Bachelor's thesis

Information and Communications Technology

2021

Van Hiep Vu

DEPLOYMENT AND HOSTING OF AN ONLINE GAME SERVER

– Case: Google Cloud Platform

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2021 | 49

Van Hiep Vu

DEPLOYMENT AND HOSTING OF AN ONLINE GAME SERVER

- Case: Google Cloud Platform

This thesis aimed to utilize the knowledge of system administrator and networking to create and configure a private online game server on the cloud. The general concepts of cloud computing and different aspect of Google Cloud as a platform: history, features, virtual machine types and pricing are introduced in the thesis. For the practical part of the thesis, CS:GO was the selected game based on the requests from players. Because the instance used Linux as the main operating system, it required a computer processor with at least one core running at over 2 GHz of frequency, along with at least 1 GB of memory, 30 GB of minimum disk space and network with more than 5 Mbps of bandwidth to handle input and output among 10-12 players at the same time. Next, a standard N1 machine which met the above requirements was selected on Google Cloud as the instance to host the game server. A private SSH key was created and used in PuTTY to establish a remote session to the instance. Through a PuTTY terminal window, LinuxGSM and its CS:GO dedicated server components were installed in order. With Notepad and FileZilla Client, several configuration files in both server and game directories could be accessed and edited to meet the requirements from players. Third-party plugins such as Metamod and SourceMod were added to enable game modifications and administrative commands on the server, while PUG setup plugin provided suitable settings for 5-on-5 matches. Finally, automated tasks were scheduled to ensure that the game server was able to be up to date and run constantly.

After finishing the deployment and configuration, several settings related to Metamod, SourceMod and PUG setup were reviewed. While 10 players and 1 observer were in the server, the hardware performance of the N1 instance was monitored and recorded for over an hour. The presented results showed that configurations operated as expected.

The goal of the thesis, which was to create and host an online game server on a cloud platform, has been achieved. However, the game server can be developed further with additional functions and improvement of quality in the future. In general, cloud servers have been proven increasingly appealing to users because of their capabilities.

KEYWORDS:

Cloud computing, Google Cloud Platform, server, virtual machine, installation, configuration

CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS

AWS	Amazon Web Services
CLI	Command Line Interface
CPU	Central Processing Unit
FTP	File Transfer Protocol
GB	Gigabyte
Gbps	Gigabits per second
GCE	Google Compute Engine
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
HDD	Hard Disk Drive
IO	Input/Output
IP	Internet Protocol
KB	Kilobyte
MB	Megabyte
Mbps	Megabits per second
OS	Operating System
PC	Personal Computer
RDP	Remote Desktop Protocol
SSD	Solid-State Drive
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VM	Virtual Machine
VPC	Virtual Private Cloud
VPN	Virtual Private Network

1 INTRODUCTION

The years 2020 and 2021 were years unlike the others in 21st century because people from all around the globe had to deal with the global pandemic they might never see before. The Covid pandemic had made and is still making a huge impact on daily life of every person to this day. One of the aspects that affected the most was entertainment. Since many public places such as cinemas, music theatres, and libraries were forced to close, people needed to find different ways to entertain themselves at home. Among all indoor activities, playing multiplayer games might have greatest increase in the number of people who participate in them. According to PCMag UK, Steam – the most popular online gaming platform on PC – hit the highest concurrent user record with over 26,4 million at the beginning of February 2021 (Vincent 2021).

Although all online games have their own official servers which are configured and provided by game developers/publishers, some people still prefer gathering at the same place and creating a local server to play together. There are many reasons why they want to do it, for example: To have lower latency and better network connection; to have the ability to play in their own ways instead of using premade settings on official servers; or to simply have fun and be able to interact directly with each other.

Because of the global pandemic, people cannot stay together in the same place. Therefore, it is impossible to make a local game party so the only way to play is through the Internet at home. However, many players want an online private server to play with their friends instead of random people. Third-party providers also offer pre-made servers ready for a group of players to play with each other, but they require extra tasks from users before joining. Players need a private server that can meet the following requirements:

- They should be able to connect from their computers or laptops any time, anywhere.
- No additional account or software is required for players except the game itself.
- Smooth and stable connection between players and the server.
- Players can change any game modes or settings if needed.

Specifically, the purpose of this thesis is to use Google Cloud Platform (GCP) as the main infrastructure to deploy an always-on private game server that can host online

session for 8-10 players and 1-2 observers simultaneously. The server should meet all requirements listed above. In order to avoid spending unnecessary budget on the project, the suitable offer for the machine will be selected. Tools needed for deployment and configuration processes are Linux Game Server Managers (LinuxGSM), PuTTY, FileZilla Client and Notepad.

The first part of the thesis (Chapter 2) will include introduction of GCP, its feature and functions, all machine types and pricing models that GCP provides. In the second part (Chapter 3), the server implementation process will be presented and explained in detail, along with the performance of the server when all settings have been in place.

2 GOOGLE CLOUD PLATFORM

To understand why more and more people choose Google Cloud to set up their servers, this chapter presents the definition of cloud computing, concept of GCP, current state of GCP in the public cloud infrastructure services market, essential features and functions GCP provides, different types of virtual machines and pricing offered by Google.

2.1 Overview

According to Investopedia (Frankenfield 2020), cloud computing is the delivery of different services through the Internet. These resources include tools and applications like data storage, servers, databases, networking, and software. Cloud computing is a popular option for people and businesses for a number of reasons including cost savings, increased productivity, speed and efficiency, performance, and security. Google Cloud is a could computing vendor, it provides various types of service from infrastructure owned by Google to customers.

GCP is a part of Google Cloud among other tools and services released by Google itself. In April 2008, the Google development team introduced a preview of their new Platform-as-a-Service (PaaS) offering named Google App Engine. (Hunter & Porter 2018). At the beginning, only 10,000 developers were invited by Google to test and give feedback on an early version of App Engine. One month later, 65,000 more developers took part in testing the service. After that, Google finally announced that App Engine was available to open signups.

In the following years, more products and features were added to the platform by Google as shown in Figure 1. Some of the most notable services were Google Cloud Storage (2010), Compute Engine (2013), Cloud SQL (2014) and Kubernetes Engine (2015). They also started to expand into different areas in technology world that needed cloud-native solutions such as infrastructure management, data analytics, IoT and machine learning. Google is currently having 21 datacenters to support all of their cloud services and they are going to operate more in the future. (Sattiraju 2020)

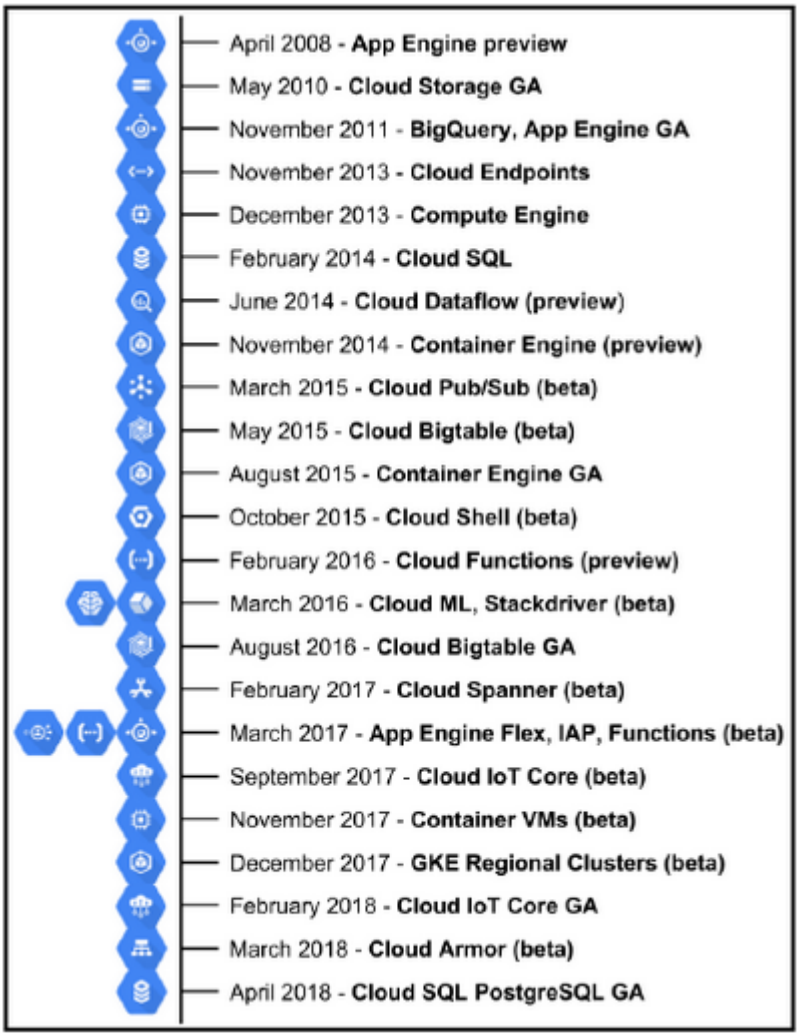

Figure 1. Released GCP products (Hunter, Porter 2018).

By providing powerful global infrastructure which is always ready to operate, Google wants to reduce the cumbersome tasks for developers so they can spend more time focusing on creating applications and solving problems, instead of configuring and tweaking virtual machines.

Nowadays, GCP covers many crucial aspects of technology and industries with its large range of products and services, while still continues to grow at an astonishing speed to compete with others popular cloud service providers such as AWS and Azure, as show in the figure below:

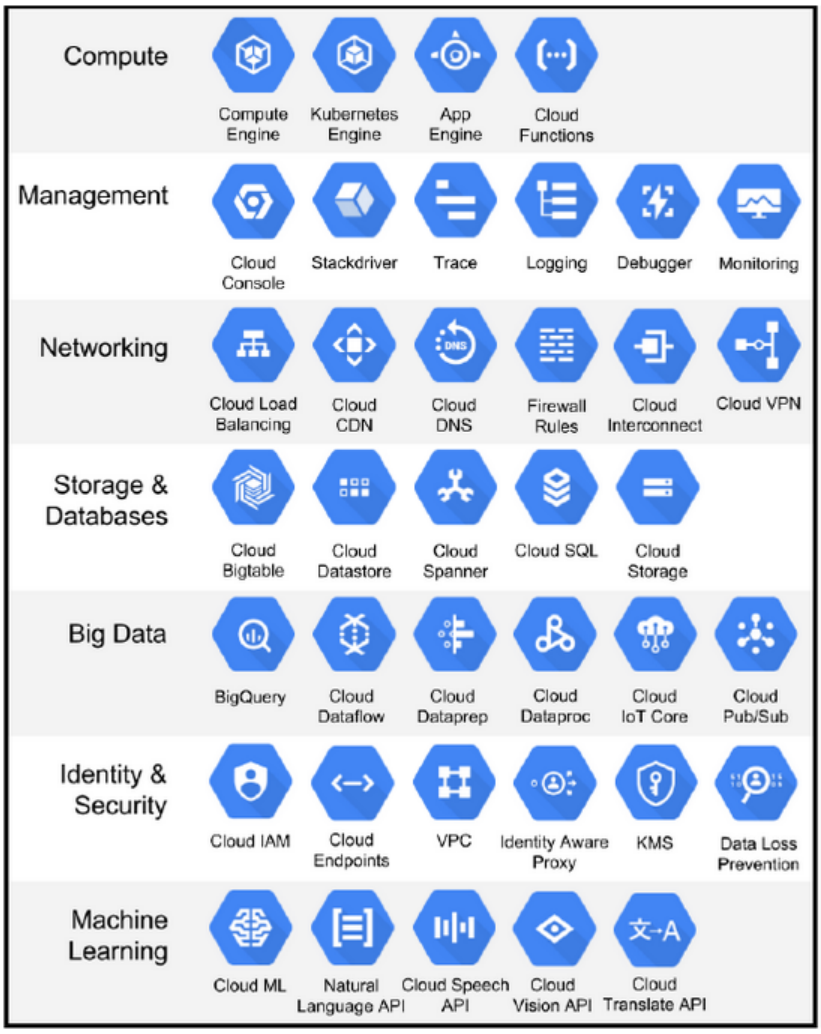

Figure 2. GCP catalogue (Hunter, Porter 2018).

As more and more companies look forward to moving their infrastructure and database to public clouds, the battle to become the best providers for customers has been more competitive day by day. The success of customers will determine popularity and reliability of their chosen cloud provider, prove that they have made the right choice. Although there are plenty of cloud infrastructure providers which bring large increase of revenue year after year, over half of the market still belongs to three biggest names in technology world: Amazon, Microsoft and Google.

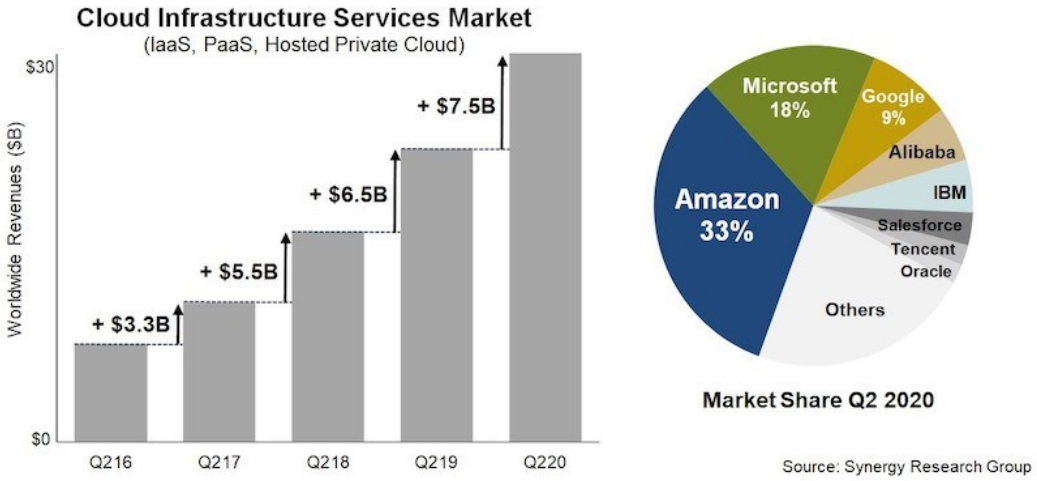

Figure 3. Cloud Market Share: Q2 2020 (Panettieri 2020).

It's not difficult to explain the dominance of Amazon and their service AWS in the public cloud market. Launched in 2006, AWS was the first to enter cloud computing with offerings such as Simple Storage Service (Amazon S3) and Elastic Compute Cloud (EC2). Despite being the oldest provider among others, the combination of experience, adaption to new technology, stability and security improvement and excellent customer support helped AWS earn customers' trust to maintain a large share of the market over a decade. Today, AWS offers over 200 fully featured, ready to use services to cover almost every aspect of the industries.

Azure was released by Microsoft nearly four years after the launch of AWS. In spite of coming later into the cloud services market, Azure didn't take long to become the best choice for many enterprises, entrepreneurs and governments that have been a part of Microsoft ecosystem for a very long time. With Azure, physical infrastructures with Microsoft's products and services could be migrated seamlessly to the cloud. Besides, Azure quickly became developers' favourite cloud providers by supporting many programming languages like C#, Java, ... since the beginning, which helped them gain a large part of the market.

Despite being the newest one among the trio, Google rapidly secured 10 percent of the cloud market by offering lots of crucial and up-to-date services with reasonable pricing to their customers.

2.2 Features and functions

Like other cloud service vendors, Google also include plenty of features to ensure that developers and enterprises will have a safe, stable and powerful platform to transform ideas to actual products. Because of the project, only features and functions that benefits deploying a server on the cloud will be mentioned below.

2.2.1 Google Compute Engine

Cloud computing completely changes the way virtual machine (VM) operates. On one hand, users neither need to own any physical machines nor worry about maintaining and upgrading them. On the other hand, they can create as many VMs as possible to satisfy their demands. Those VMs on the cloud are called Infrastructure-as-a-Service (IaaS).

Google Compute Engine (GCE) is the IaaS component of the GCP that lets users create and run VM on Google's infrastructure. Each VM is called a Compute Engine instance. A Compute Engine instance can run Linux and Windows Server images provided by Google or any customized versions of these images. Users can also build and run images of other operating systems (Ravi, Raj et al. 2018).

With GCE, users can assign the number of virtual CPUs (vCPUs) and the amount of memory (in GB) for their instances, by choosing one from a wide range of predefined machine types or by creating their own custom machine types.

Since the release in December 2013, GCE had added many tools and features to help its users create, deploy, and manage large numbers of VMs in more effective ways.

2.2.2 Storage solutions for GCP

Google provide various available storage solutions on the platform to satisfy different needs from their customers. Those solutions are persistent disks, local SSDs, boot disks and external storage solutions like Google Cloud Storage (GCS) and managed databases. GCS and managed database will not be explained as they are out of scope.

Like many other cloud service vendors, Google also use a technology called block storage (or block-level storage) to store data on their environment. Each of those blocks can be configured separately to work with different VMs or operating systems. Because of its flexibility and availability, block storage has become a standard solution to handle

data on the cloud. With GCE, the block storage solution for their VMs is known as persistent disks.

Persistent disks are network attached drives that have been engineered to provide extremely consistent IO performance within Google's datacenters. These disks are available in both standard platter and SSD, and Compute Engine provides a number of methods to act on them. Instead of a physical disk, every persistent disk on Compute Engine is actually composed of data spread across several physical disks. This provides redundancy for all data stored on persistent disks, allows Google to abstract away disk hardware failures from the VMs. Persistent disks offer a number of out-of-the-box features, including automatic encryption at rest and online resizing. (Hunter & Porter 2018)

There are two types of persistent disks on GCE, standard and SSD. Standard persistent disks are backed by HDDs, with good IO performance while maintaining the low cost and are usually used in cases where large data storage is more important than performance. As the name shows, SSD persistent disks are backed by SSDs. Although SSD persistent disks provide better performance than standard ones, their cost per GB is relatively higher. Because persistent disks are distributed network storage devices, their performance is limited by hardware and network. The methods to reduce bottleneck depend on the type of persistent disks: With standard one, increasing the size of the disk will improve IO performance. With SSD, increasing the number of vCPU on a Compute Engine will improve IO performance.

No matter which types of persistent disks is selected, every GCE instance must have a boot disk. When creating a new VM, users must also create a boot disk by specify the type (standard or SSD), size of the disk and an image for the disk. The VM image can be a public image, a custom image or another boot disk’s snapshot image. While other persistent disks can be added and removed from running instances, the boot disk must always stay with an instance, because it includes OS and kernel files so that the instance can operate. Normally when a VM is destroyed, its boot disk is also removed. This function can be disabled by using command or web console.

As mentioned above, persistent disks are implemented as a network storage solution, so their IO performance is sometimes not high enough for applications. GCE provide a solution called local SSDs to deal with the situation. They are dedicated SSD drives which are attached directly to the physical servers hosting the running VM. Although local

SSDs significantly boost the IO performance of the instance, they do not have the same level of redundancy as persistent disks. If a VM is stopped or destroyed, attached local SSDs cannot be preserved. Therefore, they are the best choice for applications such as caching or transferring unnecessary and temporary data through the network. The size of each local SSD is 375 GB, a GCE instance can add up to eight local SSDs. Furthermore, they must be attached when creating an instance, since a running VM is unable to add local SSDs.

2.2.3 Networking in GCP

In order to have two-way connection between users and the cloud, network must be established. With GCP, networking is very crucial. Similar to others, a network in GCP has the following components: private networks, subnets, firewall rules, routes and IP addresses. To make full use of GCP, users must know how these components operate and interact with each other.

The fundamental building block of networking on Google Cloud is the Virtual Private Cloud (VPC) network, often referred to simply as network (Hunter & Porter 2018). AWS also has VPC in their network. Despite sharing the same term, these networks operate differently. AWS VPC networks are regional, meaning that two VMs in two separate regions will reside in different VPCs. Meanwhile, VPC networks in GCP are global resources with project-specific scope, two GCE instances in two different continents can stay in the same VPC. A single network is able to connect Google Cloud resources across zones and regions, but not across projects. It creates the isolation between cloud resources, hence improves the security in GCP. There are many products and services on GCP that can utilize VPC networks, GCE is one of those which benefits the most.

Every Google Cloud project comes with a single default network when it is created. GCE instances will use the default network if no network is chosen. Because Google VPC networks are global, their compute resources are able to communicate with each other across all regions and zones, without leaving the network. Beside default network, additional custom networks can also be created. Despite being called “default network”, it can be modified or deleted like other network resources.

Networks provide a primary layer of isolation between systems. Components within the same network can communicate securely with one another using internal IPs, whether

they are in the same datacenter or on different continents. Conversely, resources in different networks cannot communicate with one another unless explicitly configured to do so, such as when using public IPs or VPN tunnels. As a result, systems can be isolated from one another while still existing in the same project by simply running on different networks. (Hunter & Porter 2018)

Each VPC network contains one or more subnetworks, or subnets which segment the network into dedicated IP address ranges. Unlike networks which are global, subnets are regional. A network can have one or multiple subnets on a region, but each subnet can only belong to one network and one region, for example:

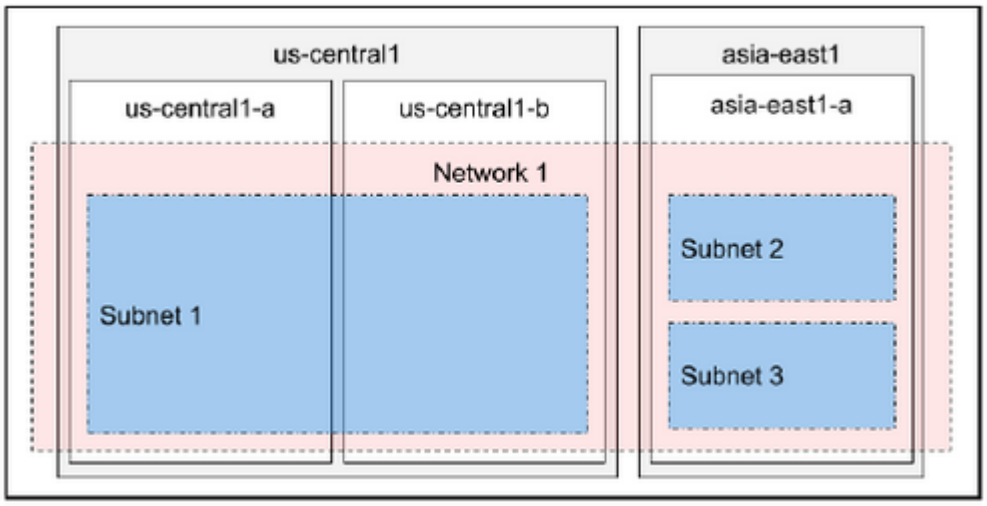

Figure 4. Example of Google network and subnets (Hunter, Porter 2018)

According to Hunter, Ted and Steven Porter (2018), networks are created with one of two subnet modes: custom or auto. Networks using auto mode are created with one pre-configured subnet for each region. Subnets within a network must have non-overlapping IP address ranges, so auto mode provides a convenient way to avoid the need to carefully plan out subnet address ranges. Conversely, networks created in custom mode are created without subnets. This is useful in situations where teams want more control over the allocation of IP addresses within the network, such as when using many subnets within one region or using Google Cloud Interconnect.

Ingress and egress traffic of GCE instances in the network are controlled by firewall rules. Firewall rules can be applied to the entire network, subnetworks or specific compute resources based on tags. No matter which types of VM or OS, as long as they are in the same network, they will have similar firewall rules. Each firewall rule includes six

components, which will together determine whether the incoming or outgoing connection is allowed or denied. Those components are:

- Action: The firewall rule action can be either *allow* or *deny*. Depends on the given criteria, an *allow* action permits the connections, while a *deny* action blocks the connections.
- Direction: The direction of a firewall rule can be either *ingress* or *egress*. Incoming requests to the instance are called *ingress*, while outgoing requests from the instance are called *egress*. When the direction is unspecified, Google will use *ingress* as default behaviour of direction.
- Target: The target of a firewall rule is based on the direction of the rule. For *ingress* rules, the target is the destination instance. For *egress* rules, the target is the source instance. Google Cloud provide three options to specify a target: *all instances in the network*, *instances by target tags* and *instances by target service accounts*. If no target is specified, the firewall rule applies to all instances in the network.
- Source or destination: Either source or destination can be specified, depending on the direction of the firewall. For ingress rules, one of the following source parameters is accepted: *IP address ranges*, *tags*, or *service accounts*. IP address ranges can be combined with either tags or service accounts to use for source parameter. For egress rules, the destination parameter only accepts IP address ranges. If no source or destination is specified, the firewall will permit all traffic with an IP address range of 0.0.0.0/0.
- Protocol and port: The scope of a firewall rule can be narrowed by one or many combinations of protocol and destination port. Protocol is mandatory, while destination port is optional and may be a port number or port range, together they will define the firewall rule on that protocol and its destination port, or destination port range. If no protocol is specified, the firewall rule will apply to all protocols and their destination ports.
- Priority: The priority of firewall rule is an integer from 0 to 65 535. The lower the integer is, the higher the priority will be, which means 0 is the highest priority, while 65 535 is the lowest priority. Higher priority rules always override lower priority rules, if two rules have identical priority, the rule with deny action overrides the rule with allow action.

All Google VPC networks are created with two hidden firewall rules. The first rule is an egress rule with allow action, destination 0.0.0.0/0 and lowest possible priority 65535, this rule allows outgoing traffic from any instance in the network to any destination. The second rule is an ingress rule with deny action, source 0.0.0.0/0 and lowest possible priority 65535, this rule denies incoming traffic to all instances in the network. Although these hidden rules cannot be removed, it is not difficult to override them because they have the lowest possible priorities.

When the default network is created, it has the following pre-populated firewall rules with same second-to-lowest priority 65534:

- *default-allow-internal*: Allows ingress connections for TCP, UDP and ICMP on all ports within the network.
- *default-allow-icmp*: Allows ingress ICMP traffic from any source to any instance in the network.
- *default-allow-ssh*: Allows ingress connections to access SSH via TCP on port 22 from any source to any instance in the network.
- *default-allow-rdp*: Allow ingress connections to access instances by using Microsoft RDP via TCP on port 3389 from any source to any instance in the network.

While firewall rules determine which packets may be sent and received by instances on the network, routes determine how those packets are directed through the network. VPC networks automatically provide routes for directing traffic internally between instances, as well as a route for directing egress traffic to external addresses. In most cases, these default routes are sufficient for handling network traffic (Hunter & Porter 2018). Each route has the following core components: A destination IP range, a priority, an optional description, optional instance tags and an optional next hop. When a network is created, it includes two types of system-generated (default) routes: The first route is default route with destination IP range 0.0.0.0/0 and priority 1000, this route defines the path to the Internet. The second routes are subnet routes, they define paths to resources in a network. The default subnet route of each subnet has the same destination IP range as the primary IP range of the subnet itself. The destination IP range of a subnet route always has the longest subnet mask. Because Google Cloud assigns 0 as the priority for all default subnet routes, they cannot be overridden by other routes.

Similar to physical devices in a network, IP addresses are also used on GCE instances to communicate among VMs and the Internet. Addresses can be a combination of two types:

- Internal/external IP addresses: As the name suggests, internal IP (or private IP) addresses are used to establish connections between resources in the same VPC network, they cannot be reached from outside of the network unless using Google Cloud VPN, while external IP (public IP) addresses provide access to resources in the network from the Internet. Two resources in two separate regions can communicate via internal IP addresses, as long as they are in the same VPC. However, two resources in different VPCs have to communicate via external IP addresses, even if they are in the same region. Network interfaces in GCE instances can be assigned one primary internal IP address, one external IP address and any number of secondary internal IP addresses. Primary internal IP address of each interface is unique in the VPC network. A specific primary internal IP address can be assigned when creating a VM instance, or it can be reserved for the project and assigned to resources in that project later. If no primary internal IP address is selected, GCE will automatically assign one to the instance. Secondary internal IP addresses are optional, they are useful for running multiple services on a single VM because each service can be assigned a secondary internal IP address to access independently. External IP addresses are also optional. Although instances are not directly accessible from the outside of VPC network without an external IP address, they can be connected indirectly by using routes and proxies. With protocol forwarding, a single instance can attach multiple external IP addresses on itself. All internal IP addresses can only be IPv4 because VPC networks don't support IPv6 traffic among resources in it, while external IP addresses supported both IPv4 and IPv6 traffic.
- Ephemeral/static IP addresses: Ephemeral IP addresses are generated and assigned to instances by Google Cloud when IP addresses are not specified. As the name implies, ephemeral IP addresses depend on the lifetime of the instance they belong to. When the instance is stopped (by shutting down or restart) or removed (deleted), the attached ephemeral IP address is released and will be assigned to another instance then. The way ephemeral IP addresses work is similar to DHCP in local network, yet client IP address of DHCP will be re-assigned or renewed when the lease time expires, while ephemeral IP address will stay until the instance goes down. On the contrary, static IP addresses must

be reserved before using, they will remain even when the attached instances are stopped or removed.

Both internal and external static IP addresses are supported by GCP. In internal networking, reserving an internal static IP address simply removes that address from the pool of automatically assignable addresses. Because internal addresses belong to a subnet, with a specific IP address range, static internal IP addresses are regional, and can only be assigned to resources attached to their subnet (Hunter & Porter 2018). When creating an instance, an existing internal static IP address needs to be assigned because it cannot be changed afterwards. If no static IP is specified, Google will assign an ephemeral internal IP address to the instance, it can be promoted to a static internal IP address later. Static external IP addresses can be reserved and assigned to a GCE instance on any network within the project. By default, static external IP addresses are IPv4. They can be attached or removed from current GCE instances of the project. When an instance is removed, its external static IP address is available to use on one of the resources within the project. Similar to internal IP addresses, Google will also assign an ephemeral external IP address to the instance when there is no specified static IP, that address can be promoted to a static external IP address later.

2.2.4 Security on GCP

Hackers and cybersecurity breaches are one of the biggest issues for many online service providers and Google is no exception. Before entering the cloud market, Google was known for their remarkable search engine and their biggest video sharing platform on the Internet – YouTube. In order to improve the precision of a search result for a website or a video, Google must store the generated data from users so that they can analyse for better outcome. Engineers and staffs at Google, without a doubt, were fully understood that their valuable data might be at risk if they didn't have any proper security measures. Google and other big cloud service providers have invested lots of money and time to achieve the highest possible level of protection for their platforms. Here are some of the main features being used on GCP:

- Datacenter physical security: Each Google datacenter is equipped with the following items: anti-climb fences, vehicle barriers, biometric sensors, high technology cameras and alarms etc. Together, all of them create a solid defense system to protect the area.

- Custom hardware and trusted booting: To protect all servers within a local network in datacenter from malicious code injection, Google has designed and built every component in-house: Ordered items from vendors are validated, server boards and network devices for server machines are customized. All low-level components: BIOS, bootloader, kernel, and base OS are pre-loaded with cryptographic signatures to validate during boot process. Custom hardware security chips are installed on all devices in datacenter for authentication.
- Data disposal: Data ready to be deleted permanently from persistent disks and other storage devices will be fully handled by Google. First, the drive will be wiped logically. Next, it will be inspected to ensure that data cannot be recovered by any means. Finally, the cleaned drive is reinstalled to the system for reuse. Entire procedure is logged and saved. In case of damaged drives that are unable to be wiped or reuse, they will be moved to a secure place for complete destruction in the future.
- Data encryption: By default, all customer data on GCP is always encrypted automatically by Google. Disk drives and master keys are also encrypted.
- In-built DDoS protections: Networking and load balancing services within GCP are fortified by Google to protect instances against DDoS attacks such as IP packet flood, port exhaustion, ...
- Insider risk and intrusion detection: All devices in entire Google infrastructure are continually monitored by Google to check for uncommon behaviours. Employees' accounts are secured by using compatible security keys with Universal 2nd Factor (U2F) authentication standard.

Besides above features, human is also a key factor. Very few Google employees and security guards are allowed to work in a datacenter, their daily activities inside the area are always recorded and monitored. In fact, only 1% members of Google have the opportunity to visit their company's datacenter and the inspection process for them to leave the building is stricter than coming in.

2.3 Compute Engine types and pricing models

VMs on Compute Engine are available in a number of configurations, known as machine types. Machine types can be categorized by use case as well as scale, where use case determines relative resource allocations, and scale represents total resource allocation

for that type (Hunter & Porter 2018). Each machine types includes system memory (RAM), virtual CPU (vCPU, each vCPU equals one physical CPU core), persistent storage for boot disk and data, and network interfaces. Google offers a variety of machine types with the following specifications:

- General-purpose machine types: Provide a wide range of options for beginners with great price/performance ratio to handle day-to-day workloads.

Table 1. General-purpose machine specifications (Google 2021a).

Machine types	Memory per vCPU (GB)	vCPUs	Maximum egress bandwidth (Gbps)	CPU platforms
E2 standard	4	2-32	4-16	Intel Skylake Intel Broadwell Intel Haswell AMD EPYC Rome
E2 high-memory	8	2-16	4-16	
E2 high-CPU	1	2-32	4-16	
N2 standard	4	2-80	10-32	Intel Cascade Lake
N2 high-memory	8	2-80	10-32	
N2 high-CPU	1	2-80	10-32	
N2D standard	4	2-224	10-32	AMD EPYC Rome
N2D high-memory	8	2-96	10-32	
N2D high-CPU	1	2-224	10-32	
N1 standard	3.75	1-96	2-32	Intel Skylake Intel Broadwell Intel Haswell Intel Sandy Bridge Intel Ivy Bridge
N1 high-memory	6.50	2-96	10-32	
N1 high-CPU	0.90	2-96	10-32	

- Custom machine types: General-purpose machine types with custom number of vCPUs and amount of memory defined by users.

Table 2. Custom machine specifications (Google 2021a).

Machine types	Memory per vCPU (GB)	vCPUs	CPU platforms
E2 custom	0.5-8 Extended memory: Not available	2-32	Intel Skylake Intel Broadwell Intel Haswell AMD EPYC Rome
N2 custom	0.5-8 Extended memory: Up to 640 GB per VM	2-80	Intel Cascade Lake
N2D custom	0.5-8 Extended memory: Up to 768 GB per VM	2-96	AMD EPYC Rome
N1 custom	0.9-6.5 Extended memory: Up to 624 GB per VM	1-96	Intel Skylake
		1-64	Intel Broadwell Intel Haswell Intel Ivy Bridge

- Compute-optimized machine types: Are optimal for compute-intensive workloads with great CPU performance. They only include predefined C2 machine types. C2 machine run on Intel Scalable Processors (Cascade Lake), support 4 GB of memory per vCPU, a wide range of 4-60 vCPUs, and 10-32 Gbps of default egress bandwidth. (Google 2021b)
- Memory-optimized machine types: Are optimal for memory-intensive workloads such as databases and analytics thanks to high amount of memory per vCPU (at least 14GB to 28GB memory per vCPU).

Table 3. Memory-optimized machine specifications (Google 2021c).

Machine types	Memory per vCPU (GB)	vCPUs	Maximum egress bandwidth (Gbps)	CPU platforms
M2-ultramem	28	208/416	32	Intel Cascade Lake
M2-megamem	14	416		
M1-ultramem	24	40/80/160		Intel Skylake
M1-megamem	14	96		Intel Broadwell E7

- Accelerator-optimized machine types: Unlike other machine type families, accelerator-optimized VMs (A2) feature NVIDIA A100 GPUs with 40 GB of dedicated graphic memory per GPU, which are optimized for parallel computing jobs such as machine learning and high-performance computing.

Table 4. Accelerator-optimized machine specifications (Google 2021d).

Machine types	Memory per vCPU (GB)	vCPUs	Maximum egress bandwidth (Gbps)	GPUs
A2-highgpu	7	12-96	24-100	1-8
A2-megagpu	14	96	100	16

A2-megacpu machine types don’t support Windows OS.

- Shared-core machine types: Are cost-effective and ideal for applications that require lower resource than general-purpose machine types by sharing a physical core between vCPUs. They also have a process called “bursting”. Bursting is enabled automatically and takes effect whenever processes exceed the available

resources. The ability to burst at any given time is not guaranteed (Hunter & Porter 2018). No extra fee is charged during bursting. However, a vCPU can only burst for a short period.

Table 5. Shared-core machine specifications (Google 2021a).

Machine types	vCPUs	Fractional vCPUs	Memory (GB)	Maximum egress bandwidth (Gbps)
E2 shared-core	2	0.25/0.5/1	1/2/4	1/1/2
N1 shared-core	1	0.2/0.5	0.60/1.70	1

Fractional vCPUs are the amount of total vCPU time (in percent) to sustain one or two vCPUs on a physical core for a short period of time, depends on the machine types. For example, e2-micro machine sustains 2 vCPUs with 12.5% of CPU time each, therefore it has 25% vCPU time in total, which equals 0.25 fractional vCPUs.

Most of the resources on GCE must be paid before being in use. vCPU, memory, disk, and networking will be charged to run each VM instance. Users also have to pay for custom images and GPU if needed. For predefined machines, Google calculates the price based on the selected type of machine. For custom machines, Google determines the cost based on the amount of resources. Each instance is charged by measuring its uptime between starting and stopping in second, with a minimum time of 1 minute. Google provides different pricing models for VM instances such as:

- On-demand: Also known as pay-as-you-go, no upfront capital expenditures or time-limited contract are required for this model. Users are charged based on their cloud resources usage. Costs can be adjusted by changing machine types, number of vCPUs, memory, ... Billing will stop instantly when its instances are terminated and destroyed.
- Preemptible: Compared to regular instances, preemptible VMs are up to 80% cheaper with fixed pricing and is a great option for low-budget users who do not require high availability for their applications and services. However, they can be shut down by GCE at any time to provide resources for other workloads, therefore no Service Level Agreement is guaranteed for them. Preemptible instances are

not capable of live migration to a regular instance and cannot be restarted automatically after maintenance.

- Committed use contract: Users can get discount up to 57% for most of the machine types and up to 70% for memory-optimized machine types from Google when they commit to paying for their required compute resources in 1-year or 3-year plan.

Pricing varies between regions and some machine types are not available in a specific region. All requirements must be checked before selecting the region to host GCE instances. For example, if latency of the VM instance is not a concern, customers can choose region with lowest price instead of the one that is close to their location.

3 IMPLEMENTING THE GAME SERVER

This chapter focuses on practical part of the project, which includes introduction of selected game, reasons why the game was chosen; benefits of self-deploying a server instead of using third-party services; hardware requirements to implement a server for the game and to run the game itself. It also shows all steps of installation and configuration process so that the server can run 24/7. In the end of this chapter, measured performance of the server will be presented.

3.1 Game selection

The chosen game for this project is Counter-Strike: Global Offensive (CS:GO), a multiplayer first-person shooter (FPS) developed by Valve and Hidden Path Entertainment. It is the fourth major game in Counter-Strike franchises. After over two years of development, CS:GO was released for Windows, macOS, Xbox 360 (X360), PlayStation 3 (PS3) on August 21, 2012, and for Linux later in 2014. However, Valve has dropped X360 and PS3 versions of the game to fully focus on PC platform. They also stopped collaborating with Hidden Path Entertainment – the company that wanted to develop the game on consoles (X360, PS3) – one year after the release. To this day, CS:GO are still being updated regularly on all OSs with small patches to fix bugs or balance the gameplay, and large patches to add more content. CS:GO runs on Source – an in-house game engine developed by Valve and debuted in 2004. Steam – an online gaming platform which is also developed by Valve – is required to launch and play CS:GO. In the beginning, users must purchase CS:GO at around 15€ directly through Steam or third-party game stores to unlock the game. Nevertheless, on December 6, 2018, Valve decided to remove 15€ price tag for CS:GO, made it fully free to play and introduced a new feature called “Prime”. Players who bought the game beforehand received an exclusive “Loyalty Badge” each and free Prime upgrade for their account. Players with Prime will be in the same queue during matchmaking, while Non-Prime accounts cannot be matched with Prime ones. After recent changes, new players can only get the Prime status for their account by purchasing CS:GO Prime upgrade at about 13€. CS:GO is one of the games with highest concurrent players on Steam at approximately 700,000 per day.

Figure 5. Current CS:GO cover art on Steam (Valve 2021).

CS:GO is currently having 8 different game modes and the most popular one is Ranked Competitive. In Ranked Competitive mode, 10 players with similar skill groups (ranks) will be put into two teams of 5 on the same map which they selected during matchmaking. Each competitive match has a maximum amount of 30 rounds, the first team to reach 16 rounds will win the match, if the final score is 15-15, the match will end in a tie. Valve uses a hidden Elo rating system to determine skill group for each player. This system will adjust individual ranking when the match is over: players on winning team will increase their rating and vice versa.

Many rules of a CS:GO Competitive match are taken from older games in the CS series, with some adjustments. There are always two sides: Terrorists (Ts) and Counter-Terrorists (CTs), with different objectives to win a round. When round 15 of the match is over, both teams receive a short break before switching their side. All of the most played maps in CS:GO are in Bomb Defusal map group (they have the prefix “de_” in the filenames), such as: Dust 2, Inferno, Mirage, ...

To combat against hackers and cheaters in CS:GO, Valve has implemented a system called “Valve Anti-Cheat” (VAC) to the game since 2012. It is still being updated time to time to improve its effective, along with other solutions such as: Machine learning system, Overwatch system (where players become investigators, watch others’ demos which have a suspect and decide if that suspect cheated or not), Prime system, trust factor, ... Players who receive a VAC ban (detected by VAC) or a game ban (detected by Overwatch) will be permanently prevented from joining official CS:GO servers created by Valve. Currently Valve is owning 24 official CS:GO matchmaking servers across multiple regions. All of them are hosted at the same tick rate (frequency) of 64 Hz, which means client and server will exchange data and events 64 times per second. The higher the tick rate, the better the responsiveness is. Nowadays a CS:GO server can handle highest tick rate at 128 Hz. Players are not allowed to choose servers manually, instead they will be put in an official matchmaking server with lowest latency (ping) to their network.

The project began based on the demand for a private CS:GO server by a group of over 10 people so that they can play the game together. There were some alternative options to create and play private CS:GO match below:

- Using FACEIT: FACEIT is the most popular free to play platform for CS:GO players. Everybody must create a FACEIT account, link it with a CS:GO account having Prime status, then download and install its client and anti-cheat system on the computer to play. However, those players don’t want to have an extra account or an anti-cheat which may affect the performance of their computers. Besides, FACEIT anti-cheat does not support Linux or macOS and can only run in Windows when the OS has the latest update.
- Renting a server: Although third-party CS:GO servers can be rented at a decent price with guarantee of performance, they lack the freedom to customise because everything on the server is pre-configured. Furthermore, administrators sometimes refuse to make a change on servers even when it is requested by the renters. For example: Being locked at 10 players per server, no observers or referees are allowed; unable to add community maps to the server; not knowing when the server might go down and when it is up again; ...
- Playing on public community servers: It is difficult for those players to start a match with others because irrelevant players will also look for these free servers as well. Some community servers can only host 2-3 matches simultaneously,

therefore people who come in when the third session has begun will be put into a queue and must wait until one of those 3 matches finishes to join.

With all reasons above, a self-deployed server is the best choice to meet all requirements from this group of players: No extra account or software for client; various maps; infinite configurations; completely private; accessible for new players who don't have Prime status; higher tick rate than official servers by Valve (128 Hz); ... In addition, Valve also released a tool named Source Dedicated Server (SRCDS) which helps users setup and run the server component of a Source engine-based game by themselves; Google ensures that a single Compute Engine instance will not go down longer than 3,6 hours per month (assume that the uptime of a month is 720 hours), so the server can almost stay 24/7.

3.2 Requirements

Google account is required to initialize and configure a GCE instance; CS:GO is exclusive on Steam, therefore a Steam account must also be created. In terms of hardware for a CS:GO dedicated server, system must meet the following specifications:

- CPU: An Intel or AMD CPU with at least 1 core running at 2 GHz of frequency or higher.
- Memory: More than 1 GB is recommended because this server will include community-made modifications (mods) and plugins.
- Disk space: At least 30 GB to store all the game files, mods, and plugins.
- OS: Windows (7 or newer for consumer version, 2012 or newer for server version), macOS (OS X 10.6.6 or newer), Linux (Ubuntu 18.04 or newer).
- Network egress bandwidth: Over 5 Mbps (or 640 KB/s) for 12 players on the server at once.

On client side, here are the minimum system requirements to play CS:GO:

- CPU: An Intel or AMD CPU with 2 cores running at 2.4 GHz of frequency or better.
- Memory: 2 GB of RAM.
- Graphics: Video card with at least 256 MB of video memory.
- Disk space: At least 25 GB.

- OS: Windows (XP or newer), macOS (OS X 10.11 or newer), Linux (Ubuntu 12.04 or newer).

Keep in mind that Linux has many distributions (distros), Ubuntu is only one of them. Nevertheless, this thesis mainly focuses on Ubuntu, so other distros will not be mentioned.

Regarding software, below are the needed tools for installation and configuration:

- PuTTY (Tatham 2021): A free and open-source software terminal emulator for Windows and Unix platforms. It lets users remote computers running supported network protocols such as SSH, Telnet, rlogin, ... with a CLI and can also connect to a serial port. Released on January 8, 1999 by Simon Tatham, PuTTY is still one of the most popular terminal emulator for secure computer remote session. Its latest stable version, 0.75, was released publicly on May 8, 2021. In this project PuTTY will be used instead of built-in Google Cloud Shell to manage GCE instance.
- PuTTYgen (PuTTY Key Generator) (Tatham 2021): A key generator utility for creating pairs of public and private SSH keys. These keys can be used in PuTTY to initiate secure remote connection between computers. Because PuTTYgen is one of the components of PuTTY package, it is also available on both Windows and Unix platforms.
- LinuxGSM (Gibbs 2020): An open-source, easy to use command-line tool on Linux which helps admins set up their game servers quicker by automating the installation and adding extra features for management. Some of the features offered by LinuxGSM are updater, monitor, alerts, backups, debugger, ... It is currently supporting 117 games (including CS:GO) in server deployment, with more to come in the future.
- Notepad: A built-in simple text editor included with all versions of Windows for reading, creating, and editing plaintext files. By default, Notepad saves the files in “txt” format (extension). Text files with compatible formats (batch files, logs, INI files, config files, ...) can also be read and modified in Notepad, as long as they don’t contain special formatting or non-plain text.
- FileZilla Client (Kosse 2021): A free and open-source, cross-platform file transfer solution supports multiple protocols such as FTP, FTP over TLS, and SFTP. It was released on June 22nd, 2001 by Tim Kosse. The latest stable version of

FileZilla Client, 3.54.1, was released publicly on May 13th, 2021 for all OSs. FileZilla also offers Server version – a free application to share files with other systems, and Pro version – a paid application to transfer files between computer and cloud storage solutions (Amazon S3, Dropbox, Google Cloud/Drive, Microsoft Azure/OneDrive, ...).

3.3 Deployment

First, a GCE instance must be created. On “Create an instance” page of Google Cloud Console, configurate as below:

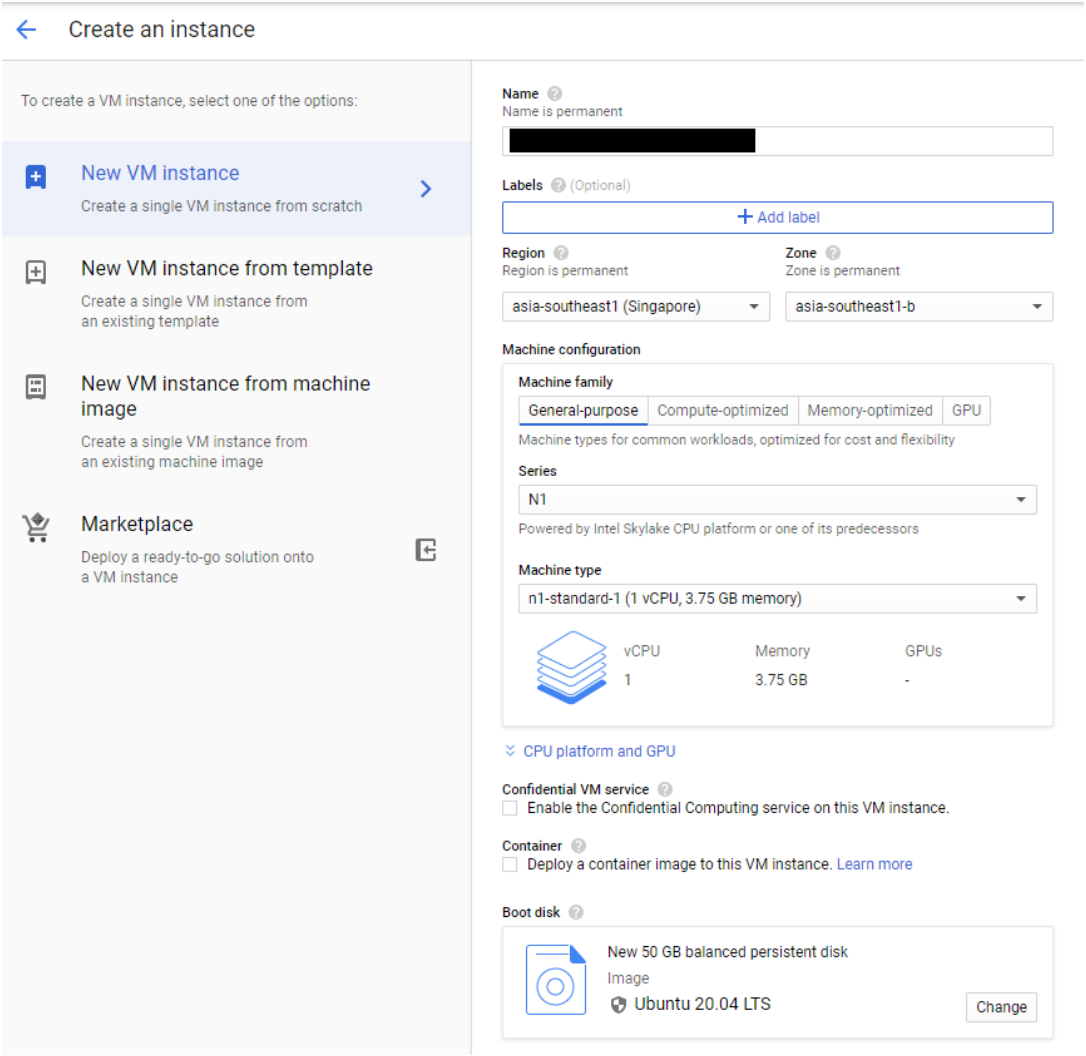

Figure 6. “Create an instance” page on Google Cloud Console.

A unique name is mandatory for each instance in the same project. For the sake of privacy, some sensitive information such as name or IP address will not be shown in the thesis. Because most of the players are living in Asia, “asia-southeast 1 (Singapore)” is chosen as the region. In machine configuration section, a standard N1 machine with 1 vCPU at 2.2 GHz and 3.75 GB of memory is good enough to operate a CS:GO server for 10-12 people. In boot disk settings, a Linux distro is required to run LinuxGSM, so Ubuntu 20.04 LTS is the perfect choice since it is the latest stable version with long-term support from Ubuntu, which guarantees the security and reliability for the server. In terms of disk size, 50 GB of balanced persistent disk is abundant to ensure disk space will not run out. At this point GCE instance is ready to be created and will cost approximately 30€ per month (total uptime is about 750 hours each month).

Next, connection must be set so that PuTTY can establish a session to the instance via public IP address later. By default, GCE will create a primary internal IP address and an ephemeral external IP address. Since the instance region is “asia-southeast1”, its primary internal IP address must be within the range of 10.148.0.0/20. External IP address needs to be switched from ephemeral to static, so the instance will not change its external IP address after each reboot.

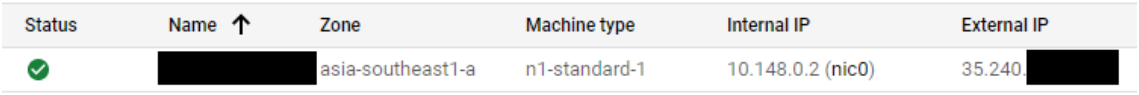

Figure 7. Running instance and its IP addresses.

To guarantee that Steam clients can connect to this instance with given external (public) IP address, an additional firewall rule needs to be created with the following configurations, below ports are used by Steam to exchange traffic with clients:

- Targets: All instances in the network.
- Source IP ranges: 0.0.0.0/0 (any IP address can connect).
- TCP ports: 27015-27030, 27036-27037.
- UDP ports: 4380, 27000-27031, 27036.

Then, PuTTYgen is used to generate a pair of public and private keys, texts being added to the “Key comment” box will be defined as a username of GCE instance. Those keys are saved and backed up in a personal storage to avoid being lost in the future. On GCE menu, open “VM instance detail” page, copy entire key values to the “SSH Keys” section. With this method, instance’s directory can be accessed via applications (in this project

are PuTTY and FileZilla Client) by using generated key to establish a session with SSH connection. On top of that, the directory is protected from unauthorized access because only people who have the private key can make changes on it.

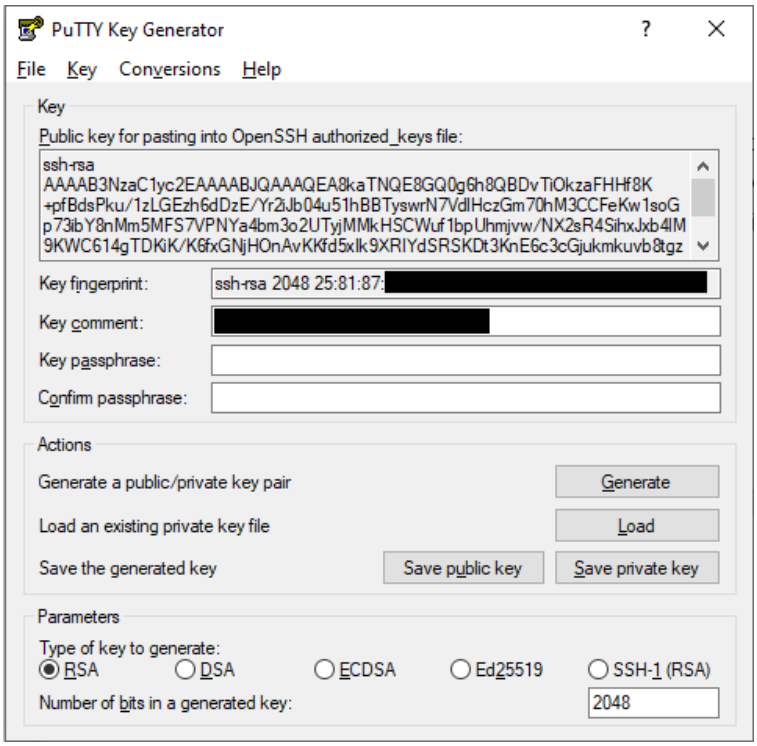

Figure 8. PuTTYgen window.

Finally, open PuTTY and configure as below to create a new session to the instance:

- Session category: Type instance's external IP address into the *Host Name* textbox, ensure that *port* number is 22 and *connection type* is SSH.
- Connection > Data category: Copy the text of *Key comment* box on PuTTYgen to *Auto-login username* box.
- Connection > SSH > Auth category: Browse and select the generated private key file with "ppk" extension for authentication.

Before LinuxGSM and CS:GO server components can be deployed, required dependencies must be installed and updated (including i386 architecture for 32-bit application support, file archiver/compression tools, programming languages and libraries, essential utility tools for LinuxGSM, and Steam Console Client, or SteamCMD to download and update game server files via Steam) with the following command:

sudo dpkg --add-architecture i386; sudo apt update; sudo apt install curl wget file tar bzip2 gzip unzip bsdmainutils python util-linux ca-certificates binutils bc jq tmux netcat lib32gcc1 lib32stdc++6 libsdl2-2.0-0:i386 steamcmd

When dependencies installation and update process has completed, download and install LinuxGSM with this command:

wget -O linuxgsm.sh https://linuxgsm.sh && chmod +x linuxgsm.sh && bash linuxgsm.sh csgoserver

Type *./csgoserver install* on the terminal console to activate the CS:GO server installer script and begin the installation. LinuxGSM will start downloading and installing the CS:GO server and other dependencies into the directory. This process will take approximately 30 minutes.

3.4 Server configuration and automation

After finishing the installation, LinuxGSM will ask for a Game Server Login Token (GSLT). All unofficial CS:GO (and other Source-based games) servers must have such tokens so that Steam can verify the servers and allow connection from users. Each GSLT is unique. Although a Steam account can register multiple tokens, each token can only be used on a single server. GSLTs are created and obtained on Steam Game Server Account Management page (Valve 2015). Type “730” on App ID textbox (each game/application on Steam includes an unique App ID, CS:GO’s App ID is 730) to create a token, then copy and paste it to the terminal.

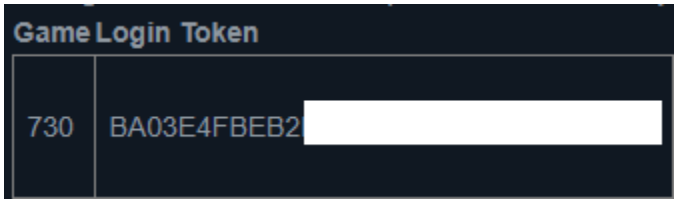

Figure 9. Generated GSLT.

LinuxGSM has multiple config files in its own directory to set up the settings for itself, as well as for its game server instances. In this case all config files are located in */lsgm/config-lgsm/csgoserver* directory, including:

- *_default.cfg*: As the name implies, this file can be seen as a template config with all available default settings. Editing it is not recommended because *_default.cfg* always loads first when LinuxGSM starts, some of its settings will be overwritten when additional config files are loaded after that.
- *common.cfg*: This file stores common settings for all game server instances within a LinuxGSM directory, therefore same settings don't have to be configured multiple times. Nevertheless, this project has only one game server, so no changes will be made for *common.cfg*.
- *csgoserver.cfg*: The general name of this file is *instance.cfg* but it changed based on the name of the directory (*csgoserver*). It is used to modify the settings of CS:GO server, because all configurations in each *instance.cfg* can only apply to a specific game server instance.

When LinuxGSM starts, it will load those configs in the following orders: *_default.cfg -> common.cfg -> csgoserver.cfg*. Any setting configured in *csgoserver.cfg* will overwrite that setting in *common.cfg* which in sequence will also overwrite the setting in *_default.cfg*. The optimal way is copying settings that need to be changed from *_default.cfg* to *csgoserver.cfg* and beginning the modification.

Although all configs file in the server instance can be edited with PuTTY terminal's CLI, it is not ideal. FileZilla supports not only file transfer but also file editor via Notepad, it can save plenty of time for configuration process. On FileZilla window, navigate to File > Site Manager, create a new site with information as below:

- Protocol: SFTP – SSH File Transfer Protocol.
- Host: GCE instance's external IP address.
- Port: 22.
- Logon type: Key file.
- User: Username generated from PuTTYgen earlier (in "Key comment" section).
- Key file: Browse to the folder containing the private key and the select the file with "ppk" extension.

After establishing the session, CS:GO server directory will appear under the "Remote site" section. Navigate to */lsgm/config-lgsm/csgoserver* in the directory to confirm that all config files have been created:

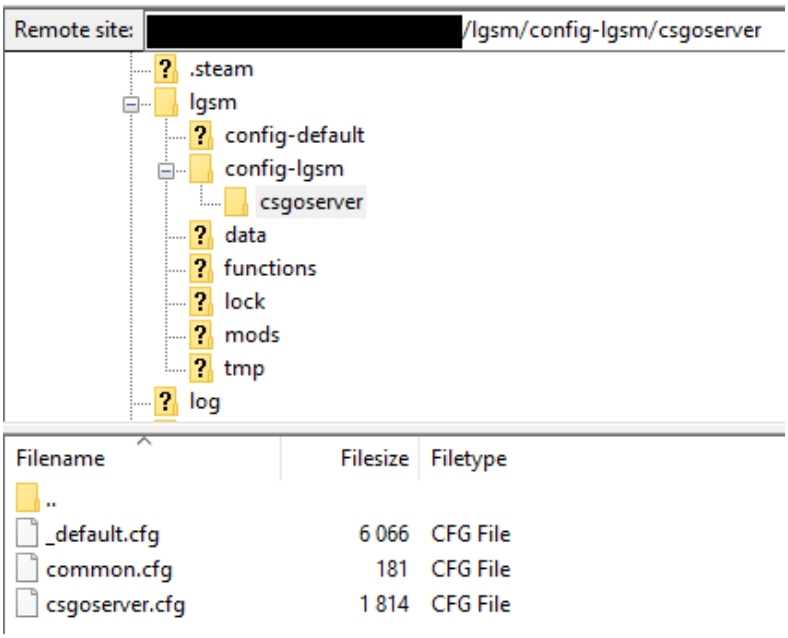

Figure 10. Config files on CS:GO server directory.

Right-click on *_default.cfg* and select “View/Edit”, a new Notepad window will open and show all lines of configuration in the file, open *csgoserver.cfg* with the same method. Copy only the settings that required changes to overwrite the default template and start editing as figure 11 (Edited/added settings are in red borders):

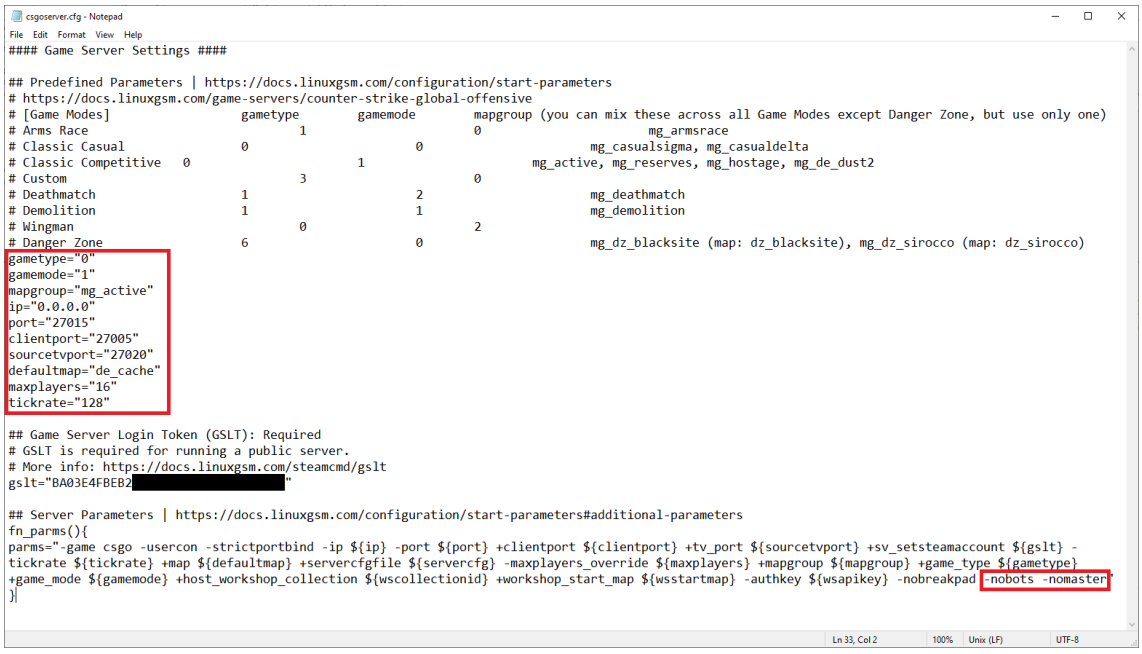

Figure 11. *csgoserver.cfg* configuration for the server.

Here are the explanations for each setting in red borders:

- Game type 0, game mode 1 is Classic Competitive mode.
- Map group “mg_active” is the Active Duty Map Pool (Dust2, Inferno, Mirage, ...).
- Default IP route to allow all connections to the server.
- TCP and UDP ports 27015 are used by server to connect to Steam.
- UDP port 27005 are used by server to handle traffic with clients.
- UDP port 27020 are used by server to provide connection for observers.
- Cache is selected as default map when the server starts/restarts.
- Maximum players can join the server are 16 (observers are not included).
- Increase server’s tickrate to 128 (default tickrate is 64).
- “nobots” command: Remove all bots from the server.
- “nomaster” command: Hide the server from public server browser, only people having the server’s IP address can join.

The next step is modifying the main configuration file of the game which is also named *csgoserver.cfg*. However, it is read by CS:GO itself instead of LinuxGSM and is located at */serverfiles/csgo/cfg*, among other configuration files used by the game. Open the file with Notepad via FileZilla and add the remote console password, extra 128-tickrate configurations between servers and clients, auto-kick removal and setup for observers:

```
// RCON - remote console password.
rcon_password "[redacted]" // Use this password to change settings directly in-game

// Server password - for private servers.
sv_password ""

// Config for 128 tickrate
cl_cmdrate 128         // Client will send 128 data packets/second to server
cl_updaterate 128      // Server will send 128 data packets/second to client
sv_mincmdrate 128      // Set server's minimum tickrate to 128
sv_minupdaterate 128   // Set server's minimum update rate to 128

// Disable auto kick
mp_autokick 0          // Player won't be kicked automatically after 3 friendly fire kills

// GOTV config
tv_enable 1            // Enable GOTV
tv_autorecord 0        // Disable auto-record demo
tv_title "[redacted]"  // Title of GOTV
tv_name "[redacted]"   // Name of GOTV
tv_maxclients 6        // Maximum observers can join GOTV at once are 6
tv_maxrate 0           // Limit bandwidth for GOTV observers, max=0
tv_port 27035          // Dedicated port 27035 for observers when joining GOTV
tv_snapshotrate 128    // Set tickrate for GOTV observers to 128
```

Figure 12. *csgoserver.cfg* configuration for the game.

At this point, the server is playable but it is still required manual commands from server owner to start a match. Third-party mods are able to deal with this issue and provide a complete, automated setup for the match. In order to run desired mods on the server, two addons (plugins) must be installed beforehand:

- Metamod:Source (or Metamod): Is a free, open-source C++ plugin environment for Source engine games. It acts as a "metamod" which sits in between the Game and the Engine, and allows plugins to intercept calls that flow between. It provides a mechanism called SourceHook, a very powerful library for intercepting, overridding, and superseding virtual function calls (commands). (AlliedModders 2005). The first version of Metamod – 1.0 – was released on May 6, 2005. Today Metamod is having two different builds: Stable and Dev. This server will use the latest version 1.11 from Stable build 1145 (compiled on July 11, 2021) to guarantee the stability.
- SourceMod: Is a Metamod plugin (dependency) for Source engine games. It provides comprehensive scripting for the Source engine and mods written using the Source SDK. It has features for administration systems, commands, console variables, events, network messages, timed actions, math and string routines, entity modification, and more. It also features a safely versioned, object oriented API usable from "extensions" written in C++. The extension API can be used to add scripting language callbacks and native features. (AlliedModders 2006) SourceMod was officially announced on October 7, 2004. However, after many difficulties and interruptions during development, the first build was finally available to the public on March 16, 2007. Similar to Metamod, Sourcemod is also having two different builds nowadays: Stable and Dev. To ensure the consistency with Metamod, the server will use the latest version 1.10 from Stable build 6510 (compiled on July 11, 2021).

Both latest versions of Metamod and SourceMod can be installed directly through LinuxGSM by using the command: *./csgoserver mods-install.* Type in *metamodsource* and *sourcemod* to install Metamod and SourceMod respectively. In order to have all permissions and be able to run any command on SourceMod, identity of the server owner must be added to admin list. Navigate to *serverfiles/csgo/addons/sourcemod/configs* and open *admins_simple.ini* file to edit. SourceMod requires three settings for each admin: identity, permissions and password (password is optional). Identity can be a SteamID, name or an IP address. In this case SteamID will be selected because it is unique and

fixed, while name or IP address can be changed frequently. SteamID can be obtained on a website named Steam ID Finder (Network N Ltd 2021) by analysing the URL of Steam account. All SteamIDs follow the same format: STEAM_X:Y:Z, where X, Y and Z are integers. At the end of *admins_simple.ini*, insert *"<SteamID>" "z"* (z stands for all permissions), then save and close the file. With this method, admin can open an in-game menu to quickly run multiple general commands related to players and server, rather than accessing the console with a remote password and typing commands one by one.

The next step is installing a mod called "CS:GO PUG setup". PUG stands for "pick-up game", means that random people can join the server, play the match with other random people, or leave at any time without receiving penalties, unlike Valve official servers where players who abandon an ongoing CS:GO match will be prevented from matchmaking in a fixed duration. CS:GO PUG setup was orginally developed by Sean Lewis (GitHub username: splewis), he rolled out the first stable version 1.0.0 of the mod on July 14, 2014. Latest version 2.0.5 was released by Sean on August 6, 2018 with the help of more than 20 contributors. The main goal of CS:GO PUG setup mod is to provide proper in-game settings to start the match automatically without server administrators, while still making improvements compare to official matchmaking setup, such as:

- Unlimited warmup time: A competitive match on official server gives only 5 minutes for both teams to join and practice before the match begins. PUG setup mod removes the time limit, match will only starts when all players on both teams are ready (by typing *.ready* command in the chat).
- Unlimited timeout/pause time: With competitive match on official server, each timeout will last only 1 minute and the match resumes immediately. PUG setup also removes the time limit for all timeout, match will continue when and only when both teams agree to unpause it.
- Extra round to choose starting side: In a competitive match on official server, players who are on CT side during the warmup are forced to start and continue playing on that side until first half is over (after round 15). PUG setup provides a "knife" round where players can only use melee weapon to eliminate all opponents right after the warmup. Team won that round can decide the starting side for the match.
- Overtime: All competitive matches on official servers don't have overtime, they will stop at 15-15 score when round 30 – final round of the match – ends, results

in a draw. PUG setup enables this function with 6 rounds per overtime to determine the winner. Whoever gets 4/6 rounds of overtime will win the match.

- Demo files: Each demo from official server always has long and complex name in its file. With PUG setup, server administrators can define name of all recorded demo files in simple and straightforward format. Because recording function is not mandatory on PUG setup, this feature will be disabled to save resources and bandwidth for the server.
- Various modes for players: In a competitive match on Valve official server, teammates and opponents of one or more players in a party are chosen randomly to fill two teams of 5 people (except 5 players in the same party, they will always play together on the same team). PUG setup provides extra options such as: "Captains" mode where two captains (one from each team) choose from the pool of remaining 8 players to their lineup in turn; "manual" mode where all 10 players can join the team manually; or "random" mode which is similar to official server's method. This private server will always use manual mode to maintain the freedom for all players. In addition, server administrators are able to switch the match to 1-on-1; 2-on-2; 3-on-3; 4-on-4 alongside with classic 5-on-5 format in case there are less than 10 players in the server through PUG setup.
- Extra addon plugins: Some addons include in PUG setup file package like: *teamlocker* (block extra players from joining team when match is live), *chatmoney* (print teammates' current money to the chat at the beginning of each round) and *damageprinter* (print damage done/taken from other players when round is over) will be used in the private server to bring better experience for all players during the match.

To install PUG setup mod to the server, use command *./csgoserver mods-install* on PuTTY terminal window and type *pug*. After the installation is completed, open FileZilla and navigate to */serverfiles/csgo/cfg/sourcemod/pugsetup*, open *pugsetup.cfg* and start editing. Although there are over 30 settings in the file, only some of them will be configured as below:

- *sm_pugsetup_autosetup*: Change value from *"0"* to *"1"* to allow a PUG to be configured automatically using the default setup options.
- *sm_pugsetup_exclude_spectators*: Change value from *"0"* to *"1"* to exclude observers from ready-up counts and prevent team captains from picking observers to the team.

- *sm_pugsetup_quick_restarts*: Change value from *"0"* to *"1"* to let the match do a single restart instead of 3 times before being live.
- *sm_pugsetup_randomize_maps:* Change value from *"1"* to *"0"* so that the order of map during vote/veto is not randomised.

The next step is modifying content of files in PUG configuration directory to have suitable setup options, permissions and map list for the players. Files are located in *serverfiles/csgo/addons/sourcemod/configs/pugsetup*. First file which needs to be changed is *setupoptions.cfg*. As the name implies, it can be used to modify the values of all options and decide whether the option will appear on the in-game menu. Here are the options to be displayed on the menu and their values:

- Map type: vote. All players will vote to choose the map they want to play.
- Team type: manual. All players can join either one of two teams manually.
- Autolive: enabled. Match will start automatically when all players are ready.
- Knife round: enabled. A "knife" round will occur after the warmup and before the match begins.
- Team size: 5. All matches will be played in 5-on-5 format.

The second file to modify is *permissions.cfg*. Server administrators can use this file to assign the corresponding permission for each command. There are 5 allowed permissions: *"all"*, *"captain"*, *"leader"*, *"admin"* and *"none"* with the order from high to low: admin -> leader -> captain -> all. A command will be disabled when it has *"none"* permission. On this server, only administrators can adjust setup options, command aliases and map pool; force start or end the match promptly. Chosen leaders and captains are able to pick players to their team or use command to let the system pick a player randomly. Everybody can operate commands to pause/unpause the match or view the map pool, players whose team won the "knife" round can decide to keep or change starting side (CT/T).

maps.txt is the last file which requires modification in this directory. In this file, all maps shown on the list can be voted by players. Map with highest votes will be selected for the match. Based on players' feedback, maps can be voted on this server are (all are Bomb Defusal maps): Ancient (*de_ancient*), Cache (*de_cache*), Dust 2 (*de_dust2*), Inferno (*de_inferno*), Mirage (*de_mirage*), Nuke (*de_nuke*), Overpass (*de_overpass*), Train (*de_train*), and Vertigo (*de_vertigo*).

In order to make sure that the server will always run smoothly and be up to date, a utility called “cron” will be used to schedule necessary tasks to run regularly at specific moments. cron (or “cron job”) runs the scheduled commands put in a cron table (“crontab”). This table can be edited on the terminal with root privilege by command: *sudo crontab -e*. In the file, add these commands to schedule 3 tasks for the server:

- *@hourly su - <username> -c '/home/<username>/csgoserver update' > /dev/null 2>&1*: Checks for new update of CS:GO and installs it if possible every hour at xx:00.
- *0 0 * * * su - <username> -c '/home/<username>/csgoserver restart' > /dev/null 2>&1*: Restarts CS:GO server daily at 00:00 UTC to avoid memory overflow which crashes entire system.
- *@reboot su - <username> -c '/home/<username>/csgoserver start' > /dev/null 2>&1*: Starts LinuxGSM and CS:GO server automatically after system reboot.

In the commands, username can be found in the private key, “*> /dev/null 2>&1*” means that any output of the command, even error will always be discarded.

Finally, run *./csgoserver restart* command on PuTTY terminal to apply all changes to the server.

3.5 Verification and testing

Because this is a private server, it can only be joined via the in-game console. In CS:GO and other Source engine games, Valve implemented an UNIX-terminal-like CLI directly in the game called “console”. Although console is mainly designed for developers as a debugger and monitoring tool, players can utilize it to access advanced configuration which does not exist in the game settings. By default, console is disabled. In CS:GO, there are two ways to enable the in-game console:

- On Steam client, right-click on CS:GO located on the left column, choose *Properties*, a new window will appear, on the *GENERAL* section, add “*-console”* under the *LAUNCH OPTIONS* textbox.

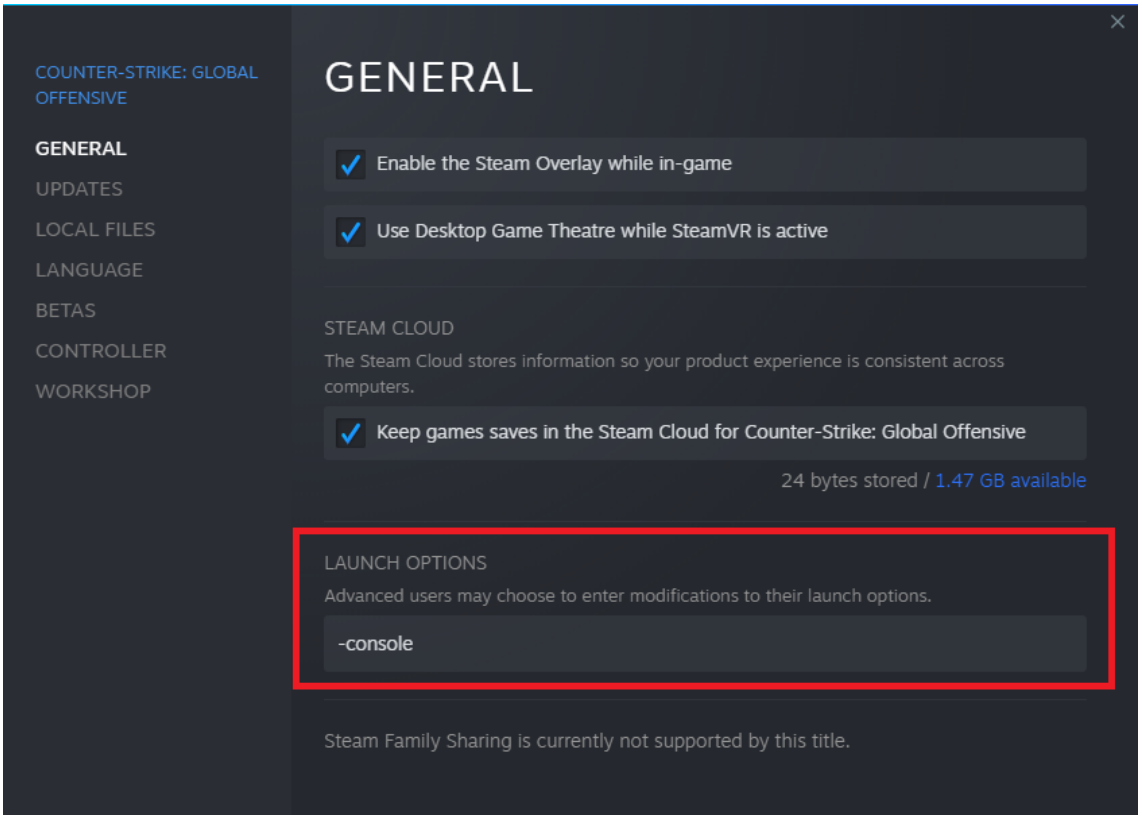

Figure 13. Enabling console via Launch Options.

– Open CS:GO via Steam, navigate to Settings menu on the left column (the one with cogwheel icon), select *Game*, look for *Enable Developer Console* and change the settings to "*YES*".

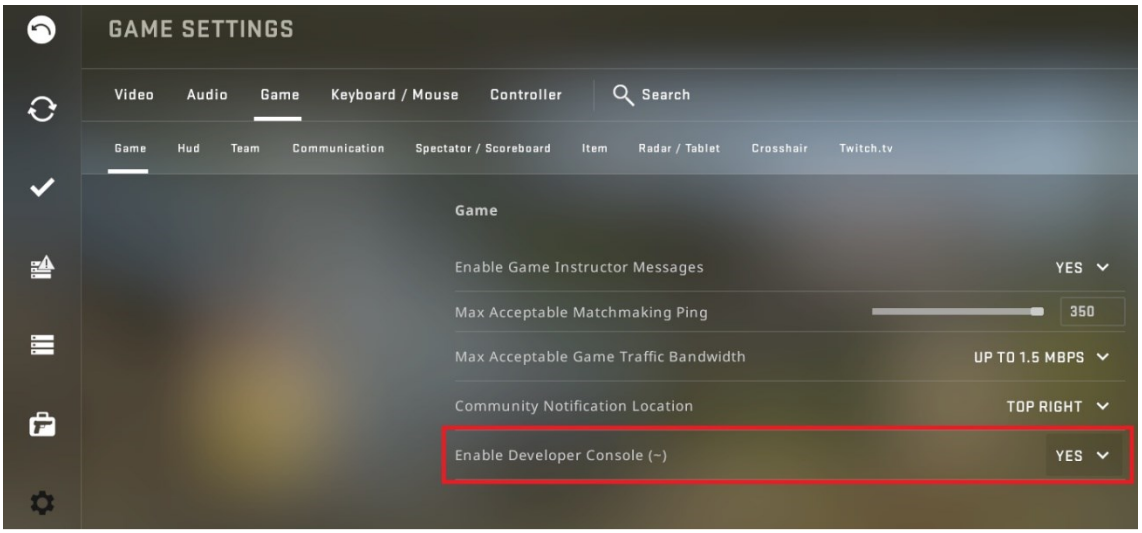

Figure 14. Enabling console in the game.

The default hotkey to open console in the game is “~” (tilde), which is under the Escape (ESC) button and on top of TAB button. However, that button in Finnish keyboard layout is “§” (sector sign), not “~”, so it cannot open in-game console. To fix this issue, select *Keyboard/Mouse* settings in CS:GO and bind “§” button to *Toggle Console*, although it still shows the button is “\” (backslash), it actually is “§”. Open console window with the new-bound button, type *connect <private server’s IP address>* and wait for CS:GO to load the map (depends on the hardware of client, the loading time may vary between 20-40 seconds). The team selection screen shows up, therefore it means that the connection is successful.

LinuxGSM also provide a “console” feature so that administrators can access CS:GO server console without opening the game through the command *./csgoserver* console. To check whether Metamod and SourceMod are running and their current versions, use two commands: *meta version* and *sm version* on the console. The output should show as figure 15:

```
meta version
 Metamod:Source Version Information
    Metamod:Source version 1.11.0-dev+1145
    Plugin interface version: 16:14
    SourceHook version: 5:5
    Loaded As: Valve Server Plugin
    Compiled on: Jul 11 2021 22:32:15
    Built from: https://github.com/alliedmodders/metamod-source/commit/0bb53f2
    Build ID: 1145:0bb53f2
    http://www.metamodsource.net/
sm version
 SourceMod Version Information:
    SourceMod Version: 1.10.0.6510
    SourcePawn Engine: 1.10.0.6510, jit-x86 (build 1.10.0.6510)
    SourcePawn API: v1 = 5, v2 = 12
    Compiled on: Jul 11 2021 23:24:12
    Built from: https://github.com/alliedmodders/sourcemod/commit/0b468f2
    Build ID: 6510:0b468f2
    http://www.sourcemod.net/
```

Figure 15. Metamod and SourceMod details.

Since SourceMod is up and running, admin commands are also available to execute. To see the command menu, open the text chat (default hotkey is U for team chat and Y for all chat) and type *!admin*, the menu will appear on the left of the screen with functions for players (kick, ban, rename, ...), servers (change map, load configuration files, refresh admins list) or voting (vote for map, vote to kick or ban a player, ...)

Figure 16. SourceMod admin menu.

To confirm that PUG setup has been installed successfully and its setup options match the configuration in *setupoptions.cfg*, open the text chat and type *.setup*, the menu will appear on the left of the screen. Keep in mind that only one menu can be shown for interaction at a time.

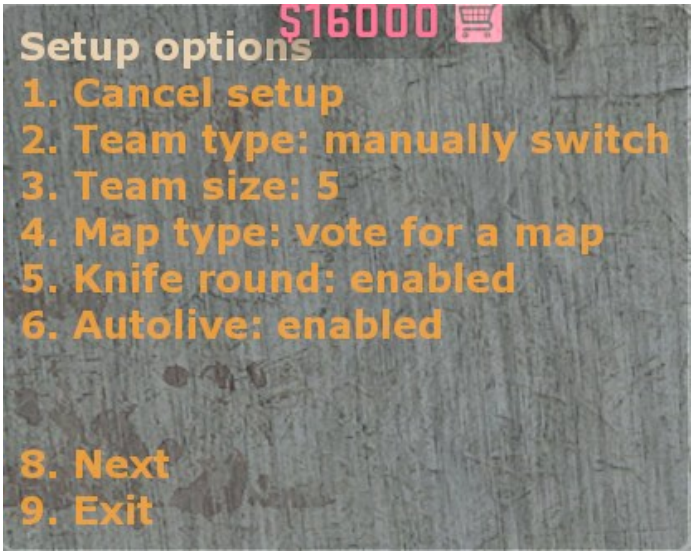

Figure 17. PUG setup options.

Server verification is completed, players are allowed to join with the same method as above (connect via in-game console). They can vote to select the map, start the match by themselves when everyone is ready because the game is running all setups automatically now.

Figure 18 demonstrates the performance results of a 5-on-5 match on Inferno (with 1 spectator) which last over an hour:

Figure 18. LinuxGSM's CSGO server performance on GCE N1 instance.

Overall, the server remains very stable, there are no signs of input lag or interruption on both sides. Pings of all players fluctuate between 40 and 50, which is acceptable and totally playable. High tick rate brings significant effect in terms of experience during the match, server responses quicker to all actions made by players, hence smoother gameplay. Group of players who wanted a private CS:GO server is very happy because they are having one to join any time, anywhere. Besides, spectators are also satisfied with the intuitive control to watch any player in the match, thanks to the dedicated configurations for observers on the server. All scheduled jobs operate as planned to prevent memory overflow on the instance, to update the game whenever a new patch is out and to start LinuxGSM services immediately after system reboot.

4 CONCLUSION

The objective of this thesis was to gain knowledge about cloud computing and GCP, to take advantage of that knowledge to deploy an always-online private game server and to monitor the outcome. Throughout the process, deeper understanding about VMs' technology on Google Cloud has been achieved. The thesis has pointed out the highlights of storage solutions, network, and security measures within GCP that can attract people to choose cloud servers instead of traditional on-premises ones. All pricing options offered by Google were examined carefully to end up with the best choice for budget while still meeting the requirements. Knowledge about network, system administration and the game (CS:GO) is very helpful for the entire implementation progress. During the practical part, an GCE instance was initiated, the game server was configured successfully with the assistance of various tools and software. After finishing the setup, the connection to the server and all configurations were verified, performance data during the match was gathered along with feedback from players who played on the server.

Even though the project has been completed, there are many ideas for improving the server in the future, for example: allowing the server to record all matches into demos and letting players download them via a file share; selecting a GCE instance with higher resources to setup multiple 5-on-5 sessions; trying different game modes and community maps on the server; configuring the server to send notifications and alerts to email when it goes down; or deploying a server for different game in the same instance.

In conclusion, the thesis has proven that self-deploying a game server on cloud platform can be a better option, compared to existing services for online private sessions among players. Cloud servers showed the ability to bring outstanding performance through the Internet to consumers in terms of connection, resources, and security at reasonable prices. Cloud servers are gradually replacing traditional on-premises infrastructure, especially during the pandemic when many people are working from home. Without a doubt, servers on the cloud are becoming a more popular choice, not only for business, but also for individuals.

REFERENCES

ALLIEDMODDERS, Jan 12, 2006-last update, SourceMod - AlliedModders Wiki . Available: https://wiki.alliedmods.net/SourceMod [June 20, 2021].

ALLIEDMODDERS, 2005-last update, Metamod:Source - About . Available: https://www.sourcemm.net/about [June 20, 2021].

FRANKENFIELD, J., Jul 28, 2020-last update, Cloud Computing Definition . Available: https://www.investopedia.com/terms/c/cloud-computing.asp [Feb 13, 2021].

GIBBS, D., 2020. *LinuxGSM_.*

GOOGLE, 2021a-last update, Accelerator-optimized machine famliy | Compute Engine Documentation . Available: https://cloud.google.com/compute/docs/accelerator-optimized-machines [Mar 5, 2021].

GOOGLE, 2021b-last update, Compute-optimized machine family | Compute Engine Documentation . Available: https://cloud.google.com/compute/docs/compute-optimized-machines [Mar 5, 2021].

GOOGLE, 2021c-last update, General-purpose machine family | Compute Engine Documentation . Available: https://cloud.google.com/compute/docs/general-purpose-machines [Mar 5, 2021].

GOOGLE, 2021d-last update, Memory-optimized machine family | Compute Engine Documentation . Available: https://cloud.google.com/compute/docs/memory-optimized-machines [Mar 5, 2021].

HUNTER, T. and PORTER, S., 2018. *Google Cloud Platform for Developers : Build Highly Scalable Cloud Solutions with the Power of Google Cloud Platform.* Birmingham: Packt Publishing, Limited.

KOSSE, T., 2021. *FileZilla Client* .

NETWORK N LTD, 2021-last update, Steam ID Finder . Available: https://www.steamidfinder.com [Jun 25, 2021].

PANETTIERI, J., Nov 3, 2020-last update, Cloud Market Share 2020: Amazon AWS, Microsoft Azure, Google, IBM. Available: https://www.channele2e.com/channel-partners/csps/cloud-market-share-2020-amazon-aws-microsoft-azure-google-ibm/ [Feb 25, 2021].

RAVI, J., RAJ, J. and SRINIVASAN, V., 2018. *Google Cloud Platform for Architects : Design and Manage Powerful Cloud Solutions.* Birmingham: Packt Publishing, Limited.

SATTIRAJU, N., Apr 2, 2020-last update, The Secret Cost of Google's Data Centers: Billions of Gallons of Water to Cool Servers. Available: https://time.com/5814276/google-data-centers-water/ [Feb 21, 2021].

TATHAM, S., 2021. *PuTTY.*

VALVE, 2021. *CS:GO cover art* . Wikipedia.

VALVE, 2015-last update, Steam Game Server Account Management . Available: https://steamcommunity.com/dev/managegameservers [June 15, 2021].

VINCENT, B., Feb 7, 2021-last update, Steam Hits New Concurrent Player Record, Toppling January's Numbers. Available: https://uk.pcmag.com/pc-games/131630/steam-hits-new-concurrent-player-record-toppling-januarys-numbers/ [Feb 13, 2021].