

REPORT

KU 건국대학교
KONKUK UNIV.



과목명 | 클라우드IOT서비스

담당교수 | 정 갑 주 교수님

학과 | 컴퓨터공학부

학년 | 4학년

학번 | 201714151

이름 | 박 민 기

제출일 | 2020. 04. 27

목 차

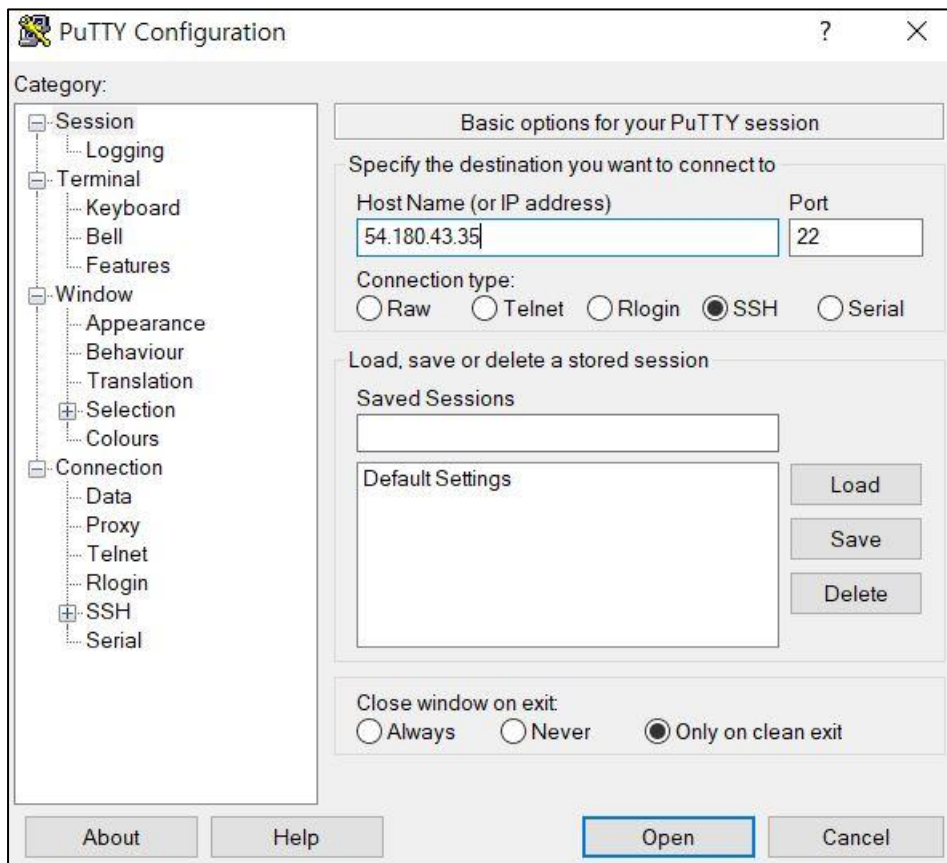
1. Task #1	3
1-1. Message Broker 생성	3
1-2. Putty 연결	3
1-3. Putty Key Authentication	4
1-4. ubuntu 접속	4
1-5. Mosquitto 설치	5
1-6. Localhost를 통하여 Topic이라는 주제를 가진 message Subscribe	6
1-7. Localhost를 통하여 Topic이라는 주제를 갖고 message Publish	6
1-8. 결과 확인 (Message Broker 정상 작동)	7
1-9. 요약 정리	7
2. Task #2	8
2-1. 인스턴스 구성(Sender & Receiver) 및 동적 IP 할당	8
2-2. FileZila를 통하여 FileSender EC2 서버에 file 저장	8
2-3. Sender.js Source Code	8
2-4. Receiver.js Source Code	9
2-5. Vscod 에서 EC2에 접속하여 Receiver.js 실행	11
2-6. Vscod에서 EC2에 접속하여 Sender.js 실행	11
2-7. S3에서 받은 사진과 보낸 사진 비교 확인	12
2-8. 요약 정리	13

1. Task #1

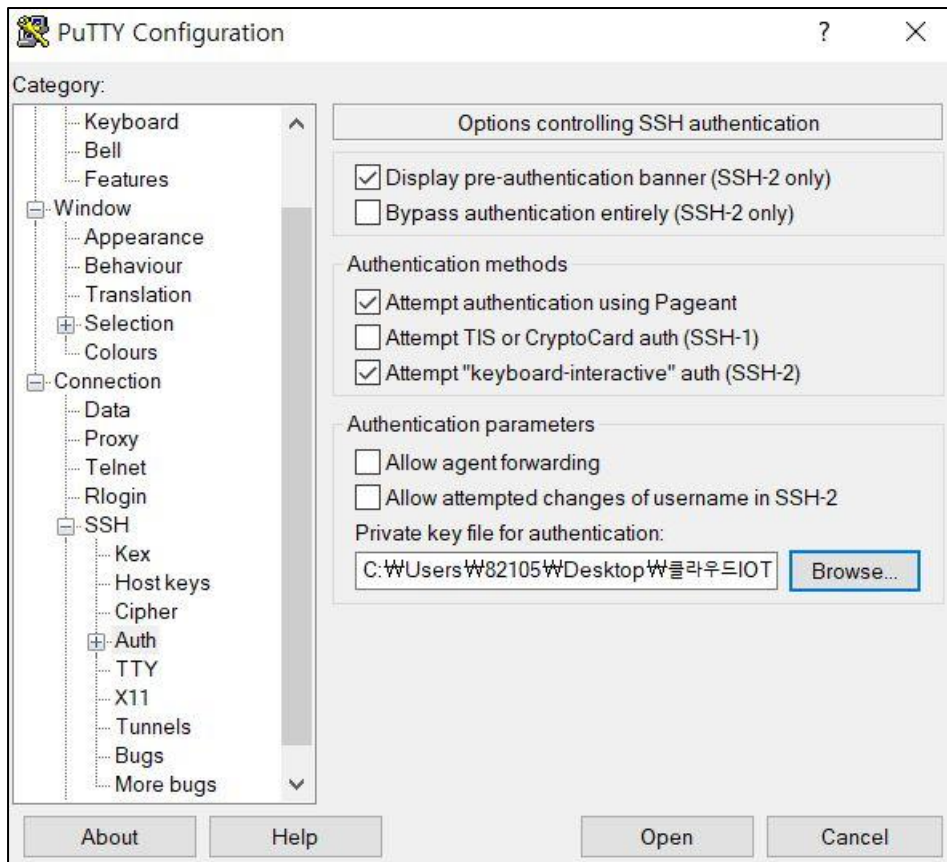
1-1. Message Broker 생성

Name	인스턴스 ID	인스턴스 유형	가용 영역	인스턴스 상태	상태 검사	경보 상태	퍼블릭 DNS(IPv4)	IPv4 퍼블릭 IP
FileReciever	i-089ef2fe6a10b775a	t2.micro	ap-northeast-2c	stopped		없음	ec2-13-125-16-69.ap-n...	13.125.16.69
FileSender	i-090ce3c3f61c79677	t2.micro	ap-northeast-2c	stopped		없음	ec2-13-125-1-156.ap-n...	13.125.1.156
MQTT Message Broker	i-09d9cbd8ef625c654	t2.micro	ap-northeast-2c	running	2/2개 검사 ...	없음	ec2-54-180-43-35.ap-n...	54.180.43.35
2020-cit-vs4	i-0f29179295aa8844f	t2.micro	ap-northeast-2c	stopped		없음		-

1-2. Putty 연결



1-3. Putty Key Authentication



1-4. ubuntu 접속

```
ubuntu@ip-172-31-45-88: ~  
login as: ubuntu  
Authenticating with public key "imported-openssh-key"  
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Mon Apr 20 12:25:32 UTC 2020  
  
System load:  0.0          Processes:            92  
Usage of /:   20.1% of 7.69GB  Users logged in:    0  
Memory usage: 15%          IP address for eth0: 172.31.45.88  
Swap usage:   0%  
  
* Kubernetes 1.18 GA is now available! See https://microk8s.io for docs or  
install it with:  
  
    sudo snap install microk8s --channel=1.18 --classic  
  
* Multipass 1.1 adds proxy support for developers behind enterprise  
firewalls. Rapid prototyping for cloud operations just got easier.  
  
https://multipass.run/
```

1-5. Mosquitto 설치

```
ubuntu@ip-172-31-45-88: ~  
login as: ubuntu  
Authenticating with public key "imported-openssh-key"  
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Mon Apr 20 12:26:52 UTC 2020  
  
System load:  0.0           Processes:            92  
Usage of /:   20.1% of 7.69GB Users logged in:       1  
Memory usage: 15%          IP address for eth0: 172.31.45.88  
Swap usage:   0%  
  
* Kubernetes 1.18 GA is now available! See https://microk8s.io for docs or  
install it with:  
  
    sudo snap install microk8s --channel=1.18 --classic  
  
* Multipass 1.1 adds proxy support for developers behind enterprise  
firewalls. Rapid prototyping for cloud operations just got easier.  
  
    https://multipass.run/  
  
35 packages can be updated.  
0 updates are security updates.  
  
Last login: Mon Apr 20 12:25:33 2020 from 222.239.45.63  
ubuntu@ip-172-31-45-88:~$ sudo apt install mosquitto
```

1-6. Localhost를 통하여 Topic이라는 주제를 가진 message Subscribe



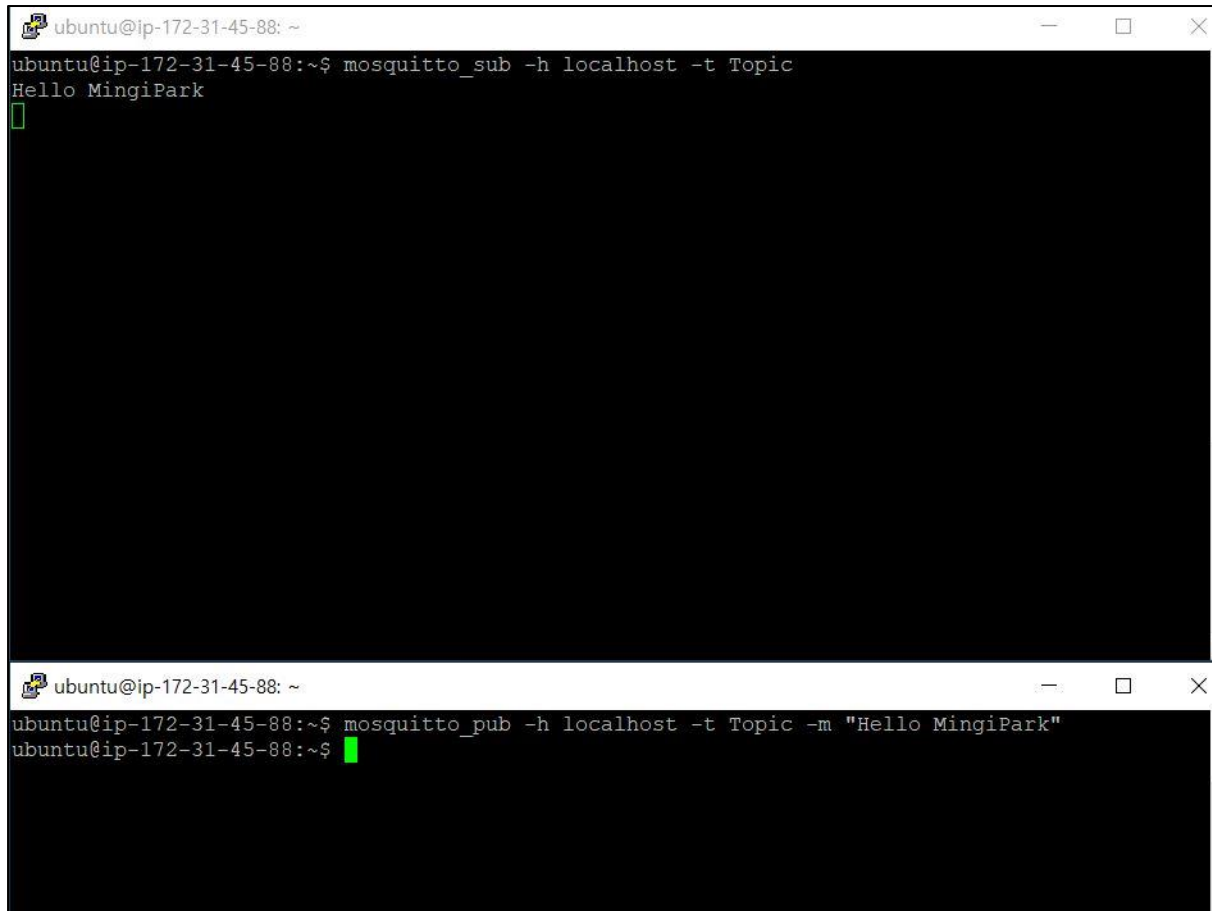
```
ubuntu@ip-172-31-45-88: ~  
ubuntu@ip-172-31-45-88:~$ mosquitto_sub -h localhost -t Topic  
█
```

1-7. Localhost를 통하여 Topic이라는 주제를 갖고 message Publish



```
ubuntu@ip-172-31-45-88: ~  
ubuntu@ip-172-31-45-88:~$ mosquitto_pub -h localhost -t Topic -m "Hello MingiPark"  
█
```


1-8. 결과 확인 (Message Broker 정상 작동)



The image shows two terminal windows from an Ubuntu system with IP 172-31-45-88. The top window shows a subscription command: `mosquitto_sub -h localhost -t Topic`, which receives the message "Hello MingiPark". The bottom window shows a publication command: `mosquitto_pub -h localhost -t Topic -m "Hello MingiPark"`, which is executed successfully.

```
ubuntu@ip-172-31-45-88: ~  
ubuntu@ip-172-31-45-88:~$ mosquitto_sub -h localhost -t Topic  
Hello MingiPark  
[  
  
ubuntu@ip-172-31-45-88: ~  
ubuntu@ip-172-31-45-88:~$ mosquitto_pub -h localhost -t Topic -m "Hello MingiPark"  
ubuntu@ip-172-31-45-88:~$
```

1-9. 요약 정리

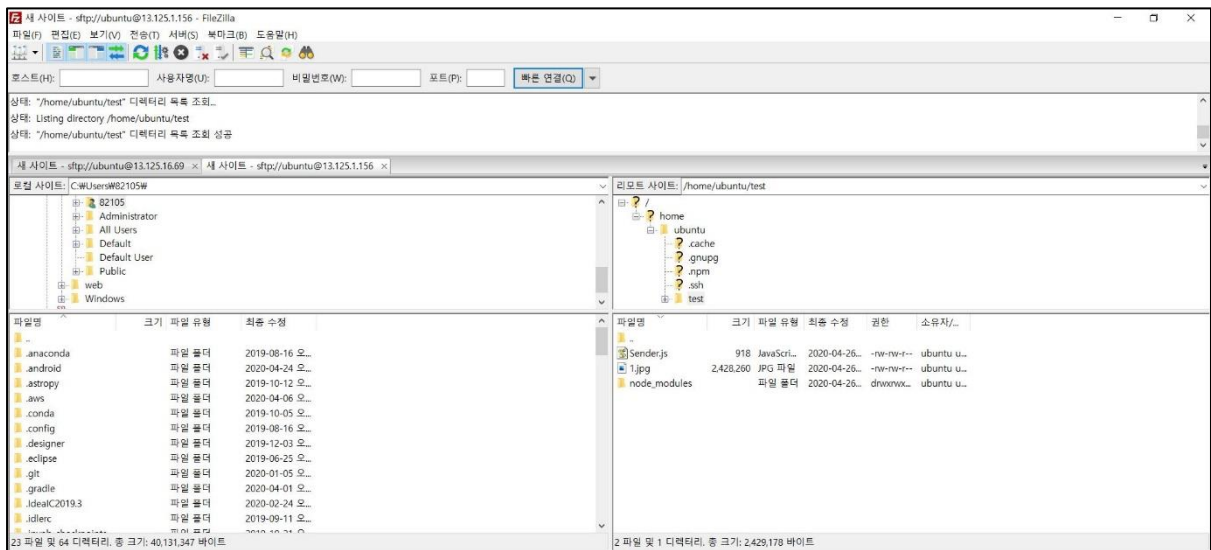
다음의 Task#1 을 수행함으로써 EC2 생성 및 EC2 instance(Virtual Server)에 MQTT Message Broker 을 설치하고 `mosquitto_pub` 과 `mosquitto_sub` command 를 수행해봄으로써 MQTT 구동원리에 대한 이해를 할 수 있었다.

2. Task #2

2-1. 인스턴스 구성(Sender & Receiver) 및 동적 IP 할당

	Name	인스턴스 ID	인스턴스 유형	가용 영역	인스턴스 상태	상태 검사	경보 상태	퍼블릭 DNS(IPv4)	IPv4 퍼블릭 IP
	FileReceiver	i-089ef2fe6a10b775a	t2.micro	ap-northeast-2c	running	2/2개 검사 ...	없음	ec2-13-125-16-69.ap-n...	13.125.16.69
	FileSender	i-090ce3c3f61c79677	t2.micro	ap-northeast-2c	running	2/2개 검사 ...	없음	ec2-13-125-1-156.ap-n...	13.125.1.156
	MQTT Message Broker	i-09d9cbd8e625c654	t2.micro	ap-northeast-2c	running	2/2개 검사 ...	없음	ec2-54-180-43-35.ap-n...	54.180.43.35

2-2. FileZilla를 통하여 FileSender EC2 서버에 file 저장



2-3. Sender.js Source Code

```
var AWS = require('aws-sdk');
var mqtt = require('mqtt');
var fs = require('fs')
var util = require('util');

var s3 = new AWS.S3();

let file;
let sender;

param = { filePath : '1.jpg'}

async function sendFile(x) {
  try {
    sender = mqtt.connect('mqtt://54.180.43.35');
    sender.on('connect', () => sender.subscribe('hw1', () => { console.log('File Sender subscribes hw1!'); }));
```



```

        file = fs.createReadStream(x.filePath);
        let buffer = []
        for await(let chunk of file) {
            buffer.push(chunk);
        }
        let realContents = Buffer.concat(buffer);

        sender.publish('hw1', realContents);

        console.log('Finish the upload image file on mqtt!\n\n');
        console.log('\n');
        sender.end();

    } catch (err) {
        console.log(err);
    }
}

sendFile(param);

```

2-4. Receiver.js Source Code

```

var AWS = require('aws-sdk');
var s3 = new AWS.S3();
var mqtt = require('mqtt');
var receiver = mqtt.connect('mqtt://54.180.43.35');

var x = {
    bucket: "myreceiver"
}

var Readable = require('stream').Readable;
function bufferToStream(buffer) {
    var stream = new Readable();
    stream.push(buffer);
    stream.push(null);
    return stream;
}

function createBucket(bucket_name) {
    var cb_params = {
        Bucket: bucket_name
    };
    return new Promise(function(resolve, reject) {
        s3.createBucket(cb_params, function(err, data) {
            if (err) reject(err);
            else resolve(data);
        });
    });
}

```

```

    })
  })
}

function createObject(bucket_name, stream) {
  const co_params = {
    Bucket: bucket_name,
    Key: "hw1/Image/1_copy.jpg",
    ACL: "public-read",
    Body: stream
  }

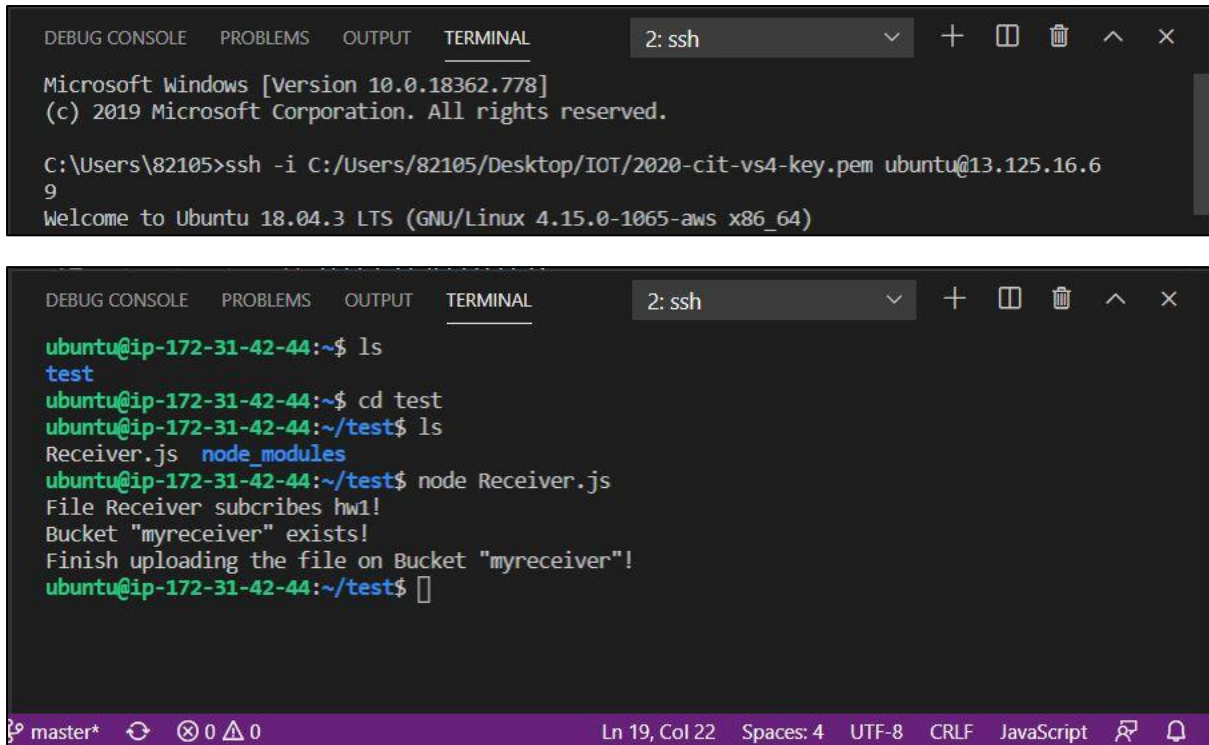
  return new Promise(function(resolve, reject) {
    s3.upload(co_params, function(err, data) {
      if (err) reject(err);
      else resolve(data);
    })
  })
}

receiver.on('connect', () => {
  receiver.subscribe('hw1', () => {console.log('File Receiver subscribes hw1!')})
});

receiver.on('message', async function(topic, message){
  let readableStream = bufferToStream(message);
  try {
    try {
      const bucket = await s3.headBucket({Bucket : x.bucket}).promise();
      console.log(`Bucket "${x.bucket}" exists!`);
    } catch (err) {
      var newBucket = await createBucket(x.bucket);
      console.log(`Create Bucket "${x.bucket}"!`);
    }
    var newObj = await createObject(x.bucket, readableStream);
    console.log(`Finish uploading the file on Bucket "${x.bucket}"!`)
    receiver.end();
  } catch (err) {
    console.log(err);
  }
});

```

2-5. Vscode 에서 EC2에 접속하여 Receiver.js 실행



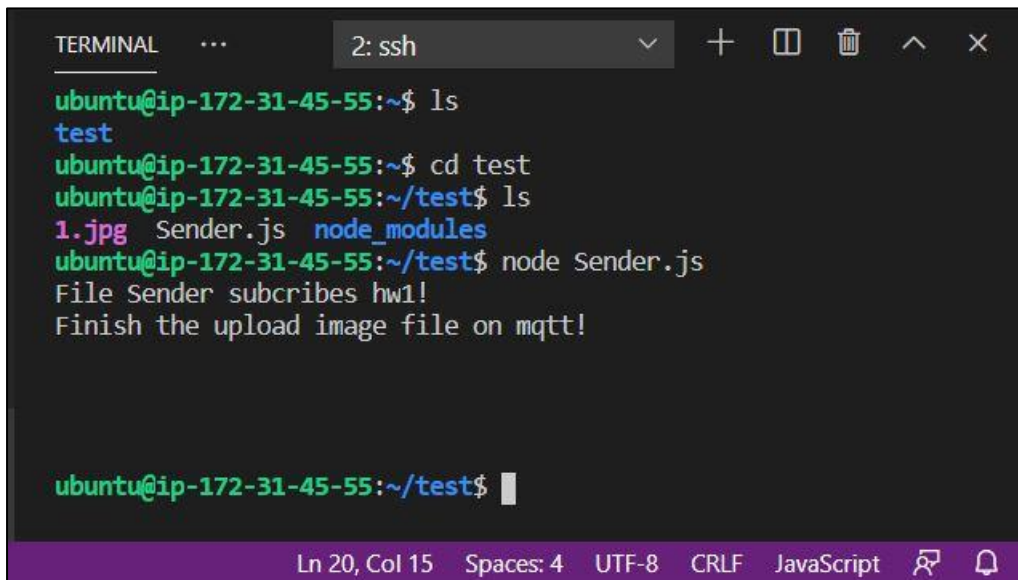
```
DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL  2: ssh  +  [ ]  [X]  ^  X

Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\82105>ssh -i C:/Users/82105/Desktop/IOT/2020-cit-vs4-key.pem ubuntu@13.125.16.6
9
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1065-aws x86_64)

ubuntu@ip-172-31-42-44:~$ ls
test
ubuntu@ip-172-31-42-44:~$ cd test
ubuntu@ip-172-31-42-44:~/test$ ls
Receiver.js  node_modules
ubuntu@ip-172-31-42-44:~/test$ node Receiver.js
File Receiver subscribes hw1!
Bucket "myreceiver" exists!
Finish uploading the file on Bucket "myreceiver"!
ubuntu@ip-172-31-42-44:~/test$
```

2-6. Vscode에서 EC2에 접속하여 Sender.js 실행

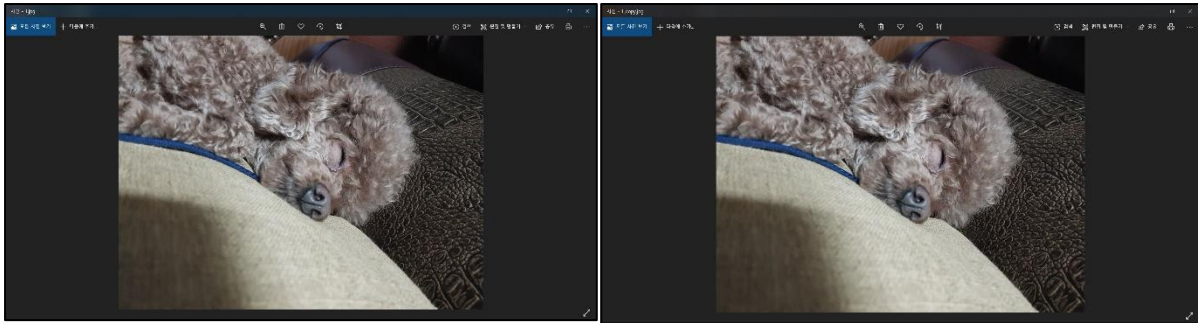


```
TERMINAL  ...  2: ssh  +  [ ]  [X]  ^  X

ubuntu@ip-172-31-45-55:~$ ls
test
ubuntu@ip-172-31-45-55:~$ cd test
ubuntu@ip-172-31-45-55:~/test$ ls
1.jpg  Sender.js  node_modules
ubuntu@ip-172-31-45-55:~/test$ node Sender.js
File Sender subscribes hw1!
Finish the upload image file on mqtt!

ubuntu@ip-172-31-45-55:~/test$
```

2-7. S3에서 받은 사진과 보낸 사진 비교 확인



<원본 파일>

<전송해서 받은 파일>

Amazon S3

버킷 (3) ARN 복사 비어 있음 삭제 버킷 만들기

버킷은 Amazon S3에서 데이터 스토리지에 사용되는 기본 컨테이너입니다. 사용자의 버킷에 있는 객체에 다른 사용자가 액세스할 수 있도록 하려면 해당 사용자에게 명시적으로 권한을 부여해야 합니다. [자세히 알아보기](#)

이름으로 버킷 찾기

	이름	리전	액세스	버킷 생성됨
<input type="radio"/>	mingi	아시아 태평양(서울) ap-northeast-2	객체를 퍼블릭으로 설정할 수 있음	2020-04-11T04:05:25.000Z
<input type="radio"/>	mingibucket	아시아 태평양(서울) ap-northeast-2	퍼블릭 아님	2020-04-04T07:10:11.000Z
<input type="radio"/>	myreceiver	아시아 태평양(서울) ap-northeast-2	객체를 퍼블릭으로 설정할 수 있음	2020-04-20T13:44:59.000Z

<버킷 목록>

Amazon S3 > myreceiver

myreceiver

개요 **속성** 권한 관리 액세스 지점

업로드 + 폴더 만들기 다운로드 작업

아시아 태평양(서울)

이 버킷은 비어 있습니다. 시작하려면 새 객체를 업로드합니다.

객체 업로드
 객체 속성 설정
 객체 권한 설정

<보내기 전 비어있는 버킷>



<전송 이후 파일을 받은 상태의 버킷>

2-8. 요약 정리

Task#2를 수행함에 있어서는 조금의 어려움이 존재하였다. 먼저, 파일을 EC2 서버에 올려놓기 위하여 FileZilla Program을 사용하여 직접 해당 라이브러리에 접근하였다. 그래서 Sender 서버에 복사할 파일(1.jpg)를 올려두고 Sender.js 파일을 구동시켜 해당 파일의 복사본(1_copy.jpg)를 만들었다. 그리고 Broker의 서버에 'hw1' 이라는 topic으로 파일을 전송하였다. 그런 다음 Receiver.js 파일을 FileReceiver EC2 instance에서 수행하여 해당 파일을 받고, 파일을 S3에 옮겨 그 결과를 확인하였다.