

삼성 청년 SW 아카데미

Node.js

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

1장. 클라이언트와 서버

챕터의 포인트

- 클라이언트와 서버
- IP 와 Port

클라이언트와 서버

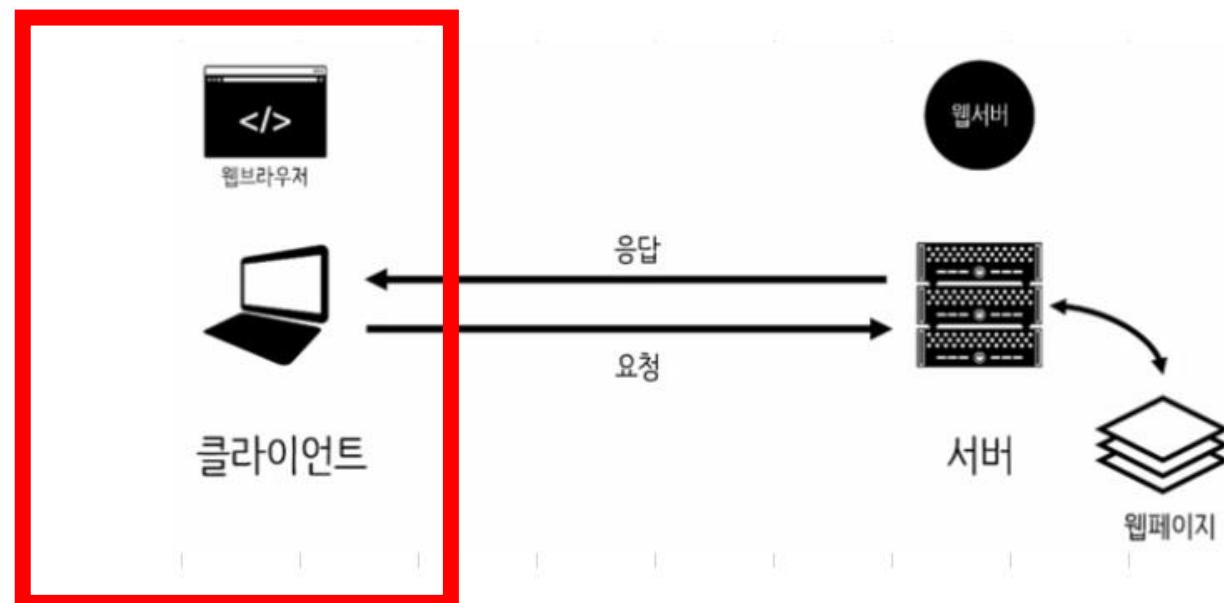
최종목표

- 클라이언트 요청 발생 시
- DB에 접근해서 CRUD 한 후
- 적절한 결과를 JSON으로 리턴하는
- Node.js Express REST API 서버 제작

우선, 클라이언트와 서버의 개념부터 알아보자.

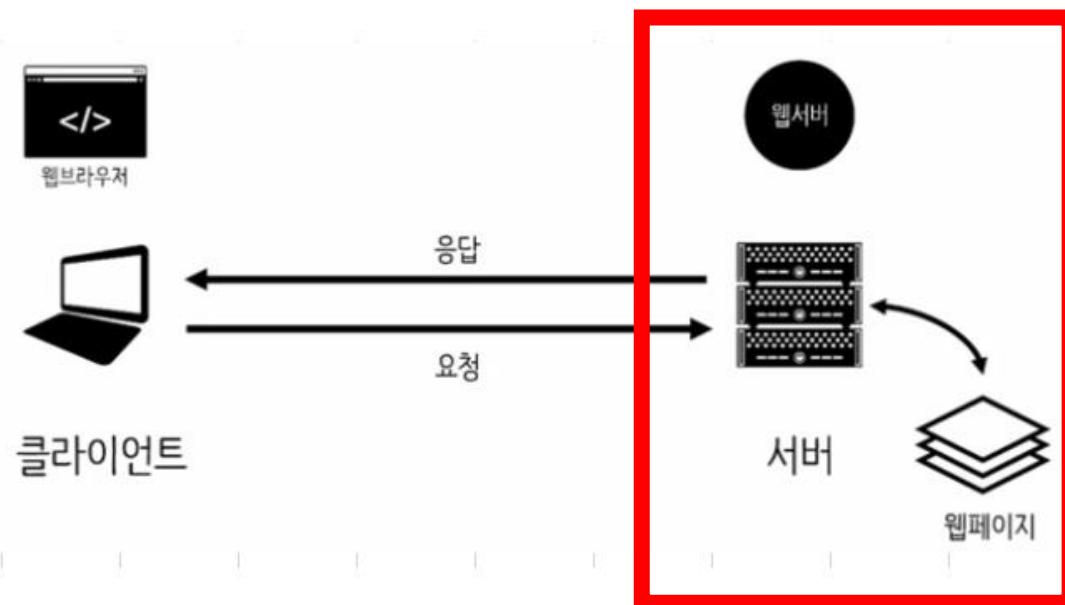
프론트엔드

- 웹에서는 클라이언트가 프론트엔드
- 사용자 눈에 보이고 상호작용하는 부분
- 서버를 통해 접속할 수 있는 애플리케이션이나 서비스
 - ex) 브라우저, 웹사이트
- 서버에게 요청 (request) 하면, 서버로부터 응답 (response) 을 받는다.
- 언어 및 기술: HTML, CSS, JavaScript, React.js, Vue.js ...



다른 말로 백엔드

- 사용자 눈에 보이지 않는 곳에서 작동하는 부분
- 클라이언트에게 정보나 서비스를 제공하는 시스템
 - ex) 웹서버, 데이터베이스 서버
- 클라이언트의 요청 (request) 을 받으면, 적절한 처리 후 클라이언트에게 응답 (response)
- 언어 및 기술: Java, Python, Node.js, Server, DB, API ...



타워형, 랙형으로 나뉨

- 일반 데스크톱도 얼마든지 쓸만한 서버가 될 수 있으나,
- 서버 컴퓨터의 CPU (인텔 제온) 는 데스크톱 CPU 와는 다르게
- 멀티스레드 기능을 보강



각 서버는 역할이 존재

- Client 들에게 HTTP 문서를 전달해주는 컴퓨터: 웹서버
- Client 들의 DB 를 관리하는 서버 : DB 서버
- Client 들의 게임 데이터를 관리 & 처리 해주는 컴퓨터 : 게임 서버

IP 와 Port

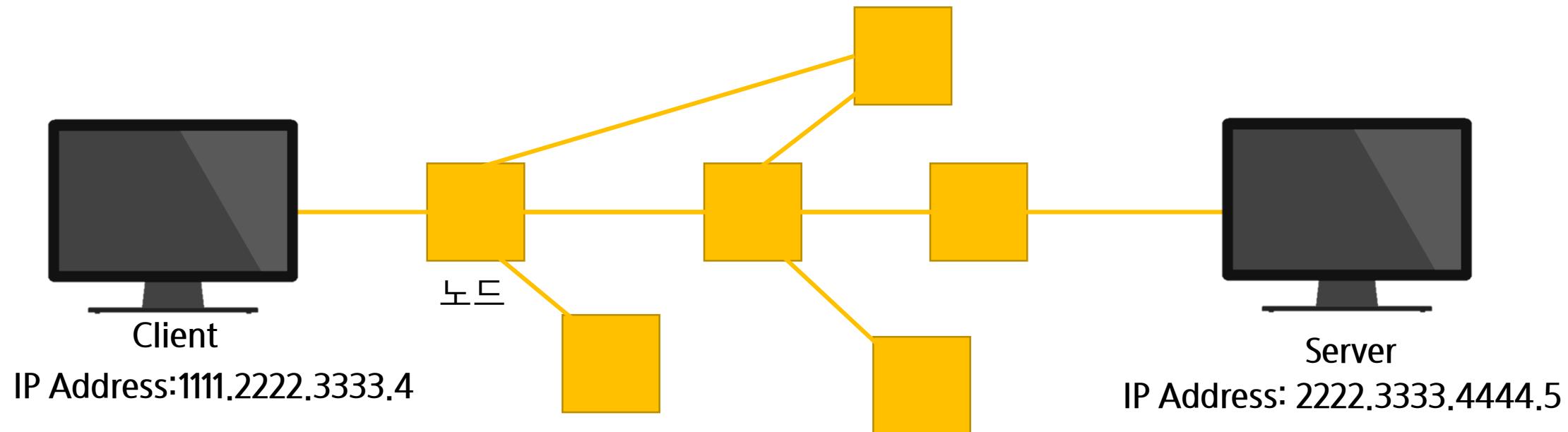
IP(Internet Protocol)

- Protocol : 규칙
- 인터넷에서, 송신 호스트와 수신 호스트가 정보를 주고받는데 사용하는 규칙
 - Host : 네트워크에 연결된 컴퓨터 또는 주변기기를 의미 ex) PC, 프린터, 서버, 노트북, 스마트폰 ...
- 송신 호스트와 수신 호스트는 **동일한 프로토콜을 사용해야 통신 가능**



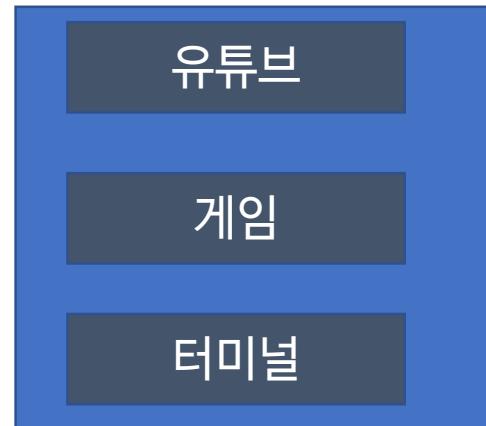
IP Address

- 인터넷상에 있는 컴퓨터의 고유한 주소



예시

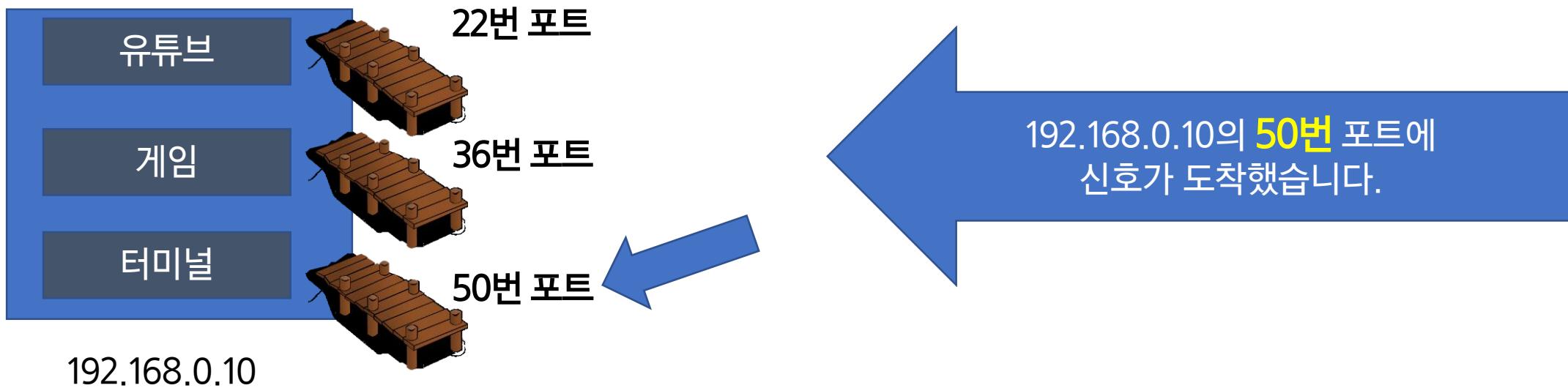
- 한 컴퓨터에서 통신을 하는 프로그램이 여러개가 있다.



컴퓨터
(192.168.0.10)

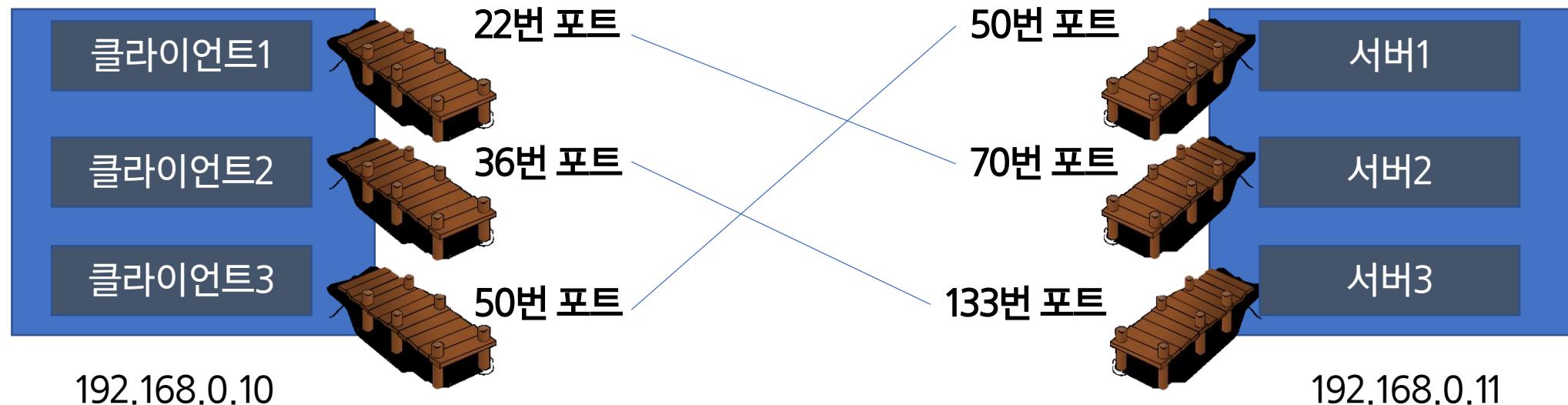
Port

- 어떤 신호가 들어왔을때,
- 어떤 프로그램에게 도착한 신호인지, 구분을 위한 번호



IP 포트 번호

- IP + 포트번호 까지 알아야, 프로그램끼리 통신이 가능



22번 포트

- ssh, sftp 프로토콜을 쓸 때 자주 사용되는 포트 번호 (원격 접속 시 사용)

80번 포트

- http 프로토콜을 쓸 때 자주 사용되는 포트번호
- 80번 이외에도 8000, 8080, 8081 ... 등 사용
- (Node.js 수업 때는 8080, 8081 번 포트를 지정해서 쓸 것)

국제관리기구에서 정한 포트 번호이다.
well-known port number 라고 하며,
권장사항이다.

Well-known Port Number (0 ~ 1023)

- Document에서 Well-known 은 예약된 이라는 뜻으로 쓰인다.
 - 80 / 443 : HTTP / HTTPS
 - 22 : FTP / SSH

Registered Port (1024 ~ 49151)

- 여러 Tool이 쓰는 포트들
 - 3306 : MySQL
 - 8080 : HTTP 대체용

Dynamic Port (49152 ~ 65535)

- 필요할 때마다 임의의 번호를 사용하는 곳

2장. 클라우드 서비스

챕터의 포인트

- 클라우드 서비스란
- 리눅스 개요

클라우드 서비스란

- cloud = 구름
 - 구름 속에 무엇이 있는지 몰라도, 얼마든지 사용자가 원하는 것을 꺼내어 사용
- 클라우드 서비스는 컴퓨터를 대여해주는 서비스
 - 그러나, 하나의 완성된 서버 컴퓨터를 빌려주는 게 아니라,
고객 요청 시 적절한 컴퓨터 자원을 가져와 **가상의 컴퓨터**를 만들어 대여함
 - ex) CPU, 메모리, 디스크 ...
 - 정확하게 말하자면, 컴퓨터 자체가 아니라, **컴퓨터의 자원을 빌려오는 것**
 - 사용자는 초당 사용료 지급



아마존의 AWS 사업

- Black Friday, 추수감사절, 아마존 이벤트시 최고 트래픽 발생
- 트래픽 관리 노하우를 AWS로 운영 시작



전산실 운영 비용 아낄 수 있음

- 월 정기 비용 : 다량의 서버관리자 급여 / 전기비 / 사무실 임대료 / 회선비용
- 일시적 비용 : L4 스위치 / 방화벽 / 대당 서버 비용 / 랙 / 항온항습장치 / UPS

클라우드 이용시

- 월 정기 비용 : 한 명의 서버관리자 급여 / 시스템 사용료
- 일시적 비용 : 없음

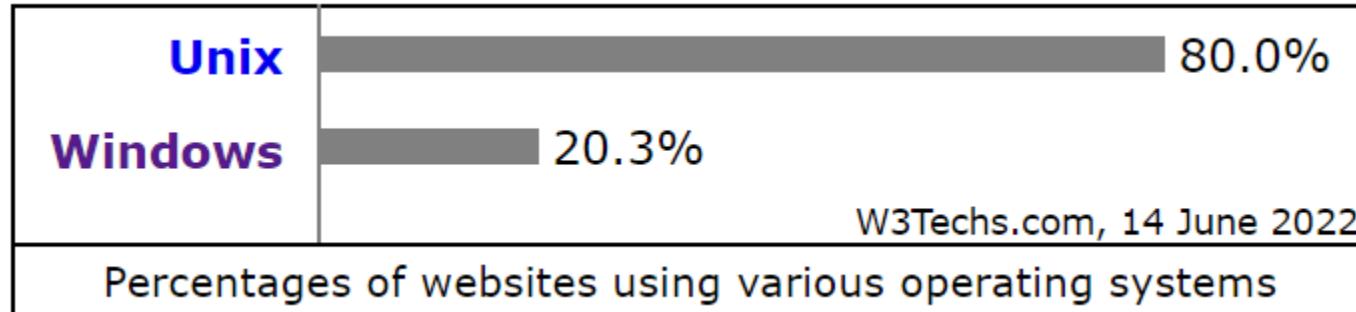
리눅스 개요

여러가지 운영체제

- Windows
 - PC시장(Personal Computer) 지배
- 리눅스
 - 서버
 - 임베디드 OS 시장 지배
- Android
 - 리눅스로 만들어진 Mobile OS
- MacOS
 - 유닉스로 만들어진 Apple OS

서버용

- 2022년 기준, Unix + Linux 가 서버의 약 80% 차지
- W3Techs.com 자료에는 Linux가 Unix 계열이기에, Unix로 표기가 되어있음
 - Unix : C 언어 제작자가 만든 OS, 이 계열로 대표적으로 iOS, macOS, Linux 등이 있음
 - Linux : Unix 계열 운영체제로, 리눅스토발즈가 Unix 를 토대로 만든 OS



Ubuntu

- 무료
- 리눅스 배포판 중 가장 널리 쓰이는 배포판
- 영국 캐노니컬에서 유지보수 중



스택

- 클라우드 서비스: AWS
- 운영체제: Ubuntu 18.04 LTS
- 서버: Node.js
- 데이터베이스: MySQL



3장. AWS

챕터의 포인트

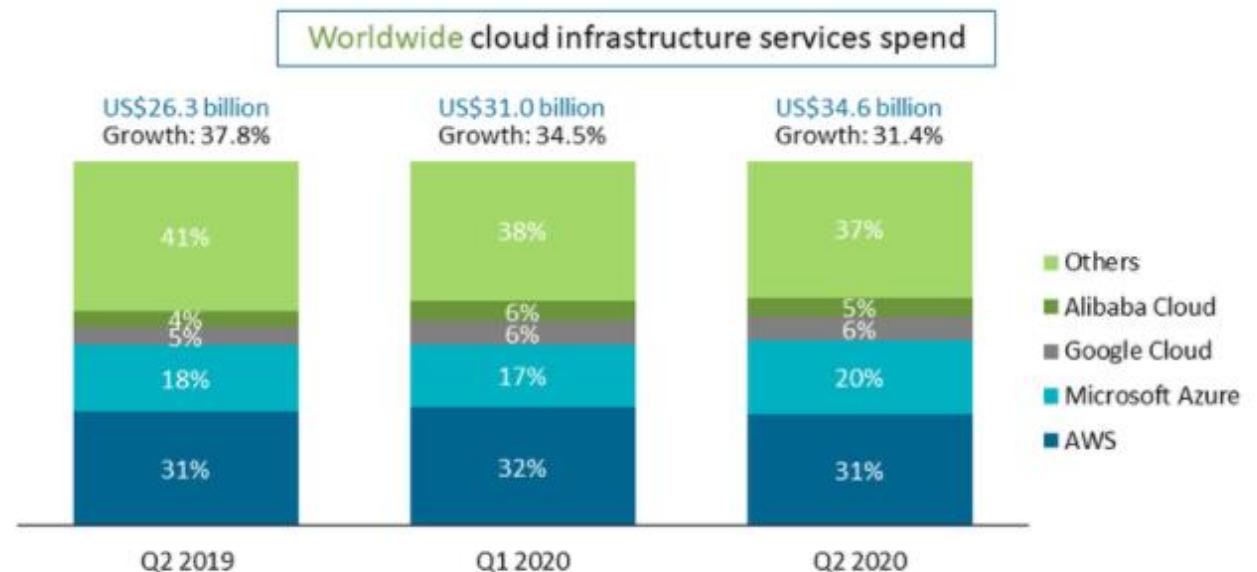
- AWS 개요
- AWS 인스턴스 생성
- SSH 연결

AWS 개요

Amazone Web Service

- AWS은 2016년 한국 리전 시작, 가용영역 4개 (전 세계에서 네 번째 규모)
- Google Clude Platform은 2020년 한국 리전 시작

Top four providers account for 63% of cloud spend

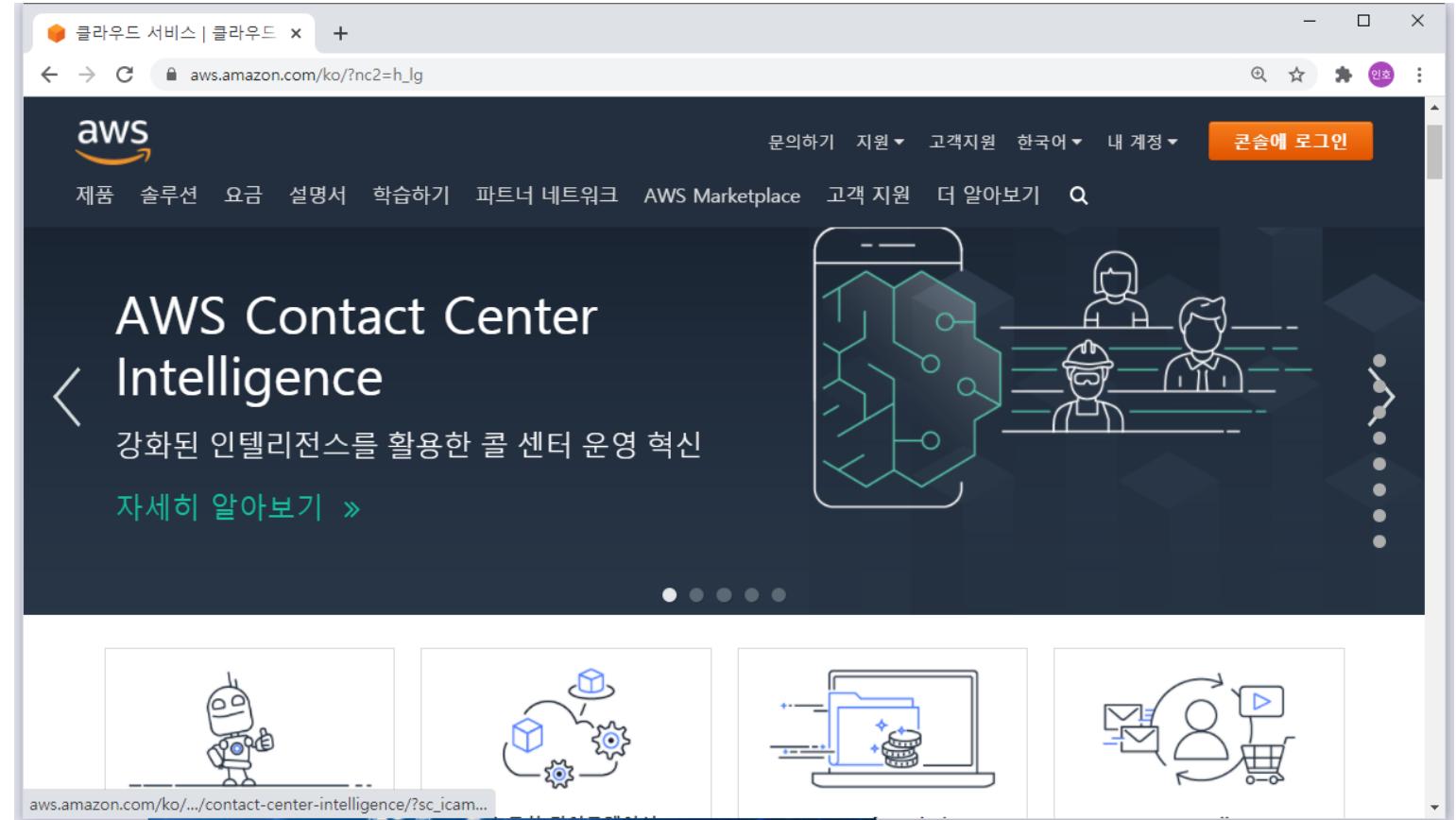


Source: Canalys estimates, July 2020

 canalys

신용카드 기입 필요

- 1 달러 테스트 결제가 된 후
2 ~ 3 주 후 취소 & 환불 됨

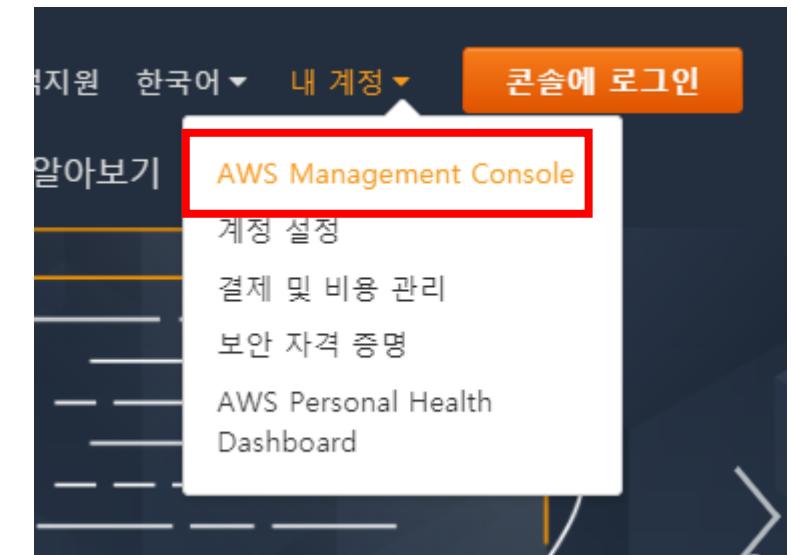
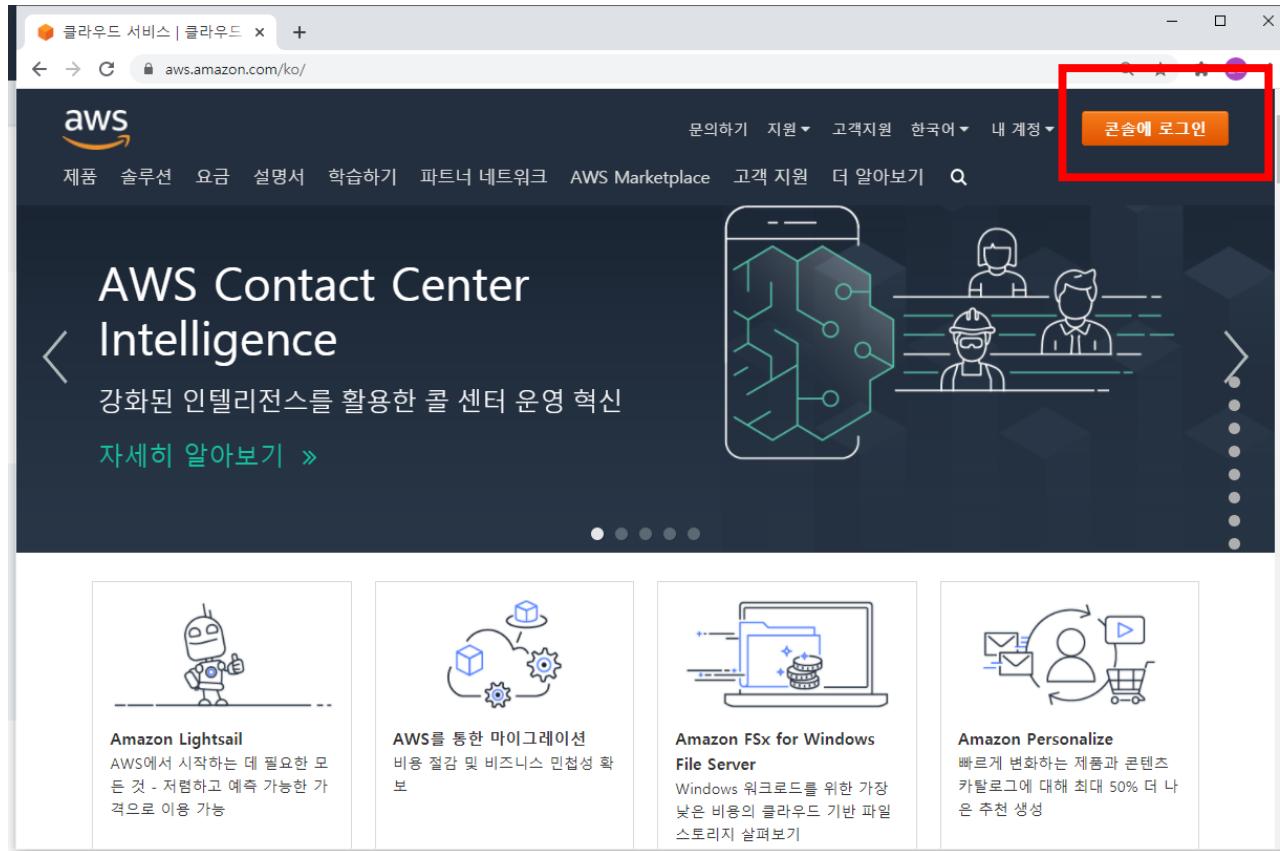


The screenshot shows the AWS Billing Management Console interface. The left sidebar has a tree view with nodes like 'Home', '결제' (Payment), '청구서' (Bills), '결제' (Payment), '크레딧' (Credit), '구매 주문' (Purchase Order), 'Cost & Usage Reports', 'Cost Categories', '비용 할당 태그' (Cost Allocation Tag), 'Free Tier', 'Billing Conductor', 'Cost Management', 'Cost Explorer', 'Budgets', 'Budgets Reports', '절감 계획' (Savings Plan), '기본 설정' (Basic Settings), '결제 기본 설정' (Payment Basic Settings), '결제 방법' (Payment Methods) which is highlighted in orange, '통합 결제' (Unified Payment), and '세금 설정' (Tax Settings). The main content area has a message box: '이제 결제 프로필 기능을 사용할 수 있습니다.' (You can now use the payment profile feature). Below it is a section titled '결제 방법' (Payment Methods) under '기본 결제 설정' (Basic Payment Settings). It shows a VISA card icon. To the right are fields for '청구지 주소/기본 결제 방법 주소' (Bill-to Address/Billing Method Address) and '전화번호' (Phone Number). A '결제 프로필 보기' (View Payment Profile) button is present. At the bottom, there's a table with columns '신용카드' (Credit Card), '카드에 기입된 이름' (Name on Card), and '만료 날짜' (Expiration Date). A VISA card is listed. Buttons at the bottom include '편집' (Edit), '삭제' (Delete), and '현재 기본값' (Current Default). A red box highlights the '카드 추가' (Add Card) button at the bottom left.

카드 추가 버튼을 클릭해 결제카드 등록

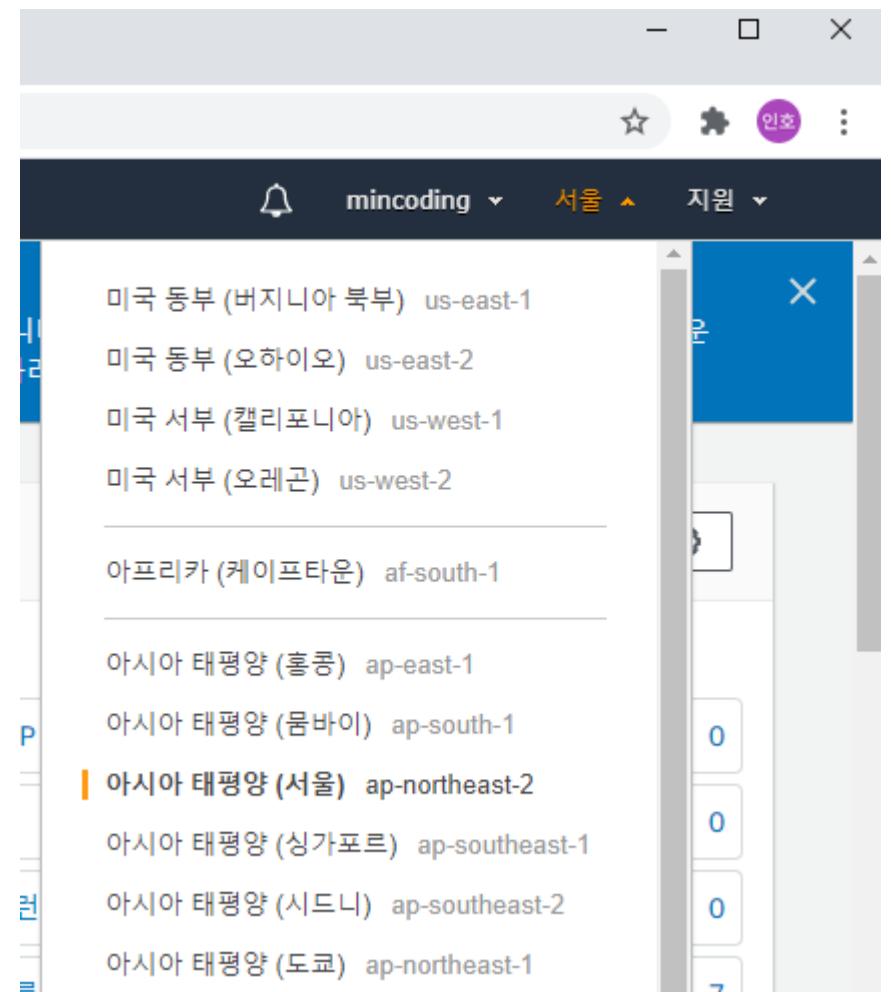
AWS 서비스를 이용하려면 “콘솔에 로그인” 필요

- 모든 서비스는 콘솔에서 시작함
- AWS Management Console = 콘솔



서울 리전을 선택

- 서버가 운영될 지역을 선택
- 가까울 수록 속도가 빠름
- 리전별 가격이 다름



사용량 알림받기

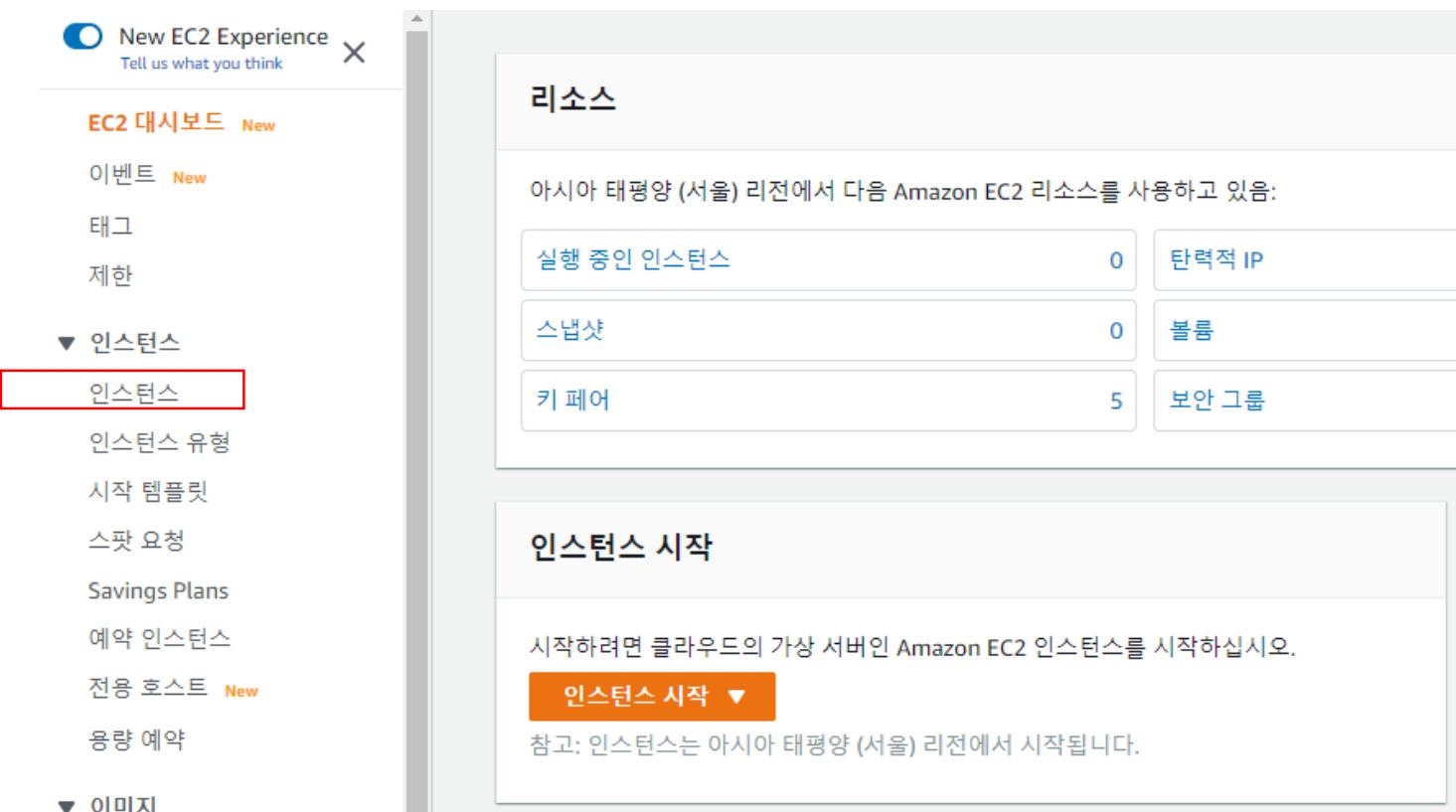
- 프리티어는 1년간 무료제공
- 소멸시 알림이 되도록 체크

The screenshot shows the AWS Billing console's 'Basic Settings' page. On the left, there is a sidebar with various navigation options like '결제 기본 설정', '결제 방법', '통합 결제', and '세금 설정'. The main content area is titled '기본 설정' and contains two sections: '결제 기본 설정' and '비용 관리 기본 설정'. In the '비용 관리 기본 설정' section, there is a checked checkbox for '프리 티어 사용량 알림 받기' (Receive Free Tier usage alerts) and a text input field for '이메일 주소' (Email address) containing 'test@example.com'. This entire section is highlighted with a red box.

AWS 인스턴스 생성

Elastic Compute Cloud (EC2)

- 가상컴퓨터
- AWS 의 대표 서비스
- 컴퓨터 1대 = 인스턴스



1. 리전 서울로 지정

새 위젯 출시 공지입니다. 콘솔 홈의 하단에서 찾을 수 있습니다.

최근에 방문한 서비스 정보

EC2

Database Migration Service

VPC

AWS Billing Conductor

AWS Organizations

Route 53

RDS

S3

2. EC2 클릭

AWS 시작

AWS 시작하기

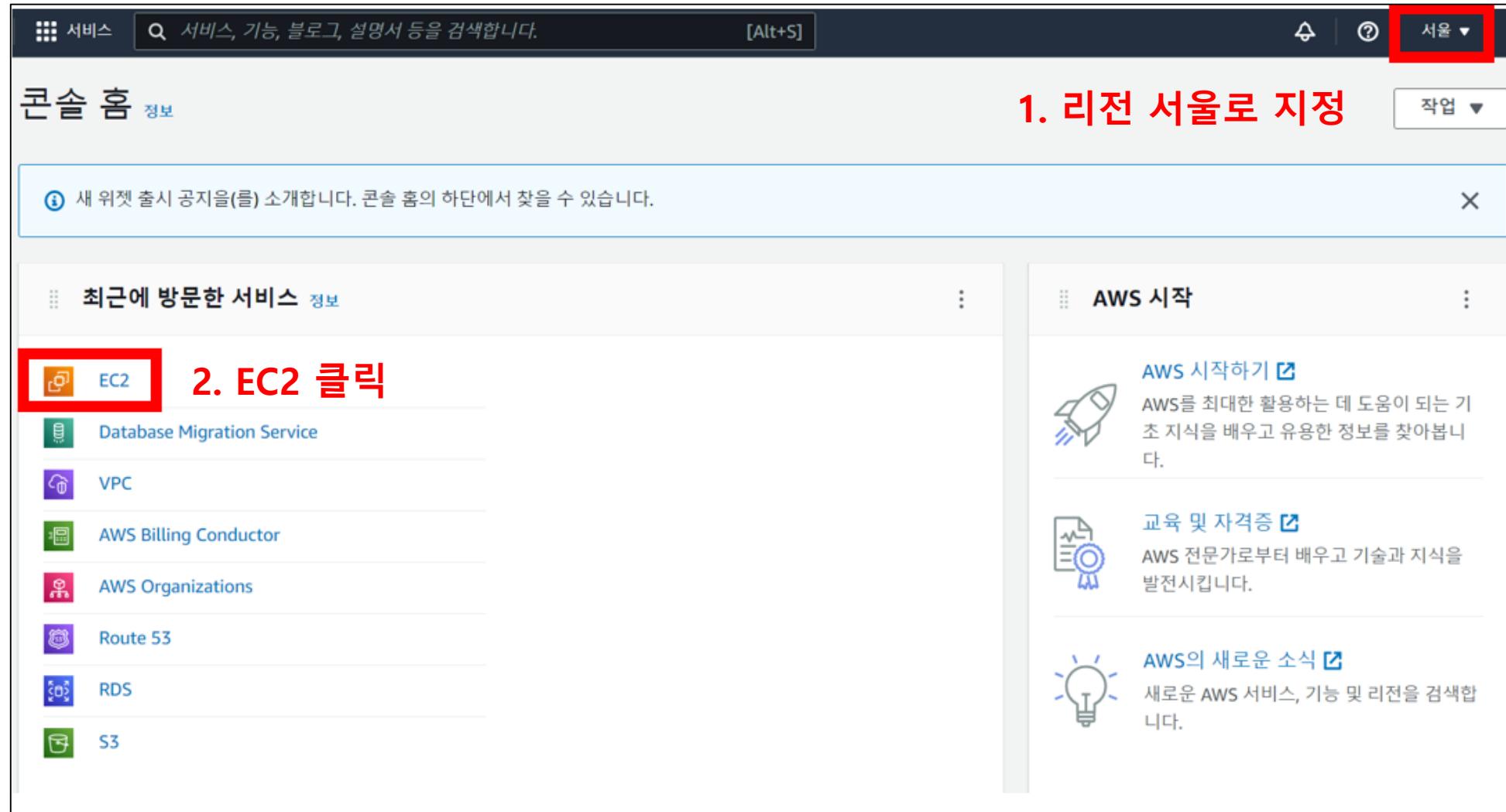
AWS를 최대한 활용하는 데 도움이 되는 기초 지식을 배우고 유용한 정보를 찾아봅니다.

교육 및 자격증

AWS 전문가로부터 배우고 기술과 지식을 발전시킵니다.

AWS의 새로운 소식

새로운 AWS 서비스, 기능 및 리전을 검색합니다.



리소스

아시아 태평양 (서울) 리전에서 다음 Amazon EC2 리소스를 사용하고 있음:

인스턴스(실행 중)	1	로드 밸런서	0	배치 그룹	0
보안 그룹	2	볼륨	1	스냅샷	1
인스턴스	1	전용 호스트	0	키 페어	1
탄력적 IP	1				

인스턴스 시작

시작하려면 클라우드의 가상 서버인 Amazon EC2 인스턴스를 시작하십시오.

인스턴스 시작 ▾

서버 마이그레이션

참고: 인스턴스는 아시아 태평양 (서울) 리전에서 시작됩니다.

서비스 상태

리전
아시아 태평양 (서울)

상태
✓ 이 서비스가 정상적으로 작동 중입니다.

EC2 > 인스턴스 > 인스턴스 시작

인스턴스 시작 정보

Amazon EC2를 사용하면 AWS 클라우드에서 실행되는 가상 머신 또는 인스턴스를 생성할 수 있습니다. 아래의 간단한 단계에 따라 빠르게 시작할 수 있습니다.

이름 및 태그 정보

이름

ssafy-8th-node-aws

추가 태그 추가

식별하기 쉽게, 이름을 지어주자.

▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용 Quick Start

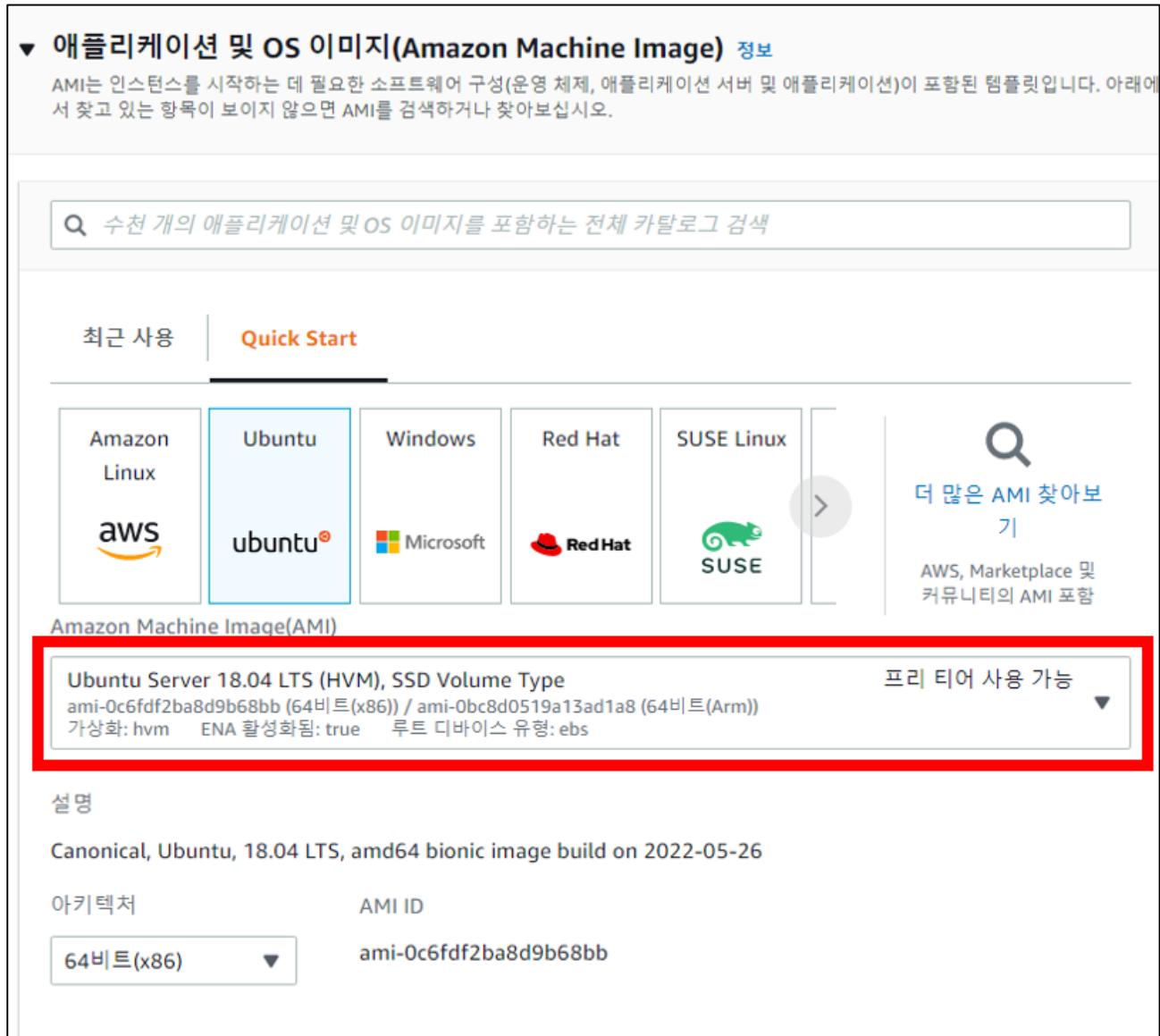
Amazon Linux Ubuntu Windows Red Hat SUSE Linux > 더 많은 AMI 찾아보기 AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Machine Image(AMI)

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
 ami-0c6fdf2ba8d9b68bb (64비트(x86)) / ami-0bc8d0519a13ad1a8 (64비트(Arm))
 가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

설명
Canonical, Ubuntu, 18.04 LTS, amd64 bionic image build on 2022-05-26

아키텍처 AMI ID
64비트(x86) ami-0c6fdf2ba8d9b68bb



프리티어 사용 가능한
Ubuntu Server 18.04 LTS



키 페어 생성

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

아래에 키 페어의 이름을 입력합니다. 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

키 페어 이름
 키 파일 이름 지정

이름은 최대 255개의 ASCII 문자를 포함할 수 있습니다. 선택 또는 후행 공백은 포함할 수 없습니다.

키 페어 유형

RSA RSA 암호화된 프라이빗 및 퍼블릭 키 페어

ED25519 ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식

.pem OpenSSH와 함께 사용

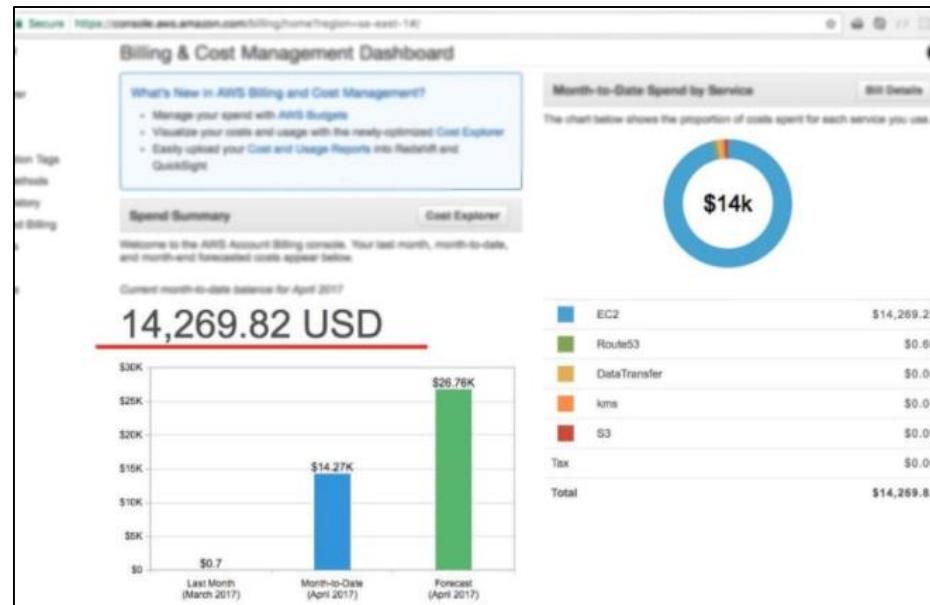
.ppk PuTTY와 함께 사용

취소 **키 페어 생성**

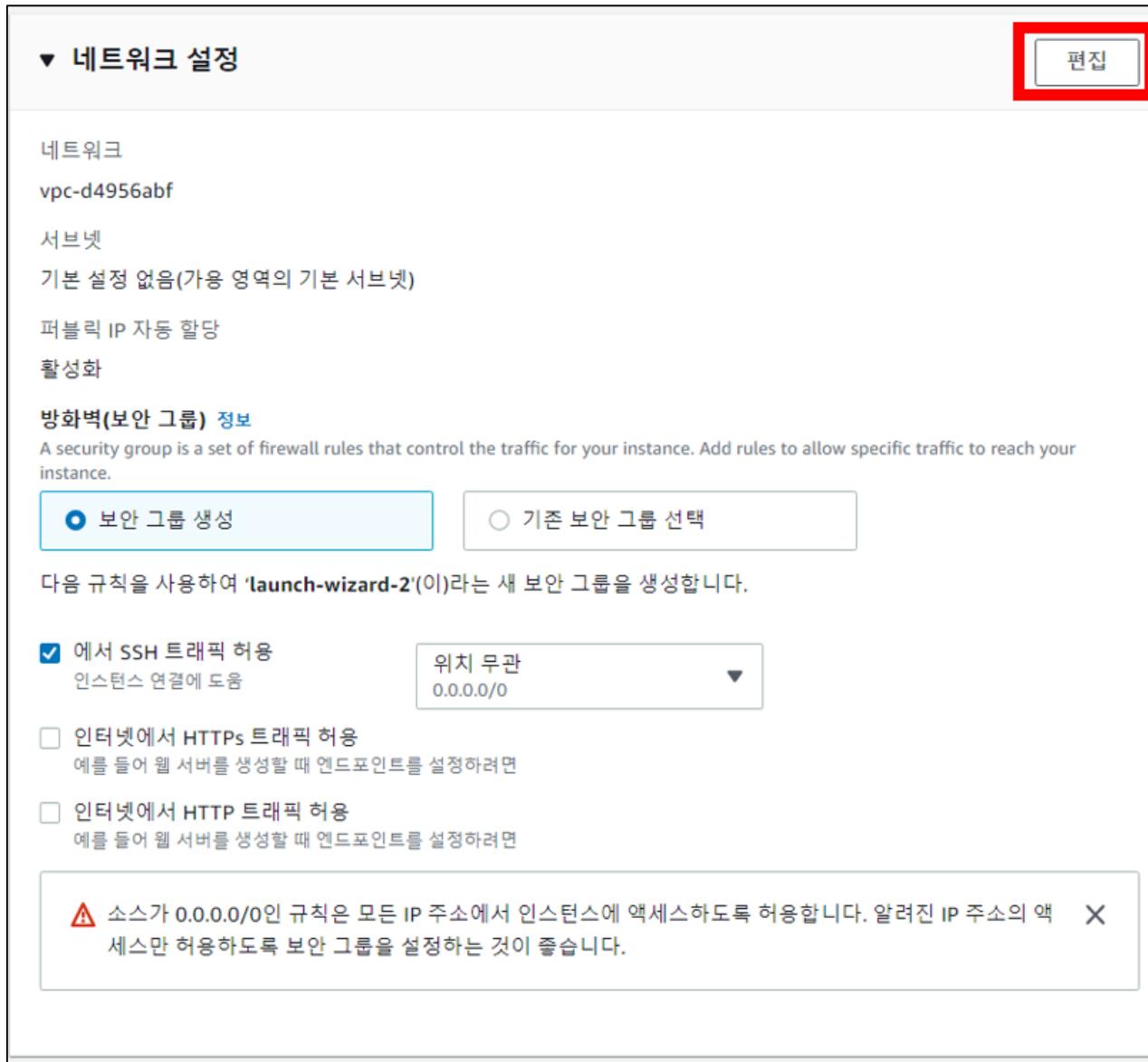
잃어버리지 않게 잘 저장

주의 : KEY 파일을 GitHub 에 업로드하게되면 생기는 일

- ‘KEY’ 파일을 인터넷 업로드 하는 경우



<https://dev.to/juanmanuelramallo/i-was-billed-for-14k-usd-on-amazon-web-services-17fn>



▼ 네트워크 설정

VPC - 필수 정보
vpc-d4956abf (기본값) 172.31.0.0/16

서브넷 정보
기본 설정 없음 새 서브넷 생성

퍼블릭 IP 자동 할당 정보
활성화

방화벽(보안 그룹) 정보
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

보안 그룹 생성 기존 보안 그룹 선택

보안 그룹 이름 - 필수
ssafy-8th-node-security

이 보안 그룹은 모든 네트워크 인터페이스에 추가됩니다. 보안 그룹을 만든 후에는 이름을 편집할 수 없습니다. 최대 길이는 255자입니다. 유효한 문자는 a~z, A~Z, 0~9, 공백 및 _-:/()#,@[]+=;&();!\$*입니다.

설명 - 필수 정보
ssafy-8th-node-security created 2022-06-13

인바운드 보안 그룹 규칙
▼ 보안 그룹 규칙 1 (TCP, 22, 0.0.0.0/0)

제거

식별 가능한 이름 및 설명 지정

▼ 요약

인스턴스 개수 [정보](#)

1

소프트웨어 이미지(AMI)

Canonical, Ubuntu, 18.04 LTS, ... [더 보기](#)
ami-0c6fdf2ba8d9b68bb

가상 서버 유형(인스턴스 유형)

t2.micro

방화벽(보안 그룹)

새 보안 그룹

스토리지(볼륨)

1개의 볼륨 – 8GiB

취소

인스턴스 시작



오른쪽 인스턴스 시작 버튼 클릭

실행중인 인스턴스 확인

Confidential

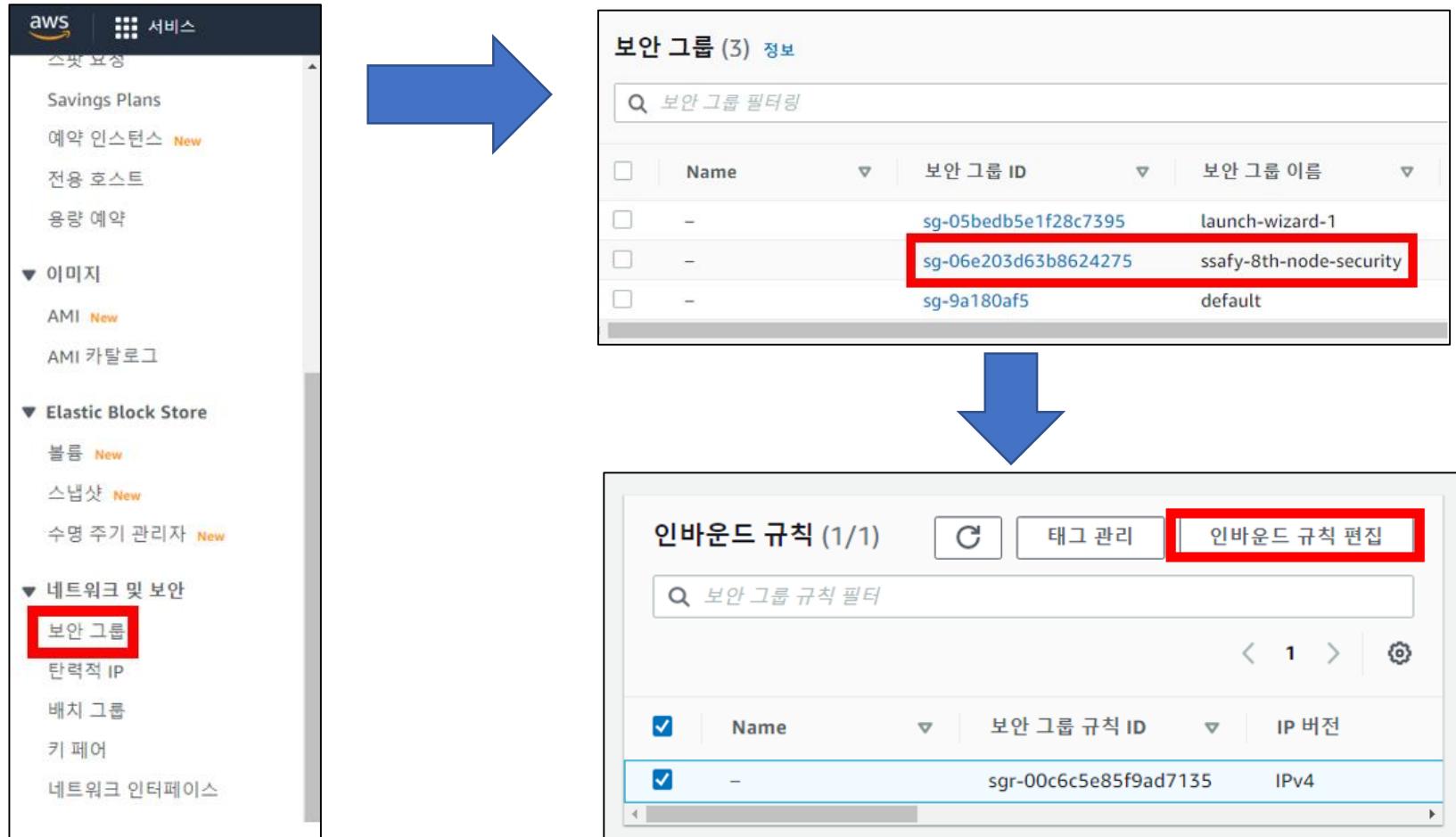
The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with options like EC2 대시보드, EC2 글로벌 보기, 이벤트, 태그, 제한, and 인스턴스. The '인스턴스' section is expanded, and the '인스턴스 New' button is highlighted with a red box. The main area displays a table of instances:

	Name	인스턴스 ID	인스턴스 상태	
<input type="checkbox"/>	jony-server	[REDACTED]	실행 중	
<input type="checkbox"/>	ssafy-8th-node-aws	[REDACTED]	실행 중	

A red box highlights the second row of the table, which corresponds to the instance named 'ssafy-8th-node-aws'. Below the table, a message in Korean states: '방금 만든 인스턴스가 실행중인것이 확인됨'.

방금 만든 인스턴스가 실행중인것이 확인됨

원활한 실습을 위해, 보안그룹을 미리 지정하자



인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

인바운드 규칙 정보

보안 그룹 규칙 ID

sgr-0833b982a6141e005

유형 정보

MySQL/Aurora

프로토콜 정보

TCP

포트 범위 정보

3306

소스 정보

사용자 지정

설명 - 선택 사항 정보

0.0.0.0 X

sgr-0ecd74faf6dc40e0b

사용자 지정 TCP

TCP

8081

Anywhere-...

sgr-0e218e9d7ded0a4f2

사용자 지정 TCP

TCP

8000

Anywhere-...

sgr-0c17d86f2634929fa

HTTP

TCP

80

Anywhere-...

sgr-0e4eb33a27dc498a7

사용자 지정 TCP

TCP

8080

Anywhere-...

sgr-00c6c5e85f9ad7135

SSH

TCP

22

Anywhere-...

-

모든 ICMP - IPv4

ICMP

전체

Anywhere-...

보안그룹 설정

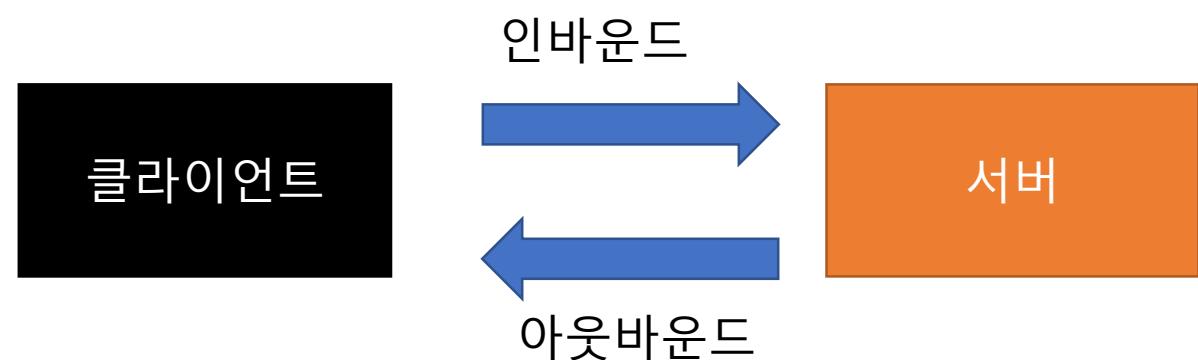
- SSH (22번 포트) : 원격 연결
- MySQL/Aurora (3306 포트) : MySQL 원격접속
- HTTP (80번 포트) : HTTP 통신
- 사용자 지정 TCP (8000, 8080, 8081) : HTTP 예비 포트
- 모든 ICMP - IPv4: PING 테스트
- 모두 **Anywhere-IPv4** 로 지정

• 인바운드

- 클라이언트가 자신의 서버 데이터에 들어올 수 있는 규칙
- 서버 내부로 들어오는 것
- 서버에 업로드할 때

• 아웃바운드

- 서버에서 나갈 수 있는 데이터에 대한 규칙
- 서버에서 다운로드 할 때

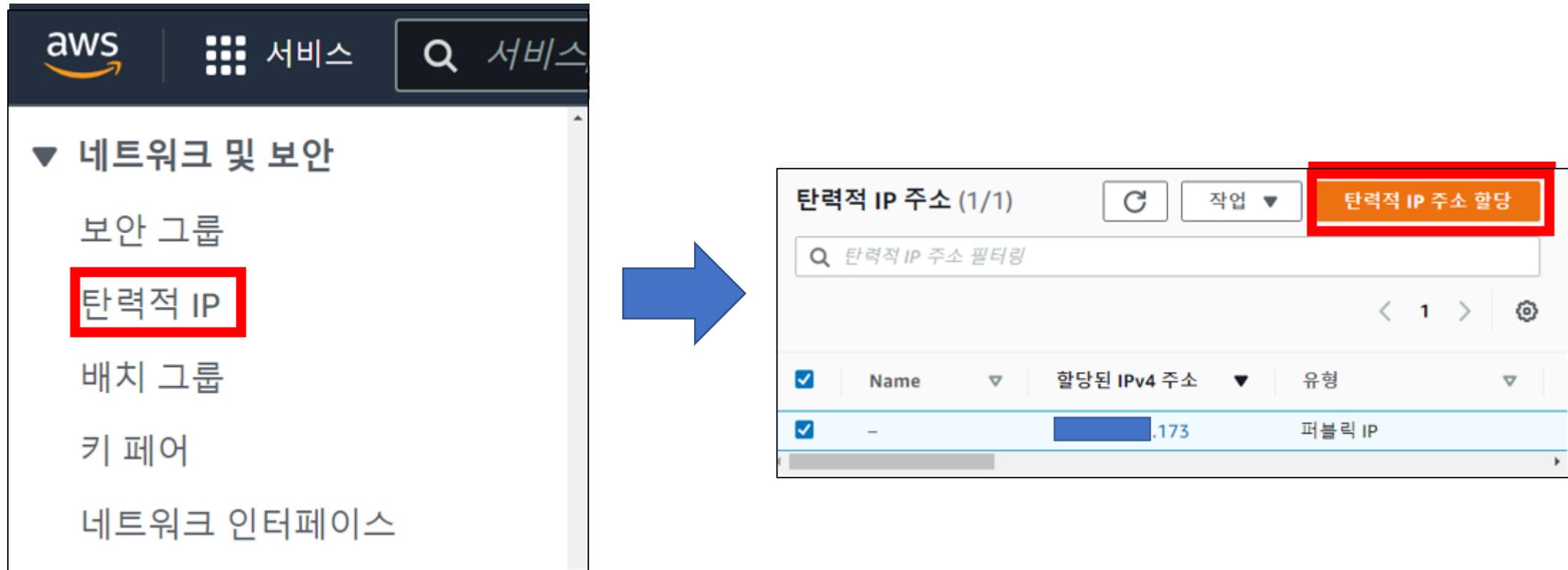


탄력적 IP

- 인스턴스를 중지하게 되면 다시 시작할 때 IP가 변경
- IP 가 바뀌는 불편함에 대한 개선점으로 탄력적 IP를 활용
- 탄력적 (Elastic) IP
 - AWS 리전 당 총 5개 무료로 사용 가능한 고정 IP
 - 추가요금 없으나, 만들어놓고 사용하지 않으면 요금 발생

탄력적 IP 설정하기

Confidential



탄력적 IP 설정하기

Confidential

탄력적 IP 주소 설정 정보

네트워크 경계 그룹 정보

ap-northeast-2

퍼블릭 IPv4 주소 풀

Amazon의 IPv4 주소 풀

AWS 계정으로 가져오는 퍼블릭 IPv4 주소 (풀을 찾을 수 없으므로 옵션이 비활성화됨) 자세히 알아보기

IPv4 주소의 고객 소유 풀 (고객 소유 풀을 찾을 수 없기 때문에 옵션이 비활성화됨) 자세히 알아보기

글로벌 정적 IP 주소

AWS Global Accelerator는 AWS 엣지 로케이션의 애니캐스트를 사용하여 전 세계에 발표된 글로벌 정적 IP 주소를 제공할 수 있습니다. 이를 통해 Amazon 클로브 네트워크를 사용하여 사용자 트래픽의 가용성과 지연 시간을 개선할 수 있습니다. 자세히 알아보기

액셀러레이터 생성

태그 - 선택 사항

태그는 사용자가 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.

리소스에 연결된 태그가 없습니다.

새로운 태그 추가

최대 50개의 태그를 더 추가할 수 있습니다.

취소 할당

이 탄력적 IP 주소 연결

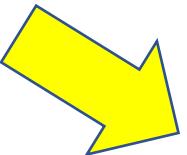
✓ 탄력적 IP 주소가 할당되었습니다.
탄력적 IP 주소 52.78.194.164



성공 시 다음 초록색 알림 발생

탄력적 IP 주소 (2)		
<input type="text"/> 탄력적 IP 주소 필터링		
Name	▼	할당된 IPv4 주소
-		.143.173
-		194.164

방금 생성된 탄력적 IP 를 클릭해보자



EC2 > 탄력적 IP 주소 > 3.39.96.142

작업 ▾

요약			
할당된 IPv4 주소	유형	할당 ID	역방향 DNS 레코드
<input type="button" value="."/> .96.142	<input type="checkbox"/> 퍼블릭 IP	<input type="button" value="."/>	-
연결 ID	범위	연결된 인스턴스 ID	프라이빗 IP 주소
-	<input type="checkbox"/> VPC	-	-
네트워크 인터페이스 ID	네트워크 인터페이스 소유자 계정 ID	퍼블릭 DNS	NAT 게이트웨이 ID
-	-	-	-
주소 팔	네트워크 경계 그룹	Amazon	<input type="checkbox"/> ap-northeast-2

탄력적 IP 설정하기

Confidential

탄력적 IP 주소 연결

이 탄력적 IP 주소에 연결할 인스턴스 또는 네트워크 인터페이스를 선택합니다. (43.200.26.24)

탄력적 IP 주소: [REDACTED].26.24 여기 있는 주소로 할당될 예정

리소스 유형

탄력적 IP 주소를 연결할 리소스의 유형을 선택합니다.

인스턴스

네트워크 인터페이스

⚠ 탄력적 IP 주소가 이미 연결되어 있는 인스턴스에 탄력적 IP 주소를 연결하면 이전에 연결된 탄력적 IP 주소가 연결 해제되지만 계정에는 계속 할당된 상태로 남아 있습니다. [자세히 알아보기](#)

인스턴스

[] 7aae 방금 만든 인스턴스 선택 × C

프라이빗 IP 주소

탄력적 IP 주소를 연결할 프라이빗 IP 주소입니다.

[] .229 ×

재연결

이미 리소스에 연결되어 있는 탄력적 IP 주소를 다른 리소스에 재연결할 수 있는지 여부를 지정합니다.

이 탄력적 IP 주소를 재연결하도록 허용

취소

연결

탄력적 IP 주소가 연결되었습니다.

탄력적 IP 주소 [REDACTED].26.24이(가) 인스턴스에 연결되었습니다.i-04d24f99a1d627aae

인스턴스 (2) 정보		
<input type="text"/> 검색		
블릭 IPv4 DNS	퍼블릭 IPv4 ...	탄력적 IP
2-13-209-143-173.ap-northeast...		
2-43-200-26-24.ap-northeast-2...	1.26.24	1.26.24

인스턴스로 가보면 고정IP 가 잘 할당되었음
이제 인스턴스 중지 후 재시작하더라도 IP 는 바뀌지 않는다.

SSH 연결

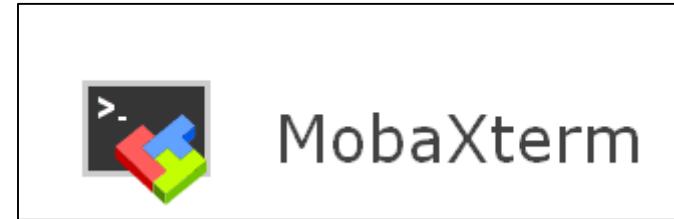
원격 접속 터미널

- Putty : 가장 널리 사용됨
- MobaXterm : 편리한 오픈소스 터미널



클라이언트 프로그램

- 터미널 프로그램 = mobaXterm
- 키보드 입력을 서버에 전달



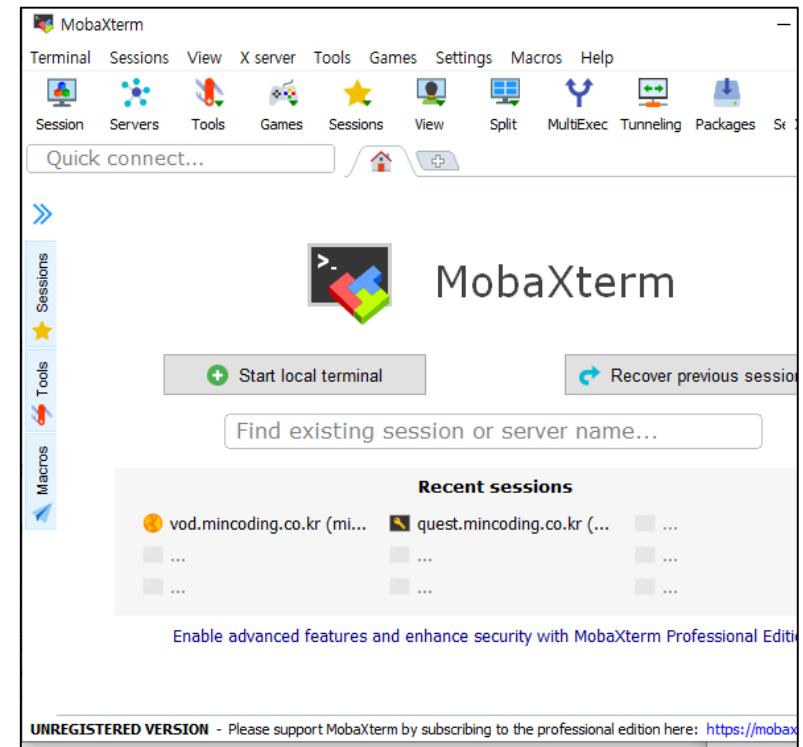
서버 프로그램

- 클라이언트의 키보드 입력 → 원격지 PC에 입력해주는 프로그램
- 결과 화면을 클라이언트에 전달한다.

Ubuntu Server 엔 기본적으로 openssh-server 가 설치되어 있다

MobaXterm

- <https://mobaxterm.mobatek.net/>
- 다운 받아서 설치하기



The screenshot shows the MobaXterm website homepage. At the top, there is a navigation bar with links: Home, Demo, Features, **Download**, Plugins, Help, Contact, and social media icons for Facebook, Twitter, and Google+. To the right are links for Customer area and Buy.

Home Edition (Left Column):

Free

- Full X server and SSH support
- Remote desktop (RDP, VNC, Xdmcp)
- Remote terminal (SSH, telnet, rlogin, Mosh)
- X11-Forwarding
- Automatic SFTP browser
- Master password protection
- Plugins support
- Portable and installer versions
- Full documentation
- Max. 12 sessions
- Max. 2 SSH tunnels
- Max. 4 macros
- Max. 360 seconds for Tftp, Nfs and Cron

Download now

Professional Edition (Right Column):

\$69 / 49€ per user*

* Excluding tax. Volume discounts [available](#)

Every feature from Home Edition +

- Customize your startup message and logo
- Modify your profile script
- Remove unwanted games, screensaver or tools
- Unlimited number of sessions
- Unlimited number of tunnels and macros
- Unlimited run time for network daemons
- Enhanced security settings
- 12-months updates included
- Deployment inside company
- Lifetime right to use

[Subscribe online / Get a quote](#)

 MobaXterm

Home Demo Features **Download** Plugins Help Contact [!\[\]\(57673b4ed67b5e9d6314d6d3026eb8ed_img.jpg\)](#) [!\[\]\(5e4cf2f4deef4e1c3a49f3b4348744b6_img.jpg\)](#) [!\[\]\(a33950fe47829b077e3d8930299956d5_img.jpg\)](#) [!\[\]\(5d84870b2589d050a98fafdf3f1097d2_img.jpg\)](#)

[Customer area](#) [Buy](#)

MobaXterm Home Edition

Download MobaXterm Home Edition (current version):

 [MobaXterm Home Edition v21.5
\(Portable edition\)](#)

 [MobaXterm Home Edition v21.5
\(Installer edition\)](#)

Download previous stable version: [MobaXterm Portable v21.4](#) [MobaXterm Installer v21.4](#)

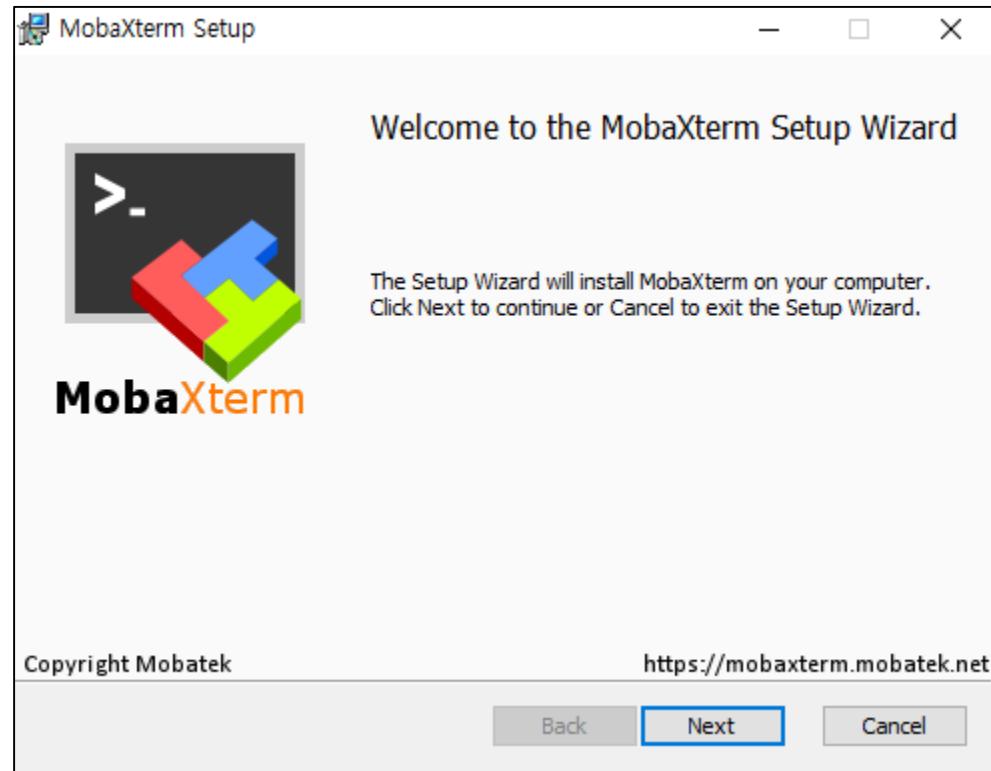
You can also get early access to the latest features and improvements by downloading MobaXterm Preview version:

[MobaXterm Preview Version](#)

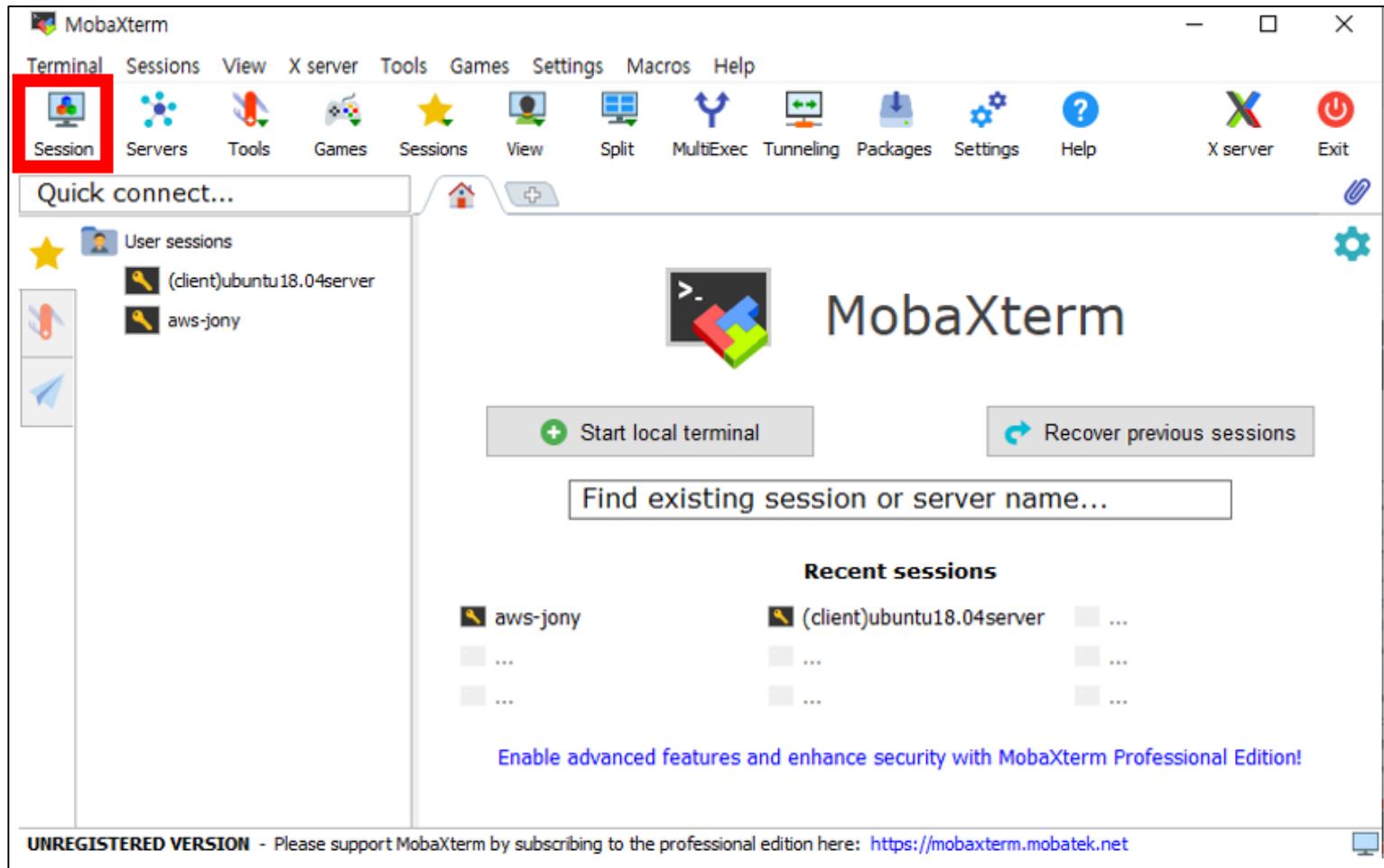
By downloading MobaXterm software, you accept [MobaXterm terms and conditions](#)

You can download the third party plugins and components sources [here](#)

설치 진행



내 Windows PC에서 AWS 인스턴스에 연결해보자



인스턴스 (1/2) 정보

G 연결 인스턴스 상태 ▾

인스턴스 시작 ▾

검색

퍼블릭 IPv4 ... 탄력적 IP

42.26.24 42.26.24

연결하고 싶은 AWS 인스턴스 퍼블릭 IP

Session settings

SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial File Shell Browser M

유저네임은 ubuntu

Basic SSH settings

Remote host: 42.26.24 Specify username: ubuntu Port: 22

Advanced SSH settings Terminal settings Network settings Bookmark settings

X11-Forwarding Compression Remote environment: Interactive shell

Execute command: Do not exit after command ends

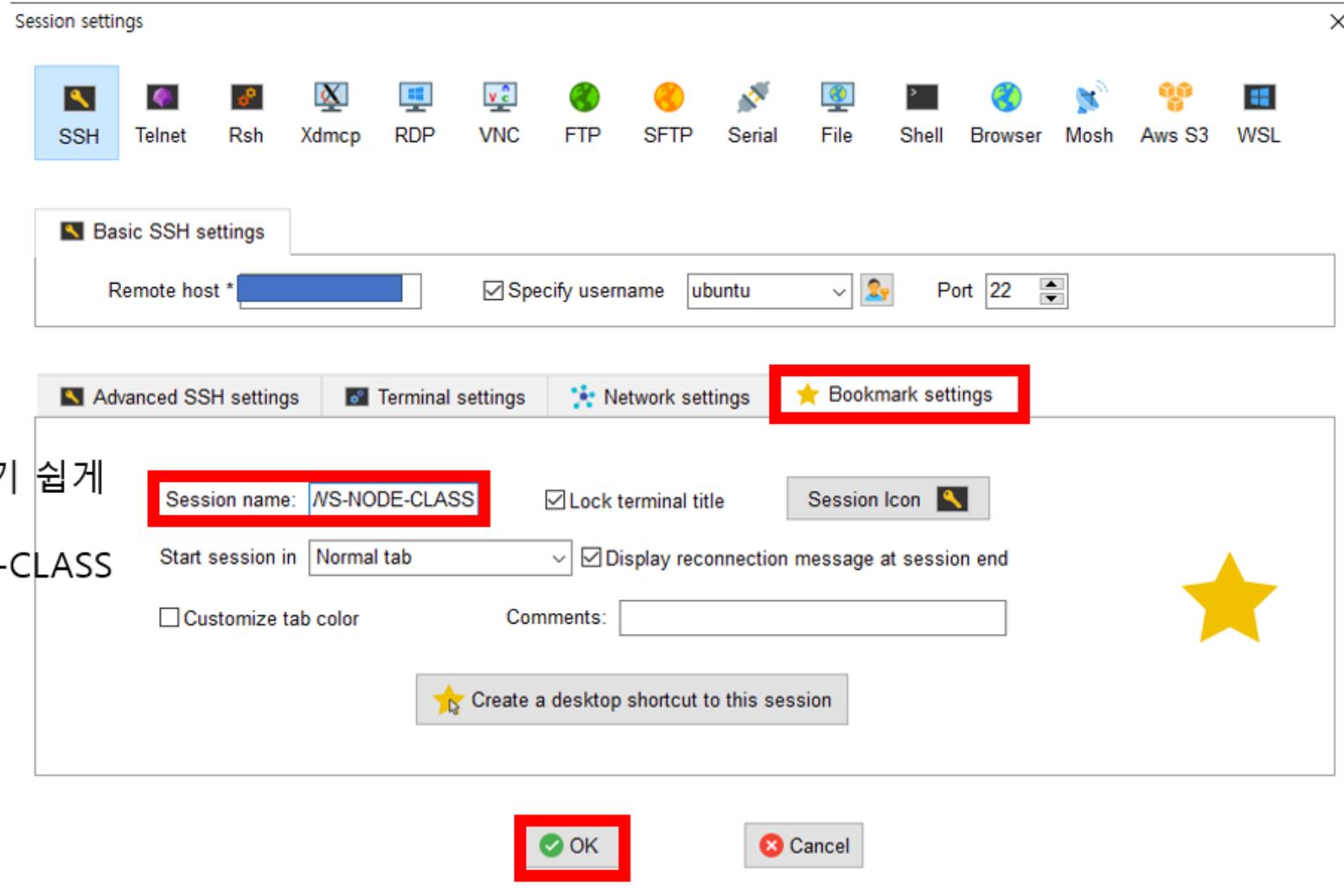
SSH-browser type: SFTP protocol Follow SSH path (experimental)

Use private key C:\Users\mincoding\Documents\ssh-key.pem Adapt locales on remote server

Execute macro at session start: <none>

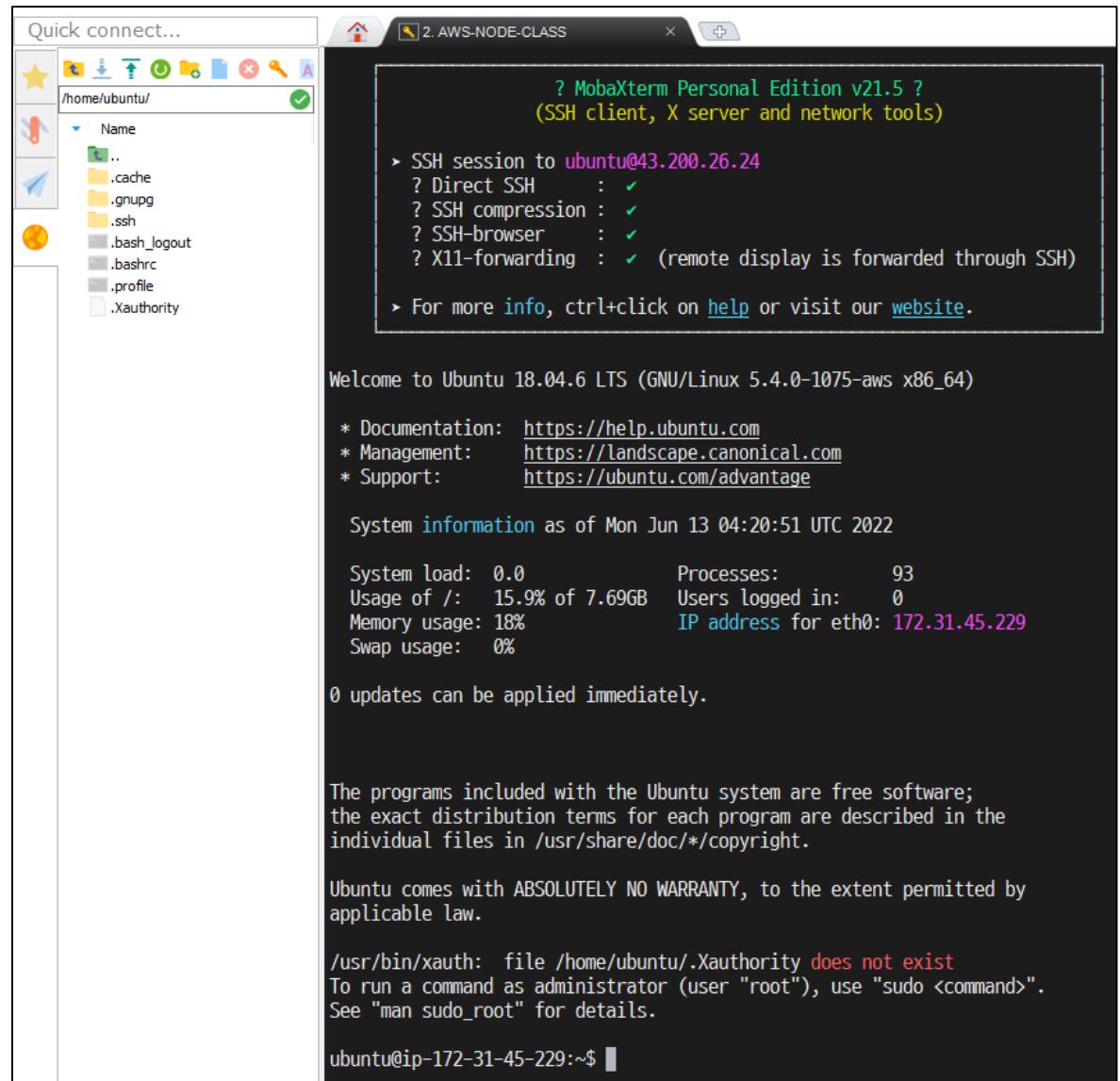
OK Cancel

아까 저장한 Key



접속 성공

- Windows에서, AWS Instance에 원격접속 성공

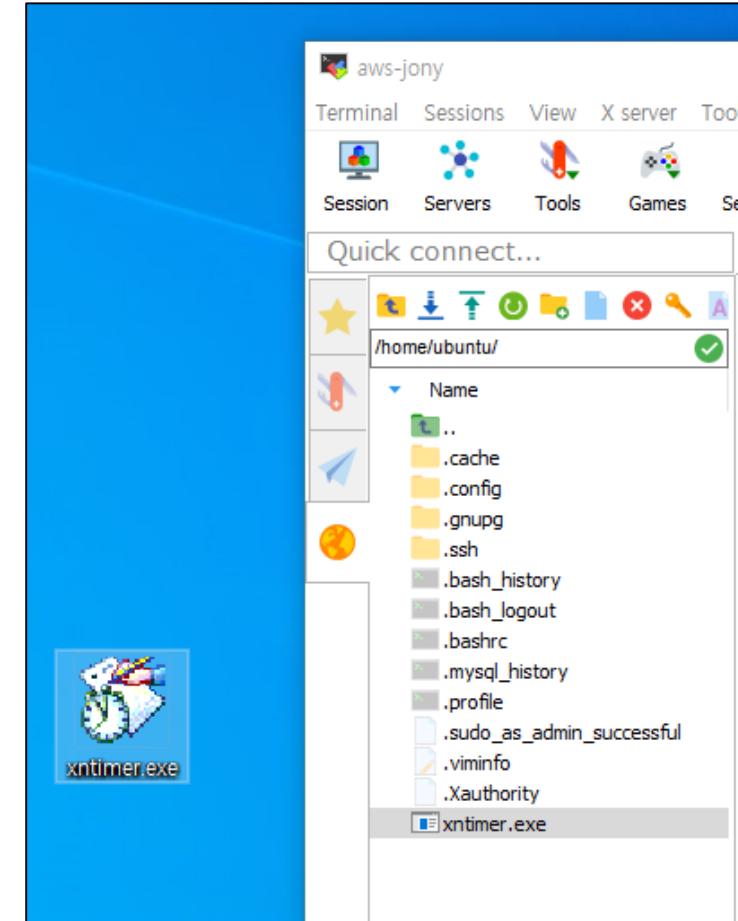


한국 시간대인 KST로 변경

- \$ sudo timedatectl set-timezone Asia/Seoul
- \$ date

파일 업로드

- 서버에 파일을 업로드하고, 다운로드해보자.
- MobaXterm 왼쪽 네비게이션에
- 마우스 드래그로 업로드/다운로드 가능



4장. 기본적인 리눅스 명령어

챕터의 포인트

- 기본 명령어
- 파일관리 명령어

기본 명령어

ls -al

- a 옵션 : all, 숨김파일까지 출력
- l 옵션 : list, 리스트 형태로 상세 보기

주의사항

- !!이라고 입력해도 되지만, ls -al을 쓰자.
- !!은 임베디드 리눅스에서 지원안하는 경우 많음

```
ubuntu@ip-172-31-45-229:~$ ls -al
total 44
drwxr-xr-x 6 ubuntu ubuntu 4096 Jun 13 04:39 .
drwxr-xr-x 3 root   root   4096 Jun 13 02:11 ..
-rw----- 1 ubuntu ubuntu   62 Jun 13 04:20 .Xauthority
-rw-r--r-- 1 ubuntu ubuntu  220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Apr  4 2018 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Jun 13 04:20 .cache
drwx----- 3 ubuntu ubuntu 4096 Jun 13 04:20 .gnupg
-rw-r--r-- 1 ubuntu ubuntu  807 Apr  4 2018 .profile
drwx----- 2 ubuntu ubuntu 4096 Jun 13 02:11 .ssh
drwxrwxr-x 5 ubuntu ubuntu 4096 Jun 13 04:39 .vscode-server
-rw-rw-r-- 1 ubuntu ubuntu 183 Jun 13 04:39 .wget-hsts
ubuntu@ip-172-31-45-229:~$ █
```

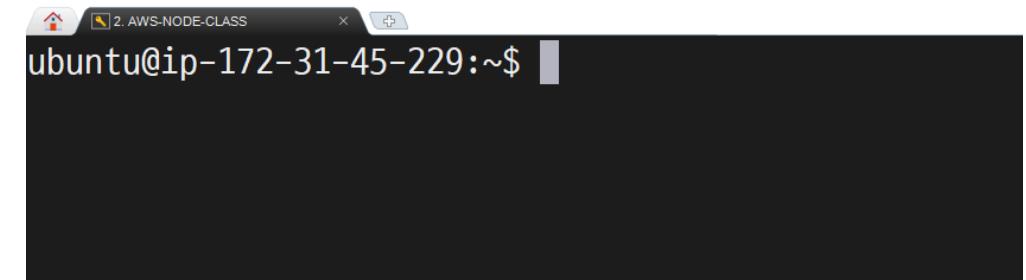
pwd

- 현재 디렉토리 확인
 - 리눅스 CLI 환경에서는 폴더가 아닌 “디렉터리”라는 명칭을 사용

```
ubuntu@ip-172-31-45-229:~$ pwd  
/home/ubuntu
```

clear

- 화면 지우기



cd

- **cd ..**
 - 이전 디렉토리로 이동하기
- **cd “디렉토리 이름”**
 - 해당 디렉토리로 이동하기
- **cd ~**
 - 홈 디렉토리로 이동
- **cd -**
 - 이전 디렉토리로 되돌아가기



```
ubuntu@ip-172-31-45-229:~$ ls
ubuntu@ip-172-31-45-229:~$ cd ..
ubuntu@ip-172-31-45-229:/home$ ls
ubuntu
ubuntu@ip-172-31-45-229:/home$ cd ..
ubuntu@ip-172-31-45-229:$ ls
bin etc initrd.img.old lost+found opt run srv usr vmlinuz.old
boot home lib media proc sbin sys var
dev initrd.img lib64 mnt root snap tmp vmlinuz
ubuntu@ip-172-31-45-229:$ █
```

복사 붙여넣기 단축키

- 터미널 창에서는 Ctrl + C / Ctrl + V 가 아니다.
 - Ctrl + Insert (복사)
 - Shift + Insert (붙여넣기)
 - 임베디드 리눅스 환경, 모두 동일
 - 윈도우에서도 사용 가능

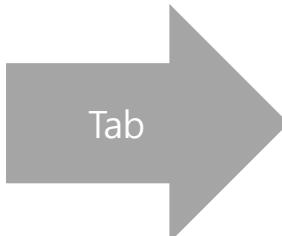
리눅스 명령어와 친해지기(제한시간 1분)

1. home directory 이동 후, 전체 디렉토리 확인
2. 현재 디렉토리가 어디인지 확인한다.
3. root directory 이동 후, 전체 디렉토리 확인
4. 화면을 모두 지운다.
5. /usr/bin 으로 이동한다.
6. 현재 디렉토리에 있는 파일 중 yes 파일의 생성 날짜 정보를 확인한다. (ls 사용)
7. 화면을 모두 지운다.
8. 이전 디렉토리로 이동한다.
9. 현재 디렉토리가 어디인지 출력한다.
10. 터미널을 종료한다. (exit 명령어)

Tab을 누르면 타이핑을 다 안해도 자동 완성이 된다.

- 후보가 하나일때는 Tab 한번 누르면 자동완성이 됨
- 후보가 여러개일때는 Tab Tab 눌러야 후보 목록이 나옴

```
ubuntu@ip-172-31-45-229:/usr/bin$ cd /
ubuntu@ip-172-31-45-229:$ ls
bin  initrd.img   media  run   tmp
boot initrd.img.old  mnt   sbin  usr
dev   lib          opt    snap  var
etc   lib64        proc   srv   vmlinuz
home lost+found   root   sys   vmlinuz.old
ubuntu@ip-172-31-45-229:$ ls l
lib/      lib64/    lost+found/
ubuntu@ip-172-31-45-229:$ ls lo█
```



```
ubuntu@ip-172-31-45-229:/usr/bin$ cd /
ubuntu@ip-172-31-45-229:$ ls
bin  initrd.img   media  run   tmp
boot initrd.img.old  mnt   sbin  usr
dev   lib          opt    snap  var
etc   lib64        proc   srv   vmlinuz
home lost+found   root   sys   vmlinuz.old
ubuntu@ip-172-31-45-229:$ ls l
lib/      lib64/    lost+found/
ubuntu@ip-172-31-45-229:$ ls lost+found/█
```

```
ubuntu@ip-172-31-45-229:$ cd /
ubuntu@ip-172-31-45-229:$ ls
bin  initrd.img   media  run   tmp
boot initrd.img.old  mnt   sbin  usr
dev   lib          opt    snap  var
etc   lib64        proc   srv   vmlinuz
home lost+found   root   sys   vmlinuz.old
ubuntu@ip-172-31-45-229:$ ls l█
```



```
ubuntu@ip-172-31-45-229:$ cd /
ubuntu@ip-172-31-45-229:$ ls
bin  initrd.img   media  run   tmp
boot initrd.img.old  mnt   sbin  usr
dev   lib          opt    snap  var
etc   lib64        proc   srv   vmlinuz
home lost+found   root   sys   vmlinuz.old
ubuntu@ip-172-31-45-229:$ ls l
lib/      lib64/    lost+found/
ubuntu@ip-172-31-45-229:$ ls l█
```

파일관리 명령어

파일을 다루는 기본 명령어

- 파일 생성 / 삭제
- 디렉토리 생성 / 삭제
- 파일 이동 / 이름 변경
- 파일 복사

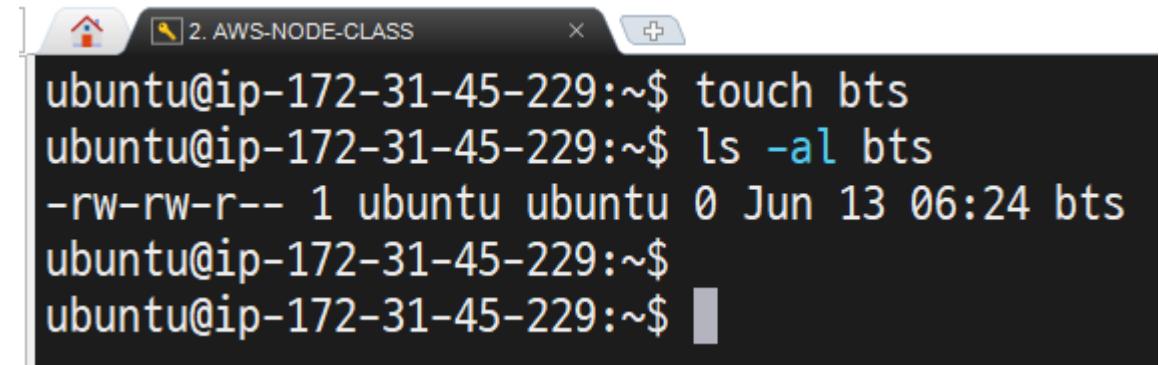
touch [파일명]

- 새로운 빈 파일을 생성한다.
- 이미 있는 파일이라면, 변경된 시간을 현재 시간으로 업데이트 한다.
- 예시 : touch ./bts

./ 는 현재 디렉토리를 의미한다.
생략해도 되지만,
가독성 측면에서 파일명임을 나타낸다.

rm [파일명]

- 파일을 삭제한다.
- 예시 : rm ./bts



```
ubuntu@ip-172-31-45-229:~$ touch bts
ubuntu@ip-172-31-45-229:~$ ls -al bts
-rw-rw-r-- 1 ubuntu ubuntu 0 Jun 13 06:24 bts
ubuntu@ip-172-31-45-229:~$
```

홈 디렉토리에서 실습해보기

a, b, c 파일을 생성하자

- TIP : 리눅스는 윈도우와 달리 확장자 개념이 없다.

파일을 각각 touch 해주자.

- 수정시간을 현재 시간으로 업데이트 한다.

파일을 삭제하자.

mkdir “디렉토리명”

- 디렉토리가 생성된다.
 - 예시 : mkdir a b c : a, b, c 디렉토리가 생성된다.
- -p 옵션 : 디렉토리 하위메뉴 까지 모두 한꺼번에 생성
 - 예시 : mkdir -p ./aaa/bbb/ccc/ddd

```
ubuntu@ip-172-31-45-229:~$ rmdir ./aaa  
rmdir: failed to remove './aaa': Directory not empty  
ubuntu@ip-172-31-45-229:~$ █
```

rmdir “디렉토리명”

- 디렉토리를 삭제한다.
- 디렉토리 안에 파일이 있으면 삭제가 되지 않는다.
- rmdir 대신, rm -r “경로” 를 사용하면 디렉토리 내부 파일까지 모두 삭제 가능
 - rmdir 대신 사용하기 더 간편한 rm -r 을 사용하자.

```
ubuntu@ip-172-31-45-229:~$ rm -r ./aaa  
ubuntu@ip-172-31-45-229:~$ █
```

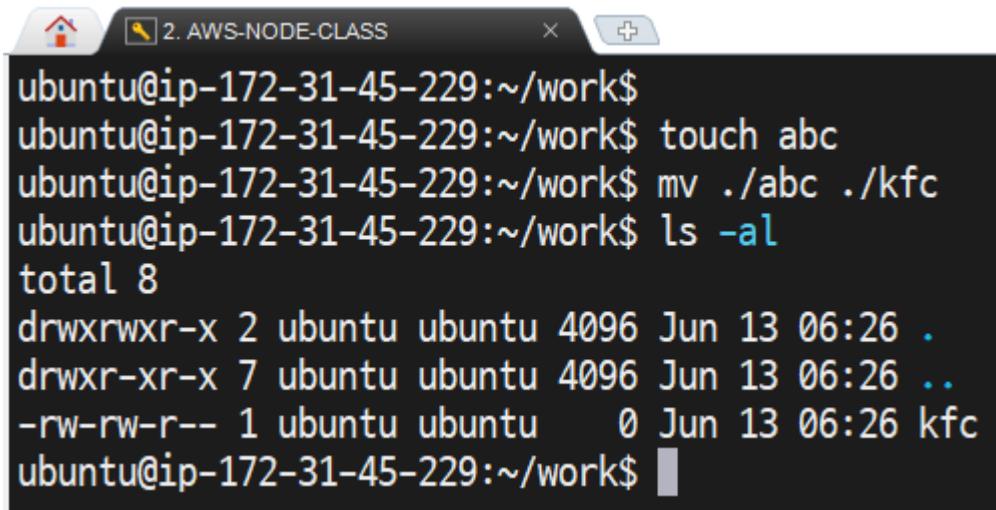


생성 삭제를 연습 (제한시간 1분)

1. 홈 디렉토리로 이동 후 현재 디렉토리 확인하기
2. 현재 디렉토리에서 ./a/b/c 디렉토리 생성
3. ./a/b/c 디렉토리로 이동하기
4. 빈 파일 두 개 “kfc”, “bts” 생성
5. 생성된 파일 확인하기
6. 상위 디렉토리로 이동하기
7. 빈 파일 “mc” 하나 생성
8. 상위 디렉토리로 이동하기
9. 현재 디렉토리 내부에 있는 파일 모두 삭제 (`rm -r ./*` : 현재 디렉토리에서 전체 삭제)
10. 홈 디렉토리로 이동
11. a 디렉토리 삭제
12. 화면 지우기

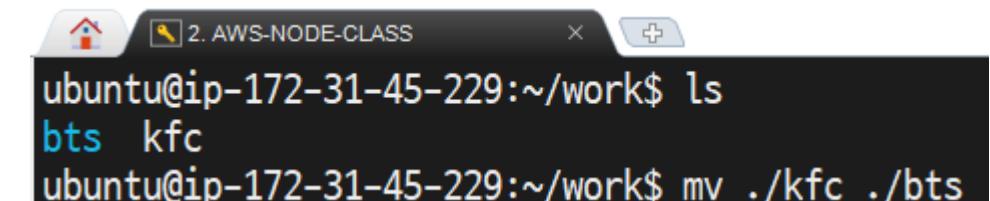
mv 명령어

- 이동 : mv [파일] [옮길 곳]
- 이름변경 : mv [파일] [파일이름]



```
ubuntu@ip-172-31-45-229:~/work$ touch abc
ubuntu@ip-172-31-45-229:~/work$ mv ./abc ./kfc
ubuntu@ip-172-31-45-229:~/work$ ls -al
total 8
drwxrwxr-x 2 ubuntu ubuntu 4096 Jun 13 06:26 .
drwxr-xr-x 7 ubuntu ubuntu 4096 Jun 13 06:26 ..
-rw-rw-r-- 1 ubuntu ubuntu     0 Jun 13 06:26 kfc
ubuntu@ip-172-31-45-229:~/work$
```

mv로 이름 바꾸는 방법



```
ubuntu@ip-172-31-45-229:~/work$ ls
bts  kfc
ubuntu@ip-172-31-45-229:~/work$ mv ./kfc ./bts
```

mv로 옮기는 법

이름 변경 및 파일이동 (제한시간 1분)

1. work directory 하나 생성 후 이동
2. mc, bbq, kfc 파일 생성
3. shop 디렉토리 생성
4. mc와 bbq를 shop 디렉토리로 옮김
5. kfc를 bts로 파일이름 변경
6. shop 디렉토리로 이동
7. bbq를 상위 디렉토리로 옮기기
8. mc를 상위 디렉토리에 옮기면서 파일이름을 good 으로 수정
9. ls -al 로 work 디렉토리 내용 확인
10. work directory 삭제

```
ubuntu@ip-172-31-45-229:~/work$ ls  
bbq  bts  good  shop
```

work directory
최종 결과

cp 명령어

- cp [파일] [경로] : 한 파일을 경로로 복사하기
- -r 옵션 : 디렉토리 복사하기

디렉토리 복사는 rm과 마찬가지로 -r 옵션을 붙이자.

미션 수행하기 (제한시간 1분)

1. abc1 파일 생성
2. abc1 파일을 사용하여 abc2 파일 복사
3. chocho/haha 디렉토리 생성
4. abc1, abc2 를 chocho 로 이동시키기
5. chocho 를 tacktack 으로 디렉토리 복사
6. chocho 를 tacktack 안으로 이동시키기
7. tacktack 디렉토리 제거하기

미션 1

- ~/kfc/bbq/bts/ssafy 디렉토리를 한꺼번에 생성
- 해당 디렉토리로 이동 후, a.log 파일 생성
- 파일 이름을 c.log 로 변경 후, 해당 파일을 ~/ 로 copy

미션 2

- ~/kfc/bbq/bts 안에 있는 ssafy 디렉토리 제거 (파일 함께 제거)
- ~/kfc/bbq/ 안에있는 bts 디렉토리를 ~/kfc 에 copy

미션 3

- home directory 에 test.txt 파일 생성
- test.txt 파일을 ~/kfc/bbq로 이동
- ~/c.log 파일을 ~/kfc/bbq로 이동
- ~/kfc 디렉토리 제거

5장. 나의 첫 웹서버

챕터의 포인트

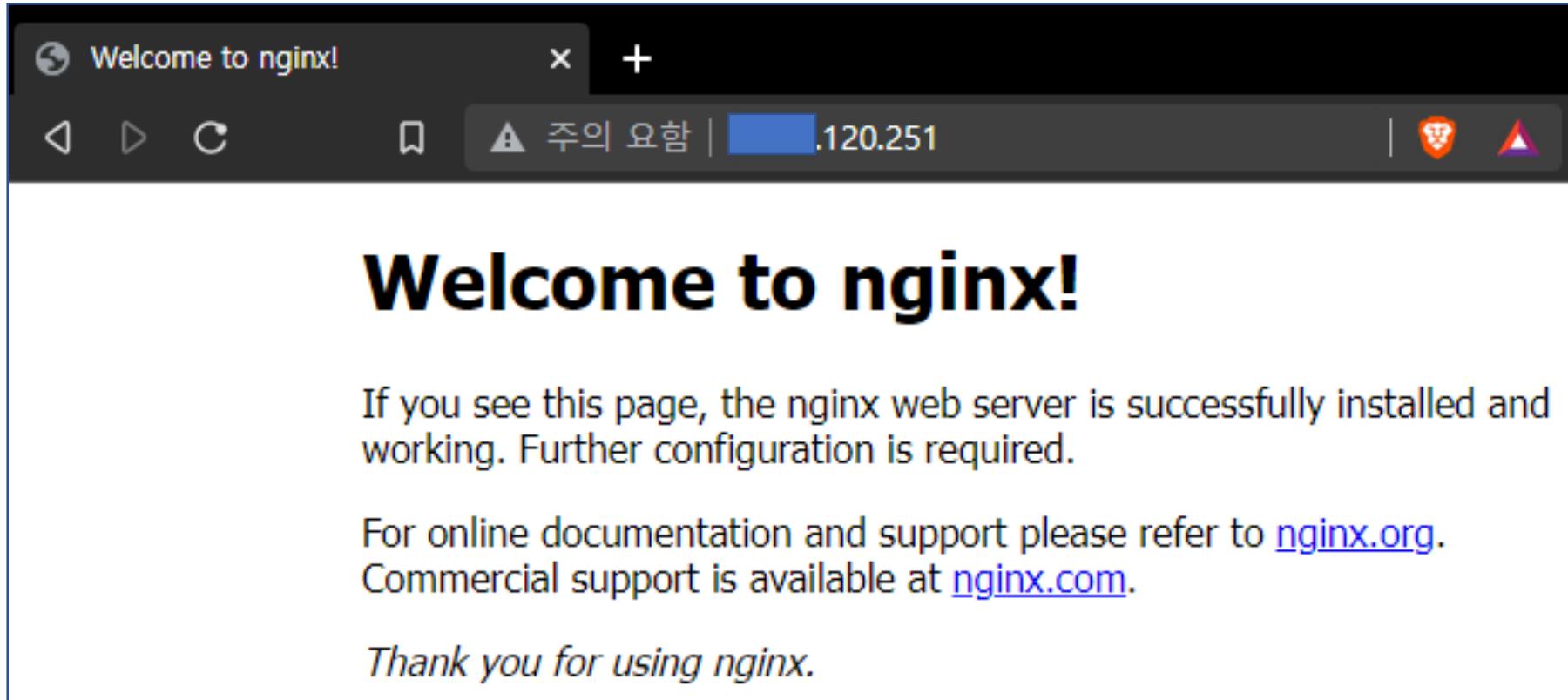
- 간단한 nginx 서버
- 도메인 연결
- 인스턴스 중지와 종료

간단한 nginx 서버

NGINX 서버 구축하기

- 다른 사람의 웹사이트로 도메인 접속 => 웹서버와 친해지기
- MobaXterm 을 사용해 nginx 설치
 - \$ sudo apt update
 - sudo는 관리자 권한을 의미
 - apt(advvanced packaging tool) 로써 리눅스의 패키지 관리 도구
 - \$ sudo apt install nginx
- \$ nginx -v
 - 설치 후 nginx 버전 확인

```
ubuntu@ip-172-31-40-222:~/back-end$ nginx -v
nginx version: nginx/1.14.0 (Ubuntu)
```

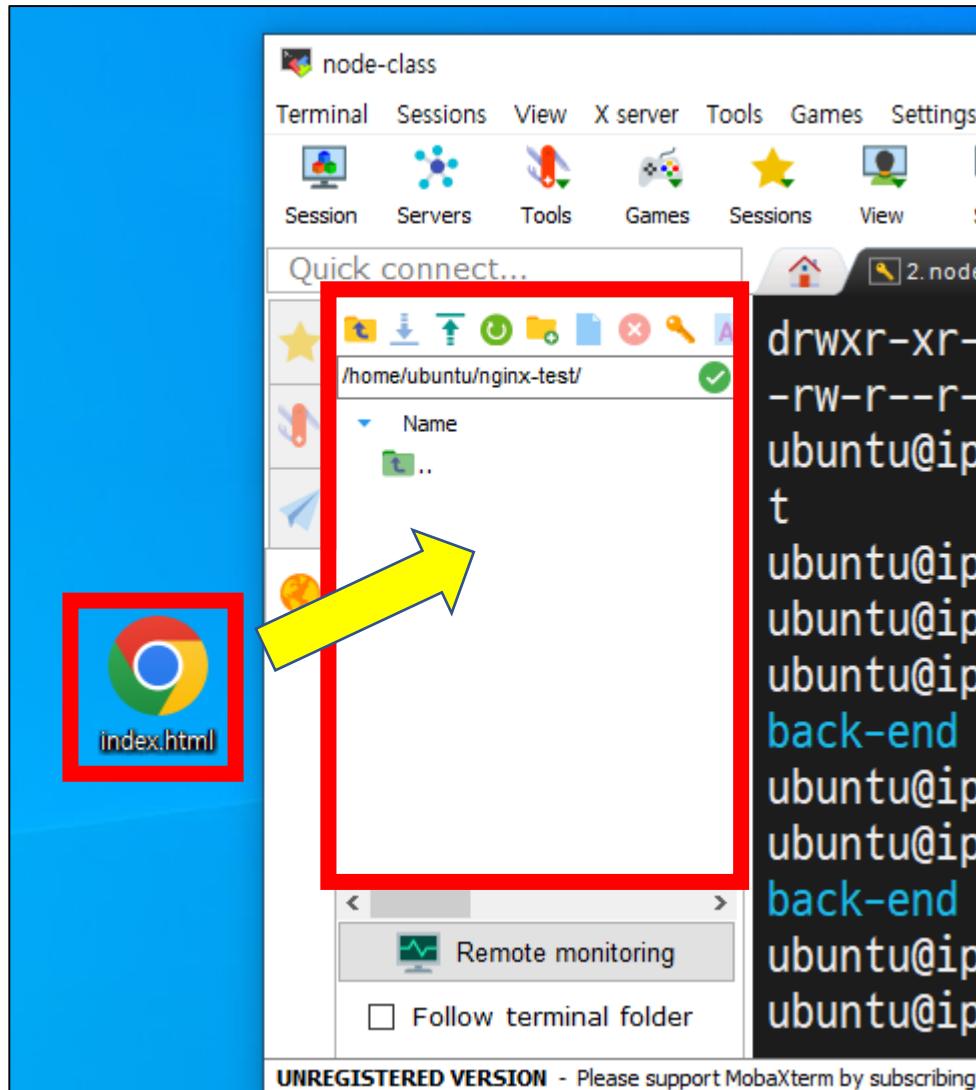


AWS 인스턴스 IP 입력 시, nginx 소개 페이지가 뜬다면 성공

디렉토리 생성

- 홈디렉터리에, 서비스를 위한 디렉터리 하나 생성
- \$ mkdir ~/nginx-test
- \$ cd ~/nginx-test
- 간단한 index.html 을 windows 에서 작성
- pwd로 경로를 파악한 후 복사(Ctrl + Insert)

```
1  <!DOCTYPE html>
2  <html lang="en">
3  | <head>
4  | | <meta charset="UTF-8" />
5  | | <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6  | | <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7  | | <title>nginx 테스트</title>
8  | </head>
9  | <body>
10 | | <h1>nginx 테스트</h1>
11 | </body>
12 </html>
```



드래그 앤 드롭

즉, 윈도우의 파일을 AWS 인스턴스로 이동 !

nano

- 리눅스 용 명령 텍스트 에디터
- 일반 텍스트 에디터와 유사하게 사용 가능

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    #  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    #  
    # include snippets/snakeoil.conf;  
  
    root /var/www/html;  
  
    # Add index.php to the list if you are using PHP  
    index index.php index.html index.htm index.nginx-debian.html;  
  
    server_name _;  
  
    location / {  
        # First attempt to serve request as file, then  
        # as directory, then fall back to displaying a 404.  
    }  
}
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^Y Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo

vi/vim

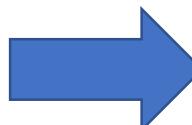
- 임베디드 개발자가 가장 많이 사용하는 텍스트 에디터
- 입문시 난이도가 있는편 (단축키에 익숙해져야 한다.)
- 임베디드 수업 때 본격적으로 사용할 예정



만든 프로젝트 디렉터리를 nginx에 연결

- nginx 설치시 /etc/nginx 라는 디렉터리가 자동으로 생성된다.
- sudo nano /etc/nginx/sites-available/default
 - 기존 /var/www/html 으로 되어있는경로를 생성한 디렉토리로 변경한다.
 - /home/ubuntu/nginx-test
 - 복사한 디렉토리 붙여넣기 (shift + insert)
 - 붙여 넣은 후 Ctrl + O로 작성 후 Ctrl + X로 나가기

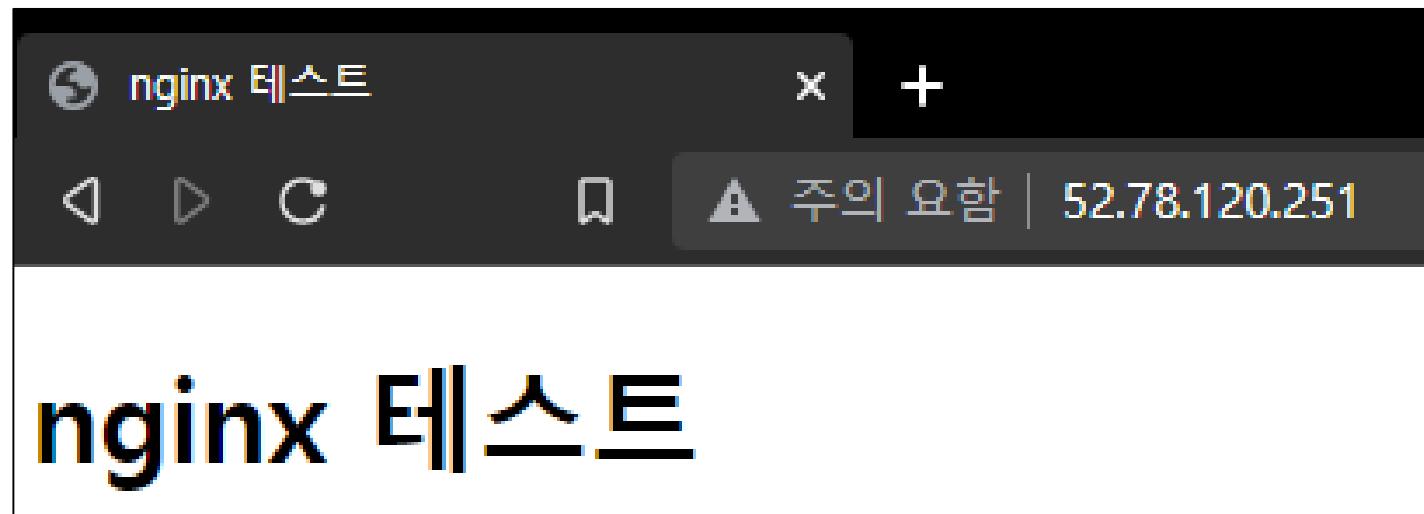
```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    #  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    #  
    #include snippets/ssl-nginx.conf;  
  
    root /var/www/html;  
  
    # Add index.php to the list if you are using PHP  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name _;
```



```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    # SSL configuration  
    #  
    # listen 443 ssl default_server;  
    # listen [::]:443 ssl default_server;  
    #  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    #  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    #  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    #  
    #include snippets/ssl-nginx.conf;  
  
    root /home/ubuntu/nginx-test;  
  
    # Add index.php to the list if you are using PHP  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name _;
```

디렉토리 연결이 끝났다면 명령어 실행

- sudo service nginx reload
- EC2 IP 주소를 브라우저 주소창에 입력해서 확인

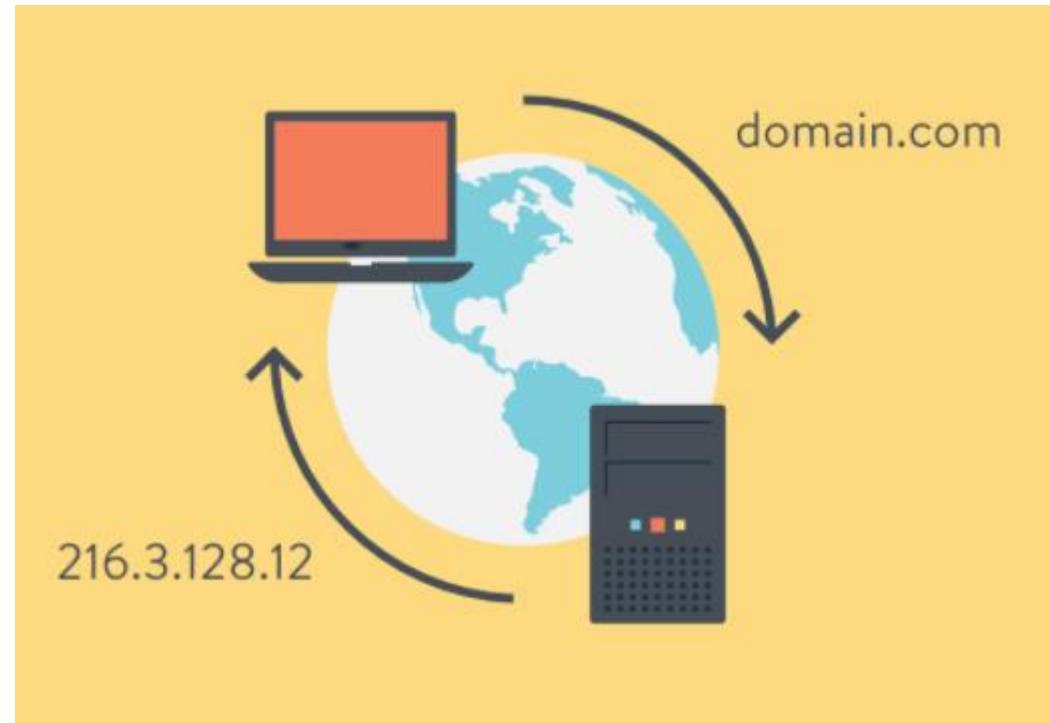


IP 입력 시, 성공적으로 테스트파일이 보인다.

도메인 연결

DNS(Domain Name System)

- 사람이 이해하기 쉬운 도메인 주소를 IP로 변환
- 인터넷상의 전화번호부



무료 도메인 적용하기

- 나의 AWS IP 에 무료 도메인을 적용해보자
- <https://내도메인.한국>
- 회원가입 진행



원하는 키워드를 검색해보고 도메인 선택

일반 도메인 검색

.p-e.kr / .o-r.kr / .n-e.kr / .r-e.kr / .kro.kr /

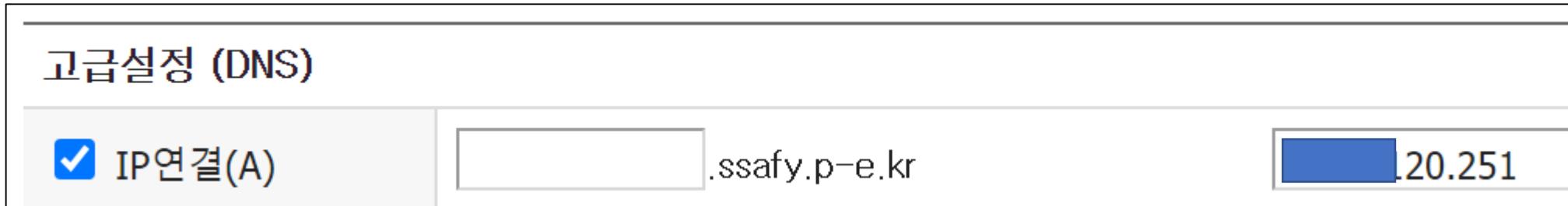
※ 도메인 검색 결과 ※

- | | |
|---|--------------|
| 1 | ssafy.p-e.kr |
| 2 | ssafy.o-r.kr |
| 3 | ssafy.n-e.kr |
| 4 | ssafy.r-e.kr |
| 5 | ssafy.kro.kr |

- | | |
|----------------------------------|------|
| <input type="radio"/> | 등록하기 |
| <input checked="" type="radio"/> | 등록불가 |

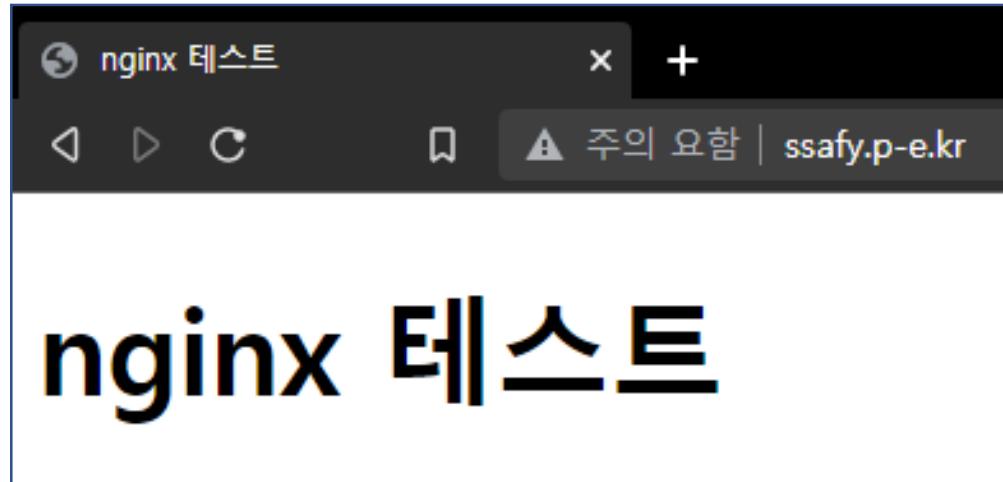
고급설정 AWS

- 인스턴스 아이피를 기입



- 수정하기 누른 후, 브라우저 주소창에
- 부여받은 도메인을 입력해보자





도메인으로 AWS 인스턴스에 접속 성공!



인스턴스 중지와 종료

- EC2 중지 vs EC2 종료

- 종료: 완전 삭제
- 정지: 사용중지, 과금중지

- 탄력적 IP

- 탄력적 IP를 설정하면 인스턴스를 중지한 후 재시작해도 IP가 변하지 않는다.
- 사용하지 않을때는 중지 버튼을 누르자

인스턴스 종료 = 과금 종료

- 목록에 없어지지 않아도,
종료됨이라고 뜨면 과금종료
- 목록은 차후에 자동으로 삭제됨

인스턴스 (1/6) 정보

Name	인스턴스 ID	인스턴스 상태
–	i-0583818c7bd377d3f	실행 중
–	i-095f9a95130bb2b72	실행 중
–	i-03d158422e294fe60	실행 중
–	i-0f928013a775ceb3b	실행 중
–	i-0b0bdbf60c9337623	실행 중
<input checked="" type="checkbox"/>	i-05033a46683d2c073	

인스턴스 시작
템플릿으로 인스턴스 시작
Migrate a server
연결
인스턴스 중지
인스턴스 시작
인스턴스 재부팅
인스턴스 최대 절전 모드
인스턴스 종료

세부 정보 | 보안 | 네트워킹 | 스토리지

▼ 인스턴스 요약 정보

탄력적 IP 유의 사항

- 인스턴스 종료 후 탄력적 IP 미사용 시 요금이 부과
- 해당 탄력적 IP 갱신 or 제거 하기(릴리스 작업)

The screenshot shows the AWS Management Console interface for managing elastic IPs. On the left, a sidebar menu under '네트워크 및 보안' (Network & Security) has '탄력적 IP' (Elastic IP) selected, indicated by a red box. The main content area is titled '탄력적 IP 주소 (1/1)' and shows a single entry in a table:

Name	할당된 IPv4 주소	유형
-	■■■.120.251	퍼블릭 IP

A red box highlights the IP address '■■■.120.251'. Below the table, the text 'IP 클릭' (Click IP) is displayed.

탄력적 IP 릴리스

- 연결 해제 이후 릴리즈버튼 활성화
- 해제 후 릴리스



나의 포트폴리오 사이트 자랑하기

- **인스턴스 삭제** 후 모든과정을 처음부터 다시 진행하기
- WEB PROJECT 첫 주간 마지막날에 진행한 PJT를 서버에 업로드 하기
- 업로드가 완료되었다면 친구의 포트폴리오 페이지에 도메인 접근하기

About



David. Moms
IoT Developer
(Web + Embedded)

안녕하세요. 언제나 달려가는 개발자 David.Moms 입니다.
함께 일하시고 싶으시다면 언제든 연락주셔도 됩니다.

010-9360-2485
inho.choi@mincoding.co.kr
경기도 성남시 분당구 정자동
mincoding.co.kr

6장. Node.js 시작

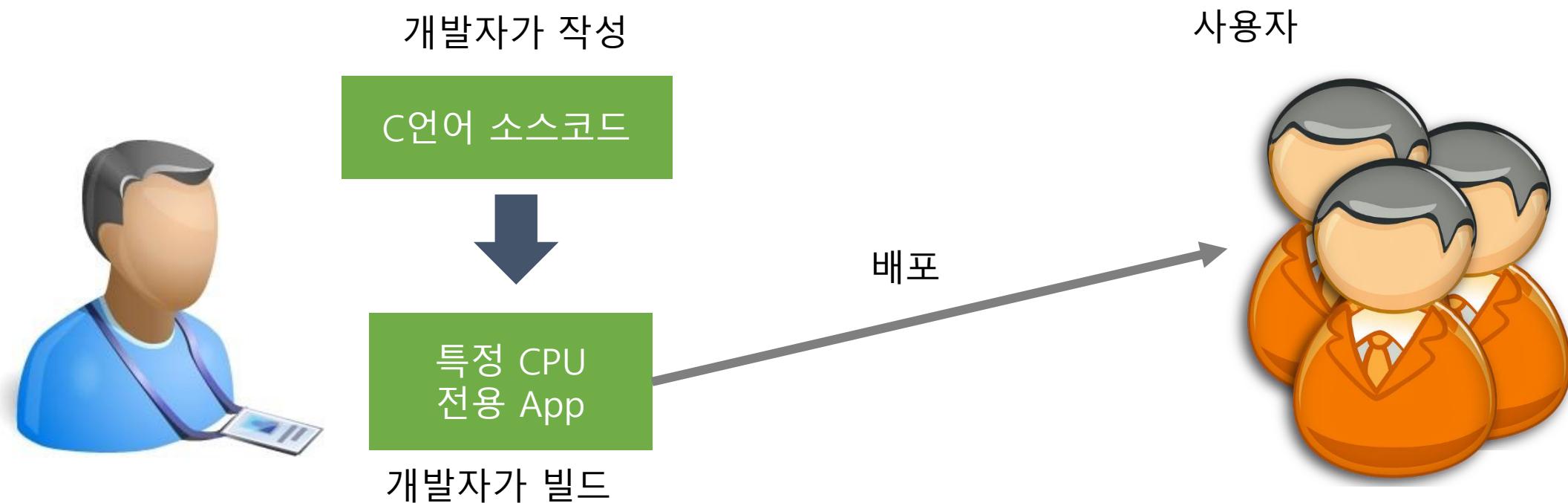
챕터의 포인트

- Node.js 이해
- Node.js 설치하기
- Node.js 활용 예
- 모듈

Node.js 이해

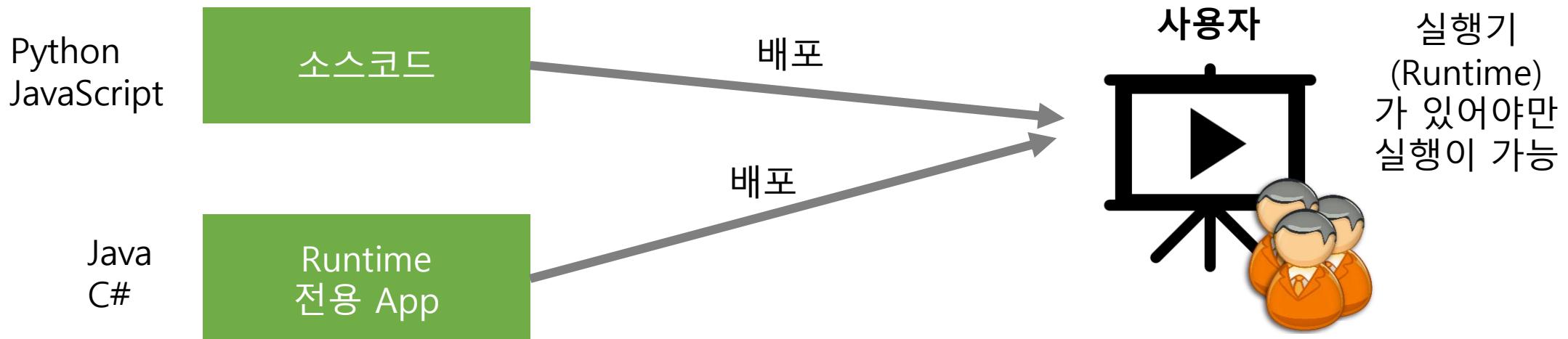
컴파일 언어

- 기존 컴파일 언어는 CPU가 알아들을 수 있는 2 진수 명령어로 번역 후, 수행하는 방식
- 번역 완료된 2 진수 명령어들은, 실행기 없이 App 동작 가능하다.



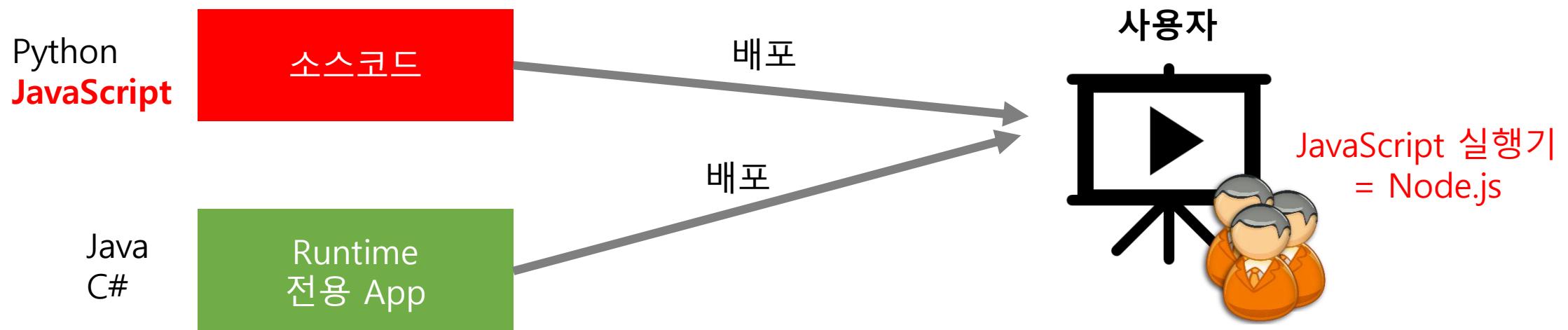
Runtime Environment

- 실행에 필요한 것들은 모아둔 환경을 “Runtime 환경”이라고 하며, 줄여서 Runtime이라고 함
- Runtime은 하나의 Run System이며, “실행기”이다.
- 특정 언어는 실행기가 있어야만, App 동작이 가능하다.



Node.js

- JavaScript의 Runtime (실행기, Runtime Environment)



JavaScript를 Python 처럼 사용할 수 있다.

- JavaScript는 웹브라우저 전용 언어가 아니다.
- Node.js라는 Runtime만 설치 해 두면, JavaScript로 만든 App들을 사용 가능 함

Python Source code

```
1 from sympl import (
2     TimeDifferencingWrapper, NetCDFMonitor)
3 import climt
4 from datetime import timedelta
5
6 # Define model timestep in minutes
7 model_timestep = timedelta(minutes=1)
8
9 # Create Components
10 radiation = climt.RRTMGLongwave()
11 convection = climt.EmanuelConvection()
12 boundary_layer = TimeDifferencingWrapper(
13     climt.SimplePhysics())
14 time stepper = GFSDynamicalCore(
15     [radiation, convection, boundary_layer])
```

Runtime 이름 : Python

JavaScript Source code

```
wrapper = function(tag){
    return function(v){return '<' + tag + '>' + v}
}

initArr = function(arr){
    return Array.isArray(arr) ? arr : []
}

var people = [
    ['김민수', '3', '아들'],
    ['임선영', '39', '엄마'],
    ['김태완', '43', '아빠']
]
```

Runtime 이름 : Node.js

JavaScript로 만든 소스코드 동작 방법

1. 웹브라우저에서 JavaScript 코드 수행

웹브라우저 안에 실행 환경이 구축 되어 있음 == JavaScript Engine

Chrome에는 “V8”이라는 이름의 엔진이 사용되고 있음

2. 웹브라우저 “밖에서” JavaScript 코드 수행 :

Node.js 필요 = JavaScript의 실행기가 필요하기 때문

JavaScript Engine 이상의 동작 환경을 제공 함 (외부 모듈 로딩 / Thread 등 동작 가능)

서버에서, Node.js 이외의 기술 비교

Confidential



기술명	언어	장점	단점
Spring	Java	대한민국에서 가장 많이 쓰임 정부 사업은 오로지 Spring 으로 진행 (전자정부 프레임워크) 객체 지향 개발에 특화됨	난이도 어려움 Java 언어의 난해한 가독성 간단한 프로젝트시엔 지나치게 복잡함
Django	Python	Python 의 간결한 문법. 가독성 좋음 일반적인 사이트에 필요한 모듈 기본 제공 ex) 인증, admin ... 간단한 REST API 를 만들기에 적합 (Django REST Framework)	파이썬의 느린 성능 사용률이 낮음 => 취업 힘듦 Django ORM 강요
Node.js	JavaScript	개발자에게 높은 자유도 부여 ex) ORM 강제 안함 클라이언트와 동일한 언어인 JS 사용 싱글스레드, 비동기에 기반한 빠른 처리속도	자유도가 너무 높아 직접 설계해야 할 일이 많음 JS 의 비동기에 익숙해져야

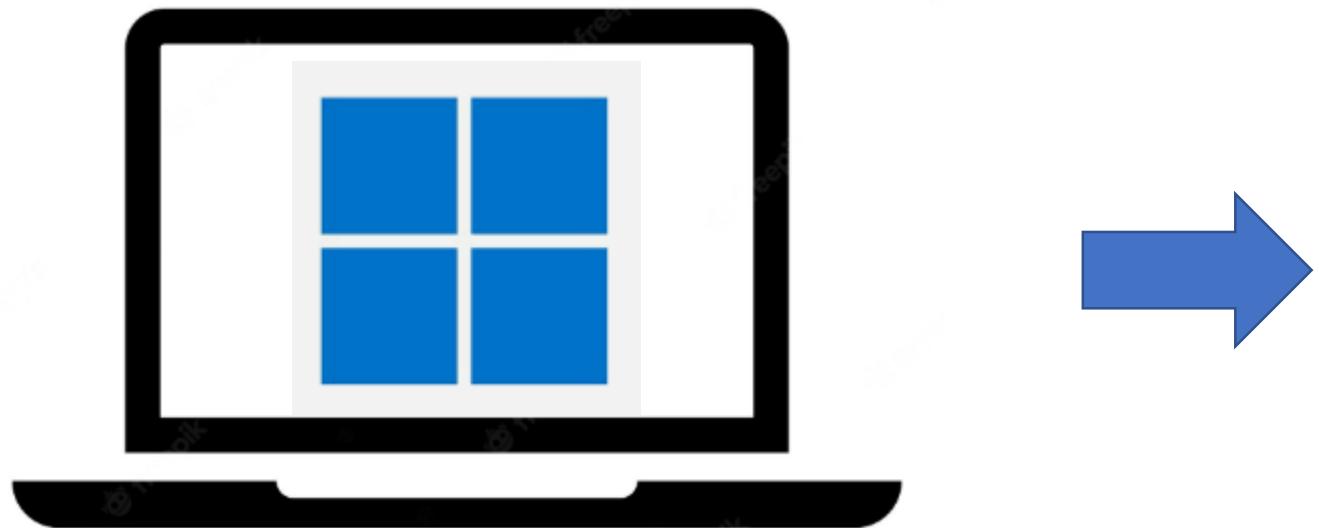
Node.js 설치하기

프리티어용 인스턴스 유형(t2.micro) 스펙

- CPU 1 vCPU
 - 메모리 1 GiB
 - SSD 8 GiB
-
- 라즈베리 파이보다 낮은 스펙으로써 서비스 배포 목적으로 사용해야한다

AWS 서버 활용하기

- 개발은 윈도우, 배포는 AWS
- 즉, 윈도우에서 Node.js 를 설치해 개발 예정



LTS (롱텀서포트)

- 가장 안정적인 버전
- 3년간의 유지보수, 업데이트 지원

Current

- 현재 개발 중인 버전
- 최신 기술 사용 가능

The screenshot shows the official Node.js website. At the top, there's a dark header bar with the Node.js logo and a navigation menu in Korean: 홈 | ABOUT | 다운로드 | 문서 | 참여하기 | 보안 | 뉴스 | CERTIFICATION. Below the header, a green banner states: "Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다." A dark green box contains the text: "Changing the End-of-Life Date for Node.js 16 to September 11th, 2023". In the center, there are two download buttons: "다운로드 - Windows (x64)" with options for "16.15.1 LTS" (highlighted with a red border) and "18.3.0 현재 버전". At the bottom, links for "다른 운영 체제 | 변경사항 | API 문서" are shown for both LTS and Current versions.

Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

Changing the End-of-Life Date for Node.js 16 to September 11th, 2023

다운로드 - Windows (x64)

16.15.1 LTS
안정적, 신뢰도 높음

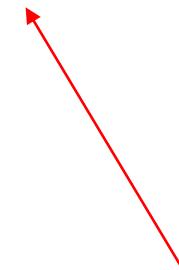
18.3.0 현재 버전
최신 기능

다른 운영 체제 | 변경사항 | API 문서 다른 운영 체제 | 변경사항 | API 문서

LTS 일정은 여기서 확인하세요

명령 프롬프트 창 수행 후 명령어 입력

- node - v
- npm - v



노드 패키지 매니저
파이썬의 pip와 같은 역할을 하는 툴

```
명령 프롬프트
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mincoding>node -v
v16.15.1

C:\Users\mincoding>npm -v
npm WARN config global`--global`, `--local` are
```

Node를 수행 후 Hello World를 출력

- CMD 창에서 node 입력하면 Node Shell이 수행된다.

```
C:\Users\minco>node
```

```
Welcome to Node.js v16.15.1.  
Type ".help" for more information.  
> console.log("Hello world");  
Hello world  
undefined  
>
```

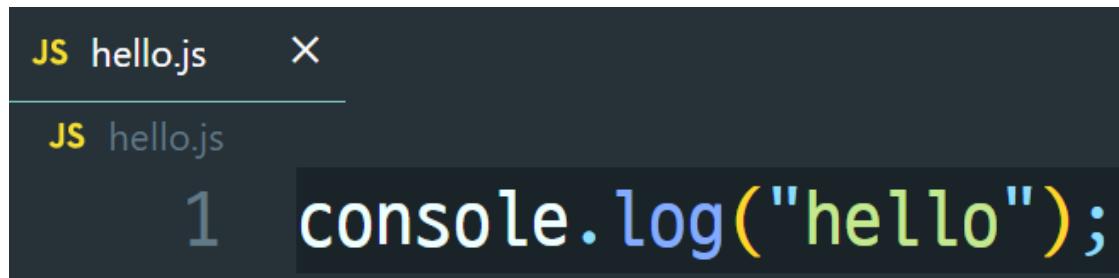
REPL은 Node의 Shell을 뜻한다.

- 사용자 명령을 읽고, 수행 후 출력하는 동작을 반복하는 터미널을 뜻함
- Read(읽고) Eval(해석하고) Print(출력하고) Loop (반복하고)의 줄임말
- Python Shell 과 같은 역할.

```
> let a = 100;
undefined
> console.log(a + 200);
300
undefined
> "HI MINCO" + "HAHAHA"
'HI MINCOHAHAHA'
> const sum = (a, b)=> a + b;
undefined
> sum(35, 40);
75
>
```

vscode에 파일 생성하기

- `hello.js` 파일에 `console.log("hello")` 입력
- vscode 터미널을 켠다
 - 상단 메뉴중 터미널클릭 -> 새 터미널 열기 or Ctrl + Shift + `
 - 터미널에서 `node hello.js` 입력후 결과 확인하기



```
JS hello.js ×  
JS hello.js  
1 console.log("hello");
```



```
C:\Users\mincoding\Desktop\node-test>node hello.js  
hello
```

Javascirpt로 사칙 연산 계산기 만들기

- JavaScript로 사칙연산 계산기를 만든 후,

Node.js로 작동시켜보기

- 파일명: calc.js
- 입력: **하드코딩**
 - ex) a = 2, b = 4
- 출력
 - 오른쪽 참고
- 조건
 - 화살표 함수를 사용해 객체 calc를 생성
 - calc.add(a, b) 형태 식으로 코드를 작성

```
C:\Users\mincoding\Desktop\node-test>node calc.js  
주어진 수: 2 4  
덧셈 결과: 6  
뺄셈 결과: -2  
곱셈 결과: 8  
나눗셈 결과: 0.5
```

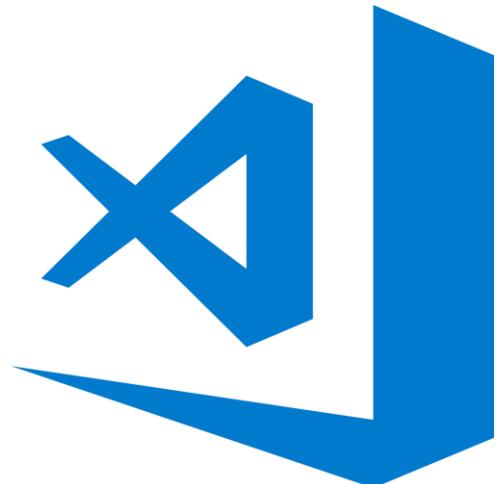
Node.js 활용 예

Node.js란

- 지금까지 경험했듯 Node.js 는 브라우저 바깥에서 사용하는 JavaScript
- Node.js 를 써야만 하는 분야
 - 서버 (REST API, SQL DB 연결)
 - 모바일 앱
 - 데스크톱 앱

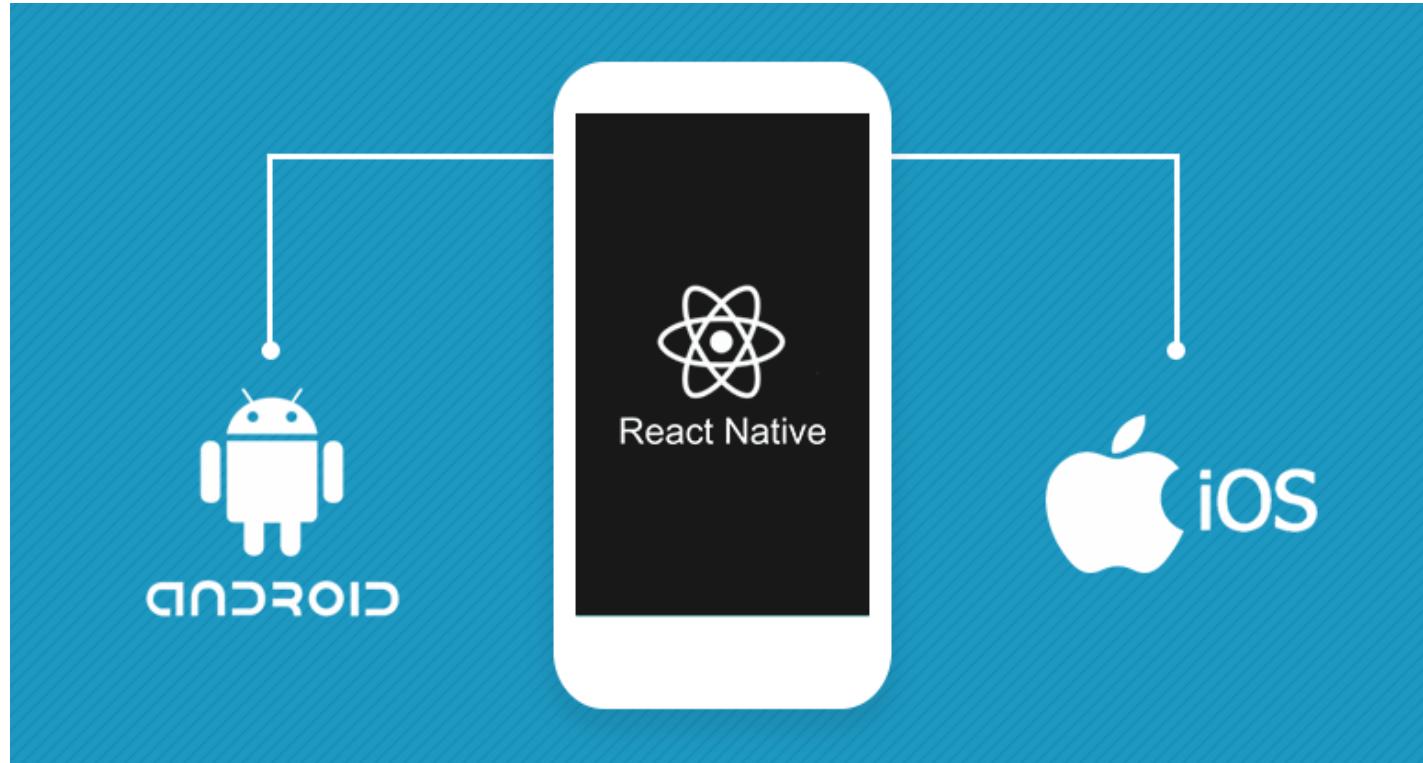
Desktop App 개발

- **Electron Framework** 으로 개발한다.
 - App 1개에 Node.js + Chromium 브라우저를 포함되어 있음
즉, App에 Node.js Runtime이 내포되어, 사용자는 따로 Node.js 설치 필요 없음



Mobile App 개발

- React-Native : iOS + Android App을 동시에 개발 가능
- 인스타, 페이스북이 React-Native로 개발 되었음



Web 개발 : Server Side

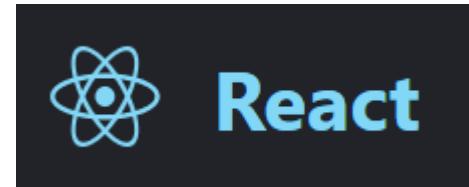
1. Node.js 에는 간단한 웹 서버가 내장되어 있다. (테스트 용도로만 사용)
2. **Express.js** 를 통해 HTTP protocol 로 서버의 JavaScript 메서드를 호출할 수 있음
→ REST API 제작 가능



Express

Web 개발 : Client Side

- Vue.js / React / Angular를 통해 비동기방식 웹사이트를 쉽게 만들 수 있음
- 비동기 웹사이트 제작으로, 서버 부하를 줄이기 위해 사용



수업 예정



모듈

객체로 묶인 맴버들을, 변수로 분해할 수 있는 문법

- destructing - 객체
- name1 / name2 / run 변수에 철수, 영희, run함수가 각각 대입된다.

```
1 const friends = {  
2   name1: "철수",  
3   name2: "영희",  
4   run: (word) => console.log(word, "가자"),  
5 };  
6  
7 const { name1, name2, run } = friends;  
8  
9 console.log(name1); // 철수  
10 console.log(name2); // 영희  
11 run("하핳"); // 하핳 가자
```

hi.js에서 lib.js 모듈을 불러오기

- lib.js : module.exports라는 Object 객체에 넣어둔다.
- hi.js : require 메서드를 사용하면, 해당 파일의 module.exports 객체에 접근할 수 있다.

```
JS lib.js      X  JS hi.js
JS lib.js > ...
1 const friends = {
2   name1: "철수",
3   name2: "영희",
4   run: (word) => console.log(word, "가자"),
5 };
6
7 module.exports = friends;
```

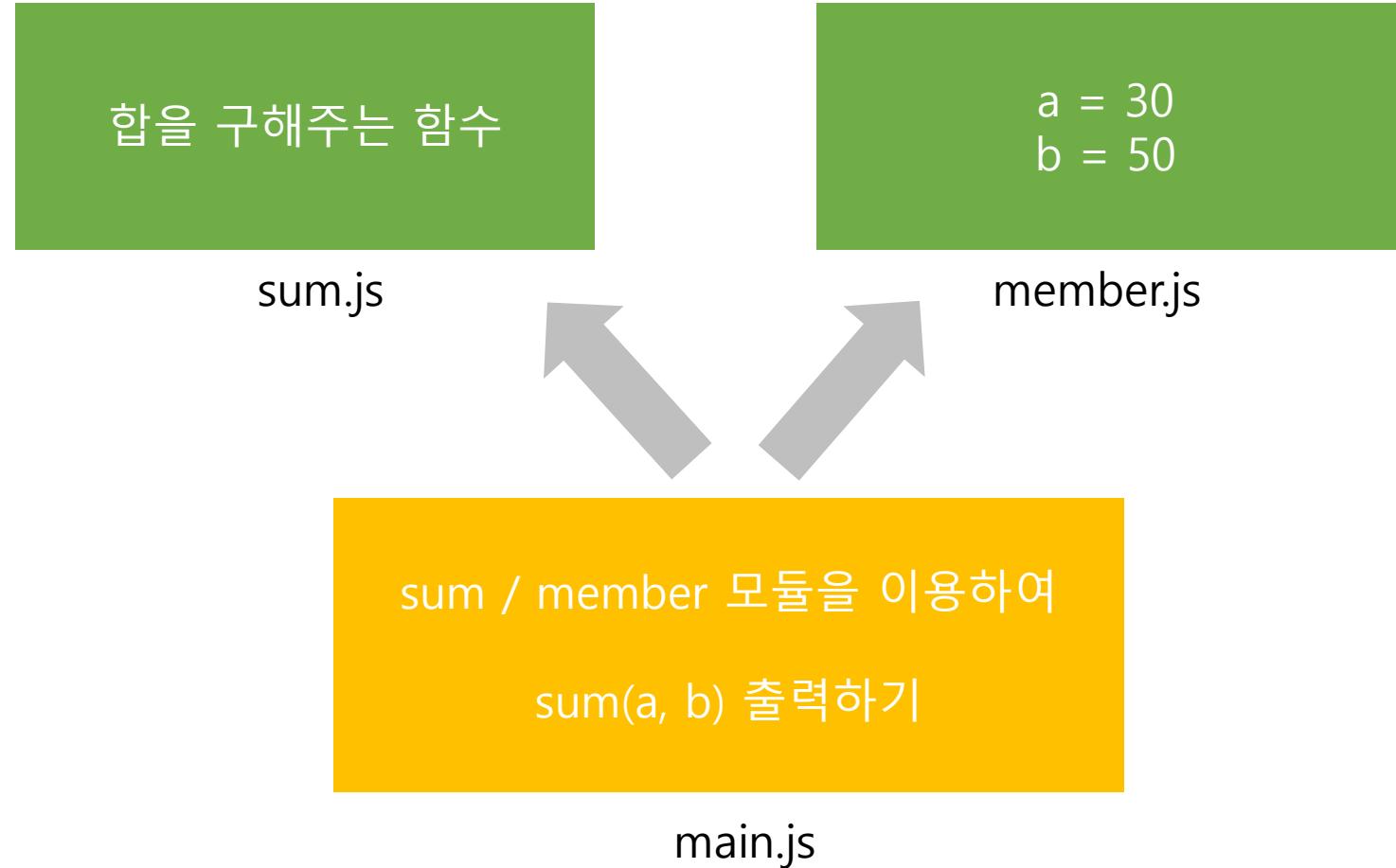
```
1 const { name1, name2 } = require("./lib");
2 ...
3 console.log(name1, name2);
```

js 확장자는 생략이 가능하다.

```
C:\Users\mincoding\Desktop\node-test>node hi.js
철수 영희
```

간단한 모듈을 만들어보자

- sum.js
- member.js
- main.js



7장. 웹 크롤러

챕터의 포인트

- 웹 크롤러 개념
- cheerio
- 크롤링 응용

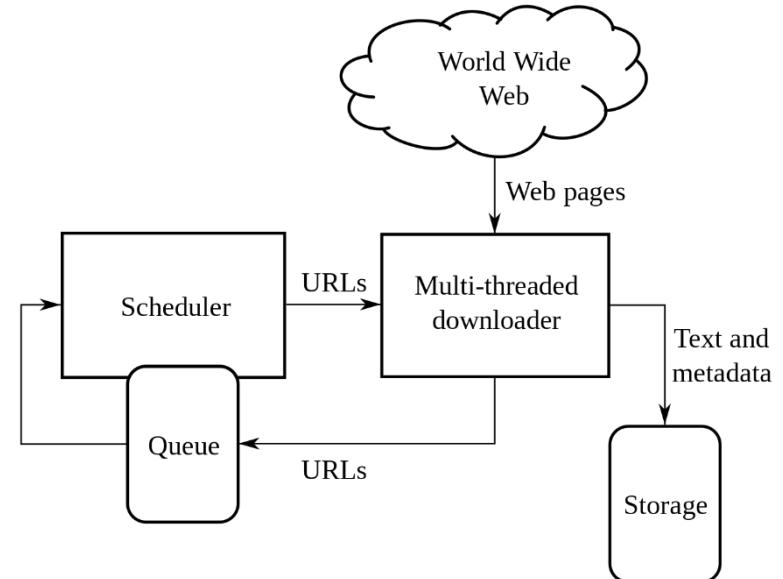
웹 크롤러 개념

Web Scraping

- 웹 사이트에서 데이터를 추출하는 프로그래밍 기술
- 사이트에 있는 정보를 가져오는 행동

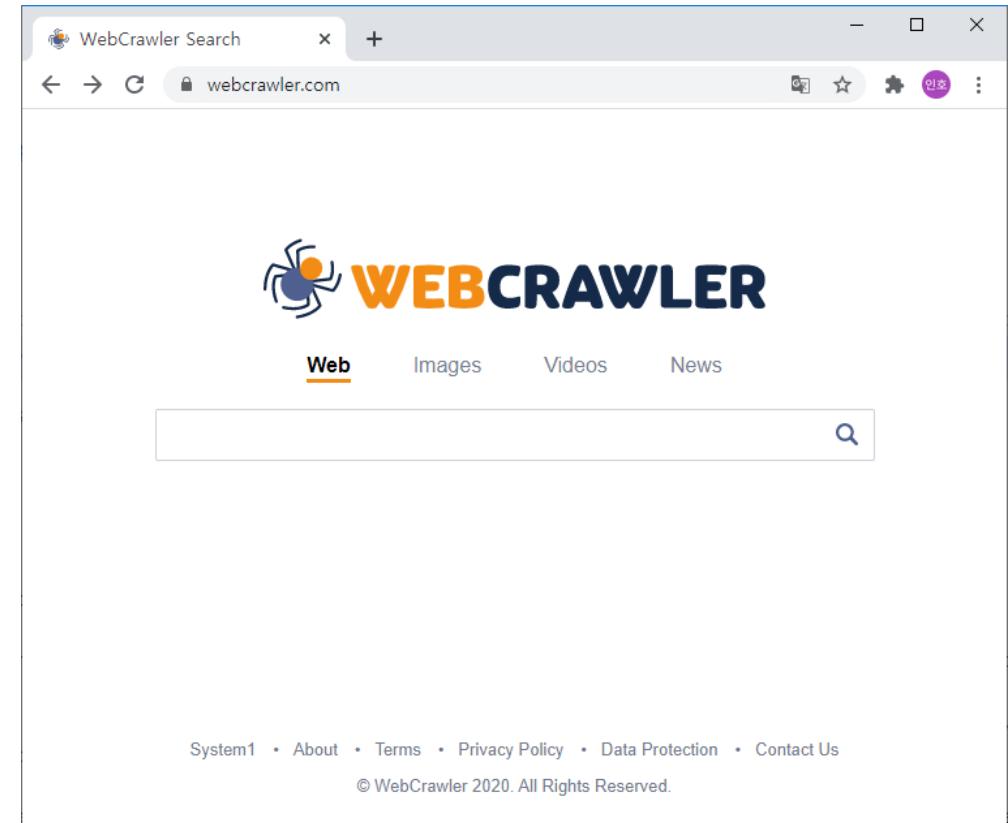
Web Crawler

- URL를 가져온 후, 또 다시 URL에 접속을 반복하여 웹을 탐색하며 Web Scraping을 하는 봇
(Spiderbot이라고도 함)
 - 재귀적으로 웹을 탐색하며 사이트를 가져오는 프로그램
-
- https://en.wikipedia.org/wiki/Web_scraping
 - https://en.wikipedia.org/wiki/Web_crawler



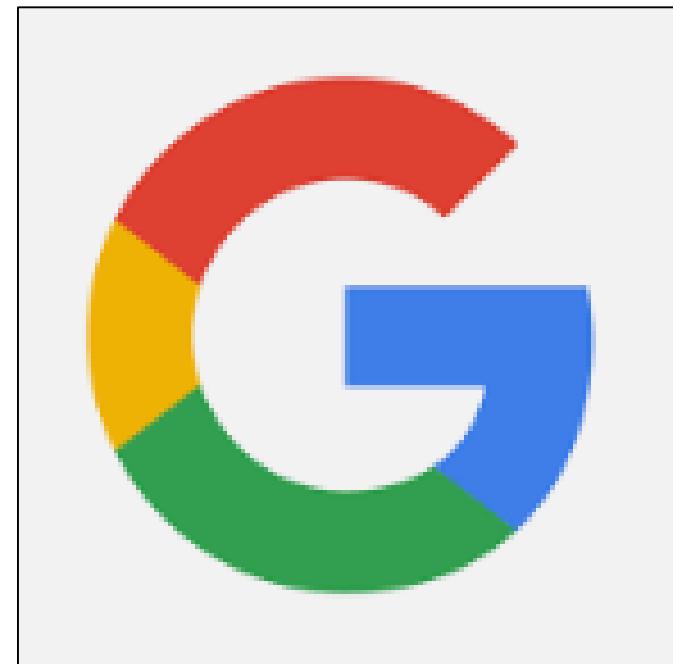
WEBCRAWLER.com

- 현대식 검색엔진의 시초
- 1994년 출시



Google.com

- 1998년 시작
- 2006년 Youtube 인수
- 2006년 Google Korea 법인 설립
- 2010년 세계 1위 포털 사이트



robots.txt

- Robots exclusion standard 표준 규약
- 각 홈페이지 뒤에 robots.txt 를 붙이면 조회할 수 있다. (ex: <https://www.google.com/robots.txt>)
- 웹 크롤러들이 접근하지 못하도록 제한하는 규약
- 교육 목적 외 해당 robots로 접근을 막아놨는데도 불구하고
서비스 출시 및 영리적 목적으로 사용되는 경우
법적인 분쟁이 발생할 수 있다.



```
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
Disallow: /?hl=&
Allow: /?hl=&gws_rd=ssl
Disallow: /?hl=&&gws_rd=ssl
Allow: /?gws_rd=ssl
Allow: /?pt1=true
Disallow: /imgres
Disallow: /u/
Disallow: /preferences
Disallow: /setprefs
Disallow: /default
Disallow: /m?
Disallow: /m/
Allow: /m/finance
Disallow: /wml?
Disallow: /wml/?
```

puppeteer

- 크롬 환경을 제어하기 위한 라이브러리
- 크롬의 기반인 Chromium 으로 작성되었기 때문에 가장 빠른 크롤링 라이브러리중 하나
- headless 지원
- pdf, 스크린샷 지원
- javascript를 통해 제어 가능
- SPA 크롤링 가능

Install

```
> npm i puppeteer
```

Repository

❖ github.com/puppeteer/puppeteer

Homepage

🔗 [github.com/puppeteer/puppeteer#read...](https://github.com/puppeteer/puppeteer#readme)

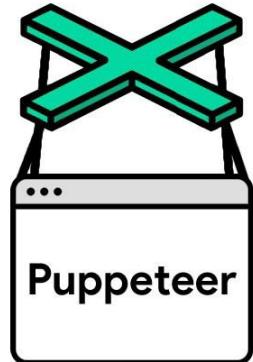
⬇ Weekly Downloads

3,527,096



headless browser

- 창이 없다는 의미
- 브라우저는 직접 띄우는것 대신 프로그램만으로 동작하는 브라우저



puppeteer

- **puppeteer.launch**

- 브라우저를 생성.
 - headless : false
 - 기본 값은 true이며 false로 지정시 브라우저가 보인다.

- **browser.newPage()**

- 해당 브라우저에 새 창을 띄운다.

- **page.goto('주소')**

- 해당 페이지를 해당 주소로 이동

```
const puppeteer = require('puppeteer')

const main = async() => {
  const browser = await puppeteer.launch({
    headless:false
  });
  const page = await browser.newPage();
  await page.goto('https://www.naver.com');

}

main();
```

puppeteer로 pdf 만들기

- **headless: true**
 - pdf는 headless:true 모드에서만 동작
- **page.pdf()**
 - test.pdf 의 A4 형식으로 pdf 생성
- **browser.close**
 - 브라우저 종료

```
const puppeteer = require('puppeteer')

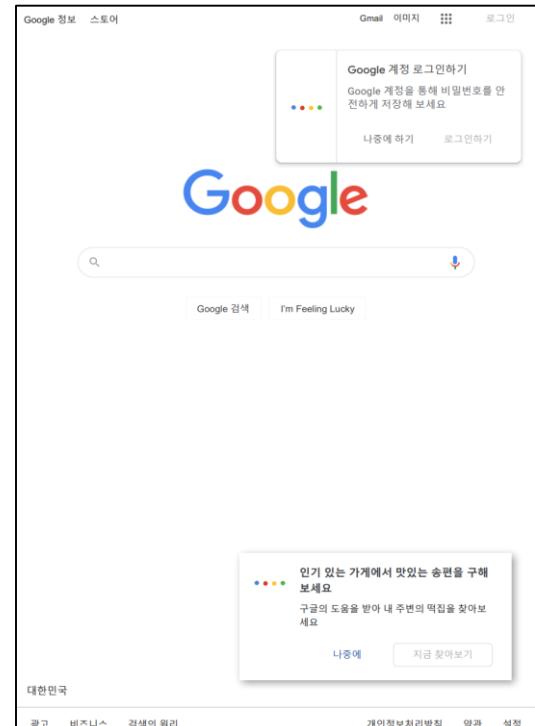
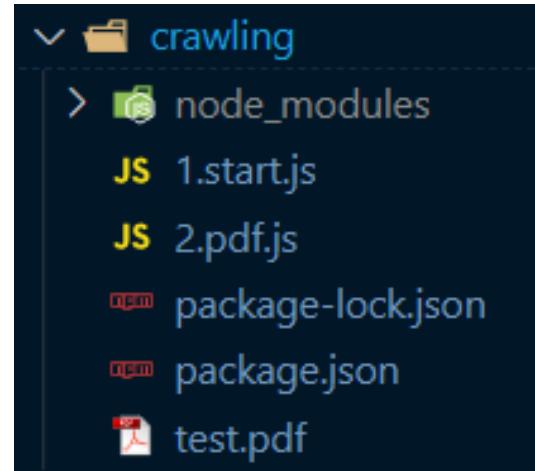
const main = async () => {

  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.pdf({ path: 'test.pdf', format: 'A4' });

  await browser.close();
}

main();
```



puppeteer로 screenshot 찍기

- **headless: true**
 - screenshot은 headless true/false 상관없이 동작한다.
- **page.screenshot**
 - path: 경로
 - fullpage
 - true 일시 전체 스크롤을 스크린샷 찍는다.

```
const puppeteer = require('puppeteer')

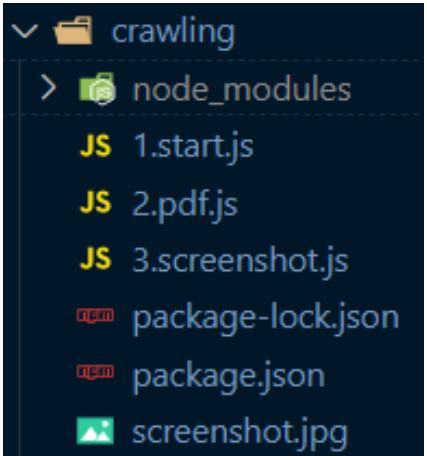
const main = async () => {

  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.screenshot({ path: 'screenshot.jpg', fullPage: true });

  await browser.close();
}

main();
```



네이버 웹툰 크롤링하기

- 월요 웹툰 전체의 제목과 링크를 가져와 보기

The screenshot shows the NAVER Webtoon homepage. At the top, there's a search bar and a login button. Below the header, there are tabs for Home, Webtoon (selected), Best, Trend, Manga, and My Page. Under the 'Webtoon' tab, there are sub-tabs for Today's Recommendation, Webtoon (selected), Best Webtoon, Author, Work Type, Period, Tag Webtoon, and Original Webtoon.

The main content area displays the "Weekly Recommended Webtoons" for today, July 4, 2022. It features three panels: "Mal Bak Wang" (44 chapters, 9.91 rating), "Cham Gyu" (87 chapters, 9.96 rating), and "Ip Sul Eun Namja" (17 chapters, 9.97 rating). Below this, there's a section for "Weekly Complete Webtoons" with various titles like "Cham Gyu", "2022 Nege Wa?", "Saini", etc., each with a rating and a 'View Details' button.

To the right, there's a sidebar titled "Webtoon Ranking" showing the top 30 real-time recommended webtoons, with titles like "Kim Bu Joong", "Ha Bok Young", "Lee Jin Ho", etc., along with their ratings and a 'View Details' button.

네이버 월요웹툰 크롤링하기

- 먼저 개발자 도구에서 querySelectorAll로 해당 선택자를 가져온다.

The screenshot shows the Naver Monday Webtoon page. It features two comic strips: one titled '참교육' (Chamgeok) with a rating of 9.86 and another titled '뷰티풀 군바리' (Beautiful Gunbari) with a rating of 9.81. The developer tools console on the right displays the following code and output:

```
length: 80
[[Prototype]]: NodeList
> document.querySelectorAll("ul.img_list >li dl>dt")
< NodeList(80) [dt, dt, dt, dt, dt, dt, dt, dt, dt,
, dt, dt,
, dt] i
  ▶ 0: dt
  ▶ 1: dt
  ▶ 2: dt
  ▶ 3: dt
  ▶ 4: dt
  ▶ 5: dt
  ▶ 6: dt
```

네이버 웹툰 크롤링 하기

- 네이버 월요 웹툰으로 이동
- `page.evaluate`
 - evaluate안에서는 `document` 접근이 가능
(기존 node에서는 접근이 되지 않는다.)
 - `Array.from`
 - 배열로 만들어주는 함수
 - `document.querySelectorAll`에 담기는 배열은 배열 메서드를 사용할수 없다.
 - `for`문만 사용 가능

```
const main = async () => {  
  
    const browser = await puppeteer.launch({  
        headless: true  
    });  
    const page = await browser.newPage();  
  
    await page.goto('https://comic.naver.com/webtoon/weekdayList?week=mon');  
  
    const data = await page.evaluate(() => {  
        const webToonLists = document.querySelectorAll('ul.img_list > li dl > dt');  
  
        const titles = Array.from(webToonLists).map(li => li.textContent.trim());  
  
        return titles;  
    })  
  
    console.log(data);  
  
    await browser.close();  
}  
  
main();
```

네이버 웹툰 크롤링 하기

- 결과 확인
 - fullname의 형태로 나와있지 않다.

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\8기-node\code\crawling>node 4.naver-webtoon.js
[
  '최후의 금빛아이',  '경비실에서 안...',  '세번째 로망스',  '아마도',
  '파견체',          '그림자 신부',      '싸이코 리벤저',  '백호랑',
  '왕따협상',        '매지컬 급식:...',  '나만의 고막남친', '지옥연애환담',
  '악녀 18세 ...',   '칼가는 소녀',      '모노마니아',    '흔들리는 세계...',
  '또다시, 계약...',  '슈퍼스타 천대리',  '결혼공략',     '역주행!',
  '사막에 편 달',   '기사님을 지켜줘',  '헬로맨스',    '디나운스',
  '오로지 오로라',   '이제야 연애',     '남주서치',    '별을 쓸는 소...'
]
```

네이버 웹툰 크롤링하기

- **fullname 가져오기**
 - a tag의 attribute중에 title에 fullName이 담겨있는것을 확인

```
▼<dl>
  ▼<dt>
    <a href="/webtoon/list?titleId=795297&weekday=mon" title="신화급 귀속 아이템을 손에 넣었다" 신화급 귀속 ...
  </dt>
  ▶<dd class="desc">...</dd>
  ▶<dd>...</dd>
  ▶<dd class="more">...</dd>
</dl>
```

네이버 웹툰 크롤링 하기

- **fullName 가져오기**

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://comic.naver.com/webtoon/weekdayList?week=mon');

  const data = await page.evaluate(() => {
    //.textContent로 가져오기
    //const webToonLists = document.querySelectorAll('ul.img_list >li dl>dt');

    //const titles = Array.from(webToonLists).map(li => li.textContent.trim())

    // fullname 가져오기
    const webToonLists = document.querySelectorAll('ul.img_list >li dl>dt>a');

    const titles = Array.from(webToonLists).map(li => li.getAttribute('title').trim())
  });

  return titles;
}

console.log(data);

await browser.close();
}

main();
```

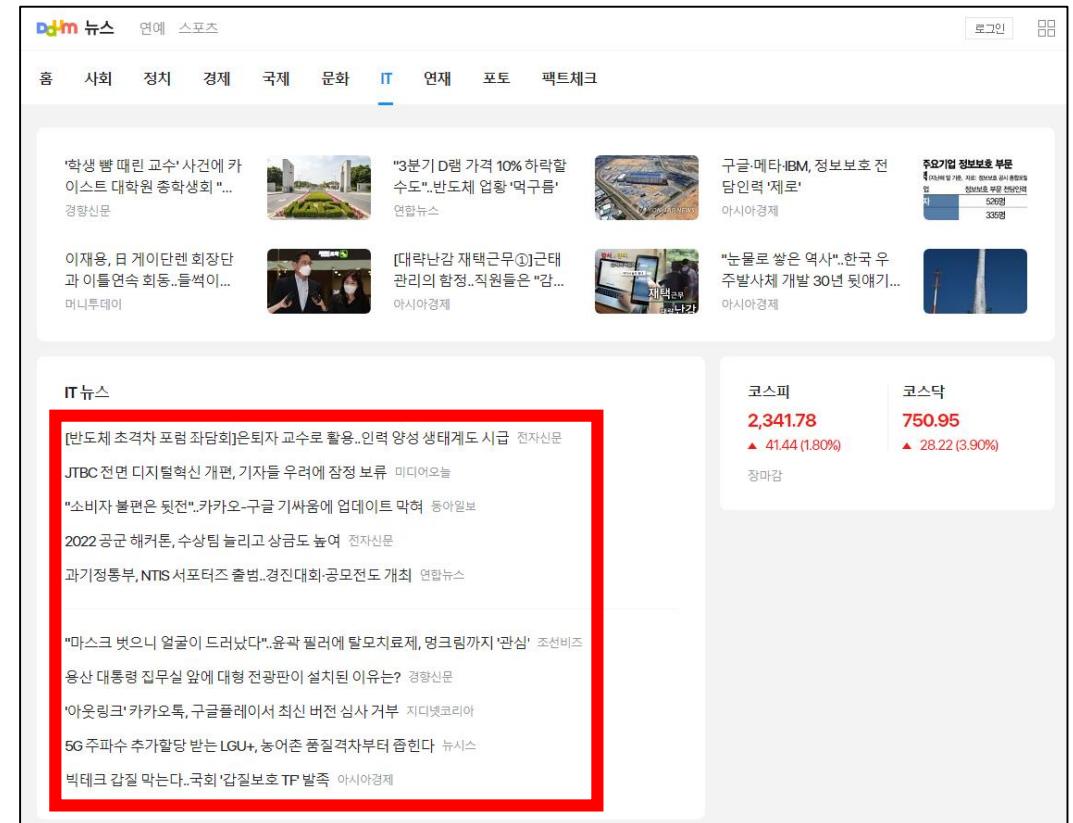
네이버 웹툰 크롤링 하기

- 결과 확인
 - fullName의 형태로 가져왔다.

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\8기-node\code\crawling>node 4.naver-webtoon.js
[
  '참교육',
  '뷰티풀 군바리',
  '신의 탑',
  '퀘스트지상주의',
  '원드브레이커',
  '장씨세가 호위무사',
  '팔이피플',
  '호랑신랑뎐',
  '백수세끼',
  '소녀의 세계',
  '신화급 귀속 아이템을 손에 넣었다',
  '앵무살수',
  '버림받은 왕녀의 은밀한 침실',
```

Daum 뉴스 <IT> 섹션의 헤드라인 기사를 크롤링하자

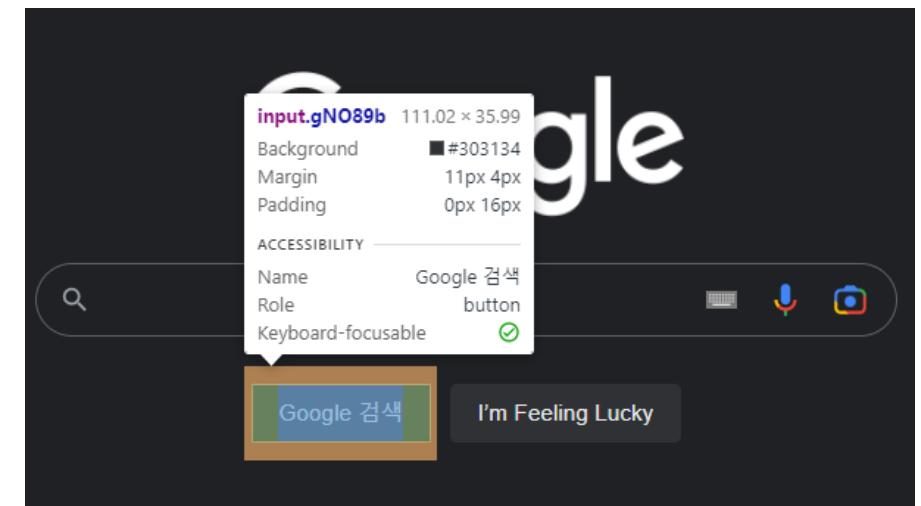
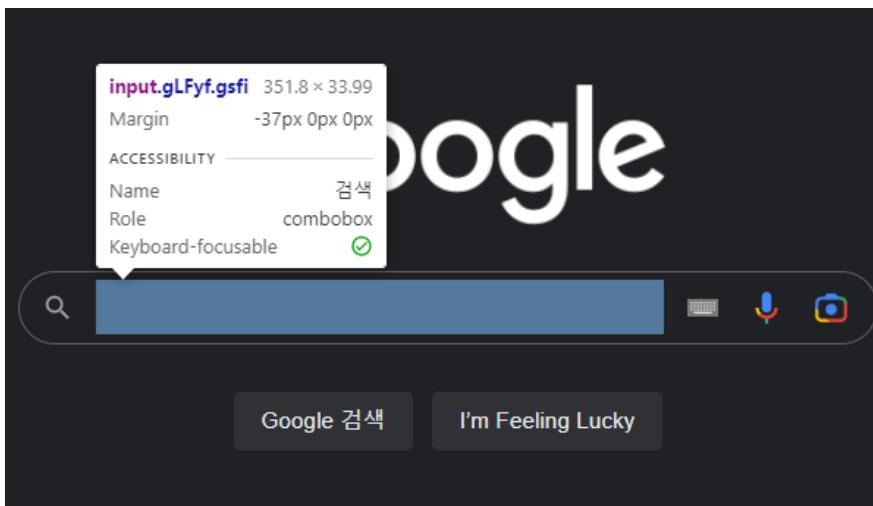
- <https://news.daum.net/digital#1>
 - 빨간색 박스 친 곳만 가져오기
 - 신문사를 제외한 내용만 가져오기



크롤링 응용

검색후 이동해서 캡쳐 찍기

- 먼저 필요한 selector 파악하기
 - input 입력에 필요한 selector input.gsf1
 - 검색 버튼을 위한 selector



검색후 이동해서 스크린샷 찍기

- 동작과정

- value=치킨
- input 버튼을 찾아서 클릭
- 스크린샷 찍기

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gNO89b").click();
  })

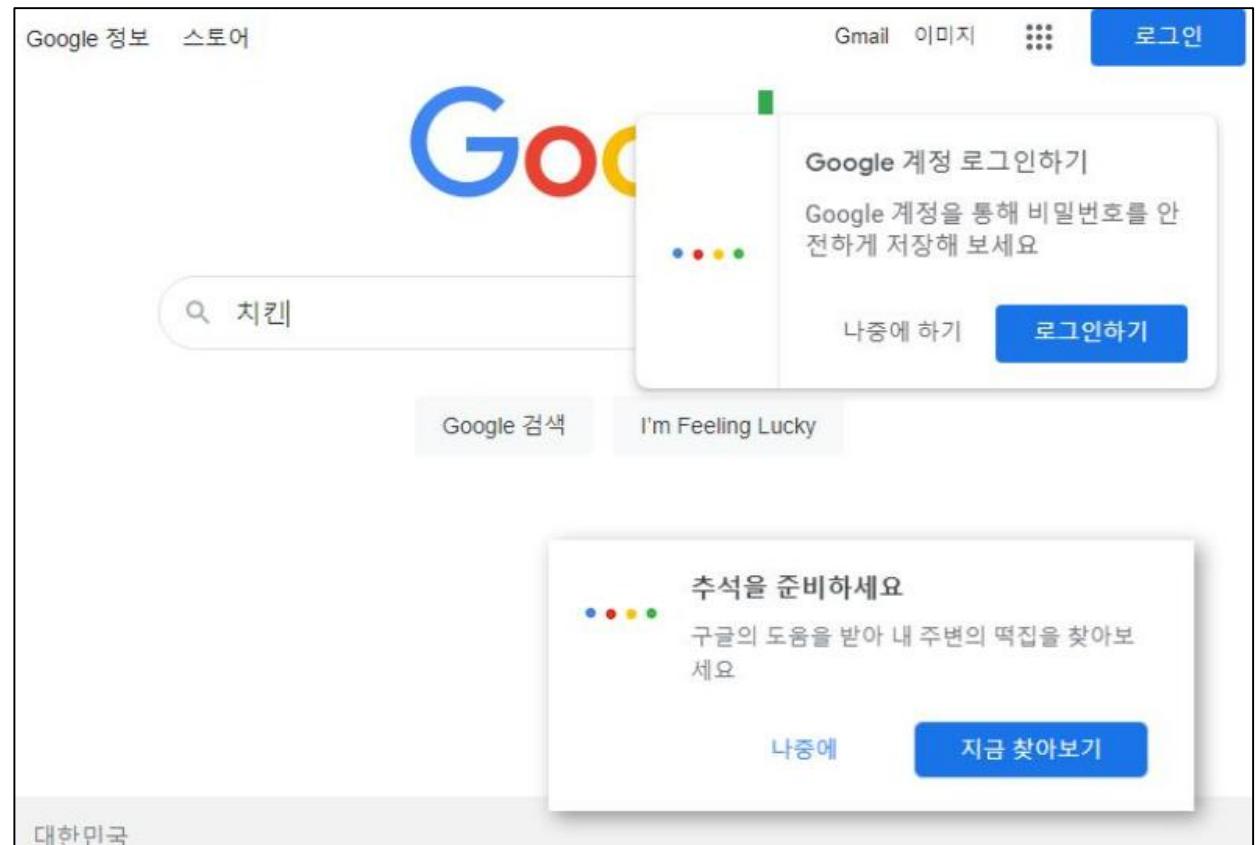
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

결과 확인하기

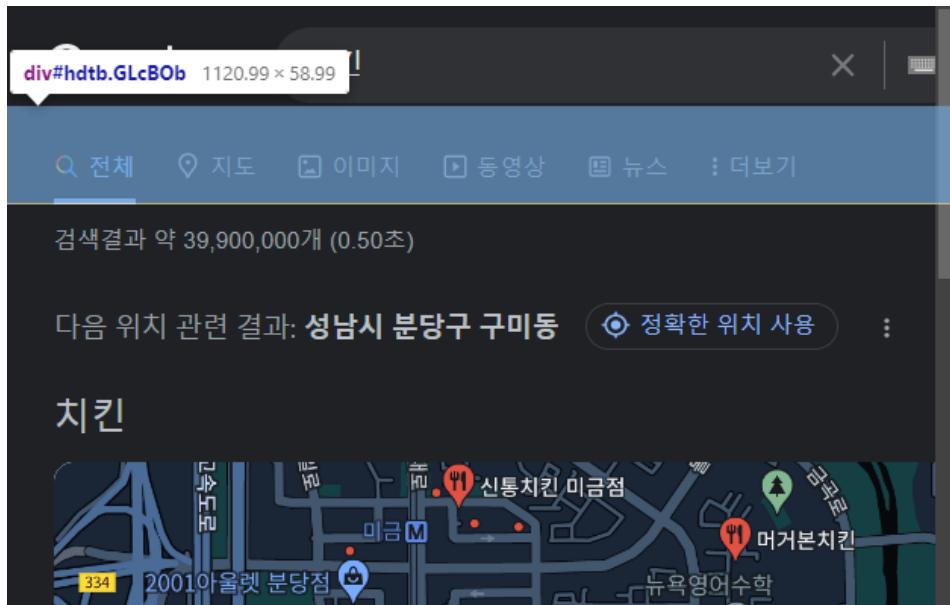
- 의도한 바와 다르게 치킨만 입력되고 동작하지 않는다.



검색 후 이동 해서 스크린샷 찍기

- **waitForSelector**

- 특정 선택자가 생성되기까지 대기
- 검색하면 네비게이션 부분이 생기기 때문에 해당 선택자가 생성되기까지 대기한다.



```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: false
  });
  const page = await browser.newPage();

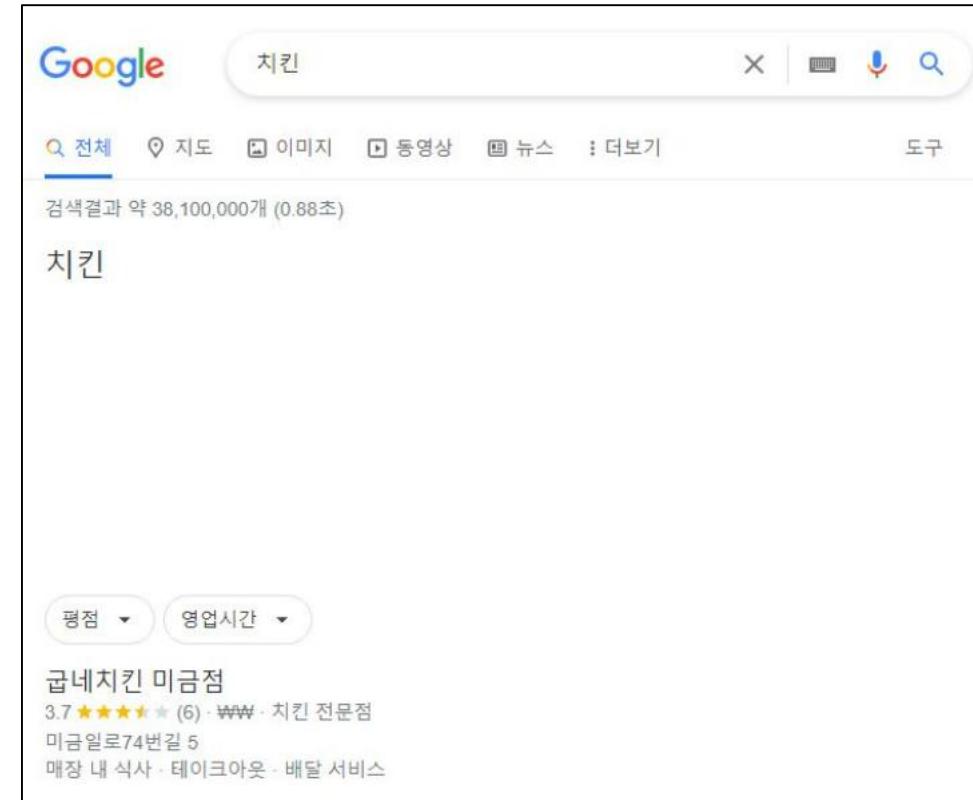
  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gN089b").click();
  })
  // selector가 오키로 대기
  await page.waitForSelector(".GLcB0b")
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

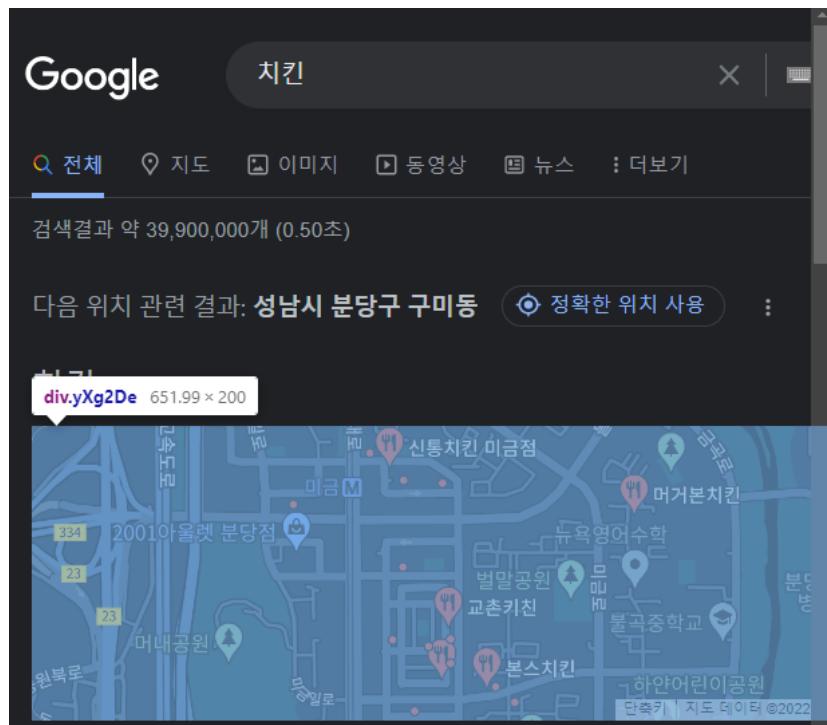
결과 확인하기

- 치킨 검색 후 이동까지는 나오게 되었다.
- 그 외 지도등 내용이 보이지 않는다.



검색후 이동해서 스크린샷 찍기

- 이번에는 해당 지도 부분이 나올수있도록 대기를 걸어보자.



```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gN089b").click();
  })
  // selector가 올바르게 걸려야만 작동합니다.
  await page.waitForSelector(".yXg2De")
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

결과 확인하기

- 모든 정보들을 잘 가져오게 되었다.

Google 치킨

전체 지도 이미지 동영상 뉴스 더보기 도구

검색결과 약 35,500,000개 (0.51초)

치킨

지도 데이터 ©2022 TMap Mobility

평점 영업시간

굽네치킨 미금점

3.7 ★★★★☆ (6) - 월화수요 치킨 전문점
미금일로74번길 5
매장 내 식사 · 테이크아웃 · 배달 서비스

머거본치킨

5.0 ★★★★★ (1) - 닭 튀김
구미동 77
매장 내 식사 · 테이크아웃 · 배달 서비스

신풍치킨 미금점

4.0 ★★★★☆ (80) - 월화수요 음식점
금곡동 147
매장 내 식사 · 테이크아웃 · 배달 서비스

메가박스 크롤링

- <https://www.megabox.co.kr/movie>
- 로딩(비동기 통신)에 대한 비동기 데이터 크롤링
- 각 포스터 별 src 가져오기



메가박스 이미지 크롤링

- 필요한 선택자 파악하기

전체영화

박스오피스	상영예정작	특별상영	필름소사이어티

— 메인페이지 상영예정작이 검색되었습니다.
img.poster.lozad 230 x 331



15 헌트

예매율 16.5% | 개봉일 2022.08.10

1.6k 예매 Dolby CINEMA



12 육사오(6/45)

예매율 11.3% | 개봉일 2022.08.24

457 예매



All 아라시 20주년 투어 콘서트

예매율 11% | 개봉일 2022.08.31

639 예매 Dolby CINEMA

```
> 78: img
> 79: img
length: 80
[[Prototype]]: NodeList
> document.querySelectorAll(".movie-list img.poster")
< NodeList(20) [img.poster.lozad, img.poster.lozad, img.poster.lozad]
0: img.poster.lozad
1: img.poster.lozad
2: img.poster.lozad
3: img.poster.lozad
4: img.poster.lozad
5: img.poster.lozad
6: img.poster.lozad
7: img.poster.lozad
8: img.poster.lozad
9: img.poster.lozad
10: img.poster.lozad
11: img.poster.lozad
12: img.poster.lozad
13: img.poster.lozad
14: img.poster.lozad
15: img.poster.lozad
16: img.poster.lozad
17: img.poster.lozad
18: img.poster.lozad
19: img.poster.lozad
length: 20
[[Prototype]]: NodeList
```

메가박스 이미지 크롤링

- **waitForSelector 활용**

- loading 후 해당 div가 생성
 - 해당 선택자가 생성되기를 대기

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.megabox.co.kr/movie');
  await page.waitForSelector(".movie-list img.poster")
  const data = await page.evaluate(()=>{
    const images = document.querySelectorAll('.movie-list img.poster')
    const result = Array.from(images).map(li=> li.getAttribute('src'));
    return result
  })
  console.log(data);

  await browser.close();
}

main();
```

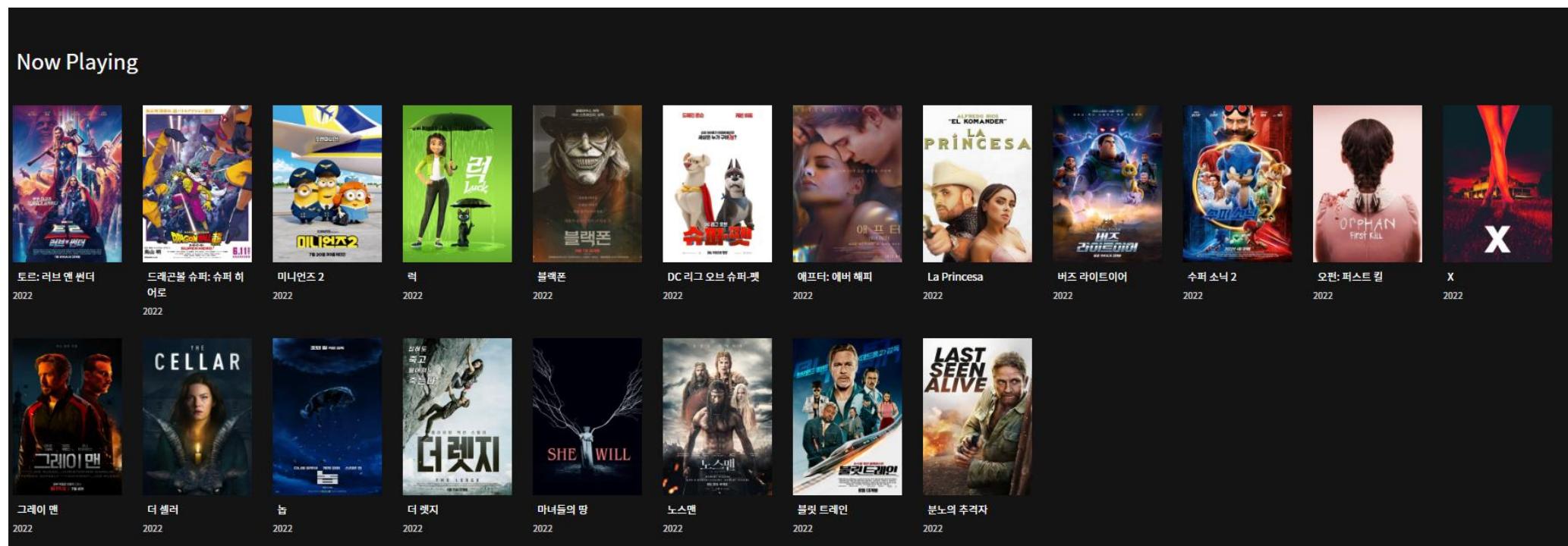
메가박스 이미지 크롤링

- 결과 확인

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\87|node\code\10_crawling>node 8.megabox.js
[
  'https://img.megabox.co.kr/SharedImg/2022/08/05/QDUC0cjm2bnWDCCQPYpQvelnoFe1CCfH_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/wMbAzWp0ahUuTxhD5hq8DzSXEjQD6gNY_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/05/09/6zfAYe6IrZ8BWnruqEfafwakt5cUjWgX_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/11/fcfDt05PrS4H5YusAg7BjUZ3JfhS5MRj_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/28/1ogGcwYxCNJ9MTnizlZLdZ6REjg6xX1z_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/oUqrNQTflUqvHUQG6kv1zF8SEhJSomfh_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/02/hQsvDEd41AY0pm0N6fAhJ55ouwS5K3wb_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/04/oPoYTP4zSq5iWCCGYj4SQ3tuSU5J89Rl_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/B5P9rlhS5BwMjvPi35pyH60I5Dv75kDm_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/ypNwU8S72y0b03rgLEc01ih8uG6mcZgm_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/v1Y6EQN1rpaADkfQhCPwnSxFgGYZwfB0_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/HJQ3kqIL5kUJBmKQSc3HOxZV7SUpHWGx_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/04/wo9hN6dpVFzHp4d3MpGmPEGTC0CE2yt_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/06/07/S3GJQZbpshoIx0Lelerscl9rlI14FHqK_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/CyFJcEEUDngBC07vzJra2mSocDg1Trmk_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/06/KnVXVmorgtWJC8EauyxVAUGEGLuDx95_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/10/sdYAZCCnn0gvx119La32kjjInsNB9CuB_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/19/Wx0F5W0amgT0s5fmLXzMuiSQvttecJMLc_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/26/0yefBJeFcMey7RXhnYWLAeGPBTWMQiUx_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/02/17/djm7aYuL9bQGZRxyUH75wATz9ub9ouk_420.jpg'
]
```

도전 - 영화 페이지 크롤링

- <https://loy124.github.io/vue-movie/> 사이트 접속
 - 해당 사이트는 Vue.js로 생성되어 있다.
 - 해당 포스터 src, 제목 배열, 객체 형식으로 가져오기



도전 - 영화 페이지 크롤링

- 목표 결과

```
[  
  {  
    title: '토르: 러브 앤 썬더',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//bZLrpWM065h5bu1msUcPmLFsHBe.jpg'  
  },  
  {  
    title: '드래곤볼 슈퍼: 슈퍼 히어로',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//uohymzBVaIYjbnoQstbnlia6ZPJ.jpg'  
  },  
  {  
    title: '미니언즈 2',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//1heBUD8o0sgdqLWyeXkyLR2POKb.jpg'  
  },  
  {  
    title: '럭',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//nN2iHej7TyISjXC6Jl577aqpDHJ.jpg'  
  },  
  {  
    title: '블랙폰',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//eJPIkRiYGnvQaPCZtTRTDmhMJVK.jpg'  
  },  
]
```

8장. express

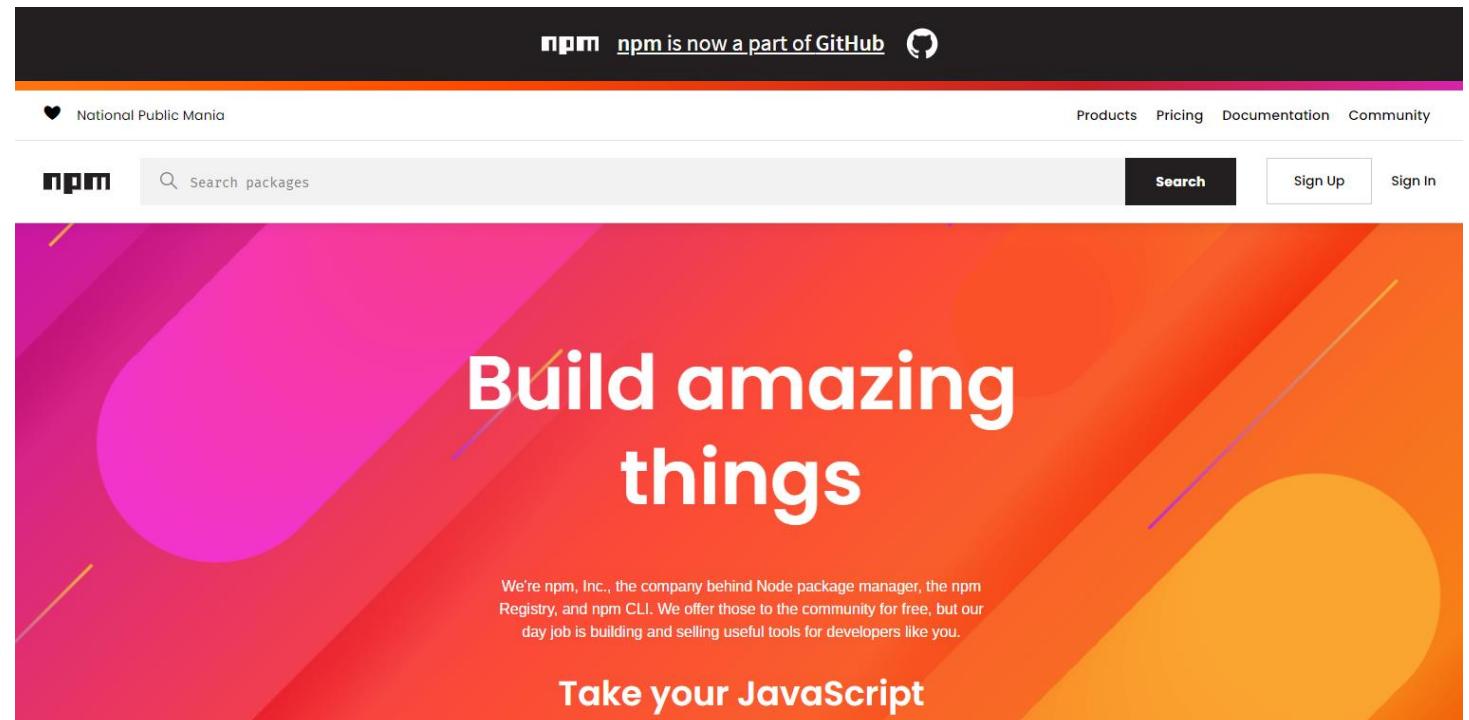
챕터의 포인트

- npm
- 간단한 express 웹서버
- 요청과 응답
- 클라이언트와 CORS
- Routing

npm

node package manager

- 자바스크립트 프로그래밍 언어를 위한 패키지 관리자
- <https://www.npmjs.com/>



This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allow us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, see our [Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

Accept

Decline

npm init - 패키지 생성

```
C:\Users\mincoding\Desktop\node-test>npm init
npm WARN config global `--global`, `--local` are
deprecated. Use `--location=global` instead.
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.
Press ^C at any time to quit.
package name: (test) node-practice
version: (1.0.0)
description: for ssafy node.js class
entry point: (index.js)
test command:
git repository:
keywords:
author: Jon Jaryong Lee
license: (ISC)
About to write to C:\Users\mincoding\Desktop\node-test\package.json:
```

```
1  {
2      "name": "node-practice",
3      "version": "1.0.0",
4      "description": "for ssafy node.js class",
5      "main": "index.js",
6      ▷ Debug
7      "scripts": {
8          "test": "echo \\\"Error: no test specified\\\" && exit 1"
9      },
10     "author": "Jon Jaryong Lee",
11     "license": "ISC"
12 }
```

간단한 express 웹서버

nodemon 이란?

- 서버 코드 수정시마다 매번 서버를 재부팅해야하는 불편함
- nodemon 은 코드가 수정되면 자동으로 서버를 재시작



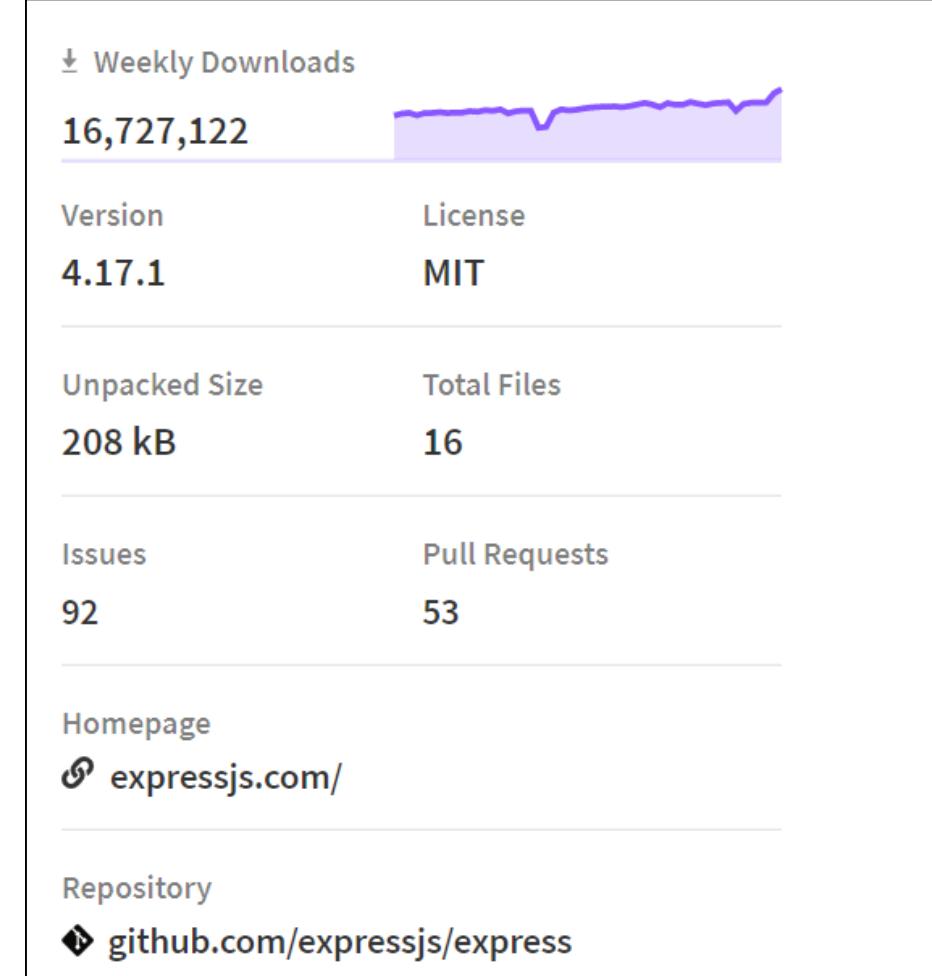
global install (전역 설치)

- package.json 이 위치한 프로젝트 디렉터리 이외에도,
 - 어떤 경로에서든 패키지 호출 가능
 - 커멘드라인에서 실행하는 프로그램은 전역 설치가 기본
-
- **npm install --location=global nodemon**
 - nodemon 전역 설치
 - **npm list --location=global --depth=0**
 - 전역 패키지 확인

```
C:\Users\mincoding\Desktop\node-test>npm list --location=global --depth=0
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
C:\Users\mincoding\AppData\Roaming\npm
└─ nodemon@2.0.16
```

Node.js의 대표 웹 프레임워크

- http와 Connect 컴포넌트를 기반으로 만들어짐
- Node.js의 API들을 단순화
- 쉬운 확장성



전역설치가 아니라, 지역 설치해보자.

package.json 이 위치한 경로로 이동 후,

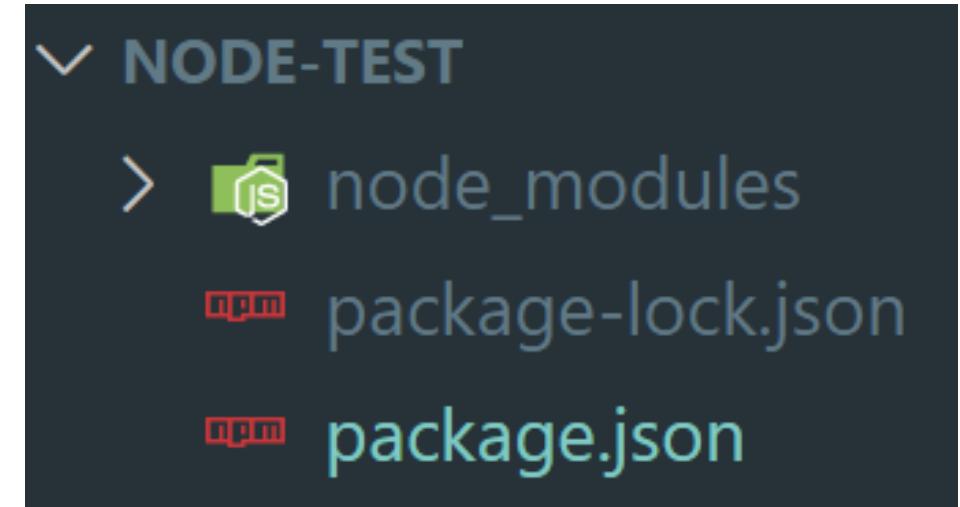
npm install express // 현재 프로젝트에 express 설치

```
C:\Users\mincoding\Desktop\node-test>npm i express
```

install 의 약어인 i 를 써도 된다.

각각의 개념들이 무엇을 의미하는지 숙지하기

- **node_modules**
 - 실제 설치된 패키지
- **package.json**
 - 명세서
 - 프로젝트에 관련된 데이터가 담긴다
 - 프로젝트 정보 및 빌드 방식
 - 필요한 패키지 목록 정리
- **package-lock.json**
 - 상세 명세서
 - 각 패키지 별 정확한 버전명이 명시 되어 있음



package.json 안에 dependencies 가 추가됨

- **dependencies**

- 한국어로 의존성
- 우리가 만든 패키지(node-practice) 를 실행시키기 위해 필요한 "다른 패키지" 목록
- node-practice 패키지는 express 가 설치되어 있지 않으면 작동 하지 않는다.
즉, 의존(depend)하고 있다

```
"dependencies": {  
    "express": "^4.18.1"  
}
```

package-lock.json

- 열어보면 1000 줄이 넘어가는 파일
 - package.json 만으로 충분하지 않다.
 - package.json 요약 명세서
 - package-lock.json 상세 명세서
 - express 역시도 하나의 패키지이므로, "다른 패키지들" 을 dependencies 로 가지고 있다.
 - 다른 개발 PC 또는 서버 컴퓨터로 우리가 만든 패키지(node-practice) 를 옮겼을 때,
 - 원래 개발 PC 와 완벽하게 동일한 환경에서 우리 패키지를 동작시키기 위함
 - 즉, 협업 시엔 package-lock.json 까지 공유

```
1015      "integrity": "sha512-BNGbL  
1016    }  
1017  }  
1018  }  
1019  |
```

[참고] express 의 dependencies 를 확인해보자

Confidential

- npm list --depth=0

- 현재 node-practice 패키지에서 확인되는 dependencies 는 express 하나밖에 없다.

```
node-practice@1.0.0
└── express@4.18.1
```

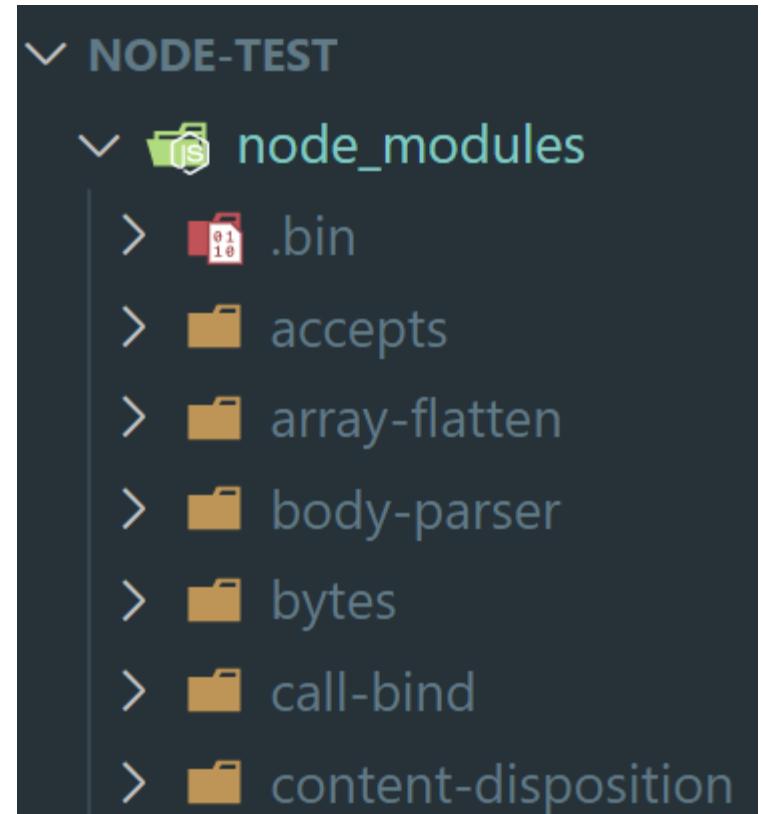
- npm list --depth=1

- express 패키지의 dependencies 출력됨
- 즉, --depth 숫자가 늘어날수록 패키지 안에 패키지 안에 패키지...의 dependencies 확인 가능

```
node-practice@1.0.0 C:\Users\min
└── express@4.18.1
    ├── accepts@1.3.8
    ├── array-flatten@1.1.1
    ├── body-parser@1.20.0
    ├── content-disposition@0.5.4
    ├── content-type@1.0.4
    ├── cookie-signature@1.0.6
    ├── cookie@0.5.0
    ├── debug@2.6.9
    └── depd@2.0.0
```

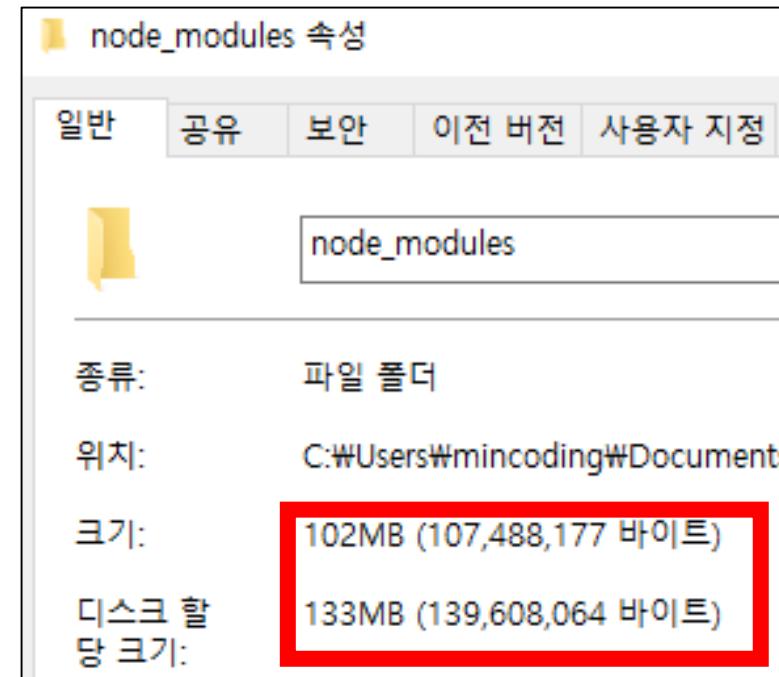
node_modules

- package-lock.json 에 기입된대로 설치된 실제 dependencies
- 협업할때나, github 업로드 시 node_modules 은 제외하고 공유
 - package.json, package-lock.json 두 개의 명세서만 있으면 된다.
 - 위 명세로 dependencies 설치가 가능하다.
 - node_modules 의 용량은 어마어마하다.
- 간단한 테스트:
 - node_modules 디렉터리를 삭제한 후,
 - 커멘드라인에 npm i 를 입력해보자.



node_modules 크기 확인하기

- todo list 를 만들기위한 vue.js 프로젝트의 node_modules 크기



100MB 는 기본이다.

샘플 코드 작성하기

- 간단한 express code를 작성하기.
- index.js 생성



```
1 const express = require("express");
2 const app = express();
3 const PORT = 8080;
4
5 app.get("/api/info", (req, res) => {
6   return res.json({
7     name: "jony",
8     job: "tutor",
9   });
10 });
11
12 app.listen(PORT, () => console.log(`this server listening on ${PORT}`));
```

```
1 // express 모듈 가져옴
2 const express = require("express");
...  
3 // express 함수 호출 후 app 객체 생성
4 const app = express();
5 // PORT 상수. 8080 사용 예정
6 const PORT = 8080;
```

코드 분석

- 사용자가 /api/info 로 http 요청을 보내면,
JSON Format 으로 name 과 job 을 응답
- **get**
 - http 요청의 한 종류
 - REST API 챕터에서 배움
- **"/"**
 - API 라우트 경로
다음 장 Routing 에서 배움
- **req, res**
 - 요청객체 (request), 응답객체 (response)
- **return res.json()**
 - JSON Format 으로 응답 객체 res 에 name, job 을 실어서 보낸다.

```
8 app.get("/api/info", (req, res) => {  
9   return res.json({  
10    name: "jony",  
11    job: "tutor",  
12  });  
13});
```

마지막 줄에서, 서버를 작동시킨다.

- **listen**
 - 서버 요청 대기 상태
- **PORT 8080**
 - 즉, express 서버는 실행과 동시에 8080 포트에서 클라이언트 요청(request) 대기중
- **두번째 파라미터 콜백함수**
 - listen 성공 시 실행. 터미널에 어느 포트에서 서비스되고있는지 안내하는 용도

```
12     app.listen(PORT, () => console.log(`this server listening on ${PORT}`));
```

express 서버 실행하기

- `nodemon ./index.js`

```
C:\Users\mincoding\Desktop\node-test>nodemon index.js
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
this server listening on 8080
```

8080 포트에 작동중이라는 안내문 확인

구글에서 크롬 웹스토어 검색

The screenshot shows a Google search results page with a dark theme. The search bar at the top contains the query "크롬 웹스토어". Below the search bar, there are several search filters: 전체 (selected), 쇼핑, 동영상, 이미지, 뉴스, and 더보기. To the right of the filters is a "도구" button. The main search results area displays a single result from the Chrome Web Store:

<https://chrome.google.com/webstore> ▾
Chrome 웹 스토어
Chrome 웹 스토어. 확장 프로그램, 테마, 앱으로 데스크톱 컴퓨터에서 Chrome을 맞춤설정합니다. Chrome에 새로운 기능 추가. Chrome에 확장 프로그램을 설치하여 새 ...

테마
브라우저를 맞춤설정할 수 있는 맞춤 브라우저 스킨입니다.

로그인
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

다크 앤 블랙 테마
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

모두 보기
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

[google.com 검색결과 더보기 »](#)

JSON 검색 후, JSON Formatter 설치

The screenshot shows the Chrome Web Store search results for the query "JSON". The search bar at the top left contains "JSON". To the right of the search bar, there is a user profile icon with the email "jaryong.lee@mincoding.co.kr". Below the search bar, a list of extensions is displayed:

- json
- jsonview
- json formatter
- json viewer
- jsonview hateoas
- json beautifier
- json editor

The "json formatter" extension is highlighted with a larger preview card. This card features a dark gray background with the text "JSON Formatter" and two large curly braces "{}". Below the preview card, the extension's details are shown:

JSON Formatter
callumlocke.com 추천
Makes JSON easy to read. Open source.
★★★★★ 1,810 개발자 도구

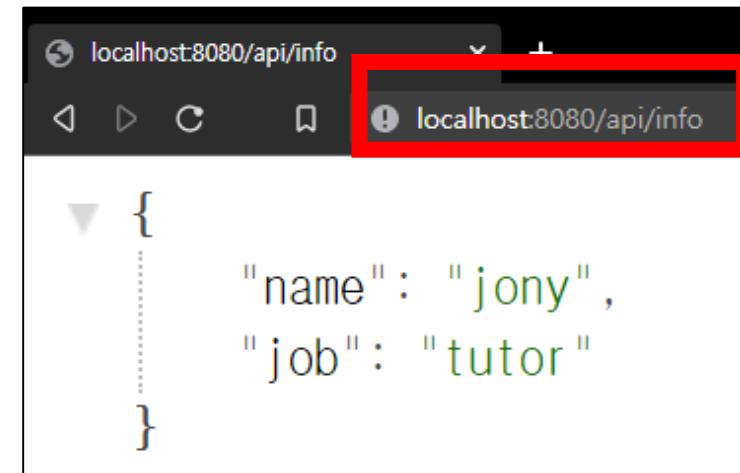
A green "추가설치" (Install) button is visible at the bottom left of the preview card.

- localhost:8080/api/info

- 접속 시, 작성해둔 객체 리턴
- localhost 내 (local) 컴퓨터 (host)
- /api/host 코드에서 지정한 경로

- 이렇게 활용되는 서버를 API 서버라고 한다.

- 활용: 사용자 요청을 받아 SQL DB 접근 후
- 결과를 JSON Format으로 리턴
- (Node.js 수업의 **최종 목표**)

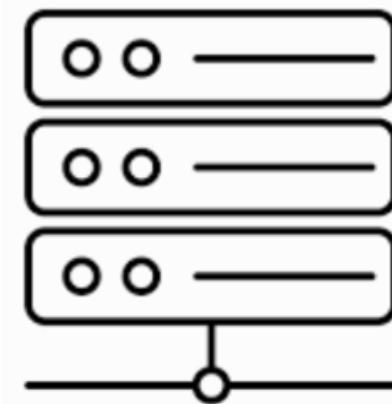
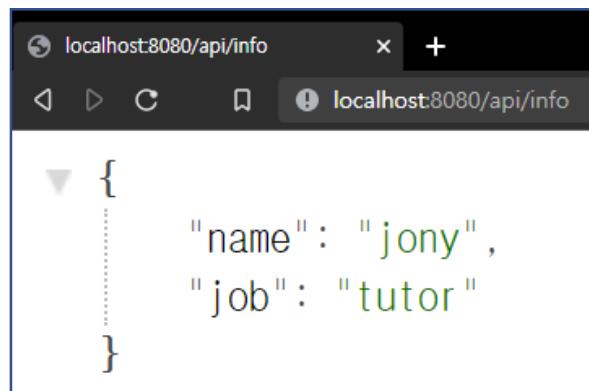
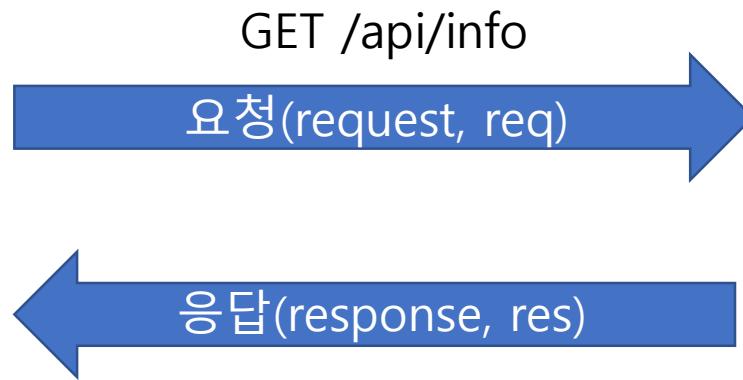


요청과 응답

요청과 응답은 통신의 기본이다.



Client (Frontend)



Server (backend)

express의 req, res 알아보기

- **req, res**

- 각각 요청(req)과 응답(res)에 대한 정보 및, 활용 가능한 메서드를 가진 **거대한 객체**
- 각각 console.log 를 사용해서, 터미널에 출력해보자
- 오른쪽 이미지는 req 객체의 아주 작은 일부분
- 우린 어떻게 활용했는가?
 - res 객체 안에, json 함수를 사용해,
 - 객체를 JSON Format 으로 바꾼 후, 클라이언트로 보냄

```
app.get("/api/info", (req, res) => {
  console.log(req);
  // console.log(res);
  return res.json({
    name: "jony",
    job: "tutor",
  });
});
```

```
<ref *2> IncomingMessage {
  _readableState: ReadableState {
    objectMode: false,
    highWaterMark: 16384,
    buffer: BufferList { head: null,
      length: 0,
      pipes: [],
      flowing: null,
      ended: false,
      endEmitted: false,
      reading: false,
      constructed: true,
      sync: true,
      needReadable: false,
      emittedReadable: false,
      readableListening: false,
      resumeScheduled: false,
      errorEmitted: false,
      emitClose: true,
      autoDestroy: true,
      destroyed: false,
      errored: null,
      closed: false,
    }
  }
}
```

req 객체

클라이언트와 CORS

API 서버에 접속하는 클라이언트를 만들어보자

- 단, 현재 작업 디렉터리 이외의 공간에 만들 것
 - ex) 바탕화면

```
<body>
  <h1>API TEST</h1>
  <button>서버의 데이터 가져오기</button>
  <script src="https://cdn.jsdelivr.net/npm/axios@0.27.2/dist/axios.min.js"></script>
  <script>
    const btn = document.querySelector("button");
    btn.addEventListener("click", async () => {
      try {
        const response = await axios.get("http://localhost:8080/api/info");
        console.log(response.data);
        if (response.data) {
          console.log(response.data);
        }
      } catch (error) {
        console.log(error);
      }
    });
  </script>
</body>
```

axios CDN
(jsdelivr)

버튼 누를 시 동
자

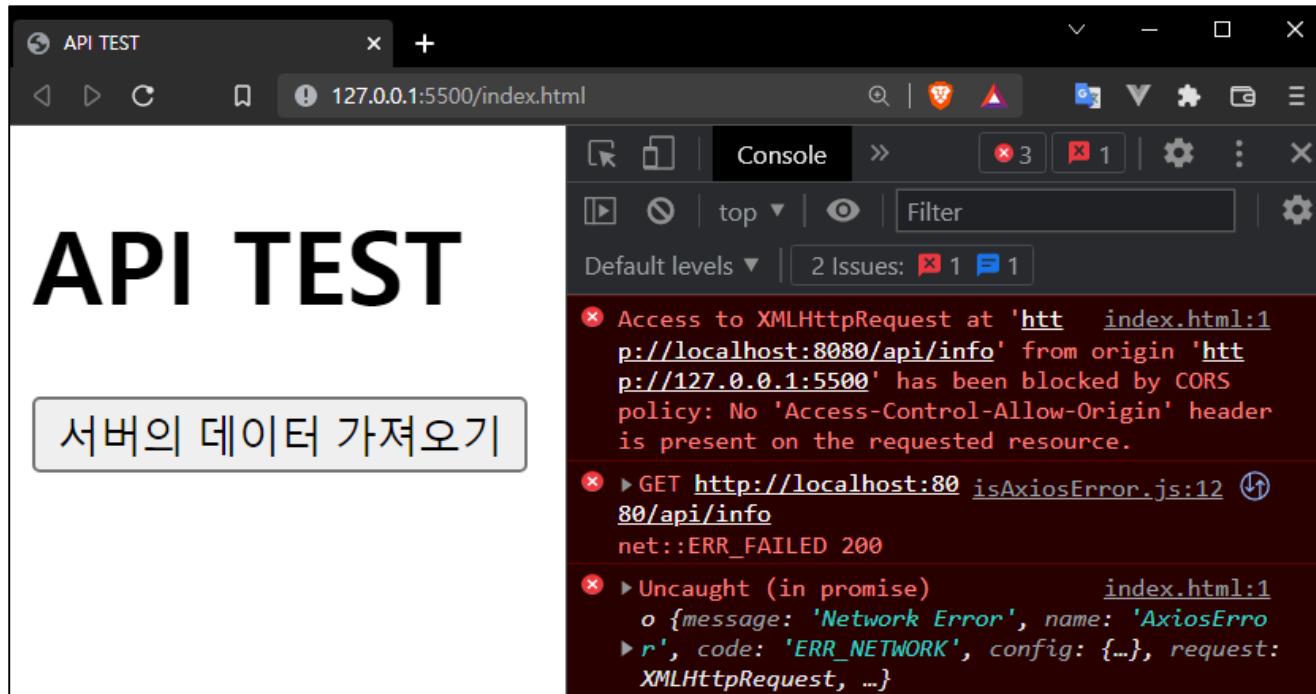
해당 경로에, GET 요청
보냄

- 클라이언트 (프론트엔드)
 - 코드는 바탕화면에 위치
- 서버 (백엔드)
 - 코드는 우리가 만든 node.js 패키지에 위치
- 핵심은, 두 코드가 다른 곳에 위치한다는 것!



클라이언트 실행하기

- live server 로 통신 해보기작동시켜 보았으나... 통신이 되지 않는다.
- 이것은 **CORS** 때문에 발생한다.



현재 live server 포트는 5500 번

다른 출처 리소스 공유

CORS (Cross-Origin Resource Sharing) Policy

- 기본적인 웹 Policy (정책)
- IP 뿐만 아니라 PORT 까지 일치해야만 리소스 공유 가능하다는 정책(규칙)
- 여기서 리소스? 우리 상황에선 JSON
- 즉, JSON을 공유하고자 하는 클라이언트와 서버
 - IP와 PORT가 일치해야 정확하게 통신이 가능하다는 규칙
- 현재 상태 정리
 - 클라이언트
 - Live Server - 5500번 포트
 - 서버
 - Express - 8080번 포트
 - 둘다 localhost인건 동일하나
서로 포트가 다르기 때문에 정책에 위배

- 왜 CORS Policy 를 정해서 우릴 힘들게 할까?

- 북한이나 중국에서 내 웹서버에 요청을 한다면, 신뢰할 수 있는가?
- 서버 개발자는 통신을 허용할 범위를 정해야 한다!

- 해결책

- 서버 코드인 Express 에서, CORS 패키지 설치
 - 특정 IP및 포트를 개방시키기

npm 패키지 cors 설치하기

- 여기서 다운받는 CORS는 정책이 아니라, NPM 패키지 이름이다

The screenshot shows the npmjs.com page for the 'cors' package. At the top, it displays the package name 'cors' with a version of '2.8.5', marked as 'Public' and published 4 years ago. Below this are tabs for 'Readme' (highlighted in yellow), 'Explore' (BETA), '2 Dependencies', '10,510 Dependents', and '34 Versions'. The 'Readme' tab contains a link to the command 'npm i cors' which is highlighted with a red box. To the right of the command, there's an 'Install' button. Further down, there are sections for 'Repository' (github.com/expressjs/cors), 'Homepage' (github.com/expressjs/cors#readme), and 'Weekly Downloads' (8,223,991). At the bottom, it shows the 'Version' as '2.8.5' and 'License' as 'MIT'.

cors 2.8.5 • Public • Published 4 years ago

[Readme](#) [Explore](#) BETA [2 Dependencies](#) [10,510 Dependents](#) [34 Versions](#)

cors

npm v2.8.5 downloads 36M/month build passing coverage 100%

CORS is a node.js package for providing a [Connect/Express](#) middleware that can be used to enable [CORS](#) with various options.

Follow me (@troygoode) on Twitter!

- Installation
- Usage
 - Simple Usage
 - Enable CORS for a Single Route
 - Configuring CORS
 - Configuring CORS Asynchronously
 - Enabling CORS Pre-Flight

Install

```
> npm i cors
```

Repository

Homepage

Weekly Downloads

8,223,991

Version

2.8.5

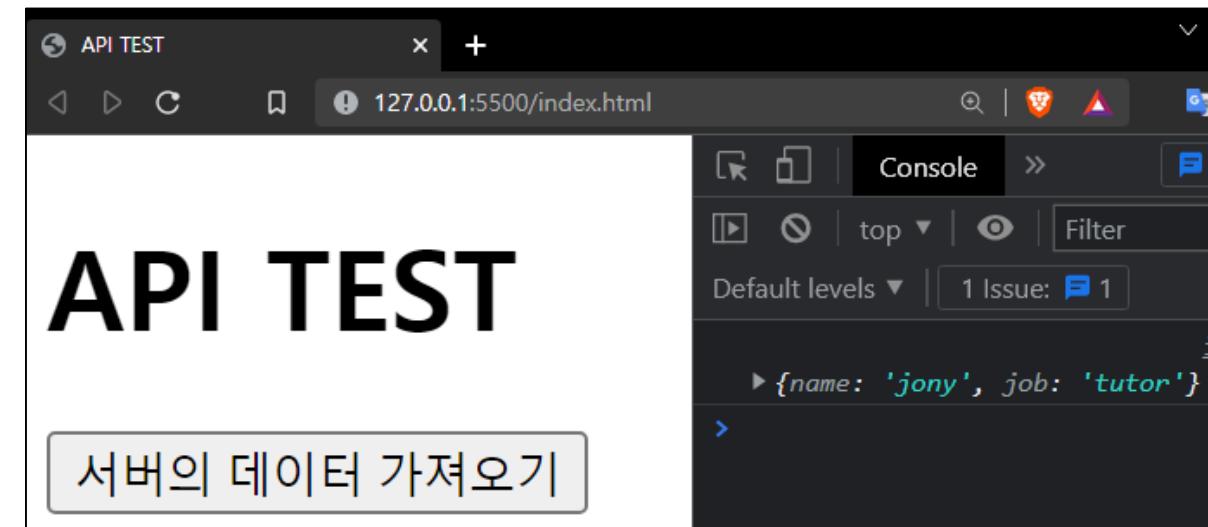
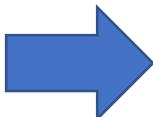
License

MIT

기존 코드에 cors 관련 코드 추가하기

- `const cors = require("cors");`
 - cors 패키지 가져오기
- `app.use(cors());`
 - 해당 cors를 사용
 - 바로 app.use로 cors()를 사용하게 되면 모든 접속이 허용되는 방식
 - 추가 옵션을 부여해서 제어하는 것 또한 가능

```
1 const express = require("express");
2 const app = express();
3 const PORT = 8080;
4
5 const cors = require("cors");
6 app.use(cors());
```

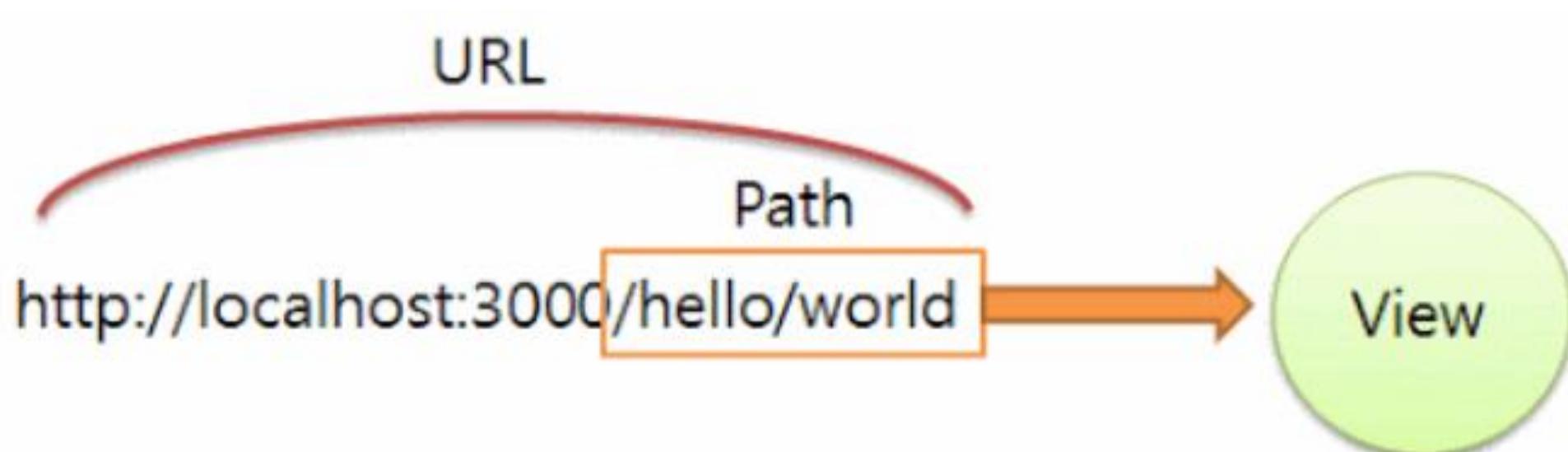


cors 적용 후 통신에 성공

Routing

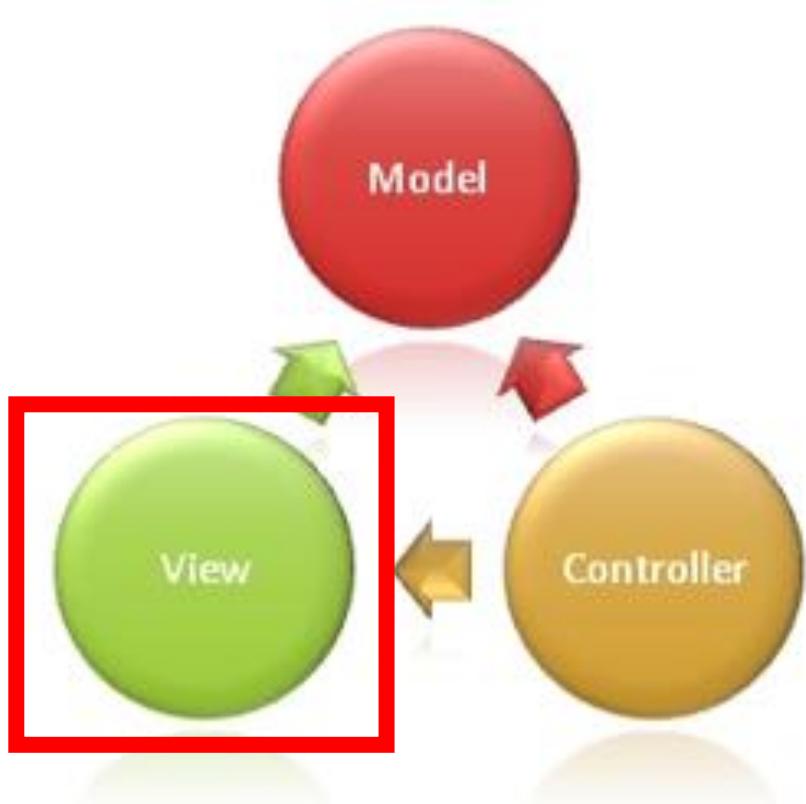
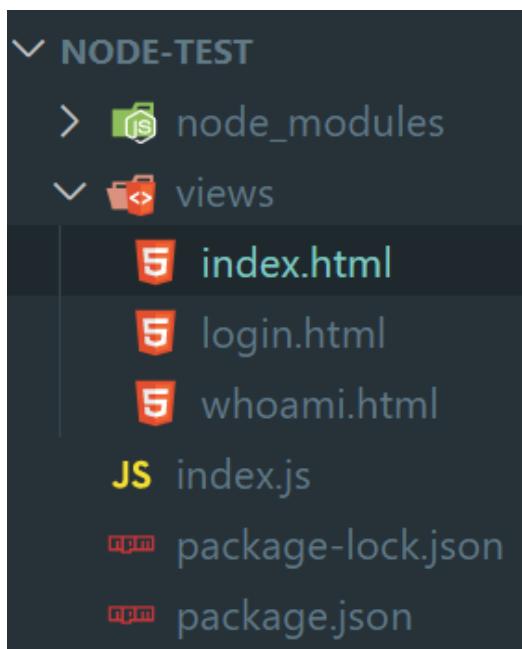
URL 라우팅

사용자가 접근한 경로에 따라서 그에 맞는 메소드를 호출해주는 기능



node.js에서 view 디렉토리 생성

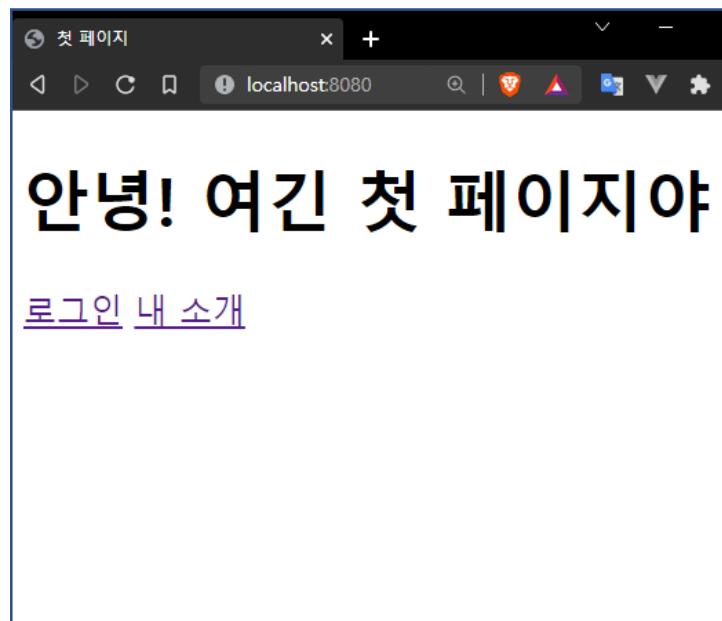
- 클라이언트 코드가 들어가는 디렉토리는 관습적으로 views 라고 이름 짓는다.



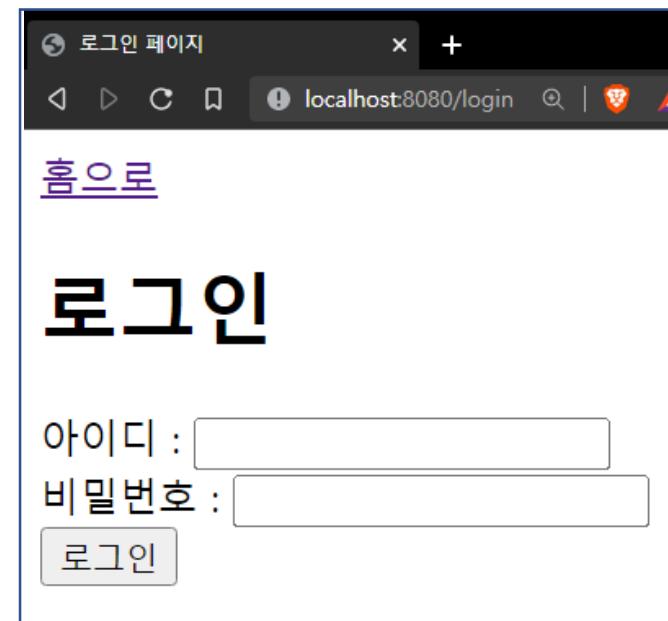
페이지 생성하기

Confidential

index.html



login.html



whoami.html



```
<a href="/login">로그인</a>
<a href="/whoami">내 소개</a>
```

```
<a href="/">홈으로</a>
```

주요 메서드 정리

- **sendFile**
 - res에서, 파일 전송 시 사용하는 메서드. 여기선 html을 보낼 용도
 - 상대 경로가 아닌 절대 경로로만 동작한다.
- **_dirname**
 - 현재 기기에서 작동중인 서버의 경로를 의미하는 상수
 - 왜? 각각의 기기마다 서버의 경로가 전부 다르다. (또한 sendFile은 절대 경로로만 동작)
 - 따라서 현재 서버의 경로 + 파일의 위치 조합으로 절대경로를 만들어 조합.
 - **_dirname**과, 각각의 html 파일 경로를 합쳐 사용

```
app.get("/", (req, res) => {
  return res.sendFile(__dirname + "/views/index.html");
});

app.get("/login", (req, res) => {
  return res.sendFile(__dirname + "/views/login.html");
}

app.get("/whoami", (req, res) => {
  return res.sendFile(__dirname + "/views/whoami.html");
})
```

app.get("/라우터주소", 콜백함수)

- 해당 라우터 주소에 get 방식으로 도착을 하면 콜백 함수를 실행
- /에 도착하면 index.html 파일을 보여준다.
- /login에 도착하면 login.html파일을 보여준다.
- /whoami에 도착하면 whoami.html을 보여준다.

```
app.get("/", (req, res) => {
  return res.sendFile(__dirname + "/views/index.html");
});

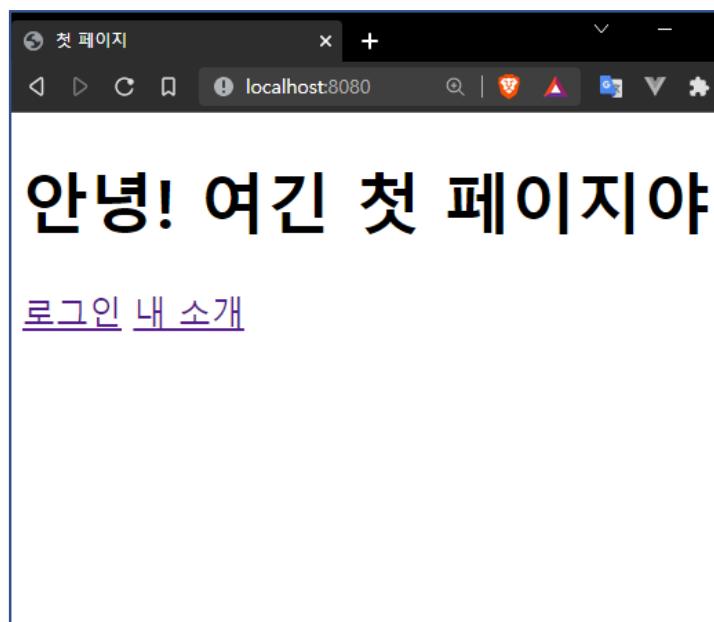
app.get("/login", (req, res) => {
  return res.sendFile(__dirname + "/views/login.html");
}

app.get("/whoami", (req, res) => {
  return res.sendFile(__dirname + "/views/whoami.html");
})
```

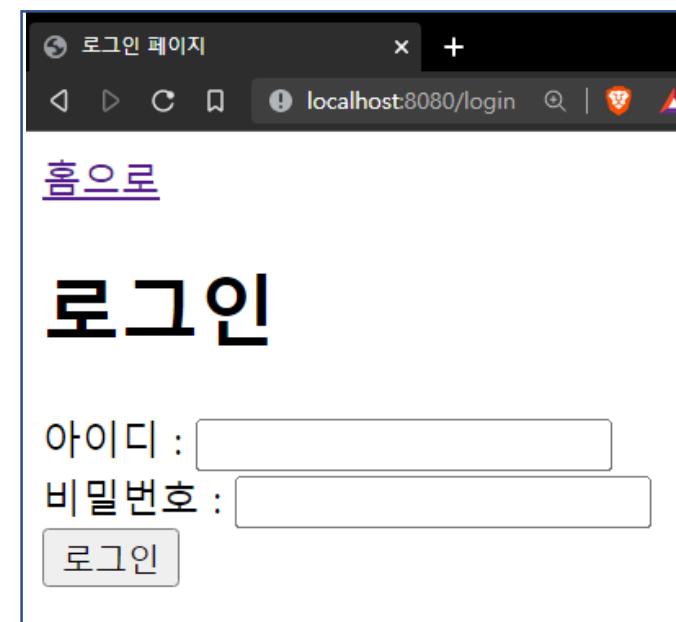
유의사항

- url에 써준 내용들은 파일의 경로가 아니라 요청의 경로
- /로 요청을 보내면 app.get("/", 콜백함수) / 부분에 요청을 보내는것
- /login으로 요청을 보내면 app.get("/login", 콜백함수) /login 부분에 요청을 보내는것

/



/login



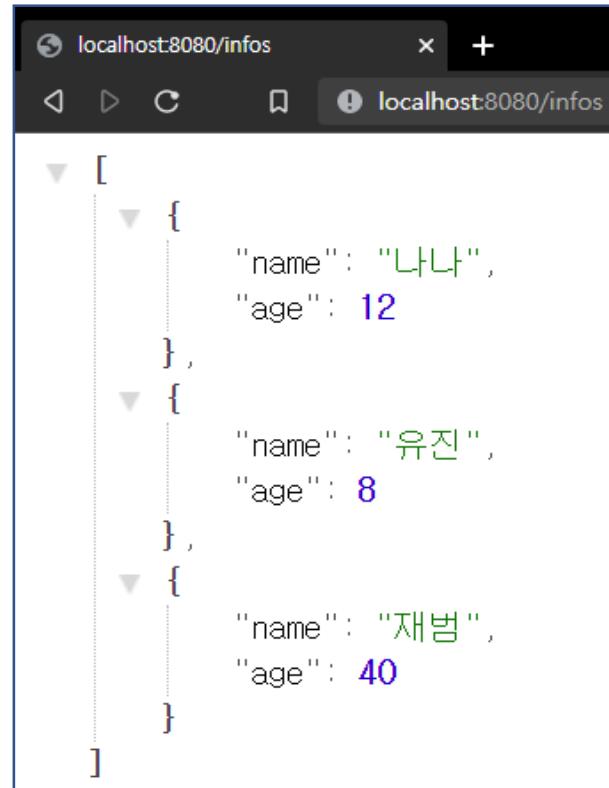
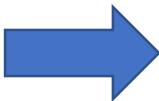
/whoami



이번엔 파일이 아니라 JSON을 리턴해보기

- /infos에 요청을 보낼시에 return res.json으로 배열을 리턴

```
13 app.use(express.json());
14
15 const infos = [
16   {
17     name: "나나",
18     age: 12,
19   },
20   {
21     name: "유진",
22     age: 8,
23   },
24   {
25     name: "재범",
26     age: 40
27   },
28 ]
29
30 app.get("/infos", (req, res) => {
31   return res.json(infos);
32 })
```



배열 안에 동일한 타입의 객체를 정의

/infos 접속 시, res.json() 을 사용해
배열을 JSON 형식으로 리턴

해당 경로 접속시 명시된 리턴을 수행

- **/infos/names**
 - infos 중 이름만 파싱한 배열을 JSON 리턴
 - map 활용해보기
 - `Array.map((element) => { return element.name })`
- **/infos/ages**
 - infos 중 나이만 파싱한 배열을 JSON 리턴
- **/infos/0**
 - infos 배열의 0번째 리턴
- **/infos/1**
 - infos 배열의 1번째 리턴
- **/infos/2**
 - infos 배열의 2번째 리턴

```
const infos = [  
  {  
    name: "나나",  
    age: 12,  
  },  
  {  
    name: "유진",  
    age: 8,  
  },  
  {  
    name: "재범",  
    age: 40,  
  },  
];
```

```
[  
  "나나",  
  "유진",  
  "재범"  
]
```

/infos/names

```
{  
  "name": "유진",  
  "age": 8  
}
```

/infos/1

9장. Express 와 MySQL 연동

챕터의 포인트

- MySQL 설치 (AWS)
- mysql2 패키지 설치

MySQL 설치 (AWS)

DB 서버 구축하기

- 이번엔 Windows 가 아니라,
AWS Instance 에 DB 서버를 구성하여
MySQL 원격 접속을 시도해보자.



MobaXterm에 접속

- AWS 서버에 ssh 접속
- \$ sudo apt update
- \$ sudo apt install mysql-server
- \$ mysql --version

```
ubuntu@ip-172-31-45-229:~$ mysql --version
mysql Ver 14.14 Distrib 5.7.38, for Linux (x86_64) using EditLine wrapper
```

이후, 보안 설정을 진행해주자.

- **\$ sudo mysql_secure_installation**

- 비밀번호 유효성 검사 플러그인을 추가해 MySQL 의 보안을 강화할 것인지 묻는 창.
- 당장 필요없으므로 엔터로 넘긴다.

```
ubuntu@ip-172-31-40-222:~$ sudo mysql_secure_installation
```

```
Securing the MySQL server deployment.
```

```
Connecting to MySQL using a blank password.
```

```
VALIDATE PASSWORD PLUGIN can be used to test passwords  
and improve security. It checks the strength of password  
and allows the users to set only those passwords which are  
secure enough. Would you like to setup VALIDATE PASSWORD plugin?
```

```
Press y|Y for Yes, any other key for No:
```

비밀번호 지정

- root 패스워드 지정을 위해, 적당히 긴 패스워드를 두 번 입력
- 어차피, root 계정은 특별한 조치를 취하지 않는 한 비밀번호로 로그인할 수 없다.
- MySQL 사용 시엔 root 계정을 사용하지 않는 것이 관습
- 별도의 관리자를 만들 예정

Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 50

Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y

기타 옵션 설정

- **anonymous user 삭제 여부**
 - y 를 눌러 임의의 유저를 제거한다

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

기타 옵션 설정

- root 계정의 원격접속을 허용하는지 체크
 - 기본적으로 root 계정은 원격접속을 허용하지 않는게 원칙이므로 y

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

기타 옵션 설정

- test db를 제공하는데 이를 삭제할건지 묻는 질문
 - 사용하지 않을 예정이므로 y

```
Remove test database and access to it? (Press y\Y for Yes, any other key for No) : y
- Dropping test database...
Success.

- Removing privileges on test database...
Success.
```

기타 옵션 설정

- privilege 를 reload 해서, 변경한 설정을 즉시 적용할것인지 여부설정
 - y 입력
 - All done! 이 나오게 되면 기초 설정은 끝이 난다.

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? (Press y\Y for Yes, any other key for No) : y
Success.

All done!
ubuntu@ip-172-31-40-222:~\$ █

\$ sudo mysql

- 새로운 관리자 계정을 만들기 위해 mysql root 계정으로 로그인

```
ubuntu@ip-172-31-40-222:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.38-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

관리자 권한 부여

- **GRANT ALL PRIVILEGES ON *.* TO '유저이름'@'%' IDENTIFIED BY '비밀번호';**
 - GRANT ALL PRIVILEGES ON *.* TO 'ssafy'@'%' IDENTIFIED BY 'ssafy_8_A';
 - 아이디 **ssafy** 비밀번호 **ssafy_8_A**
 - 비밀번호는 문자, 대문자, 숫자를 같이 써야한다
 - ON ***.*** TO ***** 은 DB 이름, 뒤에 *****은 권한 내용이다.
 - 즉, 해당 유저에, 모든 DB 의 모든 권한을 부여한다는 뜻
 - % 는 모든 외부 IP 를 뜻한다. 즉, 전세계에서 연결 허용
 - FLUSH PRIVILEGES;
 - 변경한 권한 적용

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ssafy'@'%' IDENTIFIED BY 'ssafy_8th_A';
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

관리자 계정 로그인

- mysql에서 exit 입력후 ubuntu shell로 돌아가기
- 방금 만든 관리자 계정으로 로그인
 - sudo mysql -u 아이디 -p
 - 작성후 비밀번호
 - 아이디 ssafy
 - 비밀번호 ssafy_8_A

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> exit
Bye
ubuntu@ip-172-31-30-8:~$ sudo mysql -u ssafy -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.7.39-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

DB 생성하기

- CREATE DATABASE 디비이름 DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
 - CREATE DATABASE jony DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
 - DB 를 생성하는데, 한글이 안 깨지도록 utf8 인코딩을 적용한 DB 생성
 - DB 생성 후 show DATABASES로 DB가 생성되었는지 조회
 - use jony를 사용해서 해당 데이터베이스를 사용하도록 명시 (workbench의 데이터베이스 더블클릭과 동일하다)

```
mysql> CREATE DATABASE jony DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
Query OK, 1 row affected (0.00 sec)

mysql> show DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| jony          |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.01 sec)

mysql> use jony;
Database changed
```

```
CREATE TABLE `menus` (
  `menu_id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `menu_name` VARCHAR(20) NOT NULL,
  `menu_description` TEXT NOT NULL,
) default character set utf8 collate utf8_general_ci;
```

```
INSERT INTO `menus`
(`menu_name`, `menu_description`)
VALUES
("아이스 아메리카노", "여름엔 아아가 진리");
```

```
INSERT INTO `menus`
(`menu_name`, `menu_description`)
VALUES
("카페라떼", "Latte is horse");
```

```
INSERT INTO `menus`
(`menu_name`, `menu_description`)
VALUES
("복숭아 아이스티", "내 입안 복숭아향 가득");
```

데이터가 잘 들어갔는지 확인해보자

- **show tables;**
- **select * from menus;**
- 확인 되었으면 **exit**로 DB 접속 종료

```
mysql> show tables;
+-----+
| Tables_in_jony |
+-----+
| menus          |
+-----+
1 row in set (0.00 sec)

mysql> select * from menus;
+-----+-----+-----+
| menu_id | menu_name           | menu_description      |
+-----+-----+-----+
| 1       | 아이스 아메리카노   | 여름엔 아아가 진리  |
| 2       | 카페라떼             | Latte is horse      |
| 3       | 복숭아 아이스티     | 내 입안 복숭아향 가득|
+-----+-----+-----+
3 rows in set (0.01 sec)
```

Listen IP 대역폭 변경

- 현재 MySQL 은 오로지 localhost 에서만 사용하도록 설정되어있다.
- 모든 IP 에서 원격접속할 수 있도록 바꿔보자.
- `sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf`

```
ubuntu@ip-172-31-40-222:~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf █
```

mysqld.cnf 변경내용

- **bind-address = 127.0.0.1**
 - bind-address = 0.0.0.0로 변경
 - 127.0.0.1(localhost)에서만 접근허용 됐던 DB를 모든 곳에서 접속이 가능하게 변경시켜주는것
 - 수정 후 Ctrl + O Enter
 - Ctrl + X로 저장후 나가기
 - cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep bind
 - 명령어로 수정된 bind-address 파트만 조회

```
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
# bind-address          = 127.0.0.1  
bind-address          = 0.0.0.0
```

```
ubuntu@ip-172-31-30-8:~$ cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep bind  
bind-address          = 0.0.0.0
```

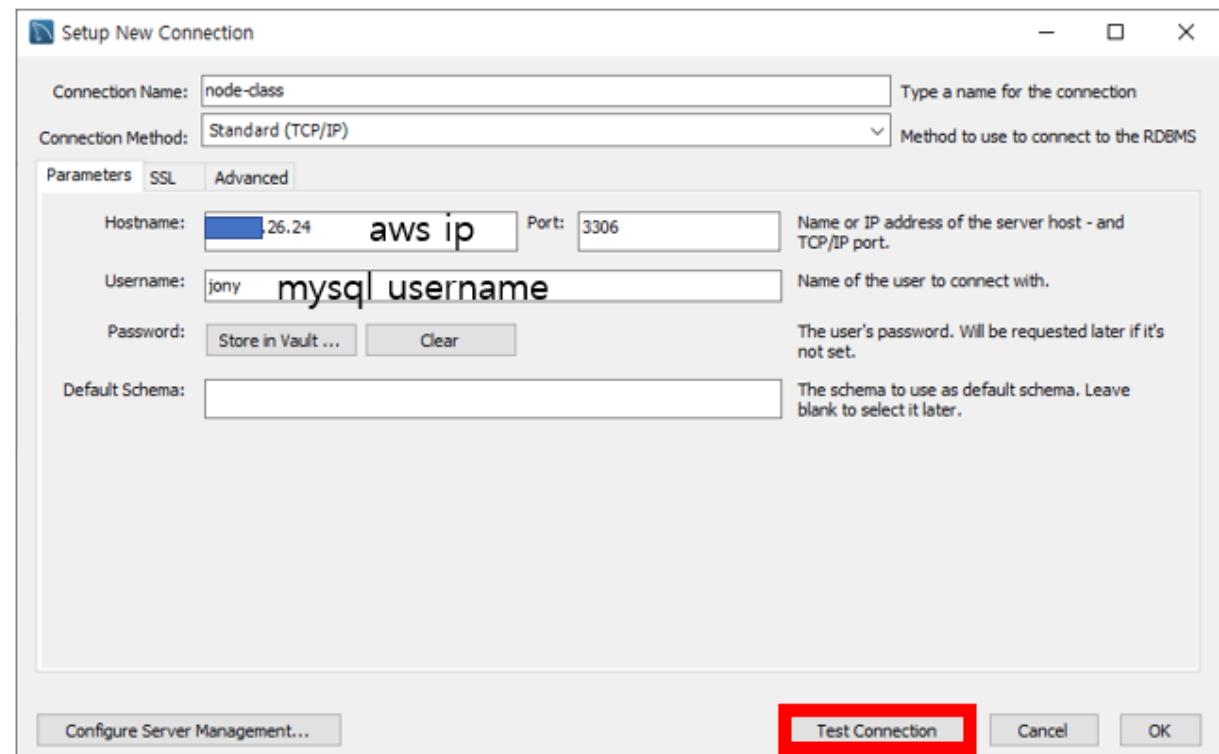
설정 내용 적용하기

- 변경된 내용을 적용하기 위해 MySQL 서버를 재시작하는 작업이 필요하다
- `sudo service mysql restart`
- 재시작 까지 진행하면 이제 외부에서 DB 접속을 위한 작업은 끝이난다.

```
ubuntu@ip-172-31-40-222:~$ sudo service mysql restart
ubuntu@ip-172-31-40-222:~$ █
```

MySQL Workbench 기동

- DB 외부 접속 셋팅이 끝났기 때문에 Window Workbench에서도 접근이 가능
 - hostname : AWS의 IP 주소
 - username 외부 접속을 위해 가입한 아이디
 - 아이디 ssafy
 - 비밀번호 ssafy_8_A



눌렀을 때 비밀번호 창 뜨면 성공

AWS DB서버 접속 완료

- 방금 만든 데이터베이스 jony 와, menus 더미데이터 확인하기
- 이로서, 원격에서 MySQL 을 외부에서 접근하는 법을 마쳤다.

The screenshot shows the MySQL Workbench interface with a successful connection to an AWS database. The left sidebar displays the Navigator with the Schemas section expanded, showing the 'jony' schema containing a 'menus' table, along with 'Views', 'Stored Procedures', and 'Functions'. Below the schemas is the 'sys' schema. The middle section shows the 'Query 1' tab with the query `SELECT * FROM jony.menus;` and its results. The results grid displays four rows of menu data:

	menu_id	menu_name	menu_description	menu_img_link
▶	1	아이스 아메리카노	여름엔 아아가 진리	/menus/ice-americano.jpg
▶	2	카페라떼	Latte is horse	/menus/cafe-latte.jpg
▶	3	복숭아 아이스티	내 입안 복숭아향 가득	/menus/peach-icetea.jpg
*	NULL	NULL	NULL	NULL

mysql2 패키지 설치

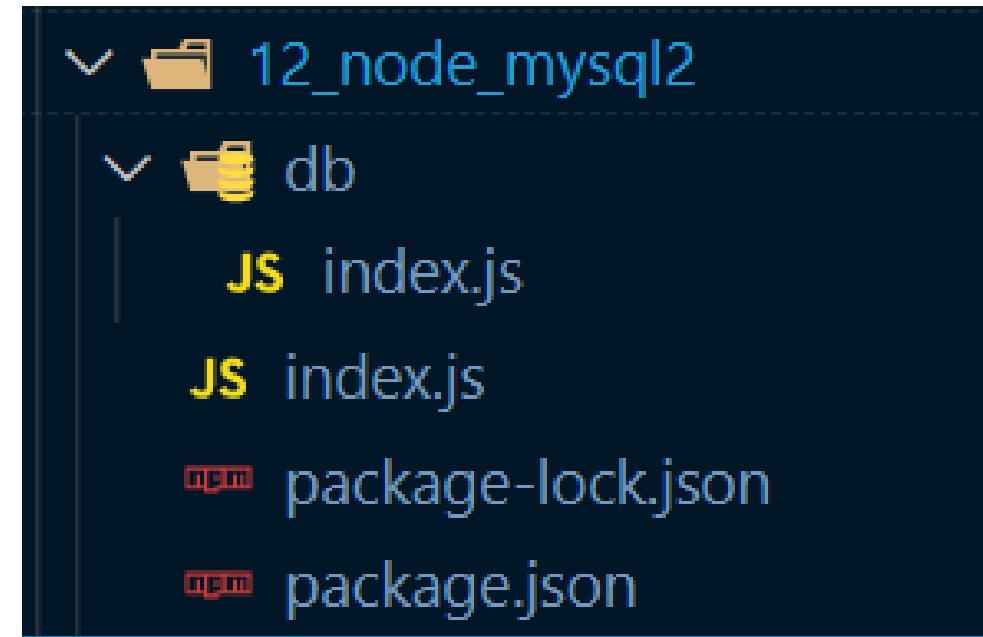
mysql2를 사용하는 이유

- node에서는 기본적으로 mysql에 접근하기 위해서는 라이브러리 사용이 필요
- mysql2라는 라이브러리를 사용한다.
 - 왜 mysql1 이 아니라 2를 사용할까?
 - mysql2부터 promise 방식이 사용 가능 및 개편된 버전
 - npm i mysql2

```
"dependencies": {  
    "cors": "^2.8.5",  
    "express": "^4.18.1",  
    "mysql2": "^2.3.3"  
}
```

폴더 구조 확인하기

- npm init 명령어 실행
- npm i mysql2 express cors
- db 폴더 생성 후 그 안에 index.js 생성



db 폴더 내 index.js를 수정

- **host:** 접속하기 위한 ip 주소 작성
 - localhost의 db에 접근하기 위해서는 localhost 나 127.0.0.1 을 작성
 - 외부 DB를 생성했기 때문에 AWS의 주소를 작성
- **user**
 - DB에 접근하기 위해 생성해둔 ID
 - ssafy
- **password**
 - DB에 접근하기 위해 생성해둔 PASSWORD
 - ssafy_8_A
- **database**
 - 접근하기 위한 DB(스키마) 의 이름

```
const mysql = require("mysql2/promise");

const pool = mysql.createPool({
    // aws ip
    host: "3.39.183.206",
    // mysql username
    user: "ssafy",
    // mysql user password
    password: "ssafy_8_A",
    // db name
    database: "jony",
    waitForConnections: true,
    connectionLimit: 10,
    queueLimit: 0,
});

module.exports = { pool };
```

DB 접속 코드 작성

- db 폴더에는 index.js로 서버 접속을 위한 셋팅이 되어있다.
- 해당 부분에서 pool 부분만 사용하기 위해 가져온 코드
 - const { pool } = require('./db')
 - db 폴더의 index.js 의 객체에서 pool 내용만 따로 가져와서 사용하겠다.
 - const db = require('./db');
 - const pool = db.pool 과 같다

```
1  const express = require("express");
2  const { pool } = require("./db");
3
4  const app = express();
5  const PORT = 8080;
```

DB 접속 코드 작성

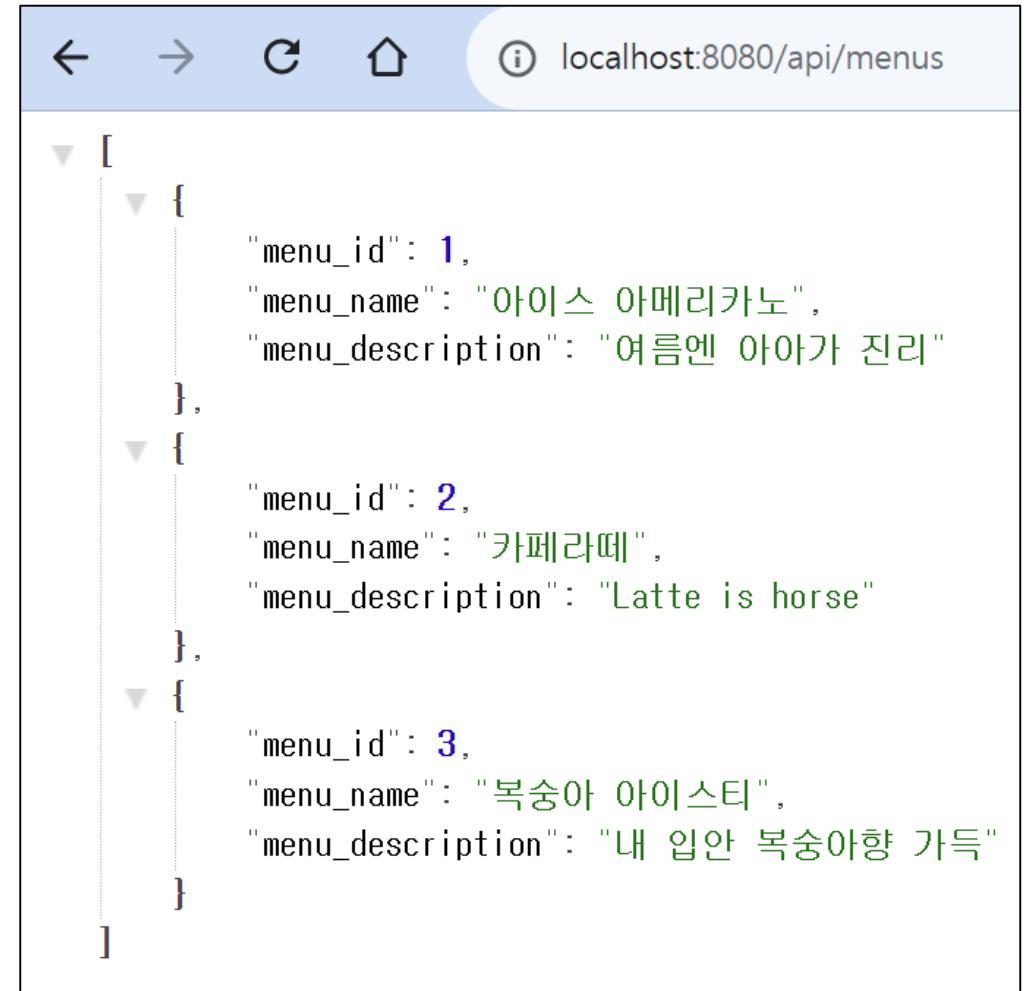
- **async await 활용**

- mysql2는 promise 형식으로 값을 리턴하기 때문에 async/await 활용이 가능
- pool.query 안에 SQL 쿼리문을 작성한다.
- 작성후 첫번째 배열을 리턴

```
app.get("/api/menus", async (req, res) => {
  try {
    const data = await pool.query("SELECT * FROM menus");
    if (data[0]) {
      return res.json(data[0]);
    }
  } catch (error) {
    return res.json(error);
  }
});
```

결과 확인

- localhost:8080/api/menus 접근하기
- 해당 정보를 리턴한다면 성공

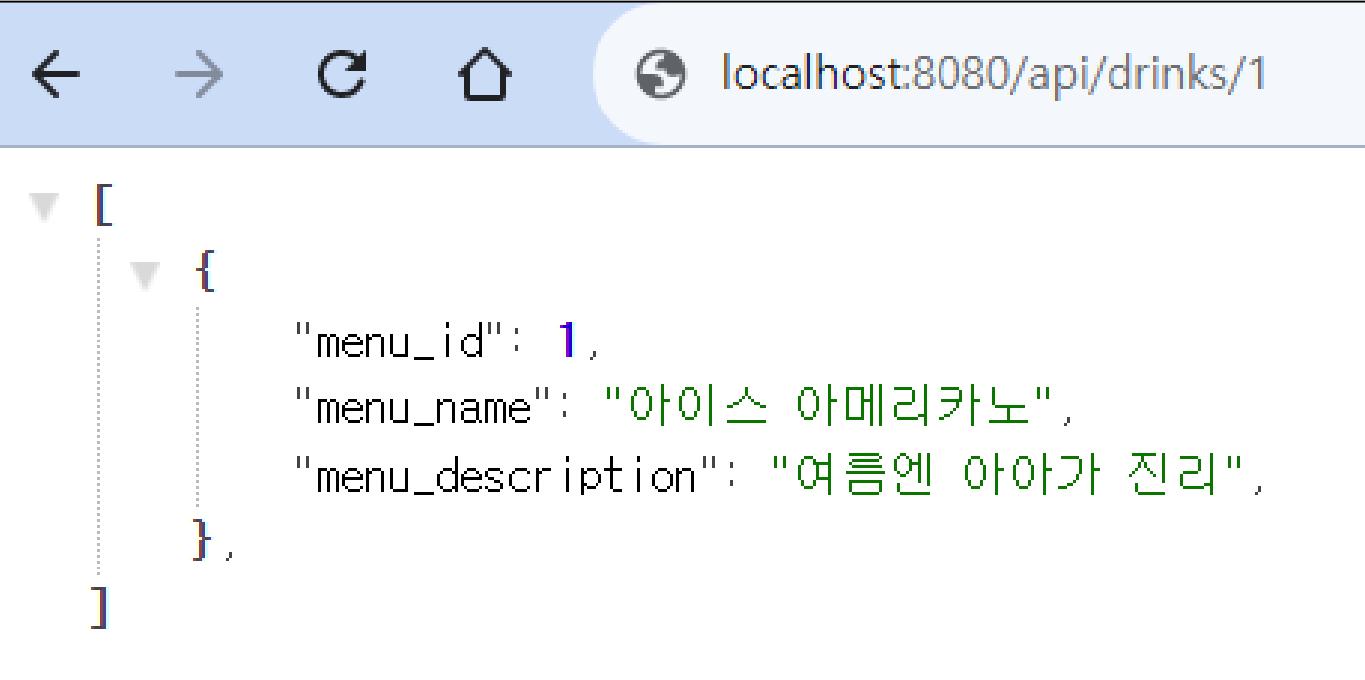


A screenshot of a web browser window displaying a JSON array of menu items. The URL in the address bar is "localhost:8080/api/menus". The JSON data is as follows:

```
[{"menu_id": 1, "menu_name": "아이스 아메리카노", "menu_description": "여름엔 아아가 진리"}, {"menu_id": 2, "menu_name": "카페라떼", "menu_description": "Latte is horse"}, {"menu_id": 3, "menu_name": "복숭아 아이스티", "menu_description": "내 입안 복숭아향 가득"}]
```

DB 연동 다시 진행하기

- 처음부터 DB 연동까지 다시 진행한다.
- app.get("/api/drinks/1") 요청시 id가 1인 데이터값만 가져올수 있도록 하기



A screenshot of a web browser window displaying a JSON object. The address bar shows "localhost:8080/api/drinks/1". The JSON content is:

```
[{"menu_id": 1, "menu_name": "아이스 아메리카노", "menu_description": "여름엔 아아가 진리"}, ]
```

10장. REST API

챕터의 포인트

- REST API 개념
- Postman
- HTTP Response Status Code
- params, query, body

REST API 개념

- 리모콘 "전원버튼" 누르기
TV 가 켜진다.
- 리모콘 "음량버튼" 을 누르기
 - 소리를 조절한다.
- 리모콘 "채널버튼" 을 누르기
 - 채널이 바뀐다.



인터페이스란?

- 사람은 “리모컨”만 있으면 TV 세부 사용법을 알 필요가 없다.
- 사람이 TV를 제어할 수 있도록 해주는 중간 다리 역할을 “인터페이스” 역할이라고 한다.
- **사용자가 쉽게 동작 및 사용하는데 도움을 주는 시스템**을 뜻한다.
HW Interface 은 물리적 접점을 뜻한다.
 - SW Interface 는 API 소스코드 접근 형태, 추상화 구현 등을 나타낸다.

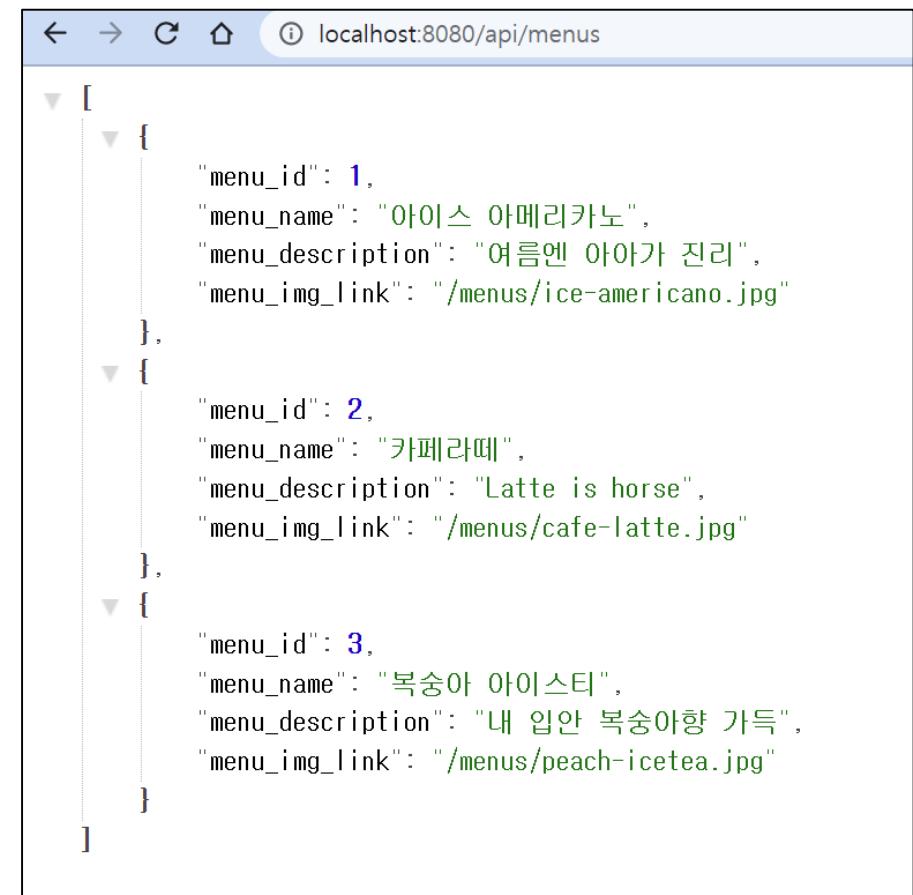


API

- Application Programming Interface
- 즉, 인터페이스를 소스코드 형태로 구현한 것을 의미

GET /api/menus

모든 메뉴 가져오기

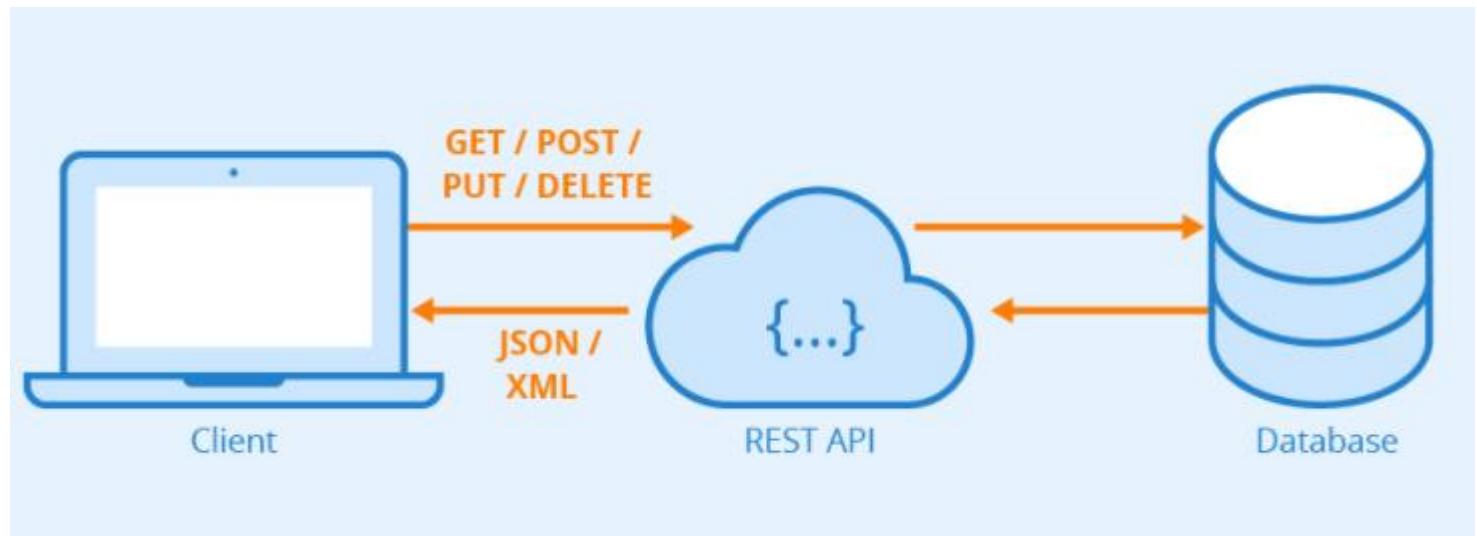


A screenshot of a web browser window displaying a JSON array of menu items. The URL in the address bar is 'localhost:8080/api/menus'. The JSON data shows three menu entries, each with an ID, name, description, and image link.

```
[{"menu_id": 1, "menu_name": "아이스 아메리카노", "menu_description": "여름엔 아아가 진리", "menu_img_link": "/menus/ice-americano.jpg"}, {"menu_id": 2, "menu_name": "카페라떼", "menu_description": "Latte is horse", "menu_img_link": "/menus/cafe-latte.jpg"}, {"menu_id": 3, "menu_name": "복숭아 아이스티", "menu_description": "내 입안 복숭아향 가득", "menu_img_link": "/menus/peach-icetea.jpg"}]
```

REST API(Representational State Transfer)

- 소프트웨어 개발 아키텍처의 한 형식
- 자원을 이름으로 구분하고 자원의 상태를 주고 받는 모든 것
- 일반적으로 REST라 하면 **HTTP를 통해 CRUD를 실행하는 API**



GET

- 데이터를 읽거나 검색을 할 때 주로 사용된다.
- URL에 데이터를 붙여서 보낸다.
- 캐싱이 가능하다.
- ex) 영화 목록 받아 오기, 검색 결과 확인하기

DELETE	/status/{codes}	Return status code or random status code if more than one are given
GET	/status/{codes}	Return status code or random status code if more than one are given
PATCH	/status/{codes}	Return status code or random status code if more than one are given
POST	/status/{codes}	Return status code or random status code if more than one are given
PUT	/status/{codes}	Return status code or random status code if more than one are given

POST

- 새로운 리소스를 생성할 때 사용한다.
- URL이 아닌 BODY 부분에 데이터를 넣어서 보낸다.
- ex) 게시글 작성하기

DELETE	/status/{codes}	Return status code or random status code if more than one are given
GET	/status/{codes}	Return status code or random status code if more than one are given
PATCH	/status/{codes}	Return status code or random status code if more than one are given
POST	/status/{codes}	Return status code or random status code if more than one are given
PUT	/status/{codes}	Return status code or random status code if more than one are given

- **PUT**

- 전체 데이터를 변경 및 갱신할 때 사용한다
- ex) 회원 정보 수정하기, 게시글 수정하기

- **PATCH**

- 일부 데이터를 변경 할 때 사용한다
- 전체를 갱신하는 PUT과 다르게 일부를 수정하기 때문에
UPDATE 부분에 더 적합하다

DELETE	/status/{codes}	Return status code or random status code if more than one are given
GET	/status/{codes}	Return status code or random status code if more than one are given
PATCH	/status/{codes}	Return status code or random status code if more than one are given
POST	/status/{codes}	Return status code or random status code if more than one are given
PUT	/status/{codes}	Return status code or random status code if more than one are given

DELETE

- 리소스를 삭제 할 때 사용한다.
- ex) 게시글 삭제하기

DELETE /status/{codes} Return status code or random status code if more than one are given

GET /status/{codes} Return status code or random status code if more than one are given

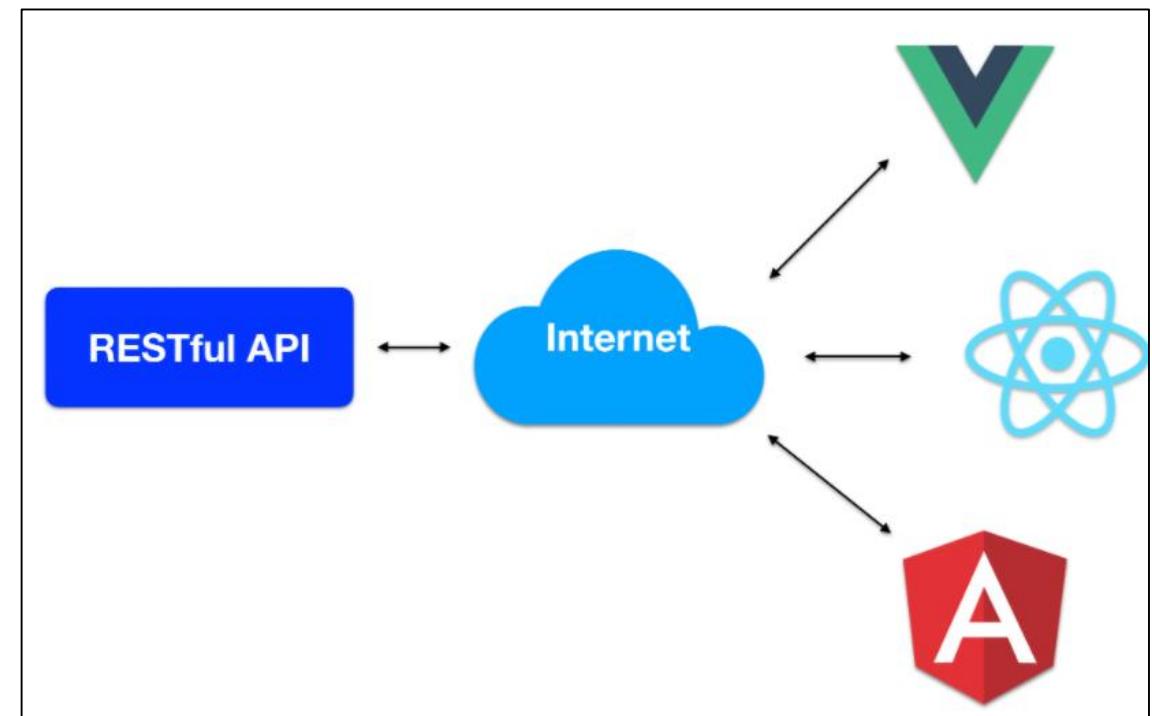
PATCH /status/{codes} Return status code or random status code if more than one are given

POST /status/{codes} Return status code or random status code if more than one are given

PUT /status/{codes} Return status code or random status code if more than one are given

RESTful API

- REST 원리를 따르는 시스템
- 이해하기 쉽고 사용하기 쉬운 REST API를 만드는 것
- REST API 설계법에 근거한 REST API 설계하기



잘못된 API 설계법

- **유저 정보 관련 설계하기**
 - 유저 전체 정보 조회
 - GET /users/all-get
 - 유저 등록
 - POST /users/adduser
 - 특정 유저 조회
 - GET /users/:id/get-information
 - 특정 유저 수정
 - PATCH /users/:id/update
 - 특정 유저 삭제
 - DELETE /users/:id/delete

좋은 API 설계하기

- **유저 정보 조회**
 - 유저 전체 정보 조회
 - GET /users
 - 유저 등록
 - POST /users
 - 특정 유저 조회
 - GET /users/:id
 - 특정 유저 수정
 - PATCH /users/:id
 - 특정 유저 삭제
 - DELETE /users/:id
- 대상에 대한 행동은 모두 HTTP REQUEST METHOD로 표시한다.
 - 행동과 리소스를 구별해서 설계하는것이 핵심이다.
 - 리소스 : 유저
 - 행동: 조회(GET), 등록(POST), 수정(PATCH), 삭제(DELETE)

POST 주요 코드

- **app.use(express.json())**
 - 클라이언트에서 JSON Format 의 형태로 서버에 요청을 보낼수 있도록 해주는 설정
- **req.body**
 - POST 요청시 http body로부터 받아오는 내용

```
11  app.use(express.json());
31  app.post("/api/menu", async (req, res) => {
32    try {
33      const data = await pool.query(`
34        INSERT INTO menus (menu_name, menu_description, menu_img_link)
35        VALUES
36        ("${req.body.menu_name}", "${req.body.menu_description}", "${req.body.menu_img_link}")
37        `);
38      return res.json(data);  // 쿼리문 실행 결과 리턴
39    } catch (error) {
40      return res.json(error);
41    }
42  });
```

POST 주요 코드

- **app.use(express.json())**
 - 클라이언트에서 JSON Format 의 형태로 서버에 요청을 보낼수 있도록 해주는 설정
- **req.body**
 - POST 요청시 http body로부터 받아오는 내용

```
app.use(express.json());
app.post("/api/menus", async (req, res) => {
  try {
    const data = await pool.query(`INSERT INTO menus (menu_name, menu_description) VALUES (?, ?)`, [req.body.menu_name, req.body.menu_description]);
    // 위 or 아래 선택

    // const data = await pool.query(`INSERT INTO menus (menu_name, menu_description, menu_img_link) VALUES ("${req.body.menu_name}", "${req.body.menu_description}")`);
    // `);
    return res.json(data);
  } catch (error) {
    return res.json(error);
  }
});
```

일반 문자열로 넣기

- `INSERT INTO menus`에 백틱을 활용해서 변수 형태로 한꺼번에 넣는 방법

```
const data = await pool.query(`  
    INSERT INTO menus (menu_name, menu_description)  
    VALUES  
    ("${req.body.menu_name}", "${req.body.menu_description}")  
`);
```

? 활용하기

- ?가 들어간 부분들은 pool.query(“쿼리문”, [?에 들어갈 내용]) 식으로 변경이 가능하다.
- 문자열이 섞여서 복잡할 경우 아래와 같이 해결한다.

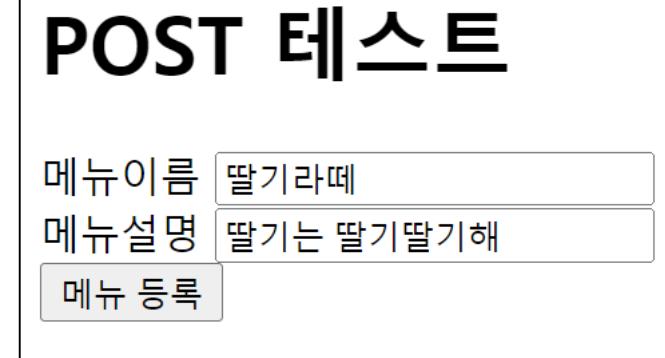
```
const data = await pool.query(`INSERT INTO menus (menu_name, menu_description) VALUES (?, ?)`,  
| [req.body.menu_name, req.body.menu_description]);
```

API 서버에 접속하는 클라이언트를 만들기

- 단, 현재 작업 디렉터리 이외의 공간에 만들 것
 - ex) 바탕화면

```
<h1>POST 테스트</h1>
<div>
  <div>
    <label for="menu_name">메뉴이름</label>
    <input id="menu_name" type="text" />
  </div>
  <div>
    <label for="menu_description">메뉴설명</label>
    <input id="menu_description" type="text" />
  </div>

  <div>
    <button>메뉴 등록</button>
  </div>
</div>
```



API 서버에 접속하는 클라이언트 만들기

- axios.post로 post 요청을 보내면 서버도 post로 요청을 받는다.
- axios.post("주소", { 넘겨줄 객체값 })

```
<script src="https://cdn.jsdelivr.net/npm/axios@0.27.2/dist/axios.min.js"></script>
<script>
  const btn = document.querySelector("button");
  btn.addEventListener("click", async () => {
    const menu_name = document.querySelector("#menu_name").value;
    const menu_description =
      document.querySelector("#menu_description").value;
    const response = await axios.post("http://localhost:8080/api/menu", {
      menu_name: menu_name,
      menu_description: menu_description,
    });
    console.log(response.data);
  });
</script>
```



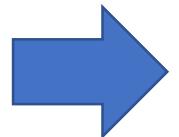
▼ 요청 페이로드 소스 보기
▼ {menu_name: "딸기", menu_description: "딸기는딸기딸기"}
 menu_description: "딸기는딸기딸기"
 menu_name: "딸기"

성공 후 console.log(response.data) 결과값 보기

- **affectedRows:1**

- DB에 영향을 받은 row의 수를 의미
- 하나의 row만 추가했기 때문에 1이 나오게 된다.
- 알맞게 요청이 성공했다는 의미

```
▼ (2) [ {...}, null ] i
  ▼ 0:
    affectedRows: 1
    fieldCount: 0
    info: ""
    insertId: 4
    serverStatus: 2
    warningStatus: 0
    ▶ [[Prototype]]: Object
  1: null
  length: 2
  ▶ [[Prototype]]: Array(0)
```



```
mysql> select * from menus;
+-----+-----+-----+
| menu_id | menu_name | menu_description |
+-----+-----+-----+
| 1 | 아이스 아메리카노 | 여름엔 아아가 진리 |
| 2 | 카페 라떼 | Latte is horse |
| 3 | 보수아 아이스티 | 내 인아 보수아 향 가득 |
| 4 | 딸기 라떼 | 딸기는 딸기 딸기 해 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

처음부터 다시 셋팅하기

- npm init 을 통해 package.json 생성부터
- node에서 DB 연결 + 클라이언트 작성까지 처음부터 다시 진행해보기

Postman

테스트를 위해 매번 클라이언트를 만드는 건 불편하다

- GET 의 경우, 브라우저 접속만 하면 되지만
POST, PUT, PATCH, DELETE 테스트를 위해선 클라이언트 코드를 따로 작성해야 한다.
- 그러나, 서버 코드는 클라이언트 코드 없이도 테스트 가능해야 한다!

개발한 API를 테스트할 수 있는 플랫폼

- <https://www.postman.com/>

The screenshot shows the official Postman website homepage. At the top, there's a dark header bar with the Postman logo (an orange circle with a white pen nib icon), navigation links for Product, Use Cases, Pricing, Enterprise, API Network, and Learning Center, and buttons for Dashboard and Download. The main title "The Collaboration Platform for API Development" is prominently displayed in large white font. Below it, a subtitle reads: "Simplify each step of building an API and streamline collaboration so you can create better APIs—faster." A large orange "Learn More" button is located at the bottom left. To the right, there's a stylized illustration of three robotic or satellite-like entities in space, connected by lines, set against a dark background with stars.

Workspaces 클릭

The screenshot shows the Postman application interface. At the top, there is a navigation bar with links for Home, Workspaces (which is currently selected and highlighted with a red box), API Network, and Explore. To the right of the navigation is a search bar labeled "Search Postman".

The main content area features a greeting message "Good afternoon, 이온유!" followed by the instruction "Pick up where you left off.". Below this, there is a section titled "Recently visited workspaces" which lists a workspace named "test".

On the left side of the interface, there is a sidebar with three main options: "Workspaces", "Integrations", and "Reports". The "Workspaces" option is highlighted with a red box.

Below the sidebar, there is a section titled "Postman works best with teams" with the sub-instruction "Collaborate in real-time and establish a single source of truth for all API workflows." It includes a "Create Team" button and a "Create New" button under the "Get started with Postman" section.

The "Get started with Postman" section also contains two other buttons: "Start with something new" and "Import an existing file".

Workspace 생성

The screenshot shows the 'Your Workspaces' page in Postman. At the top left, there are navigation links for 'Home' and 'Workspaces'. Below that, the title 'Your Workspaces' is displayed, followed by the subtitle 'A directory of your workspaces.' On the right side of the page, there is an orange 'Create Workspace' button, which is highlighted with a red rectangular border. Below the title, there are several filter and sorting options: 'Search workspaces', 'Visibility', 'Created ...', and 'Sort by A to Z'. The main content area lists two workspaces: 'My Workspace' and 'test'. Each workspace entry includes a small user icon and a circular icon with a gear and a play button.

Home / Workspaces

Your Workspaces

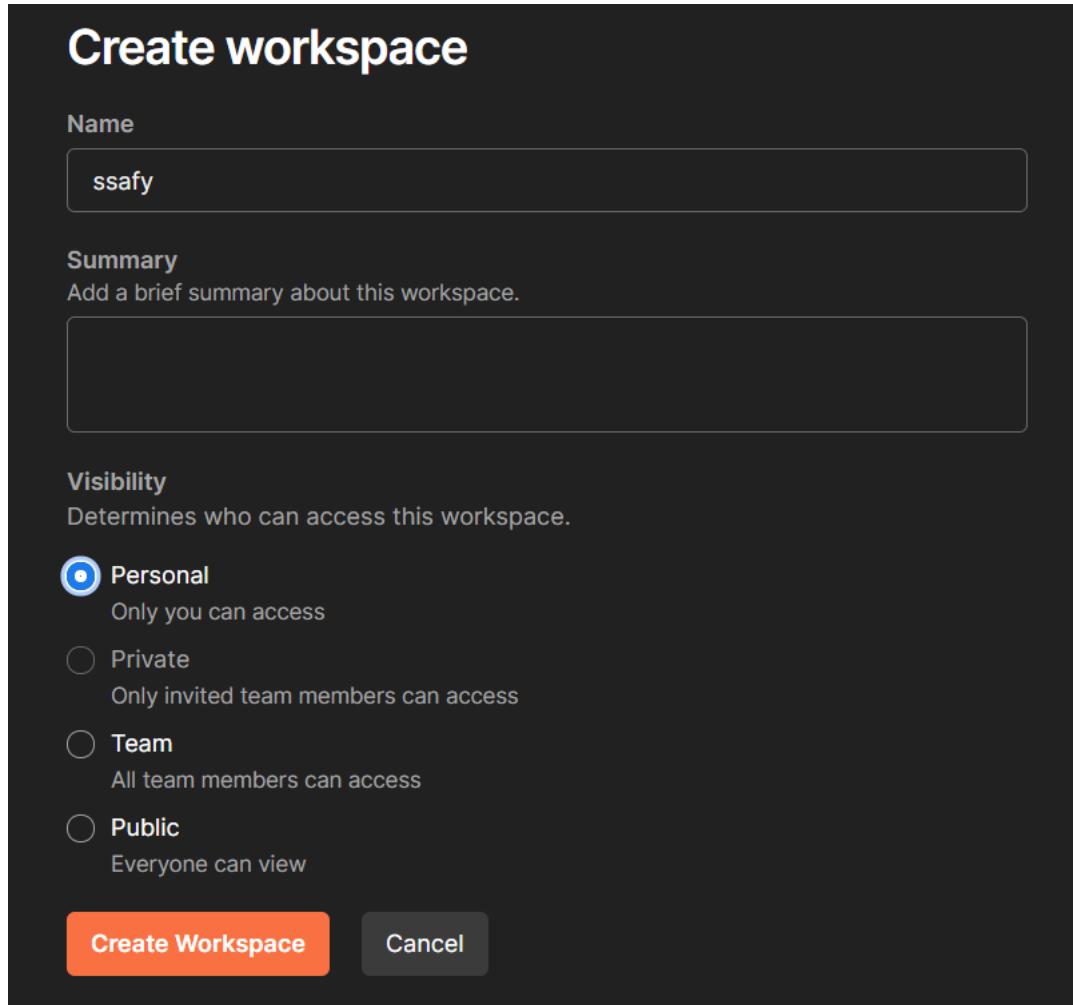
A directory of your workspaces.

Search workspaces Visibility Created ... Sort by A to Z

Create Workspace

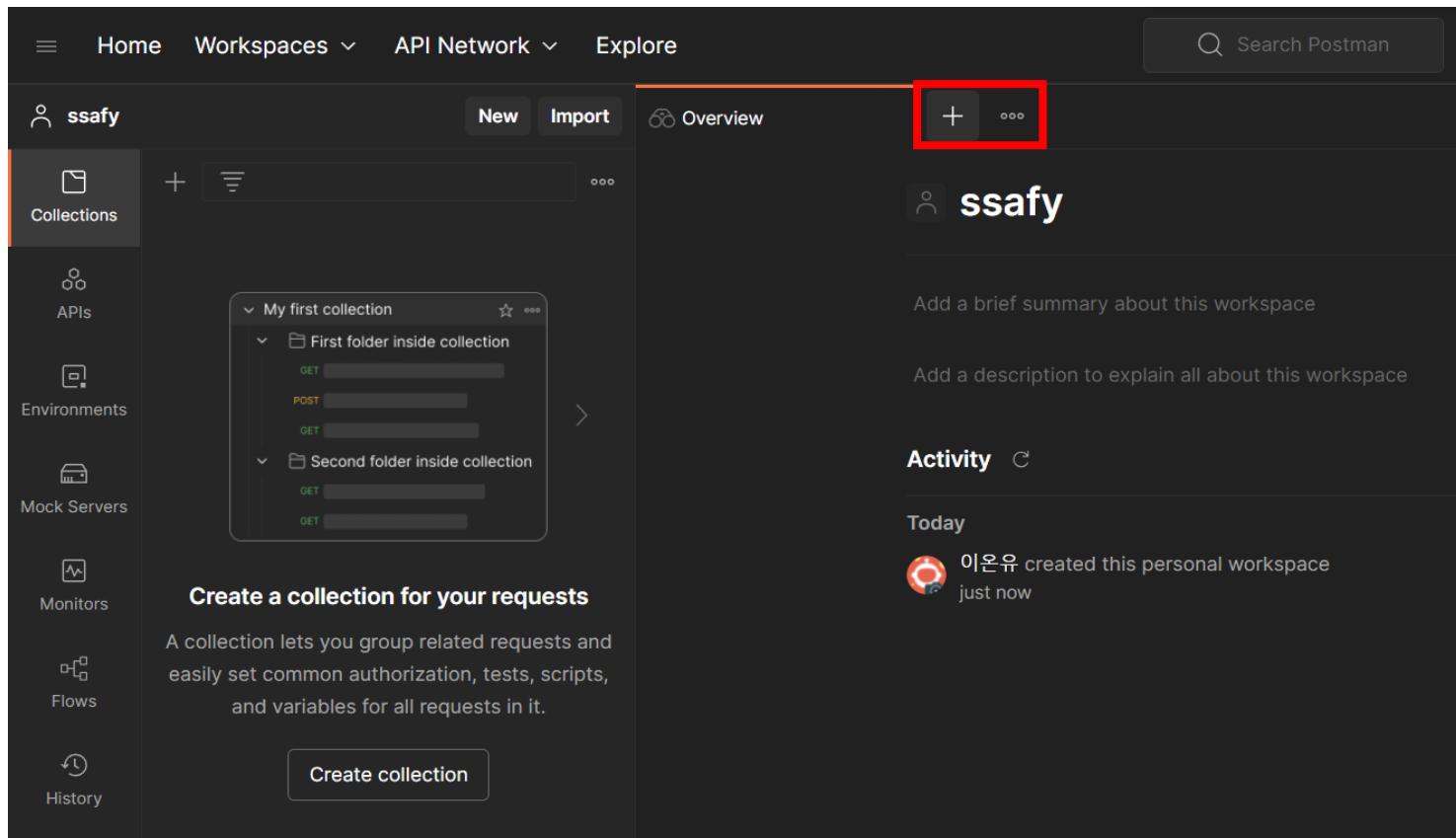
Workspace	Icon
My Workspace	User icon
test	gear and play button icon

Workspace 생성하기



+버튼 누르기

- + 버튼을 눌러 기초 작업 환경을 구성해준다.



POSTMAN 사용하기

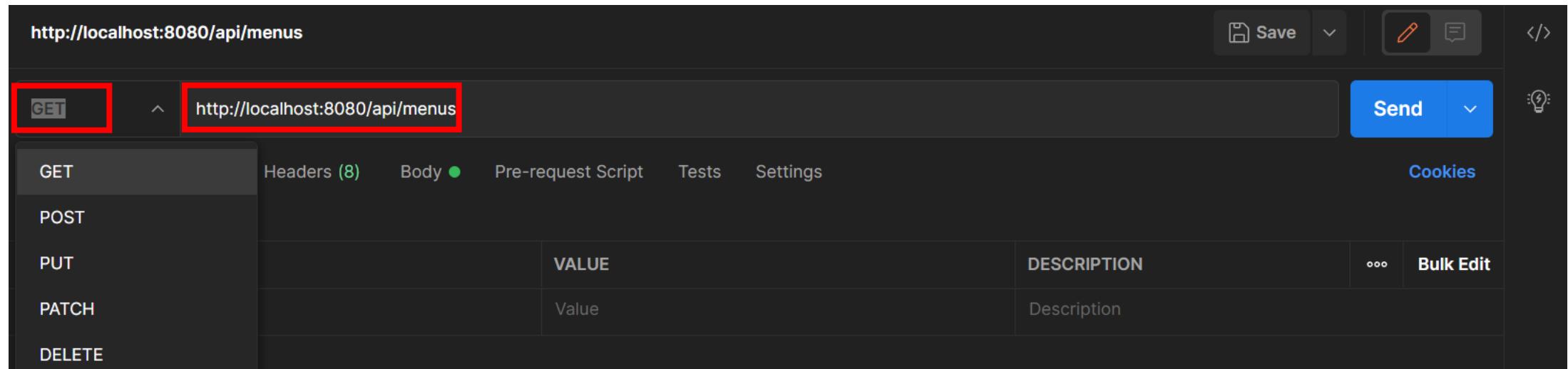
- **HTTP REQUEST METHOD 확인하기**

- GET 방식이면 서버에서 GET 방식으로 요청을 받게 되고
POST 방식이면 서버에서 POST 방식으로 요청을 받게 된다.
서로 **일치하는 요청인지를 확인!**

- **주소 입력**

- 접근할 서버의 API주소를 기입

- **Send 버튼을 누르면 해당 METHOD로 주소에 요청을 보낸다.**



send로 요청

- 아래 body에 요청 결과에 따른 값이 리턴
- GET, POST, PATCH, PUT, DELETE 모든 요청이 가능

GET http://localhost:8080/api/menus

Params Authorization Headers (8) Body Pre-request Script

Query Params

KEY	VALUE
Key	Value

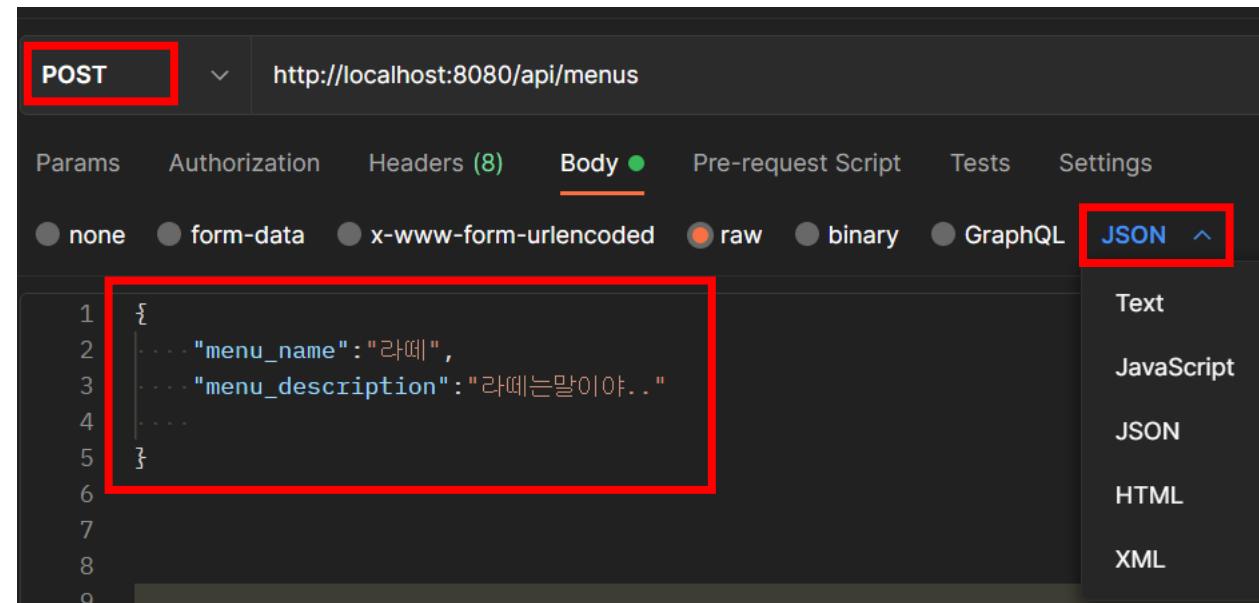
Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 [ ]  
2 {  
3   "menu_id": 1,  
4   "menu_name": "아이스 아메리카노",  
5   "menu_description": "여름엔 아야가 진리"  
6 },  
7 {  
8   "menu_id": 2,  
9   "menu_name": "캬페라떼",  
10  "menu_description": "Latte is horse"  
11 },  
12 {  
13   "menu_id": 3,  
14   "menu_name": "녹송아 아이스티",  
15   "menu_description": "내 입안 녹송아향 가득"  
16 },  
17 {  
18   "menu_id": 4,  
19   "menu_name": "딸기",  
20   "menu_description": "딸기는 딸기딸기해"  
21 }  
22 ]
```

POST 요청해보기

- POST로 HTTP REQUEST METHOD 변경
 - url도 서버와 일치하는지 확인
- Postman에서 요청하는법
 - body -> raw 클릭 후 형식을 JSON 형식으로 바꿔준다.
 - 해당 body의 JSON DATA들을 express 에서는 req.body() 형태로 받아온다.



POST 요청해보기

- 성공적으로 요청이 수행되었다.

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8080/api/menus`. The request body is set to `JSON` and contains the following data:

```
1 {
2   ...
3   "menu_name": "라떼",
4   ...
5 }
```

The response status is `200 OK`, and the response body is:

```
1 [
2   {
3     "fieldCount": 0,
4     "affectedRows": 1,
5     "insertId": 5,
6     "info": "",
7     "serverStatus": 2,
8     "warningStatus": 0
9   },
10  null
11 ]
```

POST 요청후 값 다시 조회하기

- GET으로 변경한 후 다시 요청을 보낸다.

```
GET http://localhost:8080/api/menus

Params Authorization Headers (8) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2   "menu_name": "라떼",
3   "menu_description": "라떼는말이야.."
4 }
5

Body Cookies Headers (8) Test Results
Pretty Raw Preview Visualize JSON ↻

1 [
2   {
3     "menu_id": 1,
4     "menu_name": "아이스 아메리카노",
5     "menu_description": "여름엔 아아가 진리"
6   },
7   {
8     "menu_id": 2,
9     "menu_name": "캬페라떼",
10    "menu_description": "Latte is horse"
11   },
12   {
13     "menu_id": 3,
14     "menu_name": "복숭아 아이스티",
15     "menu_description": "내 입안 복숭아향 가득"
16   },
17   {
18     "menu_id": 4,
19     "menu_name": "딸기",
20     "menu_description": "딸기는 딸기딸기해"
21   },
22   {
23     "menu_id": 5,
24     "menu_name": "라떼",
25     "menu_description": "라떼는말이야.."
26   }
27 ]
```

HTTP Response Status Code

http 로그를 상세히 보여주는 npm 패키지

morgan

npm v1.10.0 downloads 13.7M/month travis passing coverage 82%

HTTP request logger middleware for node.js

Named after **Dexter**, a show you should not watch until completion.

API

```
var morgan = require('morgan')
```

Install

```
> npm i morgan
```

Repository

❖ github.com/expressjs/morgan

Homepage

🔗 github.com/expressjs/morgan#readme

Weekly Downloads

2,688,665



npm i morgan

- app.use(morgan('dev')) 형태로 사용

```
{  
  "dependencies": {  
    "cors": "^2.8.5",  
    "express": "^4.18.1",  
    "morgan": "^1.10.0",  
    "mysql2": "^2.3.3"  
  }  
}
```

```
const morgan = require("morgan");  
app.use(morgan("dev"));
```

Postman으로 요청을 보내고 node의 log 확인

- postman으로 요청 보내기 -> 서버에서 수신 -> morgan이 로그를 기록
- 200 32.830 ms
 - 200
 - Http Response Status Code
 - 32.830 ms
 - 통신 성공까지 걸린 시간

```
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
this server listening on 8080
GET /api/menus 200 32.830 ms - 672
```

- 1xx(Information)

- 요청을 받았으며 프로세스를 진행하는 상태
- 100(요청 후 대기중)

- 2xx(Success)

- 요청을 성공적으로 받았으며 인식했고 진행한 상태
- 200(요청 성공)
- 201(생성 완료)

- 3xx(Redirection)

- 클라이언트는 요청을 마치기 위해 추가적으로 동작을 취해줘야 한다.
- 301(새위치로 영구적으로 이동)
- 302(임시이동)

- 4xx(Client Error)

- **클라이언트** 단에서 에러가 발생한 경우 해당 오류가 발생한다.
- 400(잘못된 요청)
- 401(권한 없음)
- 404(찾을 수 없음)

5xx(Server Error)

- **서버** 측에 에러가 발생 했을 경우 해당 상태코드가 나타난다.
- 500(내부 서버 오류)
- 503(서버를 사용 할 수 없음)
- 504(시간 초과)

정리

- 가장 많이 보게 되는 상태 코드
 - 200번대
 - 응답이 성공
 - 400번대
 - 클라이언트 이슈(보통 대부분의 에러가 본인에 의해서 나온다.)
 - 500번대
 - 서버 이슈

params, query, body

path

- **http://localhost:8080/api/menus/1**
 - menus 데이터의 첫번째를 가져온다.
 - ex) <http://localhost:8080/api/group/1/student>
 - group 1번의 모든 student
 - ex) <http://localhost:8080/api/group/1/student/2>
 - group 1번의 2번 student
 - 계층적 구조를 나타낼때 사용
 - express에서는 req.params()로 받아온다

The screenshot shows the Postman interface with a dark theme. At the top, there's a dropdown for 'Method' set to 'GET' and a URL input field containing 'http://localhost:8080/api/menus/1'. To the right of the URL is a 'Send' button. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. The 'Params' tab is currently active, showing a table titled 'Query Params'. The table has columns for 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. A single row is present with 'Key' in the KEY column and 'Value' in the VALUE column. The DESCRIPTION column contains the placeholder 'Description'.

app.get('/주소/:변수')

- app.get('/api/user/:id')
 - :id 부분은 변수로써 어떤 값이 들어올지 모른다
 - ex) /api/user/3
 - /api/user의 3번째
 - /api/user/chicken
 - /api/user의 chicken user

```
app.get("/api/user/:id", (req, res) => {  
  console.log(req.params);  
});
```



```
{ id: '2' }  
GET /api/user/2 200 17.147 ms - 15  
{ id: '1' }  
GET /api/user/1 200 2.484 ms - 15  
{ id: 'chicken' }  
GET /api/user/chicken 200 1.288 ms - 15
```

query

- **http://localhost:8080/api/menus?키값=value**
 - ?로 시작해서 key값 = value 형태
- **http://localhost:8080/api/menus?키값=value&키값1=value1**
 - 추가로 전달하고 싶은 경우 &를 붙여서 전달한다.
- **http에서는 query params, query string 이라고 한다.**
 - express 에서는 req.query() 형태로 받아온다.
 - POSTMAN에서 Params라고 표시하는 이유는 Query Params 중 Query 부분을 줄여서 표시한것

The screenshot shows the Postman application interface. At the top, it displays a 'GET' method and the URL `http://localhost:8080/api/menus?key=value&key1=value1`. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is selected, showing a table with two rows of query parameters. The first row has 'key' in the 'KEY' column and 'value' in the 'VALUE' column. The second row has 'key1' in the 'KEY' column and 'value1' in the 'VALUE' column. There is also a third row with 'Key' in the 'KEY' column and 'Value' in the 'VALUE' column, which appears to be a placeholder or a header entry.

KEY	VALUE
key	value
key1	value1
Key	Value

주소?키값=value

- 추가로 전달하고 싶은 경우 &를 붙여서 전달한다.

GET http://localhost:8080/api/user/chicken?test=value

Params

KEY	VALUE
test	value



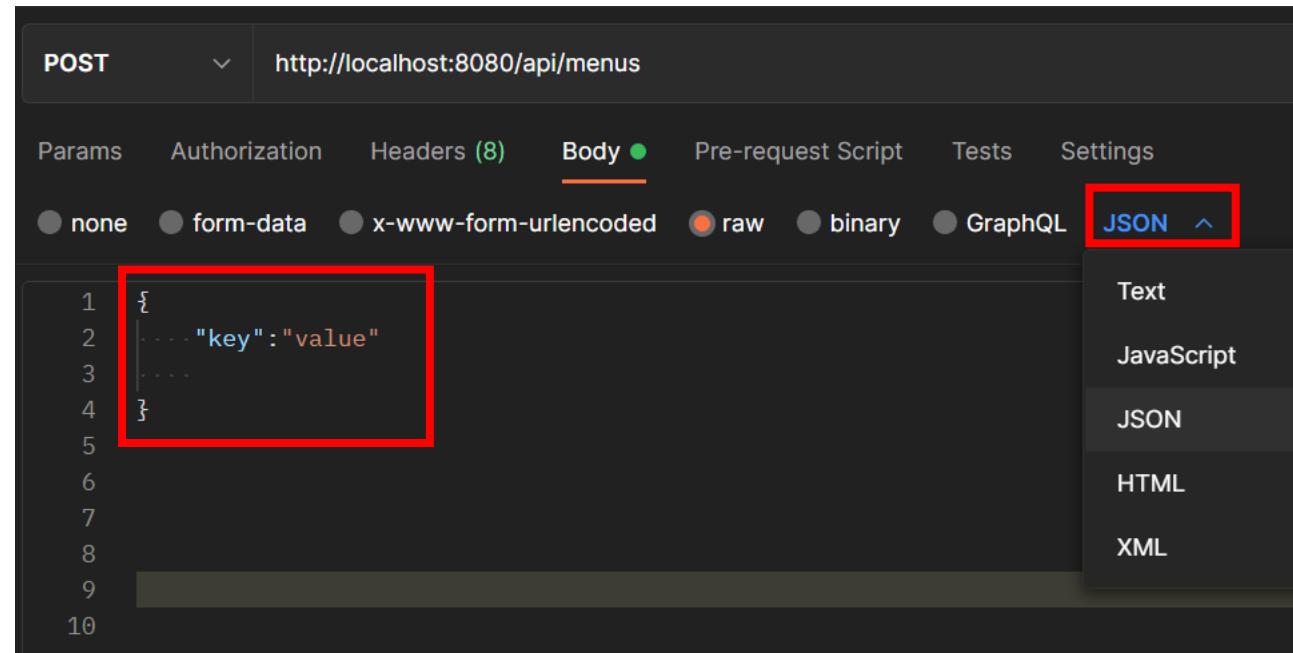
```
app.get("/api/user/:id", (req, res) => {  
  console.log(req.query);  
})
```



```
{ id: 'chicken' }  
GET /api/user/chicken?test=value 200 1.142 ms - 15
```

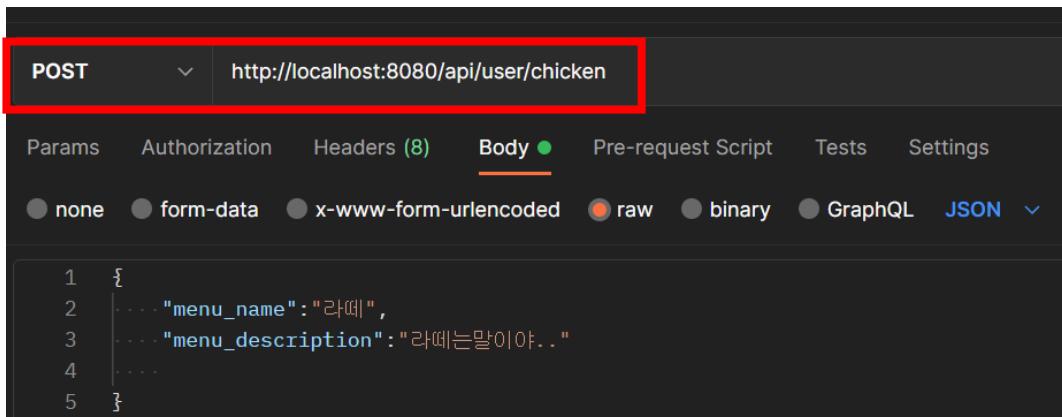
body

- POST, PATCH 요청에 사용
- Postman에서 요청하는법
 - body -> raw 클릭 후 형식을 JSON 형식으로 바꿔준다.
- express 에서는 req.body() 형태로 받아온다.



body 안에 json 형식으로 데이터를 보낸다

- body 데이터는 POST, PATCH에서 사용한다.
- req.body 안에 데이터가 담긴다



```
app.post("/api/user/:id", (req, res) => {  
  console.log(req.body);  
});
```



```
{ menu_name: '라떼', menu_description: '라떼는말이야...' }  
POST /api/user 200 2.140 ms - 16
```

body 데이터를 가질수 있는 경우

- POST, PUT, PATCH
- GET 과 DELETE는 일반적으로 body 데이터를 가질수 없다.

Request method	Request has payload body
GET	Optional
POST	Yes
PUT	Yes
DELETE	Optional
PATCH	Yes

표를 보고 API 만들어 보기

- 넘겨받은 params, query, body는 모두 console.log로 출력한다.
- json 방식으로 리턴 한다.
- 기술된 params, query, body가 정상적으로 넘어오지 않았을 경우 실패 시(json)를 반환한다
- 순서대로, 차근차근 같이 해보자.**

	GET	POST	PATCH	DELETE
라우터	/user	/user	/user	/user
params	:id		:id	:id
query	name=chicken			
body		id : num, password: num1	name : hello	
리턴 값(json)	{ getid: true, id : id, name : chicken }	{ signup: true, id : num, password: num1 }	{ update: true, id: id, name: hello }	{ delete: true, id: id }
실패 시(json)	{ getid: false }	{ signup: false }	{ update: false }	{ delete : false }

GET /user/123?name=chicken

- GET - HTTP 요청 메서드
- /user - 라우터 주소
- 123 - params
- ?name=chicken - query 파라미터

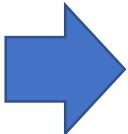
GET	
라우터	/user
params	:id
query	name=chicken
body	
리턴 값(json)	{ getid: true, id : id, name : chicken }
실패 시(json)	{ getid: false }

GET /user/123?name=chicken

코드 작성하기

- 명세서를 보고 실제로 코드로 변경

GET	
라우터	/user
params	:id
query	name=chicken
body	
리턴 값(json)	{ getid: true, id : id, name : chicken }
실패 시(json)	{ getid: false }



완성된 전체 코드부터
index.js에 작성해보자.

```

15 // GET /user/123?name=chicken
16 app.get("/user/:id", (req, res) => {
17   console.log(req.params); // { id: '123' }
18   console.log(req.query); // { name: 'chicken' }
19   console.log(req.body); // {}
20   try {
21     if (req.params.id && req.query.name) {
22       return res.json({
23         getid: true,
24         id: req.params.id,
25         name: req.query.name,
26       });
27     }
28     return res.json({ getid: false });
29   } catch (error) {
30     return res.json({ getid: false });
31   }
32 });

```

유의 사항

- params
 - 콜론 (:) 쓰고 parameter 이름
 - 실제 사용할때는 url에 직접 입력
 - ex) /user/:id 라고 가정
 - /user/123
 - id는 123
 - /user/jonylee
 - id는 jonylee
 - params, query, body는 전부 req 객체로 부터 존재하는 프로퍼티

```
15 // GET /user/123?name=chicken
16 app.get("/user/:id", (req, res) => {
17   console.log(req.params); // { id: '123' }
18   console.log(req.query); // { name: 'chicken' }
19   console.log(req.body); // {}
```

결과 확인

- params
 - /:id, 즉, 123
- query
 - ?name=chicken
- body
 - 담겨있지 않다.

GET localhost:8080/user/123?name=chicken



```
this server listening on 8080
{ id: '123' }
{ name: 'chicken' }
{}
GET /user/123?name=chicken 200 4.548 ms - 42
```

추가 코드 작성하기

- params의 id와 query의 name이 존재하는 경우에만 정확한 json을 리턴
- 해당 조건을 만족하지 않거나 에러가 발생하는 경우(통신 실패) getid:false를 리턴

```
20  try {
21    if (req.params.id && req.query.name) {
22      return res.json({
23        getid: true,
24        id: req.params.id,
25        name: req.query.name,
26      });
27    }
28    return res.json({ getid: false });
29  } catch (error) {
30    return res.json({ getid: false });
31 }
```

결과 확인

- 요청에 따른 결과가 잘 나오는 것을 확인 할 수 있다.
- 이렇게 구축한 서버는 멋진 REST API 서버가 된다.

```
1
2      "getid": true,
3      "id": "123",
4      "name": "chicken"
5
```

나머지 명세서를 보고 API를 만들어보자

- GET 요청을 했던 내용을 분석을 기반으로
- POST, PATCH, DELETE 명세를 보고 API 서버를 작성

	GET	POST	PATCH	DELETE
라우터	/user	/user	/user	/user
params	:id		:id	:id
query	name=chicken			
body		id : num, password: num1	name : hello	
리턴 값(json)	{ getid: true, id : id, name : chicken }	{ signup: true, id : num, password: num1 }	{ update: true, id: id, name: hello }	{ delete: true, id: id }
실패 시(json)	{ getid: false }	{ signup: false }	{ update: false }	{ delete : false }

정적 파일

- 직접 변화시키지 않는 이상 변하지 않는 파일
 - ex) image, file
- 해당 정적 파일들을 보여줄 서버 구축이 필요

정적 파일 서비스하기

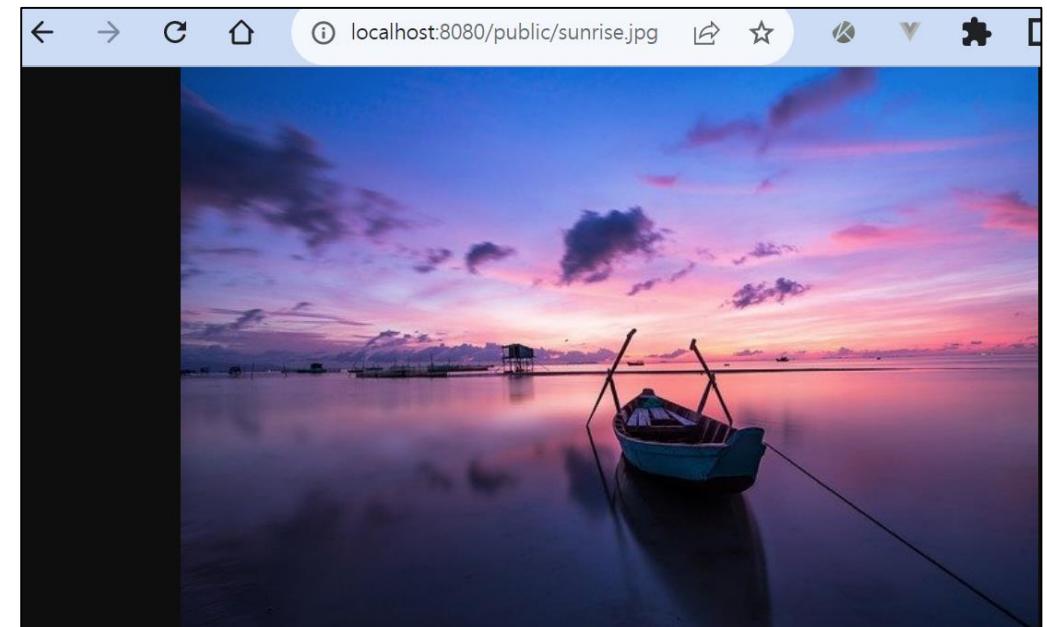
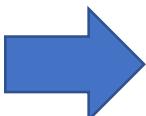
- public 폴더 생성
- `app.use('/경로', express.static('public'))`
 - public 폴더를 정적 폴더로 생성
 - `app.use('/public', express.static('public'))`
 - public 라우터에 접근시 public 안에 있는 폴더 내용들을 보여줄수 있다.
 - `http://localhost:8080/public/sunrise.jpg`

```
> public
JS index.js
npm package-lock.json
npm package.json

const express = require("express");

const app = express();
const PORT = 8080;

app.use("/public", express.static("public"));
```

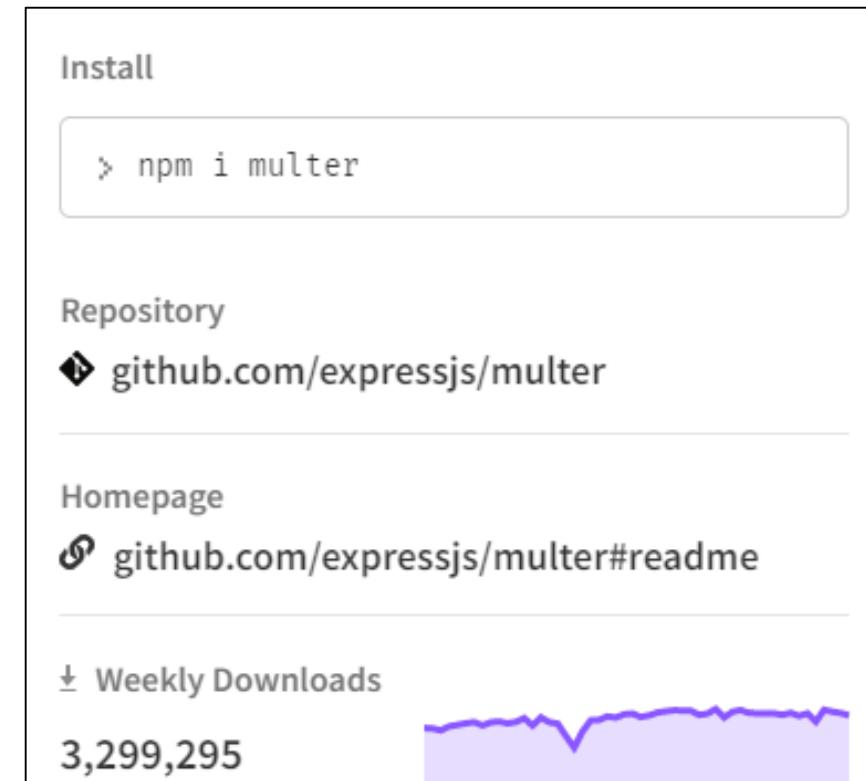


이미지 업로드하기

- multer 라이브러리 활용
- 서버
 - multer 라이브러리를 통해 요청한 파일 업로드
 - 정적 폴더에 업로드 해서 정적 파일 서비스가 진행되도록 하기
 - POSTMAN 을 활용해서 테스트
- 클라이언트
 - axios를 활용해 파일 전송

multer 라이브러리 활용

- 이미지 업로드를 손쉽게 진행하게 해주는 라이브러리
- npm i multer
- 클라이언트로부터 요청 받은 파일을 서버 폴더에 업로드



multer 정의하기

- **storage**
 - 저장 관련 정보
 - destination
 - 파일이 저장될 경로 지정
 - filename
 - 파일이 저장될 이름 지정
- **limits**
 - 파일 업로드 사이즈 제한

```
const multer = require("multer");
const upload = multer({
  storage: multer.diskStorage({
    destination: (req, file, done) => {
      done(null, "public/");
    },
    filename: (req, file, done) => {
      done(null, file.originalname);
    },
  }),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```

정의된 upload 객체를 요청 앞에 집어넣는다.

- (req,res) 전에 요청을 새로 집어넣게 되면 해당 부분을 먼저 진행하고 나서 그다음으로 넘겨준다.
- `upload.single('file')`
 - file이라는 이름으로 요청이 들어오게 되면 해당 file을 multer 정의에 따라 업로드한다.
- file upload 는 반드시 **POST** 요청만 가능하다.

```
app.post('/api/file', upload.single('file'), (req, res) => {
  return res.json({test:"OK"})
})
```

POSTMAN에서 업로드하기

- **body**

- form-data 활용
- 기존 Text -> File로 클릭하면 File 또한 전송이 가능
- file이라는 key값으로 selectFiles를 담아서 보낸다.

The screenshot shows the POSTMAN interface for sending a POST request to `http://localhost:8080/api/file`. The 'Body' tab is selected, and the 'form-data' option is chosen. A table is displayed with one row. The first column contains a checkbox which is checked, the second column has the key 'file', and the third column has a 'File' button with a dropdown menu open, showing 'Select Files'. The entire 'File' button area is highlighted with a red box.

	KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/>	file	File <small>▼</small> Select Files	...	<small>...</small>

form-data를 활용해 전송

- 전송후 public 폴더에 정상적으로 profile.jpg가 업로드 된것을 확인

POST http://localhost:8080/api/file

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

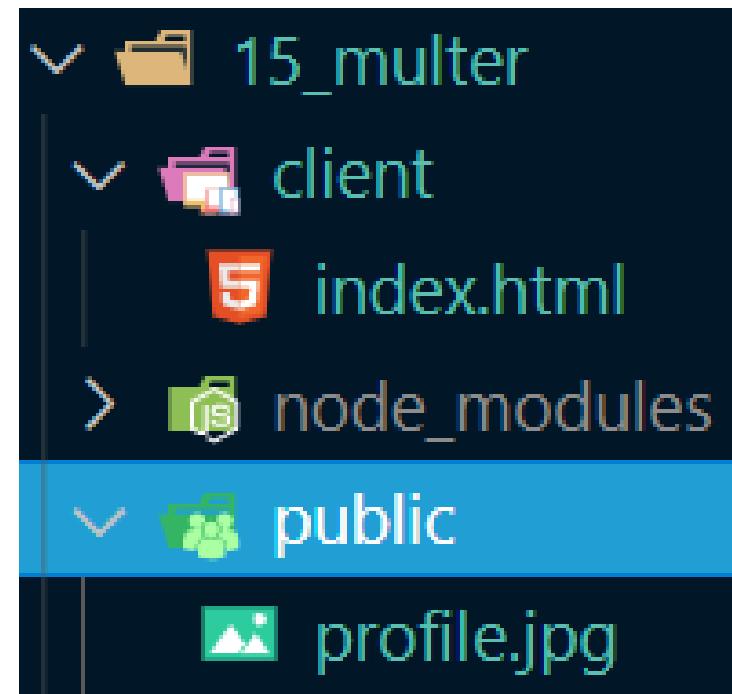
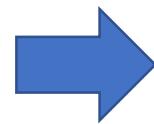
form-data

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> file	profile.jpg			

Body 200 OK 17 ms 279 B Save Response

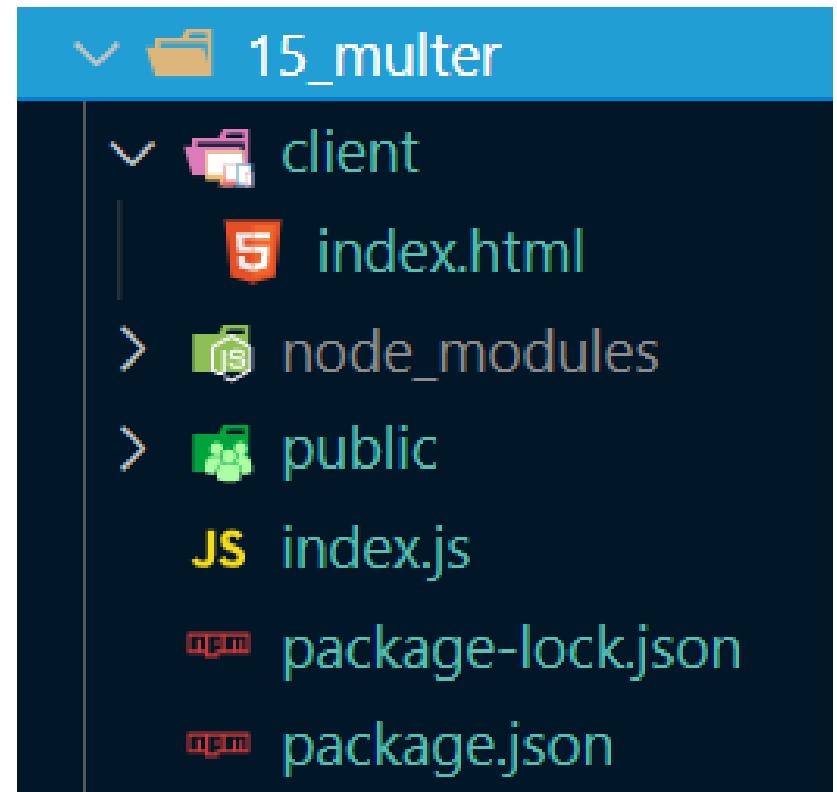
Pretty Raw Preview Visualize JSON

```
1 {  
2   "test": "OK"  
3 }
```



client 폴더 생성

- index.html 만들기
- axios를 활용해 업로드 진행하기



axios를 활용한 파일 업로드시 유의사항

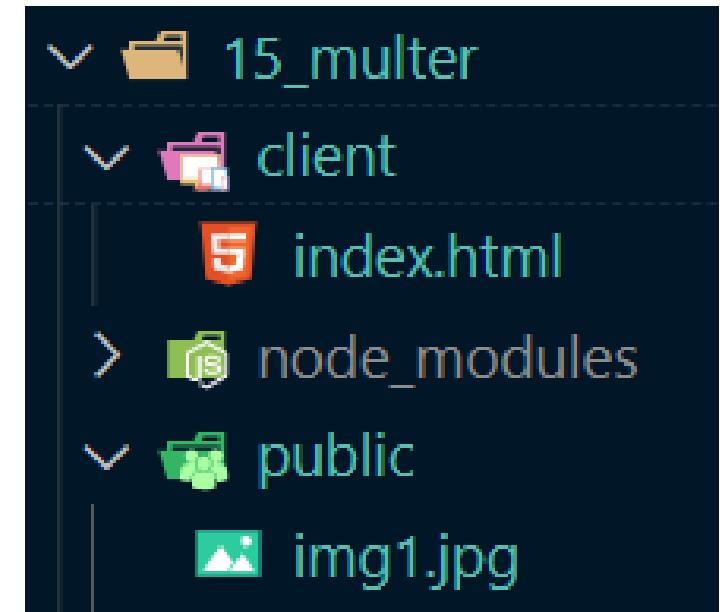
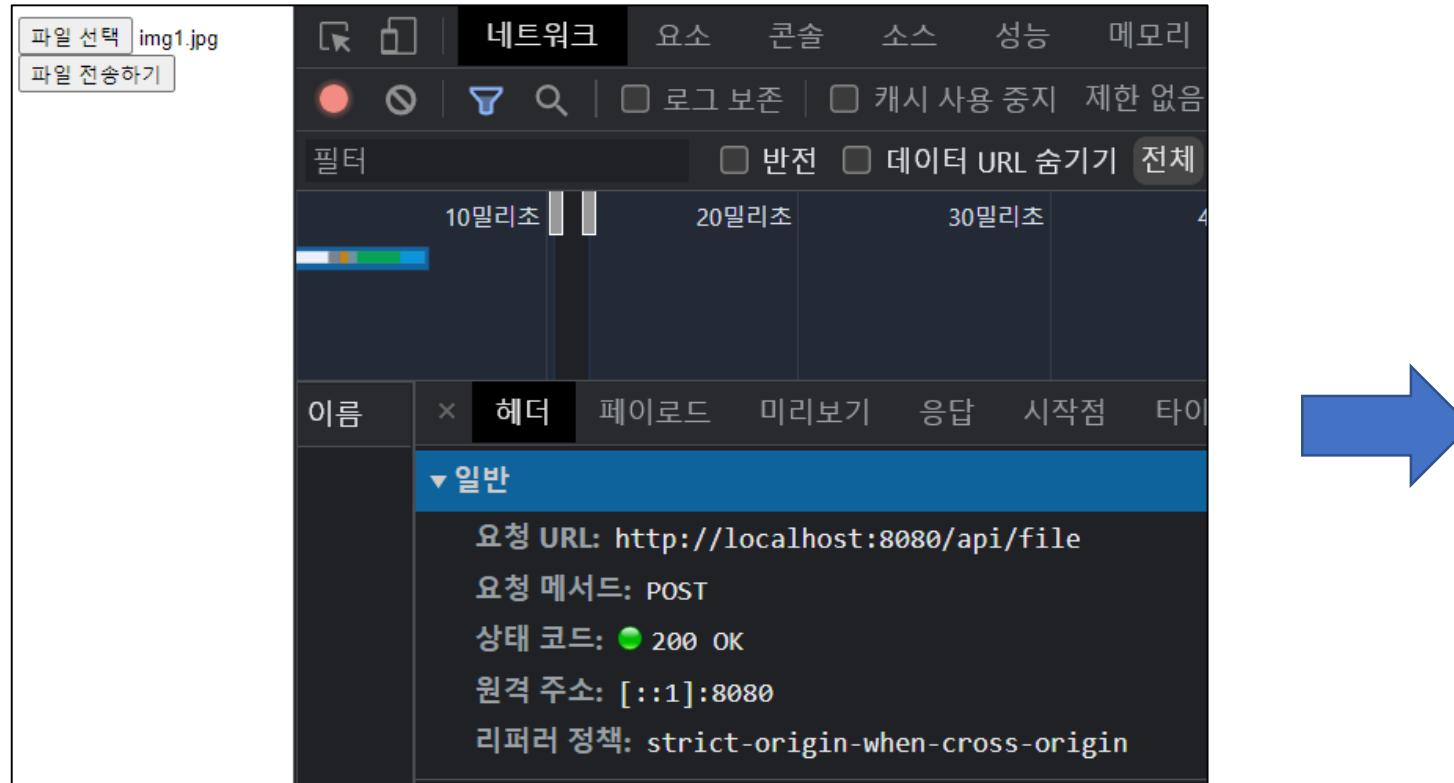
- 파일업로드는 반드시 POST 요청만 가능
- formData
 - form 객체를 만들어서 그안에 file의 정보를 넣어서 file:전달할 파일 식으로 작성
- axios에 header를 추가한다
 - multipart/form-data
 - 파일 업로드를 위한 속성

```
<input id="input-file" type="file">
<button type="button" id="submit-button">파일 전송하기</button>

<script src="https://cdn.jsdelivr.net/npm/axios@0.27.2/dist/axios.min.js"></script>
<script>
    const inputFile = document.querySelector('#input-file');
    const submitButton = document.querySelector('#submit-button');
    submitButton.addEventListener('click', function () {
        console.log(inputFile.files);
        const formData = new FormData();
        formData.append('file', inputFile.files[0])
    })
    axios.post("http://localhost:8080/api/file", formData, {
        headers: {
            "Content-Type": "multipart/form-data",
        }
    })
</script>
```

결과 테스트

- 파일 전송하기 클릭시 서버에 잘 업로드 된것을 확인



실제로 운영하는방식

- 클라이언트에서 서버에 파일 업로드
- 서버에서는 해당 파일을 업로드하고 해당 파일의 경로를 DB에 저장

파일업로드

- 서버 만들기
- 클라이언트 만들기
- 서버 < - > 클라이언트간 통신하게 만들기
- <https://pixabay.com/ko/> 에 접속해서 마음에 드는 이미지를 업로드하기

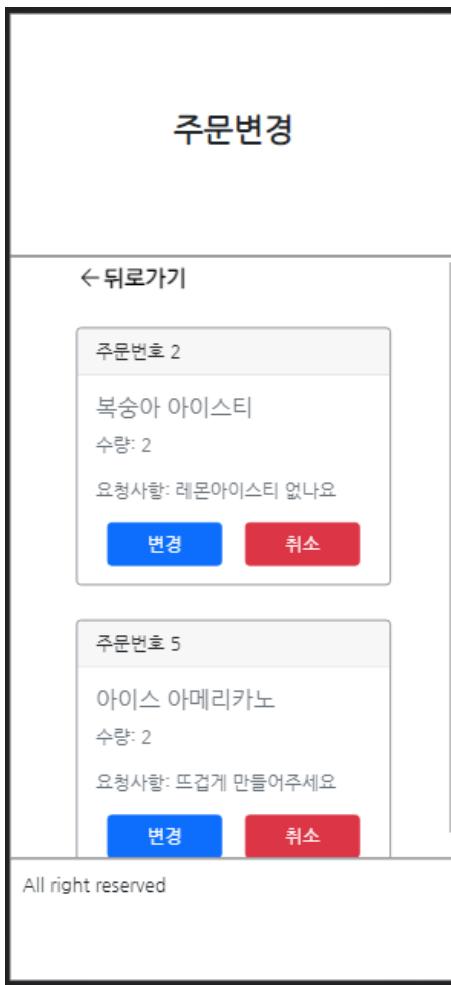
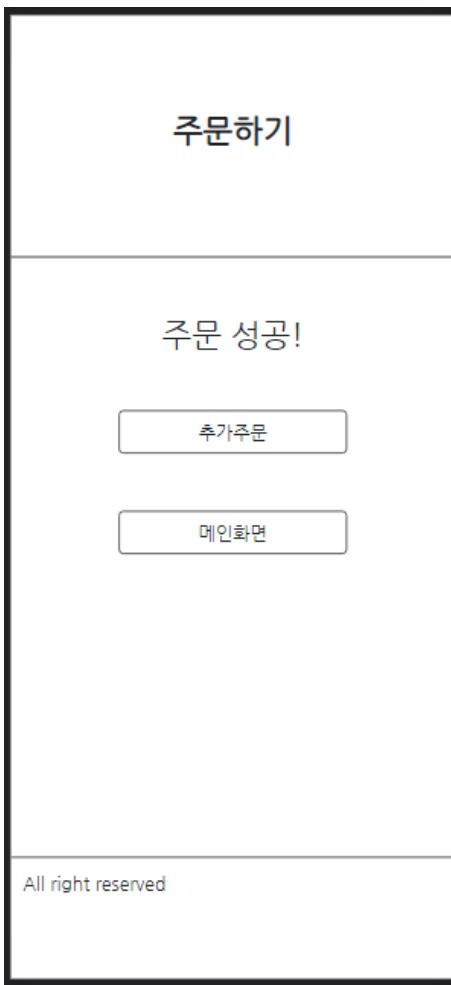
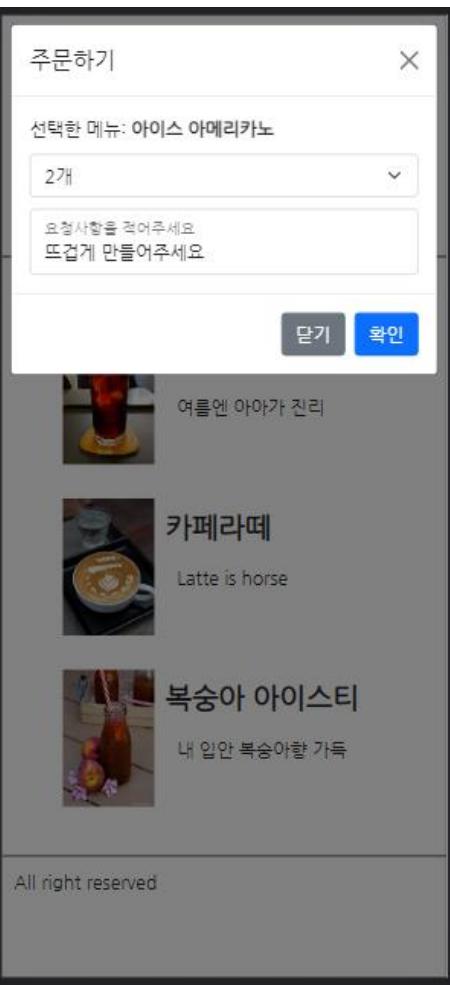
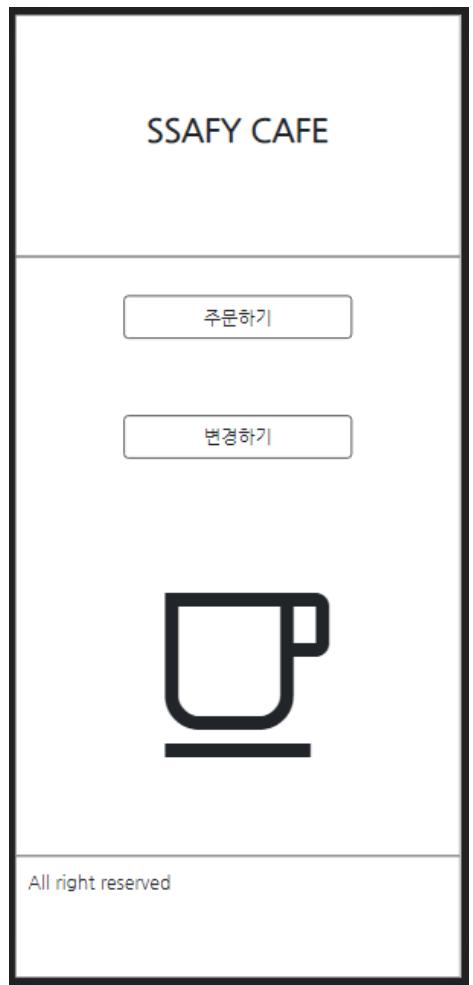
11장. REST API + MySQL

챕터의 포인트

- 카페 주문관리 API
- API 서버 배포

카페 주문관리 API

여러분은 다음과 같은 UI 를 가진, 카페 직원용 주문관리 시스템의 백엔드 파트를 담당하게 되었다.



요구 기능 분석

- **메뉴**

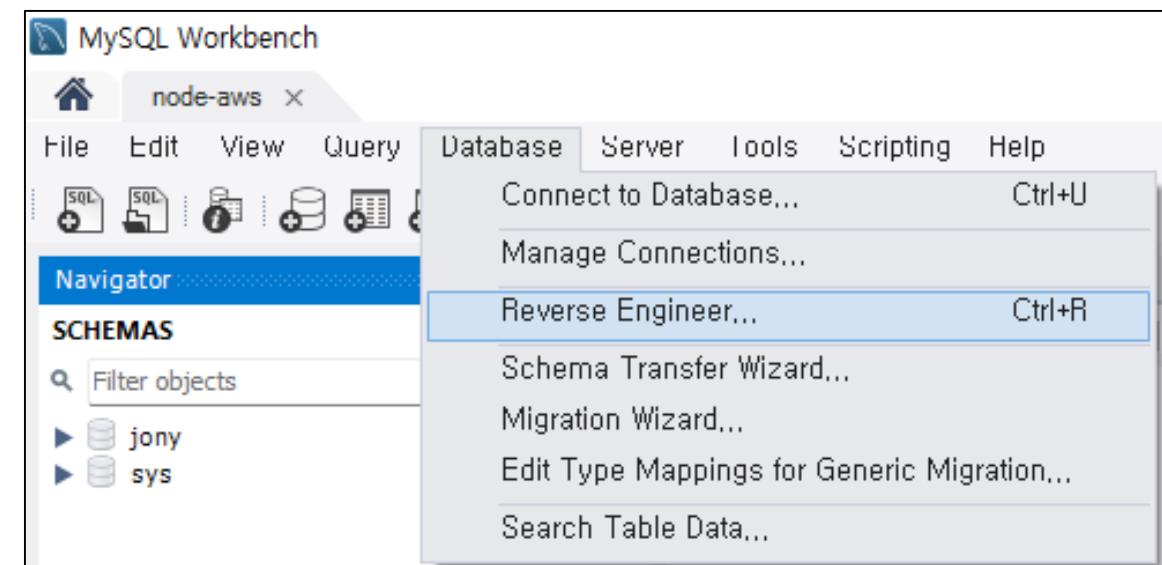
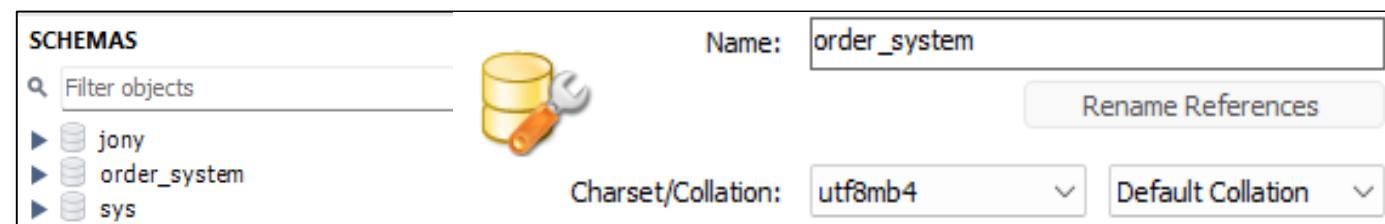
- 메뉴의 이름
- 메뉴의 정보
- 메뉴의 이미지 (파일 경로)

- **주문**

- 주문할 메뉴
- 주문할 메뉴의 갯수
- 요청사항

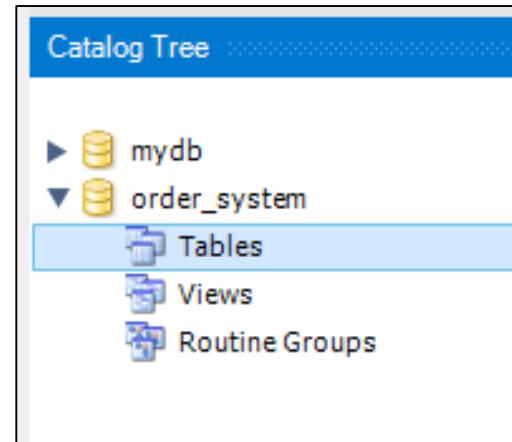
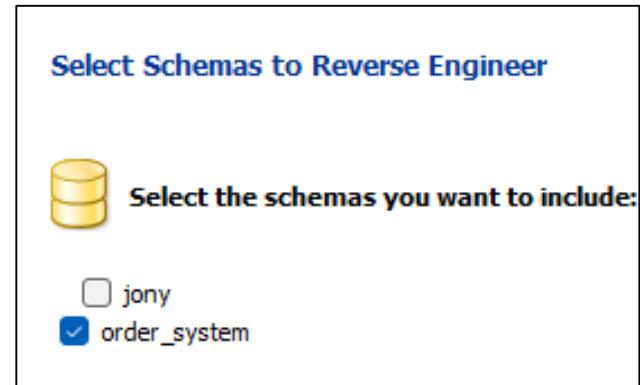
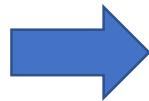
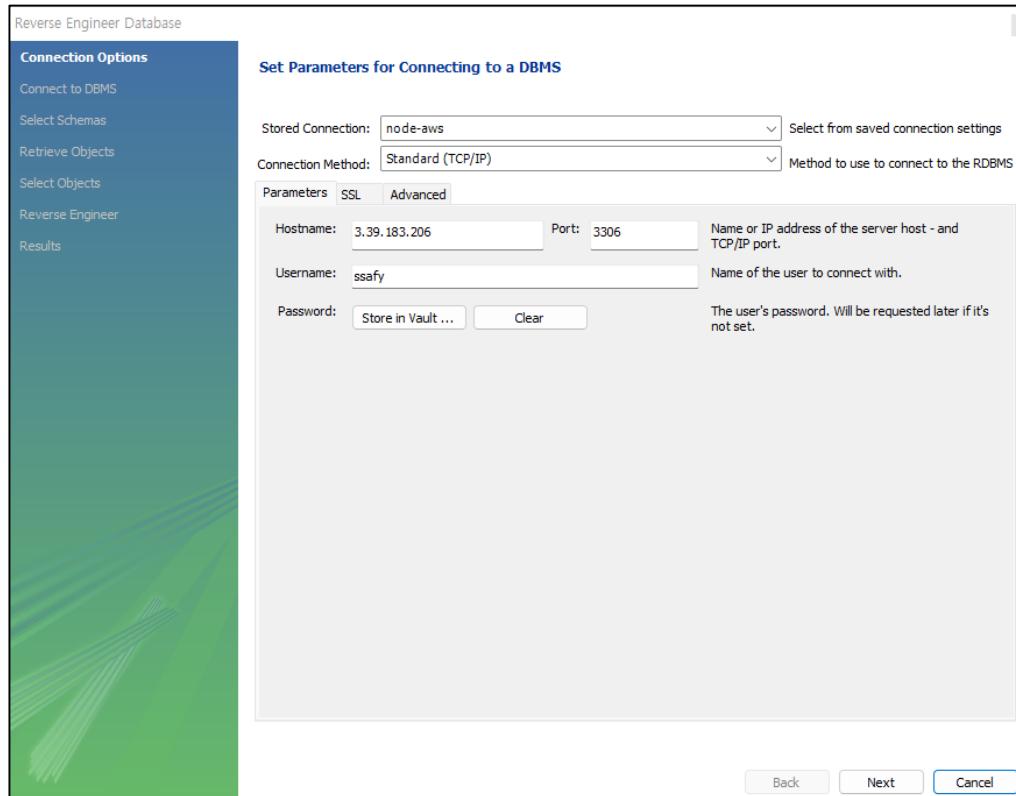
workbench 접속

- mysql workbench 접속
- order_system 스키마 작성
 - utf8mb4 로 작성
- 메뉴 Database -> Reverse Engineer 접속



Reverse Engineer Database

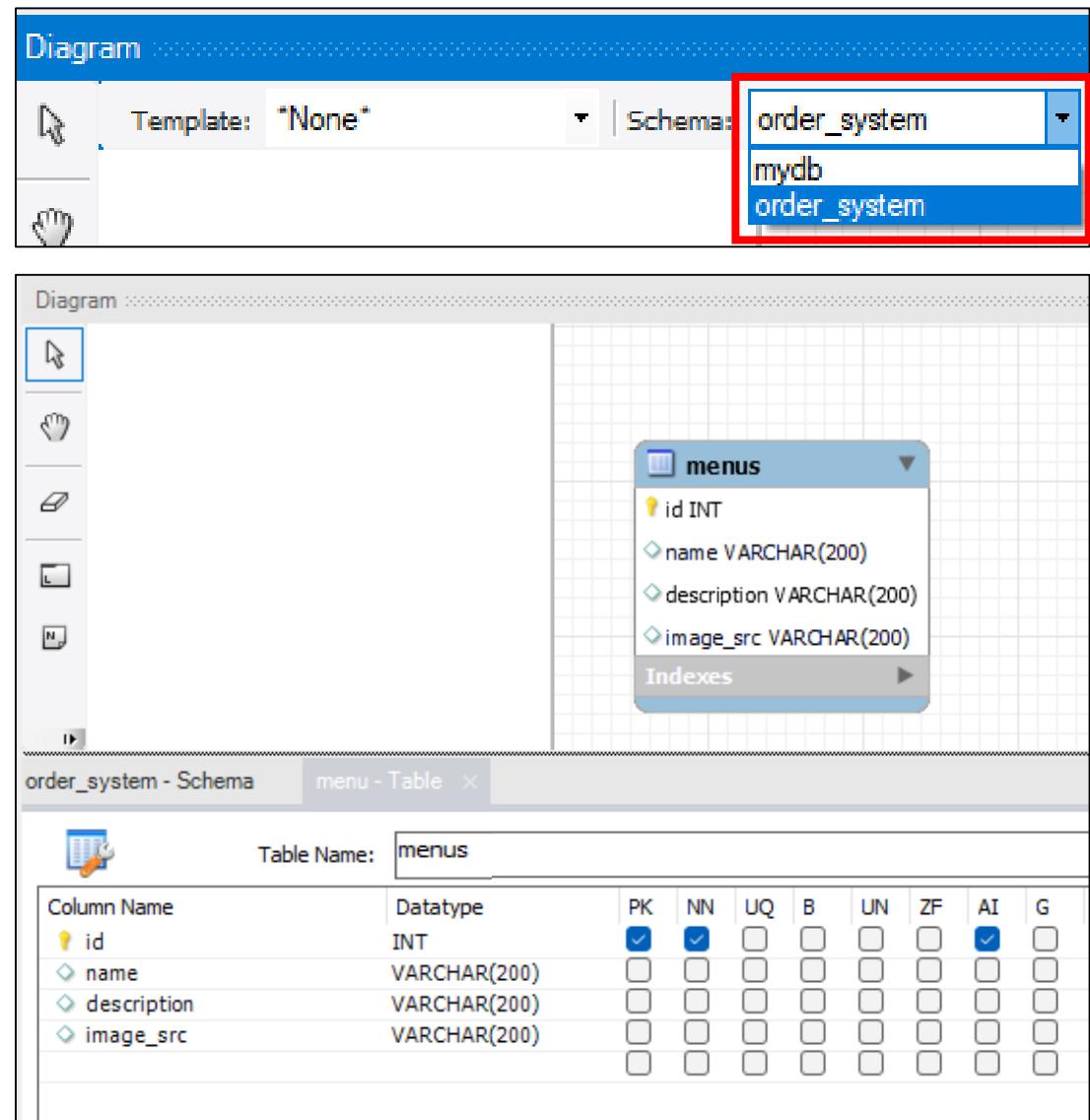
- 다이어그램 형식을 활용해서 테이블 정의 및 관계형 설정이 가능
- 작성해둔 AWS DB서버에 접근
- 새로만든 `order_system` 스키마를 포함시키기



테이블 설계하기-메뉴

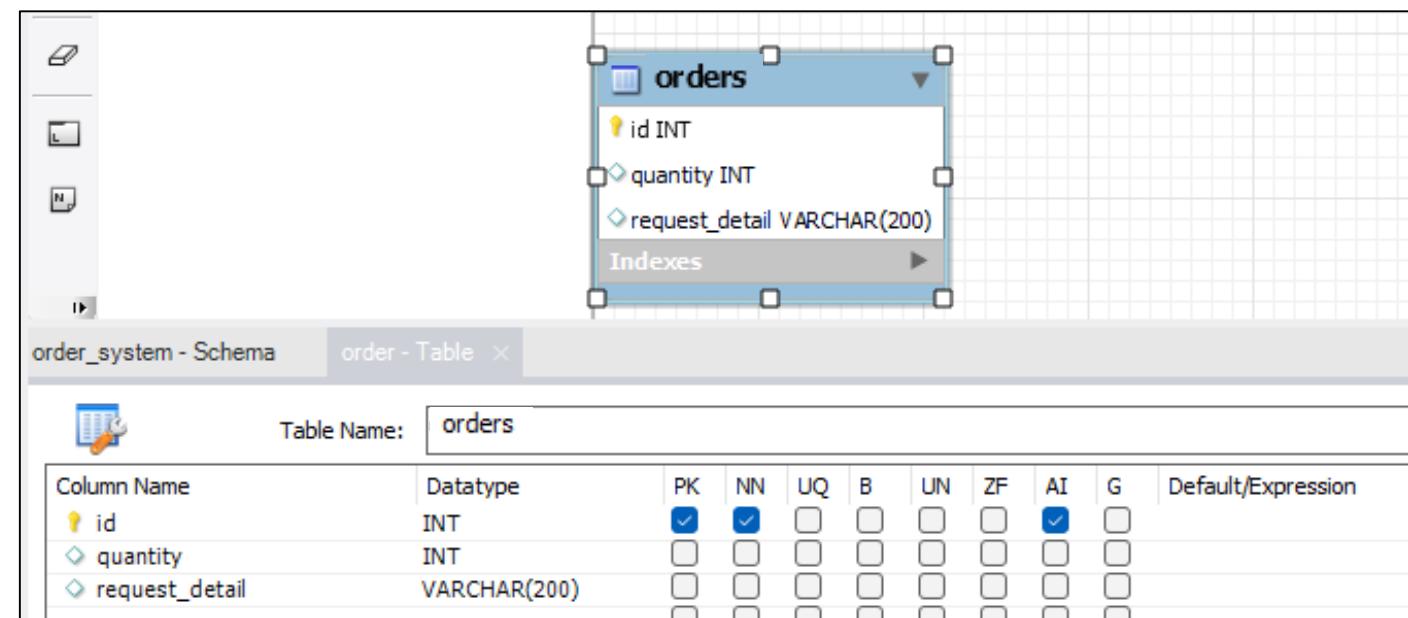
• 메뉴 설계

- 테이블 클릭시 **스키마 선택**에 유의
 - id
 - 고유 ID (중복 방지)
 - PK, NN, AI
 - name
 - 메뉴의 이름
 - VARCHAR(200)
 - description
 - VARCHAR(200)
 - image_src
 - VARCHAR(200)



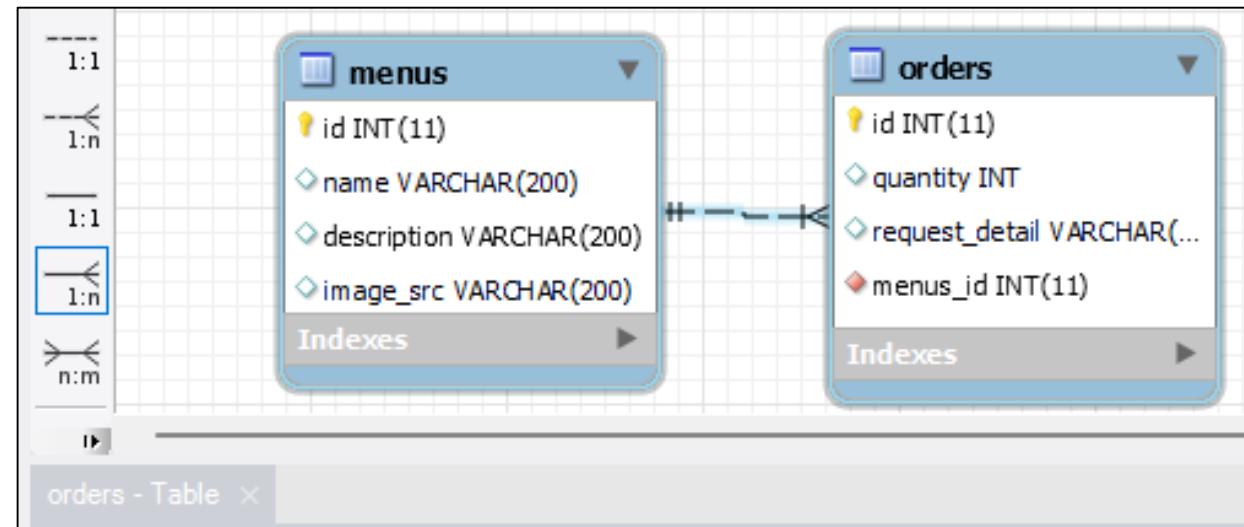
테이블 설계하기- 주문

- **orders**
 - id
 - 고유 ID (중복 방지)
 - PK, NN, AI
 - quantity
 - 주문할 메뉴의 수
 - INT
 - request_detail
 - 요청사항
 - VARCHAR(200)



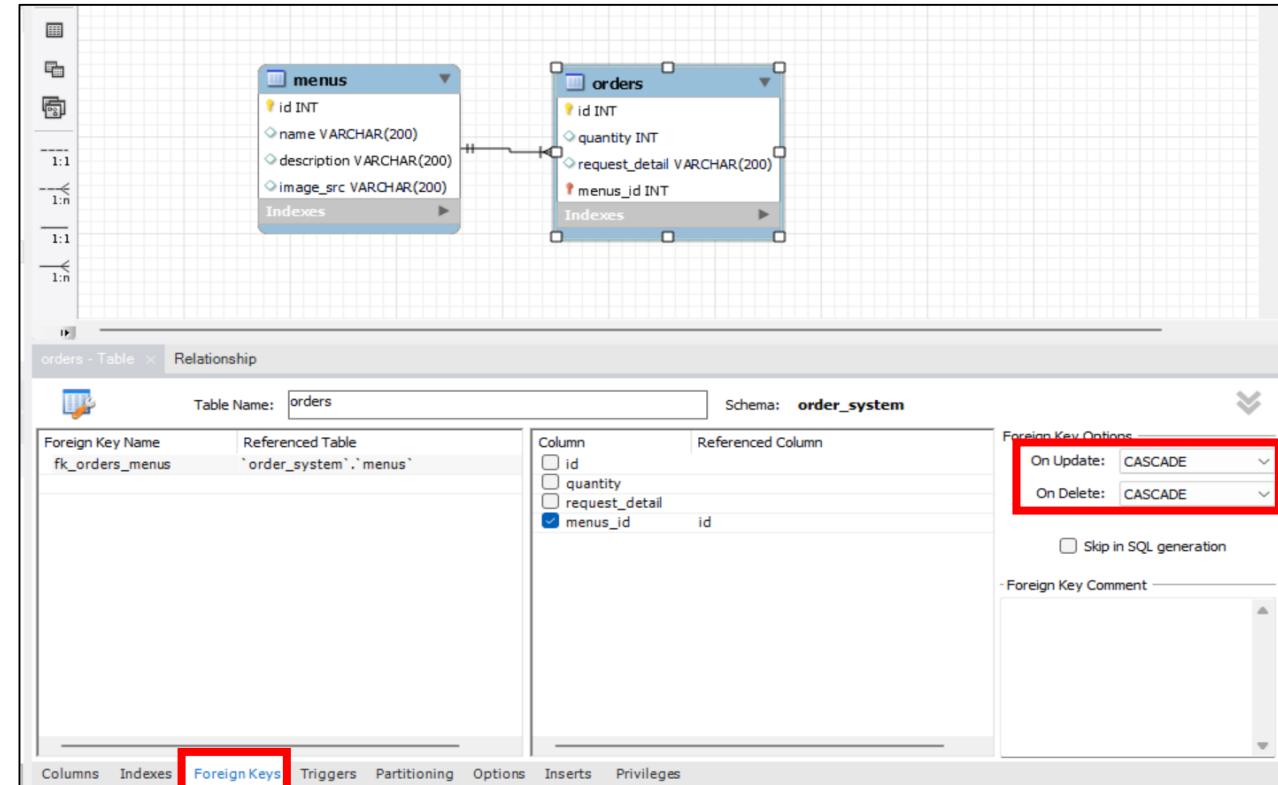
관계형 테이블 매핑

- orders에서는 menus 테이블의 정보가 필요
 - 메뉴의 정보는 menus 테이블에 존재한다.
 - id는 고유값이기 때문에 menus의 id를 통해 끌어오는 작업이 필요
 - order 테이블에서 menus_id 는 외래키가 된다.
 - 1:N 클릭 -> orders 클릭 후 menus 클릭
 - 주문할때 orders는 여러 개의 menus를 가질수 있다.



CASCADE 옵션 부여

- 외래키에 삭제/업데이트시 CASCADE 옵션부여
- orders 에는 menus_id가 존재한다.
 - 관련된 menus 삭제시에 외래키 조건에 의해 에러가 발생
 - CASCADE 옵션을 부여하면 menus 삭제시 menus 뿐만 아니라 해당 menus_id가 들어있는 모든 데이터를 삭제한다.



```
Error: Cannot delete or update a parent row: a foreign key constraint fails (`order_system`.`orders`, CONSTRAINT `fk_orders_menus` FOREIGN KEY (`menus_id`) REFERENCES `menus` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION)
```

CASCADE 미 부여상태로 관계된 테이블 데이터를 지울때 나오는 에러

• 1:1 관계

- 각 관계는 반드시 하나로만 매핑되어야 한다.
- ex) 남자 - 결혼 - 여자



• 1:N 관계

- 한쪽 개체는 다른 관계를 맺은쪽의 여러 개체를 가질 수 있다.
- ex) 유저(users)는 게시글(posts)을 여러 개 작성 할 수 있다.
- 부모는 여러명의 자식을 가질수 있다.



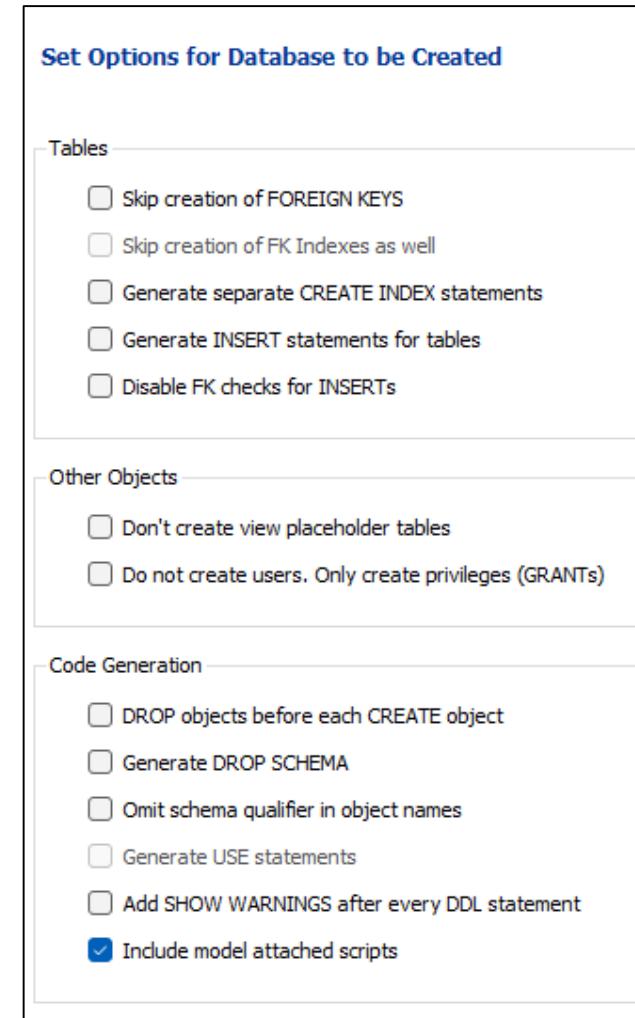
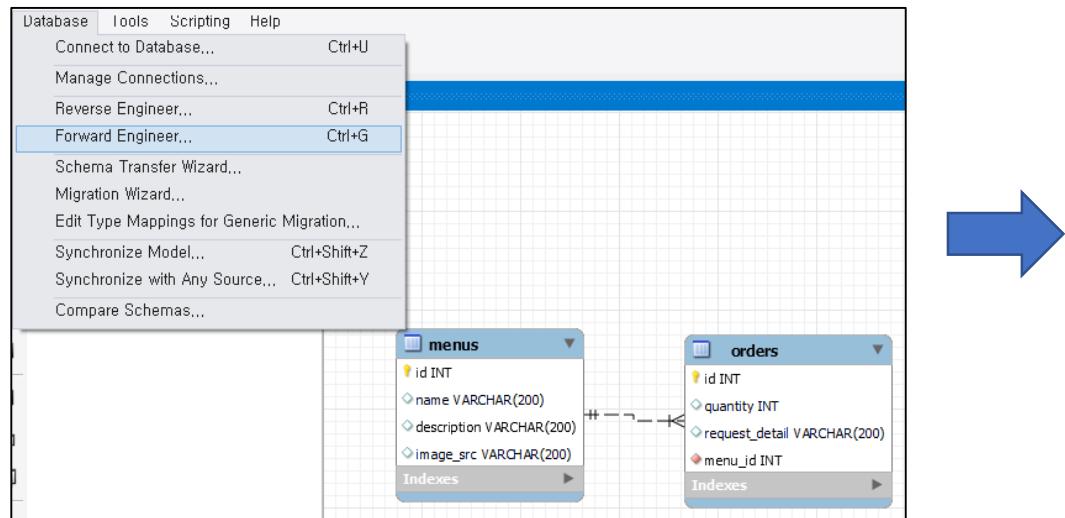
• N:M 관계

- 관계를 가진 양쪽에서 서로 1:N관계를 가지고 있는것
- 학원은 여러명의 수강생을 거느릴수 있다.
- 수강생도 여러 학원을 다닐 수 있다.



만든 다이어그램을 기반으로 테이블 생성하기

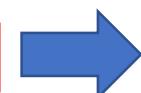
- 왼쪽 상단 Database → Forward Engineer
 - 작성한 테이블을 기반으로 테이블을 생성



forward engineer시 에러 발생 대처법

- window에 있는 workbench(8.x)의 mysql 버전과 aws의 mysql(5.x~) 버전이 다르게 되면 에러가 발생하기도 한다.
- INDEX 'fk~' VISIBLE 부분 제거후 실행

```
CREATE TABLE IF NOT EXISTS `order_system`.`orders` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `quantity` INT NULL,
  `request_detail` VARCHAR(200) NULL,
  `menus_id` INT NOT NULL,
  PRIMARY KEY (`id`, `menus_id`),
  INDEX `fk_orders_menus_idx` (`menus_id` ASC) VISIBLE,
  CONSTRAINT `fk_orders_menus`
    FOREIGN KEY (`menus_id`)
    REFERENCES `order_system`.`menus` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



```
CREATE TABLE IF NOT EXISTS `order_system`.`orders` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `quantity` INT NULL,
  `request_detail` VARCHAR(200) NULL,
  `menus_id` INT NOT NULL,
  PRIMARY KEY (`id`, `menus_id`),
  CONSTRAINT `fk_orders_menus`
    FOREIGN KEY (`menus_id`)
    REFERENCES `order_system`.`menus` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

왜 테이블을 둘로 나눴을까?

- 다음 상황을 가정해보자.
- 만약 100,000 개의 데이터를 저장한다면, 중복이 지나치게 많아 매우 비효율적
 - DB 용량 증가, 속도 저하
 - 비용 증가: Oracle 등의 유료 데이터베이스를 사용한다면, 훨씬 많은 비용 지불
 - 만약, menu_description 을 바꾸고 싶다면? 100,000 개의 데이터에서 찾아 전부 바꿔야한다

id	name	description	quantity	request_detail
1	카페라떼	라떼는.. 말이야	1	얼음 많이많이 주세요.
2	카페라떼	라떼는.. 말이야	3	시럽 넣지 말아주세요.
3	아메리카노	아메아메 좋아	1	
4	복숭아 아이스티	상큼한 복숭아!	2	

관계형 데이터베이스 (Relational Database)

- 여러 테이블의 "관계"로 이루어진다.
- 핵심은 FK (Foreign Key, 외래키) 와 JOIN
- 중복 제거
 - 만약 100,000 개의 데이터를 저장하더라도,
 - DB 용량을 훨씬 효율적으로 사용
 - 만약, menu_description 을 바꾸고 싶다면? menus 테이블에서 "단 하나만" 변경하면 끝

menus

id	name	description	image_src
1	카페라떼	라떼는.. 말이야	public/image1.jpg
2	카푸치노	언빌리 "버블"	public/image2.jpg
3	아메리카노	아메아메 좋아	public/image3.jpg
4	복숭아 아이	상큼한 복숭아!	public/image4.jpg

orders

id	quantity	request_detail	menus_id
1	1	얼음 많이많이 주세요.	1
2	3	시럽 넣지 말아주세요.	2
3	1	우유 많이 주세요.	2
4	2	복숭아 좋아요.	4

JOIN 문법을 사용하는 이유

- `orders`에서, 주문마다 `menus_id` 보유
- 즉, `id`만 알고 있다면 `menus`에 접근해서
 - 메뉴 이름, 메뉴 설명, 메뉴 이미지에 대한 정보를 가져올 수 있음
 - 이를 위해 **JOIN 문법** 사용

menus			
id	name	description	image_src
1	카페라떼	라떼는.. 말이야	public/image1.jpg
2	카푸치노	언빌리 "버블"	public/image2.jpg
3	아메리카노	아메아메 좋아	public/image3.jpg
4	복숭아 아이	상큼한 복숭아!	public/image4.jpg

orders			
id	quantity	request_detail	menus_id
1	1	얼음 많이많이 주세요.	1
2	3	시럽 넣지 말아주세요.	2
3	1	우유 많이 주세요.	2
4	2	복숭아 좋아요.	4



menus

- 메뉴 부터 먼저 작성해야한다.
- 메뉴 <-> 주문이 외래키로 엮여있기 때문에 주문을 작성하기 위해서는 메뉴 id가 필요하다.
 - 메뉴에 존재하지 않는 id를 주문의 menus_id에 넣는다면 에러가 발생할것
- worbench를 통해 작성하기

menus			
id	name	description	image_src
1	카페라떼	라떼는.. 말이야	public/image1.jpg
2	카푸치노	언빌리 "버블"	public/image2.jpg
3	아메리카노	아메아메 좋아	public/image3.jpg
4	복숭아 아이	상큼한 복숭아!	public/image4.jpg

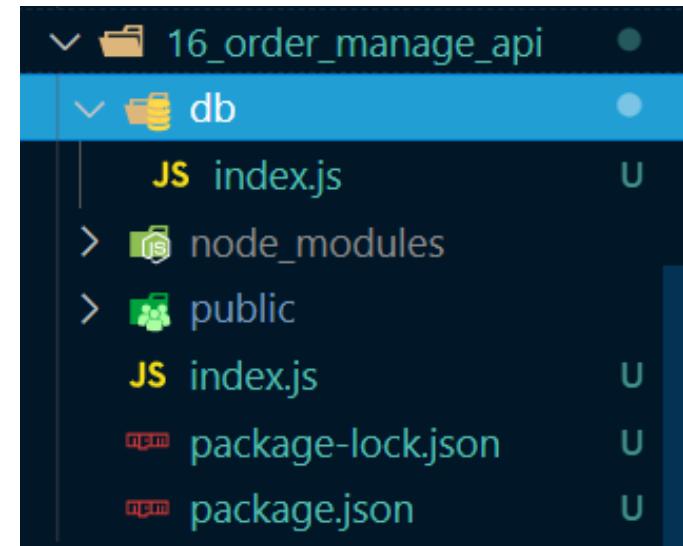
orders

- menus id값이 존재하는 경우만 menus_id 에 데이터를 기입할 수 있다.
- worbench를 통해 작성하기

orders			
id	quantity	request_detail	menus_id
1	1	얼음 많이많이 주세요.	1
2	3	시럽 넣지 말아주세요.	2
3	1	우유 많이 주세요.	2
4	2	복숭아 좋아요.	4

필요 라이브러리

- **express**
 - node.js 프레임워크
- **mysql2**
 - mysql 접속 라이브러리
- **cors**
 - cors 이슈 해결
- **morgan**
 - HTTP 요청에 대한 로그
- **multer**
 - 이미지 업로드를 위한 라이브러리
- **npm init으로 package.json 생성**
 - npm i express mysql2 cors morgan multer



```
"dependencies": {  
    "cors": "^2.8.5",  
    "express": "^4.18.1",  
    "morgan": "^1.10.0",  
    "multer": "^1.4.5-lts.1",  
    "mysql2": "^2.3.3"  
}
```

좋은 API 설계하기

- 유저 정보 조회
 - 유저 전체 정보 조회
 - GET /users
 - 유저 등록
 - POST /users
 - 특정 유저 조회
 - GET /users/:id
 - 특정 유저 수정
 - PATCH /users/:id
 - 특정 유저 삭제
 - DELETE /users/:id
- 대상에 대한 행동은 모두 HTTP REQUEST METHOD로 표시한다.
 - 행동과 리소스를 구별해서 설계하는것이 핵심이다.
 - 리소스 : 유저
 - 행동: 조회(GET), 등록(POST), 수정(PATCH), 삭제(DELETE)

필요한 기능을 생각해본다

- 메뉴 등록, 조회, 수정, 삭제
 - menus
 - GET /api/menus
 - 메뉴 전체 조회
 - GET /api/menus/:id
 - 메뉴 일부 조회
 - POST /api/menus
 - 메뉴 등록
 - PATCH /api/menus/:id
 - 메뉴 수정
 - POST /api/menus/:id/image
 - 메뉴 이미지 수정
 - DELETE /api/menus/:id
 - 메뉴 삭제
- 주문 하기, 주문 조회
 - orders
 - GET /api/orders
 - 주문 전체 조회
 - POST /api/orders
 - 주문하기

GET /api/menus

- 모든 메뉴 목록을 가져온다.
- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
 - [{
 "id": 1,
 "name": "아이스 아메리카노",
 "description": "여름엔 아아가 진리",
 "image_src": "/public/ice-americano.jpg"
 }, ...]
- 실패 시
 - { success : false, message: "전체 메뉴 목록 조회에 실패하였습니다." } 리턴

GET /api/menus 구현

```
app.get("/api/menus", async (req, res) => {
  try {
    const data = await pool.query("SELECT * FROM menus");
    return res.json(data[0]);
  } catch (error) {
    return res.json({
      success: false,
      message: "전체 메뉴 목록 조회에 실패하였습니다."
    });
  }
});
```

GET /api/menus 요청 결과

GET http://localhost:8080/api/menus

Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies

Body 200 OK 66 ms 687 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [ ]  
2 {  
3   "id": 1,  
4   "name": "카페라떼",  
5   "description": "라떼=.. 말이야",  
6   "image_src": "/public/image1.jpg"  
7 },  
8 {  
9   "id": 2,  
10  "name": "캬푸치노",  
11  "description": "언빌리 \"버블\"",  
12  "image_src": "/public/image2.jpg"  
13 },  
14 {  
15  "id": 3,  
16  "name": "아메리카노",  
17  "description": "아메아메 좋아",  
18  "image_src": "/public/image3.jpg"  
19 },  
20 {  
21  "id": 4,  
22  "name": "녹송아 아이스티",  
23  "description": "상큼한 녹송아!",  
24  "image_src": "/public/image4.jpg"  
25 }  
26 ]
```

GET /api/menus/:id

- 메뉴 목록중 id에 해당하는 한가지를 가져온다.
- 성공 시, 메뉴 객체 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
 - {

```
"id": 1,  
"name": "아이스 아메리카노",  
"description": "여름엔 아아가 진리",  
"image_src": "/public/ice-americano.jpg"
```

}
- 실패 시
 - { success : false, message: " 메뉴 조회에 실패하였습니다." } 리턴

GET /api/menus/:id 구현

```
app.get("/api/menus/:id", async (req, res) => {
  try {
    const data = await pool.query("SELECT * FROM menus WHERE id = ?", [req.params.id]);
    console.log(data[0]);
    return res.json(data[0][0]);
  } catch (error) {
    console.log(error);
    return res.json({
      success: false, message: "메뉴 조회에 실패하였습니다."
    });
  }
});
```

GET /api/menus/:id 요청 결과

The screenshot shows a Postman request configuration for a GET request to `http://localhost:8080/api/menus/1`. The response status is 200 OK with a 21 ms response time and 369 B size. The response body is displayed in Pretty JSON format:

```
1 {  
2   "id": 1,  
3   "name": "카페라떼",  
4   "description": "라떼는.. 말이야",  
5   "image_src": "/public/image1.jpg"  
6 }
```

POST /api/menus

- 메뉴를 추가한다.
- 클라이언트에서 서버로 보낼 JSON은 다음과 같은 형태여야 한다.

```
• {  
    "name": "아이스 아메리카노",  
    "description": "여름엔 아아가 진리",  
    "file": "파일 첨부"  
}
```

- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
 - {
 success:true,
 message: "메뉴 등록에 성공하셨습니다."
 }
- 실패 시
 - {success : false, message: "메뉴 등록에 실패하였습니다." } 리턴

POST /api/menus 구현

- multer 라이브러리를 정의하여 이미지 업로드가 가능하게 해야한다.
- 이미지 업로드 후 해당 파일의 정보를 활용해 DB에 저장할 수 있어야 한다.
- 순서
 - 메뉴의 정보 + 파일 업로드
 - multer를 통해 파일 업로드 진행
 - 파일 업로드 완료후 해당 파일의 경로를 받아온다.
 - 받아온 경로 + 메뉴의 정보를 DB Mysql에 넣는다.

POST /api/menus

- multer 라이브러리 정의
 - path.basename
 - 파일의 확장자만 가져온다
 - fileNameExeptExt
 - 확장자를 제외한 파일 이름을 가져온다
 - saveFileName
 - 확장자 제외 파일이름 + 현재 시간 + 확장자
 - ex) profile1661157882293.jpg

```
const path = require("path");
const multer = require("multer");
const upload = multer({
  storage: multer.diskStorage({
    destination: (req, file, done) => {
      done(null, "public/");
    },
    filename: (req, file, done) => {
      console.log(file);
      const ext = path.basename(file.originalname, ext);
      console.log(ext);
      const fileNameExeptExt = path.basename(file.originalname, ext);
      console.log(fileNameExeptExt);
      const saveFileName = fileNameExeptExt + Date.now() + ext;
      done(null, saveFileName)
    },
  }),
  limits: { fileSize: 5 * 1024 * 1024 },
});
```

파일 이름에 Date.now() 를 추가하는 이유

- 확장자 제외 파일이름 + 현재 시간 + 확장자
- 예시
 - profile.jpg 라는 파일을 업로드했다.
 - 하지만 다른 사용자가 profile.jpg라는 파일을 또 업로드했다.
 - 기존의 profile.jpg가 덮어씌워진다.
 - 따라서 Date.now() 를 붙여서 현재 시 분 초의 정보를 같이 업로드하게 되면 사실상 파일 이름이 중복돼 덮어씌워질 일이 없다.

POST /api/menus 구현

- multer가 성공적으로 수행되면 req.file에 파일 정보가 담겨진다.
- pool.query("쿼리문?", [값1])
 - ? 개수만큼 배열에 대입이 가능

```
app.post('/api/menus', upload.single('file'), async(req, res) => {
  try {
    console.log(req.file);
    console.log(req.file.path);
    console.log(req.body);
    const data = await pool.query(`INSERT INTO menus (name, description, image_src)
      VALUES (?, ?, ?)`, [req.body.name, req.body.description, req.file.path]);
  }
  return res.json({
    success: true, message: "메뉴 등록에 성공하였습니다."
  });
} catch (error) {
  console.log(error);
  return res.json({
    success: false, message: "메뉴 등록에 실패하였습니다."
  });
}
})
```

POST /api/menus 결과

POST http://localhost:8080/api/menus

Send

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies

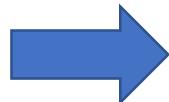
form-data

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> file	img3.jpg			
<input checked="" type="checkbox"/> name	아이스 바닐라 라떼			
<input checked="" type="checkbox"/> description	아 바 라!			

Body **Pretty** Raw Preview Visualize JSON ↻

200 OK 28 ms 335 B Save Response ↻

```
1 "success": true,
2 "message": "메뉴 등록에 성공하였습니다."
```



Result Grid Filter Rows: Edit: Export/Import:

	id	name	description	image_src
1	카페라떼	라떼는.. 말이야	public/image1.jpg	
2	카푸치노	언빌리 "버블"	public/image2.jpg	
3	아메리카노	아메아메 좋아	public/image3.jpg	
4	복숭아 아이스티	상큼한 복숭아!	public/image4.jpg	
10	아이스 바닐라 라떼	아 바 라!	public/img31661162914971.jpg	NULL
*	NULL	NULL	NULL	NULL

PATCH /api/menus/:id

- **id를 받아와서 이미지를 제외한 메뉴를 수정한다.**
 - 이미지는 POST 요청으로 보내야 한다.
- **클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.**
 - {
 "name": "아이스 아메리카노",
 "description": "여름엔 아아가 진리",
}
- **성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.**
 - {
 success:true,
 message: "메뉴 정보 수정에 성공하셨습니다."
}
- **실패 시**
 - {success : false, message: "메뉴 정보 수정에 실패하였습니다." } 리턴

PATCH /api/menus/:id 구현

```
app.patch('/api/menus/:id', async(req, res) => {
  try {
    console.log(req.params);

    const data = await pool.query(`UPDATE menus SET name = ?, description = ? WHERE id = ?`,
      [req.body.name, req.body.description, req.params.id]);

    return res.json({
      success: true, message: "메뉴 정보 수정에 성공하였습니다."
    });
  } catch (error) {
    console.log(error);
    return res.json({
      success: false, message: "메뉴 정보 수정에 실패하였습니다."
    });
  }
})
```

PATCH /api/menus/:id 결과

PATCH http://localhost:8080/api/menus/10

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies

raw JSON Beautify

```
1 {  
2   "name": "바닐라 아이스크림",  
3   "description": "부드러운 바닐라 아이스크림"  
4 }
```

Body **Pretty** Raw Preview Visualize JSON **Save Response**

```
1 {  
2   "success": true,  
3   "message": "메뉴 정보 수정에 성공하였습니다."  
4 }
```

GET http://localhost:8080/api/menus/10

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies

raw JSON Beautify

```
1 {  
2   "id": 10,  
3   "name": "바닐라 아이스크림",  
4   "description": "부드러운 바닐라 아이스크림",  
5   "image_src": "public\\img31661162914971.jpg"  
6 }
```

Body **Pretty** Raw Preview Visualize JSON **Save Response**

POST /api/menus/:id/image

- **id를 받아와서 이미지만 수정한다**
 - 이미지는 POST 요청으로 보내야 하므로 POST로 처리한다.
- **클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.**
 - {
 “file”: “file 정보”,
}
- **성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.**
 - {
 success:true,
 message: “메뉴 이미지 수정에 성공하셨습니다.”
}
- **실패 시**
 - {success : false, message: ”메뉴 이미지 수정에 실패하였습니다.” } 리턴

POST /api/menus/:id/image 구현

- 파일 업로드
- 업로드 후 src 업데이트

```
app.post('/api/menus/:id/image', upload.single('file'), async(req, res) => {
  try {
    const data = await pool.query(`UPDATE menus SET image_src= ? WHERE id = ?`,
      [req.file.path, req.params.id]);
    return res.json({
      success: true, message: "메뉴 이미지 수정에 성공하였습니다."
    });
  } catch (error) {
    console.log(error);
    return res.json({
      success: false, message: "메뉴 이미지 수정에 실패하였습니다."
    });
  }
})
```

POST /api/menus/:id/image 결과

GET http://localhost:8080/api/menus/10/

Body

KEY	VALUE	DESCRIPTION	...	Bulk Edit
file	img1.jpg			

Body

```
Pretty Raw Preview Visualize JSON
1 {
2   "id": 10,
3   "name": "바닐라 아이스크림",
4   "description": "부드러운 바닐라 아이스크림",
5   "image_src": "public\\img11661163813635.jpg"
6 }
```

GET http://localhost:8080/api/menus/10/

Body

```
Pretty Raw Preview Visualize JSON
1 {
2   "id": 10,
3   "name": "바닐라 아이스크림",
4   "description": "부드러운 바닐라 아이스크림",
5   "image_src": "public\\img11661163813635.jpg"
6 }
```

DELETE /api/menus/:id

- id에 해당되는 menus를 삭제한다.
- 요청시 보내는 JSON 데이터는 존재하지 않으며 :id로 삭제만 수행
- 성공 시, 메뉴 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.
 - {
 success: true,
 message: “메뉴 삭제에 성공하셨습니다.”
}
- 실패 시
 - {success : false, message: ”메뉴 삭제에 실패하였습니다.” } 리턴

DELETE /api/menus/:id 구현

```
app.delete('/api/menus/:id', async(req, res) => {
  try {
    const data = await pool.query(`DELETE FROM menus WHERE id = ?`,
      [req.params.id]);

    return res.json({
      success: true, message: "메뉴 삭제에 성공하였습니다."
    });
  } catch (error) {
    console.log(error);
    return res.json({
      success: false, message: "메뉴 삭제에 실패하였습니다."
    });
  }
})
```

DELETE /api/menus/:id 결과

DELETE http://localhost:8080/api/menus/10 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 28 ms 345 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "success": true,  
3   "message": "메뉴 이미지 삭제에 성공하였습니다."  
4 }
```

GET http://localhost:8080/api/menus/10 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 17 ms 224 B Save Response

Pretty Raw Preview Visualize JSON

```
1
```

GET /api/orders

- 모든 주문 목록을 가져온다.
- 성공 시, 주문 목록 배열을 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.

- [

```
"id": 1,  
"name": "아이스 아메리카노",  
"quantity": 1,  
"request_detail": "뜨겁게 만들어주세요"  
}, ...]
```

힌트: JOIN을 사용해 menus_id에 해당하는 name을 가져와야한다.

- 실패 시

- { success: false, message: "전체 주문 목록 조회에 실패하였습니다." } 리턴

GET /api/orders 구현

- JOIN 활용

- 주문수, 주문시 요청, 음료 이름, 음료 설명
- a(orders)의 id인지 b(menus)의 id인지 명시
 - 명시 안하게 되는 경우 에러 발생
 - 컬럼의 이름이 서로 같으면 에러가 발생하기 때문에 id를 명시해줘야 한다.
- id로 order by 정렬 진행

```
app.get("/api/orders", async (req, res) => {
  try {
    const data = await pool.query(`SELECT a.id, quantity, request_detail, name, description
    FROM orders as a
    INNER JOIN menus as b
    ON a.menus_id = b.id
    ORDER BY a.id DESC
  `);
    return res.json(data[0]);
  } catch (error) {
    console.log(error);
    return res.json({
      success: false,
      message: "전체 주문 목록 조회에 실패하였습니다."
    });
  }
});
```

GET /api/orders 결과

- Join을 통해 데이터가 잘 합쳐서 들어옴

The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: http://localhost:8080/api/orders
- Headers: Headers (8) (Auth, Content-Type, Accept, User-Agent, Host, Connection, Pragma, Cache-Control)
- Body: Body (Pretty, Raw, Preview, Visualize, JSON, Copy, Find, Save Response)
- Response status: 200 OK (88 ms, 923 B)

```
1 {
2   "id": 4,
3   "quantity": 2,
4   "request_detail": "복숭아 좋아요.",
5   "name": "복숭아 아이스티",
6   "description": "상큼한 복숭아!"
7 },
8 {
9   "id": 3,
10  "quantity": 1,
11  "request_detail": "우유 많이 주세요.",
12  "name": "카푸치노",
13  "description": "언빌리 \"버블\""
14 },
15 {
16  "id": 2,
17  "quantity": 3,
18  "request_detail": "시럽 넣지 말아주세요.",
19  "name": "카푸치노",
20  "description": "언빌리 \"버블\""
21 },
22 {
23  "id": 1,
24  "quantity": 1,
25  "request_detail": "얼음 많이많이 주세요.",
26  "name": "카페라떼",
27  "description": "라떼는.. 말이야"
28 },
29 }
```

POST /api/orders

- 새로운 주문을 추가한다.
- 클라이언트에서 서버로 보낼 JSON 은 다음과 같은 형태여야 한다.

- {
 "menus_id": 1
 "quantity": 1,
 "request_detail": "뜨겁게 주세요"
}

- 성공 시, 다음 형태의 JSON 을 리턴한다.

- {
 success : true,
 id: 1 // 주문 번호이자 post시 생성된 orders의 id
}

- 실패 시

- {success : false, message: "주문에 실패하였습니다." } 리턴

POST /api/orders 구현

```
app.post('/api/orders', async(req, res) => {
  try {

    const data = await pool.query(`INSERT INTO orders (quantity, request_detail, menus_id)
      VALUES (?, ?, ?)`, [req.body.quantity, req.body.request_detail, req.body.menus_id]);

    return res.json({
      success: true, message: "주문에 성공하였습니다."
    });
  } catch (error) {
    console.log(error);
    return res.json({
      success: false, message: "주문에 실패하였습니다."
    });
  }
})
```

POST /api/orders 결과

POST http://localhost:8080/api/orders Send

Params Auth Headers (8) Body ● Pre-req. Tests Settings

raw JSON

```
1 {  
2   "quantity" : 3,  
3   "request_detail" : "안전하게 배달해 주세요",  
4   "menus_id" : 1  
5 }  
6  
7
```

Body 200 OK 80 ms 328 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "success": true,  
3   "message": "주문에 성공하였습니다."  
4 }
```

GET http://localhost:8080/api/orders Send

Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body 200 OK 75 ms 888 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [  
2   {  
3     "quantity": 3,  
4     "request_detail": "안전하게 배달해 주세요",  
5     "name": "카페라떼",  
6     "description": "라떼는.. 말이야"  
7   },  
8   {  
9     "quantity": 2,  
10    "request_detail": "복숭아 좋아요.",  
11    "name": "복숭아 아이스티",  
12    "description": "상큼한 복숭아!"  
13   },  
14   {  
15     "quantity": 1,  
16     "request_detail": "우유 많이 주세요.",  
17     "name": "카푸치노",  
18     "description": "언밸리 \"버블\""  
19   },  
20 ]
```

orders 에 기능을 추가하기

- **GET /api/orders/:id**
 - 주문 내역 상세 조회 기능
- **PATCH /api/orders/:id**
 - 주문내역 수정 기능
- **DELETE /api/orders/:id**
 - 주문 내역 삭제 기능

GET /api/orders/:id

- 주문 내역을 가져온다.
- 성공 시, 주문 내역 객체를 리턴하되, 각각의 프로퍼티는 다음과 같은 형태여야 한다.

- ```
• {
 "id": 1,
 "name": "아이스 아메리카노",
 "quantity": 1,
 "request_detail": "뜨겁게 만들어주세요"
}
```

**힌트:** JOIN 사용해 menus\_id 에 해당하는 name 가져와야한다.

- 실패 시
  - { success: false, message: "주문 내역 조회에 실패하였습니다." } 리턴

## GET /api/orders/:id 결과

The screenshot shows a Postman request for `http://localhost:8080/api/orders/6`. The response is a 200 OK status with a response time of 80 ms and a body size of 403 B. The response body is displayed in JSON format:

```
1 [
2 {
3 "id": 6,
4 "quantity": 3,
5 "request_detail": "안전하게 배달해 주세요",
6 "name": "카페라떼",
7 "description": "라떼는.. 말이야"
8 }]
9]
```

## PATCH /api/orders/:id

- 기존 주문 내역을 수정한다.
- 클라이언트에서 서버로 보낼 JSON은 다음과 같은 형태여야 한다.

- {  
    "menus\_id": 1  
    "quantity": 1,  
    "request\_detail": "뜨겁게 주세요"  
}

- 성공 시, 다음 형태의 JSON을 리턴한다.

- {  
    success : true,  
    message: "주문 수정에 성공하셨습니다."  
}

- 실패 시

- {success : false, message: "주문 수정에 실패하였습니다." } 리턴

## PATCH /api/orders/:id 결과

PATCH http://localhost:8080/api/orders/6 Send

Params Auth Headers (8) Body **Pre-req.** Tests Settings

raw JSON

```
1 {
2 "quantity": 3,
3 "request_detail": "종이 빨대는 빼고 주세요.",
4 "menus_id": 2
5 }
```

Body Pretty Raw Preview Visualize JSON

```
1 {
2 "success": true,
3 "message": "메뉴 정보 수정에 성공하였습니다."
4 }
```



GET http://localhost:8080/api/orders/6 Send

Params Auth Headers (8) Body **Pre-req.** Tests Settings Cookies

raw JSON

```
1 {
2 "quantity": 3,
3 "request_detail": "종이 빨대는 빼고 주세요.",
4 "menus_id": 2
5 }
```

Body Pretty Raw Preview Visualize JSON

```
1 [{
2 "id": 6,
3 "quantity": 3,
4 "request_detail": "종이 빨대는 빼고 주세요.",
5 "name": "카푸치노",
6 "description": "언밸리 \"버블\""
7 }]
```

## DELETE /api/orders/:id

- 기존 주문 내역을 삭제한다.
- 성공 시, 다음 형태의 JSON 을 리턴한다.

- {

- success : true,

- message: “주문 삭제에 성공하셨습니다.”

- }

- 실패 시

- {success : false, message: ”주문 삭제에 실패하였습니다.” } 리턴

## DELETE /api/orders/:id 결과

The screenshot shows a Postman request for `http://localhost:8080/api/orders/6`. The method is set to `DELETE`. The response status is `200 OK` with a response time of `76 ms` and a size of `342 B`. The response body is:

```
1 {
2 "success": true,
3 "message": "주문 내역 삭제에 성공하였습니다."
4 }
```

## API 서버 배포

## MobaXterm 접속

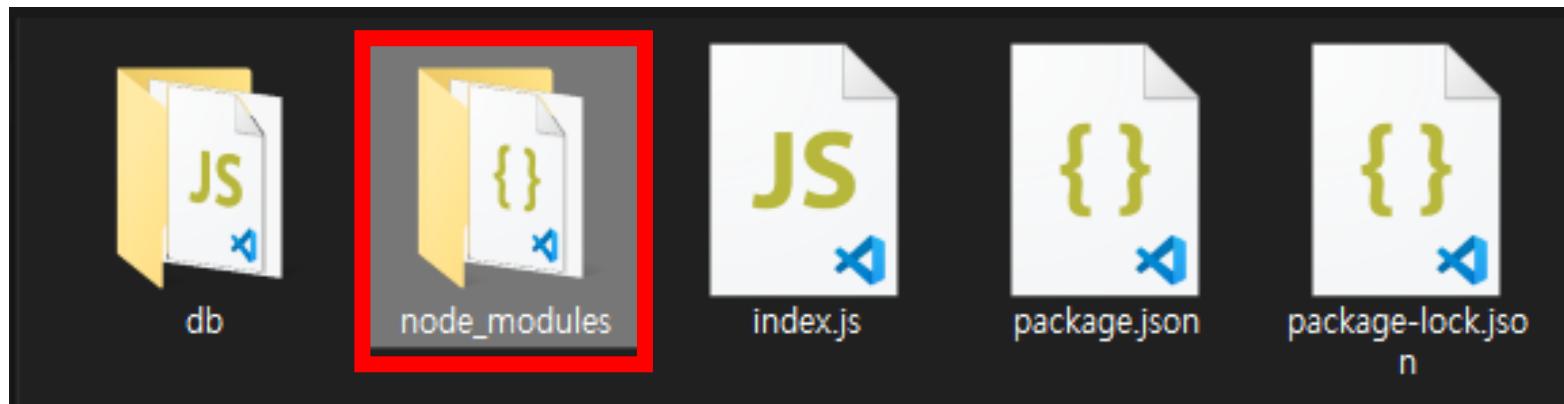
- AWS 인스턴스에 Node.js LTS 16 버전 설치하기

- curl -fsSL https://deb.nodesource.com/setup\_lts.x | sudo -E bash -
- sudo apt update
- sudo apt install -y nodejs
- 버전체크
  - node -v
  - npm -v

```
ubuntu@ip-172-31-40-222:~$ node -v
v16.15.1
ubuntu@ip-172-31-40-222:~$ npm -v
8.11.0
```

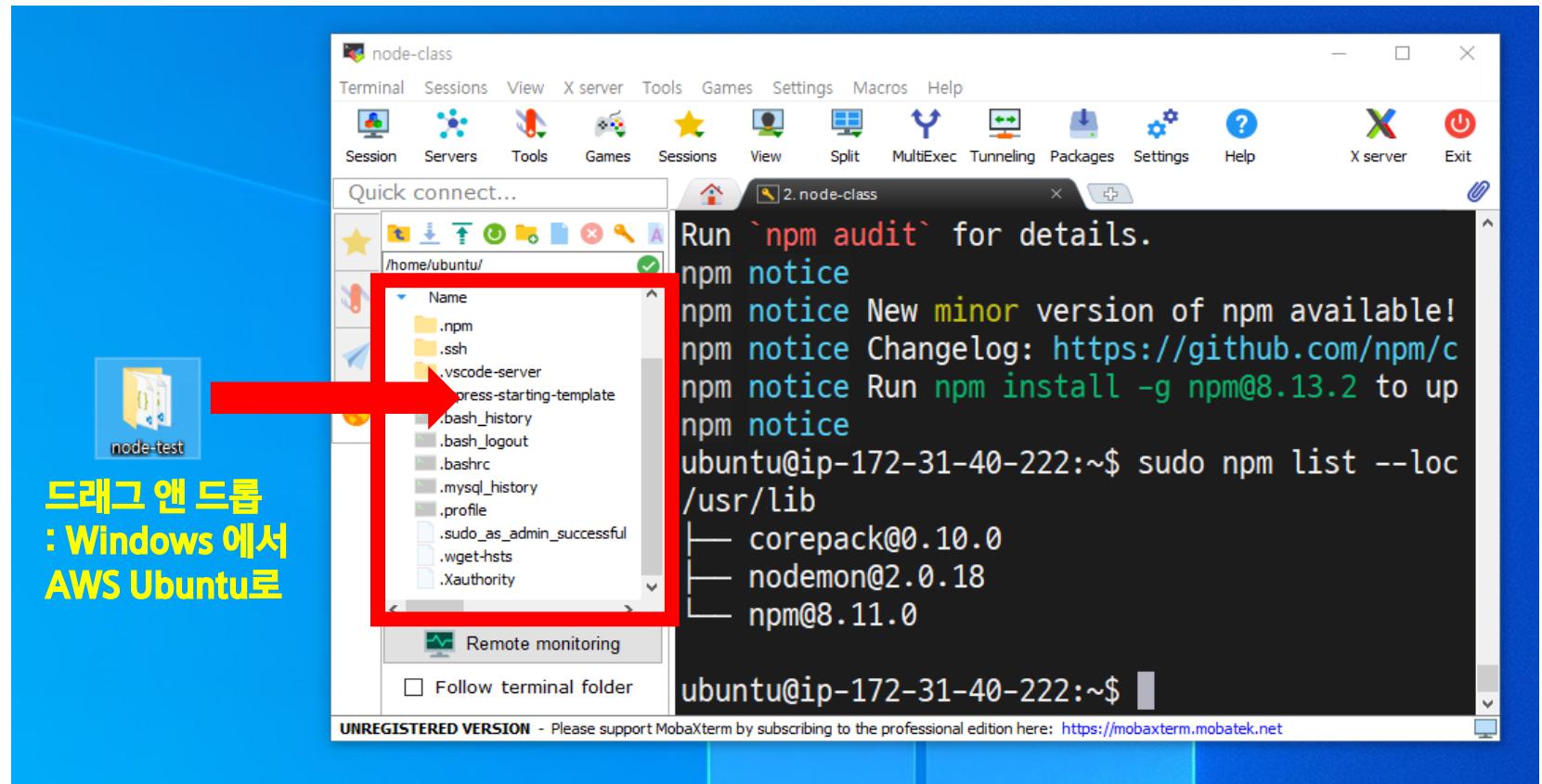
## node\_modules 제외하기

- node\_modules는 용량을 상당히 차지한다.
- 프로젝트 디렉터리에서 node\_modules를 삭제하거나 해당 부분을 빼고 업로드
- 업로드 된 서버에서 해당 node\_modules를 package.json을 통해 재 설치
- github를 통해 협업시 .gitignore 파일에 node\_modules를 추가
  - github에 node\_modules가 올라가지 않도록 한다.



# API 서버 배포

- 작성한 node 폴더를 드래그&드롭을 통해 AWS ubuntu에 넣는다.



## 프로젝트 디렉터리로 이동

\$ cd ~/프로젝트이름

\$ ls -al

## 필요한 패키지 설치

- sudo npm i

## 서버 구동

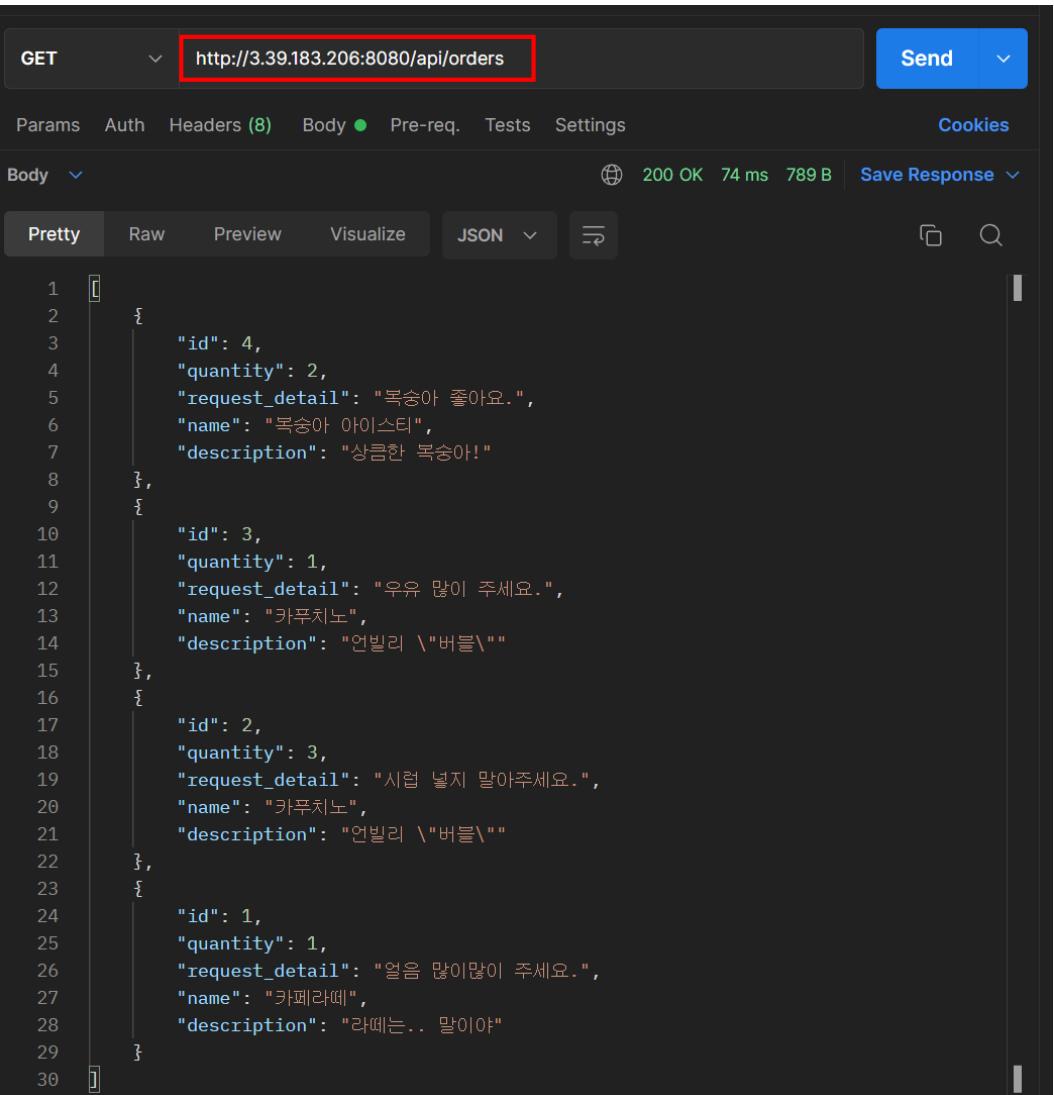
- sudo node index.js
- nodemon을 사용하지 않는 이유
  - 운영 환경은 자동 재시작이 되면 X

```
ubuntu@ip-172-31-30-8:~/16_order_manage_api$ sudo npm i
changed 96 packages, and audited 97 packages in 3s
7 packages are looking for funding
 run `npm fund` for details

ubuntu@ip-172-31-30-8:~/16_order_manage_api$ node index
this server listening on 8080
```

# POSTMAN 요청 주소 변경

- AWS 주소:8080에 요청을 보내보기
- 이전에 지정한 도메인 주소
  - https://내도메인.한국에 접속 및 로그인
  - 해당 도메인:8080/api/orders에 요청 보내기



```
GET http://3.39.183.206:8080/api/orders
```

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Body

Pretty Raw Preview Visualize JSON

```
1 [
2 {
3 "id": 4,
4 "quantity": 2,
5 "request_detail": "복숭아 좋아요.",
6 "name": "복숭아 아이스티",
7 "description": "상큼한 복숭아!"
8 },
9 {
10 "id": 3,
11 "quantity": 1,
12 "request_detail": "우유 많이 주세요.",
13 "name": "카푸치노",
14 "description": "언빌리 \"버블\""
15 },
16 {
17 "id": 2,
18 "quantity": 3,
19 "request_detail": "시럽 널지 말아주세요.",
20 "name": "카푸치노",
21 "description": "언빌리 \"버블\""
22 },
23 {
24 "id": 1,
25 "quantity": 1,
26 "request_detail": "얼음 많이많이 주세요.",
27 "name": "캬파라떼",
28 "description": "라떼는.. 말이야"
29 }
30]
```

## 서버 상시 유지하기

- 현재 MobaXterm을 접속 종료하거나  
Ctrl + C 를 눌러 서버를 나가게 되면 더 이상 요청이 동작하지 않는다.
- 백그라운드에서 서버를 실행시켜 주는 작업이 필요
- **sudo nohup node index.js &**
  - 해당 명령어를 실행하면 node가 터미널이 닫히더라도 백그라운드에서 실행이 가능하다.

```
ubuntu@ip-172-31-30-8:~/16_order_manage_api$ sudo nohup node index.js &
[1] 30981
ubuntu@ip-172-31-30-8:~/16_order_manage_api$ nohup: ignoring input and appending output to 'nohup.out'
```

## 백그라운드 된 서버 종료하는 법

- **ps -ef |grep node 명령어 실행**
  - 빨간색으로 가리킨 곳이 PID 파트
  - kill -9 PID
    - 실행하면 백그라운드로 서비스되고있는 node를 종료 가능하다.
    - ex) kill -9 30991
      - 30991 PID를 가지고있는 서비스 종료

```
ubuntu@ip-172-31-30-8:~/16_order_manage_api$
ubuntu@ip-172-31-30-8:~/16_order_manage_api$
ubuntu@ip-172-31-30-8:~/16_order_manage_api$ ps -ef |grep node
root 30981 29429 0 11:42 pts/0 00:00:00 sudo nohup node index.js
root 30982 30981 0 11:42 pts/0 00:00:00 node index.js
ubuntu 30991 29429 0 11:43 pts/0 00:00:00 grep --color=auto node
ubuntu@ip-172-31-30-8:~/16_order_manage_api$ sudo kill -9 30928 30981
```

내일  
방송에서  
만나요!

삼성 청년 SW 아카데미