

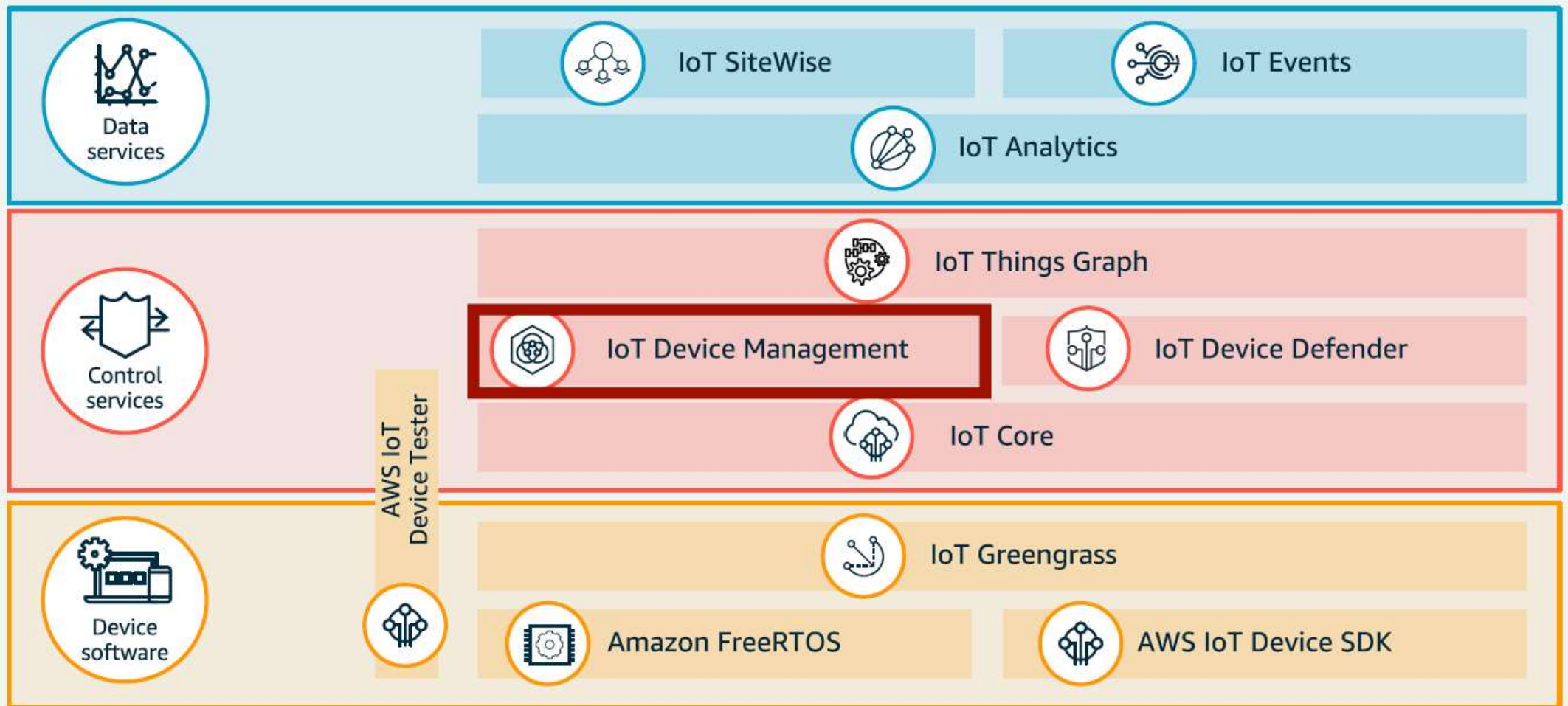
# AWS 활용 IoT

## [2] AWS IoT : Job



강사 : 고병화

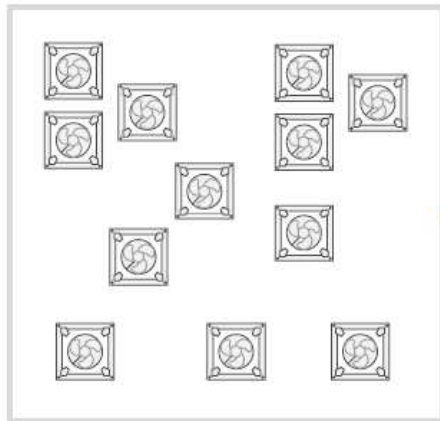
# AWS IoT architecture



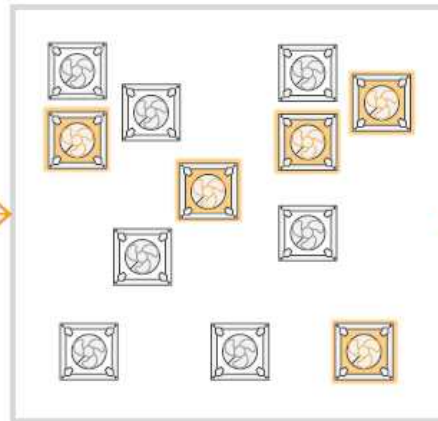


# AWS IoT Device Management

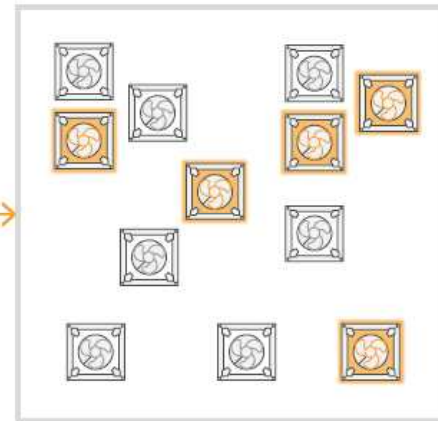
AWS IoT Device Management helps you register, organize, monitor, and remotely manage your growing fleet of connected devices.



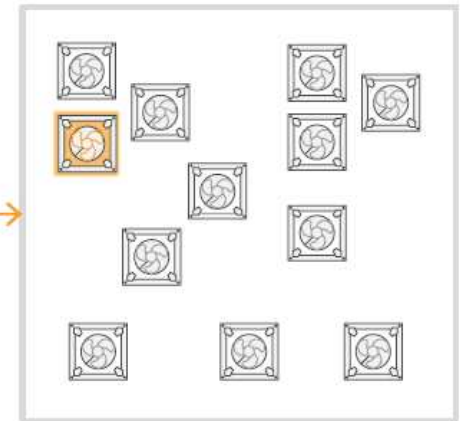
Fast device  
registration at scale



Real-time fleet  
indexing and search



Monitoring and  
updating devices



Access individual device  
securely



Connectivity  
& Control  
Services

IoT Jobs

# IoT Jobs : What is "IoT Job"?

AWS IoT 작업을 사용하여 AWS IoT에 연결된 하나 이상의 디바이스로 전송되고 실행되는 원격 작업 집합을 정의할 수 있다.

예를 들어 애플리케이션을 다운로드하여 설치하거나, 펌웨어 업데이트를 실행하거나, 재부팅 하거나, 인증서를 교체하거나, 원격 문제 해결 작업을 수행하도록 디바이스 집합에 지시하는 작업을 정의할 수 있다.

[https://docs.aws.amazon.com/ko\\_kr/iot/latest/developerguide/iot-jobs.html](https://docs.aws.amazon.com/ko_kr/iot/latest/developerguide/iot-jobs.html)

# AWS IoT Jobs

Define a set of remote operations that are sent to and executed on one or more devices connected to AWS IoT.

Typical set of operations could be –

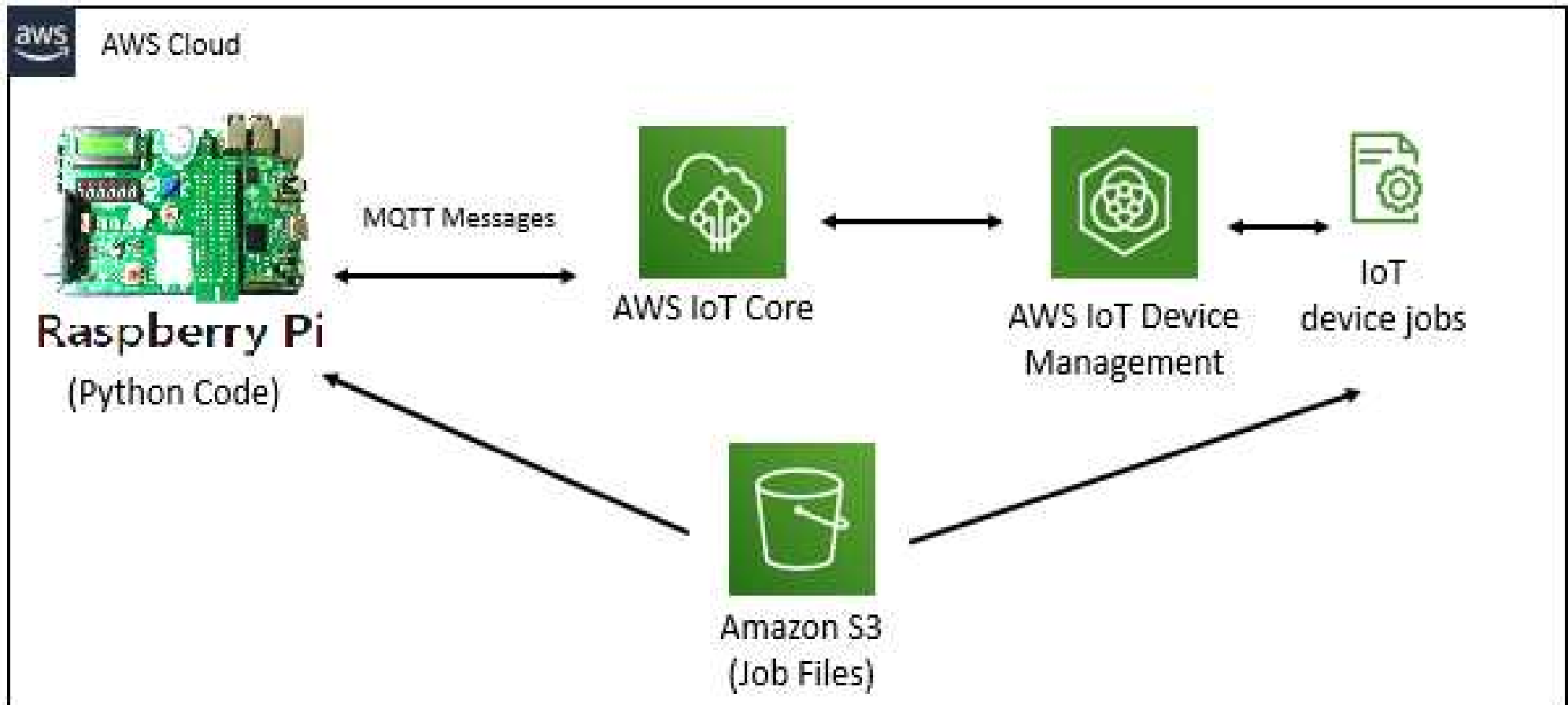
- Download a file
- Install application or upgrade firmware
- Reboot
- Rotate certificate

AWS IoT Device Management manages the jobs.

# IoT Jobs : Job 구성

- **롤아웃(Rollout)**: 이 구성은 분당 작업 문서를 수신하는 디바이스 수를 정의한다(작업 개시).
- **중단(Abort)**: 일부 디바이스가 작업 알림을 받지 못하거나 디바이스에서 작업 실행에 대한 실패를 보고하는 경우와 같은 사례에서 작업을 취소하려면 이 구성을 사용한다.
- **시간 제한(Timeout)**: 작업 실행이 시작된 후 일정 기간 내에 작업 대상에서 응답이 없는 경우 작업이 실패할 수 있다.
- **재시도(Retry)**: 디바이스가 작업 실행을 완료하려고 할 때 실패를 보고하거나 작업 실행 시간이 초과된 경우 이 구성을 사용하여 디바이스에 대한 작업 실행을 재시도할 수 있다.

# IoT Jobs





# IoT Jobs

## AWS IoT Jobs – Job File



Device

Receive \ Subscribe Job

Process based on Job  
(business logic code)

Updates status

Job File  
(json)

Configure Job

Publish Job

Shows updates status

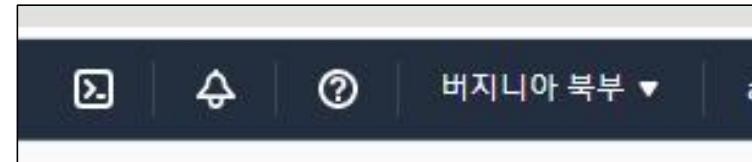
Mark Job as Complete /  
Fail



AWS IoT Device  
Management

# IoT Jobs : S3 버킷 생성

AWS 콘솔의 우측상단의 CloudShell 아이콘을 눌러 AWS CloudShell을 실행시킨다



CloudShell에서 아래 S3 버킷 생성 명령을 실행시킨다  
(버킷의 이름은 다른 사람과 다른 고유한 이름을 사용한다, '-'사용불가)

**S3\_BUCKET=iot-job-bucket-0504**

```
aws s3api create-bucket ₩  
  --bucket $S3_BUCKET ₩  
  --region us-east-1
```

# IoT Jobs : Job document 파일 업로드

AWS 콘솔의 s3의 버킷으로 가서 생성된 버킷을 클릭하고  
job-document.json 파일을 버킷에 업로드 한다

**버킷 (1)** [정보](#)  
버킷은 S3에 저장되는 데이터의 컨테이너입니다. [자세히 알아보기](#)

	이름	AWS 리전
<input type="radio"/>	iot-job-bucket-0504	미국 동부(버지니아 북부) us-east-1

**업로드** [정보](#)  
S3에 업로드할 파일 및 폴더를 추가합니다. 160GB보다 큰 파일을 업로드하려면 AWS CLI, AWS SDK 또는 Amazon S3 REST API를 사용합니다. [자세히 알아보기](#)

여기에 업로드할 파일과 폴더를 끌어서 놓거나, [파일 추가] 또는 [폴더 추가]를 선택합니다.

**파일 및 폴더 (1 합계, 86.0B)** [제거](#) [파일 추가](#) [폴더 추가](#)  
이 테이블의 모든 파일과 폴더가 업로드됩니다.

<input type="checkbox"/>	이름	폴더	유형	크기
<input type="checkbox"/>	job-document.json	-	application/json	86.0B

# IoT Jobs : Job 생성

AWS CloudShell에서 아래 Job 생성 명령을 실행시키다

**JOB\_ID=job01**

**JOB\_THING\_ARN=arn:aws:iot:us-east-1:844311781633:thing/RaspberryPi**

**aws iot create-job --job-id \$JOB\_ID \**

**--targets \$JOB\_THING\_ARN \**

**--document-source https://s3.amazonaws.com/\$S3\_BUCKET/job-document.json**

```
[cloudshell-user@ip-10-0-59-5 ~]$ aws iot create-job --job-id $JOB_ID \  
> --targets $JOB_THING_ARN \  
> --document-source https://s3.amazonaws.com/$S3_BUCKET/job-document.json  
{  
  "jobArn": "arn:aws:iot:us-east-1:844311781633:job/job01",  
  "jobId": "job01"  
}
```

# IoT Jobs : Job 생성

AWS IoT 콘솔의 [관리]→[원격 작업]→[작업]에서 생성된 job이 보인다  
job1을 클릭하여 [작업 문서]탭을 클릭하면 JSON 문서 내용이 보인다

**작업 (1) 정보**  
작업은 AWS IoT에 연결된 하나 이상의 디바이스로 전송하고 실행할 원격 작업 세트를 정의합니다. 원격 작업이 자주 수

<input type="checkbox"/>	이름	유형	상태
<input type="checkbox"/>	job01	스냅샷	진행 중 - 롤아웃 완료

**job01 정보**

**작업 문서 정보**  
작업 문서는 대상 디바이스가 수행할 원격 작업을 설명합니다. 작업 문서는

```
{
  "operation": "sys-info",
  "sys-info": "uptime",
  "topic": "sys/info"
}
```

# IoT Jobs : IoT 정책 추가

AWS IoT콘솔의 [보안]→[정책]에 있는 “RaspberryPi-Policy”를 클릭하고  
[활성 버전 편집]→[JSON]을 클릭하고

- 16번 라인 썸에

" arn:aws:iot:us-east-1:844311781633:topic/sys/info“를 추가하고

- 28번 라인 썸에

"arn:aws:iot:us-east-1:844311781633:topicfilter/sys/info"를 추가해 준다  
(다음 페이지 참조)

[편집한 버전을 이 정책의 활성 버전으로 설정]를 체크하고  
[새버전으로 저장] 버튼을 클릭한다

# IoT Jobs : IoT 정책 추가

```
10 ▼    "Resource": [  
11        "arn:aws:iot:us-east-1:844311781633:topic/sdk/test/java",  
12        "arn:aws:iot:us-east-1:844311781633:topic/sdk/test/Python",  
13        "arn:aws:iot:us-east-1:844311781633:topic/iot/sensor",  
14        "arn:aws:iot:us-east-1:844311781633:topic/$aws/things/RaspberryPi/shadow/*",  
15        "arn:aws:iot:us-east-1:844311781633:topic/$aws/things/RaspberryPi/jobs/*",  
16        "arn:aws:iot:us-east-1:844311781633:topic/sys/info"  
17    ]  
18 },  
19 ▼    {  
20        "Effect": "Allow",  
21        "Action": "iot:Subscribe",  
22 ▼    "Resource": [  
23        "arn:aws:iot:us-east-1:844311781633:topicfilter/sdk/test/java",  
24        "arn:aws:iot:us-east-1:844311781633:topicfilter/sdk/test/Python",  
25        "arn:aws:iot:us-east-1:844311781633:topicfilter/iot/sensor",  
26        "arn:aws:iot:us-east-1:844311781633:topicfilter/$aws/things/RaspberryPi/shadow/*",  
27        "arn:aws:iot:us-east-1:844311781633:topicfilter/$aws/things/RaspberryPi/jobs/*",  
28        "arn:aws:iot:us-east-1:844311781633:topicfilter/sys/info"  
29    ]
```

# IoT Jobs : Shell Script 준비

Pi3보드의 터미널에서 아래 명령을 실행 시킨다

```
tail -1 start.sh > run_job_uptime.sh
```

```
chmod +x run_job_uptime.sh
```

생성된 스크립트 파일을 확인해본다(endpoint 주소 값은 각자 다르다  
AWS IoT 콘솔의 설정에서 “디바이스 데이터 엔드포인트” 확인 가능)

```
cat run_job_uptime.sh
```

```
python3 aws-iot-device-sdk-python-v2/samples/jobs_uptime.py --endpoint  
a1mp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com --ca_file root-CA.crt --  
cert RaspberryPi.cert.pem --key RaspberryPi.private.key --client_id  
Raspi_Dev01 --thing_name RaspberryPi(다음 페이지의 변경 후 결과임)
```



# IoT Jobs : Shell Script 준비

생성된 run\_jobs\_uptime.sh 파일을 편집기로 아래 부분을 수정해준다

[원본]

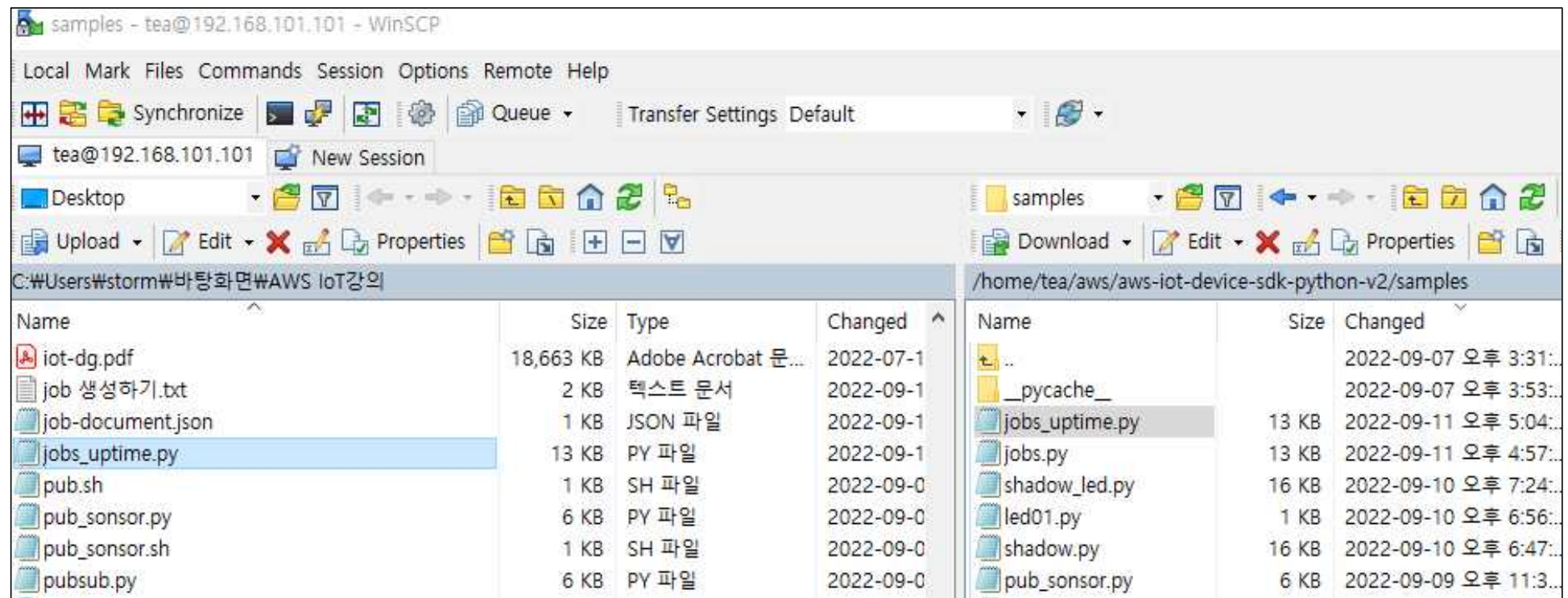
```
python3 aws-iot-device-sdk-python-v2/samples/pubsub.py --endpoint  
a1mp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com --ca_file root-CA.crt --  
cert RaspberryPi.cert.pem --key RaspberryPi.private.key --client_id  
basicPubSub --topic sdk/test/Python --count 0
```

[변경]

```
python3 aws-iot-device-sdk-python-v2/samples/jobs_uptime.py --  
endpoint a1mp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com --ca_file  
root-CA.crt --cert RaspberryPi.cert.pem --key RaspberryPi.private.key --  
client_id Raspi_Dev01 --thing_name RaspberryPi
```

# IoT Jobs : 디바이스 소스파일 복사

WinSCP를 사용하여 배포된 jobs\_uptime.py 소스 파일을 Pi3 보드의 sample 디렉토리 아래에 복사한다



WinSCP interface showing file transfer between a local PC and a Raspberry Pi 3. The local side shows a directory listing with files like jobs\_uptime.py. The remote side shows the /home/tea/aws/aws-iot-device-sdk-python-v2/samples directory, where jobs\_uptime.py is being copied.

Name	Size	Type	Changed
iot-dg.pdf	18,663 KB	Adobe Acrobat 문...	2022-07-1
job 생성하기.txt	2 KB	텍스트 문서	2022-09-1
job-document.json	1 KB	JSON 파일	2022-09-1
jobs_uptime.py	13 KB	PY 파일	2022-09-1
pub.sh	1 KB	SH 파일	2022-09-0
pub_sensor.py	6 KB	PY 파일	2022-09-0
pub_sensor.sh	1 KB	SH 파일	2022-09-0
pubsub.py	6 KB	PY 파일	2022-09-0

Name	Size	Changed
..		2022-09-07 오후 3:31..
__pycache__		2022-09-07 오후 3:53..
jobs_uptime.py	13 KB	2022-09-11 오후 5:04..
jobs.py	13 KB	2022-09-11 오후 4:57..
shadow_led.py	16 KB	2022-09-10 오후 7:24..
led01.py	1 KB	2022-09-10 오후 6:56..
shadow.py	16 KB	2022-09-10 오후 6:47..
pub_sensor.py	6 KB	2022-09-09 오후 11:3..

# IoT Jobs : MQTT 테스트 클라이언트 준비

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]의 [주제 구독]에 “sys/info”를 입력하고 [구독] 버튼을 클릭한다  
아직 아무런 메시지도 나타나지 않는다

## MQTT 테스트 클라이언트 정보

MQTT 테스트 클라이언트를 사용하여 AWS 계정에서 전달되는 MQTT 메시지를 변경 사항 및 이벤트를 알립니다. MQTT 테스트 클라이언트를 사용하여 MQTT

주제 구독

주제 게시

주제 필터 정보

주제 필터는 구독할 주제를 설명합니다. 주제 필터에는 MQTT 와일드카드 문자가 포함될

sys/info

▶ 추가 구성

구독

구독

sys/info

sys/info

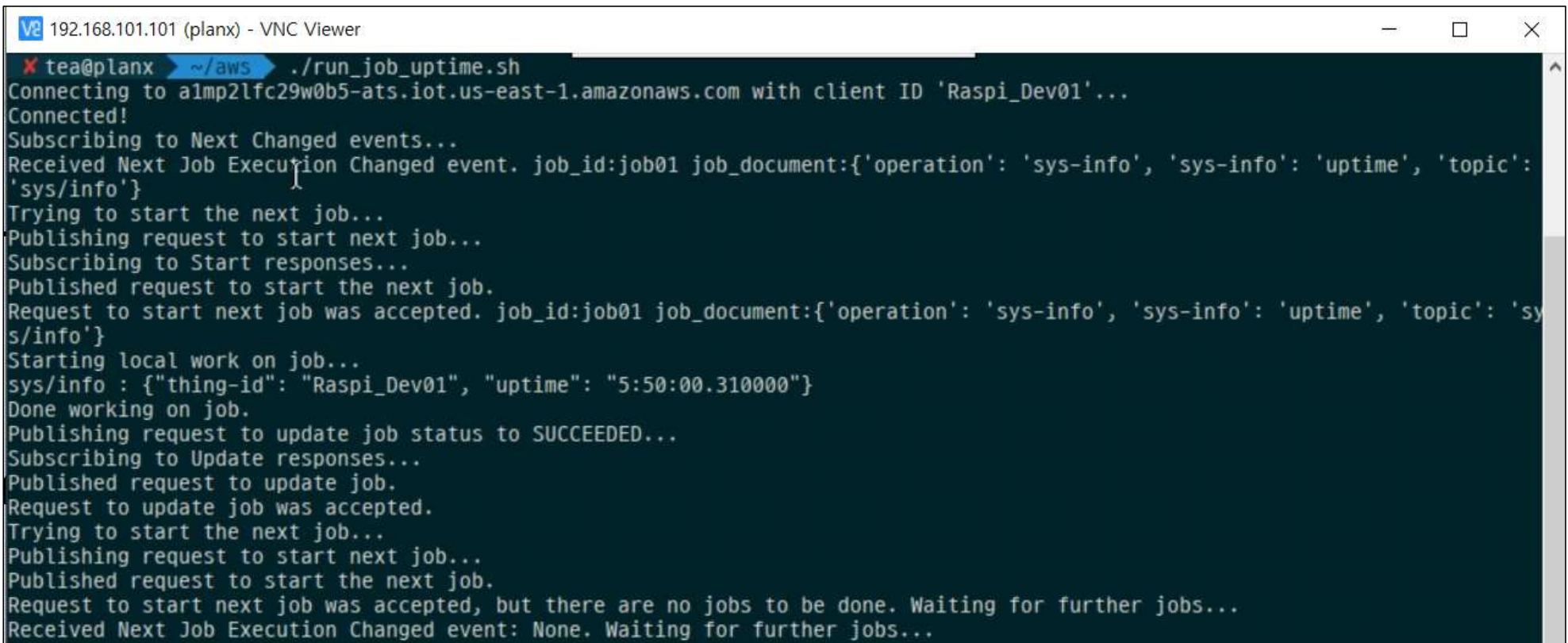
♡ ✕

No messages have been sent to this

# IoT Jobs : Job 소스실행(디바이스)

Pi3보드에서 run\_job\_uptime.sh 을 실행한다

./run\_job\_uptime.sh

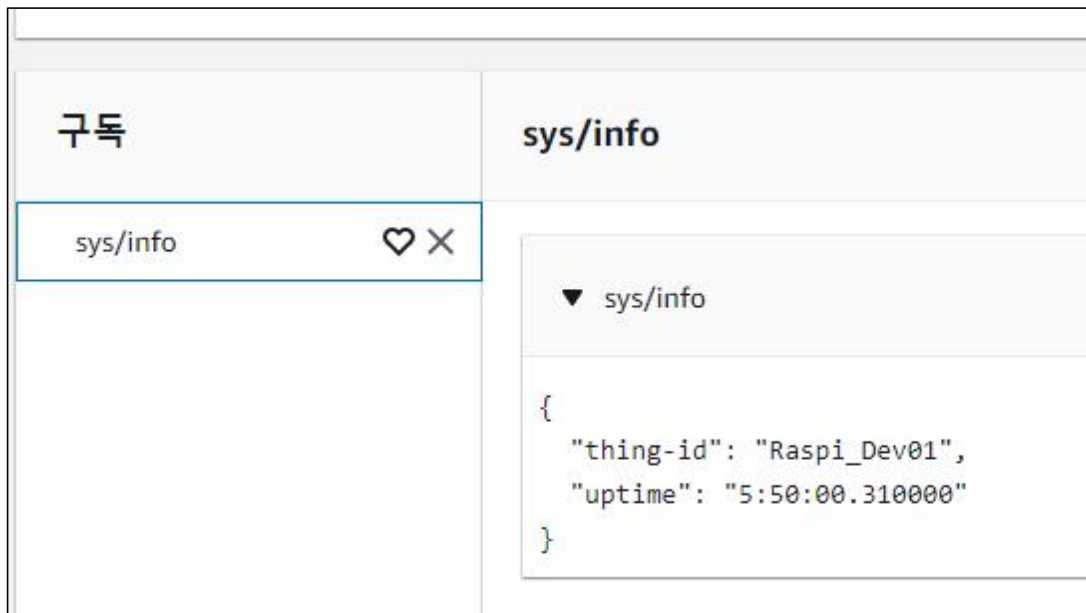


```
192.168.101.101 (planx) - VNC Viewer
tea@planx ~/aws$ ./run_job_uptime.sh
Connecting to almp2lfc29w0b5-ats.iot.us-east-1.amazonaws.com with client ID 'Raspi_Dev01'...
Connected!
Subscribing to Next Changed events...
Received Next Job Execution Changed event. job_id:job01 job_document:{'operation': 'sys-info', 'sys-info': 'uptime', 'topic': 'sys/info'}
Trying to start the next job...
Publishing request to start next job...
Subscribing to Start responses...
Published request to start the next job.
Request to start next job was accepted. job_id:job01 job_document:{'operation': 'sys-info', 'sys-info': 'uptime', 'topic': 'sys/info'}
Starting local work on job...
sys/info : {"thing-id": "Raspi_Dev01", "uptime": "5:50:00.310000"}
Done working on job.
Publishing request to update job status to SUCCEEDED...
Subscribing to Update responses...
Published request to update job.
Request to update job was accepted.
Trying to start the next job...
Publishing request to start next job...
Published request to start the next job.
Request to start next job was accepted, but there are no jobs to be done. Waiting for further jobs...
Received Next Job Execution Changed event: None. Waiting for further jobs...
```

# IoT Jobs : MQTT 테스트 클라이언트

AWS IoT 콘솔의 [MQTT 테스트 클라이언트]에 수신된 디바이스의 uptime 값이 보인다

( “5:50:00.310000”은 디바이스 부팅 후 5시간 50분 경과를 나타낸다)



생성된 **job01**의 작업 상태는 **완료됨**으로 변경된다



# IoT Jobs : Job 삭제

만일 동작이 잘 되지 않을 경우 AWS CloudShell 에서 생성된 작업 job01을 삭제하고 다시 만들어 보드에서 프로그램을 재 실행해 본다

\* Job 삭제

**JOB\_ID=job01**

**aws iot delete-job --job-id \$JOB\_ID --force**

\* 작업 목록 조회

**aws iot list-jobs**



# IoT Jobs : 작업 템플릿 사용

AWS 관리형 템플릿을 사용하면 이미 만들어 놓은 JSON 작업 파일을 그대로 복사하여 사용할 수 있다

- 템플릿에서  
작업 템플릿을 선택하여 작업 문서 및 작업 구성을 재사용합니다. 배포 전에 파일 및 해당 구성을 사용자 지정할 수 있습니다.

<b>AWS-Download-File</b> A managed job template for downloading a file. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Install-Application</b> A managed job template for installing one or more applications. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Reboot</b> A managed job template for rebooting the device. 환경: LINUX 최신 버전: 1.0	AWS 관리형 ✓
<b>AWS-Remove-Application</b> A managed job template for uninstalling one or more applications. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Restart-Application</b> A managed job template for restarting one or more system services. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Start-Application</b> A managed job template for starting one or more system services. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Stop-Application</b> A managed job template for stopping one or more system services. 환경: LINUX 최신 버전: 1.0	AWS 관리형
<b>AWS-Reboot</b> A managed job template for rebooting the device. 환경: LINUX 최신 버전: 1.0	AWS 관리형 ▲

# IoT Jobs : 작업 템플릿 사용(AWS-Reboot)

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```



# IoT Jobs : 작업 템플릿 (AWS-Download-File)


<https://catalog.us-east-1.prod.workshops.aws/workshops/6d30487a-48e1-4631-b6bc-5602582800b5/en-US/chapter5-jobs/20-dc-setup>

### Job template

Template type

AWS managed templates ▼

Template

AWS-Download-File AWS managed View 

A managed job template for downloading a file.  
Environments: LINUX Latest version: 1.0

AWS-Download-File version

1.0 - current ▼  
LINUX

downloadUrl

URL of file to download.

<https://deviceclient-workshopbucket.s3.amazonaws.com/AWSManagedJobsWelcomeF>

filePath

Path on the device where downloaded file is written.

/home/ubuntu/workshop\_dc/downloadLocation/firstFile.zip

### Job document

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

The End