ChatScript - Setting up an Amazon EC2 Server

Copyright Bruce Wilcox, gowilcox@gmail.com Revision 8/23/2020 cs10.6

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 HTTP if this will be a web server. 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.

Click the `Apply Rule Changes` button to apply both of these rules. If this machine is just to be a chatscript server instead with the webpage on some other machine, you need to enable the port for ChatScript. Select Inbound, custom TCP, select port 1024 (or whichever you want), select ANYWHERE instead of custom. Then `Save`.

You'll need a public-private key to talk to your machine. You may have had this generated when you created the server, else find Keypairs and 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). The default login id for an AMI machine is ec2-user.

If you choose an Ubuntu linux, the default user is ubuntu. Once the instance is launched, it will return you to the console where it will show it starting up. Eventually it will be running. Nowaways the interface to selecting your machine is different but the choices are the same.

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 `WEBINTERFACE/SIMPLE` 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.

To compile CS src or even just to execute it, you will need to have Curl installed, which for some machines is:

```
sudo yum -y install libcurl libcurl-devel
```

For AMI linux do:

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. I needed php to run my webpage, so I did yum install php‘
5. If you want to be able to debug using gdb the engine itself, and compiling with make and g++ you want to do yum groupinstall “Development Tools”. You may have to do this anyway or use a different package to get the std c library

In fact, for some machines like CentOS, to get json stuff and evserver stuff, and you want to compile the source, you may need

```
sudo yum -y install -y libc -devel gcc libc6 -compat libcurl libcurl-devel libstdc++ libgcc
```

For Ubuntu linux:

1. `sudo apt-get install make` - to get make installed
2. `sudo apt-get update` - to get current versions of things
3. `sudo apt-get install g++` - to get the compiler
4. if you want postgres: `sudo apt-get install postgresql postgresql-contrib`
5. if you want your web page to be here; `sudo apt-get install apache2`and `sudo apt-get install php5`
6. `sudo apt-get install gdb` — for debugger for C++ if you can use it (unlikely)

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

```
cd ChatScript
chmod +x 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.

You need to set the IP address in the file `WEBINTERFACE/SIMPLE/index.php` to be the IP address of your instance. If your webpage is on the same machine as your chatscript server, you should use `localhost` as the ip address, meaning the current machine. My webpage is on a different machine than my chatscript server. In the Amazon EC2 instances display is a line corresponding to my EC2 machine, which names both my public DNS and my public IP. Either can be used in the script, but I use the IP since it's simpler. I edited the ChatScript test webpage to:

```
$host = "1.22.08.4"; // <<<<<<<<<<<<<<<< YOUR CHATSCRIPT SERVER IP ADDRESS GOES HERE
```

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:

```
sudo cp WEBINTERFACE/SIMPLE/*.* /var/www/html
```

Then I ran ChatScript via

```
./LinuxChatScript32
```

and tried to browse see my site again. And talked to my bot!

If it doesn't talk, you can test to see if its the server or not that is having the problem by using another terminal window and while in the ChatScript directory typing

```
./LinuxChatScript32 client=localhost:1024
```

and try talking thru that. If that works, the server is fine and your index.php script isn't right for some reason.

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.

BETTER WEB PAGE

The SIMPLE webinterface just shows two lines of text. The BETTER webinterface has a scrollable text box to see the history and supports the loopback/callback/alarm OOB messages. You install that just like you did the SIMPLE interface, by

```
sudo cp WEBINTERFACE/BETTER/*.* /var/www/html
```

> Thanks to Dave Morton for the improved interface. You need to edit ui.php to set your IP address and bot name.

And beyond that is a SPEECH WEB INTERFACE