

Cloud IoT Services

Homework Assignment #1 Report

컴퓨터공학부 201814121 이연희

1. The configuration of the MQTT message broker
 - 1) elastic IP: 52.78.92.146
 - 2) 퍼블릭 IPv4 DNS: ec2-52-78-92-146.ap-northeast-2.compute.amazonaws.com
 - 3) topic name: topic1
 - 4) 사용자이름: ubuntu
 - 5) 추가한 인바운드 규칙: 사용자지정 TCP, TCP, 1883, 0.0.0.0/0

| 인바운드 규칙 (3) | | | | 인바운드 규칙 편집 |
|-------------|------|-------|-----------|------------|
| 유형 | 프로토콜 | 포트 범위 | 소스 | 설명 - 선택 사항 |
| SSH | TCP | 22 | 0.0.0.0/0 | - |
| SSH | TCP | 22 | ::/0 | - |
| 사용자 지정 TCP | TCP | 1883 | 0.0.0.0/0 | - |

2. The steps to have been taken to accomplish the tasks
 - 1) aws ec2 가상 서버에서 초기 세팅: npm, mosquitto, mosquitto-client 등 설치
 - 2) 인바운드 규칙 창에서 포트 넘버가 1883 인 TCP 유형 생성
 - 3) 가상 서버에서 receiver.js 와 sender.js 소스 파일 생성
 - 4) mqtt 프로토콜을 통해 전송할 파일인 testfile.txt 을 생성
 - 5) receiver.js 를 먼저 실행시켜서 토픽(topic1)에 해당하는 데이터가 오기를 대기
 - 6) sender.js 를 실행시켜서 testfile.txt 를 토픽(topic1)의 데이터로 브로커에게 전달
 - 7) subscriber 가 전송을 받은 내용을 received.txt 파일 생성하고 화면에 출력

3. The screen snapshots

1) receiver.js, sender.js 와 전송할 파일인 testfile.txt 를 생성했다.

The image contains four terminal window screenshots arranged in a 2x2 grid, all from an Ubuntu machine with IP 172-31-13-254.

- Top-left:** Shows the creation of `sender.js`. The code connects to an MQTT broker at `localhost:1883`, reads the contents of `testfile.txt` (which contains "Hello world", "12345", and "abcd"), and publishes this data to the `topic1` MQTT topic.
- Top-right:** Shows the creation of `receiver.js`. The code connects to the same MQTT broker, subscribes to `topic1`, and when a message is received, it parses the JSON message and writes the data to `received.txt`.
- Bottom-left:** Shows the execution of `sender.js`. The terminal output displays the contents of `testfile.txt`:
Hello world
12345
abcd
- Bottom-right:** Shows the directory listing (`ls`) in the current directory, displaying `node-js.png`, `package-lock.json`, `sender.js`, `node_modules`, `receiver.js`, and `testfile.txt`.

2) receiver.js 를 먼저 실행시켜서 topic1 을 구독한 상태이다. Subscribe 을 성공하고 화면에 "Subscribe topic1"을 출력했다.

The image contains two terminal window screenshots from the same Ubuntu machine.

- Left window:** Shows the execution of `sender.js`. The terminal output is:
node-js.png package-lock.json sender.js
node_modules receiver.js testfile.txt
- Right window:** Shows the execution of `node receiver.js`. The terminal output is:
Subscribe topic1

- 3) sender.js 를 실행시켜서 메시지 브로커에게 topic1 에 해당하는 텍스트 파일 데이터 testfile.txt 를 보냈다. receiver.js 를 실행시킨 쉘에서는 받은 데이터를 received.txt 로 저장하고 화면에 출력하게 했다.

```
ubuntu@ip-172-31-13-254: ~  
ubuntu@ip-172-31-13-254:~$ ls  
node-js.png  package-lock.json  sender.js  
node_modules receiver.js  testfile.txt  
ubuntu@ip-172-31-13-254:~$ node sender.js  
Send "Hello world\n12345\nabcde\n" to topic1  
ubuntu@ip-172-31-13-254:~$  
  
ubuntu@ip-172-31-13-254:~$ node receiver.js  
Subscribe topic1  
==received.txt==  
Hello world  
12345  
abcde  
  
ubuntu@ip-172-31-13-254:~$
```

- 4) testfile.txt 와 동일한 데이터가 received.txt 에 저장된 것을 확인했다.

```
ubuntu@ip-172-31-13-254: ~  
ubuntu@ip-172-31-13-254:~$ ls  
node-js.png  package-lock.json  sender.js  
node_modules receiver.js      testfile.txt  
ubuntu@ip-172-31-13-254:~$ node sender.js  
Send "Hello world\n12345\nabcde\n" to topic1  
ubuntu@ip-172-31-13-254:~$  
ubuntu@ip-172-31-13-254:~$  
ubuntu@ip-172-31-13-254:~$ ls  
node-js.png  package-lock.json  receiver.js  testfile.txt  
node_modules received.txt      sender.js  
ubuntu@ip-172-31-13-254:~$
```

[receiver.js]

```
var mqtt = require('mqtt');
var client = mqtt.connect('mqtt://localhost:1883');
var fs = require('fs');

client.on('connect', function(){
  client.subscribe('topic1',()=>{console.log('Subscribe topic1')}});
});

client.on('message', function(topic, message){
  var data = JSON.parse(message);
  fs.writeFileSync('received.txt', data);
  console.log('==received.txt==\n'+fs.readFileSync('received.txt', 'utf8'));
  client.end();
});
```

[sender.js]

```
var mqtt = require('mqtt');
var client = mqtt.connect('mqtt://localhost:1883');

var fs = require('fs');
var file = 'testfile.txt';
var data = fs.readFileSync(file, 'utf8');

client.on('connect', function(){
  client.subscribe('topic1');
  client.publish('topic1', JSON.stringify(data));
});

client.on('message', function(topic, message){
  console.log("Send "+ message +" to " + topic);
  client.end();
});
```