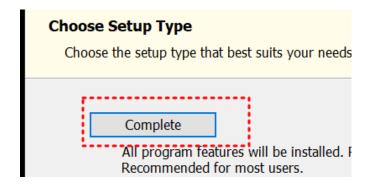
## 몽고 db

- 1. 다운로드 <a href="https://www.mongodb.com/try/download/community">https://www.mongodb.com/try/download/community</a>
- 2. 설치

((1))



((2))

MongoDB 8.0.11 2008R2Plus SSL (64 bit) Service Custo —				
Service Configuration				
Specify optional settings to configure MongoDB as a service.				
✓ Install MongoD as a Service				
Run service as Network Service user				
Run service as a local or domain user:				
Account Domain:				
Account Name:	MongoDB			
Account Password:				

3. MongoDB Compass 실행

### 4. Connection 만들기

### New Connection \*

Manage your connection settings Edit Connection String How do I find my connection -mongodb://localhost:27017/ string in Atlas? If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to Color which you wish to connect. See example 🗗 No Color react-app ☐ Favorite this connection How do I format my Favoriting a connection will pin it to the top of your list of connections connection string? See example 2 **∨** Advanced Connection Options TLS/SSL General Authentication Proxy/SSH In-Use Encryption Advanced **Connection String Scheme** mongodb mongodb+srv Standard Connection String Format. The standard format of the MongoDB connection URI is used to Cancel Connect Save & Connect

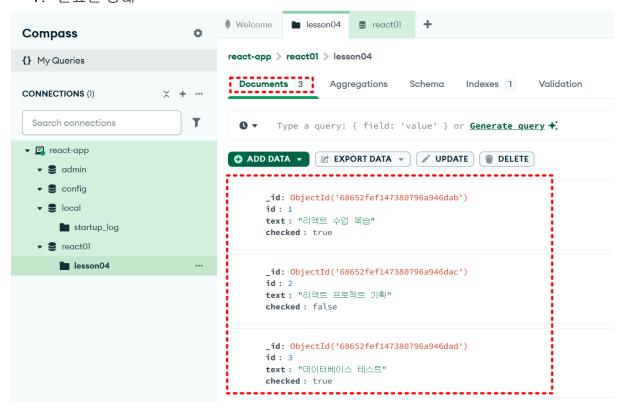
### 5. 데이터베이스 만들기

### **Create Database** Database Name react01 **Collection Name** lesson04 ☐ Time-Series Time-series collections efficiently store sequences of measurements over a period of time. Learn More ➤ Additional preferences (e.g. Custom collation, Clustered collections) ☐ Use Custom Collation Collation allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks. Learn More $^{\mbox{\scriptsize CS}}$ ☐ Clustered Collection Clustered collections store documents ordered by a user-defined cluster key. Learn More Cancel Create Database

### 6. document 에 데이터 저장하기

```
×
Insert Document
To collection react01.lesson04
                                                        VIEW {}
    1 ▼ [
   2 ▼
3 4
5 6
7 ▼
            "id": 1,
"text": "리액트 수업 복습",
            "checked": true
            "id": 2,
"text": "리액트 프로젝트 기획",
    9
            "checked": false
   10
   11
         },
   11
12 ▼
         {
            "id": 3,
"text": "데이터베이스 테스트",
   13
   14
  15
16
   15
            "checked": true
   17
                                                       Cancel
                                                                  Insert
```

### 7. 완료된 상태



## ∰ REST API의 핵심 개념

REST API는 웹 개발에서 서버와 클라이언트 간의 통신을 위한 표준화된 방식이에요. REST는 *Representational State Transfer*의 약자로, 자원을 URI로 표현하고 HTTP 메서드를 통해 자원을 조작하는 아키텍처 스타일

개념 설명

자원(Resource) URI로 식별되는 데이터 (예: /users/1)

표현(Representation) 자원의 상태를 표현하는 방식 (보통

JSON)

메서드(Method) HTTP 메서드로 자원에 대한 작업 수행

상태 전이(State Transfer) 클라이언트가 서버의 자원 상태를 변경

### **1.** 주요 **HTTP** 메서드

메서드 설명 예시

GET 자원 조회 GET /users/1

POST 자원 생성 POST /users

**PUT** 자원 전체 PUT /users/1

수정

PATCH 자원 일부 PATCH /users/1

수정

**DELETE** 자원 삭제 DELETE /users/1

### 2. REST의 6가지 설계 원칙

- 클라이언트-서버 구조: 역할 분리로 독립적인 개발 가능
- 무상태성(Stateless): 요청 간 상태를 서버가 저장하지 않음
- 캐시 가능(Cacheable): 응답을 클라이언트가 저장 가능
- 계층화 구조(Layered System): 중간 서버를 통한 확장 가능
- 일관된 인터페이스(Uniform Interface): 표준화된 URI와 메서드 사용
- 요청 시 코드 전송(Code on Demand) (선택 사항): 클라이언트에 스크립트 전송 가능

## 웹서버 구현 - Node.js

### 1. Node.js 핵심 개념

항목 설명

콜백 함수 (req, res) => {} 요청(request)과 응답(response)을 처리하는 핵심

함수

req.url 요청 경로 확인용

res.writeHead() 응답 헤더 설정 (여기서는 상태코드 200 또는 404)

res.end() 응답 본문을 작성하고 종료

server.listen() 해당 포트로 서버를 실행

### 2. 참고

- 비동기 이벤트 기반 구조: 요청이 들어오면 이벤트 루프가 이를 처리하고, 동시에 다른 요청도 받을 수 있음
- 싱글 스레드: Node.js는 하나의 스레드에서 동작하지만, I/O 작업은 백그라운드에서 처리
- 모듈 시스템: import 사용

### 3. 전체 구조 요약

- Express 프레임워크로 REST API 서버 구현
- MongoDB와 연동해 데이터 저장 및 조회
- CORS 처리로 프론트엔드와 연동 가능
- RESTful API 구성: GET, POST, PUT, DELETE

### 4. 구현 코드 설명

개념 설명

express 웹 서버 프레임워크로 라우팅, 요청 처리 등을 담당

mongodb NoSQL 데이터베이스 클라이언트

cors 다른 도메인의 요청을 허용하는 미들웨어

nodemon 소스 변경 시 자동 재시작 해주는 개발 툴

async/await 비동기 처리 방식으로 MongoDB 작업 처리

REST API /api/todos 경로에 CRUD 엔드포인트 정의

미들웨어 app.use(cors()), app.use(json()) 등

### ✓ API 요약

메서드 경로 기능

GET /api/todos 모든 Todo 목록 조회

POST /api/todos 새 Todo 추가 (최대 id + 1 방식)

PUT /api/todos/:id 특정 Todo의 checked 상태 업데이트

DELETE /api/todos/:id 특정 Todo 삭제

### 5. 주요 코드 개념 정리

### MongoDB 연결

```
MongoClient.connect(MONGODB_URI).then((client) => {
  db = client.db(DB_NAME);
});
```

• 비동기 연결 연결 성공 시 db 객체에 저장하여 이후 API에서 사용

### ☑ app.use() 미들웨어

```
app.use(cors());
app.use(json());
```

- cors(): 프론트엔드 요청 허용
- json(): req.body 파싱 가능하게 설정

# 🔽 미들웨어

Express에서 미들웨어(Middleware)란, 요청(Request)과 응답(Response) 사이에서 동작하며, 공통 로직을 가로채어 처리할 수 있는 함수입니다.

# Express

Express.js는 Node.js 기반의 경량 웹 애플리케이션 프레임워크로, 웹 서버를 손쉽게 구축할 수 있도록 도와줍니다. HTTP 요청에 대한 라우팅, 미들웨어 처리, 에러 처리 등을 직관적으로 구성할 수 있습니다.

# 🔁 라우팅(Routing)

라우팅은 클라이언트가 특정 경로(URL)와 메소드(GET, POST 등)로 서버에 요청을 보냈을때, 어떤 로직/함수가 실행될지 정의하는 것.

◎ 예시: /api/todos로 GET 요청이 오면 목록을 반환

```
app.get("/api/todos", async (req, res) => {
    // 데이터베이스에서 todos를 조회하여 응답
});
```

# 📚 Express에서의 라우팅 구조

### **1.** 기본 구조

app.METHOD(PATH, HANDLER)

요소 설명

app Express 인스턴스

METHOD HTTP 메소드 (GET, POST, PUT, DELETE 등)

PATH URL 경로

HANDLER 요청을 처리할 함수 (req, res 인자 받음)

### 2. 실제 예시

```
// 모든 todo 목록 반환
app.get("/api/todos", (req, res) => { ... });

// 새 todo 등록
app.post("/api/todos", (req, res) => { ... });

// 특정 todo 상태 수정
app.put("/api/todos/:id", (req, res) => { ... });

// 특정 todo 삭제
app.delete("/api/todos/:id", (req, res) => { ... });
```

- :id는 동적 파라미터 (Path Parameter) 로 요청 시 실제 숫자로 대체됨.
- 예:/api/todos/3 → req.params.id = "3"

## 🧠 라우팅이 필요한 이유

- RESTful API 구성
- 기능별 경로 분리 (/users, /posts, /products)
- 클라이언트 요청에 따라 적절한 응답 반환
- MVC 아키텍처와 연계해 컨트롤러 분리 가능

## CORS란? (Cross-Origin Resource Sharing)

CORS는 \*\*다른 도메인(출처, origin)\*\*에서 웹 페이지가 API 요청을 보내는 것을 \*\*브라우저보안 정책(CORS Policy)\*\*에 따라 허용 또는 차단하는 메커니즘입니다.

```
⊕ Origin(출처)이란?
```

```
Origin = 프로토콜 + 도메인 + 포트
```

예시:

- http://localhost:3000 (React 개발 서버)
- http://localhost:5000 (Express API 서버)

이 두 서버는 다른 Origin입니다. 이 경우 브라우저는 CORS 정책에 의해 자동 차단합니다.

○ 문제 상황 예시

프론트엔드 (http://localhost:3000) → 백엔드 (http://localhost:5000)에 요청 시:

```
fetch("http://localhost:5000/api/todos")
```

("Access to fetch at ... from origin ... has been blocked by CORS policy")

해결 방법: Express에서 CORS 허용

Express 서버에서 다음과 같이 설정합니다:

```
import cors from 'cors';
app.use(cors());
```

• 위 설정은 모든 도메인(origin) 요청을 허용합니다.

ण 보안 강화 예시 (특정 도메인만 허용)

```
app.use(cors({
   origin: 'http://localhost:3000',
   credentials: true // 필요한 경우: 쿠키, 인증 포함 요청
}));
```

- origin: 허용할 프론트엔드 주소
- credentials: 인증 토큰, 쿠키 등을 주고받기 위한 설정

### 코드 예시 설명

```
const maxTodo = await db
   .collection(COLLECTION_NAME)
   .findOne({}, { sort: { id: -1 } });
```

구성 요소설명db.collection(COLLECTION\_NAME)MongoDB에서 lesson04 컬렉션을 선택.findOne({})모든 문서 중에서 하나를 조회하되{ sort: { id: -1 } }id 필드를 기준으로 내림차순 정렬하여 가장 큰 값을 가진 문서를 선택await비동기 함수이므로 결과가 올 때까지 기다림maxTodo가장 큰 id 값을 가진 todo 문서 객체를 담음