Machine Problem 1

1. The Load Balancer

Load Balancing is a feature in cloud computing that enables different computers to handle a task. In this way, if one has a set of servers and one of the servers crashes, the load balancer can redistribute the traffic to the other computers.

In this problem, you will be creating a load balancing system.

Create a Load Balancer from the Amazon toolbar on the left, name it TEAM(YOURTEAMNUMBER)LoadBalancer.

Example: Team01LoadBalancer

Configure the port on the load balancer to http:80. Leave your LoadBalancer Online.

2. Server

A server is a system (software and suitable computer hardware) that responds to requests across a computer network to provide, or help to provide, a network service. Servers can be run on a dedicated computer, which is also often referred to as "the server", but many services require a network of server computers.

The chat servers you need to create receive characters from many clients and re-broadcast each character to every other client. The architecture of the system is depicted in figure 1.

Figure 1.
The architecture of the Chat system

The server(s) need to create chat rooms, which act as a way to filter the chat traffic so that not all client's chat . The clients, servers and the load balancer should all use cookies to track which server instance handles which clients' data. The goal is to have each chat room handled by a different server on a different EC2 instance, and use the load balancer to send chat traffic for each chat room to the correct server. The load balancer handles sticky sessions using HTTP cookies as described in this link:


You can choose to write your server as either a Java program, or an Apache Tomcat servlet. It might sound harder to use the Tomcat, but in the long run it will save you more time. For instance, It can easily work with cookies:

http://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/Cookie.html

The service must also charge per character ($0.10) sent by any chat client. The amount charged can be displayed however you choose in the client, either in real time or otherwise. But the servers have to track the usage per client.

Example:

"This is a cloud computing class." 32 characters (including period) * $0.10 = $3.20

Upload the server program as an html webpage (index.html), on to your server, edit the webpage to include your team number. Set up the port to 8080. If everything is installed correctly, you should be able to see the webpage at a url "http://yourec2instanceurl:8080/". Now, create an AMI Image of this webserver, which should create an exact duplicate of your instance. Then, launch two more web servers using the image, and rename them to Team(YOURTEAMNUMBER)-A and TEAM(YOURTEAMNUMBER)-B. You will have to manually start your servers on each one. If done correctly, you should be able to go to the DNS name of your load balancer and it should automatically redirect to one of the internal servers. Also, trying to stop a server to see if your load balancer will still work in distributing traffic to the other servers.

Machine Problem One

In this MP you will be creating a chat software as a service (SaaS). If you run into problems creating EC2s, let me know if there are any problems. The due date for MP1 is scheduled for Tuesday, February 25th, 11:59pm.
3. Client

A client is a piece of computer hardware or software that accesses a service made available by a server. The server is often (but not always) on another computer system, in which case the client accesses the service by way of a network.

For this MP, you will create a client program in your other instances. This client program must be able to send, receive and broadcast messages. This does not need to be a GUI, it can all be run from the command line. Alternatively, it can also be a Javascript program running in a browser. Choose a method that makes it easiest for you to program the service.

Lastly have an exit message that signs out your clients. This message can be anything and should also display the charge for that specific client. For the demo, you need to show 5 clients talking on 2 different chat rooms.

Also, put the source code for your servlet and your team's private key to the instances into a zipped folder and email it to me. Only 1 person from each team needs to submit. Do this by the due date.

Note: for the regular grading, we do not look at how beautiful or ugly your programs look like!

4. Extra Credit

There are ways to receive extra credit. Note: a group can not receive more than a maximum of 10 extra credit points (total of 60 points for the MPs) for the whole semester.

Create a graphical user interface for the servers or clients. (extra 12 points)

A running character counter displayed to the user in the client. (extra 8 points)

5. Demo

You must sign up for a demo slot at the following link in order to receive a grade. Below is a rubric of requirements for this demo. If you cannot make any of the demo times, email me and we will work something out. The Demo location is in Siebel.

Demo Sign-Up: https://docs.google.com/spreadsheet/ccc?key=0AgzlVSWE8pIvdFEwdFEwdXJmZl9420hrNzBraWdsOEwyWVE&usp=drive_web#gid=0

Rubric for Demo: Rubric