Media Sharing Website
Part 1: Media Uploads

Table of Contents

Introduction

The objective of this lab is to create a simple media sharing website. In this first part, we will create the core architecture of the system, providing basic features such as browsing, uploading and deleting content. Media content will be limited to images, but the concepts covered here also apply to other types of media such as documents (PDF, RTF, presentations, etc.), music and videos.

In this first part, we will create a system that provides a web interface for users to browse and store images.

Start your *qwikLAB*™

1. Start your *qwikLAB*™
   Use the 'Start Lab' button to start your lab.
   (Hint: If you are prompted for a token, please use one you purchased or were given.)

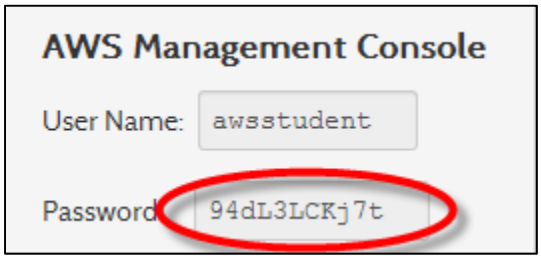
   You will see the lab creation in progress.

2. Note a few properties of the lab.

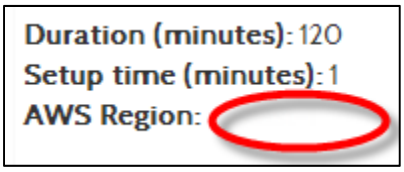
   a. **Duration -** The time the lab will run for before shutting itself down.
   b. **Setup Time -** The estimated lab creation time on starting the lab.
   c. **AWS Region** - The AWS Region the lab resources are being created in.

3. Copy the Password provided.

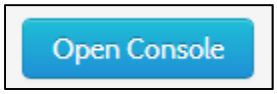
   a. Hint: selecting the value shown and using Ctrl+C works best



4. Note the AWS Region set for your lab in *qwikLAB*™:



5. Click the 'Open Console' button.



6. Make sure that you are not logged into any other instances of the AWS console (in a student account or your own account), as this may cause conflicts when you open the console and log in below for this lab.

7. Login to the AWS Management Console

   Enter the User Name '**awsstudent**' and paste the password you copied from the lab details in *qwikLAB*™ into the Password field.

   Click on the 'Sign in using our secure server' button.

   In this step you logged into the AWS Management Console using login credentials for a user provisioned via AWS Identity Access Management in an AWS account by *qwikLAB*™.

Step 1 – Media storage

We shall be creating a system for uploading and storing image files.

We could store these images on *EBS volumes* (virtual disks attached to Amazon EC2 instances) but this would involve provisioning capacity in advance and manually scaling-up storage by adding volumes. Also, these volumes would need to be attached to an Amazon EC2 instance to serve the content via HTTP. This creates a single point of failure in the system unless the data is replicated and served from another instance.

A better approach is to use **Amazon Simple Storage Service (S3)** as storage repository for our media files. Amazon S3 provides a high durability of data and the ability to serve content directly via HTTP. It also saves on Amazon EC2 capacity.

Files in Amazon S3 are called **objects** and they are stored in **buckets**. There is no limit to the number of objects that can be stored in a bucket and there is no variation in performance whether you use many buckets or just a few. You can store all of your objects in a single bucket or you can organize them across several buckets.

For more information about Amazon S3, visit:
http://docs.amazonwebservices.com/AmazonS3/latest/dev/Introduction.html

Select the Amazon S3 Service

1. Select "**S3**" from the Console Home:

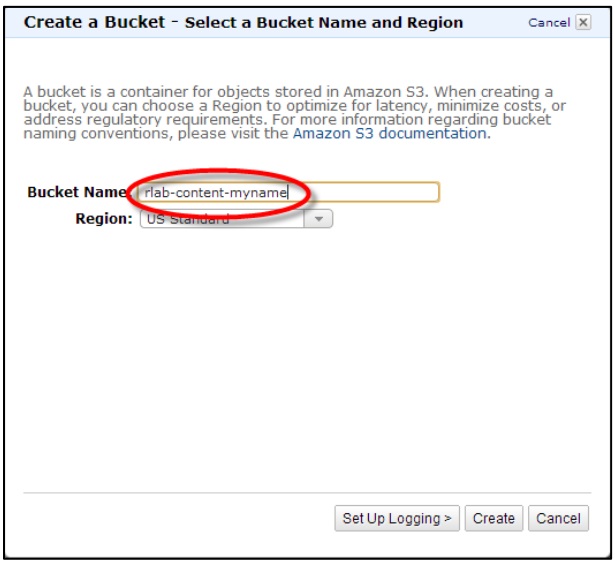
Confirm your AWS Region

2. This lab requires the **US East Region** (also known as **US Standard**).

Creating an Amazon S3 Bucket

3. Click the **Create Bucket** button:

4. Type a name for your bucket:



Your bucket name must be lowercase and at least 3 characters long. (For more naming rules, see Bucket Restrictions and Limitation):

http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html

Bucket names must be unique across all of Amazon S3. In case the name you are using is already in use, please provide a different option. Within a bucket, you can use any names for your *objects*.

5. **Write down the name of your bucket** as it will be required in future steps.
6. Set the region to **US Standard**.
7. Click the **Create** button to create your bucket.

Assign a Bucket Policy

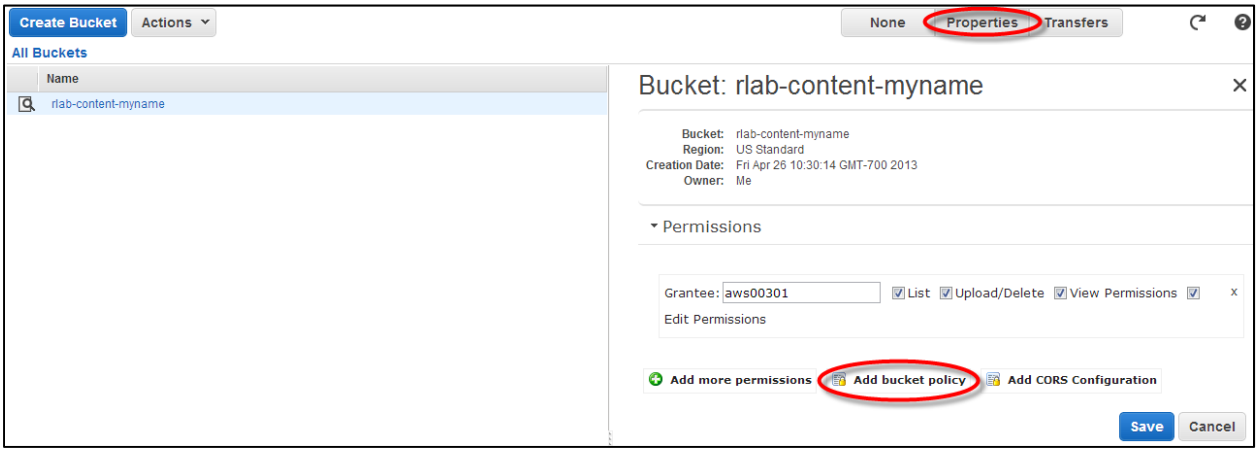
By default, content stored in Amazon S3 buckets cannot be accessed publicly. For this lab, we need to allow public access to the bucket to permit users to download content. While we want to allow *downloads* from the bucket, we do not want to permit users to *list* the contents of the bucket or *delete* any objects.

Amazon S3 manages access to buckets and objects using **bucket policies**. Bucket policies are a collection of JSON statements written in an *access policy language*. For more information about bucket policies, visit:

http://docs.amazonwebservices.com/AmazonS3/latest/dev/UsingBucketPolicies.html

To add a policy to your bucket:

8. Click on your bucket name
9. Click **Properties** (in the top-right)
10. Click on **Permissions** to show more options
11. Click **Add bucket policy**



12. In the **Bucket Policy Editor** dialog box, add the following policy to your bucket:

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AddPerm",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME/*"
    }
  ]
}
```

This code is also available from:
https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-11/lab11-bucket_policy.json

13. **Replace** `YOUR_BUCKET_NAME` with the name of your bucket.

Your policy should look like this:

14. Click **Save** to apply the policy to your bucket.

Step 2 – Media Database

When users upload an image, we will need to store metadata such as the title, comment, publication date and entry type. We need a database for storing this information. Since we don't know how many entries our system will have to contain, and since we want our system to be scalable, we will use **Amazon DynamoDB** to store the metadata information.

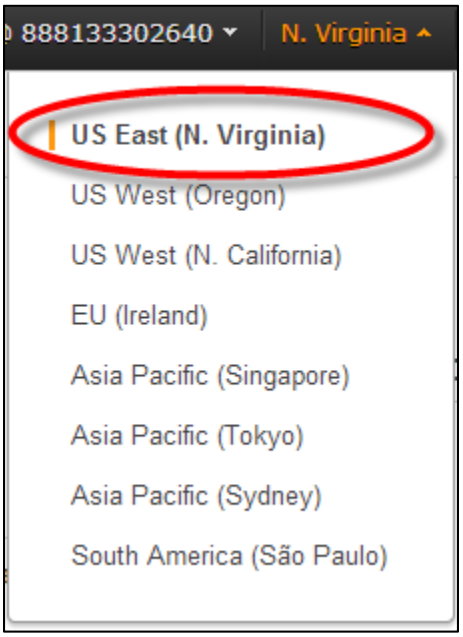
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. With a few clicks in the AWS Management Console, you can launch a new Amazon DynamoDB database table, scale up or down the request capacity for the table without downtime or performance degradation, and gain visibility into resource utilization and performance metrics. Amazon DynamoDB enables you to offload the administrative burdens of operating and scaling distributed databases to AWS, so you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. For more information about Amazon DynamoDB, visit:

http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/Introduction.html

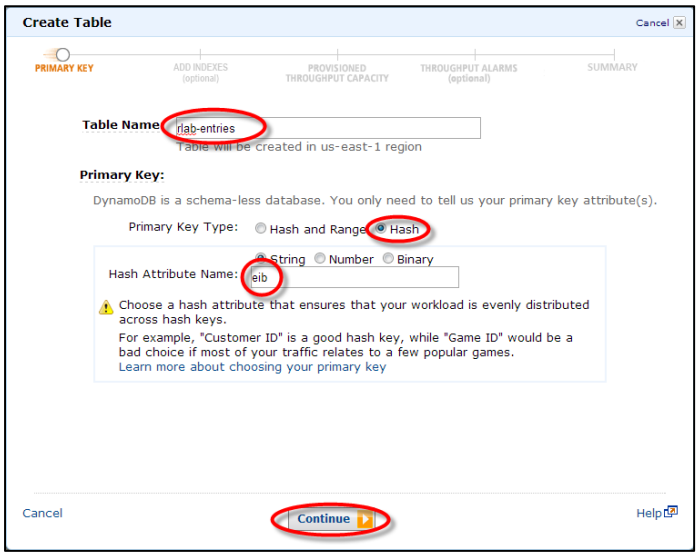
Creating an Amazon DynamoDB table

1. Click **Services** at the top-left of the window and select **DynamoDB**:

2. In the top-right of the screen (next to **Help**), ensure that the AWS Region is set to **US East (N. Virginia)**:



3. Click the **Create Table** button.
4. **Choose your own name** for the table. It must be a unique name, so you might have to try a few times to find a valid name. No spaces are permitted. **Write down the name of your table** as it will be required to complete future steps.
5. Set the **Primary Key Type** to **Hash, String.**
6. Set the **Hash Attribute Name** to: `eib`



Information about the DynamoDB data model is available at:

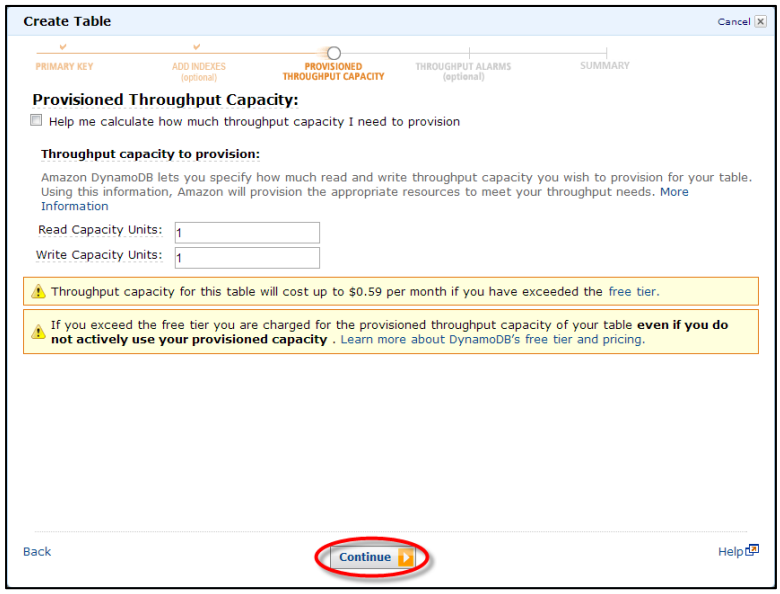
http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/DataModel.html

7. Click **Continue** to create the table.
8. On **Add Indexes (optional)**, click **Continue**.

Provisioned Throughput Capacity

When creating an Amazon DynamoDB table, you specify how much *capacity* you wish to reserve for reads and writes. Amazon DynamoDB will then reserve the necessary machine resources to meet your throughput needs while ensuring consistent, low-latency performance.

9. For this lab, the default settings will be enough, click **Continue**.

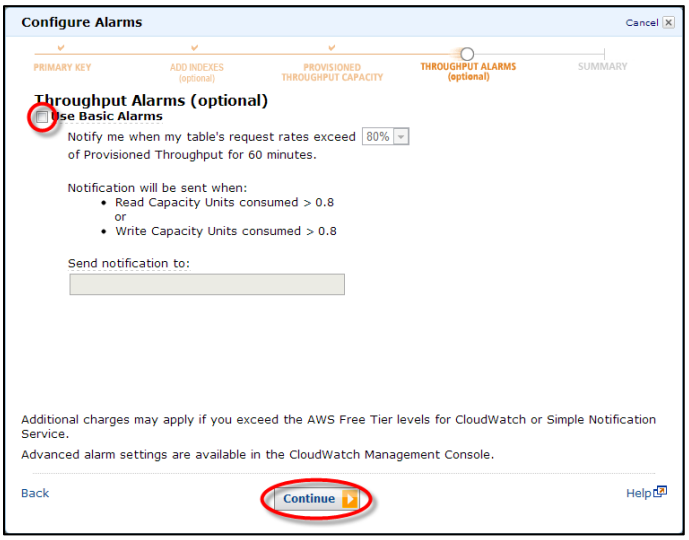


Throughput Alarms

In production systems, alarms warn you when your table's throughput capacity needs adjusting. For more information about Amazon DynamoDB provisioned throughput model, visit:

http://docs.amazonwebservices.com/amazondynamodb/latest/developerguide/ProvisionedThroughputIntro.html

10. **Uncheck "Use Basic Alarms"** and click **Continue**.



11. On the **Review** screen, click **Create**.

The table status will show as **CREATING**. The table creation process may take a few minutes:

Step 3 – Access Control

So far, we created two resources on AWS: an Amazon S3 Bucket and an Amazon DynamoDB table. To access them, we need specific **security credentials**.

We could use AWS Account Credentials to access those resources, but such practice is strongly discouraged in production systems since AWS Account Credentials have full access to all AWS resources.

A better approach is to use **AWS Identity and Access Management (IAM)** to securely control access to Amazon Web Services and your account resources. AWS IAM can assign **roles** to Amazon EC2 instances, which makes it easy to securely access AWS service APIs from Amazon EC2 instances. The normal process is to create an IAM role, assign it a set of permissions and launch Amazon EC2 instances with the IAM role. The Amazon EC2 instances then have access to their own AWS access keys with specific limited permissions.

For more information about AWS IAM, visit:

http://docs.amazonwebservices.com/IAM/latest/UserGuide/Welcome.html

A short video explaining IAM roles for Amazon EC2 instances is available here:

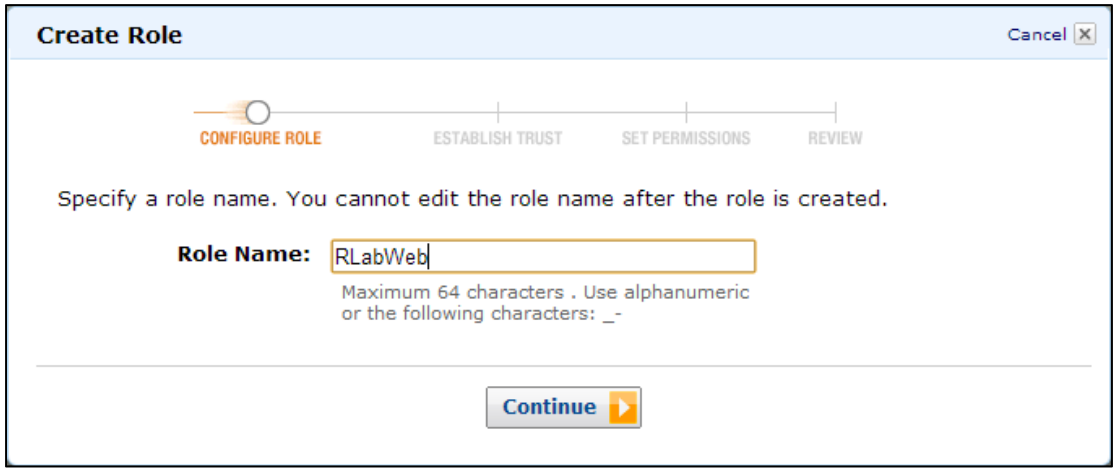
http://www.youtube.com/watch?v=XuRM4Id6uDY

Creating an IAM role

1. Open the AWS IAM in the console:

2. Select the **Roles** (on the left) and click the **Create New Role** button:

3. Enter `RLabWeb` as the Role Name and click **Continue**:

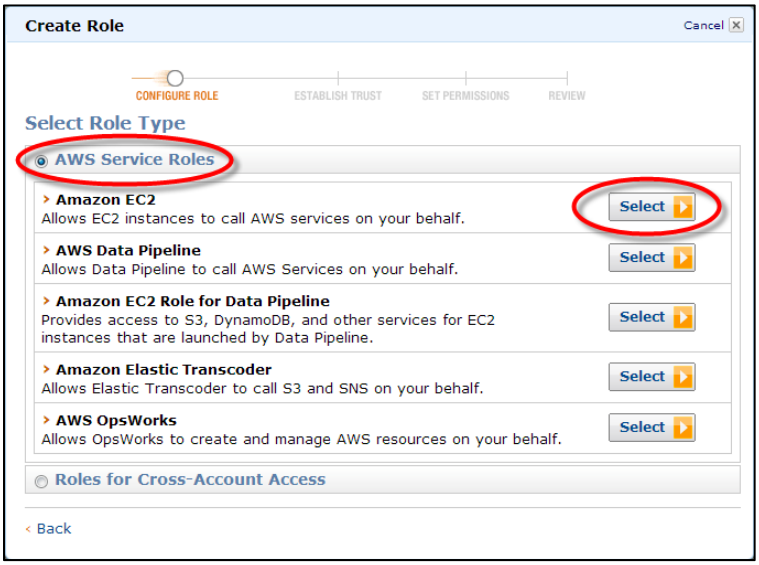


Select Role Type: We can now specify a *policy* describing what actions are allowed for this role. A policy is a document that formally states one or more permissions. The distinction between a *permission* and a *policy* is important. To give a particular IAM role a *permission*, you write a policy using the IAM access policy language, then attach the policy to the desired role (a particular user or group in your AWS account). You do not actually specify the entity in the policy itself; the act of *attaching* the policy to the role grants it the permission stated in the policy.

Policies are described in JSON format. For information about the access policy language, see

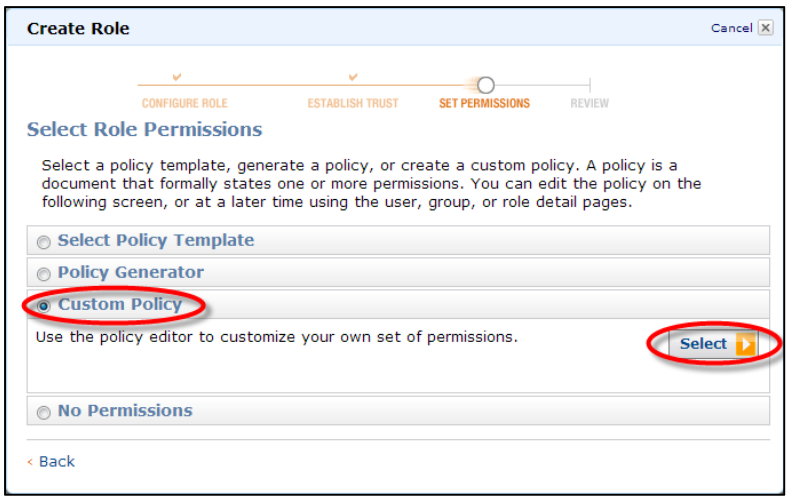
http://docs.amazonwebservices.com/IAM/latest/UserGuide/AccessPolicyLanguage.html

4. On **Select Role Type**, select **Amazon EC2**:



Select Role Permissions: A policy can be selected by using a template, using the policy generator, or by providing a custom policy.

5. For this lab, we provide the policy below, so chose **Custom Policy** and click **Select**:

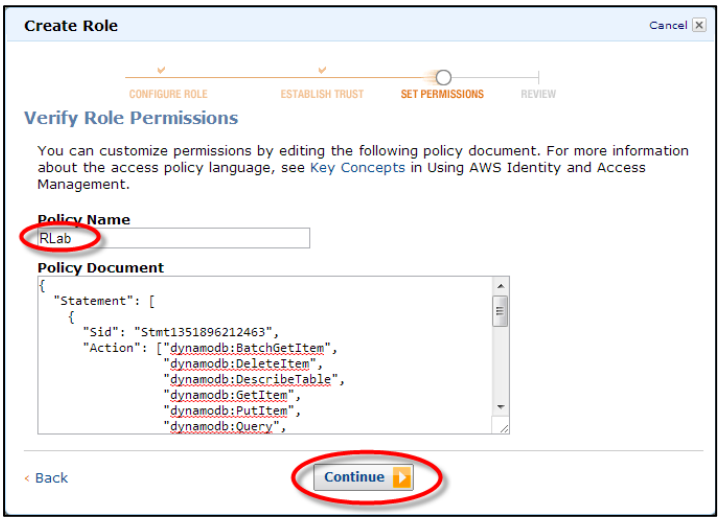


Verify Role Permissions: The web server needs to be able read, write, delete and list objects in the S3 bucket. The web server also needs to query, scan, read, write and delete items in the DynamoDB table.

6. In **Policy Name**, enter `RLab`.

Copy the policy shown on the next page into the **Policy Document** section to set appropriate permissions, and replacing the highlighted strings by your own values:

7. **Replace** `YOUR_ACCOUNT_ID` with the number shown in the top-right of your AWS Web Console. It is a 12-digit number.
8. **Replace** `YOUR_BUCKET_NAME` (twice) with the name you chose in Step 1 of this lab.
9. **Replace** `YOUR_TABLE_NAME` with the table name you chose in Step 2 of this lab.
10. The region has already been entered into the policy as `us-east-1`

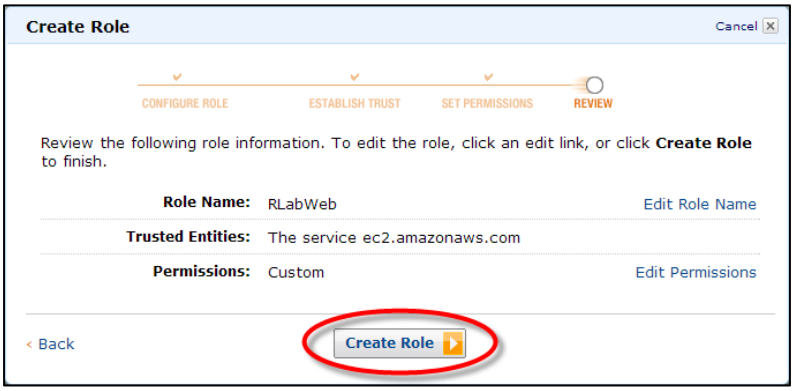



```
{
  "Statement": [
    {
      "Sid": "Stmt1351896212463",
      "Action": ["dynamodb:BatchGetItem",
                                "dynamodb:DeleteItem",
                                "dynamodb:DescribeTable",
                                "dynamodb:GetItem",
                                "dynamodb:PutItem",
                                "dynamodb:Query",
                                "dynamodb:Scan",
                                "dynamodb:UpdateItem"],
      "Effect": "Allow",
      "Resource": "arn:aws:dynamodb:us-east-1:YOUR_ACCOUNT_ID:table/YOUR_TABLE_NAME"
    },
    {
      "Sid": "Stmt1351896363046",
      "Action": ["s3:DeleteObject",
                                "s3:GetObject",
                                "s3:PutObject",
                                "s3:ListBucket"],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::YOUR_BUCKET_NAME/*", "arn:aws:s3:::YOUR_BUCKET_NAME"
      ]
    }
  ]
}
```

The script is also available from:

https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-11/lab11-role_policy.json

11. Click **Continue**.
12. Click **Create Role** to complete the role creation process:

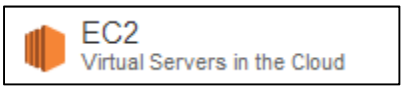


Step 4 – Web Front-End

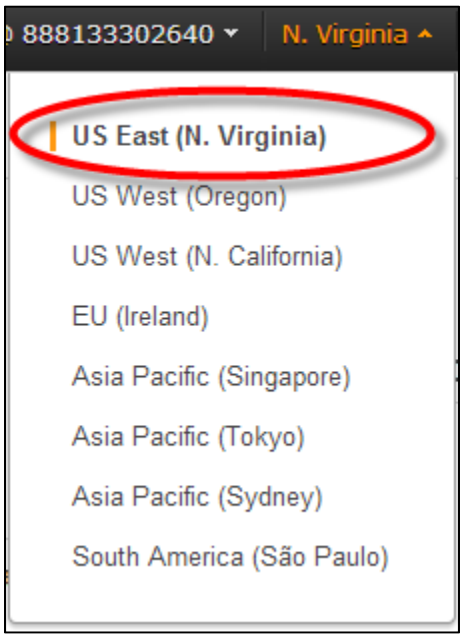
You have now everything ready to create the web server. You will use an Amazon Elastic Compute Cloud (EC2) instance to host the web server. The web application used for this lab is already packaged in an Amazon Machine Image (AMI).

Deploying the web server

1. Open the Amazon EC2 console:

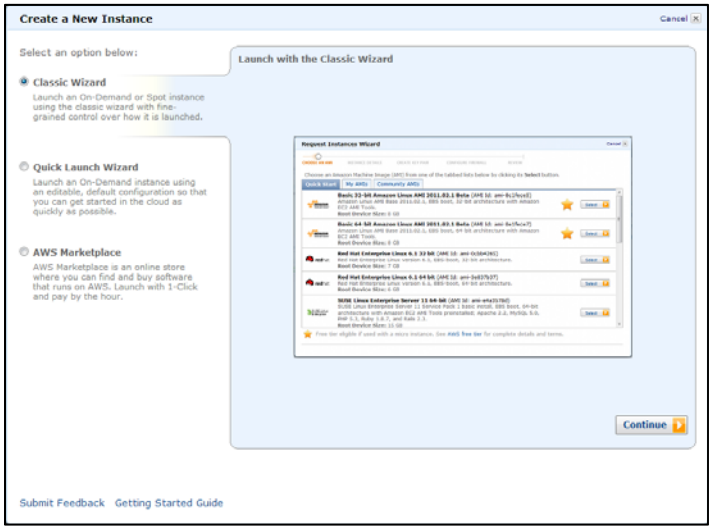


2. In the top-right of the screen (next to **Help**), ensure that the AWS Region is set to **US East (N. Virginia)**:

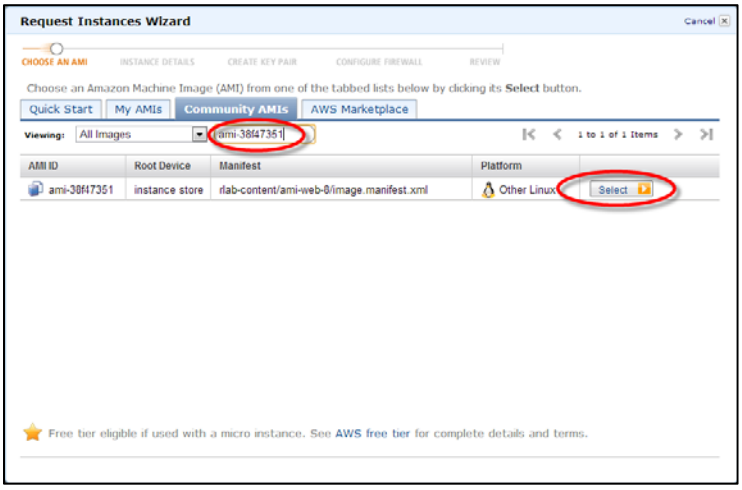


3. Click the **Launch Instance** button:

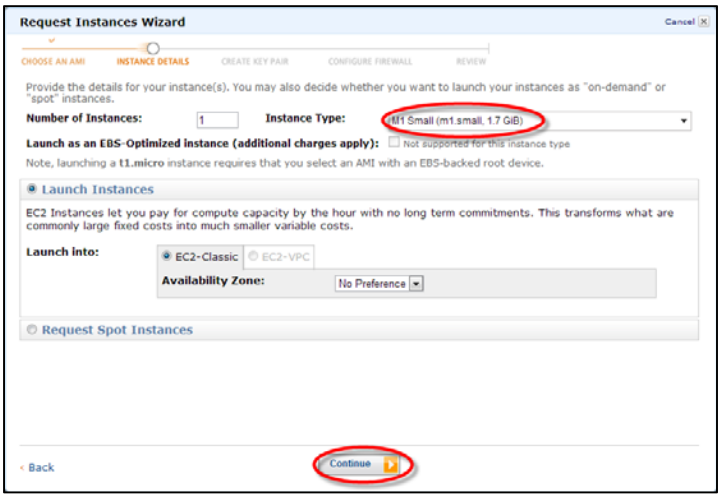
4. The **Classic Wizard** will already be selected, so click **Continue**:



5. Select the **Community AMIs** tab.
6. In the search box, enter: **`ami-38f47351`** and click **Select**. If you copied and pasted the ami code into the search box and the AMI isn't found, try deleting then retyping the dash in the AMI code (PDFs sometimes have issues with copying and pasting dashes). Then search again. If you get a JSON error, please wait a few moments, and retry your search.



7. Leave the Instance Type as **M1 Small**.
8. Click **Continue**:



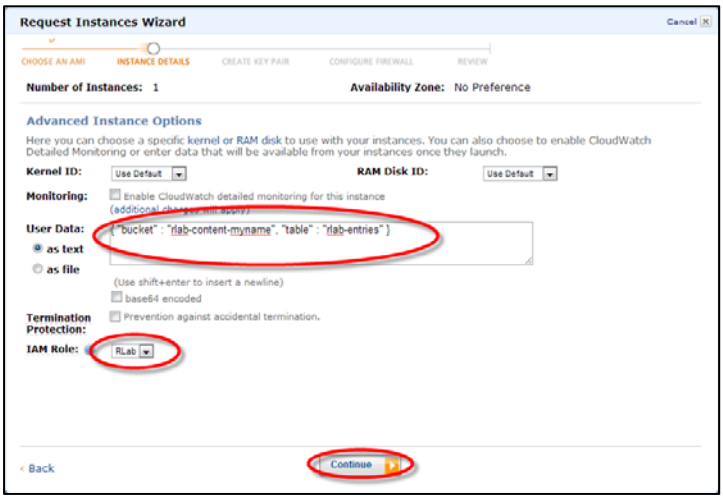
Next, we will specify some **User Data** to pass to the instance. Amazon EC2 instances can access instance-specific metadata, as well as data supplied when launching the instances.

9. **Copy this JSON data** into the **User Data** section:

```
{ "bucket" : "YOUR_BUCKET_NAME", "table" : "YOUR_TABLE_NAME" }
```

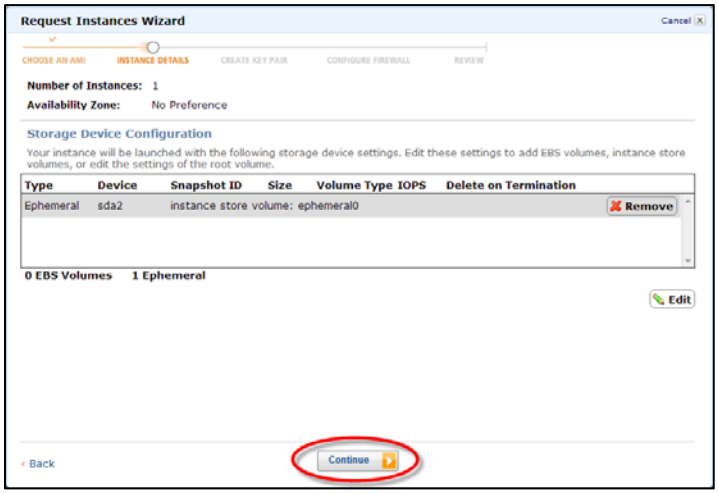
Replace the highlighted strings by your own values:

10. **Replace** `YOUR_BUCKET_NAME` with the name you chose in Step 1 of this lab.
11. **Replace** `YOUR_TABLE_NAME` with the table name you chose in Step 2 of this lab.
12. Set **IAM Role** to **`RLabWeb`** (which we created previously).
13. Click **Continue**:



As we don't need specific storage for the web server, you can ignore the Storage Device Configuration and tagging section.

14. Click **Continue** twice:



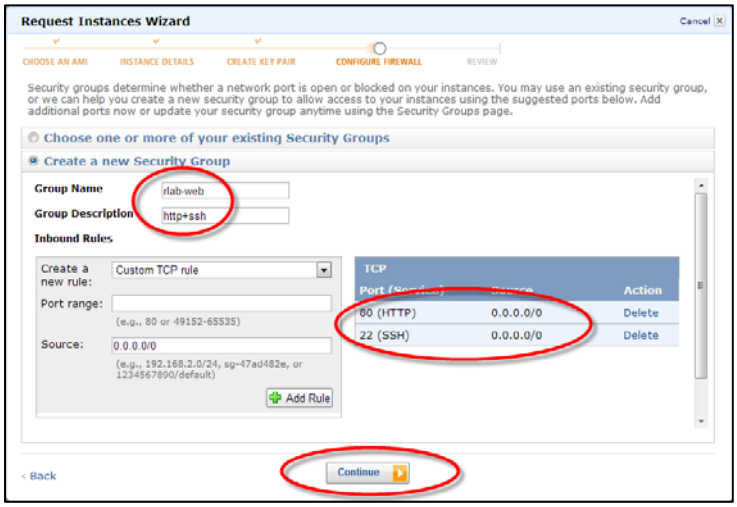
15. Write down the name of the existing Amazon EC2 Key Pair *qwik*LAB™ created for you.
    Your will need this later.
    Click **Continue:**

The next step is to configure **firewall settings** for this instance. Amazon EC2 provides **Security Groups** that act as a firewall to control what traffic is allowed to reach instances. You can add **rules** to each security group that control the inbound traffic allowed to reach the instances associated with the security group. All other inbound traffic is discarded.

The instance we're creating is a web server so we need to assign a rule for inbound HTTP traffic. We will also want to access the instance via SSH.

16. Click **Create a new Security Group**.
17. Enter any **Name** and **Description** you wish.

18. HTTP Rule: In the **Create a new rule** pull-down, select **HTTP** and the click **Add Rule**.
19. SSH Rule: In the **Create a new rule** pull-down, select **SSH** and the click **Add Rule**.
20. Click **Continue**:



21. Review your instance settings, then click **Launch** and **Close**.

The console will show the instance as **pending**. Once ready, the instance status will switch to **running**:

Testing the deployment

22. To test the deployment, **click on the row** shown above and copy the **public DNS** of the instance. It will look like the URL shown below:

23. **Paste the URL** into a new tab of your web browser.

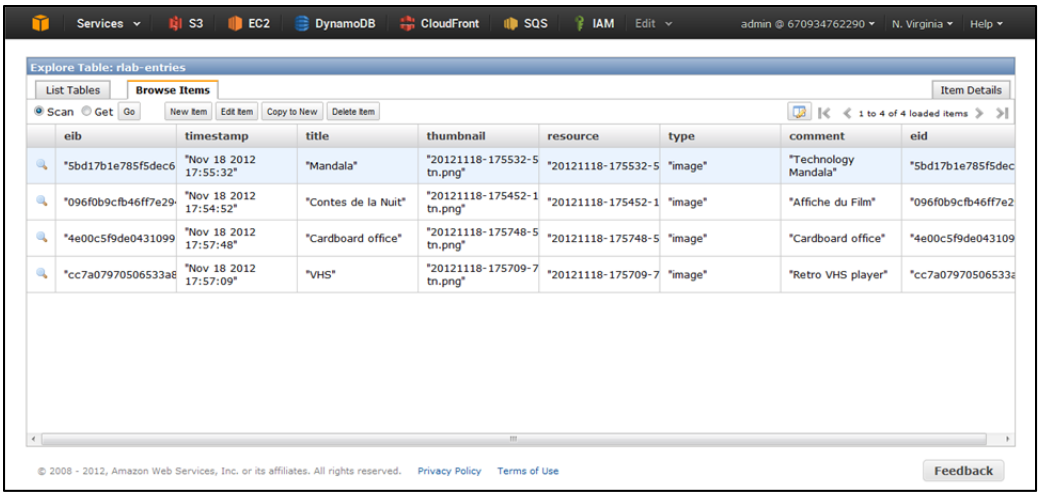
You should see the home page of the web application:

(If you received an “Internal Server Error”, then the User Data configuration may have been wrong. You will need to terminate the Amazon EC2 instance and create it again.)

24. **Add a couple of images and play around with the application**. Note: You will need to provide both the Title and Comment when uploading images.

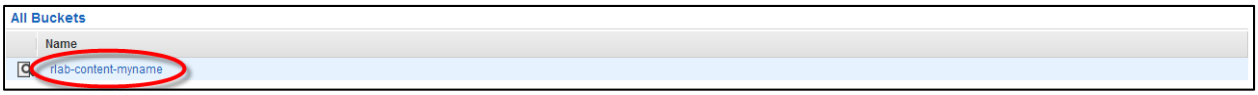
25. Once you added a couple of images in the web application, go back to the AWS Web Console and open **DynamoDB**.
26. Click on your table and click the **Explore Table** button:

Click the **Browse Items** tab. You should see your content metadata as stored by the web application:



You can also check how media files have been stored in the S3 Bucket.

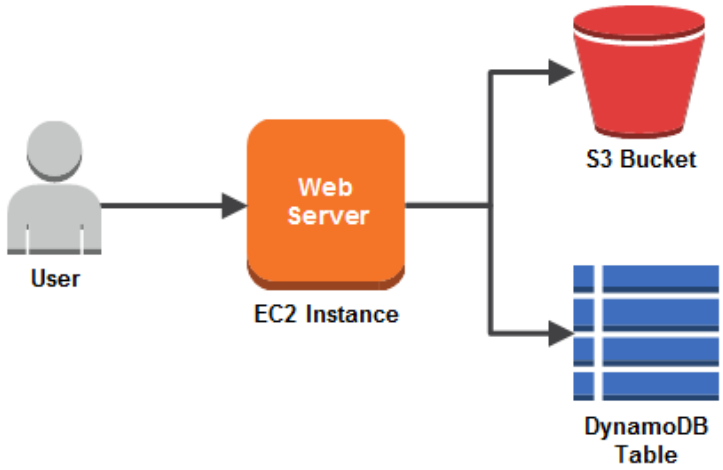
27. In the AWS Management Console, open **S3.**
28. **Click the bucket** you created for this lab:



You will see the list of image files that the application uploaded to the bucket, together with thumbnail images (with the “-tn.png” extension):

Architecture Overview

So far, we created the following system:



When the user uploads an image, the **web server** receives it and creates a thumbnail. It will then upload the image and the thumbnail to the **S3 bucket** and insert the image metadata in the **DynamoDB table**.

While Amazon S3 and Amazon DynamoDB are both scalable and fault-tolerant systems, our web server running on a single Amazon EC2 instance is clearly a **single point of failure** (e.g. if the web application fails, the system is not accessible and cannot recover) and a **bottleneck** (e.g. with an important load of incoming requests, the system might even become unavailable). Both issues will be fixed in the next section.

Step 5 – Scalable Architecture Deployment

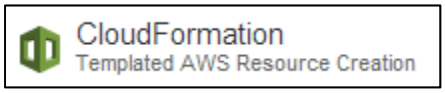
To solve both the scalability and the single point of failure issues, we will use **Auto Scaling**. Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define, thus removing the bottleneck problem.

However, with multiple web servers in the front-end, you will also need to distribute incoming HTTP traffic to them. **Elastic Load Balancing** (ELB) automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored, or replaced by Auto Scaling.

To deploy the new front-end, you will use an **AWS CloudFormation** script which will create the Auto Scaling groups and Elastic Load Balancers for you. AWS CloudFormation enables you to create and delete related AWS resources together as a unit called a **stack**. You define the characteristics of a stack parameters, mappings, resource properties, and output values using a JSON template.

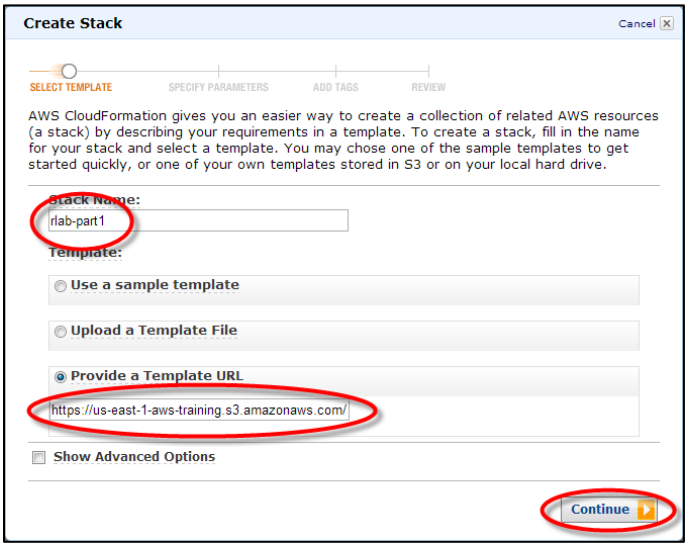
Deployment using CloudFormation

1. In the AWS Console, open **CloudFormation**:



2. Ensure that the Region (top-right of window) is still set to **US East**.
3. Click **Create New Stack**.
4. Provide a **Stack Name** of your own choosing.
5. Click **Provide a Template URL** and enter this as the URL for the template:

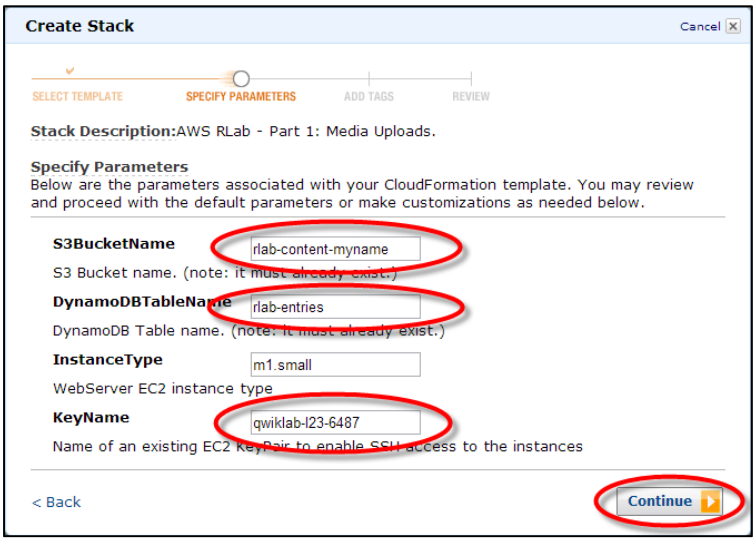
https://us-east-1-aws-training.s3.amazonaws.com/self-paced-lab-11/rlab-part1-cfn.template



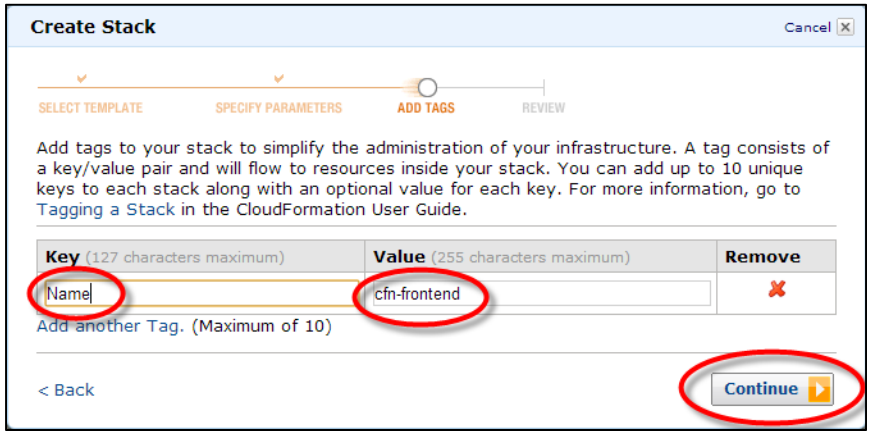
6. Click **Continue**.

The stack template contains *parameter placeholders*. Please enter:

7. Your **S3 Bucket Name** (from Step 1)
8. Your **DynamoDB Table Name** (from Step 2)
9. The **Key Pair name** you used when creating the EC2 instance (in Step 4):

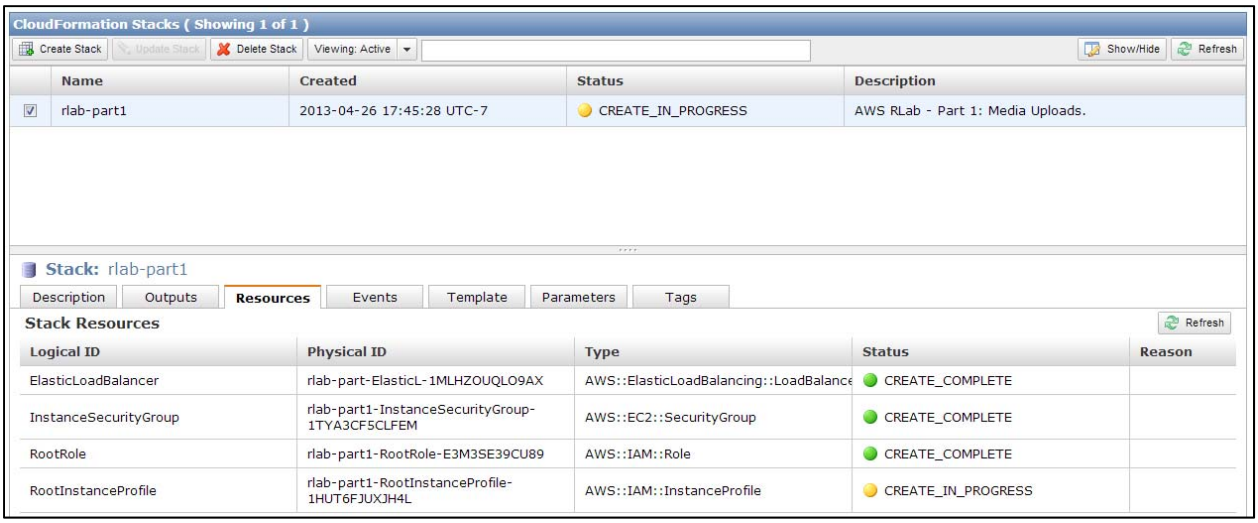


10. Click **Continue**.
11. You can provide some tags which will be assigned to the resources created by this stack:

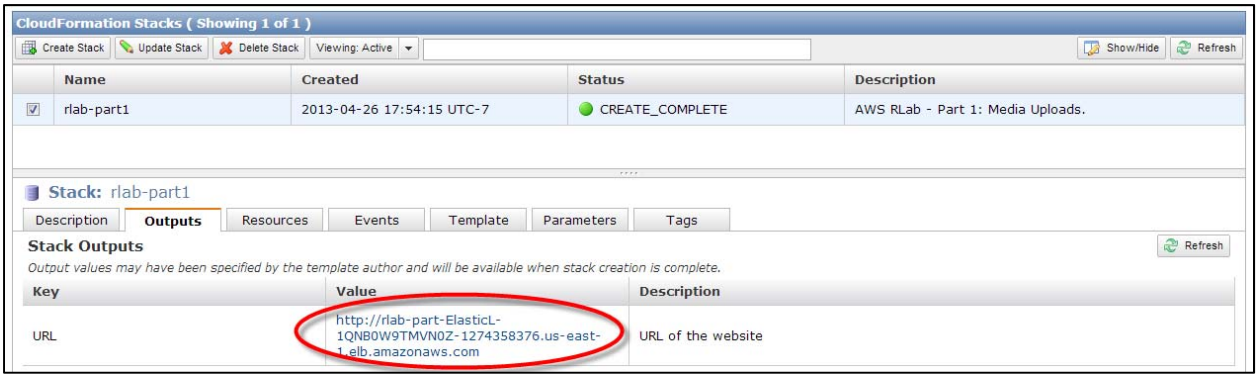


12. Finally, review your stack parameters, then click **Continue** and **Close**.

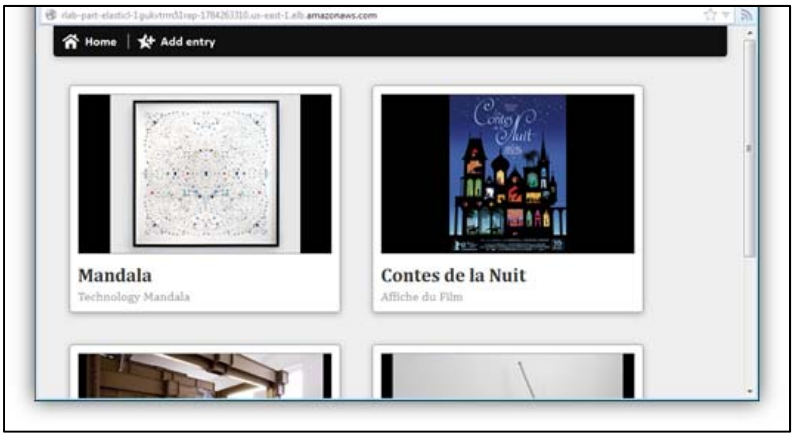
The stack creation process will take a couple of minutes. You can follow the stack creation process by checking the **Resources tab**:



13. Once the stack creation is completed, the **Outputs tab** will show the URL endpoint of the Elastic Load Balancer. Note: If your status says “Rollback_Complete” scroll through the items in the Events tab. In the right column, you should be able to see what went wrong (i.e. “The key pair 'qwiklab-l2308471' does not exist” or some other error).

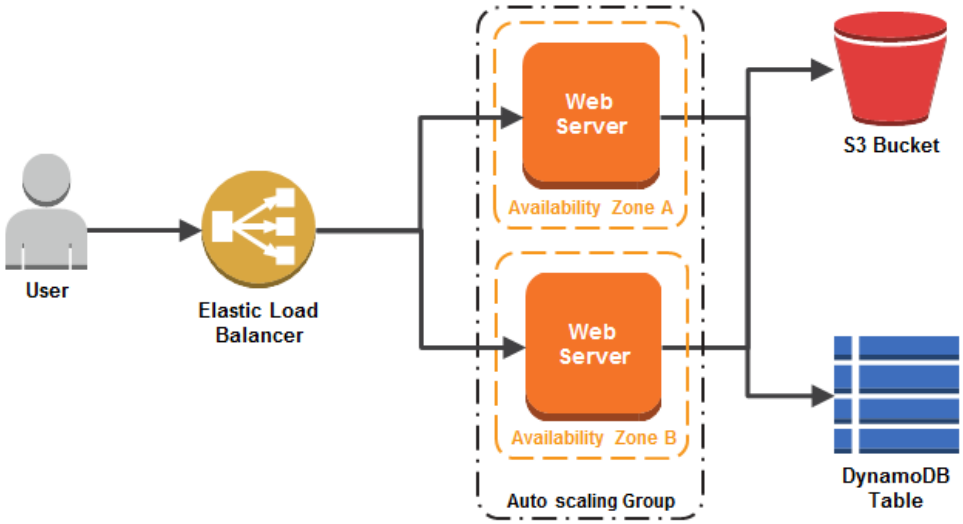


14. **Copy and paste this URL into a new tab** of your web browser. You should see the web application home page, which is now being access via the Elastic Load Balancer:



Final Architecture Overview

The following diagram represents the final architecture you built in this lab:



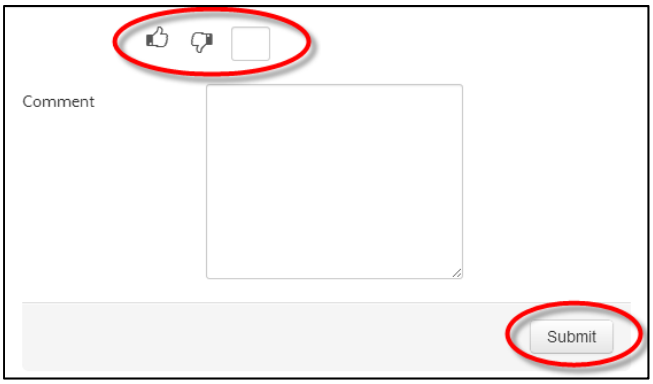
The front end is now as fault-tolerant and scalable as the data storage systems. In case of increased inbound traffic, the Auto Scaling group will **automatically provision new instances**, and **replace instances** in case of failure.

End Lab

This concludes the lab. The topic continues in *Lab 12 Part 2: Video Transcoding*.

1. **Sign out** of the AWS Management Console.
2. Click the **End Lab** button in *qwikLAB*™:

We appreciate your feedback. Please give the lab a thumbs-up/down, enter a comment and click **Submit**:



Errors in these lab directions can be reported to aws-course-feedback@amazon.com.

