## 搭建minio服务端

### 环境说明

服务端搭建在Linux环境，使用docker进行搭建。本机使用Linux虚拟机，版本为Centors7.

### 搭建步骤

#### <1> docker环境安装

1、确保linux系统内核在3.10以上，使用

uname -r

命令查看版本信息，如图

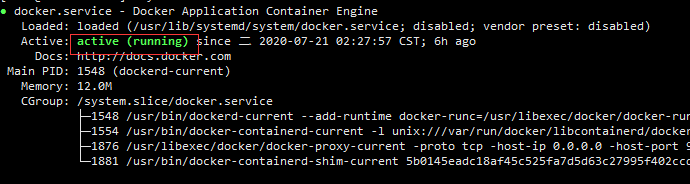
IMG_256；

2、使用 yum -y install docker 命令安装docker；

3、docker安装完成后使用

systemctl start docker 启动docker命令

systemctl status docker 查看docker服务状态



如上图所示，说明docker服务搭建成功

#### <2> minio服务搭建

1. 使用 docker pull minio/minio 命令获取minio镜像；

2、minio安装完成后，使用 docker swarm init 创建集群， 管理docker；

3、使用如下命令搭建minio服务端

docker run -p 9000:9000 --name minio -e "MINIO\_ACCESS\_KEY=minio" -e "MINIO\_SECRET\_KEY=minio123" -v /mnt/data:/data -v /mnt/config:/root/.minio minio/minio server /data

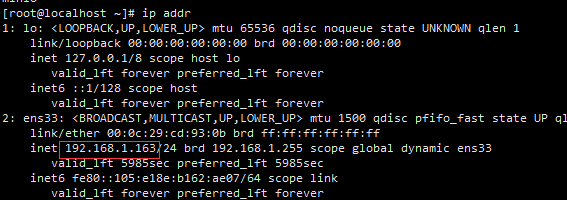
注：--name 后是容器的名字

MINIO\_ACCESS\_KEY和MINIO\_SECRET\_KEY对应账号和密码

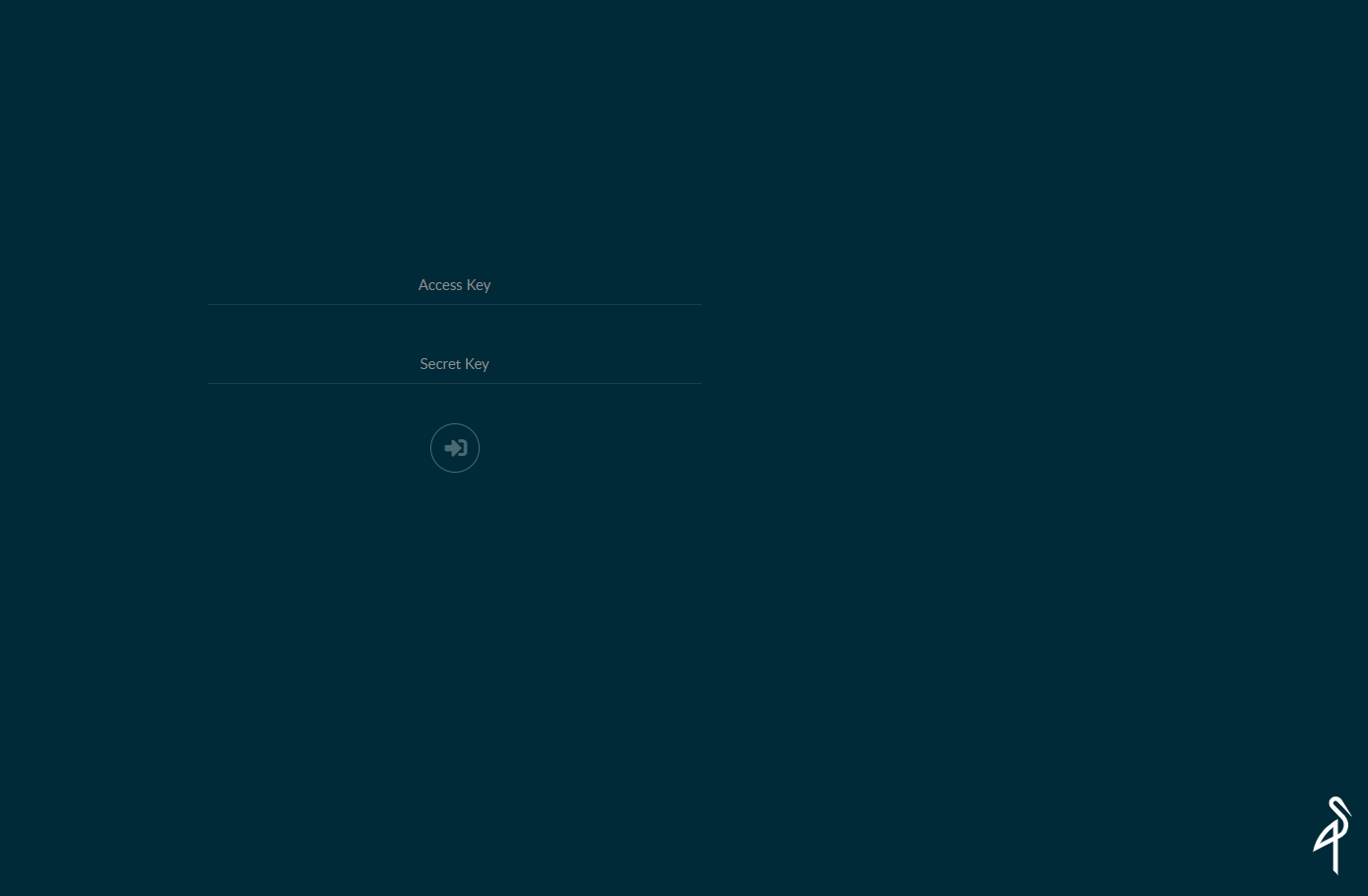
-p 9000:9000 处理端口映射

4、使用Linux的ipv4地址进行登录。首先使用 ip addr

命令查看ipv4地址，如图



，然后使用 [http://192.168.1.163:9000](http://192.168.174.129:9000) 访问minio文件 服务。如下图



### 3、遇到异常处理

Error response from daemon: Conflict. The container name "/minio" is already in use by container

说明“minio”容器已存在，可以创建其他名字的容器。如果“minio”容器无用 则可以使用

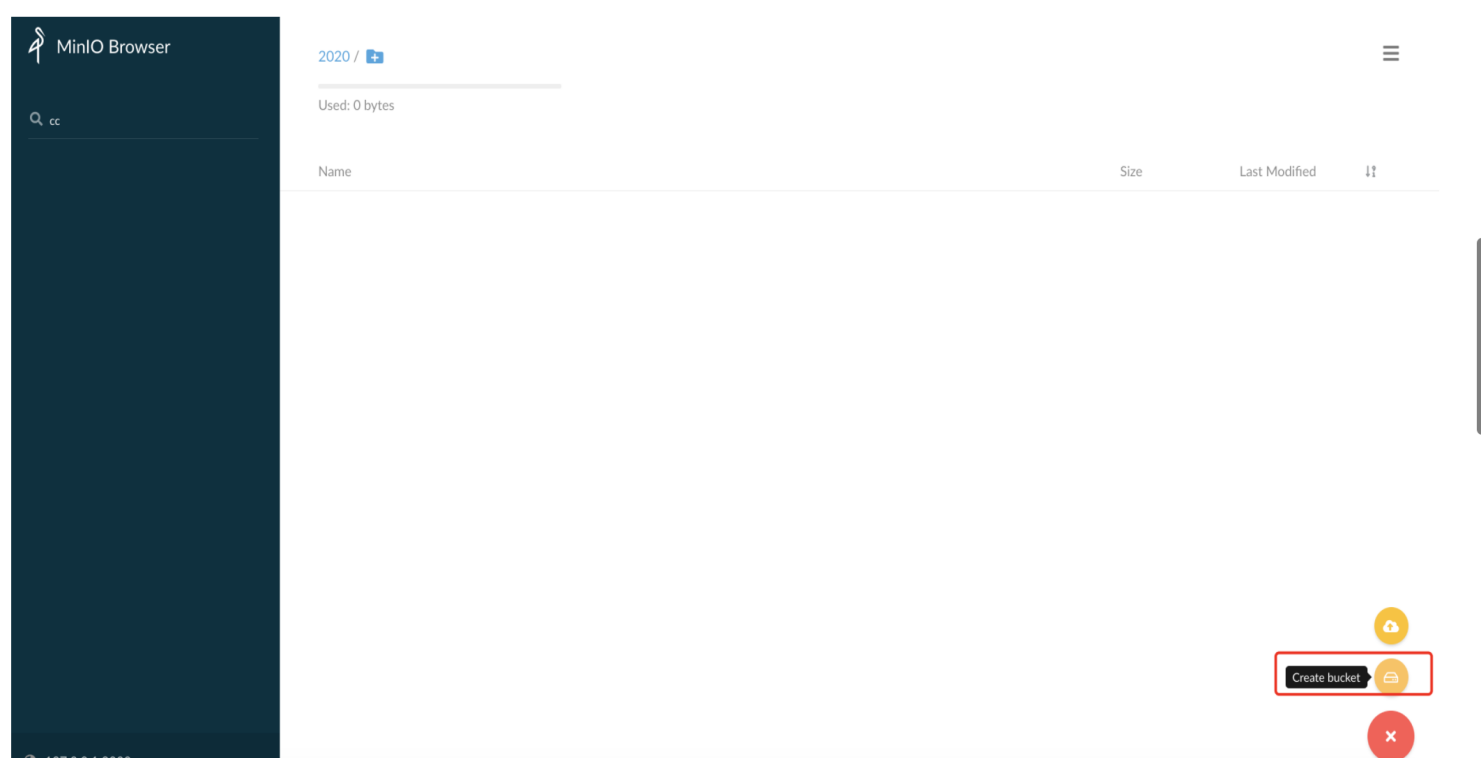
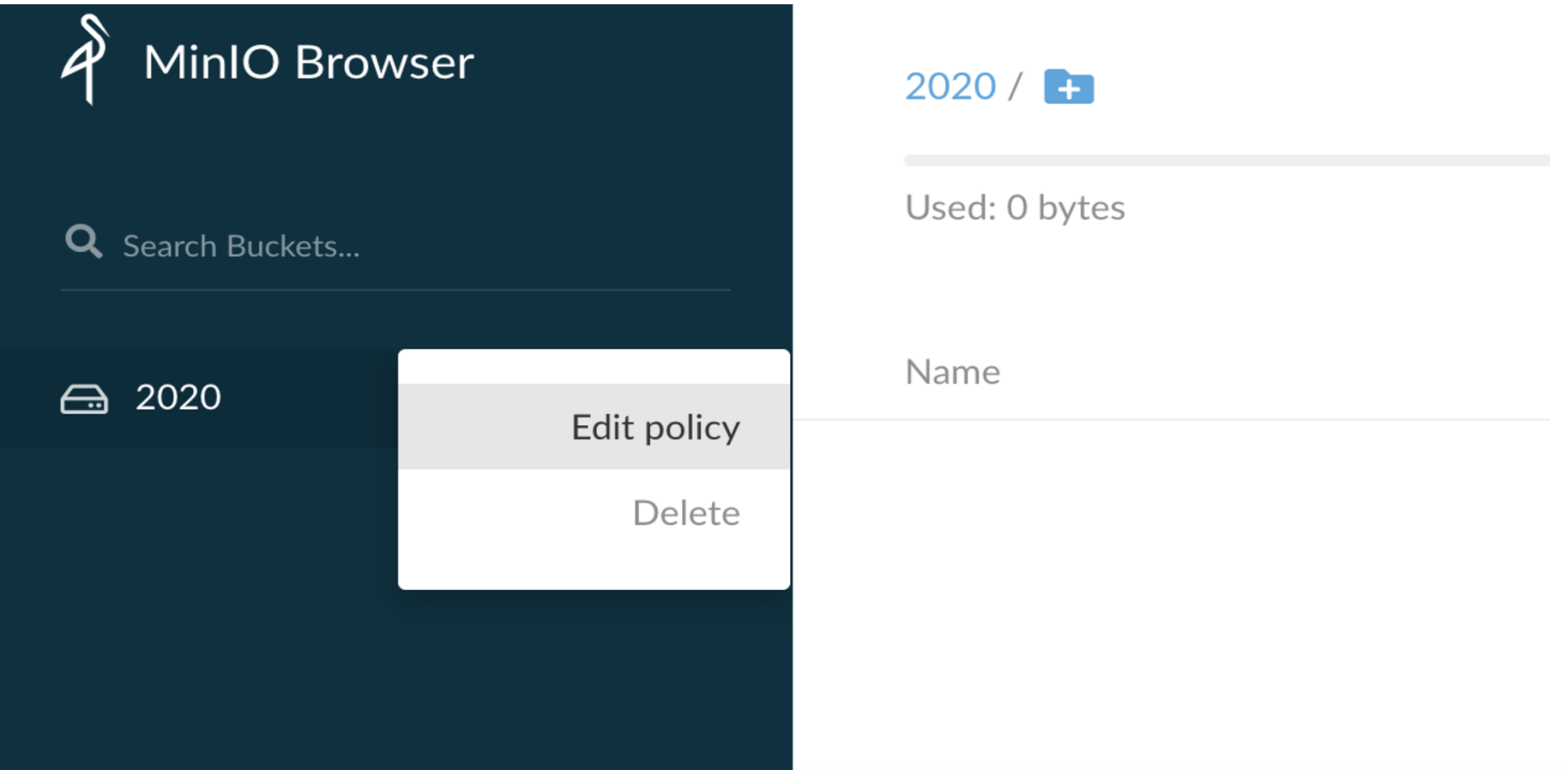
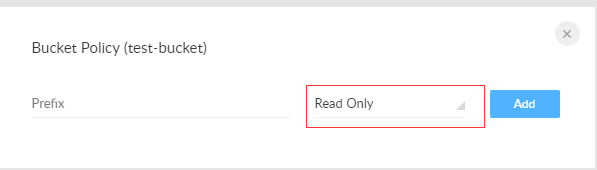
docker stop minio 和 docker rm minio

清除容器

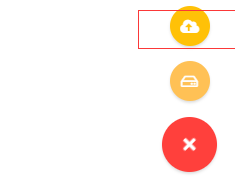
使用 docker ps -a 命令可以查看存在的容器

### 4、Minio服务端使用介绍

1、首先创建桶(bucket)，所有文件都是存在对应桶中的；

2、可以对桶设置读写权限  
  


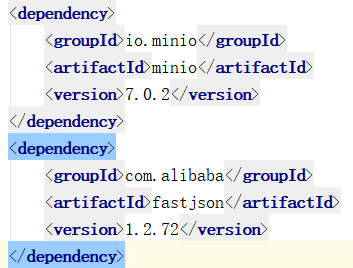
3、选择桶，并上传文件



## springboot集成minio

### maven依赖说明

除了springboot自生成的依赖外，还需压如下图依赖



### 环境配置



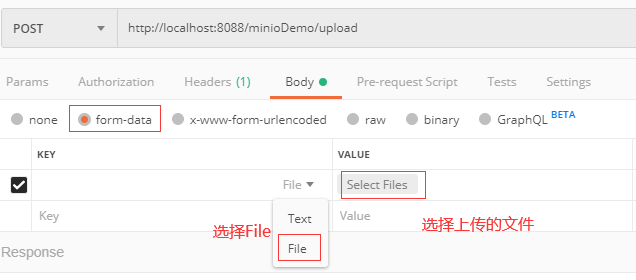
### 代码实现上传功能及测试

#### <1> 代码



#### <2> 测试

这里以postman工具进行测试，如下图



Postman发送请求后。登录minio服务器，可以看到对应桶下生成了当前日期的文件夹，如图；



文件夹里是刚刚上传的文件，如图。



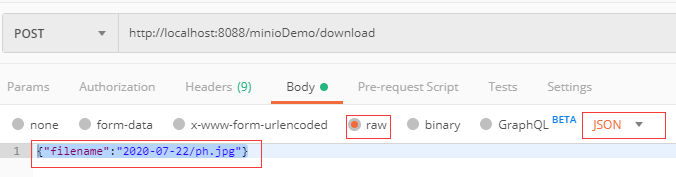
### 代码实现下载功能及测试

#### <1> 代码



#### <2> 测试

这里以postman工具进行测试，如下图



Postman发送请求后。查看请求结果

IMG_256

说明下载成功。

### 代码附录

1、pom.xml

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
 <**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.3.1.RELEASE</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  
 <**groupId**>com.gc.springboot\_minio</**groupId**>  
 <**artifactId**>demo</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 <**name**>demo</**name**>  
 <**description**>Demo project for Spring Boot</**description**>  
  
 <**properties**>  
 <**java.version**>1.8</**java.version**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>io.minio</**groupId**>  
 <**artifactId**>minio</**artifactId**>  
 <**version**>7.0.2</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>fastjson</**artifactId**>  
 <**version**>1.2.72</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 <**exclusions**>  
 <**exclusion**>  
 <**groupId**>org.junit.vintage</**groupId**>  
 <**artifactId**>junit-vintage-engine</**artifactId**>  
 </**exclusion**>  
 </**exclusions**>  
 </**dependency**>  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
</**project**>

2、application.yml

server:  
 port: 8088  
  
spring:  
 servlet:  
 multipart:  
 enabled: true #开启文件上传  
 max-file-size: 500MB  
 max-request-size: 500MB  
  
minio:  
 endpoint: http://192.168.1.163:9000 #Minio服务所在地址  
 bucketName: test-bucket #存储桶名称  
 accessKey: minio #访问的key  
 secretKey: minio123 #访问的秘钥

3、java代码

**import** com.alibaba.fastjson.JSONObject;  
**import** io.minio.MinioClient;  
**import** io.minio.PutObjectOptions;  
**import** io.minio.Result;  
**import** io.minio.messages.Item;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.beans.factory.annotation.Value;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.web.bind.annotation.\*;  
**import** org.springframework.web.multipart.MultipartFile;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.BufferedInputStream;  
**import** java.io.InputStream;  
**import** java.io.OutputStream;  
**import** java.net.URLEncoder;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** java.util.Iterator;  
  
@RestController  
@RequestMapping(**"/minioDemo"**)  
**public class** MinioDemoController {  
  
 **private static final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(MinioDemoController.**class**);  
  
 @Value(**"${minio.endpoint}"**)  
 **private** String **ENDPOINT**;  
 @Value(**"${minio.bucketName}"**)  
 **private** String **BUCKETNAME**;  
 @Value(**"${minio.accessKey}"**)  
 **private** String **ACCESSKEY**;  
 @Value(**"${minio.secretKey}"**)  
 **private** String **SECRETKEY**;  
  
 *//文件创建* @RequestMapping(value = **"/upload"**)  
 **public** String upload(MultipartFile file) {  
 String s=**null**;  
 **try** {  
 MinioClient minioClient = **new** MinioClient(**ENDPOINT**, **ACCESSKEY**, **SECRETKEY**);  
 *//存入bucket不存在则创建* **if** (!minioClient.bucketExists(**BUCKETNAME**)) {  
 minioClient.makeBucket(**BUCKETNAME**);  
 }  
 String filename = file.getOriginalFilename();  
 SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  
 *// 文件存储的目录结构* String objectName = sdf.format(**new** Date()) + **"/"** + filename;  
 *// 存储文件* minioClient.putObject(**BUCKETNAME**, objectName, file.getInputStream(),  
 **new** PutObjectOptions(file.getInputStream().available(), -1));  
 ***LOGGER***.info(**"文件上传成功!"**);  
 s=**ENDPOINT** + **"/"** + **BUCKETNAME** + **"/"** + objectName;  
 } **catch** (Exception e) {  
 ***LOGGER***.info(**"上传发生错误: {}！"**, e.getMessage());  
 e.printStackTrace();  
 }  
 **return** s;  
 }  
  
 @RequestMapping(value = **"/download"**)  
 **public void** downloadFiles(@RequestBody JSONObject param, HttpServletResponse httpResponse) {  
 **try** {  
 String filename = param.getString(**"filename"**);  
 MinioClient minioClient = **new** MinioClient(**ENDPOINT**, **ACCESSKEY**, **SECRETKEY**);  
 InputStream object = minioClient.getObject(**BUCKETNAME**, filename);  
 **byte** buf[] = **new byte**[1024];  
 **int** length = 0;  
 httpResponse.reset();  
  
 httpResponse.setHeader(**"Content-Disposition"**, **"attachment;filename="** + URLEncoder.*encode*(filename, **"UTF-8"**));  
 httpResponse.setContentType(**"application/octet-stream"**);  
 httpResponse.setCharacterEncoding(**"utf-8"**);  
 OutputStream outputStream = httpResponse.getOutputStream();  
 **while** ((length = object.read(buf)) > 0) {  
 outputStream.write(buf, 0, length);  
 }  
 outputStream.close();  
 } **catch** (Exception ex) {  
 ***LOGGER***.info(**"导出失败："**, ex.getMessage());  
 }  
 }

}