

IxOS Monitoring Solution.

Deploying the monitoring solution.

1) Clone / Download IxPortUtilizationPlotter locally

```
git clone https://github.com/yourusername/IxPortUtilizationPlotter.git
```

```
cd IxPortUtilizationPlotter
```

2) Setup the credentials and setup details to start the influxDB, Graphana and Prometheus container.

- Rename .env and add relevant values in it (cp env.example .env)
- Add the Chassis you want to monitor inside **.env** OR set it in the **config.py**

```
4      # Service Ports (customize if defaults conflict with existing services)
5      INFUXDB_PORT=8086
6      PROMETHEUS_PORT=9090
7      GRAFANA_PORT=3000
8
9      # InfluxDB Configuration
10     INFUXDB_ADMIN_USER=admin
11     INFUXDB_ADMIN_PASSWORD=admin
12     INFUXDB_ORG=keyight
13     INFUXDB_BUCKET=ixosChassisStatistics
14     INFUXDB_TOKEN=your-super-secret-token-change-me
15     INFUXDB_RETENTION=0 # 0 = infinite retention, or specify in seconds
16
17      # Grafana Configuration
18     GRAFANA_ADMIN_USER=admin
19     GRAFANA_ADMIN_PASSWORD=admin
20
21      # =====
22      # Poller Configuration (Optional)
23      # =====
24      # The poller runs on your HOST machine (not in Docker)
25      # These variables are read by config.py if set
26
27      # If not set, config.py uses its default values
28
29      # InfluxDB Connection URL for poller
30      # Use localhost if services run on same machine, or remote host IP
31      INFUXDB_URL=http://localhost:8086
32
33
34      # Polling interval in seconds - This is for my influxDB to select metrics push intervals
35      POLLING_INTERVAL=10
36      | %L to chat, %K to generate
```

3) Start InfluxDB, Prometheus and Grafana containers from the docker-compose file withing the repo

> docker compose up -d

```
base ~/IxPortUtilizationPlotter git:(main) (0.032s)
docker compose up -d
WARN[0000] /Users/ashwiyash/IxPortUtilizationPlotter/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 1/2
  ✓ Network ixos-network   Created
  [+] Running 6/6getheus-data" Creating
    ✓ Network ixos-network   Created
    ✓ Volume "prometheus-data" Created
    ✓ Volume "grafana-data"   Created
    ✓ Container influxdb     Healthy
    ✓ Container prometheus   Healthy
    ✓ Container grafana      Started
      0.2s
      0.0s
      0.2s
      0.0s
      0.0s
      6.7s
      6.7s
      6.8s

base ~/IxPortUtilizationPlotter git:(main) (0.239s)
docker ps
CONTAINER ID IMAGE           COMMAND          CREATED        STATUS          PORTS          NAMES
61d10b3d502e  grafana/grafana:latest  "/run.sh"       51 seconds ago  Up 44 seconds (healthy)  0.0.0.0:3000->3000/tcp, [::]:3000->3000/tcp  grafana
28c1f3b71b3   influxdb:2           "/entrypoint.sh infl..."  51 seconds ago  Up 50 seconds (healthy)  0.0.0.0:8086->8086/tcp, [::]:8086->8086/tcp  influxdb
90bf1f812ae60  prom/prometheus:latest  "/bin/prometheus --c..."  51 seconds ago  Up 50 seconds (healthy)  0.0.0.0:9090->9090/tcp, [::]:9090->9090/tcp  prometheus

  A Help me manage running containers. % ↵
```

4) Confirm that containers started OK by logging in these portals (Note the UN/PW you set in .env OR then the defaults configured)

localhost:3000/?orgId=1&from=now-6h&to=now&timezone=browser

Grafana

Home > Dashboards > Home

Welcome to Grafana

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND D...
Grafana fundame...
Set up and understand... experience. This tutori... and covers the "Data s... right.

localhost:9090/query

Prometheus

Query

Alerts

Status

> Enter expression (press Shift+Enter for newlines)

Table Graph Explain

Evaluation time

No data queried yet

+ Add query

localhost:8086/orgs/bdb1c7807daeb978

influxdb

admin keysight

Load Data

Data Explorer

Notebooks

Dashboards

Tasks

Alerts

Settings

View more

Get Started

Write and query data using the programming language of your choice

Python

Node.js

Go

Arduino

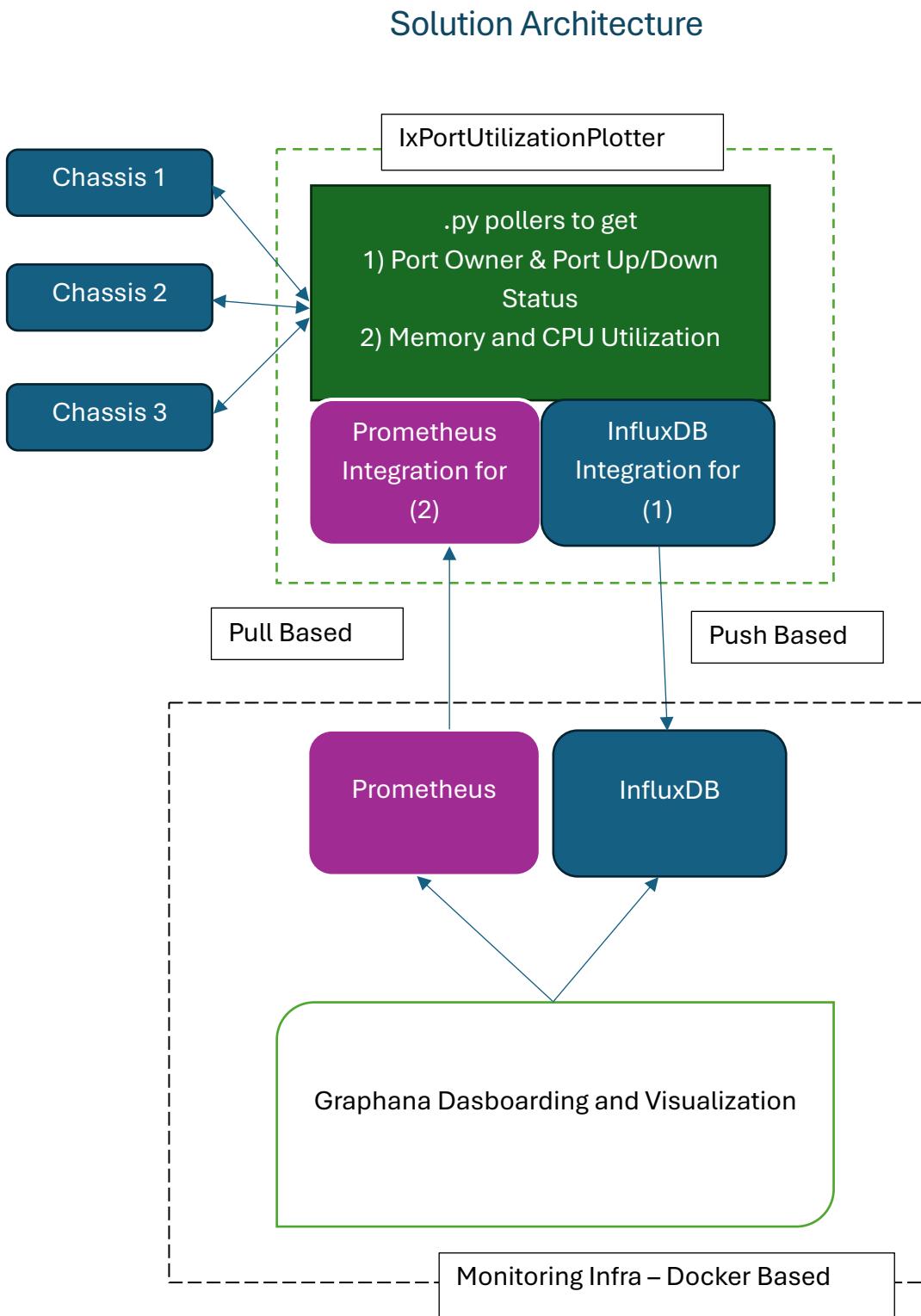
InfluxDB CLI

Write and query data using the InfluxDB Command Line Interface. Supports CSV and Line Protocol.

Server Agent (Telegraf)

Easily collect and write data using custom stand-alone agent plugins

4) Metrics Collection: We will be taking Hybrid approach here, using Prometheus (for numeric metrics) and InfluxDB (for string-based metrics)



Feature	Prometheus	InfluxDB
Data model	Metrics: numeric, time-series only	Time-series with both numeric and string ("tag") support
Data source style	Pull-based (scrapes targets)	Push-based (you write data to it)
Typical use	System & application metrics (CPU, latency, errors, etc.)	Sensor data, IoT, logs, or custom app data (string or mixed types)
Query language	PromQL	InfluxQL / Flux
Storage type	Ephemeral (not long-term)	Long-term (retention policies, downsampling, etc.)

Let's get stated with Metrics Collection:

Step 1: Create the Automation TOKEN for you InfluxDB. You need this to push the metric from python code into Influx DB. Prometheus needs no auth.

Once you get the TOKEN add it to “<path>/lxPortUtilizationPlotter/config.py”

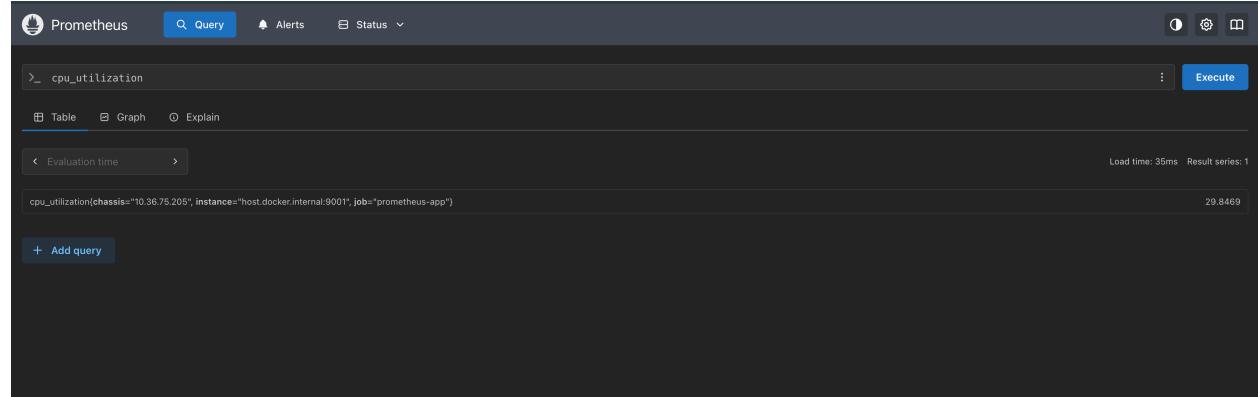
```
61     # InfluxDB URL (use 'influxdb' hostname in Docker, 'localhost' outside Docker)
62     INFLUXDB_URL = os.getenv(
63         'INFLUXDB_URL',
64         'http://localhost:8086'
65     )
66
67     # InfluxDB API Token (required for authentication)
68     INFLUXDB_TOKEN = os.getenv(
69         'INFLUXDB_TOKEN',
70         'eegHpR9kkxg5KG7rklj2zQI86-5z7yNETx0P0qQpSnw1owDxSL5IF-uQru0P-J8M_xmrhT3KWECh-QGbsdyYA=='
71     )
72
```

Step 1: Start to pollers

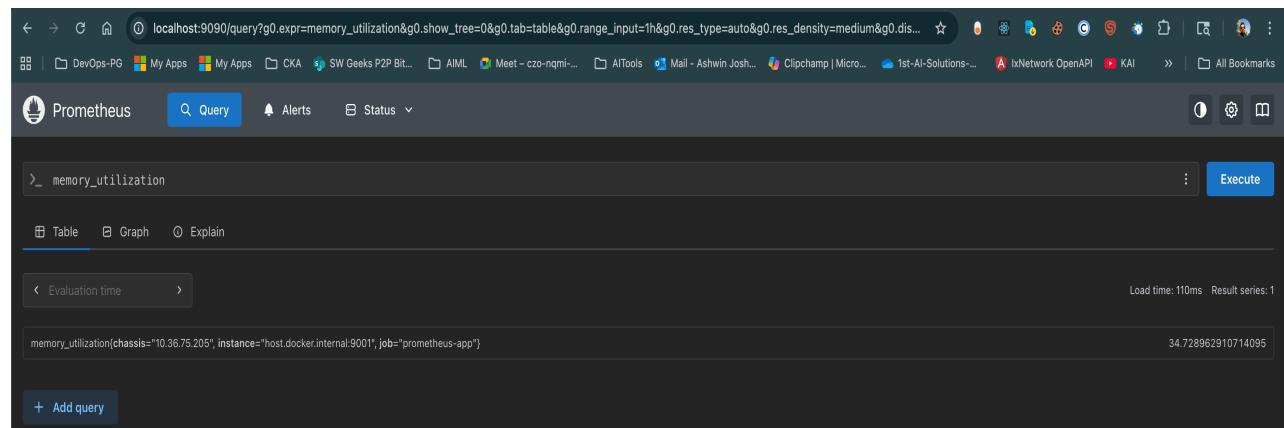
```
run_pollers.sh
```

Let's confirm that data is being populated by querying prometheus , influxDB

In Prometheus:



The screenshot shows the Prometheus web interface. The top navigation bar includes links for 'Query', 'Alerts', and 'Status'. Below the navigation, a search bar contains the query 'cpu_utilization'. Underneath the search bar, there are three tabs: 'Table' (selected), 'Graph', and 'Explain'. A timestamp selector shows 'Evaluation time'. The main content area displays a single result row: 'cpu_utilization(chassis="10.36.75.205", instance="host.docker.internal:9001", job="prometheus-app")' with a value of '29.8469'. At the bottom left is a '+ Add query' button, and at the bottom right are 'Execute' and settings icons.



The screenshot shows the Prometheus web interface. The top navigation bar includes links for 'Query', 'Alerts', and 'Status'. Below the navigation, a search bar contains the query 'memory_utilization'. Underneath the search bar, there are three tabs: 'Table' (selected), 'Graph', and 'Explain'. A timestamp selector shows 'Evaluation time'. The main content area displays a single result row: 'memory_utilization(chassis="10.36.75.205", instance="host.docker.internal:9001", job="prometheus-app")' with a value of '34.728962910714095'. At the bottom left is a '+ Add query' button, and at the bottom right are 'Execute' and settings icons. The browser's address bar shows the URL 'localhost:9090/query?g0.expr=memory_utilization&g0.show_tree=0&g0.tab=table&g0.range_input=1h&g0.res_type=auto&g0.res_density=medium&g0.dis...'.

In InfluxDB

The screenshot shows a "Data Explorer" interface with a dark theme. At the top, there are buttons for "Simple Table" and "CUSTOMIZE", and dropdowns for "Local" and "SAVE AS". The main area displays a table with the following data:

table	_measurement	_field	_value	_time	card	chassis	port
unique	group	group	no group	dateTime:RFC3339	group	group	group
0	portUtilization	portNumber	1	2025-11-03T18:22:59.130Z	1	10.36.75.205	1
1	portUtilization	portNumber	2	2025-11-03T18:22:59.352Z	1	10.36.75.205	2
2	portUtilization	portNumber	3	2025-11-03T18:22:59.359Z	1	10.36.75.205	3
3	portUtilization	portNumber	4	2025-11-03T18:22:59.368Z	1	10.36.75.205	4

5) Visualization of collected metrics into Graphana:

Now, that we see data is populated, let's connect these metric collectors to Graphana. Since all the docker containers are running locally we can use

host.docker.internal OR container_name.

1) InfluxDB (with container name)

The screenshot shows the "InfluxDB-IxOS" data source configuration in Grafana. The "Type" is set to "InfluxDB" and is marked as "Supported". The "Settings" tab is active. The configuration includes:

- Name:** InfluxDB-IxOS
- Query language:** Flux
- HTTP:**
 - URL:** http://Influxdb:8086
 - Allowed cookies:** A note stating "Grafana proxy deletes forwarded cookies by default. Specify cookies by name that should be forwarded to the data source." followed by a "New tag (enter key to add)" input field and an "Add" button.
 - Timeout:** HTTP request timeout in seconds

OR **host.docker.internal**. BOTH WORK WELL

Home > Connections > Data sources > InfluxDB-IxOS

Type: InfluxDB

Settings

Name: InfluxDB-IxOS | Default |

Query language

Flux

HTTP

URL
Your access method is **Server**, this means the URL needs to be accessible from the grafana backend/server.

Allowed cookies
Grafana proxy deletes forwarded cookies by default. Specify cookies by name that should be forwarded to the data source.
New tag (enter key to add) Add

Timeout
HTTP request timeout in seconds

Auth

Basic auth
TLS Client Auth
Skip TLS Verify
Forward OAuth Identity

Custom HTTP Headers
+ Add header

InfluxDB Details

Organization	keysight
Token	configured
Default Bucket	ixosChassisStatistics
Min time interval	10s
Max series	1000

✓ datasource is working. 3 buckets found

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the Explore view.

Delete **Save & test**

✓ datasource is working. 3 buckets found

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the Explore view.

2) Prometheus

The screenshot shows the Grafana interface for configuring a Prometheus data source. The top navigation bar includes 'Home', 'Connections', 'Data sources', 'Prometheus', 'Search...', and various dashboard and alerting options. The main area is titled 'Prometheus' and shows the configuration for a 'Prometheus' type data source named 'Prometheus'. A prominent message box at the top says: 'Configure your Prometheus data source below' and 'Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the [free-forever Grafana Cloud plan](#)'. Below this, there's a 'Name' input field set to 'Prometheus', a 'Default' toggle switch, and a note about required configuration. A message at the bottom states: 'Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#)'. Fields marked with * are required.

The screenshot shows the 'Authentication' configuration section. It starts with a heading 'Authentication methods' and a note: 'Choose an authentication method to access the data source'. A dropdown menu labeled 'Authentication method' contains the option 'No Authentication'. Below this, the 'TLS settings' section is shown with a note: 'Additional security measures that can be applied on top of authentication'. Three checkboxes are available: 'Add self-signed certificate', 'TLS Client Authentication', and 'Skip TLS certificate validation'. At the bottom, the 'HTTP headers' section is visible with a note: 'Pass along additional context and metadata about the request/response'.

The screenshot shows the configuration page for a Prometheus data source in Grafana. The configuration includes:

- Prometheus type: Prometheus
- Prometheus version: Please select
- Cache level: Low
- Incremental querying (beta): Enabled
- Disable recording rules (beta): Enabled
- Other settings:
 - Custom query parameters: Example: max_source_resolution=5m&tin
 - HTTP method: POST
 - Series limit: 40000
 - Use series endpoint: Enabled
- Exemplars: A button to '+ Add'
- A success message: ✓ Successfully queried the Prometheus API.
- A note: Next, you can start to visualize data by [building a dashboard](#), or by querying data in the Explore view.
- A link: Open in Metrics Drilldown
- Action buttons at the bottom: Delete (pink), Save & test (blue)

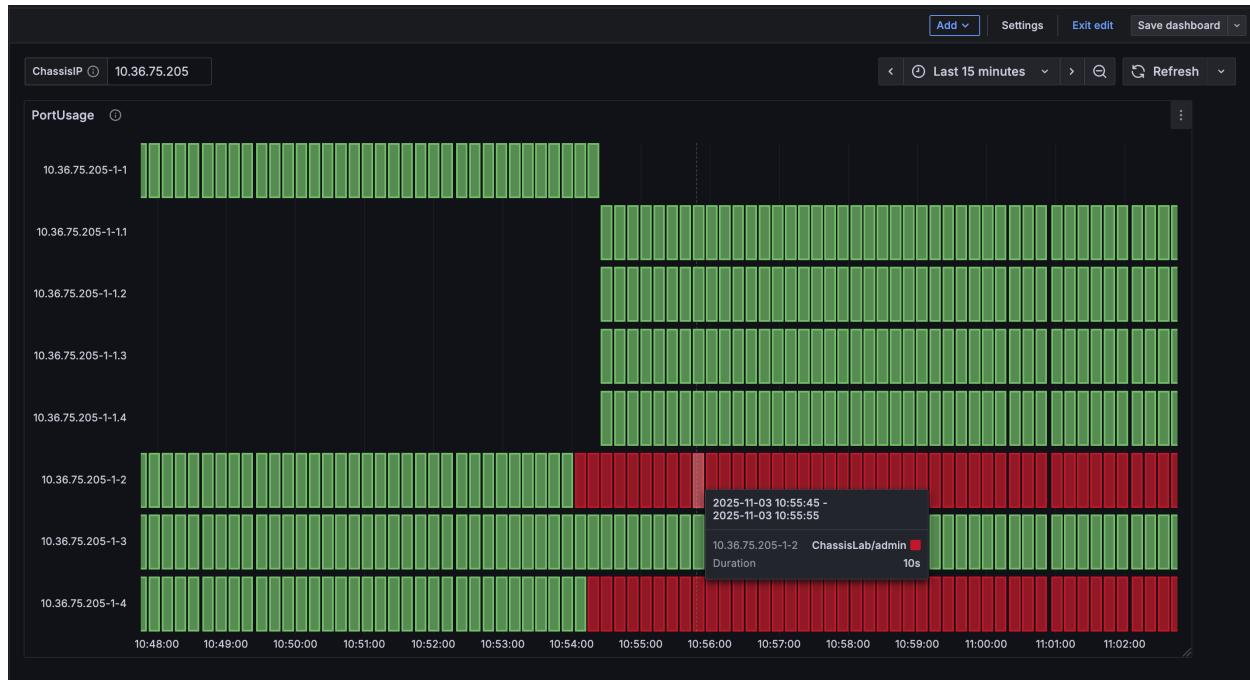
OK. So now that both monitoring sources are connected let's get to charting.

Port Usage Visualization Panel:

- We create a Dashboard and add visualization to it.
- Then we create a Dashboard Variable for ChassisIP

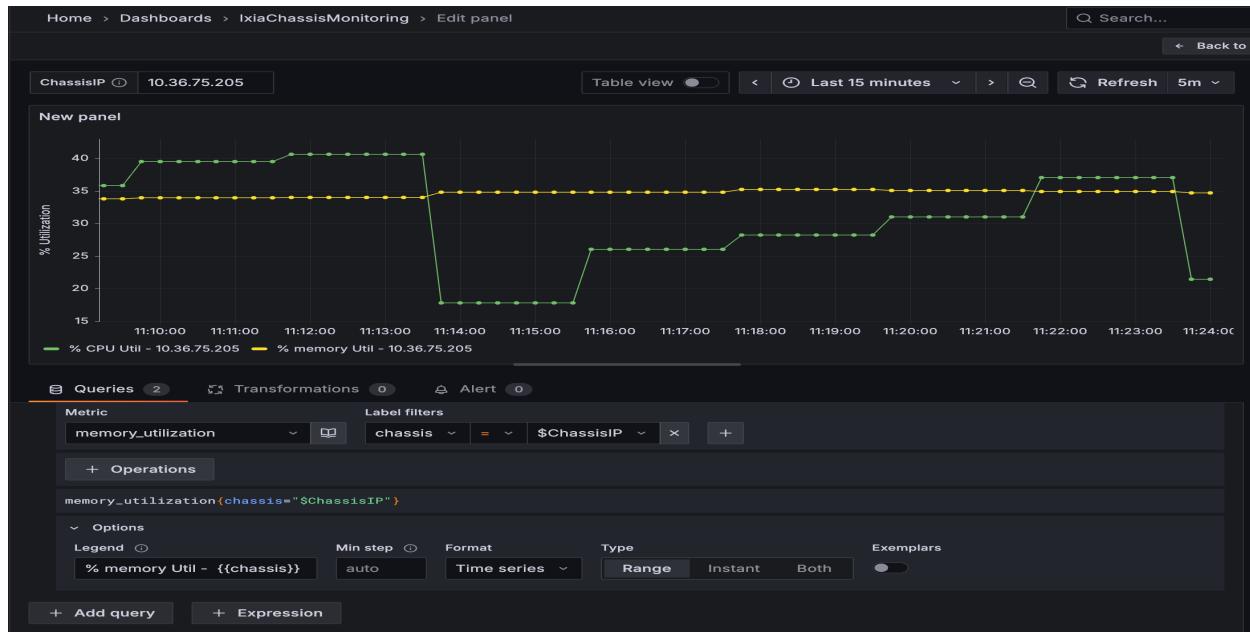
> Example Query for Ownership Visualization

```
from(bucket: "ixosChassisStatistics")  
|> range(start: -24h)  
|> filter(fn: (r) => r["_measurement"] == "portUtilization")  
|> filter(fn: (r) => r["chassis"] == "${ChassisIP}")  
|> filter(fn: (r) => r["_field"] == "owner")
```



We add few more panels to view **PortTransmit State** and **Port UP/DOWN state**

Next, we will visualize performance metrics data coming from Prometheus. You can user Query Builder or the drop down query maker is good enough



Final Dashboard:

