

Kibana and Taranta Integration

The scope of this document is to study the integration between Kibana and Taranta. Taranta is a web application that allows users to create a graphical user interface to interact with Tango devices and see the [documentation](#). Kibana is a user interface that allows users to visualize Elasticsearch data and navigate the Elastic Stack.

Introduction

The current Taranta version (v. 1.3.1) does not have a native widget to see logs coming from Elasticsearch. In the following report, we explain the possible solutions to integrate the SKA logs into Taranta to permit SKA Tango developers to use a single dashboard to test the devices and see the logs.

In the document, we explain how the Elasticsearch logs are used by SKA developers and how it is possible, the impact in terms of performance and layout, and the limitations to visualize logs using an existing widget in Taranta. Furthermore, we show possible ways to extend Taranta functionalities to add more functionalities to use Kibana more efficiently.

Logs in the SKA context

In the SKA context, all the logs coming from every element (such as a Tango device, a pipeline, or a GitLab runner) are stored in a common Elasticsearch repository hosted in the SKA cluster. Users can visualize and explore the logs with Kibana, which is installed in the same cluster. It is also possible to query directly to Elasticsearch, for example in Grafana a tool to visualize logs has been configured using an Elasticsearch query.

For now, in the SKA project, the use of Kibana is not very intense, mostly Matteo Di Carlo and the system team are using it for debugging. We believe it will be a very useful tool for all in the future. Occasionally, also CSP-LMC developers use it for debugging devices in the integration environment. Probably, other teams also use logs, but we have no direct evidence of this.

Logs in SKA are currently used to:

- to check logs of pods as we check logs in the local development setup
- To understand the status of the pod through logs
- To see the logs generated by the pipelines
- To see the logs generated by the application hosted in the pods
- All the logs generated in the application that run into the cluster, are collected in the same Elasticsearch database and available in Kibana

More details about the section are available in the following document.

[W Kibana_Usage.docx](#)

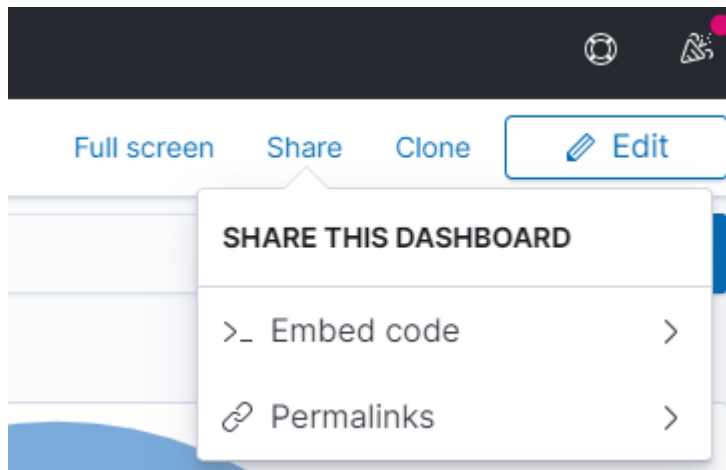
Integrate Kibana using the Embed page Taranta widget

Even though Taranta hasn't got a native widget to visualize logs, we studied how to utilize the Embed page widget to import a Kibana page. In the following sections, we explore how to import a Kibana page, its pros and cons, and limitations.

Export Kibana dashboard

Visualization is the core Kibana features classic graphing interfaces: pie charts, histograms, line graphs, etc. Placing together various visualizations on one dashboard panel creates a more straightforward data overview.

This generates iFrame code for the Kibana dashboard. Users can mark "Top menu", "Query", "Time filter", and "Filter bar" checkboxes if this needs to be shown in iFrame. This iFrame can be used in HTML files to show the Kibana dashboard.



```
<iframe src="https://k8s.stfc.skao.int/kibana/goto/90906f7455f6e4b30033ae73c00e49b6" height="600" width="800"></iframe>
```

Permalink generates a URL/link which can be used to open the given dashboard from anywhere. Also, this can be used to share dashboards with users or can be used in an iFrame src attribute.

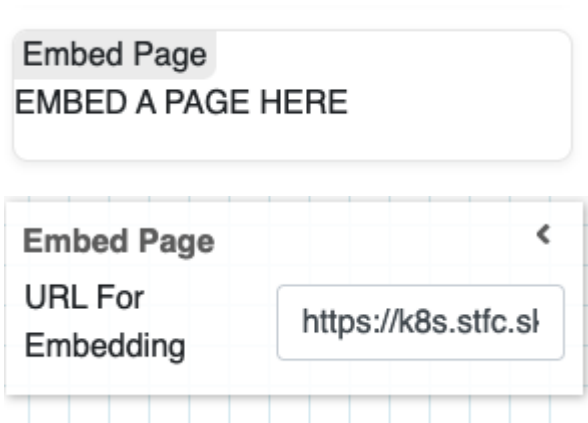
In Taranta dashboards we can use this link as Url for Embedding to view the Kibana dashboard exported

More details about the section are available in the following document.

Using Taranta to configure and save the Kibana object

Once exported a Kibana object using the procedure shown in the last section, Taranta is able to host a Kibana page inside a Taranta dashboard using the Embed page widget.

To import the link copied it is required to use the Taranta Embed Page widget and paste the link in the URL For Embedding in the Inspector, during Edit Mode.



Note

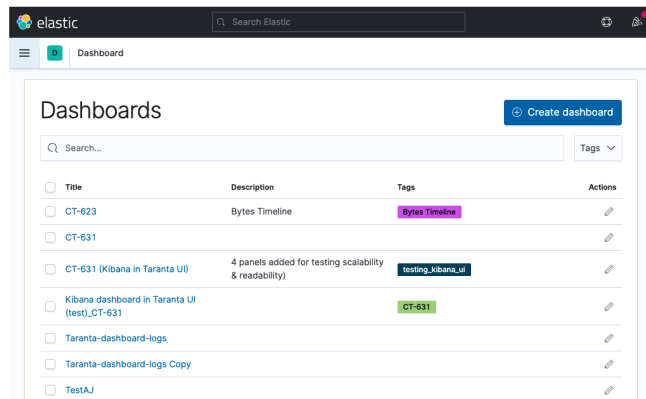
In Edit mode it is not possible to use the page, but it is necessary to run the dashboard in order to use it

We can define 3 configurations:

- configuring a dashboard
- configuring a query in discovery mode
- configuring a filter in a dashboard

Configuring a dashboard

To configure a dashboard it is necessary to go into the Dashboard section and click on Create a Dashboard.



In this way, the user can create and configure a dashboard directly inside Taranta

Note

Taranta stores only the original link so, if a user creates and opens a new dashboard when s/he clicks on edit and run, Taranta restores the link saved in the Taranta configuration. It means that the new dashboard is properly created and viewed, but the URL in the Taranta configuration is not updated: it will restore the original dashboard visualization.

Configuring a query in Discovery mode

Clicking in discover mode it is possible to discover the logs stored in the Elasticsearch repository and create a filter or perform some full-text search.

🏠 **Home**

Recently viewed ▼

Taranta-dashboard-logs

CT-631

New Dashboard

TestCream



Analytics ▼

Overview

Discover

Dashboard

Canvas

Maps

Machine Learning

Visualize Library

📄 Search

+ Add filter

EDIT FILTER [Edit as Query DSL](#)

Field: Operator:

Create custom label?

Pop

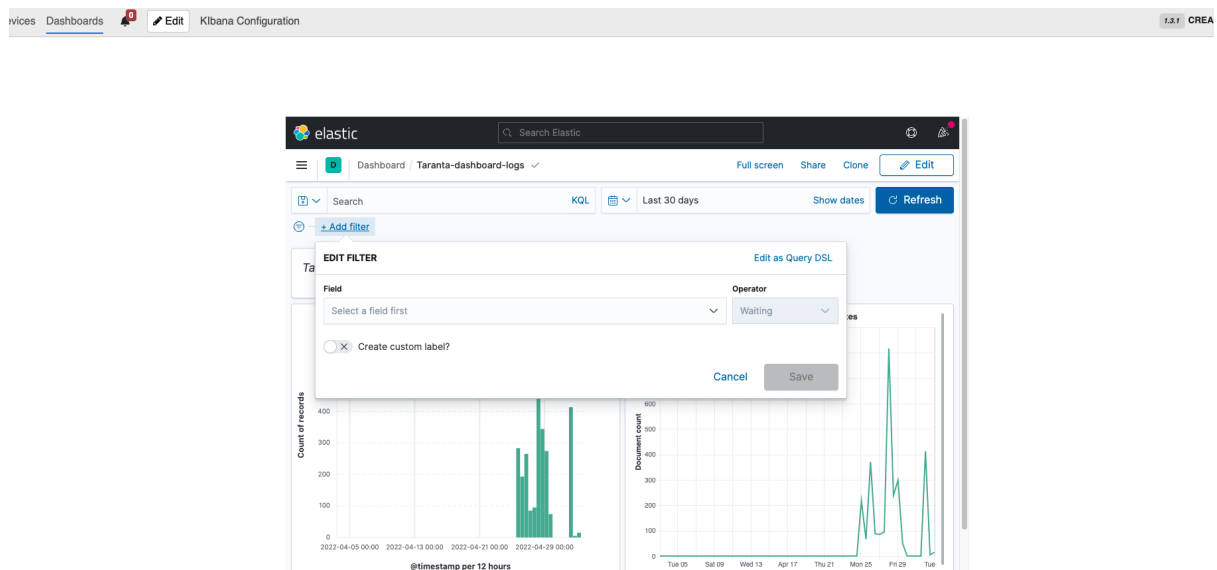
- log.file.path
- message

Time Document

Also in this case there is no difference between using Kibana through a Taranta dashboard or directly in the browser

Configuring a Filter in the dashboard

It is also possible to configure a filter in a dashboard. In this case, we used a dashboard created for this story named “Taranta-dashboard-logs” and imported it to Taranta.

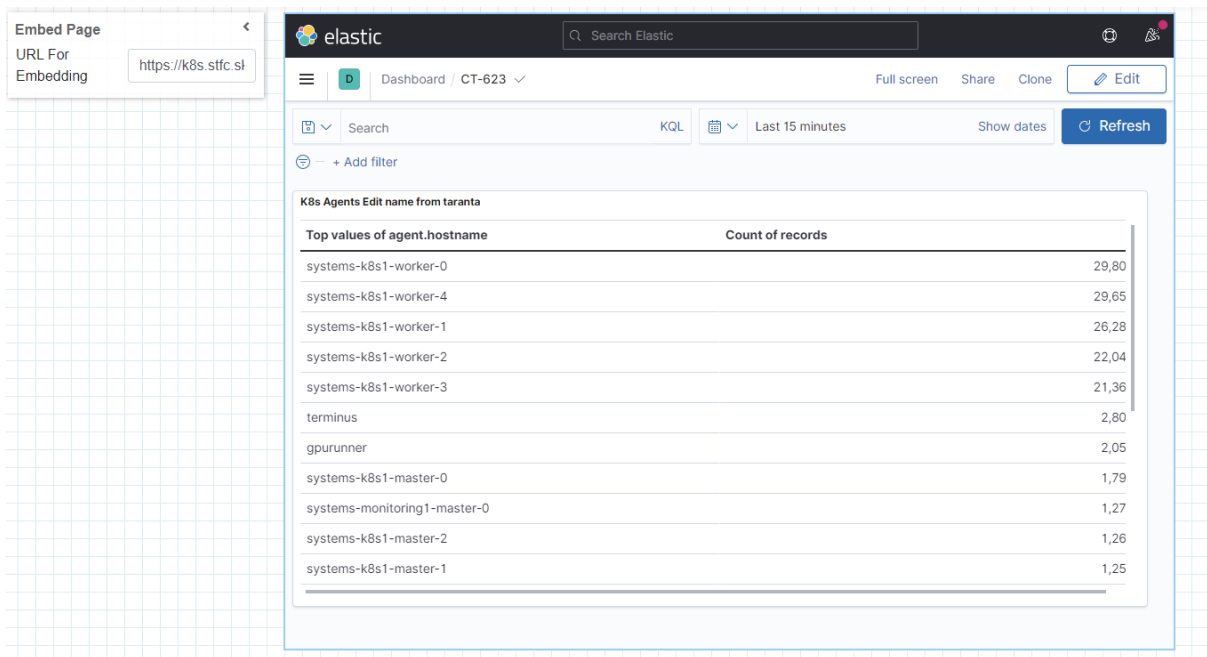


Also for this case, there is no limit to using Kibana inside Taranta or directly in the browser.

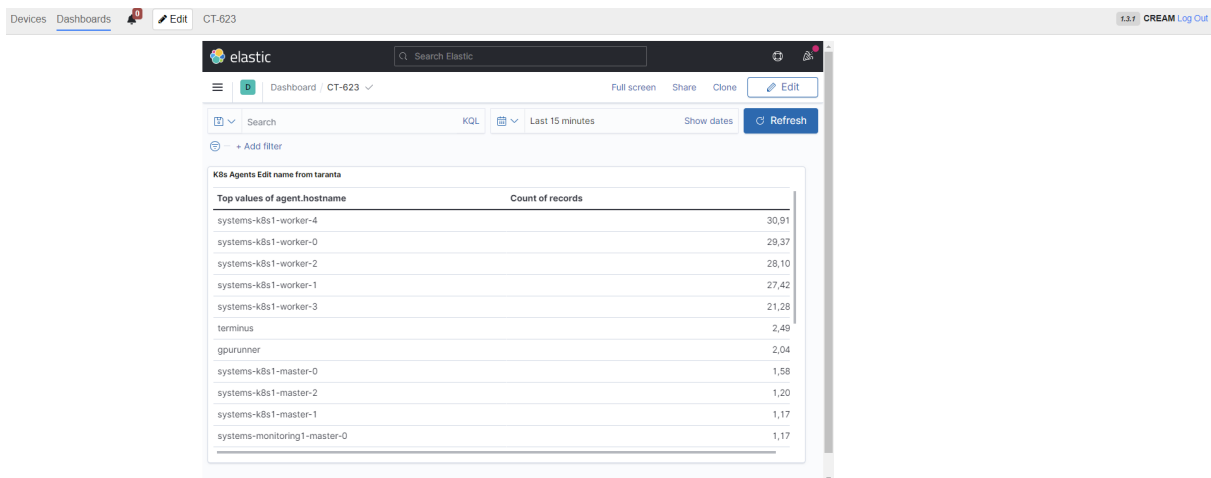
Save states from kibana

1. Insert an embedded widget on taranta and set the URL for example to [https://k8s.stfc.skao.int/kibana/app/dashboards#/view/c77fe700-c487-11ec-b915-ed8fd26275d0?g=\(filters%3A!\(\)%2CrefreshInterval%3A\(pause%3A!t%2Cvalue%3A0\)%2Ctime%3A\(from%3Anow-15m%2Cto%3Anow\)\)](https://k8s.stfc.skao.int/kibana/app/dashboards#/view/c77fe700-c487-11ec-b915-ed8fd26275d0?g=(filters%3A!()%2CrefreshInterval%3A(pause%3A!t%2Cvalue%3A0)%2Ctime%3A(from%3Anow-15m%2Cto%3Anow)))

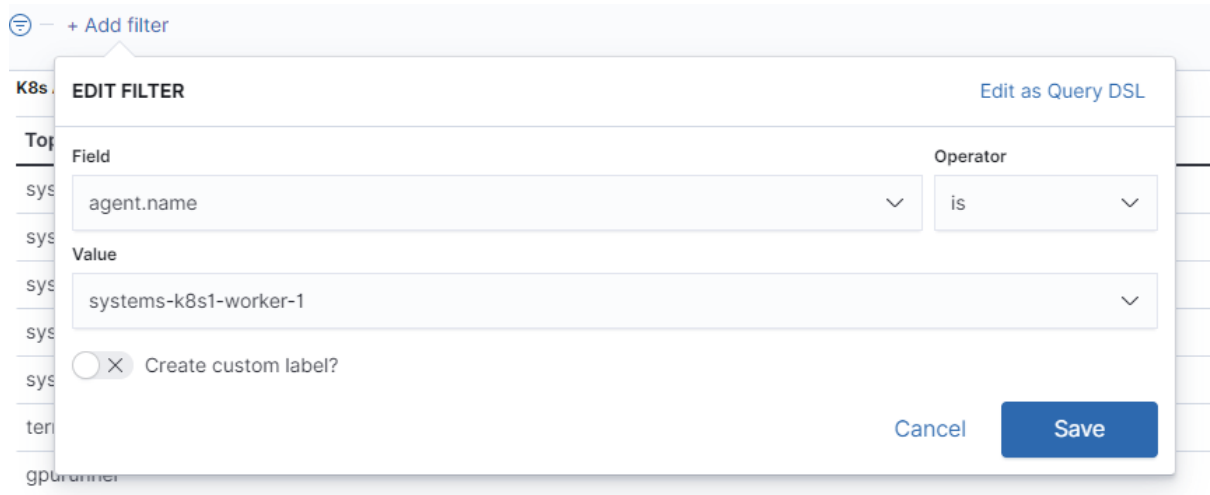
This is the output:



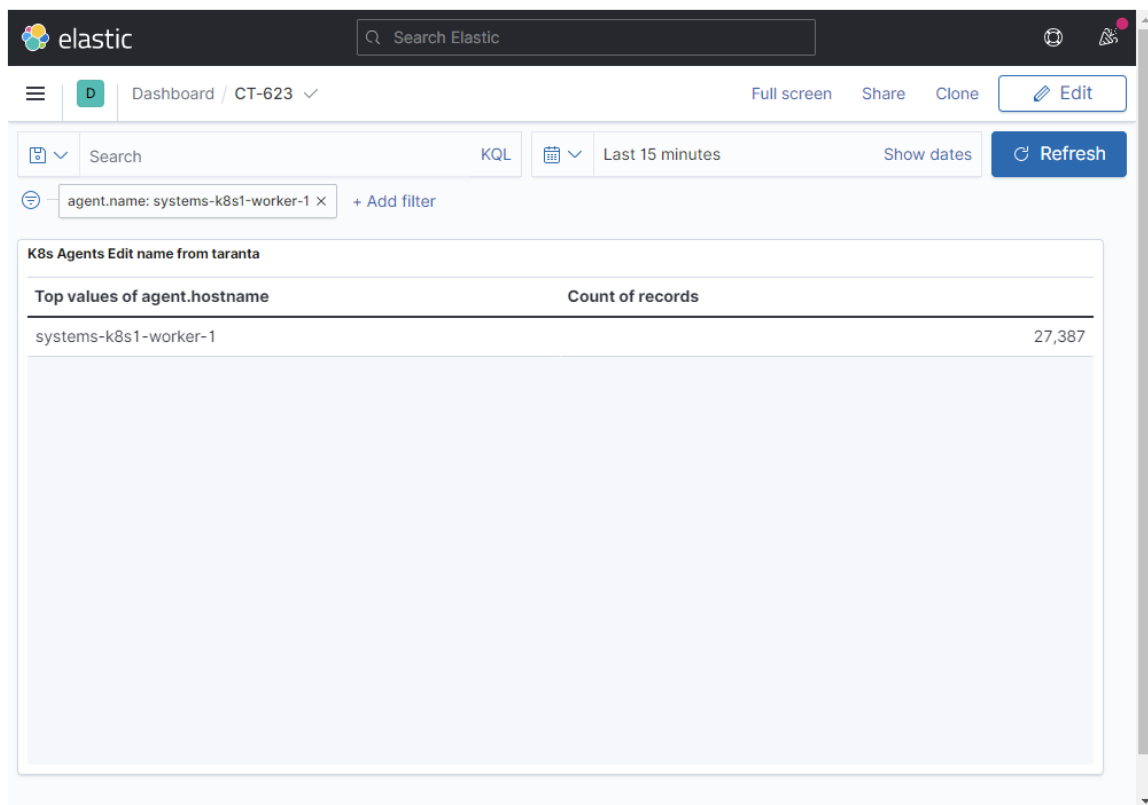
2. Start the dashboard:



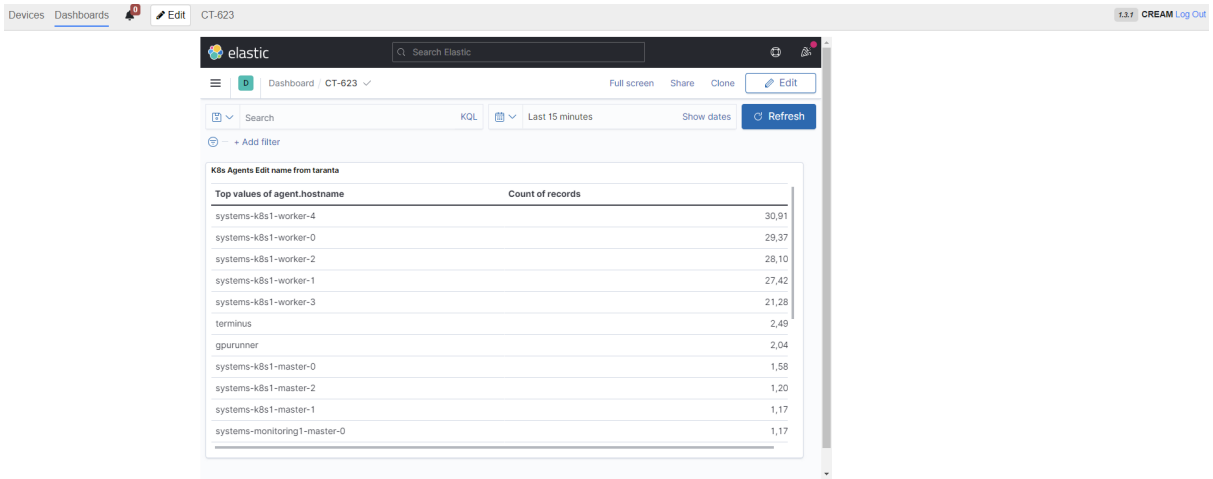
3. Click add filter



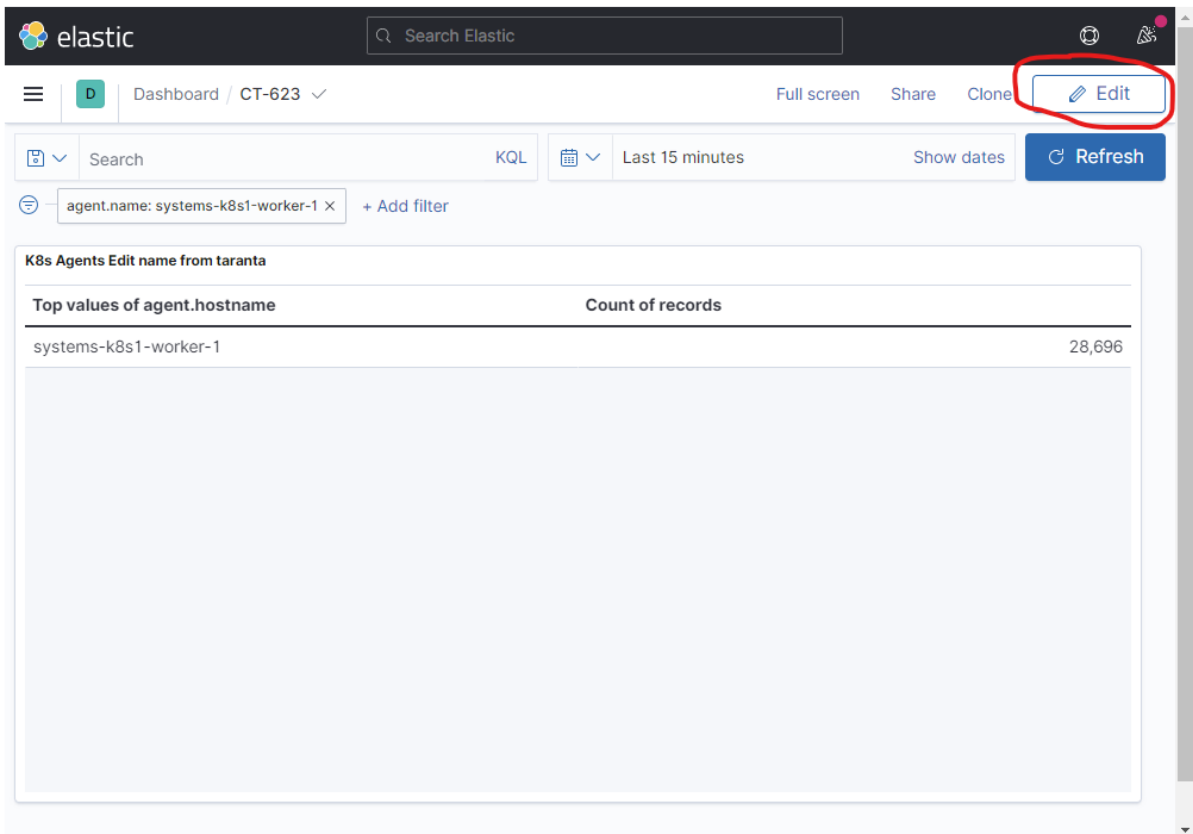
4. Save



5. If now you refresh the page all changes will be gone and the filter is not really saved, if you just press refresh you will see the first run image:



6. You need to click edit:



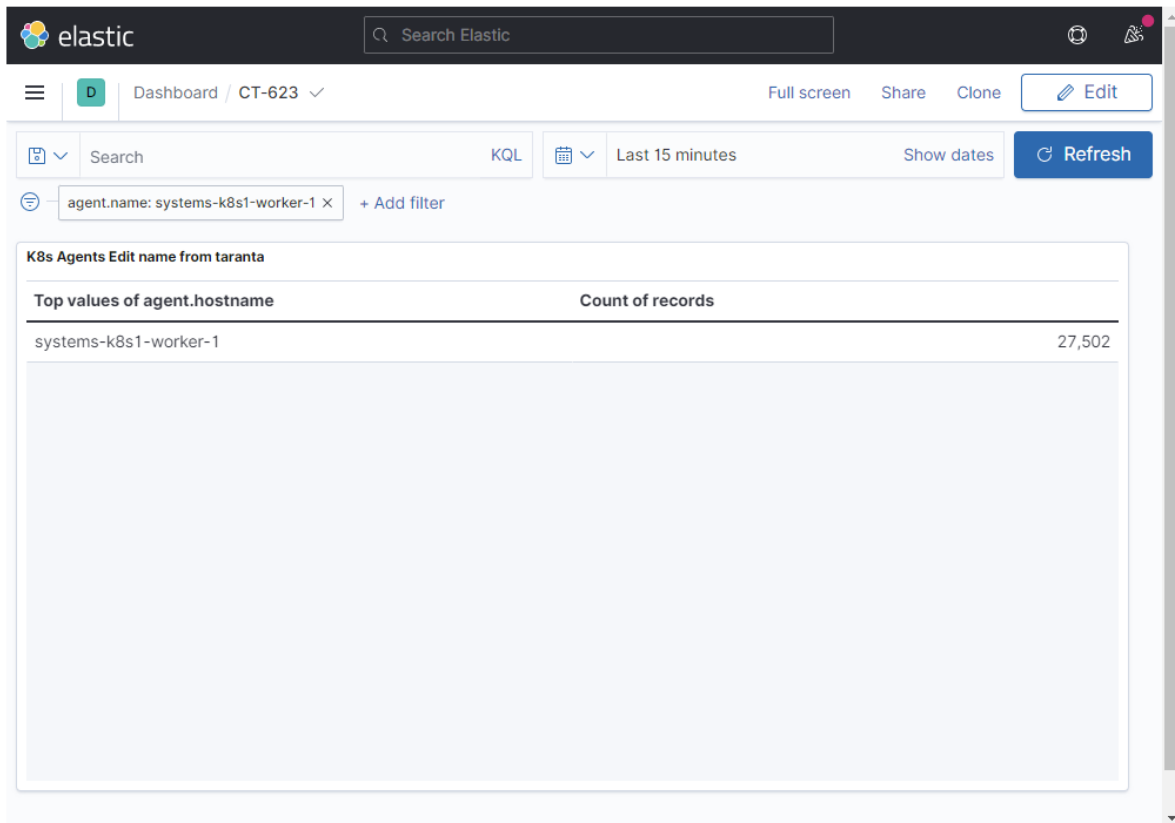
7. Click save:

The screenshot shows the Elastic dashboard interface. At the top, there is a search bar with the text "Search Elastic". Below it, the navigation bar includes "Dashboard / Editing CT-623", a green "Unsaved changes" indicator, and buttons for "Options", "Share", "Save as", "Switch to view mode", and a "Save" button which is circled in red. The main content area features a search bar with "agent.name: systems-k8s1-worker-1" and a "+ Add filter" button. Below this are "Create panel" and "Add from library" buttons. The primary widget is titled "K8s Agents Edit name from taranta" and displays a table with the following data:

Top values of agent.hostname	Count of records
systems-k8s1-worker-1	27,812

Now if you refresh the page the filter will be present on the widget:

This screenshot shows the same Elastic dashboard interface after a refresh. The "Save" button in the top navigation bar is now a simple button with a floppy disk icon. A confirmation message box is visible at the bottom right of the dashboard, containing the text: "✓ Dashboard 'CT-623' was saved". The main widget content remains the same as in the previous screenshot.



In general, we can say that Kibana has no clear configuration limits when it is hosted in a Taranta dashboard. The only concern is the URL generated in this mode is not stored. If the user configures and opens a different link, Taranta will not save the navigation and reopen the initial URL in case of a refresh.

Environments

Since Kibana is running also on the STFC cluster and is protected by GitLab authentication, accessing Kibana from localhost is not possible, it would need an open port for public access. But since Taranta is also running on STFC, developers can access directly from it to Kibana without any special privileges.

More details about the section are available in the following documents


- [CT-622 - Investigate what properties we can use to configure kibana under Taranta an...](#)
- [CT-623 Investigate save states from kibana configuration](#)

Taranta performance impact

We conducted a performance analysis to test the impact of Kibana hosted by Taranta.

From the performance tests it's possible to conclude that although there are some known issues with using IFrames, they are not impacting Taranta software in a meaningful way. There is some time addition of loading time in using the IFrame, but in general, it seems very fast, this might also relate to both Kibana and Taranta being deployed on the same server, highly reducing ping and loading times between the two software.

Test and iFrame issues are listed in the following document:

 [CT-630 Investigate how kibana integration impacts taranta performance](#)

Taranta layout impact

We conducted tests to study how Kibana impacts the general layout and what are the limitations to use it with iFrame.

Implementing the Kibana dashboard into the Taranta dashboard as UI is possible without limitations. Automatic and manual testing done in the different browsers and for the devices with the different screen resolutions give a proof that working with such UI gives the expected result:

- data can be viewed properly
- some limitations in viewing data refer to the way of scaling relevant data panels from Elasticsearch
- those limitations are mainly relevant to the number of data panels used in one row at the Kibana dashboard (no more than 2 panels is the best solution)
- Since we do not consider using a number of the different screen resolutions for dashboard operators/users (usually an organization provides its employees with standard devices), these restrictions will not have much impact in terms of usability.

Positive outcome

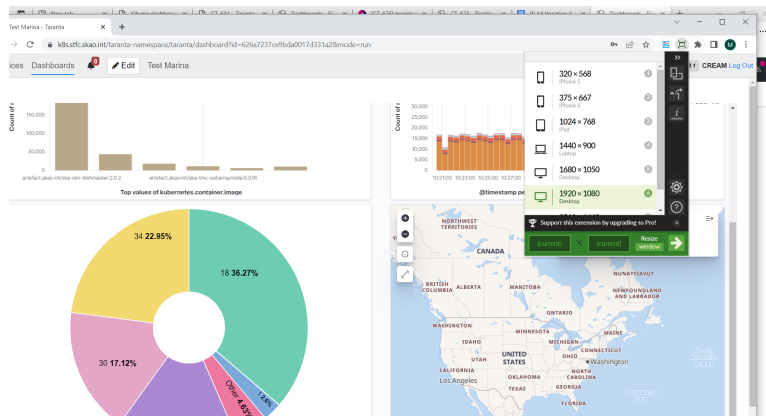
Using the Kibana dashboard has a large advantage on the ease of implementation as Elasticsearch is based on the drag-and-drop principle.

It will save time and coding efforts.

Additional advantage: using Elasticsearch will allow users to work with the rich query library provided by Elasticsearch.

Limitations

Testing the Taranta dashboard with 2 panels added in 1 row is possible but produces weird results as its content in Chrome is pushed to the left side of the browser.



To exit Full-screen mode (to come back to the Normal view): the user needs to be aware of where a corresponding button (button **Exit full screen**) is located. It is at the left bottom corner of the Kibana dashboard. The user has to know this & find this.

Testing the Taranta dashboard with 2 panels added in 1 row, but not filling the full length of the dashboard brings 2 browser scrollbars to view the full content & not considered good usability.

The user has to remember all the time which scrollbar refers to which dashboard.

To view the content of the Kibana dashboard he has:

- to go to the end of the Taranta dashboard by using the scrollbar for maximum
- then s/he will be able to catch the Kibana dashboard scrollbar to rescale it to the full view mode
- only this way Kibana dashboard`s content will be accessible.

This issue requires a specific solution.

Details and tests are available in the following document

[CT-631 Investigate How Kibana layout plays with Taranta layout display](#)

Authentication and authorization

Kibana provides different methods for authentication and authorization based on users and roles and methods like:

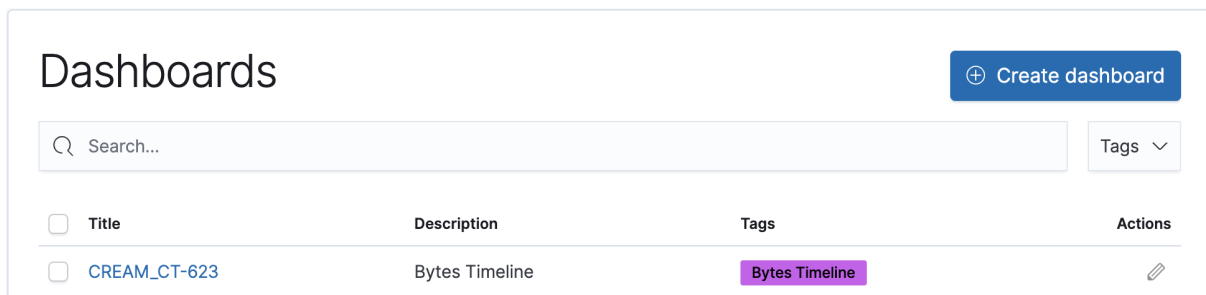
- Multiple authentication providers
- Basic authentication
- Token authentication
- Public key infrastructure (PKI) authentication
- SAML single sign-on
- OpenID Connect single sign-on
- Kerberos single sign-on
- Anonymous authentication
- HTTP authentication.

In the current SKA implementation, we cannot use Kibana users under the current software restrictions. It means that we cannot provide access to the Kibana dashboards, protected by username and password through Taranta.

The root of the problem is that no user can be added to the Kibana dashboard at the moment so everyone can edit or delete someone's dashboard. Authentication is entrusted by GitLab. If the user is inside the GitLab SKA group, he will be able to access/ edit /delete all the dashboards.

Most of the requests from actual users allow Kibana to be embedded in an iframe without login.

Currently, the way to solve this problem is by bringing some rules such as: adding a prefix (for example the team name) in the Kibana dashboard name.



In the future we can have 2 options:

- to check if this approach is satisfying enough (we will collect the users' feedback)
- to add user/user roles.

More details and ways to implement different authentication systems are listed in the following document

[CT-625 Investigate how kibana links \(urls\) can be used to integrate it into taranta](#)

Future implementation

In order to improve the current prototype that we created using native widgets, we studied Kibana API and how to extend existing widgets. It is possible to extend the attribute logger widget. We also studied Kibana APIs in order to understand what can be implemented in the SKA scenario.

Attribute Log Widget

This widget is used to display the attribute value of a device continuously. This shows a list of all previous values of the selected device attributes with the most recent value at the bottom.

Recent short_scalar 231

Time	Log Message
2022-05-09 18:00:53.693	235
2022-05-09 18:00:54.701	235
2022-05-09 18:00:55.710	231
2022-05-09 18:00:56.727	231

In our Taranta implementation, we can ask the user to input the query in the inspector panel and we can make API calls to the elastic server.

The image shows two screenshots of the Kibana interface. The top screenshot shows the Configuration panel with the Kibana URL set to `https://k8s.stfc.skao.int/kibana/app/` and the Show specific fields checkbox unchecked. The Query String panel shows a query string with a time range filter. The bottom screenshot shows the Configuration panel with the Kibana URL set to `https://k8s.stfc.skao.int/kibana/app/` and the Show specific fields checkbox checked. The Query String panel shows a query string `agent.name : systems-k8s1-worker-1, ...`. The inspector panel in both screenshots displays log messages with fields like `@timestamp`, `agent.ephemeral_id`, `agent.hostname`, `agent.id`, `agent.name`, `agent.type`, `agent.version`, `container.id`, and `container.image.name`.

Attribute logger shows logs of its previous values and the Kibana log feature shows log information of pods, pipelines, and applications. We need to poll the elastic server every x seconds to pull the latest

logs. This feature can be extended to the Attribute log with the same effort of creating a new widget for the Kibana log.

If we do not want to embed (insert) a Kibana page in a Taranta dashboard we can create a new widget for making a query to the Kibana/Elastic server. This will require some effort in making API calls to the elastic server, processing the JSON response, and rendering it on the UI. The effort for creating this new widget is evaluated to be around **3 FP**.

More details are available in the following document

 [CT-629](#)

Kibana APIs

The API study is useful if we want to extend the Taranta functionality in order to integrate Taranta with Kibana without using the User Interface.

Currently, there is not a clear Kibana user scenario for SKA, because not many users are using Kibana right now. We made some hypotheses, starting from an interview with Matteo Di Carlo, who is the System Team member who mainly uses Kibana to see logs.

The number of Kibana APIs is really high. The approach we adopted is to understand what could be the user scenario for a Taranta end-user and identify the APIs that satisfy such needs.

In the SKA user scenarios, there are some functionalities useful, such as:

- managing objects including dashboards, visualizations in order to retrieve the information stored in Kibana into the SKA cluster;
- import and export dashboard. It is useful if the user wants to migrate a set of dashboards created in a local cluster to the SKA cluster
- Create a short URL API. It permits creating a short URL also if the configuration is complex. If the URL is too long, it can exceed the limit of an URL that can be imported into Taranta
- Task Manager health API. In this way, it is possible to check Kibana's health status and alert the user that something should be wrong. Otherwise, the dashboard imported cannot be visualized without feedback from Taranta.

Other useful APIs are:

- APIs to retrieve a single Kibana saved object. It is useful to guarantee that the Kibana dashboards saved are retro compatible if Kibana will be upgraded
- API for user session validation. It will be useful if SKA will implement users in Kibana. In this way, if a Taranta user is linked to a Kibana user, when the user performs the logout in Taranta, also the Kibana session expires

The list of the APIs, documentation links, and how to use them is available in the following document [📄 CT-627 Investigate Kibana API export data only \(NO UI\)](#)

Summary

Kibana is a tool that helps developers to debug devices, especially when they run in an integrated environment. Taranta can facilitate developers to build dashboards to aggregate Tango device attributes and commands and logs in a single view.

Although Taranta has no native widgets to visualize logs, based on our findings, the Embed page widget allows users to integrate Kibana dashboards with a few limitations described in the document and it is a good tradeoff to have a fast way to visualize logs inside Taranta.