**Final Report for Assignment 2**

**Paper number: 158.383**

**Paper name: Information Technology Project**

**Group members: *Chen Songyue***

***Stanley Roy Barnes***

***Jacob Ratima***

***Wang Yacong***

**Summary**

We have been tasked with developing our initial solution for support of teaching the first year course 158.120. Our initial solution provided the teachers and administration staff with a simple way of setting up students with their own instances of moodle of the aws platform. What we have now done is implement two new areas into our solution, these being system provision and system operation. The areas within system provision consist of advanced system architecture, automation in system creation, cost forecasting/optimization and communication with students. The areas within system operation consist of monitoring of system operation, cost monitoring and security monitoring. For advanced system architecture, instead of emailing the instance ids to each student we have created a website that they can visit and choose an instance themselves. This means that students who drop out of the course will not need an instance and it can be deleted. We have improved the automation of system creation by setting up one instance, installing moodle onto it and then cloning that instance for as many students as there is in the course. This makes the creation of instances much more quicker than our initial solution. For cost forecasting/optimization we have implemented a script that can be run on the admins page on our website that will forecast host much it will cost between two dates that can be set. For the communication with students we have created a chatroom where the students and teachers or admin staff can access which will allow them to communicate directly with each other. We also set up a web page to help students when they forget their moodle password or when they want to change their FTP password.When students forget their password, they no longer need to send an email to the teacher or contact the administrator to fix it, which saves students a lot.Through this page, students can better use moodle to study.

For our second focus point and the monitoring of system operations we have implemented a script that is called with the press of a button on the admins page of our website that will give them a list of all instances and the state they are currently in, that being pending, running or stopped. This will allow them to be able to quickly tell which instance is having problems if a student is having trouble. For cost monitoring we have created script on the admins webpage that is run when the admin enters a range of dates. This will run and show them details of the costs on their aws account. This will allow them to manage the costs and the way they have been using their account. Lastly for our second focus, for security monitoring we looked into ways to stop students from selecting multiple instances or people who aren’t enrolled from selecting them in general.

**Description of Base Solution**

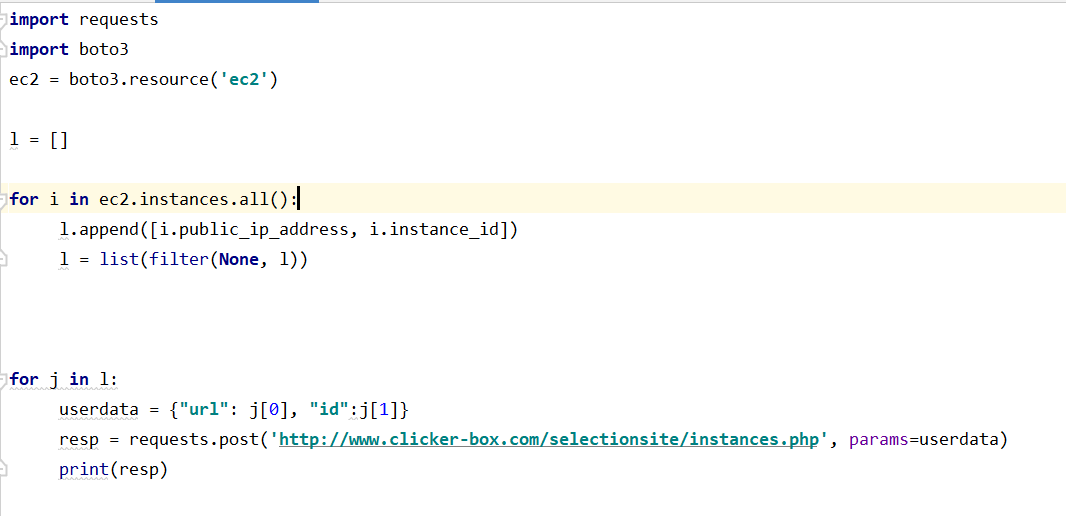
|  |  |
| --- | --- |
| Architecture Choice | We have created a system that sets up as many ec2 instances as there are students in the class and then SSH into all of them one by one to install moodle on each of them. An email script is then run to send out the relevant information for each student to log into their instances. |
| Tools which have been used | * Python * AWS cli * gmail |
| Cloud Provider | Amazon Web Services |
| Specification of virtual machines | * Size of VM = t2.micro * Storage = EBS * Memory = 1 GiB * Processor = 3.3 GHz Intel Scalable Processor |
| Summary of setup steps | * Read CSV to find how many students in the class * Automatically create as many instances as there are students in the class * SSH into each instance one by one and install moodle on each one * Email script is run to send relevant information to students |
| Functionality | * Python script reads how many lines there are in the csv file, this is how many students are in the class * We used boto3 commands along with our own script that we create to set up the instances * Using various python modules including paramiko we were able to SSH into each instance |
| Cost overview | Based on a class of 100 it is estimated the cost to be $3500 |

**Description of first focus**

**System Provision**

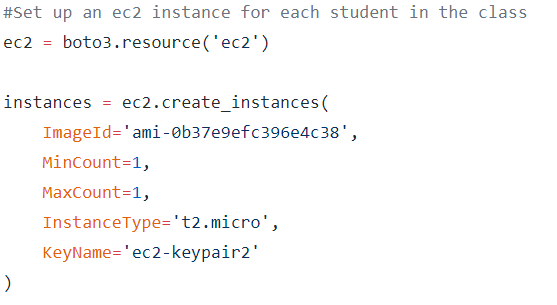
**Advanced System Architecture**

Our first focus area was system provision which consisted of four areas, these being advanced system architecture, automation in system creation, cost forecasting/optimization and communication with students. For the first area advanced system architecture, in our first solution we automatically emailed all of the students their instance ids. The students require their instance ids in order for them to log into their moodle site. For our new solution we have created a website that students can log into which then allows them to select an instance. Once one instances have been selected no other user can select it. Our website platform is advantageous because if some students drop out of the course after they have enrolled, then they will not have the chance to select an instance therefore it will be shut down. With our previous solution, if a student were to drop out of the course, the administration staff would have no idea and their instance would remain running which would result in more costs for the university. We implemented this by using php to set up a webpage. We then created a script that we could give python data to which would then update the website accordingly.

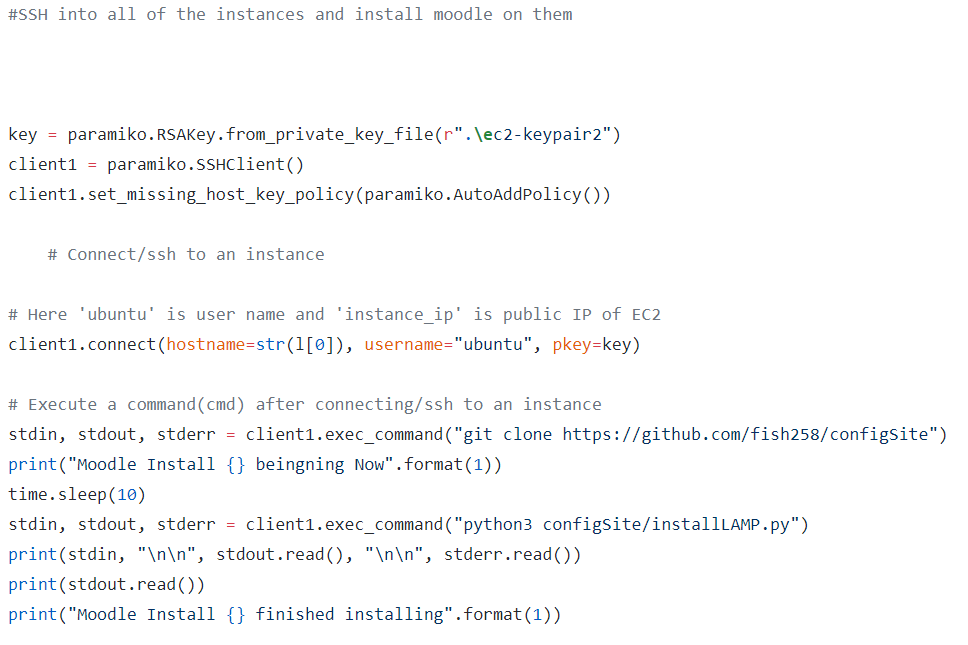


The code above loops over every instance on our account and then gets every instance id and stores them in a list. We can give this list to the userdata which is then sent over to the website and is updated accordingly along with the database.

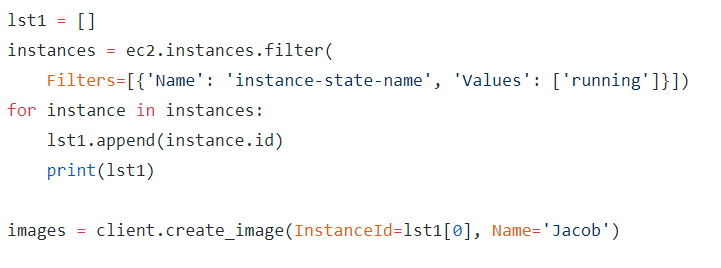


**Automation of System Creation**

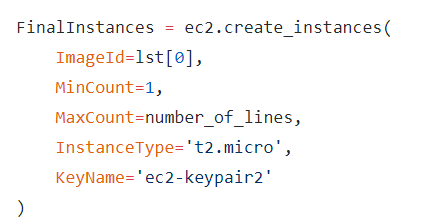
For the second area, automation of system creation we have created a system where we set up an instance which is then cloned into however many students there are in the course. Our first solution sets up an instance for each student which then installed moodle on each instance one at a time. This process took roughly 7 hours to set up a class of 150 students, whereas our new solution would take less than 1 hour. This is a big advantage for teaching and administration staff as they would be able to manage their time a lot better now that our solution will save them roughly 6 hours in setting up time. We implemented this by using our initial solution of creating instances to set up one and installed moodle on it.



We then created a python script using boto3 to take a snapshot of this instance and everything on it



Finally we set up as many instances as there are students in the class with the snapshot ami



**Cost forecasting/optimization**

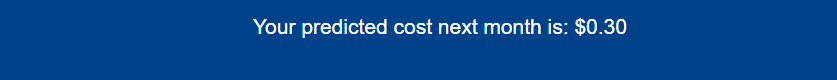
For the third area cost forecasting/optimization we have created a script that is run when two dates are input into a form on the admins webpage that will then Tell them what the predicted cost is for the time between those dates. In our previous solution we had no way of forecasting the cost of our account and would have to set up another service on our cloud platform to allow this to happen. It is good to have this feature available to the administration staff because they will accurately be able to plan how much the system is going to cost for the rest of the semester. We implemented this by creating a form that is in the body of the webpage. This form can be input with two dates.



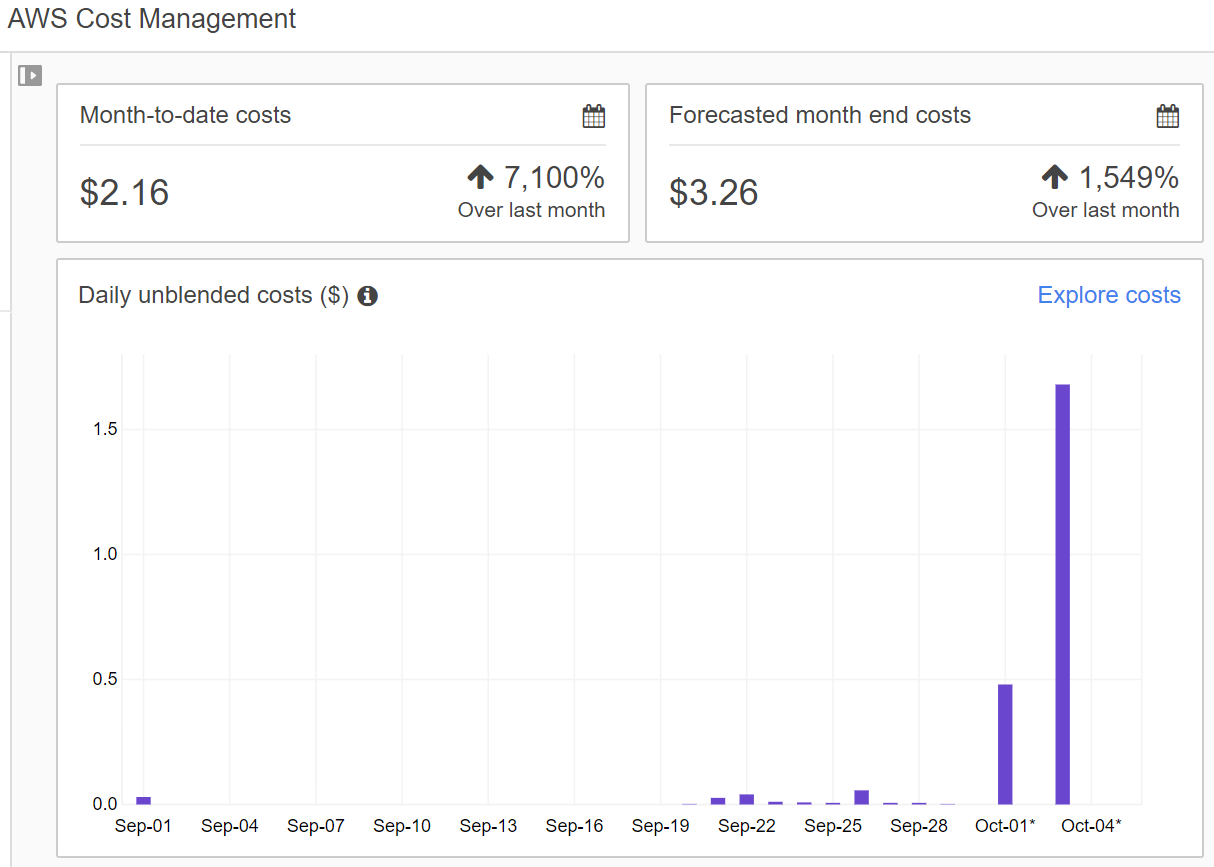
Once this is done it will call our python script which uses boto3 to create a prediction from previous usage of the cost between these two dates.



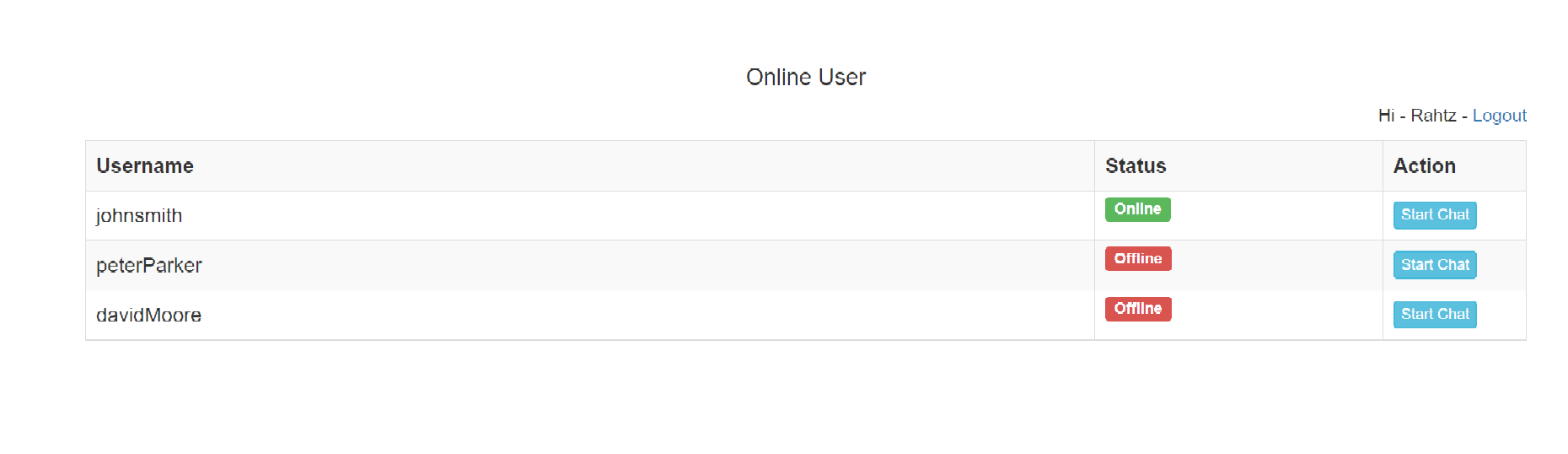
We display this on the web page for an admin user to look at for cost forecasting



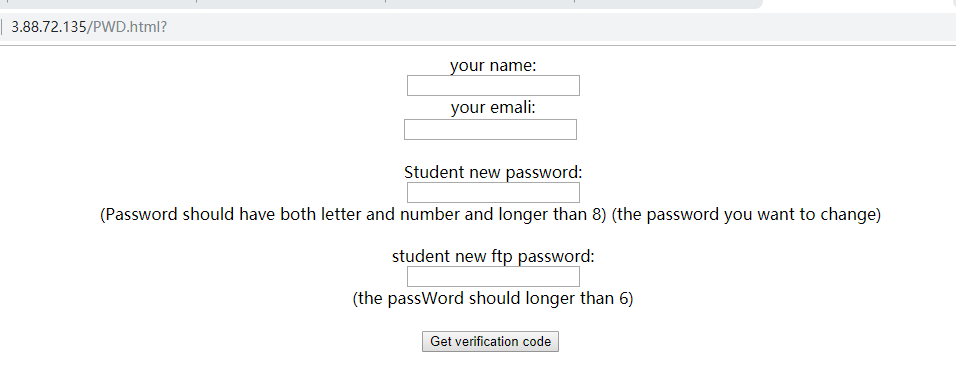
To enable this all to work we needed to activate cost explorer which comes with your aws account



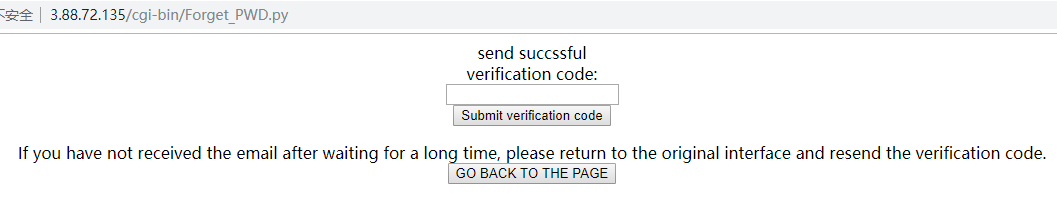
For the fourth and final area of system provision we are looking at communication with students. We have created an online chatroom that will be linked to the webpages of students so that they can communicate with teaching and administration staff and students. In our previous solution we had no way of communicating with students other then sending them emails. This process was a lengthy one as it would take a long time to send emails to the students. Our solution of an online chatroom quickens this process and can make the communication process a lot quicker for both sides of the conversation.



When something unexpected happens, such as students forget the password of their moodle website, usually students need to send an email to the teacher or contact the website administrator to change the password.There is no doubt that in either case, the students need to wait for a period of time to use the site again, so we created a web page to help students, students can access ‘http://ip/PWD.html’ page, enter your registered moodle when user name, email and the new password, students will be received in your email verification code from the web site, enter the correct verification code, the site will be strictly comparing students with information and data in the database, only when the data is consistent, system will change the password.On the other hand, since the initial password of our FTP is the same, when a student wants a special FTP password, he can still modify it through the above page. We created the chatroom in php and are implementing it into our website by creating a link on each users page that they can easily go to if they need to communicate with students or staff.

Here is a sample web-page:

When students enter their username, email and new password. An email containing the verification code will be sent to the student (Now, the email will be sent from my personal gmail, so sometimes it will be regarded as spam, but when it is actually used, it can be sent from an official email box). When the student enters the correct captcha, the password will be changed only after strict verification by the database . The same thing happens when you want to change the password for FTP:



Some code from Forget\_pwd.py:



Some code from change.py:





All the above parts about changing password are installed and run automatically in cgiscript.py,wdpwd.sh, pwd.html, forget\_pwd.py,change.py respectively.Students only need to enter the website address to enter the web page.

**Description of second focus**

**System Operation**

**Monitoring of System Operations**

Our second focus area was system operation which consisted of three areas, these being monitoring of system operation, cost monitoring and security monitoring. For the first area monitoring of system operations we have created a script that can be run from the press of a button on the admins webpage that will display every moodle instance and the state they are in, these being pending, running and stopped. In our previous solution the administration staff would have no way of knowing whos instance is running or not other than going into their aws account and manually checking the state of every instance. We also create alarms by using scripts, which can allow managers to be free to monitor all instances. We implemented this by creating a button on the admins webpage.

This then calls our python script that uses boto3 commands to display all of the current instances and the state they are in. We use the get\_cost\_usage command which returns a dictionary. We then have to loop through the dictionary and extract specific values from it.



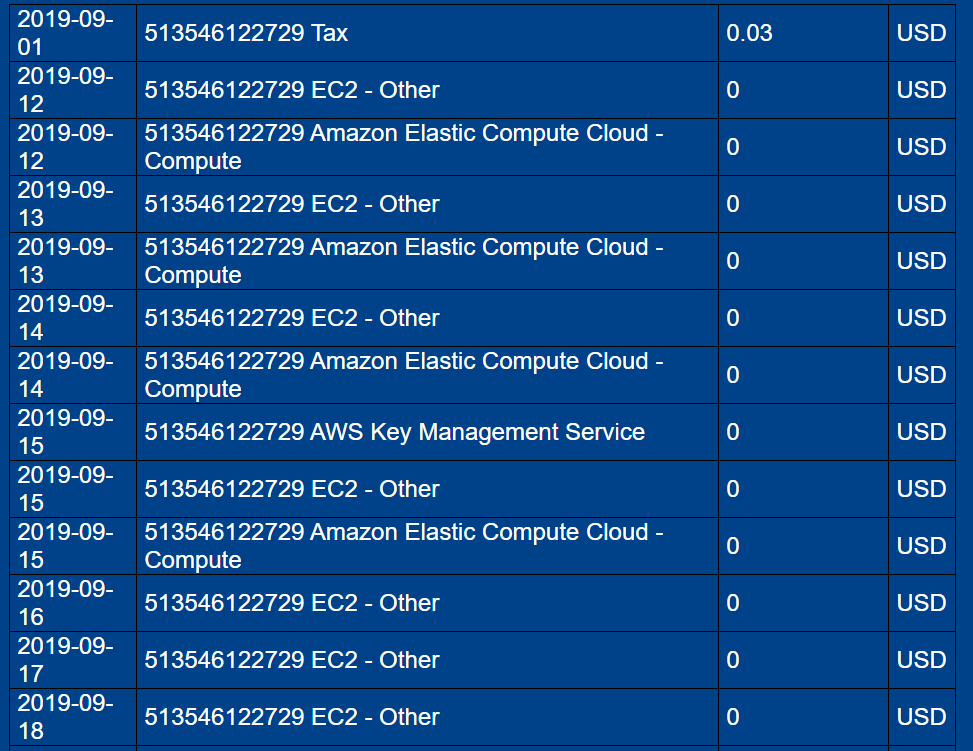
**Cost Monitoring**

For the second area cost monitoring we implemented a form on the admins webpage that calls a python script that displays a table of all the information related to the cost of the account. In our initial solution the admin would have no way of knowing all of their cost information without manually looking into the cost history of their account. Having this information from the press of one button means that the administration staff can see their previous costs on the account. From this they will be able to tell whether there has been a spike in activity which means a student may be using the ec2 instance for something other than work related things. We implemented this by creating a form form that is on the web page of the admin user.

Once the form is activated our python script in run which uses boto3 to get information from our account which is stored in a dictionary



We have implemented a table on the website that this data will be sent to and displayed in an easy to read way.



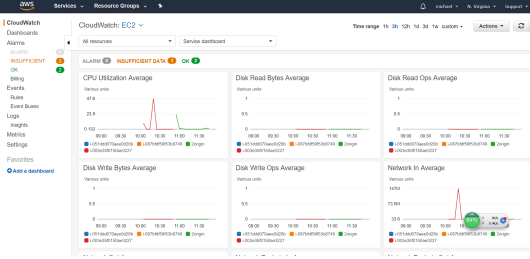
**Operation System Monitoring**

**Introduction:**

In many cases, the good functioning of the equipment, networks and systems will be the key to continue operating.To detect and prevent failures, it is very convenient to have a good monitoring tool, that is why you need a monitoring system to monitor operation system.

###### **1. Basic monitor of instances**

Aws Manager can easiliy see all resources of instances on the cloudwatch once instances are created**.**



In order to make it easier for manager to manage these instances, we create some alarms which can do some actions on the instances (e.g. stop, terminate, reboot) and send emails to remind the manager.

When we run the scripts, we already set **aws configure** in the beginning. And then we create

We should monitor many metrics such as: CPU steal time, CPU softirq time cpu, Free disk space, Free inodes, Read: Bytes/sec, Write: Bytes/sec, Total memory, Available memory and so on, But according to the aws free tier account’s alarm number is limited. For now, We just create 5 alarms to monitor the threshold of operation system:

1. name: cpu-alarm+%instance\_id. It alarms when CPUUtilization's average value is over 70% within 2 evaluation periods.

2. name: cpu-CreditUsage+%instance\_id. It alarms when CPUCreditUsage's average value is over 70% within 2 evaluation periods.

3. name: lb-mon+%instance\_id. It alarms when Load Balancer Latency's average value exceeds 100s within 3 evaluation periods.

4. name: ebs-mon+%instance\_id. It alarms when storage throughput's average value exceeds 100MB within 3 evaluation periods.

5. Name: cpu-monStop+$instance\_id. It will stop the instance when CpuUtilization’s average value is over 70% within 2 evaluation periods.

**The first 4 alarms can send emails to manager if the alarm be triggered. The last one will do action on instance.**

**And the Installation instructions is in the appendices.**

## **2. Cost Monitor**

### **Introduction:**

We use AWS billing and cost management to monitor the cost of aws account. AWS Billing and Cost Management is the service that you use to pay your AWS bill, monitor your usage, and budget your costs.

**(Introduction and installation instruction)We introduce four functions which you can directly use on it and operations about how to set it.**

###### **1. Dashboard**

Sign in to the AWS Management Console and open the Billing and Cost Management console at <https://console.aws.amazon.com/billing/>. The console opens to the Dashboard, where you can see your current month-to-date usage graphs.

On the dashboard you can view the following graphs:

1. Spend Summary

2. Month-to-Date Spend by Service

3. Month-to-Date Top Services by Spend

**2. Budget**

**We use aws budget to be able to view:**

1. How close your plan is to your budgeted amount or to the free tier limits

2. Your usage to date, including how much you have used of your Reserved Instances (RIs)

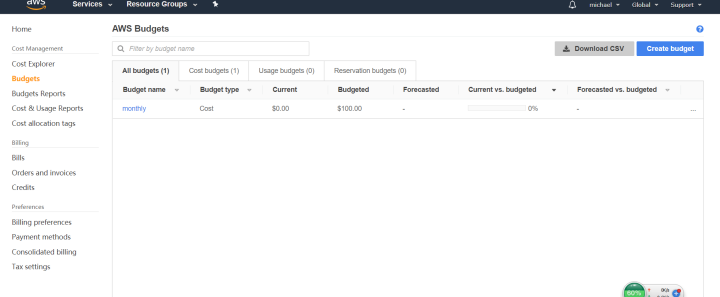
3. Your current estimated charges from AWS and how much your predicted usage will incur in charges by the end of the month

4. How much of your budget has been used

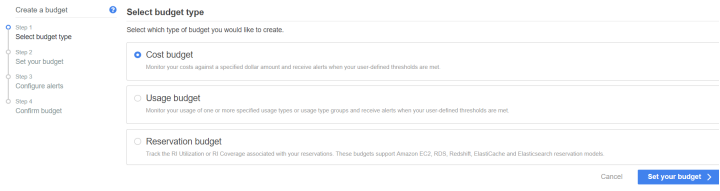
And we can create the 4 types of budgets: cost budget, usage budget, RI utilization budget and RI coverage budgets.

**To create budget ( OPERATION ):**

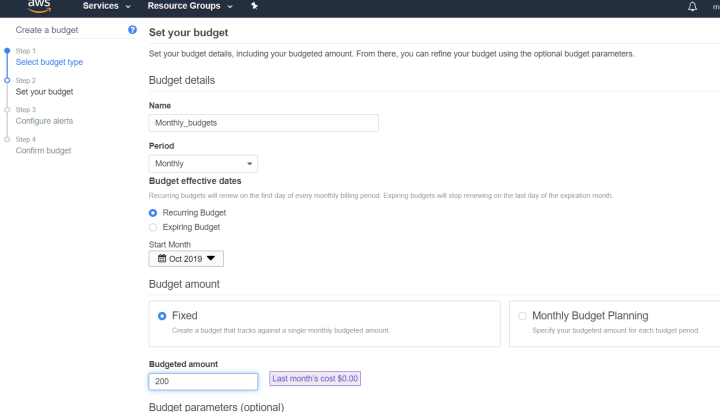
1. Click budgets on the navigation and click the blue “create budget” button.



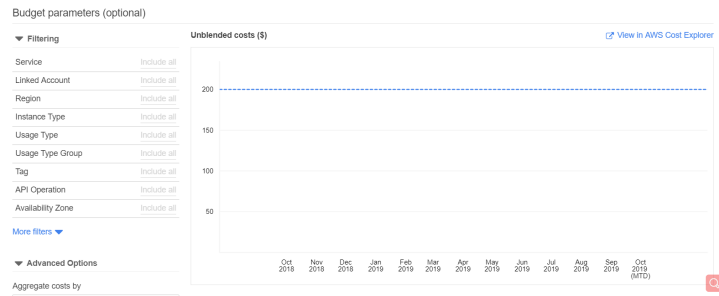
2. Then we can see 3 types of budget.



3. After clicking set your budget, you need give a name and enter the budgeted amount.Just let the Budget amount be Fixed.

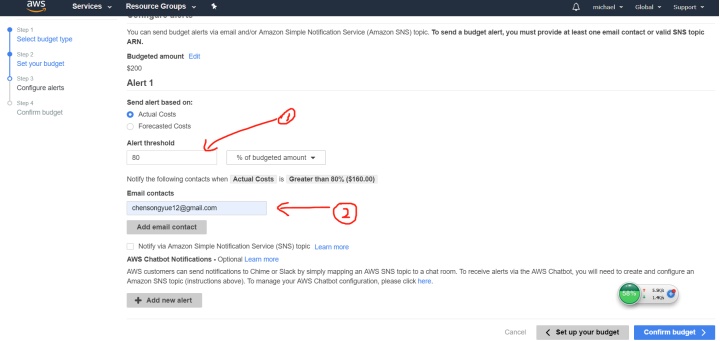


And you can set filter to restrict the range of monitoring.



4. And then click Continue.

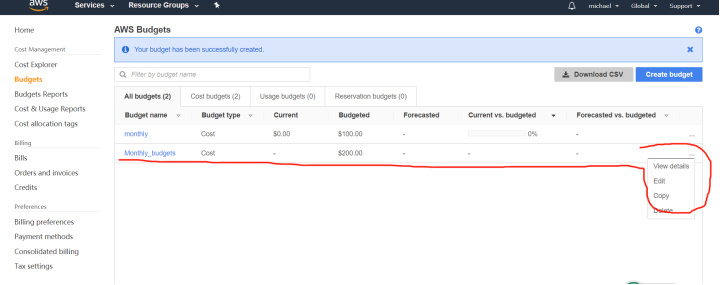
5. Then set the alert threshold and enter an email address to receive the alert when the cost exceeds the threshold.(We already create an SNS topic in the scripts, so no worries about the ARN)



6. Click continue and click create, this one alert has been created successfully.

If you want more alerts, please repeat it once again :).

(After created, You can see details in this page)

****

**4. Budgets Report**

**Introduction:**

AWS Budgets enables you to configure a report to monitor the performance of your existing budgets on a daily, weekly, or monthly cadence and deliver that report to up to 50 email addresses.

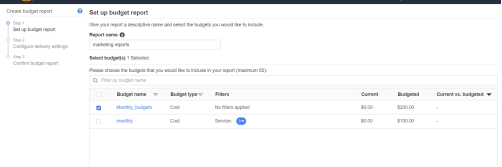
This enable you know the details of account without opening the website and before the alarm be triggered.

**Create budgets report - Operation:**

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at [https://console.aws.amazon.com/billing/home#/](https://console.aws.amazon.com/billing/home).

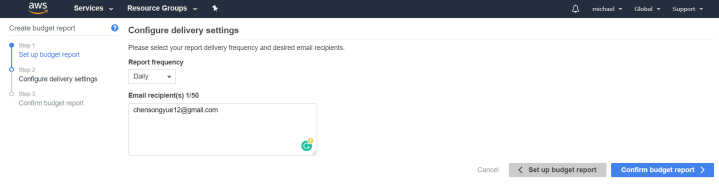
2. In the navigation pane, choose Budget Reports.and choose create budget Reports.

3. Enter the name of report

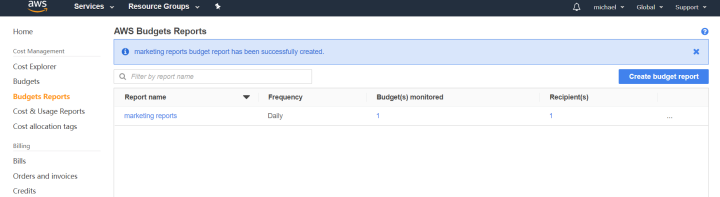


4. Enter the email address which can receive emails.(up to 50 emails)

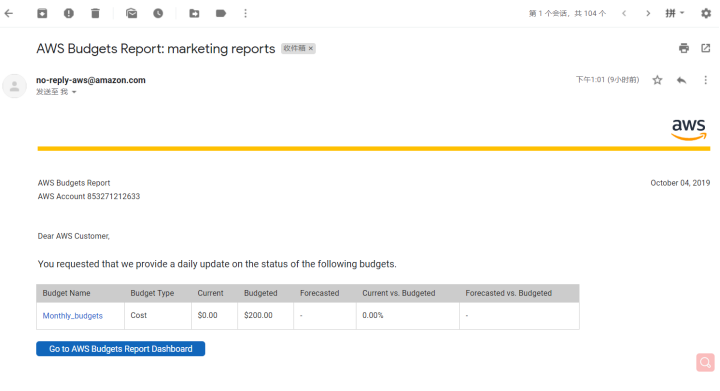
And you can change the frequence.(daily, weekly or monthly)



5. Click create, and will be successful.

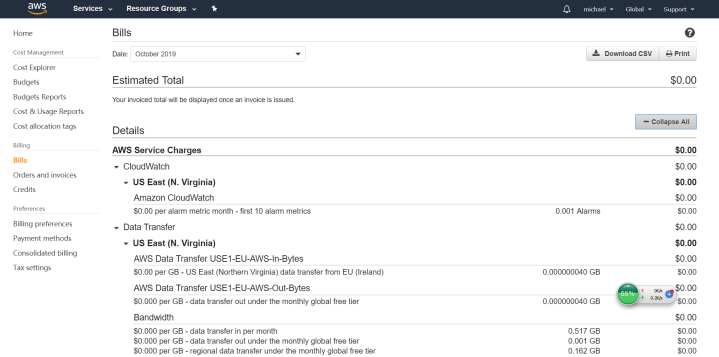
****

**And here is the email you will receive:**

****

**5. Bills**

Choose **Bills on dashboard** to see details about your current charges.

**~~~~**

**For the third and final area Security monitoring. We decide to use aws service: SecurityHub**

**Introduction:**

AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your compliance with the security industry standards and best practices. Security Hub collects security data from across AWS accounts, services, and supported third-party partner products and helps you analyze your security trends and identify the highest priority security issues.

Reason we choose it:

·It belongs to aws service.

· The GUI is very convenient and easy to use.

·Consolidated view of security issues

·Automated, continuous compliance checks

·Custom response and remediation actions

·Multi-account support

**Cost:**

In the first month, you have 30 free trial.

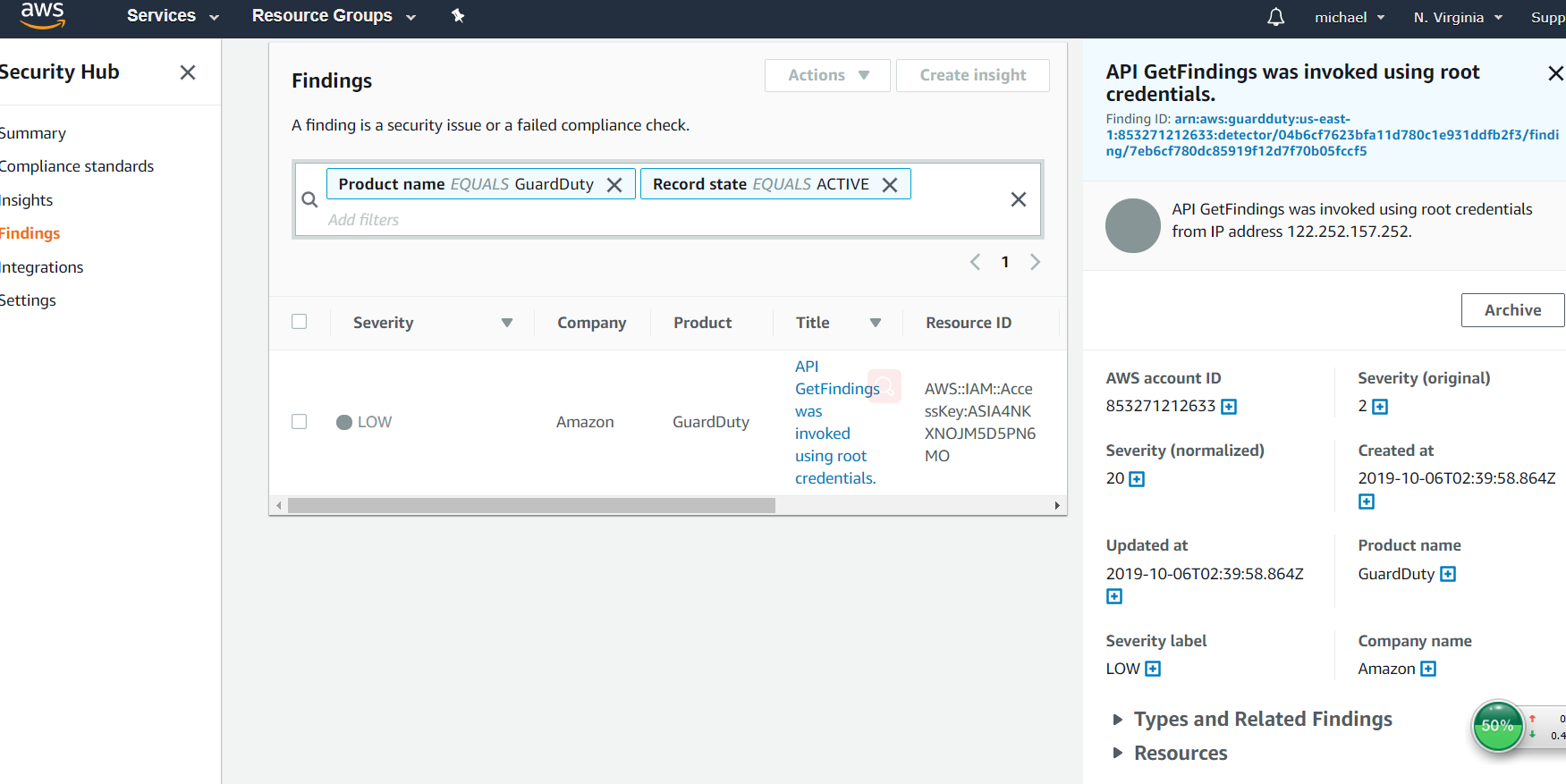
After that, the cost is:

|  |  |  |
| --- | --- | --- |
| cost | Compliance checks | Finding ingestion events |
| <=10,000 | $0.0010/check | Free |
| 10,001 - 100,000 | $0.0010/check | $0.00003/finding |
| 100,001-500,000 | $0.0008/check | $0.00003/finding |
| >=500,001 | $0.0005/check | $0.00003/finding |

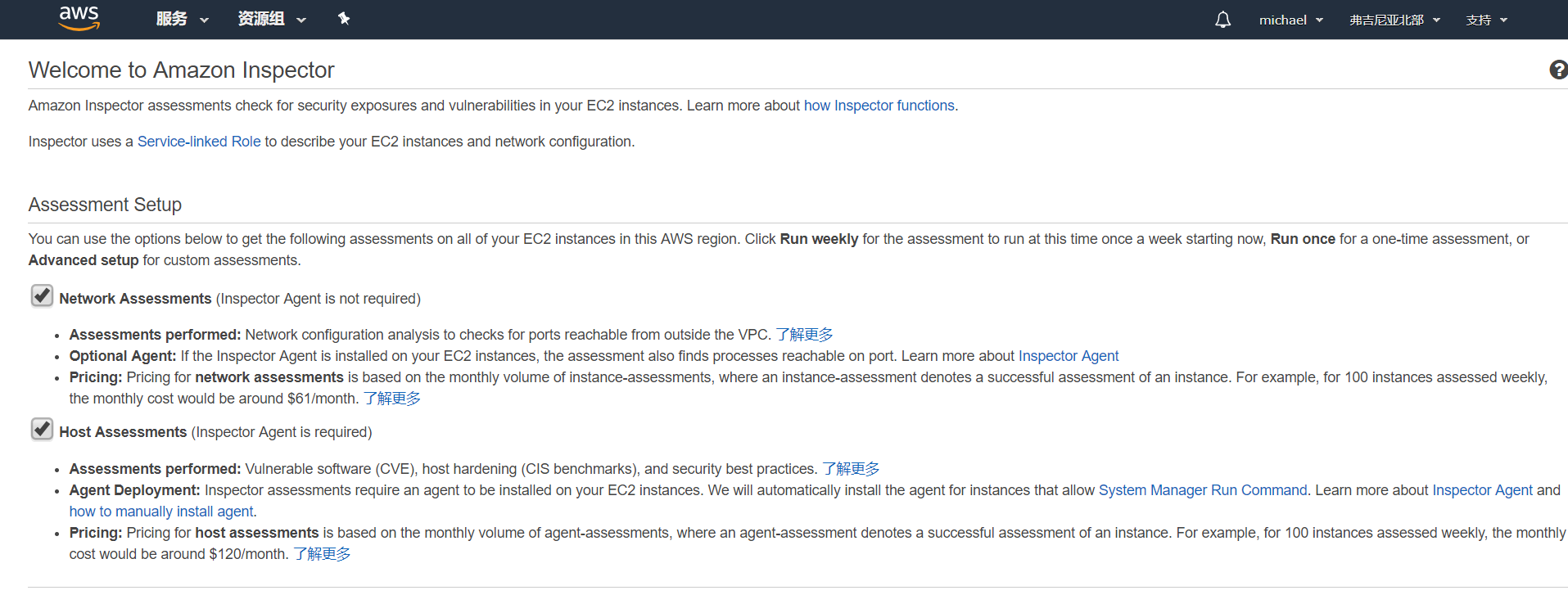
It’s very convenient to set up the SecurityHub on aws console. Manager can easily see details on this website.

There are three more modules in SecurityHub,

Amazon GuardDuty: Protect AWS accounts and workloads.



Amazon Inspector: Analyze the behavior of AWS resources and help users identify potential security issues.



Amazon Macie：

Use data visibility security service that helps classify and protect users sensitive and business-critical content.

Another security issue we addressed was the fact that any person with access to the url that offered the selection of instances could go on and select any instance with no limit to how many they could select. We looked into multiple different avenues to combat this for example logging the IP addresses of people when they select an instance and preventing them from selecting a second on that IP, only allowing people with an enrolled email address have access to selection and the same for only enrolled student ids or also pre assigning the students instances. Logging ip addresses wasn't suitable due to the fact that non enrolled students could still select instances or people could spoof ip addresses to select multiple instances, this would also mean that the teacher would have no name to an instance if its privileges were being neglected. Using the students emails would be effective but students may have multiple addresses and not know which one they applied to university with. Assigning the students instances could mean more overhead and work for the teacher which could be avoided by having a select yourself system. We ended up deciding on the student ID system due to the fact that every student has a student id, uses this on a daily basis and also you generally don't know someone else's student ID.

**Appendices**

**Installation Instructions for the base solution**

### **Required software**

1. python 3
2. AWScli (install guide below)
3. boto3 for python (install guide below)
4. paramiko for python (install guide below)

### **AWScli Setup**

1. Create a new IAM user(This can be done in the user tab in the My security credentials page)
2. Once the user is created press the security credentials tab to access keys
3. Press create access key
4. Save these 2 keys somewhere on your computer
5. In the command Prompt enter 'aws configure'
6. For access key and security key enter the 2 keys that you have saved
7. For region enter the region your account is in
8. For output format enter json
9. You will now be able to access any aws cli commands

### **Setup**

### Install boto3, and paramiko using the commands below in your command line.

### pip install boto3

### pip install paramiko

### Install awscli.

### for windows using [AWScli (64bit)](https://s3.amazonaws.com/aws-cli/AWSCLI64PY3.msi) / [AWScli (32bit)](https://s3.amazonaws.com/aws-cli/AWSCLI32PY3.msi)

### for linux/mac the pip command pip install awscli

### Change the gmail address and password settings in the \_\_\_\_\_\_ file and turn on Less secure app access [here](https://myaccount.google.com/u/3/lesssecureapps?utm_source=google-account&utm_medium=web)

### Run command aws configure in your command line and input your access key, security key ([shown on this page](https://console.aws.amazon.com/iam/home?#/users) by creating a user), and region.

### Run python script AutomaticEC2Creator.py and select the csv file containing the students you want to create instances for (this will complete the setup for every instance).

### Once the instance are set up and you are ready for the students to access their sites you need to allow all traffic on the default security group. This is done by right clicking on the default security group ([shown on the page](https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#SecurityGroups:sort=desc:tag:Name)) and selecting edit inbound rules. Once this is done select add rule and change the Type to "all traffic".

### When ready to give students the urls to their instances run the email.py which will send an email to every user with the url.

### 

**Installation Instructions for the two focus areas**

### **Required software**

1. Xampp
2. AWScli (install guide below)
3. boto3 for python (install guide below)

### **AWScli Setup**

1. Create a new IAM user(This can be done in the user tab in the My security credentials page)
2. Once the user is created press the security credentials tab to access keys
3. Press create access key
4. Save these 2 keys somewhere on your computer
5. In the command Prompt enter 'aws configure'
6. For access key and security key enter the 2 keys that you have saved
7. For region enter the region your account is in
8. For output format enter json
9. You will now be able to access any aws cli commands

### **Setup**

1. Install boto3 using the commands below in your command line.
   * pip install boto3
2. Install awscli.
   * for windows using [AWScli (64bit)](https://s3.amazonaws.com/aws-cli/AWSCLI64PY3.msi) / [AWScli (32bit)](https://s3.amazonaws.com/aws-cli/AWSCLI32PY3.msi)
   * for linux/mac the pip command pip install awscli
3. Install [xampp](https://www.apachefriends.org/index.html)
4. Navigate to the htdocs folder in the xampp file structure and paste all the website data.
5. Click mysql admin on the xampp control panel. create a database table called itproject.
6. Import the mysql database from the file in the given folder.
7. Create a new user account on the database, (hostname = localhost) then add all this information to the connect.php file.
8. If portforwarding this you can run the python script from [this repo](https://github.com/stanleychilton/ITproject) on any computer just make sure to change the address how ever if not portforwarding for the test you will need to change the url in the python scripts in [this repo](https://github.com/stanleychilton/ITproject) to localhost and run them on the same machine testing over a network can be tested using [this site](http://www.clicker-box.com/selectionsite/)
9. You should now be able to use the website

### **Chatroom Setup**

1. Click Admin on the xampp control panel to go to the database
2. Create a new database called chatroom
3. Copy the sql queries from chat\_database.sql and paste them into the sql section of the database
4. Copy the 2 queries seperately from chat\_database1.sql and paste them one by one into the sql section of the database
5. Chatroom should be all setup now

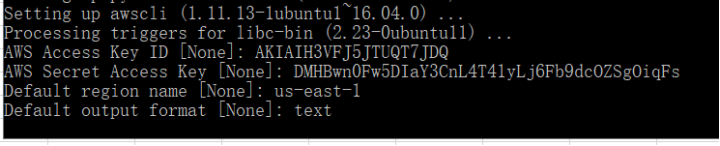
**Alarm Set up**

In the first instance, in the running scripts process, the command line will ask for 4 things:

① AWS access key id and AWS Secret Access Key can be get from AWS account.

② Default region name: should be the region where the server is located.(e.g. us-east-1 If your availability zone of instance is us-east-1b, eliminate the "b" at the end and put the default region name be us-east-1)

③ Default output format: text (If you didn't fill with text, the code may not work properly.)



So this instance can remember all this parameters. We can use script to create topic, subscribe alarm and create alarm.

But there’s a problem that if we put creating alarm scripts in the file which will run automatically when the vm starts, the file won’t work well. So I create one more script named AutoCreateAlarm.py in configSite repository. When student log into their own instance, they just need to run this file once and all alarms will be set done.

Sudo python3 /home/ubuntu/configSite/AutoCreateAlarm.py

**Password Changing website setup**

When students want to change the moodle password and they know their original password, they can do so directly from the moodle web page.When students forget their password, they can change the new password by using the pd.html page.Since the password stored in moodle database is the password after MD5 encryption, we cannot reverse decrypt it, so we can only help students change it into a new password.

All of this information can be found in the README files of the two github repositories that we are working out of