SpringBoot3 JDK21 Vue3开源后台RBAC管理系统 | 2024年好用的开源RBAC管理系统 | Sz-Admin发布部署

==========

序言

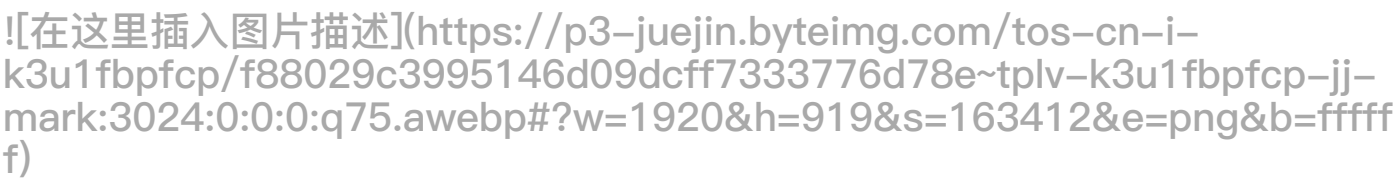
--

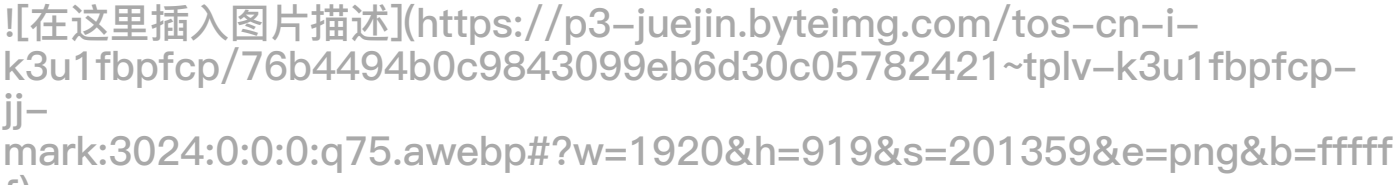
> 项目现已全面开源，商业用途完全免费！
>
>
> 当前版本：**v0.6.3**。
>
>
> 如果喜欢这个项目或支持作者，欢迎Star、Fork、Watch
> [一键三连](http://cxyroad.com/ "https://github.com/feiyuchuixue/sz-boot-parent")！！

在构建此代码框架的过程中，我已投入了大量精力，力求使其功能完善、结构清晰。然而，鉴于个人技术水平和经验的限制，框架中可能存在一些问题或可以改进的地方。对于任何可能的缺陷或不足，我在此表示诚挚的歉意，并恳请各位给予谅解。 我非常期待收到您的反馈和建议，您的每一条意见都是帮助我改进框架、提升技术水平的宝贵资源。让我们共同协作，不断优化和完善这个框架。 感谢您的使用与支持，期待与您的交流和合作！

简介

--





[Sz Admin](http://cxyroad.com/ "https://github.com/feiyuchuixue")

，一个基于 Spring Boot 3、JDK21、Vue 3 和 Element-Plus 的开源后台管理框架，致力于为您提供一个流畅、直观且功能强大的开发框架。它不仅融合了最新的技术趋势，而且通过精心设计，确保了系统的简洁性和高效，让使用者可以专注业务。

在线体验
----

* 官网地址：[szadmin.cn](http://cxyroad.com/ "https://szadmin.cn")
* 文档地址：[szadmin.cn/md/Help/doc...](http://cxyroad.com/ "https://szadmin.cn/md/Help/doc/start.html")
* 预览地址：[preview.szadmin.cn](http://cxyroad.com/ "https://preview.szadmin.cn")
* 代码仓库：
+ 前端：
- **Github**：**[sz-admin](http://cxyroad.com/ "https://github.com/feiyuchuixue/sz-admin.git")**
- **Gitee**：**[sz-admin](http://cxyroad.com/ "https://gitee.com/feiyuchuixue/sz-admin.git")**
+ 后端：
- **Github**：**[sz-boot-parent](http://cxyroad.com/ "https://github.com/feiyuchuixue/sz-boot-parent.git")**
- **Gitee**：**[sz-boot-parent](http://cxyroad.com/ "https://gitee.com/feiyuchuixue/sz-boot-parent.git")**
+ 部署：
- **Github**：**[sz-deploy](http://cxyroad.com/ "https://github.com/feiyuchuixue/sz-deploy.git")**
- **Gitee**：**[sz-deploy](http://cxyroad.com/ "https://gitee.com/feiyuchuixue/sz-deploy.git")**

部署
==

> 我们采用 ***Docker*** 方式进行部署，通过提供完整而充分的部署脚本支持，帮助您轻松部署项目并管理各种部署场景。无论您是初次接触 Docker 还是有丰富的部署经验，本文档都将为您提供清晰的步骤和详细的说明，以确保您能够顺利完成项目的部署工作。同时，我们也提供了针对多种情况的部署脚本，以满足不同环境和需求的部署需求。

仓库地址

Github：

```
git clone https://github.com/feiyuchuixue/sz-deploy.git
```

Gitee：

```
git clone https://gitee.com/feiyuchuixue/sz-deploy.git
```

目录结构

```
sz_deploy/
├── LICENSE
├── README.md
└── shell
    ├── deploy-minio.sh# minio部署脚本
    ├── build-sz-admin.sh# sz-admin前端Build脚本
    ├── common.env# 部署脚本环境变量
    ├── common.sh# 通用脚本（通用函数）
    ├── deploy-nginx.sh# Nginx部署脚本
    ├── deploy-sz-service-admin.sh# sz-service-admin服务部署脚本
    ├── deploy-sz-service-websocket.sh# sz-service-websocke服务部署脚本
    └── env-init.sh# 环境初始化脚本
```

部署手册

> [!TIP] 提示
>
>
> 系统配置要求：2C 4G 20G 以上。 内存至少为4G，否则有可能无法成功Build前端。
>
>
> `env-init.sh` 是一个初始脚本，主要为目标环境提供 ***Centos7*** 的配置。若您选择其他系统，请确保事先安装所需的依赖。
>
>
> 其他脚本不受系统影响。

在我们部署时，要遵循以下顺序进行：

File	描述	执行顺序
common.env	环境变量：根据自己情况配置	-
common.sh	通用函数	-
env-init.sh	环境初始化：安装Git、Docker、提供Docker代理配置可选项、Docker开机自启	***1***
deploy-sz-service-admin	Java部署：创建 docker network	***2***
deploy-sz-service-websocket.sh	Java部署：创建 docker network	**2**
build-sz-admin.sh	*前端编译，Nginx静态资源挂载，Nginx配置等*	***2***
deploy-minio.sh	Minlo部署，创建 docker network	***2***
deploy-nginx.sh	Nginx部署	***3***

1. 环境准备

在执行脚本之前，我们需要确保脚本具有执行权限。请按照以下步骤设置权限：

```
cd /sz_deploy/shell
```

```
chmod +x *.sh
mv * /home/deploy
cd /home/deploy
```

完成权限设置后，您可以运行 `env-init.sh` 脚本：

> [!IMPORTANT]注意
>
>
> 添加了Docker Proxy支持，以便支持复杂网络环境。
>
>
> **脚本核心配置摘要：**
>
>
>
> ```
> #!/bin/bash
>
> # 环境初始化
>
> USE_DOCKER_PROXY="false"# 标识是否需要启用docker代理 // [!code focus] // [!code warning]
> DOCKER_PROXY="http://192.168.124.7:7890/"  # docker代理地址 // [!code focus] // [!code warning]
> echo "Docker proxy is set to $DOCKER_PROXY"
>
> ...
>
> ```

```
bash env-init.sh
```

这样，环境就准备好了，您可以继续进行后续操作。

---

2. 修改环境变量

在运行其他脚本前，请编辑`common.env`文件并按需调整其中参数

```
# common.env
export ENV_ROOT_DIR="/home";
export ENV_APP_DIR="app";
export ENV_DOWNLOAD_DIR="download";
export ENV_CONF_DIR="conf";
export ENV_DOCKER_NETWORK_NAME="sz";

export ENV_GIT_USERNAME="账户";// [!code highlight]// [!code error] // [!code focus] # 修改为你的git账户
export ENV_GIT_PASSWORD="密码";// [!code highlight]// [!code error] // [!code focus]  # 修改为你的git密码
export ENV_GIT_HOST="服务器HOST地址";  // [!code highlight]  // [!code error]  // [!code focus] # 设置你的git密码服务器地址
export ENV_GIT_PROTOCOL="http"; // [!code highlight]  // [!code error] // [!code focus] # 设置你的git协议 ['http','https']
export ENV_GIT_HOST_SSH="github.com";    // [!code highlight] // [!code warning] // [!code focus] # 当然您也可以选用git ssh，这里我们演示的是github

export ENV_NGINX_ROOT_DIR="$ENV_ROOT_DIR/$ENV_APP_DIR/nginx"; # nginx 根路径
export ENV_NGINX_LOG_DIR="$ENV_NGINX_ROOT_DIR/logs";          # nginx 日志目录
export ENV_NGINX_STATIC_DIR="$ENV_NGINX_ROOT_DIR/static";     # nginx 静态资源目录
export ENV_NGINX_CNF_DIR="$ENV_NGINX_ROOT_DIR/conf.d";        # nginx conf 配置文件目录
export ENV_NGINX_CERT_DIR="$ENV_NGINX_ROOT_DIR/cert";         # nginx ssl证书目录

export ENV_PUBLIC_MAVEN_REPOSITORY="$ENV_ROOT_DIR/repository/mvn"; # maven 仓库路径(项目隔离)
export ENV_PUBLIC_NODE_MODULES_REPOSITORY="$ENV_ROOT_DIR/repository/node_modules";  # node_modules 模块仓库路径(项目隔离)
```

```

---

### 3. SpringBoot服务部署

> [!IMPORTANT]注意
>
>
> 脚本中需要注意的环境变量：
>
>
> * **USE\_GIT\_SSH**：该参数用于指示是否使用 Git SSH。如果设置为 `true`，将使用 Git SSH 进行克隆操作。若使用 GitHub SSH 克隆，则需要配置认证密钥，认证密钥应存放在 `~/.ssh/id_rsa` 目录下。
> * **USE\_ENV\_CONFIG**: ['true','false']。该参数用于指示是否使用环境变量配置文件。如果设置为 `true`，在部署时会优先使用位于 ***CONFIG\_DIR*** 路径下的`同名`配置文件进行`覆盖`。
>
>
> **脚本核心配置摘要：**
>
>
>
> ```
> #!/bin/bash
>
> # 引用公共函数和变量
> source ./common.sh
>
> set -e # 如果命令失败，则立即退出
>
> BRANCH_NAME="main" #分支名// [!code focus] // [!code warning]
> PROJECT_NAME="sz-service-admin"
> PORT=9991   # 服务端口
> PROFILE_ACTIVE="preview" # java -jar -Dspring.profiles.active=dev java部署环境// [!code focus] // [!code warning]
>
> APP_TMP_PATH="${ENV_ROOT_DIR}/${ENV_DOWNLOAD_DIR}/${PROJECT_NAME}" #项目下载路径
>
> DEPLOY_VERSION=$(date +%Y%m%d%H%M%S)
> APP_POM_PATH="${APP_TMP_PATH}/${DEPLOY_VERSION}"
```

> APP_TARGET_PATH="${APP_POM_PATH}/sz-service/sz-service-admin/target"
> APP_CONFIG_PATH="${APP_POM_PATH}/sz-service/sz-service-admin/src/main/resources/config" # 子级配置文件目录
>
DEPLOY_DIR="${ENV_ROOT_DIR}/${ENV_APP_DIR}/${PROJECT_NAME}/${DEPLOY_VERSION}" # 部署文件备份地址
>
LOG_DIR="${ENV_ROOT_DIR}/${ENV_APP_DIR}/${PROJECT_NAME}/logs"
>
> USE_GIT_SSH=true # 是否使用git（github） ssh // [!code focus] // [!code warning]
> GIT_PATH="dev/sz-boot-parent.git"
git项目路径
> GIT_PATH_SSH="feiyuchuixue/sz-boot-parent.git"
git项目路径
>
PROJECT_REPO="${ENV_GIT_PROTOCOL}://${ENV_GIT_USERNAME}:${ENV_GIT_PASSWORD}@${ENV_GIT_HOST}/${GIT_PATH}" # git clone 地址
> PROJECT_REPO_SSH="git@${ENV_GIT_HOST_SSH}:$GIT_PATH_SSH"
>
> USE_ENV_CONFIG=true # 是否使用环境变量配置文件 // [!code focus] // [!code warning]
>
CONFIG_DIR="${ENV_ROOT_DIR}/${ENV_CONF_DIR}/${PROJECT_NAME}" # 环境变量配置文件目录 // [!code focus] // [!code warning]
>
> ...
>
> ```

接下来，我们部署SpringBoot：

```
bash deploy-sz-service-admin.sh
```

执行后展示信息如下（部分截取）：

```
```

```
[INFO] sz-boot-parent ..... SUCCESS [  0.147 s]
[INFO] sz-common ..... SUCCESS [  0.004 s]
[INFO] sz-common-core ..... SUCCESS [  6.332 s]
[INFO] sz-common-db-mongodb ..... SUCCESS [  0.047 s]
[INFO] sz-common-db-redis ..... SUCCESS [  0.932 s]
[INFO] sz-common-security ..... SUCCESS [  0.946 s]
[INFO] sz-common-log ..... SUCCESS [  1.001 s]
[INFO] sz-common-db-mysql ..... SUCCESS [  0.439 s]
[INFO] sz-common-excel ..... SUCCESS [  1.121 s]
[INFO] sz-common-generator ..... SUCCESS [  2.084 s]
[INFO] sz-common-minio ..... SUCCESS [  0.734 s]
[INFO] sz-common-mq ..... SUCCESS [  0.037 s]
[INFO] sz-common-wechat ..... SUCCESS [  0.522 s]
[INFO] sz-service ..... SUCCESS [  0.364 s]
[INFO] sz-service-admin ..... SUCCESS [  5.590 s]
[INFO] sz-service-websocket ..... SUCCESS [  1.044 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  22.022 s
[INFO] Finished at: 2024-05-22T02:32:43Z
[INFO] ------------------------------------------------------------------------
========== 打包完成 ==========
==================== 删除旧镜像 ====================
Deleted Images:
deleted: sha256:e26f18c7abae3cdeab303f625e0ffcda924b751bf53b7c7b23c3ab78ee46399c

Total reclaimed space: 0B
ID                                RECLAIMABLE     SIZE            LAST ACCESSED
vr6dboqzx599rgusu8l4q2y68*        true            0B              4 days ago
vcpicgtc21ymi2zonnphi8xio*        true            107.6MB         4 days ago
r8l0qoabf6va3forkjfbni0r2         true            107.6MB         4 days ago
j9fi2l1be6hthif0nid7kpsu5*        true            296B            4 days ago
Total:  215.3MB
==================== 构建 Docker 镜像 ====================
[+] Building 5.0s (7/7) FINISHED
                                                  docker:default
```

```
 => [internal] load build definition from Dockerfile
                                                                       0.0s
 => => transferring dockerfile: 335B
                                                                       0.0s
 => [internal] load .dockerignore
                                                                       0.0s
 => => transferring context: 2B
                                                                       0.0s
 => [internal] load metadata for docker.io/azul/zulu-openjdk:21
                                                                       1.6s
 => [internal] load build context
                                                                       0.7s
 => => transferring context: 107.66MB
                                                                       0.7s
 => CACHED [1/2] FROM docker.io/azul/zulu-openjdk:21@sha256:cea84772f8715343b52e37eb3a9cde3aecdccb92e59eed6595fc5ca0ec552615    0.0s
 => [2/2] COPY *.jar app.jar
                                                                       2.1s
 => exporting to image
                                                                       0.5s
 => => exporting layers
                                                                       0.5s
 => => writing image sha256:3ae939077758875309852ac6e8078f5d46429d48ad4e0ceee15dce3bb0c78618
0.0s
 => => naming to docker.io/library/sz-boot-parent
                                                                       0.0s
==================== 停止旧应用容器 ====================
sz-service-admin
sz-service-admin
==================== 启动新应用容器 ====================
1e060ad9caf753562d755d5be06918b5fed0061dc8b8d06f4b5e55e69e9090ae
检查 Docker 网络 sz 已存在
==================== 整体执行耗时为: 40 秒====================
==================== 查看日志 ====================
...
====================== app is running finish ...
======================

                              __           _
                             | ]          (_)
 .--.  ____  ______  ,--.   .--.| |  _ .--..--.   __   _ .--.
( (`\] [_  ]|______|`'_\ : / /'`\' | [ `.-. .-. | [  | [ `.-. |
 `'.'.  .' /_       // | |,| \__/  | | | | | | | | |  | | | | |
[\__) )[_____]      \'-;__/ '.__.;__][___||__||__][___][___||__]
-----------------https://szadmin.cn  (v0.6.0 Beta)-------------
```

```

---

### 4. Nginx部署

> [!CAUTION]警告
>
>
> 在使用 Docker Network 的环境中，Nginx 的部署依赖于其他服务的启动顺序。为了确保 Nginx 能够正常工作，请先启动所有相关服务，然后再运行 Nginx。这样可以保证 Nginx 能够正确连接到其他服务。

> [!IMPORTANT]注意
>
>
> 脚本将自动检测 Nginx 服务状态和配置文件状态，并根据检测结果执行相应操作：
>
>
> 1. 容器不存在或服务未启动：
> * 脚本会执行初始化操作，重新生成容器，但不会影响现有配置。
> 2. 容器存在且服务正在运行：
> * 脚本将重载配置文件，而无需重启容器。
>
>
> 此外，脚本还包含一个手动确认步骤，以便用户决定是否需要重启容器。
>
>
> 与 deploy-sz-boot.sh 一样，脚本提供了敏感配置覆盖功能。此功能确保在部署过程中，敏感配置能够被安全地更新和覆盖。
>
>
> 通过`USE_ENV_CONFIG`和 `CONFIG_DIR`属性进行配置。
>
>
> 在脚本中，`PORT_MAPPINGS` 是一个关键配置项，它定义了Docker容器中Nginx服务的端口映射情况。通过调整此参数，您可以灵活配置需要对外暴露的
```

端口。
>
>
> **脚本核心配置摘要**：
>
>
>
> ```
> #!/bin/bash
> # 引用公共函数和变量
> source ./common.sh
> set -e # 如果命令失败，则立即退出
>
> # 映射端口变量
> PORT_MAPPINGS=(# --开放端口--// [!code focus] // [!code warning]
>   "9800:9800"  # 9800端口 // [!code focus] // [!code warning]
>   "80:80"  # 80端口 // [!code focus] // [!code warning]
>   "443:443"  # 443端口 // [!code focus] // [!code warning]
> )# --开放端口-- // [!code focus] // [!code warning]
>
> PROJECT_NAME="nginx"                    # 项目名称
> USE_ENV_CONFIG=true                                 # 是否使用环境变量配置文件// [!code focus] // [!code warning]
>
CONFIG_DIR="${ENV_ROOT_DIR}/${ENV_CONF_DIR}/${PROJECT_NAME}" # 环境变量配置文件目录 // [!code focus] // [!code warning]
>
> ...
>
> ```

运行如下命令：

```
bash deploy-nginx.sh
```

执行后展示信息如下：

```
NGINX容器存在且处于运行状态
NGINX平滑重载
```

========== 替换环境变量配置文件 ==========
==================== 平滑重载NGINX配置 ====================
是否需要重启容器？(y/n): y
重新启动容器
发现同名容器 nginx，删除中...
同名容器 nginx 已删除
==================== 启动新应用容器 ====================
711847945c7944eba1e8a4441e0ba1e022353139897fede61882ebd5a6e2a329
==================== 加入网络 ====================
部署完成

```

---

### 5. 前端部署

> [!CAUTION]警告
>
>
> 使用 Vite 编译 Vue3 时，内存占用较高：至少需要保证有 2.5GB 的空闲内存！！！否则可能导致编译失败。编译完成后内存会释放。

> [!IMPORTANT]注意
>
>
> 若要开启**NPM镜像加速**，请修改`USE_NPM_PROXY`、`NPM_CONFIG_REGISTRY`配置
>
>
> **脚本核心配置摘要：**
>
>
>
> ```
> #!/bin/bash
> # 引用公共函数和变量
> source ./common.sh
> set -e # 如果命令失败，则立即退出
>
```

> # 定义变量
> BRANCH_NAME="main"  # 分支名
> PROJECT_NAME="sz-admin" # 项目名称
>
> USE_GIT_SSH=true  # 是否使用Git SSH  // [!code focus] // [!code warning]
> GIT_PATH="feiyuchuixue/sz-admin.git"  # git项目路径
> GIT_PATH_SSH="feiyuchuixue/sz-admin.git"  # git项目路径

>
PROJECT_REPO="${ENV_GIT_PROTOCOL}://${ENV_GIT_USERNAME}:${ENV_GIT_PASSWORD}@${ENV_GIT_HOST}/${GIT_PATH}" # git clone 地址
> PROJECT_REPO_SSH="git@${ENV_GIT_HOST_SSH}:$GIT_PATH_SSH"
>
> USE_NPM_PROXY=true  # 是否使用npm镜像代理 // [!code focus] // [!code warning]
> NPM_CONFIG_REGISTRY="https://mirrors.cloud.tencent.com/npm/" # npm镜像加速地址（默认腾讯云）  // [!code focus] // [!code warning]
>
>
APP_TMP_PATH="${ENV_ROOT_DIR}/${ENV_DOWNLOAD_DIR}/${PROJECT_NAME}"  # 项目下载路径
> DEPLOY_VERSION=$(date +%Y%m%d%H%M%S)  # 部署版本号
>
APP_PACKAGE_JSON_PATH="${APP_TMP_PATH}/${DEPLOY_VERSION}"  # 项目package.json路径
>
PUBLIC_NODE_MODULES_REPOSITORY="${ENV_PUBLIC_NODE_MODULES_REPOSITORY}/${PROJECT_NAME}" # 挂载node_modules地址
>
DEPLOY_DIR="${ENV_ROOT_DIR}/${ENV_APP_DIR}/${PROJECT_NAME}/${DEPLOY_VERSION}"  # 部署文件备份地址
>
> ...
>
> ```

运行如下命令：

```
bash build-sz-admin.sh
```

执行后展示信息如下（部分截取）：

```
 built in 33.42s
========== 打包完成 ==========
total 196
drwxr-xr-x  3 root root   4096 May 22 10:45 dist
-rw-r--r--  1 root root     38 May 22 10:45 env.d.ts
-rw-r--r--  1 root root    331 May 22 10:45 index.html
-rw-r--r--  1 root root  11336 May 22 10:45 LICENSE
drwxr-xr-x  2 root root   4096 May 22 10:45 node_modules
-rw-r--r--  1 root root   1577 May 22 10:45 package.json
-rw-r--r--  1 root root 137261 May 22 10:45 pnpm-lock.yaml
drwxr-xr-x  2 root root   4096 May 22 10:45 public
-rw-r--r--  1 root root   1675 May 22 10:45 README.md
drwxr-xr-x 16 root root   4096 May 22 10:45 src
-rw-r--r--  1 root root    260 May 22 10:45 tsconfig.app.json
-rw-r--r--  1 root root    139 May 22 10:45 tsconfig.json
-rw-r--r--  1 root root    321 May 22 10:45 tsconfig.node.json
-rw-r--r--  1 root root   1452 May 22 10:45 vite.config.ts
/home/download/sz-admin/20240522104507
DEPLOY_DIR == /home/app/sz-admin/20240522104507
total 8
drwxr-xr-x 3 root root 4096 May 22 10:45 dist
drwxr-xr-x 2 root root 4096 May 22 10:45 nginx
========== 清理 ==========
==================== 整体执行耗时为: 53 秒====================
==================== 查看日志 ====================
```

---

6. MinIO部署

> [!IMPORTANT]注意
>
>
> MinIO 对系统时间有一定的要求，因此在部署过程中会先进行 NTP 时间服务器同步。

>
>
> **脚本核心配置摘要：**
>
>
>
> ```
> #!/bin/bash
>
> # 引用公共函数和变量
> source ./common.sh
> set -e # 如果命令失败，则立即退出
>
> TZ="Asia/Shanghai" # 时区
>
> PROJECT_NAME="minio"            # 项目名称
> ROOT_USER="用户名"               # minio 用户名// [!code focus] // [!code warning]
> ROOT_PWD="密码"                  # minio 密码 // [!code focus] // [!code warning]
> DATA_DIR="/mnt/minio/data"      # minio文件存储磁盘
> CONFIG_DIR="/mnt/minio/config" # minio配置路径
>
> WEB_PORT="9000"                                  # minio控制台
> SERVER_PORT="9001"                                 # minio API
> MINIO_DOMAIN="https://your.domain.com"          # minio API域名// [!code focus] // [!code warning]
> MINIO_DOMAIN_BROWSER="https://your.domain.com" # minio 域名// [!code focus] // [!code warning]
>
> ...
>
> ```

运行如下命令：

```
bash base-minio.sh
```

部署结构概览

> 这里我们以**common.env** `ENV_ROOT_DIR`="/home"为例。

部署完成后，脚本将在指定的目录下创建以下结构：

根目录结构：

```
home/
├── app# app 应用
├── conf# 环境（敏感）变量配置目录
├── deploy# 部署脚本
├── download# 下载源码目录，脚本执行后会清理。
└── repository# 依赖仓库。项目级隔离
```

完整结构：

```
home/
├── app# // [!code warning]
│   ├── nginx# nginx 目录 // [!code highlight]
│   │   ├── cert# ssl证书
│   │   │   ├── fullchain.cer
│   │   │   ├── szadmin.cn.cert
│   │   │   └── szadmin.cn.key
│   │   ├── conf.d# conf.d 配置
│   │   │   ├── default.conf
│   │   │   ├── szadmin.cn.conf
│   │   │   └── szadmin.cn.conf
│   │   ├── logs# log日志
│   │   │   ├── access.log
│   │   │   ├── default.access.log
│   │   │   ├── default.error.log
│   │   │   ├── error.log
│   │   │   ├── szadmin.cn.access.log
```

```
│   │   │   └── szadmin.cn.error.log
│   │   └── static# 静态资源，站点
│   │       ├── vitepress
│   │       └── sz-admin
│   ├── sz-admin# sz-admin 目录，编译后的版本备份 // [!code highlight]
│   │   ├── 20240424214153# 版本号
│   │   │   ├── dist
│   │   │   └── nginx
│   │   ├── 20240424214541
│   │   │   ├── dist
│   │   │   └── nginx
│   │   └── ...
│   ├── sz-service-admin# sz-service-admin 目录，编译后的版本备份 // [!code highlight]
│   │   ├── 20240424214736# 版本号
│   │   │   ├── Dockerfile
│   │   │   └── sz-service-admin-1.0-SNAPSHOT.jar
│   │   ├── 20240426151321
│   │   │   ├── Dockerfile
│   │   │   └── sz-service-admin-1.0-SNAPSHOT.jar
│   │   ├── ...
│   │   └── logs# log日志
│   │       └── admin-service
│   └── vitepress# vitepress 目录，编译后的版本备份 // [!code highlight]
│       └── 20240522095819
│           └── dist
├── conf# // [!code warning]
│   ├── nginx
│   │   ├── default.conf
│   │   ├── preview.szadmin.cn.conf
│   │   └── szadmin.cn.conf
│   └── sz-service-admin
│       └── preview
│           ├── flyway.yml
│           ├── knife4j.yml
│           ├── minio.yml
│           ├── mybatis-flex.yml
│           ├── mysql.yml
│           ├── page-helper.yml
│           ├── redis.yml
│           └── sa-token.yml
├── deploy#  // [!code warning]
│   ├── build-sz-admin.sh
│   ├── build-vitepress.sh
│   ├── common.env
│   ├── common.sh
```

```
│       ├──── deploy-minio.sh
│       ├──── deploy-nginx.sh
│       ├──── deploy-sz-boot.sh
│       └──── env-init.sh
├──── download#  // [!code warning]
│       ├──── sz-admin
│       ├──── sz-boot-parent
│       └──── vitepress
├──── repository#  // [!code warning]
│       ├──── mvn# maven仓库（项目级隔离）
│       │       └──── sz-boot-parent
│       │           ├──── ...
│       └──── node_modules# node_modules仓库（项目级隔离）
│           ├──── sz-admin
│           │   ├──── ...
│           └──── vitepress
│               ├──── ...
```

原文链接: https://juejin.cn/post/7378459137418526720