COSC349 - Assignment 2

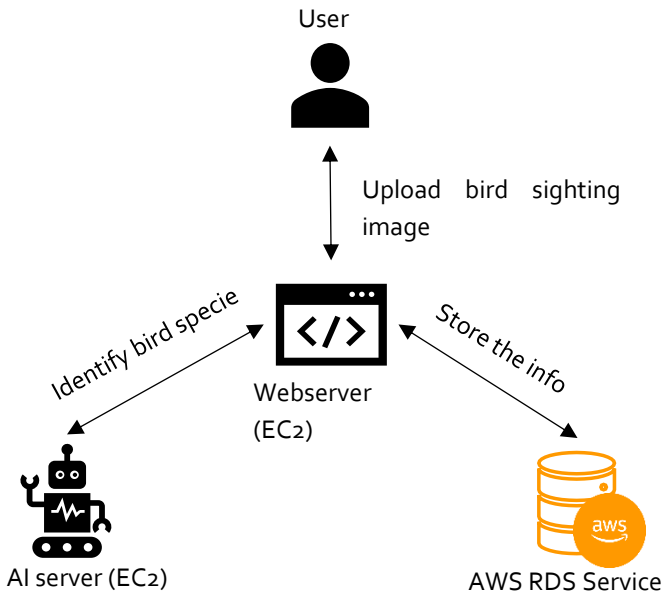
CLOUD PROVISIONING

Report

Name Ubaada Altaf Qureshi

ID 5827382

DESIGN OF VMS AND CLOUD SERVICE



The project is called "**AI-Birdwatcher-Diary-AWS**".

The project for Assignment 2 is a fork of the project from Assignment 1 "AI-Birdwatcher-Diary" used to identify bird species from images. This project is modified to be deployed on Amazon's cloud platform. It consists of 3 VMs all under the **same Security Group** with appropriate rules so they can communicate with each other.

Webserver:

Vagrant deploys the web server as an EC2 instance.

AI server:

AI server is also deployed as an EC2 instance. Vagrant sets a static private IP for the AI server so that the web server knows how to connect to it. Otherwise a random private is assigned to the VM.

DB server:

The project uses AWS RDS, a database service provided by Amazon. First a database instance is manually created using AWS RDS service. Then the details of the database configuration are put in to a php file. The file contains 4 variables i.e. database host, name, username, and password. The table setup script then uses these details to create the required table upon provisioning. The configuration file is also used by the php scripts in the web server to store, delete, and fetch data. Since the VMs are agnostic to the type of service running the database any type can be used.

Note: After creating a new database instance, Amazon Cloud may take up to 30 minutes for the database to be ready for table creation. Hence it is better to do it beforehand rather than creating it during vagrant provisioning.

DEPLOYMENT

1. Set up a Security Group in AWS console:

Inbound Rules:

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	Public web access
HTTP	TCP	80	::/0	Public web access
All TCP	TCP	0 - 65535	[id-of-same-security group]	Inter-VM (ec2-db) Communication
SSH	TCP	22	0.0.0.0/0	SSH for admin

Outbound rules:

Type	Protocol	Port range	Source	Description - optional
All traffic	All	All	0.0.0.0/0	Allow all outbound traffic

2. Set up SSH:

For SSH to work create a key-pair named "cosc349-l.pem" in AWS console. AWS does not allow SSH login via passwords.

Place the key file in `~/.ssh`

3. Create an AWS RDS MySQL database using AWS console.

4. Clone the git repo and CD into it. Edit the VMs in Vagrantfile to have your AWS Security Group.

5. Set up database details in `.config/db-config.php` file in the cloned repo:

```
<?php
$db_host   = '[end-point-here]:[port-number-usually-3306]';
$db_name   = '';
$db_user   = '';
$db_passwd = '';
?>
```

6. Install vagrant-aws plugin: `$ vagrant plugin install vagrant-aws`

7. Set up the AWS environment variables for Vagrant to be able to talk with AWS.

8. Start Vagrant deployment: `$ vagrant up --provider=aws`

ACCESSING APP

Go to

```
http://ec2-3-88-3-5.compute-1.amazonaws.com/
```

If you are deploying the app yourself, go to the AWS console and look for the Webserver EC2 instance's public IP or DNS Name. Navigate to that address.

EVIDENCE OF DEPLOYMENT

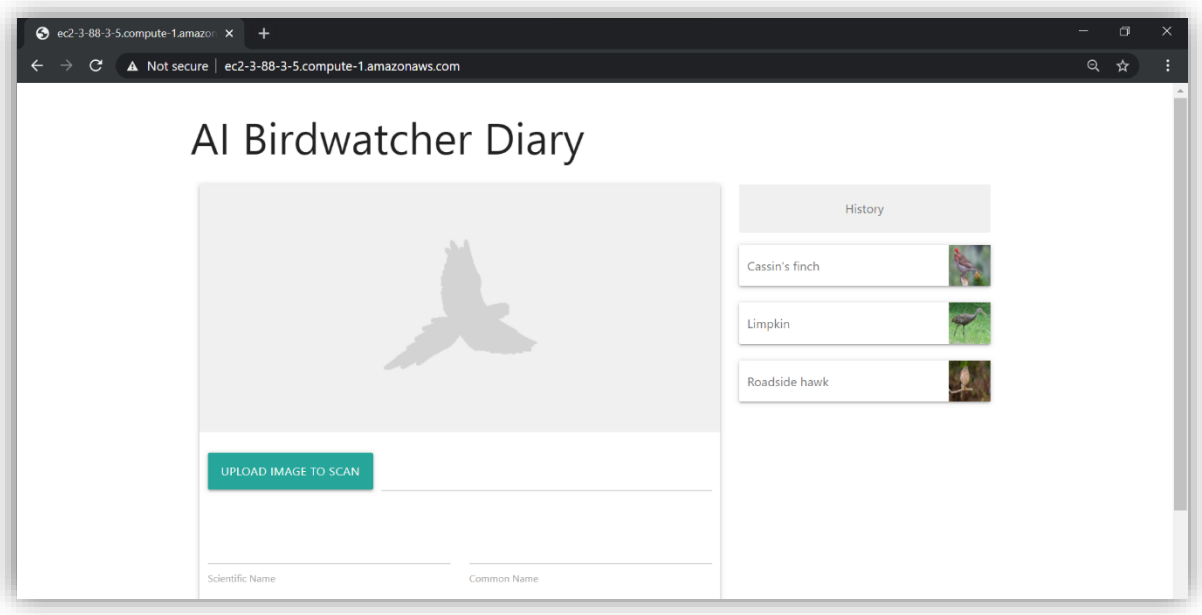


Figure 1: App reachable on the web. Webserver successfully fetches records from the database.

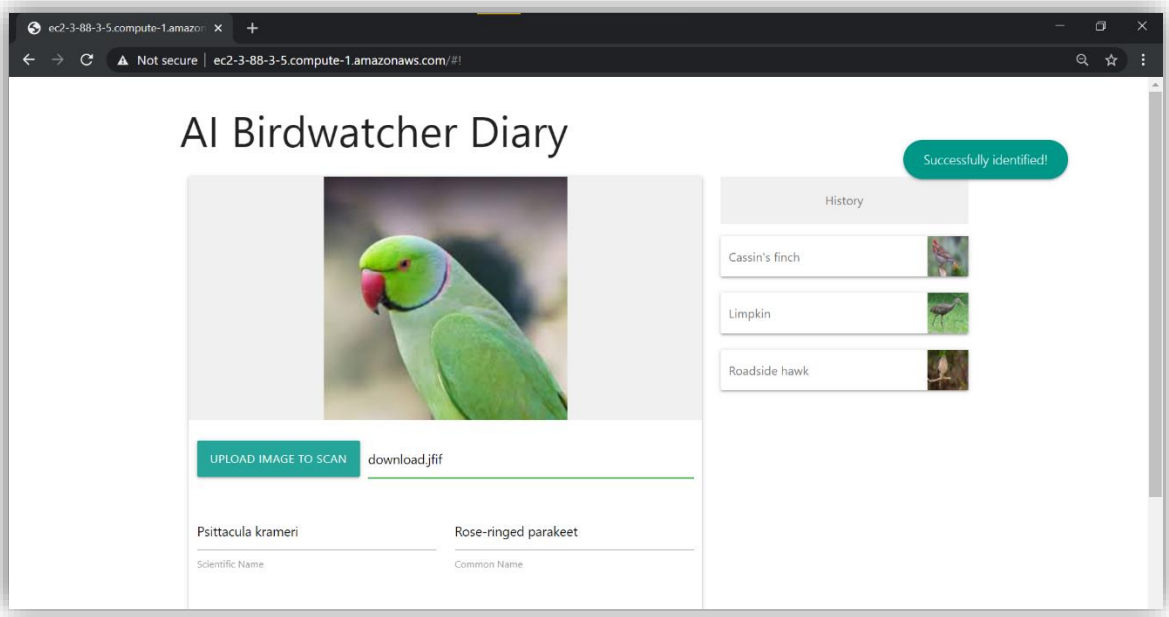


Figure 2: App successfully identifies the bird using the ai server.

Complete video demonstration: `https://streamable.com/soaje`