AWS CI/CD for Amazon ECS ハンズオン

~ Cloud9, Docker, Code Services を用いた開発効率向上 ~

Amazon Web Services Japan K.K.

2020/04/28

本ハンズオンで学ぶこと

ハンズオン①

AWS 上での Docker 管理環境の構築方法

- Cloud9 を使ったクラウド開発環境
- ECS/ECR を使ったコンテナ環境／コンテナイメージ管理
- AWS Fargate を使ったクラスターレスコンテナ運用

ハンズオン②

AWS Code Servicesを利用したCI/CD環境構築方法

- CodeCommitを使ったソースコード管理
- CodeBuildを使ったDockerイメージの自動ビルド
- CodeDeployを使ったECSへのBlue/Green Deployment
- CodePipelineを使ったCI/CDパイプラインの構築

事前準備

サービス概要／用語の理解

- Amazon Elastic Container Service
  https://www.slideshare.net/AmazonWebServicesJapan/20190731-black-belt-online-seminar-amazon-ecs-deep-dive-162160987
- AWS CodeCommit/CodeBuild/CodePipeline
  https://www.slideshare.net/AmazonWebServicesJapan/aws-black-belt-online-seminar-2017-aws-code-services-codecommit-codebuild
- AWS Cloud9
  https://www.slideshare.net/AmazonWebServicesJapan/20180613-aws-black-belt-online-seminar-aws-cloud9

ハンズオンコンテンツ

サンプルソースは以下から取得してください。

- サンプルソース：https://pages.awscloud.com/rs/112-TZM-766/images/cicd_handson_sample.zip

zipファイルはダウンロード後に解凍してください。

以降のハンズオンはすべて
N.Virginiaリージョンで実施します

Amazon ECS

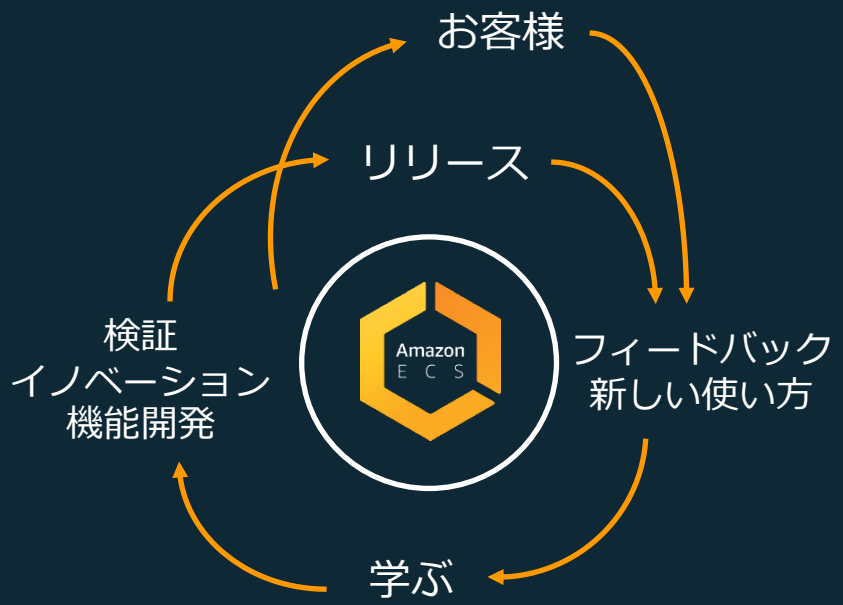
Amazon Elastic Container Service

450+%
年間アクティブユーザの成長
(2016年と比較)

数億コンテナ
が毎週起動

数百万もの
インスタンス上で

お客様がAmazon ECS成長の鍵



50+

2015年のGA以来
リリースした機能の数

AWS上の本番環境のコンテナを支える

- AWS VPCネットワークモード
- タスク配置
- 他のAWSサービスとの深い連携
- ECS CLI
- グローバル展開
- 強力なスケジューラ
- オートスケーリング
- CloudWatch メトリクス
- ロードバランサ

アプリケーションの開発に集中したい

計算リソースの使い方を根本的に変える

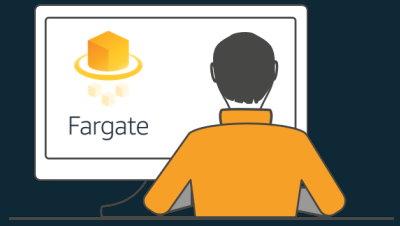
AWS Fargate

インスタンス
管理不要

タスク
ネイティブAPI

リソース
ベースの価格

=



簡素で、使いやすく、
強力な
新しいリソース消費モデル

タスクに割り当てるCPUとメモリの設定

柔軟な設定の選択肢

– **50** のCPU/メモリ設定から

CPU	Memory
256 (.25 vCPU)	512MB, 1GB, 2GB
512 (.5 vCPU)	1GB to 4GB (1GB 刻み)
1024 (1 vCPU)	2GB to 8GB (1GB 刻み)
2048 (2 vCPU)	4GB to 16GB (1GB 刻み)
4096 (4 vCPU)	8GB to 30GB (1GB 刻み)

ECSでFargateタスクに指定できる主なもの

タスクの機能として

- コンテナイメージ
- CPU，メモリ
- IAMロール
- VPCサブネット
- セキュリティグループ
- CloudWatch Logs

サービスの機能として

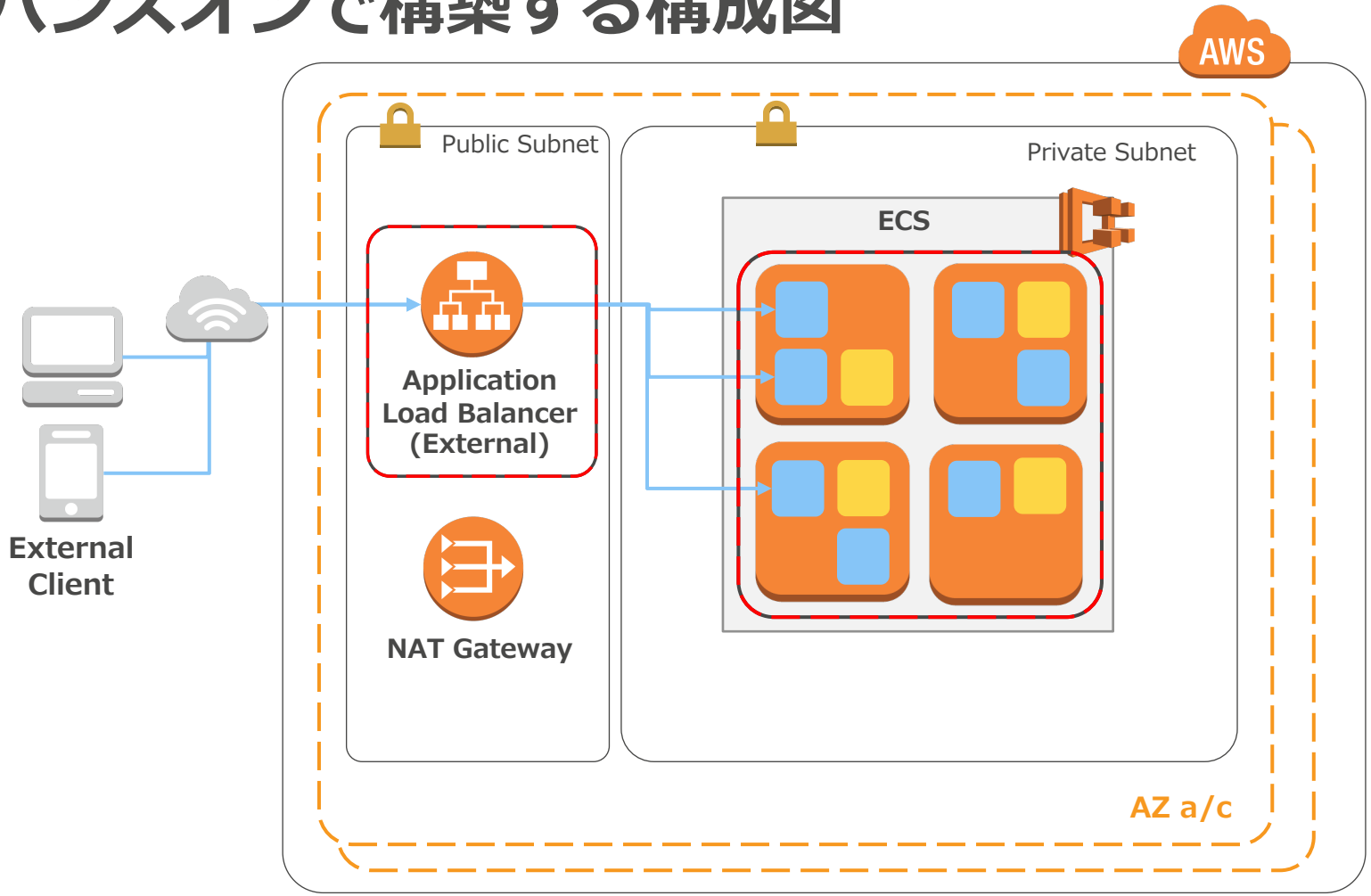
- ELB (ALB/NLB)
- オートスケーリング

aws

ハンズオン①

ECS環境の構築

本ハンズオンで構築する構成図



ハンズオン1の手順

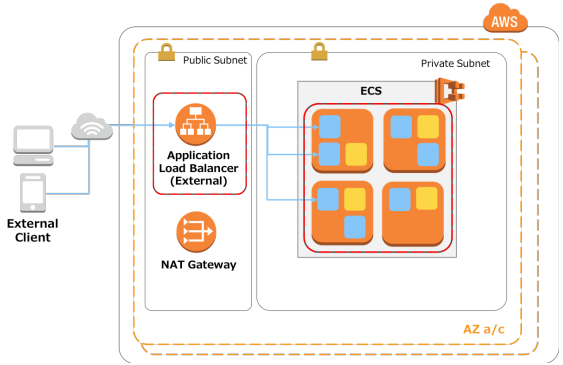
1. VPC作成
2. ALB作成
3. ECSクラスター作成
4. Dockerアプリ構築
    - ECRレジストリ作成
    - Cloud9環境の構築
5. ECSタスク・サービス作成
6. ECSサービスAuto Scaling設定

VPC作成

CloudFormationの実行手順の説明 (1)

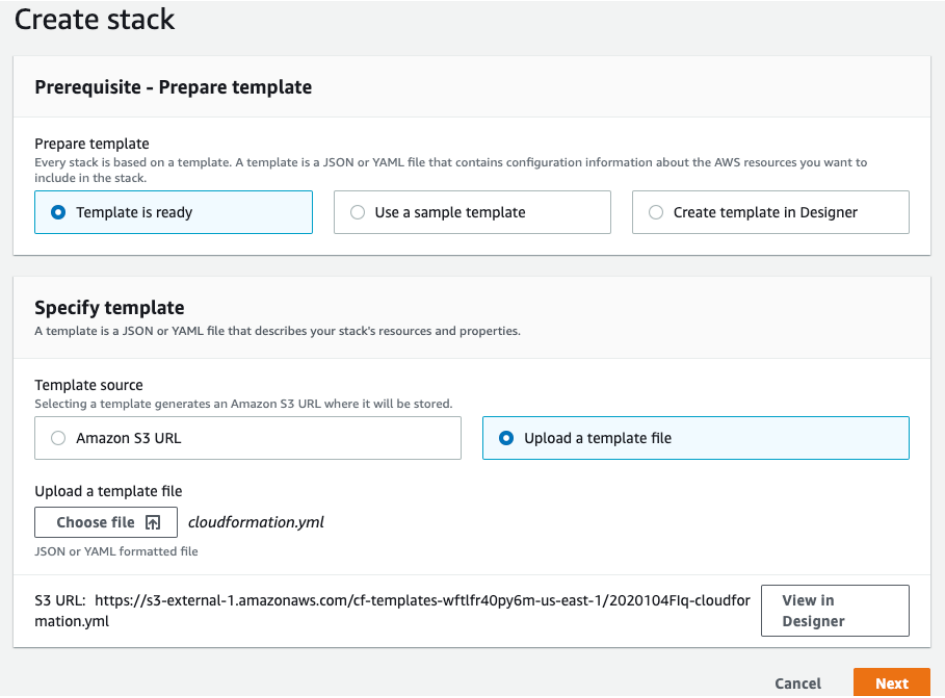
本日のハンズオンに必要となるネットワークを構築する。

- 主題の ECS/Cloud9/CICD に集中するため、今回は CloudFormation で以下を自動構築する。
  - VPC [10.1.0.0/16]
  - Internet Gateway & EIP
  - NAT Gateway & EIP * 2
  - Subnet
    - Public Subnet [10.1.0.0/24, 10.1.2.0/24]
    - Private Subnet [10.1.1.0/24, 10.1.3.0/24]
  - Route Table



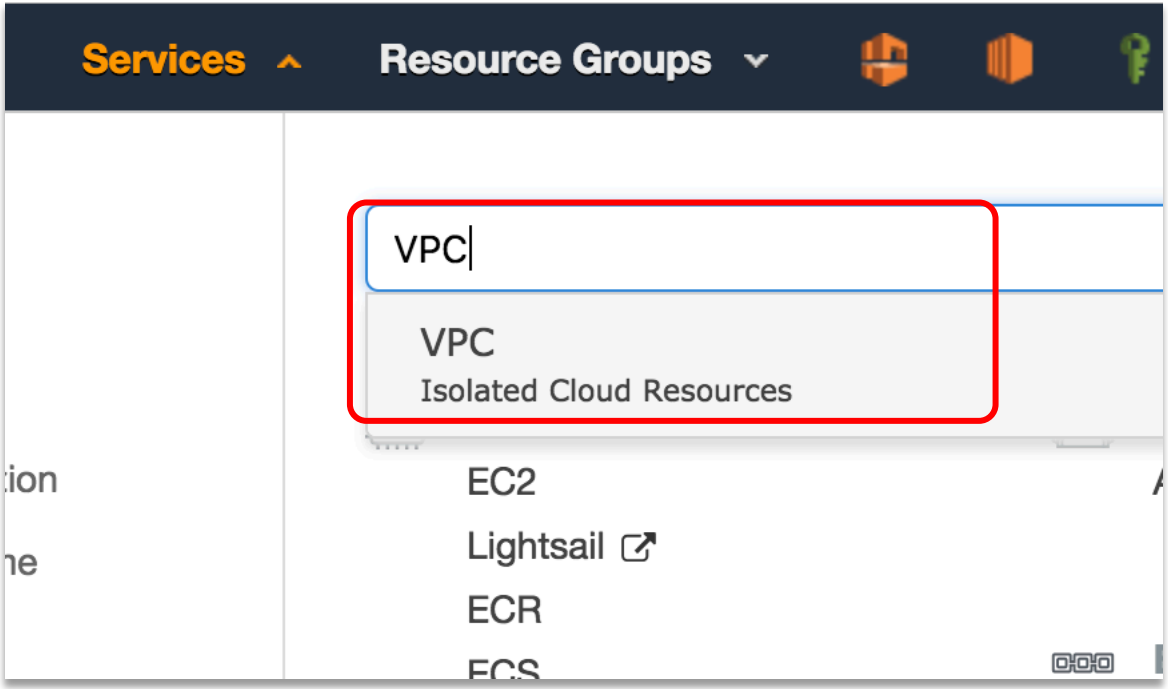
CloudFormationの実行手順の説明 (2)

1. https://console.aws.amazon.com/console を開く
2. N.Virginia Region を選択する
3. Services -> CloudFormation を開く
4. Create Stack を押す
5. Specify templateのUpload template file を選択
6. Choose fileをクリックし、解凍したcloudformation.ymlを選択し、Nextをクリック
7. Stack name を "vpc-for-handson-cicd" と入力
8. 他は特に入力せず Next → Next -> Create Stack



ALB作成

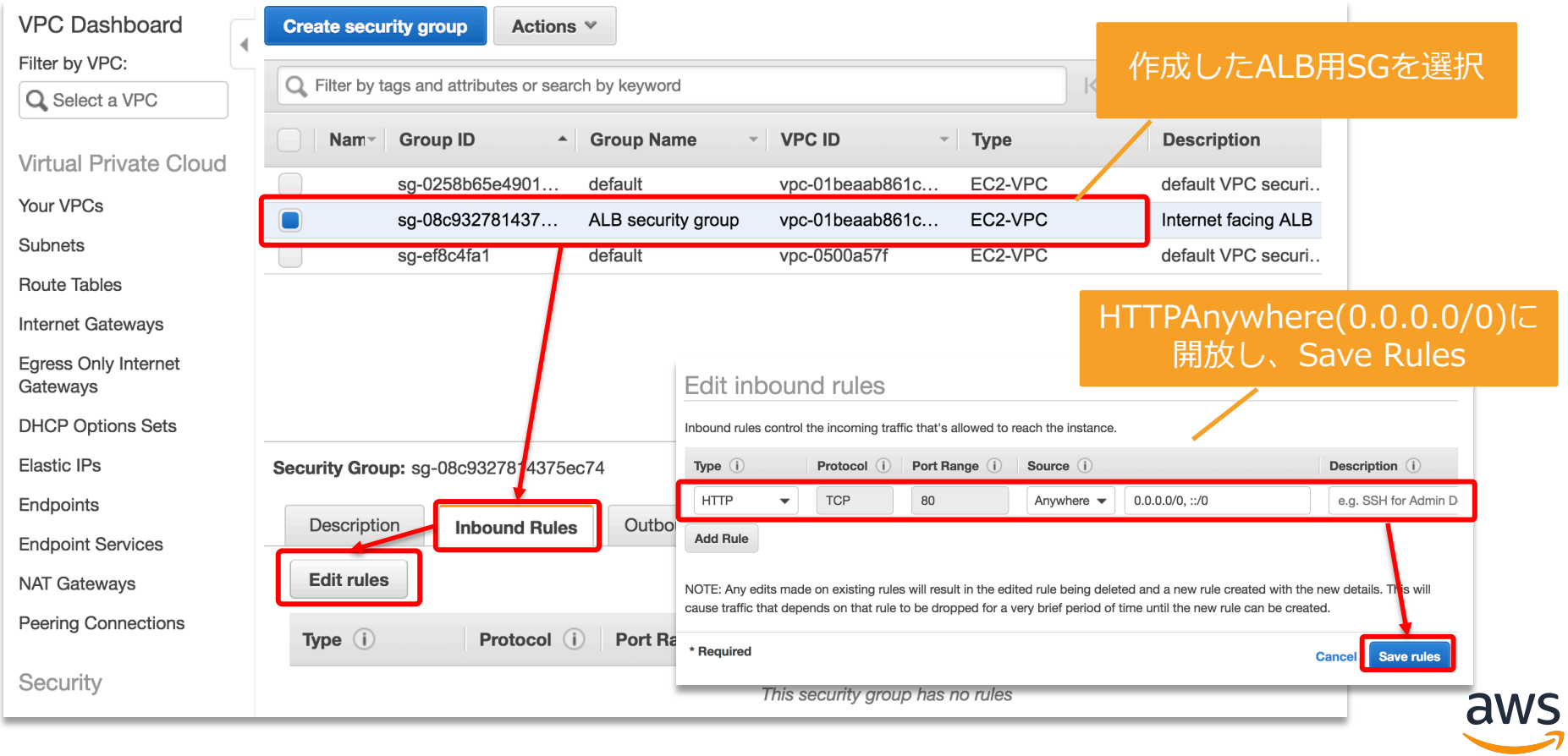
Services -> VPC へ



ALB用セキュリティグループ作成(1)

CloudFormationで作成されたVPC(handson-vpc)を選択
Cloud9の作成時に選択するのでvpc idを保存しておく

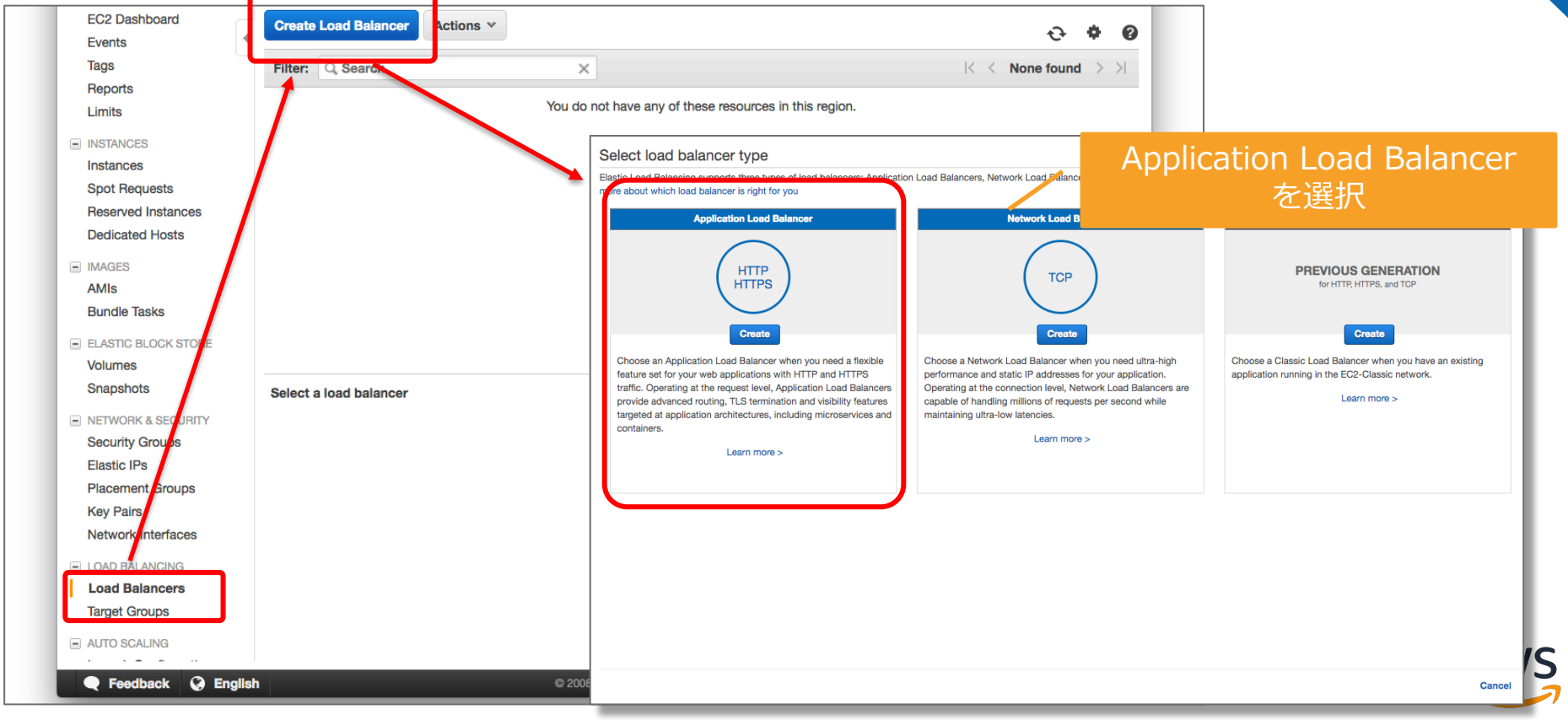
ALB用セキュリティグループ作成(2)



ALB(Application Load Balancer)作成

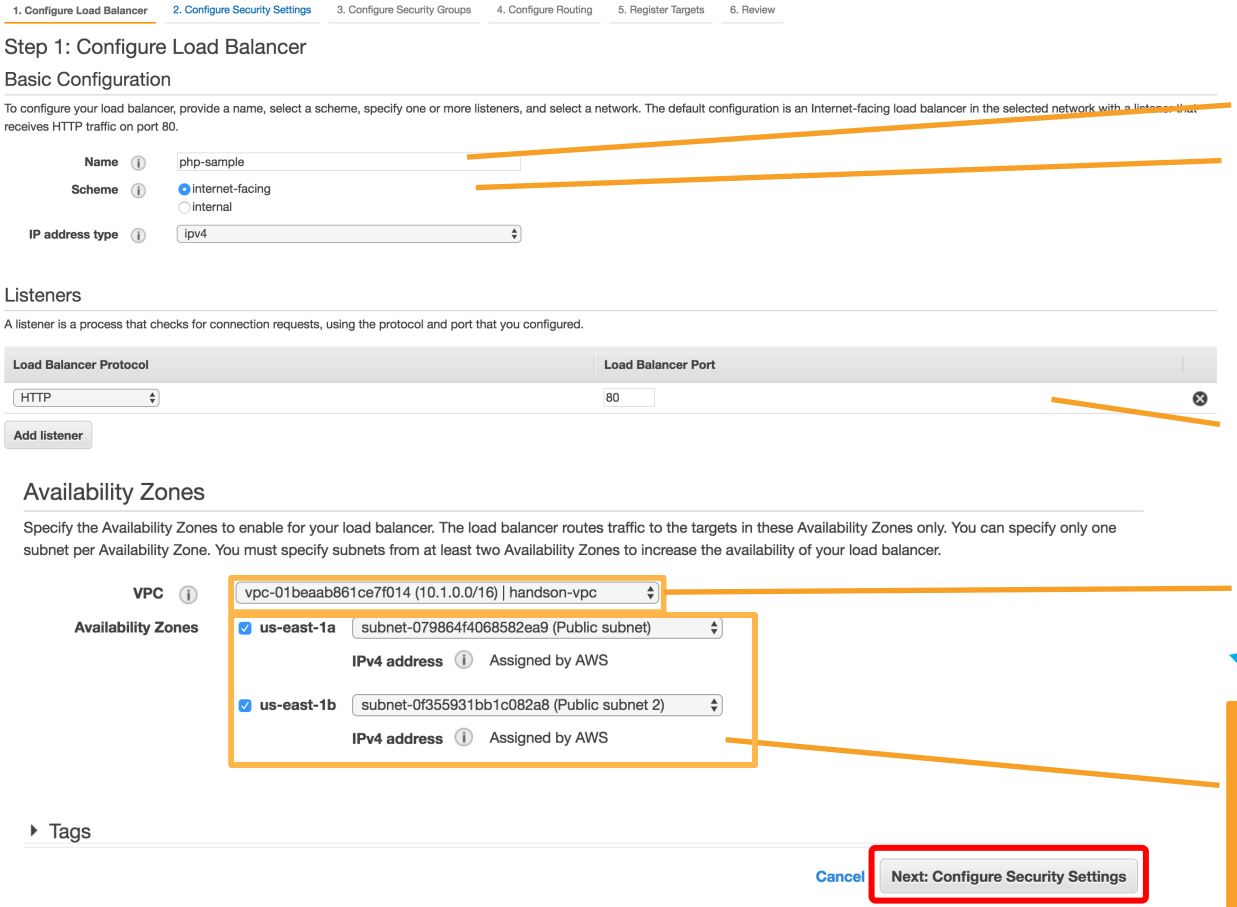
EC2 コンソール

Services -> EC2 へ



Application Load Balancer
を選択

基本設定



Nameは php-sample

Internet-facingを選択

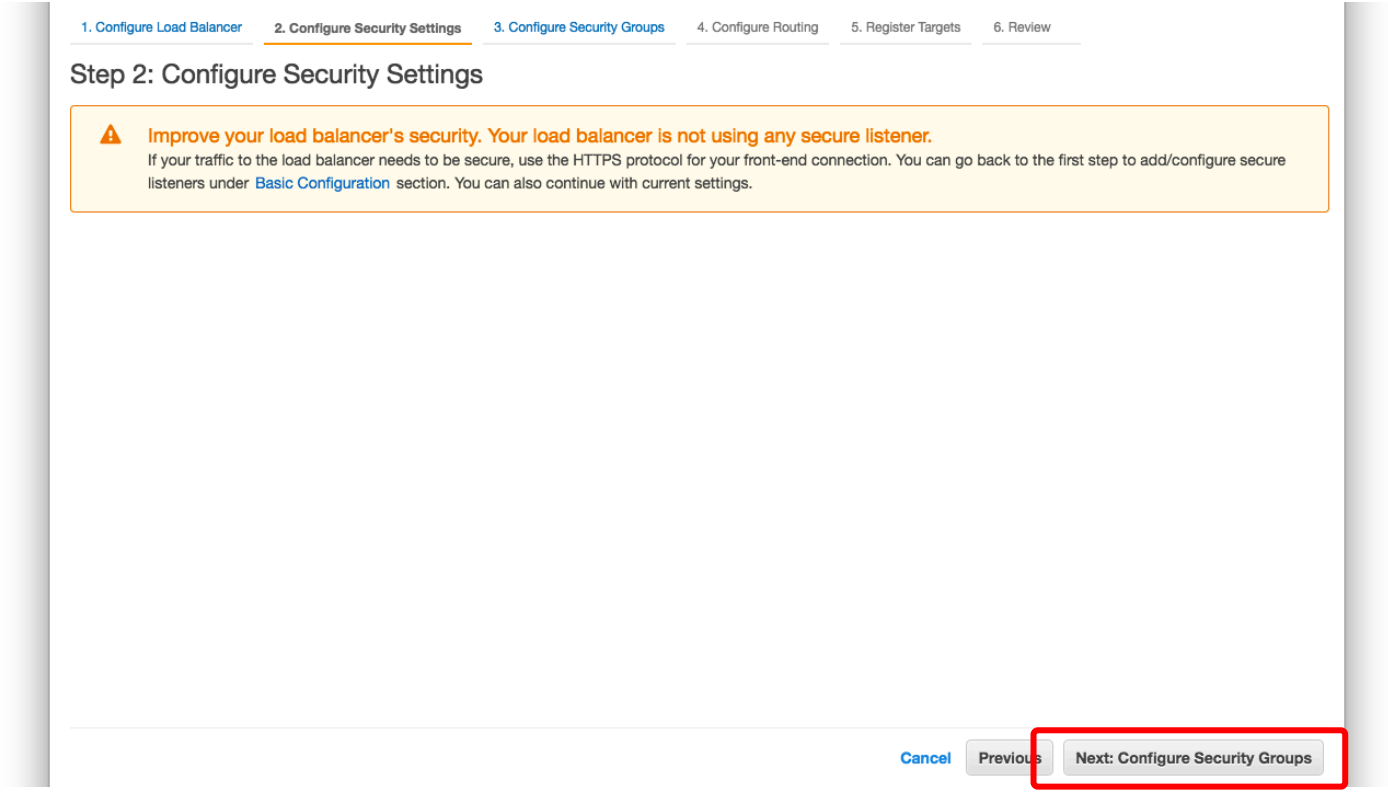
Listenerはデフォルト

handsonで作成したvpcを選択

よくある間違いポイント

サブネットにはus-east-1aの
Public subnetとus-east-1bの
Public subnet 2 を選択
Cloud9作成時のためにsubnet id
を記録しておく

セキュリティ設定(Nextをクリック)



セキュリティグループ設定

1. Configure Load Balancer | 2. Configure Security Settings | 3. Configure Security Groups | 4. Configure Routing | 5. Register Targets | 6. Review

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group:

- ○ Create a **new** security group
- ◉ Select an **existing** security group

Filter: VPC security groups

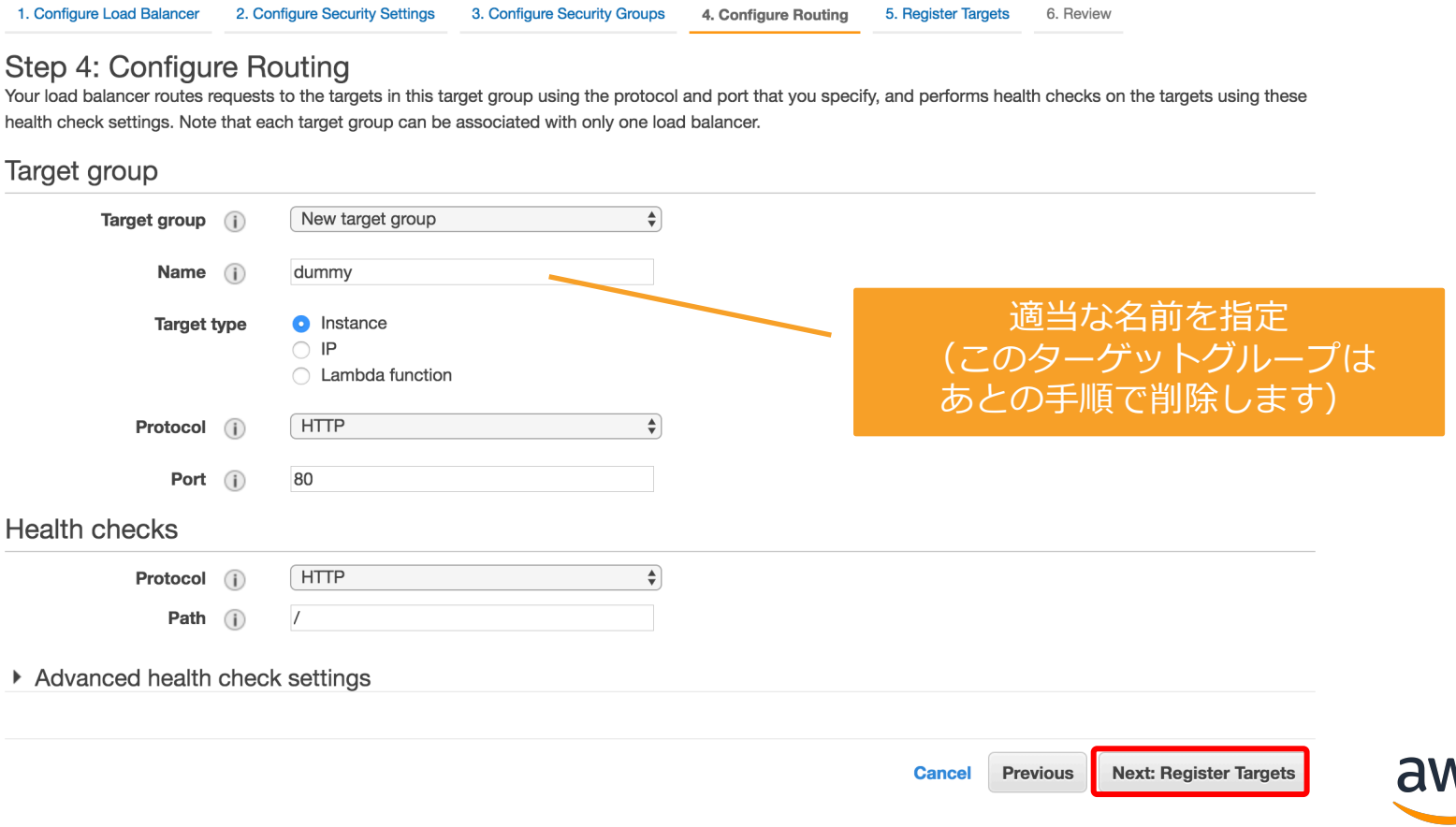
	Security Group ID	Name	Description	Actions
☑	sg-08854e7c10ef43eae	ALB security group	Internet facing ALB	Copy to new
☑	sg-0f76a8c692156303c	default	default VPC security group	Copy to new

★ よくある間違いポイント

先程作成したALB用のSGと
defaultのSGを **両方**チェック

Cancel | Previous | Next: Configure Routing

ルーティング設定(スキップ)



ターゲットレジスター(スキップ)

1. Configure Load Balancer | 2. Configure Security Settings | 3. Configure Security Groups | 4. Configure Routing | 5. Register Targets | 6. Review

Step 5: Register Targets

Register targets with your target group. If you register a target in an enabled Availability Zone, the load balancer starts routing requests to the targets as soon as the registration process completes and the target passes the initial health checks.

Registered targets

To deregister instances, select one or more registered instances and then click Remove.

Remove

Instance	Name	Port	State	Security groups	Zone
No instances available.					

Instances

To register additional instances, select one or more running instances, specify a port, and then click Add. The default port is the port specified for the target group. If the instance is already registered on the specified port, you must specify a different port.

Add to registered on port 80

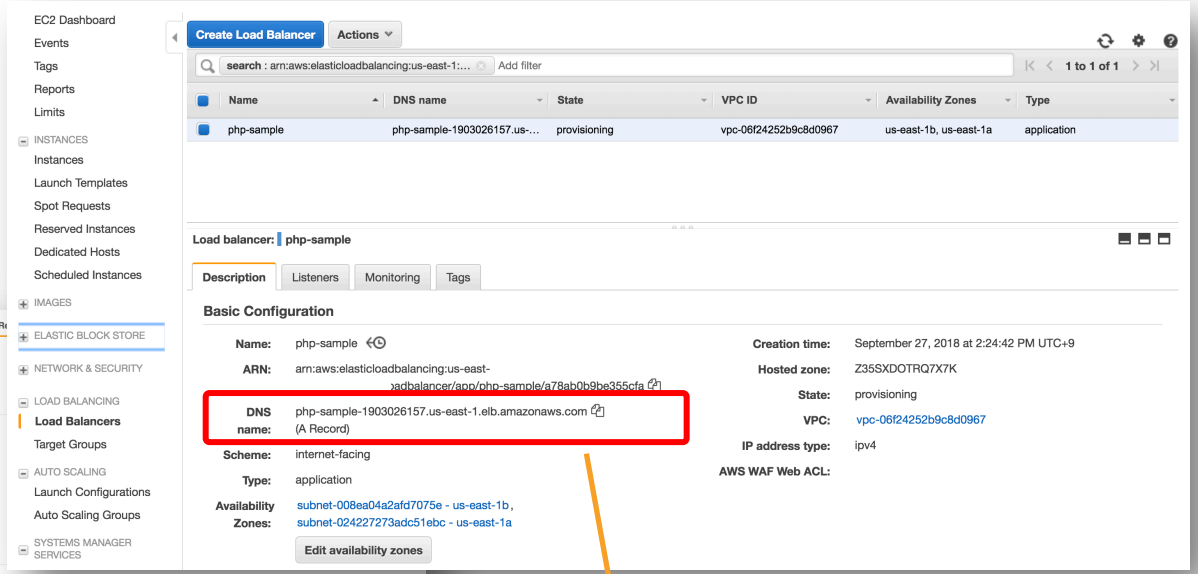
Search Instances

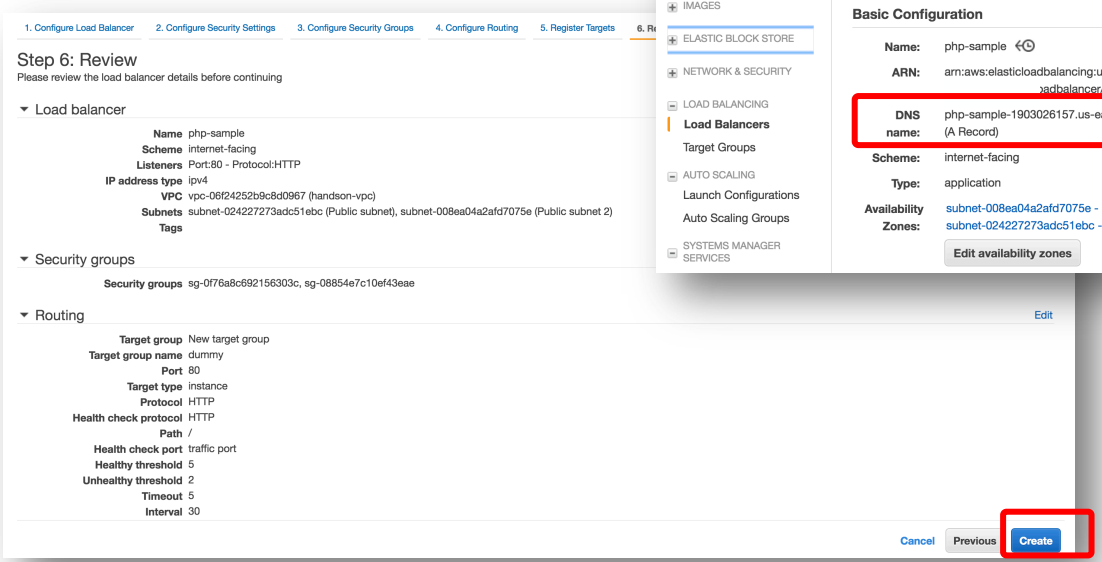
Instance	Name	State	Security groups	Zone	Subnet ID	Subnet CIDR

Cancel | Previous | Next: Review

aws

ALB作成

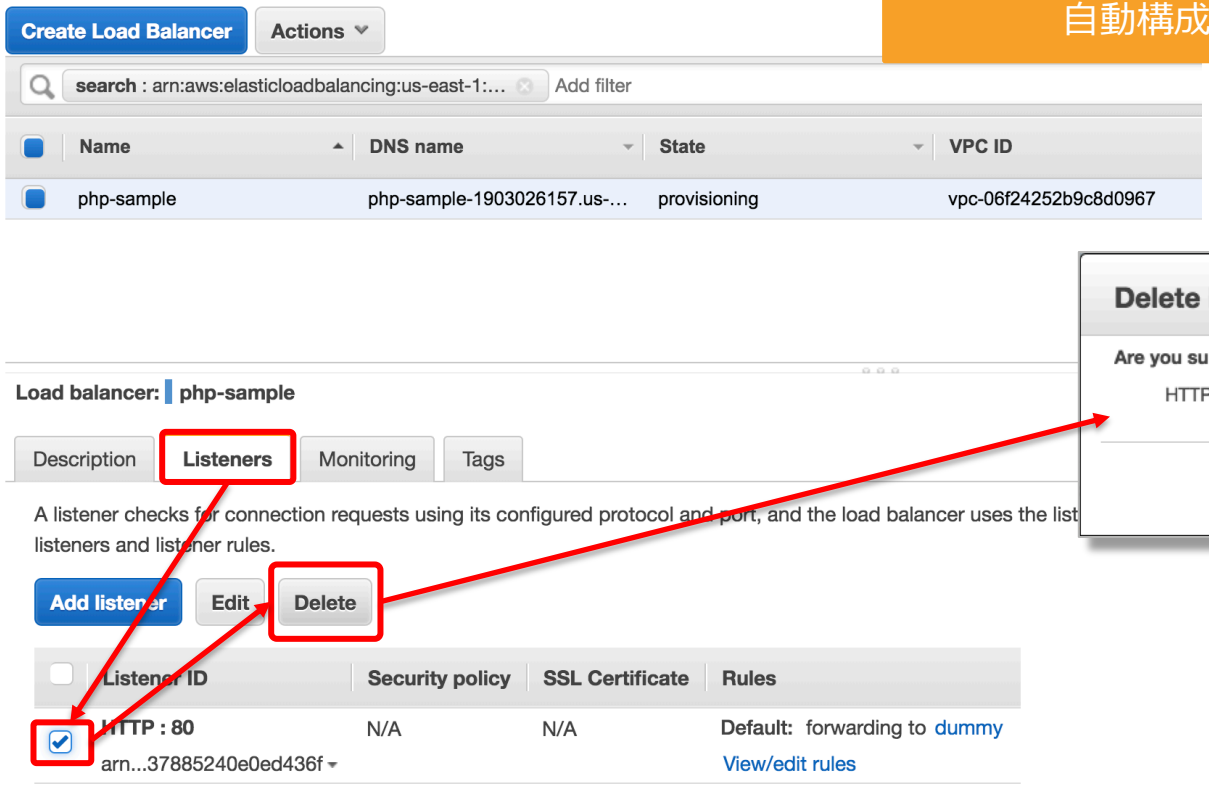




アプリケーションのエンドポイントとなるDNS名をメモしておく

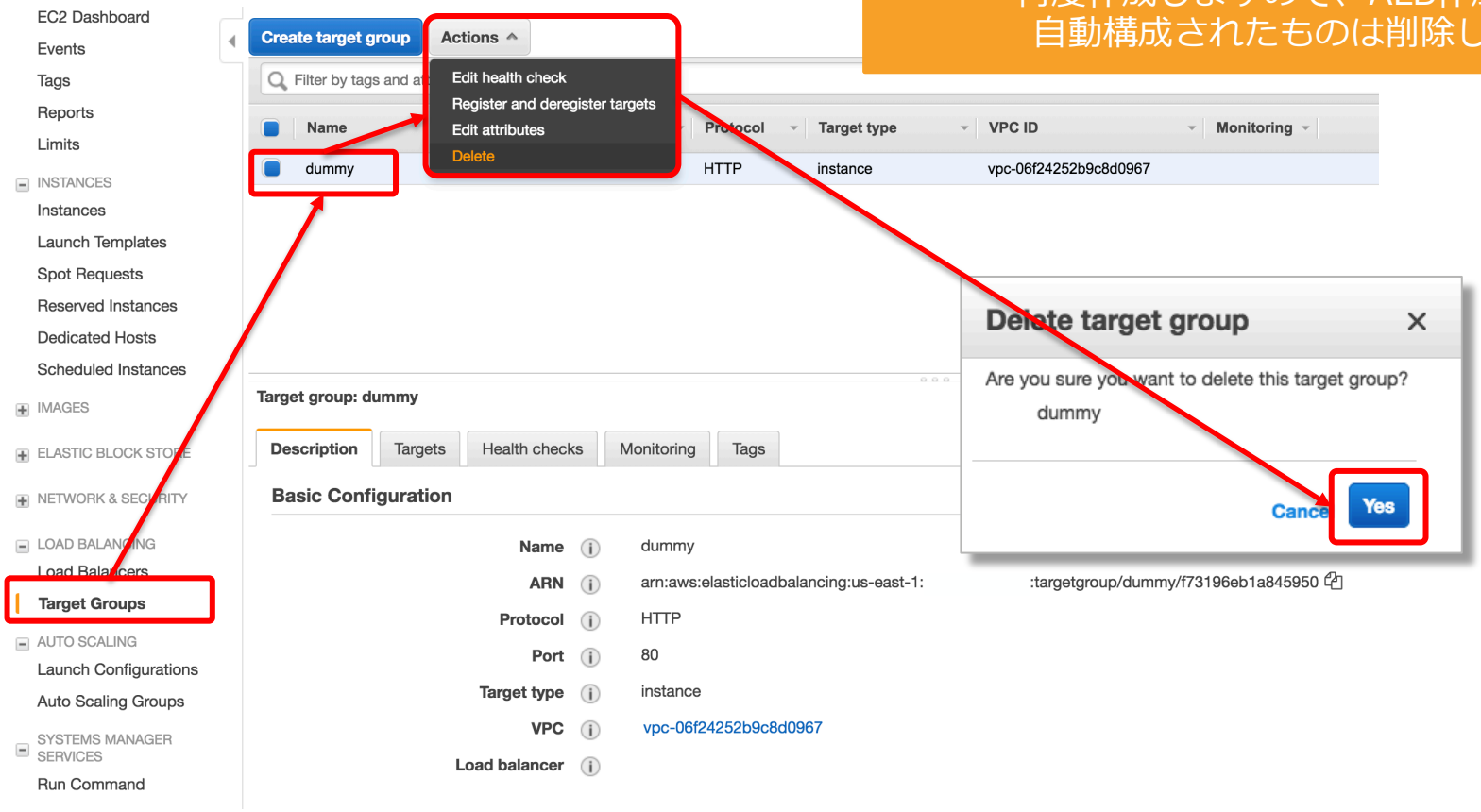
ALBリスナー削除

リスナーおよびターゲットはECSサービス構成時に
再度作成しますので、ALB作成時に
自動構成されたものは削除します



ターゲットグループ削除

リスナーおよびターゲットはECSサービス構成時に
再度作成しますので、ALB作成時に
自動構成されたものは削除します

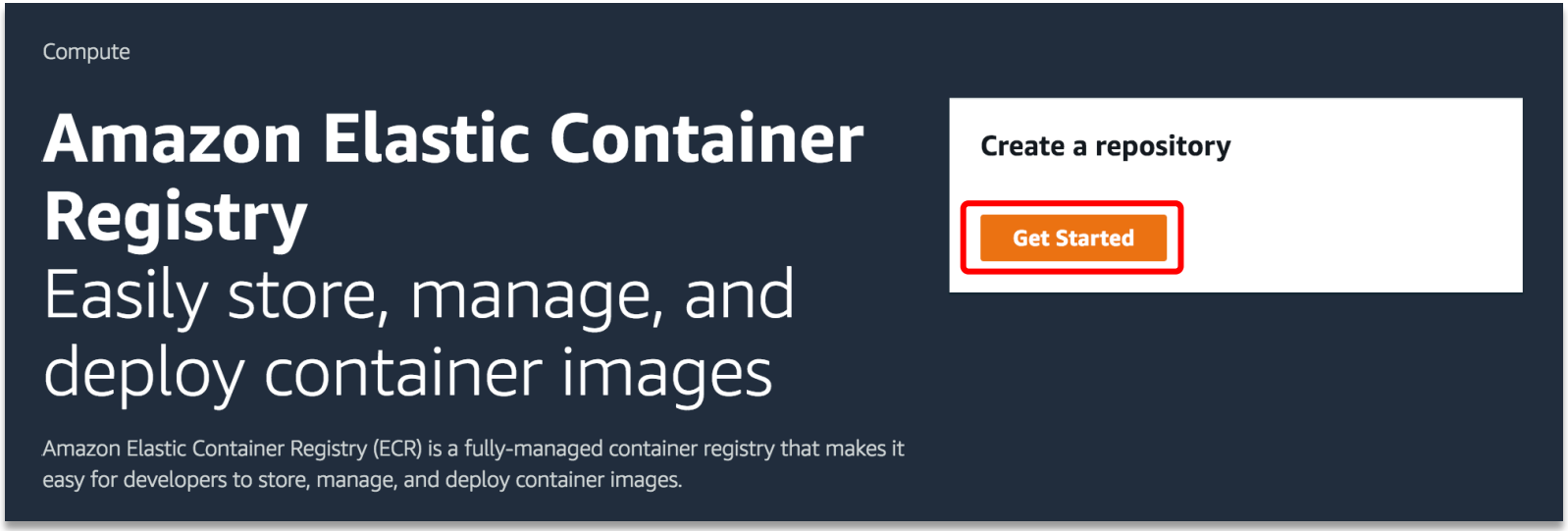


Dockerアプリ開発

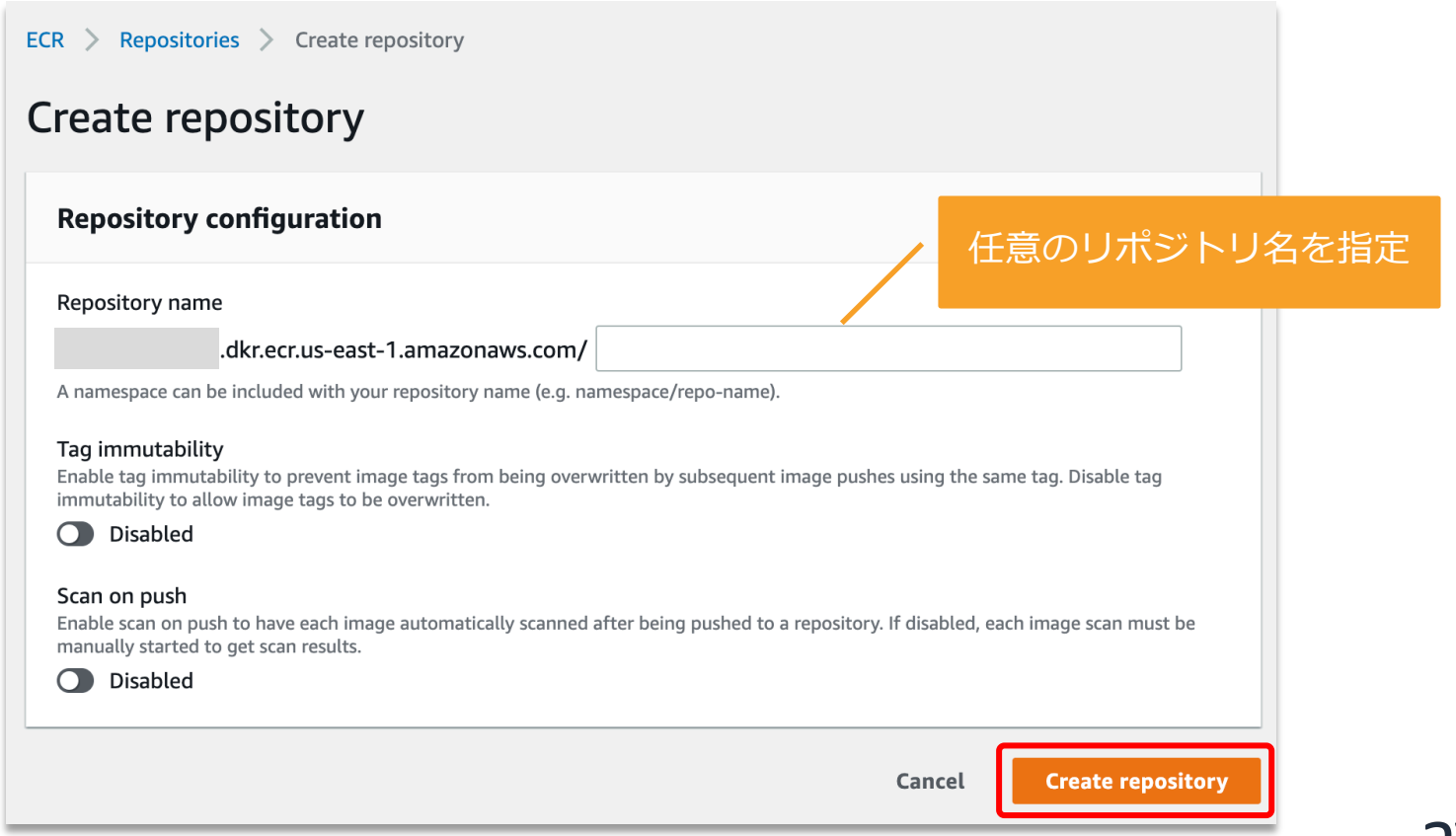
ECRリポジトリ作成

Services -> Elastic Container Registry へ

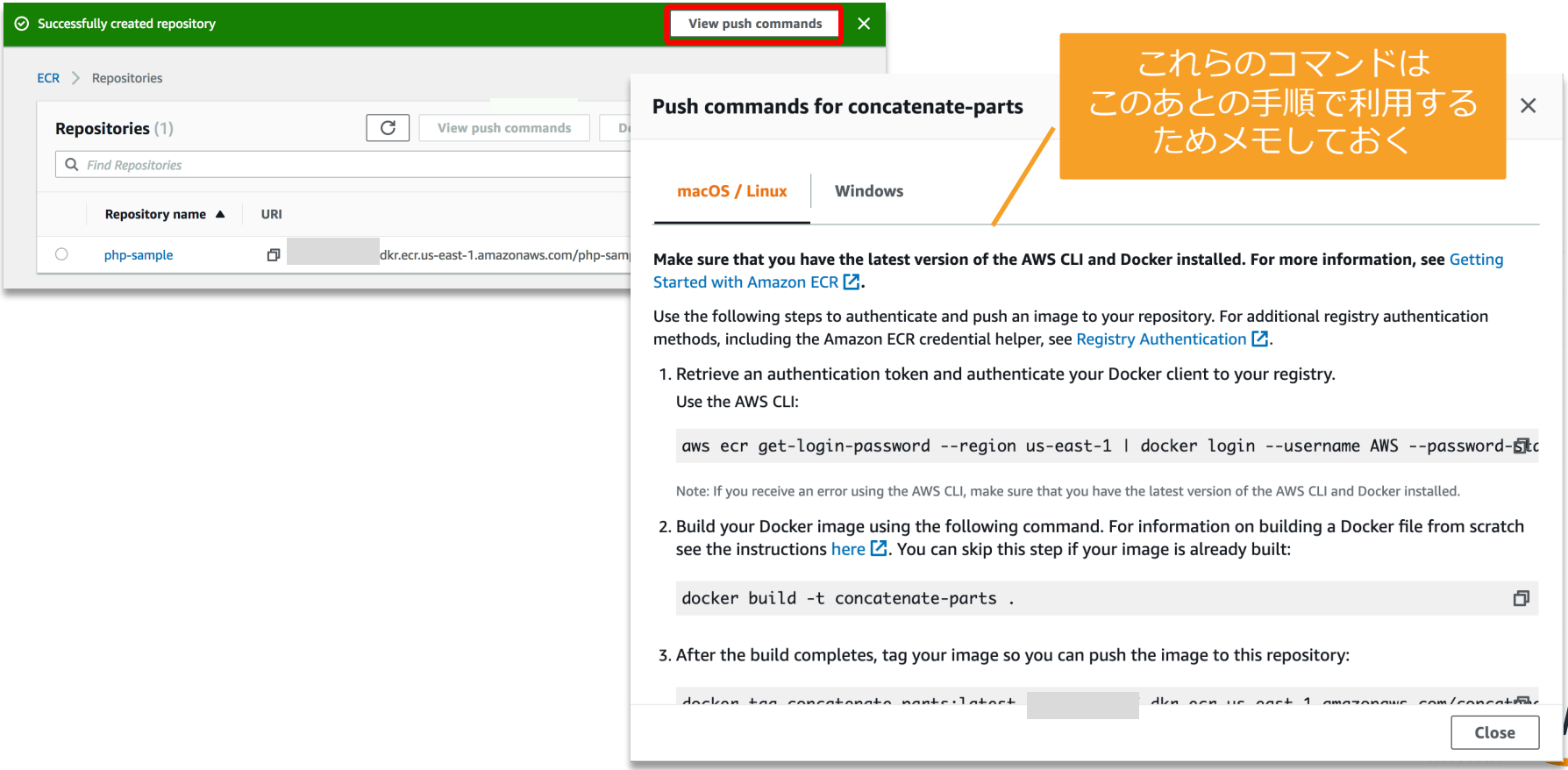
ECR コンソール



ECRリポジトリ作成


ECR > Repositories > Create repository
Create repository
Repository configuration
任意のリポジトリ名を指定
Repository name
.dkr.ecr.us-east-1.amazonaws.com/
A namespace can be included with your repository name (e.g. namespace/repo-name).
Tag immutability
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.
Disabled
Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.
Disabled
Cancel
Create repository

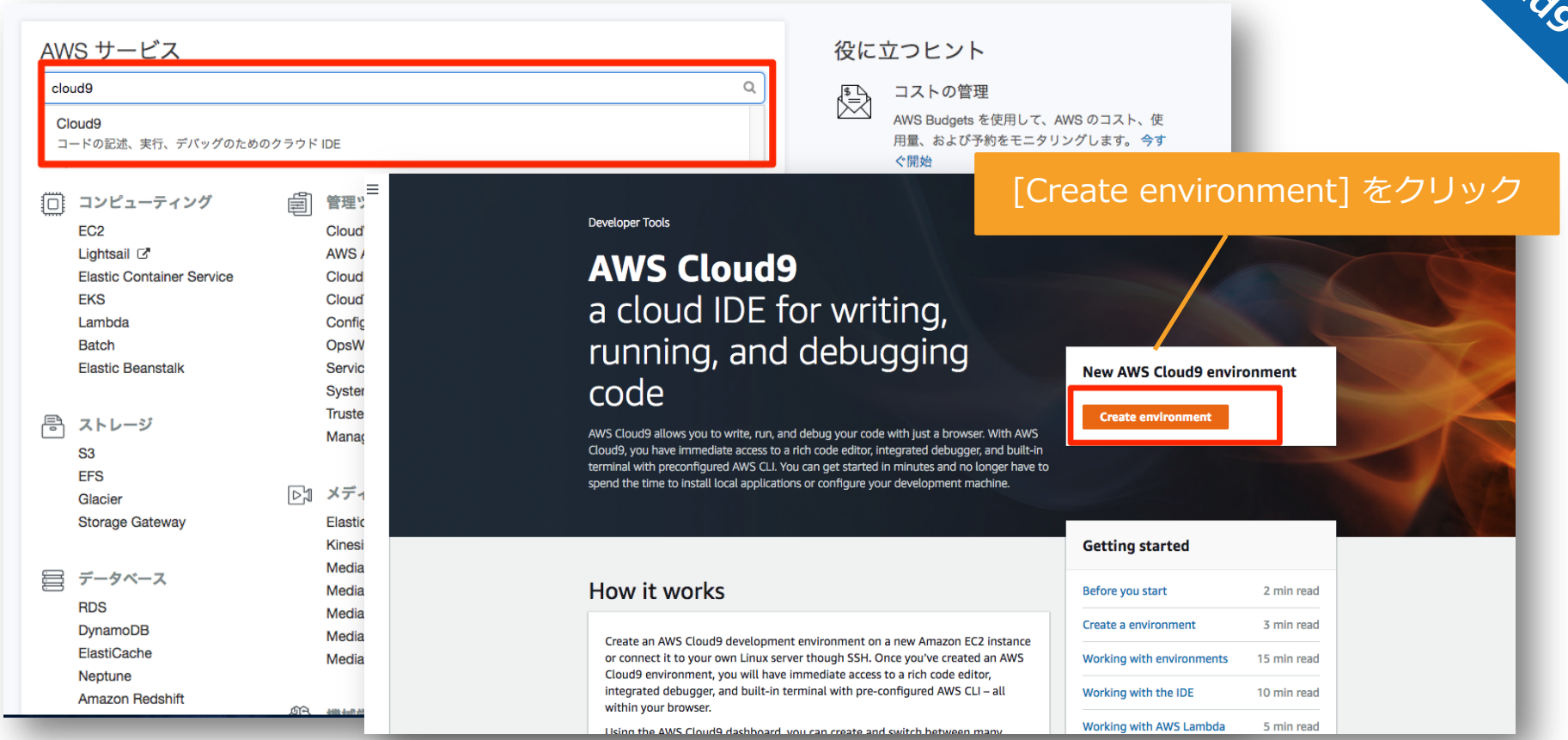
ECRリポジトリ作成


Successfully created repository
View push commands
ECR > Repositories
Repositories (1)
View push commands
Find Repositories
Repository name
URI
php-sample
dkr.ecr.us-east-1.amazonaws.com/php-sam
Push commands for concatenate-parts
これらのコマンドは
このあとの手順で利用する
ためメモしておく
macOS / Linux
Windows
Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.
Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.
1. Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-std
Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:
docker build -t concatenate-parts .
3. After the build completes, tag your image so you can push the image to this repository:
Close

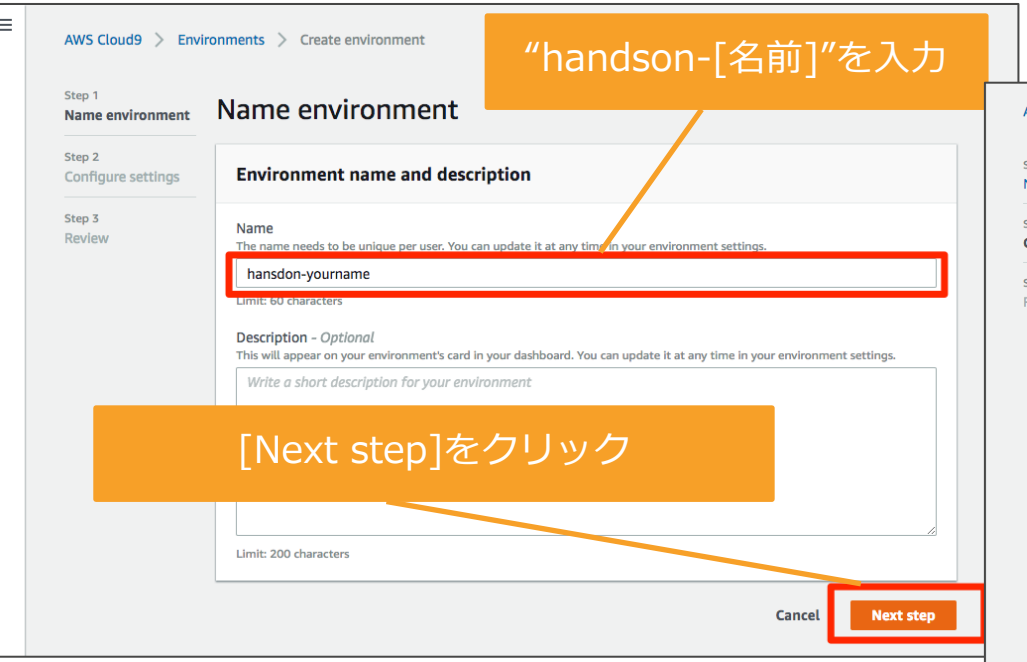
AWS Cloud9環境の構築

AWS Cloud9環境の構築

Cloud9

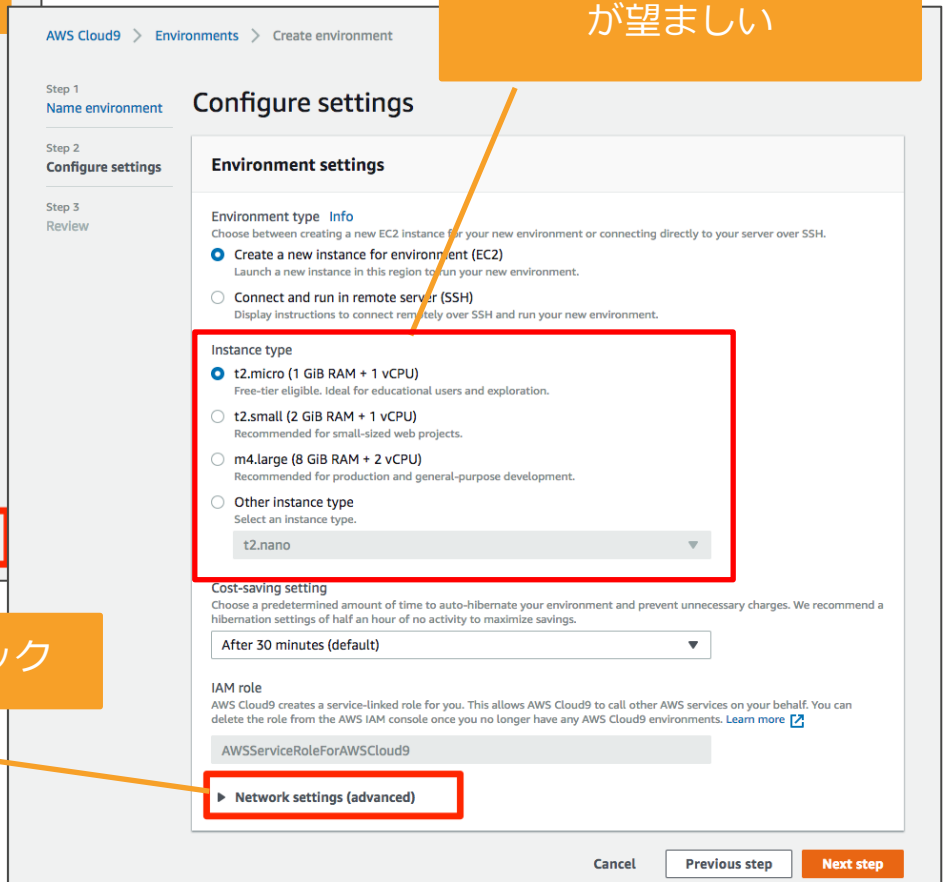


AWS Cloud9環境の構築



“handson-[名前]”を入力

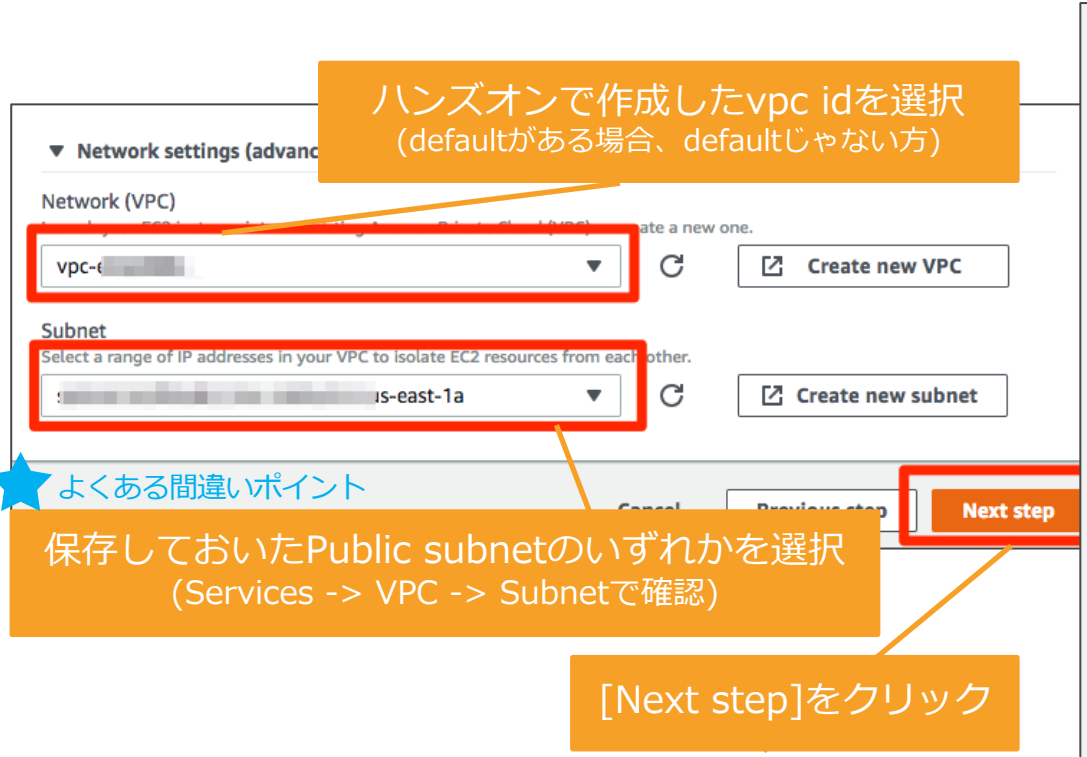
[Next step]をクリック

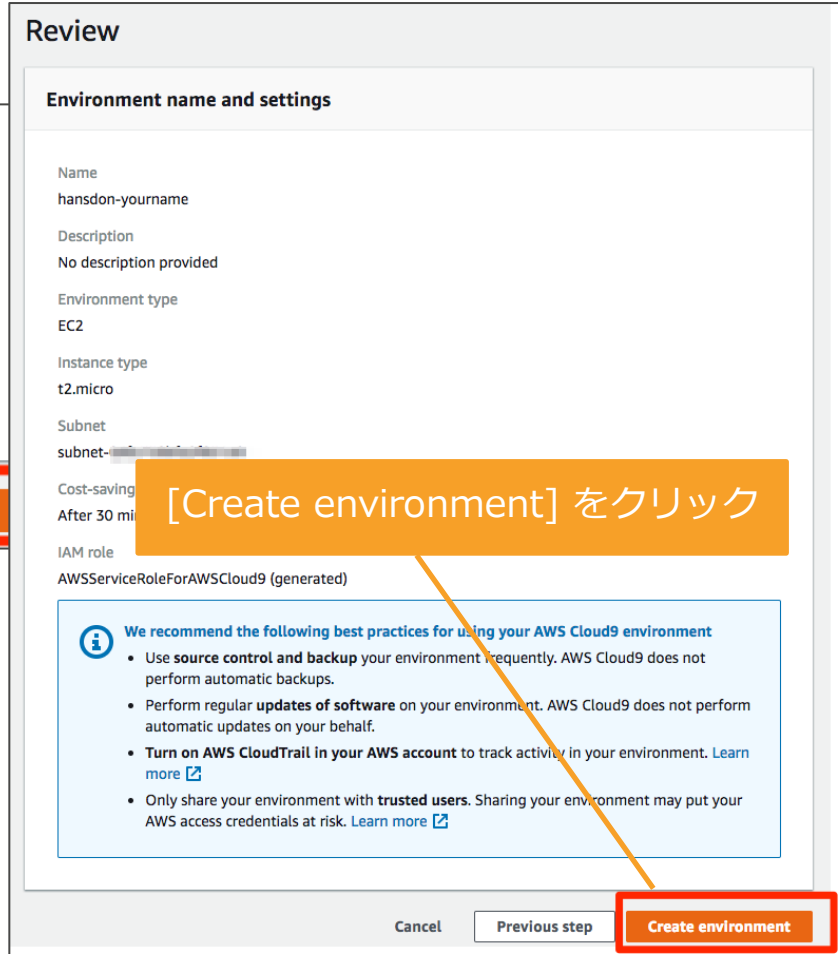


※ docker build 時などでメモリが不足することがあるので、t3.small 以上が望ましい

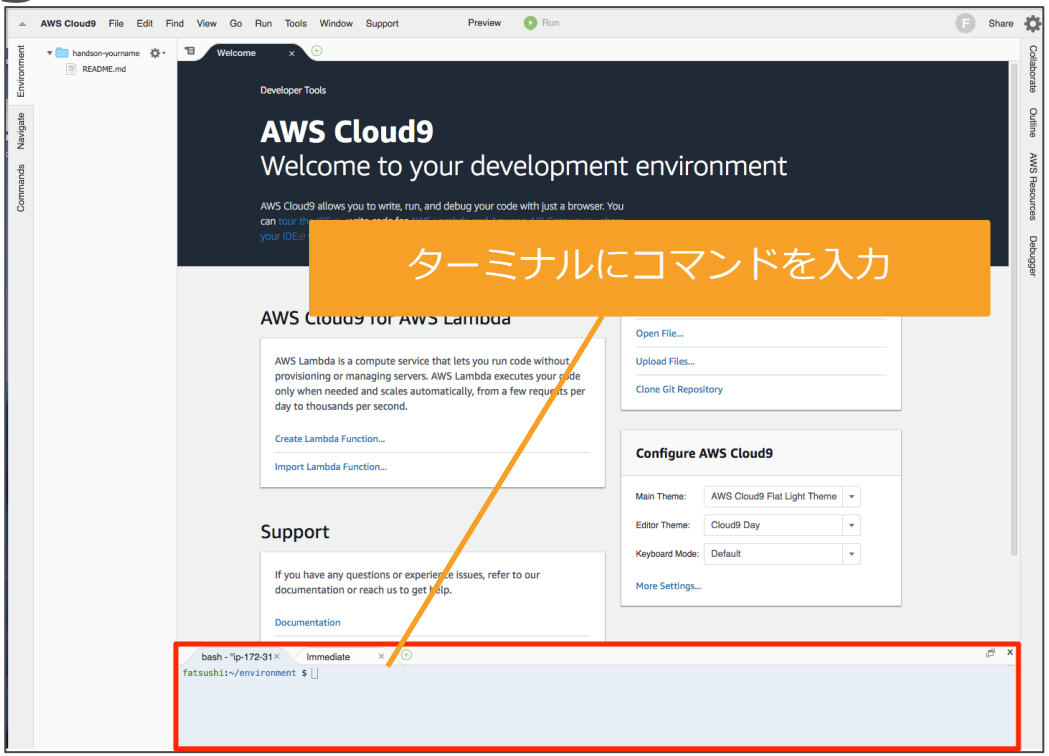
[Network settings(advanced)]をクリック

AWS Cloud9環境の構築



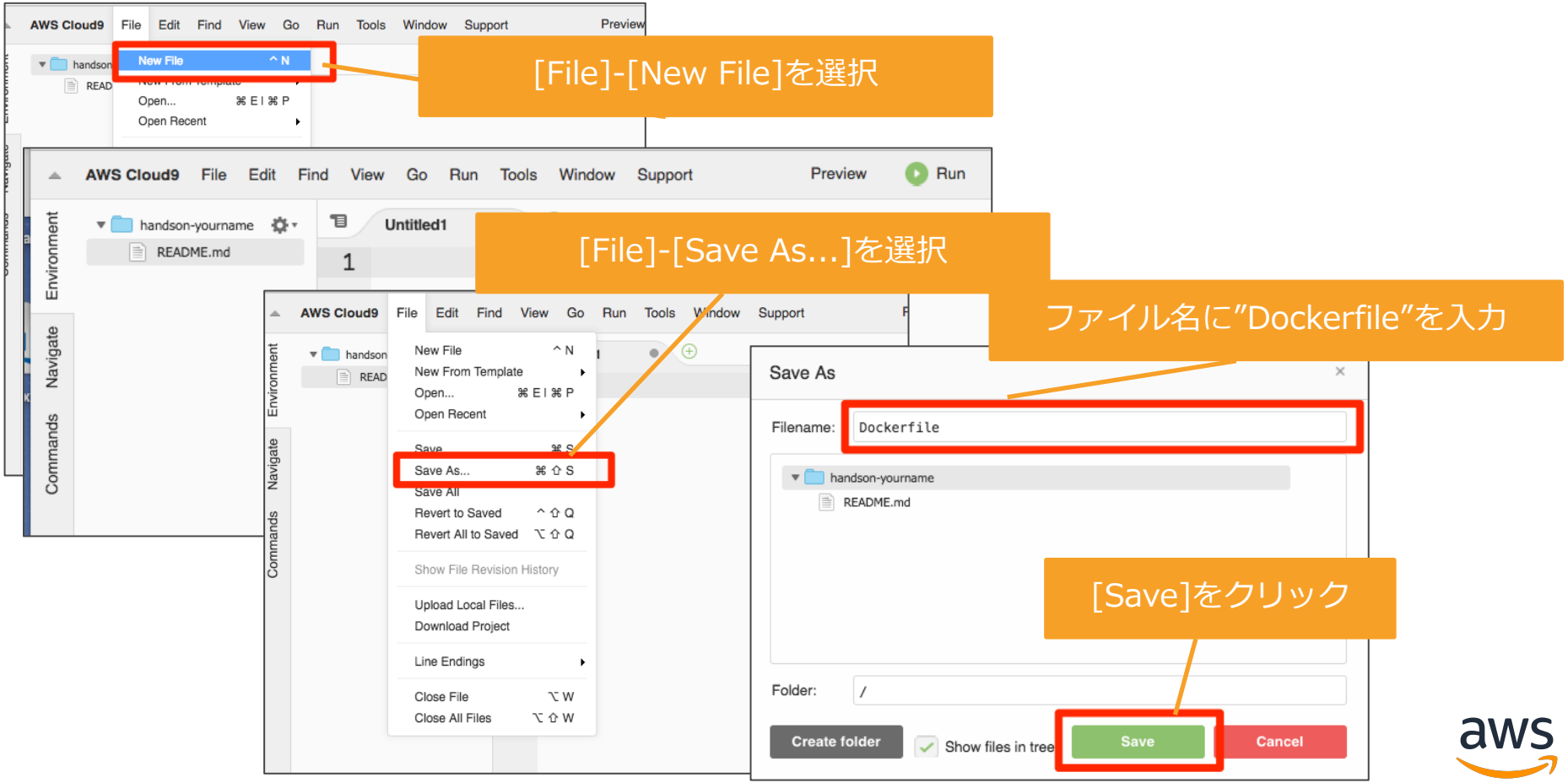


git環境の設定




```
$ git config --global user.name ”yourname”
$ git config --global user.email yourname@abc.com
$ git config --global credential.helper '!aws codecommit credential-helper $@'
$ git config --global credential.UseHttpPath true
```

Dockerfileの作成



Dockerfile作成(PHPサンプルアプリ)

1. 前ページで作成したDockerfileに以下の内容を入力して保存

```
FROM php:7.4.0-apache
COPY src/ /var/www/html/
```

Dockerfile

Docker Hubの公式イメージから
PHP/apacheのDockerイメージを取得
https://hub.docker.com/_/php/

2. handson-xxxx配下に、srcディレクトリとindex.phpファイルを作成

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <title>PHP Sample</title>
  </head>
  <body>
    <?php echo gethostname(); ?>
  </body>
</html>
```

src/index.php

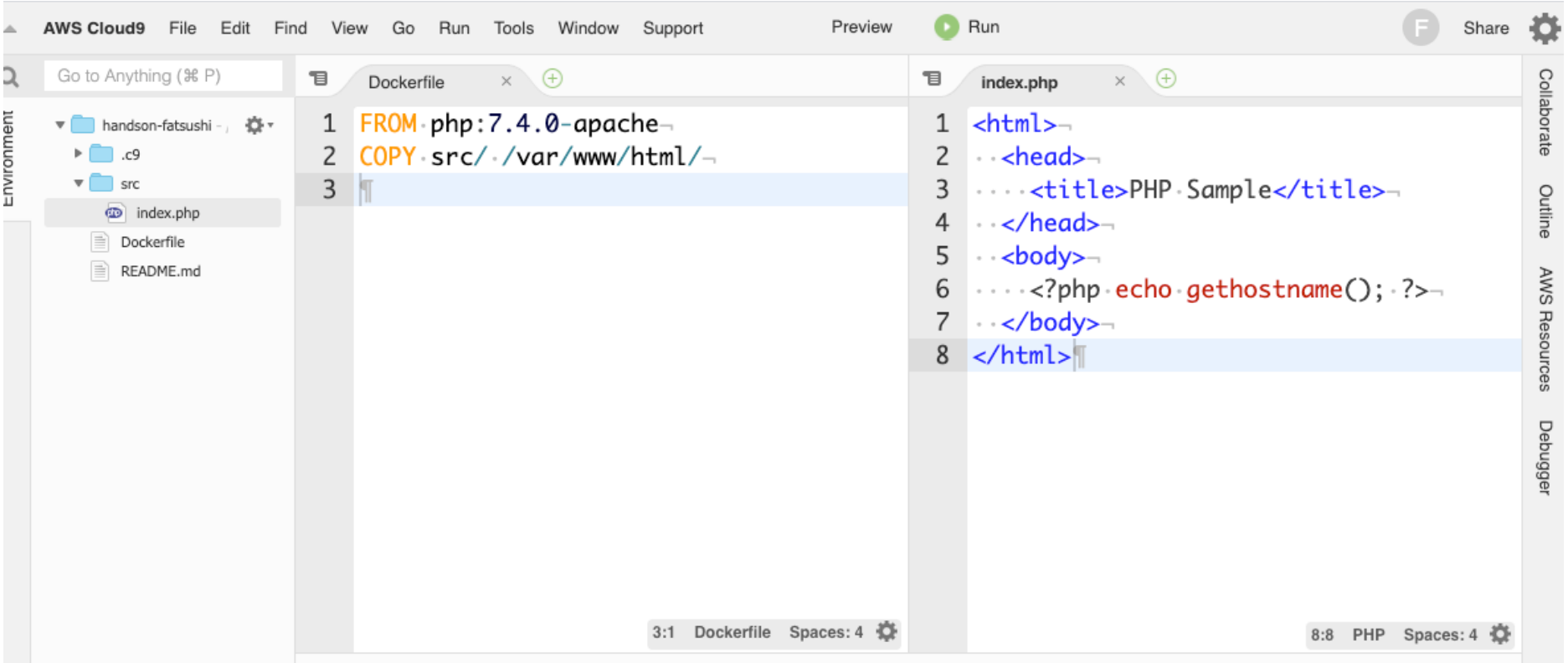
ローカルのsrcディレクトリ配下を
コンテナ内の/var/www/html
ディレクトリにコピー

ホスト名を表示するPHPアプリ
(余裕のある方は好きなコードを書いてください)

※お急ぎの方はZipファイル内のサンプルコードをご利用ください。

aws

Dockerfile作成(PHPサンプルアプリ) – 例



Dockerイメージ作成 ～ コンテナ起動

Cloud9ターミナルで作業

1. Dockerイメージ作成

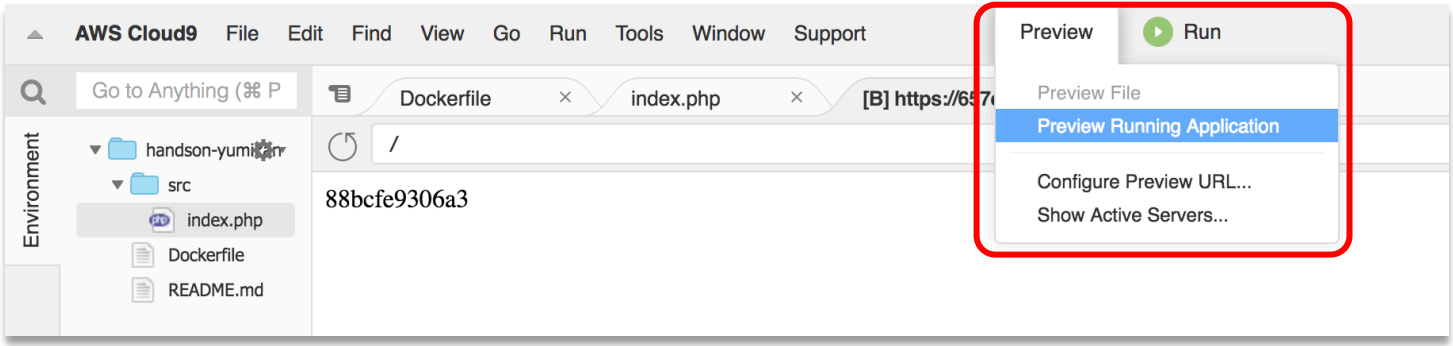
```
$ docker build -t php-sample . #メモしておいたコマンド
$ docker images
REPOSITORY      TAG       IMAGE ID          CREATED              SIZE
php-sample      latest    bc47e3ede49a      3 minutes ago        390 MB
```

2. Dockerコンテナ起動

```
$ docker run --rm -p 8080:80 -d php-sample:latest #ホストPort8080でコンテナport80にアクセス
$ docker ps
CONTAINER ID  IMAGE                ~ ~ ~ PORTS                  NAMES
67776a2787dd    php-sample:latest ~ ~ ~ 0.0.0.0:8080->80/tcp  modest_wozniak
```

Dockerイメージ作成 ～ コンテナ動作確認と停止

1. [Preview]メニューの[Preview Running Application]を実行しPHPのページが表示されることを確認



2. Dockerコンテナ停止

```
$ docker stop 67776a2787dd #`docker ps`で表示されたコンテナIDを指定
```

ECRへのDockerイメージPush

ECR 作成時にメモしたコマンドを実行する

1. ECRへのログイン

```
$ $(aws ecr get-login --no-include-email)
...
Login Succeeded # ログインに成功することを確認する
```

2. タグ付け

先ほどメモしておいたコマンド

```
$ docker tag php-sample:latest XXXXXXXXXX.dkr.ecr.us-east-1.amazonaws.com/php-sample:latest
```

3. ECRへのイメージPush

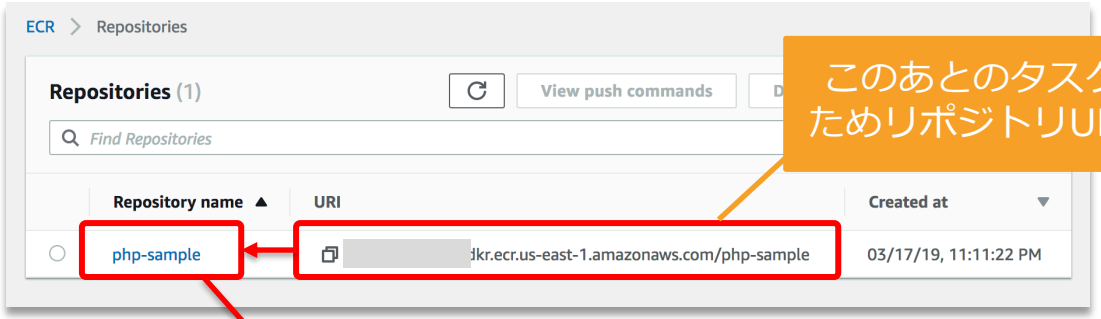
先ほどメモしておいたコマンド

```
$ docker push XXXXXXXXXX.dkr.ecr.us-east-1.amazonaws.com/php-sample:latest
...
latest: digest: sha256:... size: 3450 # イメージ Push に成功することを確認する
```

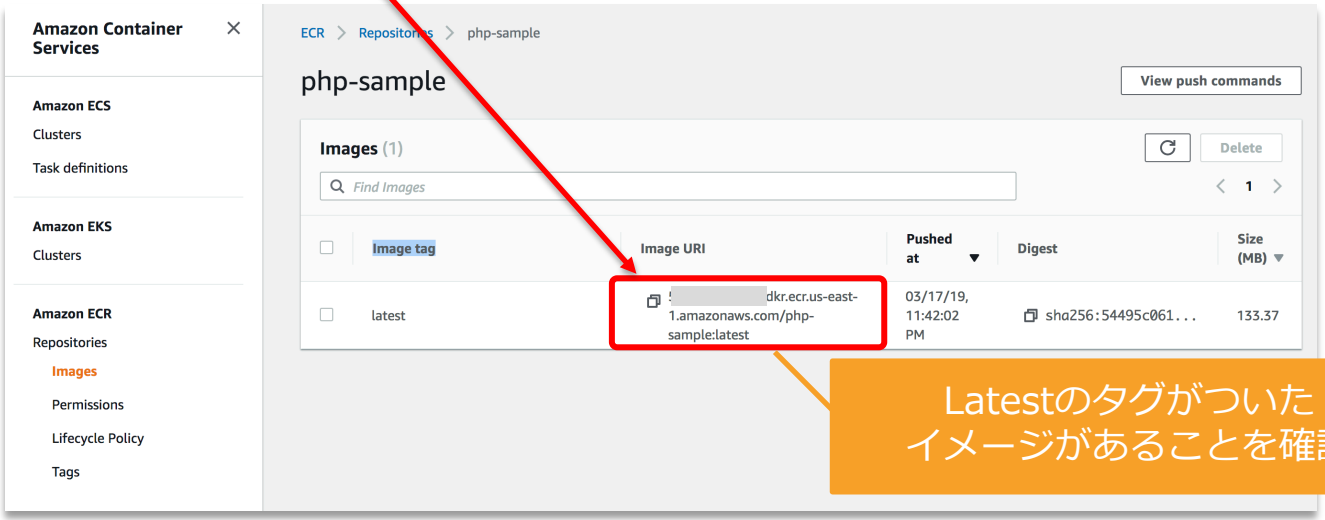
ECRコンソールでのイメージ確認

Services -> ECR へ

ECS コンソール



このあとのタスク定義で利用する
ためリポジトリURIをメモしておく



Latestのタグがついた
イメージがあることを確認

ハンズオン②

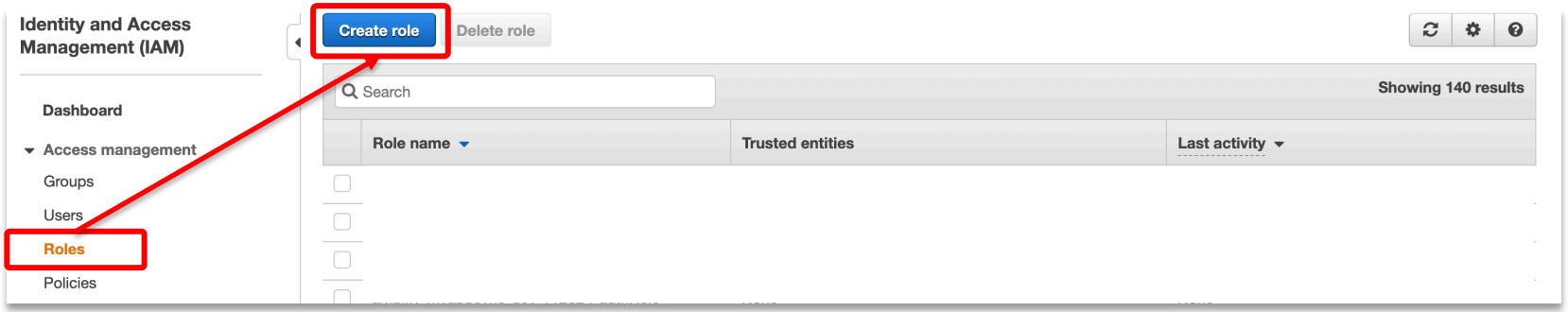
AWS Fargate環境の構築

IAMロール設定(1)

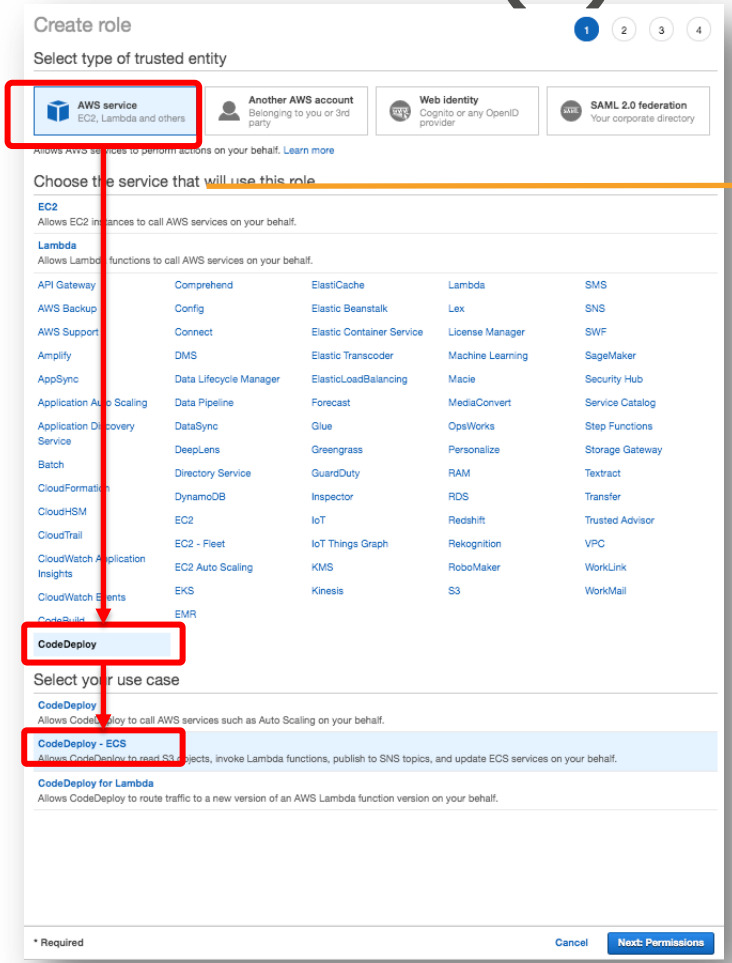
IAM コンソール

CodeDeployが利用するIAMロールを作成します。

Services -> IAM へ

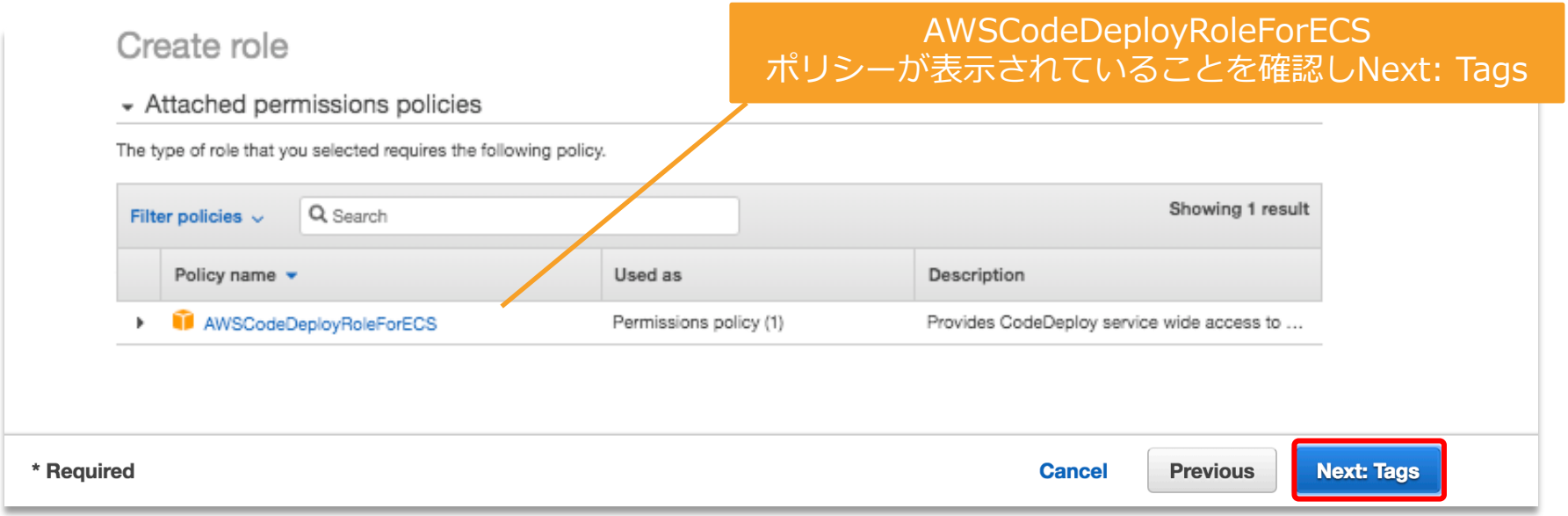


IAMロール設定(2)

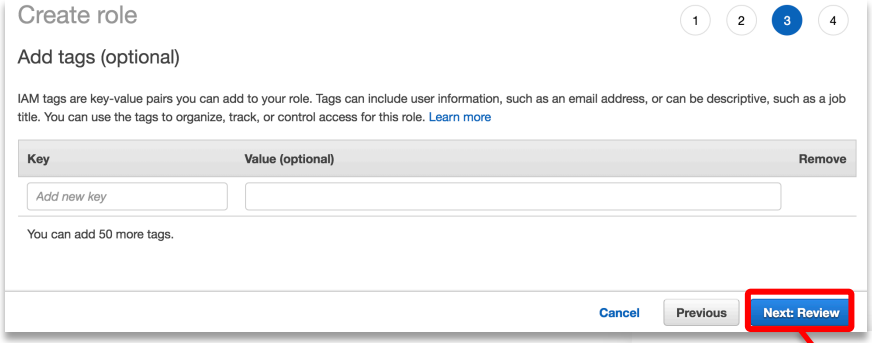


Roleタイプで
AWS Service >
CodeDeploy >
CodeDeploy – ECS を選択してから
Next: Permissionを選択

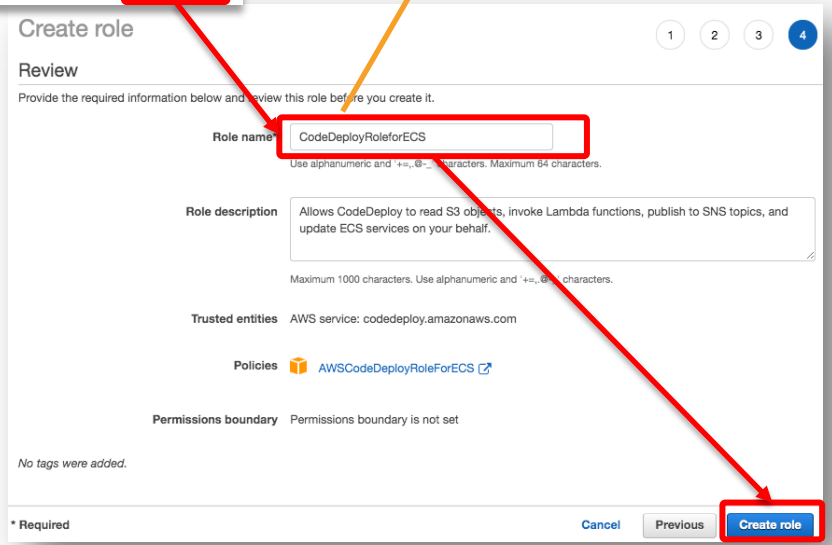
IAMロール設定(3)


AWSCodeDeployRoleForECS
ポリシーが表示されていることを確認しNext: Tags
Create role
Attached permissions policies
The type of role that you selected requires the following policy.
Filter policies
Search
Showing 1 result
Policy name
Used as
Description
AWSCodeDeployRoleForECS
Permissions policy (1)
Provides CodeDeploy service wide access to ...
* Required
Cancel
Previous
Next: Tags

IAMロール設定(4)

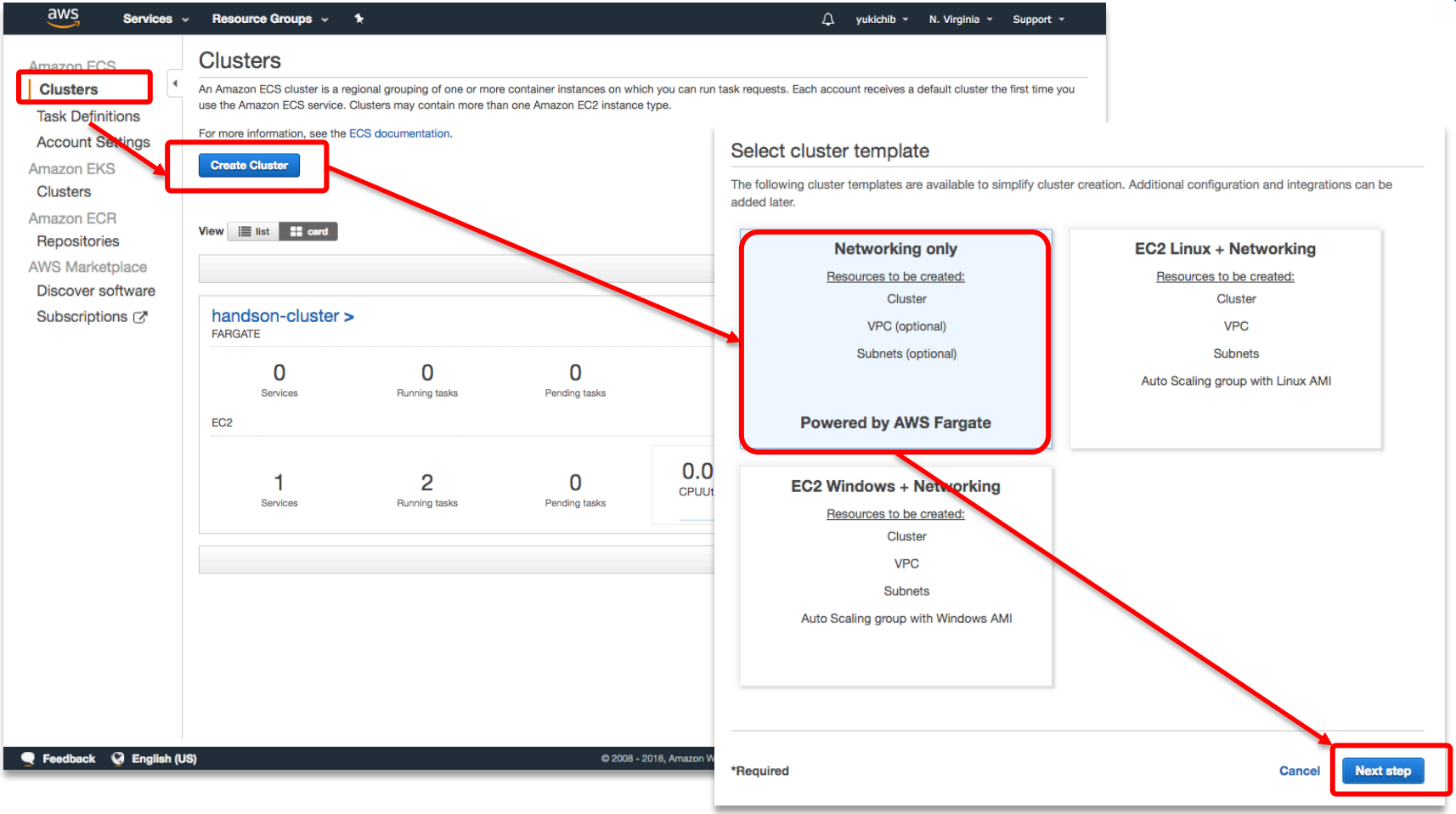


ロール名はCodeDeployRoleforECSにして
Create role

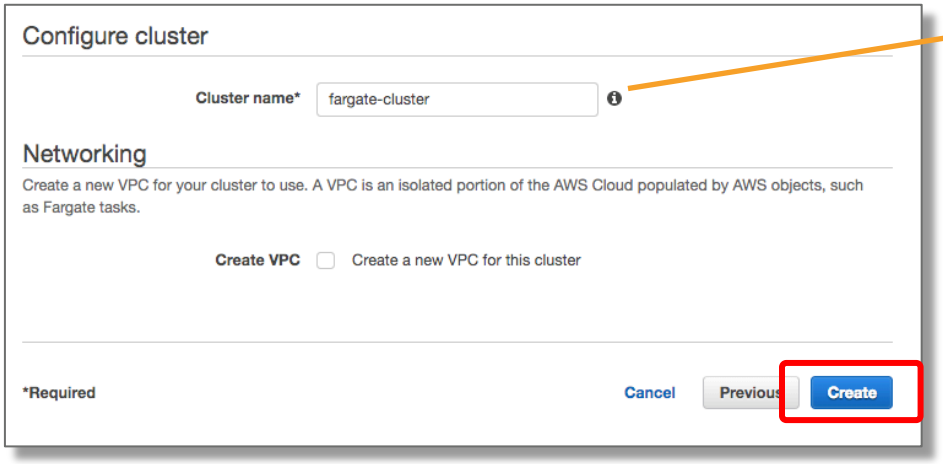


Fargateクラスター作成(1) Services->ECS

ECS コンソール

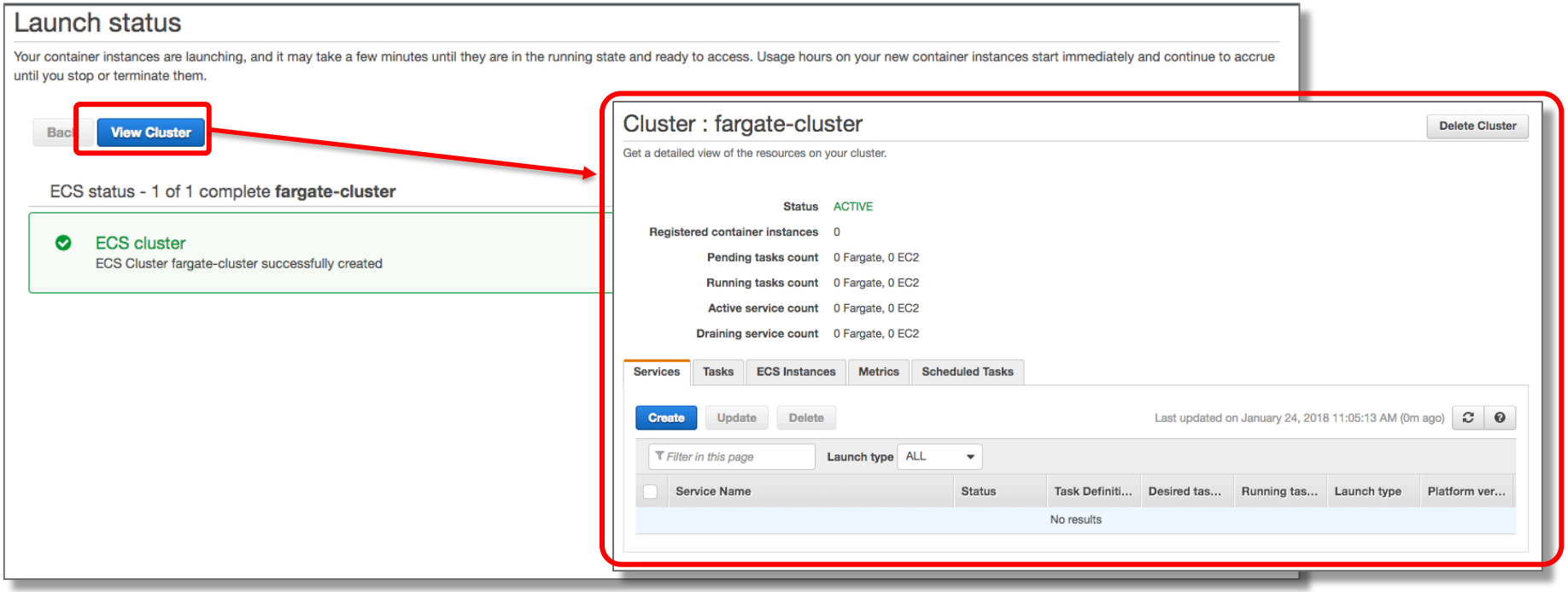


Fargateクラスター作成(2)



クラスター名を指定
(fargate-cluster)

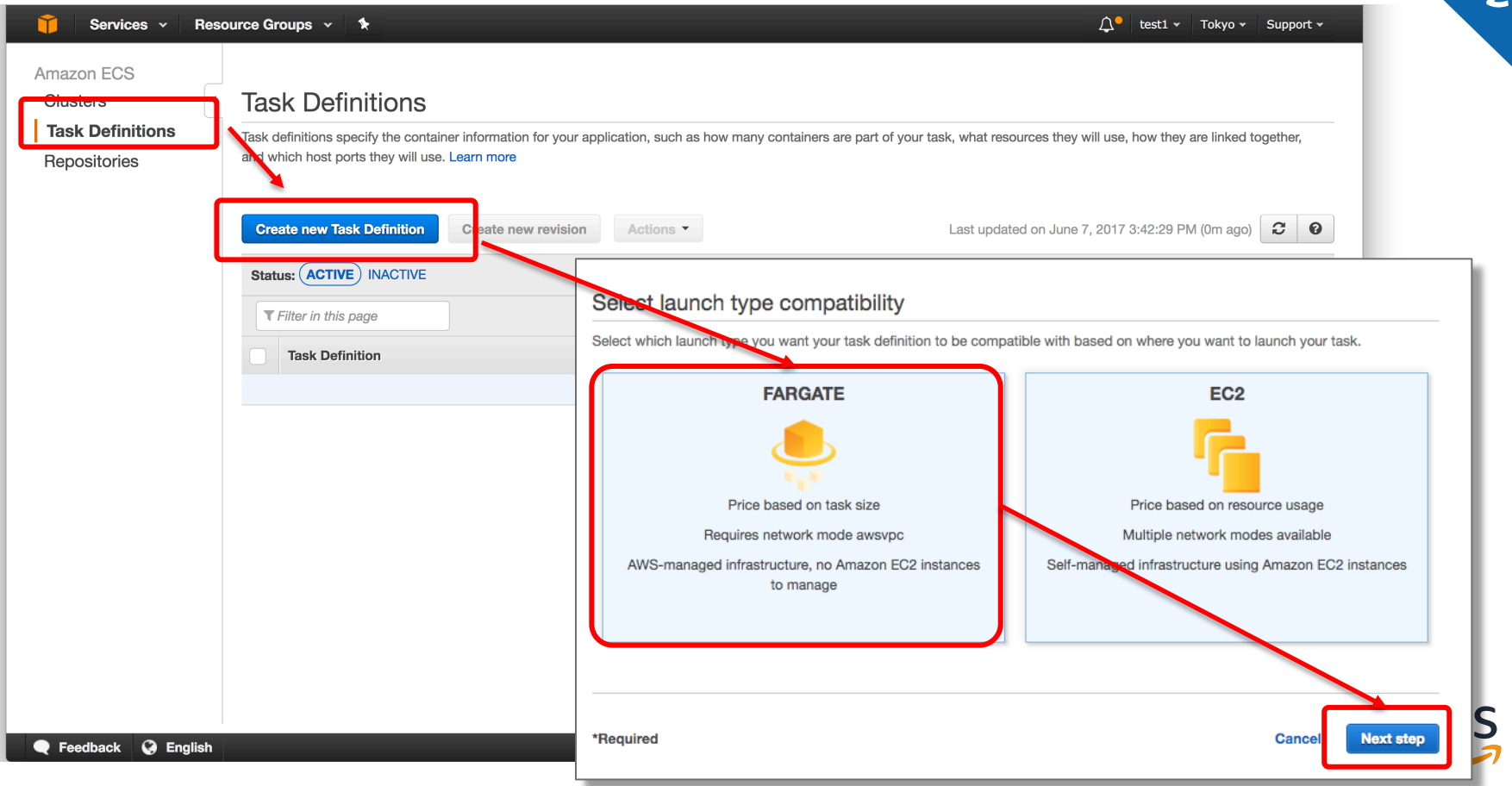
Fargateクラスターの確認



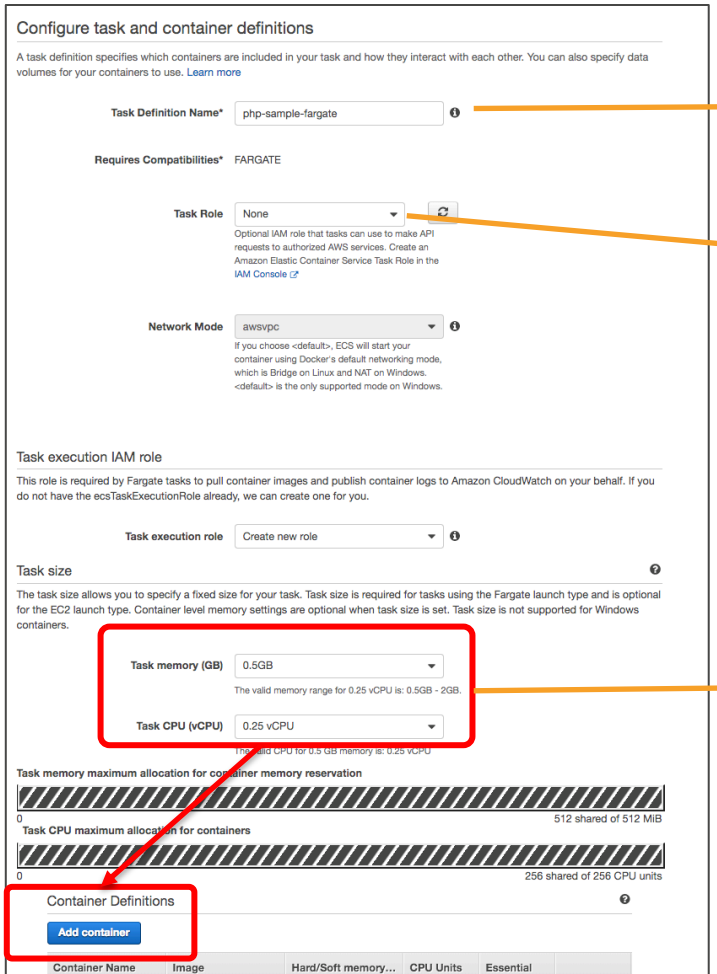
Fargate用タスク定義作成(1)

Services -> ECS へ

ECS コンソール



タスク定義作成(2)

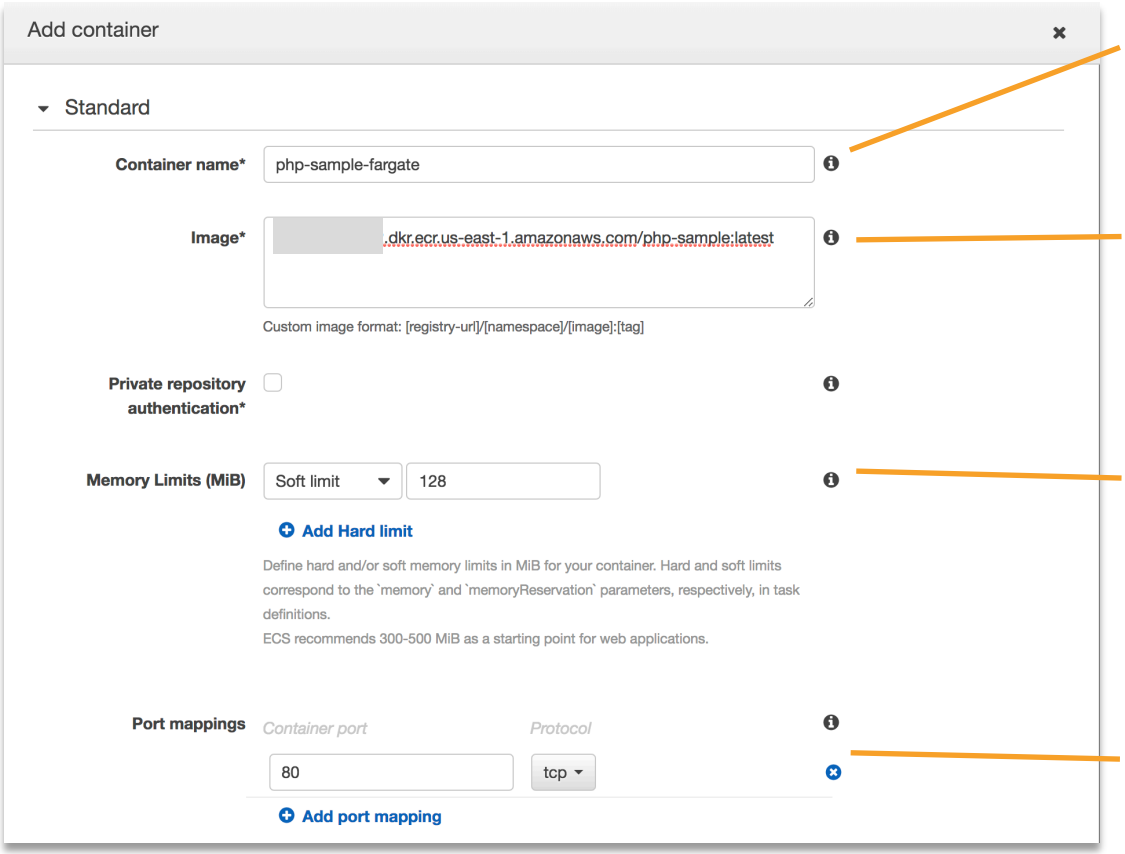


タスク名を指定
(php-sample-fargate)

Task Role は None

タスクに割り当てるメモリ／CPUを
0.5GB/0.25vCPUに指定

タスク定義作成(3)



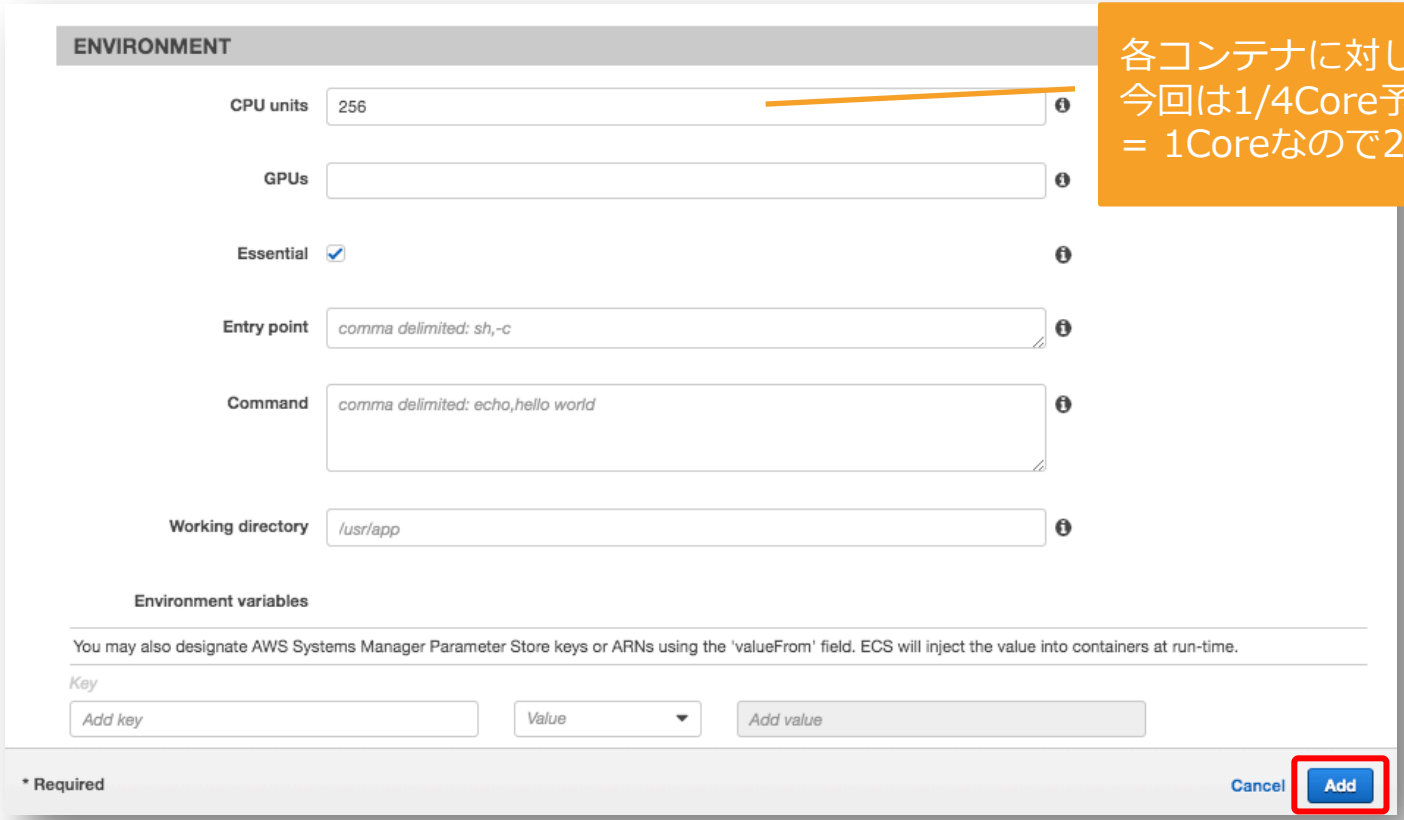
コンテナ名を指定(後続の手順の関係上今回はphp-sample-fargateに固定)

利用するDockerイメージを<ECRのリポジトリURI>:<タグ>の形式で指定(今回のタグはlatest)
リポジトリは以前の手順と同じものを指定する

コンテナに割り当てられるMemoryのリミットを定義
今回はSoftLimitで128MBを指定

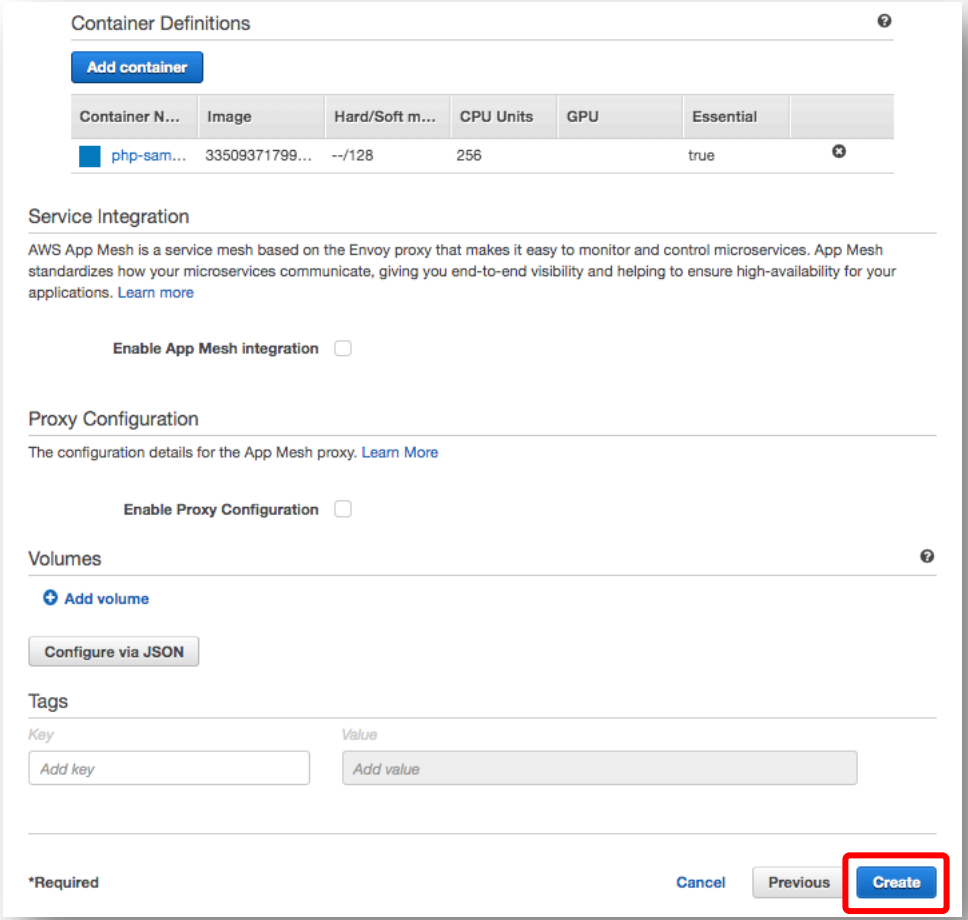
コンテナportを80に設定

タスク定義作成(3)

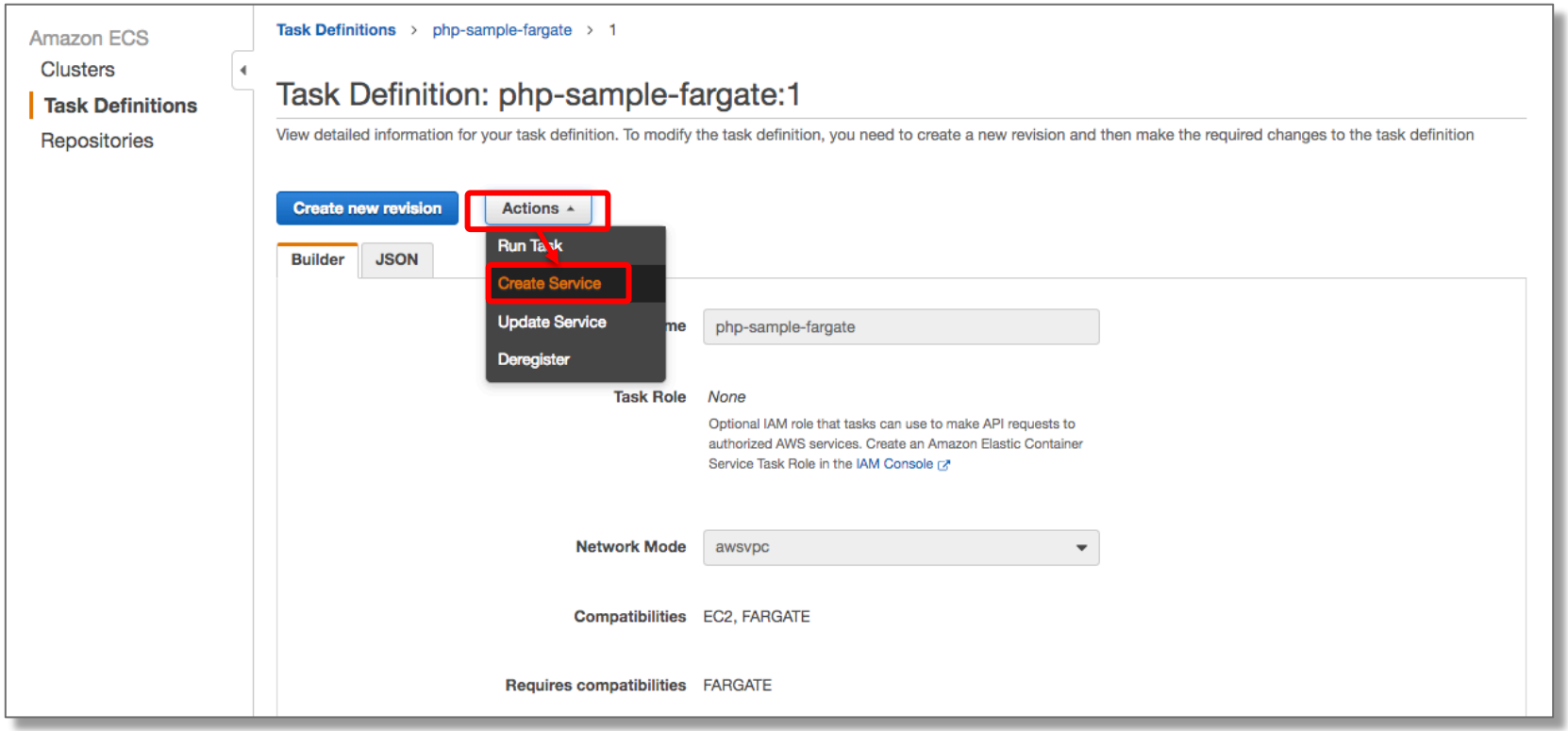


各コンテナに対してのCPU予約を定義
今回は1/4Core予約(1024 CPU units = 1Coreなので256を指定)

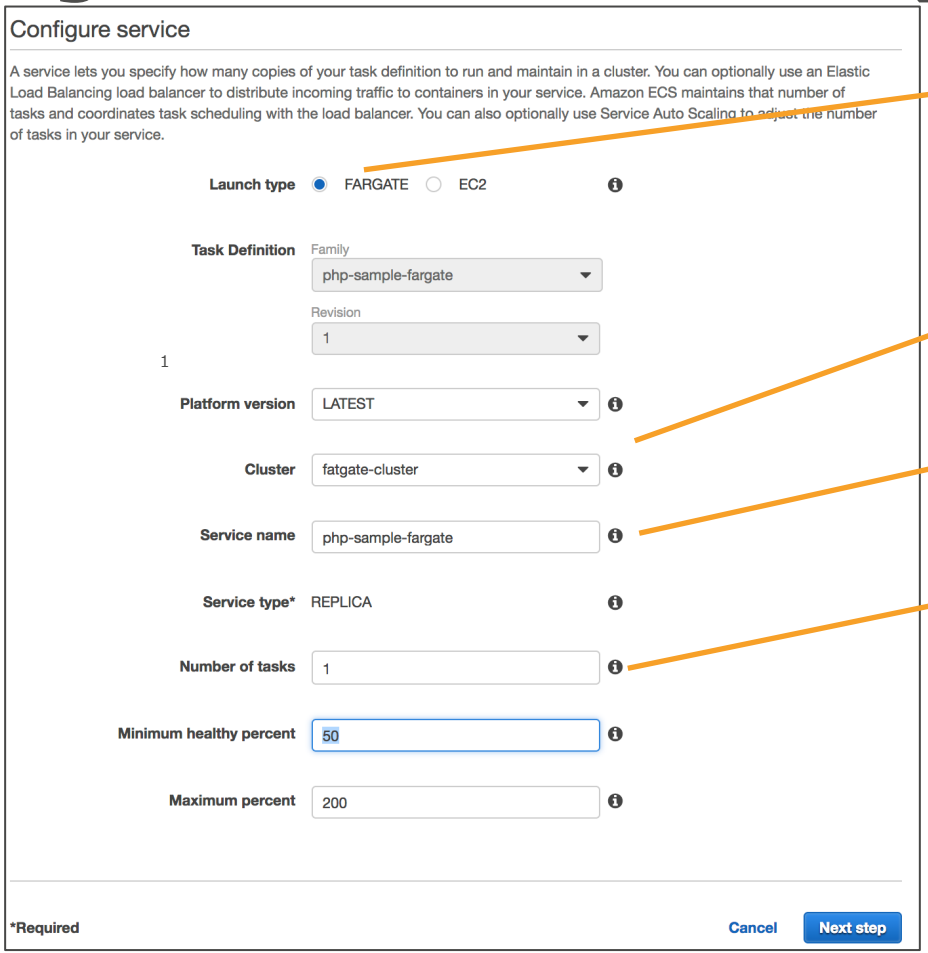
タスク定義作成(4)


Container Definitions
Add container
Container N...
Image
Hard/Soft m...
CPU Units
GPU
Essential
php-sam...
33509371799...
--/128
256
true
Service Integration
AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications. Learn more
Enable App Mesh integration
Proxy Configuration
The configuration details for the App Mesh proxy. Learn More
Enable Proxy Configuration
Volumes
Add volume
Configure via JSON
Tags
Key
Value
Add key
Add value
*Required
Cancel
Previous
Create

Fargate上でのサービス作成(1)



Fargate上でのサービス作成(2)



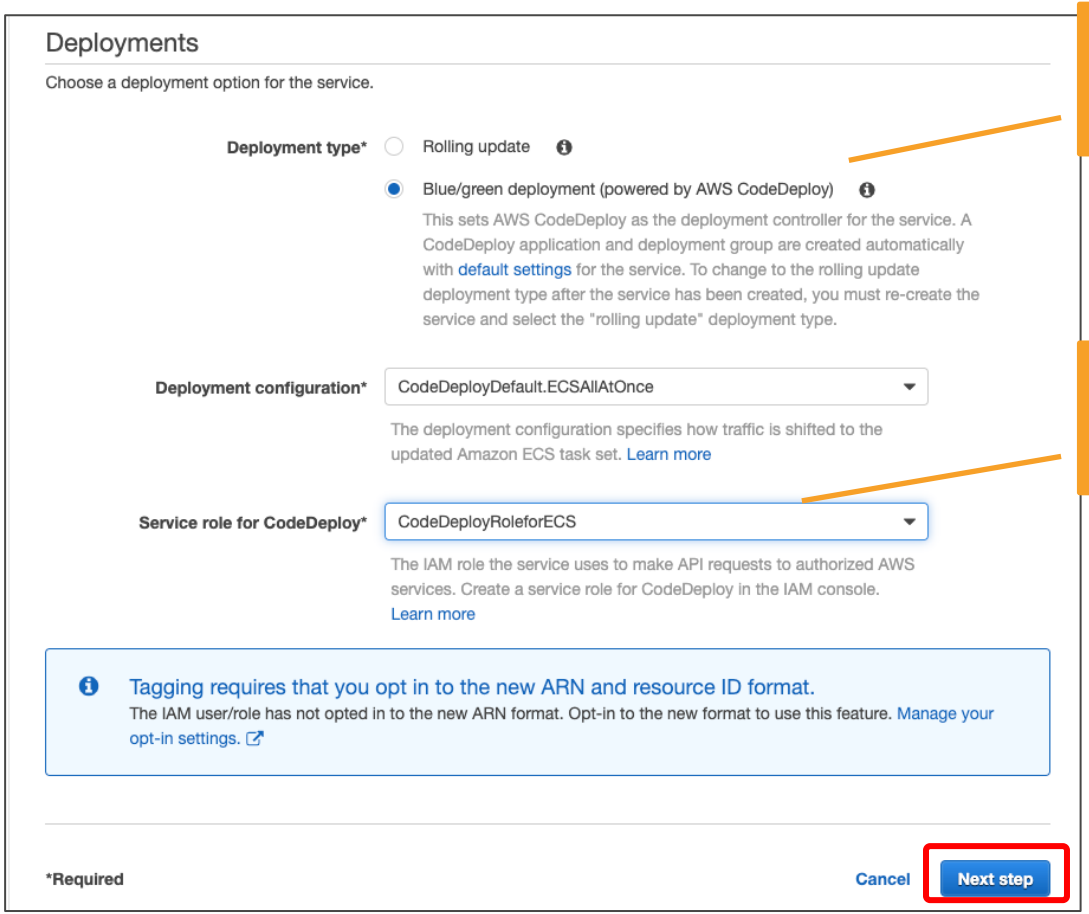
Launch TypeはFARGATEを選択

先ほど作成したタスク定義とクラスターを指定

Service name は php-sample-fargate

初期起動タスク数を指定
今回は1タスクを起動する

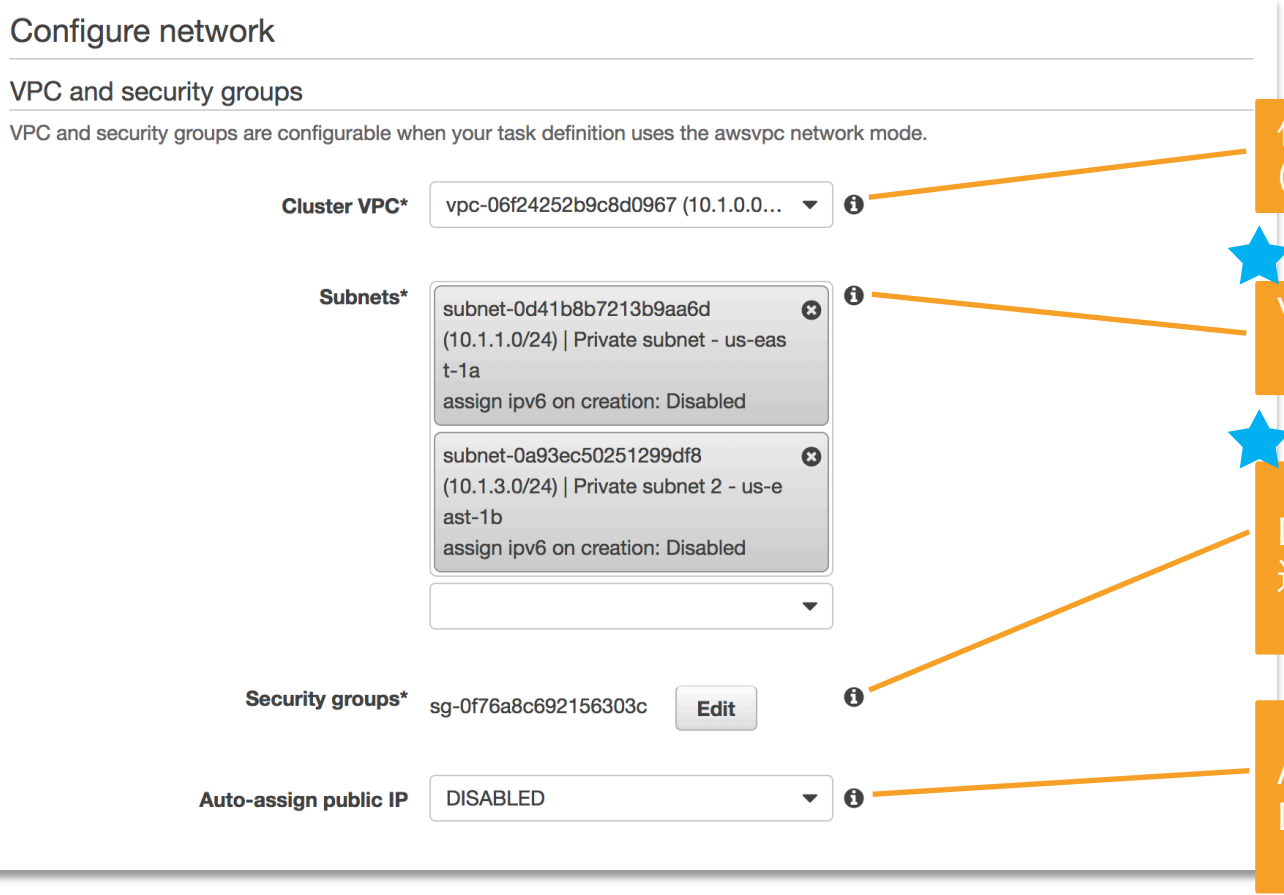
Fargate上でのサービス作成(3)



Deployment Typeは
Blue/green deploymentを選択

Service Roleは先ほど作成した
CodeDeployRoleforECSを選択

Fargate上でのサービス作成(3)



作成したVPCを選択
(10.1.0.0/16)

よくある間違いポイント

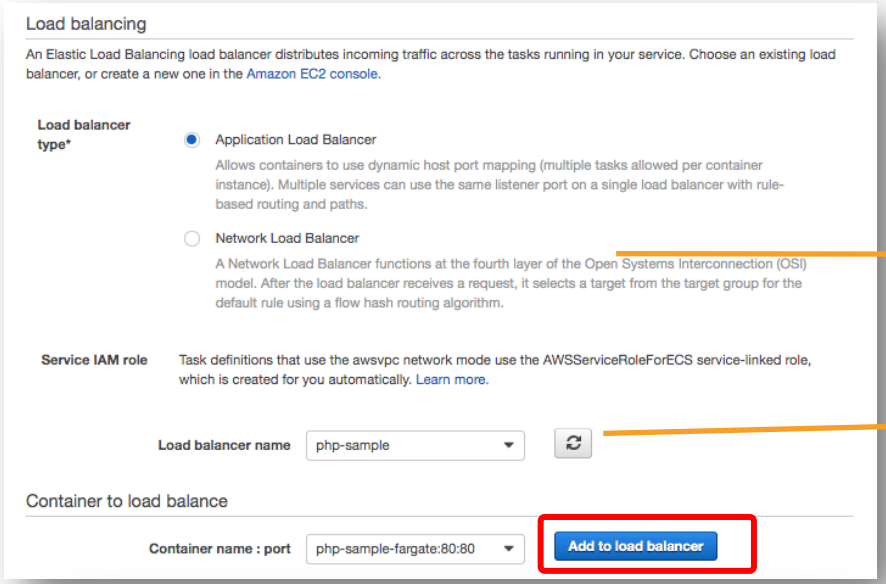
VPC内のPrivateSubnetを
２つ選択

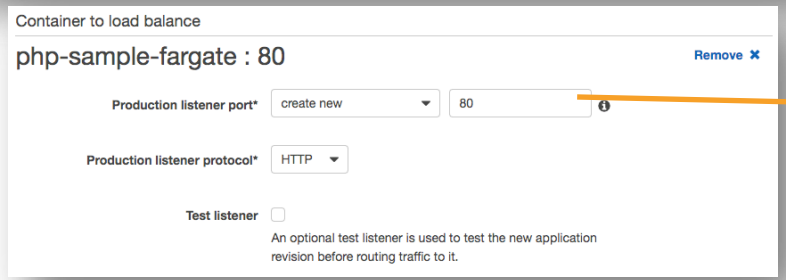
よくある間違いポイント

DefaultのSecurityGroupを
選択(Editして変更)

Auto-assign public IPを
DISABLEDに変更

Fargate上でのサービス作成(4)



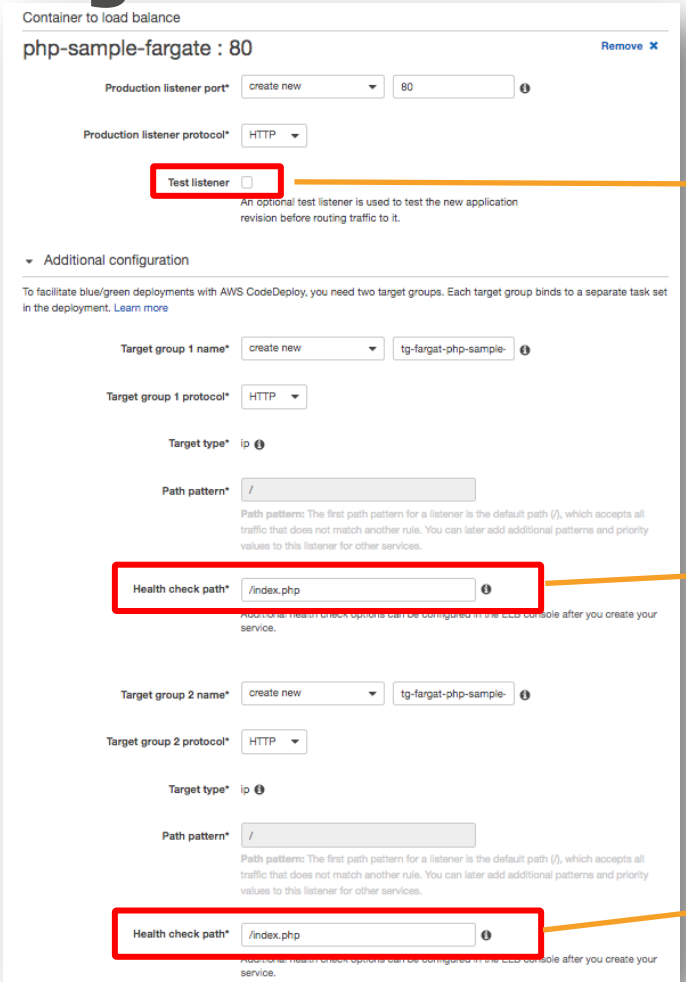


Application Load Balancerを選択

作成したALB名を選択

ポート番号に80を入力

Fargate上でのサービス作成(5)



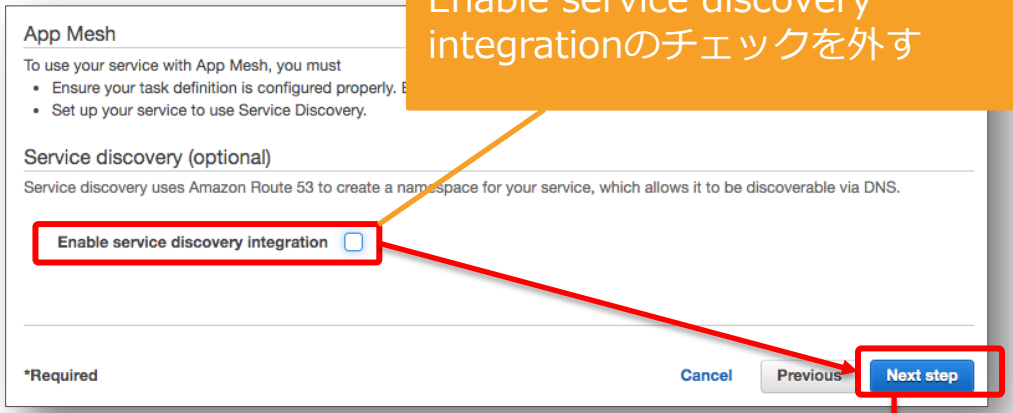
Test Listnerのチェックを外す

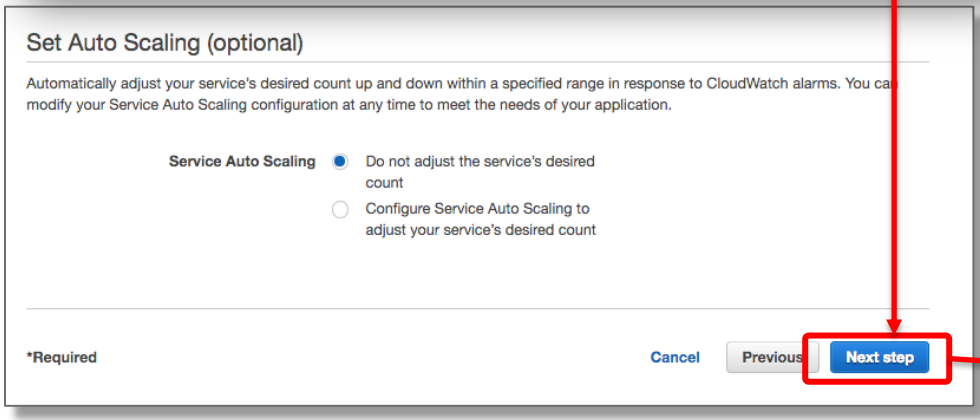
今回は/index.phpというページを公開するため、このページをヘルスチェックパスに設定

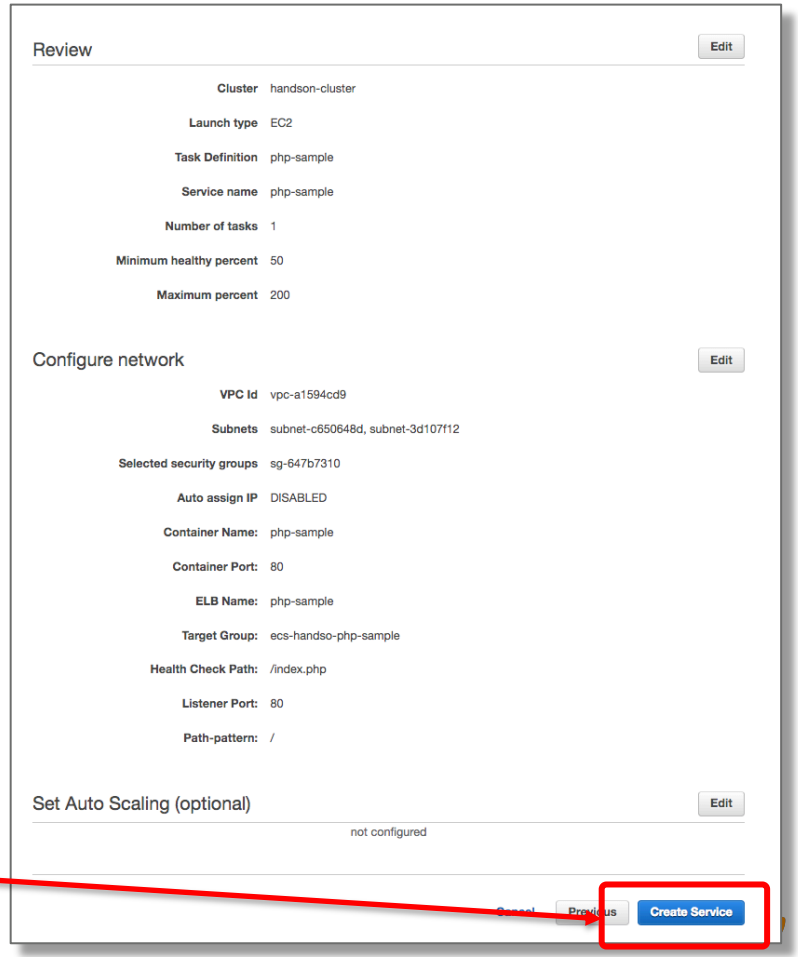
上記同様、/index.phpをヘルスチェックパスに設定

Fargate上でのサービス作成(6)

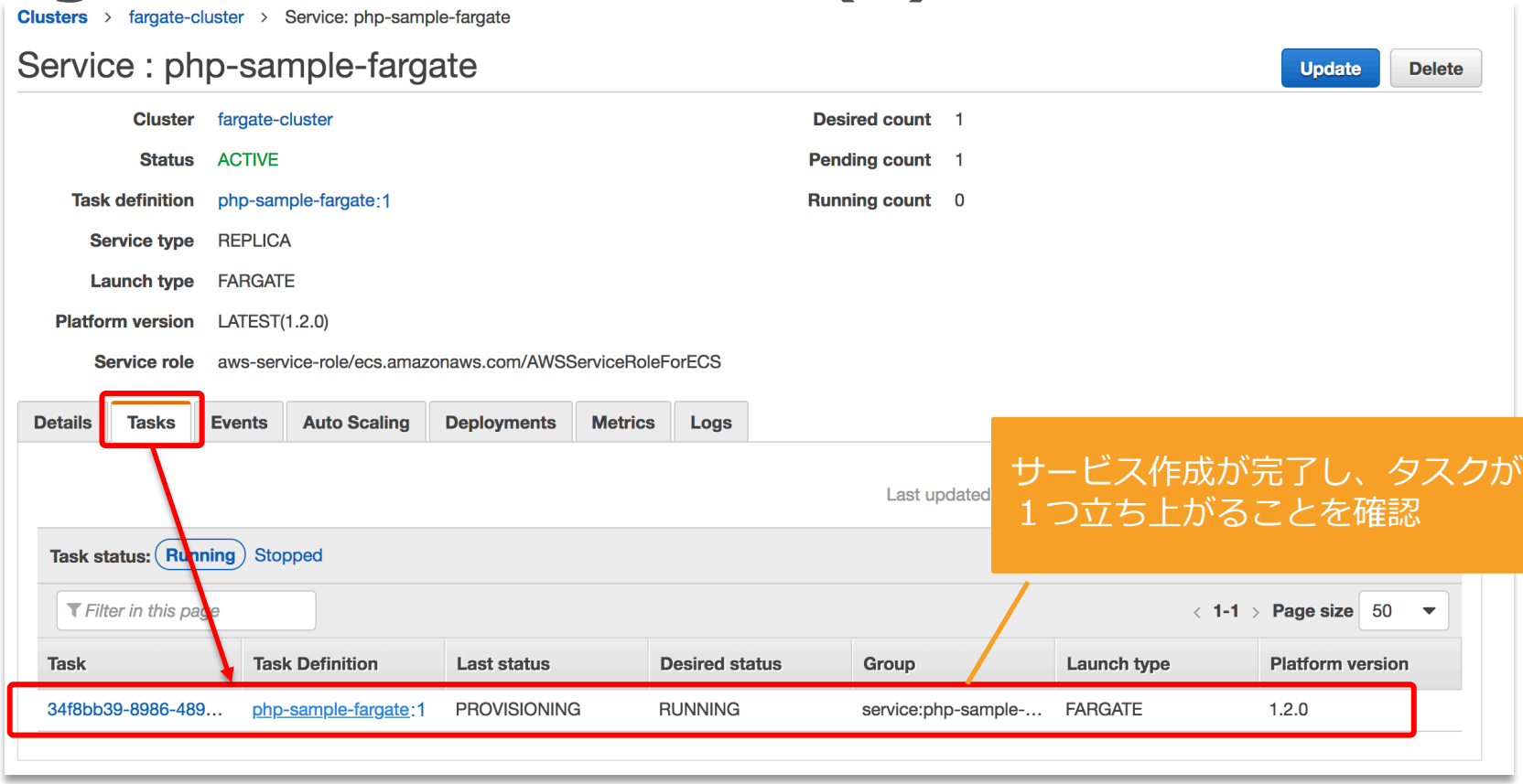
Enable service discovery integrationのチェックを外す







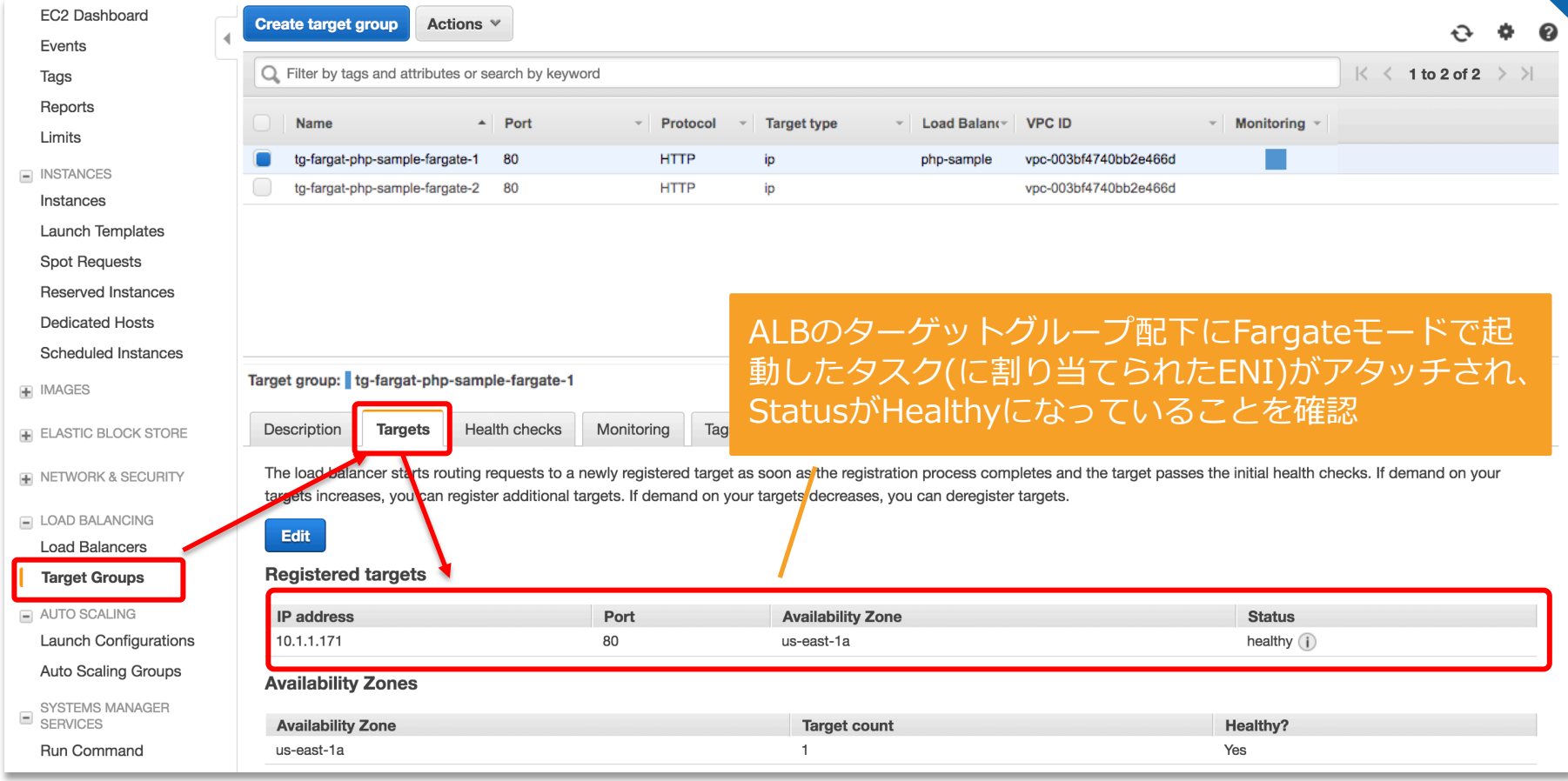
Fargate上でのサービス作成(7)



Fargate上でのサービス作成(8)

Services -> EC2 へ

EC2 コンソール



ALBのターゲットグループ配下にFargateモードで起動したタスク(に割り当てられたENI)がアタッチされ、StatusがHealthyになっていることを確認

サービス作成(9)


Amazon EC2 Container Servic
EC2 Management Console
VPC Management Console
CloudWatch Management Con
PHP Test
Not Secure
php-sample-1903026157.us-east-1.elb.amazonaws.com
ip-10-1-1-171.ec2.internal
保存していたALBのエンドポイントにアクセスし、PHP
アプリケーションのページが表示されることを確認

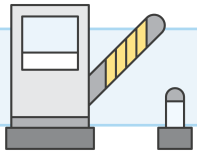
AWS CodeServices

CI/CDパイプライン

Source → Build → Test → Production

継続的インテグレーション

継続的デリバリー



継続的デプロイメント

aws

CI/CDパイプラインを実現するAWSサービス

ソースコードの
バージョン管理

AWS CodeCommit

ビルド自動化

AWS CodeBuild

デプロイ自動化

AWS CodeDeploy

AWS CloudFormation

AWS Elastic Beanstalk

ワークフロー管理

AWS CodePipeline

AWS CodeStar

AWS Code Services

AWS CodeCommit

- セキュア、スケーラブルなGit互換のリポジトリサービス
- スタンダードなGit Toolからアクセス可能
- Pushなどのイベントをトリガーに SNS/Lambdaを呼び出し可能

AWS CodeBuild

- スケーラビリティに優れたビルドサービス
- ソースのコンパイル、テスト、パッケージ生成をサポート
- Dockerイメージの作成も可能

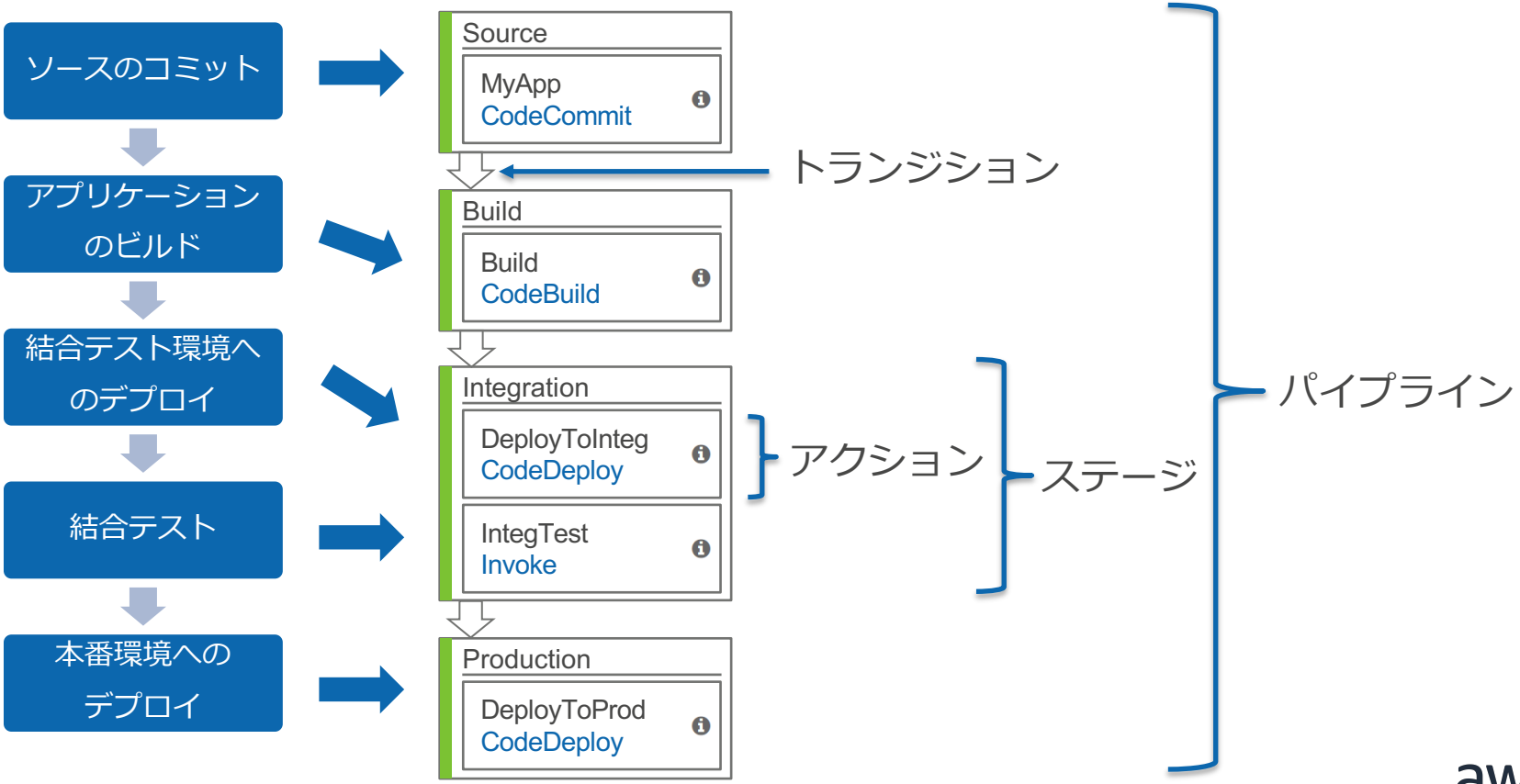
AWS CodeDeploy

- S3またはGitHub上のコードをあらゆるインスタンスにデプロイ
- デプロイを安全に実行するための様々な機能を提供
- In-place(ローリング) およびBlue/Greenのデプロイをサポート

AWS CodePipeline

- リリースプロセスのモデル化と見える化を実現
- カスタムアクションによる柔軟なパイプライン作成が可能
- 様々なAWSサービスや3rdパーティ製品との統合をサポート

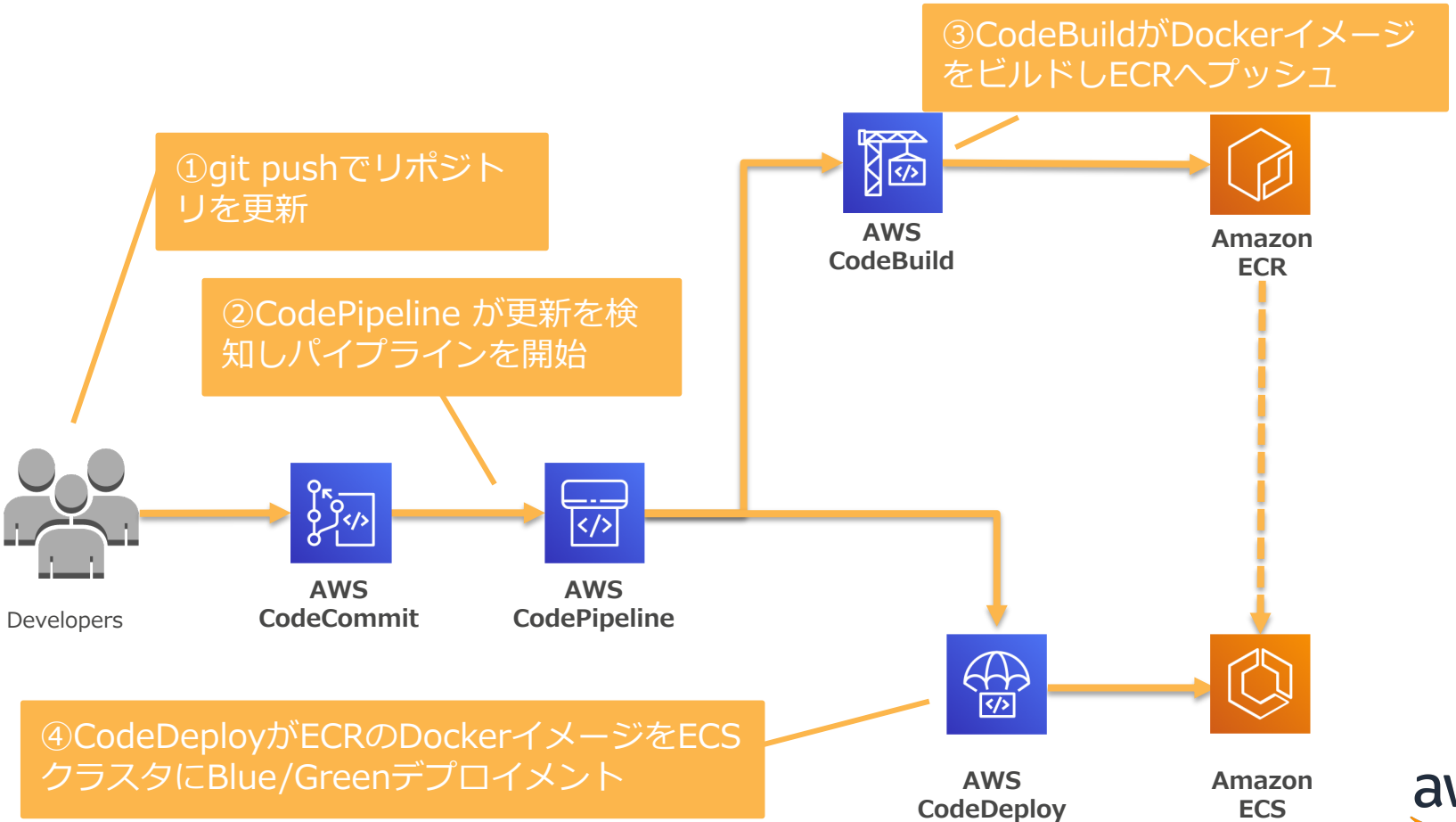
CodePipelineでのリリースプロセスモデル化



ハンズオン③

AWS Code Servicesを利用したCI/CDパイプラインの構築

本ハンズオンで作成する継続的デプロイメント構成

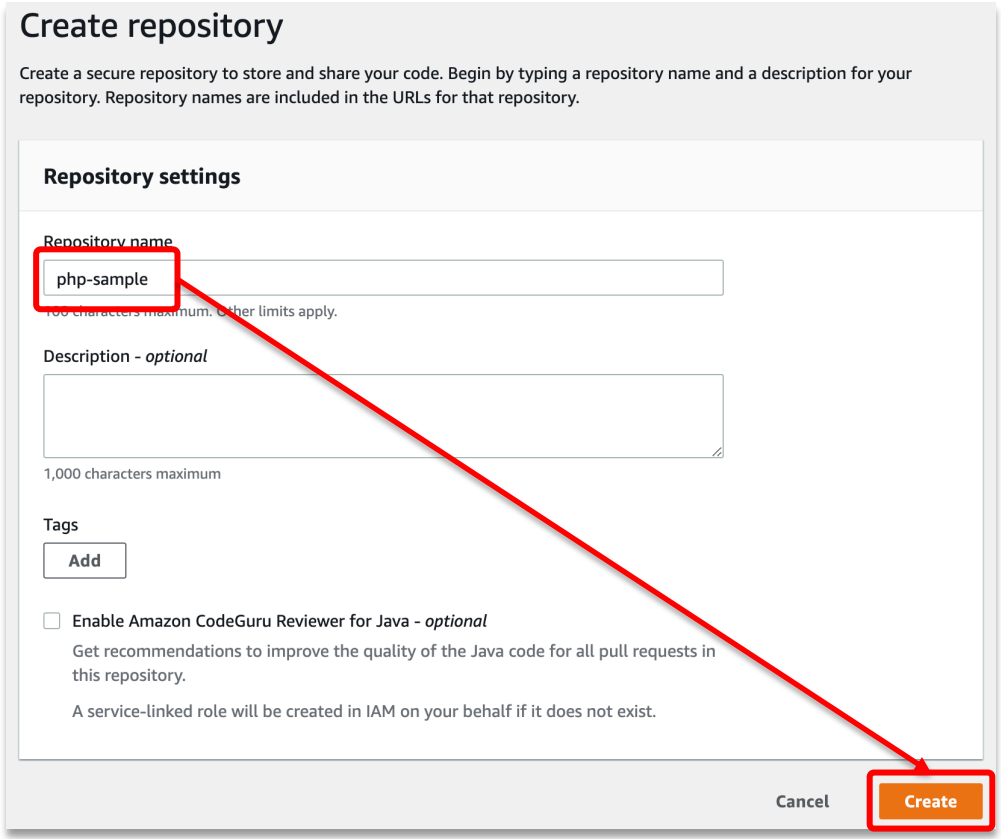


CodeCommit設定

CodeCommitリポジトリ

Services -> CodeCommit -> Create repository へ

CodeCommit コンソール



CodeCommitリポジトリのクローン(1)

Cloud9

CodeCommitでの操作後、Cloud9 terminal へ

Clone URLをコピーし、Cloud9のターミナルで git clone $URL を実行

aws

コンテンツ準備

リポジトリ配下に以下のファイルを作成

- Dockerfile と src/index.php は以前のものを php-sample/ へコピー
- buildspec.yml を新規作成
  - CodeBuildのビルドプロジェクト定義ファイル
  - 内容は次ページ
- appspec.yml を新規作成
- taskdef.json を新規作成

```
php-sample/
├── Dockerfile
├── buildspec.yml
├── appspec.yml
├── taskdef.json
└── src
    └── index.php
```

Cloud9 上のローカルリポジトリディレクトリ構成

buildspec.yml

```
version: 0.2
phases:
  pre_build:
    commands:
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=01234567890.dkr.ecr.us-east-1.amazonaws.com/php-sample
      - IMAGE_TAG=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
  build:
    commands:
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - printf '{"Version":"1.0","ImageURI":"%s"}' $REPOSITORY_URI:$IMAGE_TAG > imageDetail.json
artifacts:
    files: imageDetail.json
```

pre_buildフェーズではECRレポジトリへのログインをおこない、ビルドIDのPrefixをDockerイメージのタグに指定

buildフェーズでは環境変数で指定するレポジトリURIを取得し、Dockerイメージのビルドを実施

post_buildフェーズではビルドしたDockerイメージをECRレポジトリにPush

CodePipelineで後続のフェーズにタグ情報を渡すため、アウトプットアーティファクトを指定

buildspec.yml の拡大版は次ページを参照

buildspec.yml の修正

```
version: 0.2
phases:
  pre_build:
    commands:
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=${Your account ID}.dkr.ecr.us-east-1.amazonaws.com/php-sample
      - IMAGE_TAG=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)

  build:
    commands:
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG

  post_build:
    commands:
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - printf '{"Version":"1.0","ImageURI":"%s"}' $REPOSITORY_URI:$IMAGE_TAG > imageDetail.json

artifacts:
    files: imageDetail.json
```

アカウントIDを自身のものに修正してください

※お急ぎの方はZipファイル内のサンプルコードをご利用ください。

appspec.yml

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "<TASK_DEFINITION>"
        LoadBalancerInfo:
            ContainerName: "php-sample-fargate"
            ContainerPort: "80"
```

AppSpecファイルはCodeDeployがデプロイを管理するために使用するファイルです

TASK_DEFINITIONプレースホルダーはPipelineが走ると自動的に置き換わります

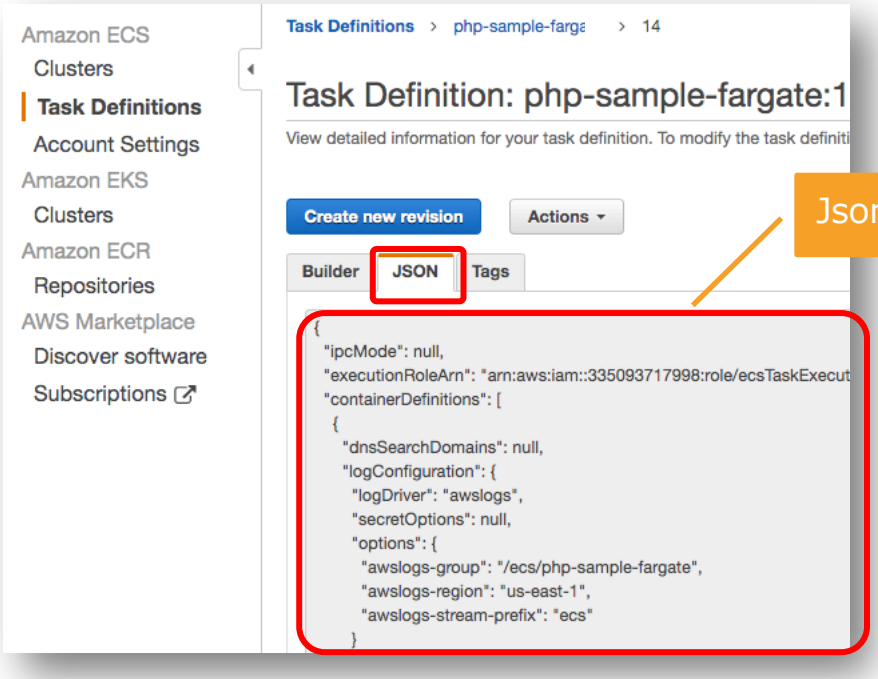
今回の対象はECSなのでECSタスク定義、ロードバランサー情報などを含める必要があります

修正せずそのままお使いください

※お急ぎの方はZipファイル内のサンプルコードをご利用ください。

taskdef.json

ECS -> Task Definition -> php-sample-fargate -> 最新のRevisionを表示



Jsonタブを表示し、内容を全てコピー

taskdef.json の修正

```
…
 "memoryReservation": 128,
      "volumesFrom": [],
      "stopTimeout": null,
      "image": "<IMAGE1_NAME>",
      "startTimeout": null,
      "dependsOn": null,
      "disableNetworking": null,
      "interactive": null,
      "healthCheck": null,
…
```

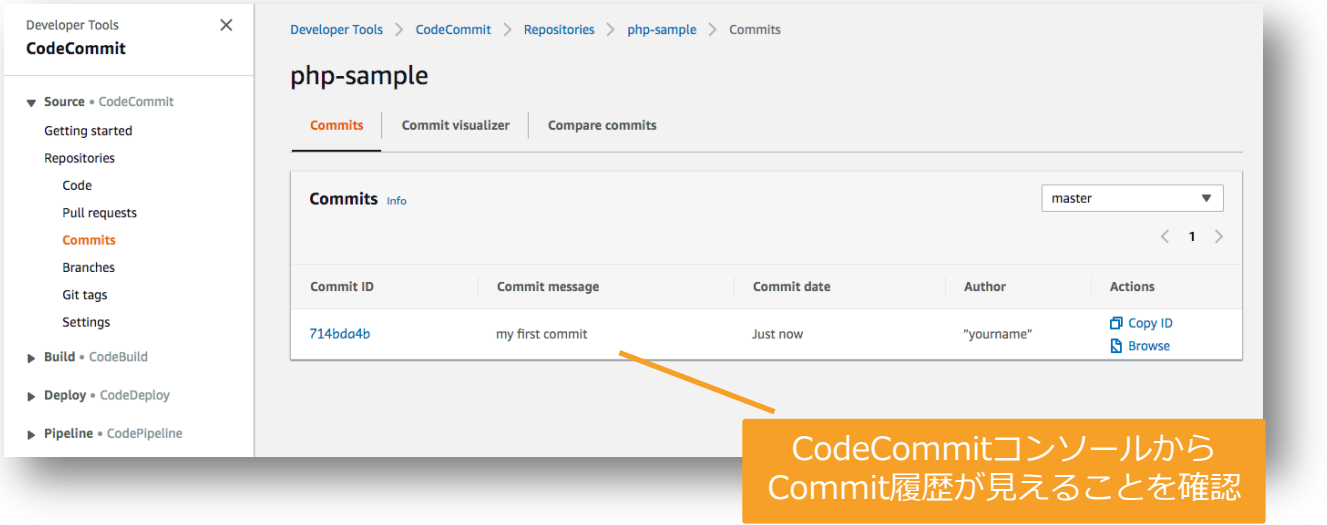
taskdef.jsonはFargateで起動されるタスクの情報が記載されたファイルです。今回はPipelineがトリガーされると動的にタスク定義を更新します

先ほどコピーした内容を全て貼り付ける

imageを<IMAGE1_NAME>に変更。このプレースホルダーはBuildアクション後に動的にアップデートされます

リポジトリへのpush

```
#リモートリポジトリへのPush
$ cd php-sample
$ git add -A
$ git commit -m ”my first commit”
$ git push origin master
```



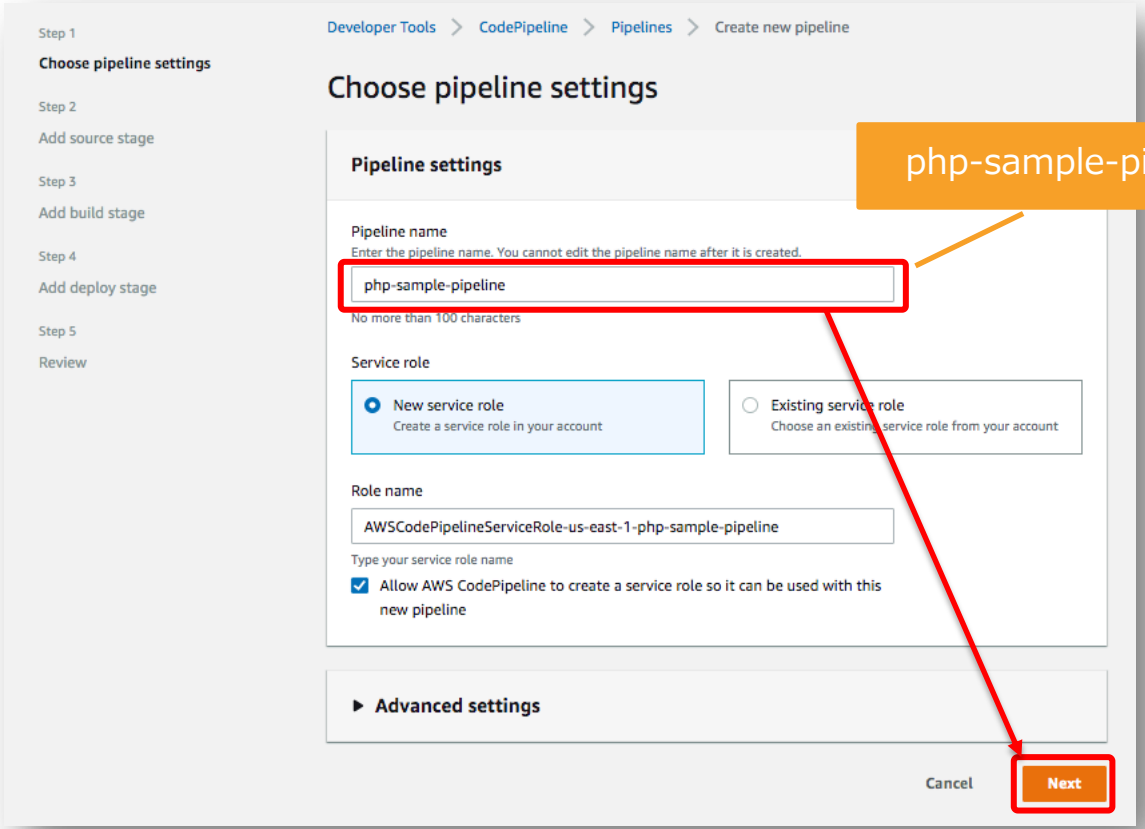
https://docs.aws.amazon.com/ja_jp/codecommit/latest/userguide/troubleshooting-ch.html#troubleshooting-macoshttps

CodePipeline設定

パイプライン作成(1)

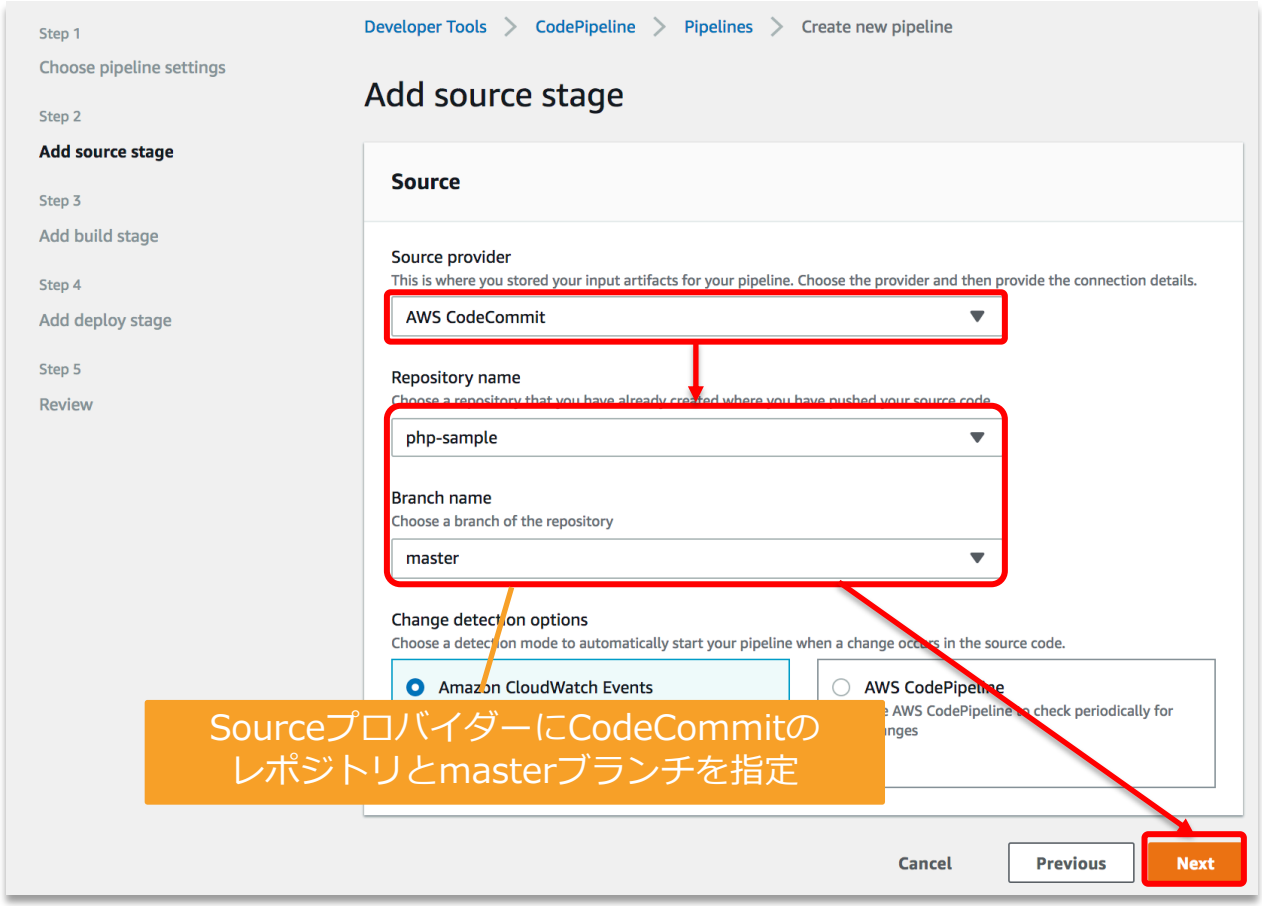
Services -> CodePipeline -> Create pipeline へ

CodePipeline コンソール

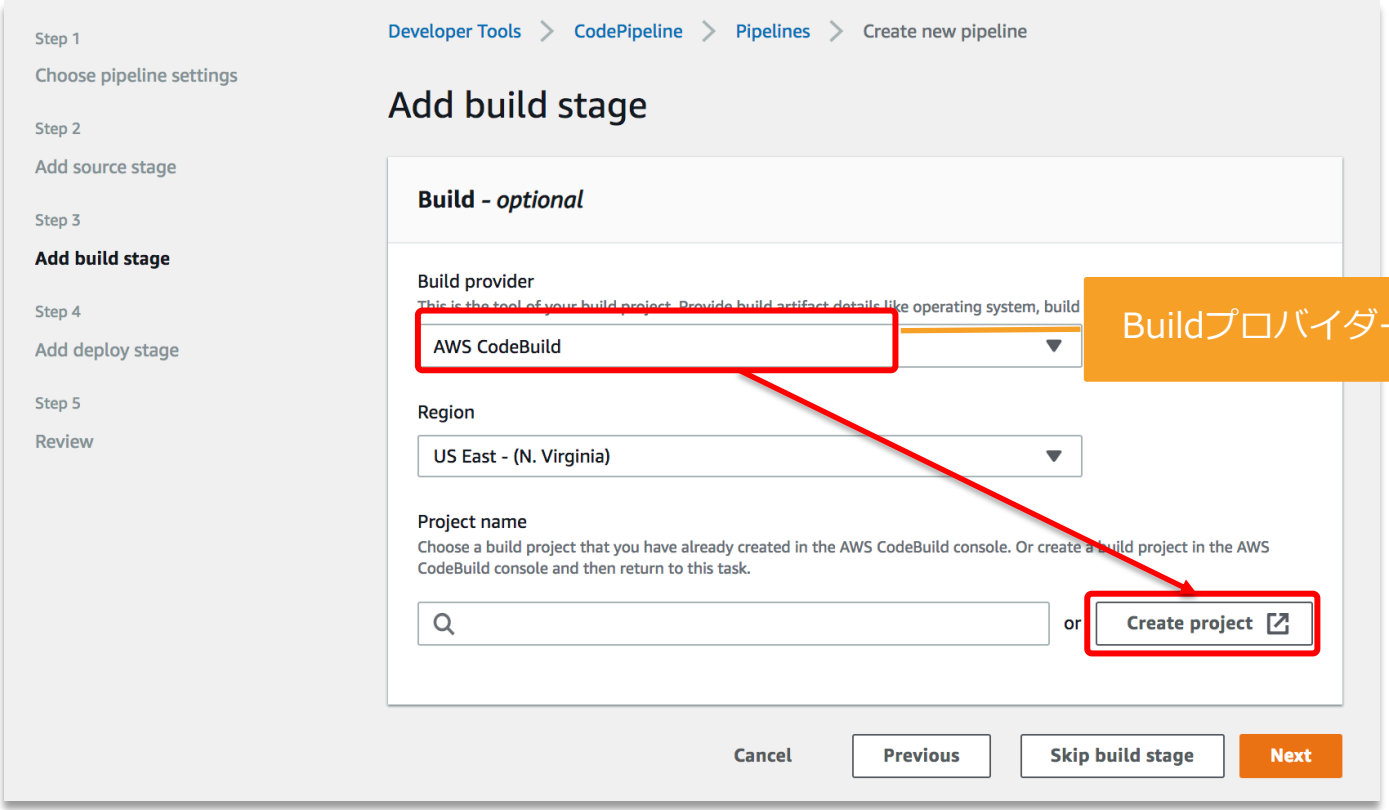


php-sample-pipeline と入力

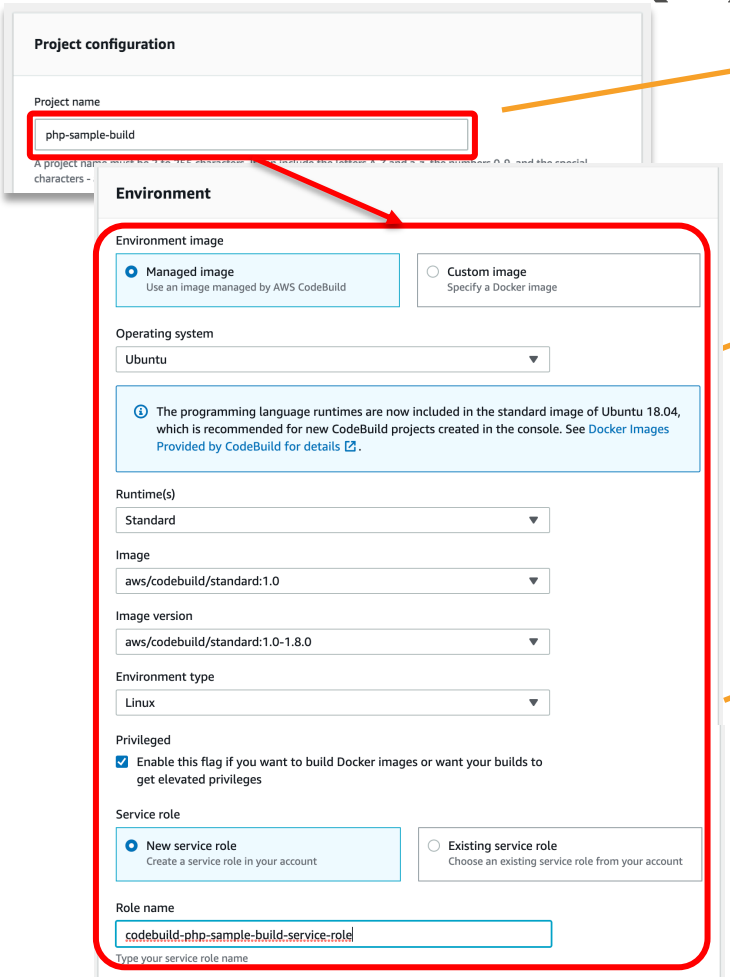
パイプライン作成(2)


Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Add source stage
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
AWS CodeCommit
Repository name
php-sample
Branch name
Choose a branch of the repository
master
Change detection options
Choose a detection mode to automatically start your pipeline when a change occurs in the source code.
Amazon CloudWatch Events
AWS CodePipeline
Cancel
Previous
Next
SourceプロバイダーにCodeCommitの
レポジトリとmasterブランチを指定

パイプライン作成(3)


Step 1
Choose pipeline settings
Step 2
Add source stage
Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Developer Tools > CodePipeline > Pipelines > Create new pipeline
Add build stage
Build - optional
Build provider
AWS CodeBuild
Region
US East - (N. Virginia)
Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
or
Create project
Cancel
Previous
Skip build stage
Next
BuildプロバイダーにCodeBuildを指定

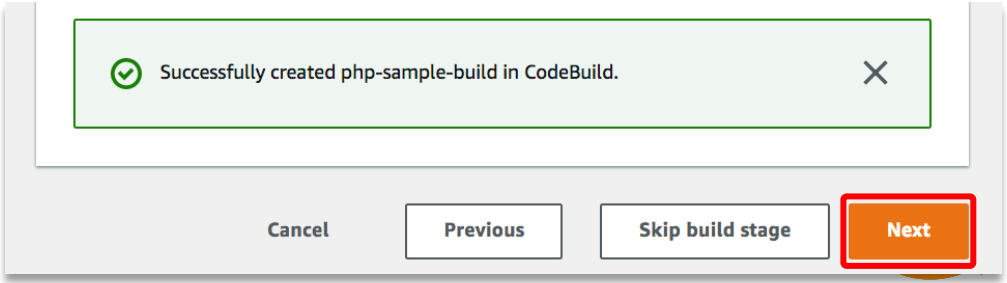
パイプライン作成(4)



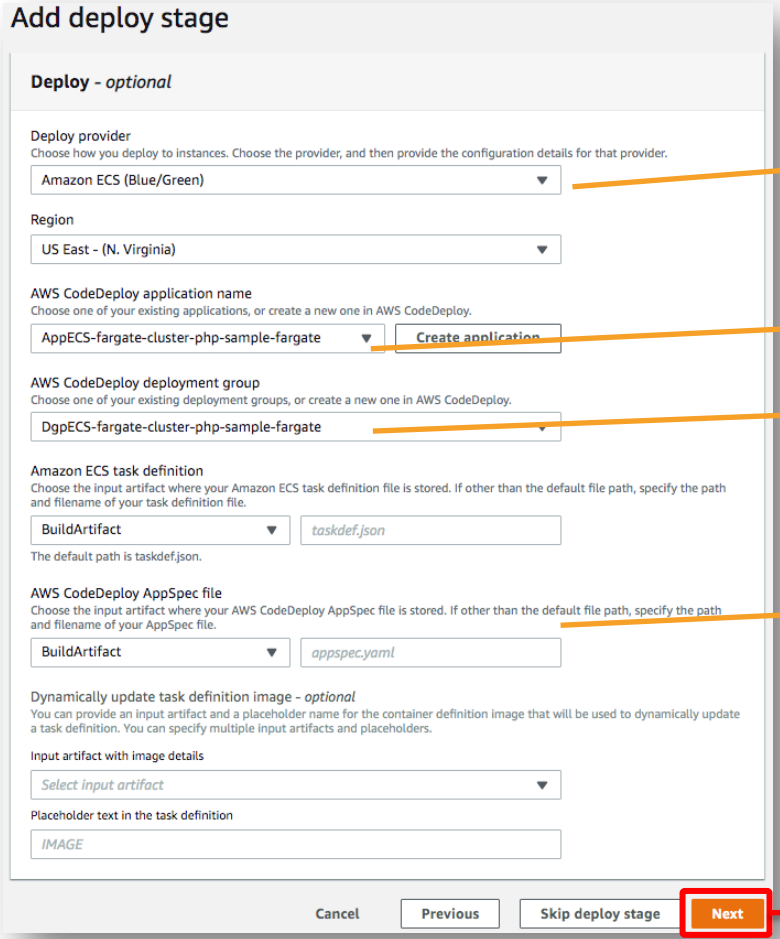
Buildプロジェクト名を指定
php-sample-build

Build環境を指定
- Environment Image：Managed Image
- Operating System：Ubuntu
- Runtime: Standard
- Image: aws/codebuild/standard:1.0
- Image version: aws/codebuild/standard: 1.0-1.8.0
- Privileged の項目にチェックをつける
- Role name に codebuild-php-sample-build-service-roleを入力

その他はデフォルトのままにしContinue to Codepipelineをクリック、CodePipeline へ戻り、Nextを選択



パイプライン作成(5)



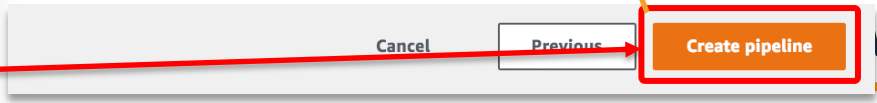
DeployプロバイダーにAmazon ECS (Blue/Green)を指定

Application name に AppECS-xxx を指定

Deployment groupに DgpECS-xxx を指定

Task DefinitionとAppSpec Fileのinput artifactにBuildArtifactを選択、これは後ほど修正します

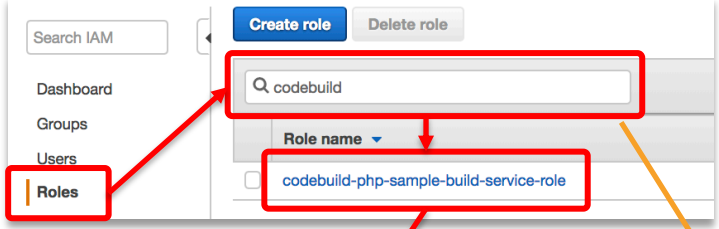
Reviewページ確認後、
Create pipelineを選択

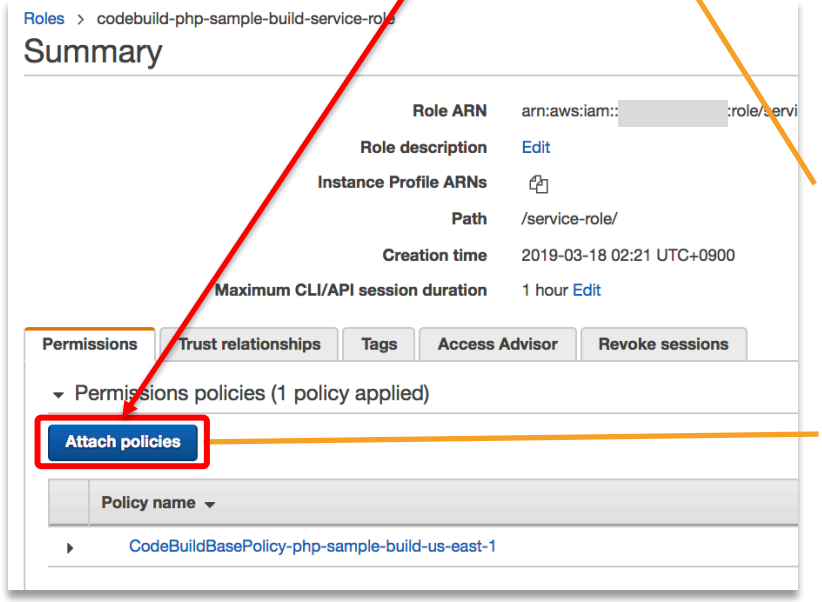


CodeBuildのIAM Role編集 (1)

Services -> IAM へ

IAM コンソール



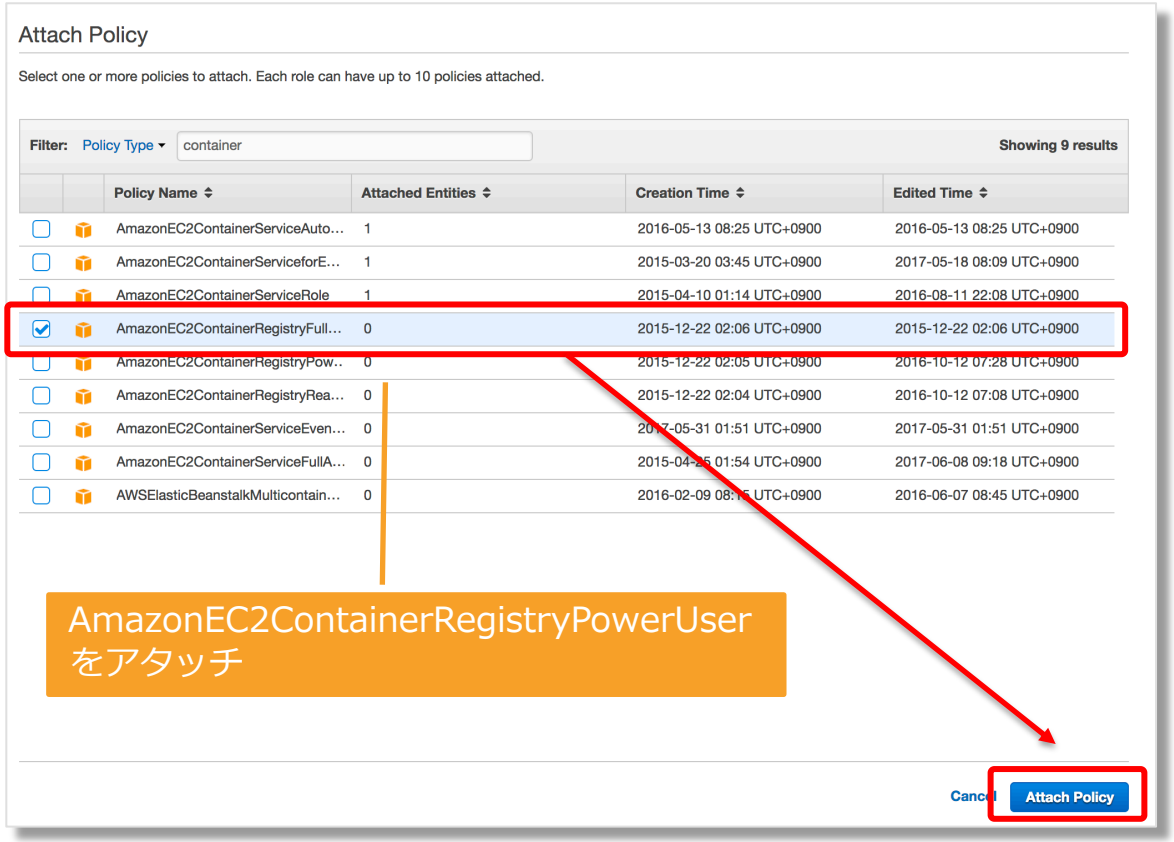


今回定義したビルド定義(buildspec.yml)では、ビルド内でECRに対してDockerイメージをPushしているため、CodeBuildに割り当てたIAMロール(codebuild-php-sample-build-service-role)に対してECRの操作権限を付与する必要があります

IAM Roleより、codebuildを検索し、先ほど自動作成されたロール codebuild-php-sample-build-service-role を選択

Attach policiesを選択

CodeBuildのIAM Role編集 (2)

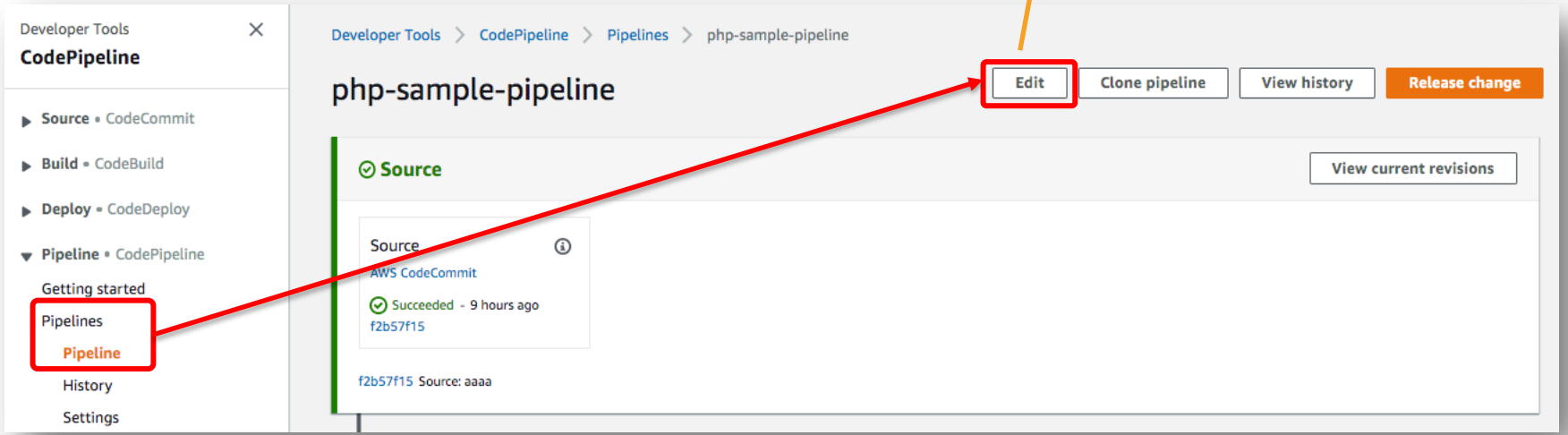


CodePipeline の設定編集（1）

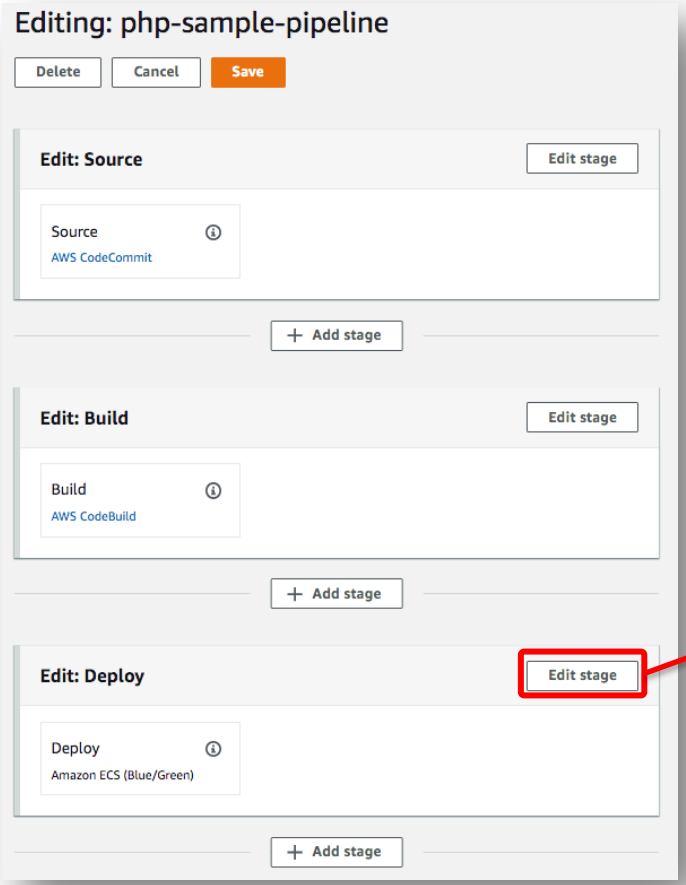
Services -> CodePipeline へ

Code Pipeline コンソール

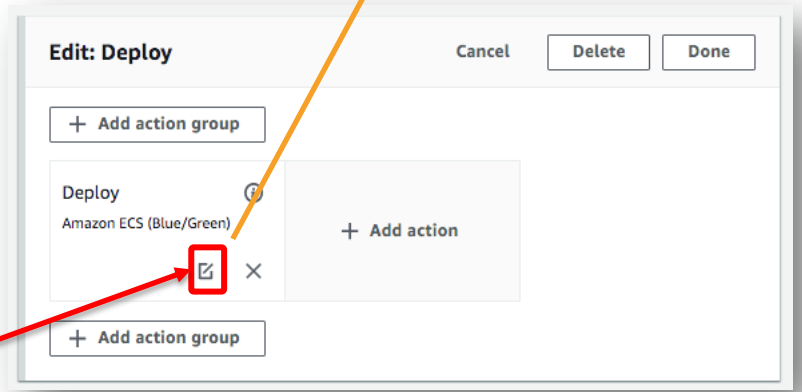
先ほど作成したphp-sample-pipelineを選択し、Edit



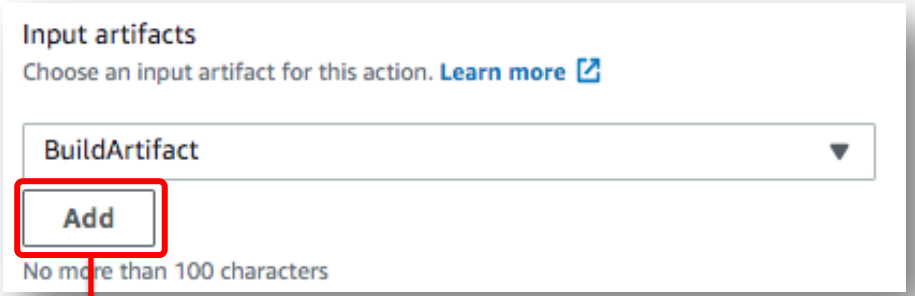
CodePipeline の設定編集 (2)

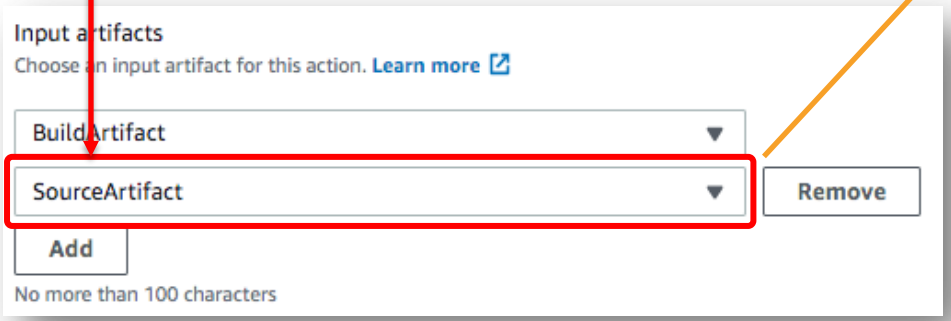


Deployステージを編集



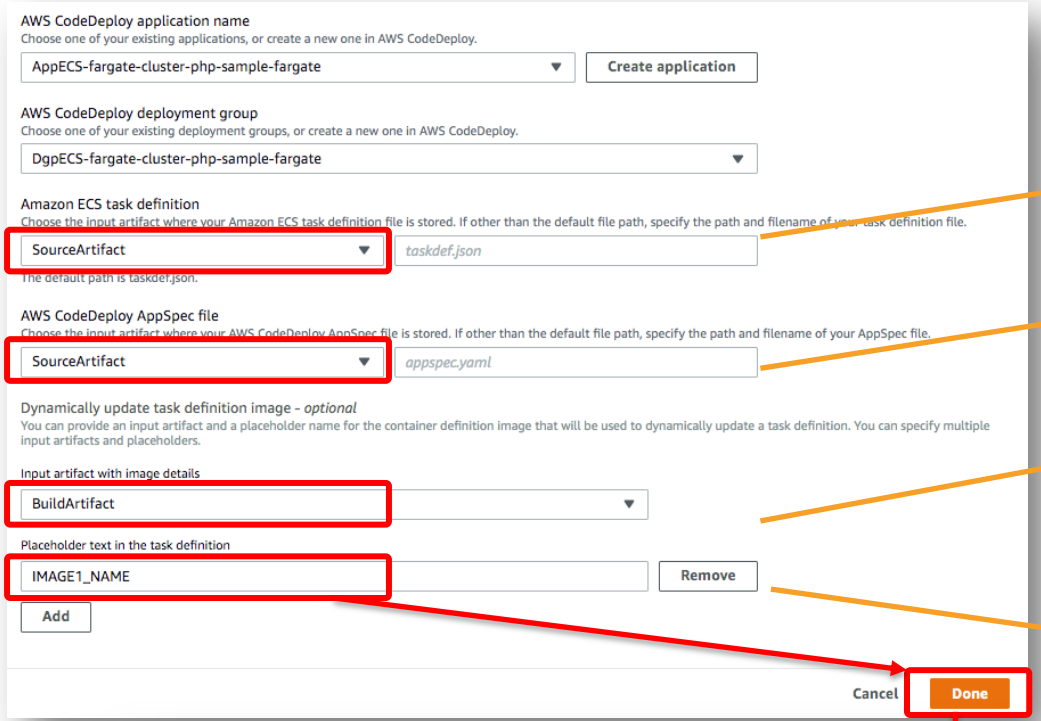
CodePipeline の設定編集 (3)





Input artifactsで Addを押し、
SourceArtifactを追加する

CodePipeline の設定編集 (4)



ECS task definitionのinput artifactを **SourceArtifact**とする

CodeDeploy AppSpec file のinput artifactを **SourceArtifact**とする

Task definitionのimageを動的に変更するための情報が記載されたinput artifactを **BuildArtifact**とする

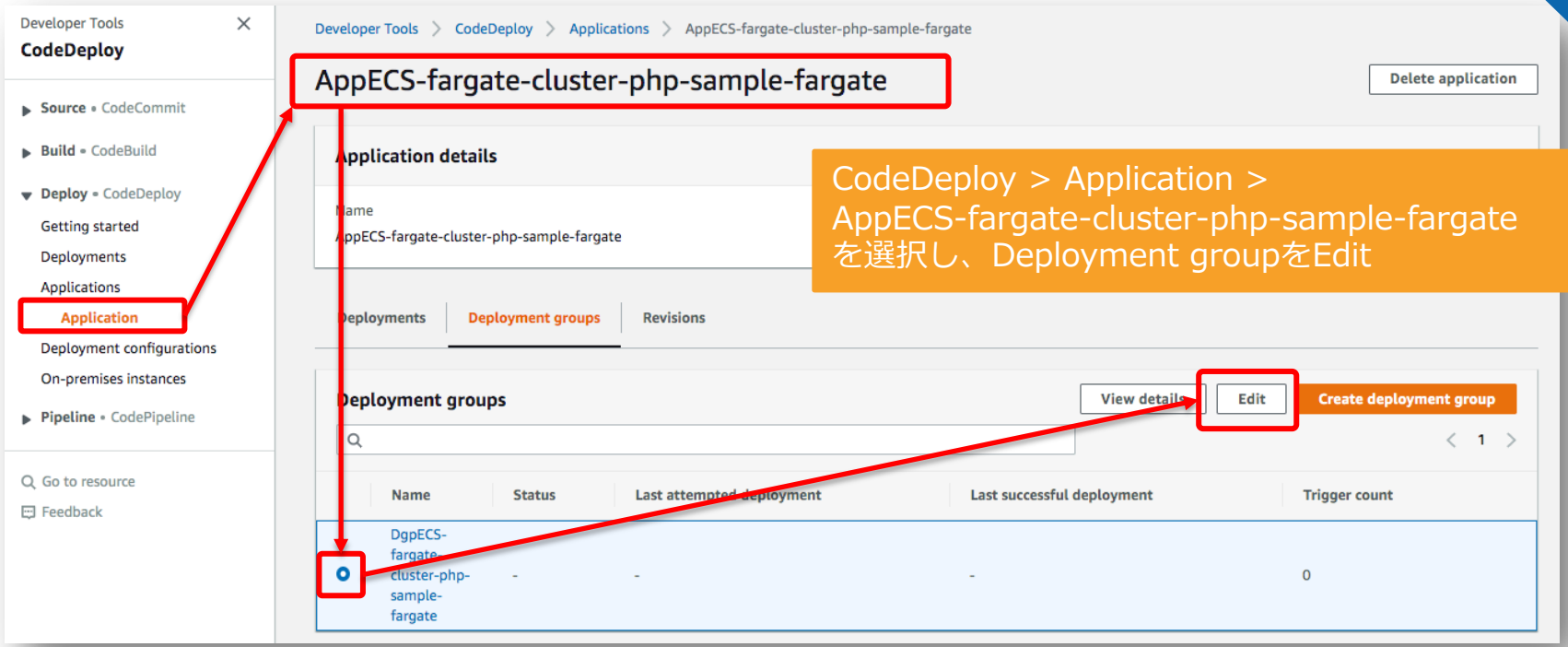
Image placeholderを IMAGE1_NAME とする

Pipeline編集のページでSaveする

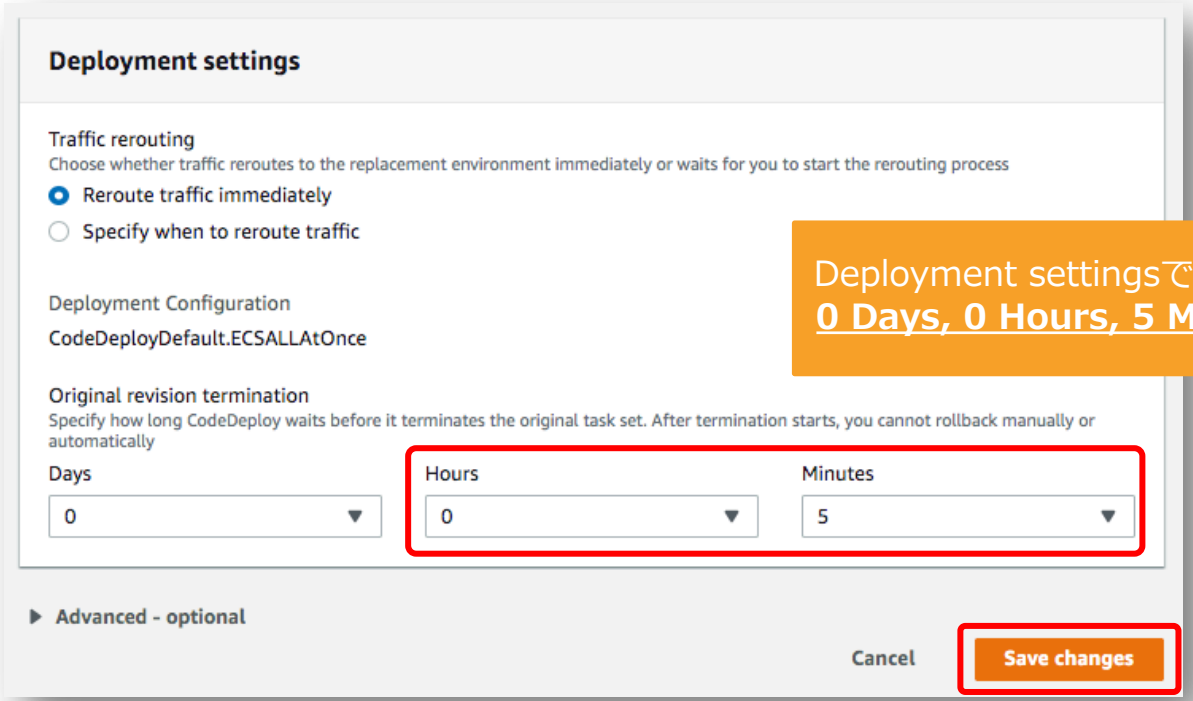
CodeDeploy の設定編集（1）

CodeDeploy 選択

Code Deploy コンソール



CodeDeploy の設定編集 (2)

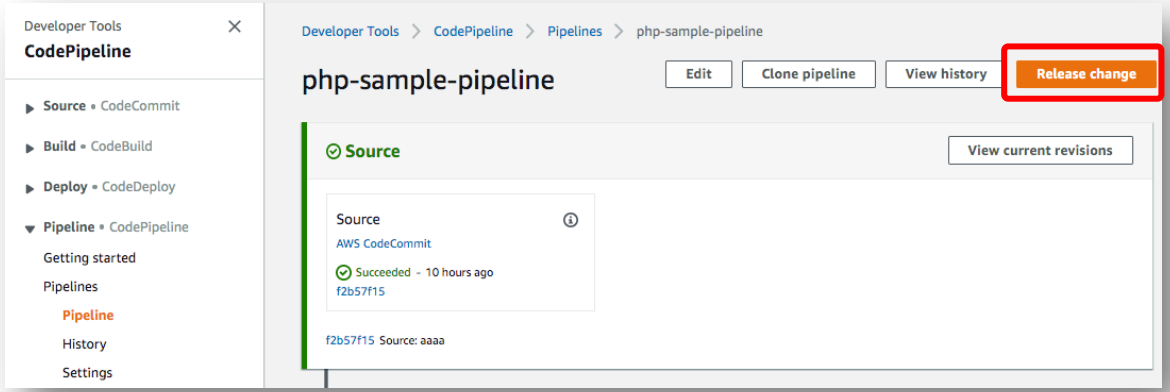


Deployment settingsで Original revision termination を
0 Days, 0 Hours, 5 Minutesに変更

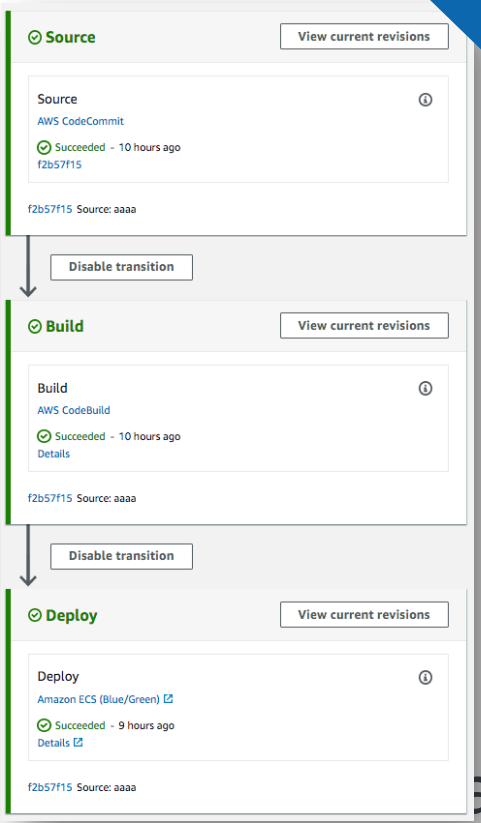
パイプライン実行

CodePipeline 選択

CodePipelineからphp-sample-pipelineを選択し、Release changeをクリック



パイプラインのリリースが始まります。パイプラインが正常に動作しリリースが完了することを確認して下さい。
パイプラインの完了には10分程度かかります。

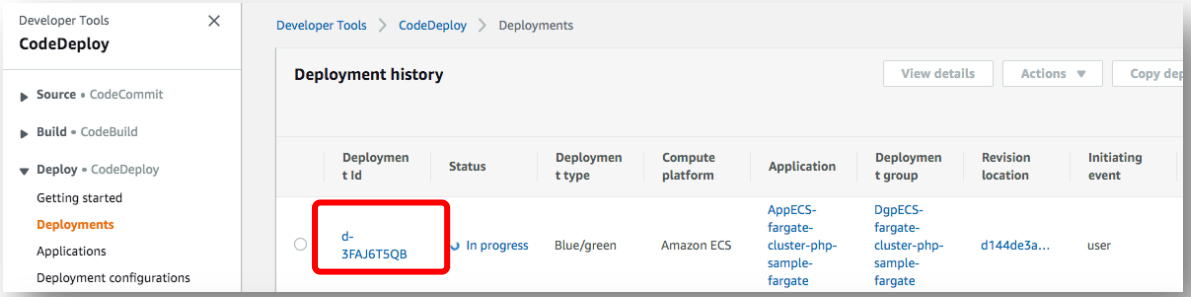


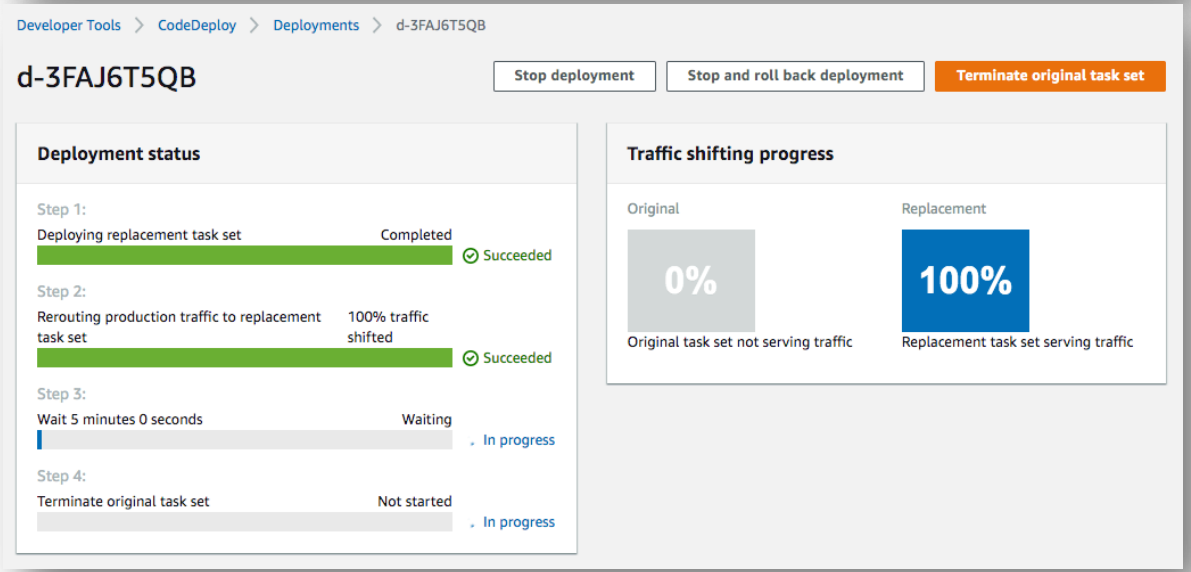
Code Pipeline コンソール

デプロイ状況の確認（1）

CodeDeploy -> Deployments へ

Code Deploy コンソール





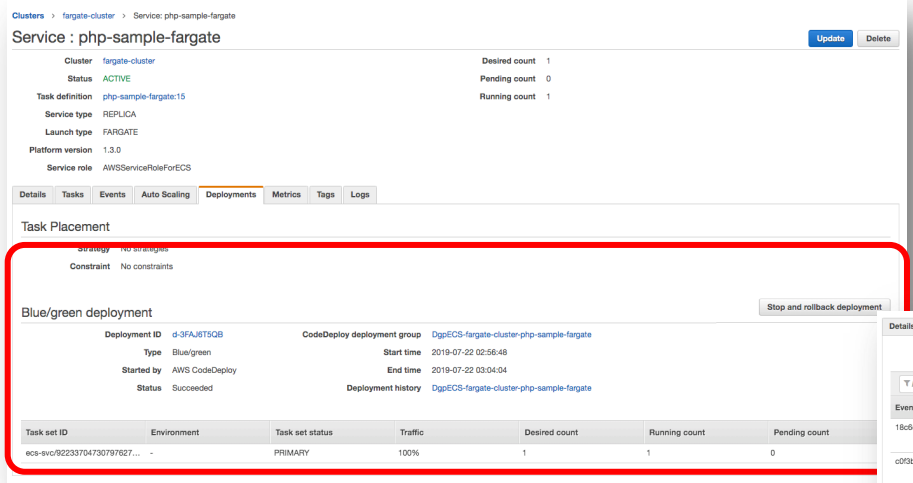
Deploymentの進行状況をCodeDeployコンソールより確認できます。

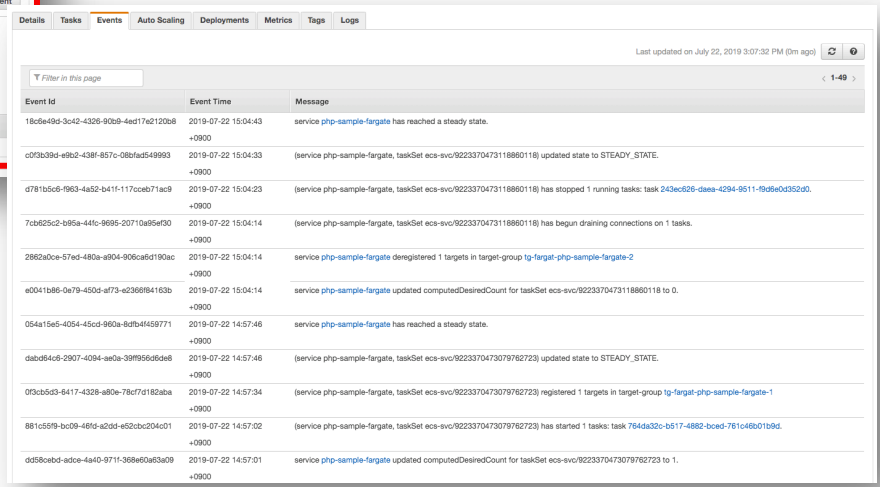
Original Taskが削除されるまで5分間待ちます。

デプロイ状況の確認 (2)

Services -> ECS -> Cluster -> Service へ

ECS コンソール





ServiceのDeploymentsタブよりBlue/Green Deploymentの状況を確認することもできます

Eventsタブでは、タスクの起動や、Drainingの履歴を確認することができます

継続的デプロイメント
パイプライン動作確認

リポジトリへのpush

Services -> Cloud9 へ

Cloud9

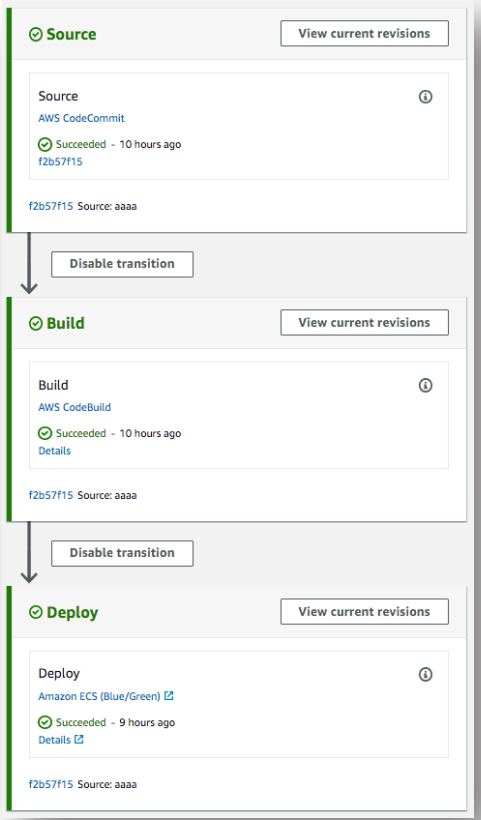
```
#アプリのバージョンアップ(index.phpを適当に書き変える)
$ vim src/index.php

#リモートリポジトリへのPush
$ git add -A
$ git commit -m “pipeline test”
$ git push origin master
```

aws

継続的デプロイメント動作確認

以下のポイントを確認する

- CodePipelineのパイプラインが自動的に開始され、正常に完了すること
- CodeBuildで新しいBuild runが開始されていること
- CodeDeployで新しいDeploymentが開始されていること
- ECSで新しいタスク定義でサービス更新が行われていること
- サイトにアクセスし、新しいアプリケーションがデプロイされていること

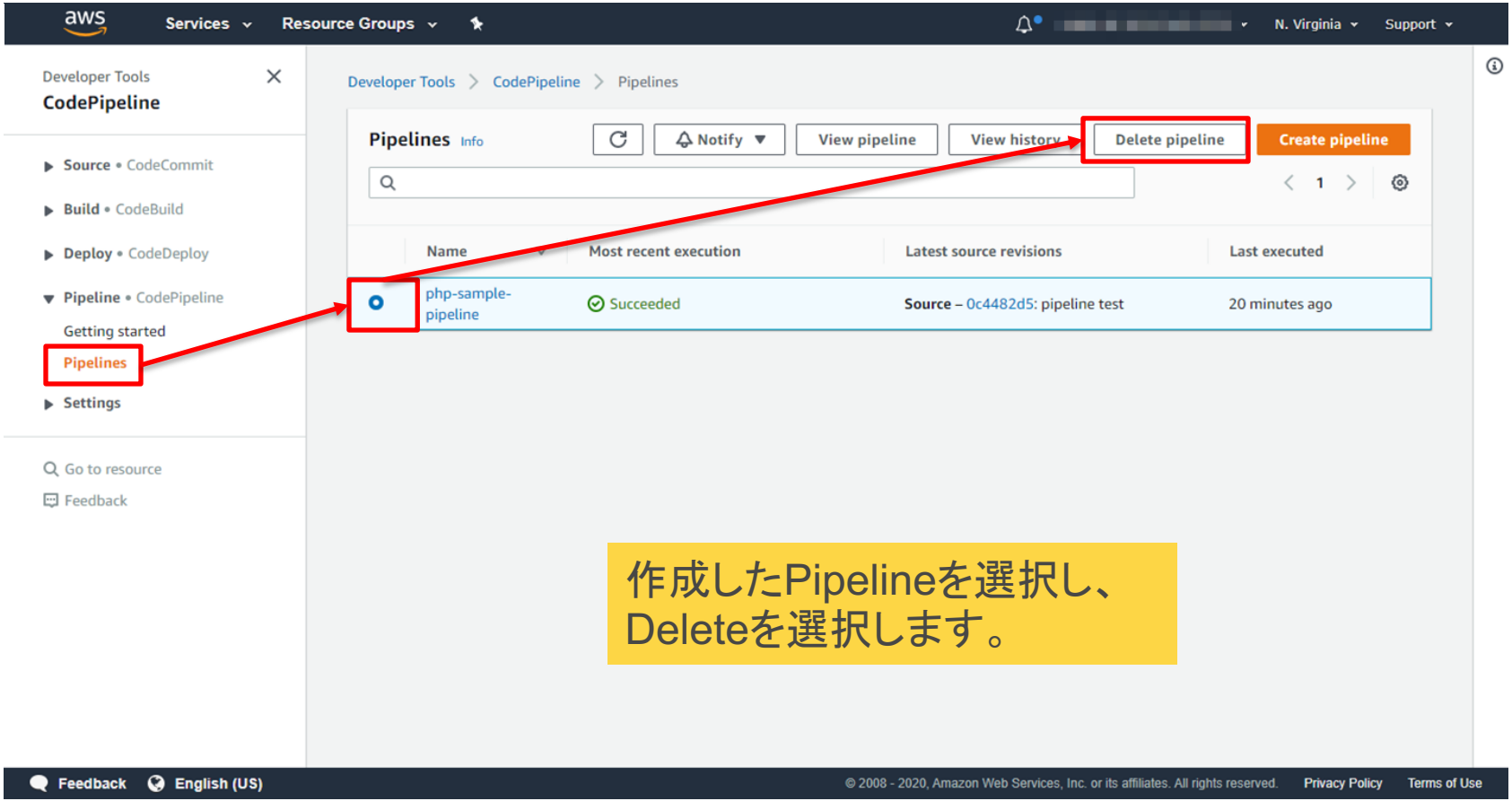


後片付け

作成したリソース削除

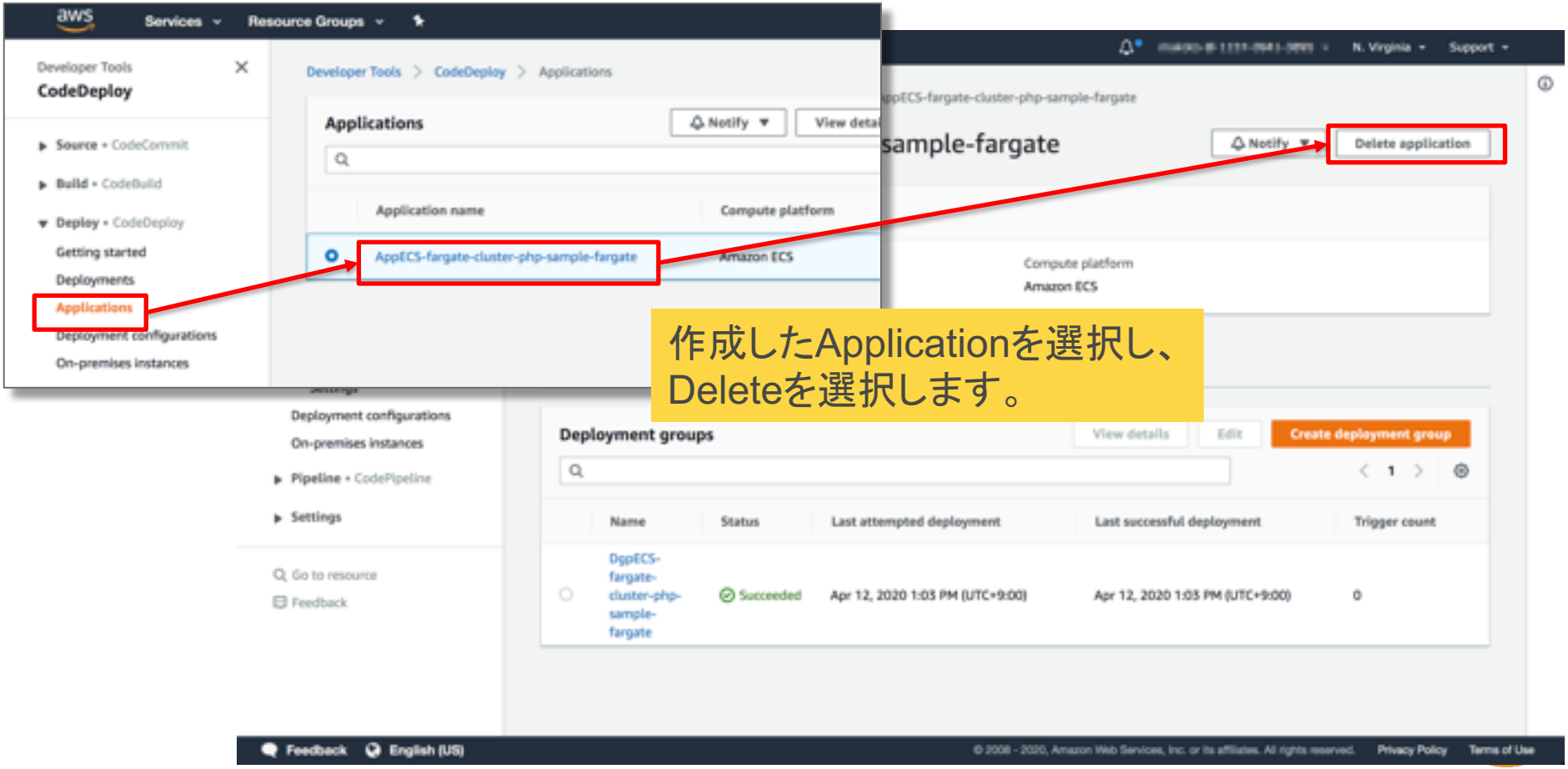
- CodePipelineパイプライン
- CodeDeployアプリケーション
- CodeBuildビルドプロジェクト
- CodeCommitレポジトリ
- S3バケット
- 各種IAMロール
- 各種CloudWatchログ
- ECRレポジトリ
- ECSクラスター
- ECSタスク定義
- ALBターゲットグループ
- ALB
- Cloud9環境
- CloudFormation

CodePipeline の削除

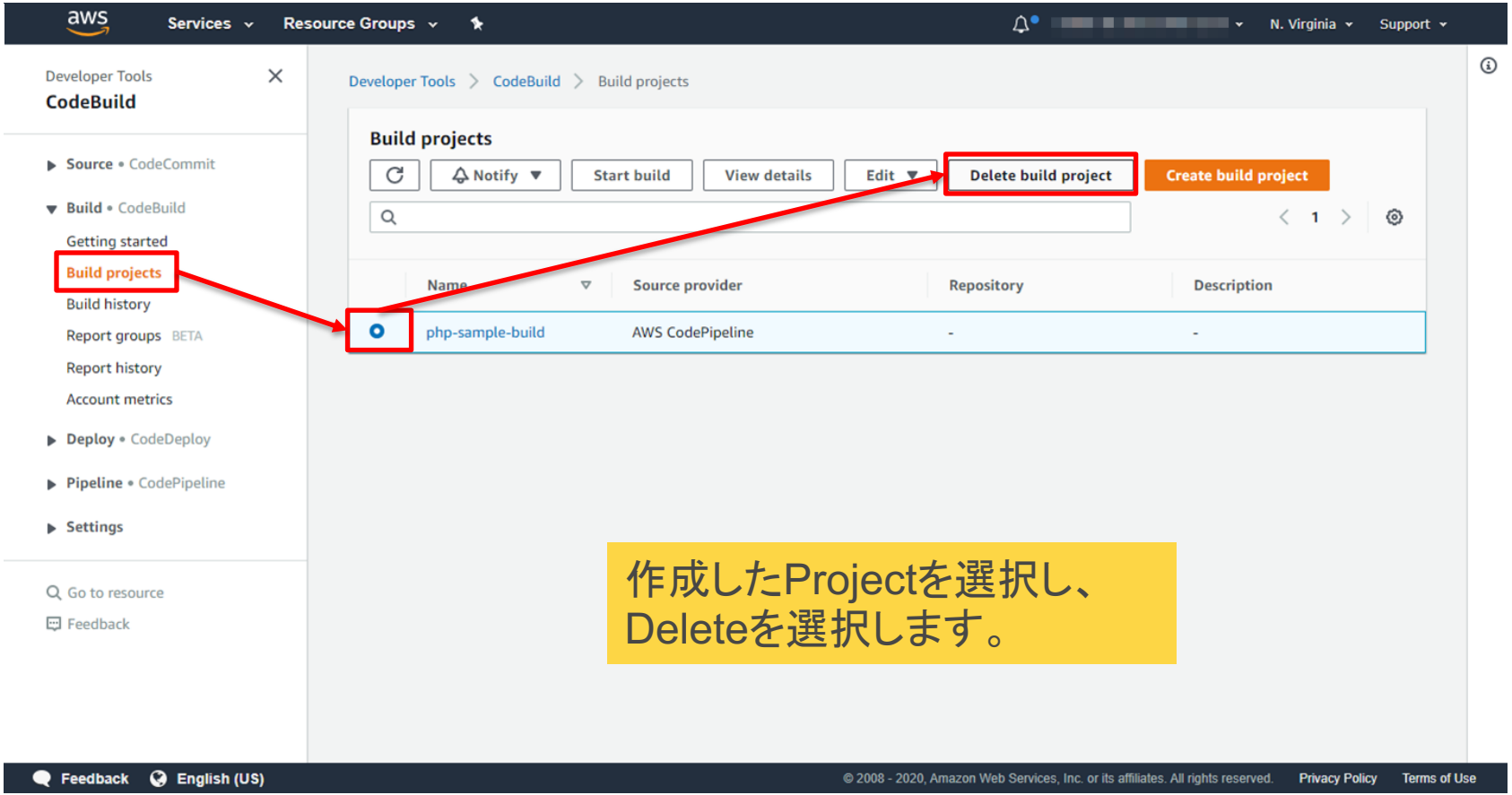


作成したPipelineを選択し、
Deleteを選択します。

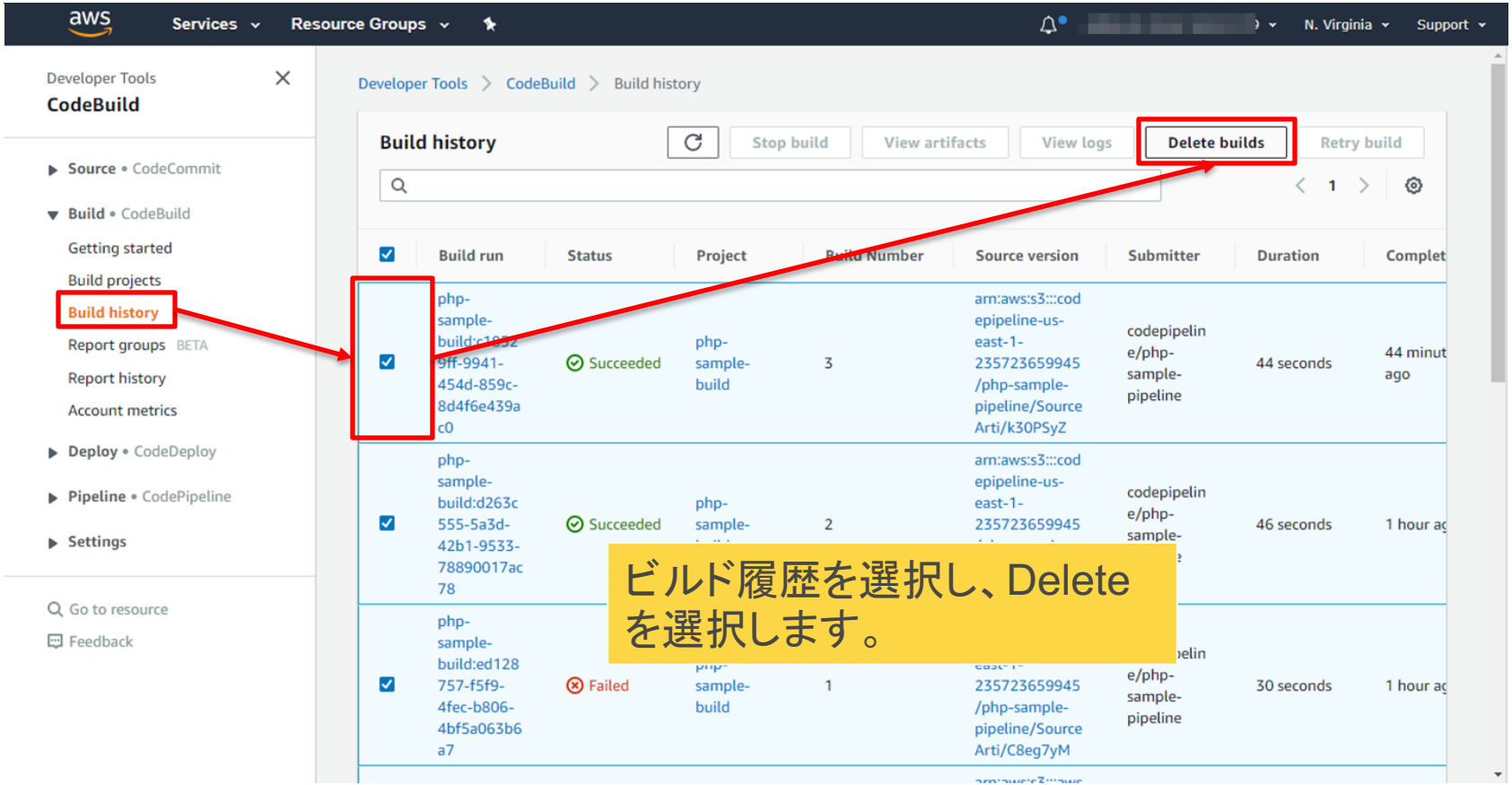
CodeDeploy の削除



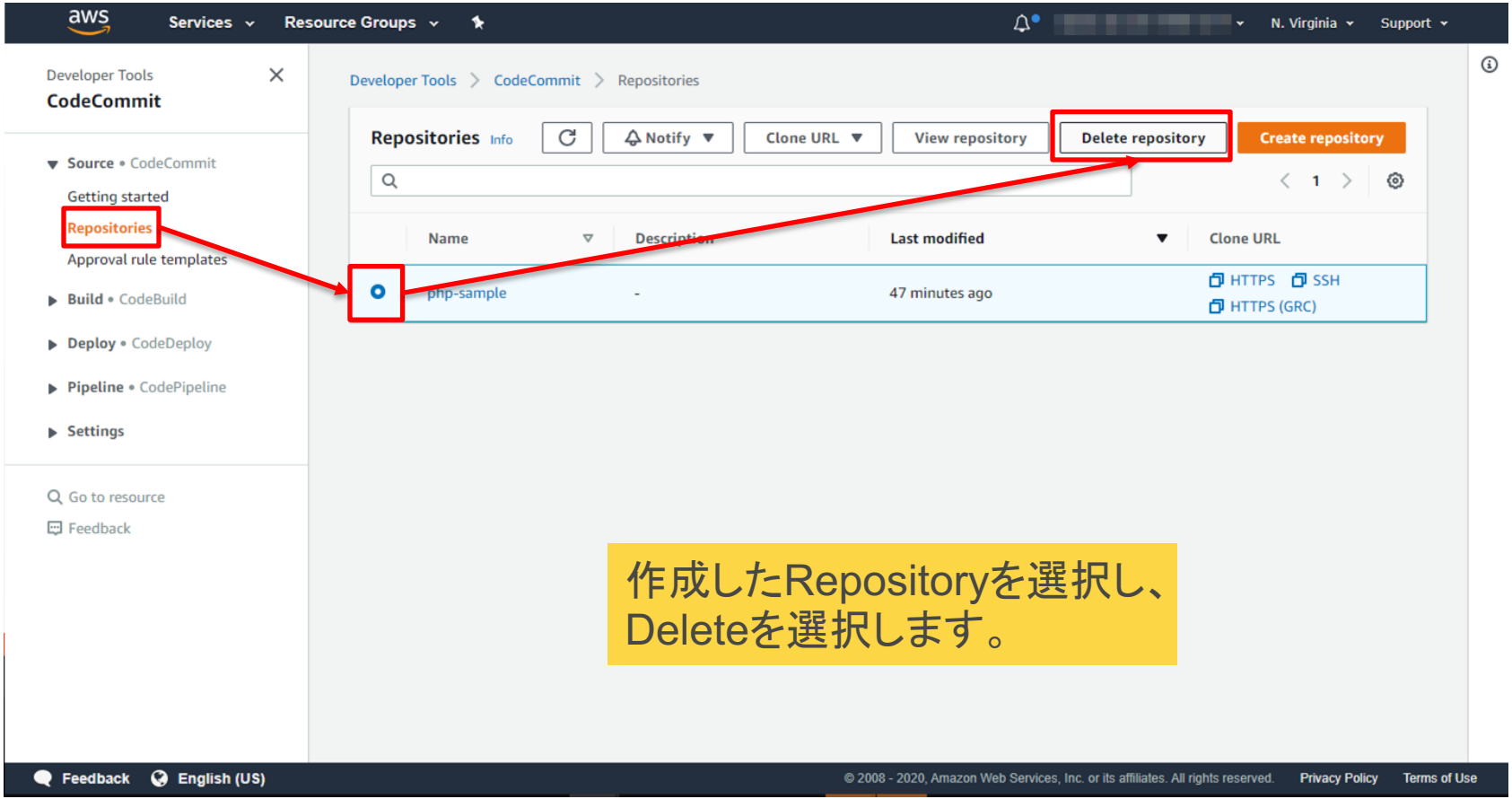
CodeBuild の削除



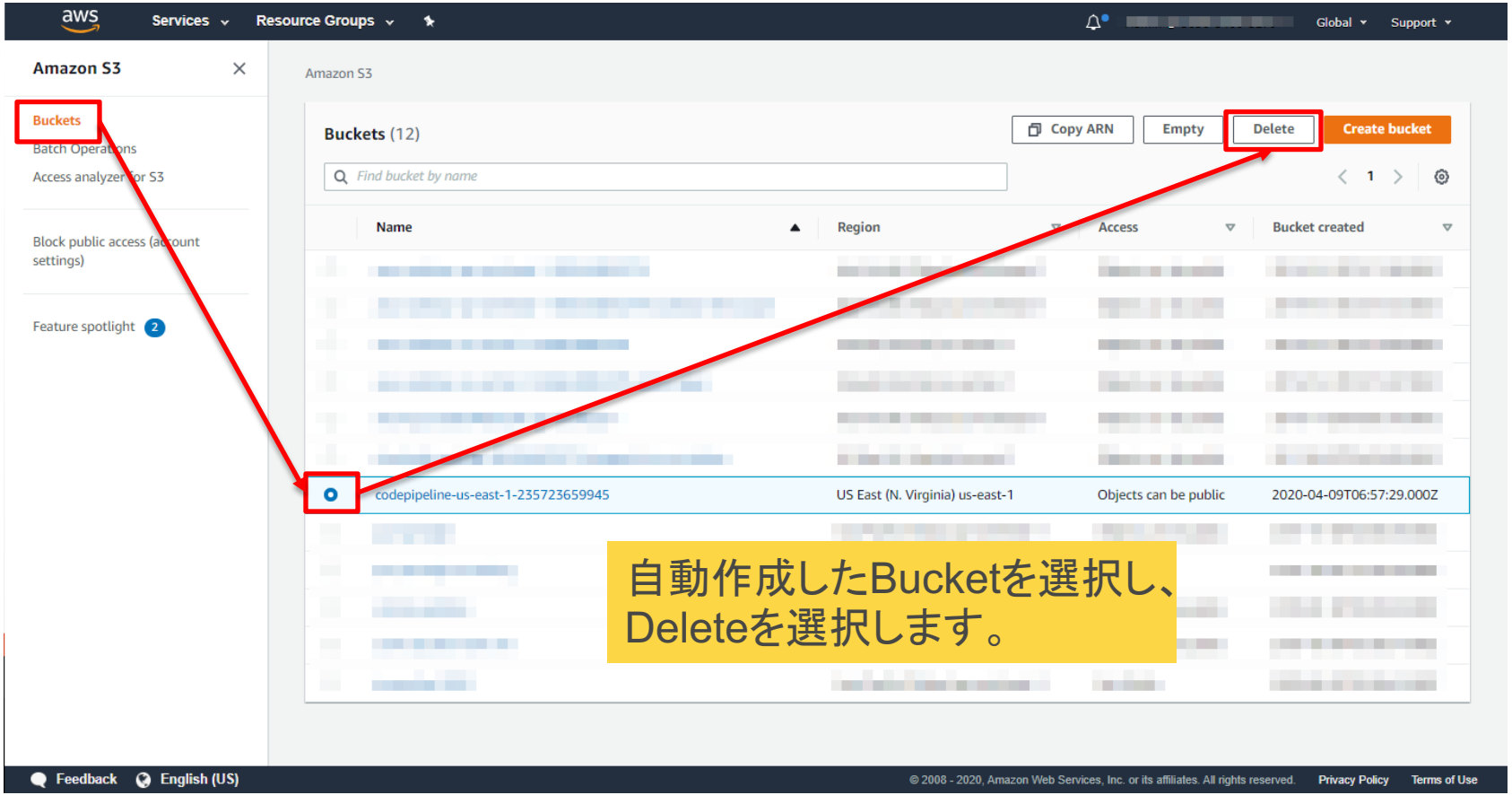
CodeBuild の削除



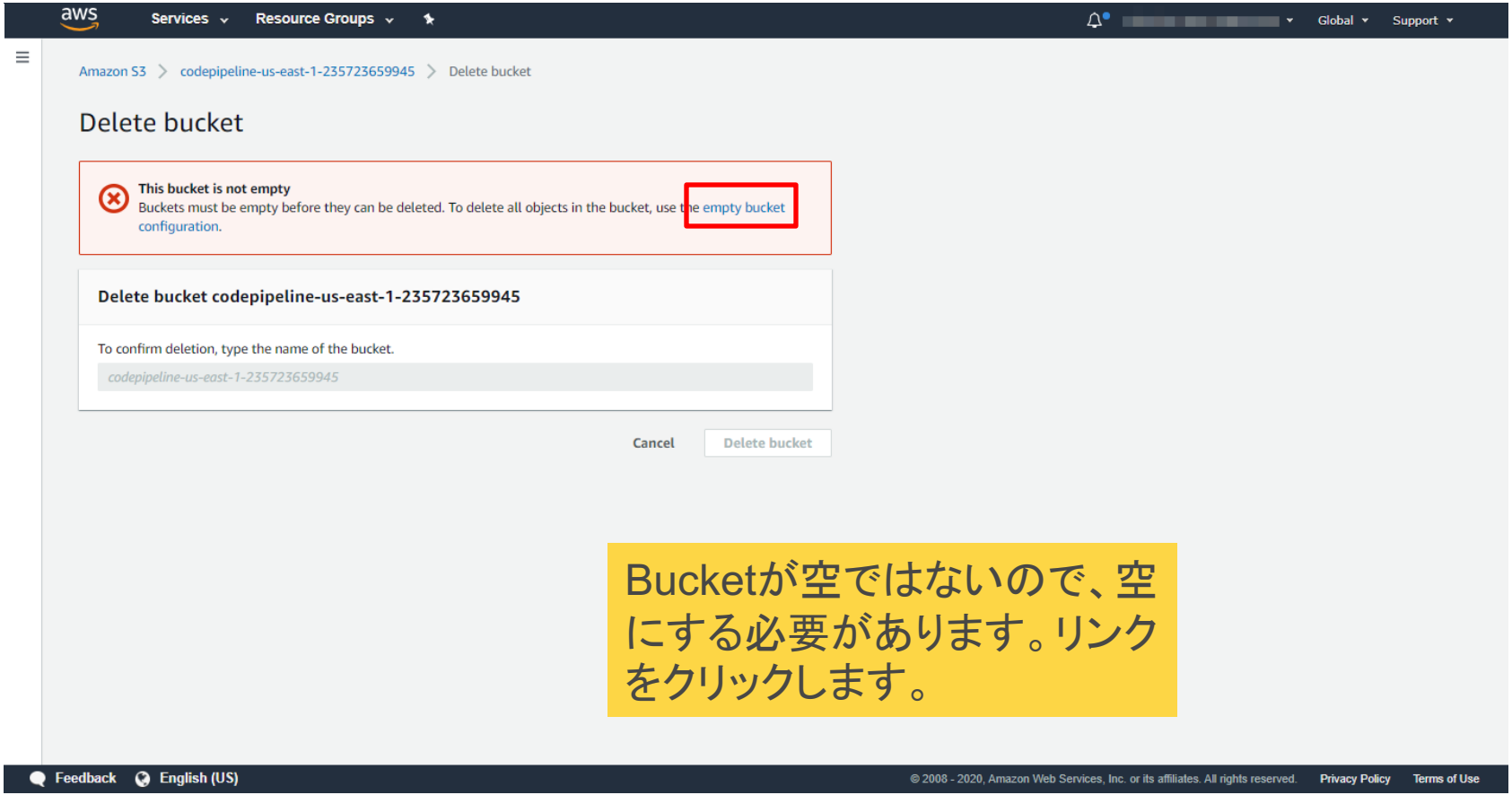
CodeCommit の削除



S3 bucket の削除

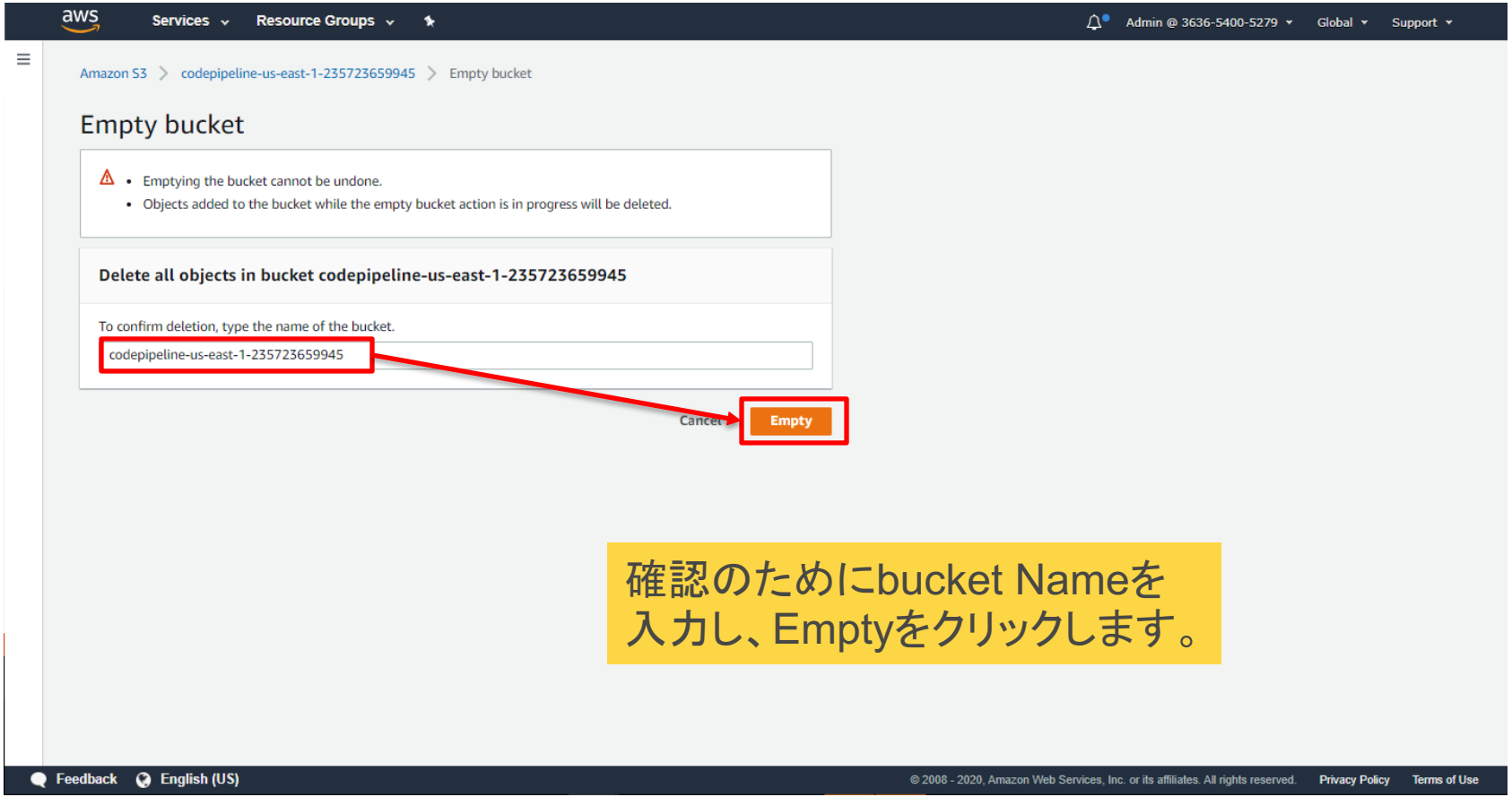


S3 bucket の削除



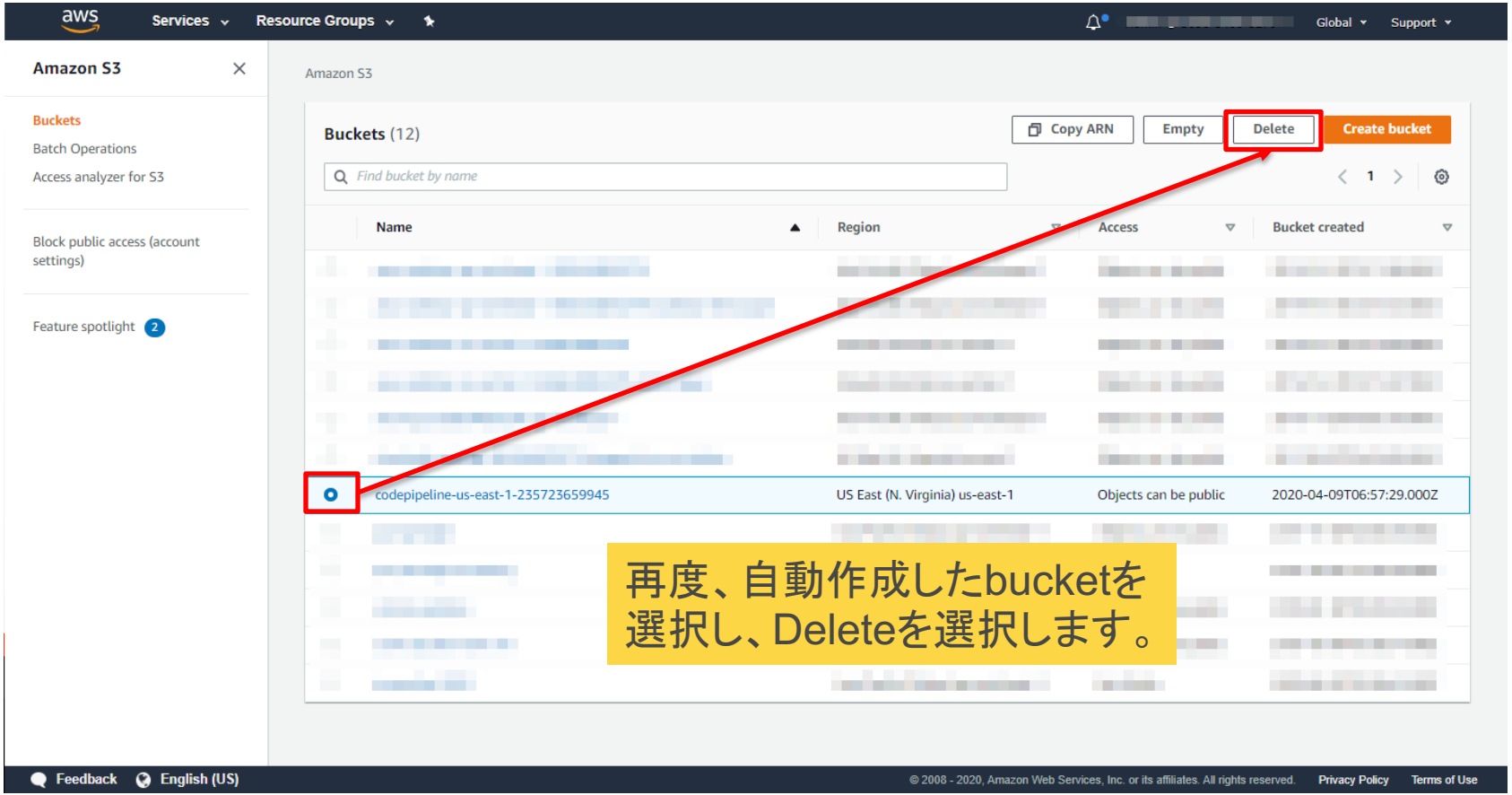
Bucketが空ではないので、空にする必要があります。リンクをクリックします。

S3 bucket の削除

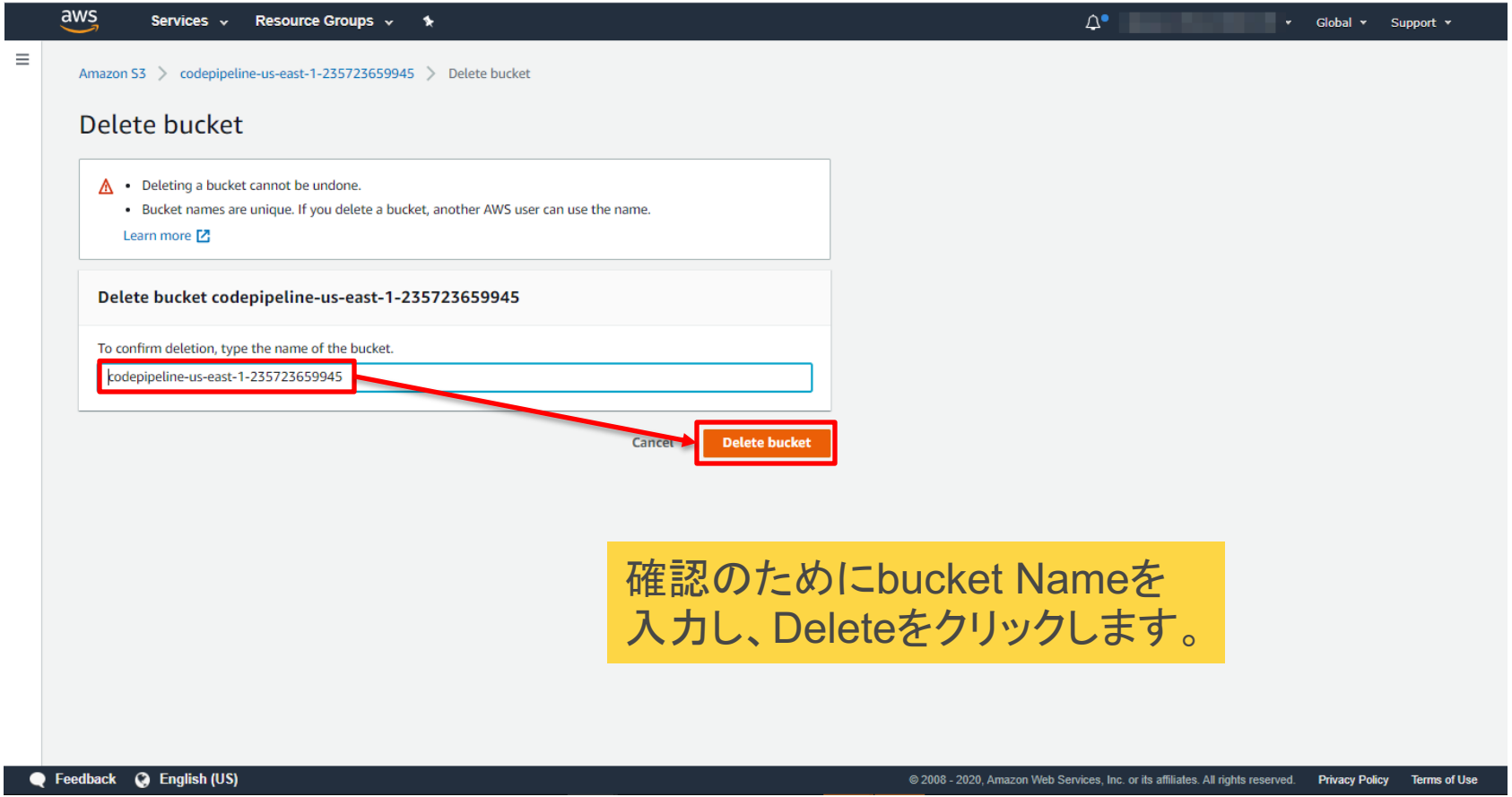


確認のためにbucket Nameを
入力し、Emptyをクリックします。

S3 bucket の削除

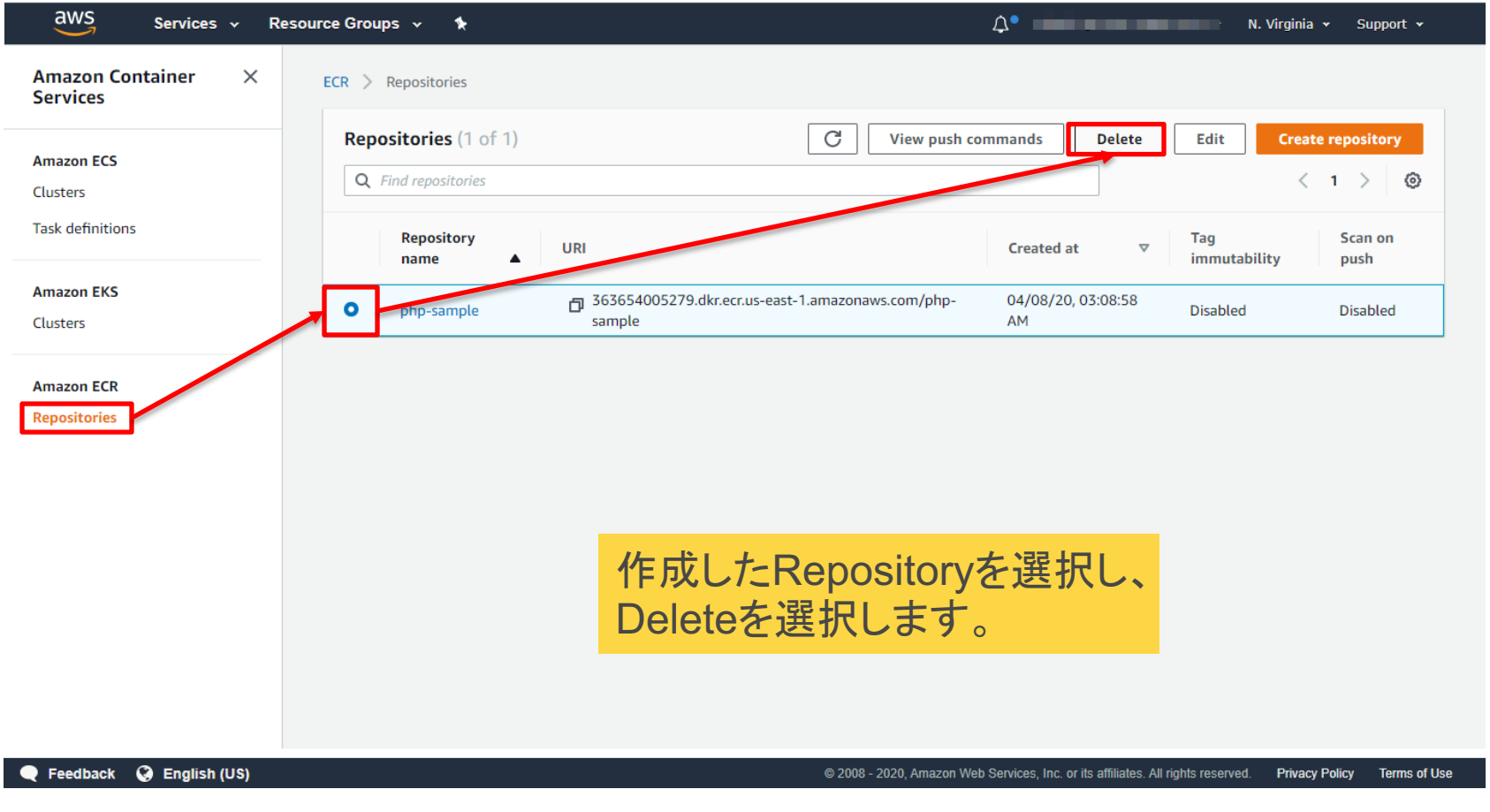


S3 bucket の削除

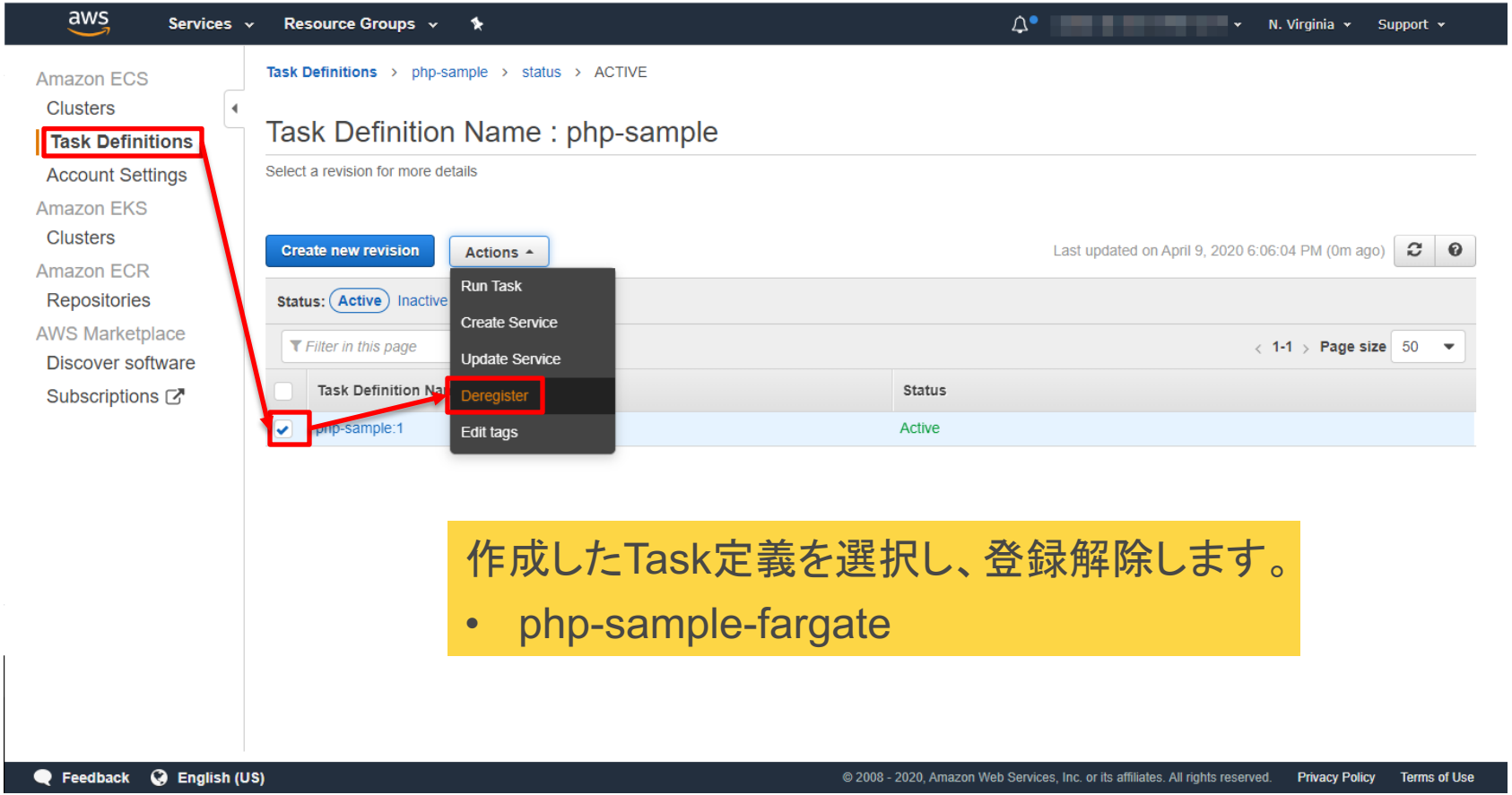


確認のためにbucket Nameを
入力し、Deleteをクリックします。

ECR の削除


aws
Services
Resource Groups
N. Virginia
Support
Amazon Container Services
Amazon ECS
Clusters
Task definitions
Amazon EKS
Clusters
Amazon ECR
Repositories
ECR > Repositories
Repositories (1 of 1)
View push commands
Delete
Edit
Create repository
Find repositories
1
Repository name
URI
Created at
Tag immutability
Scan on push
php-sample
363654005279.dkr.ecr.us-east-1.amazonaws.com/php-sample
04/08/20, 03:08:58 AM
Disabled
Disabled
作成したRepositoryを選択し、
Deleteを選択します。
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

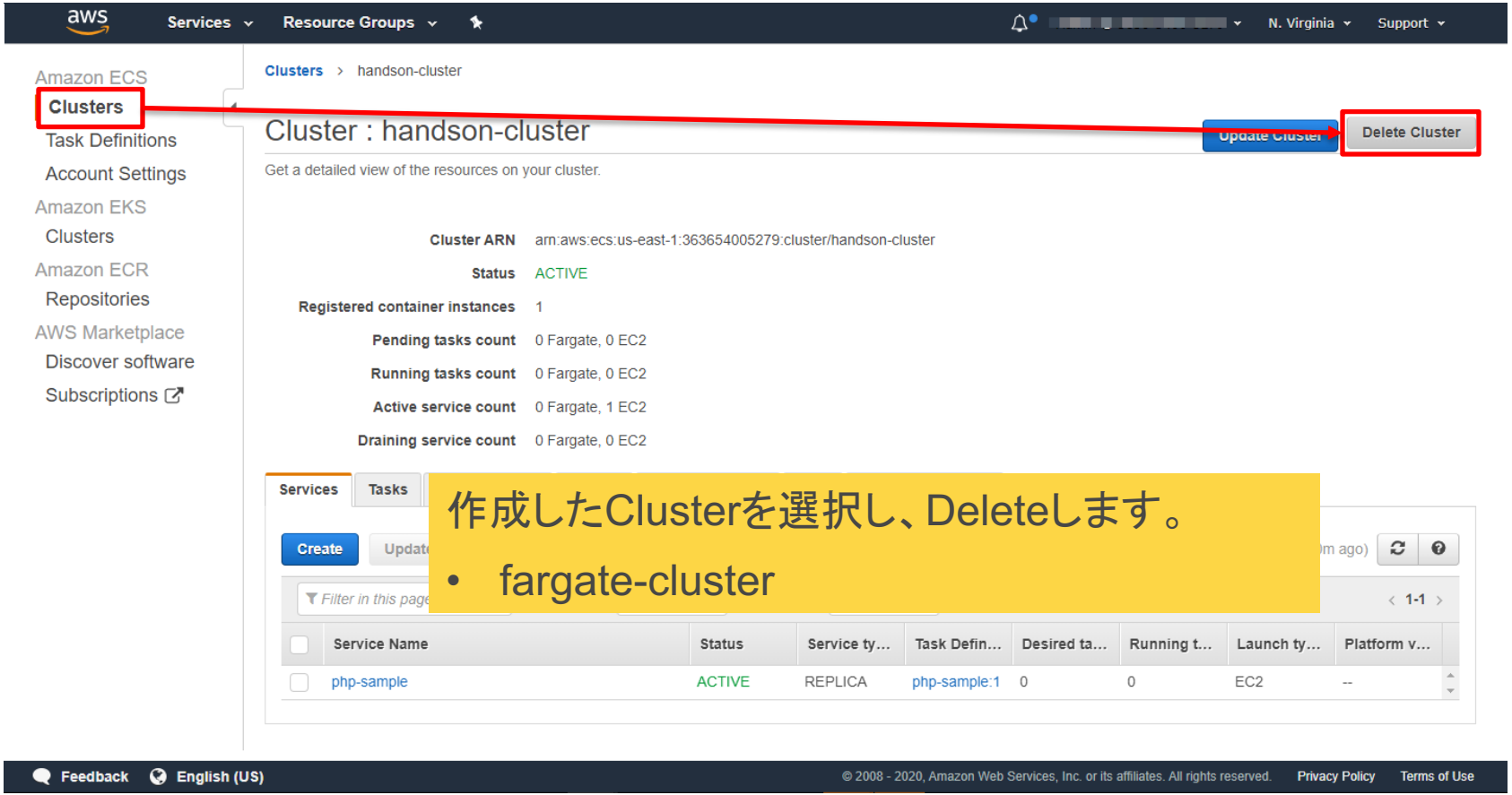
ECS Task definitions の削除



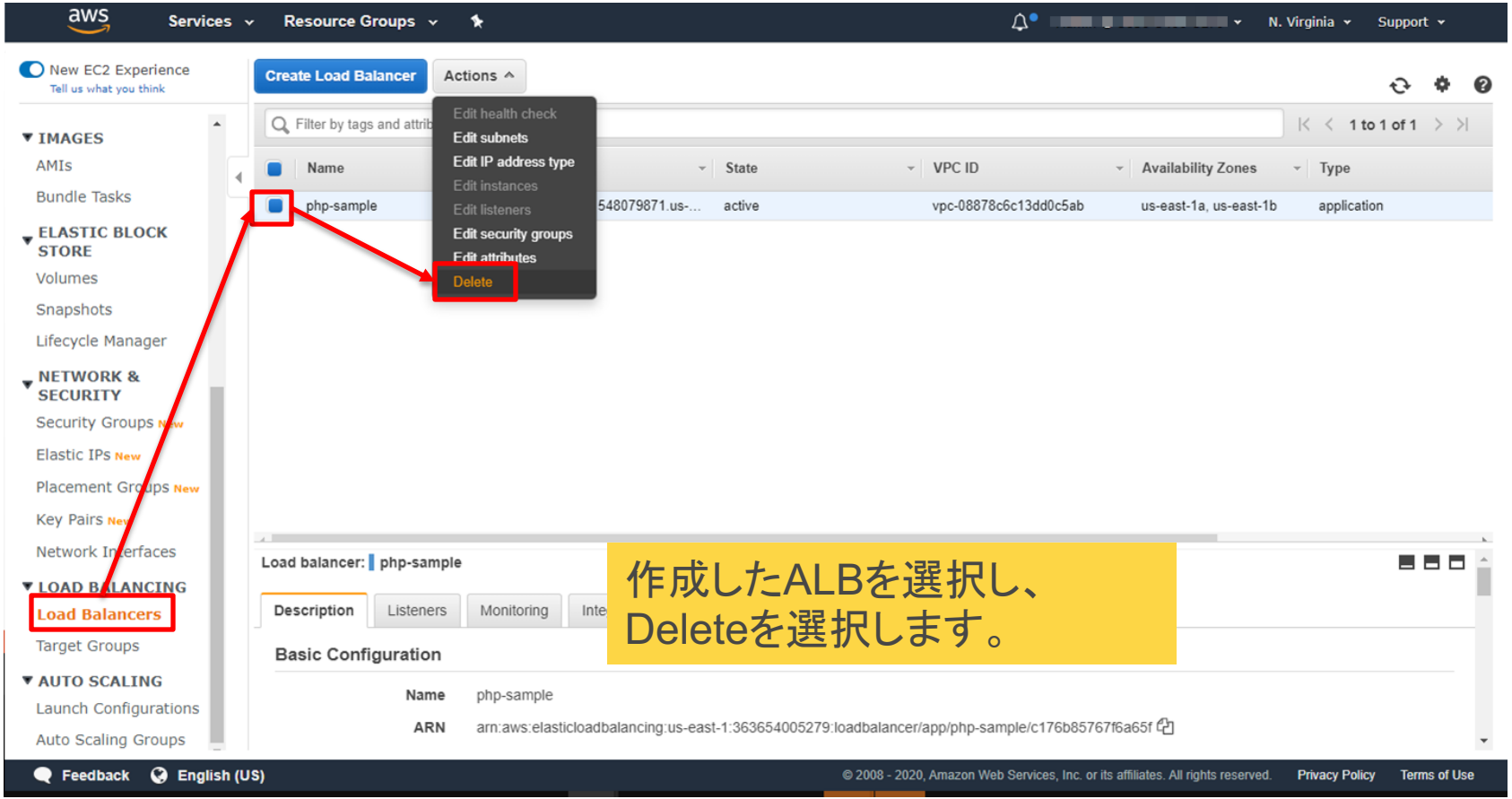
作成したTask定義を選択し、登録解除します。

- php-sample-fargate

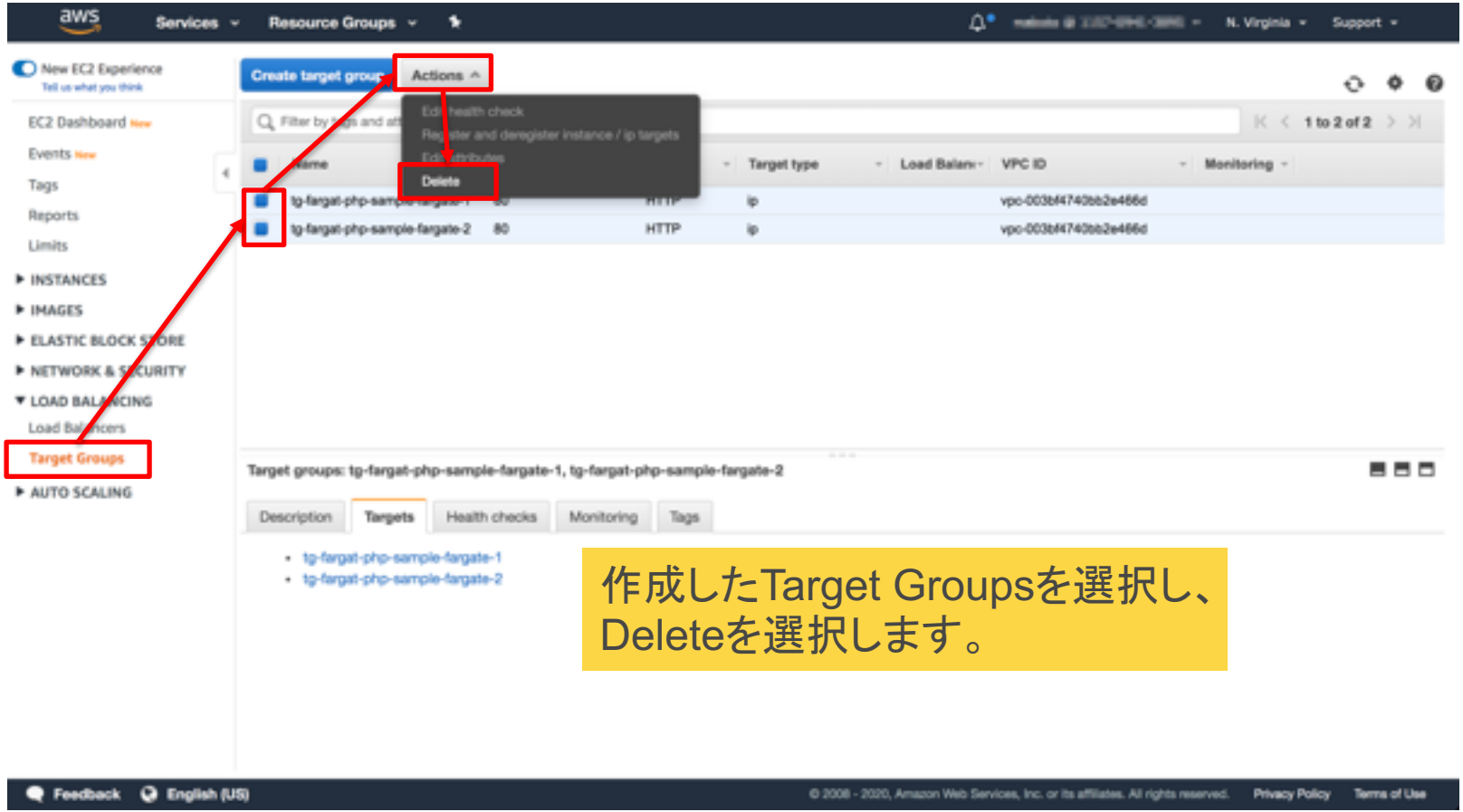
ECS Clusters の削除



ALB の削除

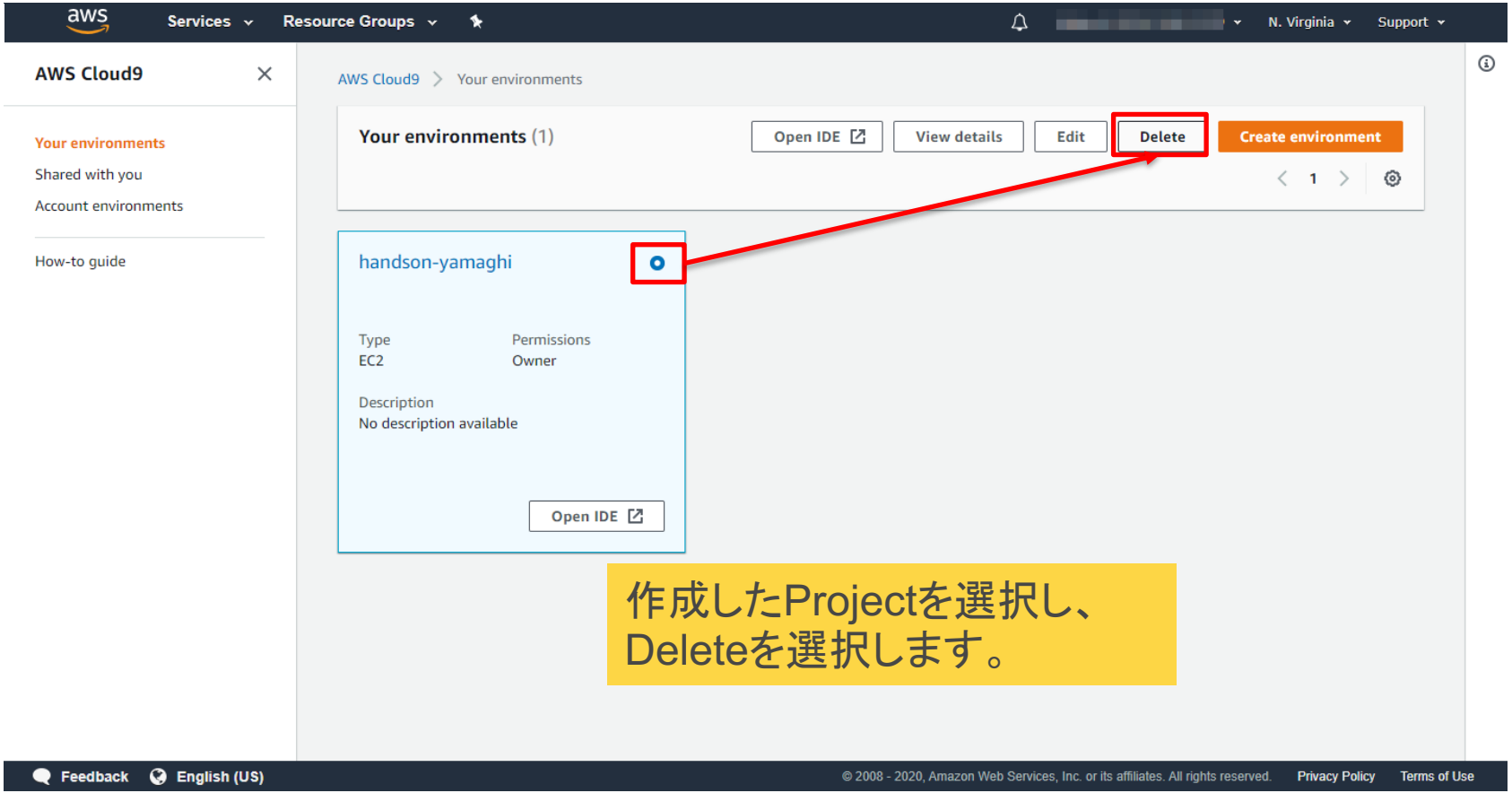

Services
Resource Groups
N. Virginia
Support
New EC2 Experience
Tell us what you think
Create Load Balancer
Actions
Edit health check
Edit subnets
Edit IP address type
Edit instances
Edit listeners
Edit security groups
Edit attributes
Delete
Filter by tags and attrib
1 to 1 of 1
Name
State
VPC ID
Availability Zones
Type
php-sample
548079871.us-...
active
vpc-08878c6c13dd0c5ab
us-east-1a, us-east-1b
application
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
Lifecycle Manager
NETWORK & SECURITY
Security Groups New
Elastic IPs New
Placement Groups New
Key Pairs New
Network Interfaces
LOAD BALANCING
Load Balancers
Target Groups
AUTO SCALING
Launch Configurations
Auto Scaling Groups
Load balancer: php-sample
Description
Listeners
Monitoring
Basic Configuration
Name
php-sample
ARN
arn:aws:elasticloadbalancing:us-east-1:363654005279:loadbalancer/app/php-sample/c176b85767f6a65f
作成したALBを選択し、
Deleteを選択します。
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

ALB Target Groups の削除

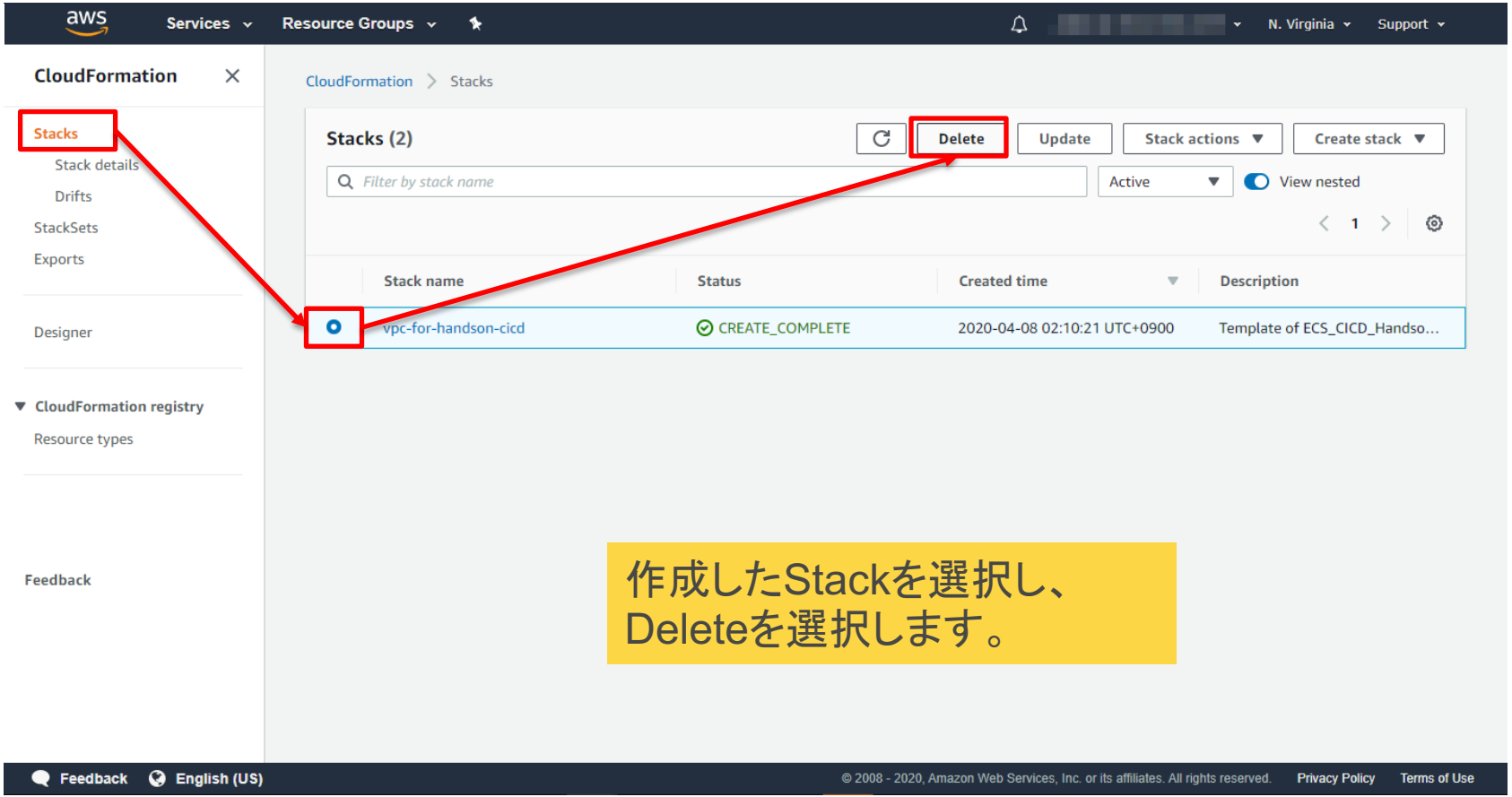


作成したTarget Groupsを選択し、
Deleteを選択します。

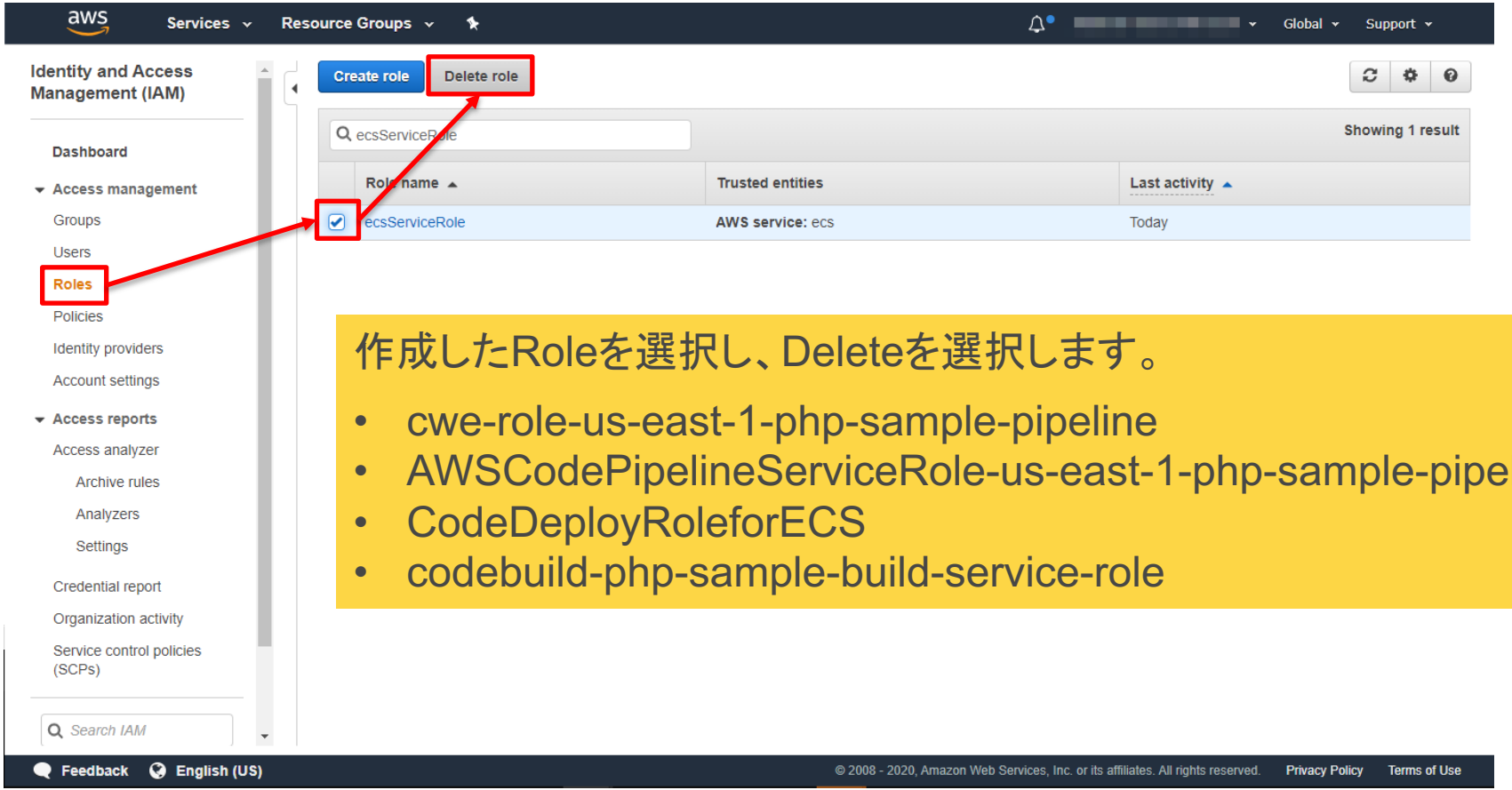
Cloud9 の削除


aws
Services
Resource Groups
N. Virginia
Support
AWS Cloud9
Your environments
Shared with you
Account environments
How-to guide
AWS Cloud9 > Your environments
Your environments (1)
Open IDE
View details
Edit
Delete
Create environment
1
handson-yamaghi
Type
EC2
Permissions
Owner
Description
No description available
Open IDE
作成したProjectを選択し、
Deleteを選択します。
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

CloudFormation の削除



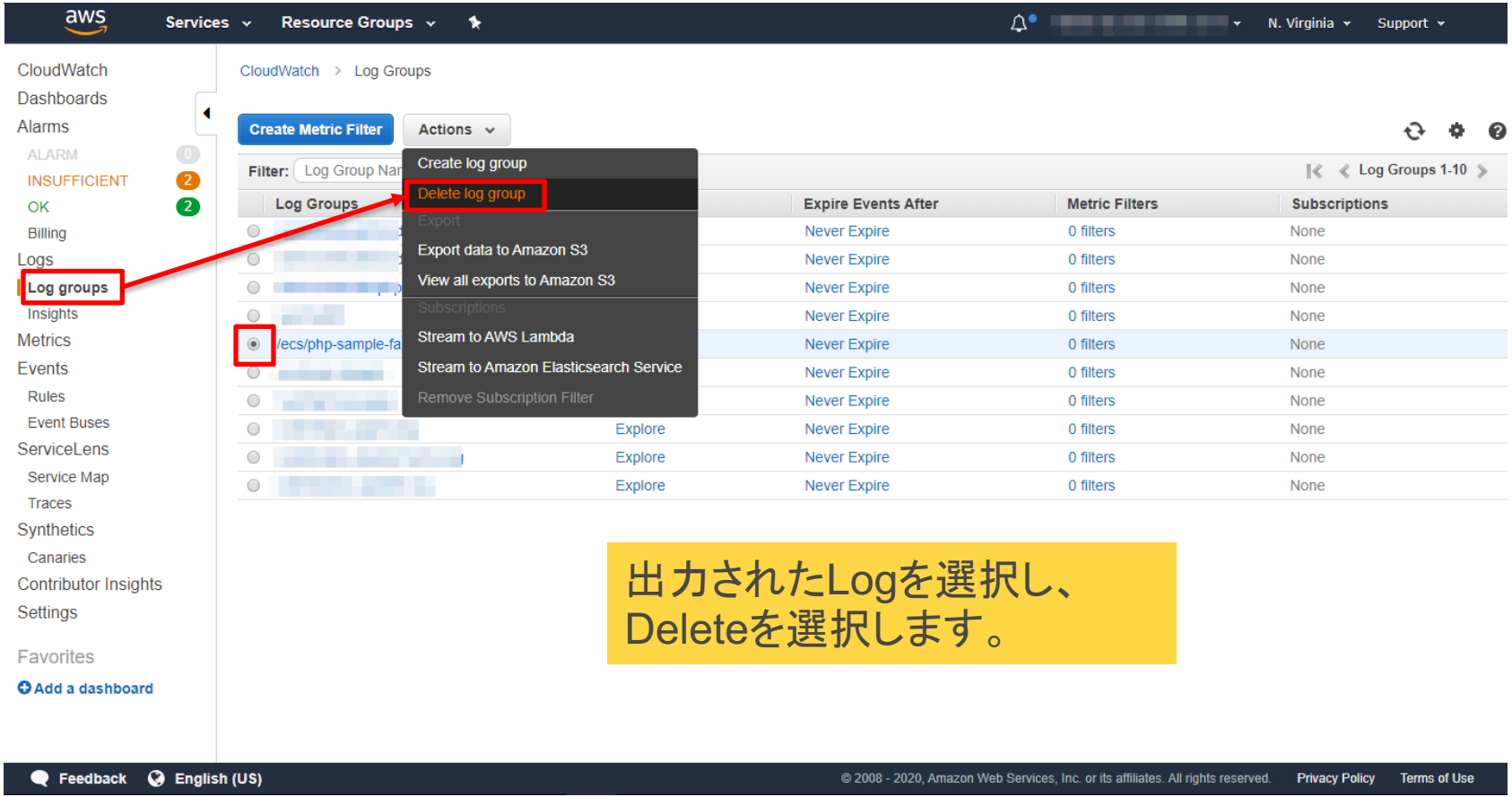
IAM Role の削除



作成したRoleを選択し、Deleteを選択します。

- cwe-role-us-east-1-php-sample-pipeline
- AWSCodePipelineServiceRole-us-east-1-php-sample-pipeline
- CodeDeployRoleforECS
- codebuild-php-sample-build-service-role

CloudWatch Logs の削除



出力されたLogを選択し、
Deleteを選択します。

Thank you!