

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 4150-A Fall 2021
Lab: Photo Gallery Application - SQL and NoSQL Variants

References:

- [1] A. Bahga, V. Madisetti, "Cloud Computing Solutions Architect: A Hands-On Approach", ISBN: 978-0996025591
- [2] <https://aws.amazon.com/documentation/>

Due Date:

The lab report will be **due on November 5, 2021 at 11:59 PM.**

In this lab, we will create a Photo Gallery application composed of albums and photos using two variations to store records of photos: SQL and NoSQL. We are going to integrate the same method we used in the previous lab.

In the SQL variant of the application, the records of albums and photos are maintained in a MySQL database instance on Amazon RDS. Whereas in the NoSQL variant of the application, the records of photos are held in a DynamoDB table. This application is implemented in Python and uses the Flask web framework. We are going to deploy the application on an Amazon EC2 instance.

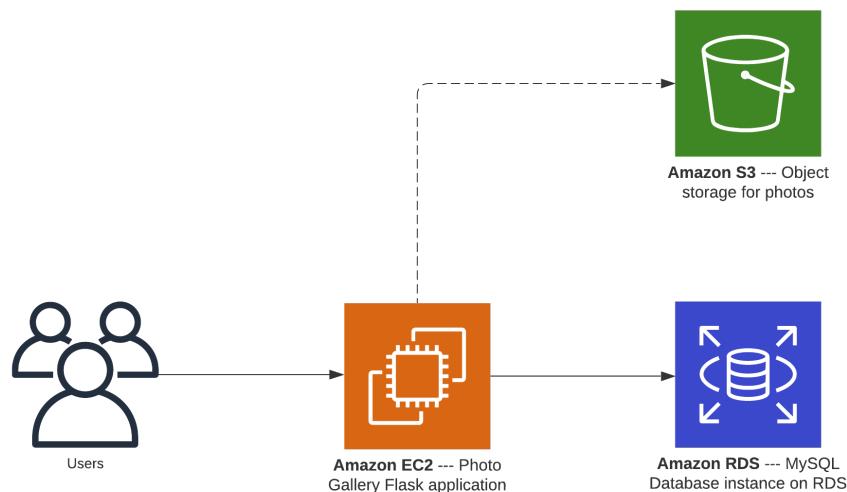


Fig.1 Architecture diagram of the Photo Gallery application - SQL Variant

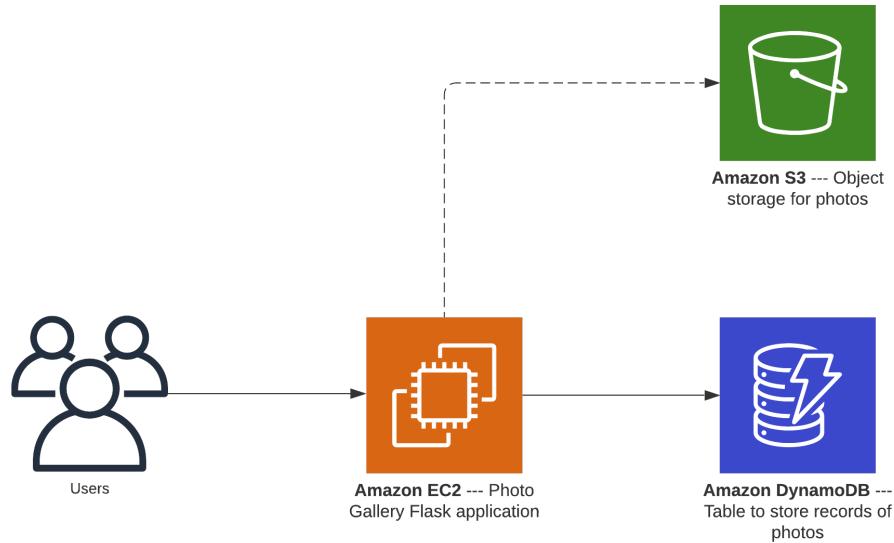
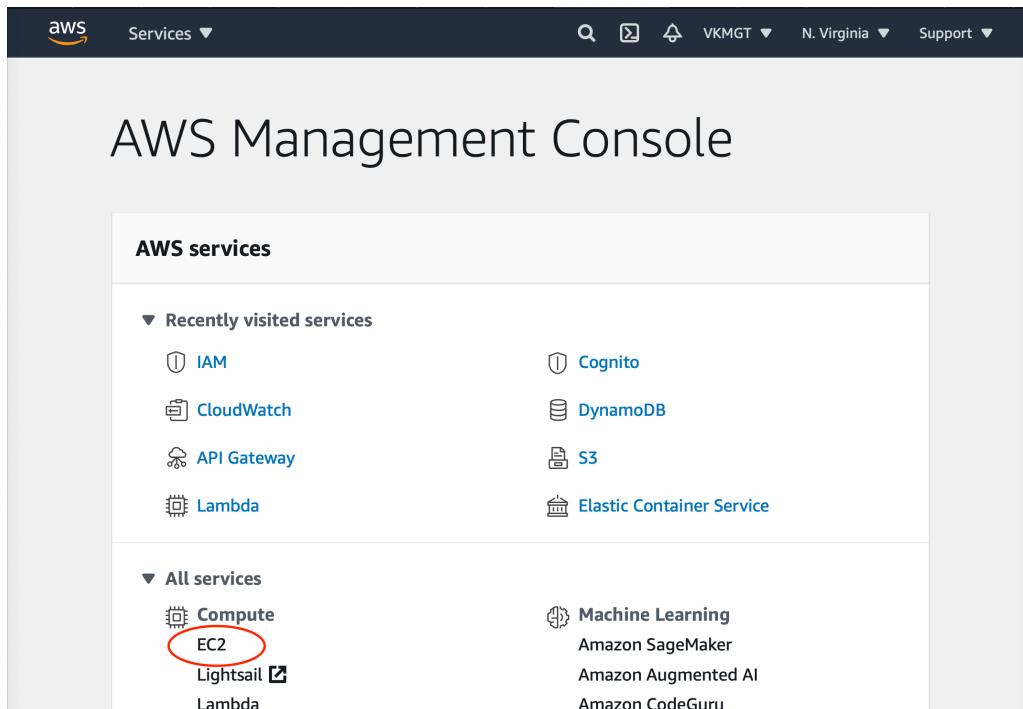


Fig. 2 Architecture diagram of the Photo Gallery application - SQL Variant

1. Create EC2 instance for hosting the static website and resources (10 points)

- In the AWS Management Console, under Compute, click **EC2**. You can also use the search input in the navigation bar by typing “**EC2**”



- Once in the EC2 console, click **Instances** under the instances dropdown menu in the navigation bar on the left

Welcome to the new EC2 console!
We're redesigning the EC2 console to make it easier to use and improve performance. We'll release new screens periodically. We encourage you to try them and let us know where we can make improvements. To switch between the old console and the new console, use the New EC2 Experience toggle.

Resource	Count
Instances (running)	0
Dedicated Hosts	0
Elastic IPs	1
Instances	0
Key pairs	4
Load balancers	0

- Lunch an instance by clicking the orange button "**Launch instances**"

Welcome to the new instances experience!
We're redesigning the EC2 console to make it easier to use. To switch between the old console and the new console, use the New EC2 Experience toggle above the navigation panel. We'll release updates continuously based on customer feedback.

Instances Info

Actions ▾

Launch instances

Name	Instance ID	Instance state
------	-------------	----------------

- In the first step to launching an instance, you have the option to choose the **Amazon Machine Image (AMI)** that you would like to use for your server instance. Go through the list to get familiar with the images that AWS offers. For our lab, we are going to use an Ubuntu image to run our flask server

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Cancel and Exit

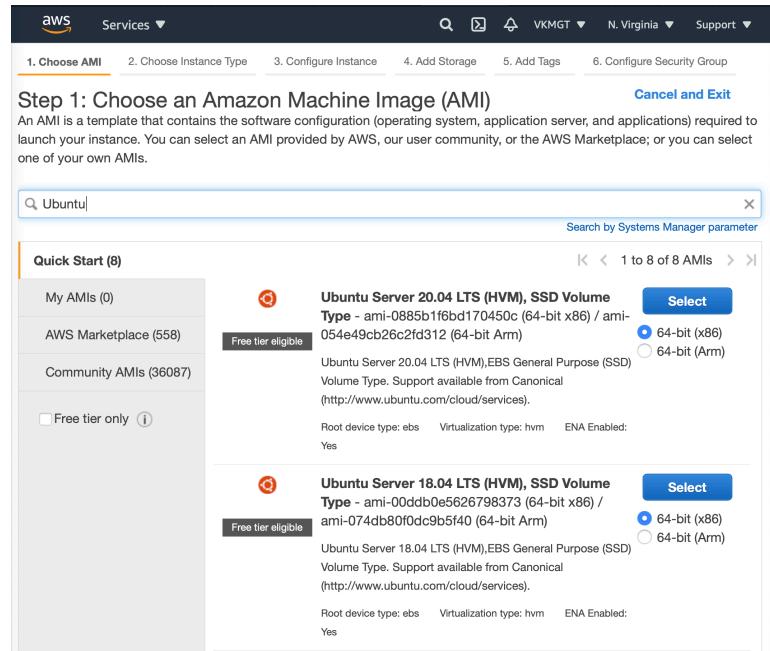
Quick Start

- My AMIs
- AWS Marketplace
- Community AMIs
- Free tier only *(i)*

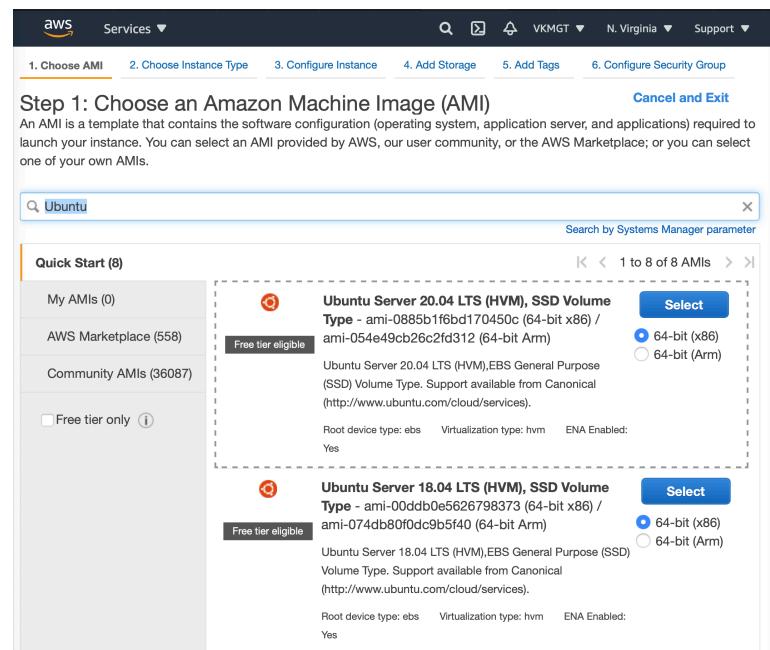
1 to 41 of 41 AMIs

Amazon Linux	Free tier eligible	Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-047a51fa27710816e (64-bit x86) / ami-03c5cc3d1425c6d34 (64-bit Arm)	Select
		Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
		Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	
macOS Catalina 10.15.7	ami-0f981206a71da3cbc	Select	64-bit (Mac)
The macOS Catalina AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.			
		Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	
macOS Mojave 10.14.6	ami-0c9d808cfac0d1ec3	Select	64-bit (Mac)
The macOS Mojave AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.			
		Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	

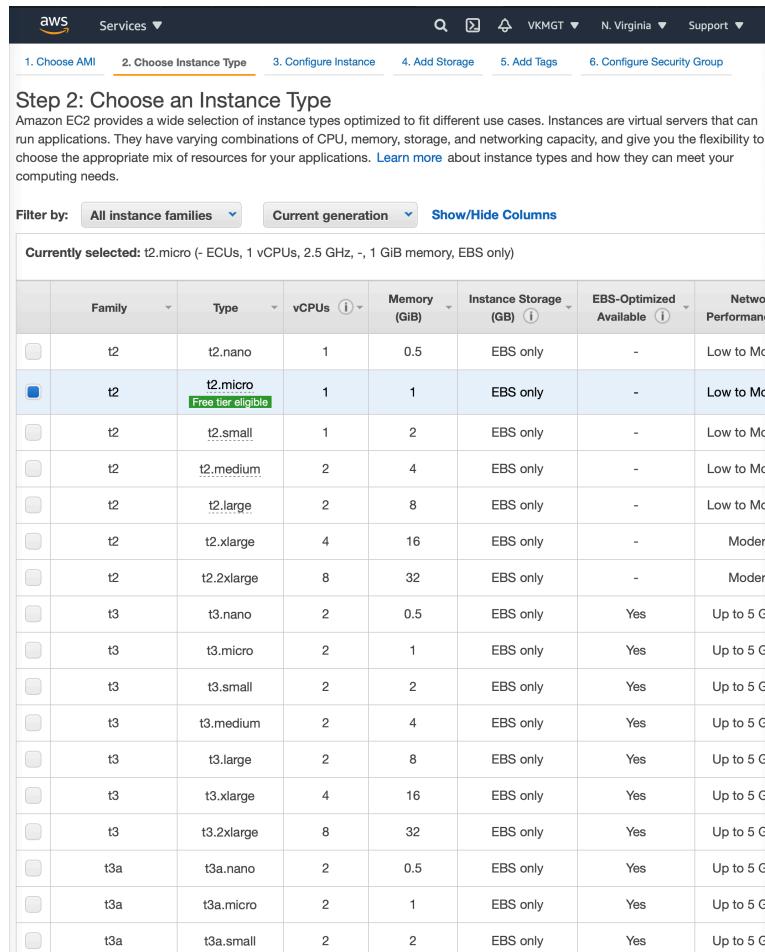
- To find the right Ubuntu image, type “Ubuntu” in the search input and find the most recent Ubuntu version. In this given time, the most recent snapshot is **20.04**



- Select the **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** image



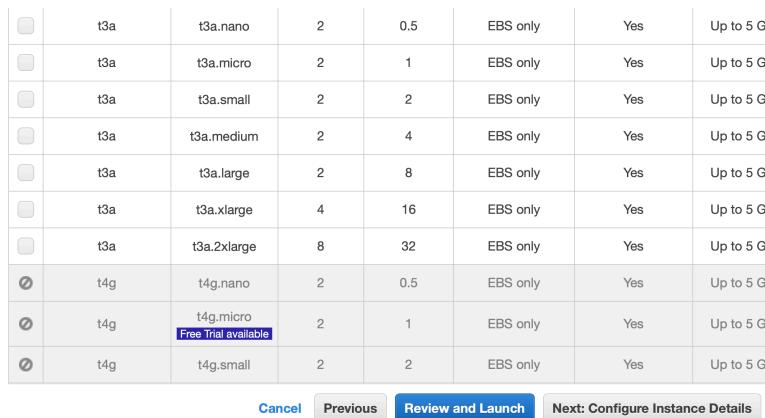
- In the second step, we can select the type of instance that we would like to use. For our lab, we will choose the **t2.micro** since it is the free tier eligible version



The screenshot shows the AWS EC2 instance selection interface. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type (which is underlined), 3. Configure Instance, 4. Add Storage, 5. Add Tags, and 6. Configure Security Group. Below the tabs, a heading says "Step 2: Choose an Instance Type". A sub-instruction states: "Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs." A filter bar at the top allows filtering by "All instance families" (selected), "Current generation" (selected), and "Show/Hide Columns". A note below the filter says "Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)". The main table lists various instance types with their details:

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.small	2	2	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.medium	2	4	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.large	2	8	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.xlarge	4	16	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3	t3.2xlarge	8	32	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.nano	2	0.5	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.micro	2	1	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.small	2	2	EBS only	Yes	Up to 5 G

- After selecting the instance type, click "**Review and Launch**"



The screenshot shows the "Review and Launch" step of the instance creation process. The table from the previous step is shown again, with the "t4g.micro" row selected. Below the table, there are navigation buttons: "Cancel", "Previous", "Review and Launch" (which is highlighted in blue), and "Next: Configure Instance Details".

<input type="checkbox"/>	t3a	t3a.nano	2	0.5	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.micro	2	1	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.small	2	2	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.medium	2	4	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.large	2	8	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.xlarge	4	16	EBS only	Yes	Up to 5 G
<input type="checkbox"/>	t3a	t3a.2xlarge	8	32	EBS only	Yes	Up to 5 G
<input checked="" type="checkbox"/>	t4g	t4g.nano	2	0.5	EBS only	Yes	Up to 5 G
<input checked="" type="checkbox"/>	t4g	t4g.micro <small>Free Trial available</small>	2	1	EBS only	Yes	Up to 5 G
<input checked="" type="checkbox"/>	t4g	t4g.small	2	2	EBS only	Yes	Up to 5 G

- The last step is to review the configuration of the instance (e.g., AMI, instance type, and other details that we will create by default as part of this lab). Once you review the instance configuration, click “**Launch**”

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠ Improve your instances' security. Your security group, launch-wizard-11, is open to the world.

Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.

You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

AMI Details [Edit AMI](#)

Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0885b1f6bd170450c
Free tier eligible
 Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
 Root Device Type: ebs Virtualization type: hvm

Instance Type [Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

Security Groups [Edit security groups](#)

Security group name launch-wizard-11
Description launch-wizard-11 created 2021-02-03T22:07:11.081-05:00

Type <i>(i)</i>	Protocol <i>(i)</i>	Port Range <i>(i)</i>	Source <i>(i)</i>	Description <i>(i)</i>
SSH	TCP	22	0.0.0.0/0	

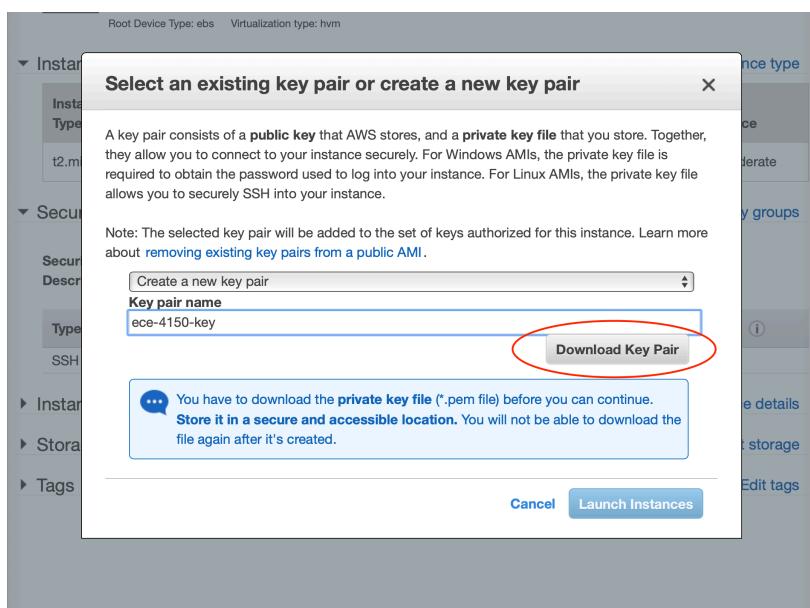
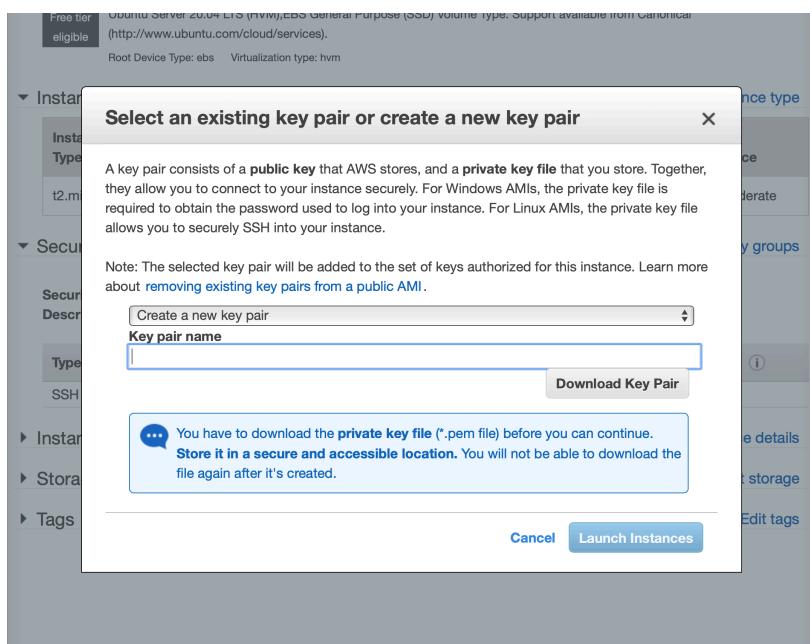
Instance Details [Edit instance details](#)

Storage [Edit storage](#)

Tags [Edit tags](#)

[Cancel](#) [Previous](#) **Launch**

- After you clicked the Launch button, you will have the option to create a new key pair or attach it to an existing key pair. To access an instance, AWS gives you a private key that you can use to access the instance remotely. To access your instance, you will use the SSH protocol to connect via the terminal or PowerShell
- To create a new one, select "**Create a new key pair**" and name the key pair. Once you placed the key's name, download the key by clicking the "**Download Key Pair**" button. After you download the Key Pair, you will be allowed to launch the instance. **Ensure you store that key in a secure place because you won't have access to the instance again if you lose it**



- It might take a few seconds or minutes for the server instance to be available. However, you can view the instance status by clicking the “View Instances” button

The screenshot shows the AWS Launch Status page. At the top, there's a green success message: "Your instances are now launching" with a note about instance ID i-03cb8c4ddfcf0cef and a link to "View launch log". Below it is a blue info message: "Get notified of estimated charges" with a link to "Create billing alerts". A section titled "How to connect to your instances" provides instructions and links to "How to connect to your Linux instance", "Amazon EC2 User Guide", "Learn about AWS Free Usage Tier", and "Amazon EC2: Discussion Forum". It also lists "Create status check alarms", "Create and attach additional EBS volumes", and "Manage security groups". At the bottom right is a blue "View Instances" button.

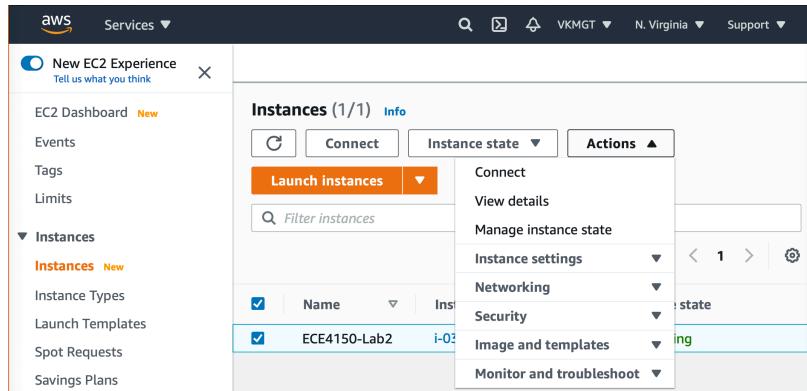
- You should see your instance running, as shown below

The screenshot shows the AWS EC2 Instances page. The left sidebar has "Instances" selected. The main table shows one instance row for "i-03cb8c4ddfcf0cef" which is currently "Pending". The table columns include Name, Instance ID, Instance state, Instance type, Status check, and Alarm status.

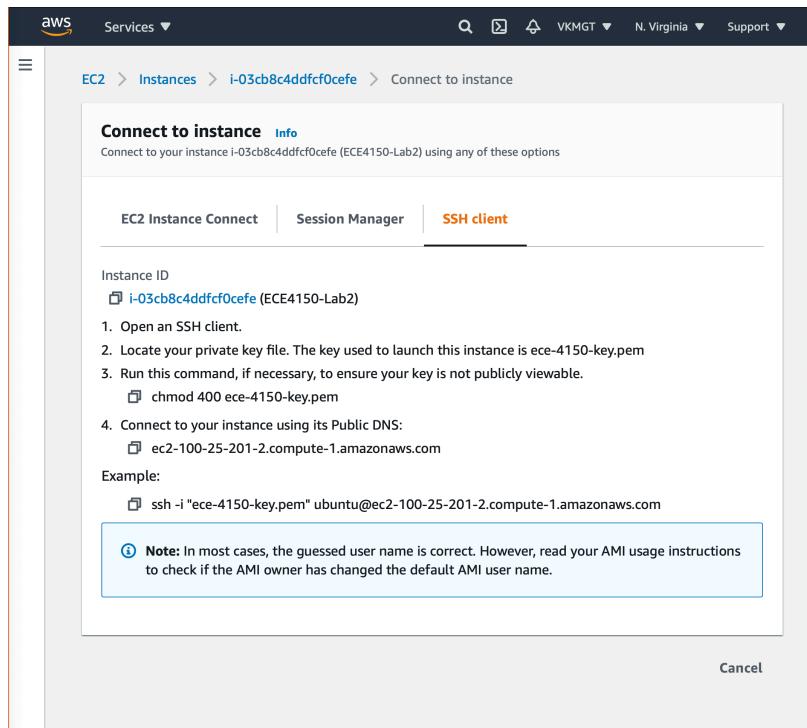
- If you want to change the name of the instance, click in the section below the Name column

The screenshot shows the AWS EC2 Instances page with the instance name "i-03cb8c4ddfcf0cef" highlighted and circled in red. A modal dialog box titled "Edit Name" is open over the table, showing the current value "ECE4150-Lab2". At the bottom of the dialog are "Cancel" and "Save" buttons.

- To connect to your new server instance, select your instance, click the “Actions” dropdown, and click connect

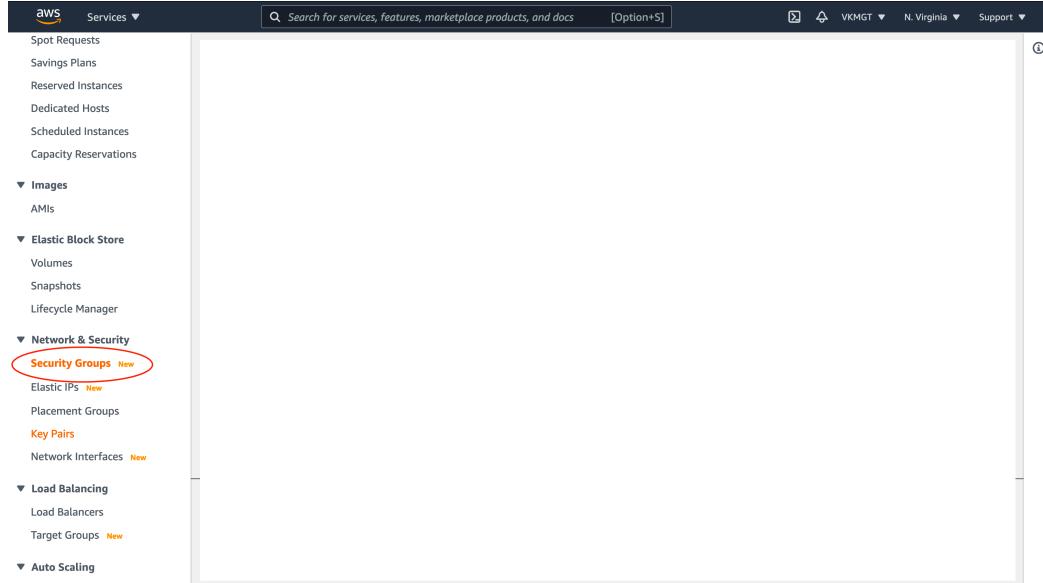


- Under the **SSH client** tab, you will find instructions to understand how to connect to your instance remotely

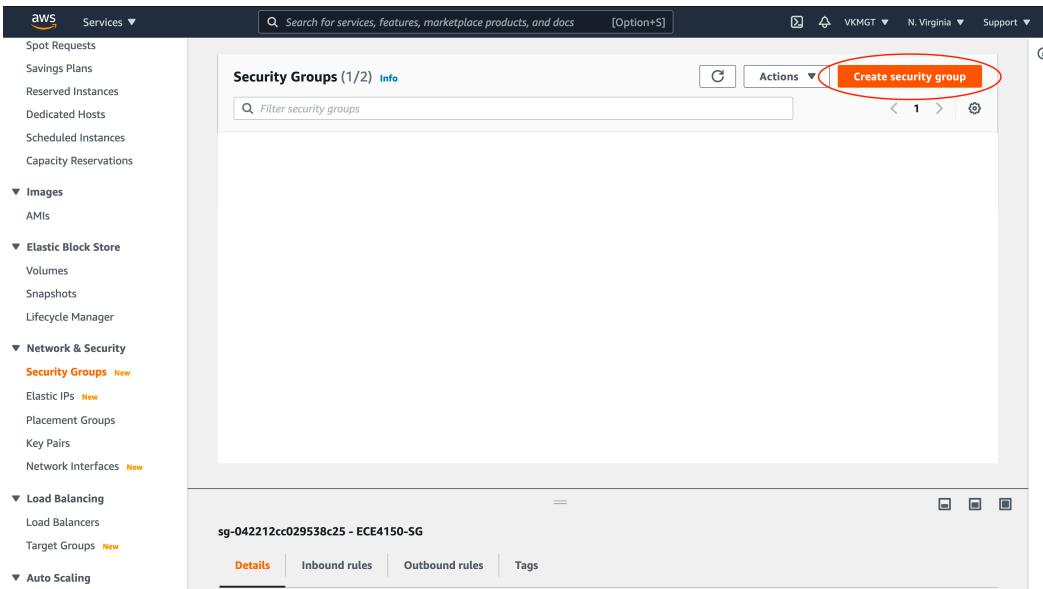


2. Create a Security Group (SG) to allow access from your EC2 from anywhere or a specific location (4 points)

- In your EC2 Console, under **Network & Security**, click **Security Groups**. You can also use the search input in the navigation bar by typing “**Security Groups**” Under “Features”



- Create a new Security Group by clicking the orange button called “**Create security group**” in the top right corner



- Add a name and description to the security group as shown below

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)
ECE4150-SG
Name cannot be edited after creation.

Description [Info](#)
Allows SSH and HTTP access to instance

VPC [Info](#)
vpc-5e663f24

Inbound rules [Info](#)

This security group has no inbound rules.

Add rule

- At this time, we are going to add one single inbound rule. We will allow access through SSH from anywhere, as shown below. Do not change the configuration for the outbound rule

Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
SSH	TCP	22	Custom <input type="text" value="0.0.0.0"/> <input type="button" value="X"/>	<input type="text" value=""/> <input type="button" value="Delete"/>

Add rule

Outbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info
All traffic	All	All	Custom <input type="text" value="0.0.0.0"/> <input type="button" value="X"/>	<input type="text" value=""/> <input type="button" value="Delete"/>

Add rule

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

- Attached the new security group to the EC2 instance. Click “**instances**” in the left navigation menu. Select your server instance, and under **Actions**, click **Security** and **Change security groups**

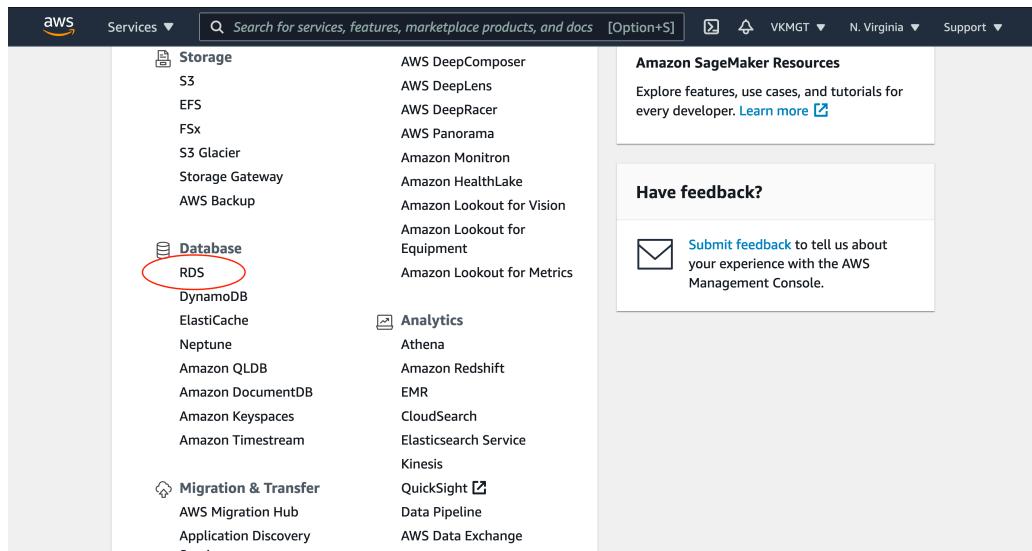
The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like EC2 Dashboard, Events, Tags, Limits, Instances (with Instance Types, Launch Templates, etc.), Images (AMIs), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security. The main area shows a table with one row for the instance ECE4150-Lab2. The Actions menu is open, and the Security section is expanded, showing links for Change security groups, Get Windows password, and Modify IAM role.

- Remove the attached security group and add your new security group, then click “**Save**”

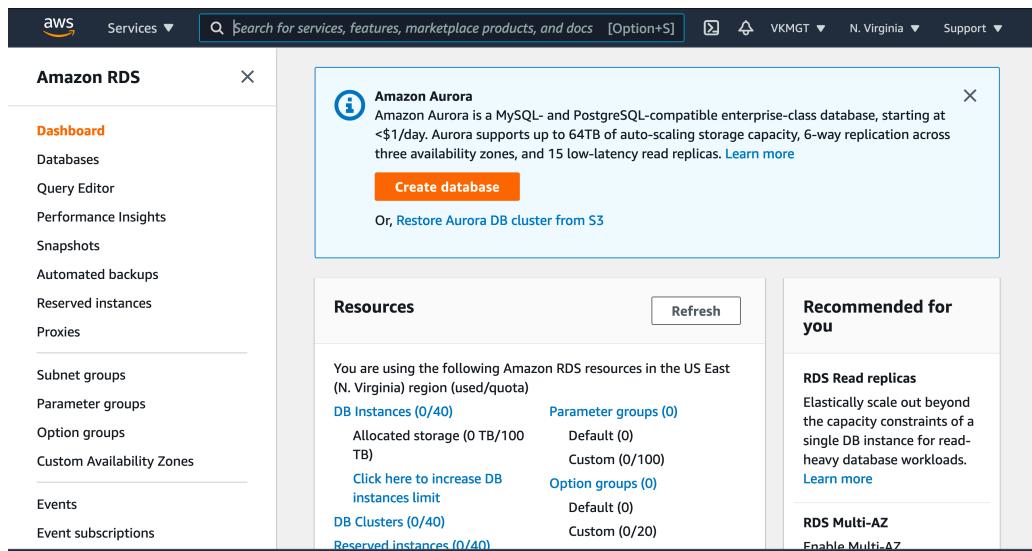
The screenshot shows the 'Change security groups' dialog box. At the top, it says 'EC2 > Instances > i-03cb8c4ddfcf0cefe > Change security groups'. The main area has two sections: 'Instance details' (showing Instance ID i-03cb8c4ddfcf0cefe (ECE4150-Lab2) and Network interface ID eni-01a06a9979b070009) and 'Associated security groups' (listing 'Security group name' ECE4150-SG and 'Security group ID' sg-042212cc029538c25). There are 'Add security group' and 'Remove' buttons. At the bottom are 'Cancel' and 'Save' buttons.

3. Create an RDS database instance for storing records of photos (10 points)

- In the AWS Management Console, under Database, click **RDS**. You can also use the search input in the navigation bar by typing "**RDS**"



- Launch a database instance by clicking the orange button "Create database"



- Follow the configurations as shown below

The screenshot shows the 'Create database' wizard in the AWS RDS console. In the 'Choose a database creation method' step, 'Standard create' is selected. The 'Engine options' step displays the following engine type choices:

- Amazon Aurora**: Unselected.
- MySQL**: Selected.
- MariaDB**: Unselected.
- PostgreSQL**: Unselected.
- Oracle**: Unselected.
- Microsoft SQL Server**: Unselected.

The screenshot shows the continuation of the 'Create database' wizard. Under 'Edition', 'MySQL Community' is selected. The 'Known Issues/Limitations' section contains a note about potential compatibility issues. The 'Version' dropdown is set to 'MySQL 8.0.20'. In the 'Templates' section, the 'Free tier' option is selected, which is described as using the RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

- Under settings, for the “DB instance identifier”, type the name of your DB instance. For this lab, it is going to be **photogallerydb**. For the Master username, type the the login ID use to connect to the database instance; we are going to use **root** for the login ID. Create a password for authentication.

The screenshot shows the AWS RDS 'Settings' page for creating a new database instance. On the left, there's a sidebar with 'Services ▾' and a search bar at the top. The main area has tabs for 'Settings', 'Storage', and 'Autoscaling'. Under 'Settings', the 'Database name' is set to 'photogallerydb'. The 'Master username' is 'root'. Under 'Master password', two fields are shown: one with '*****' and another with '*****'. The 'DB instance size' section is partially visible at the bottom. On the right, a modal window titled 'Database name' provides a detailed description of what a database name is and how it's used.

- Follow the default configurations as shown below

The screenshot shows the 'DB instance size' configuration page. It includes sections for 'DB instance class' (with 'Burstable classes (includes t classes)' selected), 'Storage' (with 'General Purpose (SSD)' selected), and 'Autoscaling' (with 'Enable storage autoscaling' checked). A note at the bottom says 'Enabling this feature will allow the storage to increase once the specified threshold is reached.'

Storage

Storage type: [Info](#) General Purpose (SSD)

Allocated storage: 20 GiB (Minimum: 20 GiB, Maximum: 16,384 GiB) Higher allocated storage [may improve](#) IOPS performance.

Storage autoscaling: [Info](#) Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

Maximum storage threshold: [Info](#) Charges will apply when your database autoscales to the specified threshold

1000 GiB (Minimum: 21 GiB, Maximum: 16,384 GiB)

Availability & durability

Multi-AZ deployment: [Info](#)

- Do not create a standby instance
- Create a standby instance (recommended for production usage) Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

- Make sure you select the make your database publicly available over the internet, as shown below

Connectivity

Virtual private cloud (VPC) [Info](#) VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-5e663f24)

Only VPCs with a corresponding DB subnet group are listed.

[Info](#) After a database is created, you can't change the VPC selection.

Subnet group [Info](#) DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default

Public access [Info](#)

Yes Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

No RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

Choose existing Choose existing VPC security groups

Create new Create new VPC security group

Existing VPC security groups Choose VPC security groups

- Attach the security group that you created previously

Connectivity

Subnet group default-vpc-5e663f24

Security group List of DB security groups to associate with this DB instance.

Choose security groups ECE4150-SG

Certificate authority rds-ca-2019

Additional configuration

- On **Additional configuration**, under **Initial database name**, type the same name you use for the **DB instance identifier**, “**photogallerydb**”, and under Monitoring, “**Enable Enhanced Monitoring**” as shown below

Additional configuration

Database options, backup enabled, backtrack disabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled

Database options

Initial database name [Info](#)
photogallerydb

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)
default.mysql8.0

Option group [Info](#)
default:mysql-8-0

Backup

Creates a point-in-time snapshot of your database

Enable automatic backups
Enabling backups will automatically create backups of your database during a certain time window.

⚠ Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details [here](#).

Backup retention period [Info](#)
Choose the number of days that RDS should retain automatic backups for this instance.
7 days

Backup window [Info](#)
Select the period for which you want automated backups of the database to be created by Amazon RDS.

Select window
 No preference

Copy tags to snapshots

Monitoring

Enable Enhanced monitoring
Enabling Enhanced monitoring metrics are useful when you want to see how different processes or threads use the CPU

Granularity
60 seconds

Monitoring Role
default

Clicking "Create database" will authorize RDS to create the IAM role rds-monitoring-role

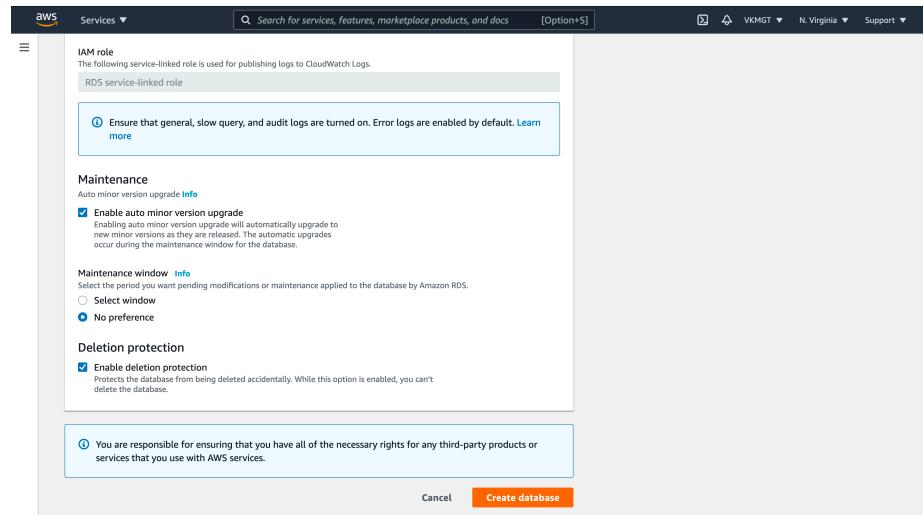
Log exports
Select the log types to publish to Amazon CloudWatch Logs

Error log
 General log
 Slow query log

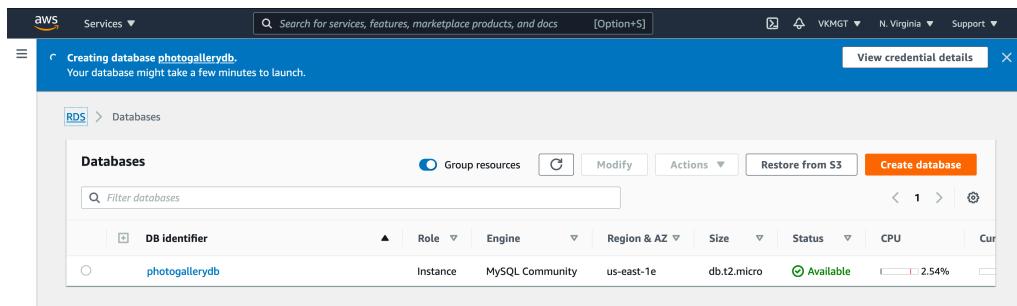
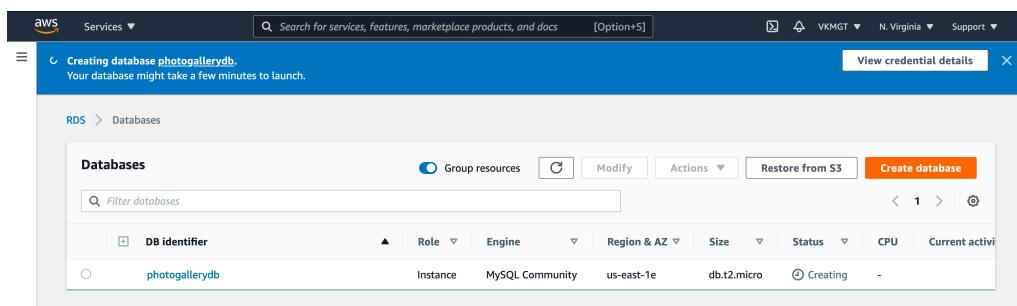
IAM role
The following service-linked role is used for publishing logs to CloudWatch Logs.
RDS service-linked role

ⓘ Ensure that general, slow query, and audit logs are turned on. Error logs are enabled by default. [Learn more](#)

- Once you completed the configuration, create the database by clicking **Create database**



- It might take around 10-20 minutes to create, depending on the allocation. It will change from Creating to Available under Status



- Click on the newly created database, copy and save the endpoint to the database instance located under **Endpoint & port**, as shown below

The screenshot shows the AWS RDS console for the 'photogallerydb' database. The left sidebar has 'Databases' selected. The main area shows the database summary with various metrics like CPU usage and status. Below the summary is a tab navigation bar with 'Connectivity & security' selected. The 'Endpoint & port' section contains the database endpoint (photogallerydb.c2mvmn6pdqo1.us-east-1.rds.amazonaws.com) and port (3306), which is circled in red.

DB identifier	CPU	Status	Class
photogallerydb	2.33%	Available	db.t2.micro

Role	Current activity	Engine	Region & AZ
Instance	2 Connections	MySQL Community	us-east-1e

Connectivity & security		Networking	Security
Endpoint	photogallerydb.c2mvmn6pdqo1.us-east-1.rds.amazonaws.com	Availability zone us-east-1e	VPC security groups ECE4150-SG (sg-042212cc029538c25) (active)
Port	3306	VPC vpc-5e663f24	Public accessibility Yes
		Subnet group default-vpc-5e663f24	Certificate authority rds-ca-2019
		Subnets subnet-6258085c	Certificate authority date

- Go back to the Security Group section under the EC2 Console and add another inbound rule for a **Custom TCP** type from port **3306** (You also can add a **MySQL/Aurora** rule type) and configure the source coming from anywhere (**0.0.0.0/0**)

4. Create a DynamoDB table for storing records of photos (2 points)

- Create a DynamoDB table named **PhotoGallery** with a partition key and a sort key named **albumID** and **photoID**, respectively, as shown below

The screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The 'Table details' step is active. It requires a table name (PhotoGallery) and defines a primary key with a partition key ('albumID') and a sort key ('photoID'). Both keys are of type 'String'.

Table details

DynamoDB is a schema-less database that only requires a table name and a primary key.

Table name
This will be used to identify your table.
PhotoGallery

Partition key
The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table, as well as allocate data across hosts for scalability and availability.
albumID String

Sort key - optional
The sort key can be the second part of the table's primary key. The sort key allows for searching within an item collection.
photoID String

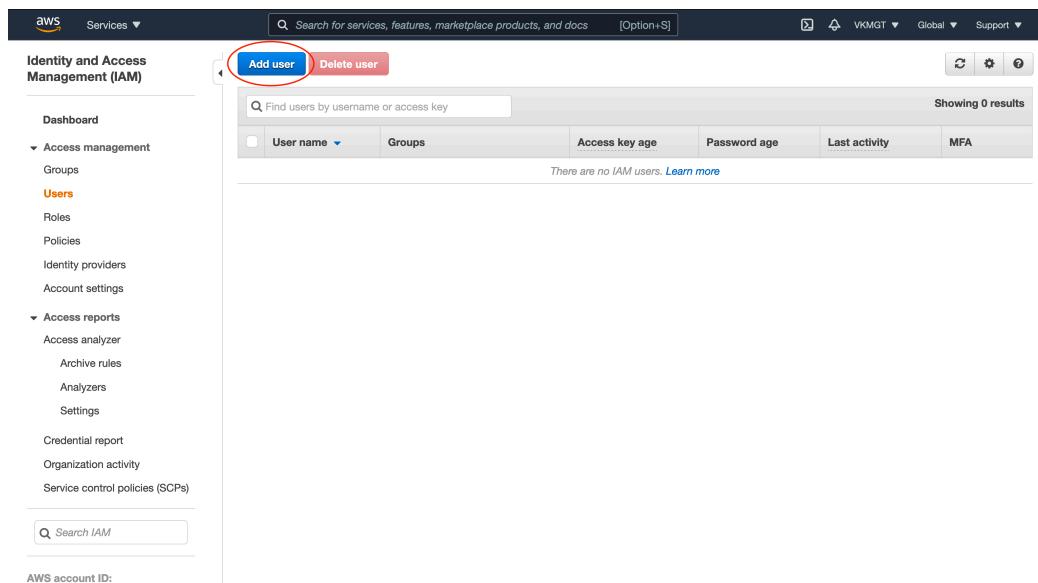
5. Create an S3 Bucket for storing photos (2 points)

- Create a new S3 bucket for storing photos. **Uncheck Block all public access under Bucket setting for Block Public Access section**
- For the name of the bucket, use **photobucket-lastname-year-courseNumber** (e.g., **photobucket-corporan-2021-4150**)
- Create two folders in this bucket. One called '**photos**' and another one called '**thumbnails**'
- Add a bucket policy below to enable public access to the photos uploaded. Replace '**mybucketname**' with the name of the S3 bucket created

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "PublicReadGetObject",  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "s3:GetObject",  
            "Resource": "arn:aws:s3:::mybucketname/photos/*"  
        }  
    ]  
}
```

6. Create IAM User (2 points)

- In the AWS IAM console, create a new IAM user called **ec2_instance_access** for the EC2 instance to S3 service, as shown below. Make sure that the Access Type is only **programmatic**. Attach policy called **AmazonS3FullAccess** and **AmazonDynamoDBFullAccess** to this user. Save the **Access key ID** and **Secret access key**





Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Programmatic access

Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access

Enables a password that allows users to sign-in to the AWS Management Console.



Add user

1 2 3 4 5

Set permissions

 Add user to group	 Copy permissions from existing user	 Attach existing policies directly
Create policy		
Showing 6 results		
Policy name	Type	Used as
<input type="checkbox"/> AmazonDMSRedshiftS3Role	AWS managed	None
<input checked="" type="checkbox"/> AmazonS3FullAccess	AWS managed	Permissions policy (1)
<input type="checkbox"/> AmazonS3OutpostsFullAccess	AWS managed	None
<input type="checkbox"/> AmazonS3OutpostsReadOnlyAccess	AWS managed	None
<input type="checkbox"/> AmazonS3ReadOnlyAccess	AWS managed	None
<input type="checkbox"/> QuickSightAccessForS3StorageManagementAnalyticsReadOnly	AWS managed	None

Cancel Previous Next: Tags



Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name
AWS access type Programmatic access - with an access key
Permissions boundary Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonS3FullAccess
Managed policy	AmazonDynamoDBFullAccess

Tags

No tags were added.

Cancel Previous Create user



Add user

1 2 3 4 5

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

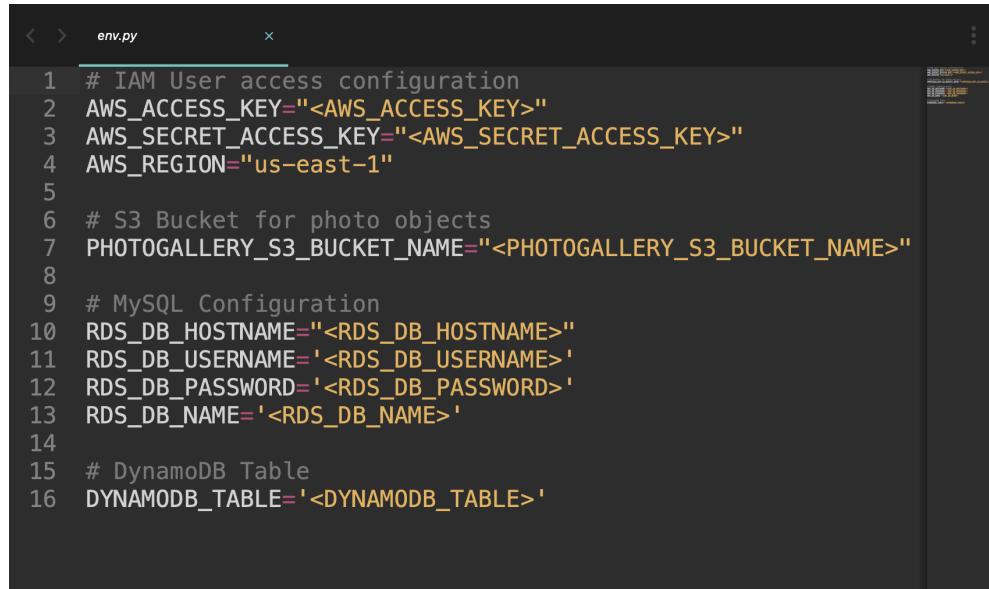
Users with AWS Management Console access can sign-in at: <https://vkmgt.sigin.aws.amazon.com/console>

[Download.csv](#)

User	Access key ID	Secret access key
ec2_instance_access	AKIA3TPWV54KKVD67ZXZ	***** Show

7. Update env.py script and move the entire code directory to the EC2 instance (2 points)

- In the files provided, locate the **env.py** script under the **utils** directory and update all the variables required to run the Photo Gallery Application using the two variants.
- **DO NOT PLACE THE NAME INSIDE THE ANGLE BRACKETS.** For example, if the name of the S3 bucket to store photos is **photobucket-corporan-2021-4150** replace **<PHOTOGALLERY_S3_BUCKET_NAME>** for the name of the bucket.



```
1 # IAM User access configuration
2 AWS_ACCESS_KEY="<AWS_ACCESS_KEY>"
3 AWS_SECRET_ACCESS_KEY="<AWS_SECRET_ACCESS_KEY>"
4 AWS_REGION="us-east-1"
5
6 # S3 Bucket for photo objects
7 PHOTOGALLERY_S3_BUCKET_NAME="<PHOTOGALLERY_S3_BUCKET_NAME>"
8
9 # MySQL Configuration
10 RDS_DB_HOSTNAME="<RDS_DB_HOSTNAME>"
11 RDS_DB_USERNAME='<RDS_DB_USERNAME>'
12 RDS_DB_PASSWORD='<RDS_DB_PASSWORD>'
13 RDS_DB_NAME='<RDS_DB_NAME>'
14
15 # DynamoDB Table
16 DYNAMODB_TABLE='<DYNAMODB_TABLE>'
```

8. Transferring Files between your computer to and Amazon EC2 instance (2 points)

- For linux/Unix/Mac system, we can use a command-line tool “**scp**” (secure copy) to transfer files between your laptop and Amazon instance. With **scp** you can copy files between computers on a network.
 - To **upload** a file from your computer to the EC2 instance:

```
$scp -i amazonkey.pem lab2/code.py ubuntu@ec2-host.amazonaws.com:~/data/
```

- To **upload** a directory from your computer to the EC2 instance:

```
$scp -i amazonkey.pem -r lab2 ubuntu@ec2-host.amazonaws.com:~/data/
```

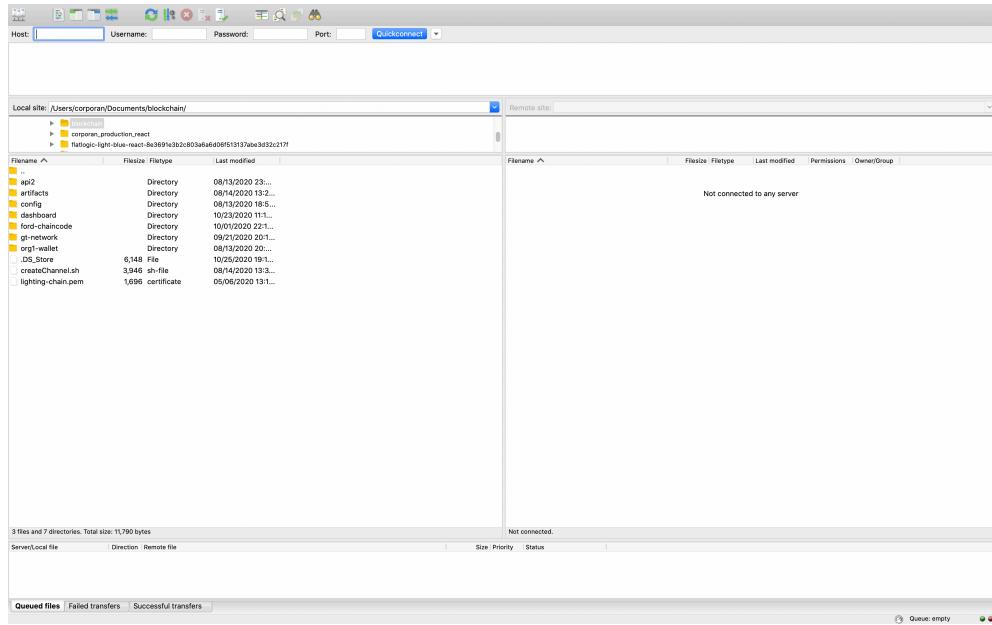
- To **download** a file from the EC2 instance to your computer to:

```
$scp -i amazonkey.pem ubuntu@ec2-host.amazonaws.com:/data/code.py ~/Download
```

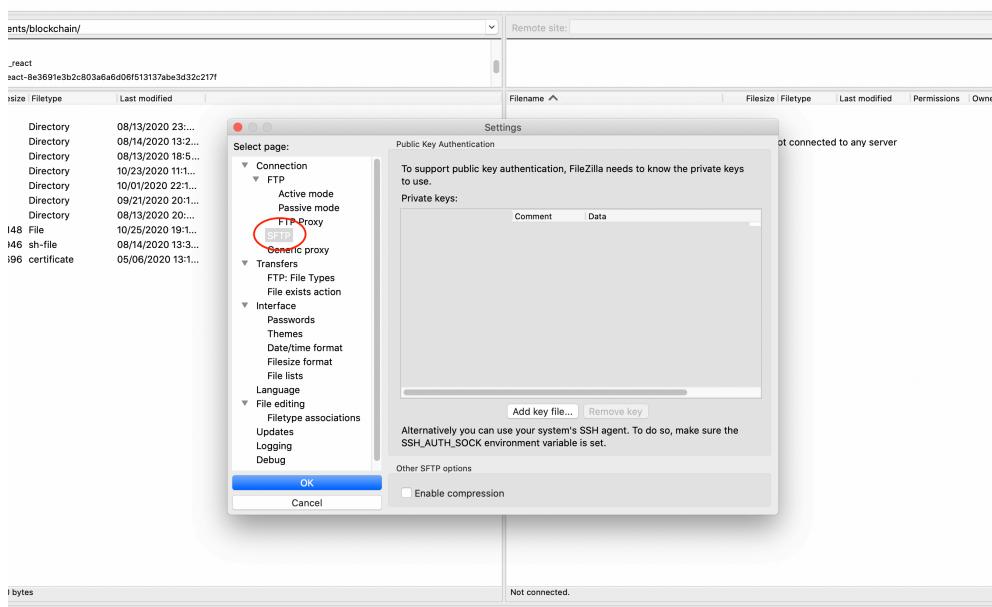
- To **download** a directory from your computer to the EC2 instance:

```
$scp -i amazonkey.pem -r ubuntu@ec2-host.amazonaws.com:/data/lab2 ~/Download
```

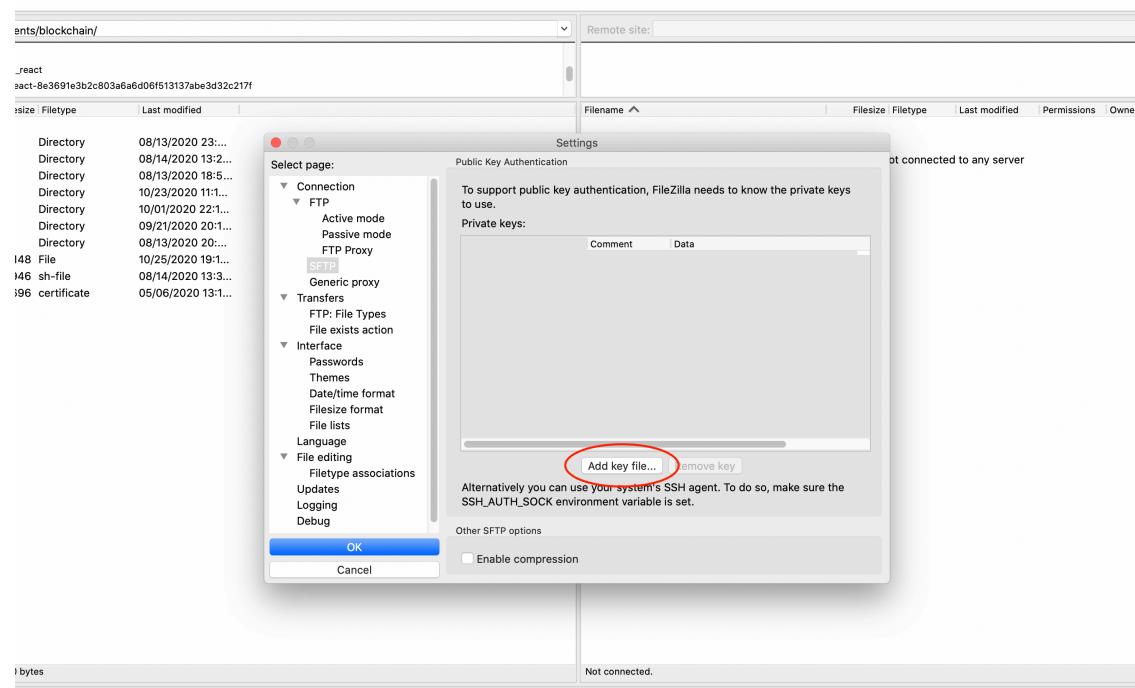
- If you want a more user-friendly tool to transfer data, FileZilla is the right choice. It is free, supports Windows/Linux/Mac systems, and has a friendly user interface. It supports FTP, SFTP, and other file transfer protocols. Go to <https://filezilla-project.org/> to download it.



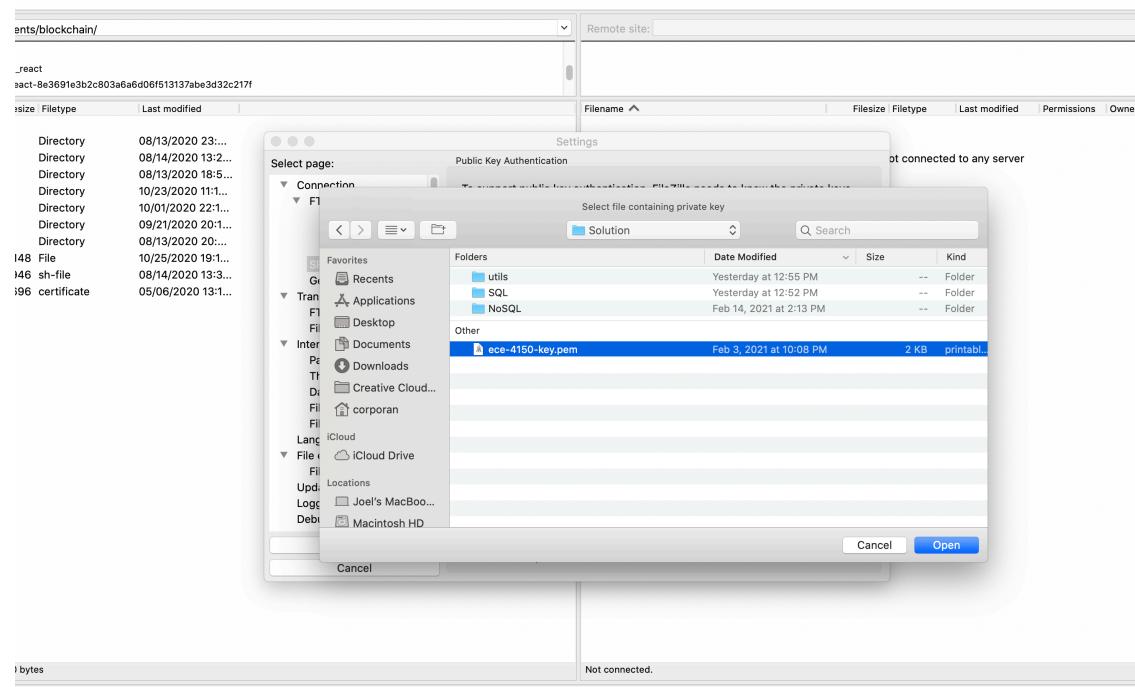
- If you want to use FileZilla to upload to or download data from a normal FTP server if you have the user and password, just put the information in the "Host", "Username", "Password" box and connect. However for Amazon instance, we use key-pair to log in instead of password for better safety. So it is a little bit more complicated to configure.
- Under "Settings" and click "SFTP":



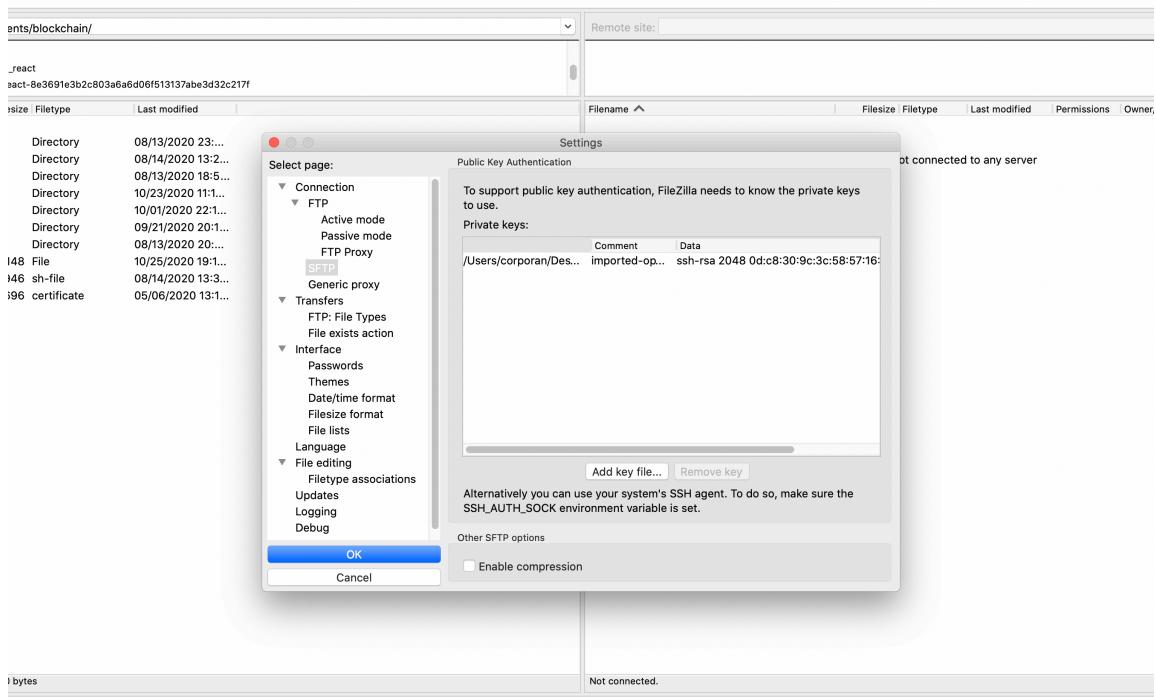
- Click "Add key file...":



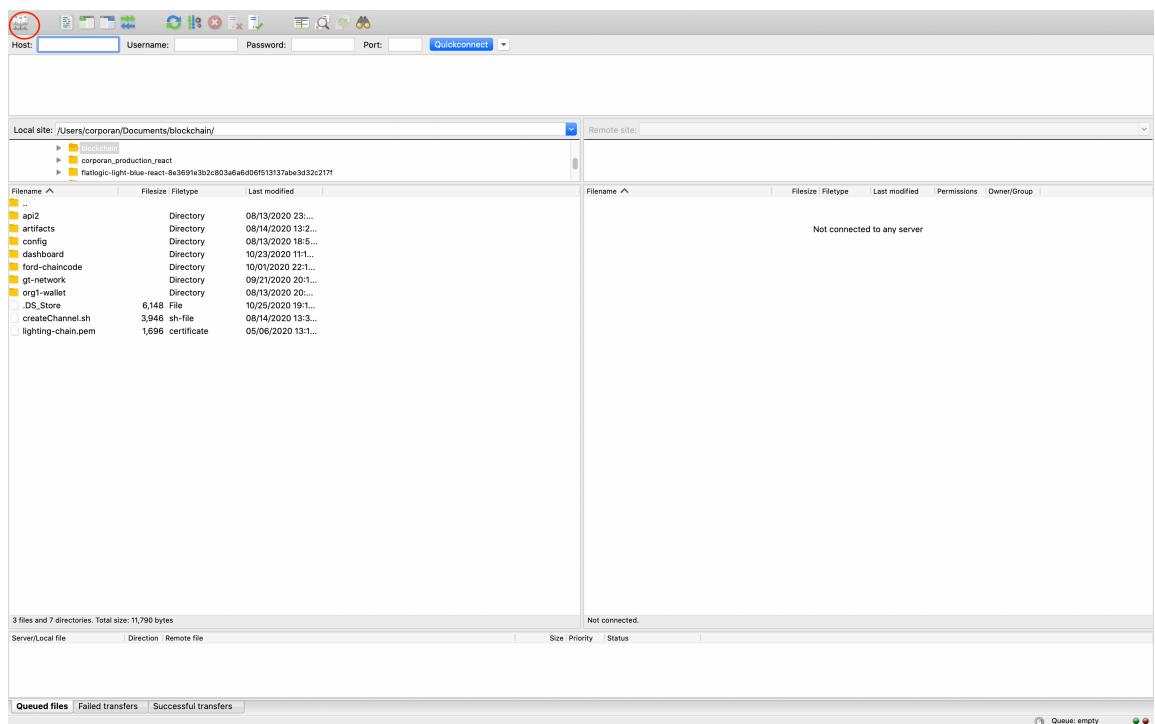
- Then select the ".pem" file you used to connect to Amazon instance using SSH.



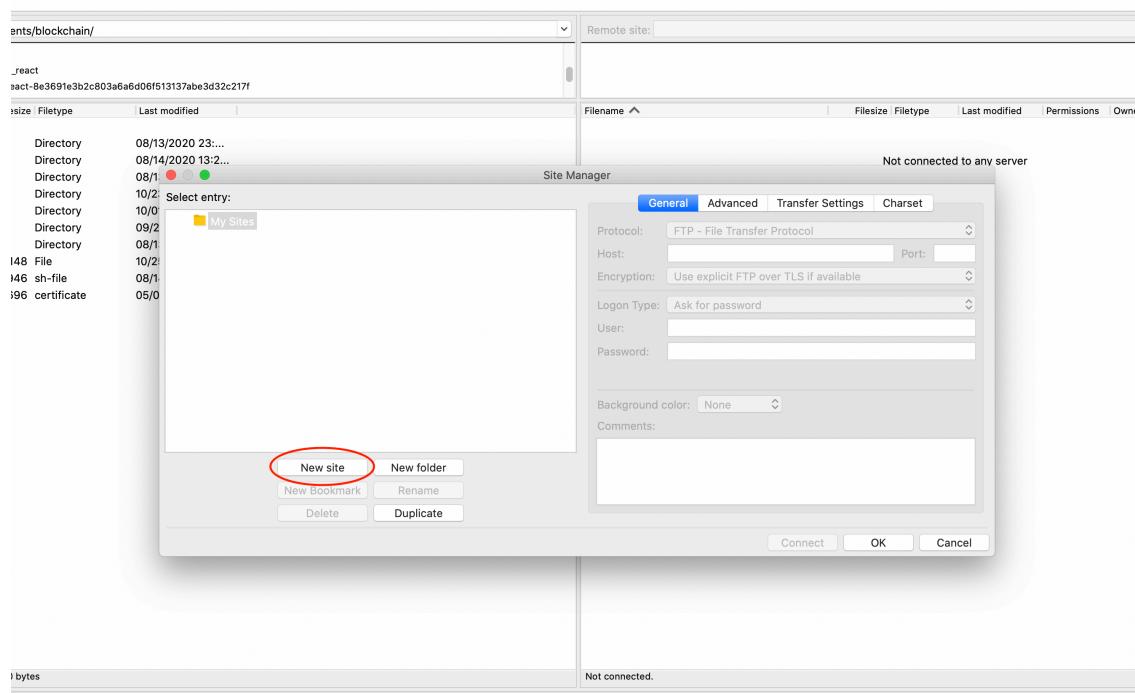
- You will see the a private key has been added.



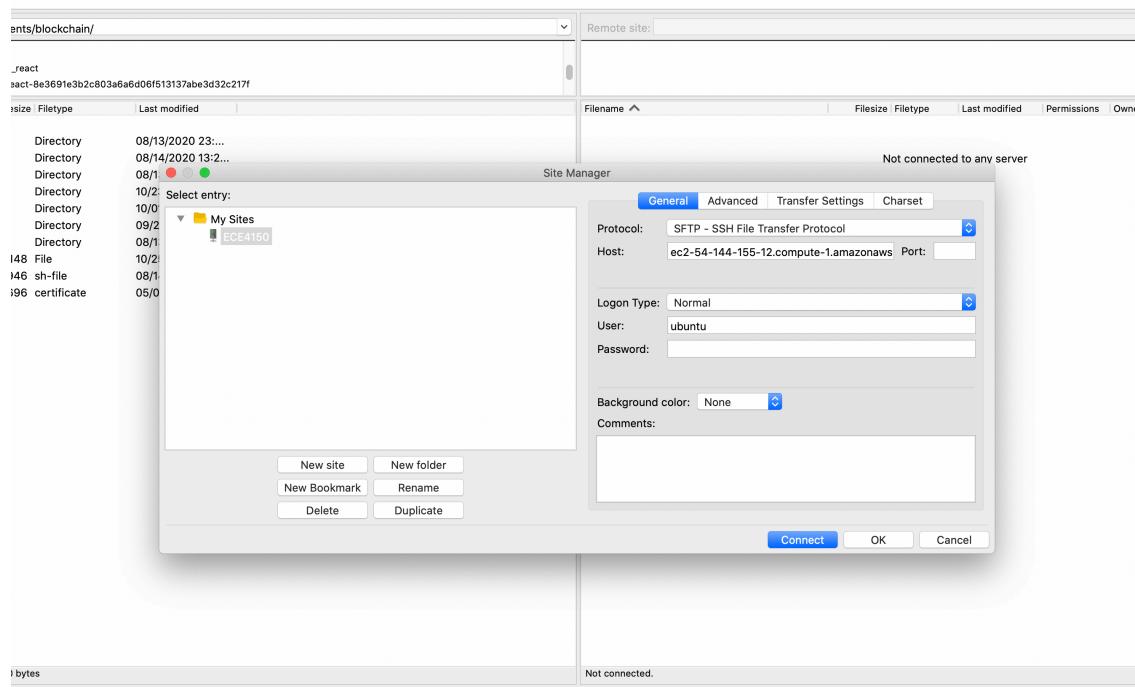
- Close "Settings" and go back to the main interface and click the button to open the "Site manager", as shown below



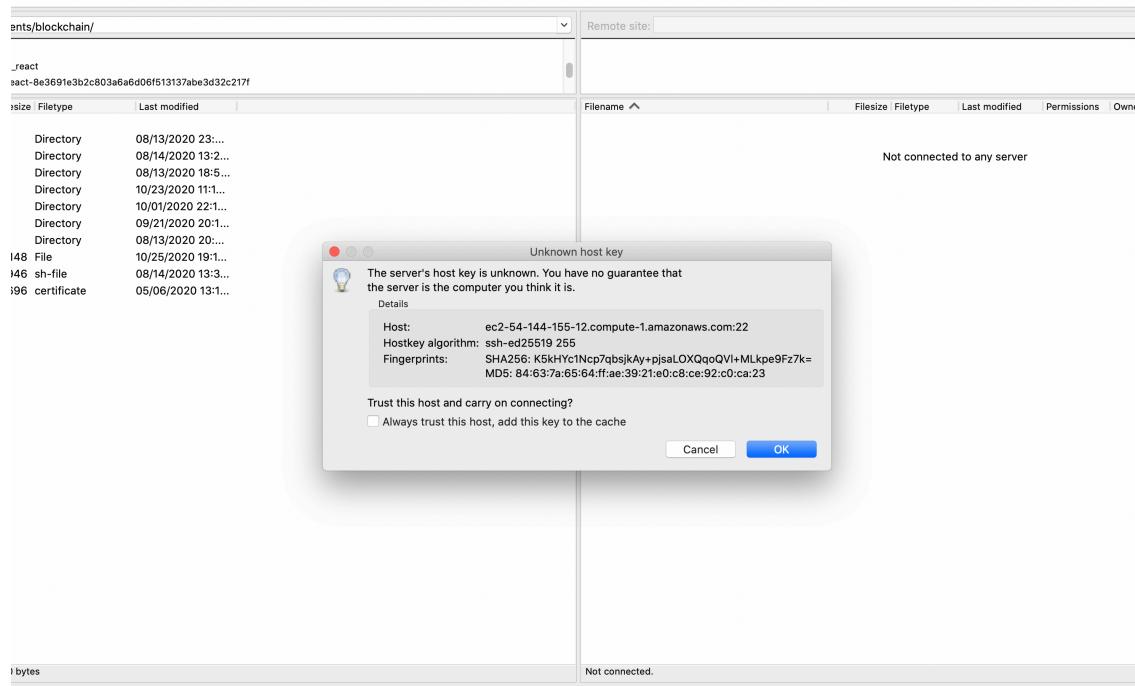
- Click “New Site” and give a name.



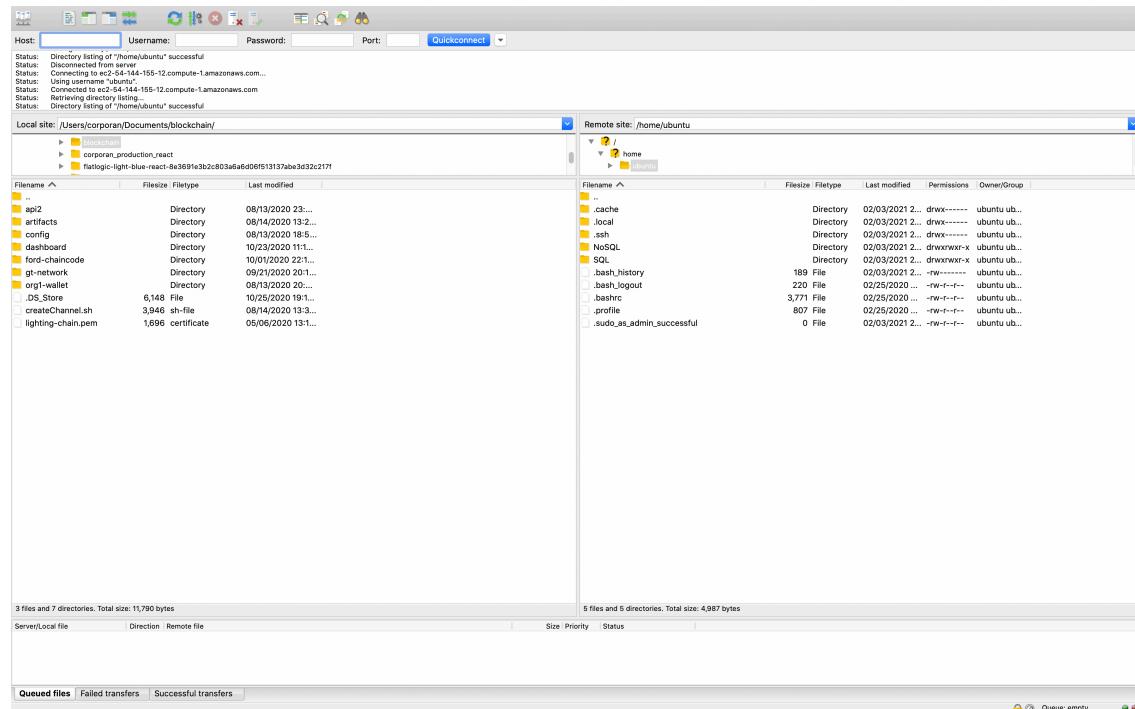
- Put the host URL of the EC2 instance in the “Host” box. Set “Protocol” as “SFTP”, “Logon Type” as “Normal”, “User” as “ubuntu” and leave “Password” as blank. Then click “Connect”.



- There will be a dialogue box to ask you about "Unknown host key", just click "Ok".



- All right. Now you have logged in the EC2 instance. You can drag and drop to transfer the files between your computer and the remote machine.



9. Getting familiar with the code and installing the necessary libraries inside the EC2 instance (2 points)

- Update your EC2 instance by running this command once you are inside the instance:

`sudo apt update`

- Take a look at the NoSQL and SQL directory and get familiar with the code provided to run the Flask web framework (app.py). To run the framework, we need a couple of libraries using the terminal. First, make sure you have the package installer pip3 and Python3 installed on the instance. You can use this command to instance these dependencies:

`sudo apt install python3-pip`

- You can also use Homebrew to install these packages (<https://brew.sh>) and any other method. After installing the Homebrew, use the following command to install the lastest version of Python3 and pip3:

`brew install python3`

*****The python version used for this lab was 3.9.0 and pip3 20.3*****

- Now, we will install a couple of libraries that we will use for both SQL and NoSQL variants. After installing python3 and pip3, install the following libraries using the package installer pip

`pip3 install exifread flask PyMySQL`

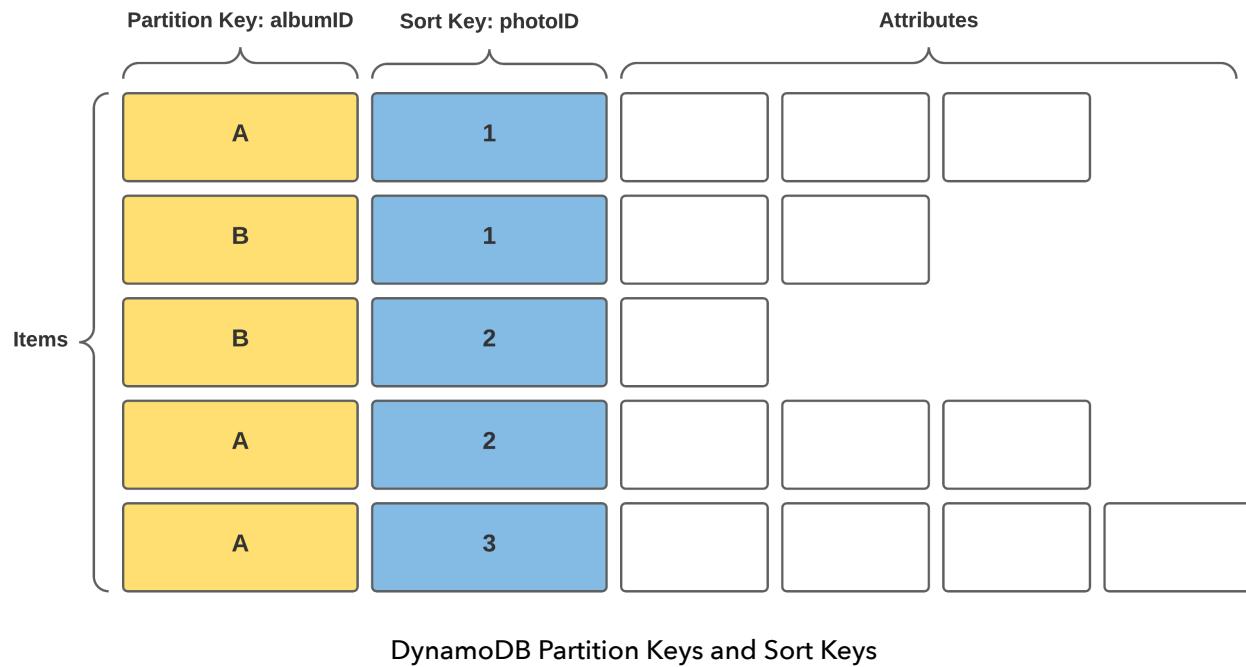
- If you get an error about missing libraries, use pip3 to install those libraries.
- Go back to the Security Group section under the EC2 Console and add another inbound rule for a **Custom TCP** type from port **5000** and configure the source coming from anywhere (**0.0.0.0/0**). This configuration will allow you to access your application inside the EC2 through the 5000 port.

You can also test each of the variants outside your EC2. If you have Python3 and pip3 installed on your computer, install all the dependencies mentioned above and run the following command inside the directory the variant that you would like to test:

`python3 app.py`

10. Photo Gallery using NoSQL (2 points)

- For the NoSQL variant, we will store the album and photo records within the same table using the partition key and sort key for the albumID and photoID, respectively.



- Inside the NoSQL directory, run the following command to run the Flask web framework using the NoSQL variant:

python3 app.py

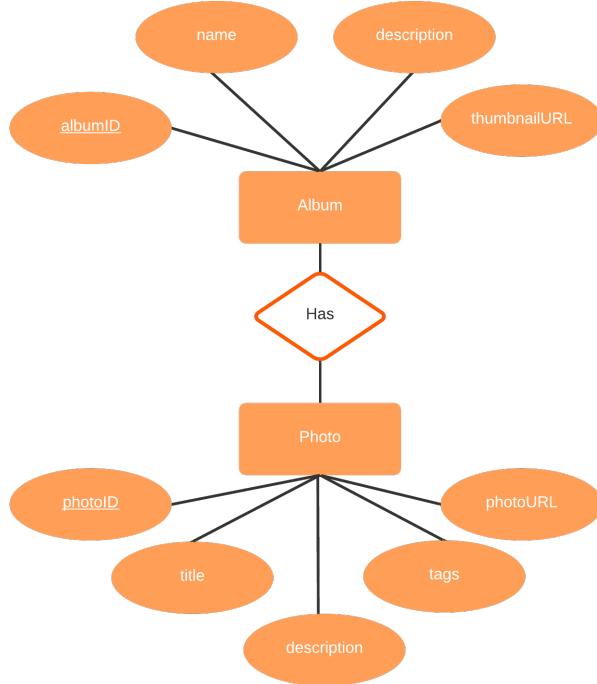
- Copy the hostname (e.g., URL or endpoint) from your EC2 instead of the link provided in the terminal when you run the Flask web framework. Add the port name (5000) and add it to the browser. It is advised to use the following browsers: Chrome and Firefox.

yourEC2InstanceHostName.com:5000

*****Remember to turn off your application if you want to use the other variant*****

11. Photo Gallery using SQL (2 points)

- For the SQL variant, we will store the album and photo records in different tables.



Entity Relationship Diagram that show the relation between albums and photos

- Inside the **utils** directory, you will find a **sqlcommands.sql** that includes the **Album** and **Photo** tables (the **User** table will be used in the last exercise).
- Within the same directory, you will find a python script to create the SQL tables for **Album** and **Photo**. Get familiar with the script's code structure because the method to call the RDS instance is in some way similar to the one used in the web framework. Run the following command to create the **Album** and **Photo** table:

python3 album-photo-tables.py

- Run the following command within the SQL root directory to run the Flask web framework using the SQL variant:

python3 app.py

- Copy the hostname (URL) from your EC2 instead of the link provided in the terminal when you run the Flask web framework. Add the port name (5000) and add it to the browser. It is advised to use the following browsers: Chrome and Firefox.

yourEC2InstanceHostName.com:5000

*****Remember to turn off your application if you want to use the other variant*****

12.Exercises: User Management and authentication, besides other CRUD operations, to support the Photo Gallery (60 points)

- Modify both NoSQL and SQL in the following exercises:
 1. Design and implement a new method for user management and authentication without using external services, such as AWS Cognito. **(30 points)**
 1. Allow users to create an account (sign up) and sign in (log in) through the website.
 2. Store the information of the user in a new table called **User** (For SQL variant, we provided a sample schema that you can use. However, you might need include other attributes within that schema. For NoSQL variant, create a new table called **PhotoGalleryUser**). A few requirements for the use case:
 - i. Use password hashing when signing up (**Appendix C**).
 - ii. The **userID** is a universally unique identifier (UUID).
 - iii. UserID and email **must be** unique. A user cannot sign up multiple times with the same email.
 - iv. To login the user uses the **email** and **password** to login.
 3. To validate a new account, send an email to the user (See **Appendix A**) with a confirmation token with a URL to the server (See **Appendix B**) (the confirmation token should include the **userID**). When the user clicks the email, the URL should route to a resource in your Flask web framework. The resource will take the incoming request, decode the token string to get the **userID**, and confirm the user in the database. The resource's response should redirect to the login page. A few requirements for the use case:
 - i. The user cannot use the website until the user is verified.
 - ii. The token should last 10 minutes. If the user does not click the link within that time, the token will be invalid.
 4. When the user logs in, the web server should store a session token in the user's browser. This token will be used to validate the navigation of that particular user in the website. The token should last only 5 minutes. After the token is expired, the user should be redirected to the login page.
 2. Add a new method to allow the user to cancel its account. **(15 points)**
 1. Remove all the albums created by a user when this one cancels its account. A few requirements for the use case:

- i. For NoSQL, attached a new attribute when the album is created to identify the creator.
 - ii. For SQL, update the album schema to add foreign key from the user schema. This strategy will allow you to remove the all albums created by the user.
3. Add a new method to delete a photo in the PhotoGallery. **(5 points)**
 4. Add a new method to delete a whole album in the PhotoGallery. This should delete all the photos inside that particular album. **(5 points)**
 5. Add a method to update the following information of a photo: title, description, and tags. For NoSQL, whenever you make an update, update the updatedAt attribute as well. In SQL, it is done automatically. **(5 points)**

*****For this exercise, there are designated sections (with comments) where you need to add your functions, routes, and environment variables needed. Make sure you use these sections to insert your code. If you need to modify any other function, route, or environment variable outside this section, put a comment that identifies your modification.*****

Deliverables:

1. A video showing all the full functionalities of your Photo Gallery application, including the ones in the final exercise. Please show your AWS username clearly to indicate that you are using your account. You may briefly explain the functions that you created for the last activity, **but please make sure to limit the total length to 12 minutes.**
2. The complete code with the modifications needed to complete each exercise, including the modified SQL schema (include the modified schema in the **sqlcommands.sql** file).

Appendix

A. How to generate confirmation token

The email confirmation should contain a unique URL that a user needs to click to confirm his/her account. Typically, a confirmation URL has this form:

```
http://yourEC2InstanceHostName.com/confirm/{id}
```

Where the id is an encoded string containing the email of the particular user, a timestamp, and salt (a random string used in hashed data to safeguard passwords in storage).

Python offered a lot of libraries to create this type of tokens. However, a recommended library for this lab is **itsdangerous** (<https://itsdangerous.palletsprojects.com/en/1.1.x/>). This library has a function called **URLSafeTimedSerializer** that allows to create a serialize (hash) token for URLs with a record of the time of the signing. It is a good strategy if you want to delegate the validation and expiration of signatures. Here is a simple example:

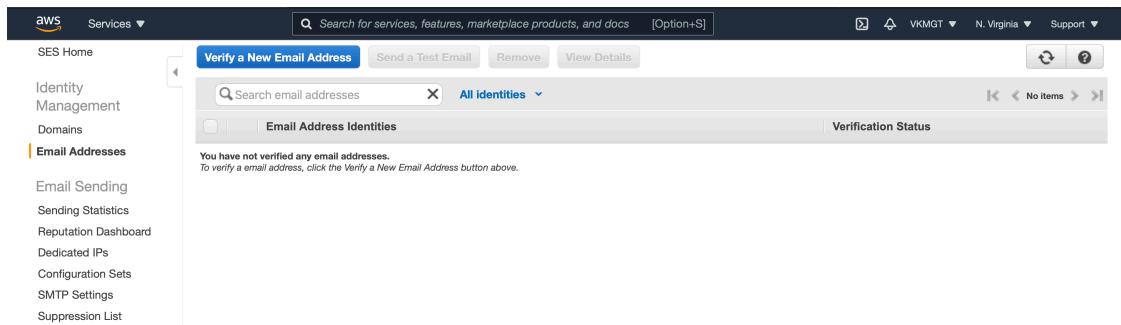
```
from itsdangerous import URLSafeTimedSerializer  
  
email = 'myemailaddress@gatech.edu'  
  
serializer = URLSafeTimedSerializer('some_secret_key')  
token = serializer.dumps(email, salt='some-secret-salt-for-confirmation')  
  
print(token)  
# eyJpZCI6NSwibmFtZSI6Iml0c2Rhbmddcm91cyJ9.6YP6T0Ba067XP--9UzTrmurXSmg  
  
try:  
    email = serializer.loads(  
        token,  
        salt='some-secret-salt-for-confirmation',  
        max_age=3600  
    )  
  
    print(token)  
  
    #myemailaddress@gatech.edu  
  
except Exception as e:  
    print('expired token')
```

B. How to send an email (AWS SES)

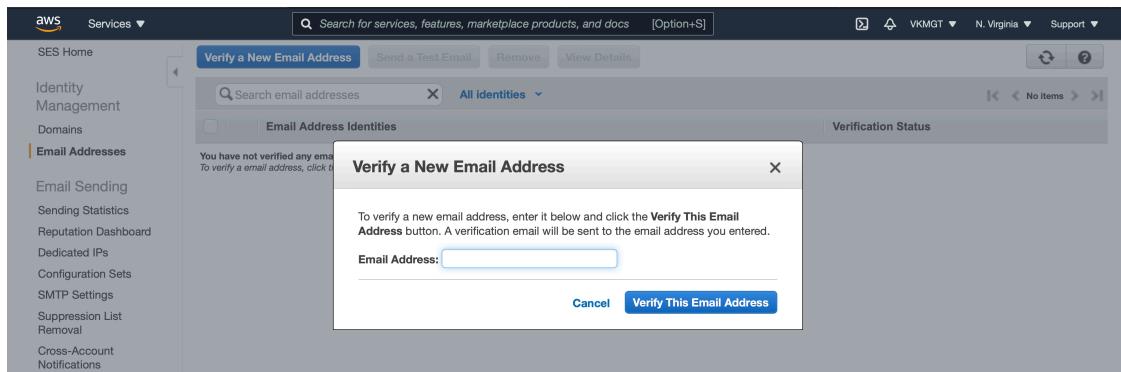
If you are trying to send a user an email to confirm their account, you need a transactional email service. AWS SES is the right solution for your application. Amazon Simple Email Service is a platform within AWS that allows you to send and receive emails using your email address or registered domain.

Amazon SES has proven to be a cost-effective email service. It is designed to send both bulk and transactional emails. AWS SES allows you to send emails directly from the SES console, via the Simple Mail Transfer Protocol (SMTP) interface, or through the API (You can use Boto3 to make calls to the API).

Before you can send emails with SES, you must first verify that you own the email address you wish to use as a sender. Navigate to Amazon SES, click "Email Addresses" in the sidebar, and then click the "Verify a New Email Address" button.



Enter the email you'd like to use and click "Verify This Email address".



The screenshot shows the AWS SES console under the 'Email Addresses' section. A search bar at the top right contains the placeholder 'Search for services, features, marketplace products, and docs [Option+S]'. Below the search bar are buttons for 'Verify a New Email Address', 'Send a Test Email', 'Remove', and 'View Details'. A 'All identities' dropdown menu is open. On the left, a sidebar lists various SES features: SES Home, Identity Management, Domains, Email Addresses (which is selected and highlighted in orange), Email Sending, Email Receiving, and Email Templates. The main content area is titled 'Email Address Identities' and shows a single entry: 'joel.cororan@gmail.com' with a status of 'pending verification (resend)'. Navigation icons for back, forward, and search are visible at the top right of the main content area.

Then, after clicking the verification link in your email inbox, you should see your email verified back on SES.

This screenshot is identical to the one above, but the email address 'joel.cororan@gmail.com' now has a green 'verified' status indicator next to it, indicating that the verification process has been completed.

New accounts are automatically placed in a sandbox mode to help prevent fraud. You can only send emails to addresses you have personally verified with Amazon. If you want to remove this restriction, you must request Amazon to move out of the sandbox mode. Fortunately, this is enough since you are not deploying this application to a production environment.

Below is an example of how to send an email using boto3:

```
import boto3
from botocore.exceptions import ClientError

# Create a new SES resource and specify a region.
ses = boto3.client('ses',
                    region_name='AWS_REGION',
                    aws_access_key_id='AWS_ACCESS_KEY_ID',
                    aws_secret_access_key='AWS_SECRET_ACCESS_KEY')

SENDER = 'MyOtherVerifiedEmailAddress@gatech.edu'
RECEIVER = 'myVerifiedEmailAddress@gatech.edu'

# Try to send the email.
try:
    #Provide the contents of the email.
    response = ses.send_email(
        Destination={
            'ToAddresses': [RECEIVER],
        },
        Message={
            'Body': {
                'Text': {
                    'Data': 'This is an email from AWS SES',
                },
            },
            'Subject': {
                'Data': 'Hi, I\'m sending this email from AWS SES'
            },
        },
        Source=SENDER
    )

    # Display an error if something goes wrong.
except ClientError as e:
    print(e.response['Error']['Message'])

else:
    print("Email sent! Message ID:")
    print(response['MessageId'])
```

C. Password Hashing

As we know now, it is a terrible idea to stored a password as text without any encryption. For that reason, you do not want to include passwords in your database as plain text, as this would make your users' passwords exposed if your server got hacked and your database vulnerable. That is why you need to protect sensitive data by using a hash method before storing them in your database. This is a simple step to provide a lot of security.

The article below from Dustin Boswell will help you to understand and implement hashing algorithms, such as **bcrypt** (recommended by the author). Also, you will find a flask version of **bcrypt** below.

<http://cs.wellesley.edu/~cs304/lectures/bcrypt/dustwell.html>

<https://pypi.org/project/bcrypt/>