Monitoring Namespaces with Grafana

This guide describes how developers can use Grafana to monitor resource usage and server performance of their projects.

Conceptual Overview

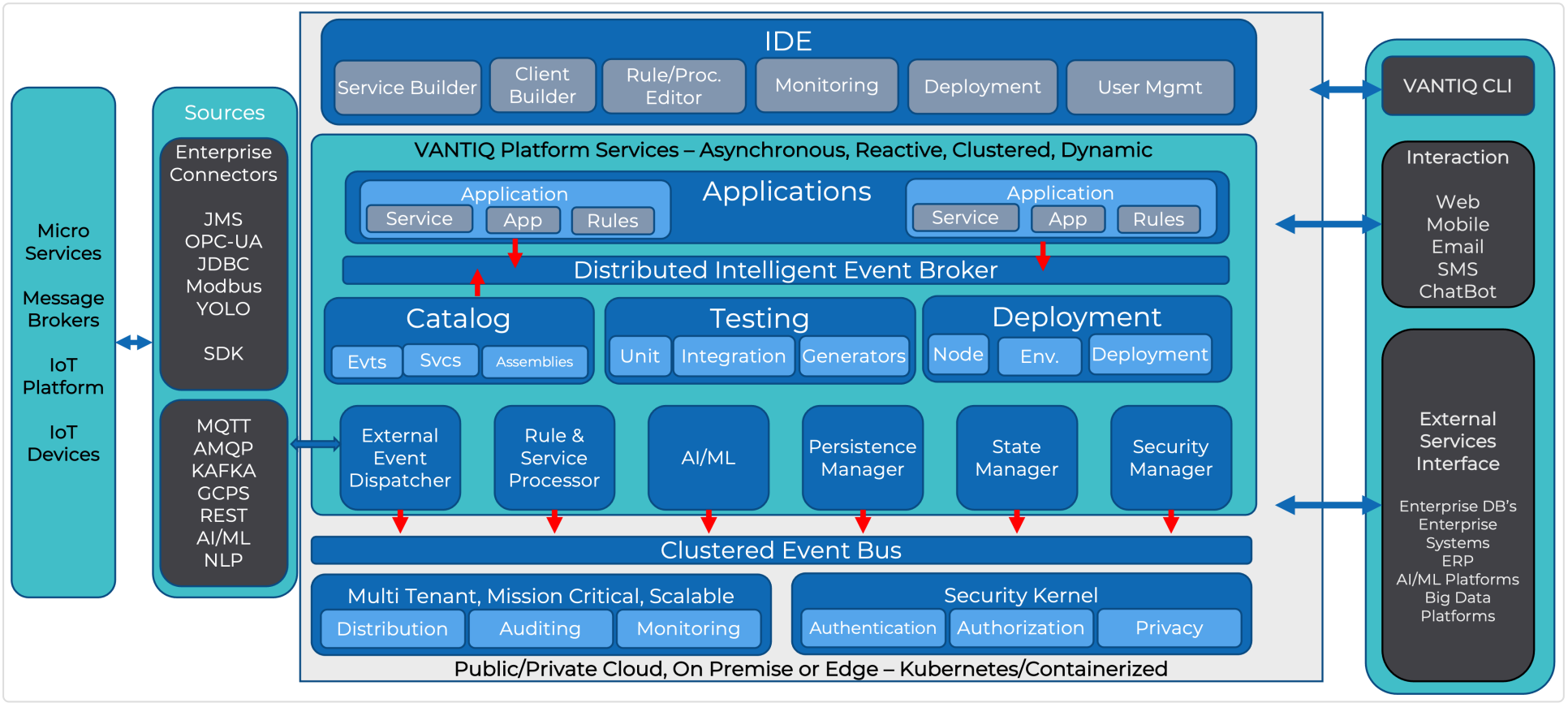
The Vantiq cloud-based platform features several tools to make run-time application activity transparent to developers, from the Design and Development through to Operations and Maintenance phases of the software development life cycle. One of these tools is a 3rd-party browser-based server performance monitoring service called Grafana. Developers and administrators access Grafana dashboards to observe in real-time how their applications are performing under load.

A specific configuration of Grafana is embedded into the Vantiq platform for each scope of Vantiq user:

- **Developers** view Grafana dashboards for namespace metrics
- **Organization Administrators** access Grafana charts and statistics across all namespaces in the Organization
- **System Administrators** see Grafana charts and statistics across all organizations

This guide will focus on Namespace-level Grafana metrics for Project Developers. Grafana offers different features and dashboards for each scope, but knowing how to use Grafana to observe resources usage and performance within a single Vantiq Namespace is foundational to understanding the Organization and Installation system scopes.

Within the Vantiq Platform, system metrics and monitoring capabilities work through the interaction of several functional components:



A clustered event bus traverses server entities, under load from application systems, feeding metrics from them to a monitoring time-series database. Grafana's graphical dashboards and statistics are visual representations of pre-configured query results against this database, accessible to an application developer from the IDE.

Grafana Usage

Developers primarily use Grafana to:

- Discover consequential performance bottlenecks in their application systems
- Plan capacity during development, testing and ongoing operations
- Confirm the continued smooth operation of deployed applications

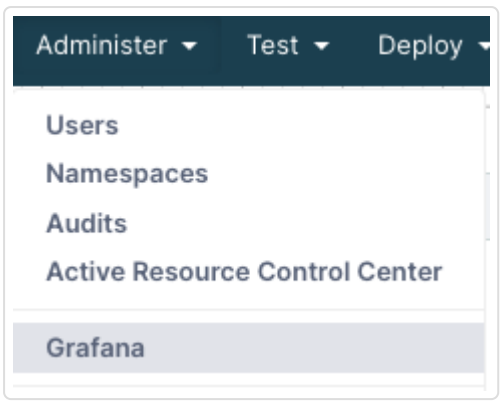
Grafana brings to light the largest impacts on scalability and latency including:

- Database DML statements
- Inbound and Outbound event rates
- Computational costs (related to message size and content)
- Internal processing delays

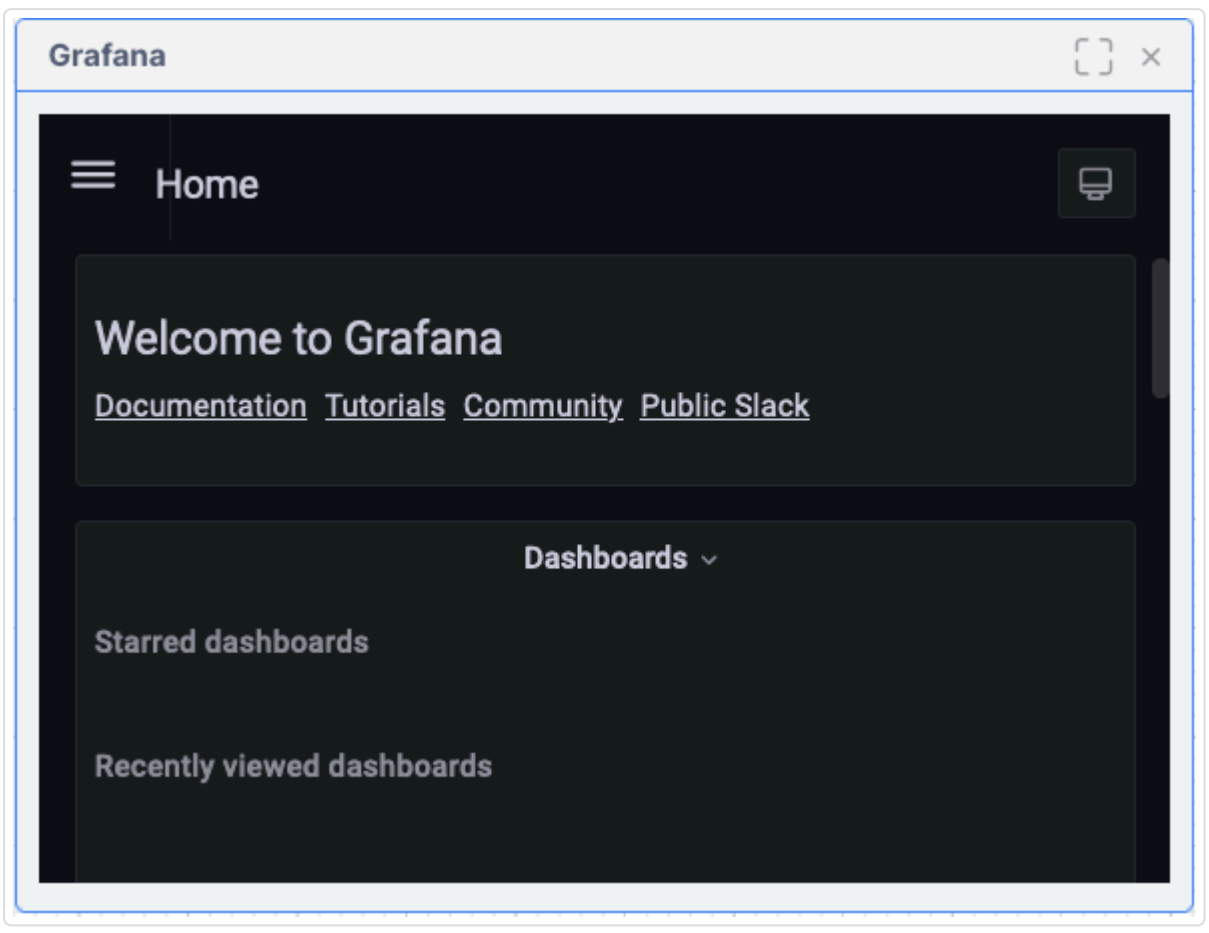
Note: Grafana is only available for clustered installations - Vantiq public and private clouds. It is not supported on Edge servers.

Getting Started with Grafana

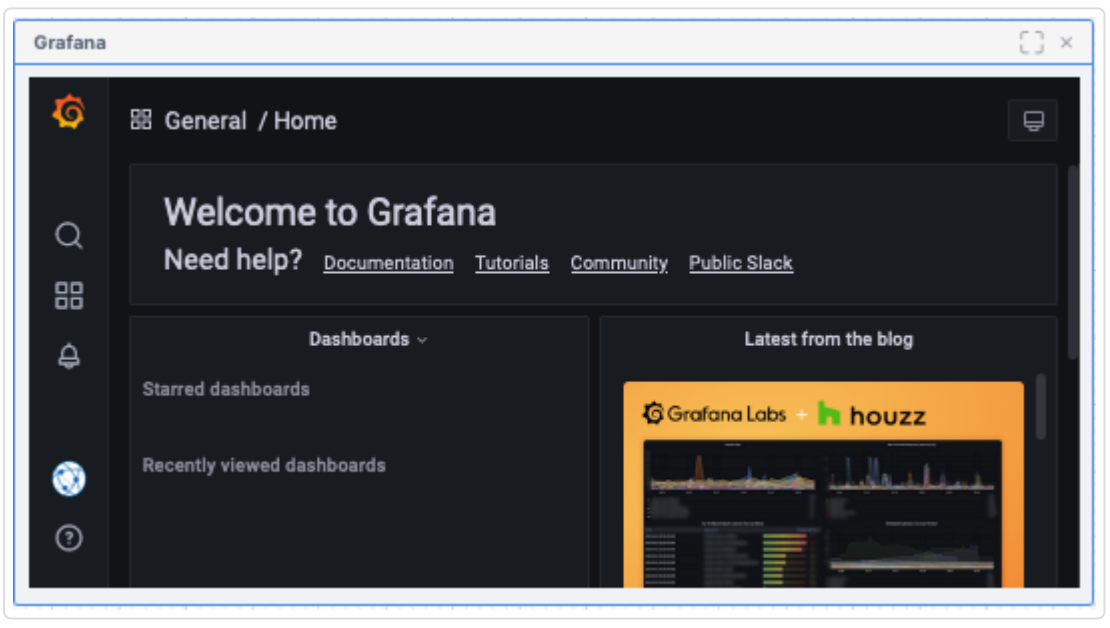
To access Grafana, select the **Administer** menu tab at the top of the Vantiq IDE, and choose **Grafana**:



The Grafana window will open as a pane in the IDE. It might look like this:

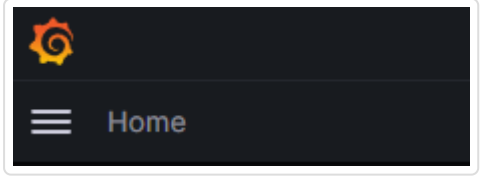


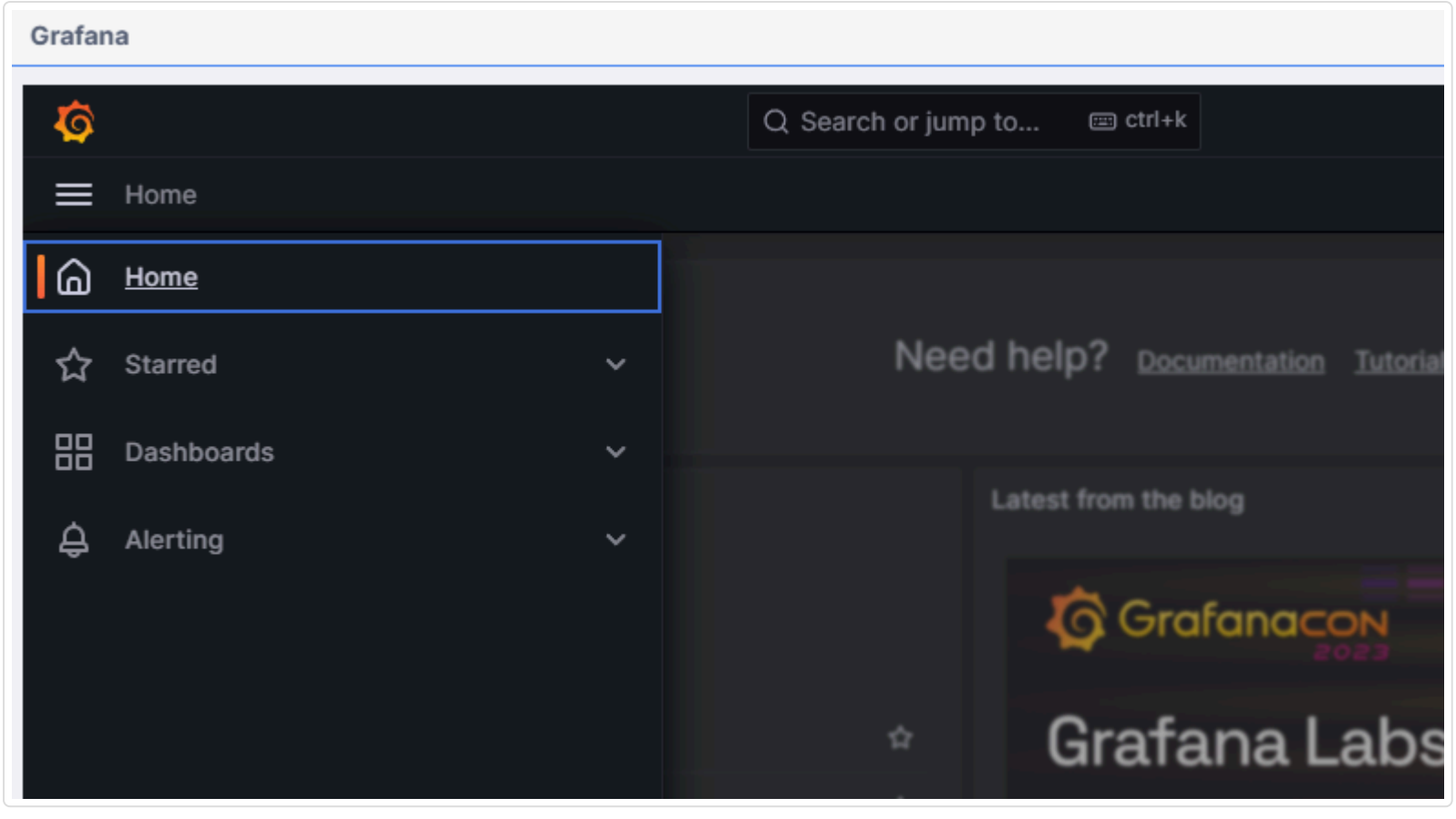
Or it might look slightly different, especially if the pane is resized:



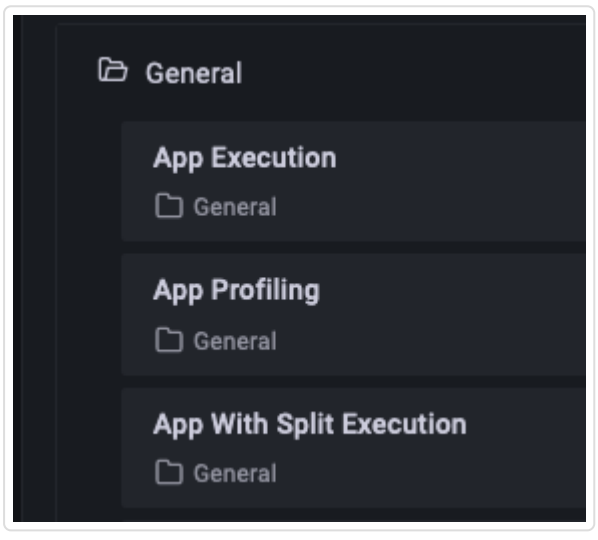
You can also open Grafana in a new browser tab rather than a pane, by using *Ctrl-Click* on the **Administer->Grafana** menu-item (or *Cmd-Click* on a Mac).

The Dashboards available from this Grafana pane are pre-configured with queries to display resource information tailored to the current Vantiq namespace. To see them, click the “three-bars” menu icon next to Home, then click on Dashboards to browse the list of dashboards:





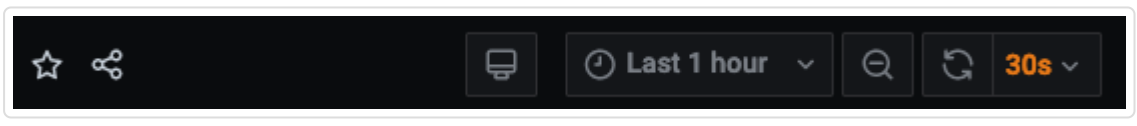
A list of available dashboards appears under the **General** folder icon.



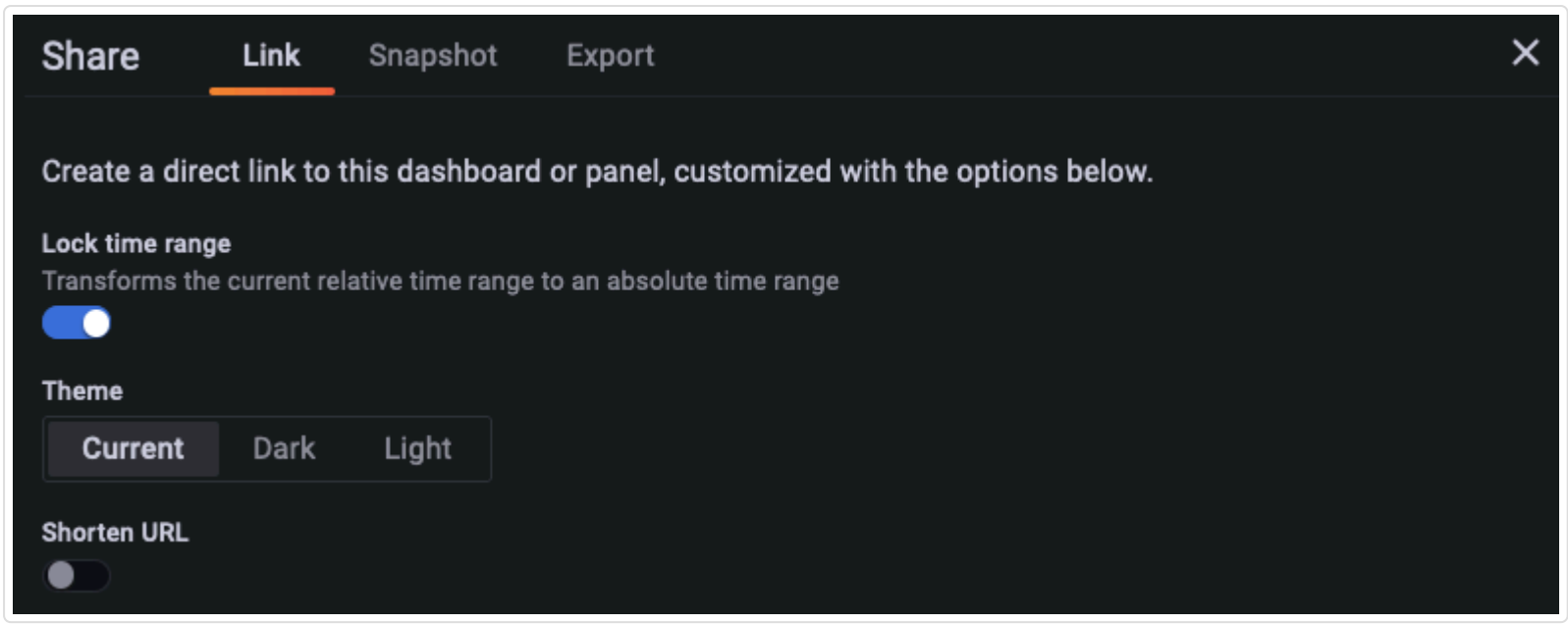
Grafana Monitoring Dashboards for Project Developers

Dashboard Navigation Bar

Clicking on any option in the list of dashboards will bring up a new pane with a GUI display of the relevant area of interest. Most of the dashboards have similar viewing options at the top of the pane:

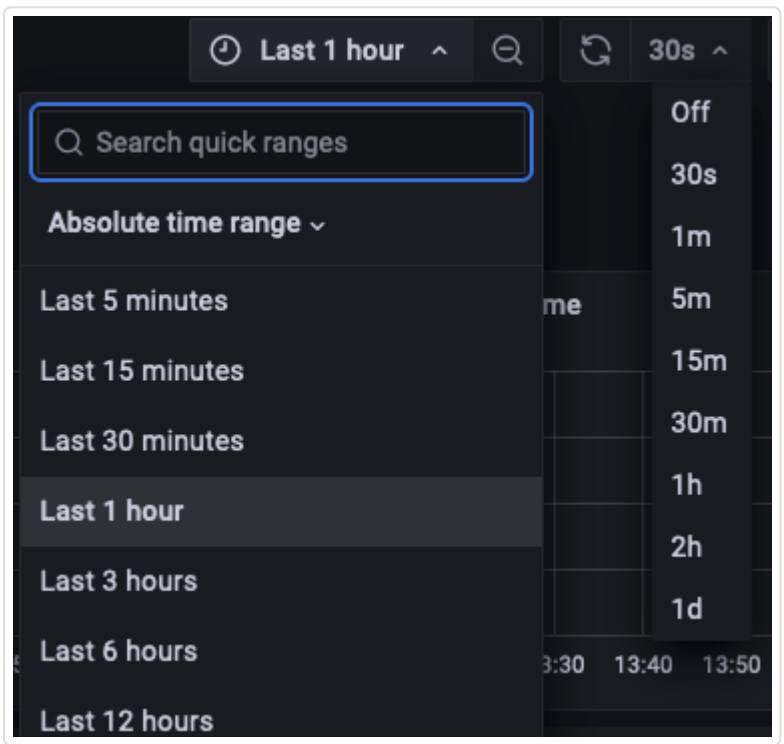


Clicking the star will make this dashboard part of a “Favorites” list. The “Share” icon next to it is very useful for viewing the dashboard in a separate browser window, and giving others access to it via the URL:



Continuing to the right in the navigation bar, the monitor icon toggles between the standard and “kiosk” viewing modes. (Press Esc to return to the original view mode.) Clicking on the clock icon next to it allows the developer to select or customize the time window over which the information will display.

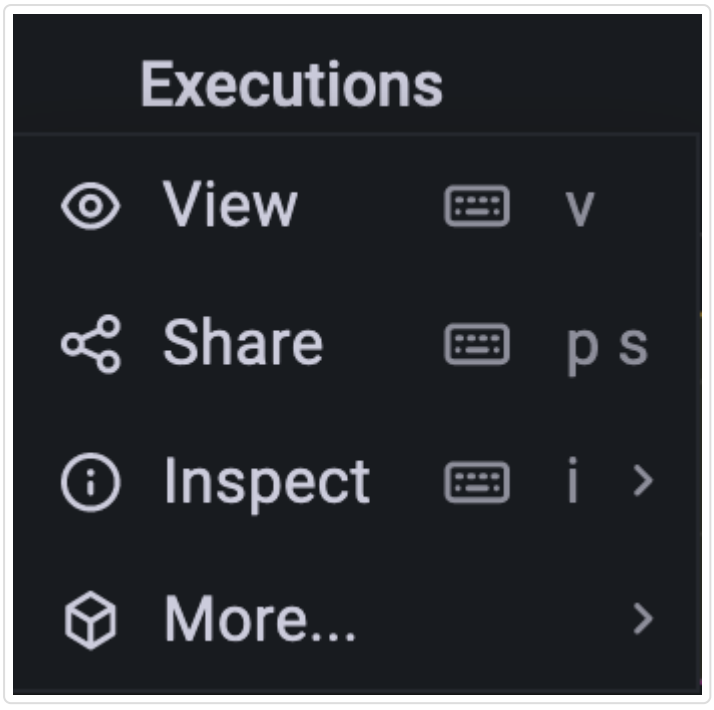
The magnifying glass icon widens the time window of the display, for a “zoom out” effect. On the other side of the “refresh display” icon, the time period drop-down sets refresh frequency.



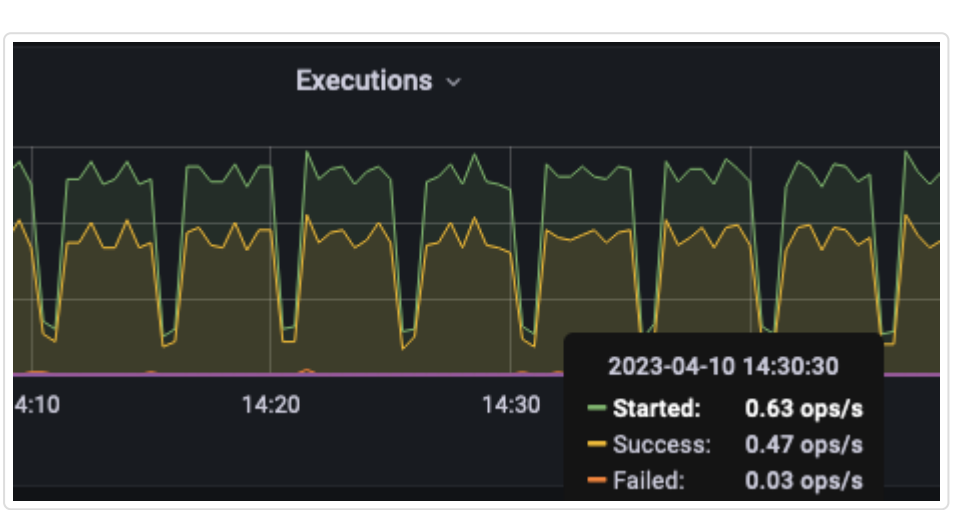
General Dashboard Behavior

Within a chart, dragging the mouse across an area of interest on the graph itself will “zoom in” to that time window. This action will also pause information updates. To resume updates, or select how often they will occur, choose the drop-down menu at the top right of the pane and choose the new timeframe desired.

The title bars for each panel, when clicked, provide a dropdown to expand the view, share the view in another browser window, perform deeper inspection (including downloading it to Excel) or toggle the panel legend on or off.



Hover the mouse over the chart to see specific measurements for that moment in time.

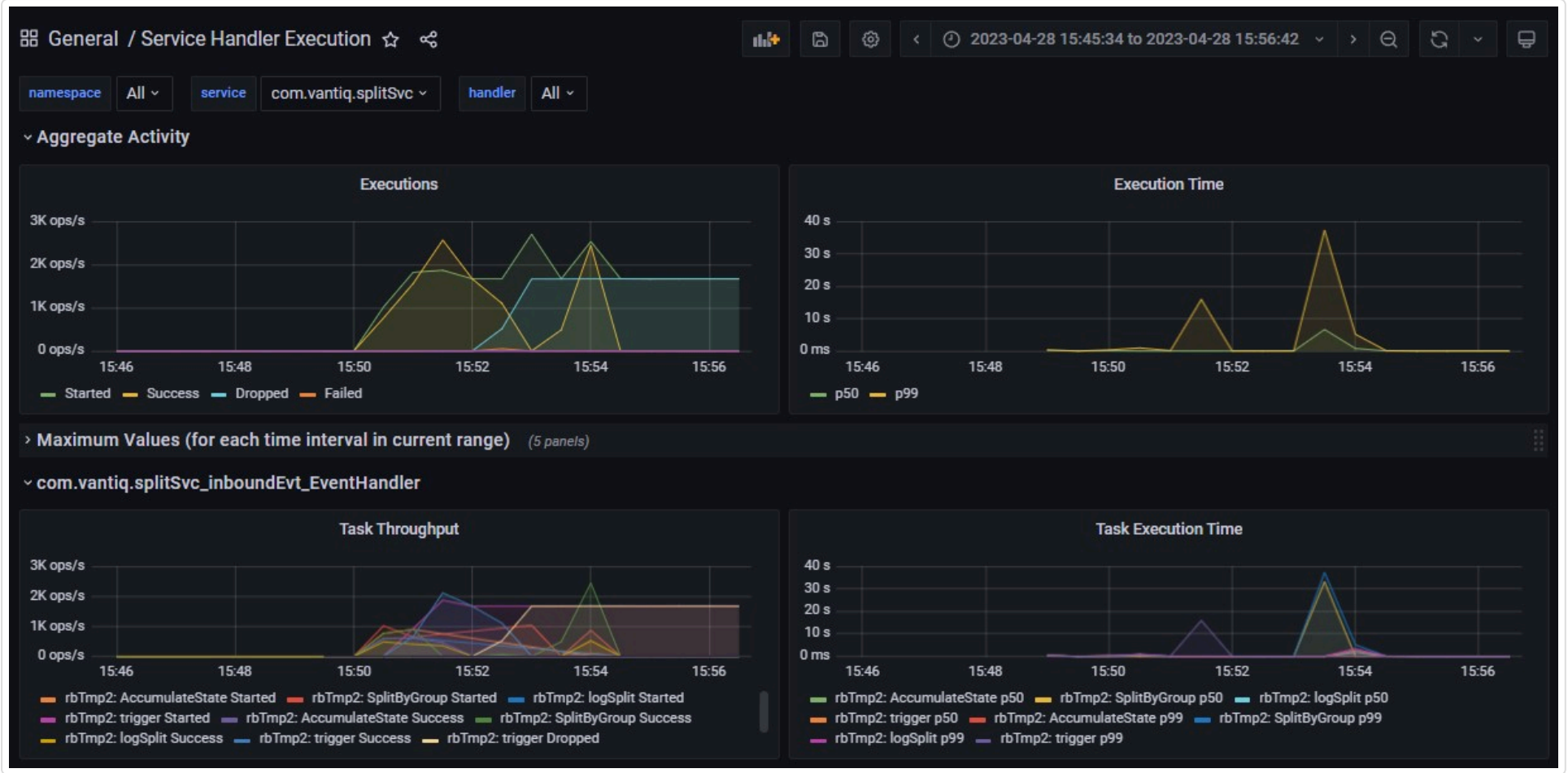


To view just one line of the chart graph, click on its key listing below the chart. (Click the key listing again to restore the full view.) In the example above, the Failed line is barely visible, but by clicking on the label below the display, the chart will rescale to make that measurement more apparent.

Organization Level Behavior

Note that each dashboard is available at the Namespace level and at the Organization level. If you are logged into the Organization namespace and have admin privileges there, you can see metrics across *all* the namespaces in the organization.

The main difference is that the detail-level graphs will include the namespace name along with the detail-names. See example below. Also, there is another drop-down list for `namespace` at the top of the dashboard. Using that, you can restrict the details to a specific namespace, if needed.



Namespace Monitoring Dashboards

The following are Grafana dashboards available for the scope of a single namespace:

- Event Processing - Event flow by source, topic and type throughout the system
- Procedure Execution - Execution times and event throughputs of developer procedures
- Procedure Profiling - Threaded, additional metrics available when “profiling” has been enabled for a given Procedure
- Reliable Event - Details about reliable events and their topics, including redelivery metrics
- Resource Usage - API activity affecting both developer and system types in the system
- Resource Usage Profiling - Threaded, additional metrics available when “profiling” has been enabled for resource usage
- Rule Execution - Execution times and event throughputs of developer rules and rules within developer-created collaborations
- Rule Profiling - Threaded, additional metrics available when “profiling” has been enabled for a given Rule
- Service Execution - Executions times and event throughputs of both developer and system service procedures and service events
- Service Handler Execution - Metrics on event processing through Visual Event Handler apps, including Activity Task granularity
- Source Activity - Throughput and response time of sources used by the system, excluding MOCK sources
- Storage Manager - Throughput details for serviceConnector Storage Managers
- Type Storage - Database usage by developer-created standard types

Most Commonly Used Dashboards

By using the Grafana dashboards available, developers will gain insights into the most minute details of their application function. More commonly, however, only a few of these panels provide the most benefit in locating the largest performance bottlenecks:

- Service Handler Execution - Most applications employ Service Visual Event Handlers. The Task Throughput graph displays the Activity Tasks for the handler and Task Execution shows their latencies.
- Service Execution - Displays response time latencies for procedures defined in a Service’s API
- Event Processing - Provides confirmation that event rates are as expected, ensure that there are no events “dropping.”
- Resource Usage - This is where developers can discover unexpected database activity by watching what happens to every part of the running application, including system resources. See also Resource Usage Profiling.
- Source Activity - Displays Source event throughput latencies.

Service Execution Dashboard

Services encapsulate behavior associated with a specific functional aspect of an application. The Service interface consists of a list of procedures and Event Types. Services also manage their own State.

The Dashboard for Service Execution provides metrics for service procedures and service event types.

Note that Visual Event Handlers will show up on the Service Handler Dashboard and VAIL Event Handlers will show up on the Rule Execution Dashboard.

In this view of the Service Execution dashboard for a namespace, two developer-created Services, PDC and Tests, are visible.

If your application makes calls to system-service procedures, those will also show up on this dashboard. Vantiq includes them so they can be expanded for more in-depth analysis of how they are running in the system. System procedures used in the generation of an app can also show up in this list.

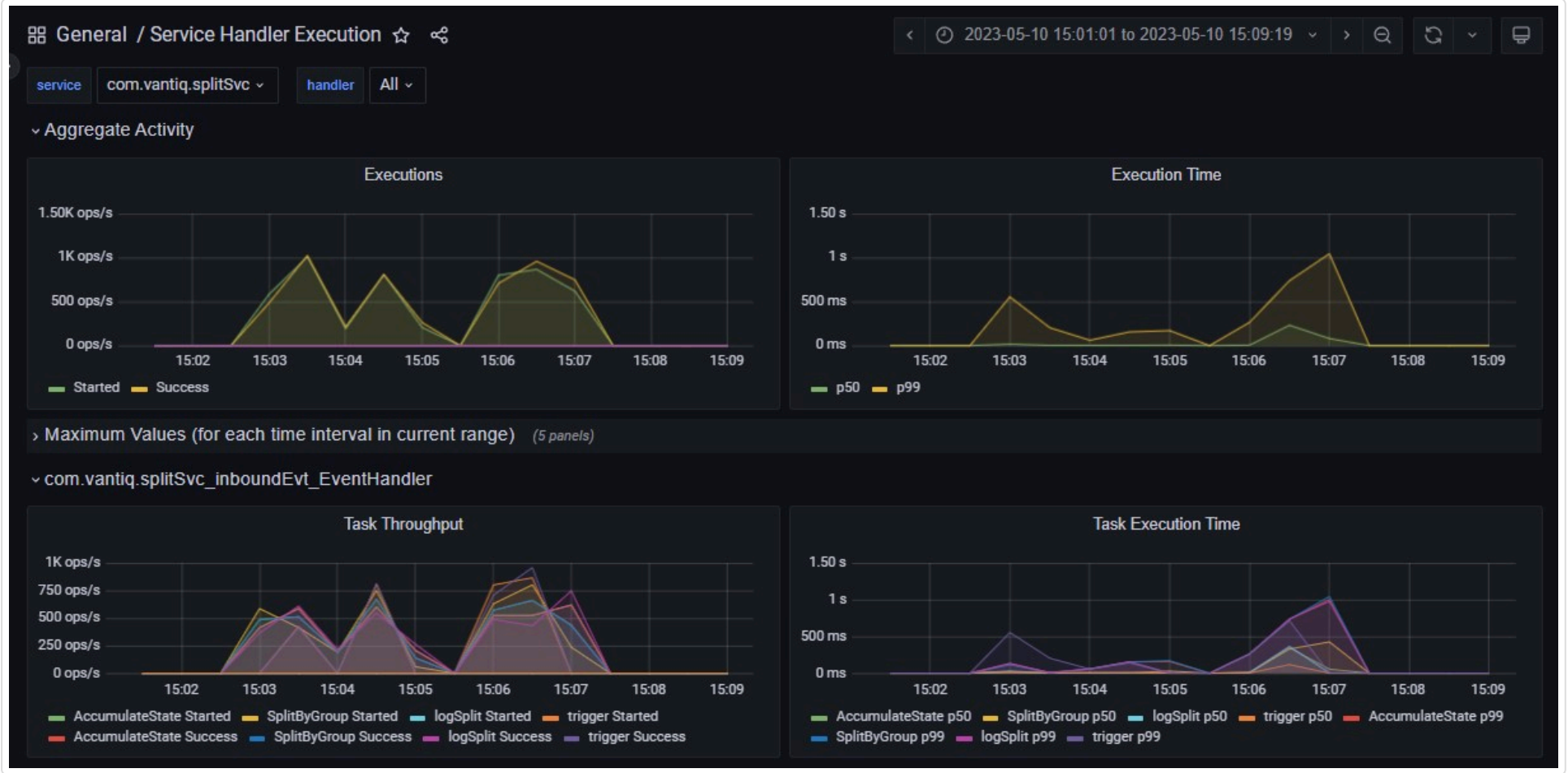
Let’s say we’re curious about which procedures were executed from a particular service.

The chart below is for the *com.vantiq.splitSvc* service. We see throughput entries and execution times for the procedures in the service that were executed by the running application within the last ten minutes. We also see throughput for any Service Events that occurred in the same time frame.

Note: The initializer procedures for services are not included in the metrics for the Service Execution dashboard.

Service Handler Dashboard

The Service Handler Execution Dashboard visualizes event activity in Visual Event Handlers within the Namespace, expressed in operations per second.



Here are some of the critical Event Handler behaviors to observe in this dashboard:

- Executions must match the volume of work expected for a given workload
- Ensure that execution times remain stable as loads change
- Observe Event Handlers individually to detect potential bottlenecks

As a general rule of thumb, for high-volume systems, response times should remain under 1000 ms. Also, if execution times grow disproportionally higher as the load increases, it is a scalability problem. Using this dashboard, developers can identify the individual Event Handlers and the Activity Tasks within them that are taking the longest to process.

Two panels for **Aggregate Activity** display the sum total of executions for all Visual Event Handlers; the first expressed in operations per second, and the second as a function of the total execution time. At the bottom, the green line **p50** is the line for median execution time, and **p99** is the time overall. The aggregate information is useful to verify that there are no problems, but if there is, looking at Event Handlers individually will pinpoint the one(s) where there is an issue.

Below the Aggregate Activity display panel, click the symbol next to **Maximum Values** to reveal the highest throughput rate levels, expressed in operations per second, of each of the Activity Tasks listed for each of the Event Handlers running in the namespace.

The Event Handler-specific charts below the Aggregate information display a graph of activity task throughput, with each line representing starting and success rates of each. Here, we've achieved a "close-up" of part of this graph by clicking on the chart and dragging across the region of interest.

Profiling Dashboards

Grafana offers Profiling Dashboards for the following resources:

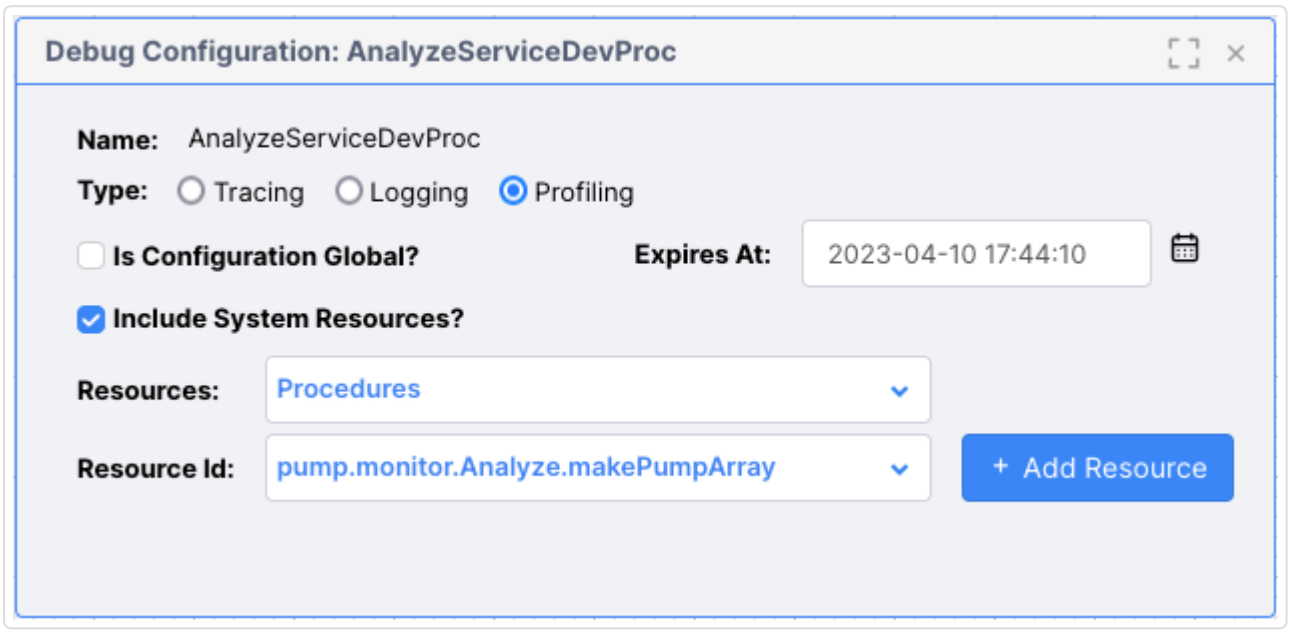
- App
- Procedure
- Rule
- Resource Usage, as defined by REST/HTTP calls

By default, Profiling is not enabled for any of these resources, so no data will appear in the dashboards. By enabling profiling, the developer directs the Vantiq server to collect additional statistics that aren't normally available, and these are threaded so the data shown is solely related to the "root" profiled resource.

Enabling Resource Usage profiling requires REST commands with the URI parameter *vq_enableProfiling* set to *true*. Tracing takes place from the REST layer down.

Enabling Profiling for Procedures, Rules, Apps

To enable profiling for Procedure, Service or App, choose **Debug Configurations** from **Test -> Advanced** at the top of the IDE, and then click the New button in the Debug Configurations window. The resulting pane allows the developer to choose a resource, with or without related System Resources, to be profiled for a specified length of time.

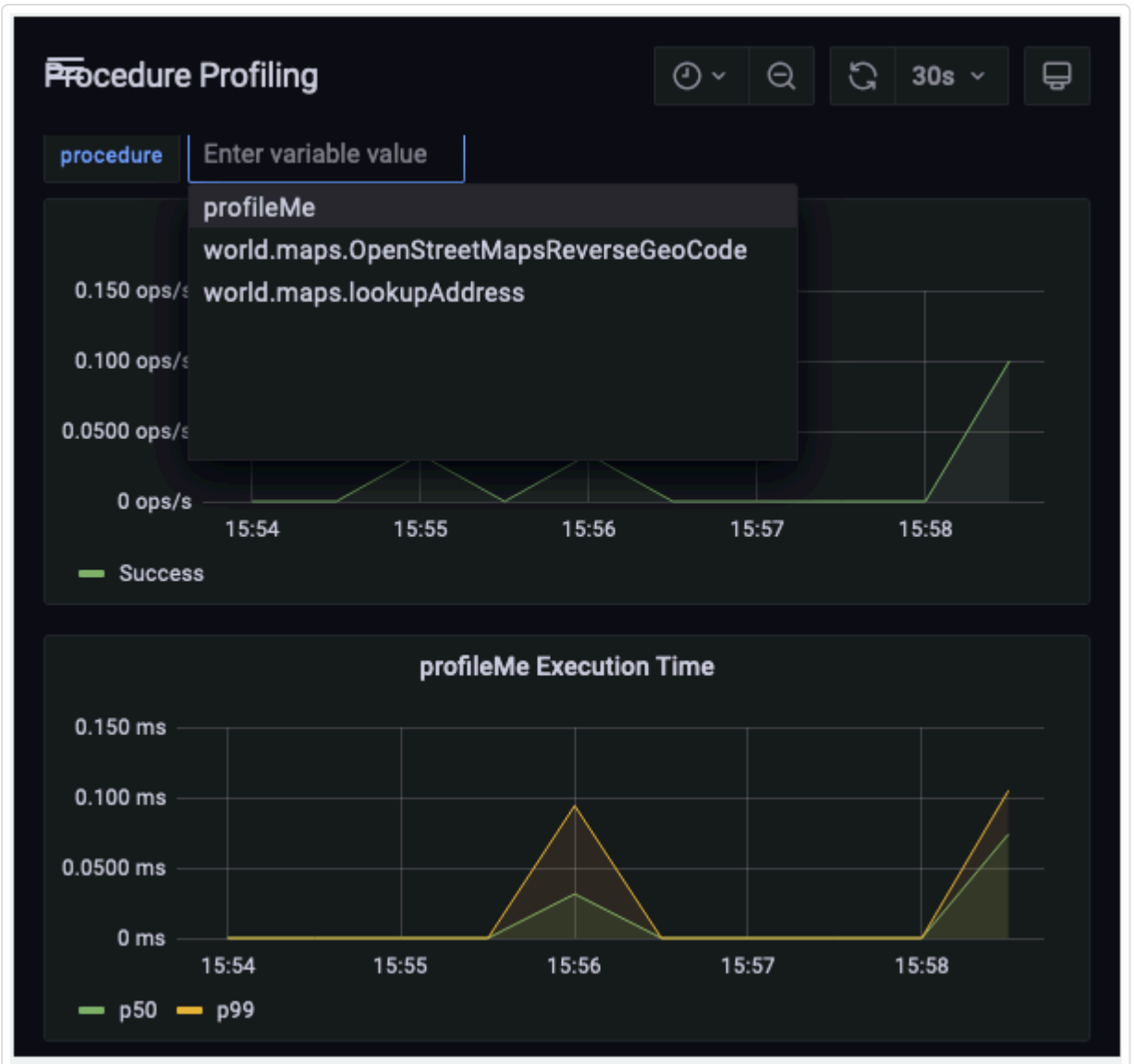


If **Is Configuration Global?** is checked, then all applicable resources will have profiling enabled for them. Otherwise, developers choose first which type of resource to use in profiling, and then click the **+ Add Resource** to confirm the selection.

By default, Profiling will only remain enabled for two hours; the developer can modify this in the **Expires At:** text field.

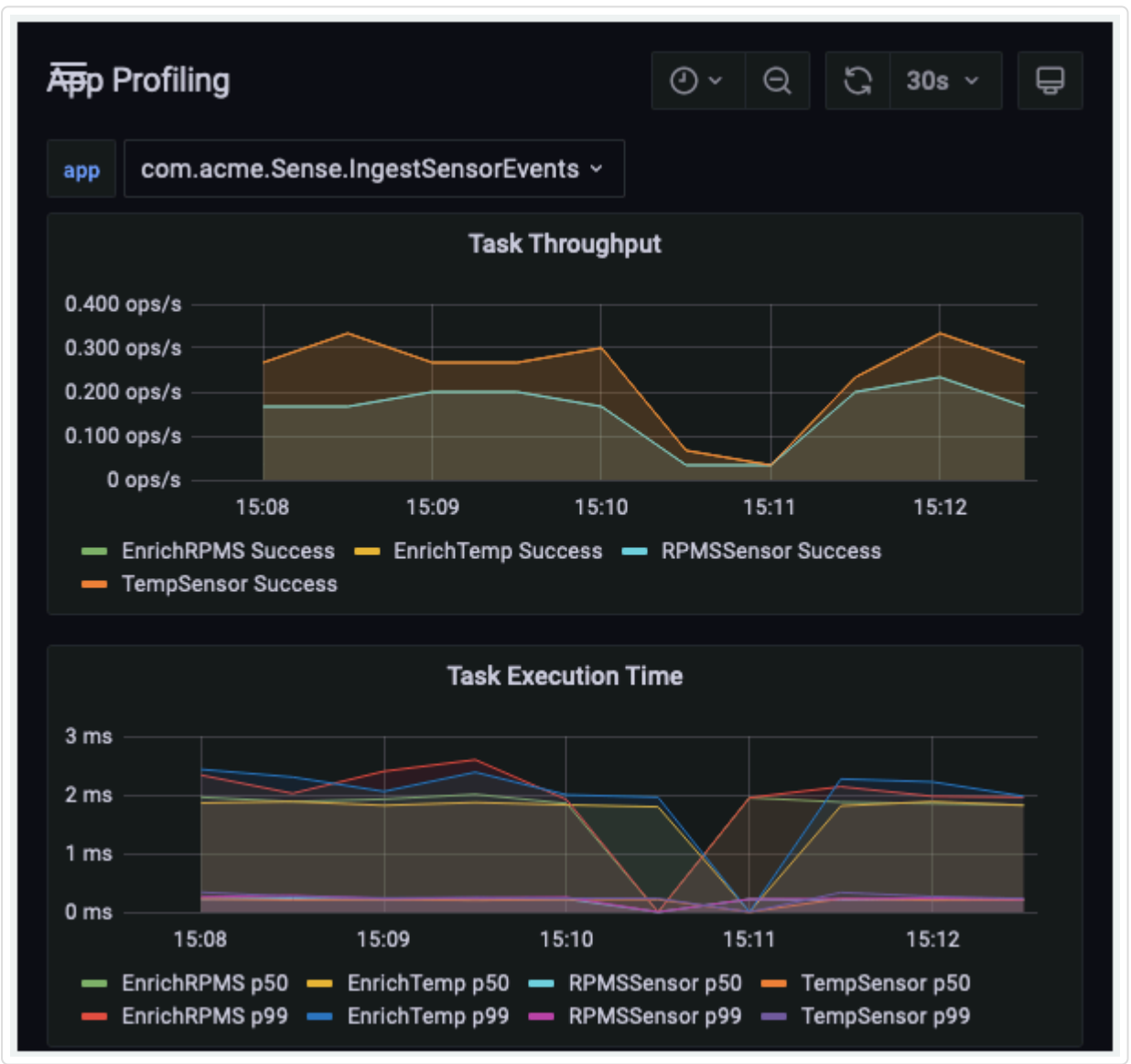
Note: The **Administer -> Advanced -> Profiles** menu option is for configuring *user* profiles, not specifying debug configurations, including this resource profiling.

For resources where several of the same type have profiling turned on, such as when a user checked the **Is Configuration Global?** option in the Debug Configuration pane, a drop-down menu provides a list from which to choose. In the example below, the Grafana Procedure Profiling Dashboard is open, but because there are multiple procedures all being profiled, the developer chooses which to analyze from the drop-down.



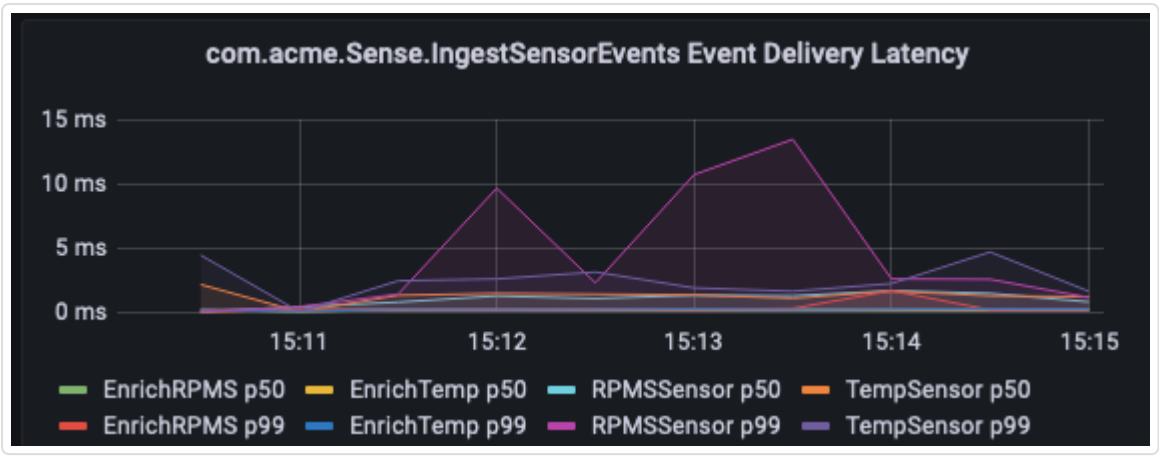
Note: Only procedures not in packages will show data in this dashboard.

The profiled resources share the same types of panels. The top two are ** Task Throughput **and ** Task Execution Time** graphs, which show much the same data as in the resource Execution Dashboards, such as in this example from an App Profiling dashboard.

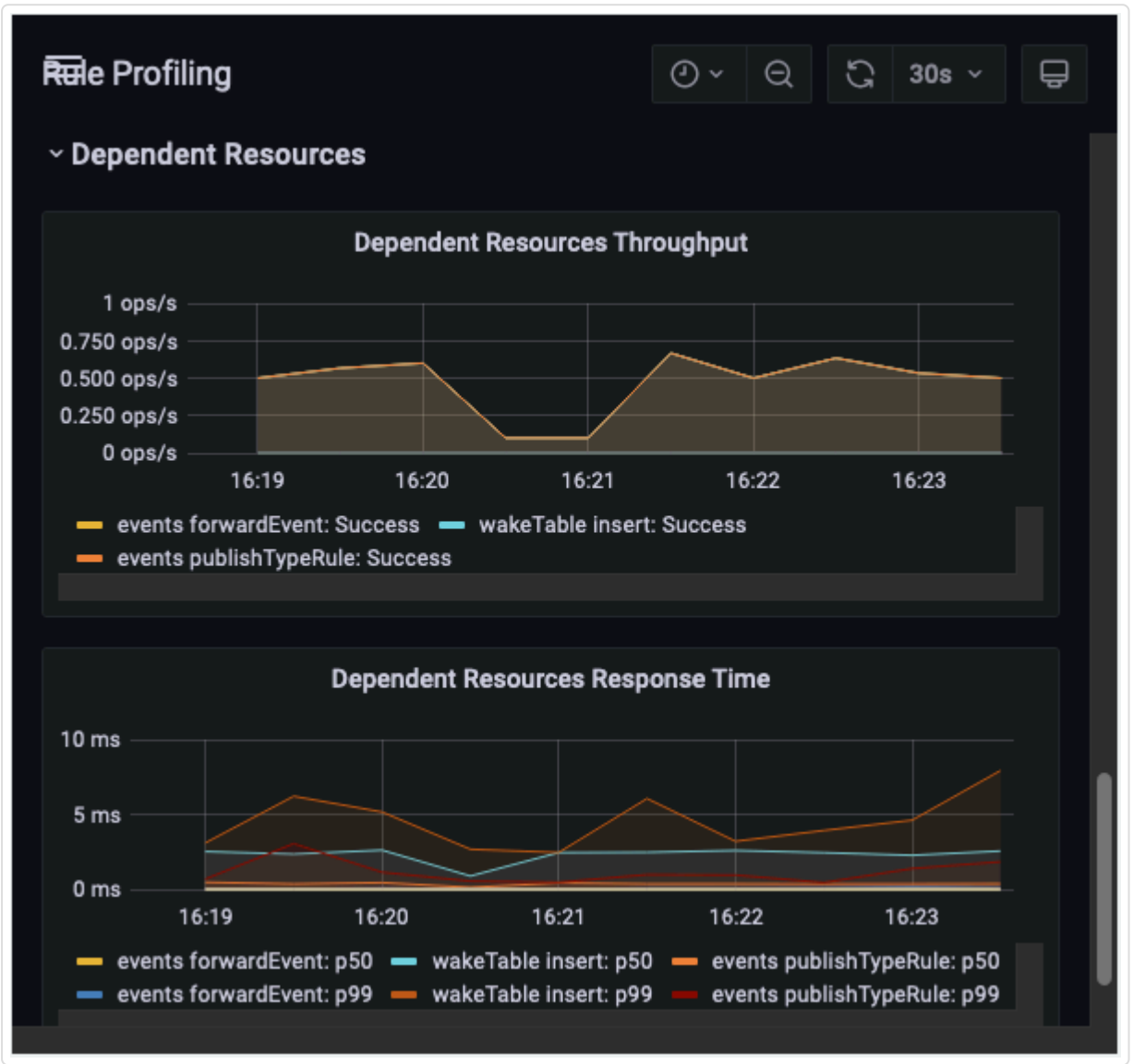


Below these are panels with more extensive information from the enabled profiling.

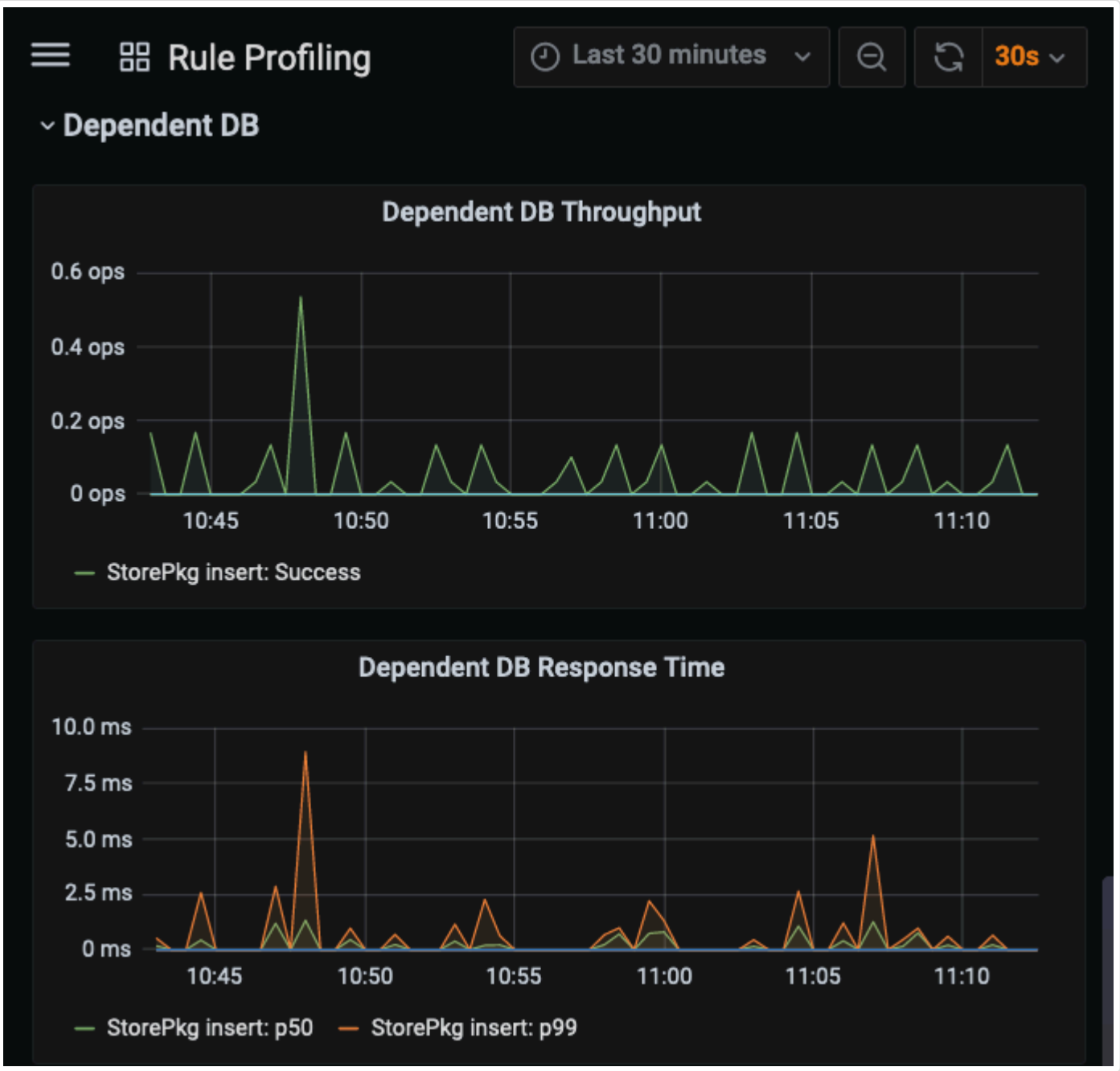
The **Event Delivery Latency** graph helps developers pinpoint time delays between event processes.



The panels below this show throughputs and response times for dependent resources, first for any procedures processing, and then for other resources, as seen in this example, where a triggered rule results in an Insert operation on a persistent standard type.



The final couple of panels are for the **DependentDB** involved, in this case, because of the Rule.



Event Processing Dashboard

Events are at the heart of Vantiq platform applications, so it's not surprising that Event Processing Dashboard is replete with event-related information touching all parts of the system.

Aggregate Activity shows all event arrivals into the system by kind. From this panel, a developer can learn a lot about incoming messages, including:

- The average arrival load - does it meet expected rates?
- Commonality and severity of event "bursts"
- Event droppage

Dropped events are lost events. Developers must never ignore them, but drops associated with diagnostic data are not as concerning. These would include events associated with rule or procedure snapshots, audit records, profiling events; all of which will not occur in the live application with diagnostic and debugging turned off.

Event drops associated with system operation are much more critical because it means that the application is not receiving and processing expected events. Generally, these happen because the rate of event consumption is slower than event generation. By inspecting the Event Processing Dashboard, the developer can determine if:

- Events are generated at too high a rate for the system
- Events are processed by the system at too slow a rate - this would appear as latencies that grow exponentially when the event rate grows

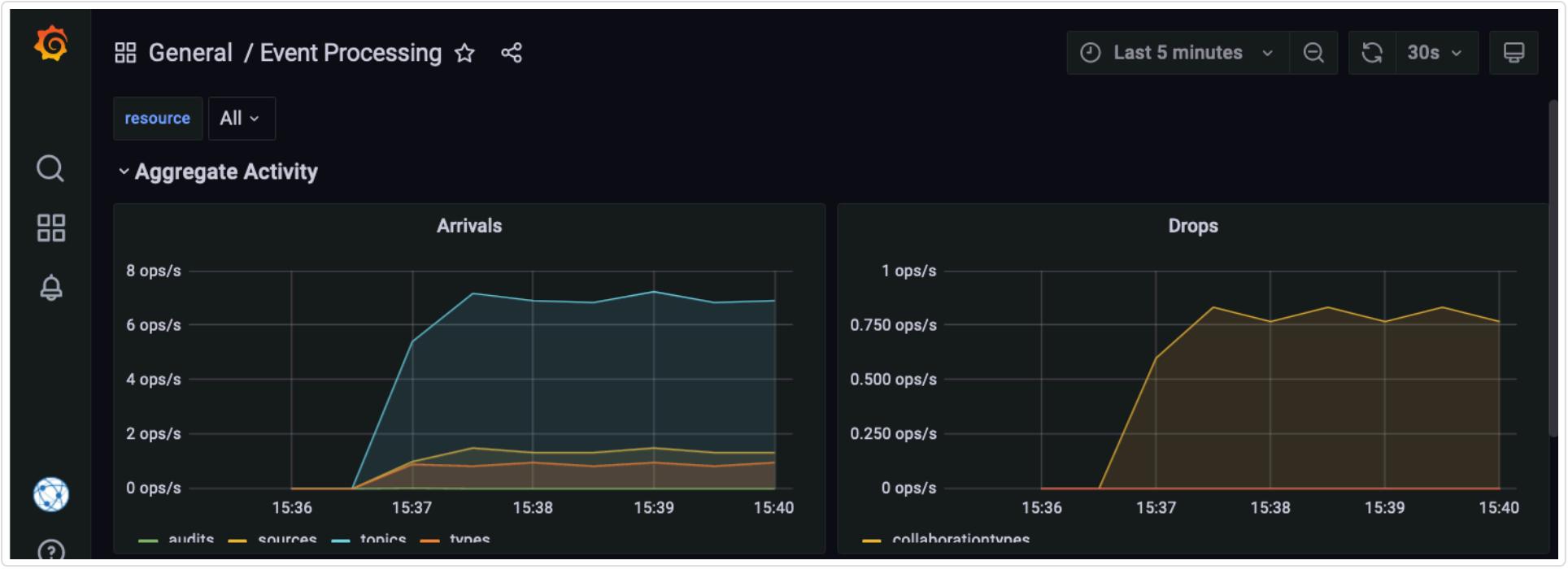
Solutions include:

- Slowing event generation
- Making the event handler more efficient
- Sending some processing to be done asynchronously by another event handler
- Avoiding database interactions where possible, as this is always a synchronous operation; using service state is much more performant

The Vantiq **Workload Management** Document details causes and effects of quota violations that result in dropped events.

In the screenshot below, none of the events coming from audits, sources, topics or types are creating a high enough load to force the system to drop events, and yet there are drops shown by the line with the legend "collaborationtypes." Further investigation reveals that there is a compilation error in a service procedure, which is preventing events from being processed.

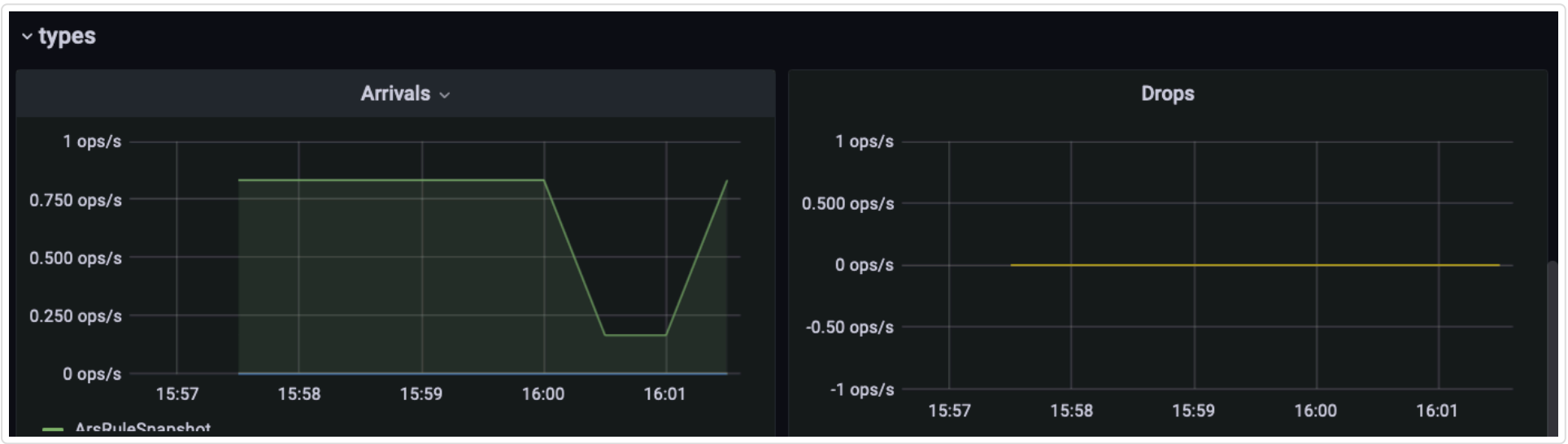
Developers considering system scalability will want to pay close attention to this dashboard. Heavy incoming event loads or frequent high event bursts might benefit from distributing the system to increase parallel processing.



Developers can expand the relevant sections to see more detail about events originating from:

- audits - occurs when an audit record for a non-type resource is generated
- collaborationtypes - these are Apps, aka Service Visual Event Handlers
- sources
- topics
- types - both user-defined and system

Looking again at the Aggregate Activity Event Processing pane above, we see the orange line for “types” in the arrivals graph, but the project isn’t interacting with database data, so what is that? Scrolling down and opening the pane for “types” reveals that the type causing these events is “ArsRuleSnapshot,” which is a system type, specifically one that persists error information, in this case caused by the procedure compilation error from before.



Procedure and Rule Execution Dashboards

The Dashboards for Rule Execution and Procedure Execution provide metrics for rules and procedures respectively, including system rules and developer-written procedures, but not service procedures. For the most part, developers should avoid creating rules in favor of handling events with Apps, aka Visual Event Handlers (from Services), as the activity tasks are already designed to be performant.

Procedures are generally executed synchronously; developers should be looking in this dashboard to ensure that procedure execution latencies are low.

If procedure latencies are unacceptably high, consider:

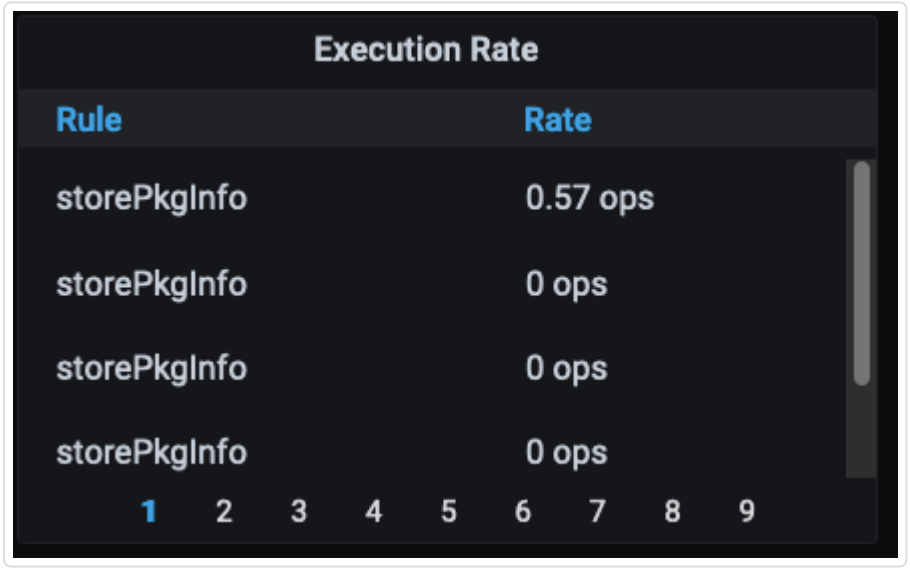
- Invoking the procedure less often
- Determining if the procedure can be implemented more efficiently
- Moving some of the procedure’s processing to another asynchronous event handler

At the top of the dashboard, the Aggregate Activity is displayed, by executions per second in the first panel and execution times in the other. The **p50** line shows the median execution time, and the **p99** is the overall execution time.

The **Maximum Values** keep statistics on the highest measurements encountered by the procedure or rule in the timeframe specified, for

- Start Rate
- Execution Rate
- Execution Time
- Dropped Event Rate
- Failure Rate

Here is an example of the maximum executions rates kept for intervals of 30 seconds. Past history is clickable from the page numbers below.



Below the Maximum Values panels are the variable list of application procedures or rules present in the namespace. Expand the panels for any of these to see data for that rule or procedure individually.

Note: Service procedures are not visible in these panels, but are visible in the Service Execution Dashboard.

Reliable Event Dashboard

The Reliable Event Dashboard displays data about the behavior of reliable events in the namespace. Metrics include number of events Persisted, Acknowledged, and Redelivered.

Resource Usage Dashboard

The Resource Usage Dashboard displays system REST, VAIL, and WS (websocket) operations between resources on the Vantiq platform for a running application. Generally, developers will be interested in this dashboard with respect to database interactions with resources:

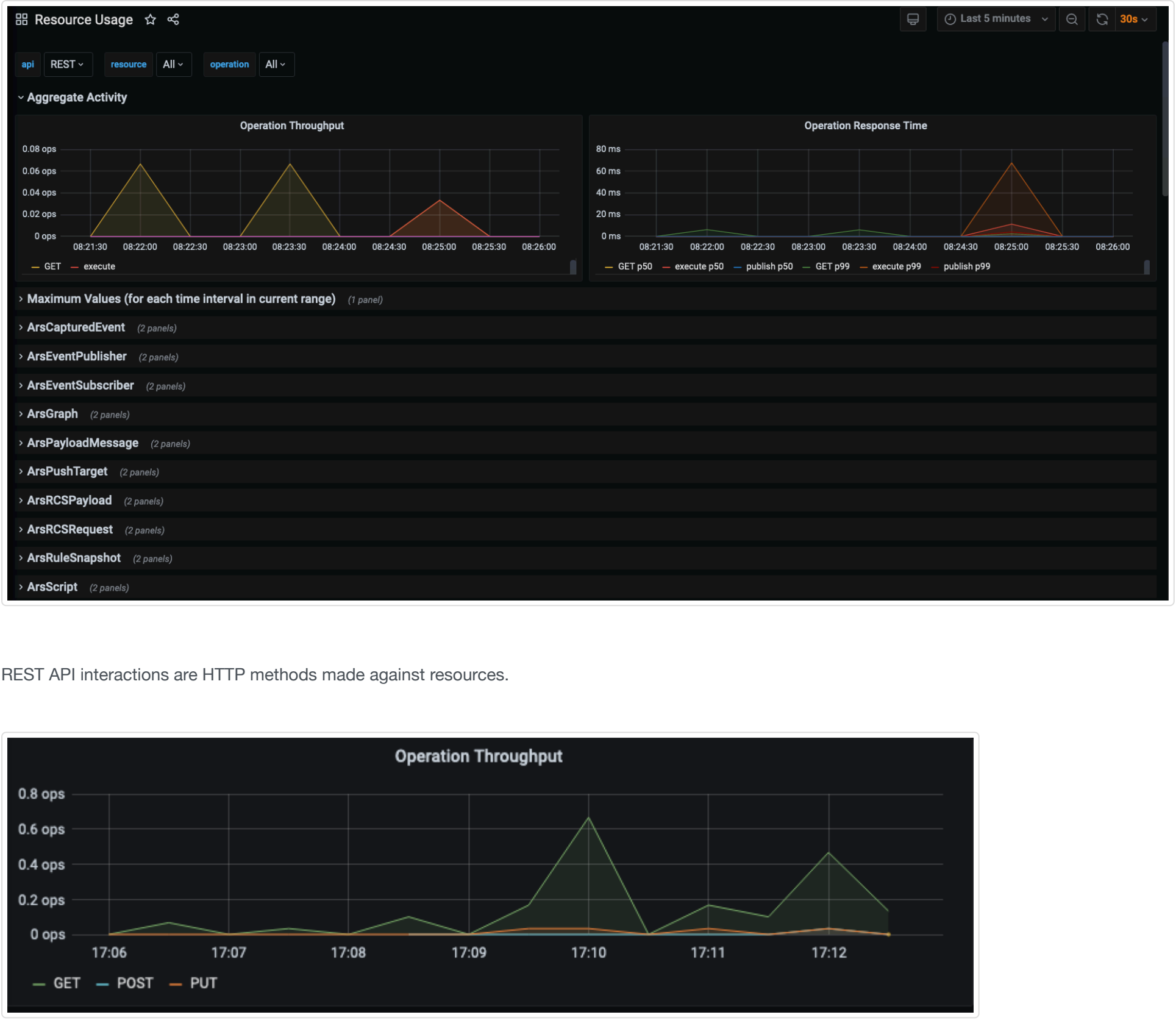
- How many database requests come from the system as a whole, both from the running application and the Vantiq system infrastructure
- The volume of data requested

This is because databases have limited throughput, and therefore are the most frequent cause of performance issues. Developers should work to reduce database requests as much as possible.

Developers can choose which API view they want to see, depending on what is most relevant to the application. Options available to the developer vary by what the project contains, but will include at least:

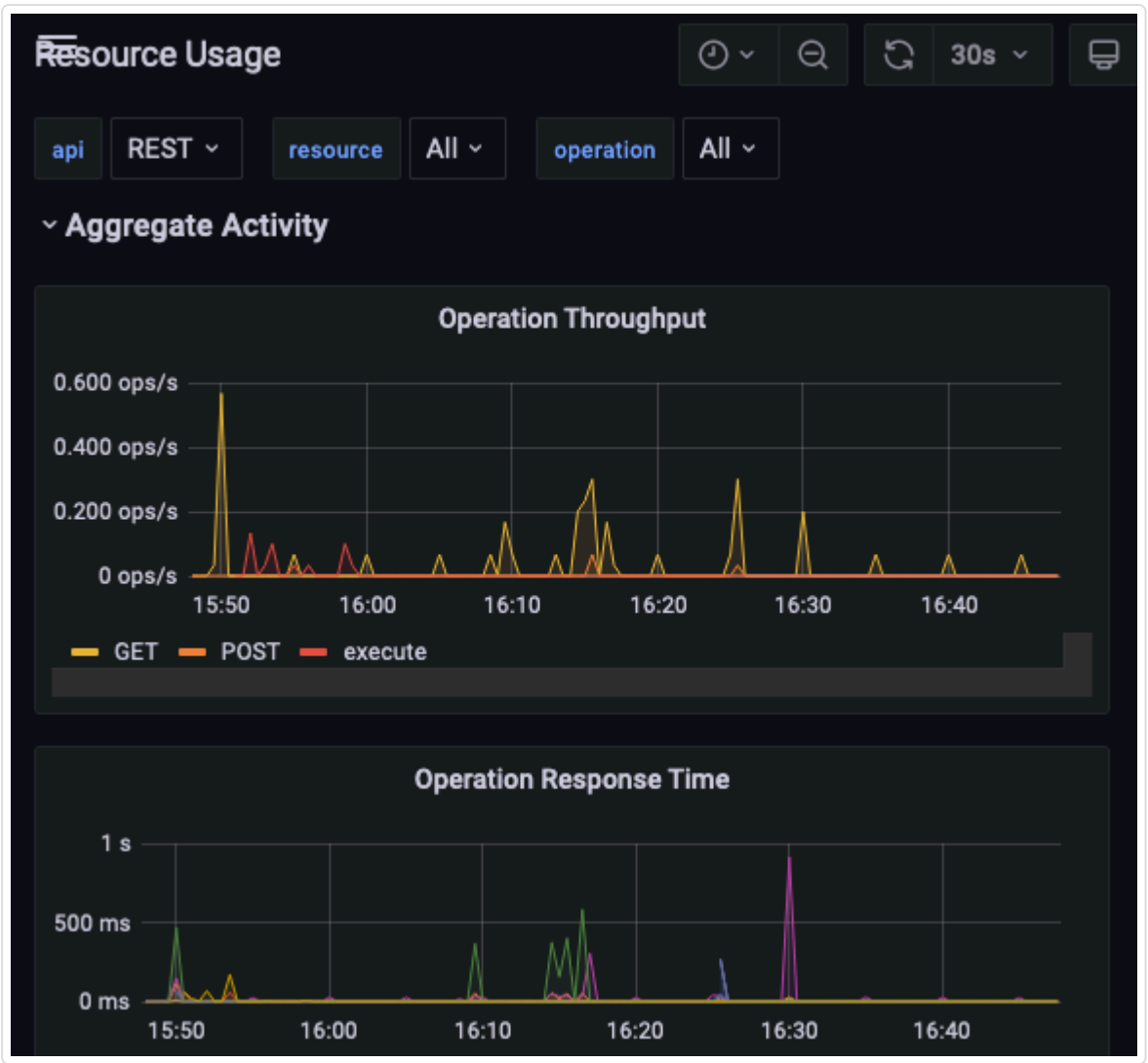
- REST - HTTPS REST methods
- VAIL - VAIL interactions with resources
- WS - Websocket

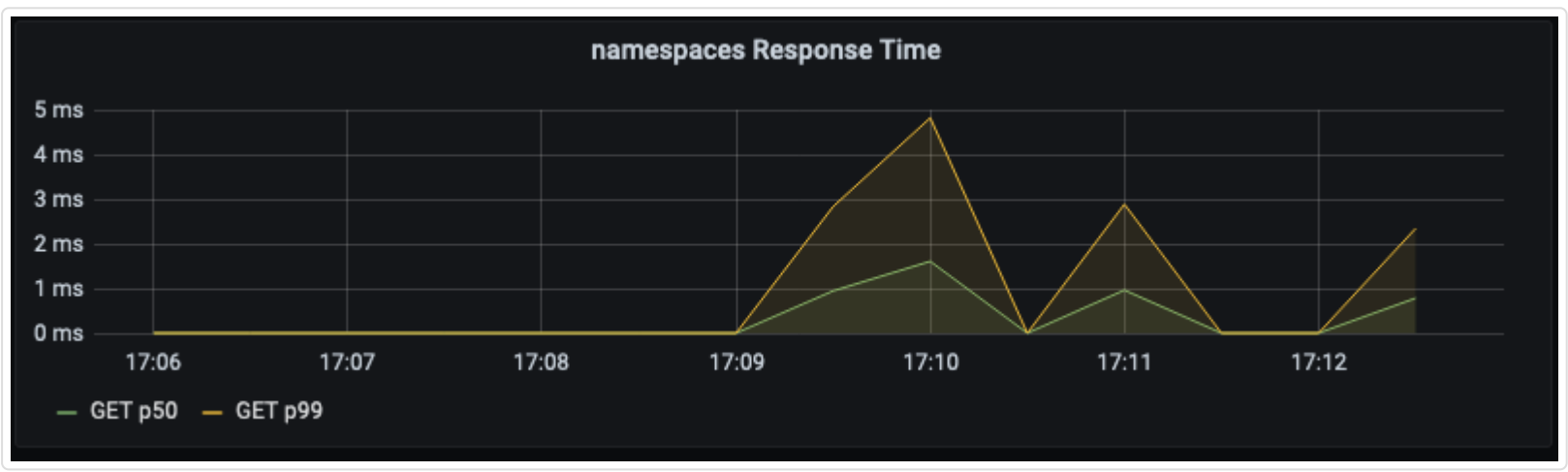
Aggregate panels show the throughput and response time for all resources in the namespace. Below these panels are expandable panels for individual resources.



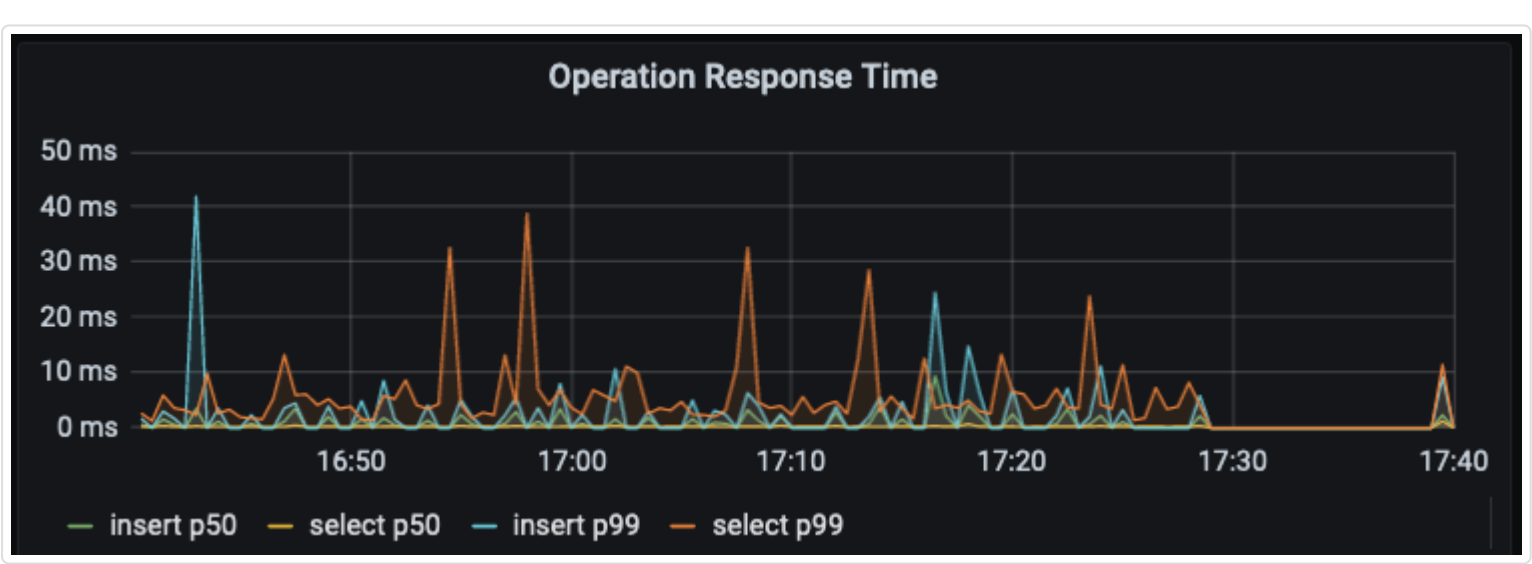
REST API interactions are HTTP methods made against resources.

The affected resources can be expanded to see specifically the throughput and response times for REST calls used on them.

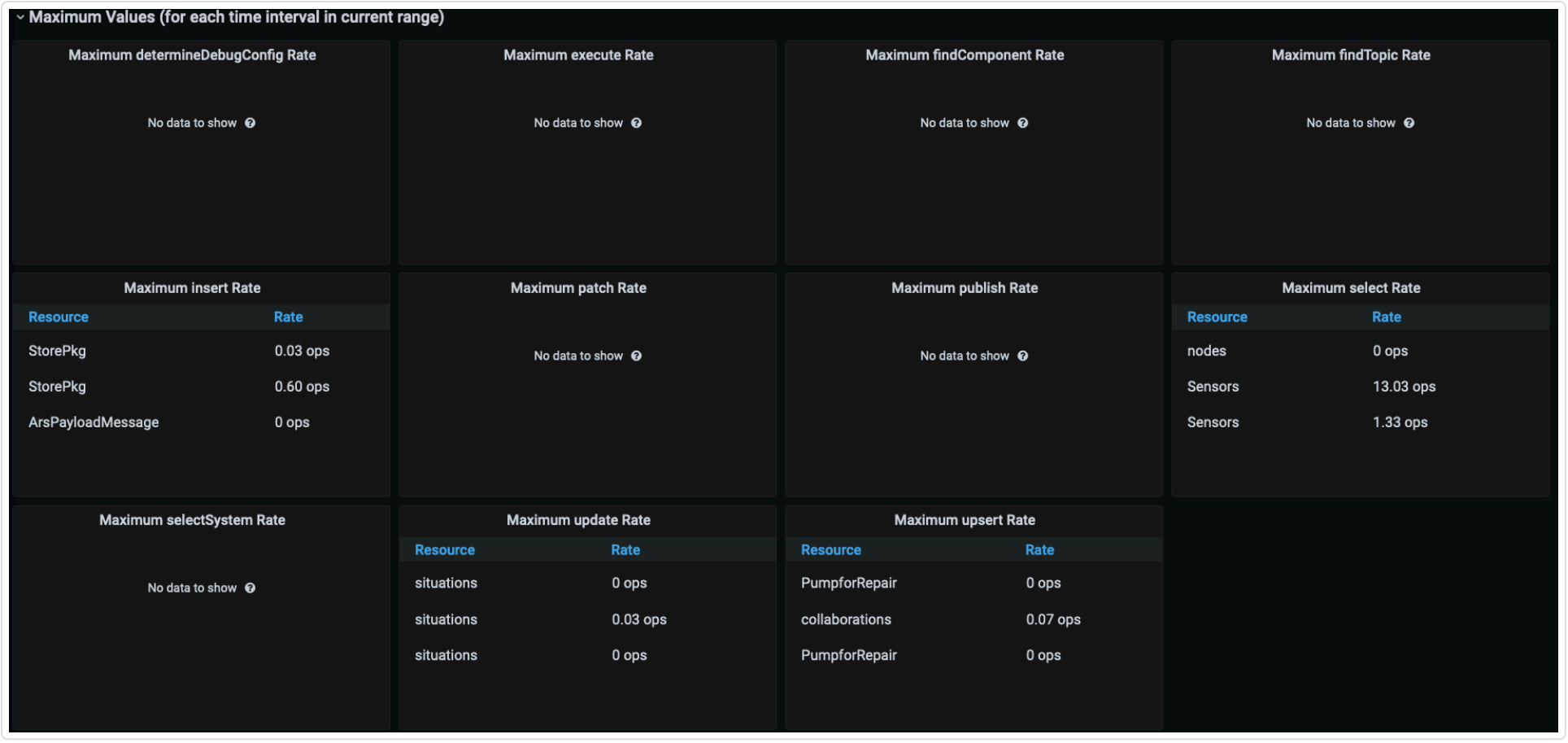




The VAIL resource usage maps to SQL-like DML calls (insert, select, update, upsert). Here is an aggregate VAIL view showing the VAIL commands **insert** and **select**.

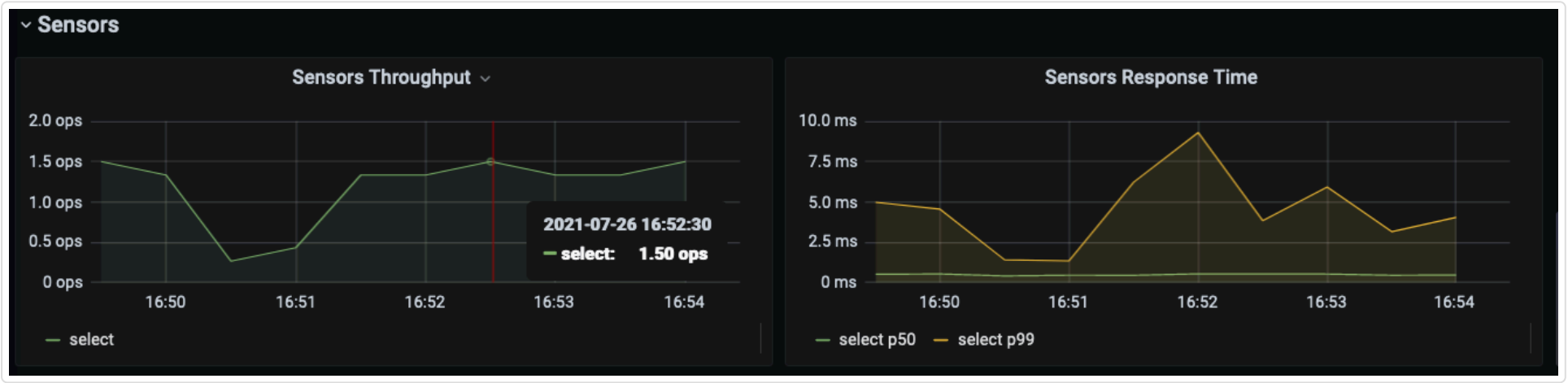


Below the aggregate panels is the Maximum Values option. Expand it to see all the highest quantities of operations experienced by the system for each time interval in the current timeframe.

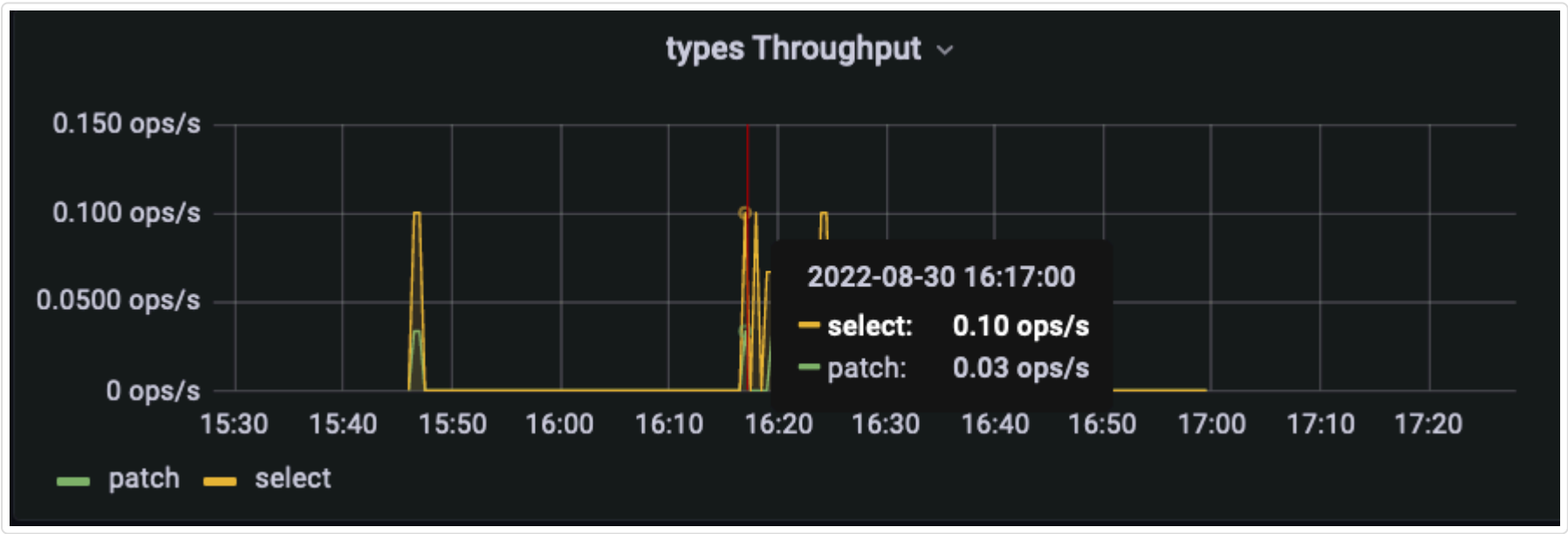


After the Maximum Values, there appears a list of all the components that make up the whole of the namespace activity in particular. Developers can delve deeper into the parts by expanding their corresponding panels.

Here we see a VAIL view of the numbers and speed of select statements to the Sensors datatype in an application, expressed in operations per second.



The Resource Usage Grafana view is helpful for developers who want to see if there are any unexpected database interactions taking place in the system. One of the panes available to show greater detail is “types,” which details what database interactions are taking place:



In this example, we see that there are very few database interactions, and most of those are reads. Since the App in this case enriches sensor input with database information, this is expected behavior.

Source Activity Dashboard

The Source Activity Dashboard monitors sources from standard industry messaging protocols and push notifications.

Note: MOCK sources, SMS and REMOTE sources, and artificial events generated by the **Test.sendMockSourceEvent()** or **createSourceEvent()** VAIL procedures won't display here.

Developers using this dashboard are usually looking for dropped events from sources, and their cause(s). Dropped events are lost events, and this is a problem. Either too many events are coming into the system at once, or the processing latencies are too long. Even before the system starts to drop events, processing latencies longer than the rate on event input are an early warning that eventually, the message queue will be exhausted and drops will occur. Consult the Vantiq Workload Managment document for more details about quotas, rates and credit limits.

By inspecting the Source Activity Dashboard, the developer can determine if:

- Events are generated at too high a rate for the system
- Events are processed by the system at too slow a rate - this would appear as latencies that grow exponentially when the event rate grows

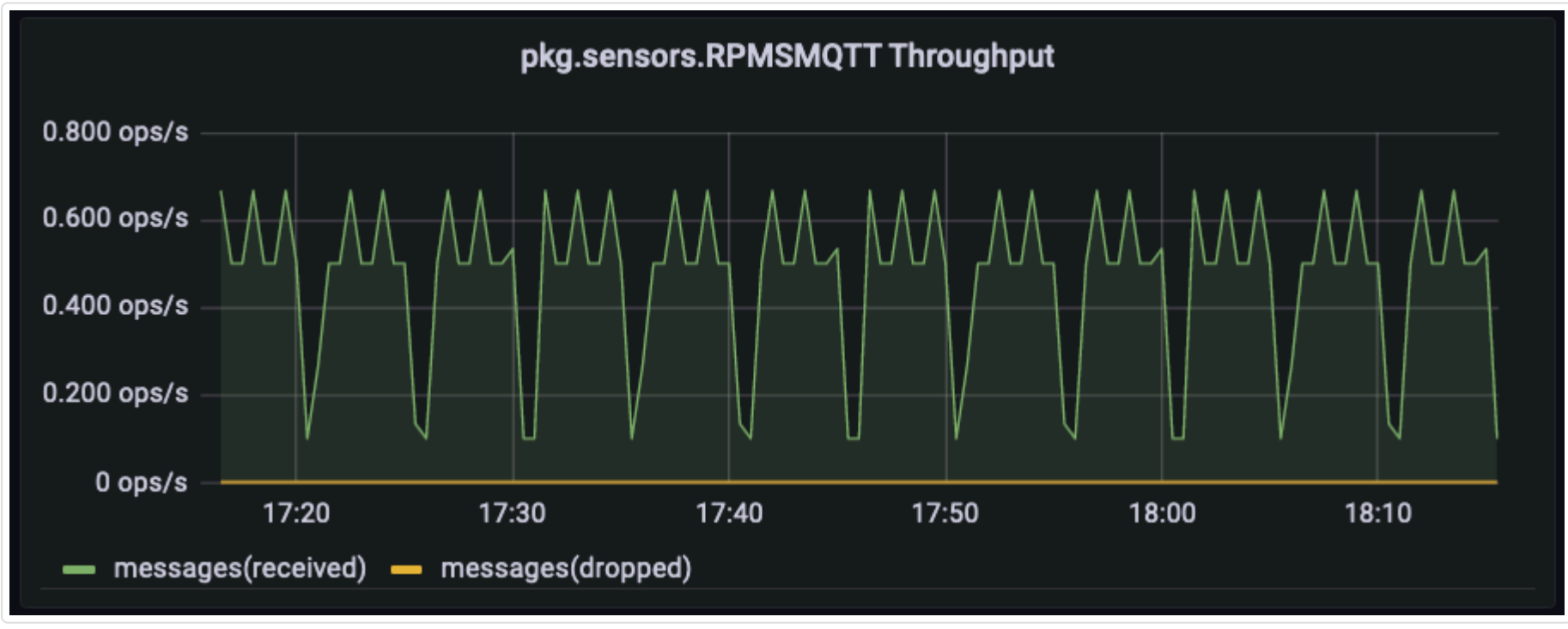
Solutions include:

- Slowing event generation
- Making the event handler more efficient
- Sending some processing to be done asynchronously by another event handler
- Avoiding database interactions where possible, as this is always a synchronous operation; using service state is much more performant

In our example below, see two MQTT sources established by the developer: RPMSMQTT and TempMQTT. The event rate, drops, and other statistics display for each source. This is useful for developers who need to know more about the load and balance of source messages flowing into the system.

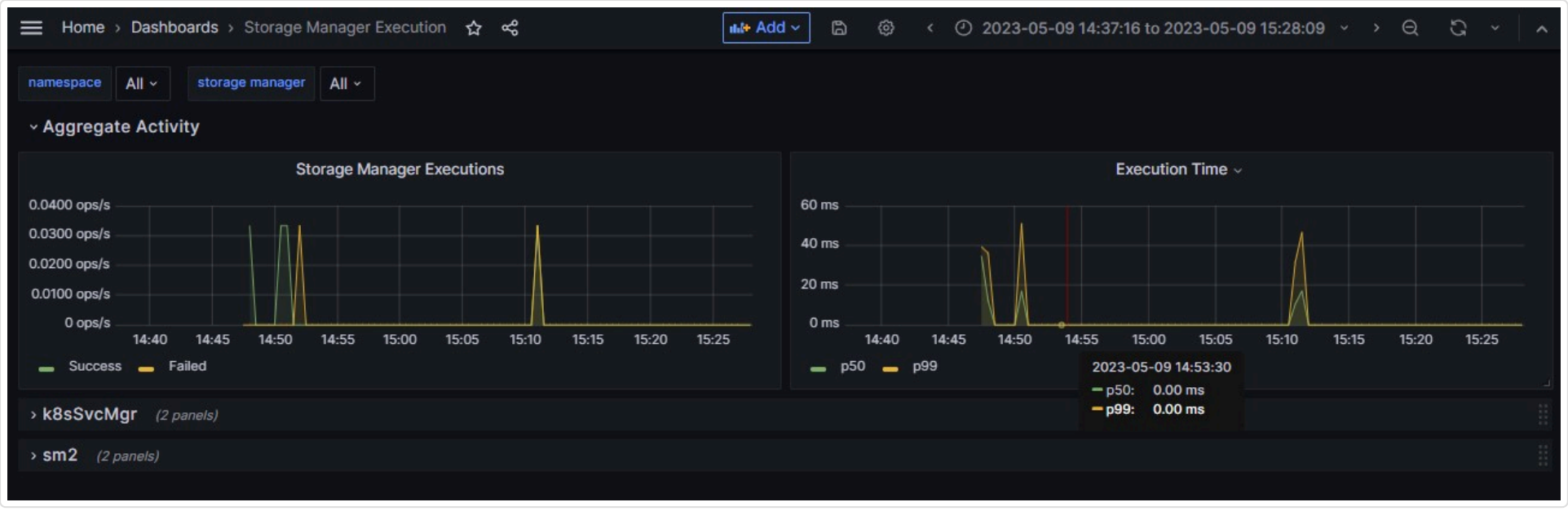
Much of this information is also displayed in the App Execution dashboard if incoming events are ingested by an App. The Event Processing dashboard also displays Sources as one line in the aggregate graph, with a separate pane for sources below.

Expanding the panels for RPMSMQTT reveals the throughput rate, including messages received and dropped. A second panel is for Operation Response Time.



Storage Manager Dashboard

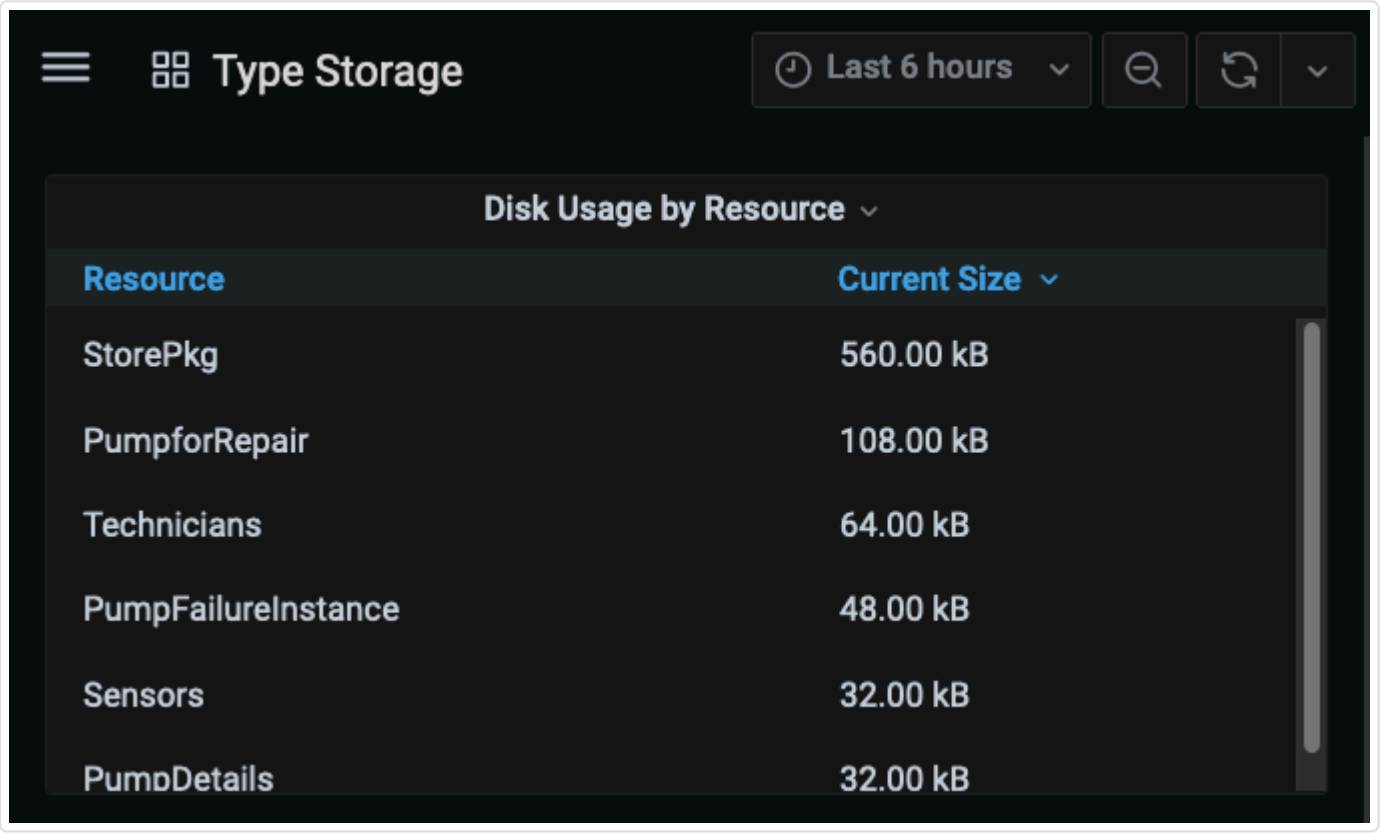
The Storage Manager Dashboard displays performance data and resource usage for Storage Manager operations running in the namespace.



Type Storage Dashboard

The Type Storage dashboard displays disk usage for developer-defined standard data types in the project. The information is up-to-date and not based on what has occurred in the selected time frame, though developers might have to choose a time window longer than the default in order for the data to appear.

As a general rule, developers should have a model of the application and all the resources it is expected to use. Use this dashboard as a “sanity check” to ensure that the system is utilizing the expected amount of storage. If there is a substantial divergence, investigate whether or not the model is flawed, or if the system isn’t implemented properly.



Deprecated Dashboards

Since standalone Apps are deprecated, these dashboards are also deprecated.

- App Execution - Metrics on event processing through standalone Apps, including Activity Task granularity
- App Profiling - Threaded, additional metrics available when “profiling” has been enabled for a given App
- App With Split Execution - App Execution Dashboard, with the addition of data for SplitByGroup Activity Tasks

App Execution Dashboard

The App Execution Dashboard visualizes event activity in standalone Apps within the Namespace, expressed in operations per second.

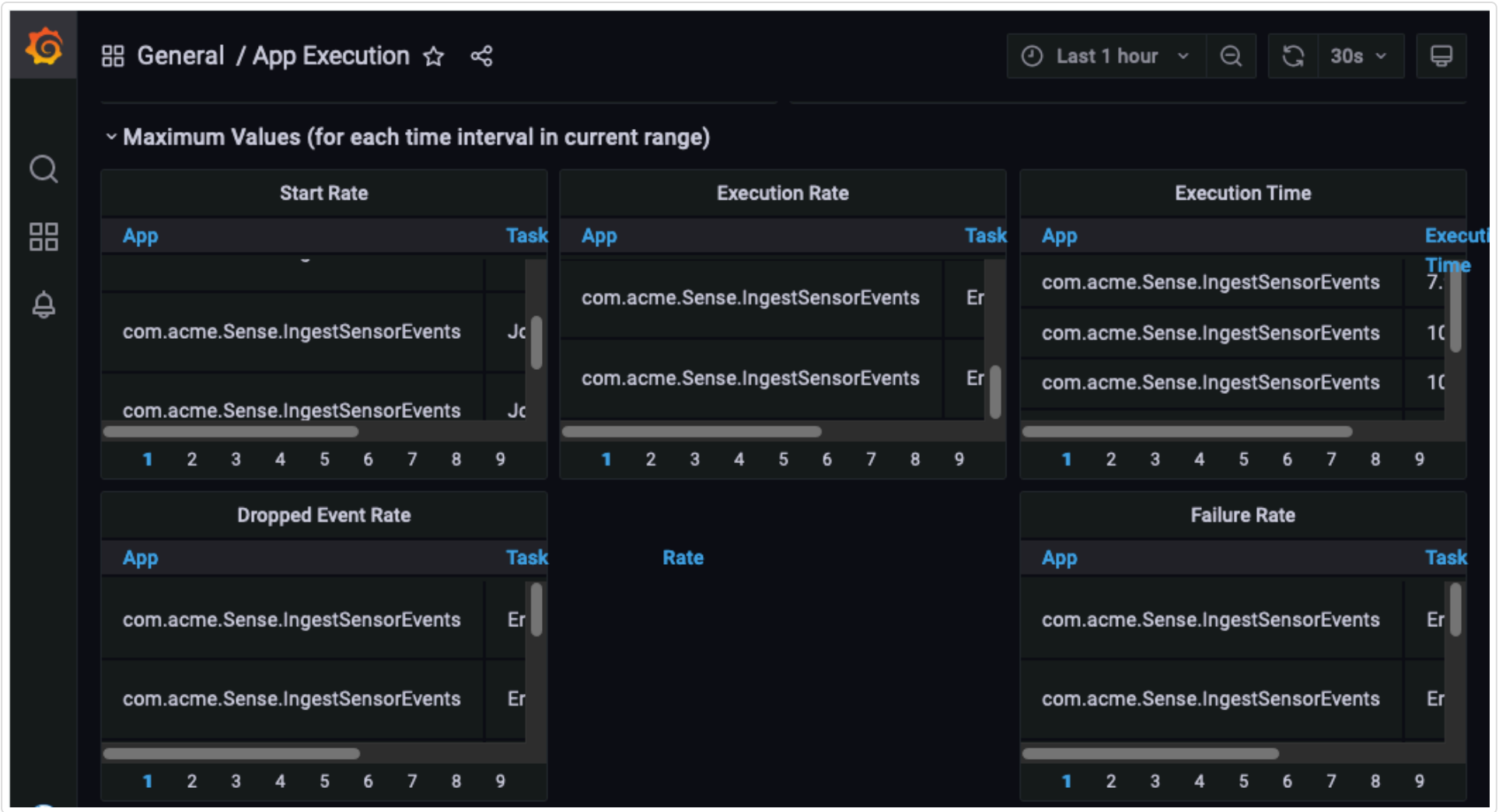
Here are some of the critical App behaviors to observe in this dashboard:

- Executions must match the volume of work expected for a given workload
- Ensure that execution times remain stable as loads change
- Observe standalone Apps individually to detect potential bottlenecks

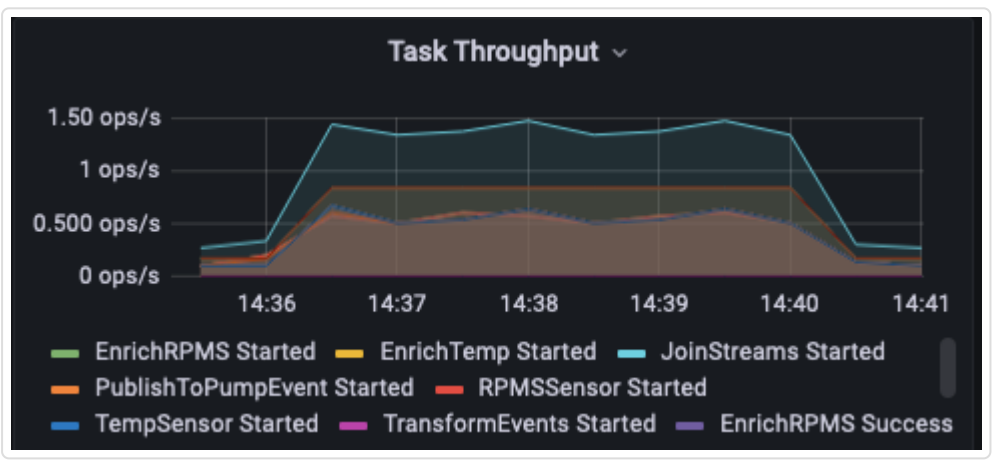
As a general rule of thumb, for high-volume systems, response times should remain under 1000 ms. Also, if execution times grow disproportionally higher as the load increases, it is a scalability problem. Using this dashboard, developers can identify the individual Apps and the Activity Tasks within them that are taking the longest to process.

Two panels for **Aggregate Activity** display the sum total of executions for all Apps; the first expressed in operations per second, and the second as a function of the total execution time. At the bottom, the green line **p50** is the line for median execution time, and **p99** is the time overall. The aggregate information is useful to verify that there are no problems, but if there is, looking at Apps individually will pinpoint the one(s) where there is an issue.

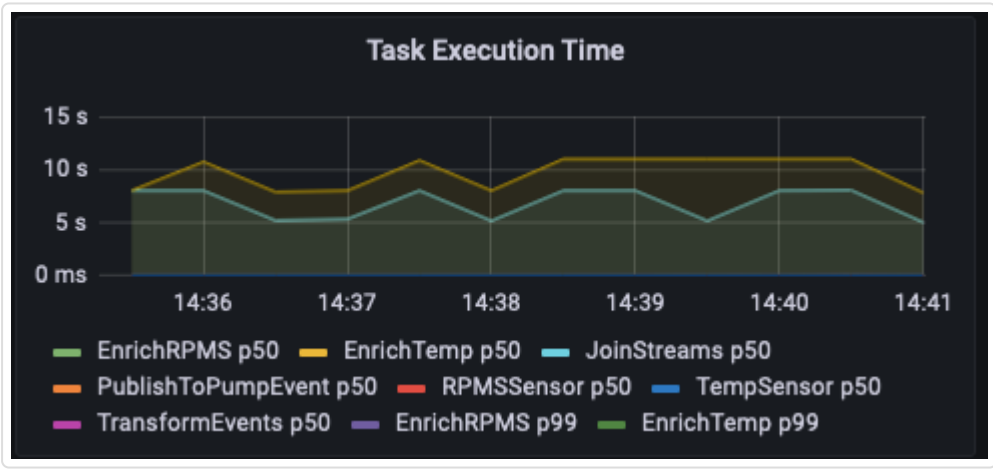
Below the Aggregate Activity display panel, click the symbol next to **Maximum Values** to reveal the highest throughput rate levels, expressed in operations per second, of each of the Activity Tasks listed for each of the Apps running in the namespace.



The App-specific charts below the Aggregate information display a graph of activity task throughput, with each line representing starting and success rates of each. Here, we've achieved a "close-up" of part of this graph by clicking on the chart and dragging across the region of interest.



The **Task Execution Time** line chart shows the median and overall execution times by activity task for the App. Notice in this case, only two lines are obvious, but many other displays are blended into them. Click on the named labels to see individual displays.

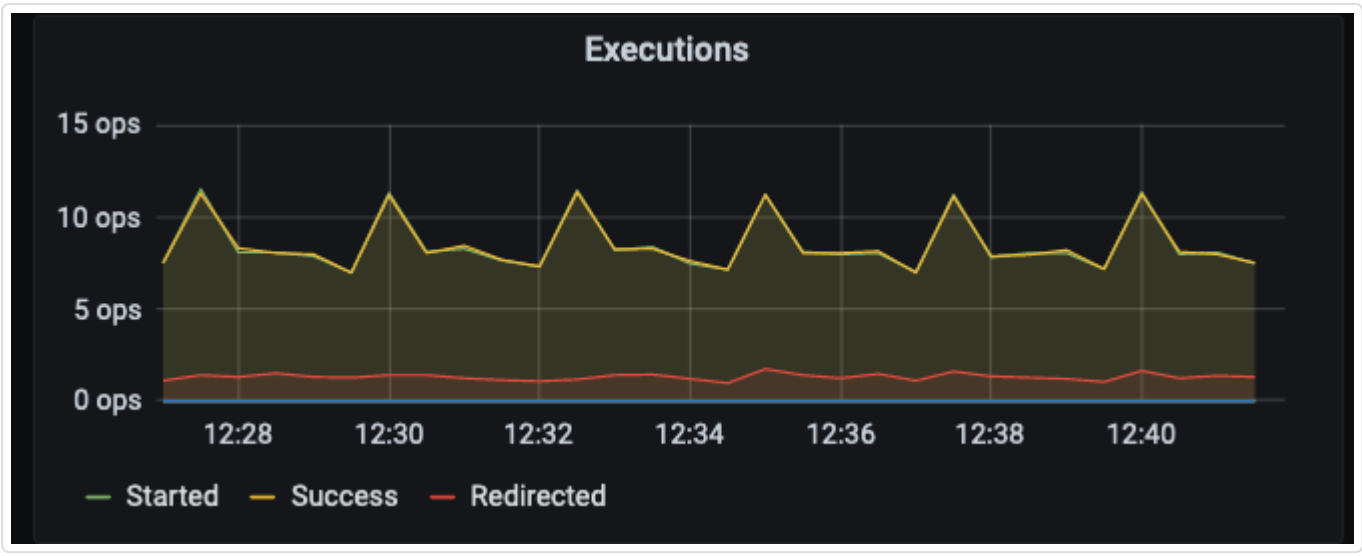


The App Execution Dashboard helps developers investigate the throughput performance of their standalone Apps by each of their Tasks, or as a whole.

App With Split Dashboard

The App With Split Execution Dashboard includes the same query panels available in the App Execution Dashboard, plus more for SplitByGroup activity tasks within standalone Apps.

The panel for Aggregate Executions includes a separate line for Redirected Executions, which will not be visible unless one or more Apps in the namespace is using a SplitByGroup activity task.



There is an aggregated panel showing collective redirect latencies for all standalone Apps' SplitByGroup activities. The developer can also see the Split Rates and Redirect Latencies for each individual App. Apps that do not use a SplitByGroup task will show no data. In this example, the app has a SplitByGroup activity task called "SplitById;" rates and median/total redirect latencies are displayed for it.