eBook Gratuit

APPRENEZ amazon-web-services

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#amazon-web-services

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: amazon-web-services

It is an unofficial and free amazon-web-services ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official amazon-web-services.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec amazon-web-services

Remarques

Cette section fournit une vue d'ensemble de ce qu'est amazon-web-services et pourquoi un développeur peut vouloir l'utiliser.

Il convient également de mentionner tous les grands sujets au sein des services Web amazoniens et d'établir un lien avec les sujets connexes. La documentation de amazon-web-services étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
1.0.0	2017-01-10

Examples

Avant qu'il ne soit trop tard

Trucs et astuces pour éviter les situations désagréables

EC2 Instances et EBS

- Définissez les rôles IAM.

Contrairement aux tags, le rôle IAM est **défini une fois pour toutes sur l'instanciation EC2** (même après 4 ans)! Essayez d'identifier et de classer au préalable vos instances afin de pouvoir leur attribuer des rôles IAM appropriés. Les rôles IAM sont un bon moyen d'identifier vos machines. Cela permettra à amazon de stocker automatiquement les informations d'identification du profil d'instance sur vos machines, et vous pourrez facilement accorder des privilèges supplémentaires.

Considérez la situation suivante où vous avez des serveurs de base de données et vous réalisez que vous souhaitez surveiller l'utilisation de la mémoire / disque. Amazon CloudWatch ne fournit pas cette métrique prête à l'emploi et vous devrez configurer des privilèges supplémentaires pour envoyer des données personnalisées à CloudWatch. Si vous avez un rôle IAM "Database", vous pouvez facilement associer de nouvelles stratégies à vos instances de base de données existantes pour leur permettre d'envoyer des rapports de mémoire à CloudWatch. Pas de rôles IAM? Vous devez recréer vos instances de base de données ou leur donner une autorisation individuelle.

- Attention à l'intégrité des instantanés

Amazon vous permet de capturer des volumes EBS, mais si vous utilisez plusieurs volumes sur la même machine (en configuration RAID, plusieurs volumes EBS pour votre base de données), il est impossible de garantir l'intégrité de ces instantanés, ce qui peut se produire différents volumes EBS.

Assurez-vous toujours qu'aucune donnée n'est écrite (arrêtez la machine virtuelle ou utilisez le code spécifique à l'application (par exemple, `db.fsyncLock()`) pour vous assurer qu'aucune donnée n'est écrite pendant l'instantané.

CloudWatch

- Utilisez les alertes Amazon Cloudwatch + SNS en plus des notifications d'erreur de l'application

Créez des alertes pour le comportement normal de vos machines et configurez-les pour envoyer des notifications via Amazon SNS (par exemple, des adresses e-mail) en cas de problème. Avoir des notificateurs d'exception sur votre application ne vous aidera pas si votre serveur ne peut même pas être ping. D'un autre côté, hormis 500 codes d'erreur, Amazon dispose de peu d'informations sur votre application et sur la manière dont elle est censée fonctionner. Vous devriez envisager d'ajouter une surveillance de la santé spécifique à l'application.

Lire Démarrer avec amazon-web-services en ligne: https://riptutorial.com/fr/amazon-web-services/topic/798/demarrer-avec-amazon-web-services

Chapitre 2: Amazon Cognito

Examples

Gestion des identités utilisateur à l'aide d'Amazon Cognito

```
var app = {};

app.signUp = function(){

app.userName        =           $('#userName').val();
app.password        =           $('#password').val();
app.email           =           $('#form-email').val();
app.phoneNumber     =           $('#form-phone').val();
app.emailRegex      =
/^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\

/*
    Put the User input validation logic here.
*/
if (!app.userName) {
    alert("Please provide a user name");
    return;
}

if (!app.password) {
    alert("Please provide a password");
    return;
}

if (!app.email) {
    alert("Please provide an Email address");
    return;
}

if(!app.emailRegex.test(app.email)){
   alert("Please provide a valid Email address");
   return;
}

if (!app.phoneNumber) {
    alert("Please provide a Phone Number");
    return;
}

AWS.config.region = 'us-east-1'; // Region
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
IdentityPoolId: '...' // your identity pool id here
});

AWSCognito.config.region = 'us-east-1';
AWSCognito.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: '...' // your identity pool id here
});

// Need to provide placeholder keys unless unauthorised user access is enabled for user pool
```

```
AWSCognito.config.update({accessKeyId: 'anything', secretAccessKey: 'anything'})

var poolData = {
    UserPoolId :     APP_CONSTANT.USER_POOL_ID,
    ClientId :     APP_CONSTANT.CLIENT_ID
};
userPool = new   AWSCognito.CognitoIdentityServiceProvider.CognitoUserPool(poolData);

var attributeList = [];

var dataEmail = {
    Name : 'email',
    Value : app.email //Email Id where the confirmation code would be sent.
};
var dataPhoneNumber = {
    Name : 'phone_number',
    Value : app.phoneNumber
};
var attributeEmail = new
AWSCognito.CognitoIdentityServiceProvider.CognitoUserAttribute(dataEmail);
//Uncomment below once the phone number format is confirmed
/*var attributePhoneNumber = new
AWSCognito.CognitoIdentityServiceProvider.CognitoUserAttribute(dataPhoneNumber);*/

attributeList.push(attributeEmail);
 /* attributeList.push(attributePhoneNumber);*/
// Put the user id and password collected from user below for signup
userPool.signUp(app.userName, app.password, attributeList, null, function(err, result){
    if (err) {
        alert(err);
        return;
    }
    cognitoUser = result.user;
// Return the user name once the user signed up, still need to confirm with confirmation code
send to mail.
$("#form-confirmCode").css("display", "block");
    alert('user name is ' + cognitoUser.getUsername());
// Now since the user is signed up and pending for confirmaion, disable all the pervious input
but confirmation code.
$("#userName").prop("readonly", true);
$("#password").prop("readonly", true);
$("#form-email").prop("readonly", true);
$("#form-phone").prop("readonly", true);
$("#signUpBtn").hide();
$("#confirm-block").show();

var confirmationCode = prompt("Hello "+cognitoUser.getUsername+" Enter the confirmation code
sent to your email address.","Confirmation code here");
cognitoUser.confirmRegistration(confirmationCode,true,function(err,result){
            if(err){
                    alert(err);
                    return;
            }
            console.log('Call Result: '+result);
    });
return;

});
};
```

Lire Amazon Cognito en ligne: https://riptutorial.com/fr/amazon-web-services/topic/3425/amazon-

cognito

Chapitre 3: Amazon DynamoDB

Examples

DynamoDB basic Opération Crud avec NodeJS

```
let doc = require('dynamodb-doc');
let dynamo = new doc.DynamoDB();
var tblName = "MyTable";

exports.handler = (event, context, callback) => {
    readOperation(context);
}

function readOperation(cnxt) {
    var params = {
        TableName: tblName,
        Key: {
            "id": "2013",
            "topic": "Turn It Down, Or Else!"
        },
        AttributesToGet: [
            "id", "client_name", "info"
        ],
        ConsistentRead: false
    };
    dynamo.getItem(params, function(err, data) {
        if (err) console.log("Error: "+err); // an error occurred
        else {
            var jsonDoc = JSON.parse(data.Item.info);   // successful response
            cnxt.succeed(jsonDoc);
        }
    });
}
```

Lire Amazon DynamoDB en ligne: https://riptutorial.com/fr/amazon-web-services/topic/6429/amazon-dynamodb

Chapitre 4: AWS CloudFormation

Examples

Exemple de script CloudFormation permettant de créer une instance EC2 avec un groupe de sécurité à associer.

Cet exemple crée une instance EC2 de **t2.micro** type dans la région **N.Virginia** exécutant **Amazon Linux.** Pendant l'exécution, il vous sera demandé de sélectionner le KeyPair à utiliser et un IP CIDR à partir duquel vous pouvez SSH sur l'instance, utilisez la valeur par défaut pour ouvrir SSH sur Internet.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template EC2InstanceWithSecurityGroupSample:
Create an Amazon EC2 instance running the Amazon Linux AMI. This example creates an EC2
security group for the instance to give you SSH access. ",

  "Parameters" : {
    "KeyName": {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription" : "must be the name of an existing EC2 KeyPair."
    },
    "SSHLocation" : {
      "Description" : "The IP address range that can be used to SSH to the EC2 instances",
      "Type": "String",
      "MinLength": "9",
      "MaxLength": "18",
      "Default": "0.0.0.0/0",
      "AllowedPattern": "(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})\\.(\\d{1,3})/(\\d{1,2})",
      "ConstraintDescription": "must be a valid IP CIDR range of the form x.x.x.x/x."
   }
  },

  "Resources" : {
    "EC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "InstanceType" : "t2.micro",
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "KeyName" : { "Ref" : "KeyName" },
        "ImageId" : "ami-6869aa05"
      }
    },

    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
```

```
          "ToPort" : "22",
          "CidrIp" : { "Ref" : "SSHLocation"}
        } ]
      }
    }
  },

  "Outputs" : {
    "InstanceId" : {
      "Description" : "InstanceId of the newly created EC2 instance",
      "Value" : { "Ref" : "EC2Instance" }
    },
    "AZ" : {
      "Description" : "Availability Zone of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "EC2Instance", "AvailabilityZone" ] }
    },
    "PublicDNS" : {
      "Description" : "Public DNSName of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicDnsName" ] }
    },
    "PublicIP" : {
      "Description" : "Public IP address of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "EC2Instance", "PublicIp" ] }
    }
  }
}
```

AWS CloudFormer dans VPC

Le modèle CloudFormer convertit les entités AWS existantes en un nouveau modèle CloudFormation pour accélérer le temps nécessaire pour recréer l'environnement. CloudFormer se lance dans un VPC par défaut qui pourrait ne pas convenir dans les cas où le VPC par défaut est supprimé. Cette base de code est un fork de CloudFormer original qui serait lancé dans un nouveau VPC.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "A custom CloudFormer template forked from AWS provided to extend the
capability to provide the ability to specify the VPC. Creates a Separate Stand-Alone VPC,
Subnet - 10.0.0.0/16 to launch the CloudFormer Instance. AWS CloudFormer Beta - template
creation prototype application. This tool allows you to create an AWS CloudFormation template
from the AWS resources in your AWS account. **Warning** This template creates a single EC2
instance in your account to run the application - you will be billed for the instance at
normal AWS EC2 rates.",

  "Parameters" : {

    "Username" : {
      "Description" : "Username to log in to CloudFormer",
      "Type" : "String"
    },
    "Password" : {
      "Description" : "Password to log in to CloudFormer",
      "Type" : "String",
      "NoEcho" : "true"
    },
    "CommonNameTag" : {
```

```
        "Description" : "Common Identifier / Friendly Name for the Stack",
        "Type" : "String",
        "Default" : "CloudFormer - Non Default VPC"
    }
  },

  "Mappings" : {
    "NetworkValues" : {
        "VPC" : {"CIDR" : "10.0.0.0/16"},
        "Subnet" : {"CIDR" : "10.0.10.0/16"}
    },

    "Region2Examples" : {
      "us-east-1"      : { "Examples" : "https://s3.amazonaws.com/cloudformation-examples-us-
east-1" },
      "us-west-2"      : { "Examples" : "https://s3-us-west-2.amazonaws.com/cloudformation-
examples-us-west-2" },
      "us-west-1"      : { "Examples" : "https://s3-us-west-1.amazonaws.com/cloudformation-
examples-us-west-1" },
      "eu-west-1"      : { "Examples" : "https://s3-eu-west-1.amazonaws.com/cloudformation-
examples-eu-west-1" },
      "eu-central-1"   : { "Examples" : "https://s3-eu-central-1.amazonaws.com/cloudformation-
examples-eu-central-1" },
      "ap-southeast-1" : { "Examples" : "https://s3-ap-southeast-
1.amazonaws.com/cloudformation-examples-ap-southeast-1" },
      "ap-northeast-1" : { "Examples" : "https://s3-ap-northeast-
1.amazonaws.com/cloudformation-examples-ap-northeast-1" },
      "ap-southeast-2" : { "Examples" : "https://s3-ap-southeast-
2.amazonaws.com/cloudformation-examples-ap-southeast-2" },
      "ap-northeast-2" : { "Examples" : "https://s3-ap-northeast-
2.amazonaws.com/cloudformation-examples-ap-northeast-2" },
      "sa-east-1"      : { "Examples" : "https://s3-sa-east-1.amazonaws.com/cloudformation-
examples-sa-east-1" },
      "cn-north-1"     : { "Examples" : "https://s3.cn-north-
1.amazonaws.com.cn/cloudformation-examples-cn-north-1" }
    },

    "Region2Principal" : {
      "us-east-1"      : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "us-west-2"      : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "us-west-1"      : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "eu-west-1"      : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "ap-southeast-1" : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "ap-northeast-1" : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "ap-southeast-2" : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "ap-northeast-2" : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "sa-east-1"      : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" },
      "cn-north-1"     : { "EC2Principal" : "ec2.amazonaws.com.cn", "OpsWorksPrincipal" :
"opsworks.amazonaws.com.cn" },
      "eu-central-1"   : { "EC2Principal" : "ec2.amazonaws.com", "OpsWorksPrincipal" :
"opsworks.amazonaws.com" }
    },
```

```
  "AWSInstanceType2Arch" : {
    "t1.micro"    : { "Arch" : "PV64"   },
    "t2.nano"     : { "Arch" : "HVM64"  },
    "t2.micro"    : { "Arch" : "HVM64"  },
    "t2.small"    : { "Arch" : "HVM64"  },
    "t2.medium"   : { "Arch" : "HVM64"  },
    "t2.large"    : { "Arch" : "HVM64"  },
    "m1.small"    : { "Arch" : "PV64"   },
    "m1.medium"   : { "Arch" : "PV64"   },
    "m1.large"    : { "Arch" : "PV64"   },
    "m1.xlarge"   : { "Arch" : "PV64"   },
    "m2.xlarge"   : { "Arch" : "PV64"   },
    "m2.2xlarge"  : { "Arch" : "PV64"   },
    "m2.4xlarge"  : { "Arch" : "PV64"   },
    "m3.medium"   : { "Arch" : "HVM64"  },
    "m3.large"    : { "Arch" : "HVM64"  },
    "m3.xlarge"   : { "Arch" : "HVM64"  },
    "m3.2xlarge"  : { "Arch" : "HVM64"  },
    "m4.large"    : { "Arch" : "HVM64"  },
    "m4.xlarge"   : { "Arch" : "HVM64"  },
    "m4.2xlarge"  : { "Arch" : "HVM64"  },
    "m4.4xlarge"  : { "Arch" : "HVM64"  },
    "m4.10xlarge" : { "Arch" : "HVM64"  },
    "c1.medium"   : { "Arch" : "PV64"   },
    "c1.xlarge"   : { "Arch" : "PV64"   },
    "c3.large"    : { "Arch" : "HVM64"  },
    "c3.xlarge"   : { "Arch" : "HVM64"  },
    "c3.2xlarge"  : { "Arch" : "HVM64"  },
    "c3.4xlarge"  : { "Arch" : "HVM64"  },
    "c3.8xlarge"  : { "Arch" : "HVM64"  },
    "c4.large"    : { "Arch" : "HVM64"  },
    "c4.xlarge"   : { "Arch" : "HVM64"  },
    "c4.2xlarge"  : { "Arch" : "HVM64"  },
    "c4.4xlarge"  : { "Arch" : "HVM64"  },
    "c4.8xlarge"  : { "Arch" : "HVM64"  },
    "g2.2xlarge"  : { "Arch" : "HVMG2"  },
    "g2.8xlarge"  : { "Arch" : "HVMG2"  },
    "r3.large"    : { "Arch" : "HVM64"  },
    "r3.xlarge"   : { "Arch" : "HVM64"  },
    "r3.2xlarge"  : { "Arch" : "HVM64"  },
    "r3.4xlarge"  : { "Arch" : "HVM64"  },
    "r3.8xlarge"  : { "Arch" : "HVM64"  },
    "i2.xlarge"   : { "Arch" : "HVM64"  },
    "i2.2xlarge"  : { "Arch" : "HVM64"  },
    "i2.4xlarge"  : { "Arch" : "HVM64"  },
    "i2.8xlarge"  : { "Arch" : "HVM64"  },
    "d2.xlarge"   : { "Arch" : "HVM64"  },
    "d2.2xlarge"  : { "Arch" : "HVM64"  },
    "d2.4xlarge"  : { "Arch" : "HVM64"  },
    "d2.8xlarge"  : { "Arch" : "HVM64"  },
    "hi1.4xlarge" : { "Arch" : "HVM64"  },
    "hs1.8xlarge" : { "Arch" : "HVM64"  },
    "cr1.8xlarge" : { "Arch" : "HVM64"  },
    "cc2.8xlarge" : { "Arch" : "HVM64"  }
  },

  "AWSInstanceType2NATArch" : {
    "t1.micro"    : { "Arch" : "NATPV64"   },
    "t2.nano"     : { "Arch" : "NATHVM64"  },
    "t2.micro"    : { "Arch" : "NATHVM64"  },
```

```
      "t2.small"    : { "Arch" : "NATHVM64"  },
      "t2.medium"   : { "Arch" : "NATHVM64"  },
      "t2.large"    : { "Arch" : "NATHVM64"  },
      "m1.small"    : { "Arch" : "NATPV64"   },
      "m1.medium"   : { "Arch" : "NATPV64"   },
      "m1.large"    : { "Arch" : "NATPV64"   },
      "m1.xlarge"   : { "Arch" : "NATPV64"   },
      "m2.xlarge"   : { "Arch" : "NATPV64"   },
      "m2.2xlarge"  : { "Arch" : "NATPV64"   },
      "m2.4xlarge"  : { "Arch" : "NATPV64"   },
      "m3.medium"   : { "Arch" : "NATHVM64"  },
      "m3.large"    : { "Arch" : "NATHVM64"  },
      "m3.xlarge"   : { "Arch" : "NATHVM64"  },
      "m3.2xlarge"  : { "Arch" : "NATHVM64"  },
      "m4.large"    : { "Arch" : "NATHVM64"  },
      "m4.xlarge"   : { "Arch" : "NATHVM64"  },
      "m4.2xlarge"  : { "Arch" : "NATHVM64"  },
      "m4.4xlarge"  : { "Arch" : "NATHVM64"  },
      "m4.10xlarge" : { "Arch" : "NATHVM64"  },
      "c1.medium"   : { "Arch" : "NATPV64"   },
      "c1.xlarge"   : { "Arch" : "NATPV64"   },
      "c3.large"    : { "Arch" : "NATHVM64"  },
      "c3.xlarge"   : { "Arch" : "NATHVM64"  },
      "c3.2xlarge"  : { "Arch" : "NATHVM64"  },
      "c3.4xlarge"  : { "Arch" : "NATHVM64"  },
      "c3.8xlarge"  : { "Arch" : "NATHVM64"  },
      "c4.large"    : { "Arch" : "NATHVM64"  },
      "c4.xlarge"   : { "Arch" : "NATHVM64"  },
      "c4.2xlarge"  : { "Arch" : "NATHVM64"  },
      "c4.4xlarge"  : { "Arch" : "NATHVM64"  },
      "c4.8xlarge"  : { "Arch" : "NATHVM64"  },
      "g2.2xlarge"  : { "Arch" : "NATHVMG2"  },
      "g2.8xlarge"  : { "Arch" : "NATHVMG2"  },
      "r3.large"    : { "Arch" : "NATHVM64"  },
      "r3.xlarge"   : { "Arch" : "NATHVM64"  },
      "r3.2xlarge"  : { "Arch" : "NATHVM64"  },
      "r3.4xlarge"  : { "Arch" : "NATHVM64"  },
      "r3.8xlarge"  : { "Arch" : "NATHVM64"  },
      "i2.xlarge"   : { "Arch" : "NATHVM64"  },
      "i2.2xlarge"  : { "Arch" : "NATHVM64"  },
      "i2.4xlarge"  : { "Arch" : "NATHVM64"  },
      "i2.8xlarge"  : { "Arch" : "NATHVM64"  },
      "d2.xlarge"   : { "Arch" : "NATHVM64"  },
      "d2.2xlarge"  : { "Arch" : "NATHVM64"  },
      "d2.4xlarge"  : { "Arch" : "NATHVM64"  },
      "d2.8xlarge"  : { "Arch" : "NATHVM64"  },
      "hi1.4xlarge" : { "Arch" : "NATHVM64"  },
      "hs1.8xlarge" : { "Arch" : "NATHVM64"  },
      "cr1.8xlarge" : { "Arch" : "NATHVM64"  },
      "cc2.8xlarge" : { "Arch" : "NATHVM64"  }
    },

    "AWSRegionArch2AMI" : {
      "us-east-1"        : {"PV64" : "ami-d4f7ddbe", "HVM64" : "ami-2df5df47", "HVMG2" : "ami-
95f7c0ff"},
      "us-west-2"        : {"PV64" : "ami-a9ae4ec9", "HVM64" : "ami-42b15122", "HVMG2" : "ami-
83a744e3"},
      "us-west-1"        : {"PV64" : "ami-14f68074", "HVM64" : "ami-f7f58397", "HVMG2" : "ami-
ee62138e"},
      "eu-west-1"        : {"PV64" : "ami-a93484da", "HVM64" : "ami-3c38884f", "HVMG2" : "ami-
25d76556"},
```

```
      "eu-central-1"     : {"PV64" : "ami-e8233884", "HVM64" : "ami-d8203bb4", "HVMG2" : "ami-
8fadb6e3"},
      "ap-northeast-1"   : {"PV64" : "ami-c8aca8a6", "HVM64" : "ami-eeabaf80", "HVMG2" : "ami-
71e6e01f"},
      "ap-northeast-2"   : {"PV64" : "NOT_SUPPORTED", "HVM64" : "ami-431fd12d", "HVMG2" :
"NOT_SUPPORTED"},
      "ap-southeast-1"   : {"PV64" : "ami-6702cc04", "HVM64" : "ami-8504cae6", "HVMG2" : "ami-
1e7ab47d"},
      "ap-southeast-2"   : {"PV64" : "ami-4f04232c", "HVM64" : "ami-a30126c0", "HVMG2" : "ami-
68a1860b"},
      "sa-east-1"        : {"PV64" : "ami-daf477b6", "HVM64" : "ami-e2f4778e", "HVMG2" :
"NOT_SUPPORTED"},
      "cn-north-1"       : {"PV64" : "ami-0534fc68", "HVM64" : "ami-3f36fe52", "HVMG2" :
"NOT_SUPPORTED"}
    }

  },

  "Resources" : {
    "VPC" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
          "CidrBlock" : { "Fn::FindInMap" : ["NetworkValues", "VPC", "CIDR"] },
          "EnableDnsHostnames" : "true",
          "Tags" : [
              {"Key": "Name", "Value": {"Ref":"CommonNameTag"} }
          ]
      }
    },
    "Subnet" : {
       "Type" : "AWS::EC2::Subnet",
       "Properties" : {
           "VpcId" : { "Ref" : "VPC" },
           "CidrBlock" : { "Fn::FindInMap" : ["NetworkValues", "Subnet", "CIDR"] },
            "MapPublicIpOnLaunch" : "true",
           "Tags" : [
              {"Key": "Name", "Value": {"Ref":"CommonNameTag"} }
          ]
       }
    },

    "RouteTable" : {
       "Type" : "AWS::EC2::RouteTable",
       "Properties" : {
           "VpcId" : { "Ref" : "VPC" }
       }
    },

    "InternetGateway" : {
        "Type" : "AWS::EC2::InternetGateway",
        "Properties" : {
            "Tags" : [
                {"Key": "Name", "Value": {"Ref":"CommonNameTag"} }
            ]
        }
    },

    "InternetGatewayAttachment" : {
       "Type" : "AWS::EC2::VPCGatewayAttachment",
       "Properties" : {
           "InternetGatewayId" : { "Ref" : "InternetGateway"},
```

```
            "VpcId" : { "Ref" : "VPC"}
        }
    },

    "Route" : {
        "Type" : "AWS::EC2::Route",
        "Properties" : {
            "DestinationCidrBlock" : "0.0.0.0/0",
            "GatewayId" : { "Ref" : "InternetGateway" },
            "RouteTableId" : { "Ref" : "RouteTable" }
        }
    },

    "SubnetRouteTableAttachment" : {
         "Type" : "AWS::EC2::SubnetRouteTableAssociation",
         "Properties" : {
            "RouteTableId" : { "Ref" : "RouteTable" },
            "SubnetId" : { "Ref" : "Subnet" }
         }
    },

    "CFNRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": { "Service": { "Fn::FindInMap" : [ "Region2Principal", {"Ref" :
"AWS::Region"}, "EC2Principal"]}},
            "Action": [ "sts:AssumeRole" ]
          }]
        },
        "Path": "/"
      }
    },

    "CFNRolePolicy": {
      "Type": "AWS::IAM::Policy",
      "Properties": {
        "PolicyName": "CloudFormerPolicy",
        "PolicyDocument": {
          "Statement": [ {
            "Effect": "Allow",
            "Action"   : [
              "autoscaling:Describe*",
              "cloudformation:Describe*",
              "cloudformation:List*",
              "cloudfront:List*",
              "cloudFront:Get*",
              "cloudtrail:Describe*",
              "cloudtrail:Get*",
              "cloudwatch:Describe*",
              "dynamodb:List*",
              "dynamodb:Describe*",
              "elasticbeanstalk:Describe*",
              "ec2:Describe*",
              "elasticloadbalancing:Describe*",
              "elasticache:Describe*",
              "rds:Describe*",
              "rds:List*",
              "route53:List*",
```

```
            "route53:Get*",
            "s3:List*",
            "s3:Get*",
            "s3:PutObject",
            "sdb:Get*",
            "sdb:List*",
            "sns:Get*",
            "sns:List*",
            "sqs:Get*",
            "sqs:List*",
            "opsworks:Describe*",
            "redshift:Describe*",
            "kinesis:Describe*",
            "kinesis:List*"
          ],
          "Resource": "*"
        } ]
      },
      "Roles": [ { "Ref": "CFNRole" } ]
    }
  },

  "CFNInstanceProfile": {
    "Type": "AWS::IAM::InstanceProfile",
    "Properties": {
      "Path": "/",
      "Roles": [ { "Ref": "CFNRole" } ]
    }
  },

  "WebServer": {
    "Type": "AWS::EC2::Instance",
    "Metadata" : {
      "AWS::CloudFormation::Init" : {
        "configSets" : {
          "full_install" : ["base", "cloudformer"]
        },
        "base" : {
          "packages" : {
            "yum" : {
              "gcc"              : [],
              "gcc-c++"          : [],
              "make"             : [],
              "libxml2-devel"    : [],
              "libxslt-devel"    : [],
              "sqlite-devel"     : [],
              "patch"            : [],
              "readline"         : [],
              "readline-devel"   : [],
              "zlib"             : [],
              "zlib-devel"       : [],
              "libyaml-devel"    : [],
              "libffi-devel"     : [],
              "openssl-devel"    : [],
              "bzip2"            : [],
              "autoconf"         : [],
              "automake"         : [],
              "libtool"          : [],
              "bison"            : [],
              "ruby-devel"       : []
            }
```

```
            }
          },
          "cloudformer" : {
            "sources" : {
              "/home/ec2-user/cloudformer" : {"Fn::Join" : ["/", [
                                                 {"Fn::FindInMap" : ["Region2Examples", {"Ref" :
"AWS::Region"}, "Examples"]},
                                                 "AWSCloudFormer041.zip" ]]}
            },
            "files" : {
              "/home/ec2-user/setup_cloudformer" : {
                "content" : { "Fn::Join" : ["", [
                  "#!/usr/bin/env bash\n",
                  "cd /home/ec2-user/cloudformer\n",
                  "# Setup the CloudFormer service\n",
                  "mkdir -p vendor/bundle\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/rake-
10.4.2.gem\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/bundler-
1.7.11.gem\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/bundle-
0.0.1.gem\n",
                  "/usr/local/bin/bundle install --local --path /home/ec2-
user/cloudformer/vendor/bundle\n",
                  "/usr/local/bin/rake RAILS_ENV=production db:migrate\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/rack-
1.6.0.gem\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/eventmachine-
1.0.4.gem\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/daemons-
1.1.9.gem\n",
                  "gem install --local /home/ec2-user/cloudformer/vendor/cache/thin-
1.6.3.gem\n",
                  "# Create certificate and private key for SSL\n",
                  "mkdir -p /home/ec2-user/cloudformer/.ssl\n",
                  "cd /home/ec2-user/cloudformer/.ssl\n",
                  "openssl genrsa -des3 -passout pass:\"" , { "Ref" : "Password" }, "\" -out
server.pass.key 1024\n",
                  "openssl rsa -passin pass:\"", { "Ref" : "Password" }, "\" -in
server.pass.key -out server.key\n",
                  "openssl req -new -key server.key -out server.csr -subj
\"/C=US/ST=Washington/L=Seattle/O=Amazon Web Services/OU=CloudFormer\"\n",
                  "openssl x509 -req -days 365 -in server.csr -signkey server.key -out
server.crt\n",
                  "rm server.pass.key server.csr\n"
                ]]},
                "mode"  : "000755",
                "owner" : "root",
                "group" : "root"
              },
              "/home/ec2-user/cloudformer/config/initializers/user.rb" : {
                "content" : { "Fn::Join" : ["", [
                  "USER_NAME = \"", { "Ref" : "Username" }, "\"\n",
                  "PASSWORD = \"", { "Ref" : "Password" }, "\"\n"
                ]]},
                "mode"  : "000400",
                "owner" : "root",
                "group" : "root"
              },
              "/usr/bin/cloudformer" : {
                "content" : { "Fn::Join" : ["", [
```

```
                  "#!/usr/bin/env bash\n",
                  "cd /home/ec2-user/cloudformer\n",
                  "/usr/local/bin/thin start -p 443 -e production -d --ssl --ssl-key-file
/home/ec2-user/cloudformer/.ssl/server.key --ssl-cert-file /home/ec2-
user/cloudformer/.ssl/server.crt\n"
                ]]},
                "mode"  : "000755",
                "owner" : "root",
                "group" : "root"
              }
            },
            "commands" : {
              "01_install_cloudformer" : {
                "command" : "/home/ec2-user/setup_cloudformer &>
/var/log/setup_cloudformer.log",
                "cwd" : "/home/ec2-user/cloudformer"
              },
              "02_setup_boot" : {
                "command" : "echo '/usr/bin/cloudformer' >> /etc/rc.local",
                "cwd" : "/"
              }
            }
          }
        }
      },
      "Properties": {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" },
                          { "Fn::FindInMap" : [ "AWSInstanceType2Arch", "t2.medium", "Arch" ]
} ] },
        "InstanceType"       : "t2.medium",
        "SecurityGroupIds"     : [ {"Ref" : "WebServerSecurityGroup"} ],
        "SubnetId" : { "Ref" : "Subnet" },
        "IamInstanceProfile" : { "Ref" : "CFNInstanceProfile" },
        "UserData"           : { "Fn::Base64" : { "Fn::Join" : ["", [
          "#!/bin/bash -xe\n",
          "yum update -y aws-cfn-bootstrap\n",

          "/opt/aws/bin/cfn-init -v ",
          "         --stack ", { "Ref" : "AWS::StackId" },
          "         --resource WebServer ",
          "         --configsets full_install ",
          "         --region ", { "Ref" : "AWS::Region" }, "\n",

          "/opt/aws/bin/cfn-signal -e $? ",
          "         --stack ", { "Ref" : "AWS::StackId" },
          "         --resource WebServer ",
          "         --region ", { "Ref" : "AWS::Region" }, "\n"
        ]]}}
      },
      "CreationPolicy" : {
        "ResourceSignal" : {
          "Timeout" : "PT30M"
        }
      }
    },

    "WebServerSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable HTTPS access via port 443",
        "VpcId" : { "Ref" : "VPC" },
```

```
        "SecurityGroupIngress" : [
          {"IpProtocol" : "tcp", "FromPort" : "443", "ToPort" : "443", "CidrIp" : "0.0.0.0/0"}
        ]
      }
    }
  },

  "Outputs" : {
    "WebsiteURL" : {
      "Value" : { "Fn::Join" : ["", ["https://", { "Fn::GetAtt" : [ "WebServer",
"PublicDnsName" ]} ]] },
      "Description" : "URL for CloudFormer"
    }
  }
}
```

Lire AWS CloudFormation en ligne: https://riptutorial.com/fr/amazon-web-services/topic/6988/aws-cloudformation

Chapitre 5: AWS Lambda

Introduction

AWS Lambda est un service qui vous permet d'exécuter du code back-end sans avoir à configurer ou à gérer des serveurs. AWS Lambda prend en charge la mise à l'échelle et la haute disponibilité. Le coût dépend directement de la fréquence et de la durée d'exécution de votre code.

Vous trouverez des exemples sur la création et le déploiement de fonctions AWS Lambda dans différentes langues.

Remarques

- Le code AWS Lambda doit être écrit sans état. Bien que l'instance d'un lambda puisse être conservée et réutilisée, vous ne devez jamais vous y attendre.

Examples

Projet Basic Gradle Java

Pour déployer du code Java sur AWS Lambda, vous devez créer un fichier zip de distribution contenant toutes les dépendances nécessaires lors de l'exécution. Exemple de projet dans Gradle:

```
apply plugin: 'java'

repositories {
    mavenCentral()
}

dependencies {
    compile 'com.amazonaws:aws-lambda-java-core:1.1.0'
}

task buildZip(type: Zip) {
    from compileJava
    from processResources
    into('lib') {
        from configurations.runtime
    }
}

build.dependsOn buildZip
```

L'exécution de la `gradle build` créera un fichier zip avec toutes les dépendances intégrées à votre code, prêt à être déployé.

Code Lambda de base en Java

Un lambda a besoin d'un gestionnaire, qui servira de point d'entrée à votre application. Chaque gestionnaire doit implémenter l'interface `RequestHandler<I, O>` où `I` est le type d'entrée et `O` le type de sortie. Le type d'entrée est ensuite transmis à la méthode `handleRequest()` et la méthode renvoie le type de sortie.

Un exemple simple pourrait ressembler à ceci:

```
package com.example.lambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class HelloNumber implements RequestHandler<Integer, String> {

    @Override
    public String handleRequest(Integer input, Context context) {
        return "You sent " + input;
    }
}
```

Code Lambda de base en JavaScript (NodeJs)

Un lambda a besoin d'un gestionnaire, qui servira de point d'entrée à votre application. Dans le cas le plus simple, vous obtenez votre entrée du `context` et transmettez le résultat en utilisant la fonction `callback()` :

exports.handler = (event, context, callback) => {callback (null, 'Vous avez envoyé' + event.number); };

Création d'AWS Lambda à l'aide de l'interface graphique Web (Java)

Pour créer un lambda Java à l'aide de l'interface graphique, assurez-vous d'abord que votre zip de distribution est prêt. Vous aurez également besoin d'un compte AWS pouvant créer des lambdas.

- Basculez vers AWS Lambda, fermez l'écran d'introduction (ou lisez la documentation Get Started) et appuyez sur le bouton `Create a Lambda Function` .
- Sélectionnez un plan approprié. Un modèle est un exemple de modèle qui peut vous aider à créer votre lambda. Pour l'instant, commencez par une `Blank Function` .
- Pour l'instant, ne vous inquiétez pas des déclencheurs et ignorez cet écran.
- Nommez votre fonction et téléchargez le fichier zip.
- Sur le même écran de configuration, saisissez votre nom de gestionnaire en tant que classe Java complète avec package, éventuellement avec la méthode qui gérera les requêtes. Si vous avez utilisé l'interface Java à partir de l'exemple Basic Lambda Code, le nom de la classe est suffisant.
- Sur le même écran, sélectionnez `Create a new role from template(s)` et nommez votre rôle. Le rôle est utilisé pour donner à votre lambda l'accès à d'autres ressources AWS. Dans ce cas, laissez-le vide car nous n'avons pas besoin d'autres ressources.

- Sélectionnez la mémoire disponible la plus faible (128 Mo). La mémoire influe également sur le coût. Enregistrez votre fonction.

Test du code Lambda de base

Si vous avez déployé le Lambda avec succès, vous pouvez également le tester directement dans l'interface graphique. Lorsque vous cliquez sur le bouton bleu `Test` pour la première fois, il vous présente une création d'un nouvel événement de test.

Si vous testez le code Java à partir du *code Lambda de base dans l'exemple Java* , supprimez le corps entier et ajoutez simplement un chiffre `1` .

Si vous testez le code Javascript à partir du code *Basic Lambda dans l'exemple Javascript (Node)* , créez un fichier json qui ressemble à ceci:

```
{
    "number": "1"
}
```

Exécutez le test. Vous devriez voir le résultat:

"Vous avez envoyé 1"

Ajout du déclencheur AWS API Gateway

La méthode la plus courante pour appeler un Lambda consiste à utiliser API Gateway, qui mappe un appel REST au code lambda. Vous pouvez ajouter plusieurs déclencheurs à votre Lambda à tout moment, y compris lors de la création d'un nouveau lambda.

Si vous connaissez bien API Gateway, vous pouvez associer n'importe quelle méthode à un lambda déployé. Créez simplement une nouvelle méthode avec le type d'intégration défini sur la fonction Lambda et pointez sur votre lambda. Vous pouvez mapper les deux entrées REST aux entrées et sorties lambda sur les sorties REST.

TODO: Décidez s'il faut décrire API Gateway ici ou référencer une autre partie de la documentation.

Lire AWS Lambda en ligne: https://riptutorial.com/fr/amazon-web-services/topic/8918/aws-lambda

Chapitre 6: Classe Racine

Examples

La classe de racine Amazon api est la suivante.

```
public class AmazonRootobject
{
    public Itemsearchresponse ItemSearchResponse { get; set; }
}

public class Itemsearchresponse
{
    public string xmlns { get; set; }
    public Operationrequest OperationRequest { get; set; }
    public Items Items { get; set; }
}

public class Operationrequest
{
    public Httpheaders HTTPHeaders { get; set; }
    public string RequestId { get; set; }
    public Arguments Arguments { get; set; }
    public string RequestProcessingTime { get; set; }
}

public class Httpheaders
{
    public Header Header { get; set; }
}

public class Header
{
    public string Name { get; set; }
    public string Value { get; set; }
}

public class Arguments
{
    public Argument[] Argument { get; set; }
}

public class Argument
{
    public string Name { get; set; }
    public object Value { get; set; }
}

public class Items
{
    public Request Request { get; set; }
    public string TotalResults { get; set; }
    public string TotalPages { get; set; }
    public string MoreSearchResultsUrl { get; set; }
    public Item[] Item { get; set; }
}
```

```
public class Request
{
    public string IsValid { get; set; }
    public Itemsearchrequest ItemSearchRequest { get; set; }
}

public class Itemsearchrequest
{
    public string Keywords { get; set; }
    public string[] ResponseGroup { get; set; }
    public string SearchIndex { get; set; }
    public string Sort { get; set; }
}

public class Item
{
    public string ASIN { get; set; }
    public string ParentASIN { get; set; }
    public string DetailPageURL { get; set; }
    public Itemlinks ItemLinks { get; set; }
    public Smallimage SmallImage { get; set; }
    public Mediumimage MediumImage { get; set; }
    public Largeimage LargeImage { get; set; }
    public Imagesets ImageSets { get; set; }
    public Itemattributes ItemAttributes { get; set; }
    public OfferSummary OfferSummary { get; set; }
    public Offers Offers { get; set; }

    public Variationsummary VariationSummary { get; set; }
}

public class Variationsummary
{
    public Highestprice HighestPrice { get; set; }
    public Lowestprice LowestPrice { get; set; }
    public Highestsaleprice HighestSalePrice { get; set; }
    public Lowestsaleprice LowestSalePrice { get; set; }
}

public class Highestprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Lowestprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Highestsaleprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Lowestsaleprice
```

```
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Itemlinks
{
    public Itemlink[] ItemLink { get; set; }
}

public class Itemlink
{
    public string Description { get; set; }
    public string URL { get; set; }
}

public class Smallimage
{
    public string URL { get; set; }
    public Height Height { get; set; }
    public Width Width { get; set; }
}

public class Height
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Width
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Mediumimage
{
    public string URL { get; set; }
    public Height1 Height { get; set; }
    public Width1 Width { get; set; }
}

public class Height1
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Width1
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Largeimage
{
    public string URL { get; set; }
    public Height2 Height { get; set; }
    public Width2 Width { get; set; }
}
```

```
public class Height2
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Width2
{
    public string Units { get; set; }
    public string text { get; set; }
}

public class Imagesets
{
    public object ImageSet { get; set; }
}

public class Itemattributes
{
    public string Binding { get; set; }
    public string Brand { get; set; }
    public string Color { get; set; }
    public string Model { get; set; }
    public string Manufacturer { get; set; }
    public string ProductGroup { get; set; }
    public string Title { get; set; }
    public ListPrice ListPrice { get; set; }
}

public class ListPrice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class OfferSummary
{
    public Lowestnewprice LowestNewPrice { get; set; }
    public Lowestusedprice LowestUsedPrice { get; set; }
    public string TotalNew { get; set; }
    public string TotalUsed { get; set; }
    public string TotalCollectible { get; set; }
    public string TotalRefurbished { get; set; }
    public Lowestrefurbishedprice LowestRefurbishedPrice { get; set; }
}

public class Lowestnewprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Lowestusedprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}
```

```
public class Lowestrefurbishedprice
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Offers
{
    public string TotalOffers { get; set; }
    public string TotalOfferPages { get; set; }
    public string MoreOffersUrl { get; set; }
    public Offer Offer { get; set; }
}

public class Offer
{
    public Merchant Merchant { get; set; }
    public Offerattributes OfferAttributes { get; set; }
    public Offerlisting OfferListing { get; set; }
}

public class Merchant
{
    public string Name { get; set; }
}

public class Offerattributes
{
    public string Condition { get; set; }
}

public class Offerlisting
{
    public string OfferListingId { get; set; }
    public string PricePerUnit { get; set; }
    public Price Price { get; set; }
    public string Availability { get; set; }
    public Availabilityattributes AvailabilityAttributes { get; set; }
    public string IsEligibleForSuperSaverShipping { get; set; }
    public string IsEligibleForPrime { get; set; }
    public Saleprice SalePrice { get; set; }
    public Amountsaved AmountSaved { get; set; }
    public string PercentageSaved { get; set; }
}

public class Price
{
    public string Amount { get; set; }
    public string CurrencyCode { get; set; }
    public string FormattedPrice { get; set; }
}

public class Availabilityattributes
{
    public string AvailabilityType { get; set; }
    public string MinimumHours { get; set; }
    public string MaximumHours { get; set; }
}
```

```
    public class Saleprice
    {
        public string Amount { get; set; }
        public string CurrencyCode { get; set; }
        public string FormattedPrice { get; set; }
    }

    public class Amountsaved
    {
        public string Amount { get; set; }
        public string CurrencyCode { get; set; }
        public string FormattedPrice { get; set; }
    }
```

Classe affaire

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Linq;
using ApplicationDataServices.SBEntityBox;

namespace ApplicationManagementLayer.Affiliate
{
    public class Amazon
    {
        private int ItemPage { get; set; }
        public int TotalNumberOfItem { get; set; }

public Amazon()
{
    ItemPage = 1;
    TotalNumberOfItem = 0;
}


string XMLURL = string.Empty;



 public async Task<AmazonRootobject> getProductsByKeywords(string q, Dictionary<string,
string> CategoryNames, int ItemPage, string categorynamebyid)
{
    try
    {
        AWSSignedRequestHelper helper = new AWSSignedRequestHelper("para1", "para2",
"webservices.amazon.in", "para3");
        IDictionary<string, string> r1 = new Dictionary<string, String>();
        r1["Service"] = "AWSECommerceService";

        r1["Operation"] = "ItemSearch";


        if (CategoryNames != null && CategoryNames.Any() && CategoryNames.Where(o =>
o.Key.Equals("AmazonCategoryName")).Any())
```

```
            r1["SearchIndex"] = CategoryNames.Where(o =>
o.Key.Equals("AmazonCategoryName")).First().Value;
        else
            r1["SearchIndex"] = "All";

        if (!r1["SearchIndex"].Equals("All") && CategoryNames != null && CategoryNames.Any()
&& CategoryNames.Where(o => o.Key.Equals("AmazonReferenceCategoryId")).Any())
            r1["BrowseNode"] = CategoryNames.Where(o =>
o.Key.Equals("AmazonReferenceCategoryId")).First().Value;

        if (!string.IsNullOrEmpty(q))
            r1["Keywords"] = q;
        else if (!string.IsNullOrEmpty(categorynamebyid))
            r1["Keywords"] = categorynamebyid;
        else if (CategoryNames != null && CategoryNames.Any() && CategoryNames.Where(o =>
o.Key.Equals("AmazonCategoryName")).Any())
            r1["Keywords"] = CategoryNames.Where(o =>
o.Key.Equals("AmazonCategoryName")).First().Value;
        else
            return null;

        r1["ResponseGroup"] = "Images,ItemAttributes,OfferFull,Offers,Variations";
        r1["Version"] = "2013-08-01";
        r1["ItemPage"] = ItemPage.ToString();
        //r1["Sort"] = "salesrank";

        string strRequestUrl = helper.Sign(r1);

        string output = null;
        using (System.Net.Http.HttpClient wc = new System.Net.Http.HttpClient())
        {
            var request = new System.Net.Http.HttpRequestMessage()
            {
                RequestUri = new Uri(strRequestUrl),
                Method = System.Net.Http.HttpMethod.Get,
            };

            /*var task =*/
            await wc.SendAsync(request)
                .ContinueWith((taskwithmsg) =>
                {
                    var response = taskwithmsg.Result;

                    var jsonTask = response.Content.ReadAsStringAsync();
                    jsonTask.Wait();
                    output = jsonTask.Result;
                });
            //task.Wait();

        }

        XmlDocument doc = new XmlDocument();
        doc.LoadXml(output);
        string outputJson = XmlToJSON(doc);

        var pro = new
System.Web.Script.Serialization.JavaScriptSerializer().Deserialize<AmazonRootobject>(outputJson);

        TotalNumberOfItem = !string.IsNullOrEmpty(pro.ItemSearchResponse.Items.TotalResults) ?
Convert.ToInt32(pro.ItemSearchResponse.Items.TotalResults) : 0;
        return pro;
```

```
        //return "";
    }
    catch
    {
        return null;
    }
}

http://stackoverflow.com/documentation/amazon-web-services/drafts/87373#



    }
}
```

Lire Classe Racine en ligne: https://riptutorial.com/fr/amazon-web-services/topic/7357/classe-racine

Chapitre 7: Déployer une image de conteneur d'ancrage à l'aide d'ECS

Remarques

Avant de pouvoir ajouter des instances ECS à un cluster, vous devez d'abord accéder à la console de gestion EC2 et créer `ecs-optimized` instances `ecs-optimized` par `AmazonEC2ContainerServiceforEC2Role` avec un rôle IAM auquel la stratégie `AmazonEC2ContainerServiceforEC2Role` associée.

1. Accédez à votre tableau de bord EC2 , puis cliquez sur le bouton `Launch Instance` .
2. Sous `Community AMIs` , recherchez `ecs-optimized` et sélectionnez celui qui correspond le mieux aux besoins de votre projet. Tout va travailler. Cliquez sur Suivant.
3. Lorsque vous arrivez à `Configure Instance Details` , cliquez sur le `create new IAM role link` créer un nouveau rôle `create new IAM role link` et créez un nouveau rôle appelé `ecsInstanceRole` .
4. Attachez la stratégie `AmazonEC2ContainerServiceforEC2Role` à ce rôle.
5. Par défaut, votre instance de conteneur est lancée dans votre cluster `default` . Si vous souhaitez lancer votre propre cluster au lieu de celui par défaut, choisissez la liste `Advanced Details` et collez le script suivant dans le champ `User data` , en remplaçant `your_cluster_name` par le nom de votre cluster.

```
#!/bin/bash
echo ECS_CLUSTER=your_cluster_name >> /etc/ecs/ecs.config
```

6. Ensuite, terminez la configuration de votre instance ECS.

REMARQUE: Si vous créez un serveur Web, vous souhaiterez créer un groupe de `securityGroup` pour autoriser l'accès au port 80.

7. Créer un référentiel: `aws ecr create-repository --repository-name example-repository`
8. Authentifiez votre client Docker dans votre registre: `aws ecr get-login --region us-east-1 | sh`
9. Construisez votre image Docker: `docker build -t example-image .`
10. Marquez votre image afin de pouvoir transférer l'image vers ce référentiel: `docker tag example-image:latest example-namespace/example-image:latest`
11. Poussez cette image dans votre dépôt AWS nouvellement créé: `docker push example-namespace/example-image:latest`
12. Enregistrer une tâche ECS Définition: `aws ecs register-task-definition --cli-input-json example-task.json`
13. Exécutez la tâche: `aws ecs run-task --task-definition example-task`

Examples

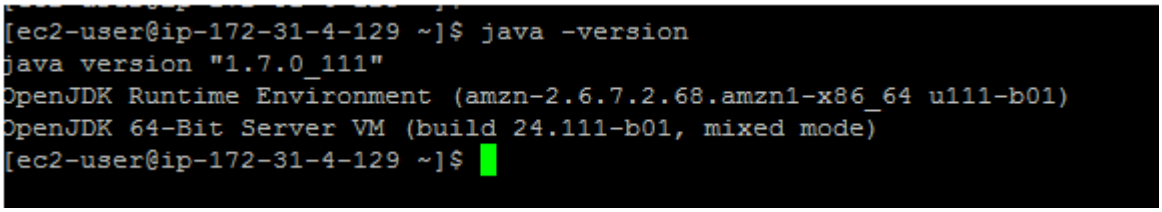
example-task.json

```
{
  "family": "example-task",
  "containerDefinitions": [
    {
        "environment": [],
        "name": "example-container",
        "image": "example-namespace/example-image:latest",
        "cpu": 10,
        "memory": 500,
        "portMappings": [
            {
                "containerPort": 8080,
                "hostPort": 80
            }
        ],
        "entryPoint": [],
        "essential": true
    }
  ]
}
```

Déployer un exemple d'application sur le service AWS ECS en tant que preuve de concept

Suivez les étapes suivantes pour essayer un exemple d'application sur le service AWS ECS en tant que preuve de concept.

1. Connectez-vous à la console de gestion AWS et accédez au **catalogue de services AWS -> Compute -> Ec2**
2. Créez une VM (instance EC2) en utilisant le système d'exploitation Linux 64 bits d'Amazon, ceci nous permettra de configurer docker, git, l'agent AWS ECS et d'autres outils. Nous utiliserons également la même machine virtuelle en tant que nœud dans le cluster ECS pour déployer des applications basées sur des conteneurs. Suivez les étapes ci-dessous pour créer une machine virtuelle.
   a) Suivez les étapes habituelles pour créer une instance EC2, donnez une attention particulière aux étapes suivantes lors de la création d'une instance EC2.
   b) Sélectionnez un rôle IAM avec les autorisations suivantes - AmazonEC2ContainerServiceforEC2Role
   c) Assurez-vous que java est installé sur la machine virtuelle



3. Installer docker [exécuter les commandes ci-dessous]
   mettre d'abord à jour le référentiel de paquets yum

```
sudo yum update -y
```

maintenant installer docker exécuter yum install

```
sudo yum install -y docker
```

4. Démarrer le service docker

```
sudo service docker start
```

5. Ajoutez l'utilisateur ec2 au groupe docker pour pouvoir exécuter les commandes Docker sans utiliser sudo.

```
sudo usermod -a -G docker ec2-user
```

6. Déconnectez-vous de l'EC2 et reconnectez-vous pour récupérer les nouvelles autorisations du groupe de base.
7. Vérifiez que l'utilisateur ec2 peut exécuter les commandes Docker sans sudo.

```
docker info
```

8. Installer Git

```
sudo yum install -y git
```

9. Clonez l'exemple d'application PHP sur l'instance Ec2 à partir de git. Nous utiliserons cette application pour notre POC.

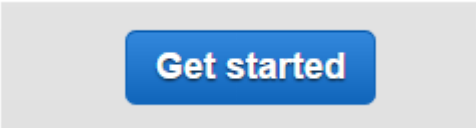
```
git clone https://github.com/awslabs/ecs-demo-php-simple-app

cd ecs-demo-php-simple-app
```

vérifier que Dockerfile existe en listant le contenu du répertoire

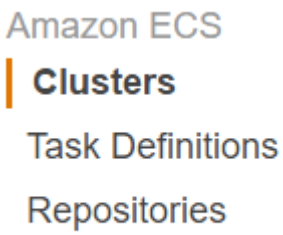
```
ls
```

10. Accédez au catalogue de services AWS -> Compute -> Service Conteneur Ec2
11. Cliquez sur Commencer



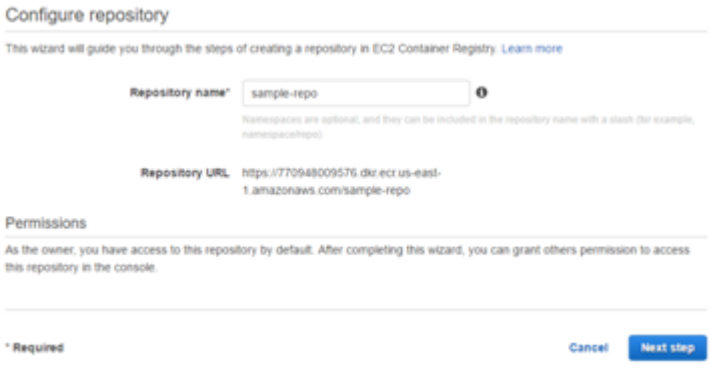
12. Cliquez sur annuler

13. Cliquez sur les référentiels du menu Référentiels à gauche



14. Cliquez sur Commencer

15. Entrez le nom du référentiel et cliquez sur suivant



16. Configurer les outils Ec2

```
aws configure
```

fournir l'ID de clé d'accès AWS, la clé d'accès secret, le nom de région par défaut de votre compte

17. Construire, étiqueter et pousser l'image Docker
    a) Récupérez la commande de connexion au docker que vous pouvez utiliser pour authentifier votre client Docker dans votre registre:

```
aws ecr get-login --region us-east-1
```

b) Exécuter la commande return en sortie de l'étape précédente

18. Construisez l'image Docker à partir de votre fichier Docker. (Rappelez l'étape 9, où vous avez téléchargé un exemple d'application docker)
    une)

```
docker build -t amazon-ecs-sample .
```

(Notez le "." Signifie répertoire courant)
b) Exécutez les images du docker pour vérifier que l'image a été créée correctement et que le nom de l'image contient un référentiel dans lequel vous pouvez transférer vos modifications sur

l'image du docker.

```
docker images
```

```
[ec2-user@ip-172-31-4-129 ecs-demo-php-simple-app]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
amazon-ecs-sample   latest              9431571817d8        About a minute ago
 228.6 MB
```

c) Exécutez l'image nouvellement construite. L'option -p 80:80 associe le port exposé 80 du conteneur au port 80 du système hôte (instance Ec2 dans ce cas).

```
docker run -p 80:80 amazon-ecs-sample
```

Ignorer l'avertissement «apache2: Impossible de déterminer de manière fiable le nom de domaine complet du serveur, en utilisant 172.17.0.2 for ServerName»

19. Essayez d'accéder à la page Web de l'application exemple sur le navigateur, assurez-vous que le port 80 est ouvert dans les groupes de sécurité associés à l'instance

```
http://<ec2-instance-dns-address>
```

20. Appuyez sur la touche ctrl + c, cela arrêtera l'image du docker. L'exemple d'application ne doit pas être accessible.
21. Après avoir vérifié avec succès notre exemple d'application Docker, nous allons essayer de configurer un cluster pour qu'il exécute automatiquement l'application exemple. En outre, pour les besoins de la démonstration, nous essaierons d'utiliser l'instance ec2 existante en tant que nœud dans le cluster. Cela peut être réalisé en installant un programme d'agent sur l'instance ec2.
22. Installation de Amazon ECS Container Agent sur l'instance ec2
    une)

```
sudo yum install -y ecs-init
```

b) Redémarrer le démon docker

```
sudo service docker restart
```

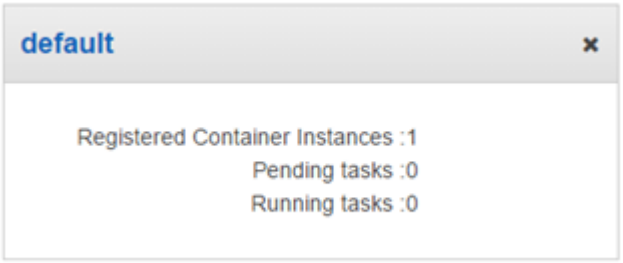
c) Lancez le job de démarrage ecs-init

```
sudo start ecs
```

d) (Facultatif) Vous pouvez vérifier que l'agent est en cours d'exécution et voir des informations sur votre nouvelle instance de conteneur avec l'API d'introspection d'agent. Assurez-vous que le port 51678 est ouvert dans le groupe de sécurité.

```
curl http://localhost:51678/v1/metadata
```

23. Accédez au catalogue de services AWS -> Calculer -> Service Conteneur Ec2 -> Cluster et vérifiez qu'un cluster par défaut est créé.



24. Nous procédons maintenant à la création d'un groupe de tâches et à l'ajout de notre image de docker comme tâche à exécuter sur le cluster.
    a) Examinez le fichier simple-app-task-def.json dans le dossier ecs-demo-php-simple-app.
    b) Modifiez le fichier simple-app-task-def.json et répétez le momeory afin qu'il puisse s'exécuter sur une instance éligible de niveau gratuit (je suppose que l'on utilise une instance ec2 éligible de niveau gratuit pour ce POC, sinon pas besoin de réduire la mémoire) limite)
    c) Mettre à jour la **mémoire = 250** dans toute l'occurrence du fichier simple-app-task-def.json
    d) Enregistrez une définition de tâche avec le fichier simple-app-task-def.json.

```
aws ecs register-task-definition --cli-input-json file://simple-app-task-def.json
```

e) Accédez à la définition de tâche dans la page de service de conteneur ec2, vous trouverez la définition de tâche enregistrée

f) Utilisez la commande AWS CLI suivante pour exécuter une tâche avec la définition de tâche console-sample-app.

```
aws ecs run-task --task-definition console-sample-app
```

g) Ouvrez l'exemple d'application Web dans le navigateur, il devrait être accessible (reportez-vous à l'étape 19)

Merci d'avoir lu, partagez vos commentaires et vos questions pour une discussion de suivi.

Lire Déployer une image de conteneur d'ancrage à l'aide d'ECS en ligne: https://riptutorial.com/fr/amazon-web-services/topic/7711/deployer-une-image-de-conteneur-d-ancrage-a-l-aide-d-ecs

Chapitre 8: Haricot Élastique

Remarques

Limitations actuelles (à partir du 2016-10-03)

- Les balises d'environnement ne peuvent pas être modifiées une fois l' *environnement* créé, alors choisissez judicieusement.
- La mise à l'échelle automatique dans Elastic Beanstalk est limitée à *Simple* et *Planifié* . Si vous souhaitez utiliser *Step-Scaling* , déterminez si Elastic Beanstalk convient parfaitement.

Automatisation avec Jenkins

Il existe un excellent plug-in de déploiement AWSEB pour Jenkins qui se connectera pour un déploiement sur Elastic Beanstalk (les déploiements bleu / vert avec terminaison automatique d'inactivité ne sont qu'une case à cocher).

Examples

Introduction au haricot élastique

Elastic Beanstalk (EB) est essentiellement un hybride entre Golden AMI et CloudFormation , tout en simplifiant considérablement la courbe d'apprentissage de Puppet ou de Chef .

Un déploiement Elastic Beanstalk est divisé en deux composants: Application et Environnement.

Application

Considérez ceci comme votre regroupement de premier niveau, votre application elle-même. Par exemple, une seule application ("MyWebApp") peut avoir plusieurs environnements ("Production" et "Staging").

Environnement

Chaque environnement consistera en un déploiement d'architecture complet (instances EC2 , Elastic Load Balancer , groupe de calibrage automatique et alarmes Cloudwatch). Toute la configuration de l'environnement est configurée et maintenue pour votre compte automatiquement.

Déploiement d'une application

Le déploiement de votre application est aussi simple que le téléchargement d'un fichier zip contenant votre code. Chaque fichier zip (appelé *Application Version*) que vous téléchargez est associé à une *application* . Vous pouvez donc le télécharger et le déployer dans plusieurs *environnements* .

Personnalisation de l'environnement

Par défaut, Elastic Beanstalk déploiera des AMI "stockées" gérées par Amazon. Pour la plupart des applications, cela est suffisant, mais il se peut que vous souhaitiez modifier l'environnement (par exemple, modifier le fuseau horaire, ajouter des packages / dépendances non présents dans le code, etc.).

Il existe deux manières de personnaliser l'AMI EB utilisée: ebextensions ou une AMI personnalisée.

ebextensions - Un dossier, littéralement appelé «.ebextensions», qui peut éventuellement être placé à la racine de votre *version de l'application* (le fichier zip que vous avez téléchargé contenant votre code). Dans le dossier ebextensions, vous pouvez placer des fichiers YAML définissant les scripts personnalisés, les dépendances, etc. que vous souhaitez exécuter côté serveur pendant le processus de déploiement. Un certain nombre de points d'accroche sont disponibles. Pour obtenir les dernières informations, consultez la documentation correspondante: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html

---

Gotchas / Choses à savoir

Case à cocher du VPC - Lors de la création d'un environnement, l'option est disponible discrètement pour déterminer si l'environnement doit être créé / placé dans un VPC. Si vous avez besoin de votre application pour communiquer avec les ressources existantes que vous avez créées, vérifiez cette boîte. Sinon, Elastic Beanstalk créera un nouveau groupe de sécurité isolé du reste de votre VPC. *Bien que vous puissiez ajuster manuellement les paramètres du groupe de sécurité après la création, essayer de l'ajouter essentiellement à un VPC entraînera ultérieurement toute une série de problèmes.*

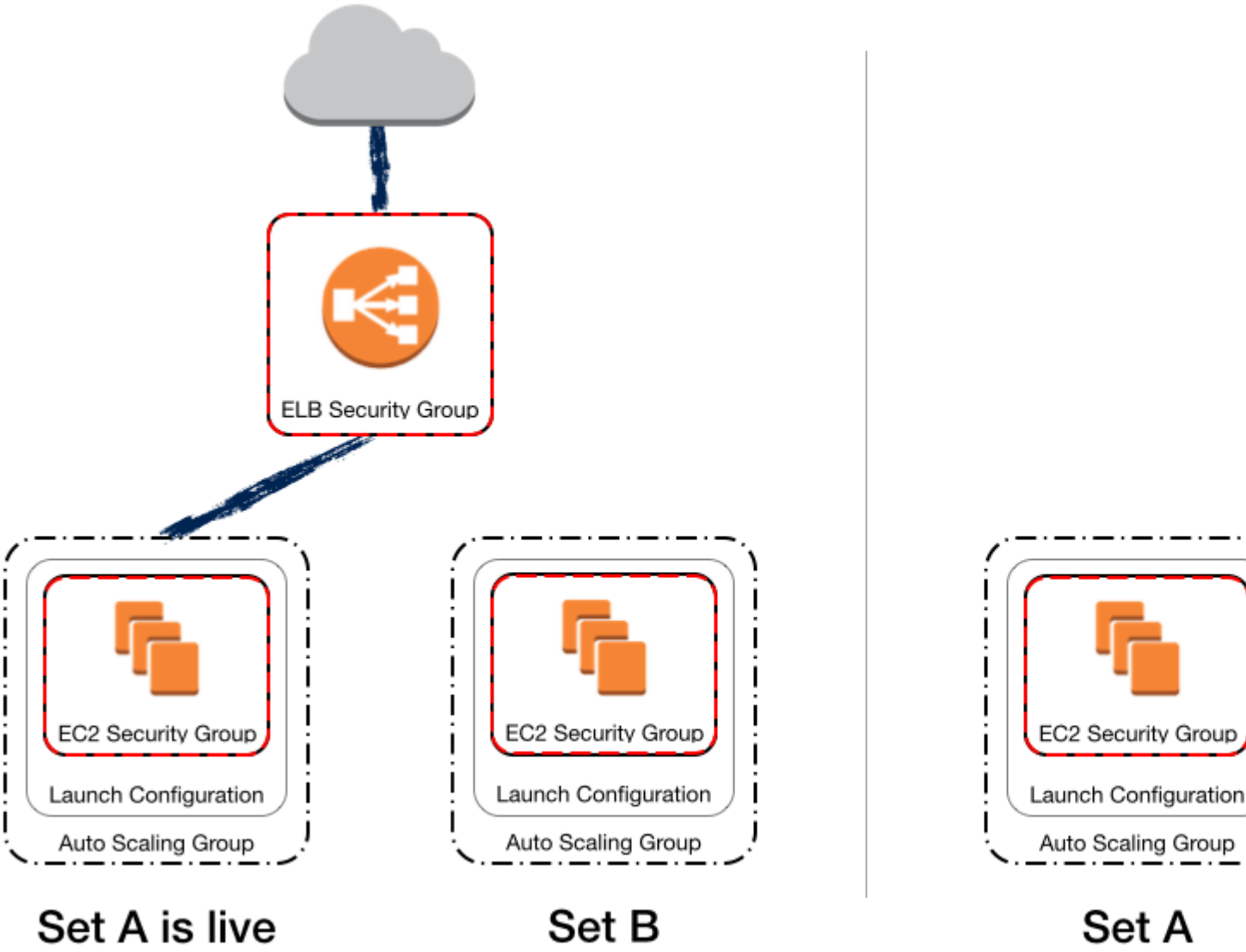
RDS - Lors de la création d'un environnement, vous avez la possibilité de créer une instance RDS dans le cadre de l'environnement. Il n'est pas recommandé de l'utiliser, car chaque fois que vous devez «reconstruire» l'environnement (par exemple, déploiements bleu / vert, résolution des problèmes), il détruira et recréera l'instance RDS (avec toutes les données).

Déploiements bleu / vert sur Elastic Beanstalk

Le déploiement Blue / Green est une technique de publication qui réduit les temps d'arrêt et les risques en exécutant deux environnements de production identiques (l'un appelé «Blue», l'autre appelé «Green»). À tout moment, un seul des environnements sert le trafic en direct, tandis que l'autre est inactif.

Lors du déploiement d'une nouvelle version, le code est déployé dans l'environnement inactif (par exemple, vert) et après confirmation d'un déploiement réussi, le trafic en direct est commuté (par exemple, le vert contient du nouveau code, le trafic bleu est désormais routé vers vert). Le prochain déploiement de code se produira dans le nouvel environnement inactif (en suivant l'exemple, qui sera désormais bleu).

Exemple / Aide visuelle :



source d'image: https://cloudnative.io/statics/blog/aws-blue-green-deployment.png

---

Lorsque vous utilisez Elastic Beanstalk (EB), vous pouvez facilement créer plusieurs environnements qui sont des clones exacts les uns des autres pour le déploiement de code. Après avoir confirmé que le nouvel environnement est prêt à fonctionner, il suffit d'utiliser la fonction «Swap URLs» dans EB pour échanger des environnements.

Instructions pas à pas: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.CNAMESwap.html

Lire Haricot Élastique en ligne: https://riptutorial.com/fr/amazon-web-services/topic/7207/haricot-elastique

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec amazon-web-services	Community, Cyril Duchon-Doris, Karan Shah, Lynn Langit
2	Amazon Cognito	Jeet
3	Amazon DynamoDB	swapnil kadu
4	AWS CloudFormation	Hardeep Singh, Naveen Vijay
5	AWS Lambda	jarmod, sm4
6	Classe Racine	vicky
7	Déployer une image de conteneur d'ancrage à l'aide d'ECS	mrded, Satish
8	Haricot Élastique	Lorenzo Aiello