

# **Week 8. Final.**

# **Server, Backend and Deployment**

- Server, that is just program
- deploy.. just opening port.

**Review.. Very Important**

# File I/O

Java I / O (Input / Output)는, java.io 패키지에 있다.

```
1 import java.io.File;  
2 import java.io.FileReader;  
3 import java.io.FileWriter;  
4 import java.io.BufferedReader;  
5 import java.io.BufferedWriter;
```

이번 시간에 우리는, Image 같은 바이트 단위로 읽는 것이 아닌 문자 단위로, Reading을 할 것이다.

```
1 BufferedImage image = ImageIO.read(new File("images/input.png"));  
2 ImageIO.write(image, "png", new File("images/output.png"));
```

# Absolute Path / Relative Path

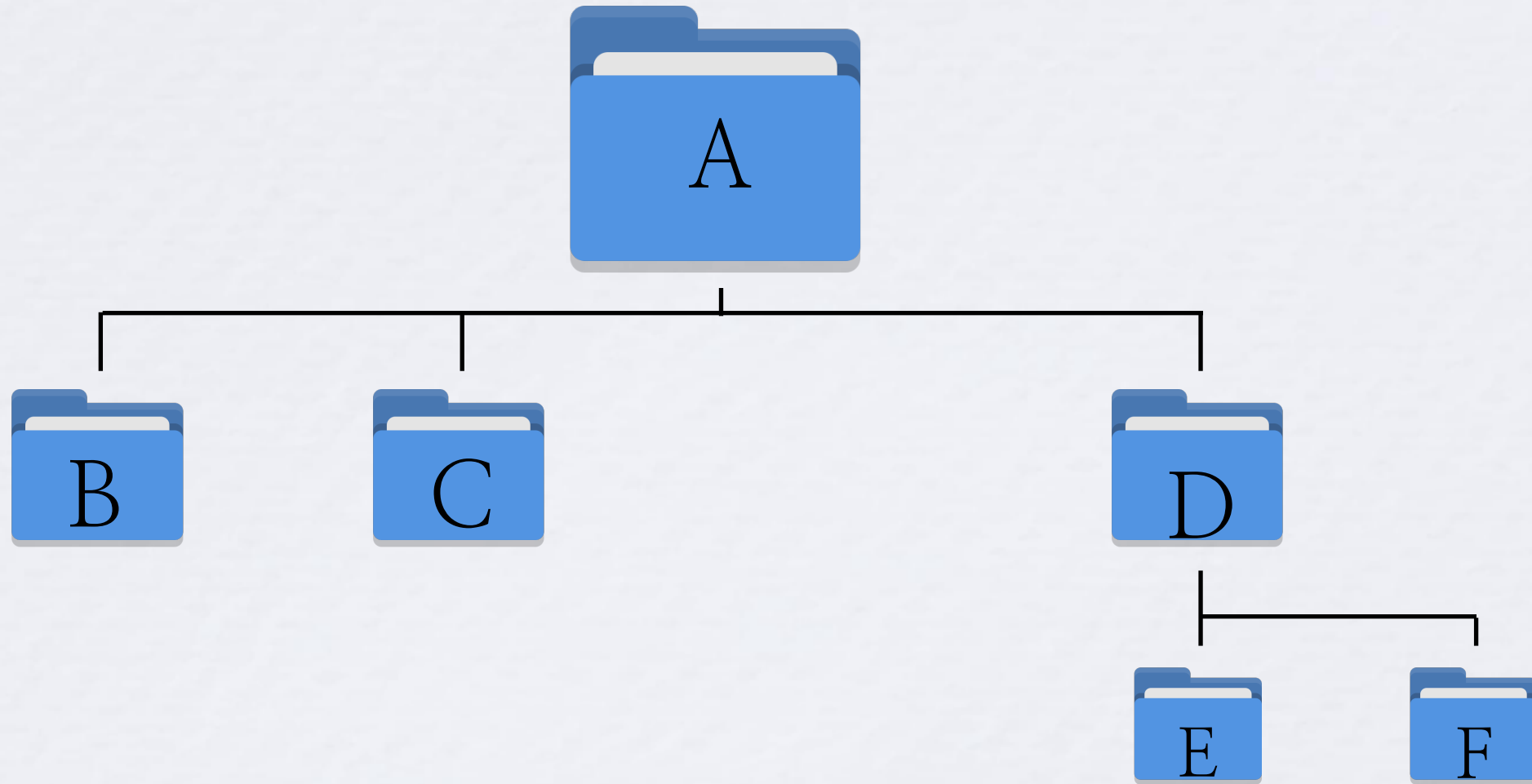
폴더 안에는 자신을 가리키는 폴더가 존재합니다.

또한, 부모를 가리키는 폴더가 존재합니다.

```
~/s/class_programing3 ls -al
```

```
> ls -al
total 456
drwxr-xr-x  20 hyeonseok staff   640  5 22 16:18 .
drwxr-xr-x  13 hyeonseok staff   416  5  6 17:18 ..
-rw-r--r--@  1 hyeonseok staff 10244  5 16 03:46 .DS_Store
-rwxr-xr-x   1 hyeonseok staff 33432  3  7 11:17 helloworld
-rw-r--r--   1 hyeonseok staff   275  5 26 01:43 main.java
-rwxr-xr-x   1 hyeonseok staff 83560  4  2 17:07 suffix_tree
```

# Absolute Path / Relative Path



.. 은 부모 폴더로 이동하는 것이고, .은 현재 폴더로 이동하는 것이다.

.. 을 자유롭게 쓰면, 원하는 폴더로 상대경로로 이동할 수 있다.

# Mission..

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         for (int i=1;i<=10;i++) {  
4             BufferedWriter bw = new BufferedWriter(new FileWriter("./outputSet/" + i + ".txt"));  
5  
6             bw.write(i);  
7             bw.close();  
8         }  
9     }  
10 }
```

practice ~/IdeaProjects  
    > .idea  
    > out  
    outputSet

ain.java    outtest.txt x

SOH  
STX  
ETX  
EOT  
ENQ  
ACK  
BEL  
BS

outputSet  
    1.txt  
    2.txt  
    3.txt  
    4.txt  
    5.txt  
    6.txt  
    7.txt  
    8.txt  
    9.txt  
    10.txt

bw.write()에 숫자를 적었는데 이게 뭐죠…?

# Appending Data to a File in Java

```
1 public class Main {  
2     public static void main(String[] args) throws IOException {  
3         BufferedWriter bw = new BufferedWriter(new FileWriter("test.txt", true));  
4     }  
5 }
```

파일이 자꾸 덮어쓰기가 되는데, 이어쓰기는 어떻게 하나요..?

FileWriter의 매개변수 뒤에, true를 하여, append 기능을 활성화할 수 있습니다.



# Serialization & Deserialization

직렬화 (Serialization) : 객체를 파일 등에 저장할 수 있도록 byte 형태로 변환 하는 과정

역직렬화(Deserialization) : 저장된 byte 데이터를 다시 객체로 복원하는 과정

```
1  class Person implements Serializable{
2      public String name;
3      public int age;
4
5      public Person(String name, int age) {
6          this.name = name;
7          this.age = age;
8      }
9  }
```



# Serialization & Deserialization

```
1 List<Person> people = new ArrayList<>();
2 people.add(new Person("Alice", 25));
3 people.add(new Person("Bob", 30));
4 people.add(new Person("Charlie", 28));
5
6 ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("people.dat"));
7 oos.writeObject(people); // 리스트 전체를 직렬화
8 oos.close();
```

```
1 ObjectInputStream ois = new ObjectInputStream(new FileInputStream("people.dat"));
2 List<Person> people = (List<Person>) ois.readObject();
3 ois.close();
```

리스트 자체를 직렬화 해버려서, 한꺼번에 몽땅 저장 할 수 있습니다.

# Table of Contents

1. Server란 무엇인가?
2. backend란 무엇인가?
3. 간단하게 띄어보는 API 통신 서버
  - A. localhost..? port..?
4. DeployMent
  - A. 공유기 Router
  - B. 공인 IP 와 사설 IP
  - C. PortFoward
  - D. DDNS Dynamic DNS
5. Backend FrameWorks

# What is Server..?

## \* 서버란?

서버는 다른 컴퓨터나 장치(클라이언트)에게 필요한 정보를 제공하는 컴퓨터이다.

항상 켜져 있고, 클라이언트의 요청을 기다리는 컴퓨터입니다.

해당 컴퓨터는 요청을 받으면, 응답을 합니다. 이것이 서버의 전부입니다.

클라이언트는 그럼 Server에 어떻게 요청을 하는 것일까요?

# What is Server..?

## \* 요청 (Request/Fetch)

클라이언트에서, 서버에 인터넷을 통해 어떠한 정보를 보내는 것

특정 IP로 데이터를 요청한다. 인터넷을 타고 패킷(데이터 조각) 형태로 전달된다.

Http (HyperText Transfer Protocol) (가장 흔한 방식) :

- 웹사이트 접속할 때 쓰는 방식이다.
- 브라우저가 Get, Post, Put, Delete 같은 요청을 서버에 보내고, 서버는 페이지나 데이터를 응답한다.
- 클라이언트가 요청해야 서버가 응답하는 구조이다.

# What is Server..?

## \* 요청 (Request/Fetch)

소켓 통신 (Socket):

실시간 게임이나 채팅처럼, 계속 연결된 상태에서 서로 데이터를 주고 받는 방식

연결을 한 번 맺으면, 그 후로는 계속 주고받을 수 있다.

TCP/IP 소켓, WebSocket 이라는 것이 있다.

ex ) 유저 A, B, C가 소켓 서버에 연결함

유저 A가 “안녕!” 메시지를 보냄 -> B, C에게 브로드캐스트

이때, B, C는 요청을 하지도 않았는데, 서버에서 응답을 받음

# What is \*Backend..?

시스템의 요구사항에 따라 HTTP 통신 방식 또는 소켓 통신 방식을 활용한 서버를 설계, 구축, 구현 및 운영관리를 하는 직무

반면.. \*FrontEnd 는 그럼 뭔가요?

사용자가 실제로 보고, 클릭하고, 입력하는 모든 화면을 만드는 영역입니다.

화면에 보이는 입력창, 로그인 상태에 따른 페이지의 구성, 애니메이션 및 팝업 요소 등을 구현하고, Backend 서버와의 통신 로직을 개발 및 유지보수 하는 직무

즉, 클라이언트가 Backend 서버로 요청을 하는 화면이나 UI를 구축합니다.



# Experience Backend with Http Server

이제 자바로 간단하게 Http통신을 받아들이고 응답하는 Server를 만들어 봅시다. (Java 16부터 나온 기능을 사용함)

```
1  class Handler implements HttpHandler {
2      @Override
3      public void handle(HttpExchange exchange) throws IOException {
4          String text = "안녕 너의 서버야.";
5          byte[] bytes = text.getBytes(StandardCharsets.UTF_8);
6
7          exchange.getResponseHeaders().add("Content-Type", "text/plain; charset=UTF-8");
8          exchange.sendResponseHeaders(200, bytes.length);
9          OutputStream os = exchange.getResponseBody();
10         os.write(bytes);
11         os.close();
12     }
13 }
14
15 public class Main {
16     public static void main(String[] args) throws IOException{
17         try {
18             HttpServer server = HttpServer.create(new InetSocketAddress(1231), 0);
19             server.createContext("/", new Handler());
20             server.setExecutor(null);
21             server.start();
22             System.out.println("Server Start! http://localhost:1231");
23         } catch (IOException e) {
24             e.printStackTrace();
25         }
26     }
27 }
```



# Experience Backend with Http Server

```
1  try {  
2      HttpServer server = HttpServer.create(new InetSocketAddress(1231), 0);  
3      server.createContext("/", new Handler());  
4      server.setExecutor(null);  
5      server.start();  
6      System.out.println("Server Start! http://localhost:1231");  
7  } catch (IOException e) {  
8      e.printStackTrace();  
9  }
```

InetSocketAddress(1231) 라는 객체로 HttpServer.create 매서드로 객체를 만듭니다.

해당 HttpServer 객체는 정말 서버 그 자체입니다.

# Experience Backend with Http Server

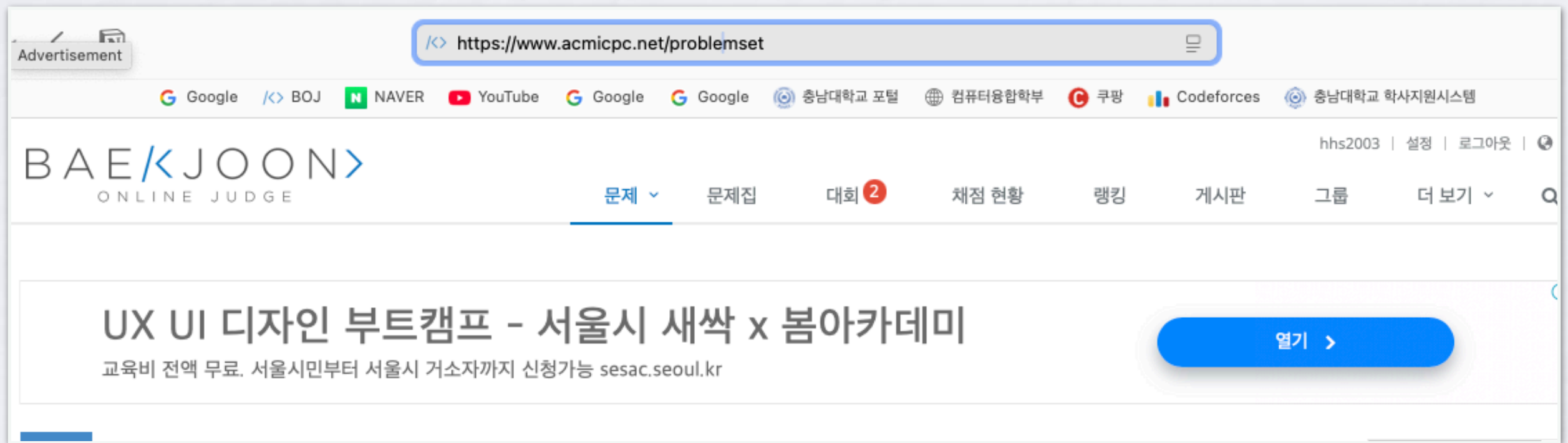
```
1  try {  
2      HttpServer server = HttpServer.create(new InetSocketAddress(1231), 0);  
3      server.createContext("/", new Handler());  
4      server.setExecutor(null);  
5      server.start();  
6      System.out.println("Server Start! http://localhost:1231");  
7  } catch (IOException e) {  
8      e.printStackTrace();  
9  }
```

createContext("/", new Handler()) :

“/” 라는 \*엔드포인트 (endpoint)에 요청할 경우, Handler() 클래스에 정의되어 있는 handle 함수를 실행시킵니다.

\*엔드포인트 : 클라이언트가 서버에게 요청을 보낼 수 있는 ‘주소 경로’입니다.

# Experience Backend with Http Server



백준 사이트의 “/problemset” 엔드포인트로 들어가면, 해당 서버에서는 그에 맞는 함수를 실행시키는 것입니다.

# Experience Backend with Http Server

```
1  try {  
2      HttpServer server = HttpServer.create(new InetSocketAddress(1231), 0);  
3      server.createContext("/", new Handler());  
4      server.setExecutor(null);  
5      server.start();  
6      System.out.println("Server Start! http://localhost:1231");  
7  } catch (IOException e) {  
8      e.printStackTrace();  
9  }
```

setExecutor(null):

서버가 클라이언트 요청을 처리할 “Thread full”을 어떻게 만  
들지 지정하는 메서드이다.

# Experience Backend with Http Server

setExecutor(null):

null로 설정 할 경우, Java 가 내부적으로 기본 Thread 풀을 자동으로 만들어서 사용한다.

```
1 server.setExecutor(Executors.newFixedThreadPool(10));
```

다음과 같이, 명시적으로, 10개의 스레드 풀을 고정 시킬 수도 있다.

좀 더 효율적인 병렬 처리를 하고 싶을 때, 직접 설정한다.

Java가 내부적으로 기본 스레드 풀을 자동으로 만들어서 사용한다.

- 요청이 들어올 때마다, 새 Thread 라는 것을 생성한다.

# Experience Backend with Http Server

setExecutor(null) :

요청마다 새 Thread를 만드는 것은 성능 저하로 이어질 수 있다.

```
1 server.setExecutor(Executors.newFixedThreadPool(10));
```

다음과 같이 쓰레드를 고정 시켜, 재사용 시키면, 안정된 서버 성능을 뽑아 낼 수 있다.



# Experience Backend with Http Server

setExecutor(null) :

요청마다 새 Thread를 만드는 것은 성능 저하로 이어질 수 있다.

```
1 server.setExecutor(Executors.newFixedThreadPool(10));
```

다음과 같이 쓰레드를 고정 시켜, 재사용 시키면, 안정된 서버 성능을 뽑아 낼 수 있다.



# Experience Backend with Http Server

```
1  try {  
2      HttpServer server = HttpServer.create(new InetSocketAddress(1231), 0);  
3      server.createContext("/", new Handler());  
4      server.setExecutor(null);  
5      server.start();  
6      System.out.println("Server Start! http://localhost:1231");  
7  } catch (IOException e) {  
8      e.printStackTrace();  
9  }
```

HttpServer.start() 하나의 쓰레드에서, 이제 포트 1231로 들어오는 요청을 기다린다.

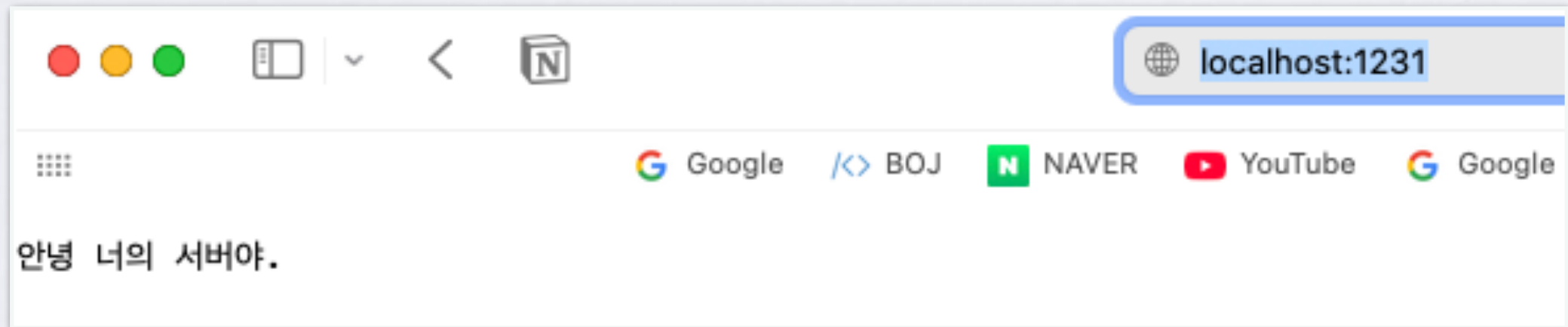
# Experience Backend with Http Server

```
1  class Handler implements Handler {
2      @Override
3      public void handle(HttpExchange exchange) throws IOException {
4          String text = "안녕 너의 서버야.";
5          byte[] bytes = text.getBytes(StandardCharsets.UTF_8);
6
7          exchange.getResponseHeaders().add("Content-Type", "text/plain; charset=UTF-8");
8          exchange.sendResponseHeaders(200, bytes.length);
9          OutputStream os = exchange.getResponseBody();
10         os.write(bytes);
11         os.close();
12     }
13 }
```

“/” 엔드포인트로 들어갈 시, 해당 Handler 클래스의 handle 함수가 실행된다.

“text/plain” format 형식으로, “안녕 너의 서버야.” 라는 응답을 준다.

# Experience Backend with Http Server



해당 자바 프로그램은 포트 1231로 들어오는 http 요청에 응답한다.

자신의 포트 요청을 날릴 때는 주소, localhost를 사용하면 된다.

http://localhost:1231/ 로 들어가면, handle 함수가 호출되며, 다음과 같은 응답을 전송한다

# Experience Backend with Http Server

Q. 이게 무슨 서버인가요? 서버는 데이터 저장도 하고, 로그인 인증도하고 그래야 하지 않나요?

A. 이게 서버의 전부입니다. 지금은 응답만 주지만 우리는 서버의 특정 함수를 실행시킬 수 있다는 것에 집중해야 합니다. 함수가 실행되면, 복잡한 로직연산을 시작 할 수 있겠죠?

Q. 네? 그럼 이게 서버의 전부고, 외부랑 연결만 해서 진짜 사용하는 건가요?

A. 네, 진짜 이게 서버의 전부고 열심히 코딩해서 필요한 로직들을 다 만들어놓고, 외부랑 연결만 하면 됩니다.

# Experience Backend with Http Server

Q. 포트는 뭔가요...? 왜 필요한거죠?

포트는 컴퓨터 안에서 실행 중인 여러 프로그램(서비스)를 구분해주는 논리적 번호입니다.

하나의 컴퓨터에 여러 프로그램 (서버) 가 실행 될 수 있습니다. ex) 웹 서버, 파일 서버, 게임 서버

모든 서버가 하나의 IP를 쓰면, 당연히 외부 요청이 어느 쪽 프로그램으로 들어가야 하는지 모르게 됩니다.

따라서, 각 프로그램(서버)이 자기만의 포트 번호를 갖고, 그걸로 통신합니다.

# Router...!!!!!!

Router (공유기) : 컴퓨터 네트워크 간의 데이터 패킷을 전송하는 네트워크 장치

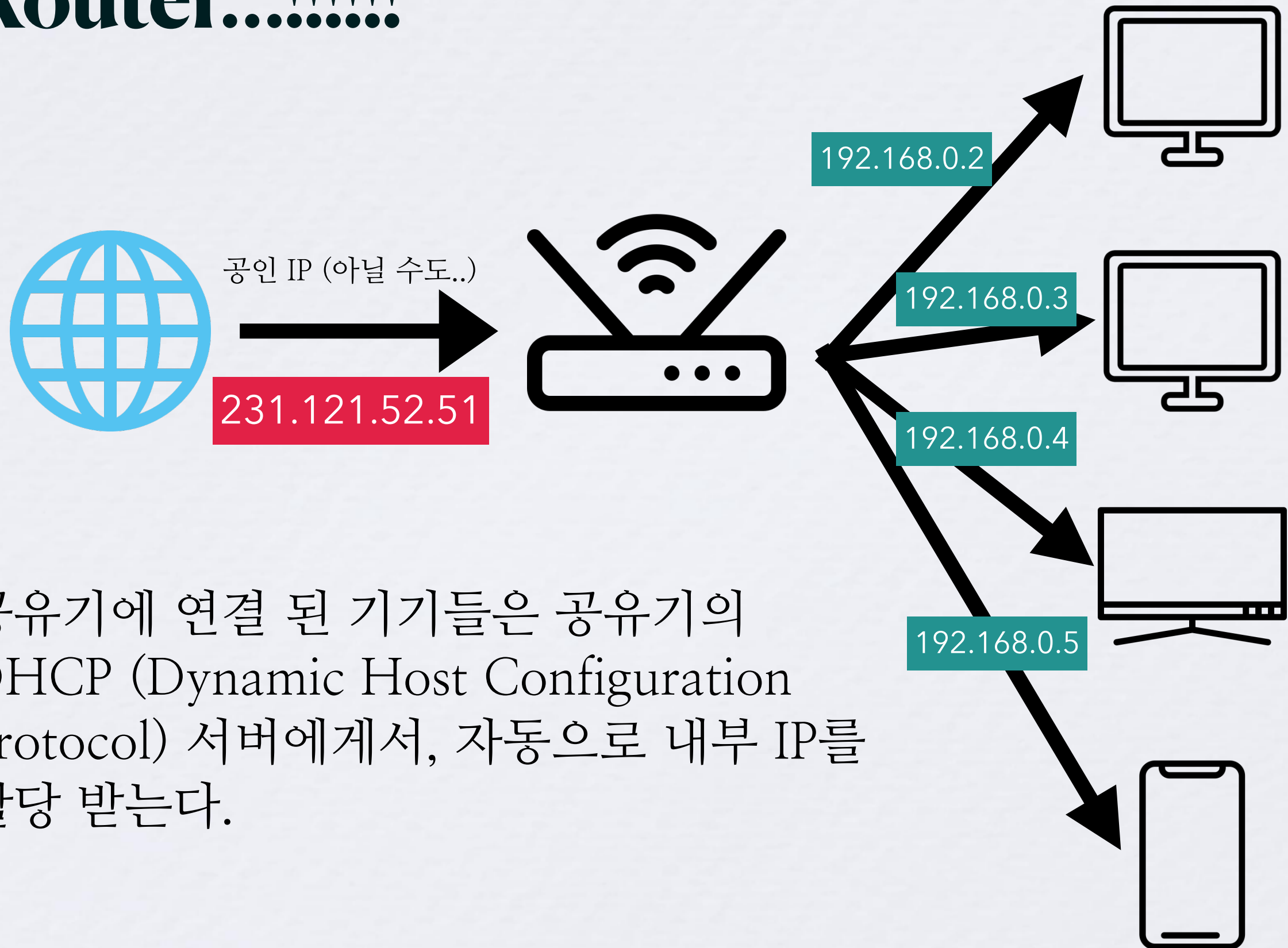
공유기는 인터넷 하나를 여러 기기들이 함께 쓸 수 있게 나눠주는 네트워크 장치입니다.

Wifi 많이 쓰시죠? Wifi를 사용하는 것은 인터넷에 연결하는 것이 아닙니다.

하나의 외부 인터넷에 연결된 공유기에 연결하는 것입니다.



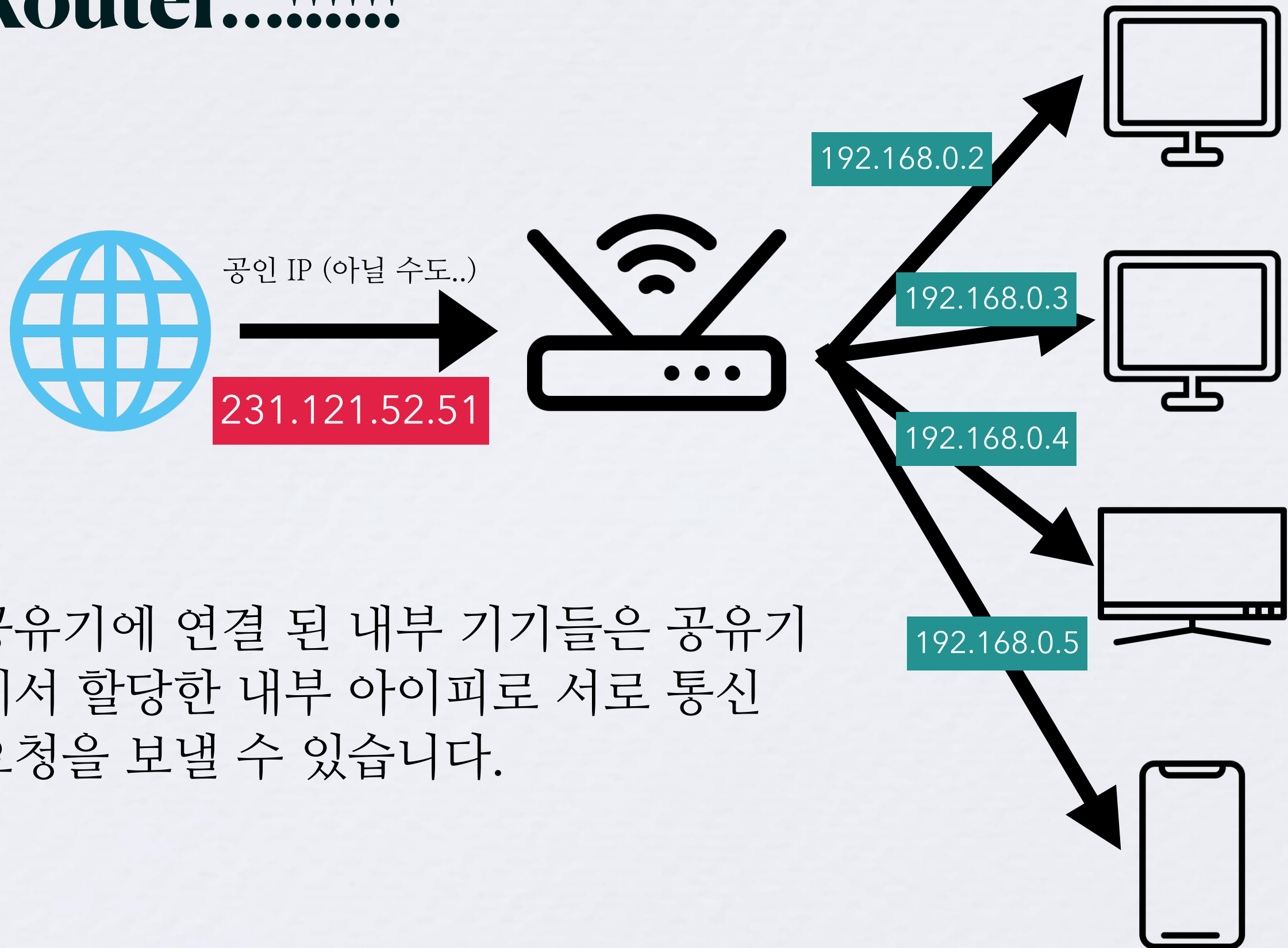
# Router...!!!!!!



공유기에 연결 된 기기들은 공유기의 DHCP (Dynamic Host Configuration Protocol) 서버에게서, 자동으로 내부 IP를 할당 받는다.

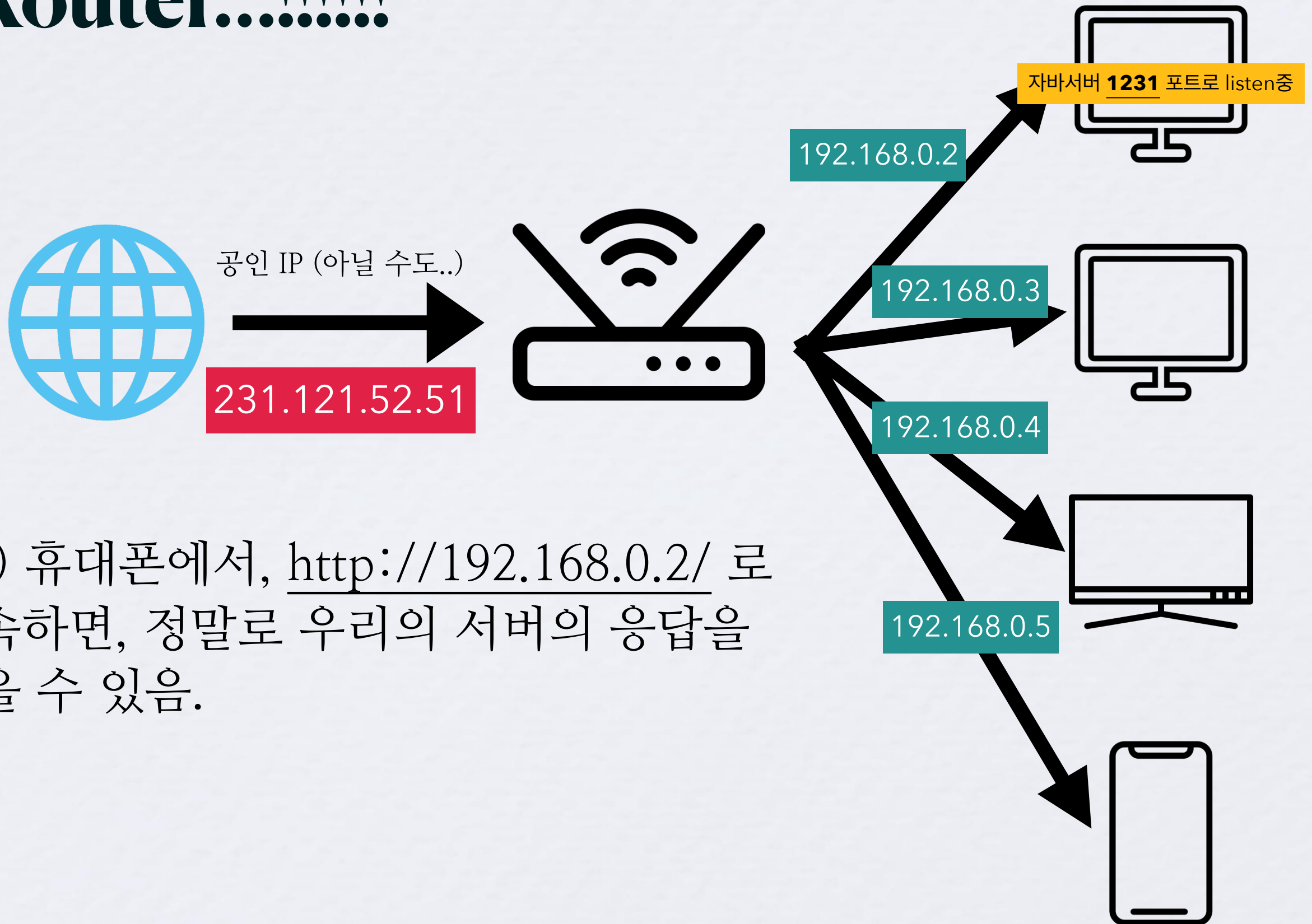


# Router.....!!!!!!



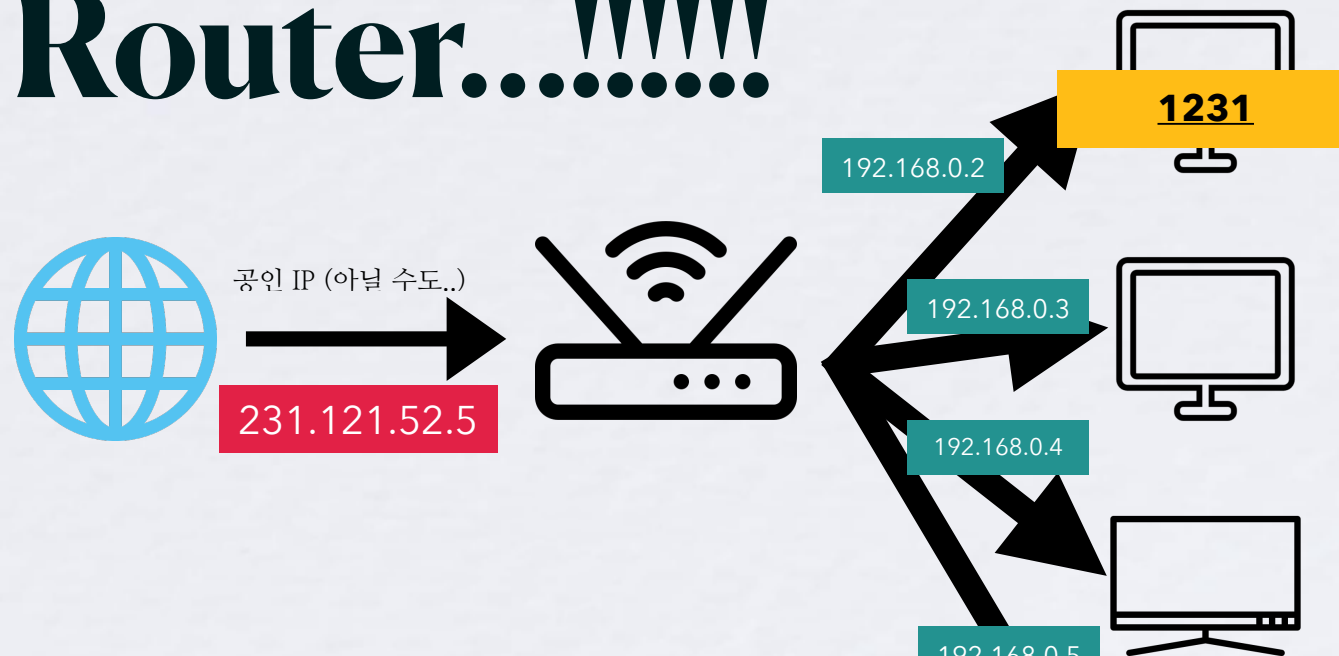
공유기에 연결 된 내부 기기들은 공유기  
에서 할당한 내부 아이피로 서로 통신  
요청을 보낼 수 있습니다.

# Router...!!!!!!



ex ) 휴대폰에서, <http://192.168.0.2/> 로 접속하면, 정말로 우리의 서버의 응답을 받을 수 있음.

# Router.....!!!!!!

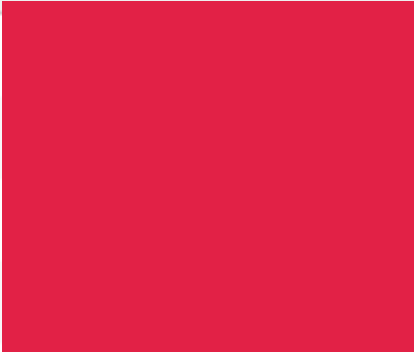


사용중인 IP 주소 정보 (4개사용중)		🔍 재검색 + 등록 <input type="checkbox"/>	
192.168.0.9		DESKTOP-IUV9H4H	무선:자동할당 <input type="checkbox"/>
192.168.0.12		MacBookAir	무선:자동할당 <input type="checkbox"/>
192.168.0.13		hwanghyuiiphone	무선:자동할당 <input type="checkbox"/>
192.168.0.15		- :수동설정	<input type="checkbox"/>

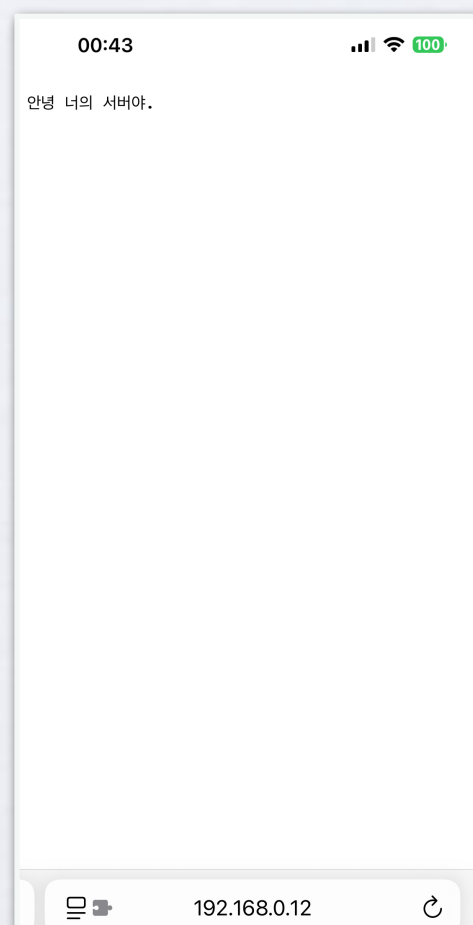
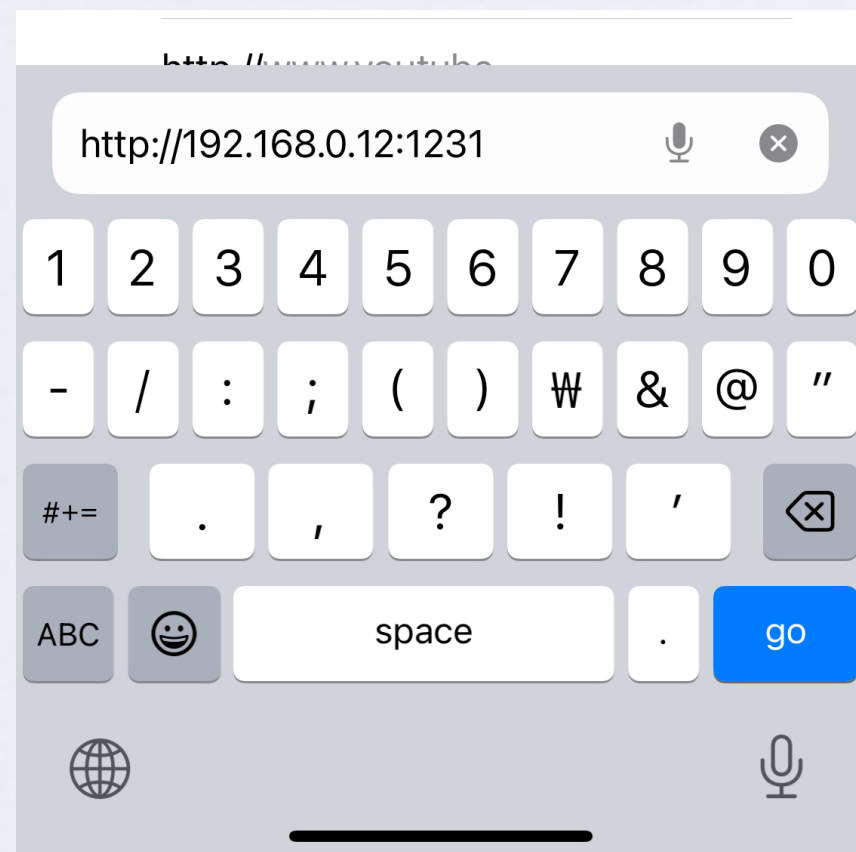
\*실제 : 예시.. 사용자는 iptime 공유기를 이용중

MacBookAir 192.168.0.12:1231에서 자바서버 실행중..

# Router..!!!!!!

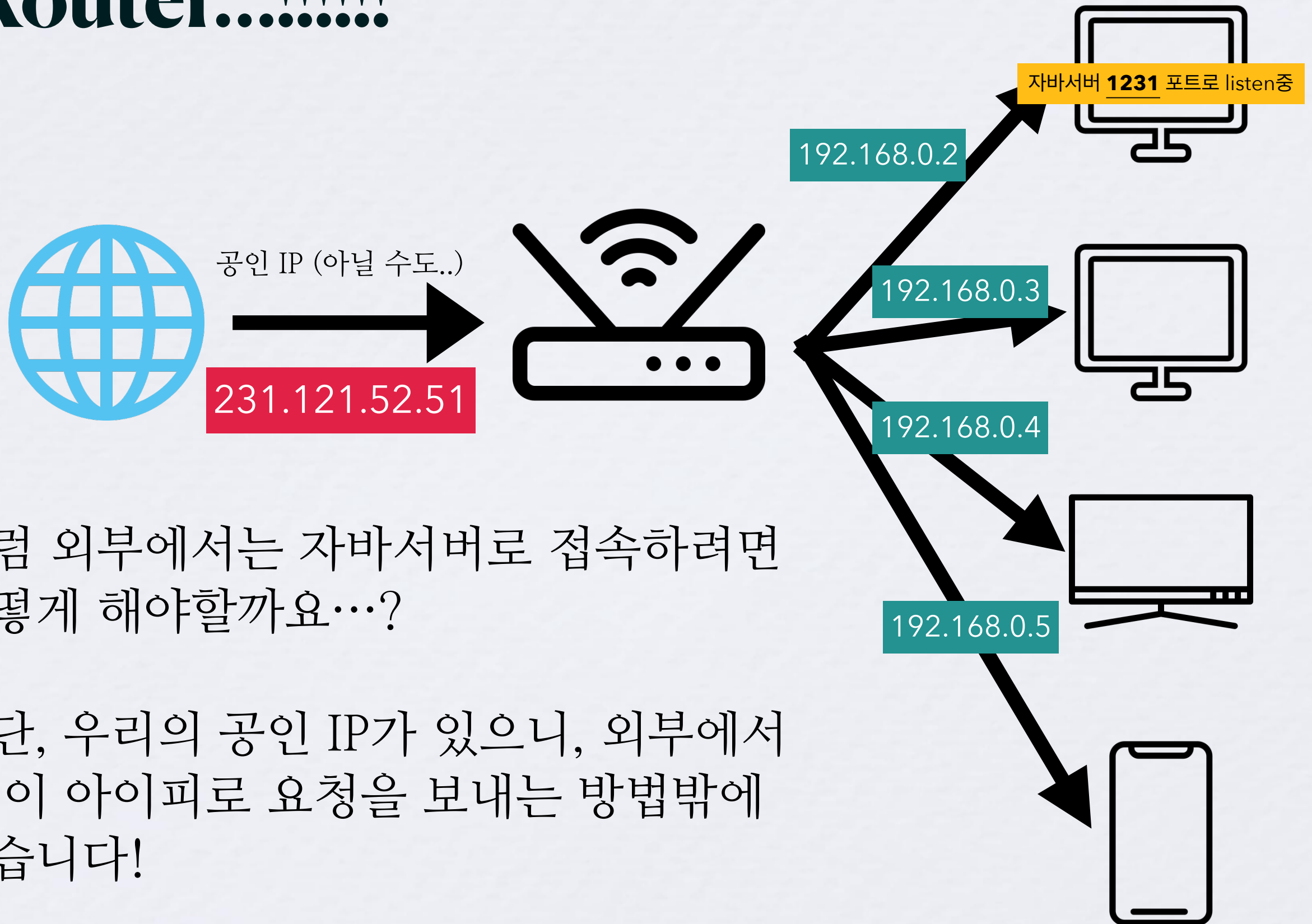
사용중인 IP 주소 정보 (4개사용중)		재검색	등록	
192.168.0.9		DESKTOP-IUV9H4H	무선:자동할당	<input type="checkbox"/>
192.168.0.12		MacBookAir	무선:자동할당	<input type="checkbox"/>
192.168.0.13		hwanghyuiiphone	무선:자동할당	<input type="checkbox"/>
192.168.0.15		- :수동설정		<input type="checkbox"/>

MacBookAir 192.168.0.12:1231에서 자바서버 실행중..



정말로 됩니다.

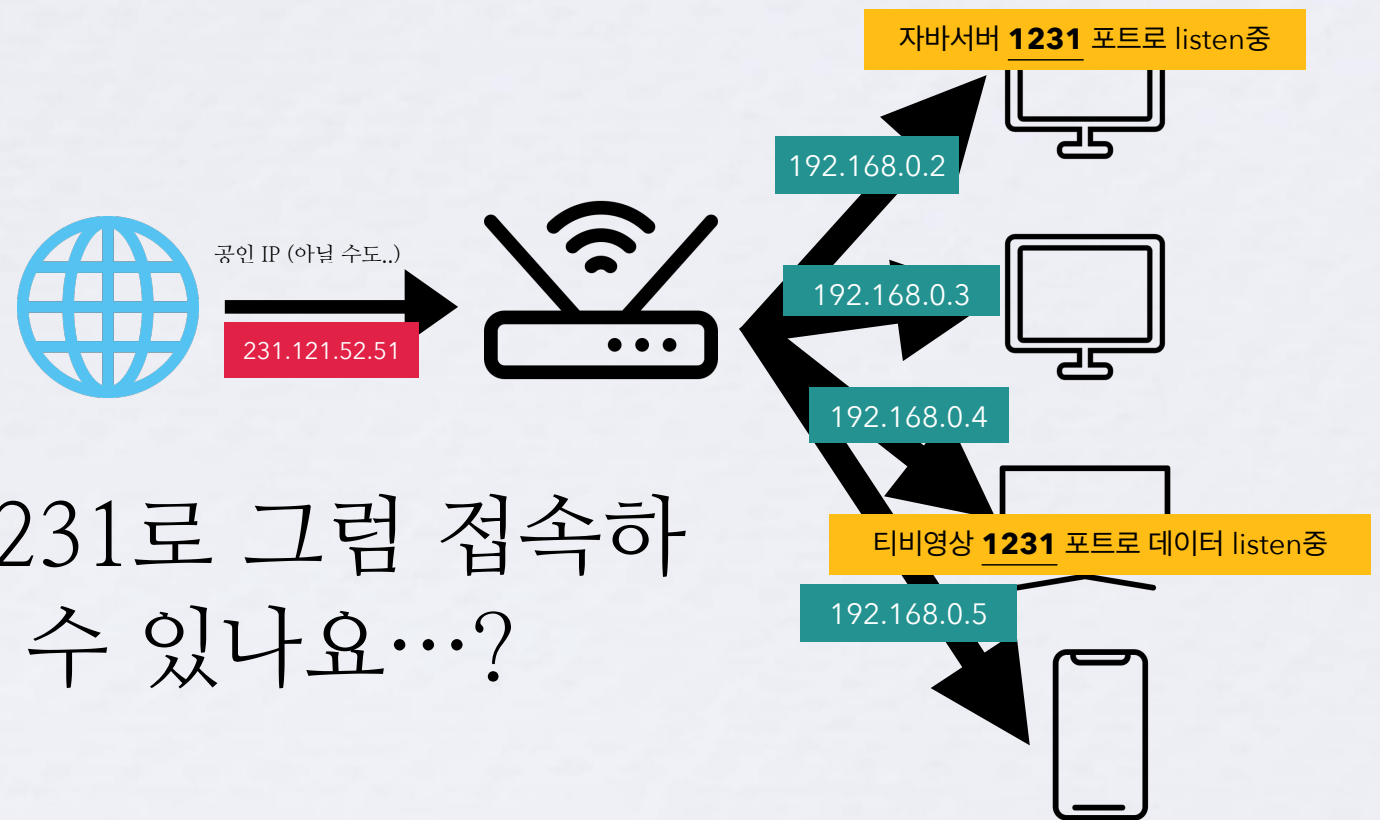
# Router...!!!!!!



그럼 외부에서는 자바서버로 접속하려면 어떻게 해야할까요...?

일단, 우리의 공인 IP가 있으니, 외부에서는 이 아이피로 요청을 보내는 방법밖에 없습니다!

# Router...!!!!!!



외부에서, 231.121.52.51:1231로 그림 접속하면, 자바서버로 요청을 보낼 수 있나요…?

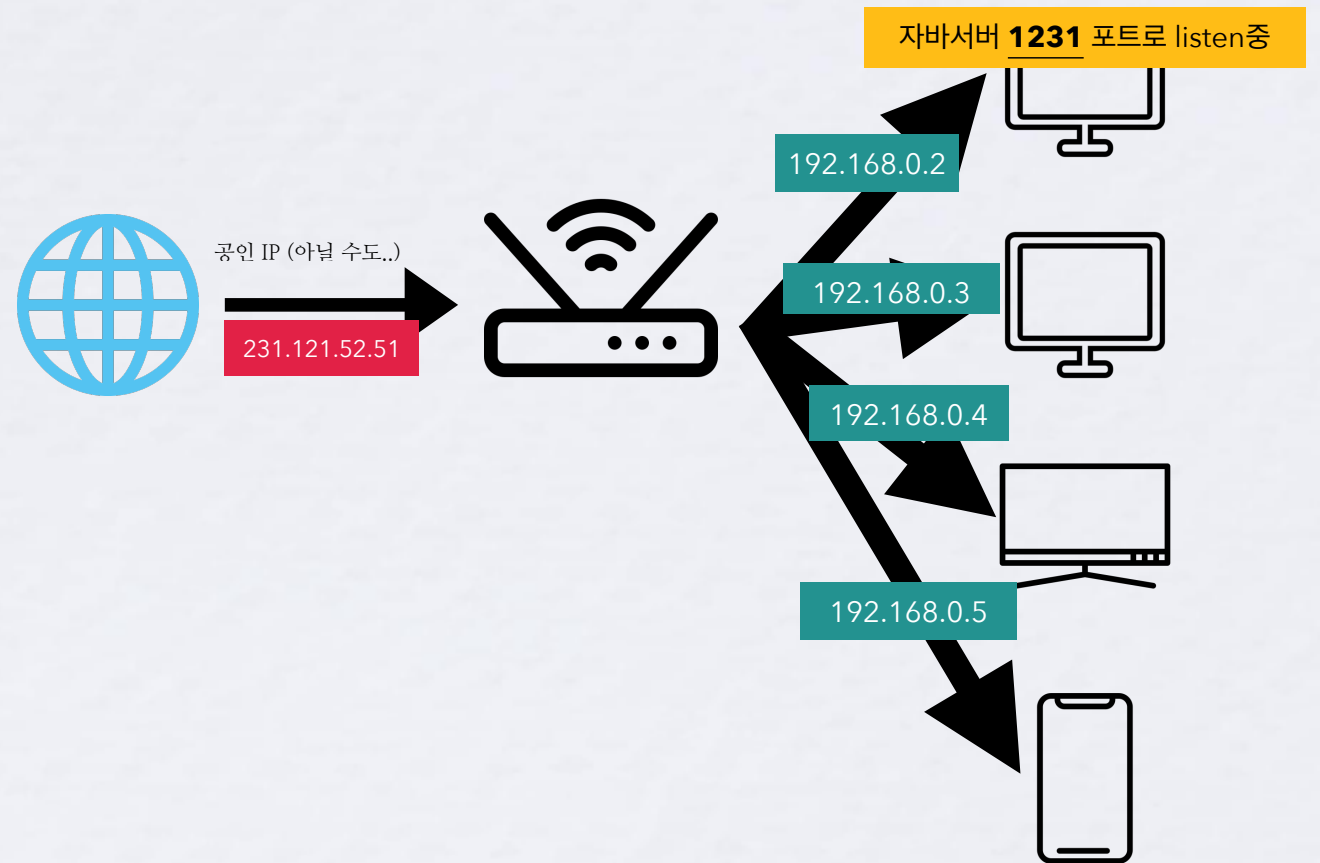
공유기는 지금 인터넷과 연결되어 있으므로, 정확히는 공유기에 먼저 요청을 하게 됩니다.

기본적으로 공유기는 모든 입력 포트가 닫혀져 있습니다.

따라서, 공유기의 1231포트로 입력이 들어오면, 192.168.0.2의 1231포트로 응답을 보내주어야 합니다.



# Router...!!!!!!



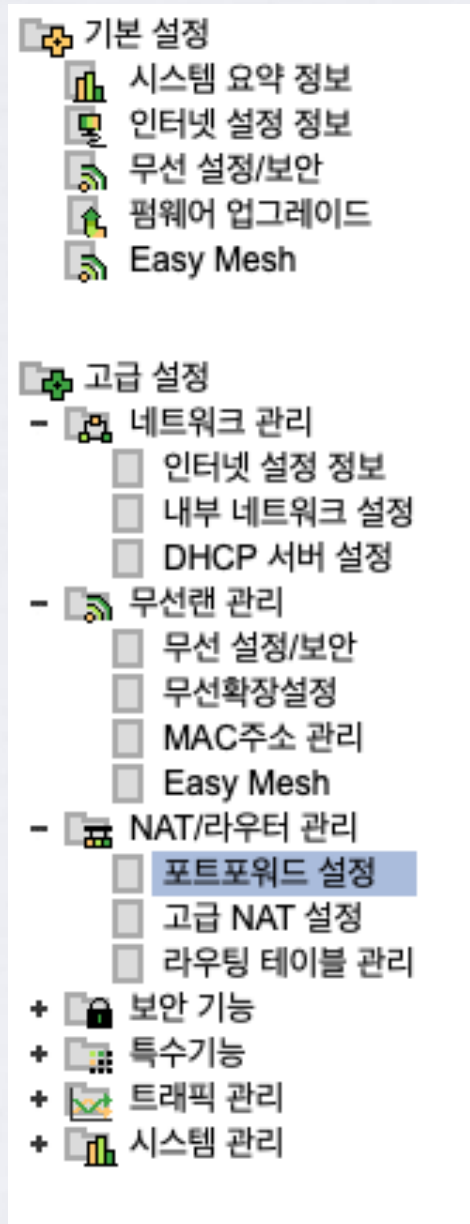
공유기의 1231포트로 입력이 들어오면, 192.168.0.2의 1231포트로 요청을 보내주어야 합니다.

공유기 들어가서 직접 설정해야하고, 포트를 이어주어야 합니다.

이 과정을, portForward(포트 전달) 설정, 포트포워딩 한다고 합니다.



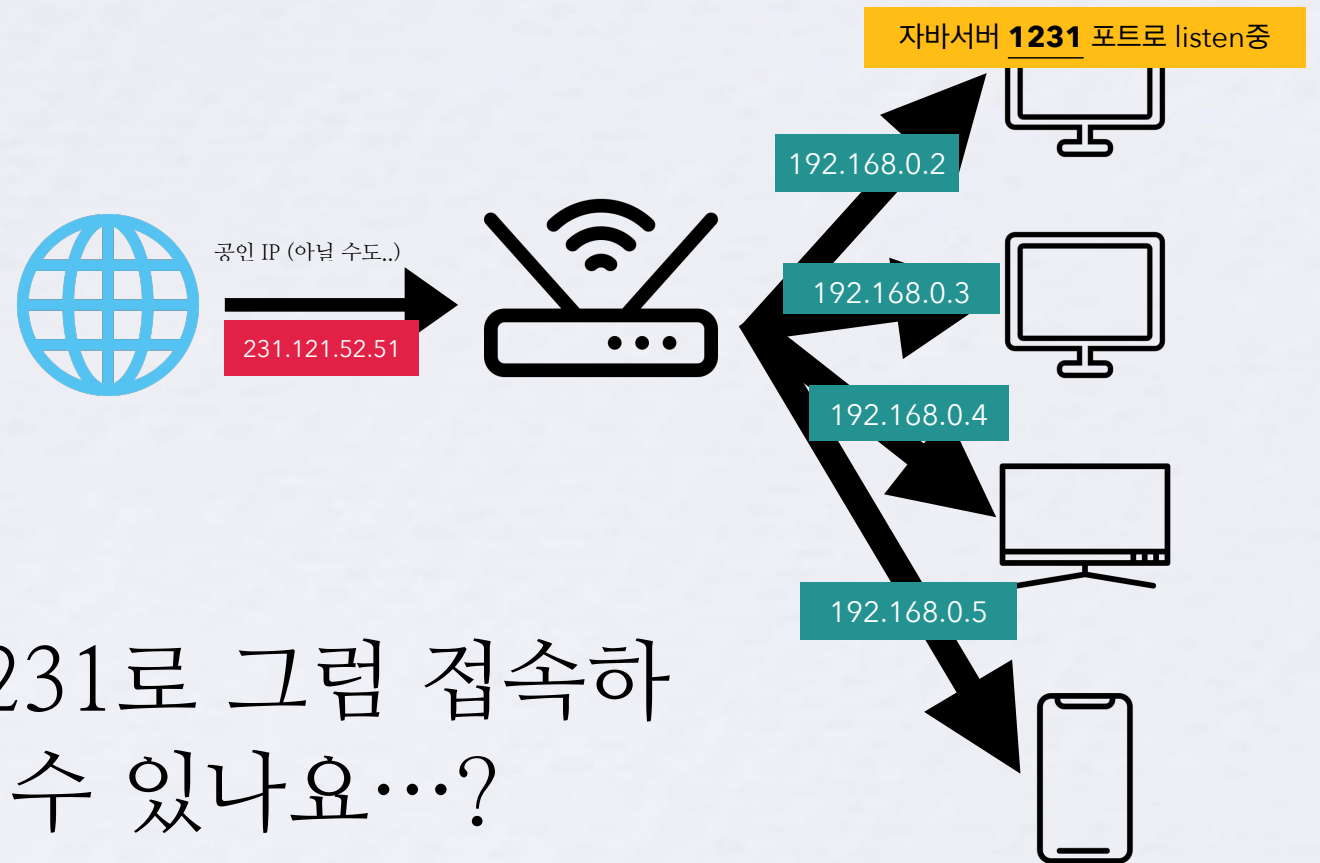
# Router..!!!!!!



The screenshot shows the '포트포워드 설정' (Port Forwarding Settings) page. The '규칙이름' (Rule Name) is '포트포워드 설정'. The '포트포워드 사용자정의' (Port Forward User Defined) checkbox is checked. The '규칙 비활성화' (Rule Deactivation) checkbox is unchecked. The '내부 IP주소' (Internal IP Address) is 192.168.0.2. The '현재 접속된 IP 주소' (Current Connected IP Address) checkbox is unchecked. The '프로토콜' (Protocol) is TCP/UDP. The '외부 포트' (External Port) is 1231, and the '내부 포트' (Internal Port) is 1231. The '순위' (Order) is 1. The '순위높임' (Increase Order) and '순위낮춤' (Decrease Order) buttons are visible. At the bottom, there are buttons for 'PC<-규칙저장' (Save Rule to PC), 'PC->규칙복원' (Restore Rule from PC), '파일 선택' (Select File), '선택한 파일 없음' (No selected file), '새규칙' (New Rule), '적용' (Apply), and '취소' (Cancel).

다음과 같이, 외부에서 1231 포트에 요청이 오면, 내부 IP주소인 192.168.0.2의 1231포트로 요청을 보내주도록 설정합니다.

# Router...!!!!!!

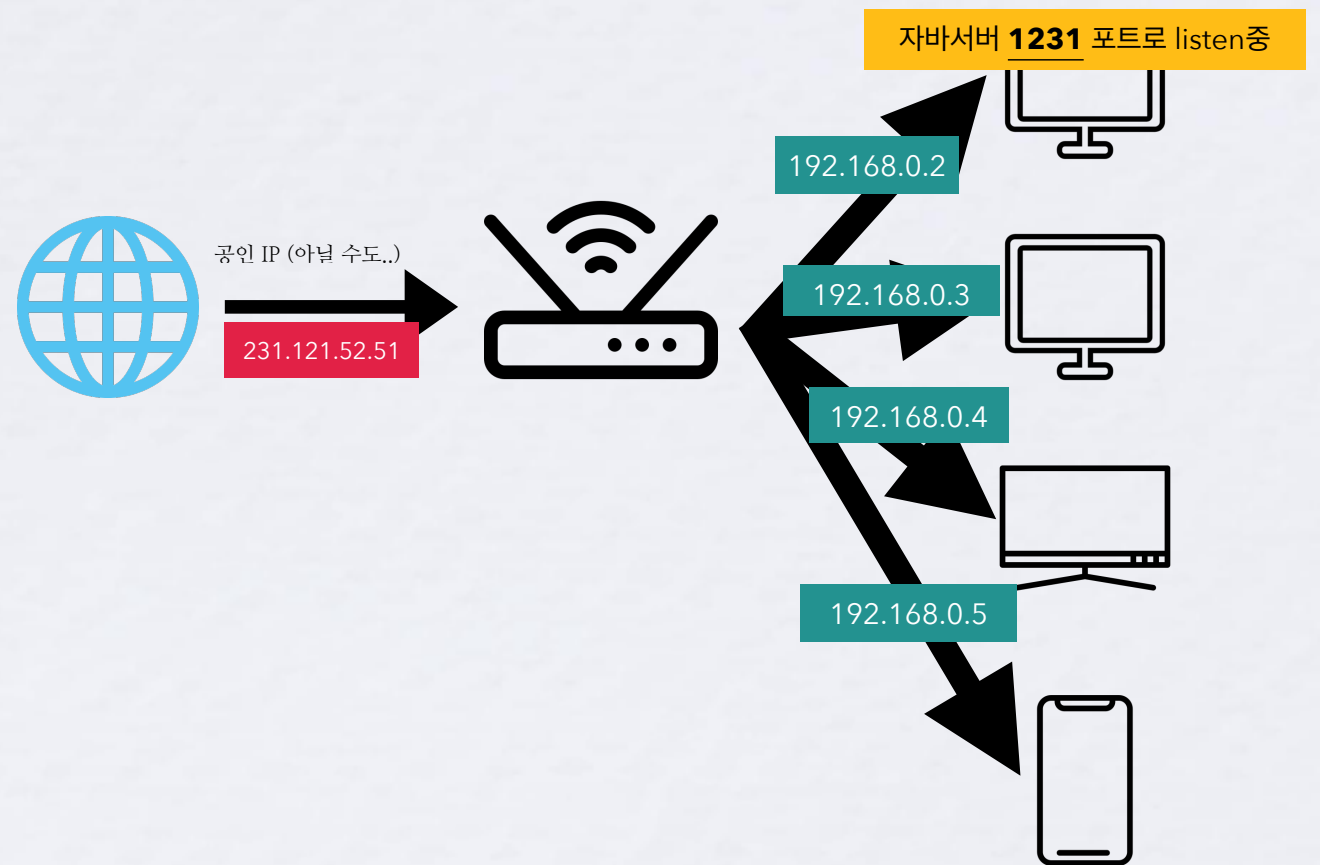


외부에서, 231.121.52.51:1231로 그림 접속하면, 자바서버로 요청을 보낼 수 있나요…?

네, 요청은 이제, 공유기에서 1231 포트에 들어온 요청은 내부 아이피 192.168.0.2의 1231 포트로 전달(forwarding) 합니다.

정말로 접속이 됩니다.

# Router...!!!!!!



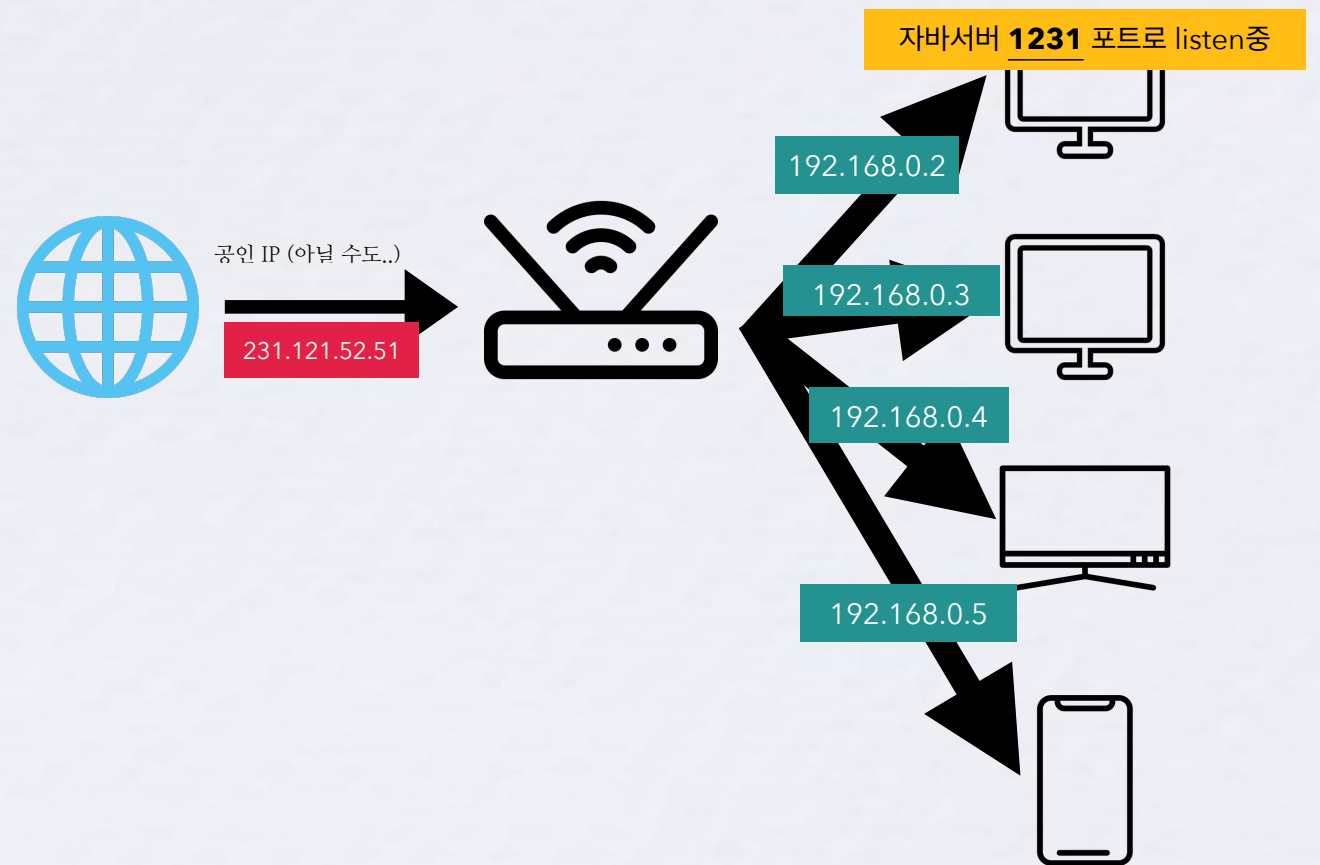
이게 끝인가요...?

질문하나 하겠습니다. 공유기는  
192.168.0.2:1231로 포트전달을 하고 있습니다.

자바 서버를 실행시키고 있는 컴퓨터가 받은 내부 IP는 공  
유기의 DDNS서버에 의해 자동으로 할당됩니다.

DDNS서버가, 컴퓨터의 내부 IP를 변경하지 않을까요?

# Router...!!!!!!



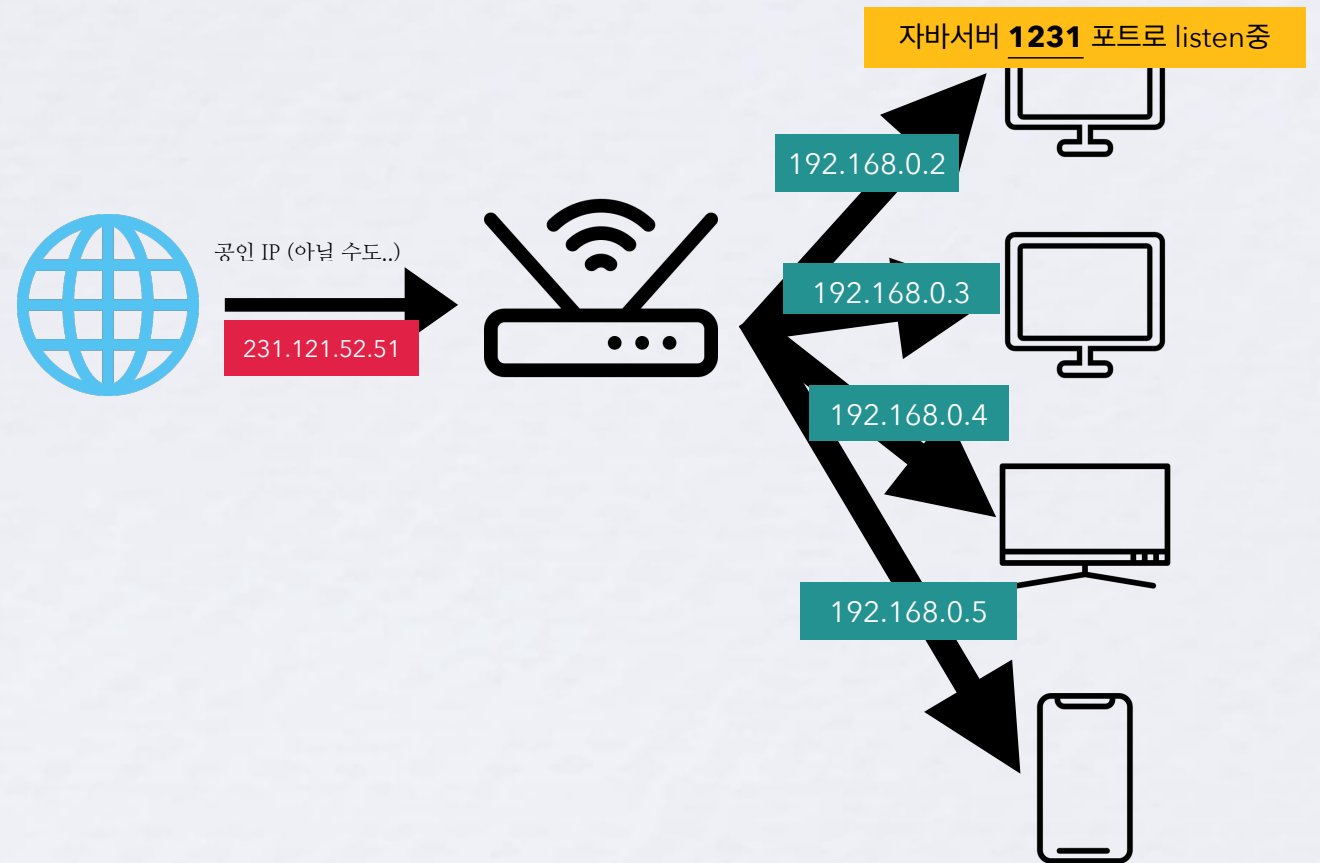
이게 끝인가요...?

질문하나 하겠습니다. 공유기는  
192.168.0.2:1231로 포트전달을 하고 있습니다.

자바 서버를 실행시키고 있는 컴퓨터가 받은 내부 IP는 공  
유기의 DHCP서버에 의해 자동으로 할당됩니다.

DHCP서버가, 컴퓨터의 내부 IP를 변경하지 않을까요?

# Router...!!!!!!

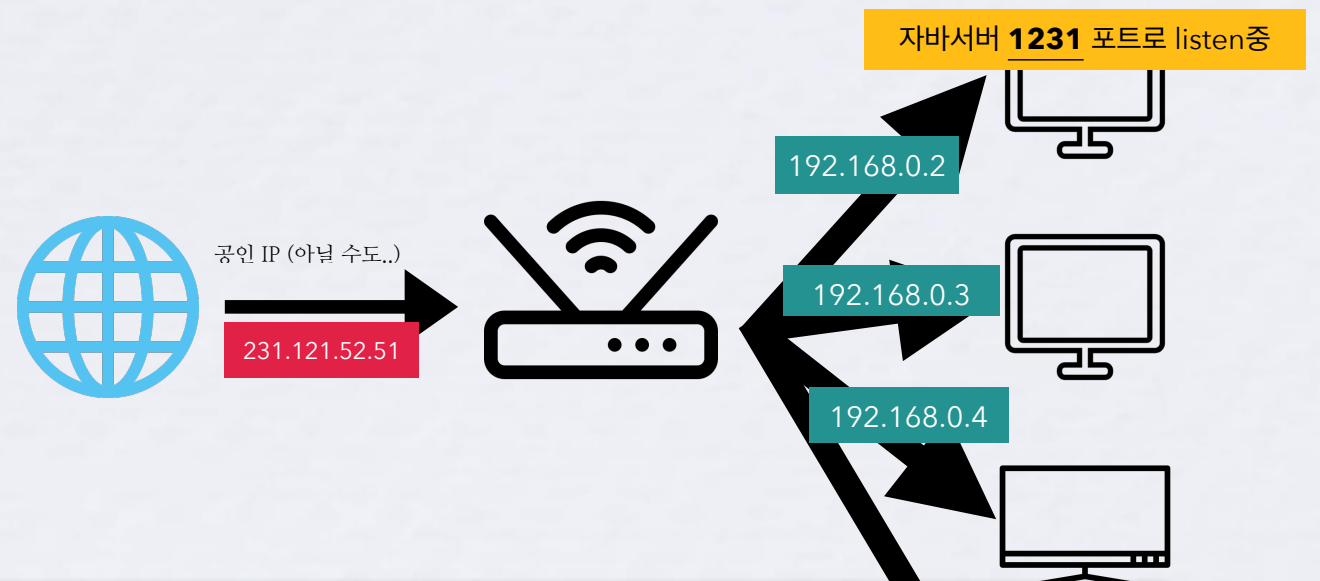


네, 바꿉니다.

자바 서버를 실행시키는 컴퓨터의 내부 아이피가 바뀌면,  
포트전달에 실패하게 됩니다.

쉽게 해결하는 방법은, DHCP서버가, 자바서버를 실행시  
키는 컴퓨터에게는 고정된, 내부 IP를 할당하게 하면 됩  
니다.

# Router.....!!!!!!



DHCP 서버 주소관리				<input type="checkbox"/> 등록 주소만 인터넷 허용	<input type="checkbox"/> 자동할당 주소만 인터넷 허용
등록된 주소 관리 (1/256 개)				<input checked="" type="checkbox"/> 삭제	<input type="checkbox"/>
192.168.0.9	14-BA-1F-15-4D-FA	DESKTOP-IUV9H4H	무선	<input type="checkbox"/>	<input type="checkbox"/>

주로 컴퓨터의 물리적 주소(MAC)을 특정 내부 IP에 할당하도록 예약 할 수 있습니다.

이제 할당된 내부 IP가 바뀌는 것은 걱정 하지 않아도 됩니다!



# Router...!!!!!!

Q. 완벽하게 따라했는데, 안됩니다.

A. 당신이 얻은 공인아이피로 외부에서 접속이 안된다면, 그건 사실 공인아이피가 아니라, 이중 공유기 이고, 상위 공유기 아이피가 공인 아이피라고 나오는 것입니다.

흔히, 이중 공유기 문제이고, 이는 상위 공유기에서 한번더 포트포워드 설정을 해야합니다.

안될 수가 없습니다.

# Router...!!!!!!

Q. 마인크래프트 서버를 친구들과 하기 위해 똑같은 절차를 밟으면 되는건가요?

맞습니다. 마인크래프트는 주로 25565포트로 요청을 받는 socket 서버입니다.

포트를 25565로 연결하고, 외부응답을 본인의 컴퓨터 주소로 공유기가 portForward(포트전달) 하여, 통신 할 것입니다.

안될 수가 없습니다.

# Backend FrameWorks

Http 서버를 만드는 프레임 워크들이 있습니다.

## \*FrameWork..?

프레임워크는 어떤 프로그램을 만들 때, 그 기본적인 구조나 틀을 미리 짜둔 재사용 가능한 소프트웨어입니다.

특정 목적을 가진 애플리케이션을 개발할 때, 개발자가 일일이 모든 것을 만들지 않고, 정해진 방식대로 조립하거나 확장해서 만들 수 있도록 도와주는 구조와 규칙의 집합입니다.

# Backend FrameWorks

Spring.. Java 언어로 작성 된 FrameWork



```
1  import org.springframework.boot.SpringApplication;
2  import org.springframework.boot.autoconfigure.SpringBootApplication;
3  import org.springframework.web.bind.annotation.GetMapping;
4  import org.springframework.web.bind.annotation.RestController;
5
6  @SpringBootApplication
7  public class DemoApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(DemoApplication.class, args);
11     }
12
13     @RestController
14     class HelloController {
15
16         @GetMapping("/")
17         public String home() {
18             return "Hello, Spring Boot!";
19         }
20     }
21 }
```



훨씬 더, 직관적이고 @annotation 문법으로 간단하게 엔드포인트 함수를 구현 할 수 있습니다.

# Backend Frameworks

NestJs..??

Node.js 위에서 동작하는, 타입스크립트 기반의  
백엔드 애플리케이션 프레임워크



```
1  import { Controller, Get } from '@nestjs/common';
2
3  @Controller()
4  export class AppController {
5    @Get()
6    getHello(): string {
7      return 'Hello, NestJS!';
8    }
9  }
```

# Backend Frameworks

## Flask..?

Flask는 파이썬(Python)으로 작성된 가볍고 간단한 웹 프레임워크로, 웹 서버를 빠르게 만들 수 있게 도와주는 도구입니다.



# Flask

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route("/", methods=["GET"])
6  def home():
7      return "Hello, Flask!"
8
9  if __name__ == "__main__":
10     app.run(debug=True, port=5000)
```



# Backend FrameWorks

## 다양한 백엔드 프레임 워크들

언어	프레임워크	주요 특징	주요 용도
JavaScript/TypeScript	Express	Node.js 기반, 미니멀하고 유연한 마이크로 프레임워크	REST API, 웹 서버
	NestJS	TypeScript 기반, 모듈화와 의존성 주입 지원	대규모 백엔드, 마이크로서비스
Python	Django	완전한 기능 제공, ORM, 인증, 관리 페이지 내장	웹 애플리케이션, REST API
	Flask	마이크로 프레임워크, 매우 가벼움	REST API, 간단한 웹 서버
Java	Spring Boot	대규모 엔터프라이즈용, 다양한 모듈 제공	대규모 웹 서비스, 마이크로서비스
Ruby	Ruby on Rails	컨벤션 중심, 생산성 높음	스타트업 웹앱, 프로토타입
PHP	Laravel	현대적 PHP 프레임워크, ORM, 인증, 라우팅 지원	웹 애플리케이션, API
Go	Gin	경량, 빠른 HTTP 웹 프레임워크	고성능 REST API, 마이크로서비스
	Echo	성능 중심, 미들웨어 지원	REST API, 웹 서비스
C#	ASP.NET Core	Microsoft 지원, 크로스 플랫폼, 고성능	엔터프라이즈 웹 서비스, API
Elixir	Phoenix	실시간 기능 강점, BEAM VM 기반	채팅, 실시간 앱, 고가용성 시스템

# End.

- 그 동안 제 이야기 들어주셔서 감사합니다.