# FixedIT Data Agent quickstart guide

This file contains step-by-step instructions on how to start using the FixedIT Data Agent to monitor your Axis devices.

This guide will focus primarily on using the FixedIT Data Agent as a data collection agent for the Axis devices and to push system metrics to an InfluxDB/Grafana dashboard. The application is much more powerful than that and can easily be extended with your own custom application logic or even used as a generic platform to build your own edge applications for the Axis devices.

The FixedIT Data Agent is an advanced wrapper around the Telegraf software, which is a platform that allows the creation of custom processes using input plugins, output plugins and different kinds of processors, filters and aggregators. By default, the FixedIT Data Agent comes preloaded with a configuration for advanced monitoring of the Axis devices, but this configuration can be extended or replaced with your own custom configuration. For the advanced users, this means that you can run essentially any Telegraf workload directly on the Axis devices.

For a detailed manual on everything that is possible with the application, see the `FIXEDIT_DATA_AGENT_CONFIG_SPEC.pdf` file.

**Table of contents**

## Installing the application

The FixedIT Data Agent application is built for both `armv7hf` and `aarch64` architectures. The right version should be installed depending on each device's architecture. Some newer versions of the device software will display the device's architecture in the application upload pop-up window, as seen below:

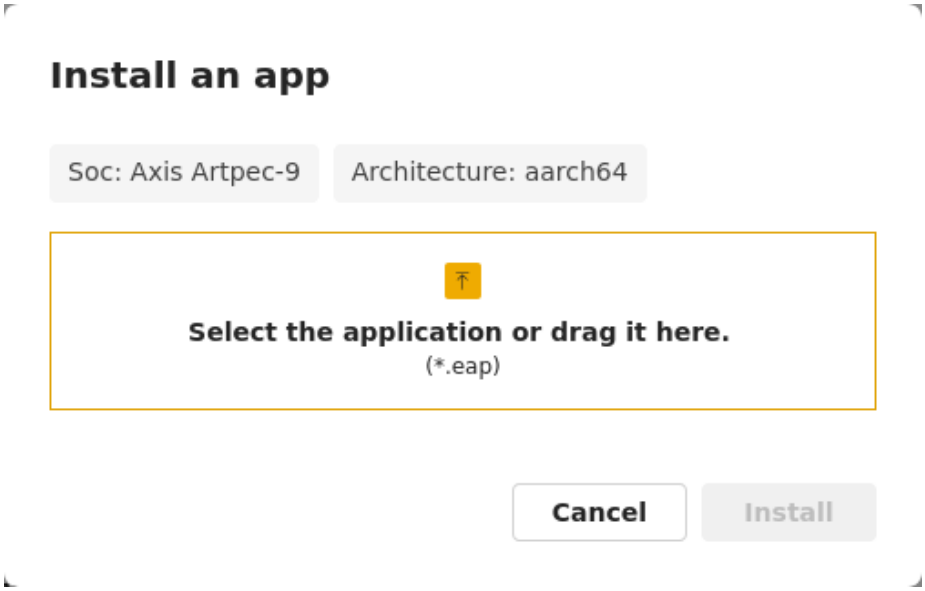



Figure 1: Install application pop-up window

If the architecture is not displayed in your device, you can find it by visiting:

`http://<DEVICE_IP>/axis-cgi/param.cgi?action=list&group=root.Properties.System.Architecture`

Where `DEVICE_IP` is the IP address of your device. This will display the following line:

`root.Properties.System.Architecture=<DEVICE_ARCH>`

## Activating the license

This application requires a license to run. After installation, the application can be turned on, but if the license is not activated, it will not perform any tasks.

If the license is not activated, a message will appear on the application in the camera's UI, as shown in the image below.

The license can be activated through the camera's UI by clicking on the three dots next to the application and either selecting `Activate license with a key` (if you have a key file) or `Activate license automatically` if you have a license code.

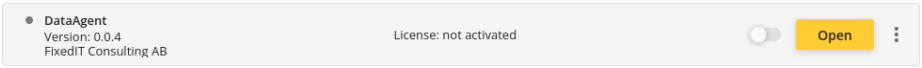



Figure 2: License not activated camera UI

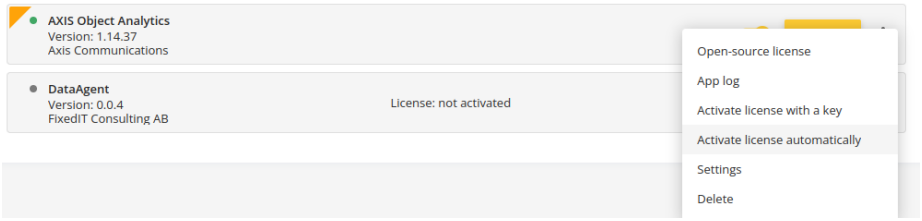



Figure 3: Activating the license options

Using automatic license activation is usually easier, but it requires the camera to be connected to the internet. If you have a license code but no license key, and the camera is not connected to the internet, you can generate a license key using the code at the Axis "License Key Registration" page.

After activating the license, a new text will show on the application with the license's validity period.

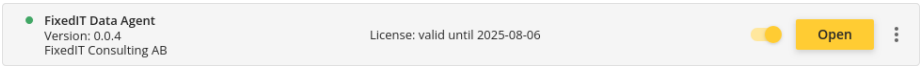



Figure 4: FixedIT Data Agent installed with license activated

If you use a yearly subscription, make sure to install a new license before the previous one expires to make sure the application does not stop working.

If you install a lifetime license, the text on the application will state `License: valid`, for a time limited license it will state `License: valid until <DATE>`.

## Configuring the application settings

By default, the application is configured to collect system metrics from the Axis device it is running in and push them to an InfluxDB/Grafana dashboard. To get this working, the only thing you need to do is to configure the InfluxDB instance details in the application settings.

The application settings can be configured before starting it or while it is running. This can be done from the camera's UI, in the application's `Settings` tab.

These parameters can also be configured in bulk through automation (for example, using VAPIX). This guide will focus on the most impor-

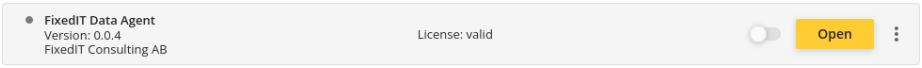



Figure 5: Application using lifetime license

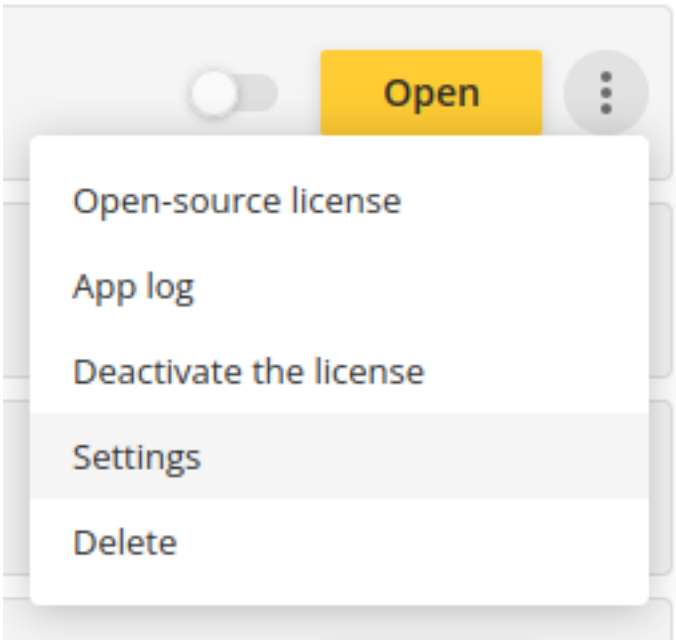



Figure 6: Settings tab

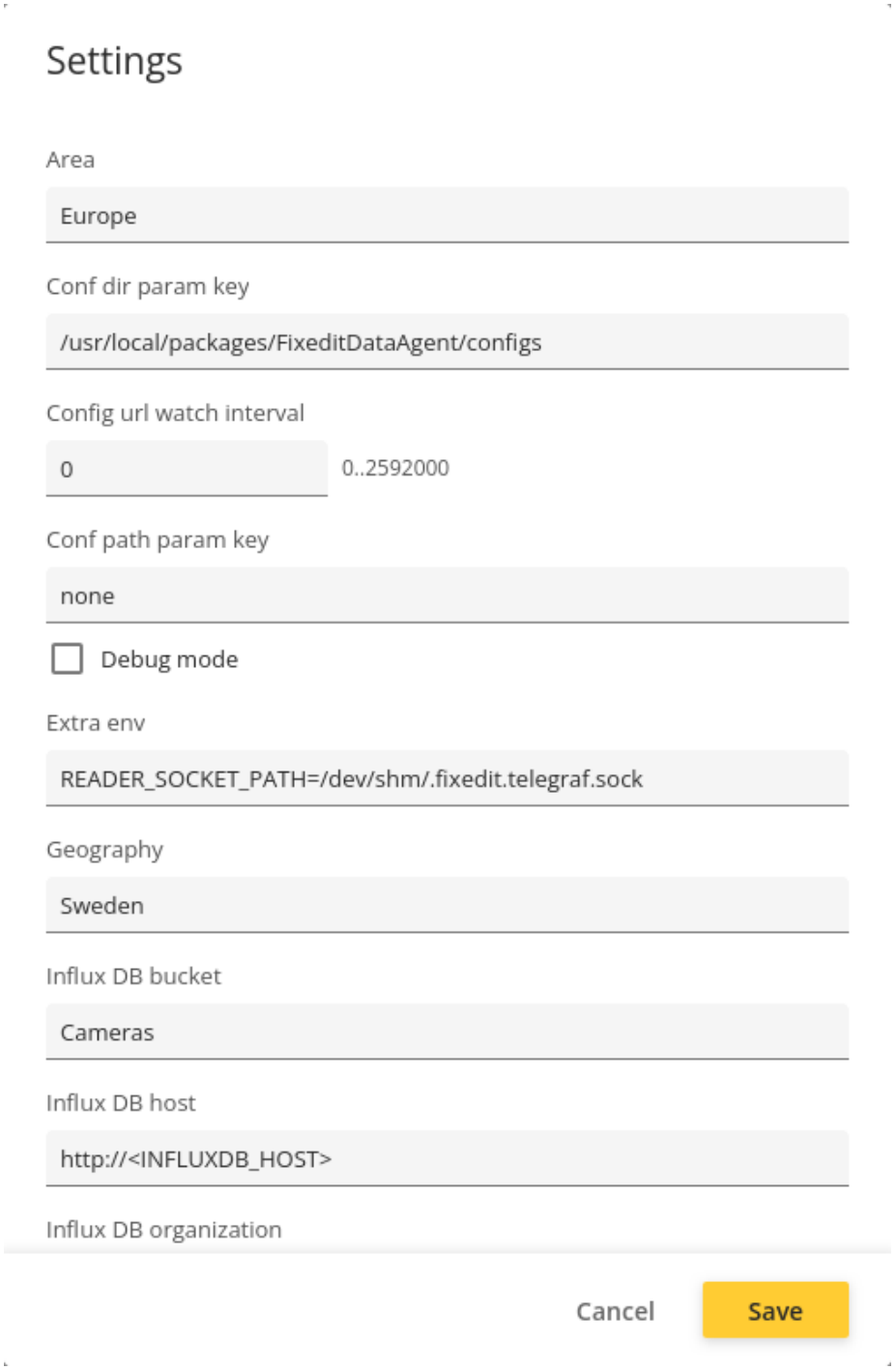



Figure 7: Settings window

tant parameters, but you can find a complete list of parameters in the `FIXEDIT_DATA_AGENT_CONFIG_SPEC.pdf` file.

### InfluxDB instance configuration

In order for the application to send data to the InfluxDB database, the InfluxDB instance details need to be configured in the application settings:

- `InfluxDBHost`: Hostname or IP address of the InfluxDB instance including the protocol (e.g. `http://192.168.0.123` or `https://influxdb.example.com`).
- `InfluxDBPort`: Port number of the InfluxDB instance.
- `InfluxDBToken`: Authentication token for accessing InfluxDB.
- `InfluxDBOrganization`: Name of the InfluxDB organization.
- `InfluxDBBucket`: Name of the InfluxDB bucket where metrics are stored.

### Device tagging configuration

To easier keep track of your Axis devices, we recommend using geo tags and type tags. These are added to the system metrics and can be used to filter devices in the dashboard. Despite our recommendations on how to use the geo tags, there is no enforcement and you are free to group the geography in whatever way makes sense to your business.

- `Area`: The top-most geo, e.g. `Europe`
- `Geography`: The second level, e.g. `Sweden`
- `Region`: The third level, e.g. `Stockholm`
- `Site`: The specific location, e.g. `Arena 5`
- `Type`: The type of installation, e.g. `Surveillance` or `Analytics`

### Debug mode configuration

By default, the Telegraf process will not log anything on successful sends of data to InfluxDB. By enabling the `DebugMode` option, you will see more detailed log messages in the application UI. The following image shows how the logs include messages such as `Wrote batch of 28 metrics in 147.209235ms` and `Buffer fullness: 0 / 100000 metrics`. These messages will be logged a few minutes after the application starts (during the first data sync).

- `DebugMode`: Enables or disables debug logging in Telegraf. When enabled, Telegraf will output more verbose logging information, which can be useful for troubleshooting configuration or connectivity issues.

## Starting the application

The application can be started by clicking the toggle in the UI.

You can check the status of the application from its UI.

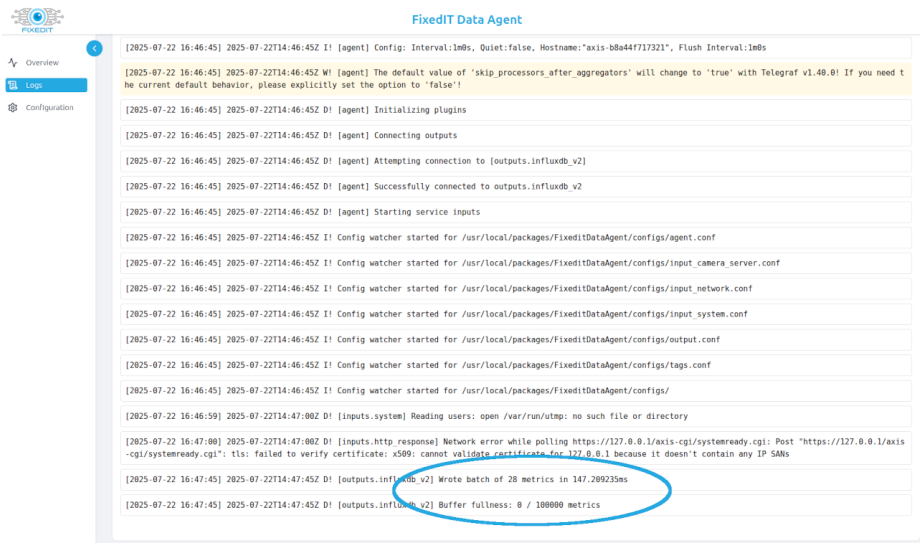



Figure 8: Debug mode enabled causing more verbose logging

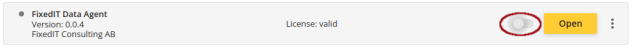



Figure 9: Toggle for starting and stopping application

After starting the application, it will wait for the interval specified in the `agent.conf` file before sending the first data package (currently 1 minute). This means that it might take several minutes before the first data is visible in the dashboard.

## Navigating the application user interface

The application's UI can be accessed by clicking the `Open` button in the camera's "Apps" section, as shown below.

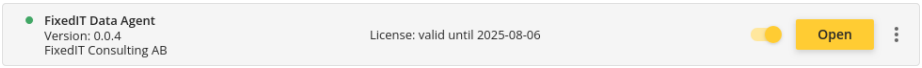
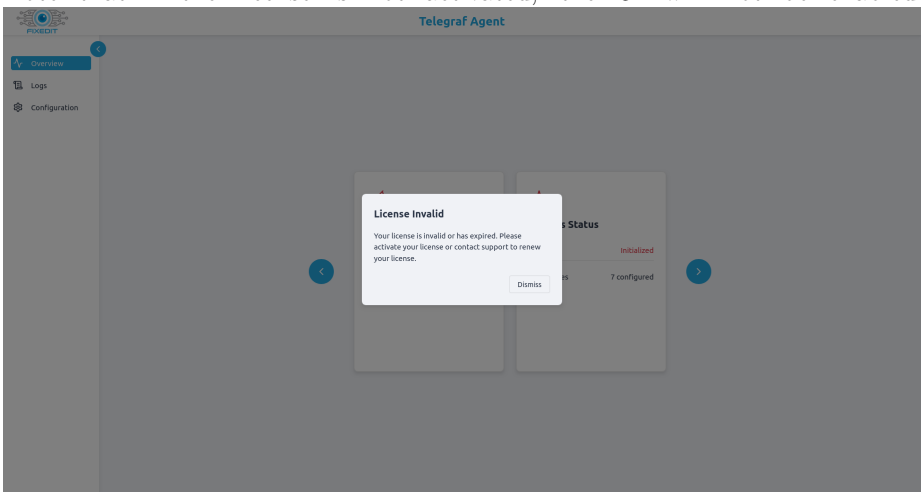



Figure 10: FixedIT Data Agent UI Open button

The UI can also be accessed at:

`https://<CAMERA_IP>/local/FixeditDataAgent/index.html`

Note that if the license is not activated, the UI will not be enabled:




To the very left, there is a navigation tab showing all the available pages: `Overview`, `Logs` and `Configuration`.

### Overview page

The default page that opens when accessing the UI from the Open button is the `Overview`.

The box on the left shows the License Status. For instructions on how to install a license, see Activating the license.

The box on the right shows the process status. This indicates the status of the Telegraf process. If everything is running correctly, the status will show as

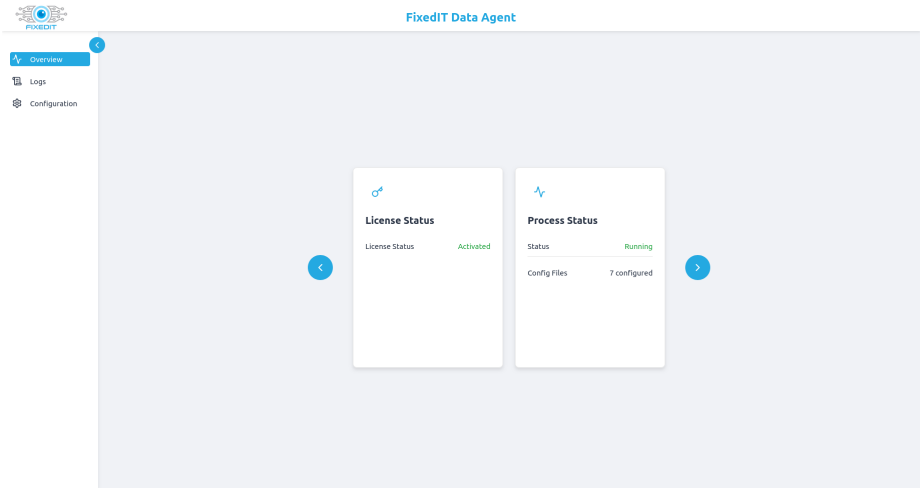



Figure 11: Overview page for FixedIT Data Agent

`Running` in green.

### Logs page

The `Logs` page shows the logs from the Telegraf process. The page is automatically updated with the latest logs, where the newest logs will show at the bottom. Errors are indicated with a pink background and warnings are indicated with a yellow background.

### Configuration page

The `Configuration` page can be used to manage the configuration files used by the Telegraf process. This includes viewing the content of the currently available configuration files in the camera, adding new files, removing files, or enabling/disabling files.

For more details on how to use the configuration page to add your own config files, read the Advanced application configuration section.

## Dashboards

As part of the FixedIT Data Agent, we provide access to an open source set of dashboards for Grafana to visualize the data sent from the FixedIT Data Agent. This section describes how to access the dashboards to visualize the system metrics from your devices.

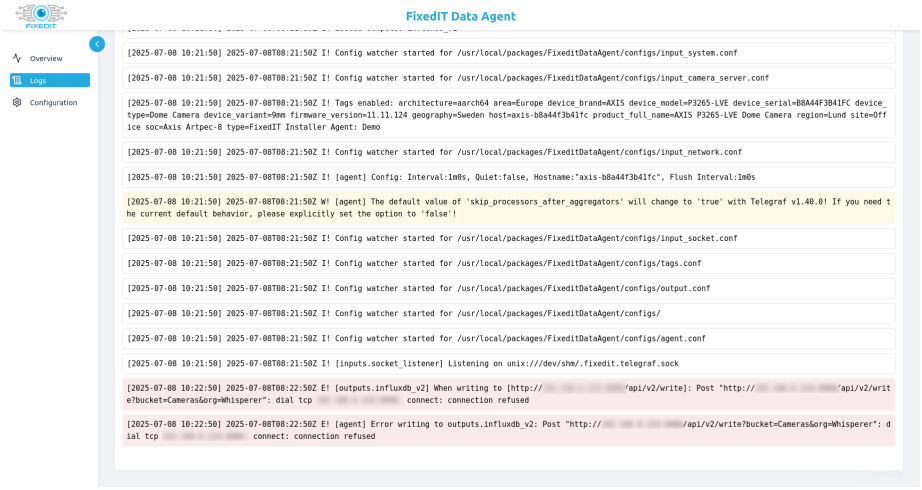



Figure 12: Example of errors and warnings in the logs page

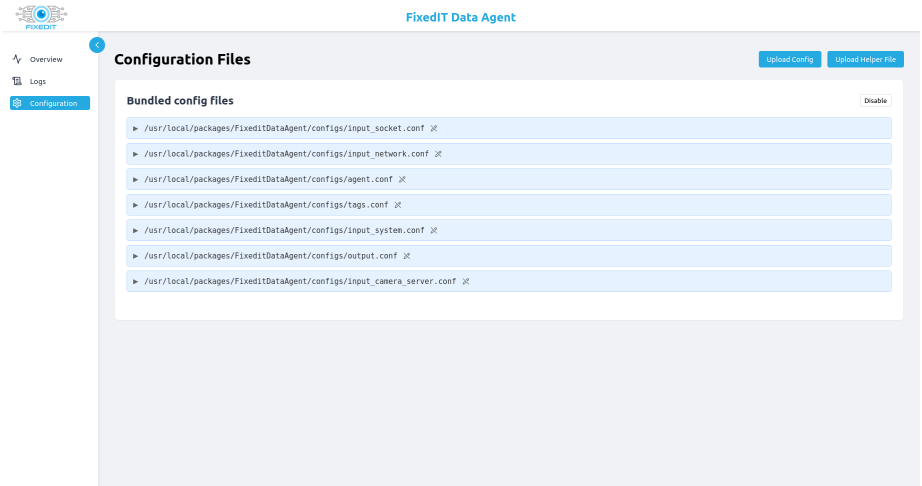



Figure 13: Configuration page

### Navigating the dashboards

After accessing the Grafana instance, you can check the available dashboards by clicking on the Grafana icon in the top left corner -> `Dashboards`.

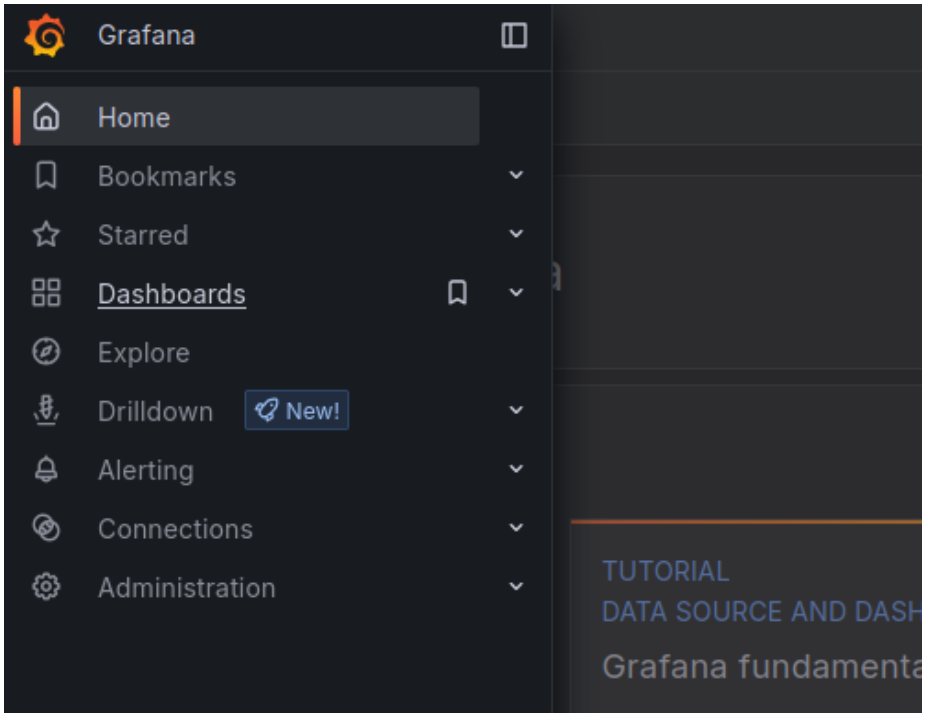



Figure 14: Dashboards tab in Grafana

Dashboards can be organized in different folders. In the example below, our dashboards are in a folder named `Cameras`.

### System overview

The System overview dashboard provides information about the entire fleet of devices that have sent data to the instance. This includes metrics such as the number of connected devices, recently lost devices (devices that have not reported any metrics during the selected time period), and outliers (devices that are the worst performers in different aspects, such as uptime or RAM usage). This dashboard intentionally does not display metrics individually for each device, as this would make the dashboard too cluttered to use in very large installations.

### Overview of devices

The Overview of devices dashboard provides detailed information about individual connected devices, including each device's last report date and time, history

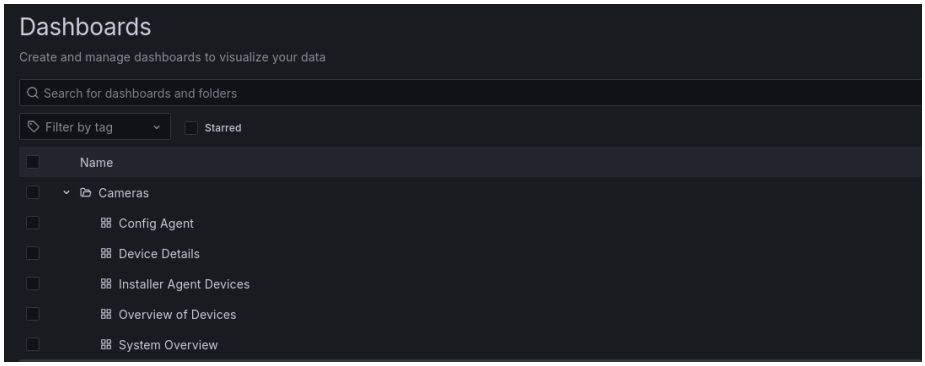



Figure 15: Folder containing our dashboards

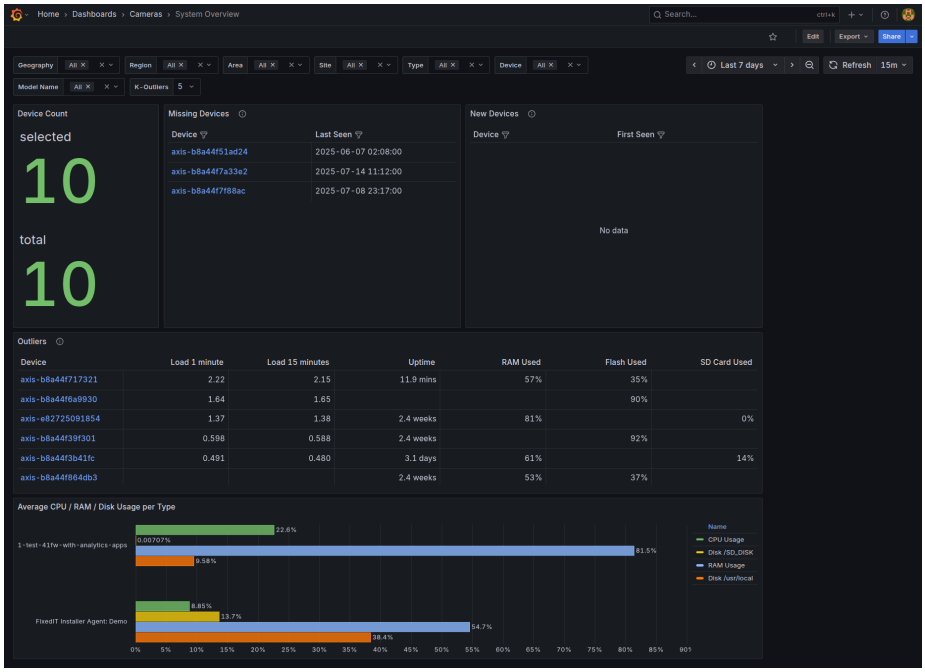



Figure 16: The “System overview” dashboard

of AXIS OS versions used, external and internal IP addresses, CPU and RAM usage graphs, and more.

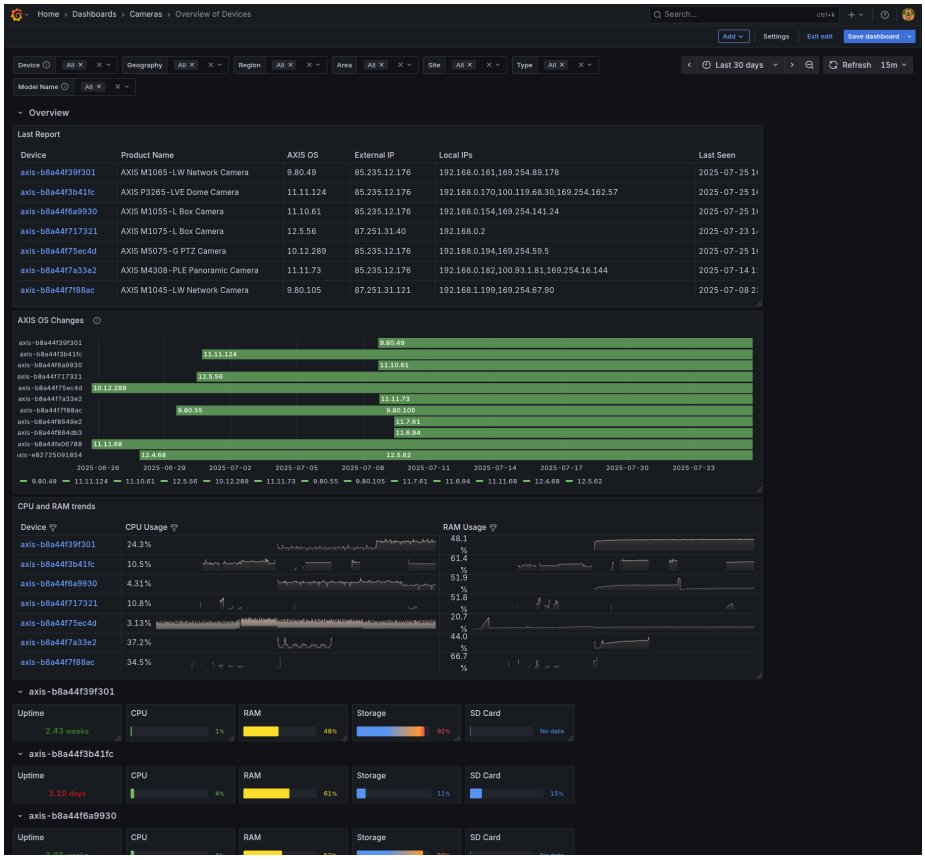



Figure 17: The “Overview of devices” dashboard

### Device details

The Device details dashboard provides information about a specific device. In both the `System overview` and `Overview of devices` dashboards, clicking on a device’s identifier opens the `Device details` dashboard for that particular device.

This dashboard provides a detailed view of the internal system metrics of a specific device, along with additional information such as its external and local IP addresses, geo tags (set through the `Settings` for the FixedIT Data Agent, as explained in the Device tagging configuration section), and DNS response time on the device’s network.

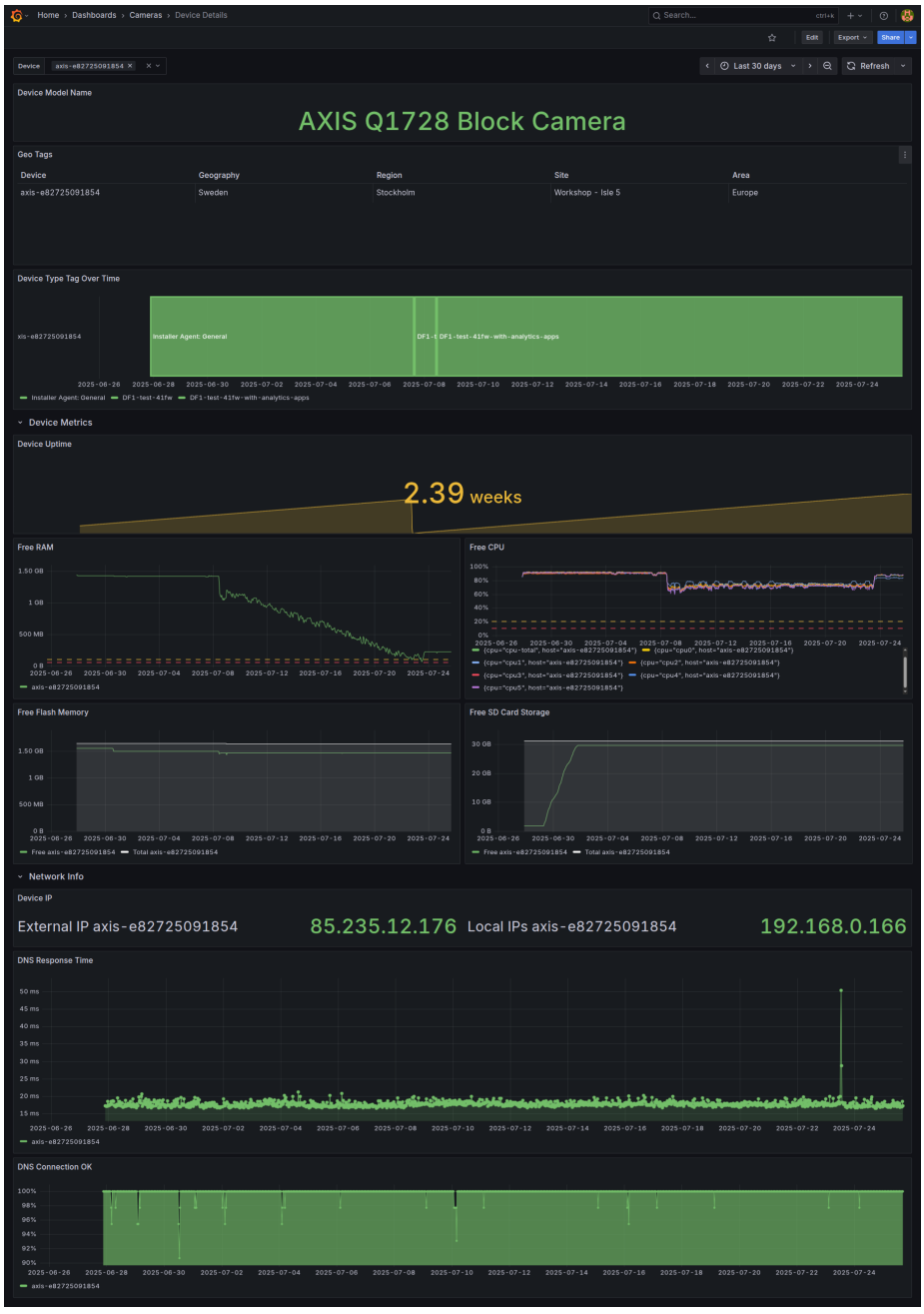



Figure 18: The “Device details” dashboard

### Self-hosting the dashboards

It is also possible to self-host the InfluxDB and Grafana stack, as long as the devices can connect to the instance. For more details about that, see the `README.md` file in the dashboard stack.

Note that these steps are not required if we have already provided a hosted setup for you.

### Importing dashboards to an existing Grafana instance

If you already have an InfluxDB/Grafana instance up and want to import the dashboards for the FixedIT Data Agent, you can do so from the `Dashboards` page.

Click on the `New` button -> `Import`.

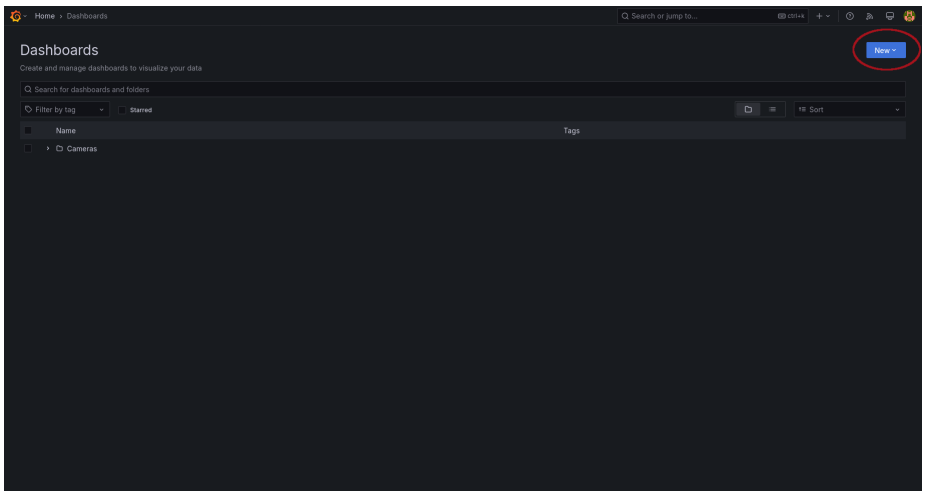



Figure 19: Dashboard page new button

This will open the `Import dashboard` page. You can either upload the JSON file for the dashboard you want to import or paste its contents.

Once you have uploaded your dashboard file, you can click `Load` to import it.

In the `Import dashboard - Options` page, you can choose the name of the dashboard and which folder you want to import it to, as well as its unique identifier. You can click `Import` to import it to your Grafana instance.

## Advanced application configuration

This section explains more advanced configuration of the application. This includes adding your own custom configuration files and setting custom environment variables.

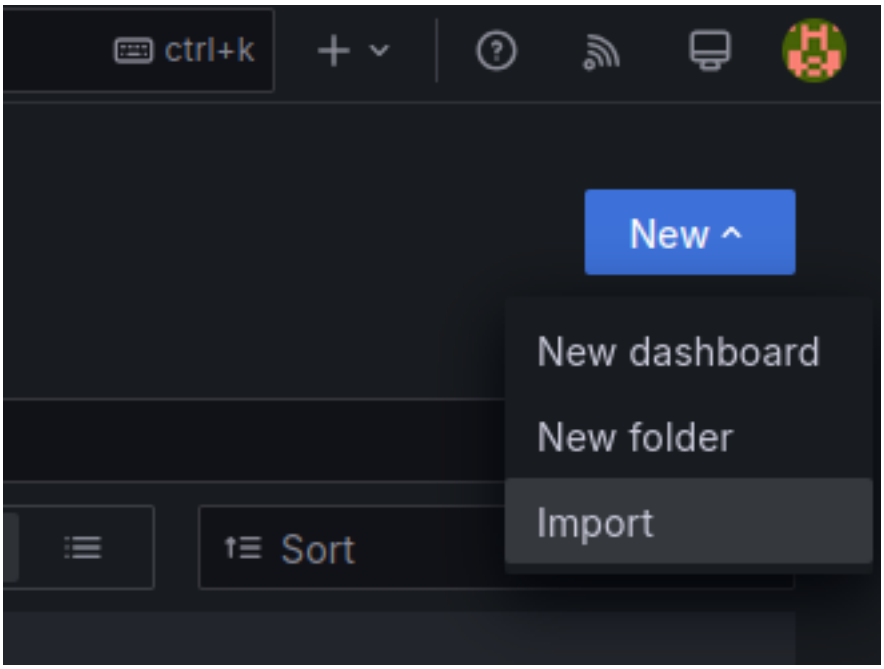



Figure 20: Import dashboard button

**Managing custom config files**

As explained above, the UI contains three different pages. The **Configuration page** can be used to manage configuration files, both the bundled ones and custom ones.

**Bundled config files**

Before any new files are uploaded, the page will only show the bundled config files in the UI. Files are listed using their full path. The icon of a crossed off pencil to the right of each file indicates that the file is read-only.

The content of the files can be viewed by clicking on the arrow to the left of each filename.

When the content of a file is viewed, the `Refresh` button also appears to the right of the filename. Clicking on this will update the content of the file shown in the UI to the latest version of the file in the camera.

These files can be disabled and enabled as a group, i.e. either all of them or none of them are used by Telegraf. The bundled files are updated when the application is updated, and the content and dependencies of the config files might be refactored. Only having some of the bundled files enabled might break functionality after an update, so to ensure compatibility, they are treated as a

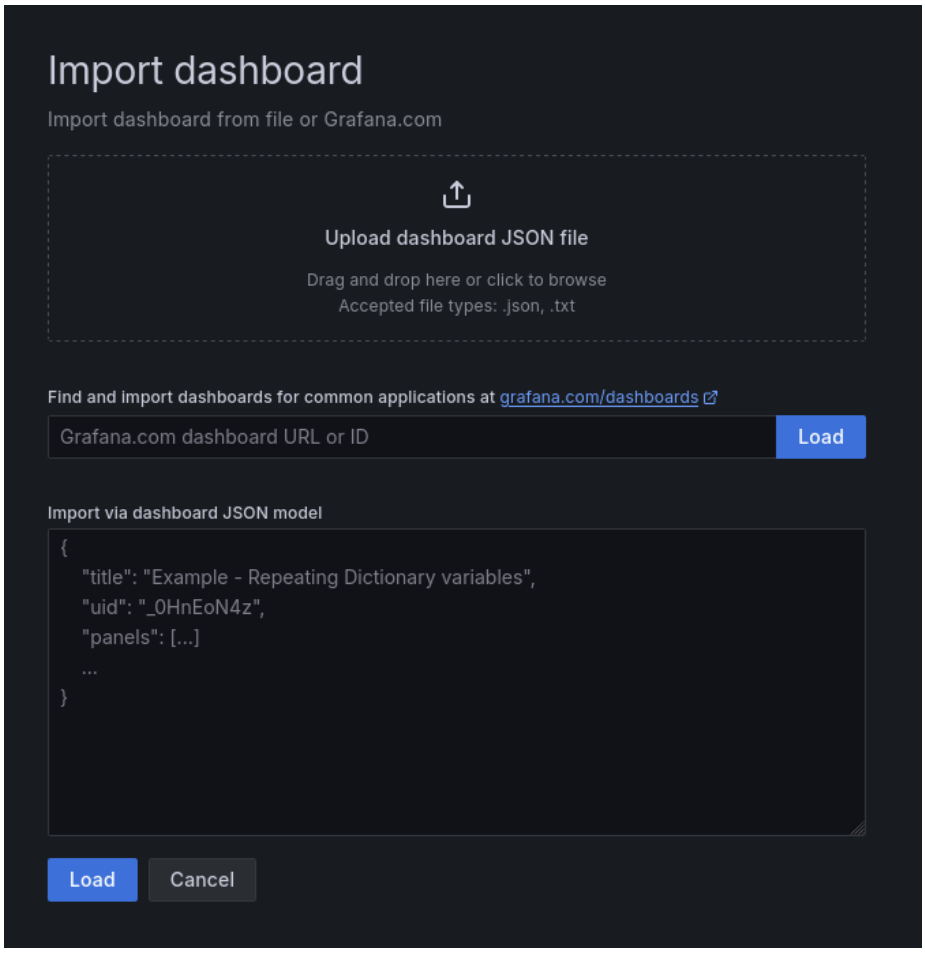



Figure 21: Import dashboard page

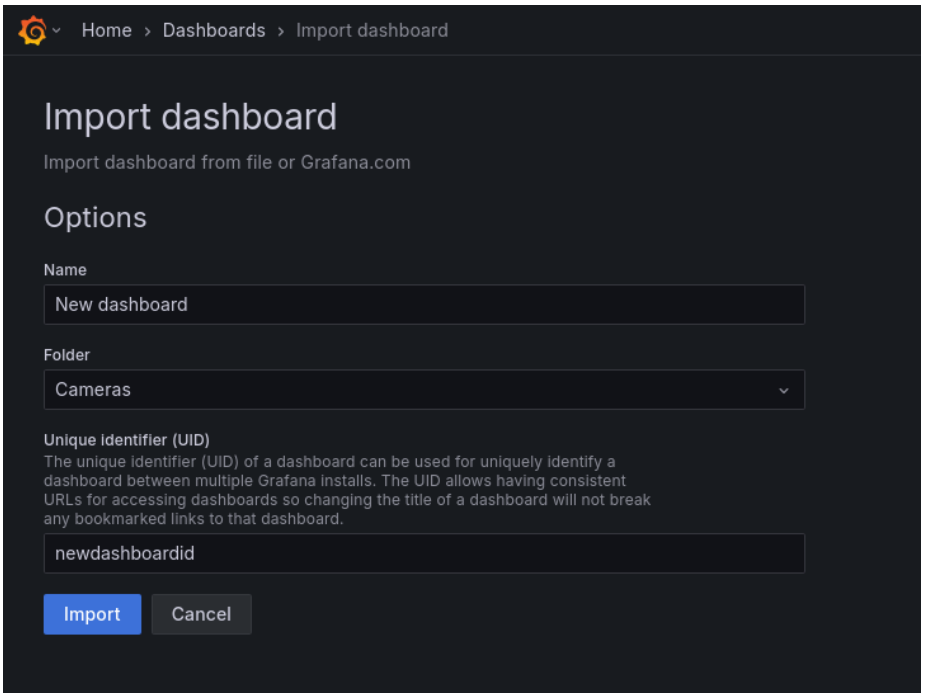



Figure 22: Import dashboard options

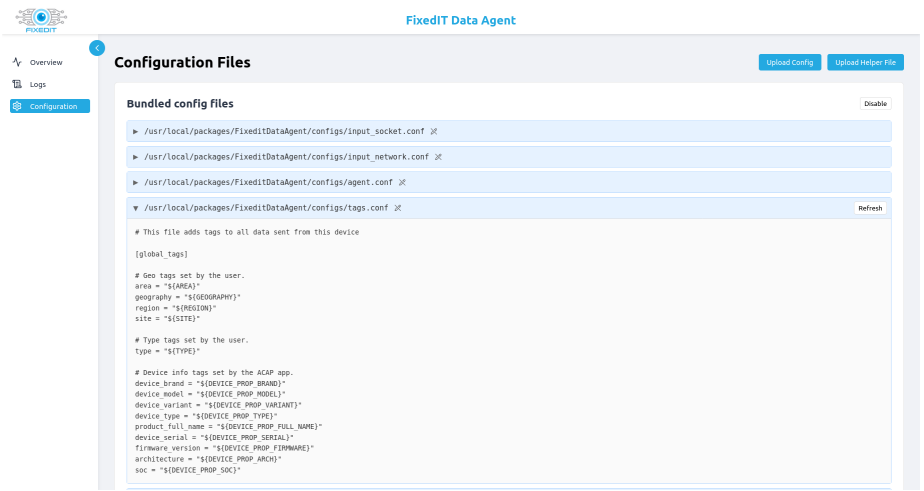



Figure 23: Config file content window

group. They can be disabled/enabled using the `Disable`/`Enable` button at the top right of the `Bundled config files` section. If you only want to use some of the bundled files, it is recommended to disable all bundled files and upload your own copies of the files you want to use.

### Uploading new config files

In addition to the bundled configuration files, you can upload new configuration files. This can be done by clicking the `Upload Config` button located in the top right corner of the UI. This will open up a pop-up window to choose a file to upload.

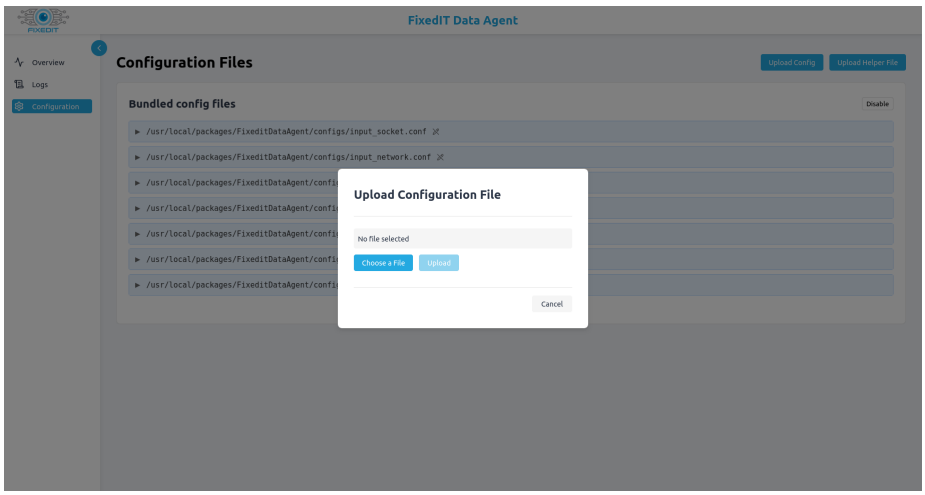



Figure 24: Config file upload pop-up window

After uploading the file, it will show up under the `Uploaded config files` section.

The pencil icon to the right of each file indicates that the file is modifiable.

Each uploaded config file will show two buttons to the right: `Delete` and `Enable`.

- The `Delete` button removes the file from the camera completely, after removing the file it will also be removed from the list in the UI.
- The `Enable` button will make Telegraf use the config file. By default, Telegraf will not use the uploaded file until it is enabled. The file can be disabled again after enabling it. Files that are currently enabled cannot be deleted, they need to be disabled first.

### Example use of a custom config file

For debugging purposes, you might want to see in the application's user interface what metrics are being sent to InfluxDB. This can be done by adding the following config file to the application:

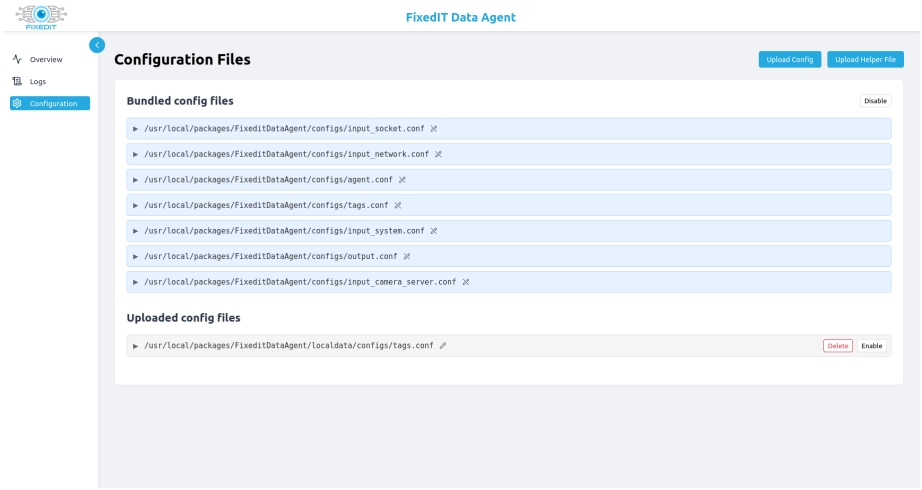



Figure 25: Uploaded config file

```
# Configuration for stdout output plugin
# This plugin prints the pipeline data to the console for debugging

[[outputs.file]]
  # Output file path (stdout)
  files = ["stdout"]

  # Data format to use
  data_format = "json"
```

This will print the metrics to the standard output of the Telegraf process, which will be captured by the application and can be seen in the `Logs` page of the application. The standard output is displayed first and colored with blue background.

Another use case for a custom configuration file might be if you also want to collect analytics data from a 3rd party application. You can do so by creating a config file that will pull data from the 3rd party application using their APIs (or by using an input that accepts data to be pushed to it). The following example shows how to pull occupancy data from AXIS Object Analytics. For this to work, you first need to start the AXIS Object Analytics application on the device and configure an occupancy scenario.

```
# This configuration file is used to collect occupancy data from AXIS Object
# Analytics. For this to work, you first need to start the AXIS Object Analytics
# application on the device and configure an occupancy scenario.

# You should replace the scenario number with the one you have configured in
```


Figure 26: Logs page with stdout output

```
# AXIS Object Analytics.

[[inputs.http]]
  name_override = "camera_occupancy"

  # Note that we have to use HTTPS here since the Axis devices does not
  # allow BASIC auth over HTTP and the Telegraf HTTP input plugin does not
  # support DIGEST auth. We need to allow insecure HTTPS connections to the
  # device since the devices by default use a self-signed certificate from
  # Axis.
  urls = ["https://127.0.0.1/local/objectanalytics/control.cgi"]
  insecure_skip_verify = true
  timeout = "5s"

  method = "POST"
  data_format = "json"
  headers = { "Content-Type" = "application/json" }
  body = '''
  {
    "apiVersion": "1.6",
    "context": "telegraf",
    "method": "getOccupancy",
    "params": {"scenario": 1}
  }
  '''

  # Update the credentials to match your device's credentials. A user
  # with 'Administrator' permissions is necessary.
```

```
username = "myadmin"
password = "myadminpass"

# Parse field from the nested `data` object and only
# keep the `total` field
json_query = "data"
fieldinclude = ["total"]
```

You can now create a new visualization in Grafana to display the occupancy data using the following query:

```
from(bucket: "${bucket}")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "camera_occupancy")
  |> filter(fn: (r) => r["_field"] == "total")
  |> group(columns: ["host"])
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
```

An example of a visualization of the occupancy data is shown below:

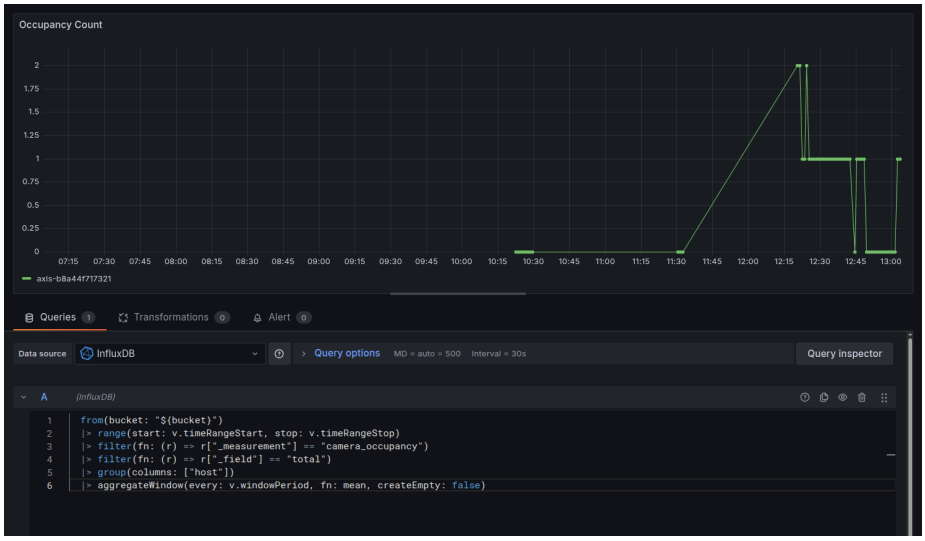



Figure 27: Example of a visualization of the occupancy data in Grafana

### Uploading new helper files

Some configurations might require additional files. A helper file is considered any file that is referenced in a configuration file and used by the Telegraf process. This means that helper files are not used directly by the application, the application only uploads the file to the device to make it accessible to the Telegraf process. This can be text files, binary files, executable files or anything else. These files can be uploaded by clicking on the `Upload Helper File` button

located in the top right corner.

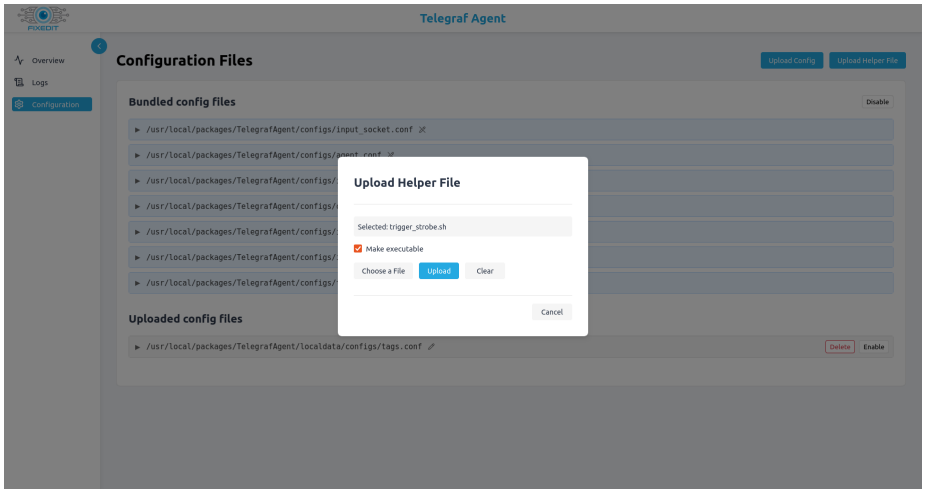



Figure 28: Helper file upload pop-up window

The file can be made executable by checking the `Make executable` option. Once uploaded, the file will show up under `Uploaded helper files`. Executable files will be shown in green, as shown below.

The pencil icon to the right of each file indicates that the file is modifiable. The code icon indicates that the file is executable.

Like for config files, a `Delete` button will appear to the right of each uploaded helper file. Clicking on this removes the file from the camera and UI.

Helper files are never used directly by the application, only by the config files. Therefore, they do not have `Enable`/`Disable` buttons.

**Example use of config files with helper files**

Let's take the following config file as an example.

```
# Configuration for executing a script
[[outputs.exec]]
  # Command to execute when metrics are received
  command = ["${HELPER_FILES_DIR}/trigger_strobe.sh"]

  # Data format to use for input
  data_format = "json"
```

It runs the `trigger_strobe.sh` script when metrics are received. The script needs to be uploaded as a helper file. All helper files are uploaded to the same directory in the camera, and the application is setting the environment

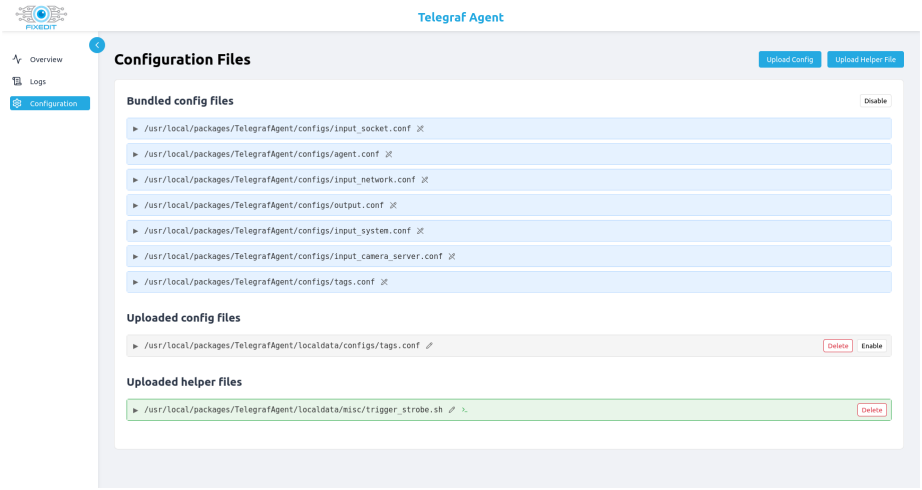



Figure 29: Uploaded helper file

variable `HELPER_FILES_DIR` to this path. This variable can then be used from configuration files as seen in the example above, in the following line:

```
command = ["${HELPER_FILES_DIR}/trigger_strobe.sh"]
```

For a more complex use case example involving config files and helper files, see this video.

### Setting custom environment variables

Environment variables can be configured in the application and will be readable from the Telegraf configuration files. Some environment variables used frequently have their own parameters for simplicity, as seen in the Configuring the application settings section, but more can be specified using the `ExtraEnv` parameter in the application's `Settings` tab.

- `ExtraEnv`: User defined variables, key-value pairs separated by ";". E.g.: `KEY=value;KEY2=value2`. These will be set in Telegraf's environment and can be used from the configuration files.

In the Configuring the application settings section, it is explained that setting some of the parameters also sets a corresponding environment variable (e.g. setting `InfluxDBHost` will set the `INFLUX_HOST` environment variable). Note that those environment variables must not be specified in the `ExtraEnv` parameter.

The application automatically sets some environment variables like `HELPER_FILES_DIR` and `HOME`. These variables must not be set in the `ExtraEnv` parameter. For more details about automatically set environment variables, see the `FIXEDIT_DATA_AGENT_CONFIG_SPEC.pdf` file.