

テスト(課題)

キム テブン

jkdnekki@naver.com

目次

■ 環境構築

- ・ 環境構築について

■ 課題

- ・ TEST 1. どちらか一方の EC2 インスタンスを Stopし、その後 Start
- ・ TEST 2. 障害原因の調査及び回避対策

□ その他(参考)

- ・ Cloud Formation template作成
- ・ AWS CLIの設定

環境構築について

下記のテンプレートを使うためには**事前準備**が必要となります。

<https://s3-ap-northeast-1.amazonaws.com/cm-ope-exam-prod/CM-Ope-CFnTemplates.yaml?versionId=null>

- **VPC Subnet**はデフォルトでも可能ですが、勉強を含めてテンプレート作成し、それを使ってみました。最後の【その他】のページに載せましたのでご参考ください。

詳細の指定

スタック名とパラメータ値を指定します。AWS CloudFormation テンプレートに定義づけられるデフォルトのパラメータ値を使用、または変更することができます。 [詳細はこちら](#)。

スタックの名前

パラメータ

InstanceType	t2.micro	WebServer EC2 instance type
KeyName	検索	Name of an existing EC2 KeyPair to enable SSH access to the instances
SSHLocation	0.0.0.0/0	The IP address range that can be used to SSH to the EC2 instances
Subnets	ID または Name タグ値で検索	List of two or more SubnetIds existing in different AZs in the Virtual Private Cloud
VpcId	ID または Name タグ値で検索	VpcId of your existing Virtual Private Cloud (VPC)

事前準備

•キーペア

ネットワーク&セキュリティ→キーペア

•AWS CLI Installと AWS Configureの設定

•VPCとsubnetを作成(その他のページ参考、必須X)

•JQコマンドインストール(その他のページ参考、必須X)

■ 課題

- TEST 1. どちらか一方の EC2 インスタンスを Stopし、その後 Start
- TEST 2. 障害原因の調査及び回避対策

TEST 1. どちらか一方の EC2 インスタンスを Stop し、その後 Start

一方のEC2インスタンスをstopする前に、Auto Scalingグループに設定されている インスタンスIDを検索します。
以下のようにELBに繋がれているインスタンスIDの確認ができます。

```
$ aws autoscaling describe-auto-scaling-instances --query 'AutoScalingInstances[].InstanceId'
[
  "i-059e6a5fcc14c5e3b",
  "i-06bff9f37aa74858f"
]
```

事前にインストールしたJQ で、インスタンスのHealth Statusをチェックすると正常に動いていることが確認できます。

```
$ aws autoscaling describe-auto-scaling-instances | jq '.AutoScalingInstances[] | [{InstanceId, HealthStatus}]'
{
  "InstanceId": "i-06bff9f37aa74858f",
  "HealthStatus": "HEALTHY"
}
{
  "InstanceId": " i-059e6a5fcc14c5e3b ",
  "HealthStatus": "HEALTHY"
}
```

* 赤色のインスタンスを以下のAWS CLIコマンドで停止させます。

```
$ aws ec2 stop-instances --instance-ids i-059e6a5fcc14c5e3b
```

← インスタンスを Stop!

Stop後、Auto Scalingをチェックしてみると、stop前に**Healthy**だったステータスがstop後に**Unhealthy**になることを確認できます。

```
$ aws autoscaling describe-auto-scaling-groups
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "AutoScalingGroupARN": "arn:aws:autoscaling:ap-northeast-1:115757856347:autoScalingGroup:228ed108-8589-4ccc-bfaa-610dd904ec76:autoScalingGroupName/ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
      "MinSize": 2,
      "MaxSize": 2,
      "DesiredCapacity": 2,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "ap-northeast-1a",
        "ap-northeast-1c"
      ],
      "LoadBalancerNames": [],
      "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:ap-northeast-1:115757856347:targetgroup/ELBSt-ALBTa-1D97IKGWMHXXO/f8f8e03f02671681"
      ],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 0,
      "Instances": [
        {
          "InstanceId": "i-059e6a5fcc14c5e3b",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "ap-northeast-1a",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
          "ProtectedFromScaleIn": false
        },
        {
          "InstanceId": "i-06bff9f37aa74858f",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "ap-northeast-1c",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
          "ProtectedFromScaleIn": false
        }
      ]
    }
  ],
  "省略～
```

```
$aws autoscaling describe-auto-scaling-groups
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "AutoScalingGroupARN": "arn:aws:autoscaling:ap-northeast-1:115757856347:autoScalingGroup:228ed108-8589-4ccc-bfaa-610dd904ec76:autoScalingGroupName/ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
      "MinSize": 2,
      "MaxSize": 2,
      "DesiredCapacity": 2,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "ap-northeast-1a",
        "ap-northeast-1c"
      ],
      "LoadBalancerNames": [],
      "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:ap-northeast-1:115757856347:targetgroup/ELBSt-ALBTa-1D97IKGWMHXXO/f8f8e03f02671681"
      ],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 0,
      "Instances": [
        {
          "InstanceId": "i-059e6a5fcc14c5e3b",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "ap-northeast-1a",
          "LifecycleState": "Terminating",
          "HealthStatus": "Unhealthy",
          "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
          "ProtectedFromScaleIn": false
        },
        {
          "InstanceId": "i-06bff9f37aa74858f",
          "InstanceType": "t2.micro",
          "AvailabilityZone": "ap-northeast-1c",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
          "ProtectedFromScaleIn": false
        }
      ]
    }
  ],
  "省略～
```

***原因:**最少1つのインスタスが必要なところが、2以下になった場合、AutoScalingが**Unhealthy**と判断します。
MinSizeの設定が**2**になっているので停止させたインスタスは**Terminate**され、新しいインスタスが生成されます。

AWS CLIでAuto Scalingグループの状況確認すると
以下のようにELBに新しいインスタンス(緑色)が作成されていることが確認できます。

```
$ aws autoscaling describe-auto-scaling-instances --query 'AutoScalingInstances[].InstanceId'
[
  "i-059e6a5fcc14c5e3b",
  "i-06bff9f37aa74858f",
  "i-0bbdc09f9a55c37de"
]
```

*Stopになっているインスタンスを再びstartしてもAutoScalingにより削除されているので、以下のようにインスタンスが存在しないというメッセージが出ます。

```
$ aws ec2 start-instances --instance-ids i-059e6a5fcc14c5e3b
```

```
An error occurred (InvalidInstanceID.NotFound) when calling the StartInstances
operation: The instance ID 'i-059e6a5fcc14c5e3b' does not exist
```

← インスタンスを Start!

*参考: stopコマンドを叩いた直後に(stopping中)startコマンドを叩くと「pending→InService」になって一時的にInServiceの状態になりますが、少し時間がたつとAutoScalingより削除されます。

既存のインスタンス(i-059e6a5fcc14c5e3b)が削除されても、ELB経由で問題なく接続は出来るのが確認できました。

```
$ aws elbv2 describe-load-balancers --query "LoadBalancers[][LoadBalancerName,DNSName,LoadBalancerArn]"
[
  [
    "ELBSt-Appli-7Q2M2W6ZVQ9U",
    "ELBSt-Appli-7Q2M2W6ZVQ9U-1141534294.ap-northeast-1.elb.amazonaws.com",
    "arn:aws:elasticloadbalancing:ap-northeast-1:115757856347:loadbalancer/app/ELBSt-Appli-7Q2M2W6ZVQ9U/ba5e5816edcbfe52"
  ]
]

$ curl ELBSt-Appli-7Q2M2W6ZVQ9U-1141534294.ap-northeast-1.elb.amazonaws.com
<h3>You have successfully launched the Classmethod Operation Team AWS CloudFormation sample</h3>
```

再びAWS CLIでAuto Scalingグループの状況再確認すると
以下のようにHealth statusが正常になっていることを確認できました。

```
$ aws autoscaling describe-auto-scaling-instances | jq '.AutoScalingInstances[] | [{InstanceId, HealthStatus}]'
{
  "InstanceId": "i-06bff9f37aa74858f",
  "HealthStatus": "HEALTHY"
}
{
  "InstanceId": "i-0bbdc09f9a55c37de",
  "HealthStatus": "HEALTHY"
}
```

新しく作成されたインスタンス

参考: 次のコマンドでAutoScalingのactivityを確認できます。

```
$ aws autoscaling describe-scaling-activities
```

他に試したこと(参考):

HealthCheckType をELBに変更し、一つのインスタンスはをstop後start

-502 Bad Gatewayエラーが表示され、正常に表示されるまで時間が少し掛かることを分かりました。HealthCheckTypeがEC2になっている場合は502エラーが出ずにELB経由でテストページが正常に表示出来ました。

```
aws autoscaling describe-auto-scaling-groups
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "AutoScalingGroupARN": "arn:aws:autoscaling:ap-northeast-1:115757856:ouputName/ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "LaunchConfigurationName": "ELBStack0119-LaunchConfig-1X98XPEQA0LRM",
      "MinSize": 2,
      "MaxSize": 2,
      "DesiredCapacity": 2,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "ap-northeast-1a",
        "ap-northeast-1c"
      ],
      "LoadBalancerNames": [],
      "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:ap-northeast-1:115757856347:target"
      ],
      "HealthCheckType": "ELB"
    }
  ]
}
```

```
$ curl ELBSt-Appli-7Q2M2W6ZVQ9U-1141534294.ap-northe
<html>
<head><title>502 Bad Gateway</title></head>
<body bgcolor="white">
<center><h1>502 Bad Gateway</h1></center>
</body>
</html>
```


TEST 2. 障害原因の調査及び回避対策

【概要】

ELB → EC2 (AutoScaling使用) → RDSの構成で、突然ページが表示できなくなりました。
(現在のところ、エラーレスポンスは不明)

1.考えられる障害原因

- ・ ELB のscale out等によるELB IPの変更で、EC2(ウェブサーバー)のApache HTTPDの応答遅延が発生する場合

サーバー台数を増やしてシステムの処理能力を高めるのは**Scale Out**とします。

* 参考:サーバーのCPU等のスペックを高めるのはScale Up

Scale Outをすると(EC2インスタンスの増加)**ELBのIPが変わる**ので、
その場合**EC2(Apache HTTPDの場合)**側からの**応答遅延**が発生する可能性があります。

【Apacheの仕様による応答遅延】

Apache HTTPD起動の時に自分が使う**IPを収集**します。

最初、正しい**IPをELBから**もらい、ちゃんと応答しますが、**ELBのIP変わると**最初もらったIPとことなるので

アクセス失敗になります。

Apache HTTPDには**3回アクセス失敗した時に初期化する機能**があるので、**最初もらったIPを連動の対象から除外し**,変わったELBのIPを再び収集します。その結果、正常に戻るまで時間がかかります。

そのタイミングで接続すると障害が発生し、ページ表示出来なくなる可能性があります。

- ・ 参考:ELBのIP確認は以下のコマンドで確認できます。

```
$ aws ec2 describe-network-interfaces --filters "Name=description,Values=ELB app/ELBSt-Appli-7Q2M2W6ZVQ9U/ba5e5816edcbfe52" | jq -r '.NetworkInterfaces[] | .Association.PublicIp' | sort
52.199.6.222
54.248.53.71
```

TEST 2. 障害原因の調査及び回避対策

1.考えられる障害原因

- ・ ELBの設定でヘルスチェックタイプがELBに設定され、インスタンスがInService以外になった場合

AutoScalingのヘルスチェックタイプが**ELBタイプ**になっている場合、再起動された**インスタンス**がAutoScalingヘルスチェックにより**NG**と判断され、そのインスタンスは**Terminate**されてELBから外されます。AutoScalingから新しいインスタンスが作られ、ELBが正常に戻る前に少し時間がかかります。そのタイミングでページに接続するとページが表示出来なくなることがあります。

- ・ SSL証明による接続問題の可能性

ELB経由でHTTPS接続出来るようにする時にはセキュリティーの為、**SSL証明**を使います。その**SSL証明書**の**期間切れ**もしくは**インストールされてない**時にELBへの**接続が出来なくなり**ページ表示が出来なくなる可能性も考えられます。

- ・ インスタンスの最大許容接続制限の問題

インスタンスで実行中のapache web serverがある場合、**httpd.conf**にある**MaxClient**、**ServerLimit**の非効率な設定で、メモリ負荷になり、応答が出来なくなることがあります。その時にページ表示が出来なくなる問題が発生する可能性があります。

・ 参考:以下のようにhttpd.confの設定を確認できます。

```
/etc/httpd/conf
[root@ip-10-1-10-21 conf]# vi httpd.conf

# prefork MPM
# StartServers: number of server processes to
start
# MinSpareServers: minimum number of server
processes which are kept spare
# MaxSpareServers: maximum number of server
processes which are kept spare
# ServerLimit: maximum value for MaxClients for
the lifetime of the server
# MaxClients: maximum number of server
processes allowed to start
# MaxRequestsPerChild: maximum number of
requests a server process serves
<IfModule prefork.c>
StartServers      8
MinSpareServers   5
MaxSpareServers   20
ServerLimit       256
MaxClients        256
MaxRequestsPerChild 4000
</IfModule>
```

参考:

- MaxClient:同時接続数、最大子プロセスの数になります。
 - デフォルトは256です。
 - Topコマンドを叩いてRES(物理メモリ使用率)を確認し、
サーバー物理メモリ量がhttpdプロセスのメモリ使用率 **X** MaxClient
+他のプロセスのメモリ使用率より**大きくなるようにMaxClientを設定**します。
- ServerLimit:設定可能なサーバープロセス数
 - MaxClientが256以上設定しない場合ServerLimitも設定する必要があります。
- StartServers:Apache起動の時の子プロセスの数
- MinspareServers:待機する子プロセスの数
 - 子プロセスの数がこの設定値より少なくなった場合、その値まで子プロセスの数を増加させます。MaxSpareServerはその反対です。
- MaxRequestPerChild:一個の子プロセスが処理するリクエストの数です。

TEST 2. 障害原因の調査及び回避対策

1. 考えられる障害原因

- ・パブリックサブネットのルートテーブルの設定ミスの可能性

ルートテーブルの0.0.0.0/0の宛先がIGWになってない場合、外部からELB経由でインスタンスに接続出来なくなります。

- ・セキュリティグループ設定の問題

セキュリティグループはインスタンスの仮想ファイアウォールとして機能をするので設定ミスをするとなんと通らないこともあります。

間違ってセキュリティグループを変更するとELB経由でEC2に行けなくなり、ページが表示できなくなる可能性もあります。

- ・参考:セキュリティグループにルールを追加する場合は以下のコマンド出来ます。

```
aws ec2 authorize-security-group-ingress --group-name GroupName1 --protocol tcp --port 22 --cidr 123.456.789.012/32
```

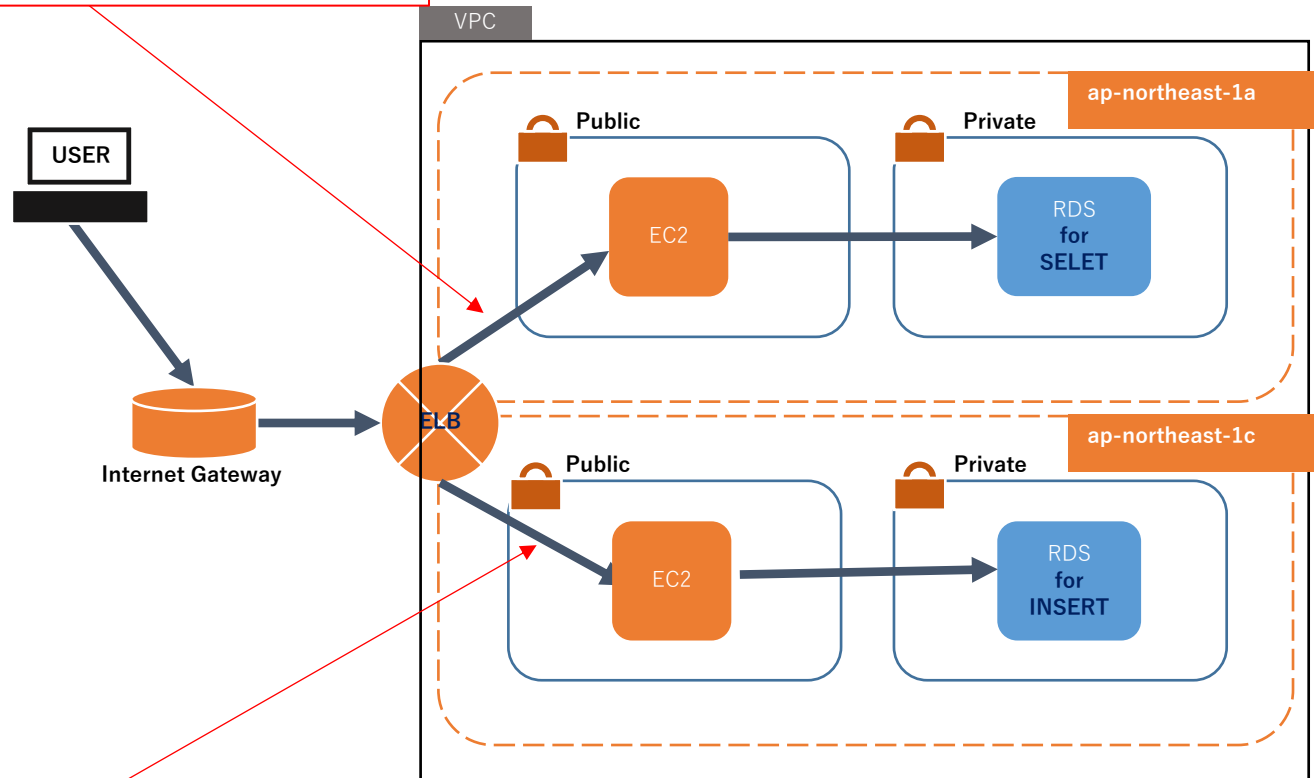
- ・間違ったELB構成によるトラブル

例:ユーザーのINSERTリクエストをELBが1CにあるDBにSAVEするようにして、次にユーザーがSELECTリクエストするとELBが1AのDBからそのリクエスト応答するような構成をした場合です。

当たり前だともいますが、共通で使えるRESOURCEがあり時に以下のように構成しないように気を付けないと問題が発生します。(RDS AZ構成をする時に勘違いして構築しないように注意する必要があります。)

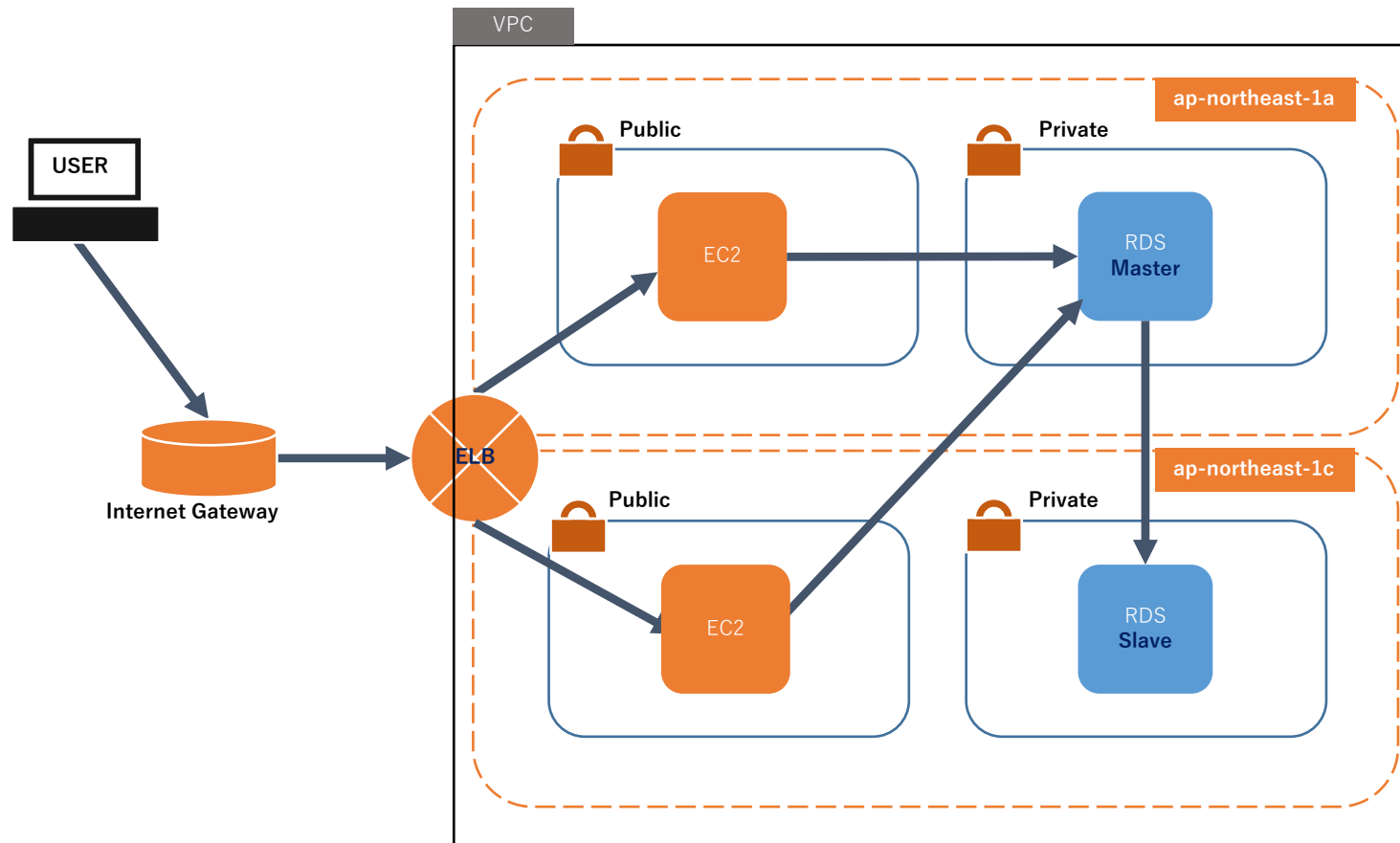
・ 間違ったELB構成の例)

②入力されているデータを見る時に
SELECTのrequestをする



①テキスト等を入力した後
INSERTのrequestをする

- 正しいELB構成の例)



TEST 2. 障害原因の調査及び回避対策

2. どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

まずは以下のようにローカルからaws cliコマンドでELBの状況を確認します。（コンソールでも確認可能です。次のページをご参考下さい。）

```
$ aws elbv2 describe-load-balancers
{
  "LoadBalancers": [
    {
      "LoadBalancerArn": "arn:aws:elasticloadbalancing:ap-northeast-1:115757856347:loadbalancer/app/class-Appli-P1AJBARDIPT4/2dfa90911ebe89fd",
      "DNSName": "class-Appli-P1AJBARDIPT4-1664726406.ap-northeast-1.elb.amazonaws.com",
      "CanonicalHostedZoneId": "Z14GRHDCWA56QT",
      "CreatedTime": "2019-12-29T02:57:30.240Z",
      "LoadBalancerName": "class-Appli-P1AJBARDIPT4",
      "Scheme": "internet-facing",
      "VpcId": "vpc-0c7d08c78089ea78a",
      "State": {
        "Code": "active"
      },
      "Type": "application",
      "AvailabilityZones": [
        {
          "ZoneName": "ap-northeast-1c",
          "SubnetId": "subnet-0640337f58b3338f3",
          "LoadBalancerAddresses": []
        },
        {
          "ZoneName": "ap-northeast-1a",
          "SubnetId": "subnet-09a4c5c825e429262",
          "LoadBalancerAddresses": []
        }
      ],
      "SecurityGroups": [
        "sg-043307d255bbff4c1"
      ],
      "IpAddressType": "ipv4"
    }
  ]
}
```

ELBが正常に動いているのかを確認します。

TEST 2. 障害原因の調査及び回避対策

2. どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

コンソールから確認はAuto Scaling Group→Activity Historyでstatusを確認します。



The screenshot displays the AWS Management Console interface for an Auto Scaling Group. The left sidebar shows the navigation menu with categories like 'イメージ', 'ELASTIC BLOCK STORE', 'ネットワーク & セキュリティ', 'ロードバランシング', and 'AUTO SCALING'. The main content area shows the 'Auto Scaling グループの作成' page for the group 'class-stack-WebServerGroup-PEUCLP1WDK7R'. Below the group details, the 'アクティビティ履歴' (Activity History) tab is selected, showing a table of activities. The table has columns for 'ステータス' (Status), '説明' (Description), '開始時刻' (Start Time), and '終了時刻' (End Time). All four activities listed are '成功' (Success).

ステータス	説明	開始時刻	終了時刻
成功	Launching a new EC2 instance: i-078e0542d1546d88f	2019 December 30 13:42:48 UTC+9	2019 December 30 13:43:21 UTC+9
成功	Terminating EC2 instance: i-060a5d6eeee9c23c2	2019 December 30 13:42:16 UTC+9	2019 December 30 13:48:08 UTC+9
成功	Launching a new EC2 instance: i-060a5d6eeee9c23c2	2019 December 29 11:57:46 UTC+9	2019 December 29 11:58:18 UTC+9
成功	Launching a new EC2 instance: i-04362371427f02348	2019 December 29 11:57:46 UTC+9	2019 December 29 11:58:18 UTC+9

ステータスが失敗になった時にクリックするとどんな段階で失敗したのか把握出来ると思います。

TEST 2. 障害原因の調査及び回避対策

2.どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

AutoScalingのStatusCodeを確認し問題がないか、もしあったら原因は何なのか確を以下のように確認出来ます。

```
$ aws autoscaling describe-scaling-activities
{
  "Activities": [
    {
      "ActivityId": "3055bfcd-ae3d-daba-6d81-83fd2658d73a",
      "AutoScalingGroupName": "ELBStack0119-WebServerGroup-1ITTZ6D129R3L",
      "Description": "Launching a new EC2 instance: i-0716e93405c80cf7a",
      "Cause": "At 2020-01-29T22:28:28Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.",
      "StartTime": "2020-01-29T22:28:30.198Z",
      "EndTime": "2020-01-29T22:29:03Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{¥\"Subnet ID¥\":¥\"subnet-0120b27868913fc65¥\",¥\"Availability Zone¥\":¥\"ap-northeast-1a¥\"}"
    }
  ],
  "諸略～"
}
```

問題の原因はCauseで把握出来ます。

Statusで正常の有無を確認します。

TEST 2. 障害原因の調査及び回避対策

2. どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

AWS CLIでロードバランサーとインスタンスのステータス確認する時にDebugオプションを付けると
どんな段階でエラーが出たのか確認出来ます。

```
$ aws elbv2 describe-load-balancers --debug
2020-01-02 17:27:27,817 - MainThread - awscli.clidriver - DEBUG - CLI version: aws-cli/1.16.309 Python/3.7.0 Linux/3.10.0-
957.12.2.el7.x86_64 botocore/1.13.45
2020-01-02 17:27:27,818 - MainThread - awscli.clidriver - DEBUG - Arguments entered to CLI: ['elbv2', 'describe-load-balancers', '--
debug']
2020-01-02 17:27:27,818 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
add_scalar_parsers at 0x7f6cbe7e4510>
2020-01-02 17:27:27,818 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
register_uri_param_handler at 0x7f6cbf2fcae8>
2020-01-02 17:27:27,818 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
inject_assume_role_provider_cache at 0x7f6cbf2d7bf8>
2020-01-02 17:27:27,821 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
attach_history_handler at 0x7f6cc0
```

```
$ aws ec2 describe-instances ¥
> --instance-ids i-078e0542d1546d88f --debug
2020-01-02 17:44:00,411 - MainThread - awscli.clidriver - DEBUG - CLI version: aws-cli/1.16.309 Python/3.7.0 Linux/3.10.0-
957.12.2.el7.x86_64 botocore/1.13.45
2020-01-02 17:44:00,411 - MainThread - awscli.clidriver - DEBUG - Arguments entered to CLI: ['ec2', 'describe-instances', '--instance-ids',
'i-078e0542d1546d88f', '--debug']
2020-01-02 17:44:00,411 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
add_scalar_parsers at 0x7f36c9400510>
2020-01-02 17:44:00,412 - MainThread - botocore.hooks - DEBUG - Event session-initialized: calling handler <function
register_uri_param_handler at 0x7f36c9f18ae8>
```

TEST 2. 障害原因の調査及び回避対策

2.どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

他に、Instance,ELBのセキュリティグループの設定が正しくなっているかを確認します。

```
$ aws ec2 describe-security-groups --group-ids sg-043307d255bbff4c1
{
  "SecurityGroups": [
    {
      "Description": "Enable HTTP access on the inbound port",
      "GroupName": "class-stack-ALBSecurityGroup-44VFRK8HCG9B",
      "IpPermissions": [
        {
          "FromPort": 80,
          "IpProtocol": "tcp",
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "Ipv6Ranges": [],
          "PrefixListIds": [],
          "ToPort": 80,
          "UserIdGroupPairs": []
        }
      ],
    }
  ],
  "NextToken": null
}
```

省略…

TEST 2. 障害原因の調査及び回避対策

2.どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

他に、インスタンスのステータス情報は以下のように確認できます。

```
aws ec2 describe-instances --instance-id instance_id
```

```
"StateReason": {  
  "Message": "Client.VolumeLimitExceeded: Volume limit  
exceeded",  
  "Code": "Server.InternalError"  
},
```

・参考:インスタンスの情報をscreenshotは以下のコマンドで作成出来ます。

```
aws ec2 get-console-screenshot instance_id --instance-id --query 'ImageData' --output text | base64 -d > ./screenShot.jpg
```

DBの状況は以下のコマンドで確認出来ます。

```
aws rds describe-db-instances --db-instance-identifier DB 識別子
```

・参考:DBストレージの問題があった場合以下のコマンドで解決出来ます。

```
aws rds modify-db-instance ¥  
  --db-instance-identifier DB 識別子 ¥  
  --allocated-storage 60 ¥  
  --apply-immediately
```

TEST 2. 障害原因の調査及び回避対策

2.どこを確認してどういう状態ならその箇所が原因であるといえそうなのか？

Linux インスタンスで Apache を実行している場合は、
sudo service httpd status コマンドを叩いて、Apache が実行中であることを確認します。。

```
[ec2-user@ip-10-1-10-83 ~]$ sudo service httpd status  
httpd (pid 3024) is running...
```

正常に動いている(running)のかを確認します。Stop場合apacheサーバーが問題がある場合がありますので
アクセスログを確認し原因を調査します。(トラブルシューティング)

インスタンスからELBへの接続テストをする為(設定したポート80で通るか確認)、以下のようにコマンドで確認できます。

```
[ec2-user@ip-10-1-10-83 ~]$ nc -v class-Appli-P1AJBARDIPT4-1664726406.ap-northeast-1.elb.amazonaws.com 80  
Connection to class-Appli-P1AJBARDIPT4-1664726406.ap-northeast-1.elb.amazonaws.com 80 port [tcp/http] succeeded!
```

接続テストの結果がsucceededになっていることを確認します。

上記のテスト等でトラブルがあったら、サーバー内のアクセスログの中身を開いて、問題の原因を調べます。

```
[root@ip-10-1-10-21 httpd]# vi access_log  
10.1.20.105 - - [29/Jan/2020:22:29:42 +0000] "GET / HTTP/1.1" 200 46 "-"  
"ELB-HealthChecker/2.0"
```

TEST 2. 障害原因の調査及び回避対策

3.障害をAWSで回避または影響を小さくするためにはどうすればいいのか？

- ・環境構築する時にAutoScaling設定や通信設定が適切なのかを確認します。
 - ヘルスチェックのタイプ等の設定を確認。
 - SSL証明書の期間を確認(割り当てられなかったら)、ポート設定(80, 443等)が正しいのかを確認。

ロードバランサー: `class-Appli-P1AJBARDIPT4`

説明

リスナー

モニタリング

統合サービス

タグ

リスナーは設定されたプロトコルおよびポートを使用して接続リクエストを確認し、ロードバランサーはリスナールールを使用してリ削除または更新することができます。

リスナーの追加

編集

削除

<input checked="" type="checkbox"/>	リスナー ID	セキュリティポリシー	SSL 証明書	ルール
<input checked="" type="checkbox"/>	HTTP : 80 arn...569d41dde95cfa54 ▼	該当なし	該当なし	デフォルト: 転送先 <code>class-ALBTa-1TP8H1J1LT58R</code> ルールの表示/編集

- ・環境構築をする時に「dry run」や「debug」 optionを使って事前に障害を見つけて対応したほうが良いかと思います。

- ・環境構築後cloudwatchからアラーム定期的にもらうようにします。
次のページにあるようにインスタンスの状態を定期的にメールに送信するようにすると障害を事前に回避出来る一つの手段として使えらと思います。

- ・参考:次のページにCloudWatch metricsの情報をメールでもらる設定(aws cliで)を一つの例として載せました。

TEST 2. 障害原因の調査及び回避対策

以下のようにAWS CLIでメールを送信を設定します。

```
$ aws sns create-topic --name test_topic2
{
  "TopicArn": "arn:aws:sns:ap-northeast-1:1157578xxxxx:test_topic2"
}
```

```
$ aws sns subscribe --topic-arn arn:aws:sns:ap-northeast-1:1157578xxxxx:test_topic2 ¥
> --protocol email ¥
➤ --notification-endpoint メールアドレス

{
  "SubscriptionArn": "pending confirmation"
}
```

Confirmメールが来るので、confirm subscriptionをクリックするとメール設定完了です。

AWS Notification - Subscription Confirmation 受信トレイ x

AWS Notifications no-reply@sns.amazonaws.com amazonses.com 経由

To 自分 ▼

🌐 英語 ▼ > 日本語 ▼ [メッセージを翻訳](#)

You have chosen to subscribe to the topic:

arn:aws:sns:ap-northeast-1:115757856347:test_topic2

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):

[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)



Simple Notification Service

Subscription confirmed!

You have subscribed kim.taehoon@nifty.co.jp to the topic:
test_topic2.

Your subscription's id is:
arn:aws:sns:ap-northeast-1:115757856347:test_topic2:99bcf22d-dc85-4a69-ae8e-1d3d2232bdfc

If it was not your intention to subscribe, [click here to unsubscribe.](#)

TEST 2. 障害原因の調査及び回避対策

以下のようにmetrics情報を得るスクリプトを作成して、その結果をSNSを使ってメール送信するようにするとメールでもインスタンスの色んなステータスを確認できます。cron 設定をすると定期的にメールでそのインスタンス情報を見ることが出来ます。

```
#!/bin/bash
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time 2020-01-15T00:00:00Z --end-time 2020-01-16T13:00:00Z --per 600 --namespace AWS/EC2
--statistics Average --dimensions Name=InstanceId,Value=i-07f9162916d000129 --query "sort_by(Datapoints,&Timestamp)[*]" >>log.txt

test=$(cat log.txt)
aws sns publish --topic-arn arn:aws:sns:ap-northeast-1:115757856347:test_topic --subject "log test mail" --message "$test"
```

log test mail 受信トレイ ×

AWS Notifications no-reply@sns.amazonaws.com amazonses.com 経由

To 自分 ▼

📄 英語 ▼ > 日本語 ▼ [メッセージを翻訳](#)

```
[
  {
    "Timestamp": "2020-01-05T09:20:00Z",
    "Average": 0.3005834954153931,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2020-01-05T09:30:00Z",
    "Average": 0.2997730851162359,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2020-01-05T09:40:00Z",
    "Average": 0.316685190330647,
    "Unit": "Percent"
  },
  {
    "Timestamp": "2020-01-05T09:50:00Z",
    "Average": 0.30030100953968686,
    "Unit": "Percent"
  },
  .
```


TEST 2. 障害原因の調査及び回避対策

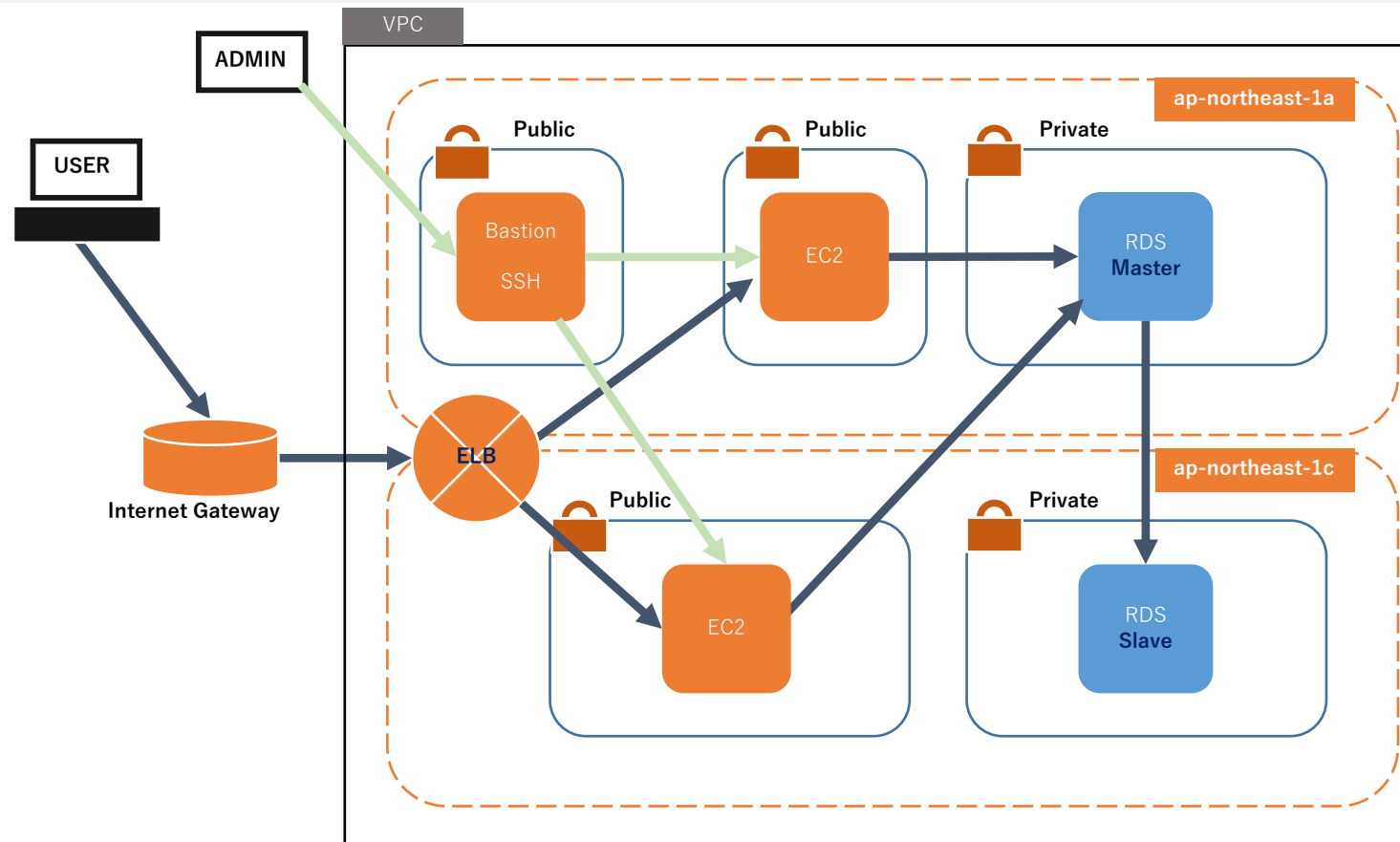
3. その障害をAWSで回避または影響を小さくするためにはどうすればいいのか？

最後に...

・ Bastion(踏み台)サーバーを立てて、障害がが発生したら踏み台サーバを経由でメンテナンスをおこない、メンテナンス中でもサービスを継続させます。

踏み台サーバ経由でメンテナンスすることで、障害影響をある程減らすことが出来ると思います。

踏み台サーバ以外も必要に応じて（private serverのex） yum updateなど） NATサーバーやNATゲートウェイも用意した方がいいと思います。



□ その他(参考)

- Cloud Formation template作成
- AWS CLIの設定

VPC 作成-ClassFormation template

VPCは下記の表のようになります。

アベイラビリティーゾーン			
ap-northeast-1a		ap-northeast-1c	
Name	IPv4 CIDR	Name	IPv4 CIDR
test-public-a	10.1.10.0/24	test-public-c	10.1.20.0/24
test-private-a	10.1.30.0/24	test-private-c	10.1.40.0/24

テンプレート 作成手順

設定ファイル(yml)の作成します。

テンプレートの以下ようになります。

①設定の宣言

```
AWSTemplateFormatVersion: "2010-09-09"
Description:
  VPC and Subnet Create
```

テンプレート 作成手順

パラメータを以下のように作成。

```
Metadata:
  "AWS::CloudFormation::Interface":
    ParameterGroups:
      - Label:
          default: "VPC Name "
        Parameters:
          - VPCName
      - Label:
          default: "Create VPC, Subnet"
        Parameters:
          - VPCCIDR
          - PublicSubnetACIDR
          - PublicSubnetCCIDR
          - PrivateSubnetACIDR
          - PrivateSubnetCCIDR
    ParameterLabels:
      VPCCIDR:
        default: "VPC CIDR Block"
      PublicSubnetACIDR:
        default: "PublicSubnetA CIDR Block"
      PublicSubnetCCIDR:
        default: "PublicSubnetC CIDR Block"
      PrivateSubnetACIDR:
        default: "PrivateSubnetA CIDR Block"
      PrivateSubnetCCIDR:
        default: "PrivateSubnetC CIDR Block"
```

パラメータのデフォルト値を作成。

```
Parameters:
  VPCName:
    Type: String
  VPCCIDR:
    Type: String
    Default: "10.1.0.0/16"
  PublicSubnetACIDR:
    Type: String
    Default: "10.1.10.0/24"
  PublicSubnetCCIDR:
    Type: String
    Default: "10.1.20.0/24"
  PrivateSubnetACIDR:
    Type: String
    Default: "10.1.30.0/24"
  PrivateSubnetCCIDR:
    Type: String
    Default: "10.1.40.0/24"
Resources:
```

VPC、インターネットゲートウェイ作成及びattach設定

```
VPC:
  Type: "AWS::EC2::VPC"
  Properties:
    CidrBlock: !Ref VPCCIDR
    EnableDnsSupport: "true"
    EnableDnsHostnames: "true"
    InstanceTenancy: default
  Tags:
    - Key: Name
      Value: !Sub "${VPCName}-vpc"

# InternetGateway Create
InternetGateway:
  Type: "AWS::EC2::InternetGateway"
  Properties:
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-igw"

# IGW Attach
InternetGatewayAttachment:
  Type: "AWS::EC2::VPCGatewayAttachment"
  Properties:
    InternetGatewayId: !Ref InternetGateway
    VpcId: !Ref VPC
```

テンプレート 作成手順

②Subnet作成設定

```
PublicSubnetA:
  Type: "AWS::EC2::Subnet"
  Properties:
    AvailabilityZone: "ap-northeast-1a"
    CidrBlock: !Ref PublicSubnetACIDR
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-public-subnet-a"

# Create Public SubnetC
PublicSubnetC:
  Type: "AWS::EC2::Subnet"
  Properties:
    AvailabilityZone: "ap-northeast-1c"
    CidrBlock: !Ref PublicSubnetCCIDR
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-public-subnet-c"

# Private SubnetA Create
PrivateSubnetA:
  Type: "AWS::EC2::Subnet"
  Properties:
    AvailabilityZone: "ap-northeast-1a"
    CidrBlock: !Ref PrivateSubnetACIDR
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-private-subnet-a"

# Create Private SubnetC
PrivateSubnetC:
  Type: "AWS::EC2::Subnet"
  Properties:
    AvailabilityZone: "ap-northeast-1c"
    CidrBlock: !Ref PrivateSubnetCCIDR
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-private-subnet-c"
```

③RouteTable作成設定

```
PublicRouteTableA:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-public-route-a"

# Create Public RouteTableC
PublicRouteTableC:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-public-route-c"

# Create Private RouteTableA
PrivateRouteTableA:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-private-route-a"

# Create Private RouteTableC
PrivateRouteTableC:
  Type: "AWS::EC2::RouteTable"
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub "${VPCName}-private-route-c"
```

④Routingの設定

```
PublicRouteA:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId: !Ref PublicRouteTableA
    DestinationCidrBlock: "0.0.0.0/0"
    GatewayId: !Ref InternetGateway

# PublicRouteC Create
PublicRouteC:
  Type: "AWS::EC2::Route"
  Properties:
    RouteTableId: !Ref PublicRouteTableC
    DestinationCidrBlock: "0.0.0.0/0"
    GatewayId: !Ref InternetGateway

# -----#
# RouteTable Associate
# -----#
# PublicRouteTable Associate SubnetA
PublicSubnetARouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnetA
    RouteTableId: !Ref PublicRouteTableA

# PublicRouteTable Associate SubnetC
PublicSubnetCRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PublicSubnetC
    RouteTableId: !Ref PublicRouteTableC

# PrivateRouteTable Associate SubnetA
PrivateSubnetARouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnetA
    RouteTableId: !Ref PrivateRouteTableA

# PrivateRouteTable Associate SubnetC
PrivateSubnetCRouteTableAssociation:
  Type: "AWS::EC2::SubnetRouteTableAssociation"
  Properties:
    SubnetId: !Ref PrivateSubnetC
    RouteTableId: !Ref PrivateRouteTableC
```

⑥出力

```
Outputs:
# VPC
VPC:
  Value: !Ref VPC
  Export:
    Name: !Sub "${VPCName}-vpc"

VPCCIDR:
  Value: !Ref VPCCIDR
  Export:
    Name: !Sub "${VPCName}-vpc-cidr"

# Subnet
PublicSubnetA:
  Value: !Ref PublicSubnetA
  Export:
    Name: !Sub "${VPCName}-public-subnet-a"

PublicSubnetACIDR:
  Value: !Ref PublicSubnetACIDR
  Export:
    Name: !Sub "${VPCName}-public-subnet-a-cidr"

PublicSubnetC:
  Value: !Ref PublicSubnetC
  Export:
    Name: !Sub "${VPCName}-public-subnet-c"

PublicSubnetCCIDR:
  Value: !Ref PublicSubnetCCIDR
  Export:
    Name: !Sub "${VPCName}-public-subnet-c-cidr"

PrivateSubnetA:
  Value: !Ref PrivateSubnetA
  Export:
    Name: !Sub "${VPCName}-private-subnet-a"
```

```
PrivateSubnetACIDR:
  Value: !Ref PrivateSubnetACIDR
  Export:
    Name: !Sub "${VPCName}-private-subnet-a-cidr"

PrivateSubnetC:
  Value: !Ref PrivateSubnetC
  Export:
    Name: !Sub "${VPCName}-private-subnet-c"

PrivateSubnetCCIDR:
  Value: !Ref PrivateSubnetCCIDR
  Export:
    Name: !Sub "${VPCName}-private-subnet-c-cidr"

# Route
PublicRouteTableA:
  Value: !Ref PublicRouteTableA
  Export:
    Name: !Sub "${VPCName}-public-route-a"

PublicRouteTableC:
  Value: !Ref PublicRouteTableC
  Export:
    Name: !Sub "${VPCName}-public-route-c"

PrivateRouteTableA:
  Value: !Ref PrivateRouteTableA
  Export:
    Name: !Sub "${VPCName}-private-route-a"

PrivateRouteTableC:
  Value: !Ref PrivateRouteTableC
  Export:
    Name: !Sub "${VPCName}-private-route-c"
```

テンプレート 作成手順

CloudFormationで先程作成したVPC設定yamlファイル(テンプレート)を使ってVPC構築

スタックの名前

test

Stackの名前を入力

パラメータ

Project Name

ProjectName

任意の名前を入力

Network Configuration

VPC CIDR

10.1.0.0/16

PublicSubnetA CIDR

10.1.10.0/24

PublicSubnetC CIDR

10.1.20.0/24

PrivateSubnetA CIDR

10.1.30.0/24

PrivateSubnetC CIDR

10.1.40.0/24

VPCとsubnetの値はdefaultそのままでもOK

キャンセル

戻る

次へ

スタックの作成

アクション

テンプレートのデザイン

問題なく完了するとCREATE_COMPLETE表示されます。

7 個のスタックを表示

スタックの名前	作成時間	状況	ドリフトステータス	説明
<input type="checkbox"/> vpc-stack	2019-12-29 11:51:22 UTC+0900	CREATE_COMPLETE	NOT_CHECKED	VPC and Subnet Create

環境構築(本番)

以下のテンプレートを使って環境構築を進めます。

<https://s3-ap-northeast-1.amazonaws.com/cm-ope-exam-prod/CM-Ope-CFnTemplates.yaml?versionId=null>

CloudFormation ▾ スタック > スタックの作成

スタックの作成

テンプレートの選択

詳細の指定
オプション
確認

テンプレートの選択

作成したいスタックの内容が書かれたテンプレートを選択してください。スタックは、単一のユニットとして管理できる関連リソースのグループです。

テンプレートをデザインする

既存のテンプレートの作成や修正は、AWS CloudFormation Designer をご利用ください。詳細はこちら。
テンプレートのデザイン

テンプレートの選択

テンプレートは、スタックのリソースとそのプロパティを説明する、JSON/YAML 形式のテキストファイルです。詳細はこちら。

☐ サンプルテンプレートの選択

☐ テンプレートを Amazon S3 にアップロード

ファイルを選択 選択されていません

☒ Amazon S3 テンプレート URL の指定

https://s3-ap-northeast-1.amazonaws.com/cm-ope-exam-prod/CM-Ope-CFnTemplates.ya

デザイナーで表示

URLの入れます

キャンセル 次へ

環境構築(本番)

以下のように値を入力して環境構築をします。

スタックの作成

- テンプレートの選択
- 詳細の指定
- オプション
- 確認

詳細の指定

スタック名とパラメータ値を指定します。AWS CloudFormation テンプレートに定義づけられるデフォルトのパラメータ値を使用、または変更することができます。 [詳細はこちら](#)。

スタックの名前

Stackの任意の名前を入力

パラメータ

InstanceType

WebServer EC2 instance type

KeyName

Name of an existing EC2 KeyPair to enable SSH access to the instances

SSHLocation

The IP address range that can be used to SSH to the EC2 instances

Subnets

subnet-09a4c5c825e429262 (10.1.10.0/24) (test-public-a) ✕

subnet-0640337f58b3338f3 (10.1.20.0/24) (test-public-c) ✕

List of two or more SubnetIds existing in different AZs in the Virtual Private Cloud

VPC設定の時に作成したpublic subnetを入力

VpcId

VpcId of your existing Virtual Private Cloud (VPC)

作成したVPCを選択

キャンセル

戻る

次へ

AWS CLIの設定

下記のURLでAWS Access Key ID とAWS Secret Access Keyを作ります。

https://console.aws.amazon.com/iam/home#/users/kim2?section=security_credentials

上記のURLに入って「Identity and Access Management (IAM)」の設定をします。

1

Identity and Access Management (IAM)

ユーザーを追加

ユーザーの削除

ユーザー名またはアクセスキーIDでユーザーを検索

2件の結果を表示中

ユーザー名	グループ	アクセスキーの古さ	パスワードの古さ	最後のアクティビティ	MFA
<input type="checkbox"/> kim	なし	5日間	なし	5日間	有効でない
<input type="checkbox"/> kim2	full access	5日間	なし	4日間	有効でない

ユーザー→ユーザーを追加選択

2

ユーザーを追加

1 2 3 4 5

ユーザー詳細の設定

同じアクセスの種類とアクセス権限を使用し複数のユーザーを一度に追加できます。詳細はこちら

ユーザー名*

kim3

名前を入力後次へ

別のユーザーの追加

AWS アクセスの種類を選択

これらのユーザーから AWS にアクセスする方法を選択します。アクセスキーと自動生成パスワードは前のステップで提供されています。詳細はこちら

アクセスの種類*

☒ プログラムによるアクセス

AWS API、CLI、SDK などの開発ツールの **アクセスキー ID とシークレットアクセスキー** を有効にします。

☐ AWS マネジメントコンソールへのアクセス

ユーザーに AWS マネジメントコンソールへのサインインを許可するための **パスワード** を有効にします。

* 必須

キャンセル

次のステップ: アクセス権限

3

ユーザーを追加

1 2 3 4 5

アクセス許可の設定

ユーザーをグループに追加

アクセス権限を既存のユーザーからコピー

既存のポリシーを直接アタッチ

ユーザーを既存のグループに追加するか、新しいグループを作成します。ユーザーのアクセス権限は、グループを使ってジョブ機能別に管理するのが最善の方法です。詳細はこちら

ユーザーをグループに追加

グループの作成

更新

検索

2件の結果を表示中

グループ	アタッチされたポリシー
------	-------------

ユーザーグループに追加後次へ

AWS CLIの設定

4

グループの作成

グループを作成して、そのグループにアタッチするポリシーを選択します。ユーザーのアクセス権限は、グループを使ってジョブ機能、AWS リービスへのアクセス、カスタムのアクセス権限別に管理するのが最善の方法です。 [詳細はこちら](#)

グループ名

ポリシーの作成

ポリシーのフィルタ

1件の結果を表示中

	ポリシー名	タイプ	次として使用	説明
<input checked="" type="checkbox"/>	AmazonEC2FullAccess	AWS による管理	Permissions policy (2)	Provides full access to Amazon EC2 via the AWS Management Console.

EC2FullAccess権限を検索して選択後次へ

5

成功

以下に示すユーザーを正常に作成しました。ユーザーのセキュリティ認証情報を確認してダウンロードできます。AWS マネジメントコンソールへのサインイン手順を E メールでユーザーに送信することもできます。今回が、これらの認証情報をダウンロードできる最後の機会です。ただし、新しい認証情報はいつでも作成できます。

AWS マネジメントコンソールへのアクセス権を持つユーザーは「<https://kanshi-system-development.signin.aws.amazon.com/console>」でサインインできます

.csv のダウンロード

アクセスキーIDとシークレットキーをメモして次の設定せで使います

	ユーザー	アクセスキー ID	シークレットアクセスキー
▼	kim3	AKIARV45UWZN5K355ZD5	***** 表示

AWS CLIの設定

もし、権限に問題があったら、以下のようにユーザーメニューでアクセス権限を追加します。

Identity and Access Management (IAM)

ダッシュボード

▼ アクセス管理

グループ

ユーザー

ロール

ポリシー

ID プロバイダー

アカウント設定

▼ アクセスレポート

アクセスアナライザー

アーカイブルール

アナライザーの詳細

認証情報レポート

組織アクティビティ

サービスコントロールポリシー (SCP)

Q IAM の検索

AWS アカウント ID:

115757856347

作成時刻 2019-12-24 13:42 UTC+0900

アクセス権限 グループ (1) タグ 認証情報 アクセスアドバイザー

▼ Permissions policies (1 適用済みポリシー)

アクセス権限の追加

インラインポリシーの追加

ポリシー名 ▼	ポリシータイプ ▼
グループからアタッチ済み	
▼ AdministratorAccess	グループ full_access の AWS 管理ポリシー ✕

ポリシー概要 {} JSON

ポリシーのシミュレート

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "*",
7       "Resource": "*"
8     }
9   ]
10 }
```

▼ Permissions boundary (set)

この user が持つことができる最大のアクセス権限を制御するアクセス権限の境界を設定します。これは、一般的な設定ではありませんが、アクセス権限の管理を他社に委任するために使用できます。 [詳細はこちら](#)

境界の変更

境界の削除

AWS CLIの設定

6

ローカル環境でAWS CLIを設定

```
$ aws configure
AWS Access Key ID [*****TNUQ]:
AWS Secret Access Key [*****yKXL]:
Default region name [ap-northeast-1]:
Default output format [json]:
```

IAMでメモしたキーIDとアクセスキーを入力、regionはap-northeast-1にoutput形式はjsonにします。

instance確認

```
$ aws ec2 describe-tags --query 'Tags[?ResourceType==`instance`] | [?Key==`Name`].{ResourceId:ResourceId,ResourceName:Value}'
--output table
```

動作確認

```
$ aws ec2 start-instances --instance-ids i-0xxxxxxxxxxxxx
{
  "StartingInstances": [
    {
      "CurrentState": {
        "Code": 16,
        "Name": "running"
      },
      "InstanceId": "i-0xxxxxxxxxxxxx",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

インスタンスIDのみ取得

```
$ aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
"i-0f..."
"i-04..."
"i-0c..."
"i-0c..."
"i-0k..."
```

jqコマンドのインストール

```
$ sudo yum install jq
```

AWS CLIの設定 その他

jqコマンドのインストール

```
$ sudo yum install jq
```

jqコマンドは以下のように活用出来ます。

インスタンスIDのみ取得

```
$ aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
"i-0f..."
"i-04..."
"i-0c..."
"i-0c..."
"i-0k..."
```

インスタンスID、インスタンスタイプ、publicIPを取得

```
$ aws ec2 describe-instances | jq '.Reservations[].Instances[] | {InstanceId, InstanceType, PublicIpAddress}'
```

インスタンスID、インスタンスタイプ、privateIPを取得

```
$ aws ec2 describe-instances | jq '.Reservations[].Instances[] | {InstanceId, InstanceType, PrivateIpAddress}'
```