COMPUTER SYSTEMS AND NETWORKS

SYSTEMS AND NETWORK ENGINEERING

Real-time Distributed Network Application with Security Information and Event Management

Author:
Hussein Al-Khalissi
Emeka Nzeopara
Ayomide Shoyemi
Ammar Alnajim

Instructor:
Paolo Ciancarini

January 23, 2023

Contents

List of Figures

1 Introduction

For the final project of Computer Systems and Networks course, we wanted to deploy a real-time communication software of Client-Server architecture. Ideally, this software should be capable of handling large volumes of transactions, configured in a distributed systems fashion, and able to scale both horizontally and vertically.

We have chosen to deploy a Massively Multiplayer Online Game, commonly referred to as MMO, which is an online game played by thousands of players at once. These games are sophisticated affairs, with 3D visuals and an expansive virtual environment in which players may collaborate in a wide variety of ways [3]. Over the past few years, Massive Multiplayer Online Games have rapidly emerged as a topic of interest for research among academics [1]. Concerning the many problems that these games provide, a significant number of essays have been written and distributed. Frameworks exist to aid indie developers and small teams in designing, implementing and configuring MMOs [2]. In this report, we will go over the technologies stack and frameworks we have chosen for our implementation, how we scaled it, and how we secured it.

1.1 Aims and Objectives

- Deploy an MMO Game Framework
- Scale backend architecture in a distributed way
- Implement and configure a SIEM solution

As a starter, we picked uMMORPG Framework for Unity 3D game engine. This framework will give us a boilerplate baseline code, animations, 3D models, and a virtual environment to jumpstart our project. We have considered using ARQ as the communication transport protocol over single links which is thought to have a better Round Trip Time than TCP handshakes, windowing, flow control, and acknowledgments. ARQ will offer us the ability to retransmit lost packets more rapidly than the TCP transport protocol. In addition to the rapid retransmission, ARQ operates on layer 2 data units (frames) as the header size is smaller in size than layer 3 data units (packets). This will improve the efficiency of the gaming experience. KCP is a speedy and stable protocol that is implemented as an application-level enhancement. It may reduce average latency by 30–40% and maximum delay by 3–4X, at the expense of 10–20% more wasted bandwidth compared to TCP. Due to its pure algorithm implementation [4].

Rather of handling the transmission and reception of the underlying protocol, KCP requires its users to select a transmission mode for the underlying data packet and submit it to it through callback (such as UDP). You can't use the time for anything if there aren't any incoming system calls. TCP prioritizes throughput over other factors, such as delay, for the purpose of using bandwidth effectively (the maximum number of bytes that may be delivered per second). Since KCP is geared for throughput (how quickly data can go from one end to the other), it uses 10% to 20% more bandwidth than TCP but reduces transfer speed by 30% to 40%. In contrast to the KCP channel's low throughput and high flow rate, the TCP channel's moderate throughput and high flow rate is a significant advantage. In addition to its standard pace, KCP also offers a faster "fast" option [4], which uses the following techniques to maximize throughput:

- **RTO Doubled vs Not Doubled**
  Turning on KCP fast mode reduces the TCP timeout calculation from RTOx2 to RTOx1.5, which is a substantial gain over the prior value of x2 [4].
- **Selective Retransmission vs Full Retransmission**
  KCP only retransmits the data that was lost, as opposed to TCP's approach of retransmitting all packets [4].
- **Fast Retransmission**
  The sending endpoint sends packets 1, 2, 3, 4, and 5, and then receives the remote acknowledgment (ACK) for each of these; however, when KCP receives ACK3, it knows that packet 2 is skipped once, and when it receives ACK4, it knows that packet 2 is skipped twice; thus, it can conclude that packet 2 is lost and, instead of waiting for the timeout to expire, it will immediately retransmit packet 2, greatly increasing the transmission rate when packet loss occurs [4].
- **Delayed ACK vs Non-delayed ACK**
  To maximize throughput, TCP waits to deliver an ACK (although NODELAY fails), meaning a relatively high RTT is used in the timeout calculation and, in turn, a longer judgment process when packet loss happens. KCP, on the other hand, allows users to customize how long they wait before delivering an ACK [4].

- **UNA vs ACK+UNA**
  UNA (All packets before this number received, like TCP) and ACK are the two possible types of replies in the ARQ paradigm (The packet with this number received). With the past, protocols have had to choose between using UNA or using ACK, as using UNA alone would lead to complete retransmissions and using ACK alone would have too high a penalty for packet loss. However, in the KCP protocol, all packets carry UNA information save for one ACK packet [4].

- **Non-concessional Flow Control**
  When operating in normal mode, KCP follows the same fair concession principles as TCP, which means that the size of the send window is based on four parameters: the transmit cache size, the size of the receive buffer on the receiving end, the packet loss concession, and the slow start. There is a trade-off between fairness and bandwidth consumption and the impact of smooth transmission even when BT is opened, therefore in the case of transferring little data with a high timeliness demand, the option to bypass the final two phases through setting is available [4].

2 Game Development

2.1 Unity 3D Engine

We proceed and create a new 3D project with no required dependencies in Unity 3D engine using a stable release version 2021.3.11f1. We had two choices with uMMORPG frameowork, either 2D or 3D version. Because we wanted to make this project fit to the course requirements of CSN, scale it to multiple instances, and distributed the workloads across the server instances we went with the 3D version. In the below figure, we start modifying the player prefab scripts only and no need to adjust any 3D models, animations, sounds effects, etc.

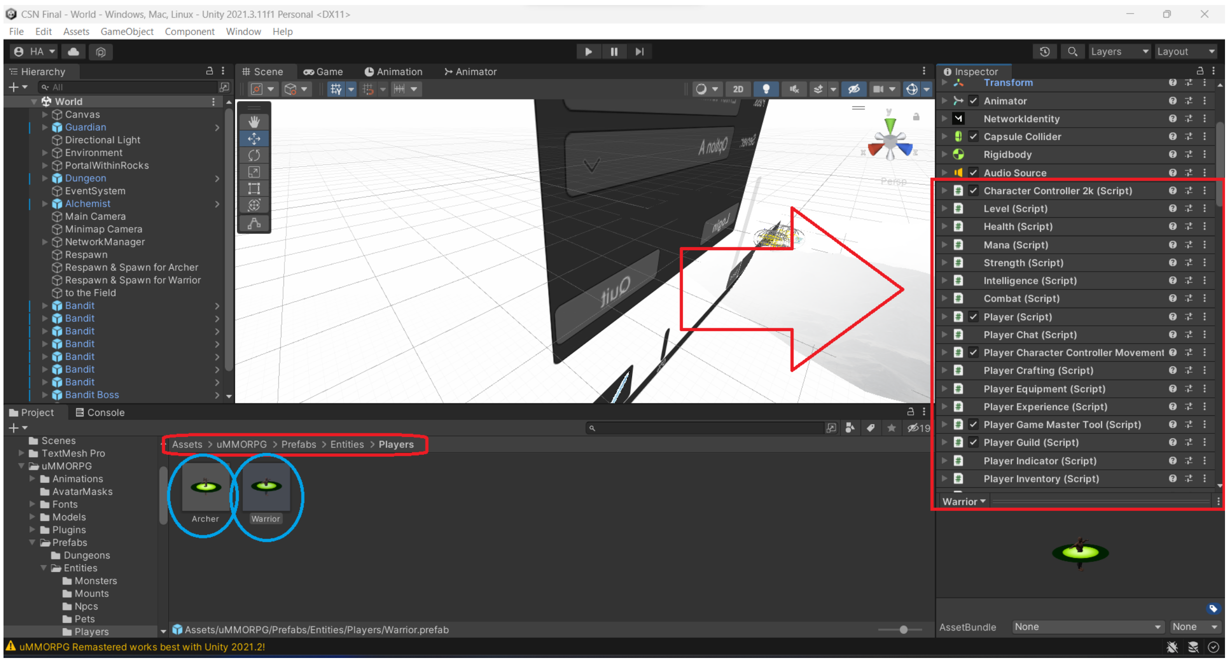


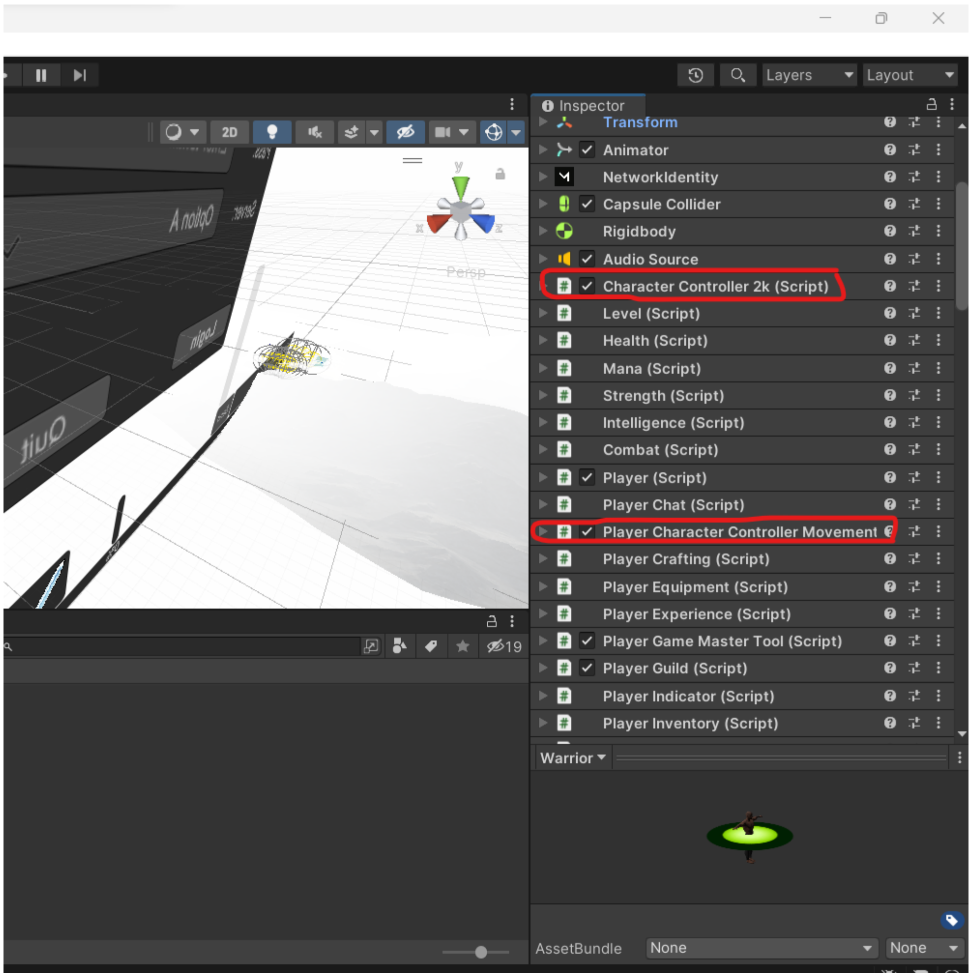
Figure 1: Prefabs/Entities/Players

2.2 uMMORGP Framework

These player prefabs all share these important scripts components that the developer could adjust and change the MMO settings. An example of the components we had to adjust is (Player Trading, Player Party, Item Mall, Experience, Chat, and Level) you can modify a whole lot of properties and headers without diving too deep into the game codebase. Simply download the codebase from github and try adjusting it to your needs.

An important part of the player prefabs that we will have to sync with the servers instances is the "Player Movement" component. This component comes precomposed of two main components that will sync using

RPC calls with the server: - CharacterControllerMovement: for physics-based movement. - NavMeshMovement: for point-and-click movement system on the 3D terrain pre-baked Navmesh. This we will use for our deployment as it utilizes very little bandwidth and its code is very simple



Transform
Animator
NetworkIdentity
Capsule Collider
Rigidbody
Audio Source
Character Controller 2k (Script)
Level (Script)
Health (Script)
Mana (Script)
Strength (Script)
Intelligence (Script)
Combat (Script)
Player (Script)
Player Chat (Script)
Player Character Controller Movement
Player Crafting (Script)
Player Equipment (Script)
Player Experience (Script)
Player Game Master Tool (Script)
Player Guild (Script)
Player Indicator (Script)
Player Inventory (Script)

Warrior

AssetBundle None None

Figure 2: Player Movement Components

These prefabs components could be adjusted and fine-tuned to lower the communication bandwidth between client and server instances.

3 Amazon Web Services Deployment

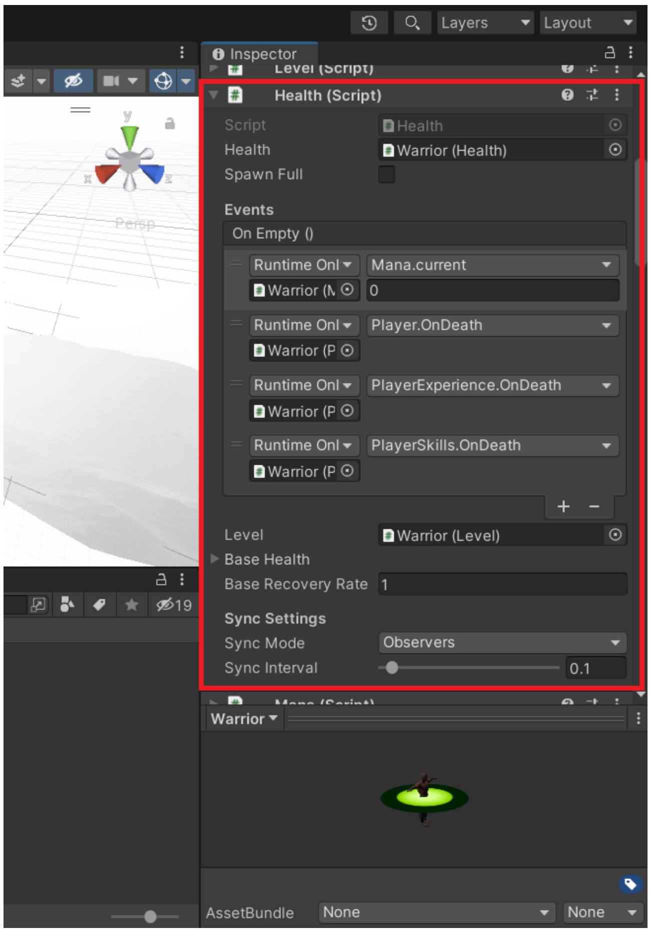
For this project, AWS was used to launch the instances that the game developed will run on. A diagram of this representation is given in the figure below. There are various AWS features that were used to implement this development; these features include:

3.1 Instance Sizes

An instance is a virtual server in the AWS Cloud. With Amazon EC2, you can set up and configure the operating system and applications that run on your instance. The size of the instance of a problem is the size of the representation of the input. The number of instances used for this project was 4 servers each of the servers represents places all the game objects code (players, enemies, quests, items, marketplace, party, guilds, etc) their states are always in sync with the client machines using RPC calls.

3.2 Public and Private Subnets

This configuration includes a public and private segment of the VPC. We use this scenario because we want to have a public-facing server. According to figure 6, this can be found as the server listed below. The instances



Inspector

Health (Script)

Script: Health

Health: Warrior (Health)

Spawn Full

Events

On Empty ()

Runtime Onl | Mana.current
Warrior (N | 0

Runtime Onl | Player.OnDeath
Warrior (P

Runtime Onl | PlayerExperience.OnDeath
Warrior (P

Runtime Onl | PlayerSkills.OnDeath
Warrior (P

Level: Warrior (Level)

Base Health

Base Recovery Rate: 1

Sync Settings

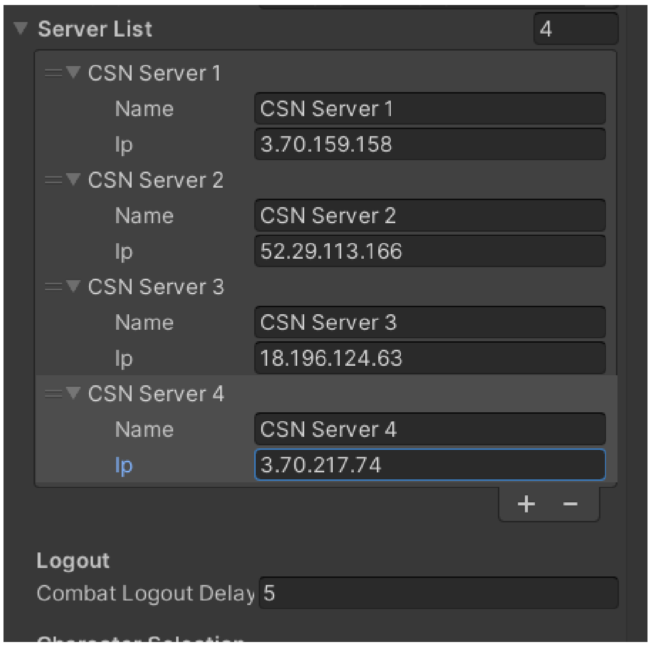
Sync Mode: Observers

Sync Interval: 0.1

Warrior

AssetBundle: None | None

Figure 3: Prefab Component that syncs only the properties



Server List: 4

CSN Server 1
Name: CSN Server 1
Ip: 3.70.159.158

CSN Server 2
Name: CSN Server 2
Ip: 52.29.113.166

CSN Server 3
Name: CSN Server 3
Ip: 18.196.124.63

CSN Server 4
Name: CSN Server 4
Ip: 3.70.217.74

Logout
Combat Logout Delay: 5

Figure 4: Server List Component for EC2 Instances

in the public subnet can send outbound traffic directly to the internet, whereas the instances in the private subnet can't. Instead, the instances in the private subnet can access the internet by using a network address translation (NAT) gateway that resides in the public subnet. The database servers can connect to the internet for software updates using the NAT gateway, but the internet cannot establish connections to the database servers.

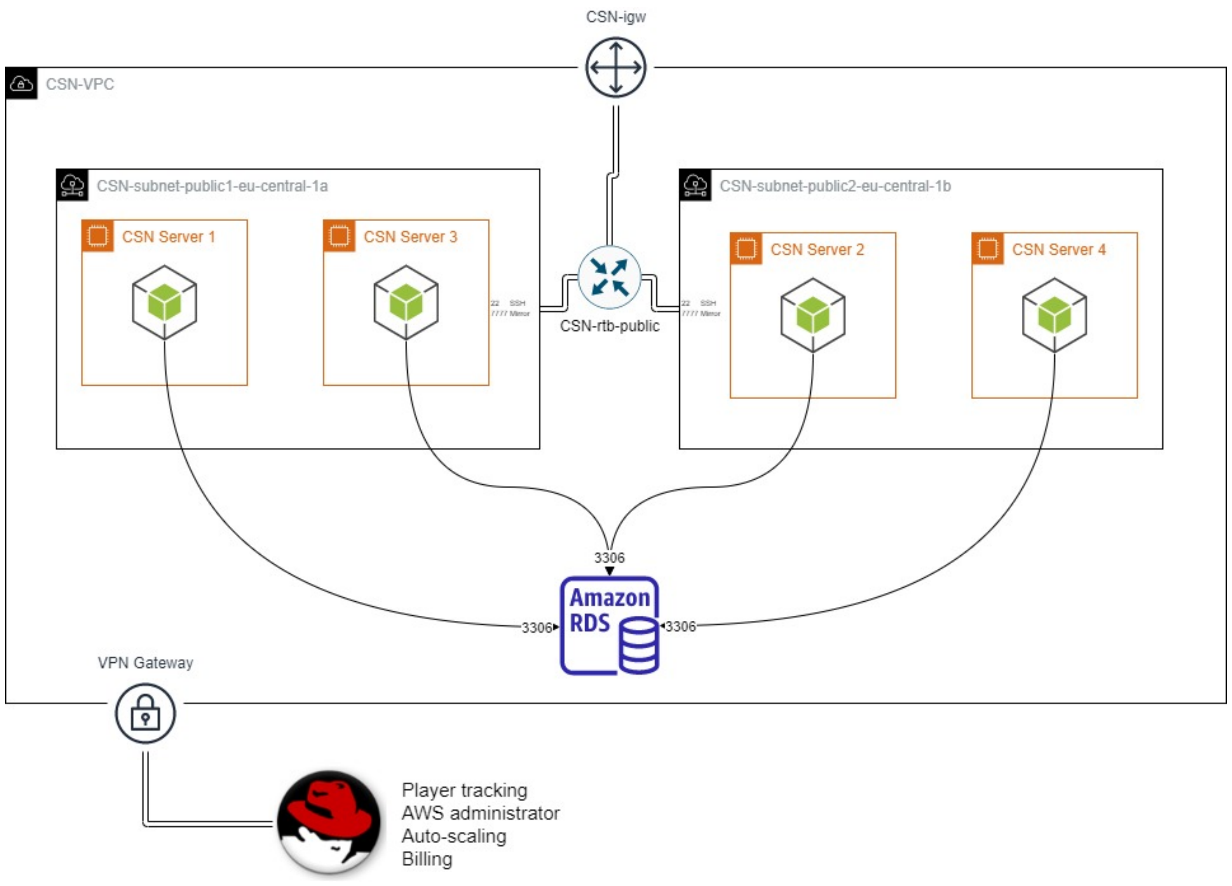


Figure 5: Architecture of the Gaming System

3.3 Routing Between Subnets

In this scenario, all traffic from each subnet that is bound for AWS goes over the internet gateway. The database servers in the private subnet can't receive traffic from the internet directly because they don't have Elastic IP addresses. However, the database servers can send and receive internet traffic through the NAT device in the public subnet.

3.4 Separating Instances

Amazon VPC makes it possible to logically enclave within the Amazon Elastic Cloud Compute (Amazon EC2) network that can house compute and storage resources. This environment can be connected to a customer's existing infrastructure through various means and in our case, it is a virtual private network (VPN) connection over the Internet; this service provides private connectivity into the AWS Cloud. In this case, by using this means we have the flexibility, security, and complete control of the network presence in the cloud. We can control the private environment including IP addresses, subnets, network access control lists, security groups, operating system firewalls, route tables, VPNs, and internet gateways. Amazon VPC provides robust logical isolation of all customer resources, including their access paths to each other and with AWS services.

3.5 Resources in Infrastructure

Amazon RDS provides a set of features to ensure that our data is securely stored and accessed. We ran our database in Amazon Virtual Private Cloud (VPC) for network-level isolation. Use security groups to control what IP addresses or Amazon EC2 instances can connect to your databases. This built-in firewall prevents any database access except through the rules you specify. For the connection of Infrastructural Resources within the VPC below are the things of consideration.

- **EC2 instances and the RDS database** had to be within the same VPC and the security controls that define the connections need to be in place to make this connection automatically happen.
- **EC2 Instances with Adaptive Features** These adaptive features include auto-scaling and load balancing. During the implementation of these, we made sure we were able to follow the conditions of ensuring

the EC2 instance is in a running state, the AMI used to launch the instance still exists, the instance is not a member of another scaling group, the availability zones are defined in the auto-scaling group and finally that the load-balancer is attached to the auto-scaling group

4 Working Principle

4.1 Server Perspective

Considering the working principle and looking at the infrastructure from the server perspective, the game objects' code (players, enemies, quests, items, marketplace, party, guilds, etc) and their states are always in sync with client machines using RPC calls on the server. The client machine connects to the server, retrieves progress if any from DB, then starts syncing every object property with the server. The master state is on the server. So if a client had an issue (latency, packet drops) in updating its state on the server, then the server state is considered since the game is basically running on the server.

4.2 Network Perspective

Considering the network perspective, there are two protocols handling the network aspect. These include Telepathy and KCP. Telepathy is a server library and also a transport protocol in some way while KCP is a custom implementation of application state sync on top of UDP. The KCP has features of TCP such as re-transmitting several packets (u can specify what size) if they are lost without the bulky header of TCP, flow control, sliding windows, etc. As they're not needed in MMOs. So we don't need to agree on a big size window to transmit large traffic at one time because of the RPC. A transport-layer protocol can also provide timing guarantees. As with throughput guarantees, timing guarantees can come in many shapes and forms. This service makes it appealing to interactive real-time applications, such as this multiplayer games, all of which require tight timing constraints on data delivery in order to be effective. In this multiplayer game or virtual interactive environment, a long delay between taking an action and seeing the response from the environment (for example, from another player at the end of an end-to-end connection) makes the application feel less realistic.

5 Security Information and Event Management Solution

Security of the Game server infrastructure was considered hence we decided to incorporate a SIEM in order to monitor We decided to use the Splunk cloud version.

Splunk

Splunk is a distributed system that is widely used for monitoring, searching, analyzing, and visualizing machine-generated data in real-time. Its architecture consists of three major components which are;

- **T**he search head,
- **T**he indexer, and
- **T**he forwarder.

5.1 The search head:

The search head is responsible for providing a user interface interactive platform where users can query the data and much more

5.2 The indexer:

The indexer is responsible for collecting data from the forwarder and turning them to invents, after with it indexes them and stores them for future lookup

5.3 The forwarder:

The forwarder is an agent is responsible for collecting logs and sending them to the indexer. Splunk has two types of forwarders:

- Universal Forwarder
- Heavy Forwarder

In our deployments the forwarders will be resident in the four (4) virtual game servers and the relational database which is located on AWS. To get set up Splunk cloud the following action was performed;
A Splunk Cloud Trial account was created, then we proceeded with configuration on the account and 5GB ingress instance was assigned to us.

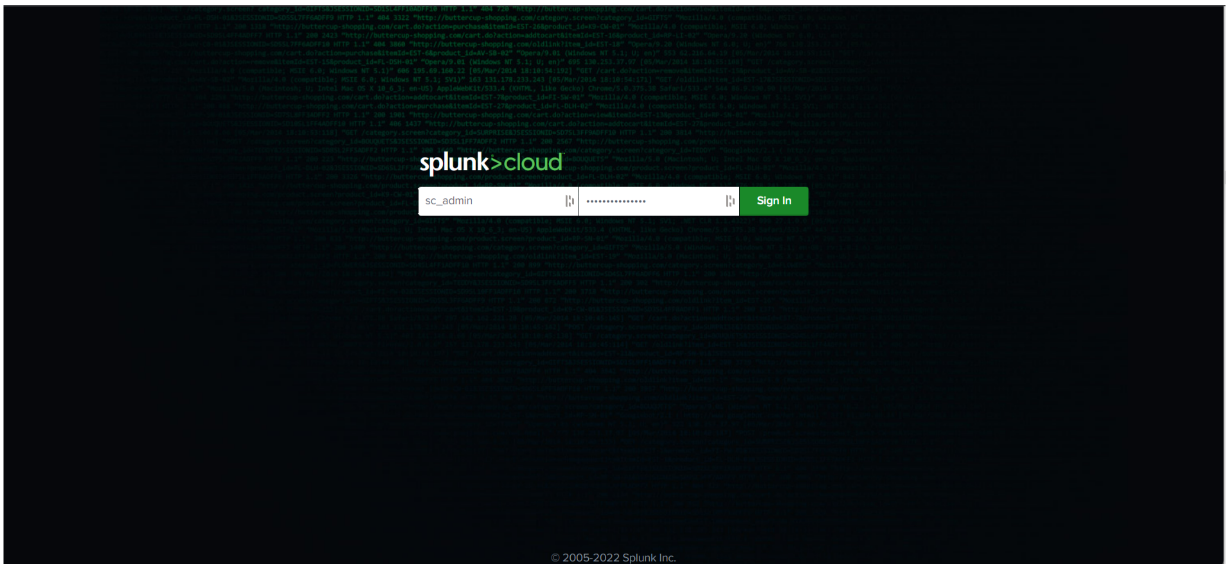
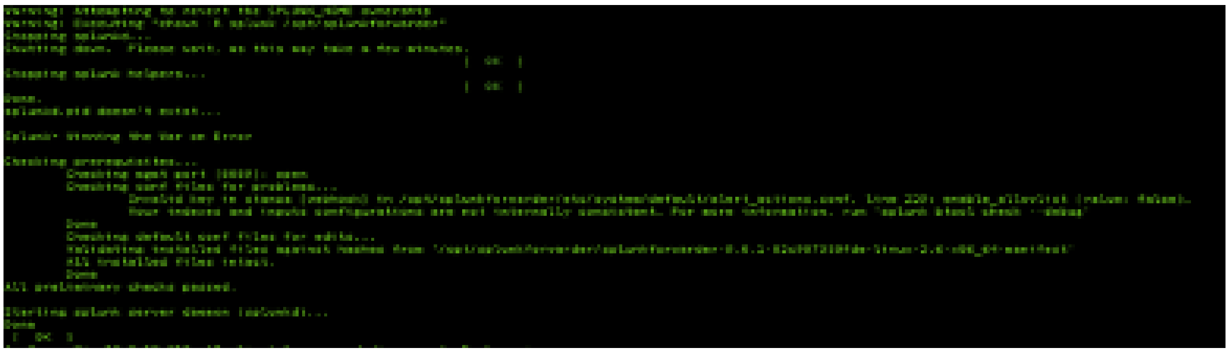


Figure 6: Splunk Cloud Login Page

Next, we needed to ingest the logs from the EC2 instances to the Splunk cloud. Hence; The universe forwarder was downloaded from the Splunk Cloud Official Website, and extracted per instructions with least-privilege access installation.
The forwarder on all hosts was then configured to send our desiered logs to the Splunk cloud on port 8089



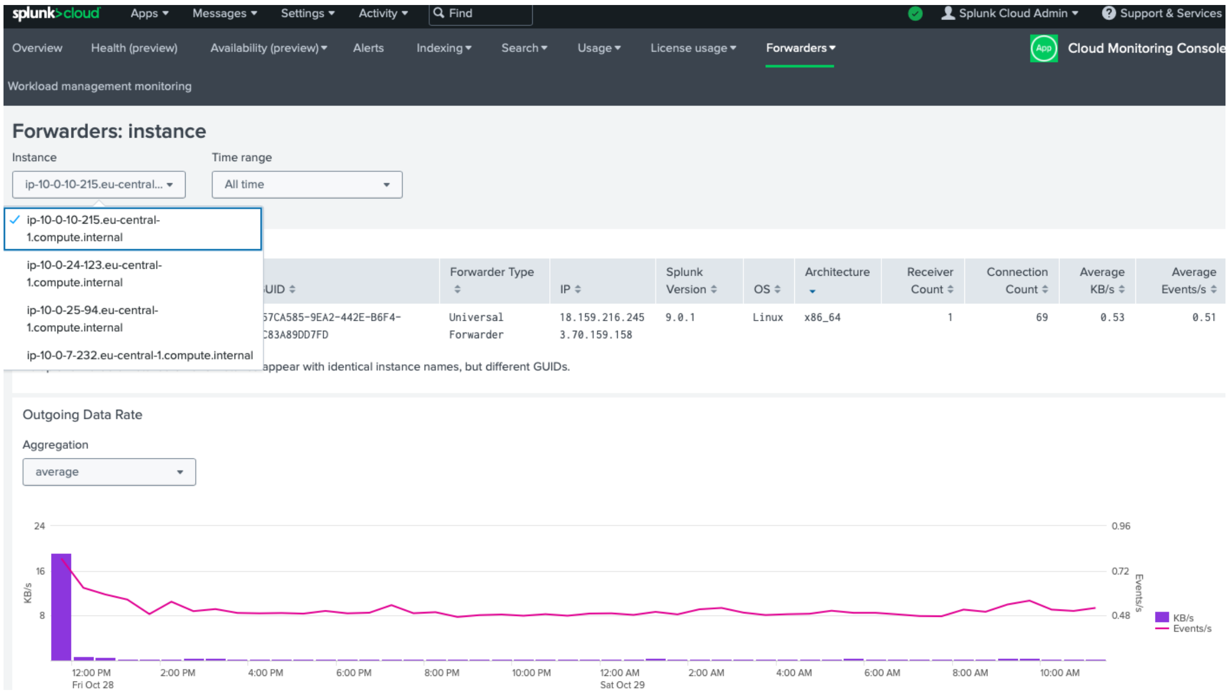
```
[illegible]
                                                            [  OK  ]
[illegible]
                                                            [  OK  ]
Done.
[illegible]

[illegible]
        Done
[illegible]
        Done
[illegible]

[illegible]
Done
 [  OK  ]
```

Figure 7: Forwarder configuration

next we verified that the ec2 instances were successfully on-boarded
we also checked the health of the all agents forwarder everything was good
Having done that Splunk was restarted and logs were sent to the Splunk cloud in real-time, for monitoring

5.4 The indexer:

To view the logs in the indexer, a simple query was made on splunk search head with the command: ”((index=”main”) (sourcetype=”Unity”)) — fields ”_time”, ”_raw”” The out put shows that the game logs were properly forwarded



Forwarders: instance

Instance: ip-10-0-10-215.eu-central...

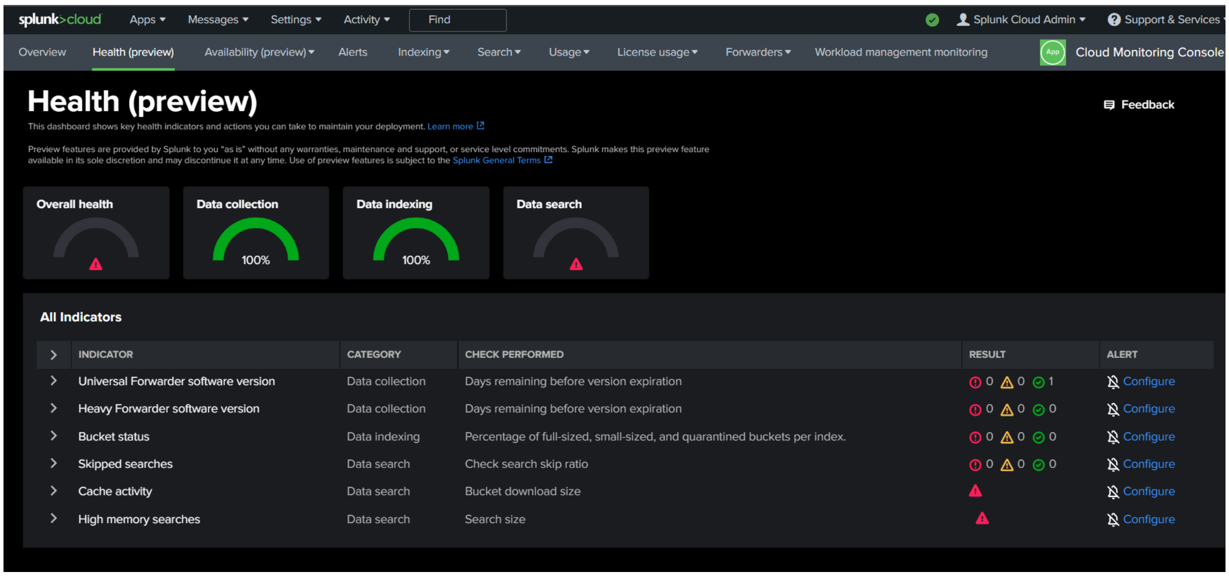
Time range: All time

✓ ip-10-0-10-215.eu-central-1.compute.internal
ip-10-0-24-123.eu-central-1.compute.internal
ip-10-0-25-94.eu-central-1.compute.internal
ip-10-0-7-232.eu-central-1.compute.internal

UID	Forwarder Type	IP	Splunk Version	OS	Architecture	Receiver Count	Connection Count	Average KB/s	Average Events/s
57CA585-9EA2-442E-B6F4-C83A89DD7FD	Universal Forwarder	18.159.216.245 3.70.159.158	9.0.1	Linux	x86_64	1	69	0.53	0.51

appear with identical instance names, but different GUIDs.

Figure 8: Instances visibility checks



Health (preview)

This dashboard shows key health indicators and actions you can take to maintain your deployment. Learn more

Preview features are provided by Splunk to you "as is" without any warranties, maintenance and support, or service level commitments. Splunk makes this preview feature available in its sole discretion and may discontinue it at any time. Use of preview features is subject to the Splunk General Terms

Overall health | Data collection 100% | Data indexing 100% | Data search

All Indicators

INDICATOR	CATEGORY	CHECK PERFORMED	RESULT	ALERT
Universal Forwarder software version	Data collection	Days remaining before version expiration	0 ⚠ 0 ✓ 1	Configure
Heavy Forwarder software version	Data collection	Days remaining before version expiration	0 ⚠ 0 ✓ 0	Configure
Bucket status	Data indexing	Percentage of full-sized, small-sized, and quarantined buckets per index.	0 ⚠ 0 ✓ 0	Configure
Skipped searches	Data search	Check search skip ratio	0 ⚠ 0 ✓ 0	Configure
Cache activity	Data search	Bucket download size	⚠	Configure
High memory searches	Data search	Search size	⚠	Configure

Figure 9: Splunk Forwarder Agent Health Dashboard

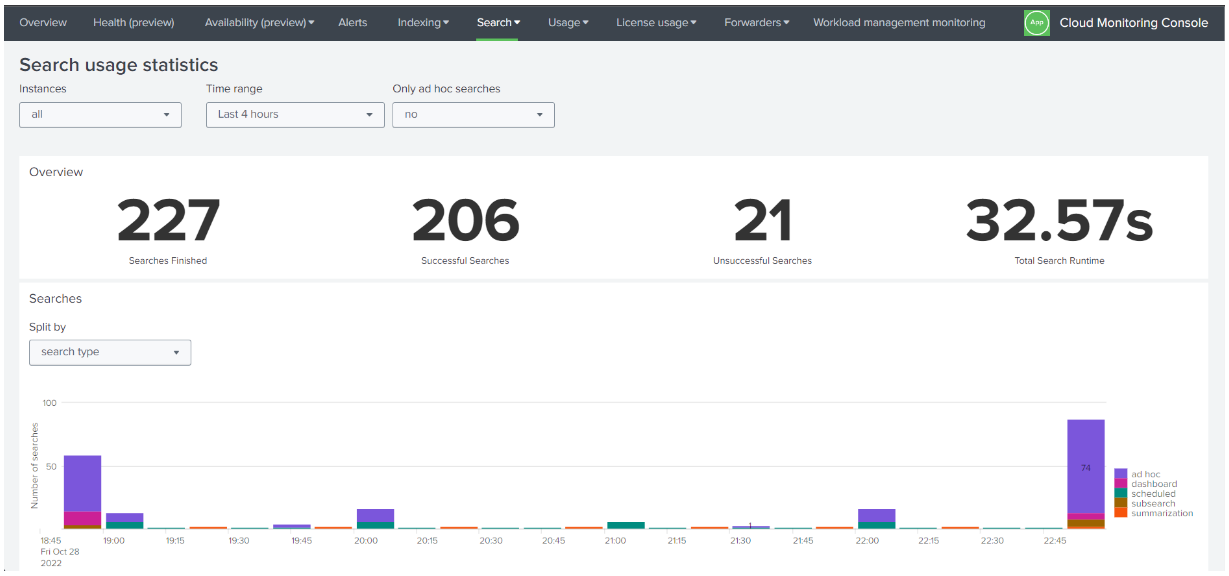
6 References

[1] Marios Assiotis and Velin Tzanov. "A Distributed Architecture for Massive Multiplayer Online Role-Playing Games". In: (June 2010).

[2] Sergio Caltagirone et al. "Architecture for a massively multiplayer online role playing game engine". In: *Journal of Computing Sciences in Colleges* 18 (2002), pp. 105–116.

[3] Tsun-Yu Hsiao and Shyan-Ming Yuan. "Practical middleware for massively multiplayer online games". In: *IEEE Internet Computing* 9.5 (2005), pp. 47–54. DOI: `10.1109/MIC.2005.106`.

[4] LIN WEI. *KCP KCP - A Fast and Reliable ARQ Protocol*. 2020. URL: `https://github.com/skywind3000/kcp` (visited on 10/29/2022).



Search usage statistics

Instances: all | Time range: Last 4 hours | Only ad hoc searches: no

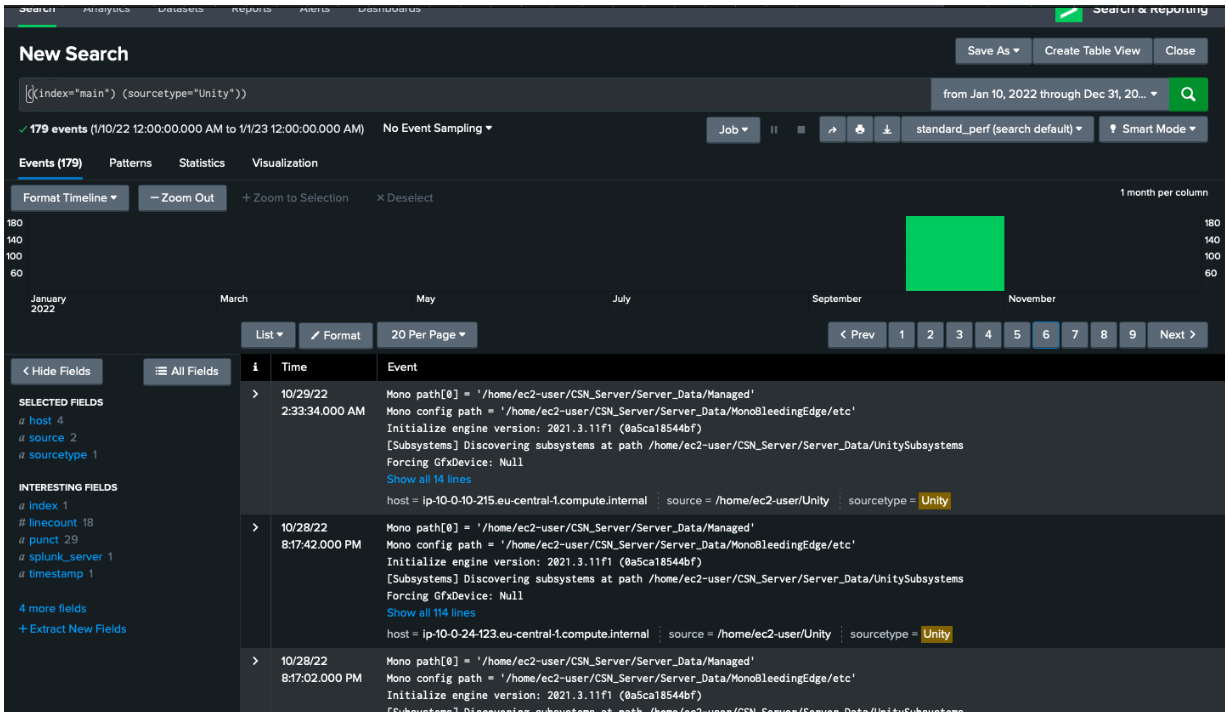
Overview

227	206	21	32.57s
Searches Finished	Successful Searches	Unsuccessful Searches	Total Search Runtime

Searches

Split by: search type

Figure 10: Splunk Forwarder Agents Use Statistics



New Search

```
(index="main") (sourcetype="Unity"))
```

from Jan 10, 2022 through Dec 31, 20...

✓ 179 events (1/10/22 12:00:00.000 AM to 1/1/23 12:00:00.000 AM) No Event Sampling

standard_perf (search default)

SELECTED FIELDS
a host 4
a source 2
a sourcetype 1

INTERESTING FIELDS
a index 1
# linecount 18
a punct 29
a splunk_server 1
a timestamp 1

4 more fields
+ Extract New Fields

i	Time	Event
>	10/29/22 2:33:34.000 AM	Mono path[0] = '/home/ec2-user/CSN_Server/Server_Data/Managed' Mono config path = '/home/ec2-user/CSN_Server/Server_Data/MonoBleedingEdge/etc' Initialize engine version: 2021.3.11f1 (0a5ca18544bf) [Subsystems] Discovering subsystems at path /home/ec2-user/CSN_Server/Server_Data/UnitySubsystems Forcing GfxDevice: Null Show all 14 lines host = ip-10-0-10-215.eu-central-1.compute.internal source = /home/ec2-user/Unity sourcetype = Unity
>	10/28/22 8:17:42.000 PM	Mono path[0] = '/home/ec2-user/CSN_Server/Server_Data/Managed' Mono config path = '/home/ec2-user/CSN_Server/Server_Data/MonoBleedingEdge/etc' Initialize engine version: 2021.3.11f1 (0a5ca18544bf) [Subsystems] Discovering subsystems at path /home/ec2-user/CSN_Server/Server_Data/UnitySubsystems Forcing GfxDevice: Null Show all 114 lines host = ip-10-0-24-123.eu-central-1.compute.internal source = /home/ec2-user/Unity sourcetype = Unity
>	10/28/22 8:17:02.000 PM	Mono path[0] = '/home/ec2-user/CSN_Server/Server_Data/Managed' Mono config path = '/home/ec2-user/CSN_Server/Server_Data/MonoBleedingEdge/etc' Initialize engine version: 2021.3.11f1 (0a5ca18544bf)

Figure 11: Splunk query