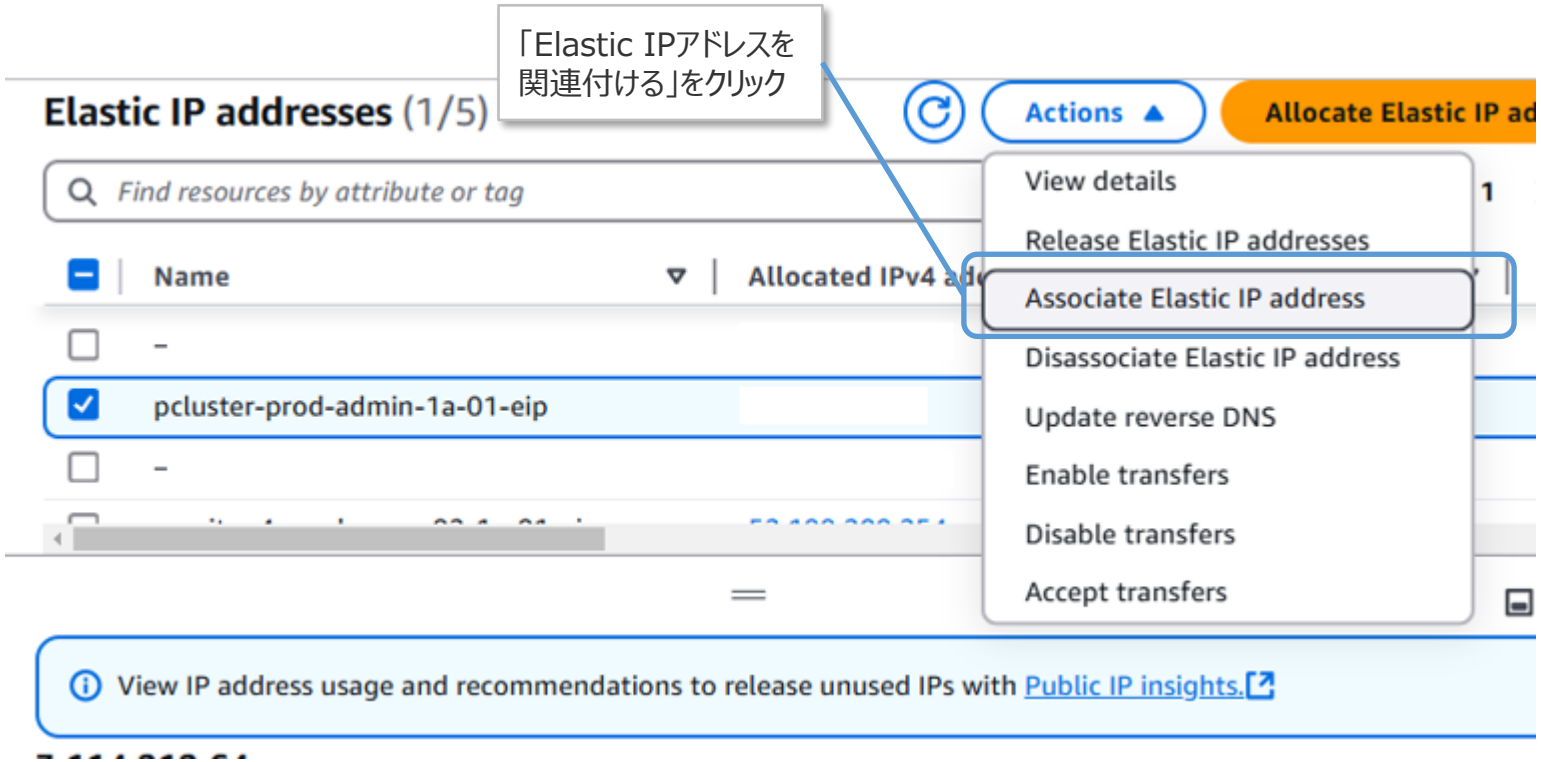
ParallelCluster実行用インスタンスの作成

3. 固定IPアドレスの関連付け(1/2)

3.1 EIP一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#Addresses

3.2 EC2インスタンスに割り当てするEIPを選択し、右上の「アクション」=>「Elastic IPアドレスを関連付ける」をクリックする。



ParallelCluster実行用インスタンスの作成

3. 固定IPアドレスの関連付け(2/2)

3.3 下記のようにパラメータを設定し、「関連付ける」をクリックする。

3.4 画面に下記メッセージが表示されるとElastic IPアドレスの関連付けが完了する。

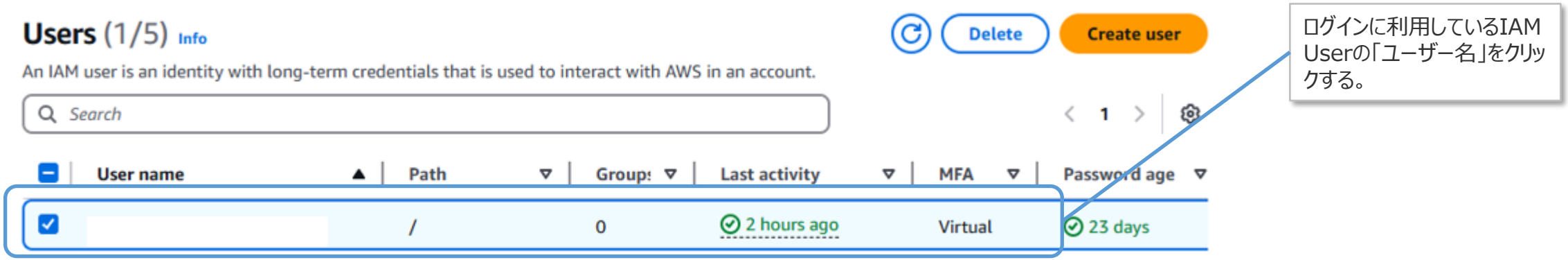
ParallelCluster実行用インスタンスの作成

4. アクセスキーの作成(1/4)

4.1 IAM User一覧ページへアクセスする。

https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-1#/users

4.2 自身がAWSマネジメントコンソールのログインに利用しているIAM Userの「ユーザー名」をクリックする。

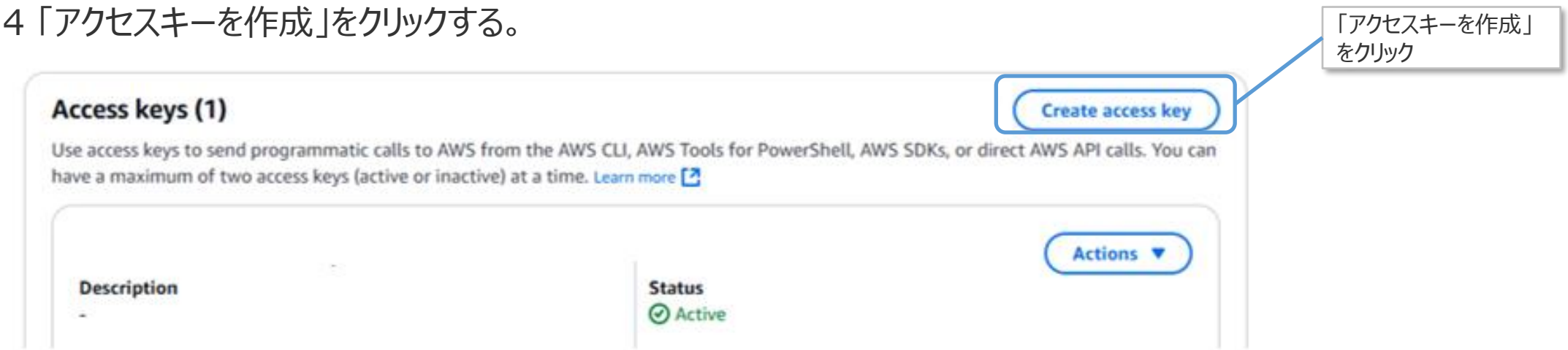


ParallelCluster実行用インスタンスの作成

4. アクセスキーの作成(2/4)

4.3「セキュリティ認証情報」をクリックする。

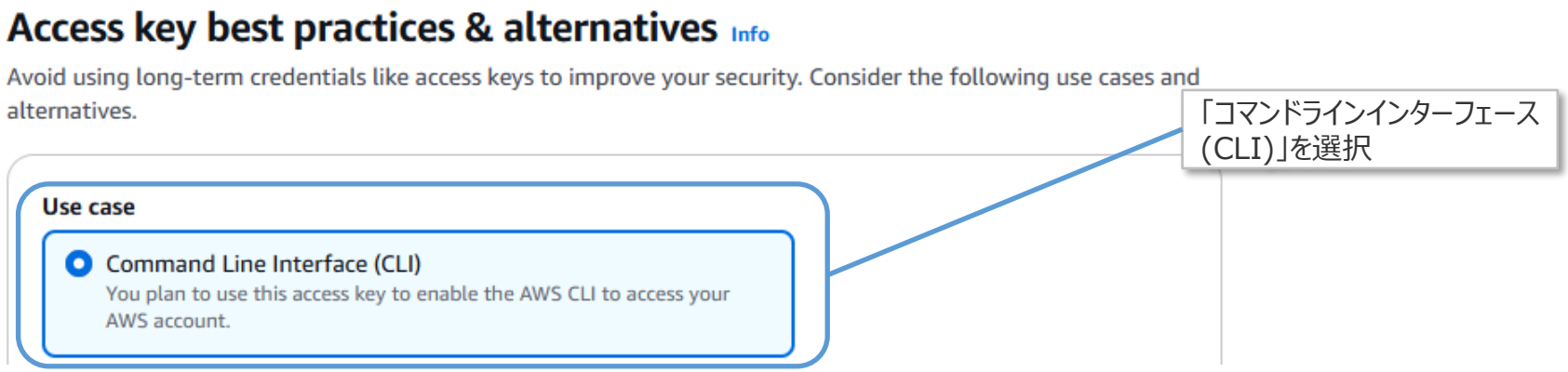
4.4「アクセスキーを作成」をクリックする。



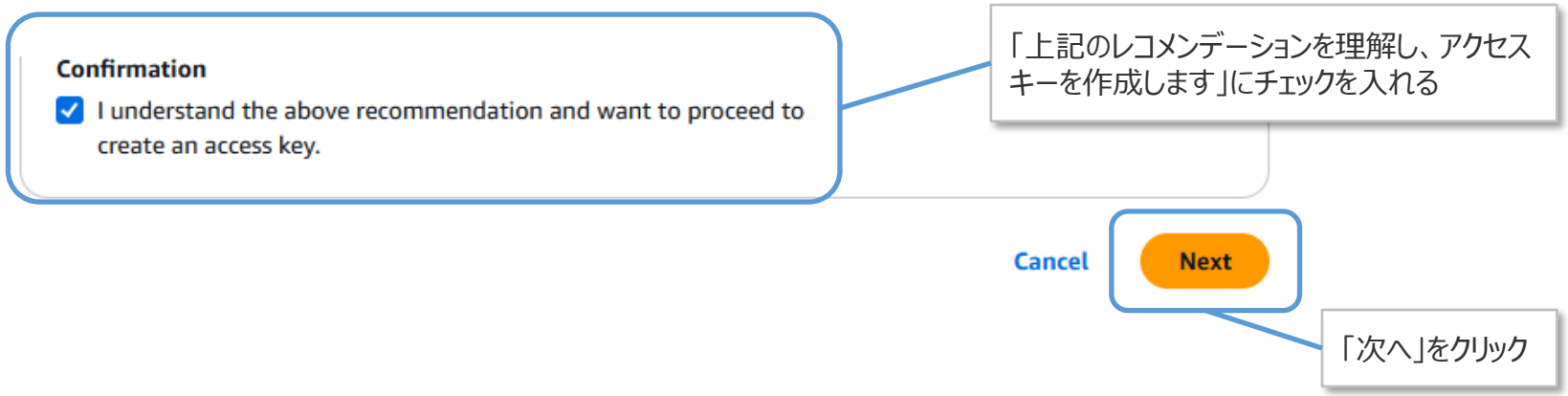
ParallelCluster実行用インスタンスの作成

4. アクセスキーの作成(3/4)

4.5「コマンドラインインターフェース(CLI)」を選択する。



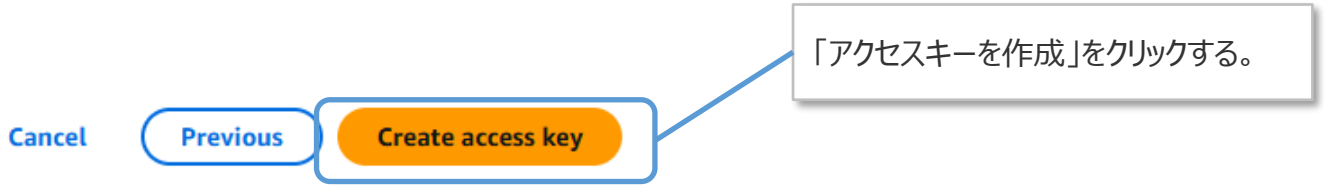
4.6「上記のレコメンデーションを理解し、アクセスキーを作成します」にチェックを入れ、「次へ」をクリックする。



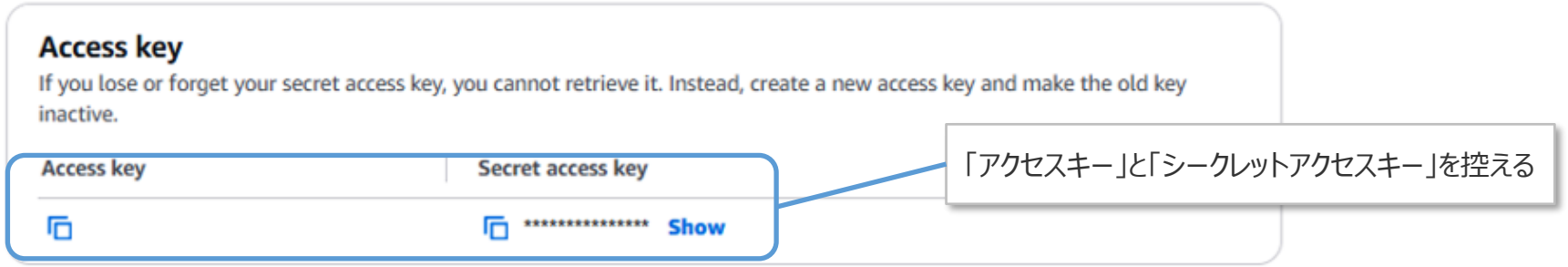
ParallelCluster実行用インスタンスの作成

4. アクセスキーの作成(4/4)

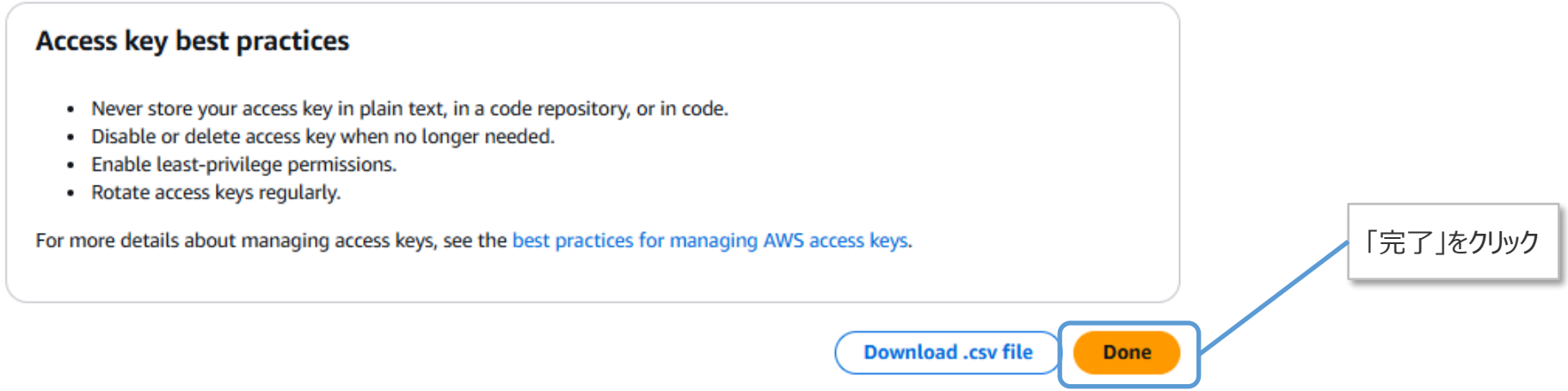
4.7 「アクセスキーを作成」をクリックする。



4.8 「アクセスキー」と「シークレットアクセスキー」を控えておく。「シークレットアクセスキー」はマスクされているので、「表示」をクリックして文字列を表示し、その文字列をシークレットアクセスキーとして控えておく。



4.9 「完了」をクリックする。



ParallelCluster実行用インスタンスの作成

ParallelCluster実行用インスタンスへのログイン

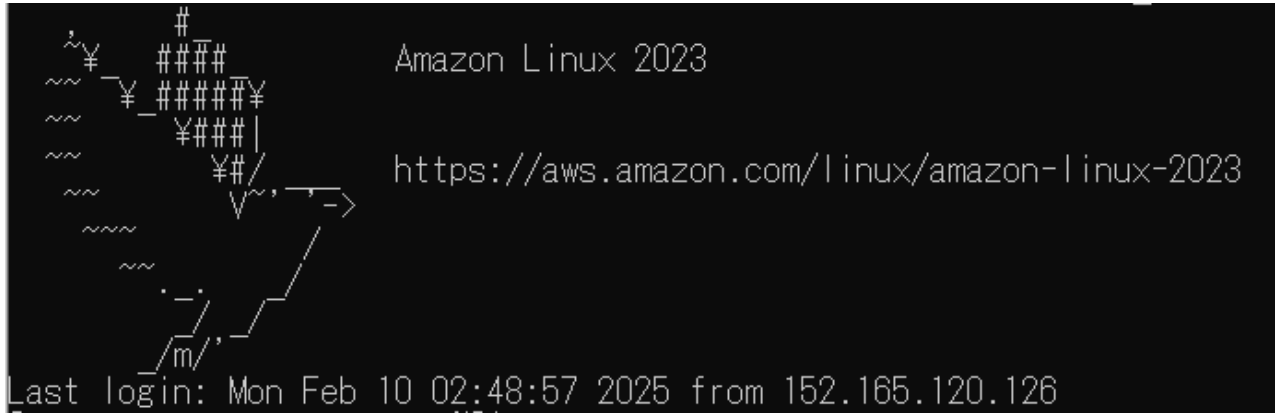
ParallelCluster実行用環境へログインする際は、構築したParallelCluster実行用インスタンスのパブリックIPアドレスへSSH接続を行う。

1. 下記のコマンドを実行してParallelCluster実行用環境にSSH接続する。

```
$ ssh -i “[SSH認証鍵名]” [ユーザー名]@[パブリックIPアドレス]
```

SSH接続設定	
ホスト名	ParallelCluster実行用インスタンスのパブリックIPアドレス
ユーザー名	ec2-user
パスワード	*** なし ***
SSH認証鍵名	例:aws-pcluster-ed25519-xxxxxxxx_tokyo.pem ※「キーペアの作成」で作成した秘密鍵

2. ターミナル上に下記のように表示されるとログイン成功である。



ParallelClusterのセットアップ

AWS ParallelClusterのセットアップの流れ

本項では、AWS ParallelCluster のセットアップ手順について解説する。AWS ParallelCluster のセットアップは、ParallelCluster 実行用インスタンス上で、以下の手順に従って実行する。以降のセクションで、それぞれの項目について具体的な手順を解説する。

1．AWS ParallelClusterのインストール
2．クラスタのセットアップ

ParallelClusterのセットアップ

1. AWS ParallelClusterのインストール(1/2)

1.1 pipのインストール

ParallelCluster実行用インスタンスにログインし下記コマンドを実行してpipをインストールする。

```
$ curl -O https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py
```

下記コマンドを実行し、バージョン情報が表示されるとインストール成功である。

```
$ pip --version
-----------------------------------------------------------
pip 24.3.1 from /home/ec2-user/apc-ve/lib/python3.9/site-packages/pip (python 3.9)
-----------------------------------------------------------
```

1.2 virtualenvのインストール

下記コマンドを実行し、Pythonの仮想環境構築ライブラリvirtualenvをインストールする。

```
$ python3 -m pip install --upgrade pip
$ python3 -m pip install --user --upgrade virtualenv
```

1.3 仮想環境の作成

下記コマンドを実行し、仮想環境を構築する。

```
$ python3 -m virtualenv ~/apc-ve
```

1.4 仮想環境をアクティブ化

下記コマンドを実行して仮想環境をアクティブ化する。

```
$ source ~/apc-ve/bin/activate
```

ParallelClusterのセットアップ

1. AWS ParallelClusterのインストール(2/2)

1.5 ParallelClusterをインストール

pipを使ってParallelClusterをインストールする。

```
(apc-ve) $ pip3 install aws-parallelcluster==3.10.1
```

下記コマンドを実行してバージョン情報が出力されるとインストール成功である。

```
(apc-ve) $ pcluster version
-------------------------------------------------------
{
"version": "3.10.1"
}
-------------------------------------------------------
```

1.6 Node.jsのインストール

下記のコマンドを実行してNode.jsをインストールする。

```
(apc-ve) $ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash
(apc-ve) $ chmod ug+x ~/.nvm/nvm.sh
(apc-ve) $ source ~/.nvm/nvm.sh
(apc-ve) $ nvm install --lts
```

下記コマンドを入力してバージョン情報が出力されるとインストール成功である。

```
(apc-ve) $ node --version
-------------------------------------------------------
{
v22.13.0
}
-------------------------------------------------------
```

ParallelClusterのセットアップ

2. クラスタのセットアップ(1/4)

2.1 AWS認証情報のセット

下記コマンドを入力して、認証情報を入力する。

```
$ aws configure
-----------------------------------------------------------
AWS Access Key ID [None]: {アクセスキーを入力}
AWS Secret Access Key [None]: {シークレットアクセスキーを入力}
Default region name [None]: ap-northeast-1
Default output format [None]: {入力せずEnterキーを押す}
-----------------------------------------------------------
```

※アクセスキー/シークレットアクセスキーは「アクセスキーの作成」で控えた文字列を入力

ParallelClusterのセットアップ

2. クラスタのセットアップ(2/4)

2.2 config作成コマンドの実行

下記コマンドを実行する。

```
(apc-ve)$ mkdir ~/pcluster-config
(apc-ve)$ pcluster configure --config ~/pcluster-config/cluster-config-01.yaml
```

2.3 パラメータ情報の入力

config作成コマンドの実行後、入力項目が表示されるのでパラメータ情報を入力していく。

（１）リージョンの選択:　「ap-northeast-1」を選択する。

```
AWS Region ID [ap-northeast-1]:ap-northeast-1
```

（２）キーペアの選択:「キーペアの作成」で作成したキーペア名を選択する。

```
EC2 Key Pair Name [aws-pcluster-ed25519-20250115_tokyo]:{ノードに利用するキーペアを選択}
```

（３）スケジューラの選択:Slurmを選択する。

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]: slurm
```

ParallelClusterのセットアップ

2. クラスタのセットアップ(3/4)

（４）OSの選択:rhel8を選択する。

```
Allowed values for Operating System:
 1. alinux2
 2. alinux2023
 3. ubuntu2004
 4. ubuntu2204
 5. rhel8
 6. rocky8
 7. rhel9
 8. rocky9 Operating System [alinux2]: rhel8
```

（５）Head Nodeのインスタンスタイプの選択:Graviton3/3Eベースのインスタンスを設定する。
例では「m7g.medium」を選択している。

```
Head node instance type [t2.micro]: m7g.medium  用途に合わせて設定する。
```

（６）Queue構成の選択:用途に合わせて設定する。（下記には入力コマンド例を記載する。）

```
Number of queues [1]: 1 ･･･queueの数を設定する。※Slurmでは計算ノードを「queue」という論理グループに分割できる。
Name of queue 1 [queue1]: queue1 ･･･queue名をqueue1とする。
[queue1]: queue1 Number of compute resources for queue1 [1]: 1  ･･･Queue1に割り当てる計算リソース数を設定する。
Compute instance type for compute resource 1 in queue1 [t2.micro]: m7g.2xlarge  ･･･計算ノードのインスタンスタイプを設定する。
Maximum instance count [10]: 10  ･･･最大利用できるnode数を設定する。例では最大10node使って計算可能。
```

Compute instance typeはGraviton3/3Eベースのインスタンスを選択する。
例では「m7g.2xlarge 」を選択している。

※インスタンスを選択する際は、目的に合わせて必要なスペックを見積り、下記URLから選択する。
インスタンスタイプに応じて利用料金が異なるため注意する必要がある。
https://aws.amazon.com/jp/ec2/instance-types/

ParallelClusterのセットアップ

2. クラスタのセットアップ(4/4)

（７）VPCの自動作成:yを選択する。

```
Automate VPC creation? (y/n) [n]: y
```

（８）AZの選択:「ap-northeast-1a」を選択する。

```
Allowed values for Availability Zone:
1. ap-northeast-1a
2. ap-northeast-1c
3. ap-northeast-1d Availability Zone [ap-northeast-1a]: ap-northeast-1a
```

（９）Head NodeとCompute Nodeの構成:1を選択する。

```
Allowed values for Network Configuration:
1. Head node in a public subnet and compute fleet in a private subnet
2. Head node and compute fleet in the same public subnet
Network Configuration [Head node in a public subnet and compute fleet in a private subnet]: Head node in a public
subnet and compute fleet in a private subnet
Network Configuration [Head node in a public subnet and compute fleet in a private subnet]: 1
```

入力完了後、ParallelCluster用のVPCやネットワーク設定が自動で作成される。作成は数分で完了し、その後はコマンド入力ができるようになる。

外部ストレージの構築

外部ストレージの構築について

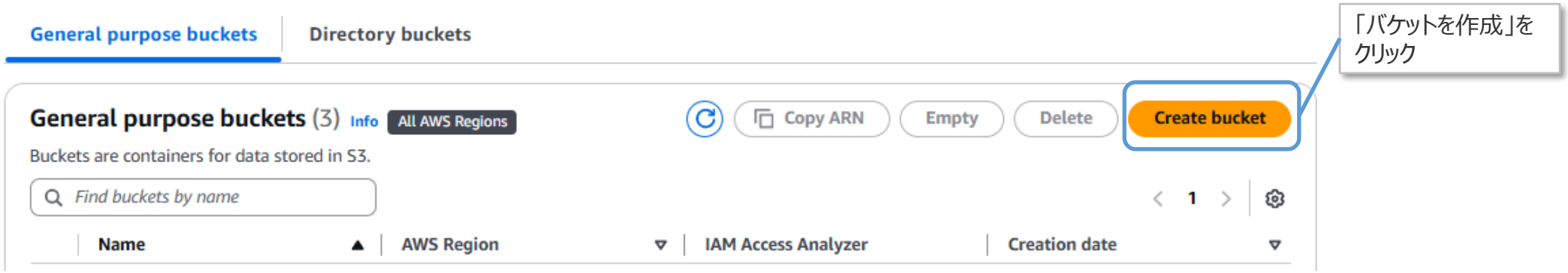
本マニュアルで構築するクラスタ環境では、計算結果などを長期保存するため、外部ストレージとしてS3 バケットを使用する。本項では、S3 バケットの作成手順について解説する。

1. S3バケットの作成(1/4)

1.1 S3バケット一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/s3/buckets?region=ap-northeast-1&bucketType=general

1.2 「バケットを作成」をクリックする。



外部ストレージの構築

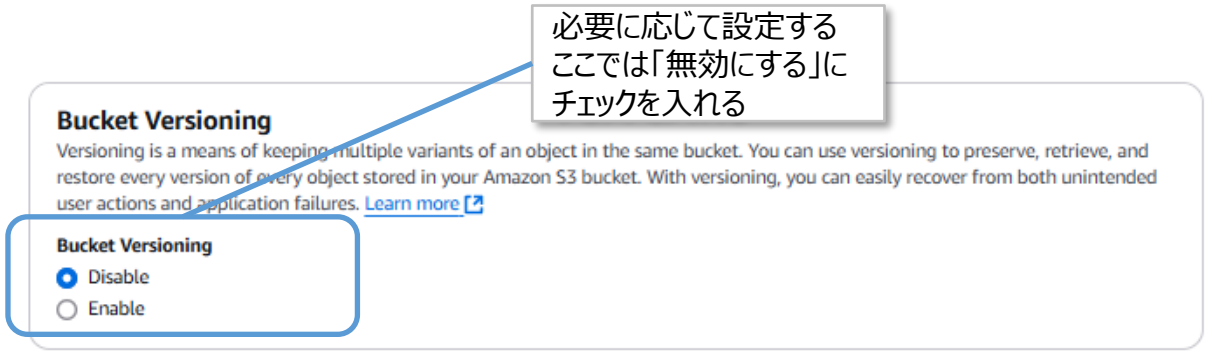
1. S3バケットの作成(2/4)

1.3 下記のようにパラメータを設定する。

外部ストレージの構築

1. S3バケットの作成(3/4)

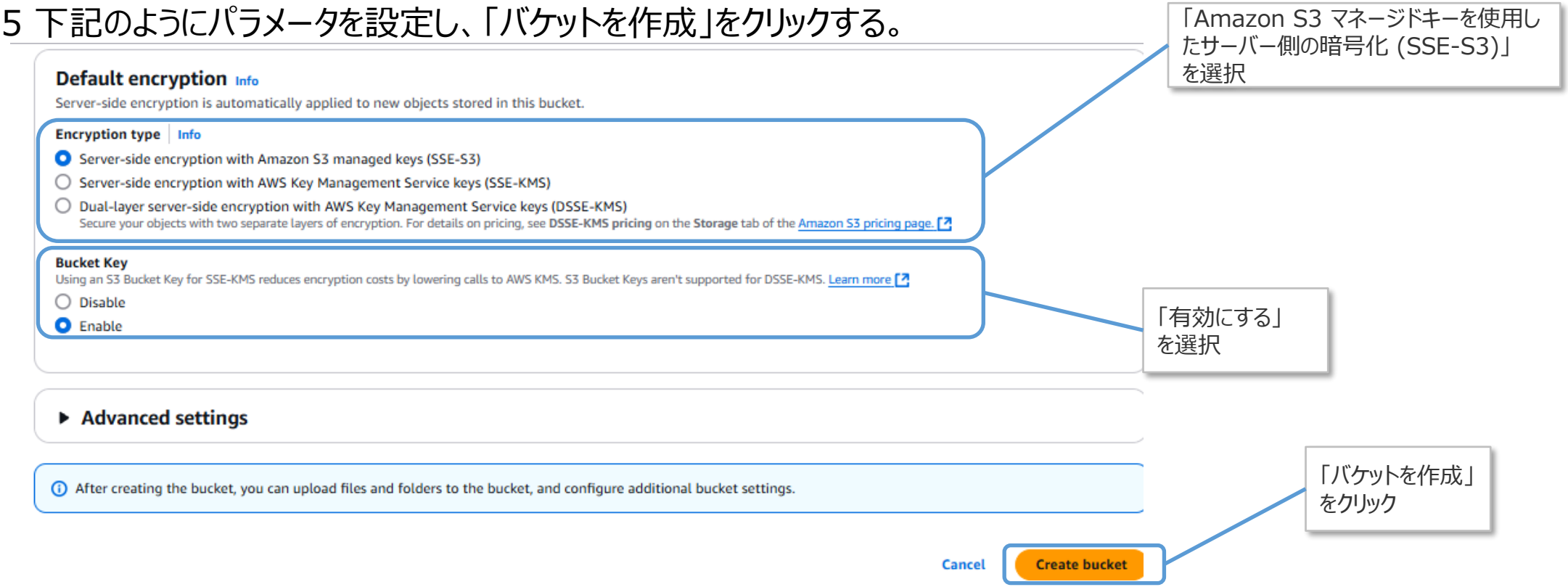
1.4 下記のようにパラメータを設定する。



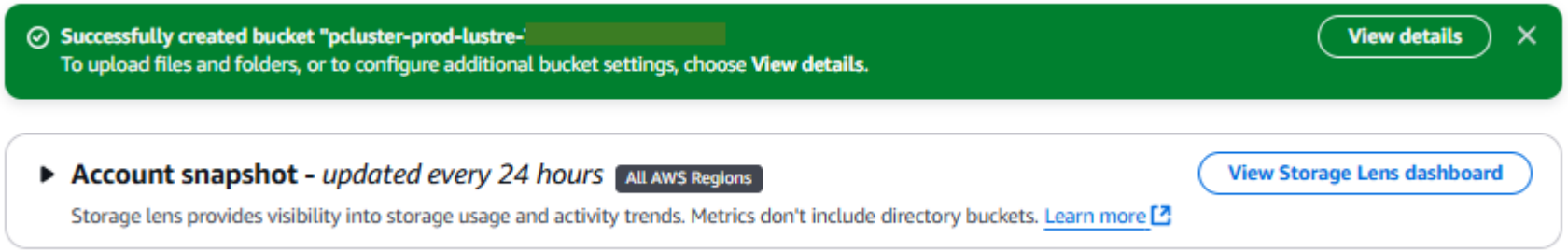
外部ストレージの構築

1. S3バケットの作成(4/4)

1.5 下記のようにパラメータを設定し、「バケットを作成」をクリックする。



1.6 下記の画面が表示されるとS3バケットの作成が完了し、外部ストレージの構築が完了する。



クラスタ構築

クラスタ構築の流れ

本項では、クラスタの作成手順について解説する。AWS ParallelCluster のセットアップと外部ストレージの構築が完了すると、クラスタを構築できるようになる。クラスタは、ParallelCluster 実行用インスタンス上で、以下の手順に従って構築する。次ページ以降では、クラスタ構築に必要な各項目の具体的な手順を解説する。

1．cluster-config-01.yamlの設定
2．クラスタの作成
3．Head Nodeへのログイン

クラスタ構築

1. cluster-config-01.yamlの設定

1.1 pcluster-configディレクトリ中に生成されたcluster-config-01.yamlに下記の赤字で示すストレージ設定を追記する。
※ 生成されたcluster-config-01.yamlには「ParallelClusterのセットアップ」にて指定した設定情報が記載される。

```
Region: ap-northeast-1
Image:
  Os: rhel8
HeadNode:
  InstanceType: m7g.medium
  Networking:
    SubnetId: subnet-xxxxxxxxxxx
  LocalStorage:
    RootVolume:
      Size: 64
      Encrypted: true
      VolumeType: gp2
      DeleteOnTermination: true
  Ssh:
    KeyName: aws-pcluster-ed25519-yyyymmdd_tokyo
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: queue1
    ComputeResources:
    - Name: m7g2xlarge
      Instances:
      - InstanceType: m7g.2xlarge
      MinCount: 0
      MaxCount: 10
    Networking:
      SubnetIds:
      - subnet-xxxxxxxxxxxxx
```

```
SharedStorage:
  - MountDir: /lustre01
    Name: lustre01
    StorageType: FsxLustre
    FsxLustreSettings:
      ImportedFileChunkSize: 1024
      DeploymentType: PERSISTENT_1
      PerUnitStorageThroughput: 100
      StorageCapacity: 1200
      AutoImportPolicy: NEW_CHANGED_DELETED
      ExportPath: s3://pcluster-prod-lustre- {Account ID}
      ImportPath: s3://pcluster-prod-lustre- {Account ID}
```

任意のストレージサイズを設定する。

Head Nodeのストレージ設定を追加する。
サイズは「バーチャル富岳」コンテナをダウンロードするため、ストレージサイズ64GB以上に設定する。

共有ストレージの設定を追加する。

作成したS3バケット名を記載する
例「pcluster-prod-lustre- {Account ID} 」と入力

共有ストレージの設定を本ファイル上で設定することで、自動的に共有ストレージが構築される。

クラスタ構築

2. クラスタの作成

2.1 クラスタ作成コマンドの実行

下記のコマンドを入力するとクラスタが作成される。作成完了までの時間は、FSx for Lustreなどが含まれているため20分程度かかる。

```
(apc-ve)$ pcluster create-cluster --cluster-name cluster01 --cluster-configuration ~/pcluster-config/cluster-config-01.yaml
```

2.2 クラスタのステータス確認

下記のコマンドを入力するとクラスタの作成状況が表示される。「cloudFormationStackStatus」が「CREATE_IN_PROGRESS」の場合はまだ作成進行中であり、「CREATE_COMPLETE」となっていればクラスタの作成は完了している。

```
(apc-ve)$ pcluster describe-cluster --cluster-name cluster01 |grep "cloudFormationStackStatus"
```

クラスタ構築

3. Head Nodeへのログイン(1/2)

Head Nodeへログインする際は、ParallelCluster実行用インスタンスにログインしてログイン操作を実施する。

3.1 仮想環境をアクティブ化

ParallelCluster実行用インスタンスへログインし、下記コマンドを実行する。

```
$ source ~/apc-ve/bin/activate
```

3.2 秘密鍵の保存

下記コマンドを実行し、キーペア秘密鍵の文字列を張り付けてキーペアの秘密鍵ファイルを作成し、
ParallelCluster実行用インスタンス内に保存する。

```
(apc-ve)$ vi ~/.ssh/aws-pcluster-ed25519-yyyymmdd_tokyo.pem
***** キーペアの秘密鍵の文字列を貼り付ける *****
```

クラスタ構築

3. Head Nodeへのログイン(2/2)

3.3 秘密鍵のパーミッションを変更

下記のコマンドを入力して秘密鍵ファイルのパーミッション設定を変更する。

```
(apc-ve)$ chmod 600 ~/.ssh/aws-pcluster-ed25519-20250115_tokyo.pem
```

3.4 Head NodeへSSH接続

下記コマンドを実行してHead Nodeへ接続する。

```
(apc-ve)$ pcluster ssh --cluster-name cluster01 -i ~/.ssh/aws-pcluster-ed25519-20250115_tokyo.pem
```

画面に下記のメッセージが表示されるとHead Nodeに接続に成功し、クラスタの構築が完了する。

```
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Tue Feb 18 04:56:09 2025 from xx.xx.xxx.xx
```

テストジョブの実行

クラスタ上でのジョブの投入

スーパーコンピュータを含むクラスタ環境では、ジョブ単位でプログラムを実行する仕組みが一般的である。本マニュアルで扱うクラスタでは、ジョブ管理システムSlurmを使用してジョブを投入し、実行する。Slurmを用いたジョブ投入では、以下の sbatch コマンドを使用する。

```
$ sbatch [ファイル名]
```

次ページ以降に簡単なテストジョブの実行手順について解説する。

テストジョブの実行

1. テストジョブ実行手順(1/2)

1.1 テスト用ジョブスクリプト作成

Head Node上で「vim sleep.sh」と入力して、ジョブスクリプトsleep.shを作成し、スクリプト内に下記内容を記載する。

```
$ vim sleep.sh
================================
#!/bin/bash
echo "sleeping..."
sleep 60
echo "done!"
================================
```

1.2 ジョブの実行

下記のsbatchコマンドを入力してジョブを投入する。しばらくするとジョブが計算ノード上で実行される。

※計算ノードはジョブが実行されると自動作成され、ジョブが完了すると自動的に削除される。

```
$ sbatch sleep.sh
-------------------------------------------------------
Submitted batch job 1
-------------------------------------------------------
```

テストジョブの実行

1. テストジョブ実行手順(2/2)

1.3ジョブの状況確認

ジョブの状況を確認したい場合はsqueueコマンドを実行することで確認できる。

```
$ squeue
-------------------------------------------------------------
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON)
1 queue1 sleep.sh ec2-user CF 0:08 1 queue1-dy-m7gmedium-12
-------------------------------------------------------------
```

※ジョブの状態コードの見方は下記URL参照方

https://slurm.schedmd.com/squeue.html#lbAG

1.4 計算終了

計算終了後slurm-xx.out(xxには番号が表示される。)というファイルが出力される。ファイル内に下記の内容が出力されていると計算完了である。

```
$ cat slurm-1.out
-------------------------------------------------------------
sleeping...
done!
-------------------------------------------------------------
```

共有ストレージへのデータ転送

共有ストレージについて

本項では、共有ストレージへのデータ転送方法について解説する。Head Node の特定のディレクトリには、Lustre ファイルシステム（FSx for Lustre）がマウントされており、共有ストレージとして利用される。Lustre ファイルシステム上のデータは、Head Node および計算ノードから保存・削除できる。

共有ストレージ (FSx for Lustre)
/lustre01

1.共有ストレージ(Lustreファイルシステム)へのデータ転送

- Head Nodeにログインし、下記のコマンドを実行するとHead Node内のデータをLustreファイルシステム内にコピーできる。
  （下記コマンドは、test.txtというファイルをLustreファイルシステムへコピーしている。）

```
$ cp test.txt /lustre01/
```

- Head Nodeにログインし、下記のコマンドを実行するとLustreファイルシステム内のデータをHead Node内にコピーできる。　（下記コマンドは、test.txtというファイルをHead Nodeへコピーしている。）

```
$ cp /lustre01/test.txt .
```

- 下記コマンドを実行することでLustreファイルシステム内のデータを確認できる。

```
$ ls -l /lustre01/
-----------------------------------------------------------
-rw-rw-r-- 1 ec2-user ec2-user 5 Jan 17 23:23 test.txt
-----------------------------------------------------------
```

外部ストレージへのデータエクスポート

外部ストレージへのデータ転送の流れ

本項では、外部ストレージへのデータ転送手順について解説する。クラスタ環境を削除すると、共有ストレージのデータも消去されるため、外部ストレージへの保管が必要となる。次ページ以降では、S3 バケットを使用した外部ストレージへのデータ転送方法について解説する。

1．共有ストレージから外部ストレージへのエクスポート
2．S3バケット内のファイル確認とダウンロード

外部ストレージへのデータエクスポート

1. 共有ストレージから外部ストレージへのエクスポート

下記コマンドを実行することで、共有ストレージからS3バケットにデータをエクスポートできる。

- 下記コマンドを実行することでLustreファイルシステム内のデータをS3バケットへコピーすることができる。

```
$ nohup find /lustre01/ -type f -print0 | xargs -0 -n 1 sudo lfs hsm_archive
```

- 下記コマンドを実行することでLustreファイルシステム内の特定のファイルをS3バケットへコピーすることができる。
（下記コマンドはtest.txtをS3バケットにコピー）

```
$ sudo lfs hsm_archive /lustre01/test.txt
```

外部ストレージへのデータエクスポート

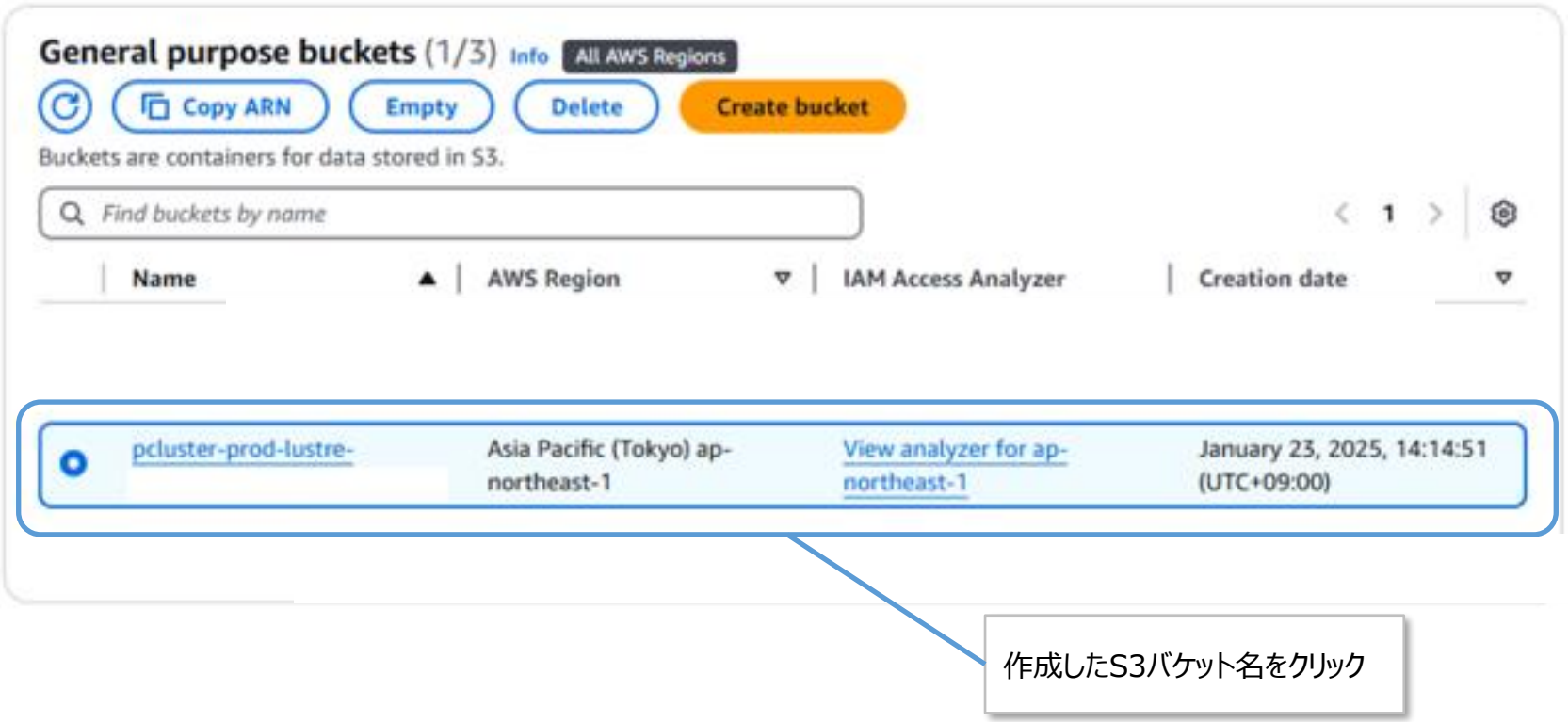
2. S3バケット内のファイル確認とダウンロード(1/3)

下記手順でS3バケット内のファイルを確認する。

2.1 S3バケット一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/s3/buckets?region=ap-northeast-1&bucketType=general

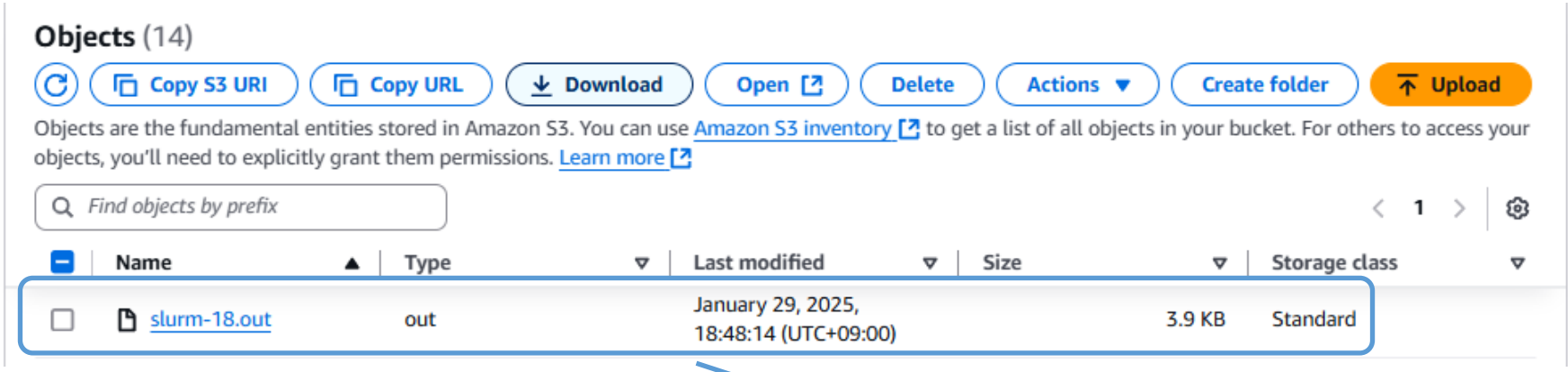
2.2 外部ストレージの構築で作成したS3バケット名をクリックする。



外部ストレージへのデータエクスポート

2. S3バケット内のファイル確認とダウンロード(2/3)

2.3 S3バケット内にLustreファイルシステムのデータが保存されていることを確認する。

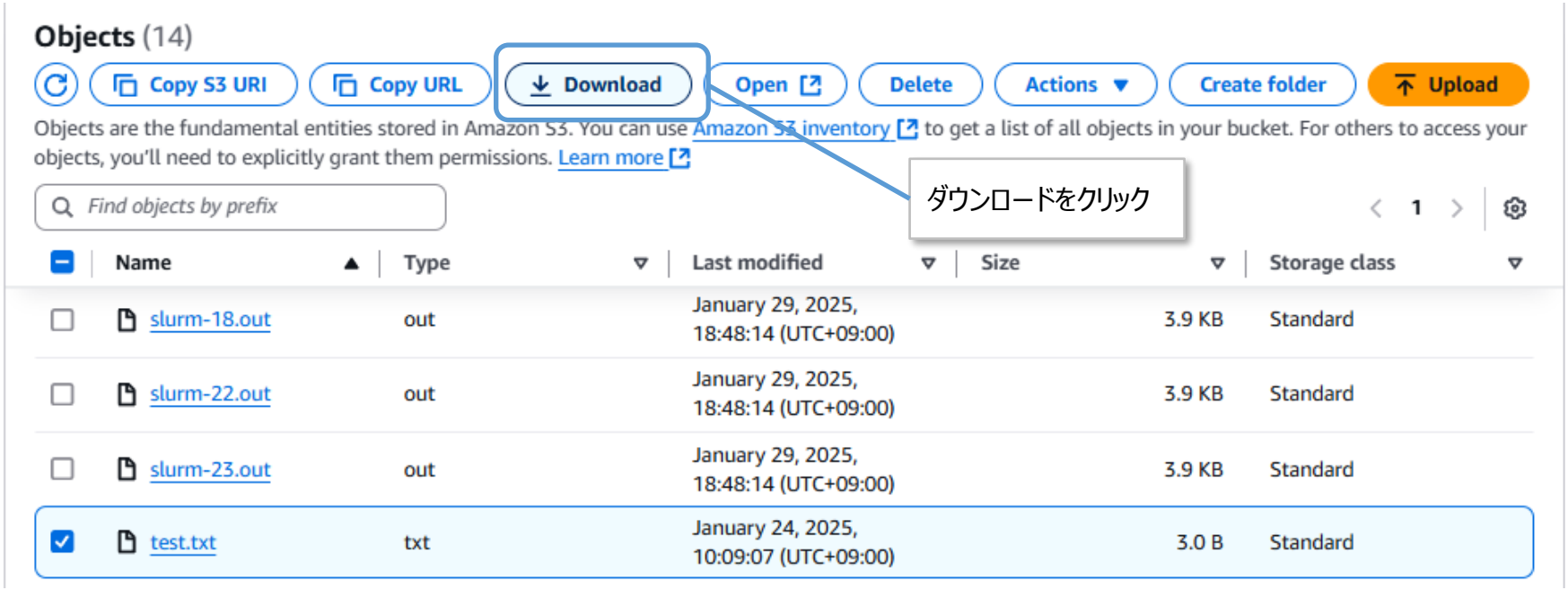


Lustreファイルシステムから転送されたデータが表示される。

外部ストレージへのデータエクスポート

2. S3バケット内のファイル確認とダウンロード(3/3)

2.4 ローカル端末に保存する必要のあるデータがあれば対象のオブジェクトを選択し、「ダウンロード」をクリックして
ローカル端末に保存する。（図ではtest.txtを選択）



ローカル環境からクラスタへのデータの転送

ローカル環境からクラスタへのデータの転送の流れ

本項では、ローカル環境からクラスタへのデータ転送手順について解説する。クラスタ上で計算を実行する際は、必要なデータを Head Node に転送する。Head Node へのデータ転送は、以下の手順で行う。次ページ以降では、それぞれの項目について具体的な操作手順を解説する。

1．ローカル環境から共有ストレージへファイルアップロード
2．共有ストレージからデータの取得

ローカル環境からクラスタへのデータの転送

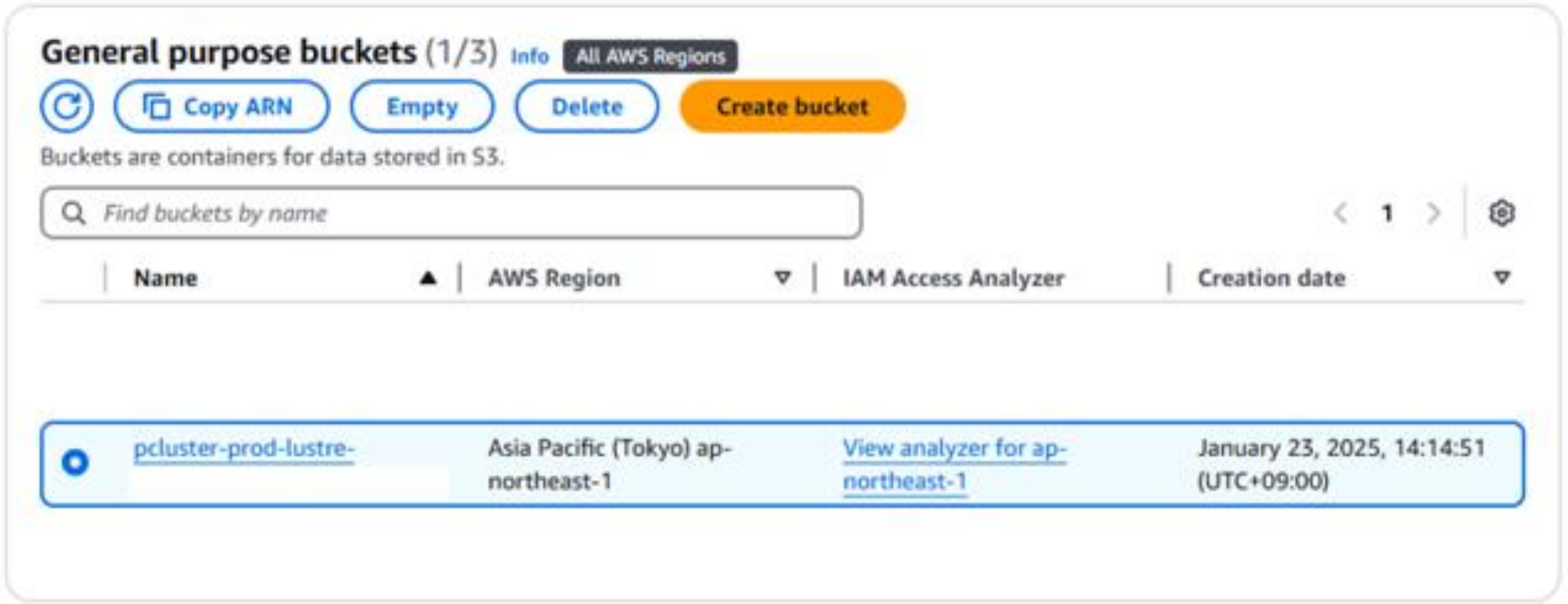
1. ローカル環境から共有ストレージへファイルアップロード(1/3)

下記手順でS3バケット内のファイルを確認する。

1.1 S3バケット一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/s3/buckets?region=ap-northeast-1&bucketType=general

1.2 外部ストレージの構築で作成したS3バケット名をクリックする。

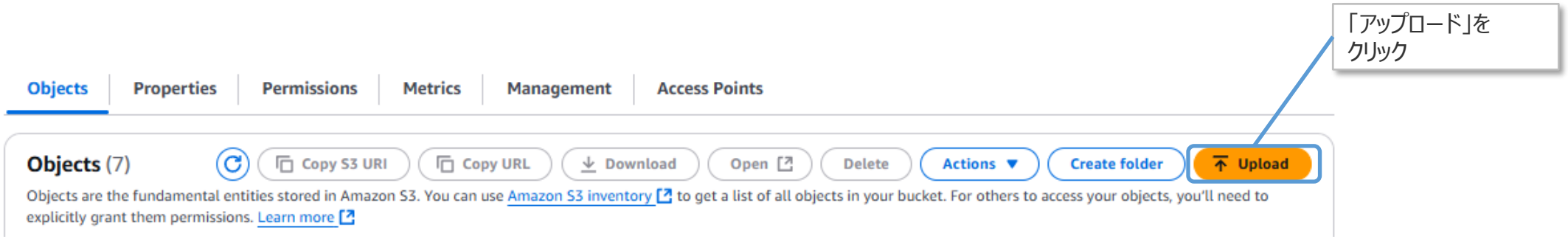


ローカル環境からクラスタへのデータの転送

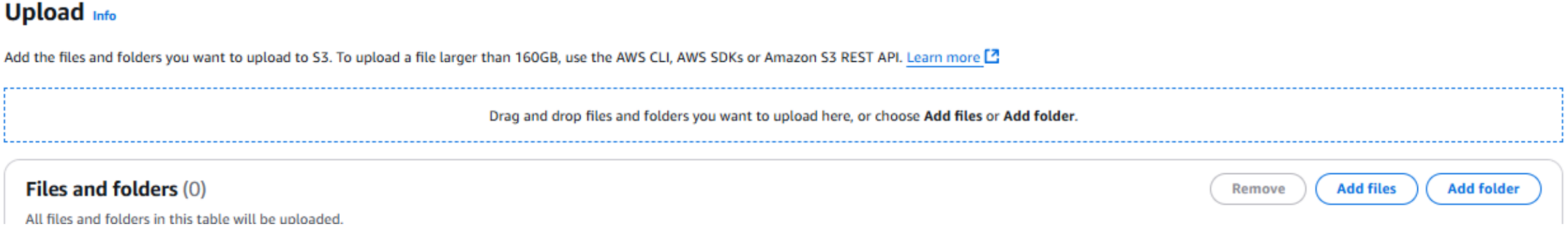
1. ローカル環境から共有ストレージへファイルアップロード(2/3)

下記手順でS3バケット内のファイルを確認する。

1.3 アップロードをクリックする。



1.4 アップロードしたいファイルをドラッグアンドドロップする。

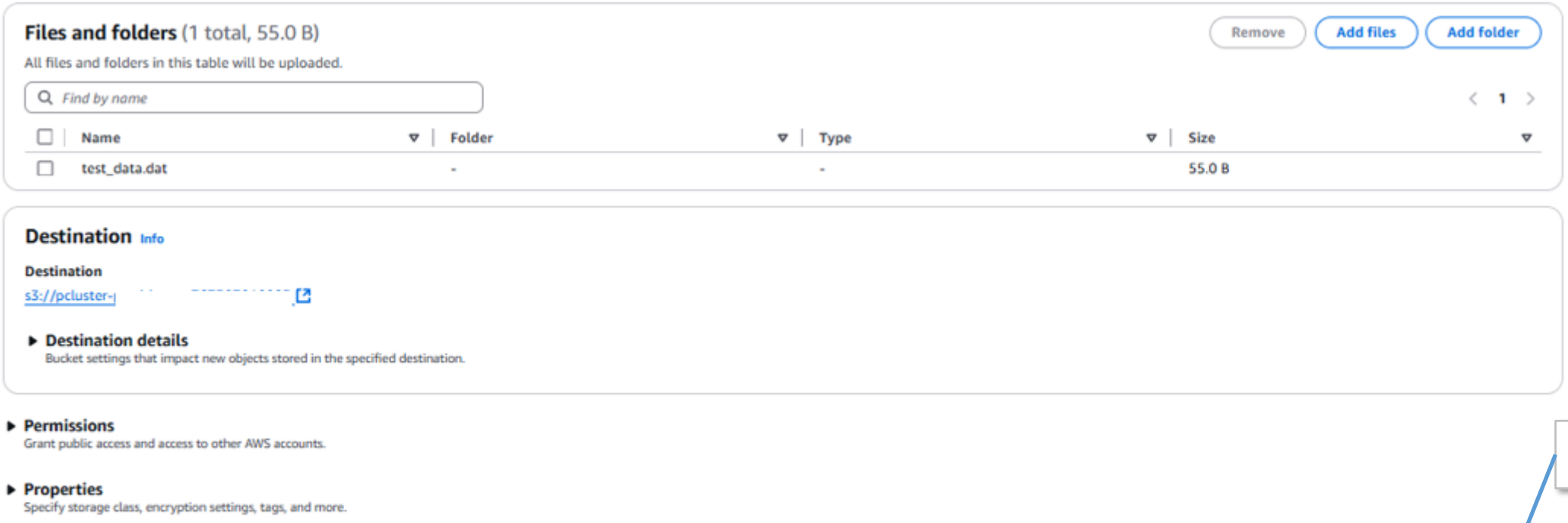


ローカル環境からクラスタへのデータの転送

1. ローカル環境から共有ストレージへファイルアップロード(3/3)

1.5 アップロードをクリック

ここでは、test_data.datというファイルをアップロードする。



「アップロード」をクリック

下記画面が表示されるとS3バケットへのアップロードが完了する。

ローカル環境からクラスタへのデータの転送

2. 共有ストレージからデータの取得

外部ストレージと共有ストレージはリンクしており、外部ストレージにアップロードされたデータは、共有ストレージから参照できる。

2.1 Head　Nodeにログインして下記コマンドを実行してLustreファイルシステム内のデータを確認する。

```
$ ls /lustre01/
------------------------------------------------------------
slurm-1.out  test_data.dat
------------------------------------------------------------
```

2.2 下記コマンドを実行してLustreファイルシステム内からHead　Nodeにデータをコピーする。
ここではtest_data.datをコピーする。

```
$ cp /lustre01/test_data.dat .
```

2.「プライベート富岳」の構築

（３）「バーチャル富岳」のインストール

Singularityとは

Singularity概要

Singularityとは主にHPCの分野で使用されている、コンテナプラットフォームである。Singularity を使うと、アプリケーションの実行に必要なシステム要件のうち、カーネル以外の部分を独立に構成・保持でき、アプリケーションと一体とできるため、同一アーキテクチャの環境であれば、実行環境をまるごと持ち運ぶことが可能になる。

また、アプリケーションに必要なランタイムライブラリの更新頻度が高い場合や、多彩なアプリケーションを駆使する必要がある場合に、それぞれの要件を個別に整えたり切り替えたりすることを考えなくてよくなる。さらに古い実行環境を保存しておくことも容易なため、データの再検証などもやりやすくなるメリットがある。

他の実装との違い

他の実装と比べた Singularity の際立った特徴は以下の4点である。

- イメージが単一ファイルで作成されるため、コンテナイメージの管理が直感的で容易である。
- イメージを展開せず、単一ファイルのまま利用できるため、アプリケーション稼働時の共有ファイルシステムへの I/O を抑制できる。
- 起動したユーザーのID・権限がコンテナ内部で継承され、同一ユーザーとしてファイル・ハードウエア・プロセスにアクセス可能である。
- イメージ作成やコンテナ実行に際して、root 権限やデーモンを一切介在させないため、セキュアな運用が可能である。

Singularityのインストール

Singularityのインストールの流れ

本項では、Singularityのインストール手順について解説する。以下の手順に従ってインストールを進める。次ページ以降で、それぞれの項目について詳しく説明する。

1．依存関係のインストール
2．Goのインストール
3．Singularityのコンパイル
4．SingularityコマンドのPath設定

Singularityのインストール

1. 依存関係のインストール

Singularityのインストール作業はHead Node上でおこなうため、Head Nodeにログインして作業を行う。

1.1 下記コマンドを実行して、依存関係をインストールする。

```
# Install basic tools for compiling
sudo dnf groupinstall -y 'Development Tools'
# Install RPM packages for dependencies
sudo dnf install -y ¥
   autoconf ¥
   automake ¥
   crun ¥
   cryptsetup ¥
   fuse ¥
   fuse3 ¥
   fuse3-devel ¥
   git ¥
   glib2-devel ¥
   libseccomp-devel ¥
   libtool ¥
   squashfs-tools ¥
   wget ¥
   zlib-devel
```

Singularityのインストール

2. Goのインストール

Singularityのインストールには Go言語が必要であるため、Go言語をインストールする。

2.1 Go言語を下記コマンドでインストールする。

```
$ sudo yum install -y golang
```

2.2 Go言語を下記コマンドを実行しバージョン情報が出力されるとインストール成功である。

```
$go version
================================================================================
go version go1.22.9 (Red Hat 1.22.9-1.module+el8.10.0+22500+aee717ef) linux/arm64
================================================================================
```

Singularityのインストール

3. Singularityのコンパイル

3.1 下記コマンドを入力し、Singularityソースコードを取得する。バージョンはv4.1.0を使用する。

```
$ export VERSION=4.1.0 && # adjust this as necessary ¥
    wget https://github.com/sylabs/singularity/releases/download/v${VERSION}/singularity-ce-${VERSION}.tar.gz && ¥
    tar -xzf singularity-ce-${VERSION}.tar.gz && ¥
    cd singularity-ce-${VERSION}
```

3.2 下記のコマンドを実行してSingularityをコンパイルする。

Singularityは計算ノードからも実行できるようにする必要があり、計算ノードと共有される領域にインストールする必要がある。そのため、--prefixで共有領域である「/opt/parallelcluster/shared」を指定してコマンドを実行する。

```
$ ./mconfig --prefix=/opt/parallelcluster/shared && ¥
    make -C ./builddir && ¥
    sudo make -C ./builddir install
```

Singularityのインストール

4. SingularityコマンドのPath設定

4.1 下記のコマンドを入力して.bashrcを開く。

```
$ vim ~/.bashrc
```

4.2 .bashrcファイル内に下記を入力し保存する。

```
export PATH=/opt/parallelcluster/shared/bin:$PATH
```

4.3 下記コマンドを実行して追加したPATH設定を反映させる。

```
$ source ~/.bashrc
```

4.4 下記コマンドを実行して4.1.0というバージョン情報が表示されるとSingularityのインストールは成功である。

```
$ singularity version
==========================
4.1.0
==========================
```

「バーチャル富岳」環境の取得

「バーチャル富岳」環境の取得方法

「富岳」と同等の環境をクラスタ上に展開するために、「富岳」にインストールされたソフトウェアスタックを含む Singularity コンテナイメージをダウンロードする。本項では、「バーチャル富岳」コンテナイメージのダウンロード方法を解説する。

1. 「バーチャル富岳」コンテナイメージファイルのダウンロード

1.1 Head Nodeにログインし、homeディレクトリ上で下記のコマンドを実行し、Sylabs Cloud LibraryからSingularity コンテナイメージをダウンロードする。

※ 以下は、バージョン1.1 のコンテナイメージの例である。実際には最新バージョンを指定してダウンロードして下さい。
最新バージョンの版数は、「バーチャル富岳」公式サイト（https://www.r-ccs.riken.jp/fugaku/virtual-fugaku/）で確認できます。

```
$ singularity pull library://riken-rccs/virtual-fugaku/vf-ver1.1
```

ダウンロードが完了すると「バーチャル富岳」のインストールが完了する。

3. アプリケーションの実行

（１）「バーチャル富岳」にインストールされたアプリケーションの実行

実行するアプリケーション

本マニュアルでは、「バーチャル富岳」にインストールされている下記の3アプリケーションの実行例について解説する。

- GENESIS

  超並列分子動力学計算ソフトウェアである。独自の計算アルゴリズムを用いて、並列計算を高効率化し、細胞環境を想定した1億個の原子で構成される系に対して、高速なシミュレーションが可能である。

  URL:https://www.r-ccs.riken.jp/software_center/jp/software/genesis/overview/

- GROMACS

  オープンソースの古典分子動力学アプリケーションソフトウェアである。主に生体分子系の計算事例が多く、高速な並列計算が可能なことが特徴である。

  URL:https://www.gromacs.org/

- SCALE

  スーパーコンピュータから汎用計算機に至るまで広く用いられることを念頭において開発されている次世代気象科学における基盤ライブラリーである。

  URL:https://scale.riken.jp/ja/

GENESISの実行

GENESIS実行の流れ

本項では、GENESISの実行手順として、必要ファイルのダウンロード、ジョブスクリプトの作成、実行、および結果の確認までの流れを解説する。

1. 実行手順(1/3)

以降の操作はHead Node上で実施する。

1.1 必要ファイルのダウンロード

任意のディレクトリ上で下記コマンドを実行する。

```
$ wget https://www.r-ccs.riken.jp/labs/cbrt/wp-content/uploads/2020/12/benchmark_mkl_ver4_nocrowding.tar.gz
$ tar -xzvf benchmark_mkl_ver4_nocrowding.tar.gz
```

1.2 ディレクトリbenchmark_mkl_ver4_nocrowdingに移動する。

```
$ cd benchmark_mkl_ver4_nocrowding
```

GENESISの実行

1. 実行手順(2/3)

1.3 ジョブスクリプトの作成

ディレクトリbenchmark_mkl_ver4_nocrowdingに下記のジョブスクリプトgenesis.shを作成する。

■ genesis.sh

```
#!/bin/bash
#SBATCH -p queue1
#SBATCH --ntasks=8
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH -J test_genesis

set -ex

SIFFILE=~/vf-ver1.1_latest.sif

export SINGULARITYENV_LD_LIBRARY_PATH=/usr/lib64:/opt/amazon/efa/lib64
export SINGULARITY_BIND=${PWD},/opt/amazon,/usr/lib64/libefa.so.1,/usr/lib64/libibverbs.so.1
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

cd npt/genesis1.6_2.5fs/jac_amber
mpiexec --use-hwthread-cpus -n ${SLURM_NTASKS} singularity -v run ${SIFFILE} spdyn p${SLURM_NTASKS}.inp
```

1.4 ジョブの実行

下記のコマンドを実行してジョブを投入する。

```
$ sbatch genesis.sh
```

GENESISの実行

1. 実行手順(3/3)

1.5 結果の確認

ジョブが完了するとジョブを投入したディレクトリにslurm-xx.outというファイルが出力される。(xxはジョブIDが表示される。)ファイルの中身を確認し、下記の情報が出力されているとアプリケーションが正常動作している。

```
Output_Time> Averaged timer profile (Min, Max)
  total time       =      56.798
   setup           =       1.057
   dynamics        =      55.741
     energy        =      42.362
     integrator    =       6.591
     pairlist      =       4.504 (       4.376,       4.620)
====================以下省略=========================================
```

GROMACSの実行

GROMACS実行の流れ

本項では、GROMACSの実行手順として、必要ファイルのダウンロード、ジョブスクリプトの作成、実行、および結果の確認までの流れを解説する。

1. 実行手順(1/3)

以降の操作はHead Node上で実施する。

1.1 必要ファイルのダウンロード

任意のディレクトリ上で下記コマンドを実行する。

```
$ wget https://ftp.gromacs.org/pub/benchmarks/ADH_bench_systems.tar.gz
$ tar -xzvf ADH_bench_systems.tar.gz
```

1.2 ディレクトリADH内のにadh_cubicディレクトリに移動する。

```
$ cd ADH/adh_cubic
```

GROMACSの実行

1. 実行手順(2/3)

1.3 ジョブスクリプトの作成

ディレクトリadh_cubic内に下記のジョブスクリプトgromacs.shを作成する。

■gromacs.sh

```
#!/bin/bash
#SBATCH -p queue1
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=2
#SBATCH --nodes=4#SBATCH --ntasks-per-node=2
#SBATCH -J test_gromacs
SIFFILE=~/vf-ver1.1_latest.sif
export SINGULARITYENV_LD_LIBRARY_PATH=/usr/lib64:/opt/amazon/efa/lib64
export SINGULARITY_BIND=/opt/amazon,/usr/lib64/libefa.so.1,/usr/lib64/libibverbs.so.1
mpiexec --use-hwthread-cpus -n 1 singularity run ${SIFFILE} gmx_mpi grompp -f pme_verlet.mdp -c conf.gro -p topol.top -o ions.tpr
mpiexec --use-hwthread-cpus -n ${SLURM_NTASKS} singularity run ${SIFFILE} gmx_mpi mdrun -ntomp ${SLURM_CPUS_PER_TASK} -s ions.tpr
```

1.4 ジョブの実行

下記のコマンドを実行してジョブを投入する。

```
$ sbatch gromacs.sh
```

GROMACSの実行

1. 実行手順(3/3)

1.5 結果の確認

ジョブが完了するとジョブを投入したディレクトリにslurm-xx.outというファイルが出力される。(xxはジョブIDが表示される。)ファイルの中身を確認し、下記の情報が出力されているとアプリケーションが正常動作している。

```
Dynamic load balancing report:
 DLB was turned on during the run due to measured imbalance.
 Average load imbalance: 1.0%.
 The balanceable part of the MD step is 68%, load imbalance is computed from this.
 Part of the total run time spent waiting due to load imbalance: 0.7%.
 Steps where the load balancing was limited by -rdd, -rcon and/or -dds: X 0 %
               Core t (s)   Wall t (s)        (%)
       Time:     2017.288      126.081     1600.0                 (ns/day)    (hour/ns)
Performance:       13.707        1.751
GROMACS reminds you: "Even if you are on the right track, you will get run over if you just sit there." (Will Rogers)
```

SCALEの実行

SCALE実行の流れ

本項では、SCALEの実行手順として、必要ファイルのダウンロード、ジョブスクリプトの作成、実行、および結果の確認までの流れを解説する。また、可視化ツールGrADSを用いた計算結果可視化方法について解説する。

1. 実行手順(1/4)

以降の操作はHead Node上で実施する。

1.1 必要ファイルのダウンロード

任意のディレクトリ上で下記コマンドを実行する

```
$ wget https://scale.riken.jp/archives/scale-5.4.5.tar.gz
$ tar -xzvf scale-5.4.5.tar.gz
```

1.2 ディレクトリscale-5.4.5に移動する。

```
$ cd scale-5.4.5
```

SCALEの実行

1. 実行手順(2/4)

1.3 ジョブスクリプトの作成

ディレクトリscale-5.4.5内に下記のジョブスクリプトscale.shを作成する。

■scale.sh

```
#!/bin/bash
#SBATCH -p queue1
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=8
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=1
#SBATCH -J test_scale
SIFFILE=~/vf-ver1.1_latest.sif
export SINGULARITYENV_LD_LIBRARY_PATH=/usr/lib64:/opt/amazon/efa/lib64
export SINGULARITY_BIND=${PWD},/opt/amazon,/usr/lib64/libefa.so.1,/usr/lib64/libibverbs.so.1
cp -pr ../scale-5.4.5/scale-rm/test/tutorial/ideal/* .
#初期値の作成
cp sample/init_R20kmDX500m.conf ./init_R20kmDX500m.conf
mpiexec --use-hwthread-cpus -n ${SLURM_NTASKS} singularity run ${SIFFILE} scale-rm_init init_R20kmDX500m.conf
#シミュレーションの実行
cp sample/run_R20kmDX500m.conf ./run_R20kmDX500m.conf
mpiexec --use-hwthread-cpus -n ${SLURM_NTASKS} singularity run ${SIFFILE} scale-rm run_R20kmDX500m.conf
#後処理
cp sample/net2g_R20kmDX500m.conf ./net2g_R20kmDX500m.conf
mpiexec --use-hwthread-cpus -n ${SLURM_NTASKS} singularity run ${SIFFILE} net2g net2g_R20kmDX500m.conf
```

SCALEの実行

1. 実行手順(3/4)

1.4 ジョブの実行

下記のコマンドを実行してジョブを投入する。

```
$ sbatch scale.sh
```

1.5 結果の出力

ジョブが完了するとジョブを投入したディレクトリにslurm-xx.outというファイルと下記の結果ファイルが出力される。
(xxはジョブIDが表示される)下記の結果ファイルをローカル環境にダウンロードする。

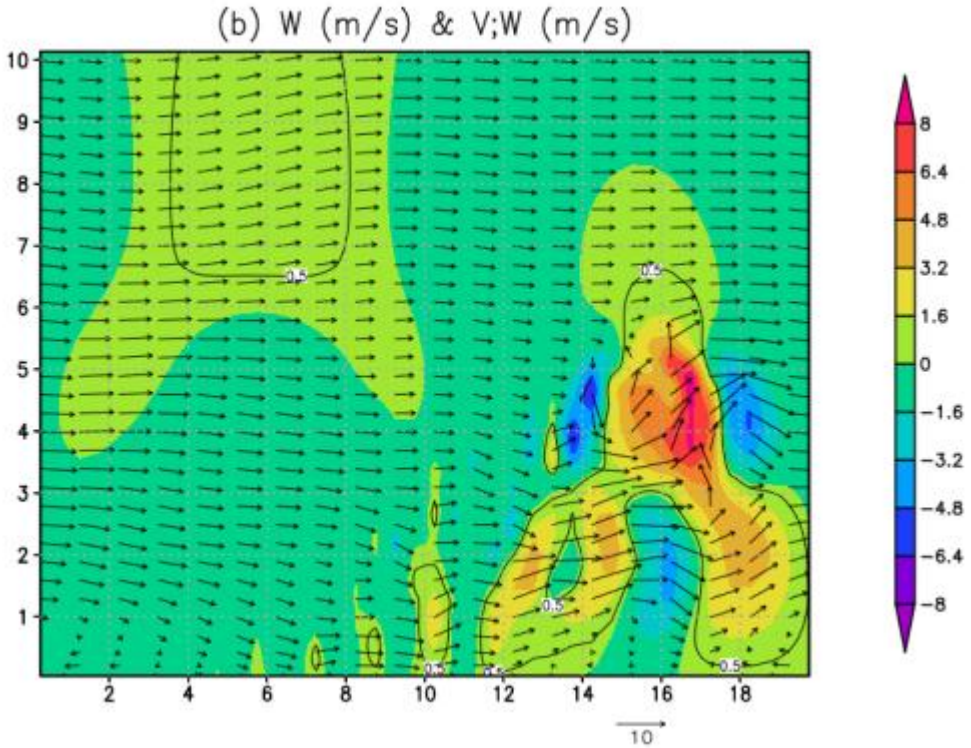
- QHYD_d01z-3d.ctl
- QHYD_d01z-3d.grd
- V_d01z-3d.ctl
- V_d01z-3d.grd
- W_d01z-3d.ctl
- W_d01z-3d.grd

SCALEの実行

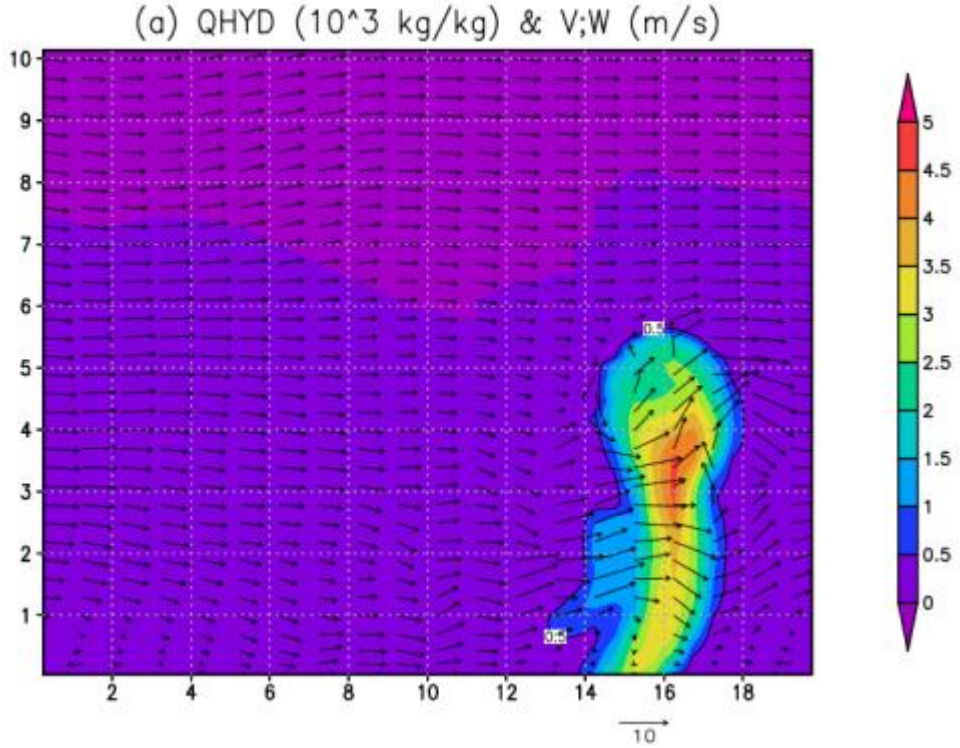
1. 実行手順(4/4)

1.6 結果の確認

ダウンロードしたファイルをローカル環境で GrADS を用いて可視化し、以下の2つの可視化結果ファイルが出力されていれば、アプリケーションは正常に動作している。（GrADSを用いた可視化方法は次ページに記載）



ideal_W.png



ideal_qhyd.png

SCALEの実行

2. GrADSによる可視化方法(1/2)

ローカル環境でのGrADSによるSCALEの結果ファイルの可視化方法を解説する。 可視化処理を行うローカル環境はUbuntu22.04.4環境を想定している。

2.1 GrADSのインストール

ローカル環境にて下記コマンドを実行してGrADSをインストールする。

```
$ sudo apt-get upgrade
$ sudo apt-install grads
```

2.2 可視化用GrADSスクリプトの取得

SCALEを実行する際にダウンロードしたディレクトリscale-5.4.5内の「scale-rm/test/tutorial/ideal」配下にcheckfig_ideal.gsという計算結果可視化用スクリプトファイルが配置されている。本ファイルを計算結果ファイルとともにローカル環境にコピーする。

SCALEの実行

2. GrADSによる可視化方法(2/2)

2.3 可視化の実行

計算結果ファイルとスクリプトcheckfig_ideal.gsが配置されたディレクトリに移動し、下記のコマンドを実行するとGrADSが起動し可視化処理が実行される。

```
$ grads -blc checkfig_ideal.gs
```

2.4 コマンド実行後、ディレクトリ内に下記の可視化結果ファイルが出力される。

- ideal_qhyd.png
- ideal_W.png

4.「プライベート富岳」の削除
（１）クラスタ環境の削除

クラスタ環境の削除

クラスタ環境削除の流れ

本項では、ParallelCluster で構築したクラスタ環境の削除手順について解説する。計算に使用したクラスタ環境が不要になった場合は、以下の手順で削除する。次ページ以降では、クラスタ削除に必要な手順を詳しく説明する。

1．クラスタの削除
2．クラスタ用VPCの削除
3．ParallelCluster実行用インスタンスの削除
4．固定IPアドレスの解放

クラスタ環境の削除

クラスタ環境の削除作業は、ParallelCluster実行用インスタンスにSSHログインして作業を進める。

1. クラスタの削除

1.1 下記コマンドでPython仮想環境をアクティブ化する。

```
$ source ~/apc-ve/bin/activate
```

1.2 下記コマンドを入力してクラスタを削除する。実行後削除完了まで10～20分ほど時間がかかる。

```
(apc-ve)$ pcluster delete-cluster --region ap-northeast-1 --cluster-name cluster01
```

1.3 クラスタの状況は下記コマンドを実行することで確認できる。「cloudFormationStackStatus」が「DELETE_IN_PROGRESS」の場合はクラスタ削除中である。クラスタ削除完了していた場合は何も出力されない。

```
(apc-ve)$ pcluster describe-cluster --cluster-name cluster01 |grep "cloudFormationStackStatus"
```

クラスタ環境の削除

2. クラスタ用VPCの削除(1/3)

2.1 下記コマンドを実行してスタックを確認する。

```
(apc-ve)$ aws --region ap-northeast-1 cloudformation list-stacks --stack-status-filter "CREATE_COMPLETE" --query
"StackSummaries[].StackName" |grep -e "parallelclusternetworking-"
-------------------------------------------------------
 "parallelclusternetworking-pubpriv-xxxxxxxxxxxxx",
-------------------------------------------------------
```

「parallelclusternetworking-pubpriv-xxxxxxxxxxxxx」はVPC作成に関するCloudFormationスタック名である。この文字列を控えておく。

2.2 スタックの削除

下記コマンドを実行してスタックを削除する。「parallelclusternetworking-pubpriv-xxxxxxxxxxxxx」の部分は先ほど控えたスタック名を入力する。

```
(apc-ve)$ aws --region ap-northeast-1 cloudformation delete-stack --stack-name parallelclusternetworking-pubpriv-
xxxxxxxxxxxxx
```

クラスタ環境の削除

2. クラスタ用VPCの削除(2/3)

2.3 VPC一覧ページへアクセス

https://ap-northeast-1.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-1#vpcs

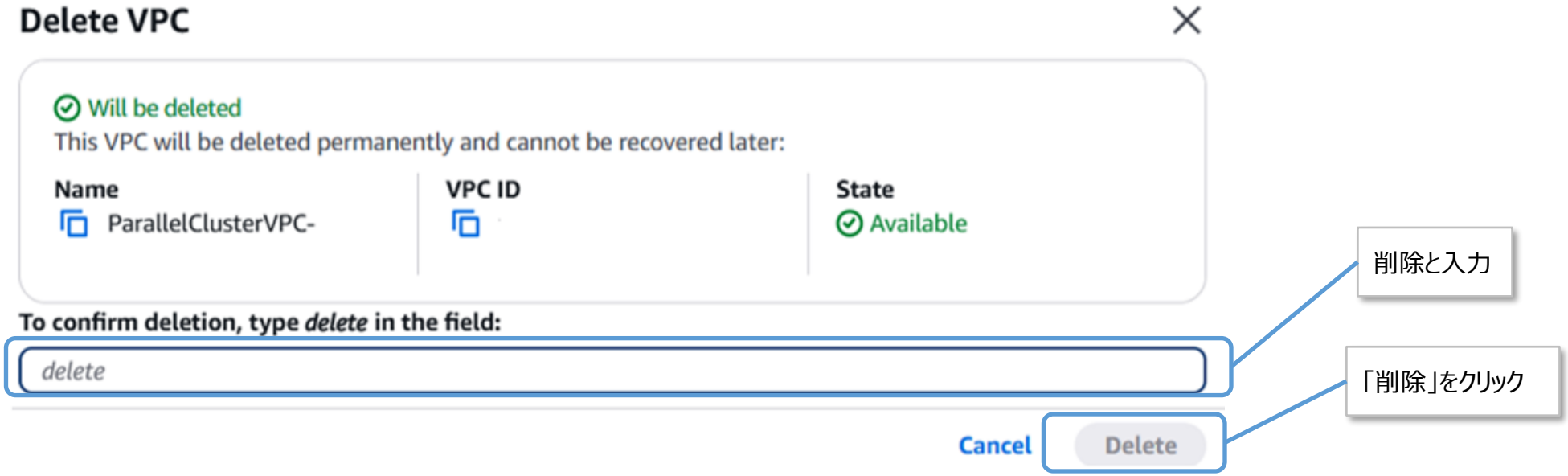
削除対象ののVPCを選択する。

クラスタ環境の削除

2. クラスタ用VPCの削除(3/3)

2.4 右上の「アクション」=>「VPCの削除」をクリックする。

2.5 削除と確認欄に入力し「削除」をクリックするとVPCが削除される。



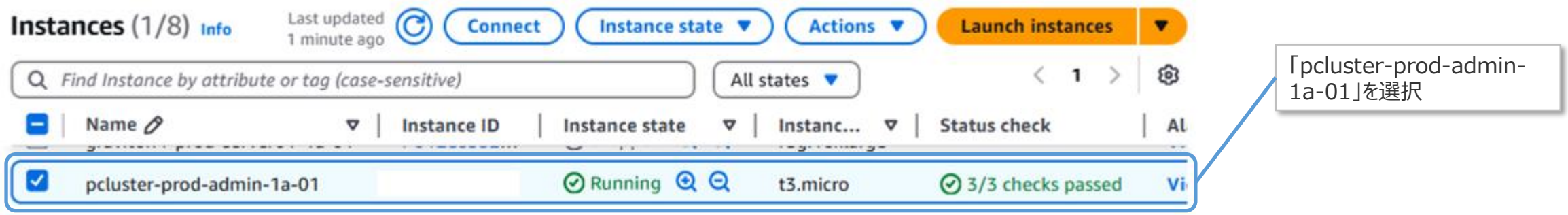
クラスタ環境の削除

3. ParallelCluster実行用インスタンスの削除(1/2)

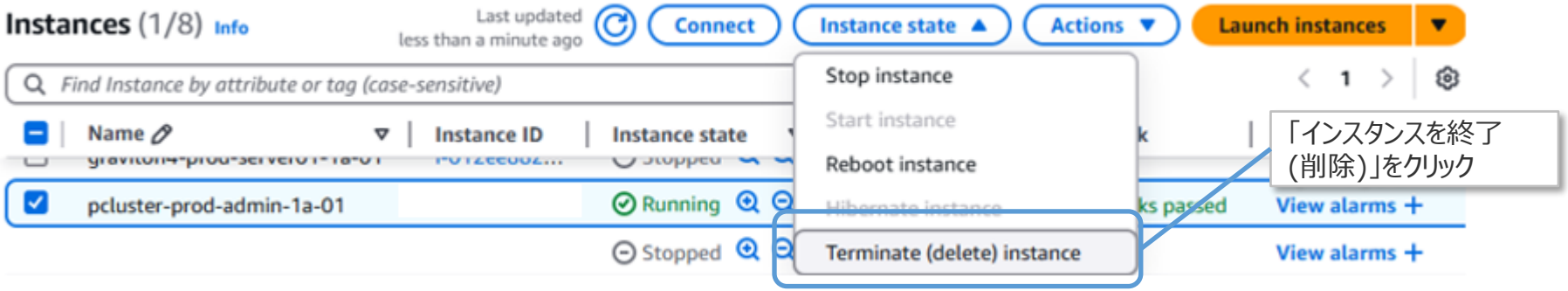
3.1 EC2インスタンス一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#Instances

3.2 「pcluster-prod-admin-1a-01」を選択する。



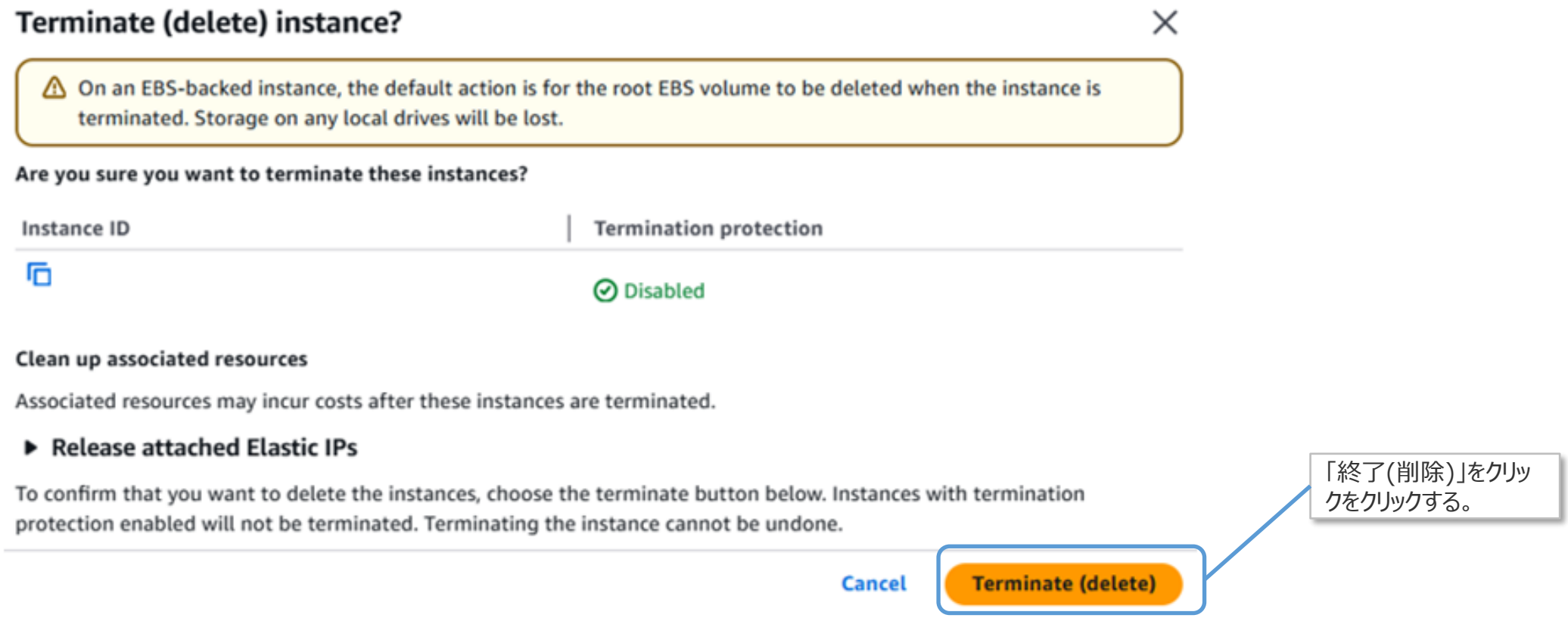
3.3 右上の「インスタンスの状態」=>「インスタンスを終了(削除)」をクリックする。



クラスタ環境の削除

3. ParallelCluster実行用インスタンスの削除(2/2)

3.4「終了(削除)」をクリックをクリックする。



クラスタ環境の削除

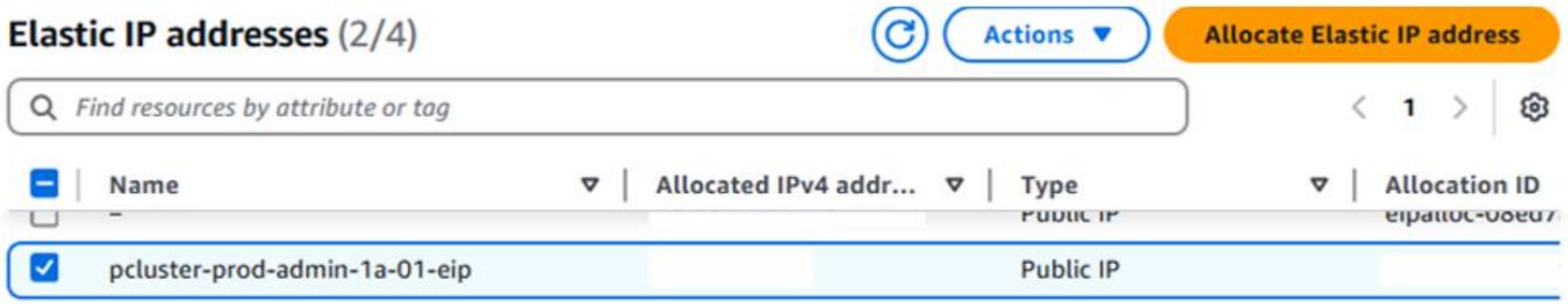
4. 固定IPアドレスの解放(1/3)

ParallelCluster実行用インスタンス削除後、 ParallelCluster実行用インスタンスに割り当てていた固定IPアドレスを開放する。

4.1 EIP一覧ページへアクセスする。

https://ap-northeast-1.console.aws.amazon.com/ec2/home?region=ap-northeast-1#Addresses：

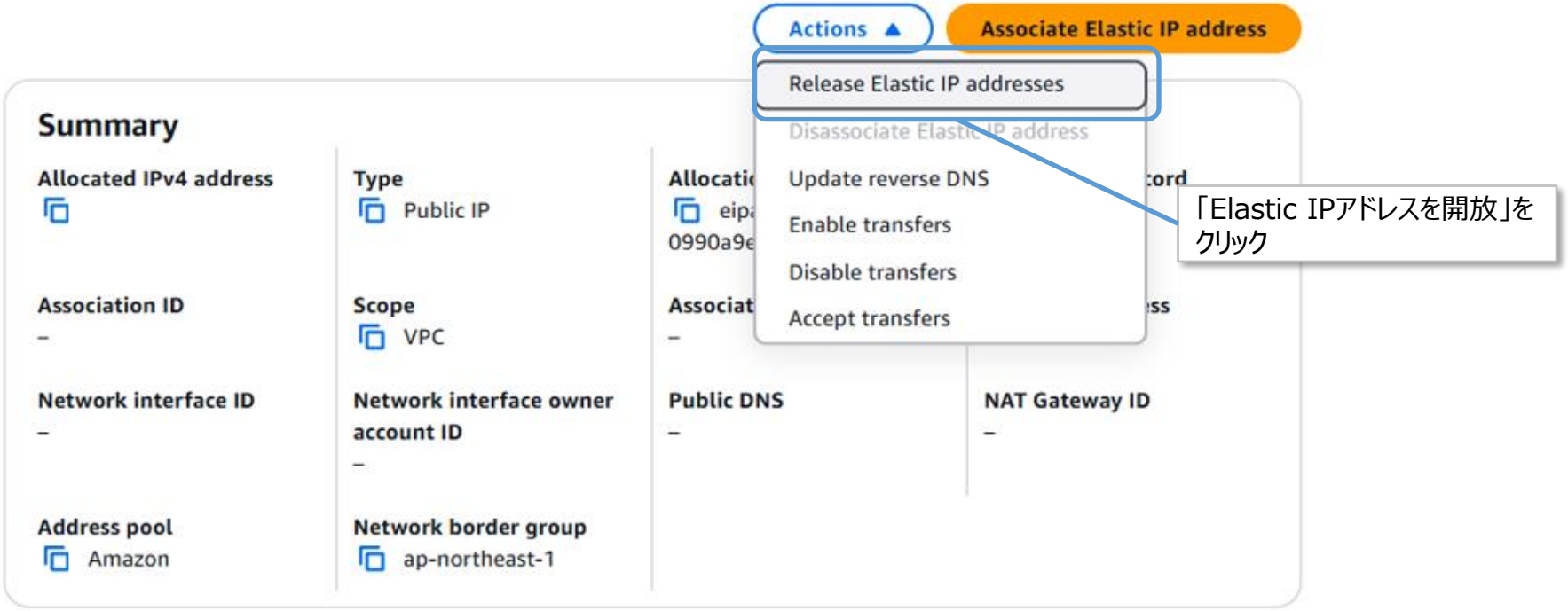
4.2 開放するElastic IPアドレスを選択する。



クラスタ環境の削除

4. 固定IPアドレスの解放(2/3)

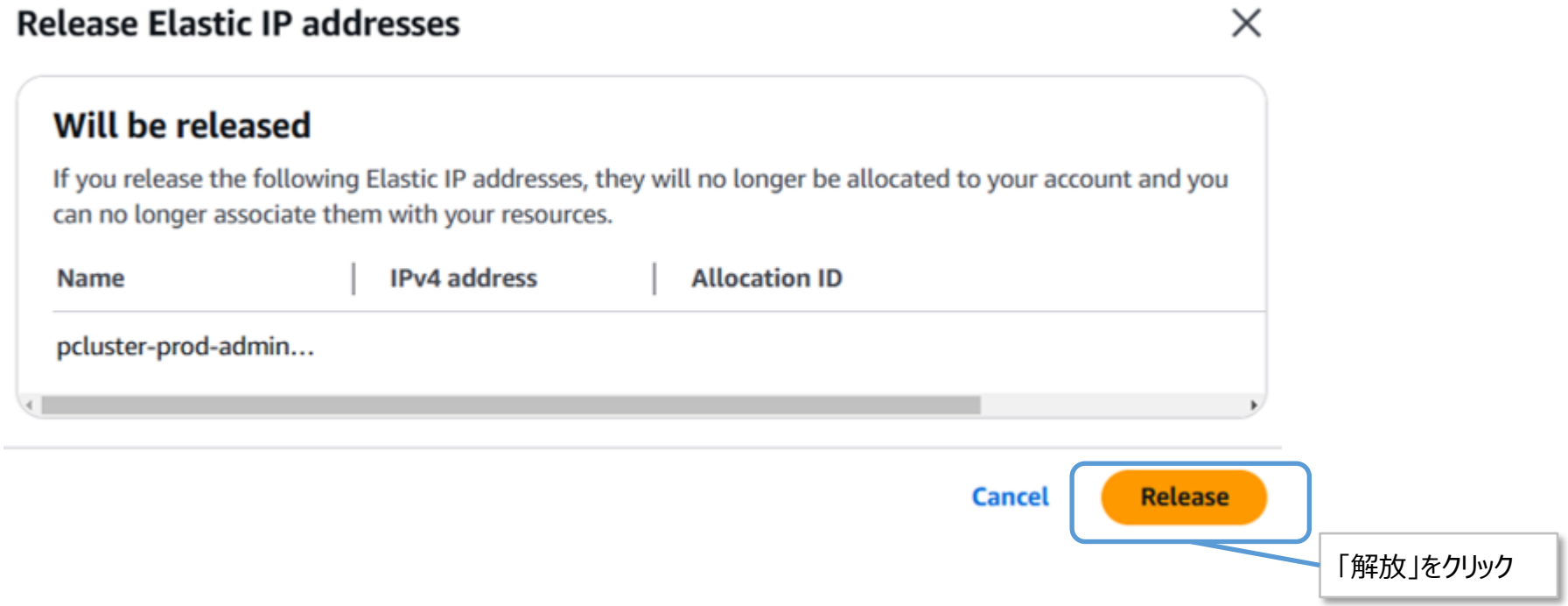
4.3 右上の「アクション」=>「Elastic IPアドレスを解放」をクリックする。



クラスタ環境の削除

4. 固定IPアドレスの解放(3/3)

4.4 「解放」をクリックする。



4.5 Elastic IPアドレスが解放されると下記のメッセージが表示される。

参考資料

「バーチャル富岳」にインストールされているパッケージ

「バーチャル富岳」にインストールされているパッケージ一覧

「バーチャル富岳」コンテナにはスーパーコンピュータ「富岳」での利用頻度が高い下記のアプリケーションがインストールされている。 (2025/2/19時点)

順次アップデートされるため、最新の状況は下記URLを参照方。

URL:https://www.r-ccs.riken.jp/fugaku/virtual-fugaku/

- GENESIS 2.1.3 (genesis)
- Gnuplot 6.0.0 (gnuplot)
- GROMACS 2024.2 (gromacs)
- GNU Scientific Library (GSL) 2.7.1 (gsl)
- Julia 1.10.2 (julia)
- LAMMPS 20230802.3 (lammps)
- Metis 5.1.0 (metis)
- Open Babel 3.1.1 (openbabel)
- OpenFoam 2312 (openfoam)
- Paraview 5.12.1 (paraview)
- Parmetis 4.0.3 (parmetis)
- PETSC 3.21.2 (petsc)
- Atomic Simulation Environment 3.21.1 (py-ase)
- Matplotlib (py-matplotlib)
- MPI for Python (py-mpi4py)
- NumPy (py-numpy)
- pandas (py-pandas)
- scikit-learn (py-scikit-learn)
- SciPy (py-scipy)
- TOML (py-toml)
- Quantume Espresso 7.3.1 (quantum-espresso)
- SCALE 5.4.4 (scale)
- tmux 3.4 (tmux)

AWS PallalelCluster　共有ストレージ一覧

ParallelCluster環境における共有ストレージ

下記表に示すHead Node内のディレクトリは計算ノードとNFSで共有されている。計算時に必要なファイルはHead Node上の「/home」や「/opt/parallelcluster/shared」に保存することで計算ノードと共有することが可能である。

共有ストレージ(EBS)
/home
/opt/parallelcluster/shared
/opt/slurm (書き込み権限なし)