

ラズパイ上にElasticsearchとkibanaを構築する方法

2020/12/15 市川

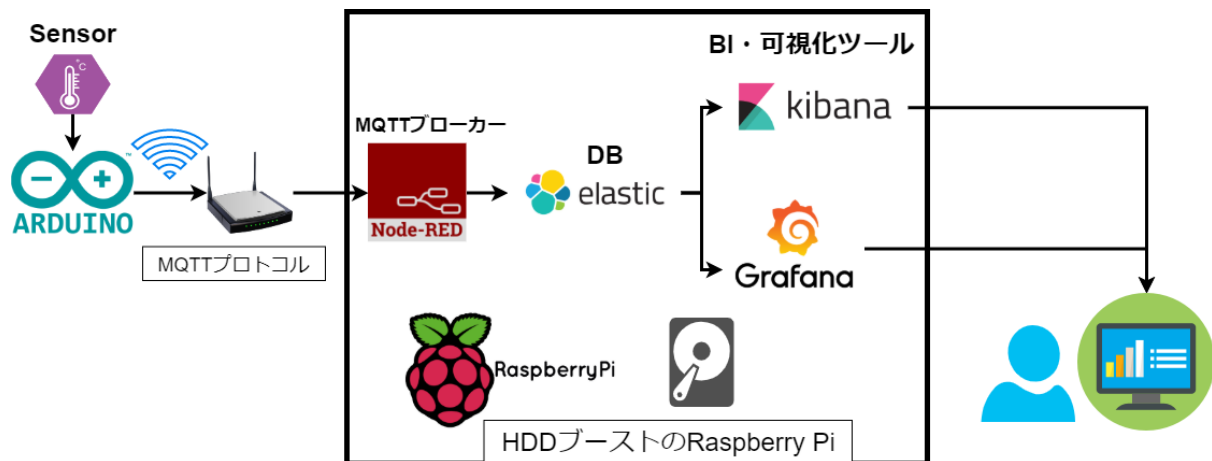
elasticの構築方法参考URL: <https://qiita.com/Y-Shikase/items/d162805aebb3a1a8447f>

arduinoのプログラムとnode-redの設定はこちら:

<https://github.com/factoryscientist/wionodekits/tree/master/nodered-mqtt-elastic>

構築イメージ

今回は四角の中を環境を構築します。



用意するもの

- ・セットアップ済みのラズパイ一式
- ・インターネット環境

Node-REDのインストール

```
bash <(curl -sL  
https://raw.githubusercontent.com/node-red/linux-installers/master/deb/u  
pdate-nodejs-and-nodered)
```

自動起動設定

```
systemctl enable nodered.service  
sudo systemctl disable nodered.service
```

セキュリティ設定など詳しくは下記の参考サイトを確認ください
Raspberry PiにおけるNode-REDの活用について
<https://qiita.com/utaani/items/7155c62d6c5e96822afb>

dockerのインストール

```
curl -sSL https://get.docker.com/ | sh  
sudo usermod -aG docker pi  
sudo systemctl isolate reboot.target
```

elasticsearchとkibanaのインストール

elasticsearchのインストール

作業フォルダを作成とファイルのダウンロード、Dockerfileの作成します

```
sudo mkdir /opt/elasticsearch  
cd /opt/elasticsearch  
sudo wget  
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.10.  
0-linux-x86_64.tar.gz  
sudo nano Dockerfile
```

Dockerfileファイルの中身をこちらです。

```
FROM adoptopenjdk:13-jdk-hotspot
COPY elasticsearch-7.10.0-linux-x86_64.tar.gz /opt
RUN tar xzf /opt/elasticsearch-*.tar.gz -C /opt &&\
    rm /opt/elasticsearch-*.tar.gz &&\
    ln -s /opt/elasticsearch-* /opt/elasticsearch &&\
    echo "network.host: 0.0.0.0" >>
/opt/elasticsearch/config/elasticsearch.yml &&\
    echo "discovery.type: single-node" >>
/opt/elasticsearch/config/elasticsearch.yml &&\
    echo "xpack.ml.enabled: false" >>
/opt/elasticsearch/config/elasticsearch.yml &&\
    echo "path.repo : ['/opt/elasticsearch/snapshot']" >>
/opt/elasticsearch/config/elasticsearch.yml &&\
    perl -pi -e "s/Xms1g/Xms512m/" /opt/elasticsearch/config/jvm.options
&&\
    perl -pi -e "s/Xmx1g/Xmx512m/" /opt/elasticsearch/config/jvm.options
ENV JAVA_HOME=/opt/java/openjdk
CMD ["/opt/elasticsearch/bin/elasticsearch"]
```

パスワードを設定したい場合はこちらを追加します。
「yourpassword」をご自身が設定したいパスワードに書き換えます。
詳しくは公式ドキュメントを参照ください。

<https://www.elastic.co/guide/en/security/current/index.html>

```
    echo yourpassword | /opt/elasticsearch/bin/elasticsearch-keystore add
"bootstrap.password" -xf&&\
    echo "xpack.security.enabled: true" >>
/opt/elasticsearch/config/elasticsearch.yml &&\
```

Dockerのビルドと試し起動

```
sudo docker build -t rpi-elasticsearch:7.10.0 .
sudo docker run --rm -it rpi-elasticsearch:7.10.0
```

少し待つと下記の表記が登場します。

```
[INFO ][o.e.x.s.s.SecurityStatusChangeListener]
```

ブラウザからアクセスすると下記のような表示がされます。
URLは「192.168.x.xxx:5601/」です。ラズパイのIPaddressの後に「:5601」と付け加えます。
ログイン設定をした場合usernameはelasticです。

ブラウザではなくコマンドでも確認できます

```
curl -u elastic 'localhost:9200?pretty'
```

下記のようなレスポンスがあれば正常に稼働しています。

```
{
  "name" : "9feb98acac4b",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "aajA3QiMRw-fcVmuA1FWtw",
  "version" : {
    "number" : "7.10.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "51e9d6f22758d0374a0f3f5c6e8f3a7997850f96",
    "build_date" : "2020-10-09T21:30:33.964949Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

問題なければctrl+cで試しを終了します。

kibanaのインストール

作業フォルダを作成とファイルのダウンロード、Dockerfileの作成します

```
sudo mkdir /opt/kibana
cd /opt/kibana
sudo wget
https://artifacts.elastic.co/downloads/kibana/kibana-7.10.0-linux-x86_64
.tar.gz
sudo nano Dockerfile
```

Dockerfileファイルの中身をこちらです。

```
FROM node:10.22.1
COPY kibana-7.10.0-linux-x86_64.tar.gz /opt
RUN tar xzf /opt/kibana-*.tar.gz -C /opt &&\
    rm /opt/kibana-*.tar.gz &&\
    ln -s /opt/kibana-* /opt/kibana &&\
    echo 'server.host: "0.0.0.0"' >> /opt/kibana/config/kibana.yml &&\
    echo 'i18n.locale: "ja-JP"' >> /opt/kibana/config/kibana.yml &&\
    echo 'elasticsearch.hosts: ["http://elasticsearch:9200"]' >>
/opt/kibana/config/kibana.yml &&\
    echo "xpack.security.enabled: false" >>
/opt/kibana/config/kibana.yml &&\
    rm /opt/kibana/node/bin/node &&\
    ln -sr `which node` /opt/kibana/node/bin/node
ENV NODE_OPTIONS=--max-old-space-size=512
CMD ["/opt/kibana/bin/kibana", "--allow-root"]
```

パスワードを設定した場合はこちらを追加します。

「yourpassword」をご自身が設定したパスワードに書き換えます。

詳しくは公式ドキュメントを参照ください。

<https://www.elastic.co/guide/en/kibana/current/using-kibana-with-security.html>

```
echo "xpack.security.enabled: true" >> /opt/kibana/config/kibana.yml &&\
に書き換える
```

```
echo 'elasticsearch.username: "elastic"' >>
/opt/kibana/config/kibana.yml &&\
    echo 'elasticsearch.password: "yourpassword"' >>
/opt/kibana/config/kibana.yml &&\
    echo 'xpack.security.encryptionKey: "暗号化キーとして、32文字以上の任意の
テキスト文字列"' >> /opt/kibana/config/kibana.yml &&\
    echo 'xpack.encryptedSavedObjects.encryptionKey: "暗号化キーとして、32文
字以上の任意のテキスト文字列"' >> /opt/kibana/config/kibana.yml &&\
    echo 'xpack.security.session.idleTimeout: "1h"' >>
/opt/kibana/config/kibana.yml &&\
    echo 'xpack.security.session.lifespan: "30d"' >>
```

```
/opt/kibana/config/kibana.yml &&\
```

Dockerのビルドと試し起動

ビルドに5分ほど時間が掛ります。

```
sudo docker build -t rpi-kibana:7.10.0 .  
sudo docker run --rm -it rpi-kibana:7.10.0
```

少し待つと下記の表記が登場します。

```
[error ][data][elasticsearch] [ConnectionError]: getaddrinfo ENOTFOUND~
```

問題なければctrl+cで試しを終了しましょう。

ネットワーク作成

elasticsearchとkibanaが通信できるようにネットワークを作成します。

```
sudo docker network create elastic
```

データを入れるフォルダ作成します。

```
sudo mkdir /opt/elasticsearch/es-data
```

elasticsearchとkibanaの起動

elasticsearchとkibanaの起動します。

```
sudo docker run --rm -d --name elasticsearch --network elastic -v  
/opt/elasticsearch/es-data:/opt/elasticsearch/data -p 9200:9200  
rpi-elasticsearch:7.10.0  
sudo docker run --rm -d --name kibana --network elastic -p 5601:5601  
rpi-kibana:7.10.0
```

ラズパイのロードアベレージとCPU温度を投入

ラズパイのロードアベレージとCPU温度をelasticsearchに投入します。

```
sudo su -
sudo mkdir bin
cd bin
sudo nano rpiinfo.sh
```

rpiinfo.shの中身はこちらです。

パスワード設定をしている場合は-sのあとにこちらを追加してください。「-u elastic:yourpassword」

```
HOST=`hostname`
DATE=`date --iso-8601="seconds"`

LOAD=`cat /proc/loadavg | cut -d' ' -f1`
CPUTEMP=`vcgencmd measure_temp | cut -d= -f2 | cut -d'"' -f1`
JSON='{"@timestamp":"'${DATE}'","${HOST}":'-temp':'${CPUTEMP}','${HOST}':
-load':'${LOAD}'}'

curl --no-keepalive -s -XPOST -H "Content-Type: application/json"
http://localhost:9200/rpi-`hostname`/_doc?pretty -d "${JSON}"
```

実行権限の付与とcronfileの作成します。

```
chmod +x rpiinfo.sh
sudo nano cronfile
```

cronfile の中身はこちらです。すでに作成済みの場合は追加します。

```
*/1 * * * * /root/bin/rpiinfo.sh >/dev/null 2>&1
```

登録します。

```
crontab cronfile
```

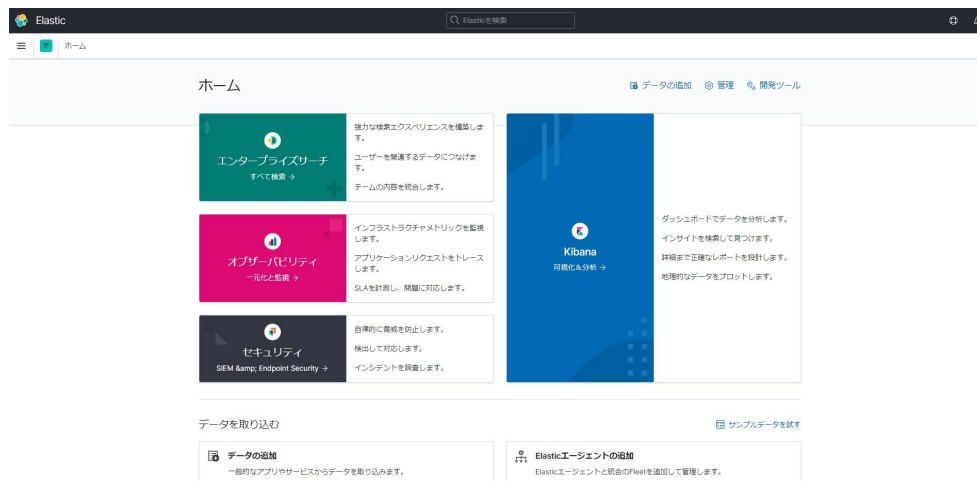
登録を確認します。

```
crontab -l
```

先ほど作成したcronfileの中身が出力されれば登録されています。

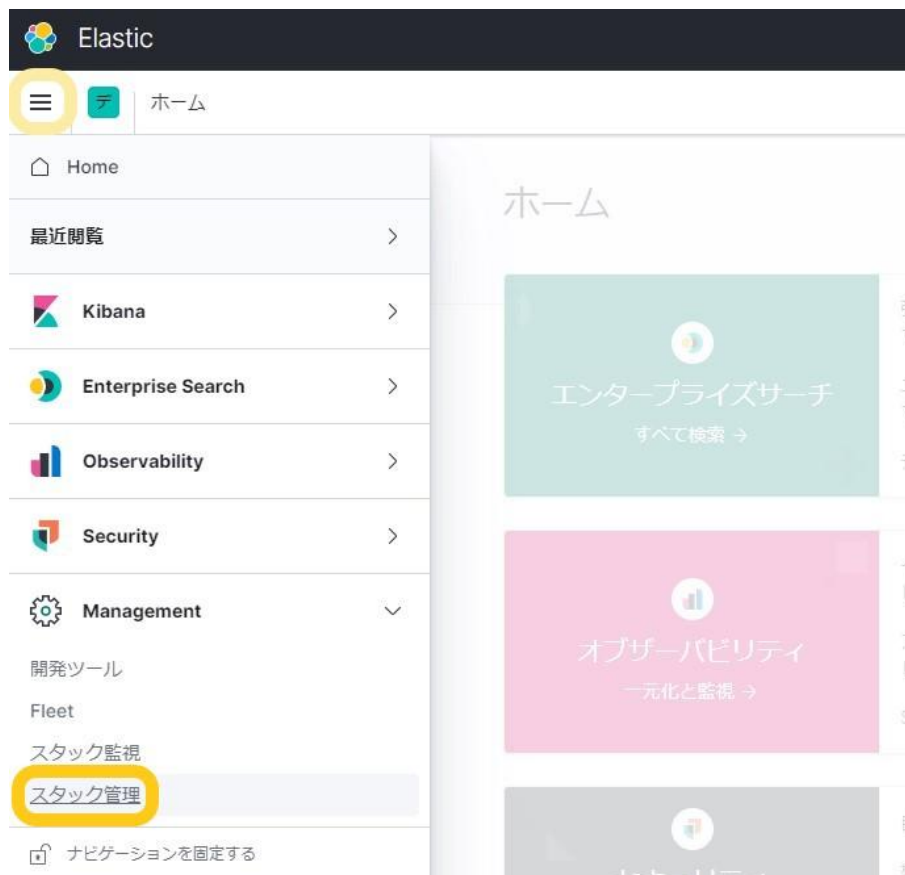
kibanaで可視化

kibanaにアクセスしデータを確認、グラフ化します。

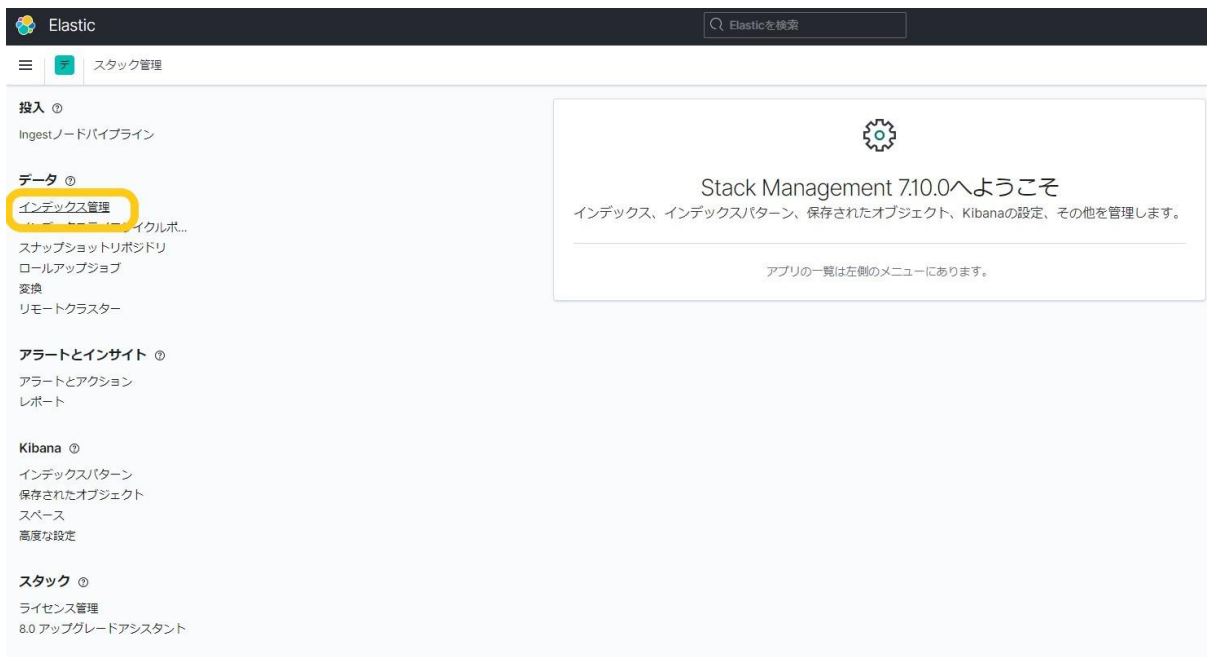


データの到着確認をします。

スタック管理を選択します。



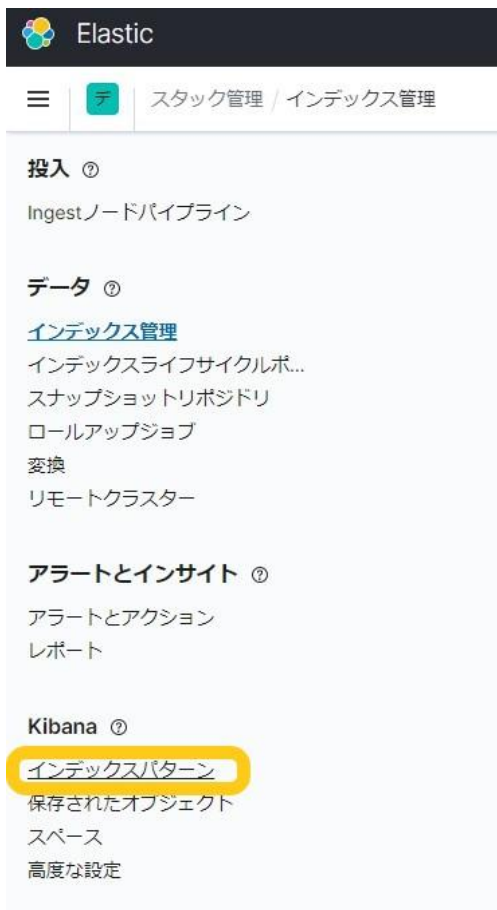
インデックス管理を選択します。



以下のようになっていればデータが登録されています。



データの設定を行います。
インデックスパターンを選択します。



インデックスパターンを作成を選択します。



インデックスパターン名に「rpi-raspi*」と入力し、次のステップへ進みます。

インデックスパターンの作成

インデックスパターンは、filebeat-4-3-22 またはmultipleデータソース、filebeat-* と一致します。
[ドキュメンテーションを表示](#)

ステップ1/2：インデックスパターンを定義

インデックスパターン名

rpi-raspi*

[次のステップ >](#)

複数インデックスの一致にアスタリスク (*) を使用。スペースと \, /, ?, ", <, >, | は使用できません。

☐ システムと非表示のインデックスを含める

✓ インデックスパターンは、1 個のソースと一致します。

rpi-raspi

インデックス

ページごとの行数: 10

時間フィールドを「@timestamp」にします。

インデックスパターンの作成

インデックスパターンは、filebeat-4-3-22 またはmultipleデータソース、filebeat-* と一致します。
[ドキュメンテーションを表示](#)

ステップ2/2：設定の構成

rpi-raspi* インデックスパターンの設定を指定します。

グローバル時間フィルターで使用するためのプライマリ時間フィールドを選択してください。

時間フィールド

[更新](#)

@timestamp

[高度なSIEM設定の表示](#)

[戻る](#)

[インデックスパターンを作成](#)

作成完了すると以下の画像のようになります。

rpi-raspi*

時刻フィールド: 「@timestamp」

このページはrpi-raspi*インデックス内のすべてのフィールドと、Elasticsearchに記録された各フィールドのコアタイプを一覧表示します。フィールドタイプを変更するにはElasticsearchを使用します [マッピングAPI](#)

フィールド (8) スクリプトフィールド (0) ソースフィルター (0)

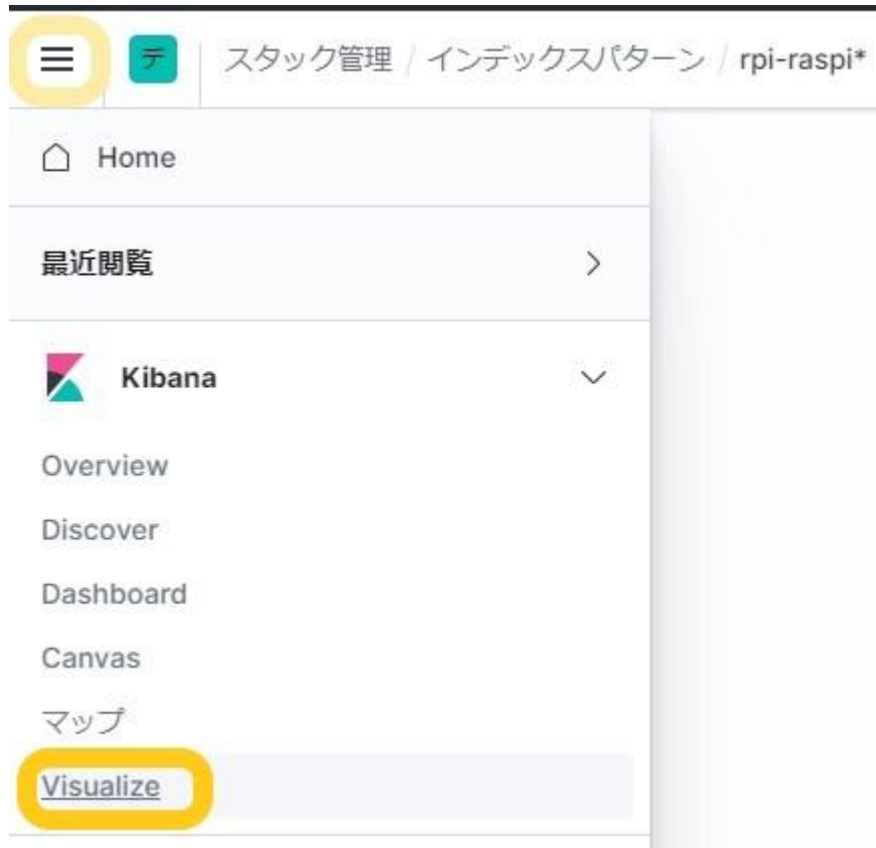
名前	タイプ	フォーマット	検索可能	集約可能	除外
@timestamp	date		●	●	
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
raspi-load	number		●	●	
raspi-temp	number		●	●	

Rows per page: 10

< 1 >

可視化をします。

visualizeを選択します。



新規ビジュアライゼーションを追加を選択します。



最初のビジュアライゼーションの作成

データに基づき異なるビジュアライゼーションを作成できます。

⊕ 新規ビジュアライゼーションを追加

レンズビジュアライゼーションを選択します。

新規ビジュアライゼーション

Filter

レンズビジュアライゼーション

Markdown

TSVB

Timelion

Vega

エリア

ゲージ

コントロール

ゴール

タグクラウド

データテーブル

パイ

ヒートマップ

マップ

メトリック

折れ線

横棒

縦棒

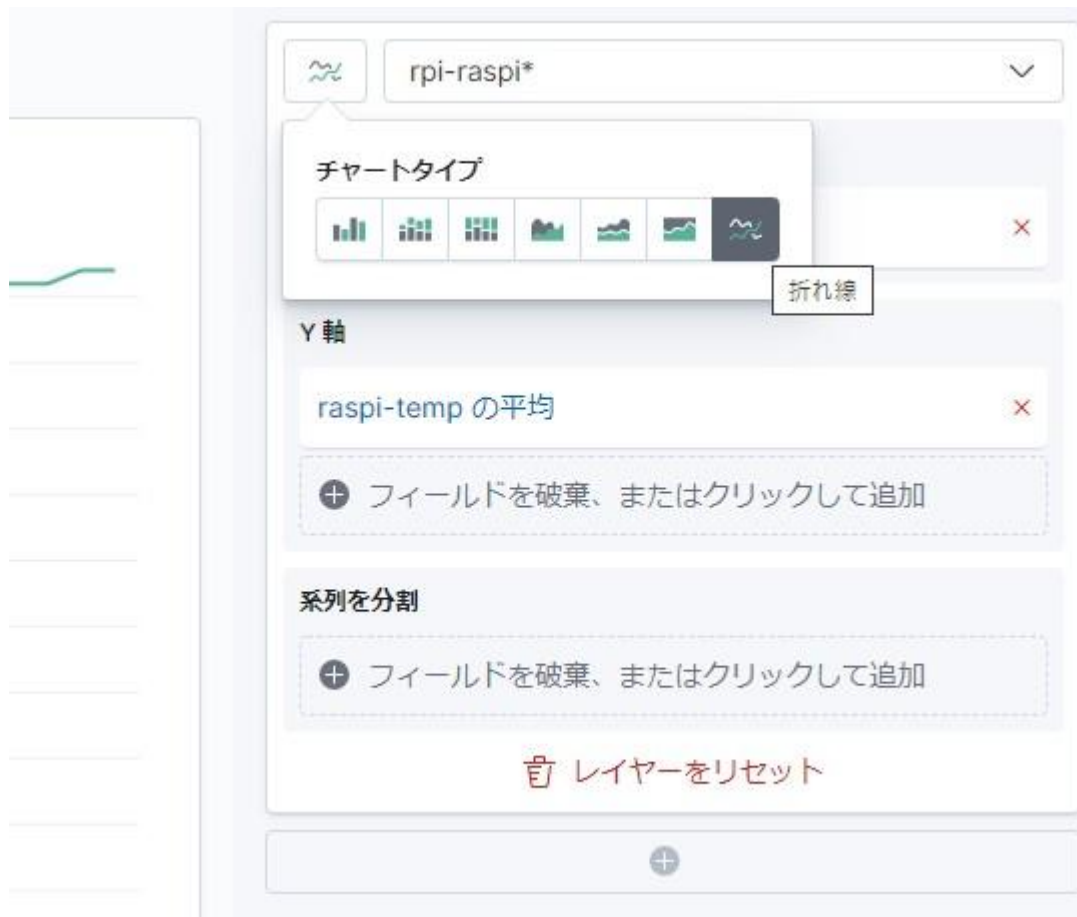
レンズビジュアライゼーション

レンズは基本的なビジュアライゼーションを作成するシンプルな方法です

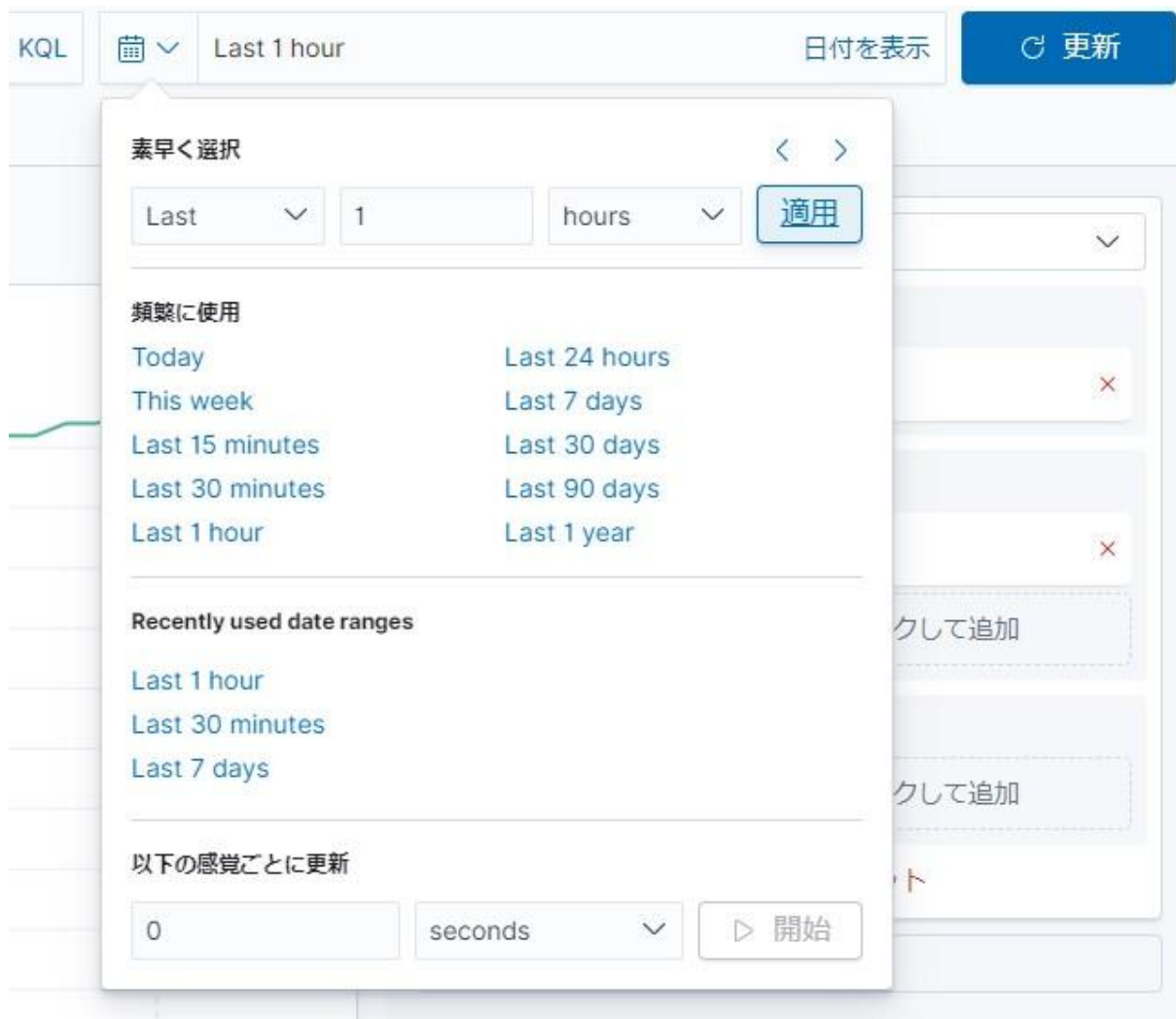
@timestampをX軸に、raspi-tempをY軸にそれぞれドラッグ & ドロップします。



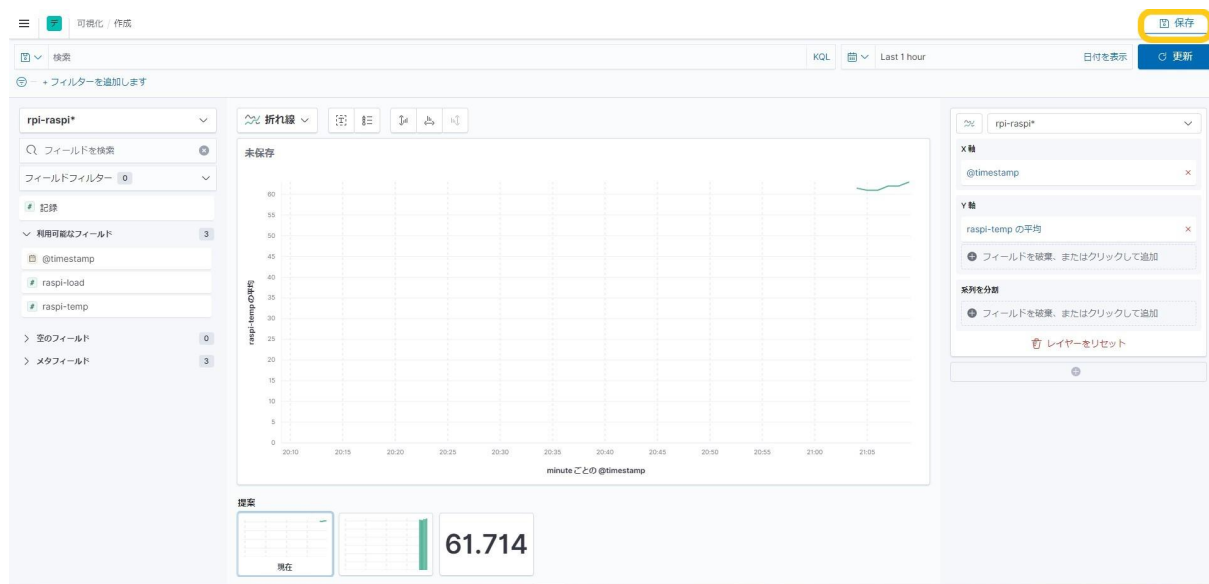
チャートタイプを折れ線にします。



時間範囲を直近1時間にします。



保存します。



タイトルを入力し、保存します。

×

レンズビジュアライゼーションを保存

タイトル

ラズパイCPU温度

説明

キャンセル

保存

ダッシュボードを選択し、新規ダッシュボードを作成を選択します。

デ

ダッシュボード

Home

最近閲覧

Kibana

Overview

Discover

Dashboard

Canvas

マップ

Visualize

Enterprise Search

Observability

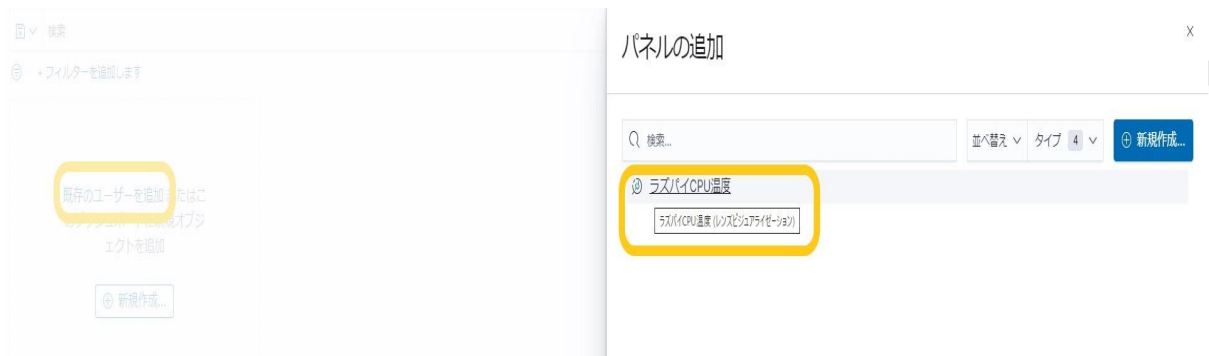
初めてのダッシュボードを作成してみよう。

あらゆるKibanaアプリからダッシュボードにデータビュを組み合わせ、すべてを1か所に表示できます。

Kibanaは初心者ですか？ サンプルデータをインストールしてお試しください。

⊕ 新規ダッシュボードを作成

既存のユーザーを追加を選択し、先ほど保存したラズパイCPU温度を選択します。



最後にダッシュボードを保存して終了です。

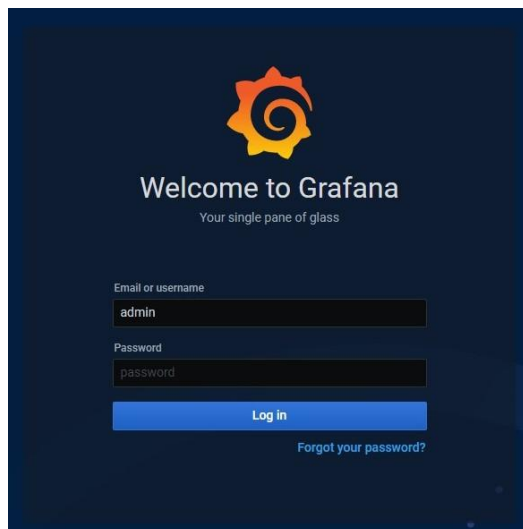
Grafanaのインストール

elasticsearchはkibana以外の可視化ツールでも接続できます。

```
wget https://dl.grafana.com/oss/release/grafana_7.0.4_armhf.deb
sudo apt install ./grafana_7.0.4_armhf.deb
sudo systemctl status grafana-server.service
sudo systemctl start grafana-server.service
```

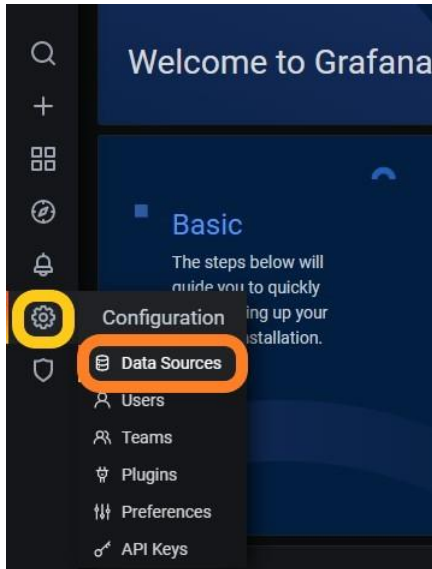
Grafanaで可視化

ログインします。usernameは「admin」です。

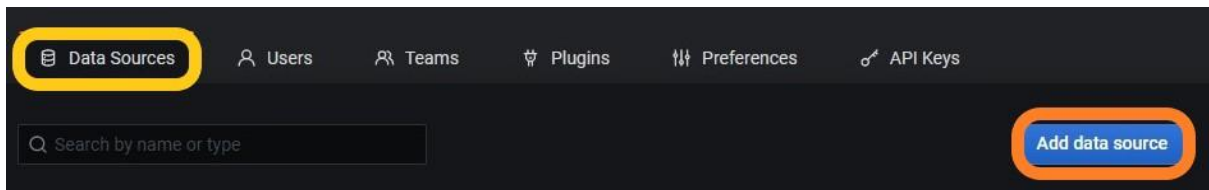


Grafanaをelasticserchに接続

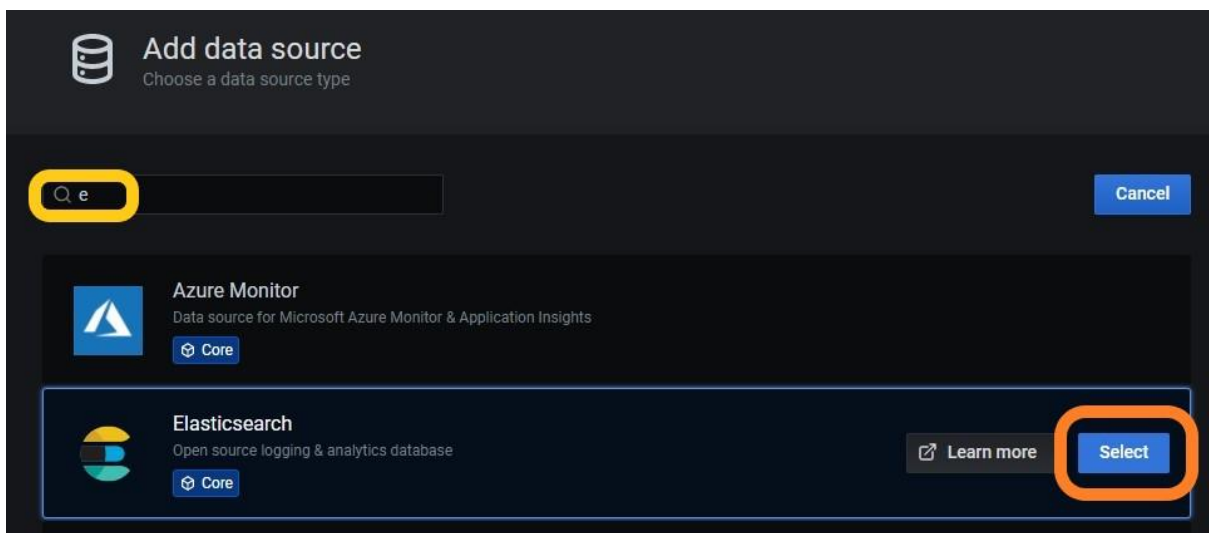
データソースを選択します。



データソースの追加を選択します。



Elasticsearchを選択します。



URLにラズパイのアドレスを入力します。
認証設定をしている場合は「Basic auth」をONにします。

Data Sources / Elasticsearch
Type: Elasticsearch

Settings

Name: Default ☒

HTTP

URL:

Access: [Help >](#)

Whitelisted Cookies: [Add](#)

Auth

Basic auth: ☒ With Credentials: ☒

TLS Client Auth: ☐ With CA Cert: ☐

Skip TLS Verify: ☐

Forward OAuth Identity: ☐

Basic Auth Details

User:

Password: [Reset](#)

Custom HTTP Headers

[+ Add header](#)

timefieldに@timestampと入力します。
versionを7.0+にします。
Save & Testを実行すると緑枠が表示されます。

Elasticsearch details

Index name	*	Pattern	No pattern ▾
Time field name	@timestamp		
Version	7.0+ ▾		
Max concurrent Shard Requests	30		
Min time interval	10s		

Logs

Message field name	_source
Level field name	

Data links

Add links to existing fields. Links will be shown in log row details next to the field value.

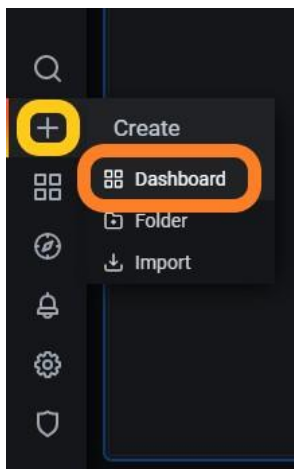
+ Add

✓ Index OK. Time field name OK.

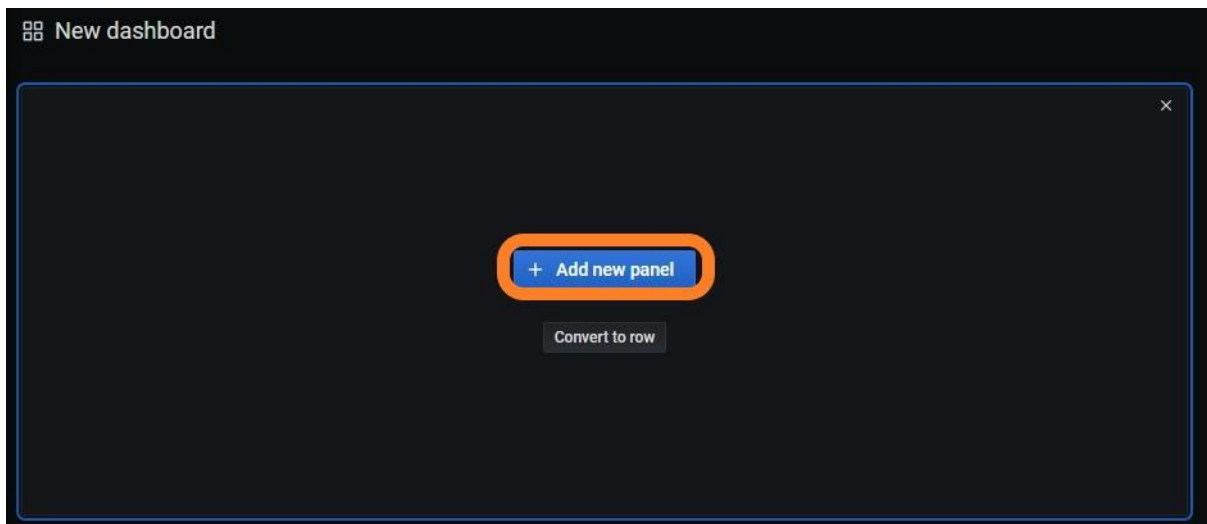
Save & Test Delete Back

Grafanaでグラフの作成

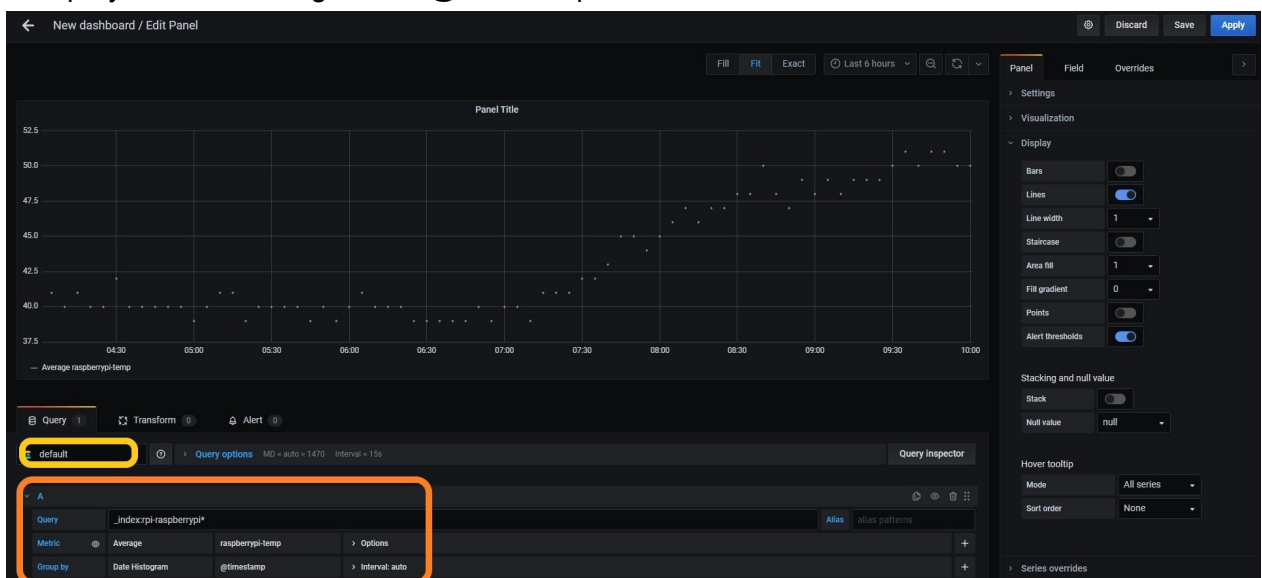
ダッシュボードを選択します。



Add new panelをくりっくします。



接続するデータソースを選択します。
Queryに「_index:rpi-raspberrypi*」
Metricに「Average」、「raspberrypi-temp」
Group by に「Date Histogram」、「@timestamp」と入力します。



panelタイトルに「ラズパイ温度」と入力します。

visualizationで「Gauge」を選択します。



Fieldを選択し、以下の設定を入力します。

unit:Celsius min:20 max:80
Displayname:CPU温度 Threshold:60,50

Panel

Field

Overrides

>

Standard options

Unit

Celsius (°C)

Min

Leave empty to calculate based on all values

20

Max

Leave empty to calculate based on all values

80

Decimals

0

Display name

Change the field or series name

CPU温度

No Value

What to show when there is no value

-

Color scheme

Select palette, gradient or single color

From thresholds

Thresholds

+ Add threshold

60

50

Base

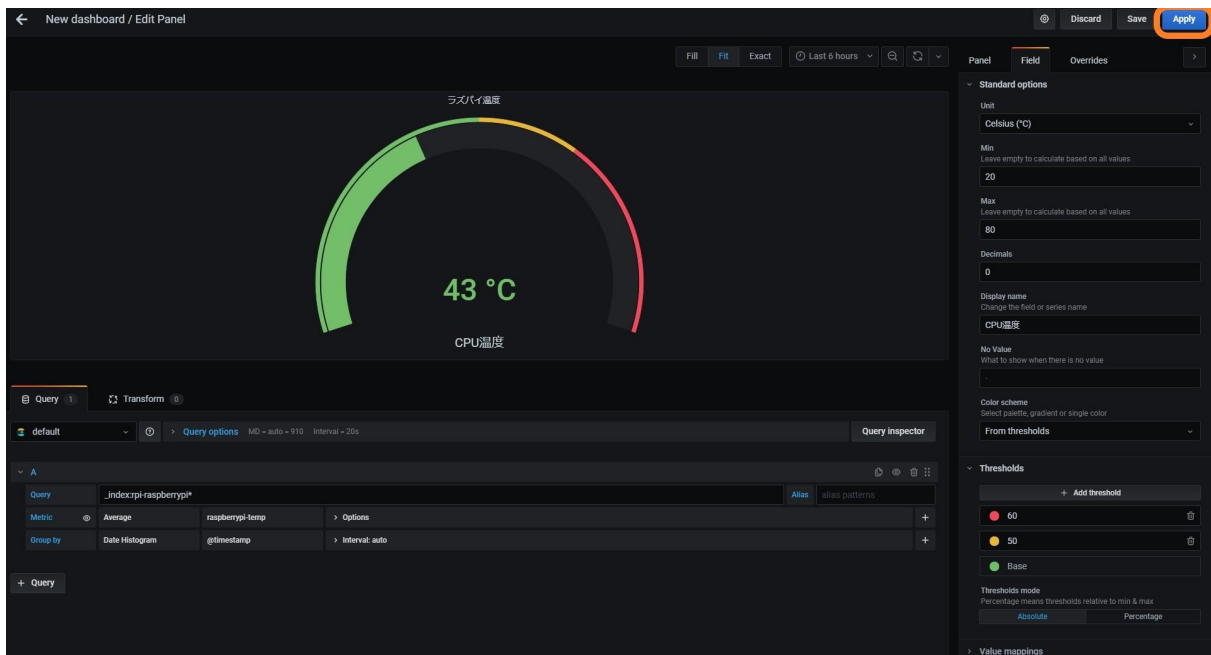
Thresholds mode

Percentage means thresholds relative to min & max

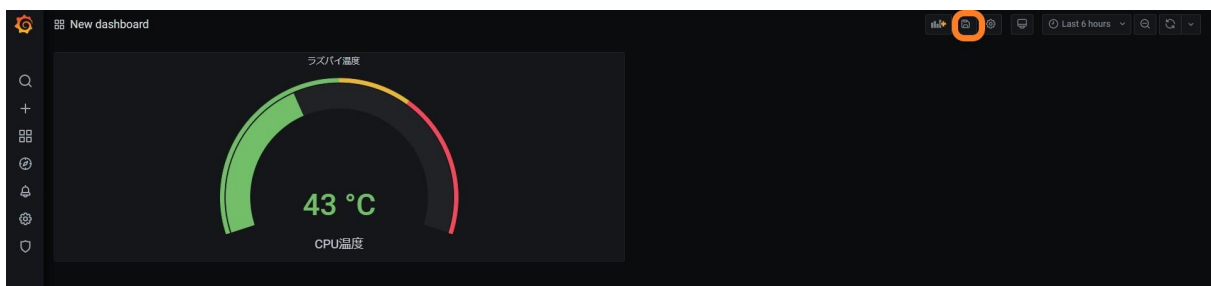
Absolute

Percentage

画像のようなグラフが作成できればグラフ化完了です。
Applyを選択します。



保存ボタンを選択します。



Dashboard nameに「ラズベリーパイ負荷状況」と入力し、保存します。

The screenshot shows the "Save dashboard as..." dialog box. The "Dashboard name" field is highlighted with a yellow circle and contains the text "ラズベリーパイ負荷状況". The "Folder" dropdown menu is set to "General". The "Copy tags" checkbox is unchecked. At the bottom, the "Save" button is highlighted with an orange circle, and the "Cancel" button is also visible.