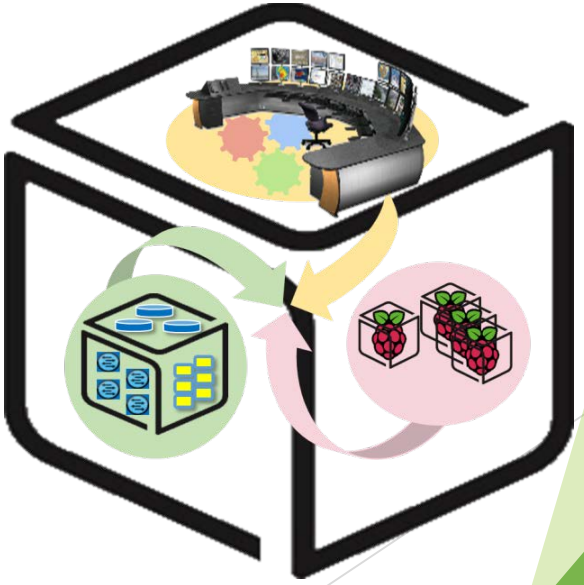
SmartX Labs
for computer systems

WebApp Lab v1.2
(2018, Spring)



NetCS Lab

History and Contributor of Box Lab (2018. 01. 24)

Version	Updated Date	Updated Contents	Contributor
			Lucas
V1.0	2016-05-22		이준기
V1.1	2017-05-10	실습자료 업데이트	권진철
V1.2	2018-01-24	실습자료 업데이트	이승형

Atom Editor

: Free and open-source source code editor

- Cross-Platform

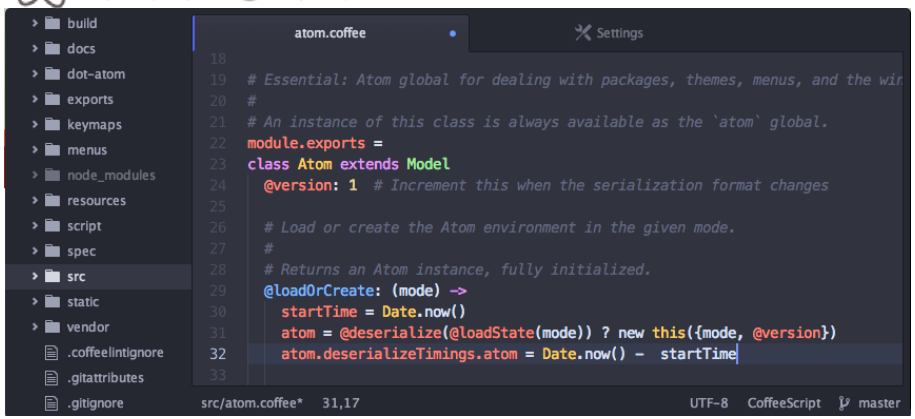
Atom works across operation systems. (OS X, Windows, Linux)

- Built-in package manager

Search for and install new packages which allows you to make your custom dev. environment.

- Smart auto-completion

Helps you write code faster with autocomplete.



Atom Editor
: Installation and Execution

- How to Install Atom

1. Download installer file (deb)
https://github.com/atom/atom/releases/download/v1.16.0/atom-amd64.deb

2. Execute the installer

- How to use

1. On CLI

    sudo atom {directory}

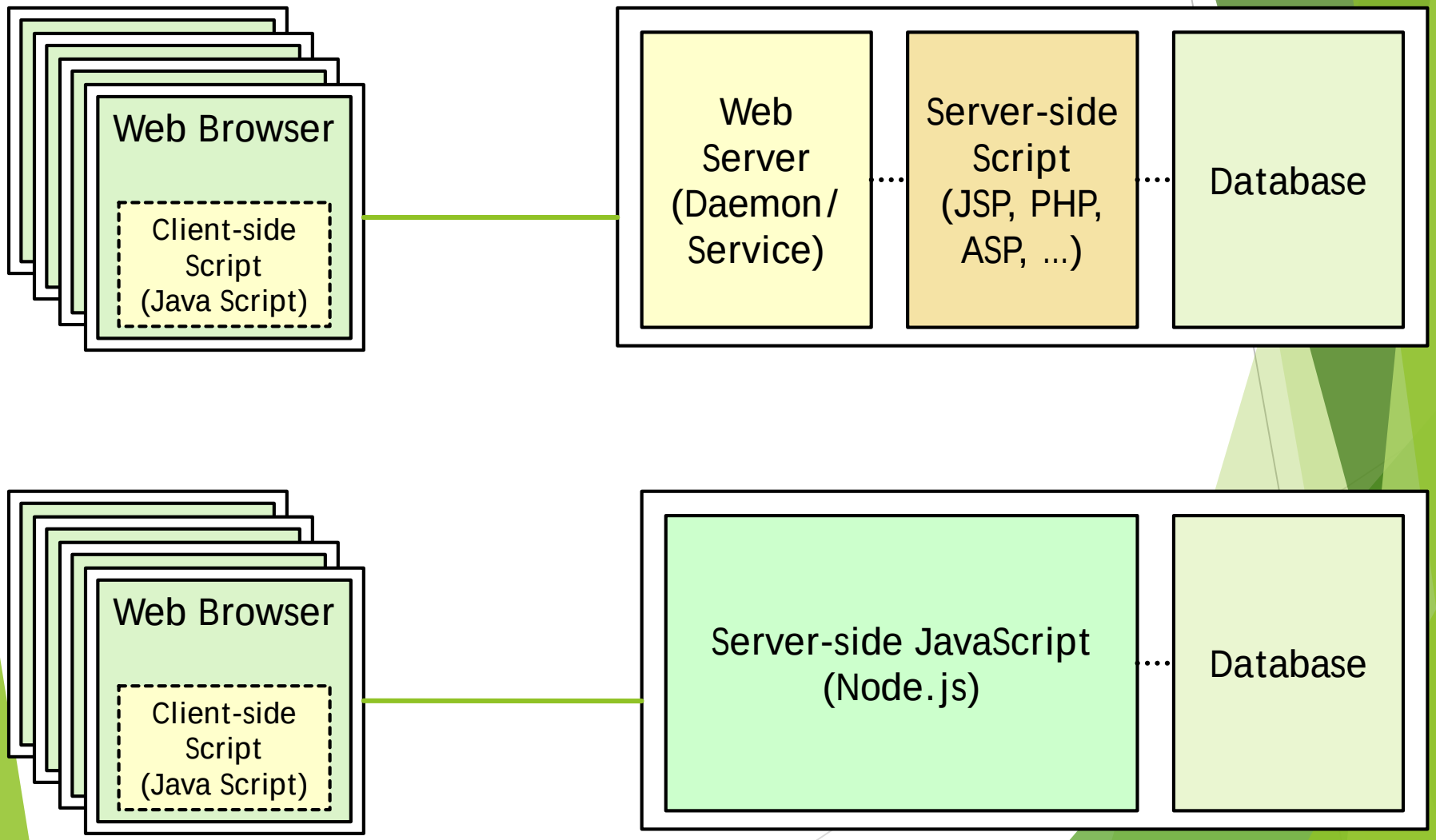
    Ex) sudo atom .

2. On GUI

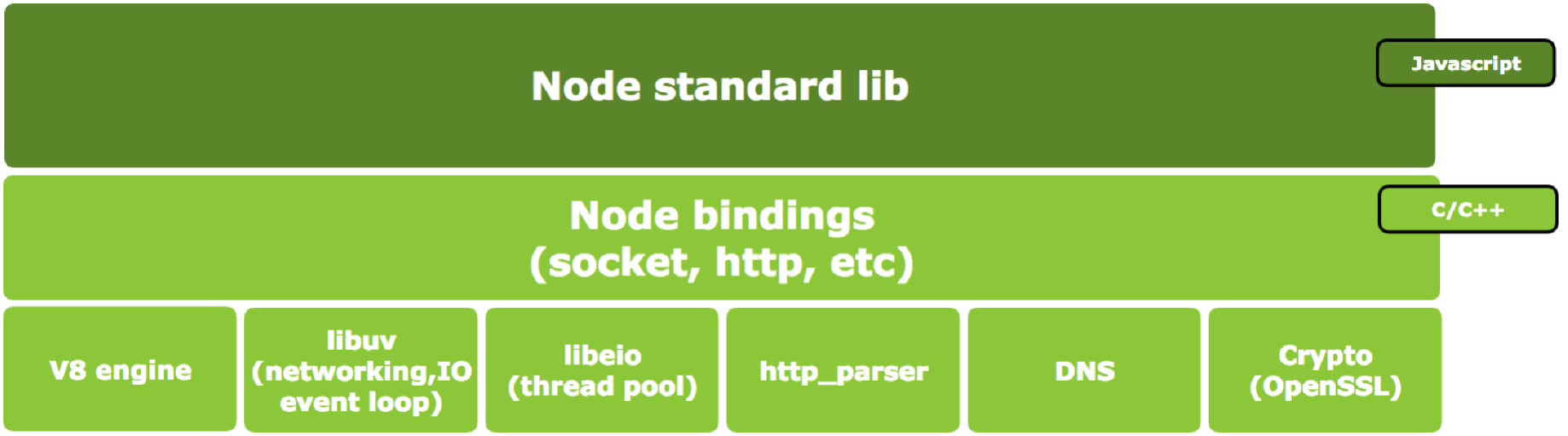
    Double Click Atom icon

웹 서버 프로그래밍

- Node.js 이전 (e.g LAMP)



Node.js Architecture



Node.js Pros and Cons

- Pros
  - Server-side JavaScript → 높은 생산성
  - Single thread, non-block I/O → 가볍고 빠름
  - I/O 직접수행 안 함 → 프로세스 Block되지 않음
- Cons
  - Single thread → 멀티코어 CPU 효율을 위해 여러 개의 프로세스를 사용해야 하고, 어떤 한 작업이 무거우면 전체 성능이 저하될 수 있음
  - Event Callback → 중첩될 경우 소스코드의 가독성 급격히 저하 (callback hell)
  - V8 engine → Garbage Collection기반 메모리관리로 순간적인 CPU 사용률 상승 가능성 있고, 이는 서버 안정성 저하

개발환경 구축

- Install Node.js on NUC (Ubuntu 16.04)

  $ sudo apt-get update

  $ sudo apt-get install nodejs npm

OS: Hypriot 1.9.0

<Web Client>

HTTP Request

HTTP Response

OS: Ubuntu 16.04
Web-Server: Node.js

<Web Server>

Test Your environment

- Make a test code

$ vim hello.js

```
console.log('Hello World');
```

- How to run?

$ nodejs hello.js

```
lucas@Mesos-Ctrl:~/nodejs/hello$ nodejs hello.js
Hello World
lucas@Mesos-Ctrl:~/nodejs/hello$
```

Example2: Simple TCP Server

```
$ vim TCP.js

var net = require('net');

var server = net.createServer(function (socket) {
  socket.write('Echo server\r\n');
  socket.pipe(socket);
});

server.listen(1337);
```

```
lucas@Mesos-Ctrl:~$ telnet 127.0.0.1 1337
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Echo server
```

```
root@Mesos-Ctrl:/home/lucas# curl telnet://127.0.0.1:1337
Echo server
^C
root@Mesos-Ctrl:/home/lucas# curl 127.0.0.1:1337
Echo server
GET / HTTP/1.1
User-Agent: curl/7.35.0
Host: 127.0.0.1:1337
Accept: */*
```

<Client>

Example3: Simple Web Server

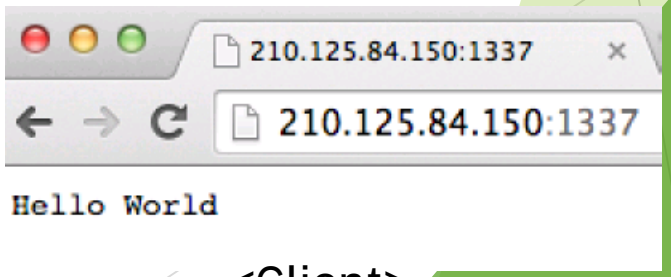
```
$ vim Web.js

var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337);
console.log('Server running at port 1337/');
```

```
lucas@Mesos-Ctrl:~/nodejs/simple$ nodejs simple.js
Server running at port 1337/
```

<Server>

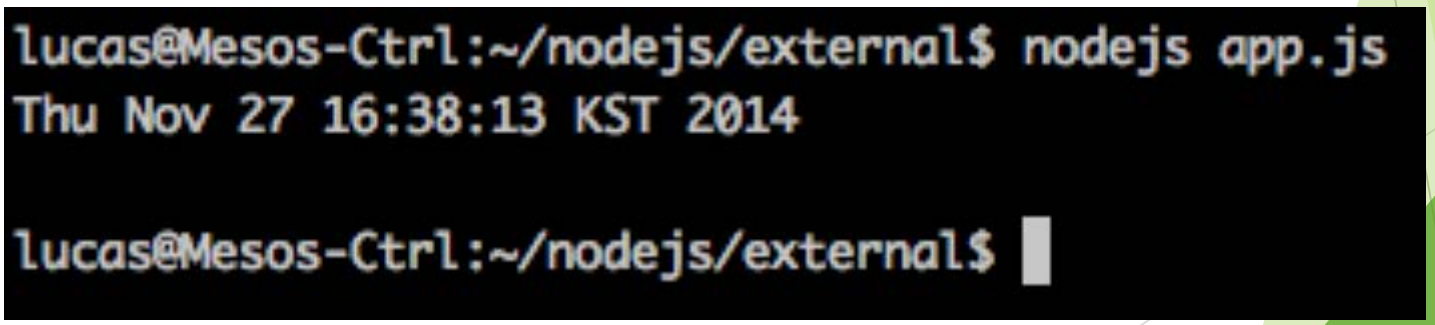
```
lucas@Mesos-Ctrl:~$ curl 127.0.0.1:1337
Hello World
lucas@Mesos-Ctrl:~$
```



<Client>

Example4: External File Execution (Simple)

```
var exec = require('child_process').exec;
exec("date", function (error, stdout, stderr) {
   console.log(stdout);
});
```



Example5: External File Execution (Web)

```
var http = require('http'),
    exec = require('child_process').exec;

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  exec("date", function (error, stdout, stderr) {res.end(stdout)} );
}).listen(1337);
   console.log('Server running');
```

```
lucas@Mesos-Ctrl:~/nodejs$ curl 127.0.0.1:1337
Thu Nov 27 16:34:27 KST 2014
lucas@Mesos-Ctrl:~/nodejs$ curl 127.0.0.1:1337
Thu Nov 27 16:34:32 KST 2014
lucas@Mesos-Ctrl:~/nodejs$ curl 127.0.0.1:1337
Thu Nov 27 16:34:34 KST 2014
lucas@Mesos-Ctrl:~/nodejs$ curl 127.0.0.1:1337
Thu Nov 27 16:34:35 KST 2014
lucas@Mesos-Ctrl:~/nodejs$
```

Node Packaged Module(NPM)

- Node.js로 만들어진 모듈 관리자
  - Ubuntu의 APT와 유사하다
- 패키지 설치
  - npm install <package name>
  - Global install: 패키지를 /usr/local/lib에 설치, 패키지에 따라 Global로 설치해야 하는 것이 있다.
  - Local install: 패키지를 현재 폴더 내에 설치하고, 폴더내의 파일에서만 불러올 수 있다.
- 패키지 검색
  - https://www.npmjs.com/ 에서 패키지의 정보를 검색할 수 있다.

File Upload (1/4)

- Install Packages: npm install formidable fs-extra

```
var formidable = require('formidable'),
    http = require('http'),
    util = require('util'),
    fs   = require('fs-extra');

http.createServer(function(req, res) {
  /* Process the form uploads */
  if (req.url == '/upload' && req.method.toLowerCase() == 'post') {
    var form = new formidable.IncomingForm();
    form.parse(req, function(err, fields, files) {
      res.writeHead(200, {'content-type': 'text/plain'});
      res.write('received upload:\n\n');
      res.end(util.inspect({fields: fields, files: files}));
    });
```

File Upload (2/4)

```
form.on('progress', function(bytesReceived, bytesExpected) {
        var percent_complete = (bytesReceived / bytesExpected) * 100;
        console.log(percent_complete.toFixed(2));
    });

    form.on('error', function(err) {
        console.error(err);
    });

    form.on('end', function(fields, files) {
        /* Temporary location of our uploaded file */
        var temp_path = this.openedFiles[0].path;
        /* The file name of the uploaded file */
        var file_name = this.openedFiles[0].name;
        /* Location where we want to copy the uploaded file */
        var new_location = process.env.PWD + '/';
```

File Upload (3/4)

```
fs.copy(temp_path, new_location + file_name, function(err) {
          if (err) {
              console.error(err);
          } else {
              console.log("success!")
          }
      });
  });

  return;
}

 /* Display the file upload form. */
 res.writeHead(200, {'content-type': 'text/html'});
 res.end(
```

File Upload (4/4)

```
'<form action="/upload" enctype="multipart/form-data" method="post">'+
  '<input type="text" name="title"><br>'+
  '<input type="file" name="upload" multiple="multiple"><br>'+
  '<input type="submit" value="Upload">'+
  '</form>'
 );

}).listen(1337);
```

```
lucas@Mesos-Ctrl:~/nodejs/fupload$ nodejs upload.js
1.02
2.03
3.05
4.06
5.08
6.10
7.11
8.13
9.14
10.16
11.18
12.19
96.52
97.54
98.56
99.57
100.00
success!
^Clucas@Mesos-Ctrl:~/nodejs/fupload$ ls
kernel.img  node_modules  upload.js
lucas@Mesos-Ctrl:~/nodejs/fupload$
```

```
lucas@Mesos-Ctrl:~$ curl -F upload=@kernel.img http://127.0.0.1:1337/upload
received upload:

{ fields: {},
  files:
   { upload:
      { domain: null,
        _events: {},
        _maxListeners: 10,
        size: 6449964,
        path: '/tmp/8374da21ed85f5d42f45325d112f65fd',
        name: 'kernel.img',
        type: 'application/octet-stream',
        hash: null,
        lastModifiedDate: Thu Nov 27 2014 21:05:37 GMT+0900 (KST),
        _writeStream: [Object] } } }lucas@Mesos-Ctrl:~$
lucas@Mesos-Ctrl:~$
```

Run a HTTP Web Server using node.js Express Module

- Express is a node.js module for web application
- Make a project folder

  $ mkdir webserver
- Create a package.json file in the project directory

  $ cd webserver

  $npm init -> Enter name, version, main and so on
- Install express module and add dependency

  $ npm install express --save
- Check installed package list

  $ npm ls or npm –g ls

Run a HTTP Web Server using node.js Express Module

- Write a source code

  $sudo vim app.js

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})
app.get('/login', function (req, res) {
  res.send('Please Login')
})

app.listen(3000, function () {
  console.log('Example web server listening on port 3000!')
})
```

Run a HTTP Web Server using node.js Express Module

- Run web server

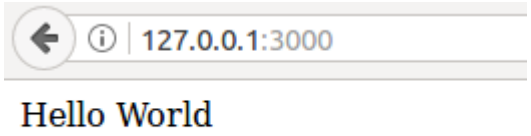
  $sudo nodejs app.js

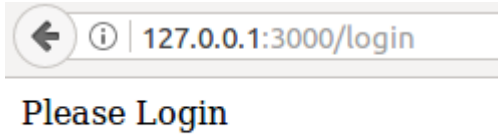
```
jckwon@jckwon-VirtualBox:~/http$ nodejs app.js
Example web server listening on port 3000!
```

- Connect to your web server

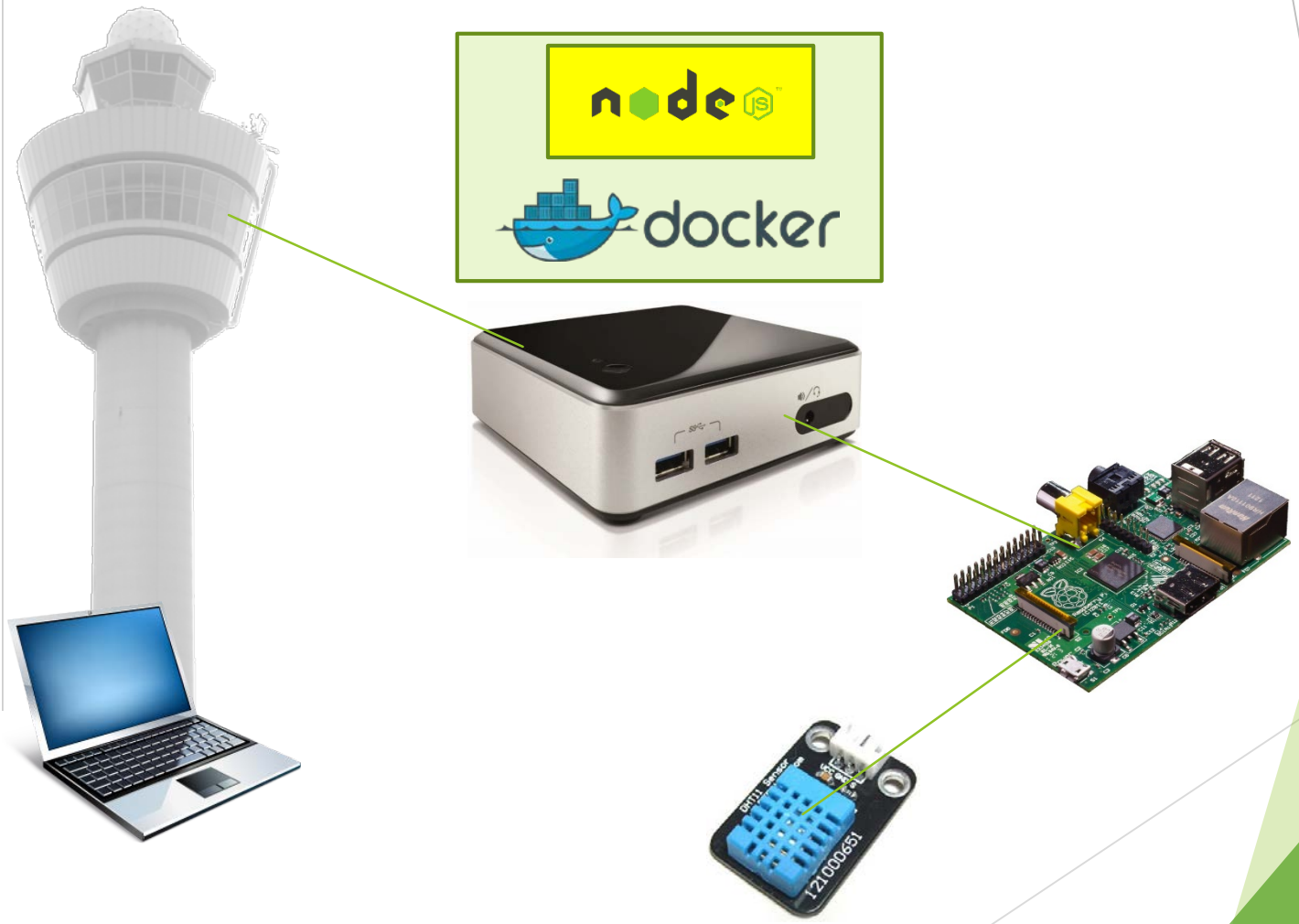
  http://{your_NUC_IP}:3000

  http://{your_NUC_IP}:3000/login





Architecture of WebApp



Create Docker Container for nodejs WebApp

- Run a Docker Container

  $ sudo docker run –it –-net=host --name=webapp ubuntu /bin/bash

- On container

  $ apt-get update

  $ apt-get install net-tools

  $ apt-get install iputils-ping

```
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)
```

Install node.js in Docker Container

- Install nodejs in Docker Container

  $ apt-get install nodejs

  $ (apt-get install nano)

- Test

  $ nano test.js (or vim nano.js)

```
 GNU nano 2.5.3

console.log('Hello World');
```

  $ nodejs test.js

```
root@netcsnuc:/# nodejs test.js
Hello World
```

Write Server Application code

- Write webapp.js code

  $ sudo nano webapp.js (or sudo vim webapp.js)

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

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 (){
    console.log('Server running...');
  });
```

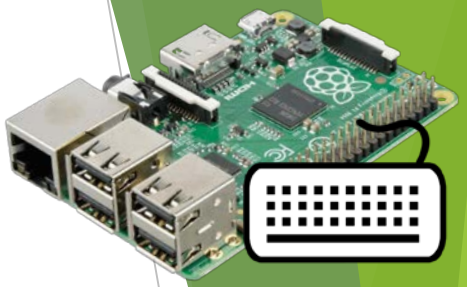
Execute Server Application

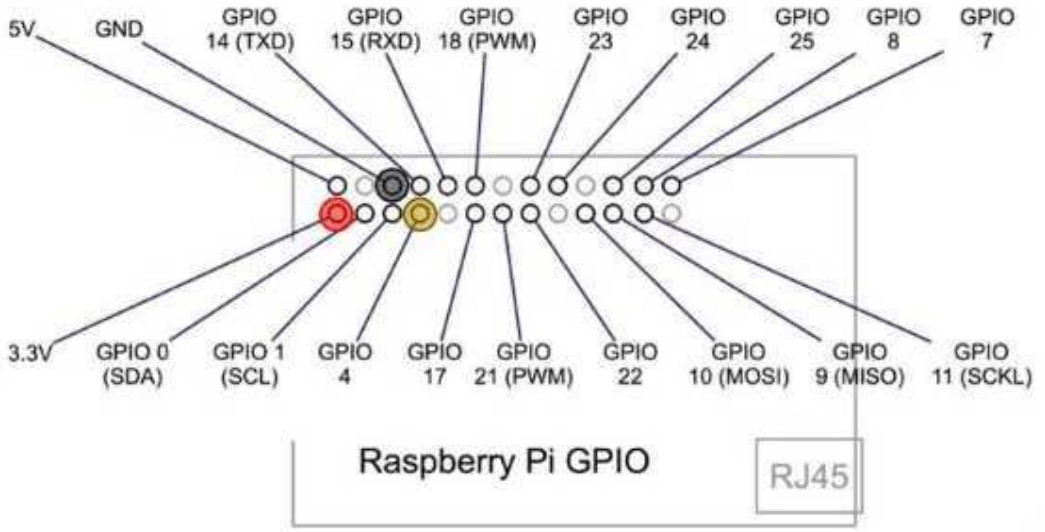
- Execute WebApp.js

  - $ nodejs webapp.js

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

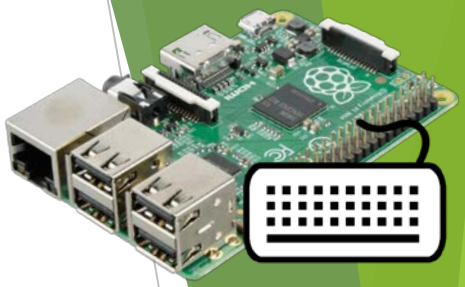
WebApp with RPi sensor





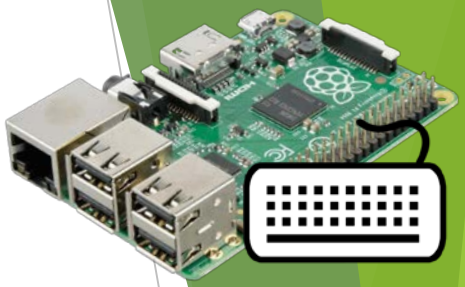
http://www.uugear.com/portfolio/dht11-humidity-temperature-sensor-module/

WebApp with RPi sensor



- Install dependencies at RPi
  - $ sudo apt-get update
  - $ sudo apt-get install libpython2.7-dev python-numpy
  - $ sudo apt-get install RPi.GPIO
  - $ sudo apt-get install mercurial

WebApp with RPi sensor



- Open Client Application source code

  Copy source code from

  https://github.com/2jungi/SmartX-Mini/blob/master/RPI_temp.py

  $ sudo nano RPI_temp.py

Revise code for WebApp

- Change IP address to <Your NUC's IP>

RPI_temp.py

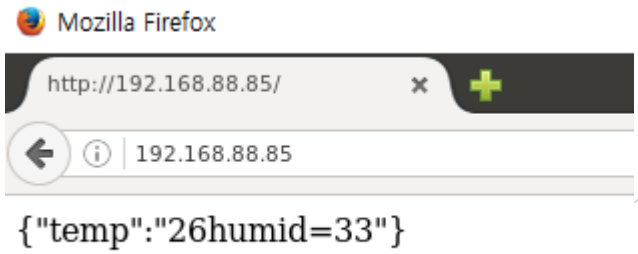
```
104  if int(Humidity) + int(Temperature) - int(bin2dec(crc)) == 0:
105          print "Humidity:"+ Humidity +"%"
106          print "Temperature:"+ Temperature +"C"
107          urllib2.urlopen("http://192.168.88.85?temp="+Temperature+"humid="+Humidity).close
```

- sudo python RPI_temp.py

```
HypriotOS: root@pi17 in ~
$ sudo python sample.py
Humidity:30%
Temperature:32C
```

WebApp in Browser

- At the Docker container in NUC
  - $ nodejs webapp.js
- At the NUC
  - Open Web browser and go to http://<IP_of_NUC>
- At the RPi
  - $ sudo python RPI_temp.py



References

- http://nodejs.org/
- http://expressjs.com
- http://pyrasis.com/nodejs/nodejs-HOWTO
- http://www.codediesel.com/nodejs/processing-file-uploads-in-node-js/
- http://www.gliffy.com/publish/2752090/
- http://kipid.tistory.com/entry/Learning-Nodejs
- http://www.nodeclipse.org/ubuntu/linux/java/nodejs/2015/2015/07/09/Starting-with-Java-and-Node.js-development-on-Ubuntu-Linux.html