README for Flow Log Parsing Program

# Project Overview

This program is designed to parse flow log data and apply tags based on a lookup table provided in a CSV file. Each row in the flow log file is mapped to a tag, determined by the combination of destination port and protocol. The program generates two primary outputs:  
1. Tag Counts: The number of occurrences of each tag.  
2. Port/Protocol Combination Counts: The number of occurrences of each destination port and protocol combination.

# File Structure

• Flow\_Log\_parser.py: The main program that parses the flow logs and generates the output.  
• protocol-numbers-1.csv: Contains mappings of protocol indexes to their respective protocol names.  
• lookup\_table.csv: Contains mappings of destination ports, protocols, and corresponding tags.  
• flow\_logs.txt: The input flow logs file to be parsed.  
• output.txt: The output file containing tag counts and port/protocol combination counts.

# Requirements

• Input files are plain text (ASCII) files.   
• The program supports AWS VPC Flow Logs version 2 only.

- Python 3.6 or higher

- Input files:

- `flow\_logs.txt`: Contains the flow log data (up to 10 MB)

- `lookup\_table.csv`: Contains the tag mappings (up to 10000 mappings)

- `protocol-numbers-1.csv`: Contains protocol number to name mappings

# Installation

No additional libraries are required beyond the Python standard library. Simply clone or download the repository to your local machine.

# Assumptions

Ensure all input files (`flow\_logs.txt`, `lookup\_table.csv`, `protocol-numbers-1.csv`) are in the same directory as the script.

• The input files must be in the correct format:  
 - Flow log file: The expected format follows AWS VPC Flow Logs version 2.  
 - Lookup table: Must contain three columns (dstport, protocol, tag).  
 - Protocol index file: Should have two columns (index, protocol), where index can either be a single number or a range (e.g., 100-200).  
• Case-Insensitive Matching: Protocol names and tags are matched without considering case sensitivity.  
• The program only processes logs with valid protocol and dstport fields.

# How to Run

1. Ensure that Python 3.x is installed on your machine.  
2. Place all necessary files (Flow\_Log\_parser.py, protocol-numbers-1.csv, lookup\_table.csv, flow\_logs.txt) in the same directory.  
3. Run the program from the command line:  
 python3 Flow\_Log\_parser.py  
4. The program will process the flow log file and generate the output in output.txt.

# Output Files

• output.txt: The program generates the output in this file with two sections:

1. Tag Counts: Shows the count of each tag, with 'Untagged' at the end

2. Port/Protocol Combination Counts: Shows the count of each unique port and protocol combination

# Code Explanation

1. FlowLogProcessorRun Class: Handles the core logic of parsing flow logs and mapping tags.  
 - create\_protocol\_index\_map(): Reads the protocol index file and creates a dictionary mapping indexes to protocol names. It also handles ranges of indexes.  
 - create\_port\_protocol\_tag\_map(): Reads the lookup table and creates a mapping between (port, protocol) pairs and tags.  
 - get\_counts\_from\_flow\_logs(): Parses the flow logs, counts occurrences of each tag, and counts each port/protocol combination.  
 - write\_counters\_to\_file(): Outputs the tag counts and port/protocol combination counts to the output file.  
 - process(): Orchestrates the flow by calling the above methods to complete the parsing and output generation.

If a log entry doesn't match any tag in the lookup table, it's counted as 'Untagged'.

2. Main Function: The script is executed from the command line. It initializes the FlowLogProcessorRun class with the necessary file paths and calls the process() function to perform the task.

# Implementation Details

- The script uses Python's built-in libraries (csv, collections, logging) to avoid dependencies on external packages.

- It implements a class-based structure for better organization and potential future extensions.

- The parsing is done in a single pass through the flow log file for efficiency.

- Dictionaries and defaultdict are used for fast lookups and counting.

# Testing

1. Basic Test: Verified with small sample files to ensure correct tag mappings and port/protocol counts.  
2. Edge Cases:  
 - Tested with multiple tags mapped to the same port and protocol combination.  
 - Tested with large flow log files (up to 10 MB) and a lookup table with 10,000 mappings.  
 - Tested with mixed-case protocol names to ensure case insensitivity.

# Conclusion

This program is efficient, modular, and handles the task of parsing flow logs and mapping them to tags based on a lookup table. The process is scalable for files up to 10 MB and supports thousands of tag mappings. The case-insensitive matching ensures that it works robustly with real-world data.