

# OSIF 개발자 가이드

## 1. OSIF 응용 S/W 개발 환경 준비

### 1.1. Python 3 설치

Python 3는 OSIF 개발자 지원 도구의 설치 및 활용을 위해 사용되며, Docker-compose의 설치를 위해서도 사용됨.

- MAC OS

```
$sudo apt-get install python3
```

- Windows

<https://www.python.org/downloads/> 에서 설치파일 다운로드

### 1.2. Docker 설치

Docker는 OSIF 모듈의 실행 환경으로 동작함.

- MAC OS

```
$ curl -fsSL get.docker.com -o get-docker.sh
$ chmod 777 get-docker.sh
$ ./get-docker.sh
$ sudo usermod -aG docker pi
```

- Windows

공식 홈페이지 참고 : <https://docs.docker.com/docker-for-windows/install/>

한글 설명 블로그 참고 : <https://steemit.com/kr/@mystarlight/docker>

### 1.3. OSIF 전용 Docker registry 접속 허용 (임시)

현재 OSIF 전용 Docker registry는 사실 Docker registry를 이용하고 있으며, HTTPS가 적용되지 않은 상태이기 때문에 다음과 같은 설정을 추가해야 함

- MAC OS

```
/etc/docker/daemon.json
```

```
C:\ProgramData\docker\config\daemon.json
```

```
{  
  "insecure-registries": ["osif.iotocean.org:5000"]  
}
```

## 1.4. Docker compose 설치

OSIF 모듈의 dockerizing을 간편하게 수행할 수 있도록 하기 위한 툴. Docker 설치 시 기본 포함 도구임. 추가로 설치 필요 시 다음 참고.

- 공식 설치 가이드 : <https://docs.docker.com/compose/install/>
- pip로 설치하는 경우

```
$ sudo pip install docker-compose
```


## 2. OSIF 응용 S/W 개발

### 2.1. OSIF Service Description

#### 2.1.1. OSIF Service Description 생성기

브라우저를 이용하여 OSIF Service Description 생성기 접속

<http://osif.iotocean.org/#!/tools-metadata>



DASHBOARD

OPEN SERVICES

ABOUT OSIF

1 USER GUIDES

2 DEVELOPERS GUIDE

3 DEVELOPERS'S TOOLS

OSIF Portal Service

Service information

2

Service name \*

Service ID

RENEW

Developer's name \*

Major \*

Minor \*

Revision \*

Service description

Open service endpoint

3

OSIF Data Grid endpoint \*

Global OSIF Data Grid port \*

Local OSIF Data Grid port \*

4

Local service endpoint

+

Global service endpoint

+

Select list item first

5

GENERATE SERVICE DESCRIPTION

- (1) “DEVELOPERS TOOLS” 메뉴 선택
- (2) 서비스에 대한 일반적인 설명 정보 입력
  - (a) Service UUID는 타 서비스에 의한 데이터 조회에 사용될 수 있음
- (3) Databus 접속 정보
  - (a) 기본값 사용 권고
- (4) Open data 편집
  - (a) 개발하고자 하는 서비스가 다른 서비스들에게 공유(공개)하고자 하는 정보를 정의
  - (b) Service UUID와 open data name의 조합으로 서비스의 데이터 조회 가능
- (5) 입력한 정보로부터 생성되는 Service Description 정보 미리보기 및 복사 기능 제공

## Service Description Sample

```
{
  "service": {
    "serviceName": "LED Monitor",
    "serviceId": "3fb8800e-edd5-473c-97fa-ce461196aaf3",
    "versionCode": {
      "major": 1,
      "minor": 0,
      "revision": 0
    },
    "creator": "KETI",
    "openData": {
      "local": [
        {
          "name": "status",
          "description": "LED status",
          "template": "ON | OFF"
        }
      ],
      "global": [
        {
          "name": "watt",
          "description": "energy consumption",
          "template": "Number"
        }
      ]
    },
    "serviceDesc": "LED monitoring service on status and energy consumption"
  },
  "databus": {
    "globalHost": "osif.synctechno.com",
    "globalPort": "5701",
    "localPort": "5701"
  }
}
```

## 2.2. Raspberry Pi에서 Node.js 응용 S/W 개발

git을 이용하여 샘플 소스코드 다운로드

```
$ git clone https://github.com/loTKETI/osif-demo-rpi-led.git
```

### 2.2.1. OSIF 적용 전

```

var GPIO = require('onoff').Gpio;

/**
 * GPIO 사용을 위한 상수 정의
 */
var ledRPinBCM = 18; // Phy 12, wPi 1, BCM 18
var ledGPinBCM = 17; // Phy 11, wPi 0, BCM 17
var ledBPinBCM = 27; // Phy 13, wPi 2, BCM 27
var buttonPinBCM = 22; // Phy 15, wPi 3, BCM 22

/**
 * GPIO 사용을 위한 GPIO pin 설정
 */
var ledR = new GPIO(ledRPinBCM, 'out');
var ledG = new GPIO(ledGPinBCM, 'out');
var ledB = new GPIO(ledBPinBCM, 'out');
var button = new GPIO(buttonPinBCM, 'in', 'both');

var LED_OFF = 0;

/**
 * 상수 값에 따라 GPIO 핀을 제어하여 LED 제어
 *
 * 0: 모두 끄기
 * 1: RED
 * 2: GREEN
 * 3: BLUE
 */
function controlLED(ledState) {

  console.log( "controlLED: ", ledState)

  switch(ledState) {
    case 0:
      ledR.writeSync(1);
      ledG.writeSync(1);
      ledB.writeSync(1);
      break;

    case 1:
      ledR.writeSync(0);
      ledG.writeSync(1);
      ledB.writeSync(1);
      break;

    case 2:
      ledR.writeSync(1);
      ledG.writeSync(0);
      ledB.writeSync(1);
      break;

    case 3:
      ledR.writeSync(1);
      ledG.writeSync(1);
      ledB.writeSync(0);
      break;
  }
}

var g_SensorControlState = 0;

```

```

/**
 * Button 이 눌렸을 때 동작을 구현하기 위한 event handler
 */
function onButtonPushed(err, state) {
  if(state === 0) {
    // 버튼이 눌렸을 때 상태 변경 처리코드 호출
    g_SensorControlState ++;
    if(g_SensorControlState > 3)
      g_SensorControlState = 1;

    // LED 제어
    controlLED(g_SensorControlState); }
  else {
    return;
  }
}

/**
 * 어플리케이션 초기화 코드
 */
function serviceStart() {
  // LED를 끈다.
  controlLED(LED_OFF);

  // button watch 시작
  button.watch(onButtonPushed);

  console.log("Service start...");
}

/**
 * 어플리케이션 종료 코드
 */
function serviceShutdown() {
  // LED 끈다.
  controlLED(LED_OFF);

  // Process 종료
  process.exit(0);
}

/**
 * 어플리케이션 종료를 위한 signal 처리
 * Ctrl+C로 종료
 */
process.on('SIGINT', function () {
  serviceShutdown();
});
process.on('SIGTERM', function () {
  serviceShutdown();
});

/**
 * 어플리케이션 시작
 */
serviceStart();

```

## 2.2.2. OSIF 적용 후

```
var GPIO = require('onoff').Gpio;

/**
 * [OSIF] OSIF client library import
 */
var OSIFClient = require('osif-client').Client;

/**
 * GPIO 사용을 위한 상수 정의
 */
var ledRPinBCM = 18; // Phy 12, wPi 1, BCM 18
var ledGPinBCM = 17; // Phy 11, wPi 0, BCM 17
var ledBPinBCM = 27; // Phy 13, wPi 2, BCM 27
var buttonPinBCM = 22; // Phy 15, wPi 3, BCM 22

/**
 * GPIO 사용을 위한 GPIO pin 설정
 */
var ledR = new GPIO(ledRPinBCM, 'out');
var ledG = new GPIO(ledGPinBCM, 'out');
var ledB = new GPIO(ledBPinBCM, 'out');
var button = new GPIO(buttonPinBCM, 'in', 'both');

var LED_OFF = 0;

/**
 * [OSIF] OSIF client 초기화
 */
var serviceOptions = require('./osif.json');
var client1 = new OSIFClient(serviceOptions);

/**
 * 상수 값에 따라 GPIO 핀을 제어하여 LED 제어
 *
 * 0: 모두 끄기
 * 1: RED
 * 2: GREEN
 * 3: BLUE
 */
function controlLED(ledState) {

  console.log( "controlLED: ", ledState)

  switch(ledState) {
    case 0:
      ledR.writeSync(1);
      ledG.writeSync(1);
      ledB.writeSync(1);
      break;

    case 1:
      ledR.writeSync(0);
      ledG.writeSync(1);
      ledB.writeSync(1);
      break;

    case 2:
      ledR.writeSync(1);
```



```

    ledG.writeSync(0);
    ledB.writeSync(1);
    break;

    case 3:
        ledR.writeSync(1);
        ledG.writeSync(1);
        ledB.writeSync(0);
        break;
    }
}

var g_SensorControlState = 0;

/**
 * Button이 눌렸을 때 동작을 구현하기 위한 event handler
 */
function onButtonPushed(err, state) {
    if(state === 0) {
        // 버튼이 눌렸을 때 상태 변경 처리코드 호출
        g_SensorControlState++;
        if(g_SensorControlState > 3)
            g_SensorControlState = 1;

        // [OSIF] 로컬 bus에 값을 변경
        client1.setLocalAppData("iotweek-sensor-control", g_SensorControlState);
    }
    else {
        console.log('button state is not 0', state);
        return;
    }
}

/**
 * 어플리케이션 초기화 코드
 */
function serviceStart() {
    controlLED(LED_OFF);

    // button watch 시작
    button.watch(onButtonPushed);

    console.log('Service start...');

    /**
     * [OSIF] OSIF 초기화 코드 실행
     * - 서비스 시작 코드 실행
     * - event listener 등록
     */
    client1.init()
        .then(function(client) {

            // [OSIF] 어플리케이션 초기화 시 "startService()" 호출
            return client.startService();

        })

        .then(function(value) {

            // [OSIF] 관심있는 값을 bus에서 읽어옴
            return client1.getLocalAppData("iotweek-sensor-control");
        })

```

```

.then(function(value){
    // 초기값으로 LED 제어
    controlLED(value);

    // [OSIF] Bus data 변경 이벤트 핸들러
    var listener = function (key) {
        console.log('LOCAL OPENDATA UPDATED : value : ', key);

        // [OSIF] 변경된 값을 넘겨주지는 않기 때문에 변경된 key 값을 이용해서 새로운 변경값을 읽어와야 함
        client1.getLocalOpendata(key)
            .then((value) => {
                console.log('LISTENER : getLocalAppData : ', value);

                // [OSIF] 변경된 값으로 LED 제어
                controlLED(value);
            })
    };

    // [OSIF] 이벤트 핸들러 등록
    client1.subscribeToLocalOpendata('iotweek-sensor-control', listener);
})

}

/**
 * 어플리케이션 종료 코드
 */
function serviceShutdown() {
    controlLED(LED_OFF);

    // [OSIF] 어플리케이션 종료 시 "stopService()" 호출
    client1.stopService();

    process.exit(0);
}

/**
 * 어플리케이션 종료를 위한 signal 처리
 * Ctrl+C로 종료
 */
process.on('SIGINT', function () {
    serviceShutdown();
});
process.on('SIGTERM', function () {
    serviceShutdown();
});

/**
 * 어플리케이션 시작
 */
serviceStart();

```

### 2.2.3. OSIF Client library 설치

소스코드 수정에 앞서 osif-client 라이브러리를 npm 으로부터 설치

```
$ npm install osif-client --save
```

#### 2.2.4. OSIF 응용 S/W 등록

pip를 이용하여 osiftool을 설치

```
$ sudo pip3 install osiftool
```

소스코드 위치에서 osiftool 실행 후 ID와 password 입력

```
$ cd {{application source code path}}
$ osiftool
Load OSIF Service description: ./osif.json
  Service name: iotweek-demo-led
    Version: 1.0.5
    Description: IoT Week 2018 Demo service

OSIF Portal login:iot01@keti.re.kr
Password: ****
$
```

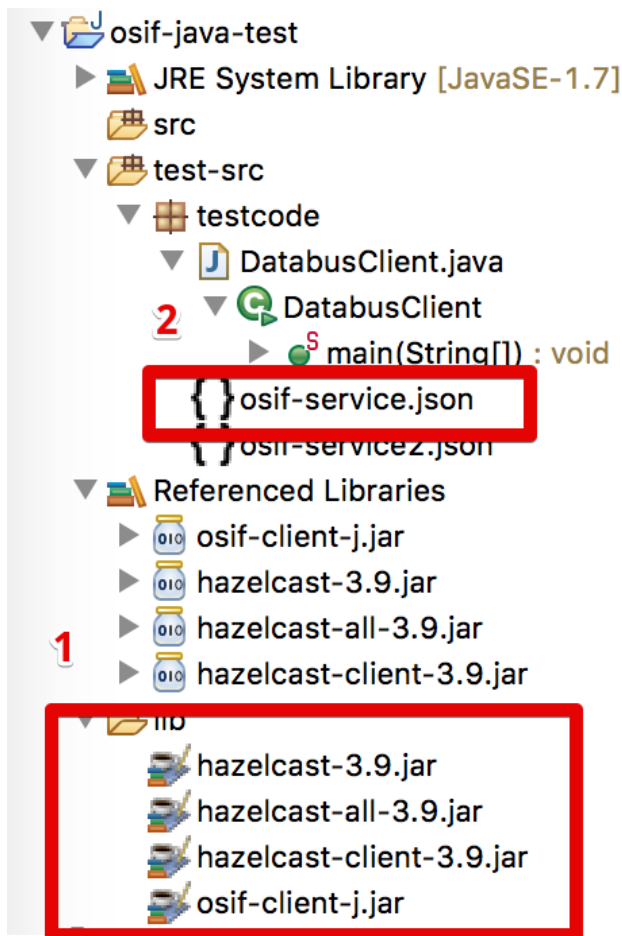
등록 여부 확인

<http://osif.iotocean.org>

#### 2.3. Java 응용 S/W 개발

Sample 코드 다운로드 링크

<https://www.dropbox.com/sh/t9qx2lse3av5qsr/AADouFdNDCewV0NoQG5uGOba?dl=0>



- (1) 필수 라이브러리
- (2) Service Description 파일

```

public static void main(String[] args) {

// 1) Service Description 파일 경로를 파라미터로 하여 Client 객체 생성

    Client c = Client.newClient("./test-src/testcode/osif-service.json");
    Client c2 = Client.newClient("./test-src/testcode/osif-service2.json");

    try {

// 2) 응용프로그램 실행 시작 시점에 startService() 함수 호출
        c.startService();
        c2.startService();

// 3) 다른서비스에서 제공하는 데이터를 조회/모니터링 하기 위해서는 해당 서비스의 Service ID를 조회해야 함
        String serviceId = "3fb8800e-edd5-473c-97fa-ce461196aaf3";

// 4) 다른 서비스의 데이터 모니터링을 위한 listener 객체
        IOpendataListener listener = new IOpendataListener() {

// 4-1) 데이터가 추가되었을 경우 호출된다.

```

```

        public void entryAdded(String key, String value) {
            System.out.println("ADDED: " + key + " : " + value);
        }

// 4-1) 데이터가 업데이트 되었을 경우 호출된다.

        public void entryUpdated(String key, String value) {
            System.out.println("UPDATED: " + key + " : " + value);
        }

};

// 4) 생성한 listener 객체를 client에 등록. Service ID와 모니터링 하고자 하는 open data의 name을 알아야 한다.
        c2.subscribeToGlobalOpendata(serviceld, "global_data_1", listener);

// 5) 나의 global open data에 데이터를 쓴다. 첫번째 파라미터는 service description에 정의된 global open data중 하나의 이름과 일치하여야 한다.
        c.setGlobalAppData("global_data_1", "new value");

// 6) 서비스 중지

        c.stopApplication();

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

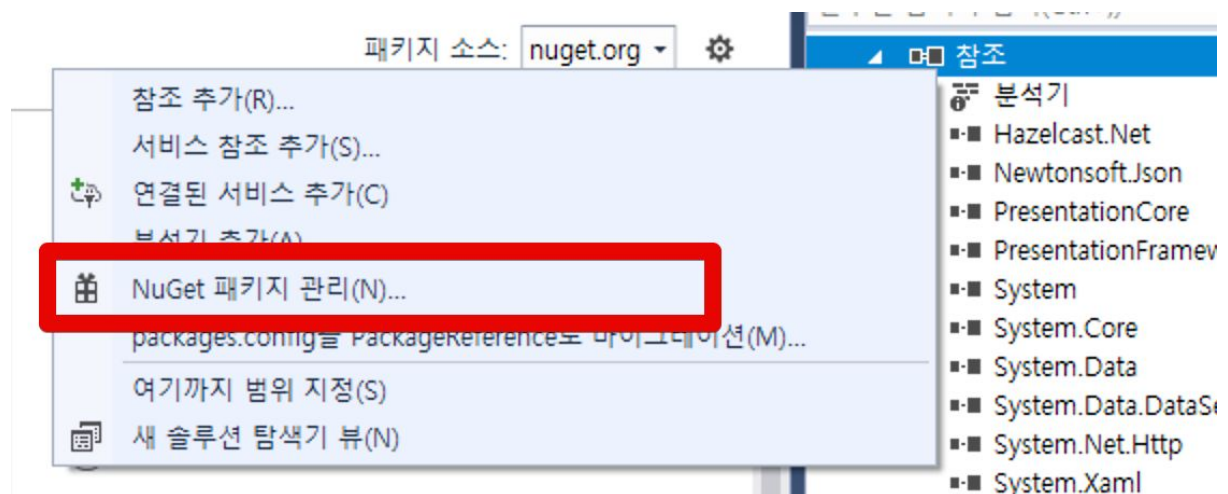
## 2.4. .Net C# 응용 S/W 개발

Sample 코드 다운로드 링크

<https://www.dropbox.com/sh/t9qx2lse3av5qsr/AADouFdNDCewV0NoQG5uGObaa?dl=0>

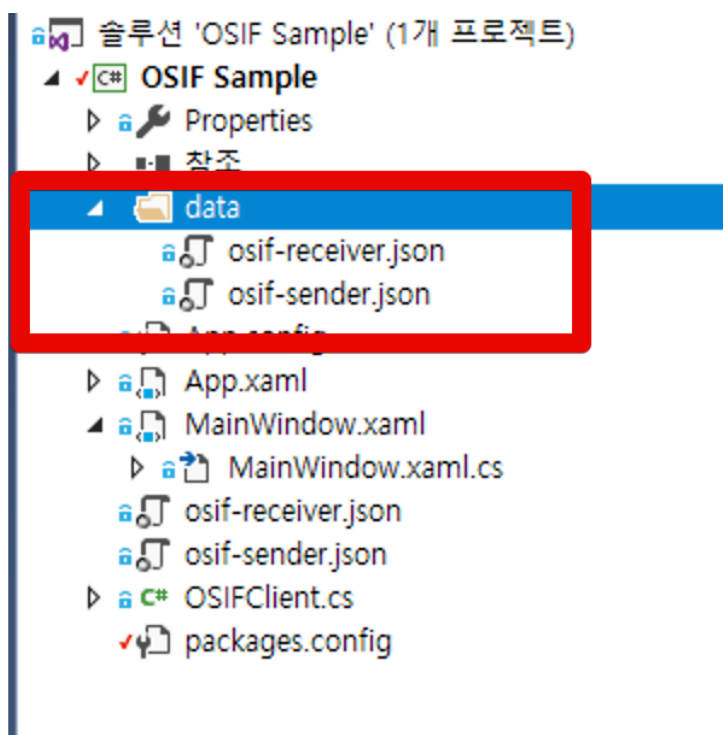
### 2.4.1. 필수 라이브러리 설치

NuGet 패키지 관리자 등을 통하여 “Hazelcast.Net”과 “Newtonsoft.Json” 라이브러리 설치



## 2.4.2. Service Description 파일 생성

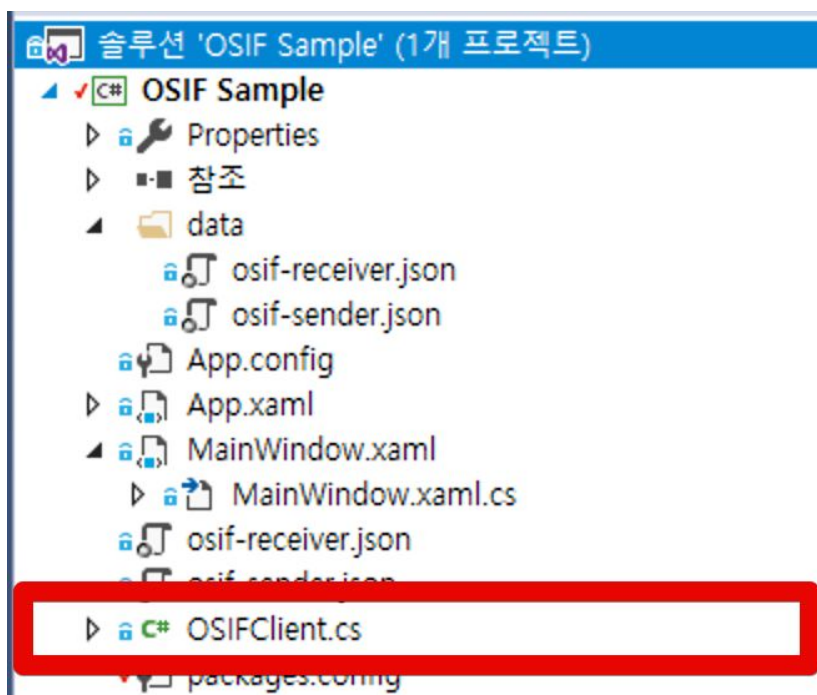
<http://osif.iotocean.org> 페이지에서 Service Description 파일을 생성하여 리소스 파일로 추가



파일 생성 후 리소스파일로 읽어들이기 위해 속성 변경 필요 “포함 리소스”



#### 2.4.3. OSIFClient.cs OSIF 클라이언트 소스파일 추가



#### 2.4.4. 응용 코드 작성

##### 2.4.4.1. 서비스 시작

```
private void btnStartSender_Click(object sender, RoutedEventArgs e)
```

```

    {
        try
        {
            // 1) 리소스로 저장된 service description 파일을 로딩하기 위한 경로
            string resourceName = "OSIF_Sample.data.osif-sender.json";

            // 2) 프로젝트 리소스로 저장된 service description 파일을 읽어온다.
            string osifDescJson = readResourceJSONResource(resourceName);

            // 3) JSON Parser를 이용하여 deserialization
            OsifDescription osifDescription =
            JsonConvert.DeserializeObject<OsifDescription>(osifDescJson);

            // 4) JSON 객체를 이용하여 Client 객체 생성
            clientSender = new OSIFClient(osifDescription);

            // 5) 서비스 시작코드 실행
            clientSender.init();
            clientSender.startService();
        }
        catch(Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

#### 2.4.4.2. Global opendata 쓰기 (데이터 제공)

```

private void Button_Click(object sender, RoutedEventArgs e)
{
    if (clientSender == null)
        return;

    // service description에 기술된 global opendata에 데이터를 쓴다. 첫번째 파라미터는
    // service description의 global databus에 정의되어 있어야 한다.
    clientSender.setGlobalAppData("global_data_1", txtSenderMessage.Text);
}

```



#### 2.4.4.3. Global monitoring (데이터 조회)

```
// 1) data listener 함수
protected void listener(string name, string value)
{
    this.txtReceivedMessage.Text = value;
}

private void btnReceiverStart_Click(object sender, RoutedEventArgs e)
{
    try
    {
// 2) 일반적인 시작 코드 작성과 동일

        string resourceName = "OSIF_Sample.data.osif-receiver.json";
        string osifDescJson = readResourceJSONResource(resourceName);

        OsifDescription osifDescription =
        JsonConvert.DeserializeObject<OsifDescription>(osifDescJson);

        clientReceiver = new OSIFClient(osifDescription);

        clientReceiver.init();
        clientReceiver.startService();

// 3) 다른 서비스( "3fb8800e-edd5-473c-97fa-ce461196aaf3" )의 global open data에
subscribe

        clientReceiver.subscribeToGlobalOpendata("3fb8800e-edd5-473c-97fa-ce461196aaf3",
        "global_data_1", listener);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

## 3. OSIF 응용 S/W 실행 환경 준비

### 3.1. 디바이스(Raspberry Pi) 실행 환경 설정

#### 3.1.1. Python 3 설치

Python 3는 OSIF 개발자 지원 도구의 설치 및 활용을 위해 사용되며, Docker-compose의 설치를 위해서도 사용됨.

```
$sudo apt-get install python3
```

#### 3.1.2. Docker 설치

Docker는 OSIF 모듈의 실행 환경으로 동작함.

```
$ curl -fsSL get.docker.com -o get-docker.sh
$ chmod 777 get-docker.sh
$ ./get-docker.sh
$ sudo usermod -aG docker pi
```

#### 3.1.3. OSIF 전용 Docker registry 접속 허용 (임시)

현재 OSIF 전용 Docker registry는 사실 Docker registry를 이용하고 있으며, HTTPS가 적용되지 않은 상태이기 때문에 다음과 같은 설정을 추가해야 함

```
/etc/docker/daemon.json

{
  "insecure-registries": ["osif.iotocean.org:5000"]
}
```

#### 3.1.4. Docker compose 설치

OSIF 모듈의 dockerizing을 간편하게 수행할 수 있도록 하기 위한 툴. Docker 설치 시 기본 포함 도구임. 추가로 설치 필요 시 다음 참고.

```
$ sudo pip install docker-compose
```

### 3.1.5. OSIF 디바이스 플랫폼 설치

```
$ curl http://osif.iotocean.org/dl/docker-compose.yml -o docker-compose.yml  
$ docker-compose up
```

## 4. OSIF 라이브러리 개발자 가이드

### 4.1. OSIF-NODE-PLATFORM 등록

#### 4.1.1. 소스코드 다운로드

```
$ git clone https://github.com/loTKETI/osif-node-platform.git
```

#### 4.1.2. Docker image build

Dockerfile.rpi

```
FROM hyriot/rpi-node:latest
```

```
# 앱 디렉터리 생성
```

```
RUN mkdir -p /usr/src/app
```

```
WORKDIR /usr/src/app
```

```
# 앱 의존성 설치
```

```
COPY package.json /usr/src/app/
```

```
COPY bower.json /usr/src/app/
```

```
COPY .bowerrc /usr/src/app/
```

```
RUN npm install && npm install -g bower
```

```
RUN bower install --allow-root
```

```
EXPOSE 8899
```

```
# 앱 소스 추가  
COPY . /usr/src/app  
  
CMD [ "npm", "start" ]
```

#### 4.1.3. Docker image build

```
$ docker build -f Dockerfile.rpi -t osif/node-platform:rpi .  
$ docker tag osif/node-platform:rpi dev.synctechno.com:5000/osif-node-platform:rpi  
$ docker push dev.synctechno.com:5000/osif-node-platform:rpi
```