3.0 Advanced Dashboard Primer

Aren't you an intrepid explorer! Look at you jumping right to the advanced section! Well, we are glad you are here. In this section we'll be looking at how you can create much more complex dashboard content.

As soon as you want to go beyond supplied panels, or direct data from inventory queries, then things are going to get a bit more complex. However, don't be daunted! We are going to give you the building blocks here for building complex and very meaningful aggregate data from FileWave sources.

Please see the articles below for the elements we'll build on:

3.1 Aggregating Data

What

Up until now, we have been talking about the capability of the FileWave dashboard to show data. We have looked at pre-built examples, and done a small amount with data from an inventory query. But the real power in the dashboard is the ability to aggregate (or summarize) data.

When/Why

Let's consider a new deployment of Java to the environment: We can create a report that shows us every client, and every version of Java on those clients, but that doesn't give an easy overview of how the rollout from build 183 to build 196 is going. But, if we could take that same data, and count the number of clients with each version and represent that in a chart, then that would give us the exact picture we might be looking for. This aggregation of data is a very powerful tool.

How

So, we know how to create new dashboards and panels, but how do we actual get aggregated data over to our (grafana) dashboard? The answer to that lies in using Prometheus and configuring a scrape file for collecting and aggregating that data.

3.1.1 Grouping Data Using Prometheus

What

In order to do summary reporting, we need to leverage the power of Prometheus.

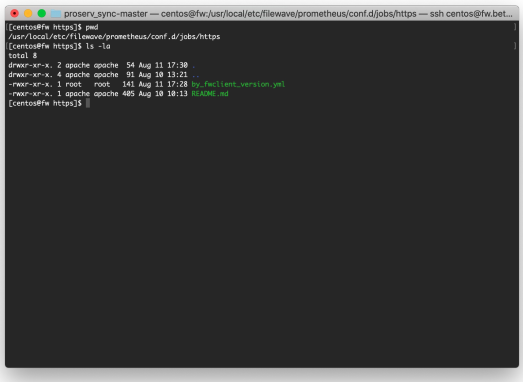
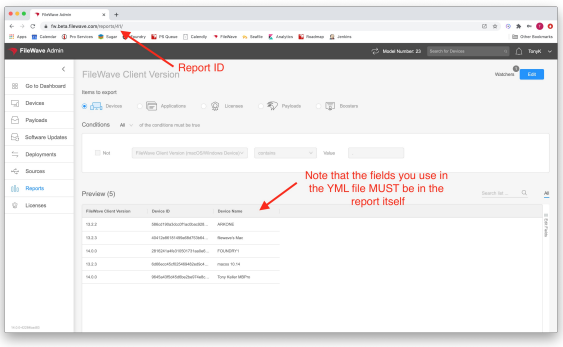
When/Why

Anytime we want to do something like report on a rollout or general status, we are going to want to summarize a report. We will accomplish this by using a Prometheus config file on the FileWave server itself.

How

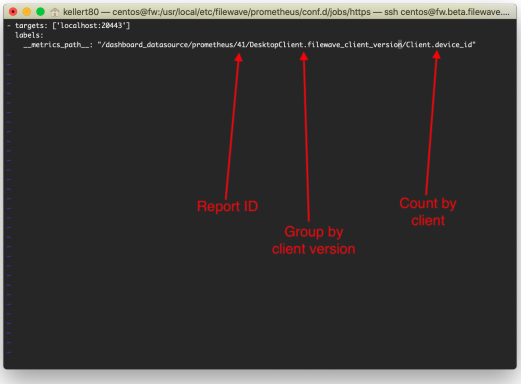
The configuration (or yml files) that we'll create will always be placed in the /usr/local/etc/filewave/prometheus/conf.d/jobs/https directory on the FileWave server. Anything placed in this directory will automatically be read by Prometheus, and the data presented to our dashboard. (Example yml files can be found in /usr/local/etc/filewave/prometheus/conf.d/jobs)

The syntax of these files is quite picky, so it is best to copy an existing one, and then modify it. It may seem complicated, but we are always going to do the following steps:

Step	Example
1. Place a new (or copied) yml file into /usr/local/etc/filewave/prometheus/conf.d/jobs/https with a meaningful name.	
2. Edit the new file to specify the following 3 things: * The inventory query (report) to use * The field you want to count by...device_id is almost always a good one if reporting by device * The field you want summarize (aggregate) by...in this case, the filewave client version	
3. Once your report is created, the report id to use is most easily accessed through the webadmin. Note that the fields you want to use for aggregation must be in the report.	
4. Get the definition for the fields you want to use from the API...the easiest way is to do a curl from the command line like this: `bash curl -s -k -H "Authorization: <Base64 API Token>" https://<my.server.address>:20445/inv/api/v1/query/<report_id> | python -mjson.tool ` Make sure and substitute in your values for the <Base64_API_Token>, <my.server.address> and <report_id> You'll get a response that includes the component and the field names as shown at right	

```
query/41 |   python -mjson.tool
{
    "criteria": {
        "expressions": [
            {
                "column": "filewave_client_version",
                "component": "DesktopClient",
                "operator": "contains",
                "qualifier": "."
            }
        ],
        "logic": "all"
    },
    "favorite": false,
    "fields": [
        {
            "column": "device_name",
            "component": "Client"
        },
        {
            "column": "filewave_client_version",
            "component": "DesktopClient"
        },
        {
            "column": "device_id",
            "component": "Client"
        }
    ],
    "group": 0,
    "id": 41,
    "main_component": "Client",
    "name": "FileWave Client Version",
    "version": 3
}
```

4. Edit the YML file to specify the 3 items as they match your report definition, then save the file. If using the sample file, remember to take out the comment # at the beginning of each line. Example at right:



Within a minute or two of creation of the file, the data should be available in your dashboard for a new panel.

3.1.2 Testing the Prometheus Scrape

What

Assume for a moment you made a typo in the yml file, or some other problem occurs and your new scrape isn't showing in Grafana...how can you see what is going on?

When/Why

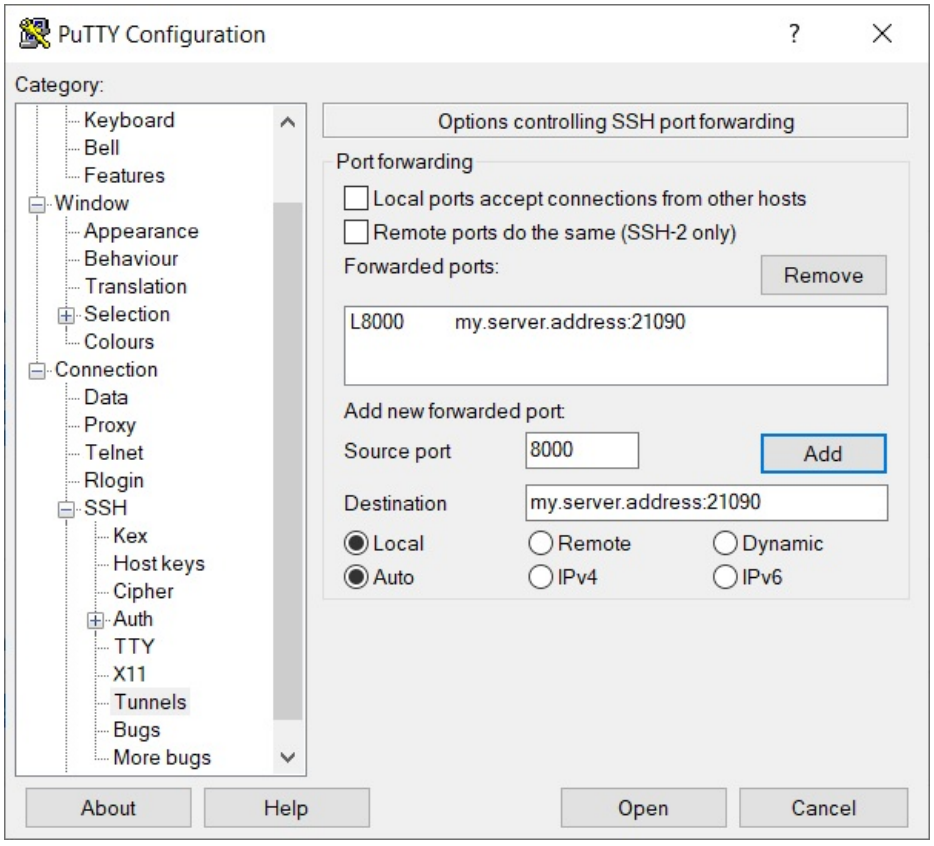
Thankfully there is a service running that allows you to see the status of all Prometheus scrapes, and will usually give you an idea about what is going on. We can check a web page to get this detailed information.

How

The webpage/port in question though is NOT open by default to external systems and we must access it in a special way. If the port were open, we would normally just go to https://my.server.address:21090/targets. But, to work around the port not being opened, we can do the following (from terminal, macOS):

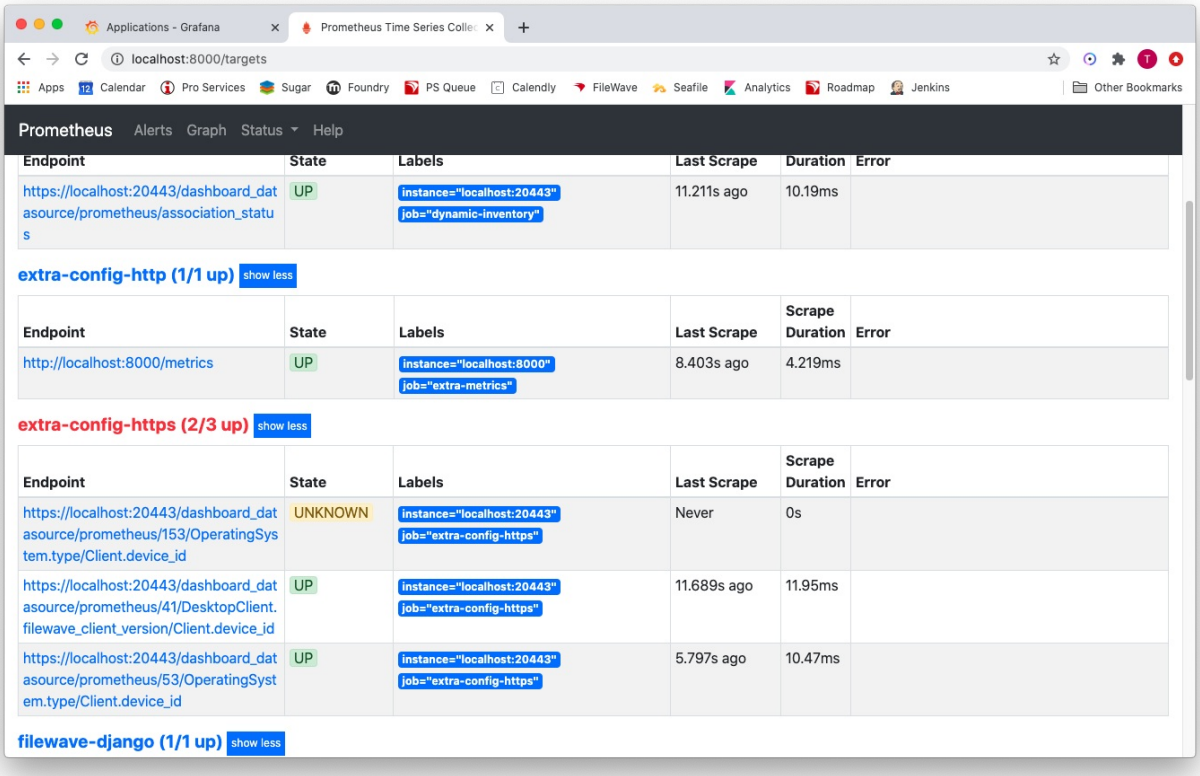
```
ssh -L 8000:localhost:21090 user@my.server.address
```

Alternatively, if you are on a Windows device, you can do the same thing through Putty by configuring a "tunnel" with local port of your choosing redirecting to the FileWave server:



These configurations redirects our requested traffic back to port 8000 on our local device, based on the ssh connection being established. The result is that in our own browser then, we can go to http://localhost:8000/targets to see the scrape data.

See below how I have a mistake in one of my jobs (query 153 doesn't actually exist):


Applications - Grafana
Prometheus Time Series Colle
localhost:8000/targets
Apps
Calendar
Pro Services
Sugar
Foundry
PS Queue
Calendly
FileWave
Seafile
Analytics
Roadmap
Jenkins
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Endpoint
State
Labels
Last Scrape
Duration
Error
https://localhost:20443/dashboard_datasource/prometheus/association_status
UP
instance="localhost:20443"
job="dynamic-inventory"
11.211s ago
10.19ms
extra-config-http (1/1 up)
show less
Endpoint
State
Labels
Last Scrape
Scrape Duration
Error
http://localhost:8000/metrics
UP
instance="localhost:8000"
job="extra-metrics"
8.403s ago
4.219ms
extra-config-https (2/3 up)
show less
Endpoint
State
Labels
Last Scrape
Scrape Duration
Error
https://localhost:20443/dashboard_datasource/prometheus/153/OperatingSystem.type/Client.device_id
UNKNOWN
instance="localhost:20443"
job="extra-config-https"
Never
0s
https://localhost:20443/dashboard_datasource/prometheus/41/DesktopClient.filewave_client_version/Client.device_id
UP
instance="localhost:20443"
job="extra-config-https"
11.689s ago
11.95ms
https://localhost:20443/dashboard_datasource/prometheus/53/OperatingSystem.type/Client.device_id
UP
instance="localhost:20443"
job="extra-config-https"
5.797s ago
10.47ms
filewave-django (1/1 up)
show less

3.1.3 "Exploring" Your New Aggregate Data

What

We don't have to jump right in to creating a proper reporting panel for our new data. Many times simply looking at the data itself can be helpful.

When/Why

It is best to always take a look at your resultant data just to give it a sanity check before you start doing formal reporting. The easiest way to do this is simply to use the "Explore" function within Grafana.

How

The data we previously aggregated from Prometheus will always be presented in the form of filewave_inventory_query_id, where id is the id of the report you based the data upon. You'll see below how we can explore that data:

Notice that the data are no longer 5 individual records, but now 3 records, because we have 3 different versions of the FileWave client between these endpoints. That data is perfect for us to create our very own pie chart.

3.1.4 Creating your Data Panel

What

Now that we have our data being passed over to the dashboard, and we know how to access it, we can build a panel using the visualization that fits our data.

When/Why

Many times, when the data we are presenting is representing the "whole" of an environment, then we might choose to use a pie chart to represent that data, and usually that data is presented in a snapshot form (i.e. only the latest data). At other times, when we want to watch the progression of something, such as number of enrolled devices, we might want to look at that data as it progresses over time in a line graph.

How

Creating this new panel isn't much harder than our previous example, but there are a few new things you'll notice in the overview below:

There are lots of options you can play around with in the different visualizations to customize the output (we won't attempt here to document that). Note though that you can reference data elements for things like the legend. In this case putting {{genericdesktopclient__filewave_client_version}} in as the legend format makes the legend use the much more meaningful field value.

One more note: We chose the "Instant" option in this case because we only want to see the last version of the data for this particular panel (and almost always "instant" for a pie chart). However, the data really is time-series data, so a bar or line chart could show these data elements changing over time to watch progression.

3.2 Extra Metrics

What

We learned in the 3.1 section how to build our own custom panels. "Extra Metrics" is an independently built tool to automate creation of a few reporting elements for us without doing it manually. This solution is NOT directly supported by FileWave, but you may find it useful in your environment.

When/Why

"Extra Metrics" gives data on applications, and generally on patch status of your devices. The patch status elements are hard-coded, but the application versions panels are driven by dynamic reports that you can tweak to fit your needs. All information for installation and upgrade of this solution is found here: https://pypi.org/project/filewave-extra-metrics/#description

How

In this example, we have Extra Metrics installed, and are adding a new report to view Firefox information on macOS devices.