

## Week12A - Visualize Sensor data using Grafana and InfluxDB

กลุ่มคำนวณ 5 รหัส B6238285 ชื่อ-สกุล นายประเมษฐ์ จิตละม่อม

### 1/4 การทำรายงาน

- เป็นการเก็บคะแนน 10 คะแนนตัดเกรด ให้ทำสุดความสามารถในเวลาที่มี
- ให้รับส่งโดยถือว่าโครงสร้างก่อนถือว่าเป็นต้นฉบับส่งที่หลังถือว่าลอกเพื่อนมา
- 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/>

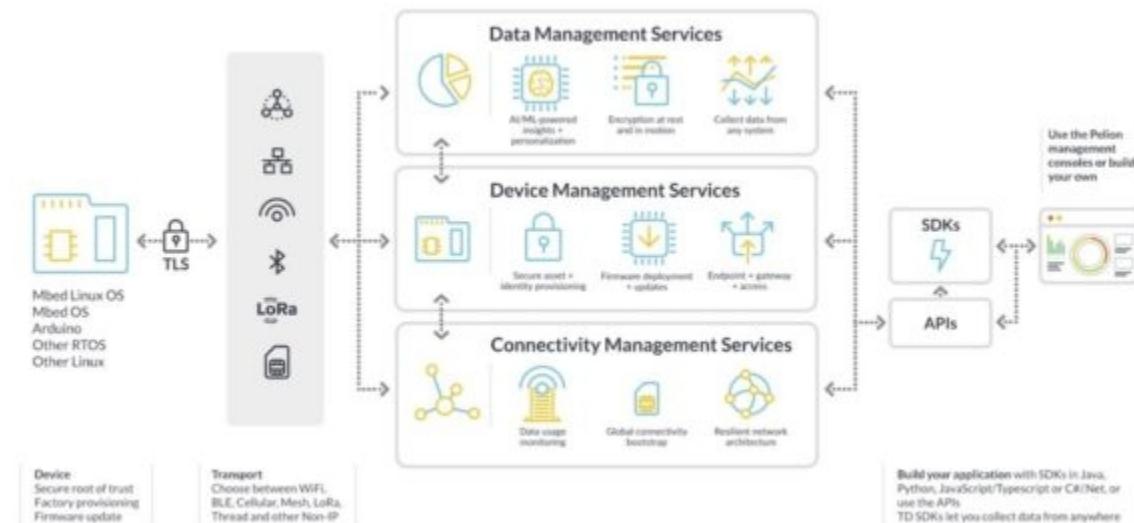
#### Arm-Pelion Full Stack IoT Platform

รู้จัก ARM Pelion แพลตฟอร์ม IoT จาก ARM จัดการทุกอย่างครบจบในที่เดียว

เมื่อวันที่ 30 กรกฎาคมที่ผ่านมา ทาง ARM ได้จัด IoT Workshop ขึ้นเพื่อนำเสนอและมุ่งด้านของการนำเทคโนโลยี IoT ไปใช้ในธุรกิจ ทีมงาน TechTalkThai ได้เข้าไปร่วมงาน และทำความรู้จักกับ ARM Pelion IoT Platform ซึ่งอย่างจะมาเล่าให้ผู้อ่านฟังกันคร่าวๆว่า เจ้าแพลตฟอร์ม IoT นี้มีความสามารถอย่างไร และเหมาะสมกับธุรกิจแบบไหนบ้าง

IoT นั้นเป็นหนึ่งในเทคโนโลยีที่หลายองค์กรยกให้เป็นยุทธศาสตร์ในปี 2019 จากความสามารถในการรวบรวมข้อมูล ซึ่งพบว่า เป็นวัตถุดีในการทำธุรกิจที่ขาดไม่ได้เลยในยุคปัจจุบัน จนถึงตอนนี้ หลายองค์กรอาจเริ่มต้นกับ IoT กันบ้างแล้ว แต่ความท้าทายใหม่ที่ ธุรกิจมักจะเผชิญกับคือการจัดการและส่งผลกระทบ IoT ให้เข้ากันเก็บข้อมูลได้จริงเต็มประสิทธิภาพ ปลอดภัย และมีระบบจัดการที่ดี ดังนั้น จึงมีการพัฒนาแพลตฟอร์ม IoT ขึ้นเพื่อช่วยธุรกิจในการแก้ปัญหา

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 ส่วนจะทำงานร่วมกันภายใต้ระบบรักษาความปลอดภัย ซึ่งเป็นหลักสำคัญที่สุด ในการพัฒนาผลิตภัณฑ์ทุกด้านของ Arm



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

Pelion จะช่วยให้ธุรกิจจัดการกับเครื่องข่าย อุปกรณ์ในเครือข่าย และข้อมูลที่เก็บมาได้ง่ายขึ้น โดยสามารถทำงานร่วมกับอุปกรณ์ ระบบเครือข่าย คลาวด์ และช่องมูลได้หลากหลายรูปแบบ อีกทั้งยังมีความสามารถปลอดภัย และสามารถช่วยในการนำข้อมูลไปวิเคราะห์และแสดงผลเบื้องต้นได้ด้วย

รู้จักแพลตฟอร์มนี้เป็นคร่าวๆแล้ว ลองมาเจาะลึกกันว่าส่วนประกอบทั้ง 3 ส่วน วันได้แก่ Connectivity Management, Device Management, และ Data Management นั้นประกอบไปด้วยอะไร และมีจุดเด่นอย่างไรบ้าง

## Connectivity Management

การเชื่อมต่อในเครือข่าย IoT นั้นมีอยู่หลายรูปแบบ และมีรายละเอียดปลีกย่อยที่ธุรกิจจะต้องจัดการอยู่พอสมควร Pelion จะช่วยให้องค์กรสามารถจัดการการเชื่อมต่อได้อย่างมีประสิทธิภาพ ปลอดภัย และพร้อมต่อการสแกลเครือข่ายขึ้นไปถึงระดับโลก โดย Connectivity Management ของ Pelion มีความสามารถที่น่าสนใจ ดังนี้

## Global Cellular

Pelion จะช่วยให้อุปกรณ์ IoT สามารถเชื่อมต่อผ่านเครือข่ายได้ไม่ว่าอุปกรณ์นั้นจะอยู่ที่ใดในโลก ผ่านเวนเดอร์เพียงเจ้าเดียว 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 โดยโปรโตคอลที่ Pelion รองรับนั้นได้แก่ MQTT(s), HTTPS, และ Sockets

## ใช้ได้ทั้ง eSIM และชิมแบบปกติ

ธุรกิจสามารถถังผลิตอุปกรณ์ที่มีระบบ eSIM ผ่าน ARM ได้ตามต้องการ โดย eSIM ที่ติดมากับอุปกรณ์นั้นจะรองรับการเชื่อมต่อกับเครือข่ายกว่า 600 เครือข่ายทั่วโลก และหากต้องการเปลี่ยนเครือข่าย ก็สามารถถอดค่าใหม่ได้ภายในหลัง และในส่วนของชิมแบบปกติเอง Pelion ก็ให้บริการซึ่งการติดในทุกขนาด อีกทั้งยังมีแผนที่จะพัฒนาไปจนถึง iSIM ที่มีขนาดเล็กกว่า eSIM มากด้วย

## Network Infrastructure

Pelion ได้พัฒนาโครงสร้างพื้นฐานของเครือข่ายให้สามารถทำงานร่วมกับผู้ให้บริการเครือข่ายทั่วโลกได้อย่างมีประสิทธิภาพ สูงสุด โดยมีทั้งความเสถียร ยืดหยุ่น และเป็นไปตามกฎข้อบังคับด้านข้อมูลของแต่ละประเทศ ในการใช้ 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 ก็จะมีโมดูลในการจัดการเรื่องต่างๆให้อย่างครบถ้วน ตั้งแต่เรื่องการอัปเดท Firmware ซึ่งไม่ง่ายเลยหากมีอุปกรณ์ที่หลากหลายและมีจำนวนที่มากในเครือข่าย, Access Management ซึ่งจะช่วยจำกัดการเข้าถึงอุปกรณ์และการควบคุมแต่ละส่วน, Connector ซึ่งจัดการการเชื่อมต่อ กับอุปกรณ์หลากหลายประเภท การออก Certificate เข้าใช้ระบบ การซื้นทะเบียนอุปกรณ์แต่ละตัว และการเก็บสถิติ, Device Directory ซึ่งช่วยในการแบ่งกลุ่ม ค้นหา เรียกดู และเข้าถึงสถานะของอุปกรณ์แต่ละตัว, ไปจนถึงการรักษาความปลอดภัยในการส่งต่อรหัสผ่านเครือข่าย Wifi

โดยทั้งหมดนี้จะเห็นได้ว่าเป็นการจัดการ Lifecycle ของอุปกรณ์ทั้งหมด ตั้งแต่การ Onboard เข้าระบบ ไปจนถึงการใช้งานและบำรุงรักษา

Device Management นั้นสามารถพูดคุยกับอุปกรณ์ IoT ผ่านโปรโตคอลหลากหลาย โดยเฉพาะ LwM2M ซึ่งช่วยให้นักพัฒนา เชื่อมต่อ กับอุปกรณ์ได้ผ่านไมเดลที่มีลักษณะคล้ายๆ REST Model และ CoAP ซึ่งจะช่วยประหยัดแบตเตอรี่ของอุปกรณ์ได้มากกว่า HTTP ราว 8-10 เท่า และในการเชื่อมต่อ กับแอปพลิเคชันซึ่งเป็นปลายทางอีกด้านหนึ่ง Pelion ก็ได้เตรียม REST API และ SDK ในภาษา Java, Python, JavaScript และ .NET ไว้ให้พัฒนาแอปพลิเคชันนี้ได้โดยง่าย

Device Management ของ Pelion นี้รองรับการทำงานร่วมกับชาร์ดแวร์ที่หลากหลาย ไม่ว่าจะเป็นอุปกรณ์แบบ Bare metal (มีระบบเชื่อมต่อที่เรียกว่า Edge รองรับ) และการทำงานร่วมกับระบบปฏิบัติการทั้ง Mbed OS และ Linux

## Data Management

เป้าประสงค์หลักของการจัดตั้งระบบ IOT นั้นคือการสร้างระบบจัดเก็บข้อมูลที่จะช่วยให้องค์กรสามารถเรียกข้อมูลเหล่านี้ ขึ้นมาวิเคราะห์เป็นความรู้ที่มีประโยชน์ต่อธุรกิจได้ แน่นอนว่า Pelion ย่อมไม่เสียความสำคัญของส่วนนี้ จึงได้พัฒนาระบบจัดการข้อมูลครบวงจรที่จะช่วยตั้งแต่การจัดเก็บ นำข้อมูลมาใช้ตัดสินใจแบบ Real-time และรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูล โดยมีกลไกการอัปเดตตัวอย่างต่อเนื่องที่ทำให้องค์กรไม่ต้องกังวลว่าระบบจะทำงานได้慢ลงหากมีข้อมูลหรืออุปกรณ์ใหม่ๆ เข้ามายังระบบ ไม่ว่าจะเป็นการซื้อขาย หรือการแนะนำสินค้าใหม่ๆ ตามความต้องการของผู้ใช้งาน

โซลูชันหลักของส่วนนี้ คือ ARM Treasure Data ซึ่งเป็นซอฟต์แวร์จัดการและวิเคราะห์ข้อมูลที่เริ่มต้นมาจาก Pelion ได้จบครบในตัวเดียว โดยมีเครื่องมือต่างๆพร้อมให้เลือกใช้งาน เช่น ระบบ Predictive Analytics การสร้าง Customer View 360 องศาจากข้อมูลการใช้งาน การวิเคราะห์ข้อมูลเพื่อ Cross-sell และ Upsell และการสร้างระบบ Recommendation เป็นต้น ซึ่งโซลูชันนี้มีหลายแพลตฟอร์ม เช่น AWS, Google Cloud, Microsoft Azure และ IBM Cloud เป็นต้น

องค์กรก็ได้นำไปใช้งานเพิ่มประสิทธิภาพให้กับการทำงานในอุตสาหกรรมมากมาย เช่น อุตสาหกรรมค้าปลีก อุตสาหกรรม พลังงาน อุตสาหกรรมการผลิต และอุตสาหกรรมอื่นๆอีกมาก

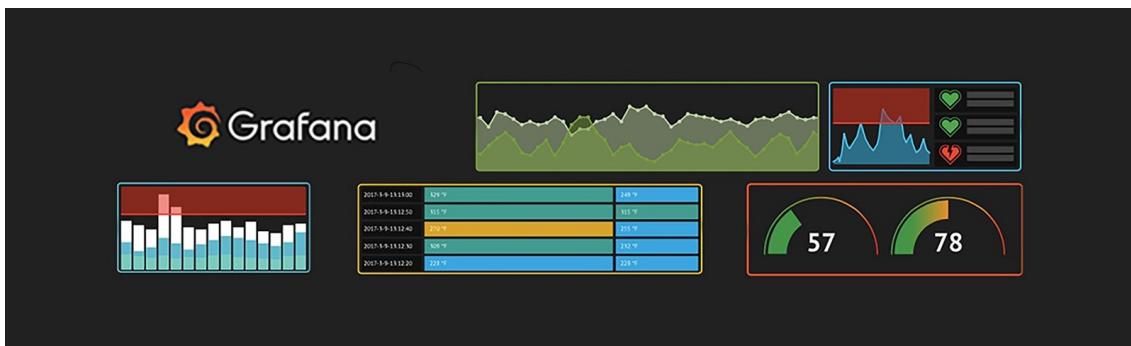
แพลตฟอร์ม Pelion นั้นปัจจุบันได้มีการนำไปใช้งานกับระบบ IoT ทั้งในプロジェクトขนาดใหญ่และขนาดเล็ก เช่น ระบบ IoT ในการดูแลสัตว์น้ำฝ่ายเซ็นเซอร์รับข้อมูลจากเสียง ระบบตรวจสอบสถานะการทำงานของเครื่องจักรในโรงงาน ระบบจัดการคลังสินค้า และระบบซึ่งเป็นส่วนประกอบของ Smart City เช่น ที่จอดรถอัจฉริยะ และเสาไฟฟ้าที่เปิดปิดตามความเคลื่อนไหวของคน และสามารถควบคุมได้จากระบบส่วนกลาง เป็นต้น

ท่านใดที่สนใจศึกษาเกี่ยวกับ Pelion เพิ่มเติม สามารถเข้าไปอ่านเกี่ยวกับกรณีศึกษาการใช้งานในอุตสาหกรรมได้ที่ <https://www.arm.com/products/iot/pelion-iot-platform> และหากต้องการข้อมูลเชิงเทคนิคโดยละเอียด สามารถอ่าน Document เดิมๆตามลิงก์นี้ <https://www.pelion.com/docs/>

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

## Grafana Dashboard

ทำความรู้จักกับ Grafana Dashboard



Grafana คือ open source Dashboard tool เรียกง่าย ๆ ก็คือเครื่องมือในการสร้าง Dashboard ฟรี นั่นเอง

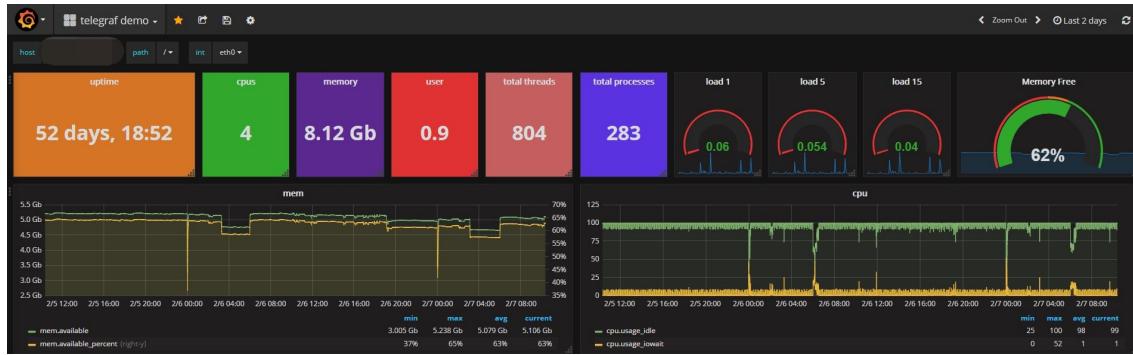
โดย Grafana จะทำงานร่วมกับ Datasource ต่าง ๆ เช่น Graphite, InfluxDB, OpenTSDB หรือ Elasticsearch ฯลฯ ช่วยให้ users สามารถสร้างและแก้ไข Dashboard ได้อย่างง่ายๆ ครอบคลุมรูปแบบกราฟหลากหลายประเภท

จุดเด่นของ Grafana

- แห้งการนำเสนอ 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 ที่ๆ

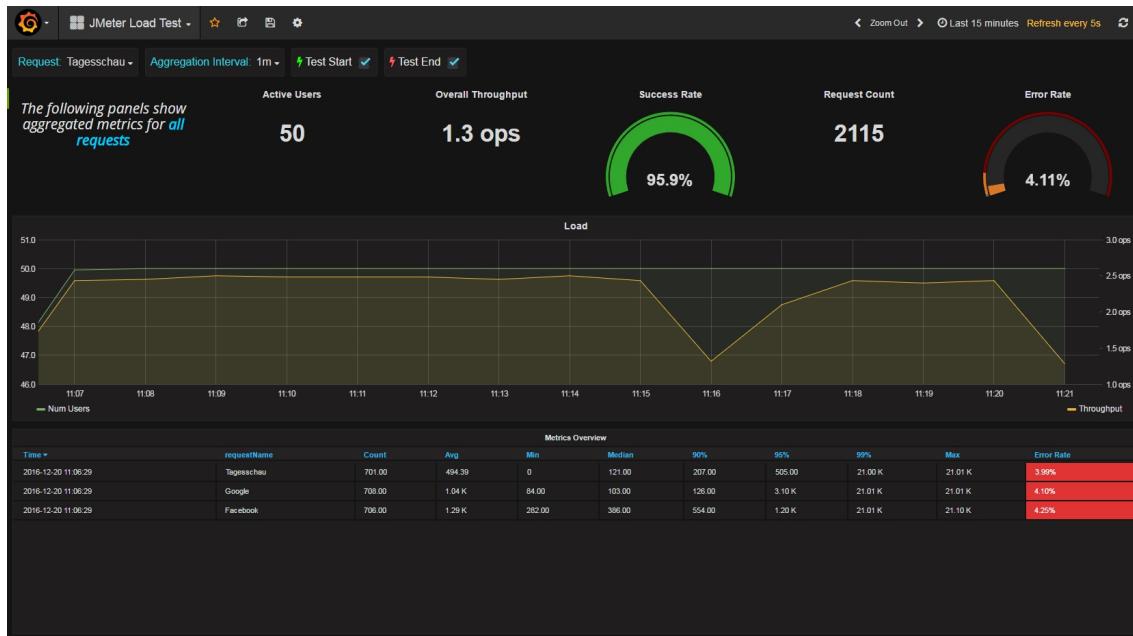
## ตัวอย่างการใช้งาน Grafana Dashboard

- Monitoring Server ใช้งานร่วมกับ Influxdb และ Telegraf



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

- Monitoring Realtime result สำหรับ Jmeter ใน non-gui mode



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

## การติดตั้ง Grafana Dashboard

- ดาวน์โหลด [ไฟล์](#)

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:\INFO+[0m[07-28 12:33:40] Starting Grafana +[32nlogg
er+[0m=main +[32nversion+[0n=4.4.1 +[32ncommit+[0n=6a9f8caa4 +[32ncompiled+[0m=2
017-07-05T14:15:04+0700
+[32nINFO+[0m[07-28 12:33:40] Config loaded from +[32nlogg
er+[0m=settings +[32nfile+[0m=C:\\grafana-4.4.1\\conf\\defaults.ini +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Config loaded from +[32nlogg
er+[0m=settings +[32nfile+[0m=conf\\custom.ini +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Path Home +[32nlogg
er+[0m=settings +[32npath+[0m=C:\\grafana-4.4.1 +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Path Data +[32nlogg
er+[0m=settings +[32npath+[0m=C:\\grafana-4.4.1\\data +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Path Logs +[32nlogg
er+[0m=settings +[32npath+[0m=C:\\grafana-4.4.1\\data\\log +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Path Plugins +[32nlogg
er+[0m=settings +[32npath+[0m=C:\\grafana-4.4.1\\data\\plugins +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Initializing DB +[32nlogg
er+[0m=sqldstore +[32nsubtype+[0m.sqlite3 +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Starting DB migration +[32nlogg
er+[0m=migrator +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Executing migration +[32nlogg
er+[0m=migrator +[32nid+[0m="copy data account to org" +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Skipping migration condition not fulfilled +[32nlogg
gger+[0m=migrator +[32nid+[0m="copy data account to org" +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Executing migration +[32nlogg
er+[0m=migrator +[32nid+[0m="copy data account_user to org_user" +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Skipping migration condition not fulfilled +[32nlogg
gger+[0m=migrator +[32nid+[0m="copy data account_user to org_user" +[32nlogg
+[32nINFO+[0m[07-28 12:33:40] Starting plugin search +[32nlogg
er+[0m=plugins +[32nlogg
+[32nINFO+[0m[07-28 12:33:41] Initializing Alerting +[32nlogg
er+[0m=alerting.engine +[32nlogg
+[32nINFO+[0m[07-28 12:33:41] Initializing CleanUpService +[32nlogg
er+[0m=cleanup +[32nlogg
+[32nINFO+[0m[07-28 12:33:41] Initializing Stream Manager +[32nlogg
+[32nINFO+[0m[07-28 12:33:41] Initializing HTTP Server +[32nlogg
er+[0m=http_server +[32naddress+[0m=0.0.0.0:3000 +[32nprotocol+[0m=http +[32nsub
Url+[0m= +[32nsocket+[0m=
```

สำหรับ Mac (Via [Homebrew](#))

```
:~ 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

The screenshot shows the Grafana Home Dashboard. At the top, there's a navigation bar with icons for back, forward, search, and refresh, followed by the title "Grafana - Home". Below the title, there are buttons for "Home", "Settings", "Zoom Out", "Last 6 hours", and a refresh icon. The main header is "Home Dashboard". A "Getting Started with Grafana" section features five cards: "Install Grafana" (green checkmark icon), "Add data source" (orange database icon, highlighted in green), "Create your first dashboard" (grey grid icon), "Invite your team" (person icon), and "Install apps & plugins" (flower icon). To the left, a sidebar lists "Starred dashboards" and "Recently viewed dashboards". To the right, sections for "Installed Apps", "Installed Panels", and "Installed Datasources" all show "None installed. Browse Grafana.net".

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

credit : <https://logz.io/blog/grafana-vs-kibana/>

## 4/4. การทดสอบ

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

1. เราจะเข้าไปที่เว็บไซต์ <https://www.influxdata.com> เพื่อทำการสร้างตัว Data เพื่อนำไปแสดงบน Dashboard ด้วย influxDB

```

1 temp < from(bucket: "Pizza-making")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r._measurement == "iot-oven")
4   |> filter(fn: (r) => r._field == "temperature")
5   |> aggregateWindow(every: 60s, fn: mean)
6
7 tempS1 = temp |> filter(fn: (r) => r.sensor == "S1")
8 tempS2 = temp |> filter(fn: (r) => r.sensor == "S2")
9
10 temp_join = join(tables: [s1: tempS1, s2: tempS2], on: ["_time"], method: "inner")
11

```

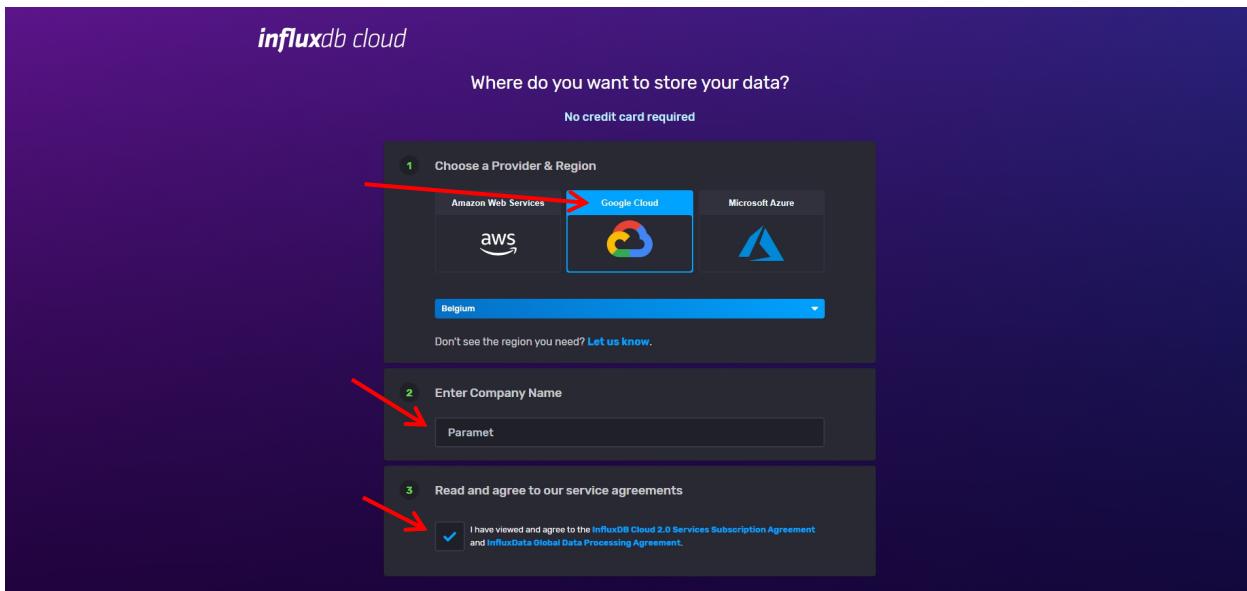
2. เราจะทำการ Login ด้วย Login to influxDB Cloud 2.0

```

1 temp < from(bucket: "Pizza-making")
2   |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3   |> filter(fn: (r) => r._measurement == "iot-oven")
4   |> filter(fn: (r) => r._field == "temperature")
5   |> aggregateWindow(every: 60s, fn: mean)
6
7 tempS1 = temp |> filter(fn: (r) => r.sensor == "S1")
8 tempS2 = temp |> filter(fn: (r) => r.sensor == "S2")
9
10 temp_join = join(tables: [s1: tempS1, s2: tempS2], on: ["_time"], method: "inner")
11

```

3.หลังจาก Login แล้วจะได้หน้าต่างแบบนี้ ทำการเลือกเป็น Google Cloud และตั้งชื่อ หลังจากนั้นกด Continue



4.เราจะทำการเข้าไปที่ตัว Data > Token > Generate Token > All access Token

The screenshot shows the Grafana Load Data interface with the "Sources" tab selected. On the left is a sidebar with icons for Data, Explore, Books, Boards, Tasks, Alerts, and Settings. The main area has a search bar at the top with placeholder text "Search data writing methods...". Below it is a "File Upload" section with options for "Annotated CSV" and "Line Protocol". Under "Client Libraries", there are two rows of icons for various programming languages: Arduino, C#, Go, Java, JavaScript/Node.js, Kotlin, PHP, Python, Ruby, Scala, and Swift.

The screenshot shows the Grafana Load Data interface with the "Tokens" tab selected. A red arrow points to the "Tokens" tab in the navigation bar. The main area displays a message: "Looks like there aren't any Tokens, why not generate one?". It includes a "Sort by Description (A → Z)" dropdown and a blue button labeled "+ Generate Token". The sidebar on the left is identical to the first screenshot.

The screenshot shows the Grafana interface with the 'Tokens' tab selected. A message at the top says, "Looks like there aren't any **Tokens**, why not generate one?" Below this is a dropdown menu with three options: '+ Generate Token' (highlighted in blue), 'Read/write Token', and 'All Access Token' (highlighted with a red arrow). The left sidebar contains icons for Data, Explore, Books, Boards, Tasks, Alerts, and Settings.

ຈະໄດ້ໜ້າຕ່າງແບບນີ້ຂຶ້ນມາ

The screenshot shows the Grafana interface with the 'Tokens' tab selected. A message at the top says, "Describe this Token" and "Created at: 2021-06-23 21:06:15". Below this is a status indicator with a blue circle and the word "Active". The left sidebar contains icons for Data, Explore, Books, Boards, Tasks, Alerts, and Settings.

5. จากนั้นเราจะทำการสร้าง Code ในการรันบนที่ ESP32 ไปที่ Data > Sources > Client Libraries และทำการเลือก Arduino

The screenshot shows the InfluxDB web interface. On the left is a sidebar with icons for Data, Explore, Books, Boards, Tasks, Alerts, and Settings. The main area has a header 'Load Data' with tabs for Sources, Buckets, Telegraf, and Tokens. Below is a search bar with placeholder 'Search data writing methods...'. Under 'File Upload', there are options for Annotated CSV and Line Protocol. The 'Client Libraries' section is titled 'Back-end, front-end, and mobile applications' and contains a grid of icons for various platforms: Arduino, C#, Go, Java, JavaScript/Node.js, Kotlin, PHP, Python, Ruby, Scala, and Swift. The Arduino icon is highlighted with a red arrow.

6. จากนั้นทำการ Copy to Clipboard ที่ตัว Initialize the Client และ Write Data

The screenshot shows the InfluxDB web interface again. The sidebar is identical. The main area now displays two code snippets. The top snippet is labeled 'Initialize the Client' and contains C++ code for setting up an InfluxDB client with WiFi credentials and a connection to an InfluxDB instance. The bottom snippet is labeled 'Write Data' and contains C++ code for a loop that clears sensor fields, stores measured values, prints the current network RSSI, and attempts to reconnect if no WiFi signal is found. Both snippets have a 'Copy to Clipboard' button at the bottom. Red arrows point to both snippets.

```
#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

// WiFi AP SSID
#define WIFI_SSID "SSID"
// WiFi password
#define WIFI_PASSWORD "PASSWORD"
// InfluxDB v2 server url, e.g. https://eu-central-1.aws.cloud2.influxdata.com (Use: InfluxDB UI -> Load Data -> Client Libraries)
#define INFLUXDB_URL "https://europe-west1.gcp.cloud2.influxdata.com"
// InfluxDB v2 server API authentication token (Use: InfluxDB UI -> Data -> Tokens -> <select token>)
#define INFLUXDB_TOKEN "0W9y9wDzJ2zqA085y3YRA0TE_PaniH0fse1D3hdyigala2yT3icv03lctBaU4of9suMU22YNdVkw"
// InfluxDB v2 organization id (Use: InfluxDB UI -> User -> About -> Common Ids )
#define INFLUXDB_ORG "b6238285@.sut.ac.th"
// InfluxDB v2 bucket name (Use: InfluxDB UI -> Data -> Buckets)
#define INFLUXDB_BUCKET "b6238285's Bucket"

// Set timezone string according to https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html
// Example: "Asia/Bangkok"
// Pacific Time: "PST8PDT"
// ...
Copy to Clipboard
```

```
void loop() {
    // Clear fields for reusing the point. Tags will remain untouched
    sensor.clearFields();

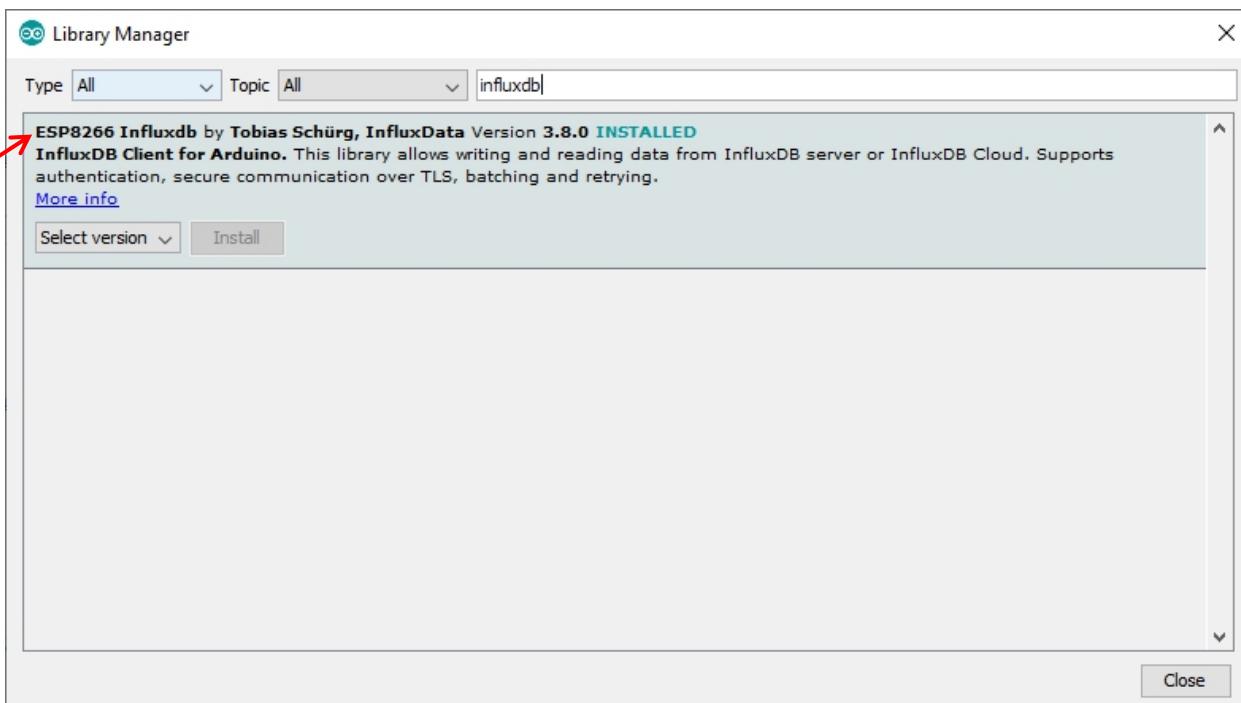
    // Store measured value into point
    // Report RSSI of currently connected network
    sensor.addField("rss", WiFi.RSSI());

    // Print what we are exactly writing
    Serial.print("Writing: ");
    Serial.println(sensor.toLineProtocol());

    // If no WiFi signal, try to reconnect
    if ((WiFi.RSSI() == 0) && (wifiMulti.run() != WL_CONNECTED)) {
        Serial.println("Wifi connection lost");
    }

    // Write point
    if (!client.writePoint(sensor)) {
        Serial.print("InfluxDB write failed: ");
    }
}
Copy to Clipboard
```

7. จាតน้ำหน้าตัวที่ Copy to Clipboard มาวางไว้ที่ Arduino ก่อนจะทำการรันให้ติดตั้งตัว InfluxDB



อันนี้จะเป็นในส่วนที่ต้องแก้ไข Code และ เพิ่ม Code เข้าไป

```
// WiFi AP SSID
#define WIFI_SSID "SSID"
// WiFi password
#define WIFI_PASSWORD "PASSWORD"
```

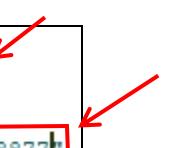
```
void loop() {
    // Clear fields for reusing the point. Tags will remain untouched
    sensor.clearFields();

    // Store measured value into point
    // Report RSSI of currently connected network
    |
    sensor.addField("rss", WiFi.RSSI());
```

หลังจากการ แก้ไข Code และ เพิ่ม Code เข้าไป

- แก้ไขเป็นเป็นชื่อ wifi ที่ใช้อยู่ตอนนี้ พัฟฟ์ password
- ทำการเพิ่มบรรทัด float randomfloat = random(100, 10000)\*0.01;
- ตรง sensor.addField เปลี่ยนเป็นชื่อที่เราตั้งที่ company name คือ Paramet ส่วนตรง WiFi.RSSI()
 เปลี่ยนเป็น randomfloat

```
// WiFi AP SSID
#define WIFI_SSID "Always2" // Red box
// WiFi password
#define WIFI_PASSWORD "0818698877" // Red box
```



```
void loop() {
    // Clear fields for reusing the point. Tags will remain untouched
    sensor.clearFields();

    // Store measured value into point
    // Report RSSI of currently connected network
    float randomfloat = random(100, 10000)*0.01; // Red box
    sensor.addField("Paramet", randomfloat); // Red box
```



อันนี้จะเป็นในส่วน Code Arduino ที่ได้รับการแก้ไขแล้ว

```
#if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

// WiFi AP SSID
#define WIFI_SSID "Always2"
// WiFi password
#define WIFI_PASSWORD "0818698877"
// InfluxDB v2 server url, e.g. https://eu-central-1-1.aws.cloud2.influxdata.com (Use: InfluxDB UI -> Load Data -> Client Libraries)
#define INFLUXDB_URL "https://europe-west1-1.gcp.cloud2.influxdata.com"
// InfluxDB v2 server or cloud API authentication token (Use: InfluxDB UI -> Data -> Tokens -> <select token>)
#define INFLUXDB_TOKEN
"9N49xPma4c1OYgI2Zaw1DBSM3YRAbTE_PanIhDFFsee1DJ3hDy1gala2yT3IcvUJLcTbAu4of9suMU22YNdVkw=="
// InfluxDB v2 organization id (Use: InfluxDB UI -> User -> About -> Common Ids )
#define INFLUXDB_ORG "b6238285@g.sut.ac.th"
// InfluxDB v2 bucket name (Use: InfluxDB UI -> Data -> Buckets)
#define INFLUXDB_BUCKET "b6238285's Bucket"

// Set timezone string according to https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html
// Examples:
// Pacific Time: "PST8PDT"
// Eastern: "EST5EDT"
// Japanesse: "JST-9"
// Central Europe: "CET-1CEST,M3.5.0,M10.5.0/3"
#define TZ_INFO "CET-1CEST,M3.5.0,M10.5.0/3"

// InfluxDB client instance with preconfigured InfluxCloud certificate
InfluxDBClient client(INFLUXDB_URL, INFLUXDB_ORG, INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);

// Data point
Point sensor("wifi_status");
```

```
void setup() {
    Serial.begin(115200);

    // Setup wifi
    WiFi.mode(WIFI_STA);
    wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

    Serial.print("Connecting to wifi");
    while (wifiMulti.run() != WL_CONNECTED) {
        Serial.print(".");
        delay(100);
    }
    Serial.println();

    // Add tags
    sensor.addTag("device", DEVICE);
    sensor.addTag("SSID", WiFi.SSID());

    // Accurate time is necessary for certificate validation and writing in batches
    // For the fastest time sync find NTP servers in your area: https://www.pool.ntp.org/zone/
    // Syncing progress and the time will be printed to Serial.
    timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");

    // Check server connection
    if (client.validateConnection()) {
        Serial.print("Connected to InfluxDB: ");
        Serial.println(client.getServerUrl());
    } else {
        Serial.print("InfluxDB connection failed: ");
        Serial.println(client.getLastErrorMessage());
    }
}

void loop() {
    // Clear fields for reusing the point. Tags will remain untouched
    sensor.clearFields();

    // Store measured value into point
    // Report RSSI of currently connected network
    float randomfloat = random(100, 10000)*0.01;
    sensor.addField("Paramet", randomfloat);
```

```
// Print what are we exactly writing
Serial.print("Writing: ");
Serial.println(sensor.toLineProtocol());

// If no Wifi signal, try to reconnect it
if ((WiFi.RSSI() == 0) && (wifiMulti.run() != WL_CONNECTED)) {
    Serial.println("Wifi connection lost");
}

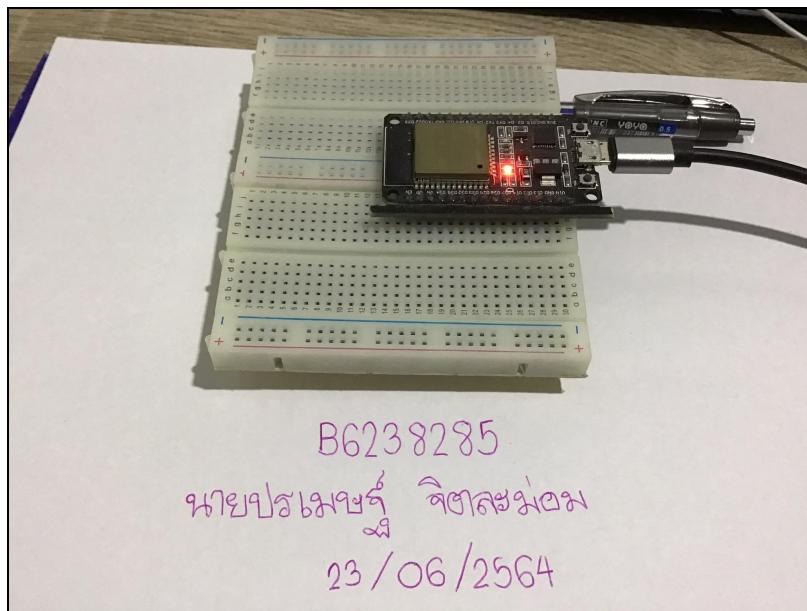
// Write point
if (!client.writePoint(sensor)) {
    Serial.print("InfluxDB write failed: ");
    Serial.println(client.getLastErrorMessage());
}

//Wait 10s
Serial.println("Wait 10s");
delay(10000);
}
```

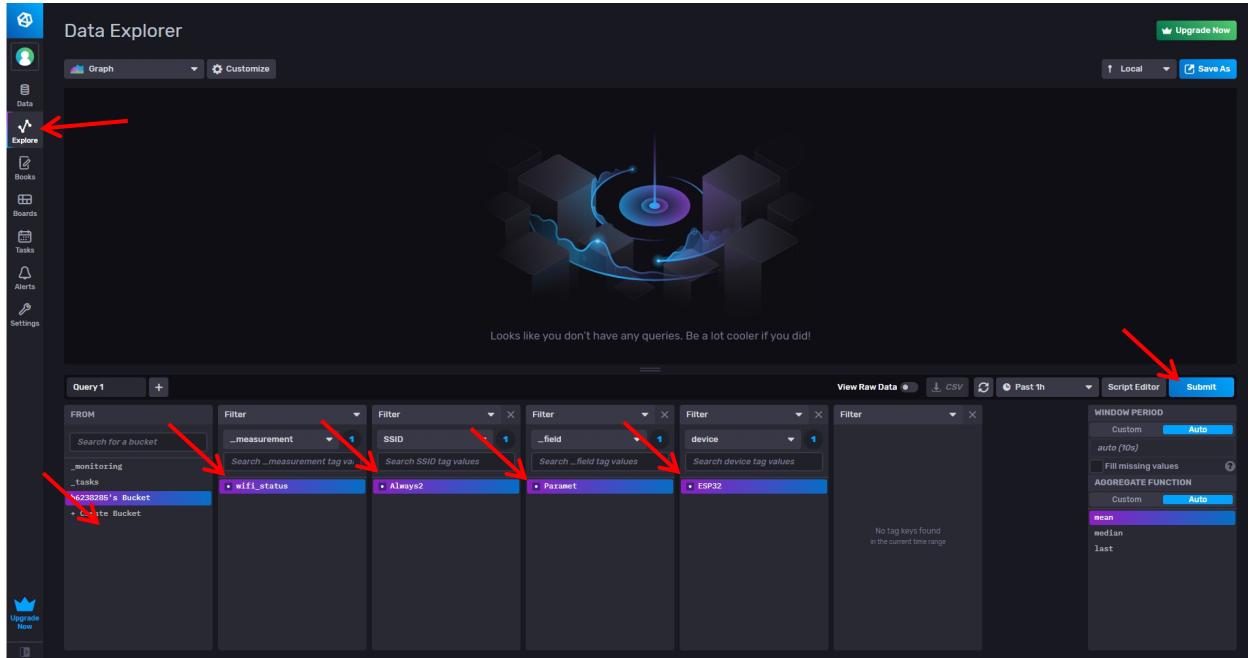
COM3

rst:0x1 (POWERON\_RESET),boot:0x13 (SPI\_FAST\_FLASH\_BOOT)  
 configip: 0, SPIWP:0xee  
 clk\_drv:0x00,q\_drv:0x00,d\_drv:0x00,cs0\_drv:0x00,hd\_drv:0x00,wp\_drv:0x00  
 mode:DIO, clock div:1  
 load:0x3fff0018,len:4  
 load:0x3fff001c,len:1044  
 load:0x40078000,len:10124  
 load:0x40080400,len:5856  
 entry 0x400806a8  
 Connecting to wifi  
 Syncing time.....  
 Synchronized time: Wed Jun 23 17:11:03 2021  
 Connected to InfluxDB: https://europe-west1-1.gcp.cloud2.influxdata.com  
 Writing: wifi\_status,device=ESP32,SSID=Always2 Paramet=4.42  
 Wait 10s  
 Writing: wifi\_status,device=ESP32,SSID=Always2 Paramet=61.89  
 Wait 10s  
 Writing: wifi\_status,device=ESP32,SSID=Always2 Paramet=21.14  
 Wait 10s  
 Writing: wifi\_status,device=ESP32,SSID=Always2 Paramet=91.47  
 Wait 10s  
 Writing: wifi\_status,device=ESP32,SSID=Always2 Paramet=93.07  
 Wait 10s

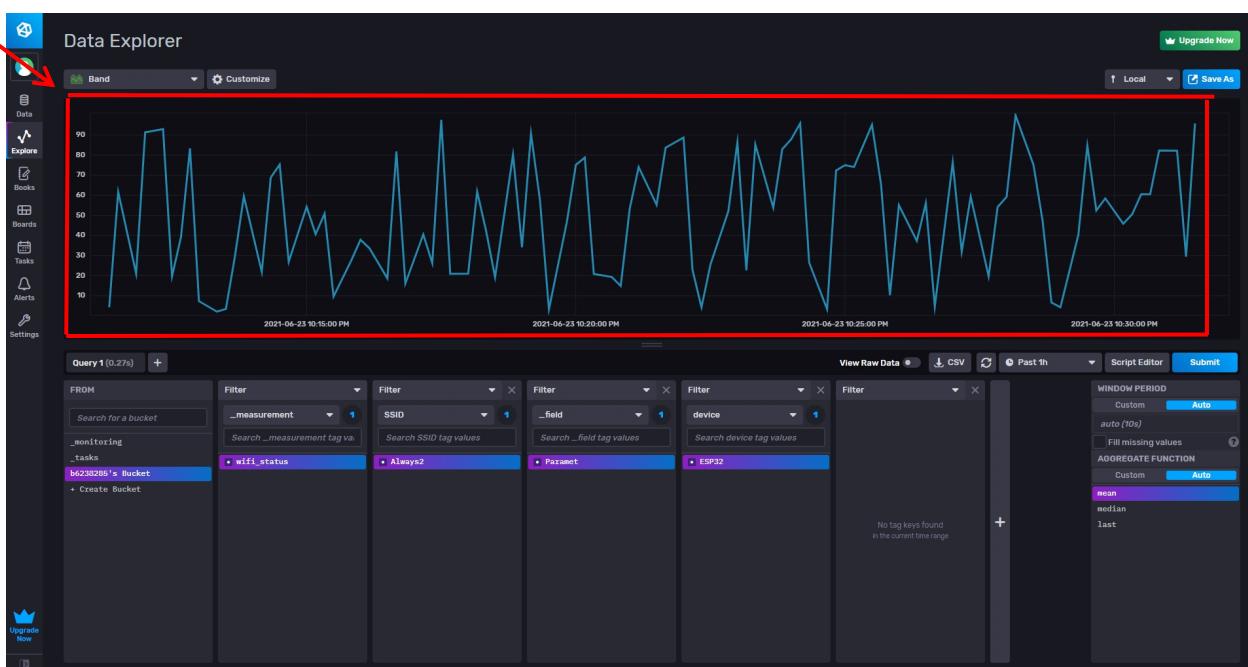
< >  
 Autoscroll  Show timestamp You've pressed Send but nothing was sent. Should you select a line ending? Newline 115200 baud Clear output



8. จากนั้นไปที่ Export เราจะมาดูค่าทำการ random ที่ตัว ESP32 แล้วทำการส่งค่ามาที่ตัว InfluxDB ไปที่ Export หลังจากนั้นทำการเลือกที่ต้องการเลือกดู และกด Submit



จะได้กราฟแบบเป็นรูปนาฬิกา

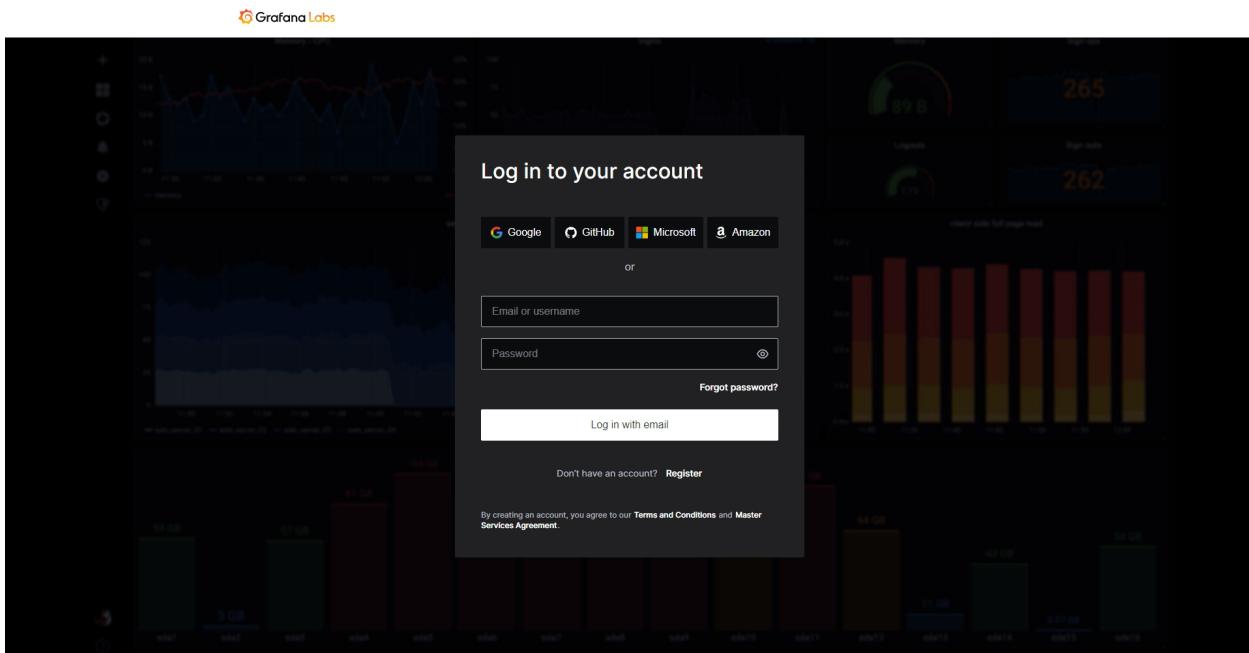




9. ในส่วนต่อมาจะเป็นในตัวของ Garafana เราจะเข้าไปที่เว็บไซต์ <https://grafana.com> เพื่อทำการสร้าง Dashboard ของตัว Garafana

The screenshot shows the Grafana Labs website homepage. At the top, there's a navigation bar with links for "Grafana Labs", "Products", "Open Source", "Learn", "Downloads", "Contact us" (which is highlighted with a red arrow), and "Login". The main content area has a dark background with orange highlights. It features the tagline "Your observability wherever you need it" and "Complete your observability picture". Below this, there are logos for various companies and organizations. A banner at the bottom promotes "Grafana 8.0 is here! Learn how to get started in our demo webinar on June 24. Sign up →".

## 10. เรากำหนดการ Login ด้วย Google



จากนั้นจะได้หน้าต่างแบบนี้ ที่ช่อง Garafana ทำการคลิกที่ Login

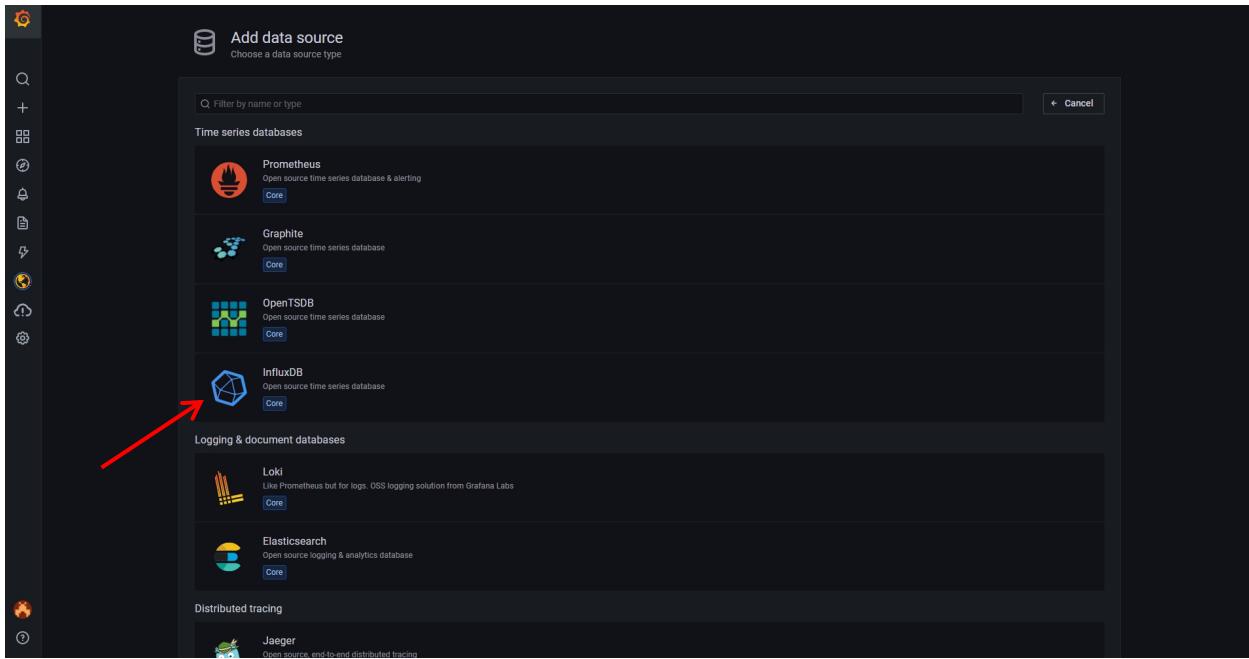
## 11. จะเดือนำต่างแบบชี้นำมา

The screenshot shows the Grafana homepage. On the left, there's a sidebar with various icons. A red arrow points to the 'Dashboards' icon. Below it, under 'Recently viewed dashboards', there's a link 'Started dashboards'. The main content area has three sections: 'Basic' (with a 'TUTORIAL DATA SOURCE AND DASHBOARDS' card), 'COMPLETE Add your first data source' (with a 'Grafana fundamentals' card), and 'COMPLETE Create your first dashboard' (with a 'Learn how in the docs' button). To the right, there's a 'Latest from the blog' section with two posts: 'Jun 22: Grafana dashboard showcase: Visualizations for Prometheus, home energy usage, GitHub, and more!' and 'Jun 21: GrafanaCONline Day 6 recap: The latest on Loki for logs, Grafana for monitoring high performance computing, the business of Grafana Labs, and more!'. Each post has a 'Learn how in the docs' button.

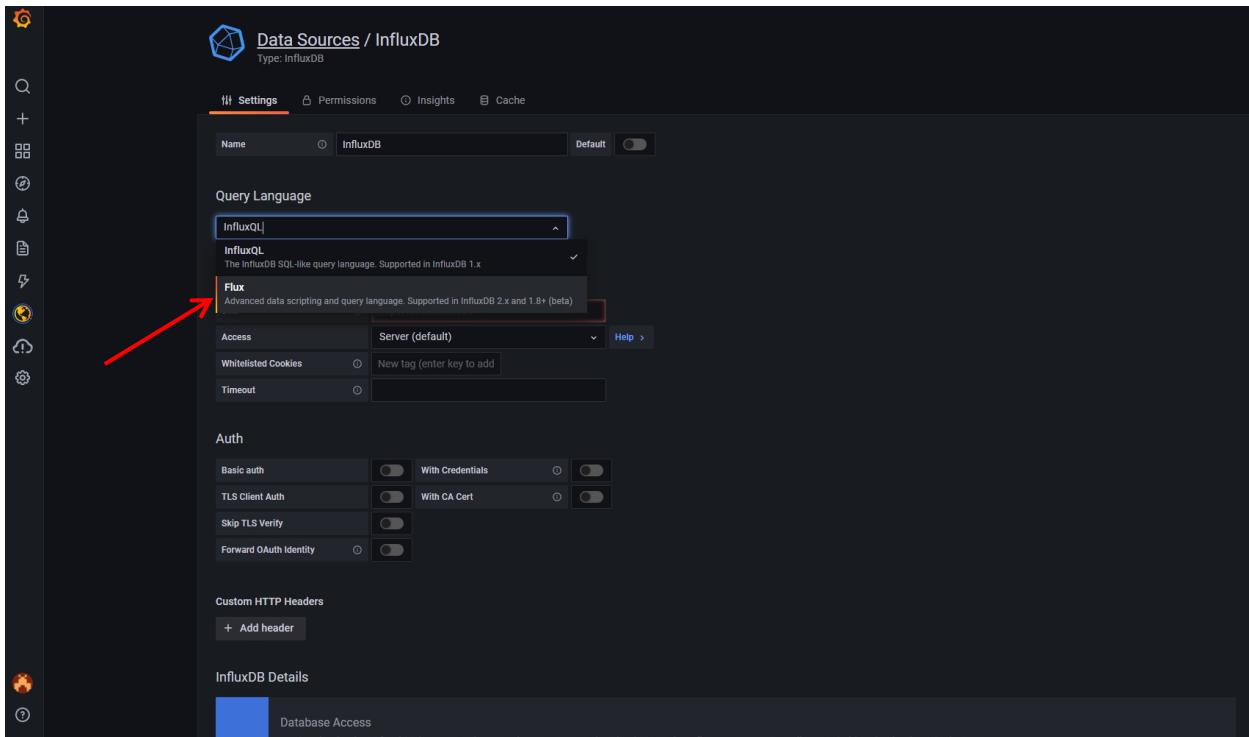
ทำการสร้าง Data source ขึ้นมา ไปที่ Configuration > Data source จากนั้นทำการ Add data source

The screenshot shows the 'Configuration' page in Grafana. On the left, there's a sidebar with 'Configuration' selected, and 'Data sources' is highlighted with a red arrow. The main content area shows a list of data sources: 'grafanacloud-b6238285-alertmanager' (Alert Manager), 'grafanacloud-b6238285-graphite' (Graphite), 'grafanacloud-b6238285-logs' (Loki), 'grafanacloud-b6238285-prom' (Prometheus), 'grafanacloud-b6238285-ruler' (Ruler), 'grafanacloud-b6238285-ruler-logs' (Ruler), 'grafanacloud-b6238285-traces' (Tempo), and 'grafanacloud-usage' (Prometheus). At the top of the list, there's a blue button labeled 'Add data source' with a red arrow pointing to it.

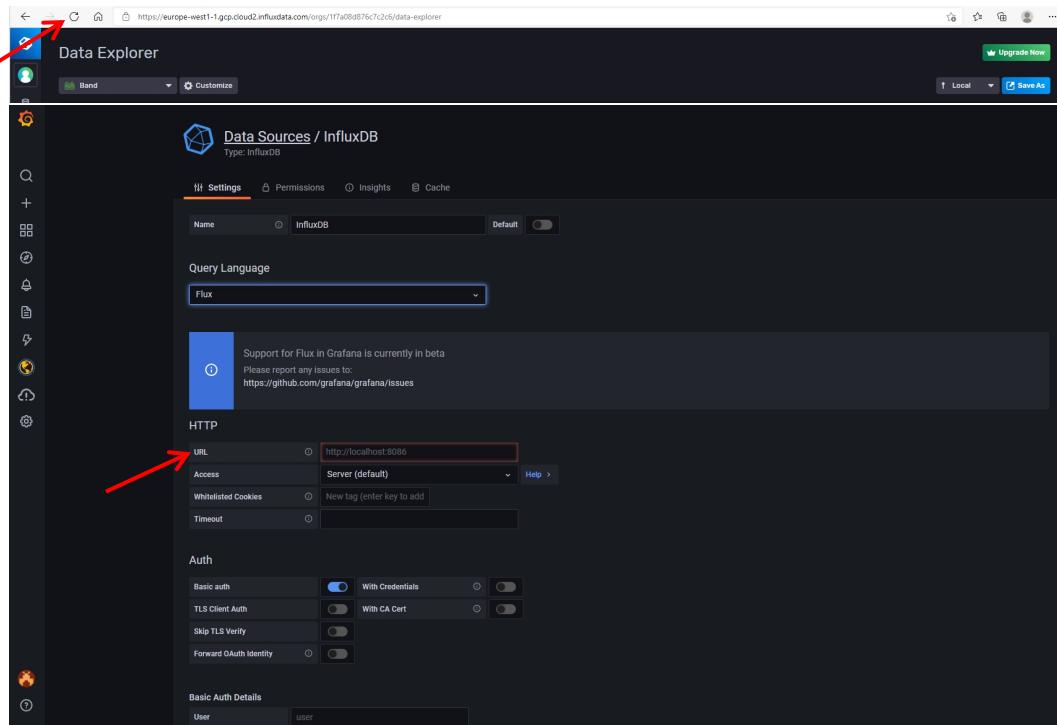
## Add InfluxDB เข้าไป



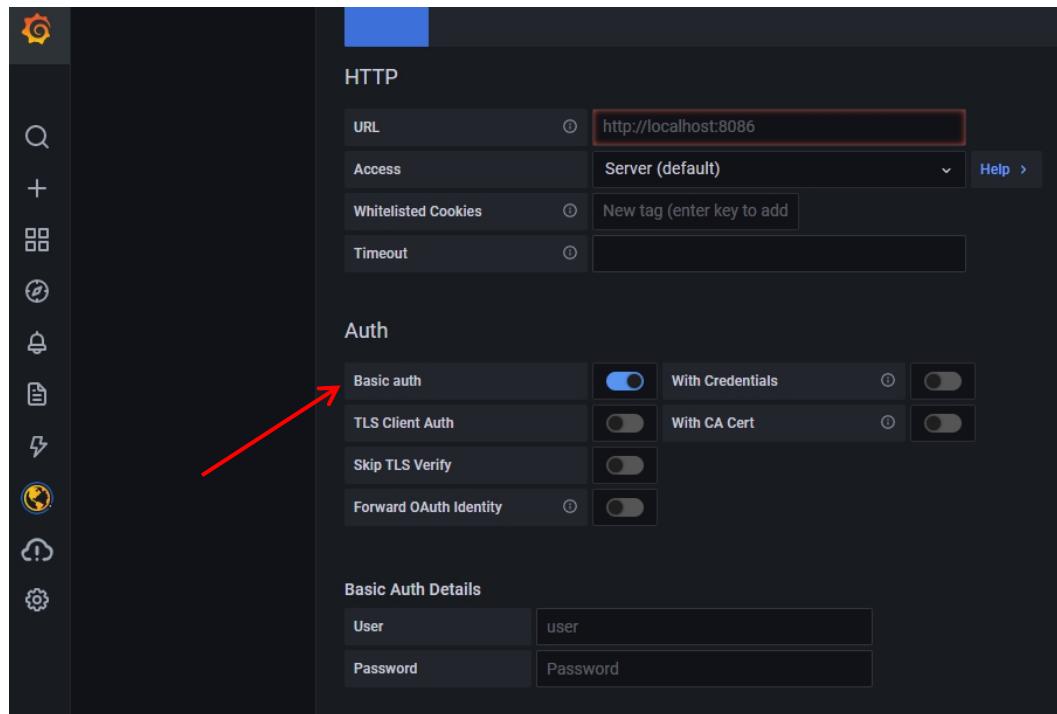
## 12. Query Language ให้เลือกเป็น Flux



13. ในส่วนของ URL นั้นเอาลิงค์ของ URL InfluxDB มา <https://europe-west1-1.gcp.cloud2.influxdata.com/> เอาเคนต์ .com/



14. ต่อมาจะเป็นในส่วน Auth ให้ปิด Basic auth



The screenshot shows the Grafana configuration interface for an 'HTTP' data source. The left sidebar contains various icons for different data sources and settings. The main configuration area is divided into sections: 'HTTP', 'Auth', 'Custom HTTP Headers', and 'InfluxDB Details'. In the 'Auth' section, there are four toggle switches: 'Basic auth' (selected), 'TLS Client Auth' (disabled), 'Skip TLS Verify' (disabled), and 'Forward OAuth Identity' (disabled). A red arrow points to the 'Basic auth' toggle switch. The 'Custom HTTP Headers' section has a '+ Add header' button. The 'InfluxDB Details' section includes fields for 'Organization' and 'Token' (with a corresponding password field).

15. จะเป็นในส่วนของ Tokens ให้ไปที่ InfluxDB > Tokens > Generate Token > Read/Write Token

The screenshot shows the InfluxDB interface with the 'Tokens' tab selected. On the right side, there is a button labeled '+ Generate Token' with a dropdown menu showing 'Read/Write Token'. A red arrow points to this button.

ทำการเลือก b6238285's Bucket เป็นทั้ง Read และ Write และทำการกด Save

The screenshot shows the 'Generate Read/Write Token' dialog. It has two main sections: 'Read' and 'Write'. Each section has a 'Scoped' button and a list of buckets. The bucket 'b6238285's Bucket' is highlighted in both sections. At the bottom, there are 'Cancel' and 'Save' buttons. Two red arrows point to the 'b6238285's Bucket' entries in the respective sections.

จากนั้นดับเบิลคลิกที่ Read

The screenshot shows the Grafana interface with the 'Tokens' tab selected. There is one token listed:

- Read**: Created at: 2021-06-23 23:06:59, Status: Active.

จะได้หน้าต่างแบบนี้ขึ้นมา ทำการ Copy to Clipboard เราจะได้ตัว Token มา

The screenshot shows a modal window titled 'Read' containing a single token value:

```
C464-alii0pZlw_Kp6K2tFiN0pSeuyWAB4-Z_syh1Gs_jAp8SqwXHQoFFXG-1E8r4idlMcWk99LViELESPv-3w==
```

Below the token value is a purple button labeled 'Copy to Clipboard' with a red arrow pointing to it. The modal also contains a 'Read' button in the top right corner and a 'Summary of access permissions' section below the token.

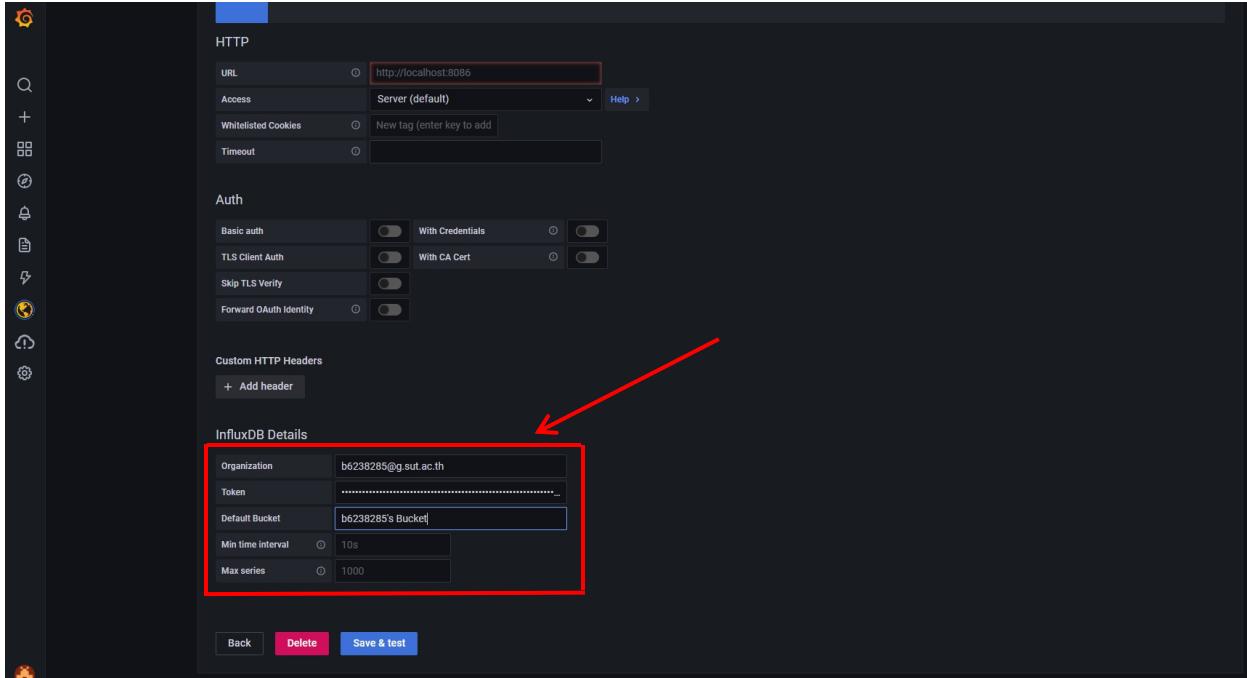
**Summary of access permissions**

buckets-b6238285's Bucket

- read
- write

### 16. จากนั้นในส่วนของ InfluxDB Details

- ในช่องของ Organization ให้ใส่ Email ที่ใช้ในการ Login ที่ InfluxDB
- ในช่อง Token ก็นำค่าที่ Copy to Clipborad ที่ได้มาจาก InfluxDB มาใส่
- ในช่องของ Default Bucket ให้นำชื่อของตัว Export ใน InfluxDB มาใส่



```
from(bucket: "b6238285's Bucket")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "wifi_status")
|> filter(fn: (r) => r["SSID"] == "Always2")
|> filter(fn: (r) => r["_field"] == "Paramet")
|> filter(fn: (r) => r["device"] == "ESP32")
|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
|> yield(name: "mean")
```

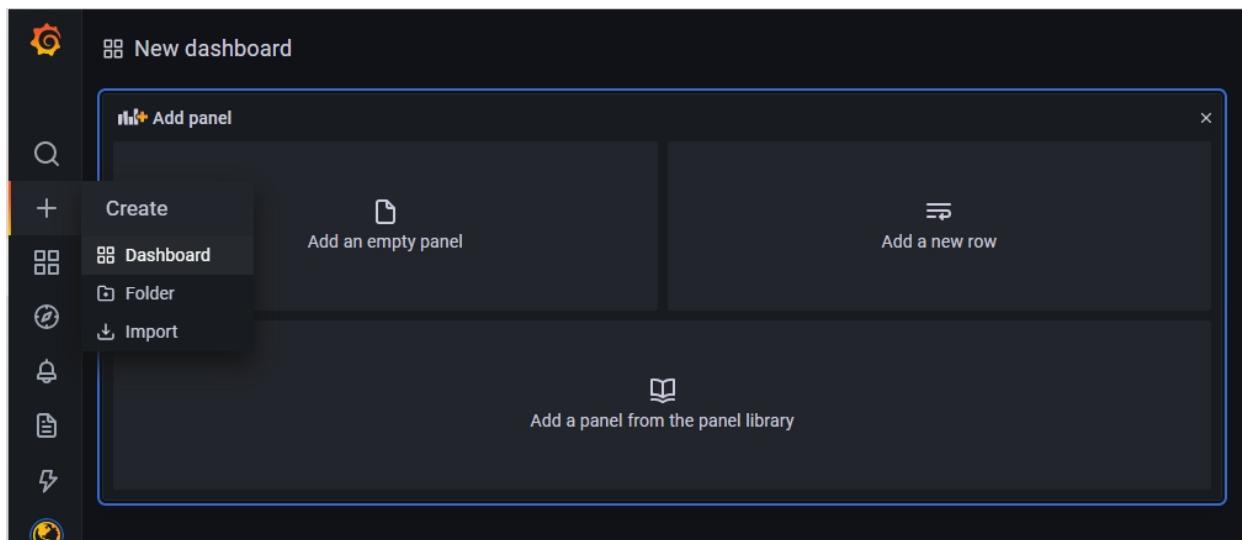
จากนั้นทำการกดปุ่ม Save & Test ถ้าสำเร็จจะขึ้น 1 buckets found

InfluxDB Details

|                   |                      |       |
|-------------------|----------------------|-------|
| Organization      | b6238285@g.sut.ac.th |       |
| Token             | configured           | Reset |
| Default Bucket    | b6238285's Bucket    |       |
| Min time interval | 10s                  |       |
| Max series        | 1000                 |       |

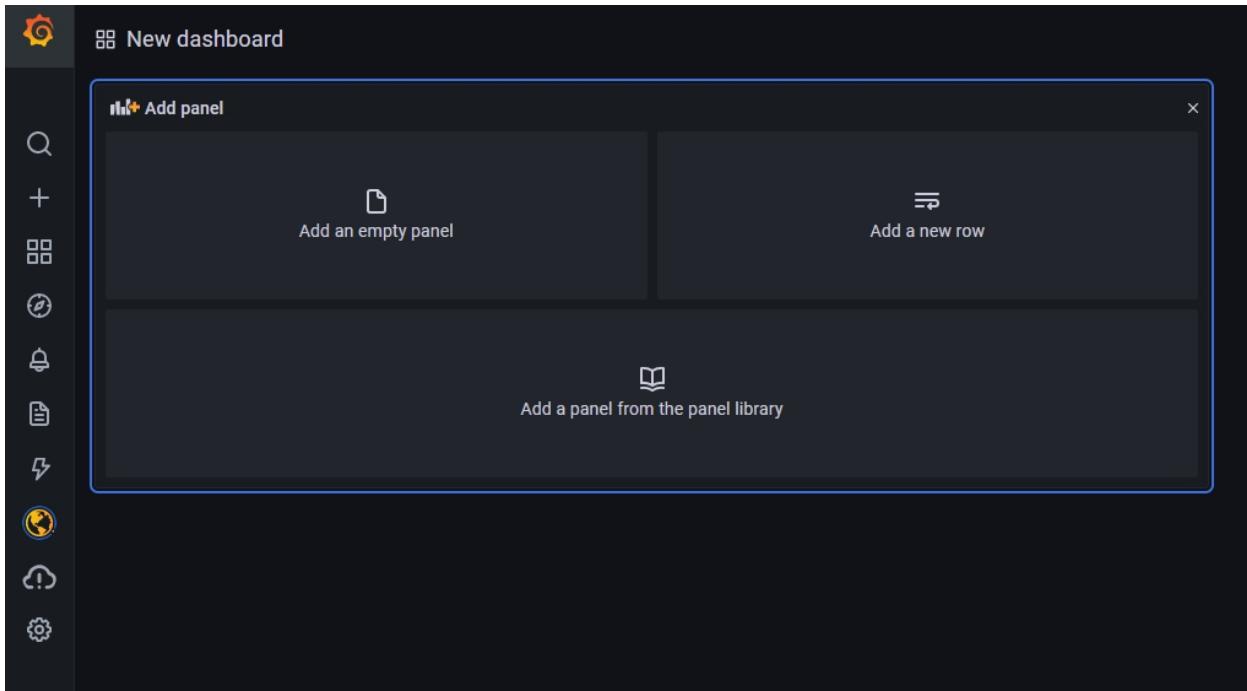
 1 buckets found

17. จากนั้นจะมาในส่วน Dashboard เพื่อใช้ Dashboard ออกมาก ไปที่ Create > Dashboard



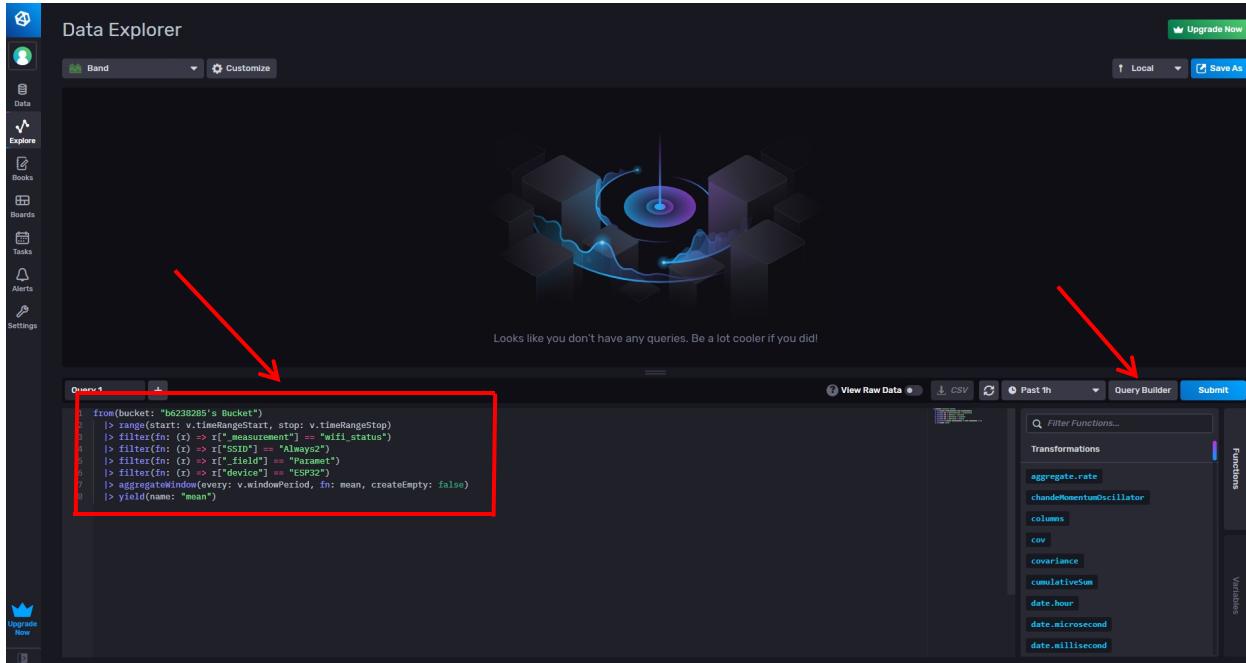
The screenshot shows the Grafana interface with the 'Create' dropdown menu open. The 'Dashboard' option is selected. A modal window titled 'Add panel' is displayed, containing three buttons: 'Add an empty panel', 'Add a new row', and 'Add a panel from the panel library'. The background shows the left sidebar with various icons and the main dashboard area.

จะได้หน้าต่างแบบนี้ขึ้นมา ทำการคลิกที่ Add an empty panel

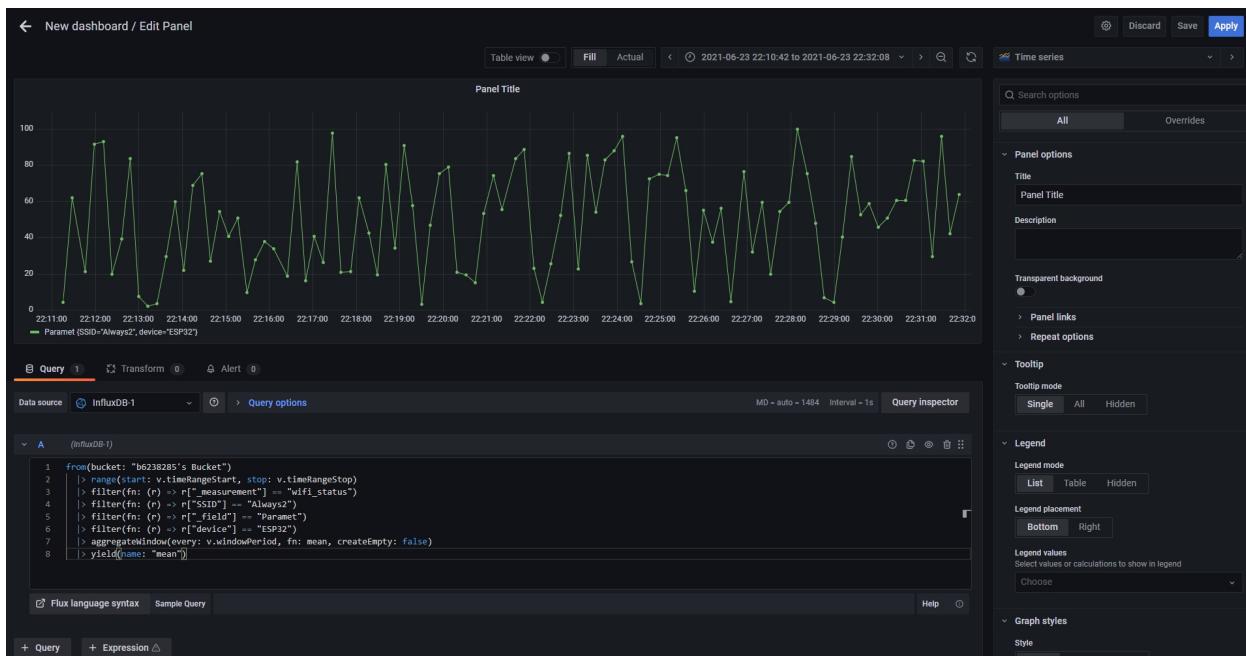


18. ในช่องของ Data source นั้นให้เลือกไปที่ InfluxDB-1 ตามชื่อที่สร้าง Data source ไว้

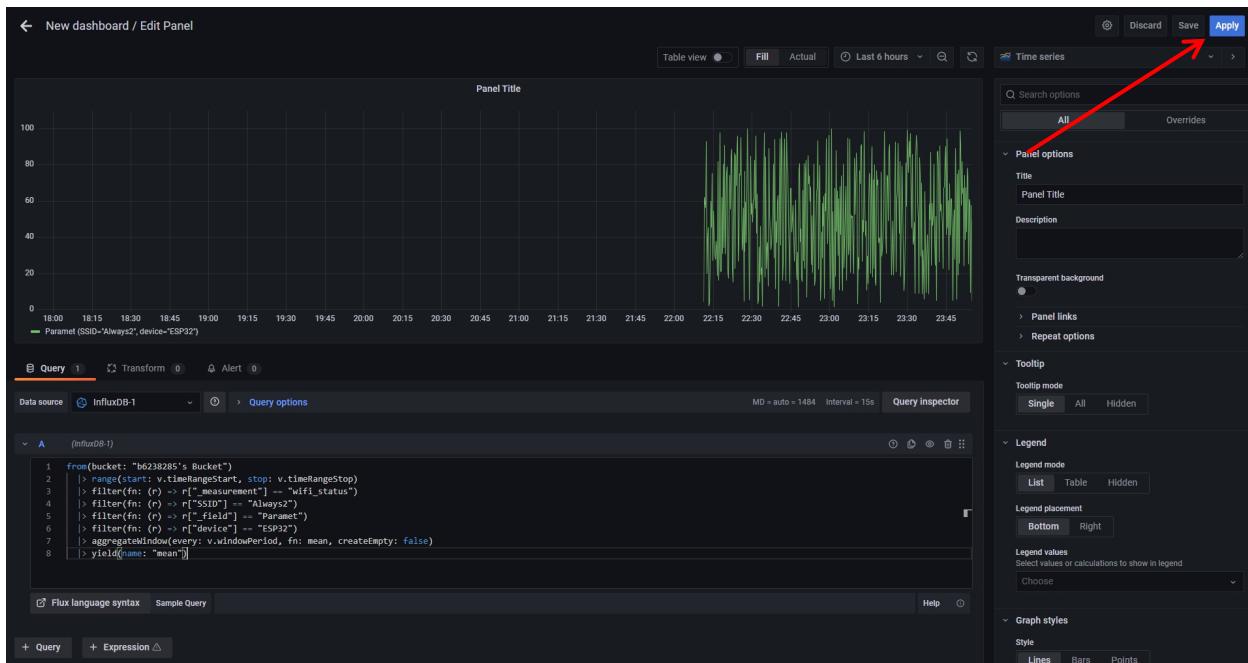
19. ต่อมาจะเป็นในส่วนของ Code ทำการ Copy จาก InfluxDB Data Export มา โดยการที่ Code จะผลลัพธ์มานั้นต้องทำการคลิกที่ปุ่ม Query Builder



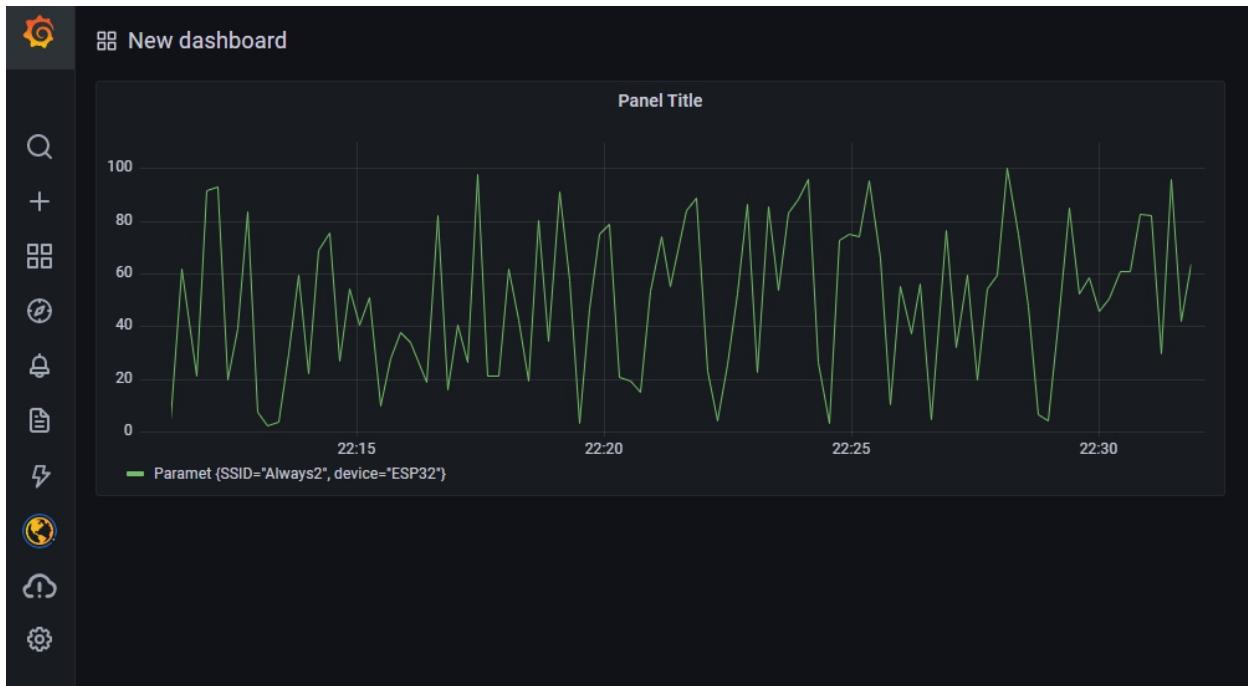
ทำการปรับช่วงเวลาที่ต้องการดู



จากนั้นทำการ Apply



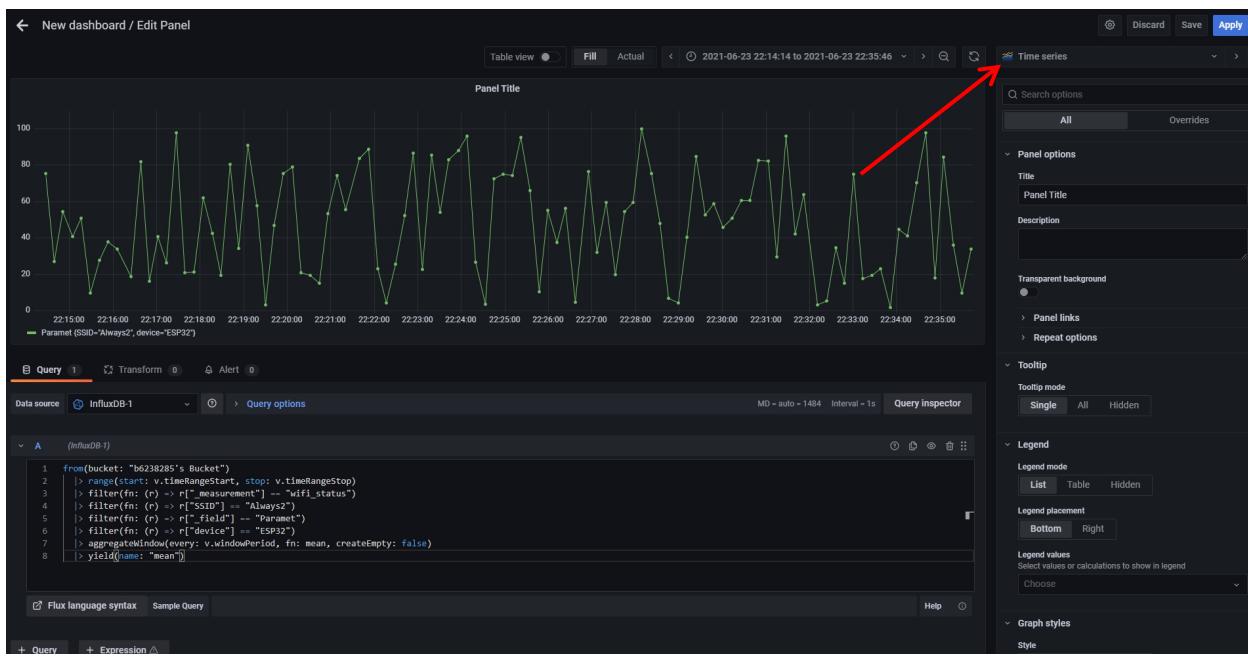
จากนั้นจะได้ตัว Dashboard ขึ้นมา



เราสามารถสร้าง Dashboard ในรูปแบบต่างๆได้อีกด้วยการไปที่ Add panel



แล้วทำการสร้าง Dashboard แบบเดิมเลย เพิ่มเติมคือ ไปที่ time series



แล้วเลือกที่ Gauge จะได้ดังรูป จากนั้นกด Apply

New dashboard / Edit Panel

Panel Title

34.0

Visualizations

- Time series
- Bar chart
- 12.4 Stat
- Gauge
- Bar gauge
- Table
- Pie chart
- State timeline
- Heatmap
- Status history
- Histogram
- Graph (old)
- Text

Query 1

InfluxDB-1

Query options

```
from(bucket: "6e238285's Bucket")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "wifi_status")
  |> filter(fn: (r) => r["SSID"] == "Always2")
  |> filter(fn: (r) => r["fieldID"] == "Paramet")
  |> filter(fn: (r) => r["device"] == "ESP32")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

Flux language syntax Sample Query

+ Query + Expression ▾

