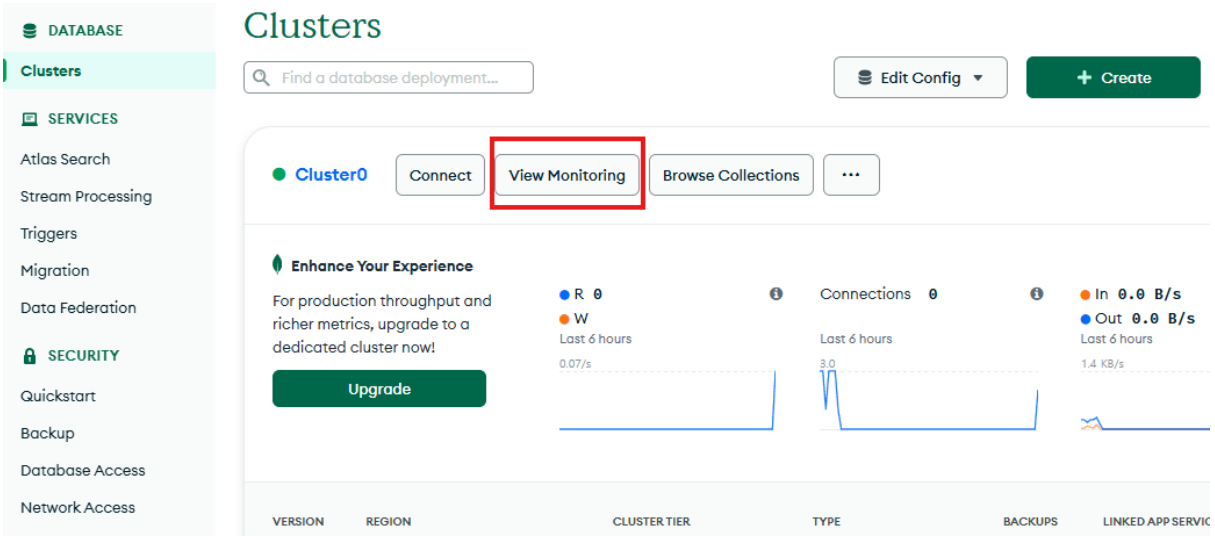
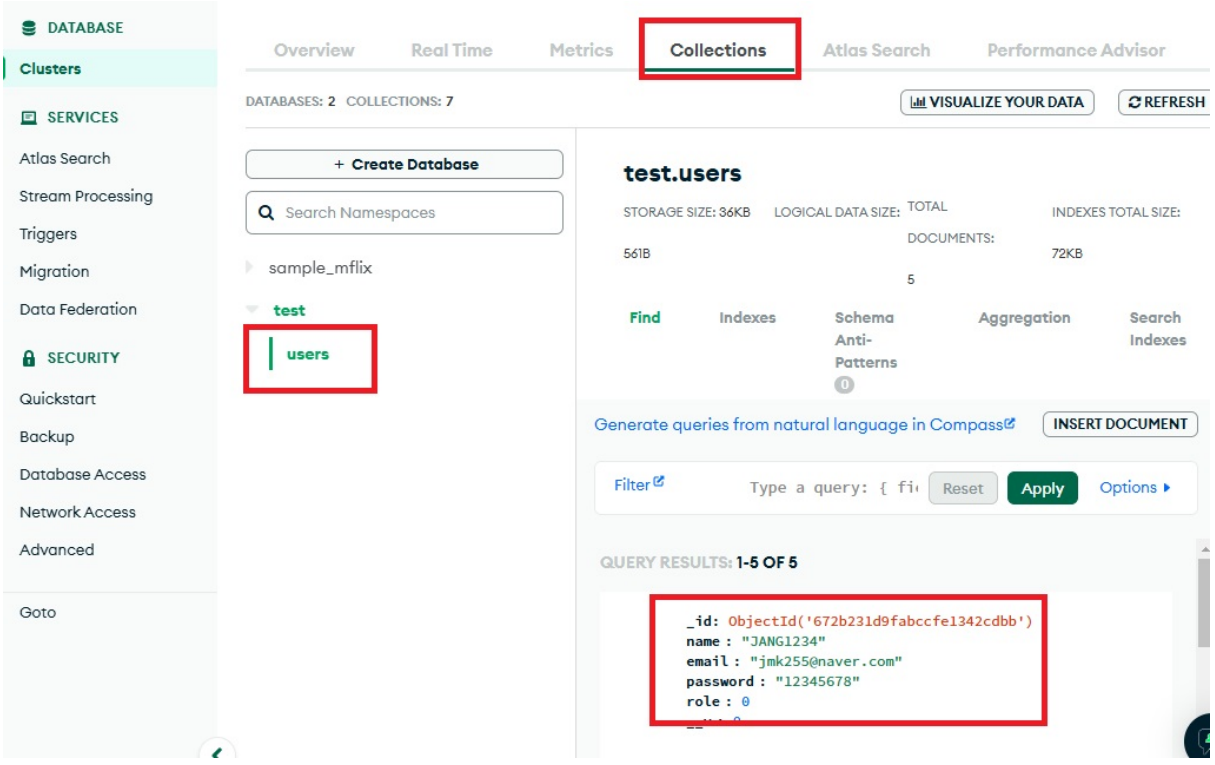
Bcrypt로 비밀번호 암호화

지난번에 postman으로 작성한 회원가입 정보를 MongoDB에서 확인해봅시다.





password부분을 보면 작성한 그대로 비밀번호가 보이는 것을 확인 할 수 있는데,

이렇게 비밀번호가 그대로 노출되게 되면 유출의 위험이 있어 보안성이 떨어집니다.

비밀번호 암호화

Bcrypt설치

```
npm install bcrypt --save
```

https://www.npmjs.com/package/bcrypt → bcrypt-npm 사이트

Usage

async (recommended)

```
const bcrypt = require('bcrypt');
const saltRounds = 10;
const myPlaintextPassword = 's0/\/\P4$$w0rD';
const someOtherPlaintextPassword = 'not_bacon';
```

To hash a password:

Technique 1 (generate a salt and hash on separate function calls):

```
bcrypt.genSalt(saltRounds, function(err, salt) {
    bcrypt.hash(myPlaintextPassword, salt, function(err, hash) {
        // Store hash in your password DB.
    });
});
```

Usage 부분을 보면 salt를 생성하고

아래 Technique 1에 비밀번호를 암호화 시키는 방법이 나와 있음.

salt = 유저가 보낸 비밀번호 데이터 외에 추가적으로 무작위 데이터를 더해서 해시값을 계산하는 방법(요리에 소금을 뿌리듯 추가하는 것)

saltRounds = 기존에 salt를 적용해서 나온 결과에 다시 동일한 salt를 적용해서 해시를 도출하는 그 횟수를 말한다.

index.js에서 정보를 user모델에 넣어주고, save하기전에 비밀번호 암호화를 해줘야 한다.

models > User.js

```
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');
const saltRounds = 10;

const userSchema = mongoose.Schema({
  name: {
    type: String,
    maxlength: 50
  },
  email: {
    type: String,
    trim: true, //띄어쓰기(빈칸)을 제거하는 역할
    unique: 1
  },
  password: {
    type: String,
    minlength: 5
  },
  role: { // 예) 넘버가 1이면 관리자고 넘버가 0이면 일반유저
    type: Number,
    default: 0
  },
  image: String,
  token: { // 토큰을 이용해 나중에 유효성 관리를 할 수 있음
    type: String
  },
  tokenExp: { //토큰을 사용할 수 있는 기간
    type: Number
```

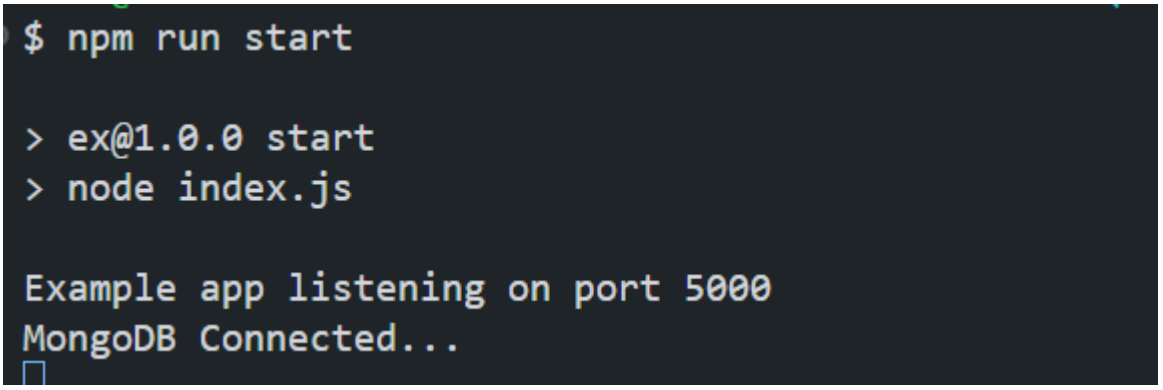
```
  }
})

//mongoose의 pre()메서드를 활용 'save'메서드가 호출되기전에 콜백함수가 실행
userSchema.pre('save', function(next){ //인자로 전달된 next는 pre메서드가 실행된 후 다시 save메서드가 호출된 위치로 실행을 넘기기 위해서 필요

    //this = userSchema
    const user = this;

    //password가 변경될 때만 비밀번호를 암호화 해주는 코드
  if(user.isModified('password')){
      //비밀번호 암호화
        bcrypt.genSalt(saltRounds, function(err, salt) {

            //에러가 발생하면 함수를 종료하고 next메서드로 err를 전달
            if(err) return next(err)

            //salt를 제대로 생성을 했다면 비밀번호를 해싱합니다.
            //hash의 첫 번째 인자 = 사용자가 입력한 비밀번호
            //두 번째 인자 = 생성한 salt
            //세 번째 인자 = 콜백함수
        bcrypt.hash(user.password, salt, function(err, hash) {
            if(err) return next(err)

            //password를 hash(암호화된 비밀번호)로 변경
          user.password = hash

          //next()안하면 save메서드가 호출된 위치로 넘어가지 않음
          next();
        });
      });
  } else {
      //비밀번호가 변경되지 않으면 암호화 해주는 코드를 실행하지 않고 n
```

```
ext로 save가 호출된 위치로 빠져나감
    next();
  }
})

const User = mongoose.model('User', userSchema);

module.exports = { User }
```

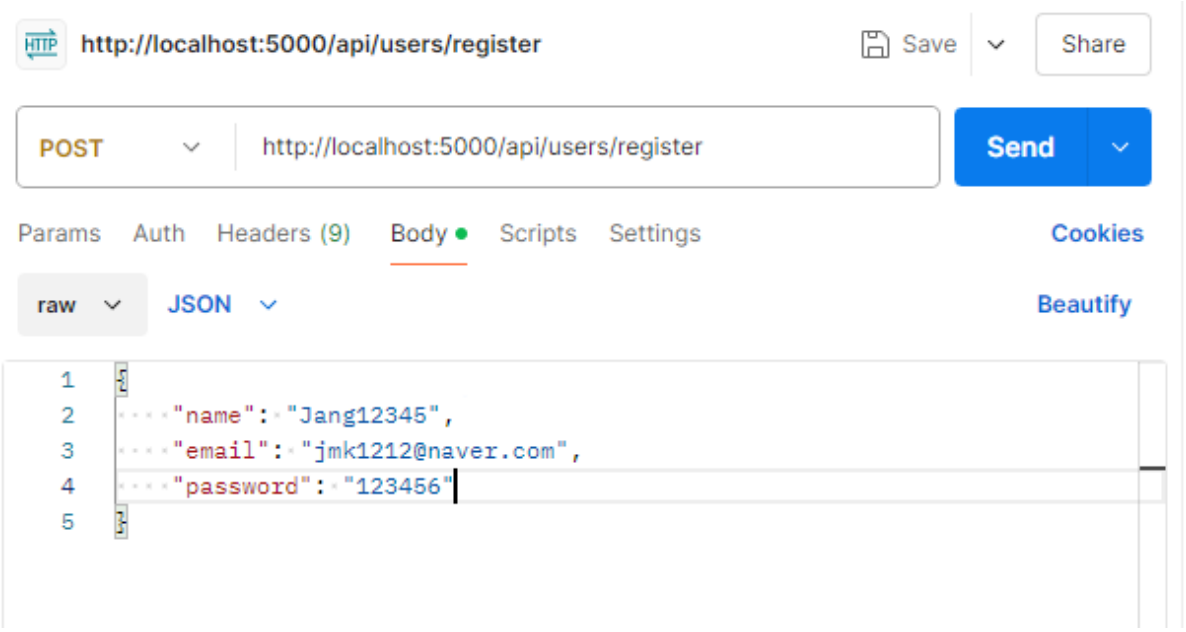
해싱(hasing) = `123456`과 같은 `plain text`를 특정 알고리즘(Hash Function)을 통해 **인간이 해독하지 못하는 문자열로 변형**해주는 것

다 작성 했으면 다시 회원가입을 해 봅시다.

서버를 실행하고



postman을 활용하여 이메일만 변경하고 send버튼 클릭



MongoDB확인 = password가 암호화 된 것을 확인 할 수 있음