

# AI&IOT 해커톤 Server 세팅

## Kakao i Cloud

1. 기본 콘솔창 입니다.

The screenshot displays the Kakao i Cloud Console interface. The top navigation bar includes the 'kakao i cloud' logo and a 'Console' tab. The left sidebar lists various service categories: COMPUTING (Virtual Machine, Image, Volume, Key pair), STORAGE (Object Storage, File Storage), NETWORKING (VPC, Topology, Router, Security Group, Public IP, Load Balancer), and MONITORING (Alarm Rule). The main content area is titled 'Summary' for the resource group 'music\_search'. It provides a high-level overview of resource usage across three categories: Compute, Storage, and Network. Below this, a table lists individual resources with columns for resource type, status, usage, and other details. The resources listed include SERVER (2), SERVER\_IMAGE (1), BLOCK\_STORAGE\_SSD (3), OBJECT\_STORAGE\_DATA (2), FLOATING\_IP (3), and LOAD\_BALANCER\_SW (1).

리소스 이름	상태	사용량기준	기준요금	단위	이용시간
> SERVER (2)					
> SERVER_IMAGE (1)					
> BLOCK_STORAGE_SSD (3)					
> OBJECT_STORAGE_DATA (2)					
> FLOATING_IP (3)					
> LOAD_BALANCER_SW (1)					

2. 공공기관 포트 정책에 의해 네트워크 설정을 먼저 한 후 Bastion이라는 VM을 설치하고 프록시 우회를 해야한다.

## 인터넷 포트 차단 정책 (inbound)

침입 차단 시스템을 이용하여 주요 서비스 포트를 차단합니다. 아래의 차단 포트 목록 이외 추가 차단이 필요한 포트는 카카오 i클라우드[공공기관용]의 기본 제공 기능인 [Security Group](#)을 통해 고객이 직접 설정하실 수 있습니다.

서비스명	차단 포트	차단방법	비고
시스템 터미널	TCP / 22, 23, 3389	네트워크 ACL 차단	외부에서 접속 불가
DBMS	TCP, UDP / 1433, 1521, 3306	네트워크 ACL 차단	외부에서 접속 불가
SMB	TCP, UDP / 135, 137, 138 , 139, 445	네트워크 ACL 차단	외부에서 접속 불가
기타	TCP / 21, TCP / 5900, ICMP	네트워크 ACL 차단	외부에서 접속 불가

- 우선, Network 생성하고, Router를 요청해서 할당받으면 아래와 같이 Router에 Public IP를 할당받아 해당 IP로 외부 네트워크에서 접속이 가능하다.

The screenshot shows the Kakao iCloud Console interface. On the left, there is a navigation menu with categories like COMPUTING, STORAGE, and NETWORKING. The 'NETWORKING' section is expanded, showing 'VPC' and 'Network'. The main content area displays the 'VPC' configuration page for a specific network. It includes a search bar and a 'Networks' button. Below this, there is a table with the following columns: 이름 (Name), 서브넷 (Subnet), 네트워크 주소 (Network Address), 공유 (Share), 외부 (External), 관리상태 (Management Status), and 상태 (Status). The table contains two entries: 'SB' with subnet 'SB-SUB' and IP '10.10.10.0/24', and 'gist\_software' with subnet 'gist\_software\_subnet' and IP '192.168.0.0/24'. Both entries show 'UP' for the external status and 'ACTIVE' for the management status.

music\_search

gist

Summary

COMPUTING

Virtual Machine

Image

Volume

Key pair

STORAGE

Object Storage

File Storage

NETWORKING

VPC

Topology

Router

Security Group

Public IP

Load Balancer

MONITORING

Alarm Rule

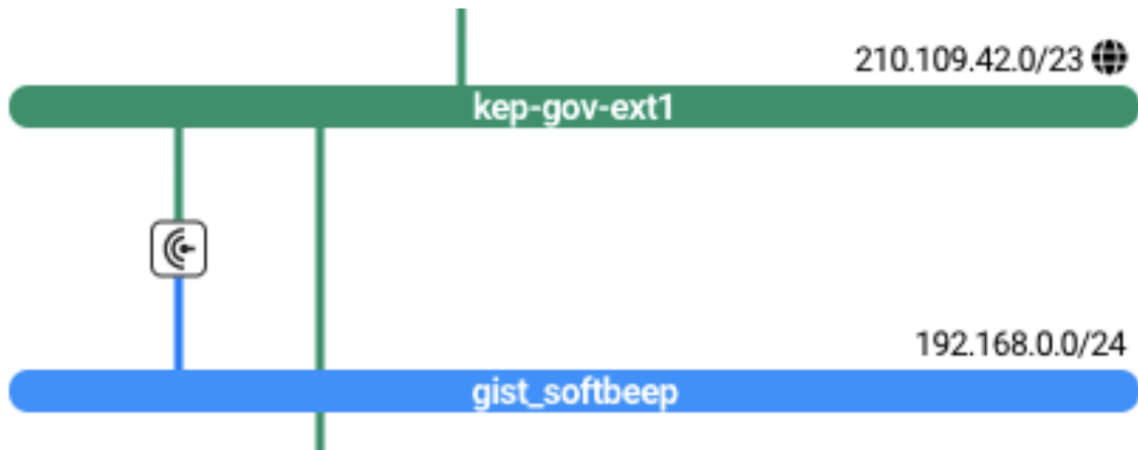
Alarm

Kakao Enterprise © 2022

Router

gist > music\_search > Networking > Router

<input type="checkbox"/>	이름	가용 구역	외부 네트워크	관리 상태	상태
<input type="checkbox"/>	gist_softbeep_tr	nova	kep-gov-ext1	UP	ACTIVE
<input type="checkbox"/>	SB	nova	kep-gov-ext1	UP	ACTIVE
<input type="checkbox"/>	SB2	nova	kep-gov-ext1	UP	ACTIVE
<input type="checkbox"/>	SB2	nova	kep-gov-access1	UP	ACTIVE



4. KeyPair생성하여 SSH 접속할 때 암호화를 설정하고, 해당 키를 설정하면, .pem이라는 ssh접속할 때 키가 발급이 된다.

gist\_softbeep\_key

gist > music\_search > Computing > Key pair > gist\_softbeep\_key

세부 정보

키페어 이름

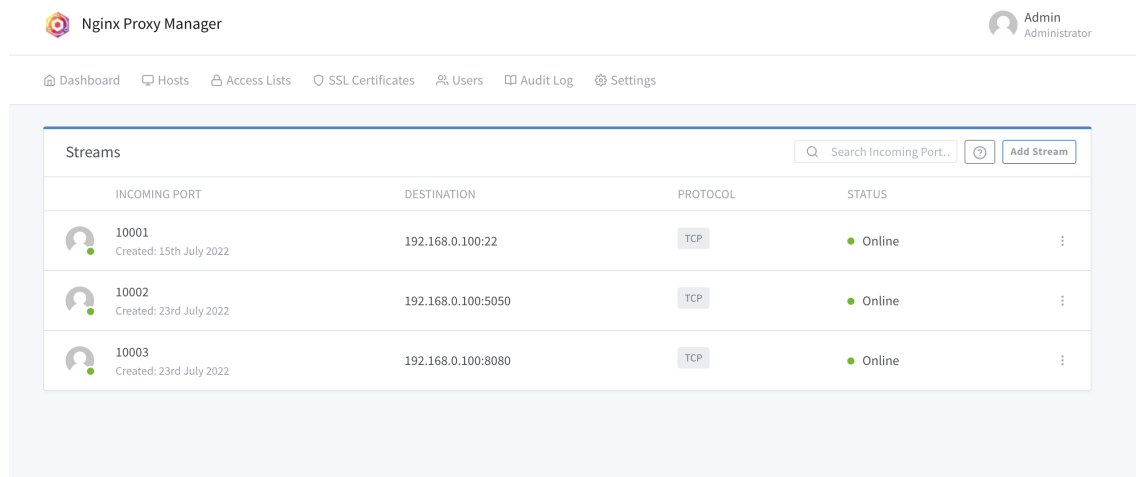
키페어 ID

Fingerprint

공개키

5. 프록시 우회를 위한 Bastion Vm을 생성하고, Bastion에서 특정 포트들을 포트포워딩 해줍니다.

- 외부 네트워크에서 10001포트로 요청 → TCP 포트 22번으로 열리고
- 외부 네트워크에서 10002포트로 요청 → 5050포트로 포워딩 된다. 이 포트는 Python Socket 통신을 위한 포트이다.
- 외부 네트워크에서 10003포트로 요청 → 8080포트로 포워딩 된다. 이 포트는 Spring에 접근하기 위한 포트이다.




Streams			
INCOMING PORT	DESTINATION	PROTOCOL	STATUS
10001 Created: 15th July 2022	192.168.0.100:22	TCP	Online
10002 Created: 23rd July 2022	192.168.0.100:5050	TCP	Online
10003 Created: 23rd July 2022	192.168.0.100:8080	TCP	Online

6. 이제 softveep\_vm을 설정하면 아래와 같이 생성이 됩니다.

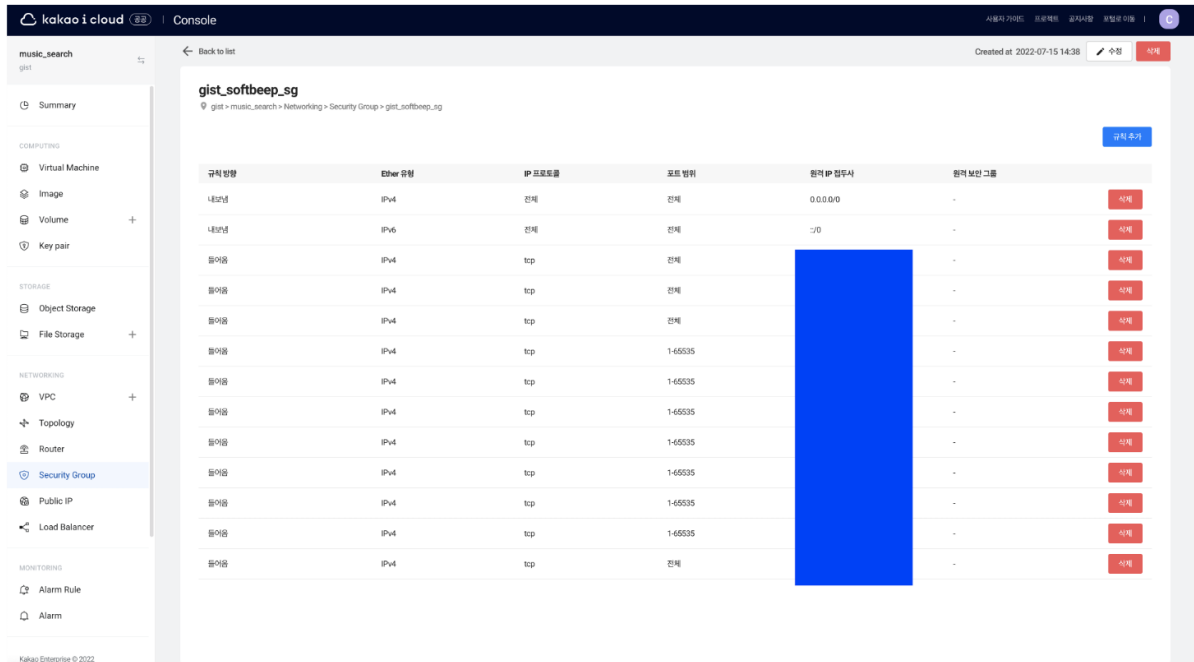
## Virtual Machine

gist > music\_search > Computing > Virtual Machine

목록은 10초 마다 자동 갱신됩니다.

<input type="checkbox"/>	이름	Private IP	Public IP
<input type="checkbox"/>	softbeep_vm		- 
<input type="checkbox"/>	gist_softbeep_bastion		

7. Security Group에서 특정 IP의 요청만 받을 수 있도록 설정해줍니다.



## GPS Module Setting

```

× softbeep@raspberrypi: ~/Softbeep (ssh)

Time:      2022-07-23T02:07:40.000Z (0)
Latitude:   35.23023167 N
Longitude:  126.84080000 E
Alt (HAE, MSL): 251.640, 183.399 ft
Speed:      1.32 mph
Track (true, var): 356.4, -7.8 deg
Climb:      0.00 ft/min
Status:     3D FIX (26 secs)
Long Err (XDOP, EPX): 3.04, +/- 149 ft
Lat Err (YDOP, EPY): 1.93, +/- 95.2 ft
Alt Err (VDOP, EPV): 0.98, +/- 74.0 ft
2D Err (HDOP, CEP): 1.64, +/- 102 ft
3D Err (PDOP, SEP): 1.91, +/- 119 ft
Time Err (TDOP): 4.60
Geo Err (GDOP): 7.89
ECEF X, VX: n/a n/a
ECEF Y, VY: n/a n/a
ECEF Z, VZ: n/a n/a
Speed Err (EPS): +/- 204 mph
Track Err (EPD): n/a
Time offset: 0.508467104 s
Grid Square: PM35kf05

Seen 13/Used 5
GNSS PRN Elev Azim SNR Use
GP 3 3 38.0 47.0 14.0 Y
GP 4 4 20.0 99.0 17.0 Y
GP 9 9 18.0 135.0 19.0 Y
GP 14 14 38.0 190.0 15.0 Y
GP 17 17 75.0 41.0 16.0 Y
GP 1 1 13.0 64.0 0.0 N
GP 2 2 4.0 266.0 0.0 N
GP 6 6 45.0 285.0 0.0 N
GP 11 11 14.0 262.0 0.0 N
GP 12 12 3.0 326.0 0.0 N
GP 19 19 59.0 332.0 0.0 N
GP 20 20 2.0 220.0 0.0 N
QZ 1 193 0.0 0.0 0.0 N
  
```



- Class 파일로 받아와서 위도 경도 데이터만 추출하여 Python Socket Programming에 보냈다.
- Client에서 보내는 코드

```
softbeep@raspberrypi: ~/Softbeep (ssh)
import socket
from gps import *
import time

SERVER_IP = "210.109.43.14"
SERVER_PORT = 10002
SERVER_ADDR = (SERVER_IP, SERVER_PORT)

gpsd = gps(mode=WATCH_ENABLE|WATCH_NEWSTYLE)

while True:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
        client_socket.connect(SERVER_ADDR)
        report = gpsd.next()
        if report['class'] == 'TPV':
            gps_data = str(getattr(report, 'lat', 0.0))+" "+str(getattr(report, 'lon', 0.0))
            print(gps_data)
            client_socket.send(gps_data.encode())
            time.sleep(1)
    ~
    ~
    ~
```

- Server에서 수신하는 코드

```

ubuntu@softbeep-vm: ~ (ssh)
import socket
import json

# 통신 정보 설정
IP = '192.168.0.100'
PORT = 5050
SIZE = 1024
ADDR = (IP, PORT)

# 서버 소켓 설정
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_socket.bind(ADDR)
    server_socket.listen()

# 무한루프 진입
while True:
    client_socket, client_addr = server_socket.accept()
    msg = client_socket.recv(SIZE)
    data = str(msg.decode()).split()
    print(data)
    if len(data) == 0:
        continue

    file_path_lat = "/home/ubuntu/lat.txt"
    file_path_lon = "/home/ubuntu/lon.txt"
    with open(file_path_lat, 'w', encoding='utf-8') as file:
        file.write(data[0])
    with open(file_path_lon, 'w', encoding='utf-8') as file:
        file.write(data[1])

~
~

```

```
ubuntu@softbeep-vm:~$ python3 server.py
```

```

[]
[]
[]
['35.230198333', '126.84028']
[]
['35.230198333', '126.84028']
['35.2302', '126.840278333']
['35.2302', '126.840278333']
['35.230198333', '126.840276667']
['35.230198333', '126.840276667']
['35.23019', '126.840273333']
['35.23019', '126.840273333']
['35.230191667', '126.84028']
['35.230191667', '126.84028']
['35.2302', '126.84027']
[]
['35.2302', '126.84027']
['35.230203333', '126.840251667']
['35.230203333', '126.840251667']
['35.230256667', '126.840286667']
['35.230256667', '126.840286667']
['35.23024', '126.840358333']
['35.23024', '126.840358333']
['35.230233333', '126.840355']
['35.230233333', '126.840355']
['35.230231667', '126.840336667']
[]
['35.230231667', '126.840336667']
['35.23023', '126.8403']
['35.23023', '126.8403']
['35.230226667', '126.840283333']
['35.230226667', '126.840283333']
['35.230223333', '126.840265']
['35.230223333', '126.840265']
[]

```

```
softbeep@raspberrypi:~/Softbeep $ python3 softbeep_gps_send_json.py
```

```

35.230198333 126.84028
35.230198333 126.84028
35.2302 126.840278333
35.2302 126.840278333
35.230198333 126.840276667
35.230198333 126.840276667
35.23019 126.840273333
35.23019 126.840273333
35.230191667 126.84028
35.230191667 126.84028
35.2302 126.84027
35.2302 126.84027
35.230203333 126.840251667
35.230203333 126.840251667
35.230256667 126.840286667
35.230256667 126.840286667
35.23024 126.840358333
35.23024 126.840358333
35.230233333 126.840355
35.230233333 126.840355
35.230231667 126.840336667
35.230231667 126.840336667
35.23023 126.8403
35.23023 126.8403
35.230226667 126.840283333
35.230226667 126.840283333
35.230223333 126.840265
35.230223333 126.840265

```

- 왼쪽이 VM
- 오른쪽이 RPI

## Spring Setting

- json 파일을 Streaming 형태로 리액트에 전달하기 위해
- Spring Web-Flux 프레임워크를 사용하여 구현하였다.
- python에서 보낸 코드를 읽어 GET mapping으로 오는 요청에 대해 Content type : application/json 형식으로 짜주는 것을 구현하였다.

The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Displays the project structure for 'spring-webflux-tutorial [webflux]'. It includes folders like 'src', 'main', 'resources', and 'test'. The 'main' folder contains 'java' and 'resources'. The 'java' folder contains 'com', 'devkuma', and 'webflux'. The 'webflux' folder contains 'HelloController', 'HelloService', 'HelloWebFluxApplication', 'server.py', and 'StreamingController.java'.
- Code Editor (Right):** Displays the code for 'StreamingController.java'. The code is as follows:
 

```

17
18 @RestController
19 public class StreamingController {
20
21     @Autowired
22     HelloService helloService;
23
24     @GetMapping(produces = TEXT_EVENT_STREAM_VALUE)
25     Flux<Data> sse(@RequestParam(value = "fail", required = false, defaultValue = "false") boolean fail) {
26         return source(fail);
27     }
28
29     @GetMapping(produces = APPLICATION_NDJSON_VALUE)
30     Flux<Data> ndjson(@RequestParam(value = "fail", required = false, defaultValue = "false") boolean fail) {
31         return source(fail);
32     }
33
34     @GetMapping
35     Flux<Data> array(@RequestParam(value = "fail", required = false, defaultValue = "false") boolean fail) {
36         return source(fail);
37     }
38
39     Flux<Data> source(boolean fail) { return fail ? failing() : successful(); }
40
41     Flux<Data> failing() {
42         return successful()
43             .concatWith(Mono.error(new RuntimeException("Oops!")));
44     }
45
46     private Flux<Data> successful() {
47         try {
48             Runtime.getRuntime().exec("python3 /home/ubuntu/server.py");
49         } catch (IOException e) {
50             e.printStackTrace();
51         }
52         return Flux.interval(Duration.ofSeconds(1))
53             .map(i -> new Data(helloService.getLat(), helloService.getLon()));
54     }
55
56     class Data {
57         private final String lat;
58     }
59
60     }
61

```

Server거치고 HTTP GET 요청으로 json 데이터가 받아지는 모습을 보여주는 ppt



- 왼쪽 : RPI, GPS를 싸줍니다.
- 중앙 : Cloud Server, Spring이 서비스 로직을 처리하고 response해줍니다.
- 오른쪽 : Client, 후에 리액트가 Response를 받아 사용자에게 View를 제공해줍니다.

The image displays three terminal windows illustrating the deployment of a Spring Boot application:

- Left Terminal (Raspberry Pi):** Shows the execution of a Java command to run the application. The output lists a series of coordinates (latitude and longitude) for various locations, such as 35.230026667, 126.840386667, 35.230028333, 126.840386667, etc.
- Middle Terminal (Cloud Server):** Shows the execution of the command `java -jar webflux-0.0.1-SNAPSHOT.jar`. The output displays the Spring Boot logo and version information (v2.4.4), followed by logs indicating the application is starting successfully on port 8080.
- Right Terminal (Client):** Shows the execution of a curl command to send an HTTP request to the application. The output displays a series of JSON responses, each containing latitude and longitude coordinates, such as `{\"lat\": \"35.230026667\", \"lon\": \"126.840386667\"}`.