ENPM818P - Final Project Report

‘Creating and Deploying a Personal Package Archive (PPA) for SSH Log Monitoring’

GROUP 4

Bavan Mooganahally Yadunath
Derick Ansignia
Piyush Goenka
Tanvi Kanchan

TABLE OF CONTENTS

SR NO.	TITLE	PAGE NO.
1.	Introduction	1
2.	Project Components	2
3.	Personal Package Archive (PPA)	4
	3.1 Introduction	4
	3.2 Main script of the python package (main.py)	4
	3.3 PPA Specifications	9
	3.4 Installation Manual	9
4.	Ansible	22
5.	Firestore Database	28
6.	Deviation from initial proposal	29
7.	Working	30
8.	Testing Results	31
9.	Future Scope	35
10.	Conclusion	36
11.	References	37

1. INTRODUCTION

Servers are the backbone of critical infrastructure in the digital landscape, hence effective log monitoring and management have become critical for ensuring security, operational efficiency and compliance. With growing data needs of any organizations, their server ecosystems expand. This has heightened the need for centralized solutions to handle log data efficiently. Traditional methods of managing logs like relying solely on individual server log files is no longer sufficient, especially in scalable environments requiring real-time insights and rapid response to anomalies.

To address these challenges, a robust log monitoring and visualization service has been designed for Ubuntu server environments. The solution is packaged as a Personal Package Archive (PPA) and integrates seamlessly with a centralized log management platform like Firebase. The deployment is automated with tools like Ansible and leveraging centralized logging on Firebase. This architecture provides a streamlined approach to capturing and analyzing log data across multiple servers offering scalable, secure and user-friendly log monitoring capabilities. It records critical information such as login IP addresses, timestamps, usernames and authentication status for remote user logins, which is captured and forwarded for centralized processing.

The report documents the development lifecycle of the project, starting with the identification of key challenges in decentralized log management. It outlines the PPA's design, implementation and deployment, as well as the integration with a centralized platform to achieve data visualization. It also highlights the project's impact on reducing downtime, improving scalability to meet the needs of growing architecture and enhanced security through real-time log analysis enhancing overall security of server infrastructures.

As the solution is highly scalable and automated, it provides a transformative approach to log monitoring, ensuring that organizations can effectively manage and secure their server environments. It positions the infrastructure to adapt to future demands with ease due to being scalable, thus offering a long-term, sustainable solution for centralized log management.

2. PROJECT COMPONENTS

Control Node:
The control node acts as the central machine responsible for managing the deployment of the Personal Package Archive (PPA) across the three host nodes. It is the server where Ansible is installed and executed. From this control node, we use Ansible to automate the tasks of installing and configuring the PPA on the host nodes.

Managed Host Nodes:
Three host nodes are configured as target servers to be managed and monitored. These nodes are deployed with the PPA, which collects log data and stores it in Firebase for further analysis and monitoring. The nodes are configured to communicate securely with the control node via SSH, with Ansible managing the installation of necessary software and configuration changes.

Personal Package Archive (PPA):
The PPA is a custom package repository that hosts a collection of software and tools. In this project, the PPA is used to deploy a log collection application. This application is responsible for gathering logs from the host nodes and pushing the collected data to Firebase. By using a PPA, we ensure that the latest version of the log collection tool is installed and maintained on the host nodes.

Launchpad:
Launchpad is a platform used to manage the development, maintenance, and distribution of software packages and repositories. In the context of this project, Launchpad serves as the source for creating and hosting the PPA. Launchpad enables the easy creation of a personal repository for packaging the log collection software and distributing it to the host nodes. It simplifies version control and ensures that the PPA is up-to-date with the latest log collection software version.

Ansible Automation:
Ansible is used to automate the process of installing the PPA and ensuring its deployment on all host nodes. The automation includes:

- Adding the PPA repository to each host node.
- Installing the log collection application from the PPA.

Log Collection and Firebase Integration:

The log collection tool deployed from the PPA is designed to pull system logs from the host nodes. The logs are then formatted and sent to Firebase for storage. Firebase is used as the cloud database where the log data is stored and can be later queried for analysis, troubleshooting, and reporting purposes.

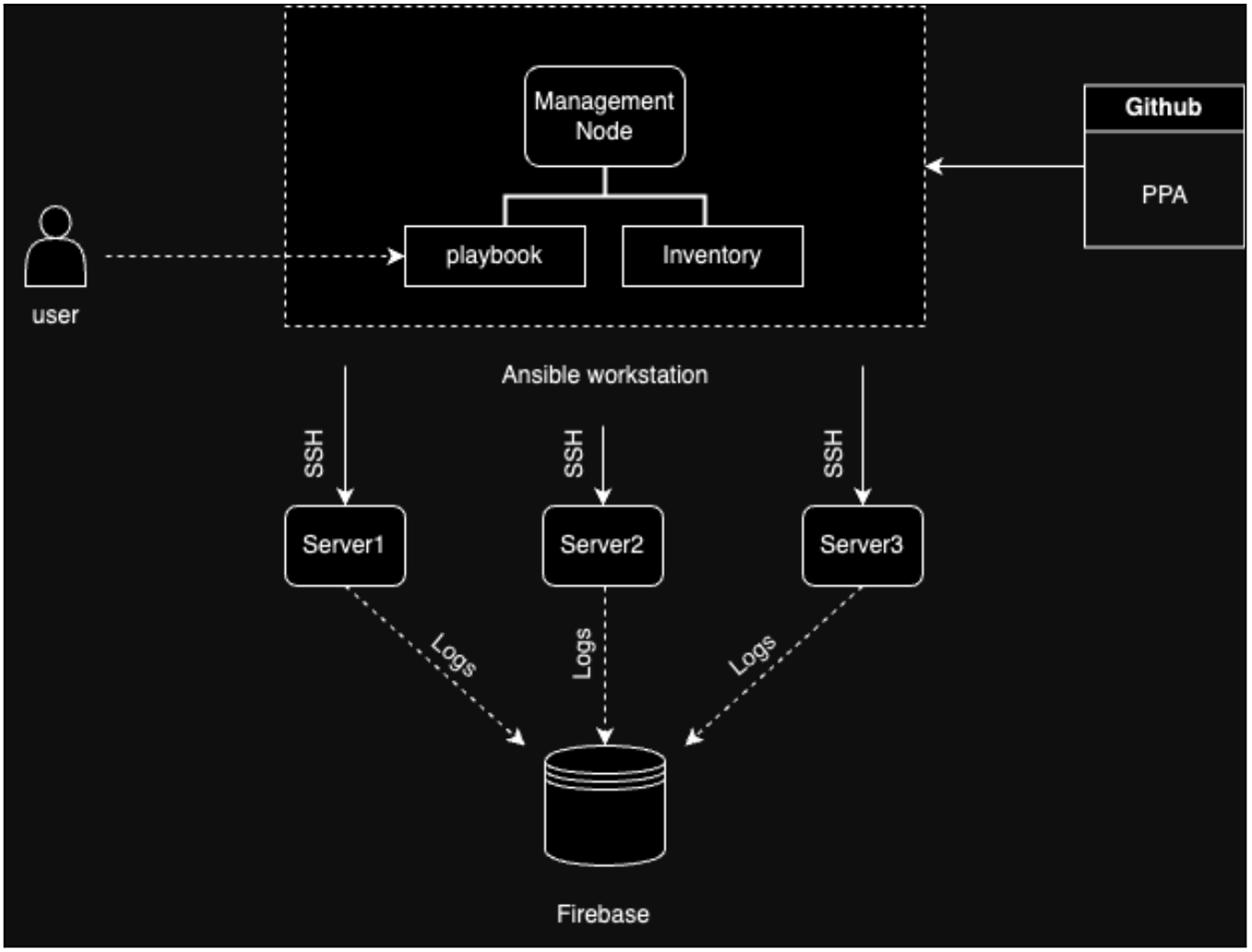


Fig: Project Architecture

3. PERSONAL PACKAGE ARCHIVE (PPA)

3.1 INTRODUCTION:

The Personal Package Archive (PPA) enables simplified deployment and scalability, allowing organizations to install this log monitoring solution effortlessly on Ubuntu 22.04 systems using the apt package manager. It processes authentication logs (***/var/log/auth.log***), extracts SSH related data and uploads the parsed information to Firebase Firestore database. This centralized storage ensures that data from multiple servers can be easily queried, visualized, and analyzed

The script captures three critical types of authentication events:

1. **Successful Logins**: Events where users with valid credentials successfully access the server.
2. **Failed Logins**: Attempts with valid usernames but incorrect passwords, often indicative of human error or brute-force attacks.
3. **Invalid Username Attempts**: Unauthorized attempts to log in with nonexistent usernames, signaling potential intrusion attempts.

Additionally, by integrating Firebase with local log monitoring, the script bridges the gap between raw data collection and actionable insights. Security administrators can now monitor server access remotely, identify patterns in login attempts, and respond to threats proactively. Furthermore, the inclusion of a console output provides a quick local summary of the parsed logs, allowing administrators to inspect recent activity without accessing the database.

3.2 MAIN SCRIPT OF THE PYTHON PACKAGE (main.py):

```
import re
import socket
import firebase_admin
from firebase_admin import credentials, firestore
import os







key = {
 "type": "service_account",
 "project_id": "log-board-f546f",
 "private_key_id": "9f18472aac5464f6956bdeb426bd79af3e0ea0b2",
 "private_key": "-----BEGIN PRIVATE
KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCv7A+gE9jQgw3l\nTh7LHSawfp2L1t98s1
clBmveA2KiPHAfRQvvnqZ2LMqbx6JZZmBfznk811QGPX9n\nG63nX0yL/3Bj4812NS9Jhnj00WAnjRGbtpk/pzu35crPyW
X5kIYeH5VEQ0Fyeb+b\nTFm3xv6P29RAwPr1wIYyIzX1ynKR/lnzfVgtnnVyWZU5BOB2KknXJQ/VKB1nKbZt\nCOVD+Odb
iwPn8dWIkVnCMFWGkuLyMPN3+iVaKwZPVw4n4dwoaffgR2iq4st7UIV0\nu2eBGEMsZp73Ua6ClrgtcSnra4ISAZeMUyyY
WJQ6tOsSK3WENQZukUGHt0Odq5ki\nA4g9PhxtAgMBAAECggEABqYobQdzAWaCpvEN7uVaKf+PGtLzPXZVy9tzRj/ggXtB
```

```
\nTZ1BdzwK1JqaycqA01owOE7IfEkzLs0yPIZqSz1p2jy8VbotcNNPxzqi5bXoKXaG\nfNko76P1eNzhjnrxNL6GoyrUbH
ESTy5U1JzNI0oNIwjO++ePVKyDJe/4wSJ2tBfV\nytBcNoe3dqyN/mOeuUSn0k4udzTVQZKs1pBQNTidPkHxkOILItXlsG
72eBmcWyh+\nOC5OnN0ySYANohx0zh39ajY3pwj6BB1SMVjLjDdZjQnMEJi9yhis4H13XhC8Pzou\n2S+/Whm2h09MvIpx
WebmjpP6FJKnxyu70pZOsJeJAQKBgQDv7Y7h0eErau24xrhn\nrngH7rnO8ytzUl/o4UT3U8DIaJHuCEFV/VE8PiS9kvNx
5btK8rl90XvSEFjlJeFr\njQFCshg5og43l3t1BXju9uLJimiOea64425RGm/5Id79mD1WZdFS2AVS1PauOdvZ\nylplz/
H1oSfOWy/CUR6UVwllrQKBgQC7tOOBHkDAiDHP8mPrt724WhrETg+iFMue\n6PjeYWghi88/IPG160KKnr7q1SBF3zUg2s
XKp9AN1LaGNpkZAniRm1xcsID2Ksyp\nOLERMBj5cAm7fuWUsgXi3xZJ8zBL19/x15bGcL0FjhFTs7TMyDGZ9KiHrxaay0
Rx\nmjVUv9bpwQKBgQDL8rbbVIpAWwO6ZwnH8LHNpGtD9McBn4sxrZPvXb1pXiJ2IYRQ\ncSXFvNmEYE8aiXsc6Zktz42e
chDusLWes9zXcgZPr2bAcL8cKUu0bh83zLS2L8+L\nEKDsxjlaoCUqkmfJcoFuF3zm/Wq3jS1KQ8118D6wWo44xqChYE1L
MXUmfQKBgH4J\nPr2C982PZYsQYYF0FaLEdbacB6k4N2UTX4/KbeYgf18PGt2hA0QaxWzfAFWKQ500\n6Jxjw7KVwp0p6f
C+FQEIBaODLISeRl/V3icBWbObRB2uXi9SvdsRGmE7NpGmeQtl\nTCsgpfsl4oKGZv8d6RhMDCFXMmFr3SA+gFLQoVjBAo
GAXwy6HU3TCjn8EIl+Oujn\nsftsneBS09SH3ipFU247Y1SquTFBLgk+i8jXVIOvUcSZWScm2GipkctGdRKdt395\nsLX/
sV81XqZaImt24vsAuaU77AsAY+C3LMhGbqZsKPJGcjxoooK3QdeBxITl3RXN\nTeuYuyzn/nm5tvsk8+OHiFY=\n-----E
ND PRIVATE KEY-----\n",
 "client_email": "firebase-adminsdk-o5km5@log-board-f546f.iam.gserviceaccount.com",
 "client_id": "103828888614300363185",
 "auth_uri": "https://accounts.google.com/o/oauth2/auth",
 "token_uri": "https://oauth2.googleapis.com/token",
 "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
 "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-o5km5%40log-board-f546f.i
am.gserviceaccount.com",
 "universe_domain": "googleapis.com"
}


# Define log file location
log_file_path = '/var/log/auth.log'
# current_directory = os.path.dirname(os.path.abspath(__file__))
# log_file_path = os.path.join(current_directory, 'auth.log')

# Define regular expressions for different log types
valid_username_correct_password_pattern = re.compile(r'Accepted password for (\S+) from (\S+)
port (\d+)')
valid_username_incorrect_password_pattern = re.compile(r'Failed password for (\S+) from (\S+)
port (\d+)')
invalid_username_pattern = re.compile(r'Invalid user (\S+) from (\S+) port (\d+)')

# Updated regex to handle timestamp format with spaces
timestamp_pattern = re.compile(r'(\w{3} \s?\d{1,2} \d{2}:\d{2}:\d{2})')

# Data storage for different buckets
valid_username_correct_password = []
valid_username_incorrect_password = []
invalid_username = []

# Get the hostname and IP address of the server
hostname = socket.gethostname()
```

```
server_ip = socket.gethostbyname(hostname)

# Firebase initialization
# file_path = os.path.join(current_directory, 'key.json')
cred = credentials.Certificate(key)
firebase_admin.initialize_app(cred)

# Firestore client
db = firestore.client()



def clear_firestore_collection(collection_name):
   collection_ref = db.collection(hostname)
   docs = collection_ref.stream()
   for doc in docs:
       doc.reference.delete()

   #db.collection("logs").document(hostname).delete()
   # print(f"Cleared all documents in the collection '{collection_name}'.")




# Function to process the logs and segregate them
def process_logs():
   with open(log_file_path, 'r') as file:
       for line in file:
           # Check for valid username with correct password
           match_valid_correct = valid_username_correct_password_pattern.search(line)
           if match_valid_correct:
               username, ip_address, port = match_valid_correct.groups()
               timestamp = extract_timestamp(line)
               valid_username_correct_password.append({
                   'timestamp': timestamp,
                   'ip_address': ip_address,
                   'port': port,
                   'username': username
               })
               continue

           # Check for valid username with incorrect password
           match_valid_incorrect = valid_username_incorrect_password_pattern.search(line)
           if match_valid_incorrect:
               username, ip_address, port = match_valid_incorrect.groups()
               timestamp = extract_timestamp(line)
               valid_username_incorrect_password.append({
                   'timestamp': timestamp,
                   'ip_address': ip_address,
                   'port': port,
```

```
                    'username': username
                })
                continue

            # Check for invalid username
            match_invalid = invalid_username_pattern.search(line)
            if match_invalid:
                username, ip_address, port = match_invalid.groups()
                timestamp = extract_timestamp(line)
                invalid_username.append({
                    'timestamp': timestamp,
                    'ip_address': ip_address,
                    'port': port,
                    'username': username
                })

# Helper function to extract timestamp from log line
def extract_timestamp(line):
    timestamp_match = timestamp_pattern.match(line)
    return timestamp_match.group(1) if timestamp_match else None

# Function to upload the data to Firebase
def upload_to_firebase():
    clear_firestore_collection('logs')

    entries_one = {}
    for entry in valid_username_correct_password:

        entries_one[entry['timestamp']]={

            'client_ip_address': entry['ip_address'],
            'client_port': entry['port'],
            'server_username': entry['username']
        }

    v_u_c_p = {"Valid Username Correct Password": entries_one}

    db.collection(hostname).document("Valid Username Correct Password").set(entries_one)

    # Upload valid username with incorrect password data
    entries_two = {}
    for entry in valid_username_incorrect_password:
        entries_two[entry['timestamp']]={

            'client_ip_address': entry['ip_address'],
            'client_port': entry['port'],
            'server_username': entry['username']
        }
```

```
    v_u_ic_p = {"Valid Username Incorrect Password": entries_two}

    db.collection(hostname).document("Valid Username Incorrect Password").set(entries_two)

    # Upload invalid username data
    entries_three = {}
    for entry in invalid_username:
        entries_three[entry['timestamp']]={

            'client_ip_address': entry['ip_address'],
            'client_port': entry['port'],
            'server_username': entry['username']
        }

    iv_u = {"Invalid Username": entries_three}



    db.collection(hostname).document("Invalid Username").set(entries_three)

# Function to print the logs in the required format
def print_buckets():
    print(f"\nServer Hostname: {hostname} | IP Address: {server_ip}")

    print("\n--- Valid Username, Correct Password ---")
    for entry in valid_username_correct_password:
        print(f"Timestamp: {entry['timestamp']}, IP Address: {entry['ip_address']}, Port:
{entry['port']}, Username: {entry['username']}")

    print("\n--- Valid Username, Incorrect Password ---")
    for entry in valid_username_incorrect_password:
        print(f"Timestamp: {entry['timestamp']}, IP Address: {entry['ip_address']}, Port:
{entry['port']}, Username: {entry['username']}")

    print("\n--- Invalid Username ---")
    for entry in invalid_username:
        print(f"Timestamp: {entry['timestamp']}, IP Address: {entry['ip_address']}, Port:
{entry['port']}, Username: {entry['username']}")


def run_logger():
    process_logs()
    upload_to_firebase()
    print_buckets()

if __name__ == "__main__":
    run_logger()
```

3.3 PPA SPECIFICATIONS:

The development and deployment of a Personal Package Archive (PPA) were integral to this project, ensuring the seamless distribution and installation of the log monitoring script across multiple Ubuntu servers. A PPA serves as a dedicated repository for custom Debian packages, allowing users to install and manage software conveniently through the *apt* package manager. By leveraging the PPA, the project delivers a streamlined, user-friendly mechanism to distribute the log monitoring tool, eliminating the need for manual script setup and configuration.

The PPA package includes all necessary dependencies and configurations required for the script to function effectively, ensuring compatibility with Ubuntu distributions.

The current PPA is supported on systems meeting the following requirements:

- Ubuntu Version: 22.04 - Jammy
- Underlying VM architecture: AMD64 Architecture

3.4 INSTALLATION MANUAL:

In order to deploy the above script (main.py) as a python package into our PPA, we need to follow two main steps:

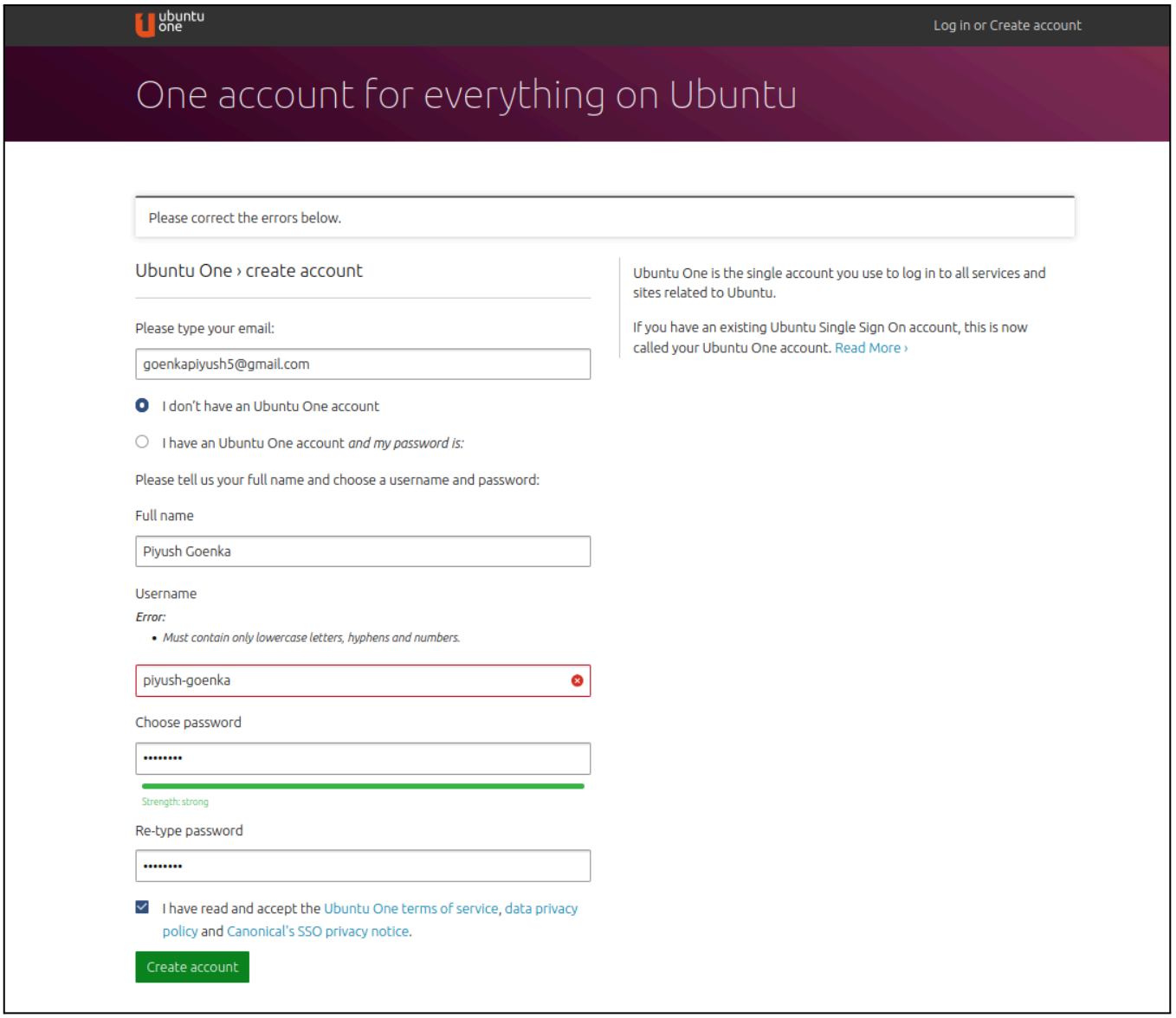
a. Setting up a PPA
   - Creating a launchpad account
   - Creating a PPA
   - Creating and registering an OpenPGP key
   - Adding the key to the launchpad account

b. Setting up and deploying the python package
   - Creating a standard directory structure for .deb packages
   - Building the python package
   - Publishing the python package in PPA

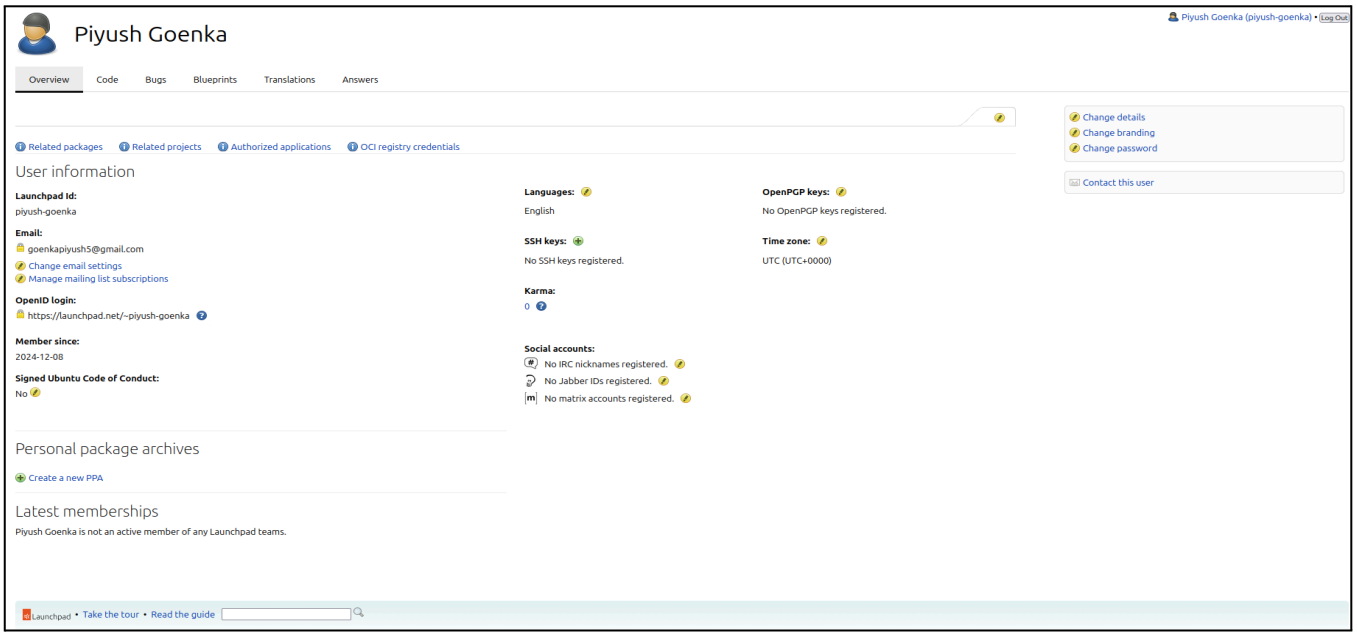
3.4.1 Setting up a PPA:

A. Creating a Launchpad account and a PPA

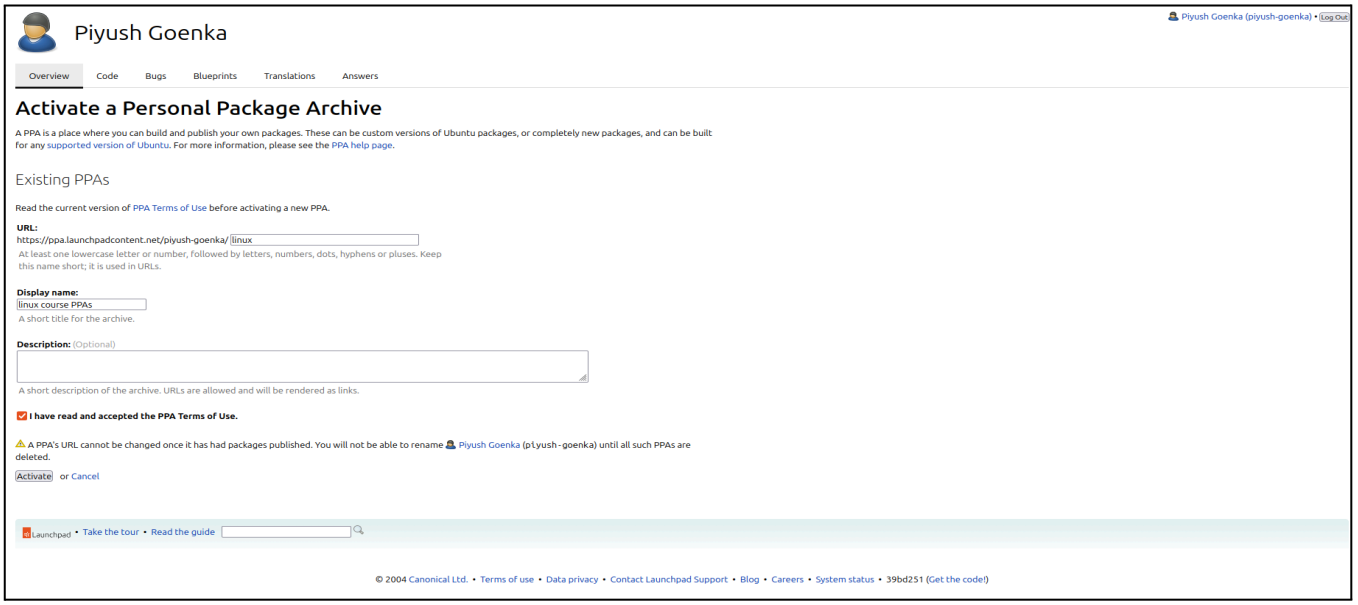
- Go to the launchpad login page and create an account. Then verify your email address.



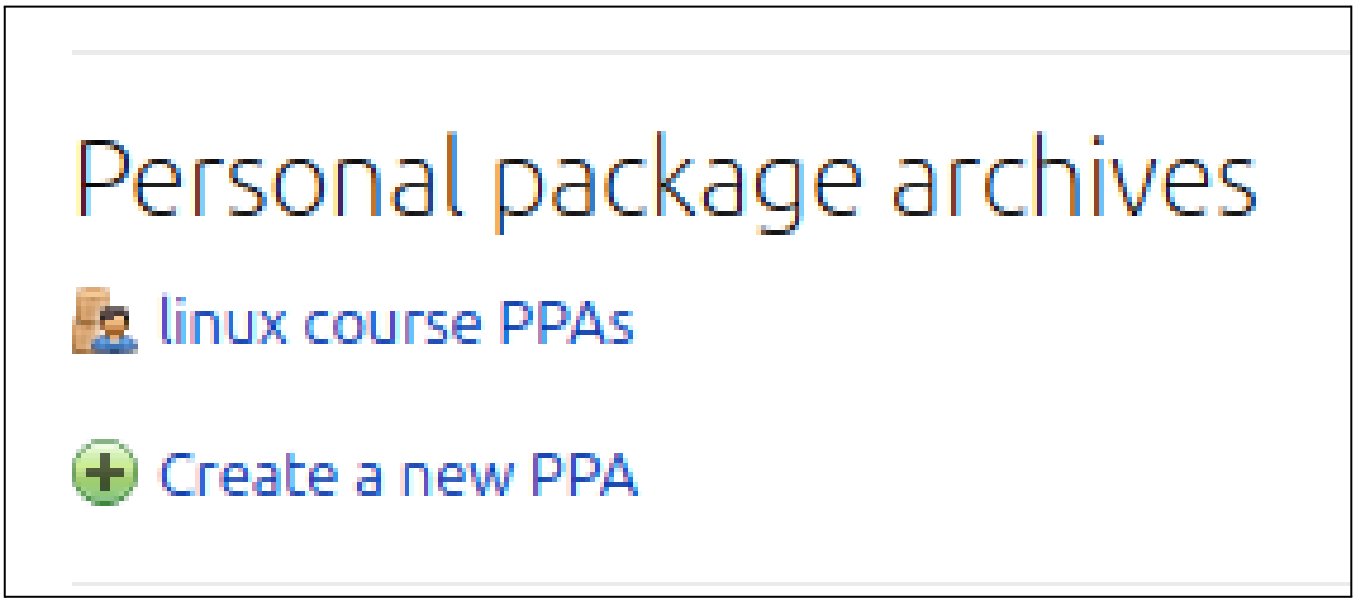
- Login to your launchpad account and go to your homepage. The homepage url is https://launchpad.net/~<username>. In this case it is https://launchpad.net/~piyush-goenka



- Click on *Create a new PPA* button located in the bottom-left corner of your launchpad homepage. Add details to the PPA and click on *Activate* to create a new PPA.



- In the homepage, in this case https://launchpad.net/~piyush-goenka, it can be seen that the PPA has been created and is present under the *Personal package archives* section.



B. Creating and Registering an OpenPGP key

- Install the gpg command-line interface and generate a gpg key

```
$ sudo apt install gpg
$ gpg --full-generate-key
```

When generating a key, follow these steps:

1. **Select Key Type:** When prompted, choose the type of key you want to create, or press Enter to accept the default option.
2. **Specify Key Size:** Enter the desired key size, or press Enter to use the default value.
3. **Set Key Validity Period:** Input the duration for which the key will remain valid. Press Enter to select the default option, which indicates the key will not expire.
4. **Confirm Selections:** Verify the information you've entered to ensure it is correct.
5. **Provide User ID Details:** Enter your user ID information when prompted, including your name and email address.
6. **Secure passphrase:** Type in a passphrase when prompted. This passphrase will be required while signing the source files.

```
robotics@wall:~$ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection?
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072)
Requested keysize is 3072 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 10
Key expires at Tue 17 Dec 2024 08:25:22 PM EST
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Piyush Goenka
Email address: goenkapiyush5@gmail.com
Comment: first gpg key
You selected this USER-ID:
    "Piyush Goenka (first gpg key) <goenkapiyush5@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 6355D2006C2DA2F9 marked as ultimately trusted
gpg: revocation certificate stored as '/home/robotics/.gnupg/openpgp-revocs.d/B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9.rev'
public and secret key created and signed.

pub   rsa3072 2024-12-08 [SC] [expires: 2024-12-18]
      B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9
uid                      Piyush Goenka (first gpg key) <goenkapiyush5@gmail.com>
sub   rsa3072 2024-12-08 [E] [expires: 2024-12-18]
```

- The gpg key would be created. To view the gpg key type the below command. In this case the GPG key is *B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9*

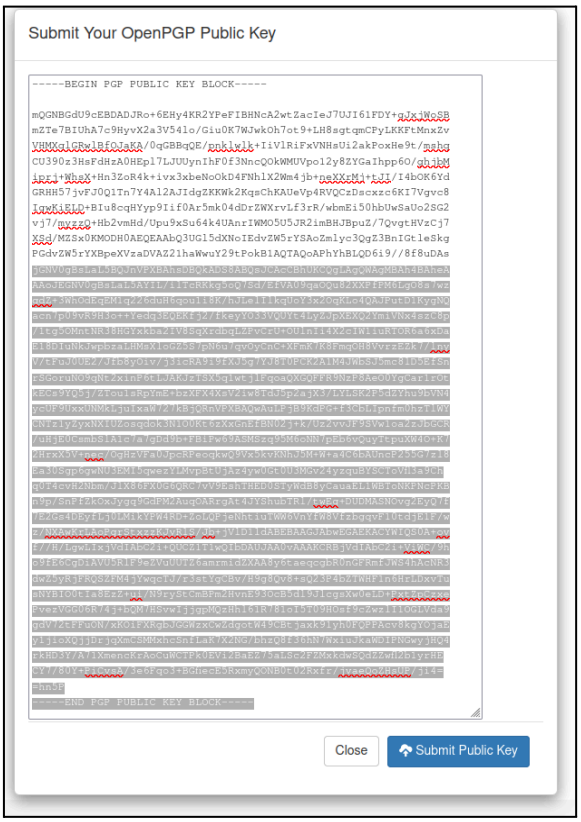
$ gpg --list-keys

```
pub   rsa3072 2024-12-08 [SC] [expires: 2024-12-18]
      B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9
uid           [ultimate] Piyush Goenka (first gpg key) <goenkapiyush5@gmail.com>
sub   rsa3072 2024-12-08 [E] [expires: 2024-12-18]
```

- Export the gpg key into a .txt file inorder to connect it with launchpad and other applications by uploading the key into a keyserver.

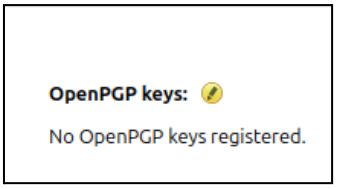
$ gpg --armor --export B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9 > gpg_key.txt

- Go to https://keyserver.ubuntu.com/ to upload the key to keyserver. Click on the *Submit* key and paste the contents from gpg_key.txt. Then click on the *Submit Public key* button.

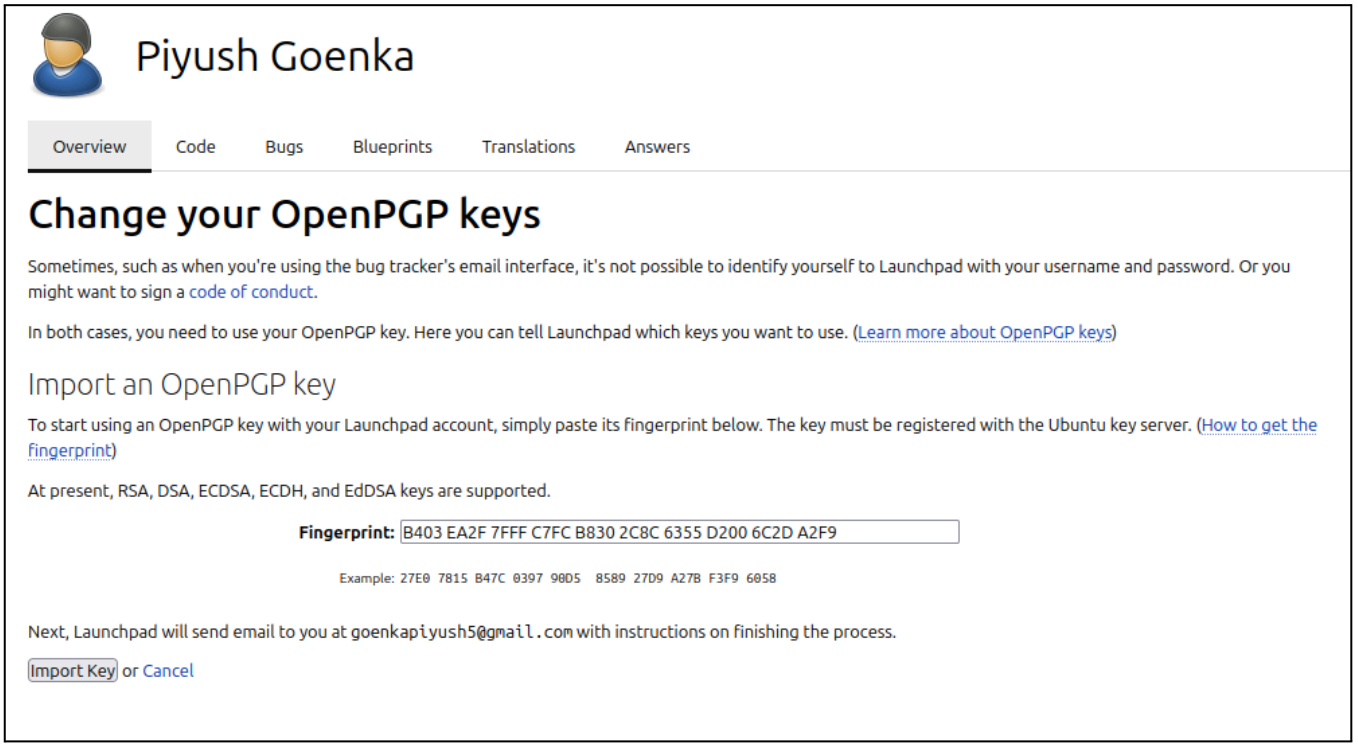


C. Importing the GPG key in Launchpad

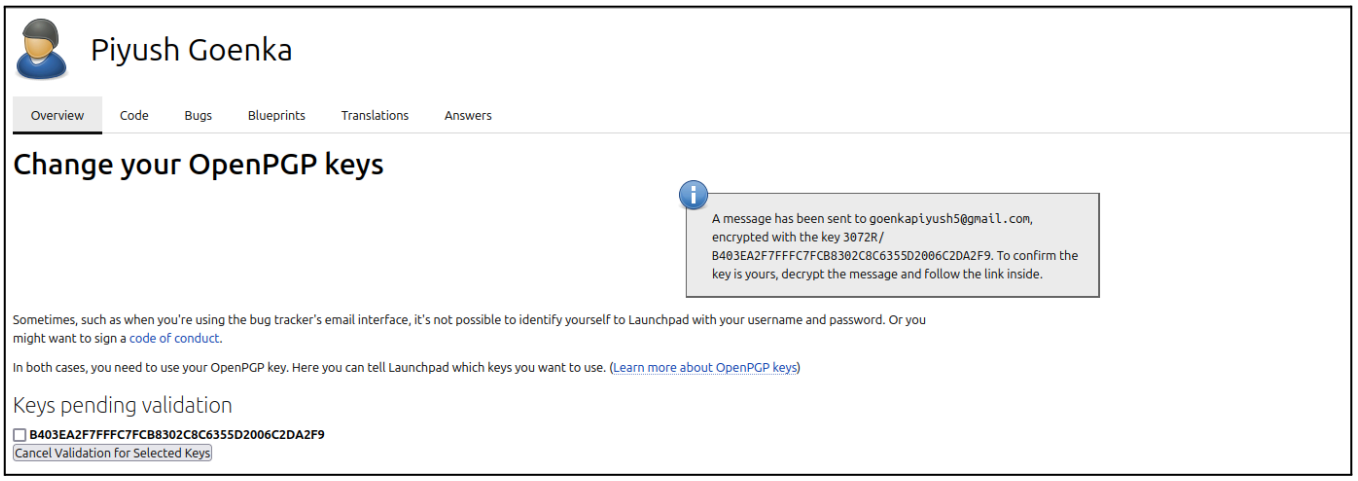
- Wait for around 30 minutes for the key to be registered. Then go to your launchpad account home page and click on the circular button next to the OpenPGP keys heading.



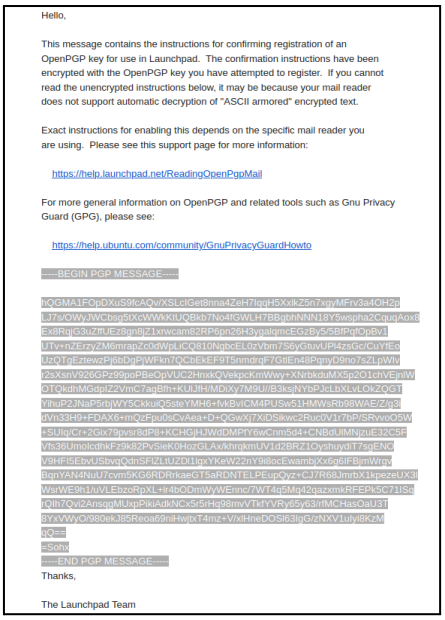
- Paste your gpg-key and click on *Import Key*



- You would then see a message that you need to verify the key.



- You will get an email from launchpad. Copy and paste the gpg-key text present in the email into a file. In this case, we created a file named decr.txt



Hello,

This message contains the instructions for confirming registration of an OpenPGP key for use in Launchpad. The confirmation instructions have been encrypted with the OpenPGP key you have attempted to register. If you cannot read the unencrypted instructions below, it may be because your mail reader does not support automatic decryption of "ASCII armored" encrypted text.

Exact instructions for enabling this depends on the specific mail reader you are using. Please see this support page for more information:

https://help.launchpad.net/ReadingOpenPgpMail

For more general information on OpenPGP and related tools such as Gnu Privacy Guard (GPG), please see:

https://help.ubuntu.com/community/GnuPrivacyGuardHowto

-----BEGIN PGP MESSAGE-----

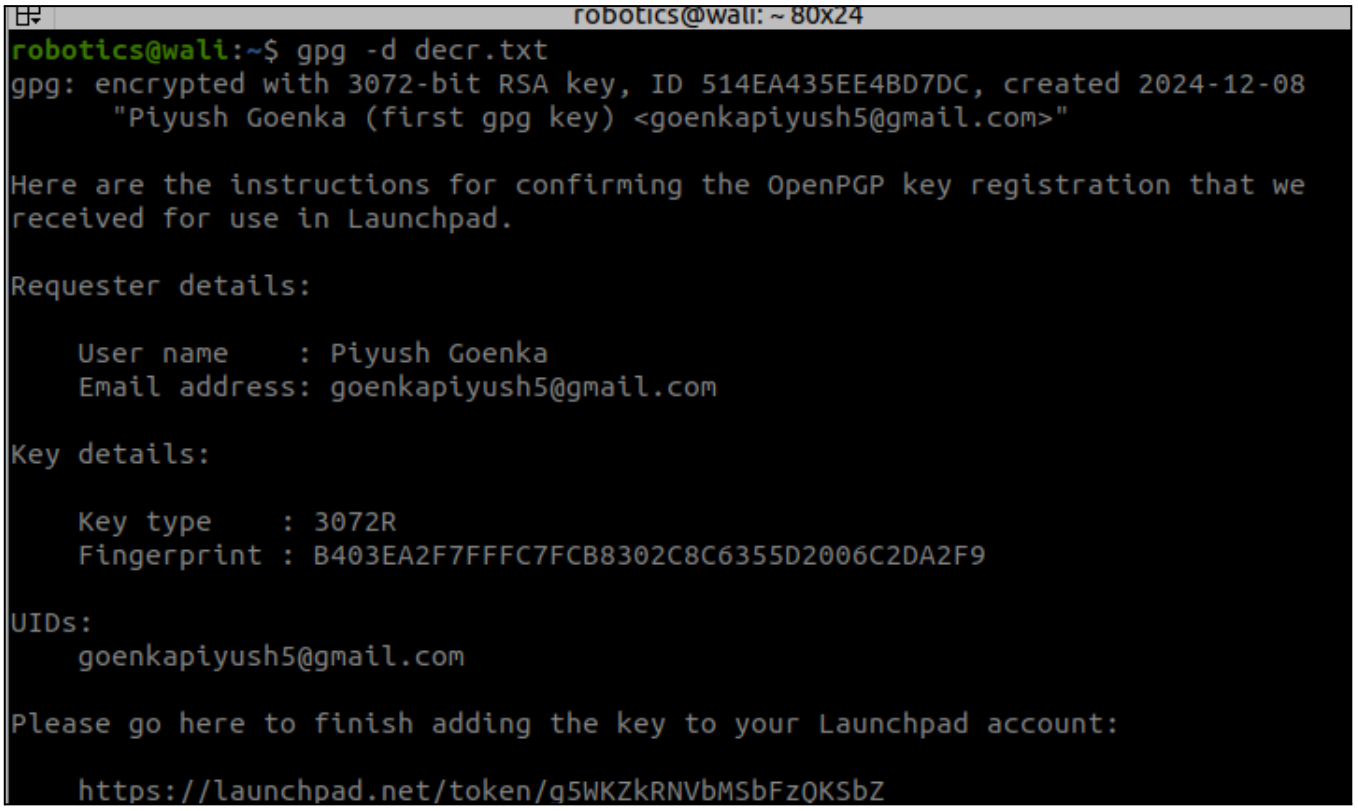
hQGMA1FOpDXuS9fcAQv/XSLcfGet8nna4ZeH7IqqH5XxlkZ5n7xgyMFrv3a4OH2p
LJ7s/OWyJWCbsq5tXcWWkKfUQBkb7No4fGWLH7BBgbhNNN18Y5wspha2CqugAcxB
Ex8RqIG3uZffUEz8gn8jZ1xrwcam82RP6pn26H3ygalqmcEGzBy5/5BfPqfOpBv1
UTv+nZErzyZM6mrapZc0dWpLiCQ810NgbcEL0zVbm7S6yGtuvUPI4zsGc/CuYfE6
UzQTqEztewzPj6bDgPiWFkn7QCbEkEF9T5nmdrqF7GtlEn48PqnyD9no7sZLpWlv
r2sXsnV926GPz99poPBeOpVUC2HnxkQVekpcKmWwy+XNrbkduMX5p2O1chVEjnlW
OTQkdhMGdpIZ2VmC7agBfh+KUlJfH/MDiXy7M9U//B3ksjNYbPJcLbXLvLOkZQGT
YlhuP2JNaP5rbjWY5CkkuiQ5steYMH6+fvkBvICM4PUSw51HMWsRb98WAE/Zlg3
dVn33H9+FDAX6+mQzFpu0sCvAea+D+QGwXj7XiDSikwc2Ruc0V1r7bP/SRvvoO5W
+SUlq/Cr+2Glx79pvsr8dP8+KCHGjHJWdDMPfY6wCnm5d4+CNBdUlMNjzuE32C5F
Vfs36UmoIcdhkFz9k82PvSieK0HozGLAxvkhrgkmUV1d2BRZ1OyshuydiT7sqENO
V9HFI5EbvUSbvgQdnSFlZLtUZDt1lqxYKeW22nY9I8ocEwambjXx6q6IFBjmWrgv
BqnYAN4NuU7cvm5KG6RDRrkaeGT5aRDNTELPEupQyz+CJ7R68JmrbX1kpezeUX3I
WsrWE9h1/uVLEbzoRpXL+lr4bODmWyWEnnc/7WT4q5Mg42qazxmkRFEPk5C71ISd
rOlh7Owi2AnsqqMUxpPikiAdkNCx5rHq98mvVTkfYVRy65y63/rfMCHasOaU3T
8YvVWyOV980ekJ85Reoa69nlHwjtxT4mz+VxlHneDOSl63IgGrzNXV1uIyI8K2M
jQ==
=SQmj
-----END PGP MESSAGE-----

Thanks,

The Launchpad Team

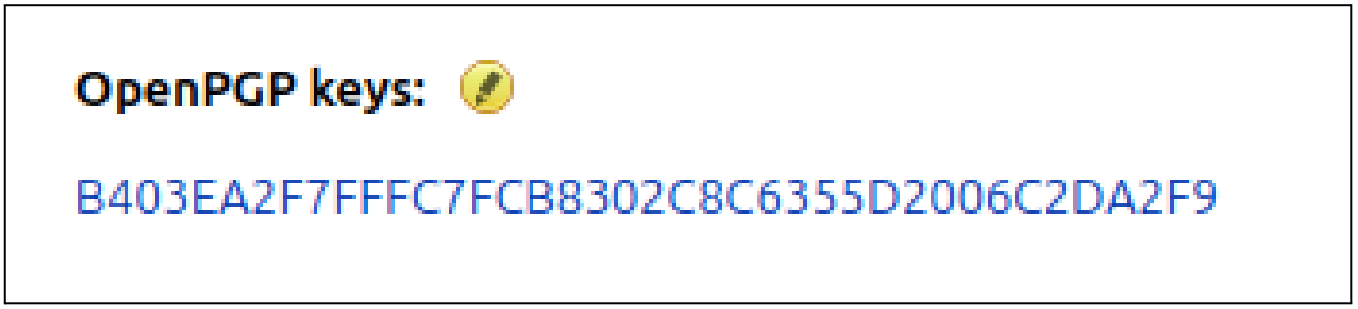
- Open terminal in the directory of the file and run the below command. Then you will get a link at the last line. Open that link in a browser.

$ gpg -d decr.txt



- Confirm the key. Click on Continue.

- Go to your launchpad homepage. You will find the openPGP key listed.



3.4.2 Setting up and deploying the python package:

A. Creating a standard directory structure for .deb packages

- In our case, we are creating a python package named *ssh-logger*. The package structure to be followed to breathe a .deb is as follows:

```
ssh-logger
    |---- debian
    |       |---- changelog
    |       |---- compat
    |       |---- control
    |       |---- rules
    |---- src
    |       |---- __init__.py
    |       |---- main.py
    |---- setup.py
```

- **debian/changelog:** This file outlines the changes made since the previous release. Equally important—if not more so—is that it adheres to a standardized format, allowing the packaging software to extract the package and version details seamlessly.

```
ssh-logger (1.0.0) jammy; urgency=medium

  * jammy release

 -- Piyush Goenka <goenkapiyush5@gmail.com>  Wed, 04 Dec 2024 14:47:24 -0400
```

- **debian/compat:** According to the documentation, this file specifies the debhelper compatibility level. In simple terms, it should contain only the number 10 and nothing else.

```
10
```

- **debian/control:** The control file serves as a guide for the packaging system, providing details about the package and instructions on handling it. Since we are uploading to a PPA, the system requires a source package rather than a binary one. Therefore, this control file must contain two distinct sections: one describing the source package and another for the binary package that the PPA will generate. The first section focuses on the source package, outlining its key details.

```
Source: ssh-logger
Maintainer: Piyush Goenka <goenkapiyush5@gmail.com>
Build-Depends: debhelper,dh-python,python3-all,python3-setuptools
Section: devel
Priority: optional
Standards-Version: 3.9.6
X-Python3-Version: >= 3.6


Package: ssh-logger
Architecture: any
Description: Log SSH events
Depends: ${python3:Depends},python3-requests
```

- **debian/rules:** Functioning as little more than a Makefile, this file instructs the build system on how to manage the package. Fortunately, there are several tools to assist with this process. For instance, you can use dh_make to create this file. Additionally, pybuild simplifies the process by taking care of the standard build requirements for Python packages.

```
#! /usr/bin/make -f
#export DH_VERBOSE = 1
export PYBUILD_NAME = ssh-logger

%:
  dh $@ --with python3 --buildsystem=pybuild
```

- **setup.py:** This file contains a few important elements such as:

    1. **install_requires:** This specifies the dependencies required by the package. For example, this package depends on *firebase-admin,* so it's included here.

2. **packages:** This is a list of all the packages included in the library. We utilize the helpful *setuptools* function *find_packages* to automatically locate and list them.
3. **entry_points / console_scripts:** By defining a console script, we instruct *setuptools* to create an executable (in this case, called *ssh_logger*) that runs the *run_logger* function from the *src.main* module. The goal is to make the package versatile—usable both as a library for other Python applications and as an executable script that can be run directly.

```
from setuptools import setup, find_packages


setup(
  name='ssh-logger',
  version='1.0.0',
  description='logging of ssh authentication events',
  author='Piyush Goenka',
  author_email='goenkapiyush5@gmail.com',
  license='MIT',
  install_requires=['firebase-admin>=6.0.0'],
  packages=find_packages(),
  entry_points=dict(
        console_scripts=['ssh_logger=src.main:run_logger']
  )
)
```

B. Building and Publishing the Debian package

- Building the package:

1. **Find Your Key ID**: Identify the key you uploaded to Launchpad. This is required to sign the package. Use the command:

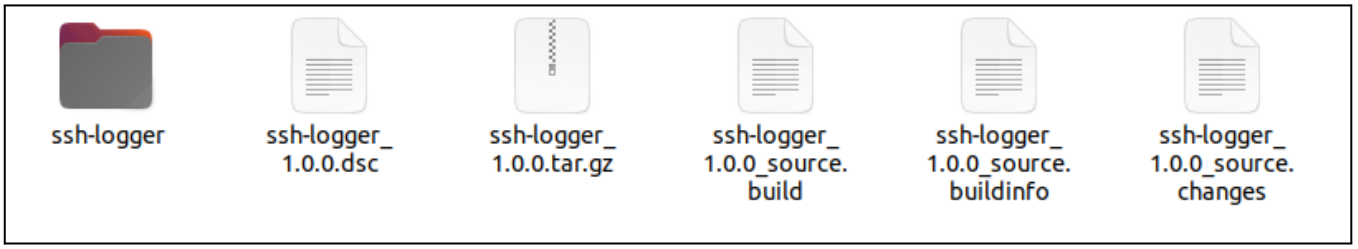
```
$ gpg --list-keys
```

2. **Build and Sign the Package**: From the repository root, build and sign the source package using the key ID:

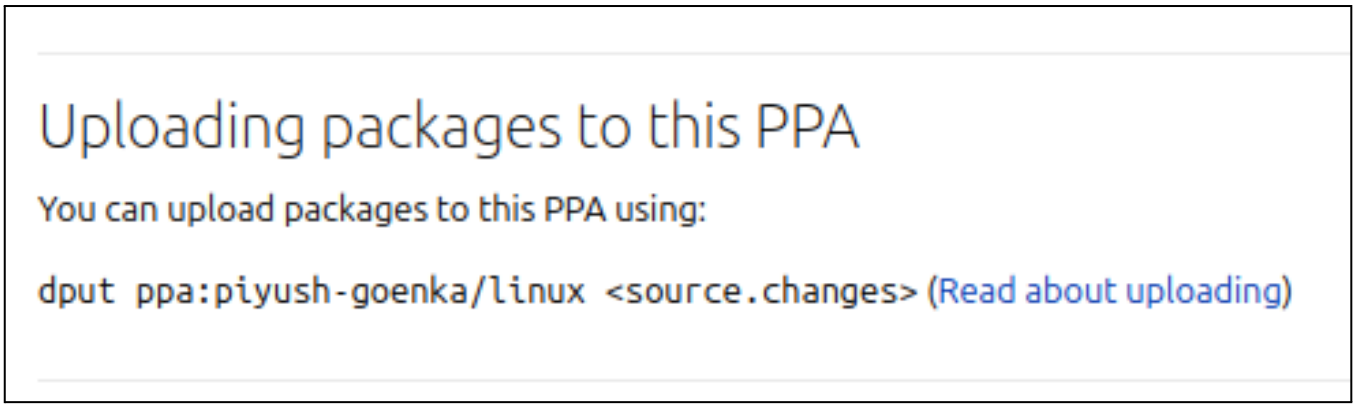
```
debuild -k"<key_id>" -S

# In this case the command looks like below
debuild -k"B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9" -S
```

```
robotics@wali:~/courses/linux/log/ssh-logger$ debuild -k"B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9" -S
 dpkg-buildpackage -us -uc -ui -S
dpkg-buildpackage: info: source package ssh-logger
dpkg-buildpackage: info: source version 1.0.0
dpkg-buildpackage: info: source distribution jammy
dpkg-buildpackage: info: source changed by Piyush Goenka <goenkapiyush5@gmail.com>
 dpkg-source --before-build .
 fakeroot debian/rules clean
dh clean --with python3 --buildsystem=pybuild
   dh_auto_clean -O--buildsystem=pybuild
I: pybuild base:239: python3.10 setup.py clean
running clean
removing '/home/robotics/courses/linux/log/ssh-logger/.pybuild/cpython3_3.10_ssh-logger/build' (and everything under it)
'build/bdist.linux-x86_64' does not exist -- can't clean it
'build/scripts-3.10' does not exist -- can't clean it
   dh_autoreconf_clean -O--buildsystem=pybuild
   dh_clean -O--buildsystem=pybuild
 dpkg-source -b .
dpkg-source: warning: no source format specified in debian/source/format, see dpkg-source(1)
dpkg-source: warning: source directory 'ssh-logger' is not <sourcepackage>-<upstreamversion> 'ssh-logger-1.0.0'
dpkg-source: info: using source format '1.0'
dpkg-source: info: building ssh-logger in ssh-logger_1.0.0.tar.gz
dpkg-source: info: building ssh-logger in ssh-logger_1.0.0.dsc
 dpkg-genbuildinfo --build=source -O../ssh-logger_1.0.0_source.buildinfo
 dpkg-genchanges --build=source -O../ssh-logger_1.0.0_source.changes
dpkg-genchanges: info: including full source code in upload
 dpkg-source --after-build .
dpkg-buildpackage: info: source-only upload: Debian-native package
Now running lintian ssh-logger_1.0.0_source.changes ...
W: ssh-logger source: debhelper-but-no-misc-depends ssh-logger
W: ssh-logger source: missing-debian-source-format
W: ssh-logger source: no-debian-copyright-in-source
W: ssh-logger source: no-versioned-debhelper-prerequisite 10
Finished running lintian.
Now signing changes and any dsc files...
 signfile dsc ssh-logger_1.0.0.dsc B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9

 fixup_buildinfo ssh-logger_1.0.0.dsc ssh-logger_1.0.0_source.buildinfo
 signfile buildinfo ssh-logger_1.0.0_source.buildinfo B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9

 fixup_changes dsc ssh-logger_1.0.0.dsc ssh-logger_1.0.0_source.changes
 fixup_changes buildinfo ssh-logger_1.0.0_source.buildinfo ssh-logger_1.0.0_source.changes
 signfile changes ssh-logger_1.0.0_source.changes B403EA2F7FFFC7FCB8302C8C6355D2006C2DA2F9

Successfully signed dsc, buildinfo, changes files
```

- Certain files get created in the same directory as the package root folder



3. **Upload the Package to the PPA**: You can find the exact command in your PPA's page in launchpad website. Run below command in the folder containing the above files.



dput <ppa_uri> <source.changes>

 # In our case the command is as below
dput ppa:piyush-goenka/linux ssh-logger_1.0.0_source.changes

 # ppa_uri: This can be found on your Launchpad PPA page, formatted as ppa:<username>/<ppa_name>.
 # source.changes: The changes file generated by the debuild command.

```
robotics@wali:~/courses/linux/log$ dput ppa:piyush-goenka/linux ssh-logger_1.0.0_source.changes
D: Splitting host argument out of  ppa:piyush-goenka/linux.
D: Setting host argument.
Checking signature on .changes
gpg: /home/robotics/courses/linux/log/ssh-logger_1.0.0_source.changes: Valid signature from 6355D2006C2DA2F9
Checking signature on .dsc
gpg: /home/robotics/courses/linux/log/ssh-logger_1.0.0.dsc: Valid signature from 6355D2006C2DA2F9
Uploading to ppa (via ftp to ppa.launchpad.net):
  Uploading ssh-logger_1.0.0.dsc: done.
  Uploading ssh-logger_1.0.0.tar.gz: done.
  Uploading ssh-logger_1.0.0_source.buildinfo: done.
  Uploading ssh-logger_1.0.0_source.changes: done.
Successfully uploaded packages.
```

- **Launchpad Notification**: After submission, Launchpad will send an email informing you whether the package was accepted or rejected.
    - If **rejected**, the email will provide the reason, allowing you to fix the issue and re-upload.
    - If **accepted**, Launchpad will begin building the package from the source.

[~piyush-goenka/ubuntu/linux/jammy] ssh-logger 1.0.0 (Accepted)

Launchpad PPA <noreply@launchpad.net>
to me

Accepted:
OK: ssh-logger_1.0.0.tar.gz
OK: ssh-logger_1.0.0.dsc
-> Component: main Section: devel

Upload Warnings:
No copyright file found.

ssh-logger (1.0.0) jammy; urgency=medium

* jammy release

--
https://launchpad.net/~piyush-goenka/+archive/ubuntu/linux
You are receiving this email because you made this upload.

- **Build Results**:
    - A successful build means the package is ready for use in your PPA.
    - If the build fails, you can inspect the build log on Launchpad to identify and resolve the issue.

Once the process completes successfully, your .deb package will be published to your PPA! You would need to wait for a few hours for the package to be published in your PPA. Then it can be installed and used.

Source		Published	Status	Series	Section	Build Status
ssh-logger - 1.1.0	(changes file)	1 hour ago	Published	Jammy	Devel	✓

Publishing details

Published 1 hour ago

Changelog

```
ssh-logger (1.1.0) jammy; urgency=medium

  * jammy release - fixed log issues

 -- Piyush Goenka <goenkapiyush5@gmail.com>  Wed, 05 Dec 2024 14:47:24 -0400
```

Available diffs

diff from 1.0.0 to 1.1.0 (614 bytes)

Builds

amd64

Built packages

ssh-logger Log SSH events

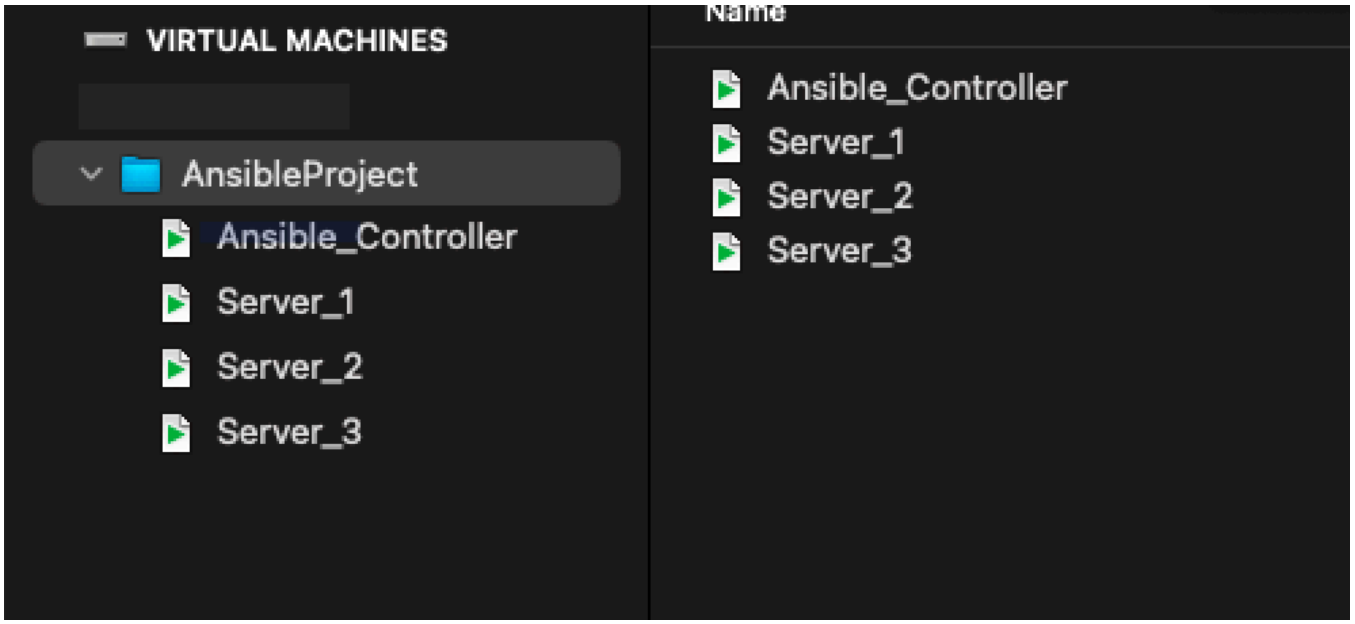
Package files

ssh-logger_1.1.0.dsc (1.2 KiB)
ssh-logger_1.1.0.tar.gz (18.0 KiB)
ssh-logger_1.1.0_amd64.deb (5.5 KiB)

4. ANSIBLE

Steps in Implementing Ansible Control Node and Managed Nodes

1. One machine for the control node (the machine where Ansible will run) and three other servers (managed nodes) are installed in the environment.



2. On the control node, generate a new SSH key pair. This will be used to securely authenticate with the managed servers.

```
ctrladmin@controller:~/Desktop/ansible_tut$ ssh-keygen -t ed25519 -C "Ansible Manager"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/ctrladmin/.ssh/id_ed25519): /home/ctrladmin/.ssh/AnsibleManager
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ctrladmin/.ssh/AnsibleManager
Your public key has been saved in /home/ctrladmin/.ssh/AnsibleManager.pub
The key fingerprint is:
SHA256:NM8mm5MsMU+JgJX301xADHnQrniS50C8ZaX64R6yoxA Ansible Manager
The key's randomart image is:
+--[ED25519 256]--+
|     ..  oBo.    |
|   o. . . = .    |
|  . .o .oB .     |
|     .oo*=+      |
|  E  .+BSo+      |
|    .  B*=*      |
|   .   oO0.      |
|    .  .++o      |
|     ...o.       |
+----[SHA256]-----+
```

3. Transfer the generated SSH public key to the managed servers to enable secure, passwordless authentication for subsequent SSH sessions.

```
ctrladmin@controller:~/Desktop/ansible_tut$ ssh-copy-id -i ~/.ssh/AnsibleManager.pub svradmin@192.168.158.146
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/ctrladmin/.ssh/AnsibleManager.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'svradmin@192.168.158.146'"
and check to make sure that only the key(s) you wanted were added.
```

```
svradmin@server1:~$ cd .ssh/
svradmin@server1:~/.ssh$ ls
authorized_keys
svradmin@server1:~/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMxPC1IHO8nTus/NBwaTv/9w7wOqquwyhDF2/JjbwUj7 Default Login
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAQEwWrxPdUDh69IDKO0F5m3qJZvJw1lsvJU+4QWCF2H ansible
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDWXR2FIdJyGsvqftUjmHov49bejVLe5vhzaSEJLOCga Ansible Manager
svradmin@server1:~/.ssh$
```

4. Test SSH authentication by logging into the managed servers from the control node to verify that key-based authentication is functioning correctly.

```
ctrladmin@controller:~/Desktop/ansible_tut$ ssh svradmin@192.168.158.146
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-126-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Sun Dec  8 01:19:34 AM UTC 2024

  System load:  1.65              Processes:             221
  Usage of /:   63.9% of 9.75GB   Users logged in:       1
  Memory usage: 25%               IPv4 address for ens33: 192.168.158.146
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Sat Dec  7 03:49:35 2024 from 192.168.158.145
svradmin@server1:~$
```

Successful login

5. To install Ansible on the control node, follow these steps:

```
sudo apt update
sudo apt install ansible -y
```

- Verify the installation

```
ansible --version
```

6. To manage the servers with Ansible, create an inventory file that lists the IP addresses of the servers to be managed.

```
ctrladmin@controller:~/Desktop/ansible_tut$ ls
ansible.cfg  inventory.ini  ppa_packages.yml  README.md  ssh_logger.yml
ctrladmin@controller:~/Desktop/ansible_tut$ cat inventory.ini
[servers]

192.168.158.146 ansible_python_interpreter=/usr/bin/python3
192.168.158.143 ansible_python_interpreter=/usr/bin/python3
192.168.158.144 ansible_python_interpreter=/usr/bin/python3
ctrladmin@controller:~/Desktop/ansible_tut$
```

7. To verify that the control node can communicate with the remote servers, use Ansible's ping module to test the connection. This ensures that Ansible can reach the managed servers and is able to execute commands.

```
ctrladmin@controller:~/Desktop/ansible_tut$ ansible all -m ping -i inventory.ini -u svradmin
192.168.158.144 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.158.146 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
192.168.158.143 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ctrladmin@controller:~/Desktop/ansible_tut$
```

8. To deploy a Personal Package Archive (PPA) onto your servers, create an Ansible playbook that automates the process. This playbook will install the desired PPA on the remote servers and ensure the necessary software is installed.

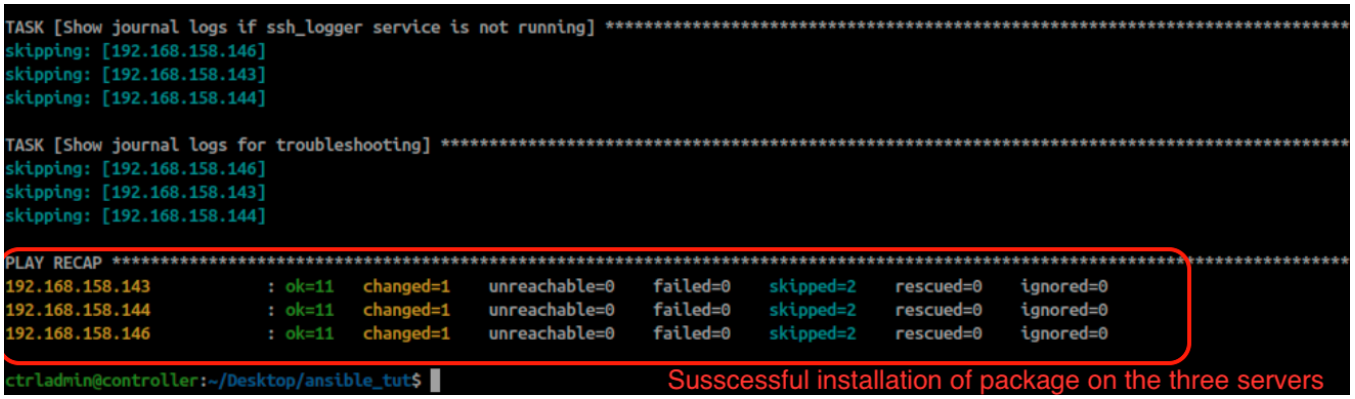
```
ctrladmin@controller:~/Desktop/ansible_tut$ cat ssh_logger.yml
---
- name: Add PPA repository, update packages, and install required packages
  hosts: all
  become: true
  tasks:
    # Add PPA repository for hello-world
    - name: Add PPA repository for hello-world
      apt_repository:
        repo: ppa:pgoenka/hello-world
        state: present

    # Update apt cache and upgrade packages
    - name: Update apt cache and upgrade packages
      apt:
        update_cache: yes
        upgrade: yes   # Regular upgrade, avoids dist-upgrade unless necessary
      environment:
        DEBIAN_FRONTEND: noninteractive  # Prevents prompts during package upgrades

    # Install Python3 and pip if they are not installed
    - name: Ensure Python3 and pip are installed
      apt:
        name:
          - python3
          - python3-pip
        state: present
        update_cache: yes

    # Install firebase-admin Python package
    - name: Install firebase-admin Python package
      pip:
        name: firebase-admin
        state: present

    # Install my-ssh-logger package from the newly added PPA
    - name: Install my-ssh-logger package from PPA
      apt:
        name: my-ssh-logger
```

9. To execute the playbook while ensuring that Ansible prompts for a sudo (elevation) password and uses a specific user to connect to the remote servers, use the following command:

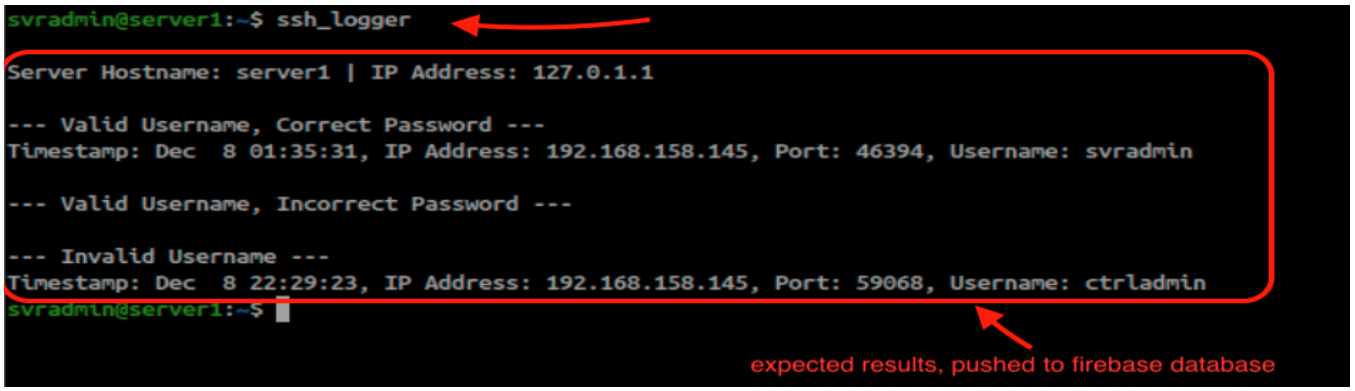
ansible-playbook –ask-become-pass -i inventory.ini -u <admin_username> –become ssh_logger.yml

```
ctrladmin@controller:~/Desktop/ansible_tut$ ansible-playbook --ask-become-pass -i inventory.ini -u svradmin --become ssh_logger.yml
BECOME password:

PLAY [Add PPA repository, update packages, and install required packages] ***********************************************************************

TASK [Gathering Facts] ***********************************************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Add PPA repository for hello-world] ****************************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Update apt cache and upgrade packages] *************************************************************************************************
ok: [192.168.158.144]
ok: [192.168.158.146]
ok: [192.168.158.143]

TASK [Ensure Python3 and pip are installed] **************************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Install firebase-admin Python package] *************************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Install my-ssh-logger package from PPA] ************************************************************************************************
ok: [192.168.158.144]
ok: [192.168.158.146]
ok: [192.168.158.143]
```

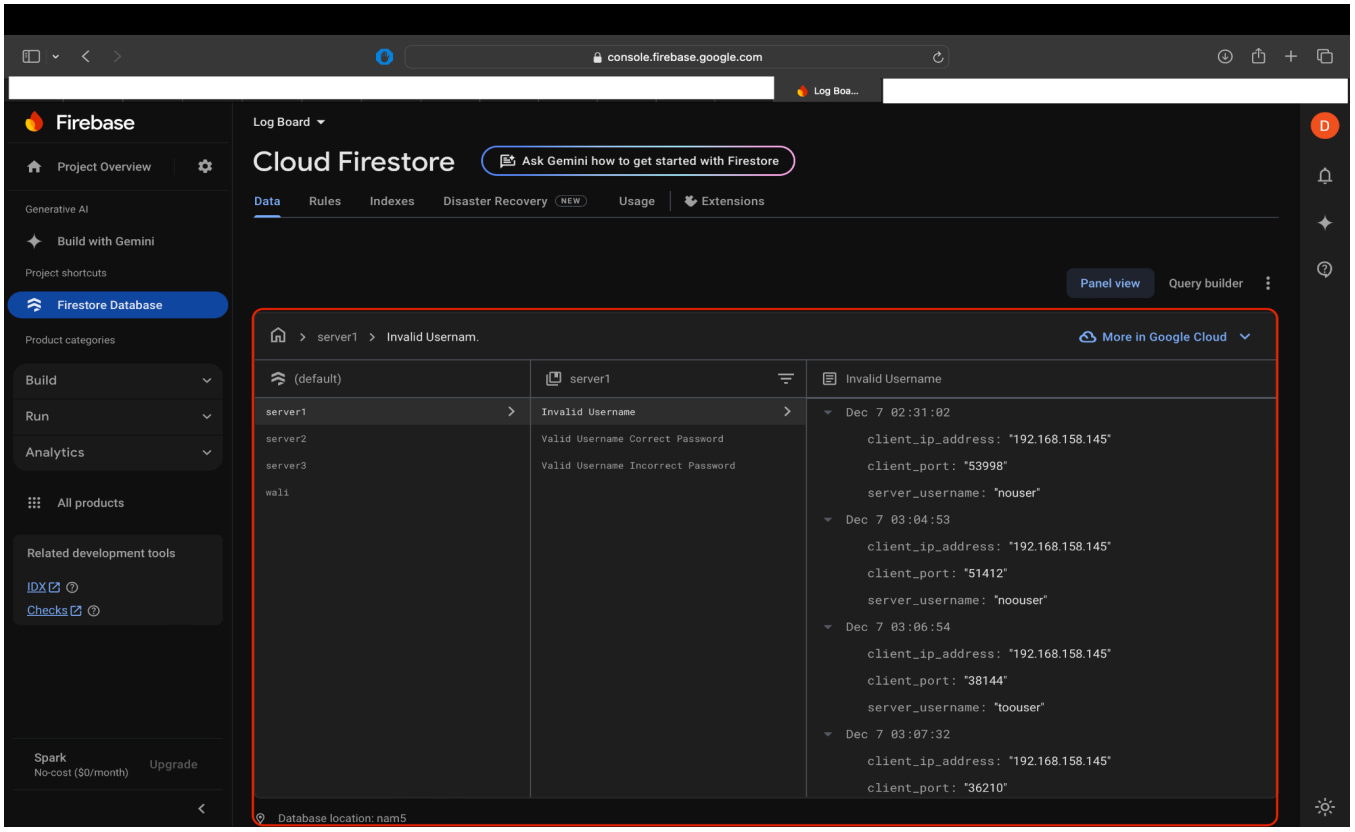
```
TASK [Create custom systemd service for ssh_logger] ******************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Reload systemd to apply new service] ***************************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Enable the ssh_logger service to start on boot] ****************************************************************************************
ok: [192.168.158.146]
ok: [192.168.158.144]
ok: [192.168.158.143]

TASK [Check if the ssh_logger service is running] ********************************************************************************************
changed: [192.168.158.146]
changed: [192.168.158.144]
changed: [192.168.158.143]

TASK [Show status of ssh_logger service] *****************************************************************************************************
ok: [192.168.158.146] => {
    "service_status.stdout": "active"
}
ok: [192.168.158.143] => {
    "service_status.stdout": "active"
}
ok: [192.168.158.144] => {
    "service_status.stdout": "active"
}
```



10. Once you've run your Ansible playbook (e.g. ssh_logger.yml) it's important to manually verify that the playbook's actions were successfully applied on the managed servers. This step ensures that the intended configuration was deployed correctly.



11. This integration helps with tracking SSH activity, conducting audits, and implementing security measures.



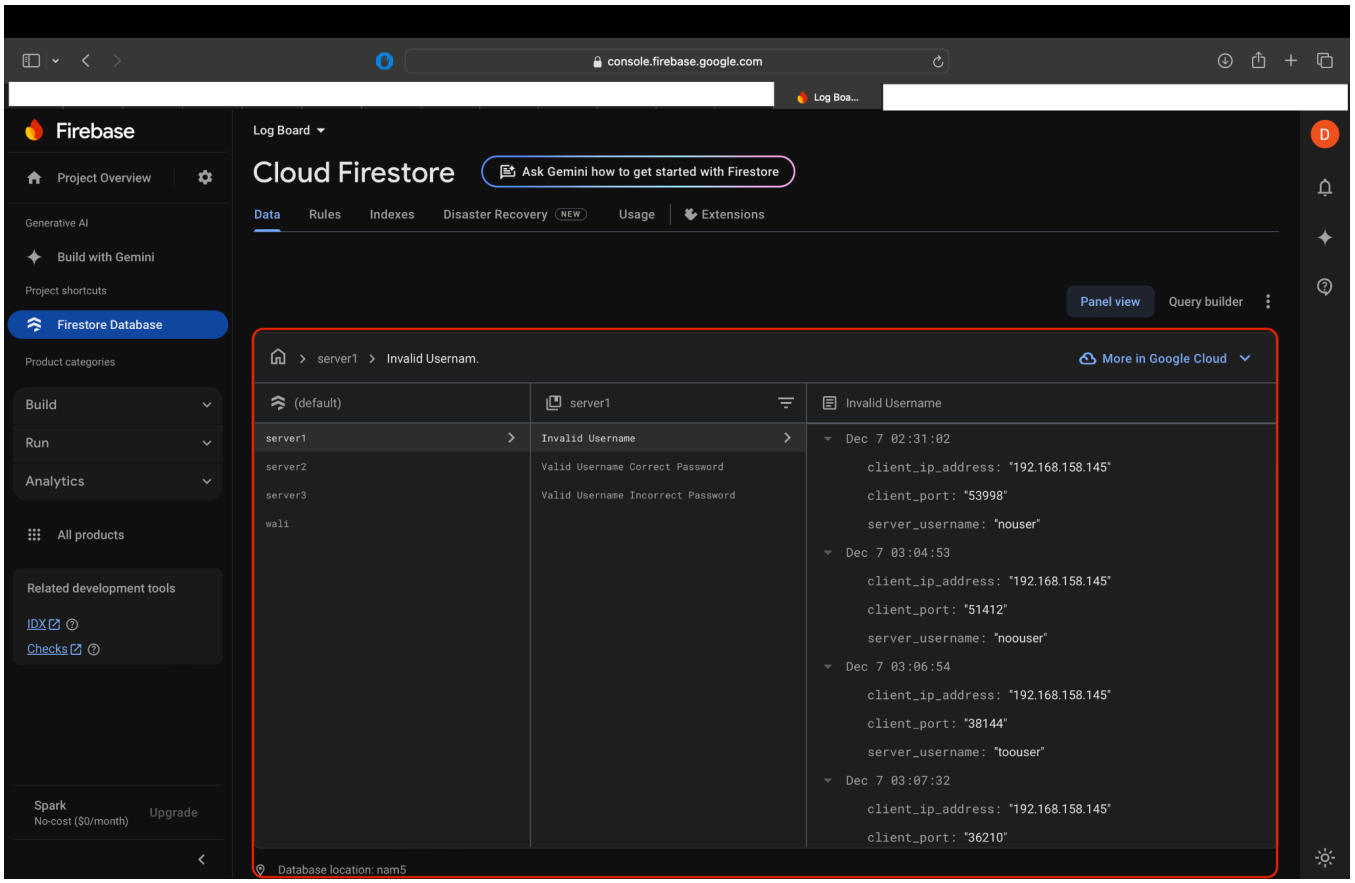
5. FIRESTORE DATABASE

Cloud Firestore is a scalable, NoSQL database provided by Firebase and Google Cloud. It organizes data into ***documents*** stored within ***collections***, allowing for flexible and hierarchical data structures, including nested objects and subcollections. Firestore supports real-time synchronization, ensuring that data stays updated across devices. Its powerful querying capabilities allow developers to filter, sort, and analyse data efficiently. Firestore is serverless, integrates with Firebase and Google Cloud products, and is designed to scale automatically with features like multi-region replication and strong consistency guarantees.

For this project, it stores data from three different "slave" servers in different tabs. Further for each server, the data is further bifurcated into three conditions:

1. **Invalid Username**
2. **Valid Username Correct Password**
3. **Valid Username Incorrect Password**

This hierarchical structure of stored data allows for easy analysis of log data. It shows the recorded entry of timestamps, Client IP address, client port and attempted server usernames. These data analysis play a crucial role for security strategies of servers.



6. DEVIATION FROM INITIAL PROPOSAL

The project initially proposed Graylog as the centralized log management platform for aggregation, analyzing and visualization of SSH logs. However, as the project developed, we transitioned to using Firebase Firestore for storing and managing logs in a centralized manner. The deviation was based on several practical considerations, outlined below:

1. Ease of Integration

Firestore offers seamless integration with Python through the Firebase Admin SDK, simplifying the process of securely uploading and retrieving log data. The robust and developer-friendly API provided by Firestore allowed us to focus more on the core functionality of log parsing and categorization. Graylog requires significant infrastructure to function effectively, including dependencies like Elasticsearch and MongoDB. Setting up and maintaining this environment demands additional time, resources, and expertise. Firebase Firestore, being a cloud-hosted NoSQL database, eliminates the need for complex setup and maintenance.

2. Real-Time Capabilities

Firestore provides real-time database features, enabling instant updates and retrieval of data. This functionality aligns well with the requirements of a log monitoring system, where timely access to logs is critical for detecting anomalies and responding to potential security threats. While Graylog supports near real-time log ingestion, its reliance on several dependencies can introduce latency in some configurations.

3. Scalability

As a fully managed NoSQL database, Firestore scales effortlessly with the volume of data, making it suitable for future expansion of the logging infrastructure. Scaling Graylog requires additional hardware or cloud instances to handle increased data throughput, adding to operational complexity.

4. Simplified Deployment

Firestore eliminates the need for on-premises setup and maintenance, as it is a cloud-based service. Unlike Graylog, which requires significant configuration and resource allocation for server deployment, Firestore's managed environment allowed us to deploy the system rapidly without the need for additional infrastructure. Graylog uses a schema that relies heavily on Elasticsearch indices, which can be complex for certain customized queries or integrations. Firestore’s flexible, document-based schema allowed us to structure logs hierarchically, grouping data by server, event type, and timestamp. This flexibility simplifies querying and makes the database easier to extend with additional features in the future.

The decision to transition from Graylog to Firestore was guided by a combination of practical and technical considerations. Firestore’s ease of use, real-time synchronization, scalability, and integration capabilities made it a superior choice for the project’s scope and objectives. While Graylog remains a powerful tool for enterprise-grade logging solutions, Firestore provided a more accessible and manageable alternative that aligned better with the project’s immediate and long-term goals. This deviation enabled the team to deliver a robust log monitoring solution while maintaining flexibility for future enhancements.

7. WORKING

1. **Control Node and Ansible Setup**:
   The control node is the central machine that runs Ansible and manages the entire deployment process. It securely communicates with the host nodes over SSH, allowing it to automate the installation and configuration of the necessary software. Ansible is installed on the control node, and it is used to define and execute tasks across the target nodes. The control node pulls the log collection tool from a Personal Package Archive (PPA) repository and ensures it is deployed to the host nodes.

2. **PPA Deployment from Launchpad**:
   The Personal Package Archive (PPA) is a custom software repository hosted on Launchpad, which contains the log collection tool. This tool is designed to collect logs from various sources on the host node (e.g., system logs, application logs) and store them in a cloud-based Firebase database. Using Ansible, the control node automatically installs the PPA repository on each host node, updates the package cache, and installs the required log collection software. This process ensures that the log collection tool is consistently deployed across all nodes, maintaining the same configuration and version.

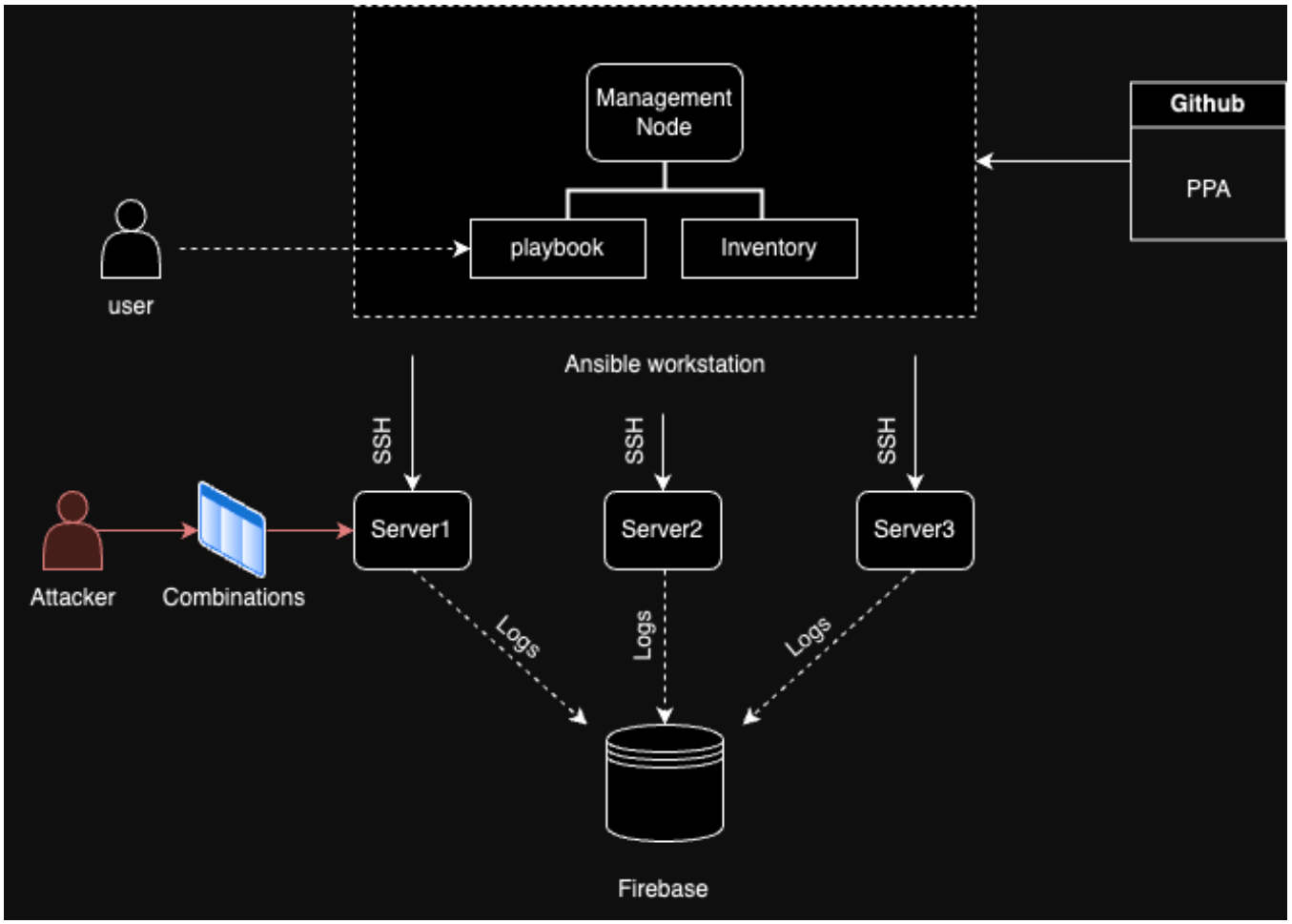
3. **Log Collection Process**:
   After the log collection tool is installed on the host nodes, it begins the process of collecting logs. The log collection tool targets specific system log files and stores them locally on the node. The logs are time stamped and saved in a specific directory to ensure proper organization and easy retrieval. This tool collects logs in real-time or at periodic intervals, depending on the configuration.

4. **Firebase Integration for Log Storage**:
   The collected logs are then sent to Firebase, a cloud-based database, where they are stored securely and can be accessed for further analysis. Firebase provides a scalable and efficient platform for storing large amounts of data, making it ideal for log management. The log collection tool integrates with Firebase to push logs in real-time, ensuring that they are available for monitoring and troubleshooting. Storing the logs in Firebase also provides the benefit of centralized data management, allowing logs from multiple host nodes to be accessed from anywhere.

5. **Automated Deployment and Monitoring**:
   Once the PPA is deployed to the host nodes, Ansible ensures that all configuration and installation tasks are completed automatically, reducing the need for manual intervention. The cron job on each host node ensures that log collection runs at regular intervals without any downtime. The integration with Firebase ensures that logs are continuously backed up and securely stored in the cloud. Regular checks and monitoring ensure that the system is functioning correctly, and any issues can be quickly identified and resolved.

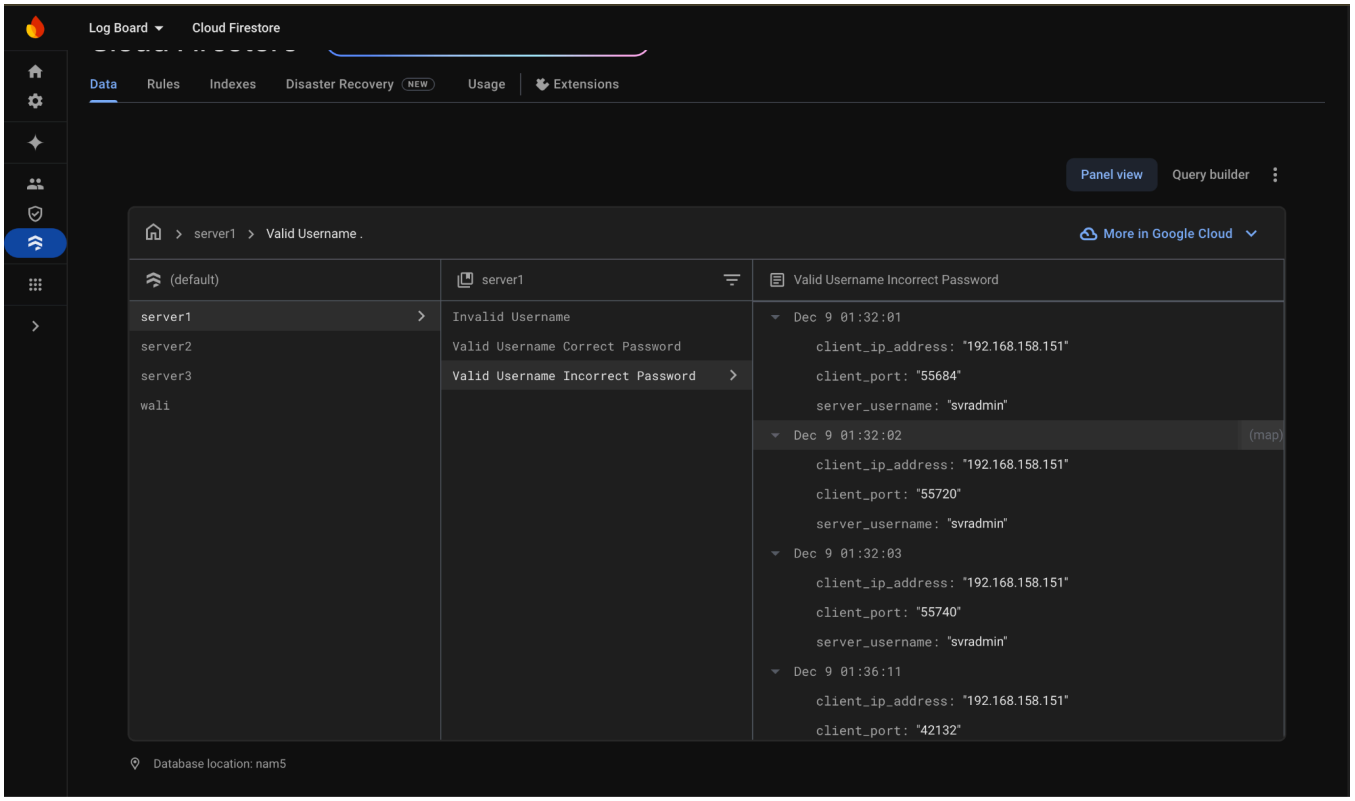
8. TESTING RESULTS

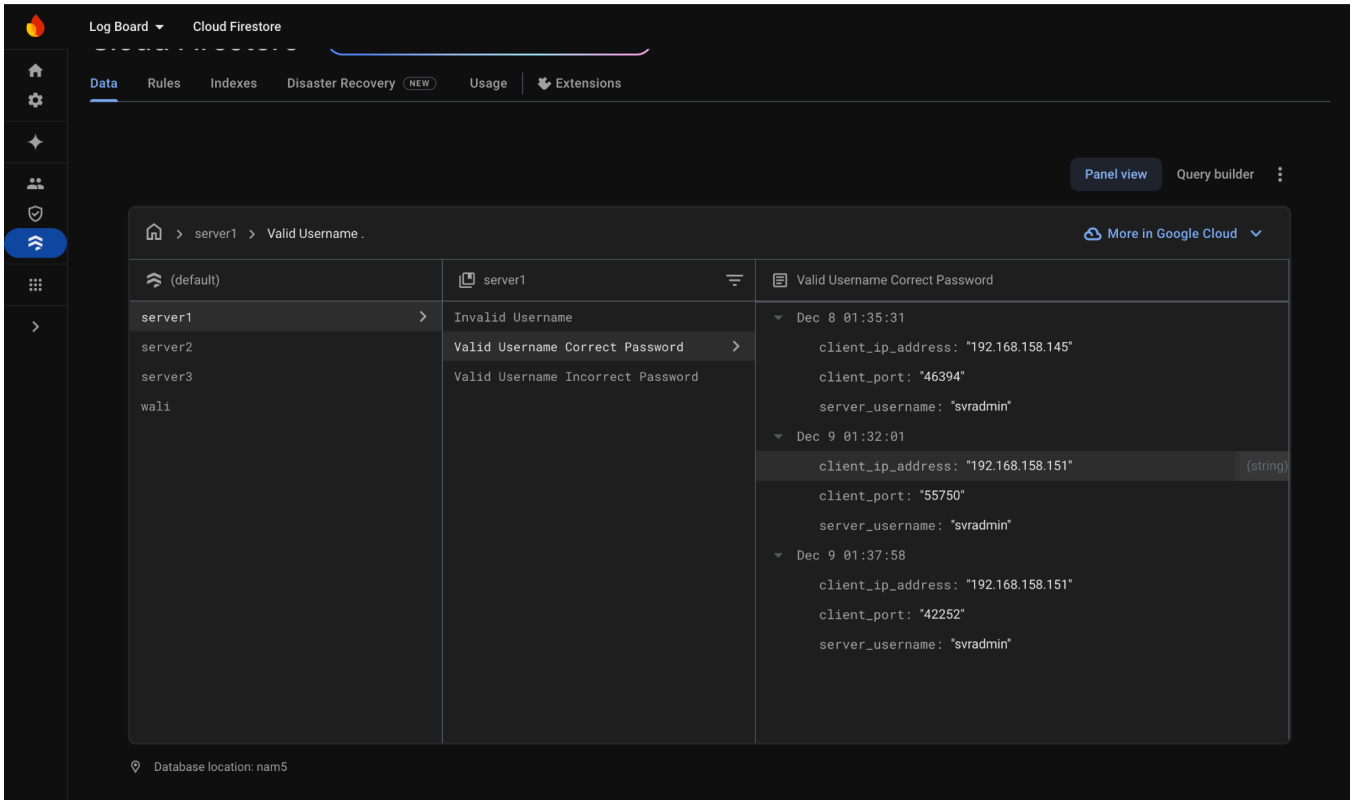
1. **Testing Environment:**
    - **Platform:** Testing was conducted on a Kali Linux system, known for its comprehensive suite of penetration testing tools.
    - **Setup:** A list of the three host servers (IP addresses) was prepared, along with separate files containing potential usernames and passwords.
2. **Brute Force Testing with Hydra:**
    - **Tool Used:** Hydra, a powerful tool for password cracking, was employed to perform brute force attacks on SSH authentication. Hydra is capable of quickly testing multiple username-password combinations to identify potential weaknesses.
    - **Methodology:**
        - A file containing the IP addresses of the three servers was created to target each server sequentially.
        - Separate files were created for possible usernames and passwords, simulating scenarios where credentials might be weak or predictable.
        - Hydra was executed with these inputs to attempt unauthorized access to the servers.



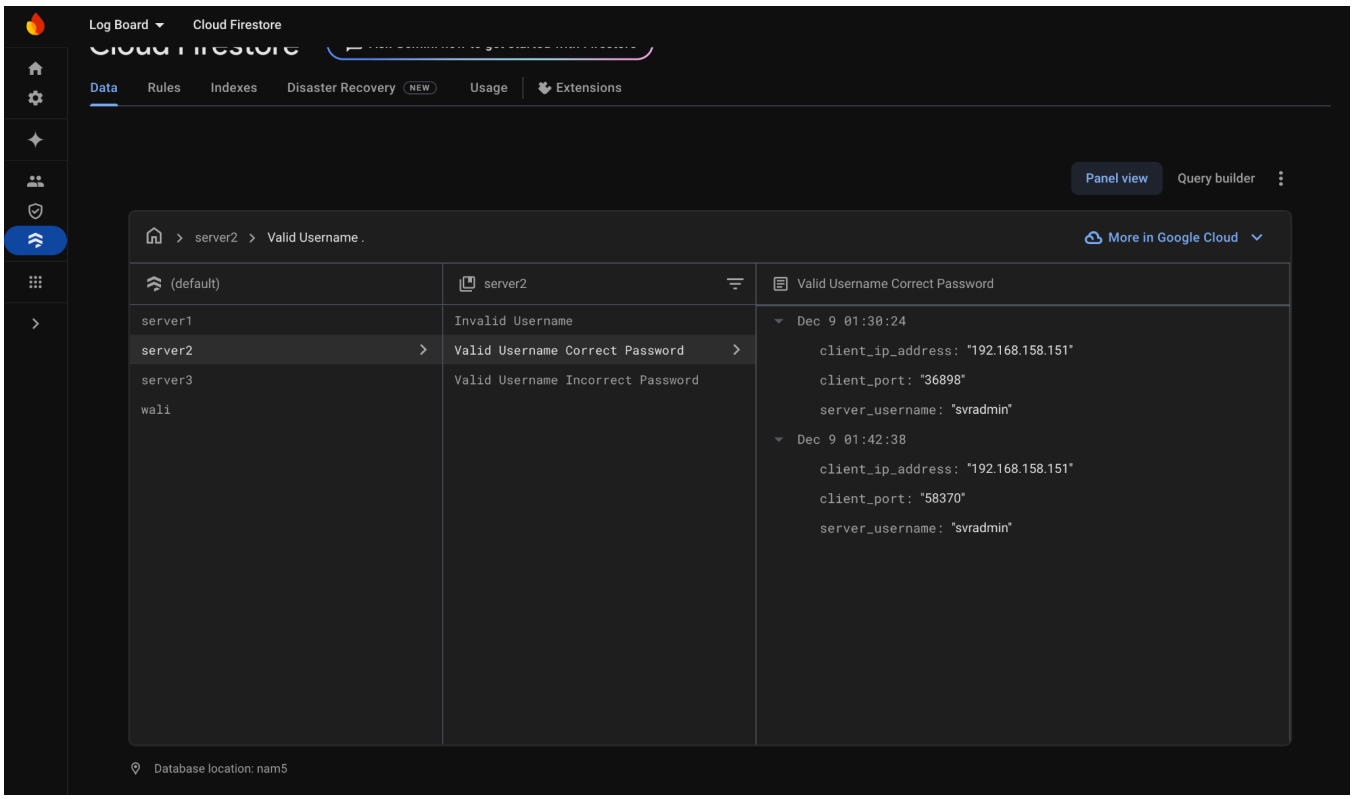
3. Results:

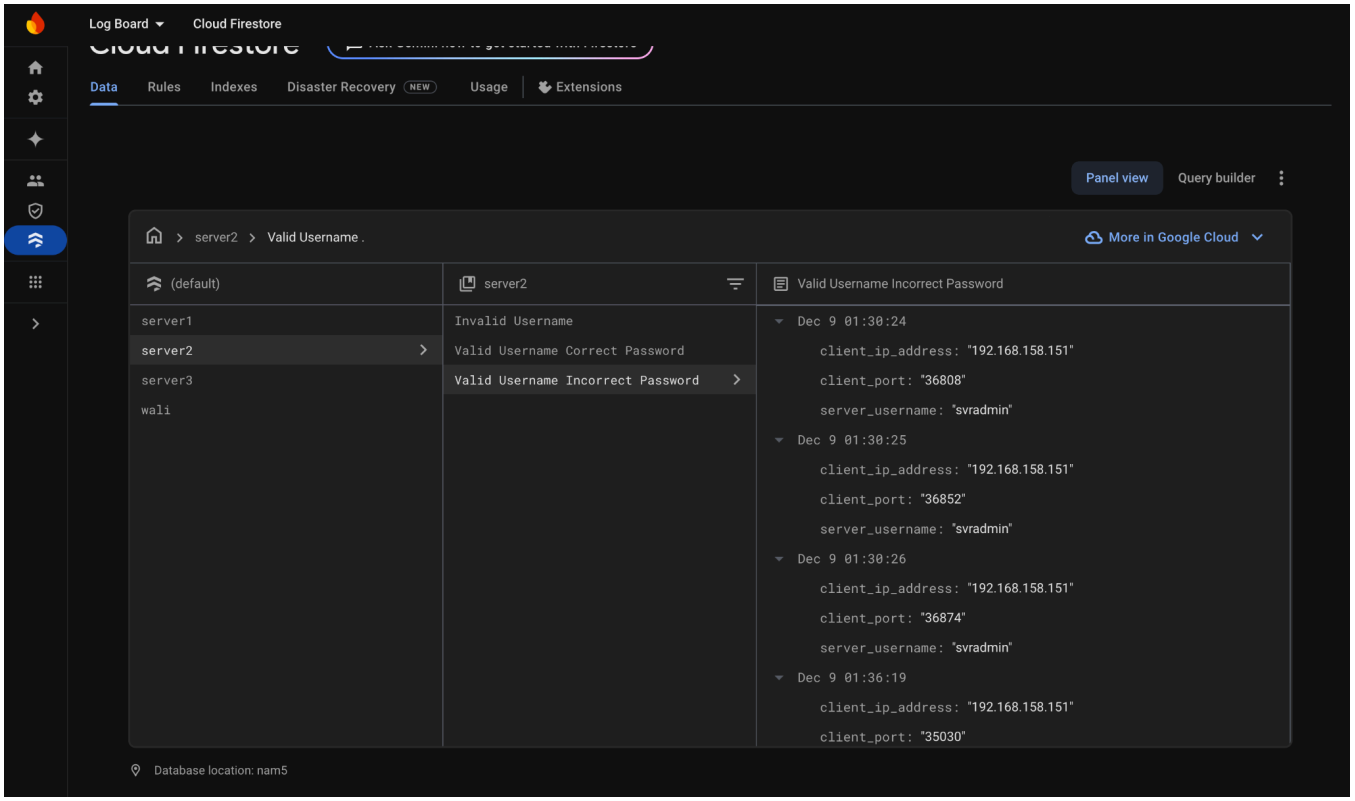
- **Server 1:**



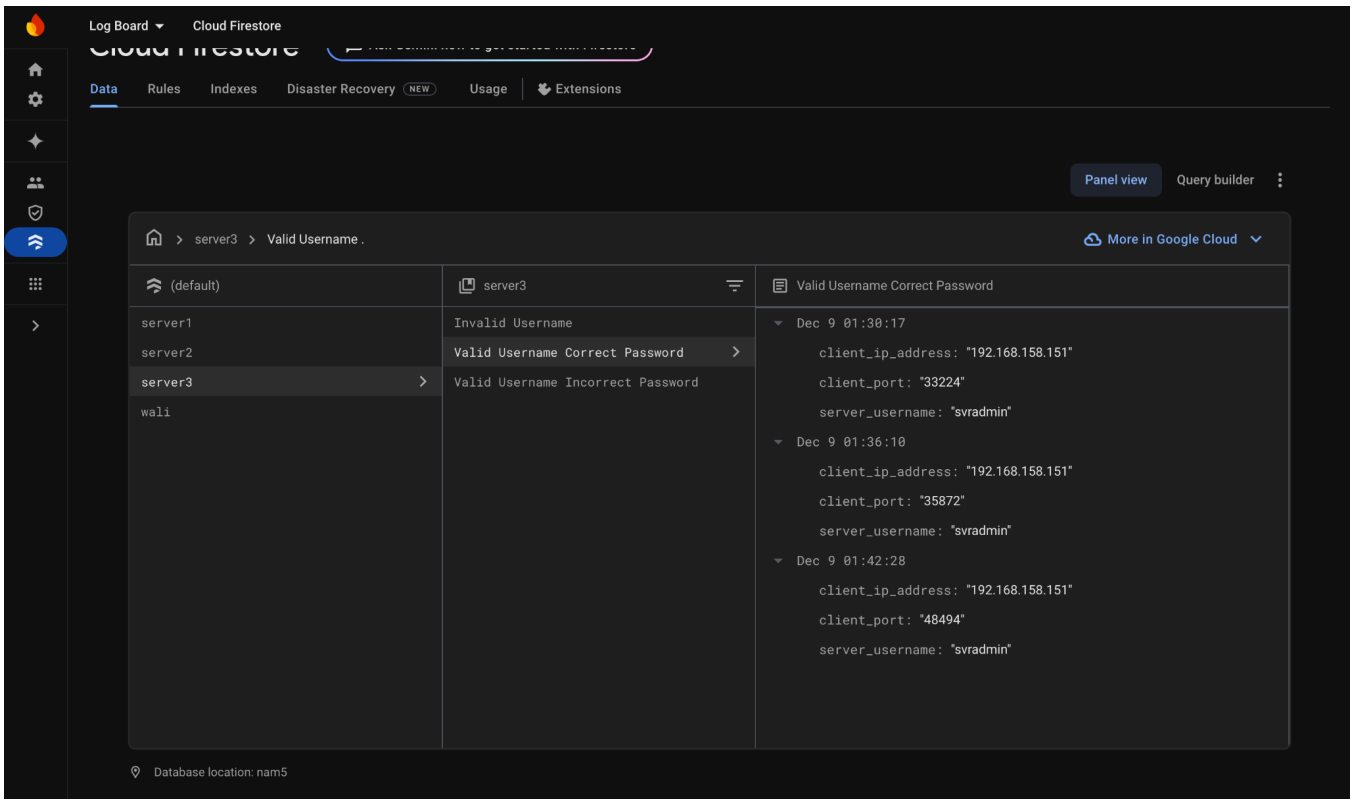


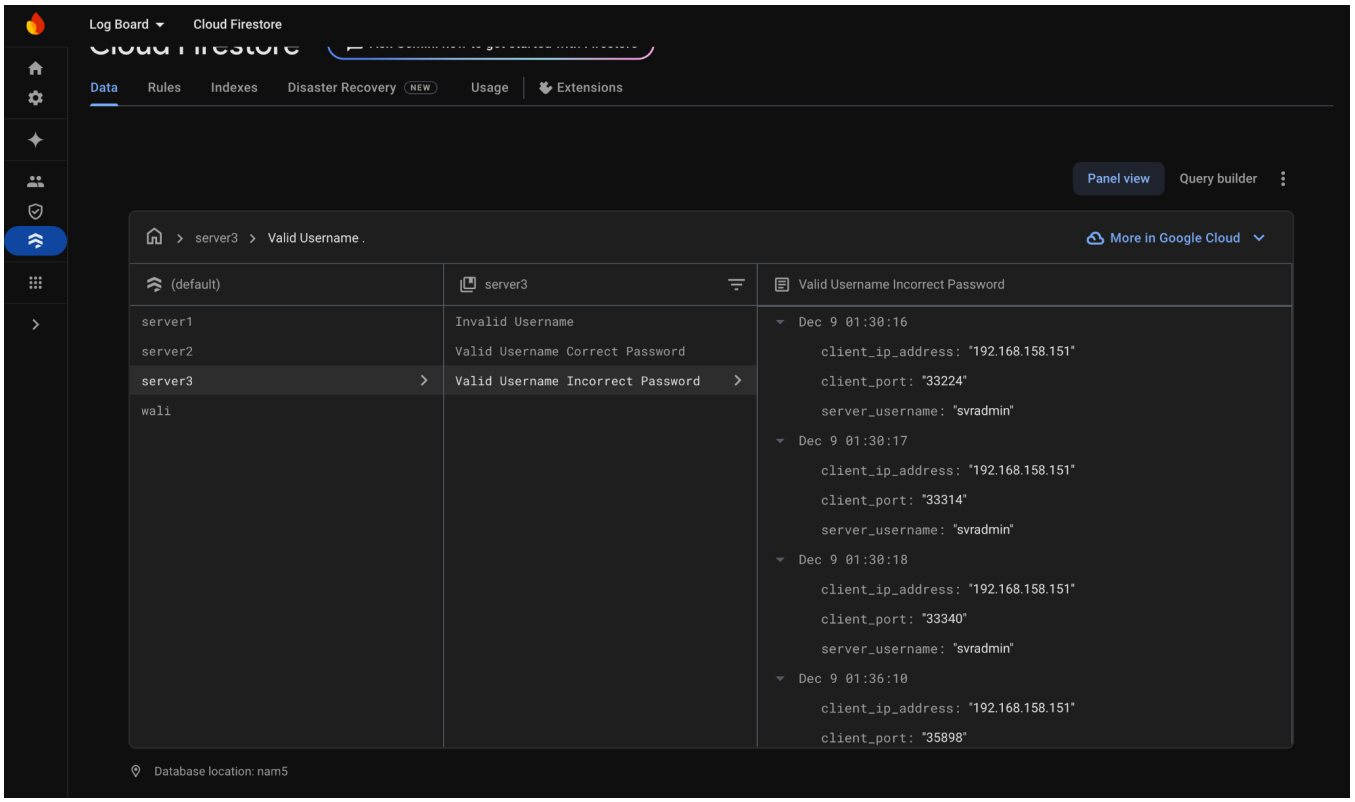
- **Server 2:**


Log Board
Cloud Firestore
Data
Rules
Indexes
Disaster Recovery
NEW
Usage
Extensions
Panel view
Query builder
server2
Valid Username .
More in Google Cloud
(default)
server2
Valid Username Correct Password
server1
server2
server3
wali
Invalid Username
Valid Username Correct Password
Valid Username Incorrect Password
Dec 9 01:30:24
client_ip_address: "192.168.158.151"
client_port: "36898"
server_username: "svradmin"
Dec 9 01:42:38
client_ip_address: "192.168.158.151"
client_port: "58370"
server_username: "svradmin"
Database location: nam5


Log Board
Cloud Firestore
Data
Rules
Indexes
Disaster Recovery
NEW
Usage
Extensions
Panel view
Query builder
server2
Valid Username .
More in Google Cloud
(default)
server2
Valid Username Incorrect Password
server1
server2
server3
wali
Invalid Username
Valid Username Correct Password
Valid Username Incorrect Password
Dec 9 01:30:24
client_ip_address: "192.168.158.151"
client_port: "36808"
server_username: "svradmin"
Dec 9 01:30:25
client_ip_address: "192.168.158.151"
client_port: "36852"
server_username: "svradmin"
Dec 9 01:30:26
client_ip_address: "192.168.158.151"
client_port: "36874"
server_username: "svradmin"
Dec 9 01:36:19
client_ip_address: "192.168.158.151"
client_port: "35030"
Database location: nam5

- **Server 3:**


Log Board
Cloud Firestore
Data
Rules
Indexes
Disaster Recovery
NEW
Usage
Extensions
Panel view
Query builder
server3
Valid Username .
More in Google Cloud
(default)
server3
Valid Username Correct Password
server1
server2
server3
wali
Invalid Username
Valid Username Correct Password
Valid Username Incorrect Password
Dec 9 01:30:17
client_ip_address: "192.168.158.151"
client_port: "33224"
server_username: "svradmin"
Dec 9 01:36:10
client_ip_address: "192.168.158.151"
client_port: "35872"
server_username: "svradmin"
Dec 9 01:42:28
client_ip_address: "192.168.158.151"
client_port: "48494"
server_username: "svradmin"
Database location: nam5


Log Board
Cloud Firestore
Data
Rules
Indexes
Disaster Recovery
NEW
Usage
Extensions
Panel view
Query builder
server3
Valid Username .
More in Google Cloud
(default)
server3
Valid Username Incorrect Password
server1
server2
server3
wali
Invalid Username
Valid Username Correct Password
Valid Username Incorrect Password
Dec 9 01:30:16
client_ip_address: "192.168.158.151"
client_port: "33224"
server_username: "svradmin"
Dec 9 01:30:17
client_ip_address: "192.168.158.151"
client_port: "33314"
server_username: "svradmin"
Dec 9 01:30:18
client_ip_address: "192.168.158.151"
client_port: "33340"
server_username: "svradmin"
Dec 9 01:36:10
client_ip_address: "192.168.158.151"
client_port: "35898"
Database location: nam5

9. FUTURE SCOPE

Our SSH Log Monitoring Personal Package Archive (PPA) is currently only compatible with Ubuntu 22.04 on AMD64 architecture. Hence, the PPA has limited platform and system compatibility, restricting its usability across diverse environments. It has significant potential for enhancement and broader applicability. The project could extend to these key future directions:

1. **System Compatibility Expansion**: Supporting additional architectures (e.g., ARM64) and other Ubuntu versions for greater adoption.
2. **Real-Time Alerts**: Implementing notifications via email or messaging platforms for critical log events.
3. **AI-Driven Analytics**: Leveraging machine learning to detect anomalies and predict potential issues.
4. **Cross-Platform Support**: Adapting the solution for other operating systems like CentOS and RHEL.
5. **Security Enhancements**: Adding encryption for logs and compliance reporting for industry standards.
6. **Scalability**: Optimizing database performance and introducing load balancing for larger infrastructures.

These advancements will transform the project into an even more robust, scalable, and versatile log monitoring solution for diverse environments.

10. CONCLUSION

The project successfully delivered a scalable and efficient SSH log monitoring solution tailored for Ubuntu environment 22.04. By integrating a Personal Package Archive (PPA) for simplified deployment with Firebase for centralized log storage, the system offers a robust approach to managing and visualizing authentication logs. The deployment of Ansible for automation ensured streamlined configuration across multiple nodes, enhancing reliability and consistency.

Key outcomes include:

1. **Simplified Deployment**: The PPA allows for easy installation and updates, reducing complexity for users managing multiple servers.
2. **Centralized Log Management**: Firebase's real-time synchronization and scalability enabled effective storage and analysis of authentication logs.
3. **Enhanced Security**: By categorizing log data into successful logins, failed attempts, and invalid username events, administrators can identify potential threats and respond proactively.
4. **Automated Monitoring**: The use of Ansible ensures continuous and reliable operation with minimal manual intervention.
5. **Comprehensive Testing**: The testing phase validated the system's resilience and accuracy under various scenarios. Brute force attack simulations using tools like Hydra verified the log monitoring system's ability to identify and log unauthorized access attempts effectively. This critical validation step ensured the system's robustness and readiness for real-world deployment.

This project demonstrates how modern tools like PPA, Firebase, and Ansible can be integrated to create a scalable, secure, and user-friendly solution for server log monitoring.

11. REFERENCES

[1] https://askubuntu.com/questions/186276/where-are-all-the-major-log-files-located

[2] https://help.launchpad.net/Packaging/PPA

[3] https://launchpad.net/+help-registry/openpgp-keys.html

[4] https://docs.ansible.com/

[5] https://firebase.google.com/docs

[6] https://www.kali.org/tools/hydra/