Create instance in AWS consul in security group allow 3000 port  
Download :  
<https://grafana.com/grafana/download>  
  
run command:  
wget <https://dl.grafana.com/enterprise/release/grafana-enterprise-12.0.2.linux-amd64.tar.gz>  
tar -zxvf grafana-enterprise-12.0.2.linux-amd64.tar.gz  
  
  
  
  
You’ve **downloaded and extracted the Grafana binary package manually**, which means you **didn’t install it via yum or system package manager**—so systemctl won’t work because there’s no systemd service file registered.

Let’s now **run Grafana manually** and optionally create a service for it.

**✅ *Option 1: Start Grafana Manually***

1. Go to the bin directory:

bash

cd ~/grafana-v12.0.2/bin

1. Start Grafana:

bash

./grafana-server web

This will start Grafana and use the default configuration file located at conf/defaults.ini.

1. Leave this terminal running. Open Grafana in your browser:

cpp

http://<your-ec2-public-ip>:3000

***Option 2: Create a Systemd Service (Optional but Recommended)***

To make it run like a service (you can systemctl start grafana), follow these steps:

**Step 1: Create the user**

bash

sudo useradd --no-create-home --shell /bin/false grafana

**Step 2: Move the Grafana directory to /usr/local**

bash

sudo mv ~/grafana-v12.0.2 /usr/local/grafana

**Step 3: Set ownership**

bash

sudo chown -R grafana:grafana /usr/local/grafana

**Step 4: Create a systemd service file**

bash

sudo nano /etc/systemd/system/grafana.service

Paste the following content:

ini

[Unit]

Description=Grafana service

After=network.target

[Service]

User=grafana

Group=grafana

Type=simple

ExecStart=/usr/local/grafana/bin/grafana-server \

--config=/usr/local/grafana/conf/defaults.ini \

--homepath=/usr/local/grafana

[Install]

WantedBy=multi-user.target

Save and exit (Ctrl+O, Enter, then Ctrl+X)

**Step 5: Reload systemd and start Grafana**

bash

sudo systemctl daemon-reexec

sudo systemctl daemon-reload

sudo systemctl start grafana

sudo systemctl enable grafana

**Step 6: Check Grafana Status**

bash

sudo systemctl status grafana

**✅ Final Step: Access in Browser**

Go to:

cpp

http://<your-ec2-public-ip>:3000

Default login:

* **Username:** admin
* **Password:** admin

**Why create a separate grafana user?**

Creating a dedicated system user like grafana is a **best practice in Linux for running services**. Here's **why it makes sense**:

**✅ 1. Security**

* If you run Grafana as root or even ec2-user, and Grafana gets exploited (like due to a vulnerability), the attacker can gain access to everything that user has.
* A limited user like grafana **can’t modify system files, install software, or damage the system**.
* This limits the blast radius.

**✅ 2. Isolation**

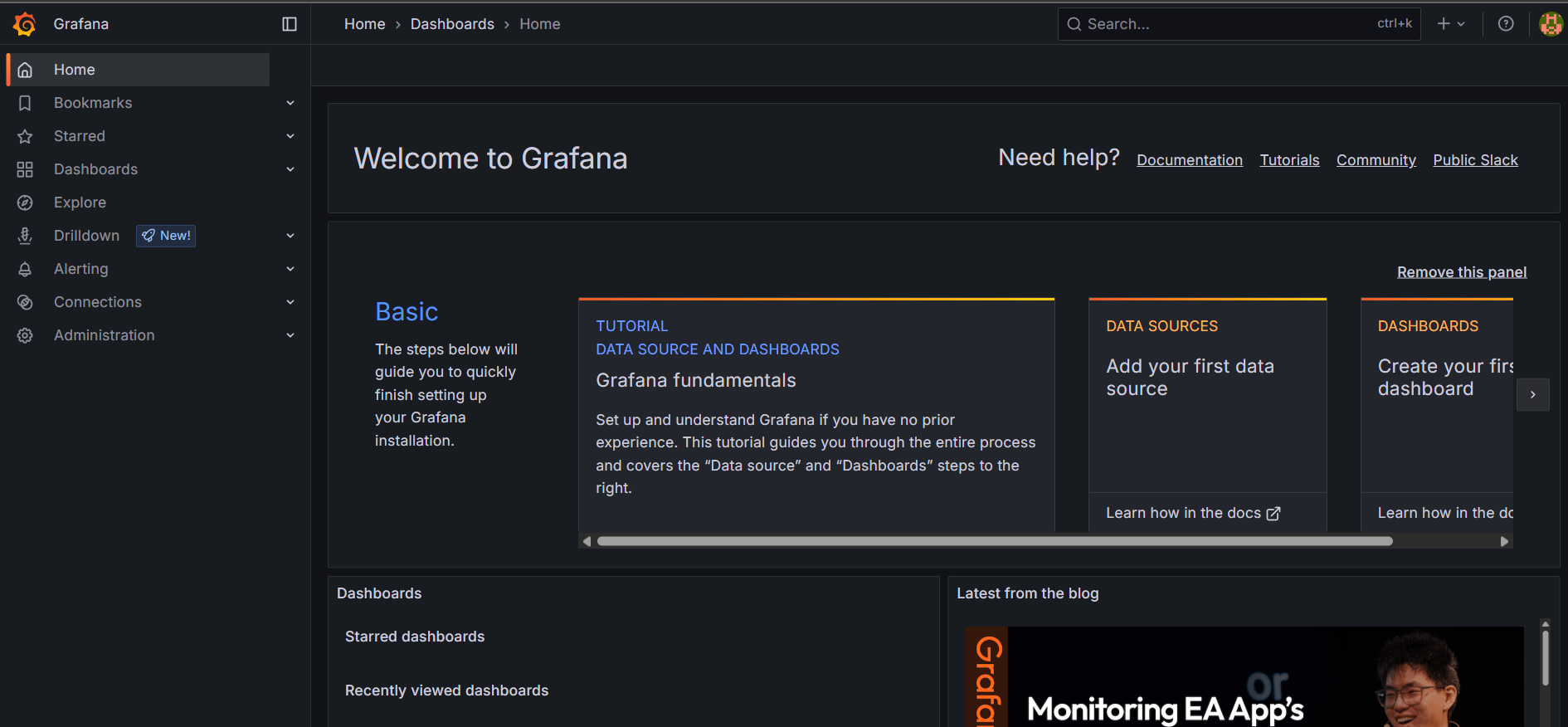
* Grafana files and processes are owned by a dedicated user.
* This keeps them separate from other apps (like NGINX, Prometheus, etc.).
* Makes debugging and permissions management much cleaner.

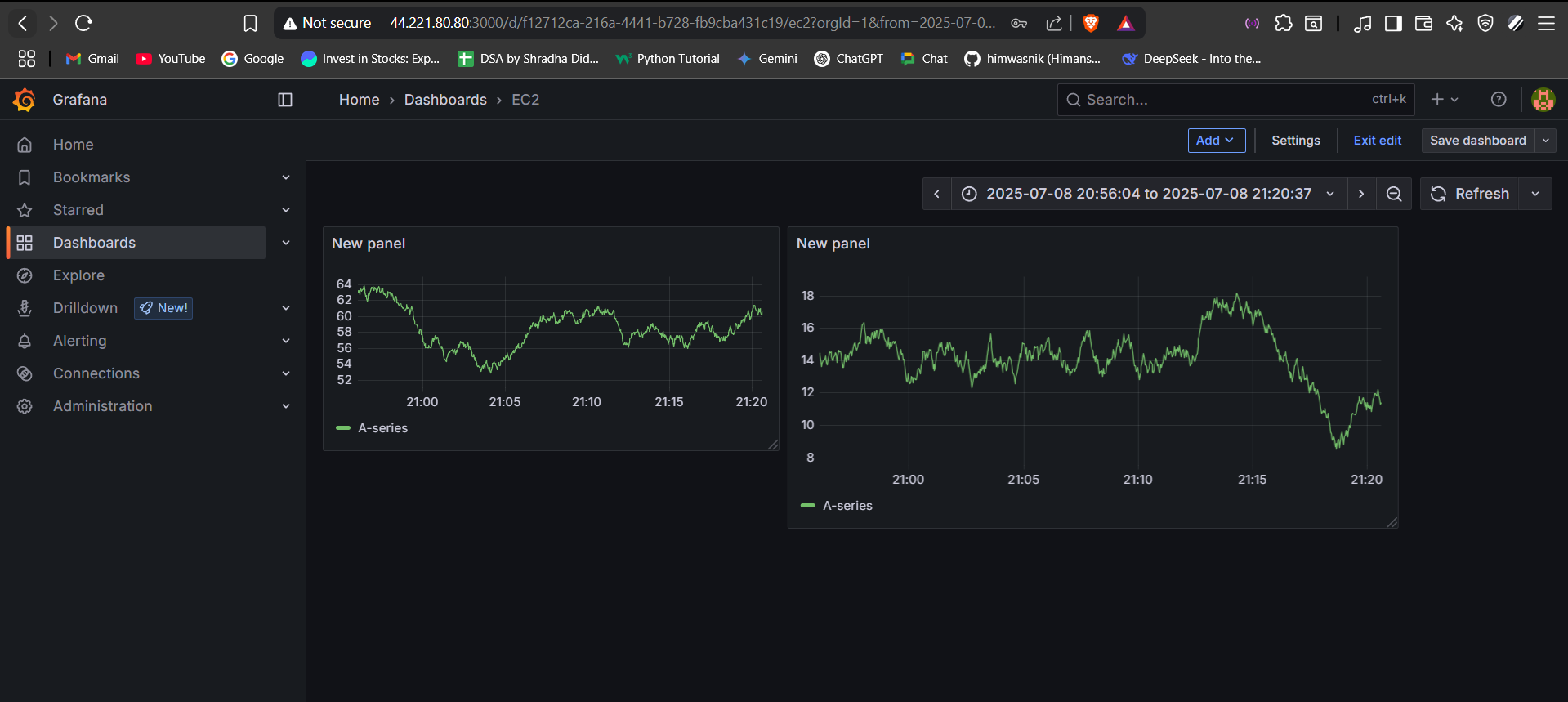
**✅ 3. Systemd expects it**

* Services like nginx, mysql, grafana, etc., are usually configured to run under their own user.
* That’s why in the systemd service file, we specify:

**✅ 4. Auditing and Logging**

* If something goes wrong or suspicious happens, logs will show which user was running it.
* If you run everything as ec2-user, it’s harder to tell what’s what.





***What is InfluxDB?***

**InfluxDB** is a **time-series database** — designed to **store and query time-stamped data**, like:

* Server CPU/memory usage over time
* IoT sensor readings
* Application logs
* DevOps/monitoring metrics

📊 It works **very well with Grafana**, which is used to visualize that time-series data.

**✅ Use Case Example with Grafana:**

You can store system metrics like:

* CPU
* Memory
* Disk
* Network

...in **InfluxDB**, and then connect Grafana to display them in dashboards.

**InfluxDB 2.x Installation on Amazon Linux EC2 (With GPG Issue Workaround)**

**1. Clean yum cache to avoid conflicts**

bash

sudo yum clean all

sudo rm -rf /var/cache/yum

**2. Create the InfluxDB repo file**

bash

sudo tee /etc/yum.repos.d/influxdb.repo <<EOF

[influxdb]

name = InfluxDB Repository - RHEL

baseurl = https://repos.influxdata.com/rhel/7/x86\_64/stable/

enabled = 1

gpgcheck = 1

gpgkey = https://repos.influxdata.com/influxdb.key

EOF

**3. (Optional) Import GPG key manually**

bash

curl -s https://repos.influxdata.com/influxdb.key | sudo tee /etc/pki/rpm-gpg/influxdb.key

sudo rpm --import /etc/pki/rpm-gpg/influxdb.key

**4. Disable GPG check temporarily to bypass the key mismatch error**

Edit the repo file:

bash

sudo nano /etc/yum.repos.d/influxdb.repo

Change:

ini

CopyEdit

gpgcheck = 1

to

ini

gpgcheck = 0

Save and exit.

**5. Install InfluxDB**

bash

sudo yum install influxdb2 -y

**6. Reload systemd daemon to recognize service files**

bash

sudo systemctl daemon-reload

**7. Start and enable InfluxDB service**

bash

sudo systemctl start influxdb

sudo systemctl enable influxdb

**8. Verify InfluxDB service status**

bash

sudo systemctl status influxdb

Expect status: **active (running)**

**9. Access the InfluxDB UI**

Open in your browser:

cpp

http://<your-ec2-public-ip>:8086

Complete initial setup (user/org/bucket creation).

**10. (Optional) Re-enable GPG check for security**

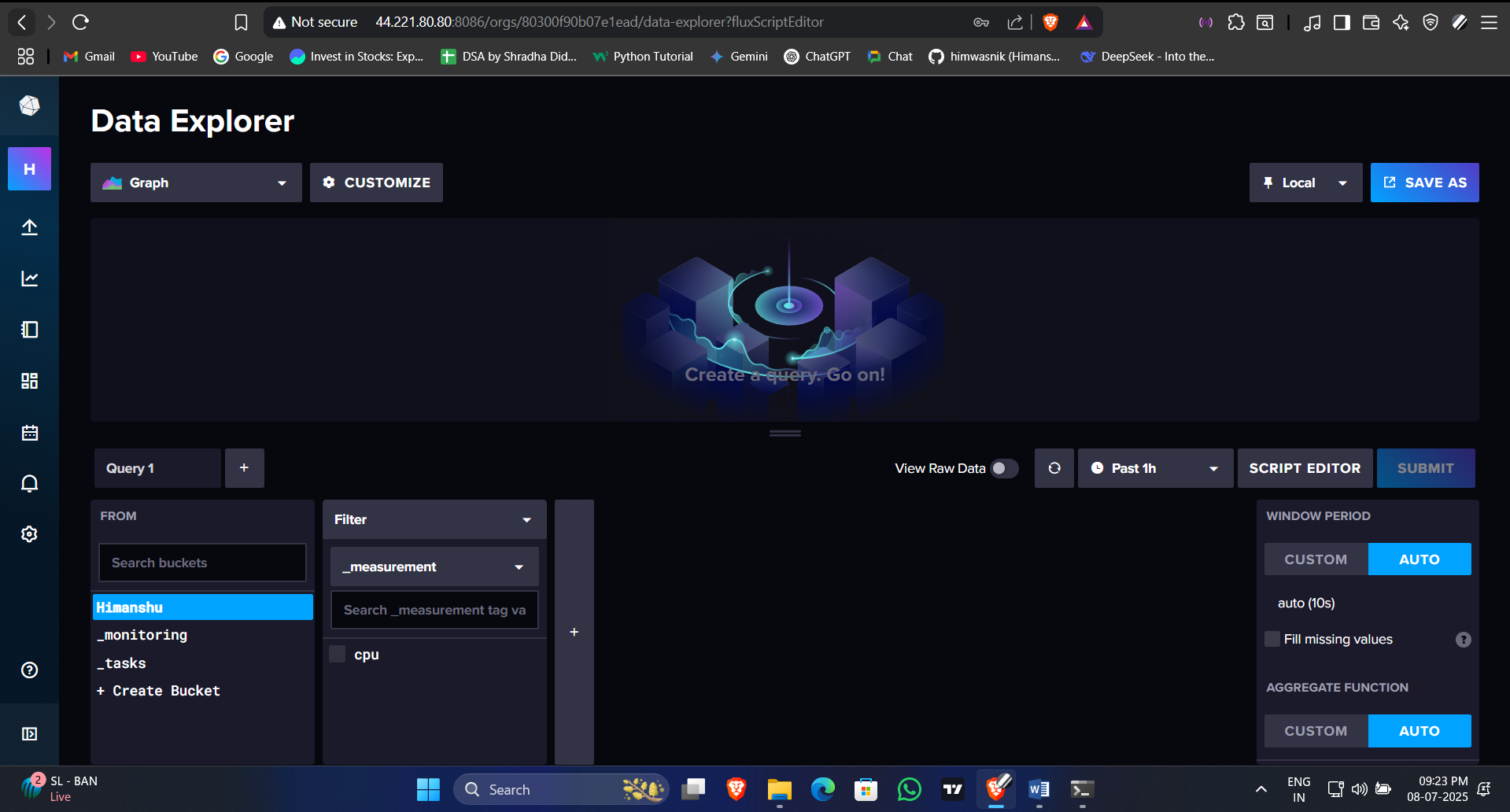
After successful install, edit repo file and set:

ini

gpgcheck = 1

**Notes:**

* The key problem was a GPG key mismatch that prevented package installation.
* Disabling gpgcheck temporarily allowed installation to proceed.
* The influxdb.service file is created by the package and points to the correct binary path.
* Always ensure the systemctl daemon-reload is run after editing or removing service files.

**

***What is Telegraf?***

**Telegraf** is an open-source **server agent** written in Go. It is used for collecting, processing, and sending metrics and events from your system to **databases like InfluxDB**.

**🔧 What Telegraf does:**

* Collects system metrics (CPU, memory, disk, network, etc.).
* Supports hundreds of input plugins (Apache, Nginx, Docker, MySQL, etc.).
* Sends this data to output destinations like **InfluxDB**, Prometheus, Graphite, etc.
* Lightweight and efficient; ideal for monitoring.

**Telegraf Installation & Configuration Steps (Amazon Linux 2023)**

**✅ Prerequisites**

* InfluxDB 2.x must be running and accessible.
* You should have a **bucket**, **org**, and **token** ready from InfluxDB.
* You should be using **Amazon Linux 2023** or compatible.

**🧩 Step 1: Add InfluxData YUM Repository**

bash

sudo tee /etc/yum.repos.d/influxdb.repo <<EOF

[influxdb]

name = InfluxDB Repository - RHEL

baseurl = https://repos.influxdata.com/rhel/7/x86\_64/stable/

enabled = 1

gpgcheck = 1

gpgkey = https://repos.influxdata.com/influxdb.key

EOF

**🔑 Step 2: Import GPG Key**

bash

curl -s https://repos.influxdata.com/influxdb.key | sudo tee /etc/pki/rpm-gpg/influxdb.key

sudo rpm --import /etc/pki/rpm-gpg/influxdb.key

**📦 Step 3: Install Telegraf**

bash

sudo yum install telegraf -y

**⚙️ Step 4: Generate a Sample Config File**

bash

telegraf --sample-config --input-filter cpu --output-filter influxdb\_v2 > telegraf.conf

This config collects only **CPU metrics** and sends them to **InfluxDB v2**.

**📁 Step 5: Move the Config to System Directory**

bash

sudo mv telegraf.conf /etc/telegraf/telegraf.conf

**✏️ Step 6: Edit the Config File**

Edit the output section to match your InfluxDB details:

bash

sudo nano /etc/telegraf/telegraf.conf

Find and modify this section:

toml

[[outputs.influxdb\_v2]]

urls = ["http://localhost:8086"] # Or remote InfluxDB IP

token = "YOUR\_INFLUXDB\_TOKEN"

organization = "YOUR\_ORG\_NAME"

bucket = "YOUR\_BUCKET\_NAME"

Save and exit (Ctrl + O, Enter, then Ctrl + X if using nano).

**🚀 Step 7: Start and Enable Telegraf Service**

bash

sudo systemctl start telegraf

sudo systemctl enable telegraf

**✅ Step 8: Verify Status**

bash

sudo systemctl status telegraf

You should see:

yaml

Active: active (running)

Loaded outputs: influxdb\_v2

Loaded inputs: cpu

**📊 Step 9: Check Metrics in InfluxDB UI**

* Login to InfluxDB web UI
* Go to **Data Explorer**
* Choose your **bucket**
* You should see cpu measurements populating every 10 seconds

**📝 Optional: Add More Input Plugins**

You can add more input plugins like this:

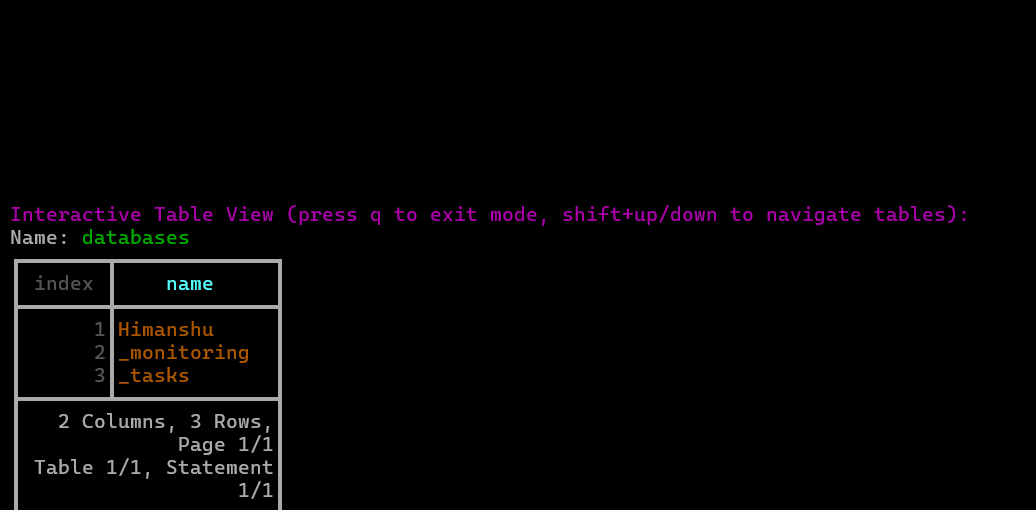
bash

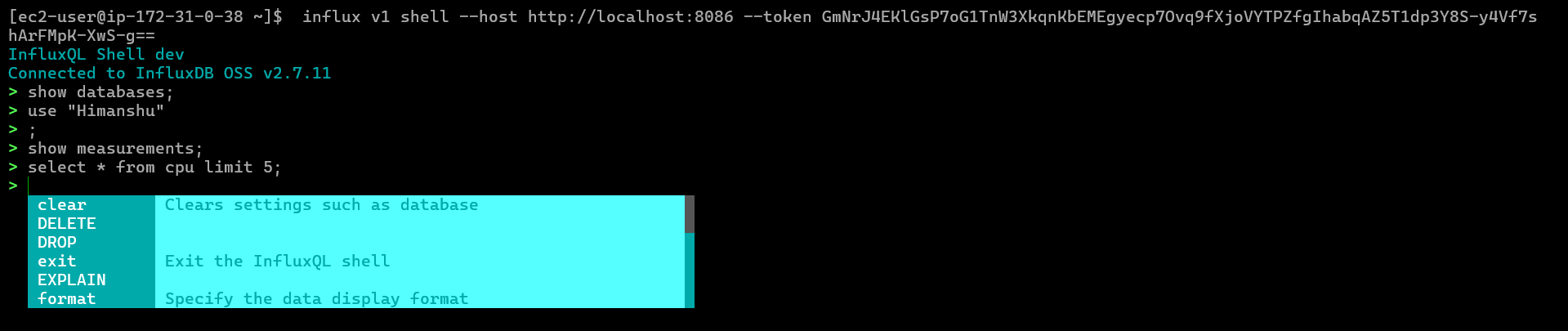
telegraf --sample-config --input-filter cpu:mem:disk --output-filter influxdb\_v2 > telegraf.conf

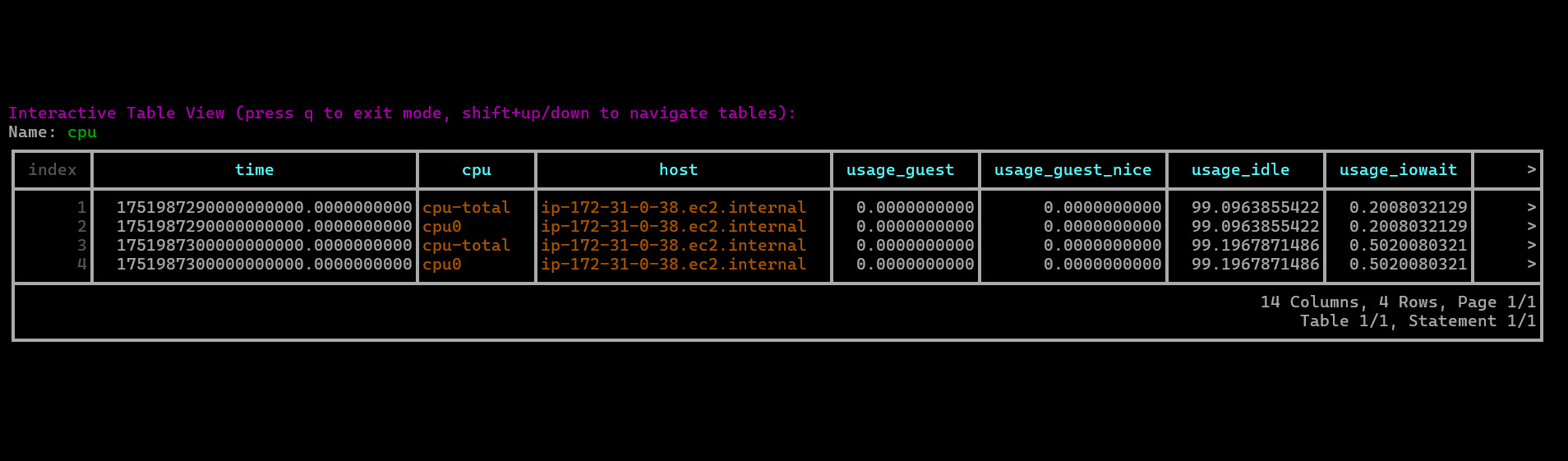
***Setup dash board:***   
  
we need to connect to influx:

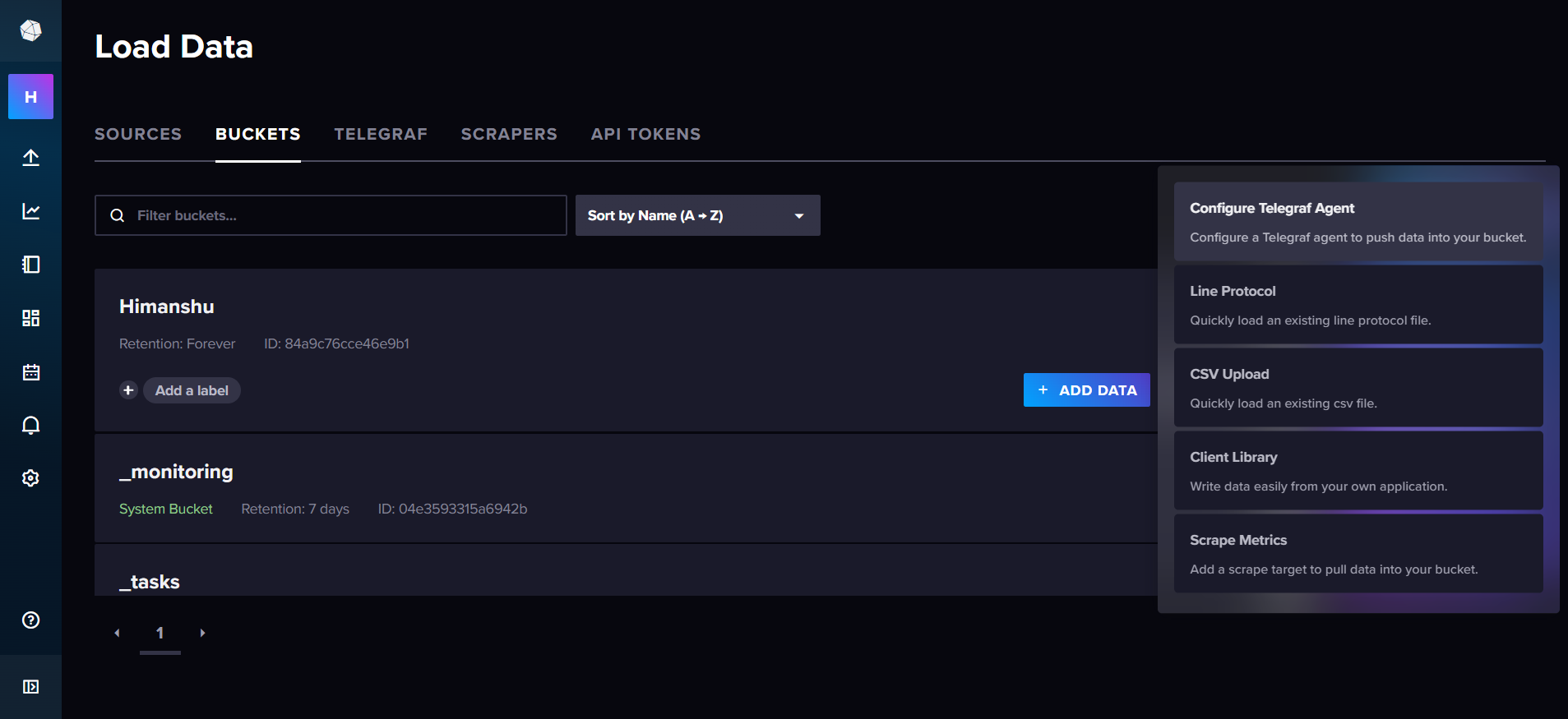
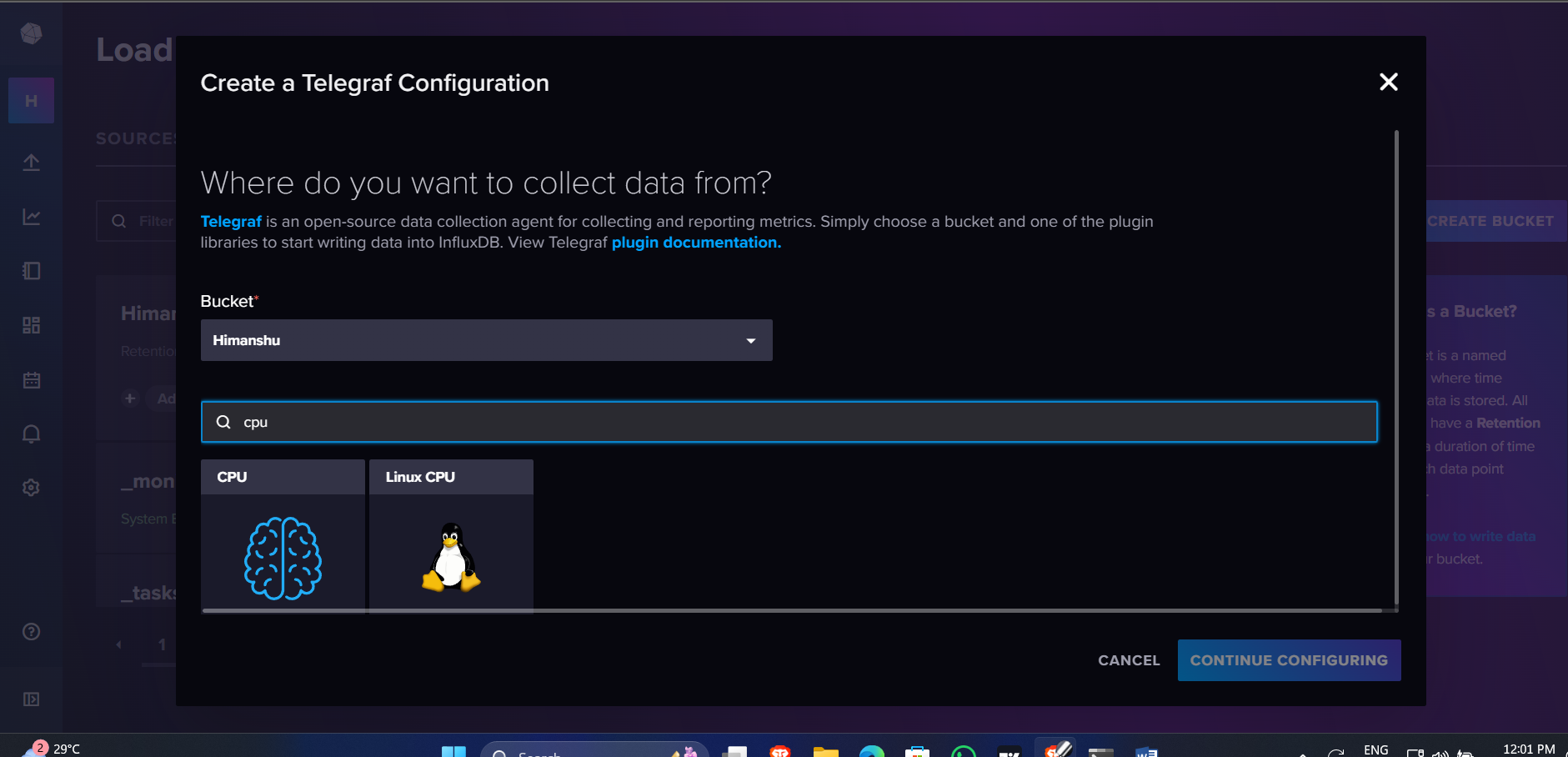
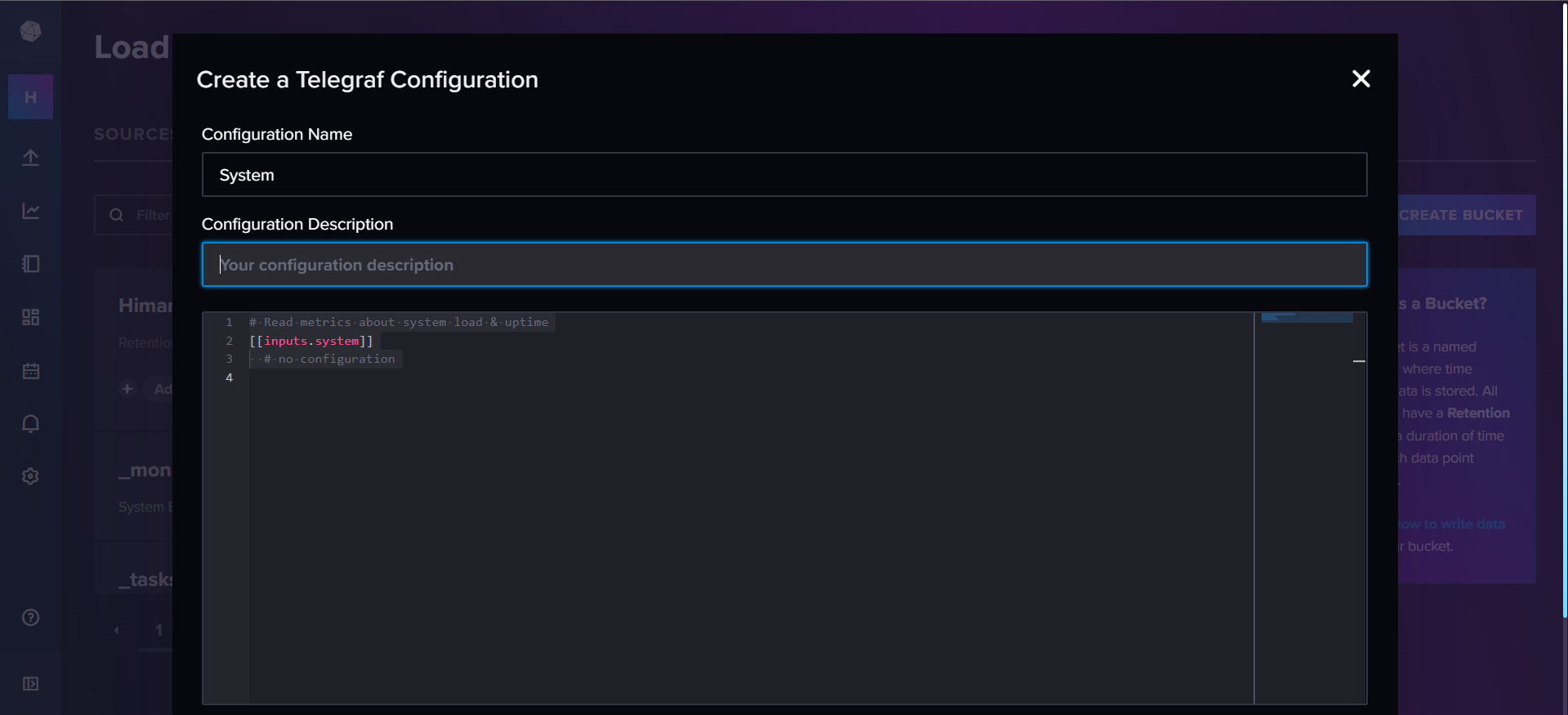
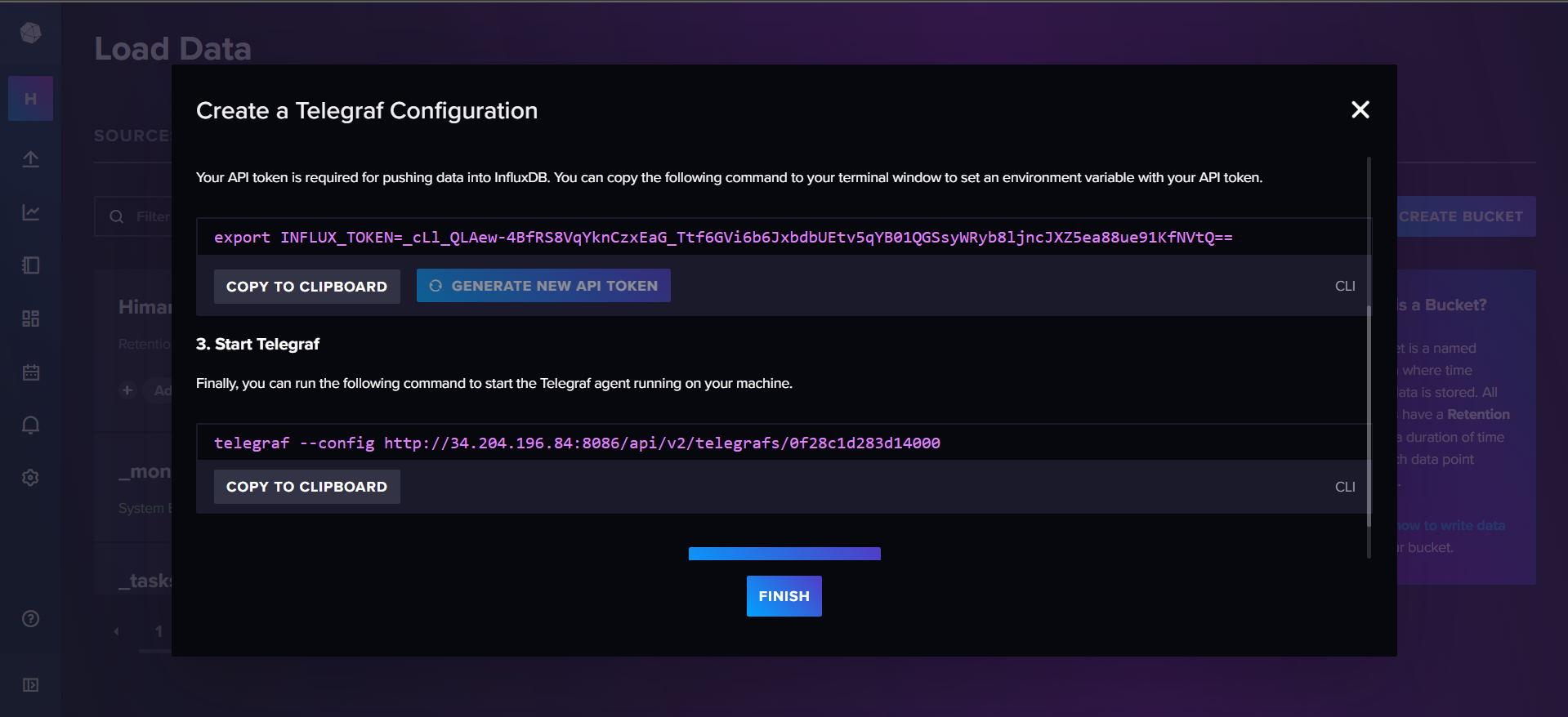
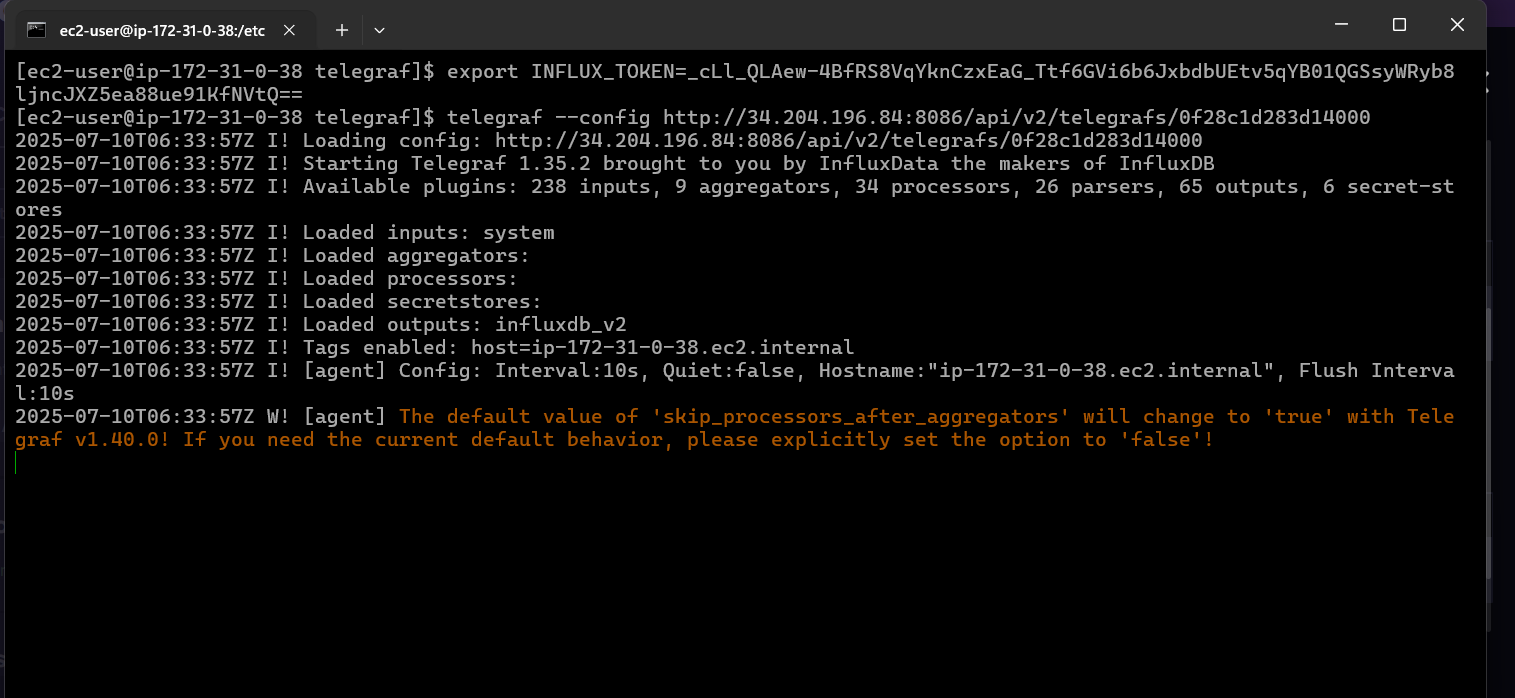
command:

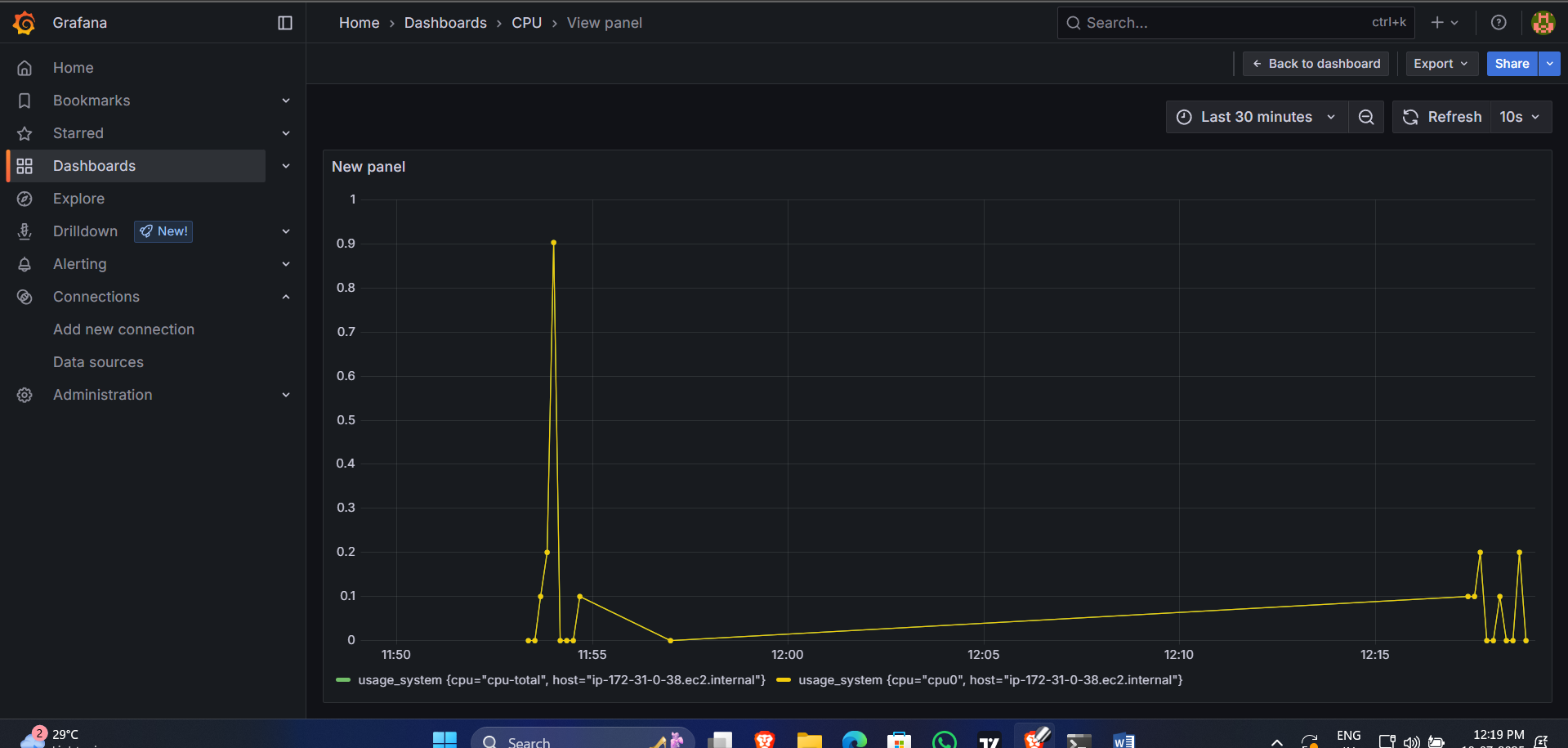
influx v1 shell --host http://localhost:8086 --token GmNrJ4EKlGsP7oG1TnW3XkqnKbEMEgyecp7Ovq9fXjoVYTPZfgIhabqAZ5T1dp3Y8S-y4Vf7shArFMpK-XwS-g==









***1. InfluxDB Setup Issues:***

**🔸 Issue: 401 Unauthorized when using influx CLI**

**Cause:** Missing or invalid token.

**Resolution:**

* You must **set a token** using:

bash

export INFLUX\_TOKEN=your-token-here

* Or use it inline:

bash

influx org list --token your-token

**🔸 Issue: influx v1 shell shows no databases or measurements**

**Cause:** Telegraf data not yet written to InfluxDB.

**Resolution:**

* Ensure Telegraf is correctly configured and running.
* Use:

bash

telegraf --config /etc/telegraf/telegraf.conf

OR

bash

telegraf --config http://localhost:8086/api/v2/telegrafs/<config-id>

* Check with:

bash

show databases;

use "YourBucket";

show measurements;

**📦 2. Telegraf Issues**

**🔸 Issue: telegraf.service failed to start**

**Cause:** Invalid config file (TOML parsing error)

**Resolution:**

* Recreate a clean config:

bash

telegraf --sample-config --input-filter cpu:mem:disk --output-filter influxdb\_v2 | sudo tee /etc/telegraf/telegraf.conf > /dev/null

* Or fetch directly from InfluxDB UI:

bash

telegraf --config http://localhost:8086/api/v2/telegrafs/<config-id>

**🔸 Issue: Telegraf couldn’t connect to InfluxDB (context deadline exceeded)**

**Cause:** Incorrect URL in config or InfluxDB not reachable

**Resolution:**

* Make sure URL in [outputs.influxdb\_v2] is correct:

toml

urls = ["http://localhost:8086"]

* Ensure InfluxDB service is running and reachable.

**📊 3. Grafana + InfluxDB Issues**

**🔸 Issue: Grafana shows unauthorized when adding InfluxDB**

**Cause:** Token missing in data source config

**Resolution:**

* In **Grafana → Add Data Source → InfluxDB**, set:
  + **Query Language:** Flux
  + **URL:** http://localhost:8086
  + **Organization Name:** (e.g. Himanshu)
  + **Token:** Paste your InfluxDB token here
  + **Default Bucket:** (e.g. Himanshu)

**🔸 Issue: No data visible in Grafana after connecting**

**Cause:** Metrics not yet written or wrong bucket

**Resolution:**

* Ensure Telegraf is collecting and sending data.
* Use a Flux query like:

flux

from(bucket: "Himanshu")

|> range(start: -5m)

|> filter(fn: (r) => r.\_measurement == "cpu")

**⚙️ 4. Additional Notes**

* To **run Telegraf as a service**:

bash

sudo systemctl enable telegraf

sudo systemctl start telegraf

* If running using remote config from Influx UI:

bash

export INFLUX\_TOKEN=your-token

telegraf --config http://localhost:8086/api/v2/telegrafs/<config-id>

* If you close this terminal, metrics stop sending. So for persistent collection, set it up via local config and systemd service.

***You (Lab/POC Setup):***

You installed **Telegraf + InfluxDB + Grafana** all on **1 server** (say: EC2-01).

* **Telegraf** collects system metrics (CPU, memory, disk) from that server.
* **InfluxDB** stores that data.
* **Grafana** visualizes it.

This is good for learning, testing, and **small-scale setups**.

**🚀 Real-World Production Setup (1000+ Servers)**

**💡 The architecture is distributed, and each component is scaled or isolated.**

**🖥️ On Each Application/Infrastructure Server (1000+ Servers)**

👉 **Telegraf agent is installed** individually on each server.

* Collects system metrics: CPU, memory, disk, network
* Sends metrics **to a central InfluxDB server** (or a cluster/load balancer).

Telegraf is **lightweight** and perfect for agent-based monitoring on each host.

**🧠 InfluxDB (Centralized Server or Cluster)**

* All Telegraf agents send metrics to this centralized **InfluxDB server or cluster**.
* Usually has high availability and redundancy (InfluxDB Enterprise or OSS + clustering).
* Can be load-balanced.

Example endpoint:

arduino

http://influxdb.mycompany.com:8086

Telegraf config on every server includes:

toml

[[outputs.influxdb\_v2]]

urls = ["http://influxdb.mycompany.com:8086"]

token = "XYZ..."

org = "company"

bucket = "system\_metrics"

**📊 Grafana (Dashboard Server)**

* **Only one or a few instances** of Grafana.
* Reads from InfluxDB to generate dashboards.
* Accessed by DevOps/SRE teams via web UI.

Grafana connects to InfluxDB via the **Data Source** plugin (with token or basic auth).

**🧱 Architecture Diagram (Simplified)**

+------------------+

| Grafana Server |◀───(HTTP GET Queries)

+------------------+

│

▼

+------------------+

| InfluxDB Server| ◀─── Telegraf (1000 Agents)

+------------------+

▲ ▲

┌─────────┘ └─────────┐

│ │

+---------+ +---------+

| Server1 | | ServerN |

| Telegraf| | Telegraf|

+---------+ +---------+

**🧩 Key Notes:**

* **Security:** Telegraf uses InfluxDB tokens or basic auth. Communication is secured (HTTPS).
* **Data Volume:** You can use InfluxDB retention policies or Kafka if the load is high.
* **Scalability:** InfluxDB Enterprise or TSI helps handle high ingest rate and high cardinality.
* **Monitoring Centralization:** Grafana is just a read-only visualization tool — doesn’t collect data.

