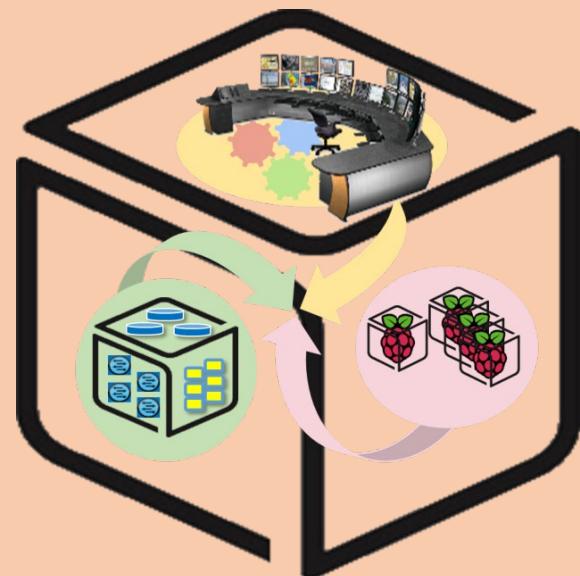


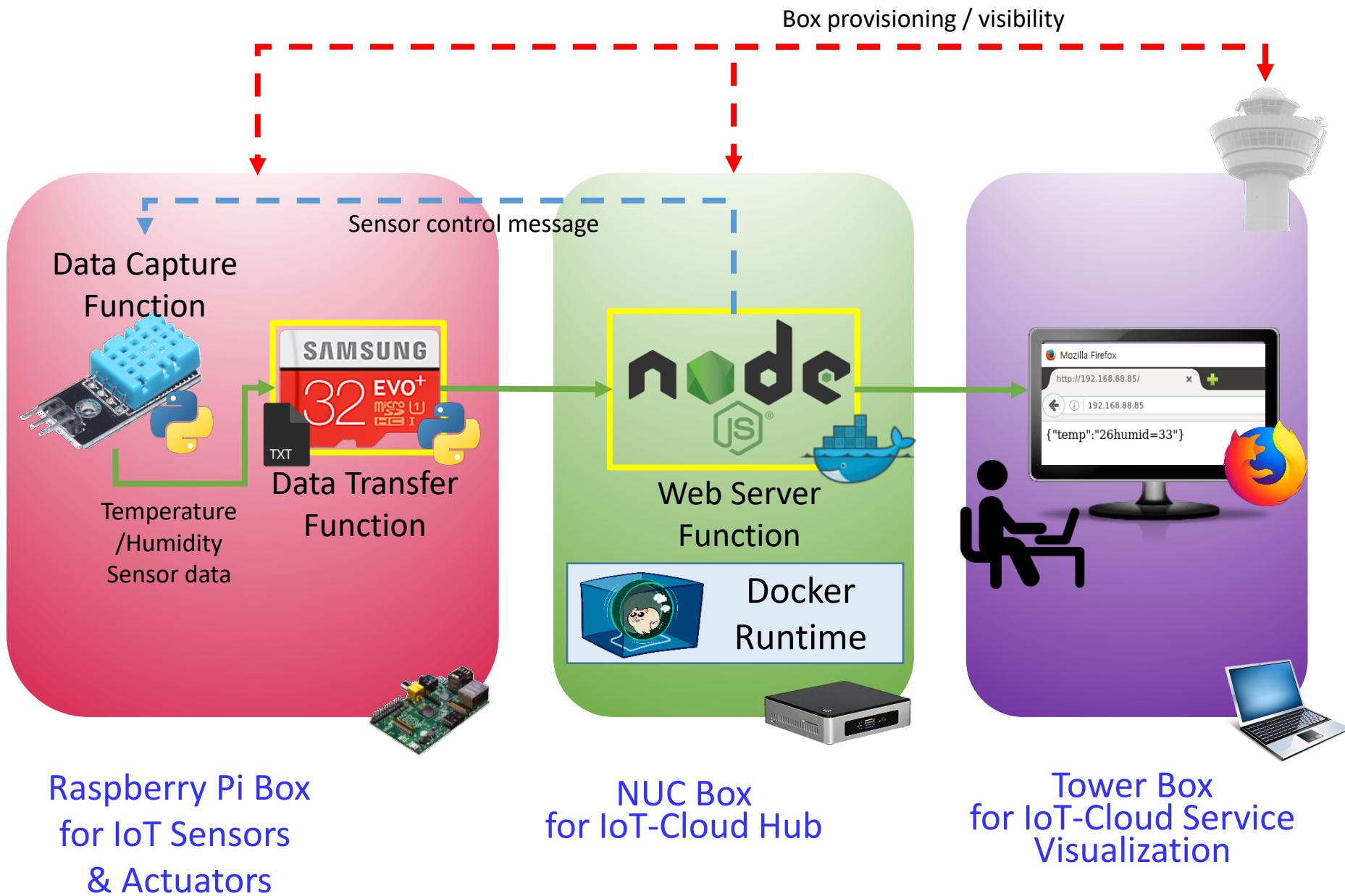
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #4: IoT



<https://github.com/SmartX-Labs/SmartX-Mini-MOOC>

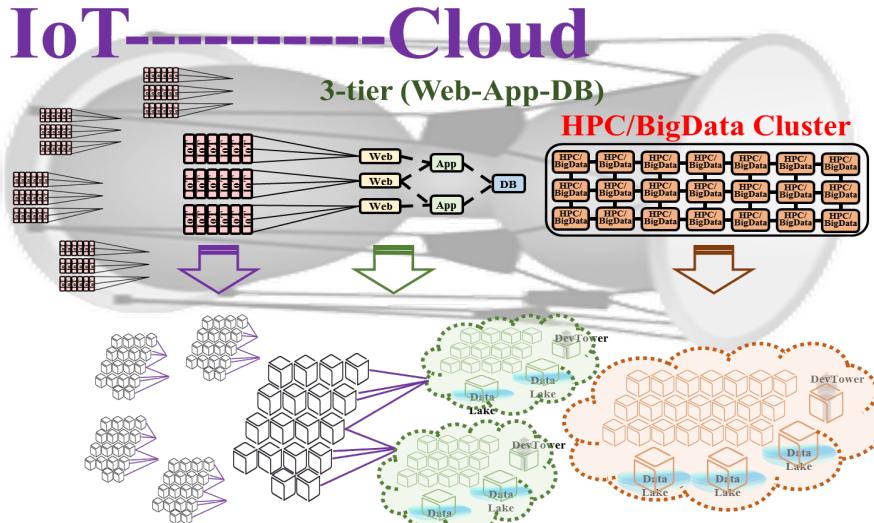
IoT Lab: Concept



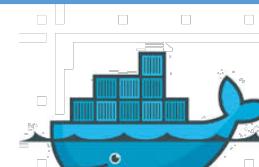
Theory



SmartX IoT-Cloud Services with Microservices-based SaaS Applications



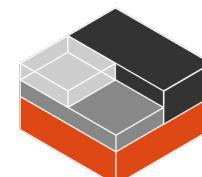
p(Baremetal)
+ c(Container)
+ v(VM)



Process containers



application containers

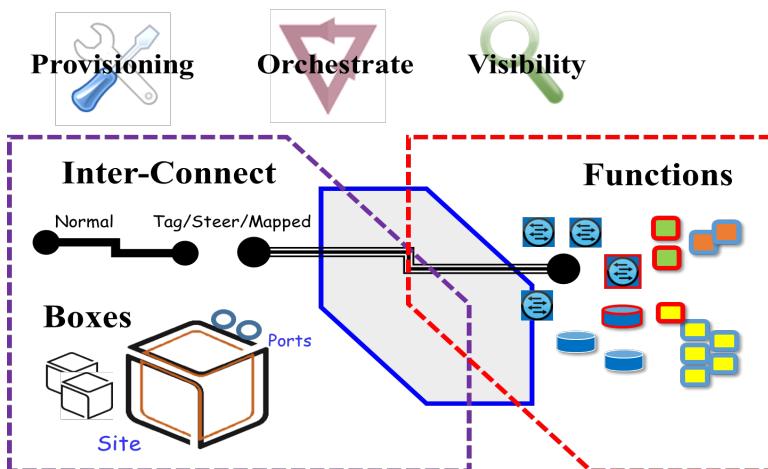


(Monster containers)

Machine containers



VM hypervisors



Open-Source-leveraged Playgrounds

(IoT-SDN/NFV-Cloud (HPC/BigData) Ready)

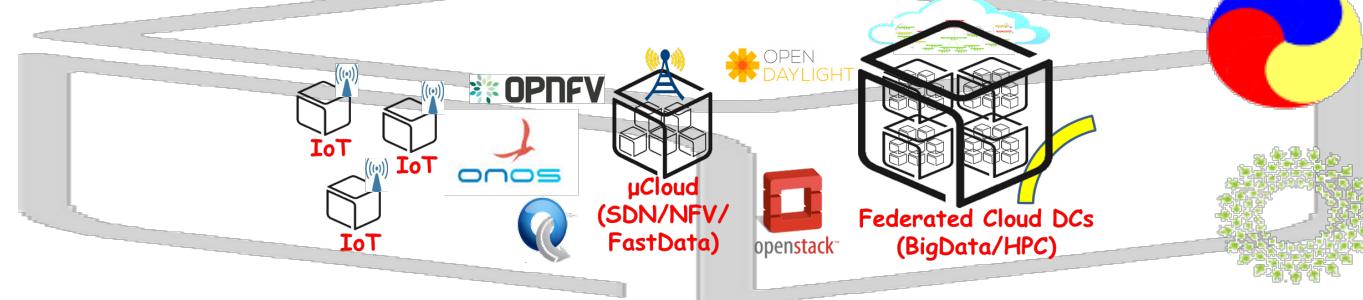
Container-leveraged SmartX IoT-Cloud Services



Open APIs
SmartX Open Platform

SmartX
Playground

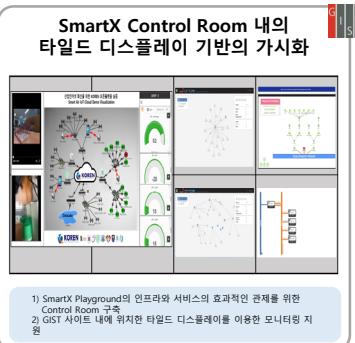
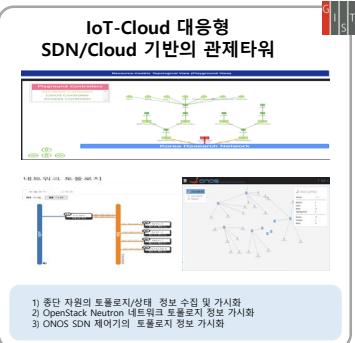
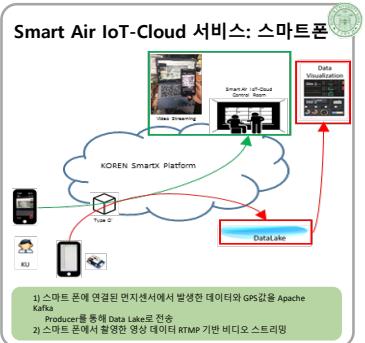
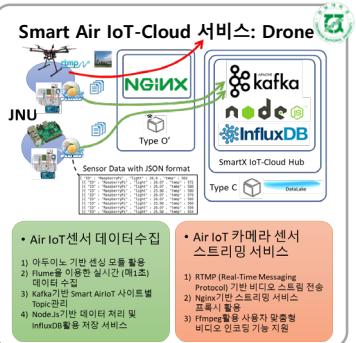
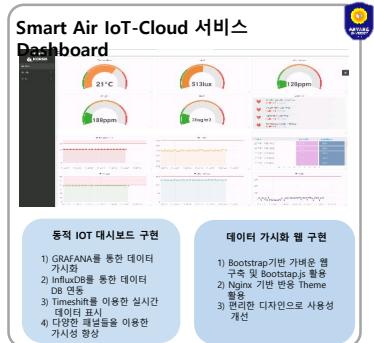
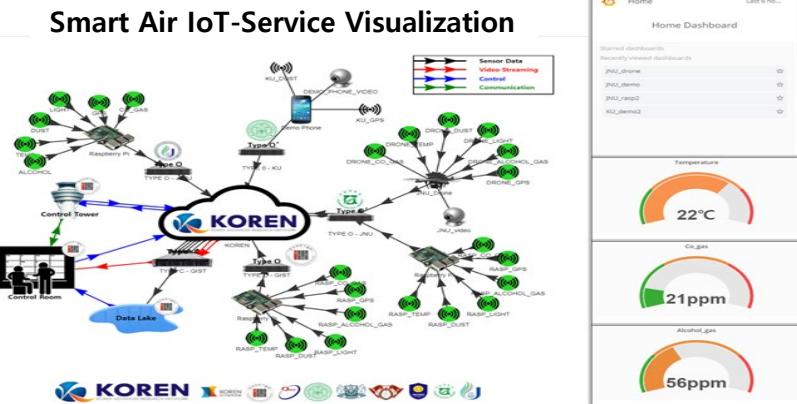
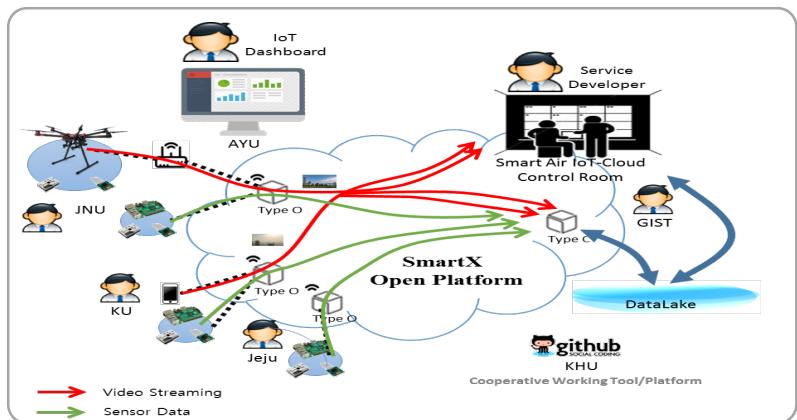
SmartX
Open
Playgrounds



Smart Air IoT-Cloud Services over SmartX Open Platform



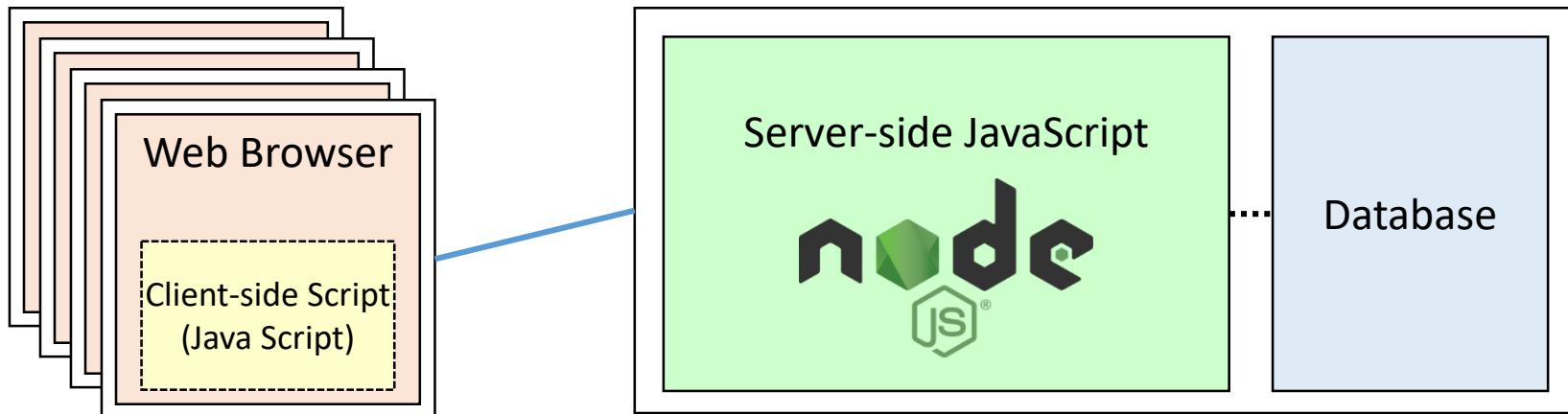
KOREN SmartX Platform을 활용한 선택된 산업인터넷 도메인에 대한 서비스 실증: **컨테이너 기반 Smart Air IoT-Cloud 서비스**



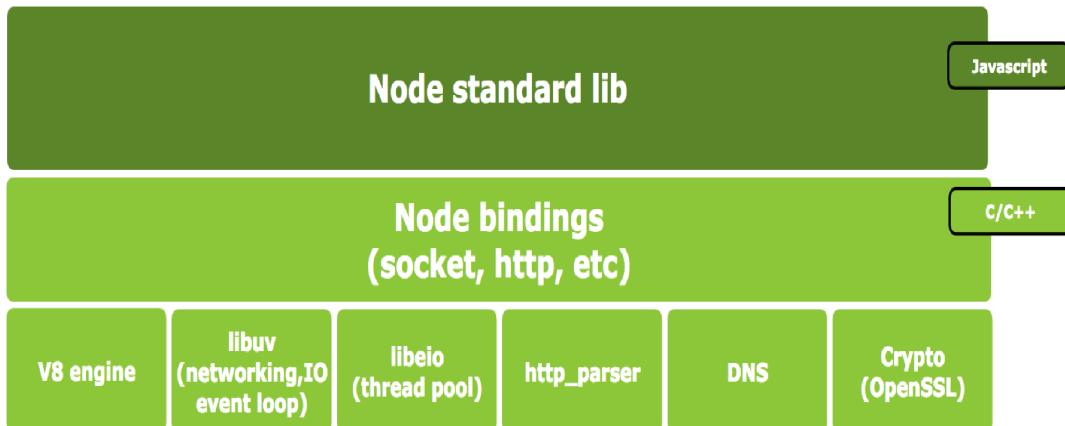
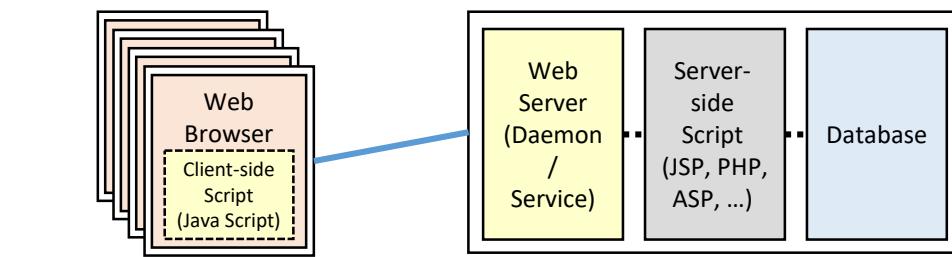
(사)개방형컴퓨터통신연구회
Open Standards and Internet Association



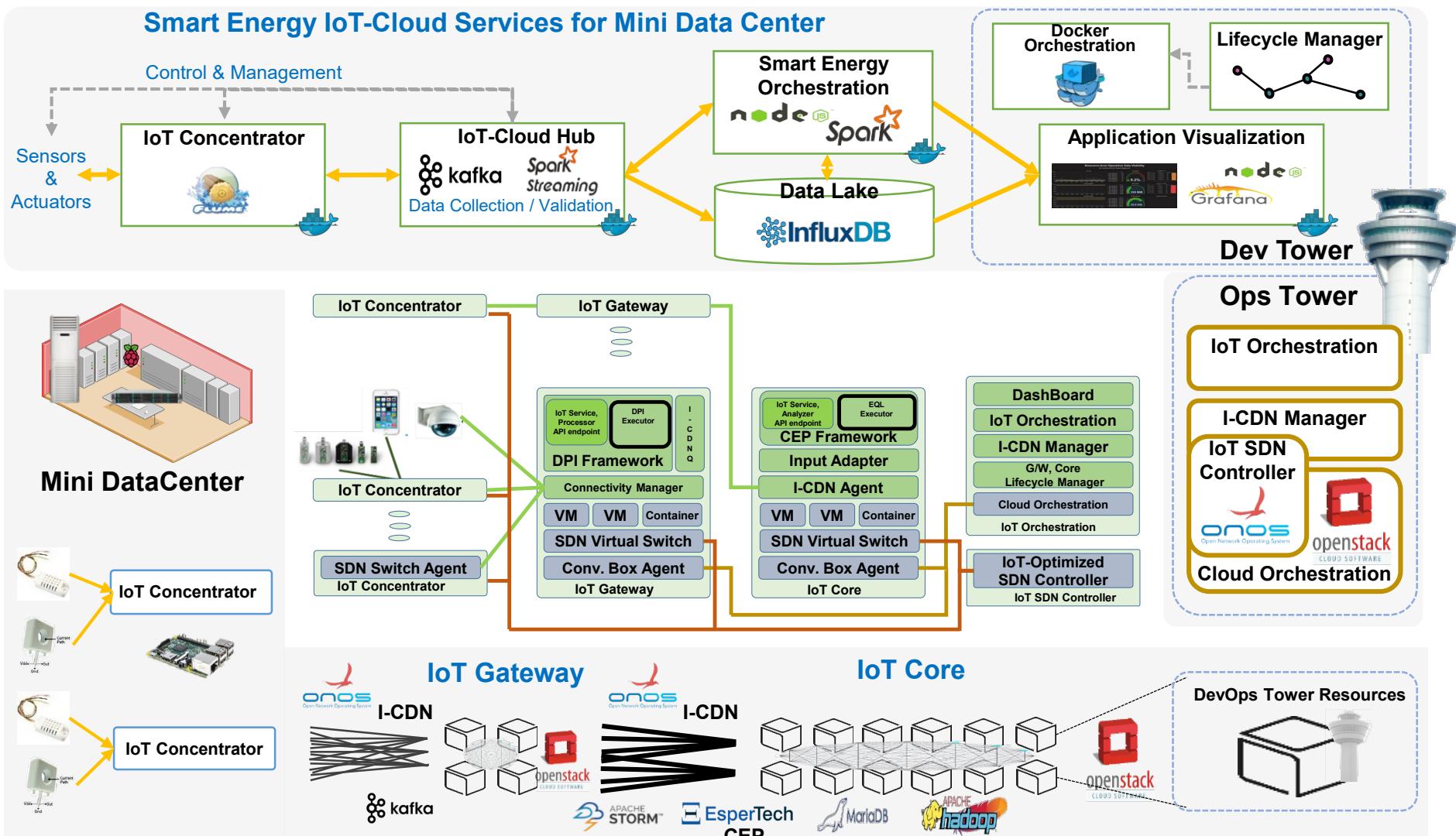
Node.js Web Server Programming



- Node.js Pros: Server-side JavaScript
 - High productivity; Single thread, non-block I/O → light-weight and agile; no direct I/O → no process blocking
- Node.js Cons: Single thread → Complicated multi-process management for multi-core CPU efficiency; Overlapped event callback → callback hell; Memory management overhead due to Garbage Collection



SmartX IoT-Cloud Services for Smart Energy Domain



Practice



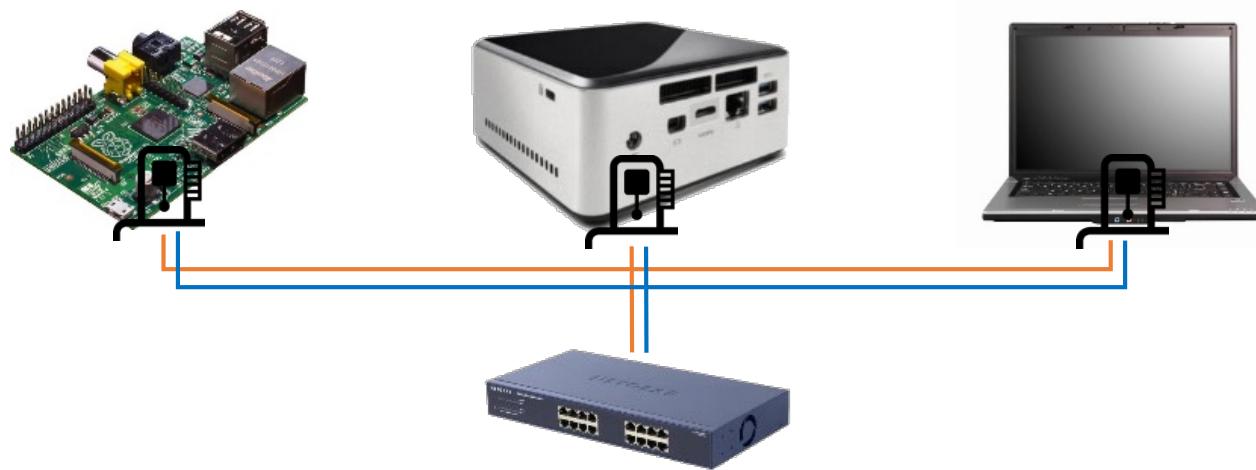
#0 Lab Preparation (1/2)

Wired connection

NAME: Raspberry Pi Model B (Pi)
CPU: ARM Cortex A7 @900MHz
CORE: 4
Memory: 1GB
SD Card: 32GB

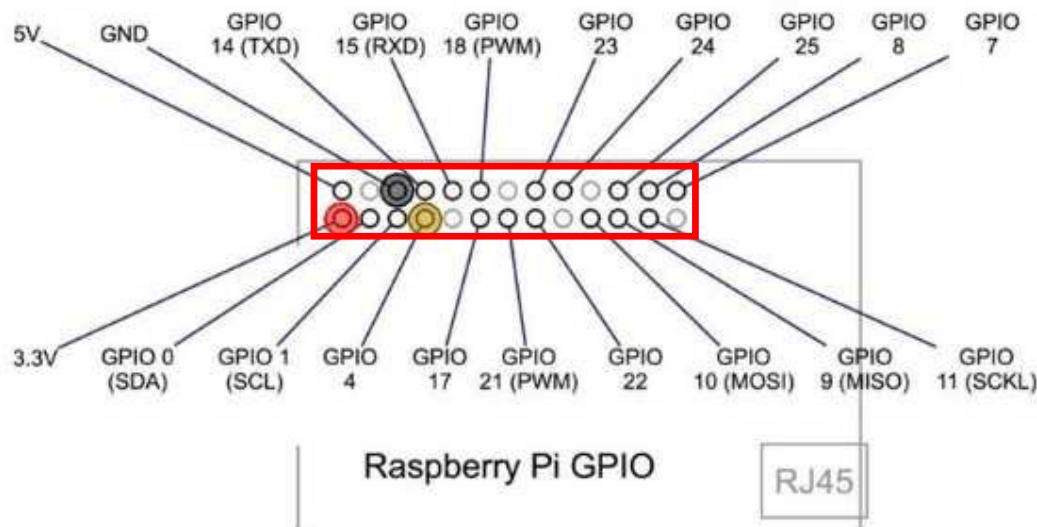
NAME: NUC5i5MYHE (NUC PC)
CPU: i5-5300U @2.30GHz
CORE: 4
Memory: 16GB DDR3
HDD: 94GB

NAME: NT900X3A
CPU: i5-2537U @1.40GHz
CORE: 2
Memory: 4GB DDR3
HDD: 128GB

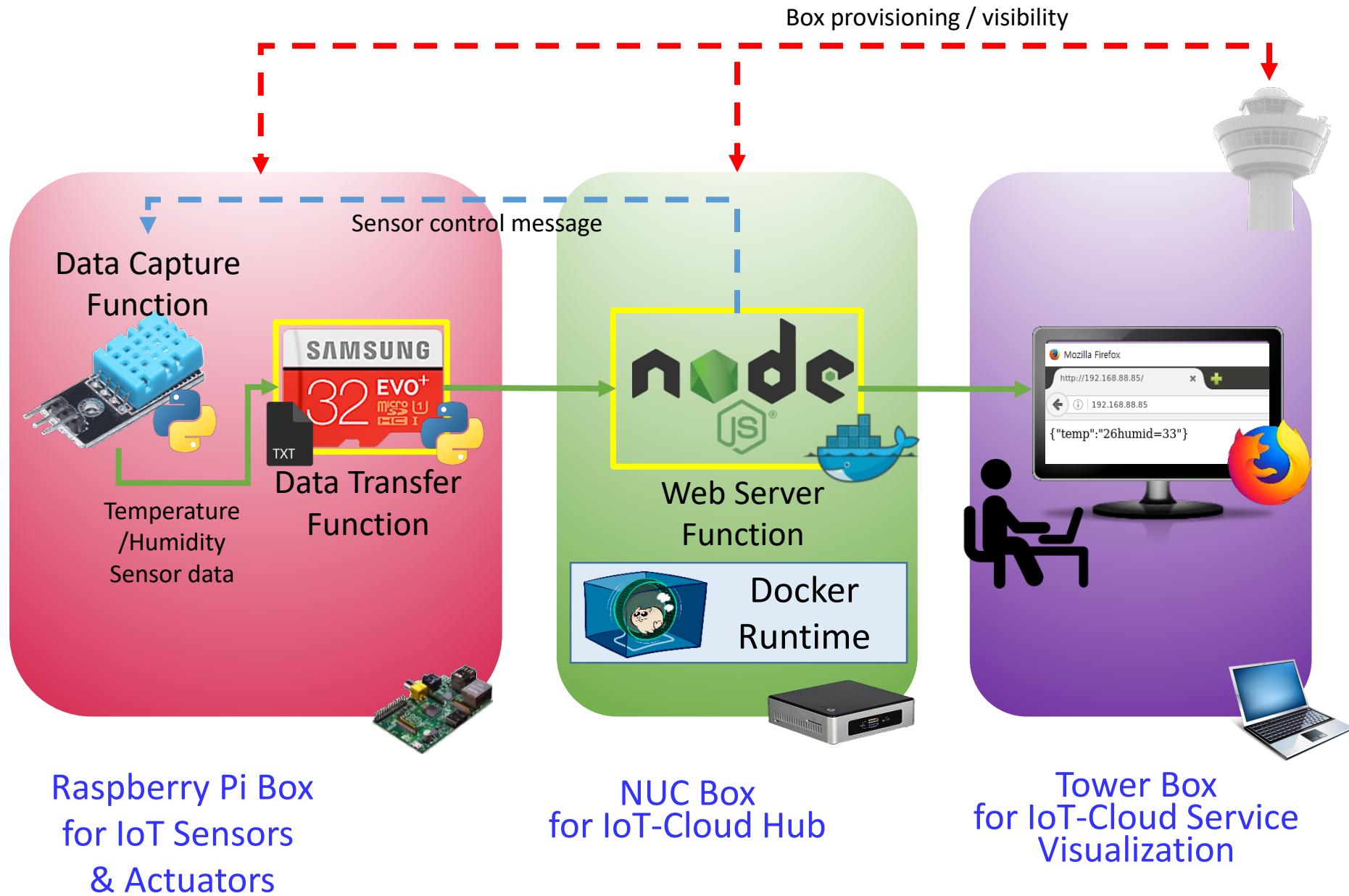


NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

#0 Lab Preparation (2/2)



IoT Lab: Concept





#1-1 Docker Container for Node.js

Web Server: Run Container

- Run a Docker Container

```
$ sudo docker run -it --net=host --name=webserver lshyeung/smartz_webserver
```

- On container

```
$ apt-get update
```

```
$ apt-get install vim
```

```
root@5daf51f2abb0:/# ifconfig
eth0      Link encap:Ethernet HWaddr 02:42:ac:11:00:05
          inet addr:172.17.0.5 Bcast:0.0.0.0 Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe11:5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:4635 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3305 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12293854 (12.2 MB) TX bytes:223481 (223.4 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

#1-2 Docker Container for Node.js

Web Server: Server code



- Open Server code and change NUC IP

```
$ vi /SmartX-mini/IoT-labs/webserver.js
```

```
var net = require('net');
var http = require('http');
var url = require('url');
var fs = require('fs');
var temp;

var server = net.createServer();

server.on('connection', function (socket) {
    socket.write("capture start! \r\n");
    //socket.pipe(socket);
    server.close(function(){
        console.log("tcp server closed.");
        console.log("IoT Temperature, Humidity service web server started");
        server.unref();
    });
});
server.listen(1337, '<NUC IP>');
http.createServer (function (request, response) {
    var query = url.parse(request.url, true).query;
    response.writeHead(200, { 'content-Type': 'text/html'});
    console.log(JSON.stringify(query));
    if (JSON.stringify(query).length > 13)
    {
        fs.writeFile('temp.txt', JSON.stringify(query), 'utf8', function (error){
            console.log('write');
        });
    }
    fs.readFile('temp.txt', 'utf8', function (error, data){
        console.log(data);
        temp = data;
    });
    response.end(temp);
}).listen(80,function (){});
```

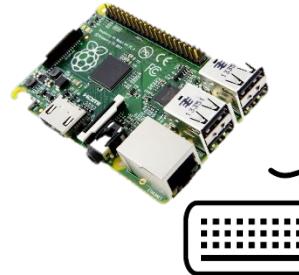


#1–3 Docker Container for Node.js

Web Server: Execute Server

- Execute webapp.js
 - \$ cd SmartX-mini/IoT-labs
 - \$ nodejs webserver.js

- Open browser and go to
http://{IP_of_your_NUC}



#2-1 Temperature / Humidity Sensor test on Raspberry PI: Install package

- Download package from github

```
$ git clone https://github.com/adafruit/Adafruit_python_DHT.git
```

- Install package

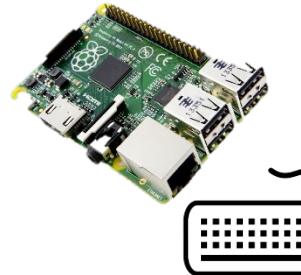
```
$ cd Adafruit_python_DHT
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install python-pip
```

```
$ sudo python -m pip install --upgrade pip setuptools wheel
```

```
$ sudo python setup.py install
```



#2-2 Temperature / Humidity Sensor test on Raspberry PI: Sensor test

- Move to example directory

```
$ cd ~/Adafruit_python_DHT/examples
```

- Modify test code (Change python to python 3)

```
$ sudo vi AdafruitDHT.py
```

```
#!/usr/bin/python  
# Copyright (c) 2014 Adafruit Industries, LLC
```



```
#!/usr/bin/python3  
# Copyright (c) 2014 Adafruit Industries, LLC
```

- Execute test code

```
$ sudo ./AdafruitDHT.py 11 4
```

```
HypriotOS: root@pi03 in ~/Adafruit_Python_DHT/examples on master  
$ ./AdafruitDHT.py 11 4  
Temp=22.0*  Humidity=20.0%
```

#3-1 Sensor Data Capture and Transfer: Sensor Data capture code



- Install dependencies at RPi

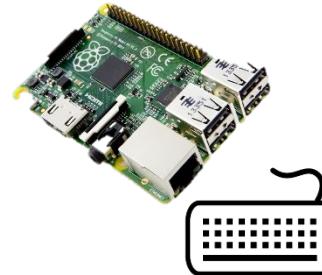
```
$ sudo apt-get update  
$ sudo apt-get install libpython2.7-dev python-numpy  
$ sudo apt-get install mercurial
```

- Open Sensor Data Capture code and Change IP Address

```
$ vi ~/SmartX-mini/IoT-Labs/RPI_capture.py
```

```
import Adafruit_DHT as dht  
import urllib2  
from time import sleep  
import socket  
  
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
sock.connect( '<NUC IP>', 1337)  
s = sock.recv(5441)  
  
sock.close()  
  
if s:  
    h,t = dht.read_retry(dht.DHT11,4)  
    print ('Temp={0:0.1f}*C Humidity={1:0.1f}%).format(t, h)  
    f=open("/root/data.txt", 'w')  
    data =str(t) + " " + str(h)  
    f.write(data)  
    f.close  
    sleep(1)
```

#3-2 Sensor Data Capture and Transfer: Sensor Data transfer code



- Open Sensor Data Capture code and Change IP Address

```
$ vi ~/SmartX-mini/IoT-Labs/RPI_transfer.py
```

```
import Adafruit_DHT as dht
import urllib2
from time import sleep

def transfer():
    f=open("./data.txt",'r')
    line = f.readline()
    words = line.split(" ")
    print(words)
    urllib2.urlopen("http://<NUC IP>?temp="+words[0]+"humid="+words[1]).close
    f.close()

transfer()
```

#4 Execute IoT Web service

Lab #4: IoT 20



- At the Docker container in NUC (Webserver)

```
$ cd ~/SmartX-mini/IoT-labs  
$ nodejs webserver.js
```

- At the Laptop (Tower)

Open Web browser and go to http://<IP_of_NUC>

- At the Rpi (Sensor Data Capture and Transfer)

```
$ cd ~/SmartX-mini/IoT-labs  
$ chmod +x process.sh  
$ sudo ./process.sh
```



Review

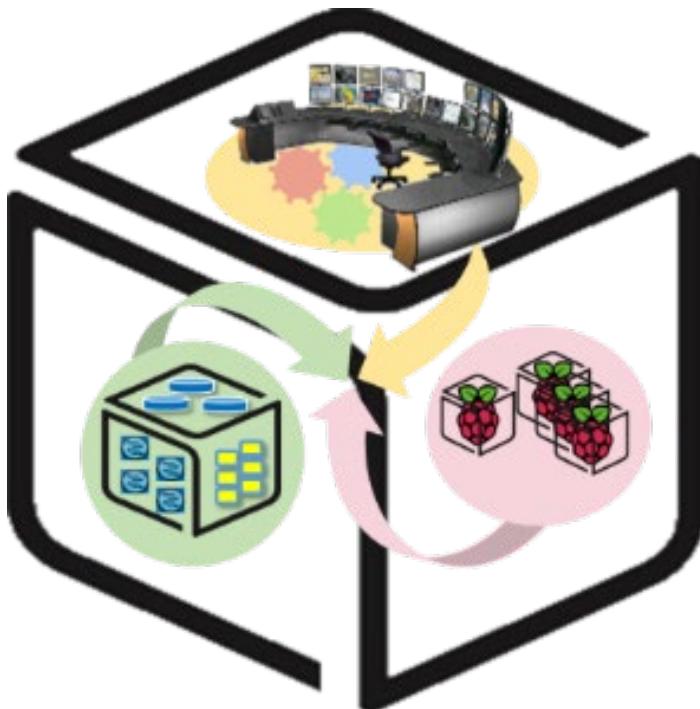


Lab Summary

With IoT Lab, you have experimented

1. An example-based realization of **container-based IoT-Cloud services** that can sense and actuate distributed IoT devices (i.e., Boxes).
2. For the cloud side, the required **Web-app server** realization is supported by utilizing **Node.js programming**.

Thank You for Your Attention Any Questions?



mini@smartx.kr