

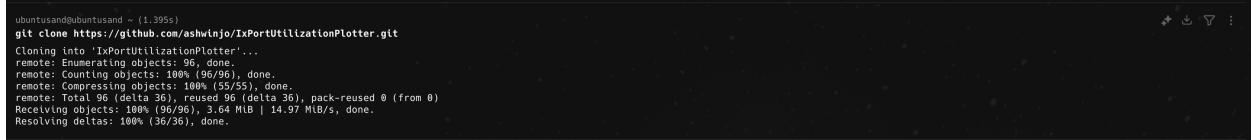
IxOS Monitoring Solution.

Deploying the monitoring solution.

1) Clone / Download IxPortUtilizationPlotter locally

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

```
cd IxPortUtilizationPlotter
```



```
ubuntusandubuntusand ~ (1.395s)
git clone https://github.com/ashwinjo/IxPortUtilizationPlotter.git
Cloning into 'IxPortUtilizationPlotter'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (85/85), done.
remote: Writing objects: 100% (96/96), done.
Receiving objects: 100% (96/96), 3.64 MiB | 14.97 MiB/s, done.
Resolving deltas: 100% (36/36), done.
```

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   INFLUXDB_PORT=8086
6   PROMETHEUS_PORT=9090
7   GRAFANA_PORT=3000
8
9   # InfluxDB Configuration
10  INFLUXDB_ADMIN_USER=admin
11  INFLUXDB_ADMIN_PASSWORD=admin
12  INFLUXDB_ORG=keyshift
13  INFLUXDB_BUCKET=ixosChassisStatistics
14  INFLUXDB_TOKEN=your-super-secret-token-change-me
15  INFLUXDB_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
25   # The poller runs on your HOST machine (not in Docker)
26   # These variables are read by config.py if set
27
28   # If not set, config.py uses its default values
29
30   # InfluxDB Connection URL for poller
31   # Use localhost if services run on same machine, or remote host IP
32   INFLUXDB_URL=http://localhost:8086
33
34
35   # Polling interval in seconds - This is for my influxDB to select metrics push intervals
36   POLLING_INTERVAL=10
37   |  %L to chat, %K to generate

```

> Setup a strong password for influxDB(**passwords must be between 8 and 72 characters long**)

> Change the ports for the containers if there is conflict

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

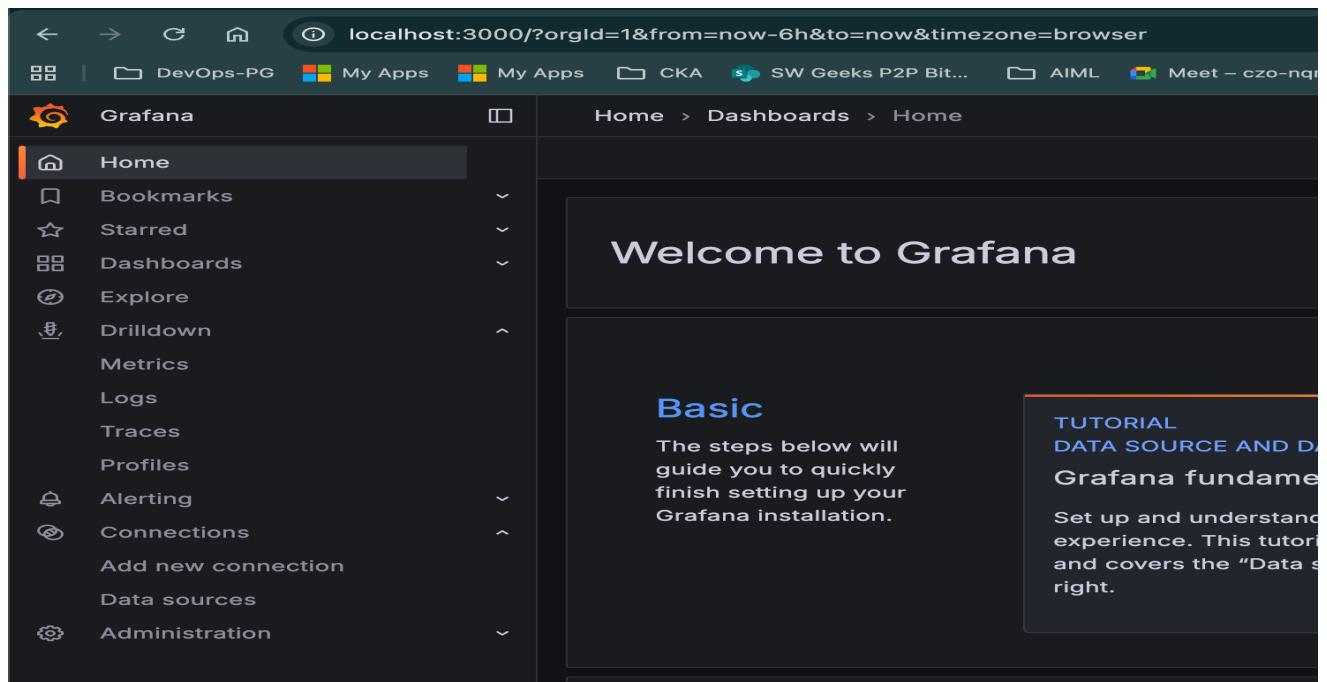
> docker compose up -d

```

ubuntu@ubuntusand:~/IxPortUtilizationPlotter$ git:(main) (8.327s)
docker compose up -d
WARN[0009] /home/ubuntu/and/IxPortUtilizationPlotter/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
WARN[0009] Found orphan containers ([ixportutilizationplotter-db-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clear them up.
[*] Running 3/3
✓ Container prometheus  Healthy
✓ Container influxdb  Healthy
✓ Container grafana  Started
          2.5s      7.5s      0.7s

```

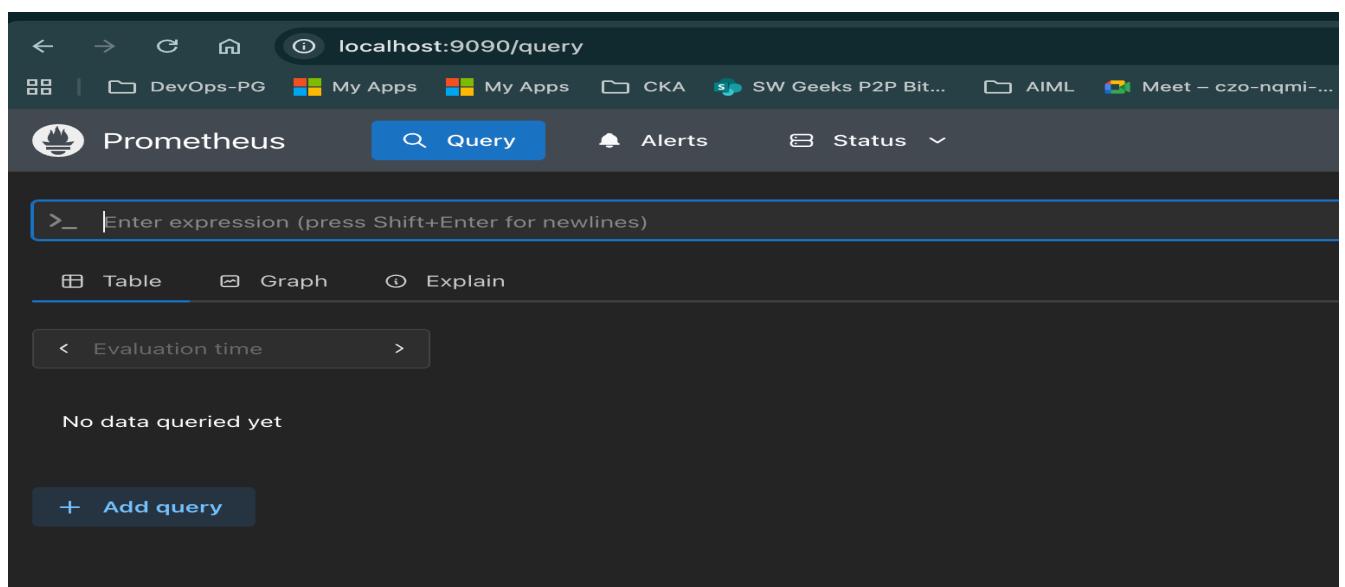
4) Confirm that containers started OK by logging in these portals (Note the UN/PW you set in .env OR then the defaults configured)



The screenshot shows the Grafana interface at localhost:3000/?orgId=1&from=now-6h&to=now&timezone=browser. The left sidebar is titled "Grafana" and contains the following navigation items:

- Home (selected)
- Bookmarks
- Starred
- Dashboards
- Explore
- Drilldown
- Metrics
- Logs
- Traces
- Profiles
- Alerting
- Connections
 - Add new connection
 - Data sources
- Administration

The main content area displays a "Welcome to Grafana" message and a "Basic" setup guide. A sidebar on the right provides links to "TUTORIAL", "DATA SOURCE AND D...", "Grafana fundame...", and a detailed "Set up and understand..." guide.



The screenshot shows the Prometheus interface at localhost:9090/query. The top navigation bar includes links for DevOps-PG, My Apps, CKA, SW Geeks P2P Bit..., AIML, and Meet. The main area is titled "Prometheus" and features a "Query" input field with placeholder text "Enter expression (press Shift+Enter for newlines)". Below the input field are three visualization options: "Table" (selected), "Graph", and "Explain". A "Status" dropdown menu is also present. A "Evaluation time" selector shows a range from "Evaluation time" to "Now". The central message states "No data queried yet". At the bottom, there is a "Add query" button.

The screenshot shows the InfluxDB web interface at localhost:8086/orgs/dbb1c7807daeb978. The left sidebar has a dark blue theme with the following navigation items:

- influxdb
- k admin keysight
- Load Data
- Data Explorer
- Notebooks
- Dashboards
- Tasks
- Alerts
- Settings

The main content area is titled "Get Started" and features the text "Write and query data using the programming language of your choice". It displays four programming language logos in a grid:

- Python: Python logo icon
- Node.js: Node.js logo icon
- Go: Go logo icon
- Arduino: Arduino logo icon

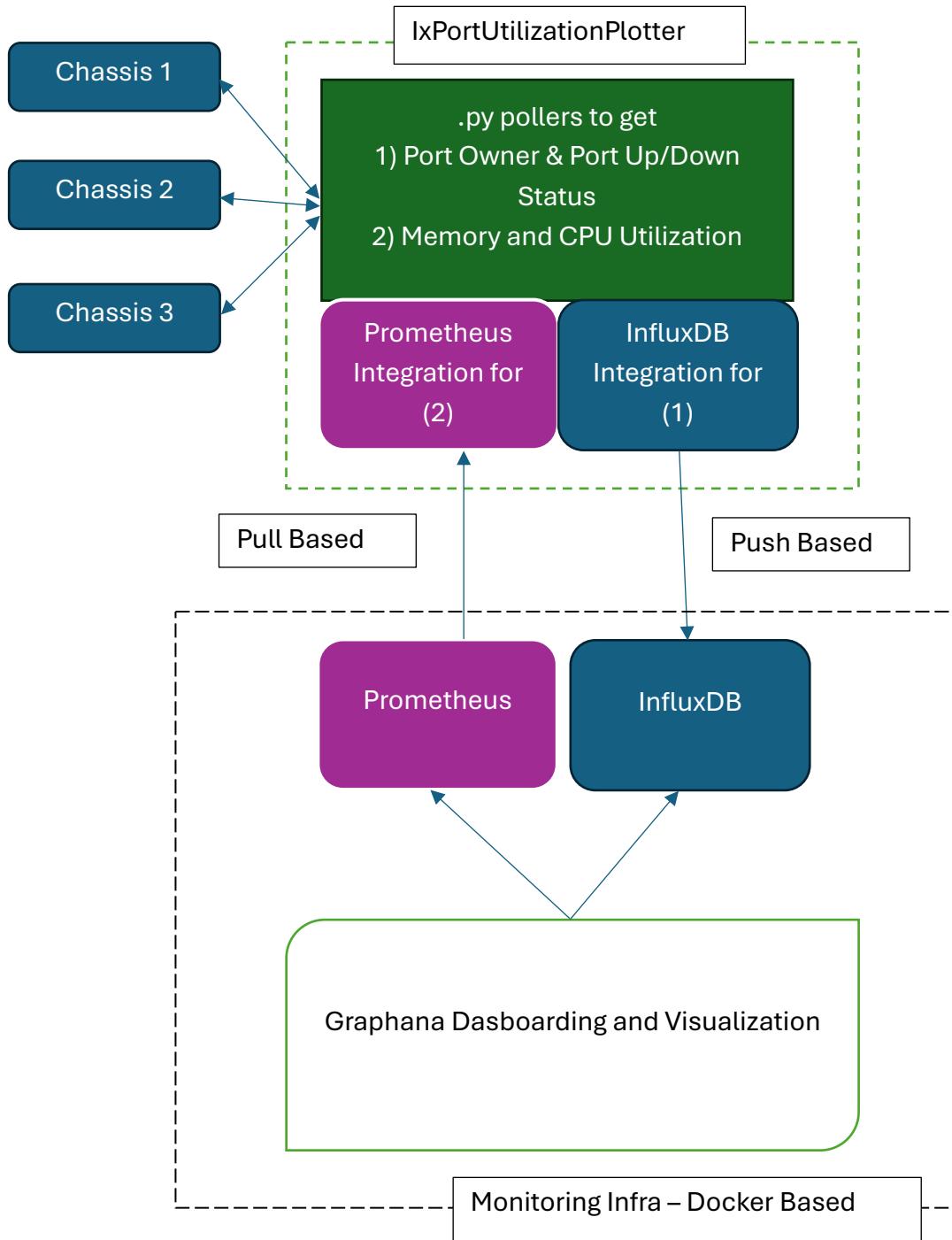
Below this, there are two additional sections with arrows pointing right:

- 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.

At the bottom left of the main content area is a "View more" link.

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

Solution Architecture



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.

Manage tokens in the InfluxDB UI

To manage InfluxDB API Tokens in the InfluxDB UI, navigate to the [API Tokens](#) management page.

In the navigation menu on the left, select **Load Data > API Tokens**.



Create a token in the InfluxDB UI

Create an All Access token

1. From the [API Tokens management page](#), click the **+ GENERATE API TOKEN** button.
2. Select **All Access API Token**.

Create a custom token

1. From the [API Tokens management page](#), click the **+ GENERATE API TOKEN** button.
2. Select **Custom API Token**.
3. When the **Generate a Personal API Token** window appears, enter a description. If you don't provide a description for the token, InfluxDB will generate a description from the permissions you assign. For example, if you select **Read** for a bucket named "`_monitoring`" and **Write** for a bucket named "`_tasks`", InfluxDB will generate the description "`Read buckets _monitoring Write buckets _tasks`".
4. Select the check boxes in the **Read** and **Write** columns to assign access permissions for the token. You can enable access to all buckets, individual buckets, Telegraf configurations, and other InfluxDB resources. By default, the new token has no access permissions.
5. When you're finished, click **GENERATE**.
6. When InfluxDB displays the token value, click **COPY TO CLIPBOARD**. This is your only chance to access and copy the token value from InfluxDB.

A screenshot of the InfluxDB UI. At the top, there is a header with the title "Load Data" and a navigation bar with links: SOURCES, BUCKETS, TELEGRAPH, SCRAPERS, and API TOKENS. Below the navigation bar, there is a search bar with the placeholder "Filter Tokens..." and a dropdown menu labeled "Sort by Description (A - Z)". On the right side of the header, there is a button labeled "+ GENERATE API TOKEN". The main content area shows a list of tokens. One token, "admin's Token", is highlighted with a yellow border. It has a small info icon next to it. Below the token list, a modal window titled "Generate All Access API Token" is open. The modal contains a warning message: "This token will be able to create, update, delete, read, and write to anything in this organization". There is also a "Description" input field containing the text "kslabmachinekey". At the bottom of the modal, there are two buttons: "CANCEL" and "SAVE".

Load Data

SOURCES BUCKETS TELEGRAF SCRAPERS API TOKENS

Filter Tokens...

Sort by Description (A → Z) ▾

admin's Token

Created at: 2025-11-09 11:32:41 Owner: admin Last Modified: 14 minutes ago

kslabmachinekey

Created at: 2025-11-09 11:46:47 Owner: admin Last Modified: 0 seconds ago

Once you get the API TOKEN add it in the .env file

```
# InfluxDB Configuration
INFLUXDB_ADMIN_USER=admin
INFLUXDB_ADMIN_PASSWORD=Keysight12345!
INFLUXDB_ORG=keysight
INFLUXDB_BUCKET=ixosChassisStatistics
INFLUXDB_TOKEN='eegHpR9kkgxg5KG7rklj2zQI86-5z7yNETx0P0qQpSnw1owDxSL5IF-uQru0P-J8M_xmrhT3KWECh-QGbsdyYA=='
INFLUXDB_RETENTION=0 # 0 = infinite retention, or specify in seconds
```

Add the chassis list in .env file with IP , Username and Password to monitor

```
CHASSIS_LIST = [{"ip": "10.36.75.205", "username": "admin", "password": "admin"}]
```

Also, add the polling intervals

```
# Polling interval in seconds - This is for my influxDB to select metrics push intervals
POLLING_INTERVAL=30
# Polling interval in seconds - This is for my prometheus to select metrics push intervals
POLLING_INTERVAL_PERF_METRICS=30
POLLING_INTERVAL_SENSOR_METRICS=30
```

Finally, modify the @prometheus.yaml with correct scrape interval and target IP. Targets are the IP on machine on which you will be running the Pollers.

```
GNU nano 1.2
global:
  scrape_interval: 30s

scrape_configs:
  - job_name: prometheus-app
    static_configs:
      - targets:
          - 10.36.237.138:9001
  - job_name: sensors-app
    static_configs:
      - targets:
          - 10.36.237.138:9002
```

Check the Prometheus targets to ensure that metrics targets configured ok.

Target	Endpoint	Labels	Last scrape	State
prometheus-app	http://10.36.237.138:9001/metrics	instance="10.36.237.138:9001" job="prometheus-app"	3.982s ago	UP
sensors-app	http://10.36.237.138:9002/metrics	instance="10.36.237.138:9002" job="sensors-app"	19.292s ago	UP

Showtime !!!!!

Step 1: Start all 3 pollers

```
chmod +x run_pollers.sh stop_pollers.sh  
./run_pollers.sh
```

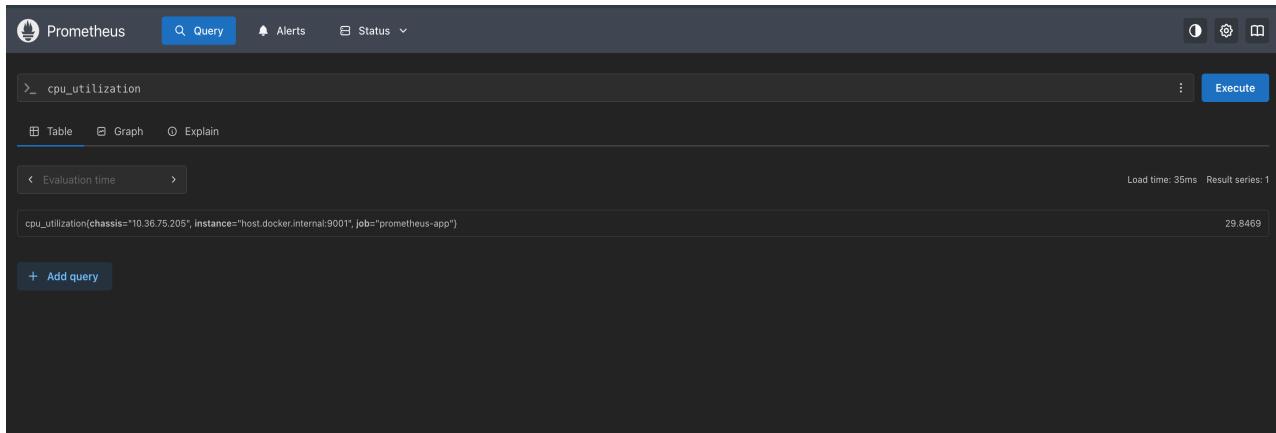
```
Starting IxOS Pollers...  
=====  
✓ Started portInfoPoller.py (PID: 3779636)  
✓ Started perfMetricsPoller.py (PID: 3779638)  
✓ Started sensorsPoller.py (PID: 3779640)  
  
View logs:  
tail -f ./logs/portInfoPoller.log  
tail -f ./logs/perfMetricsPoller.log  
tail -f ./logs/sensorsPoller.log  
  
Stop: sh ./stop_pollers.sh  
  
Note: Virtual environment 'ixmon' is activated in this script.  
The pollers will run with the dependencies from this environment.
```

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

Wait for the

> scrape interval as per @prometheus.yaml for CPU/Mem utilization & Sensor Data
> POLLING_INTERVAL_PERF_METRICS=30 for PortUsage Metrics

In Prometheus:



The screenshot shows the Prometheus web interface. At the top, there are tabs for Prometheus, Query, Alerts, and Status. The Query tab is active, showing a search bar with the query `>_ cpu_utilization`. Below the search bar are buttons for Table, Graph, Explain, and Execute. The Table button is selected. A dropdown menu for Evaluation time is open, showing arrows to navigate between time points. To the right, it says "Load time: 35ms Result series: 1". The results table has one row with the following data:

cpu_utilization(chassis="10.36.75.205", instance="host.docker.internal:9001", job="prometheus-app")	29.8469

At the bottom left is a "+ Add query" button.

The screenshot shows the Prometheus web interface at localhost:9090. The URL bar shows the 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.discriminator=`. The page title is "Prometheus". The main area contains a query input field with the text `>_ memory_utilization`, a "Table" tab selected, and an "Execute" button. Below the table, it says "Evaluation time" and "Load time: 110ms Result series: 1". The results table shows one row of data:

<code>memory_utilization(chassis="10.36.75.205", instance="host.docker.internal:9001", job="prometheus-app")</code>	34.728962910714095
---------------------------------------------------------------------------------------------------------------------	--------------------

At the bottom left is a "+ Add query" button.

In InfluxDB

The screenshot shows the InfluxDB Data Explorer. The top navigation bar includes "Data Explorer", "Graph", "CUSTOMIZE", "Local", "SAVE AS", and "Query 1 (0.15s)". The main area displays a table of data with columns: _time, _value, _field, _measurement, card, chassis, and port. The data shows measurements for ownedPorts and portUtilization over four time points. The bottom section shows the query builder with filters for measurement (ixosChassisStatistics), field (portUtilization), and card, along with other filter options like cardNumber, freePorts, linkState, owner, portNumber, totalPorts, and transmitState. The "SCRIPT EDITOR" and "SUBMIT" buttons are visible on the right.

_time	_value	_field	_measurement	card	chassis	port
2025-11-09 11:44:30	5	ownedPorts	portUtilization	1	10.36.75.205	1
2025-11-09 11:44:30	5	ownedPorts	portUtilization	1	10.36.75.205	2
2025-11-09 11:44:30	5	ownedPorts	portUtilization	1	10.36.75.205	3
2025-11-09 11:44:30	5	ownedPorts	portUtilization	1	10.36.75.205	4.1
2025-11-09 11:44:30	5	ownedPorts	portUtilization	1	10.36.75.205	4.2

5) Visualization of collected metrics into Grafana:

Now, that we see data is populated, let's connect these metric collectors to **Grafana**. Since all the docker containers are running locally we will use **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 'Supported' is highlighted. The 'Name' is 'InfluxDB-IxOS' and it is the 'Default'. The 'Query language' is set to 'Flux'. Under the 'HTTP' section, the 'URL' is 'http://influxdb:8086'. There are sections for 'Allowed cookies' (specifying forwarded cookies by name) and 'Timeout' (HTTP request timeout in seconds). A note at the bottom says 'datasource is working. 3 buckets found'.

This screenshot continues the 'InfluxDB Details' configuration. It shows fields for 'Organization' (keyesight), 'Token' (configured), 'Default Bucket' (ixosChassisStatistics), 'Min time interval' (10s), and 'Max series' (1000). At the bottom, there are 'Delete' and 'Save & test' buttons.

* **Note:** Every time you recreate the Grafana container you will need to write and save to the Token field. This is the API TOKEN you created in previous steps.

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 of the page reads: '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 three checkboxes: 'Add self-signed certificate', 'TLS Client Authentication', and 'Skip TLS certificate validation'. At the bottom, the 'HTTP headers' section is visible with the 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 box:
 - ✓ Successfully queried the Prometheus API.
 - Next, you can start to visualize data by [building a dashboard](#), or by querying data in the Explore view.
- Buttons at the bottom: Delete (pink), Save & test (blue)

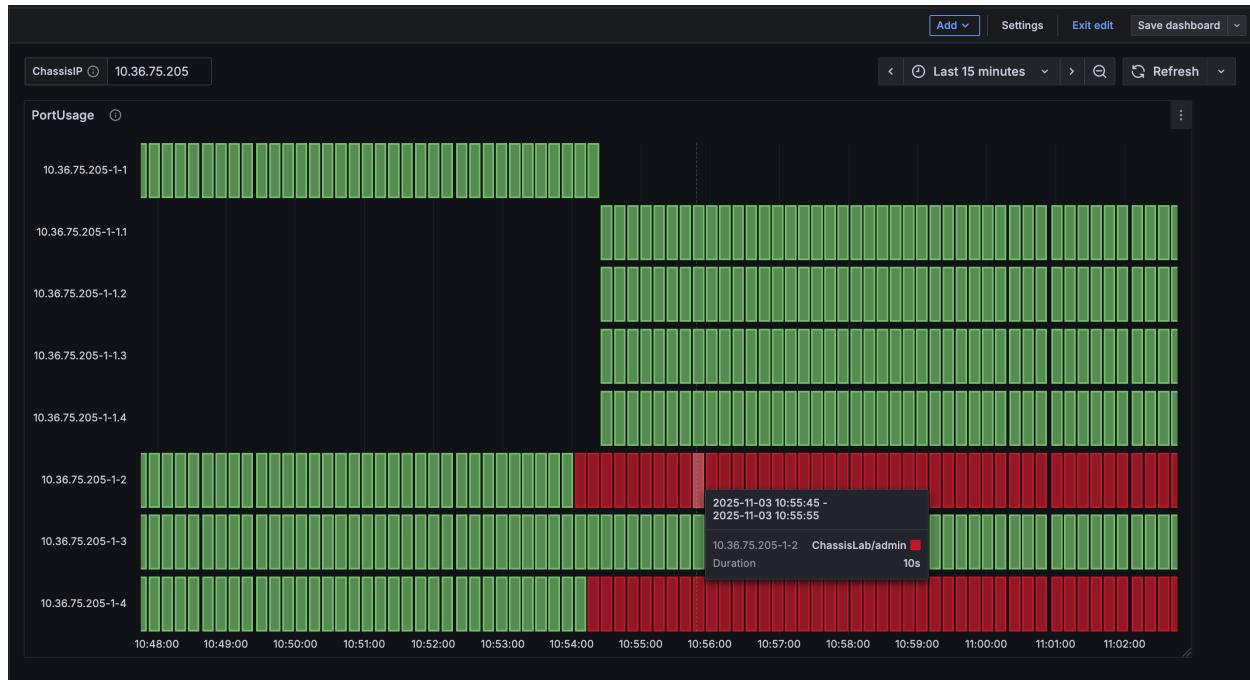
OK. So now that both monitoring sources are connected let's get to charting.

Port Usage Visualization Panel:

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

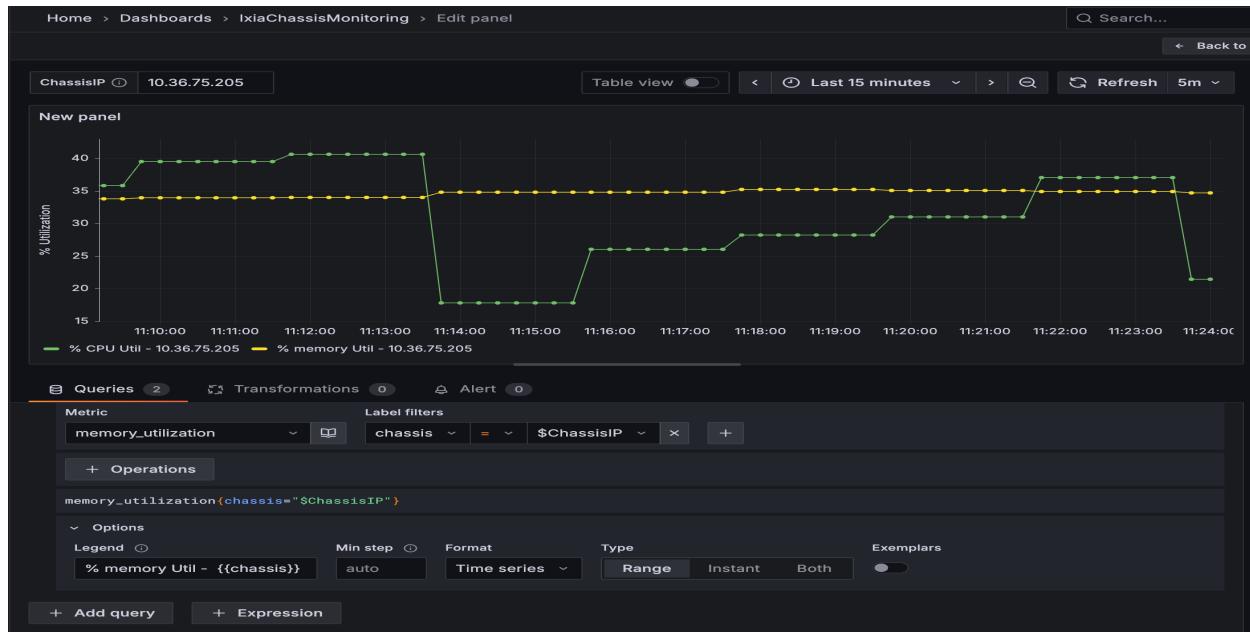
> Example Query for Ownership Visualization

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

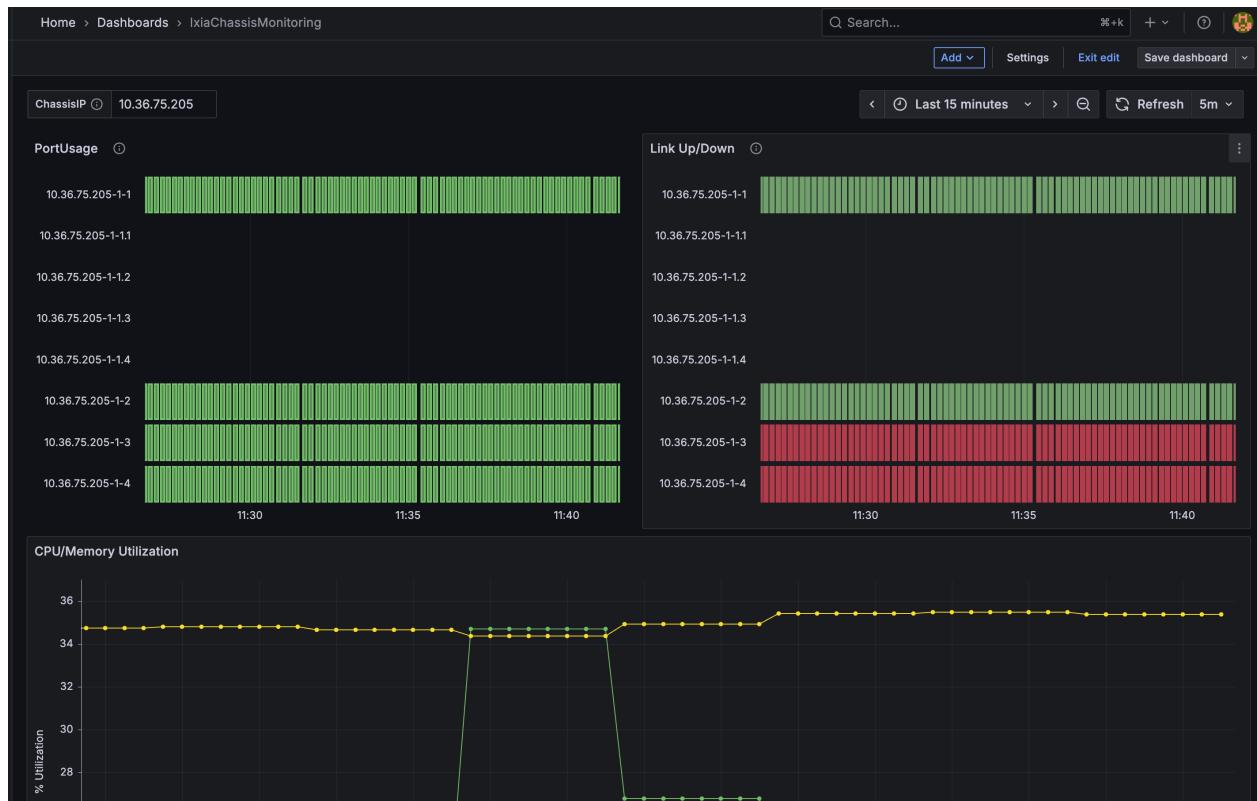


We add few more panels to view **PortTransmit State** and **Port UP/DOWN state**

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:



Sensors Dashboard:



Note: These dashboards are available at. Just go to Dashboards → Import

<https://github.com/ashwinjo/IxPortUtilizationPlotter/blob/main/IxOSSensorData-1762728124980.json>

<https://github.com/ashwinjo/IxPortUtilizationPlotter/blob/main/IxiaChassisMonitoring-1762728398844.json>