对象存储服务MinIO的基本用法

文章分类：JavaDemo； 标签：JavaCodeSnippet； 作者：Hackyle；

更新时间：Thu Jan 05 11:11:42 CST 2023

本文重要内容

- 主要介绍MinIO与SpringBoot项目整合时的基本用法，没有涉及较多的原理剖析，更注重于应用实践（功能实现）。
- 与SpringBoot整合的完整实例（代码可直接复用）：**https://github.com/HackyleShawe/JavaDemos/tree/master/Examples/minio-demo**

背景

- 在企业级开发中，会有许多类别的服务器，各个服务器专注于某一项领域
  - 应用服务器：专门部署、运行我们的应用
  - 数据库服务器：专门运行我们的数据库
  - 文件服务器：专门负责存储、获取用户上传文件的服务器
- 在我自己写博客系统时，形如图片、视频、音乐等静态资源文件的存储是个问题。不可能将其存放在SpringBoot项目内，事实上也没办法存储在SpringBoot项目里。那有没有一种专门提供文件存取的技术或服务呢？答案是肯定的
  - 对象存储解决方案：MinIO —— 运行一个minio服务，可实现文件的获取功能，通过其提供的一套SDK可实现文件的上传功能
  - 云对象存储：七牛云、青云、阿里云 —— 收费，其他功能类似于MinIO
- 所以，有了本文，介绍MinIO整合SpringBoot，实现文件的上传、在线查看（获取）文件的功能

内容导览

对象存储

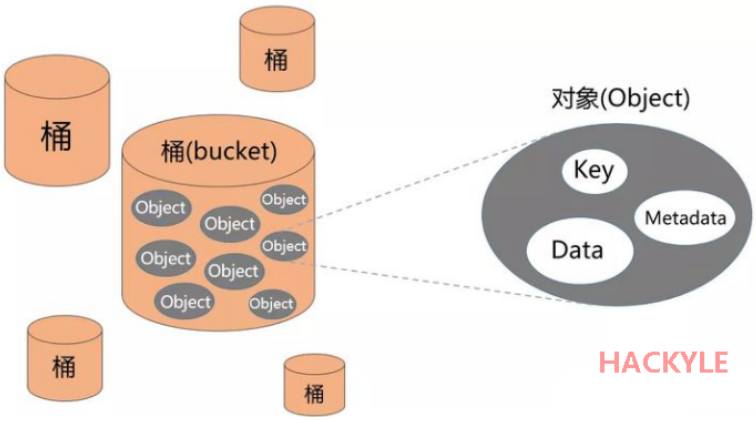
对象存储（Object Storage）

- 文件以网络资源的形式存在（可以理解为JavaWeb中的Api接口所指代的资源），不能直接打开/修改文件，只能先下载、修改，再上传文件（如网盘、FTP）
- 对象存储，主要操作对象是文件对象（File Object）

对象存储的基本思想：桶+文件Key

- 桶：是一个逻辑的概念，文件就存放在该个桶中。可以理解为一个父文件夹，文件就存放在该个父文件夹下
- 文件Key：也就是文件名，它是桶中文件的唯一标识，通过它可以从桶中获取文件、删除文件等操作

TOP



对象存储呈现出来的是一个“桶”（**bucket**），你可以往“桶”里面放“对象（**Object**）”

Key

- 可以理解文件名，是该对象的全局唯一标识符（UID）
- Key是用于检索对象，服务器和用户不需要知道数据的物理地址，也能通过它找到对象

Data

- 文件的数据本体

Metadata

- 元数据：有点类似数据的标签，标签的条目类型和数量是没有限制的，可以是对象的各种描述信息
- 如果对象是一张人物照片，那么元数据可以是姓名、性别、国籍、年龄、拍摄地点、拍摄时间等。
- 在传统的文件存储里，这类信息属于文件本身，和文件一起封装存储。而对象存储中，元数据是独立出来的，并不在数据内部封装。
- 元数据的好处非常明显，可以大大加快对象的排序，还有分类和查找。

MinIO

官网：https://min.io/

中文官网：http://www.minio.org.cn/

GitHub：https://github.com/minio/

根据对象存储的基本定义，**MinIO中没有文件夹的概念，只有桶和文件对象**

- 桶可以理解为一个根目录，例如：data-bucket
- 文件对象包含文件前缀（文件路径，/path1/path2/）和文件名（txt），例如：/path1/path2/aa.png
- 文件URI：data-bucket/path1/path2/aa.png
- 拼接上域名就可以对外访问：http://localhost:9000/data-bucket/path1/path2/aa.png

一般情况下，一种业务类型建立一个桶，可以使用文件对象的前缀来标识文件的类型。

例如，我的博客系统桶名为：**blog**，文件对象前缀为：**年份 + 月份**，最终的文件对象地址为：**/blog/2022/12/文件名.文件拓展名**，文件的完整URL为：**http://localhost:9000/blog/2022/12/文件名.文件拓展名**

环境搭建

CentOS下RPM安装

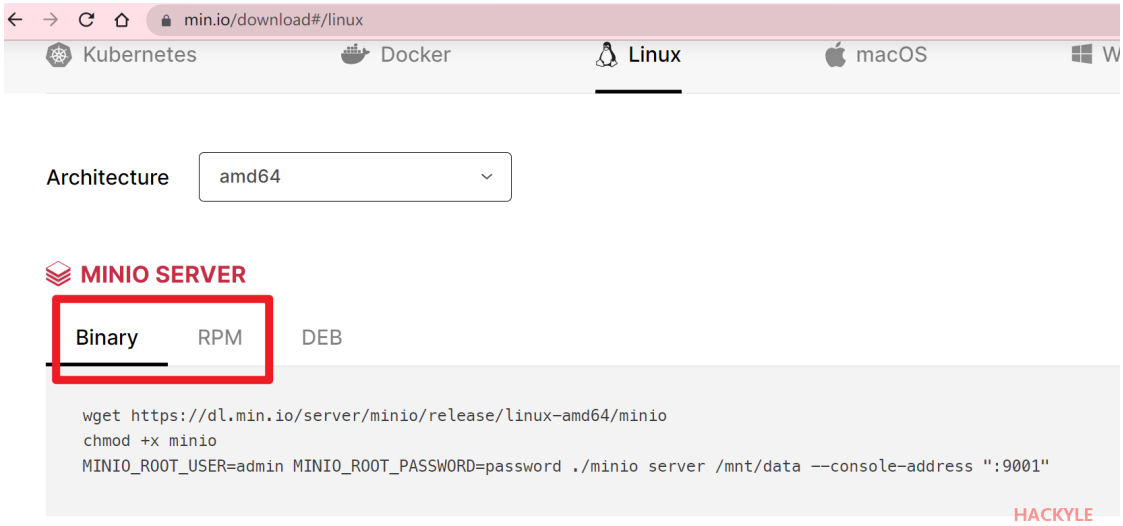
- 下载安装包：https://dl.min.io/server/minio/release/linux-amd64/minio-20220108031154.0.0.x86_64.rpm
- 执行安装：rpm -i minio-20220108031154.0.0.x86_64.rpm
- 切换至Root：su root
- 启动：**MINIO_ROOT_USER=kyle-minio MINIO_ROOT_PASSWORD=hackyle-minio minio server /var/minio-data --address ":9000" --console-address ":9001"**
  - MINIO_ROOT_USER=kyle-minio 访问MinIO时的**用户名**
  - MINIO_ROOT_PASSWORD=hackyle-minio 访问MinIO时的**密码**
  - minio server **启动**服务
  - /var/minio-data 指定**文件存储位置**
  - --address ":9000" MinIO**对外**提供服务的**端口**
  - --console-address ":9001" MinIO后台**管理端的端口**

TOP

```
[root@HackyleCent-0 Downloads]# MINIO_ROOT_USER=kyle-minio MINIO_ROOT_PASSWORD=hackyle-minio minio
 server /var/minio-data --address ":9000" --console-address ":9001"
API: http://10.100.134.170:9000  http://192.168.122.1:9000  http://127.0.0.1:9000
RootUser: kyle-minio
RootPass: hackyle-minio

Console: http://10.100.134.170:9001 http://192.168.122.1:9001 http://127.0.0.1:9001
RootUser: kyle-minio
RootPass: hackyle-minio

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc alias set myminio http://10.100.134.170:9000 kyle-minio hackyle-minio

Documentation: https://docs.min.io
```

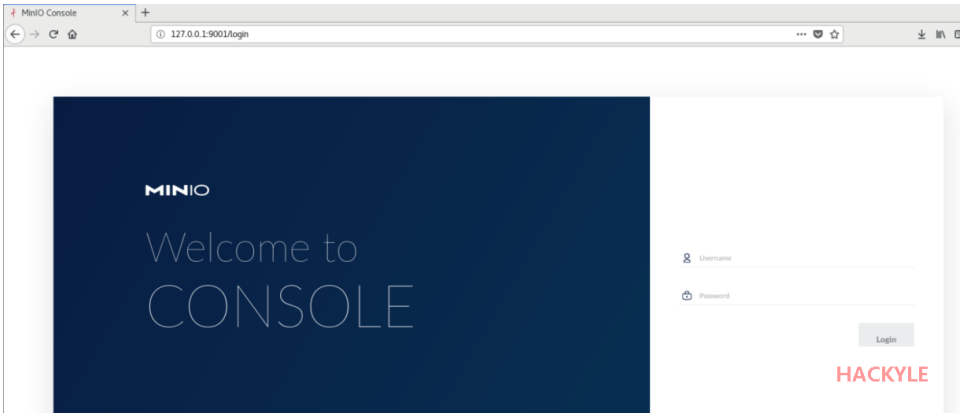
不要关闭此Terminal，否则会关闭MinIO；可以使用后台启动的方式

- 官方参考文档：https://min.io/download#/linux



CentOS二进制文件安装

- **拉取：**wget https://dl.min.io/server/minio/release/darwin-amd64/minio
- **赋予可执行权限：**chmod +x minio
- **启动：MINIO_ROOT_USER=kyle-minio MINIO_ROOT_PASSWORD=hackyle-minio minio server /var/minio-data --address ":9000" --console-address ":9001"**



进入管理界面（Console）：http://127.0.0.1:9001

Windows环境安装

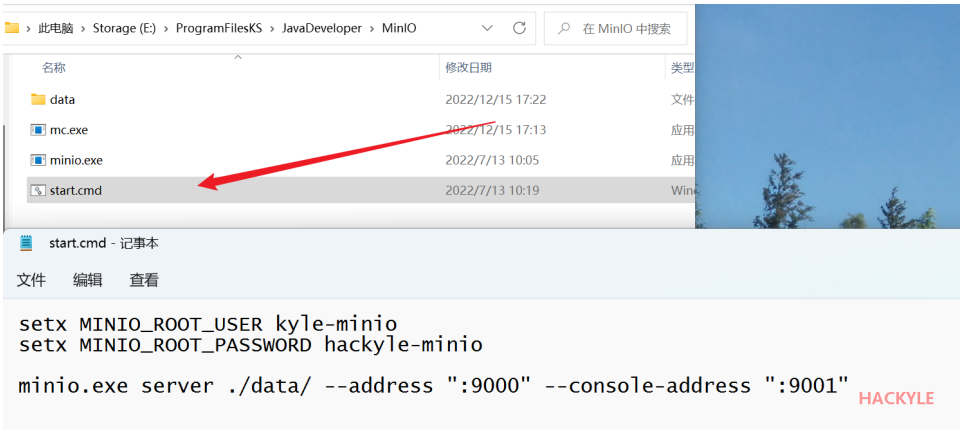
- 下载：https://min.io/download#/windows
- 进入minio所在目录，打开CMD
- 设置临时环境变量：
  1. **setx MINIO_ROOT_USER kyle-minio**
  2. **setx MINIO_ROOT_PASSWORD hackyle-minio**
- 启动：**minio.exe server E:\ProgramFiles\SystemTools\MinIO\Data --address ":9000" --console-address ":9001"**
  - E:\ProgramFiles\SystemTools\MinIO\Data 指定文件存储位置
  - --address ":9000" MinIO对外提供服务的端口
  - --console-address ":9001" MinIO后台管理端的端口
- 成功启动MinIO

TOP

```
Microsoft Windows [版本 10.0.19042.1415]
(c) Microsoft Corporation。保留所有权利。

E:\ProgramFiles\SystemTools\MinIO>setx MINIO_ROOT_USER admin

成功: 指定的值已得到保存。

E:\ProgramFiles\SystemTools\MinIO>setx MINIO_ROOT_PASSWORD hackyle

成功: 指定的值已得到保存。

E:\ProgramFiles\SystemTools\MinIO>minio.exe server E:\ProgramFiles\SystemTools\MinIO\Data --address ":9000"
 --console-address ":9001"
API: http://192.168.8.200:9000  http://192.168.137.1:9000  http://192.168.2.1:9000  http://127.0.0.1:9000
RootUser: kyle-minio
RootPass: hackyle-minio

Console: http://192.168.8.200:9001 http://192.168.137.1:9001 http://192.168.2.1:9001 http://127.0.0.1:9001
RootUser: kyle-minio
RootPass: hackyle-minio

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc.exe alias set myminio http://192.168.8.200:9000 kyle-minio hackyle-minio

Documentation: https://docs.min.io
```

HACKYLE

- 可以将启动命令写在一个cmd脚本中，后续就可以快速启动MinIO了



HACKYLE

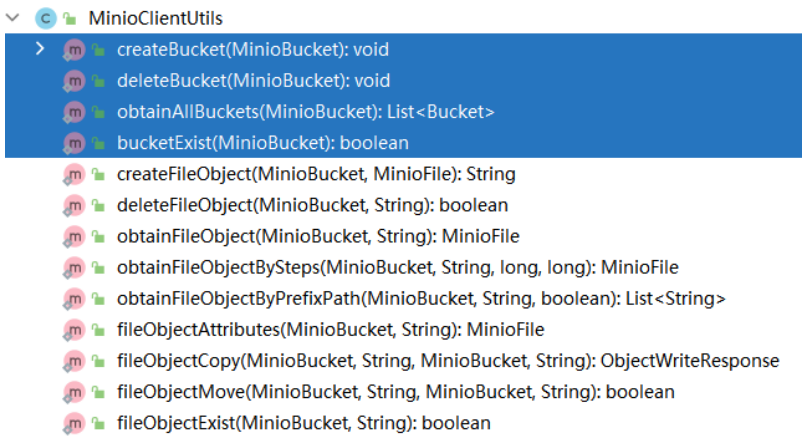
整合Spring

General Steps

1. 导入POM依赖
2. yml配置MinIO访问的URL、用户名、密码
3. 写一个配置类，读取配置文件中的参数，注入MinioClient
4. 写两个JavaBean：桶对象MinioBucket、文件对象MinioFile
5. MinioClientUtils：定义桶的创建、删除、修改、检查是否存在的操作，定义文件对象的创建、删除、获取文件信息等操作
6. 在Service层调MinioClientUtils处理桶、文件对象

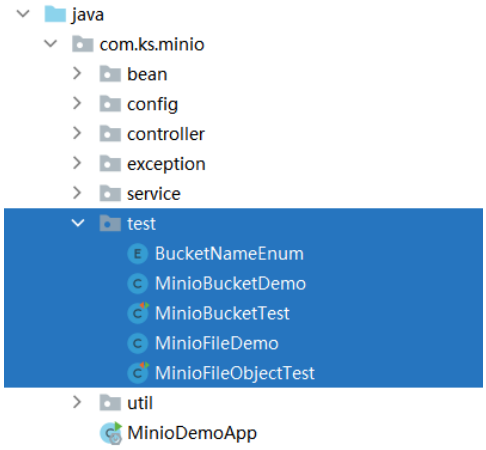
整合到SpringBoot的完整实例：
https://github.com/HackyleShawe/JavaDemos/tree/master/Examples/minio-demo

MinioClientUtils



TOP

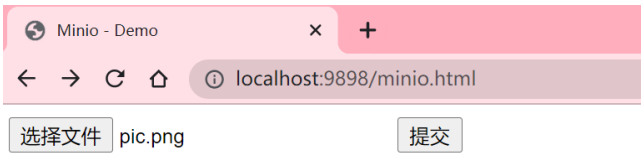
- 方法中带“bucket”关键词的，是操作桶的
- 方法中带“fileObject”关键字的，是操作（增删改查）文件对象的
- Demo项目的test包，是一系列针对桶、文件对象的测试类，可以从中窥见MinIO的一般用法



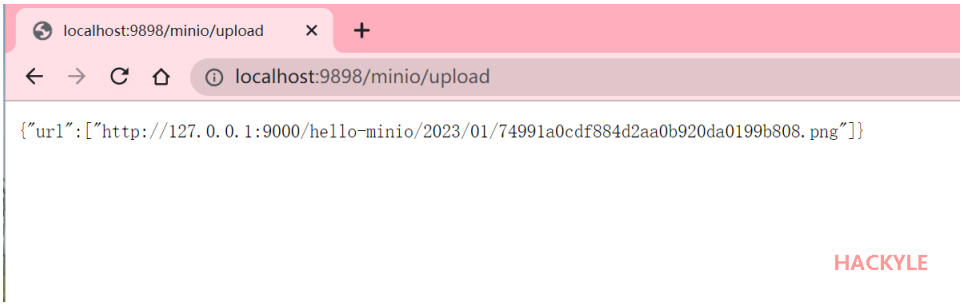
测试

启动MinIO服务

启动SpringBoot项目，访问http://localhost:9898/minio.html



上传文件

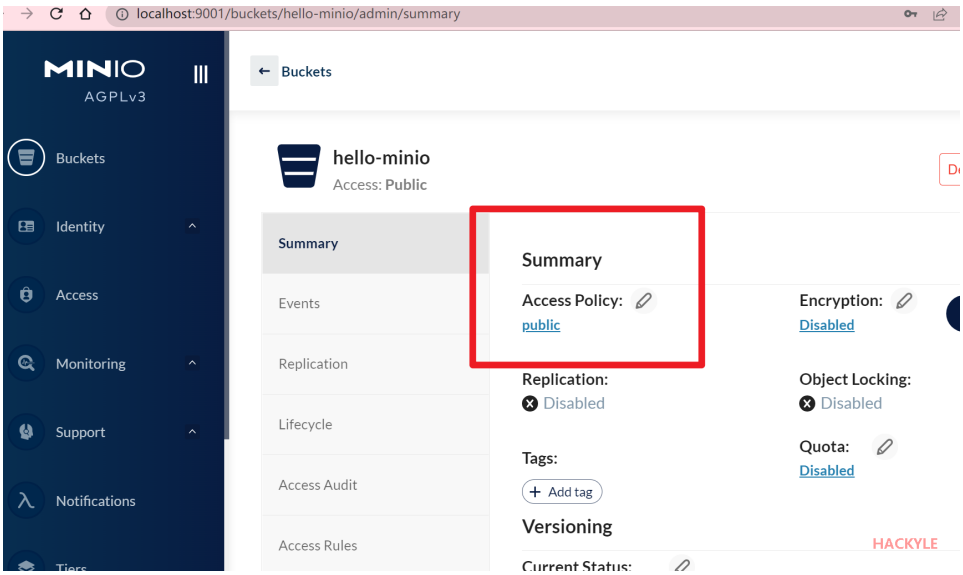


上传成功

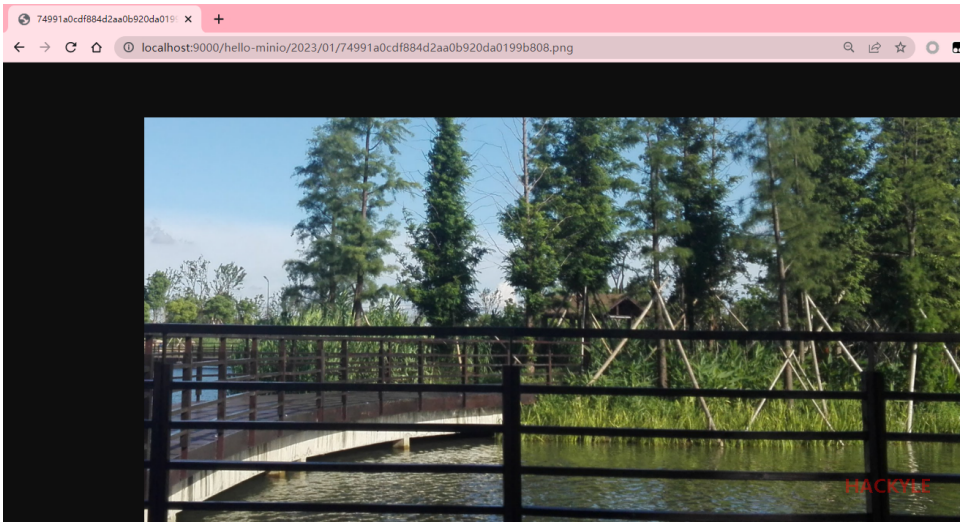
进入桶管理后台，查看刚刚上传的文件

TOP

通过MinIO访问刚刚上传的文件，发现拒绝访问



此时，需要进入MinIO的管理后台，修改桶的可见性为：public



成功访问

测试文件对象的删除

TOP

QA

报错：Unsupported OkHttp library found. Must use okhttp >= 4.8.1

```
D:\ProgramFilesKS\Java\JDK11\bin\java.exe ...
Exception in thread "main" java.lang.ExceptionInInitializerError Create breakpoint
    at io.minio.MinioClient$Builder.build(MinioClient.java:2735)
    at com.hackyle.boot.util.MinioClientUtils.<clinit>(MinioClientUtils.java:35)
Caused by: java.lang.RuntimeException Create breakpoint : Unsupported OkHttp library found. Must use okhttp >= 4.8.1
    at io.minio.S3Base.<clinit>(S3Base.java:100)
    ... 2 more
Caused by: java.lang.NoSuchMethodError Create breakpoint : okhttp3.RequestBody.create([BLokhttp3/MediaType;)Lokhttp3/RequestBody;
    at io.minio.S3Base.<clinit>(S3Base.java:98)
    ... 2 more
```

解决方案：引入okhttp

```xml
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>4.9.0</version>
</dependency>
```

报错：S3 API Request made to Console port. S3 Requests should be sent to API port.

```
io.minio.errors.InvalidResponseException: Non-XML response from server. Response code: 403, Content-Type: text/xml; charset=utf-8, body: <?xml version="1.0"
 encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>S3 API Request made to Console port. S3 Requests should be sent to API port.</Message>
  <RequestId>0</RequestId>
</Error>
```

原因：使用了连接Console的接口

解决方案：使用API的接口去访问，而不是Console

```
[root@HackyleCent-0 Downloads]# MINIO_ROOT_USER=kyle-minio MINIO_ROOT_PASSWORD=hackyle-minio minio
 server /var/minio-data --address ":9000" --console-address ":9001"
API: http://10.100.134.170:9000  http://192.168.122.1:9000  http://127.0.0.1:9000
RootUser: kyle-minio
RootPass: hackyle-minio

Console: http://10.100.134.170:9001 http://192.168.122.1:9001 http://127.0.0.1:9001
RootUser: kyle-minio
RootPass: hackyle-minio

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc alias set myminio http://10.100.134.170:9000 kyle-minio hackyle-minio

Documentation: https://docs.min.io
```

p 0 words tiny

SUBMIT RESET

TOP