前端学习

AVUE-DATA

数据大屏解决方案

一个很多骚操作的前端框架

目　录

项目介绍

数据大屏解决方案

介绍

Avue-Data数据大屏(Vue全家桶+ElementUi+Echart+dataV)开发，丰富的交互控件和图表组件，提供智能图形推荐，报表图形任意切换，且不受维度,度量的限制。

特点

- 缩短开发周期

  提供丰富的二次开发接口
- 快速响应

  支持动态局部刷新，秒级响应
- 多端部署

  适配各种拼接大屏场景
- 实时数据

  支持多种数据源接入数据

功能清单

点击查看详情

丰富模板

政府行业

> BI大数据可以打破信息壁垒，形成统一数据共享大平台，打破“信息孤岛”问题。

教育行业

BI大数据可以有效查看各基地的情况，通过对教、学、研多层面的数据整合，制定更行之有效的方案。

金融行业

BI大屏拥有收集、运用数据的能力，从而做出快速精准的应对策略。

销售行业

BI大屏可以通过信息技术将各个渠道的数据进行整合，更好的对客户信息进行挖掘、创造价值。

交通行业

BI大屏在整个交通运输系统中的应用能有效解决城市化加速而导致的各种问题，在大客流运营常态下有着重要的价值。


智能交通综合执法平台-执法管理
桌面
冯亮神（财务总监）
退出
首页
案件办理
联合惩戒
执法监督
执法系统分析
社会公示和服务
待办事项
预警审核 (80条)
立案登记 (22条)
登记审核 (6条)
处理意见 (4条)
案件移送 (35条)
违法通知 (19条)
处罚决定 (0条)
结案归档 (12条)
预警数 1425件
立案数 5305件
结案数 3300件
出具文书合计 2560件
违法行为类型统计
单位（件）
本月
上个月
超限超载
抽烟
不按路线行驶
违停
违规上下客
打瞌睡
疲劳驾驶
行政处罚程序统计
30%
45%
25%
简易程序 4860件
一般程序 1370件
听证程序 2450件
近3个月违法车辆类型统计
违法最多车辆类型：公交车
违法最少车辆类型：网约车
货车
客车
公交车
出租车
网约车
0 100 200 300 400 500 600 700 800 次数
时间：2019.04.10-2019.07.10
执法单位信息
立警为公 执法为民
单位名称：深圳市南山区粤海街道交通管理局
所属管辖范围：南山区粤海街道
机构：交通管理执法机构
科室：交通执法科室
人数：23人
通知公告
1、严格纪律作风，强制提升交通行政执法队伍形象；
2、加强执法队伍班子建设，不断增强班子的凝聚力、向心力和战斗力；
3、加强基层基础建设，夯实交通执法工作根基；
4、强化培训兵，提升执法人员业务素质。
案件办理结果统计
单位（件）
行政处罚数
责令改正数
不予处罚数
案件移送数
强制执行数
8月 9月 10月 11月 12月 1月 2月 3月 4月 5月 6月 7月
今年与去年环比分析
+35%
+14%
-12%
违法数
立案数
结案数

常见例子库

> 常见的一些例子和操作

- 数据赋值
- 组件交互
- 组件事件交互
- 动态Echart配置切换
- 任意Echart配置
- 定时任务交互
- 自定义组件
- 自定义组件交互
- 自定义高德地图
- 自定义组件综合例子
- 3D视角

项目启动

项目启动

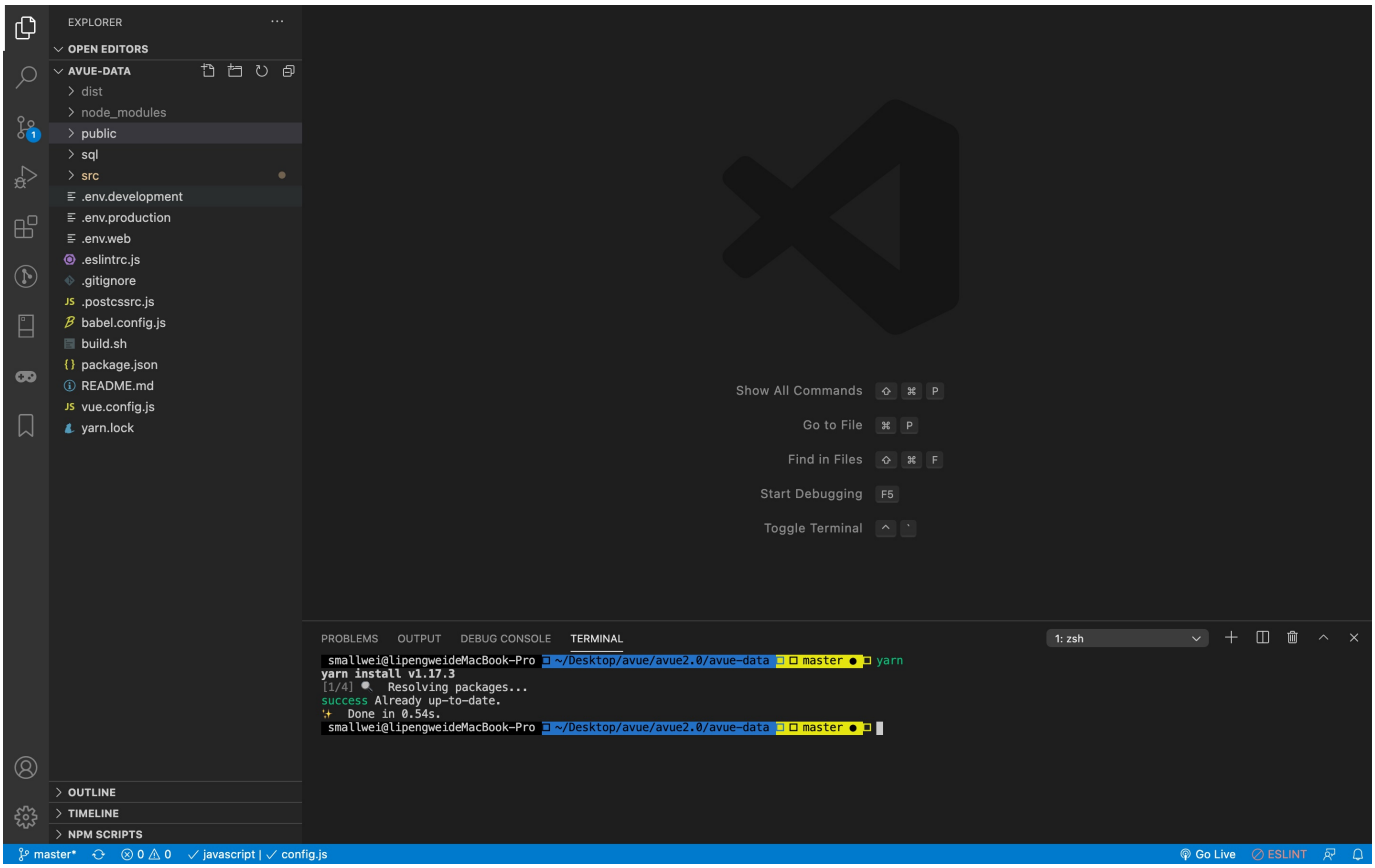
> 需要启动前端和后端2个端才可以正常运行

前端启动

一、工程导入

1. 使用Avue商业账号登录git私服：https://git.avuejs.com/avue/avue-data

2. 导入工程,在终端执行yarn install或者npm install

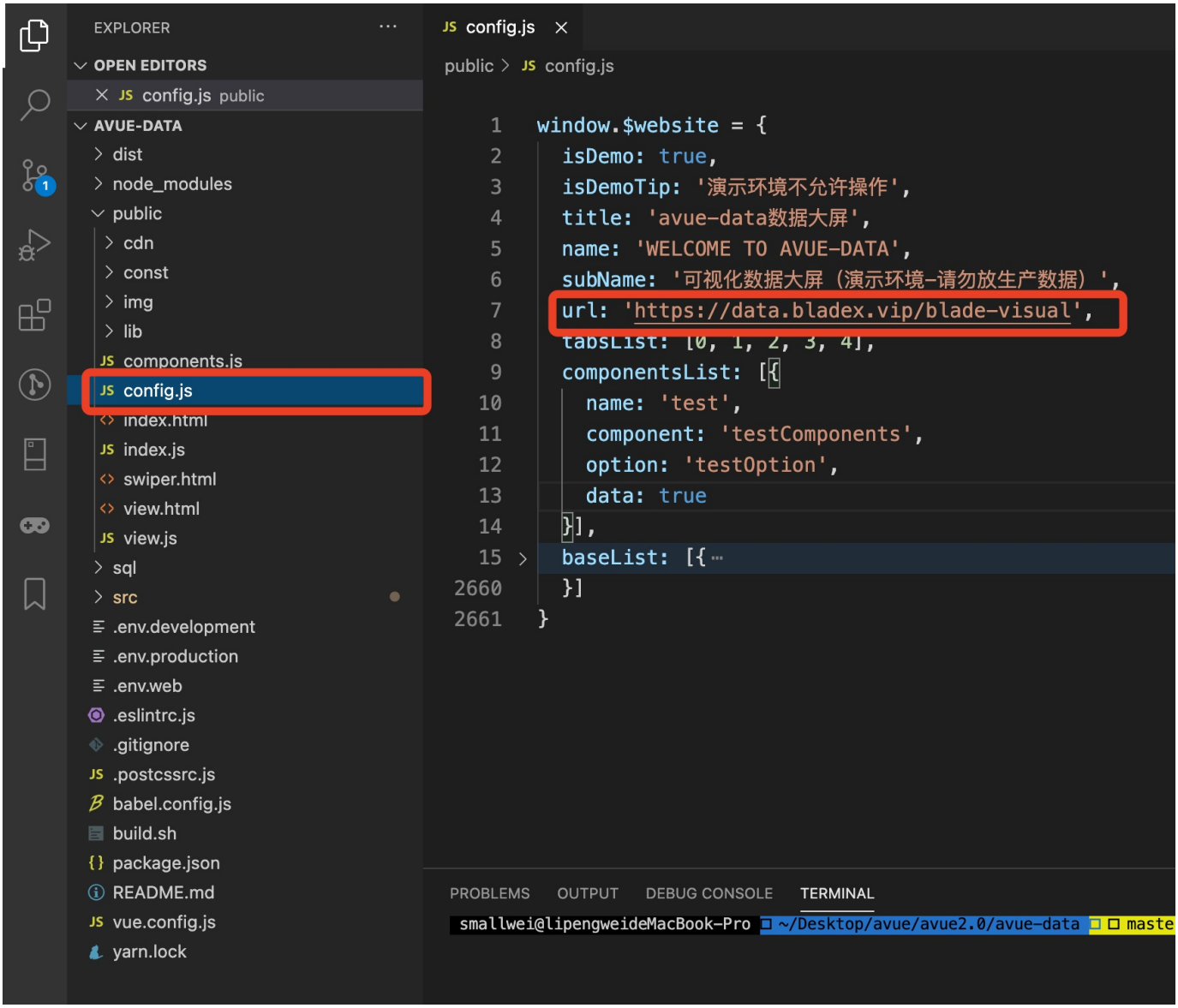


二、配置修改

1.这里要运行大屏后端

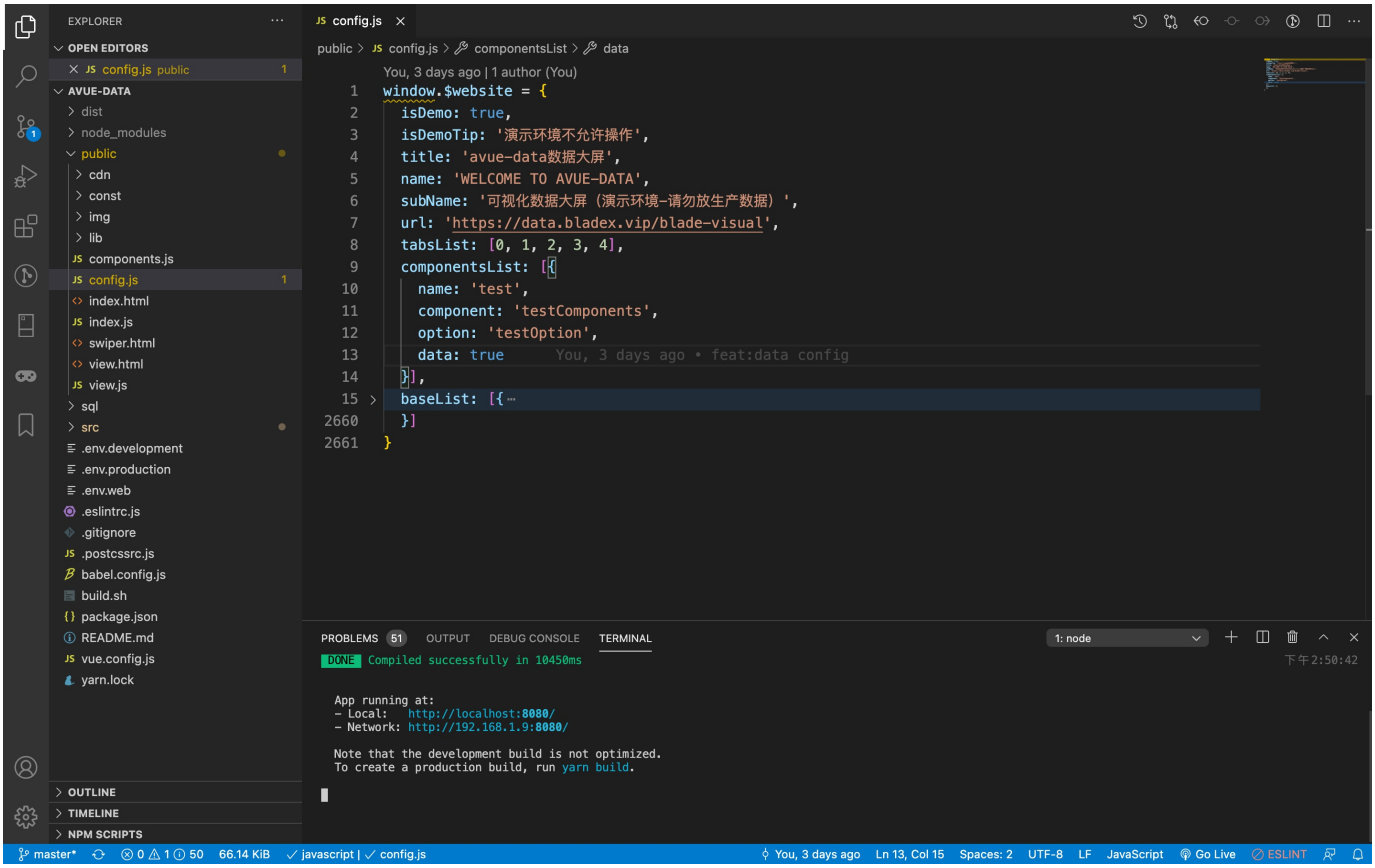
- JAVA版后端启动
- NODE版后端启动

2. 修改你自己的后端地址，默认是演示线上接口



三、工程启动

1. 命令行执行 `yarn run serve` ，看到如下日志则说明启动成功

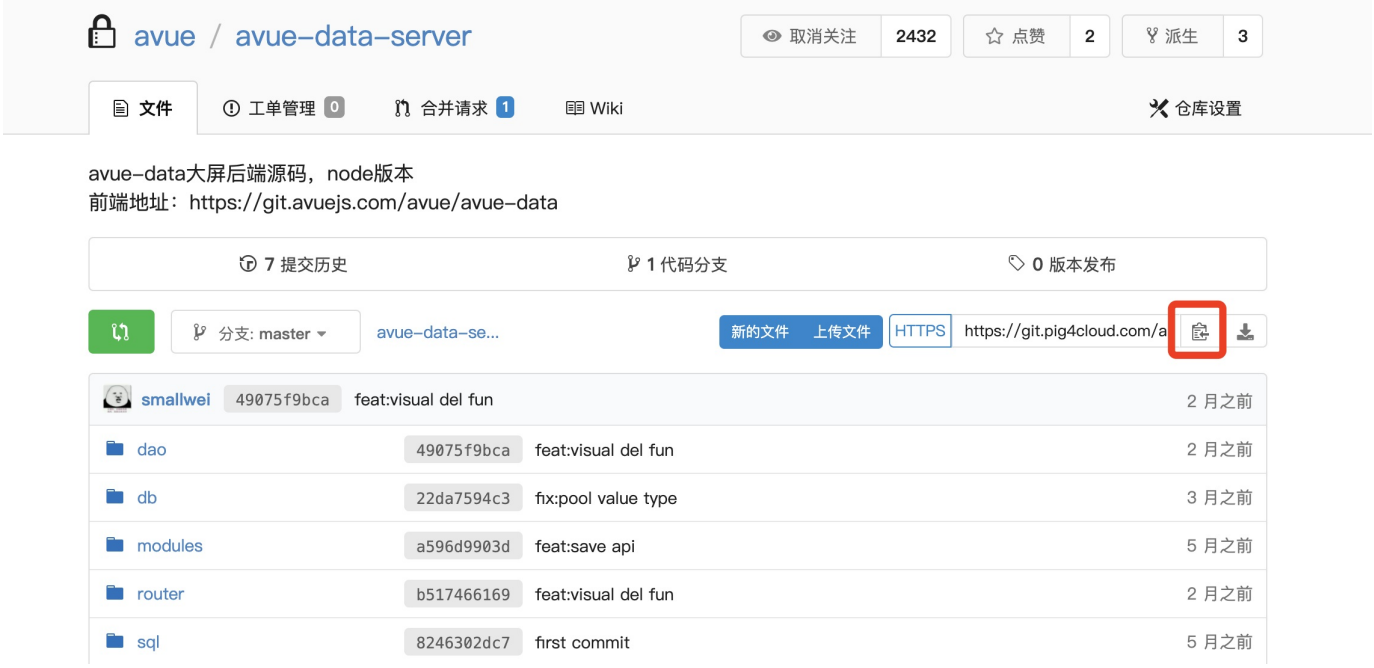


2. 访问 http://localhost:8080 查看效果

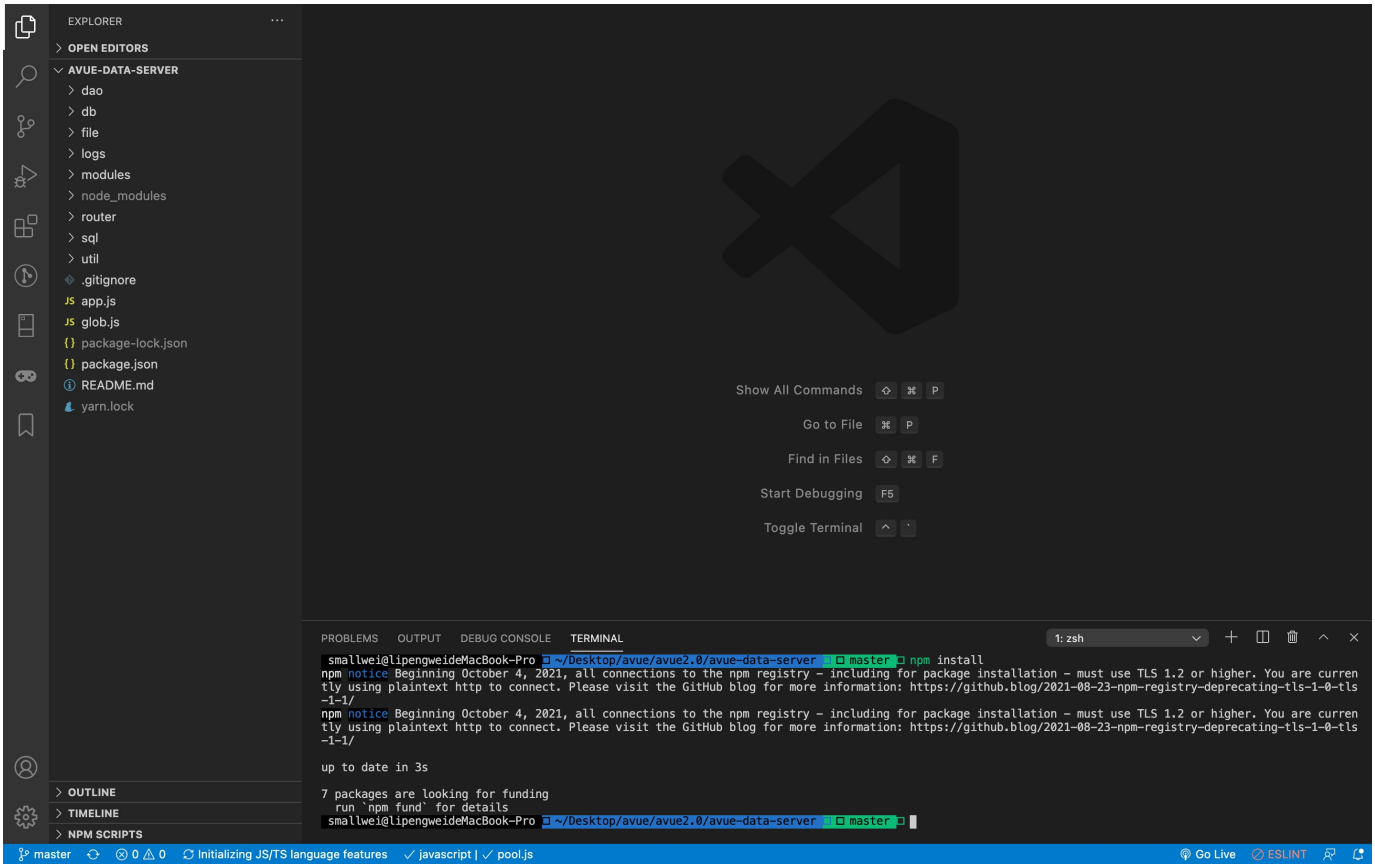
Node版后端启动

一、工程导入

1. 使用Avue商业账号登录git私服：https://git.avuejs.com/avue/avue-data-server

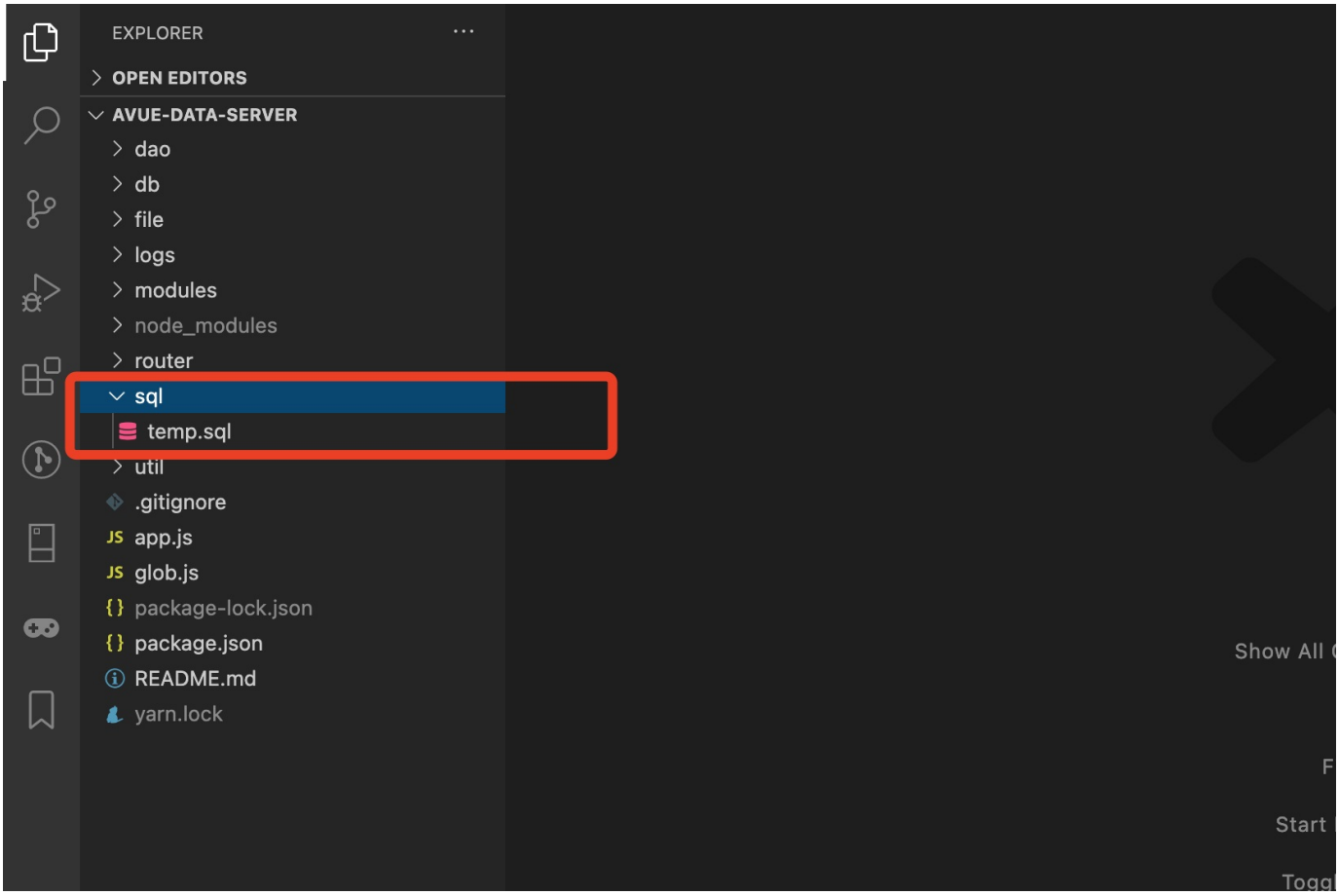


2. 导入工程,在终端执行yarn install或者npm install



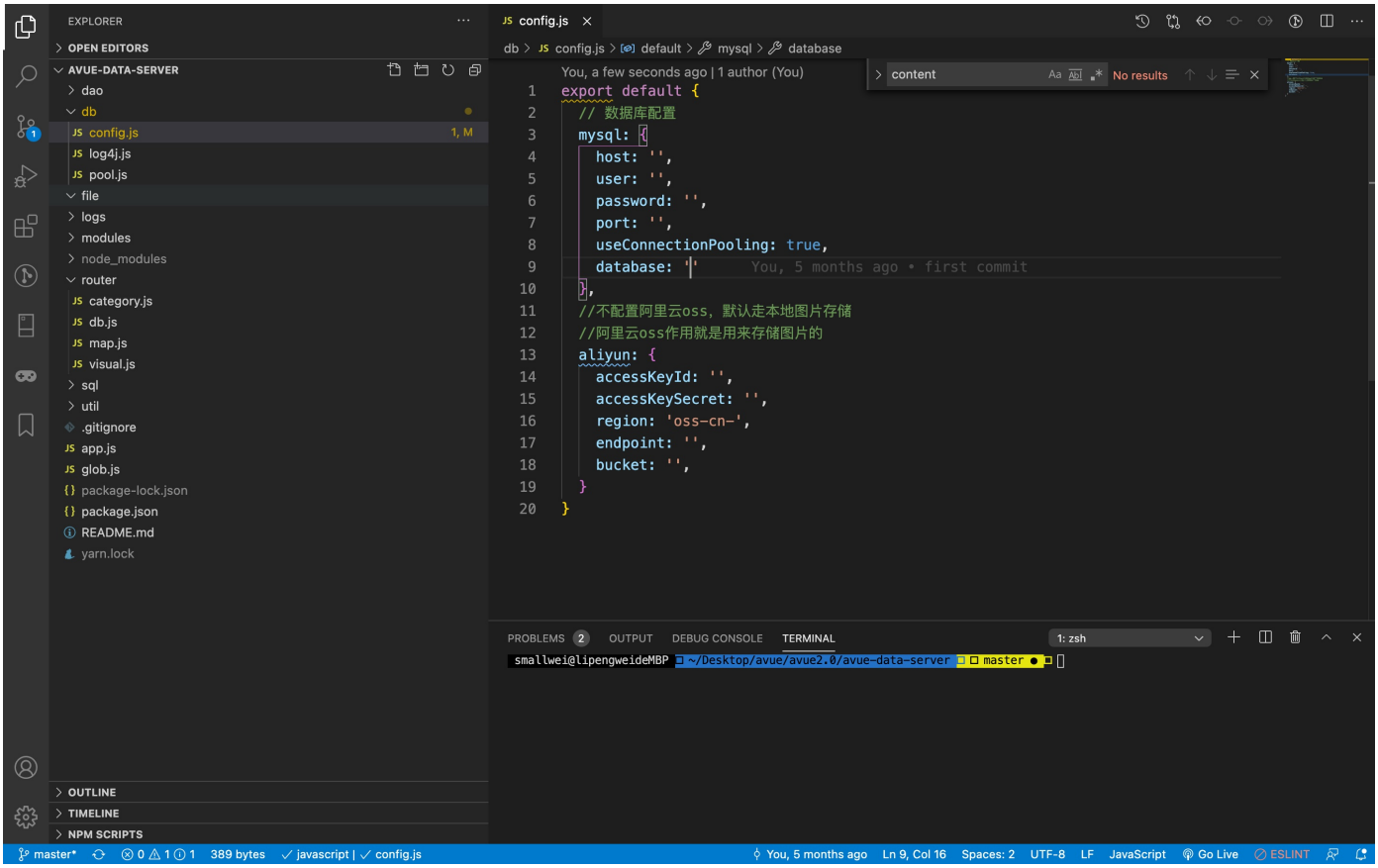
二、配置修改

1. 导入Sql数据文件



2. 修改数据库的连接地址和阿里云oss(阿里云oss作用就是用来存储图片)

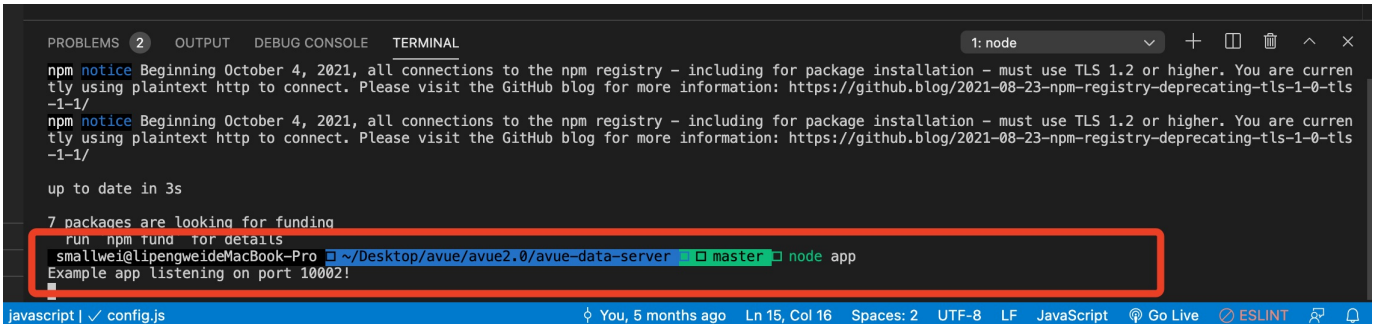
> 说明:不配置阿里云oss，默认走本地图片存储



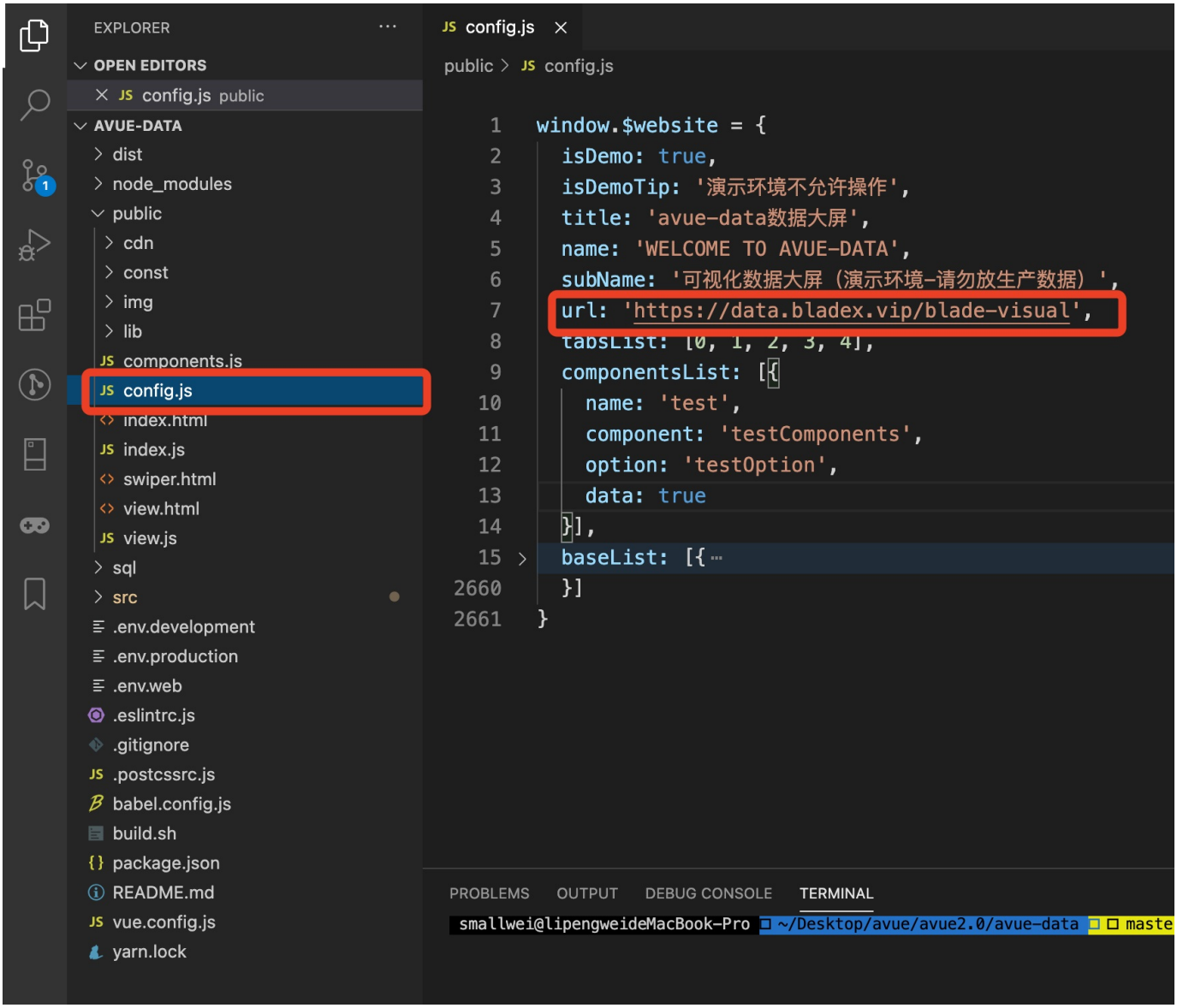
> !

三、工程启动

1. 命令行执行 `node app` ，看到如下日志则说明启动成功



2. 前端项目中修改成改地址这里是修改前端项目



3. 访问前端http://localhost:8080 查看效果

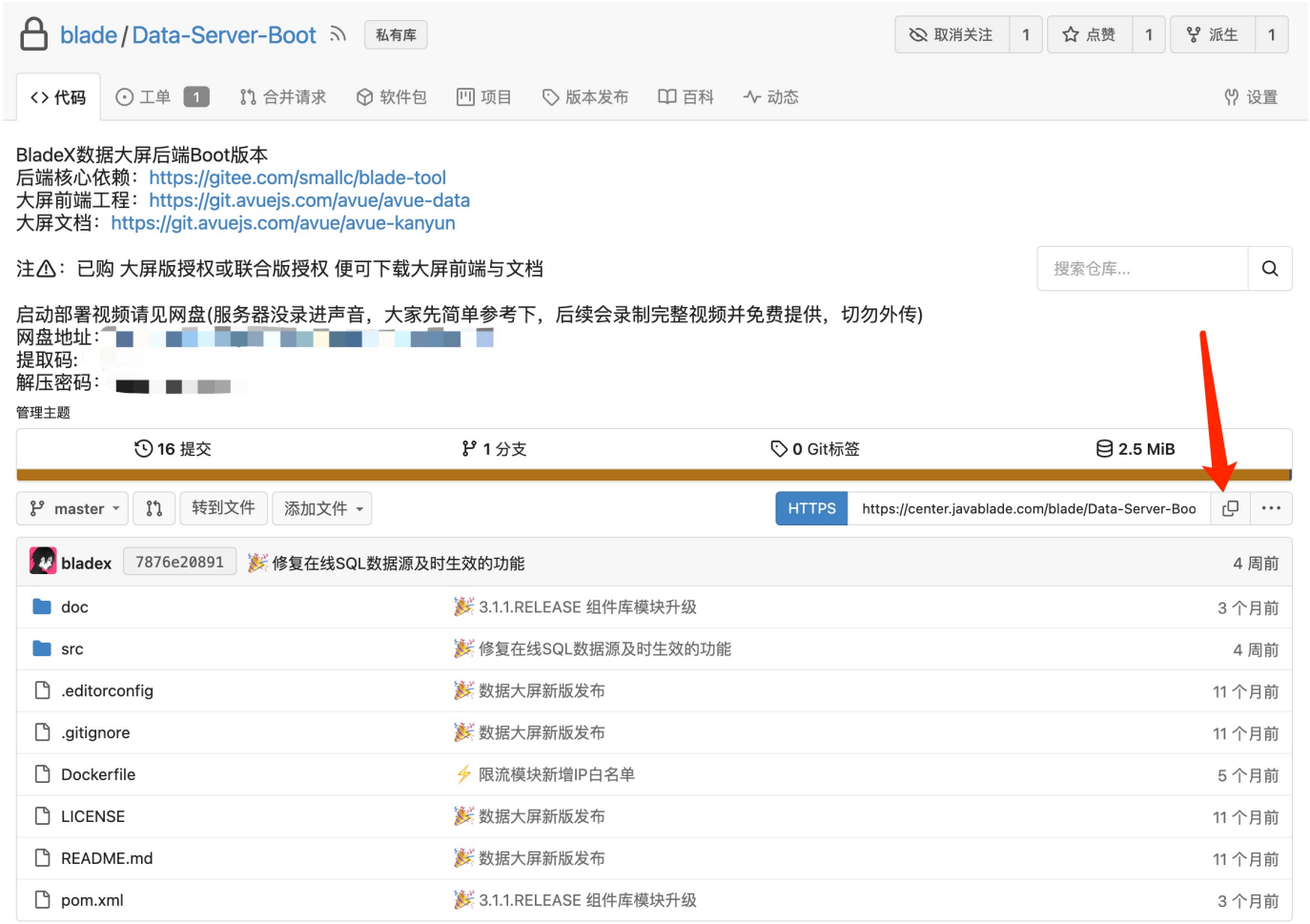

WELCOME TO AVUE-DATA
可视化数据大屏（演示环境-请勿放生产数据）
大屏管理
地图管理
分类管理
数据源管理
综合功能
选择下面的方式进行创建
精选模板
组件库
测试4
测试6
wasu
新建大屏
共 4994 条
50条/页
1
2
3
4
5
6
100
前往
1
页
数据大屏模板
已发布

Java版后端启动

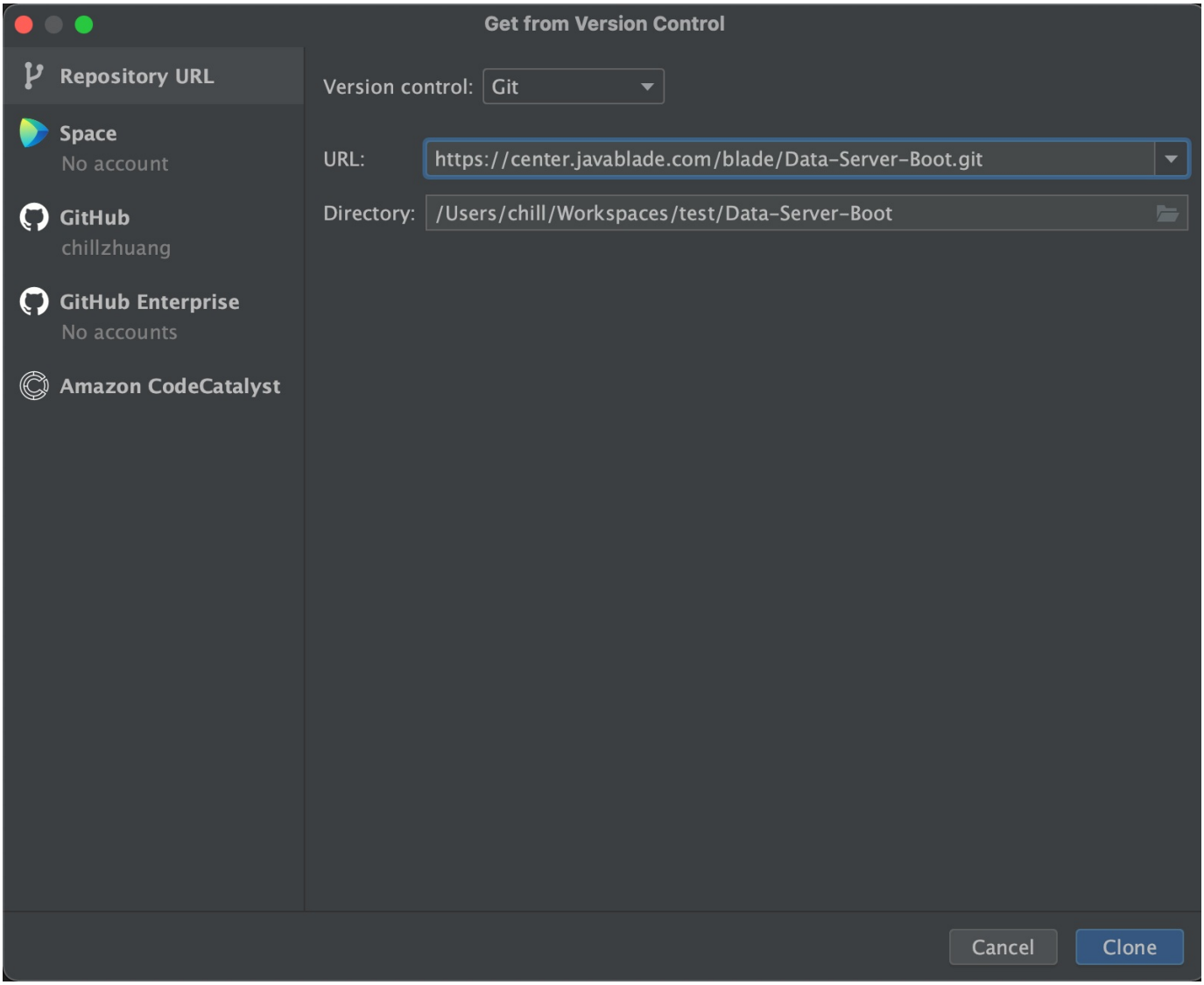
Boot版后端启动

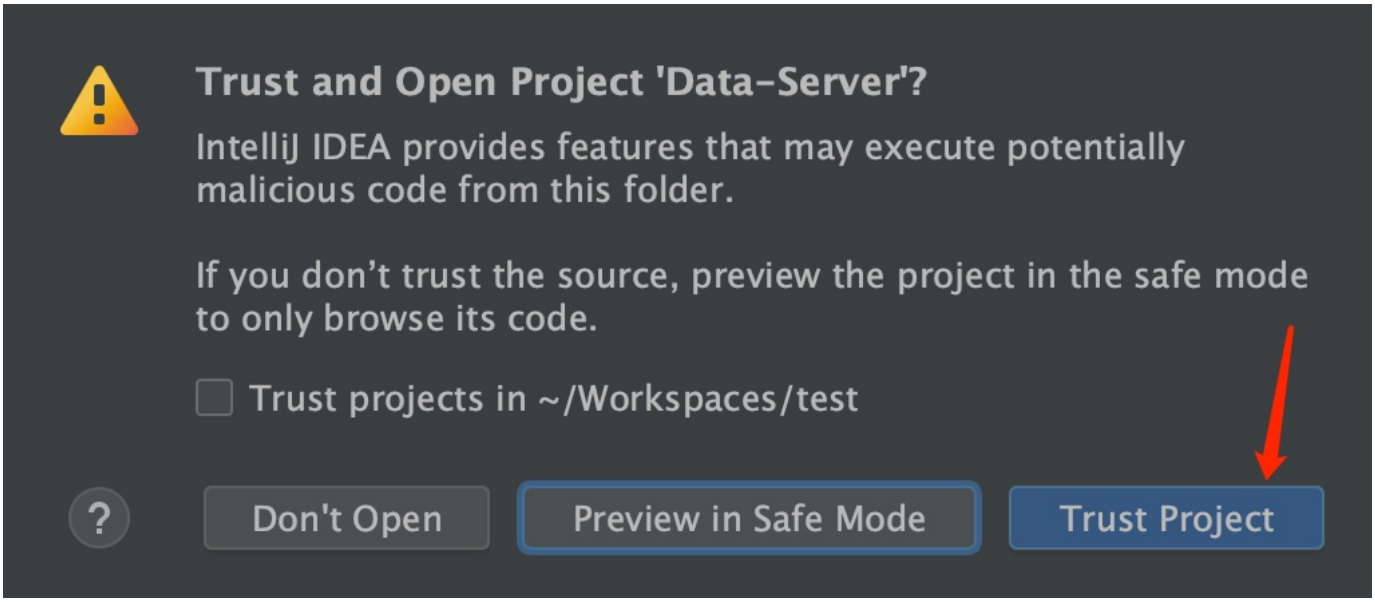
一、工程导入

1. 使用BladeX商业账号登录git私服：https://center.javablade.com/blade/Data-Server
2. 复制地址

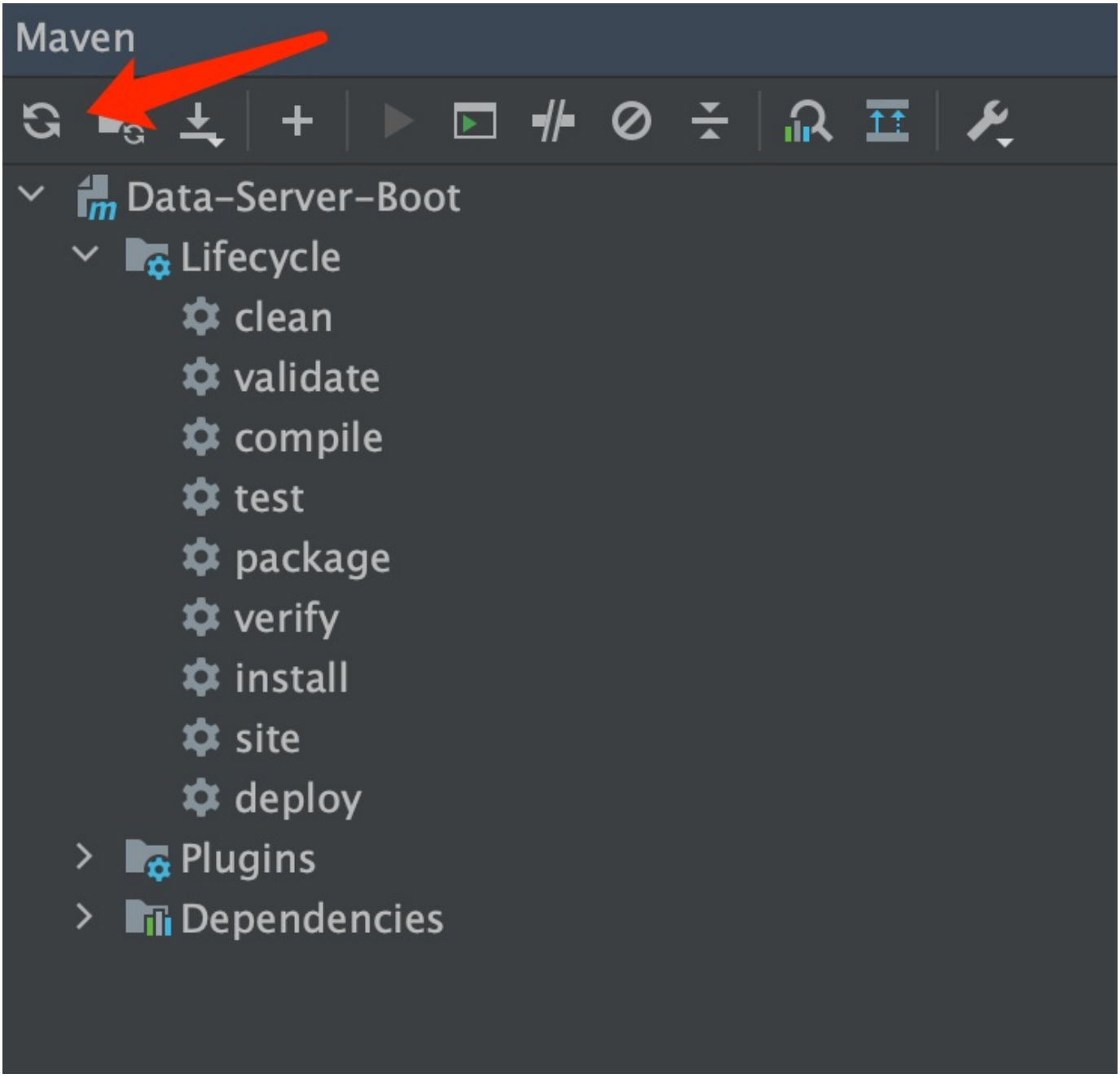


3. 导入工程

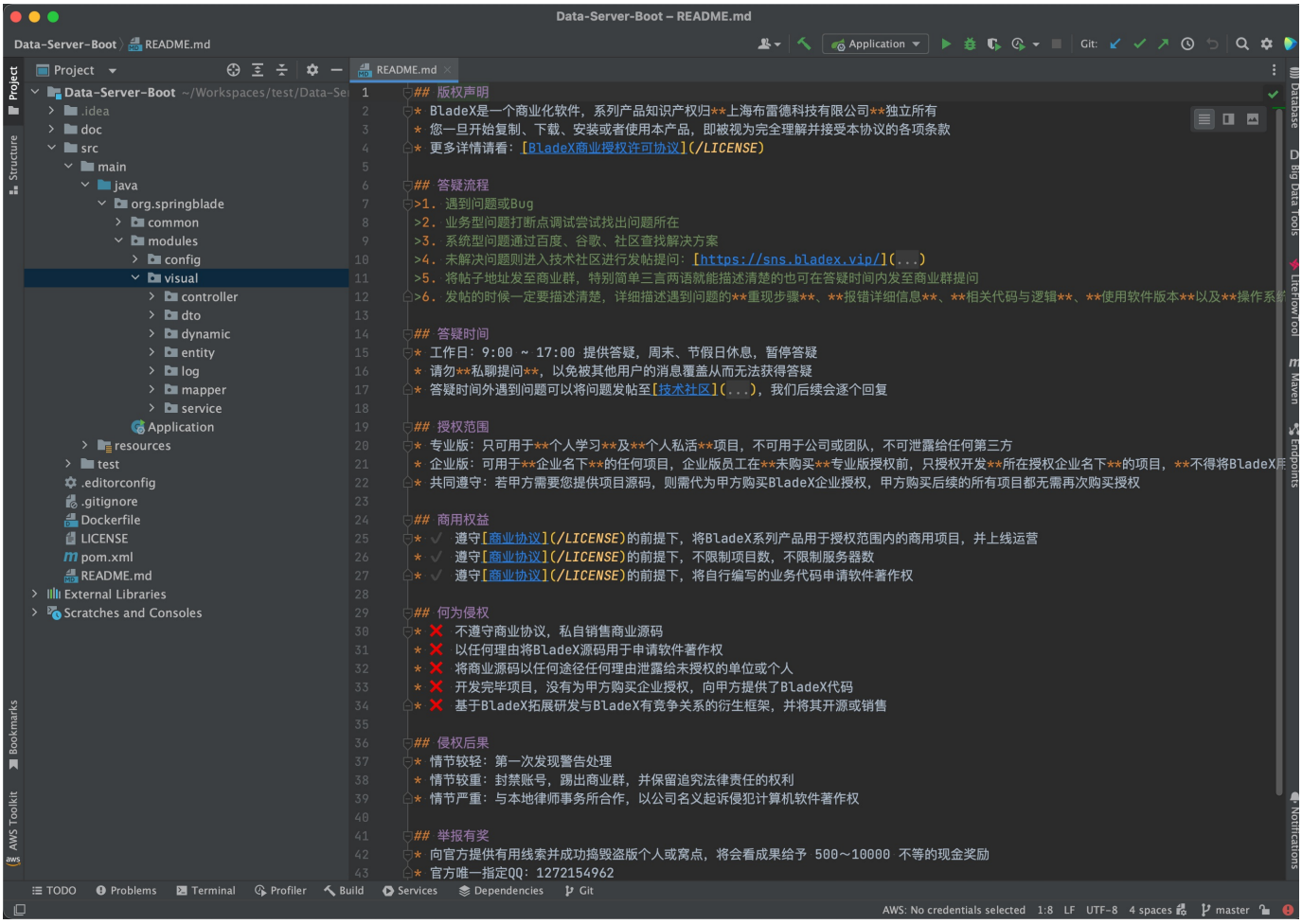




4. 导入成功后等待maven依赖加载完毕，如果依赖下载失败请多点击刷新按钮

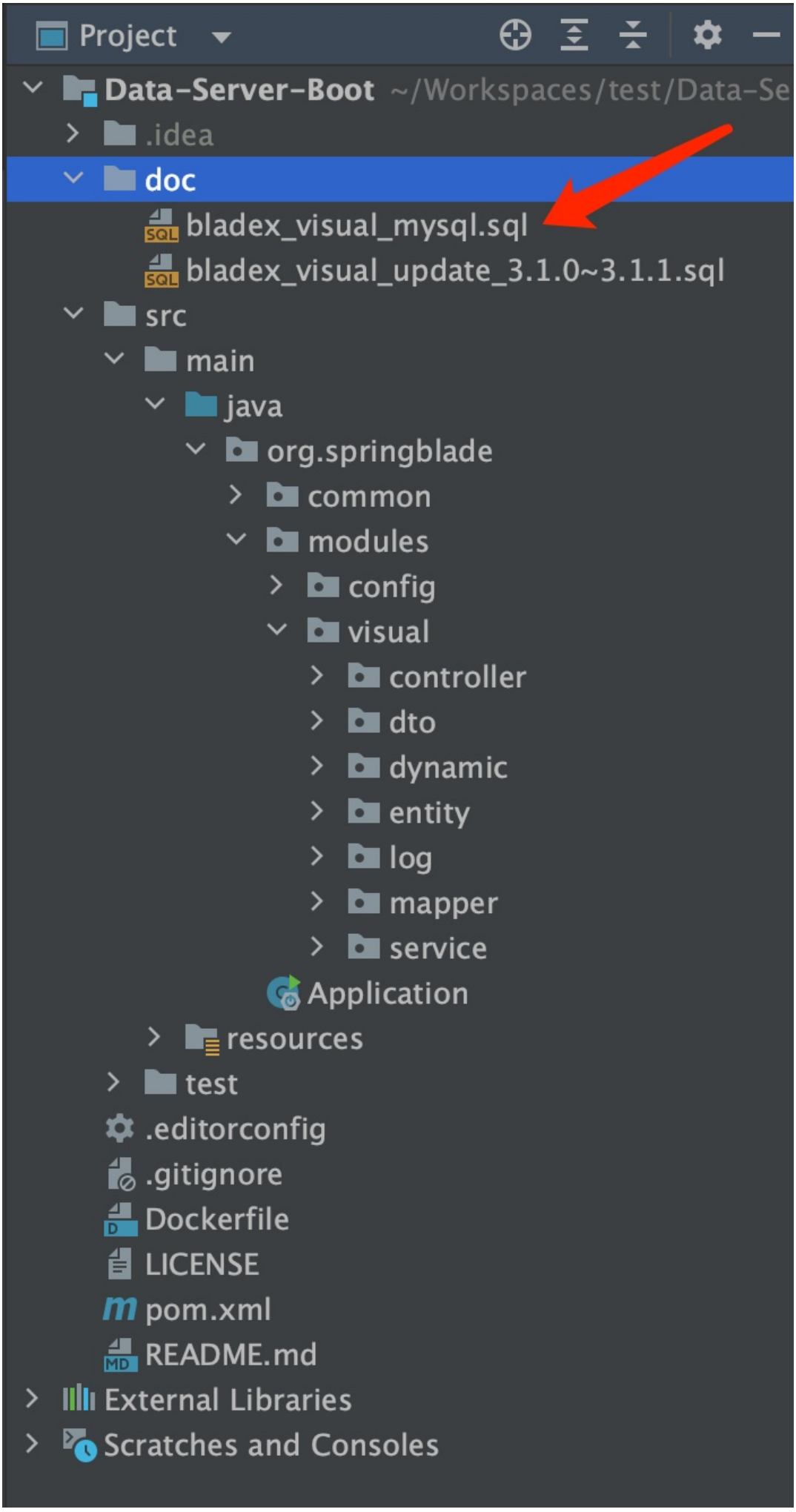


5. 工程构建完毕后，便可以看到核心服务了

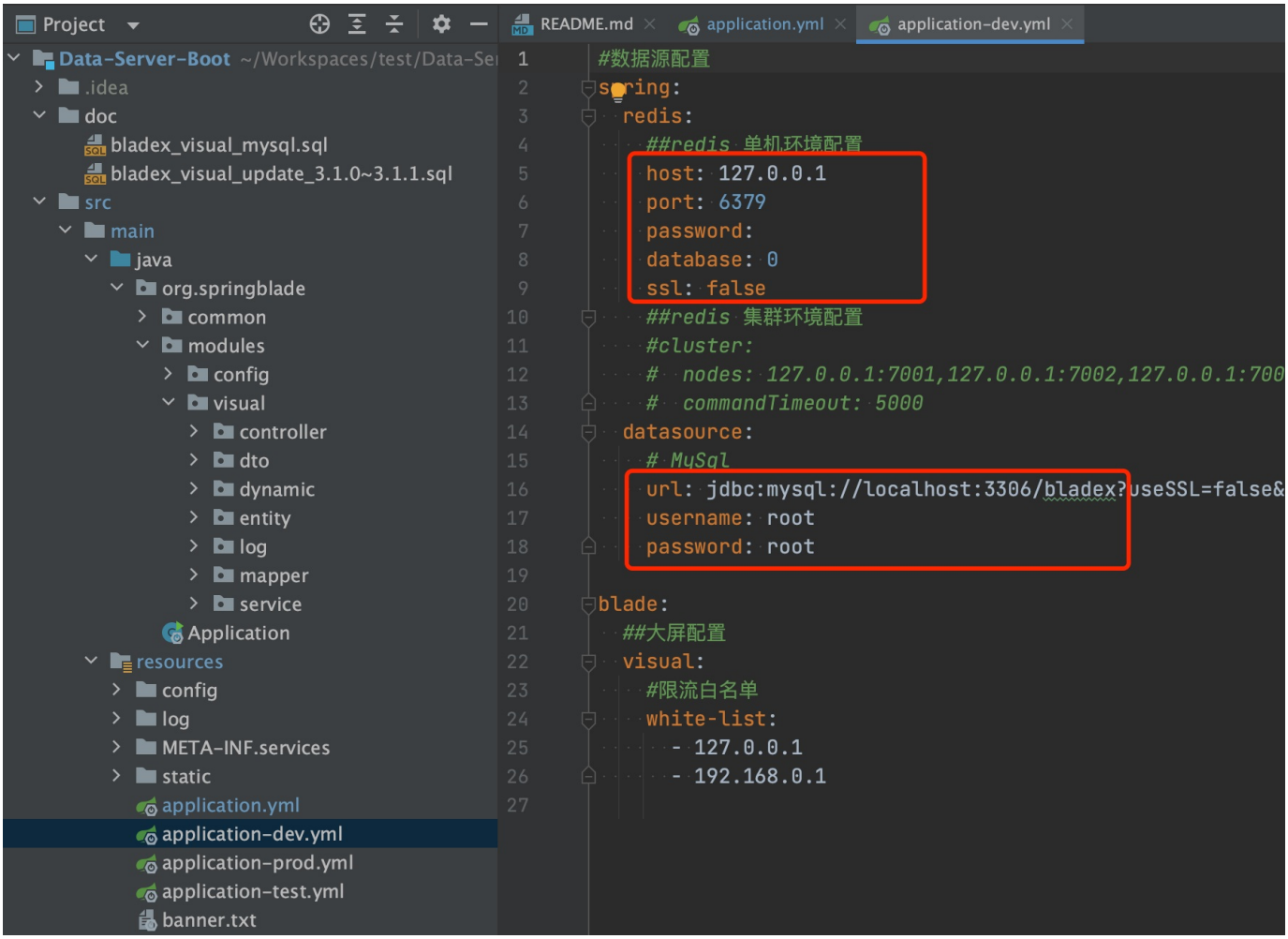


二、数据库配置

1. 创建一个空白数据库并导入如下sql (注意⚠：如果是最新版本，则无需关注下面的结构更新脚本)

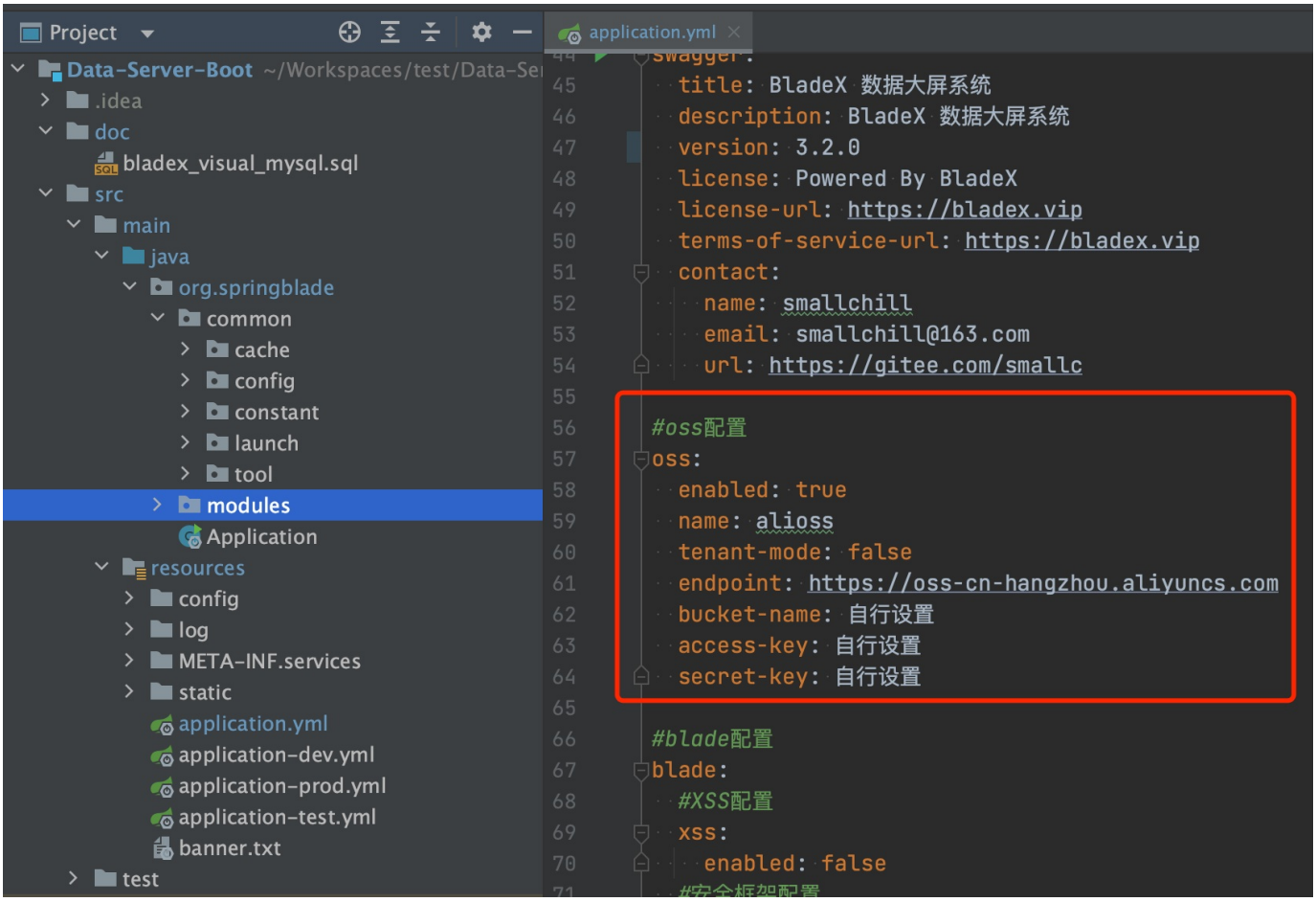

Project
Data-Server-Boot ~/Workspaces/test/Data-Se
.idea
doc
bladex_visual_mysql.sql
bladex_visual_update_3.1.0~3.1.1.sql
src
main
java
org.springblade
common
modules
config
visual
controller
dto
dynamic
entity
log
mapper
service
Application
resources
test
.editorconfig
.gitignore
Dockerfile
LICENSE
pom.xml
README.md
External Libraries
Scratches and Consoles

2. 到对应配置文件将数据库信息配置准确



三、对象存储配置

1. 框架支持minio、alioss、qiniu三种对象存储
2. 将对应配置到application.yml文件



3. 其中minio需要自行部署，如何启动部署请参考官网文档：http://docs.minio.org.cn/docs/

⚠minio版本使用注意：

- 自从2021年minio开源协议从Apache2.0变更为AGPLv3后，新协议就不适合免费商用了
- AGPLv3规定，只要系统部署后接入minio服务，具有文件存储或读取行为，不论是否改造源码，不论发布在何处，都需要将整个产品源码开源
- 所以如果使用minio开源版，请选择Apache2.0的最后一个版本：
  `minio-RELEASE.2021-04-22T15-44-28Z`
- 具体官方说明请见下图

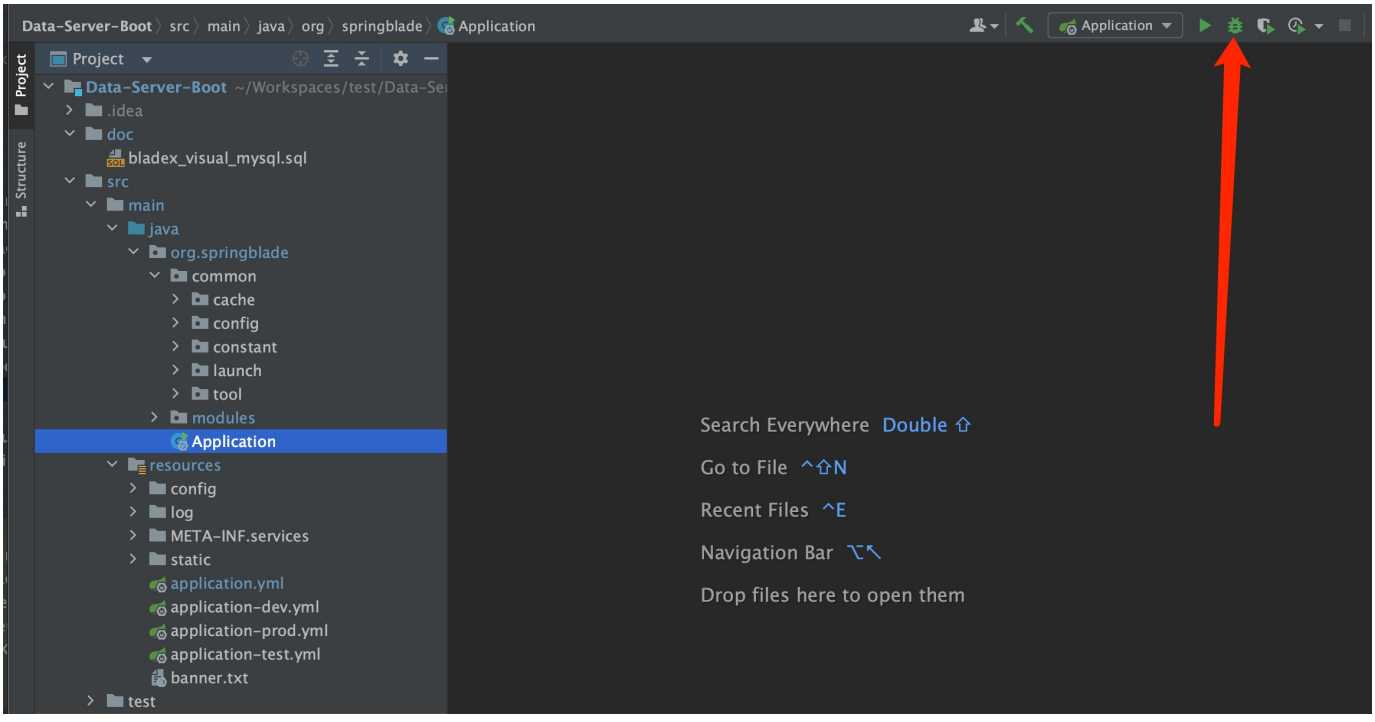
MINIO Customer Login

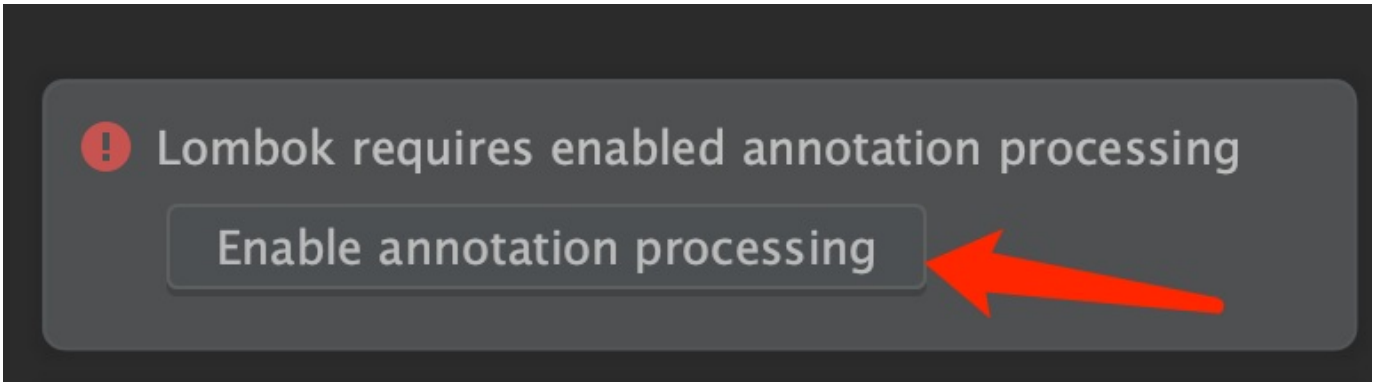
Product Solutions Docs Blog Resources Partner Pricing Download

License	MINIO AGPLv3 Free Software	MINIO STANDARD	MINIO ENTERPRISE
	Designed for developers who are building open source applications in compliance with the GNU AGPL v3 license and are able to support themselves. It is fully featured. If you distribute, host or create derivative works of the MinIO software over the network, the GNU AGPL v3 license requires that you also distribute the complete, corresponding source code of the combined work under the same GNU AGPL v3 license. This requirement applies whether or not you modified MinIO.	Designed for customers who require a commercial license and can mostly self-support but want the peace of mind that comes with the MinIO's unique direct-to-engineer interaction model and suite of operational capabilities delivered through the SUBNET portal. The Standard version is fully featured but with SLA limitations. To learn more about the features and capabilities available to commercial license holders click here.	Designed for mission critical environments where both a license and strict SLAs are required. The Enterprise version is fully featured but comes with additional capabilities over the Standard license. To learn more about the features and capabilities available to commercial license holders click here.
FEATURES			
Unit Price		$10 per TiB per Month (Minimum of 200TiB)	$20 per TiB per Month (Minimum of 100TiB)
Software Release	Upstream	Enterprise Hardened	Enterprise Hardened
Release Support	Current	1 Year Long Term Support	5 Years Long Term Support
SLA	No SLA	<48 Hours (Local Business Hours)	<1 Hour

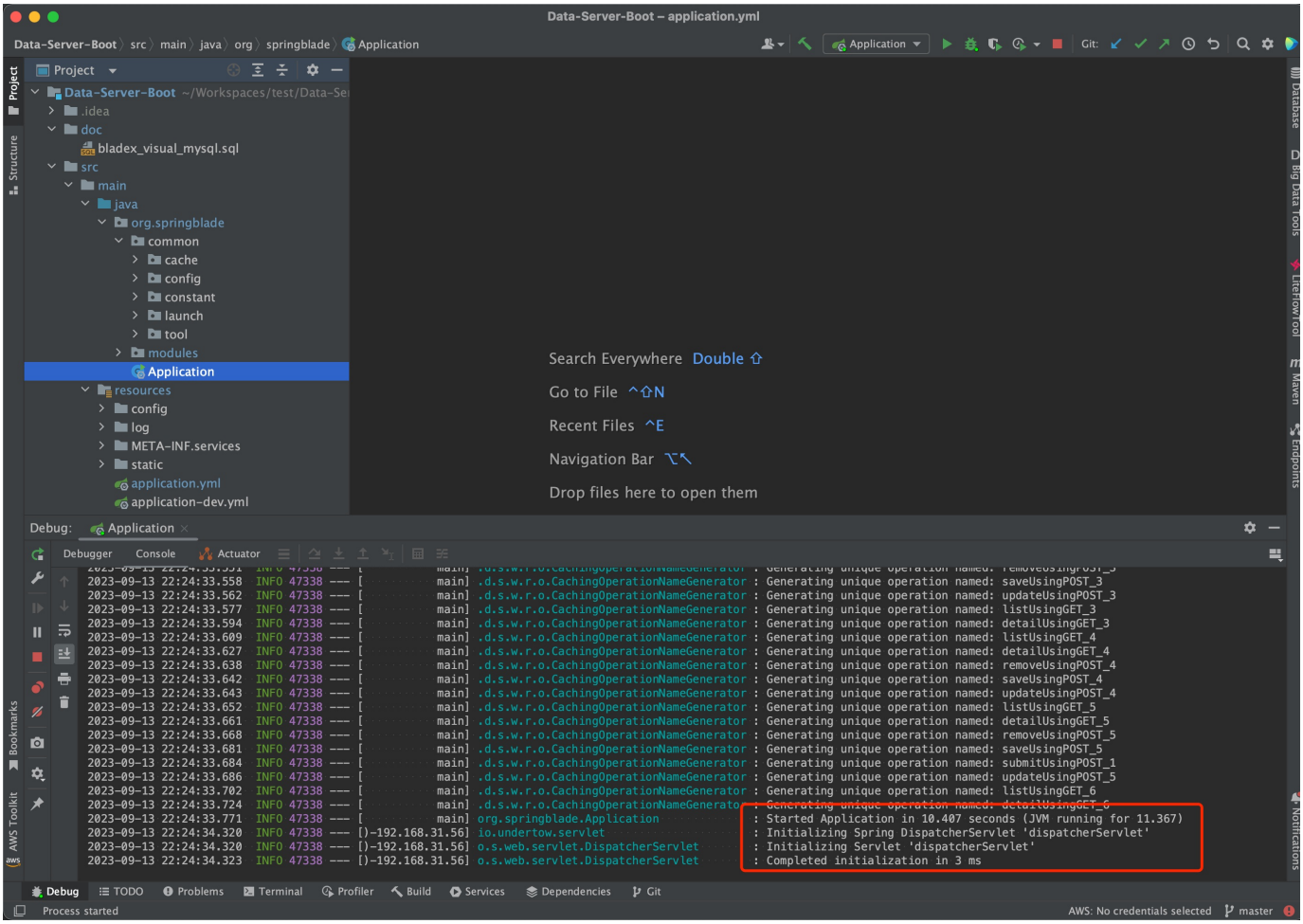
四、工程启动

1. 一切准备完毕，我们启动Application，启动若出现Lombok的提示，点击Enable按钮即可

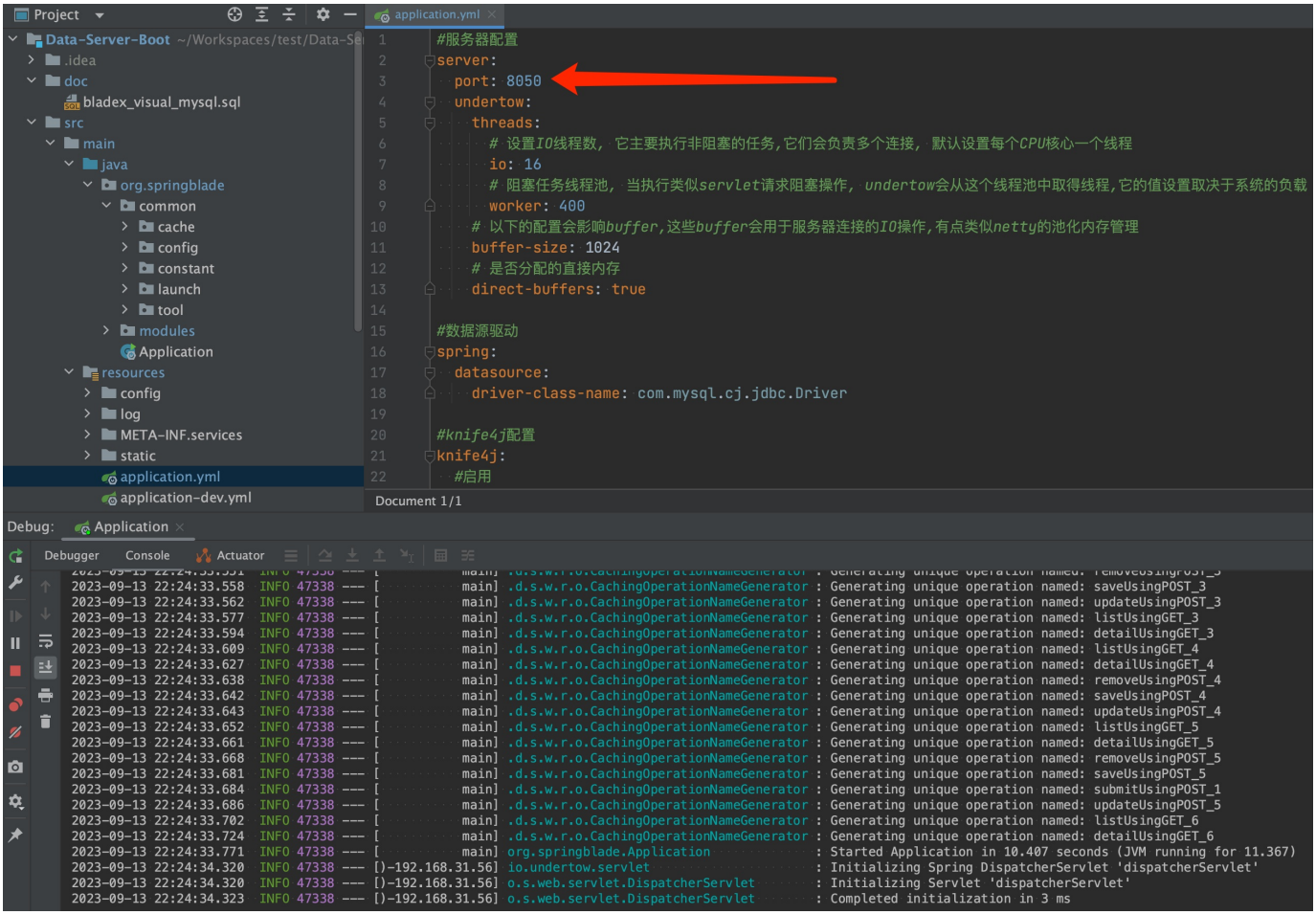




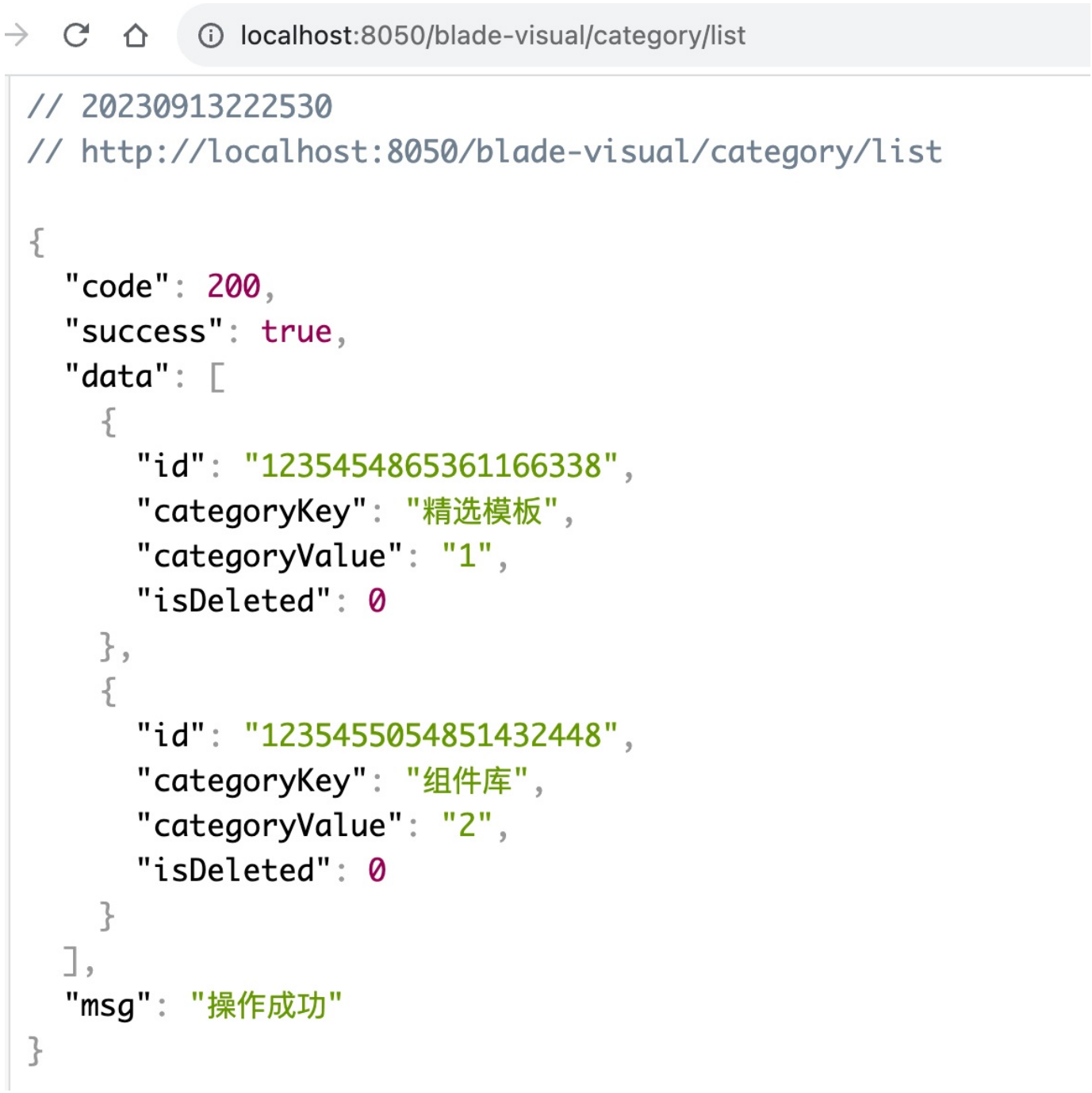
2. 若配置无误，我们就可以看到启动成功的日志



3. 服务端口为8050，我们访问如下地址 ，测试是否有正确数据返回 http://localhost:8050/blade-visual/category/list



4. 若正确返回数据，则说明服务启动成功

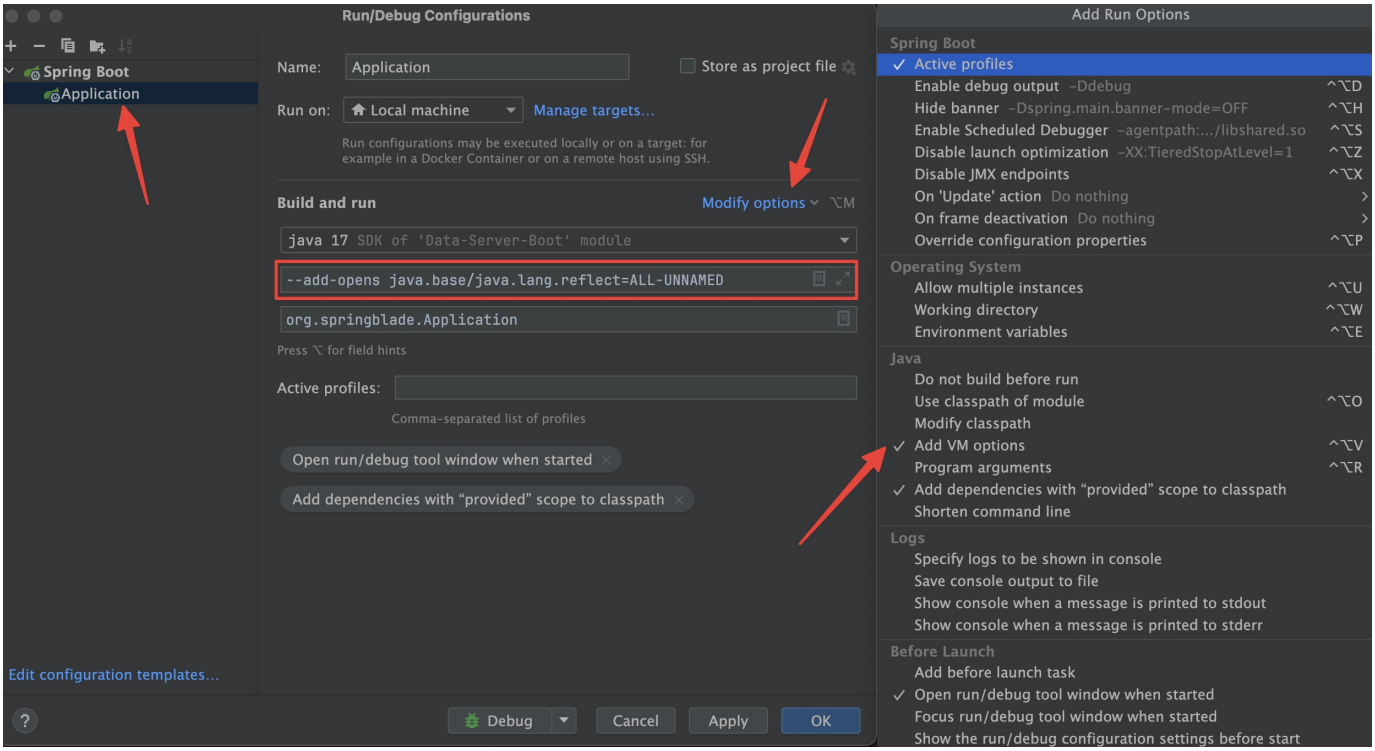


5. 若返回异常，是因为升级JDK17后，Java 模块化系统（Java Module System）的安全限制导致的针对反射等场景有可能会出现如下错误：

```
Caused by: java.lang.reflect.InaccessibleObjectException: Unable to make field
protected java.lang.reflect.InvocationHandler java.lang.reflect.Proxy.h accessi
ble: module java.base does not "opens java.lang.reflect" to unnamed module @23c
30a20
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(Acces
sibleObject.java:354)
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(Acces
sibleObject.java:297)
    at java.base/java.lang.reflect.Field.checkCanSetAccessible(Field.java:178)
    at java.base/java.lang.reflect.Field.setAccessible(Field.java:172)
    at org.apache.ibatis.reflection.invoker.GetFieldInvoker.invoke(GetFieldInvo
```

```
ker.java:38)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.getBeanProperty(BeanWra
pper.java:158)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.get(BeanWrapper.java:50
)
    at org.apache.ibatis.reflection.MetaObject.getValue(MetaObject.java:115)
    at org.apache.ibatis.reflection.MetaObject.metaObjectForProperty(MetaObject
.java:123)
    at org.apache.ibatis.reflection.wrapper.BaseWrapper.getChildValue(BaseWrapp
er.java:117)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.get(BeanWrapper.java:46
)
    at org.apache.ibatis.reflection.MetaObject.getValue(MetaObject.java:115)
    at org.springblade.core.mp.plugins.SqlLogInterceptor.intercept(SqlLogInterc
eptor.java:46)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy216.query(Unknown Source)
    at org.apache.ibatis.executor.SimpleExecutor.doQuery(SimpleExecutor.java:65
)
    at org.apache.ibatis.executor.BaseExecutor.queryFromDatabase(BaseExecutor.j
ava:336)
    at org.apache.ibatis.executor.BaseExecutor.query(BaseExecutor.java:158)
    at com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor.interc
ept(MybatisPlusInterceptor.java:81)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy215.query(Unknown Source)
    at org.apache.ibatis.session.defaults.DefaultSqlSession.selectList(DefaultS
qlSession.java:154)
    ... 115 common frames omitted
```

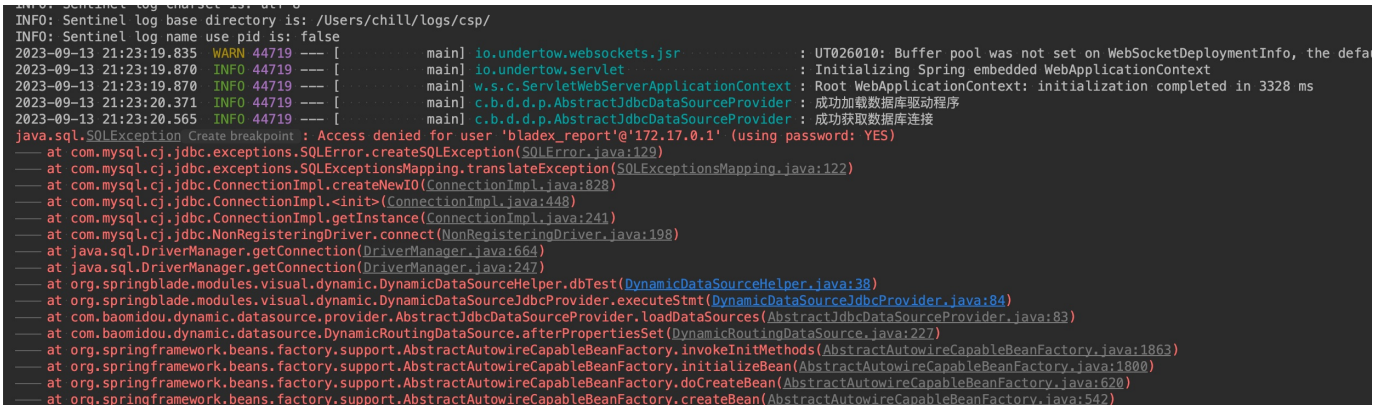
6. 遇到上述情况，在启动类增加命令 `--add-opens java.base/java.lang.reflect=ALL-UNNAMED` 即可



7. JAR 包启动时也需要加入此配置，具体命令如下

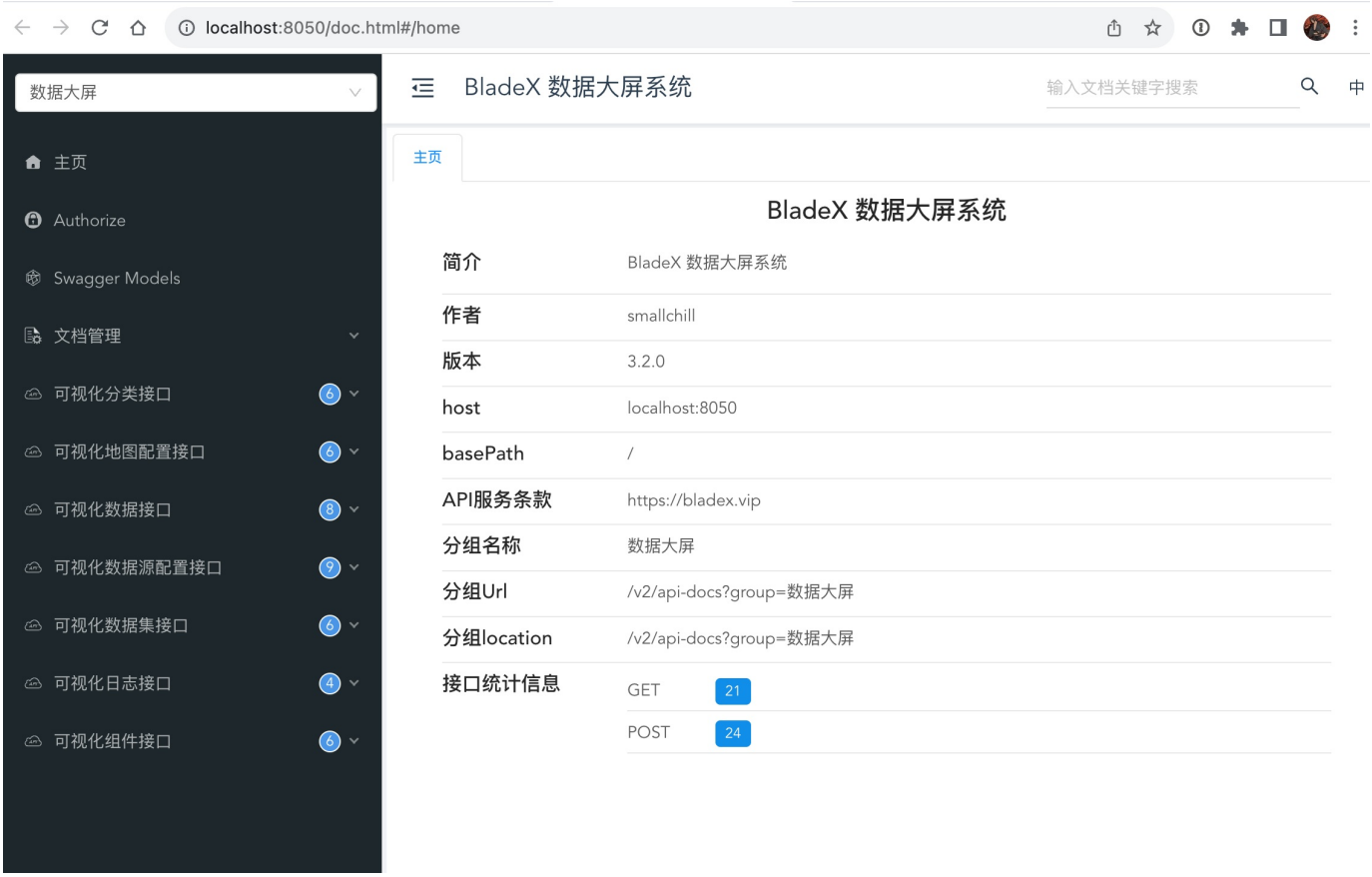
```
java --add-opens java.base/java.lang.reflect=ALL-UNNAMED -jar your-application.jar
```

8. 若启动期间出现数据库加载失败，但是最后又启动成功，不必担心，这是因为大屏后端有动态数据源的功能，会实时加载配置的多个数据源，若连接失败，会抛出异常并加载下一个链接，不会影响整体使用。若不想出现这些提示，将blade_visual_db表的默认数据清空即可

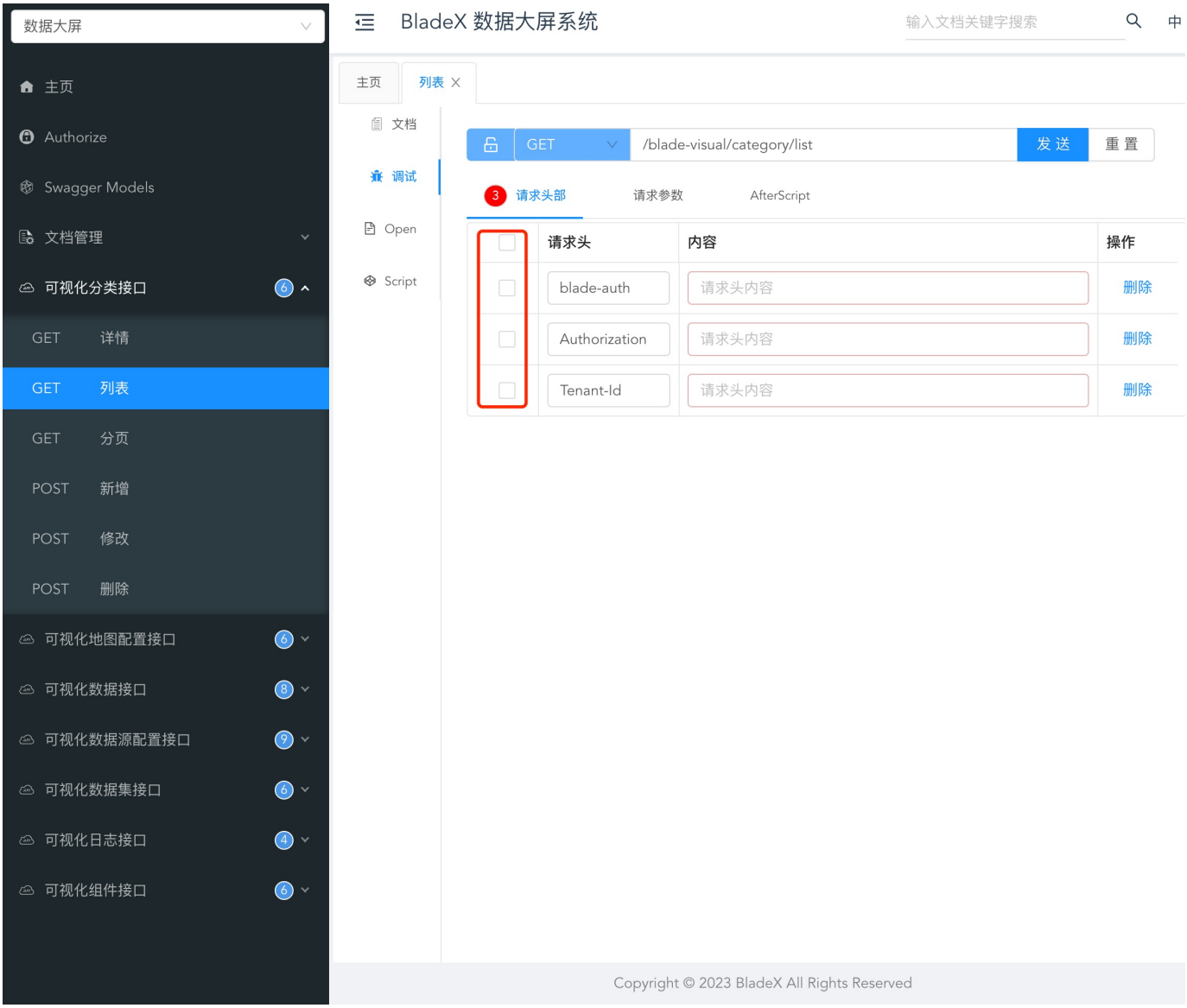


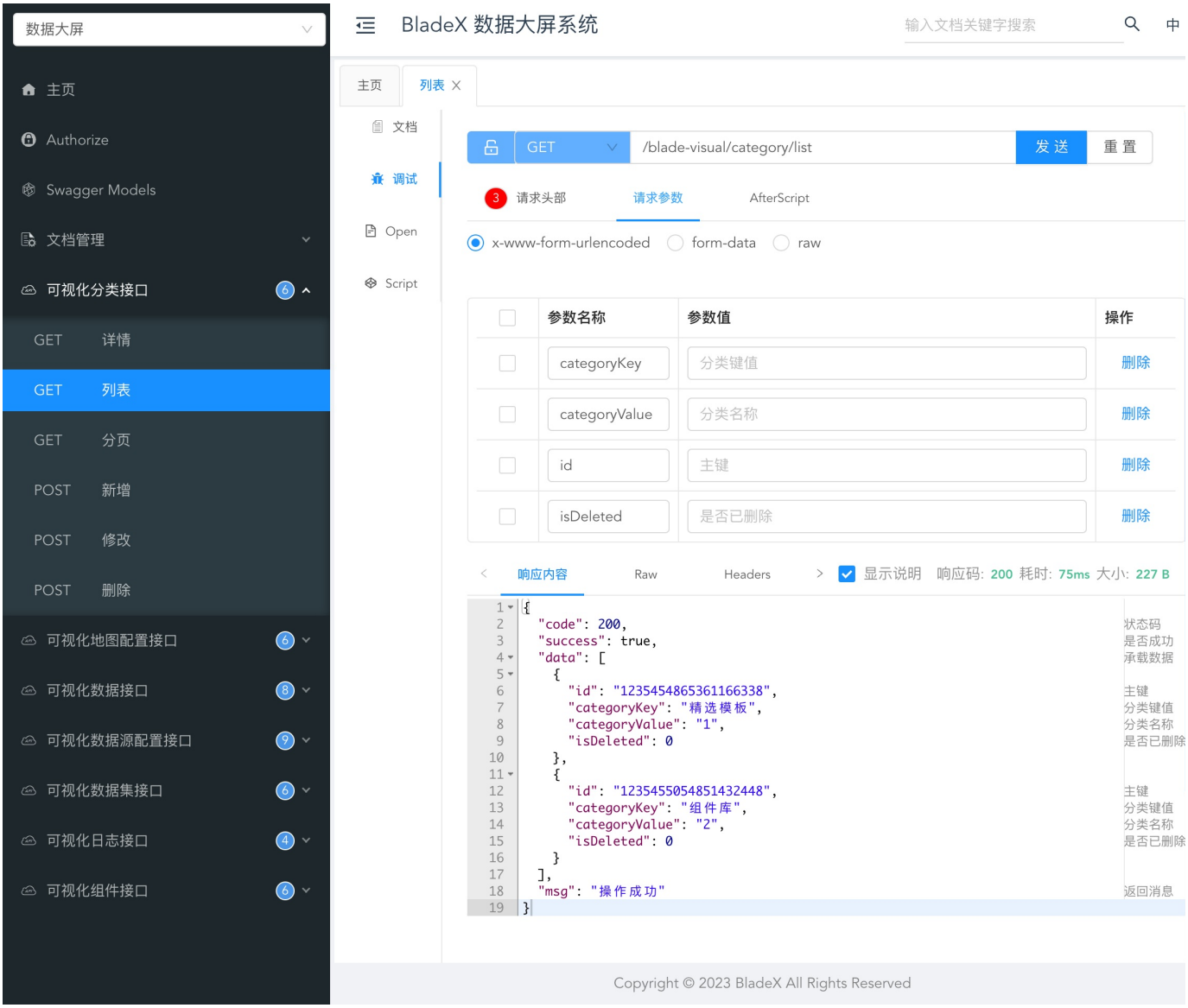
五、接口文档

1. 启动服务成功后，便可以使用接口文档，访问接口文档地址：http://localhost:8050/doc.html



2. 找一个接口点击查看数据正确返回

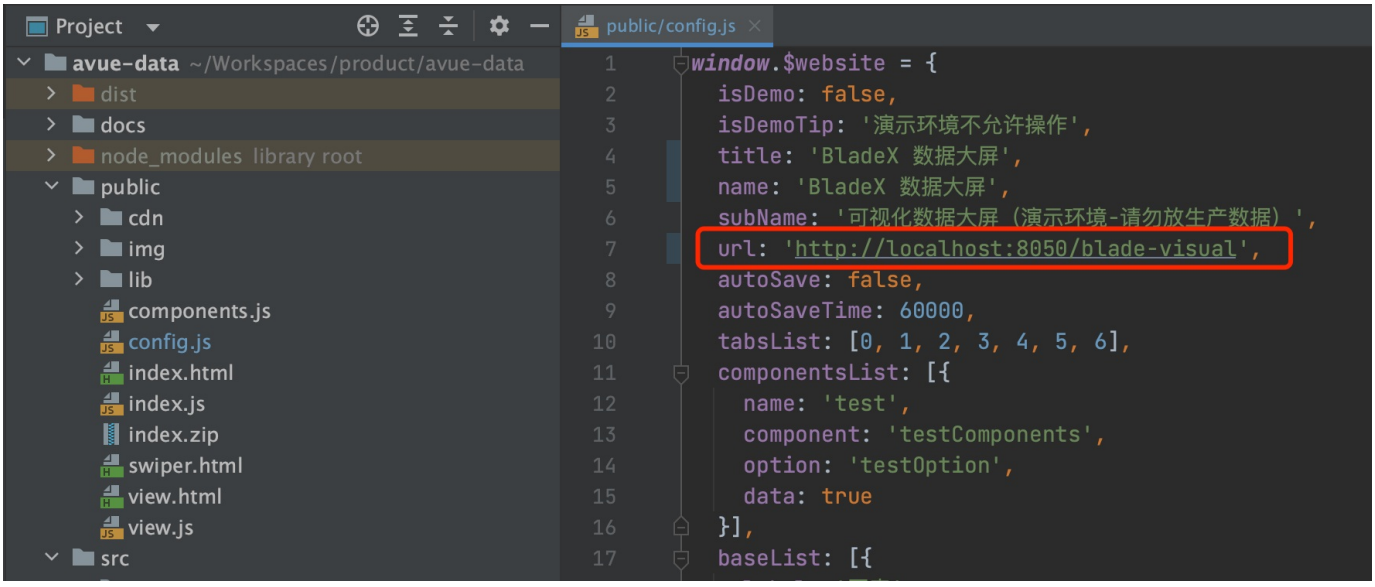

数据大屏
主页
Authorize
Swagger Models
文档管理
可视化分类接口
GET 详情
GET 列表
GET 分页
POST 新增
POST 修改
POST 删除
可视化地图配置接口
可视化数据接口
可视化数据源配置接口
可视化数据集接口
可视化日志接口
可视化组件接口
BladeX 数据大屏系统
输入文档关键字搜索
中
主页
列表
文档
调试
Open
Script
GET
/blade-visual/category/list
发 送
重 置
3 请求头部
请求参数
AfterScript
请求头
内容
操作
blade-auth
请求头内容
删除
Authorization
请求头内容
删除
Tenant-Id
请求头内容
删除
Copyright © 2023 BladeX All Rights Reserved



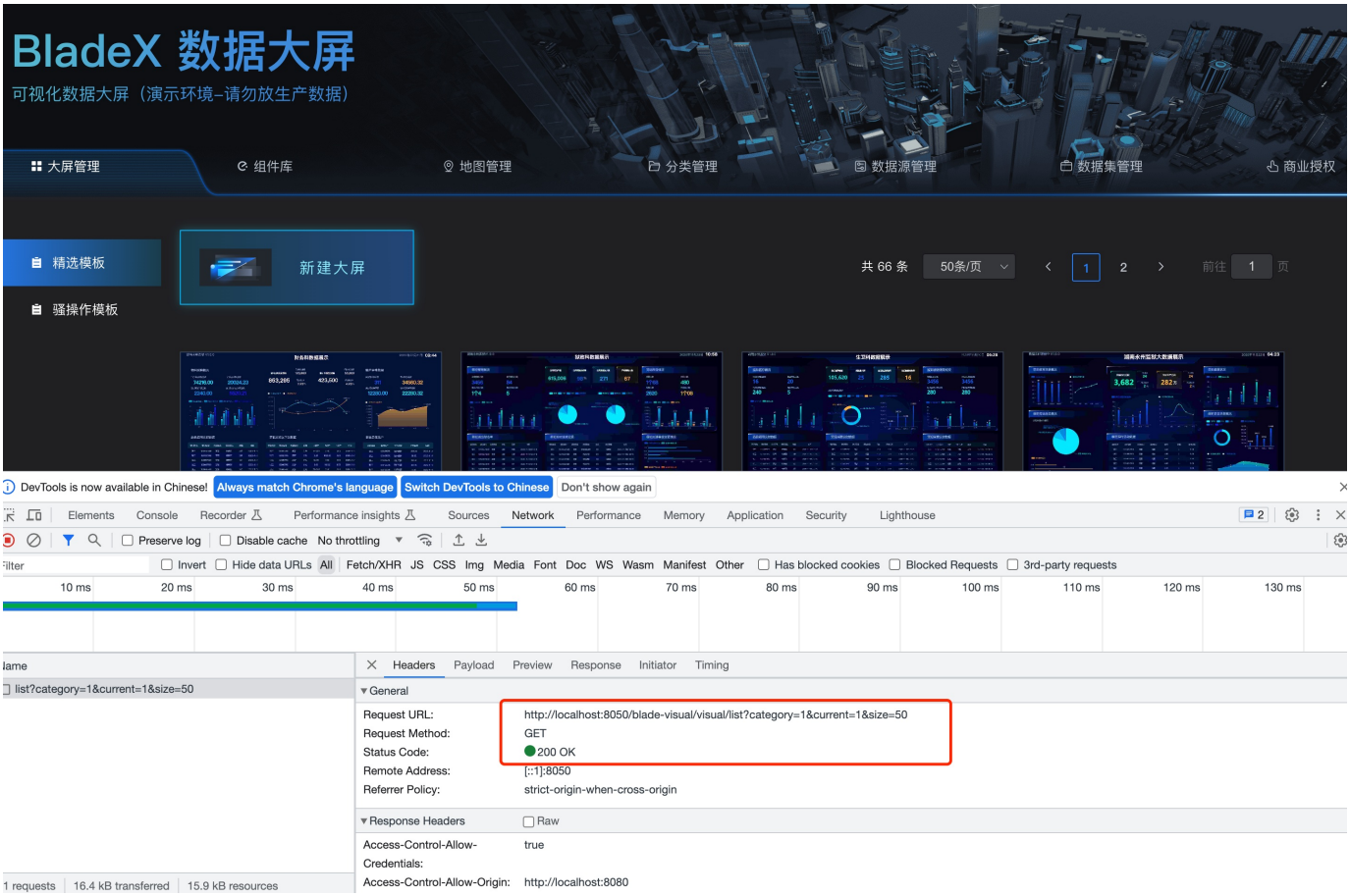
3. 至此后端部署全流程结束

前端配置地址

1. 前往前端的config.js文件，将后端接口地址改为：http://localhost:8050/blade-visual



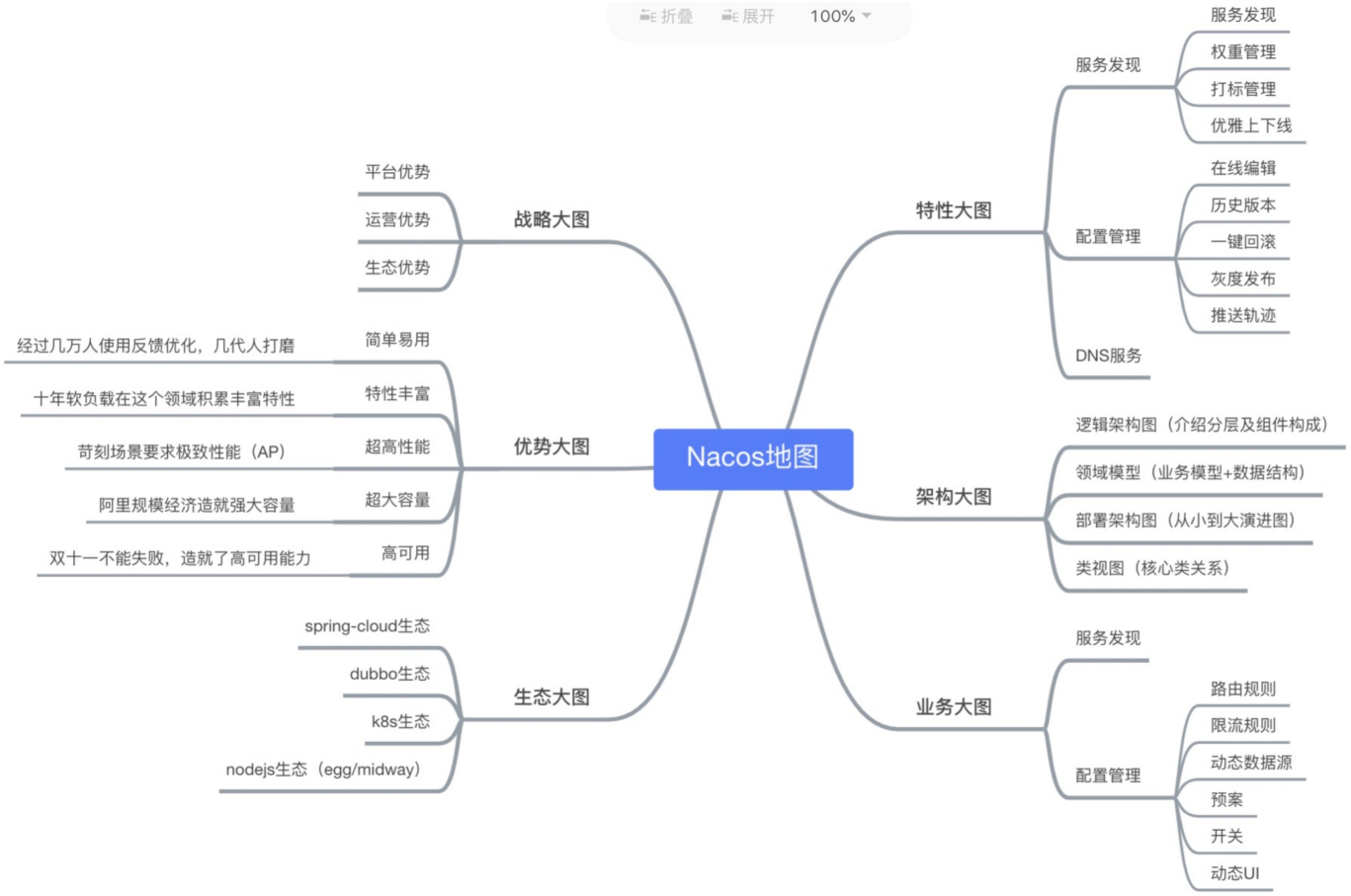
2. 刷新前端大屏可以看到数据获取成功



Nacos启动

概念

Nacos是阿里巴巴开源的一款支持服务注册与发现，配置管理以及微服务管理的组件。用来取代以前常用的注册中心（zookeeper，eureka等等），以及配置中心（spring cloud config等等）。Nacos是集成了注册中心和配置中心的功能，做到了二合一。



Nacos安装

1.编译安装

- 参考官方文档：https://nacos.io/zh-cn/docs/quick-start.html
- 下载 2.x 版本的nacos

2.Docker安装

- 拉取镜像： `docker pull nacos/nacos-server:v2.1.2`
- 运行镜像：

```
docker run --name nacos-standalone -e MODE=standalone -d -p 8848:8848 -p 9848:
```

3.访问地址

- 地址：http://localhost:8848/nacos

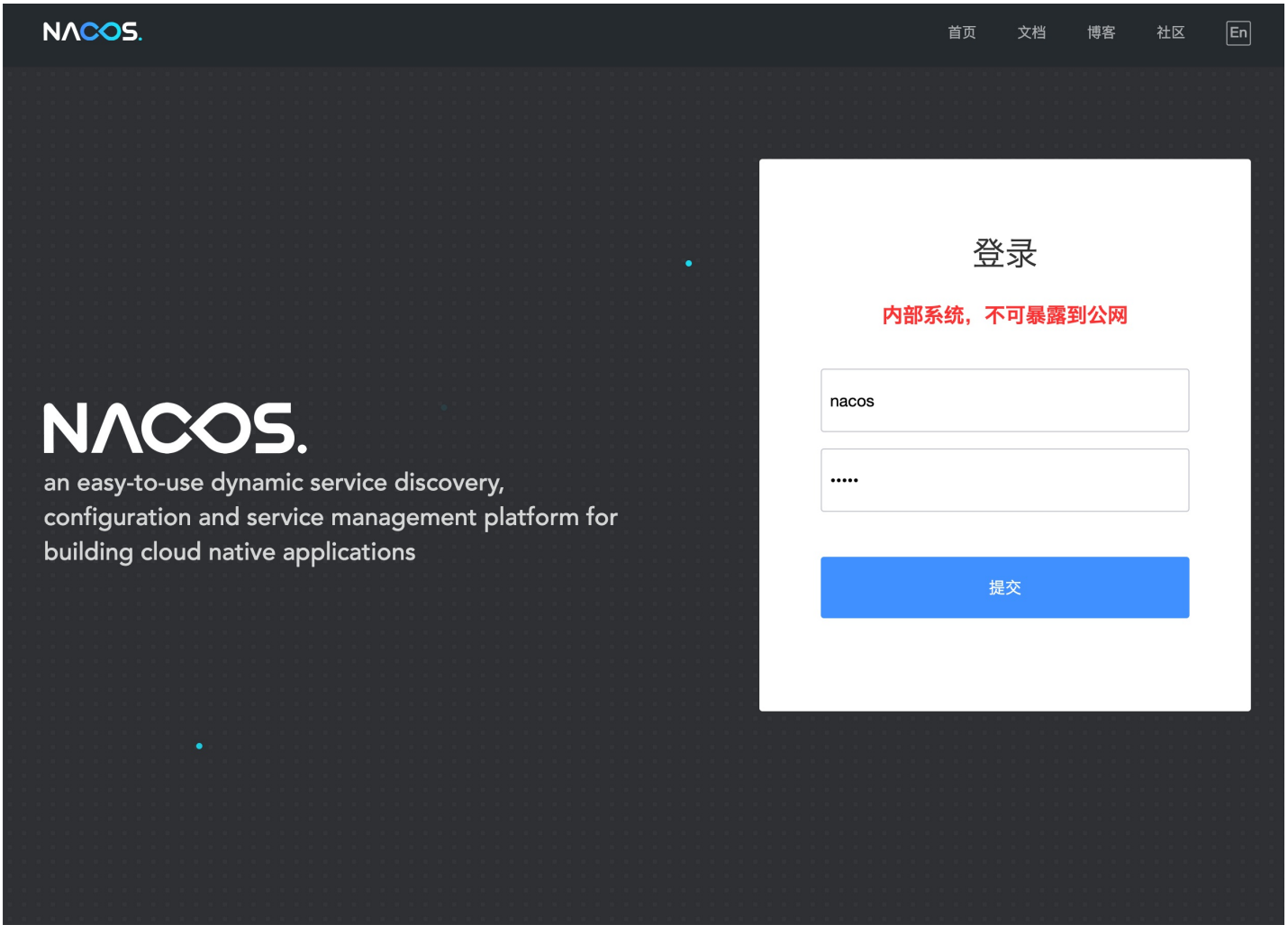
- 帐号密码都为：nacos

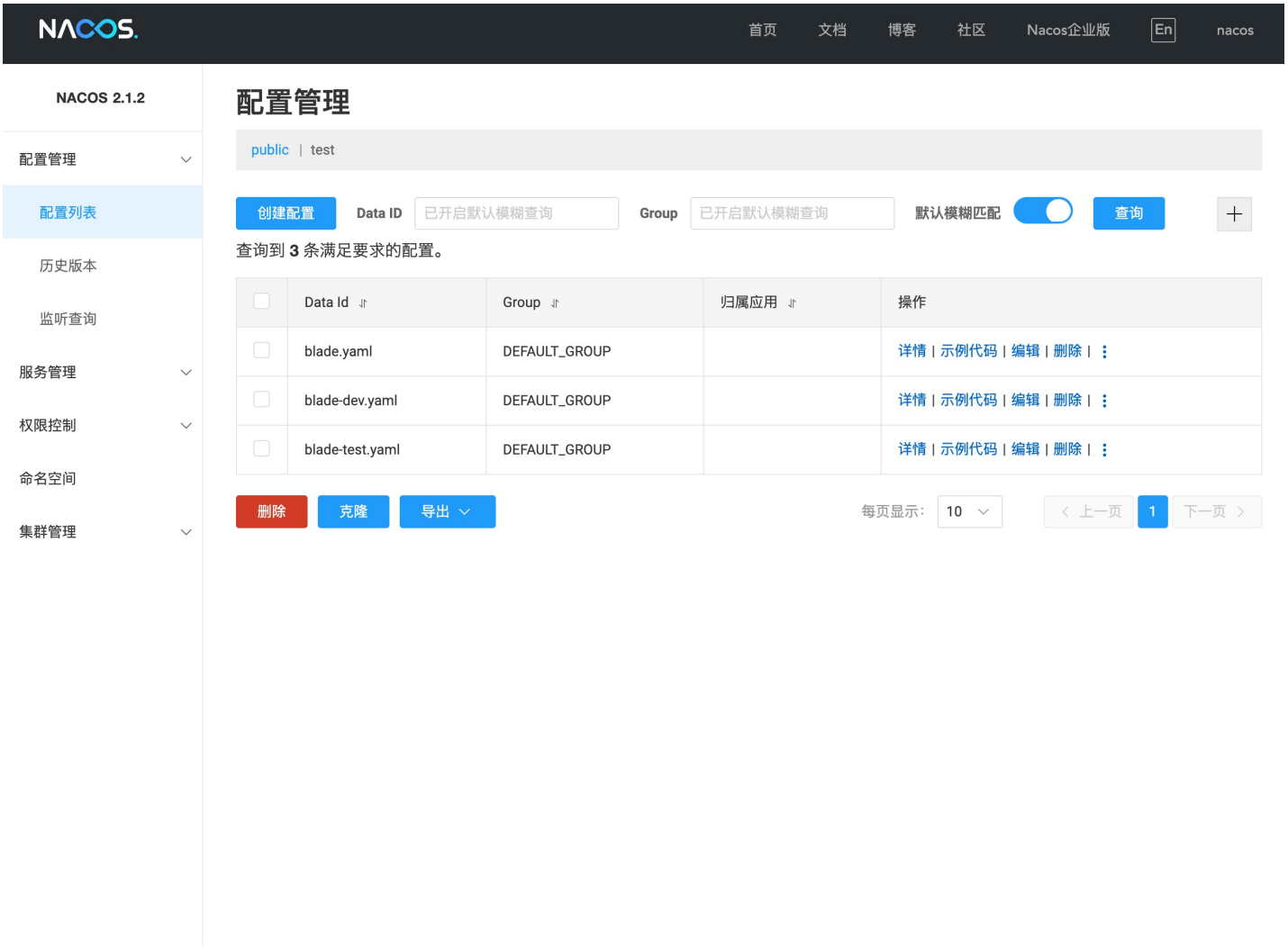
4.注意点⚠

- Nacos2.0版本相比1.X新增了gRPC的通信方式，需要增加对外开放2个端口。新增端口是在配置的主端口(server.port)基础上，进行一定偏移量自动生成。
- 客户端拥有相同的计算逻辑，用户如同1.X的使用方式，配置主端口(默认8848)，通过相同的偏移量，计算对应gRPC端口(默认9848)。如果客户端和服务端之前存在端口转发，或防火墙时，需要对端口转发配置和防火墙配置做相应的调整。

端口	与主端口的偏移量	描述
9848	1000	客户端gRPC请求服务端端口，用于客户端向服务端发起连接和请求
9849	1001	服务端gRPC请求服务端端口，用于服务间同步等

Nacos界面

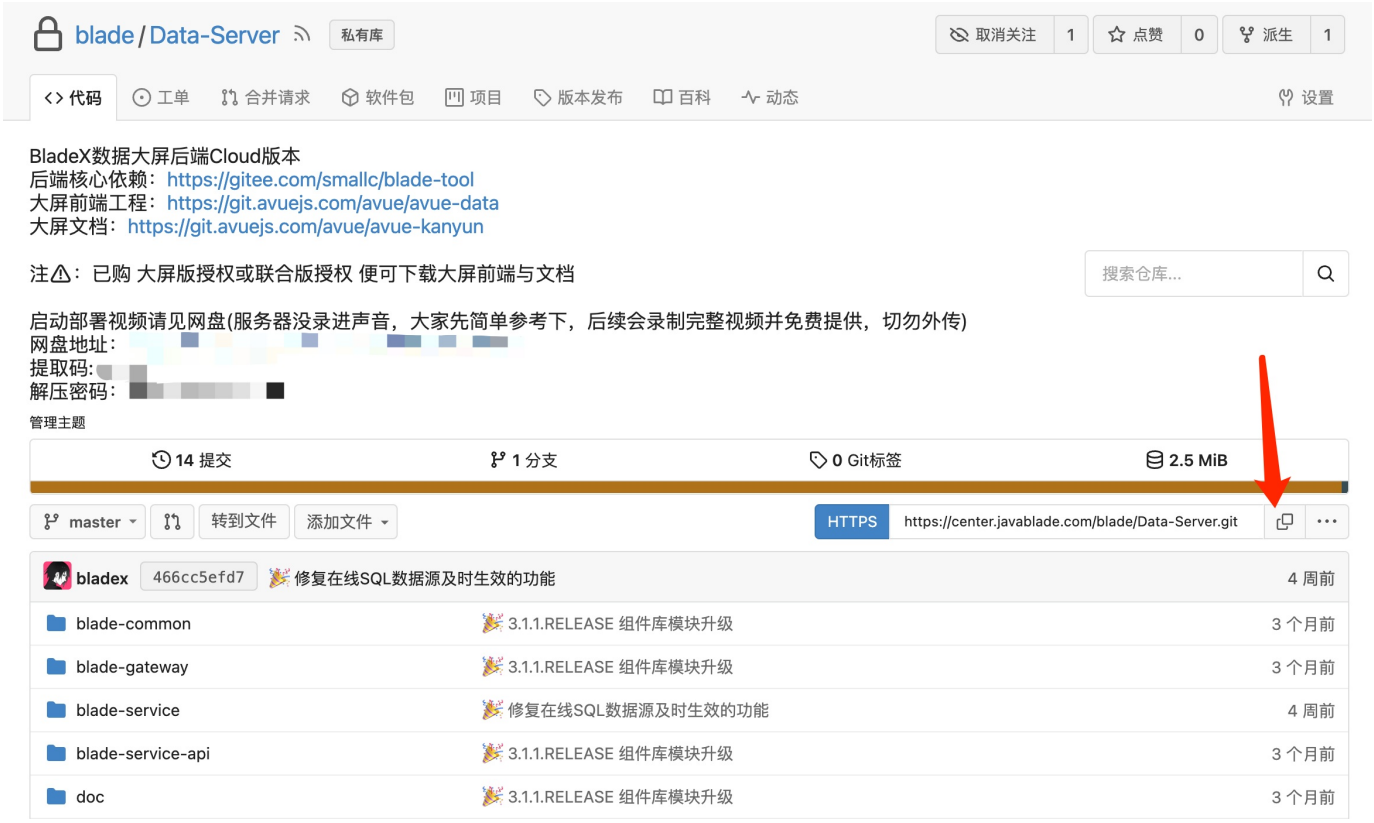



NACOS.
首页
文档
博客
社区
Nacos企业版
En
nacos
NACOS 2.1.2
配置管理
配置列表
历史版本
监听查询
服务管理
权限控制
命名空间
集群管理
配置管理
public | test
创建配置
Data ID
已开启默认模糊查询
Group
已开启默认模糊查询
默认模糊匹配
查询
查询到 3 条满足要求的配置。
Data Id
Group
归属应用
操作
blade.yaml
DEFAULT_GROUP
详情 | 示例代码 | 编辑 | 删除 |
blade-dev.yaml
DEFAULT_GROUP
详情 | 示例代码 | 编辑 | 删除 |
blade-test.yaml
DEFAULT_GROUP
详情 | 示例代码 | 编辑 | 删除 |
删除
克隆
导出
每页显示: 10
上一页
1
下一页

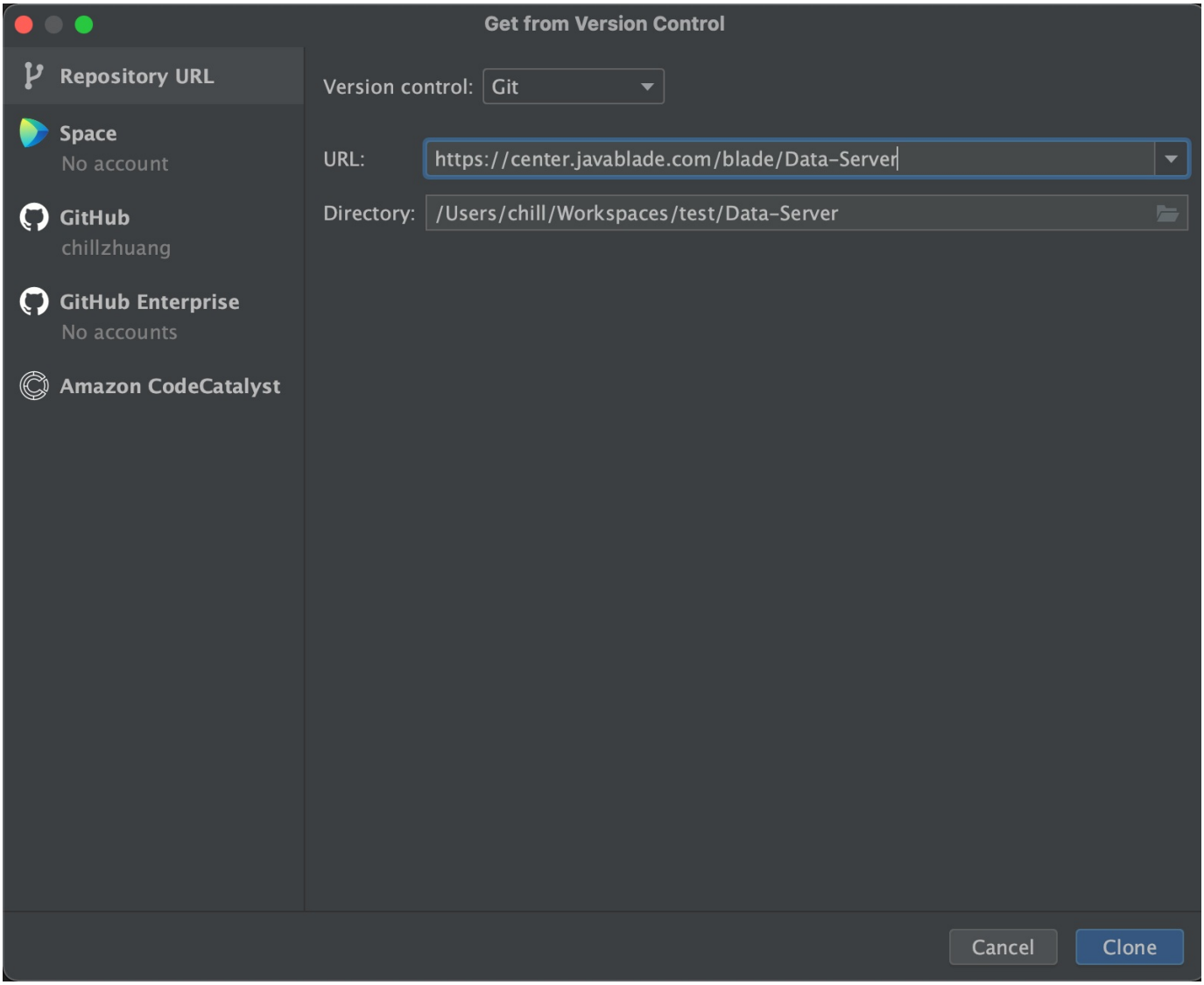
Cloud版后端启动

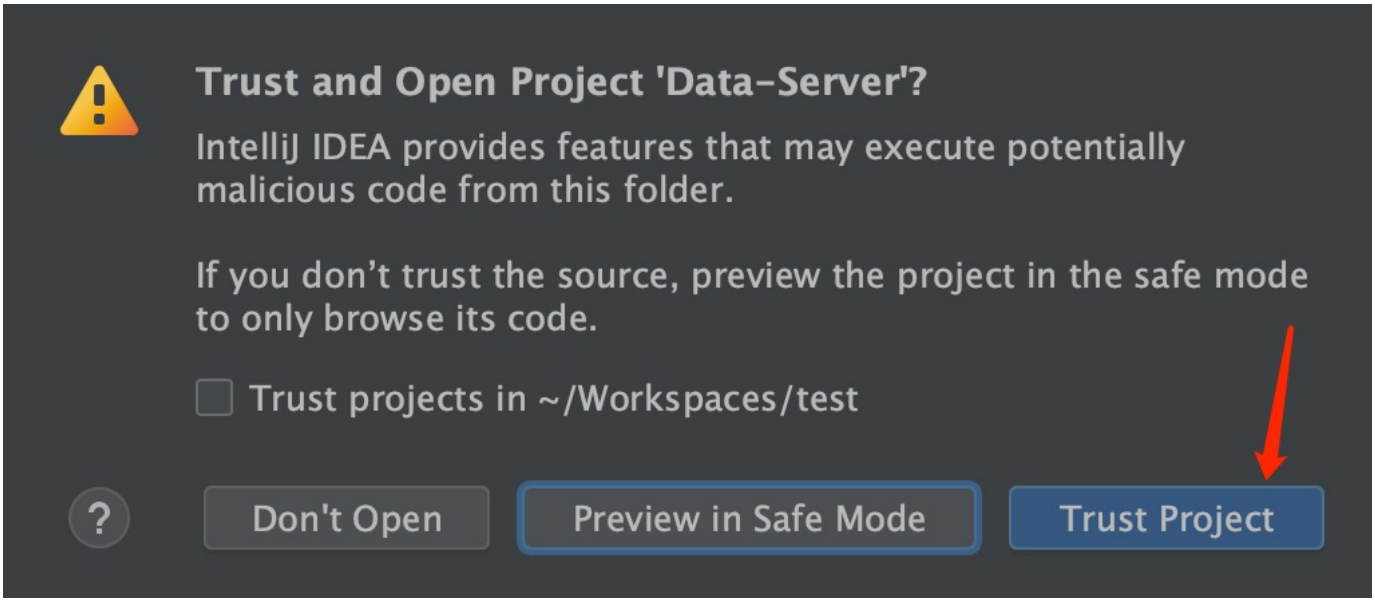
一、工程导入

1. 使用BladeX商业账号登录git私服：https://center.javablade.com/blade/Data-Server
2. 复制地址

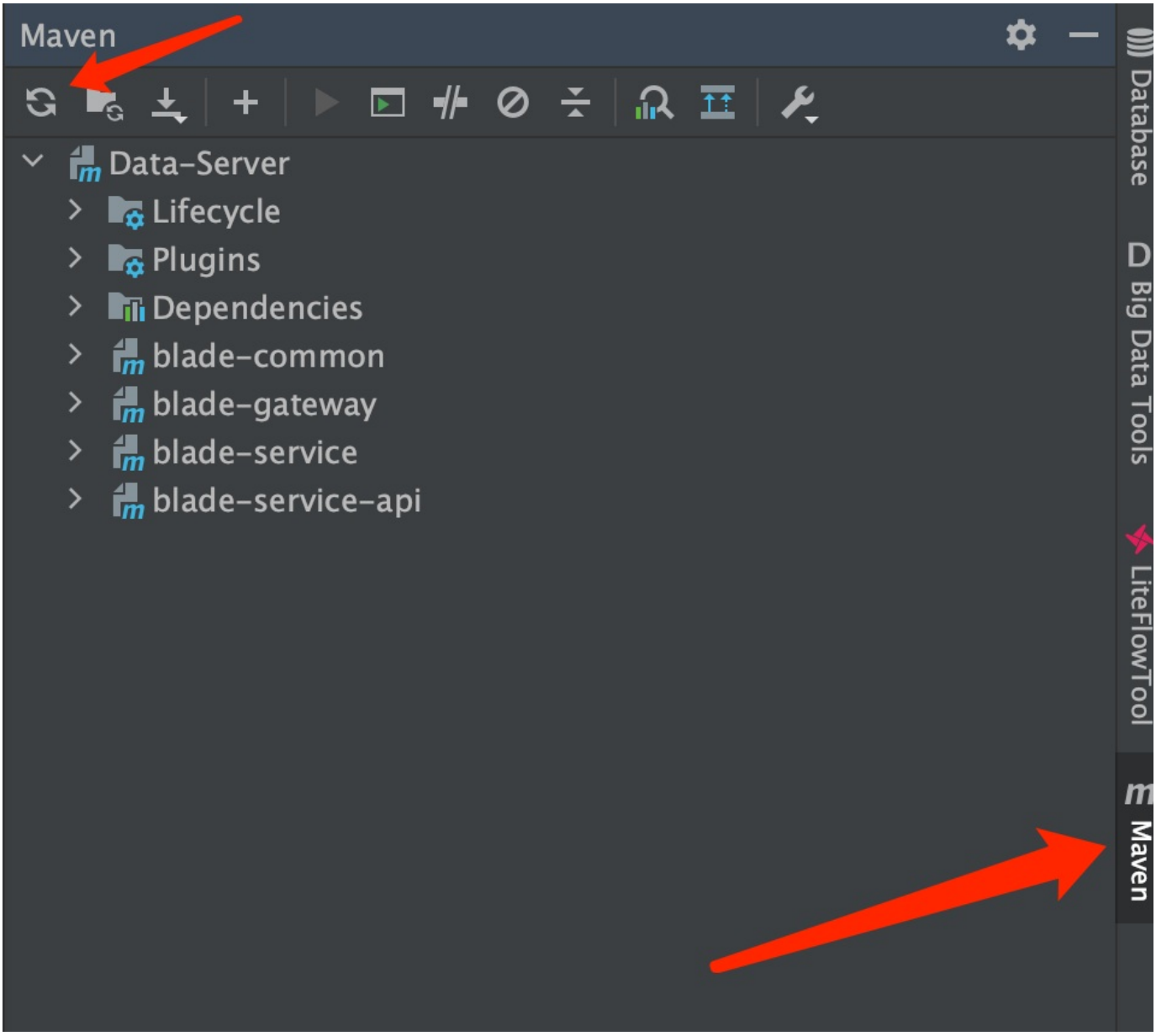


3. 导入工程

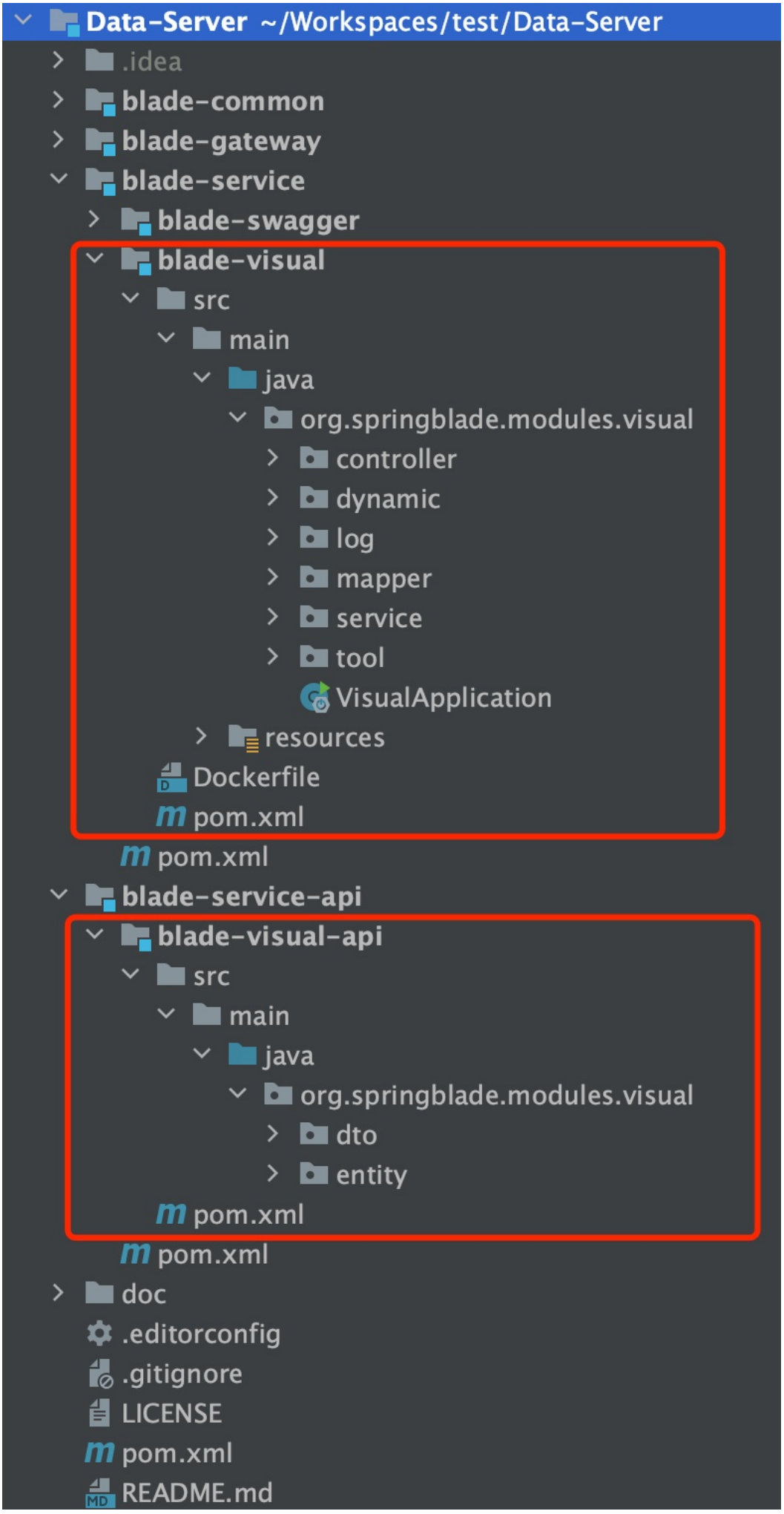




4. 导入成功后等待maven依赖加载完毕，如果依赖下载失败请多点击刷新按钮

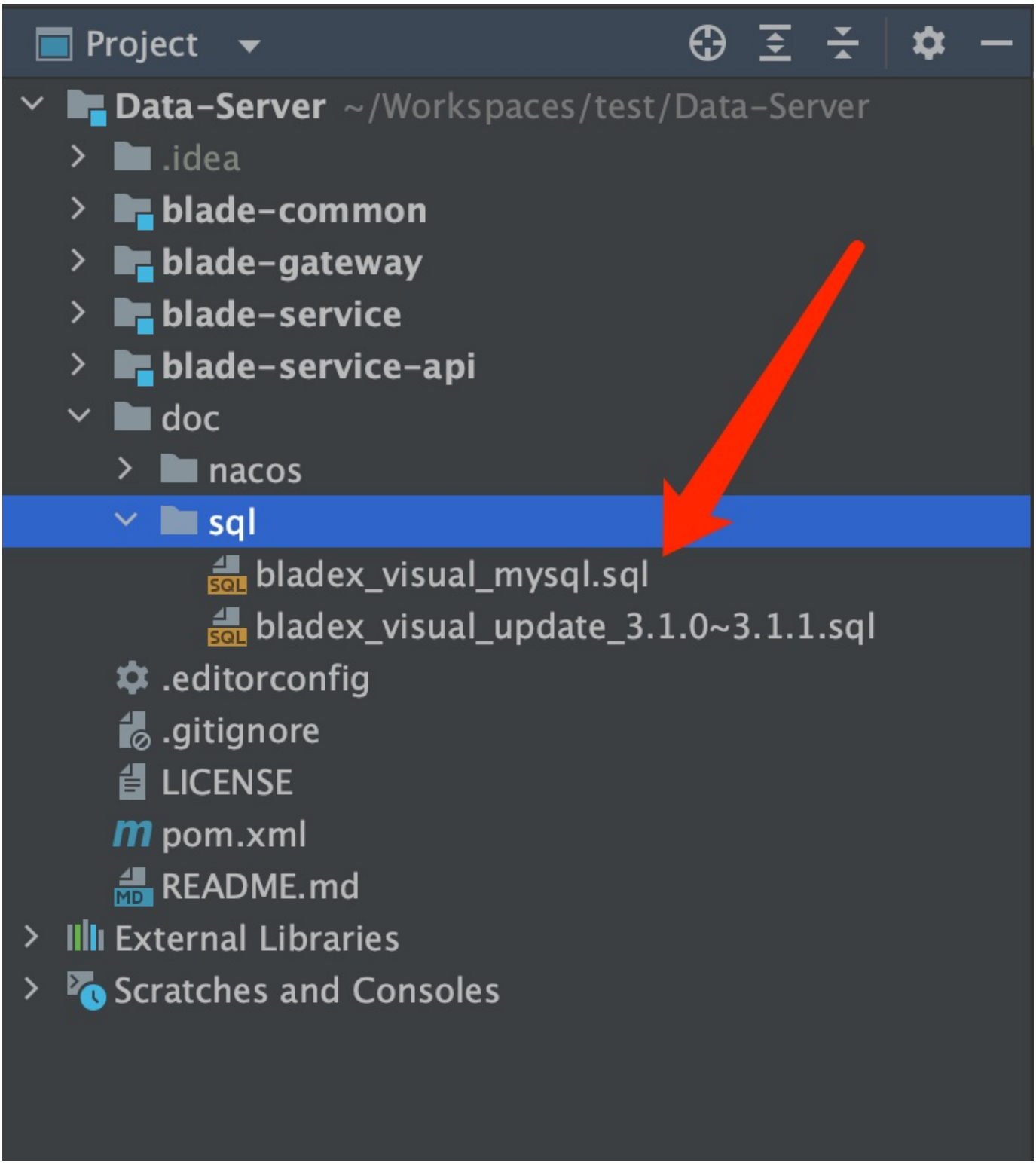


5. 工程构建完毕后，便可以看到核心服务了

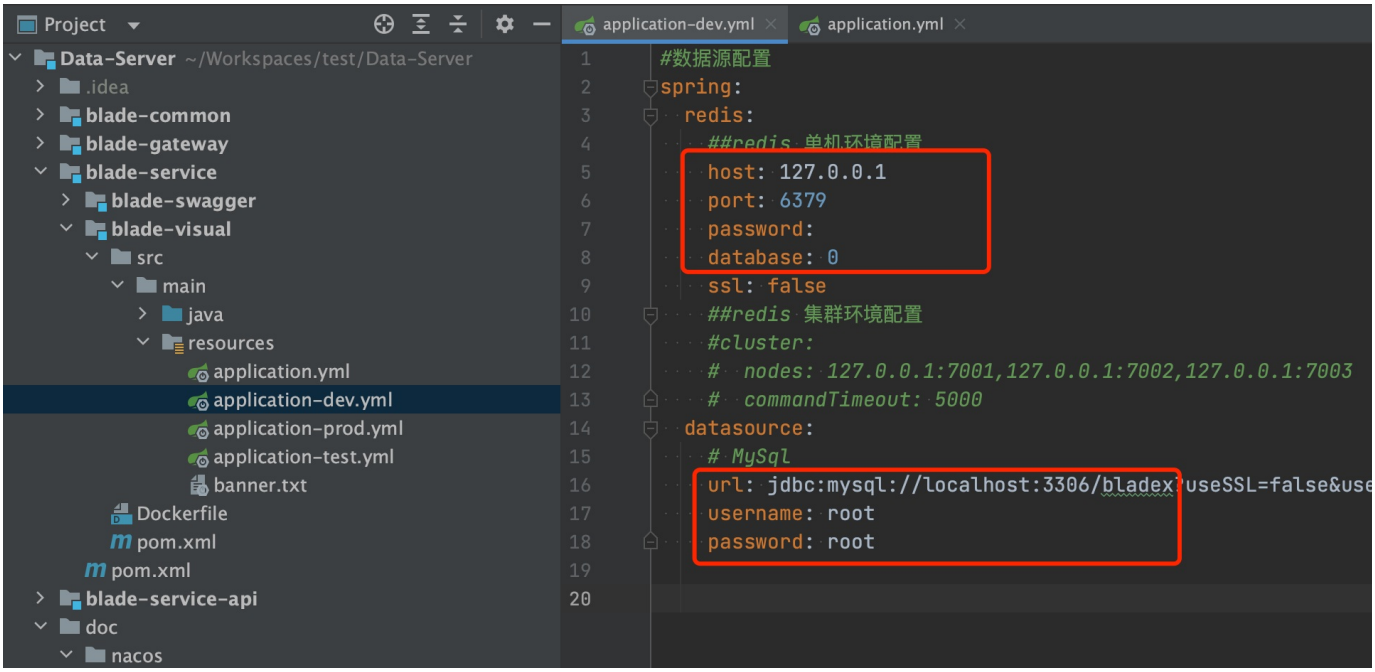

Data-Server ~/Workspaces/test/Data-Server
.idea
blade-common
blade-gateway
blade-service
blade-swagger
blade-visual
src
main
java
org.springblade.modules.visual
controller
dynamic
log
mapper
service
tool
VisualApplication
resources
Dockerfile
pom.xml
pom.xml
blade-service-api
blade-visual-api
src
main
java
org.springblade.modules.visual
dto
entity
pom.xml
pom.xml
doc
.editorconfig
.gitignore
LICENSE
pom.xml
README.md

二、数据库配置

1. 创建一个空白数据库并导入如下sql (注意⚠：如果是最新版本，则无需关注下面的结构更新脚本)

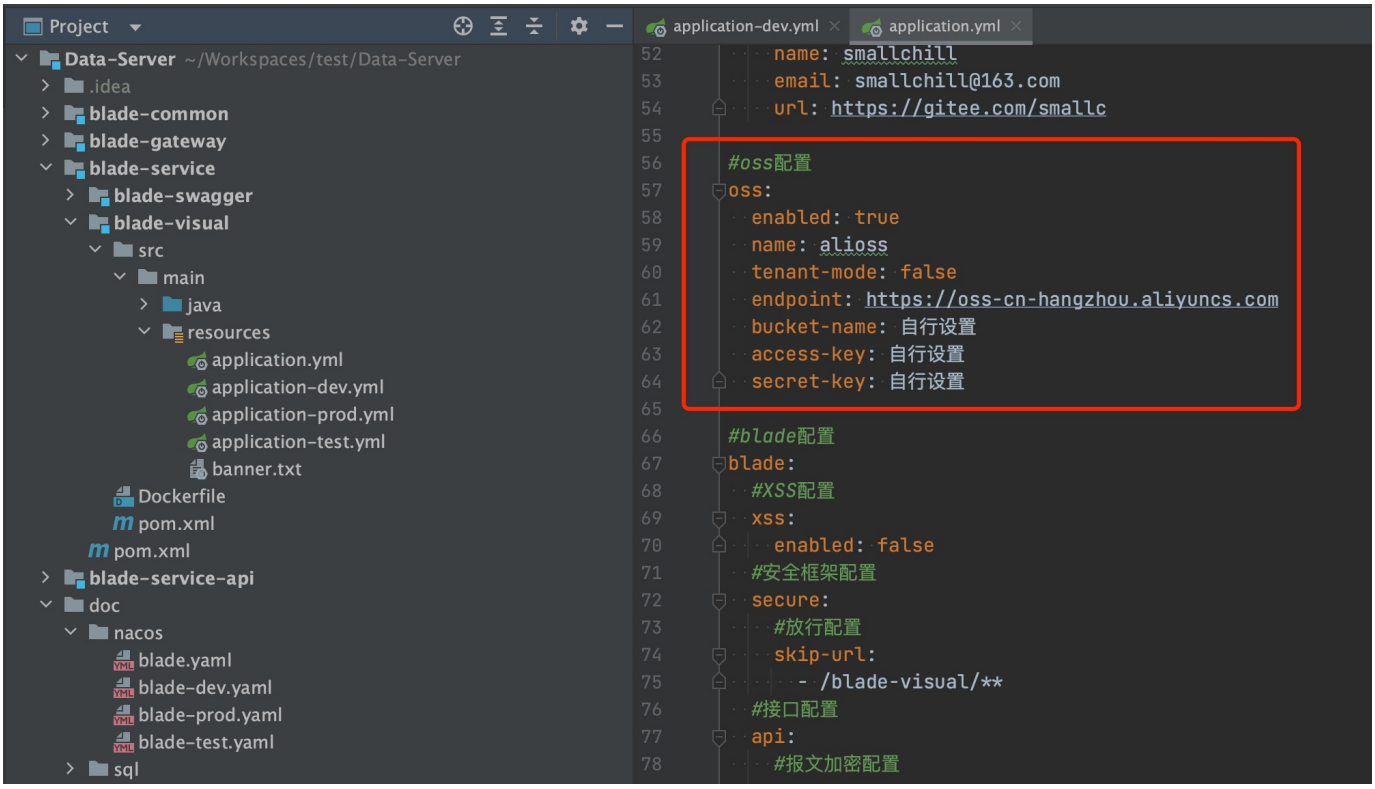


2. 到对应配置文件将数据库信息配置准确



三、对象存储配置

1. 框架支持minio、alioss、qiniu三种对象存储
2. 将对应配置到application.yml文件



3. 其中minio需要自行部署，如何启动部署请参考官网文档：http://docs.minio.org.cn/docs/

⚠minio版本使用注意：

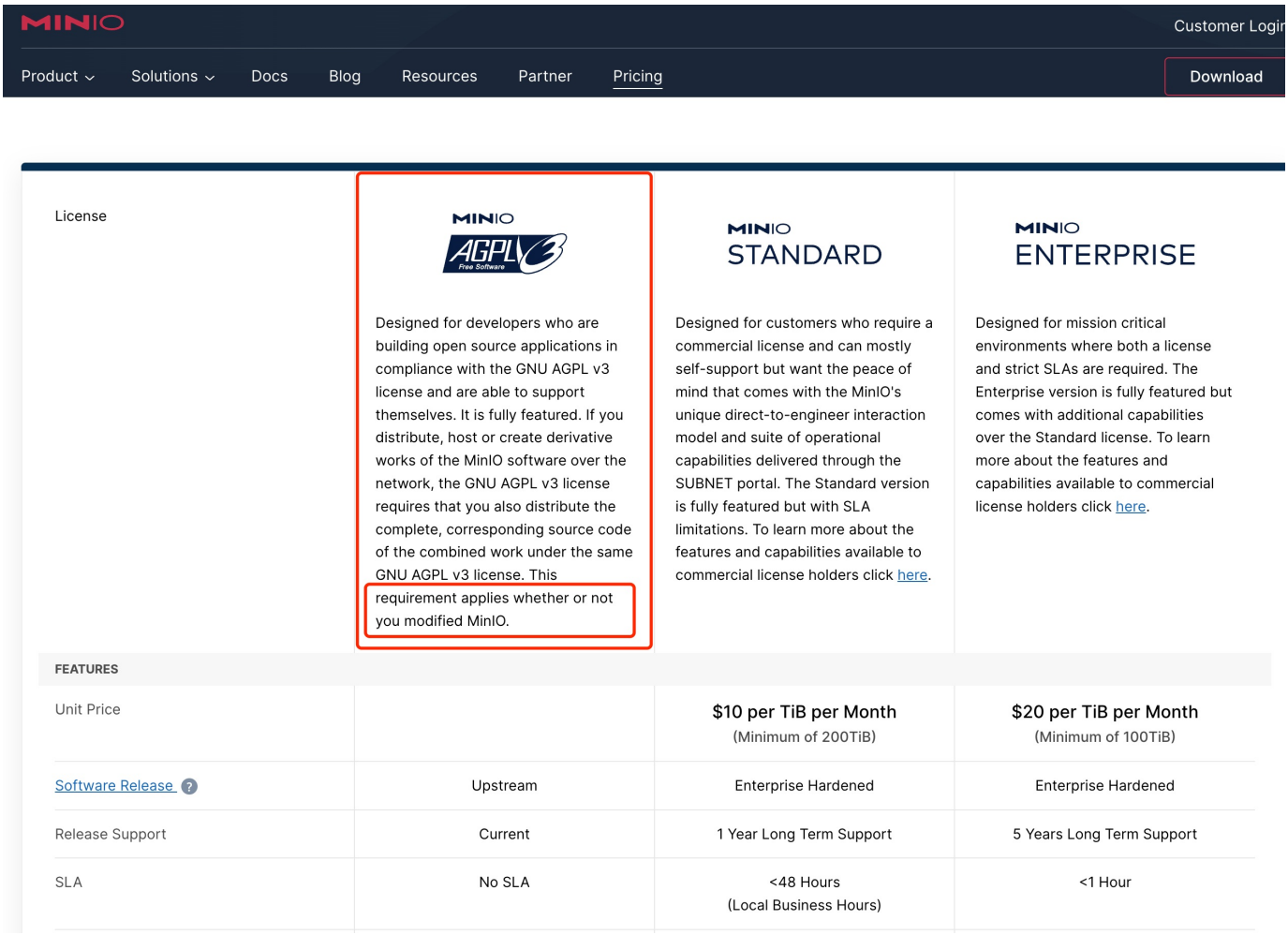
- 自从2021年minio开源协议从Apache2.0变更为AGPLv3后，新协议就不适合免费商用了
- AGPLv3规定，只要系统部署后接入minio服务，具有文件存储或读取行为，不论是否改造源码，不论发布在

何处，都需要将整个产品源码开源

- 所以如果使用minio开源版，请选择Apache2.0的最后一个版本：

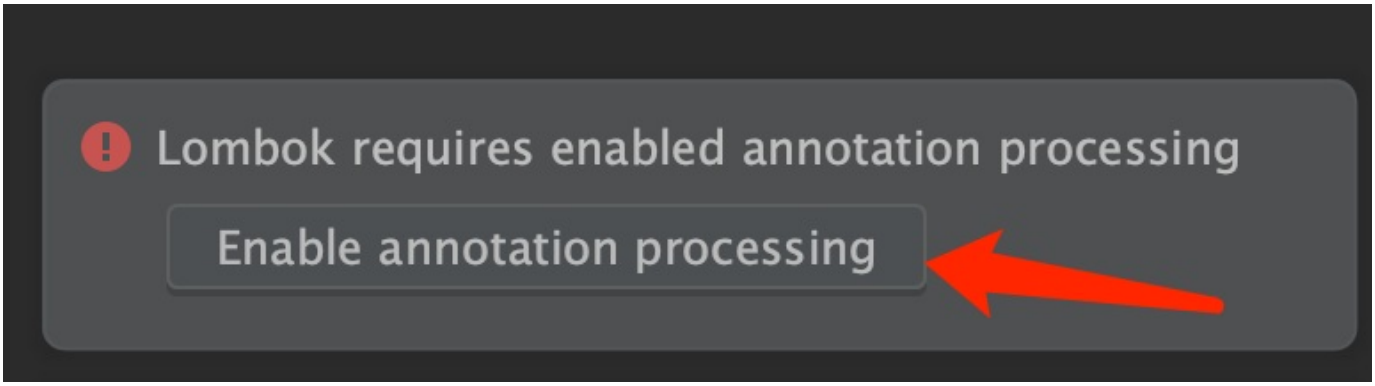
  `minio-RELEASE.2021-04-22T15-44-28Z`

- 具体官方说明请见下图



四、工程启动

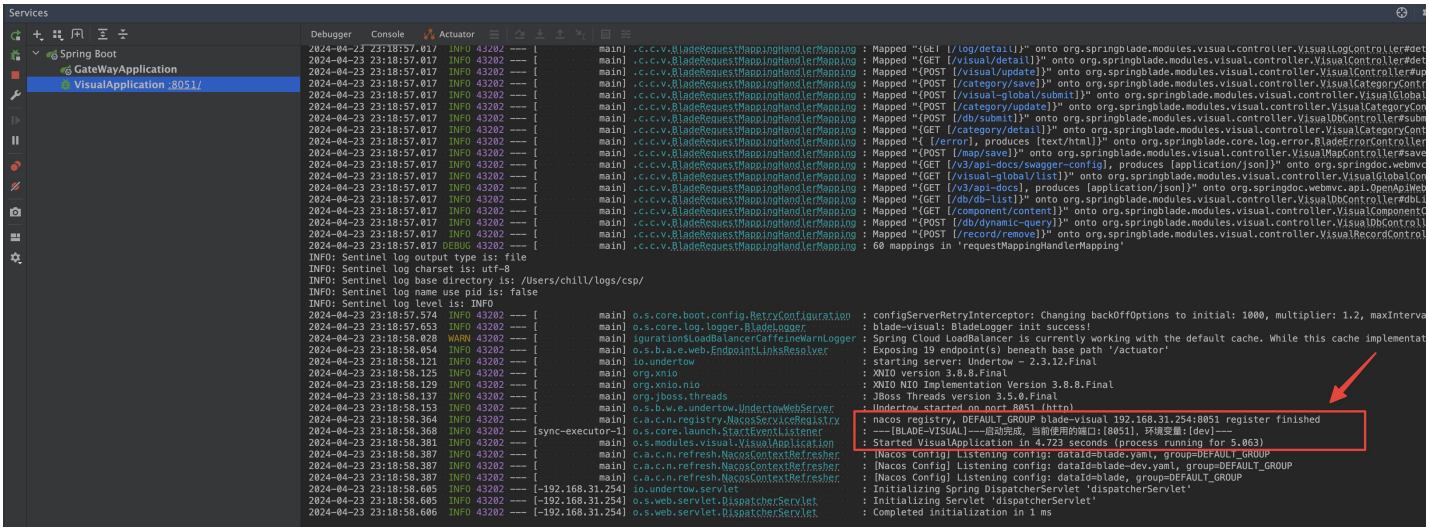
1. 一切准备完毕，我们打开VisualApplication，右键启动若出现Lombok的提示，点击Enable按钮即可



2. 先将nacos配置改为本地配置测试，若配置无误，我们就可以看到如下日志

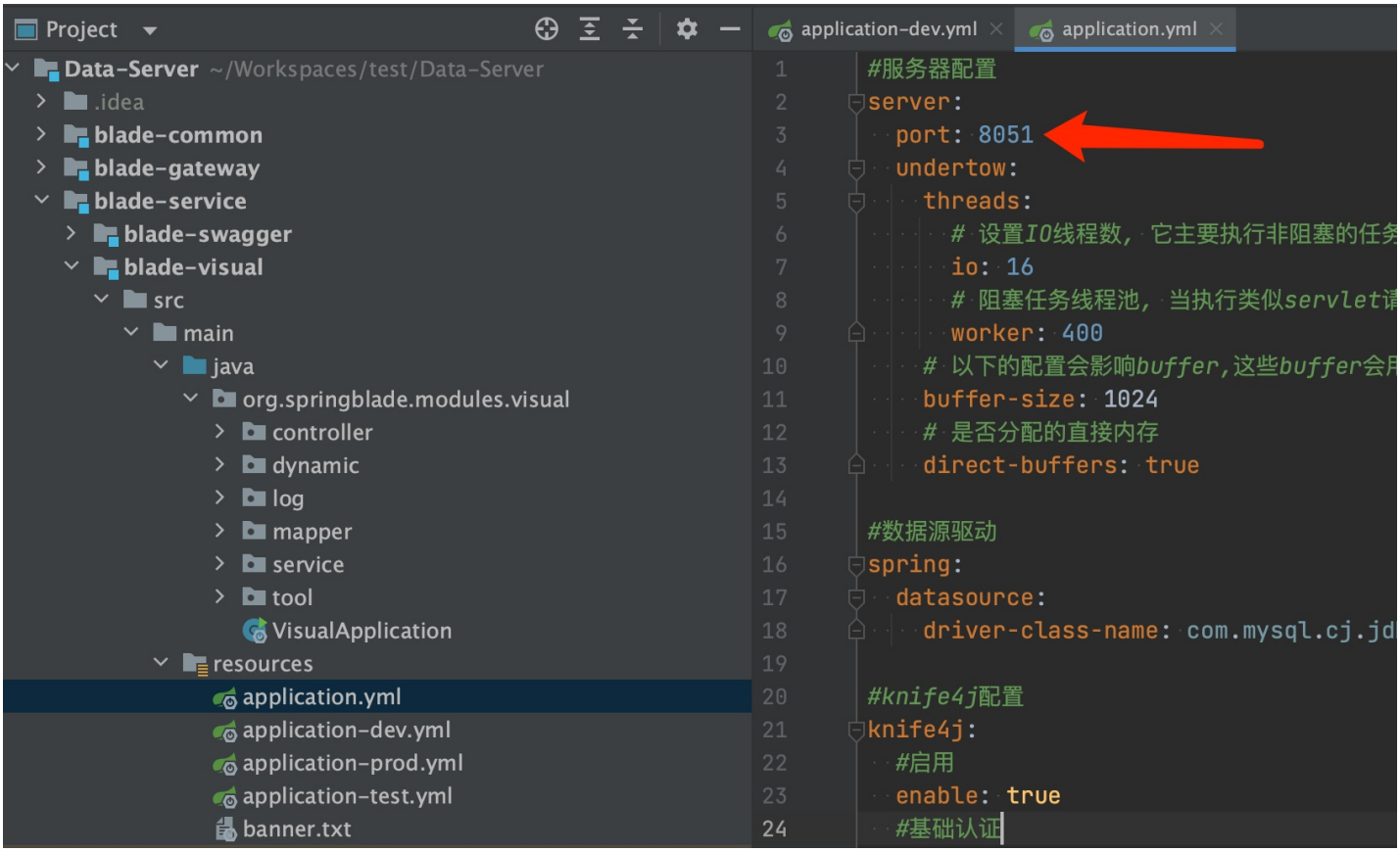
```
#数据源配置
```

```
spring:
  data:
    redis:
      ##redis 单机环境配置
      ##将docker脚本部署的redis服务映射为宿主机ip
      ##生产环境推荐使用阿里云高可用redis服务并设置密码
      host: 127.0.0.1
      port: 6379
      password:
      database: 0
      ssl:
        enabled: false
      ##redis 集群环境配置
      #cluster:
      #  nodes: 127.0.0.1:7001,127.0.0.1:7002,127.0.0.1:7003
      #  commandTimeout: 5000
  datasource:
    url: jdbc:mysql://localhost:3306/bladex_data?useSSL=false&useUnicode=true&c
haracterEncoding=utf-8&zeroDateTimeBehavior=convertToNull&transformedBitIsBoole
an=true&serverTimezone=GMT%2B8&nullCatalogMeansCurrent=true&allowPublicKeyRetri
eval=true
    username: root
    password: root
```

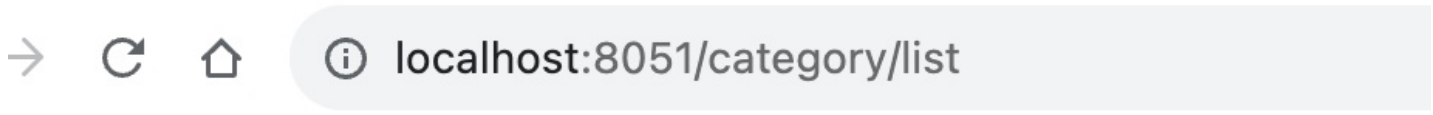


3. blade-visual的服务端口为8051，我们访问如下地址 ，测试是否有正确数据返回

http://localhost:8051/category/list



4. 若正确返回数据，则说明服务启动成功

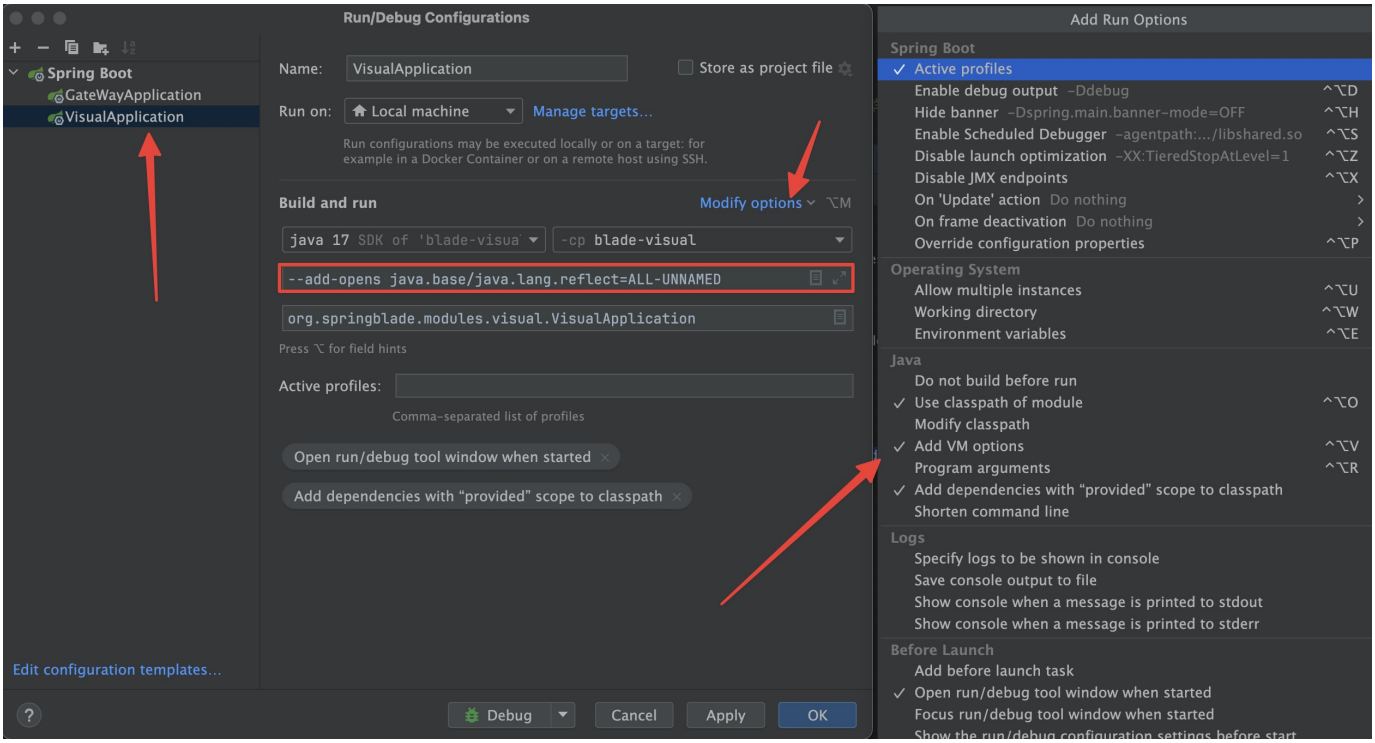



```
// 20230913210545
// http://localhost:8051/category/list

{
  "code": 200,
  "success": true,
  "data": [
    {
      "id": "1235454865361166338",
      "categoryKey": "精选模板",
      "categoryValue": "1",
      "isDeleted": 0
    },
    {
      "id": "1235455054851432448",
      "categoryKey": "组件库",
      "categoryValue": "2",
      "isDeleted": 0
    }
  ],
  "msg": "操作成功"
}
```

5. 若返回异常，是因为升级JDK17后，Java 模块化系统（Java Module System）的安全限制导致的针对反射等场景有可能会出现如下错误：

```
Caused by: java.lang.reflect.InaccessibleObjectException: Unable to make field
protected java.lang.reflect.InvocationHandler java.lang.reflect.Proxy.h accessi
ble: module java.base does not "opens java.lang.reflect" to unnamed module @23c
30a20
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(Acces
sibleObject.java:354)
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(Acces
sibleObject.java:297)
    at java.base/java.lang.reflect.Field.checkCanSetAccessible(Field.java:178)
    at java.base/java.lang.reflect.Field.setAccessible(Field.java:172)
    at org.apache.ibatis.reflection.invoker.GetFieldInvoker.invoke(GetFieldInvo
ker.java:38)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.getBeanProperty(BeanWra
pper.java:158)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.get(BeanWrapper.java:50
)
    at org.apache.ibatis.reflection.MetaObject.getValue(MetaObject.java:115)
    at org.apache.ibatis.reflection.MetaObject.metaObjectForProperty(MetaObject
.java:123)
    at org.apache.ibatis.reflection.wrapper.BaseWrapper.getChildValue(BaseWrapp
er.java:117)
    at org.apache.ibatis.reflection.wrapper.BeanWrapper.get(BeanWrapper.java:46
)
    at org.apache.ibatis.reflection.MetaObject.getValue(MetaObject.java:115)
    at org.springblade.core.mp.plugins.SqlLogInterceptor.intercept(SqlLogInterc
eptor.java:46)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy216.query(Unknown Source)
    at org.apache.ibatis.executor.SimpleExecutor.doQuery(SimpleExecutor.java:65
)
    at org.apache.ibatis.executor.BaseExecutor.queryFromDatabase(BaseExecutor.j
ava:336)
    at org.apache.ibatis.executor.BaseExecutor.query(BaseExecutor.java:158)
    at com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor.interc
ept(MybatisPlusInterceptor.java:81)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy215.query(Unknown Source)
    at org.apache.ibatis.session.defaults.DefaultSqlSession.selectList(DefaultS
qlSession.java:154)
    ... 115 common frames omitted
```

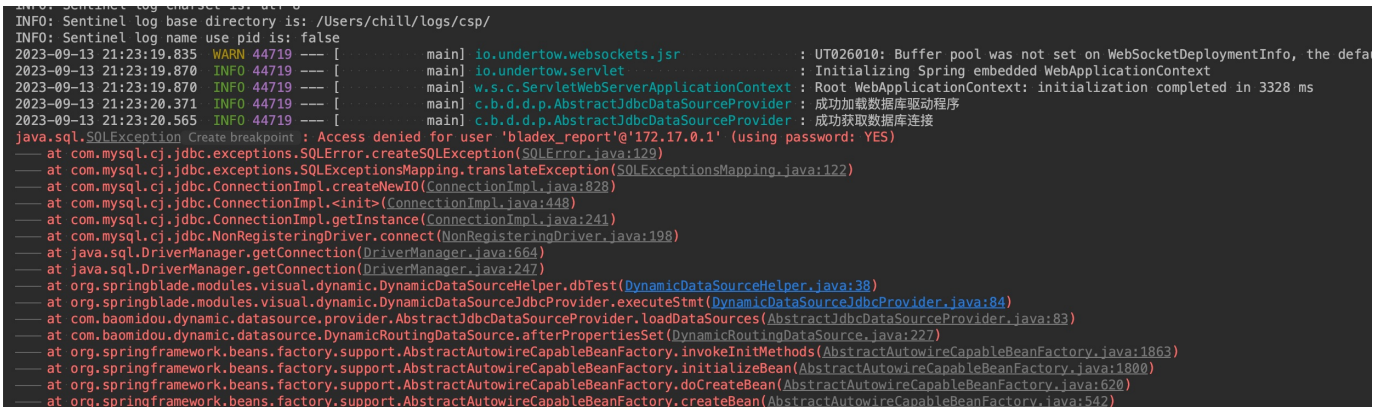
6. 遇到上述情况，在启动类增加命令 `--add-opens java.base/java.lang.reflect=ALL-UNNAMED` 即可

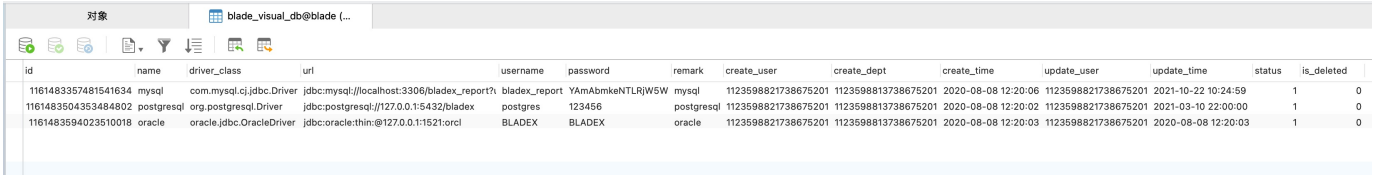


7. JAR 包启动时也需要加入此配置，具体命令如下

```
java --add-opens java.base/java.lang.reflect=ALL-UNNAMED -jar your-application.jar
```

8. 若启动期间出现数据库加载失败，但是最后又启动成功，不必担心，这是因为大屏后端有动态数据源的功能，会实时加载配置的多个数据源，若连接失败，会抛出异常并加载下一个链接，不会影响整体使用。若不想出现这些提示，将blade_visual_db表的默认数据清空即可

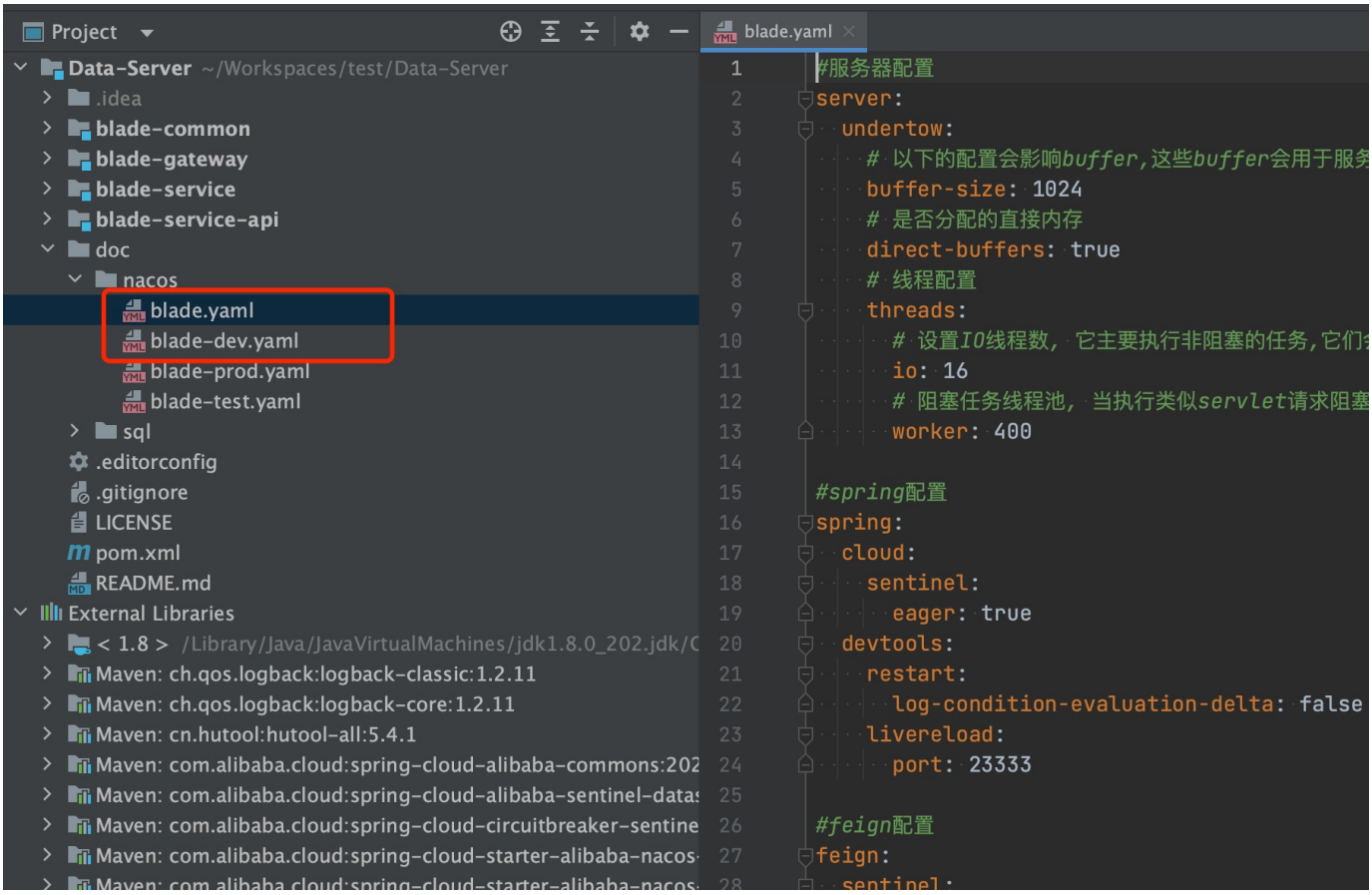


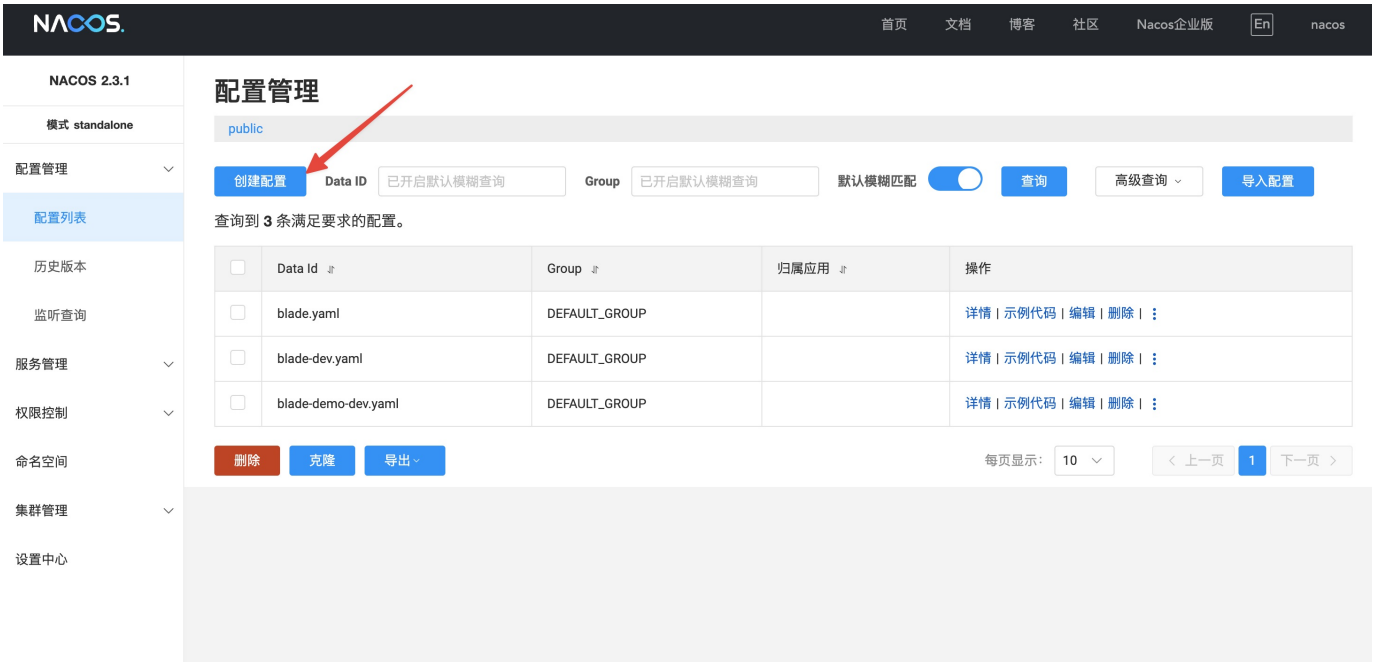


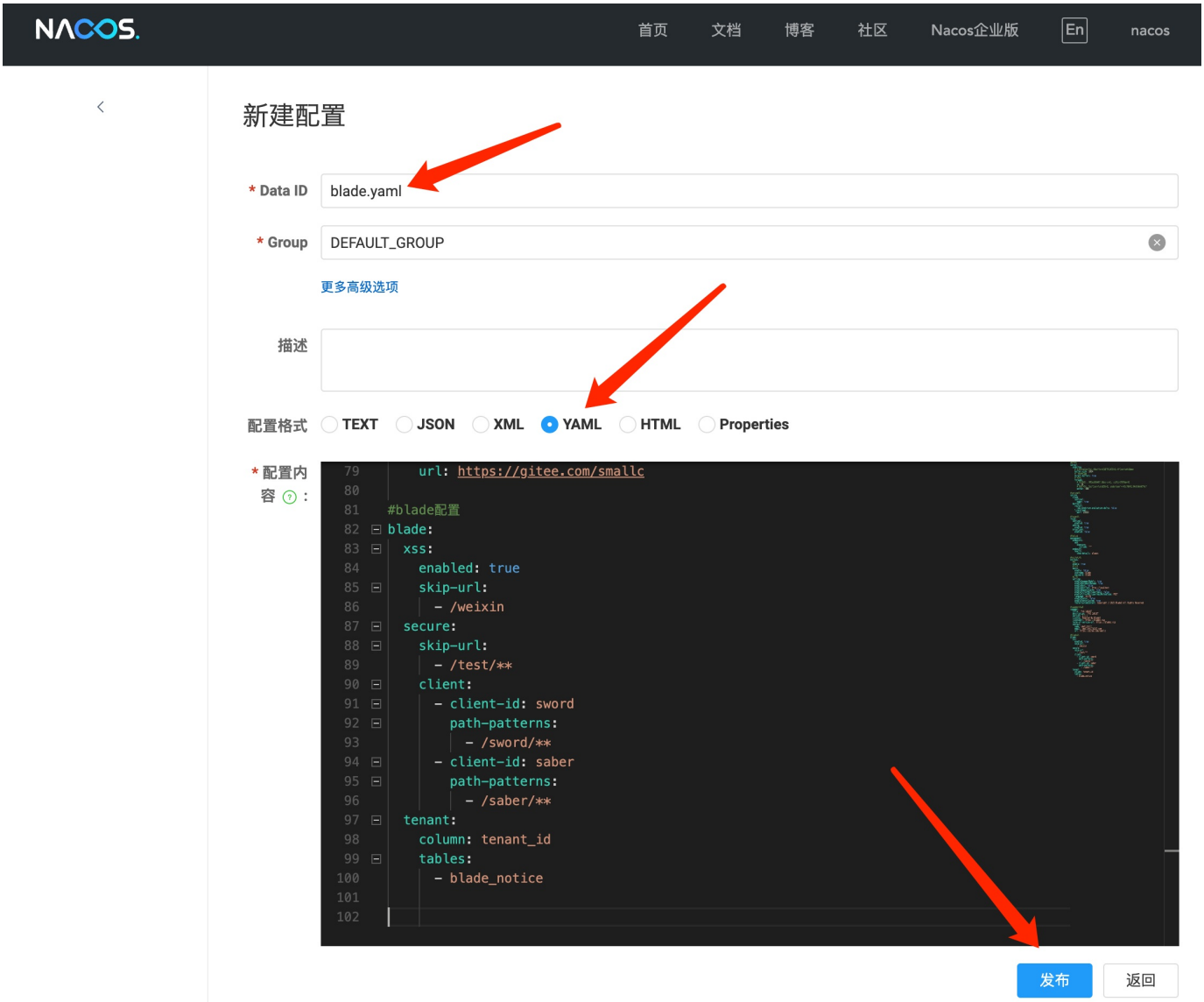
五、对接Cloud

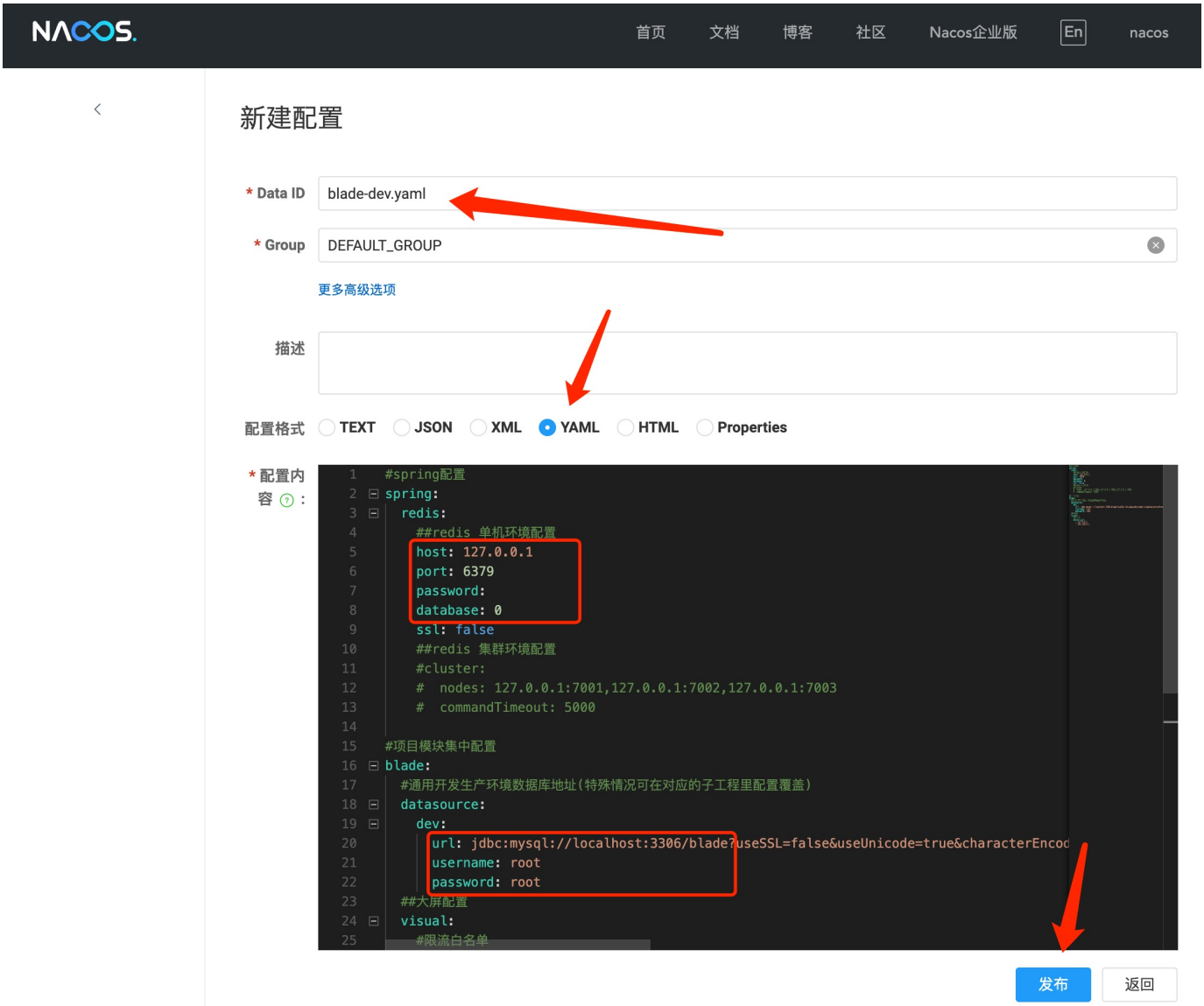
1. 由上一章节启动好Nacos为前提，我们现在将配置文件拷贝到nacos

2. 依次创建blade.yaml与blade-dev.yaml两个配置文件



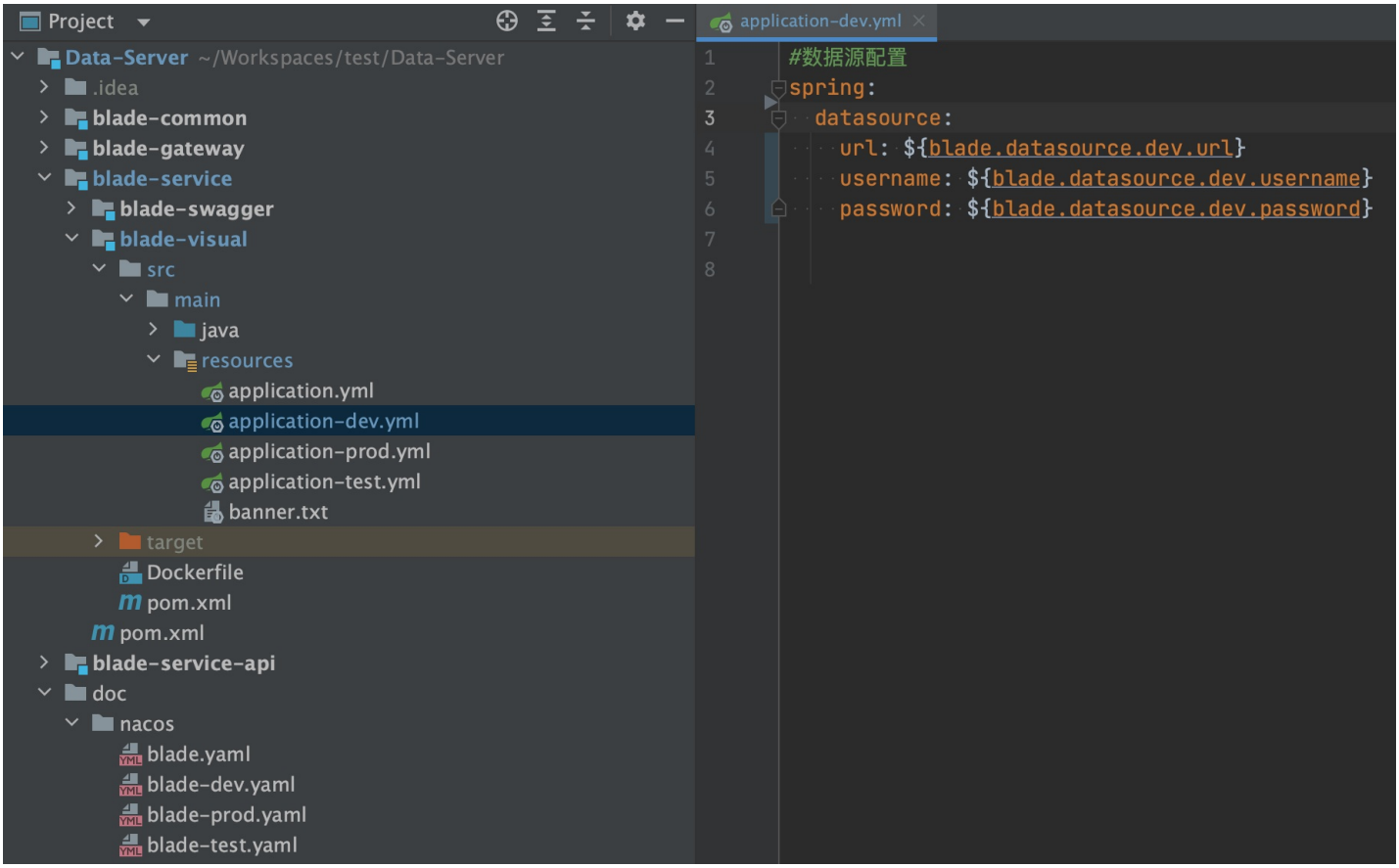



NACOS.
首页
文档
博客
社区
Nacos企业版
En
nacos
新建配置
* Data ID
blade.yaml
* Group
DEFAULT_GROUP
更多高级选项
描述
配置格式
TEXT
JSON
XML
YAML
HTML
Properties
* 配置内容：
url: https://gitee.com/smallc
#blade配置
blade:
xss:
enabled: true
skip-url:
- /weixin
secure:
skip-url:
- /test/**
client:
- client-id: sword
path-patterns:
- /sword/**
- client-id: saber
path-patterns:
- /saber/**
tenant:
column: tenant_id
tables:
- blade_notice
发布
返回

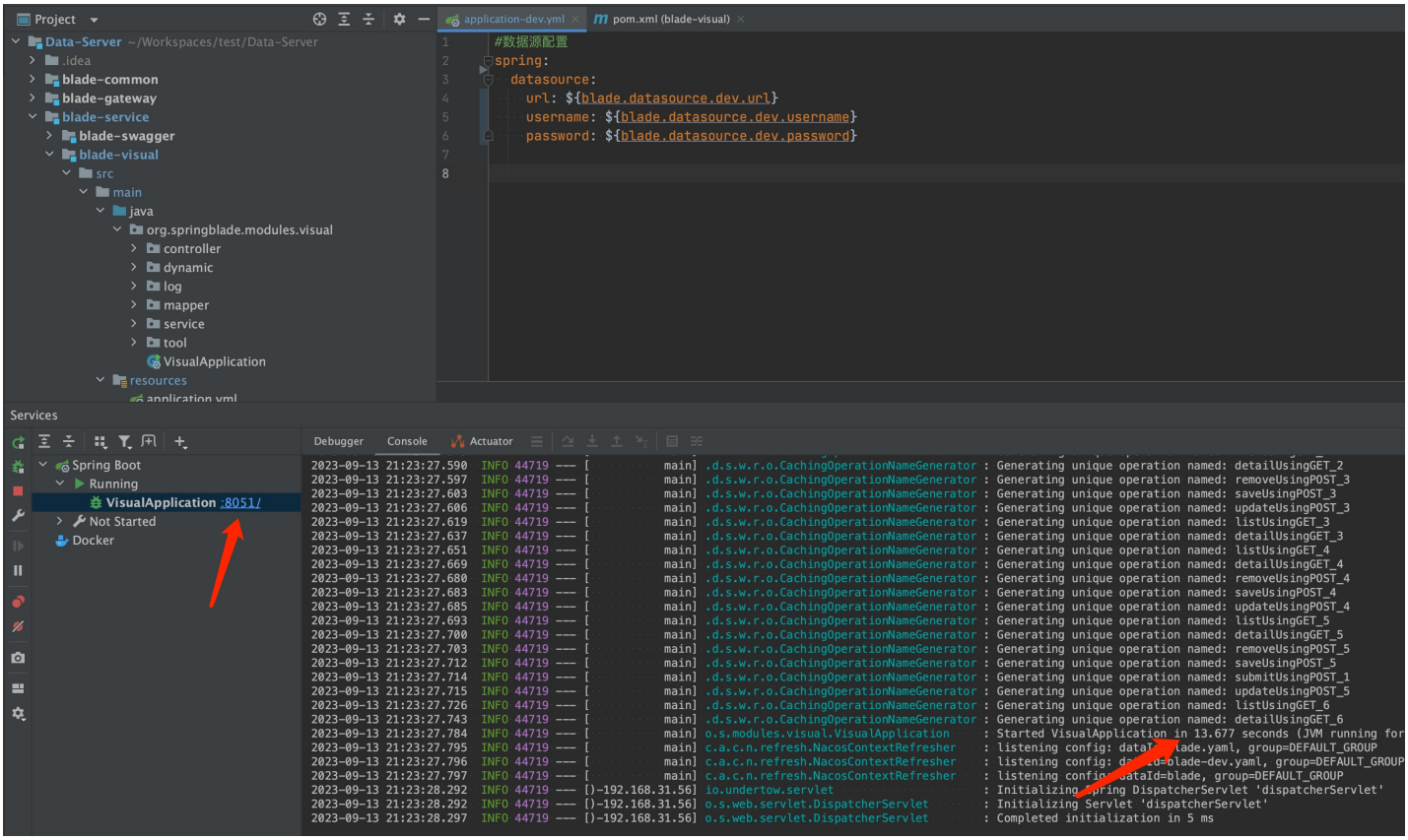


3. 修改blade-visual服务的配置，改为从nacos读取

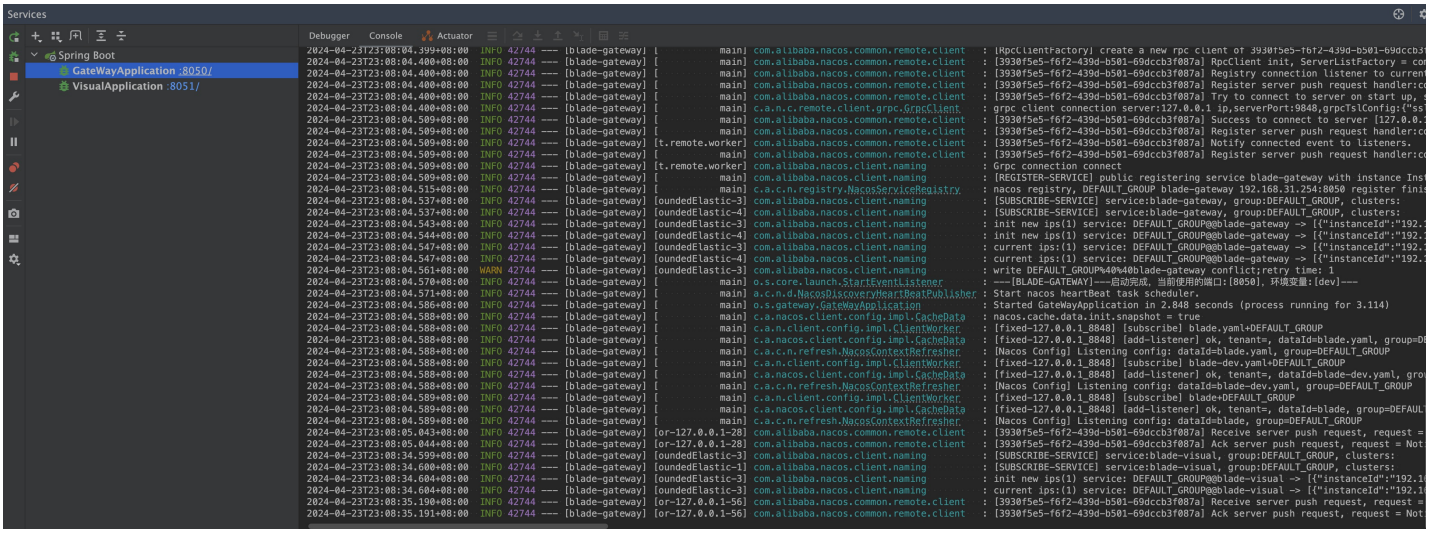
```
spring:
  datasource:
    url: ${blade.datasource.dev.url}
    username: ${blade.datasource.dev.username}
    password: ${blade.datasource.dev.password}
```



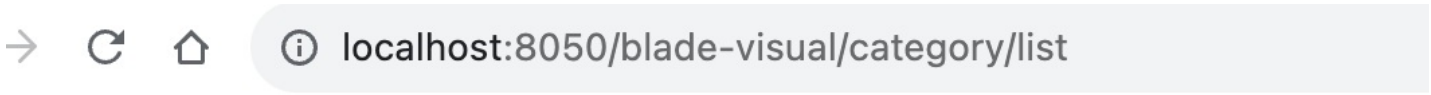
4. blade-visual启动依旧成功



5. 接着启动 blade-gateway，等待启动成功



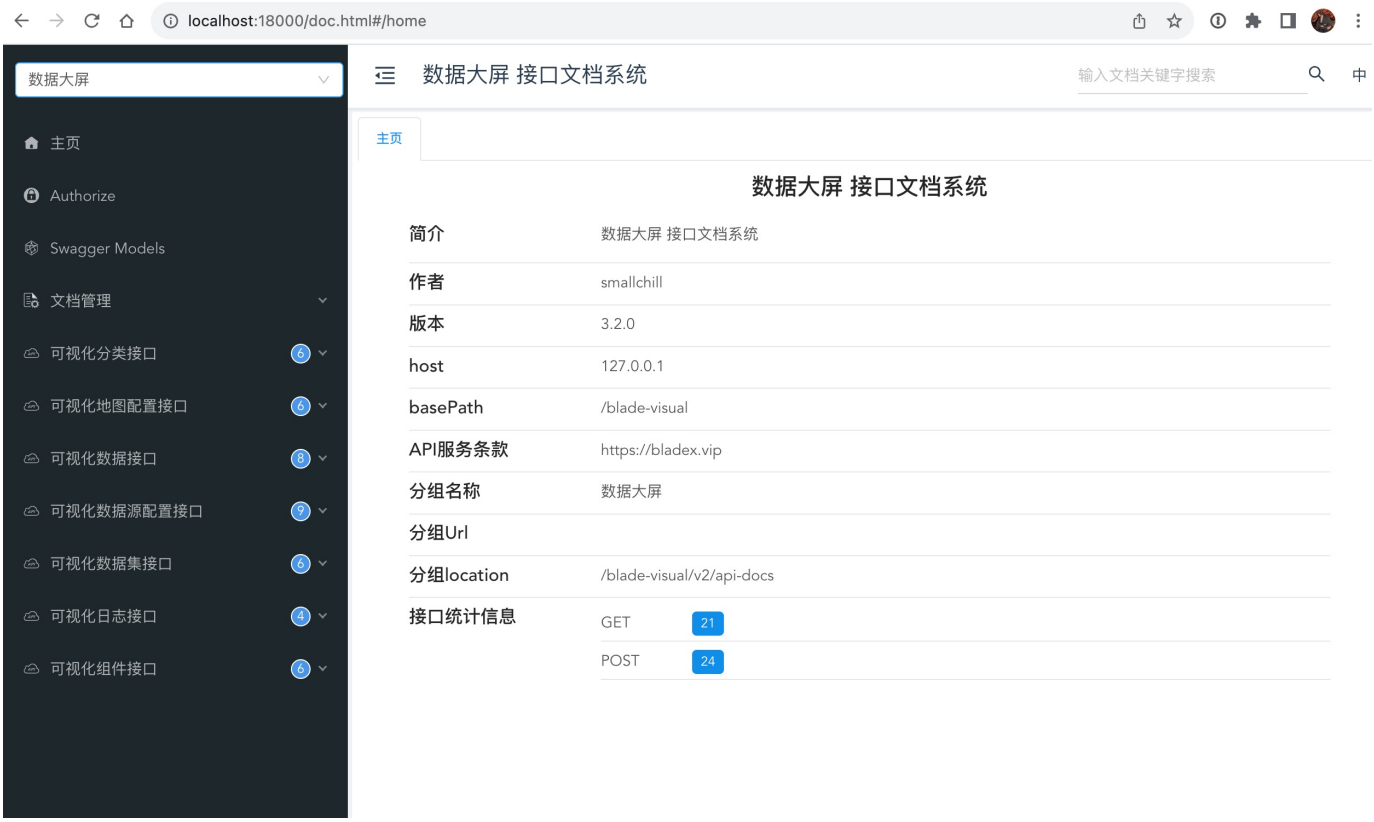
6. 由于加入了网关的服务，我们现在可以通过网关来调用服务，测试地址改为 http://localhost:8050/blade-visual/category/list ，这里 `8050` 端口便是网关的端口， `blade-visual` 便是服务的名称，网关通过服务名去nacos调用真是的服务地址。现在访问测试地址，可以看到数据返回成功，整个服务也启动完毕



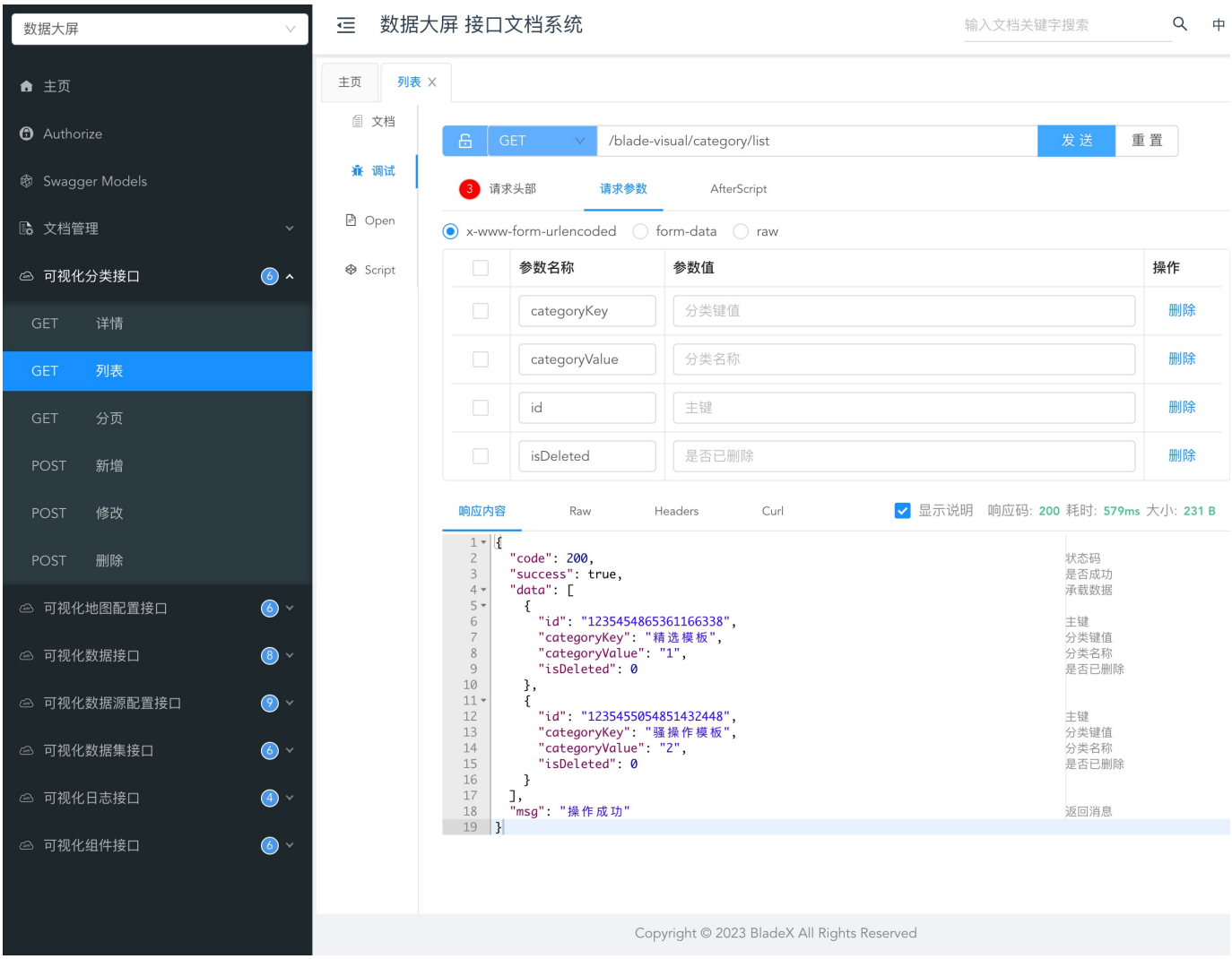

```
// 20230913212908
// http://localhost:8050/blade-visual/category/list

{
  "code": 200,
  "success": true,
  "data": [
    {
      "id": "1235454865361166338",
      "categoryKey": "精选模板",
      "categoryValue": "1",
      "isDeleted": 0
    },
    {
      "id": "1235455054851432448",
      "categoryKey": "骚操作模板",
      "categoryValue": "2",
      "isDeleted": 0
    }
  ],
  "msg": "操作成功"
}
```

六、接口文档

1. 启动blade-swagger服务后，便可以使用接口文档，访问接口文档地址：http://localhost:8050/doc.html



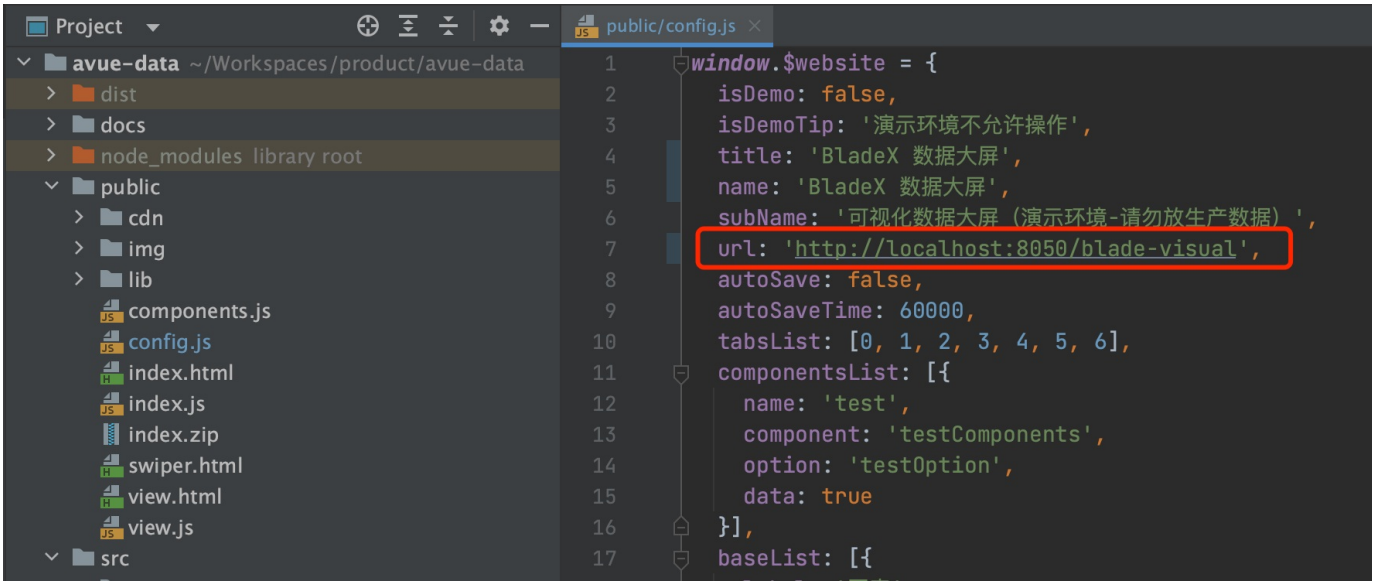
2. 找一个接口点击查看数据正确返回



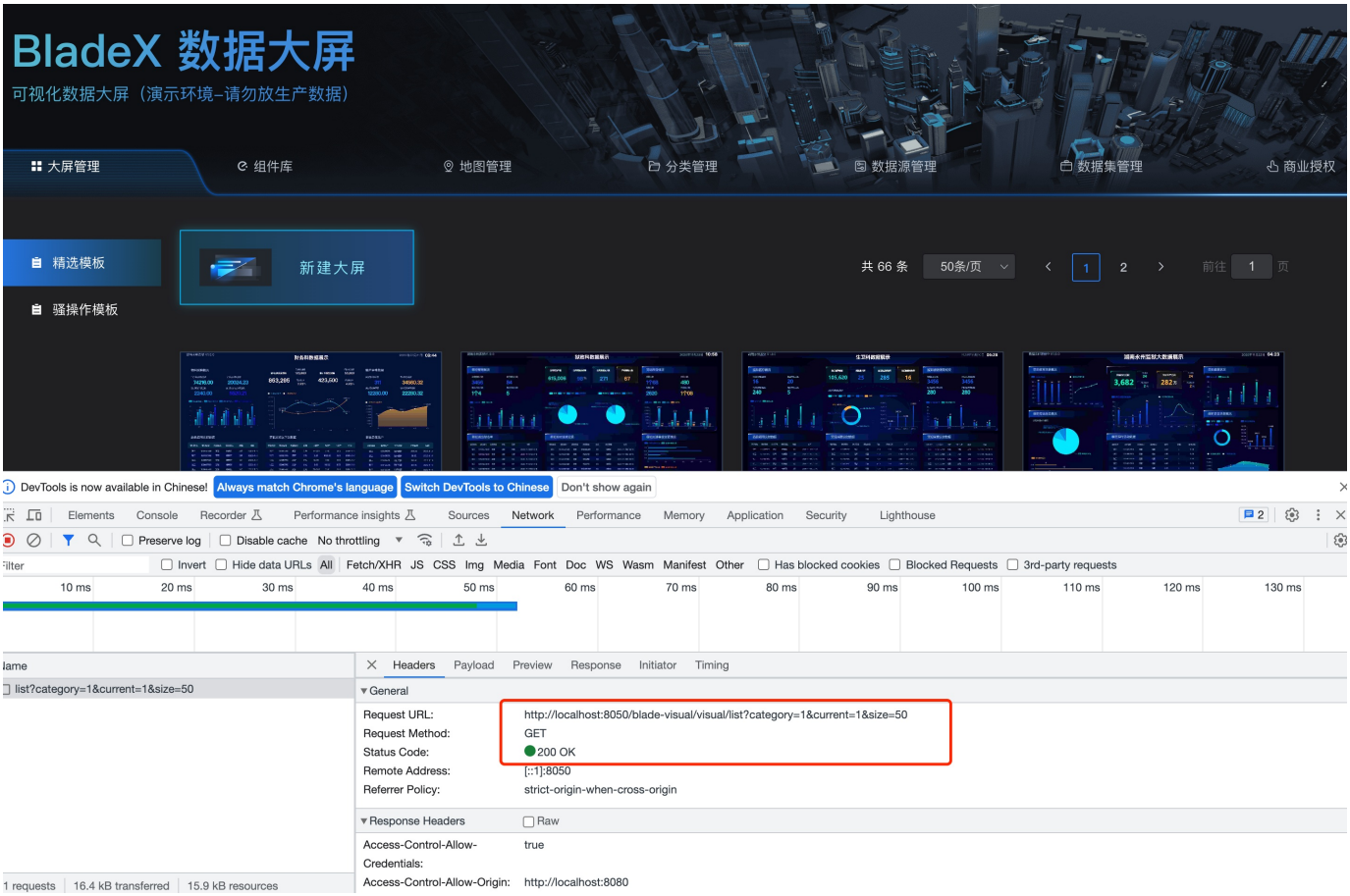
3. 至此后端部署全流程结束

前端配置地址

1. 前往前端的config.js文件，将后端接口地址改为：http://localhost:8050/blade-visual



2. 刷新前端大屏可以看到数据获取成功



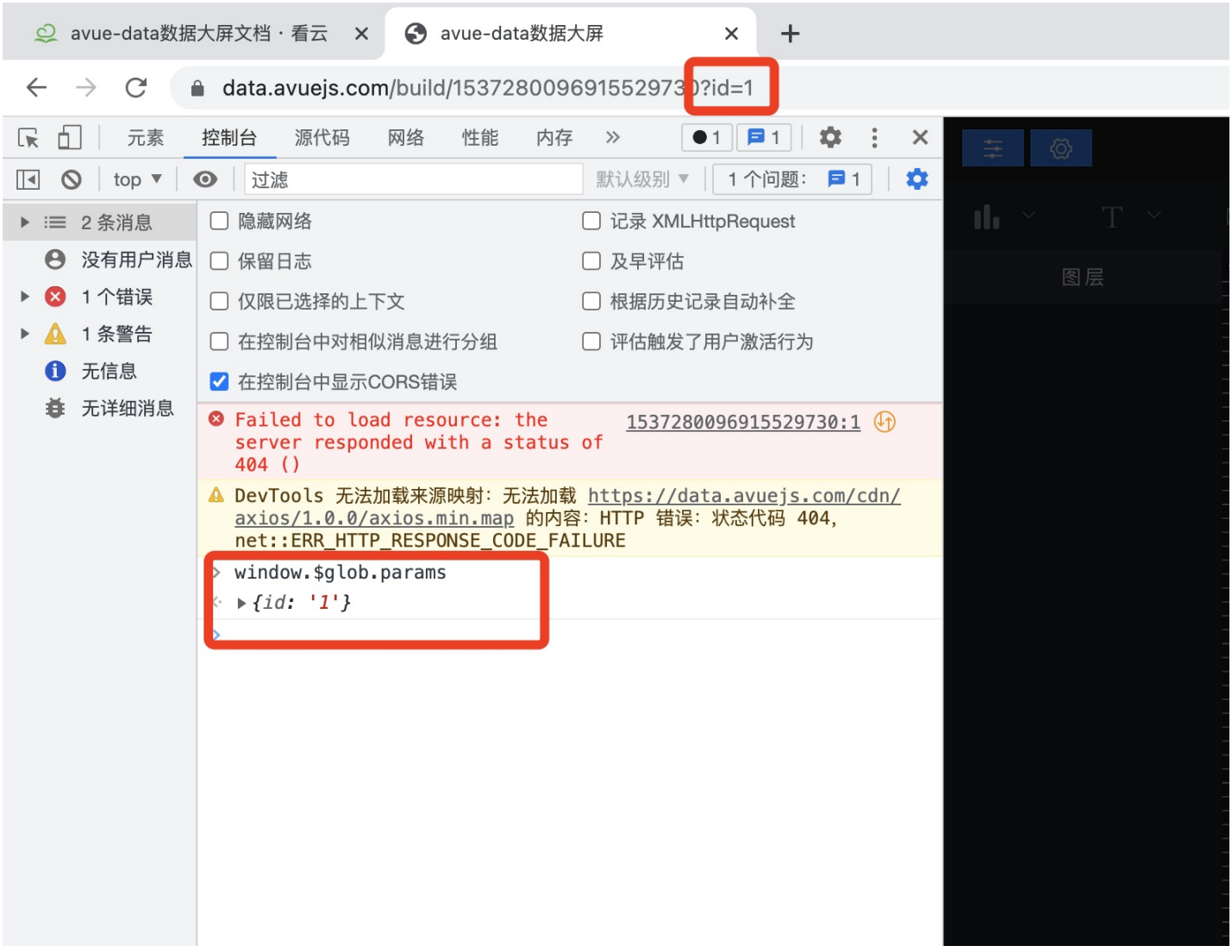
全局变量

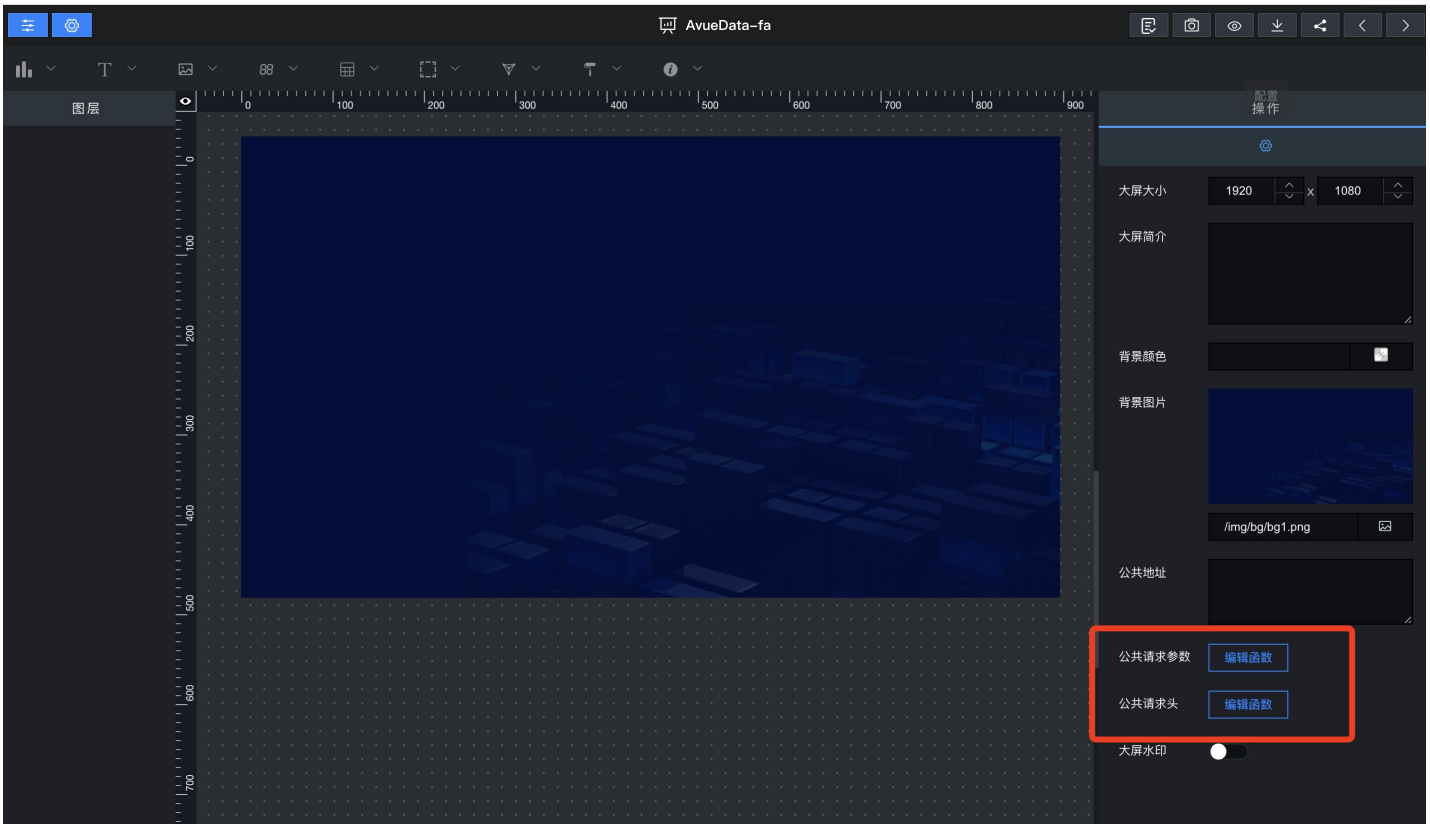
> 全局变量，它可以作用于axios发送数据中

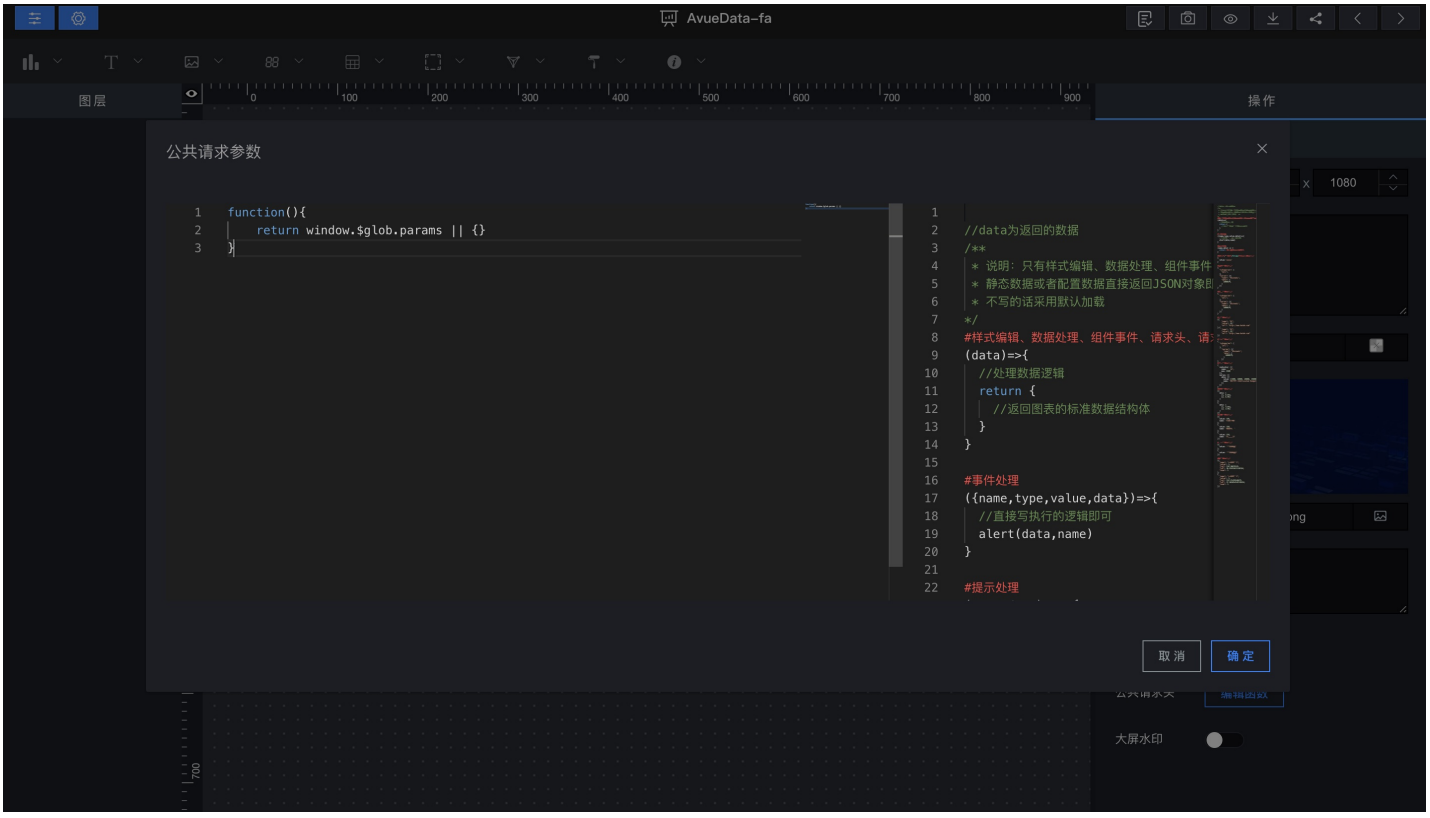
作用域

- header头部
- body发送体
- url参数后面

他可以手动配置，也可以自动获取window.$glob全局变量下的参数




AvueData-fa
图层
配置
操作
大屏大小
1920
1080
大屏简介
背景颜色
背景图片
/img/bg/bg1.png
公共地址
公共请求参数
编辑函数
公共请求头
编辑函数
大屏水印

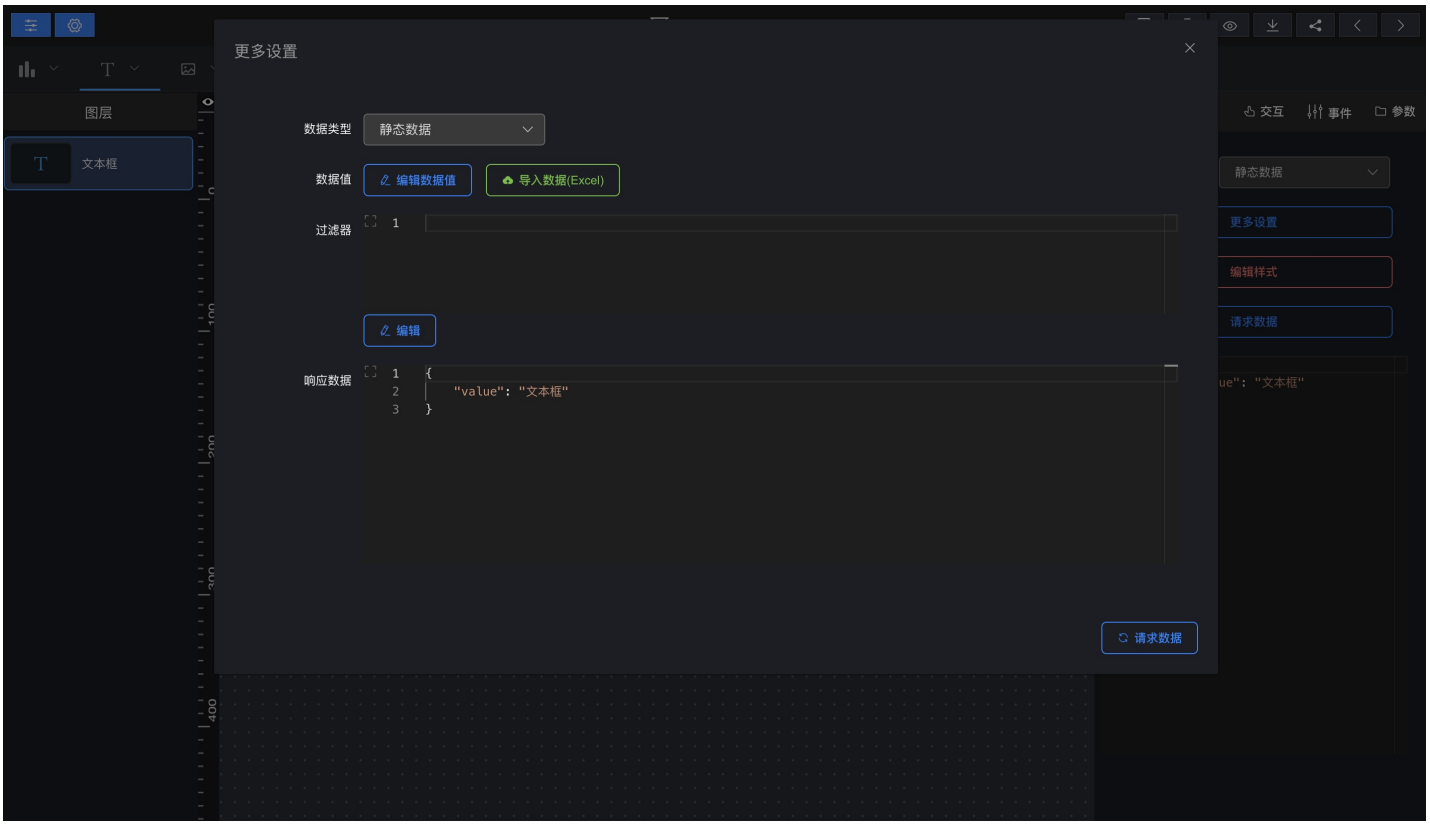

AvueData-fa
图层
操作
公共请求参数
function(){
return window.$glob.params || {}
}
//data为返回的数据
/**
* 说明：只有样式编辑、数据处理、组件事件
* 静态数据或者配置数据直接返回JSON对象即
* 不写的话采用默认加载
*/
#样式编辑、数据处理、组件事件、请求头、请
(data)=>{
//处理数据逻辑
return {
//返回图表的标准数据结构体
}
}
#事件处理
({name,type,value,data})=>{
//直接写执行的逻辑即可
alert(data,name)
}
#提示处理
取 消
确 定
大屏水印

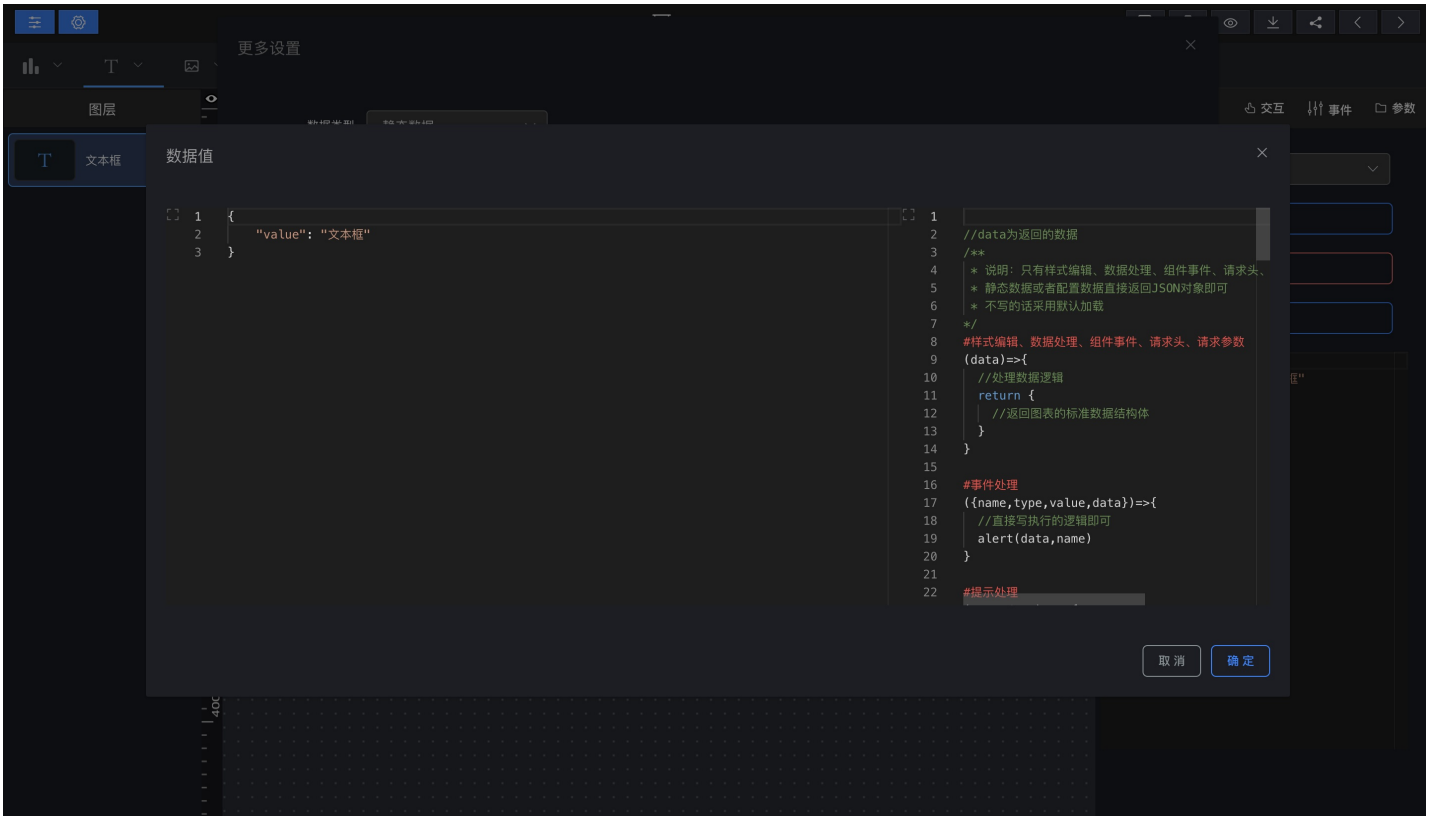
组件数据

> 组件请求回数据、完后返回组件可以识别的标准格式即可点击我跳转参考例子

一、静态数据

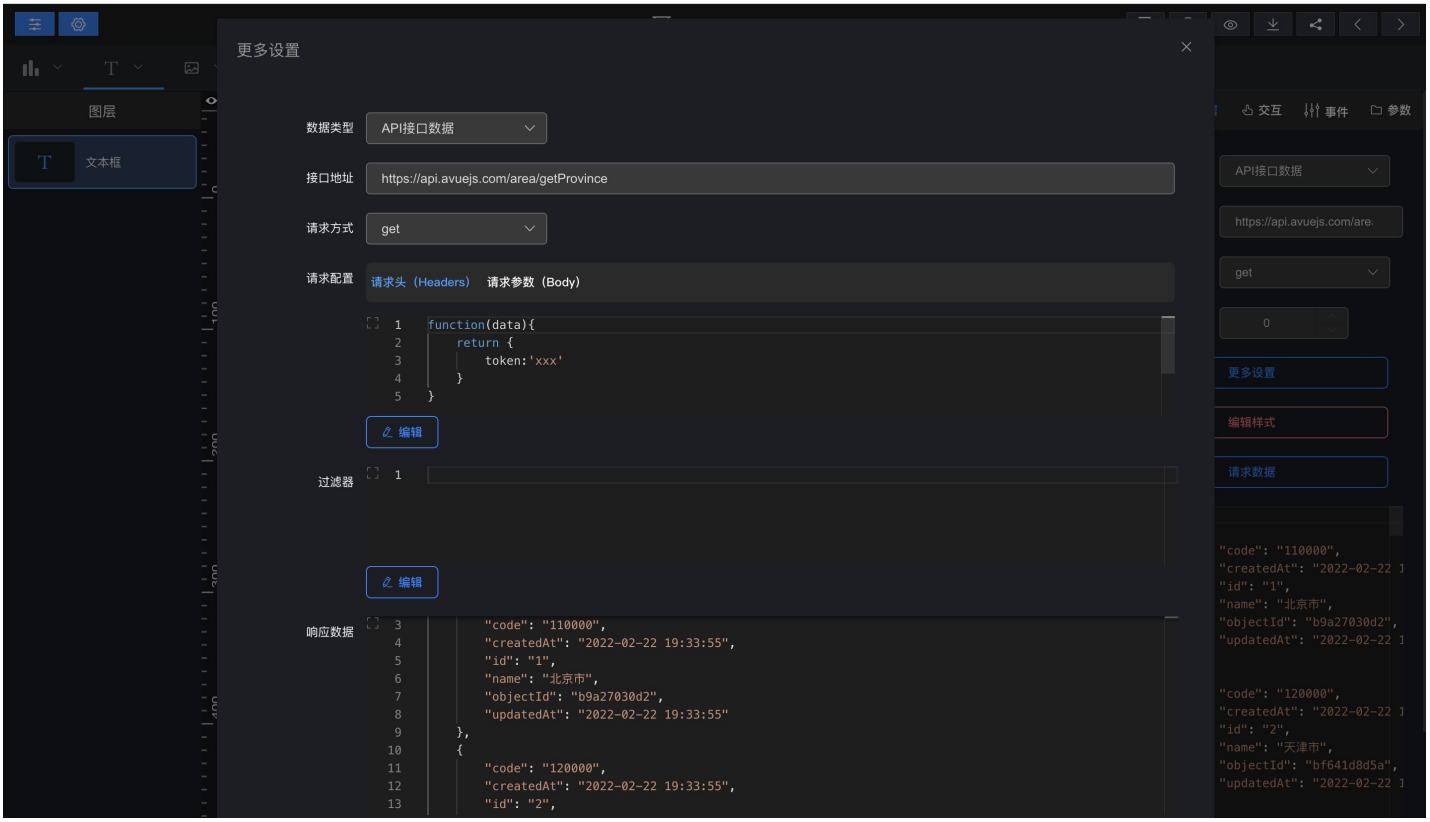
直接配置成组件对应的数据格式





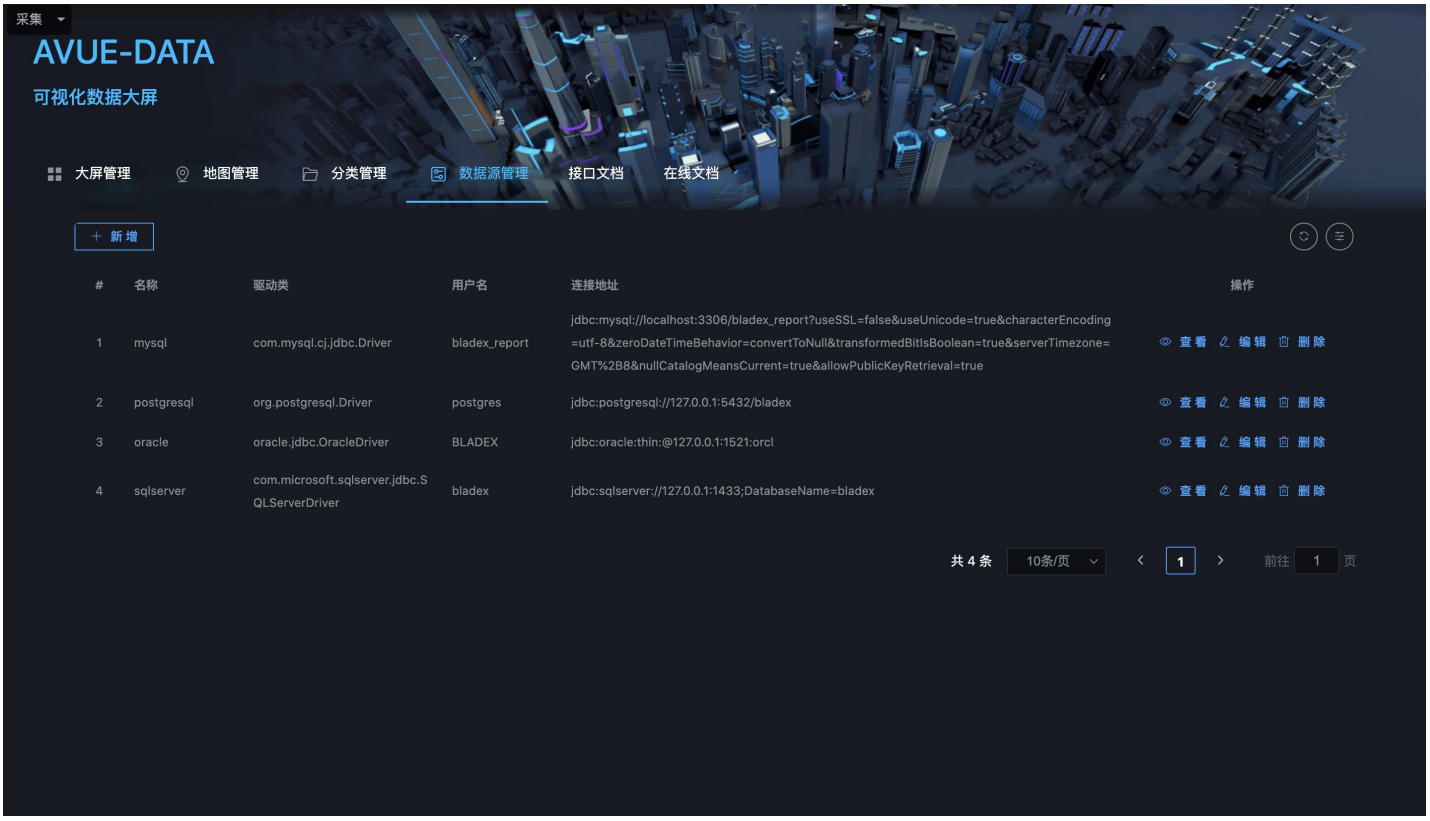
二、API接口

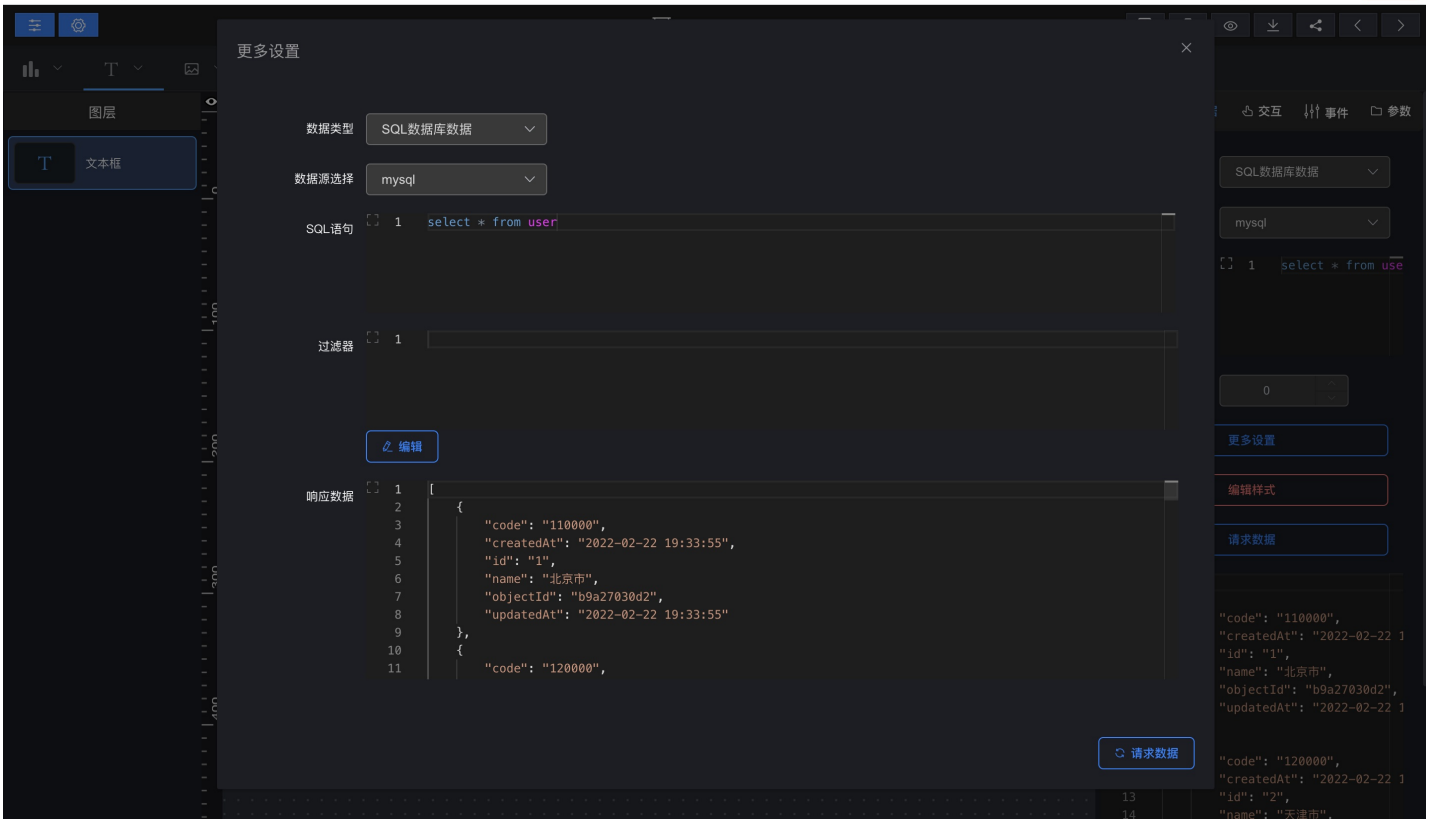
填写API地址，配置相关请求参数，返回的数据处理成组件对应的数据格式



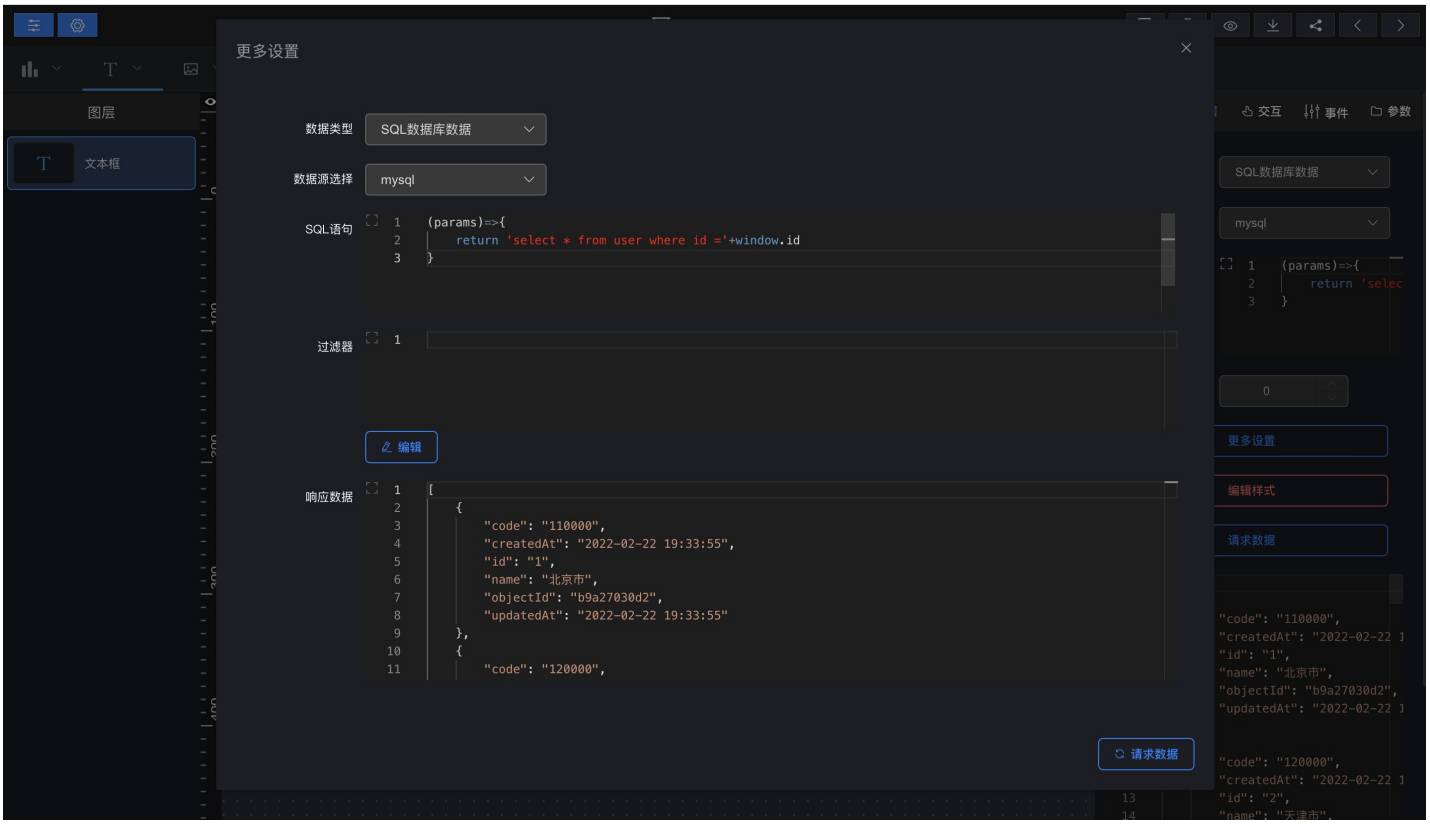
三、SQL数据源

配数据源，在数据类型中选择对应的数据源，写SQL语句去查询数据



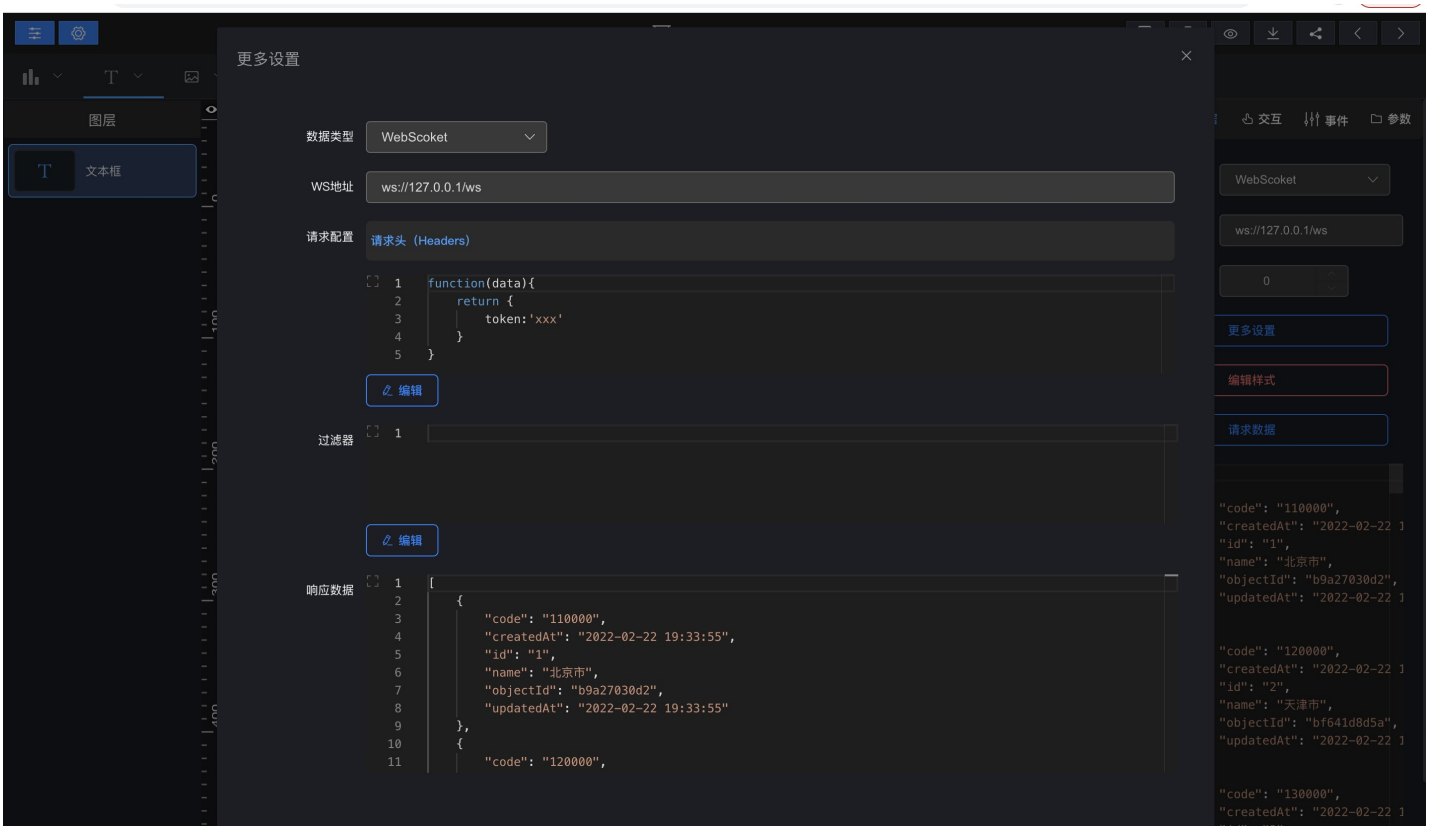


这里的SQl是可以配置成函数形式，这样就可以取各种参数和做一些处理逻辑,返回最终的查询语句



四、Websoket长链接

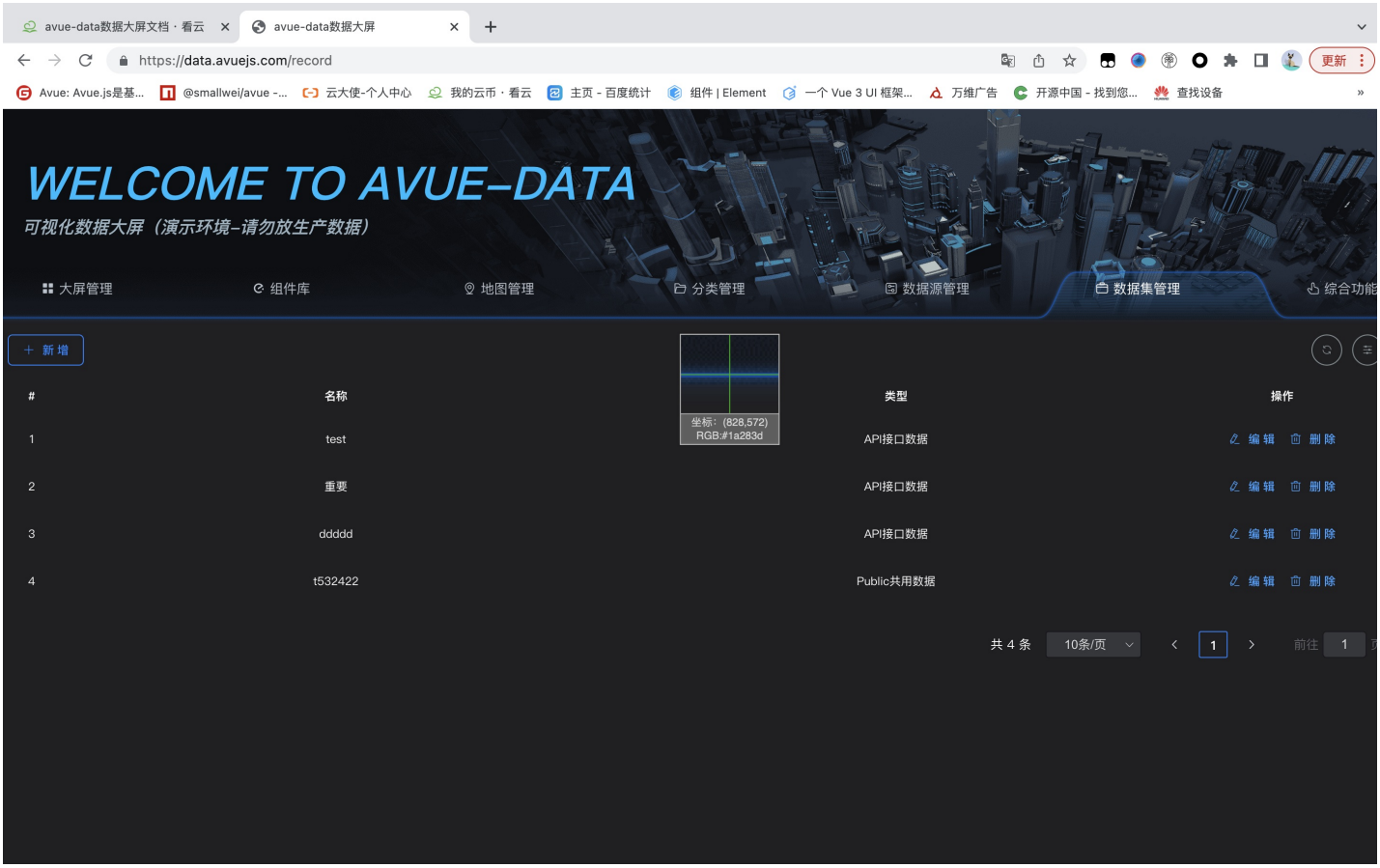
配置长链接地址返回数据，用法和API类似

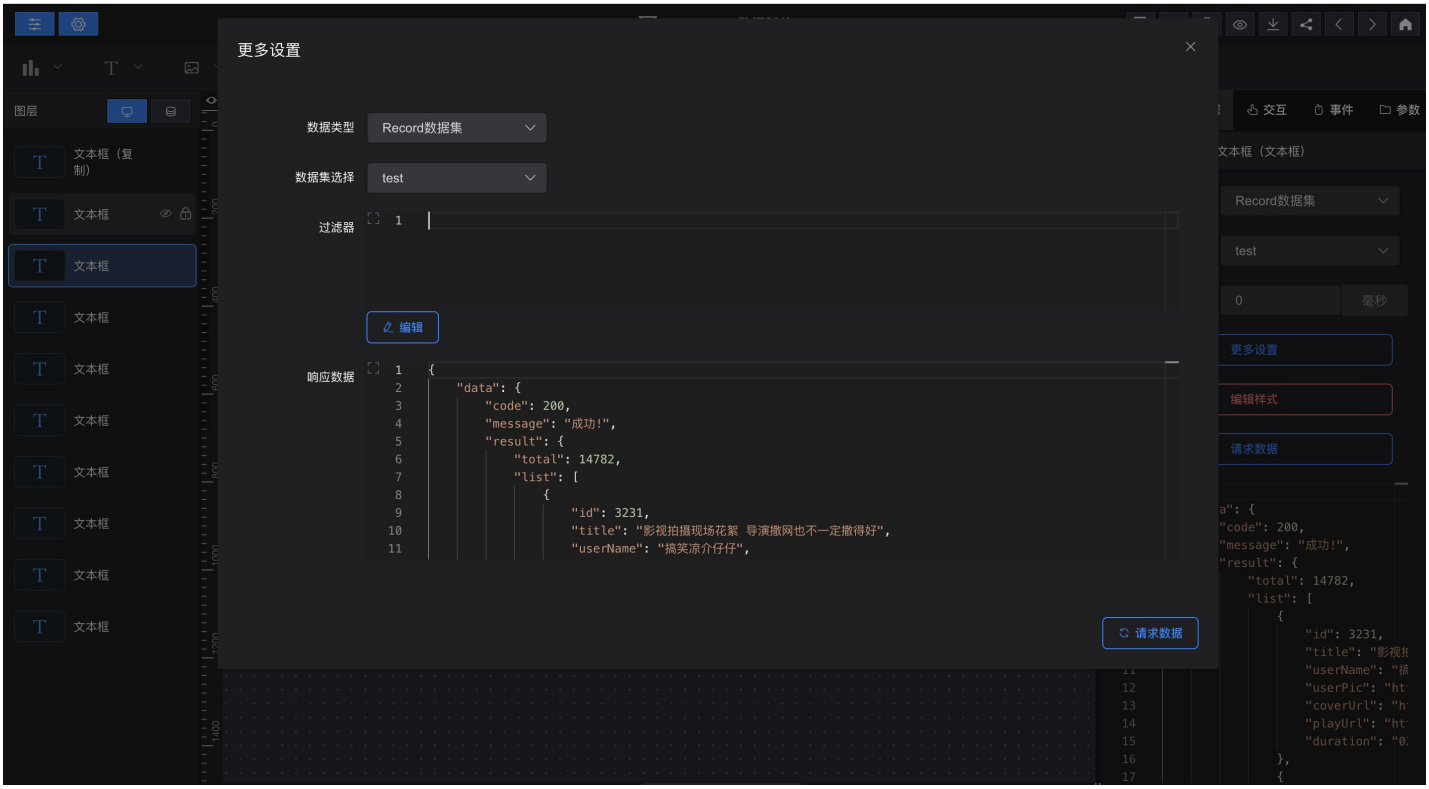


五、Record数据集

数据集就是以上数据的公用集合，比如多个组件用用一个数据接口，就不要重复配置，配置数据集，直接给对应

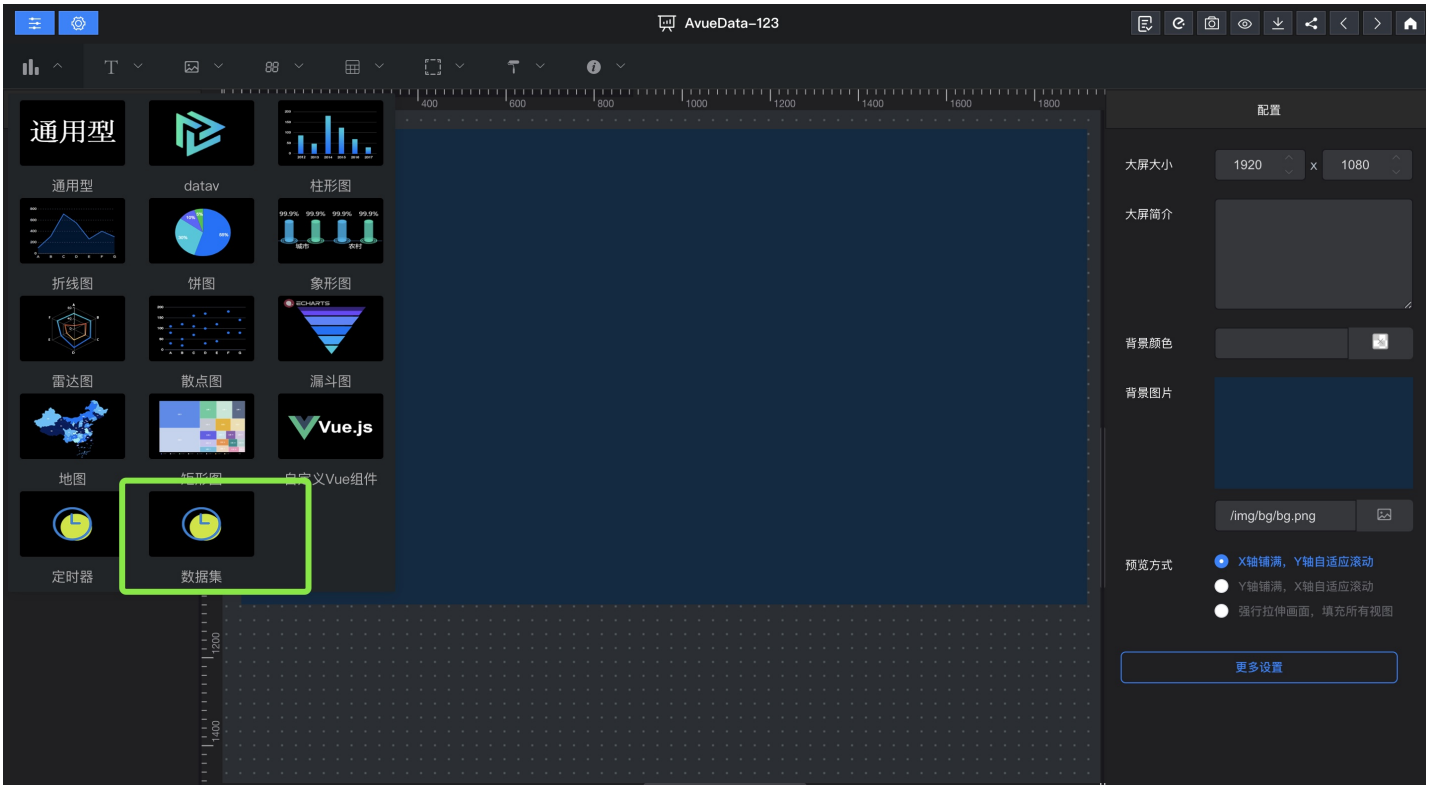
的组件引用即可，它每次都会请求新的接口

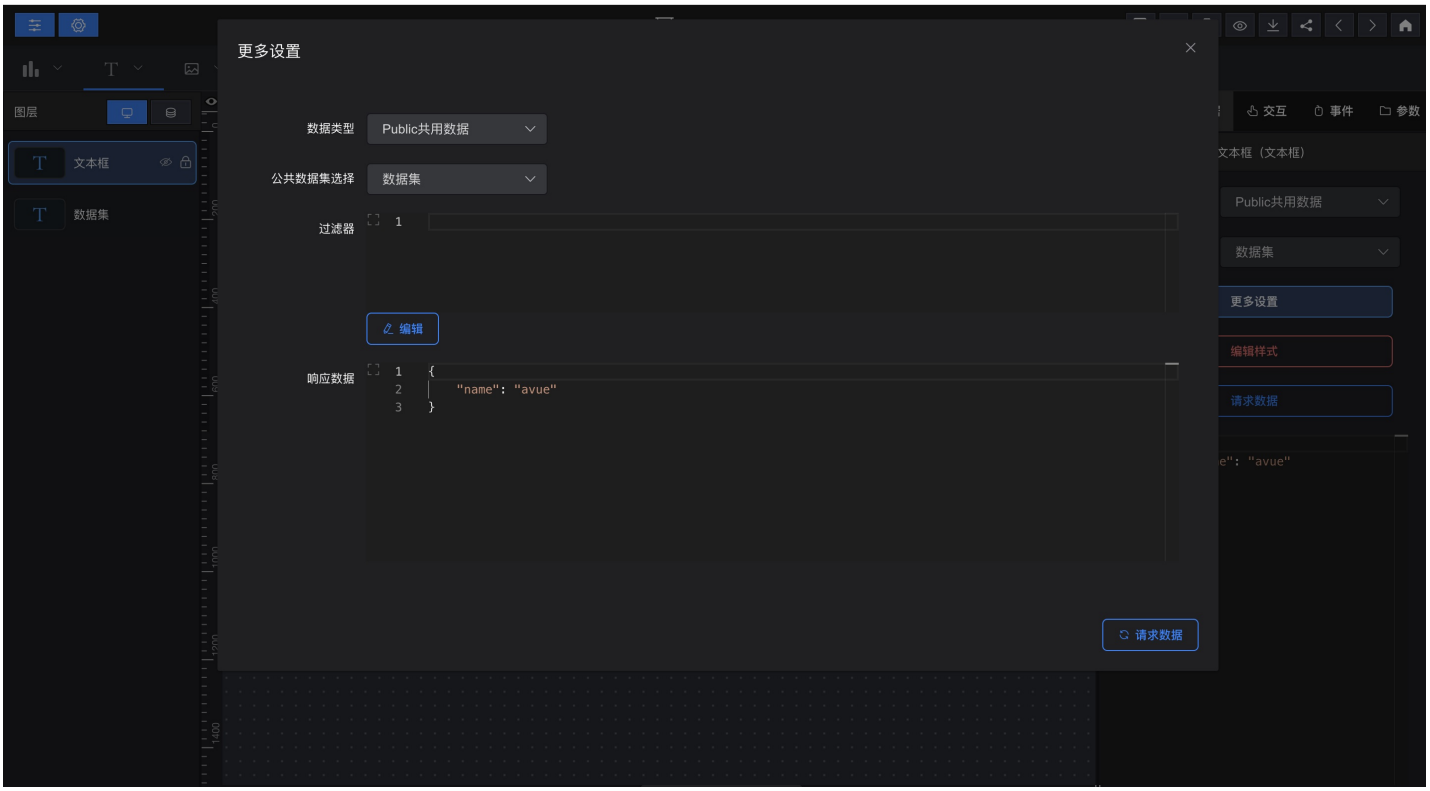




六、Public公共数据

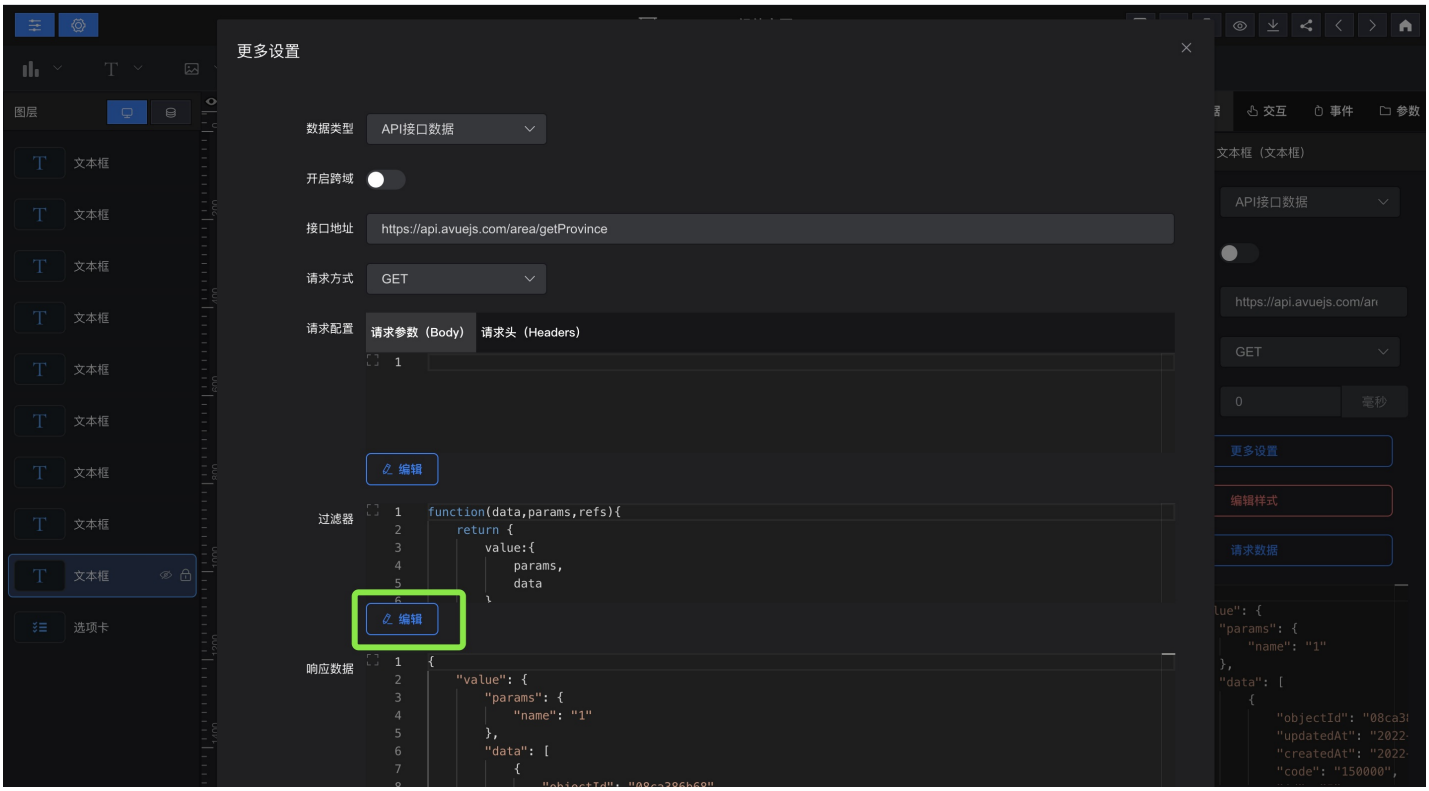
他和Record数据集的用法一样，他是相对于页面的，他需要在页面配置，完后对应的组件去引用，它只会发送一次请求，把数据结果下发到每个组件使用


AvueData-123
通用型
通用型
datav
柱形图
折线图
饼图
象形图
雷达图
散点图
漏斗图
地图
矩形图
自定义Vue组件
定时器
数据集
配置
大屏大小
1920
x
1080
大屏简介
背景颜色
背景图片
/img/bg/bg.png
预览方式
X轴铺满，Y轴自适应滚动
Y轴铺满，X轴自适应滚动
强行拉伸画面，填充所有视图
更多设置


更多设置
数据类型
Public共用数据
公共数据集选择
数据集
过滤器
编辑
响应数据
{
"name": "avue"
}
请求数据
图层
文本框
数据集
交互
事件
参数
文本框（文本框）
Public共用数据
数据集
更多设置
编辑样式
请求数据

数据过滤器

在实际应用中，不同的数据源返回的数据格式可能并不完全符合组件需要的数据格式要求，这时候就需要使用过滤器进行数据格式转换。过滤器可以将原始数据转换成组件需要的标准数据格式，以便组件能够正常显示和处理数据。



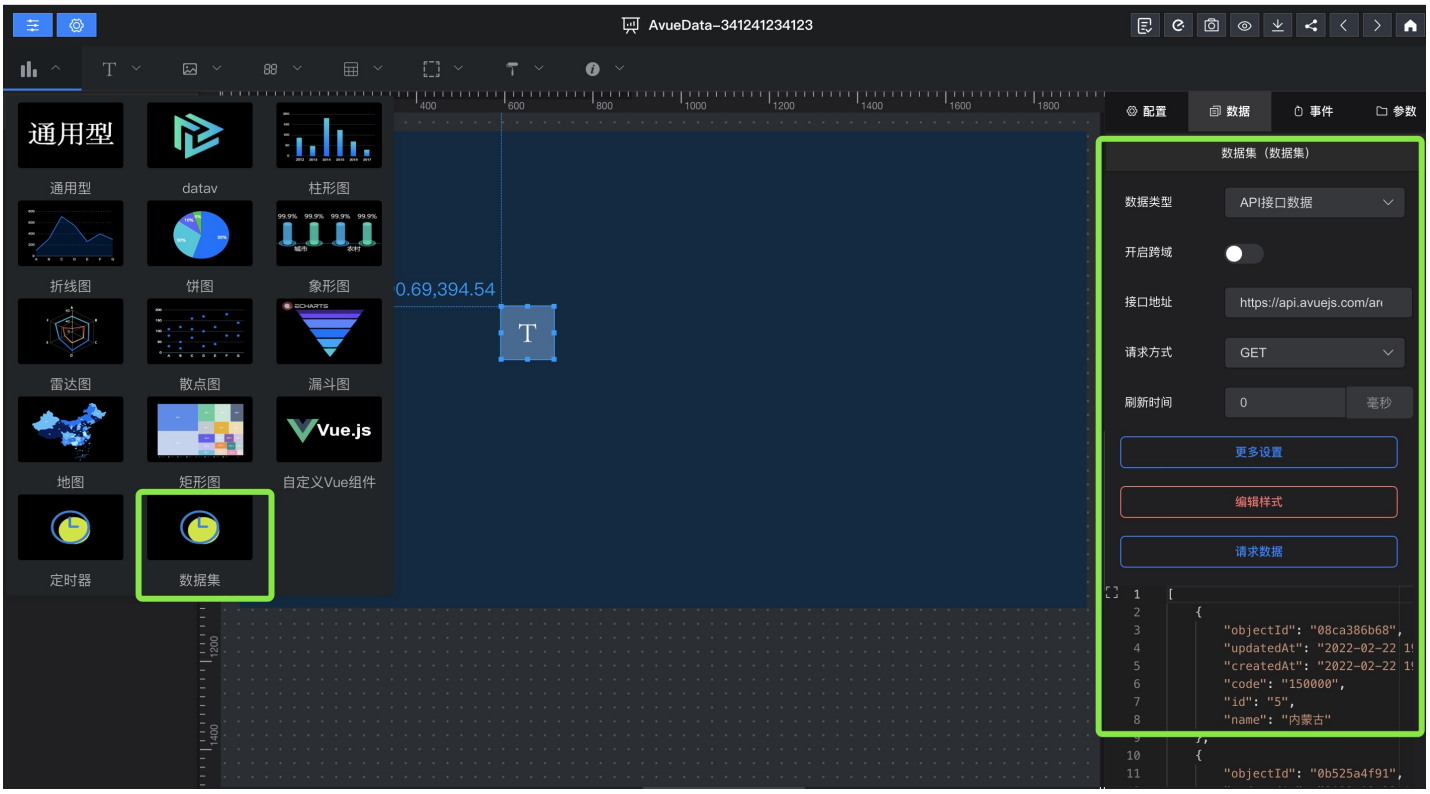
下方的例子是将数据处理成一个数组返回

```
//data为数据源返回的原始数据
//params为组件参数交互传递的参数
//refs为其他的组件的实例对象
function(data,params,refs){
    return  {
        value:{
            list:[data]
        }
    }
}
```

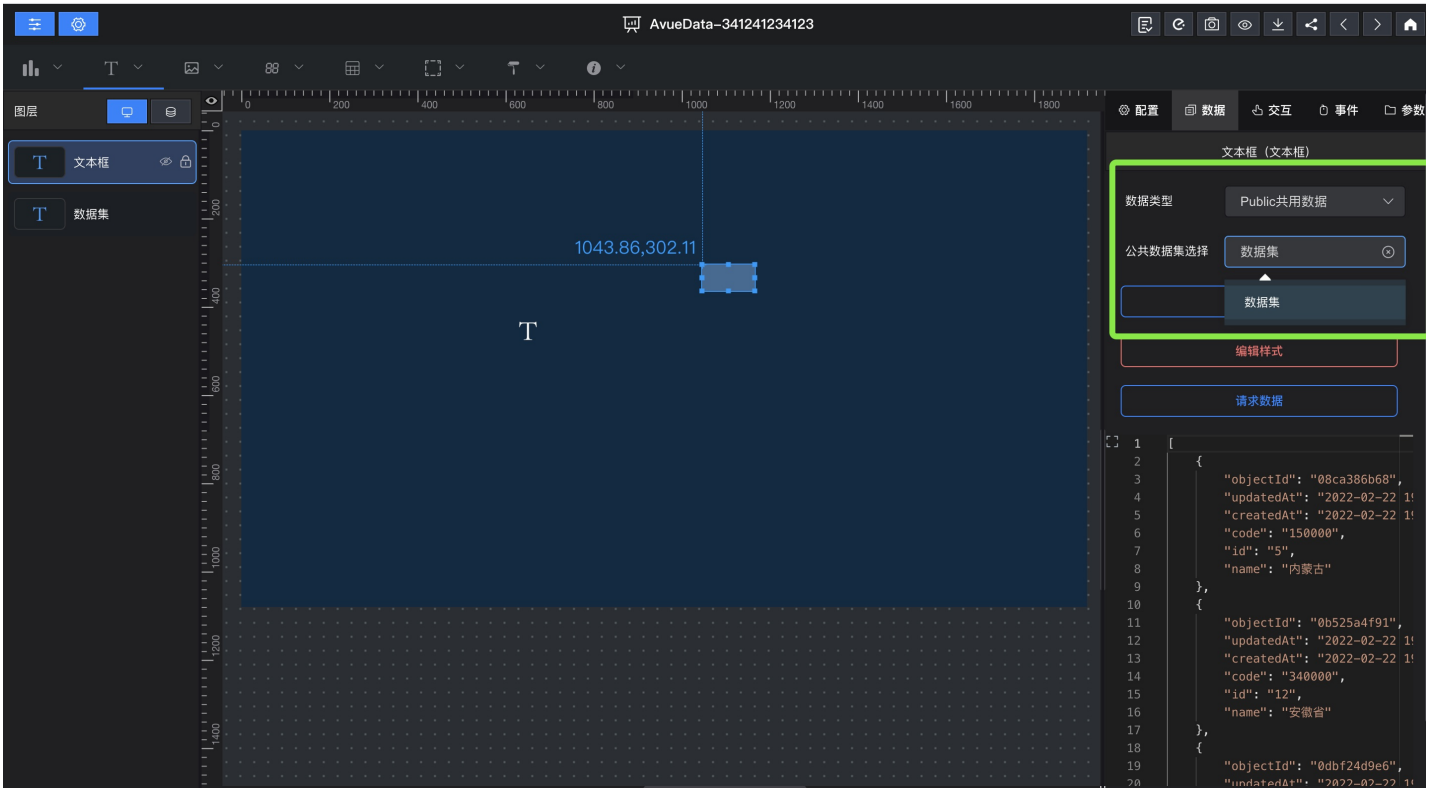
数据集共用

多个组件共用一个数据接口，如果按照正常写法一个组件就会请求一次，可以配置Public数据集去达到共用的目的

1.添加数据集，并且配置数据源



2.需要使用数据的组件选择Public共公数据，并且选择下拉对应的数据集，多个组件可以选择当前数据集，达到数据共用的目的



组件参数

> 组件之间的参数交互点击我跳转参考例子

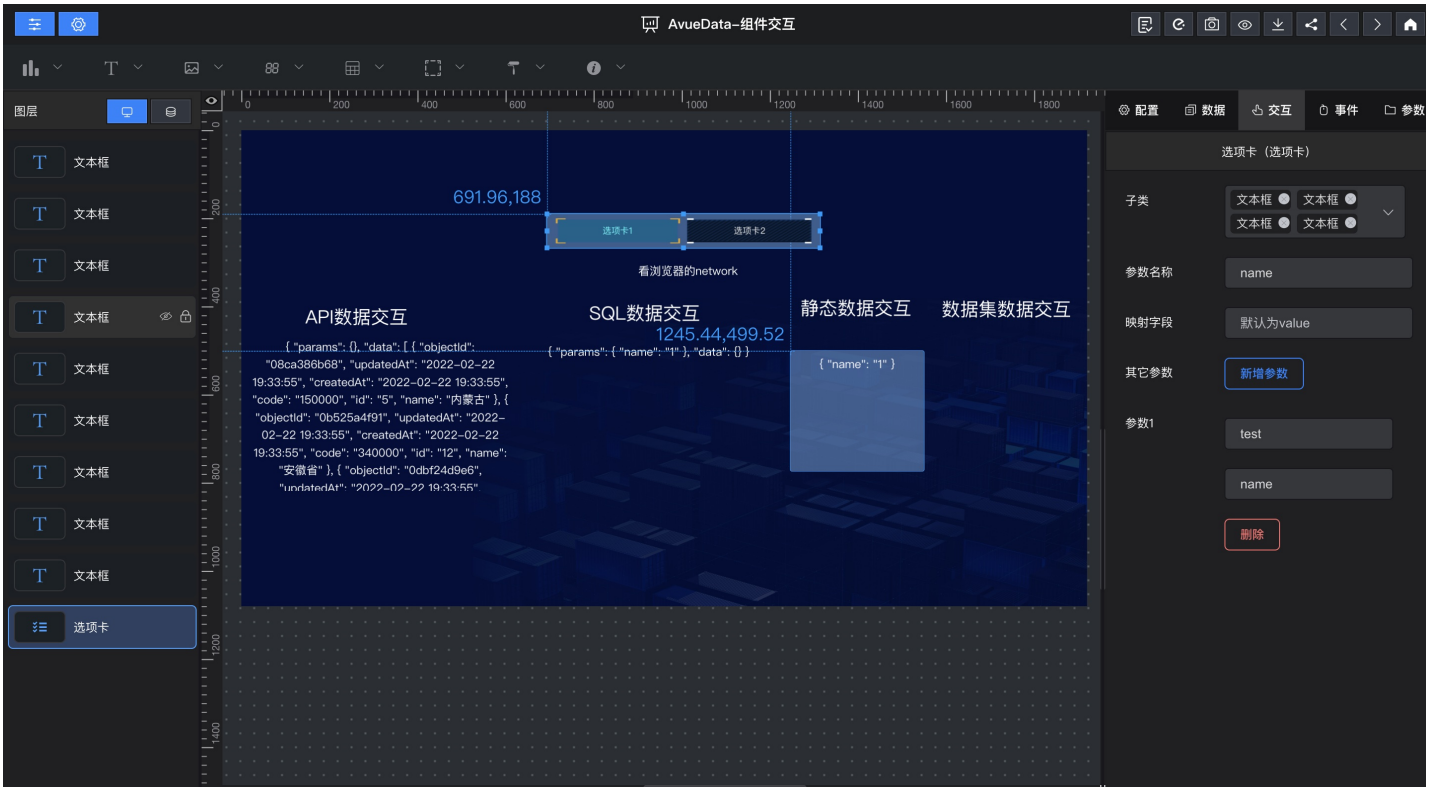
你可以选择多个需要交互的组件，然后为组件设置对应的参数名称和映射字段。

参数名称指传递给组件的字段名称，映射字段则指数据源中需要取出的数据。

例如数据格式如下，当点击选项卡时，传递数据到子类组件

```
{
    name:'123',
    value:'我是值'
}
```

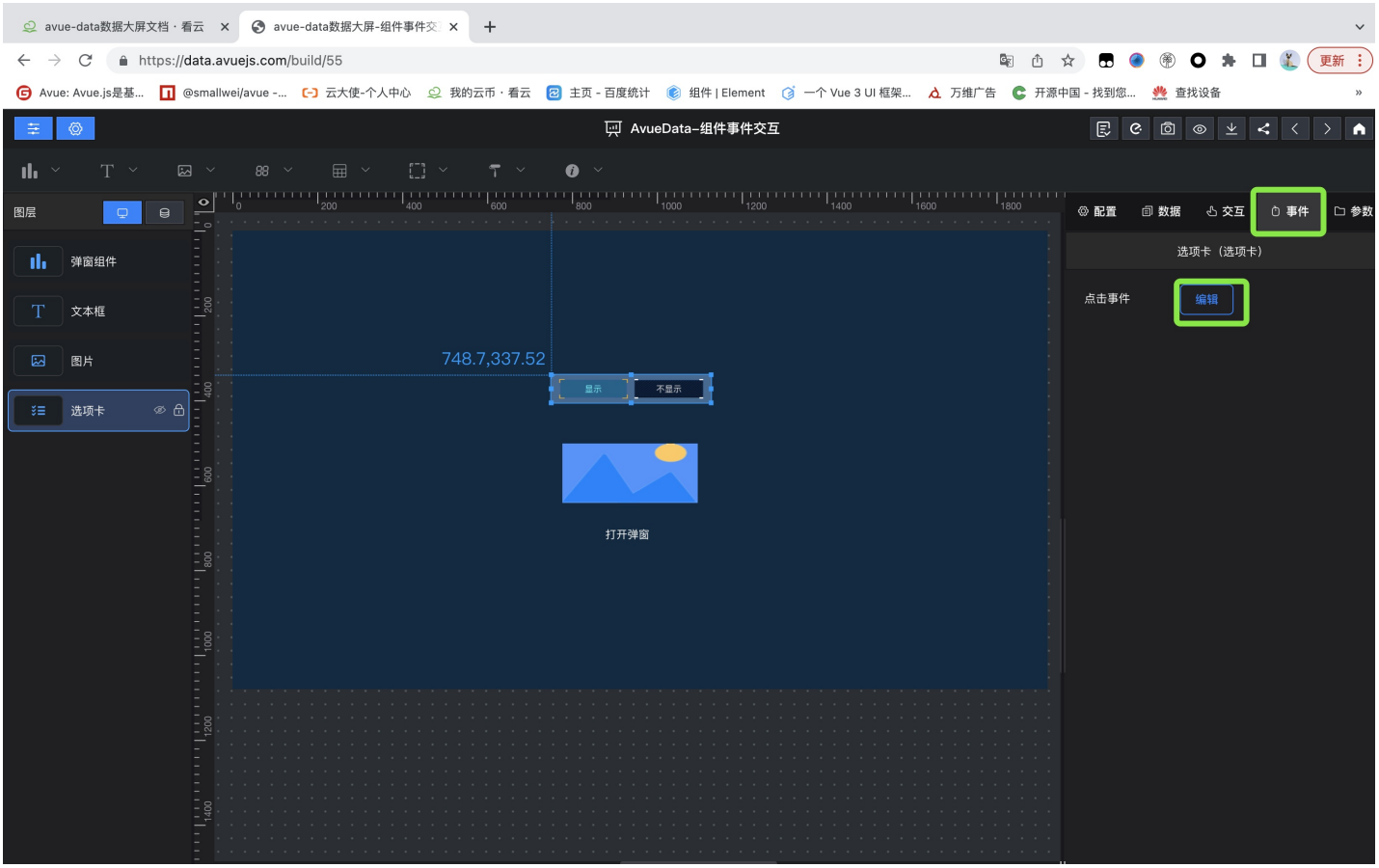
参数名称为test，映射字段为value，传递过去会是test=我是值这样的格式



组件交互

> 组件之间的动态联动点击我跳转参考例子

1.要操作组件中找到"事件"选项卡去写点击事件操作函数

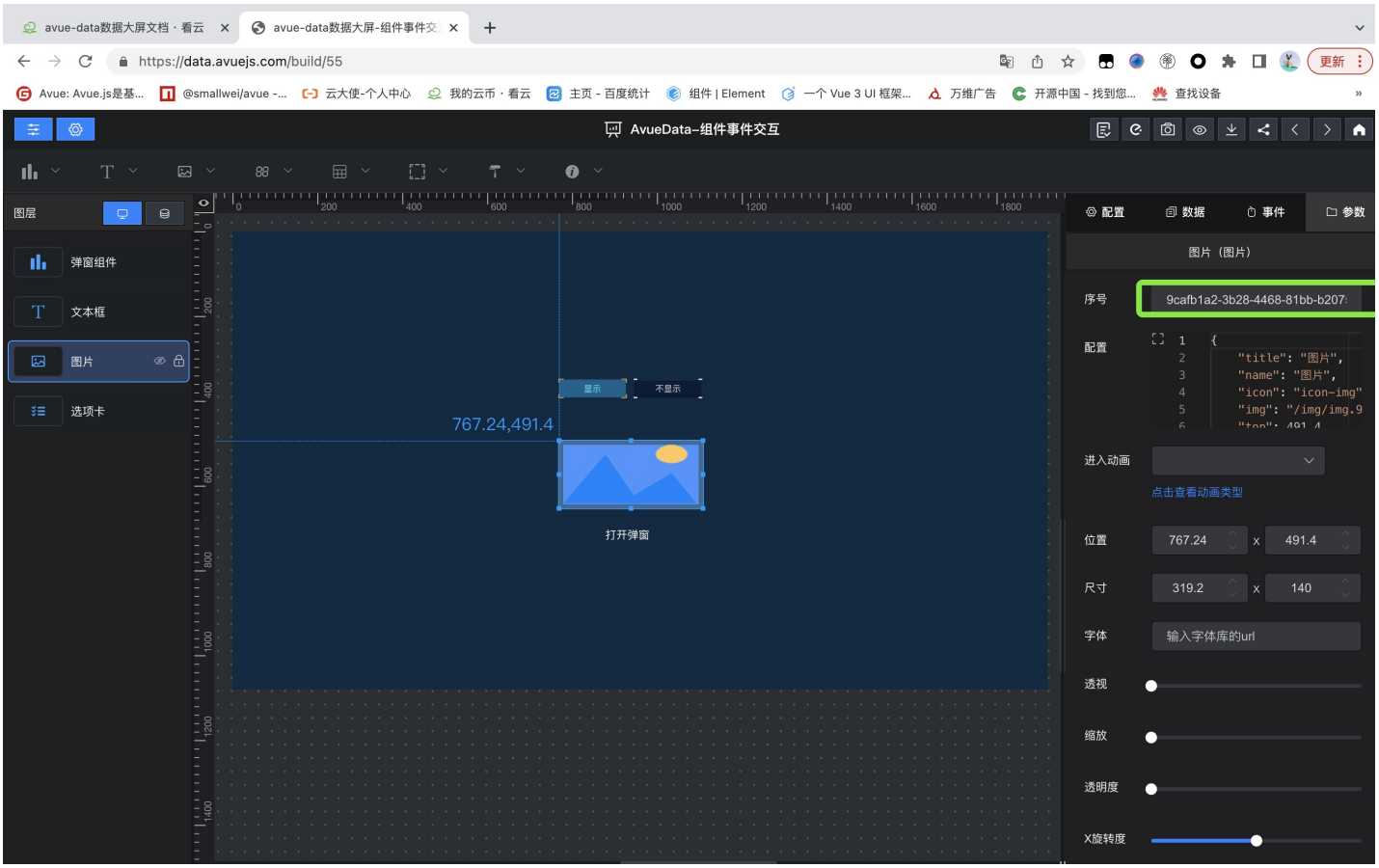


这里可JS处理逻辑，同时也可以拿到window.$glob全局变量

- params参数为点击时候的数据
- refs为画布中全部组件对象，其中ID就是组件的对应序号

```
function(params,refs){
    //这里写点击逻辑
}
```

2.我们做一个动态显隐和赋值，画布中新建3个对应组件，"参数"选项卡中找到组件的序号值

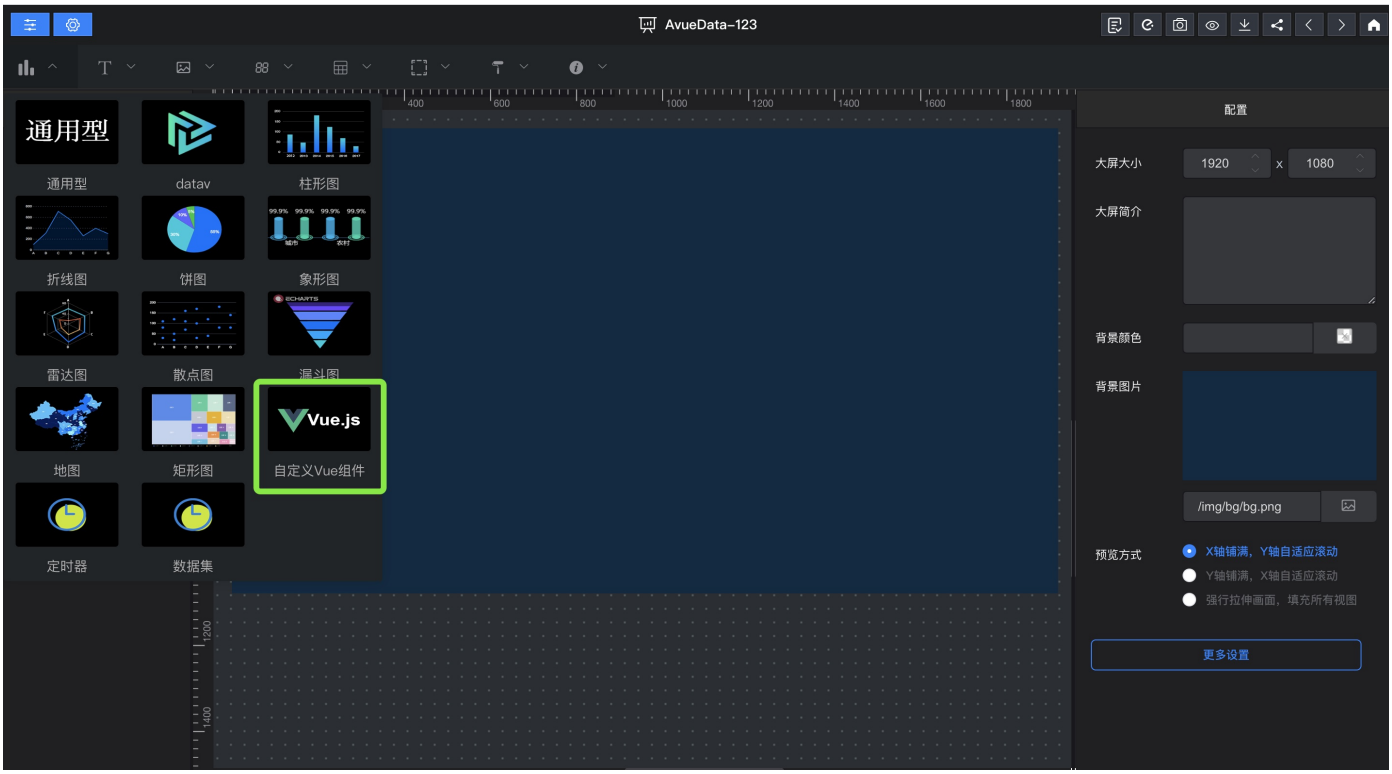


显隐逻辑如下:

```
function(params,refs){
    #图片组件
    let imgObj=refs['9cafb1a2-3b28-4468-81bb-b207501ea036']
    #文本组件
    let textObj=refs['7def40b9-61c1-4941-8637-7f46f408547c']
    if(params.value==1){
        imgObj.$el.style.display="block"
        textObj.dataChart.value='显示'
    }else{
        imgObj.$el.style.display="none"
        textObj.dataChart.value='隐藏'
    }
}
```

自定义Vue组件

> 可以在线自定义Vue组件和引入三方库点击我跳转参考例子



正常编写vue语法组件

1.需要引入三方插件库，利用全局函数window.$loadScript方法

```
//引入jquery
window.$loadScript('js','https://cdn.staticfile.org/jquery/2.1.2/jquery.js').th
en(()=>{

//引入执行后的逻辑

})
```

2.他会把组件请求的数据赋值给dataChart变量，使用dataChart即可拿到多种数据源请求回来的数据

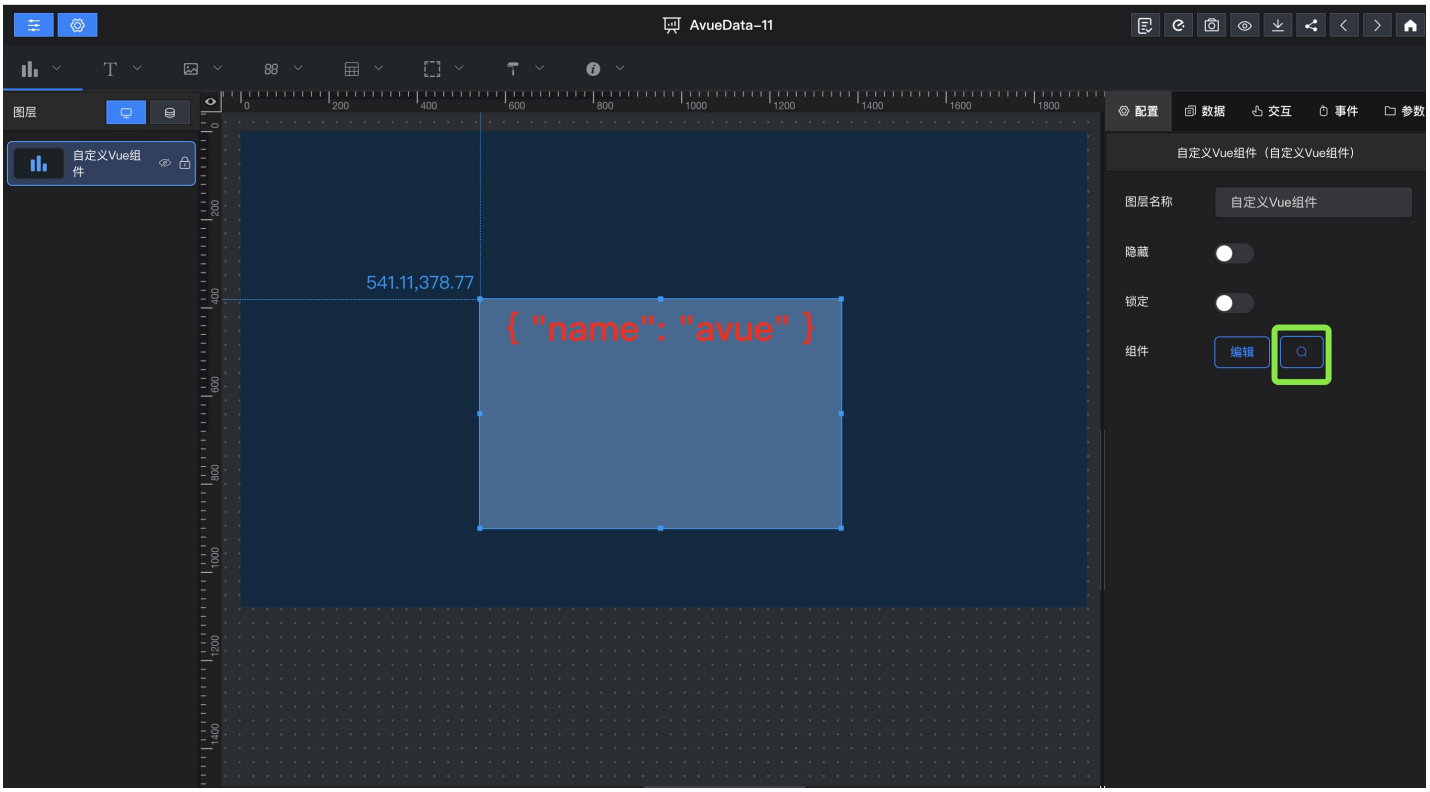
综合例子:

```
<template>
    <div class = "test">
        <el - button type = "success"@ click = "handleClick" > 点击我弹窗 < /el-
button>
        <h1>{{dataChart}}<h1>
    </div>
</template>
<script>
```

```
export default{
    data(){
        return{}
    },
    created(){ },
    methods:{
        handleClick(){
            this.$message.success(this.dataChart.name)
            window.$loadScript('js','https:/ / cdn.staticfile.org / jquery / 2.
1.2 / jquery.js').then(()=>{
                return window.$loadScript('js ','https: //data.avuejs.com/layer
/layer.js')
            }).then(() = > {
                return window.$loadScript('js','https://cdn.staticfile.org/jque
ry.qrcode/1.0/jquery.qrcode.min.js')
            }).then(() = > {
                    layer.open({
                        title: '生成二维码',
                        content: '<div id="qrcode"></div>',
                        success: function(layero, index) {
                            var config = {
                                width: 200,
                                height: 200,
                                text: "苦逼的程序员"
                            }
                            $("#qrcode").qrcode(config);
                        }
                    });
            })
        }
    }
}
< /script>
<style>
    .test{
        text-align:center;
        color:red;
        font-size:40px;
    }
</style >
```

3.同时也可以把常用的组件新增到组件库中，多个大屏去使用

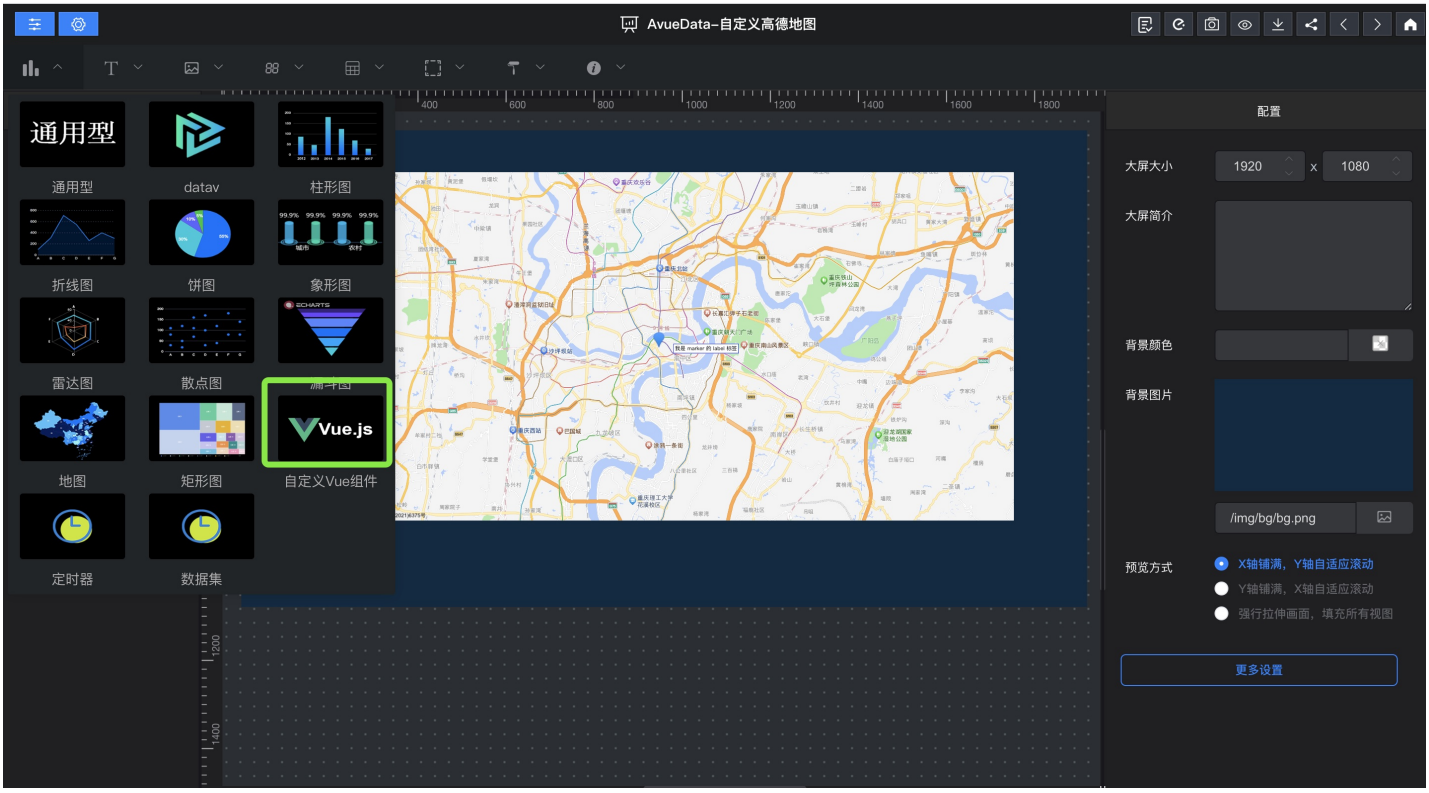

WELCOME TO AVUE-DATA
可视化数据大屏（演示环境-请勿放生产数据）
大屏管理
组件库
地图管理
分类管理
数据源管理
数据集管理
综合功能
新建组件
vue组件
echart组件
共 20 条
50条/页
1
前往
1
页
按钮组
vue组件
步骤条
vue组件
test
vue组件
组件库
vue组件

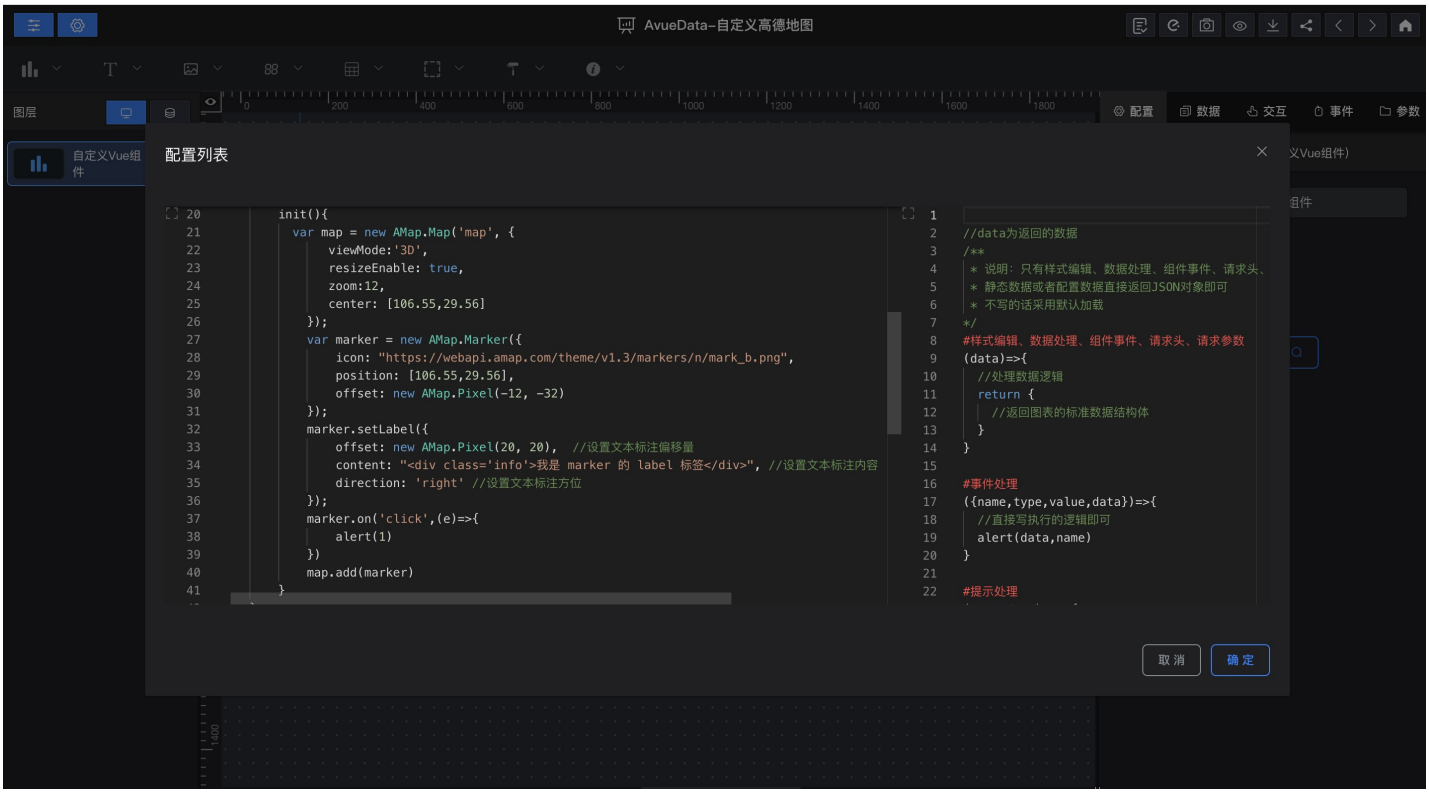

AvueData-11
图层
自定义Vue组件
541.11,378.77
{ "name": "avue" }
配置
数据
交互
事件
参数
自定义Vue组件（自定义Vue组件）
图层名称
自定义Vue组件
隐藏
锁定
组件
编辑

使用高德地图

利用Vue自定义组件使用高德地图参考例子

具体的用法可以参考高德地图官网





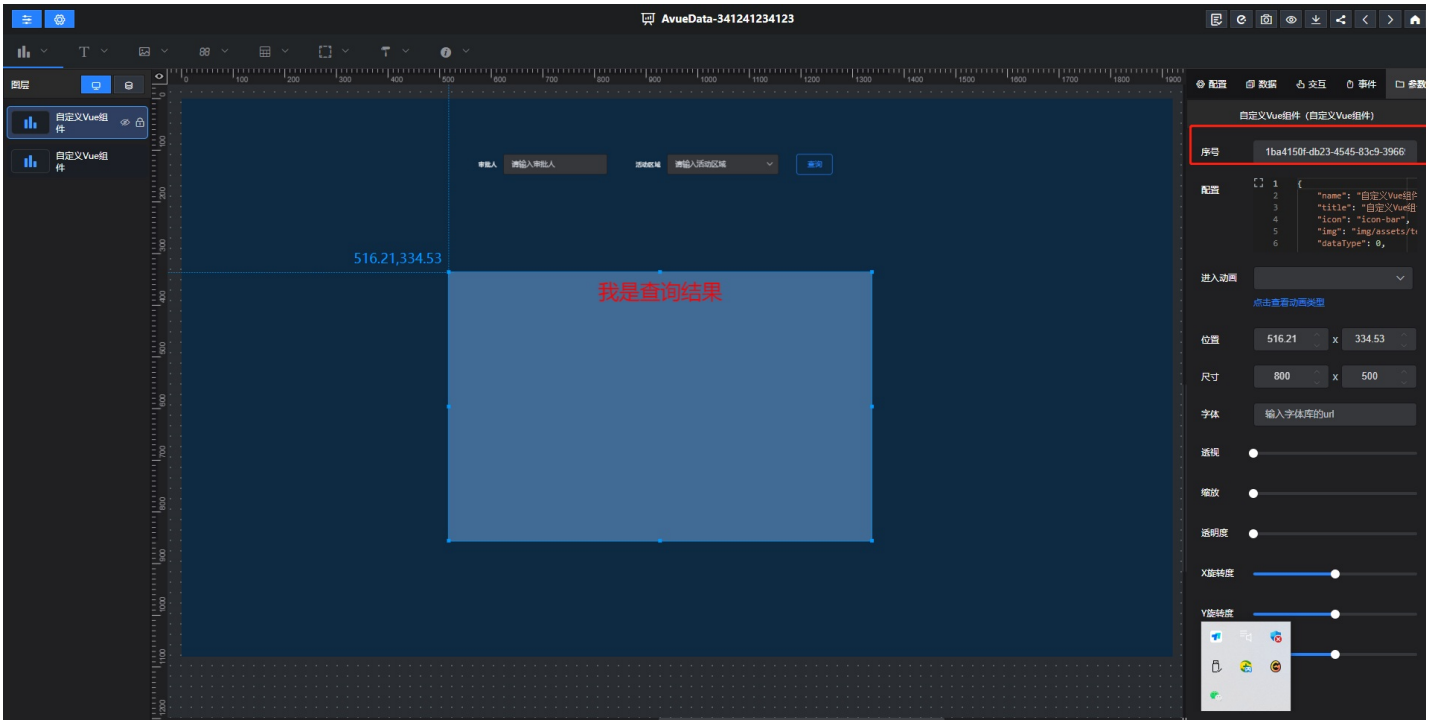
自定义Vue组件之间交互

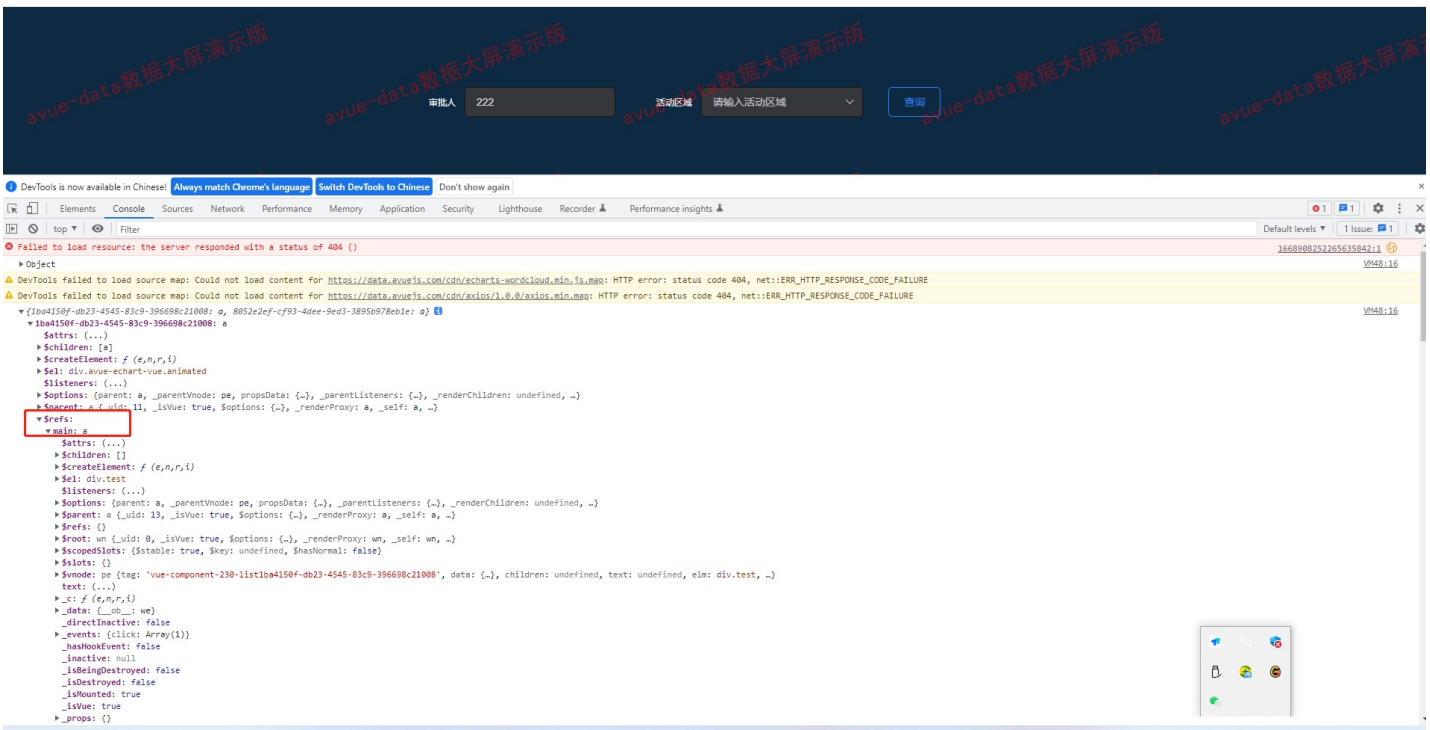
vue组件之间的交互参考例子

如果想操作其他组件，可以在自定义组件内调用父类this.$parent.getItemRefs();方法获取其他组件的实例

```
# 以下是自定义表单，将查询参数传递给其他组件，复制给其他组件的text变量
<template>
    <el-form :inline="true" :model="form">
        <el-form-item label="审批人">
            <el-input v-model="form.user" placeholder="请输入审批人"></el-input>
        </el-form-item>
         <el-form-item label="活动区域">
            <el-select v-model="form.region" placeholder="请输入活动区域">
                 <el-option label="区域一" value="shanghai"></el-option>
                 <el-option label="区域二" value="beijing"></el-option>
         </el-select>
        </el-form-item>
        <el-form-item>
             <el-button type="primary" @click="onSubmit">查询</el-button>
        </el-form-item>
    </el-form>
 </template>
 <script>
 export default {
    data() {
        return {
             form: { user: '', region: '' }
        }
     },
    methods: {
        onSubmit() {
            let refs=this.$parent.getItemRefs();
            console.log(refs);
            refs['1ba4150f-db23-4545-83c9-396698c21008'].$refs.main.text=this.f
orm
         }
     }
 }
</script>
```

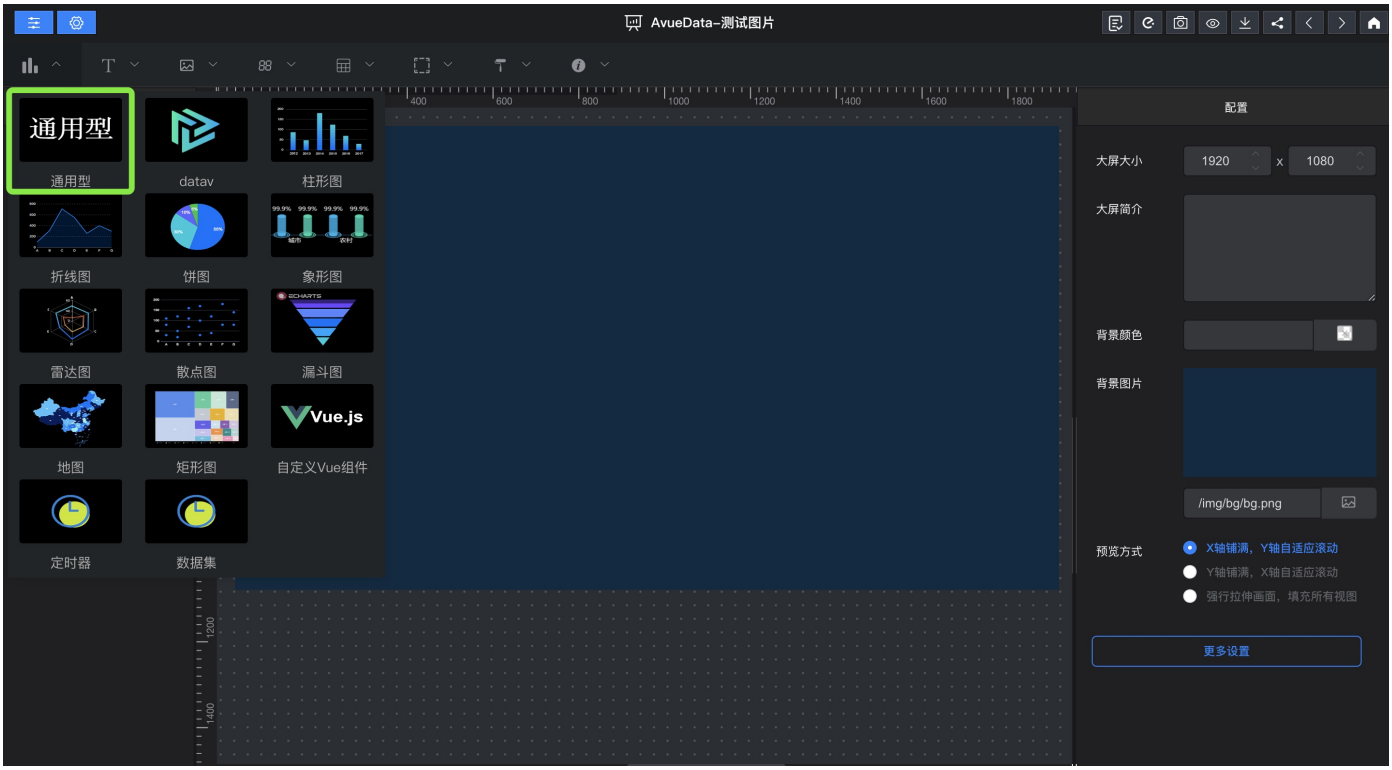
选择其中要传递组件的序号,选择要操作组件内部对象需要获取$refs下的main


AvueData-341241234123
图层
自定义Vue组件
自定义Vue组件
516.21,334.53
我是查询结果
配置
数据
交互
事件
自定义Vue组件（自定义Vue组件）
序号
1ba4150f-db23-4545-83c9-3966
进入动画
位置
516.21
334.53
尺寸
800
500
字体
输入字体库的url
透视
缩放
透明度
X旋转度
Y旋转度


avue-data数据大屏演示版
审批人
222
活动区域
请输入活动区域
查询
DevTools is now available in Chinese!
Always match Chrome's language
Switch DevTools to Chinese
Don't show again
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Failed to load resource: the server responded with a status of 404 ()
DevTools failed to load source map: Could not load content for https://data.avuejs.com/cdn/echarts-wordcloud.min.js.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
DevTools failed to load source map: Could not load content for https://data.avuejs.com/cdn/axios/1.0.0/axios.min.map: HTTP error: status code 404, net::ERR_HTTP_RESPONSE_CODE_FAILURE
$refs:
main: a

自定义Echart组件

> 可以在线自定义echart组件点击我跳转参考例子



正常编写echart语法组件

1.需要引入三方echart插件库，利用全局函数window.$loadScript方法

```
//引入jquery
window.$loadScript('js','https://cdn.staticfile.org/jquery/2.1.2/jquery.js').th
en(()=>{
//引入执行后的逻辑
})
```

2.他会把组件请求的数据赋值给函数中的data变量。

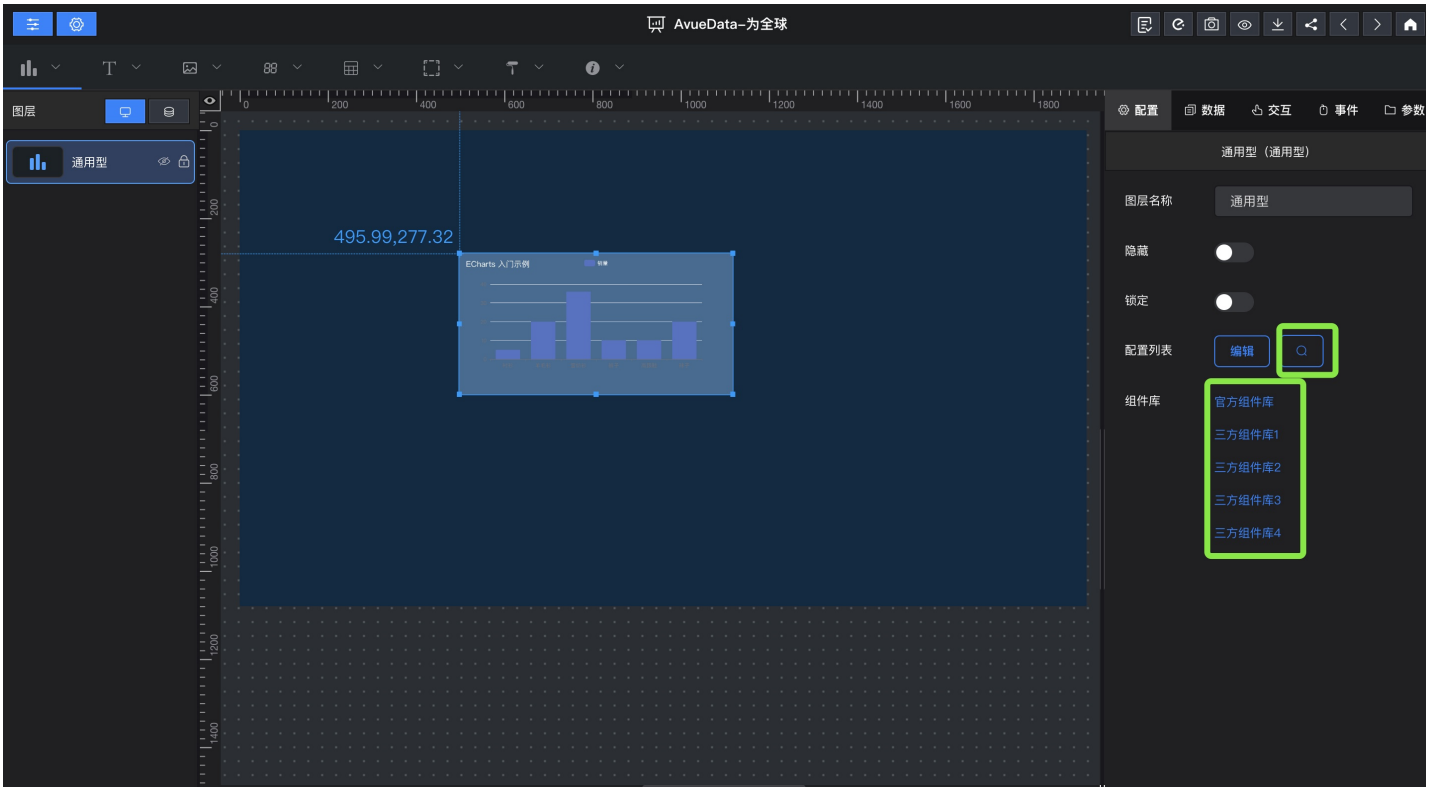
```
//myChart为echart的实例
//option为echart的配置数据
function  (data,params){
    const myChart = this.myChart;
    return option
}
```

综合例子:

```
function  (data,params){
```

```
    const myChart=  this.myChart;
    return  {
        title:  {
            textStyle:  {
            fontWeight:  'normal',
            color:  '#fff'
        },
            text:  'ECharts 入门示例'
        },
        tooltip:  {},
        legend:  {
            data:['销量'],
            textStyle:  {
                fontWeight:  'normal',
                color:  '#fff'
            },
        },
        xAxis:  {
            data:  ['衬衫','羊毛衫','雪纺衫','裤子','高跟鞋','袜子']
        },
        yAxis:  {},
        series:  [{
            name:  '销量',
            type:  'bar',
            data:  \[5,  20,  36,  10,  10,  20\]
        }]
    };
}
```

3.同时也可以把常用的组件新增到组件库中，多个大屏去使用，提供了多个三方组件库去使用


AvueData-为全球
图层
通用型
495.99,277.32
ECharts 入门示例
配置
数据
交互
事件
参数
通用型（通用型）
图层名称
通用型
隐藏
锁定
配置列表
编辑
组件库
官方组件库
三方组件库1
三方组件库2
三方组件库3
三方组件库4

外部组件开发

外部组件

> 组件文件路径/public/components.js

组件模块

编写组件的模块/public/components.js

```
You, 5 days ago | 1 author (You)
const testComponents = {
  template: `
  <div :style="[styleSizeName,styleName]"
    :class="className">
    <div :style="styleChartName">
      <h2>自定义组件</h2><br />
      <h3>我是参数:{{option}}</h3><br />
      <h3>data:{{dataChart}}</h3><br />
      <h3>params:{{(dataAxios.config || {}).params}}</h3><br />
    </div>
  </div>
`,
  name: 'test',
  props: {
    option: Object,
    component: Object
  },
  computed: {
    styleName () {
      return {
        fontSize: this.fontSize,
        color: this.color
      }
    },
    color () {
      return this.option.color || '#fff'
    },
    fontSize () {
      return (this.option.fontSize || 30) + 'px'
    }
  }
}
```

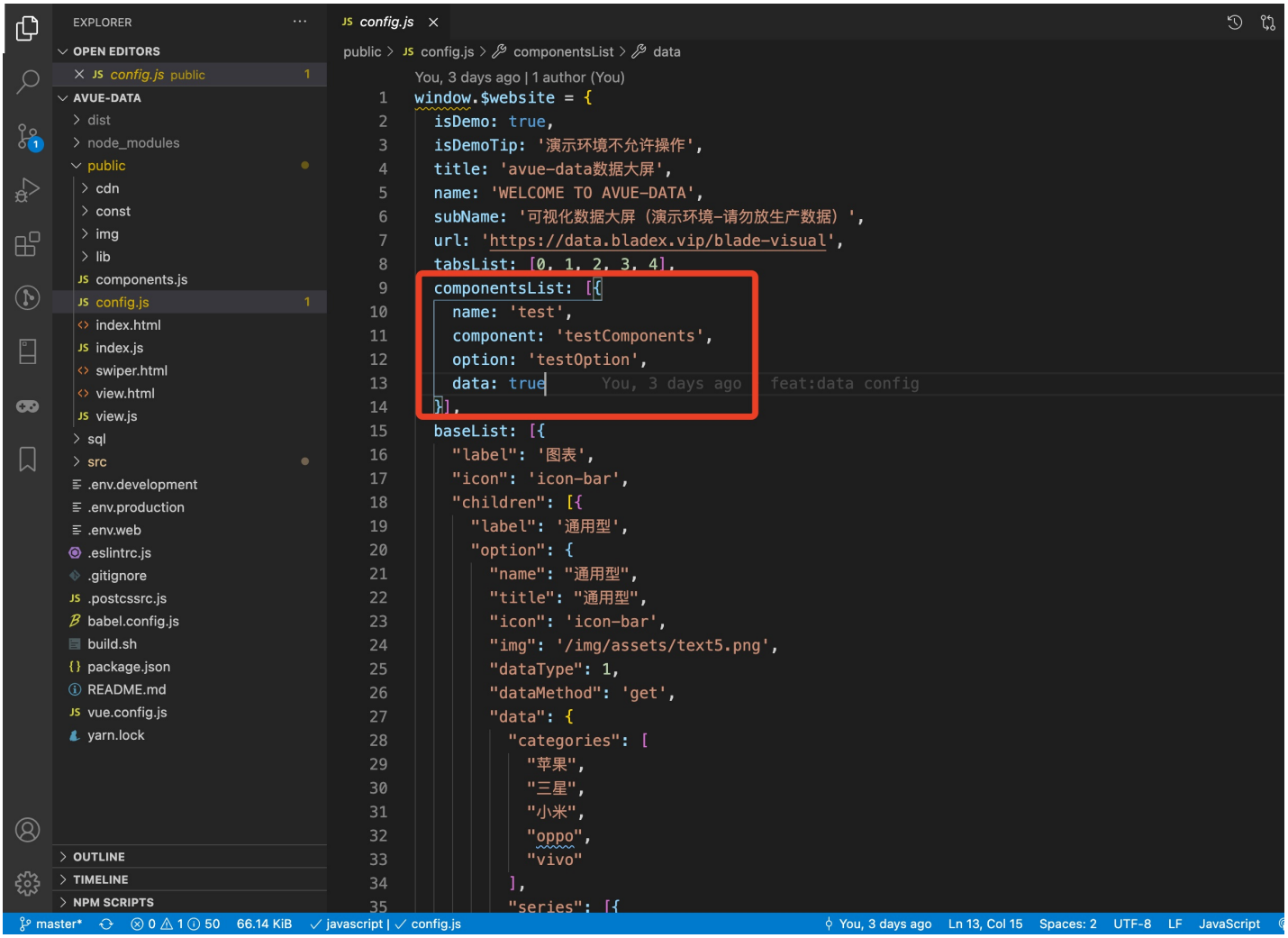
配置模块

编写组件配置项模块/public/components.js

```
const testOption = {
  template: `
  <div>
    <el-form-item label="字体大小">
      <avue-input-number v-model="main.activeOption.fontSize"></avue-input-number>
    </el-form-item>
    <el-form-item label="字体颜色">
      <avue-input-color v-model="main.activeOption.color"></avue-input-color>
    </el-form-item>
  </div>
`,
  name: 'test',
  inject: ["main"]
}
```

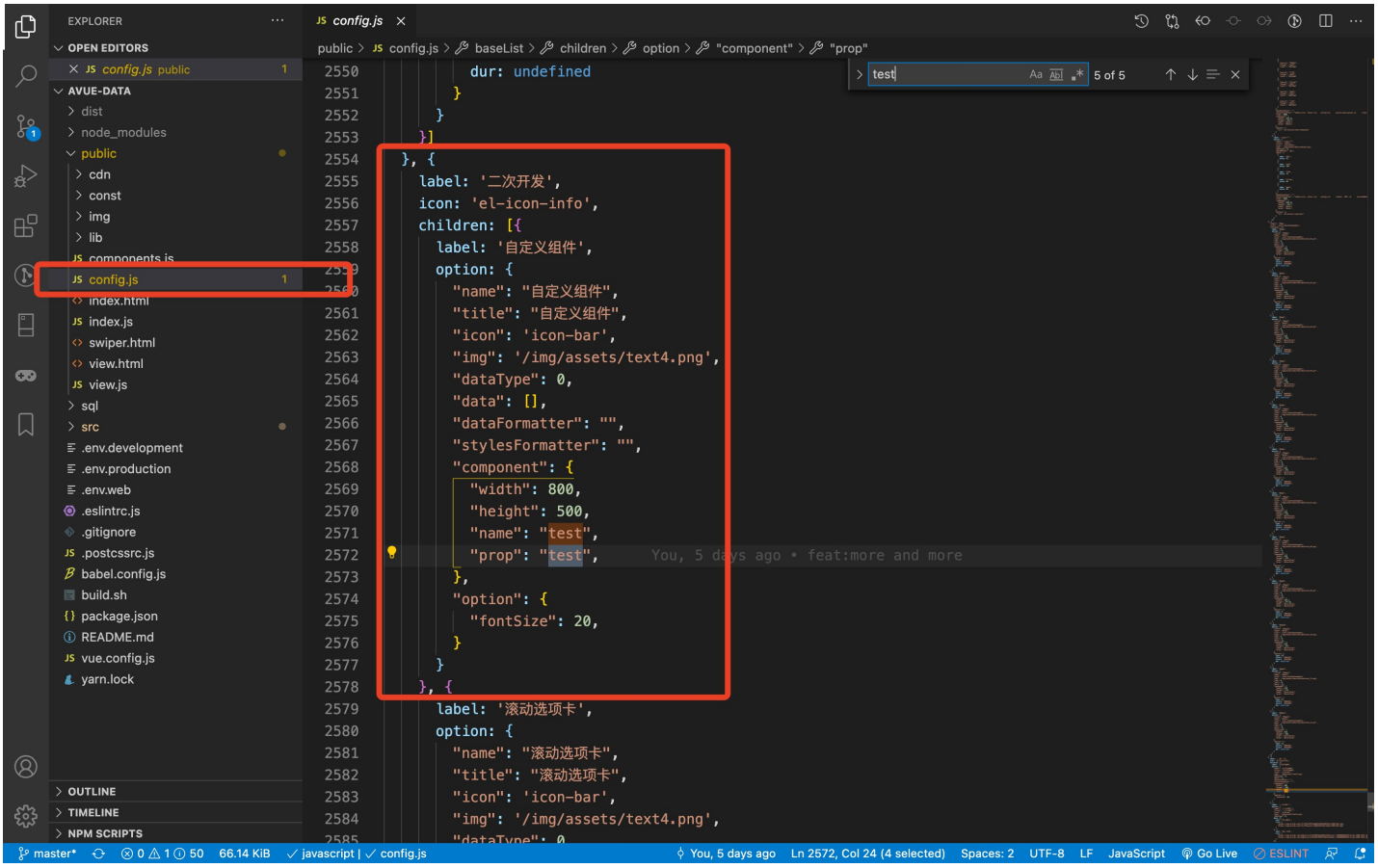
组件配置

配置文件文件路径/public/config.js

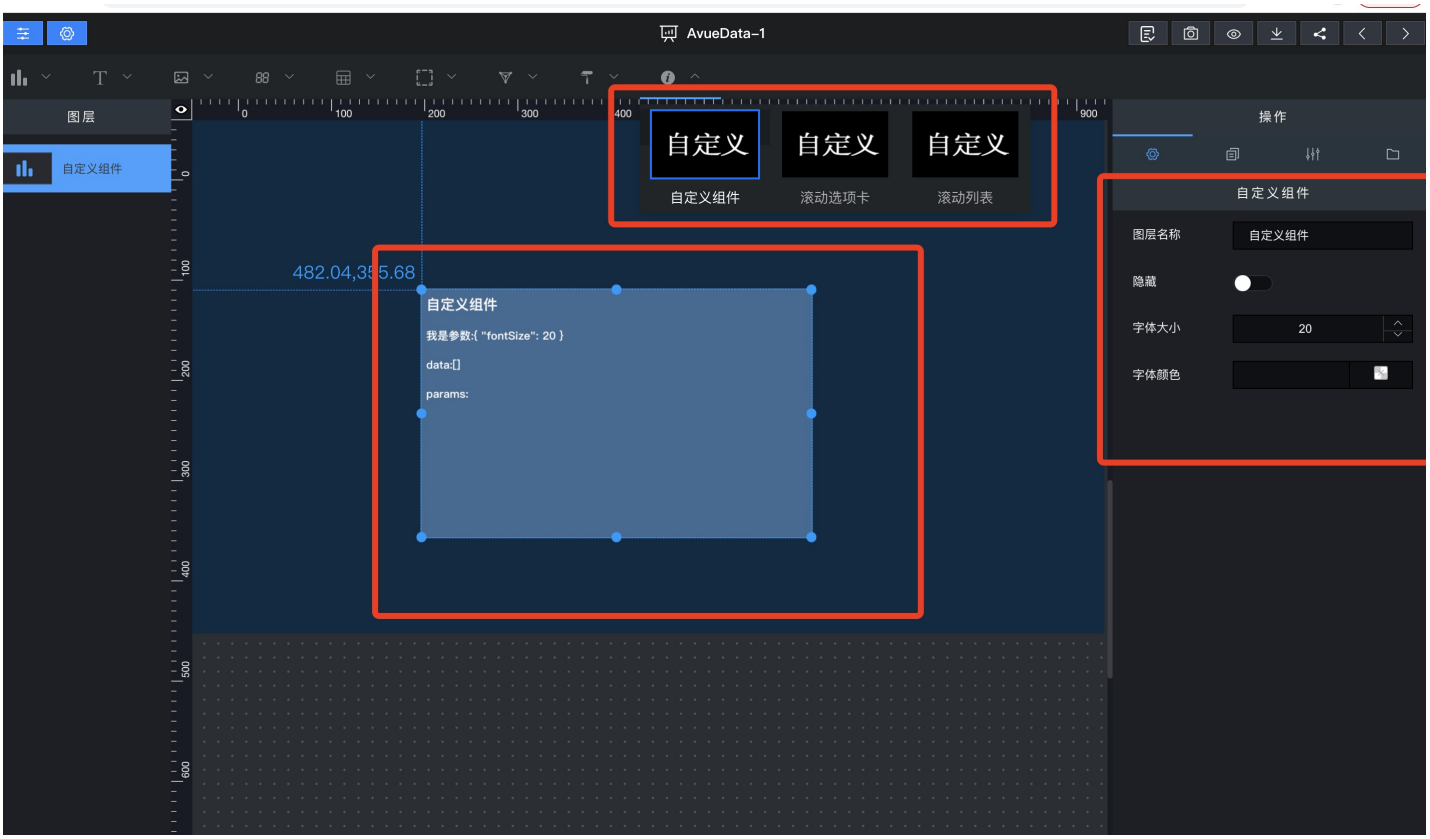


组件引用

配置文件文件路径/public/config.js


EXPLORER
OPEN EDITORS
config.js public
AVUE-DATA
dist
node_modules
public
cdn
const
img
lib
components.js
config.js
index.html
index.js
swiper.html
view.html
view.js
sql
src
.env.development
.env.production
.env.web
.eslintrc.js
.gitignore
.postcssrc.js
babel.config.js
build.sh
package.json
README.md
vue.config.js
yarn.lock
OUTLINE
TIMELINE
NPM SCRIPTS
public > config.js > baseList > children > option > "component" > "prop"
test
5 of 5
dur: undefined
label: '二次开发',
icon: 'el-icon-info',
children: [{
label: '自定义组件',
option: {
"name": "自定义组件",
"title": "自定义组件",
"icon": 'icon-bar',
"img": '/img/assets/text4.png',
"dataType": 0,
"data": [],
"dataFormatter": "",
"stylesFormatter": "",
"component": {
"width": 800,
"height": 500,
"name": "test",
"prop": "test",
You, 5 days ago • feat:more and more
},
"option": {
"fontSize": 20,
label: '滚动选项卡',
option: {
"name": "滚动选项卡",
"title": "滚动选项卡",
"icon": 'icon-bar',
"img": '/img/assets/text4.png',
master*
javascript | config.js
You, 5 days ago
Ln 2572, Col 24 (4 selected)
Spaces: 2
UTF-8
LF
JavaScript
Go Live
ESLINT

最终效果


AvueData-1
图层
自定义组件
自定义
自定义
自定义
自定义组件
滚动选项卡
滚动列表
操作
自定义组件
图层名称
自定义组件
隐藏
字体大小
20
字体颜色
自定义组件
我是参数:{ "fontSize": 20 }
data:[]
params:

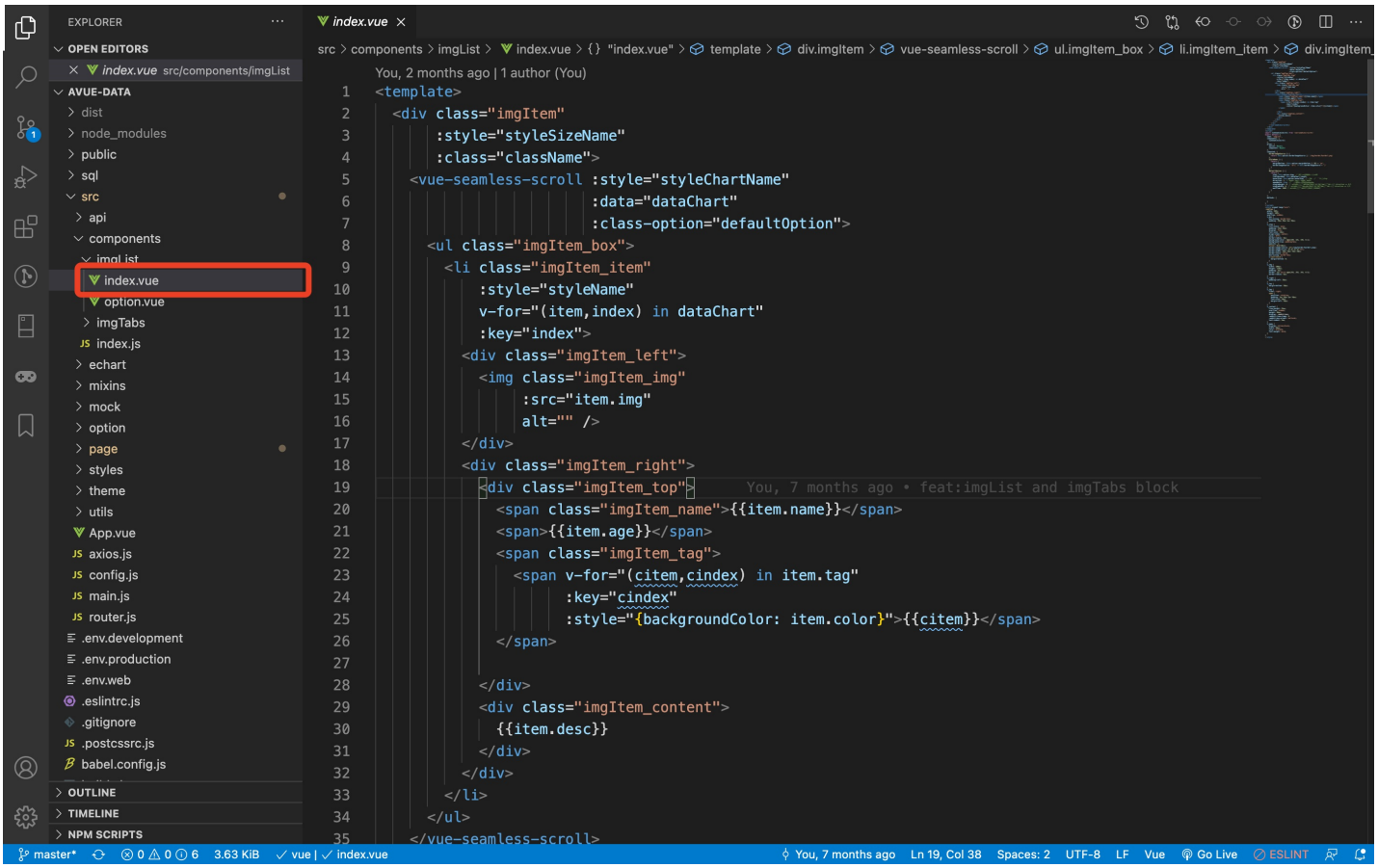
内部组件开发

内部组件

组件文件路径/src/components/**.js，配置对应组件，系统会自动引入模块

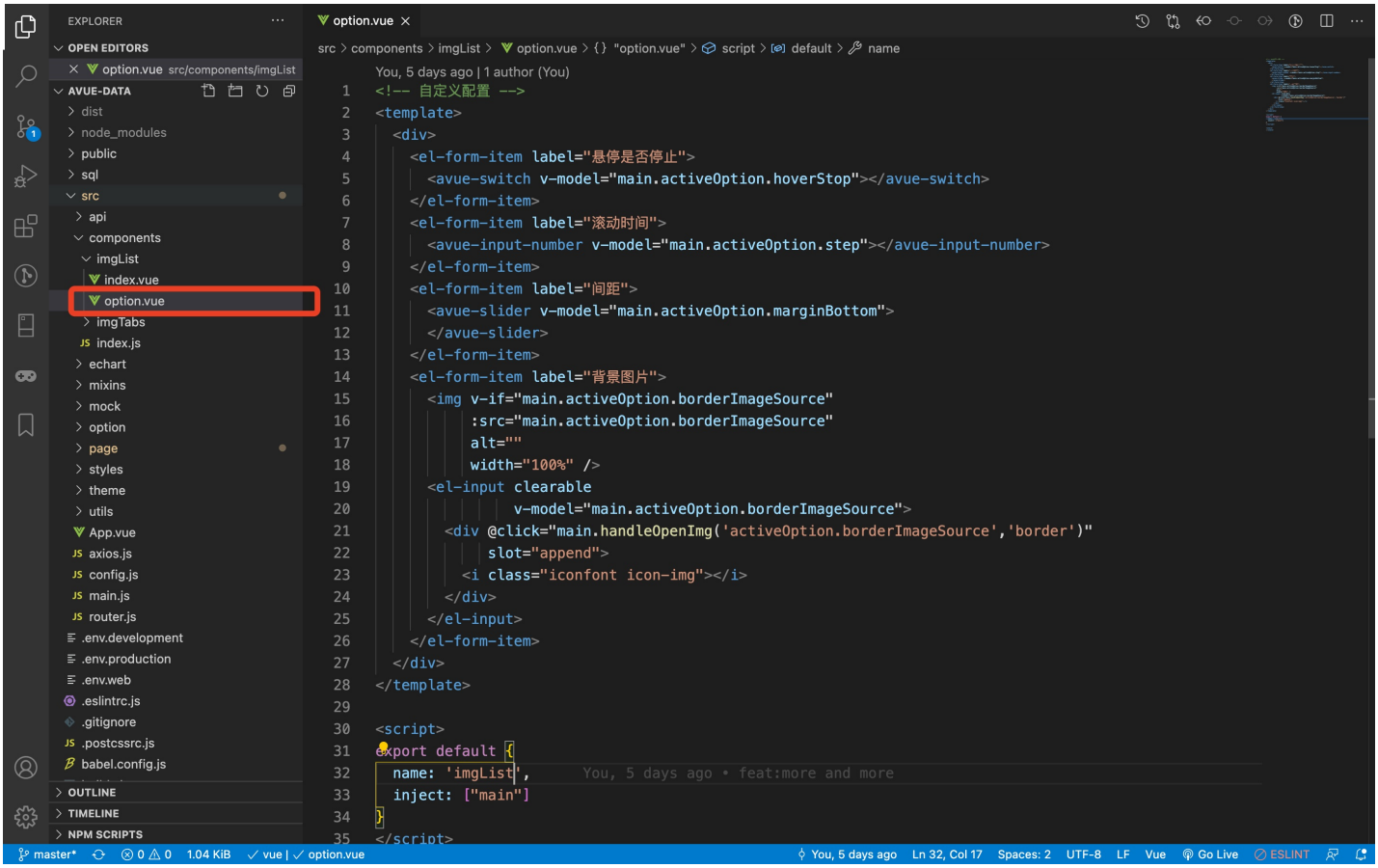
组件模块

编写组件的模块/src/components/imgList/index.vue



配置模块

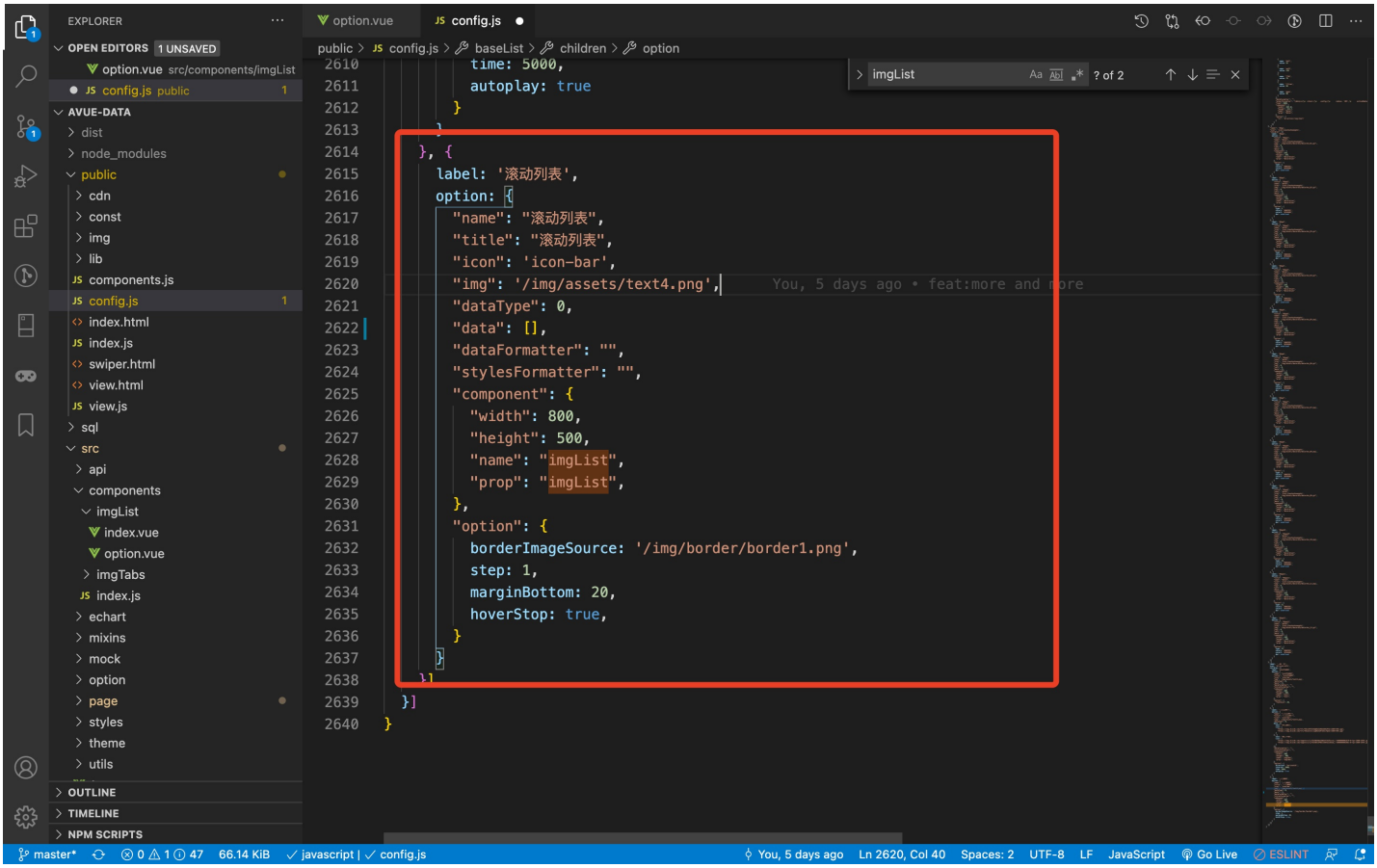
编写配置模块/src/components/imgList/option.vue



```vue
<!-- 自定义配置 -->
<template>
  <div>
    <el-form-item label="悬停是否停止">
      <avue-switch v-model="main.activeOption.hoverStop"></avue-switch>
    </el-form-item>
    <el-form-item label="滚动时间">
      <avue-input-number v-model="main.activeOption.step"></avue-input-number>
    </el-form-item>
    <el-form-item label="间距">
      <avue-slider v-model="main.activeOption.marginBottom">
      </avue-slider>
    </el-form-item>
    <el-form-item label="背景图片">
      <img v-if="main.activeOption.borderImageSource"
           :src="main.activeOption.borderImageSource"
           alt=""
           width="100%" />
      <el-input clearable
                v-model="main.activeOption.borderImageSource">
        <div @click="main.handleOpenImg('activeOption.borderImageSource','border')"
             slot="append">
          <i class="iconfont icon-img"></i>
        </div>
      </el-input>
    </el-form-item>
  </div>
</template>

<script>
export default {
  name: 'imgList',
  inject: ["main"]
}
</script>
```

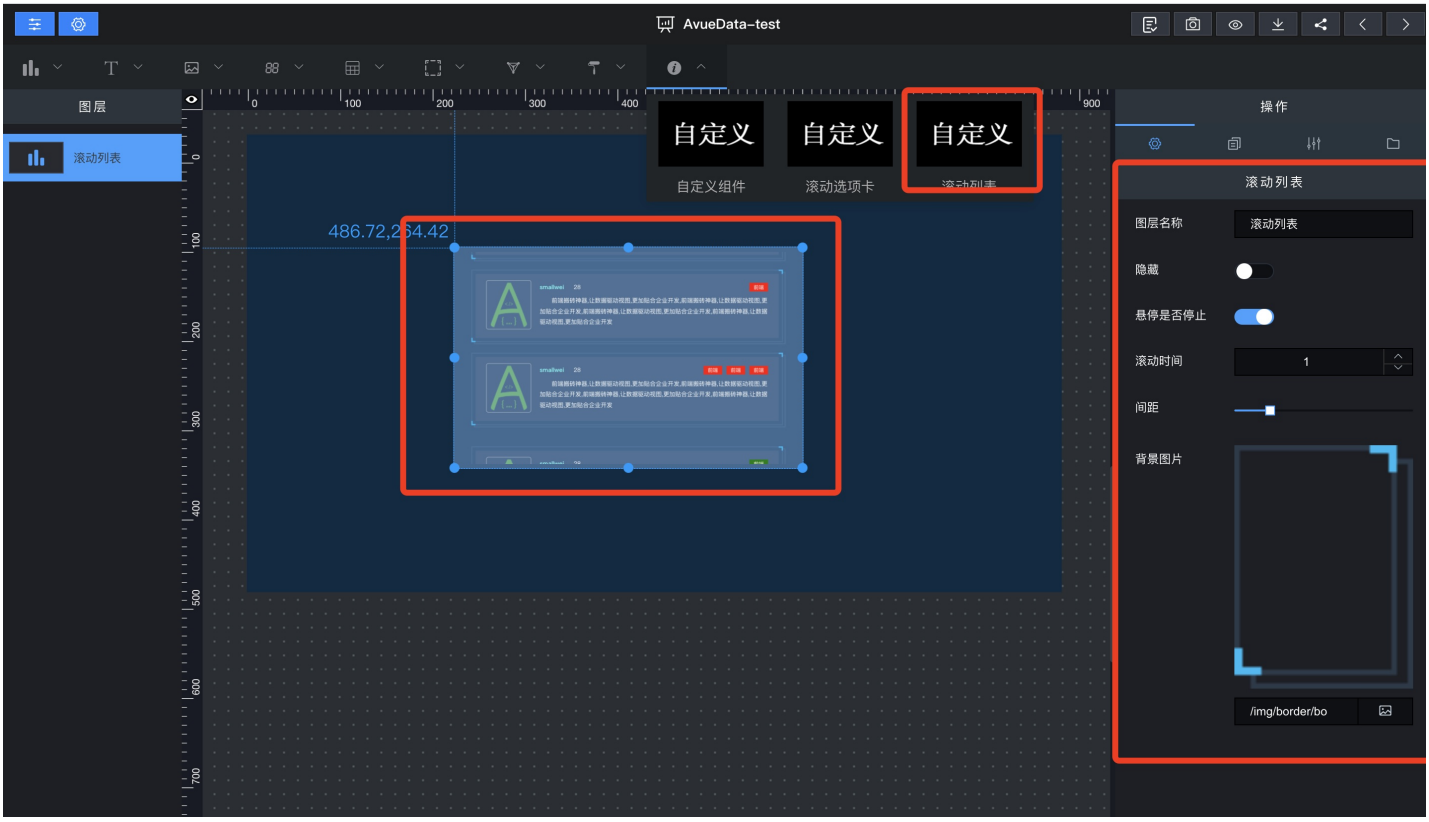
组件引用

配置文件文件路径/public/config.js



```json
{
  "name": "滚动列表",
  "title": "滚动列表",
  "icon": 'icon-bar',
  "img": '/img/assets/text4.png',
  "dataType": 0,
  "data": [],
  "dataFormatter": "",
  "stylesFormatter": "",
  "component": {
    "width": 800,
    "height": 500,
    "name": "imgList",
    "prop": "imgList",
  },
  "option": {
    borderImageSource: '/img/border/border1.png',
    step: 1,
    marginBottom: 20,
    hoverStop: true,
  }
}
```

最终效果

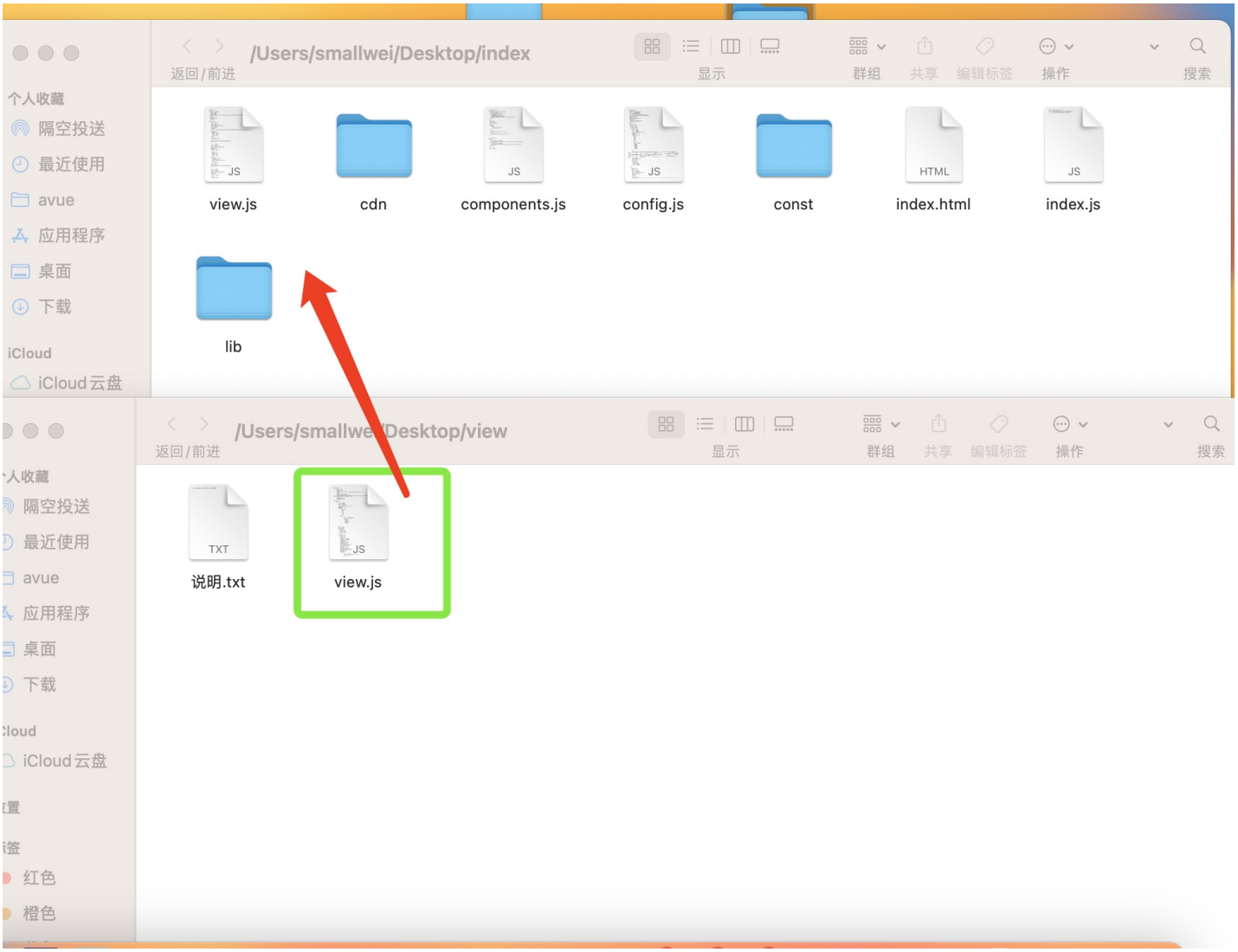

AvueData-test
图层
滚动列表
自定义
自定义
自定义
自定义组件
滚动选项卡
操作
滚动列表
图层名称
滚动列表
隐藏
悬停是否停止
滚动时间
1
间距
背景图片
/img/border/bo

单独大屏导出部署

1.选择对应的大屏模块，点就导出，会导出2个压缩包(index.zip和view.zip)

- index.zip大屏运行基础依赖
- view.zip大屏的结构数据

2.将2个压缩包解压，将view中的view.js替换index中的view.js



3.完后将index文件夹整个目录部署到服务器中即可，以下为nginx为例子

```
server
{
    listen 80;
    server_name 127.0.0.1;
    index index.php index.html index.htm default.php default.htm default.html;
    root /www/wwwroot/index; #index静态页目录路径
}
```

教学视频

教学视频

- 多种数据源的使用
- 三方网站数据解析
- 组件传参交互
- 自定义组件传参交互1
- 自定义组件传参交互2
- 用Ref去做组件交互
- 自定义组件加载三方包
- 多屏幕切换
- 定时器组件使用
- 全局变量和全局过滤器
- 消息模板推送
- 语音消息告警
- 大屏导出独立部署

项目部署

Nginx部署

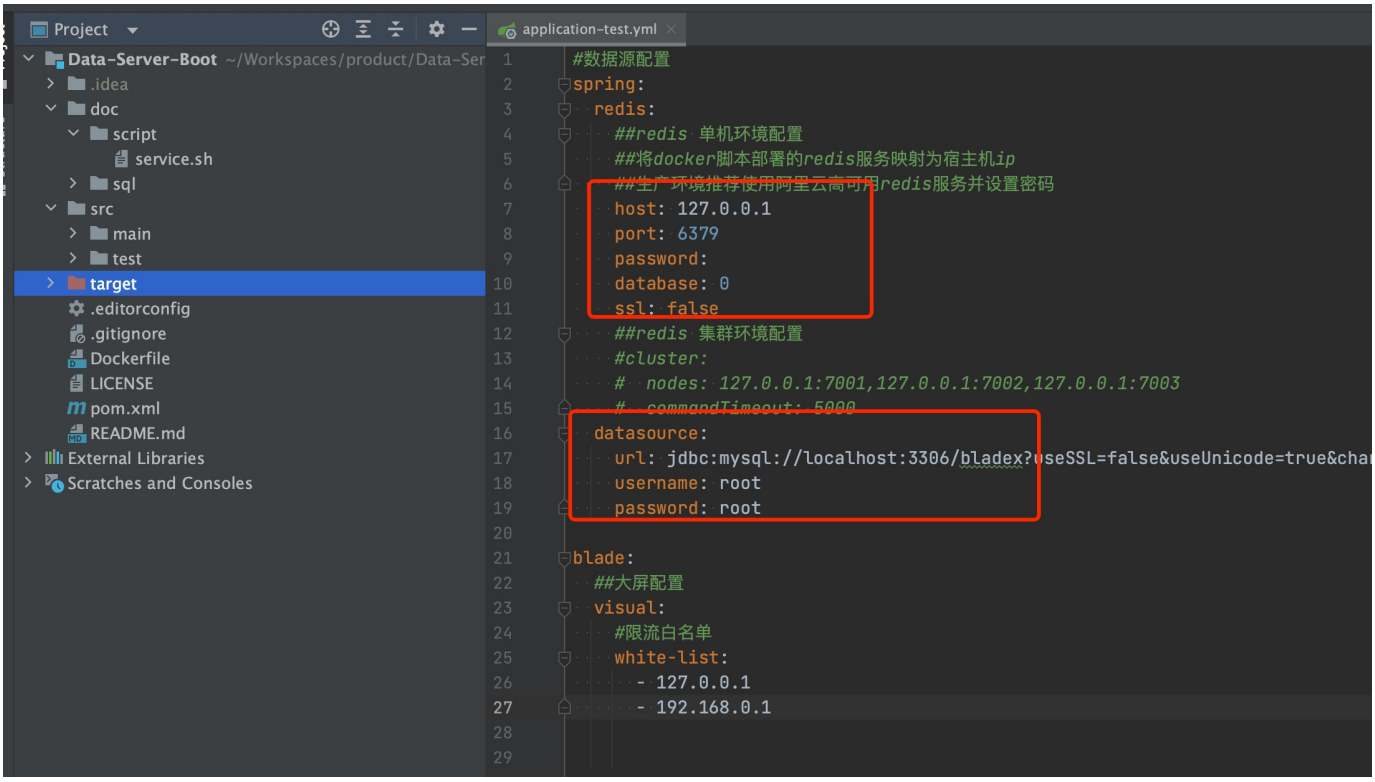
> 执行npm run build打出对应的目录

```
//nginx为例子
location /{
    root /data/avue/avue-data;  //打包出的路径
    index index.html;
    error_page 404 /index.html; //根据vue-router路由history特性，这句一定要配置，否则会出现404问题
}
```

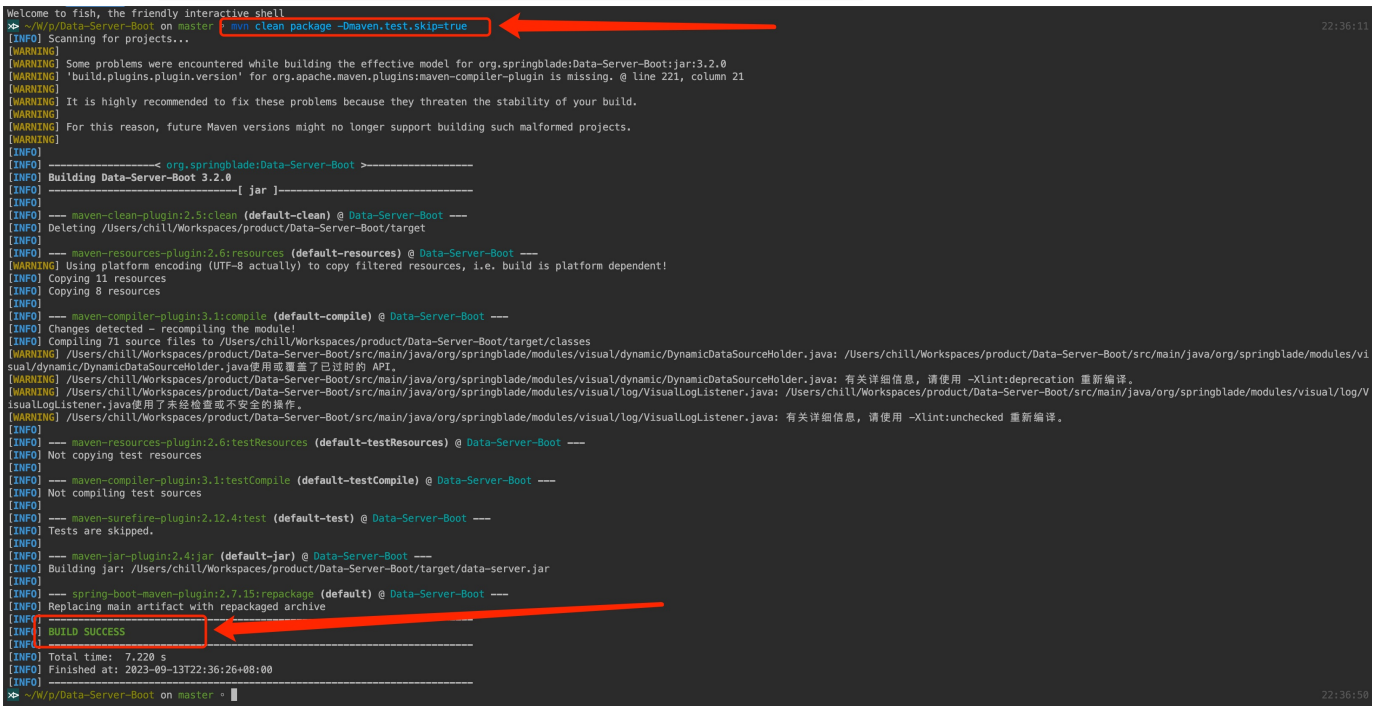
1.项目打包

一、后端打包

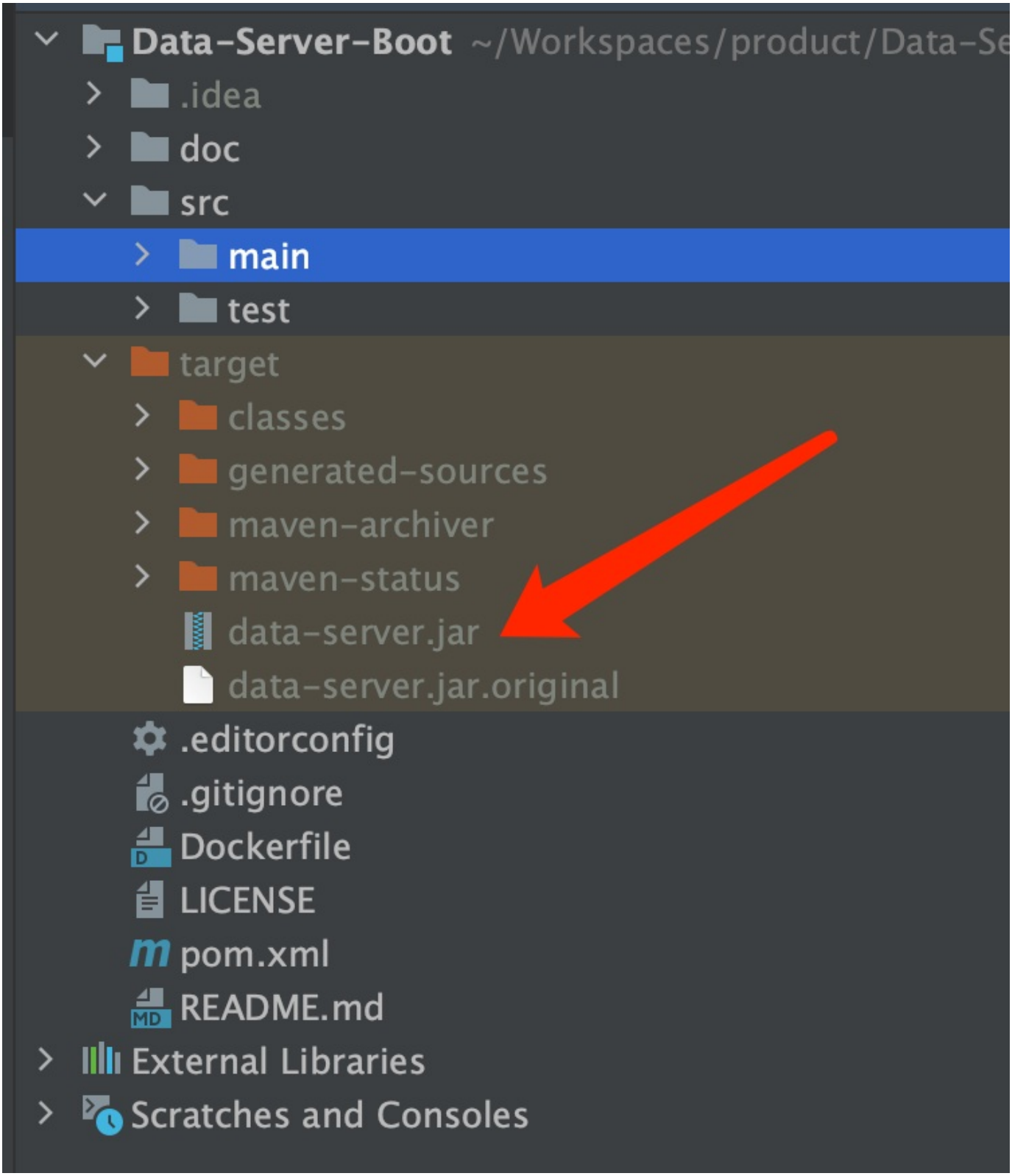
1. 将相关配置放到对应的文件



2. 根目录执行 `mvn clean package -Dmaven.test.skip=true`



3. 打包成功后我们便可以看到data-server.jar这个jar包，拷贝出来备用 ，重命名为 blade-visual.jar



二、前端打包

1. 找到 config.js 配置好服务器ip与端口到对应的接口url

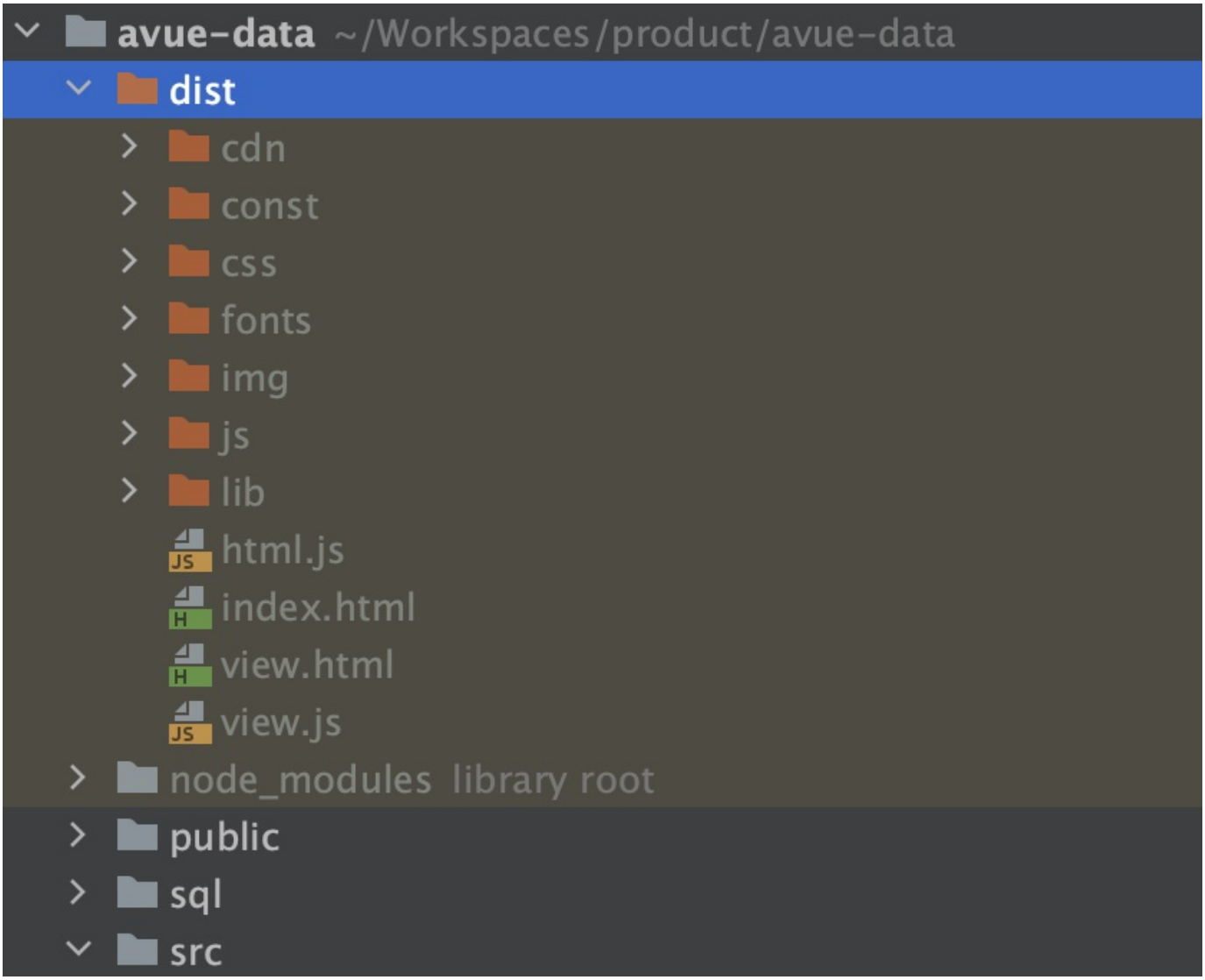
config.js

```
window.$website = {
  isDemo: false,
  isDemoTip: '演示环境不允许操作',
  title: 'BladeX 数据大屏',
  name: 'BladeX 数据大屏',
  subName: '可视化数据大屏（演示环境-请勿放生产数据）',
  url: 'http://服务地址:服务端口/blade-visual',
  autoSave: false,
  autoSaveTime: 60000,
  tabsList: [0, 1, 2, 3, 4, 5, 6],
  componentsList: [{
    name: 'test',
    component: 'testComponents',
    option: 'testOption',
    data: true
  }],
```

2. 根目录执行 `yarn run build`

```
Terminal:  Local  +
  dist/cdn/vue/2.5.2/vue.min.js                          91.47 KiB    33.30 KiB
  dist/cdn/vue-router/3.0.1/vue-router.min.js            23.98 KiB    8.40 KiB
  dist/cdn/axios/1.0.0/axios.min.js                      12.67 KiB    4.59 KiB
  dist/cdn/vuex/3.1.1/vuex.min.js                        11.38 KiB    3.16 KiB
  dist/cdn/vuex/2.4.1/vuex.min.js                        9.07 KiB     2.88 KiB
  dist/js/page.cfa5c502.js                               2234.95 KiB  620.20 KiB
  dist/cdn/element-ui/2.13.2/index.js                    553.87 KiB   141.08 KiB
  dist/js/chunk-vendors.bf41d7b1.js                      435.38 KiB   126.92 KiB
  dist/view.js                                           75.63 KiB    11.04 KiB
  dist/js/app.74c1de03.js                                19.39 KiB    4.96 KiB
  dist/js/chunk-2d0ae937.e9d597f6.js                     17.89 KiB    6.46 KiB
  dist/js/chunk-2d2308b7.bcae8994.js                     5.61 KiB     2.07 KiB
  dist/const/theme/halloween.project.js                  4.47 KiB     1.01 KiB
  dist/const/theme/wonderland.project.js                 4.32 KiB     0.92 KiB
  dist/const/theme/avue.project.js                       4.29 KiB     0.95 KiB
  dist/html.js                                           2.57 KiB     0.92 KiB
  dist/css/chunk-vendors.2406d2da.css                    242.71 KiB   36.25 KiB
  dist/cdn/element-ui/2.13.2/theme-chalk/index.css       226.98 KiB   34.55 KiB
  dist/cdn/avue/index.css                                74.14 KiB    12.54 KiB
  dist/cdn/animate/3.5.1/animate.css                     51.55 KiB    3.86 KiB
  dist/css/page.6c9fa4d7.css                             46.67 KiB    8.64 KiB
  dist/lib/index.css                                     4.98 KiB     1.37 KiB
  dist/css/app.29b235bd.css                              3.89 KiB     1.15 KiB

  Images and other types of assets omitted.

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html

✨  Done in 47.60s.
```

3. 打包成功后，我们便可以看到 `dist` 文件夹下的目录，拷贝出来备用


avue-data ~/Workspaces/product/avue-data
dist
cdn
const
css
fonts
img
js
lib
html.js
index.html
view.html
view.js
node_modules library root
public
sql
src

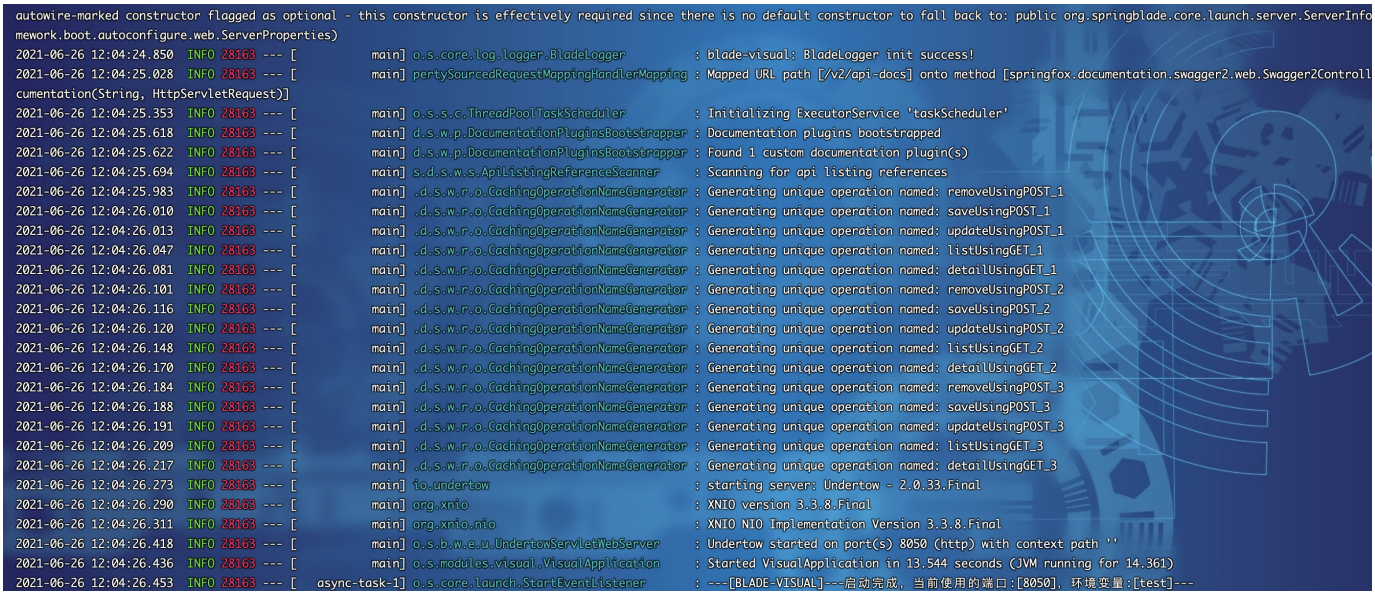
2.Linux部署

一、服务器准备

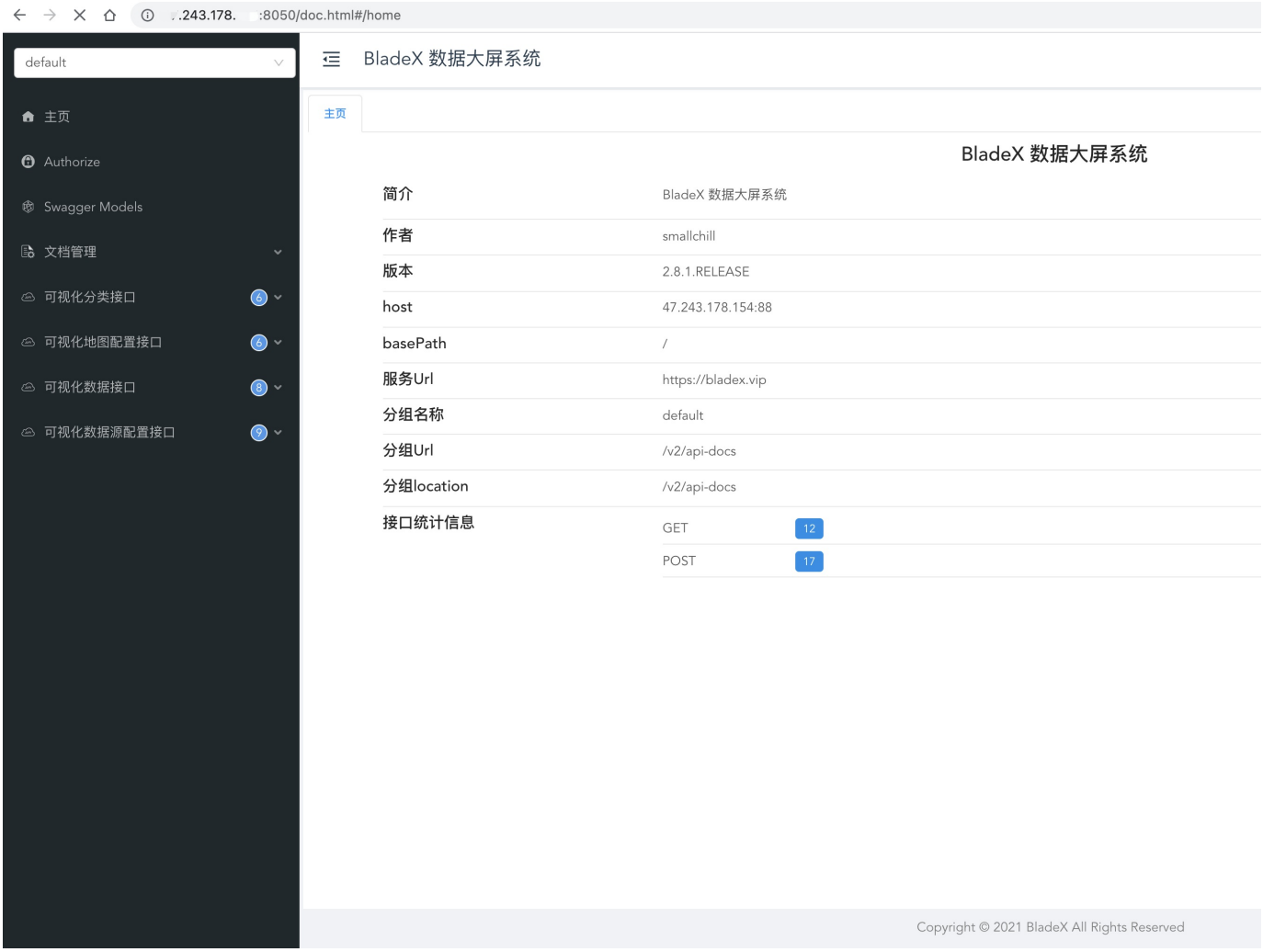
1. 服务器上部署好mysql、redis、minio(若采用alioss或qiniu可以略过)、java，此处不再赘述

二、后端部署

1. 将上一章节打包好的jar包上传，注意要提前将服务器的配置加上
2. 执行命令 java -jar blade-visual.jar --spring.profiles.active=test 启动



3. 后端启动成功后访问对应地址，若显示如下界面则说明后端启动成功



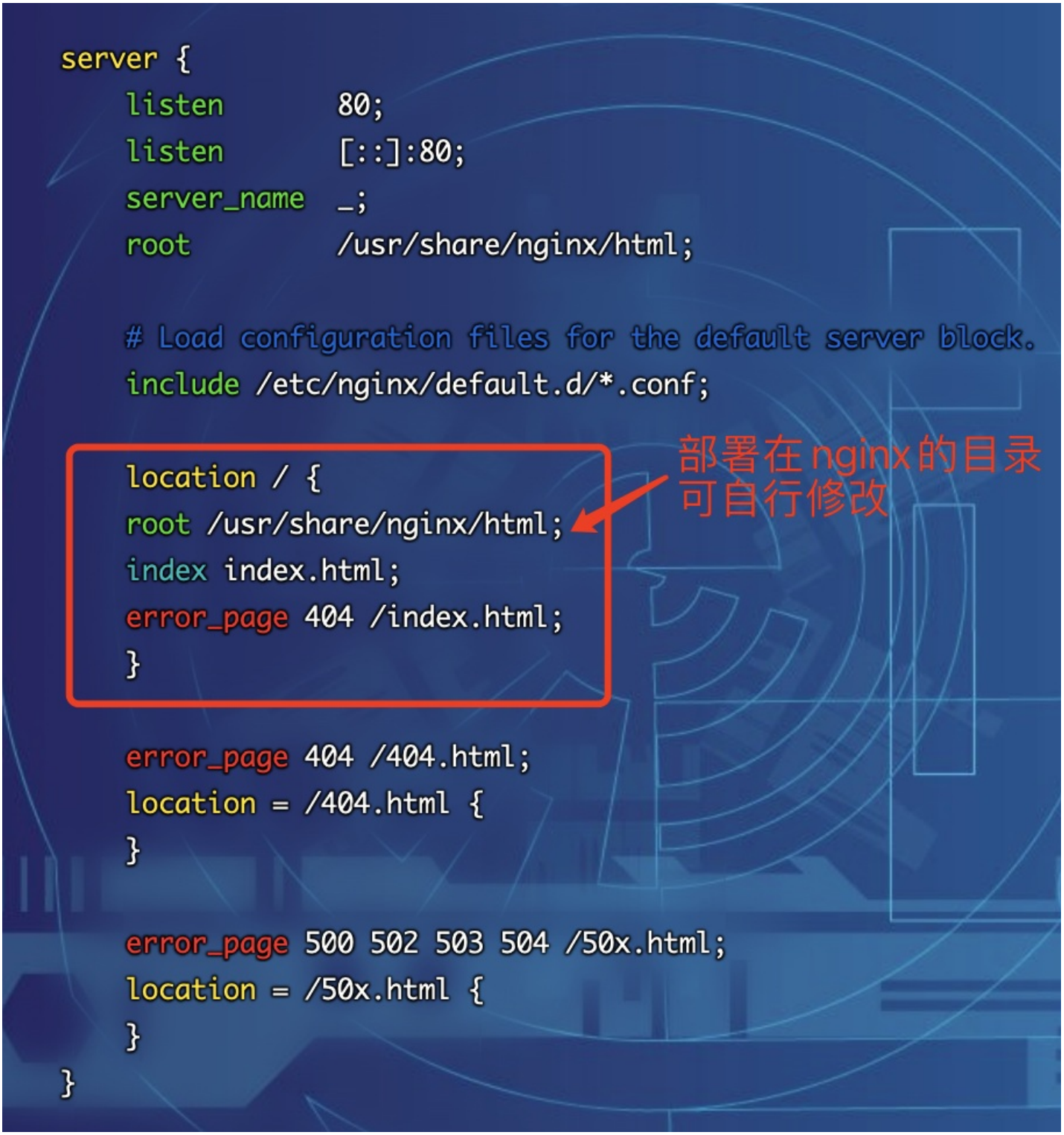
三、前端部署

1. 将上一章节打包好的dist上传，注意要提前配置好后端api地址

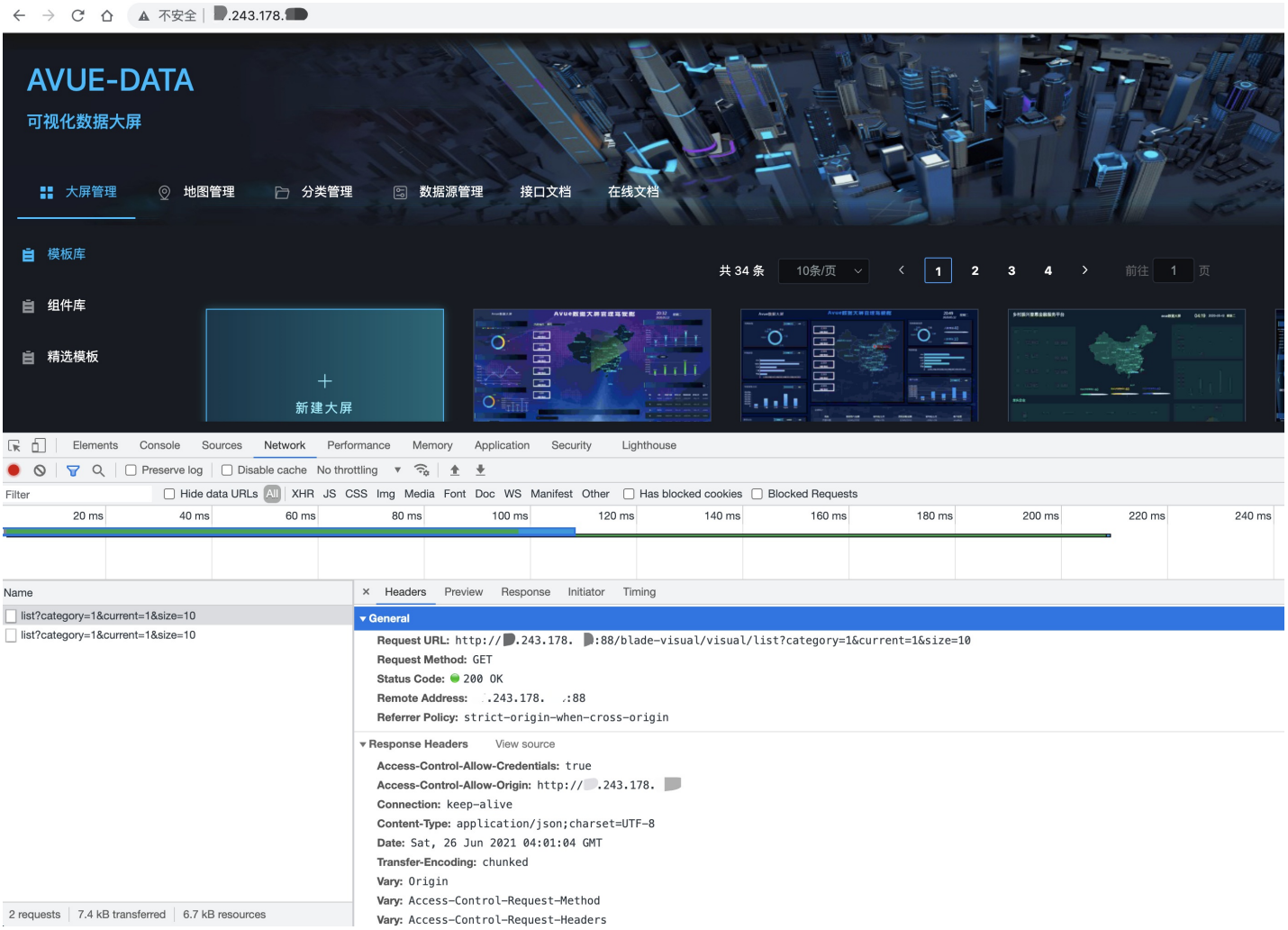
```
[root@test /]# cd /usr/share/nginx/html/
[root@test html]# ls
cdn  const  css  fonts  html.js  img  index.html  js  lib  view.html  view.js
```

2. 到nginx配置文件进行配置增加如下配置，否则会出现404

```
location /{
    root /usr/share/nginx/html;
    index index.html;
    error_page 404 /index.html;
}
```



3. 打开服务器地址，查看结果，可以正常访问，说明部署成功


不安全
.243.178.
AVUE-DATA
可视化数据大屏
大屏管理
地图管理
分类管理
数据源管理
接口文档
在线文档
模板库
组件库
精选模板
共 34 条
10条/页
1
2
3
4
前往
1
页
新建大屏
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Preserve log
Disable cache
No throttling
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Has blocked cookies
Blocked Requests
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
140 ms
160 ms
180 ms
200 ms
220 ms
240 ms
Name
list?category=1&current=1&size=10
list?category=1&current=1&size=10
Headers
Preview
Response
Initiator
Timing
General
Request URL: http://.243.178. :88/blade-visual/visual/list?category=1&current=1&size=10
Request Method: GET
Status Code: 200 OK
Remote Address: .243.178. :88
Referrer Policy: strict-origin-when-cross-origin
Response Headers
View source
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://.243.178.
Connection: keep-alive
Content-Type: application/json;charset=UTF-8
Date: Sat, 26 Jun 2021 04:01:04 GMT
Transfer-Encoding: chunked
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
2 requests
7.4 kB transferred
6.7 kB resources

3.Baota部署

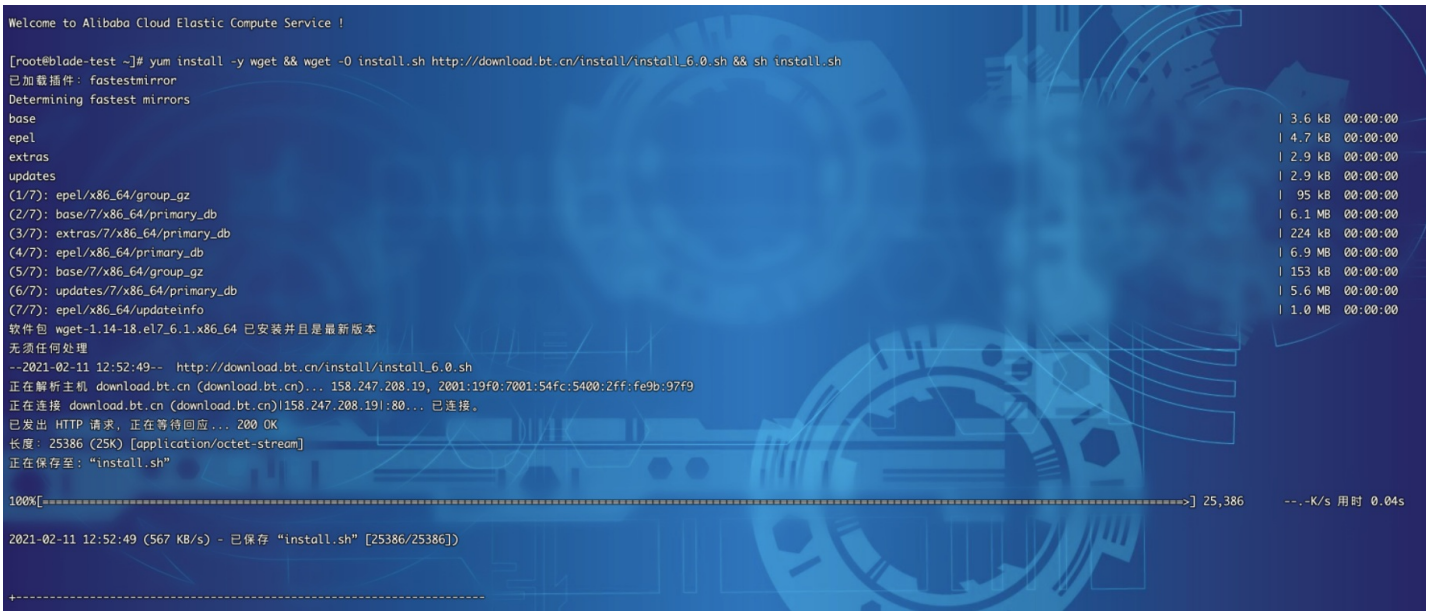
3.1安装宝塔

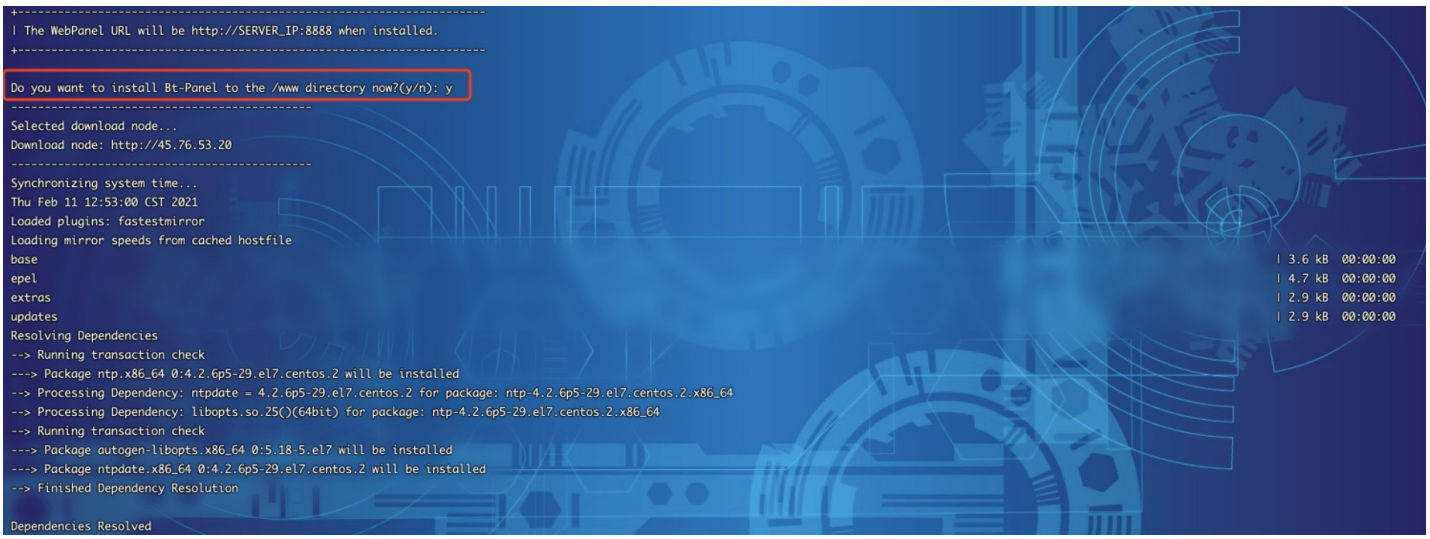
一、开始安装

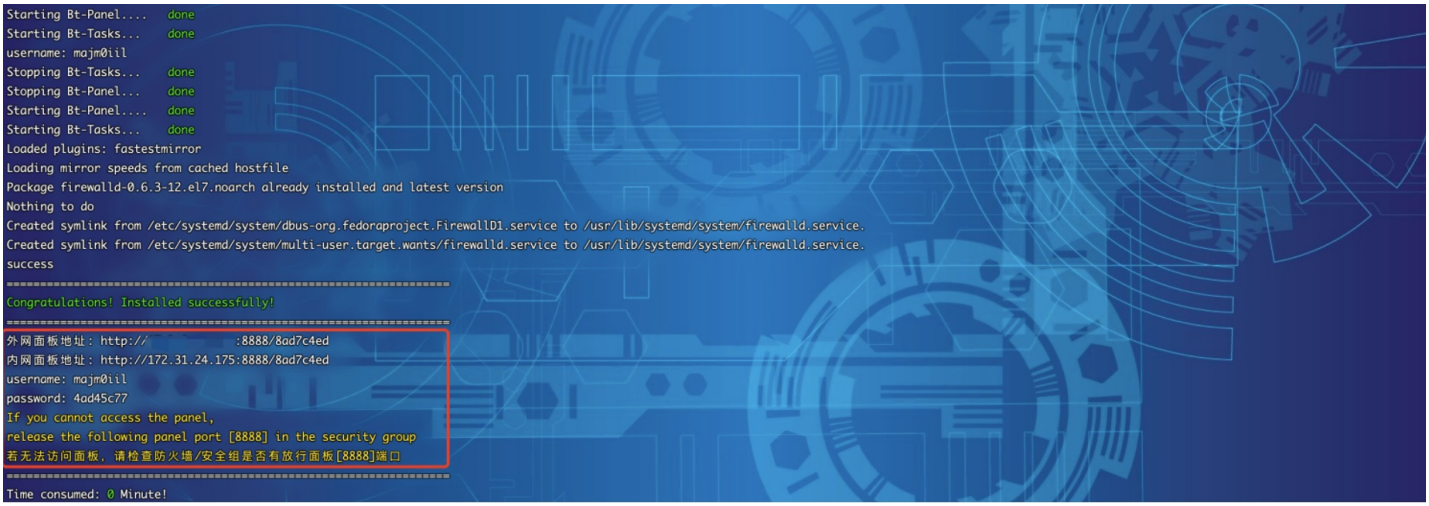
> 注意⚠：宝塔部署需要一个纯净的操作系统，切勿在已有环境尤其是生产服务器上安装！

1. 登录已经准备好纯净系统的服务器, 请自行选择熟悉的软件连接登录
2. 执行如下命令自动安装

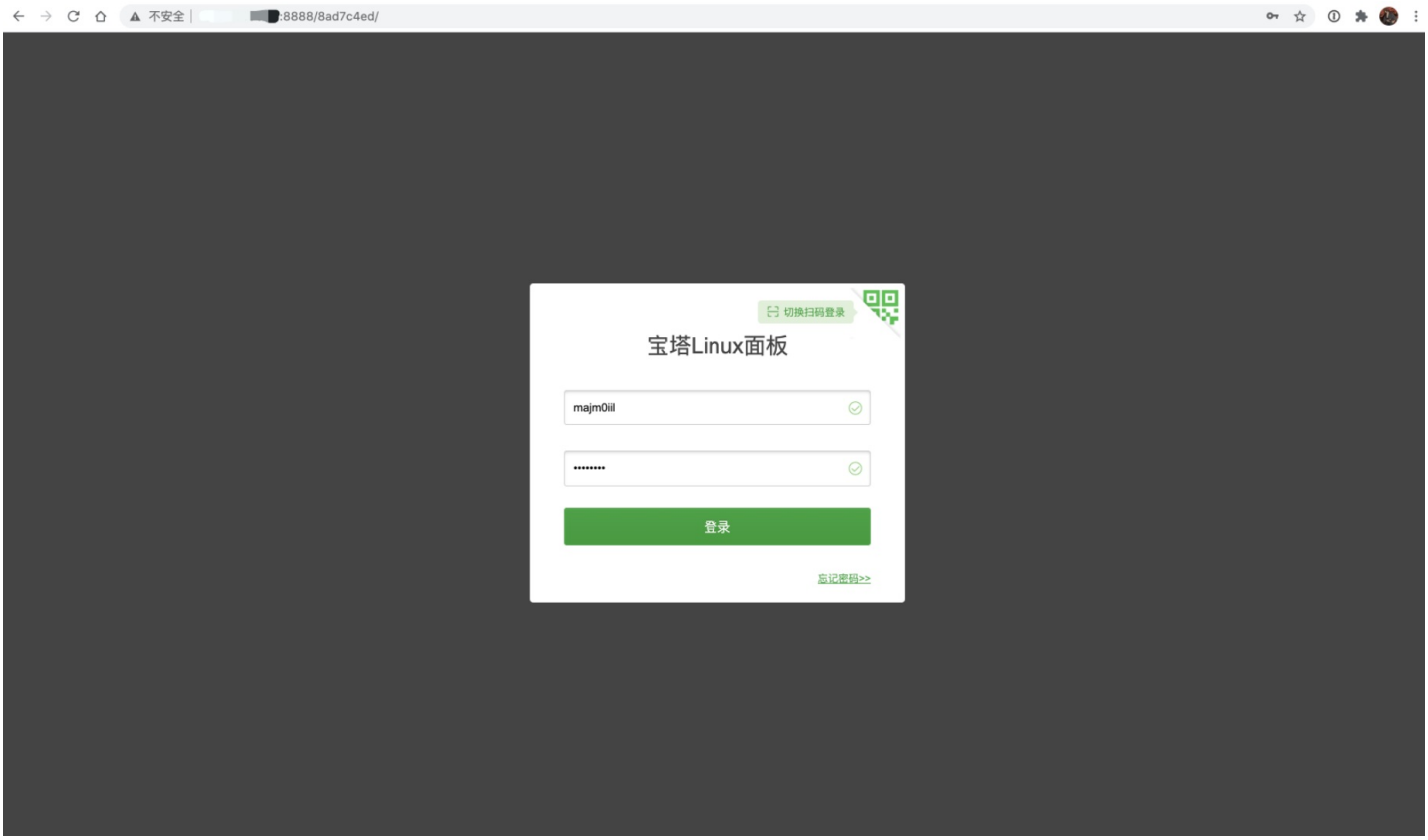
```
[root@blade-test ~]# yum install -y wget && wget -O install.sh http://downl
oad.bt.cn/install/install_6.0.sh && sh install.sh
```





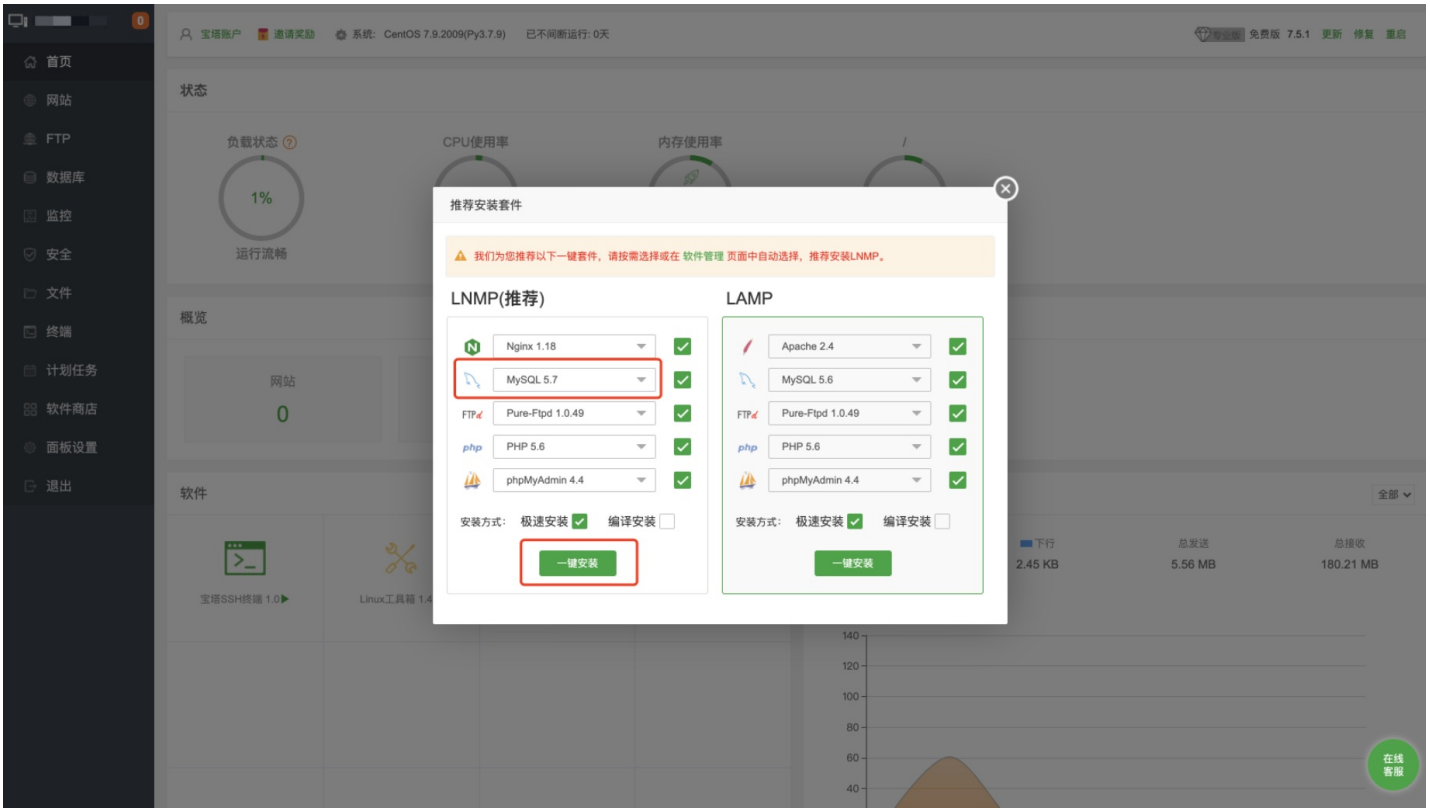


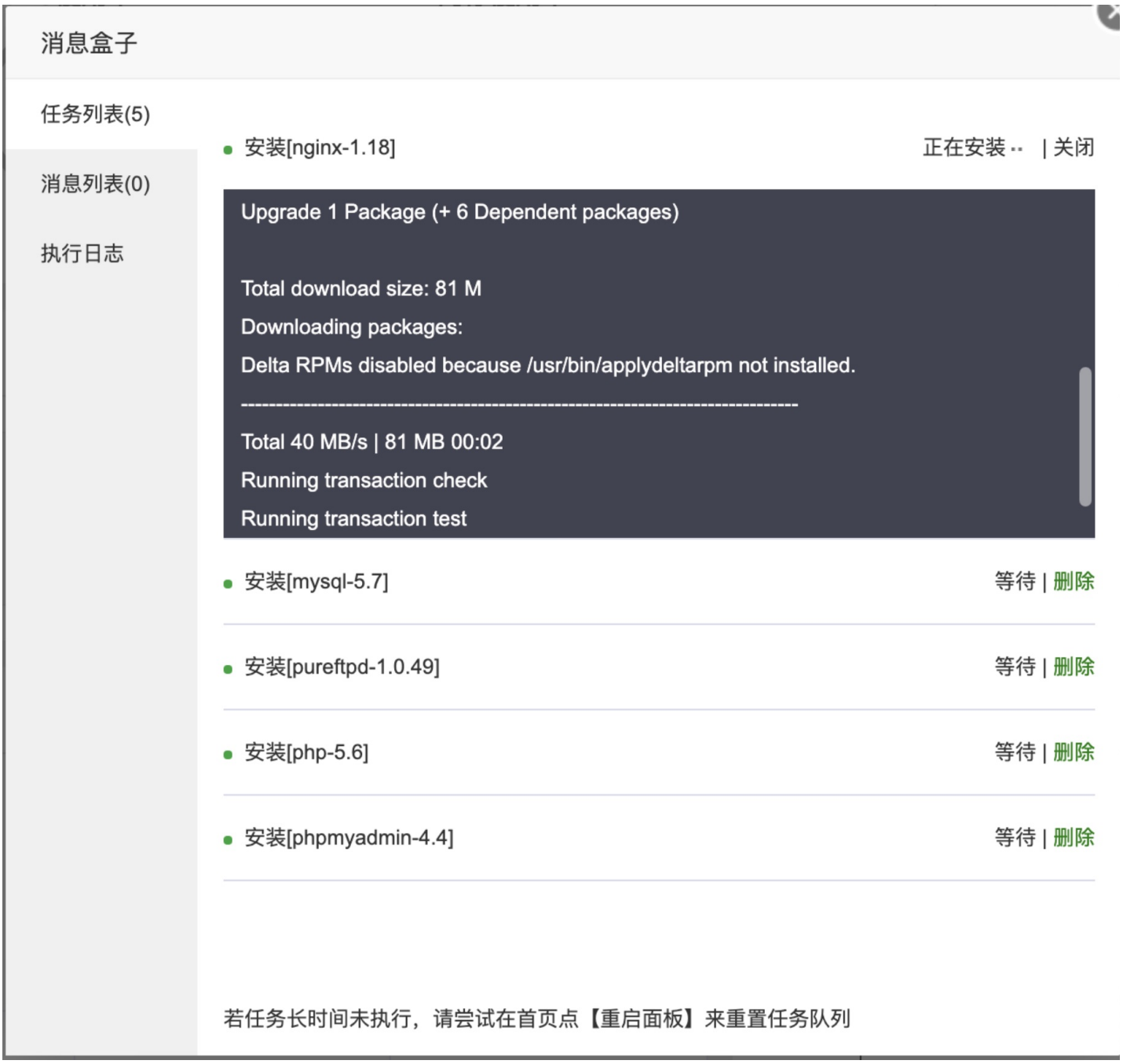
3. 注意记录最底部的面板地址、username、password，不要丢失，然后访问地址输入帐号密码进行登录



4. 初次登陆也许会相对耗时，此时系统在初始化，稍等片刻就可以看到进入了面板欢迎页，选择同意并进入面板

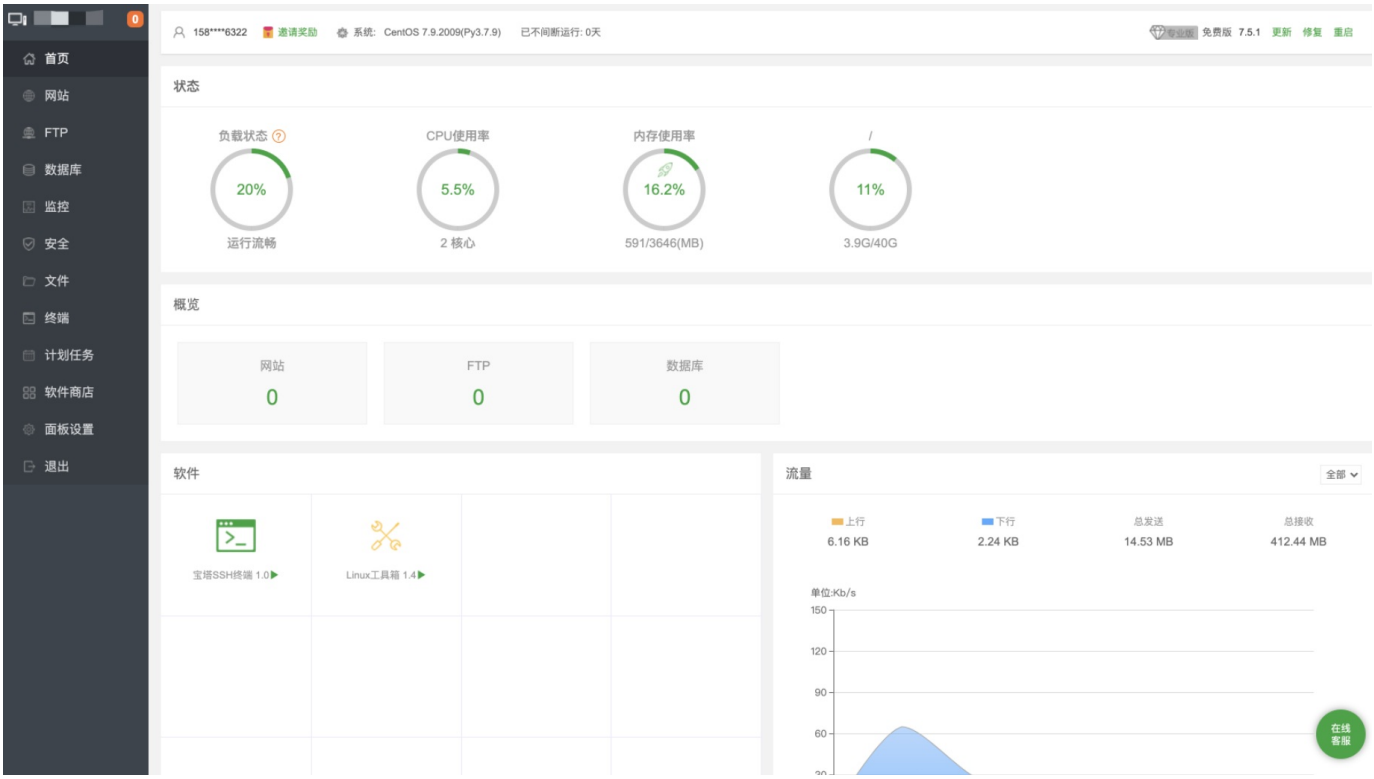
5. 进入首页后会弹出默认环境安装，我们选择左侧的一键套件，选择mysql5.7，点击一键安装后耐心等待安装完毕


首页
网站
FTP
数据库
监控
安全
文件
终端
计划任务
软件商店
面板设置
退出
状态
负载状态
1%
运行流畅
CPU使用率
内存使用率
概览
网站
0
软件
推荐安装套件
我们为您推荐以下一键套件，请按需选择或在 软件管理 页面中自动选择，推荐安装LNMP。
LNMP(推荐)
Nginx 1.18
MySQL 5.7
Pure-Ftpd 1.0.49
PHP 5.6
phpMyAdmin 4.4
安装方式：
极速安装
编译安装
一键安装
LAMP
Apache 2.4
MySQL 5.6
Pure-Ftpd 1.0.49
PHP 5.6
phpMyAdmin 4.4
安装方式：
极速安装
编译安装
一键安装
下行
2.45 KB
总发送
5.56 MB
总接收
180.21 MB
在线客服



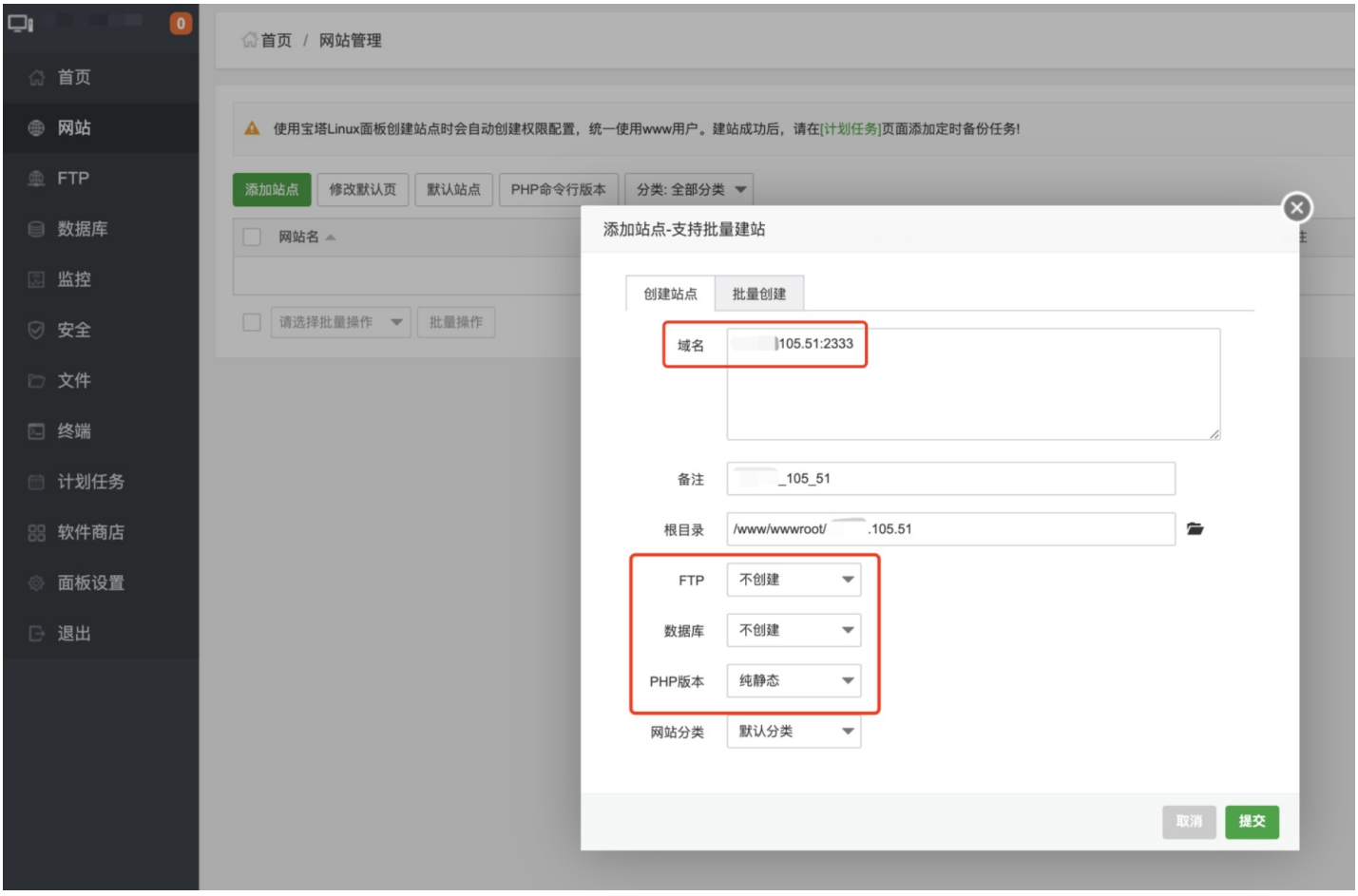
6. 绑定宝塔官方的帐号，若未注册可访问这个链接：https://www.bt.cn/?invite_code=MV9namxtdXM=

7. 完整首页，看到这一步就说明可以了



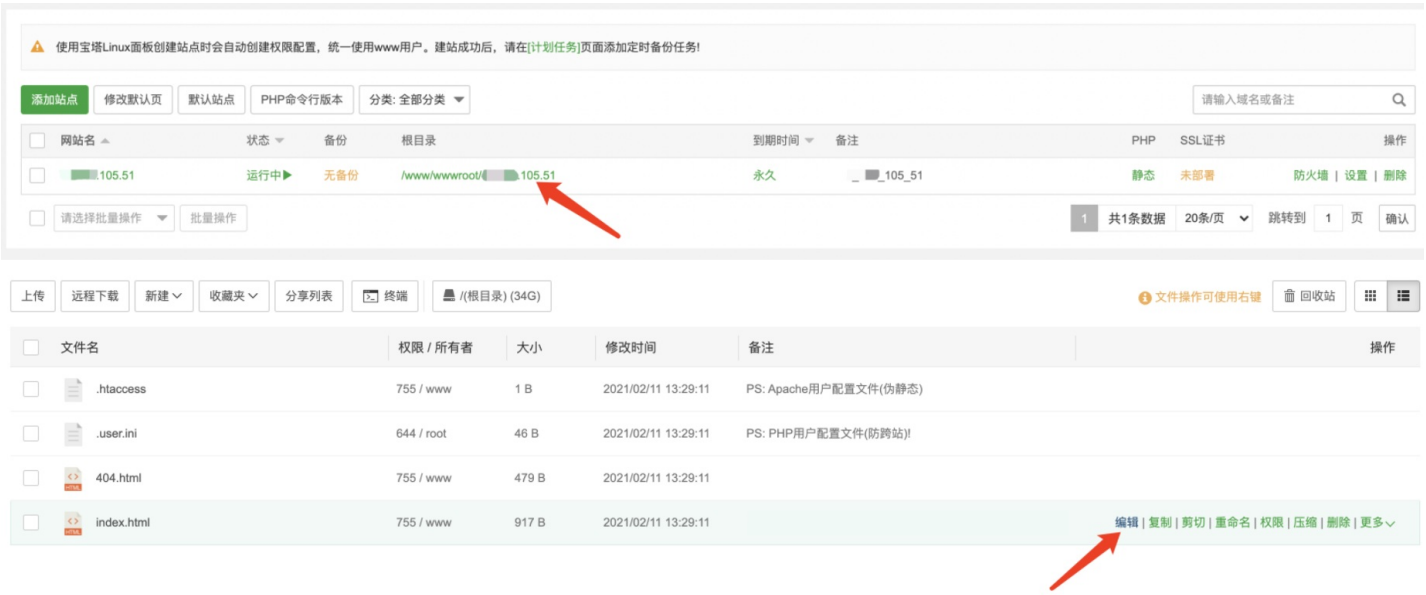
二、创建站点

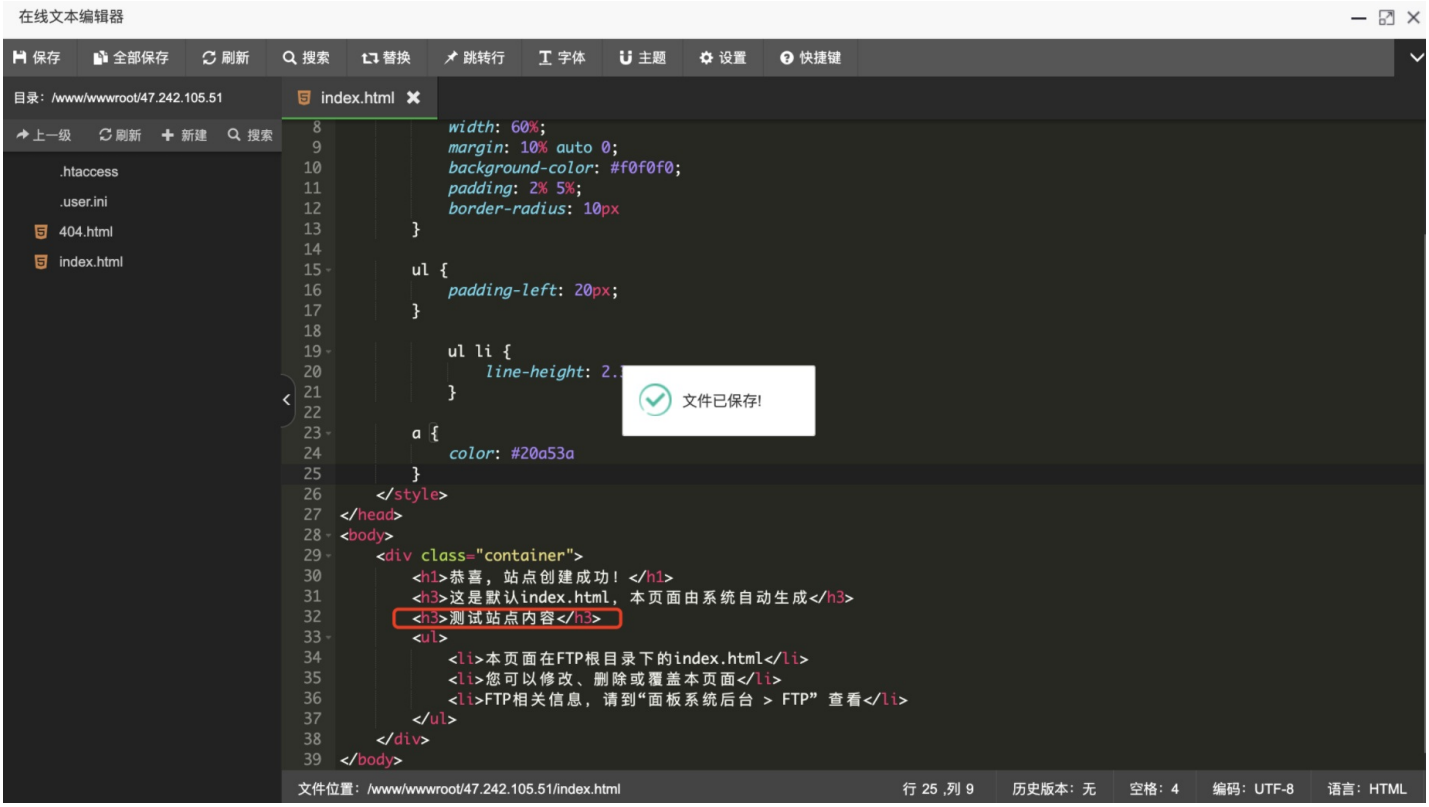
1. 宝塔基础环境安装完毕，下面我们来尝试创建一个站点，用于给后续的前端部署做准备
2. 我们先创建一个站点，格式为外网ip+端口，注意到云服务器的安全组规则给端口开放访问权限



3. 访问设定的ip和端口，可以看到显示创建成功

4. 点进根目录，找到index.html，修改一下内容





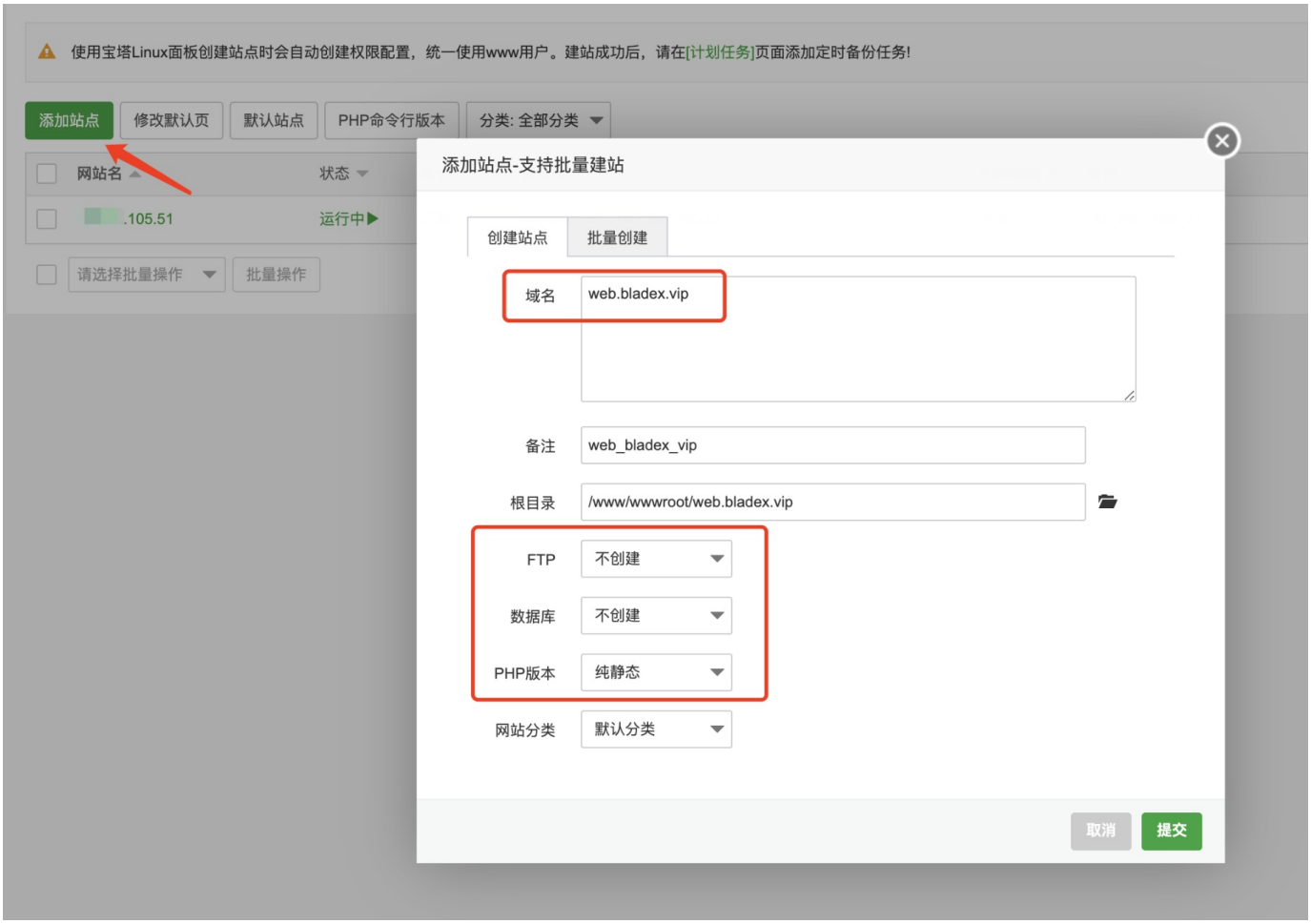
5. 刷新界面，发现刚添加的信息已经显示成功，说明站点创建成功

三、部署域名

- 很多时候项目的正式环境都是用于域名访问
- 使用域名的同时也会使用https的形式
- 下面我们演示如何部署一个域名并且配置免费https证书

四、新建站点

1. 给站点起名使用域名形式，这里我们写为 `web.bladex.vip` 并且给域名做好映射



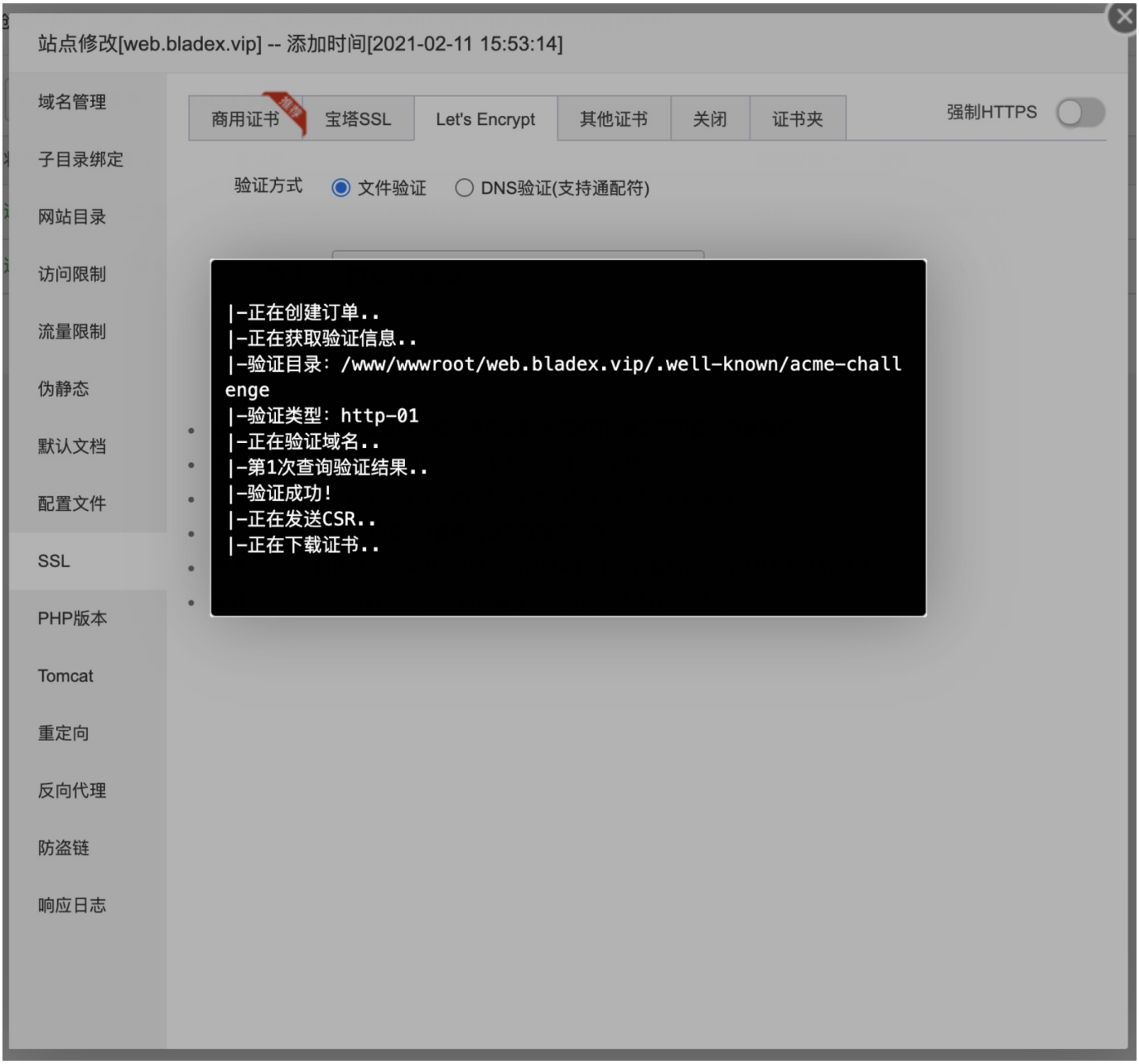
2. 稍等片刻，访问http://web.bladex.vip，可以看到域名访问生效

```
web.bladex.vip
```

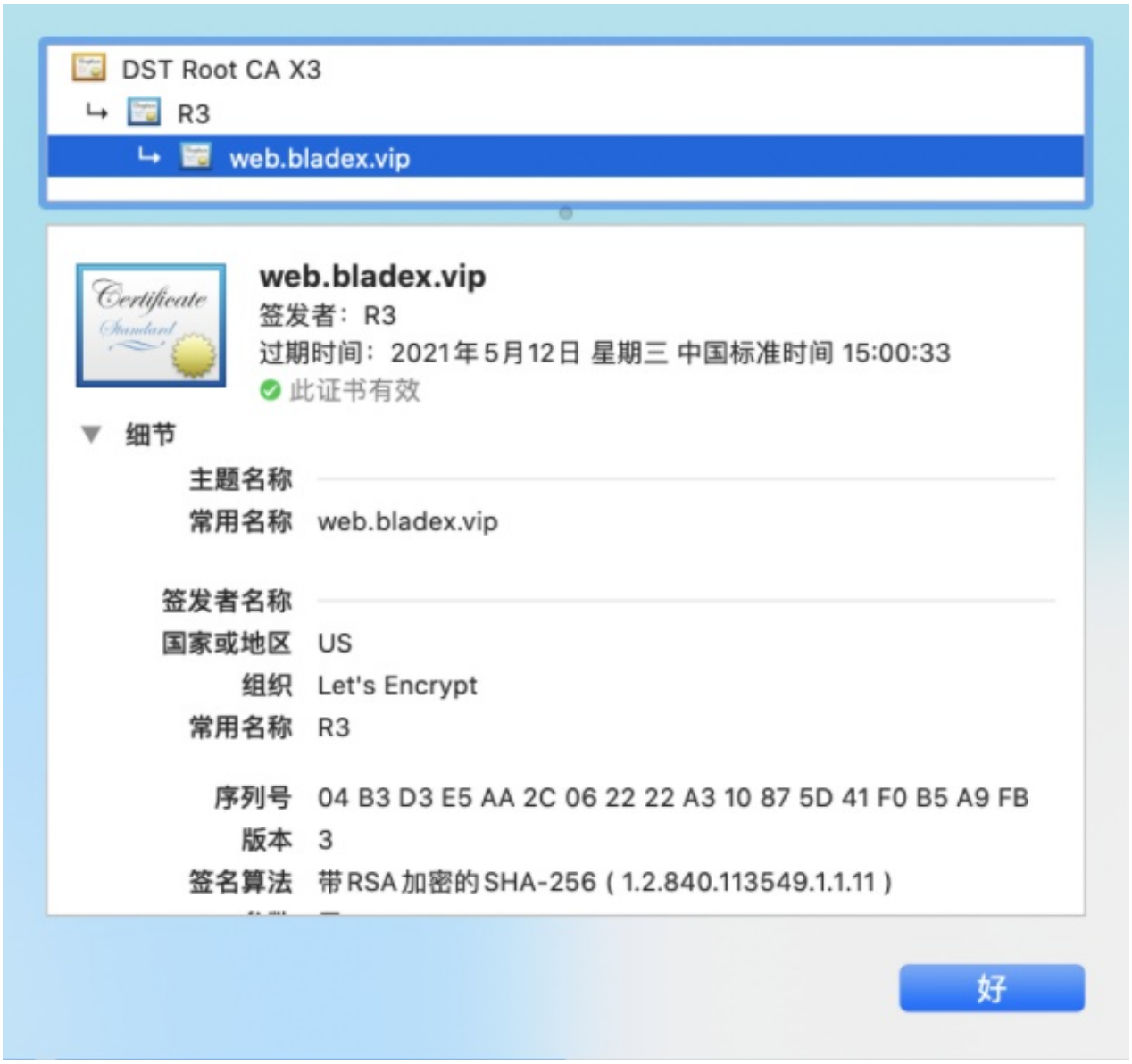
五、配置Https

1. 打开SSL栏目，点击免费的Let's Encrypt，选中域名并点击申请


站点修改[web.bladex.vip] -- 添加时间[2021-02-11 15:53:14]
域名管理
子目录绑定
网站目录
访问限制
流量限制
伪静态
默认文档
配置文件
SSL
PHP版本
Tomcat
重定向
反向代理
防盗链
响应日志
商用证书
推荐
宝塔SSL
Let's Encrypt
其他证书
关闭
证书夹
强制HTTPS
验证方式
文件验证
DNS验证(支持通配符)
域名
web.bladex.vip
申请
Let's Encrypt因更换根证书，部分老旧设备访问时可能提示不可信，考虑购买[商用SSL证书]
申请之前，请确保域名已解析，如未解析会导致审核失败
Let's Encrypt免费证书，有效期3个月，支持多域名。默认会自动续签
若您的站点使用了CDN或301重定向会导致续签失败
在未指定SSL默认站点时,未开启SSL的站点使用HTTPS会直接访问到已开启SSL的站点
如开启后无法使用HTTPS访问，请检查安全组是否正确放行443端口


站点修改[web.bladex.vip] -- 添加时间[2021-02-11 15:53:14]
域名管理
子目录绑定
网站目录
访问限制
流量限制
伪静态
默认文档
配置文件
SSL
PHP版本
Tomcat
重定向
反向代理
防盗链
响应日志
商用证书
推荐
宝塔SSL
Let's Encrypt
其他证书
关闭
证书夹
强制HTTPS
验证方式
文件验证
DNS验证(支持通配符)
|-正在创建订单..
|-正在获取验证信息..
|-验证目录: /www/wwwroot/web.bladex.vip/.well-known/acme-challenge
|-验证类型: http-01
|-正在验证域名..
|-第1次查询验证结果..
|-验证成功!
|-正在发送CSR..
|-正在下载证书..

2. 再次刷新https://web.bladex.vip，可以正确访问，并且可以看到证书信息

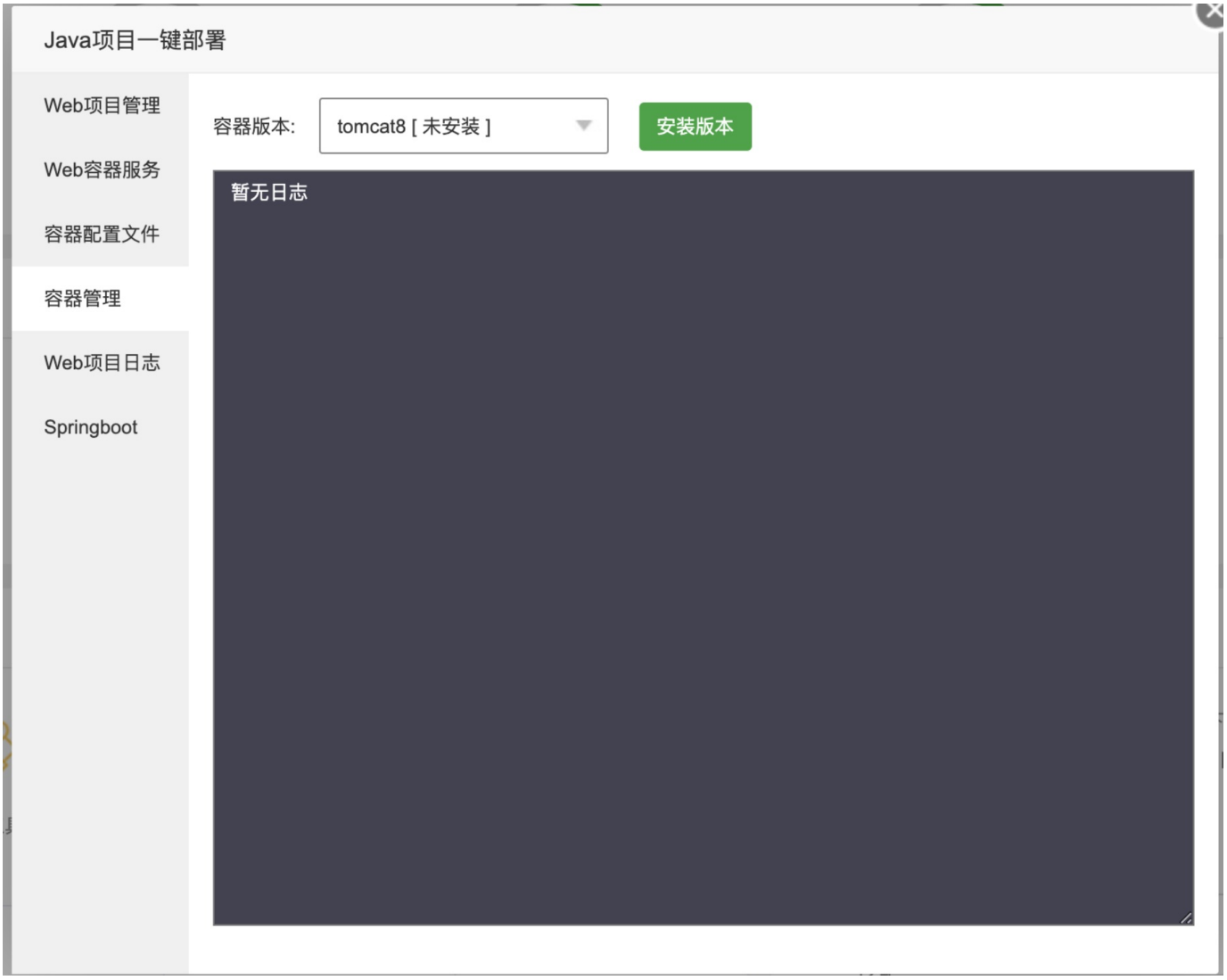

DST Root CA X3
R3
web.bladex.vip
web.bladex.vip
签发者：R3
过期时间：2021年5月12日 星期三 中国标准时间 15:00:33
此证书有效
细节
主题名称
常用名称 web.bladex.vip
签发者名称
国家或地区 US
组织 Let's Encrypt
常用名称 R3
序列号 04 B3 D3 E5 AA 2C 06 22 22 A3 10 87 5D 41 F0 B5 A9 FB
版本 3
签名算法 带RSA加密的SHA-256 (1.2.840.113549.1.1.11)
好

3.2基础部署

一、安装Java环境

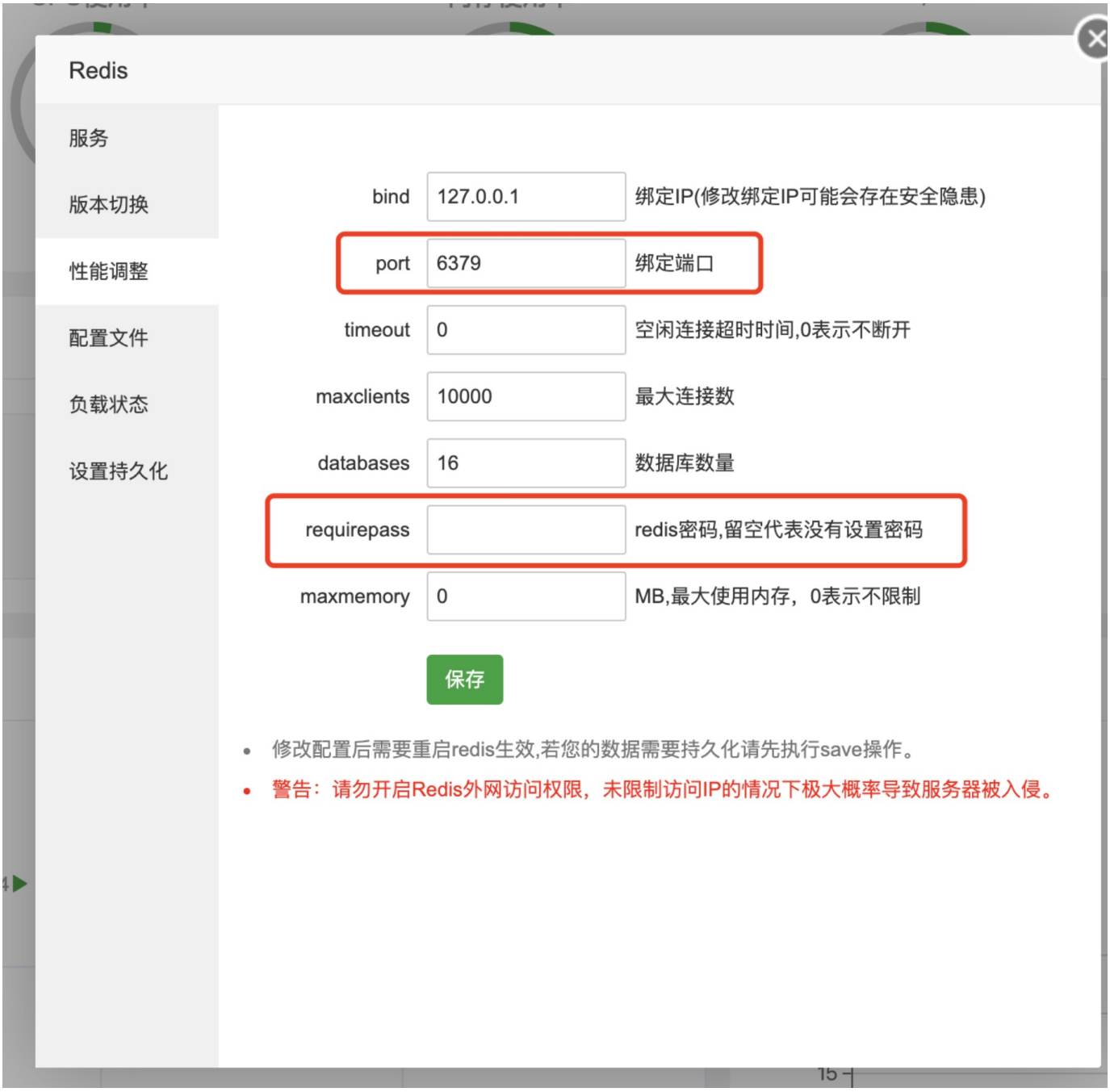
1. 前往软件商店搜索 `java` 点击安装，并首页显示

2. 点击设置，选择安装tomcat8服务，这里主要是为了安装java8的环境


Java项目一键部署
Web项目管理
Web容器服务
容器配置文件
容器管理
Web项目日志
Springboot
容器版本:
tomcat8 [未安装]
安装版本
暂无日志

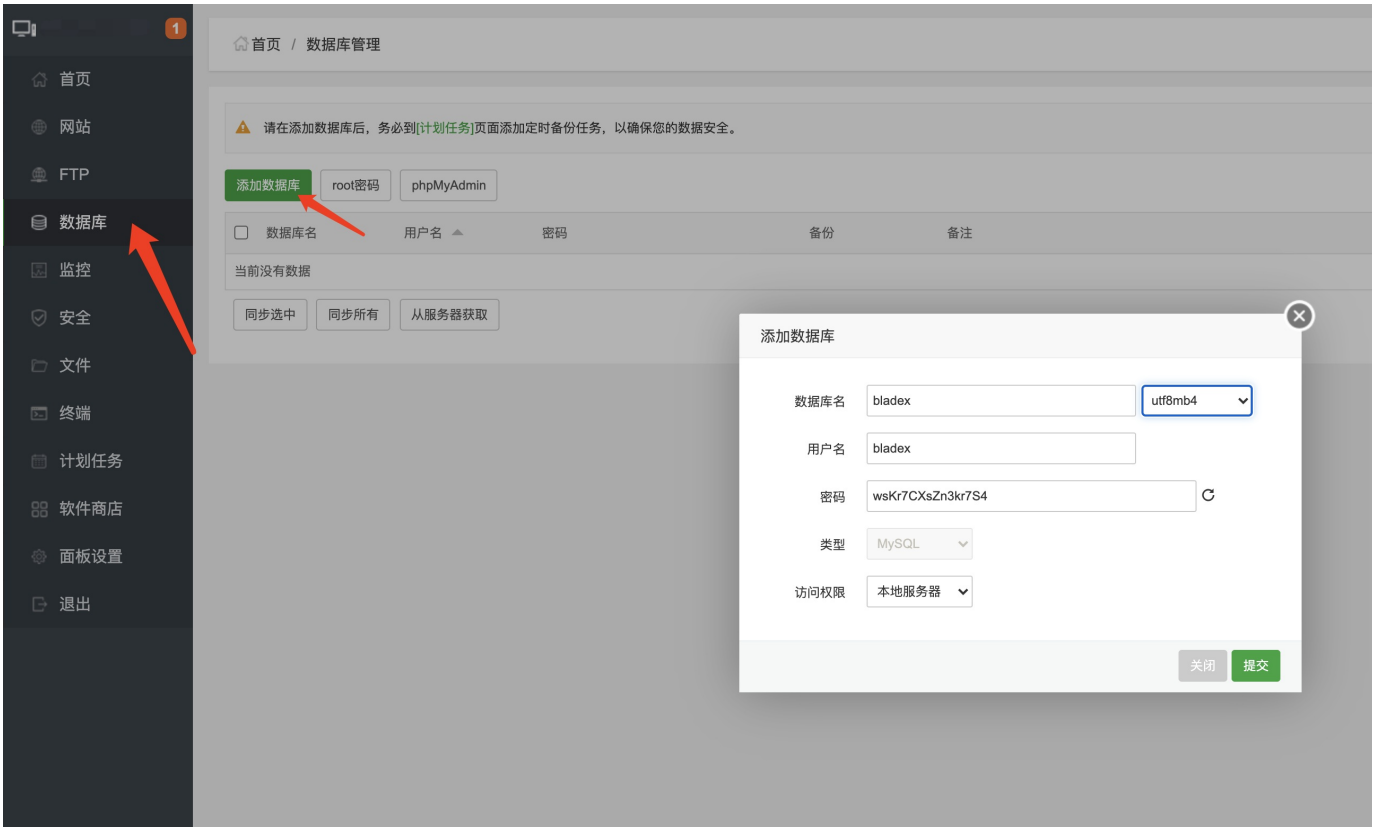
二、安装Redis环境

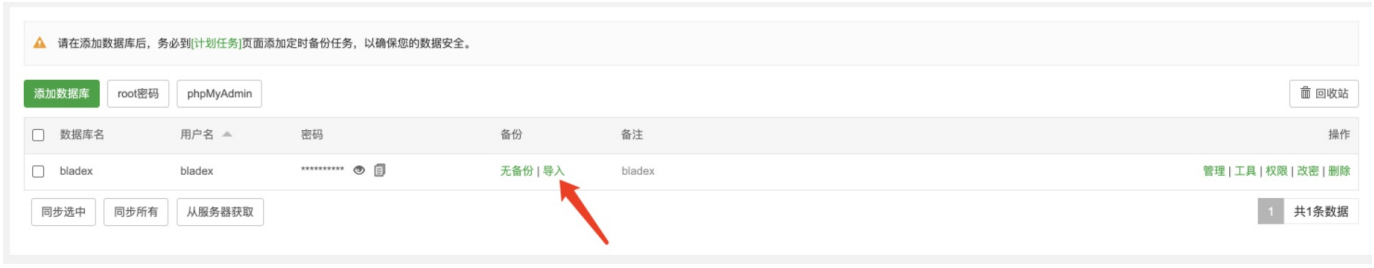
1. 前往软件商店搜索 `redis` 点击安装，并首页显示，若是生产环境，建议修改端口以及密码，并且不开放至外网访问

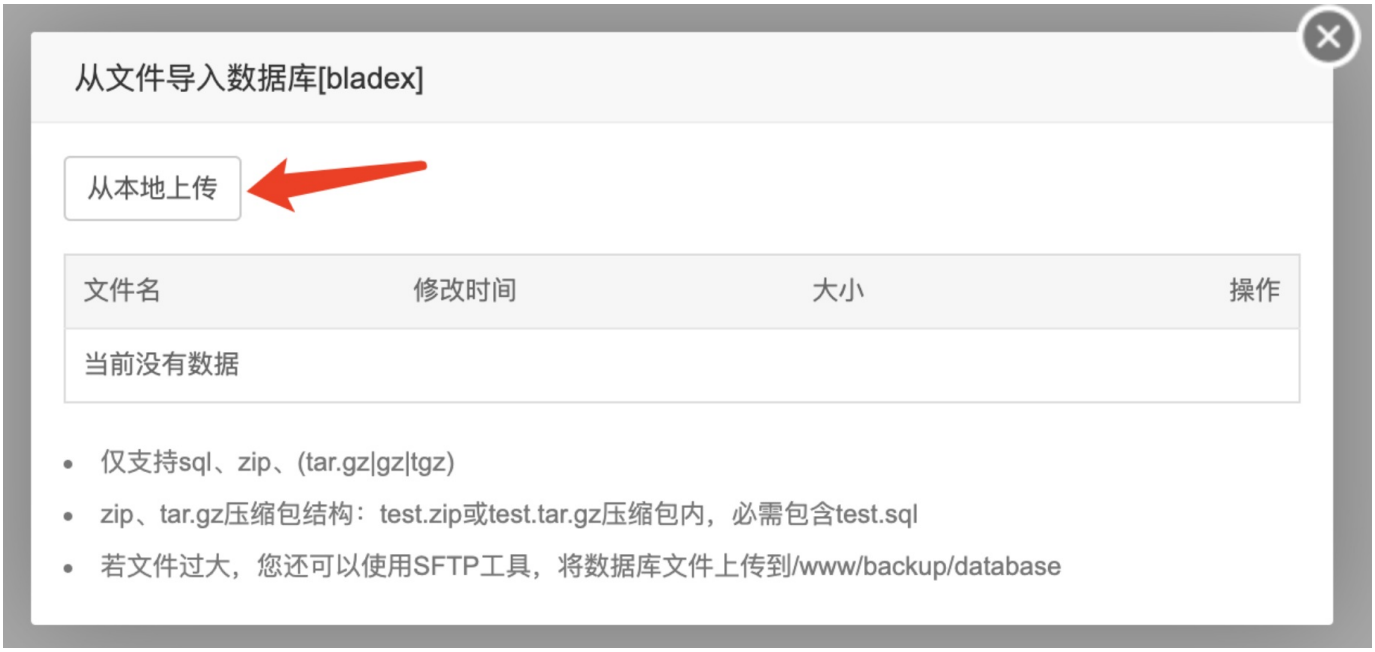


三、配置Mysql

1. 创建bladex数据库，导入脚本，并将帐号密码记录好后续放到 `blade-visual` 的 `application-test.yml` 配置文件


首页
网站
FTP
数据库
监控
安全
文件
终端
计划任务
软件商店
面板设置
退出
首页 / 数据库管理
请在添加数据库后，务必到[计划任务]页面添加定时备份任务，以确保您的数据安全。
添加数据库
root密码
phpMyAdmin
数据库名
用户名
密码
备份
备注
当前没有数据
同步选中
同步所有
从服务器获取
添加数据库
数据库名
bladex
utf8mb4
用户名
bladex
密码
wsKr7CXsZn3kr7S4
类型
MySQL
访问权限
本地服务器
关闭
提交


请在添加数据库后，务必到[计划任务]页面添加定时备份任务，以确保您的数据安全。
添加数据库
root密码
phpMyAdmin
回收站
数据库名
用户名
密码
备份
备注
操作
bladex
bladex
无备份 | 导入
bladex
管理 | 工具 | 权限 | 改密 | 删除
同步选中
同步所有
从服务器获取
1
共1条数据

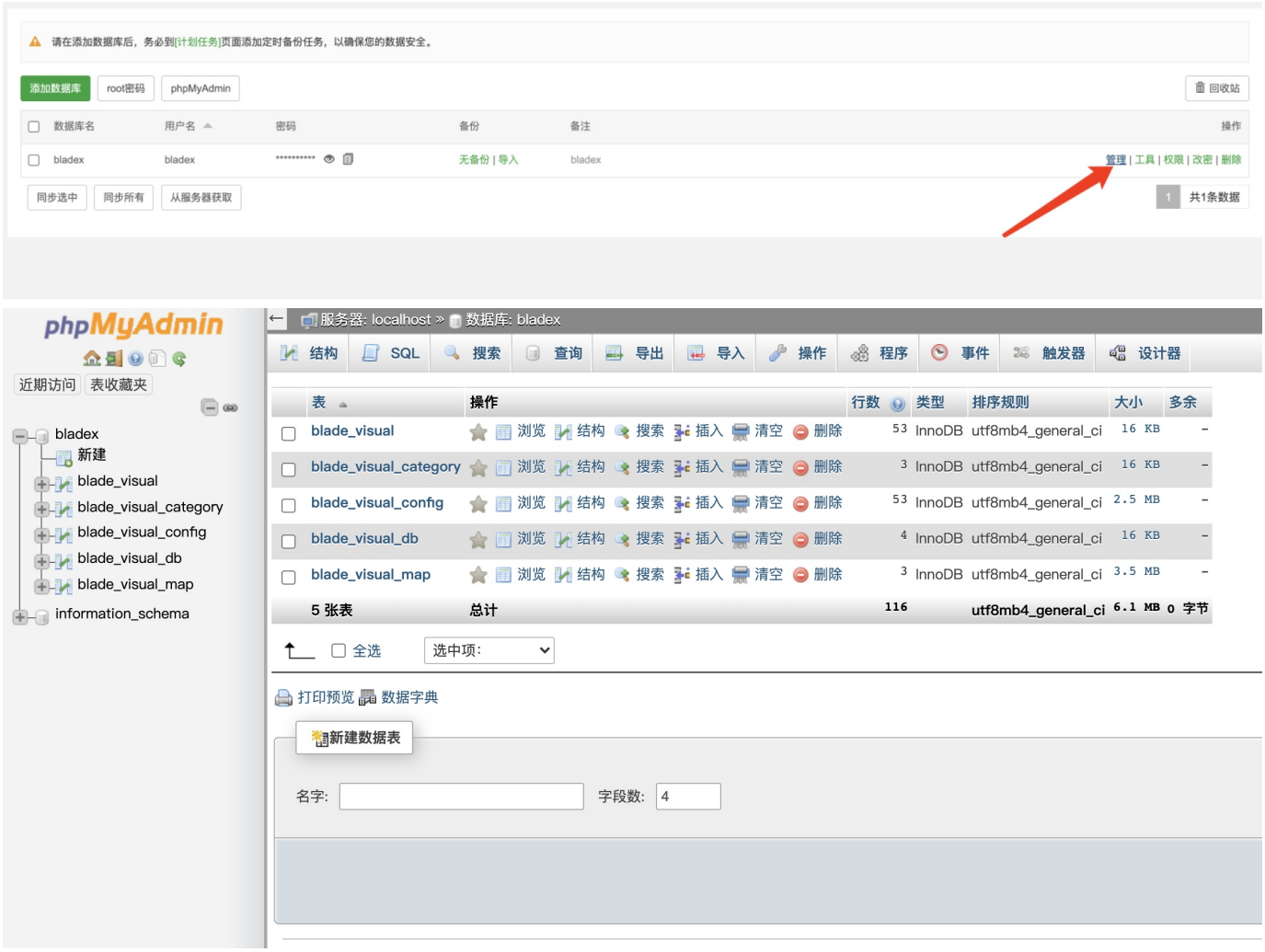

从文件导入数据库[bladex]
从本地上传
文件名
修改时间
大小
操作
当前没有数据
仅支持sql、zip、(tar.gz|gz|tgz)
zip、tar.gz压缩包结构：test.zip或test.tar.gz压缩包内，必需包含test.sql
若文件过大，您还可以使用SFTP工具，将数据库文件上传到/www/backup/database


码
备份
备注
****
上传文件到[/www/backup/database/] --- 请上传sql或zip或tar.gz压缩包
选择文件
请将需要上传的文件拖到此处
关闭
开始上传


备份
备注
上传文件到[/www/backup/database/] --- 请上传sql或zip或tar.gz压缩包
选择文件
bladex_visual_mysql.sql
4.01 MB
等待上传
关闭
开始上传


从文件导入数据库[bladex]
从本地上传
文件名
修改时间
大小
操作
bladex_visual_mysql.sql
2021/06/26 11:28:28
4.01 MB
导入 | 删除
仅支持sql、zip、(tar.gz|gz|tgz)
zip、tar.gz压缩包结构：test.zip或test.tar.gz压缩包内，必需包含test.sql
若文件过大，您还可以使用SFTP工具，将数据库文件上传到/www/backup/database

2. 点击管理，查看数据是否导入成功



注意

- java、redis、mysql三个基础环境是必须要的，其中尤其是redis与mysql，务必确认设置密码并且日常关闭外网访问
- 若都采用默认配置，不出意外，服务器很快就会被人挂上挖矿病毒，这个请再三确认，不希望大家的服务器

因此挂掉

3.3大屏部署

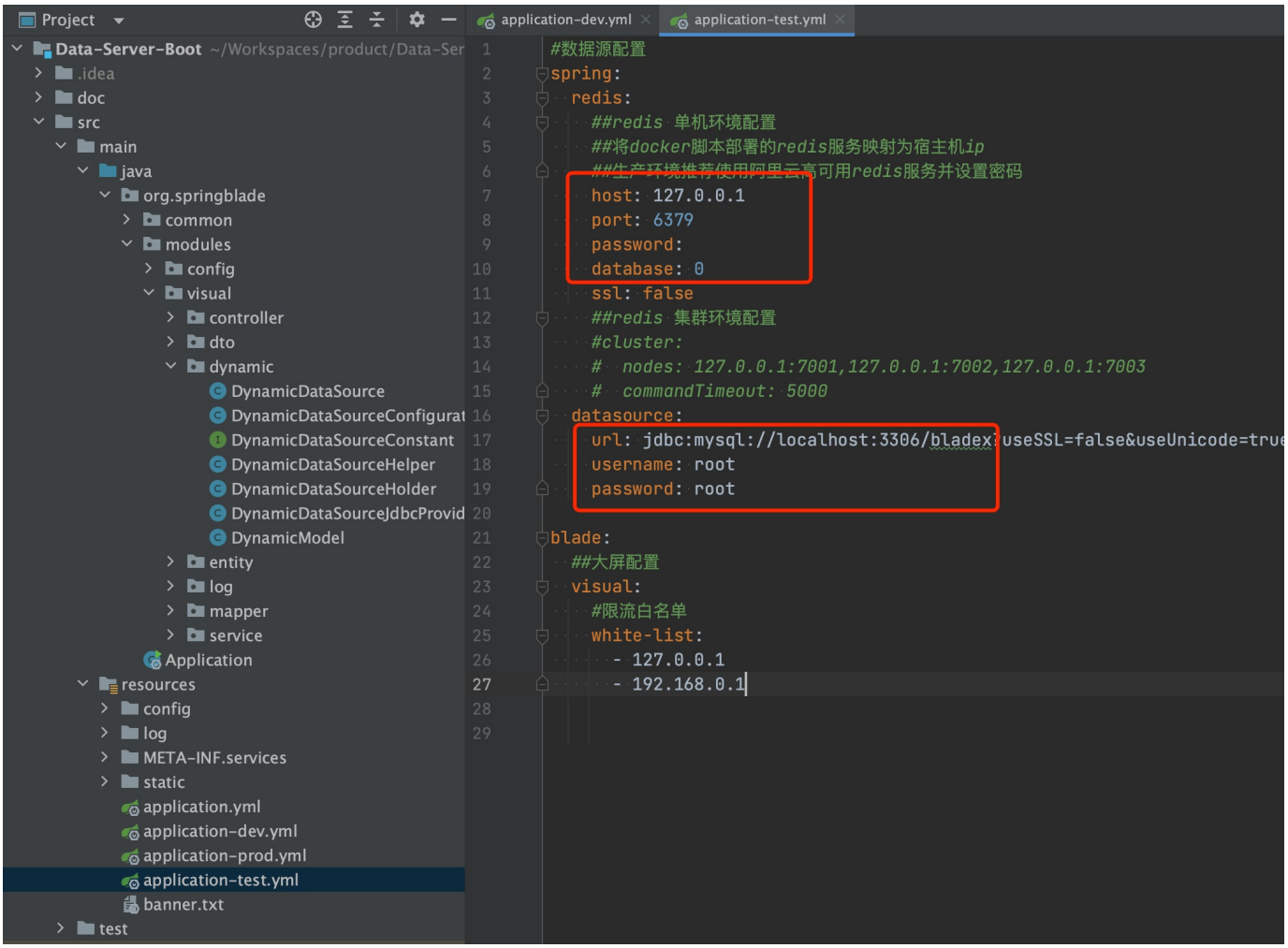
一、站点创建

1. 前往宝塔新建一个web站点

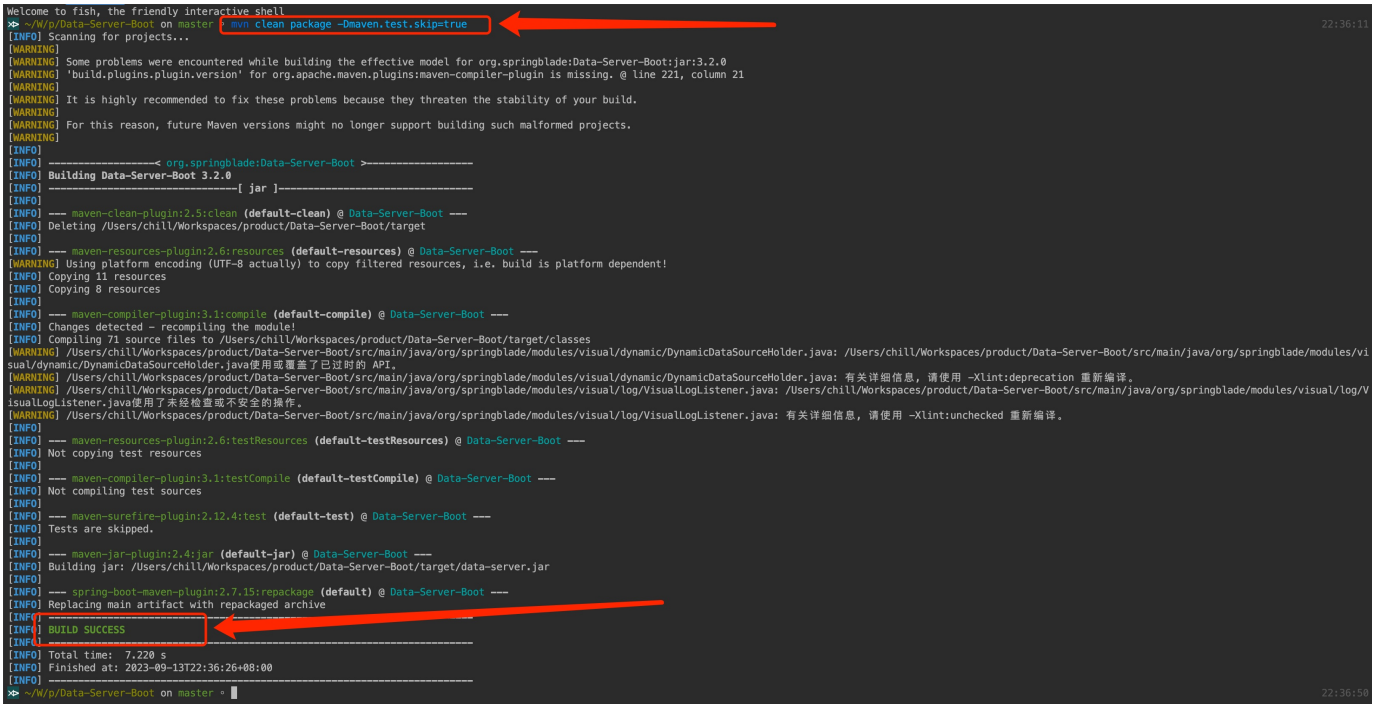
2. 站点访问成功

二、后端部署

1. 将部署好的配置放入application-test.yml

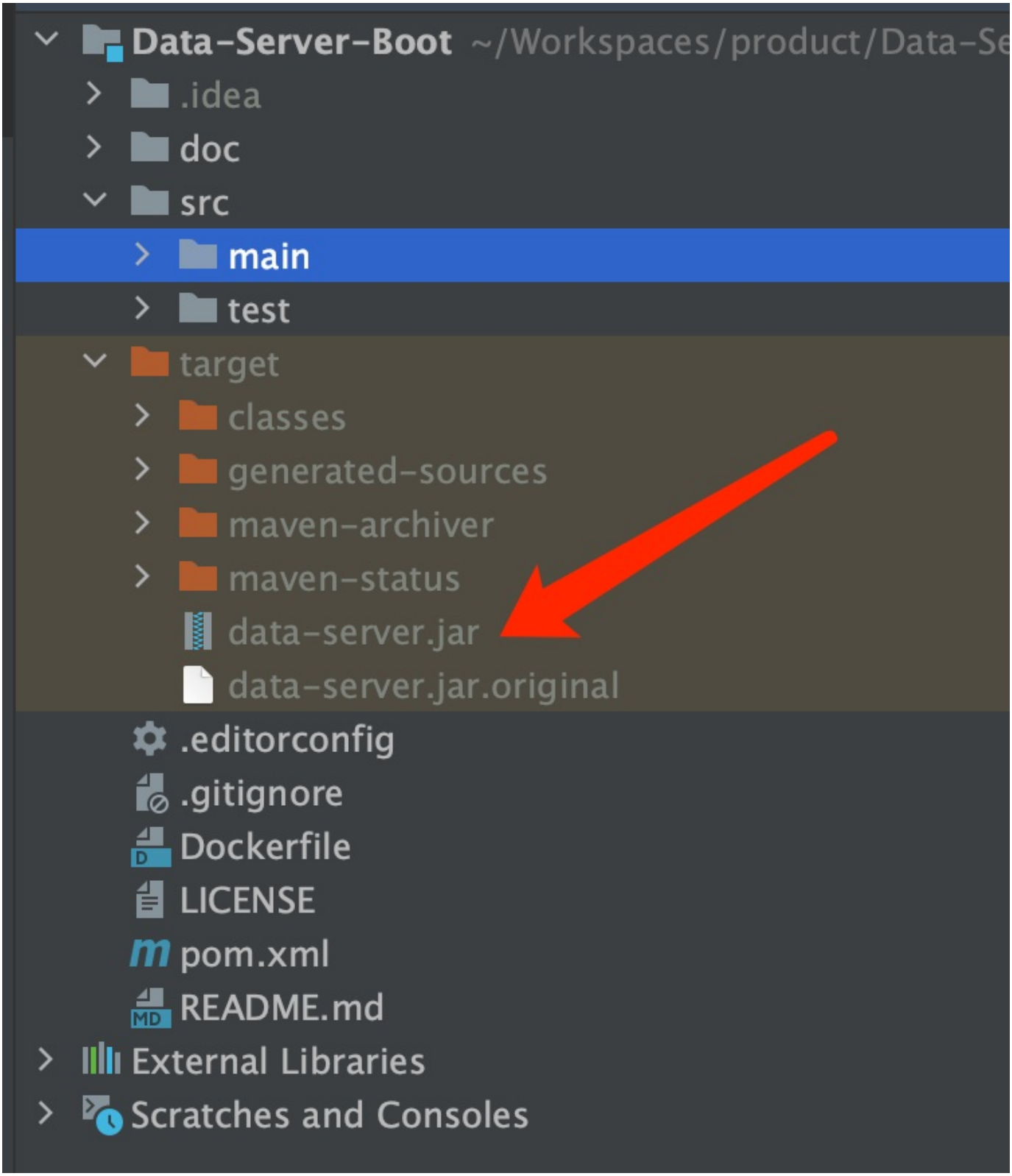


2. 根目录执行 `mvn clean package -Dmaven.test.skip=true`



3. 打包成功后我们便可以看到data-server.jar这个jar包，如有需要我们可以对其改名比如改为

   `blade-visual.jar`

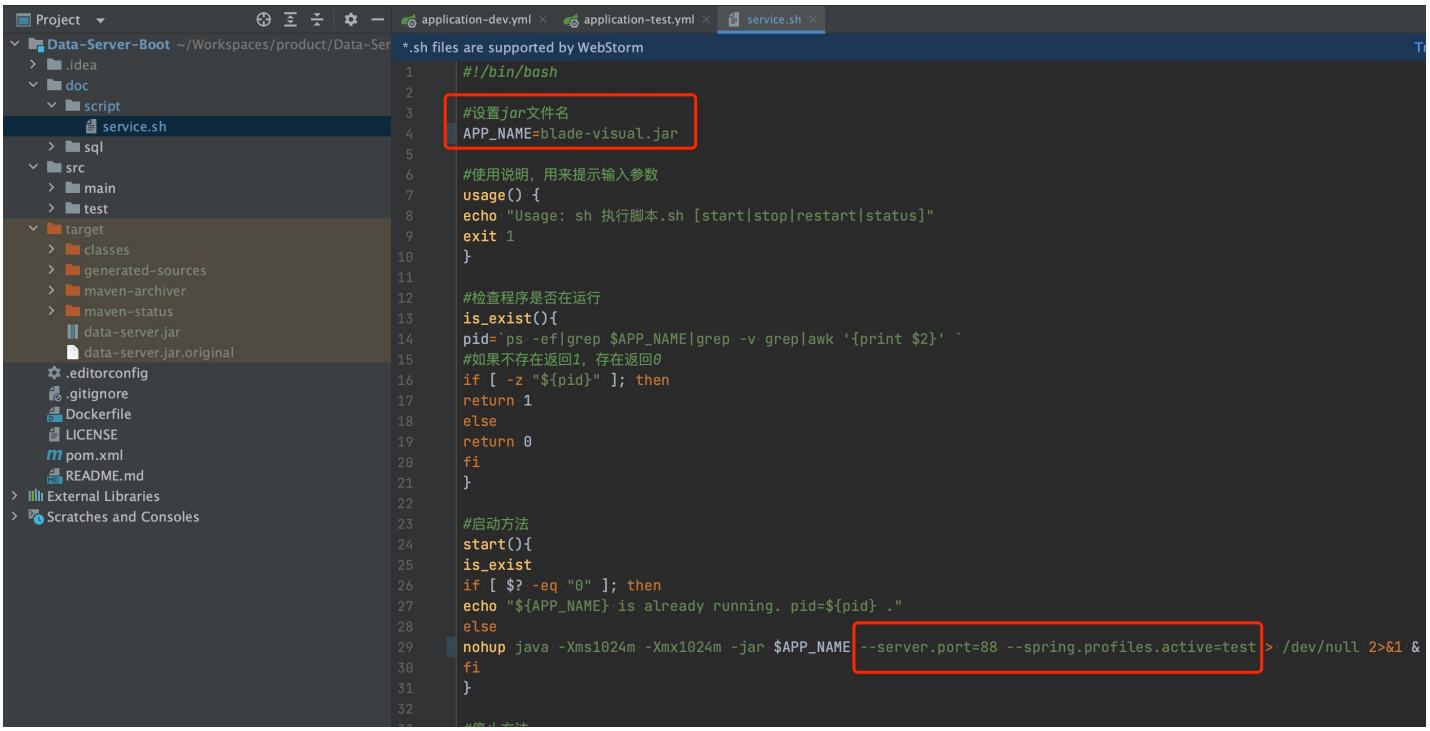


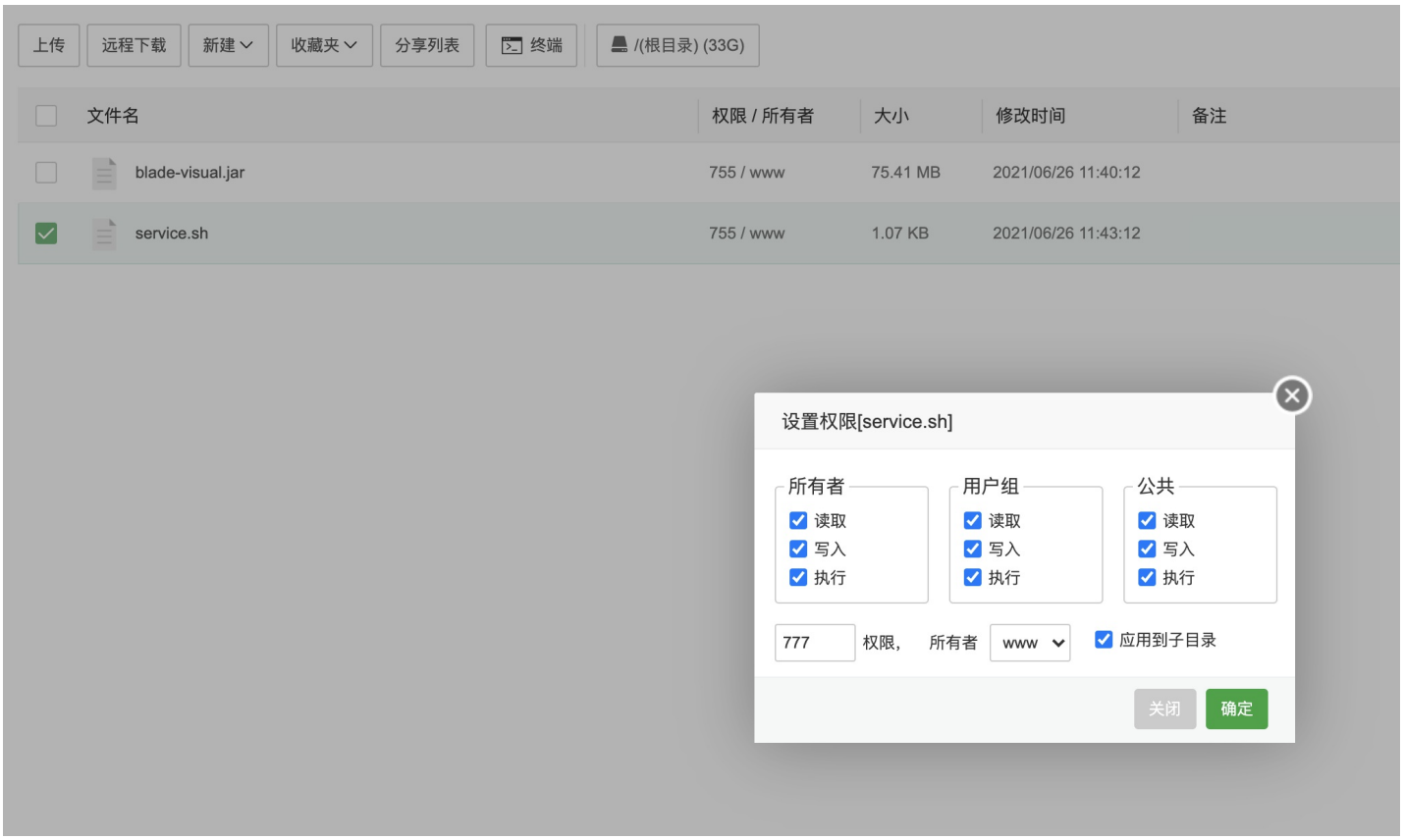
4. 点击进入站点目录，并创建api文件夹

5. 将blade-visual.jar上传至api文件夹

6. 将启动脚本稍作修改也上传至api文件夹并配置好权限

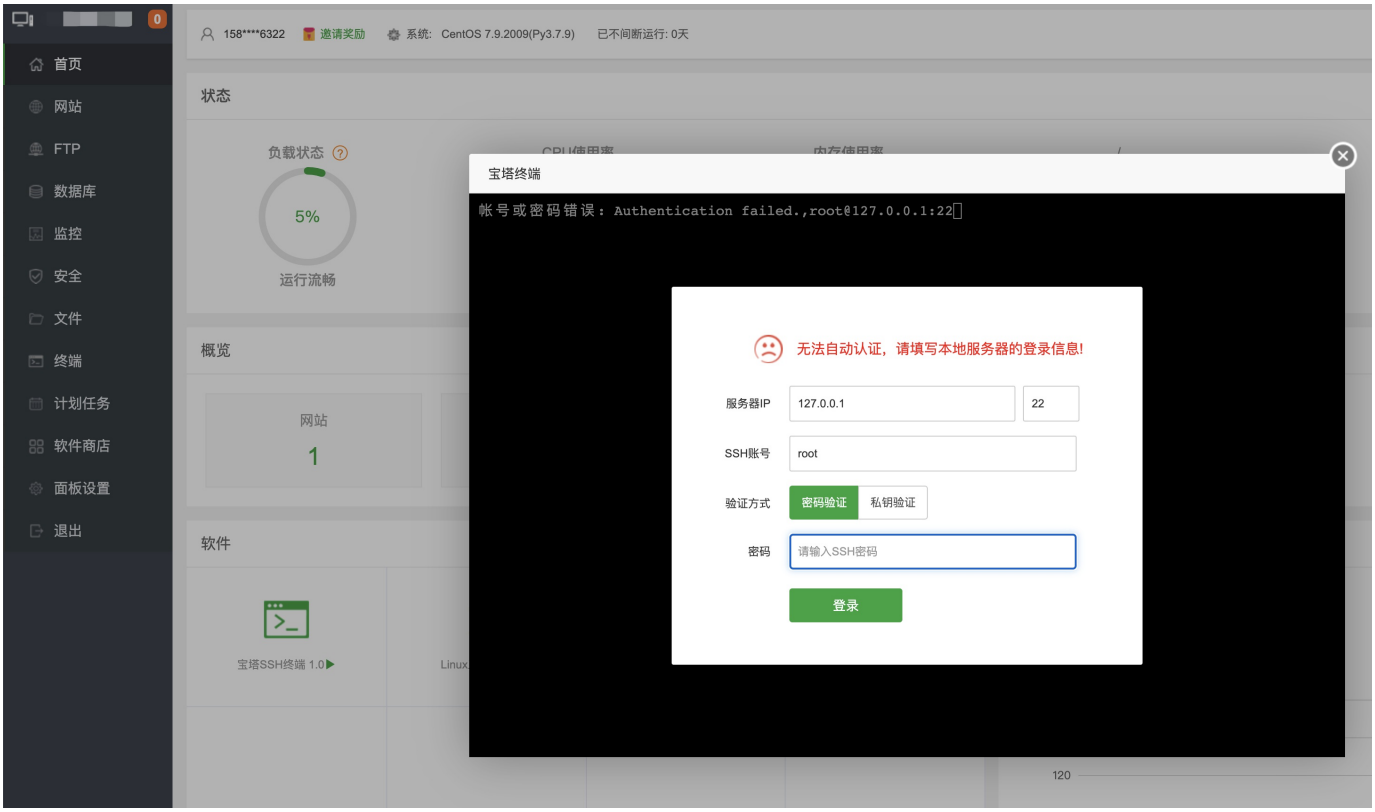
> 默认端口为8050，若我们需要改成其他端口，比如88，给启动命令加上 `--server.port=88`
> 为了生效 `application-test.yml` ，给启动命令加上 `--spring.profiles.active=test`

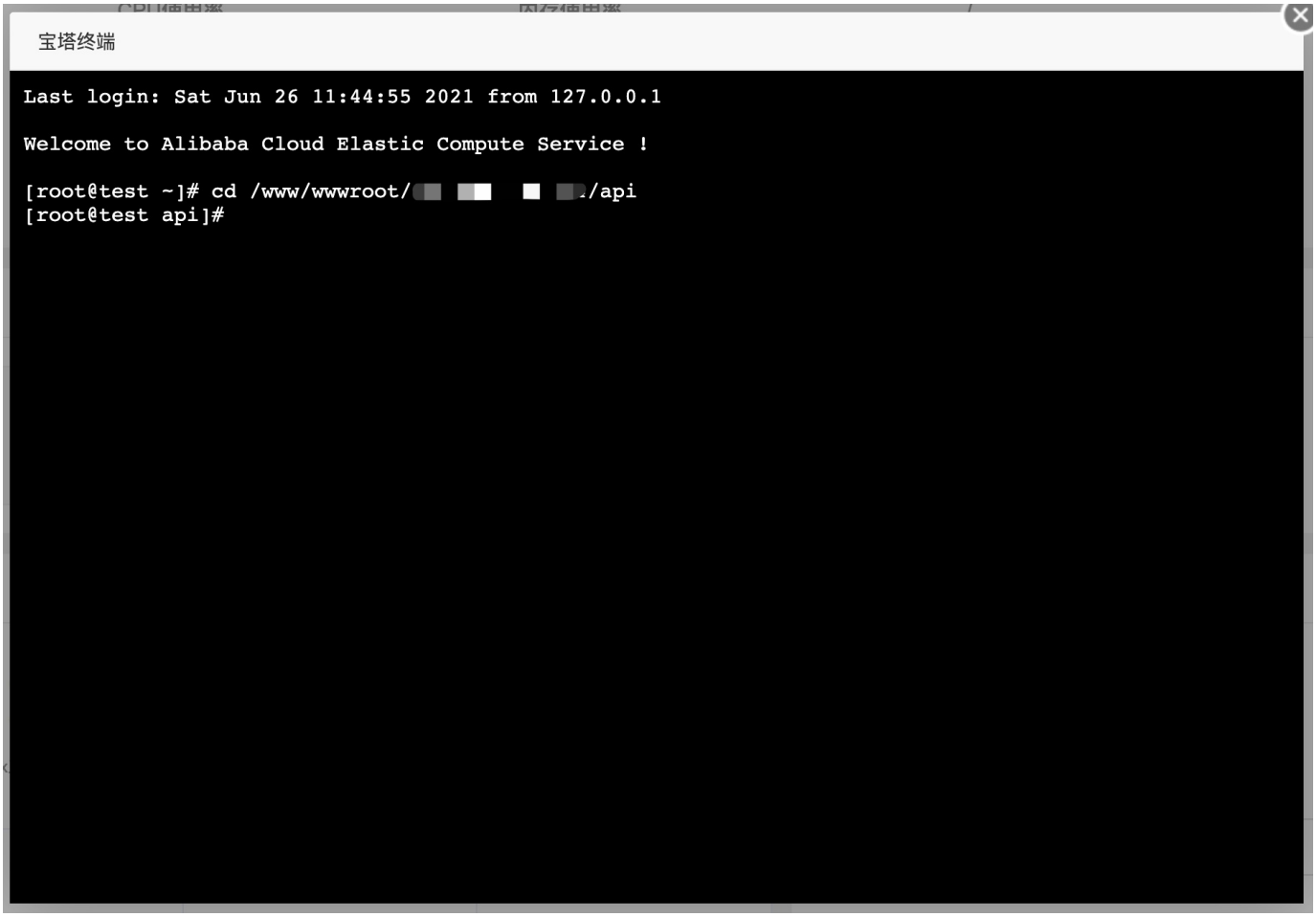




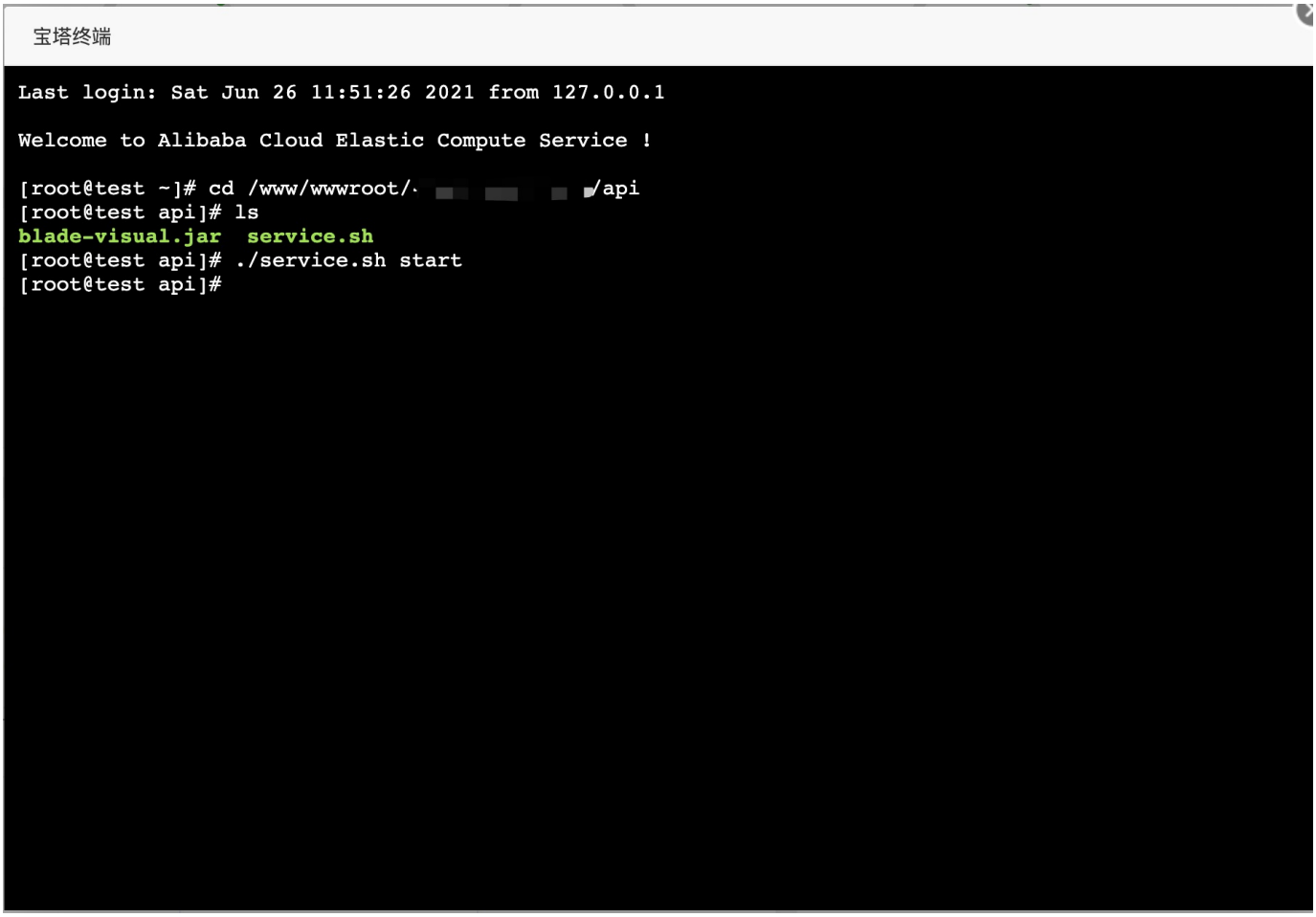
三、后端运行

1. 前往首页点击ssh终端并输入服务器密码

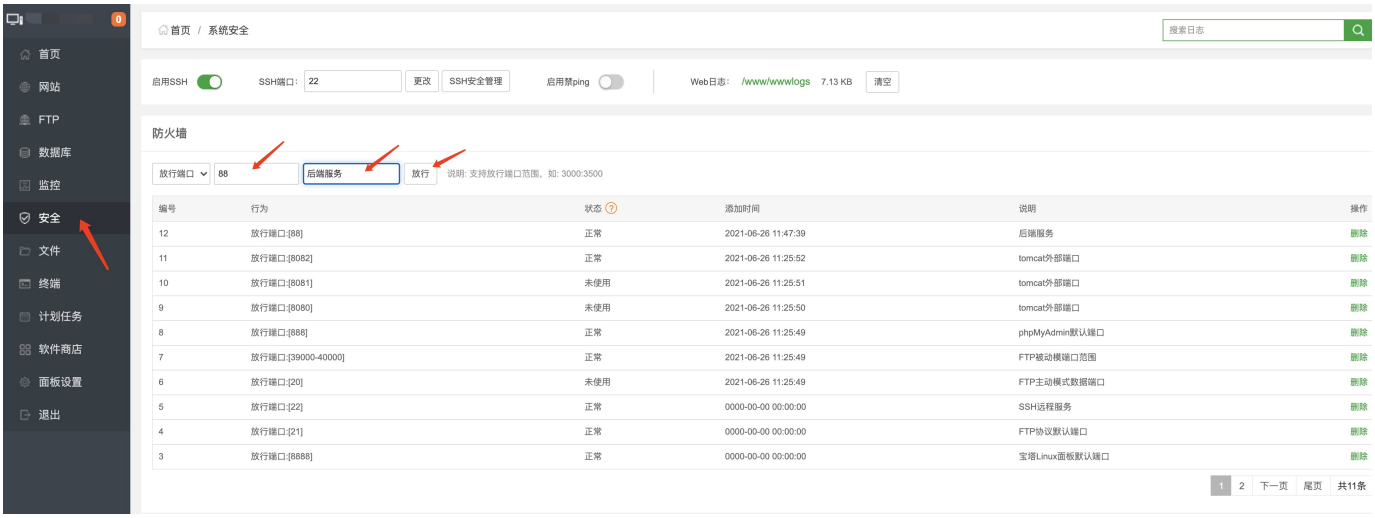

首页
网站
FTP
数据库
监控
安全
文件
终端
计划任务
软件商店
面板设置
退出
状态
负载状态
5%
运行流畅
概览
网站
1
软件
宝塔SSH终端 1.0
宝塔终端
帐号或密码错误：Authentication failed.,root@127.0.0.1:22
无法自动认证，请填写本地服务器的登录信息!
服务器IP
127.0.0.1
22
SSH账号
root
验证方式
密码验证
私钥验证
密码
请输入SSH密码
登录


宝塔终端
Last login: Sat Jun 26 11:44:55 2021 from 127.0.0.1
Welcome to Alibaba Cloud Elastic Compute Service !
[root@test ~]# cd /www/wwwroot/
/api
[root@test api]#

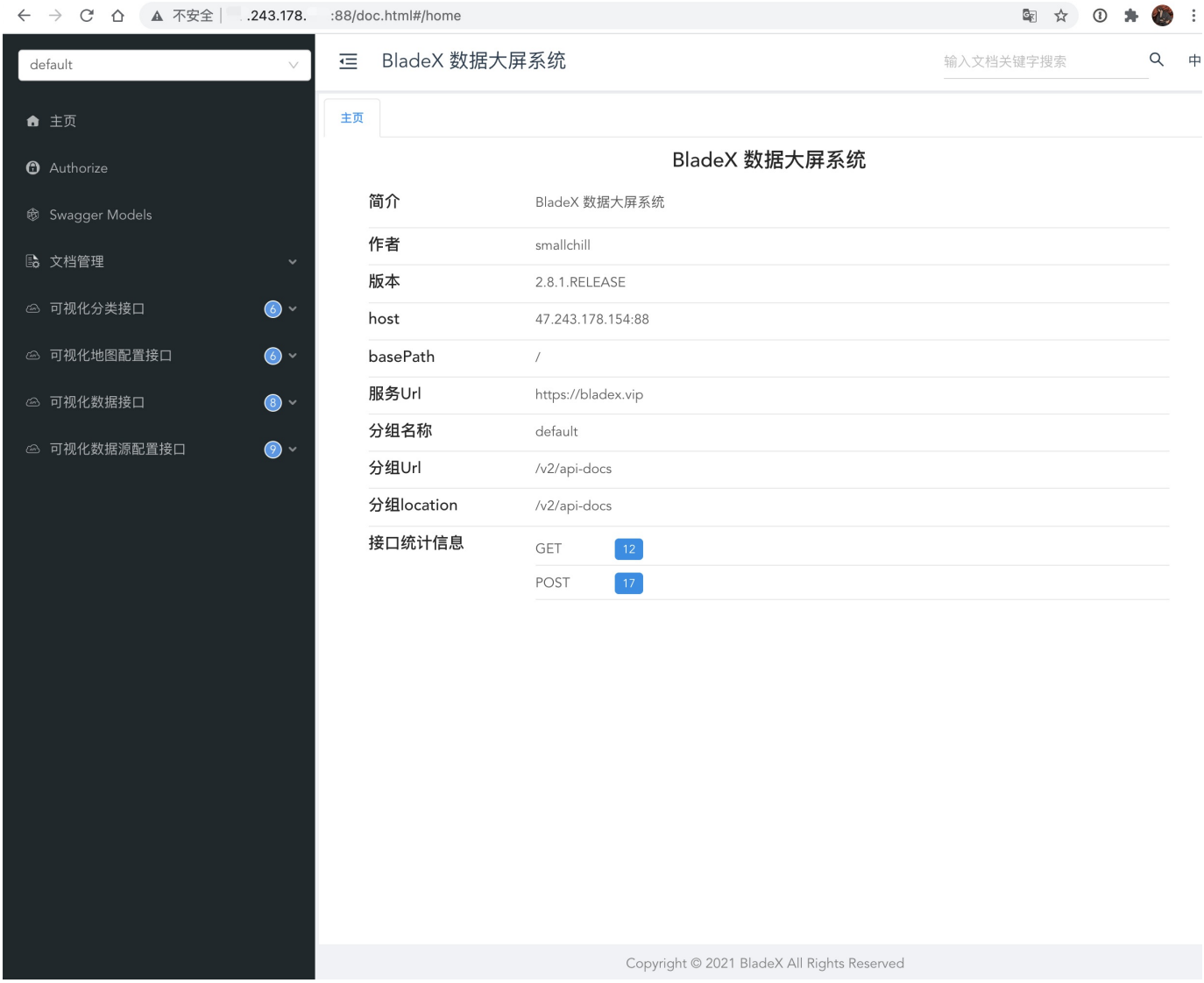
2. 进入目录运行脚本



3. 开放端口（云服务器安全组策略以及宝塔的安全端口都需要开放）



4. 稍等片刻，等待服务启动完毕，我们访问对应地址,看到如下界面则说明后端部署成功



四、前端部署

1. 前端配置刚刚部署好的后端地址

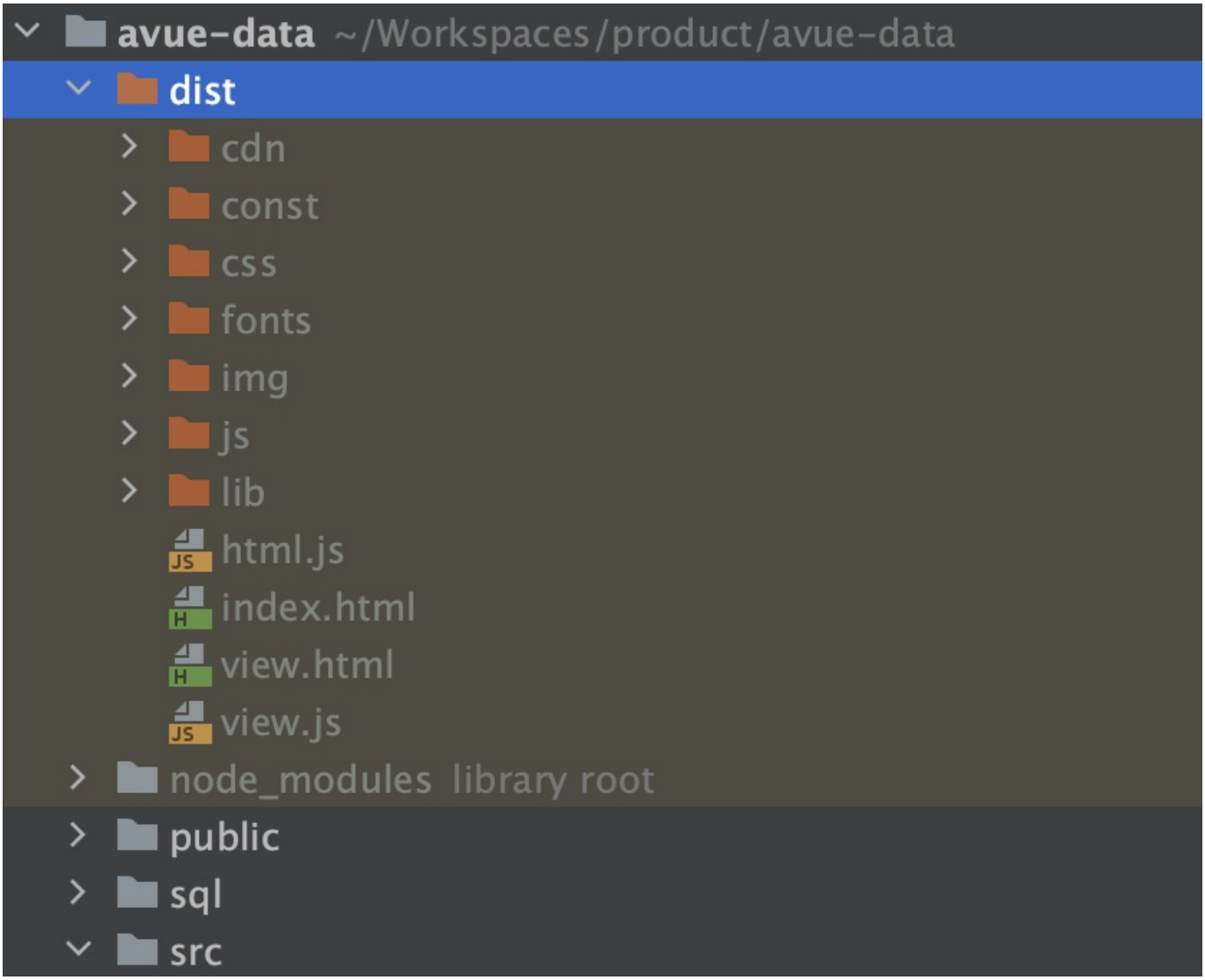
config.js

```
window.$website = {
  isDemo: false,
  isDemoTip: '演示环境不允许操作',
  title: 'BladeX 数据大屏',
  name: 'BladeX 数据大屏',
  subName: '可视化数据大屏（演示环境-请勿放生产数据）',
  url: 'http://服务地址:服务端口/blade-visual',
  autoSave: false,
  autoSaveTime: 60000,
  tabsList: [0, 1, 2, 3, 4, 5, 6],
  componentsList: [{
    name: 'test',
    component: 'testComponents',
    option: 'testOption',
    data: true
  }],
```

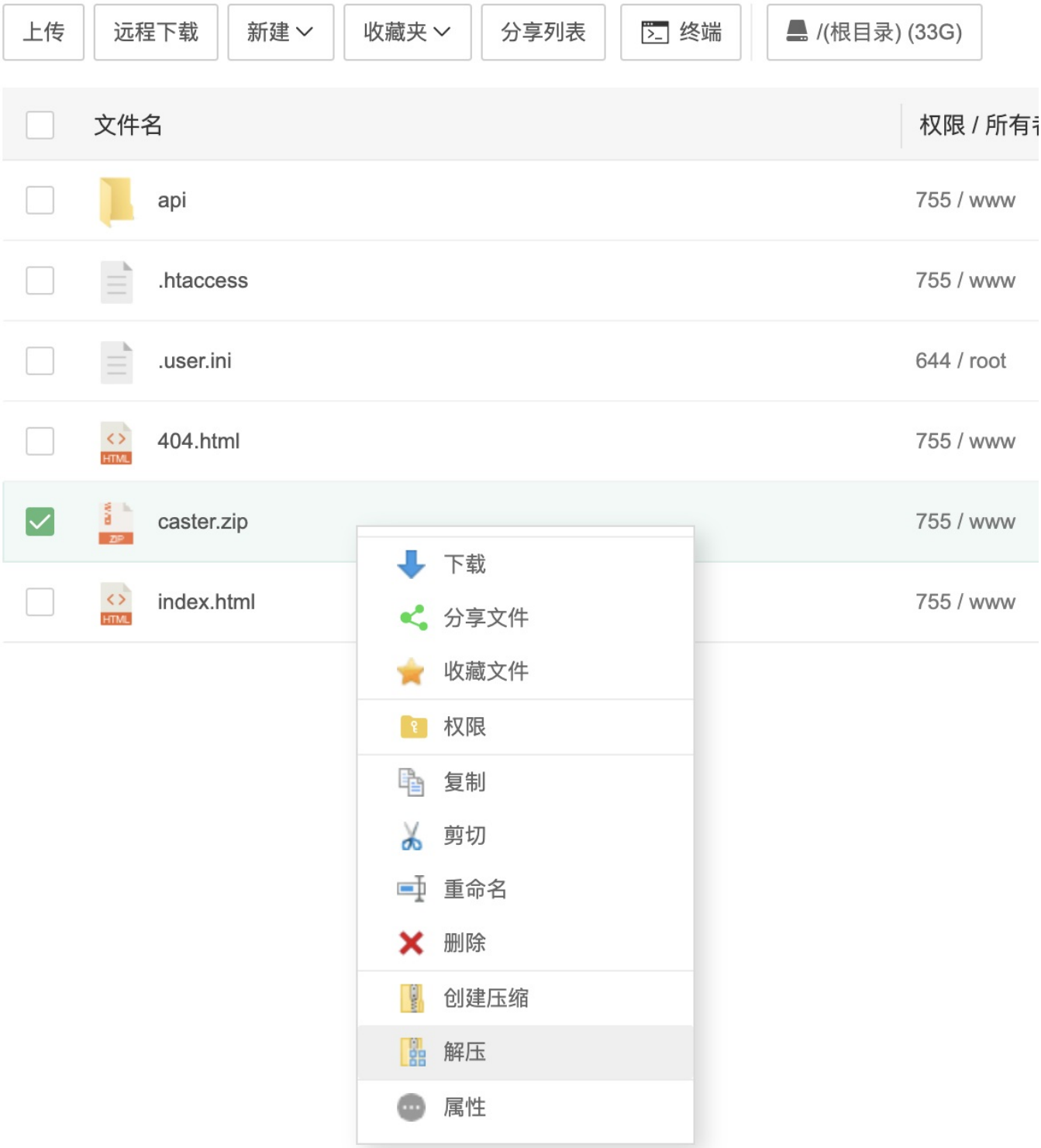
2. 根目录执行 `yarn run build`

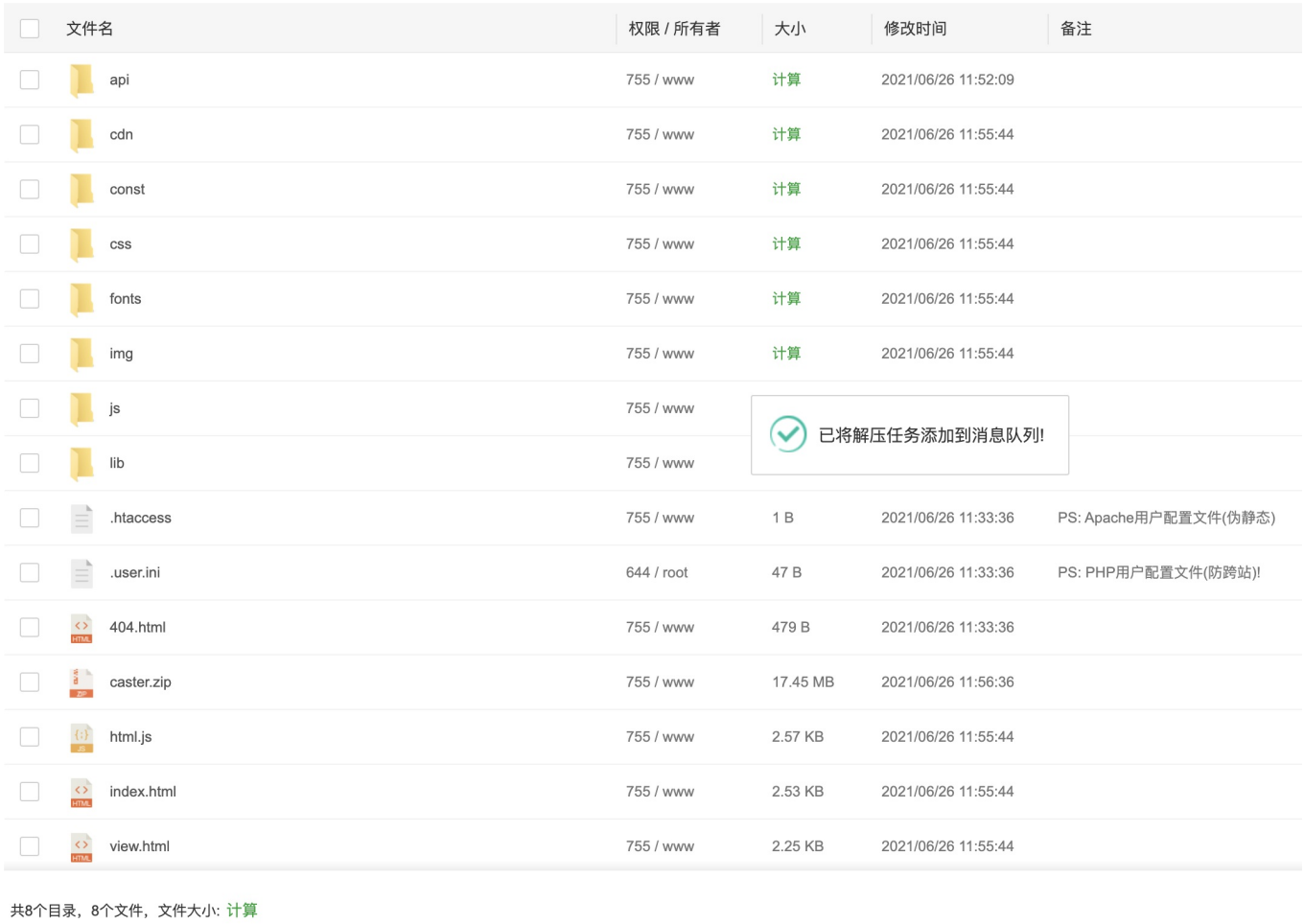
```
Terminal:  Local  +
  dist/cdn/vue/2.5.2/vue.min.js                          91.47 KiB                33.30 KiB
  dist/cdn/vue-router/3.0.1/vue-router.min.js            23.98 KiB                8.40 KiB
  dist/cdn/axios/1.0.0/axios.min.js                      12.67 KiB                4.59 KiB
  dist/cdn/vuex/3.1.1/vuex.min.js                        11.38 KiB                3.16 KiB
  dist/cdn/vuex/2.4.1/vuex.min.js                        9.07 KiB                 2.88 KiB
  dist/js/page.cfa5c502.js                               2234.95 KiB              620.20 KiB
  dist/cdn/element-ui/2.13.2/index.js                    553.87 KiB               141.08 KiB
  dist/js/chunk-vendors.bf41d7b1.js                      435.38 KiB               126.92 KiB
  dist/view.js                                           75.63 KiB                11.04 KiB
  dist/js/app.74c1de03.js                                19.39 KiB                4.96 KiB
  dist/js/chunk-2d0ae937.e9d597f6.js                     17.89 KiB                6.46 KiB
  dist/js/chunk-2d2308b7.bcae8994.js                     5.61 KiB                 2.07 KiB
  dist/const/theme/halloween.project.js                  4.47 KiB                 1.01 KiB
  dist/const/theme/wonderland.project.js                 4.32 KiB                 0.92 KiB
  dist/const/theme/avue.project.js                       4.29 KiB                 0.95 KiB
  dist/html.js                                           2.57 KiB                 0.92 KiB
  dist/css/chunk-vendors.2406d2da.css                    242.71 KiB               36.25 KiB
  dist/cdn/element-ui/2.13.2/theme-chalk/index.css       226.98 KiB               34.55 KiB
  dist/cdn/avue/index.css                                74.14 KiB                12.54 KiB
  dist/cdn/animate/3.5.1/animate.css                     51.55 KiB                3.86 KiB
  dist/css/page.6c9fa4d7.css                             46.67 KiB                8.64 KiB
  dist/lib/index.css                                     4.98 KiB                 1.37 KiB
  dist/css/app.29b235bd.css                              3.89 KiB                 1.15 KiB

  Images and other types of assets omitted.

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html

✨  Done in 47.60s.
```

3. 打包成功后，我们便可以看到 `dist` 文件夹下的目录，拷贝出来准备上传



4. 将目录打成zip包，并上传至刚刚创建的站点根目录并右键解压

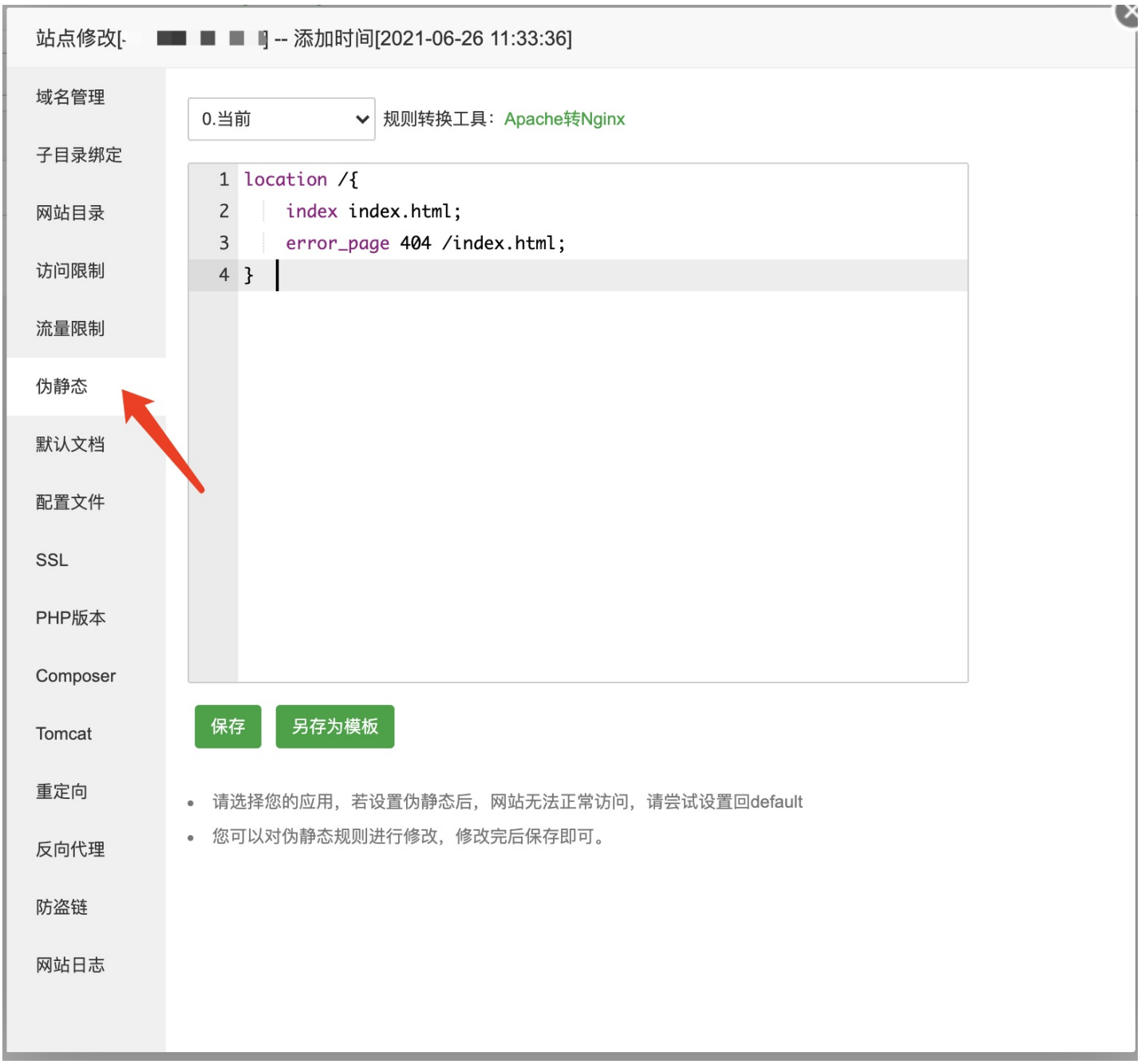

上传
远程下载
新建
收藏夹
分享列表
终端
/(根目录) (33G)
文件名
权限 / 所有
api
755 / www
.htaccess
755 / www
.user.ini
644 / root
404.html
755 / www
caster.zip
755 / www
index.html
755 / www
下载
分享文件
收藏文件
权限
复制
剪切
重命名
删除
创建压缩
解压
属性



5. 返回站点列表，进行nginx伪静态配置，否则会出现404

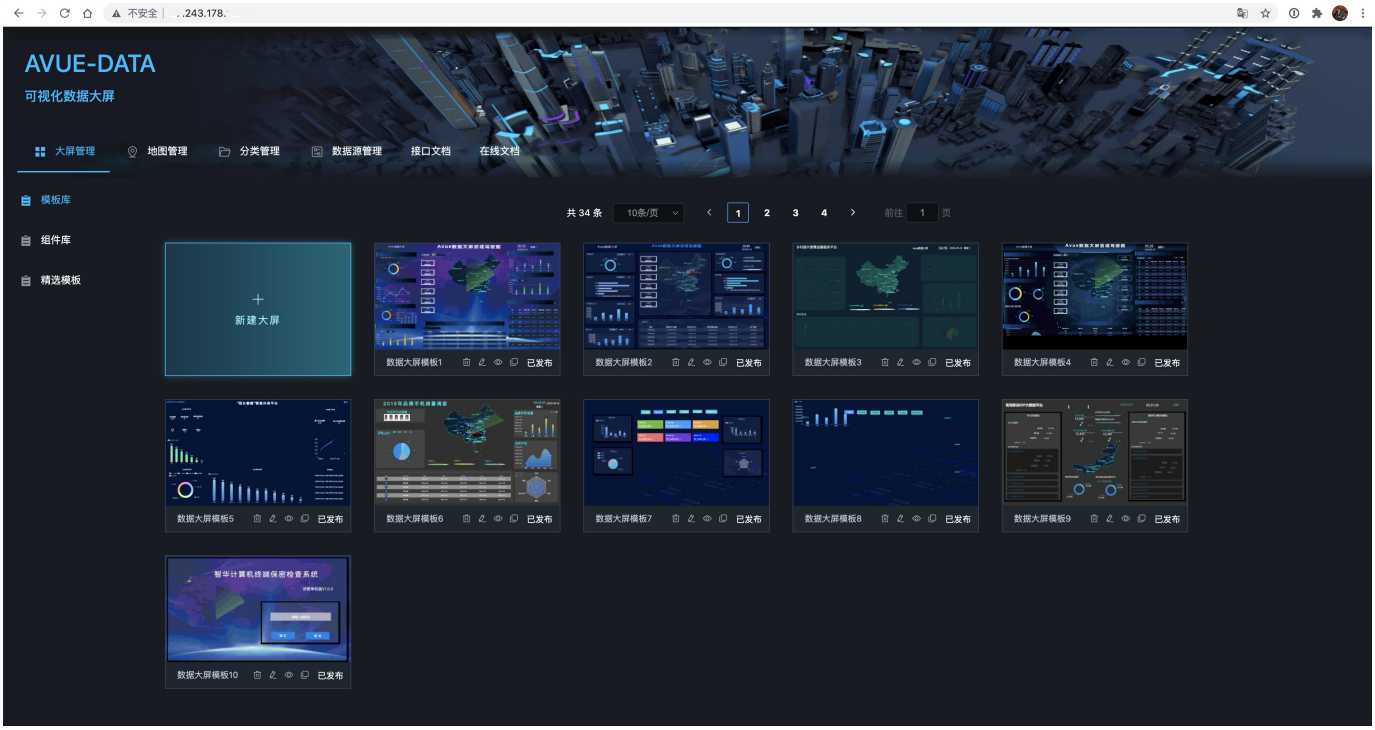
6. 加入配置并保存

```
location /{
    index index.html;
    error_page 404 /index.html;
}
```

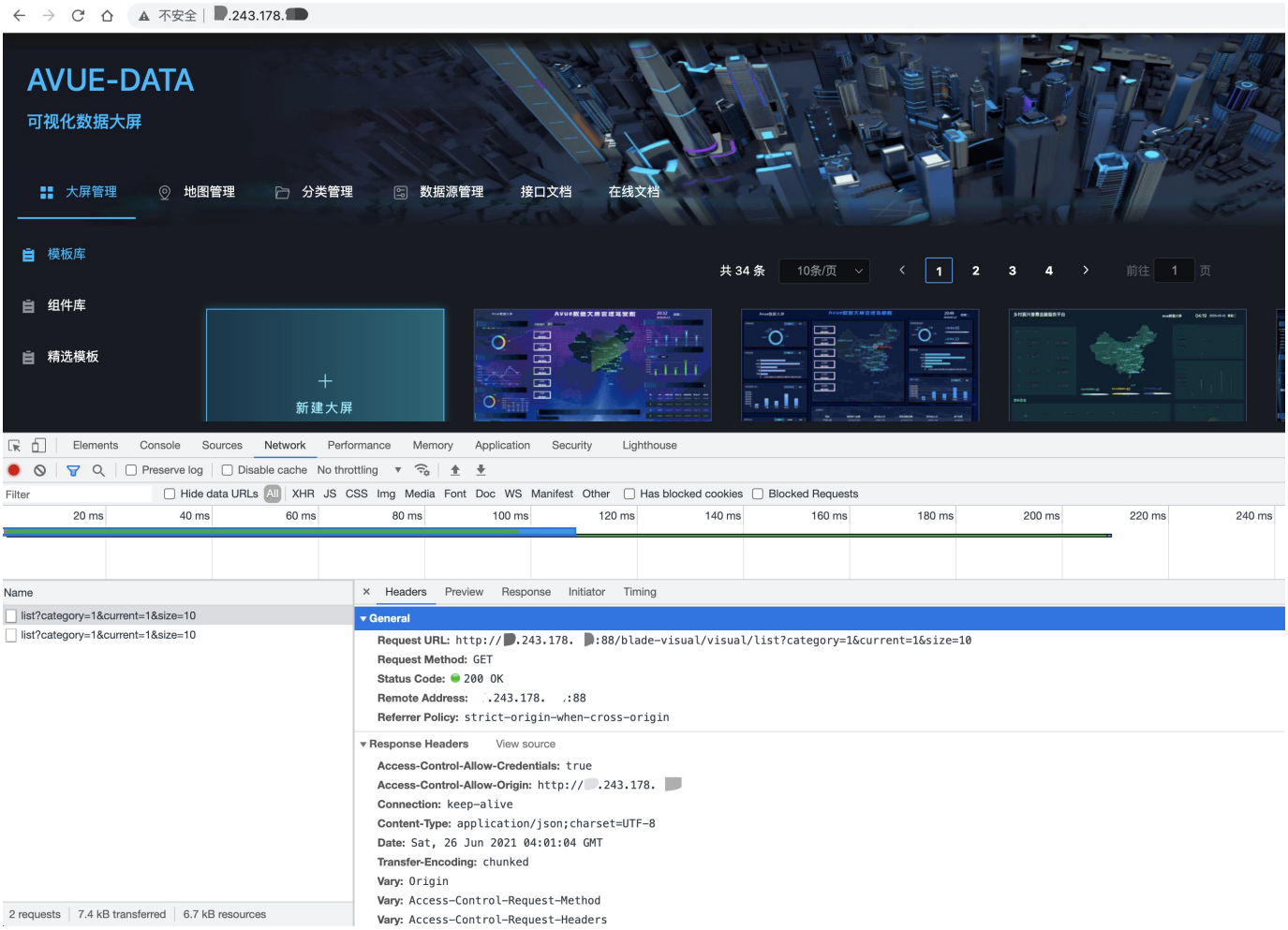


五、前端运行

1. 刷新站点网站，可以看到系统已经部署成功



2. 打开f12，可以看到接口地址也正确无误


不安全 | .243.178.
AVUE-DATA
可视化数据大屏
大屏管理
地图管理
分类管理
数据源管理
接口文档
在线文档
模板库
组件库
精选模板
共 34 条
10条/页
1
2
3
4
前往
1
页
新建大屏
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Preserve log
Disable cache
No throttling
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
Has blocked cookies
Blocked Requests
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms
140 ms
160 ms
180 ms
200 ms
220 ms
240 ms
Name
list?category=1&current=1&size=10
list?category=1&current=1&size=10
2 requests
7.4 kB transferred
6.7 kB resources
Headers
Preview
Response
Initiator
Timing
General
Request URL: http://.243.178. :88/blade-visual/visual/list?category=1&current=1&size=10
Request Method: GET
Status Code: 200 OK
Remote Address: .243.178. :88
Referrer Policy: strict-origin-when-cross-origin
Response Headers
View source
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://.243.178.
Connection: keep-alive
Content-Type: application/json;charset=UTF-8
Date: Sat, 26 Jun 2021 04:01:04 GMT
Transfer-Encoding: chunked
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers