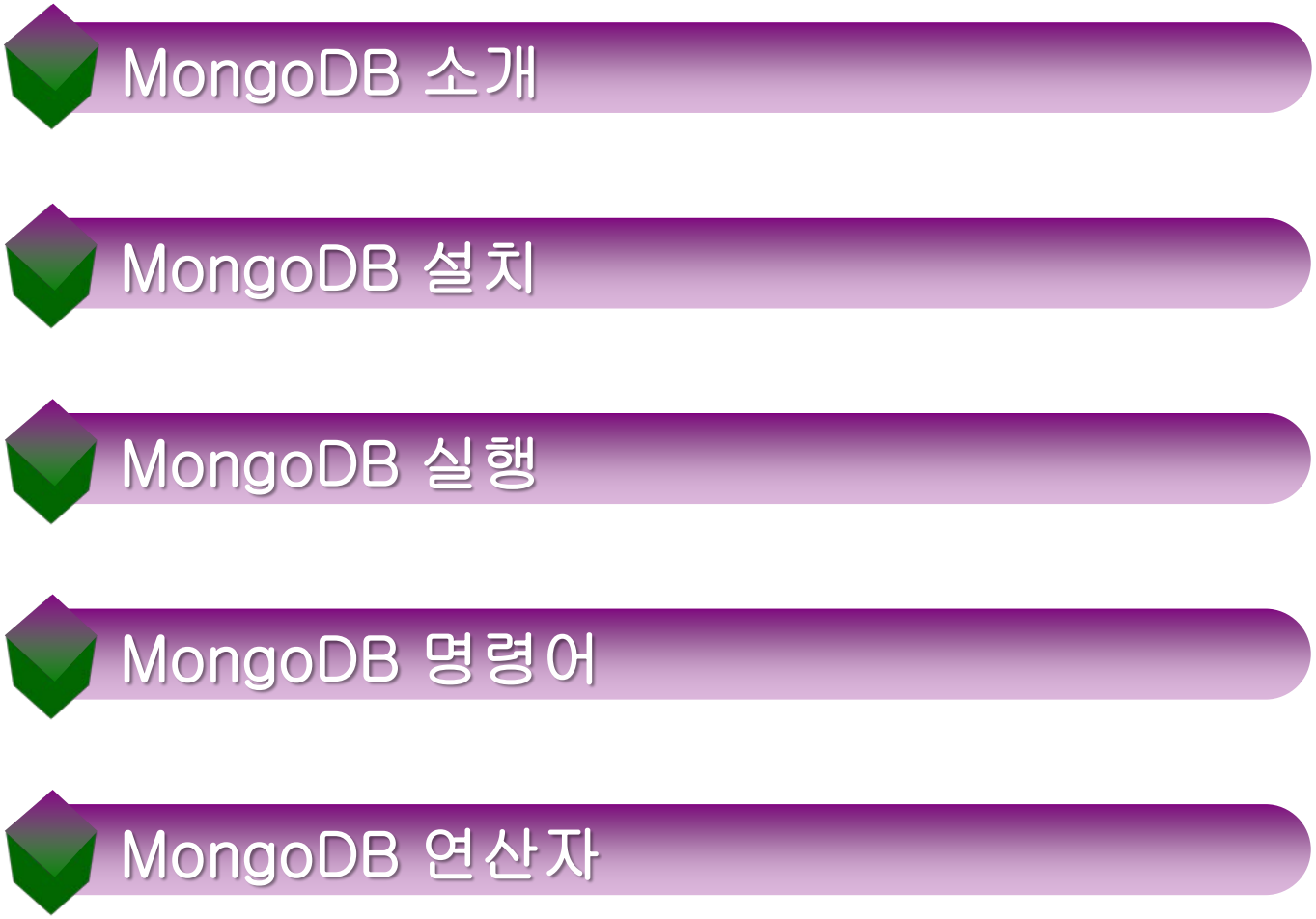
MongoDB 실습

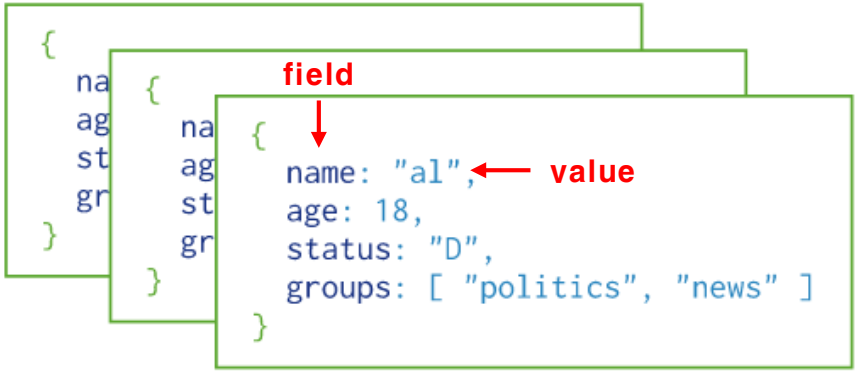
Database Programming

실습 순서



MongoDB 소개

- MongoDB
  - 크로스 플랫폼 문서를 지향하는 NoSQL 데이터베이스 시스템
  - 데이터를 필드(field)와 값(value)의 쌍으로 구성된 JSON 객체와 유사한 구조인 BSON(Binary JSON) 문서로 저장함
  - 문서(데이터)를 컬렉션에 저장하고 데이터베이스에 하나 이상의 컬렉션을 저장함
  - 무료 오픈 소스 소프트웨어

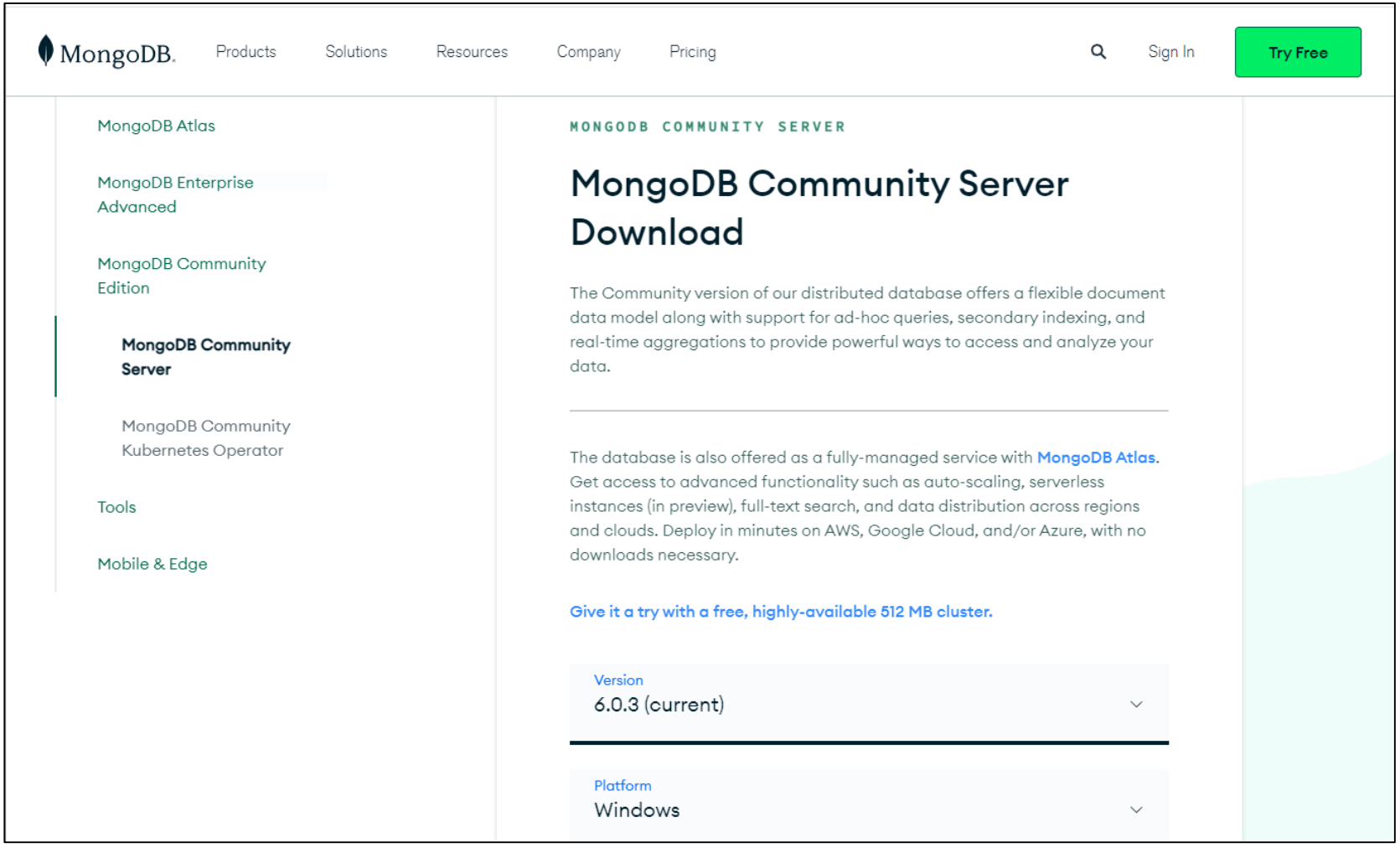


[그림] Collection 구조[1]

[1] https://docs.mongodb.com/manual/core/databases-and-collections/

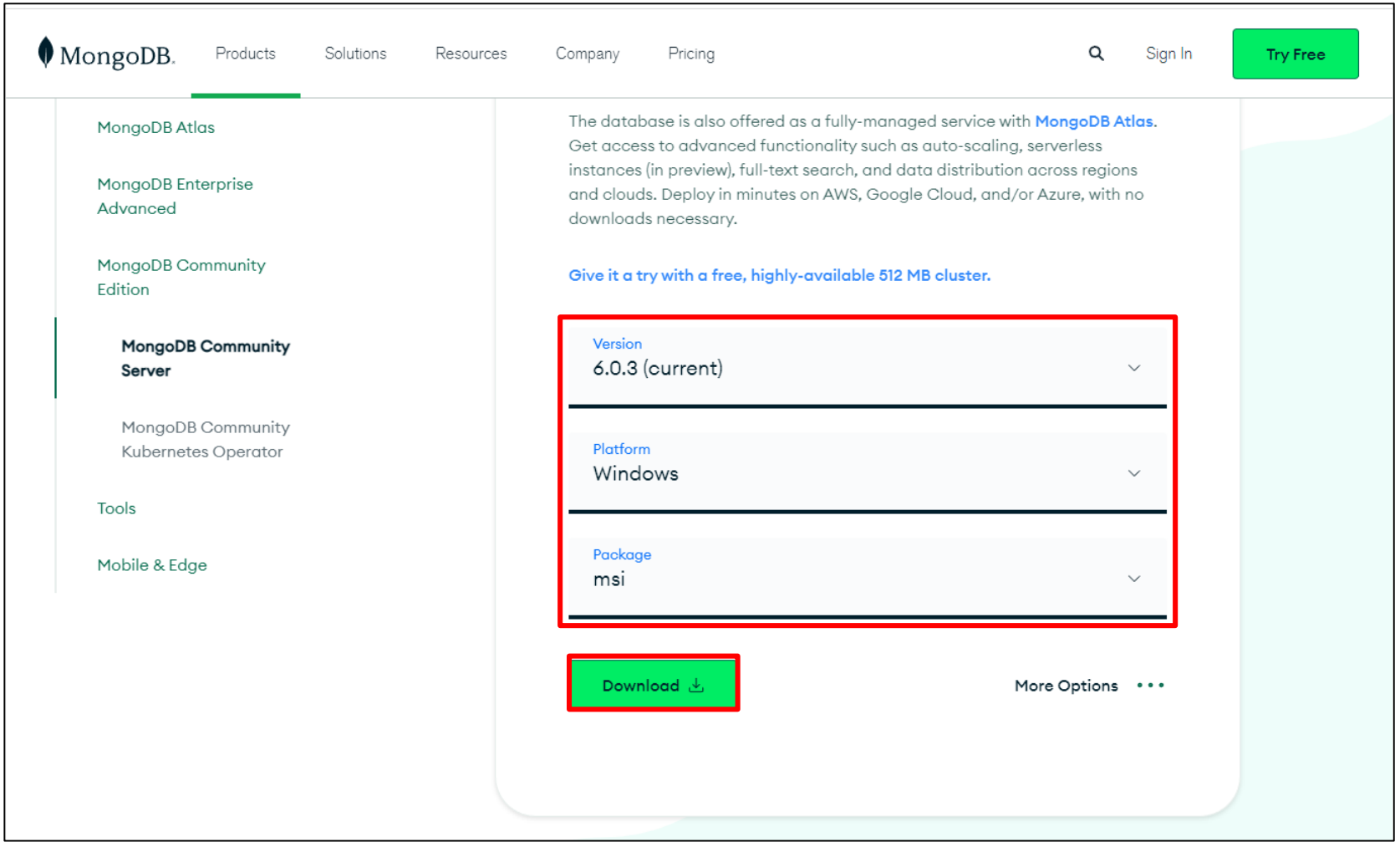
MongoDB 설치(1/10)

1) https://www.mongodb.com/try/download/community 접속



MongoDB 설치(2/10)

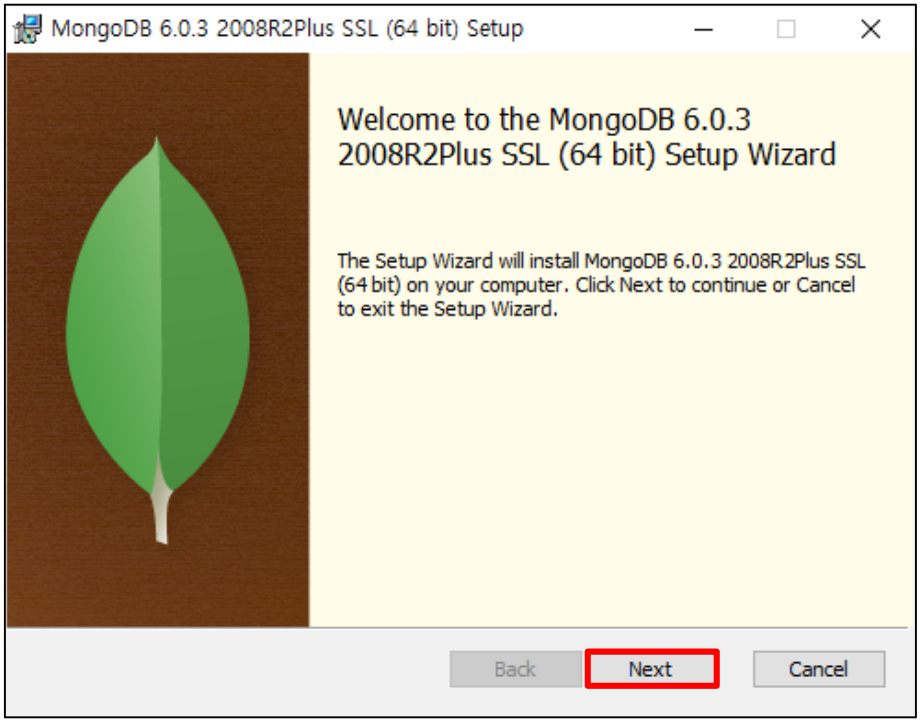
2) "6.0.3" 버전, 자신의 운영체제에 맞는 플랫폼, "msi" 패키지 선택 후 다운로드 클릭

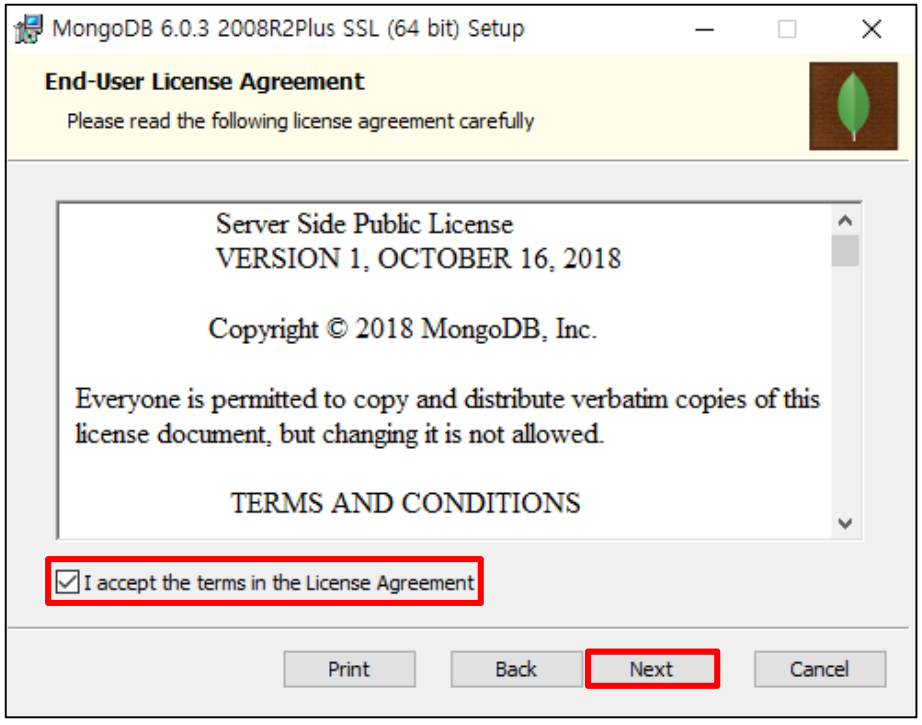


MongoDB 설치(3/10)

3) 다운로드 받은 설치 파일 실행 → Next 클릭

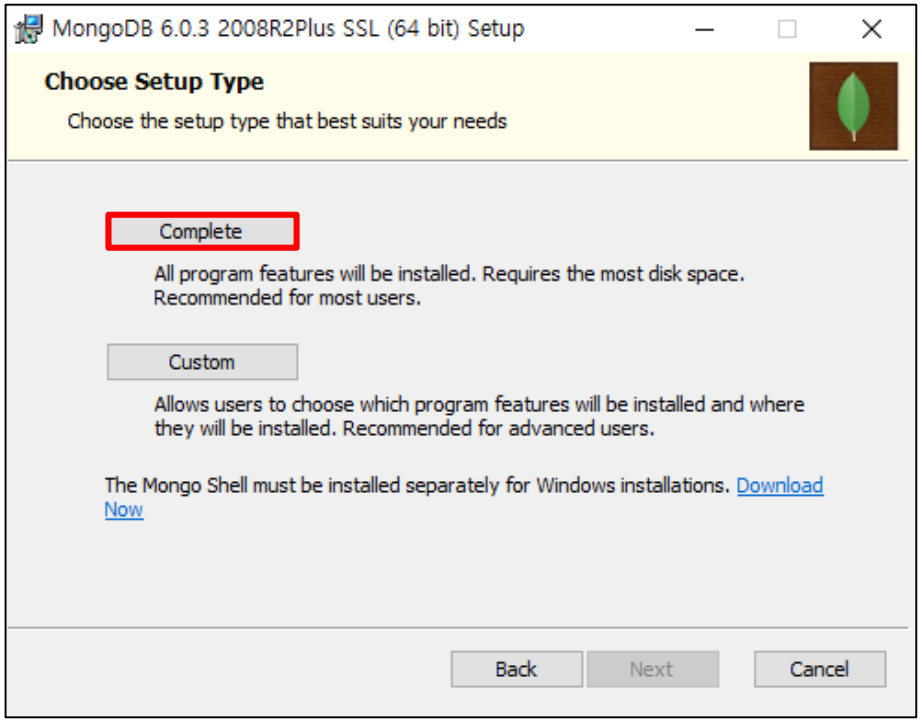
4) 라이센스 동의 → Next 클릭

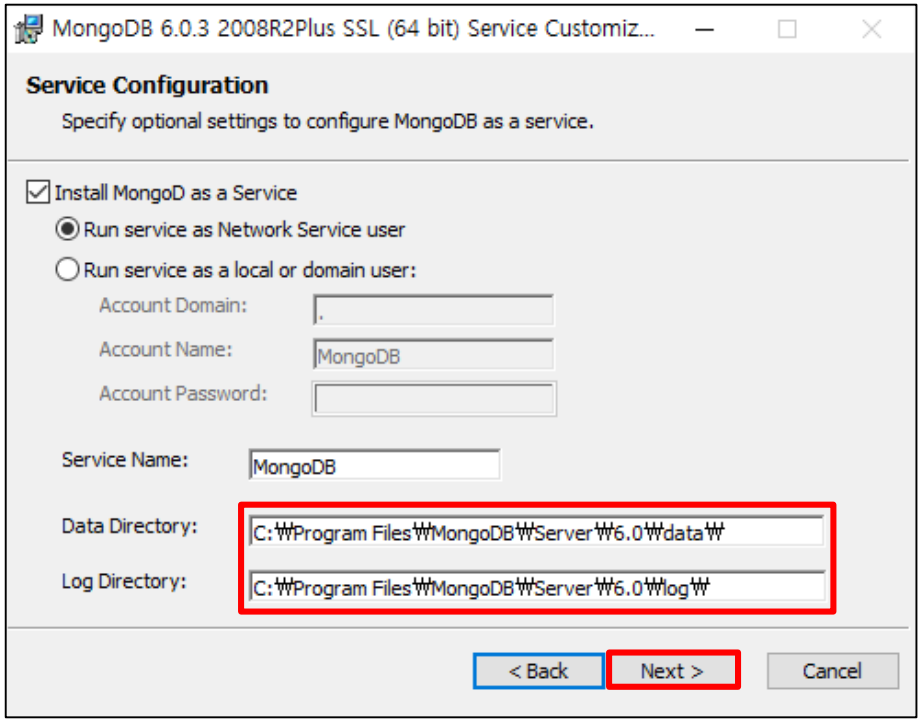




MongoDB 설치(4/10)

5) Complete 클릭
6) 설치 경로 확인 → Next 클릭

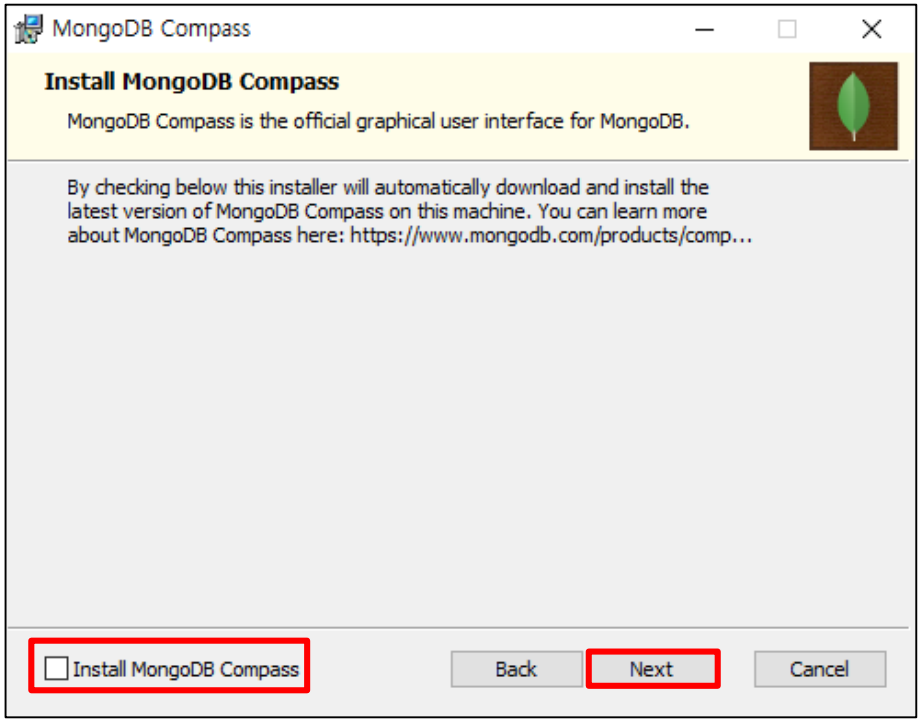


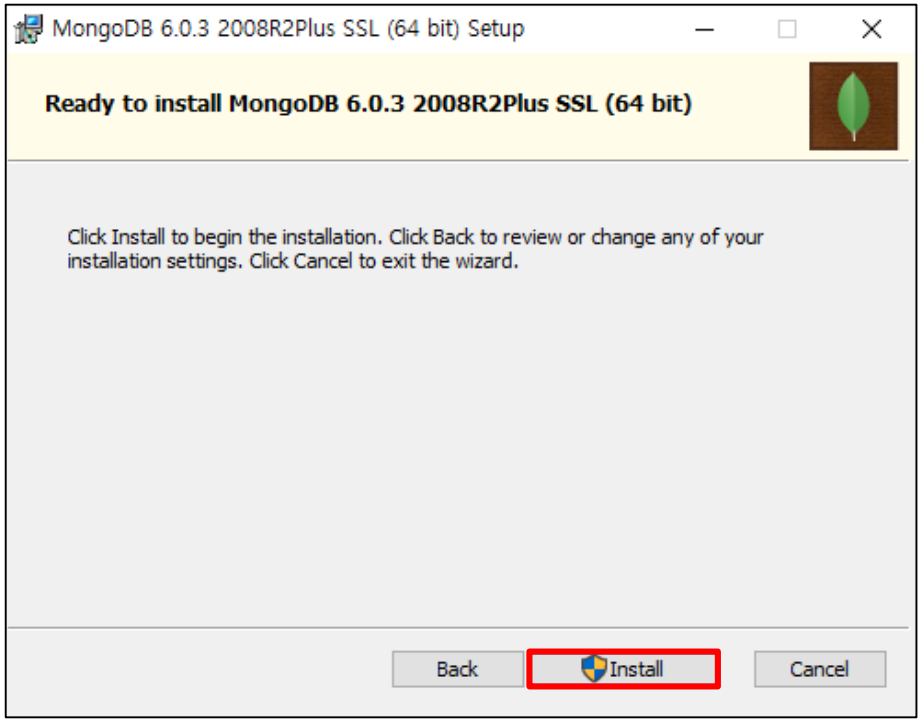


MongoDB 설치(5/10)

7) Install MongoDB Compass 체크 해제 → Next 클릭

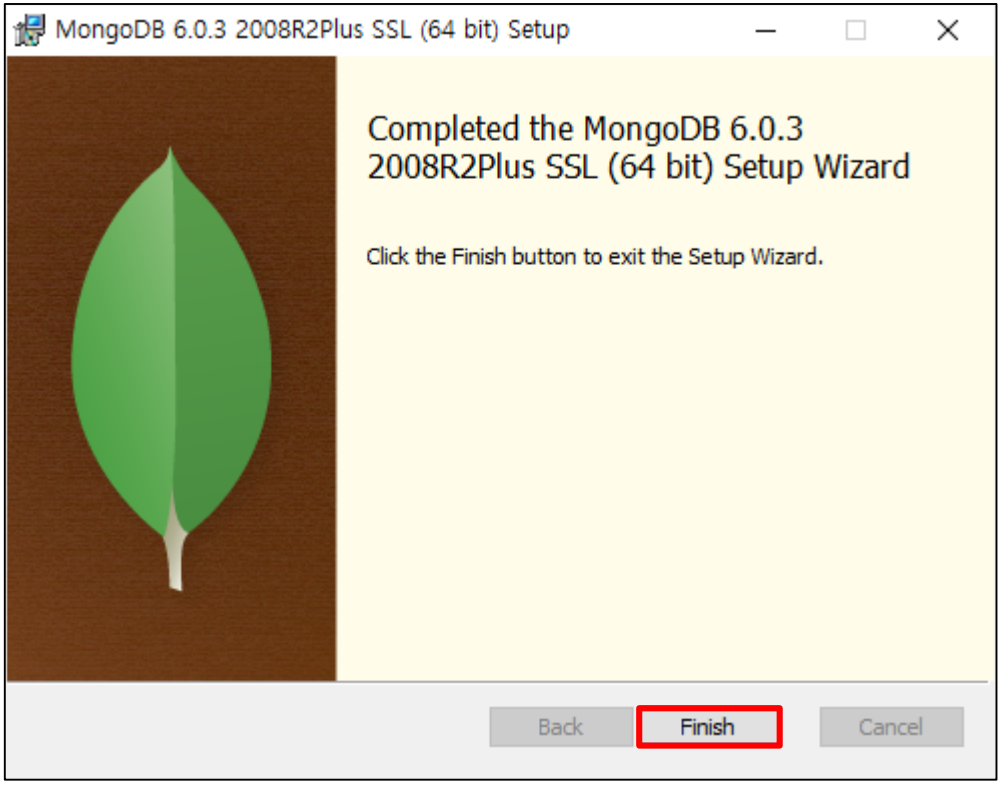
8) Install 클릭





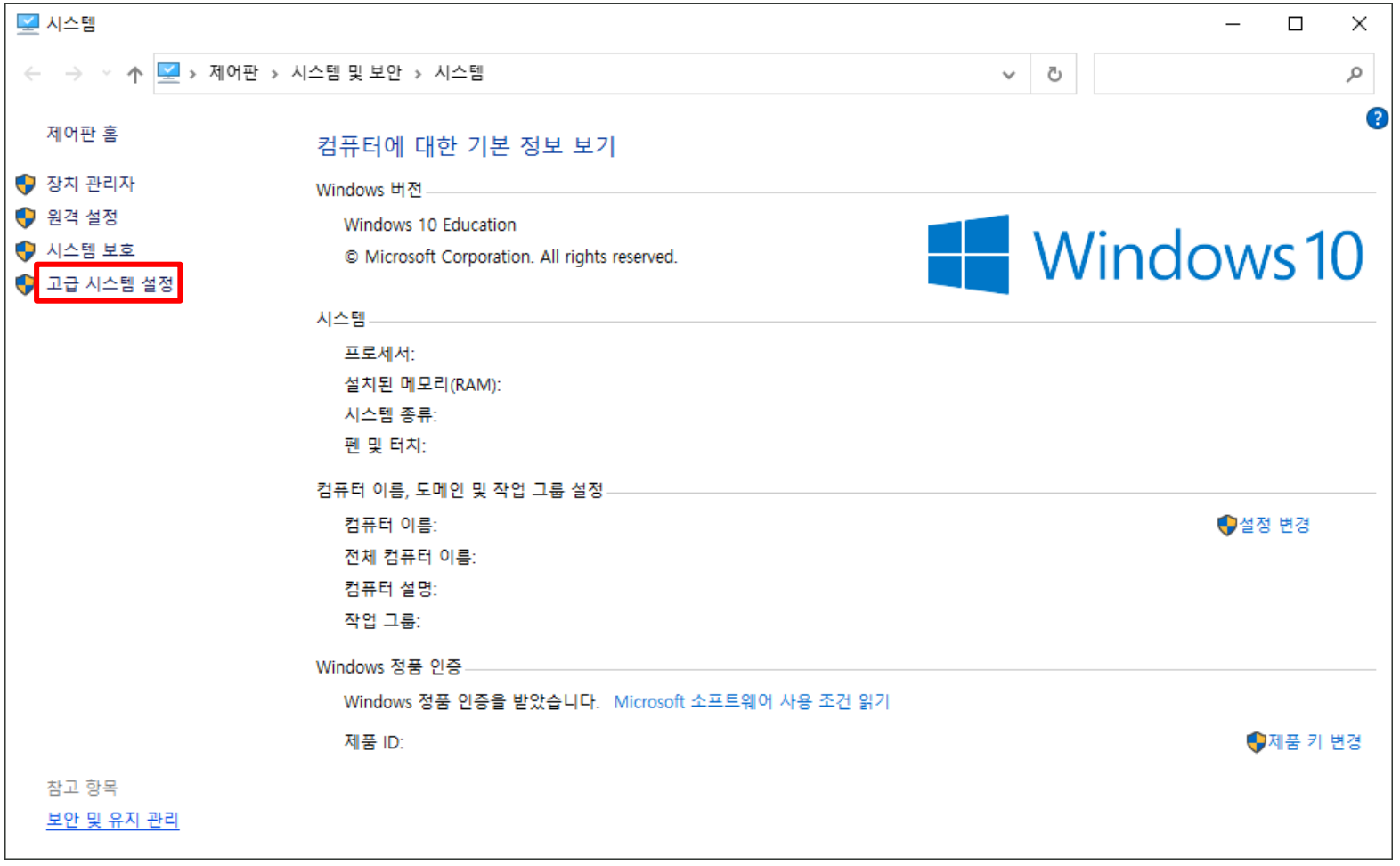
MongoDB 설치(6/10)

9) Finish 클릭



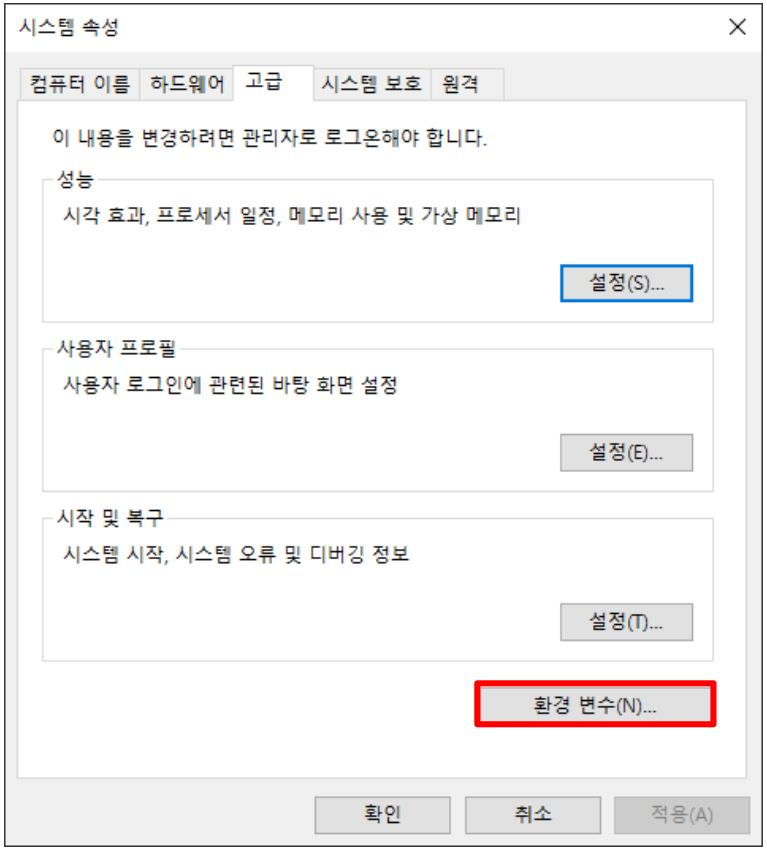
MongoDB 설치(7/10)

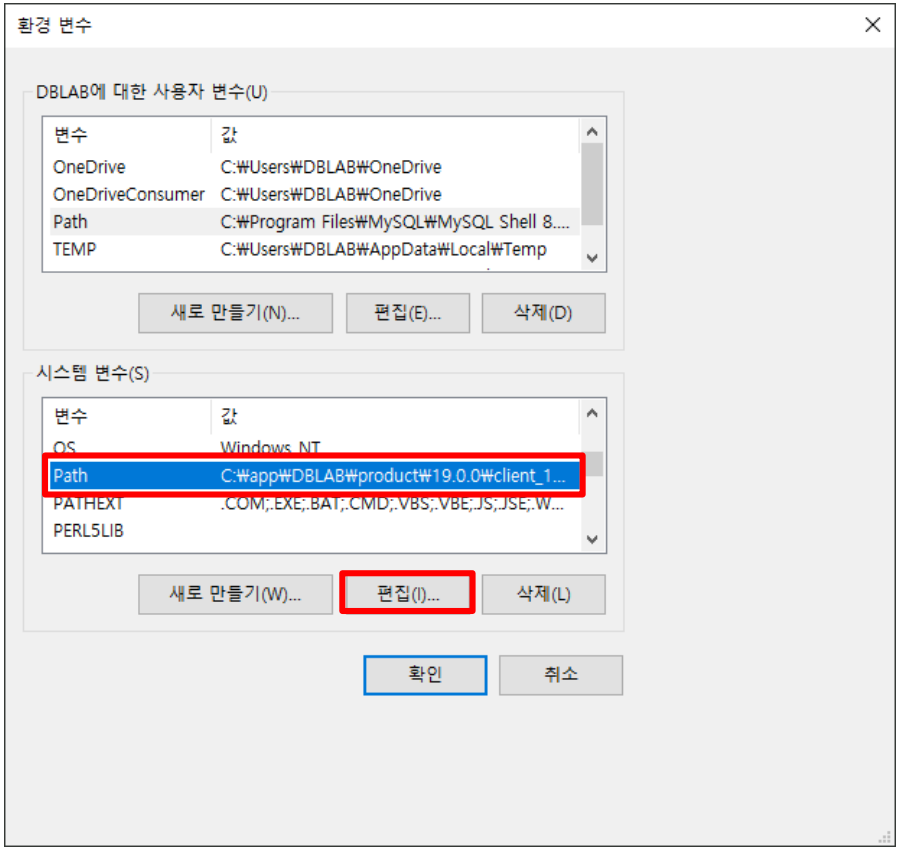
10) 내 PC 우클릭 → 속성 → 고급 시스템 설정 클릭



MongoDB 설치(8/10)

11) 환경 변수 클릭
12) 시스템 변수 → Path 선택 → 편집 클릭

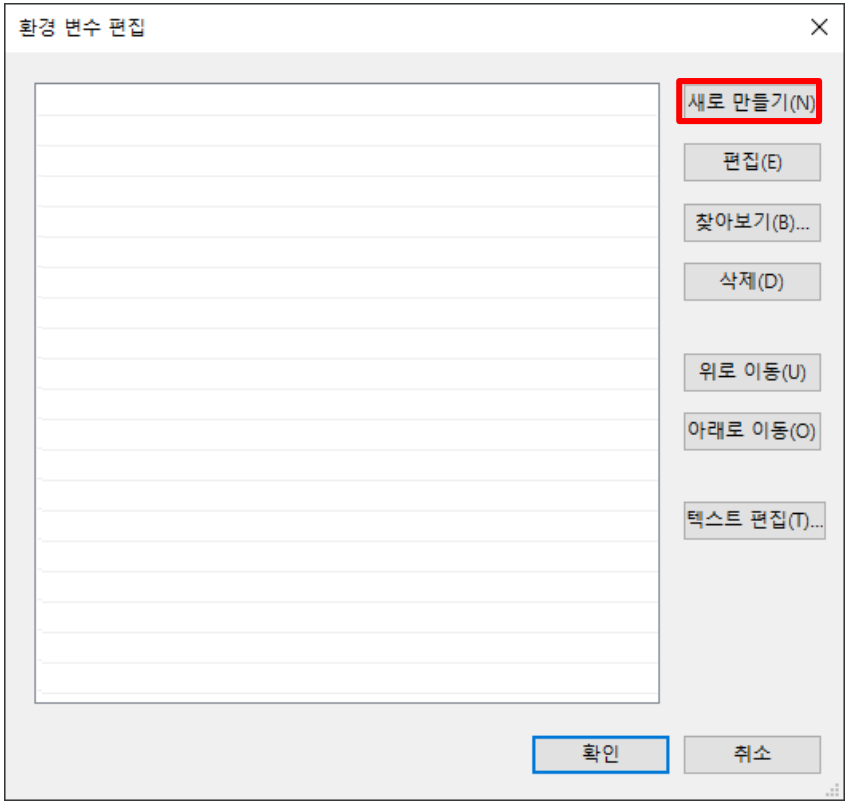


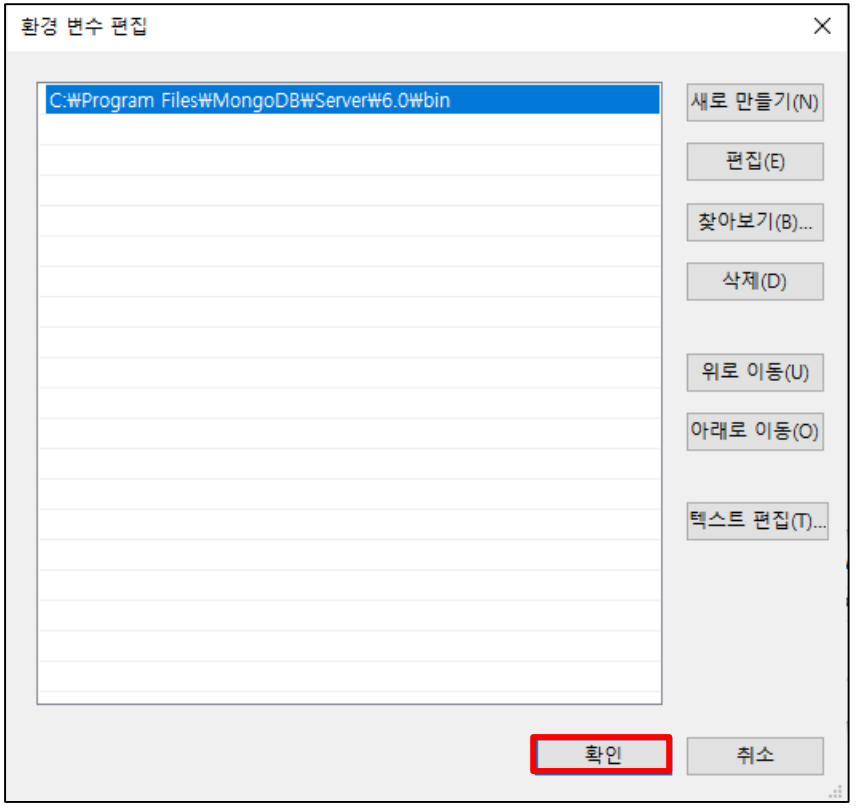


MongoDB 설치(9/10)

13) 새로 만들기 클릭

14) C:\Program Files\MongoDB\Server\6.0\bin 입력 → 확인 클릭





MongoDB 설치(10/10)

15) cmd 실행 → mongod --version 입력하여 설치 확인

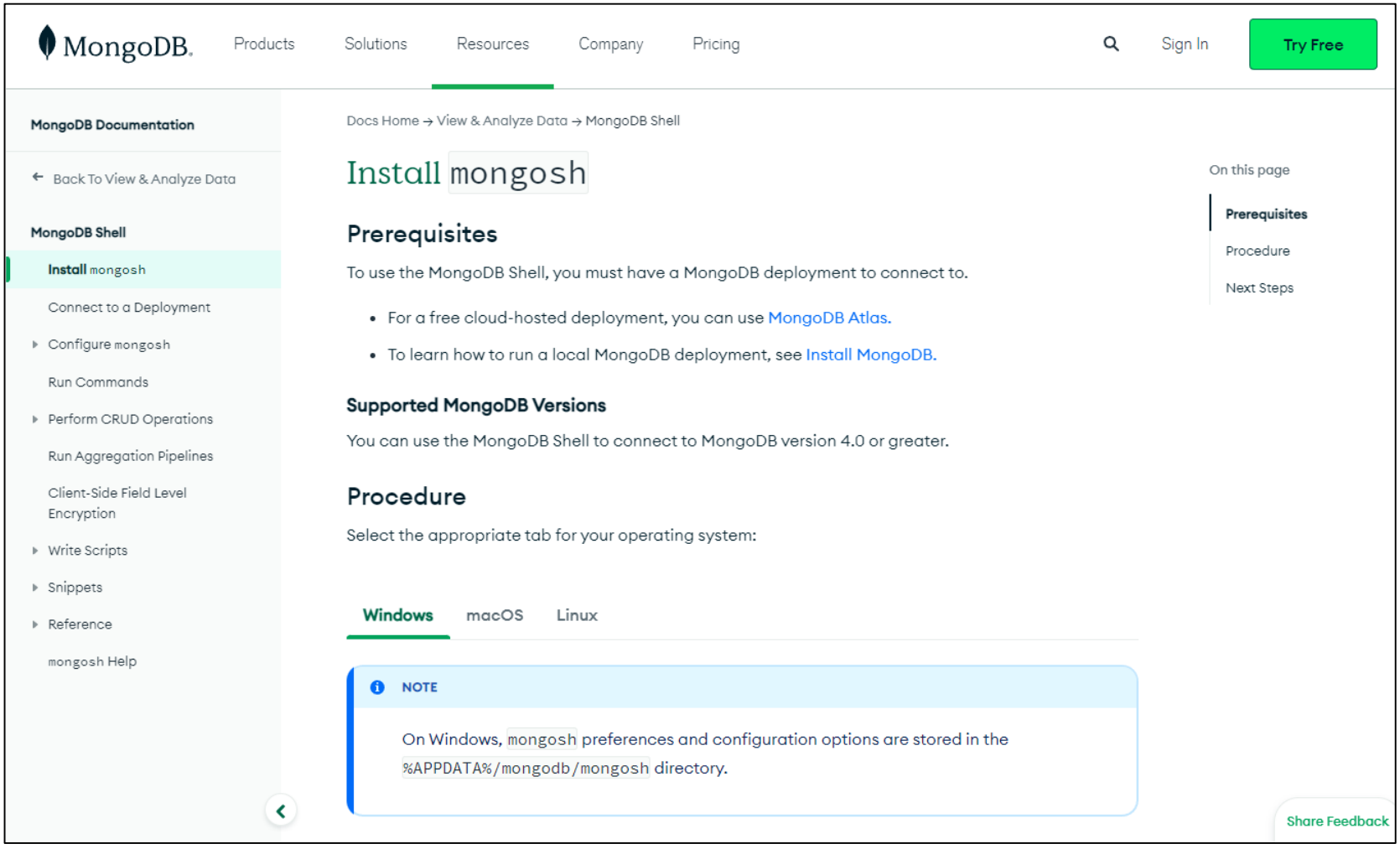
명령 프롬프트

```
C:₩Users₩DBLAB>mongod --version
db version v6.0.3
Build Info: {
    "version": "6.0.3",
    "gitVersion": "f803681c3ae19817d31958965850193de067c516",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}

C:₩Users₩DBLAB>
```

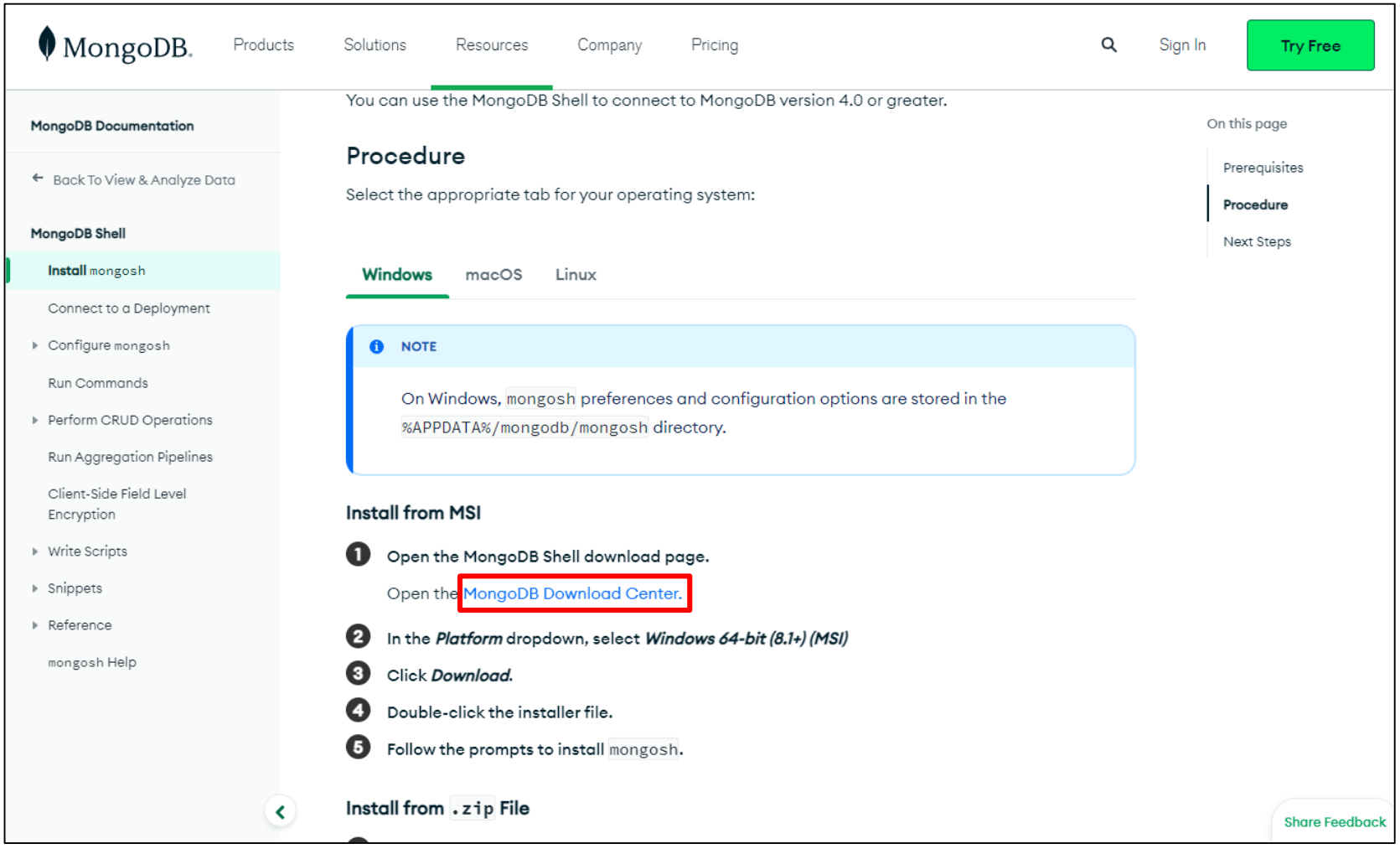
MongoDB Shell 설치(1/5)

1) https://docs.mongodb.com/mongodb-shell/install/ 접속



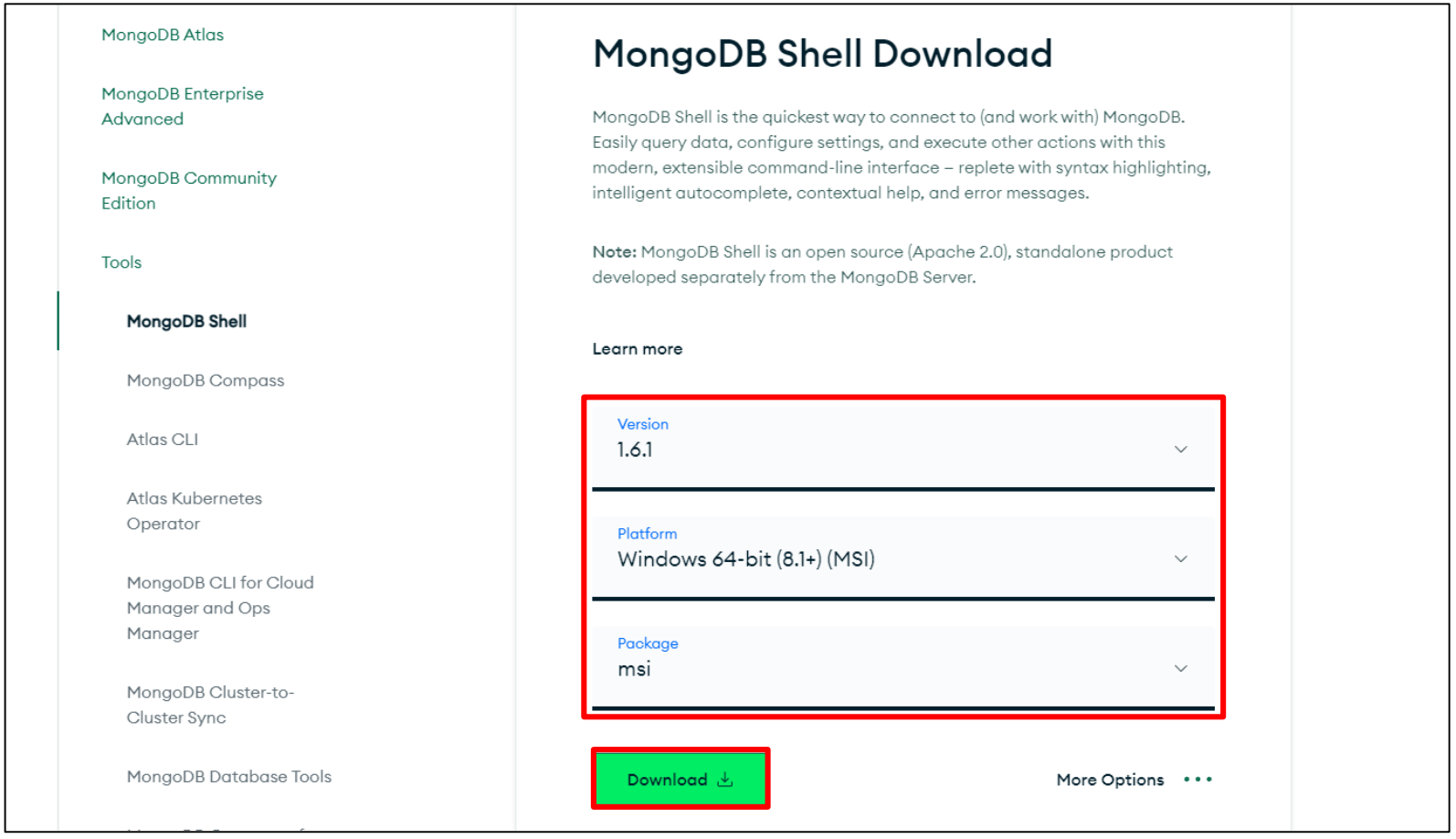
MongoDB Shell 설치(2/5)

2) "MongoDB Download Center" 클릭



MongoDB Shell 설치(3/5)

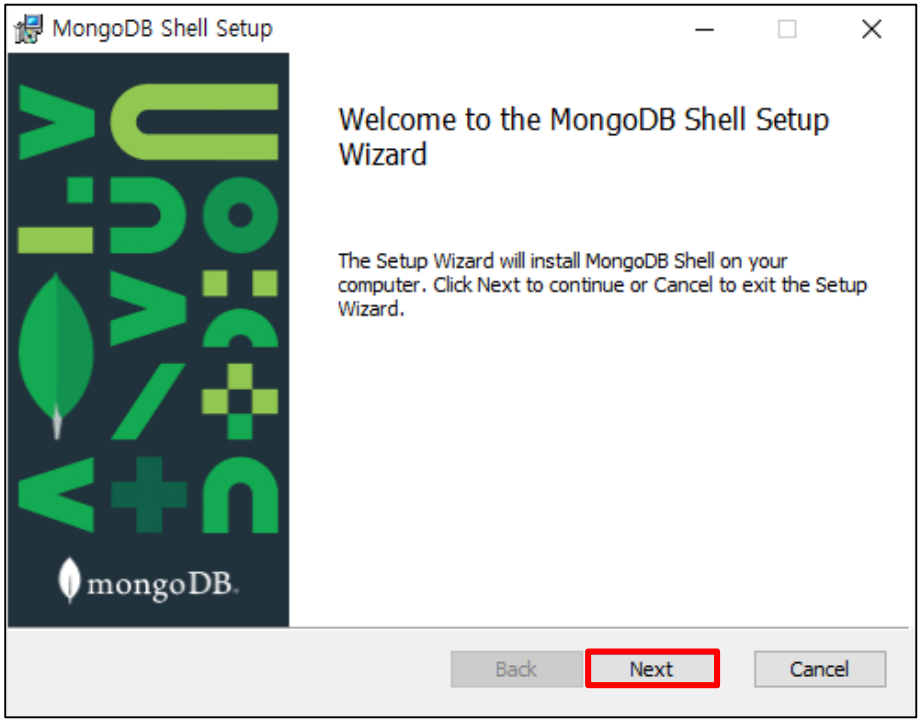
3) "1.6.1" 버전, 자신의 운영체제에 맞는 플랫폼, "msi" 패키지 선택 후 다운로드 클릭

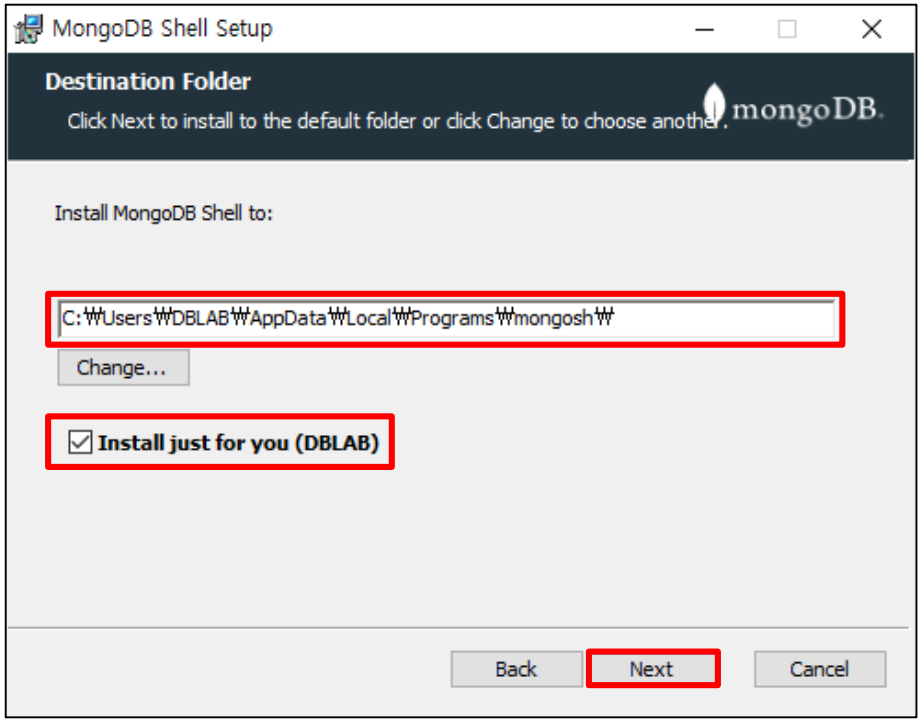


MongoDB Shell 설치(4/5)

4) 다운로드 받은 설치 파일 실행 → Next 클릭

5) 설치 경로 확인 → Install jus for you 체크 → Next 클릭

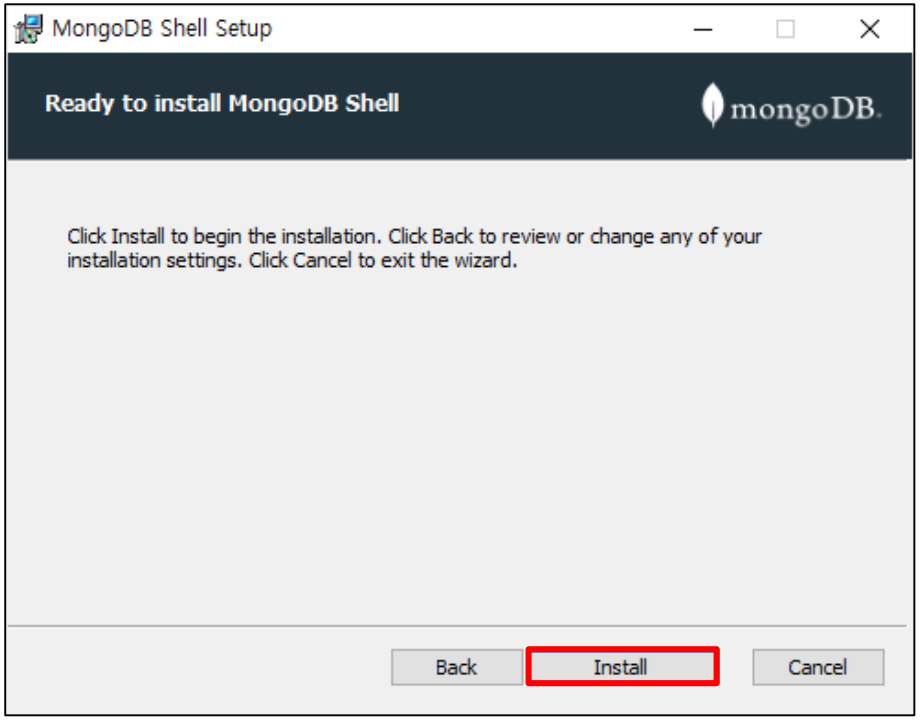


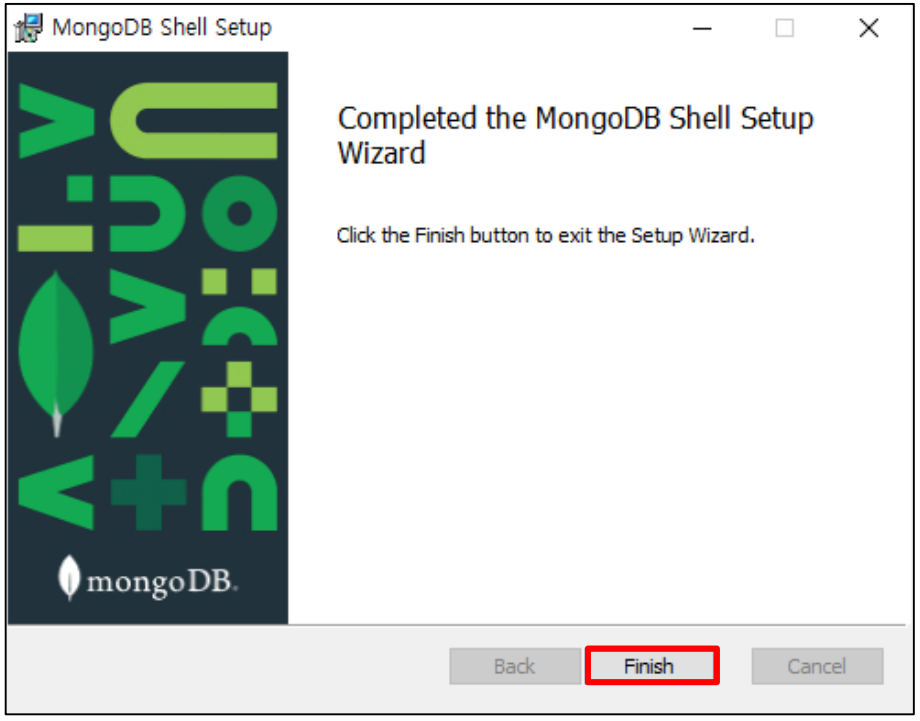


MongoDB Shell 설치(5/5)

6) Install 클릭

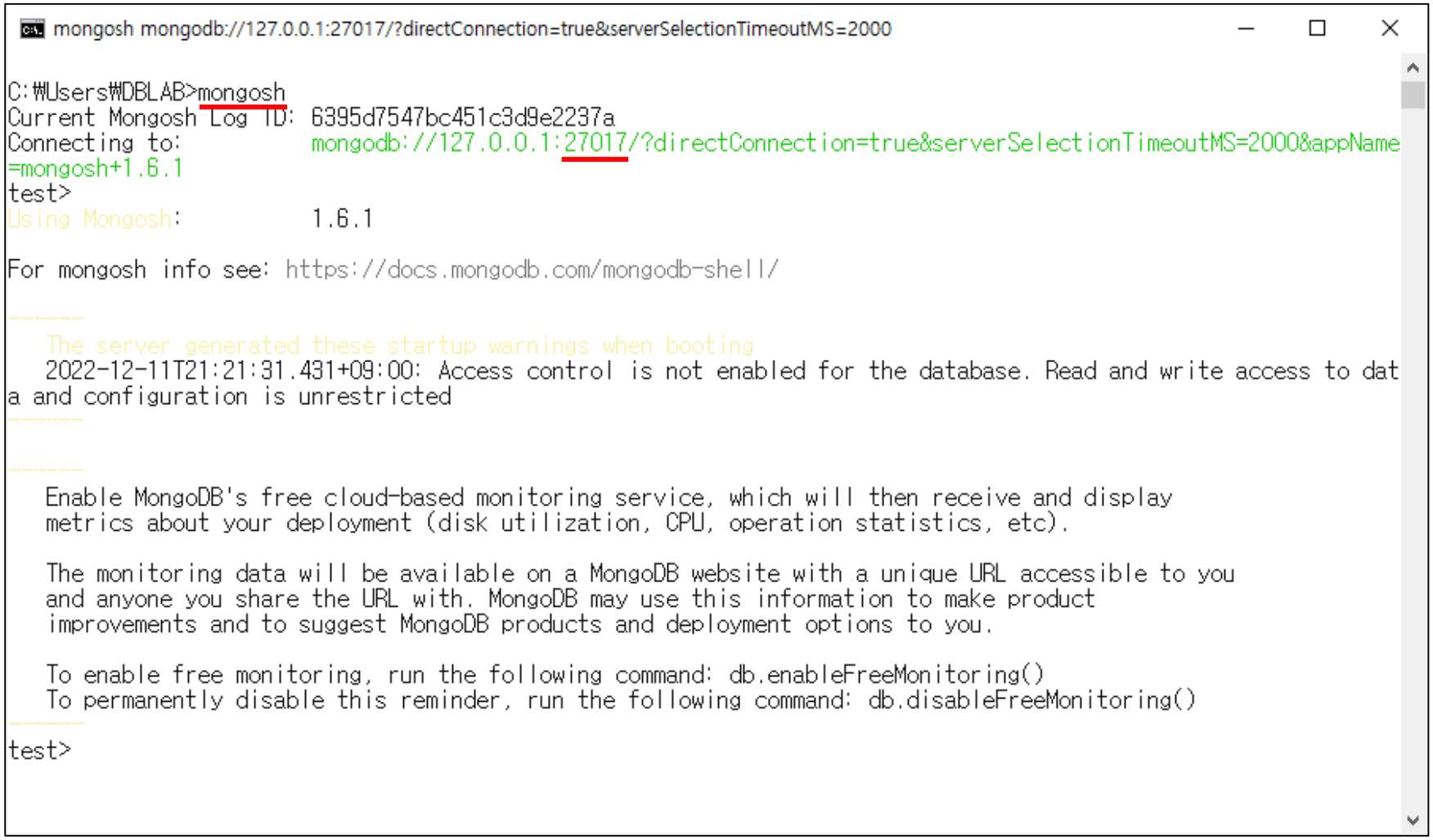
7) 설치 완료 후 Finish 클릭





MongoDB 실행(1/2)

1) cmd창에 "mongosh" 입력 → "27017" 포트로 실행됨을 확인



MongoDB 실행(2/2)

2) "show dbs" 입력

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test> show dbs
admin    40.00 KiB
config   60.00 KiB
local    80.00 KiB
testdb  144.00 KiB
test>
```

MongoDB 명령어(1/7)

- 데이터베이스 생성
  - 명령어
    - use <생성할 데이터베이스 이름>
  - 예제) testdb 라는 이름의 데이터베이스 생성
    - use testdb

```
test> use testdb
switched to db testdb
testdb>
```

MongoDB 명령어(2/7)

- 현재 사용하고 있는 데이터베이스 조회
  - 명령어
    - db

```
testdb> db
testdb
```

MongoDB 명령어(3/7)

- Collection 생성
  - Collection

    데이터베이스 내에 실제로 사용되는 도큐먼트
  - 명령어
    - db.createCollection(“<생성할 collection 이름>”)
  - 예제) customers 라는 이름의 collection 생성
    - db.createCollection(“customers”)

```
testdb> db.createCollection("customers")
{ ok: 1 }
testdb>
```

MongoDB 명령어(4/7)

- 데이터 입력
  - 명령어
    - db.<collection 이름>.insertOne(<생성할 데이터>)
    - db.컬렉션명.insertOne({필드명:"값", …})
  - 예제) customers 라는 collection 안에 이름은 gildong, 나이는 22인 고객의 데이터를 생성

```
testdb> db.customers.insertOne({name:"gildong", age: 22})
{
  acknowledged: true,
  insertedId: ObjectId("61af1d91a8497cd40f6ce5ad")
}
testdb>
```

MongoDB 명령어(5/7)

- 데이터 조회
  - 명령어
    - db.<collection 이름>.find (<검색 조건>)
    - db.컬렉션명.find ({필드명 : "조건값", …})
    - db.customers.find() 를 사용하는 경우 전체 내용 검색
  - 예제) customers 라는 collection 안에 이름이 gildong인 데이터 검색

```
testdb> db.customers.find()
[
  {
    _id: ObjectId("61af19d3a8497cd40f6ce5a6"),
    name: 'gildong',
    age: 22
  },
  {
    _id: ObjectId("61af1a78a8497cd40f6ce5a7"),
    name: 'cheolsu',
    age: 33
  },
  {
    _id: ObjectId("61af1a7ba8497cd40f6ce5a8"),
    name: 'yeonghui',
    age: 32
  }
]
testdb> db.customers.find({name: "gildong"})
[
  {
    _id: ObjectId("61af19d3a8497cd40f6ce5a6"),
    name: 'gildong',
    age: 22
  }
]
testdb>
```

MongoDB 명령어(6/7)

- 데이터 갱신
  - 명령어
    - db.<collection 이름>.updateOne
      (<갱신할 데이터 선택>, <갱신할 데이터 입력>)
    - db.컬렉션명.updateOne
      ({필드명:"조건값", …}, {$set: {필드명:"변경값", …})

```
testdb> db.customers.updateOne({name:"gildong"}, {$set: {name:"hana"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
testdb> db.customers.find()
[
  { _id: ObjectId("61af1d91a8497cd40f6ce5ad"), name: 'hana', age: 22 },
  {
    _id: ObjectId("61af1dd4a8497cd40f6ce5ae"),
    name: 'cheolsu',
    age: 33
  },
  {
    _id: ObjectId("61af1dd4a8497cd40f6ce5af"),
    name: 'yeonghui',
    age: 32
  }
]
testdb>
```

MongoDB 명령어(7/7)

- 데이터 삭제
  - 명령어
    - db.<collection 이름>.deleteOne(<삭제할 데이터 선택>)
    - db.컬렉션명.deleteOne({필드명:"조건값", …})
  - 예제) customers 라는 collection 안에 이름이 hana인 데이터를 삭제

```
testdb> db.customers.deleteOne({name:"hana"});
{ acknowledged: true, deletedCount: 1 }
testdb> db.customers.find()
[
  {
    _id: ObjectId("61af1dd4a8497cd40f6ce5ae"),
    name: 'cheolsu',
    age: 33
  },
  {
    _id: ObjectId("61af1dd4a8497cd40f6ce5af"),
    name: 'yeonghui',
    age: 32
  }
]
testdb>
```

MongoDB 명령어 모음(1/2)

Name	Methods	Description	Example

insert	insertOne	단일의 도큐먼트를 collection에 insert	db.collection.insertOne(
			<document>,
			{
			writeConcern: <document>
			}
			)

	insertMany	다수의 도큐먼트를 collection에 insert	db.collection.insertMany(
			[<document 1>, <document 2>, ...],
			{
			writeConcern: <document>,
			ordered: <boolean>
			}
			)

update	updateOne	특정 조건에 맞는 다수의 도큐먼트가 있더라도 하나의 도큐먼트만 update	db.collection.updateOne(
			<filter>, <update>,
			{
			upsert: <boolean>,
			writeConcern: <document>,
			collation: <document>,
			arrayFilters: [<filterdocument1>, ...],
			hint: <document|string>
			}
			)

	updateMany	특정 조건에 맞는 모든 도큐먼트를 update	db.collection.updateMany(
			<filter>, <update>,
			{
			upsert: <boolean>,
			writeConcern: <document>,
			collation: <document>,
			arrayFilters: [<filterdocument1>, ...],
			hint: <document|string>
			}
			)

MongoDB 명령어 모음(2/2)

Name	Methods	Description	Example

update	replaceOne	특정 조건에 맞는 다수의 도큐먼트가 있더라도 하나의 도큐먼트만 replace	db.collection.replaceOne(
			<filter>, <replacement>,
			{
			upsert: <boolean>,
			writeConcern: <document>,
			collation: <document>,
			hint: <document|string>
			}
			)

delete	deleteOne	특정 조건에 맞는 다수의 도큐먼트가 있더라도 하나의 도큐먼트만 delete	db.collection.deleteOne(
			<filter>,
			{
			writeConcern: <document>,
			collation: <document>,
			hint: <document|string>
			}
			)

	deleteMany	특정 조건에 맞는 모든 도큐먼트를 delete	db.collection.deleteMany(
			<filter>,
			{
			writeConcern: <document>,
			collation: <document>
			}
			)

MongoDB 연산자

비교 연산자

Name	Description	Example
$eq(equals)	주어진 값과 일치하는 값	Syntax: (field: { $eq: value })
$gt(greater than)	주어진 값보다 큰 값	Syntax: (field: { $gt: value })
$gte(greater than or equals)	주어진 값보다 크거나 같은 값	Syntax: (field: { $gte: value })
$in	주어진 배열 안에 속하는 값	Syntax: (field: { $in: [value1, ··· , valueN])
$lt(less than)	주어진 값보다 작은 값	Syntax: (field: { $lt: value })
$lte(less than or equals)	주어진 값보다 작거나 같은 값	Syntax: (field: { $lte: value })
$ne(not equal)	주어진 값과 일치하지 않는 값	Syntax: (field: { $ne: value })
$nin(not in)	주어진 배열 안에 속하지 않는 값	Syntax: (field: { $nin: [value1, ··· , valueN])

논리 연산자

Name	Example
$or	Syntax: ($or: [{ expression1 }, ··· , { expressionN}])
$and	Syntax: ($and: [{ expression1 }, ··· , { expressionN}])
$not	Syntax: ({ field: { $not: { operator-expression } } })
$nor	Syntax: ({ $not: [{ expression1 }, ··· , { expressionN }] })

참고 사이트

- https://docs.mongodb.com/manual/reference/operator/query/