Методы сбора, хранения, обработки и анализа данных

Лекция 17

СУБД MongoDb – безопасность

Обеспечение безопасности

- Аутентификация и авторизация
  - Логины
  - Пользователи
  - Роли
  - Привилегии
  - Объекты и субъекты защиты
- Шифрование
- Аудит

Аутентификация

- Создание пользователей
- Наделение пользователей привилегиями
- Переподключение сервера в режиме аутентификации

Режимы аутентификации

- Role-Based Access Control – для базы данных (user / password)
- Механизмы SCRAM – Salted Challenge Response Authentication Mechanism
- LDAP – Lightweight Directory Access Protocol
- Kerberos
- Microsoft Active Directory Authentication
- x.509 Certificate

Роли и привилегии

- ~ 100 привилегий
- Могут объединяться в пользовательские роли
- Есть встроенные роли
- Объект для назначения привилегий:
  - Коллекция
  - База данных
  - Кластер

Встроенные роли

read	доступ в режиме чтения
readWrite	чтение и редактирование данных
dbAdmin	выполнение административных задач, связанных со схемой, индексирование и сбор статистики
userAdmin	создание и изменение ролей и пользователей в текущей базе данных
dbOwner	чтение и редактирование данных, добавление и удаление пользователей, выдача прав пользователям и создание ролей
dbAdminAnyDatabase	создание индексов, вызов процедур сжатия данных, выполнение задач, связанных со схемой, невозможность создания и удаления пользователей
root	суперпользователь, но с ограничениями; может выполнять большинство действий, но не все: не может изменить системную коллекцию

Создание пользователей

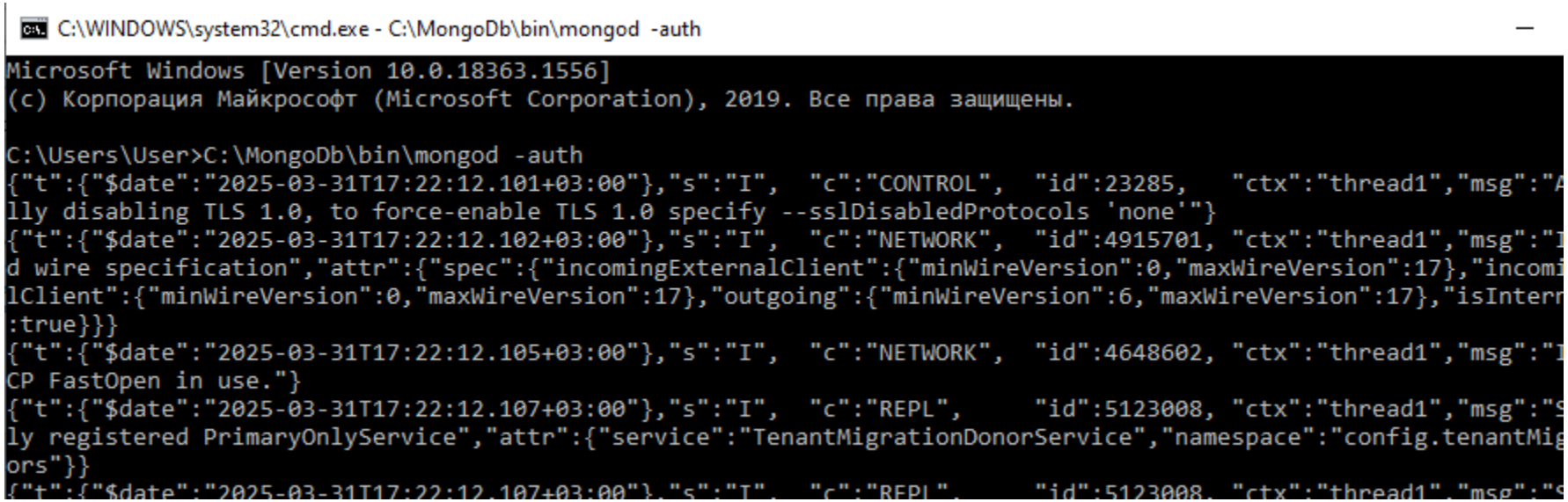
- Создать пользователя
- Указать базу данных
- Указать роль

```
> use all_subjects
```

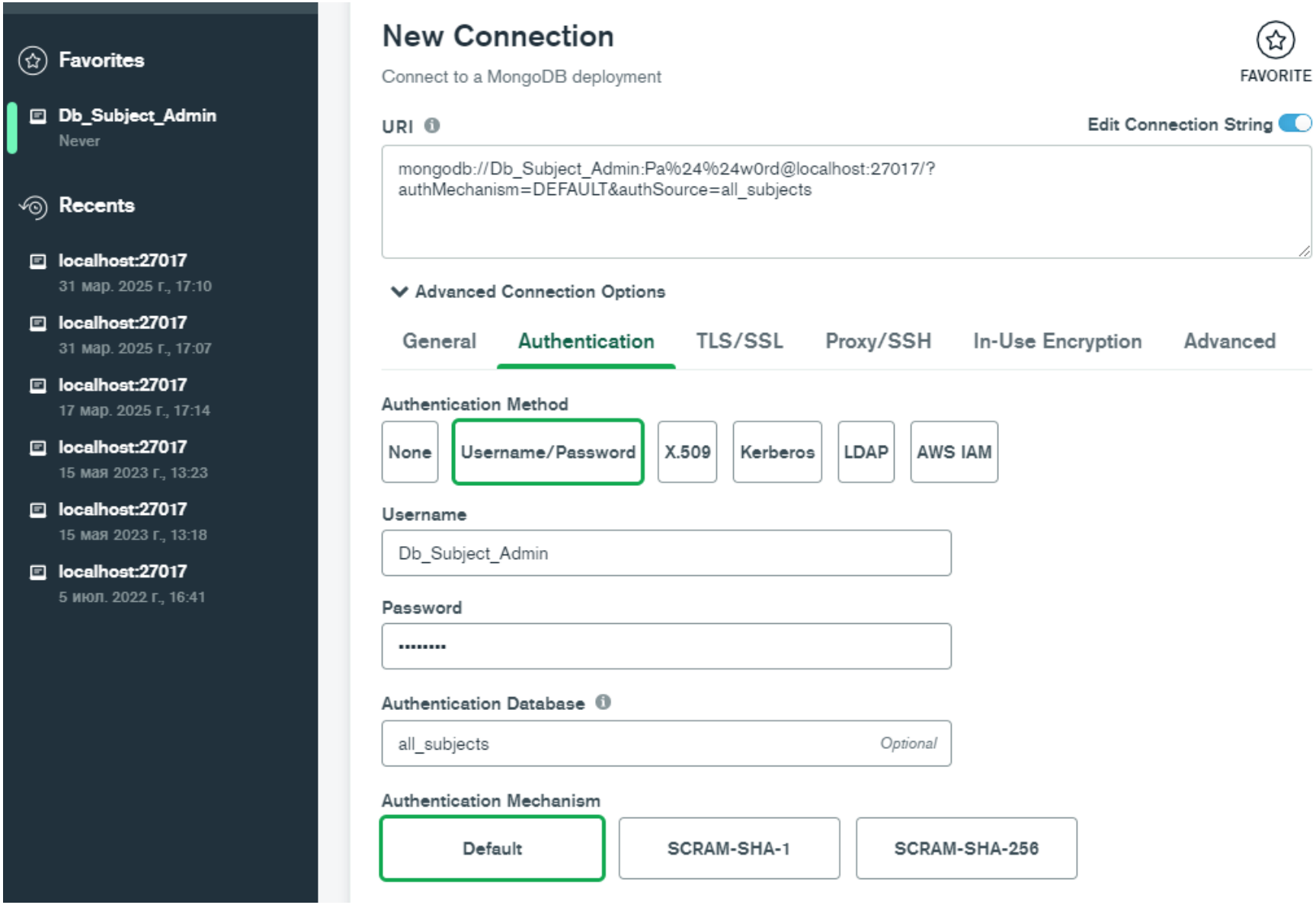
```
> db.createUser({
      user: "Db_Subject_Admin",
      pwd: "Pa$$w0rd",
      roles: ["dbOwner"]
  })
{ ok: 1 }
```

```
> db.createUser({
      user: "Db_Subject_User",
      pwd: "Pa$$w0rd",
      roles: ["readWrite"]
  })
{ ok: 1 }
```

Подключение в режиме аутентификации



Подключение

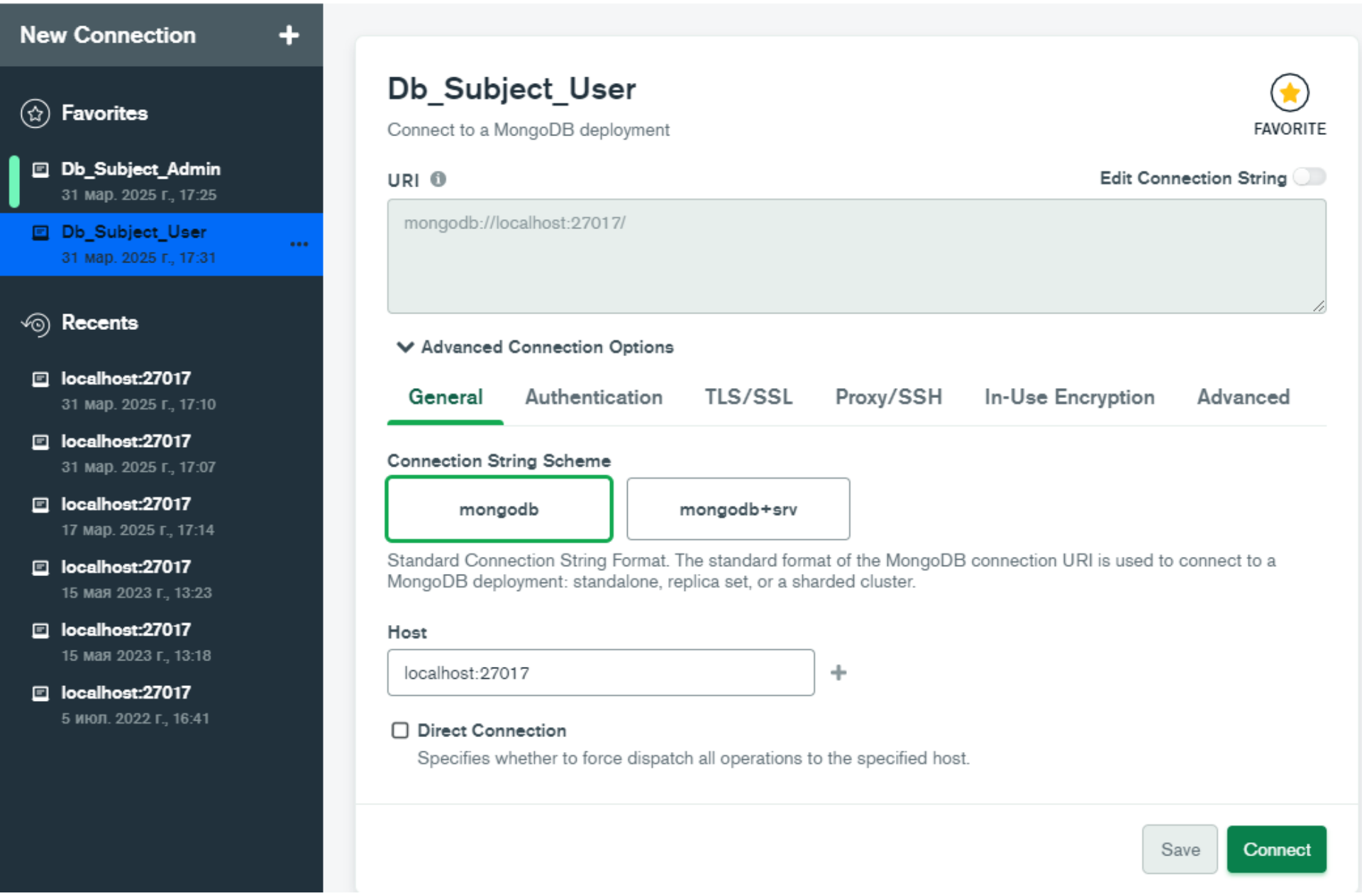


Подключение

- Пользователь может просматривать и изменять данные
- Может создать пользователя и дать ему права

```
> db.places.find().count()
< 70
> db.createUser({
      user: "New_User",
      pwd: "Pa$$w0rd",
      roles: ["userAdmin"]
  })
< { ok: 1 }
all_subjects>
```

Подключение



Подключение

- Пользователь может работать в рамках предоставленных прав

```
>_MONGOSH

> use all_subjects
< 'switched to db all_subjects'
> db.places.find().count()
< 70
> db.createUser({
      user: "Little_User",
      pwd: "Pa$$w0rd",
      roles: ["read"]
  })

✖ ▾ Error: clone(t={}){const r=t.loc||{};return e({loc:new Po
        at Object.serialize (v8.js:256:7)
        at u (C:\MongoDb\Distr\Compass\resources\app.asar.unp
        at postMessage (C:\MongoDb\Distr\Compass\resources\ap
        at i (C:\MongoDb\Distr\Compass\resources\app.asar.unp
```

Подключение

- Пользователь может работать в базе данных, в которой создан

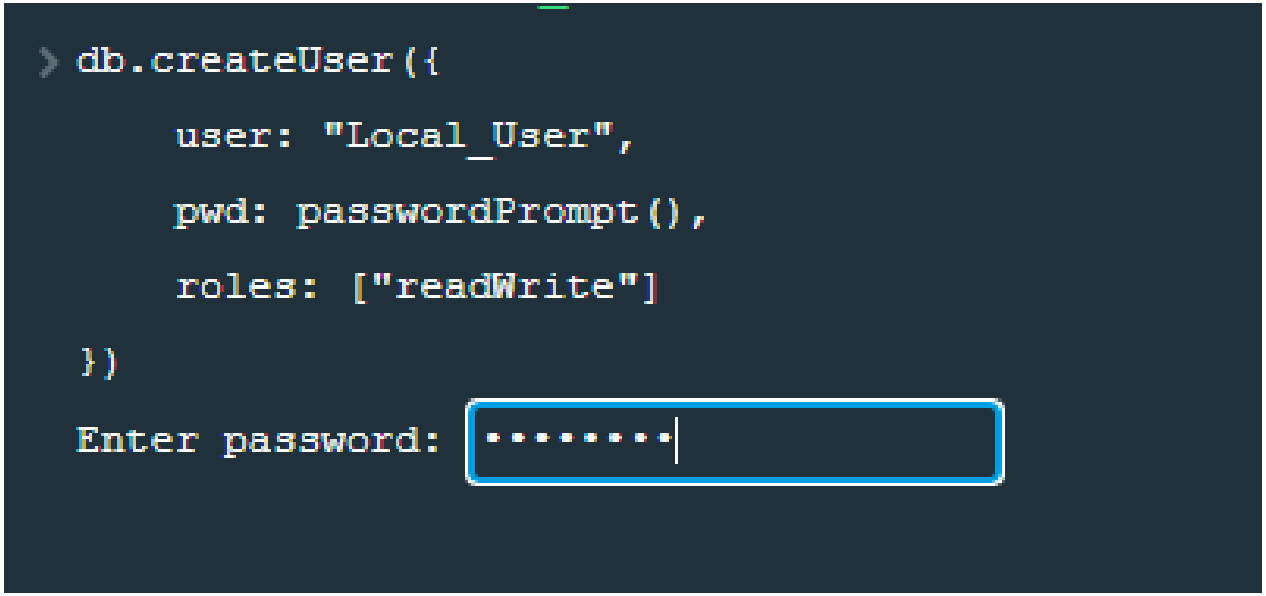
```
> use admin
< 'switched to db admin'
> db.getCollectionNames()
MongoServerError: not authorized on admin to execute command { listCollections: 1, fil
    at Connection.onMessage (C:\MongoDb\Distr\Compass\resources\app.asar.unpacked\node
    at MessageStream.<anonymous> (C:\MongoDb\Distr\Compass\resources\app.asar.unpacked
    at MessageStream.emit (events.js:315:20)
```

```
> use all_subjects
< 'switched to db all_subjects'
> db.getCollectionNames()
< [ 'places', 'reviews', 'areviews', 'locations' ]
all_subjects>
```

Пользователи

- Создание пользователя
- Просмотр всех пользователей и их ролей
- Изменение пользователя
- Изменение пароля
- Удаление пользователя

Создание пользователя



Просмотр всех пользователей

```
> db.getUsers()
< {
    users: [
      {
        _id: 'all_subjects.Db_Subject_Admin',
        userId: UUID("21e078d7-52ea-416c-bd2b-2b453d29d88a"),
        user: 'Db_Subject_Admin',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
      },
      {
        _id: 'all_subjects.Db_Subject_User',
        userId: UUID("8b85ca85-3683-491c-900c-e4f8ab0a567a"),
        user: 'Db_Subject_User',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
      },
      {
        _id: 'all_subjects.Local_User',
        userId: UUID("d6ee826f-a20e-457c-9a01-fe4d8ea8edbe"),
        user: 'Local_User',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
      },
      {
        _id: 'all_subjects.New_User',
        userId: UUID("20fdbc36-0f36-4195-afca-ee2013d6d150"),
        user: 'New_User',
```

Просмотр пользователя

```
> db.getUser("New_User")
< {
    _id: 'all_subjects.New_User',
    userId: UUID("20fdbc36-0f36-4195-afca-ee2013d6d150"),
    user: 'New_User',
    db: 'all_subjects',
    roles: [ { role: 'userAdmin', db: 'all_subjects' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
```

```
> db.getUser("New_User", {
      showCredentials: true,
      showPrivileges: true,
      showAuthenticationRestrictions: true
  })
< {
    _id: 'all_subjects.New_User',
    userId: UUID("20fdbc36-0f36-4195-afca-ee2013d6d150"),
    user: 'New_User',
    db: 'all_subjects',
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ],
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'FxJIcGHlJ8sGKIOZBL/PQA==',
        storedKey: 'OLWIRj8fbyBD5IWrXOSYBbEVwPO=',
        serverKey: 'CuweJvDjiSCaOFeSTNe421+KumU='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: '8S5xHEXR+t7QCSJ9bkI7PEW61XC+GjhhT7IxAg==',
        storedKey: '/MzQRitfXhw8qkhuB/bgWHM5Kawdkmwzd5pQ9yAwPj4=',
        serverKey: 'b7GK+nEQJ4mPVdUua1pcWDY+IeJ8Rgy3x8zSywn8vKk='
      }
    },
    roles: [ { role: 'userAdmin', db: 'all_subjects' } ],
    authenticationRestrictions: [],
    inheritedRoles: [ { role: 'userAdmin', db: 'all_subjects' } ],
    inheritedPrivileges: [ { resource: [Object], actions: [Array] } ],
    inheritedAuthenticationRestrictions: []
  }
```

Изменение пользователя

- db.updateUser()
- db.ChangeUserPassword()

- db.grantRolesToUser()
- db.revokeRolesFromUser()

Изменение пароля

```
> db.changeUserPassword("New_User", "NewPa$$w0rd")
< { ok: 1 }
```

```
> db.getUser("New_User", {
      showCredentials: true,
      showPrivileges: true,
      showAuthenticationRestrictions: true
  })
< {
    _id: 'all_subjects.New_User',
    userId: UUID("20fdbc36-0f36-4195-afca-ee2013d6d150"),
    user: 'New_User',
    db: 'all_subjects',
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ],
    credentials: {
      'SCRAM-SHA-1': {
        iterationCount: 10000,
        salt: 'h4rKoQOGF7oeOTjZc9CoVg==',
        storedKey: '5zVSEl/JqsOsJX41RvH4L38B+b4=',
        serverKey: '/QonsH3bIAPQyoshNfb/V5o3A6Y='
      },
      'SCRAM-SHA-256': {
        iterationCount: 15000,
        salt: 'RQypoG26oL4fgBt5cwSZYrlAAoRyTHkyuSteCg==',
        storedKey: '9X4Q+vKfyJUghdlyuGaCIwktYYQaMFxWQZrGeYd8Ow0=',
        serverKey: 'wml67qzZ3OlSZt59aVLfgrelnC1g8wTW7DViGqVoYAM='
      }
    },
    roles: [ { role: 'userAdmin', db: 'all_subjects' } ],
    authenticationRestrictions: [],
    inheritedRoles: [ { role: 'userAdmin', db: 'all_subjects' } ],
    inheritedPrivileges: [ { resource: [Object], actions: [Array] } ],
    inheritedAuthenticationRestrictions: []
```

Удаление пользователя

```
> db.dropUser("New_User")
< { ok: 1 }
> db.getUsers()
< {
    users: [
      {
        _id: 'all_subjects.Db_Subject_Admin',
        userId: UUID("21e078d7-52ea-416c-bd2b-2b453d29d88a"),
        user: 'Db_Subject_Admin',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
      },
      {
        _id: 'all_subjects.Db_Subject_User',
        userId: UUID("8b85ca85-3683-491c-900c-e4f8ab0a567a"),
        user: 'Db_Subject_User',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
      },
      {
        _id: 'all_subjects.Local_User',
        userId: UUID("d6ee826f-a20e-457c-9a01-fe4d8ea8edbe"),
        user: 'Local_User',
        db: 'all_subjects',
        roles: [Array],
        mechanisms: [Array]
```

Пользовательские роли

- Можно создать свои роли
- На основе ролей
- На основе привилегий

```
> db.createRole({
      role: "Places_Finder",
      privileges: [],
      roles: [
          {
              role: "read",
              db: "all_subjects"
          }
      ],
  })
< { ok: 1 }
```

```
> db.createRole({
      role: "Review_Finder",
      privileges: [
          {
              resource: { db: "all_subjects", collection: "reviews" },
              actions: [ "find" ]
          }
      ],
      roles: [],
  })
< { ok: 1 }
```

Пользовательские роли – просмотр

```
> db.getRoles()
< {
    roles: [
      {
        _id: 'all_subjects.Places_Finder',
        role: 'Places_Finder',
        db: 'all_subjects',
        roles: [Array],
        isBuiltin: false,
        inheritedRoles: [Array]
      },
      {
        _id: 'all_subjects.Review_Finder',
        role: 'Review_Finder',
        db: 'all_subjects',
        roles: [],
        isBuiltin: false,
        inheritedRoles: []
      }
    ],
    ok: 1
  }
```

Пользовательские роли – просмотр

```
> db.getRole("Places_Finder")
< {
    _id: 'all_subjects.Places_Finder',
    role: 'Places_Finder',
    db: 'all_subjects',
    roles: [ { role: 'read', db: 'all_subjects' } ],
    inheritedRoles: [ { role: 'read', db: 'all_subjects' } ],
    isBuiltin: false
  }
all_subjects>
```

Пользовательские роли

- Изменение – db.updateRole()
- Удаление – db.dropRole()
- Добавление привилегий – db.grantPrivilegesToRole()
- Отзыв привилегий – db.revokePrivilegesFromRole()
- Добавление роли – db.grantRolesToRole()
- Удаление роли из роли – db.revokeRolesFromRole()
- Назначение роли пользователю – db.grantRolesToUser()
- Отбор роли у пользователя – db.revokeRolesFromUser()

Аудит

- Необходимо запускать сервер с ключами
  - --auditDestination file
  - --auditFormat
  - --auditPath
- В файле конфигурации необходимо указать события, требующие аудита

Вопросы?