S21-78-DASH
Dashboard Tutorial
Southern Illinois University Carbondale

Team Members:

Jake Hill(EE), Cameron O'Brien(ECE), Elijah Manier(CE), Alex Manuel(CE)

Table of Contents

Introduction:

This document is meant to serve as a guide and tutorial for the upkeep and maintenance of the ECBE Dashboard created by the S21-78-DASH team. It will include tutorials on using Grafana, associated plugins, team code projects and Prometheus. This will also cover the installation of these tools. All the material used in the project is hosted on the team's GitHub located here. This includes graphics, xml files and source code files created by the team.

Installation:

The team has created a basic system install script to help facilitate getting a version up and running on a new system. You will need to download the zip file from here, then unzip and run install.exe with administrator privileges. This will create a directory in the C: drive called "Dash Project Files". Once completed it will unzip the contents of the zipped file into the created directory and remove all the unneeded files. You will then be prompted to choose whether you want to install the software as services. This includes, Grafana, Prometheus and the PingerApp. Once this has been completed the system will need restarted and all services will be accessible.

If for any reason a failure occurs the installation can be done manually, using the following steps:

1. Create Install Directory
2. Unzip Sub Directory Dash-Windows-Files.zip into new directory
3. Get file paths for NSSM.exe and Service.exe (Ex. Grafana) from new directory
4. Open a command prompt window in administrator Mode
5. Run: "NSSM-FILE-PATH" install SERVICE-NAME "SERVICE-EXE-PATH"
6. Do this for all required services

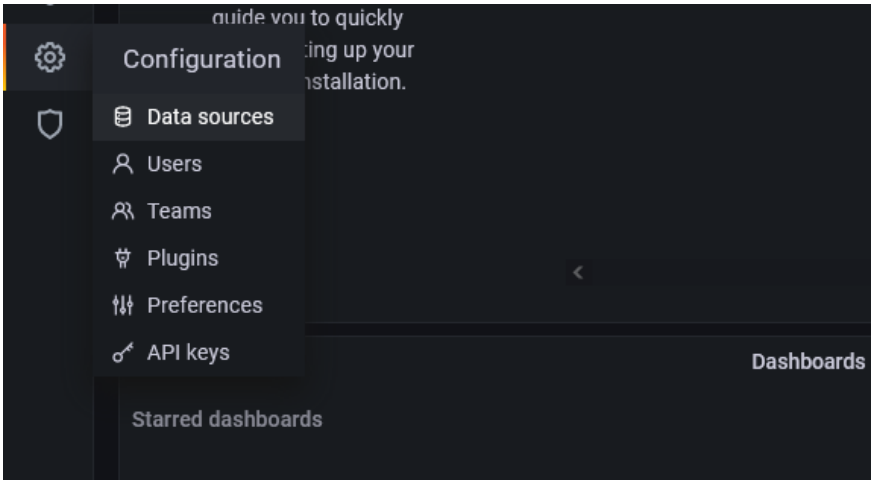
Grafana:

Grafana is the open-source software utilized by the team to display the metrics that are collected using various graphical interfaces and display methods. Can be accessed by default at localhost:3000 in any browser.

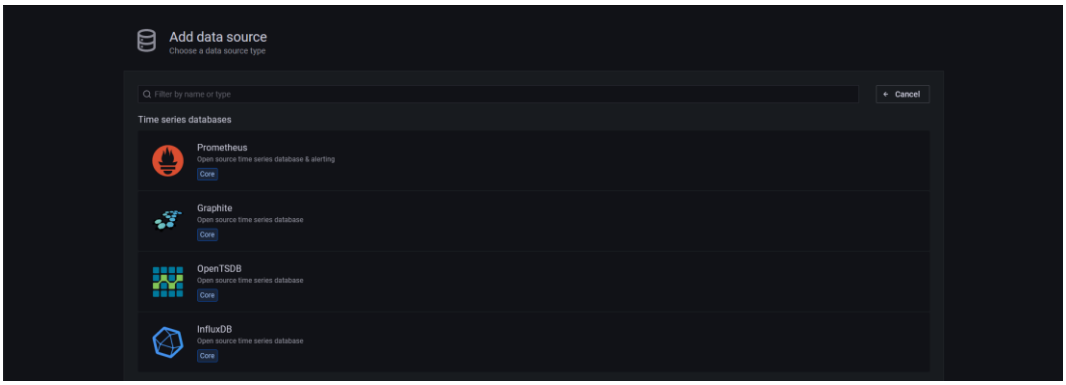
Configuration:

New Data sources:

1. Adding a new data source is accomplished by navigating to settings menu bar located on the left side the screen on any Grafana dashboard and Hovering over the cog and selecting the data sources tab:



2. Once on this screen clicking the new data source button will bring you to the screen below. The team utilized only Prometheus and TestDataDB. TestData only generates random values in order to create mockup dashboards.



3. Clicking a data source type will bring you to a configuration page where you will input the location of your data source using the IP and port combo. Various other options are source dependent.

4. Once finished you can save and test to make sure the data source is available. If it is available there will be a popup at the bottom of the screen stating the data source is working. You can now incorporate these metrics into your panels.

Plugin Installation:

Grafana has plug in support, these can be searched on the Grafana website here. These pages include guides for installation and are a great resource.

Grafana has made it straight forward to add plugins. This can be done by launching PowerShell and using cd command to navigate to the Grafana labs directory. This may vary depending on your installation. The default directory is located at:

C:\Dash Project Files\GrafanaLabs\grafana\bin

Once in this directory, run the command:

./grafana-cli plugins install “PLUGIN NAME”

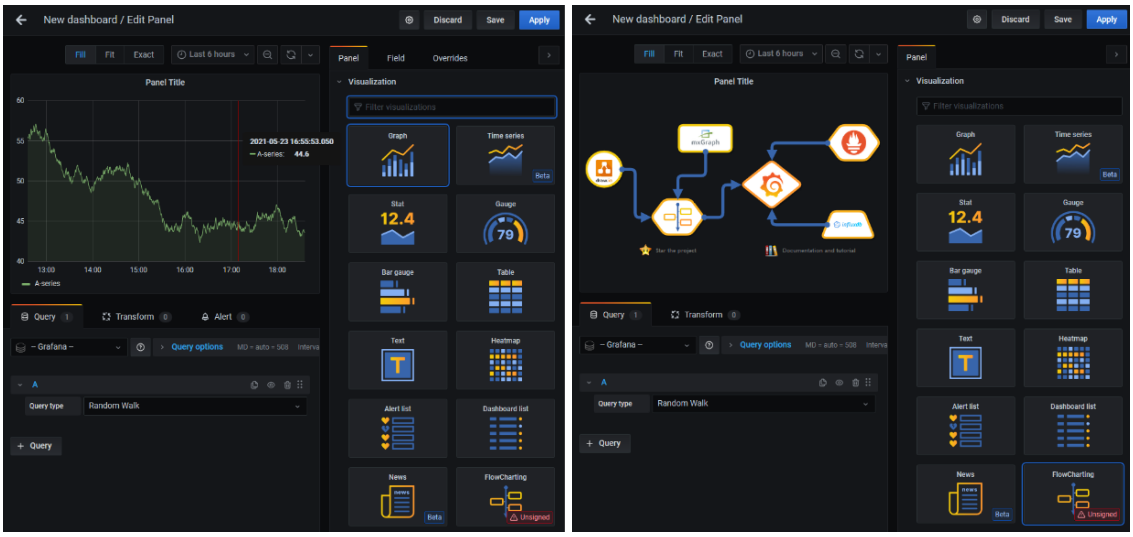
After the installation process is complete the Grafana service needs to be restarted. This can be done by restarting the machine, or by utilizing the Services application to restart the service. Once this is done the plugin will be ready for use.

Flowcharting Plugin:

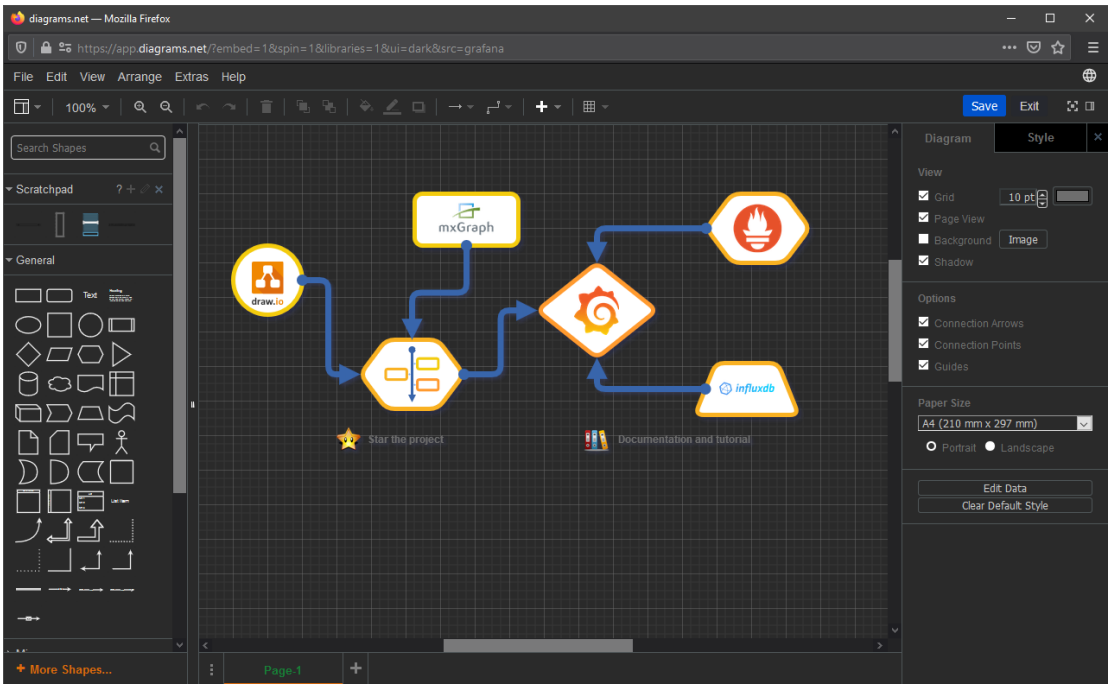
This plugin extends the use of Grafana using custom elements that allows information to be displayed visually without the need for input of the user, additionally these elements can be used to create menus useful in easily navigating through a dashboard. Additional information can be found on the plugin’s GitHub page here.

Basic Usage:

1. This process is started by creating a new panel in your Grafana dashboard. Once this is completed you should see a screen like the image on the left below.
2. This panel can be converted into flowcharting panel by clicking on the icon in the visualization tab it should look like the image on the left.



3. After the panel is created the contents of panel can be edited by navigating to the Flowchart tab in the settings. Clicking the “Edit Draw” button will bring up a new window in draw.io. From this window you can edit the elements on the panel.

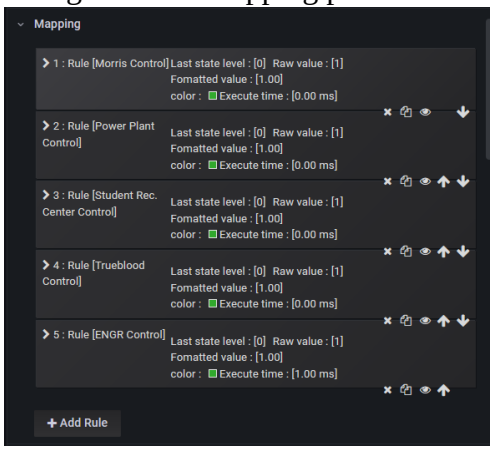


NOTE: You must click save in the draw.io window, this is in the top right of the screen. After doing this the changes must also be saved inside of Grafana for your changes to be kept. This is in the top right corner of the screen as well.

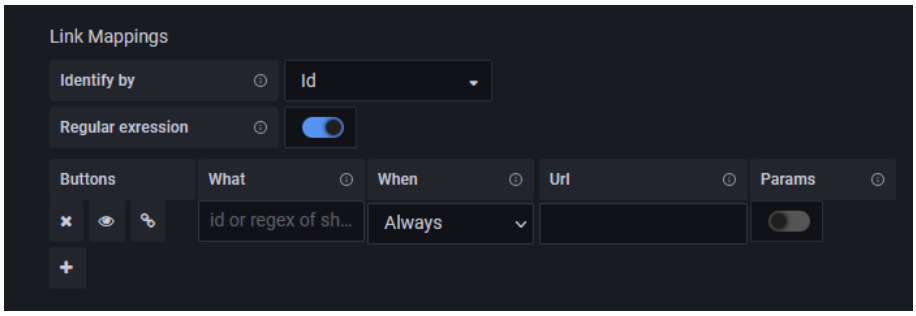
Linking:

Linking is a feature utilized very often by the team in the current implementation of the dashboard. This can be incorporated in a few easy steps.

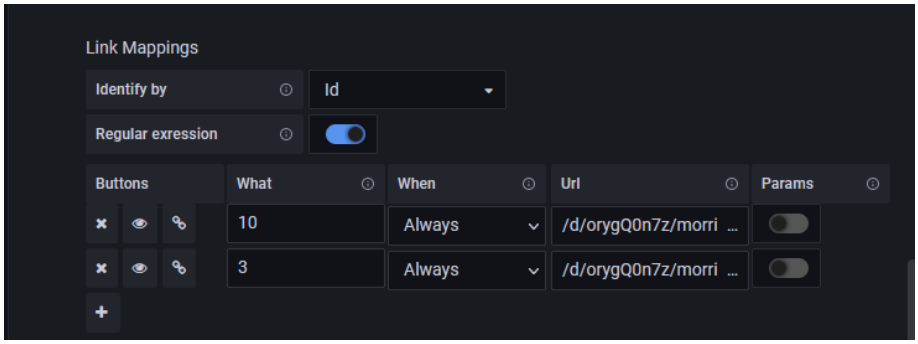
1. Click the dropdown option on and edit the flowcharting panel
2. Navigate to the mapping portion of the options page, should look similar to this.



3. Click the new rule button and navigate to link mappings portion of the settings and click the + button to create a new Link mappings rule. Your screen should look like this.



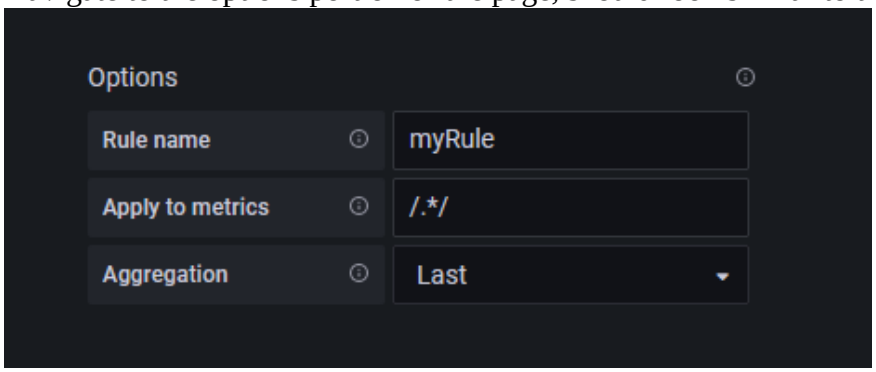
4. Pressing the two-target symbol will allow you to select the object off the diagram. Alternatively, you can just type in the ID number of the element you would like to make linked. A finished linked element should look as follows



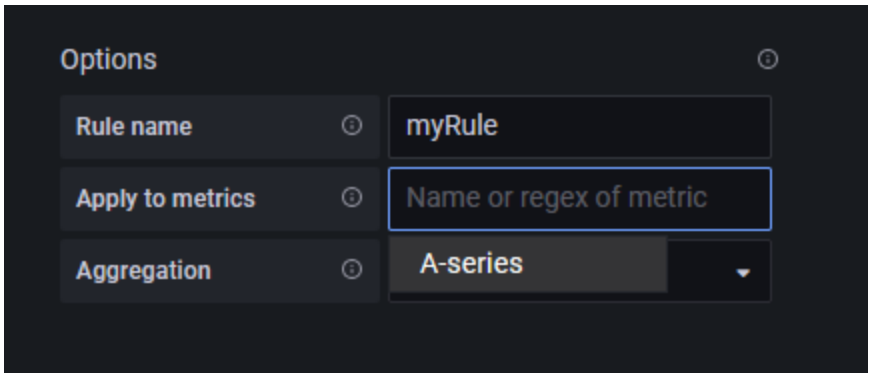
Dynamic Elements:

These elements are another key part of the current implementation of the dashboard, but unlike the previous elements these are a bit more difficult to create. If any part of this process is unclear a more detailed guide can be found here.

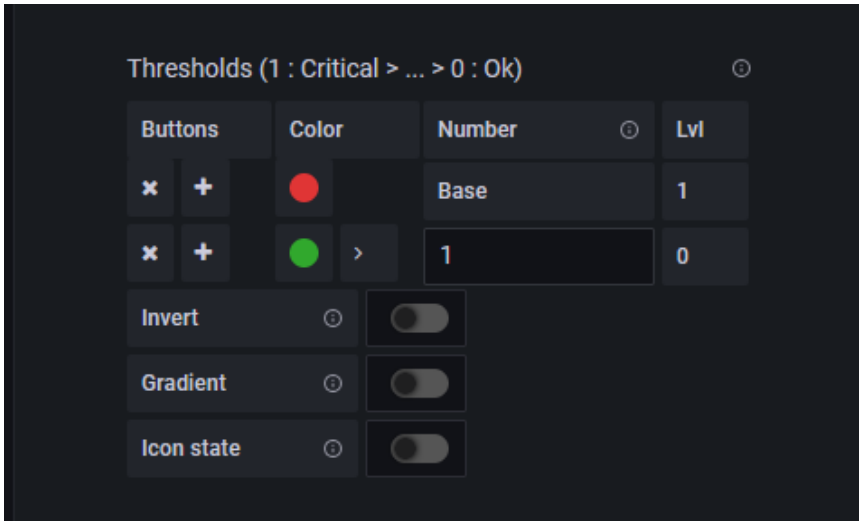
1. Click the dropdown option on and edit the flowcharting panel
2. Navigate to the options portion of the page, should look similar to this.



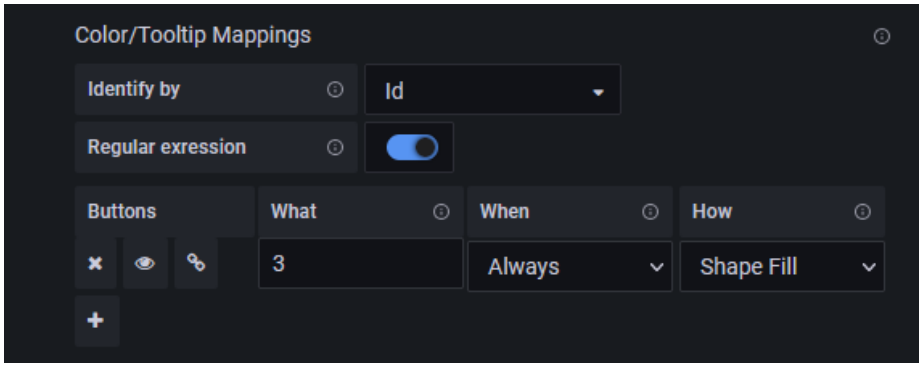
3. You must have an imported metric to be available to be used in this example we use random walk data. Here we select the datasource A-series



4. Next you can change the thresholds and various others using the settings menu below the previous section. The example below the element is green when the scraped value >1 and when it is <1 it is red.



5. Finally, pressing the two-target symbol will allow you to select the object off the diagram. Alternatively, you can just type in the ID number of the element you would like to make linked. A finished linked element should look as follows



Note: There are many different versions of dynamic elements these can be seen in the guide on the plugin GitHub these are just the most utilized in the dashboards.

Prometheus:

Prometheus collects and stores metrics into a time series database. There are additional metrics scraped generated by various sources such as Node Exporter or the PingerApp. Can be accessed by default at localhost:9090 in any browser.

Configuration:

The Prometheus installation is configured using the yaml file that by default is located at:

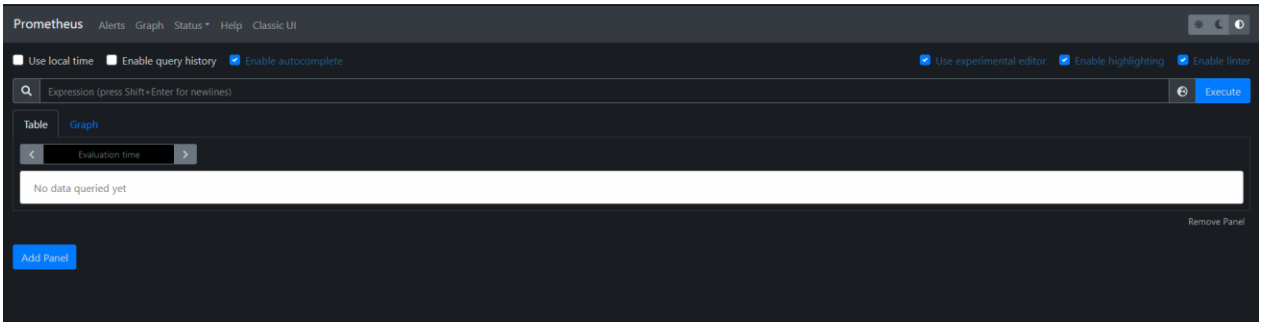
C:\Dash Project Files\Prometheus\prometheus.yml

Scrape targets can be added from this configuration file. Once Prometheus has rebooted either by restarting the program or the service, new targets will be scraped. An example of a custom target added by the team can be seen below:

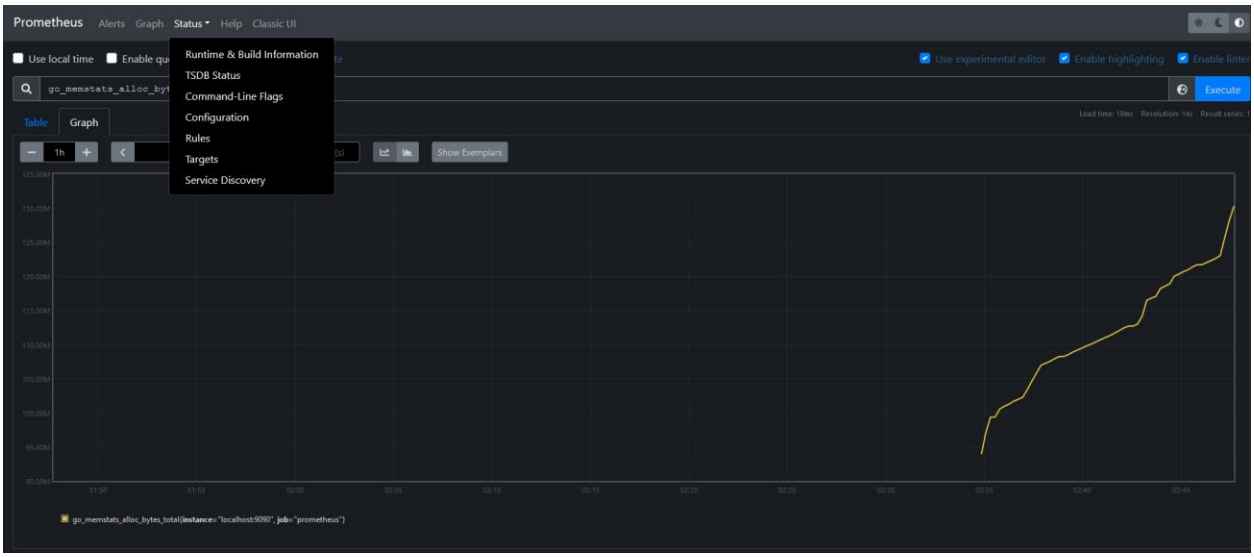
```
- job_name: 'Custom Exporter'
  # metrics_path defaults to '/metrics'
  # scheme defaults to 'http'.

  static_configs:
  - targets: ['localhost:8000']
```

Application Navigation:

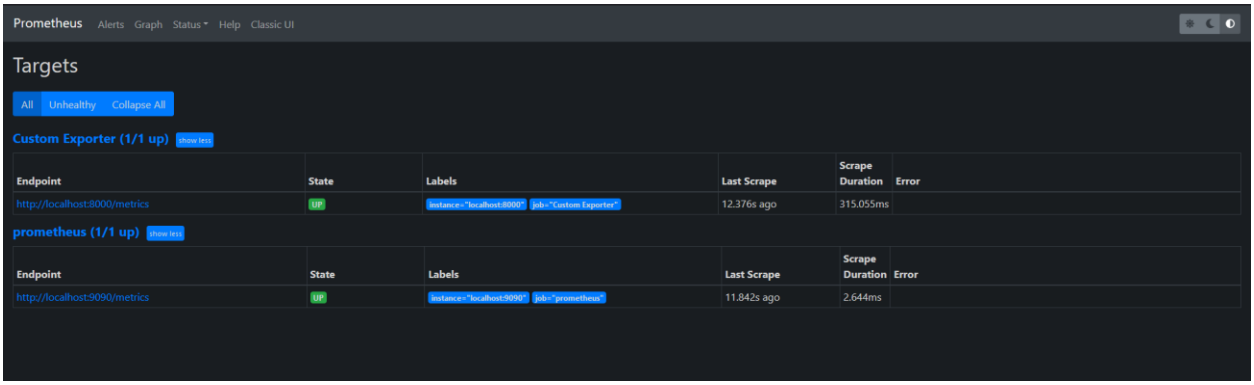
The user interface for Prometheus can be accessed by typing localhost:9090 into a browser of your choice. Once the page is loaded it should look like this:



This page is very useful, and you can search through the various metrics Prometheus is currently collecting. This allows you to find the name of the metric and easily copy and paste it into Grafana. You can also graph these metrics in a basic form in Prometheus using the graph tab on the page. Ex:



Another useful feature can be navigated by clicking the drop-down menu called Status. Once in this menu clicking Targets allow you to monitor the status of all the scrape targets Prometheus is currently using. This is extremely useful for troubleshooting. The Page should look similar to this:



PingerApp:

Configuration:

The PingerApp features a basic configuration file located at:

C:\Dash Project Files\PingerApp\TargetList.txt

The description is included inside of the file. It will be posted here as well.

DASH Senior Design Project

IP Pinger and Host for Prometheus Data Source

Accompanying Python programs host metrics on localhost:8000

The items are separated by a space do not add any blanks rows

If a space is needed use an asterisk at position 0 of the document

The line will be parsed out.

Below are the listed targets the format is as follows

Gauge_Name Method_Name IP_Address

This format allows many different scrapes to be collapsed under one

Gauge please utilize this and categorize scraps

The *? functions an interrupt and specifies the sleep interval between pings

Function Overview:

The PingerApp accomplishes a very simple task of pinging a list of targets using the ICMP protocol, then hosting this information on server at localhost:8000 to be scraped by an instance of Prometheus.

Ping():

ping(addr, timeout,number,data)

Simple function that takes in an address and timeout parameter that controls how long the function waits for a response. A Ping is sent over the ICMP protocol. If a response is received the function returns 1, if not it returns a 0.

readTargetFile():

readTargetFile(List,fileName)

Function that takes in a fileName and opens the configuration file at the location and writes all the targets into the variable List. It uses ‘*’ as an interrupt character in the configuration file. ‘*?’ specifies the sleep parameter this is stored into the global variable

Sleep if the interrupt character is followed with anything else it functions as comment character.

createGauges():

createGauages(List)

Function takes one parameter List, which is generated from the configuration file using readTargetFile(). It iterates through entries of the list and checks if the specified target has been added to the gList, if not it is added. The function adds the gauge target from the Prometheus client library as fourth entry in the list. After all the information has been added one final loop sets the Gauge and method.

Main():

Generates path name from launch directory. Calls functions to read config file and create the gauges then starts the server. Main continually pings the items on the list, waiting Sleep seconds between each iteration. If a ping fails, it retries 3 more times to make sure the dropped packet wasn't due to turbulent network conditions.

Original Design Overview:

BASIC OVERVIEW OF HOW THE SYSTEM IS SETUP ON SIU INFRASTRUCTURE