FREE eBook

LEARNING

amazon-web-services

#amazon-web-services

Table of Contents

About

Chapter 1: Getting started with amazon-web-services

Remarks

This section provides an overview of what amazon-web-services is, and why a developer might want to use it.

It should also mention any large subjects within amazon-web-services, and link out to the related topics. Since the Documentation for amazon-web-services is new, you may need to create initial versions of those related topics.

Versions

Version	Release Date
1.0.0	2017-01-10

Examples

Before it is too late

Tips & Tricks to avoid nasty situations

EC2 Instances and EBS

- Set IAM Roles.

Unlike tags, the IAM Role is **set once and for all on EC2 instanciation** (even after 4 years) ! Try to identify and categorize beforehand your instances so you can give an them appropriate IAM roles. IAM Roles are a nice way to identify your machines, it will let amazon automatically store Instance Profile credentials safely in your machines, and you will be easily able to give extra privileges.

Consider the following situation where you have Database servers, and you realize you want to monitor memory/disk usage. Amazon CloudWatch does not provide this metric out-of-the-box, and You'll need to set up extra privileges to send custom data to CloudWatch. If you have an IAM "Database" Role, you can easily attach new policies to your existing Database instances to let them send Memory reports to CloudWatch. No IAM Roles ? You have to recreate your Database instances, or give them permission individually.

- Beware of snapshot integrity

Amazon lets you snapshot EBS volumes, however in case you use several volumes on the same machine (in RAID configuration, multiple EBS volumes for your database), it is impossible to guarantee the integrity of those snapshots, which may happen at different times on the different EBS volumes.

Always ensure no data is written (stop the VM, or use application specific code (eg `db.fsyncLock()`) to ensure no data is written during the snapshot.

CloudWatch

- Use Amazon Cloudwatch + SNS alerts on top of your application error notifiers

Create alerts for anormal behavior of your machines, and configure to send notifications via Amazon SNS (eg email addresses) in case of problems. Having exception notifiers on your application won't help if your server cannot even be pinged. On the other hand, apart from 500 error codes Amazon has little information on your application and how it is supposed to work, you should consider adding application-specific health monitoring.

Read Getting started with amazon-web-services online: https://riptutorial.com/amazon-web-services/topic/798/getting-started-with-amazon-web-services

Chapter 2: Amazon Cognito

Examples

User Identity management using 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;

});
};
```

Read Amazon Cognito online: https://riptutorial.com/amazon-web-services/topic/3425/amazon-cognito

Chapter 3: Amazon DynamoDB

Examples

DynamoDB basic Crud Operation using 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);
        }
    });
}
```

Read Amazon DynamoDB online: https://riptutorial.com/amazon-web-services/topic/6429/amazon-dynamodb

Chapter 4: AWS CloudFormation

Examples

CloudFormation sample script to create an EC2 instance along with a Security Group to associate with.

This example will create an EC2 instance of **t2.micro** type in **N.Virginia** region running **Amazon Linux**. During the execution, it will ask to select the KeyPair to use and an I.P. CIDR from where you can SSH to the instance, use default to make SSH open to the 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 in VPC

CloudFormer template translates the existing AWS entities to a generate new CloudFormation template to accelerate the time to recreate the environment. The CloudFormer launches in a default VPC which might not be suitable in the cases where the default VPC is deleted. This code base is fork from the original CloudFormer which would be launched inside a new 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"
    }
  }
}
```

Read AWS CloudFormation online: https://riptutorial.com/amazon-web-services/topic/6988/aws-cloudformation

Chapter 5: AWS Lambda

Introduction

AWS Lambda is a service that lets you run back-end code without the need to provision or manage servers. AWS Lambda takes care of scaling and high availability. The cost directly depends on how often and how long your code executes.

You will find examples of how to create and deploy AWS Lambda functions in different languages.

Remarks

- AWS Lambda code must be written in a stateless manner. While the instance of a lambda might be retained and re-used, you must never expect this.

Examples

Basic Gradle Java project

To deploy Java code to AWS Lambda, you have to create a distribution zip file that contains all dependencies that are needed during the runtime. Example project in 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
```

Running `gradle build` will create a zip file with all dependencies bundled with your code, ready to deploy.

Basic Lambda Code in Java

A lambda needs a handler, which will serve as the entry point to your application. Every handler needs to implement interface `RequestHandler<I, O>` where `I` is the input type and `O` is the output type. The input type is then passed to the `handleRequest()` method and the method returns the output type.

A simple example could look like this:

```
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;
    }
}
```

Basic Lambda code in JavaScript (NodeJs)

A lambda needs a handler, which will serve as the entry point to your application. In the simplest case, you get your input from the `context` and pass the result using the `callback()` function:

exports.handler = (event, context, callback) => { callback(null, 'You sent ' + event.number); };

Creating AWS Lambda using the web GUI (Java)

To create a Java lambda using the GUI, first make sure you have your distribution zip ready. You will also need an AWS account that can create lambdas.

- Switch to the AWS Lambda, dismiss the introduction screen (or read the Get Started documentation) and press the button `Create a Lambda Function`.
- Select an appropriate blueprint. A blueprint is an example template that can help you create your lambda. For now, start with a `Blank Function`.
- For now, don't worry about triggers and skip that screen.
- Name your function and upload the zip file.
- On the same configuration screen, type in your Handler name as fully qualified Java class with package, optionally with the method that will handle the requests. If you have used the Java interface from Basic Lambda Code example, the class name is enough.
- On the same screen, select `Create a new role from template(s)` and name your role. The role is used to give your lambda access to other AWS resources. In this case, leave it empty as we don't need any other resources.
- Select the lowest available memory (128MB). The memory influnces cost as well. Save your function.

Testing the Basic Lambda code

If you have successfully deployed the Lambda, you can also test it directly in the GUI. When you click the blue `Test` button for the first time, it presents you with a creation of a new test event.

If you are testing the Java code from the *Basic Lambda code in Java example*, delete the whole body and simply add a number `1`.

If you are testing the Javascript code from the *Basic Lambda code in Javascript (Node)* example, create a json that looks like this:

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

Run the test. You should see the result:

"You sent 1"

Adding AWS API Gateway trigger

Probably the most common way to invoke a Lambda is to use API Gateway, which maps a REST call to the lambda code. You can add multiple triggers to your Lambda during at any time, including when creating a new lambda.

If you are familiar with API Gateway, you can associate any method with a deployed lambda. Simply create a new method with Integration type set to Lambda function and point to your lambda. You can map both REST inputs to lambda inputs and outputs to REST outputs.

TODO: Decide whether to describe API Gateway here or reference another part of documentation.

Read AWS Lambda online: https://riptutorial.com/amazon-web-services/topic/8918/aws-lambda

Chapter 6: Deploy a docker container image using ECS

Remarks

Before you can add ECS instances to a cluster you must first go to the EC2 Management Console and create `ecs-optimized` instances with an IAM role that has the `AmazonEC2ContainerServiceforEC2Role` policy attached.

1. Go to your EC2 Dashboard, and click the `Launch Instance` button.
2. Under `Community AMIs`, search for `ecs-optimized`, and select the one that best fits your project needs. Any will work. Click next.
3. When you get to `Configure Instance Details`, click on the `create new IAM role link` and create a new role called `ecsInstanceRole`.
4. Attach the `AmazonEC2ContainerServiceforEC2Role` policy to that role.
5. By default, your container instance launches into your `default` cluster. If you want to launch into your own cluster instead of the default, choose the `Advanced Details` list and paste the following script into the `User data` field, replacing `your_cluster_name` with the name of your cluster.

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

6. Then, finish configuring your ECS Instance.

NOTE: If you a creating a web server you will want to create a `securityGroup` to allow access to port 80.

7. Create a Repository: `aws ecr create-repository --repository-name example-repository`
8. Authenticate your Docker client to your registry: `aws ecr get-login --region us-east-1 | sh`
9. Build your Docker image: `docker build -t example-image .`
10. Tag your image so you can push the image to this repository: `docker tag example-image:latest example-namespace/example-image:latest`
11. Push this image to your newly created AWS repository: `docker push example-namespace/example-image:latest`
12. Register an ECS Task Definition: `aws ecs register-task-definition --cli-input-json example-task.json`
13. Run the task: `aws ecs run-task --task-definition example-task`

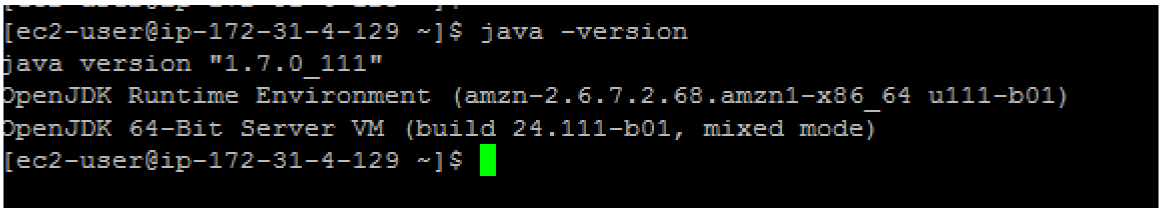
Examples

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
    }
  ]
}
```

Deploy a sample application on AWS ECS service as a proof of concept

Follow following steps to try out a sample application on AWS ECS service as a proof of concept.

1. Login to AWS management console and go to **AWS service catalog - > Compute - > Ec2**
2. Create a VM(EC2 instance) using amazon linux 64 bit OS, this we will use to configure docker, git, AWS ECS agent tool and other tools. We will also use the same VM as a node in ECS cluster to deploy container based applications. Follow below steps to create a VM.
   a) Follow usual steps to create a EC2 instance, give special embhasic on subsequent steps during EC2 instance creation.
   b) Select a IAM role with least following permissions –
   AmazonEC2ContainerServiceforEC2Role
   c) Make sure java is installed on the VM
   
```
[ec2-user@ip-172-31-4-129 ~]$ java -version
java version "1.7.0_111"
OpenJDK Runtime Environment (amzn-2.6.7.2.68.amzn1-x86_64 u111-b01)
OpenJDK 64-Bit Server VM (build 24.111-b01, mixed mode)
[ec2-user@ip-172-31-4-129 ~]$
```
3. Installing docker [execute below commands]
   first update the yum package repository

```
sudo yum update -y
```

now to install docker execute yum install

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

4. Start docker service

```
sudo service docker start
```

5. Add the ec2-user to the docker group so you can execute Docker commands without using sudo.

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

6. Log out from the EC2 and log back in again to pick up the new docker group permissions.
7. Verify that the ec2-user can run Docker commands without sudo.

```
docker info
```

8. Installing Git

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

9. Clone the sample PHP application on the Ec2 instance from git. We will use this application for our POC.

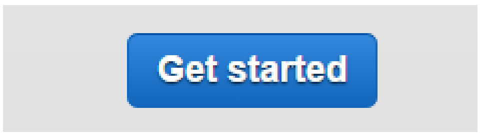
```
git clone https://github.com/awslabs/ecs-demo-php-simple-app

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

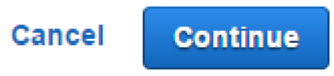
verify that Dockerfile exists by listing the directory contents

```
ls
```

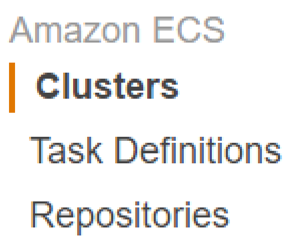
10. Go to AWS service catalog -> Compute -> Ec2 Container Service
11. Click on Get Started



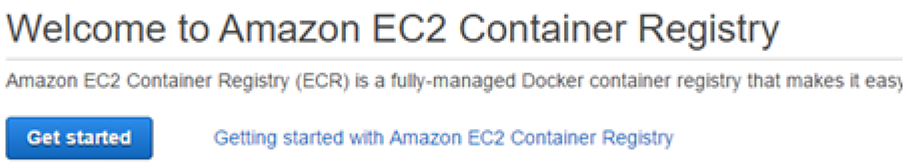
12. Click on cancel



13. Click repositories from Repositories menu in left

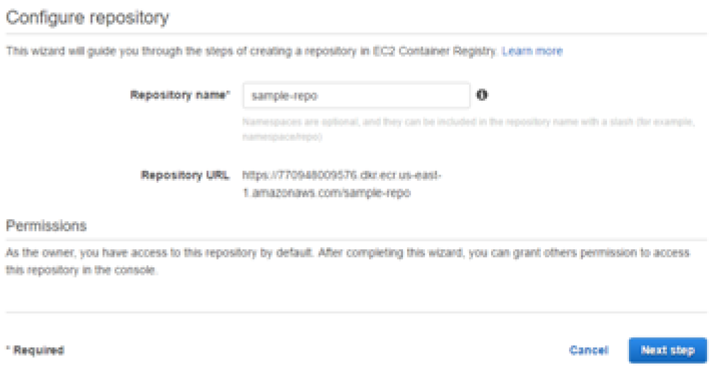


14. Click on Get Started


Welcome to Amazon EC2 Container Registry

Amazon EC2 Container Registry (ECR) is a fully-managed Docker container registry that makes it easy

15. Enter repository name and click next


Configure repository

This wizard will guide you through the steps of creating a repository in EC2 Container Registry. Learn more

Repository name* sample-repo

Namespaces are optional, and they can be included in the repository name with a slash (for example, namespace/repo)

Repository URL https://770948009576.dkr.ecr.us-east-1.amazonaws.com/sample-repo

Permissions

As the owner, you have access to this repository by default. After completing this wizard, you can grant others permission to access this repository in the console.

16. Configure Ec2 tools

```
aws configure
```

provide AWS Access Key ID, Secret Access key, default region name as per your account

17. Build, tag, and push Docker image
    a) Retrieve the docker login command that you can use to authenticate your Docker client to your registry:

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

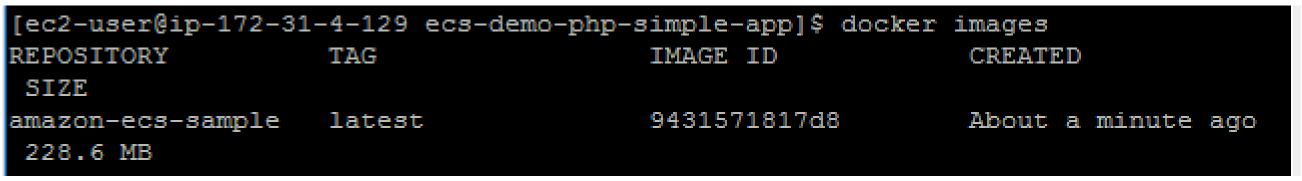
b) Run the command return as output of previous step

18. Build the Docker image from your Dockerfile. (Recall Step 9, where you downloaded a sample docker app)
    a)

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

(Note the "." stands for current directory)
b) Run docker images to verify that the image was created correctly and that the image name contains a repository that you can push your changes to the docker image

```
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
```

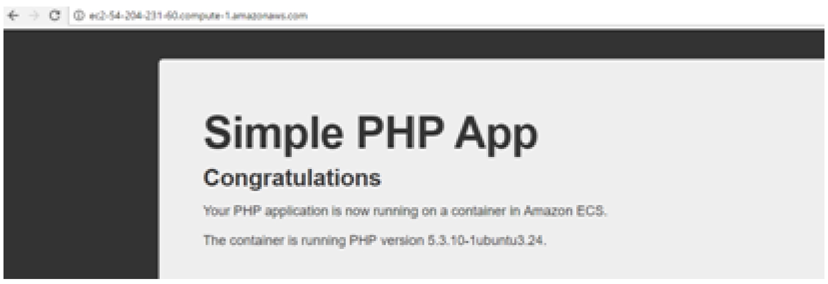
c) Run the newly built image. The -p 80:80 option maps the exposed port 80 on the container to port 80 on the host system(Ec2 instance in this case).

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

Ignore the warning “apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2 for ServerName”

19. Try to access the sample application webpage on browser, make sure port 80 is open in security groups associated with the instance

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



ec2-54-204-231-60.compute-1.amazonaws.com

Simple PHP App

Congratulations

Your PHP application is now running on a container in Amazon ECS.

The container is running PHP version 5.3.10-1ubuntu3.24.

20. Press ctrl + c key, this will stop the docker image. The sample application should not be accessible.
21. Now after successfully verifying our sample docker application, we will try to configure a cluster to run the sample application automatically. Also, for the demo purpose we will try to use the existing ec2 instance as a node in the cluster. This can be achieved by installing a agent program on the ec2 instance.
22. Installing Amazon ECS Container Agent on the ec2 instance
    a)

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

b) Restart the docker daemon

```
sudo service docker restart
```

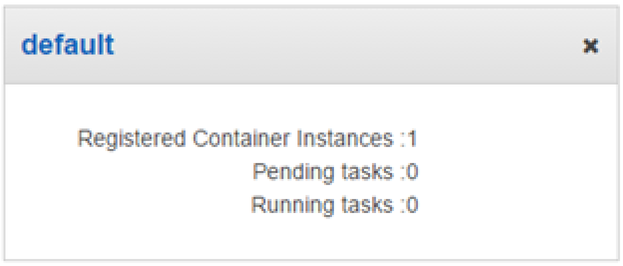
c) Start the ecs-init upstart job

```
sudo start ecs
```

d) (Optional) You can verify that the agent is running and see some information on your new container instance with the agent introspection API. Make sure the port 51678 is open in security group.

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

23. Go to AWS service catalog -> Compute -> Ec2 Container Service -> Cluster and verify a default cluster is created


default

Registered Container Instances :1
Pending tasks :0
Running tasks :0

24. Now we proceed with creating a task group and adding our docker image as task to run on the cluster
    a) Examine the simple-app-task-def.json file in the ecs-demo-php-simple-app folder.
    b) Edit the simple-app-task-def.json and redue the momeory, so that it can run on free tier eligible instance(i assume one is using free tier eligible ec2 instance for this POC, otherwise no need to reduce the memory limit)
    c) Update **memory=250** in all the occurrence on the simple-app-task-def.json file
    d) Register a task definition with the simple-app-task-def.json file.

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

e) Go to task definition in ec2 container service page, you ll find the registered task definition

f) Use the following AWS CLI command to run a task with the console-sample-app task definition.

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

g) Open the sample web app in browser, it should be accessible(refer step 19)

Thanks for reading, do share your comments and queries for follow up discussion.

Read Deploy a docker container image using ECS online: https://riptutorial.com/amazon-web-services/topic/7711/deploy-a-docker-container-image-using-ecs

Chapter 7: Elastic Beanstalk

Remarks

Current Limitations (As of 2016-10-03)

- Environment Tags cannot be changed once the *Environment* is created, so choose wisely.
- Autoscaling in Elastic Beanstalk is limited to *Simple* and *Scheduled*, so if you wish to use *Step-Scaling*, re-consider if Elastic Beanstalk is a good fit.

Automation with Jenkins

There is a great AWSEB Deployment Plugin for Jenkins that will plug-n-play for deployment to Elastic Beanstalk (blue/green deployments with automatic idle termination is just a checkbox away).

Examples

Introduction to Elastic Beanstalk

Elastic Beanstalk (EB) is essentially a hybrid between Golden AMIs and CloudFormation, while vastly simplifying the learning curve of Puppet or Chef.

An Elastic Beanstalk deployment is broken down into two components: Application and Environment.

Application

Consider this your top-level grouping, your application itself. For example, a single application ("MyWebApp") may have multiple Environments ("Production" and "Staging").

Environment

Each environment will consist of a complete architecture deployment (EC2 Instances, Elastic Load Balancer, Autoscaling Group, and Cloudwatch Alarms). The entire environment configuration is setup and maintained for your automatically.

Deploying an Application

Your application deployment is as simple as uploading a zip file containing your code. Each zip file (called *Application Version*) you upload is associated to an *Application*, so you can upload once and deploy to multiple *Environments*.

Customizing the Environment

By default, Elastic Beanstalk will deploy "stock" Amazon-Maintained AMIs. For most applications, this is sufficient, but there may be environmental tweaks that you want to make (eg. changing the timezone, adding packages/dependencies not present in the code, etc).

There are two ways of customizing the EB AMI that is used: ebextensions or a custom AMI.

ebextensions - A folder, quite literally called '.ebextensions', that can optionally be placed at the root of your *Application Version* (the zip you uploaded containing your code). Within the ebextensions folder, you can place YAML files defining any custom scripts, dependencies, etc that you want executed server-side during the deployment process. There are a number of hooking points available, for the latest information, please check the relevant documentation: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html

---

Gotchas / Things to be aware of

VPC Checkbox - When creating an environment, the option is discretely made available as to whether or not the environment should be created/placed within a VPC. If you need your application to communicate with existing resources that you have created, CHECK THIS BOX. Otherwise, Elastic Beanstalk will create a new security group that is isolated from the rest of your VPC. *While you will be able to manually adjust the security group settings after creation, trying to essentially 'add' it into a VPC will cause a variety of problems later on.*

RDS - When creating an environment, you have the option to create an RDS instance as part of the environment. It is not recommended to use this, as anytime you need to 'rebuild' the environment (eg. blue/green deployments, troubleshooting) it will destroy and recreate the RDS instance (along with all data).

Blue/Green Deployments on Elastic Beanstalk

Blue/Green deployment is a release technique that reduces downtime and risk by running two identical production environments (one called "Blue", the other called "Green"). At any one time, only one of the environments is serving live traffic, while the other is sitting idle.

When deploying a new version, the code is deployed to the idle environment (eg. Green) and after confirming a successful deployment, live traffic is switched (eg. Green has new code, traffic from Blue is now routed to Green). The next code deploy will occur on the new idle environment (following the example, that would now be Blue).

Example/Visual Aid:

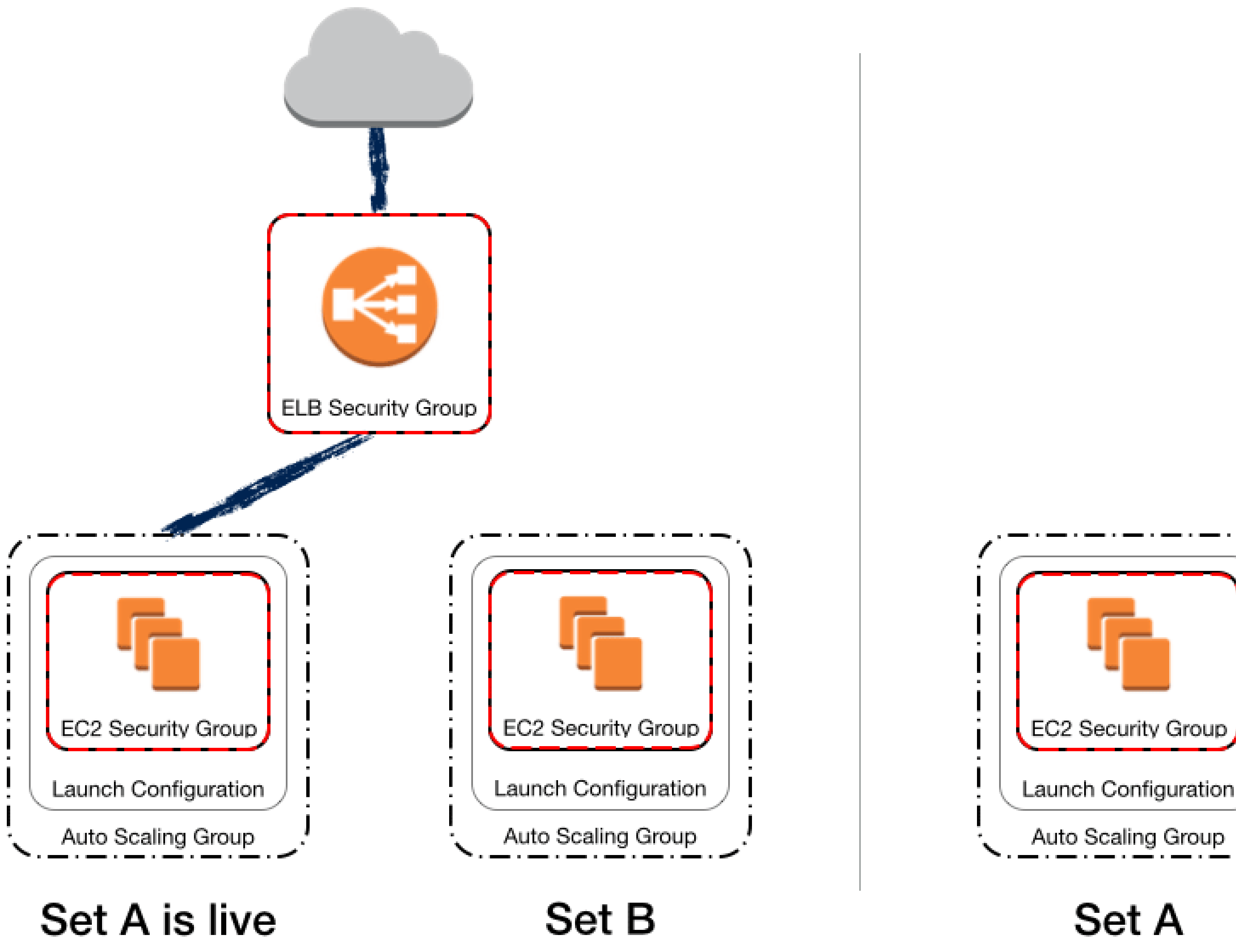


image source: https://cloudnative.io/statics/blog/aws-blue-green-deployment.png

---

When using Elastic Beanstalk (EB), you can easily create multiple environments that are exact clones of one another for code deployment. After confirming that the new environment is ready to go live, it is as simple as using the "Swap URLs" function in EB to swap environments.

Step-by-step instructions: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.CNAMESwap.html

Read Elastic Beanstalk online: https://riptutorial.com/amazon-web-services/topic/7207/elastic-beanstalk

Chapter 8: Root Class

Examples

Amazon api root class is as following.

```
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; }
    }
```

Business class

```
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#



        }
    }
```

Read Root Class online: https://riptutorial.com/amazon-web-services/topic/7357/root-class

Credits

S. No	Chapters	Contributors
1	Getting started with 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	Deploy a docker container image using ECS	mrded, Satish
7	Elastic Beanstalk	Lorenzo Aiello
8	Root Class	vicky