Building Distributed System
with Celery on Docker Swarm
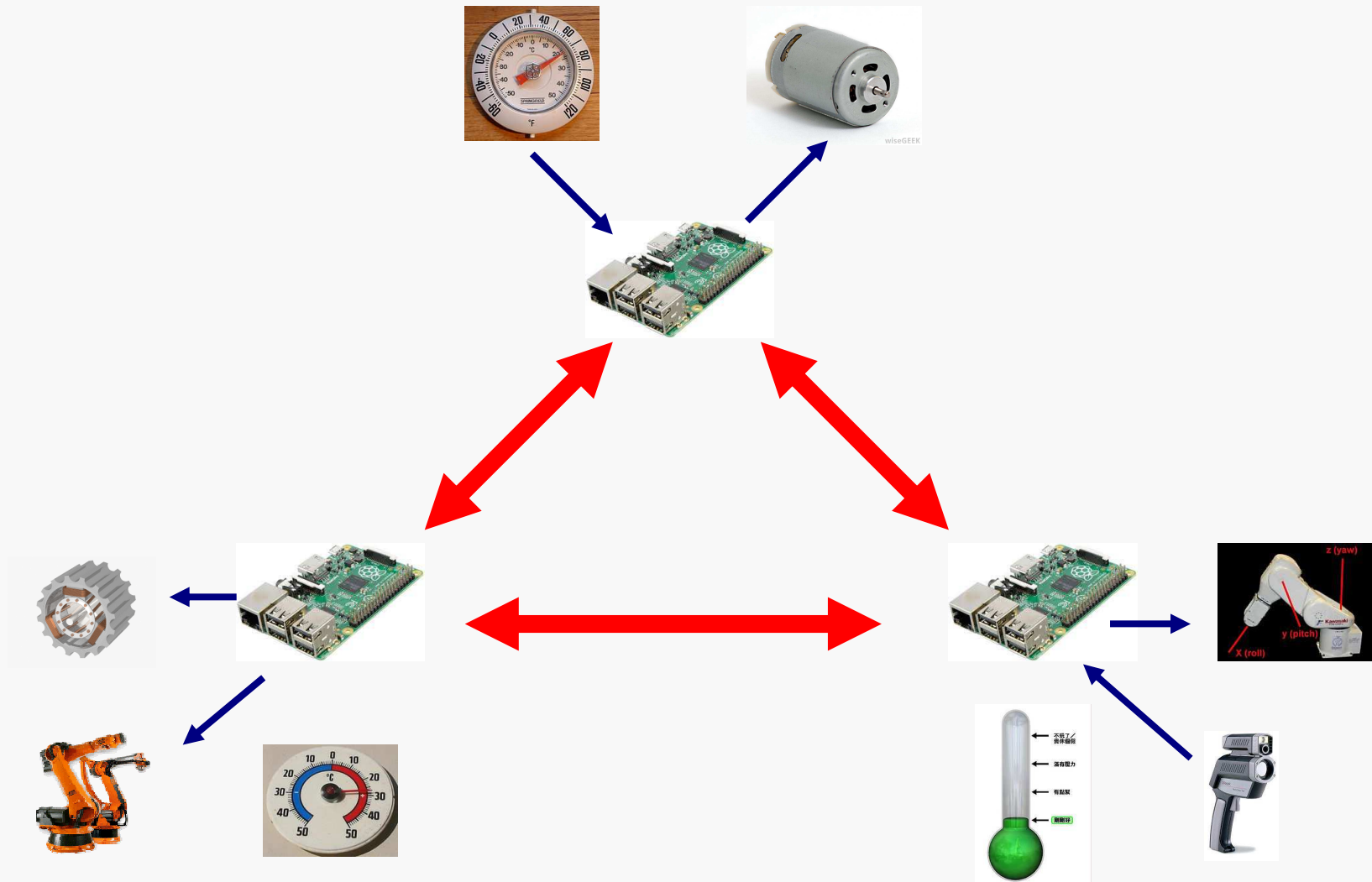
Wei Lin

PyCon JP 2016

2016/09/22

About Me

- Wei Lin
- @Wei_1144
- Wei1234c@gmail.com

- Mostly worked in the fields of:
  - marketing
  - strategy-planning
  - project-management

Parallel computing

https://resin.io/blog/what-would-you-do-with-a-120-raspberry-pi-cluster/
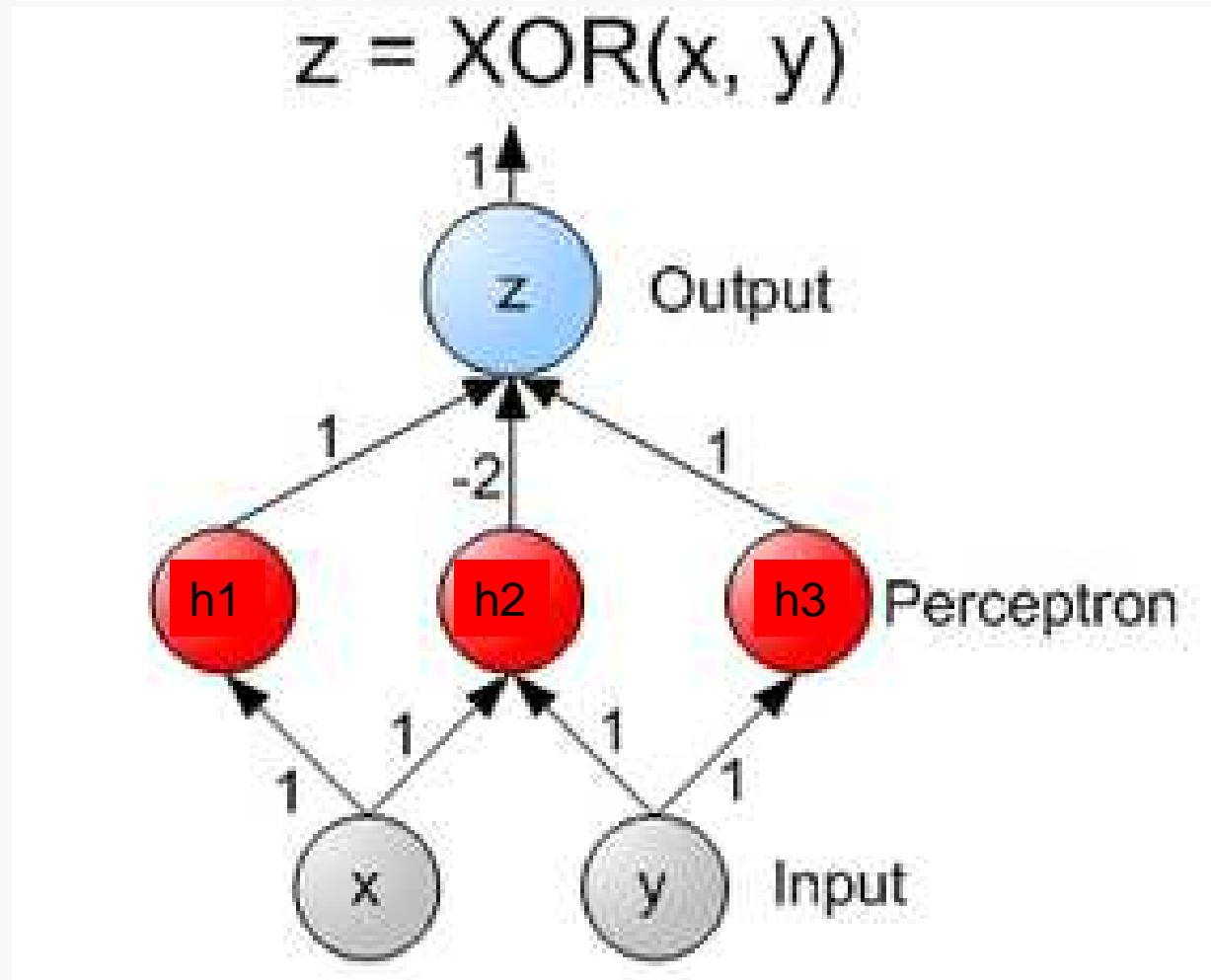
Distributed System
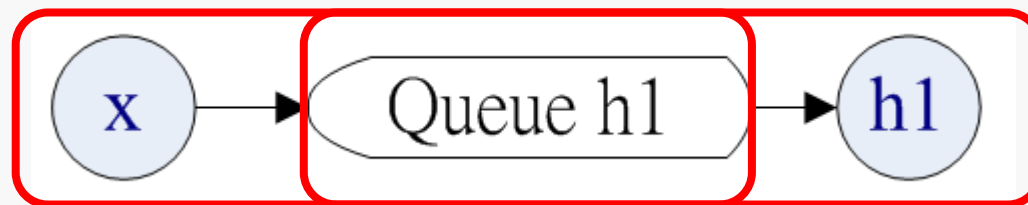
Distributed system

Distributed system (II)

Neural network
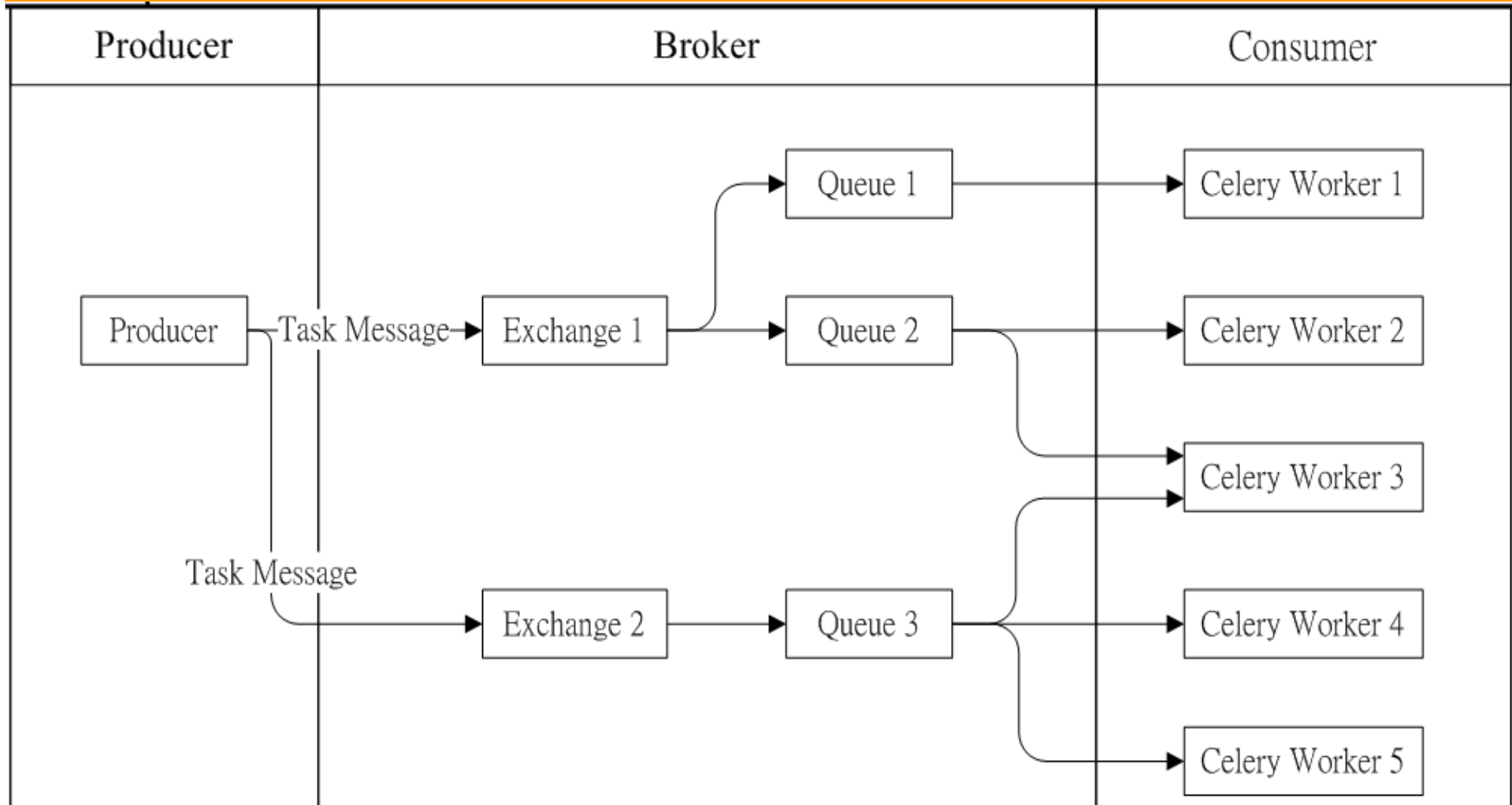
Communicate via Celery

Celery

Celery flowchart

Send message to a specific queue

Send message

```
from celery import Celery


app = Celery()
app.config_from_object('celeryconfig')

@app.task
def kick(neuron_id):
    myName = getHostname()
    log("{0} is kicking {1}.".format(neuron_id, myName))
    ...
```

kick.apply_async(['neuron_x'])

kick.apply_async(['neuron_x'], routing_key = 'neuron_h1')

12

Configuration file

```python
#_____Import Kombu classes_____
from kombu import Exchange, Queue


#_____CELERY_TIMEZONE & Misc._____
CELERY_TIMEZONE = 'Asia/Taipei'
CELERYD_POOL_RESTARTS = True


#_____BROKER_URL_____
BROKER_URL = 'redis://192.168.0.114:6379/0'


#_____CELERY_RESULT_BACKEND_____
CELERY_RESULT_BACKEND = 'redis://192.168.0.114:6379/1'


#_____CELERY_IMPORTS_____
CELERY_IMPORTS = ('IoT.neuron',)


#_____CELERY_QUEUES_____
CELERY_QUEUES = (
    Queue('neuron_x', Exchange('celery', type = 'direct'), routing_key='neuron_x'),
    Queue('neuron_y', Exchange('celery', type = 'direct'), routing_key='neuron_y'),
    Queue('neuron_h1', Exchange('celery', type = 'direct'), routing_key='neuron_h1'),
    Queue('neuron_h2', Exchange('celery', type = 'direct'), routing_key='neuron_h2'),
    Queue('neuron_h3', Exchange('celery', type = 'direct'), routing_key='neuron_h3'),
    Queue('neuron_z', Exchange('celery', type = 'direct'), routing_key='neuron_z'),
)
```

Task Message Example

```json
{"id": "4cc7438e-afd4-4f8f-a2f3-f46567e7ca77",
 "task": "celery.task.kick",
 "args": ['neuron_x'],
 "kwargs": {},
 "retries": 0,
 "routing_key": "neuron_h1"}
```

Routing Tasks

http://docs.celeryproject.org/en/latest/userguide/routing.html

```
#_____CELERY_QUEUES_____
CELERY_QUEUES = (
    Queue('neuron_x', Exchange('celery', type = 'direct'), routing_key='neuron_x'),
    Queue('neuron_y', Exchange('celery', type = 'direct'), routing_key='neuron_y'),
    Queue('neuron_h1', Exchange('celery', type = 'direct'), routing_key='neuron_h1'),
    Queue('neuron_h2', Exchange('celery', type = 'direct'), routing_key='neuron_h2'),
    Queue('neuron_h3', Exchange('celery', type = 'direct'), routing_key='neuron_h3'),
    Queue('neuron_z', Exchange('celery', type = 'direct'), routing_key='neuron_z'),
)
```

Assign an unique worker
for a specific queue

Starting the Worker

```
$ celery -A IoT worker -n neuron_h1 \
    -Q neuron_h1 --concurrency=1
```

celery_projects
  IoT

celery.py
neuron.py

```python
from celery import Celery
from kombu import Exchange, Queue


app = Celery()
app.config_from_object('celeryconfig')


if __name__ == '__main__':
    app.start()
```
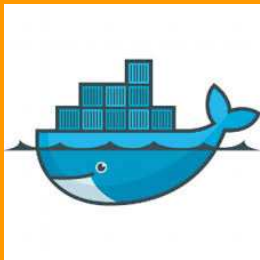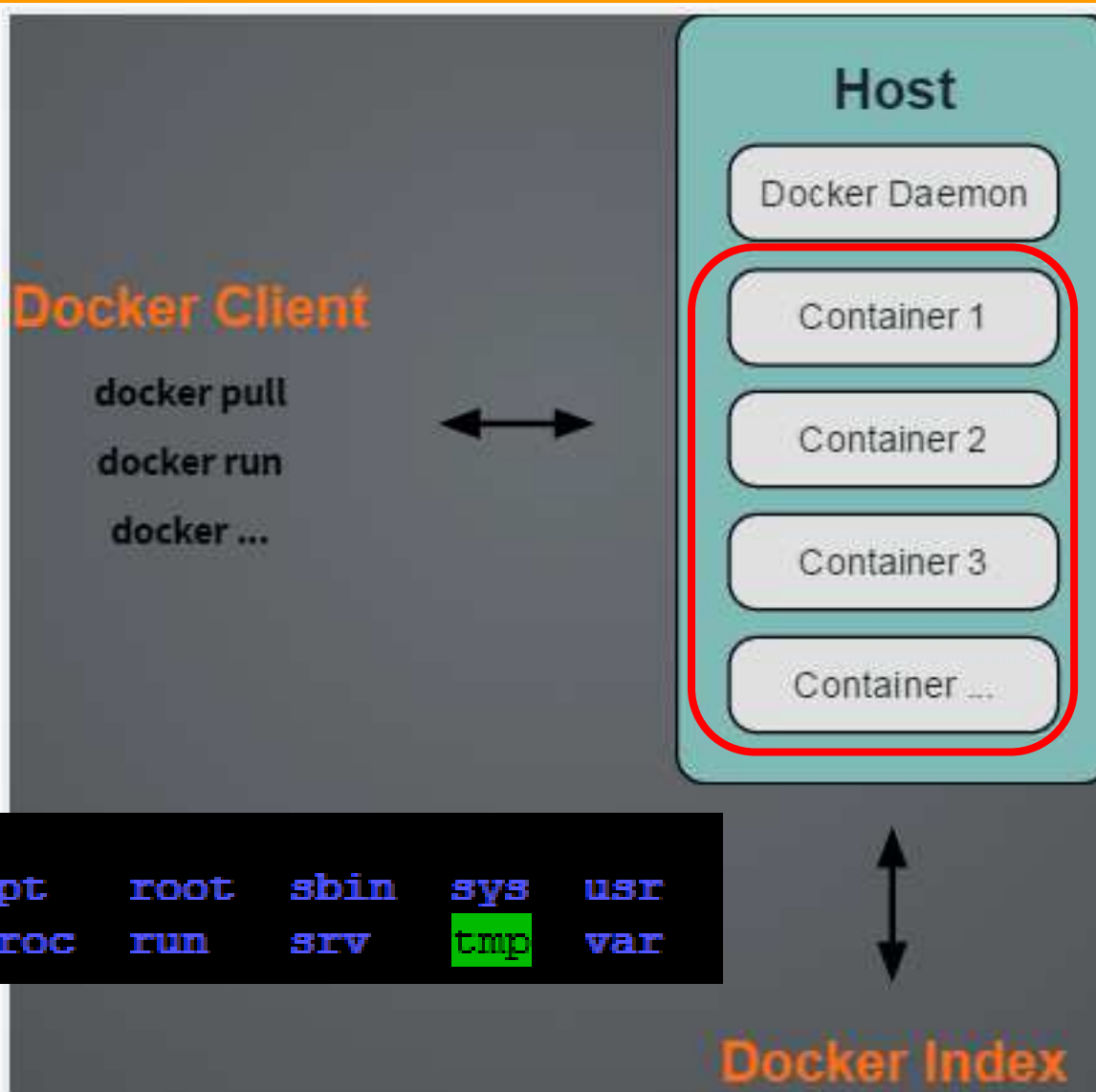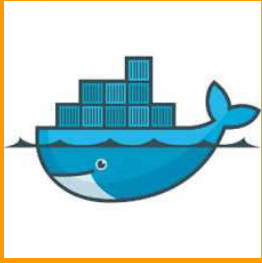
17

Docker

Host and containers

Docker Client

docker pull

docker run

docker ...

Host

Docker Daemon

Container 1

Container 2

Container 3

Container ...

Docker Index

```
root@a02e49d5ef22:/# ls
bin    dev   home   media   opt    root   sbin   sys   usr
boot   etc   lib    mnt     proc   run    srv    tmp   var
```
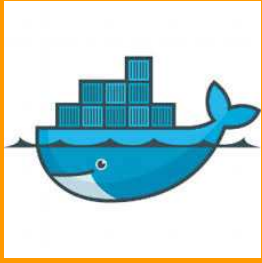
Run worker container

```
docker run -d --name=neuron_x --hostname=neuron_x \
--volume=/data/celery_projects:/celery_projects \
wei1234c/celery_armv7 \
/bin/sh -c \
"cd /celery_projects && \
celery -A IoT worker -n %h -Q neuron_x --concurrency=1"
```
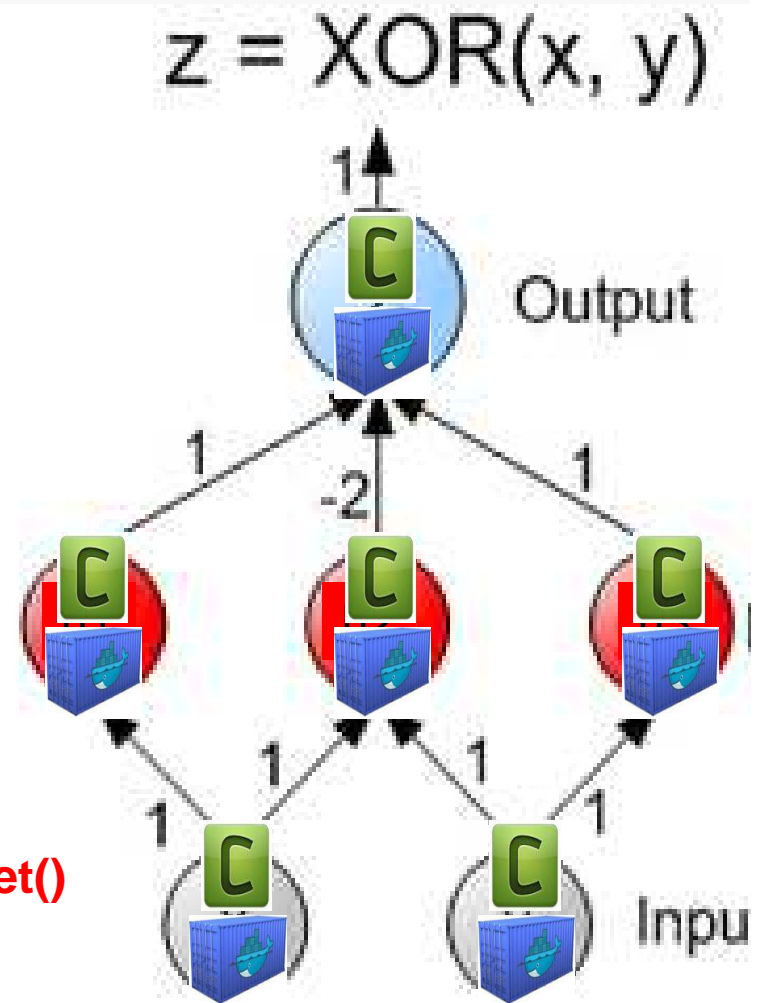
```
docker run -d --name=neuron_y --hostname=neuron_y \
--volume=/data/celery_projects:/celery_projects \
wei1234c/celery_armv7 \
/bin/sh -c \
"cd /celery_projects && \
celery -A IoT worker -n %h -Q neuron_y --concurrency=1"
```

Deploy containers
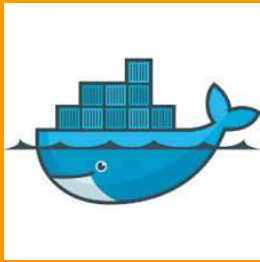
Celery worker containers as neurons

z = XOR(x, y)

kick.**apply_async**(routing_key = 'neuron_h1')

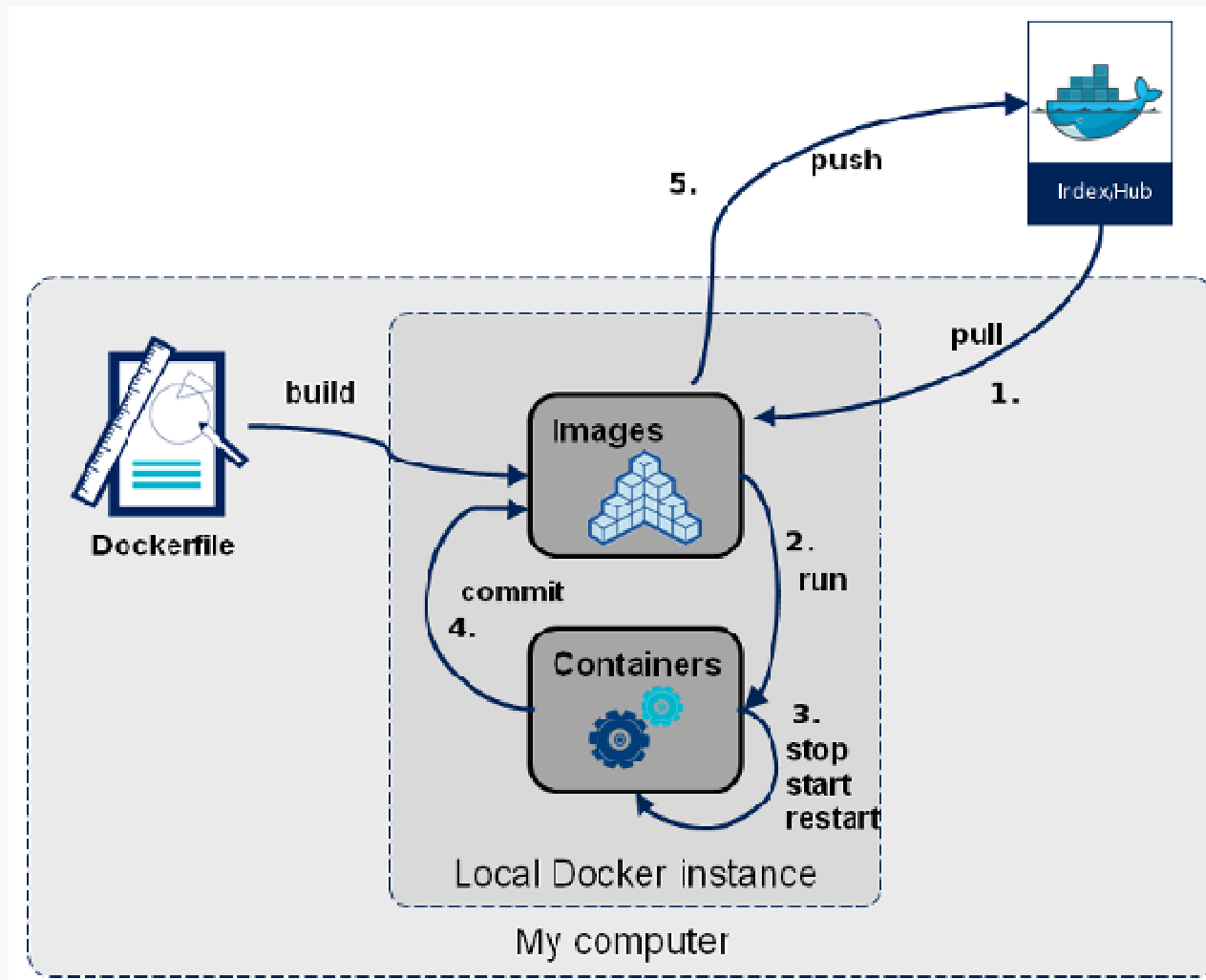getData.**apply_async(**routing_key = 'neuron_h1') **.get()**

"Celery + Docker-Swarm"

So ?

I think…

Docker file/image/container

Think OO

```
# Celery Worker Dockerfile
# 20151229

FROM wei1234c/python_armv7

MAINTAINER Wei Lin
ENV TERM linux
# Add user pi
RUN \
    useradd -G adm,sudo,users -s /bin/bash -m pi && \
    echo 'pi:raspberry' | chpasswd
# Install Celery
RUN \
    pip3 install -U redis celery flower sqlalchemy pymongo

WORKDIR /data
USER pi
EXPOSE 5555
CMD ["bash"]
```
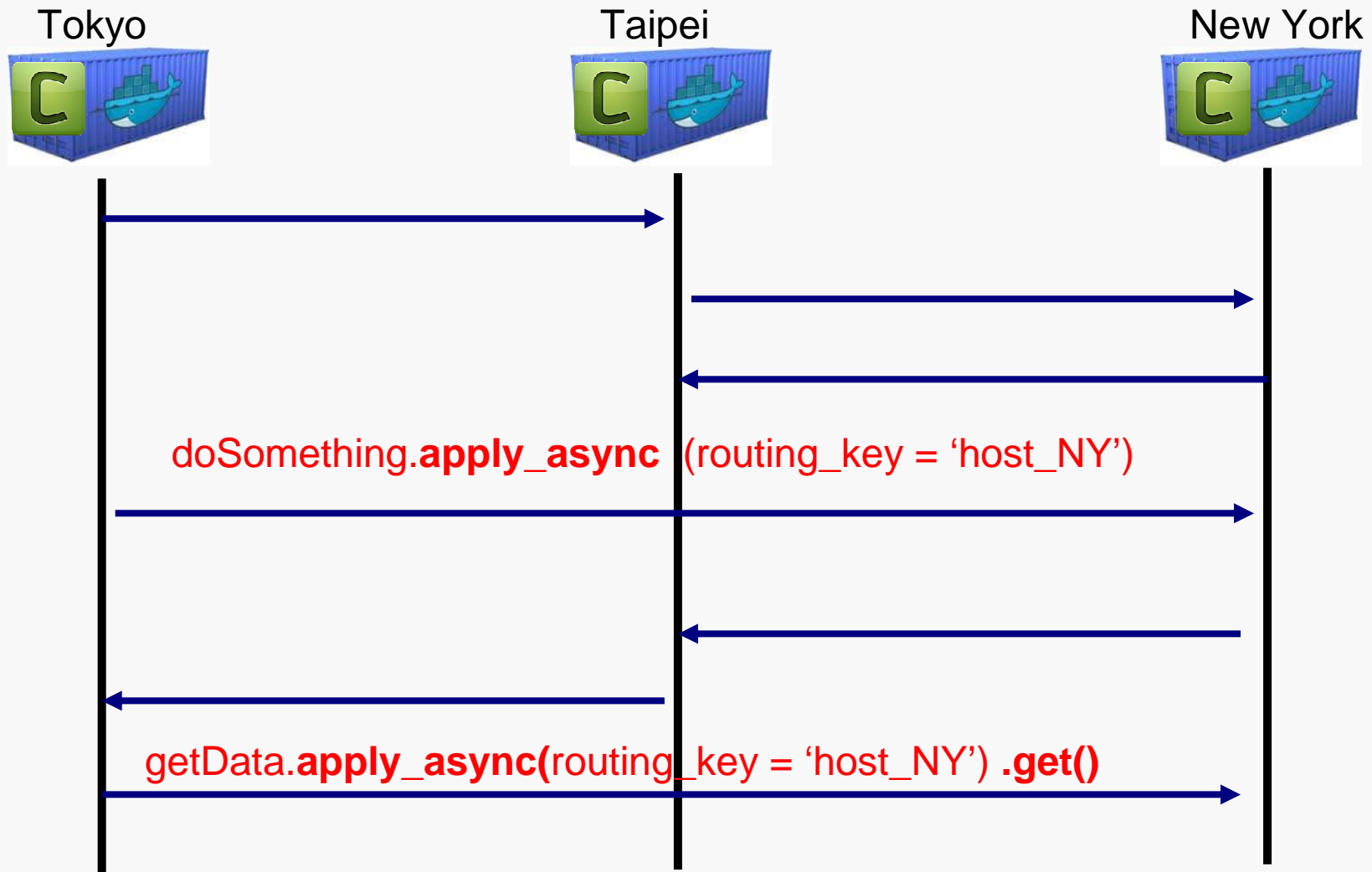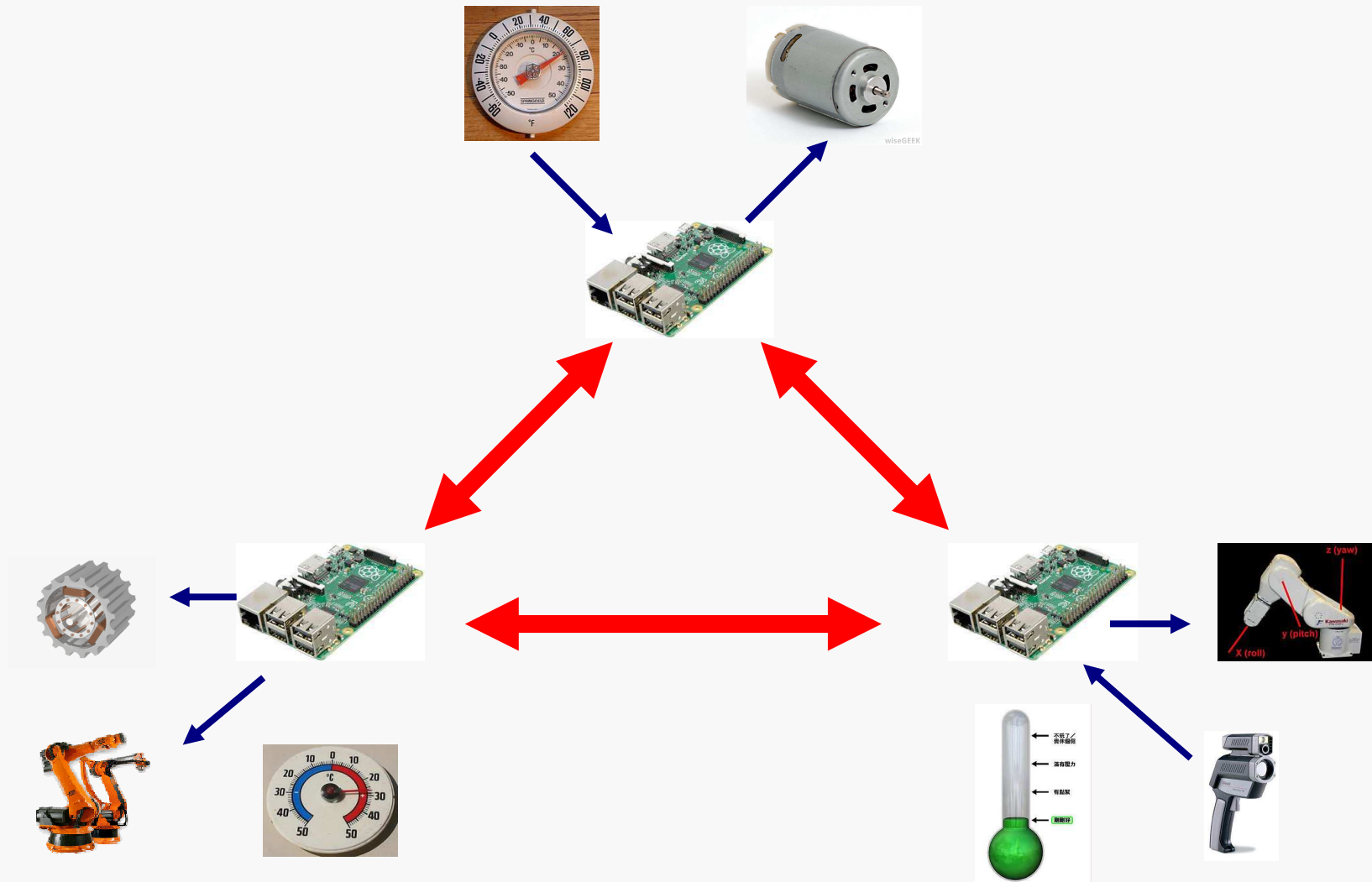
- "From" to Inherit
- Dockerfile to Encapsulate
- Docker Image as Class
- Docker Container as Object

Docker Swarm & Celery
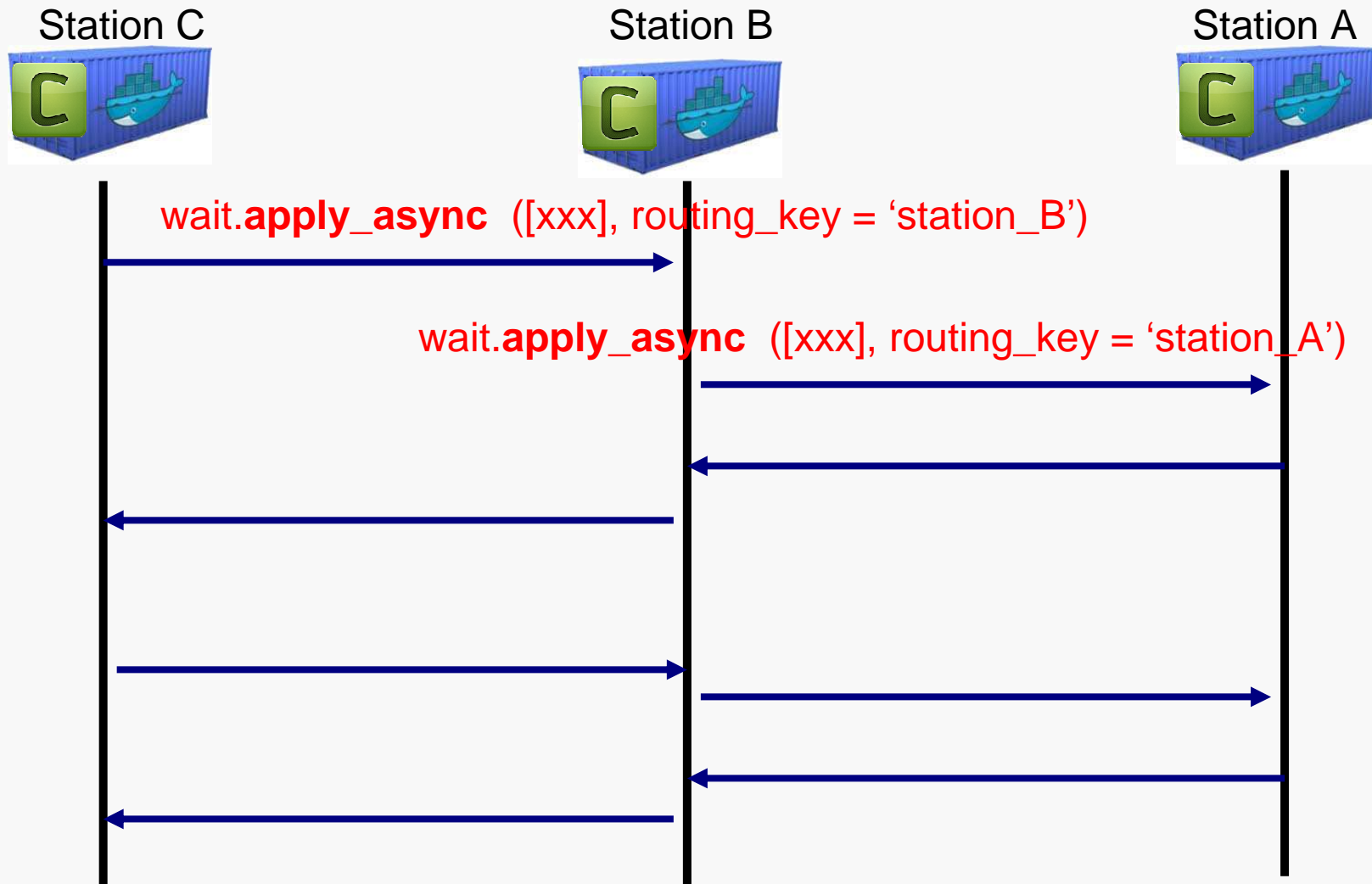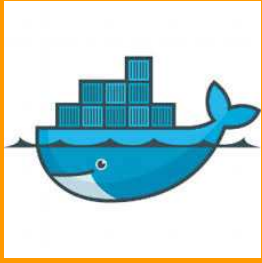
Tokyo   Taipei   New York

doSomething.**apply_async** (routing_key = 'host_NY')

getData.**apply_async(**routing_key = 'host_NY') **.get()**

Distributed System

OOAD

Station C   Station B   Station A

wait.**apply_async** ([xxx], routing_key = 'station_B')

wait.**apply_async** ([xxx], routing_key = 'station_A')

Bound services

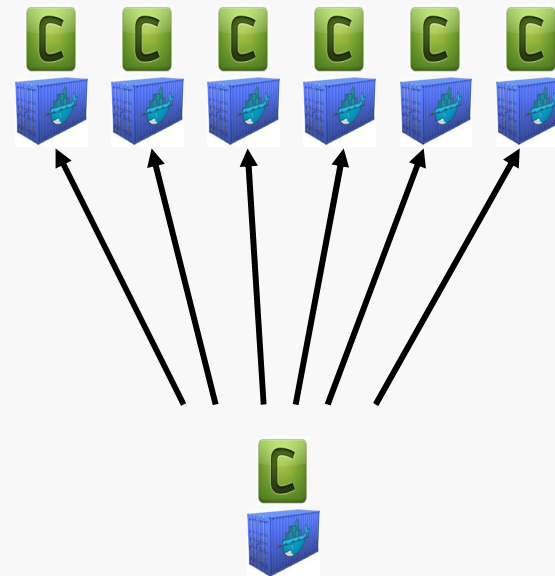# Parallel computing

https://resin.io/blog/what-would-you-do-with-a-120-raspberry-pi-cluster/
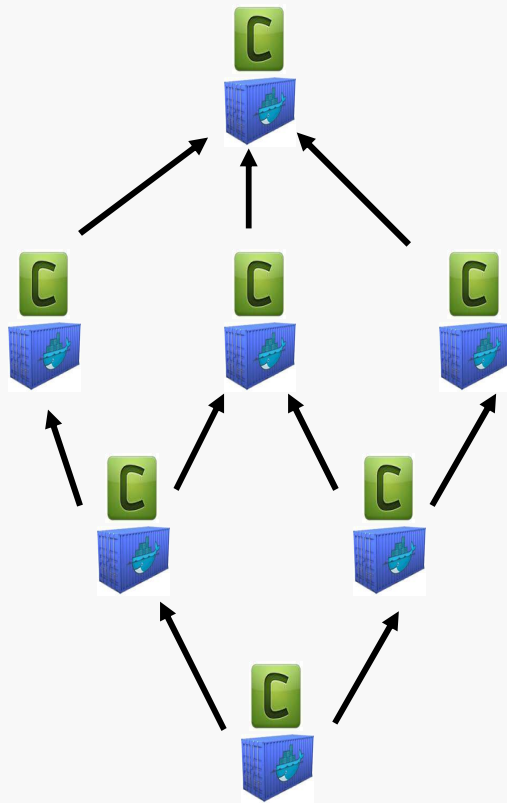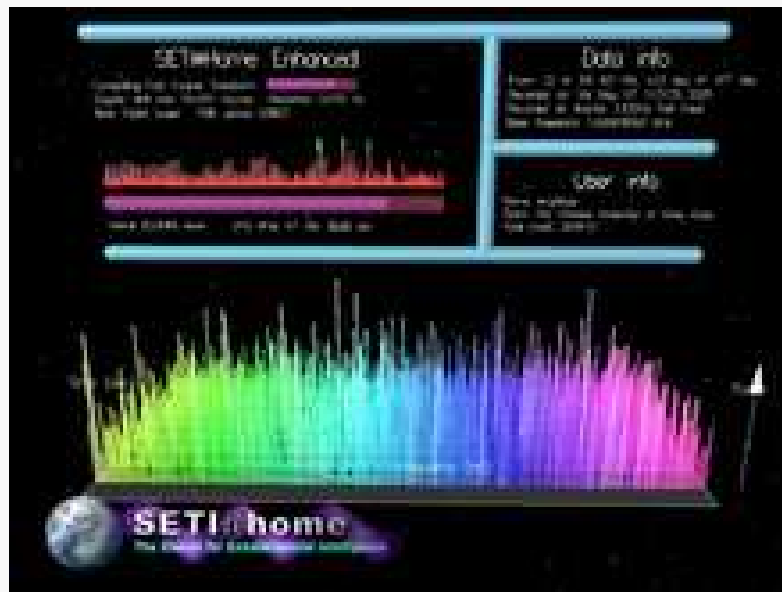
parallel computing ⊆ distributed computing

Deploy parallel computing

- Deploy containers over Docker Swarm
  - Scale up to 480 containers
    - docker-compose up –d
    - docker-compose scale celery=480 (for example)

- Send task messages to worker containers
  results = [doSomething.**apply_async**([data]) **for data in dataset**]

- But, needs root password-less SSH

SETI@home

One for all, All for one

- run a Docker image to join distributed parallel computing cluster, that's all.

  docker run –d wei1234c/musketeer   (for example)

  One for all, All for one

Summary

- Easy to use
  - doSomething.**apply_async** ([xxx], routing_key = 'host_NY')
  - getData.**apply_async**(routing_key = 'host_NY') .**get**()

- Adventages of a distributed system
  - Decoupling, DI/IC (dependency injection / inversion of control)
  - Distributed = Shared
  - Load-Balancing

- Caution:
  - Distributed systems theory for the distributed systems engineer

- Take a look of Canvas in Celery
  - Chain
  - Group

Q & A

References

References :

- **Distributed systems theory for the distributed systems engineer**
- Celery user guide
- Docker document
- MQTT Message Type and Flows
- Celery on Docker Swarm
- IoT as Brain
- One for all, All for one