**Безопасность и администрирование MongoDB**

**1. Введение в безопасность MongoDB**

MongoDB включает множество инструментов для обеспечения безопасности и администрирования:

Управление доступом через пользователей и роли.

Механизмы аутентификации и авторизации.

Бэкап и восстановление данных для защиты информации.

Эти возможности позволяют настроить базу данных так, чтобы она была защищена от несанкционированного доступа и потери данных.

**2. Управление пользователями и ролями**

**2.1. Создание пользователя**

Пользователи в MongoDB создаются для каждой базы данных. У них есть имя, пароль и роли, определяющие доступ.

Пример: Создание пользователя с ролью администратора.

Подключение к административной базе:

use admin;

Создание пользователя:

db.createUser({

  user: "adminUser",

  pwd: "securePassword",

  roles: [{ role: "userAdminAnyDatabase", db: "admin" }]

});

Проверка:

db.getUsers();

**2.2. Роли в MongoDB**

Роли предоставляют доступ к определённым действиям. Основные типы ролей:

Административные роли: userAdmin, dbAdmin.

Операционные роли: read, readWrite.

Пользовательские роли: создаются вручную для специфических задач.

Пример: Создание роли для чтения данных.

Создание роли:

db.createRole({

  role: "customReadRole",

  privileges: [

      { resource: { db: "exampleDB", collection: "" }, actions: ["find"] }

  ],

  roles: []

});

Назначение роли пользователю:

db.grantRolesToUser("exampleUser", ["customReadRole"]);

**3. Аутентификация и авторизация**

**Аутентификация**

Аутентификация проверяет личность пользователя, предоставляющего доступ. MongoDB поддерживает:

* SCRAM (по умолчанию): безопасный механизм с использованием хэшей паролей.
* LDAP: централизованная аутентификация.
* x.509: использование сертификатов.

Пример: Включение аутентификации.

Включить аутентификацию в конфигурационном файле mongod.conf:

security:

    authorization: "enabled"

Перезапуск MongoDB:

systemctl restart mongod

Подключение с аутентификацией:

mongo -u adminUser -p securePassword --authenticationDatabase admin

**Авторизация**

Авторизация управляет, что пользователь может делать в базе данных. После включения авторизации все действия требуют наличия роли.

Пример: Ограничение доступа к определённым коллекциям.

Назначение роли для работы с коллекцией:

}db.grantRolesToUser("exampleUser", [{

  role: "readWrite",

  db: "exampleDB"

}]);

**4. Бэкап и восстановление данных**

**Бэкап данных**

MongoDB поддерживает несколько способов создания резервных копий:

1. mongodump: утилита для экспорта данных.
2. Снапшоты файловой системы: используется для больших объёмов данных.

Пример: Создание резервной копии с помощью mongodump.

  mongodump --out /backup/mongodb/

/

**Восстановление данных**

Пример: Восстановление данных из резервной копии с помощью mongorestore.

  mongorestore --dir /backup/mongodb/

**Автоматизация бэкапов**

Использование планировщика задач (например, cron) для автоматических бэкапов:

Добавление задачи в cron:

  0 2 \* \* \* mongodump --out /backup/mongodb/$(date +\%F)

**5. Примеры настройки безопасности**

Пример 1: Настройка доступа для нескольких пользователей.

Создание пользователей для базы данных:

  db.createUser({

    user: "dataReader",

    pwd: "readerPassword",

    roles: [{ role: "read", db: "exampleDB" }]

});

db.createUser({

    user: "dataWriter",

    pwd: "writerPassword",

    roles: [{ role: "readWrite", db: "exampleDB" }]

});

Пример 2: Защита подключения с использованием SSL.

Генерация сертификатов и настройка MongoDB для использования SSL через конфигурационный файл:

  net:

  ssl:

      mode: requireSSL

      PEMKeyFile: /path/to/certificate.pem

**6. Преимущества и недостатки**

Преимущества:

* Гибкая настройка ролей для пользователей.
* Поддержка современных методов аутентификации.
* Возможность автоматизации бэкапов.

Недостатки:

* Сложность управления в больших проектах.
* Требуются дополнительные ресурсы для настройки безопасности.