312 Midterm Minecraft Server Setup Guide

Table of Contents (Might be broken on PDF)

Video Demo

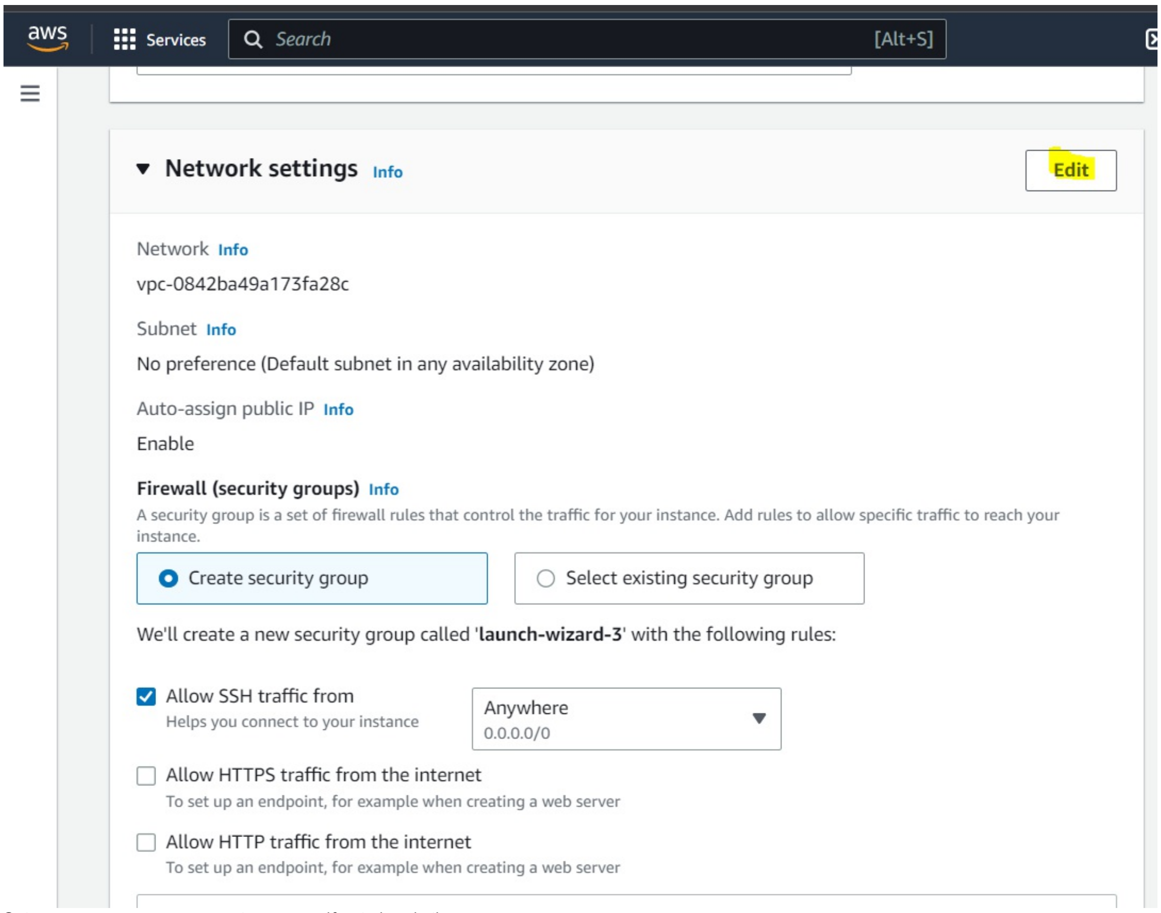
- https://youtu.be/MezIKRTq1zM
    - Read Youtube Video Description for Timestamps

References

- https://www.linkedin.com/pulse/setup-minecraft-server-java-edition-aws-ec2-keran-mckenzie/
    - Heavily inspired by this article although I did have to change the Java install steps due to some dependency errors, and this article also did not cover how to automatically start the server with systemctl
- https://techviewleo.com/install-java-openjdk-on-amazon-linux-system/
    - Helped me a lot on installing Java 17 for the server dependencies
- ChatGPT
    - Helped a lot on the systemctl part for automatically restarting server
- Markdown Table of Contents Generator: https://jsfiddle.net/remarkablemark/o0mja3hf/

EC2 Setup

- From the AWS homepage / dashboard, type in `EC2` into the upper left search bar, and click on it to go to the EC2 menu
- From the EC2 menu, click on the `Launch Instance` button
- In the Launch Instance menu, name your EC2 instance something like "Minecraft" so you can remember it
- For the `Application and OS Images (Amazon Machine Image)` menu, select an Amazon Linux, 64bit (x86) OS image
    - This should be the **default** option
- For the `Instance type` menu, select `t2.small` (should pick a t2 small or bigger for this)
- For `Key pair (login)` menu, I recommend just creating a new SSH key (click the `Create new key pair` button). I used the default options (`RSA`, `.pem`)
    - Name the keypair something memorable, like `MinecraftKey`. In my case though I'm reusing an old key called `lab7`
- In the `Network Settings` menu, click the `Edit` button to the right to edit the Network Settings

- 

  Network settings Info

  Network Info
  vpc-0842ba49a173fa28c

  Subnet Info
  No preference (Default subnet in any availability zone)

  Auto-assign public IP Info
  Enable

  Firewall (security groups) Info
  A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

  [✓ Create security group] Select existing security group

  We'll create a new security group called '**launch-wizard-3**' with the following rules:

  ✓ Allow SSH traffic from — Helps you connect to your instance — Anywhere 0.0.0.0/0
  ☐ Allow HTTPS traffic from the internet — To set up an endpoint, for example when creating a web server
  ☐ Allow HTTP traffic from the internet — To set up an endpoint, for example when creating a web server

  - Set `Auto-assign public IP` to `Enable`, if not already the case
  - Click on "Create Security Group"
  - Keep the existing default SSH option unchanged in the Security Group
  - Click `Add security group rule`. Set Type as `Custom TCP`, Protocol as `TCP`, Port Range as `25565`, Source Type as `Custom`, Source as `0.0.0.0/0`, Description as something memorable like `Minecraft Port`
    - image
- Click `Launch Instance` button in bottom right to finish EC2 instance setup

SSH into EC2 Server

- Wait a few minutes for the EC2 instance to start up, then return to the EC2 main menu (search `EC2` in the upper left search bar)
- Click on `Instances` link in left navbar
- Click on `Instance ID` on your Minecraft Server
- Once you are sure it's running (check `Instance State` status and refresh the page frequently if it hasn't started yet), click the `Connect` button, then `SSH Client`
  - image
  - Read instructions on the Connect Menu page
  - Double check with instructions, but run `chmod 400 <private SSH key>` if on Unix on your PC (If your local PC is Windows, it should work out of the box). Make sure you are in a directory where you can access the SSH private key. Maybe put that private SSH key in your PATH
  - Connect to your instance with the public DNS, e.g. `ssh -i "<private key filename>" <public DNS>`, where the values in brackets are stand-in values. Go back to the `Instances` menu for your EC2 instance if you want to double check the public DNS value
    - Example: `ssh -i "lab7.pem" ec2-user@ec2-54-175-58-38.compute-1.amazonaws.com`
    - image
- Make sure you can SSH into your EC2 instance before proceeding further

Installing Software Dependencies

Install Java 17

Run the following commands after SSH'ing into your EC2 instance:

- `wget https://download.java.net/java/GA/jdk17/0d483333a00540d886896bac774ff48b/35/GPL/openjdk-17_linux-x64_bin.tar.gz`
  - Install the tar archive for Java 17 (OpenJDK 17)
- `tar xvf openjdk-17_linux-x64_bin.tar.gz`
  - Extract the archive with `tar` command
- `sudo mv jdk-17 /opt/`
  - Move the extracted folder into `/opt/`

-
  ```
  sudo tee /etc/profile.d/jdk.sh <<EOF
  export JAVA_HOME=/opt/jdk-17
  export PATH=\$PATH:\$JAVA_HOME/bin
  EOF
  ```
    - Run these commands one line at a time! Sets the PATH for Java
- `source /etc/profile.d/jdk.sh`
    - Source your profile file
- `echo $JAVA_HOME`
    - Check where your Java executable is located (should get an output of `/opt/jdk-17`)
- `java -version`
    - Double check Java version. Should show an output of something like:

        -
          ```
          openjdk version "17" 2021-09-14
          OpenJDK Runtime Environment (build 17+35-2724)
          OpenJDK 64-Bit Server VM (build 17+35-2724, mixed mode, sharing)
          ```

Install Minecraft Server Jar

- `sudo su`
    - Probably a better way of doing this but for now being a sudo user ensures the next parts work as intended
- `mkdir /opt/minecraft/`
    - Add directories for where the `server.jar` will go
- `mkdir /opt/minecraft/server/`
    - Add directories for where the `server.jar` will go
- `cd - Add directories for where the`server.jar`will go`
    - Navigate to the directory where the `server.jar` will go
- `wget https://launcher.mojang.com/v1/objects/c8f83c5655308435b3dcf03c06d9fe8740a77469/server.jar`
    - Install the `server.jar` in the `/opt/minecraft/server/` directory

User Permissions

- At this point you should still be signed in as root (due to `sudo su`)
- `cd`
    - To return to root directory
- `exit`
    - To sign out of root user
- `whoami`
    - To check default (non root) username. It should say `ec2-user` for the default user. If not, write down what the username for the next steps and substitute accordingly
- `sudo chown -R ec2-user:ec2-user /opt/minecraft`
    - Give the `ec2-user` elevated permissions in the `opt/minecraft` directory
- `sudo chmod -R 750 /opt/minecraft`
    - chmods this `opt/minecraft` directory the right Linux file permissions. In this case, 750 gives the user read/write/execute and read/execute to people in your organization.
        - Further reference on 750 permissions: https://chmodcommand.com/chmod-750/
- `sudo visudo`
    - Edit file - Add this to the bottom of visudo file:
        - `ec2-user ALL=(ALL) NOPASSWD: /usr/bin/java -Xmx1024M -Xms1024M -jar /opt/minecraft/server.jar nogui`
    - This adds the ec2-user as a sudo user, which might be a security issue, but it was a way to let this user run the Minecraft server
        - More Information on editing visudo / sudoers file: https://www.digitalocean.com/community/tutorials/how-to-edit-the-sudoers-file

Running the Minecraft Server

- Return to the server directory with `cd /opt/minecraft/server`
- Make sure you are still signed in as `ec2-user` via `whoami`
- Run `java -Xmx1024M -Xms1024M -jar /opt/minecraft/server.jar nogui` to see if the server at least runs
- If it's your first time running the server, after running the server you will get an error relating to EULA.
    - Fix with `vi eula.txt`, set `eula=true`

Automating Server Start

- Check if this command works as a standalone before adding to systmctl
    - `sudo -u <username> <java executable path> -Xmx1024M -Xms1024M -jar <path to server.jar>/server.jar nogui`
    - Example: `sudo -u ec2-user /opt/jdk-17/bin/java -Xmx1024M -Xms1024M -jar /opt/minecraft/server/server.jar nogui`
    - If in doubt, check `whereis java` to find Java executable path
- Once you're sure that the command above works, then let's add a sytemd file:
    - `sudo nano /etc/systemd/system/minecraft.service`
    - Add the following:

        -
          ```
          Description=Minecraft Server
          After=network.target

          [Service]
          User=ec2-user
          WorkingDirectory=/opt/minecraft/server
          ExecStart=/opt/jdk-17/bin/java -Xmx1024M -Xms1024M -jar /opt/minecraft/server/server.jar nogui
          Restart=on-failure

          [Install]
          WantedBy=multi-user.target
          ```
- Run the following in root directory as `ec2-user`:
    - `sudo systemctl daemon-reload`
    - `sudo systemctl start minecraft`
    - `sudo systemctl enable minecraft`
- Reboot the EC2 instance now, ssh back in
- Use `sudo journalctl -u minecraft.service --t` to check if the server is running, press down arrow keys to scroll all the way down
    - image
    - Check back periodically if you just restarted the EC2 instance
        - `Ctrl C` to leave the logs, then check `sudo journalctl -u minecraft.service --t` again
    - Not sure on the details on how it works but `--t` helps you skip to near the bottom of the output logs

Minecraft Client

- Get Minecraft from https://www.minecraft.net/en-us
    - No need for too much detail here right? I did get the paid version of Minecraft for PC (Java edition) if it matters
- After installing and booting up the Minecraft Launcher, go to Installations menu, then New Installation, then `release 1.18.2`
    - image
    - This is important because 1.18.2 client is needed based on the server installed (e.g. Java 17) in previous steps. Other versions of Minecraft Client may not be compatible!
    - Create the 1.18.2 installation and launch it
- After launching 1.18.2 Minecraft client, select `Multiplayer`, then `Proceed`, then `Direct Connection`. Enter in the Public IPV4 address as stated on your EC2 instance
    - image
    - image
    - image
    - image
- Have fun on your server. Dig a hole in the ground