ChatScript - Setting up an Amazon EC2 Server

© Bruce Wilcox, gowilcox@gmail.com

Revision 11/18/2013 cs3.74

If you want to make your chatbot visible on the web, you need a server. As it turns out, for low traffic, Amazon will let you have a server machine for free for a year and a low fee thereafter. Here is an overview of how to create a ChatScript server on Amazon's cloud services.

First, you need an Amazon AWS account. Go to http://aws.amazon.com/account/ and sign up for one if you don't have one.

Second, you need to acquire a server machine. You get one by going to the AWS console: https://console.aws.amazon.com/console/home and clicking on the EC2 (virtual servers in the cloud) link.

You need to alter security for a machine you will create. By default only you can talk to it. Locate the security tab on the left. Click the "Create Security Group" button and name your group a name and description. When it shows up in the list, click the name. Click on the Inbound tab. This will allow you to change incoming port access to your machine. You want to enable SSH and HTTP. Select "HTTP" from the dropdown and then click "Add Rule". You'll see the rule added to the right side of the panel. It will use 0.0.0.0/0 as the IP addresses, which means let anyone come into the HTTP port. Select "SSH" from the dropdown and click "Add Rule". Now you can log into the machine via a secure socket. Click the "Apply Rule Changes" button to apply both of these rules.

You'll need a public-private key to talk to your machine. Select New Keypairs. This will prompt you for a name and then immediately start a file download of a file with .pem. You need this file to access your future server.

There is a button there that says "Launch Instance". Click on that. I use the quick launch wizard choice. Enter a machine label (like My ChatScript Server) then select an Amazon Linux AMI machine (64-bit free tier eligible). Once the instance is launched, it will return you to the console where it will show it starting up. Eventually it will be running.

Third, you need to add stuff to the machine. There are a couple of ways you might be using the ChatScript server. First, by itself, with a port open to the web and talked to from your client program or webpage. Second, from a webpage on the same machine, opening no port to users for the ChatScript server. I chose the latter, so I had to have a webserver installed and a webpage to talk to ChatScript. A demo webpage named "index.php" exists in the LINUX directory of chatscript and uses PHP to do the work. (*Thanks to Wade for the initial copy of this webpage.*)

Which means you need a way to communicate with it. I used WinSCP to talk to it along with Putty, and PuttyGen to generate a key from my keypair that WinSCP can use to authenticate itself. Let's assume (non-trivial) that you can get a Putty terminal talking to your instance. You are going to need a few pieces of software on this machine.

1. become a superuser with "sudo -s"
2. get yum to update itself with "yum -y update"
3. Apache (the web server stuff) comes preinstalled but you can try "yum install Apache" and it should tell you there is nothing to do.

4. You need C libraries to run ChatScript, so you want to install stdlibc. I wasn't sure of the options so I first did “yum search stdlibc”. It gave me almost 20 items of that.I wanted the GNU C library, so I did “yum install libstdc++46.i686”
5. I also wanted to be able to recompile the system, so I installed the g++ compiler by doing “yum install gcc-c++.noarch”
6. I needed php to run my webpage, so I did “yum install php”
7. If you want to be able to debug using gdb the engine itself, you want to do
   yum groupinstall "Development Tools"

Using WinSCP I created a top level folder named ChatScript and transferred a standard ChatScript release up to the server, having first built my bot (so the TOPICS folder was ready). I then had to make ChatScript executable, by the following from Putty:
“cd ChatScript”
“chmod +x LinuxChatScript32” or LinuxChatScript64 (depending on your machine)

I also needed the webserver running, so I did “service httpd start”. Since that starts the webserver, you can prove that works by typing in the web name of your server (found on the console) into a browser and getting back an Apache test page. The server will stop if your machine ever stops, so you might want it to automatically start up whenever the machine does. “/sbin/chkconfig httpd on” will set autorestart on apache.

Then I edited the ChatScript test webpage. You need to set the IP address in the file LINUX/index.html to be the IP address of your instance. Once you have edited that (I did it using a local editor on my laptop and then uploaded the change along with ChatScript so I didn't have to use a linux editor). With the file changed, I copied it to the web server page folder via:
“cp LINUX/index.php /var/www/html”

Then I ran ChatScript via
“./LinuxChatScript32”
and tried to browse see my site again. And talked to my bot!

Of course as soon as I leave the WinSCP connection, ChatScript comes down. And if the machine ever restarts or ChatScript ever crashes, your server is dead. So you need a cron job to automatically restart ChatScript.
“crontab LINUX/jobs.cron”
will set an auto restart attempt every 5 minutes. Now I can exit my connection to the instance and ChatScript will start up on its own and keep going.