

服务器操作说明

此处以华为云的云服务器说明。

0、补充说明：连接远程服务器

本地连接远程服务器的方式：

- 本地cmd执行打开终端

- ```
ssh root@ip
输入密码即可
```

- 使用winSCP软件;
- 使用xftp软件;
- .....

## 1、node的安装

我们安装软件使用工具：dnf;

cenos8自带dnf。

检查dnf是否可用：

```
dnf --help
```

如果我们希望安装一个软件包，可以进行如下的操作：

```
搜索软件包
dnf search nodejs
查看软件包信息：nodejs的版本是10.21.0
dnf info nodejs
安装nodejs
dnf install nodejs
```

我们会发现版本其实是10.21.0;

- 我们其实希望使用更高的版本，比如最新的LTS或者Current版本;
- 这个时候我们可以使用之前讲过的一个工具：n;

```
安装n
npm install n -g
```

```
通过n安装最新的lts和current
n install lts
n install latest
通过n切换版本
n
```

如果发现切换之后终端没有反应，可以进行重启：

- 方式一：重新通过ssh建立连接；
- 方式二：重启ssh service sshd restart

## 2、mysql的安装

### 2.1 安装

我们依然使用dnf来安装MySQL：

```
查找MySQL
dnf search mysql-server
查看MySQL，这里的版本是8.0.21
dnf info mysql-server
安装MySQL，这里加-y的意思是依赖的内容也安装
dnf install mysql-server -y
```

启动mysql-server：

```
开启MySQL后台服务
systemctl start mysqld
查看MySQL服务：active (running)表示启动成功
systemctl status mysql
随着系统一起启动
systemctl enable mysqld
```

### 2.2 配置

配置MySQL账号和密码：

```
mysql_secure_installation
接下来有一些选项，比如密码强度等等一些
MySQL8开始通常设置密码强度较强，选择2
其他的选项可以自行选择
```

接下来登录：

```
mysql -u root -p
输入密码
```

但是如果我们希望在自己的电脑上直接连接MySQL呢？

- 也就是和MySQL建立远程连接；
- 比如直接Navicat工具中连接MySQL；

这个时候必须要配置root可以远程连接：

```
使用mysql数据库
use mysql;
查看user表中，连接权限，默认看到root是localhost
select host, user from user;
修改权限
update user set host = '%' where user = 'root';
```

注意mysql默认端口是3306，需要在云服务的安全组开放这个端口；

## 2.3 数据库迁移

我们需要将之前项目中（本地）MySQL的数据库迁移到服务器中的MySQL数据库中。

第一步：连接远程服务器，创建指定数据库，如codehub。可以通过命令，也可以通过Navicat直接创建；

第二步：在Navicat工具中本地MySQL直接导出数据库，以本地coderhub为例（右键=>转储SQL文件=>结构+数据）；

第三步：远程服务器的数据库导入数据，以codehub为例（右键=>运行SQL文件=>选择本地导出的sql文件）；

## 3、nginx的安装

### 3.1 安装

后续我们部署会使用nginx，所以需要先安装一下nginx：

```
dnf install nginx
```

启动nginx：

```
启动
systemctl start nginx
查看状态
systemctl status nginx
```

```
随服务器启动而启动
systemctl enable nginx
```

### 3.2 配置

安装完默认nginx存储在**etc/nginx**目录，我们布置静态资源前端项目时，需要修改nginx.conf的配置（目录：etc/nginx/nginx.conf）。

- user设置：user root;
- 在server内修改访问地址：注释掉第一个默认nginx的访问地址 **root /usr/share/nginx/html;**
- 在下面的localtion中配置信息，localtion默认是/，表示/后面接的地址；
- localtion对象中:

```
◦ localtion {
 root /访问地址;
 index index.html index.htm;
 }

 # 以下例子
 localtion {
 root /vue3-ts-element;
 index index.html index.html;
 }

 # 后面浏览器访问http://ip/vue3-ts-element即可
```

- 修改完配置要重启nginx:

```
◦ systemctl restart nginx
```

## 4、jenkins自动化部署

### 4.1 安装Java环境

Jenkins本身是依赖Java的，所以我们需要先安装Java环境：

- 这里我安装了Java-11的环境（新的jenkins已经不支持java-8了）

```
dnf search java-11
dnf install java-11-openjdk.x86_64
```

### 4.2 安装Jenkins

因为Jenkins本身是没有在dnf的软件仓库包中的，所以我们需要连接Jenkins仓库：

- wget是Linux中下载文件的一个工具，-O表示输出到某个文件夹并且命名为什么文件；
- rpm：全称为**The RPM Package Manage**，是Linux下一个软件包管理器；

```
wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

# 导入GPG密钥以确保您的软件合法

```
rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

# 或者

```
rpm --import http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key
```

编辑一下文件/etc/yum.repos.d/jenkins.repo

- 可以通过vim编辑：
  - 终端输入(对该文件编辑)：vi jenkins.repo
  - 键盘按i键，对文件实行插入操作
  - 删除redhat后面的单词(同行-stable)
  - 完成后按esc退出，并按shift+:键，终端会显示:可输入，输入wq（表示保存并退出）

```
[jenkins]

name=Jenkins-stable

baseurl=http://pkg.jenkins.io/redhat

gpgcheck=1
```

如果上面文件不存在，可自己创建该文件，并写入上面代码：

```
1.创建该文件
cd /etc/yum.repos.d
touch jenkins.repo
vim jenkins.repo

2.写入下面内容
[jenkins]

name=Jenkins-stable

baseurl=http://pkg.jenkins.io/redhat

gpgcheck=1

3.退出保存
按esc退出，并按shift+:键；
终端会显示:可输入，输入wq（表示保存并退出）
```

## 安装Jenkins

```
dnf install jenkins # --nogpgcheck(可以不加)
```

启动Jenkins的服务：

```
手动启动jenkins
systemctl start jenkins
查看jenkins启动的状态
systemctl status jenkins
表面随服务器系统启动而启动jenkins (如果报错则按照终端提示的命令去执行)
systemctl enable jenkins
```

补充：如果start启动jenkins失败，可以查看日志原因（根据终端命令），如果发现不了其他原因，可尝试按下面方式解决试试：

```
修改Jenkins 启动文件
vim /lib/systemd/system/jenkins.service

添加如下内容
Environment="JAVA_HOME=/usr/lib/jvm/java-11"

重启Jenkins
systemctl daemon-reload
systemctl restart jenkins.service
```

安装完成后，我们可以访问页面：

- Jenkins默认使用8080端口提供服务（图形化界面），所以需要加入到服务器安全组中（自行添加8080端口），并用浏览器访问<http://ip:8080>：
- 如果开启了防火墙，则必须把对应的端口也在防火墙暴露出去！！！（命令看5中的介绍）
- 第一次登录需要获取服务器管理员的密码，打开的页面会提供一个地址，复制该地址在终端输入：[cat 地址](#)查看密码，复制到刚才的页面登录；

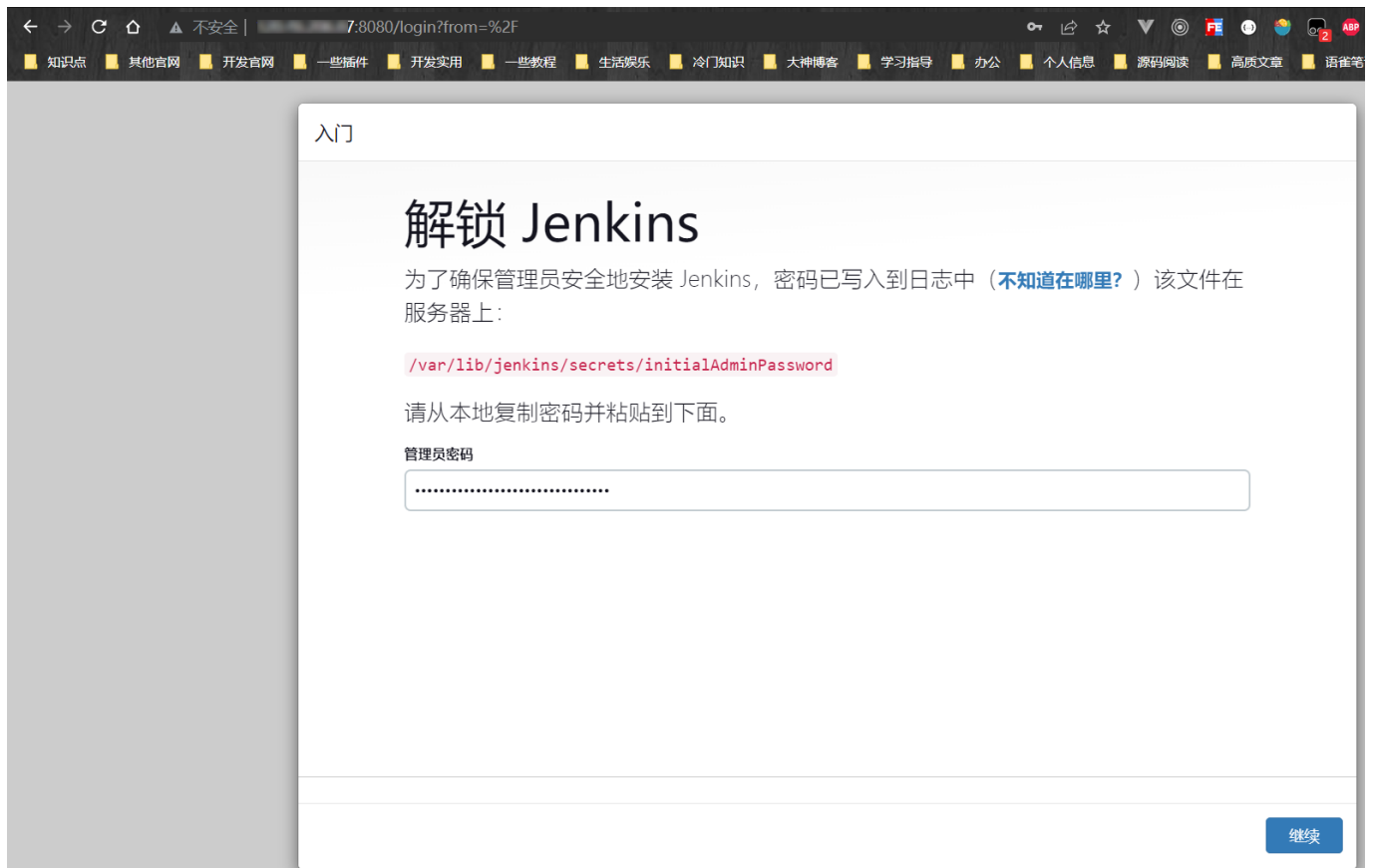
## 支持修改端口号（可选）：

如果要修改默认8080端口为其他，则按下面执行。

```
配置文件文件修改
vim /usr/lib/systemd/system/jenkins.service

修改
Environment="JENKINS_PORT=8089"
```

```
终端重启配置
systemctl daemon-reload
```



### 4.3 Jenkins用户

我们后面会访问centos中的某些文件夹，默认Jenkins使用的用户是 `jenkins`，可能会没有访问权限，所以我们需要修改一下它的用户：

```
配置文件
cd /etc/sysconfig/jenkins

原代码中: JENKINS_USER = "jenkins"改成JENKINS_USER = 'root'
JENKINS_USER = 'root'

重启jenkins
systemctl restart jenkins
```

### 4.4 Jenkins任务

#### 1. 新建任务：

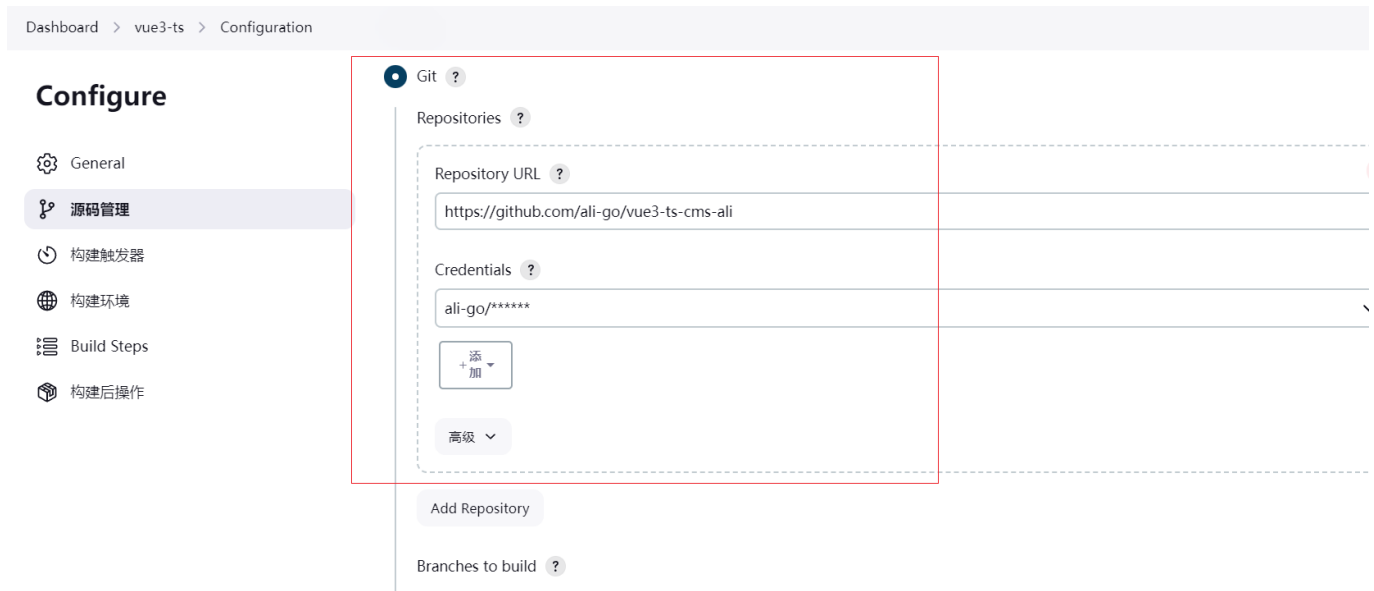
- 点击新建，输入任务名；
- 选择**freeStyle project**（构建一个自由风格的软件项目），点击确定；
- 输入描述，源码管理选择git（注意服务器要安装git，可以用`dnf install git`安装）；
- 把远程仓库git地址复制过来；

## 2. 配置项目：

### 2.1 源码管理

为了不让其他人访问当前要创建的对应上面git项目的私有jenkins仓库，需要添加认证：

- **Credentials**中选择添加：根据需求选择认证方式；
  - 类型：用户名和密码；
  - 用户名：输入github的用户名；
  - 密码：github密码；(不一定有用，github可能只允许token访问，需要去github用户头像点击setting=>developer settings => personal access tokens，如有原来就有直接取，或者generate new token 生成新的token，note写该token的名字，选择有效期，选择权限，repo可选对所有仓库都有权访问)
  - 复制token，填充到密码处；
  - 添加成功选择一个认证；
- 填写github中的分支名；



### 2.2 构建触发器：

- 选择一个来进行构建，根据实际需要；
- 定时构建和轮询构建区别是，定时构建指定时间之后必定构建，轮询会对比下代码是否有变化，再决定是否构建；
- 考虑到性能，我们可以选择定时构建；
- 选择定时构建，会显示日程表让我们填：可以填五个字符，分别代表**分****时****天****月****周**，如果填五个\*号表示每月每周每天每时每分都构建；
- 先保存；

这里的触发器规则是这样的：

- 定时字符串从左往右分别是：分 时 日 月 周

#每半小时构建一次OR每半小时检查一次远程代码分支，有更新则构建  
H/30 \* \* \* \*



```
#每两小时构建一次OR每两小时检查一次远程代码分支，有更新则构建
H H/2 * * *

#每天凌晨两点定时构建
H 2 * * *

#每月15号执行构建
H H 15 * *

#工作日，上午9点整执行
H 9 * * 1-5

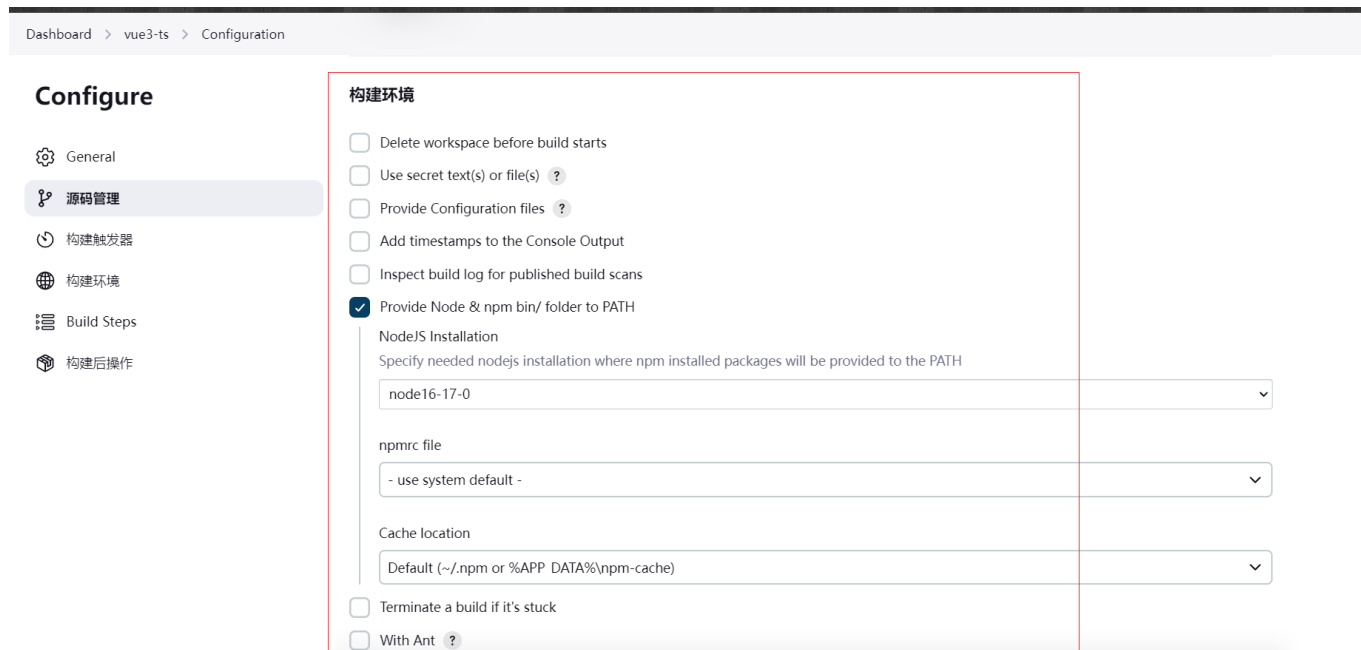
#每周1,3,5，从8:30开始，截止19:30，每4小时30分构建一次
H/30 8-20/4 * * 1,3,5
```



2.3 构建环境:

我们在构建过程执行脚本命令是，如npm run build等，需要基于node环境，因此我们此处得安装node。  
过程描述:

- 回到首页，点击manage jenkins（系统管理）；
- 选择manage plugins；
- 点击插件管理，选择可选插件，也可直接查询，输入NodeJS，勾选，并点击下方下载安装和启动按钮，成功后重启jenkins；
- 重启后重新进入，选择系统管理的全局工具配置，往下翻会有NodeJS 安装，点击新增NodeJS，取别名，可根据实际需求来，最好对应开发node版本好标识，比如此处可写node16-17-0，然后下方选择安装的node16-17-0，勾选自动安装，保存；
- 重新回到刚才新建的任务配置，来到构建环境模块；
- 此处会就会多出一个provide node & npm bin/folder to PATH选项，勾选，选中刚才安装的node16-17-0；
- 该步构建环境执行完毕；



## 2.4 构建步骤(Build Steps)

- 点击增加构建步骤;
- 选择执行shell;
- 需要输入命令框;

构建执行的任务(可根据需求自行设计):

- 查看当前路径 (默认是在root目录下为该任务名称创建文件夹)、Node的版本等是否有问题;
- 执行 `npm install` 安装项目的依赖;
- 移除原来mall\_cms文件的所有内容;
- 将打包的dist文件夹内容移动到mall\_cms文件夹;

```
pwd
node -v
npm -v

npm install
npm run build

pwd

echo '构建成功'

ls

删除/root/mall_cms文件夹里所有的内容(原来的数据全删除)
rm -rf /root/mall_cms/*

重新把打包的dist文件夹拷贝到指定目录下(也可使用mv移动)
cp -rf ./dist/* /root/mall_cms/
```

## 2.5 构建项目

- 上述完成后点击保存；
- 返回该任务，点击左侧立即构建（会直接构建，如果设置了构建触发器，则会默认按规则自动构建）
- 下面就会自动开始构建，可以进入到下面的构建任务中，进入到**控制台输出**，查看构建过程日志信息；

## 5、pm2部署node项目

### 5.1 安装

刚才我们是通过终端启动的node程序，那么如果终端被关闭掉了呢？

- 那么这个时候相当于启动的Node进程会被关闭掉；
- 我们将无法继续访问服务器；

在真实的部署过程中，我们会使用一个工具pm2来管理Node的进程：

- PM2是一个Node的进程管理器；
- 我们可以使用它来管理Node的后台进程；
- 这样在关闭终端时，Node进程会继续执行，那么服务器就可以继续为前端提供服务了； 安装pm2：

```
npm install pm2 -g
```

pm2常用的命令：

```
命名进程
pm2 start app.js --name my-api
显示所有进程状态
pm2 list
停止指定的进程
pm2 stop 0
停止所有进程
pm2 stop all
重启所有进程
pm2 restart all
重启指定的进程
pm2 restart 0
杀死指定的进程
pm2 delete 0
杀死全部进程
pm2 delete all
后台运行pm2，启动4个app.js，实现负载均衡
pm2 start app.js -i 4
```

一般我们可以这么执行：

```
查看所有pm2托管的项目
pm2 list
```

```
新增需要托管的项目
pm2 start 路径文件 -n 自定义的名字

杀死某个进程
pm2 delete 之前自定义的名字

查看pm2的日志
pm2 logs
```

## 5.2 防火墙打开（很重要）

上面我们使用pm2启动了执行目录下的node项目，但是我们在本地postman测试node项目对应的接口时会连接失败。

究其原因是由于防火墙关闭了，所有的外部交流都会阻断，因此需要我们手动打开防火墙。

```
启动防火墙
systemctl start firewalld
关闭防火墙
systemctl stop firewalld
查看状态
systemctl status firewalld
开机禁用
systemctl disable firewalld
开机启用
systemctl enable firewalld
```

**注意必须打开防火墙！！！！**

## 5.3 打开防火墙端口（很重要）

上面我们虽然打开了防火墙，但是这个时候用postman测试云服务器上的node项目接口时，此时连接服务器不会失败，但是所有接口都会超时。

究其原因还是防火墙拦截了，我们需要手动指定防火墙放开拦截的端口，即我们node项目运行的端口，让外部可以访问该端口。

注意！到这里为止，我们依旧需要保证node项目启动的服务器的端口在安全组里面有配置！！

### 1、添加防火墙端口

```
防火墙添加端口
firewall-cmd --zone=public --add-port=端口/tcp --permanent

命令含义：
-zone #作用域
-add-port=1935/tcp #添加端口，格式为：端口/通讯协议
-permanent #永久生效，没有此参数重启后失效
```

## 2、重新载入，添加端口后重新载入才能起作用

```
firewall-cmd --reload
```

## 3、查看端口是否开启

```
firewall-cmd --zone=public --query-port=端口/tcp
```

参考文章：

<https://www.cnblogs.com/AJun816/p/17301515.html#centos%E5%BC%80%E5%90%AF%E7%AB%AF%E5%8F%A3>

## 5.4 关于打开防火墙后，80端口静态资源部署也要打开端口

前面nginx配置了静态资源，可以直接在浏览器80端口(http)访问，但是当我们打开防火墙后，再次访问就失败了。

此时我们也需要手动给防火墙设置暴露80端口，操作同上，端口改成80即可。

## 5.5 查看端口监听程序

```
netstat -ntlp

-n/-numeric 以数值地址形式进行展示
-t/-tcp 显示 TCP 协议的连接情况
-l/-listening 仅显示监听中的套接字
-p/-program 显示套接字关联的程序名称和 PID
```

## 6、MongoDB的安装

参考文章：[https://blog.csdn.net/qq\\_43511063/article/details/126565777](https://blog.csdn.net/qq_43511063/article/details/126565777)

### 6.1 创建 mongodb yum 源头

```
创建源头文件（也可以用xftp或者winSCP等可视化软件进行下面操作）
下面写的5.0，实际可根据需求设置版本
进入目录
cd /etc/yum.repos.d
创建文件
touch mongodb-org-5.0.repo
编辑文件
vim mongodb-org-5.0.repo
写入内容
```

```
[mongodb-org-5.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/5.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-5.0.asc
```

## 6.2 yum安装和启动

```
安装
sudo yum install -y mongodb-org

启动 mongodb
sudo systemctl start mongod

查看 mongod 状态
sudo systemctl status mongod

设置开机启动
sudo systemctl enable mongod

关闭 mongod (有需求时再执行)
sudo systemctl stop mongod

重启 (有需求时再执行)
sudo systemctl restart mongod
```

## 6.3 端口的配置

云服务器中安装MongoDB后，默认的端口是**27017**，默认绑定IP为 127.0.0.1，这就导致外部无法访问，因此要进入 `/etc/mongod.conf` 文件中修改**bindIP**为0.0.0.0，绑定公网IP。

- 配置安全组：27017。
- 修改bindIP值：

```
使用cat查看原bindIp (可以看到是127.0.0.1)
cat /etc/mongod.conf

修改成0.0.0.0 (把bindIp改成0.0.0.0)
vim /etc/mongod.conf

重启
systemctl restart mongod
```

## 6.4 本地连接远程数据库

使用本地的**MongoDB Compass**软件连接远程数据库，在连接之前需要保证：

- 云服务器安全组配置了mongodb默认端口27017（如果在/etc/mongodb.conf中调整过端口，则需放开对应端口）；
- 保证防火墙放开了该端口：

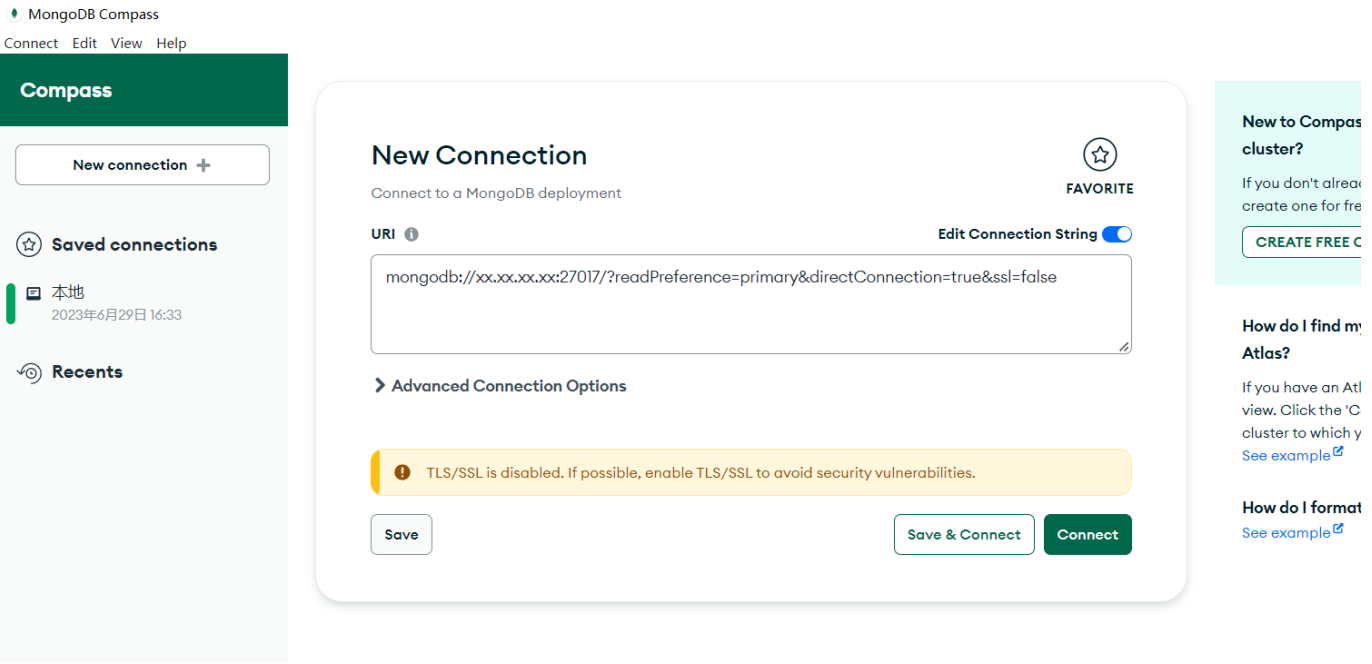
○

```
firewall-cmd --zone=public --add-port=27017/tcp --permanent

firewall-cmd --reload
```

打开**MongoDB Compass**软件新建连接：

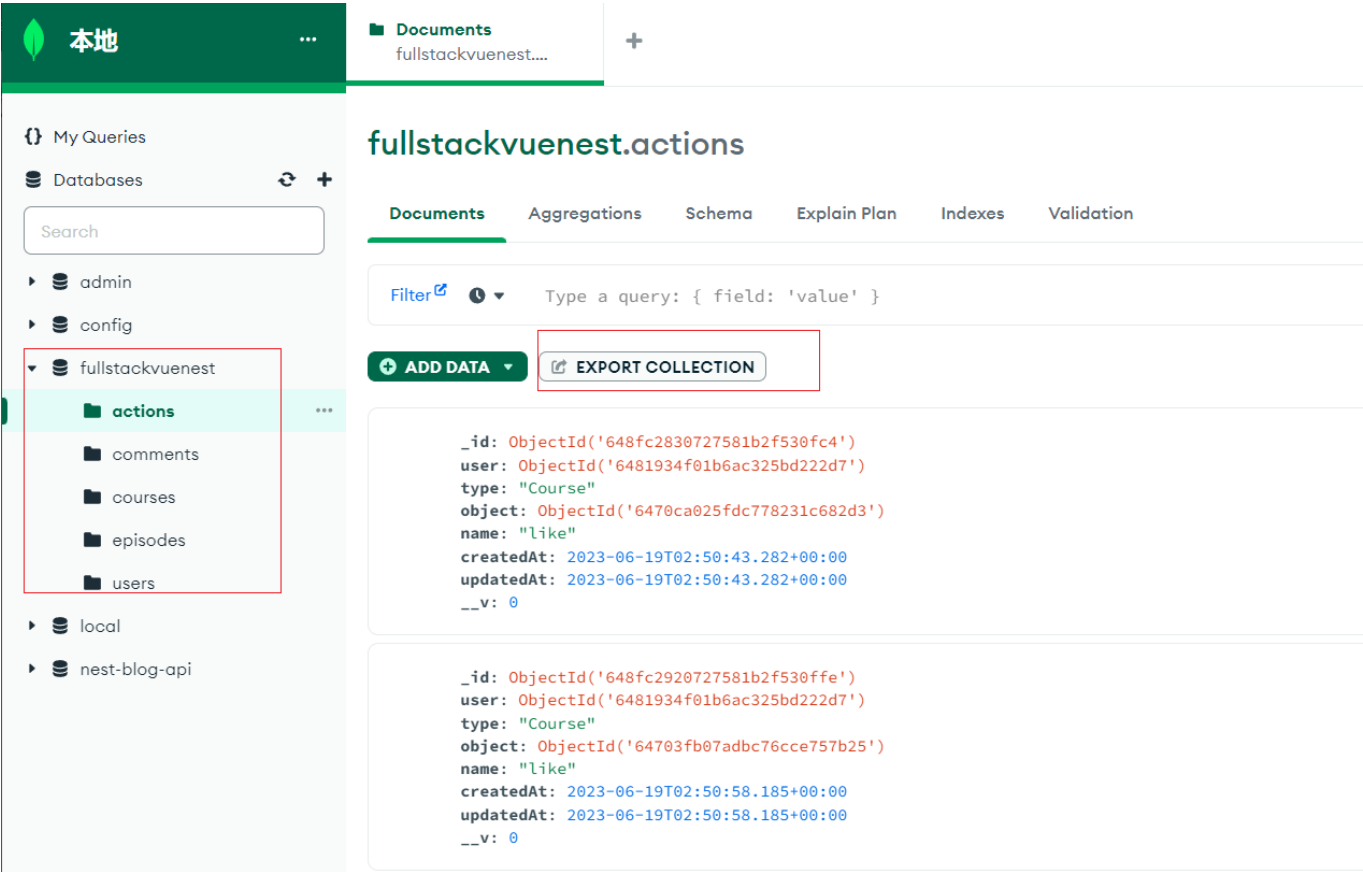
```
mongodb://xx.xx.xx.xx:27017/?
readPreference=primary&directConnection=true&ssl=false
```



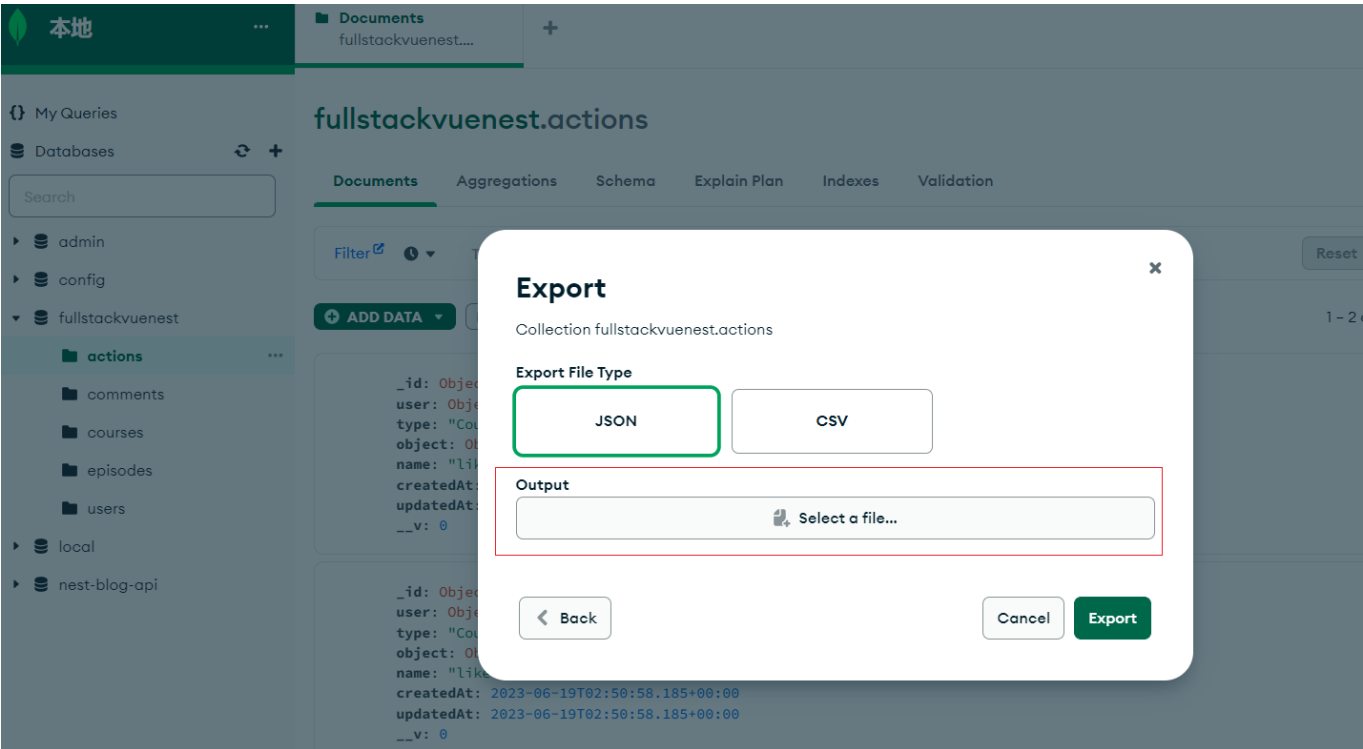
6.5 本地数据同步到远程

1.本地导出：

找到指定的数据库，选择对应的表集合，点击导出：



选择导出的条件/全部导出，并勾选导出字段，选择导出路径：

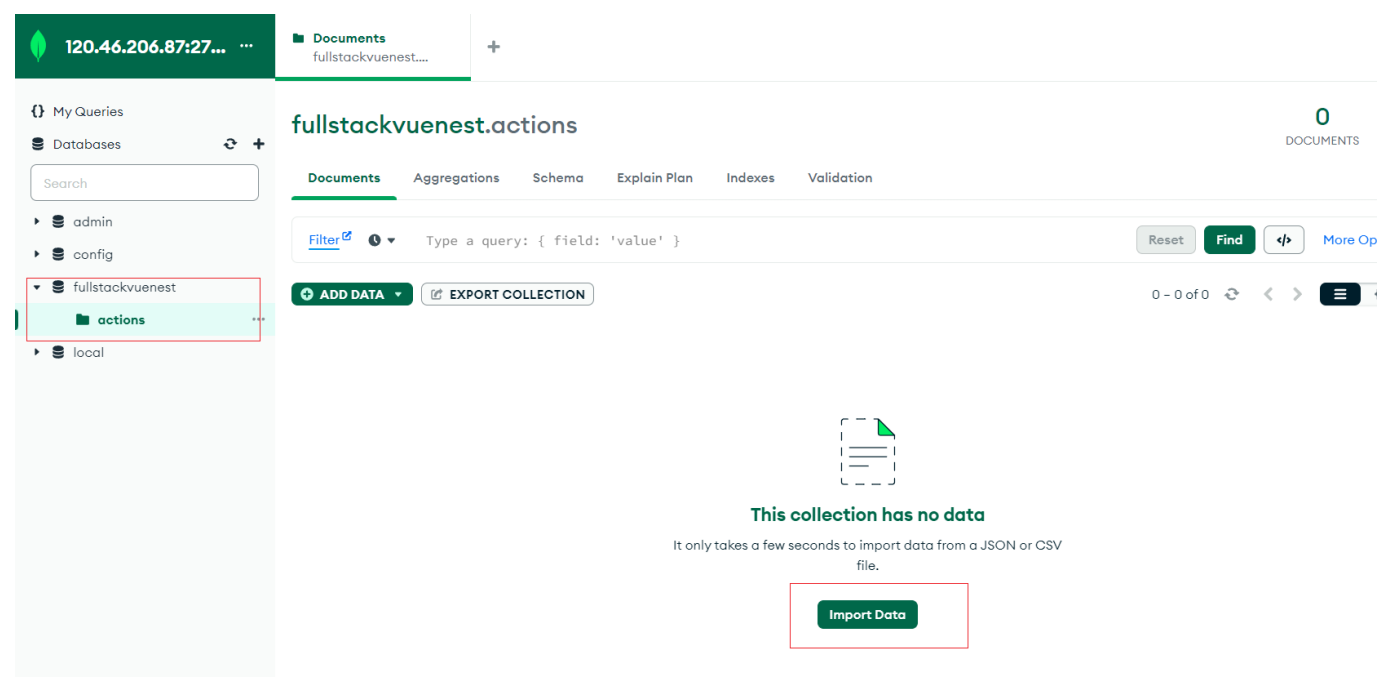


确定后，指定路径下就会有默认的json文件。

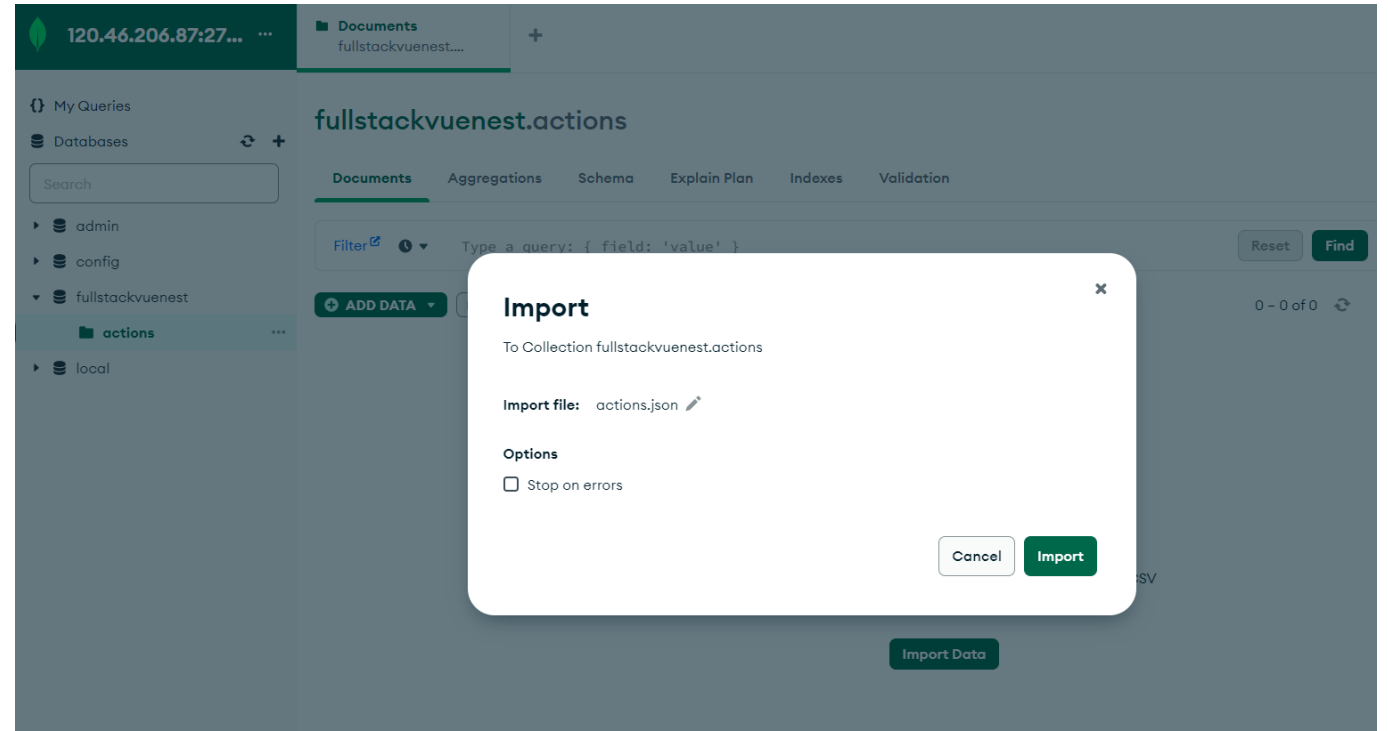
2.远程导入

手动建库，并新增指定的数据集合表，点击导入：





导入文件：



6.6 开启密码安全连接

当我们按照上面的方式远程连接数据库后，并且成功导入了本地的数据库数据，看起来已经完成了。

但是，可能过几天来看的时候，你再次连接进入，会发现很可能你之前新建的数据库和表数据消失了，因为你被别人黑掉了，并删除了你数据库了。

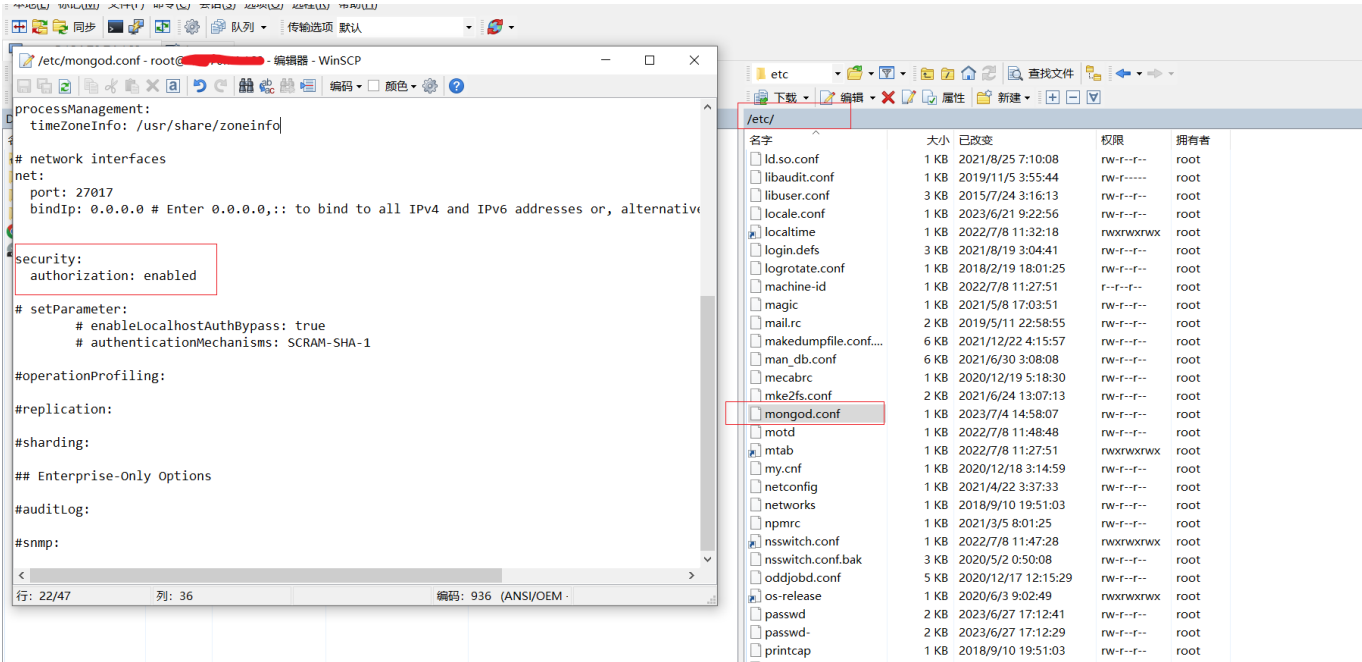
所以，我们需要开启安全连接校验，在远程连接数据库时需要验证数据库用户表的用户信息，我们也可以修改默认的端口号。

1. 修改mongod.conf配置：

找到etc内的mongod.conf文件，并写入下面代码：

```
开启安全校验
security:
 authorization: enabled

设置校验密码类型：SCRAM-SHA-1，注意此处可以不设置，在远程连接时就default，否则按指定的
类型进行连接
setParameter:
 enableLocalhostAuthBypass: true
 authenticationMechanisms: SCRAM-SHA-1
```



2. 重启mongod

```
重启
systemctl restart mongod

查看
systemctl status mongod
```

3. 新增数据库用户

```
1.终端输入mongosh进入mongodb环境
mongosh

2.查看所有数据库(在mongodb新版本里并没有admin数据库，但是并不妨碍第3步操作。)
show dbs

3.进入admin数据库
use admin

4.创建管理员账户(这个admin自定义账户就是等会远程连接需要用的账户)
mongodb中的用户是基于身份role的，该管理员账户的 role是 userAdminAnyDatabase。admin
用户用于管理账号，不能进行关闭数据库等操作。
```

```
user和pwd自行定义
db.createUser({ user: "admin", pwd: "password", roles: [{ role:
"userAdminAnyDatabase", db: "admin" }] })
5.创建root(删除数据库时需要切换到role为root的账户)
db.createUser({user: "root",pwd: "password", roles: [{ role: "root", db: "admin"
}]})

补充: 关于role角色的说明
Read: 允许用户读取指定数据库,
readWrite: 允许用户读写指定数据库
dbAdmin: 允许用户在指定数据库中执行管理函数, 如索引创建、删除, 查看统计或访问
system.profile
userAdmin: 允许用户向system.users集合写入, 可以在指定数据库里创建、删除和管理用户
clusterAdmin: 只在admin数据库中可用, 赋予用户所有分片和复制集相关函数的管理权
限。
readAnyDatabase: 只在admin数据库中可用, 赋予用户所有数据库的读权限
readWriteAnyDatabase: 只在admin数据库中可用, 赋予用户所有数据库的读写权限
userAdminAnyDatabase: 只在admin数据库中可用, 赋予用户所有数据库的userAdmin权限
dbAdminAnyDatabase: 只在admin数据库中可用, 赋予用户所有数据库的dbAdmin权限。
root: 只在admin数据库中可用。超级账号, 超级权限
```

#### 4.查看用户表

注意在mongo环境中（输入了mongosh）：

```
查看数据表
show dbs
上面可能会报错: MongoServerError: command listDatabases requires authentication
是因为没有权限, 我们上面虽然创建了用户, 但是没有用该用户认证登录

认证登录(成功的话终端返回{ok: 1})
db.auth("admin","password")

查看用户
show users
返回值如下:
[
 {
 _id: 'admin.admin',
 userId: new UUID("3abb8a14-20ba-4c0f-8829-feae65d98eb6"),
 user: 'admin',
 db: 'admin',
 roles: [{ role: 'userAdminAnyDatabase', db: 'admin' }],
 mechanisms: ['SCRAM-SHA-1', 'SCRAM-SHA-256']
 }
]

如果要删除某数据库, 需先切换到role为root的用户, 假设要删除record数据库
db.auth('root', 'password') # 用户授权切换
use record # 切到指定数据库
db.dropDatabase() # 回收该数据库
```

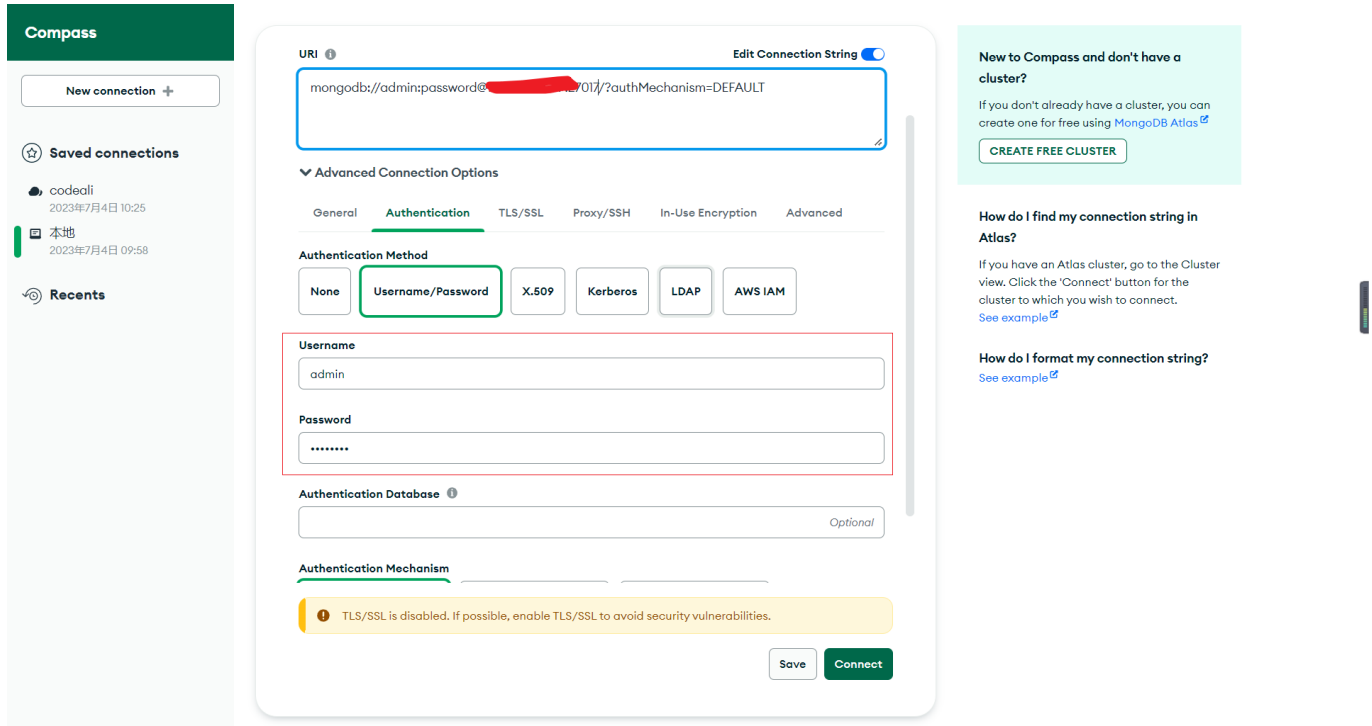
参考文章地址：

- <https://blog.csdn.net/wdw984/article/details/126154333>
- <https://www.yingsoo.com/news/database/48133.html>

#### 4. 重新MongoDB Compass远程连接

选择**Authentication**，并输入用户名和密码，如果mongod.conf有配置密码校验类型则选择对应的Mechanism，否则默认Default。

也可以看上面show users显示的用户mechanisms。



#### 6.7 部分异常处理

##### 1. 报错\$OPTIONS (code=exited, status=14):

出现错误状态为 status=14,那里主要原因就是文件权限问题，用户 mongod 没有对必需文件的写权限，导致数据库服务不能启动

首先查看数据存储目录、日志目录，和 /tmp 下的\*.sock(是通配符)的文件这三个目录或文件的执行权限，可通过 /etc 目录下的 mongod.conf 配置文件查看目录的具体位置。

1、数据存储目录 (dbpath) 的位置，该目录默认在 var/lib/mongo 下，执行如下命令修改目录权限(-R 递归处理所有文件和文件夹)

```
chown -R mongod:mongod /var/lib/mongo
```

2、再修改日志目录的权限，该文件默认在var/log/mongodb目录，命令如下

```
chown -R mongod:mongod /var/log/mongod
```

3、最后还有一个文件需要开放用户mongod的写权限，该文件叫\*.sock(是通配符)，在/tmp路径下。执行如下命令

代码中的\*为通配符，表示合法文件名

```
chown mongod:mongod /tmp/*.sock
```

接着重新执行：

启动MongoDB服务

```
systemctl start mongod
```

查看MongoDB服务状态

```
systemctl status mongod
```

重启mongo服务

```
systemctl restart mongod
```

参考文章地址：[https://blog.csdn.net/ATRI\\_/article/details/127805441](https://blog.csdn.net/ATRI_/article/details/127805441)