

Week12A - Visualize Sensor data using Grafana and InfluxDB

กลุ่มคำานาณ 3 รหัส B6130381 ชื่อ-สกุล อนุสรณ์ ศรีพรหม

1/4 การทำรายงาน

- เป็นการเก็บคะแนน 10 คะแนนตัดเกรด ให้ทำสุดความสามารถในเวลาที่มี
- ให้รีบส่งโดยถือว่าใครส่งก่อนถือว่าเป็นต้นฉบับส่งที่หลังถือว่าลอกเพื่อนมา
- ให้แก้ไขหัวกระดาษ, เพิ่มเติมเอกสารให้สมบูรณ์ รูปภาพการทำงานทั้งวงจร และโค้ดโปรแกรม
- เพิ่มเติมเนื้อหาด้านท้าย ด้วยการคัดลอกและจัดเรียงใหม่
- รูปที่เป็นการทดสอบ ESP32 ควรเป็นรูปของตัวเองที่ทดสอบ
- รูปถ่ายต้องเป็นของตัวเองและมีกระดาษรองอุปกรณ์ที่เขียน รหัส ชื่อ-สกุล ของตัวเอง
- บันทึกไฟล์ในรูป pdf, กำหนดชื่อไฟล์ **Wk12A-B3701234-Wichai-Srisuruk.pdf** (แก้ไขตามรหัส ชื่อตัวเอง)
- ส่งงานก่อน 06:00น วันพุธสุดที่ 24 มิย 64 ที่ Link >> <https://shorturl.at/efuM2>

2/4. Read More

- <https://grafana.com/blog/2021/03/08/how-i-built-a-monitoring-system-for-my-avocado-plant-with-arduino-and-grafana-cloud/>
- <https://www.metricfire.com/blog/iot-dashboards-with-grafana-and-prometheus/>
- <https://gabrieltanner.org/blog/grafana-sensor-visualization>

3/4. ให้จัดเรียบเรียงข้อมูล

- <https://www.techtalkthai.com/arm-pelion-full-stack-iot-platform/>

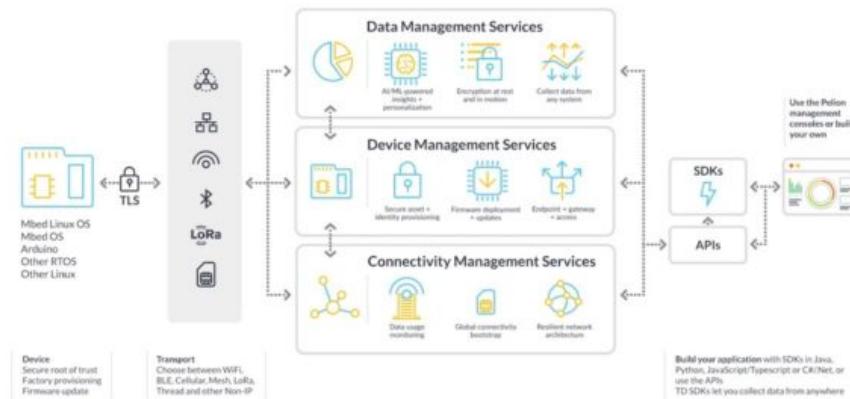
Arm-Pelion Full Stack IoT Platform

รู้จัก ARM Pelion แพลตฟอร์ม IoT จาก ARM จัดการทุกอย่างครบจบในที่เดียว

เมื่อวันที่ 30 กรกฎาคมที่ผ่านมา ทาง ARM ได้จัด IoT Workshop ขึ้นเพื่อนำเสนอแนวคิดต่าง ๆ ของการนำเทคโนโลยี IoT ไปใช้ในธุรกิจ ทีมงาน TechTalkThai ได้เข้าไปร่วมงาน และทำความรู้จักกับ ARM Pelion IoT Platform จึงอยากจะมาเล่าให้ผู้อ่านฟังกันคร่าวๆว่าเจ้าแพลตฟอร์ม IoT นี้มีความสามารถอย่างไร และเหมาะสมกับธุรกิจแบบไหนบ้าง

IoT นั้นเป็นหนึ่งในเทคโนโลยีที่หลายองค์กรยกให้เป็นยุทธศาสตร์ในปี 2019 จากความสามารถในการรวบรวมข้อมูล ซึ่งนับว่าเป็นวัตถุดิบในการทำธุรกิจที่ขาดไม่ได้เลยในยุคปัจจุบัน จนถึงตอนนี้ หลายองค์กรอาจเริ่มต้นกับ IoT กันบ้างแล้ว แต่ความท้าทายใหม่ที่ธุรกิจมักจะเผชิญกับคือการจัดการและสเกลระบบ IoT ให้ใช้งานเก็บข้อมูลได้จริงเต็มประสิทธิภาพ ปลอดภัย และมีระบบจัดการที่ดี ดังนั้นจึงมีการพัฒนาแพลตฟอร์ม IoT ขึ้น เพื่อช่วยธุรกิจในการแก้ปัญหานี้

ARM Pelion IoT Platform ก็เป็นหนึ่งในแพลตฟอร์ม IoT ที่จะเข้ามาช่วยลดความซับซ้อนของการนำ IoT ไปใช้งานในธุรกิจ แพลตฟอร์ม Pelion นี้แบ่งออกเป็น 3 ส่วน ตามการใช้งาน คือ Connectivity Management Services, Device Management Services, และ Data Management Services โดยทั้ง 3 ส่วน จะทำงานร่วมกันภายใต้ระบบรักษาความปลอดภัย ซึ่งเป็นหลักสำคัญที่สุดในการพัฒนาผลิตภัณฑ์ทุกๆตัวของ Arm



ภาพรวมของแพลตฟอร์ม Pelion ที่แบ่งการทำงานออกเป็น 3 ส่วน โดยธุรกิจสามารถเลือกใช้เพียงส่วนใดส่วนหนึ่งหรือทั้ง 3 ส่วนร่วมกันได้ (ภาพ: ARM)

Pelion จะช่วยให้ธุรกิจจัดการกับเครือข่าย อุปกรณ์ในเครือข่าย และข้อมูลที่เก็บมาได้จ่ายชิ้น โดยสามารถทำงานร่วมกับอุปกรณ์ ระบบเครือข่าย คลาวด์ และข้อมูลได้หลากหลายรูปแบบ อีกทั้งยังมีความปลอดภัย และสามารถช่วยในการนำข้อมูลไปวิเคราะห์และแสดงผลเบื้องต้นได้ด้วย

รู้จักแพลตฟอร์มนี้ไปคร่าวๆแล้ว ลองมาเจาะลึกกันว่าส่วนประกอบทั้ง 3 ส่วน อันได้แก่ Connectivity Management, Device Management, และ Data Management นั้นประกอบไปด้วยอะไร และมีจุดเด่นอย่างไรบ้าง

Connectivity Management

การเชื่อมต่อในเครือข่าย IoT นั้นมีอยู่หลายรูปแบบ และมีรายละเอียดปลีกย่อยที่ธุรกิจจะต้องจัดการอยู่ พอกล่าว Pelion จะช่วยให้องค์กรสามารถจัดการการเชื่อมต่อได้อย่างมีประสิทธิภาพ ปลอดภัย และพร้อมต่อ การสแกลเครือข่ายชิ้นไปถึงระดับโลก โดย Connectivity Management ของ Pelion มีความสามารถที่น่าสนใจ ดังนี้

Global Cellular

Pelion จะช่วยให้อุปกรณ์ IoT สามารถเชื่อมต่อผ่านเครือข่ายได้ไม่ว่าอุปกรณ์นั้นจะอยู่ที่ใดในโลก ผ่าน เวนเดอร์เพียงเจ้าเดียว โดยกลไกของ Pelion จะช่วยเชื่อมต่อสัญญาณจากชิมของอุปกรณ์ไปยังเครือข่ายท้องถิ่นที่ Pelion ได้ทำข้อตกลงไว้ ลดภาระความปวดหัวในการติดต่อกับผู้ให้บริการเครือข่ายในแต่ละประเทศ

Protocol เชื่อมต่อทั้ง IP และ Non-IP

นอกจากส่งข้อมูลผ่าน IP Network แล้ว Pelion ยังรองรับ Non-IP Network เช่น NB-IoT ด้วย โดย โปรโตคอลที่ Pelion รองรับนั้นมีได้แก่ MQTT(s), HTTPS, และ Sockets

ใช้ได้ทั้ง eSIM และซิมแบบปกติ

ธุรกิจสามารถสั่งผลิตอุปกรณ์ที่มีระบบ eSIM ผ่าน ARM ได้ตามต้องการ โดย eSIM ที่ติดมากับอุปกรณ์ นั้นจะรองรับการเชื่อมต่อกับเครือข่ายกว่า 600 เครือข่ายทั่วโลก และหากต้องการเปลี่ยนเครือข่าย ก็สามารถตั้งค่า ใหม่ได้ภายหลัง และในส่วนของซิมแบบปกติเอง Pelion ก็ให้บริการซิมการ์ดในทุกขนาด อีกทั้งยังมีแผนที่จะ พัฒนาไปจนถึง iSIM ที่มีขนาดเล็กกว่า eSIM หากด้วย

Network Infrastructure

Pelion ได้พัฒนาโครงสร้างพื้นฐานของเครือข่ายให้สามารถทำงานร่วมกับผู้ให้บริการเครือข่ายทั่วโลกได้อย่างมีประสิทธิภาพสูงสุด โดยมีทั้งความเสถียร ยืดหยุ่น และเป็นไปตามกฎข้อบังคับด้านข้อมูลของแต่ละประเทศในการใช้ Pelion ผู้ใช้จะสามารถเลือกได้ว่าจะส่งข้อมูลจากอุปกรณ์ไปยังแอปพลิเคชันผ่านเทคโนโลยีใด เช่น IPSEC, Open VPN, ผู้ให้บริการ Cloud, หรือทางเชื่อมที่ธุรกิจเข้ามาใช้โดยเฉพาะ (Leased Line)

Device Management

Device Management ของ Pelion นั้นจะช่วยให้องค์กรสามารถจัดการกับอุปกรณ์และการเชื่อมต่อกับอุปกรณ์ผ่านซอฟต์แวร์ได้โดยสะดวก ไม่ว่าจะเป็นการเขียนซอฟต์แวร์แบบ Embedded หรือว่าการเขียนแอปพลิเคชันด้านบนอย่าง Web App ก็ตาม



(ภาพ: ARM)

โดยภายในโซลูชัน Device Management ก็จะมีโมดูลในการจัดการเรื่องต่าง ๆ ให้อย่างครบถ้วน ตั้งแต่เรื่องการอัพเดท Firmware ซึ่งไม่ง่ายเลยหากมีอุปกรณ์ที่หลากหลายและมีจำนวนที่มากในเครือข่าย, Access Management ซึ่งจะช่วยจำกัดการเข้าถึงอุปกรณ์และการควบคุมแต่ละส่วน, Connector ซึ่งจัดการการเชื่อมต่อกับอุปกรณ์หลากหลายประเภท การออก Certificate เข้าใช้ระบบ การอัปเดตอุปกรณ์แต่ละตัว และการเก็บสถิติ, Device Directory ซึ่งช่วยในการแบ่งกลุ่ม ค้นหา เรียกดู และเช็คสถานะของอุปกรณ์แต่ละตัว, ไปจนถึงการรักษาความปลอดภัยในการส่งต่อรหัสผ่านเครือข่าย Wifi

โดยทั้งหมดนี้จะเห็นได้ว่าเป็นการจัดการ Lifecycle ของอุปกรณ์ทั้งหมด ตั้งแต่การ Onboard เข้าระบบไปจนถึงการใช้งานและบำรุงรักษา

Device Management นั้นสามารถพูดคุยกับอุปกรณ์ IoT ผ่านโปรโตคอลหลากหลาย โดยเฉพาะ LWM2M ซึ่งช่วยให้นักพัฒนาเขียนต่อ กับ อุปกรณ์ได้ผ่านโมเดลที่มีลักษณะคล้ายๆ REST Model และ CoAP ซึ่งจะช่วยประหยัดแบตเตอรี่ของอุปกรณ์ได้มากกว่า HTTP ราว 8-10 เท่า และในการเขียนต่อ กับ แอปพลิเคชันซึ่งเป็นปลายทางอีกด้านหนึ่ง Pelion ได้เตรียม REST API และ SDK ในภาษา Java, Python, JavaScript และ .NET ไว้ให้พัฒนาแอปพลิเคชันกันได้โดยง่าย

Device Management ของ Pelion นี้รองรับการทำงานร่วมกับอาร์ดเวย์ที่หลากหลาย ไม่ว่าจะเป็น อุปกรณ์แบบ Bare metal (มีระบบเชื่อมต่อที่เรียกว่า Edge รองรับ) และการทำงานร่วมกับระบบปฏิบัติการทั้ง Mbed OS และ Linux

Data Management

เป้าประสงค์หลักของการจัดตั้งระบบ IoT นั้นคือการสร้างระบบจัดเก็บข้อมูลที่จะช่วยให้องค์กรสามารถ เรียกข้อมูลเหล่านั้นขึ้นมาวิเคราะห์เป็นความรู้ที่มีประโยชน์ต่อธุรกิจได้ แน่นอนว่า Pelion ย่อมไม่ลืมความสำคัญ ของส่วนนี้ จึงได้พัฒนาระบบจัดการข้อมูลครบวงจรที่จะช่วยตั้งแต่การจัดเก็บ นำข้อมูลมาใช้ตัดสินใจแบบ Real-time และรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูล โดยมีกลไกของรับการสเกลเต็มที่ ทำให้องค์กรไม่ ต้องกังวลว่าระบบจะทำงานได้เยี่ยงหากมีข้อมูลหรืออุปกรณ์ในเครือข่าย IoT เพิ่มมากขึ้นเมื่อเวลาผ่านไป

โซลูชันหลักของส่วนนี้ คือ ARM Treasure Data ซึ่งเป็นซอฟต์แวร์จัดการและวิเคราะห์ข้อมูลที่เชื่อม ต่อกับ Pelion ได้จบครบในตัวเดียว โดยมีเครื่องมือต่าง ๆ พร้อมให้เลือกใช้งาน เช่น ระบบ Predictive Analytics การสร้าง Customer View 360 องศาจากข้อมูลการใช้งาน การวิเคราะห์ข้อมูลเพื่อ Cross-sell และ Upsell และการสร้างระบบ Recommendation เป็นต้น ซึ่งโซลูชันนี้หลายอย่างคือรักษาไว้ให้ทำงานเพื่อ ประสิทธิภาพให้กับการทำงานในอุตสาหกรรมมากมาย เช่น อุตสาหกรรมค้าปลีก อุตสาหกรรมพลังงาน อุตสาหกรรมการผลิต และอุตสาหกรรมอื่น ๆ อีกมาก

แพลตฟอร์ม Pelion นั้นปัจจุบันได้มีการนำไปใช้งานกับระบบ IoT ทั้งใน projekต์ขนาดใหญ่และขนาด เล็ก เช่น ระบบ IoT ในการดูแลสัตว์น้ำผ่านเซ็นเซอร์รับข้อมูลจากเสียง ระบบตรวจสอบสถานะการทำงานของ เครื่องจักรในโรงงาน ระบบจัดการคลังสินค้า และระบบซึ่งเป็นส่วนประกอบของ Smart City เช่น ที่จอดรถ อัจฉริยะ และเสาไฟฟ้าที่เปิดปิดตามความเคลื่อนไหวของคน และสามารถควบคุมได้จากระบบส่วนกลาง เป็นต้น

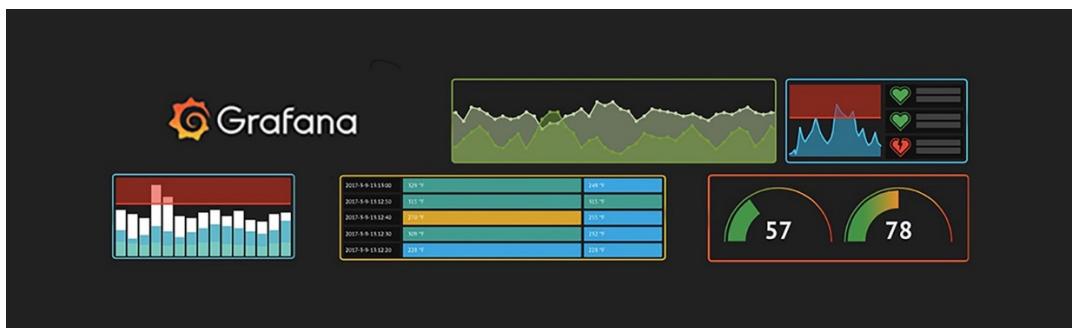
ท่านใดที่สนใจอยากศึกษาเกี่ยวกับ Pelion เพิ่มเติม สามารถเข้าไปอ่านเกี่ยวกับกรณีศึกษาการใช้งานใน อุตสาหกรรมได้ที่ <https://www.arm.com/products/iot/pelion-iot-platform> และหากต้องการข้อมูลเชิง เทคโนโลยีละเอียด สามารถอ่าน Document เต็มๆตามลิงก์นี้ <https://www.pelion.com/docs/>

สำหรับในประเทศไทย ARM ได้จับมือเป็นพาร์ทเนอร์กับ Advantech ในบริการให้บริการด้านต่าง ๆ ท่านที่ สนใจสามารถติดต่อเพื่อพูดคุยถึงโซลูชันและผลิตภัณฑ์เกี่ยวกับ IoT ของ ARM ได้ที่อีเมล์

- <https://developers.ascendcorp.com/ทำความรู้จักกับ-grafana-dashboard-1a5efe6d170a>

Grafana Dashboard

ทำความรู้จักกับ Grafana Dashboard



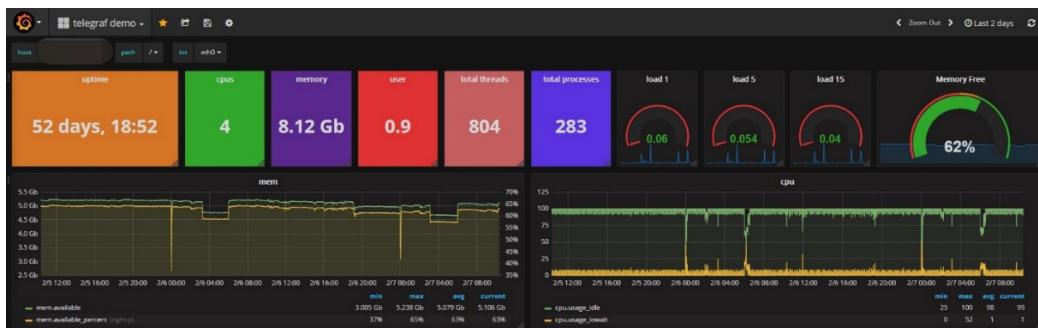
Grafana คือ open source Dashboard tool เรียกง่าย ๆ ก็คือเครื่องมือในการสร้าง Dashboard ฟรี นั้นเอง โดย Grafana จะทำงานร่วมกับ Datasource ต่าง ๆ เช่น Graphite, InfluxDB, OpenTSDB หรือ Elasticsearch ฯลฯ ช่วยให้ users สามารถสร้างและแก้ไข Dashboard ได้อย่างง่ายๆ ครอบคลุมรูปแบบกราฟ หลายประเภท

จุดเด่นของ Grafana

- เน้นการนำเสนอ Metrics ที่เฉพาะเจาะจง เช่น CPU, Memory หรือ I/O ในรูปแบบของกราฟ Time series
- มี Role-based access ในการจัดการ user ในการเข้าใช้งานให้ในตัว
- ความยืดหยุ่นในการใช้งาน มี option ให้เลือกใช้จำนวนมาก
- รองรับ datasource ที่หลากหลายและมี query editor ที่สำหรับ datasource นั้นๆ

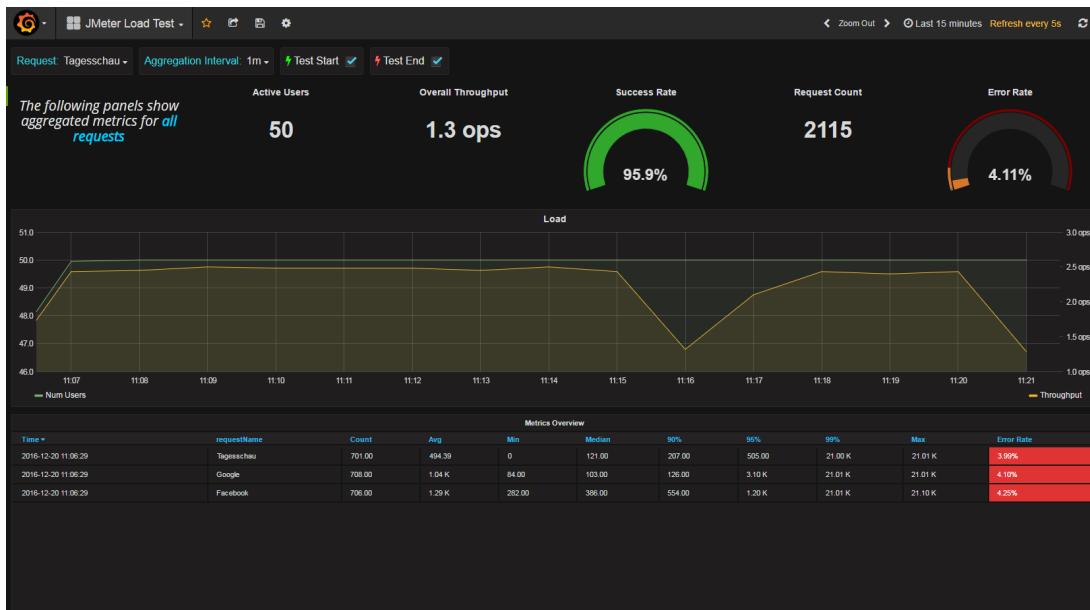
ตัวอย่างการใช้งาน Grafana Dashboard

- Monitoring Server ใช้งานร่วมกับ Influxdb และ Telegraf



เครดิต : <https://grafana.com/dashboards/1443>

- Monitoring Realtime result สำหรับ Jmeter ใน non-gui mode



เครดิต : <https://grafana.com/dashboards/1152>

การติดตั้ง Grafana Dashboard

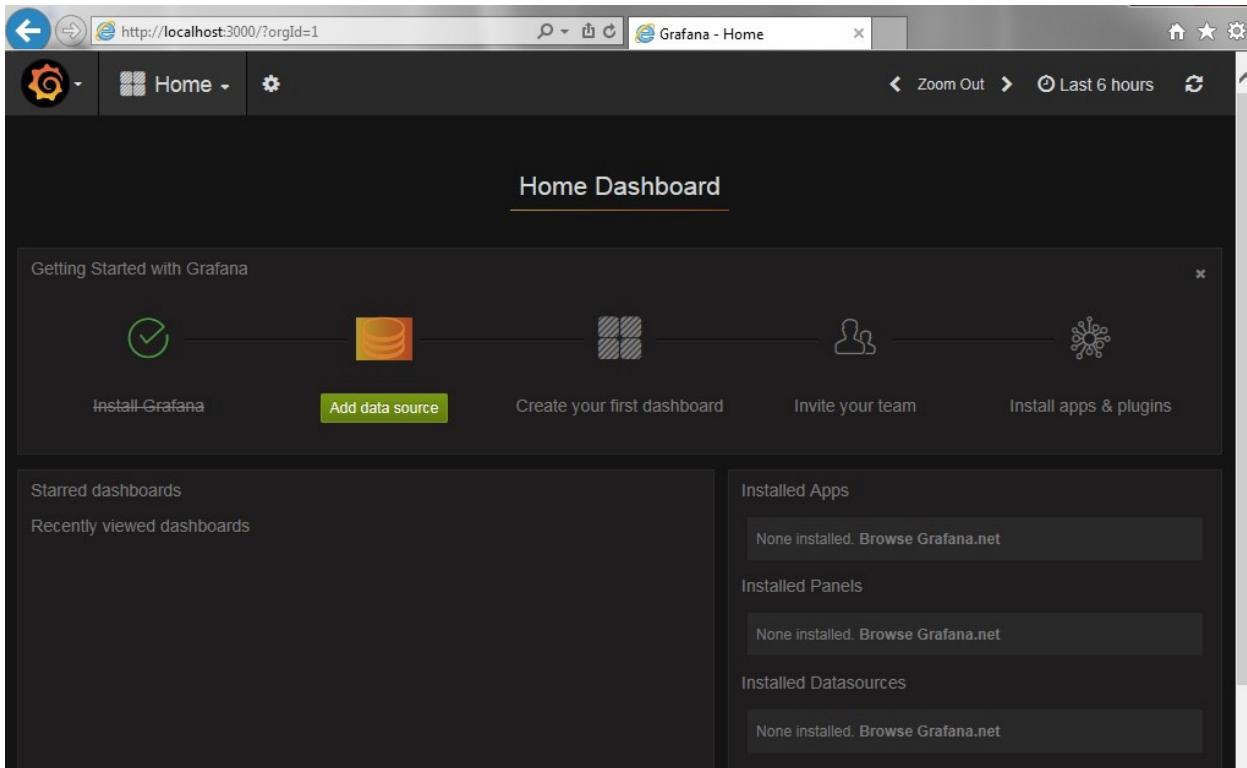
- ดาวน์โหลด ที่นี่ สำหรับ Windows (x64)
- สามารถใช้ grafana-server.exe เพื่อเริ่มใช้งานได้ทันที
- กรณีต้องการระบุ custom config ดูรายละเอียด ที่นี่

```
C:\Windows\system32\cmd.exe - bin\grafana-server.exe --config conf\custom.ini
+C:\grafana-4.4.1>bin\grafana-server.exe --config conf\custom.ini
+[32nINFO+{0m[07-28122:33:40] Starting Grafana           +[32mlogg
er+[0m=main +[32nversion+[0m=4.4.1 +[32mcommit+[0m=6a9f8caa4 +[32mcompiled+[0m=2
017-07-05T14:15:04+0700
+[32nINFO+{0m[07-28122:33:40] Config loaded from      +[32mlogg
er+[0m=settings +[32mfile+[0m=C:\grafana-4.4.1\conf\defaults.ini      +[32mlogg
er+[0m=settings +[32mfile+[0m=conf\custom.ini      +[32mlogg
+[32nINFO+{0m[07-28122:33:40] Path Home             +[32mlogg
er+[0m=settings +[32mpath+[0m=C:\grafana-4.4.1
+[32nINFO+{0m[07-28122:33:40] Path Data              +[32mlogg
er+[0m=settings +[32mpath+[0m=C:\grafana-4.4.1\data
+[32nINFO+{0m[07-28122:33:40] Path Logs              +[32mlogg
er+[0m=settings +[32mpath+[0m=C:\grafana-4.4.1\data\log
+[32nINFO+{0m[07-28122:33:40] Path Plugins           +[32mlogg
er+[0m=settings +[32mpath+[0m=C:\grafana-4.4.1\data\plugins
+[32nINFO+{0m[07-28122:33:40] Initializing DB          +[32mlogg
er+[0m=sqlstore +[32mdbtype+[0m.sqlite3
+[32nINFO+{0m[07-28122:33:40] Starting DB migration       +[32mlogg
er+[0m=migrator
+[32nINFO+{0m[07-28122:33:40] Executing migration        +[32mlogg
er+[0m=migrator +[32mid+[0m="copy data account to org"
+[32nINFO+{0m[07-28122:33:40] Skipping migration condition not fulfilled +[32mlo
gger+[0m=migrator +[32mid+[0m="copy data account to org"
+[32nINFO+{0m[07-28122:33:40] Executing migration        +[32mlogg
er+[0m=migrator +[32mid+[0m="copy data account_user to org_user"
+[32nINFO+{0m[07-28122:33:40] Skipping migration condition not fulfilled +[32mlo
gger+[0m=migrator +[32mid+[0m="copy data account_user to org_user"
+[32nINFO+{0m[07-28122:33:40] Starting plugin search         +[32mlogg
er+[0m=plugins
+[32nINFO+{0m[07-28122:33:41] Initializing Alerting         +[32mlogg
er+[0m=alerting_engine
+[32nINFO+{0m[07-28122:33:41] Initializing CleanUpService    +[32mlogg
er+[0m=cleanup
+[32nINFO+{0m[07-28122:33:41] Initializing Stream Manager
+[32nINFO+{0m[07-28122:33:41] Initializing HTTP Server          +[32mlogg
er+[0m=http_server +[32maddress+[0m=0.0.0.0:3000 +[32mprotocol+[0m=http +[32msub
Url+[0m= +[32msocket+[0m=
```

สำหรับ Mac (ผ่านHomebrew)

```
:~ brew update
:~ brew install grafana
:~ brew services start grafana
```

- เมื่อทำการติดตั้งและ start service เรียบร้อยแล้ว เริ่มต้นใช้งานโดย default port ของ grafana คือ 3000
- เข้าใช้งานโดย http://localhost:3000 และ user/password เริ่มต้นคือ admin/admin



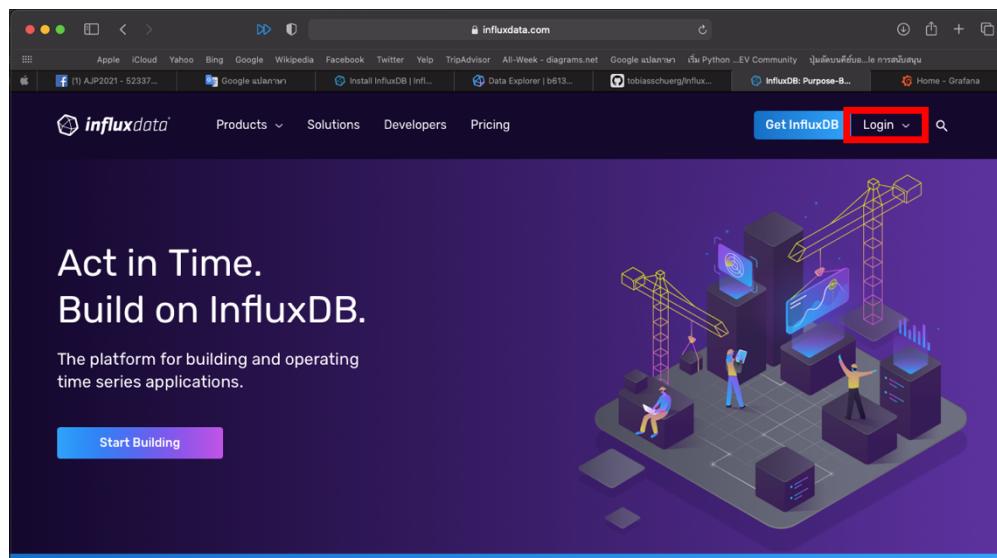
เท่านี้ก็สามารถเริ่มต้นใช้งาน Grafana Dashboard ได้แล้ว ครึ่งหน้าจะมาแนะนำการใช้งานร่วมกับ Influxdb ในการ Monitoring Server และ Monitoring realtime jmeter ~~~*

4/4. การทดสอบ

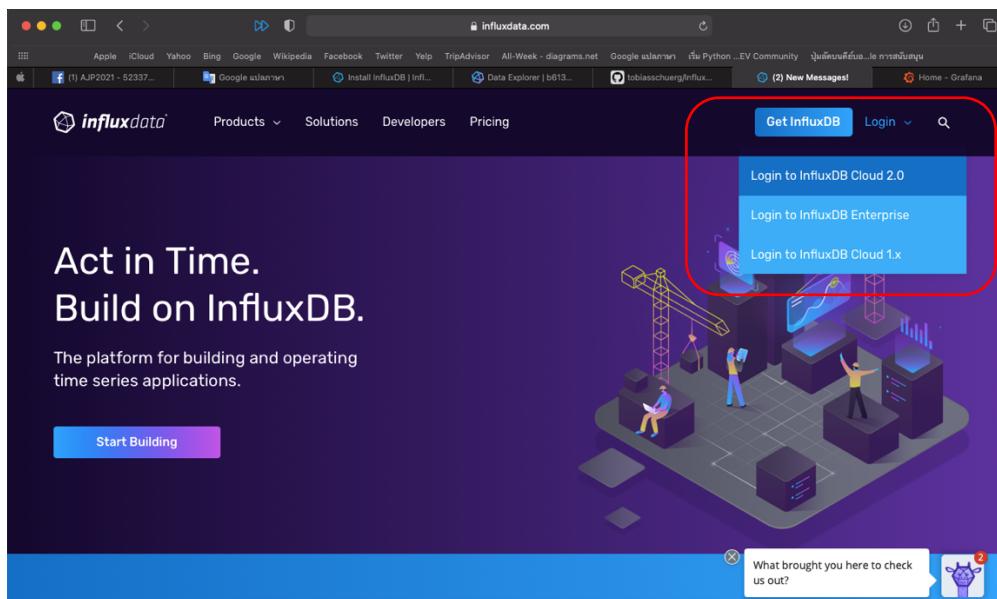
- ให้ทำการทดสอบและเขียนขั้นตอนการทดสอบ โดยใช้ ESP32 ส่งข้อมูลไปยัง MQTT Broker และใช้ Grafana .
ในการอนินิเตอร์ข้อมูล โดยปรับแก้การทดสอบจาก <https://gabrieltanner.org/blog/grafana-sensor-visualization>

Visualize Sensor data using Grafana and InfluxDB

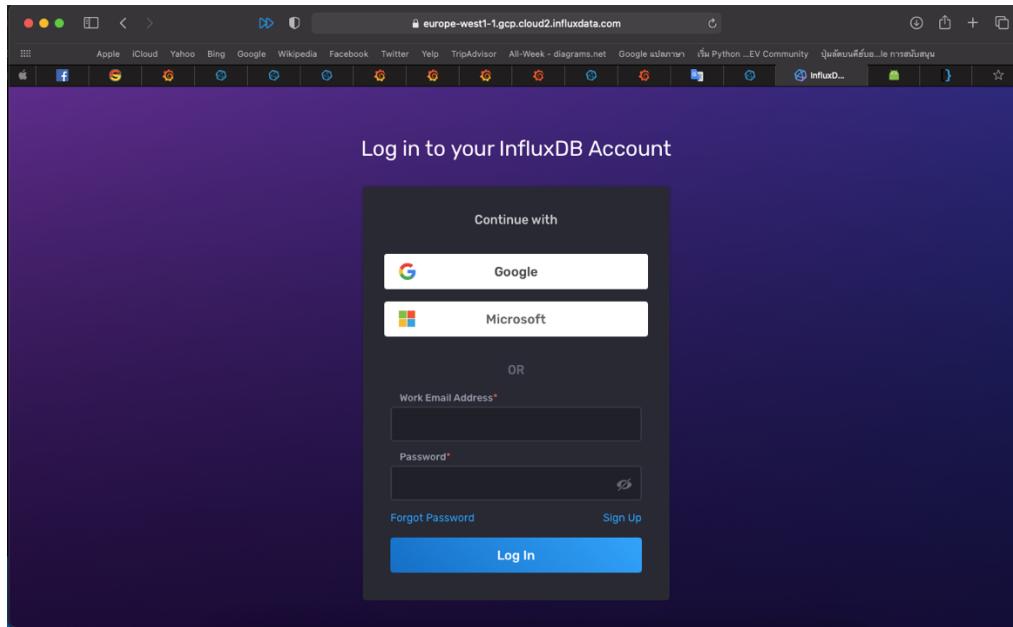
First Step: ทำการเข้า Search เข้า Web Influx



Next Step: Login to Influx Cloud 2.0



Next Step: ทำการ Login โดยใช้ Google และผูกใช้ Mail ของมหาวิทยาลัย



Next Step: ทำการเข้าไปที่ Data เพื่อสร้างพวาก Bucket และ Generate Token

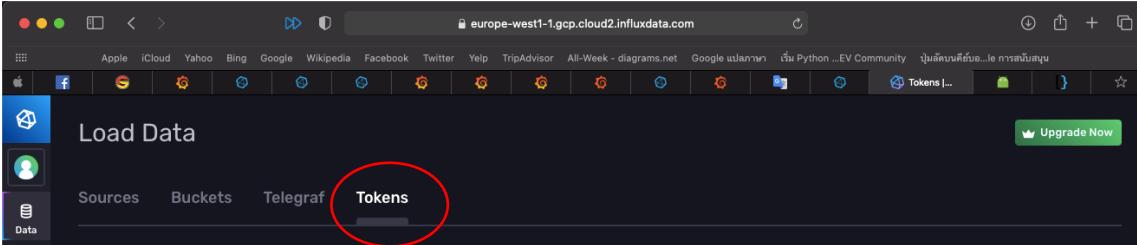
A screenshot of the InfluxDB "Getting Started" dashboard. The title bar reads "europe-west1-1.gcp.cloud2.influxdata.com". On the left, a sidebar menu includes "Data" (which is circled in red), "Explore", "Boards", "Tasks", "Alerts", and "Settings". The main area has three cards: "Load your data", "Build a dashboard", and "Set up alerting". To the right, there's a sidebar titled "Account" with "Logout" and "Recent Dashboards" sections, and a "Useful Links" section with links to "Documentation", "Community Forum", "Feature Requests", and "Report a bug". At the bottom right, it shows "Version (cbb5fd2)".

Next Step: เข้าไปที่ Buckets แล้วทำการ Create Buckets

The screenshot shows the InfluxDB interface with the 'Buckets' tab selected. The main content area displays several existing buckets: '_monitoring' (System Bucket, Retention: 7 days), '_tasks' (System Bucket, Retention: 3 days), and 'b6130381's Bucket' (Retention: 30 days, ID: a5e9d73b93373a31). A red circle highlights the '+ Create Bucket' button in the top right corner of the main content area.

The screenshot shows the InfluxDB interface with the 'Buckets' tab selected. The main content area displays the same buckets as the previous screenshot, but the 'b6130381's Bucket' entry is highlighted with a red circle. A modal at the bottom left asks 'Need more buckets?' with options to 'Get more buckets'. The right sidebar contains a 'What is a Bucket?' section with a purple background.

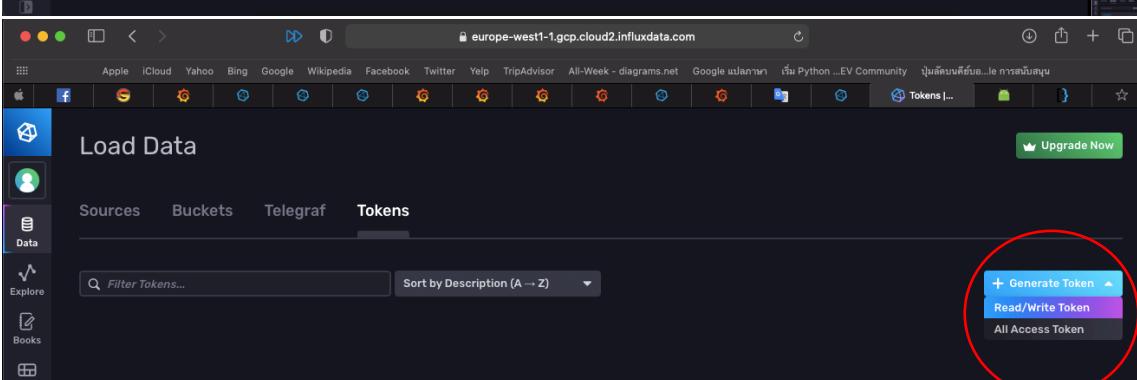
Next Step: จักนั้น เข้าไปที่ Tokens แล้ว Generate Token



The screenshot shows the InfluxDB interface with the 'Tokens' tab selected. A red circle highlights the 'Tokens' tab in the top navigation bar. Below the navigation bar, there is a search bar labeled 'Filter Tokens...' and a dropdown menu labeled 'Sort by Description (A → Z)'. On the right side, there is a green button labeled '+ Generate Token'.

Looks like there aren't any **Tokens**, why not generate one?

+ Generate Token



The screenshot shows the same InfluxDB interface, but now the '+ Generate Token' button is open, revealing a dropdown menu. The menu items are '+ Generate Token' (highlighted in blue), 'Read/Write Token' (highlighted in purple), and 'All Access Token'. A red circle highlights this dropdown menu.

Looks like there aren't any **Tokens**, why not generate one?

+ Generate Token

Read/Write Token

All Access Token

The screenshot displays two views of the InfluxDB interface. The top view shows the 'Tokens' creation dialog, where a new token named 'Token_B6130381' is being generated with 'Read' and 'Write' scopes set to 'Scoped'. Both scopes are applied to the bucket 'b6130381'. The bottom view shows the 'Tokens' list, where the newly created token is listed under the heading 'Token_B6130381'. It includes details such as 'Created at: 2021-06-23 21:06:02' and a status indicator showing it is 'Active'.

Next Step: จากนั้น Clone Git

Step on Window : แล้วแต่ไฟล์ Copy ไปไว้ที่ Directory Document และเอาไปไว้ใน Library ของ Arduino

Name	Date modified	Type	Size
Adafruit_Unified_Sensor	5/15/2021 1:56 AM	File folder	
ArduinoHttpClient	5/31/2021 11:14 PM	File folder	
ArduinoJson	5/31/2021 11:14 PM	File folder	
Blynk	5/18/2021 1:42 PM	File folder	
DHT_sensor_library	5/15/2021 1:56 AM	File folder	
DHT_sensor_library_for_ESPx	4/3/2021 10:13 PM	File folder	
DHT2pin	5/15/2021 1:56 AM	File folder	
DHT12	5/15/2021 1:56 AM	File folder	
DHT12_sensor_library	5/15/2021 1:56 AM	File folder	
DHTlib	5/15/2021 1:57 AM	File folder	
InfluxDB-Client-for-Arduino	4/1/2021 10:06 PM	File folder	
LedController	4/3/2021 4:30 PM	File folder	
PubSubClient	5/31/2021 8:19 PM	File folder	
ThingsBoard	5/31/2021 11:13 PM	File folder	
TM1638plus	4/3/2021 11:25 PM	File folder	
readme	3/23/2021 10:48 PM	Text Document	1 KB

Next Step: ทำการ Upload Code ลงบอร์ด Esp32

```
#include "DHTesp.h"
#include <WiFi.h>
#include <time.h>
#define Pin_DHT22 15 // D15
#include <InfluxDbClient.h> // https://github.com/tobiasschuerig/InfluxDB-Client-for-Arduino
#include <InfluxDbCloud.h>
DHTesp dht;
const char* ssid = "B-506";
const char* password = "600249546";
#define HOSTNAME "ESP32 Temperature Sensor"
#define LOCATION "1000 chem"
// InfluxDB server url. Don't use localhost, always server name or ip address.
// E.g. http://192.168.1.48:8086 (In InfluxDB 2 UI -> Load Data -> Client Libraries),
#define INFLUXDB_URL "https://europe-west1-1.gcp.cloud2.influxdata.com"
// InfluxDB 2 server or cloud API authentication token (Use: InfluxDB UI -> Load Data -> Tokens -> <select token>)
#define INFLUXDB_TOKEN "fU7BbJDM21OOsEST7LByA4NUJlfQvpNoO_sXy-C204UwrjMgBM-FWfv5UuEU2LuEMrqUJLnjxRkzrue3SRwfXg=="
// InfluxDB 2 organization id (Use: InfluxDB UI -> Settings -> Profile -> <name under tile> )
#define INFLUXDB_ORG "b6130381@g.sut.ac.th"
// InfluxDB 2 bucket name (Use: InfluxDB UI -> Load Data -> Buckets)
#define INFLUXDB_BUCKET "B6130381"
WiFiClient espClient;
// InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN,
InfluxDbCloud2CACert);
Point sensor("room_sensor");
void sendData()
{
    float h = dht.getHumidity();
    float t = dht.getTemperature();

    Serial.print("Temperature: ");
    Serial.print(t); Serial.print(" *C\n");
    Serial.print("Humidity: ");
    Serial.print(h); Serial.print(" %\n");
    // Store measured value into point
    sensor.clearFields();
}
```

```
// Data
sensor.addField("temperature", t);
sensor.addField("humidity", h);
// Print what are we exactly writing
Serial.print("Writing: ");
Serial.println(client.pointToLineProtocol(sensor));

// Write point
if (!client.writePoint(sensor))
{
    Serial.print("InfluxDB write failed: ");
    Serial.println(client.getLastErrorMessage());
}
}

void setup() {
Serial.begin(115200);
dht.setup(Pin_DHT22, DHTesp::DHT22);
delay(10);
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
} else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
}
```

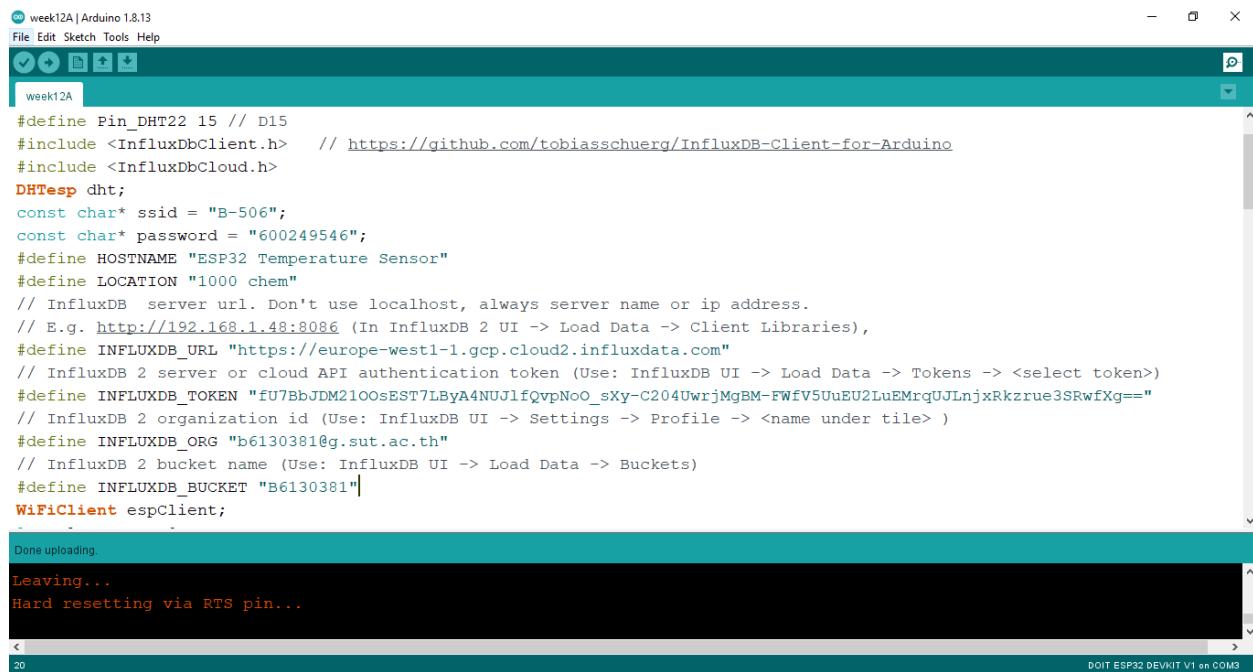
```

}

void loop() {
    float h = dht.getHumidity();
    float t = dht.getTemperature();

    Serial.print("Temperature: ");
    Serial.print(t); Serial.print(" *C\t");
    Serial.print("Humidity: ");
    Serial.print(h); Serial.print(" %\n");
    sendData();
    delay(1000);
}

```



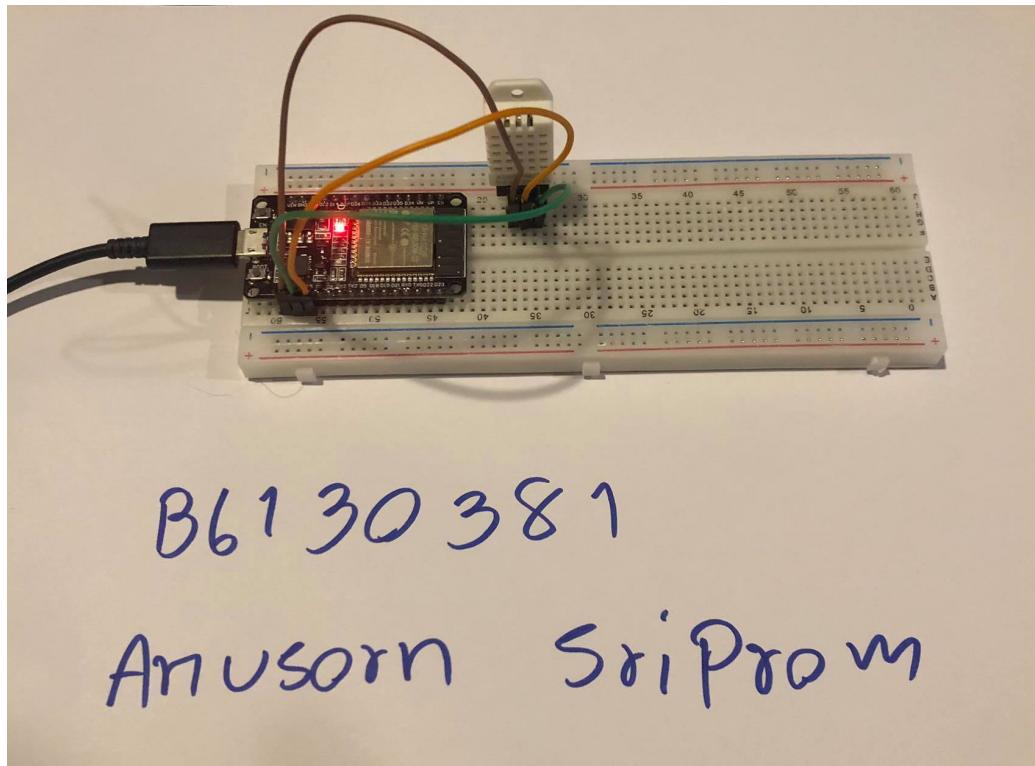
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** week12A | Arduino 1.8.13
- Menu Bar:** File Edit Sketch Tools Help
- Sketch Name:** week12A
- Code Area:**

```

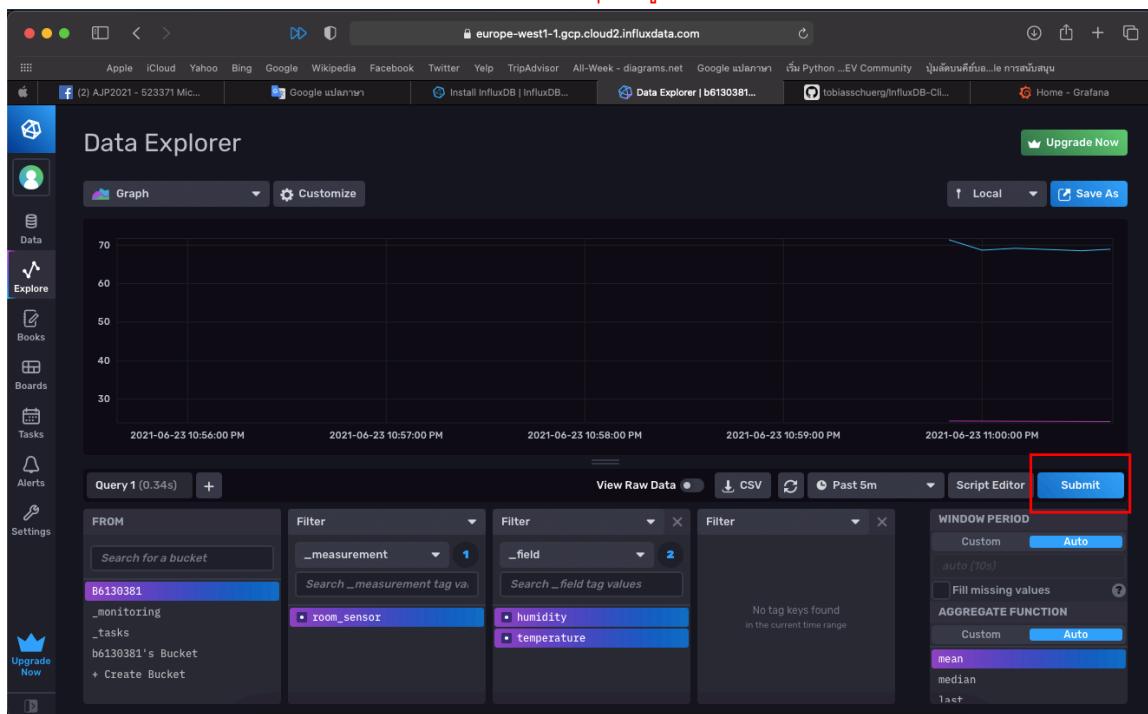
#define Pin_DHT22 15 // D15
#include <InfluxDbClient.h> // https://github.com/tobiasschuerig/InfluxDB-Client-for-Arduino
#include <InfluxDbCloud.h>
DHTesp dht;
const char* ssid = "B-506";
const char* password = "600249546";
#define HOSTNAME "ESP32 Temperature Sensor"
#define LOCATION "1000 chem"
// InfluxDB server url. Don't use localhost, always server name or ip address.
// E.g. http://192.168.1.48:8086 (In InfluxDB 2 UI -> Load Data -> Client Libraries),
#define INFLUXDB_URL "https://europe-west1-1.gcp.cloud2.influxdata.com"
// InfluxDB 2 server or cloud API authentication token (Use: InfluxDB UI -> Load Data -> Tokens -> <select token>)
#define INFLUXDB_TOKEN "fU7BbJDM21OosEST7LByA4NUJlfQvpNoO_sXy-C204UwrjMgBM-FWfv5UuEU2LuEMrqUJLnjxRkzrue3SRwfXg=="
// InfluxDB 2 organization id (Use: InfluxDB UI -> Settings -> Profile -> <name under tile> )
#define INFLUXDB_ORG "b6130381@g.sut.ac.th"
// InfluxDB 2 bucket name (Use: InfluxDB UI -> Load Data -> Buckets)
#define INFLUXDB_BUCKET "B6130381"
WiFiClient espClient;

```
- Serial Monitor:**
 - Message: Done uploading.
 - Message: Leaving...
 - Message: Hard resetting via RTS pin...
- Bottom Status Bar:** DOIT ESP32 DEVKIT V1 on COM3



Next Step: เมื่อ Upload Code ลงแล้ว

ให้เราเข้าไปเช็คที่ Bucket ที่เราสร้างขึ้นและเรกิ๊ตติก อุณหภูมิ&ความชื้นแล้ว Submit ก็จะได้กราฟขึ้นมา



Next Step: Link สำหรับ ติดตั้ง Grafana

<https://grafana.com/docs/grafana/latest/installation/>

จากนั้นทำการติดตั้ง Grafana ผ่าน HomeBrew

The screenshot shows the 'Installation' section of the Grafana documentation. The 'Install on macOS' link is highlighted with a red box. The right sidebar includes a 'Version' dropdown set to 'latest (v8.0.x)' and a 'Get Product Updates and News' sidebar with an 'Email*' input field.

Install Grafana

This section discusses the hardware and software requirements as well as the process of installing Grafana on different operating systems. This section has the following topics:

- Requirements
- Install on Debian or Ubuntu
- **Install on RPM-based Linux (CentOS, Fedora, OpenSuse, RedHat)**
- **Install on macOS**
- Install on Windows
- Run Docker image
- Deploy Grafana on Kubernetes

The screenshot shows the 'Install on macOS' page. It highlights the 'Install with Homebrew' section and the command 'brew update' followed by 'brew install grafana' in a red box. The right sidebar includes a 'Get Product Updates and News' sidebar with an 'Email*' input field.

Install on macOS

This page explains how to install Grafana and get the service running on your macOS.

Note on upgrading: While the process for upgrading Grafana is very similar to installing Grafana, there are some key backup steps you should perform. Before you perform an upgrade, read [Upgrading Grafana](#) for tips and guidance on updating an existing installation.

Install with Homebrew

Use [Homebrew](#) to install the most recent released version of Grafana using Homebrew package.

1. On the Homebrew homepage, search for Grafana. The last stable and released version is listed.
2. Open a terminal and enter:

```
brew update
brew install grafana
```

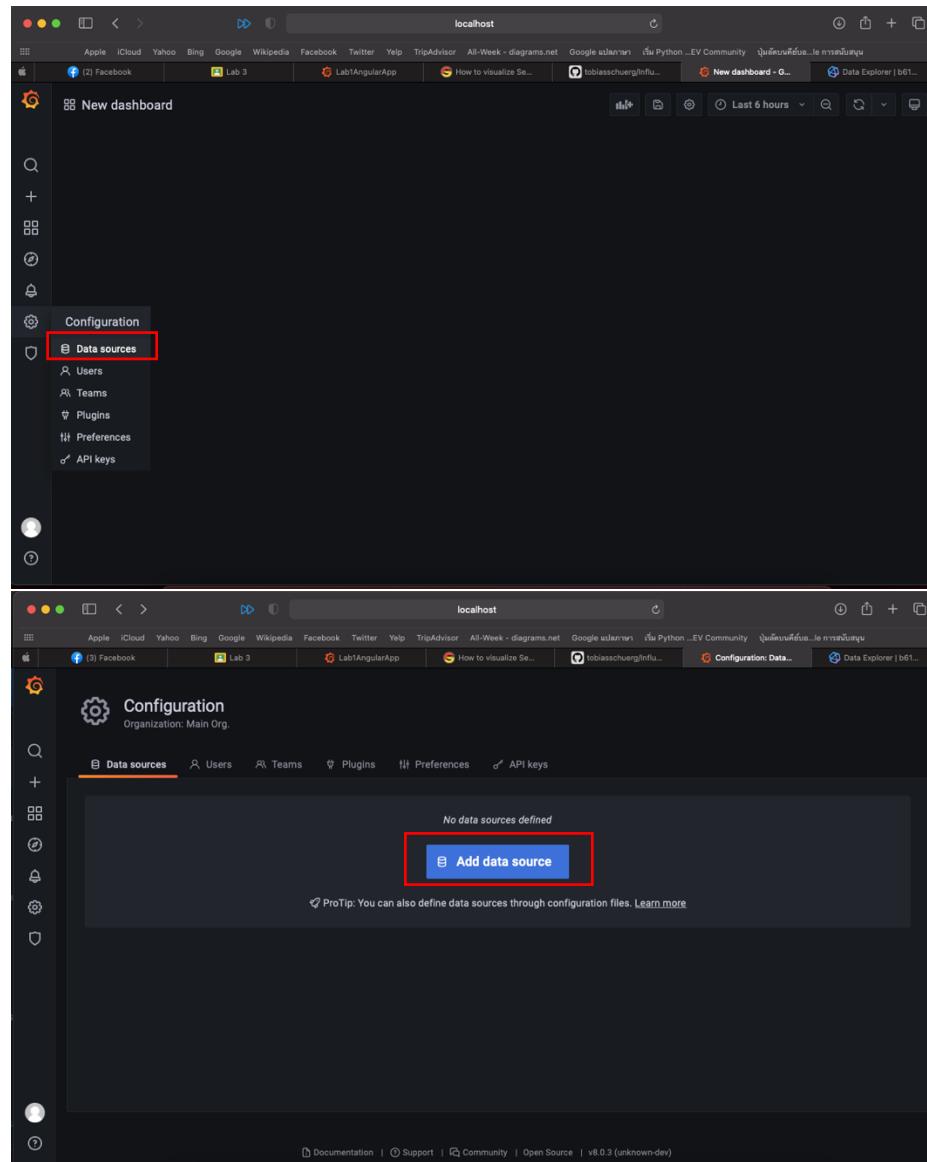
The brew page downloads and untars the files into `/usr/local/Cellar/grafana/version`.

3. Start Grafana using the command:

Next Step: ทำการเปิด Grafana ผ่าน Localhost:3000

และเข้า Setting Data Sources

จากนั้น Add Data Source



Next Step: ทำการ Add ตัว InfluxDB เข้า DashBoard

The screenshot shows the 'Add data source' screen in Grafana. Under the 'Time series databases' section, the 'InfluxDB' option is highlighted with a red box around it. A blue 'Select' button is located to the right of the InfluxDB entry. The 'Query Language' dropdown below is set to 'Flux', which is also highlighted with a red box.

Next Step: Setting ตัว InfluxDB ตามด้านล่าง

Url : <https://europe-west1-1.gcp.cloud2.influxdata.com>

The screenshot shows the 'Data Sources / InfluxDB-1' configuration screen. The 'Settings' tab is active. The 'Name' field is set to 'InfluxDB-1'. The 'Query Language' dropdown is set to 'Flux'. In the 'HTTP' section, the 'URL' field contains the value 'https://europe-west1-1.gcp.cloud2.influxdata.c...', which is also highlighted with a red box. Other settings like 'Access', 'Whitelisted Cookies', and 'Timeout' are visible below.

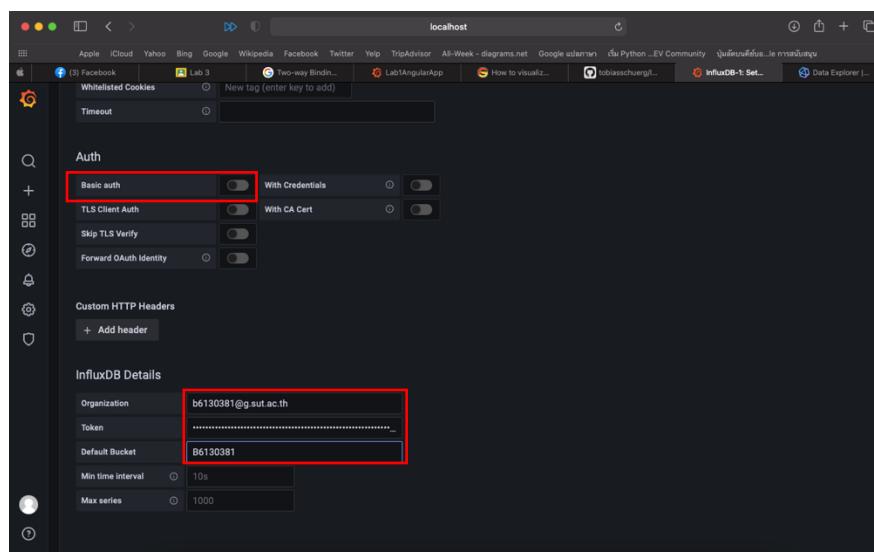
Setting :

Basic auth : ทำการ off ตัว authentication (ความปลอดภัยค้ำประกัน 2FA) แต่แลบปั้นเราไม่ได้ใช้ เพราะไม่ได้ต้องการสร้างความปลอดภัยอะไร

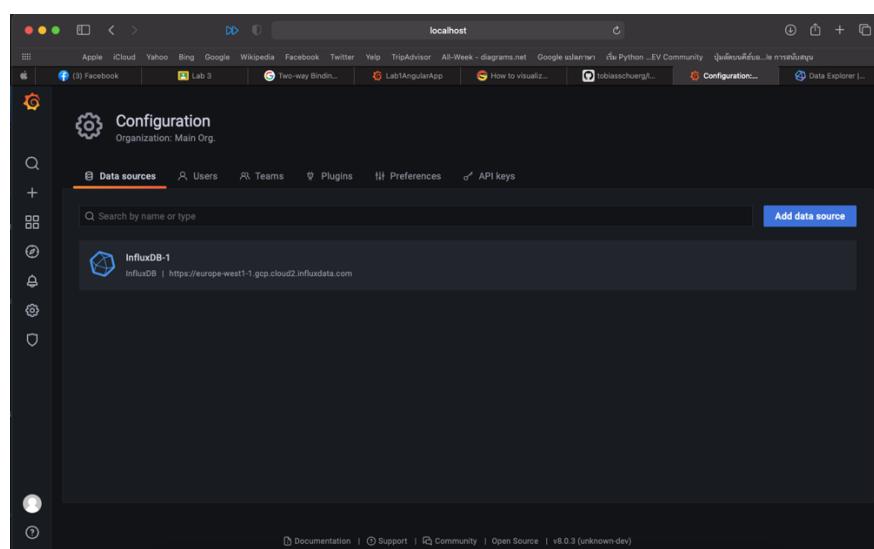
Organization : {ใช้ Mail ตัวเอง}

Token : (ได้จากการ Generate จากตัว Influx)

Bucket : (ในตัวเว็บ Influx ที่เราสร้างขึ้นมา)



จากนี้เราจะได้ตัว Data Source มา



Next Step: ทำการ Create Dashboard และ Add an empty panel

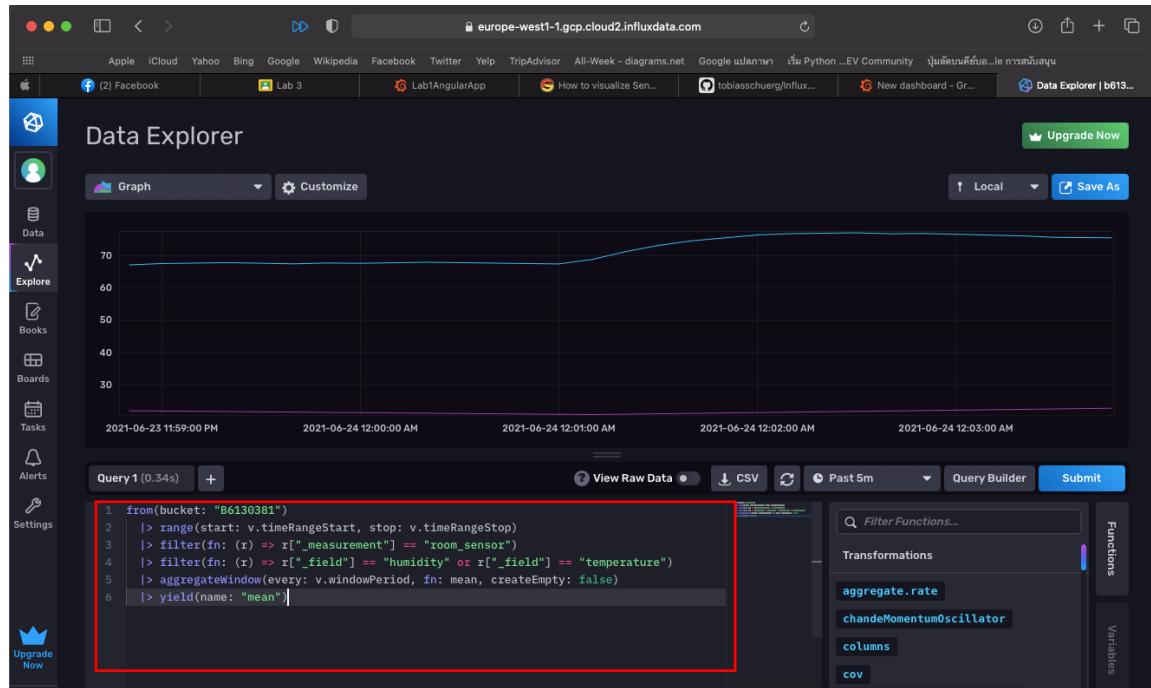
The image consists of two screenshots of the Grafana interface, one above the other.

The top screenshot shows the "Configuration" screen. On the left sidebar, there is a "Create" button with a plus sign, which is highlighted with a red box. Below it, there is a "Dashboard" button, also highlighted with a red box. The main area shows a search bar with the placeholder "Name or type" and a "Add data source" button. A single data source entry named "xDB-1" is listed under "Import".

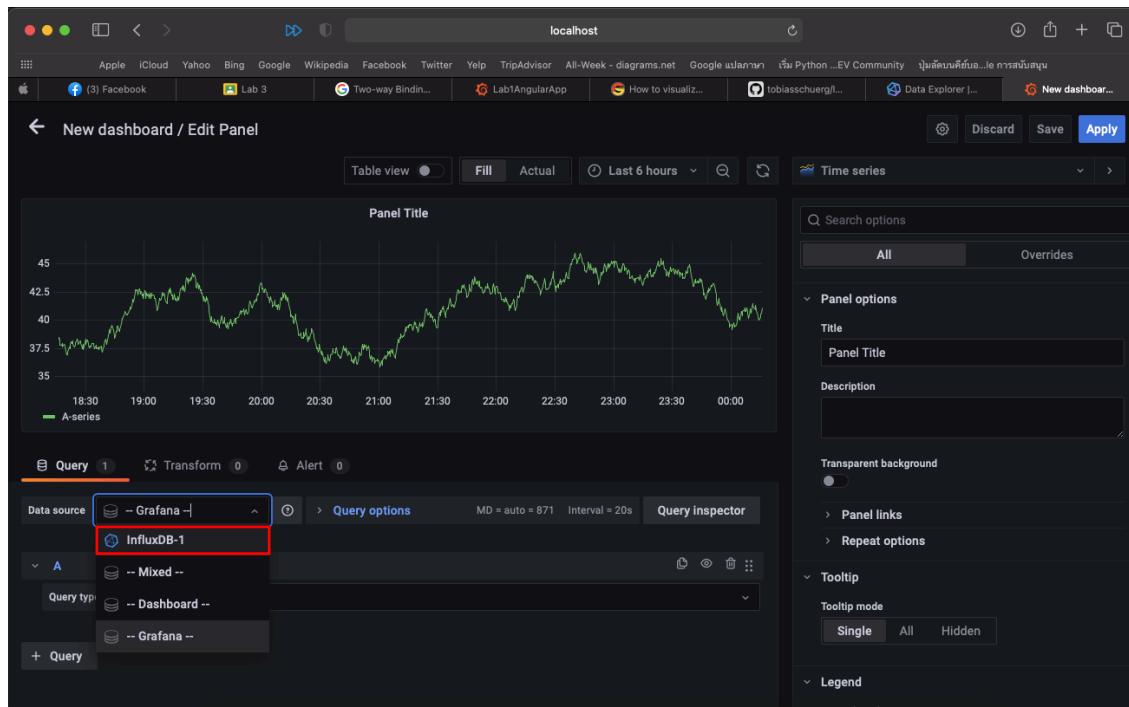
The bottom screenshot shows the "New dashboard" screen. It features a modal window titled "Add panel". Inside the modal, there are three options: "Add an empty panel" (highlighted with a red box), "Add a new row", and "Add a panel from the panel library".

Next Step: เข้าไปที่ Influx

Query Builder แล้วเราจะจะได้ การ Filter ข้อมูลตามภาพให้ Copy Code เพื่อที่จะนำไปใส่ Grafana Dashboard



Next Step: ทำการเข้ามาที่ Grafana และเลือก Data Source เป็น InfluxDB-1



Next Step: จักนั่งว่าง Code ที่ Copy มาจาก Influx และกด Apply

The screenshot shows two stacked windows of the Grafana interface.

Top Window: A "New dashboard / Edit Panel" window. It features a "Query" section containing the following InfluxDB query code, which is highlighted with a red box:

```

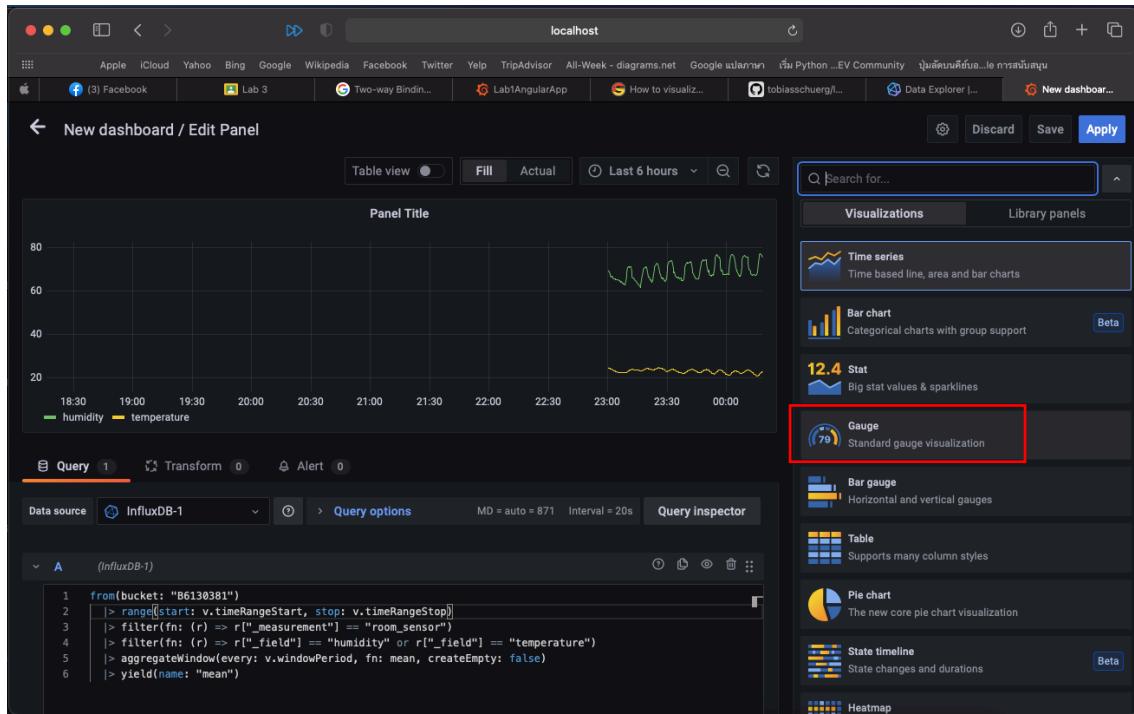
1 from(bucket: "86130381")
2 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3 |> filter(fn: (r) => r["_measurement"] == "room_sensor")
4 |> filter(fn: (r) => r["_field"] == "humidity" or r["_field"] == "temperature")
5 |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
6 |> yield(name: "mean")

```

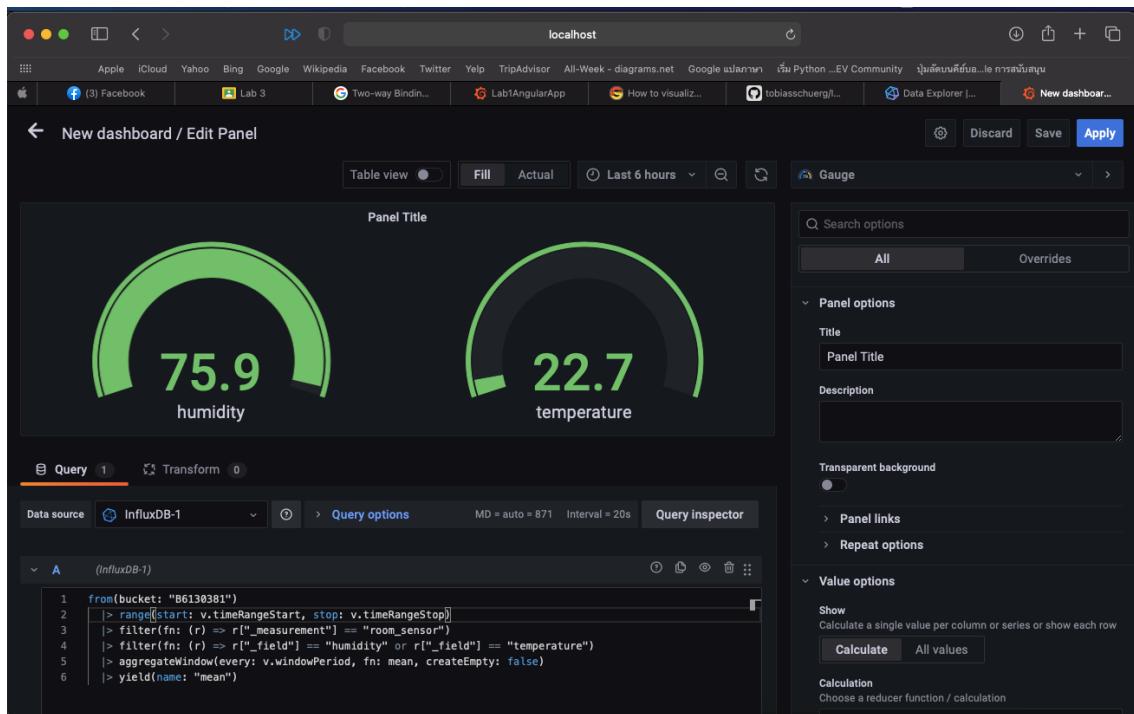
To the right of the query editor is a panel configuration sidebar. The "Apply" button at the top right of this sidebar is also highlighted with a red box.

Bottom Window: A "New dashboard" window displaying a time series chart. The chart has two data series: "temperature" (green line) and "humidity" (yellow line). The x-axis shows time from 19:00 to 00:00. The y-axis ranges from 20 to 80. The "humidity" series shows a sharp peak around 23:00, while the "temperature" series remains relatively flat.

Next Step: สร้าง Panel อีกอันและเลือก Visualization เป็น Gauge



แล้วจากนั้นก็ได้ตามนี้



เมื่อกด Apply ก็จะได้ตามภาพด้านล่างนี้

