# People & Vehicle Counter Dashboard Plan using Grafana

This document outlines the strategy for creating a dynamic, beautiful, and interactive Grafana dashboard to visualize data from a people and car tracking system.

## 1. The Core Concept: How It Works

Grafana itself doesn't store data. It's a visualization layer. It needs to connect to a **data source**. Since you have log files, the best and most modern approach is to use the**Grafana Loki** stack.
Here's the data flow:

1. **Your Application:** Generates a log file with every person or car entry.
2. **Promtail (The Agent):** A small program that runs on your server, watches your log file, and sends new entries to Loki.
3. **Loki (The Database):** A database specifically designed for efficiently storing and querying logs.
4. **Grafana (The Dashboard):** Connects to Loki as a data source, runs queries, and displays the results in beautiful panels.

## 2. Step 1: Standardize Your Log Format

For Grafana to effectively parse your data, your logs need a consistent structure. A key-value pair format or JSON is ideal. This makes it easy to query for "type=person" or "type=car".
**Recommendation:** Use a JSON format for each log line.
**Example events.log file:**

```
{"ts": "2025-10-01T22:10:01Z", "event": "entry", "type": "person"}
{"ts": "2025-10-01T22:10:04Z", "event": "entry", "type": "car"}
{"ts": "2025-10-01T22:10:05Z", "event": "entry", "type": "person"}
{"ts": "2025-10-01T22:10:15Z", "event": "entry", "type": "person"}
{"ts": "2025-10-01T22:11:20Z", "event": "entry", "type": "car"}
```

- **ts:** The timestamp (ISO 8601 format is standard).
- **event:** The action that occurred.
- **type:** The object that was detected.

## 3. Step 2: Set Up the Grafana Loki Stack

You will need to install Grafana, Loki, and Promtail. The easiest way to do this is using Docker or Docker Compose. You can find many pre-built configurations for this online.
Key Configuration:
You'll need to configure Promtail to find and read your log file.

**Example promtail-config.yml:**

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /tmp/positions.yaml

clients:
  - url: http://loki:3100/loki/api/v1/push

scrape_configs:
- job_name: people-counter
  static_configs:
  - targets:
      - localhost
    labels:
      job: people_counter
      __path__: /path/to/your/events.log # <-- IMPORTANT: Point this to your log file
  pipeline_stages:
  - json:
      expressions:
        type: type
  - labels:
      type:
```

- This configuration tells Promtail where your log file is (__path__).
- The pipeline_stages part tells it to parse the JSON and extract the type field as a queryable label.

## 4. Step 3: Design Your Grafana Dashboard

Once Grafana is running and connected to Loki as a data source, you can start building! Here is a proposed layout for a beautiful and functional dashboard:

### Dashboard Title: People & Vehicle Entry Dashboard

#### Row 1: Key Performance Indicators (KPIs)

*Use the "Stat" panel for each of these.*

| **Panel 1: People Today** | **Panel 2: Cars Today** | **Panel 3: Busiest Hour Today** |
|---|---|---|
| A large, single number showing the total count of people detected in the last 24 hours. | A large, single number showing the total count of cars detected in the last 24 hours. | Shows the hour with the most traffic (people + cars) today. Can be configured to show the |

| | | |
|---|---|---|
| | | total count during that hour. |
| **Query:** sum(count_over_time({job="people_counter", type="person"}[24h])) | **Query:** sum(count_over_time({job="people_counter", type="car"}[24h])) | **Query:** More advanced, uses topk(1, sum by (hour) (count_over_time({job="people_counter"}[24h]))) |

**Row 2: Trends Over Time**

*The time range for these panels will be controlled by the dashboard's main time picker.*

| **Panel 4: Entry Traffic Over Time (Time Series Graph)** | **Panel 5: Entries by Type (Bar Chart)** |
|---|---|
| A beautiful line or bar graph showing the rate of entries. You can stack people and cars to see the total flow. | A simple bar chart showing the total count of "people" vs. "car" for the selected time range. Excellent for quick comparisons. |
| **Query A (People):** sum by (type) (rate({job="people_counter", type="person"}[$__interval])) | **Query:** sum by (type) (count_over_time({job="people_counter"}[$__range])) |
| **Query B (Cars):** sum by (type) (rate({job="people_counter", type="car"}[$__interval])) | |

**Row 3: The Raw Data**

*This directly fulfills your request to see the log of each entry.*

| **Panel 6: Live Event Log (Logs Panel)** |
|---|
| A real-time, scrolling view of the actual log entries as they come in. You can filter them directly in the panel. This is perfect for debugging or detailed inspection. |
| **Query:** {job="people_counter"} |

## 5. Making It Interactive

To allow the end-user to filter, you can add **Dashboard Variables**.

- **Create a variable named Type:**
  - **Type:** Custom
  - **Values:** person, car
  - This will create a dropdown menu at the top of your dashboard.
- **Update Your Queries:**
  - You can then update your queries to use this variable. For example, the "People Today" query could be changed to a generic "Selected Type Today" query: sum(count_over_time({job="people_counter", type="$Type"}[24h]))
  - When the user selects "car" from the dropdown, the panel will automatically update to show the car count.

By following this plan, you will create a professional, beautiful, and highly functional dashboard that allows users to explore the people and car tracking data at any interval they choose.