

[🏠 / 开发指南 / 后端手册](#)[👤 芋道源码](#) [📅 2022-03-17](#)

📁 文件存储（上传下载）

项目支持将文件上传到三类存储器：

1. 兼容 S3 协议的对象存储：支持 MinIO、腾讯云 COS、七牛云 Kodo、华为云 OBS、亚马逊 S3 等等。
2. 磁盘存储：本地、FTP 服务器、SFTP 服务器。
3. 数据库存储：MySQL、Oracle、PostgreSQL、SQL Server 等等。

技术选型？

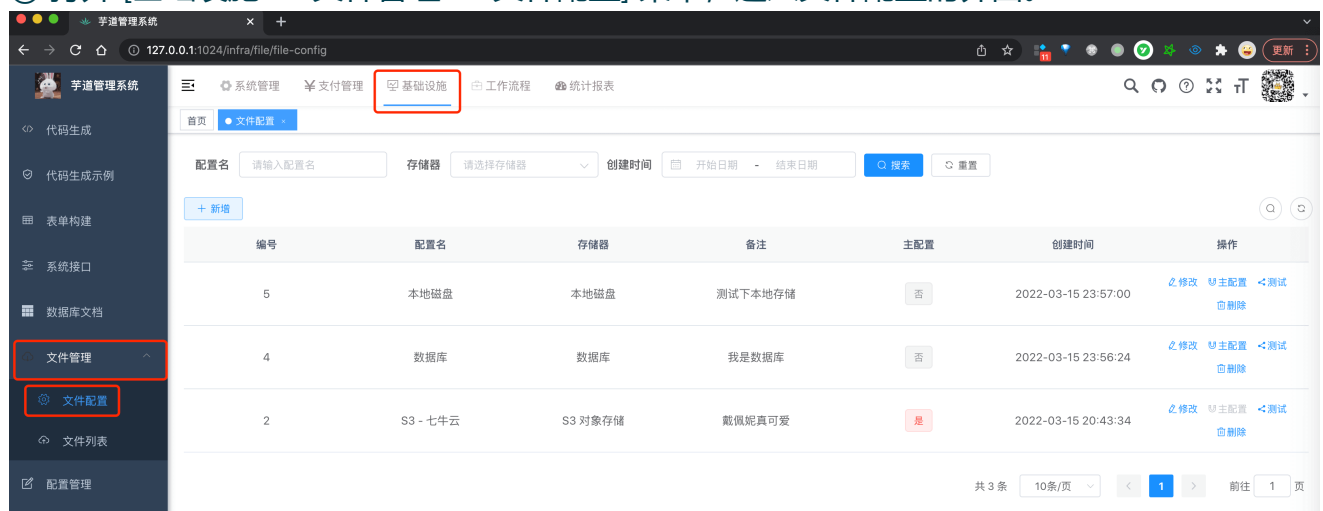
- 优先，✓ 推荐方案 1。如果无法使用云服务，可以自己搭建一个 MinIO 服务。参见 [《芋道 Spring Boot 对象存储 MinIO 入门》](#) 文章。
- 其次，推荐方案 3。数据库的主从机制可以实现高可用，备份也方便，少量小文件问题不大。
- 最后，× 不推荐方案 2。主要是实现高可用比较困难，无法实现故障转移。

1. 快速入门

本小节，我们来添加个文件配置，并使用它上传下载文件。

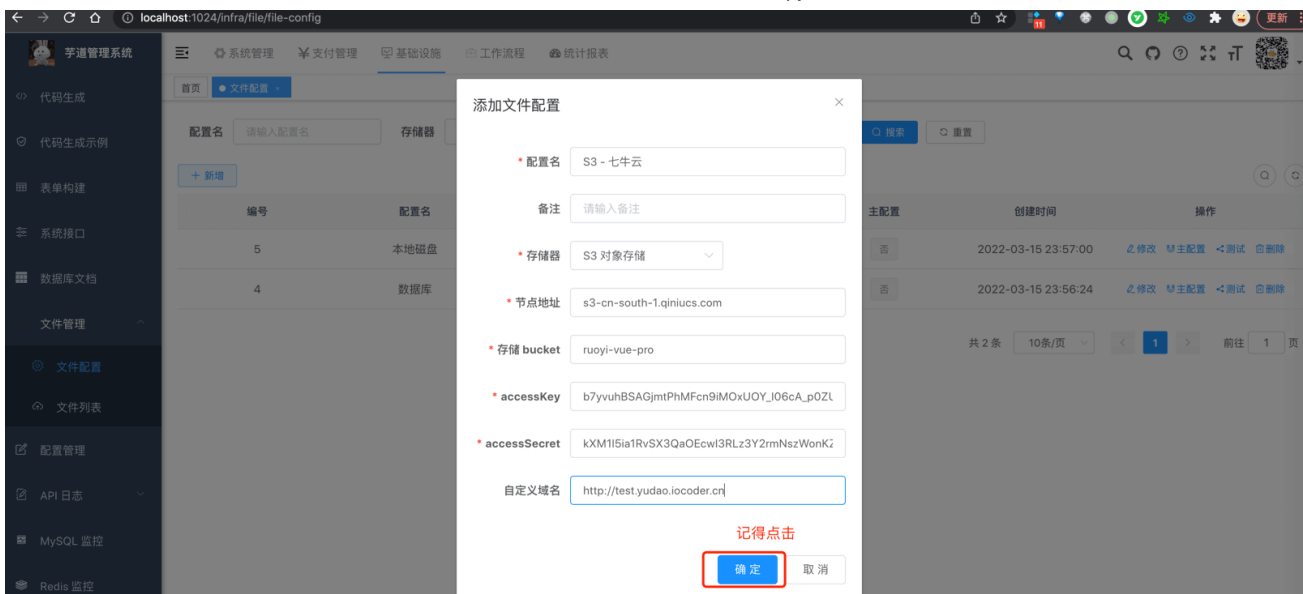
1.1 新增配置

① 打开 [基础设施 -> 文件管理 -> 文件配置] 菜单，进入文件配置的界面。



编号	配置名	存储器	备注	主配置	创建时间	操作
5	本地磁盘	本地磁盘	测试下本地存储	否	2022-03-15 23:57:00	修改 主配置 测试 删除
4	数据库	数据库	我是数据库	否	2022-03-15 23:56:24	修改 主配置 测试 删除
2	S3 - 七牛云	S3 对象存储	戴佩妮真可爱	是	2022-03-15 20:43:34	修改 主配置 测试 删除

② 点击 [新增] 按钮，选择存储器为【S3 对象存储器】，并填写七牛云的配置。如下图：

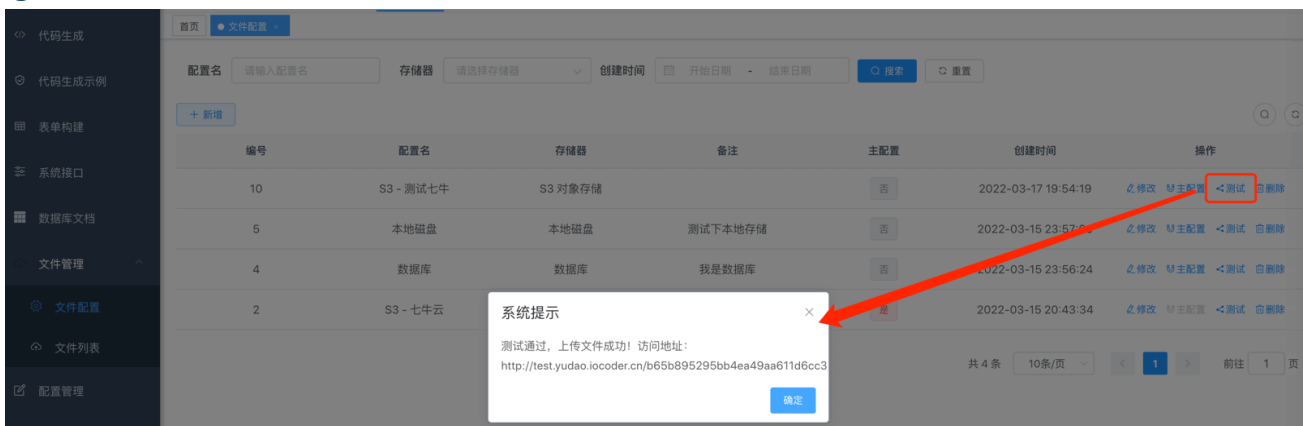


- 节点地址: s3-cn-south-1.qiniucs.com
- 存储 bucket: ruoyi-vue-pro
- accessKey: b7yvuhBSAGjmtPhMFcn9iMOxUOY_I06cA_p0ZUx8
- accessSecret: kXM1I5ia1RvSX3QaOEcwI3RLz3Y2rmNszWonKZtP
- 自定义域名: http://test.yudao.iocoder.cn

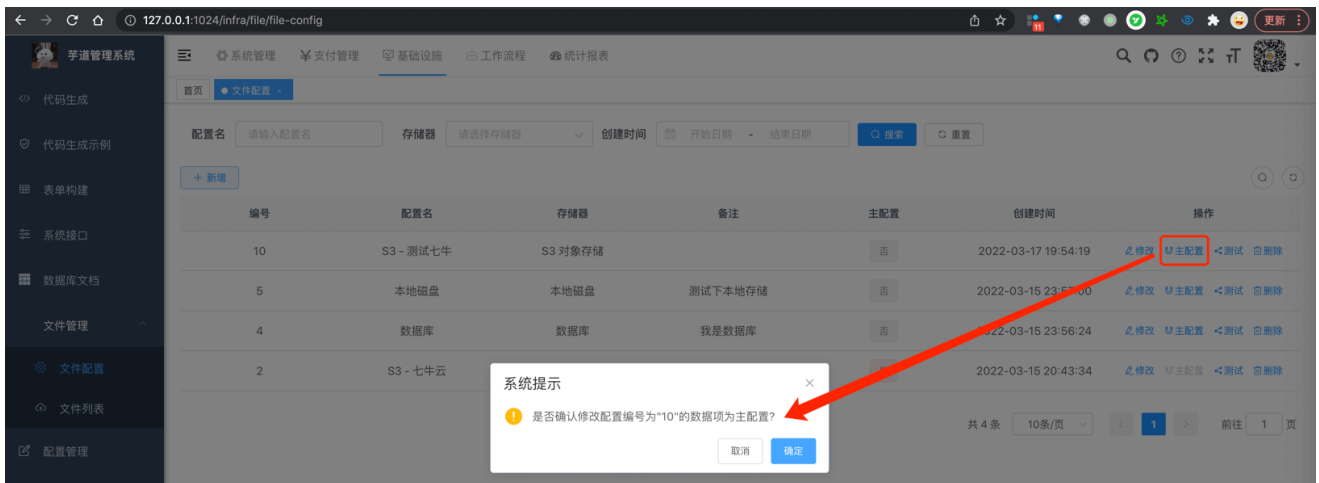
友善的眼神！

上述七牛云的配置，是芳芳为了大家方便体验，请勿在测试或生产环境体验。

③ 添加完后，点击该配置所在行的 [测试] 按钮，测试配置是否正确。

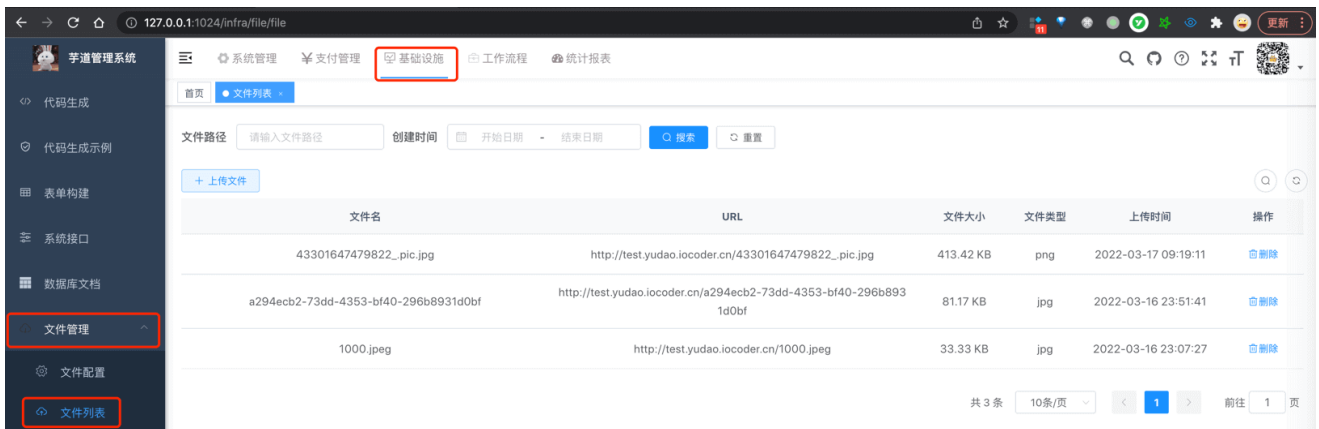


④ 测试通过后，点击该配置所在行的 [主配置] 按钮，设置它为默认的配置，后续使用它进行文件的上传。



1.2 上传文件

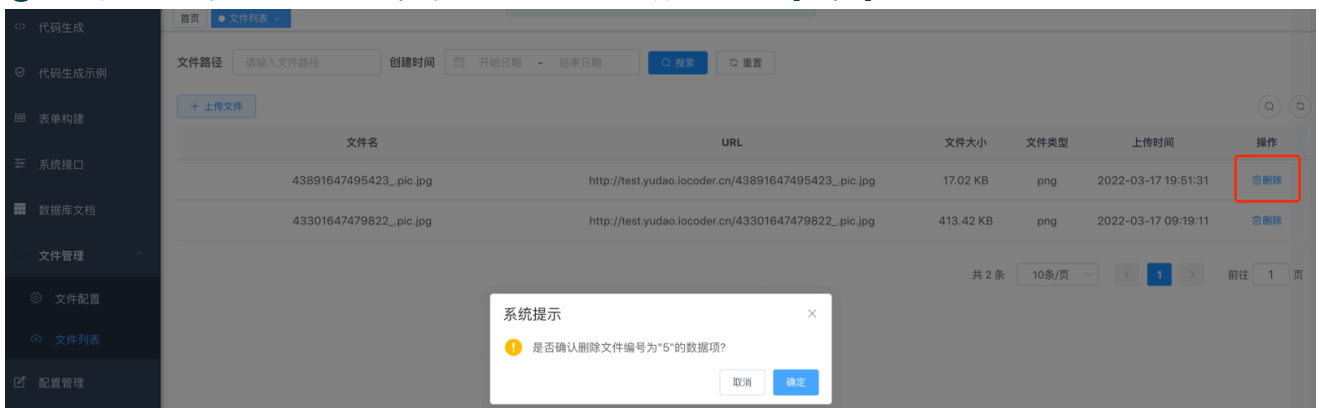
① 点击 [基础设施 -> 文件管理 -> 文件列表] 菜单，进入文件列表的界面。



② 点击 [上传文件] 按钮，选择要上传的文件。



③ 上传完成后，如果想要删除，可点击该文件所在行的 [删除] 按钮。



2. 文件上传

项目提供了 2 种文件上传的方式，分别适合前端、后端使用。

2.1 方式一：前端上传

`FileController` 提供了 `/admin-api/infra/file/upload` RESTful API，用于前端直接上传文件。

```
// FileController.java

@PostMapping("/upload")
@Operation(summary = "上传文件")
@OperateLog(logArgs = false) // 上传文件，没有记录操作日志的必要
public CommonResult<String> uploadFile(FileUploadReqVO uploadReqVO) throws Exception {
    MultipartFile file = uploadReqVO.getFile();
    String path = uploadReqVO.getPath();
    return success(fileService.createFile(file.getOriginalFilename(), path,
        IoUtil.readBytes(file.getInputStream())));
}
```

前端上传文件的代码如何实现，可见：

- 文件列表，文件上传 `index.vue`
- 个人中心，头像修改 `userAvatar.vue`

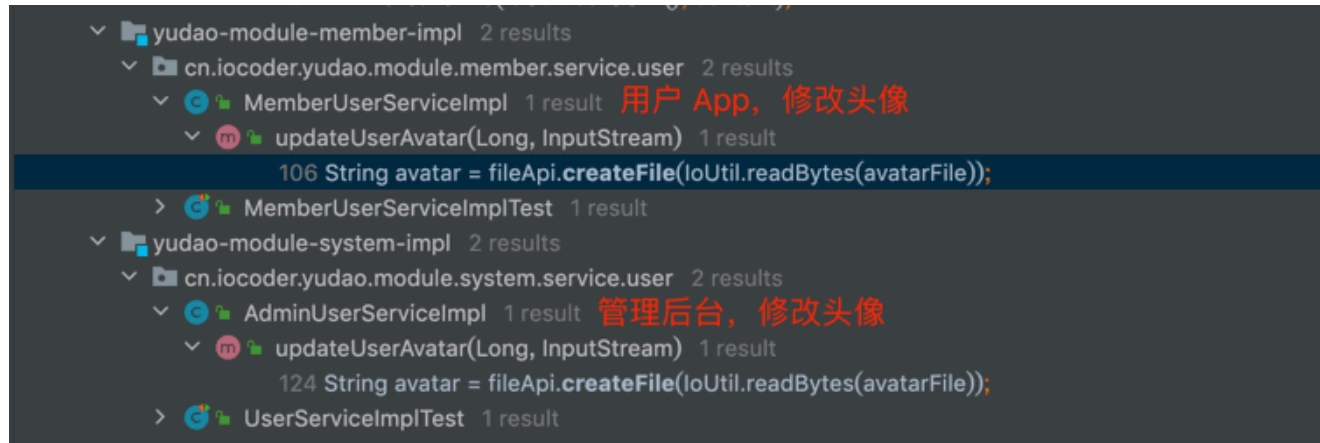
2.2 方式二：后端上传

`yudao-module-infra` 的 `FileApi` 提供了 `#createFile(...)` 方法，用于后端需要上传文件的逻辑。

```
// FileApi.java

/**
 * 保存文件，并返回文件的访问路径
 *
 * @param path 文件路径
 * @param content 文件内容
 * @return 文件路径
 */
String createFile(String path, byte[] content);
```

例如说，个人中心修改头像时，需要进行头像的上传。如下图所示：



注意，需要使用到后端上传的 Maven 模块，需要引入 `yudao-module-infra-api` 依赖。例如说 `yudao-module-system-biz` 模块的 `pom.xml` 文件，引用如下：

```
<dependency>
  <groupId>cn.iocoder.cloud</groupId>
  <artifactId>yudao-module-infra-api</artifactId>
  <version>${revision}</version>
</dependency>
```

3. 文件下载

文件上传成功后，返回的是**完整的 URL 访问路径**，例如说

<http://test.yudao.iocoder.cn/822aebded6e6414e912534c6091771a4.jpg>。

不同的文件存储器，返回的 URL 路径的规则是不同的：

- ① 当存储器是【S3 对象存储】时，支持 HTTP 访问，所以直接使用 S3 对象存储返回的 URL 路径即可。
- ② 当存储器是【数据库】【本地磁盘】等时，它们只支持存储，所以需要 `FileController` 提供的 `/admin-api/infra/file/{configId}/get/{path}` RESTful API，读取文件内容后返回。

```
// FileController.java

@GetMapping("/{configId}/get/**")
@PermitAll
@Operation(summary = "下载文件")
@Parameter(name = "configId", description = "配置编号", required = true)
public void getFileContent(HttpServletRequest request,
                          HttpServletResponse response,
                          @PathVariable("configId") Long configId) throws Exception {
    // 获取请求的路径
    String path = StrUtil.subAfter(request.getRequestURI(), "/get/", false);
    if (StrUtil.isEmpty(path)) {
```

```

        throw new IllegalArgumentException("结尾的 path 路径必须传递");
    }

    // 读取内容
    byte[] content = fileService.getFileContent(configId, path);
    if (content == null) {
        log.warn("[getFileContent][configId({}) path({}) 文件不存在]", configId,
            response.setStatus(HttpStatus.NOT_FOUND.value());
        return;
    }
    ServletUtils.writeAttachment(response, path, content);
}

```

4. 文件客户端

技术组件 `yudao-spring-boot-starter-file` [🔗](#)，定义了 `FileClient` [🔗](#) 接口，抽象了文件客户端的方法。

```

public interface FileClient {

    /**
     * 获得客户端编号
     *
     * @return 客户端编号
     */
    Long getId();

    /**
     * 上传文件
     *
     * @param content 文件流
     * @param path 相对路径
     * @return 完整路径，即 HTTP 访问地址
     */
    String upload(byte[] content, String path);

    /**
     * 删除文件
     *
     * @param path 相对路径
     */
    void delete(String path);

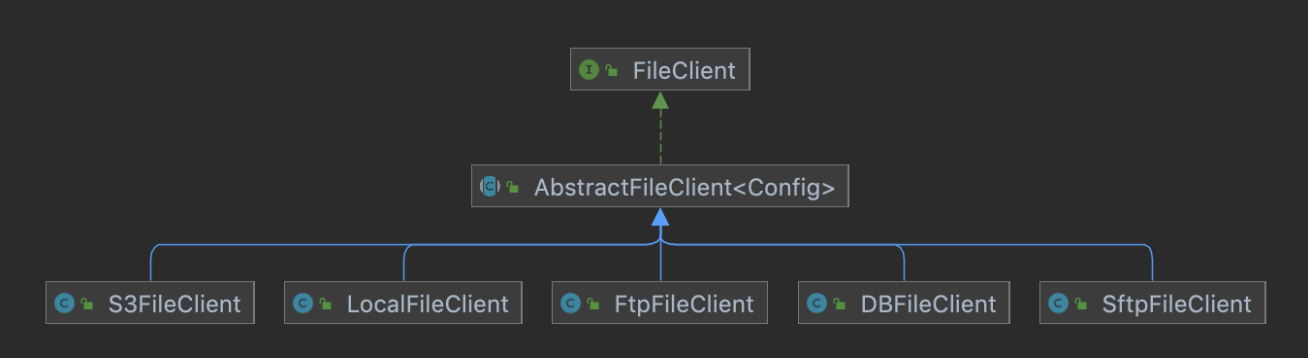
    /**
     * 获得文件的内容

```

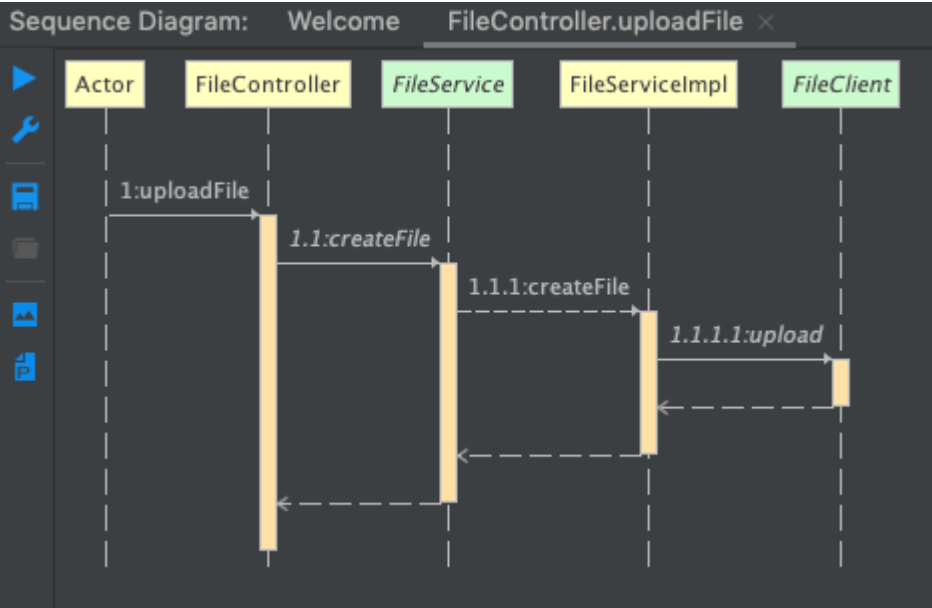
```
*
* @param path 相对路径
* @return 文件的内容
*/
byte[] getContent(String path);

}
```

FileClient 有 5 个实现类，使用不同存储器进行文件的上传与下载。UML 类图如所示：



文件上传的调用的 UML 时序图如下所示：



5. S3 对象存储的配置

做的不错的云存储服务，都是兼容 S3 协议的。如何获取对应的 S3 配置，[芳芳整理到了 S3FileClientConfig](#) 配置类。

有一点要注意，云存储服务的 Bucket 需要设置为**公共读**，不然 URL 无法访问到文件。并且，最好使用自定义域名，方便迁移到不同的云存储服务。



Theme by **Vdoing** | Copyright © 2019-2023 芋道源码 | MIT License