

MongoDB

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Настроить среду разработки, изучить простейшие CRUD операции и операторы

ПЛАН ЗАНЯТИЯ

Введение в NoSQL, типы NoSQL-баз данных, примеры

Сценарии использования NoSQL

Установка MongoDB compass

Основы работы с MongoDB

Базовые структуры данных

MongoDB Atlas - подключение

Установка расширения для VSCode



MongoDB for VS Code v1.5.0

MongoDB  mongodb.com

 1,373,007

     (38)

Connect to MongoDB and Atlas directly from your VS Code envi...

Disable 

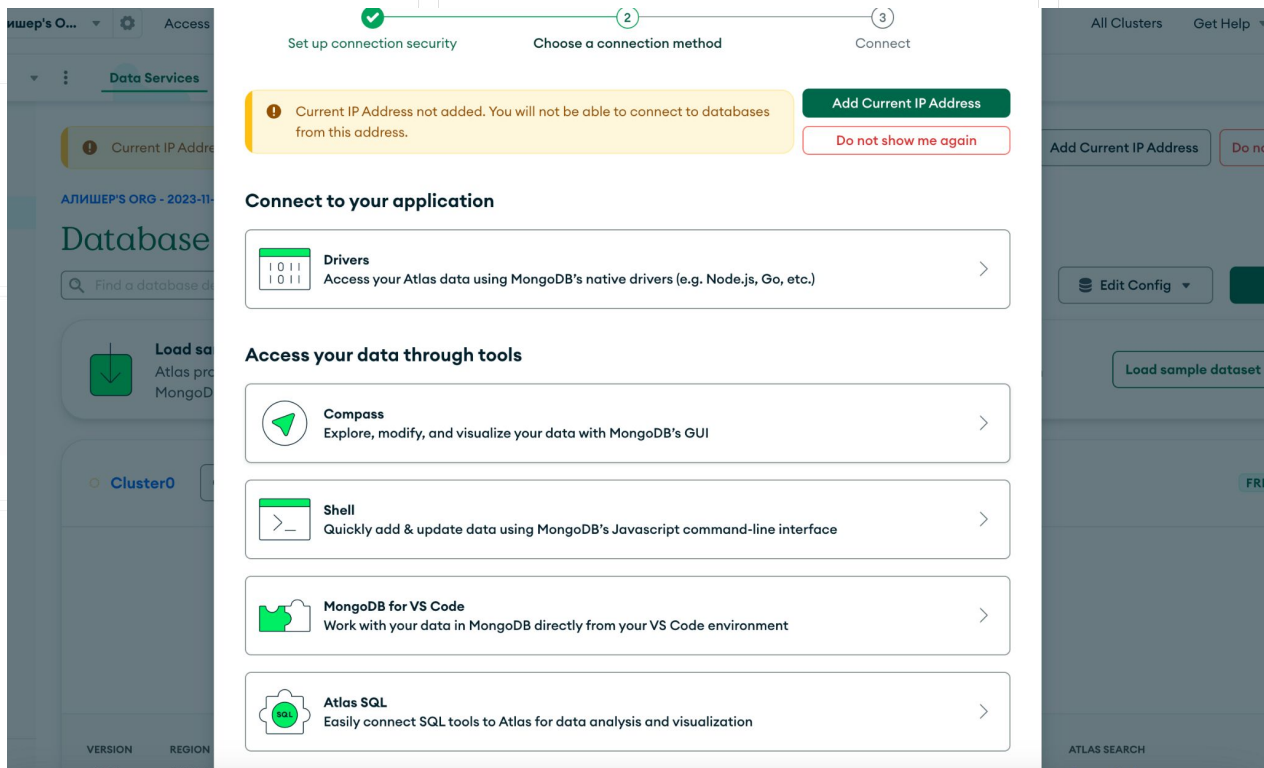
Uninstall 



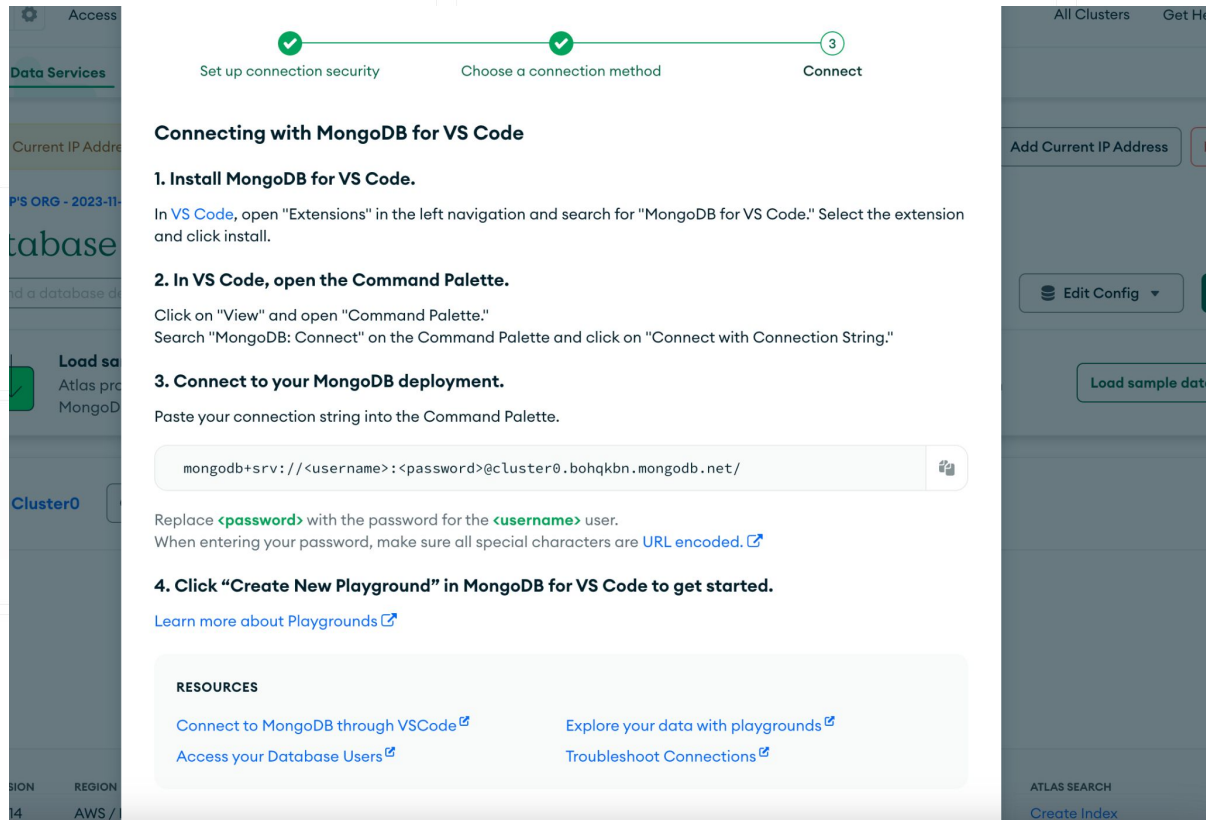
This extension is enabled globally.

Регистрация на MongoDB Atlas

<https://www.mongodb.com/cloud/atlas/register>



Копируем строку для подключения в VSCode



The screenshot shows the 'Access' page in the MongoDB Atlas console. At the top, there's a progress bar with three steps: 'Set up connection security' (completed), 'Choose a connection method' (completed), and 'Connect' (active). Below this, the title 'Connecting with MongoDB for VS Code' is followed by four numbered steps. Step 1: '1. Install MongoDB for VS Code.' with instructions to search for the extension in VS Code. Step 2: '2. In VS Code, open the Command Palette.' with instructions to search for 'MongoDB: Connect'. Step 3: '3. Connect to your MongoDB deployment.' with instructions to paste the connection string. A text box displays the connection string: 'mongodb+srv://<username>:<password>@cluster0.bohqkbn.mongodb.net/'. Step 4: '4. Click "Create New Playground" in MongoDB for VS Code to get started.' with a link to learn more about playgrounds. At the bottom, there's a 'RESOURCES' section with links to 'Connect to MongoDB through VSCode', 'Access your Database Users', 'Explore your data with playgrounds', and 'Troubleshoot Connections'. The left sidebar shows 'Data Services' and 'Cluster0'. The right sidebar shows 'All Clusters' and 'Get Help'.

Access

Data Services

Current IP Address

Load sample data

Cluster0

REGION

14

AWS /

3

Set up connection security

Choose a connection method

Connect

Connecting with MongoDB for VS Code

- 1. Install MongoDB for VS Code.**

In **VS Code**, open "Extensions" in the left navigation and search for "MongoDB for VS Code." Select the extension and click install.
- 2. In VS Code, open the Command Palette.**

Click on "View" and open "Command Palette."
Search "MongoDB: Connect" on the Command Palette and click on "Connect with Connection String."
- 3. Connect to your MongoDB deployment.**

Paste your connection string into the Command Palette.

```
mongodb+srv://<username>:<password>@cluster0.bohqkbn.mongodb.net/
```

Replace **<password>** with the password for the **<username>** user.
When entering your password, make sure all special characters are [URL encoded](#).
- 4. Click "Create New Playground" in MongoDB for VS Code to get started.**

[Learn more about Playgrounds](#)

RESOURCES

- [Connect to MongoDB through VSCode](#)
- [Access your Database Users](#)
- [Explore your data with playgrounds](#)
- [Troubleshoot Connections](#)

All Clusters

Get Help

Add Current IP Address

Edit Config

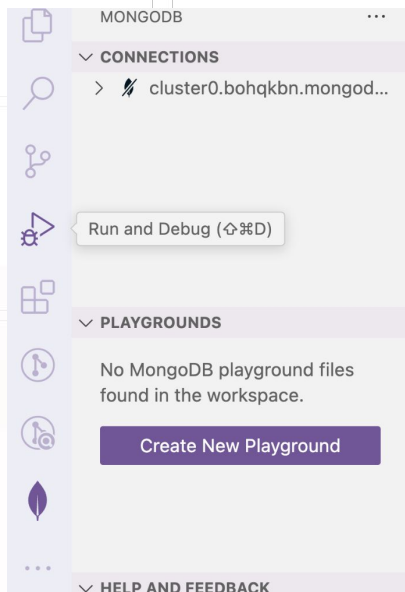
Load sample data

ATLAS SEARCH

Create Index

Подключение к MongoDB Atlas через VSCode и начинаем работать в песочнице

Не забудьте поменять пароль в connectionString на ваш пароль

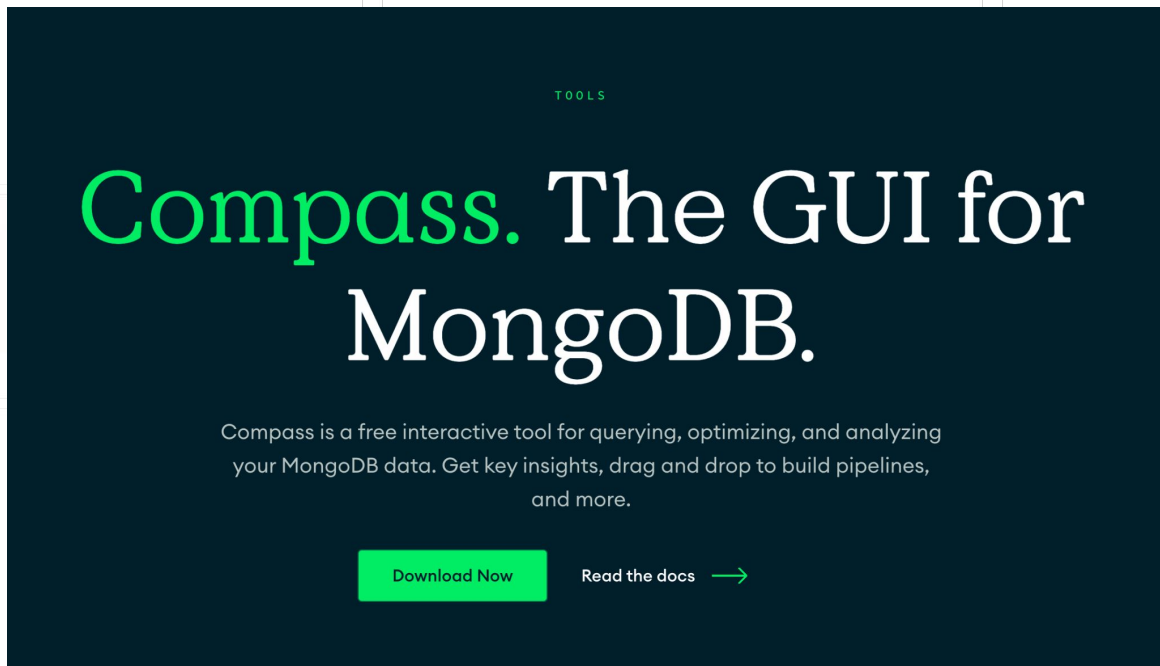


Show All Commands   P

Go to File  P

Скачивание и установка Compass

<https://www.mongodb.com/products/tools/compass>



Подключение Compass к кластеру через connection string

Connect to Cluster0

Set up connection security Choose a connection method **Connect**

Connecting with MongoDB Compass

1. Choose your version of Compass

1.12 or later

See your Compass version in "About Compass"

2. Copy the connection string, then open MongoDB Compass

mongodb+srv://<username>:<password>@cluster0.bohqbkn.mongodb.net/

Replace **<password>** with the password for the **<username>** user.
When entering your password, make sure that any special characters are [URL encoded](#).

RESOURCES

[Connect with Compass](#) [Import and Export Data](#)
[Access your Database Users](#) [Troubleshoot Connections](#)

Принципиальные отличия MongoDB от RDBMS

1. Другой язык для выполнения запросов (NoSQL - QUERY API)
2. Модель данных - **ДОКУМЕНТНАЯ** (документ - ассоц/массив)
3. Динамическая схема данных (без схемы, schemaless) - каждый документ может иметь свой набор полей
4. Использует JSON и Binary JSON (BSON)
5. Позволяет хранить вложенные структуры данных

```
users = [  
  {username: 'hacker', email: '123@example.org', is_blocked: true},  
  {username: 'user1', email: 'hello@example.org', phone: '+0000000000'}  
]
```

Database



Базовые структуры данных

1. Скаляр (скалярное значение)
2. Массив (список значений)
3. Ассоц/массив (пары ключ/значение)
4. Множество (список уникальных элементов)

Категории запросов

1. CRUD - более простые запросы
2. Aggregation - получение вычисленных данных

MongoDB: CRUD

Create

- `insertOne()` добавить один документ
 - один аргумент
 - ассоц/массив (объект)
- `insertMany()` добавить несколько документов
 - один аргумент
 - массив ассоц/массивов

MongoDB: CRUD

Read

- `findOne()` найти (выбрать) один документ
- `find()` найти (выбрать) несколько документов
 - аргументы
 - `filter`
 - `projection`
- `countDocuments()` ко-во совпадений
 - аргументы
 - `filter`

MongoDB: CRUD

Update

- `updateOne()` изменить один документ
- `updateMany()` изменить несколько документов
 - аргументы
 - `filter`
 - `action`

MongoDB: CRUD

Delete

- `deleteOne()` удалить один документ
- `deleteMany()` удалить несколько документов
 - аргументы
 - `filter`

Создание коллекции и добавления документа

Коллекция `users` создалась автоматически, когда мы попытались добавить в нее документ `{ name: "Fred" }`

Иными словами, мы записали информацию о пользователе в бд

```
db.users.insertOne({ name: "Fred" });
```

Вложенность

Пример вложенности: поле
адрес - это объект

Обратите внимание, что поля
этого пользователя
отличаются, у нас появилось
дополнительное поле **address**.

MongoDB позволяет больше
гибкости, за счет того, что нам
не требуется сначала
создавать таблицу как
PostgreSQL

```
db.users.insertOne ({  
  name: "Davit",  
  age: 34,  
  address: { city: "Berlin" }  
});
```

Создание нескольких документов

Обратите внимание, мы добавляем в коллекцию сразу несколько документов.

Соответственно, аргументом выступает массив объектов

```
db.users.insertMany([  
  {name: "Egor", age: 25},  
  {name: "Ded Igor", age: 76}  
]);
```

Пример вложенности

В данном примере мы имеем дело со вложенным массивом

```
db.users.insertOne(  
  {  
    name: "John",  
    age: 20,  
    hobbies: ["music", 'surfing', 'video-games',  
              'snowboard']  
  }  
);
```

READ: пример получения всех пользователей

Получить всех пользователей

```
db.users.find();
```

LIMIT

Получить первые три
документа из коллекции **users**

```
db.users.find().limit(3);
```


SORT

Получить отсортированный в алфавитном порядке по именам список пользователей

```
db.users.find().sort({name: 1});
```

SORT

Получить отсортированный в
обратном алфавитном порядке
по именам список
пользователей

```
db.users.find().sort({name: -1});
```

SORT

Пример более сложной
сортировки по имени, и если
имя совпадает по возрасту

```
db.users.find().sort({name: -1, age: 1});
```

find

Мы получим все документы
удовлетворяющие условию в
скобках

Все user`ы с возрастом 34 года

```
db.users.insertOne({name: "Anna", age: 34});  
db.users.find({age: 34});
```

findOne

Мы получим только первый документ, удовлетворяющий условию в скобках.

То есть первый попавшийся user с возрастом 34 года

```
db.users.findOne ({age: 34});
```

Операторы

- **\$eq** равно (equal)
- **\$ne** не равно (not equal)
- **\$gt** больше (greater than)
- **\$gte** больше или равно (greater than or equal)
- **\$lt** меньше (less than)
- **\$lte** меньше или равно (less than or equal)
- **\$in** проверка принадлежности к списку значений
- **\$nin** не принадлежит списку значений

Пример использования оператора \$gt

Получим всех пользователей
старше 25 лет

```
db.users.find({age: {$gt: 25}});
```

Пример использования оператора \$lt

Получим всех пользователей
младше 25 лет

```
db.users.find({age: {$lt: 25}});
```


Пример использования оператора \$lte

Получим всех пользователей,
меньше или равен 25 годам

```
db.users.find({age: {$lte: 25}});
```

Пример использования оператора \$lte

Получим всех пользователей, чье имя равно "Egor".

Для данного запроса мы могли бы обойтись вовсе без \$eq

```
db.users.find({ name: { $eq: "Egor" } });
```

Проекция

Если мы хотим получить не всю информацию о пользователе, а только определенную, мы можем указать, какие поля нас интересуют вторым параметром

```
db.users.find({age: 34}, {name: 1});
```

Проекция

Все то же самое, но без
отображения id

```
db.users.find({age: 34}, {name: 1, _id: 0});
```

Проекция

С отображением всех полей за исключением id

```
db.users.find({age: 34}, {_id: 0});
```

Пример использования оператора \$in

Выведем всех пользователей с именами John или Anna

```
db.users.find({name: {$in: ["John", "Anna"]}});
```

Пример использования оператора \$nin

Выведем всех пользователей с именами не соответствующими John или Anna

```
db.users.find({name: {$nin: ["John", "Anna"]}});
```

Получается своего рода "черный список"

Пример использования оператора \$and

С помощью \$and можно объединять несколько условий, например, мы получаем всех пользователей, чей возраст 34 года и чье имя не Анна.

Чаще всего мы можем обойтись без использования \$and

```
db.users.find({$and: [{age: 34}, {name: {$ne: "Anna"}}]});
```

```
db.users.find({age: 34, name: {$ne: "Anna"}});
```


Пример использования оператора \$or

С помощью \$or можно получить значения для которых выполняется хотя бы одно условие из перечисленных

Мы получим всех чей возраст 76 или имя равно "Ded Igor"

```
db.user.find({$or: [{age: 76}, {name: "Ded Igor"}]});
```

Пример использования оператора **\$not**

■ Все у кого возраст не больше семидесяти

```
db.users.find({age: {$not: {$gt: 70}}});
```

Пример использования оператора **\$exists**

Exist проверяет наличие в документе поля.

В данном примере мы получим всех, у кого есть поле age.

```
db.users.find({age: {$exists: true}});
```

Пример использования оператора **\$expr**

\$expr позволяет использовать более сложные выражения сравнения

Для примера добавим нескольких пользователей

Вывести всех, кто живет по средствам: `salary > costs`

Вывести тех, кто **не** живет по средствам: `salary < costs`

NB: перед названием свойства добавляем \$

```
db.users.insertMany([
  {name: "Fred", salary: 2000, costs: 1500},
  {name: "Kristina", salary: 1300, costs: 2500}
]);
```

```
db.users.find({$expr: {$gt: ["$salary", "$costs"]}});
```

```
db.users.find({$expr: {$lt: ["$salary", "$costs"]}});
```

Пример использования оператора \$in

```
db.users.insertMany([
  {name: "John Snow", hobbies: ["swords", "bows", "wolfs", "red-head"], age: 20},
  {name: "Han Solo", hobbies: ["space", "blasters"], age: 36}
]);
```

// выбрать людей с хотя одним из указанных хобби: "space", "snowboard"

```
db.users.find({hobbies: {$in: ["space", "snowboard"]}});
```

// Все кто не интересуется "space", "snowboard"

```
db.users.find({ hobbies: { $nin: ["space", "snowboard"] } });
```



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH