Тема 16. MongoDB. Аутентификация и авторизация. Шифрование данных. Контроль доступа. Настройка резервного копирования.

Цель занятия:

Познакомиться с концепциями аутентификации и авторизации в MongoDB. Рассмотреть шифрование данных для защиты информации. Разобраться в организации контроля доступа к базе данных и настройке резервного копирования базы данных для обеспечения безопасности данных.

Учебные вопросы:

1. Аутентификация и авторизация в MongoDB.
2. Шифрование данных в MongoDB3. Обзор встроенных инструментов мониторинга MySQL.
3. Контроль доступа (Access Control).
4. Настройка резервного копирования в MongoDB.

1. Аутентификация и авторизация в MongoDB.

Аутентификация — это процесс подтверждения личности пользователя, при котором MongoDB проверяет, кто обращается к базе данных.

Без включенной аутентификации любой пользователь может получить доступ ко всей базе данных.

MongoDB поддерживает несколько механизмов аутентификации для различных сценариев использования.

Типы аутентификации в MongoDB:

- **SCRAM** (Salted Challenge Response Authentication Mechanism). SCRAM-SHA-1 и SCRAM-SHA-256 — наиболее часто используемые механизмы. Пароль пользователя преобразуется в хэш для безопасного хранения. SCRAM включен по умолчанию в Community и Enterprise версиях MongoDB.
- **X.509**. Использует цифровые сертификаты для аутентификации клиентов. Актуален для предприятий, где используются инфраструктуры открытых ключей (PKI).
- **LDAP** (Lightweight Directory Access Protocol). MongoDB взаимодействует с LDAP-сервером для проверки учетных данных пользователя.Используется для централизованной аутентификации.
- **Kerberos**. Применяется для единого входа (Single Sign-On) в крупных организациях. Требует настройки службы Kerberos.

Этапы настройки администратора с паролем

1. Запуск MongoDB без аутентификации (временно)

Вначале MongoDB должен быть запущен без аутентификации, чтобы вы могли создать первого пользователя-администратора.

```
mongod --dbpath /var/lib/mongodb
```

--dbpath: Указывает путь к базе данных. Убедитесь, что это корректный путь к вашим данным.

На этом этапе аутентификация не включена, и вы можете свободно создать пользователя.

2. Установка пароля администратора

Подключитесь к MongoDB с помощью mongosh:

```
mongosh
```

Переключитесь на базу данных admin (это системная база данных, где обычно создаются администраторы):

```
use admin
```

Отслеживание

Создайте пользователя-администратора с именем admin и установите пароль:

```
db.createUser({
  user: "admin",
  pwd: "password123",  // Здесь вы устанавливаете пароль администратора
  roles: [ { role: "root", db: "admin" } ]
})
```

user: Имя пользователя.

pwd: Пароль, который вы задаете (например, password123).

roles: Администраторская роль root дает полный доступ ко всем базам данных.

3. Перезапуск MongoDB с включенной аутентификацией

Теперь, когда администратор создан, вы можете перезапустить MongoDB с параметром --auth, чтобы включить аутентификацию:

```
mongod --auth --dbpath /var/lib/mongodb
```

4. Подключение с учетной записью администратора

Теперь, когда аутентификация включена, для подключения необходимо использовать учетные данные администратора:

```
mongosh -u admin -p password123 --authenticationDatabase "admin"
```

-u: Имя пользователя (admin).

-p: Пароль (password123).

--authenticationDatabase: База данных, в которой хранится учетная запись (в данном случае это admin).

Авторизация в MongoDB

Авторизация — это процесс, который определяет, какие действия может выполнять пользователь в базе данных после успешной аутентификации. Это позволяет контролировать доступ к данным и операциям в MongoDB.

Роли в MongoDB

MongoDB использует ролевую модель управления доступом, что позволяет назначать пользователям определенные права в зависимости от их ролей. Вот основные категории ролей:

1. Базовые роли.

- read: Позволяет пользователю только читать данные из базы данных.
- readWrite: Позволяет пользователю как читать, так и записывать данные в базу данных.

2. Административные роли.

- dbAdmin: Предоставляет права администрирования на уровне базы данных (например, создание индексов).
- userAdmin: Позволяет управлять пользователями и ролями в базе данных.
- root: Полный доступ ко всем операциям и базам данных. Эта роль имеет все права на уровне системы.

3. Пользовательские роли.

- Вы можете создавать собственные роли для специфических нужд, определяя конкретные права доступа для выполнения определенных операций.

Пример создания пользователя с ограниченными правами

Рассмотрим, как создать пользователя с ограниченными правами, используя ролевую модель MongoDB.

Шаги:

Запустите mongosh и подключитесь к базе данных:

```
mongosh -u admin -p password123 --authenticationDatabase "admin"
```

Переключитесь на нужную базу данных:

```
use libraryDB
```

Создайте нового пользователя с ограниченными правами:

```
db.createUser({
  user: "readonlyUser",
  pwd: "password456",  // Установите безопасный пароль
  roles: [ { role: "read", db: "libraryDB" } ]  // Назначаем роль "read"
})
```

user: Имя нового пользователя (readonlyUser).

pwd: Пароль, который вы задаете для нового пользователя.

roles: Указывает, какие права будут назначены. В данном случае мы даем пользователю только право на чтение данных в базе данных libraryDB.

Проверка созданного пользователя:

Чтобы убедиться, что пользователь был успешно создан, вы можете выполнить следующую команду:

```
db.getUsers()
```

Это покажет список всех пользователей в текущей базе данных.

2. Шифрование данных в MongoDB3.

Шифрование данных необходимо для защиты конфиденциальной информации от несанкционированного доступа и перехвата.

В случае утечки данных или перехвата информации злоумышленниками, шифрование помогает гарантировать, что данные остаются недоступными без соответствующих ключей расшифровки.

Это особенно важно для организаций, которые обрабатывают личные данные, финансовую информацию или другую чувствительную информацию.

Виды шифрования в MongoDB:

- **Шифрование данных в движении** (TLS/SSL). Шифрование данных в движении защищает информацию, передаваемую между клиентами и сервером MongoDB. Это предотвращает возможность перехвата данных во время передачи по сети. TLS (Transport Layer Security) — это современный протокол, который заменил SSL (Secure Sockets Layer) и обеспечивает безопасность сетевых соединений.
- **Шифрование данных на диске** (Encryption at Rest). Это шифрование данных, когда они хранятся на диске. Оно защищает данные от несанкционированного доступа в случае физического доступа к устройству хранения. Эта функция доступна в MongoDB Enterprise и требует настройки через управление ключами.

3. Контроль доступа (Access Control).

Контроль доступа в MongoDB — это механизм, который позволяет управлять правами пользователей и определять, какие действия они могут выполнять с данными и операциями в базе данных.

Он обеспечивает безопасность и защиту данных от несанкционированного доступа.

Основные принципы контроля доступа:

1. Принцип минимальных привилегий

Определение: Каждый пользователь должен иметь доступ только к тем ресурсам и операциям, которые необходимы для выполнения его работы.

Применение: Это означает, что пользователи не должны иметь прав доступа к данным или функциям, которые не соответствуют их роли в организации.

Преимущества: Этот принцип снижает риски утечки данных и злоупотребления полномочиями, минимизируя потенциальный ущерб в случае компрометации учетной записи.

2. Модель на основе ролей

Определение: MongoDB использует ролевую модель управления доступом, где пользователям назначаются роли, определяющие их права доступа.

Применение: Роли могут быть как предопределенными (например, read, readWrite, dbAdmin), так и пользовательскими, созданными для специфических нужд.

Преимущества: Это упрощает управление правами доступа, так как изменения в ролях автоматически применяются ко всем пользователям, имеющим эти роли.

Рекомендуемые меры контроля доступа:

- Аутентификация пользователей. Всегда включайте аутентификацию для доступа к серверу MongoDB.
- Определение ролей и прав доступа. Создавайте пользователям только те роли, которые им необходимы для работы..
- Применение принципа минимальных привилегий. Ограничивайте доступ пользователей к данным на основе их ролей и потребностей.
- Использование аудита. Включите функции аудита для отслеживания действий пользователей в базе данных.
- Сетевые настройки. Ограничьте доступ к серверу MongoDB с помощью файрволлов и правил доступа по IP-адресам.
- Изоляция среды. Разделяйте среды разработки, тестирования и продакшена, чтобы предотвратить несанкционированный доступ и минимизировать риски.
- Регулярные обновления. Убедитесь, что используемая версия MongoDB обновлена до последней стабильной версии для защиты от известных уязвимостей и улучшения безопасности.
- Мониторинг. Используйте инструменты мониторинга для отслеживания активности базы данных и обнаружения потенциальных угроз.

4. Настройка резервного копирования в MongoDB.

Настройка резервного копирования в MongoDB является важным аспектом управления данными, который позволяет защитить вашу базу данных от потери данных, сбоев системы или других непредвиденных обстоятельств.

В MongoDB есть несколько методов для выполнения резервного копирования, включая физическое и логическое резервное копирование.

Методы резервного копирования в MongoDB:

- Логическое резервное копирование. Используется команда mongodump для создания резервной копии данных в формате BSON. Можно восстанавливать данные с помощью команды mongorestore.
- Физическое резервное копирование. Копирование файлов базы данных напрямую из файловой системы. Требует остановки сервера MongoDB или использования механизма fsync для обеспечения целостности данных.
- Резервное копирование с использованием MongoDB Cloud Backup. Если вы используете MongoDB Atlas, вы можете настроить автоматическое резервное копирование через интерфейс Atlas.

1. Логическое резервное копирование с помощью mongodump

Шаги для создания резервной копии:

Запустите командную строку (CMD) или PowerShell.

Используйте команду mongodump для создания резервной копии:

```
mongodump --uri="mongodb://username:password@localhost:27017/libraryDB" --out "C:\backup\"
```

--uri: URI для подключения к вашей базе данных, включая имя пользователя и пароль.

--out: Путь, где будет сохранена резервная копия. В данном случае резервная копия будет сохранена в папку C:\backup\.

2. Восстановление данных с помощью mongorestore

Шаги для восстановления из резервной копии:

Используйте команду mongorestore для восстановления данных:

```
mongorestore --uri="mongodb://username:password@localhost:27017/" "C:\backup\libraryDB\"
```

В этом примере libraryDB — это название базы данных, которую вы восстанавливаете.

3. Физическое резервное копирование

Шаги для физического резервного копирования:

- Остановите сервер MongoDB (если вы не используете репликацию).
- Скопируйте файлы базы данных. По умолчанию файлы базы данных находятся в: C:\data\db
- Переместите или создайте резервную копию директории.
- Запустите сервер MongoDB снова.

Настройка автоматического резервного копирования:

- Для автоматизации процесса резервного копирования вы можете настроить задачу с помощью **Task Scheduler** в **Windows**.
- В **Linux** можно настроить **Cron** для автоматического резервного копирования.

Практические советы и рекомендации:

- Используйте сложные пароли и двухфакторную аутентификацию.
- Включите аутентификацию и авторизацию для всех экземпляров MongoDB.
- Используйте TLS/SSL для защиты данных при передаче.
- Регулярно делайте резервные копии и проверяйте их работоспособность.
- Ограничьте доступ к базе данных по IP и ролям.
- Мониторьте базу данных на предмет несанкционированного доступа.

Домашнее задание:

1. Повторить материал лекции.

Список литературы:

1. В. Ю. Кара-ушанов SQL — язык реляционных баз данных
2. А. Б. ГРАДУСОВ. Введение в технологию баз данных
3. А.Мотеев. Уроки MySQL

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com