



ECE 1779: Introduction to Cloud Computing

Project 1: Amazon EC2

Lihua Shen (1002959934)
Yiren Shen (1003314219)

1 Description of the general architecture:

1.1 Description:

We developed a web application which contains userUI and ManagerUI that can resize its worker pool based on user demand.

1.1.1 UserUI

UserUI was mainly implemented by three python documents: 'user.py', 'image.py', 'load_test.py' and several html documents, which are listed as follows.

The 'Login' Part:

- index.html
- login.html
- register.html
- register_success.html

The 'Image' Part:

- image_list.html
- image_upload.html
- image_view.html

1.1.2 ManagerUI

ManagerUI was mainly implemented by four python documents: 'manager.py', 'grow_shrink.py', 'auto_scaling.py', 'delete.py' and several html documents, including 'index.html', 'worker_pool.html', 'worker_view.html', 'grow_shrink_complete.html', 'thre_conf.html', 'auto_scaling.html', 'delete.html' and 'delete_complete.html'.

1.2 Design:

We created three AMIs (Amazon Machine Images) which are for database, workers and manager.

- Database AMIs: We created a database with tables of images and users and provided a remote access user. We saved user information such as user ID, passwords, etc. in the user table and images users uploaded in the image table.
- Worker and manager AMIs provide related code, credentials, etc. to play their roles separately. In addition, we can create worker's instance quickly in the manager part by taking use of worker AMI and then the new worker instance can be registered into load balancer and provide user service automatically.

AMI List:

- Worker AMI ID: ami-e240e9f4
- Manager AMI ID: ami-a817bebe
- Database AMI ID: ami-9f812f89

1.3 Main technologies:

In our application, the main AWS technologies used are listed as follows:

- EC2

Boto3 is a Python library for interfacing with Amazon Web Services. Launching new instances requires an image ID and the number of instances to launch. Boto 3 collection filtering was used for terminating multiple instances given a list of instance IDs. A filter() method of the instances collection was used to retrieve all running EC2 instances.

- S3

In our application, all the images and their transformed versions are stored on S3.

- S3 bucket name: ece1779b1

- CloudWatch

The AWS CloudWatch API was used to monitor the load on the pool of workers.

- Load Balancing

EC2 Load Balancing was used to distribute request among your workers.

- Load Balancer name: syr-load-balancer

- AMI

An Amazon Machine Image (AMI) provides the information required to launch an instance, which is a virtual server in the cloud.

2 Instructions:

Our web application provides two web-based interfaces: User UI and Manager UI.

In terms of User UI, use the address <http://syr-load-balancer-810504416.us-east-1.elb.amazonaws.com/> to start the application.

In terms of Manager UI, use the address <http://ec2-54-165-94-138.compute-1.amazonaws.com:5000/> to start the application.

2.1 AWS Management Console Access

- URL: <https://565934887570.signin.aws.amazon.com/console>
- User name: ece1779ta
- Password: ece1779

If you do not have enough access to our AWS account , please sign-in using root account credentials

- User name: yiren.shen@mail.utoronto.ca
- Password: 198784syr

2.2 AWS Configure

- Access Key ID: AKIAJEJTJNOCPWSCBNIA
- Secret Access Key: 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

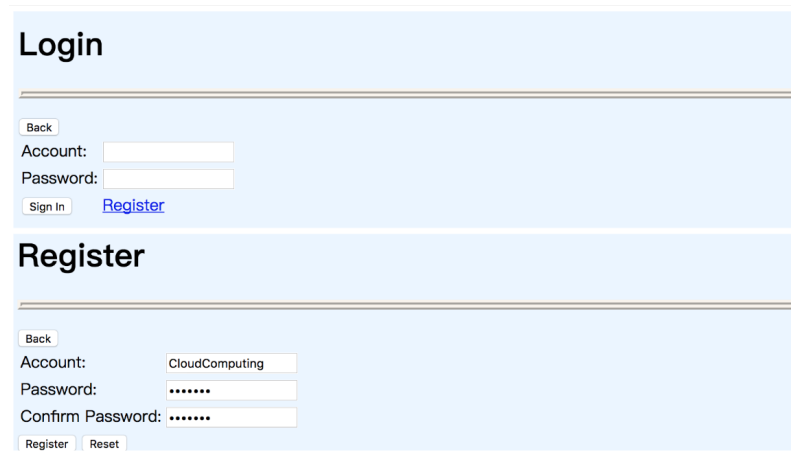
2.3 User UI:

The main tasks that a user can perform can be concluded as follows:

- Upload a new image;
- View a page with thumbnails of all the images that a particular user has uploaded;
- View a specific image and three transformed versions of it.

The detailed information is shown as follows:

1. First, our user UI part provides a welcome page that lets returning users log into their account, and enables new users to create a new account as well.

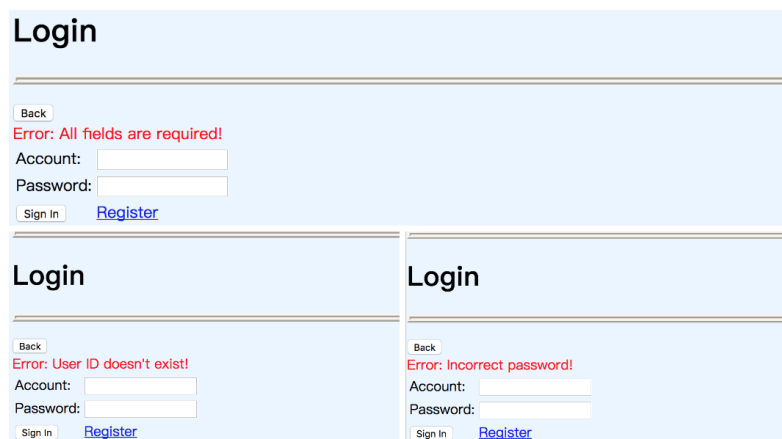


The image shows two web forms side-by-side. The top form is titled 'Login' and contains a 'Back' button, 'Account:' and 'Password:' labels with input fields, and 'Sign In' and 'Register' buttons. The bottom form is titled 'Register' and contains a 'Back' button, 'Account:' label with an input field containing 'CloudComputing', 'Password:' and 'Confirm Password:' labels with masked input fields, and 'Register' and 'Reset' buttons.

In our program, we defined a `login_verify()` function to verify the returning users. There exist some situations where an error message will be shown on the web page, which are shown as follows:

- Users didn't fill the "Account" or "Password" blanket before they clicked the "Sign In" button;
- Users intended to sign in but they hadn't registered before;
- The password that the user entered was wrong.

When the above mentioned situations happen, a corresponding error message in red color will be shown on the web page to show the users what kind of mistakes they have made.



The image shows three instances of the 'Login' form. The top form has a red error message: 'Error: All fields are required!'. The bottom-left form has a red error message: 'Error: User ID doesn't exist!'. The bottom-right form has a red error message: 'Error: Incorrect password!'. Each form includes a 'Back' button, 'Account:' and 'Password:' labels with input fields, and 'Sign In' and 'Register' buttons.

2. After a particular user signs in, he can press the “Image List” button to enter a page where he can see the thumbnails of all the images that he has uploaded before and can also upload a new image as he likes on the same web page. If he intends to log out, he can press the button of log out on this web page.



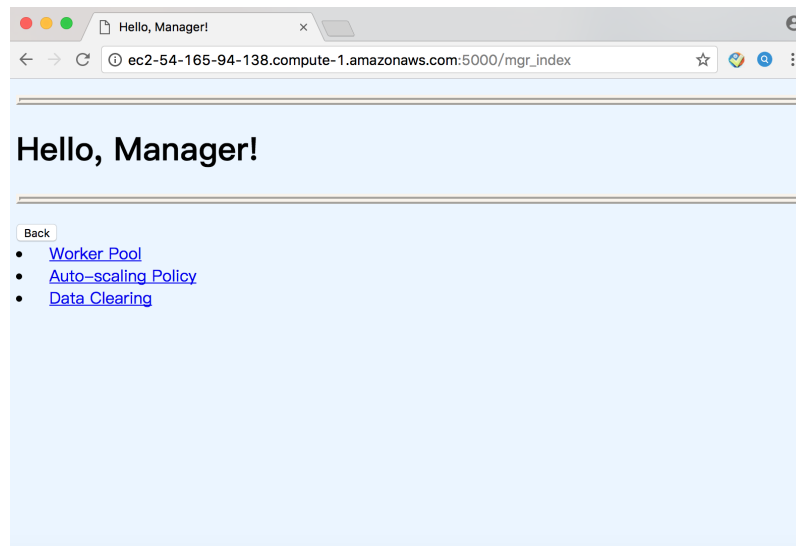
3. If we click on the thumbnail of a particular image, we will then enter a new web page where the original image and three transformed versions of it are shown.

The same thing will happen when we try to upload a new image, which is to say we can see the original image and three transformed versions of it after we successfully upload a new image. In the meantime, a message “Upload Completed!” will appear on the web page to tell the user that he has uploaded the image successfully.

2.4 Manager UI

In terms of the manager UI part, our application mainly supports the following functionality:

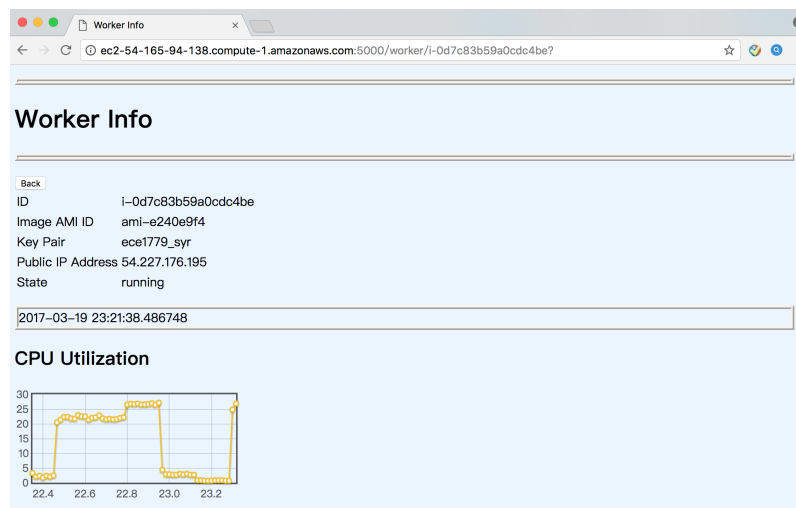
- List workers, and their CPU utilization
- option for manually growing the worker pool.
- option for manually shrinking the worker pool.
- option for configuring the auto-scaling policy by setting the following parameters:
 1. CPU threshold for growing the worker pool
 2. CPU threshold for shrinking the worker pool
 3. Ratio by which to expand the worker pool (e.g., a ratio of 2 doubles the number of workers).
 4. Ratio by which to shrink the worker pool (e.g., a ratio of 4 shuts down 75% of the current workers).
- An option for deleting all data. Executing this function should delete application data stored on the database as well as all images stored on S3.



The detailed instructions for this part are shown as follows:

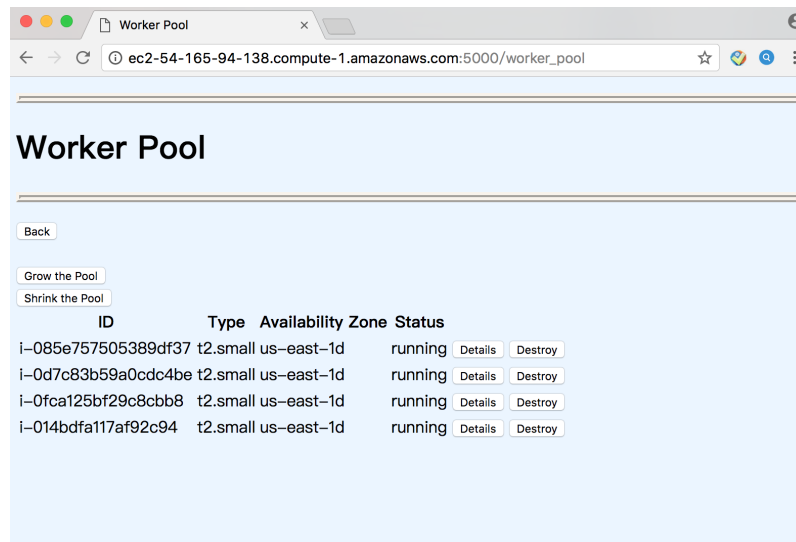
1. List workers, and their CPU utilization:

Clicking the “Worker Pool” button on the manager UI welcome page will lead us to a web page that lists the detailed worker information which includes worker ID, type, availability zone and status. Furthermore, by clicking the “Details” button beside each worker, we can also get much more information which includes image AMI ID, key pair, public IP address and current time information. In addition, we can even visualize the CPU utilization.



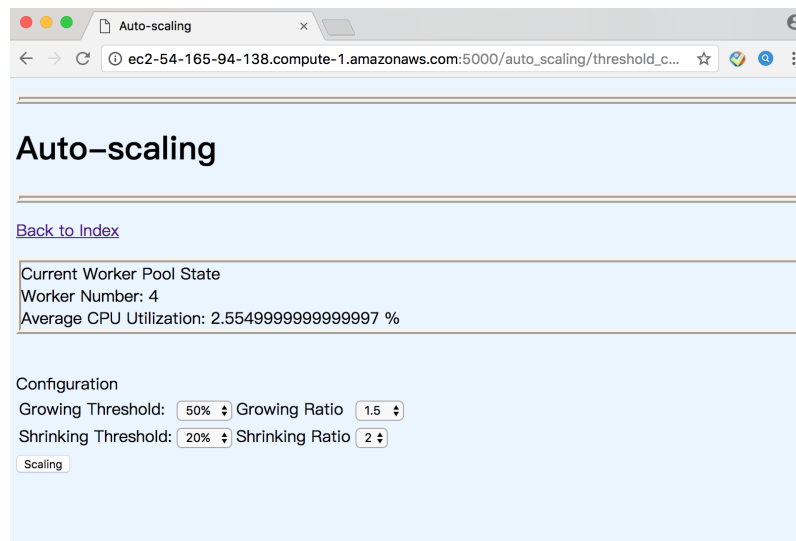
2. An option for manually growing shrinking the worker pool:

On the “Worker Pool” page, we can also click the “Grow the Pool” button to add a new worker and click the “Shrink the Pool” button to terminate the first worker that was listed. If we want to terminate a worker other than the first one listed on the web page, we can also use the “Destroy” button beside the worker.



3. An option for configuring the auto-scaling policy:

If we click the “Auto-scaling Policy” button on the manager UI welcome page, we will be led to the “Auto-scaling” page. On this page, we can see detailed information about the current worker pool state including worker number and CPU utilization. Under the information bar lies the configuration part. We can freely set four parameters: CPU threshold for growing the worker pool, CPU threshold for shrinking the worker pool, ratio by which to expand the worker pool and ratio by which to shrink the worker pool. By clicking the “scaling” button, we can automatically control the CPU utilization of current workers. To illustrate in more details, our application monitors the load on its pool of workers using the AWS CloudWatch API. When the load exceeds or drops below the configurable threshold, our application will use the AWS EC2 API to resize its worker pool. In addition, it is worth-mentioning that EC2 Load Balancing was used to distribute request among the workers. In our application, the AWS Elastic Load Balancing API was applied to add and remove workers from the load balancing pool as we reconfigure our worker pool in response to load.



4. An option for deleting all data:

If we click the “Data Clearing” button on the manager UI welcome page, all the application data stored on the database as well as all images stored on S3 will be deleted.

- Instructions for starting the application, including details on any AMI(s).
- Any required credentials.
- Anything else you think is needed to understand your code and how it works.

