

# 課題演習 No.3

(CloudFormation + Auto-scaling  
+ CloudWatch + SNS)



# 課題演習4 (CloudFormation + Auto-scaling+ CloudWatch + SNS)

## 目的:

ある企業でWebサイトを構築する際の検証用と本番用環境に対して、テンプレートを使用して同じ環境での構築を可能にしたいと考えている。今回はその検証用環境を作成したい。

Webページはサイト閲覧時にセキュアな通信で接続できるようにし、さらには運用面でテンプレート環境をベースにしてEC2の自動スケーリングや、状態監視によるアラーム通知といった機能を追加したいと考えている。

# 1:要件

■以下の要件でインフラを構築してください。

1. 下記①-③を満たす1つの環境を自動構築する。

①2つのAZでパブリックサブネットを作成

②EC2を各パブリックサブネットに設置し、WebサーバはApacheを使用する。

※②のEC2は事前にAMIを作成しておく。

③ALBを使用し、ターゲットグループには各EC2を設定する

(ヒント:CF作成時の「詳細を指定」でスタックの名前をymlファイル名で作成する。

これは2つのymlファイルがスタックの名前で連携されているため。 )

-スタックによる構築後-※以下は手動設定(マネージメントコンソール)で行う

2. Webページにはドメインを使用した暗号化されたセキュアな通信をしたい。

3. EC2がCPU負荷率が70%を超えた際に自動でスケールアウト、  
負荷率が20%を下回った際にはスケールインできるようにしたい。

4. モニタリングで閾値を超えた際にはアラートを担当者へ  
Eメールで発信できるように設定する。

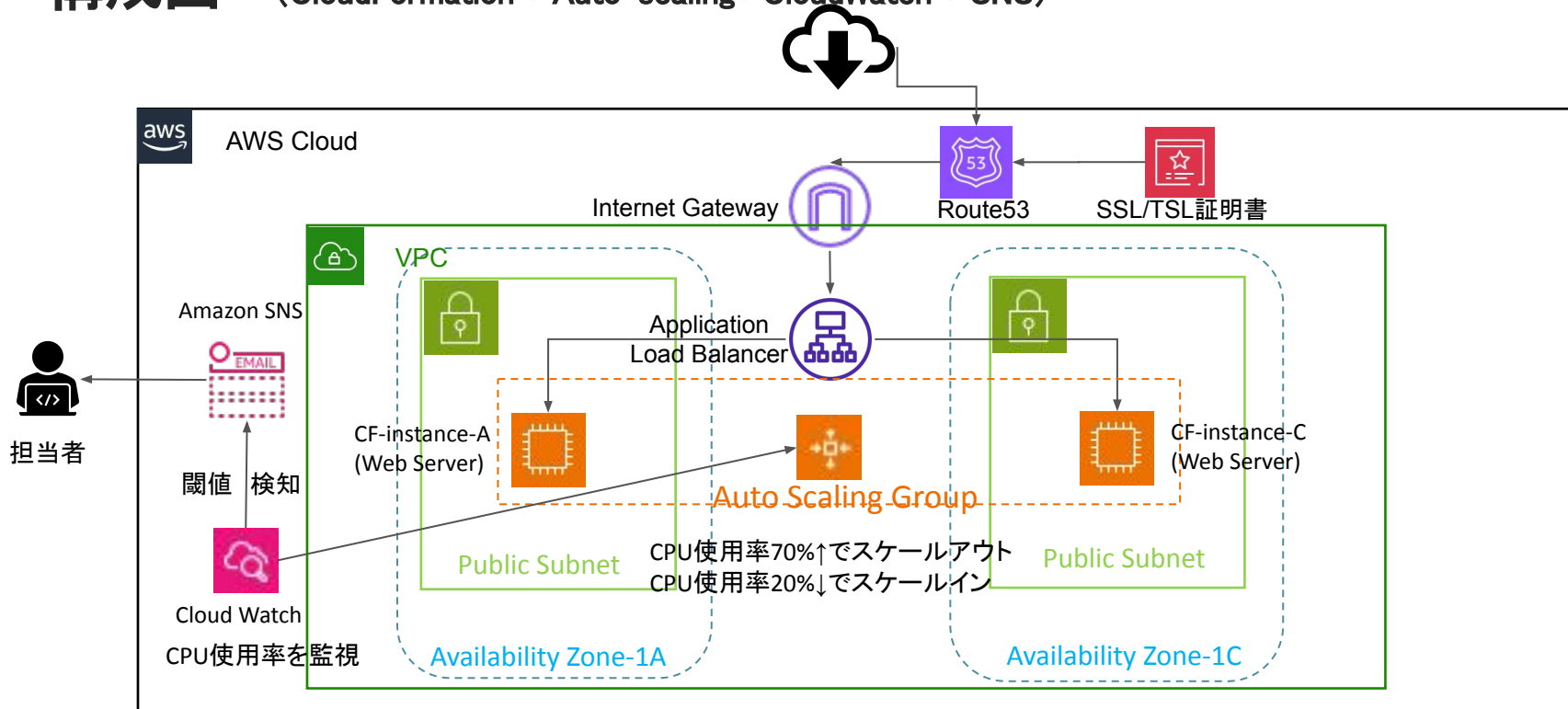
## 2: 設計

### ■要件に対しての仕様(解決策)

No	要件	仕様(解決策)
1	下記の条件を満たす1つの環境を自動構築する。 ・2つのAZでパブリックサブネットを作成 ・EC2を各パブリックサブネットに設置、WebサーバはApacheを使用する。 ・ALBを使用し、ターゲットグループには各EC2を設定する	EC2インスタンスを作成し、そのインスタンスへapacheサーバーをインストールする。そこからAMIを作成してAMIIDを控えておく。YAMLファイルの中に2つのAZでパブリックサブネットを構築を明示するコードを記載する。控えたAMIIDを使い、EC2をパブリックサブネットの中に構築を明示するコードを記載する。ALBがターゲットグループとして各EC2になるようにコードを記載する。CloudFormationでインフラをIaC化してスタック作成で自動構築。
2	Webページにはドメインを使用した暗号化されたセキュアな通信をしたい。	Route53のホストゾーン内にAレコードを作成し、そのドメインでALBをエイリアスとして指定する。
3	EC2がCPU負荷率が70%を超えた際に自動でスケールアウト、CPU負荷率が20%を下回った際にはスケールインできるようにしたい。	Auto Scalingを使用して、しきい値を超えたら自動でインスタンスの増減できるようにする。
4	モニタリングで閾値を超えた際にはアラートを、担当者へEメールで発信できるように設定する。	Cloud WatchでAuto Scaling Groupを監視し、しきい値を超えた際にはCloud Watch Alarmがアラームを発報し、SNSを経由してサブスクリプションで設定してる担当者のメールアドレスへメール送信する。

# 2: 設計

## -構成図- (CloudFormation + Auto-scaling+ CloudWatch + SNS)



# 3: 構築(実装)

## ①EC2インスタンスの作成

### インスタンスを起動

Amazon EC2 では、AWS クラウドで実行される仮想マシン(インスタンス)を作成できます。以下の順なステップに従って

#### 名前とタグ

名前

kada4-instance

さらにタグ

#### ▼ アプリケーションおよび OS イメージ (Amazon マシンイメージ) 情報

AMI は、インスタンスの起動に必要なソフトウェア(オペレーティングシステム、アプリケーションサーバー、データベースのものが以下に表示されない場合は、AMI を検索または参照してください。

Q 同様のアプリケーションイメージと OS イメージを含むカテゴリ全体を検索します。

最新

自分の AMI

クイックスタート



#### Amazon マシンイメージ (AMI)

Amazon Linux 2023 AMI  
ami-0722984356ca5cb0c4 (64 ビット (x86), uefi-preferred) / ami-0d60652e03537ad94f (64 ビット (Arm), uefi)  
提供元: Intel, NVIDIA, Intel, Linux, ルートデバイスタイプ: vda

#### 説明

Amazon Linux 2023 は、5 年間の長期サポートを備えた、最新の汎用 Linux ベースの OS です。AWS 向けに最適化された、より実行するための安全で安価な高性能な実行環境を提供するように設計されています。

Amazon Linux 2023 AMI 2023.6.20250218.2 x86\_64 HVM kernel-6.1

アーキテクチャ

64 ビット (x86)

ブートモード

uefi-preferred

AMI ID

ami-0722984356ca5cb0c4

ユーザー

ec2-user

### インスタンスタイプ

t2.micro  
ファミリー: t2 1 vCPU 1 GB メモリ 無料枠: t2.micro オンデマンド Windows ベース 料金: 0.0188 USD per Hour  
オンデマンド S2D ベース 料金: 0.0152 USD per Hour オンデマンド Ubuntu Pro ベース 料金: 0.017 USD per Hour  
オンデマンド RHEL ベース 料金: 0.0296 USD per Hour オンデマンド Linux ベース 料金: 0.0152 USD per Hour

無料利用枠の対象

すべての世代

インスタンスタイプを比較

ソフトウェアがプリインストールされた AMI には追加料金がかかります

### ▼ キーペア (ログイン) 情報

キーペアを使用してインスタンスに安全に接続できます。インスタンスを起動する前に、選択したキーペアにアクセスできることを確認してください。

キーペア名 - 必須

test\_key\_pair

新しいキーペアの作成

### ▼ ネットワーク設定 情報

VPC - 必須

172.31.0.0/20

[デフォルト]

サブネット

172.31.0.0/20

VPC: vpc-035a5e6f05151f1a 子サブネット: sg-0b643e709bc8321e アベイラビリティゾーン: ap-northeast-1d  
ゾーンのタイプ: アベイラビリティゾーン 利用可能な IP アドレス: 4091 CIDR: 172.31.16.0/20

パブリック IP の自動割り当て

有効化

無料利用枠を超える場合は追加料金がかかります

ファイアウォール (セキュリティグループ) 情報

セキュリティグループとは、インスタンスのトラフィックを制御する一連のファイアウォールルールです。特定のトラフィックがインスタンスに到達できるようにルールを追加します。

セキュリティグループを作成

既存のセキュリティグループを選択する

共通のセキュリティグループ

セキュリティグループを編集

default sg-0b643e709bc8321e

VPC: vpc-035a5e6f05151f1a

セキュリティグループのルールを比較

ここで指定または参照したセキュリティグループは、すべてのネットワークインターフェイスに追加、またはすべてのネットワークインターフェイスから削除されます。

高度なネットワーク設定

### 3: 構築(実装)

## ①作成をしたインスタンスへSSH接続しapacheをインストール

```
C:\Users\edu>ssh -i C:\Users\edu\Downloads\test_key_pair.pem ec2-user@35.77.35.116
The authenticity of host '35.77.35.116 (35.77.35.116)' can't be established.
ED25519 key fingerprint is SHA256:HaUMay/A+yC6m1rhZt07oxzdEkvrr59HujMhu/cK8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '35.77.35.116' (ED25519) to the list of known hosts.
```

[illegible]

```
[root@ip-172-31-27-140 ec2-user]# sudo yum install -y httpd
Last metadata expiration check: 0:00:18 ago on Wed Feb 26 05:35:41 2025.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.62-1.amzn2023	amazonlinux	48 k
Installing dependencies:				
apr	x86_64	1.7.5-1.amzn2023.0.4	amazonlinux	129 k
apr-util	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	98 k
generic-logos-httpd	noarch	18.0-0-12.amzn2023.0.3	amazonlinux	19 k
httpd-core	x86_64	2.4.62-1.amzn2023	amazonlinux	1.4 M
httpd-filesystem	noarch	2.4.62-1.amzn2023	amazonlinux	14 k
httpd-tools	x86_64	2.4.62-1.amzn2023	amazonlinux	81 k
libbrotli	x86_64	1.0.9-4.amzn2023.0.2	amazonlinux	315 k
mailcap	noarch	2.1.49-3.amzn2023.0.3	amazonlinux	33 k
Installing weak dependencies:				
apr-util-openssl	x86_64	1.6.3-1.amzn2023.0.1	amazonlinux	17 k
mod_http2	x86_64	2.0.27-1.amzn2023.0.3	amazonlinux	166 k
mod_lua	x86_64	2.4.62-1.amzn2023	amazonlinux	61 k

## Transaction Summary

# 3: 構築(実装)

## ①Webサーバーのファイル内を編集するコマンドを入力

```
verifying... mod_lua-2.4.62-1.amzn2023.x86_64
Installed:
  apr-1.7.5-1.amzn2023.0.4.x86_64          apr-util-1.6.3-1.amzn2023.0.1.x86_64
  apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64  generic-logos-httpd-18.0.0-12.amzn2023.0
  httpd-2.4.62-1.amzn2023.x86_64          httpd-core-2.4.62-1.amzn2023.x86_64
  httpd-filesystem-2.4.62-1.amzn2023.noarch  httpd-tools-2.4.62-1.amzn2023.x86_64
  libbrotli-1.0.9-4.amzn2023.0.2.x86_64    mailcap-2.1.49-3.amzn2023.0.3.noarch
  mod_http2-2.0.27-1.amzn2023.0.3.x86_64    mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[root@ip-172-31-27-140 ec2-user]# sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service
[root@ip-172-31-27-140 ec2-user]# sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: inactive (dead)
   Docs: man:httpd.service(8)
[root@ip-172-31-27-140 ec2-user]# sudo systemctl status httpd
○ httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: inactive (dead)
   Docs: man:httpd.service(8)
[root@ip-172-31-27-140 ec2-user]# cd /var/www/html
[root@ip-172-31-27-140 html]# nano index.html
[root@ip-172-31-27-140 html]# systemctl enable httpd
[root@ip-172-31-27-140 html]# systemctl restart httpd
```



# 3: 構築(実装)

## ①HTMLでHello Worldを記述、保存、再起動

```
GNU nano 5.8 index.html
<html><h1> Hello World! </h1></html>
```



**Hello World!**

# 3: 構築(実装)

## ②作成したインスタンスからAMIを作成し、AMIIDをYAMLファイルへ記述

**イメージを作成** 情報

イメージ (AMI と呼ばれます) は、EC2 インスタンスの起動時に適用され

---

**インスタンス ID**

☐ `i-0fc42d51cd3a90d8b (kada14-instance)`

**イメージ名**

最大 127 文字。作成後に変更することはできません。

**イメージの説明 - オプション**

最大 255 文字

☒ **インスタンスを再起動**

選択すると、Amazon EC2 はインスタンスを再起動して、アタッチされたボリューム

**インスタンスボリューム**

**ストレージタイプ**   **デバイス**   **スナップショット**

`EBSS`   `/dev/xvda`   ボリュームから新しいスナ

```
KeyName:
  Description : Name of an existing EC2 KeyPa
  Type: AWS::EC2::KeyPair::KeyName
  ConstraintDescription : Can contain only AS

Resources:
  MyEC2Instance1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: [REDACTED]
      InstanceType: t2.micro
      SubnetId: !ImportValue VPC-koutiku-Public
      BlockDeviceMappings:
        -
          DeviceName: /dev/xvda
          Ebs:
            VolumeType: gp2
            VolumeSize: 8
      Tags:
        - Key: Name
          Value: CF-public-1a
        KeyName: !Ref KeyName
        SecurityGroupIds:
          - !GetAtt "InstanceSecurityGroup.Group"

  MyEC2Instance2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: [REDACTED]
      InstanceType: t2.micro
      SubnetId: !ImportValue VPC-koutiku-Public
      BlockDeviceMappings:
        -
```

# 3: 構築(実装)

## ②Route53でホストゾーンを作成

Route 53 > ホストゾーン > ホストゾーンの作成

### ホストゾーンの作成 情報

#### ホストゾーン設定

ホストゾーンは、example.com などのドメインおよび

#### ドメイン名 情報

これは、トラフィックをルーティングするドメインの名前です。

example.com

有効な文字 :a-z、0-9、! " # \$ % & ' ( ) \* + , - / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~

#### 説明 - オプション 情報

この値で、同じ名前のホストゾーンを区別できます。

ホストゾーンは次の目的で使用されます。

説明は最大 256 文字です。0/256

#### タイプ 情報

このタイプは、インターネットまたは Amazon VPC アトラフィ

ホストゾーン

ホストゾーン

レコード (5) DNSSEC 署名 ホストゾーンのタグ (0)

#### レコード (5) 情報



レコードを削除

ゾーンファイルをインポート

レコードを作成

Automatic モードは最適なフィルタ結果に最適化された現在の検索動作です。モードを変更するには、[設定] に移動します。

検索 プロパティまたは値でレコードをフィルタリングする

タイプ

ルーティ...

エイリアス

< 1 >



<input type="checkbox"/>	レコード名	タイプ	ルーティ...	差別...	エイ...	値/トラフィックのルーテ...	TTL (秒)	ヘルス...
<input type="checkbox"/>	a02.ope-educ...	NS	シンプル	-	いいえ	ns-1293.awsdns-33.org. ns-509.awsdns-63.com. ns-748.awsdns-29.net. ns-1997.awsdns-57.co.uk.	172800	-
<input type="checkbox"/>	a02.ope-educ...	SOA	シンプル	-	いいえ	ns-1293.awsdns-33.org. aws...	900	-
<input type="checkbox"/>	_9d265d5c06...	CNAME	シンプル	-	いいえ	_180e8d4fde32d72ba68234...	300	-
<input type="checkbox"/>	www.a02.ope...	A	シンプル	-	いいえ	54.249.59.40	300	-
<input type="checkbox"/>	_1b7718bfe1...	CNAME	シンプル	-	いいえ	_342b2428e49f03f31ba695...	300	-

# 3: 構築(実装)

## ②作成したホストゾーン内のAレコードの作成、エイリアスでALBを指定

レコードを編集  | >

レコード名 [情報](#)      レコードタイプ [情報](#)

.a02.ope-education.link       ▼

ルートドメインのレコードを作成するには、空白のままにします。

☒ エイリアス

トラフィックのルーティング先 [情報](#)

▼

▼

X

エイリアスホストゾーン ID: Z14GRHDCWA56QT

ルーティングポリシー [情報](#)      ターゲットのヘルスを評価

▼      ☐ いいえ

キャンセル

# 3: 構築(実装)

## ③Route53で作成したホストゾーンを使いACMで証明書を発行し、YAMLファイルのALBのリスナーへ証明書ARNを記述

パブリック証明書をリクエスト

**ドメイン名**  
証明書は1つ以上のドメイン名を指定します。

完全修飾ドメイン名 [情報](#)

[この証明書に別の名前を追加](#)

この証明書にはさらに名前を追加できます。例えば、「www.example.com」の証明書をリクエストする場合、証明書がないか

**検証方法** [情報](#)  
ドメインの所有権を確認する方法を選択

☒ DNS 検証 - 推奨  
証明書リクエストとドメインのDNS設定を変更する場合は、このオプションを選択します。

☐ Eメール検証  
証明書リクエストとドメインのDNS設定を変更する許可がない場合、または承認許可を取得できない場合は、このオプション

**キーアルゴリズム** [情報](#)  
暗号化アルゴリズムを選択します。一部のアルゴリズムは、一部のAWSサービスでサポートされていない場合

☒ RSA 2048  
RSAは、最も広く使用されているキータイプです。

☐ ECDSA P 256  
証明書は署名はRSA 2048と同等です。

☐ ECDSA P 384  
証明書は署名はRSA 2048と同等です。

**タグ** [情報](#)  
リソースに関連付けられたタグがありません。

```
Port: 80
Protocol: HTTP

ELBListenerHTTPS:
Type: AWS::ElasticLoadBalancingV2::Listener
Properties:
  Certificates:
    - CertificateArn: arn:aws:acm:ap-northeast-1:5[REDACTED]
  DefaultActions:
    - TargetGroupArn: !Ref ELBtargetGP
      Type: forward
  LoadBalancerArn: !Ref InternetELB
Port: 443
Protocol: HTTPS
```

# 3: 構築(実装)

## ④YAMLファイルをCloudFormationでスタックの作成をし、インフラを展開

### スタックの作成

#### 前提条件 - テンプレートの準備

はC ジェネレーター で既存のリソースをスキャンしてテンプレートを作成することもできます。

#### テンプレートの準備

各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む JSON 文

##### 既存のテンプレートを選択

既存のテンプレートをアップロードまたは選択します。

#### テンプレートの指定

この GitHub リポジトリ には、新しいインフラストラクチャプロジェクトの開始に役立つサンプル CloudF

#### テンプレートソース

テンプレートを指定すると、保存先となる Amazon S3 URL が生成されます。テンプレートは、スタックのリソースおよびア

##### Amazon S3 URL

テンプレートに Amazon S3 URL を指定します。

##### テンプレートファイルのアップロード

テンプレートをコンソールに直接アップ

#### テンプレートファイルのアップロード

##### ファイルの選択

VPC.koutiku.yml

JSON または YAML 形式のファイル

### スタックの作成

#### 前提条件 - テンプレートの準備

はC ジェネレーター で既存のリソースをスキャンしてテンプレートを作成することもできます。

#### テンプレートの準備

各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む J

##### 既存のテンプレートを選択

既存のテンプレートをアップロードまたは選択します。

#### テンプレートの指定

この GitHub リポジトリ には、新しいインフラストラクチャプロジェクトの開始に役立つサンプル

#### テンプレートソース

テンプレートを指定すると、保存先となる Amazon S3 URL が生成されます。テンプレートは、スタックのリソースお

##### Amazon S3 URL

テンプレートに Amazon S3 URL を指定します。

##### テンプレートファイルのアップロード

テンプレートをコンソールに直

#### テンプレートファイルのアップロード

##### ファイルの選択

CF-instance.yml

JSON または YAML 形式のファイル

S3 URL: https://s3.ap-northeast-1.amazonaws.com/cf-templates-1dkrqk07f695q-ap-northeast-1/2f

#### スタック (2)

スタック名によるフィルター

#### ステータスのフィルター

アクティブ

ネストされているものを表示

< 1 >

#### スタック

CF-instance

2025-02-26 15:13:52 UTC+0900

CREATE\_COMPLETE

VPC-koutiku

2025-02-26 15:08:55 UTC+0900

CREATE\_COMPLETE

# 3: 構築(実装)

## ⑤AutoScalingで使う起動テンプレートの作成

SSLS3

Lambda

API Gateway

CloudFormation

LightSail

WAF & Shield

Route 53

AWS Application Migration Service

CloudWatch

スからテンプレートを作成

### 起動テンプレートを作成

起動テンプレートを作成することで、後で再利用、共有、起動できる保存済みインスタンス設定を作成できます。テンプレートには複数の

#### 起動テンプレート名と説明

起動テンプレート名 - 必須

kada14-template

このアカウントに固有である必要があります。最大 128 文字。スペースや、「&」、「\*」、「@」などの特殊文字は使用できません。

テンプレートバージョンの説明

MyApp 用の 本番ウェブサーバー

最大 255 文字

**Auto Scaling のガイダンス** | [情報](#)

EC2 Auto Scaling でこのテンプレートを使用する場合は、これを選択します

☐ EC2 Auto Scaling で使用できるテンプレートをセットアップする際に役立つガイダンスを提供

▶ テンプレートタグ

▶ ソーステンプレート

## ⑤起動テンプレートの続きの設定

## ⑤起動テンプレートの続きの設定

起動テンプレートの詳細を以下で設定します。フィールドを空白のままにすると、フィールドが起動テンプレートに含まれません。

AMI は、インスタンスの起動に必要なソフトウェア層です (オペレーティングシステム、アプリケーションサーバー、アプリケーション)。この場合、AMI を検索または参照してください。

Q 同千両のアプリケーションイメージと OS イメージを含むタログ全体を検索します。

最新

### 自分のAMI

## クイックスタート

○ 起點テンプレートの設定に含めない

☒ 自己所有

○ 自分と其物

Amazon マシンイメージ (AMI)

```
ami-02c0d19b135d29254
2025-02-26T05:52:03.000Z  概構化: hvn  ENA  起動: true  ルートデバイスタイプ: ata  ブートモード: uefi-secureboot
```

說明

アーキテクチャ

x86\_64

AMI ID

ami-02cbd19b133d29254

▼ インスタンスタイプ [情報](#) | [アドバイスを要する](#)

### インスタンスタイプ

t2.micro	
ファミリー: t2	1 vCPU 1 GiB メモリ 実行状態: true オンデマンド Windows ベース 料金: 0.0198 USD per Hour
オンデマンド SUSE Linux Enterprise Server 12 SP5 64 ビット 料金: 0.0152 USD per Hour	オンデマンド Ubuntu Pro 18.04 LTS 64 ビット 料金: 0.017 USD per Hour
オンデマンド CentOS 7.6 64 ビット 料金: 0.0236 USD per Hour	オンデマンド Linux 64 ビット 料金: 0.0152 USD per Hour

知不足而奋进

ソフトウェアがプリインストールされた AMI には追加料金がかかります

▼ インスタンスタイプ [情報](#) | [アドバイスを要する](#)

インスタンスタイプ

t2.micro

ファミリー: x2 1vCPU 1GBメモリ 実行単位: true オンデマンド Windowsベース 料金: 0.0198 USD per Hour  
 オンデマンド SUSEベース 料金: 0.0152 USD per Hour オンデマンド Ubuntu Proベース 料金: 0.017 USD per Hour  
 オンデマンド RHELベース 料金: 0.0296 USD per Hour オンデマンド Linuxベース 料金: 0.0152 USD per Hour

ソフトウェアがプリインストールされた AMI には追加料金がかかります

▼ キーヘア (ログイン) [情報](#)

キーペアを使用してインスタンスに安全に接続できます。インスタンスを起動する前に、選択し

キーヘア名

test\_key\_pair

▼ ネットワーク設定 [情報](#)

## サブネット | 情報

VPC: vpc-08de08f4960678251 私有者: 584797980125 アベイラビリティーゾーン: ap-northeast-1a  
ゾーンのタイプ: アベイラビリティーゾーン 利用可能な IP アドレス: 249 CIDR: 10.0.0.0/24

サブネットを指定すると、ネットワークインターフェイスがテンプレートに自動的に追加されます。

**ファイアウォール (セキュリティグループ) | 情報**  
セキュリティグループとは、インスタンスのトラフィックを制御する一連のファイアウォールルールです。既定

● 既存のセマンティックグループを選択する

○ サイコロリティグループを

## 共通のセキュリティグループ | 信頼

セキュリティグループを編集

CF-Instance-InstanceSecurityGro

ここに追加または削除したセキュリティグループは、各バケットのネットワークインターフェイスに追加、または削除

### ▶ 高度なネットワーク設定



# 3: 構築(実装)

## ⑥AutoScaling の作成、サブネット2つ選択、ALBにアタッチ

### 起動テンプレートまたは起動設定を選択する info

この Auto Scaling グループによって起動されたすべての EC2 インスタンスに共通の設定が含まれる起動テンプレートを選択してください。

#### 名前

##### Auto Scaling グループ名

グループ名を指定する名前を入力します。

kada14-AutoScaling

現在のリージョンにあるこのアカウントに別名で、255 文字以内にする必要があります。

### 起動テンプレート info

#### 起動テンプレート

Amazon マシンイメージ (AMI)、インスタンスタイプ、キーペア、セキュリティグループなど、インスタンスレベルの

kada14-template

起動テンプレートを作成する info

#### バージョン

Default (1)

起動テンプレートバージョンを作成する info

#### 説明

-

#### AMI ID

ami-02cbd19b133d29254

#### キーペア名

test\_key\_pair

#### 追加の詳細

##### ストレージ (ボリューム)

-

#### キーペア名

test\_key\_pair

#### セキュリティグループ ID

sg-09aa007e12050bc2 info

#### 追加の詳細

##### ストレージ (ボリューム)

-

#### 作成日

Wed Feb 26 2025 15:38:34 GMT+0900 (GMT+0900)

### インスタンスタイプの要件 info

起動テンプレートと同じインスタンスタイプまたはインスタンスタイプを維持することも、異なるインスタンスタイプを指定することも、手動で

#### 起動テンプレート

kada14-template info

it-0344bd4eacdce0148

#### バージョン

Default

#### インスタンスタイプ

t2.micro

### ネットワーク

ほとんどのアプリケーションでは、マルチ Availability ザーンを使用して、Amazon EC2 Auto Scaling でゾーン間のインスタンスのバランシングを行います。

#### Availability ザーンとサブネット

選択した VPC の Auto Scaling グループが使用できる Availability ザーンとサブネットを指定します。

Availability ザーンとサブネットを選択する

ap-northeast-1a | 10.0.2.0/24

ap-northeast-1c | 10.0.2.0/24

サブネットを作成する info

#### Availability ザーンのディストリビューション - 新規

Auto Scaling は、Availability ザーン間で自動的にインスタンスのバランシングを行います。あるゾーンで起動した場合は、他のゾーンを選択して起動します。

#### バランシング (ベストエフォート)

ある Availability ザーンに起動が失敗した場合、Auto Scaling は別の Availability ザーンに起動を試みます。

#### バランシング

ある Availability ザーンに起動が失敗した場合、Auto Scaling は別の Availability ザーンに起動を試みます。

### 他のサービスと統合する - 省略可能 info

ロードバランサーを使用して、ネットワークトラフィックを複数のサーバーに分散します。VPC Lattice でサービス間通信を可能にします。ロードバランサーを指定することができます。また、ヘルスチェックの構成やモニタリングをカスタマイズすることもできます。

### ロードバランシング info

以下のオプションを使用して、Auto Scaling グループを既存のロードバランサー、または定義した新しいロードバランサーにアタッチ

☐ ロードバランサーがありません。

Auto Scaling グループへのトラフィックが、ロードバランサーに集中することはありません。

☒ 既存のロードバランサーにアタッチする。

既存のロードバランサーから選択します。

### 既存のロードバランサーにアタッチする

Auto Scaling グループにアタッチするロードバランサーを選択します。

☒ ロードバランサーのターゲットグループから選択する。

このオプションでは、Application Load Balancer、Network Load Balancer、または Gateway Load Balancer をアタッチできます。

☐ Classic Load Balancer から

### 既存のロードバランサーターゲットグループ

Auto Scaling グループと同じ VPC にあるインスタンスのターゲットグループのみを選択できます。

ターゲットグループを選択する

ELBtargetGP | HTTP

Application Load Balancer: CF-alb

# 3: 構築(実装)

## ⑥AutoScalingのグループサイズとスケーリングを設定

### グループサイズとスケーリングを設定する - 省略可能 Info

グループの希望する容量とスケーリング制限を設定します。グループのサイズを調整するために、オプションで

#### グループサイズ Info

Auto Scaling グループの初期サイズを設定します。グループを作成したあと、手動または自動スケーリングが

##### 希望する容量タイプ

希望する容量タイプの設定値を選択します。vCPUとメモリ(GB)は、一部のインスタンス型で構成された混合インスタンス

選択 (インスタンス数)

##### 希望するキャパシティ

グループサイズを設定してください。

2

#### スケーリング Info

Auto Scaling グループのサイズは、需要の変化に合わせて手動または自動で変更できます。

##### スケーリング制限

希望する容量をどれだけ増減できるかを制限を設定します。

最小の希望する容量

2

希望する容量と同じかそれ以下

最大の希望する容量

4

希望する容量と同じかそれ以上

##### 自動スケーリング - 省略可能

ターゲットの追跡ポリシーを使用するかどうかを選択します | Info

Auto Scaling グループを作成した後に、数のメトリクスベースのスケーリングポリシーとスケジューラされたスケーリング計

☒ スケーリングポリシーなし

Auto Scaling グループは初期サイズのままとなり、需要に合わせて動的にサイズ変更されることはありません。

#### インスタンスメンテナンスポリシー Info

インスタンス置き換えイベント中の Auto Scaling グループの可用性を制御します。これには、ヘルスチェックを待つために自動的に行われるイベント(リバンスイベントと呼ばれる)が含まれます。

可用性の要件状況に応じて置き換え方法を選択してください

☒ 行動の適応

☐ ポリシーなし

リバンスイベントの場合、新しいインスタンスは古いインスタンスが終了する前に起動します。その間のイベント中は、インスタンスは終了と同時に起動します。

☐ 可用性を優先

☐ 終了する前に起動

新しいインスタンスを起動し、起動したインスタンスの準備が整った後から、古いインスタンスを終了します。これにより、希望する容量を上回ることもあり、一時的にユ

### その他のキャパシティ設定

#### キャパシティ予約の詳細設定 | Info

Auto Scaling で既存のキャパシティ予約またはキャパシティ予約リソースグループのどちらかにインス

☒ デフォルト

Auto Scaling は、起動テンプレートのキャパシティ予約設定を使用します。

☐ なし

インスタンスはキャパシティ予約には起動されません。

☐ キャパシティ予約のみ

インスタンスはキャパシティ予約のみに起動されます。キャパシティが利用できない場合、イン

☐ 最初にキャパシティ予約

インスタンスは最初にキャパシティ予約に起動しようとし、キャパシティが利用できない場

### その他の設定

#### インスタンスのスケールイン保護

スケールインからの保護が有効になっている場合、新しく起動されたインスタンスはデフォルトで

☐ インスタンスのスケールイン保護を有効にする

#### モニタリング | Info

☒ CloudWatch 内でグループメトリクスの収集を有効にする

#### デフォルトのインスタンスのウォームアップ | Info

使用状況データがまだ集積されていないため、新しいインスタンスの CloudWatch メトリクスがグルー

☐ デフォルトのインスタンスのウォームアップを有効にする

# 3: 構築(実装)

## ⑥AutoScalingの動的スケーリングポリシーを70%以上で1インスタンスを追加と20%以下で1インスタンスを削除の設定

ポリシー

### 動的スケーリングポリシーを作成する

ポリシータイプ

ステップスケーリング

スケーリングポリシー名

CPU\_Add

CloudWatch アラーム

次の場合にいつでも負荷をスケールできるアラームを選択します。

ScaleOutAlarm-CF

[CloudWatch アラームを作成する](#)

アラームのしきい値を定義: 次のメトリクスディメンションに対して 60 秒間の 1 連続期間で CPUUtilization >= 70:

AutoScalingGroupName = CF-AutoScaling

アクションを実行する

追加

1

容量ユニット

次のとき: 70

<= CPUUtilization < +無限大

[ステップを追加する](#)

インスタンスのウォームアップ

300 秒

詳細

統合・新規

オートスケーリング

**インスタンス管理**

インスタンスの更新

アクティビティ

モニタリング

ポリシー

### 動的スケーリングポリシーを作成する

ポリシータイプ

ステップスケーリング

スケーリングポリシー名

CPU\_Remove

CloudWatch アラーム

次の場合にいつでも負荷をスケールできるアラームを選択します。

ScaleInAlarm

[CloudWatch アラームを作成する](#)

アラームのしきい値を定義: 次のメトリクスディメンションに対して 60 秒間の 1 連続期間で CPUUtilization <= 20:

AutoScalingGroupName = CF-AutoScaling

アクションを実行する

削除

1

容量ユニット

次のとき: 20

>= CPUUtilization > +無限大

### インスタンス (2)

検索 インスタンスをフィルタリングする

<input type="checkbox"/>	インスタンス ID	ライフサイクル	インスタンスタ...	重み付き容量	起動テンプレ...	アベイラビリテ...	ヘルスステータス	以下のものからの...
<input type="checkbox"/>	<a href="#">i-01234567890123456</a>	InService	t2.micro	-	<a href="#">kcdai4-template</a>	ap-northeast-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-01234567890123456</a>	InService	t2.micro	-	<a href="#">kcdai4-template</a>	ap-northeast-1a	Healthy	

# 3: 構築(実装)

## ⑦ SNSでスケールアウト、インのトピックを作成

① 新機能

Amazon SNS はバイスルーブット FIFO トピックをサポートするようになりました。詳細はこちら

### トピックの作成

詳細

タイプ | 情報

トピックの作成時にトピックタイプを変更することはできません。

☐ FIFO (先入れ先出し、先出し)

- 順番に保存されたメッセージの順で受け
- 1 回のメッセージ配信
- サブスクリプションプロトコル: SQS

☒ スタンダード

- ベストエフォート型メッセージの順で受け
- 少なくとも 1 回のメッセージ配信
- サブスクリプションプロトコル: SQS, Lambda, Data Firehose, HTTP, SNS, E メール、モバイルアプリケーションエンドポイント

名前

ScaleInAlarm-SNS

最大文字数は 256 文字です。英数字、ハイフン(-)、およびアンダースコア(\_) を含めることができます。

表示名 - オプション | 情報

SMS のサブスクリプションでこのトピックを使用するには、表示名を入力します。SMS メッセージには最初の 10 文字のみが表示されます。

My Topic

最大 100 文字。

#### ▶ 暗号化 - オプション

Amazon SNS はデフォルトで、転送中に暗号化を行います。サーバー側の暗号化を有効にすると、保存中に行った暗号化をトピックに追加します。

Amazon SNS はバイスルーブット FIFO トピックをサポートするようになりました。詳細はこちら

### トピックの作成

詳細

タイプ | 情報

トピックの作成時にトピックタイプを変更することはできません。

☐ FIFO (先入れ先出し、先出し)

- 順番に保存されたメッセージの順で受け
- 1 回のメッセージ配信
- サブスクリプションプロトコル: SQS

☒ スタンダード

- ベストエフォート型メッセージの順で受け
- 少なくとも 1 回のメッセージ配信
- サブスクリプションプロトコル: SQS, Lambda, Data Firehose, HTTP, SNS, E メール、モバイルアプリケーションエンドポイント

名前

ScaleOutAlarm-SNS

最大文字数は 256 文字です。英数字、ハイフン(-)、およびアンダースコア(\_) を含めることができます。

表示名 - オプション | 情報

SMS のサブスクリプションでこのトピックを使用するには、表示名を入力します。SMS メッセージには最初の 10 文字のみが表示されます。

My Topic

最大 100 文字。

#### ▶ 暗号化 - オプション

Amazon SNS はデフォルトで、転送中に暗号化を行います。サーバー側の暗号化を有効にすると、保存中に行った暗号化をトピックに追加します。

# 3: 構築(実装)

⑦作成したトピックからサブスクリプションを作成、送信先のメールアドレスを指定

Amazon SNS はハイスループット FIFO トピックをサポートするようになりました。 [詳細はこちら](#)

## サブスクリプションの作成

**詳細**

トピック ARN

プロトコル

サブスクライバのエンドポイントのタイプ

エンドポイント

Amazon SNS から通知を受け取れる E メールアドレス。

① サブスクリプションを作成した後、それを確認する必要があります。 [情報](#)

▶ **サブスクリプションフィルターポリシー - オプション** [情報](#)

このポリシーで登録者が受け取るメッセージがフィルター処理されます。

▶ **Redrive ポリシー (デッドレターキュー) - オプション** [情報](#)

配信不能メッセージをデッドレターキューに返します。

# 3: 構築(実装)

## ⑧ CloudWatchAlarmの設定

### メトリクスの選択

タイトルなしのグラフ



参照 マルチソースクエリ グラフ化したメトリクス オプション 発信元

メトリクス (4,252)

Tokyo すべて

ApiGateway 21 ApplicationELB 523 Aut

EBS 862 EC2 1663 イ

### メトリクスの選択

タイトルなしのグラフ



参照 マルチソースクエリ グラフ化したメトリクス オプション 発信元

メトリクス (1,663)

Tokyo すべて > EC2 すべて

Auto Scaling グループ別 122 イ

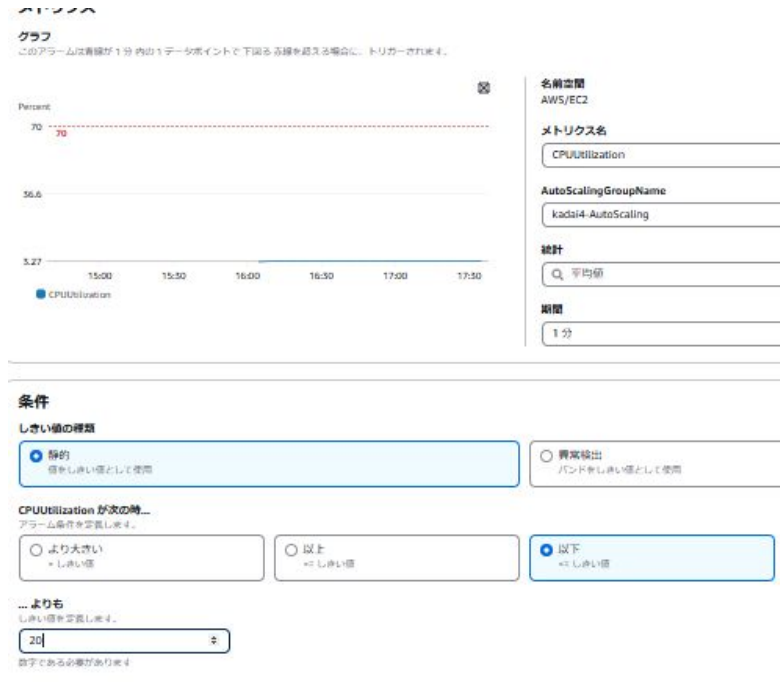
### 3: 構築(実装)

⑧メトリクス選択で関連付けるAutoScalingGroupを指定しCPU使用率のメトリクスを選択



# 3: 構築(実装)

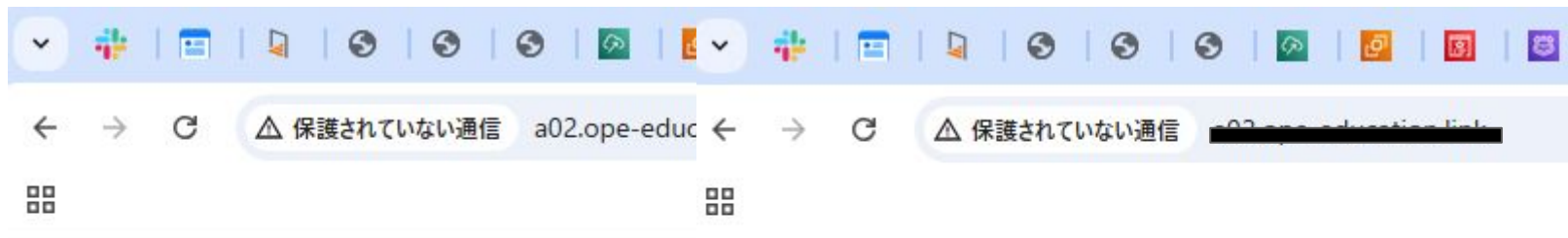
## ⑧メトリクスのしきい値を70%以上と20%以下でそれぞれ設定





## 4: テスト結果 (検証)

① 起動している二つのインスタンスへSSH接続し、Webサーバーファイル内を識別できるように1Aと1Cを加えて編集し、ALBでトラフィックの負荷分散が出来ているか確認




**Hello World! 1A**

**Hello World! 1C**

## 4: テスト結果(検証)

②AutoScalingとCloudWatchAlarmが機能しているか確認のため、EC2インスタンスへストレスがけ

```
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '52.193.199.136' (ED25519) to the list of known hosts.
```



```
#_
##### Amazon Linux 2023
#####|
####\###| https://aws.amazon.com/linux/amazon-linux-2023
      \#/_
        V__ _>
          |
         /_
       _/_/
     _/_/_/
   _/_/_/_/
  _/_/_/_/
 _/_/_/_/
/_/_/_/_/
```

```
Last login: Wed Feb 26 05:31:40 2025 from 133.32.128.50
[ec2-user@ip-10-0-0-144 ~]$ wget https://rpmfind.net/linux/dag/redhat/el7/en/x86_64/dag/RPMS/stress-1.0.2-1.el7.rf.x86_64.rpm
--2025-02-26 08:50:33-- https://rpmfind.net/linux/dag/redhat/el7/en/x86_64/dag/RPMS/stress-1.0.2-1.el7.rf.x86_64.rpm
Resolving rpmfind.net (rpmfind.net)... 195.220.108.108
Connecting to rpmfind.net (rpmfind.net)|195.220.108.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 36804 (36K) [application/x-rpm]
Saving to: 'stress-1.0.2-1.el7.rf.x86_64.rpm'
```

```
stress-1.0.2-1.el7.rf.x86_64. 100%[=====] 35.94K 151KB/s in 0.2s

2025-02-26 08:50:34 (151 KB/s) - 'stress-1.0.2-1.el7.rf.x86_64.rpm' saved [36804/36804]
```

```
[ec2-user@ip-10-0-0-144 ~]$ sudo rpm -ivh stress-1.0.2-1.el7.rf.x86_64.rpm
warning: stress-1.0.2-1.el7.rf.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID 6b8d79e6: NOKEY
Verifying... ##### [100%]
Preparing... ##### [100%]
Updating / installing...
 1:stress-1.0.2-1.el7.rf ##### [100%]
[ec2-user@ip-10-0-0-144 ~]$ rpm -qa|grep stress
stress-1.0.2-1.el7.rf.x86_64
[ec2-user@ip-10-0-0-144 ~]$ stress -c 1 -q &
[1] 8249
```

# 4: テスト結果 (検証)

## ③ CloudWatchAlarmで70%を超えていること確認



## 4: テスト結果 (検証)

④ 負荷掛け後にインスタンスが各AZ1つずつから2つずつにスケールアウトしていることを確認



インスタンス (4)

🔍 インスタンスをフィルタリングする

<input type="checkbox"/>	インスタンス ID	ライフサイクル	インスタンスタ...	割り付け容量	起動テンプレー...	アベイラビリテ...	ヘルスステータス	以下のものからの...
<input type="checkbox"/>	<a href="#">i-01234567890123456</a>	InService	t2.micro	-	<a href="#">kadal4-template</a>	ap-northeast-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-01234567890123457</a>	InService	t2.micro	-	<a href="#">kadal4-template</a>	ap-northeast-1c	Healthy	
<input type="checkbox"/>	<a href="#">i-01234567890123458</a>	InService	t2.micro	-	<a href="#">kadal4-template</a>	ap-northeast-1a	Healthy	
<input type="checkbox"/>	<a href="#">i-01234567890123459</a>	InService	t2.micro	-	<a href="#">kadal4-template</a>	ap-northeast-1c	Healthy	

ライフサイクルフック (0) [info](#)

🔄 アクション ▼ [ライフサイクルフックを作成する](#)

# 4:テスト結果(検証)

## ⑤負荷掛けコマンド終了後CPU使用率が20%を下回るのを確認



## 4: テスト結果 (検証)

⑥ CPU使用率が20%を下回ったあとインスタンスが各AZ1つずつに戻っていることを確認

詳細 | 統合 - 新規 | オートスケーリング | インスタンス管理 | インスタンスの更新 | アクティビティ | モニタリング

---

インスタンス (2) 🔄 アクション ▼

🔍 インスタンスをフィルタリングする

<input type="checkbox"/>	インスタンス ID ▲	ライフサイクル ▼	インスタンスタ... ▼	読み付き容量 ▼	起動テンプレー... ▼	アベイラビリテ... ▼	ヘルスステータス ▼	以下のものからの... ▼
<input type="checkbox"/>	██████████ <a href="#">🔗</a>	InService	t2.micro	-	<a href="#">kda4-template</a> <a href="#">🔗</a> / <a href="#">バ...</a>	ap-northeast-1a	🟢 Healthy	
<input type="checkbox"/>	██████████ <a href="#">🔗</a>	InService	t2.micro	-	<a href="#">kda4-template</a> <a href="#">🔗</a> / <a href="#">バ...</a>	ap-northeast-1c	🟢 Healthy	

---

ライフサイクルフック (0) [info](#) 🔄 アクション ▼ ライフサイクルフックを作成する

# 4:テスト結果(検証)

⑦スケールアウト、インのアクションをした内容をSNSからメール通知されていることを確認

The screenshot displays the AWS CloudWatch console interface, specifically the 'Notifications' section. It shows a list of notifications received from AWS SNS, detailing scaling actions performed on EC2 instances.

**Notification List:**

件名	送信者	日時	サイズ
ALARM: "ScaleOutAlarm-CF" in Asia Pacific (Tokyo)	AWS Notifications	09:04:36	5.46K
ALARM: "ScaleOutAlarm-CF" in Asia Pacific (Tokyo)	AWS Notifications	08:51:36	5.46K
ALARM: "ScaleInAlarm" in Asia Pacific (Tokyo)	AWS Notifications	08:44:53	5.46K
ALARM: "ScaleInAlarm" in Asia Pacific (Tokyo)	AWS Notifications	25/02/26	5.46K
ALARM: "ScaleOutAlarm-CF" in Asia Pacific (Tokyo)	AWS Notifications	25/02/26	5.46K
ALARM: "ScaleInAlarm" in Asia Pacific (Tokyo)	AWS Notifications	25/02/26	5.46K
[Redacted]		25/02/26	15.13K
[Redacted]		25/02/26	15.15K
[Redacted]		25/02/26	31.89K
[Redacted]		25/02/26	15.15K

**Notification Details:**

件名: ALARM: "ScaleOutAlarm-CF" in Asia Pacific (Tokyo)  
送信者: "AWS Notifications" <no-reply@sns.amazonaws.com>

JSON Message Body:

```
{
  "Type": "Notification",
  "MessageId": "26708-43-54-465",
  "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:ScaleOutAlarm-CF",
  "Subject": "ALARM: \"ScaleOutAlarm-CF\" in Asia Pacific (Tokyo)",
  "Message": "26708-43-54-465 (70.0) (minimum 1:584791980123)"
}
```