****

**目录**

**智慧园区运营管理平台部署手册**

[1. 介绍 4](#_Toc128121128)

[1.1. 产品特点 4](#_Toc128121129)

[1.2. 部署架构 5](#_Toc128121130)

[2. 准备工作 5](#_Toc128121131)

[3. 部署步骤 5](#_Toc128121132)

[3.1. ECS服务器安装Docker镜像 7](#_Toc128121133)

[3.2. ECS服务器安装Redis、Nginx、Nacos、MongoDB 9](#_Toc128121134)

[3.2.1. 在ECS服务器上通过Docker安装Redis 9](#_Toc128121135)

[3.2.2. 在ECS服务器上通过Docker安装Nginx 12](#_Toc128121136)

[3.2.3. ECS服务器上通过Docker安装Nacos 15](#_Toc128121137)

[3.2.4. 在ECS服务器上通过Docker安装MongoDB 28](#_Toc128121138)

[3.3. Jenkins自动化部署应用 43](#_Toc128121139)

[3.3.1. 后端应用配置 43](#_Toc128121140)

[3.3.2. 前端应用配置 45](#_Toc128121141)

[3.3.3. 安装jenkins 47](#_Toc128121142)

[3.3.4. 安装配置nodejs 48](#_Toc128121143)

[3.3.5. 配置jdk 50](#_Toc128121144)

[3.3.6. 配置maven 50](#_Toc128121145)

[3.3.7. 配置git 50](#_Toc128121146)

[3.3.8. 设置时区 50](#_Toc128121147)

[3.3.9. 后端新建item 52](#_Toc128121148)

[3.3.10. 前端新建item 62](#_Toc128121149)

[3.4. Prometheus应用监控系统部署 73](#_Toc128121150)

[3.4.1. 安装node-exporter 73](#_Toc128121151)

[3.4.2. 安装prometheus 74](#_Toc128121152)

[3.4.3. 安装grafana 76](#_Toc128121153)

[3.4.4. 监控主机资源情况 78](#_Toc128121154)

[3.4.5. 监控docker级别资源 81](#_Toc128121155)

[3.5. 步骤五：ELK应用日志系统部署 84](#_Toc128121156)

[3.5.1. ElasticSearch 84](#_Toc128121157)

[3.5.2. ElasticSearch-Head（ES数据读取插件） 85](#_Toc128121158)

[3.5.3. logstash 86](#_Toc128121159)

[3.5.4. kibana 88](#_Toc128121160)

[3.6. 步骤六：视频转码服务部署 91](#_Toc128121161)

[3.6.1. 转码所需工具 91](#_Toc128121162)

[3.6.2. 安装ffmpeg 91](#_Toc128121163)

[3.6.3. ZLM（centos7） 92](#_Toc128121164)

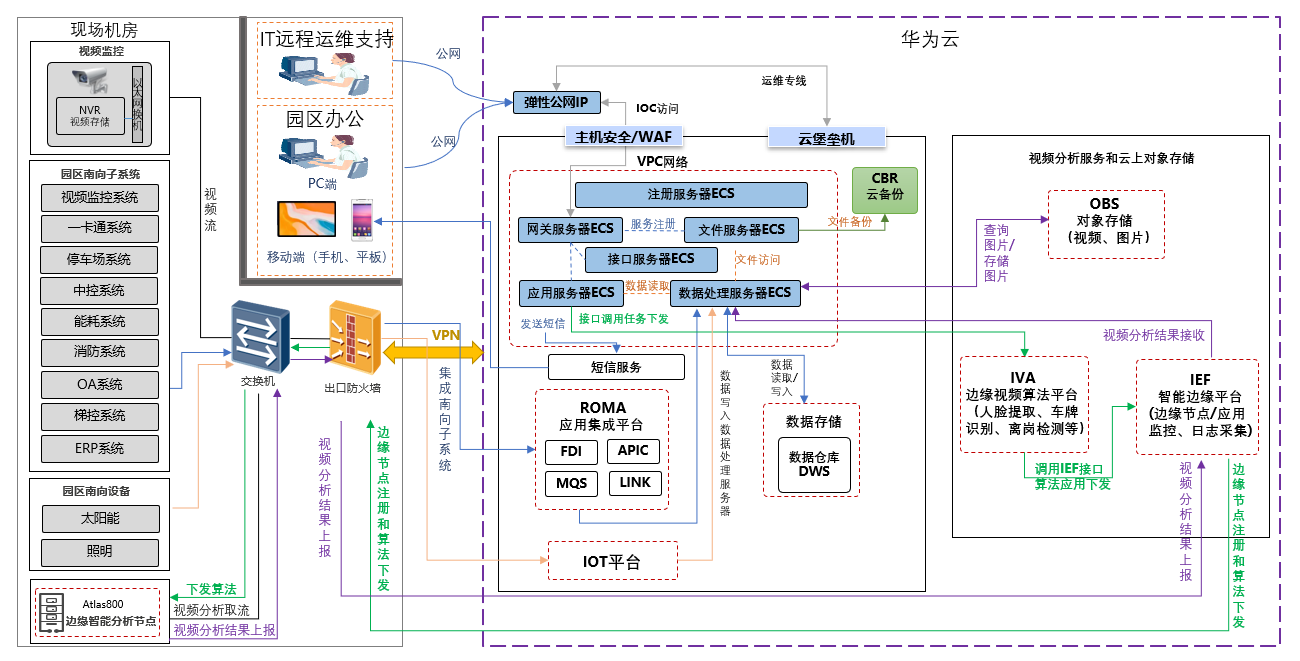
# 介绍

通过5G、物联网、大数据、人工智能等ICT技术，以科技赋能园区，打造“安全、舒适、高效 、绿色”的园区，促进园区在横向和纵向上各业务的集中管理和协同管理，提升园区社会和经济价值。

## 产品特点

* 园区数据全集成、设备全集成、系统全集成，实现园区数据可视、作业可管、运营可控。
* 通过智能视频分析助理园区安防升级，实现园区从“人防”到“技防”的转变。
* 实现园区“人、车、事、物”四大要素的集中管控，实现园区分钟级决策分析能力。
* 建设园区智慧“大脑”，实现园区可视化、精细化、流程化管控。
* 通过5G、物联网、人工智能、大数据等新ICT技术赋能园区，实现园区从单点智能到全部智慧。

## 部署架构



# 准备工作

* 创建华为云账号，开通华为云ECS服务器，开通华为云弹性公网IP。
* 在华为云平台上创建一个安全组，在其中添加一条远端任何IP地址、任何端口、任何协议、IPv4 版本、出方向的安全组规则；出方向的安全组规则，添加协议端口：80、443、22等端口。
* 华为云ECS服务器绑定安全组、绑定弹性公网IP。
* 华为云开通云数据库RDS for MySQL，创建云数据库连接账号。

# 部署步骤

* 步骤一：ECS服务器安装Docker镜像
* 步骤二：ECS服务器安装Redis、Nginx、Nacos、MongoDB
* 步骤三：Jenkins自动化部署应用
* 步骤四：Prometheus应用监控系统部署
* 步骤五：ELK应用日志系统部署
* 步骤六：视频转码服务部署

## ECS服务器安装Docker镜像

**（1）docker自动安装：**

使用一键安装脚本安装，缺点是不能自由选择版本，下载的是最新版本

curl -sSL https://get.daocloud.io/docker | sh

**（2）docker手动安装：**

**centos7：**

更新yum

yum update

卸载旧版本docker

yum remove docker docker-common docker-selinux docker-engine

yum安装依赖

yum install -y yum-utils device-mapper-persistent-data lvm2

写入软件源信息

yum-config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>

查看docker版本列表

yum list docker-ce --showduplicates | sort -r

yum安装docker

yum install docker-ce

**ubuntu18.04-20.04：**

安装系统工具：

apt-get update

apt-get -y install apt-transport-https ca-certificates curl software-properties-common

安装GPG证书

curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key add -

写入软件源信息

add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"

更新并安装Docker-CE

apt update

apt-get -y install docker-ce=5:18.09.9~3-0~ubuntu-bionic docker-ce-cli=5:18.09.9~3-0~ubuntu-bionic

启动命令：systemctl start docker

## ECS服务器安装Redis、Nginx、Nacos、MongoDB

### 在ECS服务器上通过Docker安装Redis

（1）下载Redis镜像：输入以下命令下载Redis镜像：

docker pull redis

（2）创建Redis容器：输入以下命令创建Redis容器：

docker run -d --name my-redis -p 6379:6379 redis:latest redis-server --requirepass 123456

（3）进入Redis容器：输入以下命令进入my-redis容器：

docker exec -it my-redis bash

（4）在my-redis容器中使用Redis命令：

* 输入以下命令启动Redis CLI：

redis-cli

* 或者可以直接通过交互模式(-it)，执行my-redis容器中的命令redis-cli

docker exec -it my-redis redis-cli

* 或者在宿主机内直接访问redis服务

redis-cli -h <docker-host-ip> -p 6379 -a 123456

* 也可以

redis-cli -h <docker-host-ip> -p 6379

（5）进入之后，再输入auth 123456。通常<docker-host-ip>写为localhost即可

redis-cli -h localhost -p 6379 -a 123456

（6）持久化

* 创建一个目录，用于存储 Redis 的数据：

mkdir ~/redis-data

* 创建容器。因为 “my-redis” 容器已经存在，要修改容器需要先停止并删除它，然后再使用新的命令创建容器。具体操作如下：

停止容器：docker stop my-redis

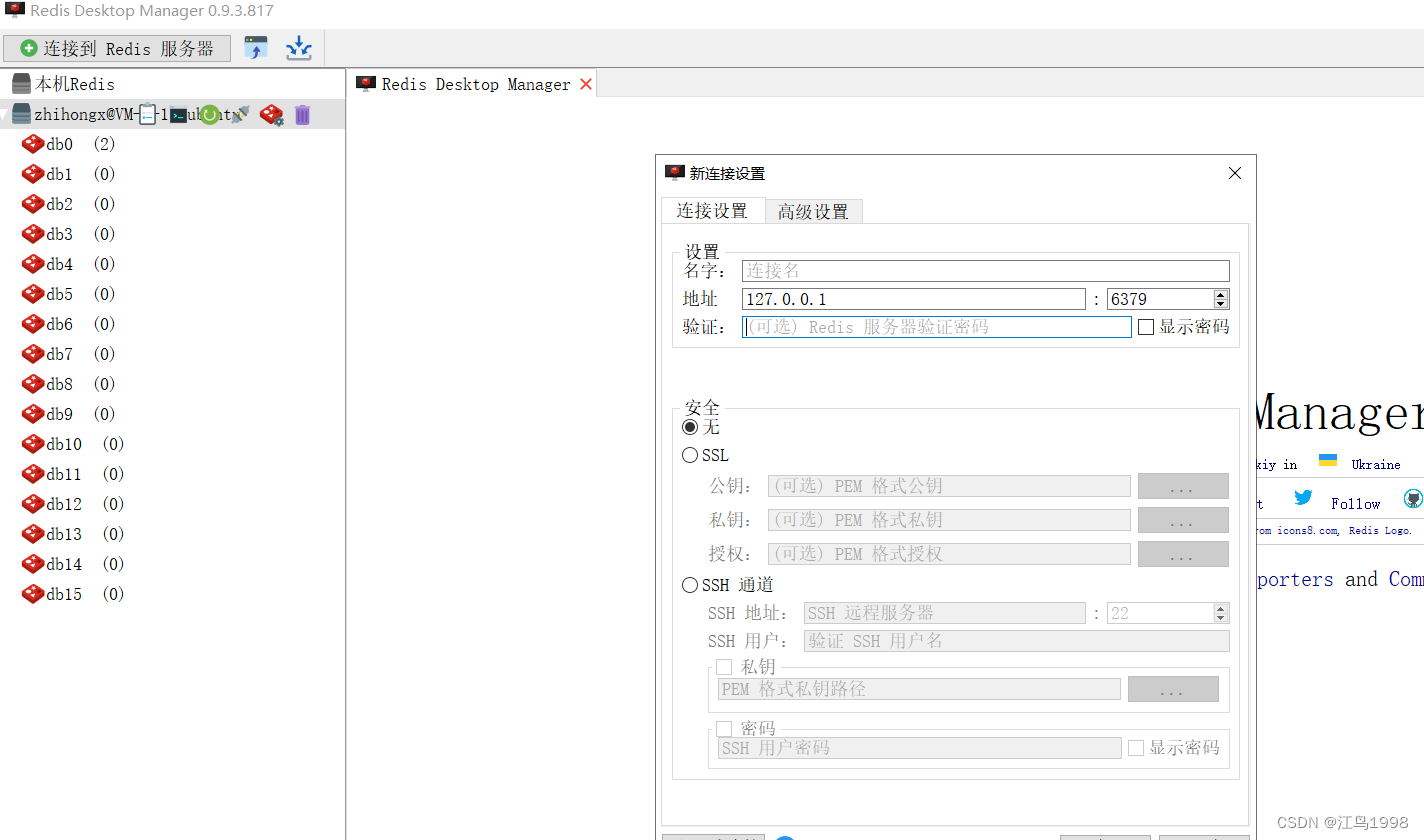
删除容器：docker rm my-redis

使用新命令创建容器：

docker run --name my-redis -p 6379:6379 -v ~/redis-data:/data -d redis:latest redis-server --appendonly yes --requirepass 123456

（7）测试

借助Redis Desktop Manager工具访问测试，登录成功代表安装成功。

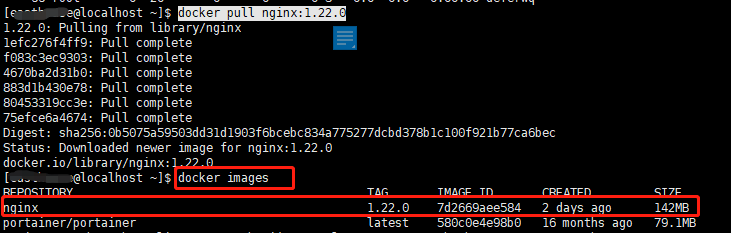


### 在ECS服务器上通过Docker安装Nginx

（1）下载镜像

docker pull nginx:1.22.0

（2）下载完成后检查镜像



（3）创建挂载目录

用于存放默认页面、日志、配置文件，把这些文件挂载到宿主机上好处是：在需要频繁修改Nginx配置的场景中，不需要关注容器中的数据，直接删除重建容器就好。

如果已经提前把对应nginx版本需要挂载的文件打包好了，这里直接解压放在对应的路径即可，不需要执行下面从容器复制文件的步骤。

* 解压目录：

cd ~

tar -zxvf docker-nginx.tar.gz

没有这些文件的话，就创建目录，从容器中复制过来。

* 创建目录

mkdir /home/用户/docker-nginx

（4）从容器中复制要挂载的文件

先简单创建容器运行，从容器里面复制要挂载的文件到宿主机上。

# 创建并运行容器，容器命名为nginx

docker run --name nginx -p 80:80 -d nginx:1.22.0

# 从容器nginx中复制nginx.conf文件到宿主机

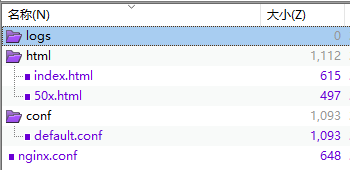
docker cp nginx:/etc/nginx/nginx.conf /home/用户/docker-nginx/

docker cp nginx:/etc/nginx/conf.d/ /home/用户/docker-nginx/conf/

docker cp nginx:/usr/share/nginx/html/ /home/用户/docker-nginx/html/

docker cp nginx:/var/log/nginx/ /home/用户/docker-nginx/logs/

复制出的docker-nginx目录结构如下图：



# 打包目录

tar -czvf docker-nginx.tar.gz docker-nginx/\*

# 停止、删除容器

docker stop nginx

docker rm nginx

（5）创建并运行容器

docker run -p 80:80 \

-v /home/用户/docker-nginx/nginx.conf:/etc/nginx/nginx.conf \

-v /home/用户/docker-nginx/logs:/var/log/nginx \

-v /home/用户/docker-nginx/html:/usr/share/nginx/html \

-v /home/用户/docker-nginx/conf:/etc/nginx/conf.d \

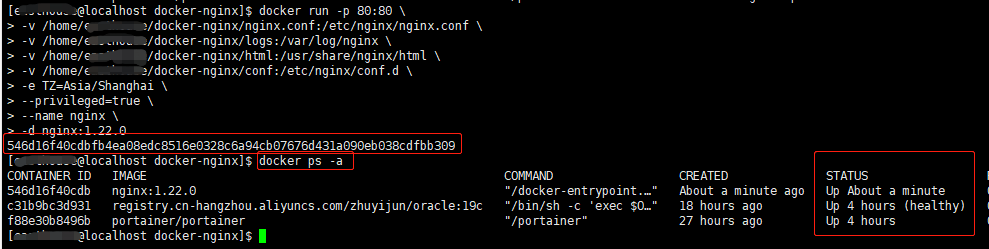
-v /etc/localtime:/etc/localtime \

--name nginx \

--restart=always \

-d nginx:1.22.0

如下图，执行后返回容器id没有报Error，查看容器列表，nginx容器状态正常，则安装成功。



浏览器访问http://ip:80，可以看到nginx页面



如果安装失败，查看容器列表是否已有nginx容器，有的话需要删除容器后再重新创建。

（6）修改Nginx配置

在宿主机上修改nginx.conf和conf目录下的配置文件后，需要重启容器重新加载配置。

### ECS服务器上通过Docker安装Nacos

（1）拉取镜像

docker pull nacos/nacos-server

（2）创建本地映射文件，custom.properties

mkdir -p /root/nacos/init.d /root/nacos/logs

touch /root/nacos/init.d/custom.properties

在文件中写入以下配置

management.endpoints.web.exposure.include=\*

（3）创建数据库 nacos\_config

（4）nacos初始化sql,需要先创建nacos数据库后，然后执行下面的Sql

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = config\_info \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `config\_info` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'id',

`data\_id` varchar(255) NOT NULL COMMENT 'data\_id',

`group\_id` varchar(255) DEFAULT NULL,

`content` longtext NOT NULL COMMENT 'content',

`md5` varchar(32) DEFAULT NULL COMMENT 'md5',

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '创建时间',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '修改时间',

`src\_user` text COMMENT 'source user',

`src\_ip` varchar(20) DEFAULT NULL COMMENT 'source ip',

`app\_name` varchar(128) DEFAULT NULL,

`tenant\_id` varchar(128) DEFAULT '' COMMENT '租户字段',

`c\_desc` varchar(256) DEFAULT NULL,

`c\_use` varchar(64) DEFAULT NULL,

`effect` varchar(64) DEFAULT NULL,

`type` varchar(64) DEFAULT NULL,

`c\_schema` text,

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_configinfo\_datagrouptenant` (`data\_id`,`group\_id`,`tenant\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='config\_info';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = config\_info\_aggr \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `config\_info\_aggr` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'id',

`data\_id` varchar(255) NOT NULL COMMENT 'data\_id',

`group\_id` varchar(255) NOT NULL COMMENT 'group\_id',

`datum\_id` varchar(255) NOT NULL COMMENT 'datum\_id',

`content` longtext NOT NULL COMMENT '内容',

`gmt\_modified` datetime NOT NULL COMMENT '修改时间',

`app\_name` varchar(128) DEFAULT NULL,

`tenant\_id` varchar(128) DEFAULT '' COMMENT '租户字段',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_configinfoaggr\_datagrouptenantdatum` (`data\_id`,`group\_id`,`tenant\_id`,`datum\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='增加租户字段';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = config\_info\_beta \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `config\_info\_beta` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'id',

`data\_id` varchar(255) NOT NULL COMMENT 'data\_id',

`group\_id` varchar(128) NOT NULL COMMENT 'group\_id',

`app\_name` varchar(128) DEFAULT NULL COMMENT 'app\_name',

`content` longtext NOT NULL COMMENT 'content',

`beta\_ips` varchar(1024) DEFAULT NULL COMMENT 'betaIps',

`md5` varchar(32) DEFAULT NULL COMMENT 'md5',

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '创建时间',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '修改时间',

`src\_user` text COMMENT 'source user',

`src\_ip` varchar(20) DEFAULT NULL COMMENT 'source ip',

`tenant\_id` varchar(128) DEFAULT '' COMMENT '租户字段',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_configinfobeta\_datagrouptenant` (`data\_id`,`group\_id`,`tenant\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='config\_info\_beta';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = config\_info\_tag \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `config\_info\_tag` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'id',

`data\_id` varchar(255) NOT NULL COMMENT 'data\_id',

`group\_id` varchar(128) NOT NULL COMMENT 'group\_id',

`tenant\_id` varchar(128) DEFAULT '' COMMENT 'tenant\_id',

`tag\_id` varchar(128) NOT NULL COMMENT 'tag\_id',

`app\_name` varchar(128) DEFAULT NULL COMMENT 'app\_name',

`content` longtext NOT NULL COMMENT 'content',

`md5` varchar(32) DEFAULT NULL COMMENT 'md5',

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '创建时间',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '修改时间',

`src\_user` text COMMENT 'source user',

`src\_ip` varchar(20) DEFAULT NULL COMMENT 'source ip',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_configinfotag\_datagrouptenanttag` (`data\_id`,`group\_id`,`tenant\_id`,`tag\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='config\_info\_tag';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = config\_tags\_relation \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `config\_tags\_relation` (

`id` bigint(20) NOT NULL COMMENT 'id',

`tag\_name` varchar(128) NOT NULL COMMENT 'tag\_name',

`tag\_type` varchar(64) DEFAULT NULL COMMENT 'tag\_type',

`data\_id` varchar(255) NOT NULL COMMENT 'data\_id',

`group\_id` varchar(128) NOT NULL COMMENT 'group\_id',

`tenant\_id` varchar(128) DEFAULT '' COMMENT 'tenant\_id',

`nid` bigint(20) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`nid`),

UNIQUE KEY `uk\_configtagrelation\_configidtag` (`id`,`tag\_name`,`tag\_type`),

KEY `idx\_tenant\_id` (`tenant\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='config\_tag\_relation';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = group\_capacity \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `group\_capacity` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`group\_id` varchar(128) NOT NULL DEFAULT '' COMMENT 'Group ID，空字符表示整个集群',

`quota` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '配额，0表示使用默认值',

`usage` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '使用量',

`max\_size` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '单个配置大小上限，单位为字节，0表示使用默认值',

`max\_aggr\_count` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '聚合子配置最大个数，，0表示使用默认值',

`max\_aggr\_size` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '单个聚合数据的子配置大小上限，单位为字节，0表示使用默认值',

`max\_history\_count` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '最大变更历史数量',

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '创建时间',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '修改时间',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_group\_id` (`group\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='集群、各Group容量信息表';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = his\_config\_info \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `his\_config\_info` (

`id` bigint(64) unsigned NOT NULL,

`nid` bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

`data\_id` varchar(255) NOT NULL,

`group\_id` varchar(128) NOT NULL,

`app\_name` varchar(128) DEFAULT NULL COMMENT 'app\_name',

`content` longtext NOT NULL,

`md5` varchar(32) DEFAULT NULL,

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00',

`src\_user` text,

`src\_ip` varchar(20) DEFAULT NULL,

`op\_type` char(10) DEFAULT NULL,

`tenant\_id` varchar(128) DEFAULT '' COMMENT '租户字段',

PRIMARY KEY (`nid`),

KEY `idx\_gmt\_create` (`gmt\_create`),

KEY `idx\_gmt\_modified` (`gmt\_modified`),

KEY `idx\_did` (`data\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='多租户改造';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 数据库全名 = nacos\_config \*/

/\* 表名称 = tenant\_capacity \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CREATE TABLE `tenant\_capacity` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`tenant\_id` varchar(128) NOT NULL DEFAULT '' COMMENT 'Tenant ID',

`quota` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '配额，0表示使用默认值',

`usage` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '使用量',

`max\_size` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '单个配置大小上限，单位为字节，0表示使用默认值',

`max\_aggr\_count` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '聚合子配置最大个数',

`max\_aggr\_size` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '单个聚合数据的子配置大小上限，单位为字节，0表示使用默认值',

`max\_history\_count` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '最大变更历史数量',

`gmt\_create` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '创建时间',

`gmt\_modified` datetime NOT NULL DEFAULT '2010-05-05 00:00:00' COMMENT '修改时间',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_tenant\_id` (`tenant\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='租户容量信息表';

CREATE TABLE `tenant\_info` (

`id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT 'id',

`kp` varchar(128) NOT NULL COMMENT 'kp',

`tenant\_id` varchar(128) default '' COMMENT 'tenant\_id',

`tenant\_name` varchar(128) default '' COMMENT 'tenant\_name',

`tenant\_desc` varchar(256) DEFAULT NULL COMMENT 'tenant\_desc',

`create\_source` varchar(32) DEFAULT NULL COMMENT 'create\_source',

`gmt\_create` bigint(20) NOT NULL COMMENT '创建时间',

`gmt\_modified` bigint(20) NOT NULL COMMENT '修改时间',

PRIMARY KEY (`id`),

UNIQUE KEY `uk\_tenant\_info\_kptenantid` (`kp`,`tenant\_id`),

KEY `idx\_tenant\_id` (`tenant\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='tenant\_info';

CREATE TABLE users (

username varchar(50) NOT NULL PRIMARY KEY,

password varchar(500) NOT NULL,

enabled boolean NOT NULL

);

CREATE TABLE roles (

username varchar(50) NOT NULL,

role varchar(50) NOT NULL,

constraint uk\_username\_role UNIQUE (username,role)

);

CREATE TABLE permissions (

role varchar(50) NOT NULL,

resource varchar(512) NOT NULL,

action varchar(8) NOT NULL,

constraint uk\_role\_permission UNIQUE (role,resource,action)

);

INSERT INTO users (username, password, enabled) VALUES ('nacos', '$2a$10$EuWPZHzz32dJN7jexM34MOeYirDdFAZm2kuWj7VEOJhhZkDrxfvUu', TRUE);

INSERT INTO roles (username, role) VALUES ('nacos', 'ROLE\_ADMIN');

（5）创建容器，这里的指定参数根据/home/nacos/conf/application.properties 配置设置的，主要是mysql配置的修改

docker run -d -p 8848:8848 \

-e MODE=standalone \

-e PREFER\_HOST\_MODE=hostname \

-e SPRING\_DATASOURCE\_PLATFORM=mysql \

-e MYSQL\_SERVICE\_HOST=127.0.0.1 \

-e MYSQL\_SERVICE\_PORT=3306 \

-e MYSQL\_SERVICE\_DB\_NAME=nacos\_config \

-e MYSQL\_SERVICE\_USER=root \

-e MYSQL\_SERVICE\_PASSWORD=root \

-e MYSQL\_DATABASE\_NUM=1 \

-v /root/nacos/init.d/custom.properties:/home/nacos/init.d/custom.properties \

-v /root/nacos/logs:/home/nacos/logs \

--restart always --name nacos nacos/nacos-server

（6）启动容器

docker start nacos

[docker-compose](https://cloud.tencent.com/developer/article/1499797" \o "docker-compose)启动只需要在 standalone-derby.yaml 配置文件的environment 属性中添加mysql的一些配置即可

（7）访问nacos

访问[http://localhost:8848/nacos/](http://localhost:8848/nacos/" \o "http://localhost:8848/nacos/) 账号默认nacos、密码默认nacos

### 在ECS服务器上通过Docker安装MongoDB

（1）创建主机挂载配置目录

mkdir -p /docker/mongodb/data && cd /docker/mongodb

（2）生成启动文件

cat <<EOF> start.sh

#!/bin/bash

MONGODB\_DIR=`pwd`

docker stop mongodb

docker rm mongodb

docker run -d \\

--name mongodb \\

--restart always \\

--privileged \\

-p 27017:27017 \\

-v \${MONGODB\_DIR}/data:/data/db \\

-e MONGO\_INITDB\_ROOT\_USERNAME=admin \\

-e MONGO\_INITDB\_ROOT\_PASSWORD=123456 \\

mongo:4.2.2 mongod --auth

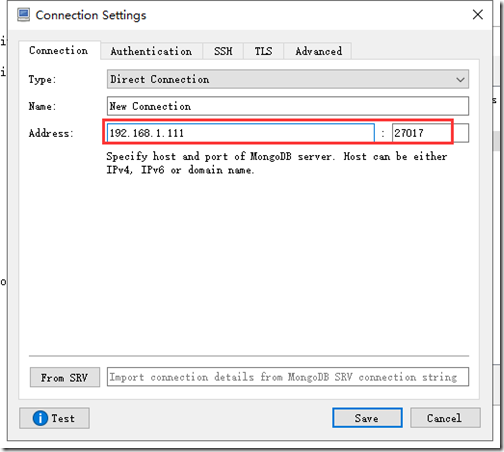
EOF

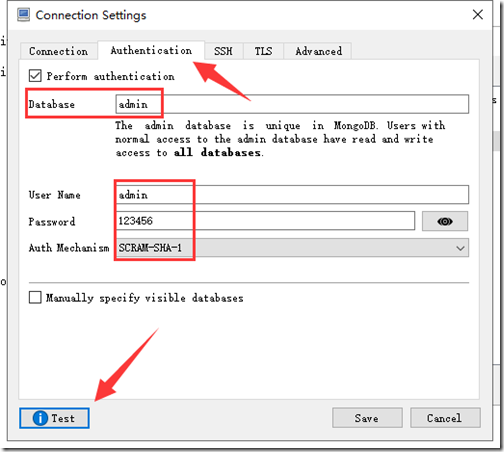
（3）运行

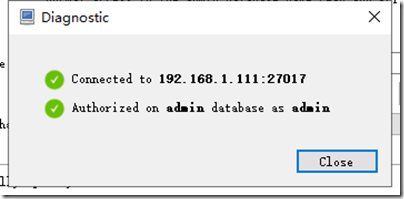
sh start.sh

停止和删除容器  
docker stop mongodb && docker rm mongodb

（4）使用Robo连接







5. 进入容器

[root@localhost mongodb]# docker exec -it mongodb bash

root@ce90018683a8:/# mongo --version

MongoDB shell version v4.2.2

git version: a0bbbff6ada159e19298d37946ac8dc4b497eadf

OpenSSL version: OpenSSL 1.1.1 11 Sep 2018

allocator: tcmalloc

modules: none

build environment:

distmod: ubuntu1804

distarch: x86\_64

target\_arch: x86\_64

root@ce90018683a8:/#

6. 进入mongodb

在连接期间进行身份验证，使用-u <username>，-p <password>和--authenticationDatabase <database>命令行选项启动一个mongo shell

root@ce90018683a8:/# mongo --port 27017 -u "admin" -p "123456" --authenticationDatabase "admin"

MongoDB shell version v4.2.2

connecting to: mongodb://127.0.0.1:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("275dac83-d84c-4147-bb4c-9516cedc534a") }

MongoDB server version: 4.2.2

Server has startup warnings:

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten]

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten] \*\* WARNING: /sys/kernel/mm/transparent\_hugepage/enabled is 'always'.

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten] \*\* We suggest setting it to 'never'

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten]

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten] \*\* WARNING: /sys/kernel/mm/transparent\_hugepage/defrag is 'always'.

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten] \*\* We suggest setting it to 'never'

2022-08-29T03:39:16.657+0000 I CONTROL [initandlisten]

---

Enable MongoDB's free cloud-based monitoring service, which will then receive and display

metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you

and anyone you share the URL with. MongoDB may use this information to make product

improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()

To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

---

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

>

7. 以admin用户身份进入，先连接后验证

将 mongo shell 连接到 mongodb，也就是先连接，后验证用户身份

root@ce90018683a8:/# mongo

MongoDB shell version v4.2.2

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("6e560085-2f03-4b6a-81a6-b42cd5b03f96") }

MongoDB server version: 4.2.2

> use admin

switched to db admin

> db.auth("admin","123456")

1

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

>

8. 使用命令创建数据库

# 进入admin数据库

[root@localhost mongodb]# docker exec -it mongodb mongo admin

MongoDB shell version v4.2.2

connecting to: mongodb://127.0.0.1:27017/admin?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("91508ef9-195d-4a87-a8ae-c36ee3dfcd12") }

MongoDB server version: 4.2.2

# 输入账号密码认证，返回1说明认证成功

> db.auth("admin", "123456")

1

# 查看所有数据库

> show dbs

admin 0.000GB

config 0.000GB

local 0.000GB

# 创建新数据库

> use oyz

switched to db oyz

# 创建 和新创建的数据库 绑定的用户

> db.createUser({ user: 'haolb', pwd: 'haolb123456', roles: [ { role: "readWrite", db: "oyz" } ] });

Successfully added user: {

"user" : "haolb",

"roles" : [

{

"role" : "readWrite",

"db" : "oyz"

}

]

}

# exit退出当前用户，否则继续认证新的用户会报错 too many users are authenticated

> exit

bye

# 进入 oyz 数据库

[root@localhost mongodb]# docker exec -it mongodb mongo oyz

MongoDB shell version v4.2.2

connecting to: mongodb://127.0.0.1:27017/oyz?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("b9a9ea61-a15f-43d8-9e4a-dc2bfa0bdb05") }

MongoDB server version: 4.2.2

# 重新认证新的用户

> db.auth("haolb","haolb123456")

1

# 随便添加一条信息才算创建成功

> db.oyz.insert({"name":"abc1111"});

WriteResult({ "nInserted" : 1 })

> show dbs

oyz 0.000GB

>

9. 使用命令删除用户

# 切换 admin 库

> use admin

switched to db admin

> db.auth("admin","123456")

1

# 创建一个 myuser 用户

> db.createUser({user: "myuser",pwd: "my123",roles: [ { role: "root", db: "admin" } ]})

Successfully added user: {

"user" : "myuser",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

]

}

> db.auth("myuser","my123")

1

> show users

{

"\_id" : "admin.admin",

"userId" : UUID("c19fdb6d-efe1-4398-b32a-77ef8c12bac3"),

"user" : "admin",

"db" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

],

"mechanisms" : [

"SCRAM-SHA-1",

"SCRAM-SHA-256"

]

}

{

"\_id" : "admin.myuser",

"userId" : UUID("ec82e9f7-9f81-4a2a-b10b-8368d4750e6c"),

"user" : "myuser",

"db" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

],

"mechanisms" : [

"SCRAM-SHA-1",

"SCRAM-SHA-256"

]

}

# 删除单个用户：myuser

> db.system.users.remove({user:"myuser"})

WriteResult({ "nRemoved" : 1 })

# 切换 admin 用户登录

> db.auth("admin","123456")

1

# 显示用户信息

> show users

{

"\_id" : "admin.admin",

"userId" : UUID("c19fdb6d-efe1-4398-b32a-77ef8c12bac3"),

"user" : "admin",

"db" : "admin",

"roles" : [

{

"role" : "root",

"db" : "admin"

}

],

"mechanisms" : [

"SCRAM-SHA-1",

"SCRAM-SHA-256"

]

}

>

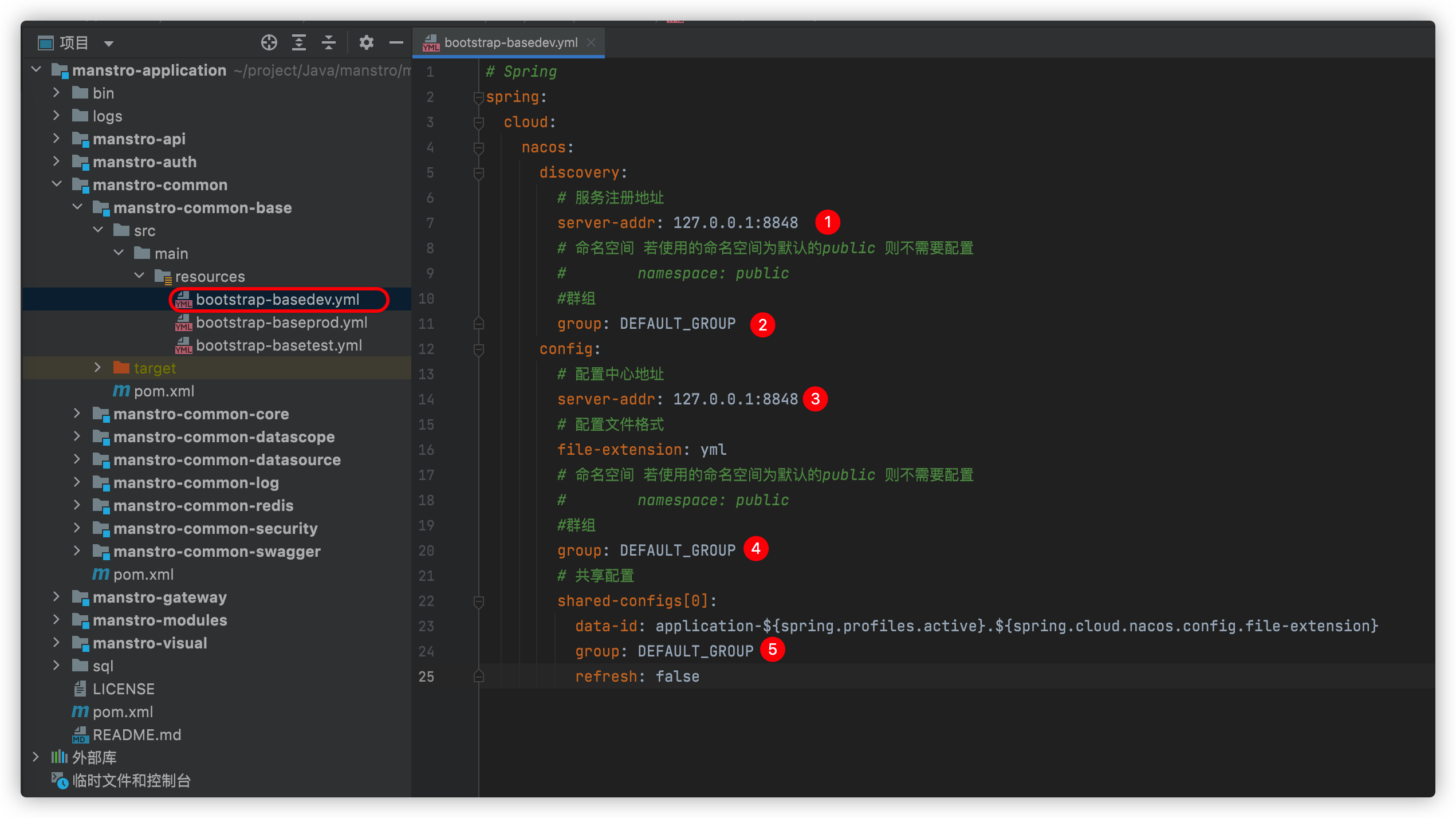
## Jenkins自动化部署应用

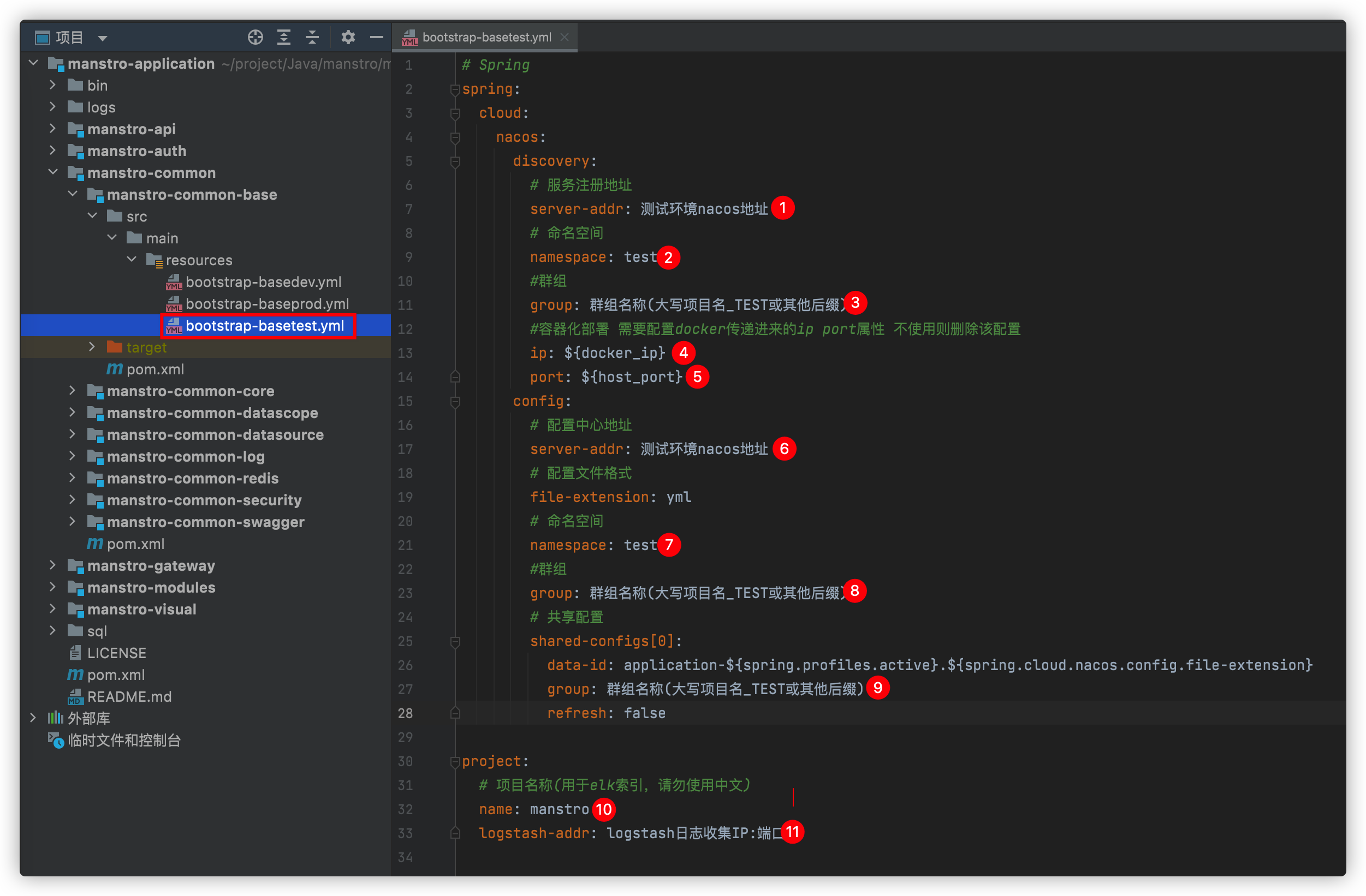
### 后端应用配置

（1）manstro-common/manstro-common-base/src/main/resources/bootstrap-base[maven环境].yml

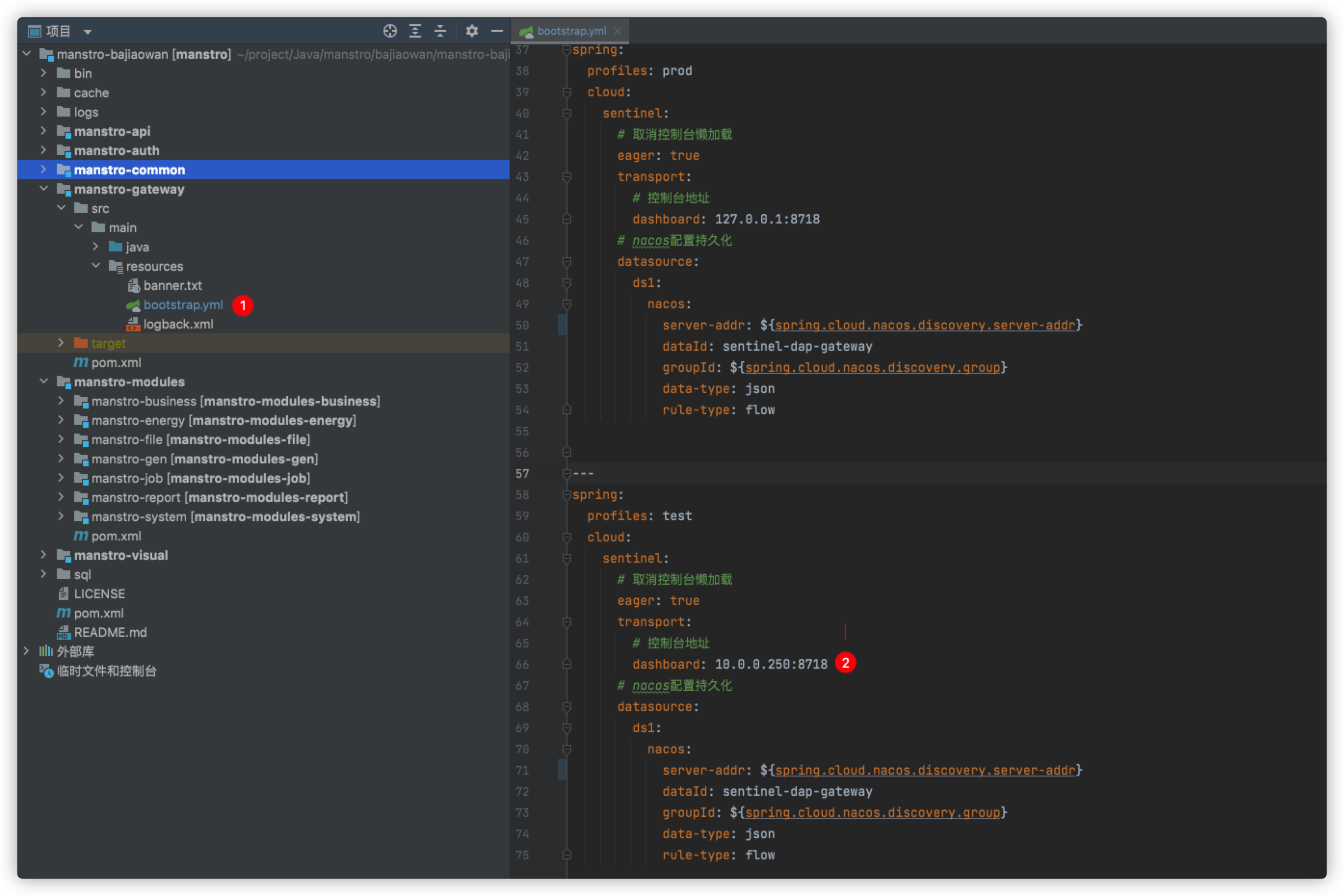
注：

* 命名空间用于区分开发、生产、测试环境，可与其他项目共用，自定义命名空间需在nacos管理页面提前配置，如果命名空间为默认的public，就不要配置namespace属性
* 群组用于区分项目，不可与其他项目共用
* ip、port为docker容器化时传入，容器化部署时必须配置，且docker命令中必须传入
* 其他按需配置即可
* test环境使用docker部署及elk日志收集，配置不同于dev环境





（2）manstro-gateway/src/main/resources/bootstrap.yml



（3）nacos配置

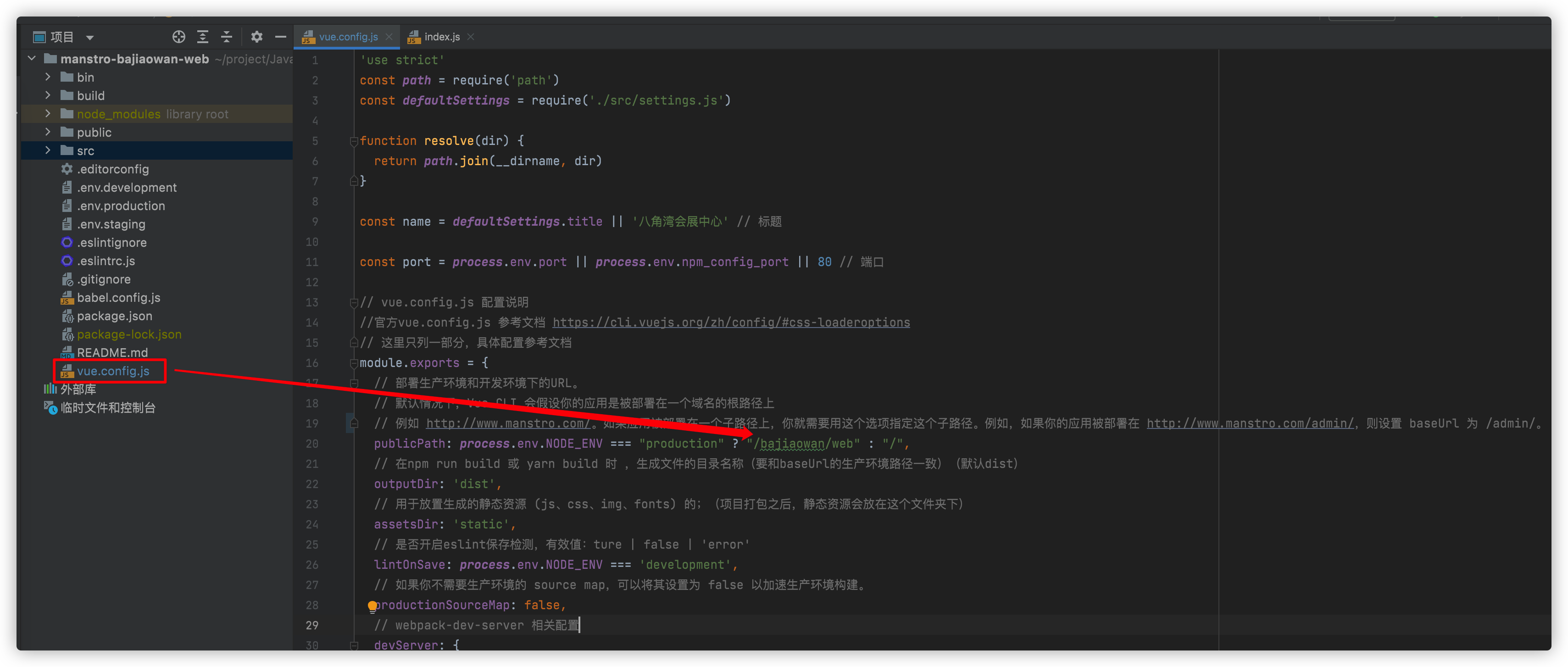
* 在yml文件配置的namespace中添加配置列表，配置列表的Group需要与yml文件spring.cloud.nacos.config.group一致
* 配置列表文件配置数据库连接信息、redis连接信息、file配置的文件保存路径(与jenkins docker脚本中的-v文件路径一致)、auth配置的证书所在目录(按需)



### 前端应用配置

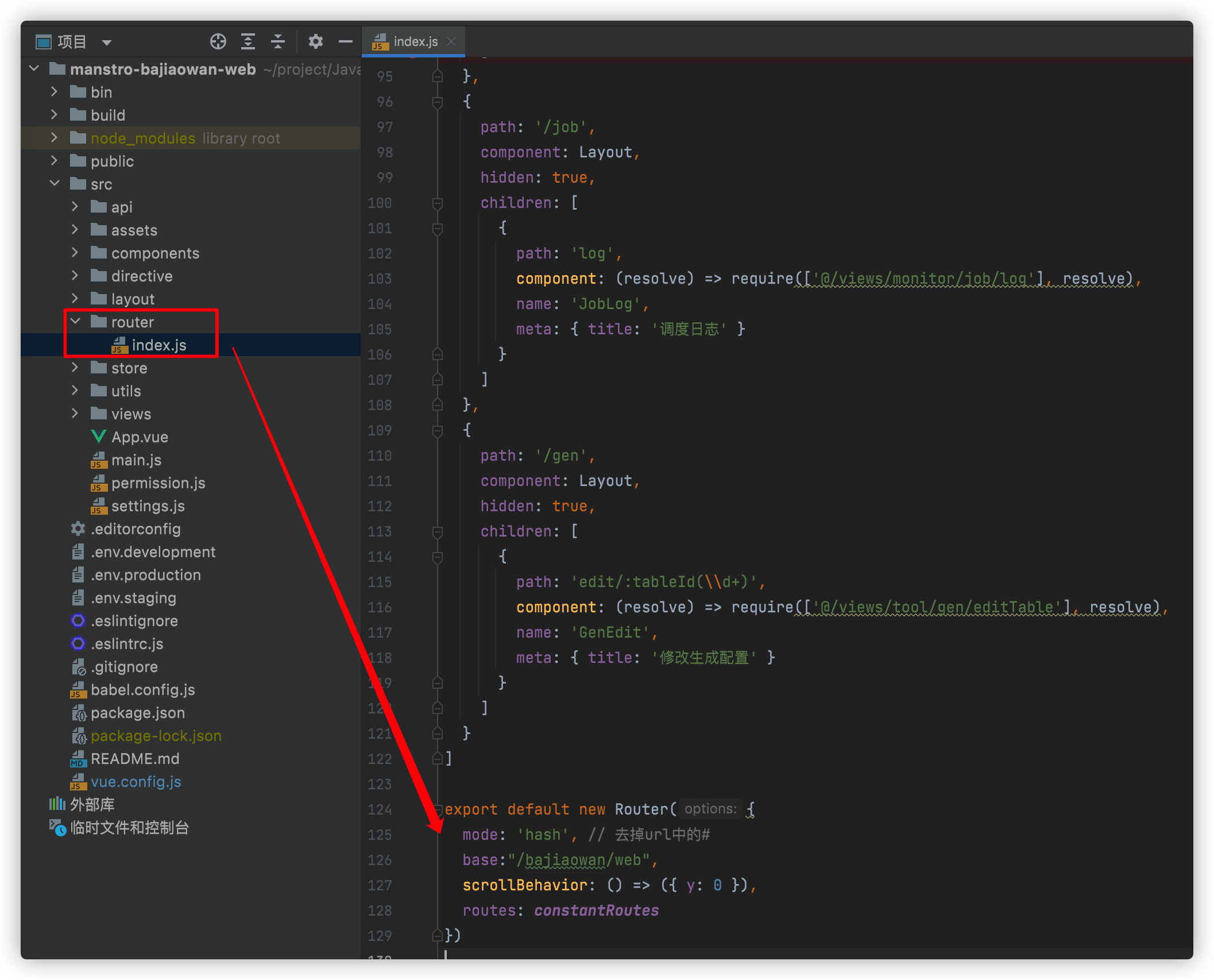
（1）vue.config

修改请求路径配置，默认为/



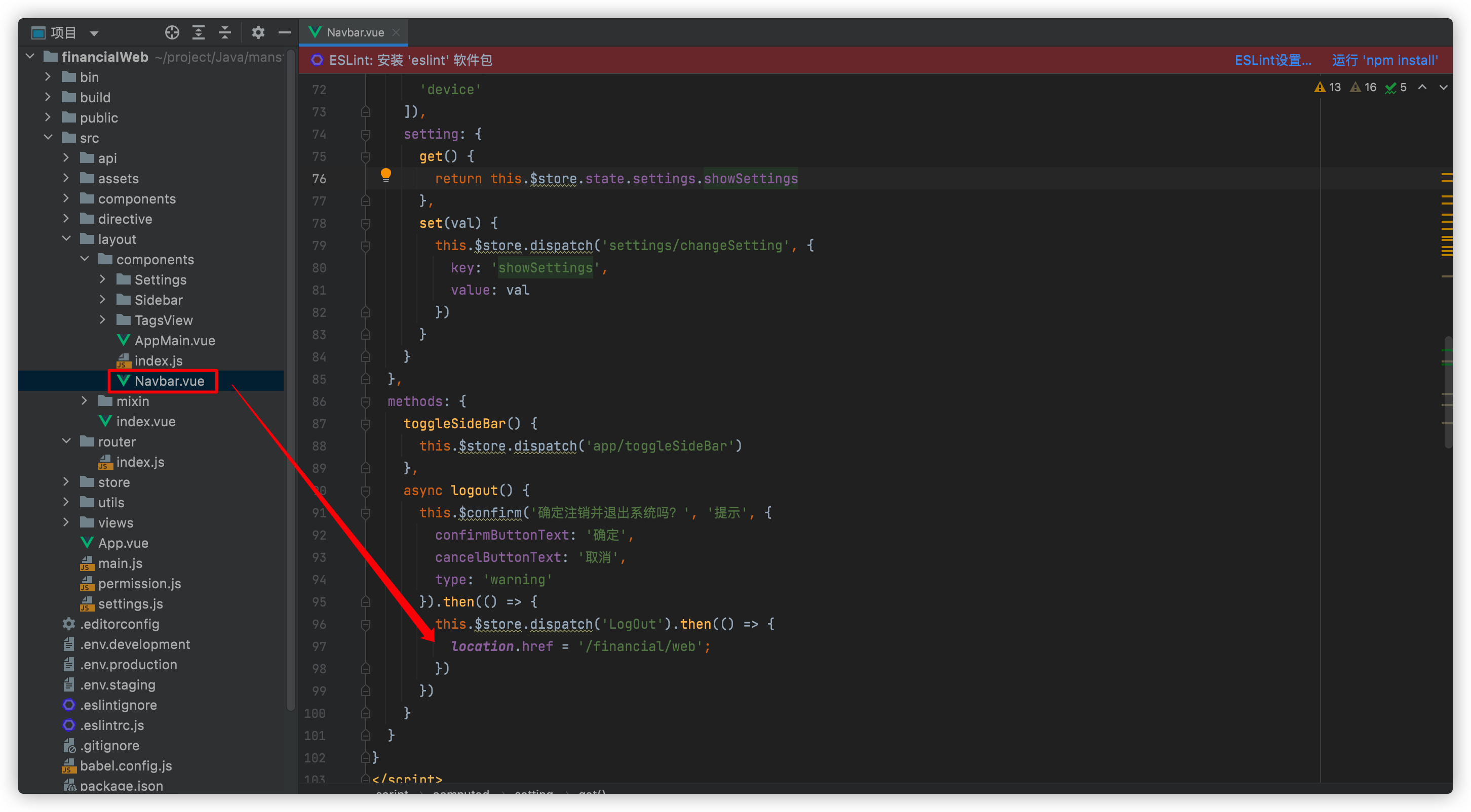
**（2）src/router/index.js**

修改路由模式为hash，修改base为请求路径



**（3）src/ layout/ components/Navbar.vue**

修改退出的跳转页面，默认为/，需要与nginx路径对应



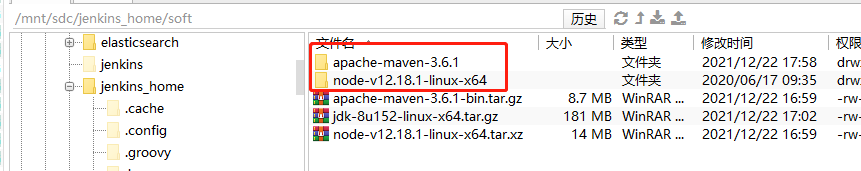
### 安装jenkins

#下载镜像

docker pull jenkins/jenkins:2.326

#将maven、node、jdk放入jenkins\_home某文件夹下

#进入容器将软件解压、安装



#启动命令并限制内存大小，启动后文件会挂载到容器

docker run --restart=always -d -p 8099:8080 -e JENKINS\_JAVA\_OPTIONS="-XX:MaxPermSize=1024m -Djava.awt.headless=true" -v /mnt/sdc/jenkins\_home/:/var/jenkins\_home/ -u 0 --name=jenkins jenkins/jenkins:2.326

#启动后进入容器

docker exec -it Jenkins /bin/bash

#由于容器只有核心的运行环境，没有完整的运行系统，所以需要执行以下命令

apt-get update

apt-get install vim

#配置环境变量

vi /etc/profile

export MAVEN\_HOME=/var/Jenkins\_home/soft/apache-maven-3.6.1

export PATH=$MAVEN\_HOME/bin:$PATH

export PATH=$PATH:/var/Jenkins\_home/soft/node-v12.18.1-linux-x64/bin

source /etc/profile



### 安装配置nodejs

# 下载nodejs

wget https://nodejs.org/dist/v12.18.1/node-v12.18.1-linux-x64.tar.xz

# 解压

tar xf node-v12.18.1-linux-x64.tar.xz

cd node-v12.18.1-linux-x64

# 修改配置文件

Vi /etc/profile

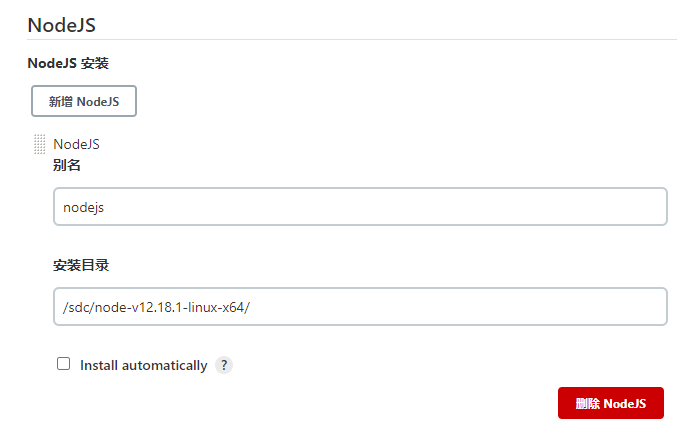
export PATH=$PATH:/sdc/node-v12.18.1-linux-x64/bin

source /etc/profile

node -v

# 切换淘宝镜像

npm config set registry <http://registry.npm.taobao.org/>



### 配置jdk



### 配置maven

在容器中将maven解压，注意修改仓库路径



### 配置git

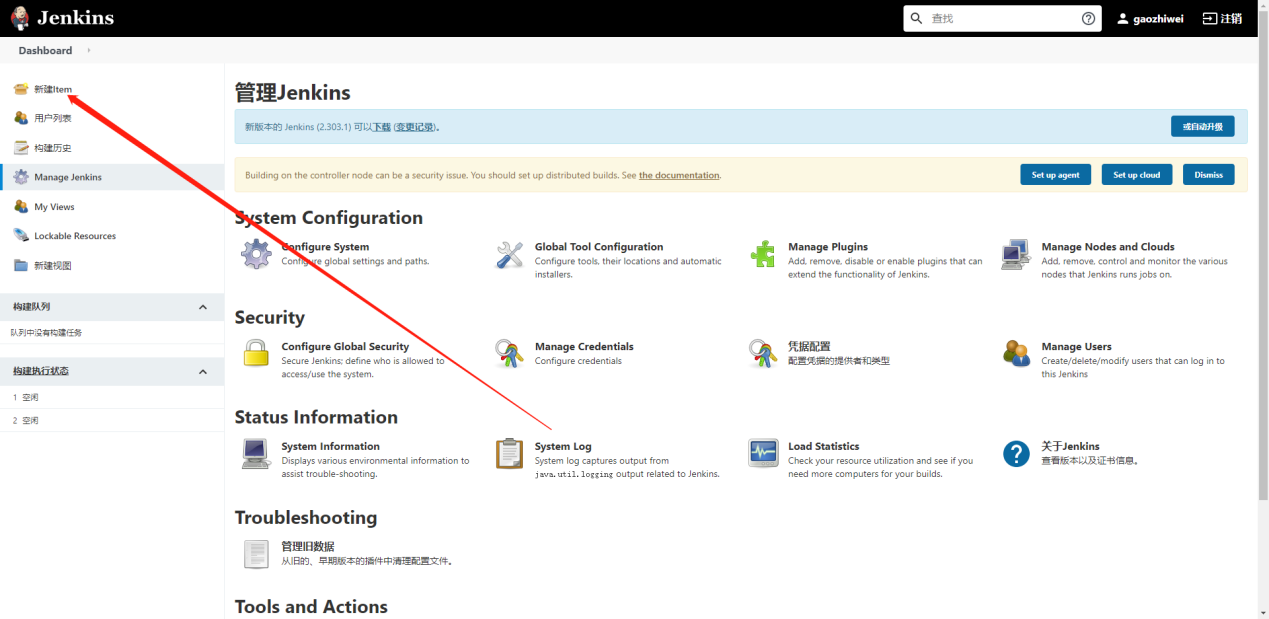
容器自带git，输入whereis git查询git路径配置到jenkins即可

### 设置时区

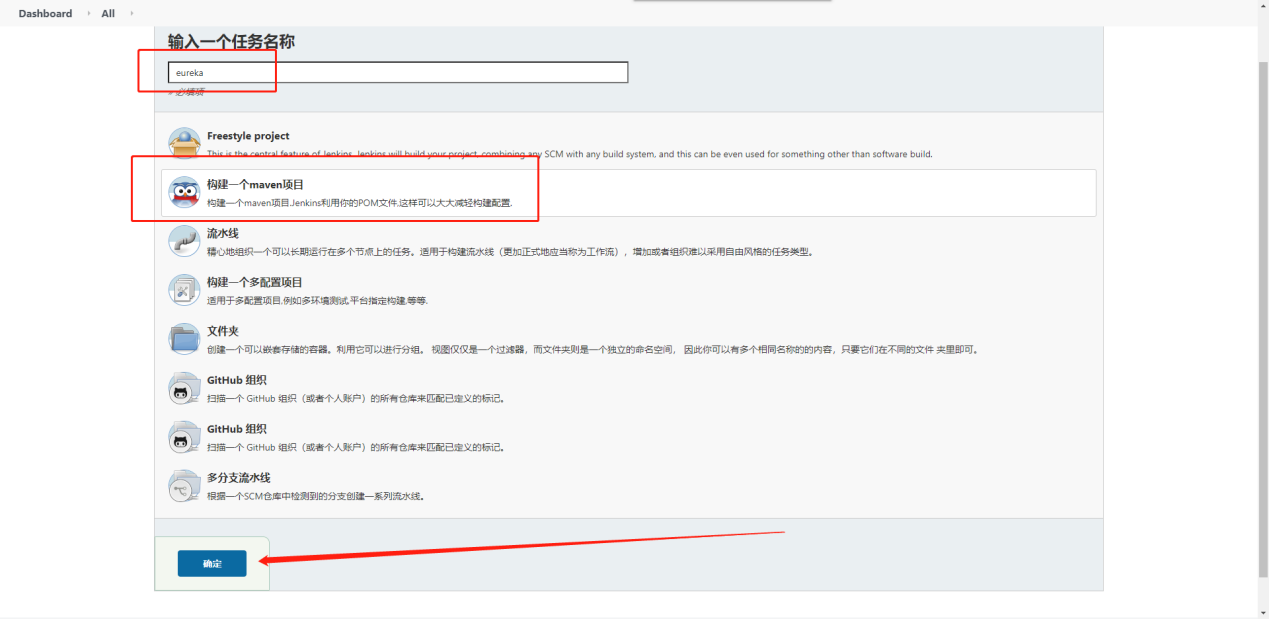
* 访问【Jenkins】。
* 点击【Manage Jenkins】选项。
* 点击【Script Console】选项
* 输入【System.setProperty('org.apache.commons.jelly.tags.fmt.timeZone', 'Asia/Shanghai')】。
* 点击【运行】选项。

### 后端新建item

（1）点击新建item



（2）输入任务名称，选择“构建一个maven项目”



（3）配置item

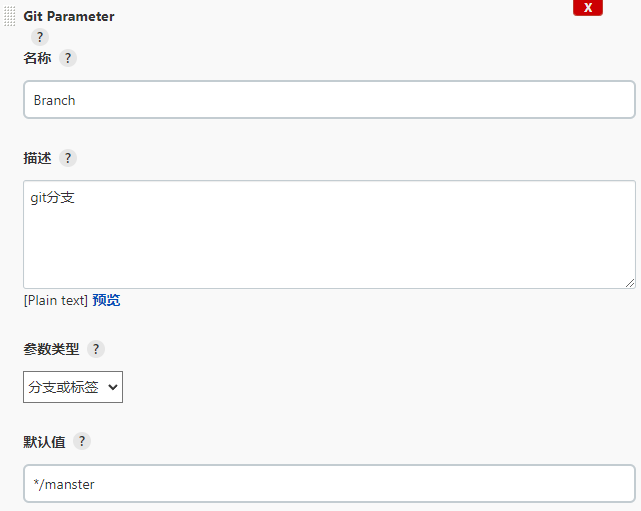
* Discard old builds

Jenkins在每一次的执行构建后，都会对该构建的项目生成一个历史构建记录以及生成一份历史构建的项目发布包，硬盘容量不够大建议不要输入数值太大。



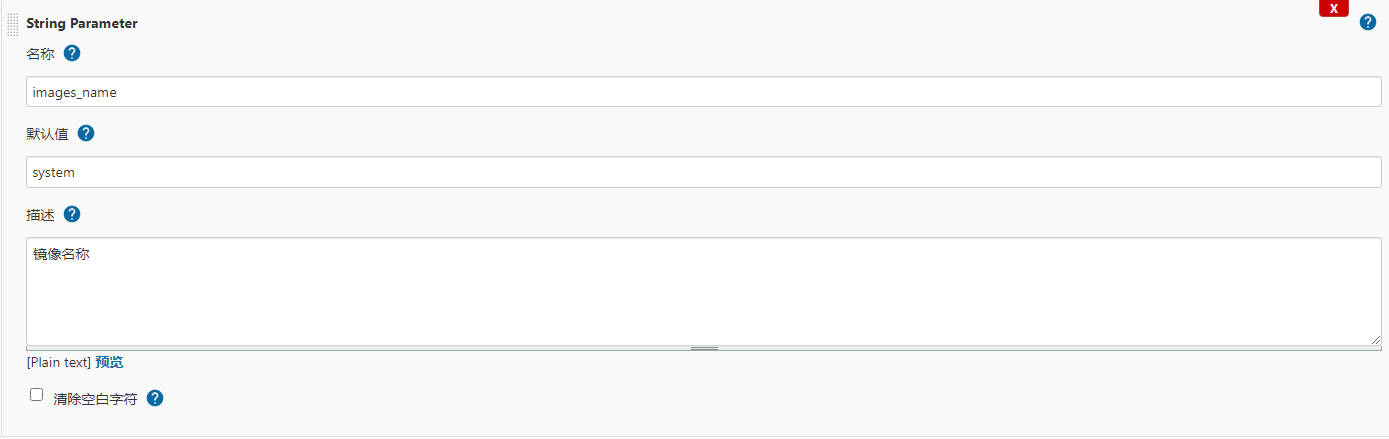
* 添加参数化变量
* Branch（必配）

选择git分支



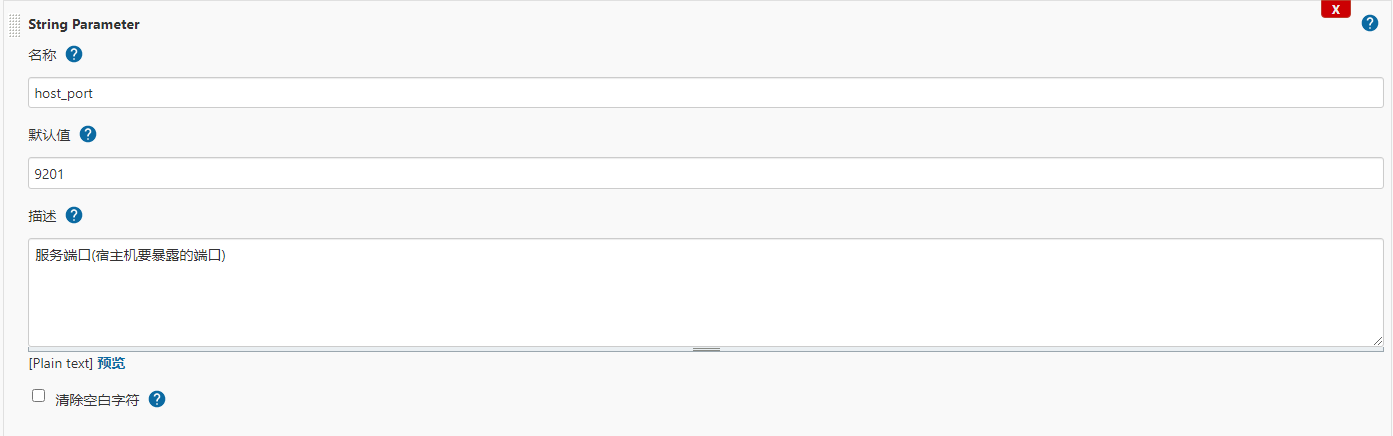
* images\_name（选配）

镜像名称，可根据镜像名称关闭容器；可以定义构建镜像名称、容器启动名称；与容器名称公用



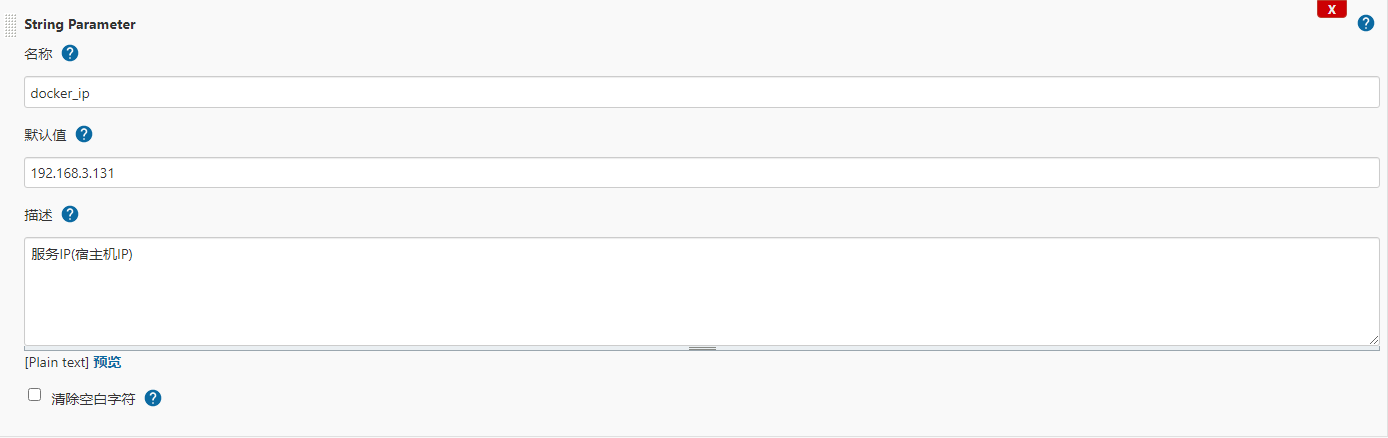
* host\_port（必配）

服务端口，用来定义宿主机暴露的端口；可以传入yml文件中名称相同的占位符参数；可用于扩容服务（具体作用可在6.springboot yml和8.docker扩容中查看）



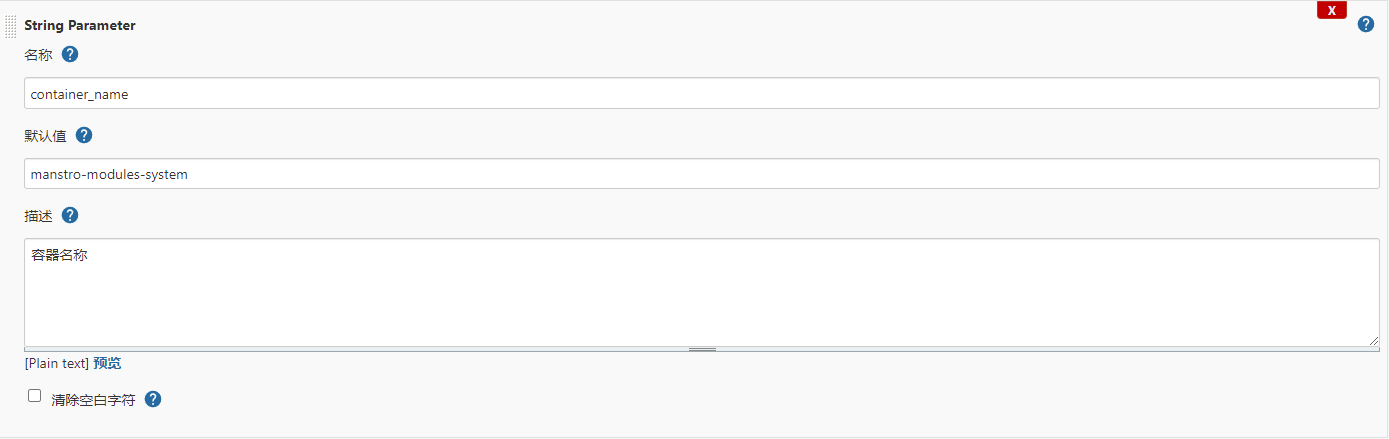
* docker\_ip（必配）

服务IP，定义宿主机IP（发布在哪台IP就是多少）以及传入yml相同名称占位符的参数（具体作用可在6.springboot yml和8.docker扩容中查看）



* container\_name（选配）

容器名称，定义启动容器名称；可用于扩容服务



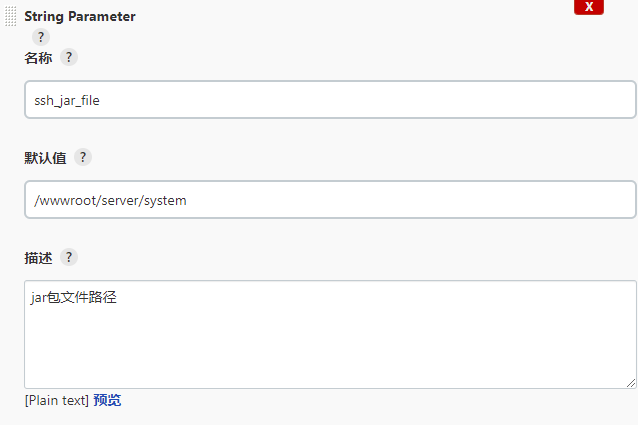
* profile\_flag（必配）

运行环境选择



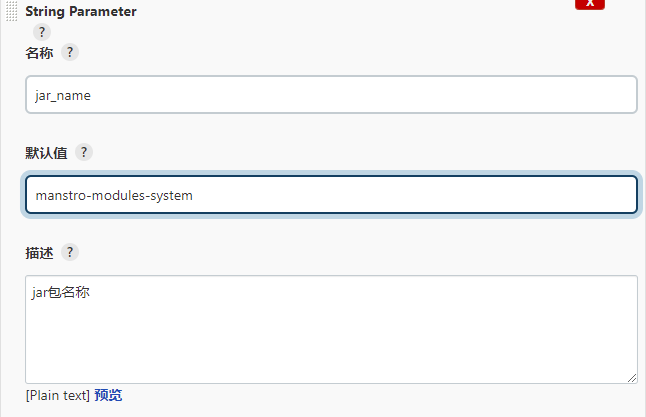
* ssh\_jar\_file（选配）

jar包文件路径

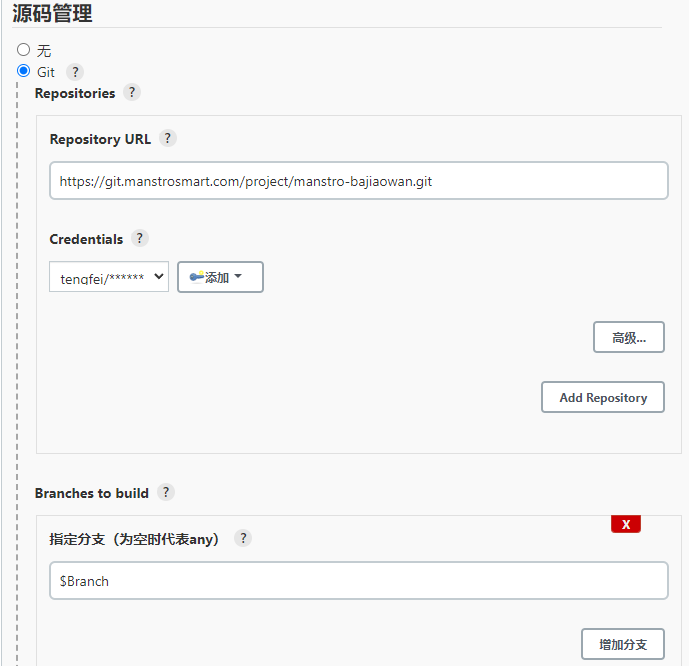


* jar\_name（选配）

jar包名称

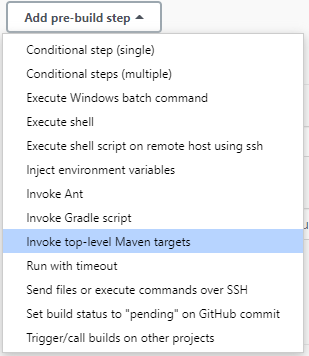


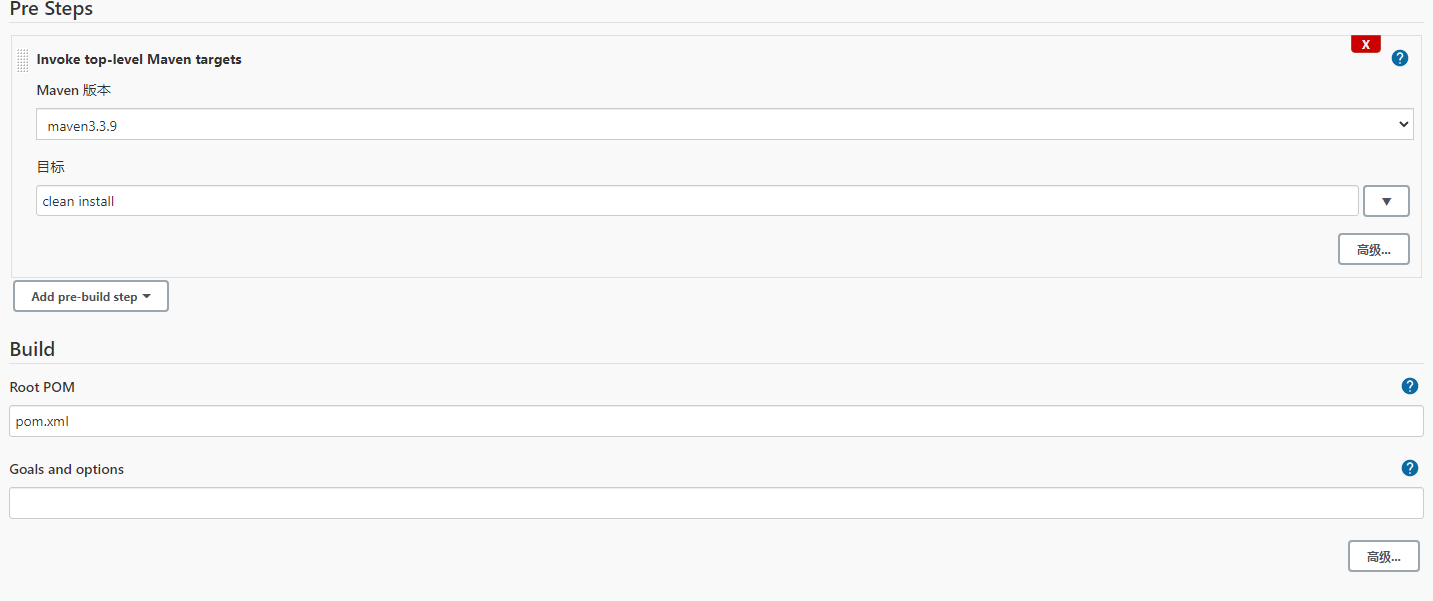
* 配置源码管理



* 配置Pre Steps

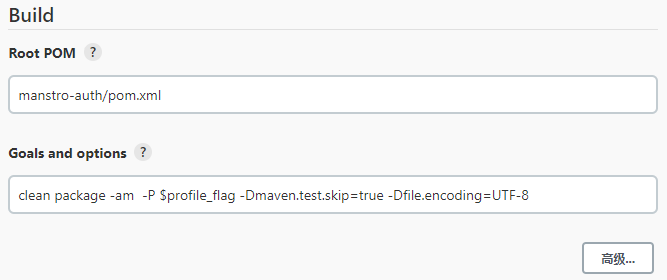
调用顶级maven目标，拉取代码后优先执行此配置





* 配置Build

打包命令执行完成后根据build配置所在目录的pom文件进行docker镜像构建

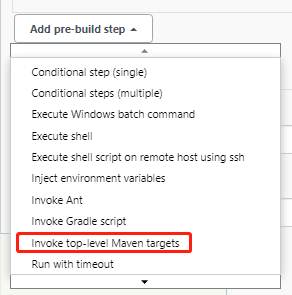


clean package -am -P $profile\_flag -Dmaven.test.skip=true -Dfile.encoding=UTF-8

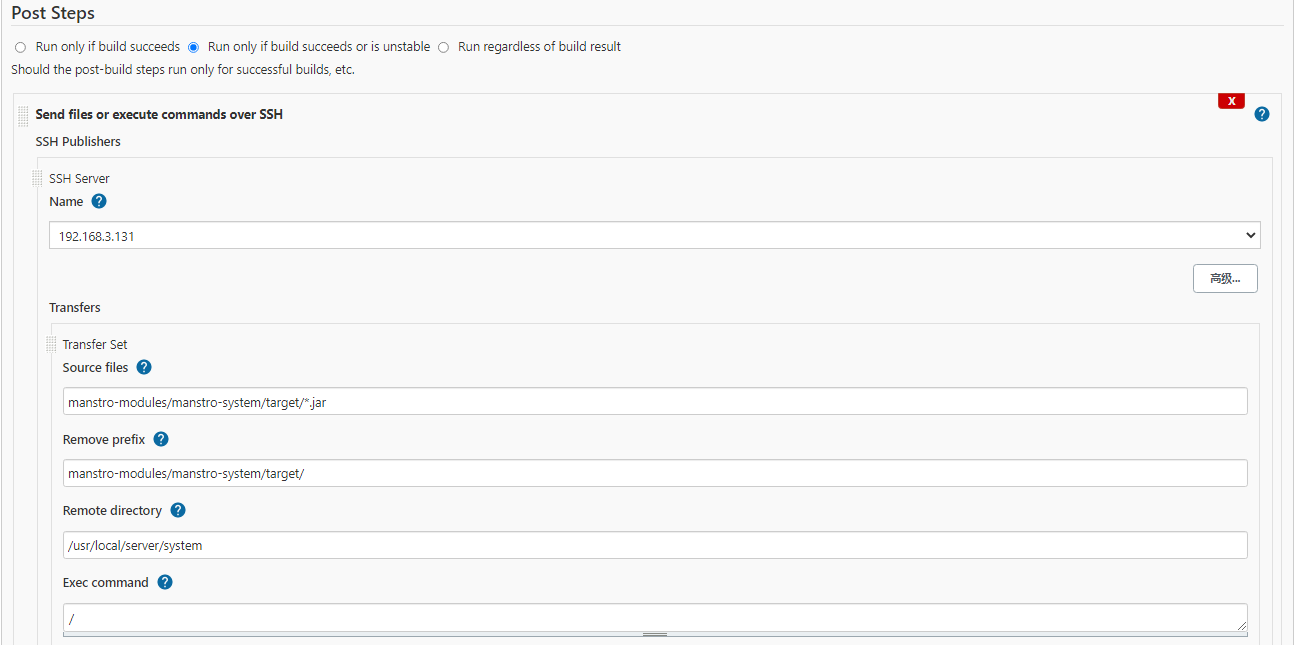
补充：

配置Invoke top-level maven targets

调用顶级maven目标，优先级高于build，配置此项后会先执行此项再执行build，可根据自己需求进行配置



* 配置Send files or execute commands over SSH



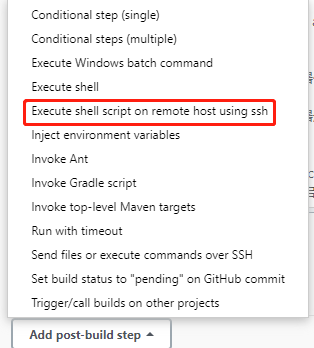
Source files要上传到服务器的文件

Remove profix删除前缀

Remote directory目标文件夹（远程服务器）

* 配置Execute shell script on remote host using ssh

在远程主机执行脚本命令



**echo '================1.判断容器是否运行================'**

**if [[ -n $(docker ps -a | grep $container\_name) ]];then**

**echo '================1.1容器正在运行================'**

**echo '================1.2停止容器================'**

**docker stop $container\_name**

**echo '================1.3强制删除容器================'**

**docker rm -f $container\_name**

**echo '================1.4强制删除镜像================'**

**docker rmi -f $images\_name**

**else**

**echo '================2.1容器未运行================'**

**fi**

**echo '================2.进入jar包文件夹================'**

**cd $ssh\_jar\_file**

**echo '================3.重新写入配置================'**

**echo "FROM centos7-jdk8u152" > Dockerfile**

**echo '# 指定字符集'**

**echo -e "ENV LC\_ALL \"zh\_CN.UTF-8\"" >> Dockerfile**

**echo '# 设置时区'**

**echo -e "RUN /bin/cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo 'Asia/Shanghai' >/etc/timezone" >> Dockerfile**

**echo '# 将本地文件挂载到当前容器'**

**echo "VOLUME /tmp" >> Dockerfile**

**echo '# 复制文件到容器'**

**echo "ADD ./$jar\_name.jar /usr/local/$jar\_name.jar" >> Dockerfile**

**echo '# 声明暴露的端口号'**

**echo "EXPOSE $host\_port">> Dockerfile**

**echo '# 配置容器启动后执行的命令'**

**echo -e "ENTRYPOINT [\"java\",\"-jar\",\"-Xmx512m\",\"-Dfile.encoding=utf-8\",\"/usr/local/$jar\_name.jar\"]" >> Dockerfile**

**echo '================4.打包镜像================'**

**docker build -t $images\_name .**

**echo '================5.启动容器================'**

**docker run -p $host\_port:$host\_port -e docker\_ip="$docker\_ip" -e host\_port="$host\_port" --name $container\_name -v /mnt/sdc/bajiaowan/logs:/logs -d $images\_name:latest**

* 点击保存，jenkins配置完成

### 前端新建item

（1）选择“Freestyle project（自由项目）”



（2）创建以下参数



说明：

file\_path：jenkins工作区目录，代码拉取、打包路径。

Profile\_flag：前端打包环境，npm run build:$profile\_flag。

Images\_name：与3.3.2.2相同。

host\_port：与3.3.2.3相同，注意这里是nginx容器对应的宿主机端口。

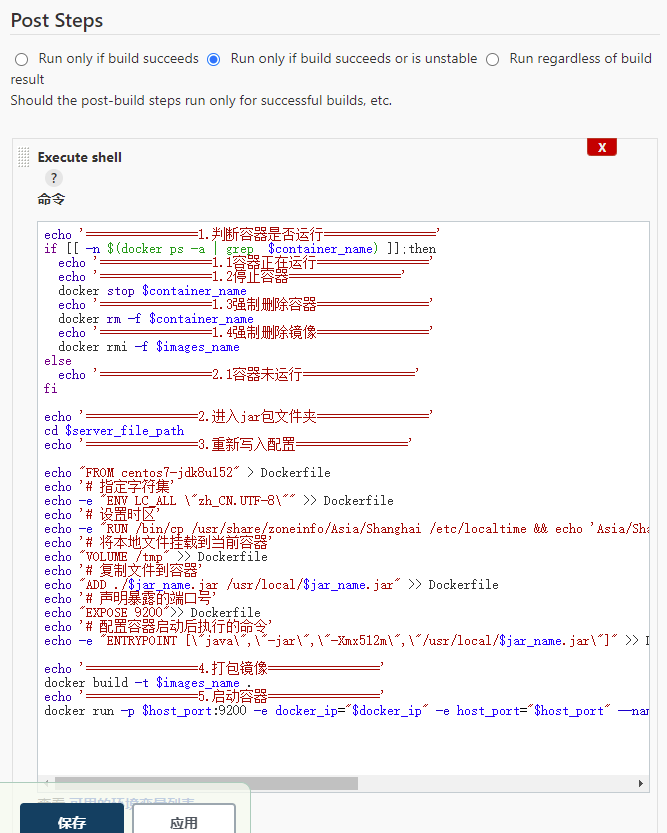
container\_name：与3.3.2.5相同。

project\_name：改为自己的项目名称，nginx为各项目共用容器，如果一个项目一个ningx，多个质量部署测试环境存在容器管理混乱，暴露端口过多等问题。共用后只需要配置不同的访问路径匹配不同的静态资源即可，便于集中管理和维护。

（3）其他配置







npm install

npm run build:$profile\_flag

cd /sdc/nginx/html/$project\_name

rm -rf web/

cd $file\_path

cp -r dist /sdc/nginx/html/$project\_name/web

echo '================1.判断容器是否运行================'

if [[ -n $(docker ps -a | grep $container\_name) ]];then

echo '================1.1容器正在运行================'

echo '================1.2停止容器================'

docker stop $container\_name

echo '================1.3强制删除容器================'

docker rm -f $container\_name

else

echo '================2.1容器未运行================'

fi

echo '================5.启动容器================'

docker run --restart=always -d -p $host\_port:8081 -v /sdc/nginx/html:/usr/share/nginx/html -v /sdc/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v /sdc/nginx/logs:/var/log/nginx --name $container\_name $images\_name:latest

（4）nginx.conf配置





配置1：后台管理前端路由，解决刷新页面路由失效问题，多项目只需要配置一个。

配置2：后台管理前端静态资源访问路径；与jenkins配置的文件复制路径一致。

配置3：后台管理前端代理转发，前端不允许直接访问网关；路径与前端框架配置一致。

配置4：IOC静态资源访问路径；与jenkins配置的文件复制路径一致。

配置5：IOC代理转发，前端不允许直接访问网关；路径与前端框架配置一致。

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log notice;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65;

server {

listen 8081;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

try\_files $uri $uri/ /index.html;

}

location /bajiaowan/web {

gzip on; #开启压缩

gzip\_http\_version 1.1; #协议版本配置

gzip\_min\_length 50k; #低于5kb的资源不压缩

gzip\_comp\_level 1; #压缩等级

gzip\_types text/plain application/xml application/javascript text/css image/png application/json; #需要压缩的MIME类型

gzip\_disable "MSIE [1-6]\."; #配置禁用gzip条件，支持正则。此处表示ie6及以下不启用gzip（因为ie低版本不支持）

gzip\_vary on; #是否添加“Vary: Accept-Encoding”响应头

root /usr/share/nginx/html;

try\_files $uri $uri/ /index.html;

index index.html;

}

location /bajiaowan/prod-api/{

gzip on; #开启压缩

gzip\_http\_version 1.1; #协议版本配置

gzip\_min\_length 50k; #低于5kb的资源不压缩

gzip\_comp\_level 1; #压缩等级

gzip\_types text/plain application/xml application/javascript text/css image/png application/json; #需要压缩的MIME类型

gzip\_disable "MSIE [1-6]\."; #配置禁用gzip条件，支持正则。此处表示ie6及以下不启用gzip（因为ie低版本不支持）

gzip\_vary on; #是否添加“Vary: Accept-Encoding”响应头

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header REMOTE-HOST $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://10.0.0.250:8080/;

}

location /bajiaowan/ioc {

gzip on; #开启压缩

gzip\_http\_version 1.1; #协议版本配置

gzip\_min\_length 50k; #低于5kb的资源不压缩

gzip\_comp\_level 1; #压缩等级

gzip\_types text/plain application/xml application/javascript text/css image/png application/json; #需要压缩的MIME类型

gzip\_disable "MSIE [1-6]\."; #配置禁用gzip条件，支持正则。此处表示ie6及以下不启用gzip（因为ie低版本不支持）

gzip\_vary on; #是否添加“Vary: Accept-Encoding”响应头

root /usr/share/nginx/html;

try\_files $uri $uri/ /index.html;

index index.html index.htm;

}

location /bajiaowan/ioc-api/{

gzip on; #开启压缩

gzip\_http\_version 1.1; #协议版本配置

gzip\_min\_length 50k; #低于5kb的资源不压缩

gzip\_comp\_level 1; #压缩等级

gzip\_types text/plain application/xml application/javascript text/css image/png application/json; #需要压缩的MIME类型

gzip\_disable "MSIE [1-6]\."; #配置禁用gzip条件，支持正则。此处表示ie6及以下不启用gzip（因为ie低版本不支持）

gzip\_vary on; #是否添加“Vary: Accept-Encoding”响应头

proxy\_set\_header Host $http\_host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header REMOTE-HOST $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://10.0.0.250:8080/;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

}

## Prometheus应用监控系统部署

### 安装node-exporter

node-exporter是prometheus一款节点监控插件，可用于监控主机节点 注意：你要监控哪个主机，就在哪个主机安装node-exporter插件，此插件专门用来监控主机级应用

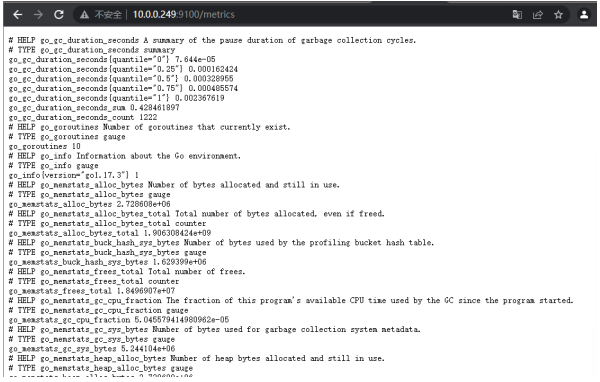
* 拉取镜像

docker pull prom/node-exporter

* 运行容器

docker run -d -p 9100:9100 \ -v "/proc:/host/proc:ro" \ -v "/sys:/host/sys:ro" \ -v "/:/rootfs:ro" \ --net="host" \ prom/node-exporter

* 访问ip:9100/metrics查看，效果如下



### 安装prometheus

* 下载镜像，创建prometheus挂载目录

docker pull prom/prometheus

mkdir /opt/prometheus

* 进入目录

cd /opt/prometheus/

* 编辑prometheus配置文件，写入如下内容

vim prometheus.yml

global:

scrape\_interval: 60s #设置prometheus抓取数据间隔，一般设置为10-60s evaluation\_interval: 60s #设置告警周期，多长时间告警一次

scrape\_configs:

- job\_name: prometheus #这里的job名称会显示在prometheus页面Lables标 签下，可自定义 s

tatic\_configs:

- targets: ['localhost:9090'] #用于监控prometheus本身，写本机+prometheus 的端口号

labels:

instance: prometheus #标签，会显示在prometheus页面Lables标签下， 可自定义

- job\_name: linux

static\_configs:

- targets: ['10.0.0.249:9100','10.0.0.251:9100'] #要监控哪台主机，就写哪台ip， 多台主机可添加多个ip

* 启动prometheus

docker run -d \

-p 9090:9090 \

-v /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml \ prom/Prometheus

* 访问ip:9090/targets，效果如下



### 安装grafana

* 下载镜像，新建空文件夹grafana-storage，用来存储数据

docker pull grafana/grafana

mkdir /opt/grafana-storage

* 设置权限

chmod 777 -R /opt/grafana-storage

* 启动grafana

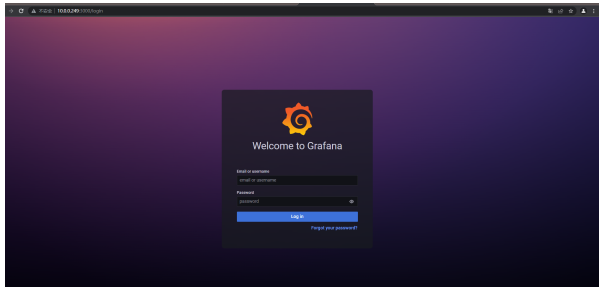
docker run -d \

-p 3000:3000 \

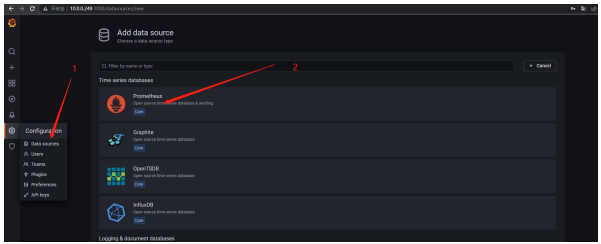
--name=grafana \

-v /opt/grafana-storage:/var/lib/grafana \ grafana/grafana

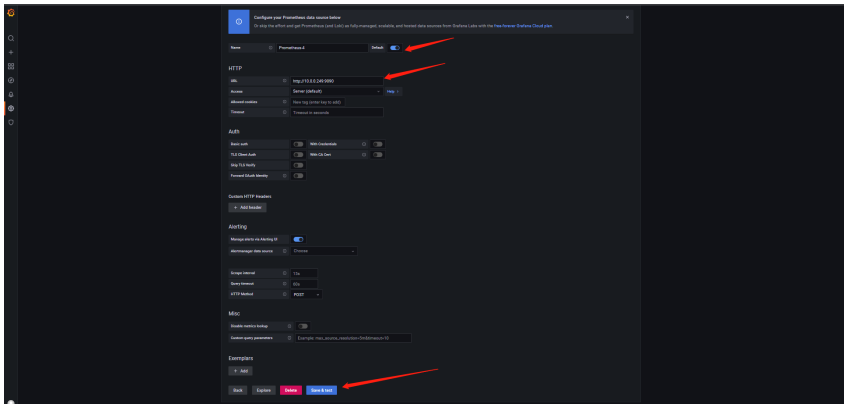
* 访问ip:3000，效果如下，账号密码都为admin，然后会提示你重新设置密码，密码设置完后，进入 首页



* 进入之后点击设置—data sources—Add data sources，添加prometheus数据源

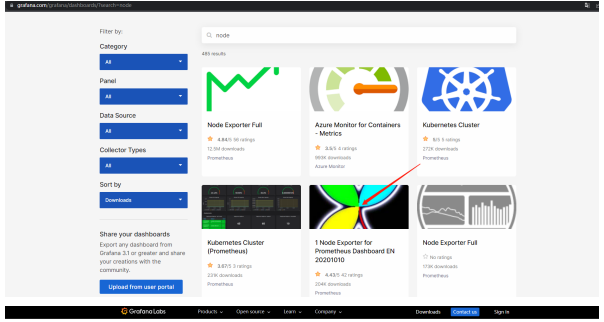


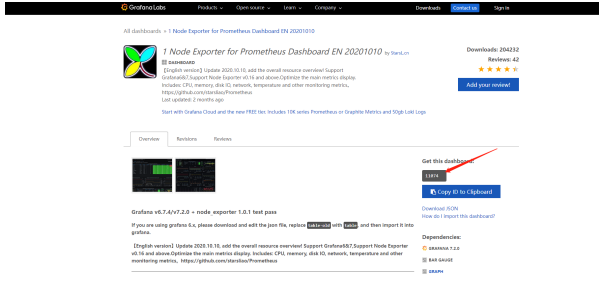
* 设置数据源地址，url 输入Prometheus的ip+端口，点击save，出现绿色对号表示设置成功



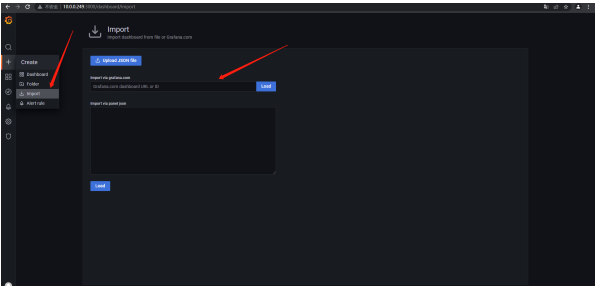
### 监控主机资源情况

* grafana官网有很多监控模板，https://grafana.com/grafana/dashboards，进入之后搜索node， 点击模板进去可以看到右侧有id号，记住id号，返回grafana

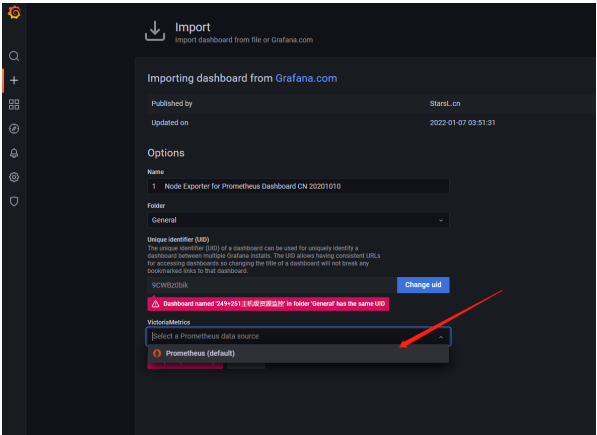




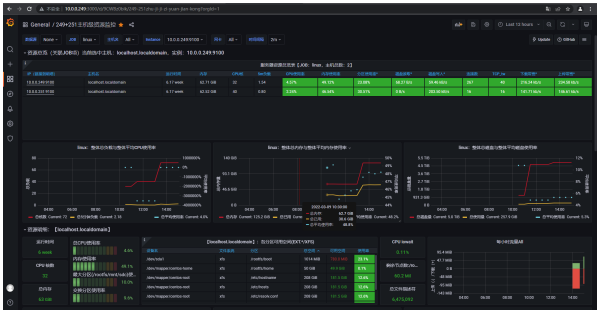
* 添加监控模板，点击左侧+号—Import—输入id，点击load （推荐8919，是中文版本）



* 选择数据源，点击import



* 最终效果显示



### 监控docker级别资源

监控docker需要安装谷歌开源的一款插件cadvisor

* 直接运行容器进行安装

docker run \

--volume=/:/rootfs:ro \

--volume=/var/run:/var/run:rw \

--volume=/sys:/sys:ro \

--volume=/var/lib/docker/:/var/lib/docker:ro \

--volume=/dev/disk/:/dev/disk:ro \

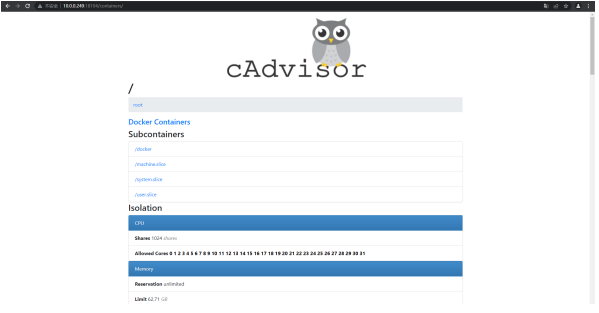
--publish=18104:8080\

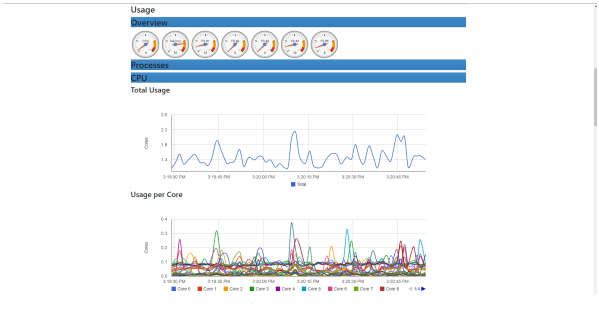
--detach=true \

--name=cadvisor \

google/cadvisor:latest

* 访问ip:18104，效果如下：





* 修改prometheus配置文件

vim /opt/prometheus.yml

global:

scrape\_interval: 60s

evaluation\_interval: 60s

scrape\_configs:

- job\_name: prometheus

static\_configs:

- targets: ['localhost:9090']

labels: instance: prometheus

- job\_name: linux

static\_configs:

- targets: ['10.0.0.249:9100','10.0.0.251:9100']

labels:

#添加下面内容

- job\_name: docker

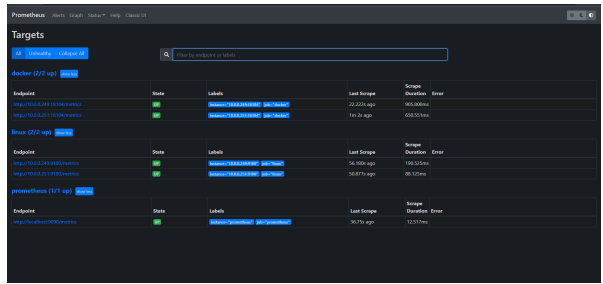
static\_configs:

- targets: ['10.0.0.249:18104','10.0.0.251:18104'] #调用cadvisor数据进行监 控，插件可安装在需要监控的主机，多 主机可配置多ip

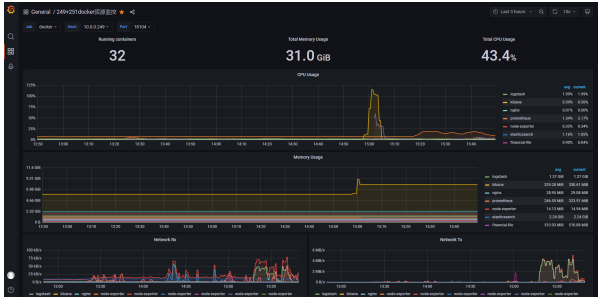
* 重启prometheus容器

docker restart prometheus

* 访问ip:9090查看是否已添加，可以看到docker栏中有两台主机



* 进入https://grafana.com/grafana/dashboards，搜索docker监控模板，建议使用8321，步骤和第 四章监控主机节点一样，最终效果展示：



## 步骤五：ELK应用日志系统部署

### ElasticSearch

（1）拉取镜像

docker pull docker.elastic.co/elasticsearch/elasticsearch:6.3.2

（2）安装

docker run -d --name es -p 6200:9200 -p 6300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2

（3）配置跨域

* 进入容器

docker exec -it es /bin/bash

* 2）修改elasticsearch.yml

vi /usr/share/elasticsearch/config/elasticsearch.yml

# 删除原有配置，添加跨域配置

http.host: 0.0.0.0

http.port: 9200

http.cors.enabled: true

http.cors.allow-origin: "\*"

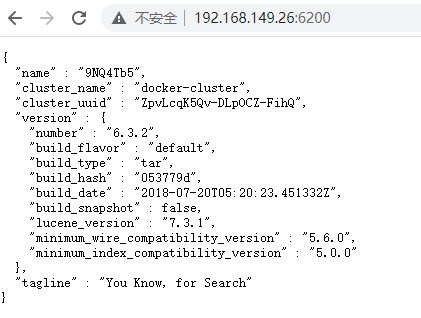
（4）重启容器

docker restart es

（5）访问地址

[http://192.168.149.26:6200](http://localhost:9200/)  填写自己ip

显示下图内容表示成功



### ElasticSearch-Head（ES数据读取插件）

（1）拉取

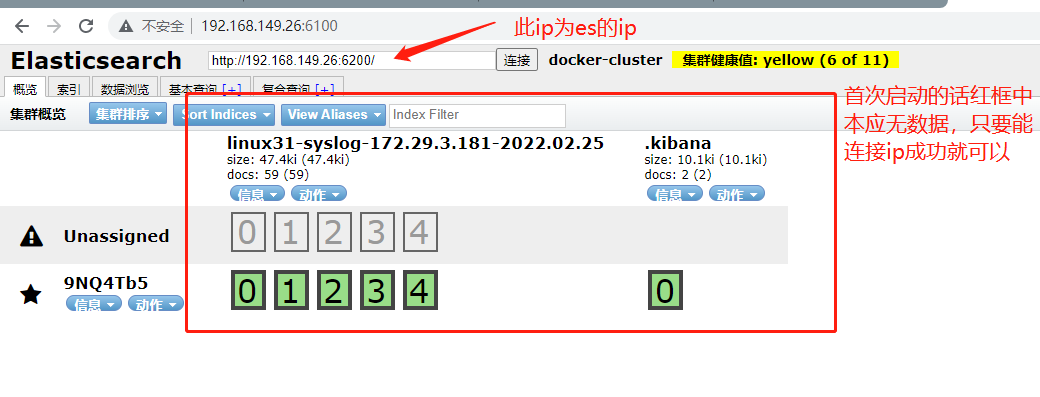
docker pull mobz/elasticsearch-head:5

（2）运行容器

docker run -d --name es\_admin -p 6100:9100 mobz/elasticsearch-head:5

（3）访问地址

[http://192.168.149.26:6100](http://192.168.149.26:6100%09)  填写自己ip



### logstash

（1）拉取镜像

docker pull docker.elastic.co/logstash/logstash:6.2.4

（2）运行

docker run -d --name es\_logstash docker.elastic.co/logstash/logstash:6.2.4

（3）进入容器

docker exec -it -u root es\_logstash /bin/bash

（4）修改logstash.yml文件

vi /usr/share/logstash/config/logstash.yml

http.host: "0.0.0.0"

xpack.monitoring.elasticsearch.url: http://192.168.149.26:6200 #url ip为本机ip，端口为elasticsearch端口

（5）重启logstash

docker restart es\_logstash

### kibana

（1）拉取镜像

docker pull docker.elastic.co/kibana/kibana:6.3.2 注意：跟ES的版本保持一致

（2）运行

docker run --name es\_kibana -p 5601:5601 -d -e ELASTICSEARCH\_URL=http://192.168.149.26:6200 docker.elastic.co/kibana/kibana:6.3.2

（3）访问地址

http://192.168.149.26:5601 填写自己ip

（4）进入容器

docker exec -it es\_logstash /bin/bash

（5）修改pipeline下的logstash.conf文件

vi /usr/share/logstash/pipeline/logstash.conf

注释掉所有原本内容，添加以下参数

input {

file {

path => "/var/log/yum.log" #收集此目录下的yum.log日志

start\_position => "beginning"

stat\_interval => "3"

type => "yumlog"

}

}

output {

if [type] == "yumlog" {

elasticsearch {

hosts => ["192.168.149.26:6200"] #本机的es地址

index => "yum-%{+YYYY.MM.dd}" #展示在kibana中的索引

}}

}

（6）授权/var/log/yum.log可读可执行权限（如不授权ELK将读取不了此文件）

chmod 755 /var/log/yum.log

重启全部容器

docker restart es\_logstash

docker restart es\_admin

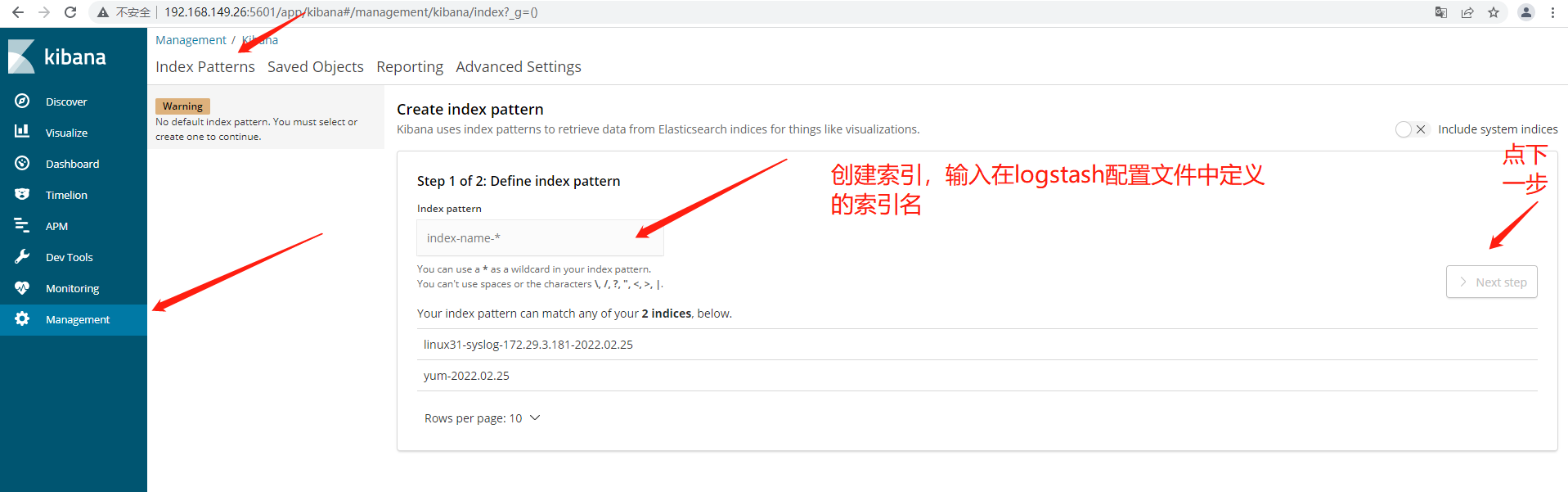
docker restart es

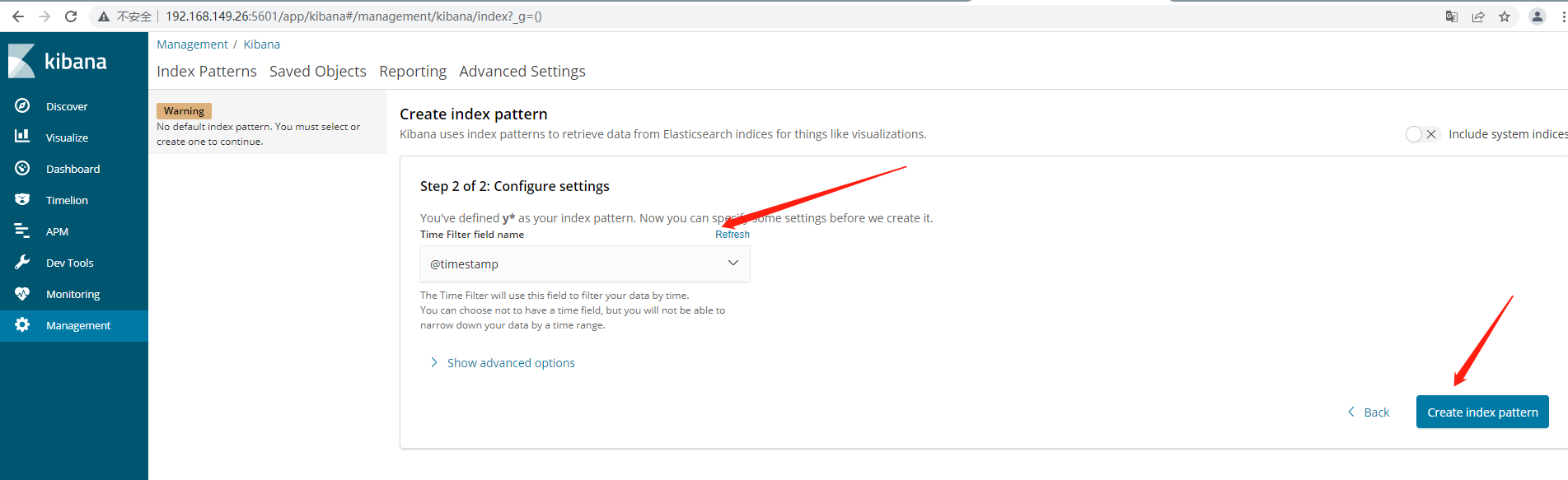
docker restart es\_kibana

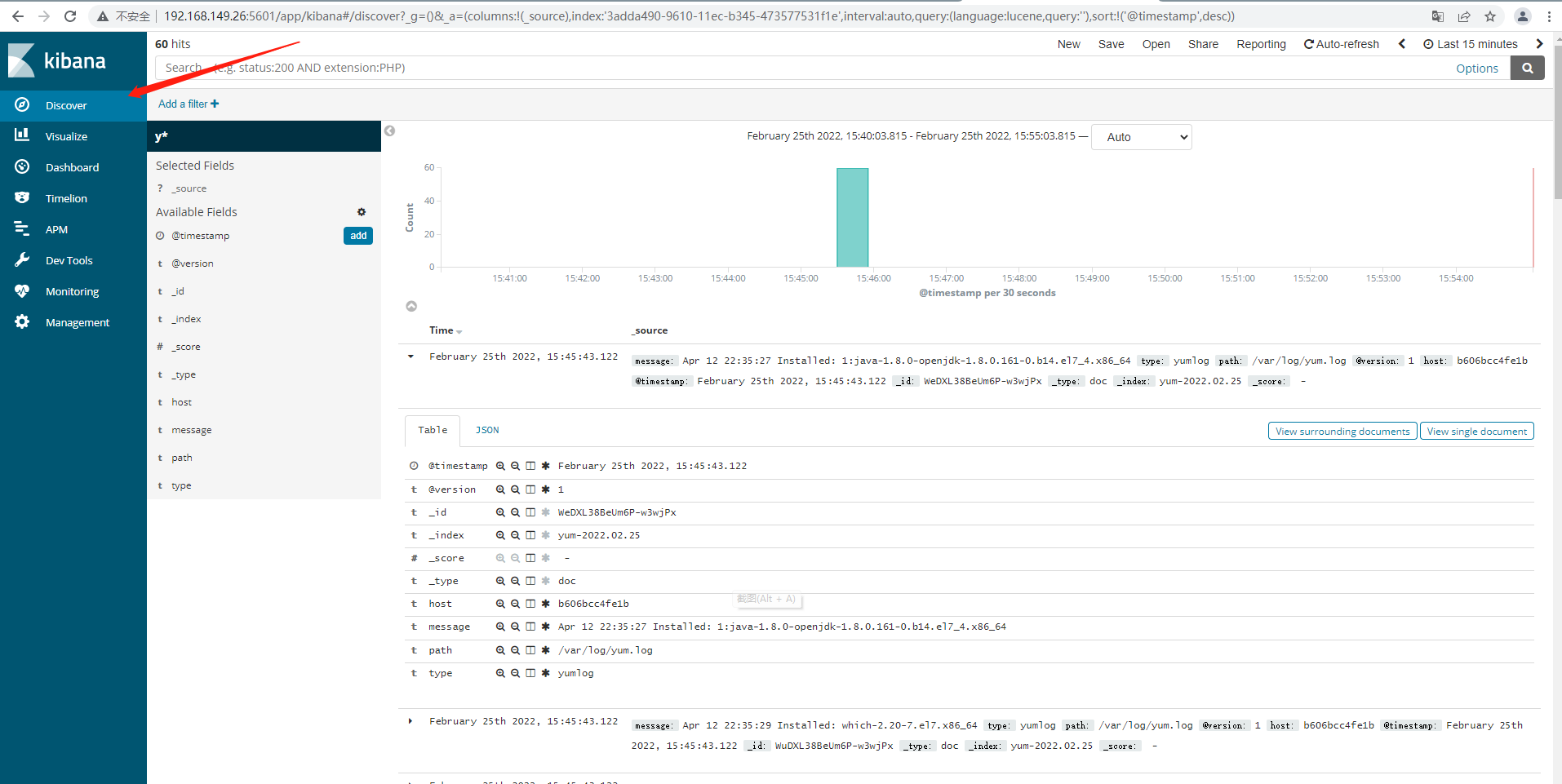
访问kibana地址：

http://192.168.149.26:5601/

（7）创建索引







创建完成会有/var/log/yum.log日志数据显示，表示搭建成功

## 步骤六：视频转码服务部署

### 转码所需工具

ffmpeg（音视频转换工具）

ZLMedia（流媒体服务框架）

WVP（28181协议视频平台）

Redis

Jessibuca (web播放器)

### 安装ffmpeg

（1）下载解压

wget http://www.ffmpeg.org/releases/ffmpeg-4.1.tar.gz

tar -zxvf ffmpeg-4.1.tar.gz

（2）进入解压后目录,输入如下命令/usr/local/ffmpeg为自己指定的安装目录

cd ffmpeg-4.1

./configure --prefix=/usr/local/ffmpeg

make && make install

（3）配置变量

vi /etc/profile

# 在最后PATH添加环境变量：

export PATH=$PATH:/usr/local/ffmpeg/bin

# 保存退出

# 查看是否生效

source /ect/profile 设置生效

### ZLM（centos7）

（1）编译机器： centos 7.6

[gcc](https://so.csdn.net/so/search?q=gcc&spm=1001.2101.3001.7020" \t "_blank) version 5.4.0 (GCC)

cmake version 3.20.5

（2）安装gcc(要求版本>=4.8)

sudo yum install gcc

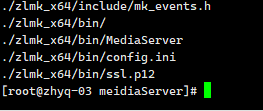
sudo yum install gcc-c++

（3）安装cmake

sudo yum -y install cmake

（4）解压文件

tar -zxvf zlmk\_2020.09.17\_x68\_64.tar.gz



（5）配置ZLM

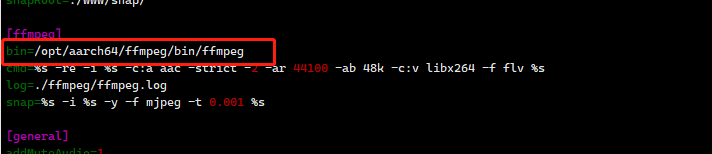
进入文件夹

cd zlmk\_x64/bin

修改配置文件

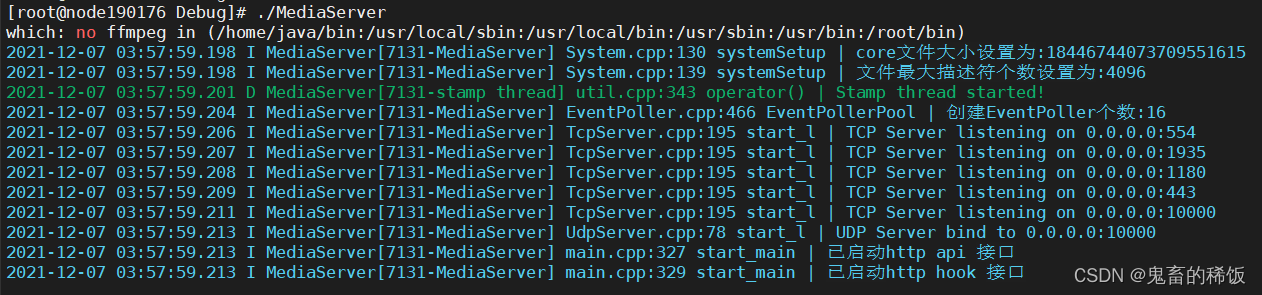
vim config.ini

修改ffmpeg.bin为安装的ffmpeg位置，保存，退出。



（6）运行ZLM

#通过-h可以了解启动参数./MediaServer -h#以守护进程模式启动./MediaServer -d &



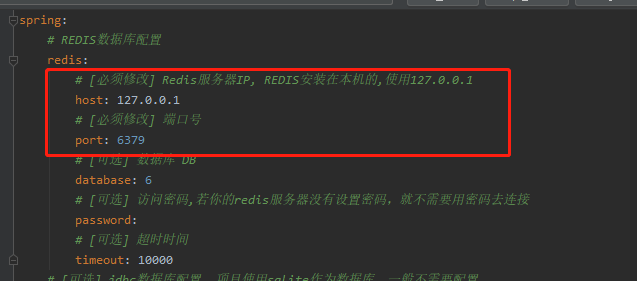
（7）WVP转码服务安装

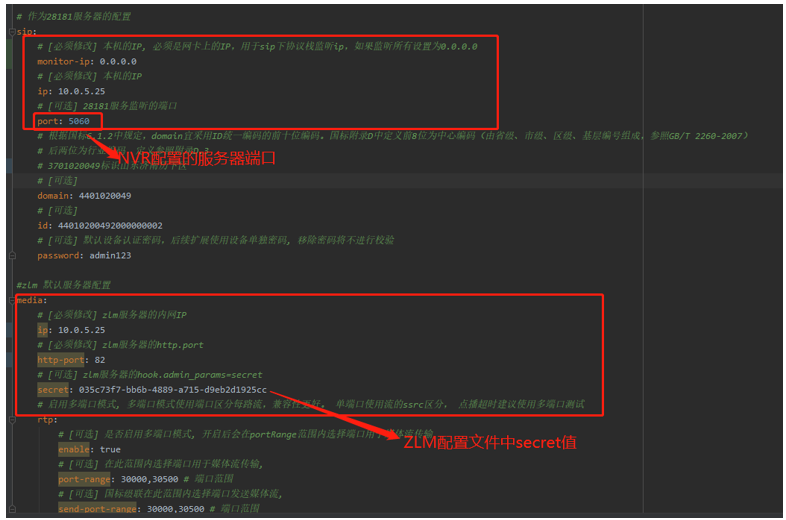
<https://github.com/648540858/wvp-GB28181-pro>

* Wvp配置文件修改

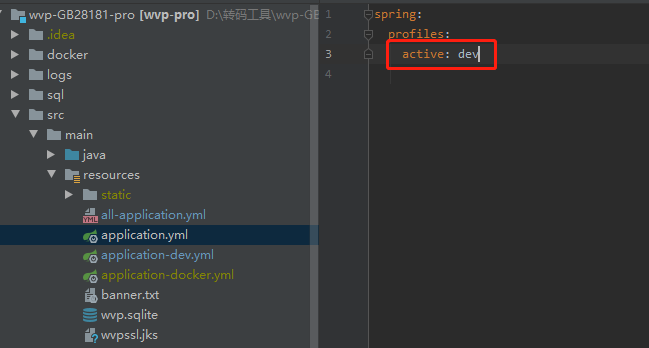
修改application-dev.yml配置文件

Redis ip、port，sip.ip，sip.monitor-ip，sip.port， Media ip、port、stream-ip，配置完成后启动wvp

部署公网环境sip.ip、media.stream-ip、media.sdp-ip需填写为公网信息

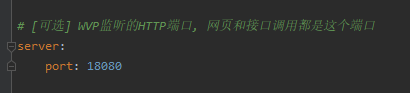


将application.yml中active参数改为dev



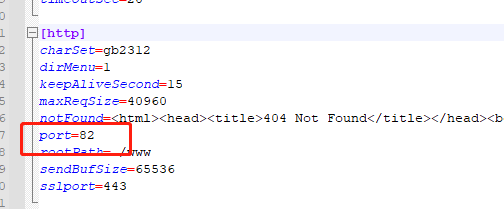
* 部署公网开放端口

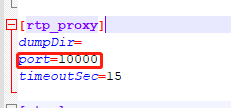
wvp转码服务端口：sip.port，server.port





ZLM端口（ZLM配置文件中配置的端口）：http.port, rtp\_proxy.port

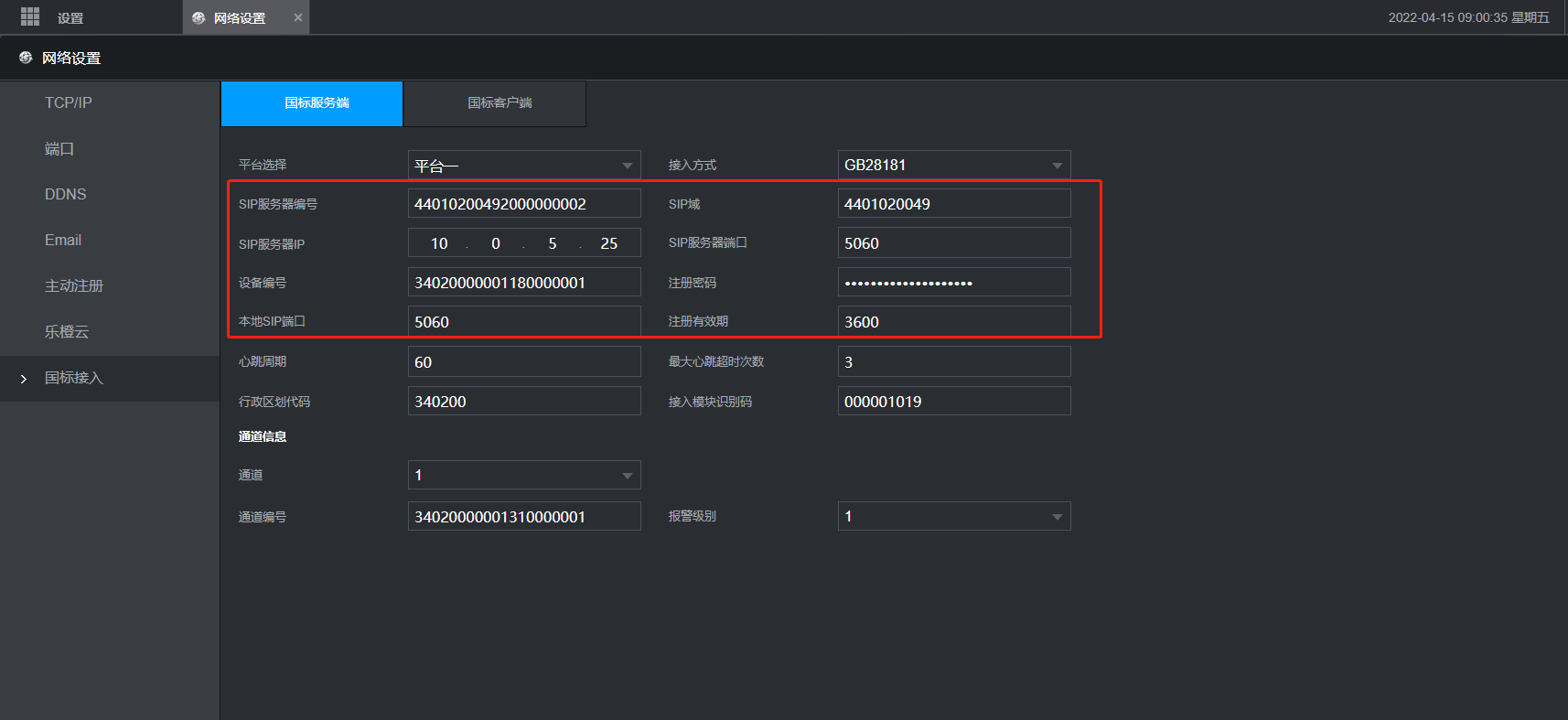




* 配置NVR

进入NVR后台，选择设置-网络设置-国标接入。按照wvp转码服务配置文件修改



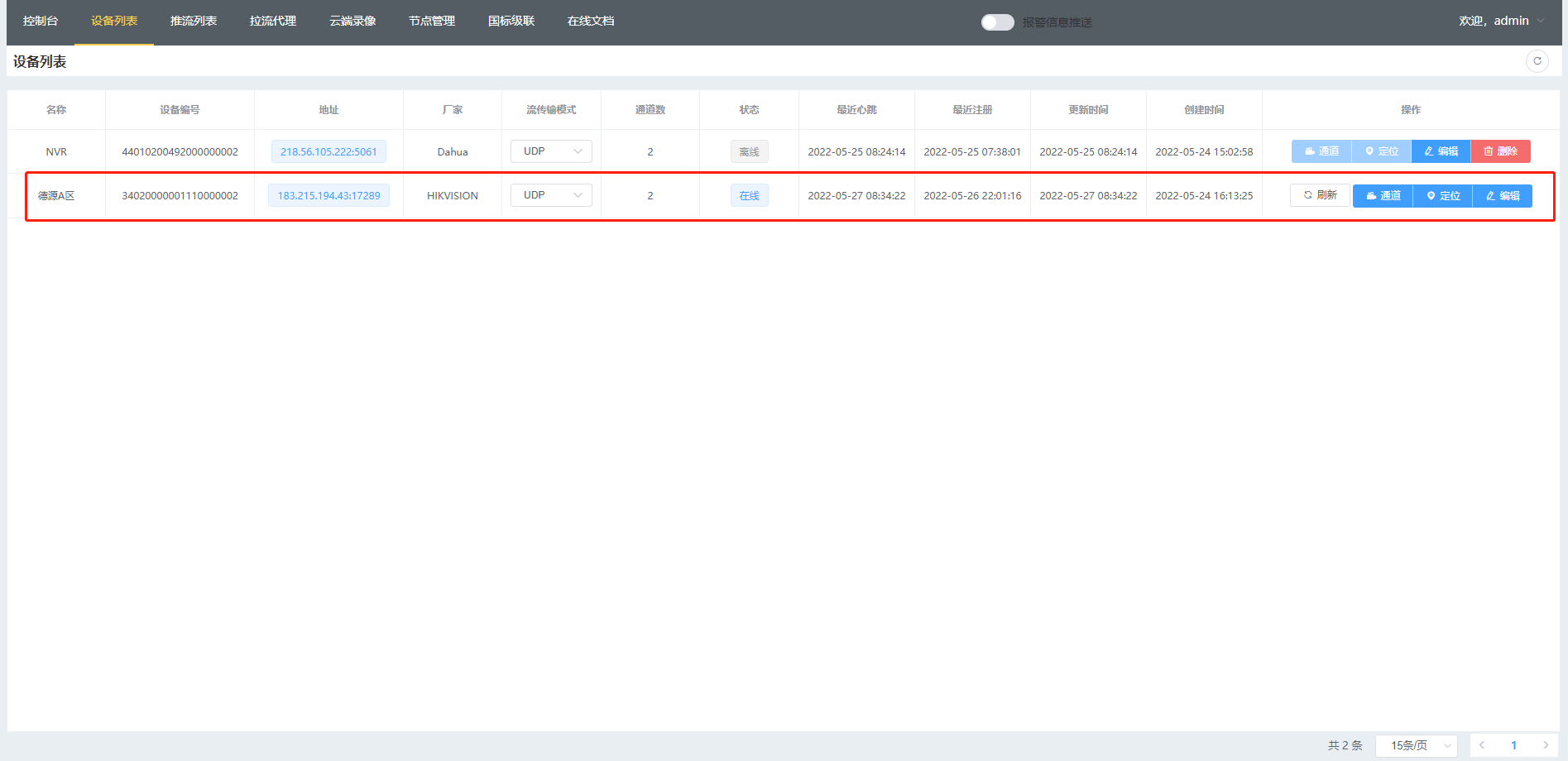


* 登录地址

http://localhost:8080/login#/login

admin admin123

NVR设备注册成功后，设备列表会有设备显示，或调用查询设备接口查看。



海康设备手动添加设备通道编号：

