Building a Media Sharing Website - Part 1: Media Upload {.headline-5}

Building a Media Sharing Website - Part 1: Media Upload

Lab Overview

In this lab, you will learn how to deploy a photo sharing website using Amazon S3 for storage, Amazon DynamoDB for the database, Amazon EC2 to host a web application.

In this lab, you 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, videos, etc. The system will also provide a web interface for users to browse and store images.

Topics Covered

By the end of this lab, you will be able to:

- Create a new Amazon S3 bucket to hold your media files
- Create a security group to restrict access to the server's resources
- Launch a new Amazon EC2 instance to run your web server
- Create an Amazon DynamoDB database to hold your data

Lab Pre-requisites

To successfully complete this lab, you should be familiar with basic navigation of the AWS Management Console and be comfortable editing scripts using a text editor.

Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.\
Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use.

Amazon EC2 enables you to increase or decrease capacity within minutes,

not hours or days. You can commission one, hundreds or even thousands of server instances simultaneously. Of course, because this is all controlled with web service APIs, your application can automatically scale itself up and down depending on its needs.

You have complete control of your instances. You have root access to each one, and you can interact with them as you would any machine. You can stop your instance while retaining the data on your boot partition and then subsequently restart the same instance using web service APIs. Instances can be rebooted remotely using web service APIs. You also have access to console output of your instances.

You have the choice of multiple instance types, operating systems, and software packages. Amazon EC2 allows you to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for your choice of operating system and application. For example, your choice of operating systems includes numerous Linux distributions, and Microsoft Windows Server.

DynamoDB

DynamoDB is a fast, fully managed NoSQL database service that makes it simple and cost-effective to store and retrieve any amount of data, and serve any level of request traffic. Its guaranteed throughput and single-digit millisecond latency make it a great fit for gaming, ad tech, mobile and many other applications.

DynamoDB delivers seamless throughput and storage scaling via API and easy-to-use management console, so you can easily scale up or down to meet your needs. Many of our customers have, with the click of a button, created DynamoDB deployments in a matter of minutes that are able to serve trillions of database requests per year.

DynamoDB tables do not have fixed schemas, and each item may have a different number of attributes. Multiple data types add richness to the data model. Secondary indexes add flexibility to the queries you can perform, without impacting performance.

Performance, reliability and security are built-in, with SSD-storage and automatic 3-way replication. Amazon DynamoDB uses proven cryptographic methods to securely authenticate users and prevent unauthorized data access.

Amazon S3

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 provides a simple web-services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

Amazon S3 is based on the idea that quality Internet-based storage

should be taken for granted. It helps free developers from worrying about how they will store their data, whether it will be safe and secure, or whether they will have enough storage available. It frees them from the upfront costs of setting up their own storage solution as well as the ongoing costs of maintaining and scaling their storage servers. The functionality of Amazon S3 is simple and robust: Store any amount of data inexpensively and securely, while ensuring that the data will always be available when you need it. Amazon S3 enables developers to focus on innovating with data, rather than figuring out how to store it.

For more information on Amazon S3, refer to:
http://aws.amazon.com/s3/

Start Lab

- Open https://808477742599.signin.aws.amazon.com/console
- Enter login credentials

Media storage

Images could be stored on Amazon Elastic Block Store volumes but you need to provision capacity in advance, and manage the operation of scaling up this storage layer by adding volumes. In addition, those volumes 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 if the data is not replicated and served from another instance.

A better approach is to use Amazon S3 as storage repository for media files. Amazon S3 provides high durability of data and the ability to serve content via HTTP. There is no limit to the number of objects that can be stored in an Amazon S3 bucket and 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.

Task 1: Create an Amazon S3 Bucket

Create a Bucket

3. In the **AWS Management Console**, on the Services menu, click **S3**.

4. Click *Create bucket* then configure:

- **Bucket name:**
- Replace **NUMBER** with a randon number
- Copy your bucket to a text edior
- Click Create

Assign a Bucket Policy

By default, content stored in Amazon S3 buckets cannot be accessed publicly. For your use case, you need to allow public access to the bucket, so users can download content. While you want to allow downloads

from the bucket, you don't want to allow users to list the contents of the bucket or delete any objects.

Amazon S3 enables you to manage access to objects and buckets using bucket policies. Bucket policies provide access control management at the bucket level for both a bucket and the objects in it. Bucket policies are a collection of JSON statements written in the access policy language.

5. Click the bucket that you created.

6. Click the **Permissions** tab.

7. Below **Public access settings for this bucket**, click Edit

8. De-select all of the boxes **

9. Click Save then configure:

- Enter
- Click Confirm

5. Click Bucket Policy

6. In the **Bucket policy editor** pane paste the following policy:

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

12. Replace **BUCKET** with the name of your bucket.

13. Click Save

Task 2: Create the Media Database

For each entry, you need to store metadata information such as the title, comment, publication date, entry type, etc. You need a database to store this information. Since you don't know how many entries your system will have to contain, and since you want your system to be scalable, you will use Amazon DynamoDB to store metadata information.

Creating an Amazon DynamoDB table

14. On the Services menu, click **DynamoDB**.

15. Click Create table then configure:

- **Table name:**
- **Primary key:**
- Click Create

The table's Overview tab will say "Table is being created". The table creation process might take a few minutes. Please proceed on to the next section and deploy the web server environment.

Task 3: Create the Web Front-End

In this task, you will use an Amazon EC2 instance to host the web server. The web application used for this lab is already packaged in an Amazon Machine Image (AMI).

Deploying Your Web Server

You will launch a default Amazon Linux Instance with a Ruby on Rails environment installed on initialization.

16. On the Services menu, click **EC2**.

17. Click Launch Instance

You first need to select a Linux AMI.

18. In the row for the **Amazon Linux AMI**, click Select

** Do not select **Amazon Linux 2 AMI**.

When you create an instance, AWS will ask you which instance type you want to use. The type you choose determines how much throughput and processing cycles are available to your instance. The **t2.micro** instance should be selected.

19. On the **Step 2**, click Next: Configure Instance Details

20. On **Step 3**, configure:

- **IAM role:** *EC2Profile*
- Scroll to the bottom of the screen, then expand **Advanced Details**

You need to specify which DynamoDB table and which Amazon S3 bucket to use. Amazon EC2 instances can access instance-specific metadata, as well as data supplied when launching the instances. You can pass information to the instance using the **User data** field.

21. In the **User data** section paste the following:

```
#!/bin/bash -ex
export HOME=/home/ec2-user
yum update -y
yum install -y gcc libxml2 libxml2-devel libxslt libxslt-devel
yum install -y ruby-devel sqlite-devel
yum install -y gcc-c++
yum install -y ImageMagick-devel ImageMagick-c++-devel
```

```
yum install -y patch
gem install bundler -v 1.16.0
curl -o /tmp/spl11.tar.gz https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-11/scripts/spl11v2.tar.gz
tar xfz /tmp/spl11.tar.gz -C /home/ec2-user/
cd /home/ec2-user/spl11/
gem install io-console -v 0.4.6
/usr/local/bin/bundle update rdoc
/usr/local/bin/bundle install
/usr/local/bin/bundle exec bin/rake db:migrate
cat > /home/ec2-user/spl11/tmp/creds.yml << EOF
region: REGION
bucket_name: BUCKET
table_name: myTable
EOF
/usr/local/bin/bundle exec bin/rails s -b 0.0.0.0 -p 80
```

22. In the **User data** text box:

- Replace **REGION** with the value of REGION located to the left of these instrucitons
- Replace **BUCKET** with the name of your bucket

** These values go into a YAML file. It is important ensure your text file is formatted correctly. There need to be a space after **Region:** and also after **bucket_name:**. For example:

- region: us-west-2
- bucket_name: mybucket2352345345

This script will be used on the instance to install packages, install the web application, provide a JSON config file with you app config parameters, and start the services on the instance.

23. Click Next: Add Storage

24. On **Step 4**, click Next Add Tags

25. On **Step 5**, click Add Tag then configure:

- **Key:**
- **Value:**

This name, more correctly known as a tag, will appear in the console once the instance launches. It makes it easy to keep track of running machines in a complex environment.

26. Click Next: Configure Security Group

You will create a security group. A *security group* acts as a firewall that controls the traffic allowed into a group of instances. When you launch an Amazon EC2 instance, you can assign it to one or more security groups. For each security group, you add rules that govern the allowed inbound traffic to instances in the group. All other inbound traffic is discarded. You can modify rules for a security group at any time. The new rules are automatically enforced for all existing and future instances in the group.

The instance you're creating is a web server so you need to assign a rule for inbound HTTP traffic.

27. On **Step 6**, configure:

- **Security group name:**
- **Description:**

27. Click Add Rule then configure:

- **Type:**
- **Source:** *Anywhere*

This will add a default handler for HTTP that will allow requests from anywhere on the Internet.

29. Click Review and Launch

You may see a warning on this screen that "Your security group … is open to the world." This is a result of not restricting SSH access to your instance, as described above. For the purposes of this lab only, you may ignore this warning.

30. On **Step 7**, review your choices, and then click Launch

You will receive a popup window to select a key pair or create a new one.

31. Select **I acknowledge that…**.

32. Click Launch Instances

Next you will see a status page, notifying you that your instances are launching.

33. Click View Instances

You will now be taken to the Instances tab of the Amazon EC2 Dashboard, which displays the list of all running Amazon EC2 instances in the currently selected region. You can see the status of your instance here.

34. Your **WebServer** should be selected.

You should see a list of details and status update for your instance in the bottom pane of the console. Scroll through the various details about your running instance.

35. Wait till the instance displays:

- **running**
- **2/2 checks passed**

The app takes an additional 3-4 minutes to initialize itself once the status check is green. You can click the **refresh** button to refresh the status.

When it is running and has passed status checks, continue on to the next section.

Task 4: Test the Deployment

Retrieve Your Host's Public DNS Address

36. Identify the **Public DNS(IPv4)** value in the detail pane at the bottom of the screen and copy it to the clipboard. It will look something like: ec2-54-84-236-205.compute-1.amazonaws.com.

37. Open a browser tab, then:

- Paste your DNS address into the address bar
- Press **ENTER**

You should see the home page of the web application:

38. Add a couple of images to the application by clicking **New**, then suppliying:

- **Title:**
- **Description:**
- **Tags:**

38. In the **AWS Management Console**, on the Services menu, click **DynamoDB**.

39. In the left navigation pane, click **Tables**.

40. Click **myTable**.

41. Click the **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 Amazon S3 bucket.

43. On the Services menu, click **S3**.

44. Click your **mybucket**.

You will see the list of image files uploaded by the application in the bucket along with the thumbnail images (with the "tn.png" extension).

Architecture Overview

So far, you 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 Amazon S3 bucket and insert the image metadata into the Amazon DynamoDB table.

While Amazon S3 and Amazon DynamoDB are both scalable and fault-tolerant systems, your web server running on a single Amazon EC2 instance is clearly a single point of failure (if the web application fails, the system is not accessible and cannot recover) and a bottleneck (with an

important load of incoming requests, the system might even become unavailable).

End Lab [step9]

Follow these steps to close the console, end your lab, and evaluate the experience.

45. Return to the AWS Management Console.

46. On the navigation bar, click **<yourusername\>@<AccountNumber\>**, and then click **Sign Out**.

Conclusion

Congratulations! You now have successfully:

- Created a new Amazon S3 bucket to hold your media files
- Created a security group to restrict access to the server's resources
- Launched a new Amazon EC2 instance to run your web server
- Created an Amazon DynamoDB database to hold your data

Additional Resources {#step11}

- AWS Training and Certification
- Source Code for Web App