Set up aws server for web site

<http://www.rndmr.com/amazon-aws-ec2-easy-web-serverset-up-guide-with-a-mac-and-coda-2-256/>

# Amazon AWS EC2 Easy Web Server Set Up Guide

*Please note: This is going to be another of those techie posts – the previous ones have been well received so since we have spent some time with this particular combo of hardware and software lately we thought we would share the info here.*

Lately we have been trialling [Amazon Web Services](http://www.amazonaws.com/" \t "_blank), and particularly their cloud server product called Amazon EC2. Now is a pretty good time to get your feet wet since they are offering a completely free tier of usage, allowing you to play around with cloud computing without any cost.

This guide will detail how to set up a web server with a LAMP (Linux, Apache, MySQL, PHP) stack on a free Amazon EC2 account, and as a bonus will also let you know how to connect to it with [Coda 2](http://www.panic.com/coda" \t "_blank) (our current favourite Mac coding environment) to start getting your web site online.

So without further ado here is the guide.

**Starting up a server**

* [Go and sign up for Amazon Web Services](http://www.amazonaws.com/" \t "_blank). You will need to enter a credit card, but it won’t be charged as we will only be using a server that is on the free usage tier.
* Sign in to the AWS Management Console (It’s a link under the dropdown in the top right) with your user name and password.
* Click on EC2: Virtual Servers in the Cloud – the first option under Compute & Networking. Amazon offers a baffling amount of services, but for our purposes (and the purposes of most people) EC2 will be the only one you need.
* Click the button marked “Launch Instance”. Instances are basically servers, so what you are doing when you are launching an instance, is effectively starting up a server.
* Click “Classic Wizard” and continue, as its a bit more explanatory than the quick launch one, though that might be a better option when you get to grips with it.
* Select a 32 bit Amazon Linux AMI type server – it has the star next to it, so that means it’s eligible for the free tier if you select a Micro (lower powered) instance.
* Ensure number of instances is set to 1 and type is set to Micro (so it’s free) . The availability zone doesn’t really matter here so select no preference on that dropdown and hit continue.
* On the next page, all of the defaults are fine, so we just hit continue again.
* Let’s give this server a name for when we are administering it later – let’s go for Webserver. Enter Webserver in the value box next to name and hit continue.
* Now on to some security. In order to securely access the server from your local machine, you will need to create a key pair, so under Create a New Key Pair enter a name for it. It doesn’t matter what it’s called really, but for this tutorial let’s call it “myawskey” and click “create and download your key pair”. This will download a file called myawskey.pem to your computer. Put it somewhere you will be able to find it later on.
* You now need to add some security to your new server. These come in the form of security groups, so let’s select an existing group (AWS offers you some defaults) to make it easy. Select “quick-start-1″ and click Continue.
* Next you will get a screen showing everything you’ve just selected – as long as nothing looks particularly odd you can click “Launch” and your new server will be up and running within a minute or so!

**Adding all the software you need to your new server to make it into a Webserver**

So we now have a brand new server, but it needs some software installing so that it can serve web pages for us, so that’s the next step. Here we will be connecting to the server with something called SSH and installing the LAMP stack Apache, MySQL and PHP (we already have the L of the LAMP stack, in that the servers operating system is Linux). These instructions are Mac-centric, although the same commands could be used with a different terminal application on a PC.

* First of all, let’s get familiar with a few things in the AWS console. Your server (an instance) is listed under the instances menu item on the left hand side. Once you click this you will see it on the right. Click the tick next to it and you will see some information about it appear at the bottom. Scroll down to “public DNS” and copy it. This is the address you will be using throughout to connect to the server.
* Let’s take the key pair you created earlier and put it in the default directory for key pairs so that our Terminal app will apply it automatically when we try to connect to our server. Copy the myawskey.pem file to “//.ssh/”. For example, mine would be /anthony/.ssh/
* Start up the Terminal application on your Mac and type “ssh ec2-user@“. For example it may look something like this: ssh ec2-user@ec2-00-00-00-00.compute-1.amazonaws.com
* Now we are in we’ll need to run some linux commands to install the software, but first let’s give ourselves the appropriate permissions to do all the installs by typing “sudo su”.
* First off lets install any updates to the packages that are already on the server by typing “yum update”. Say yes to all the updates until it has finished running.
* Now let’s install apache with “yum install httpd”.
* Now let’s get PHP installed: “yum install php php-mysql”.
* And finally let’s install mySQL: “yum install mysql-server”.
* Now let’s start those services, beginning with the webserver: “service httpd start”.
* And ending with MySQL: “service mysqld start”.
* Now we need to do some work to add a user that can manage the database from your favourite MySQL program on your local machine, so let’s run the following to start the program: “mysql”
* Decide on a username and password for your admin account and enter this command (in the example we are using username “admin” and password “password” – please pick something a bit more inventive!):  
  GRANT ALL PRIVILEGES ON \*.\* TO admin@’%’ IDENTIFIED BY ‘password’ WITH GRANT OPTION;
* Hit enter to go onto a new line and type “\g” to execute the command, then \q to quit.
* While we are on the server, let’s do a couple more setup tweaks that you may need when you start to add your webpages to the server, enabling URL rewriting and PHP short tags.
* To enable URL rewriting type “vi /etc/httpd/conf/httpd.conf” to open the apache config file in the vi editor.
* Scroll down to where it says “AllowOveride None” inside the DocumentRoot Directory Directive, normally
* Press i to enter insert mode and change “AllowOveride None” to “AllowOveride All”. Press escape, then type :wq! to save and quit the editor.
* To enable PHP short codes, type vi /etc/php.ini to start the vi editor with the appropriate file.
* Scroll down to where it says short\_open\_tag = Off and change it to short\_open\_tag = On by pressing i to go into insert mode.
* Press escape, then type :wq! to quit the editor.
* Lastly, we will need to restart the webserver so those last two changes that effect. Enter “service httpd restart”.
* That’s it! We are all done on setting up the LAMP stack.

**Using Amazon AWS Management Console to allow web and MySQL access to the server.**

So we have everything set up, but our security settings mean we can’t actually access the server through our browser (to view the webpages) or through our SQL program to add any databases, so let’s rectify that.

* Go back to your browser window that has the AWS Management Console open and click “Security groups” under “Network and Security” on the right hand menu.
* Click the tick next to “quick-start-1″ to select the security group we applied to our webserver.
* Select the “Inbound” tab in the details pane at the bottom left. You will see that SSH access on port 22 is set up, which is what we’ve just used to access the server and install all of the software.
* Now lets add HTTP access so you can view web pages in a browser. Select “HTTP” under “Create a new rule”, Leave the source as it and click “Add Rule”.
* Save the rule by clicking “Apply Rule Changes”.
* Now lets add remote MySQL access. Select “MYSQL” under “Create a new rule”, Leave the source as it and click “Add Rule”.
* Save that rule by clicking “Apply Rule Changes”.
* All done! You should now be able to visit http:// and see an apache test page from your server. You should also be able to use your favourite SQL program to access with the admin username and password we set up on the server.

**Last part: Uploading files to your webserver**

When it comes to uploading files to your webserver, most would use FTP, however for added security and convenience (since it doesn’t exactly play well with FTP) Amazon AWS uses SFTP instead. You can check that SFTP is working with the command line, but we had some issues setting it up in Coda 2, so we’ll just cover that briefly here for anyone with the same problem.

* First let’s check SFTP is actually working. Go to your terminal and type “sftp ec2-user@“. For example this may look like: sftp ec2-user@ec2-00-00-00-00.compute-1.amazonaws.com.
* If it connects, then SFTP is working fine, you would set this up in Coda by going to site settings and doing the following steps.
* On “Site”, set protocol to “SFTP”
* Set Server to ““.
* Set User Name to “ec2-user”.
* No need to enter a password or select a key file in the next box, as we put the key file in the default directory earlier on and Coda is smart enough to look there.
* Coda should now be set up to upload files via SFTP.

Set up domain

<http://www.rndmr.com/setting-up-a-domain-on-amazon-web-services-ec2-259/>

# Setting up a domain on Amazon Web Services EC2

*This post follows on from our previous post [Amazon AWS EC2 Easy Web Server Set Up Guide](http://www.rndmr.com/amazon-aws-ec2-easy-web-serverset-up-guide-with-a-mac-and-coda-2-256/" \t "_blank), showing the natural next step of pointing a domain to your new cloud server.*

Now we have a new Amazon Web Services EC2 server fully configured as a webserver, let’s point a domain to it, as this has it’s own idiosyncracies using the AWS system.

**Setting up AWS to provide an IP address for your DNS settings**

* First we will need to assign an IP address to our server using Amazon’s Elastic IP system, so click Elastic IP’s in the left hand menu.
* In the top menu under “Addresses”, click “Allocate New Address”.
* Ensure “EIP used in” is set to “EC2″ and click “Yes, Allocate”.
* Click the tick box next to the new IP address that has appeared in the main left pane and press the “Associate Address” button in the top menu.
* In the “Instance” drop down, select your webserver that we set up previously and click “Yes, Associate”.
* That’s it for management on the AWS side of things, however please note that now when accessing your webserver through SSH, SFTP or MySQL, your public DNS address that you were using for the connection will have changed. At this point it is probably better to just use the new IP address to connect that you have just associated with your webserver.

**Setting up your DNS**

* We won’t go into too much detail here as there are far too many domain and DNS providers (GoDaddy, 123-Reg to name a couple) to cover off every eventuality, just ensure that the IP address you have created in AWS Management is entered for the @ and www records for your domain.

**Setting up some VirtualHosts**

* You may want to set up more than one domain on the server, so let’s set up a VirtualHost to show the way to do this on an EC2 instance.
* [SSH to your EC2 instance as described in our previous article.](http://www.rndmr.com/amazon-aws-ec2-easy-web-serverset-up-guide-with-a-mac-and-coda-2-256/" \t "_blank)
* Your default webserver configuration after going through the previous steps should load additional configuration files from the “conf.d” directory, so let’s switch to it with the command “cd /etc/httpd/conf.d”.
* Type “sudo su” and hit enter to give us all the privileges we might require.
* Type “vi httpd-vhosts.conf” and hit enter. This will create a new file where we will store the virtual hosts information.
* Press “i” to enter insert mode and type the following, replacing your-server.com with the domain name you want to use:

NameVirtualHost \*:80

<VirtualHost \*:80>

ServerName www.EVAnalysisCorp.com

ServerAlias EVAnalysisCorp.com \*.EVAnalysisCorp.com

DocumentRoot /var/www/html/EVAnalysisCorp.com

</VirtualHost>

* Press escape, then type “:wq!” to exit the editor.
* This tells the apache server to look in the /var/www/html/your-server.com directory for the files for any request to the your-server.com domain.
* Let’s check that the rules you’ve typed in are working correctly by typing “httpd -S”, this should tell you that the syntax is ok.
* Finally, we just need to restart Apache to get it to take notice of the new configuration file. Type “apachectl restart”.
* And you’re done! Type your domain name into your browser and you should see the files from the appropriate directory!

In

/etc/httpd/conf.d/httpd-vhosts.conf

have the below content:

NameVirtualHost \*:80

<VirtualHost \*:80>

ServerName www.EVAnalysisCorp.com

ServerAlias EVAnalysisCorp.com \*.EVAnalysisCorp.com

DocumentRoot /var/www/EVAnalysisCorp.com/website/

</VirtualHost>

restart apache server

apachectl restart

sudo service httpd start

sudo service httpd restart

sudo apachectl start

sudo apachectl restart

git clone [http://github.com/EVAnalysisCorp/website.git under /var/www/EVAnalysisCorp.com](http://github.com/EVAnalysisCorp/website.git%20under%20/var/www/EVAnalysisCorp.com) directory, will put web page files in

/var/www/EVAnalysisCorp.com/website/

make any changes in the git repo EVAnalysisCorp/website and pull on server will reflect new changes.

KDD awards content is located under website/KDDAwards directory

Will figure out how to do server side authentication later

Git size change to enable large document files for the website

git config http.postBuffer 724288000

install R

install R packages

cd /var/www/EVAnalysisCorp.com/website

sudo git pull origin master

sudo apachectl restart