Group 2: Minecraft Server

James Ruh

Ya “Quinn” Huang

Christopher D’Agostino

Jenna Abbott

We started with the idea of running a Minecraft server on a cloud computing service. The idea was to take the Computer Science Club Minecraft server and migrate it to the cloud. This project would give us practical hands on experience, working with the cloud.

For our first attempt, we chose to use CloudLab. The reason for choosing CloudLab was experience and cost. We worked with CloudLab in class and thus had some exposure to it. There were also directions provided via lab instructions that we could refer to as a basis to begin. The second consideration was cost. Due to our student status and through this class, we were able to gain access to CloudLab and its computing resources at no additional cost. However, we quickly moved away from this as we ran into a few critical problems.

On CloudLab, we started with the OpenStack profile. Spinning up an experiment, likely due to coronavirus, took longer than usual.

We started with the OpenStack profile and attempted to replicate the process we used in class for uploading Alpine image. However, when we tried this process with Ubuntu, it did not work. First, adding the Ubuntu image we created with VirtualBox took a long time to upload. We were only able to upload the image on the fifth attempt. We also attempted to use a pre-built VMDK Ubuntu image, and this was not better.

In the end, we moved away from CloudLab due to the hardware limitation it has. Although we can request for more hardware, due to the nature of CloudLab being a public, shared cloud, it was difficult to get enough resources to have a Minecraft server that runs smoothly, especially given our goal. We were aiming to be able to share the server with multiple users. Google Cloud allowed us to create the project using a free trial with extra money for future projects.

CloudLab did create an issue by not being an IaaS. There has to be multiple software to be used in conjunction, whereas Google Cloud is capable of providing all of the necessary technology in one unified platform. As opposed to needing to download or create your own virtual machine file, such as a .vdi or .vmdk file, Google Compute Engine enables you to select which operating system image you would like to use as well as what type of hardware you want all within a very self explanatory, easy to navigate user interface that does not require you to manually download and create virtual machine files on your own computer. For this project, we chose to use the latest version of Ubuntu.

Additionally, because of the nature of CloudLab being an academic cloud, by requesting for extension and more hardware resources (which is necessary if we wanted multiple users on the server), those resources are not being utilized for academic research. Instead, it makes more sense to use a commercial service because we can request for more resources and we can use this experience as practice for working with industry tools. For example, Google Cloud or other commercial IaaS allow you to easily backup files from your computer instances. Another benefit is that our team will not ever have to request to use more resources, which is something that would have been detrimental to running a Minecraft server.

We did not immediately transition into Google Cloud. One of our members explored the Google Cloud option. We can immediately create an Ubuntu Server using the public image base available on Google Cloud. Therefore, we decided to switch over to Google Cloud since installing Ubuntu Server was not our learning objective. Our objective was to launch a Minecraft server on a cloud system. Therefore, we decided to go with a commercial option as it has a simple solution available. We chose Google Cloud over Microsoft Azure and Amazon AWS due to the free trial Google was offering. Our first attempt is just to install the Minecraft image directly using wget. We ran into some issues here with setting up the server and we ended our first attempt.

Then we started our second attempt. Within the Google Cloud, we started the project by going to “Compute Engine” to find the VM instances option. This function was to enable images to be uploaded to the cloud. It had similar functions to OpenStack, in the sense that it enables you to upload and instantiate an image within the infrastructure. However, Google Compute Engine on Google Cloud is much more streamlined and easier to set up. We needed to create an instance for our Minecraft server. We named our instance after Minecraft.

Within the Boot disk menu, we changed the operating system to Ubuntu. It also allowed us to change our boot disk type to a persistent SSD (solid state drive), which was not a function in OpenStack. This means that the files on our server would be stored in a SSD, giving us better receive time. The overall options were more specific than our previous trials. The storage option allowed us to create a backup that allowed us to Read Write to it.

For network tags, we added minecraft. We changed options within the network configuration menu. The server had to be created so we set the network to default for the main and subnet. The external IP was the same for the Minecraft server, so that we were able to connect to the instance and Minecraft server. There was an issue within the instances. During our second attempt, the instances could not be stopped, restarted, or deleted because the information would be gone. This should not happen and we suspected that it was an issue with Google Cloud as this did not happen during later attempts.

The files for the Minecraft server had to be downloaded and installed. The system files for Java had to be updated so that Minecraft could be run properly on Java. The `wget` command was used to get the file from the link. The end-user agreement had to be validated so we had to edit it and save it. In order for the server to be running when it is in the background, we had to use a command. The firewall rules within the VPC network menu had to be changed for the server to run. The tag that was created previously in the instance part will be used as the name of the firewall rule. The source IP had to be created so people within any IP range could connect to the server. Minecraft can use direct connect with our server address for Minecraft to be working for the players.

We were able to reach our second deliverable by the Minecraft server launching on Google Cloud using Google Compute Engine. The project was attempted on CloudLab with OpenStack, but we were not able to get our expected result. Through Google Cloud, we had the resources we needed but was also able to go into more detail with the instance.

Our third deliverable will come from improvements that we found the server needed from the second deliverable. Other ways we can improve is for the game to lessen view distance within the world by modifying the server properties file. A script can be created for the next deliverable, which will enable us to quickly create a new server without needing to repeat the same steps. We can also enable our instance to initiate automatic backups of the server's files to Google Cloud. This way, if something goes wrong with our server we always have a back up. SSH keys will be created so anyone on our team will be able to SSH into the Ubuntu instance in order to make modifications or changes. It would also be a good idea to stress test the server in order to determine whether or not we need to choose to improve the hardware of our infrastructure through adding additional RAM or choosing a better CPU.