**Title:** Telegraf Plugin for Cloud Service Billing Monitoring

**Objective**: Develop a Telegraf plugin to monitor cloud service billing metrics, tracking costs and usage for various cloud services. This plugin will help organizations manage cloud expenses and optimize resource usage.

**Project Description**:

**Introduction:** Create a new Telegraf plugin to monitor cloud service billing metrics. This project includes gathering billing data, integrating with existing systems, and visualizing the data through dashboards.

**Outcome**: Deliver a Telegraf plugin that monitors cloud service billing metrics, providing insights to help manage cloud expenses and optimize resource usage.

Telegraf Plugins:

Project Overview:

In this project you will develop Telegraf plugins. Telegraf is a open source monitoring tool that can collect data from various systems. Your Telegraf plugin will be developed based in accordance to the development guidelines set by Telegraf. Potentially your plugin can also be donated to open source. We have prioritized list of systems for which you can develop the plugin. Coding will be in Python.

Introduction to Telegraf:

Telegraf is an open-source agent written in Go for collecting, processing, aggregating, and writing metrics. It is highly extensible with plugins for various inputs, outputs, processors, and aggregators.

Setting Up the Development Environment:

*Prerequisites*

* **Python**: Ensure you have Python installed (preferably version 3.7 or higher).
* **Telegraf**: Install Telegraf on your system.

*Installing Python*

Download and install Python from the official [Python website](https://www.python.org/). Make sure to add Python to your PATH.

*Installing Telegraf*

Telegraf can be installed using various methods. Here are instructions for common operating systems:

•  **Windows**: Download the latest Telegraf .exe file from the [Telegraf Releases](https://github.com/influxdata/telegraf/releases) page and install it.

•  **MacOS**: brew install telegraf

•  **Linux**:

sudo apt-get update

sudo apt-get install telegraf

*Setting Up a Virtual Environment*

It’s a good practice to use a virtual environment for your Python projects.

python -m venv telegraf-plugin-env

source telegraf-plugin-env/bin/activate  # On Windows: telegraf-plugin-env\Scripts\activate

*Installing Required Libraries*

Install necessary libraries using pip.

pip install requests psutil

Developing a Telegraf Plugin in Python

Telegraf plugins follow specific guidelines for development. Here's a basic structure for a Telegraf plugin:

1. **Input Plugins**: Collect data from various sources.
2. **Output Plugins**: Send data to different destinations.

Sample Plugin Code

Let's create a sample input plugin that collects system memory usage.

import psutil

class MemoryUsagePlugin:

    def \_\_init\_\_(self):

        self.plugin\_name = "memory\_usage"

    def gather(self):

        memory\_info = psutil.virtual\_memory()

        return {

            "total": memory\_info.total,

            "available": memory\_info.available,

            "percent": memory\_info.percent,

            "used": memory\_info.used,

            "free": memory\_info.free

        }

    def to\_telegraf\_format(self, data):

        formatted\_data = []

        for key, value in data.items():

            formatted\_data.append(f"{self.plugin\_name},{key}={value}")

        return "\n".join(formatted\_data)

if \_\_name\_\_ == "\_\_main\_\_":

    plugin = MemoryUsagePlugin()

    data = plugin.gather()

    print(plugin.to\_telegraf\_format(data))

Testing and Running the Plugin

**Run the Plugin**:

python memory\_usage.py

**Integration with Telegraf**:

Configure Telegraf to use your plugin. Add the following to your telegraf.conf

[[inputs.exec]]

  commands = ["python /path/to/memory\_usage.py"]

  data\_format = "influx"

Developing a Telegraf plugin in Python involves setting up your development environment, writing the plugin code, and configuring Telegraf to use your plugin. The provided sample code and guidelines should help you get started with creating your own plugins.