National College of Ireland

Configuration Manual

MSc Research Project
MSc in CyberSecurity

Mahesh Gavhane
Student ID: 23111984

School of Computing
National College of Ireland

Supervisor: Prof. Khadija Hafiz

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Mahesh Gavhane
Student ID:	23111984
Programme:	MSc in CyberSecurity
Year:	2024-2025
Module:	MSc Research Project
Supervisor:	Prof. Khadija Hafiz
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Mahesh Gavhane
Date:	27th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mahesh Gavhane
23111984

1 Introduction

The purpose of this document is to provide the basic information regarding the Logic App cloud setup which had been taken place, required systems specification for this project and the tools like Wireshark for testing purpose, PROM and Jupyter Notebook for visualization. With an aim to increase security and execution of Secure File Transfer in multi cloud (Azure to AWS) and within cloud (Azure to Azure) with log analysis and process mining. This configuration manual is created for the one who wants to replicate the same project as it contains all the steps for setting up the Cloud environment to implementation steps along with all the snapshots which will assist the user. This manual has all the information such as from where all the tools downloaded with software versions, and the URLs are added for reference.

2 System Specification

This Project uses a combination of tools, all are mentioned below:

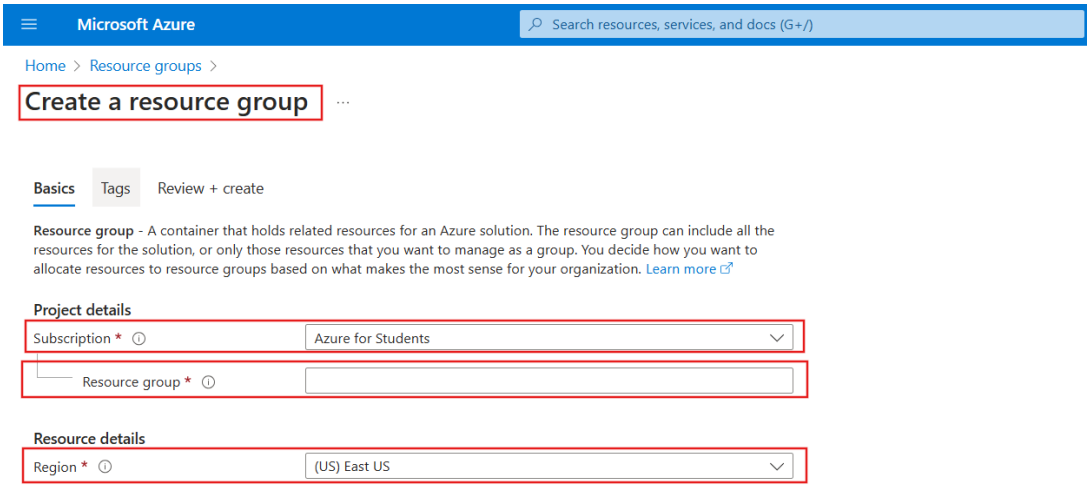
Tool Version	Version
Visual Studio	Visual Studio 2022 (64-bit) - Version 17.11.5
Postman	Version - 11.21.0
PROM Tool	PROM 6.14 with 64 bit JRE8
Jupyter Notebook	v7.2.3 version
Wireshark	Version 4.4.2
WINSCP	Version 6.3.5

Figure 1: List of tools used

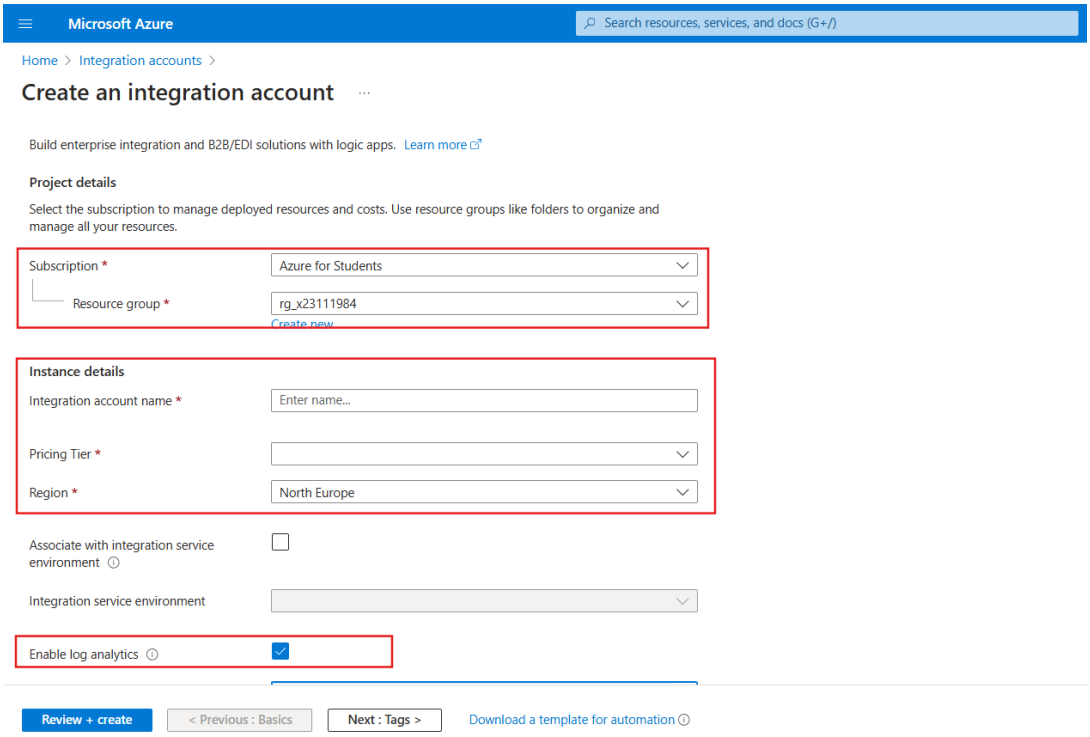
3 Environment Setups and Installation Steps Along with Scenario-based results

To execute the proposed solution, commence with Azure Environment Setup, AWS cloud setup, and install the tool required for executing the process, testing, and visualization.

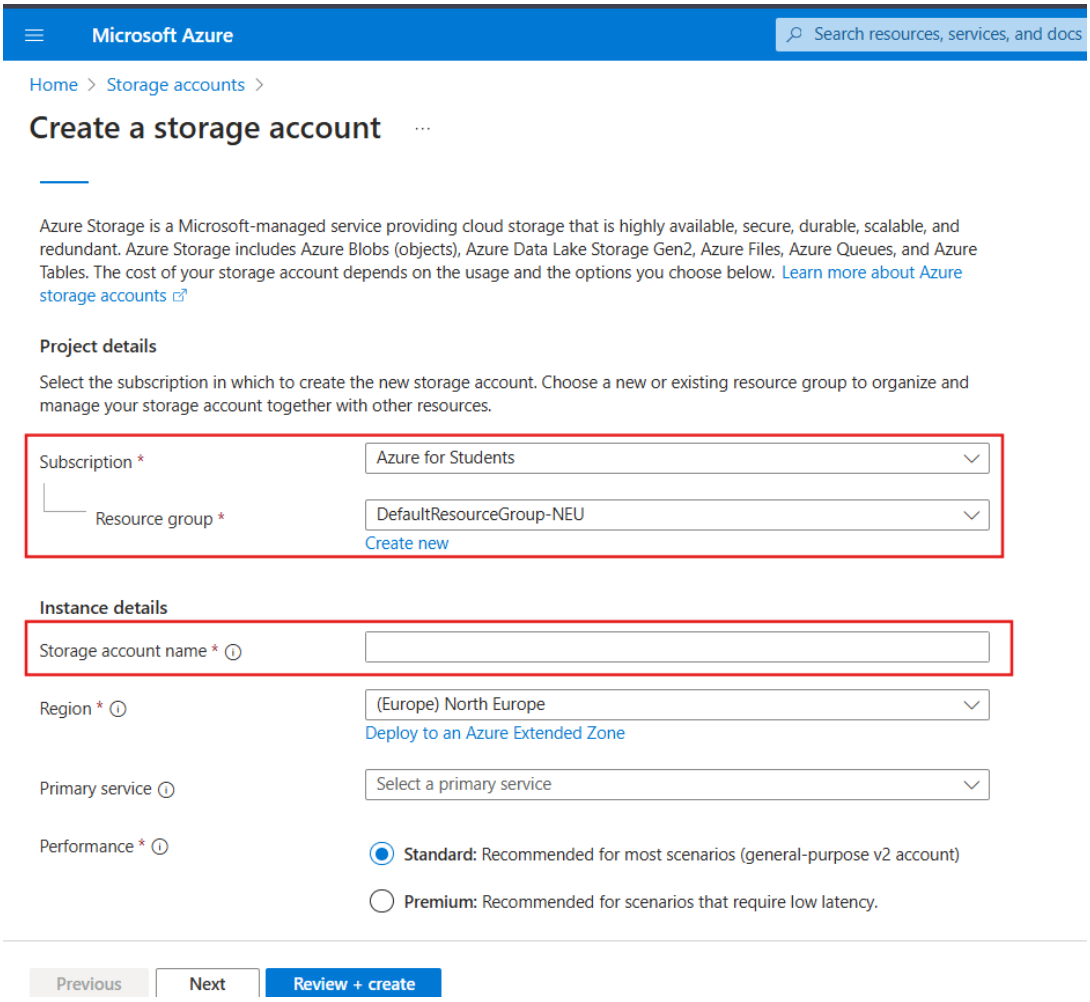
- Step 1: Create an Azure Resource Group with details like subscriptions name and the name of the Resource Group with Azure Region, for this project, North Europe has been selected as Azure Region.



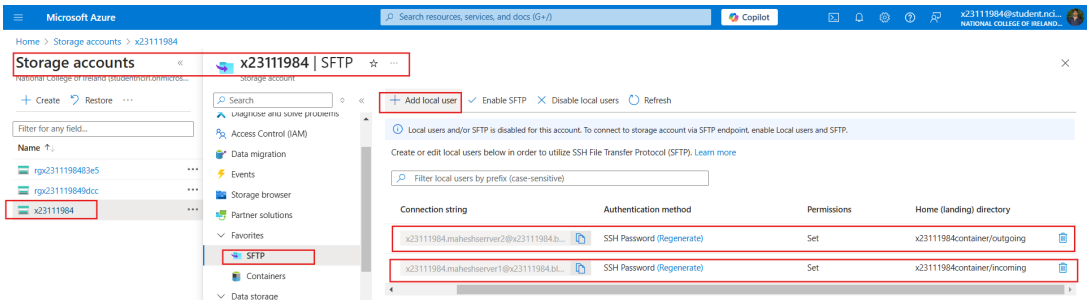
- Step 2: After Resource Group, create an Integration Account with the details like Subscription and select the resource group that is already created in Step 1. Give the name of the Integration Account select the pricing tier and Azure Region and enable the Log Analytics Workspace for further analysis.



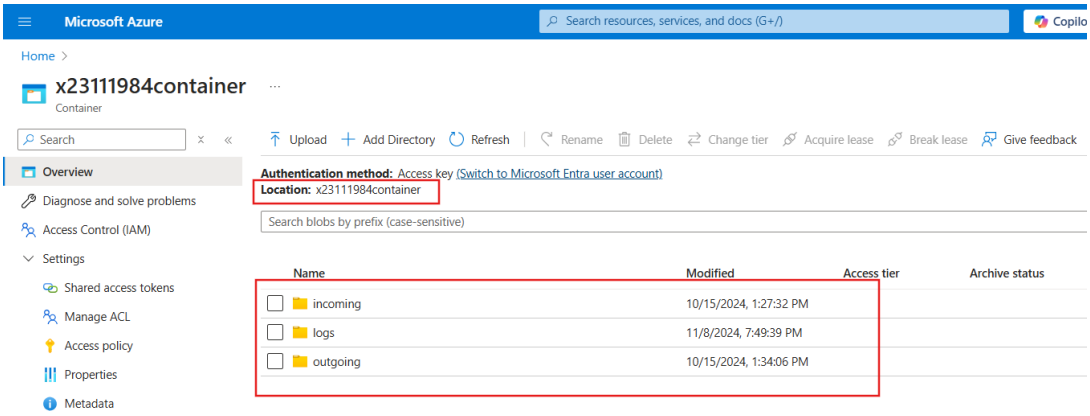
- Step 3: Create a Storage Account for hosting SFTP and setting up a Container for storing files, set up a Storage Account with a Subscription and existing Resource Group and give the proper Storage Account name.



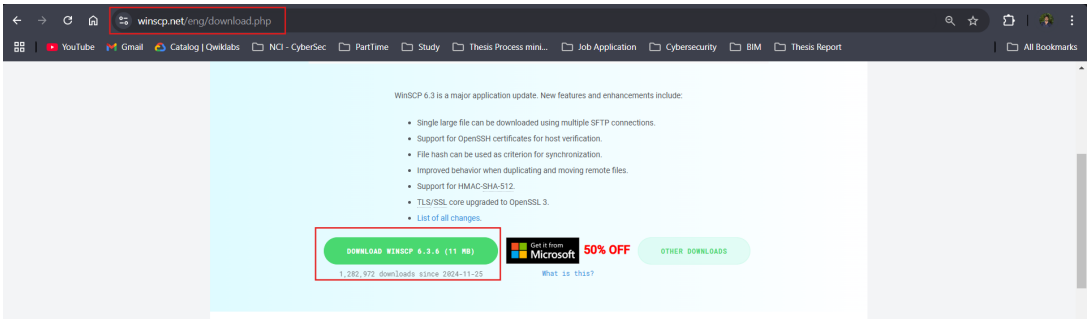
- Step 4: Set up SFTP server in Storage Account, Add 2 "Local User" and their "Authentication Method" and "Home Landing Directory". One is for the source directory and the second is for the destination Directory.

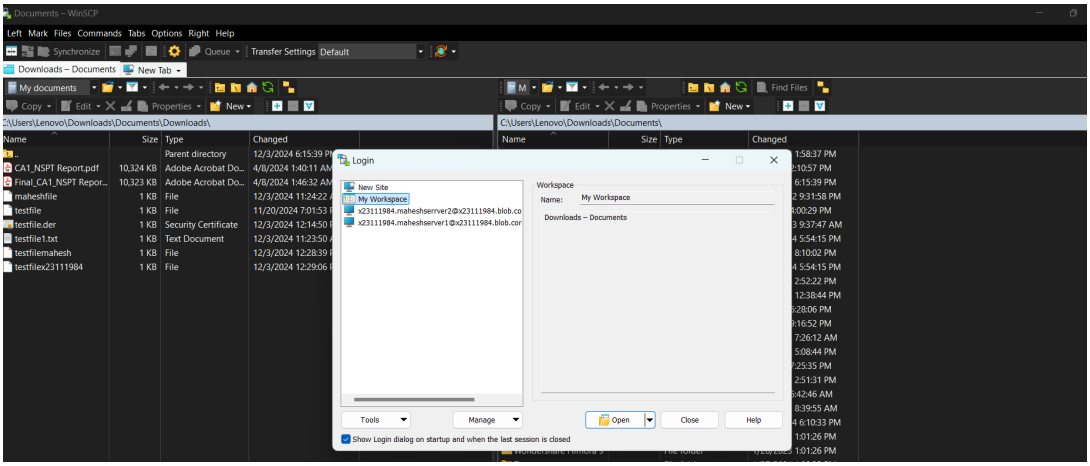


- Step 5: Setup Azure Container using a storage account for storing and processing purposes. After creating the container, create different directories for dropping and uploading the file in the source and destination directories. For this project, 3 directories have been created Incoming, Outgoing, and Logs for dropping, uploading and storing the generated logs.



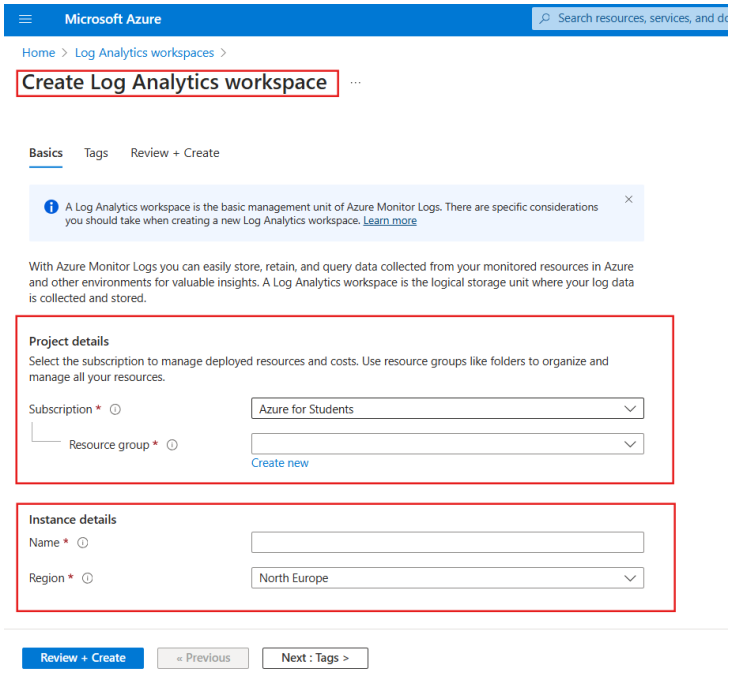
- Step 6: Install WinSCP to login to SFTP local user created in step 4 using credential and check if credentials are working properly and are able to login to WinSCP. [1]



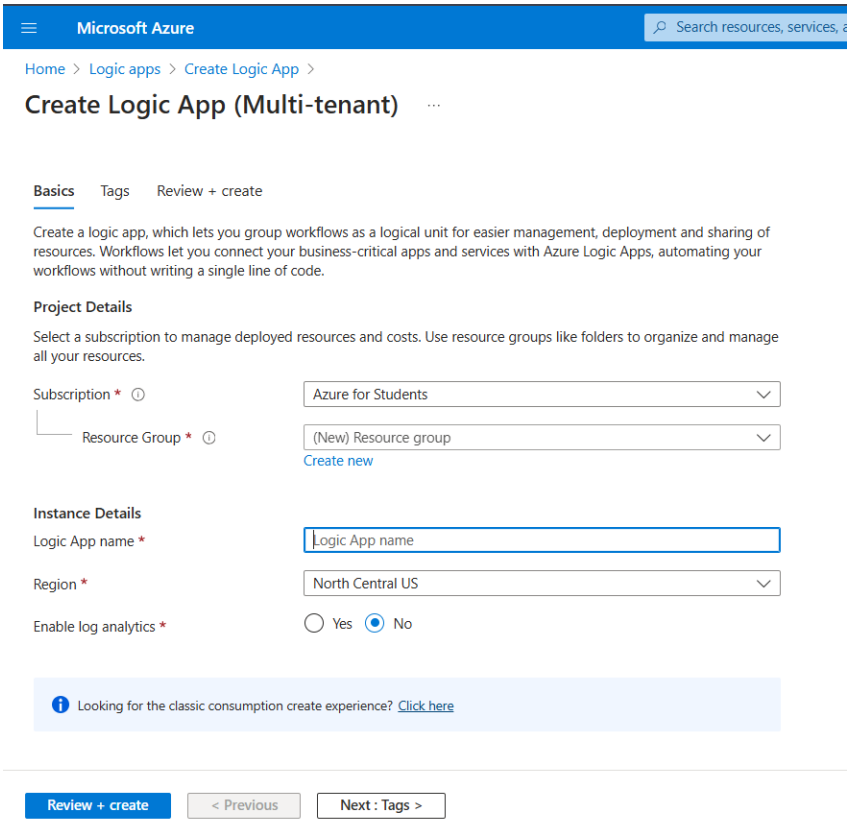


- Step 7: Create Azure Log Analytics Workspace and check Enable Log Analytics checkbox is ticked while creating the integration account. Give your workspace a proper name and select the same Resource Group, which is used while creating the Storage Account and Integration Account.

[1] https://winscp.net/eng/download.php

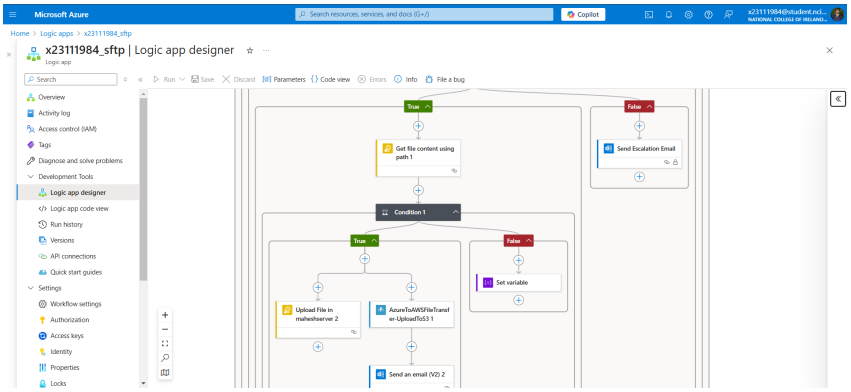


- Step 8: Search for Logic App in the Azure portal, and click on the Add option to create the New Logic App. While creating the LA, select a Hosting Option – Consumption. Use Subscription, Resource Group and give proper name to Logic App.

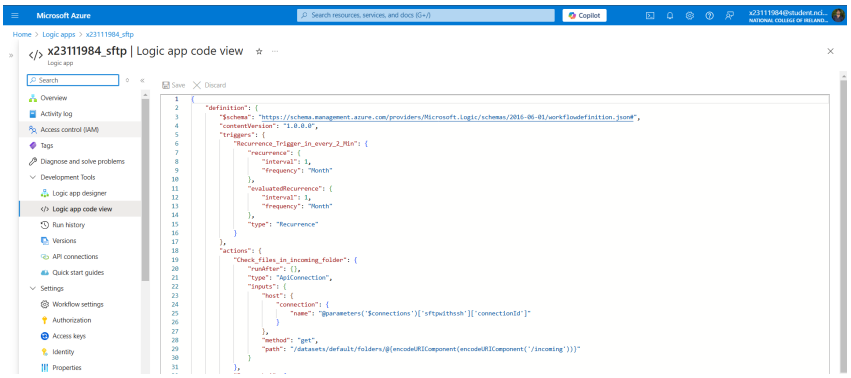


- Step 9: For this project two Logic Apps is created one is “x23111984_sftp” for Secure File Process within the cloud (AWS) and multi cloud (AWS). And another one is “x23111984_logcapture” for capturing the logs.

  – A. Logic App – “x23111984_sftp” is created by using designer and codeview. In the designer view click on add action to add action as shown below. For this project, the below flow has been created, if want to replicate the flow the same workflow. **Designer View**

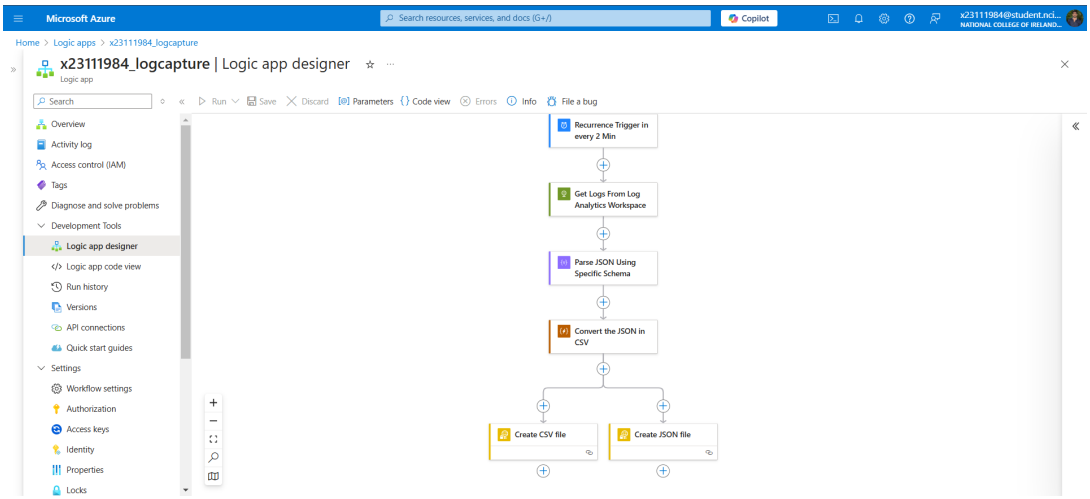


Code View

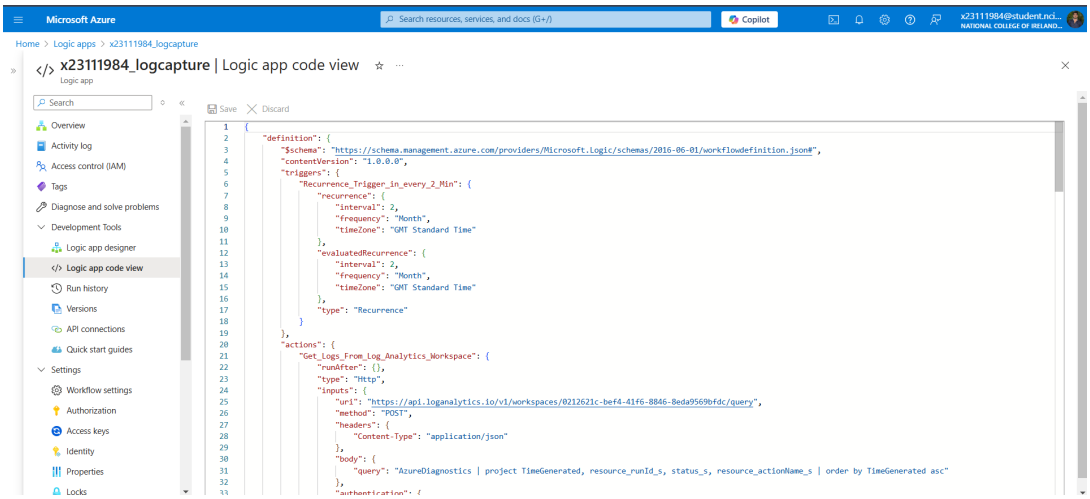


  – B. Logic App: “x23111984_logcapture” is created by using designer and codeview. Click on Logic App Designer and select an action to create a flow as shown below. For this project, the below flow has been created, if want to replicate the flow of the same workflow. This flow helps in capturing all logs and saving the generated logs in JSON and CSV format and uses the director created in the storage account to the logs in the “Logs Folder”.

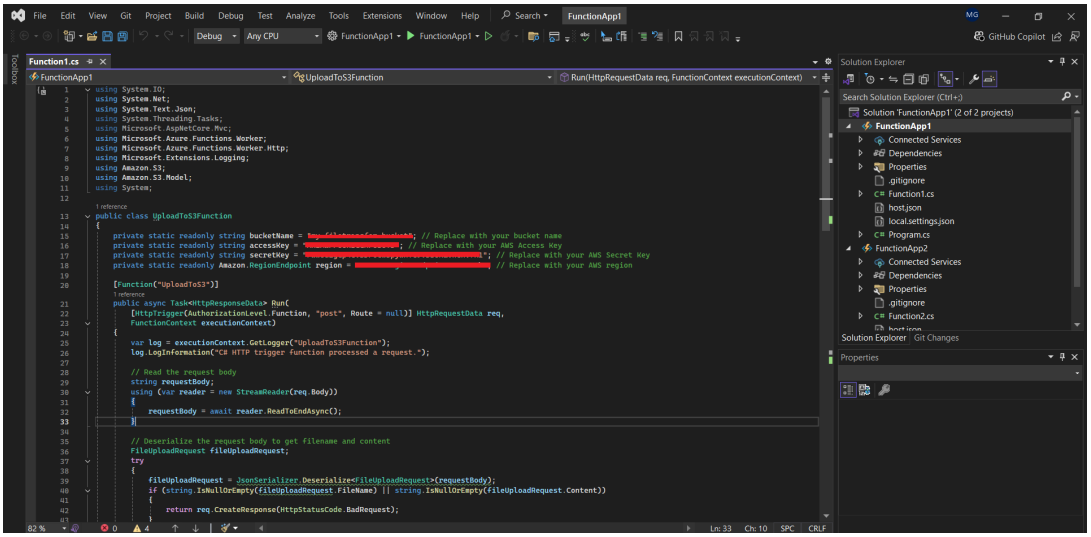
Logic App Designer View



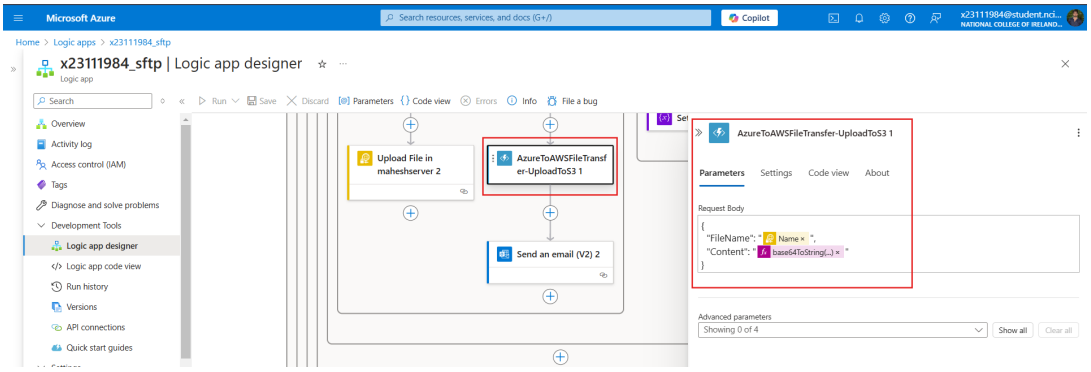
Logic App CodeView



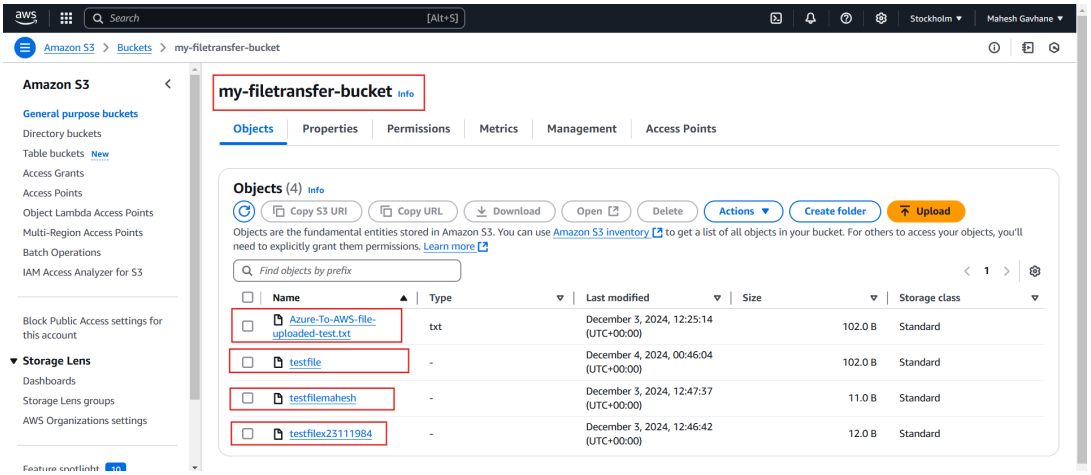
- Step 10: Search for Function App in Azure Portal, create Function App, give proper function name, select Runtime Stack, and make sure was to select the same resource group that created while setting up the Integration Account.
  Upload the created function in Azure Function App under the name of the created function.



Add Function App Action in Azure Logic App, select the created Function App, add the request body to execute, and call the function from the logic app only.

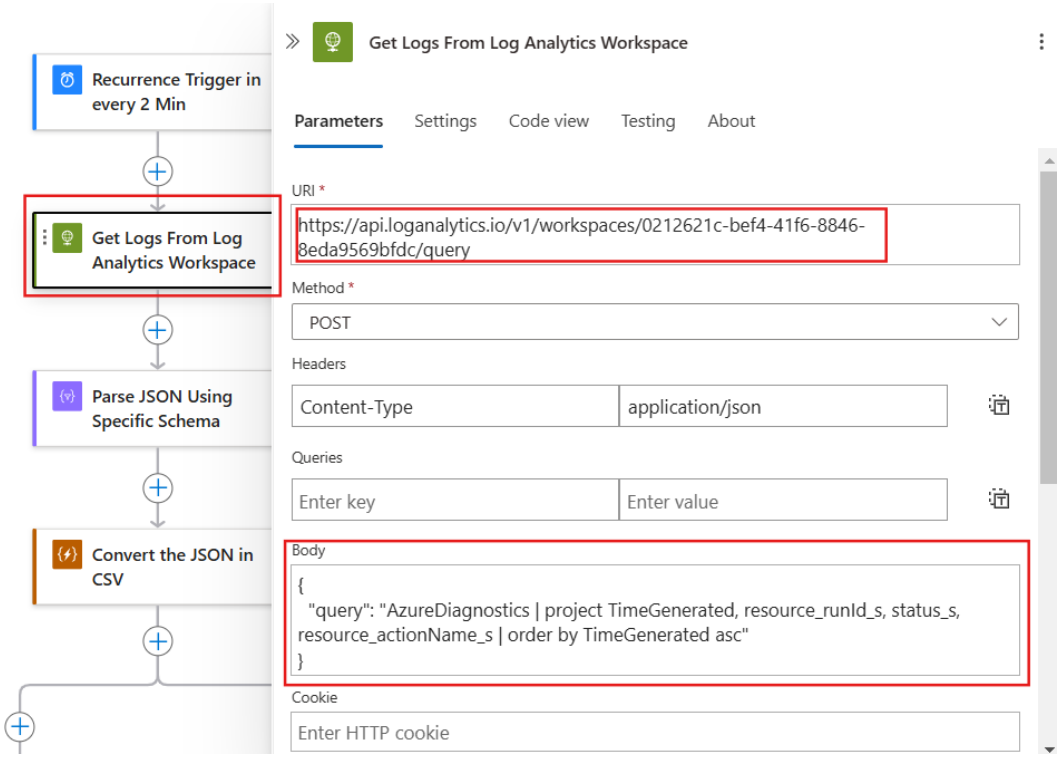


Setup AWS environment for storing processed file, create AWS S3 bucket with proper S3 bucket name. In this project, AWS S3 bucket is created to store the transmitted file which shows Azure to AWS workflow.

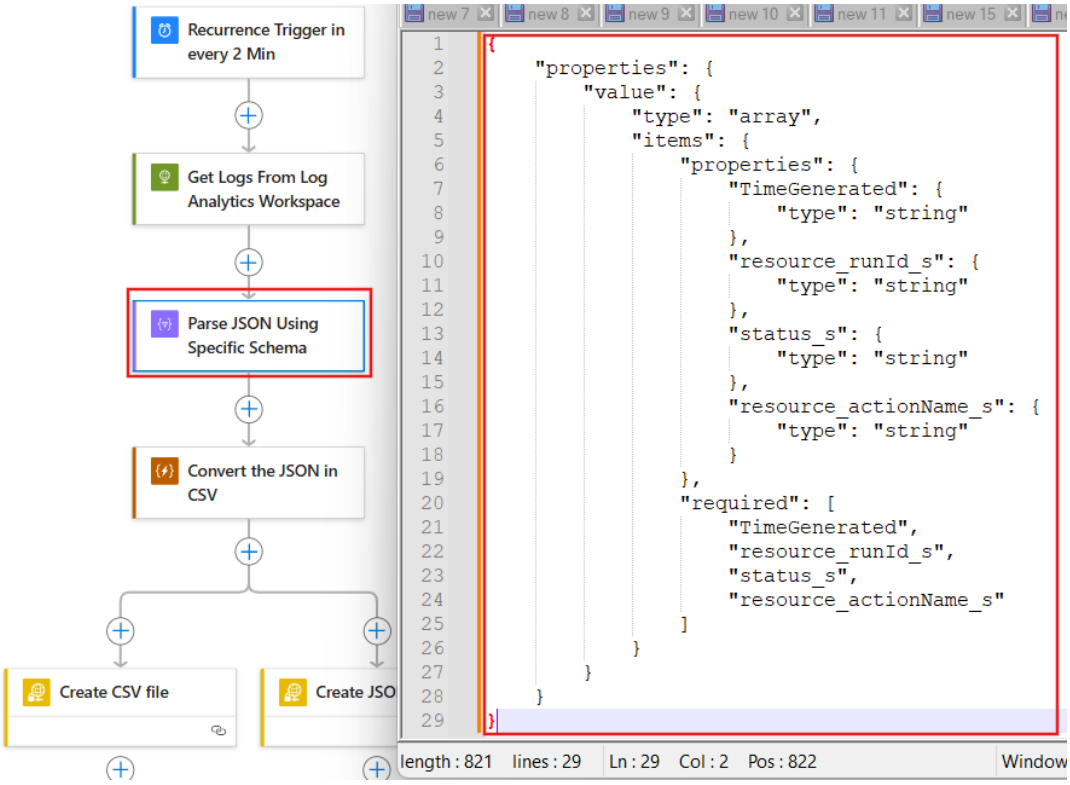


- Step 11: Using Logic App – "x23111984logcapture", generate the logs and once created, download them from Storage Account for further analysis. Below is the CSV structure generated by Logic App. To capture the logs from Log Analytics Workspace and to convert the required JSON format into CVS format KQL query, Schema, and JavaScript code have been used as shown below.

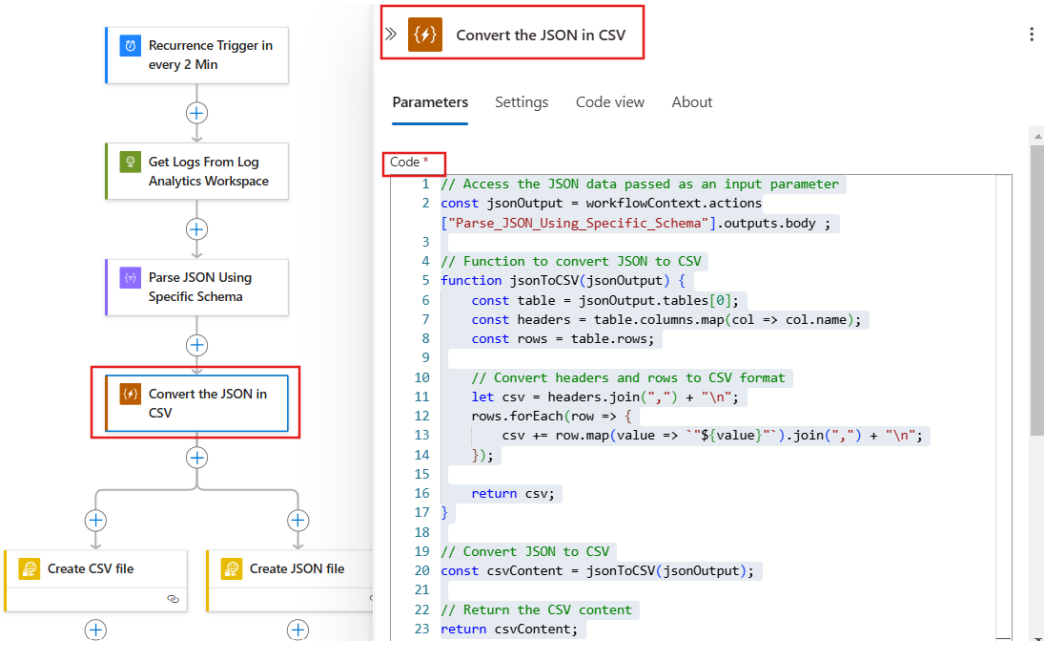
  – Use Action called "Log Analytics Workspace" with their URI and KQL query to capture the log from the workspace.



  – Use the action "Parse JSON" with the given schema to convert the JSON into format JSON format.

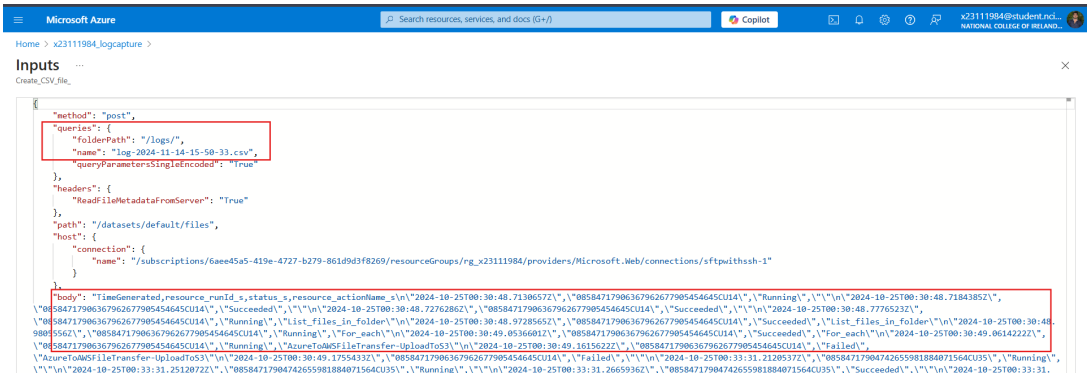


– For converting JSON to CSV format, use the "JavaScript" action with the code shown below.

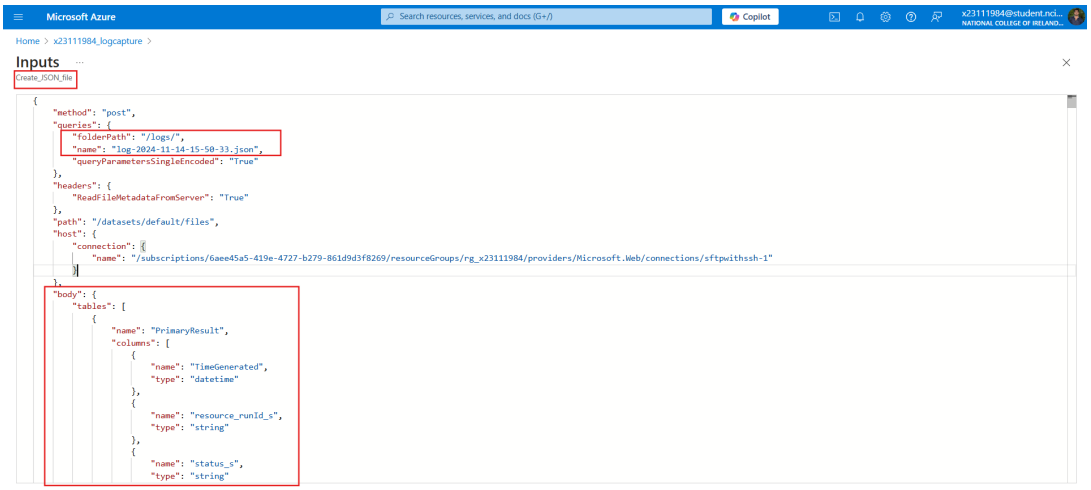


– After converting JSON to CSV, Logic App stores both 'logs.json' and 'logs.csv' in the storage account as shown below.

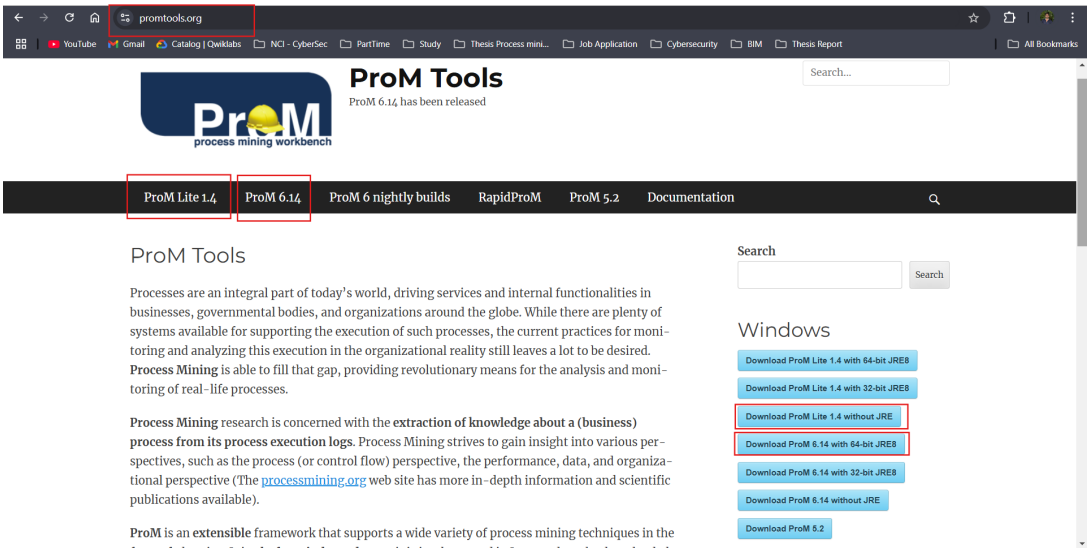
Result: Logic App generates Logs.csv



Result: Logic App generates Logs.json

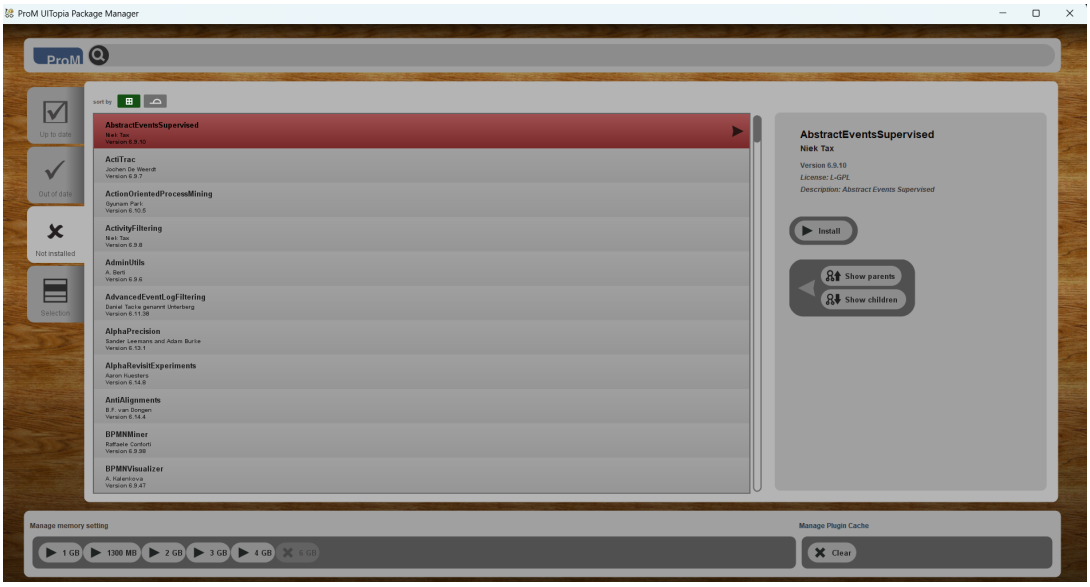


- Step 12: Install PROM tool (version 6.14) for process mining, from the official website. PROM lite 1.4 also works for executing the workflows.[2]



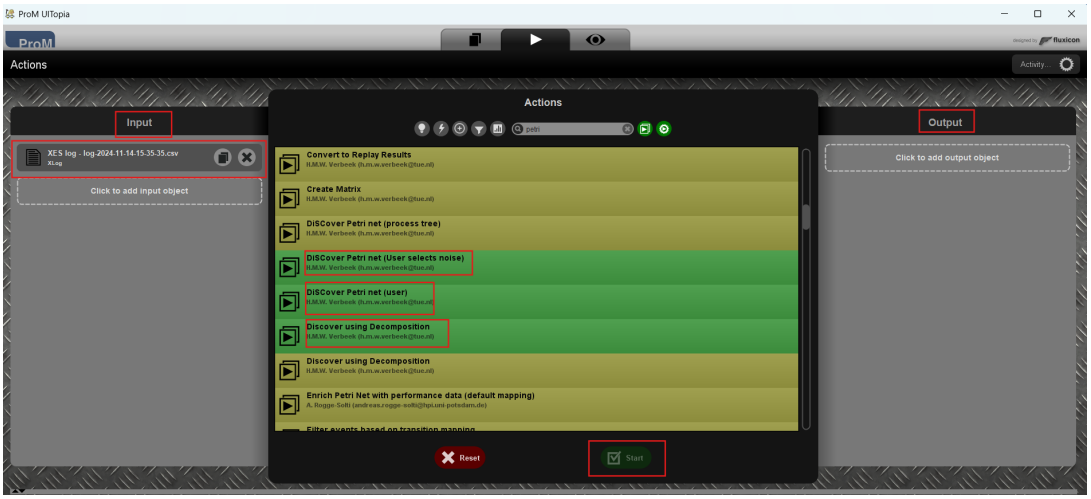
[2]https://promtools.org/

– a) Installing PROM tool, Open the Process Minig PROM tool package manager to import packages like CSV import and required packages based on Process Mining methods.

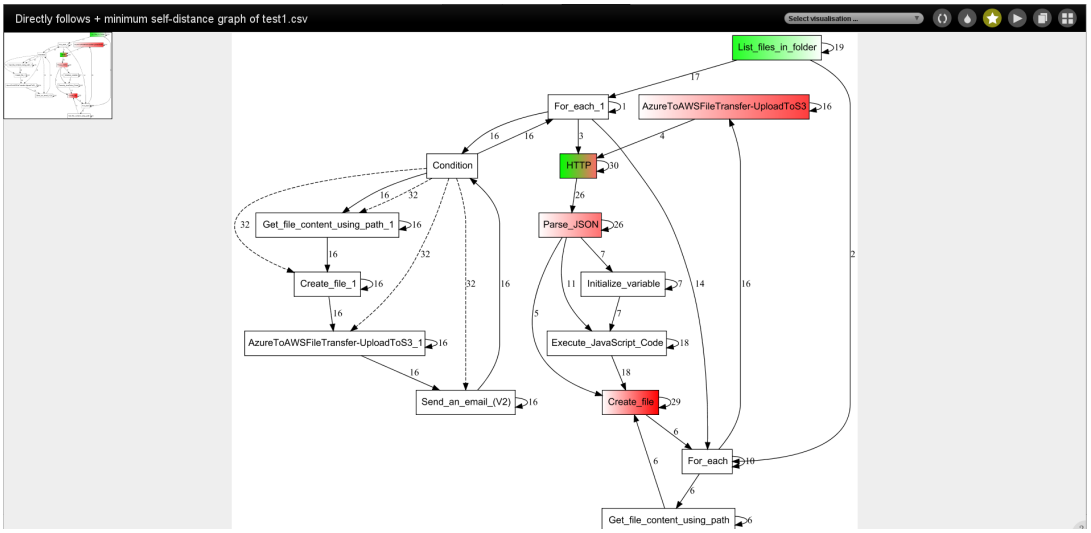


– b) By using generated logs.CSV from log capture Logic App, import that CSV file in the PROM tool and create Labelled Event Logs by using the action – "Infer Case ID", select the attribute and click on continue. The structure of Labeled Event Logs are shown below.

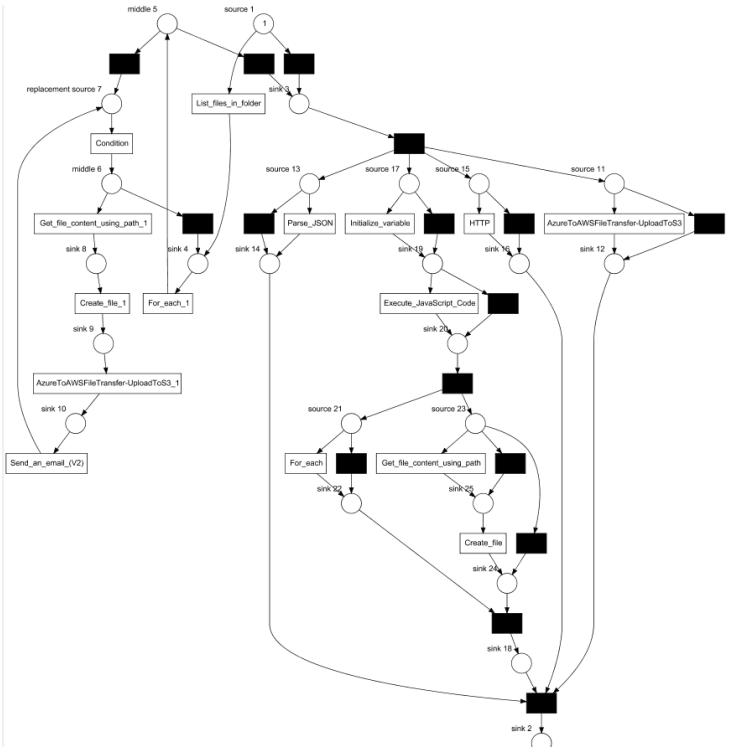
– c)Export log to workspace with name starts "XESLog" and using that created Event Log in input, apply the method which can create a graph for analyzing the workflow. For this project, we use the "Petri Net Flow method" and "Direct Follow method" as shown below.



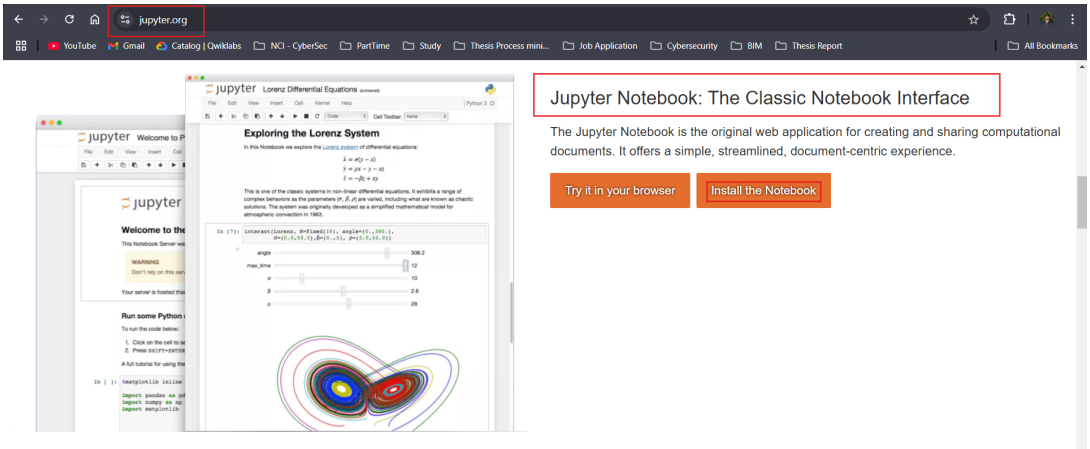
Direct Follow Graph :



Petri Net Flow Graph:



- Step 13: For visualization, install Jupyter Notebook from the official website, and use Python libraries for visualizing the graphs. [3]



  - a) Import Python libraries like “Pandas” and “Numpy” as shown below.

[3] https://jupyter.org/

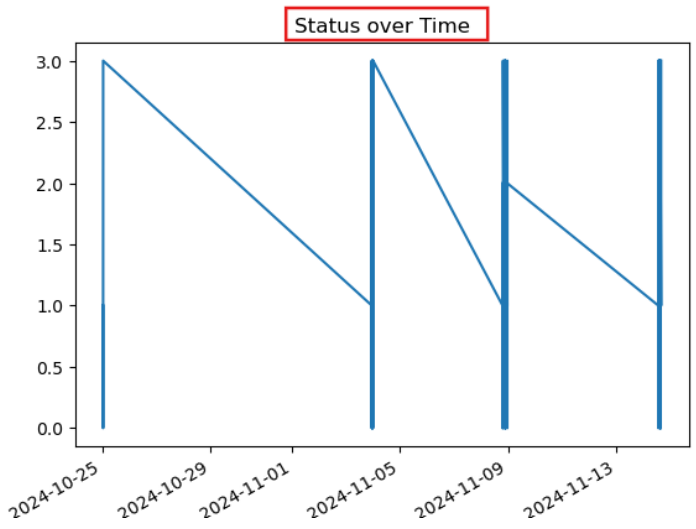
```
[210]: import pandas as pd
       import numpy as np
       from sklearn.preprocessing import LabelEncoder
       import matplotlib.pyplot as plt

[212]: data = pd.read_csv("C:\\Users\\Lenovo\\Desktop\\Thesis Part 2\\implementation\\log-2024-11-14-15-35-35.csv")

[214]: print(data.head())
```

```
                 TimeGenerated                  resource_runId_s   status_s  \
0  2024-10-25T00:30:48.7130657Z  0858471790636796267790545464 5CU14    Running
1  2024-10-25T00:30:48.7184385Z  0858471790636796267790545464 5CU14  Succeeded
2  2024-10-25T00:30:48.7276286Z  0858471790636796267790545464 5CU14  Succeeded
3  2024-10-25T00:30:48.7776523Z  0858471790636796267790545464 5CU14    Running
4  2024-10-25T00:30:48.9728565Z  0858471790636796267790545464 5CU14  Succeeded

  resource_actionName_s
0                   NaN
1                   NaN
2                   NaN
3  List_files_in_folder
4  List_files_in_folder
```

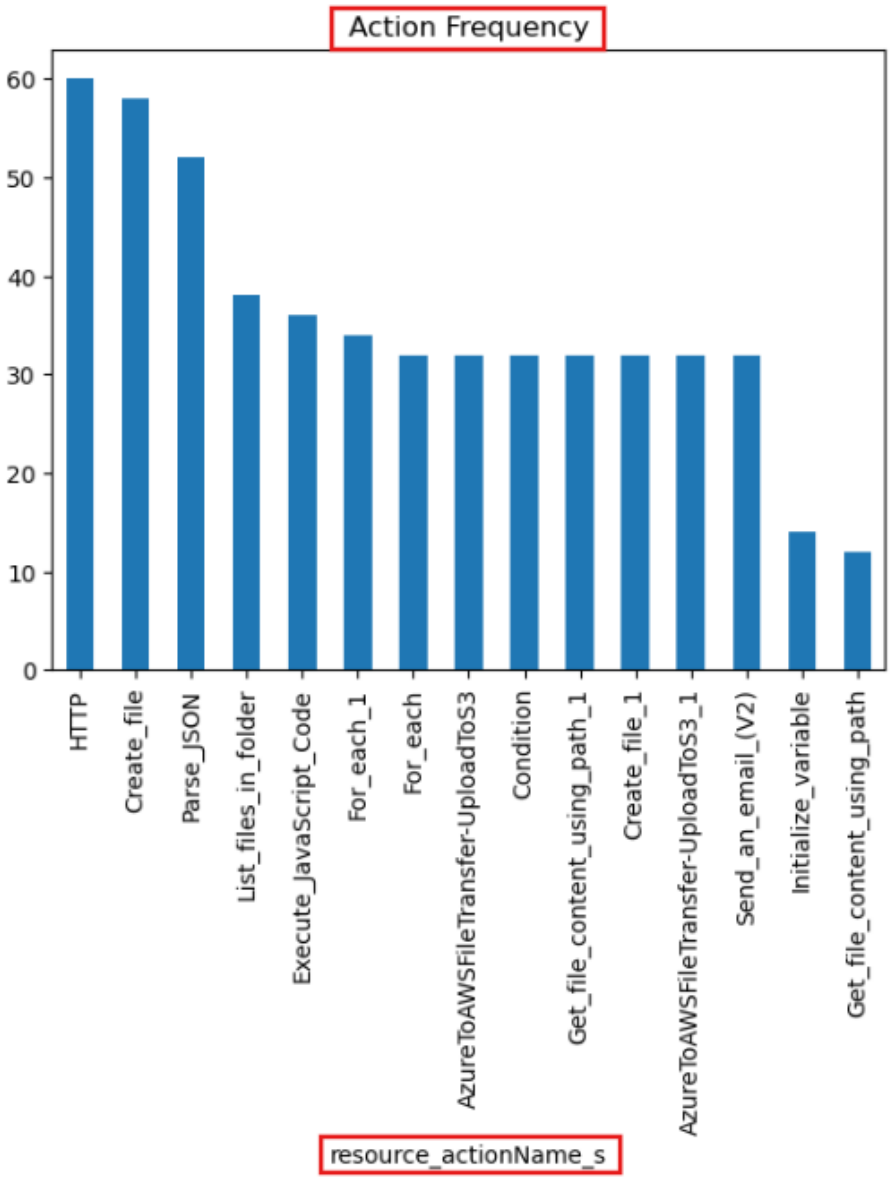
– b) After installing the jupyter and importing libraries, generate various graph which helps in analyzing the overall workflow and any irregularities and deviation in the graphs. For this project, various graphs has been generated as shown below.

```
# Calculate the duration for each `resource_runId_s`
data['duration'] = data.groupby('resource_runId_s')['TimeGenerated'].transform(lambda x: x.max() - x.min())

# Count action occurrences per run ID
action_counts = data.groupby('resource_runId_s')['resource_actionName_s'].value_counts().unstack(fill_value=0)

# Merge counts back to the dataset
data = data.merge(action_counts, on='resource_runId_s', how='left')
```

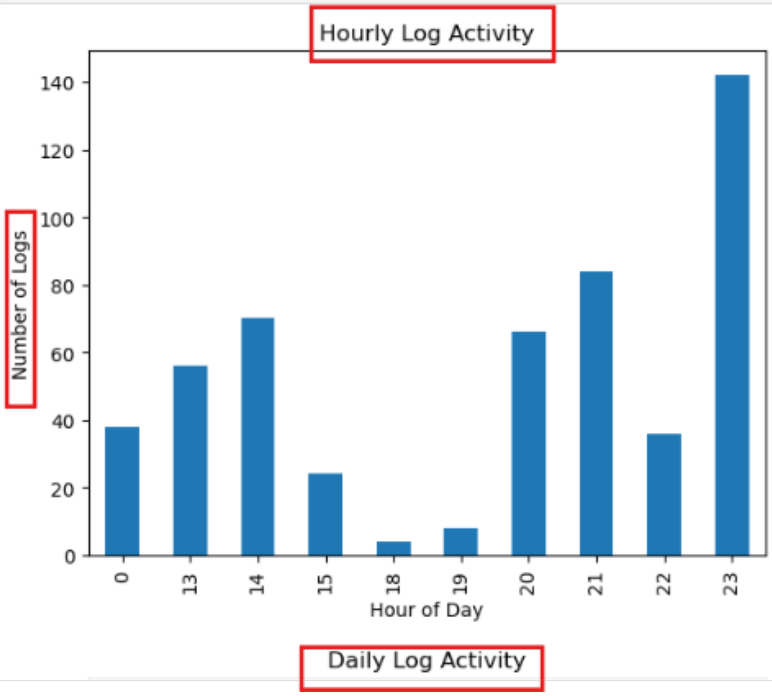
```
# Time-series analysis: Plot activity over time
data.set_index('TimeGenerated')['status_encoded'].plot(title='Status over Time')
plt.show()
```



```
# Action frequency distribution
data['resource_actionName_s'].value_counts().plot(kind='bar', title='Action Frequency')
plt.show()
```

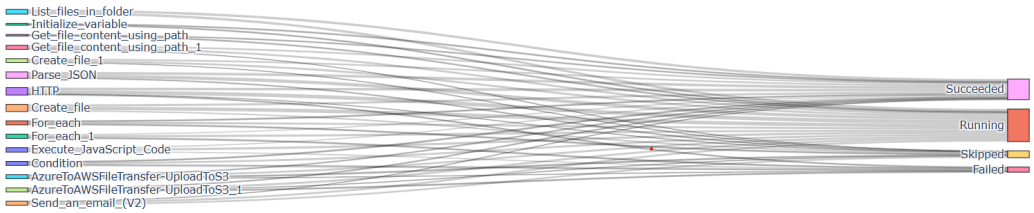


```python
# Extract hour and day from the `TimeGenerated` column
data['hour'] = data['TimeGenerated'].dt.hour
data['day'] = data['TimeGenerated'].dt.date

# Group by hour to detect hourly patterns
hourly_activity = data.groupby('hour').size()

# Plot hourly patterns
import matplotlib.pyplot as plt
hourly_activity.plot(kind='bar', title='Hourly Log Activity')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Logs')
plt.show()

# Group by day to detect daily activity patterns
daily_activity = data.groupby('day').size()

# Plot daily patterns
daily_activity.plot(title='Daily Log Activity')
plt.xlabel('Date')
plt.ylabel('Number of Logs')
plt.show()
```

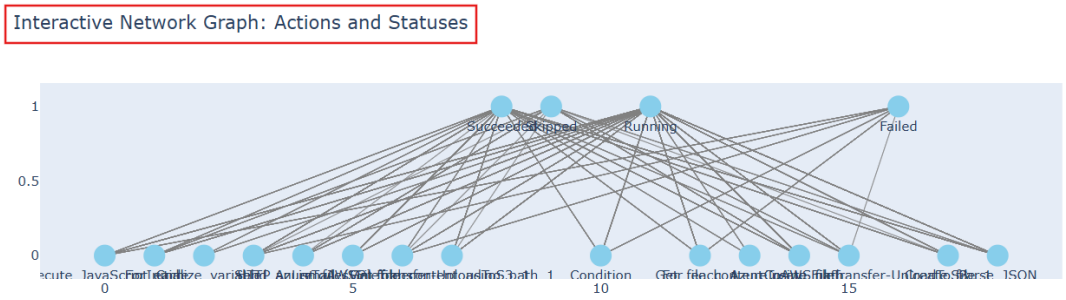


```python
import plotly.graph_objects as go

# Prepare data for Sankey diagram
sankey_data = data.groupby(['resource_actionName_s', 'status_s']).size().reset_index(name='count')

# Create mappings for source and target nodes
all_nodes = list(set(sankey_data['resource_actionName_s']).union(set(sankey_data['status_s'])))
node_map = {node: i for i, node in enumerate(all_nodes)}  # Map node names to indices

# Map source and target columns to their indices
sankey_data['source'] = sankey_data['resource_actionName_s'].map(node_map)
sankey_data['target'] = sankey_data['status_s'].map(node_map)

# Build the Sankey diagram
fig = go.Figure(go.Sankey(
    node=dict(
        pad=15,
        thickness=20,
        line=dict(color="black", width=0.5),
        label=all_nodes  # Use all unique nodes as labels
    ),
    link=dict(
        source=sankey_data['source'],  # Map sources
        target=sankey_data['target'],  # Map targets
        value=sankey_data['count']    # Use count as the weight
    )
))

# Set title and display the figure
fig.update_layout(title_text="Sankey Diagram: Actions to Statuses", font_size=10)
fig.show()
```

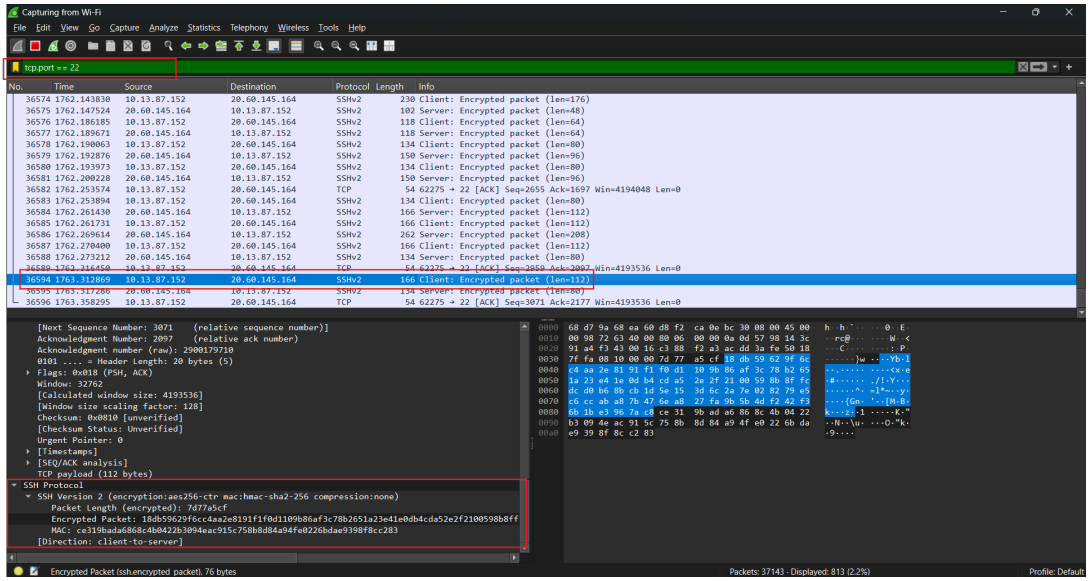
Sankey Diagram: Actions to Statuses



```
import plotly.graph_objects as go

# Step 1: Prepare Nodes and Edges
unique_nodes = list(set(data['resource_actionName_s']).union(set(data['status_s'])))
node_indices = {node: i for i, node in enumerate(unique_nodes)}

edge_x = []
edge_y = []
for action, status in edges:
    edge_x.append(node_indices[action])
    edge_y.append(node_indices[status])

# Step 2: Create Network Graph
fig = go.Figure()

# Add edges
for action, status in edges:
    fig.add_trace(go.Scatter(
        x=[node_indices[action], node_indices[status]],
        y=[0, 1],
        mode='lines',
        line=dict(width=0.5, color='gray'),
        hoverinfo='none'
```



- Step 14: Install Wireshark, after installing, start the WireShark application and start capturing the packet and simultaneously start file transmission from WinSCP to hosted Azure to an azure hosted SFTP and by using tcp.port==22 filter in Wireshark to capture the SFTP's traffic only. This step verify that the packet during the transmission is encrypted.

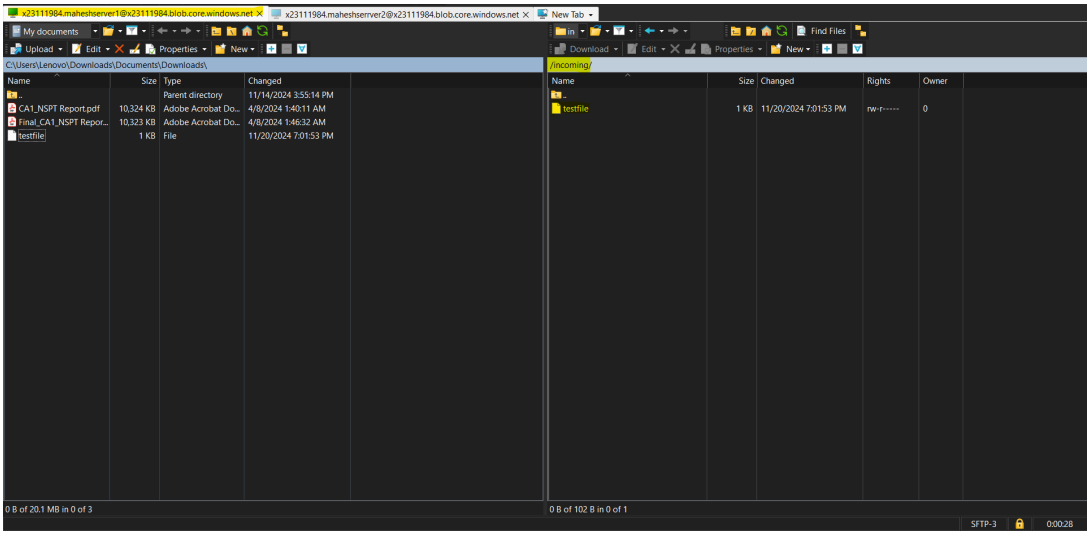


4 Senario Based Results

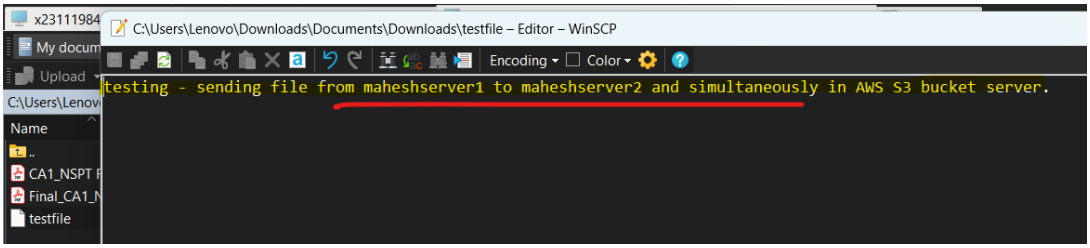
4.1 Senario I

–Within Azure: From the SFTP server to designated storage folders.

RunId - 0858468849944935512376174581 4CU47
File Name - testfile drops in maheshserver1 (Home Directory x23111984container/incoming)



–Logic App successfully triggered and after processing the file content is encoded during transit. Initially, the file Content of test file shown below:

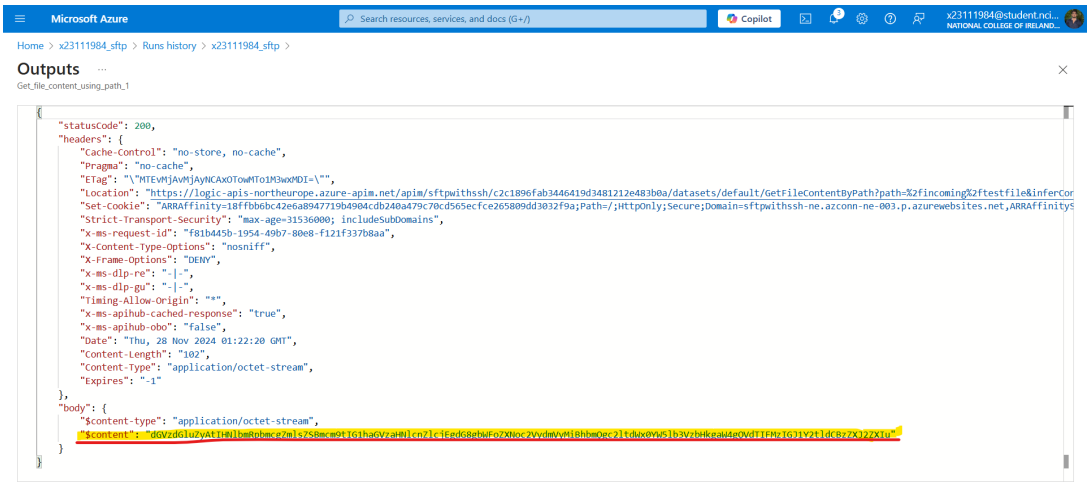


Encoded Content:

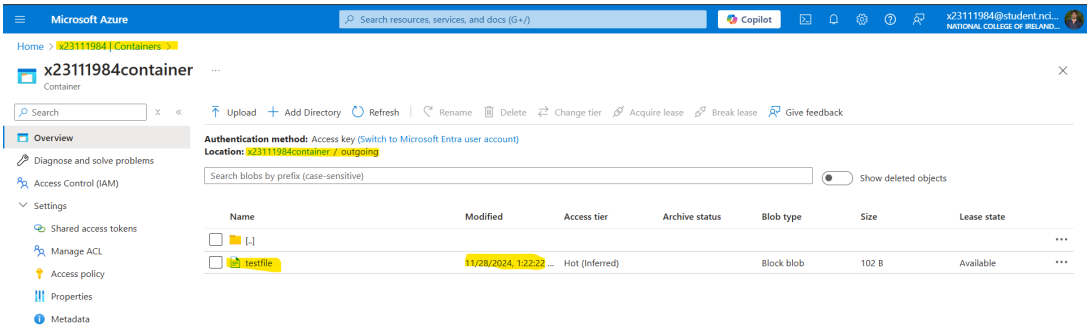
”$content”: ”dGVzdGluZyAtIHNlbmRpbmcgZmlsZSBmcm9tIG1haGVzaHNlcnZlcjEgdG8gbWFoZXNoc2VydmVyMiBhbmQgc2ltdWx0YW5lb3VzbHkgaW4gQVdTIFMzIGJ1Y2tldCBzZXJ2ZXIu”

Decode Content:

Testing - Sending file from maheshserver1 to maheshserver2 and simultaneously in AWS S3 bucket server.



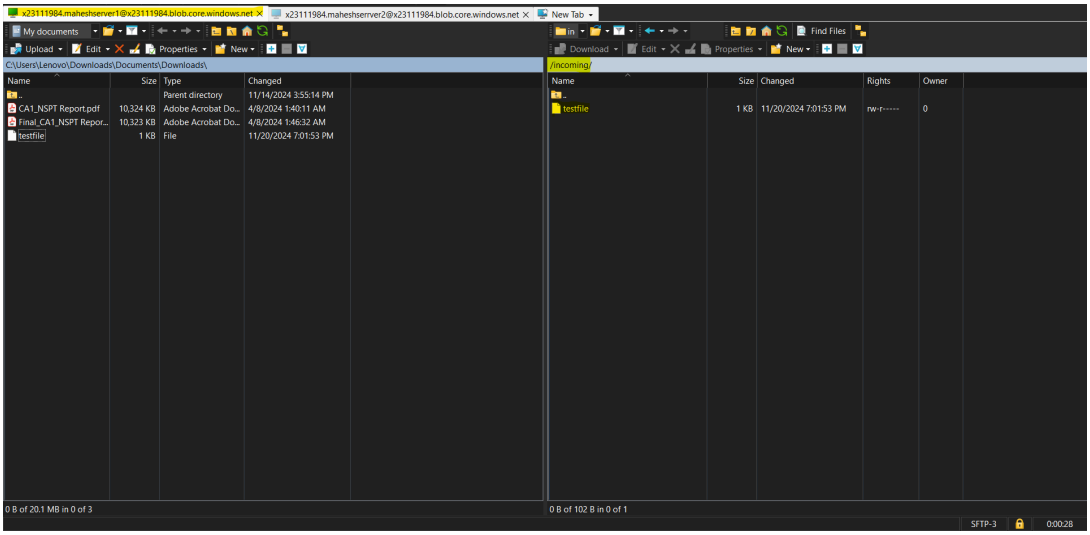
–After processing, the file is in the destination (Outgoing) folder.



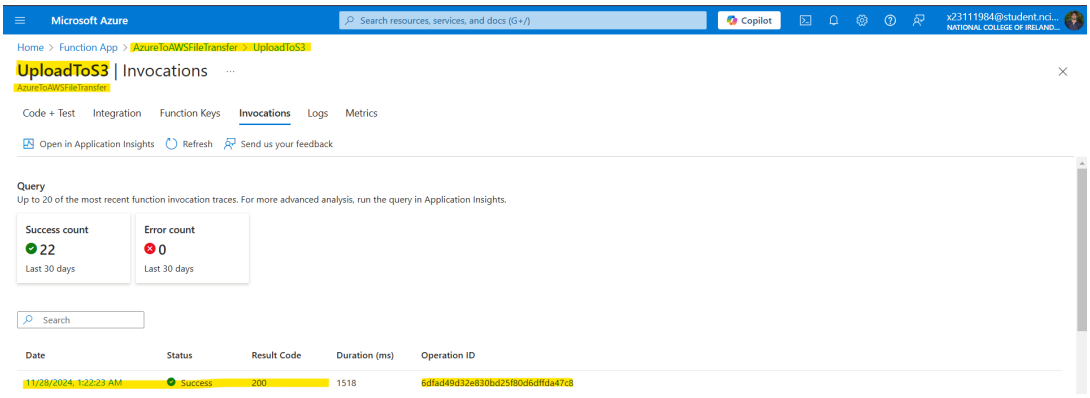
4.2 Senario II

Across Clouds: From Azure to AWS S3 using Azure Functions and the AWS S3 client.

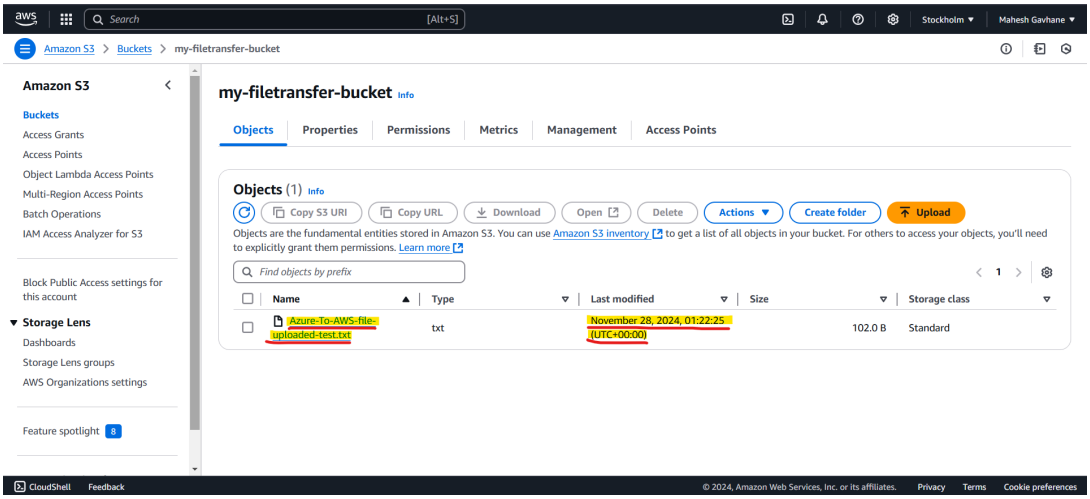
–Same test file has been dropped in an incoming folder.

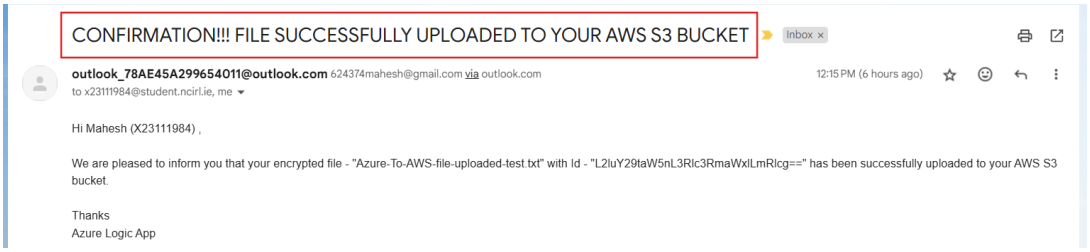


–**Azure Logic App, Action - Azure Function App is triggered and invoked function – UploadToS3 as shown below.**



–After successfully processing the logic app, testfile uploaded to AWS S3 Bucket and Logic App sent confirmation email as shown below.





–If Logic App terminates, send alert mail with subject line “Alert File Not Found”.

After processing the file successfully and capturing the log, generate the graphs for Visualization using PROM Tool and Jupyter Notebook as mentioned in Step 3.