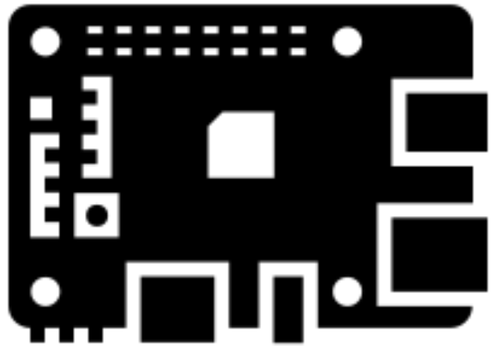
From Small Sensors to Big Queries

It's Serverless unless you include the Sensors



Vincent D. Warmerdam - GDD - @fishnets88 - koaning.io

My problem started around 1890

I bought an old house and I want to start fixing things.

My problem needs fixing in 2018

I bought an old house and I want to start fixing things.

— there's a humidity problem
— there's a temperature leak or two
— keeping doors open between rooms

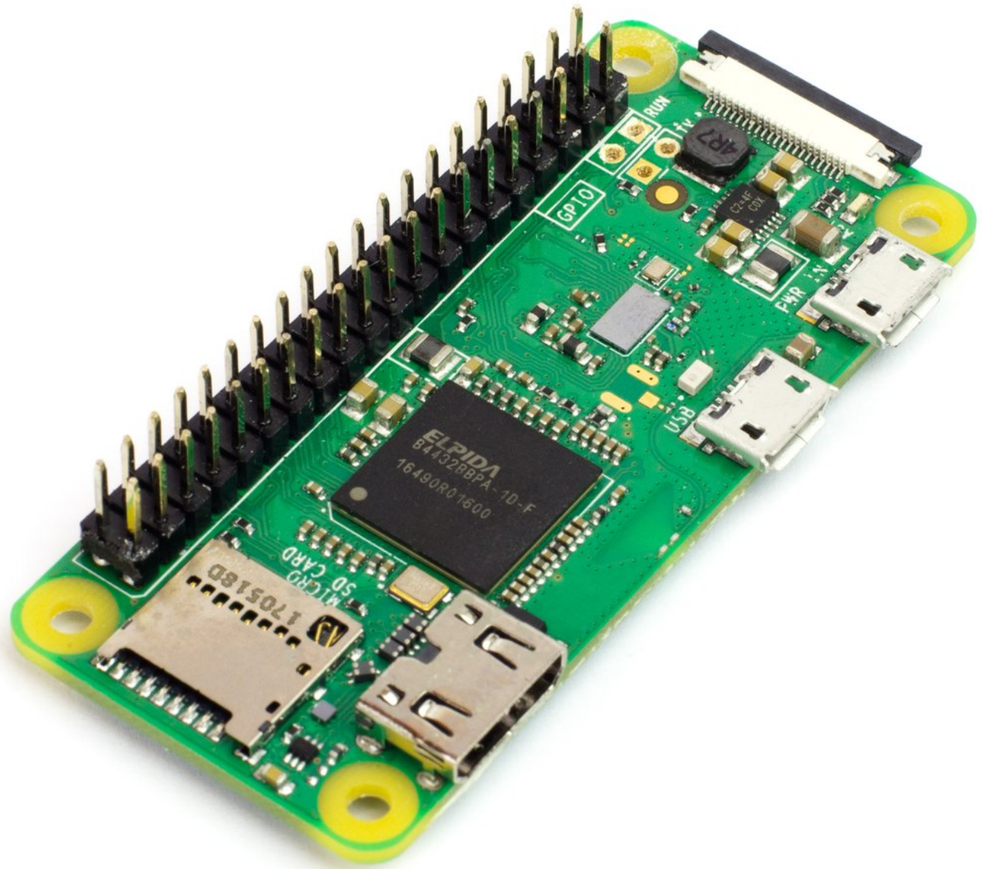
It helps that I'm a bit of a nerd and this is a great excuse to learn about electronics. I also don't mind playing with gcloud.

Today

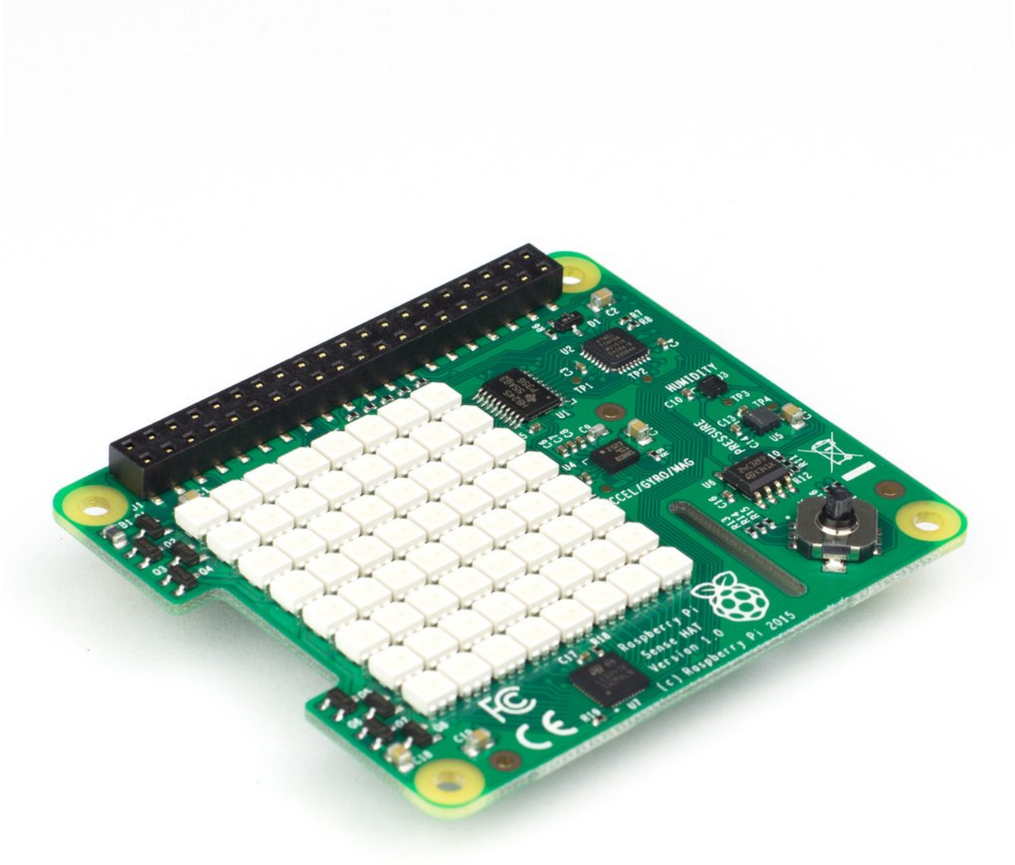
I'll talk about these topics.

— the hardware
— the software
— the cloudware
— why the setup is great
— future stuff

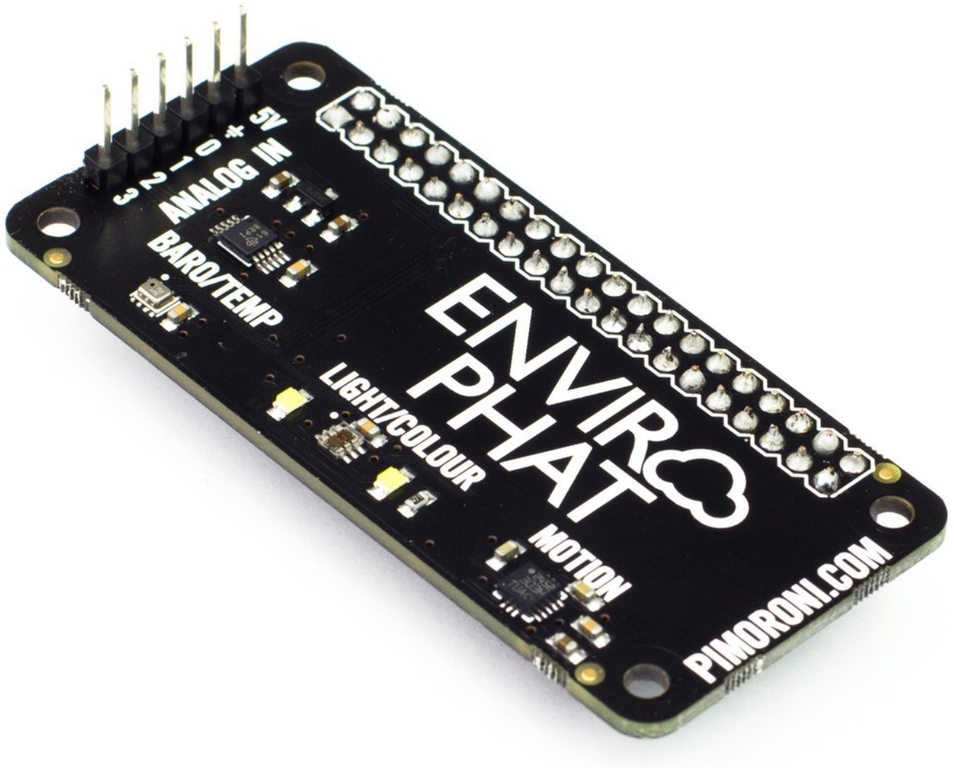
Raspberry Pi Zero: Wifi and PINS!



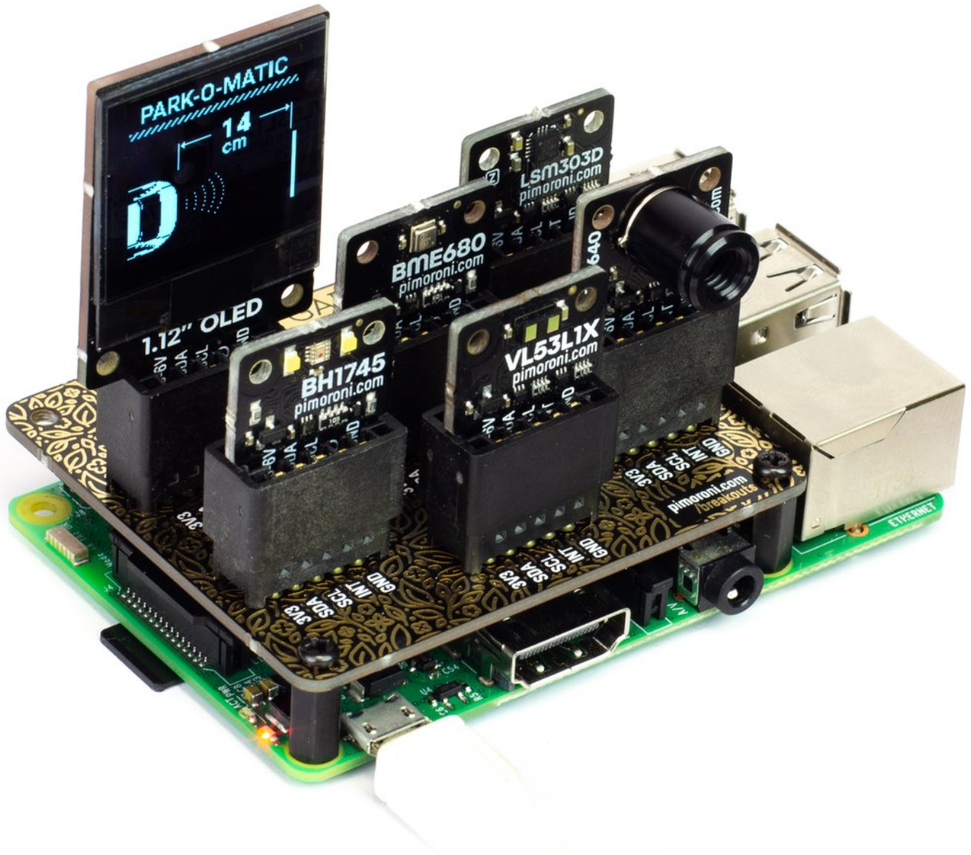
Sensors: Raspberry Pi Hat $35



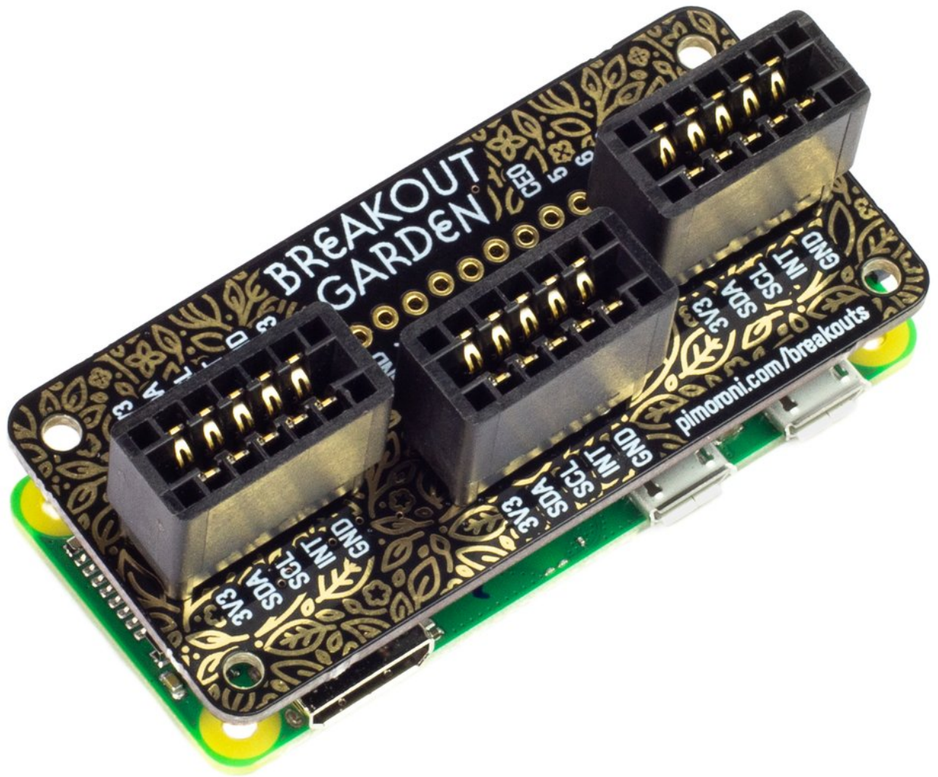
Sensors: Envirophat $11



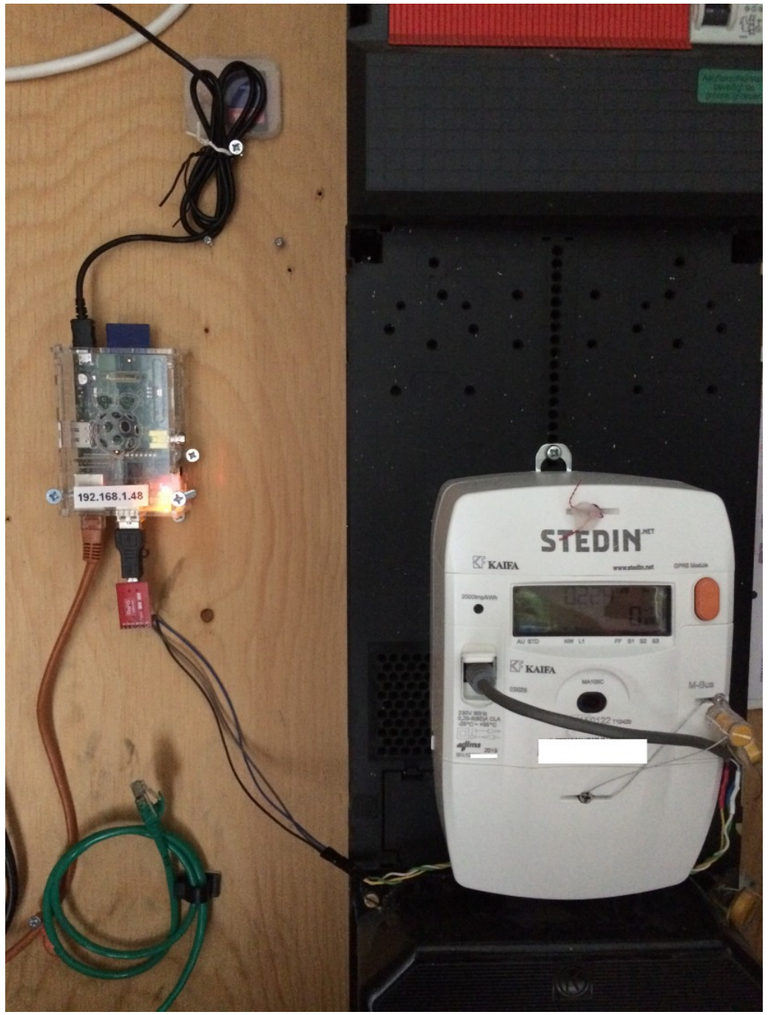
Sensors: Garden Hat $40



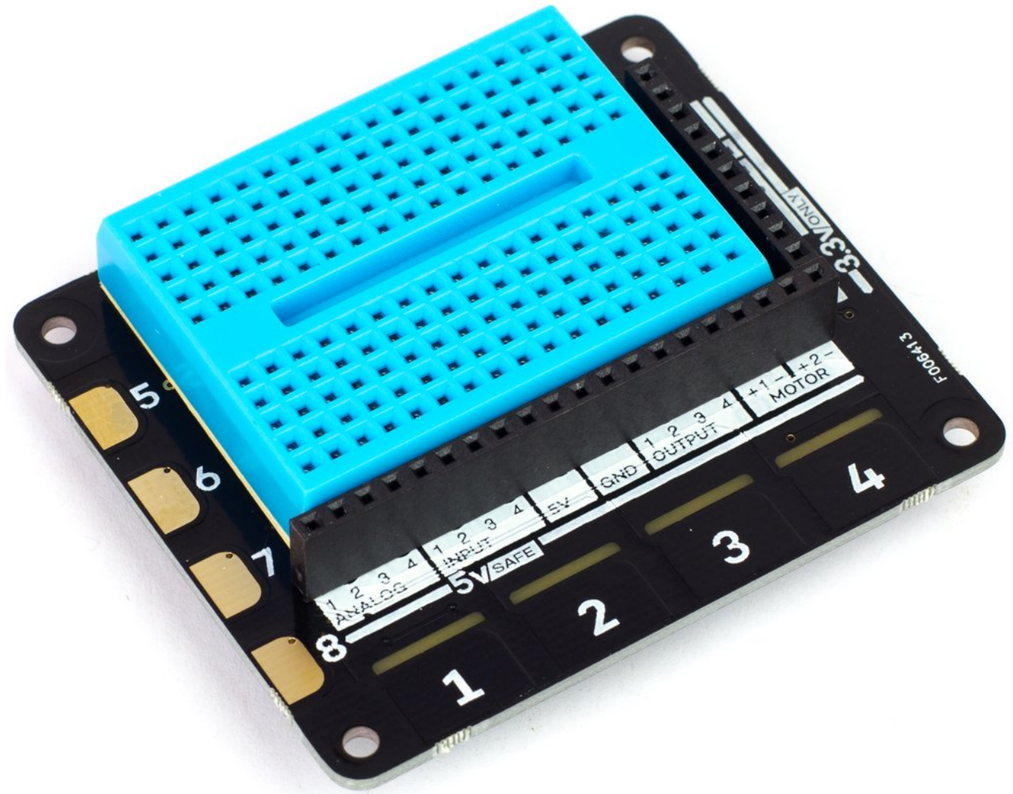
Sensors: Just a Hat $9



Sensors: Cable ($20 ?!)



Sensors: Future



Python: write a class!

```
import bme680

class Measurement:

    def __init__(self):
        with open("/credentials/hostname.j2") as f:
            self.name = f.read()
        os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/path/creds.json'
        self.sensor = bme680.BME680()
        self.sensor.set_humidity_oversample(bme680.OS_2X)
        self.sensor.set_pressure_oversample(bme680.OS_4X)
        self.sensor.set_temperature_oversample(bme680.OS_8X)
        self.sensor.set_filter(bme680.FILTER_SIZE_3)
        self.sensor.set_gas_status(bme680.ENABLE_GAS_MEAS)
        self.sensor.set_gas_heater_temperature(320)
        self.sensor.set_gas_heater_duration(150)
        self.sensor.select_gas_heater_profile(0)
```

Advice: not every raspberry will have every sensor.

```
@property
def temperature(self):
    return self.rowlog('temperature', self.sensor.data.temperature)

@property
def humidity(self):
    return self.rowlog('humidity', self.sensor.data.humidity)

@property
def light(self):
    from bh1745 import BH1745
    bh1745 = BH1745()
    bh1745.setup()
    r, g, b = bh1745.get_rgb_scaled()
    return self.rowlog('light', max([r,g,b]))
```

Advice: not every raspberry will have every sensor.

```
def rowlog(self, name, reading):
    return name, reading, self.name, self.ip, str(dt.datetime.now())

@property
def data_logs(self):
    data = [self.temperature, self.humidity]
    try:
        data.append(self.light)
    except (ImportError, RuntimeError):
        pass
    return data
```

Running the code every minute

Now that I have the python code, I can run it and log my data. But how do I run it on a regular interval?

Running the code every minute

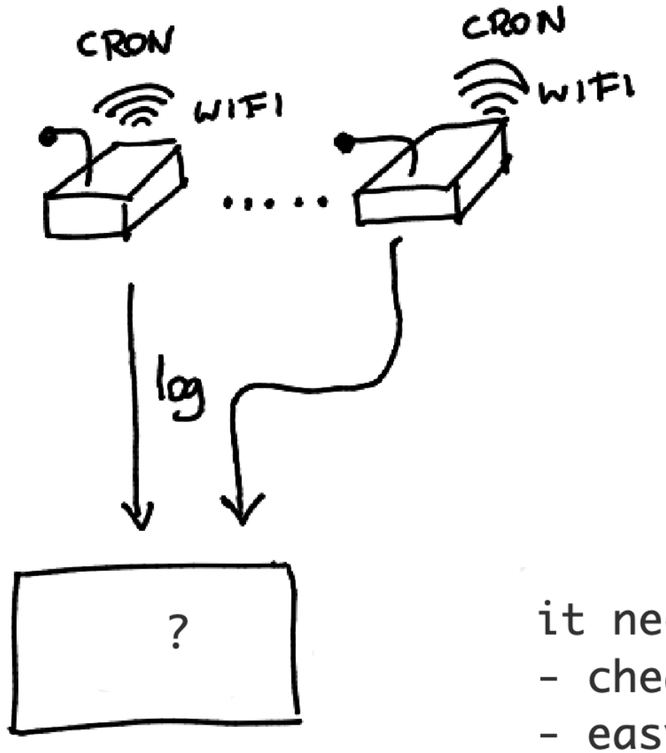
```
> crontab -e
#-----------------------------------------------
# example unix/linux crontab file format:
#-----------------------------------------------
# min,hour,dayOfMonth,month,dayOfWeek command
#
# field          allowed values
# -----          --------------
# minute         0-59
# hour           0-23
# day of month   1-31
# month          1-12 (or names, see below)
# day of week    0-7 (0 or 7 is Sun, or use names)
#-----------------------------------------------

# run the drupal cron process every hour of every day
0 * * * * python /files/python_file_runs_every_hour.py

# run this apache kludge every minute of every day
* * * * * python /files/python_file_runs_every_minute.py
```

Now what?

This kind of works but there are downsides.

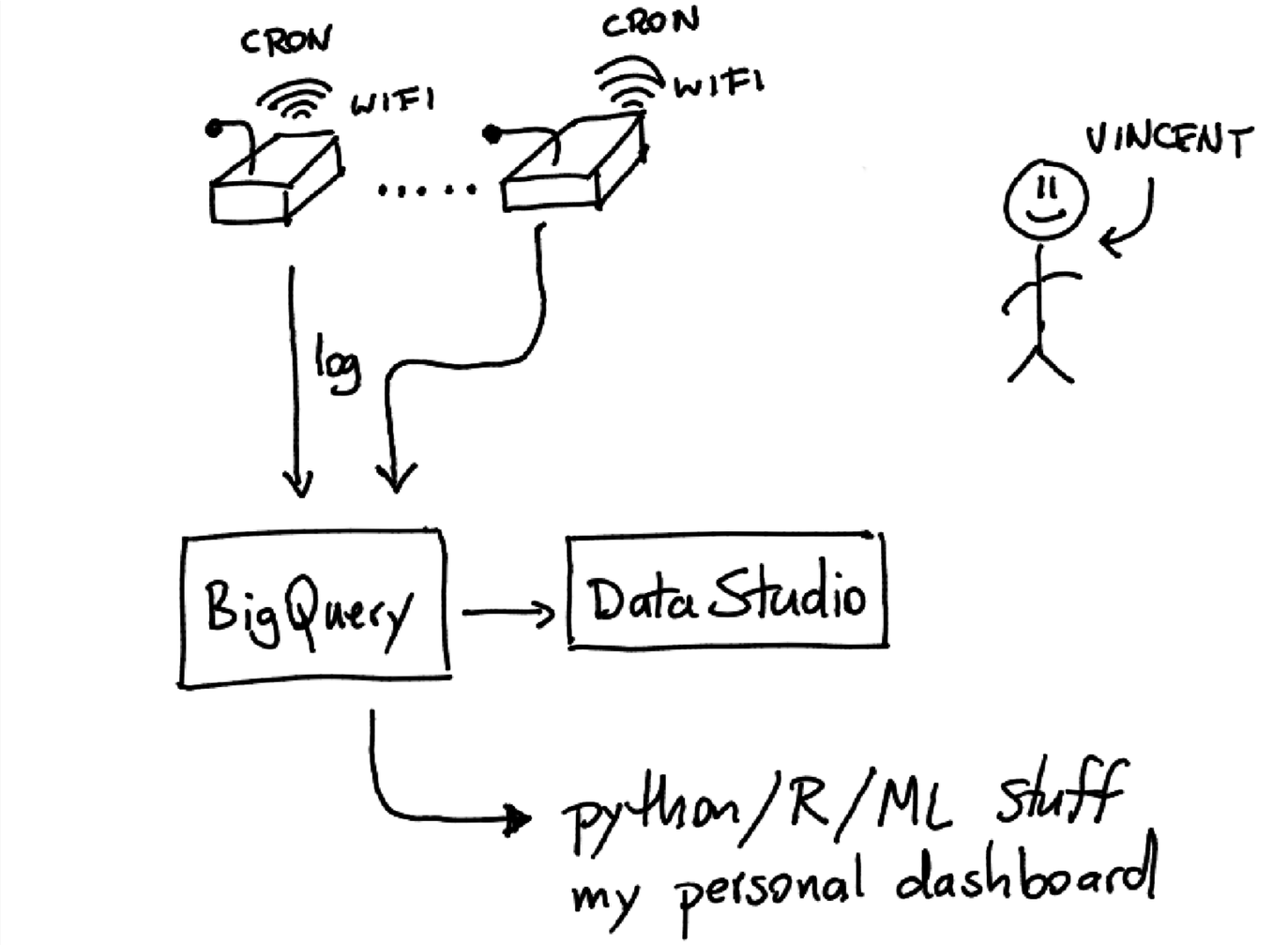
— how do I install everything on all those devices?
— how do I update my code/hardware?
— how can I analyse the data, do I need to manually copy everything?
— automatically copying data from raspberry to my machine poses a security risk
— is there a serverless way?

Situation

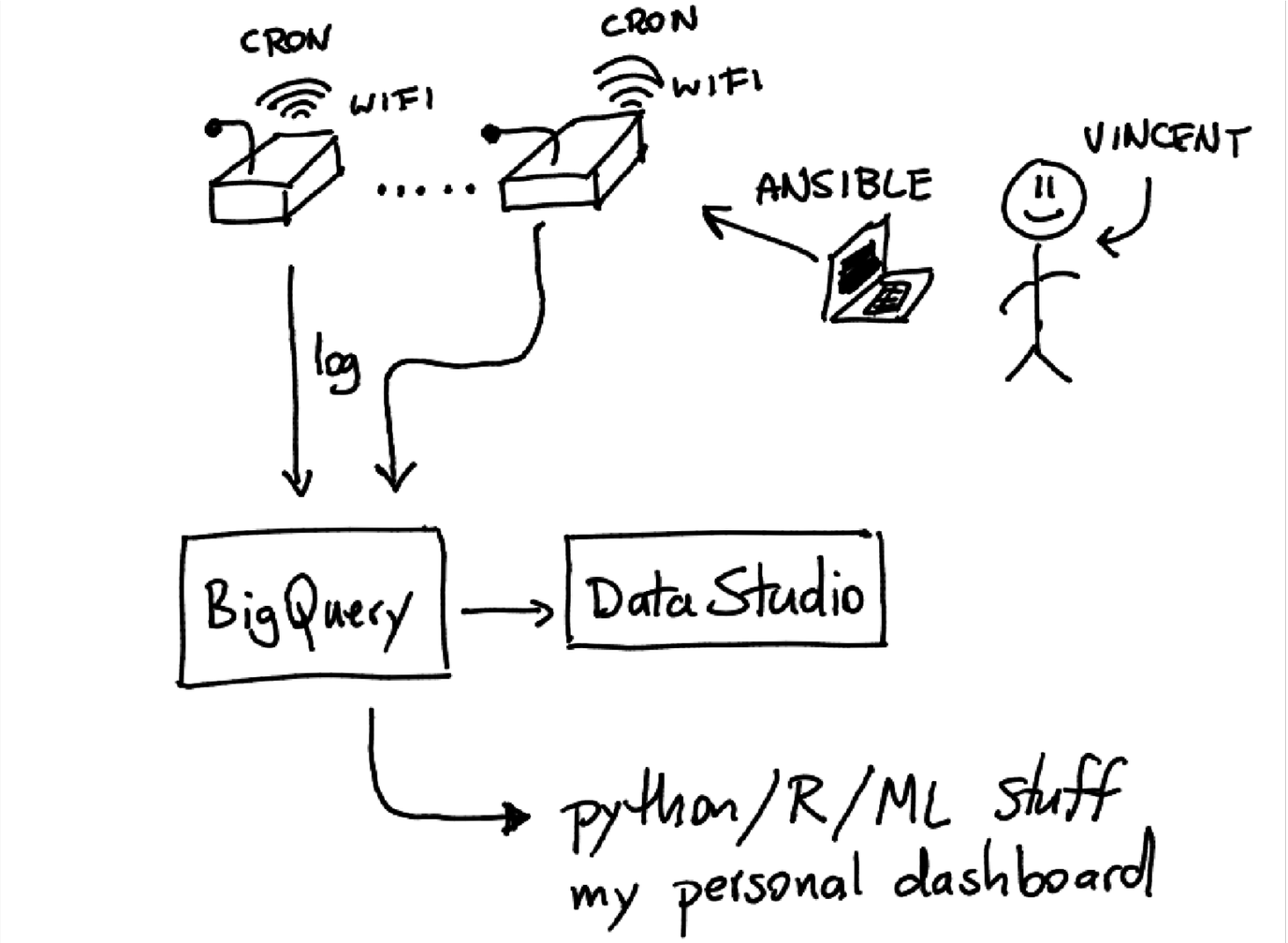


it needs to be
- cheap
- easy to analyse
- robust/backup

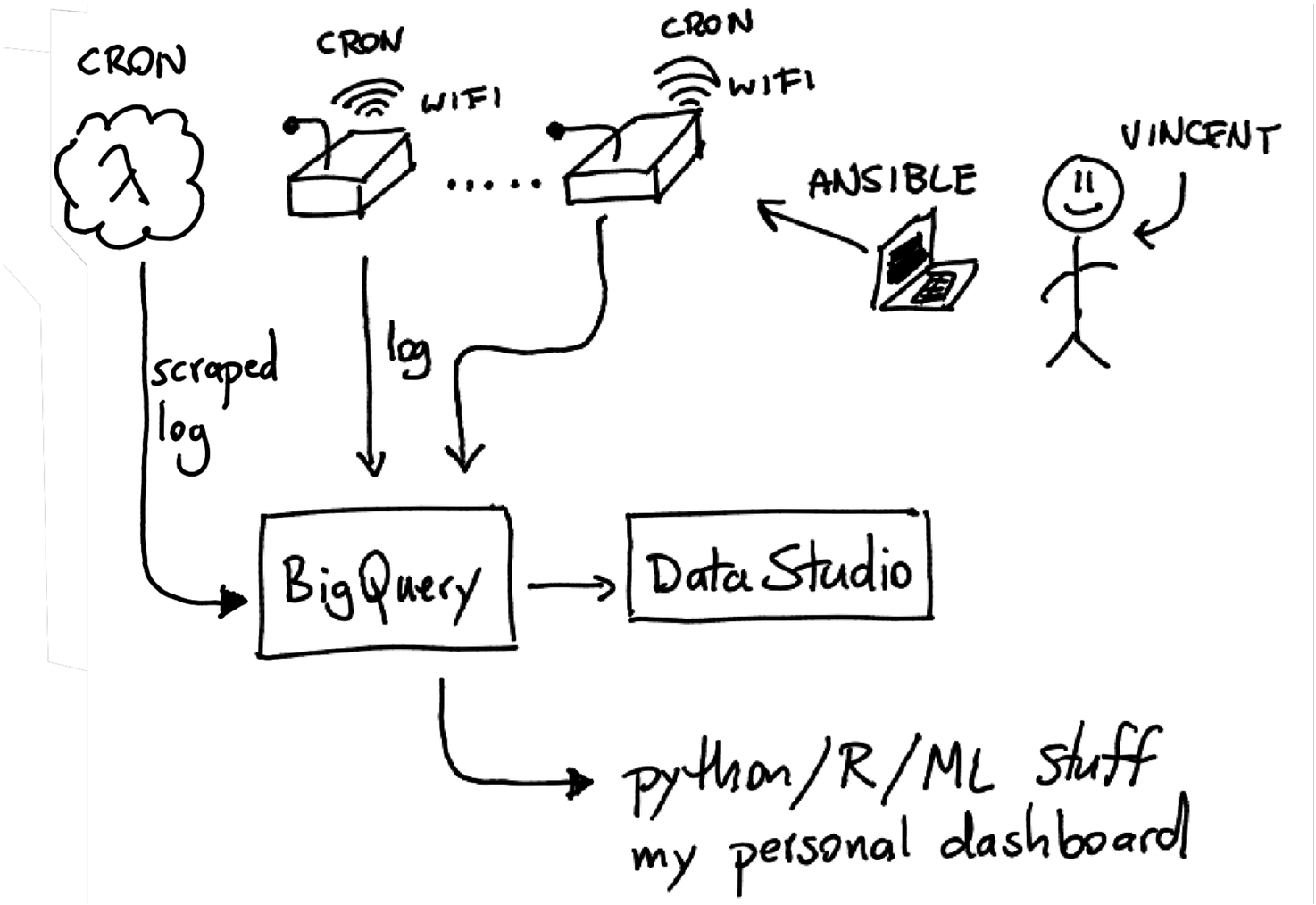
Nearly There



The Setup


CRON
WIFI
CRON
WIFI
ANSIBLE
VINCENT
log
BigQuery
Data Studio
python/R/ML stuff
my personal dashboard

The Setup: Benefits!



Observation

Spending some time thinking about how you want to do this is important. It merely takes pen and paper and it seriously saves a lot of time in the long run.

Ansible

```
> tree -L 2
.
├── README.md
├── ansible
│   ├── hostname.j2
│   ├── hosts
│   ├── provision-cronjobs-rasp-electro.yml
│   ├── provision-cronjobs-rasp-zero.yml
│   ├── provision-files-rasp3.yml
│   └── provision-tools-rasp3.yml
```

Ansible

Ansible uses your ssh config.

```
# ~/.ssh/config
Host rpi-zero-attic
    HostName 123.456.789.1
    User pi
    IdentityFile ~/.ssh/home-rpi-keyfile
Host rpi-zero-tv
    HostName 123.456.789.2
    User pi
    IdentityFile ~/.ssh/home-rpi-keyfile
...
...
```

Ansible

You can assign groups in the `hosts` file.

```
[raspberries]
rpi-zero-attic ansible_user=pi
rpi-zero-tv ansible_user=pi
rpi-zero-catroom ansible_user=pi
rpi-zero-bedroom ansible_user=pi
rpi-zero-curtain ansible_user=pi
rpi-zero-sewing ansible_user=pi
rpi-zero-kitchen ansible_user=pi

[electro]
rpi-electro ansible_user=pi
```

Ansible .yml files

```
- hosts: raspberries, electro
  become: yes
  become_user: root
  tasks:
    - name: make sure that we have most recent apt
      command: apt-get update

    - name: install all the apt get stuff, incl fail2ban
      apt: name={{item}} state=installed
      with_items:
        - fail2ban
        - postfix
        - build-essential

    - name: pip install requirements globally
      pip: name={{item}} state=present
      with_items:
        - google-cloud-bigquery
        - ipython
```

Ansible .yml for cron

```
- hosts: raspberries
  become: yes
  become_user: root
  tasks:
    - name: remove old humidity/temperature cronjob
      cron:
        name="humiditemp"
        state=absent
        user=pi
    - name: add new humidity/temperature cronjob
      cron:
        name="humiditemp"
        minute="*"
        user=pi
        job="sudo /usr/bin/python /loggers/cron-scripts/measure.py"
```

Updates are easy.

It was extra work, but locally I can update everything now in parallel with commands like this:

```
ansible -i hosts all -m ping
ansible-playbook -i hosts provision-tools-rasp3.yml
ansible-playbook -i hosts provision-files-rasp3.yml
ansible-playbook -i hosts provision-cronjobs-rasp3.yml
```

This is great. Work from one machine, deploy to everything!

BigQuery

For most intents and purposes, BigQuery is a simple/ cheap way to store/analyse a large table. The small code below just appends data to a table.

```
def postbigq(request):
    measure = Measurement()
    os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = '/path/file.json'
    client = bigquery.Client()
    dataset_id = '<ID_FOR_MY_DATASET>'
    table_id = '<ID_FOR_MY_TABLE>'
    table_ref = client.dataset(dataset_id).table(table_id)
    table = client.get_table(table_ref)
    errors = client.insert_rows(table, measure.datarows)
    assert errors == []
```

Sensor in the Garden

I was considering installing a raspberry outside with solar panels and everything.

Sensor in the Garden

I was considering installing a raspberry outside with solar panels and everything.

I'll still do this someday, but hardware takes more time than software and you cannot CMD-c + CMD-v in real life. So instead of measuring the weather outside of my house, I figured scraping the weather APIs would be a much better idea.

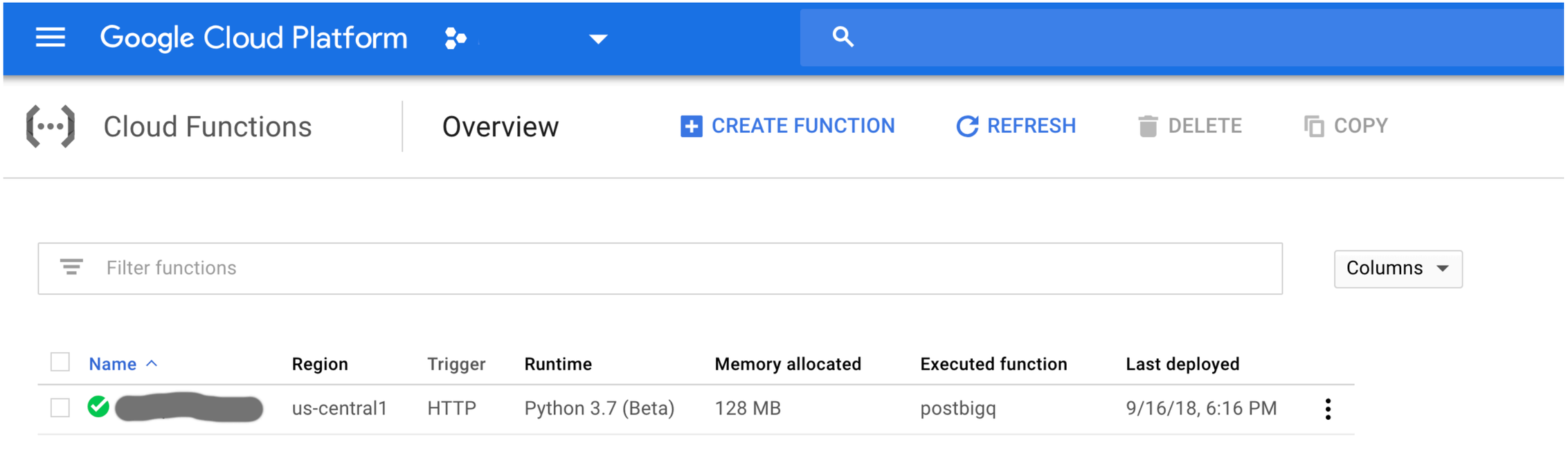
Sensor in the Garden

I was considering installing a raspberry outside with solar panels and everything.

I'll still do this someday, but hardware takes more time than software and you cannot CMD-c + CMD-v in real life. So instead of measuring the weather outside of my house, I figured scraping the weather APIs would be a much better idea.

It was.

Cloud function



Cloud function

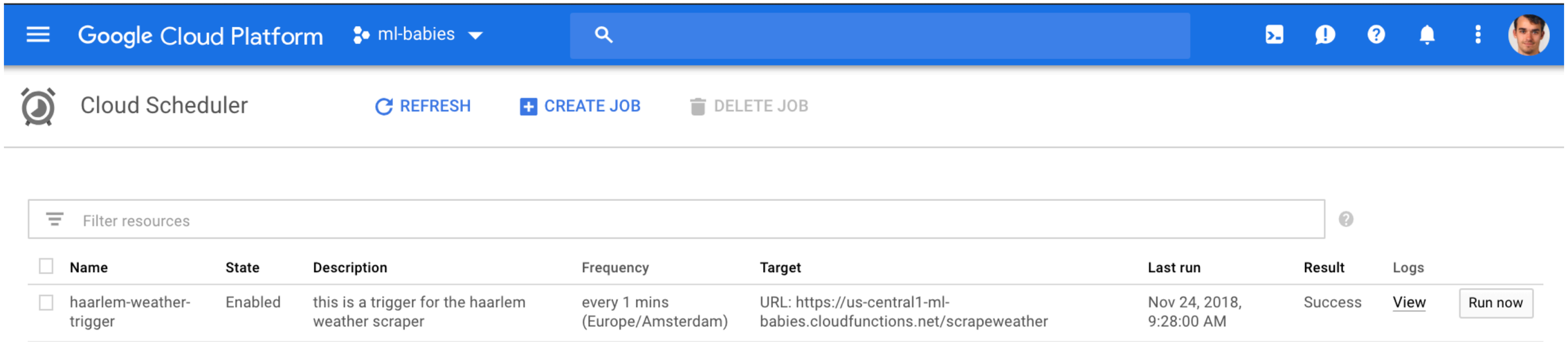
I wrote another `Measure` python class that regards APIs as if they were sensor outputs. Much of the code could be re-used.

```
> gcloud beta functions deploy scrapeweather
    --entry-point postbigq
    --project ml-babies
    --runtime python37
    --trigger-http
```

It's not just the scraper that is serverless ...

Cloud Scheduler

Even cron is serverless in my stack.



Costs

Per day I log about.

$$3 \text{ sensors} \times 7 \text{ devices} \times 1440 \text{ readings/day} \approx 30\text{K rows/day}$$

I've been running this for about 3 months.

$$30\text{K rows/day} \times 90 \text{ days} \approx 2.7\text{M rows}$$

This totals to about 150MB of data, with a dumb schema.

Costs

That's 600MB per year.

Considering the storage pricing ...

$$\$0.020\mathrm{GB}^{-1} \times 0.6\mathrm{GB} \times 12\ \mathrm{months} \approx \$0.12\ \mathrm{year}^{-1}$$

Considering the query pricing ...

$$\frac{1,000,000\ \mathrm{MB}}{600\ \mathrm{MB}} \approx 1667\ \mathrm{years\ before\ I\ start\ paying}$$

Demo

Pulling data from BQ can be done easily from `pandas` in python or `dplyr` in R. I'll give a demonstration of shiny now to show you what I am currently measuring.

Final Tips

If you're going to do this yourself, think about sensors! The heat from the raspberry influences the temperature/ humidity sensor. So does sunlight!

Either do some hardware work and move the sensors away from the raspberry. Or apply consider **physics**.

$$\text{actual temperature} = f(\text{sensor temp}, \text{cpu temp})$$

Either way, consider that sensors are **always biased**.

Final Tips

It's kind of a long story, but the wifi router that connects my entire house does not offer static ip's. Hence; it might make sense to log the ip-address of your raspberry device.

That way, if the ip-address is re-assigned you can check the data you're logging to figure out what the most recent ip dress was of each sensor.

Executive Summary: Thy Heroes

— raspberry pi
— pimoroni sensors
— ansible
— gcloud
— python
— rstudio

But the epic summary is that my only servers are the sensors themselves. The world really is changeing.

Appendix: ML for BigQuery [beta]

```
CREATE MODEL `models.natality_model`
OPTIONS
  (model_type='linear_reg',
    input_label_cols=['weight_pounds']) AS
SELECT
  weight_pounds,
  is_male,
  gestation_weeks,
  mother_age,
  CAST(mother_race AS string) AS mother_race
FROM
  `bigquery-public-data.samples.natality`
WHERE
  weight_pounds IS NOT NULL
  AND RAND() < 0.001
```