Lab Practicals Guide
IoT Massive Data Visualisation with Grafana

Fernando Illescas Martínez, Sergio Pastor Martínez,
Francisco José Alvarado Alcón, Francisco Jesús Sánchez Rubio
and José Francisco Beltrán Sánchez

November 8, 2023

1 Introduction

This hands-on lab guide aims to guide students through the process of visualising Internet of Things (IoT) data using Grafana and an InfluxDB database. Before that, we will give a brief introduction to these tools.

Grafana

Grafana is a versatile and user-friendly data visualization platform, often considered the artist's canvas for data. It serves as a crucial tool in the realm of data analytics, allowing you to create interactive and highly customizable dashboards. These dashboards offer a visually engaging way to display data collected from various sources, including databases, monitoring systems, and IoT devices. With Grafana, you can effortlessly transform raw data into actionable insights, making it an indispensable tool for professionals seeking to harness the full potential of their data.

InfluxDB

InfluxDB is a robust and scalable open-source time-series database, purpose-built for handling vast volumes of time-stamped data points. This database excels at efficiently storing, retrieving, and managing time-series data, which is especially valuable for applications generating data over time, such as IoT sensor readings, performance monitoring, and analytics. InfluxDB's simplicity, speed, and ability to handle large datasets make it a preferred choice for those in need of a reliable solution for storing and querying time-series data.

2 Materials required

To ensure the follow-up of the laboratory practice, it is necessary to have the following processes correctly installed and configured on the computer:

- A computer with Telegraf and InfluxDB processes accessible and working properly.
- Sample data in InfluxDB to work with.

3 Objectives of the Practice

The purpose of this practicum is to train students to use the Grafana platform to visualise and analyse data generated by Internet of Things (IoT) devices. As the IoT continues to grow and become embedded in a variety of applications and sectors, the need to understand and extract meaningful information from IoT data becomes crucial. Grafana is a widely used tool to achieve this goal, enabling real-time data visualisation, custom dashboard creation and efficient IoT data analysis. At the end of this course, students will be prepared to use Grafana effectively in the context of IoT projects and applications.

3.1 Challenges we will learn to solve in practice

1. Real-Time Data Visualisation: Grafana allows users to create interactive dashboards that display IoT data in real-time. This is essential for monitoring and making data-driven decisions in real-time applications, such as monitoring temperature, humidity or pressure sensors in a smart building.

2. Connecting to Data Sources: IoT data is usually stored in databases, such as InfluxDB, which is a widely used time series database. Grafana connects to these data sources to retrieve and display information. Students will learn how to configure these connections.

3. Creating Custom Dashboards: Grafana allows users to create custom dashboards that include graphs, charts and other visual elements. Students will learn how to design dashboards that fit the specific needs of their IoT application.

4. Queries and Data Filtering: Grafana includes a query language that allows students to perform queries on the InfluxDB database. This is essential for extracting relevant data from large information sets.

5. Alerts and Notifications: In an IoT environment, it is critical to be aware of critical changes in data. Students will learn how to set up alerts and notifications in Grafana to be informed in real time when values exceed certain thresholds.

6. Performance Optimisation: In IoT projects, where large amounts of data are generated, performance optimisation is essential. Students will learn how to optimise the speed of loading and visualising data in Grafana.

4 Development of the practice

4.1 Installation and configuration

- First, Grafana must be installed in each computer. For doing so, students have to follow the instructions in the following link: `https://grafana.com/docs/grafana/latest/setup-grafana/installation/debian/`

- Next, continuing the work from the previous assignment, students will connect their Chirpstack server to their InfluxDB instance through the Telegraf plugin. Instructions for this process are provided at the first annex of this document.

- Students should then sign in to Grafana for the first time.
    1. Open your web browser and go to http://localhost:3000/.
    2. On the sign-in page, enter admin for both the username and password.
- Students should access the database and check that data is indeed loaded. To do this, you can run a sample query on the InfluxDB database. Then they can add the database as data source for Grafana. The details for this step are on the annex II.

  Before you begin:
    1. Ensure that you have the proper permissions.
    2. Understand the query language of the target data source.
    3. Ensure that data source for which you are writing a query has been added.

4.2 Creating Dashboards, Panels and Graphics in Grafana

- Creating a Basic Grafana Panel. To do this, create your first dashboard and add data to visualise it.
  Please note that the source of the data is InfluxDB.
- Creating a Custom Dashboard. You should create a custom dashboard that includes multiple panels with different types of visualizations.
  Optional: You can customize the appearance and layout of their dashboard.

4.3 Querying and Filtering Data

Once we know how to create a dashboard and how to visualise the data, we need to learn how to filter the data so that the dashboards show what we want.

To do this, **let's assume that we are working with a Colombian client**. The client tells us that he wants to display the historical temperature and humidity from sensors 1 and 2 on the dashboard. He is only interested in displaying the last 24 hours and, in addition, he tells us that the data must be displayed in Colombian time.

- For the above case, modify the graphana queries and edit the dashboard configuration (or create a new one) in which everything is displayed according to the client's indications.

4.4 Configuring Alerts and Notifications

Alerting is a critical function when using Grafana for monitoring IoT systems. In the complex and dynamic world of IoT, real-time insights and timely responses are paramount. Grafana's alerting capabilities allow organizations to set up proactive notifications and triggers based on specific conditions or thresholds, ensuring that anomalies, failures, or performance issues are addressed promptly. This proactive approach not only minimizes downtime but also helps maintain the integrity of IoT networks, preventing potential data loss, system failures, or other disruptions. By leveraging Grafana's alerting features, IoT operators can maintain a high level of control and responsiveness in their systems, thereby enhancing the overall reliability and performance of their IoT deployments.

4.4.1 Alert rules

In order to set an alert using Grafana, we need to create a rule that defines when we want to be contacted. This can be done in the menu *Alerting > Alert routes*. Right now, there should be no alerts. We can now create the first clicking the 'New alert rule' button. The alert name should be short but descriptive, call this alert 'High PM10'. Then, we need to stablish the origin of the data and the time range for the query. Remember that all our data is being stored in InfluxDB, so Grafana is reading directly from the database, not from the sensor. Do not change yet the time interval of this query, as it indicates which values from the database are being read each time.

The established query returns an array with all the values inside the default time interval. We want to create an alert that warns us immediately when the PM-10 particles surpass 80 µg/m^3, which means a poor air quality that requires human intervention (opening the window). We need to select an expression that let us read the most recent value stored. Therefore, we need to reduce the array to a single numerical value: select the reduce operation last. This means that the value obtained will be the latest from the read entries. Think about the relationship between the time range of the query and the sampling frequency: what happens if the time range is shorter than the time between samples? What happens if the time range doubles or triples the time between samples? And if a sample is missing? Understanding these questions is key to selecting an appropriate time range.

After the resolving the questions, select an appropriate time range and complete the expression. 'B' returns the last value read from InfluxDB by query 'A', and 'C' is a Boolean value: whether the value obtained from the reduce expression in outside the threshold. By default, 'C' should be selected as the alert condition, do not modify the alert condition. In the Alert evaluation behaviour, add the alert to a new folder and to a new group called 'Last Value'. The folders help organize alerts and they are used for alert grouping, this would be explained later in the lesson. The Evaluation groups are also used for organizing alerts, but they have one additional function: they determine how often are rules evaluated. Every time an alert is evaluated, the query is read and tested according to the alert definition, potentially causing its state to change. Set this group evaluation time as one minute. Do not modify the Pending period.

In section 4, add a summary and description with the most important information and save it. From the alert rules menu, you should be able to see the rule and its state, whether it is Normal if everything is working properly, or Pending or Firing if the conditions are being meet. Additionally, you can see the health of the status, which can be Ok, Error, and NoData. The possible states for an alert are as follows:

- Normal. The state of an alert that is neither firing nor pending, everything is working correctly.
- Pending. The state of an alert that has been active for less than the configured threshold duration[1].
- Alerting. The state of an alert that has been active for longer than the configured threshold duration[1].

[1]Alerts must be in breach of the condition during a threshold duration called Pending period until the alert rule fires. Alerts will transition first to Pending and then Firing; thus, it will take at least two evaluation cycles before an alert is fired. Therefore, even if the pending period is lesser than two times the evaluation period, the alert would not be fired until twice the evaluation period.

- NoData. No data has been received for the configured time window.
- Error. The error that occurred when attempting to evaluate an alerting rule.

Right now, we can see the alert and its state in the Grafana page. However, we want to be notified when an alert is triggered. Access *Alerting > Contacpoints*, and define an email address as a contact point. Finally, the Notification policies determine which alerts are sent by which contact point according to labels and they also configure other settings like grouping notifications. For the time being, the default policy will be enough. We have set our first alert: when PM-10 particles surpass 80 µg/m^3 during a certain time, the alert state will change to Firing, and we will be notified. Manually trigger the alert (breathing over the device) and check if the state changes to Pending.

Let's modify the previous alert. When the conditions are met, the state changes from Normal to Pending and, if the alert has been active for longer than the configured threshold duration, it changes to Firing, therefore sending the notification. The duration of this threshold is configured in the Alert evaluation behaviour section. How much time will it take before the alert is fired? Reduce it to two minutes and save the changes.

Following the same steps, add a new rule in the same group that fires an alert if the last humidity sample is under 60% for two minutes[2].

In practical scenarios, spurious data points occur frequently due to various factors. This is why the most recent data point isn't considered a reliable measurement; it updates rapidly but lacks reliability. A more suitable alternative is to calculate the average over a specific time period. By following the same procedure, introduce a new rule within the 'Average value' group evaluated every two minutes, triggering an alert when the average temperature taken over the last 10 minutes remains above 27 °C for five minutes. This setup ensures that a notification will be generated when the average value unexpectedly rises, allowing for a more robust monitoring approach.

4.4.2 Notification policies

There are now three different alerts configured in our Grafana. However, it is expected that different alerts will have different severity, and therefore it will vary how we want to be notified. We can control that by adding labels to the alerts, and the adding Notification policies. First, edit the second alert, the one with 'below' as its condition. In section 5, add the label 'severity = low'. Edit the other two alerts and mark their severity as 'major'. Define a second contact point with the first email address plus a different one. Now, go to the Notification policies menu. We need to add two nested policies under the default policy. One for major severity alerts, that will be sent to both emails, and one for low severity alerts, that will be sent only to the first address. Additionally, modify the default policy so that if no severity is labelled, both emails should be notified. Consider the option 'Continue matching subsequent sibling nodes'. Is it required in this particular case? Manually fire the alert by holding it in your hands for a few minutes, and check if a notification has been sent to both email addresses.

Grouping alerts in Grafana is a valuable feature for better organization and management of alerting rules. When notifications are grouped by folder and name in Grafana (the default behaviour), it means that alerts with the same name within the same folder

[2]Remember that, since we are working with the same Evaluation group 'Last Value', the Evaluation interval still is 1 minute.

are treated as a single group, and therefore notified together. This default behaviour offers several benefits. **Note:** we are still working on *Alerting > Notification policies*.

- Clarity and Organization: it enhances the clarity of your alert setup by grouping similar alerts together based on their names and the folders they are placed in. This makes it easier to understand and manage alert rules, particularly in environments with a large number of alerts.
- Reduced Alert Noise: grouping alerts with the same name and within the same folder can help reduce alert noise. It ensures that you won't receive multiple notifications for the same underlying issue, as all instances of that alert are consolidated into a single group.
- Concise Dashboard View: dashboards become more concise and less cluttered when alerts are grouped in this way. Users can quickly assess the status of their systems without being overwhelmed by redundant alerts.
- Easier Troubleshooting: when investigating issues or incidents, grouped alerts simplify the troubleshooting process. You can focus on the specific issue, identify patterns, and take appropriate actions more efficiently.

Overall, grouping alerts in Grafana contributes to a cleaner, more organized, and more manageable alerting system, ultimately improving the effectiveness and user experience of alerting and monitoring within the platform. However, by default Grafana waits 30 seconds when an alert is triggered, expecting to receive alerts from the same group. To reduce the reaction time when an alert occurs, modify the Timing options of the major severity alerts by modifying its notification policy. By default, general timings are inherited from the parent (in this case, the default policy), so in order to modify them, you need to *Override general timings*. Reduce the three times to a fifth of their original value.

4.4.3 Silences

Grafana silences are useful in alerting because they allow users to temporarily suppress or mute alerts during planned maintenance, known issues, or other expected events. This prevents unnecessary alert notifications and alert fatigue while still keeping the alerting system active for critical, unexpected incidents. Silences help maintain a balance between timely incident response and reducing the noise associated with ongoing, non-emergency situations, improving the overall efficiency of alerting systems in monitoring and managing systems and services.

Before continuing, make sure that the state of the 'High PM10' alert is Normal. Then, from the menu *Alerting > Silences* , add a silence that starts immediately and lasts for 20 minutes for the 'High PM10'. You can use the label *alertname*, which refers to the name of the alert. Once created, a silence should appear with points to 1 alert. Verify that the alert firing does not get notified by email, but still appears as Firing in the Alert rules menu.

4.4.4 Alerts with multiple expressions

Grafana alerts allow for multiple expressions, queries, and math functions, which is incredibly useful for advanced and flexible alerting scenarios in monitoring systems. Multiple expressions and queries allow to define complex conditions for triggering alerts. You can combine various metrics and data sources to create sophisticated logic tailored to your specific monitoring needs. This is essential when you need to consider multiple factors or conditions before generating an alert. By using math functions in your alert expressions, you gain fine-grained control over how data is processed and analysed for alerting. Additionally, when alerts are based on multiple expressions and queries, you can optimize resource usage. For instance, you can avoid firing alerts for isolated spikes or minor fluctuations by setting conditions that require sustained or significant changes in the data before an alert is triggered. Finally, math functions can be used to dynamically calculate threshold values, allowing your alerts to adapt to changing conditions. For instance, you can set thresholds based on percentages, moving averages, or other dynamic criteria. In summary, Grafana alerts with multiple expressions, queries, and math functions empower users to create advanced, adaptive, and intelligent alerting rules. They are particularly useful in complex monitoring environments where standard alerts might not provide the necessary insights or flexibility to effectively manage and maintain systems and services. Create a new alert that takes the highest and lowest temperature from the last 10 minutes, calculates the difference between them, and send a major alert if the difference is at least 5 °C. Set the pending period as 1 minute.

Exercise. The following exercise will test all the knowledge acquired in this section. There are two users to be notified: user 1 should be notified of airborne particulate alerts, and user 2 should be notified of temperature and humidity alerts. Additionally, both users must be notified of any severe alerts. This are the alerts that the client has requested. Unless otherwise specified, all alerts have low priority.

- PM-10 particles surpass 80 µg/m^3 or PM-2.5 particles surpass 50 µg/m^3. If both of them happens simultaneously, it's considered a severe alert, otherwise it's low priority
- The last sample of humidity exceeds 60%
- The temperature is above 27 °C or below 15 °C

By default, each alert should be tested during two minutes before firing. However, if temperature stays below the threshold during at least ten minutes, it becomes a severe problem. Remember to properly configure the grouping alerts feature to make sure that similar alerts are not sent simultaneously. **Hint:** you may want to define a new 'class' label as to separate alerts in 'particles', 'humidity' and 'temperature', and modify the default policy so that alerts are grouped by class.

5 Annex I: Connecting Chirpstack to Telegraf

In order to connect the data received in Chirpstack to our database on InfluxDB, we will use the Telegraf plugin seen in the previous lesson. Since Chirpstack by default has an output MQTT integration, we can set up a MQTT Consumer input module on Telegraf to catch the incoming data and direct it to InfluxDB as output. For doing this we will need to change the configuration file on the Telegraf tab on the InfluxDB web interface. An example configuration file template is available on the GitHub repository.

Once the configuration is changed, reload the Telegraf service on your machine to apply the new configuration. You will see that your setup is working with the appearance of a *mqtt_consumer* field with your machine IP as host on the query builder.

6 Annex II: Creating a new data source

When you connect to Grafana and attempt to create a new panel, you will automatically be prompted with an option to add a data source. Data sources can also be made by going to the lift sidebar and clicking on the *Connections* option, which will show a **Data sources** tab.

We will create an InfluxDB data source and set the query language to **Flux**, which provides the necessary fields to register our database.

The database URL will be *localhost*, as Grafana and InfluxDB are installed on the same machine. Each student will have to add their own authentication details on the relevant fields: user and password, organization, bucket and API access token (create one on the InfluxDB web interface if you have not made one).

Clicking on the **Save and test** button will confirm whether the data source can be successfully accessed or not.

7 Additional Resources

- GitHub repository: `https://github.com/QartiaCube/BIP_course/tree/main/IOT_practice_4`
- Grafana Documentation: `https://grafana.com/docs/`
- InfluxDB Documentation: `https://docs.influxdata.com/influxdb/`