Lab 09: TIG IoT Monitoring Stack

Lab Presentation Date: 04/01/2025 (Tue) - 04/03/2025 (Thu)

Demo and Submission Due Date: 04/08/2025 (Tue) - 04/10/2025 (Thu)

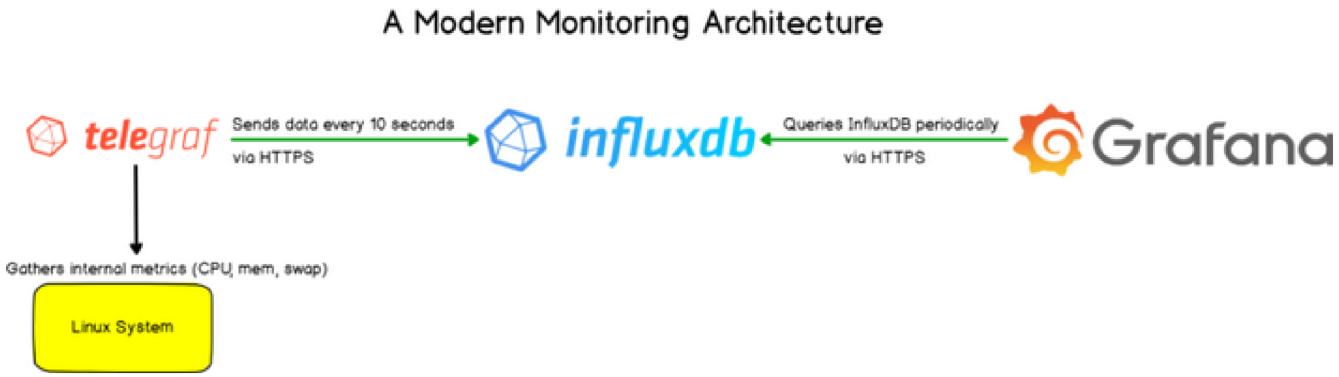
File to submit via Vocareum below (in teams of up to 2):

- Screenshots:
  - Show InfluxDB active
  - Show Telegraf active
  - Show InfluxDB using Telegraf to display CPU data
  - Show Grafana dashboard
- Question writeup (text file)

A little heads up: this lab contains a lot of configs and installations. Go through the instructions slowly, carefully and in order. Also, you will need to use your VM.

1. Introduction

One of the most important parts of modern IoT systems are data monitoring, gathering, storage and visualization. From all the existing modern monitoring tools, the TIG (Telegraf, InfluxDB and Grafana) stack is probably one of the most popular ones. This stack can be used to monitor a wide panel of different data sources: from operating systems (such as Linux Performance metrics), to databases (such as MongoDB or MySQL). Telegraf is an agent responsible for gathering and aggregating data, like the current CPU usage for example. InfluxDB will store data, and expose it to Grafana, which is a modern dashboarding solution.


A Modern Monitoring Architecture
telegraf
Sends data every 10 seconds
via HTTPS
influxdb
Queries InfluxDB periodically
via HTTPS
Grafana
Gathers internal metrics (CPU, mem, swap)
Linux System

(source: https://devconnected.com/how-to-setup-telegraf-influxdb-and-grafana-on-linux/)
You can read more here: https://hackmd.io/@lnu-iot/tig-stack

In this lab, you'll be testing each tool in the stack to visualize your CPU usage.

2. InfluxDB and Telegraf

A. Install InfluxDB on your Ubuntu VM

First, create the repository file:

```
echo "deb [trusted=yes] https://repos.influxdata.com/ubuntu focal stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

By adding [trusted=yes], we're telling the system to bypass GPG key verification for this repository.

Update the Package List and Install InfluxDB:

- Update and install:

```
sudo apt-get update && sudo apt-get install influxdb
```

Start and Verify InfluxDB:

- Start the InfluxDB service:

```
sudo systemctl start influxdb
```

Check if it's running correctly:

```
sudo systemctl status influxdb
```

If you run into permission issues, try running the first three instructions above as root (using sudo su).

B. Check if the service is available [if above steps were okay and influx is running, then no need of this]

```
sudo systemctl status influxdb.service
```

If the above set of instructions does not work and you get errors in GPG keys setup, follow this:

```
sudo nano /etc/apt/sources.list.d/influxdb.list
```

Remove everything that is in this file, and paste the below command:

```
deb [trusted=yes] https://repos.influxdata.com/debian bullseye stable[after this save and exit the file]
sudo apt-get update
sudo apt-get install influxdb]
```

To check if InfluxDB is installed on your system, you can use the following command:

```
influxd version
dpkg -l | grep influxdb
```

Check if working:

```
sudo systemctl status influxdb.service
```

C. Install Telegraf

Step 1: Install Telegraf

```
sudo apt-get install telegraf
```

Step 2: Start Telegraf Service

Start the Telegraf service:

```
sudo systemctl start telegraf
```

Check the status to confirm it’s running:

```
sudo systemctl status telegraf
```

Step 3: Troubleshooting “No outputs found” Error

If you encounter an error indicating "no outputs found," configure Telegraf as follows.
Step 4: Edit the Telegraf Configuration File

Open Telegraf’s configuration file:

```
sudo nano /etc/telegraf/telegraf.conf
```

Locate (or add) the following **[[outputs.influxdb]]** section to point Telegraf to your InfluxDB instance:

```
[[outputs.influxdb]]
urls = ["http://localhost:8086"] # Replace with your InfluxDB URL
database = "telegraf"          # Database name
username = "telegraf"          # Username if authentication is enabled
password = "password"                  # Password if authentication is enabled
```

Note: Adjust the username and password fields according to your InfluxDB authentication settings.

Step 5: Save and Close the Configuration File

Save changes by pressing Ctrl + O, then Enter, and exit with Ctrl + X.

Step 6: Restart Telegraf

Reload systemd to ensure changes take effect:

```
sudo systemctl daemon-reload
```

Restart Telegraf with the updated configuration:

```
sudo systemctl restart telegraf
```

Check the status to confirm it’s active:

```
sudo systemctl status telegraf
```

Step 7: Optional - Test Configuration

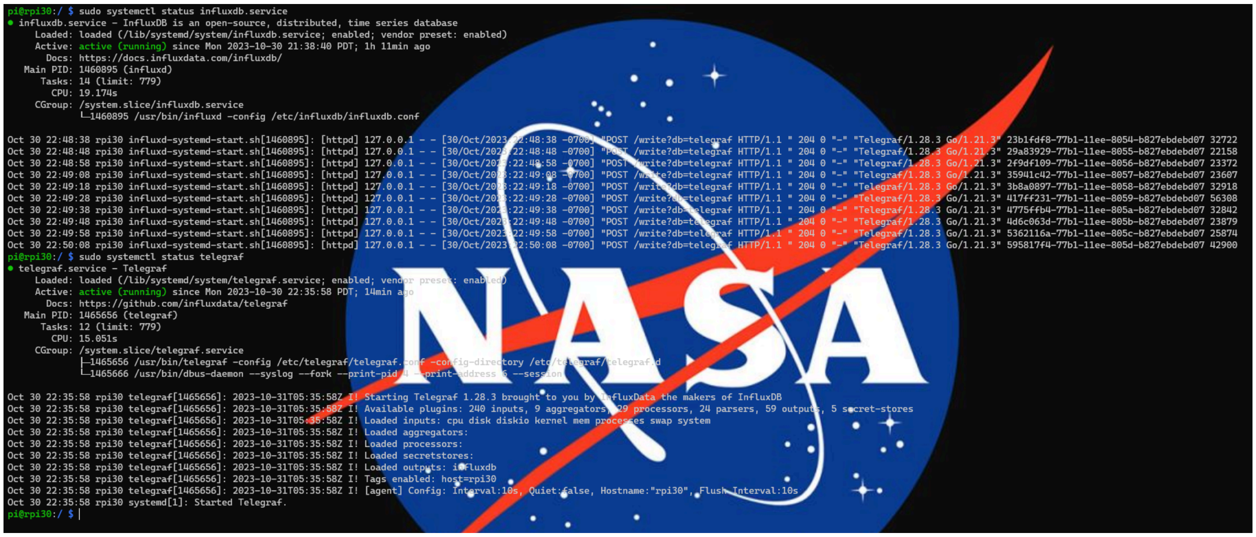
Run Telegraf in test mode to verify it’s collecting data as configured:

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

D. Check if the service is available(like mentioned above)

sudo systemctl status telegraf

Screenshot the active status of both InfluxDB and Telegraf.



```
pi@rpi30:/ $ sudo systemctl status influxdb.service
● influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-10-30 21:38:40 PDT; 1h 11min ago
       Docs: https://docs.influxdata.com/influxdb/
   Main PID: 1460895 (influxd)
      Tasks: 14 (limit: 779)
        CPU: 19.174s
     CGroup: /system.slice/influxdb.service
             └─1460895 /usr/bin/influxd -config /etc/influxdb/influxdb.conf

Oct 30 22:48:38 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:48:38 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 23b1fdf8-77b1-11ee-8054-b827ebdebd07 32722
Oct 30 22:48:48 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:48:48 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 29a83929-77b1-11ee-8055-b827ebdebd07 22158
Oct 30 22:48:58 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:48:58 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 2f9df109-77b1-11ee-8056-b827ebdebd07 23372
Oct 30 22:49:08 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:08 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 35941c42-77b1-11ee-8057-b827ebdebd07 23607
Oct 30 22:49:18 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:18 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 3b8a0897-77b1-11ee-8058-b827ebdebd07 32918
Oct 30 22:49:28 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:28 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 417ff231-77b1-11ee-8059-b827ebdebd07 56308
Oct 30 22:49:38 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:38 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 4775ffb4-77b1-11ee-805a-b827ebdebd07 32842
Oct 30 22:49:48 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:48 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 4d6c063d-77b1-11ee-805b-b827ebdebd07 23879
Oct 30 22:49:58 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:49:58 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 5362116a-77b1-11ee-805c-b827ebdebd07 25874
Oct 30 22:50:08 rpi30 influxd-systemd-start.sh[1460895]: [httpd] 127.0.0.1 - - [30/Oct/2023:22:50:08 -0700] "POST /write?db=telegraf HTTP/1.1 " 204 0 "-" "Telegraf/1.28.3 Go/1.21.3" 595817f4-77b1-11ee-805d-b827ebdebd07 42900
pi@rpi30:/ $ sudo systemctl status telegraf
● telegraf.service - Telegraf
     Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-10-30 22:35:58 PDT; 14min ago
       Docs: https://github.com/influxdata/telegraf
   Main PID: 1465656 (telegraf)
      Tasks: 12 (limit: 779)
        CPU: 15.051s
     CGroup: /system.slice/telegraf.service
             ├─1465656 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d
             └─1465666 /usr/bin/dbus-daemon --syslog --fork --print-pid 4 --print-address 6 --session

Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Starting Telegraf 1.28.3 brought to you by InfluxData the makers of InfluxDB
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Available plugins: 240 inputs, 9 aggregators, 29 processors, 24 parsers, 59 outputs, 5 secret-stores
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Loaded inputs: cpu disk diskio kernel mem processes swap system
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Loaded aggregators:
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Loaded processors:
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Loaded secretstores:
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Loaded outputs: influxdb
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! Tags enabled: host=rpi30
Oct 30 22:35:58 rpi30 telegraf[1465656]: 2023-10-31T05:35:58Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"rpi30", Flush Interval:10s
Oct 30 22:35:58 rpi30 systemd[1]: Started Telegraf.
pi@rpi30:/ $ |
```

3. InfluxDB Authentication and Encryption

Many software tools use a client-server model, or service to service model. Somewhere (locally or remotely), a server is running, and it's accessible only by this software's clients. Examples include mysql, and also influxdb. In the last step, we already got the server up and running, this step we'll use the client's command line tool to talk to the server.

Step-by-Step Guide: Final InfluxDB Configuration with Authentication and HTTPS

1. Enable Authentication in InfluxDB

1. Open the InfluxDB Configuration File:
   sudo nano /etc/influxdb/influxdb.conf
2. **Locate the `[http]` Section** and enable authentication by adding or updating the following lines:

   [http]

enabled = true

bind-address = ":8086"

auth-enabled = true

3. **Save and Close the File**.

4. **Restart InfluxDB** to apply the changes: ***sudo systemctl restart influxdb***

—-----------

2. Create InfluxDB Users for Authentication

1. **Access the InfluxDB CLI**:

   Type in your terminal: **influx**

2. **Create an Admin User**:

   CREATE USER admin WITH PASSWORD ***'your_admin_password'*** *WITH ALL PRIVILEGES;*

3. **Create a Telegraf User** (for Telegraf to connect and write data):

   CREATE USER telegraf WITH PASSWORD ***'your_telegraf_password'***;

4. **Verify Users**:

   SHOW USERS;

5. Exit Influx CLI:

   exit

Replace `'your_admin_password'` and `'your_telegraf_password'` with secure passwords of your choice.

—--------------------------------

3. Enable HTTPS in InfluxDB

1. **Generate SSL Certificates**:

   sudo mkdir -p /etc/ssl/influxdb

   cd /etc/ssl/influxdb

```
# Generate a private key

sudo openssl genpkey -algorithm RSA -out server-key.pem -pkeyopt rsa_keygen_bits:2048



# Generate a self-signed certificate

sudo openssl req -new -x509 -key server-key.pem -out server-cert.pem -days 365
```

During the `openssl req` command, you’ll be prompted to enter details for the certificate (these can be left blank for testing purposes).

2. **Set the Correct Permissions** for the SSL files:

```
sudo chown influxdb:influxdb /etc/ssl/influxdb/server-cert.pem /etc/ssl/influxdb/server-key.pem

sudo chmod 600 /etc/ssl/influxdb/server-cert.pem /etc/ssl/influxdb/server-key.pem
```

3. **Configure InfluxDB to Use HTTPS**:

- Open the InfluxDB configuration file again:

```
sudo nano /etc/influxdb/influxdb.conf
```

In the `[http]` section, update the settings to enable HTTPS and specify the paths for the SSL certificate and key:

```
[http]

  enabled = true

  bind-address = ":8086"

  auth-enabled = true

  https-enabled = true

  https-certificate = "/etc/ssl/influxdb/server-cert.pem"

  https-private-key = "/etc/ssl/influxdb/server-key.pem"
```

4. **Save and Close the File**.

5. **Restart InfluxDB** to apply the HTTPS configuration:

```
sudo systemctl restart influxdb
```

—-------------------------

4. Configure Telegraf to Use HTTPS with Authentication

1. **Edit the Telegraf Configuration File**:

```
sudo nano /etc/telegraf/telegraf.conf
```

2. **Update the `[[outputs.influxdb]]` Section** to use HTTPS and the Telegraf user credentials:

```
[[outputs.influxdb]]
  urls = ["https://localhost:8086"]   # Use HTTPS instead of HTTP
  database = "telegraf"           # Database name
  username = "telegraf"                  # InfluxDB username for Telegraf
  password = "your_telegraf_password" # InfluxDB password for Telegraf
  insecure_skip_verify = true             # Skips SSL verification for self-signed certificates
```

Replace `'your_telegraf_password'` with the password you set for the `telegraf` user.

Save and Close the File.

```
sudo systemctl restart telegraf
sudo systemctl status telegraf
```

—------------------------

5. Test the Setup

1. **Connect to InfluxDB via the CLI with HTTPS** to ensure everything is working:

```
influx -ssl -unsafeSsl -username 'admin' -password 'your_admin_password'
```

2. **Verify Data in the `telegraf` Database**:

In the InfluxDB CLI, run:

```
SHOW DATABASES;

USE telegraf;

SHOW MEASUREMENTS;

SELECT * FROM cpu LIMIT 10;
```

This confirms that Telegraf is securely sending data to InfluxDB.

Question 1a: What type of authentication are we using here (currently)? Does it use any keys?

Question 1b: Both TLS (encryption) and crypto authentication use public-private key pairs. For TLS encryption what keys are used when the client sends a message to the server? For crypto authentication, explain how the server can verify a message is from a given client?

BEFORE PROCEEDING FURTHER, CHECK THE FOLLOWING LIST

At this point, verify the following, you should have all this done already -

1) Enter into the influxdb terminal using the command influx -username ‘admin’ -password ‘password’

   (or after you enter via influx use auth to verify your username and password)

2) In influx terminal, try the following commands -

   a) SHOW DATABASES

      - You must find a database named telegraf, if not, telegraf is not sending system stats to influxdb, in which case use -
      sudo journalctl -f -u telegraf.service
      to find and fix the errors

   b) USE telegraf
      SHOW SERIES

      - you must see a list of time series fields which must contain one related to CPU

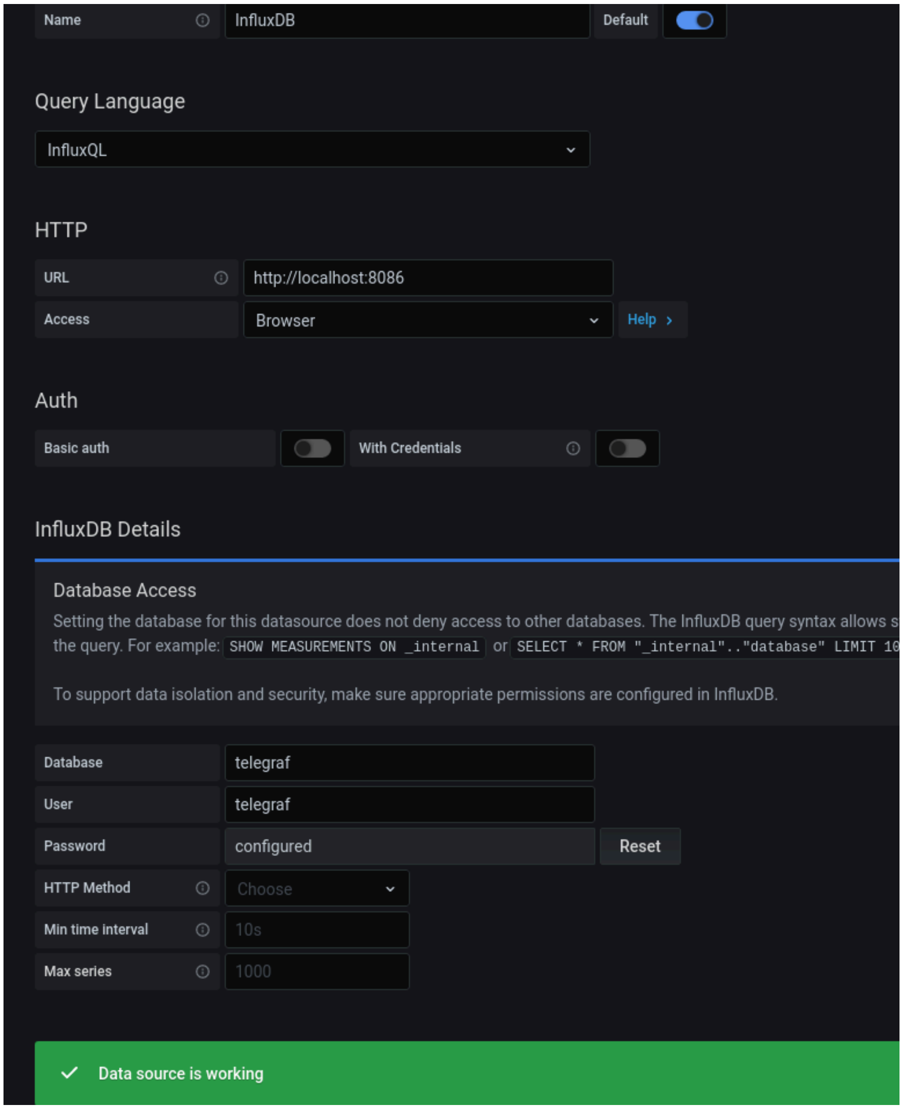
   c) select * from cpu WHERE time > now() - 30s

- you must see some data in this series, which means telegraf is correctly communicating with influxdb

3) OPTIONAL (RECOMMENDED) -

- Go to [Step 4: Grafana](#) in this document , install Grafana and come back to the following steps.
- Add a data source with the settings shown in the screenshot and make sure the data source is working when you hit Save and Test.

  NOTE:
  Username and Password for InfluxDB Database Access is -
  telegraf and password.
-



Name	InfluxDB	Default	●

Query Language

InfluxQL

HTTP

URL	http://localhost:8086	
Access	Browser	Help ›

Auth

Basic auth	○	With Credentials	○

InfluxDB Details

Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows s the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

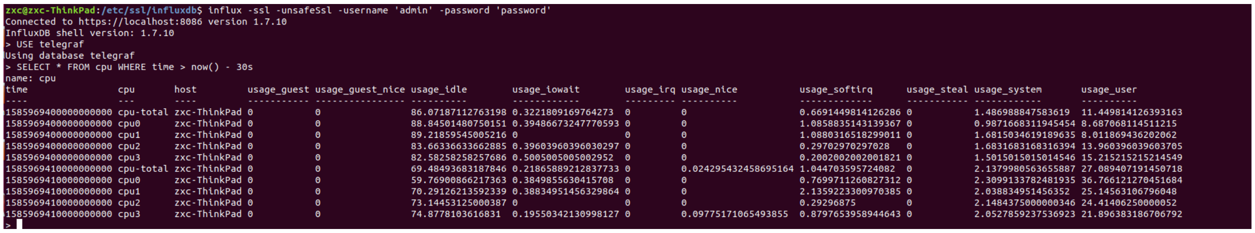
Database	telegraf	
User	telegraf	
Password	configured	Reset
HTTP Method	Choose	
Min time interval	10s	
Max series	1000	

✓ Data source is working

Question 2: Here we created a pair of asymmetric keys. What are their names? Which one is the public key and which one is the private key?

Question 3: What is a certificate authority (CA) for public keys? What kind of attack can a CA prevent?

You should see something like the table below. Screenshot this table.



```
zxc@zxc-ThinkPad:/etc/ssl/influxdb$ influx -ssl -unsafeSsl -username 'admin' -password 'password'
Connected to https://localhost:8086 version 1.7.10
InfluxDB shell version: 1.7.10
> USE telegraf
Using database telegraf
> SELECT * FROM cpu WHERE time > now() - 30s
name: cpu
time                cpu       host         usage_guest usage_guest_nice usage_idle        usage_iowait        usage_irq usage_nice           usage_softirq      usage_steal usage_system       usage_user
----                ---       ----         ----------- ---------------- ----------        ------------        --------- ----------           -------------      ----------- ------------       ----------
1585969400000000000 cpu-total zxc-ThinkPad 0           0                86.07187112763198 0.3221809169764273  0         0                    0.6691449814126286 0           1.486988847583619  11.449814126393163
1585969400000000000 cpu0      zxc-ThinkPad 0           0                88.84501480750151 0.39486673247770593 0         0                    1.0858835143139367 0           0.9871668311945454 8.687068114511215
1585969400000000000 cpu1      zxc-ThinkPad 0           0                89.21859545005216 0                   0         0                    1.0880316518299011 0           1.6815034619189635 8.011869436202062
1585969400000000000 cpu2      zxc-ThinkPad 0           0                83.66336633662885 0.39603960396030297 0         0                    0.29702970297028   0           1.6831683168316394 13.960396039603705
1585969400000000000 cpu3      zxc-ThinkPad 0           0                82.58258258257686 0.5005005005002952  0         0                    0.2002002002001821 0           1.5015015015014546 15.215215215214549
1585969410000000000 cpu-total zxc-ThinkPad 0           0                69.48493683187846 0.21865889212837733 0         0.024295432458695164 1.044703595724082  0           2.1379980563655887 27.089407191450718
1585969410000000000 cpu0      zxc-ThinkPad 0           0                59.76900866217363 0.3849855630415708  0         0                    0.7699711260827312 0           2.3099133782481935 36.766121270451684
1585969410000000000 cpu1      zxc-ThinkPad 0           0                70.29126213592339 0.38834951456329864 0         0                    2.1359223300970385 0           2.038834951456352  25.14563106796048
1585969410000000000 cpu2      zxc-ThinkPad 0           0                73.14453125000387 0                   0         0                    0.29296875         0           2.1484375000000346 24.41406250000052
1585969410000000000 cpu3      zxc-ThinkPad 0           0                74.8778103616831  0.19550342130998127 0         0.0977517106549385 5 0.8797653958944643 0           2.0527859237536923 21.896383186706792
>
```

4. Grafana

Grafana is a real time visualization tool. In this lab we will use it to fetch data real time from our database influxdb. We won't do much real time visualization though, that is for you to explore in your final project.

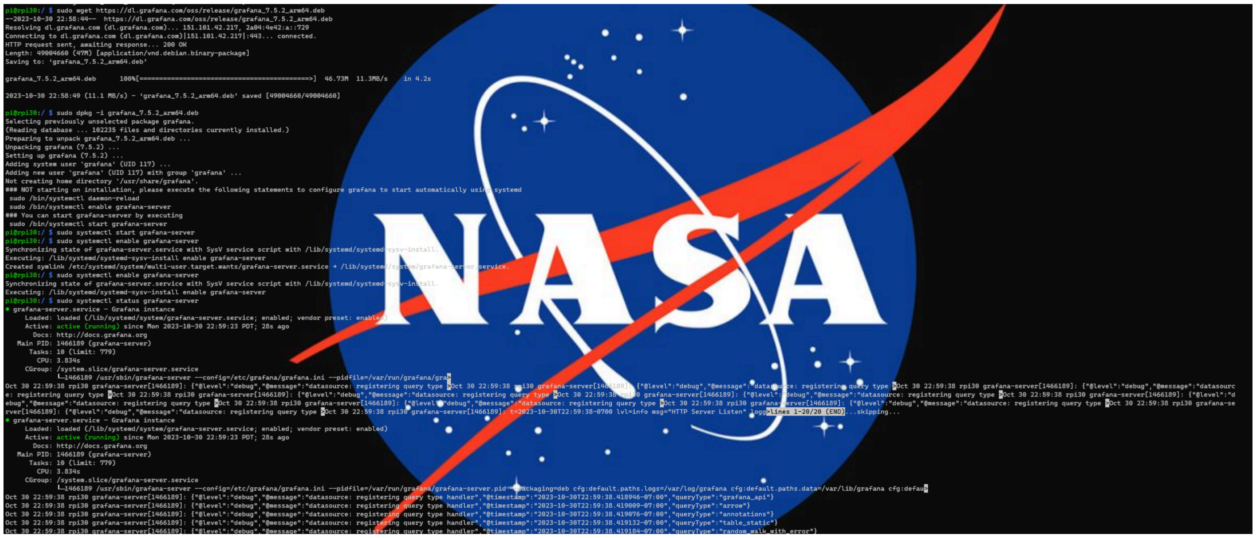
A. Install Grafana

```
sudo apt-get install -y adduser libfontconfig1
sudo wget https://dl.grafana.com/oss/release/grafana_7.5.2_amd64.deb
sudo dpkg -i grafana_7.5.2_amd64.deb
```

(this should work, in the case if your system is arm64 replace amd64 with arm64)

sudo systemctl start grafana-server

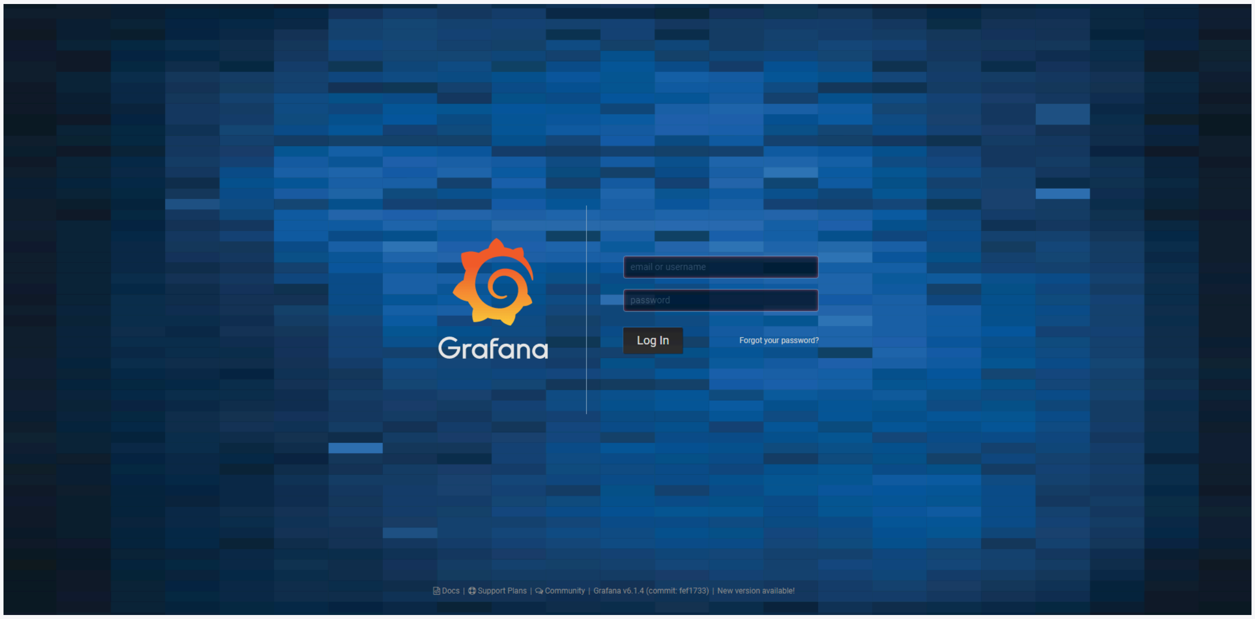
sudo systemctl status grafana-server
(my system is arm64, hence I used arm64 in my cmds)



```
pi@rpi30:/ $ sudo wget https://dl.grafana.com/oss/release/grafana_7.5.2_arm64.deb
--2023-10-30 22:58:44--  https://dl.grafana.com/oss/release/grafana_7.5.2_arm64.deb
Resolving dl.grafana.com (dl.grafana.com)... 151.101.42.217, 2a04:4e42:a::729
Connecting to dl.grafana.com (dl.grafana.com)|151.101.42.217|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 49004660 (47M) [application/vnd.debian.binary-package]
Saving to: ‘grafana_7.5.2_arm64.deb’

grafana_7.5.2_arm64.deb      100%[===========================================>]  46.73M  11.3MB/s    in 4.2s

2023-10-30 22:58:49 (11.1 MB/s) - ‘grafana_7.5.2_arm64.deb’ saved [49004660/49004660]

pi@rpi30:/ $ sudo dpkg -i grafana_7.5.2_arm64.deb
Selecting previously unselected package grafana.
(Reading database ... 102235 files and directories currently installed.)
Preparing to unpack grafana_7.5.2_arm64.deb ...
Unpacking grafana (7.5.2) ...
Setting up grafana (7.5.2) ...
Adding system user `grafana' (UID 117) ...
Adding new user `grafana' (UID 117) with group `grafana' ...
Not creating home directory `/usr/share/grafana'.
### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
pi@rpi30:/ $ sudo systemctl start grafana-server
pi@rpi30:/ $ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /lib/systemd/system/grafana-server.service.
pi@rpi30:/ $ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
pi@rpi30:/ $ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-10-30 22:59:23 PDT; 28s ago
       Docs: http://docs.grafana.org
   Main PID: 1466189 (grafana-server)
      Tasks: 10 (limit: 779)
        CPU: 3.834s
     CGroup: /system.slice/grafana-server.service
             └─1466189 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/gra>
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type >Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasourc
e: registering query type >Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type >Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"d
ebug","@message":"datasource: registering query type >Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type >Oct 30 22:59:38 rpi30 grafana-se
rver[1466189]: {"@level":"debug","@message":"datasource: registering query type >Oct 30 22:59:38 rpi30 grafana-server[1466189]: t=2023-10-30T22:59:38-0700 lvl=info msg="HTTP Server Listen" logg>lines 1-20/20 (END)...skipping...
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-10-30 22:59:23 PDT; 28s ago
       Docs: http://docs.grafana.org
   Main PID: 1466189 (grafana-server)
      Tasks: 10 (limit: 779)
        CPU: 3.834s
     CGroup: /system.slice/grafana-server.service
             └─1466189 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/var/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana cfg:default.paths.data=/var/lib/grafana cfg:defau>
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type handler","@timestamp":"2023-10-30T22:59:38.418946-07:00","queryType":"grafana_api"}
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type handler","@timestamp":"2023-10-30T22:59:38.419009-07:00","queryType":"arrow"}
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type handler","@timestamp":"2023-10-30T22:59:38.419076-07:00","queryType":"annotations"}
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type handler","@timestamp":"2023-10-30T22:59:38.419132-07:00","queryType":"table_static"}
Oct 30 22:59:38 rpi30 grafana-server[1466189]: {"@level":"debug","@message":"datasource: registering query type handler","@timestamp":"2023-10-30T22:59:38.419184-07:00","queryType":"random_walk_with_error"}
```

Now that everything is configured properly, you should be able to access the Grafana v6 Web UI. Open a web browser, and navigate to http://localhost:3000. As a reminder, the default port for Grafana is 3000.

You will be presented with this screen when launching the application for the first time.

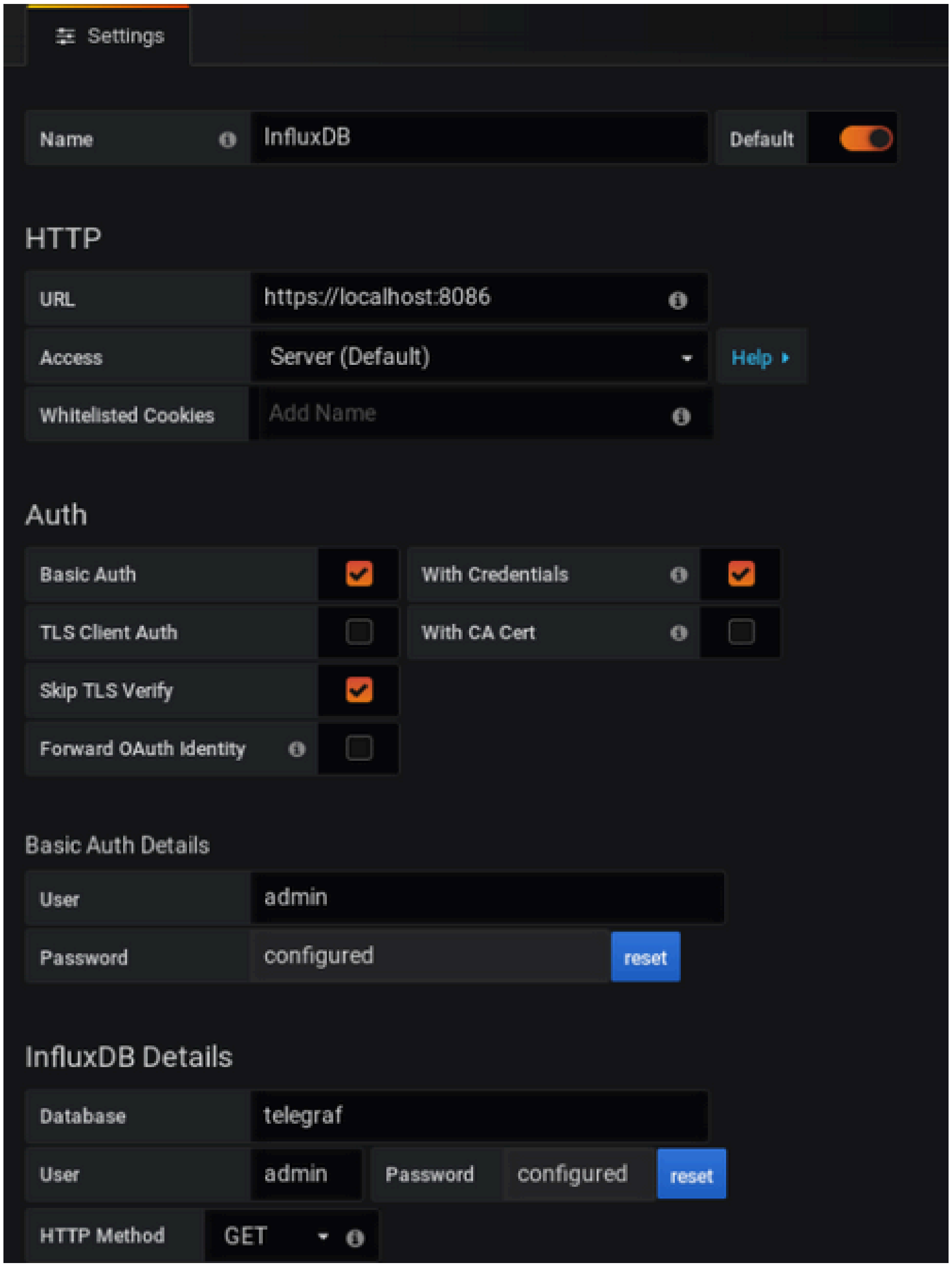

Grafana
email or username
password
Log In
Forgot your password?
Docs | Support Plans | Community | Grafana v6.1.4 (commit: fef1733) | New version available!

The default login for Grafana is ‘admin’ and the default password is also ‘admin’. When setting those credentials, you will be asked to change your password. Choose a password, and click on ‘Save‘.

B. Add influxdb as data source

NOTE: too many failed attempts of authentication will result in a lock without notification prompt

Click on the panel on the left, choose Configuration > Data Sources -> Add Data Source, choose InfluxDB. Use the below config, with password for 'Basic Auth' and the password for 'InfluxDB Details' set to the password for Influxdb. If that doesn't work, disable 'Basic Auth, or delete this influxdb and add it again.



Name: InfluxDB
Default: ●

HTTP

URL: https://localhost:8086
Access: Server (Default)

Whitelisted Cookies: Add Name

Auth

Basic Auth ☑
With Credentials ☑
TLS Client Auth ☐
With CA Cert ☐
Skip TLS Verify ☑
Forward OAuth Identity ☐

Basic Auth Details

User: admin
Password: configured

InfluxDB Details

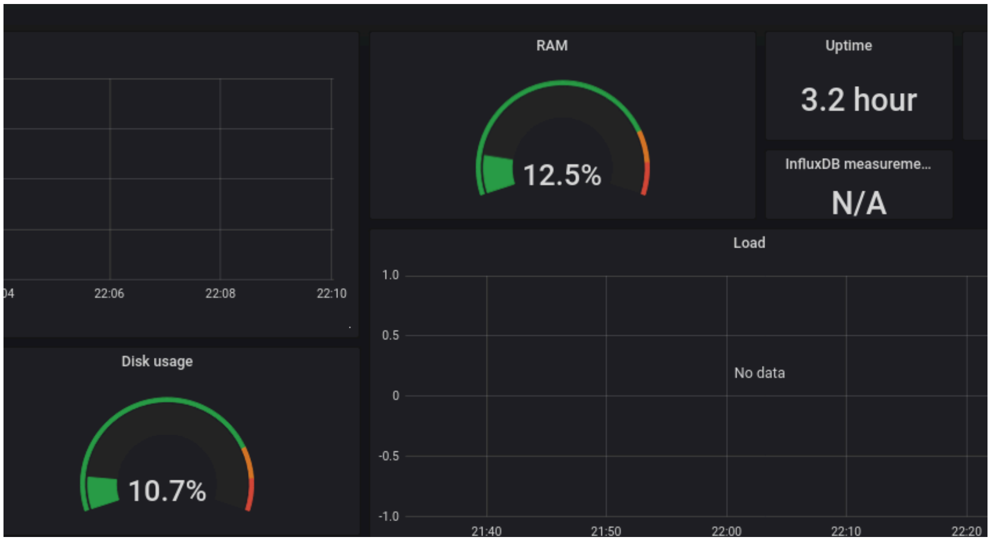
Database: telegraf
User: admin
Password: configured

HTTP Method: GET

Click "Save & Test".

Next step is to import a dashboard. We can make our own dashboards, but for this particular lab, we'll use someone else's for convenience. Click Create > Import on the left menu, then load **dashboard ID 8451** from grafana.com.

You should be able to load system info (gathered by Telegraf) in Grafana. Screenshot your data visualization, like below.



After this point, there will be many ways to configure different time slots and visualize different data real time. That's something we won't talk about here, but you can check online and explore it for your final project!

5. Demo and Code Grading Rubric

All files are to be submitted via Vocareum in teams.

Points	**Description**
Demo	
2	Show influxdb system status
2	Show telegraf system status
2	Show telegraf communicating with influxdb via HTTPS
2	Show grafana importing data from influxdb
Questions	
2	Question1
2	Question2

2	Question3
	Total points: 14

References:

https://devconnected.com/how-to-setup-telegraf-influxdb-and-grafana-on-linux/
https://devconnected.com/how-to-install-grafana-on-ubuntu-18-04/
https://grafana.com/docs/grafana/latest/installation/debian/