Current version: **2020-10-19**

Lab 9: TIG IoT Monitoring Stack

Start Date: 10.19, 2020
Code Submission & Demo Due Date: Before 11:59 PM on 10.30 via Vocareum.

File to submit via Vocareum below (one or two persons per team)

- Link to demo video
- Question writeup (text file)

A little heads up: this lab contains a lot of configs and installations. Although all the steps are there, make sure you read and understand the commands, don't just copy paste them. Installation and setup should be in strict order as the tutorial, don't run the commands back and forth as they could ruin the system (unless you know exactly what you're doing).

Also, every computer is a little different, the instructions in this tutorial is a general one, and there's no guarantee that it'll work exactly the same on every computer. If you run into problems at a particular step, try to see what's going on and how so solve the problems.

1. Introduction

In this class, we mainly talk about distributed systems and IoT stack. 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 one. 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.

(source: https://devconnected.com/how-to-setup-telegraf-influxdb-and-grafana-on-linux/)

2. Github Classroom

There will be no github classroom for this lab

3. InfluxDB and Telegraf Installation

1. Install InfluxDB on ubuntu

```
sudo wget https://dl.influxdata.com/influxdb/releases/influxdb_1.7.10_amd64.deb
sudo dpkg -i influxdb_1.7.10_amd64.deb
sudo systemctl start influxdb.service
sudo systemctl status influxdb.service    # check if service is available
```

2. Install Telegraf

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME}
stable" | sudo tee /etc/apt/sources.list.d/influxdb.list


sudo apt-get update
sudo apt-get install telegraf
sudo systemctl status telegraf  # check the status of your service
```

At this point, you should have both influxdb and telegraf running on your VM. Show their system status.

4. 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.

1. Create admin account
   Type "influx" in the terminal, and type in the influx command line tool
   ```
   > CREATE USER admin WITH PASSWORD 'password' WITH ALL PRIVILEGES
   > SHOW USERS # (verify your last command, see if users are there)
   ```

2. Create a user account for telegraf
   ```
   > CREATE USER telegraf WITH PASSWORD 'password' WITH ALL PRIVILEGES
   > SHOW USERS
   ```

3. Enable HTTP authentication on your InfluxDB server
   In file **/etc/influxdb/influxdb.conf**, modify lines:
   ```
   [http]
       # Determines whether HTTP endpoint is enabled.
       enabled = true

       # The bind address used by the HTTP service.
       bind-address = ":8086"

       # Determines whether user authentication is enabled over HTTP/HTTPS.
       auth-enabled = true
   ```

4. Configure HTTP authentication on Telegraf
   In file **/etc/telegraf/telegraf.conf**, add your credentials
   ```
   ## HTTP Basic Auth
     username = "telegraf"
     password = "password"
   ```

   Then restart the influxdb and telegraph services:
   ```
   sudo systemctl restart influxdb
   sudo systemctl restart telegraf
   sudo journalctl -f -u telegraf.service # check journal to make sure there's no
   error
   ```

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?

5. Configure HTTPS on InfluxDB to encrypt your end-to-end traffic

   Create a key for your InfluxDB server:

```
sudo apt-get install gnutls-bin
cd /etc/ssl
sudo mkdir influxdb && cd influxdb
sudo certtool --generate-privkey --outfile server-key.pem --bits 2048
```

Create another key for your InfluxDB server:

```
sudo certtool --generate-self-signed --load-privkey server-key.pem --outfile
server-cert.pem  # (NOTE: leave all entries empty. Put a random number for
expiration date. Hit 'y' in the last step!!)
```

Set the ownership permissions for the InfluxDB user and group:

```
sudo chown influxdb:influxdb server-key.pem server-cert.pem
```

Enable HTTPS on your InfluxDB server:
In file **/etc/influxdb/influxdb.conf**, ensure that these lines match.

```
# Determines whether HTTPS is enabled.
https-enabled = true

# The SSL certificate to use when HTTPS is enabled.
https-certificate = "/etc/ssl/influxdb/server-cert.pem"

# Use a separate private key location.
https-private-key = "/etc/ssl/influxdb/server-key.pem"
```

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?

Restart the services

```
sudo systemctl restart influxdb
sudo journalctl -f -u influxdb.service
```

6. Configure Telegraf for HTTPS so that it can reach influxDB via HTTPS:
   In file **/etc/telegraf/telegraf.conf,** ensure that these lines match. You'll need to find the right plugin to modify for influxdb. Hint: look for '[[outputs.influxdb]].'
   (remember to uncomment where needed -- delete the '#')

```
# Configuration for sending metrics to InfluxDB
[[outputs.influxdb]]

# https, not http!
urls = ["https://127.0.0.1:8086"]

## Use TLS but skip chain & host verification
insecure_skip_verify = true
```

Now restart the service.

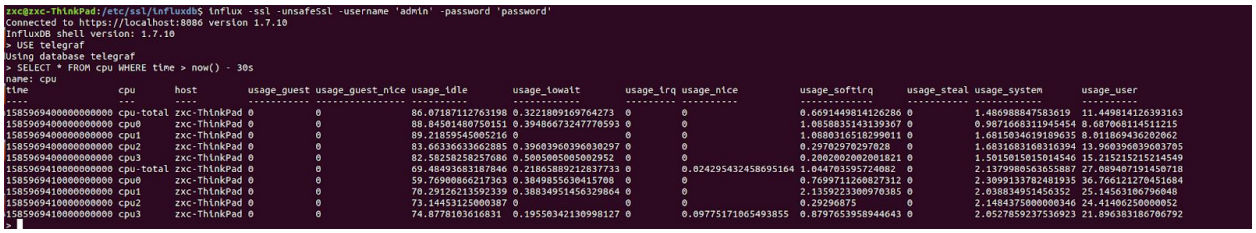
```
sudo systemctl restart telegraf
```

```
sudo journalctl -f -u telegraf.service
```

Telegraf is now using HTTPS to communicate with influxdb (so your query is client-to-server encrypted). Now connect to influx using your ‘admin’ credentials and display cpu data over the last 30 seconds.

```
influx -ssl -unsafeSsl -username 'admin' -password 'password'
> USE telegraf
> SELECT * FROM cpu WHERE time > now() - 30s
```

After following the steps above, you should see something like the table below. Show this table for checkoff.



```
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 0.3948667324777059 3 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 0.3960396039603297 0         0                   0.2970297029702828 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 0.2186588921283773 3 0         0.024295432458695164 1.044703595724082 0           2.1379980563655887 27.089487191450718
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 0.3883495145632986 4 0         0                   2.1359223300970385 0           2.038834951456352  25.145631067960848
1585969410000000000 cpu2      zxc-ThinkPad 0           0                73.14453125000387 0               0         0                   0.29296875         0           2.148437500000346 24.414062500000052
1585969410000000000 cpu3      zxc-ThinkPad 0           0                74.8778103616831  0.1955034213098127 0         0.09775171065493855 0.8797653958944643 0           2.0527859237536923 21.896383186706792
>
```

5. 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.

(having trouble installing? See https://piazza.com/class/kcl244c0dol4h0?cid=245; which indicates a missing step of:
`echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list`

before performing the `sudo apt-get install grafana` in the following guide.

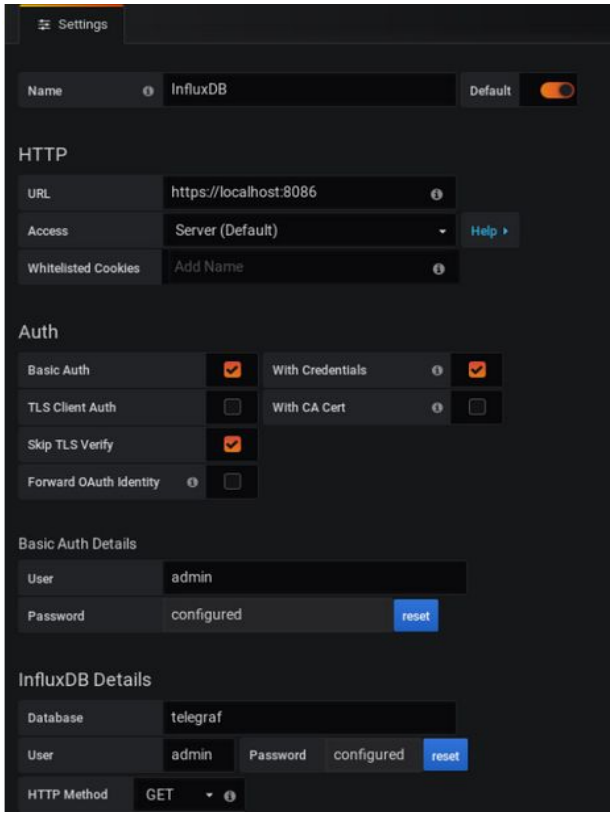
1. Install grafana

https://devconnected.com/how-to-install-grafana-on-ubuntu-18-04/ (you don’t have to look at step 7 and after)

2. Add influxdb as data source

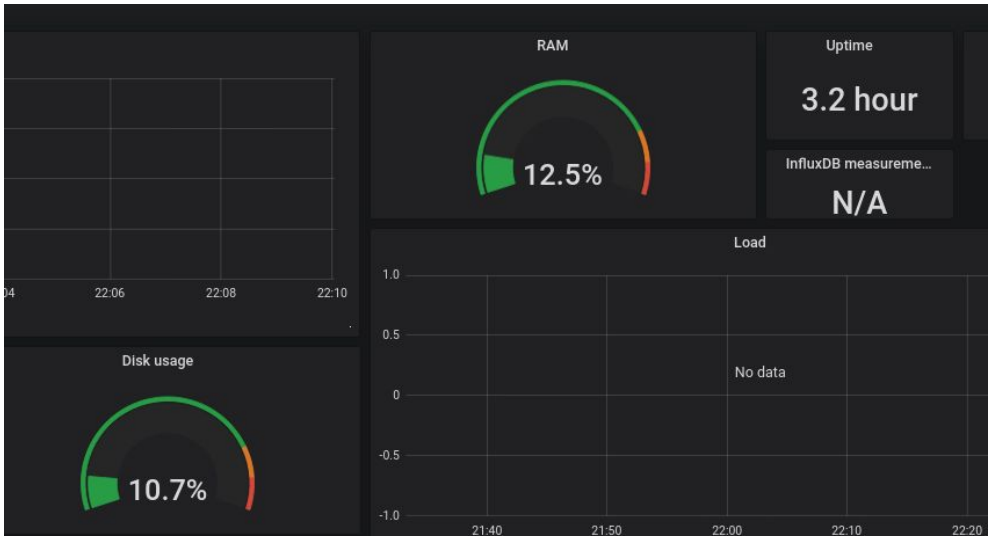
NOTE: too many failed attempts of authentication will result in a lock without notification prompt

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’ set to the password for grafana 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.



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 import on the left menu, then enter dashboard ID 8451, then click import.

You should be able to load system info (gathered by telegraf) in grafana. You need to get checked off for this.



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 it here, but you can check online and explore it for your final project!

6. 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
6	Show telegraf communicating with influxdb via HTTPS
4	Show grafana importing data from influxdb
Questions	
1+1+1	Question1
1+1	Question2
1+1	Question3
	Total points: 21

Reference:
https://devconnected.com/how-to-setup-telegraf-influxdb-and-grafana-on-linux/