

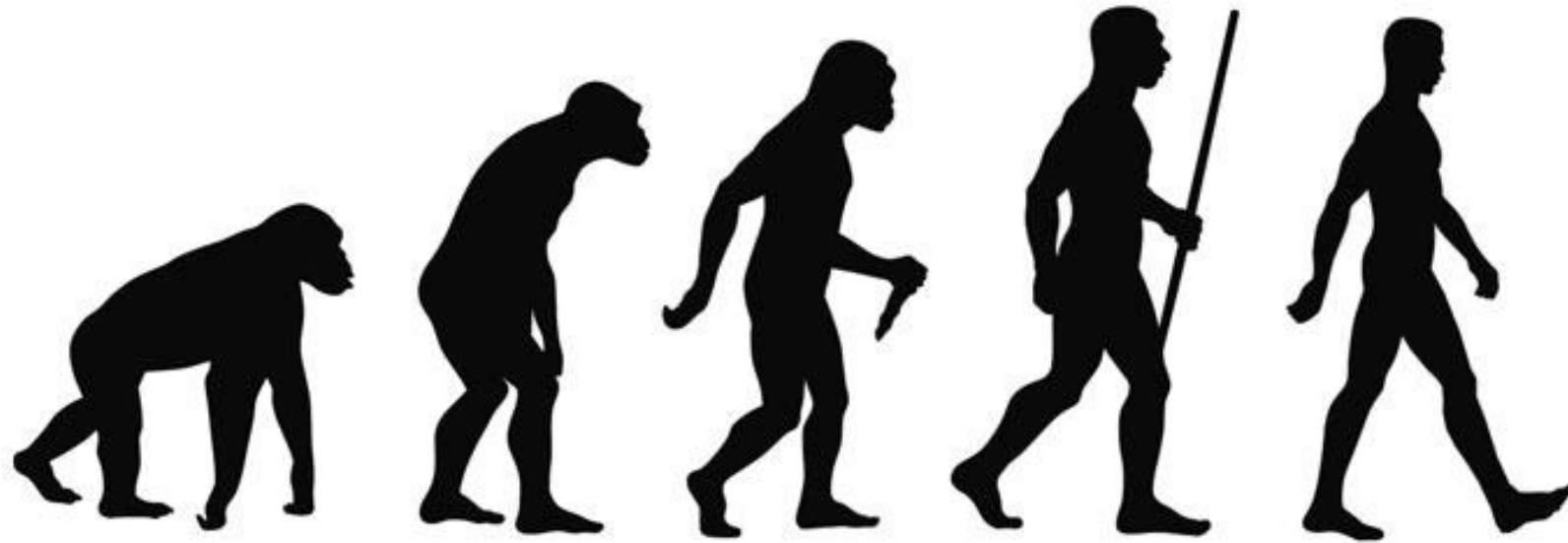
Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with





---

# Refactoring



# 현재 코드의 문제점?

- 컨트롤러와 라우터가 중복된 작업을 합니다~!
  - 하나의 곳에서 처리가 가능할 일을 굳이 둘로 나누어 처리하는 느낌?
  - 그럼!?
- 회원 가입에 대한 처리는 컨트롤러에서 전부 처리하고, 라우터는 주소 연결만 해봅시다!
- 중복 회원 체크 기능을 빼고 회원 가입 기능, 로그인 기능으로 확실하게 처리

## 기존 라우터 코드

```
router.post('/', async (req, res) => {
  const duplicatedUser = await userDB.userCheck(req.body.id);
  if (!duplicatedUser) {
    const registerResult = await userDB.registerUser(req.body);
    if (registerResult) {
      res.status(200);
      res.send('회원 가입 성공!<br><a href="/login">로그인으로 이동</a>');
    } else {
      res.status(500);
      res.send(
        '회원 가입 실패! 알 수 없는 문제 발생<br><a href="/register">회원 가입으로 이동</a>',
      );
    }
  } else {
    res.status(400);
    res.send(
      '동일한 ID를 가진 회원이 존재 합니다!<br><a href="/register">회원 가입으로 이동</a>',
    );
  }
});
```

## 수정 라우터 코드

```
const { registerUser } = require('../controllers/userController');

router.post('/', registerUser);
```



```
const MongoClient = require('./mongoConnect');
const REGISTER_DUPLICATED_MSG =
  '동일한 ID를 가진 회원이 존재 합니다!<br><a href="/register">회원 가입으로 이동</a>';
const REGISTER_SUCCESS_MSG =
  '회원 가입 성공!<br><a href="/login">로그인으로 이동</a>';
const REGISTER_UNEXPECTED_MSG =
  '회원 가입 실패! 알 수 없는 문제 발생<br><a href="/register">회원 가입으로 이동</a>';

const registerUser = async (req, res) => {
  try {
    const client = await MongoClient.connect();
    const user = client.db('kdt5').collection('user');

    const duplicatedUser = await user.findOne({ id: req.body.id });
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);

    await user.insertOne(req.body);
    res.status(200).send(REGISTER_SUCCESS_MSG);
  } catch (err) {
    console.error(err);
    res.status(500).send(REGISTER_UNEXPECTED_MSG);
  }
};

module.exports = {
  registerUser,
};
```

라우터에서 콜백으로 받아온  
req, res 사용

전체 객체를 빼는게 아니라  
필요한 함수만 빼기



# 로그인 라우터 코드 수정!



```
const loginUser = async (req, res) => {
  try {
    const client = await mongoClient.connect();
    const user = client.db('kdt5').collection('user');

    const findUser = await user.findOne({ id: req.body.id });
    if (!findUser) return res.status(400).send(LOGIN_NOT_REGISTERD_MSG);

    if (findUser.password !== req.body.password)
      return res.status(400).send(LOGIN_WRONG_PASSWORD_MSG);

    req.session.userId = req.body.id;
    req.session.login = true;
    res.cookie('user', req.body.id, {
      maxAge: 1000 * 30,
      httpOnly: true,
      signed: true,
    });

    res.status(200);
    res.redirect('/dbBoard');
  } catch (err) {
    console.error(err);
    res.status(500).send(LOGIN_UNEXPECTED_MSG);
  }
};
```

```
const { loginUser } = require('../controllers/userController');

const router = express.Router();

router.get('/', (req, res) => {
  res.render('login');
});

router.post('/', loginUser);
```







# 게시판도

# MongoDB로!



# 전체 게시글 가져오기 컨트롤러



# 새로운 전체 게시물 가져오기 컨트롤러 코드

```
const UNEXPECTED_MSG = '<br><a href="/">메인 페이지로 이동</a>';

const getAllArticles = async (req, res) => {
  try {
    const client = await MongoClient.connect();
    const board = client.db('kdt5').collection('board');

    const allArticleCursor = board.find({});
    const ARTICLE = await allArticleCursor.toArray();
    res.render('db_board', {
      ARTICLE,
      articleCounts: ARTICLE.length,
      userId: req.session.userId,
    });
  } catch (err) {
    console.error(err);
    res.status(500).send(err.message, UNEXPECTED_MSG);
  }
};
```



# 새로운 게시판 페이지 라우터

```
const { getAllArticles } = require('../controllers/boardController');  
  
// 게시판 페이지 호출  
router.get('/', isLoggedIn, getAllArticles);
```



# 글 쓰기

# 페이지로 이동



# 글쓰기로 이동하는 기능은?

- 글쓰기로 이동 할때는 컨트롤러 기능(DB에서 데이터를 받는 작업)을 전혀 사용하지 않으므로 기존 라우터 코드를 그대로 사용합니다!

```
// 게시판 페이지 호출
router.get('/', isLogin, getAllArticles);

// 글쓰기 페이지 호출
router.get('/write', isLogin, (req, res) => {
  res.render('db_board_write');
});
```



글 쓰기

기능 구현



# 새로운 글 쓰기 컨트롤러 코드

```
const writeArticle = async (req, res) => {
  try {
    const client = await mongoClient.connect();
    const board = client.db('kdt5').collection('board');

    const newArticle = {
      UserID: req.session.userId,
      TITLE: req.body.title,
      CONTENT: req.body.content,
    };
    await board.insertOne(newArticle);
    res.redirect('/dbBoard');
  } catch (err) {
    console.error(err);
    res.status(500).send(err.message + UNEXPECTED_MSG);
  }
};

module.exports = { getAllArticles, writeArticle };
```





# 새로운 글 쓰기 라우터 코드

```
const {  
  getAllArticles,  
  writeArticle,  
} = require('../controllers/boardController');  
  
// 데이터 베이스에 글쓰기  
router.post('/write', isLoggedIn, writeArticle);
```



# 글 수정하기

# 코드 수정!

# 새로운 게시물 수정 모드 이동 컨트롤러 코드



```
const getArticle = async (req, res) => {
  try {
    const client = await mongoClient.connect();
    const board = client.db('kdt5').collection('board');

    const selectedArticle = await board.findOne({
      _id: ObjectId(req.params.id),
    });
    res.render('db_board_modify', selectedArticle);
  } catch (err) {
    console.error(err);
    res.status(500).send(err.message + UNEXPECTED_MSG);
  }
};

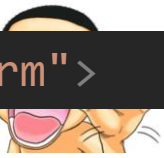
module.exports = { getAllArticles, writeArticle, getArticle };
```



# 새로운 수정 모드로 이동 라우터 코드

```
const {  
  getAllArticles,  
  writeArticle,  
  getArticle,  
} = require('../controllers/boardController');  
  
// 글 수정 모드로 이동  
router.get('/modify/:id', isLoggedIn, getArticle);
```

```
<form action="/dbBoard/modify/<%= selectedArticle._id %>" method="POST" class="board_form">
```



```
const modifyArticle = async (req, res) => {
  try {
    const client = await mongoClient.connect();
    const board = client.db('kdt5').collection('board');

    await board.updateOne(
      { _id: ObjectId(req.params.id) },
      { $set: { TITLE: req.body.title, CONTENT: req.body.content } },
    );
    res.redirect('/dbBoard');
  } catch (err) {
    console.error(err);
    res.status(500).send(err.message + UNEXPECTED_MSG);
  }
};

module.exports = { getAllArticles, writeArticle, getArticle, modifyArticle };
```

```
// 글 수정
router.post('/modify/:id', isLoggedIn, modifyArticle);
```



# 글 삭제하기

# 코드 수정!



# 새로운 삭제 컨트롤러 코드

```
const deleteArticle = async (req, res) => {  
  try {  
    const client = await MongoClient.connect();  
    const board = client.db('kdt5').collection('board');  
  
    await board.deleteOne({ _id: ObjectId(req.params.id) });  
    res.status(200).json('삭제 성공');  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(err.message + UNEXPECTED_MSG);  
  }  
};  
  
module.exports = { getAllArticles, writeArticle, getArticle, modifyArticle, deleteArticle };
```



# 새로운 삭제 라우터 코드

```
const {  
  getAllArticles,  
  writeArticle,  
  getArticle,  
  modifyArticle,  
  deleteArticle,  
} = require('../controllers/boardController');  
  
// 글 삭제  
router.delete('/delete/:id', isLoggedIn, deleteArticle);
```

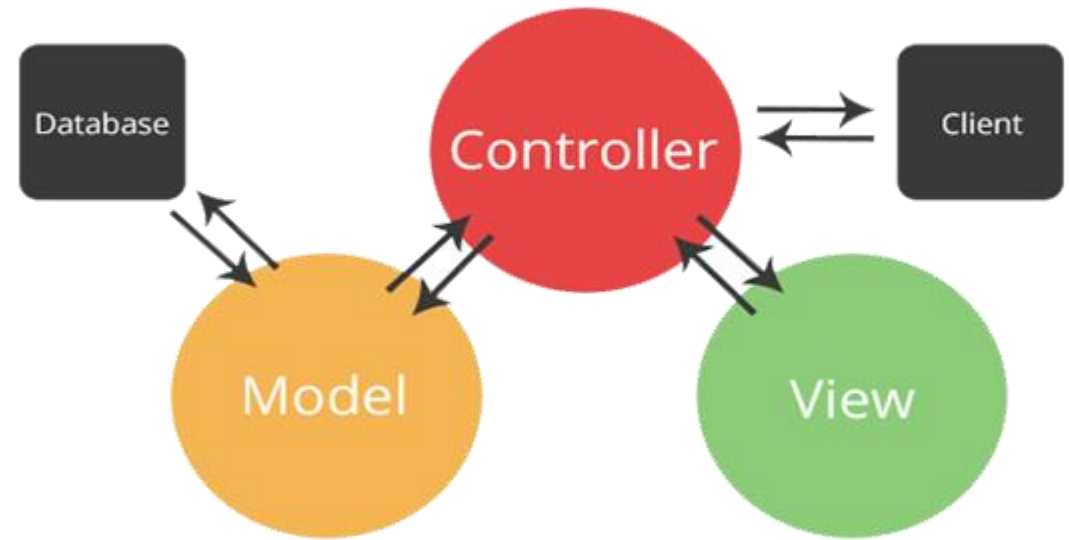




**지금 시작합니다**



# MVC 패턴





Mongoose { 🍃 }



# Mongoose?

- MongoDB는 사용상의 제약이 전혀 없어서 편리 합니다!
- 하지만 너무 편리해서 문제가 생기게 되죠.
- 예를 들어서 어제 만든 user 라는 컬렉션에 임의로 데이터를 추가해 봅시다!

# Mongoose?



- 예를 들어서 어제 만든 user 라는 컬렉션에 임의로 데이터를 추가해 봅시다!

×

Insert to Collection user

VIEW {} ≡

1

`_id: 638b706808787c75c256972b`

ObjectId

2

`userid: 11`

Int32

✖

+

`pw: "11"`

String

Cancel

Insert

```
_id: ObjectId('638aad0ee37936135e72219d')
id: "11"
password: "11"
```

```
_id: ObjectId('638b706808787c75c256972b')
userid: 11
pw: "11"
```



# Mongoose?

- 자 이렇게 되면 어떤 문제가 발생하게 될까요?
- 회원 정보를 가져 올 때 key의 값이 전혀 다르게 되겠죠? 그럼 버그가 발생합니다!
- MySQL 이었다면? 데이터 삽입을 하는 순간에 이미 제약 조건에 의해서 데이터가 삽입이 안되었을 겁니다 → 데이터의 일관성이 유지 되는 것이죠!
- 그럼 MongoDB 에서는 이런 문제를 어떻게 해결할 수 있을까요?



Mongoose { 🌿 }



# Mongoose!



- 바로 몽구스가 해결을 해줍니다!
- MongoDB의 장점은 살리면서 단점을 커버할 수 있게 해주는 것이 바로 Mongoose 모듈입니다!



몽구스

설치



# Mongoose 설치

- Npm i mongoose -S

```
lhs@DESKTOP-86MUCGC MINGW64 /d/git/4th_backend (main)
$ npm i mongoose

added 8 packages, and audited 362 packages in 3s

70 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



# Mongoose 접속 용 모듈 생성

- Controllers 폴더에 mongooseConnect.js 파일 생성
- 이번 기회에 mongoDB 접속 URI 를 dotenv 에 등록 하시죠!

```
PORT = 4000
```

```
MYSQL_USER = root
```

```
MYSQL_PASSWORD = dlrldk
```

```
MYSQL_DB = mydb1
```

```
MDB_URI =
```

```
mongodb+srv://xenosign1:qwer1234@cluster0.8sphltr.mongodb.net/?retryWrites=true&w=majority
```



```
const mongoose = require('mongoose');

const { MDB_URI } = process.env;
const connect = async () => {
  try {
    await mongoose.connect(MDB_URI, {
      dbName: 'kdt5',
      useNewUrlParser: true,
    });

    console.log('몽구스 접속 성공!');

    mongoose.connection.on('error', (err) => {
      console.error('몽고 디비 연결 에러', err);
    });
    mongoose.connection.on('disconnected', () => {
      console.error('몽고 디비 연결이 끊어졌습니다. 연결을 재시도 합니다.');
```

```
      connect();
    });
  } catch (err) {
    console.error(err);
  }
};

module.exports = connect;
```



# User

# 스키마 생성



# 스키마 설정을 위한 Models 폴더 만들기

```
> controllers
> models
> node_modules
> public
> routes
> views
```

- user 스키마를 작성을 위한 user.js 파일 생성



# 회원 스키마 정의하기

- 몽구스 모듈 импорт & 몽구스 모듈의 Schema 클래스 импорт

```
const mongoose = require('mongoose');  
  
const { Schema } = mongoose;
```





# 회원 스키마 정의하기

- userSchema 정의
- \_id 는 알아서 생성 되므로 정의할 필요가 없습니다!
- id : 문자열 / 필수 / 유니크
- password : 문자열 / 필수
- createdAt : 시간 / 생성 시간이 기본으로 삽입
- 컬렉션 이름은 mongoose-user 로 생성!
  - 해당 옵션을 붙이지 않으면 xxxSchema 의 xxx에 s 가 붙는 형태로 생성 됩니다!

```
const mongoose = require('mongoose');
const { Schema } = mongoose;
const userSchema = new Schema(
  {
    id: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    createdAt: {
      type: Date,
      default: Date.now,
    },
  },
  {
    collection: 'mongoose-user',
  },
);
module.exports = mongoose.model('User', userSchema);
```





# 컨트롤러 코드 수정

# 먼저 기본 mongoDB 버전의 컨트롤러 백업!



- 컨트롤러 코드가 수정 될 예정이므로 백업해 둡시다!

```
JS userController_mongo.js U
JS userController_SQL.js
JS userController.js      1, M
```



# 몽고 디비 접속 모듈 및 user 스키마 импорт

```
const connect = require('./mongooseConnect');  
const User = require('../models/user');  
  
connect();
```

- 클라이언트는 한 번만 접속해도 사용이 가능하니 바로 접속을 시킵시다!

```
[nodemon] starting node app.js  
서버는 4000번에서 실행 중입니다!  
몽고 디비 연결 성공  
□
```



# 회원 가입

# 컨트롤러 작성



# 컨트롤러 변경을 별게 없습니다~!

- 기존에는 컬렉션을 선택해서 컬렉션에 쿼리를 날리던 것을 이제는 모델을 불러서 날리면 됩니다~! 그리고 명령어가 조금 다릅니다!

```
const client = await MongoClient.connect();
const user = client.db('kdt5').collection('user');
const duplicatedUser = await user.findOne({ id: req.body.id });
```

```
const mongooseConnect = require('./mongooseConnect');
const User = require('../models/users');
mongooseConnect();
const duplicatedUser = await User.findOne({ id: req.body.id });
```

# 몽구스의 CRUD 쿼리



CRUD	함수명
CREATE	create
READ	find, findById, findOne
UPDATE	updateOne, updateMany, findByIdAndUpdate, findOneAndUpdate
DELETE	deleteOne, deleteMany, findByIdAndDelete, findOneAndDelete

- 생성할 때 insertOne, insertMany 가 아닌 create 문 하나만 씁니다!





# 기존 컨트롤러 코드

```
const registerUser = async (req, res) => {
  try {
    const client = await mongoClient.connect();
    const user = client.db('kdt5').collection('user');

    const duplicatedUser = await user.findOne({ id: req.body.id });
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);

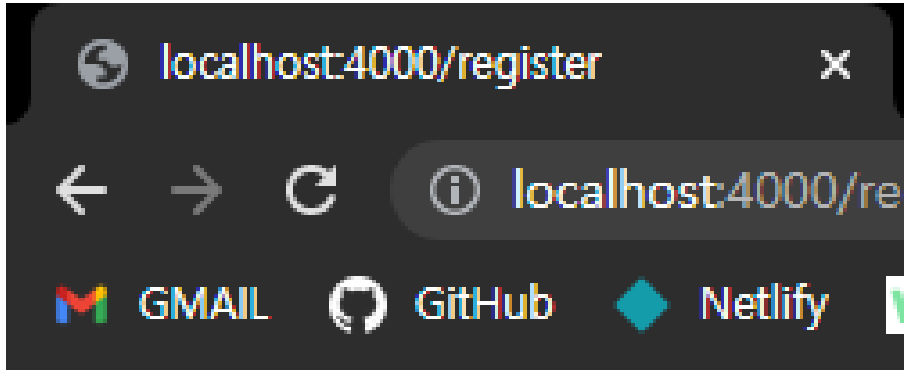
    await user.insertOne(req.body);
    res.status(200).send(REGISTER_SUCCESS_MSG);
  } catch (err) {
    console.error(err);
    res.status(500).send(REGISTER_UNEXPECTED_MSG);
  }
};
```



# 새로운 몽구스 컨트롤러 코드

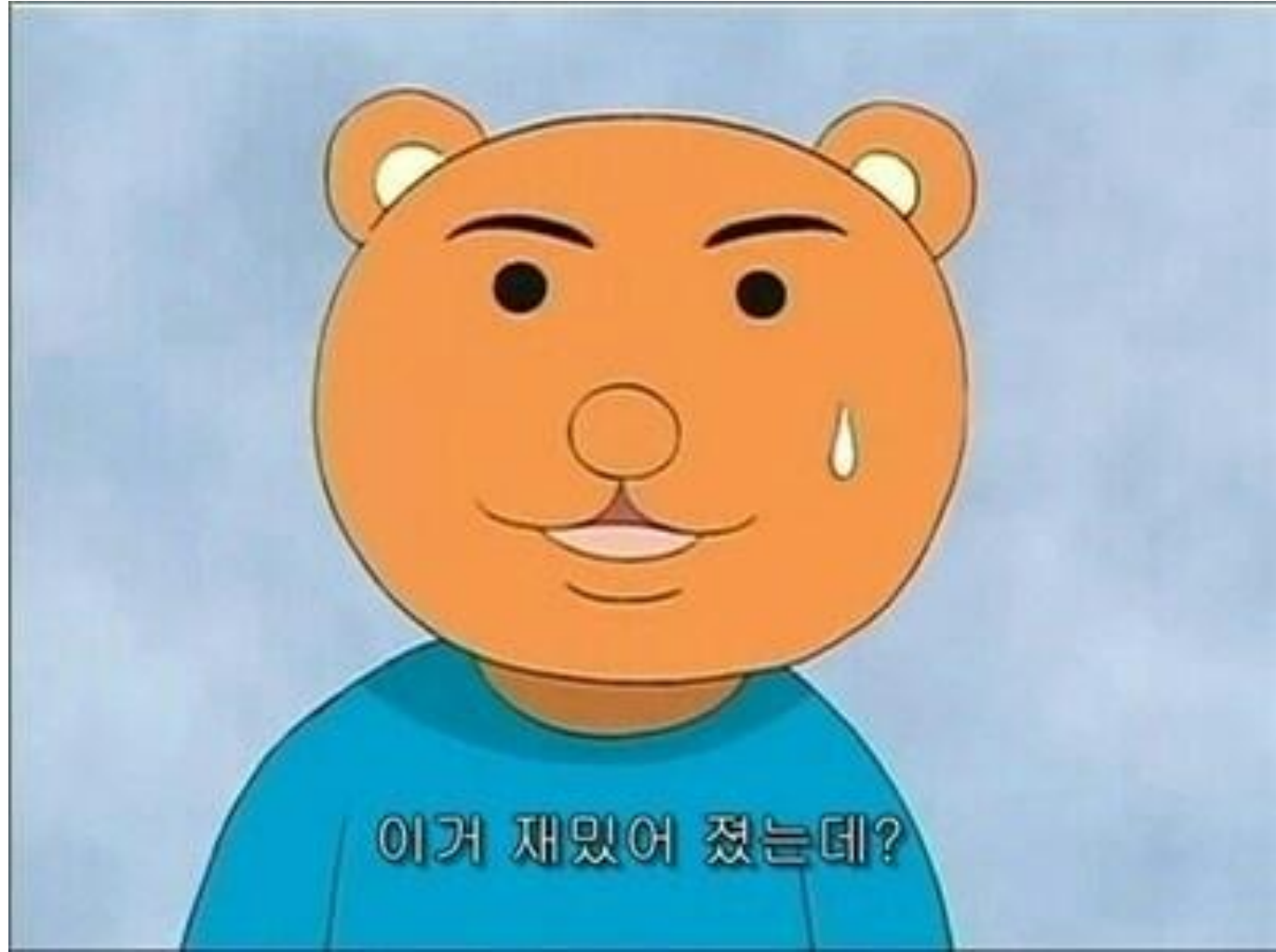
```
const registerUser = async (req, res) => {
  try {
    const duplicatedUser = await User.findOne({ id: req.body.id });
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);

    await User.create(req.body);
    res.status(200).send(REGISTER_SUCCESS_MSG);
  } catch (err) {
    console.error(err);
    res.status(500).send(REGISTER_UNEXPECTED_MSG);
  }
};
```



회원 가입 성공!  
[로그인으로 이동](#)







# 어? 생각보다 잘되네요??????

- 그런데 사실 이걸 Schema 의 역할을 하나도 테스트 못한 상태입니다
- 기존 몽고 디비처럼 그냥 데이터 받아서 넣고 있으니까요! 자 그럼 이제 Schema 에 정의한 것들이 정상 작동하는지 하나하나 보겠습니다!
- 자, 그럼 이제 회원 가입을 할 때 임의의 값을 전달해서 실제로 Schema 가 역할을 하는지 확인해 보겠습니다!



# 필수 값을 다르게 전달 해보기

```
const registerUser = async (req, res) => {  
  try {  
    const duplicatedUser = await User.findOne({ id: req.body.id });  
    if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);  
  
    await User.create({ id: req.body.id, password: '' });  
    res.status(200).send(REGISTER_SUCCESS_MSG);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(REGISTER_UNEXPECTED_MSG);  
  }  
};
```

- 기존 몽고 디비였으면 뭐 그냥 id 라는 키로 데이터를 입력 했겠죠?



서버는 4000번 포트에서 실행 중입니다!

몽구스 접속 성공!

Error: User validation failed: id: Path `id` is required.

at ValidationError.inspect (D:\git\express-board\node\_modules\mongoose\lib\error\validation.js:50:26)

at formatValue (node:internal/util/inspect:782:19)

at inspect (node:internal/util/inspect:347:10)

at formatWithOptionsInternal (node:internal/util/inspect:2167:40)

at formatWithOptions (node:internal/util/inspect:2029:10)

at console.value (node:internal/console/constructor:332:14)

at console.warn (node:internal/console/constructor:365:61)

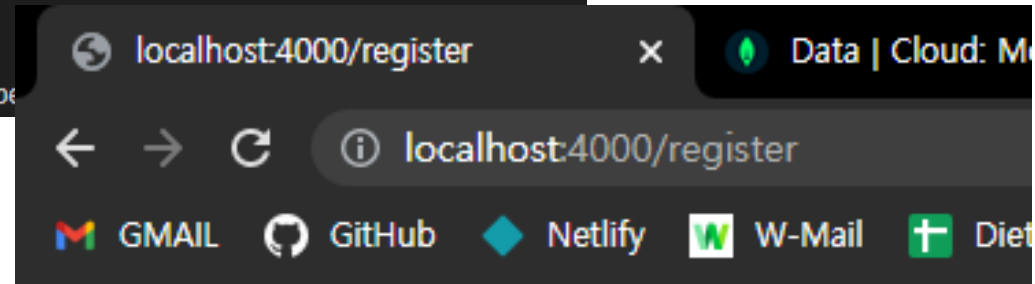
at registerUser (D:\git\express-board\controllers\userController.js:28:13)

at processTicksAndRejections (node:internal/process/task\_queues:96:5) {

errors: {

id: ValidatorError: Path `id` is required.

at validate (D:\git\express-board\node\_modules\mongoose\lib\schematype



회원 가입 실패! 알 수 없는 문제 발생

[회원 가입으로 이동](#)



# 오! 역할을 하는군요!!

- Schema 의 정의를 따르지 않았기 때문에 에러가 발생함을 볼 수 있습니다!
- 즉, 이제 데이터의 완결성은 Schema가 처리를 해주는 것이죠!





# 그렇다면!?

```
const { Schema } = mongoose;  
const userSchema = new Schema(  
  {  
    id: {  
      type: String,  
      required: true,  
      unique: true,  
    },  
  },  
);
```



- 이제 Schema 가 회원 중복 여부를 체크해 줍니다! → 회원 중복 여부를 체크하는 코드를 제거하고 회원 가입을 시도해 봅시다!



# 컨트롤러 코드에서 중복 체크 제거

```
const registerUser = async (req, res) => {  
  try {  
    // const duplicatedUser = await User.findOne({ id: req.body.id });  
    // if (duplicatedUser) return res.status(400).send(REGISTER_DUPLICATED_MSG);  
  
    await User.create(req.body);  
    res.status(200).send(REGISTER_SUCCESS_MSG);  
  } catch (err) {  
    console.error(err);  
    res.status(500).send(REGISTER_UNEXPECTED_MSG);  
  }  
};
```

# 현재 가입 된 회원 정보 확인!



- ▼ kdt5
  - board
  - | mongoose-user
  - test
  - user
  - users
- ▶ login

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('6419e2f05a949c0f')
id: "11"
password: "11"
createdAt: 2023-03-21T17:01:36.
__v: 0
```

## 회원가입

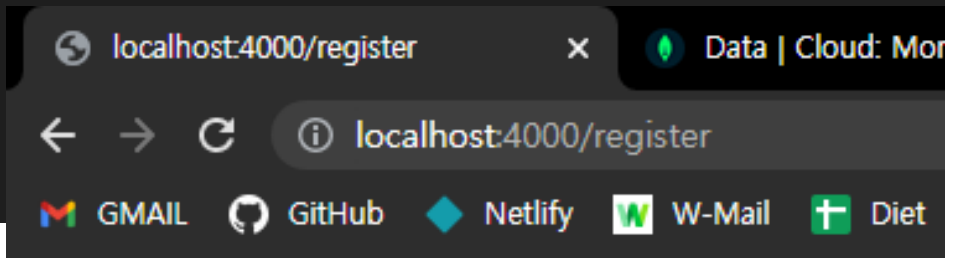
아이디

비밀번호

회원가입



```
MongoServerError: E11000 duplicate key error collection: kdt4.mongoose-user index: id_1 dup key: { id: "11" }  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\operations\insert.js:53:33  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection_pool.js:308:25  
  at D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\sdam\server.js:213:17  
  at handleOperationResult (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\sdam\server.js:329:20)  
  at Connection.onMessage (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection.js:219:9)  
  at MessageStream.<anonymous> (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\connection.js:60:60)  
  at MessageStream.emit (node:events:526:28)  
  at processIncomingData (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\message_stream.js:132:20)  
  at MessageStream._write (D:\git\4th_backend\node_modules\mongoose\node_modules\mongodb\lib\cmap\message_stream.js:33:9)  
  at writeOrBuffer (node:internal/streams/writable:389:12) {  
  index: 0,  
  code: 11000,  
  keyPattern: { id: 1 },  
  keyValue: { id: '11' },  
  [Symbol(errorLabels)]: Set(0) {}  
}
```



회원 가입 실패! 알 수 없는 문제 발생  
[회원 가입으로 이동](#)





# 실습, 로그인 기능도 Model 코드로 변경!

- User 모델이 생성이 되었으므로, 기존 로그인 컨트롤러 코드를 모델을 적용해서 변경해 봅시다!



Multer 모듈로  
이미지 업로드!



# Multer 모듈 설치

- Npm i multer -S
- 파일을 간단하게 업로드 하게 해주는 multer 모듈을 설치해 봅시다!

```
tetz@DESKTOP-P7Q40LL MINGW64 ~/Desktop/KDT/__수업 자료
$ npm i multer
npm WARN config global `--global`, `--local` are deprecated
added 16 packages, and audited 414 packages in 1s

84 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```





# db\_board\_write.ejs 파일 수정

- 이제 파일 업로드 부분이 필요하므로 해당 내용을 추가 합니다

```
<div class="form_img">
  <h3>이미지 업로드</h3>
  <input type="file" name="img" />
</div>
```

- 파일을 업로드 할 때에는 form 데이터가 더 이상 단순 텍스트가이 아니므로 인코딩 타입을 multipart/form-data 을 속성에 추가해 줍니다

```
<form action="/dbBoard/write" method="POST" class="board_form" enctype="multipart/form-data">
```

- 이걸 안써주면 파일 데이터는 안 올라 갑니다 ☹



```
<form action="/board" method="POST" enctype="multipart/form-data" class="board_form">
  <div class="form_title">
    <h3>제목</h3>
    <input type="text" name="title" required />
  </div>
  <div class="form_content">
    <h3>내용</h3>
    <textarea name="content" id="content" cols="30" rows="10" required></textarea>
  </div>
  <div class="form_img">
    <h3>이미지 업로드</h3>
    <input type="file" name="img" />
  </div>
  <button type="submit">글 작성하기</button>
</form>
```



# 글 쓰기

제목

내용

이미지 업로드

파일 선택    선택된 파일 없음

글 작성하기



# dbBoard.js 라우터 파일 수정

- 글을 쓸 때, 이미지 파일을 추가해 줘야 하므로 dbBoard.js 를 수정 합니다
- Multer 및 Filesystem 모듈 불러오기

```
const express = require('express');  
  
const multer = require('multer');  
const fs = require('fs');  
  
const router = express.Router();
```



# dbBoard.js 파일 수정

- 저장 설정
  - Destination : 업로드를 할 폴더 설정
  - Filename: 파일 이름을 설정
- 한계 설정
  - 파일 크기, 이름 같은 제한을 설정 할 수 있습니다



# 저장 폴더 만들기!

- 파일을 /uploads 로 올리기로 했으므로 해당 폴더가 반드시 존재
- 글 쓰기 요청 시, 폴더가 없으면 만들어 줍시다!
- Filesystem 모듈 호출

```
const fs = require('fs');
```

- 폴더가 없으면 만드는 코드 추가

```
if (!fs.existsSync(dir)) fs.mkdirSync(dir);
```



```
const dir = './uploads';
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, dir);
  },
  filename: (req, file, cb) => {
    cb(null, file.fieldname + '_' + Date.now());
  },
});
const limits = {
  fileSize: 1024 * 1028 * 2,
};

const upload = multer({ storage, limits });

if (!fs.existsSync(dir)) fs.mkdirSync(dir);
```

Null 은 모듈을 정상적으로 불러왔는지 테스트하기 위한 인자입니다!

파일 업로드를 위한 multer 모듈을 Upload 라는 변수에 담아 줍니다

서버의 최상단 폴더에 uploads 폴더가 있는지 확인하고 없으면 만들어 줍니다!



# 글쓰기 라우터에 파일 업로드 코드 추가!

- Multer 모듈은 현재 upload 에 들어 있습니다!
- 해당 모듈을 글쓰기 라우터에 isLogin 함수를 넣어 주었던 것 처럼, 미들 웨어로 넣어주면 됩니다!

```
// 글 쓰기  
router.post('/write', isLogin, upload.single('img'), writeArticle);
```





# 파일 업로드 여부에 따른 DB 처리

- 파일이 올라오면 파일의 정보는 req.file 에 담겨서 들어옵니다!
- 그리고 uploads 폴더가 생성되고, 해당 파일이 올라온 것도 확인 가능!

```
{
  fieldname: 'img',
  originalname: 'dog.jpg',
  encoding: '7bit',
  mimetype: 'image/jpeg',
  destination: './uploads',
  filename: 'img_1679521395566',
  path: 'uploads\\img_1679521395566',
  size: 121984
}
```

[nodemon] restarting due to changes

uploads	
img_1679521346019	U
img_1679521395566	U



```
const writeArticle = async (req, res) => {
  try {
    const client = await MongoClient.connect();
    const board = client.db('kdt5').collection('board');

    console.log(req.file);

    const newArticle = {
      UserID: req.session.userId,
      TITLE: req.body.title,
      CONTENT: req.body.content,
      IMAGE: req.file ? req.file.filename : null,
    };
    await board.insertOne(newArticle);
    res.redirect('/dbBoard');
  } catch (err) {
    console.error(err);
    res.status(500).send(err.message + UNEXPECTED_MSG);
  }
};
```

파일이 업로드 되어서  
Req.file 값이 있을 때에만  
IMAGE 프로퍼티에게 파일의 이름  
을 넣어주기



# Board.ejs 수정

- 이미지가 있을 경우 이미지를 띄워주는 코드를 추가해 봅시다!

```
<div class="content">
  <% if (ARTICLE[i].IMAGE !== null && ARTICLE[i].IMAGE !== undefined) { %>
    
  <% } %>
</div>
```

Uploads 폴더에서 파일명으로  
이미지 불러오기

IMAGE 프로퍼티에 값이 있을 때에  
만 이미지 태그를 추가



작성자 : 11

ii

ii



수정

삭제

이건 뭐야???





# Uploads 폴더가 스테틱 폴더가 아닙니다!

- 즉, client 레벨에서는 uploads 폴더에 접근이 안되죠!
- 스테틱 설정으로 고고고!

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



작성자 : 13

dd

dd



수정

삭제



# 실습, 이미지 수정하기 기능 만들기!

- 이미지 업로드는 완성 하였으므로, 이번에는 이미지 수정하기 기능을 만들어 봅시다!
- 게시글 수정 버튼을 누르면 아래와 같이 화면이 뜨도록 해주세요!

# Write Mode



제목

11

내용

11

기존 이미지



이미지 수정

파일 선택

선택된 파일 없음

글 수정하기





# 실습, 이미지 수정하기 기능 만들기!

- 여기서 이미지 업로드를 선택하고 글 수정하기를 누르면 게시글의 이미지가 변경 되도록 작성해 주시면 됩니다!
- 이미지를 업로드 하지 않고 게시글만 수정을 하면 이미지는 그대로 두고 글만 수정이 되도록 구현해 주세요!



서버 배포!







## IDC/서버호스팅 - 모든 서버

서비스 / 가격

(VAT 별도)

### IBM X3250M4 EVENT



- ✓ Model X3250M4
- ✓ CPU Xeon E3-1220v2 (3.1G/4C/4T)
- ✓ RAM DDR3 4GB PC3-10600E
- ✓ HDD SATA 1TB 7.2k
- ✓ 월 사용료 50,000원

상세보기

### Intel CP12T NVMe + 100Mbps 추천



- ✓ Model Intel CP12T NVMe + 100Mbps
- ✓ CPU Intel CPU 2.90GHz (6C/12T)
- ✓ RAM DDR4-SDRAM 8GB
- ✓ HDD M.2 NVMe 500GB
- ✓ 월 사용료 310,000원

상세보기



# PuTTY



# FileZilla





# 최신유행 프로그램









# AWS

# 회원가입



[https://aws.amazon.com/ko/free/?trk=fa2d6ba3-df80-4d24-a453-bf30ad163af9&sc\\_channel=ps&s\\_kwid=AL!4422!3!563761819834!e!!g!!aws&ef\\_id=CjwKCAjw1ICZBhAzEiwAFfvFhEwf7TLp8oTKOSI6WZKp-7\\_TQ1SWO81RJ4TMLUUO4YksgqhIEQ1MhxoCmmUQAvD\\_BwE:G:s&s\\_kwid=AL!4422!3!563761819834!e!!g!!aws&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=\\*all&awsf.Free%20Tier%20Categories=\\*all](https://aws.amazon.com/ko/free/?trk=fa2d6ba3-df80-4d24-a453-bf30ad163af9&sc_channel=ps&s_kwid=AL!4422!3!563761819834!e!!g!!aws&ef_id=CjwKCAjw1ICZBhAzEiwAFfvFhEwf7TLp8oTKOSI6WZKp-7_TQ1SWO81RJ4TMLUUO4YksgqhIEQ1MhxoCmmUQAvD_BwE:G:s&s_kwid=AL!4422!3!563761819834!e!!g!!aws&all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)

# AWS 프리 티어

AWS 플랫폼, 제품 및 서비스를 무료로 체험해 보세요

[AWS 프리 티어에 대해 자세히 알아보기](#) ⓘ

무료 계정 생성

추천

## 스타트업이 받을 수 있는 AWS Credit

AWS Activate는 자격이 있는 스타트업에 AWS 서비스에서 사용 가능한 무료 AWS Credit과 AWS Support를 비롯한 리소스 호스팅을 제공합니다.

[지금 Activate에 가입하기](#) »

## 오퍼 유형

프리 티어를 이용해 100가지가 넘는 제품을 살펴보고 AWS에 구축할 수 있습니다. 사용하는 제품에 따라 세 가지 유형의 무료 오퍼가 제공됩니다. 아래 아이콘을 클릭하여 오퍼를 살펴보세요.



### 무료 평가판

단기 무료 평가판 제품 및 서비스는 특정 서비스를 활성화한 날짜부터 시작



### 12개월 무료

처음 AWS에 가입한 날부터 12개월 동안 사용 가능



### 언제나 무료

이 프리 티어는 만료되지 않으며 모든 AWS 고객이 이용 가능



## AWS에 가입

루트 사용자 이메일 주소  
계정 복구 및 일부 관리 기능에 사용

tetz@spreatics.com

### AWS 계정 이름

계정의 이름을 선택합니다. 이름은 가입 후 계정 설정에서 변경할 수 있습니다.

tetz

이메일 주소 확인

또는

기존 AWS 계정에 로그인



## 이메일 주소 확인

새로운 AWS 계정 생성 프로세스를 시작해 주셔서 감사합니다. 사용자가 본인임을 확인하려고 합니다. 메시지가 표시되면 다음 확인 코드를 입력하세요. 계정을 생성하지 않는 경우에는 이 메시지를 무시해도 됩니다.

확인 코드

**258267**

(이 코드는 10분 동안 유효합니다.)

---

Amazon Web Services는 사용자에게 암호, 신용 카드 또는 은행 계좌 번호를 공개하거나 확인하라고 요청하는 이메일을 보내지 않습니다.

---

이 메시지는 Amazon Web Services, Inc., 410 Terry Ave. North, Seattle, WA 98109. © 2022, Amazon Web Services, Inc.에서 작성하고 배포하였습니다. All rights reserved. AWS는 [Amazon.com](https://www.amazon.com), Inc.의 등록 상표입니다. AWS [개인정보 취급방침](#)을 확인하세요.



# AWS에 가입

## 암호 생성

✔ 이메일 주소가 확인되었습니다. ✕

암호는 AWS에 대한 로그인 액세스를 제공하므로 올바른 정보를 얻는 것이 중요합니다.

루트 사용자 암호

.....

루트 사용자 암호 확인

.....

계속(1/5단계)

또는

기존 AWS 계정에 로그인

## 연락처 정보



AWS를 어떻게 사용할 계획이신가요?

- ☐ 비즈니스 - 업무, 학교 또는 조직의 경우
- ☒ 개인 - 자체 프로젝트의 경우

이 계정에 대해 누구에게 문의해야 하나요?

전체 이름

tetz

전화 번호

 +82



10-3930-1325

국가 또는 리전

대한민국



주소

105-1702

Doosan APT

시

Seodaemoon

시, 도 또는 리전

Seoul

우편 번호

03769

☒ AWS 이용약관 [AWS 이용약관](#) 및 개인정보의 수집 및 이  
용에 대한 사항 [AWS 이용약관](#)에 동의합니다.


계속(2/5단계)

# AWS에 가입



## 결제 정보

신용카드 번호

 신용카드 번호는 필수입니다.



AWS는 현지에서 발급된 대부분의 신용카드를 허용합니다. 결제 옵션에 대해 자세히 알아보려면 [FAQ](#)를 참조하세요.

만료 날짜

카드 소유자 이름

청구지 주소

☒ 내 연락처 주소 사용

105-1702

Seodaemoon Seoul 03769

KR

☐ 새 주소 사용

이메일 주소

이메일 주소는 AWS와의 거래를 위해 VAT 영수증을 발송하는 데 사용됩니다.

**확인 및 계속(3/5단계)**

확인 요금을 승인하기 위해 은행의 웹 사이트로 리디렉션될 수 있습니다.





## 카드 정보입력

카드 인증을 위해 정보를 입력합니다.

카드번호

●●●●

●●●●

●●●●

3545

비밀번호

카드 비밀번호 앞 2자리 숫자

생년월일

✓ 법인공용카드

8자리 숫자 입력(ex.19900101)

① 법인공용카드는 체크 후 사업자번호를 입력해주세요.

✓ 서비스 이용에 대한 전체동의

[약관보기](#)



# AWS에 가입

## 자격 증명 확인

AWS 계정을 사용하려면 먼저 전화번호를 확인해야 합니다. 계속하면 AWS 자동 시스템이 확인 코드 전송을 위해 연락합니다.

확인 코드를 어떻게 보내 드릴까요?

☒ 문자 메시지(SMS)

☐ 음성 통화

국가 또는 리전 코드

대한민국 (+82)

휴대전화 번호

보안 검사

	<div>🔊</div> <div>🔄</div>
--	---------------------------

위에 보이는 문자를 입력하세요.


SMS 전송(4/5단계)



## AWS에 가입

### 자격 증명 확인

코드 확인

 SMS PIN 필요

계속(4/5단계)

문제가 있으신가요? 때로는 확인 코드를 가져오는 데 최대 10분이 걸립니다. 이보다 오래 걸리면 [이전 페이지로 돌아가서](#) 다시 시도하세요.



# AWS에 가입

## Support 플랜 선택

비즈니스 또는 개인 계정에 대한 Support 플랜을 선택합니다. [플랜 및 요금 예시를 비교](#) 해 보세요.  
언제든지 AWS Management Console에서 플랜을 변경할 수 있습니다.

### ☒ 기본 지원 - 무료

- AWS를 처음 시작하는 신규 사용자에게 권장
- AWS 리소스에 대한 연중무휴 24시간 셀프 서비스 액세스
- 계정 및 청구 문제 전용
- Personal Health Dashboard 및 Trusted Advisor에 대한 액세스



### ☐ 개발자 지원 - 시작가는 29 USD/월

- AWS를 체험해보는 개발자에게 권장
- 업무 시간 중 AWS Support에 대한 이메일 액세스
- 12시간(업무 시간 기준) 이내의 응답 시간



### ☐ 비즈니스 지원 - 시작가는 100 USD/월

- AWS 기반 프로덕션 워크로드 실행에 추천
- 이메일, 전화 및 채팅을 통한 연중무휴 24시간 기술 지원
- 1시간 이내의 응답 시간
- Trusted Advisor 모범 사례 권장 사항 전체 세트



엔터프라이즈 수준의 지원이 필요하신가요?

최저 월 15,000 USD로 15분 이내에 응답을 받을 수 있으며 기술 지원 관리자가 배정된 컨시어지 스타일의 서비스를 이용할 수 있습니다. [자세히 알아보기](#)

가입 완료



## 축하합니다.

AWS에 가입해 주셔서 감사합니다.

계정을 활성화하는 중입니다. 이 작업은 몇 분 밖에 걸리지 않습니다. 이 작업이 완료되면 이메일을 받게 됩니다.

**AWS Management Console로 이동**

다른 계정에 가입or 영업 팀에 문의하세요.



## 로그인

☒ 루트 사용자

무제한 액세스 권한이 필요한 작업을 수행하는 계정 소유자입니다. [자세히 알아보기](#)

☐ IAM 사용자

일일 작업을 수행하는 계정 내 사용자입니다. [자세히 알아보기](#)

루트 사용자 이메일 주소

xenosign@naver.com

다음

계속 진행하는 경우 [AWS 고객 계약](#) 또는 AWS 서비스에 대한 기타 계약 및 [개인 정보 보호 정책](#)에 동의하게 됩니다. 이 사이트는 필수 쿠키를 사용합니다. 자세한 내용은 [쿠키 고지](#)를 참조하세요.

————— AWS를 처음 사용하십니까? —————

AWS 계정 새로 만들기



# AWS

# 서버 세팅!



③ 새 위젯 최근 AWS 블로그 게시물(들) 소개합니다. 콘솔 홈의 하단에서 찾을 수 있습니다.



최근에 방문한 서비스 정보



EC2

모든 서비스 보기

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.

AWS Health 정보

열린 문제

0

지난 7일

예정된 변경 사항

비용 및 사용량 정보







  캘리포니아 ▲	
미국 동부 (버지니아 북부)	us-east-1
미국 동부 (오하이오)	us-east-2
미국 서부 (캘리포니아)	us-west-1
미국 서부 (오레곤)	us-west-2
...	
아프리카 (케이프타운)	af-south-1
...	
아시아 태평양 (홍콩)	ap-east-1
아시아 태평양 (자카르타)	ap-southeast-3
아시아 태평양 (뭄바이)	ap-south-1
아시아 태평양 (오사카)	ap-northeast-3
아시아 태평양 (서울)	ap-northeast-2
아시아 태평양 (싱가포르)	ap-southeast-1
아시아 태평양 (시드니)	ap-southeast-2
아시아 태평양 (도쿄)	ap-northeast-1





Q ec2

X

'ec2' 검색 결과

서비스 (7)

기능 (46)

블로그 (362)

설명서 (131,187)


지식 문서 (30)

자습서 (19)


마켓플레이스 (1,564)

서비스


모든 7개 결과 보기

 **EC2** ☆


클라우드의 가상 서버

 **EC2 Image Builder** ☆

OS 이미지 빌드, 사용자 지정 및 배포를 자동화하는 관리형 서비스

 **AWS Compute Optimizer** ☆

워크로드에 최적화된 AWS 컴퓨팅 리소스 권장

 **AWS Firewall Manager** ☆

방화벽 규칙의 중앙 관리

EC2 대시보드

EC2 글로벌 보기

이벤트

태그

제한

▼ 인스턴스

인스턴스 New

인스턴스 유형

시작 템플릿

스팟 요청

Savings Plans

예약 인스턴스 New

전용 호스트

용량 예약

▼ 이미지

AMI New

AMI 카탈로그

▼ Elastic Block Store

볼륨 New

스냅샷 New

수명 주기 관리자 New

▼ 네트워크 및 보안

보안 그룹

탄력적 IP

배치 그룹

리소스

EC2 글로벌 보기 [↗](#)



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

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

ℹ AWS Launch Wizard for SQL Server를 사용하여 AWS에서 Microsoft SQL Server Always On 가용성 그룹을 손쉽게 크기 조정, 구성 및 배포할 수 있습니다. 자세히 알아보기 ×

인스턴스 시작

시작하려면 클라우드의 가상 서버인 Amazon EC2 인스턴스를 시작하십시오.

인스턴스 시작 ▲

서버 마이그레이션 [↗](#)

인스턴스 시작

템플릿으로 인스턴스 시작을 시작합니다.

예약된 이벤트



아시아 태평양 (서울)

예약된 이벤트 없음

서버 마이그레이션

서비스 상태



AWS Health 대시보드 [↗](#)

리전

아시아 태평양 (서울)

상태

✔ 이 서비스가 정상적으로 작동 중입니다.

영역

영역 이름

영역 ID

ap-northeast-2a

apne2-az1

ap-northeast-2b

apne2-az2

ap-northeast-2c

apne2-az3

ap-northeast-2d

apne2-az4





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

예: 내 웹 서버

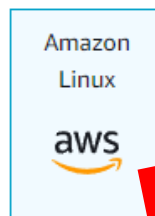
추가 태그 추가

### ▼ 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

#### Quick Start



Amazon Machine Image (AMI)

macOS



Ubuntu



Windows



Red Hat



S



더 많은 AMI 찾아보기

AWS, Marketplace 및 커뮤니티의 AMI 포함

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type

ami-01d87646ef267ced7 (64비트(x86)) / ami-0c501d5f1131cb15e (64비트(Arm))

프리 티어 사용 가능



### ▼ 인스턴스 유형 정보

#### 인스턴스 유형

t2.micro

패밀리: t2 1 vCPU 1 GiB 메모리

온디맨드 Linux 요금: 0.0144 USD 시간당

온디맨드 Windows 요금: 0.019 USD 시간당

프리 티어 사용 가능

[인스턴스 유형 비교](#)

### ▼ 키 페어(로그인) 정보

키 페어를 사용하여 인스턴스에 안전하게 연결할 수 있습니다. 인스턴스를 시작하기 전에 선택한 키 페어에 대한 액세스 권한이 있는지 확인하세요.

#### 키 페어 이름 - 필수

xenosign



새 키 페어 생성



## 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 페어 이름 입력

이름은 최대 255개의 ASCII 문자를 포함할 수 있습니다. 선행 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

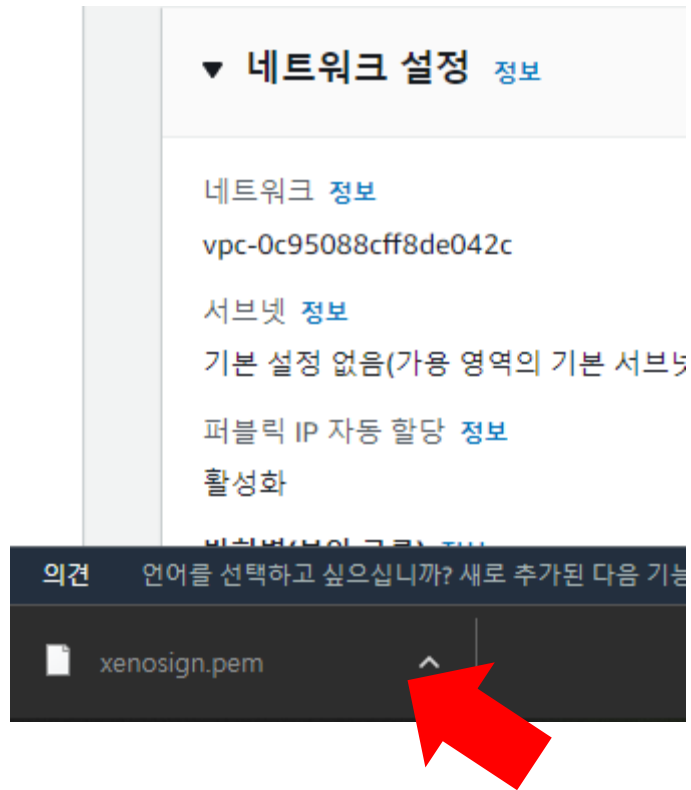
- ☒ .pem  
OpenSSH와 함께 사용
- ☐ .ppk  
PuTTY와 함께 사용

취소

키 페어 생성



## 키페어를 자신이 원하는 폴더에 저장하기





네트워크 정보

vpc-0c95088cff8de042c

서브넷 정보

기본 설정 없음(가용 영역의 기본 서브넷)

퍼블릭 IP 자동 할당 정보

활성화

방화벽(보안 그룹) 정보

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

☒ 보안 그룹 생성

☐ 기존 보안 그룹 선택

다음 규칙을 사용하여 'launch-wizard-2'라는 새 보안 그룹을 생성합니다.

☒ 에서 SSH 트래픽 허용  
인스턴스 연결에 도움

위치 무관  
0.0.0.0/0

☐ 인터넷에서 HTTPS 트래픽 허용  
예를 들어 웹 서버를 생성할 때 엔드포인트를 설정하려면

☐ 인터넷에서 HTTP 트래픽 허용  
예를 들어 웹 서버를 생성할 때 엔드포인트를 설정하려면

소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 설정하는 것이 좋습니다.







▼ 스토리지 구성 정보

[어드밴스드](#)

1x  GiB  ▼ 루트 볼륨

 프리 티어를 사용할 수 있는 고객은 최대 30GB의 EBS 범용(SSD)또는 마그네틱 스토리지를 사용할 수 있습니다. 

새 볼륨 추가

0 x 파일 시스템

[편집](#)

▶ 고급 세부 정보 정보



## 반드시 프리티어인지 확인하기!!

▼ 요약

인스턴스 개수 [정보](#)

1

[소프트웨어 이미지\(AMI\)](#)

Amazon Linux 2 Kernel 5.10 AMI...[더 보기](#)  
ami-01d87646ef267ccd7

[가상 서버 유형\(인스턴스 유형\)](#)

t2.nano

[방화벽\(보안 그룹\)](#)

새 보안 그룹

[스토리지\(볼륨\)](#)

1개의 볼륨 – 8GiB

❗

프리 티어: 첫 해에는 월별 프리 티어 AMI에 대한 t2.micro(또는 t2.micro를 사용할 수 없는 리전의 t3.micro) 인스턴스 사용량 750시간, EBS 스토리지 30GiB, IO 2백만 개, 스냅샷 1GB, 인터넷 대역폭 100GB가 포함됩니다.

×

취소

인스턴스 시작



aws

서비스

서비스, 기능, 블로그, 설명서 등을 검색합니다. [Alt+S]

서울 tetz

New EC2 Experience  
Tell us what you think

×

EC2 대시보드

EC2 글로벌 보기

이벤트

태그

제한

인스턴스 (1) 정보

↺ 연결 인스턴스 상태 ▼ 작업 ▼ 인스턴스 시작 ▼

Q 검색

인스턴스 상태 = running ✕ 필터 지우기

<input type="checkbox"/>	Name ▼	인스턴스 ID	인스턴스 상태 ▼	인스턴스 유형 ▼	상태 검사	경보 상태	가용 영역 ▼	퍼블릭 IPv4 DNS ▼	퍼블릭 IPv4 ... ▼	탄력적 IP ▼
<input type="checkbox"/>	board	i-0b3c7dfe69bcf91e7	✓ 실행 중 🔍	t2.micro	🕒 초기화	경보 없음 +	ap-northeast-2c	ec2-43-200-173-233.ap...	43.200.173.233	-





# 퍼블릭 IP 복사

## i-0b3c7dfe69bcf91e7 (board)에 대한 인스턴스 요약 정보

less than a minute 전에 업데이트됨

[🔄](#) [연결](#) [인스턴스 상태 ▼](#) [작업 ▼](#)

### 인스턴스 ID

[🔗](#) i-0b3c7dfe69bcf91e7 (board)

### IPv6 주소

-

### 호스트 이름 유형

IP 이름: ip-172-31-46-71.ap-northeast-2.compute.internal

### 프라이빗 리소스 DNS 이름 응답

IPv4(A)

### 자동 할당된 IP 주소

[🔗](#) 43.200.173.233 [퍼블릭 IP]

### IAM 역할

-

### 퍼블릭 IPv4 주소

[🔗](#) 43.200.173.233 | [개방 주소법](#) [🔗](#)

### 인스턴스 상태

🟢 실행 중

### 프라이빗 IP DNS 이름(IPv4만 해당)

[🔗](#) ip-172-31-46-71.ap-northeast-2.compute.internal

### 인스턴스 유형

t2.micro

### VPC ID

[🔗](#) vpc-0c95088cff8de042c [🔗](#)

### 서브넷 ID

[🔗](#) subnet-0ce7e077099429665 [🔗](#)

### 프라이빗 IPv4 주소

[🔗](#) 172.31.46.71

### 퍼블릭 IPv4 DNS

[🔗](#) ec2-43-200-173-233.ap-northeast-2.compute.amazonaws.com | [개방 주소법](#) [🔗](#)

### 탄력적 IP 주소

-

### AWS Compute Optimizer 찾기

📄 권장 사항을 위해 AWS Compute Optimizer에 옵트인합니다. | [자세히 알아보기](#) [🔗](#)

### Auto Scaling 그룹 이름

-



# 키페어 권한 설정하기

- 먼저 터미널 또는 Git-bash 를 이용해서 키페어 파일을 저장한 폴더로 이동
- `chmod 400 키페어이름.pem` (키페어 권한 설정)

```
Ths@DESKTOP-86MUCGC MINGW64 ~/Desktop/업무 /KDT_4th/서버
$ chmod 400 tetz.pem :
```

- Chmod → Change Mode
  - 나 | 그룹 | 전체
  - Read : 4 / write : 2 / excute : 1 의 합으로 권한 표기
  - 400 → 나한테만 읽기 권한 / 754 → 나는 읽기쓰기실행, 그룹은 읽기쓰기, 전체는 읽기만



# SSH 를 이용 서버 접속

```
Ths@DESKTOP-86MUCGC MINGW64 ~/Desktop/업무 /KDT_4th/서버
$ ssh -i tetz.pem ec2-user@13.209.97.89
The authenticity of host '13.209.97.89 (13.209.97.89)' can't be established.
ED25519 key fingerprint is SHA256:FIAY0etf1r4t43ULPHrNiQUSUF0R7KzabefcbTpNPf8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.209.97.89' (ED25519) to the list of known hosts.

  _ | _ | _ )
  _ | (   /   Amazon Linux 2 AMI
  _ | \ _ | _ |

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-34-194 ~]$ D|
```

- `ssh -i 키페어이름.pem ec2-user@퍼블릭IP주소`
- 다음 질문에서 Yes 입력



# EC2에 Node.js 설치

- [https://docs.aws.amazon.com/ko\\_kr/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html](https://docs.aws.amazon.com/ko_kr/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html)
- Nvm 을 설치
  - `curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash`
  - Nvm 활성화
  - `. ~/.nvm/nvm.sh`



```
[ec2-user@ip-172-31-34-194 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh  
/nvm/v0.34.0/install.sh | bash
```

% Total	% Received	% Xferd	Average Speed		Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left	Speed
100	13226	100	13226	0	0	42852	0	--:--:-- --:--:-- --:--:-- 42941

=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc

=> Appending bash\_completion source string to /home/ec2-user/.bashrc

=> Close and reopen your terminal to start using nvm or run the following to use it now:

```
export NVM_DIR="$HOME/.nvm"
```

```
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

```
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads  
nvm bash_completion
```

```
[ec2-user@ip-172-31-34-194 ~]$ . ~/.nvm/nvm.sh
```





# EC2에 Node.js 설치

- Nvm 으로 Node.js 16 버전 설치
  - `nvm install 16`

```
[ec2-user@ip-172-31-34-194 ~]$ nvm install 16
Downloading and installing node v16.18.1...
Downloading https://nodejs.org/dist/v16.18.1/node-v16.18.1-linux-x64.tar.xz...
##### 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v16.18.1 (npm v8.19.2)
Creating default alias: default -> 16 (-> v16.18.1)
[ec2-user@ip-172-31-34-194 ~]$
```



# EC2에 Node.js 설치

- Node 버전 확인

- `node -v`

```
[ec2-user@ip-172-31-34-194 ~]$ node -v  
v16.18.1
```



# EC2에 깃 설치

- 리눅스 명령어로 git 설치
- `sudo yum install git`

```
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-31-46-71 ~]$ sudo yum install git  
Loaded plugins: extras_suggestions, langpacks, prior
```

- 다운 로드 질문이 나오면 y 누르고 엔터

```
Total download size: 9.3 M  
Installed size: 40 M  
Is this ok [y/d/N]: y  
Downloading packages:
```



# 깃 설치 및 버전 확인 + 배포용 폴더 생성

- git --version

```
[ec2-user@ip-172-31-46-71 ~]$ git --version  
git version 2.37.1
```

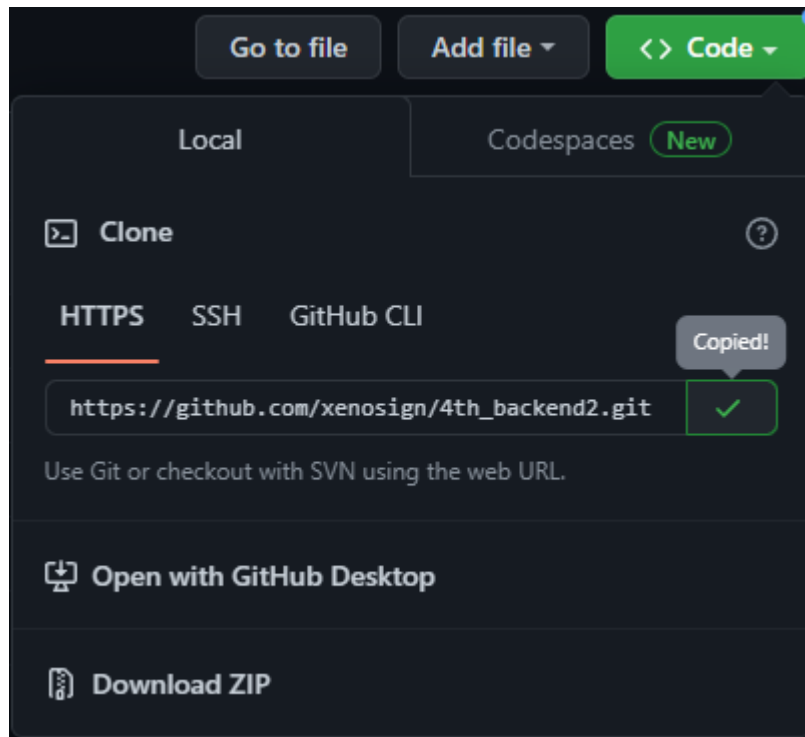
- mkdir app → cd app

```
[ec2-user@ip-172-31-46-71 ~]$ mkdir app  
[ec2-user@ip-172-31-46-71 ~]$ ls  
app  
[ec2-user@ip-172-31-46-71 ~]$ cd app  
[ec2-user@ip-172-31-46-71 app]$
```



# 배포할 git, clone 주소 복사

- 자신이 배포하려는 깃허브 Repo 에 가서 Clone 주소 복사





# Github Repo 는 퍼블릭!

- 퍼블릭이 아니면 id 와 pw 를 입력하라고 나옵니다
- 여기서 pw 는 여러분의 github 비번이 아니라 access token 입니다
- access token 발급법
  - <https://curryyou.tistory.com/344>



# 서버에 git clone 진행

- git clone 복사한 주소

```
[ec2-user@ip-172-31-46-71 app]$ git clone https://github.com/xenosign/backend.git
Cloning into 'backend'...
remote: Enumerating objects: 5017, done.
remote: Counting objects: 100% (5017/5017), done.
remote: Compressing objects: 100% (3920/3920), done.
remote: Total 5017 (delta 896), reused 4994 (delta 873), pack-reused 0
Receiving objects: 100% (5017/5017), 22.58 MiB | 16.41 MiB/s, done.
Resolving deltas: 100% (896/896), done.
[ec2-user@ip-172-31-46-71 app]$
```

- Clone 된 폴더로 이동
- Cd 폴더 명

```
[ec2-user@ip-172-31-46-71 app]$ cd backend/
[ec2-user@ip-172-31-46-71 backend]$ ls
app.js  dist  package.json  package-lock.json
[ec2-user@ip-172-31-46-71 backend]$
```



# Node module 설치

- Npm install

```
[ec2-user@ip-172-31-46-71 backend]$ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: undefined,
npm WARN EBADENGINE   required: { node: '16.x', npm: '8.11.x' },
npm WARN EBADENGINE   current: { node: 'v16.17.0', npm: '8.15.0' }
npm WARN EBADENGINE }
( [REDACTED] ) : reify:has: http fetch GET 200 https://registr
```





# 실행 테스트

- Node app.js

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ node app.js
서버는 undefined번에서 실행 중입니다!
node:events:491
  throw er; // Unhandled 'error' event
  ^

Error: connect ECONNREFUSED 127.0.0.1:3306
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1278:16)
    -----
    at Protocol._enqueue (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:144:48)
    at Protocol.handshake (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/protocol/Protocol.js:51:23)
    at Connection.connect (/home/ec2-user/app/4th_backend2/node_modules/mysql/lib/connection.js:116:18)
    at Object.<anonymous> (/home/ec2-user/app/4th_backend2/controllers/dbConnect.js:2)
    at Module._compile (node:internal/modules/cjs/loader:1155:14)
    at Object.Module._extensions..js (node:internal/modules/cjs/loader:1209:10)
    at Module.load (node:internal/modules/cjs/loader:1033:32)
    at Function.Module._load (node:internal/modules/cjs/loader:868:12)
```



# 엇!?

- 에러 메시지를 보니 서버가 undefined 에서 실행이 된다고 뜨네요? 이 건 .env 파일이 없어서 그렇겠죠?
- 그리고 아래는 mysql 서버에 접속이 안된다고 뜹니다! 이건 서버에 mysql 을 설치를 안해서 그렇습니다!
- 그럼 일단 mysql 부터 사용을 안하게 처리해 줍시다!



# 엇!?

- Mysql 코드를 전부 주석 처리 해주세요!
- 그리고 로컬에서 nodemon app.js 를 실행 시켜서 문제가 없는지 확인 합니다!

```
1 // const mysql = require('mysql');
2
3 // const connection = mysql.createConnection({
4 //   host: 'localhost',
5 //   user: 'root',
6 //   password: process.env.DB_PASSWORD,
7 //   port: '3306',
8 //   database: process.env.DB_DATABASE,
9 // });
10 // connection.connect();
11 // module.exports = connection;
12
```

```
$ nodemon app.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
서버는 4000번에서 실행 중입니다!
[]
```



# 엇!?

- 깃을 커밋하고 푸쉬해 줍니다!
- 그리고 다시 서버에서 git pull 실행!

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ git pull --all
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 497 bytes | 497.00 KiB/s, done.
From https://github.com/xenosign/4th_backend2
   3d26a1a..8f12d79  main       -> origin/main
Updating 3d26a1a..8f12d79
Fast-forward
 controllers/dbConnect.js | 22 ++++++++-----
 1 file changed, 11 insertions(+), 11 deletions(-)
[ec2-user@ip-172-31-34-194 4th_backend2]$ |
```



# 파일 질라 설치!

- .env 파일은 직접 업로드를 해줘야 하므로 파일 질라 설치
- <https://filezilla-project.org/>

## Overview

Welcome to the homepage of FileZilla®, the free FTP solution. The *FileZilla Client* not Public License.

We are also offering *FileZilla Pro*, with additional protocol support for WebDAV, Amazon

Last but not least, *FileZilla Server* is a free open source FTP and FTPS Server.

Support is available through our [forums](#), the [wiki](#) and the [bug and feature request track](#)

In addition, you will find documentation on how to compile FileZilla and nightly builds

## Quick download links

**Download  
FileZilla Client**

All platforms



**Download  
FileZilla Server**

All platforms





## Download FileZilla Client for Windows (64bit x86)

The latest stable version of FileZilla Client is 3.62.2

Please select the file appropriate for your platform below.

◆ Windows (64bit x86) 





**Download  
FileZilla Client**



This installer may include bundled offers. Check below for more options.

The 64bit versions of Windows 8.1, 10 and 11 are supported.

◆ **More download options**

Other platforms:    


Not what you are looking for?

[➔ Show additional download options](#)

## Download FileZilla Client for macOS

The latest stable version of FileZilla Client is 3.62.2

Please select the file appropriate for your platform below.





◆ macOS 

**Download  
FileZilla Client**



Requires macOS 10.13.2 or newer

◆ **More download options**

Other platforms:    

Not what you are looking for?

[➔ Show additional download options](#)



FileZilla

파일(F) 편집(E) 보기(V) 전송(T) 서버(S) 북마크(B) 도움말(H)

호스트: 사용자명(U): 비밀번호(W): 포트(P): 빠른 연결(Q)

로컬 사이트: C:\Users\lhs\ Users

- Users
  - All Users
  - Default
  - Default User
  - lhs
  - Public
- Windows
- XboxGames
- D: (새 볼륨)
- E: (Sony HDD)
- F: (LG External HDD)

리모트 사이트:

파일명	크기	파일 유형	최종 수정
..			
.config		파일 폴더	2022-09-09 오후 ...
.docker		파일 폴더	2022-03-25 오후 ...
.ssh		파일 폴더	2022-12-03 오후 ...
.vscode		파일 폴더	2022-03-24 오전 ...
3D Objects		파일 폴더	2022-03-23 오후 ...
AppData		파일 폴더	2022-03-23 오후 ...
Application Data		파일 폴더	2022-11-15 오후 ...
Contacts		파일 폴더	2022-03-23 오후 ...
Cookies		파일 폴더	2022-03-25 오전 ...
Desktop		파일 폴더	2022-11-30 오후 ...
Documents		파일 폴더	2022-11-26 오후 ...
Downloads		파일 폴더	2022-12-03 오후 ...
Favorites		파일 폴더	2022-03-23 오후 ...
Links		파일 폴더	2022-03-23 오후 ...

14 파일 및 29 디렉터리. 총 크기: 12,986,243 바이트

서버/로컬 파일    방향    리모트 파일    크기    우선 ...    상태

대기 파일    전송 실패    전송 성공

대기열: 비었음



# 사이트 관리자

항목 선택(S):

- 내 사이트
  - AWS-tetz
  - 새 사이트

새 사이트(N)

새 폴더(f)

새 북마크(M)

이름 바꾸기(R)

삭제(D)

복제(i)

일반 고급 전송 설정 문자셋

프로토콜(t): SFTP - SSH File Transfer Protocol

호스트(H): 13.209.97.89 포트(P):

로그온 유형(L): 키 파일

사용자(U): ec2-user

키 파일(K): C:\Users\Wls\Desktop\업무용 키 파일 찾기... 찾아보기...

배경색(B): 없음

비고(M):

연결(C)

확인(O)

취소





AWS-tetz - sftp://ec2-user@13.209.97.89 - FileZilla

파일(F) 편집(E) 보기(V) 전송(T) 서버(S) 북마크(B) 도움말(H)

호스트(H):

사용자명(U):

비밀번호(W):

포트(P):

빠른 연결(Q)

상태: Connected to 13.209.97.89

상태: 디렉터리 목록 조회...

상태: Listing directory /home/ec2-user

상태: "/home/ec2-user" 디렉터리 목록 조회 성공

로컬 사이트: C:\Users\lhs\

리모트 사이트: /home/ec2-user

파일명	크기	파일 유형	최종 수정
..		파일 폴더	2022-09-09 오후 ...
.config		파일 폴더	2022-03-25 오후 ...
.docker		파일 폴더	2022-12-03 오후 ...
.ssh		파일 폴더	2022-03-24 오전 ...
.vscode		파일 폴더	2022-03-23 오후 ...
3D Objects		파일 폴더	2022-03-23 오후 ...
AppData		파일 폴더	2022-03-23 오후 ...
Application Data		파일 폴더	2022-11-15 오후 ...
Contacts		파일 폴더	2022-03-23 오후 ...
Cookies		파일 폴더	2022-03-25 오전 ...
Desktop		파일 폴더	2022-11-30 오후 ...
Documents		파일 폴더	2022-11-26 오후 ...
Downloads		파일 폴더	2022-12-03 오후 ...
Favorites		파일 폴더	2022-03-23 오후 ...
Links		파일 폴더	2022-03-23 오후 ...

14 파일 및 29 디렉터리. 총 크기: 12,986,243 바이트

파일명	크기	파일 유형	최종 수정	권한	소유자/그룹
..		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
.npm		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
.nvm		파일 폴더	2022-12-03 ...	drwx-----	ec2-user ec...
.ssh		파일 폴더	2022-12-03 ...	drwxrwxr-x	ec2-user ec...
app		파일 폴더	2022-07-15 ...	-rw-r--r--	ec2-user ec...
.bash_	18	Bash Logo...	2020-07-15 ...	-rw-r--r--	ec2-user ec...
.bash_p	193	Bash Profil...	2020-07-15 ...	-rw-r--r--	ec2-user ec...
.bashrc	428	Bash RC ...	2022-12-03 ...	-rw-r--r--	ec2-user ec...

3 파일 및 4 디렉터리. 총 크기: 639 바이트

서버/로컬 파일

방향

리모트 파일

크기

우선 ...

상태

대기 파일

전송 실패

전송 성공

대기열: 비었음



리모트 사이트: /home/ec2-user/app/4th\_backend2

File Explorer View:

- /? /
  - /? home
    - ec2-user
      - /? .npm
      - /? .nvm
      - /? .ssh
      - app
        - 4th\_backend2

File List Table:

파일명	크기	파일 유형	최종 수정	권
..				
.git		파일 폴더	2022-12-03 ...	dn
.vscode		파일 폴더	2022-12-03 ...	dn
controllers		파일 폴더	2022-12-03 ...	dn
node_modules		파일 폴더	2022-12-03 ...	dn
public		파일 폴더	2022-12-03 ...	dn
routes		파일 폴더	2022-12-03 ...	dn
views		파일 폴더	2022-12-03 ...	dn
.env	70	ENV 파일	2022-12-03 ...	-rv
.eslinttrc.	288	JavaScript ...	2022-12-03 ...	-rv
.gitattributes	66	텍스트 문서	2022-12-03 ...	-rv
.gitignore	18	텍스트 문서	2022-12-03 ...	-rv

A red arrow points to the .env file in the file list.



# 실행 테스트

- Node app.js

```
[ec2-user@ip-172-31-34-194 4th_backend2]$ node app.js  
서버는 4000번에서 실행 중입니다!
```



# PM2(Process Manager 2)

- Node.js 프로그램의 프로세스 관리자
- 해당 프로세스가 죽어도 다시 살려주는 역할 이외에 많은 역할을 합니다
- 오늘은 기본 기능만 사용해 봅시다!
- PM2 설치
  - `npm i pm2 -g`



# PM2(Process Manager 2)

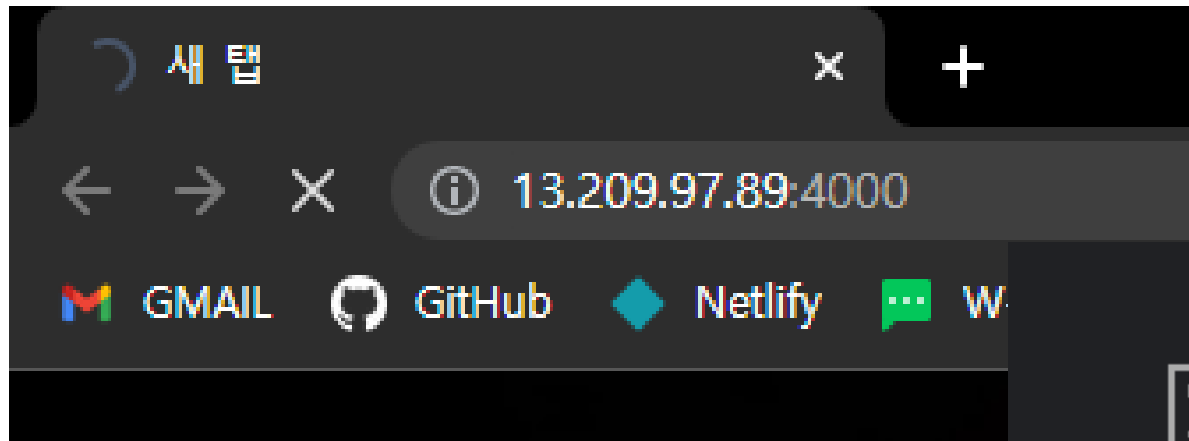
- PM2로 프로세스 실행하기
  - `pm2 start app.js`
- PM2로 프로세스 중단하기
  - `pm2 stop app.js`

```
[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ec2-user/app/backend/app.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	app	fork	0	online	0%	32.2mb







## 사이트에 연결할 수 없음

**13.209.97.89**에서 응답하는 데 시간이 너무 오래 걸립니다.

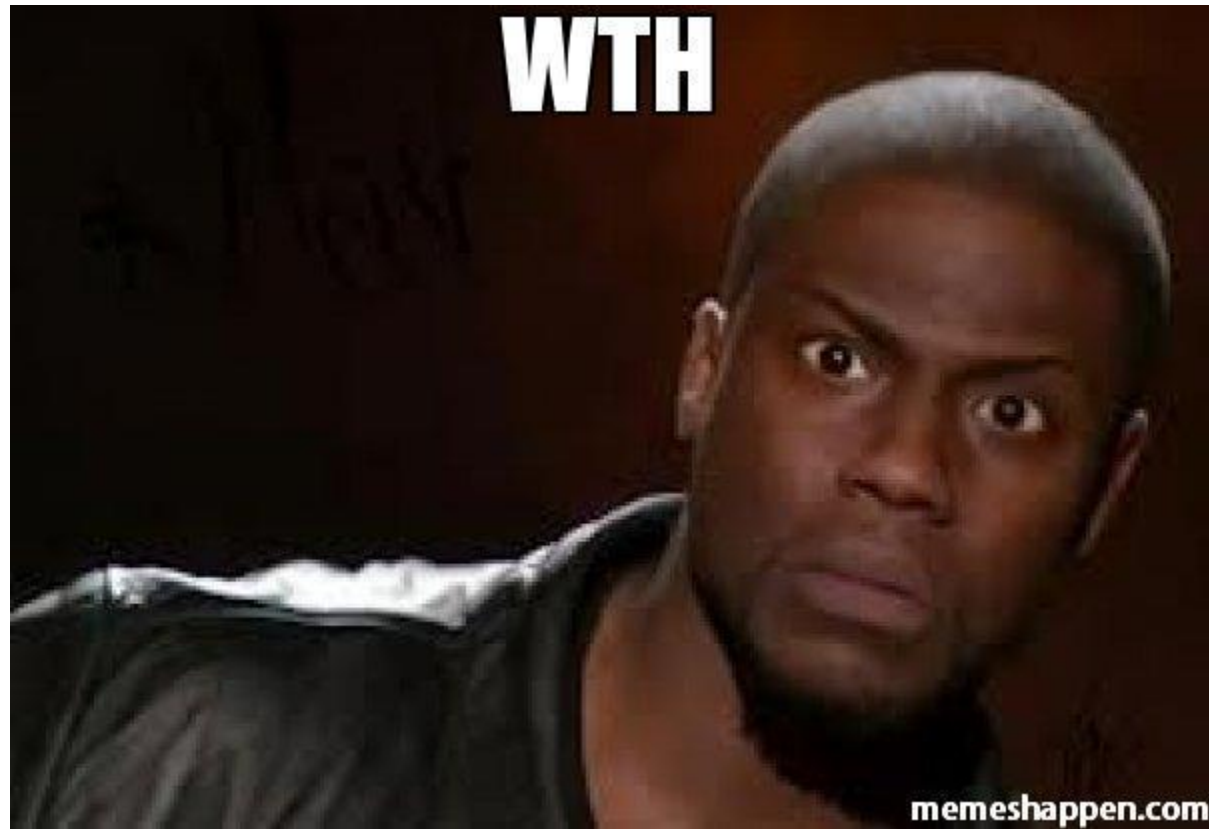
다음 방법을 시도해 보세요.

- 연결 확인
- 프록시 및 방화벽 확인
- [Windows 네트워크 진단 프로그램 실행](#)

ERR\_CONNECTION\_TIMED\_OUT

새로고침







# Port 설정이 필요합니다!

- 지금은 기본 설정이라서 port 번호 22번만 열려 있습니다!
- 직접 작성한 Port 번호를 보안 그룹에 추가하여 접속이 가능하게 만들어 봅시다!

세부 정보 | **보안** | 네트워킹 | 스토리지 | 상태 검사 | 모니터링 | 태그

▼ 보안 세부 정보

IAM 역할  
-

소유자 ID  
599697610402

보안 그룹  
sg-01b2877a9f373e986 (launch-wizard-1)

▼ 인바운드 규칙

Q 필터 규칙

보안 그룹 규칙 ID	포트 범위	프로토콜	원본	보안 그룹
sgr-04809f9db7715d339	22	TCP	0.0.0.0/0	launch-wizard-1



인바운드 규칙

아웃바운드 규칙

태그

이제 Reachability Analyzer를 사용하여 네트워크 연결을 확인할 수 있습니다.

Reachability Analyzer 실행



인바운드 규칙 (1/1)

보안 그룹 규칙 필터



태그 관리

인바운드 규칙 편집



1



<input checked="" type="checkbox"/>	Name ▾	보안 그룹 규칙 ID ▾	IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾	소스 ▾	설명
<input checked="" type="checkbox"/>	-	sgr-04809f9db7715d3...	IPv4	SSH	TCP	22	0.0.0.0/0	-



# 인바운드 규칙 편집 정보

인바운드 규칙은 인스턴스에 도달하도록 허용된 수신 트래픽을 제어합니다.

## 인바운드 규칙 정보

보안 그룹 규칙 ID	유형 <small>정보</small>	프로토콜 <small>정보</small>	포트 범위 <small>정보</small>	소스 <small>정보</small>	설명 - 선택 사항 <small>정보</small>	
sgr-04809f9db7715d339	SSH	TCP	22	사용자 지정	<input type="text" value="Q"/>	<input type="text"/>
				<input type="text" value="0.0.0.0/0"/>	<input type="button" value="삭제"/>	



## 인바운드 규칙 정보

보안 그룹 규칙 ID      유형 정보      프로토콜 정보      포트 범위 정보      소스 정보      설명 - 선택 사항 정보

sgr-04809f9db7715d339

SSH ▼

TCP

22

사용자 지정 ▼

Q

삭제

0.0.0.0/0 X

-

사용자 지정 TCP ▼

TCP

4000

사용자 지정 ▼

Q |

삭제

CIDR 블록

0.0.0.0/0

0.0.0.0/8

0.0.0.0/16

0.0.0.0/24

규칙 추가

취소

변경 사항 미리 보기

규칙 저장



인바운드 규칙 (2)

태그 관리

인바운드 규칙 편집

보안 그룹 규칙 필터

<input type="checkbox"/>	Name ▾	보안 그룹 규칙 ID ▾	IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾	소스 ▾	설명
<input type="checkbox"/>	-	sgr-02c5b04a937f12212	IPv4	사용자 지정 TCP	TCP	4000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-04809f9db7715d3...	IPv4	SSH	TCP	22	0.0.0.0/0	-



# Welcome to Tetz Express Service!

[로그인 바로가기](#)

[회원가입 바로가기](#)

[게시판 서비스](#)

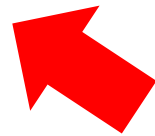


# Welcome to Tetz Express Service!

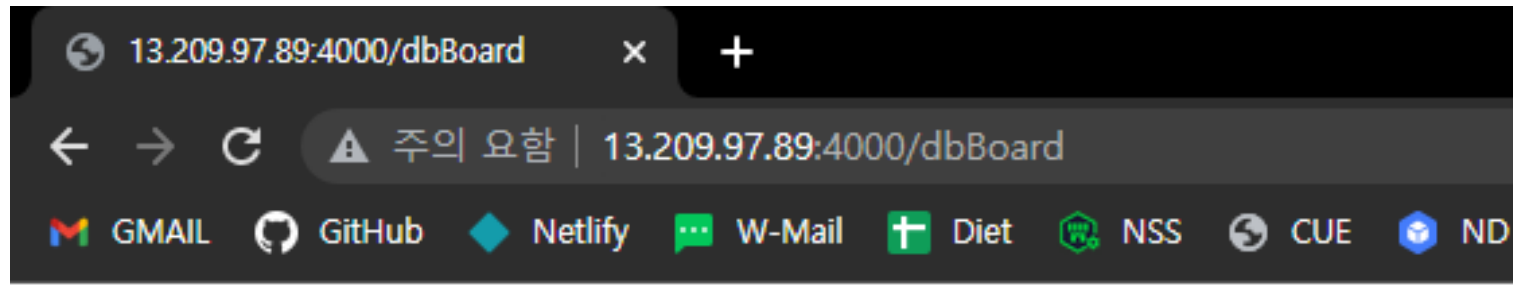
[로그인 바로가기](#)

[회원가입 바로가기](#)

[게시판 서비스](#)







로그인이 필요한 서비스 입니다.

[로그인 페이지로 이동](#)

## 로그인

아이디

11

비밀번호

..

로그인



## Tetz Board

현재 등록 글 : 2

글쓰기

로그아웃

작성자 : 11

11

11

수정

삭제

작성자 : 11

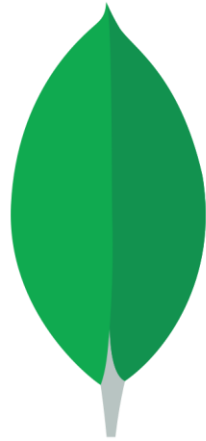
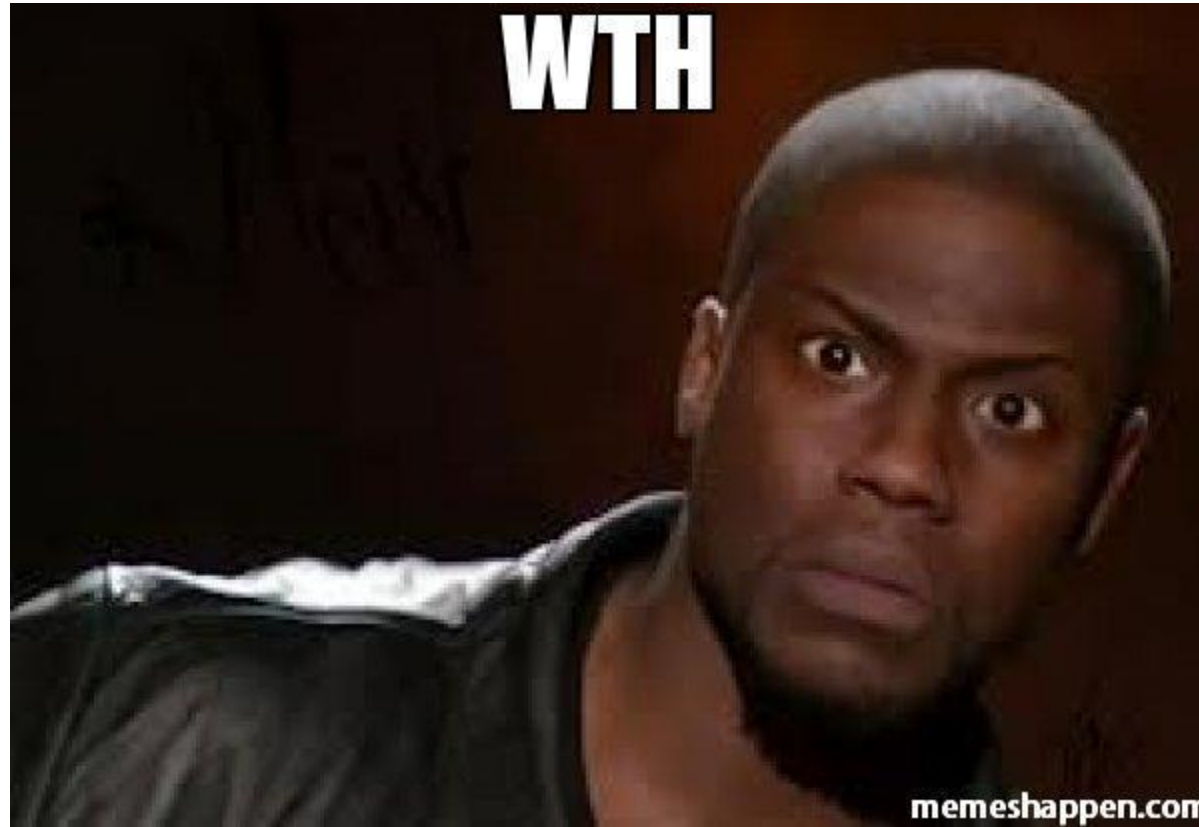
11

11

수정

삭제





mongoDB®



We are deploying your changes (current action: configuring MongoDB)

효석'S ORG - 2022-09-06 > PROJECT 0

# Network Access

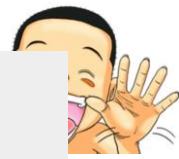
IP Access List   Peering   Private Endpoint

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
222.235.102.213/32 (includes your current IP address)		● Active	<button>⚙ EDIT</button> <button>🗑 DELETE</button>
54.177.245.233/32		🕒 Pending	<button>⚙ EDIT</button> <button>🗑 DELETE</button>



+ ADD IP ADDRESS



# Tetz Board

현재 등록 글 : 2

글쓰기

dasdadsa

dasdada

수정

삭제

21

21

수정

삭제



