How To Set Up A Minecraft Server On AWS Cloud - The Complete Guide

Minecraft is crazy fun, I don't care how old you are - or how young you are. I have played it with my kids starting both of them by age 6 and it's amazed me how fast they've picked it up, and ended up teaching me how to do things.

Minecraft is really great for their creativity, as well as their problem solving skills, and it's super fun as well.

Reasons you would set up a server - maybe you don't actually live with your kid full time (this was my situation), or maybe your kid would like his or her friends to join them. Or maybe you're the one who wants friends to be able to join. Either way, its not too hard.

Step One - Think About The Monthly Cost.

Update: I know this article is all about AWS but you can apply most of these instructions to a Digital Ocean server as well. If you use this link, you'll get a $10 credit for Digital Ocean. It's a great way to try out the process on a free server.

Also, right here I have instructions tailored toward Digital Ocean.

Size

One of the things to consider is which size AWS instance you are going to use. For a small group of people, you can choose nano or micro. A nano is going to be about the minimum that will even run Minecraft. You may run into glitches at this size, but its only about $5 a month.

A micro will run you about $10 a month, and it will be much better from a resource perspective, and probably much less glitchy.

Full Time or Part Time?

With AWS, you can power down your virtual server and you won't be charged. This is good from a dollars perspective, but I like to leave mine running all the time. For $5-10 a month, its nice to have your users be able to log in, build stuff, and then when you log in, you can go and see what they did while you were gone.

Step Two - Get started.

Create an EC2 Instance

1. Log into your aws control panel: https://us-west-2.console.aws.amazon.com/ 2. Once in, click on EC2. 3. Click "Launch Instance" 4. Select "Amazon Linux AMI" 5. AGAIN Based on the decision above, select "t2.nano" or "t2.micro" 6. Click "Next: Configure Instance Details" 1. On this screen, you will want to enable "Auto assign Public IP", so people can access the server - the rest you can leave as-is.

1. Leave it on "Create a new security group" 2. Click "Add Rule" 3. You want "Custom TCP Rule" 4. In the "Port Range" put 25565 5. In the "source" put "anywhere"

1. Find it via the terminal and you need to changemod it, if you're on Linux: 1. chmod 400 minecraft.pem (or whatever you named it)/help

Elastic IP?

Now, it will take a little bit for the server to spin up. Once it is, you can go back to your EC2 dashboard, and get the ip address. At this point, you may want to set up an elastic IP for it, to make sure your IP address never changes. If you want an IP that doesn't change, you'll have to follow these instructions:

1. On the EC2 Dashboard, look along the left hand side, and find "Elastic IPs". 2. Click "Allocate New Address" 3. Once it's been created, you need to right click it and select "Associate Address" 4. Associate it with the EC2 instance you just created. 5. Then use that new address from here on out.

If you don't do the elastic IP, you can get the ip address from the EC2 dashboard.

Connect

To connect, you type something like this:

I replaced my IP with 7s - use yours here. Also, verify the rest is correct - that you are indeed on the us-west region, etc. If you need, you can right click the instance and select "connect"

to see this info customized for you.

Also, this command assumes you're in the same directory with that pem file.

Once you're all connected, do an update:

sudo yum update

Then, make a dir for your minecraft server:

Download Minecraft Server

Find the latest one here, and copy the download link:

https://minecraft.net/en/download/server

Then in the terminal:

wget
https://s3.amazonaws.com/Minecraft.Download/versions/1.10.2/minecraft_server.1.10.2.jar

(Thats the link you copied from the website)

Then, make a sym link to it so that it's easy to start:

ln -s minecraft_server.1.10.2.jar minecraft_server.jar

If you used a t2.nano, you'll want to adjust the memory you're allowing it to use.

If you used a t2.micro:

Notice how the memory size is indicated:

sudo java -Xmx512M -jar minecraft_server.jar nogui

You could also do this on larger servers like this for 2 gig of ram:

sudo java -Xmx2G -jar minecraft_server.jar nogui

Agree to the EULA

You need to do this to allow it to run:

sudo nano eula.txt

Change "false" to "true" and save it.

To keep the server running

Since the server "start screen" is also the console, I like to use "screen" which allows you to come back to a terminal session after you've logged out.

Then, start your server with the right command from above:

sudo java -Xmx1024M -jar minecraft_server.jar nogui

Now, if you leave the terminal session, you can re-enter it later by typing

screen -r

More info on "screen" can be found here.

You'll want to connect to it with your Minecraft game, then you can setup whitelisting from the console, or add yourself and others as ops. All of this can be done via the console - and much more.