



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	
常见例子库	
项目启动	
前端启动	
Node版后端启动	
Java版后端启动	
Boot版后端启动	
Nacos启动	
Cloud版后端启动	
全局变量	
组件数据	
数据过滤器	
数据集共用	
组件参数	
组件交互	
自定义Vue组件	
使用高德地图	
自定义Vue组件之间交互	
自定义Echart组件	
外部组件开发	
内部组件开发	
单独大屏导出部署	
教学视频	
项目部署	
Nginx部署	
1.项目打包	
2.Linux部署	
3.Baota部署	
3.1安装宝塔	
3.2基础部署	
3.3大屏部署	

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。



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

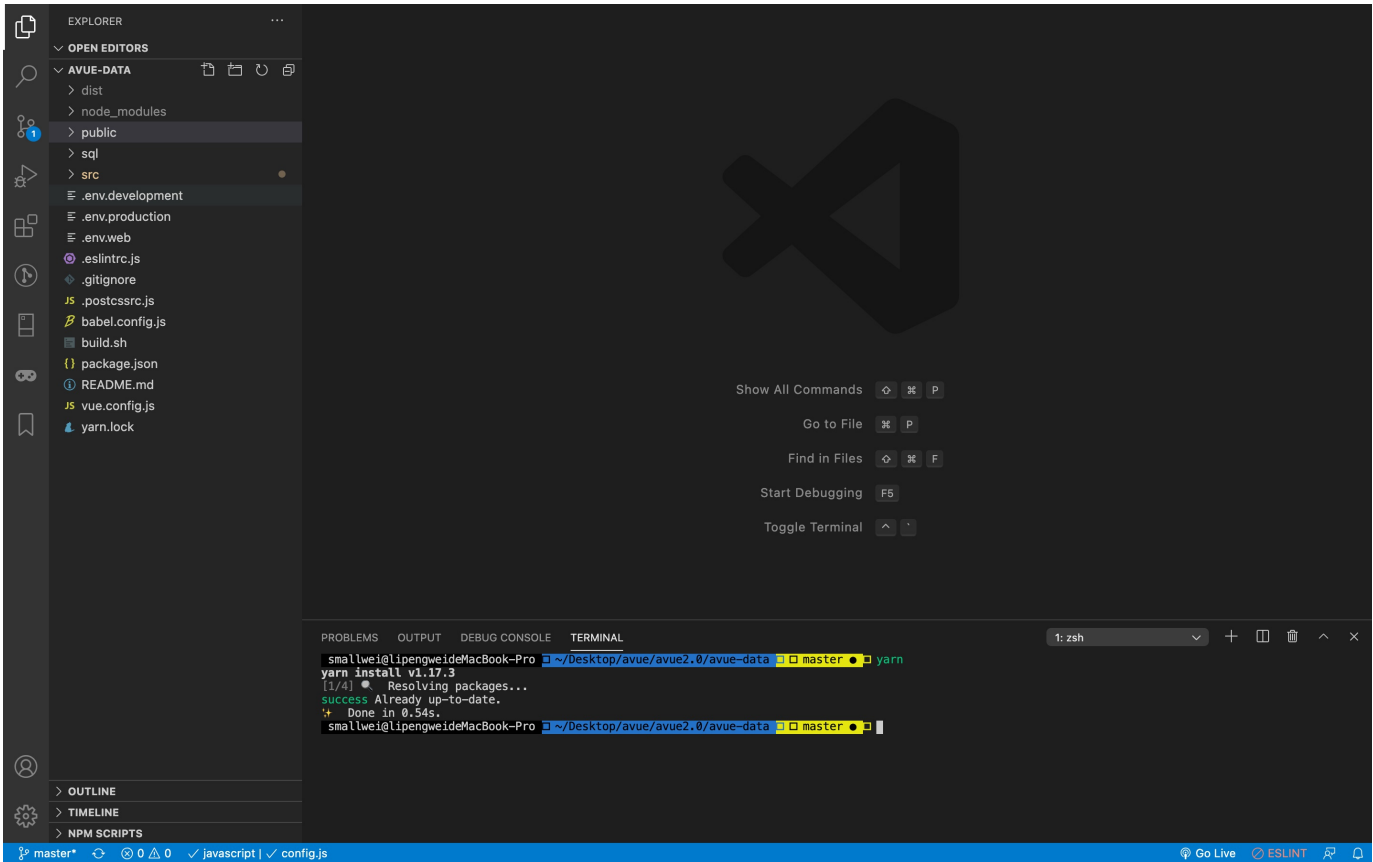
前端启动

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

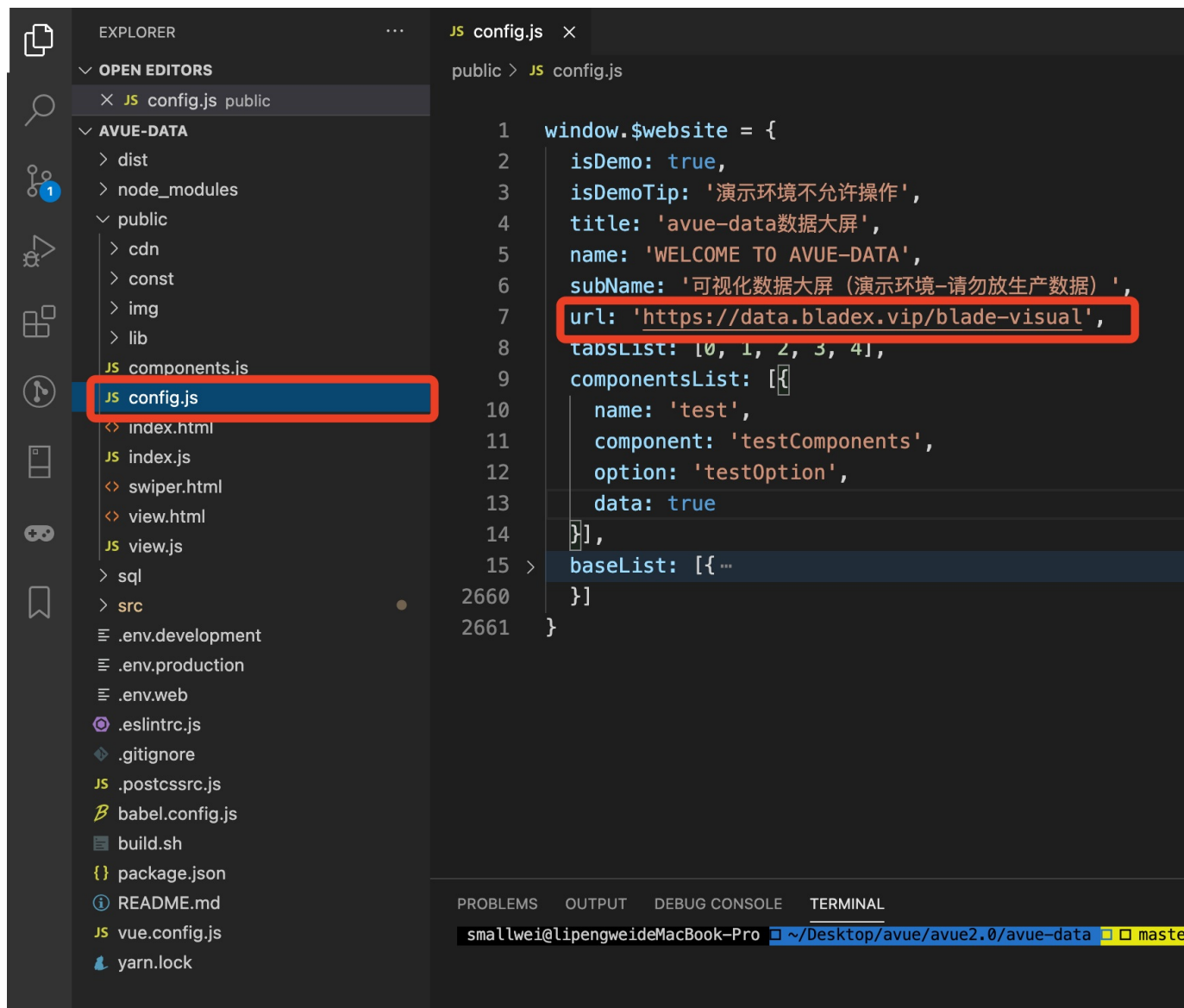


二、配置修改

1.这里要运行大屏后端

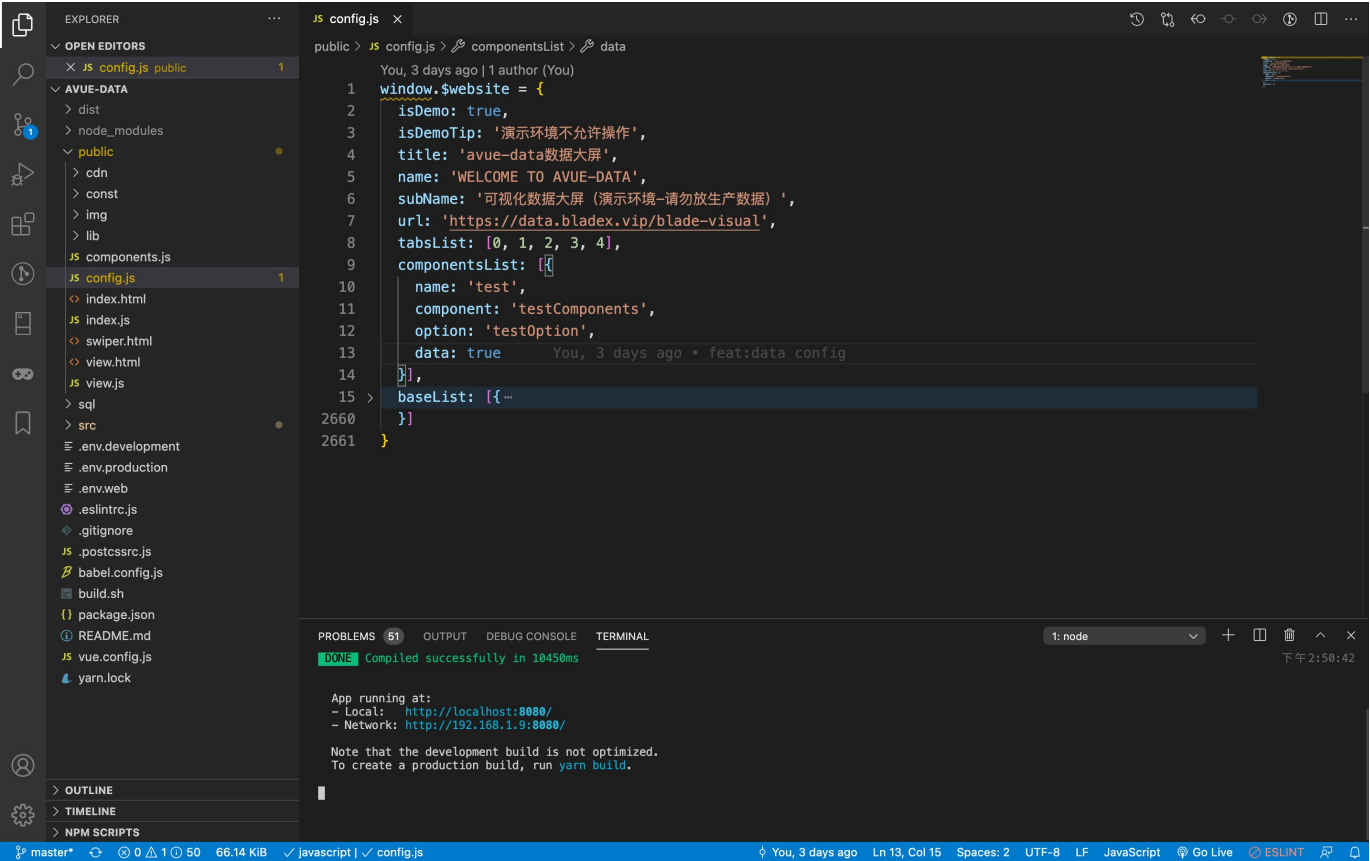
- [JAVA版后端启动](#)
- [NODE版后端启动](#)

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



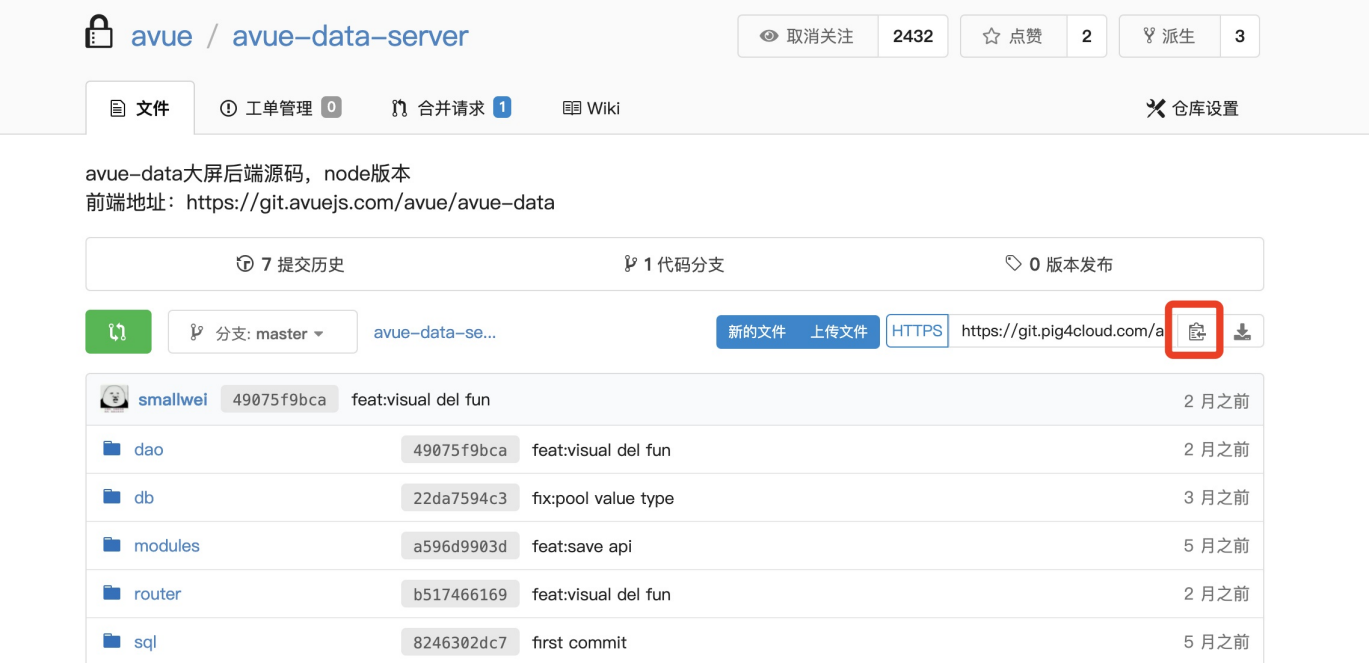
2. 访问 <http://localhost:8080> 查看效果



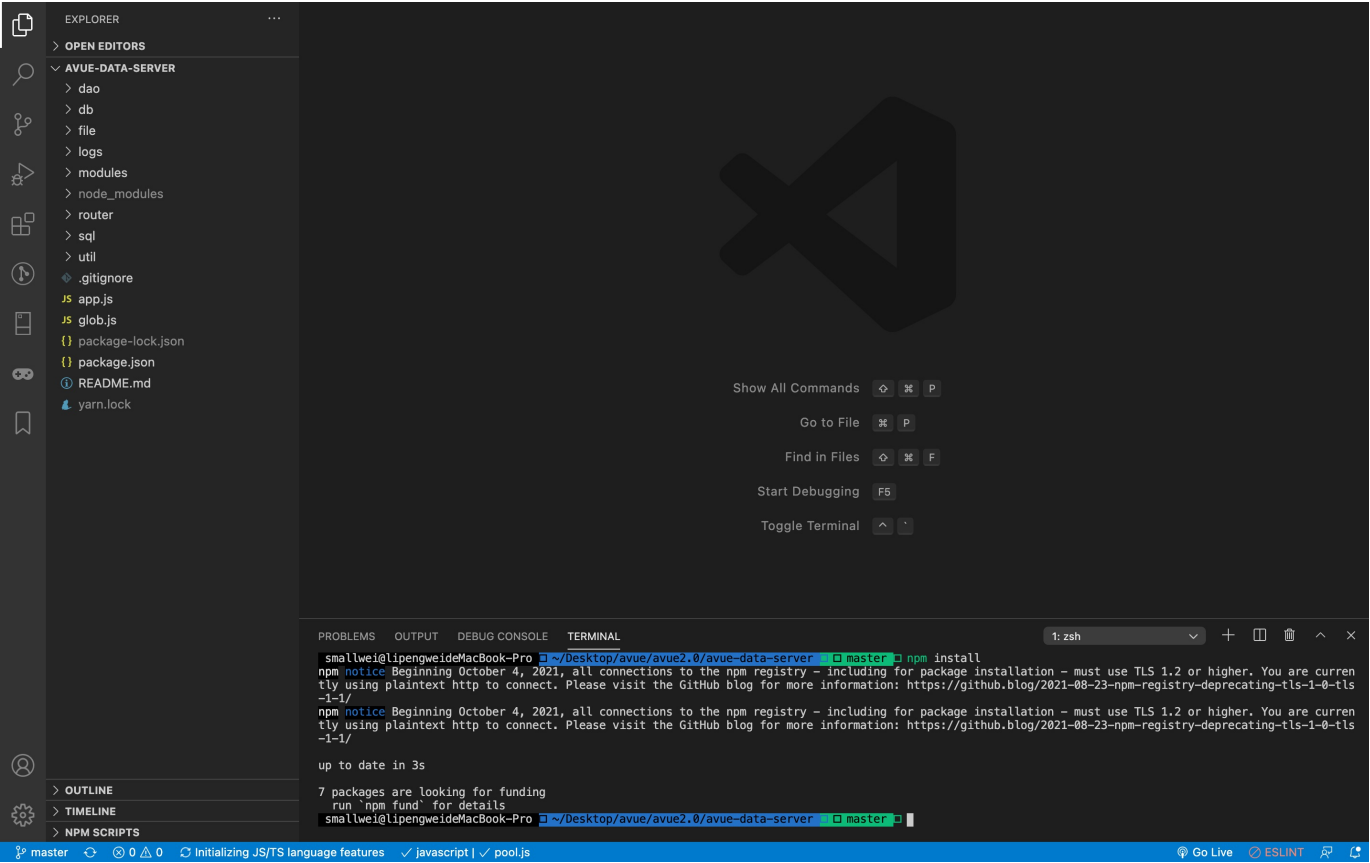
Node版后端启动

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>

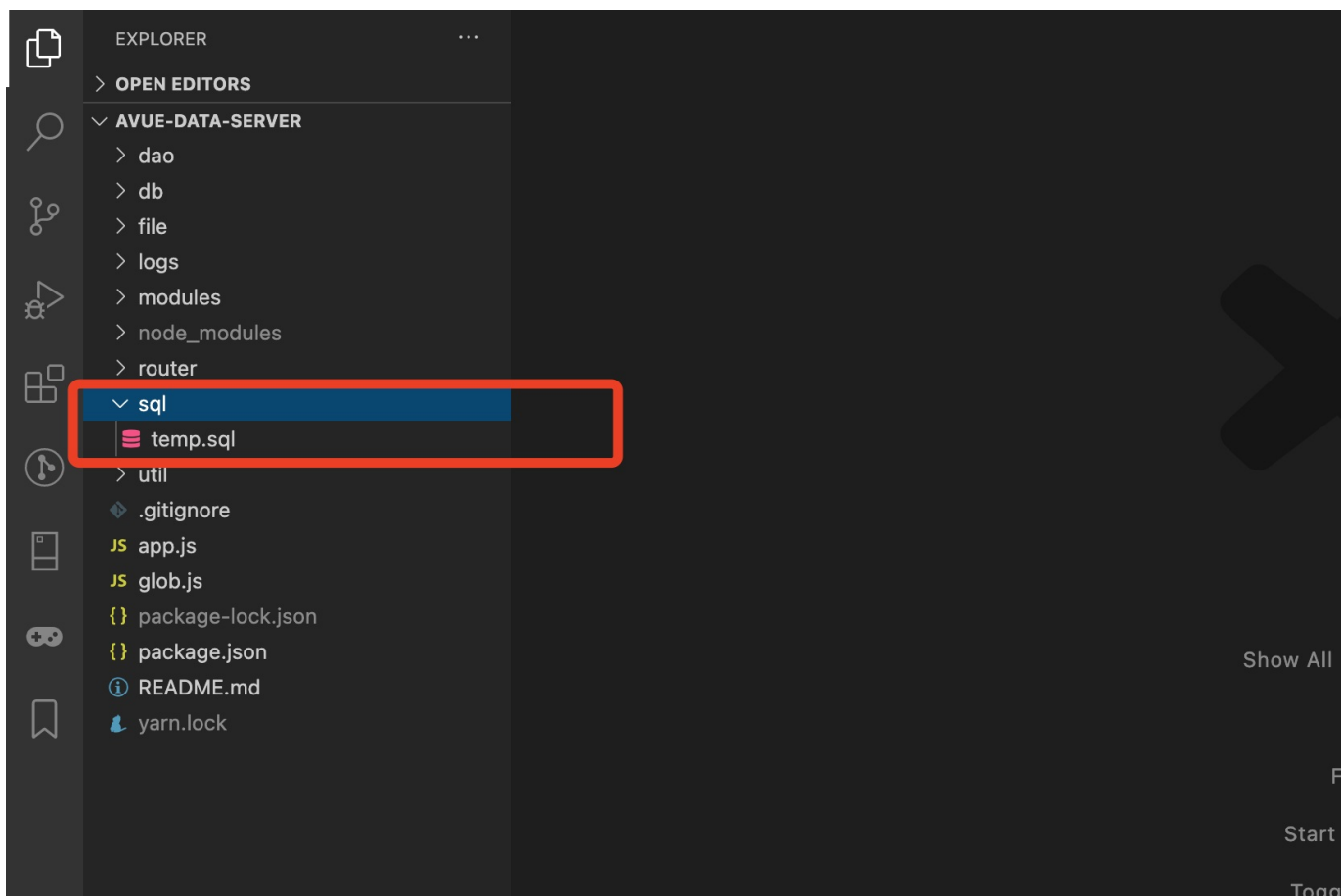


2. 导入工程,在终端执行yarn install或者npm install



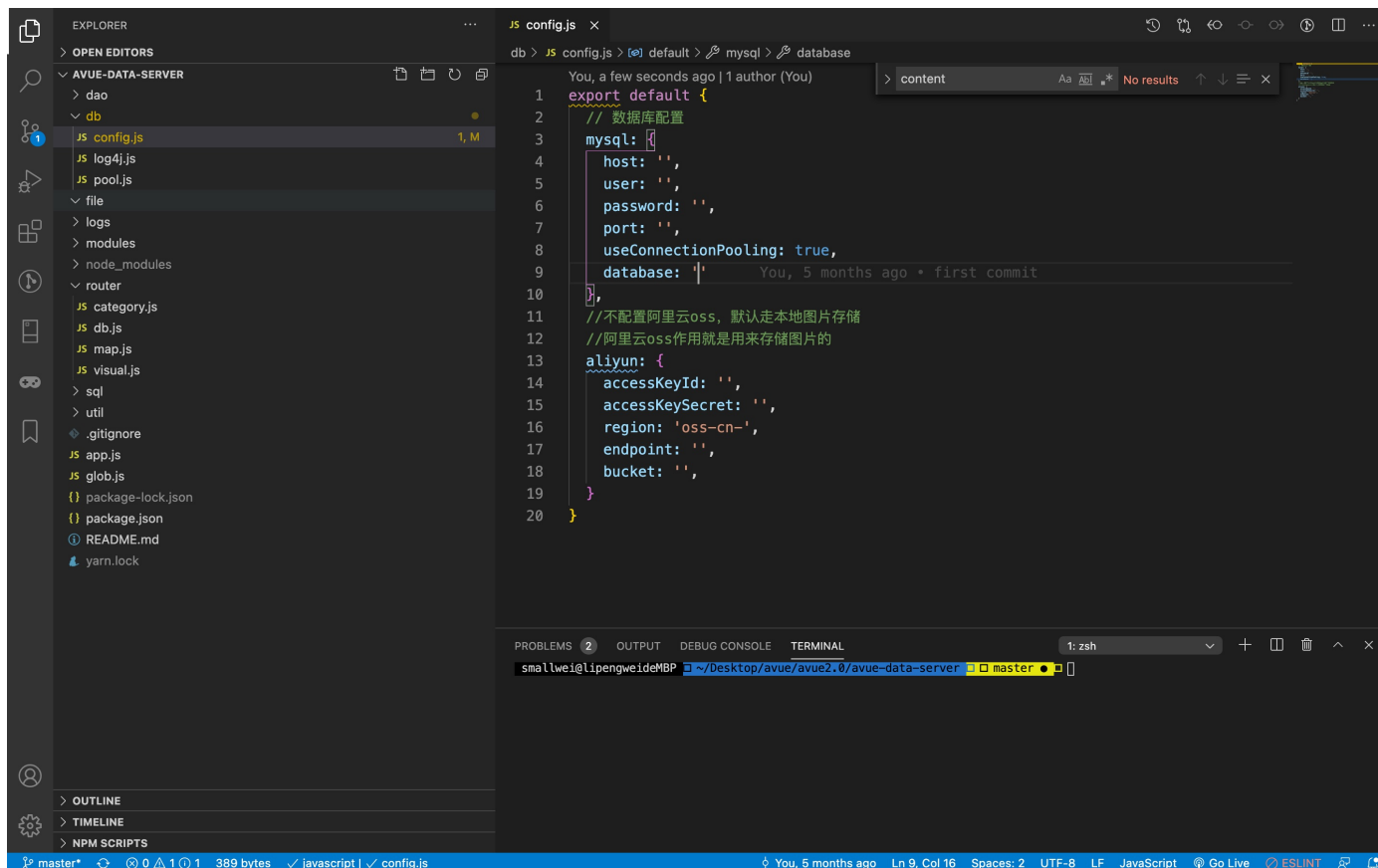
二、配置修改

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)

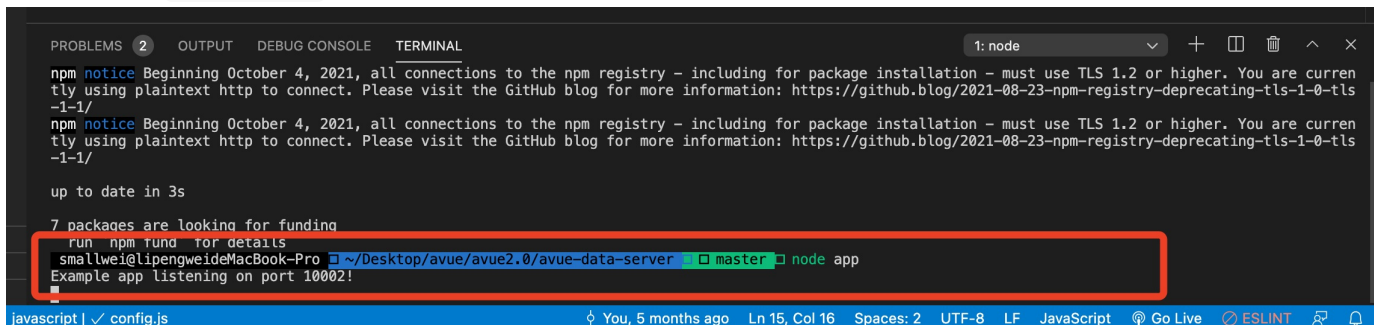
说明:不配置阿里云oss，默认走本地图片存储



!

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

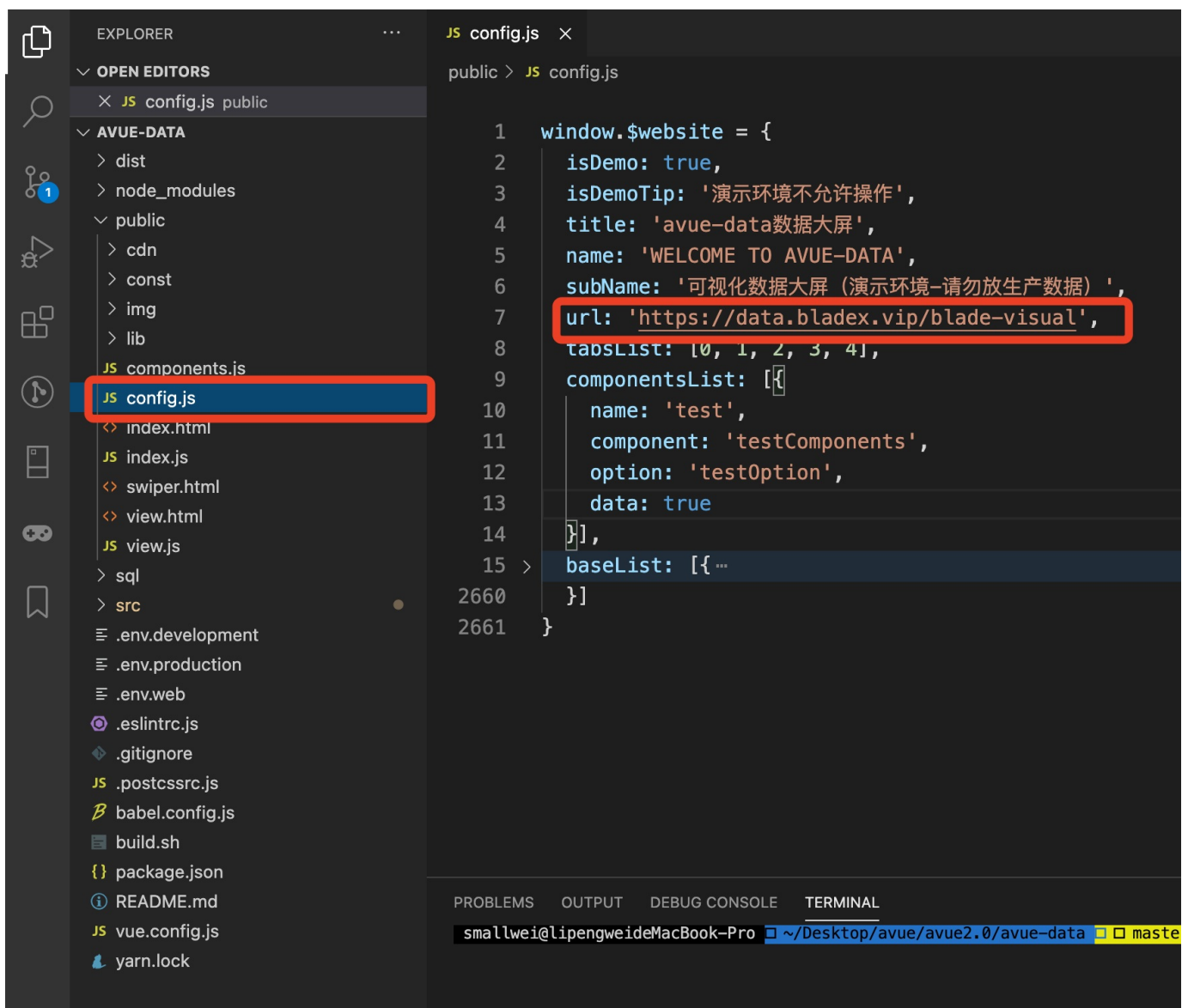


```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL 1: node
npm notice Beginning October 4, 2021, all connections to the npm registry - including for package installation - must use TLS 1.2 or higher. You are currently using plaintext http to connect. Please visit the GitHub blog for more information: https://github.blog/2021-08-23-npm-registry-deprecating-tls-1-0-tls-1-1-1/
npm notice Beginning October 4, 2021, all connections to the npm registry - including for package installation - must use TLS 1.2 or higher. You are currently using plaintext http to connect. Please visit the GitHub blog for more information: https://github.blog/2021-08-23-npm-registry-deprecating-tls-1-0-tls-1-1-1/

up to date in 3s

7 packages are looking for funding
  run npm fund for details
smallwei@lipengweideMacBook-Pro: ~/Desktop/avue/avue2.0/avue-data-server [master] node app
Example app listening on port 10002!
```

2. 前端项目中修改成改地址[这里是修改前端项目](#)



```
EXPLORER
  OPEN EDITORS
    JS config.js public
  AVUE-DATA
    dist
    node_modules
    public
      cdn
      const
      img
      lib
      JS components.js
      JS config.js
      index.html
      JS index.js
      swiper.html
      view.html
      JS view.js
    sql
    src
    .env.development
    .env.production
    .env.web
    .eslintrc.js
    .gitignore
    .postcssrc.js
    babel.config.js
    build.sh
    package.json
    README.md
    JS vue.config.js
    yarn.lock

JS config.js
1 window.$website = {
2   isDemo: true,
3   isDemoTip: '演示环境不允许操作',
4   title: 'avue-data数据大屏',
5   name: 'WELCOME TO AVUE-DATA',
6   subName: '可视化数据大屏（演示环境-请勿放生产数据）',
7   url: 'https://data.bladex.vip/blade-visual',
8   tabsList: [0, 1, 2, 3, 4],
9   componentsList: [{
10    name: 'test',
11    component: 'testComponents',
12    option: 'testOption',
13    data: true
14  }],
15  baseList: [{ ...
2660  }
2661 }
```

3. 访问前端<http://localhost:8080> 查看效果



Java版后端启动

[Boot版后端启动](#)

[Nacos启动](#)

[Cloud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

blade / Data-Server-Boot

私有库

取消关注 1 点赞 1 派生 1

<> 代码

工单 1

合并请求

软件包

项目

版本发布

百科

动态

设置

BladeX数据大屏后端Boot版本

后端核心依赖：<https://gitee.com/smallc/blade-tool>

大屏前端工程：<https://git.avuejs.com/avue/avue-data>

大屏文档：<https://git.avuejs.com/avue/avue-kanyun>

注△：已购 大屏版授权或联合版授权 便可下载大屏前端与文档

启动部署视频请见网盘(服务器没录进声音，大家先简单参考下，后续会录制完整视频并免费提供，切勿外传)

网盘地址：

提取码：

解压密码：

管理主题

16 提交

1 分支

0 Git标签

2.5 MiB

master

转到文件

添加文件

HTTPS

<https://center.javablade.com/blade/Data-Server-Boo>

...

bladex

7876e20891

修复在线SQL数据源及时生效的功能

4 周前

doc

3.1.1.RELEASE 组件库模块升级

3 个月前

src

修复在线SQL数据源及时生效的功能

4 周前

.editorconfig

数据大屏新版发布

11 个月前

.gitignore

数据大屏新版发布

11 个月前

Dockerfile

限流模块新增IP白名单

5 个月前

LICENSE

数据大屏新版发布

11 个月前

README.md

数据大屏新版发布

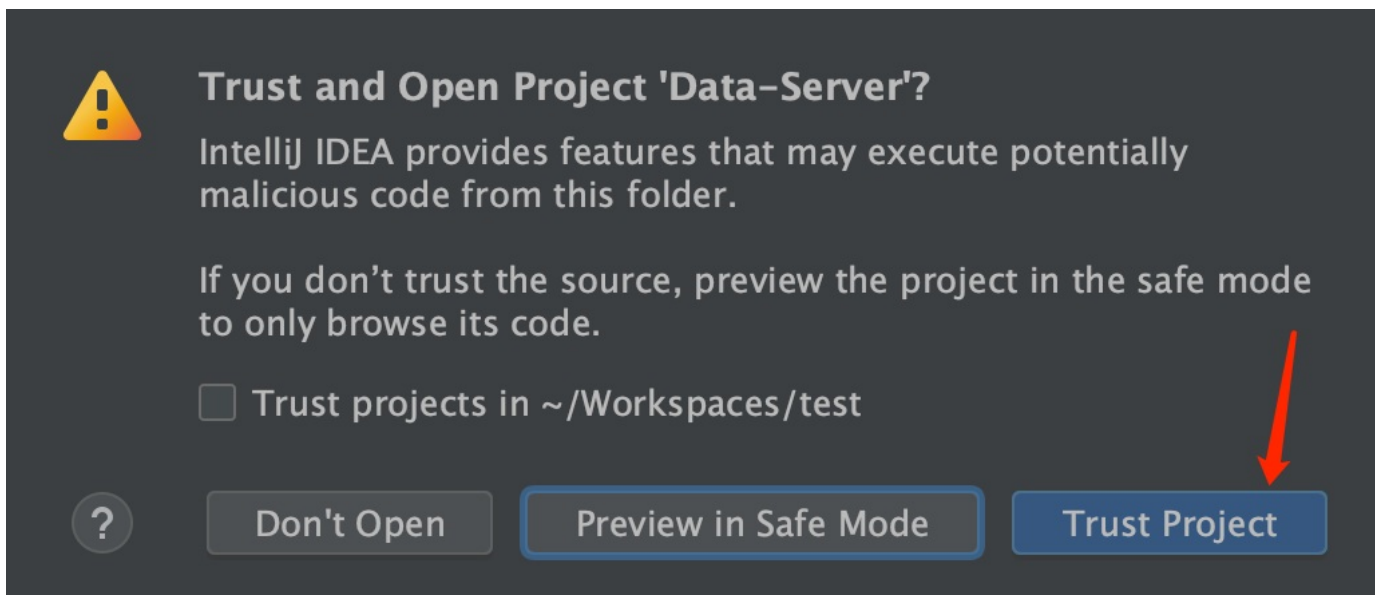
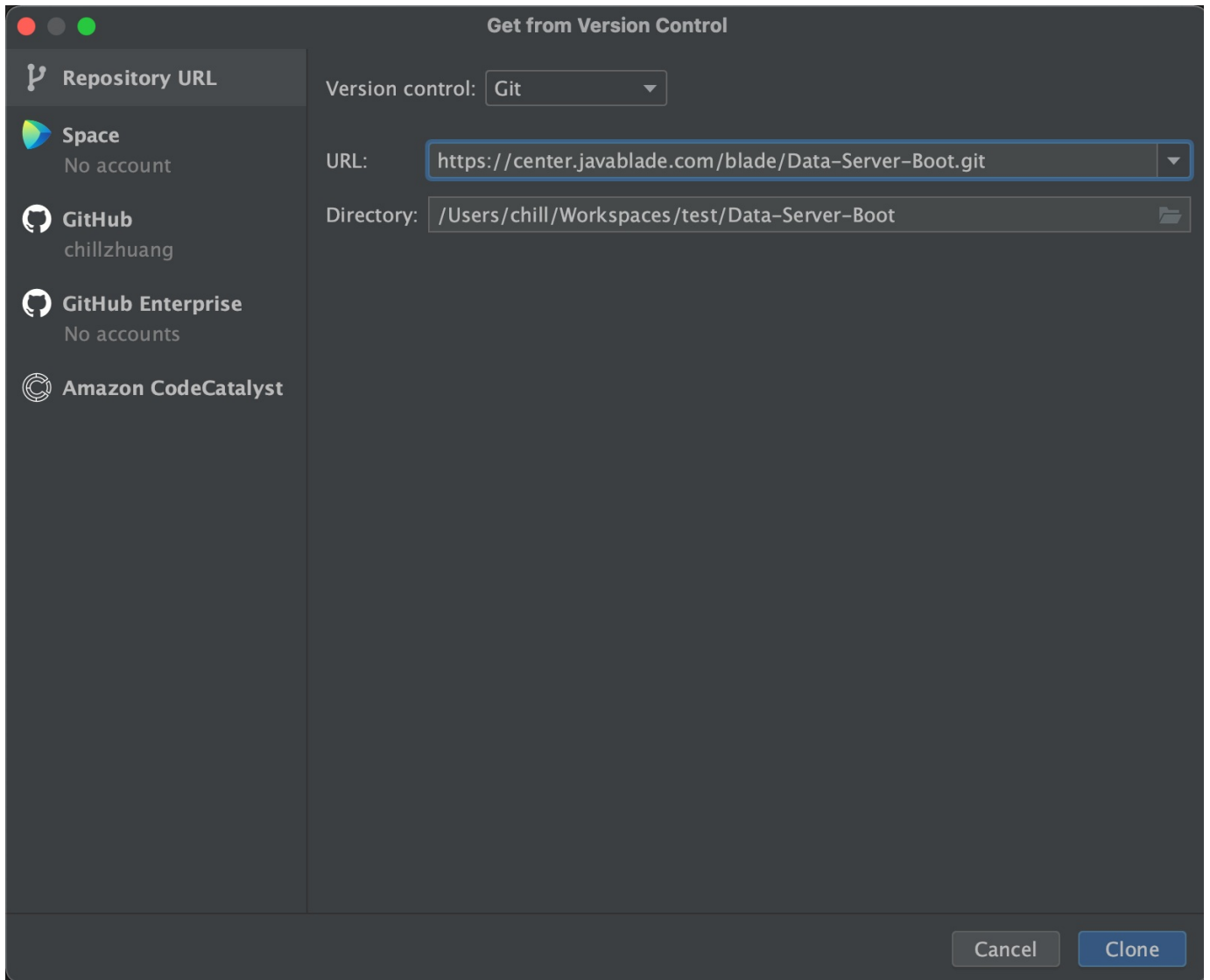
11 个月前

pom.xml

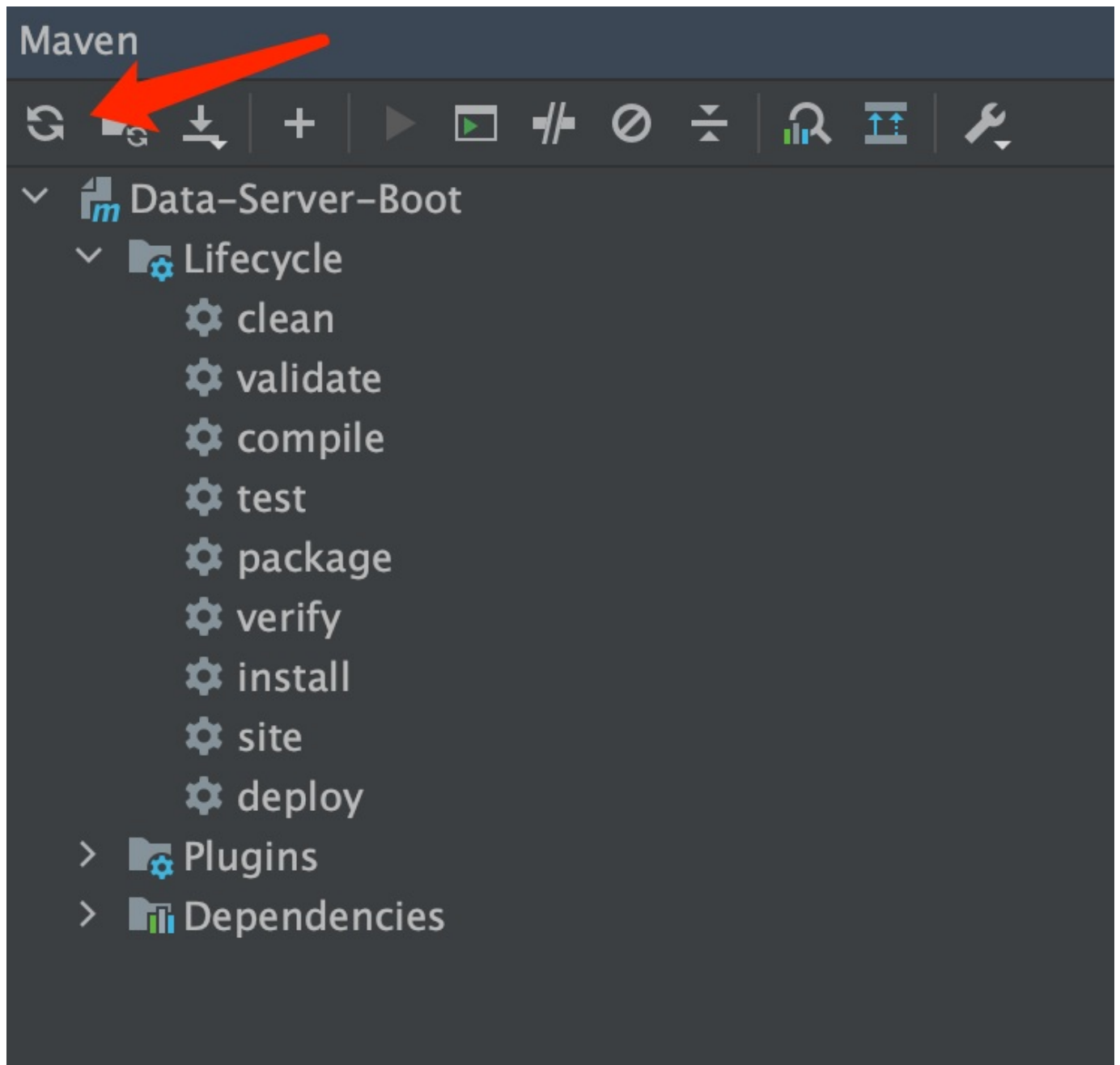
3.1.1.RELEASE 组件库模块升级

3 个月前

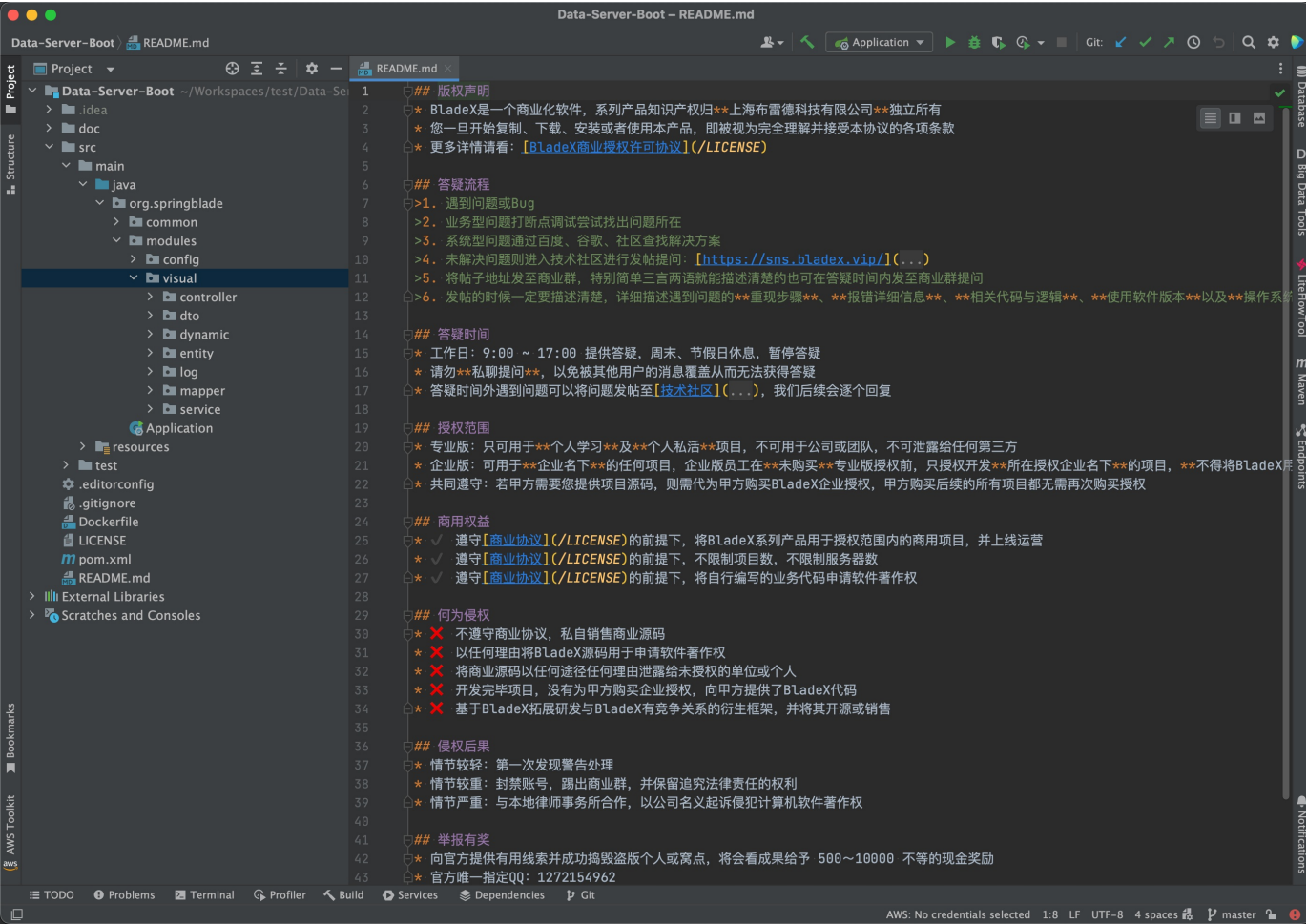
- 3. 导入工程



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

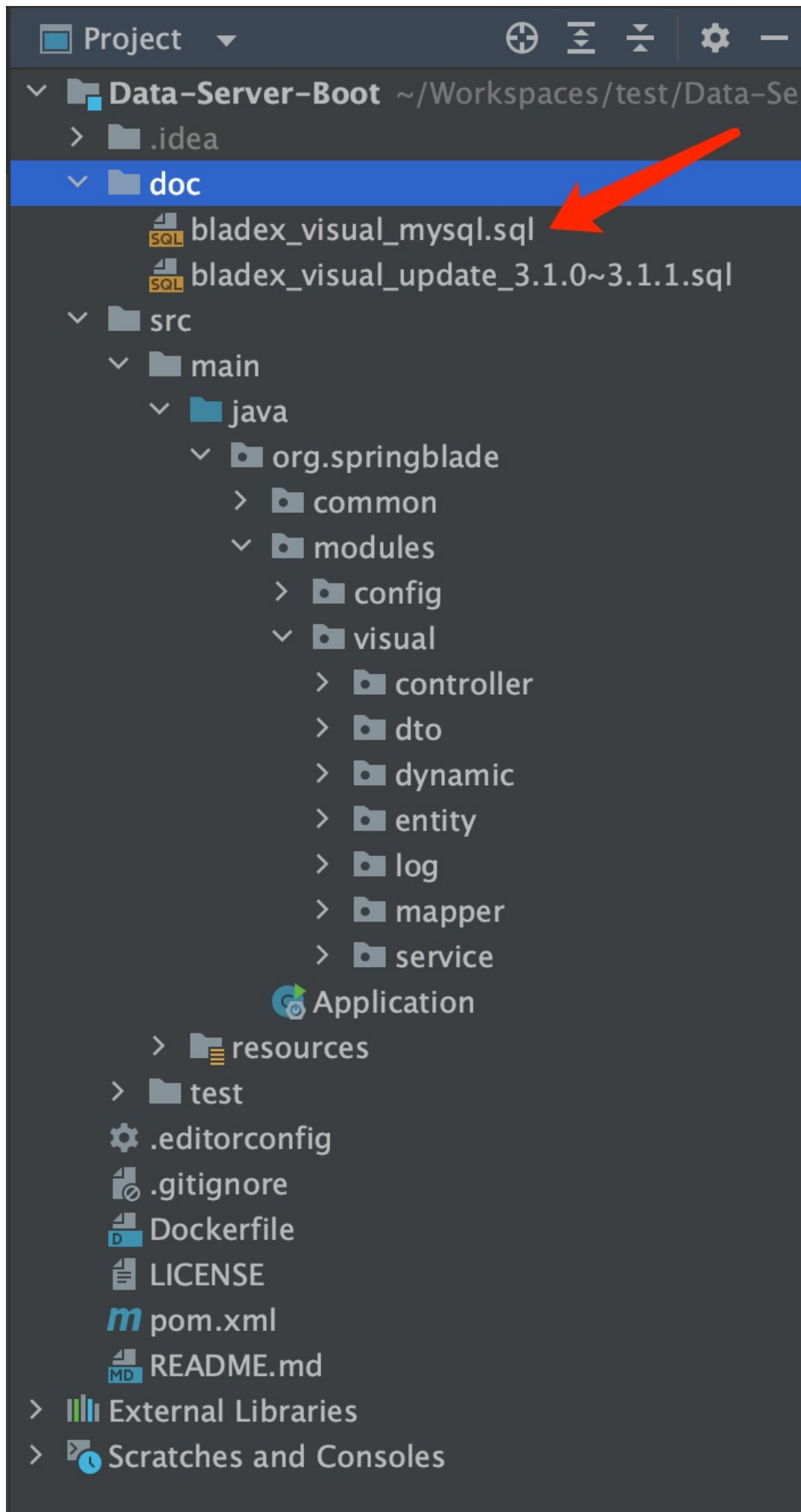


5. 工程构建完毕后，便可以看到核心服务了

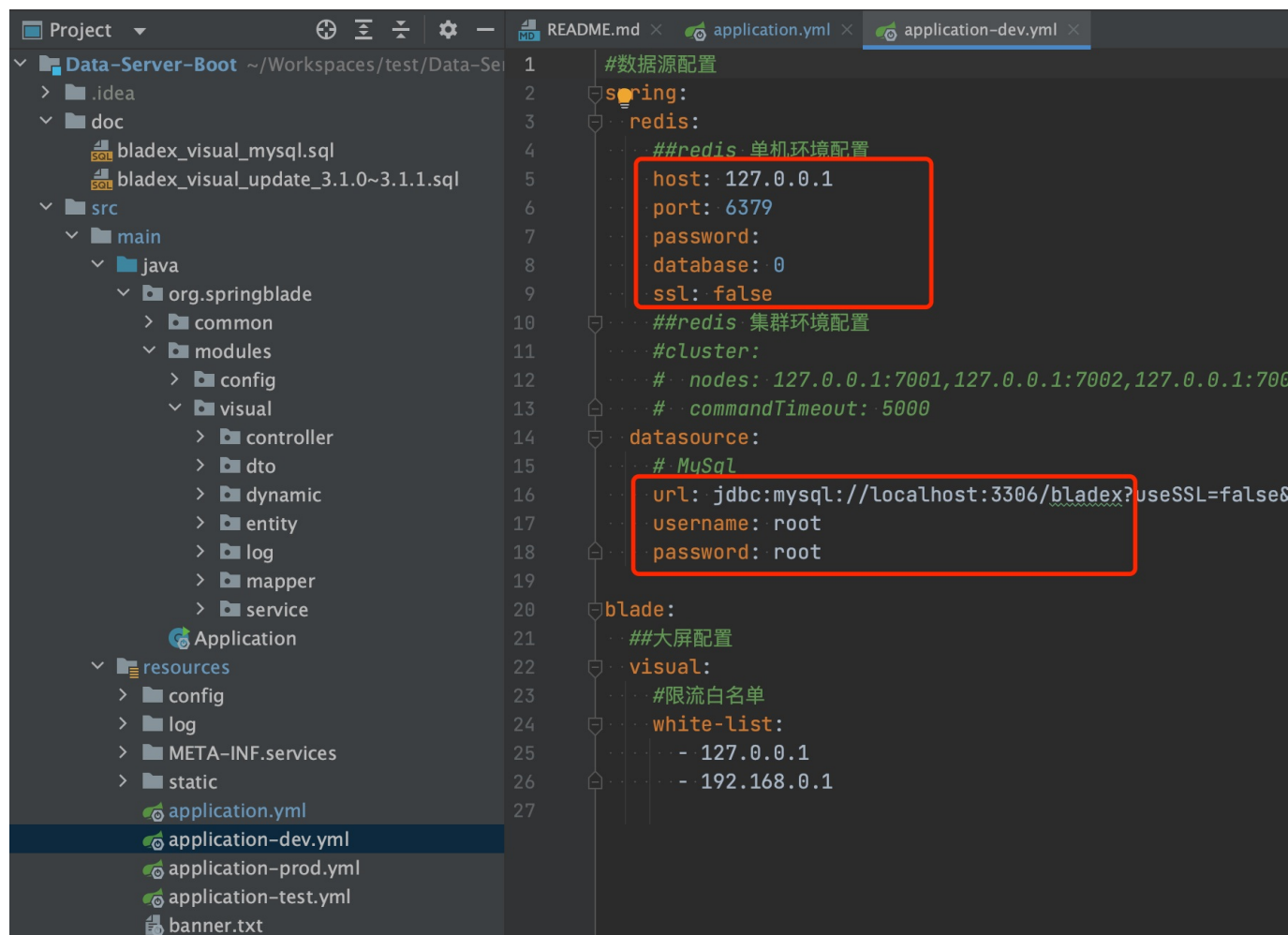


二、数据库配置

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)

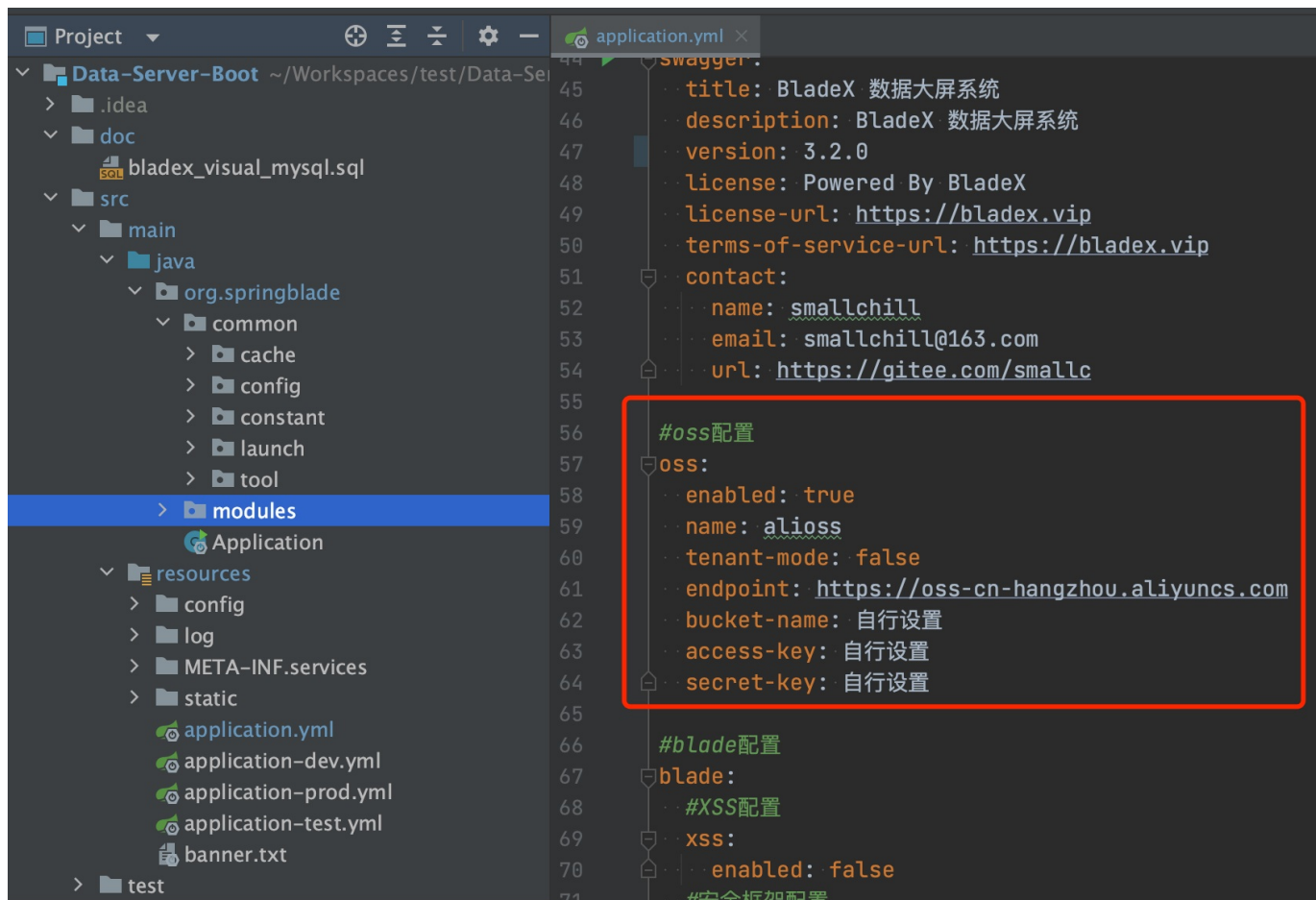


2. 到对应配置文件将数据库信息配置准确



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

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.

MINIO

STANDARD

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](#).

MINIO

ENTERPRISE

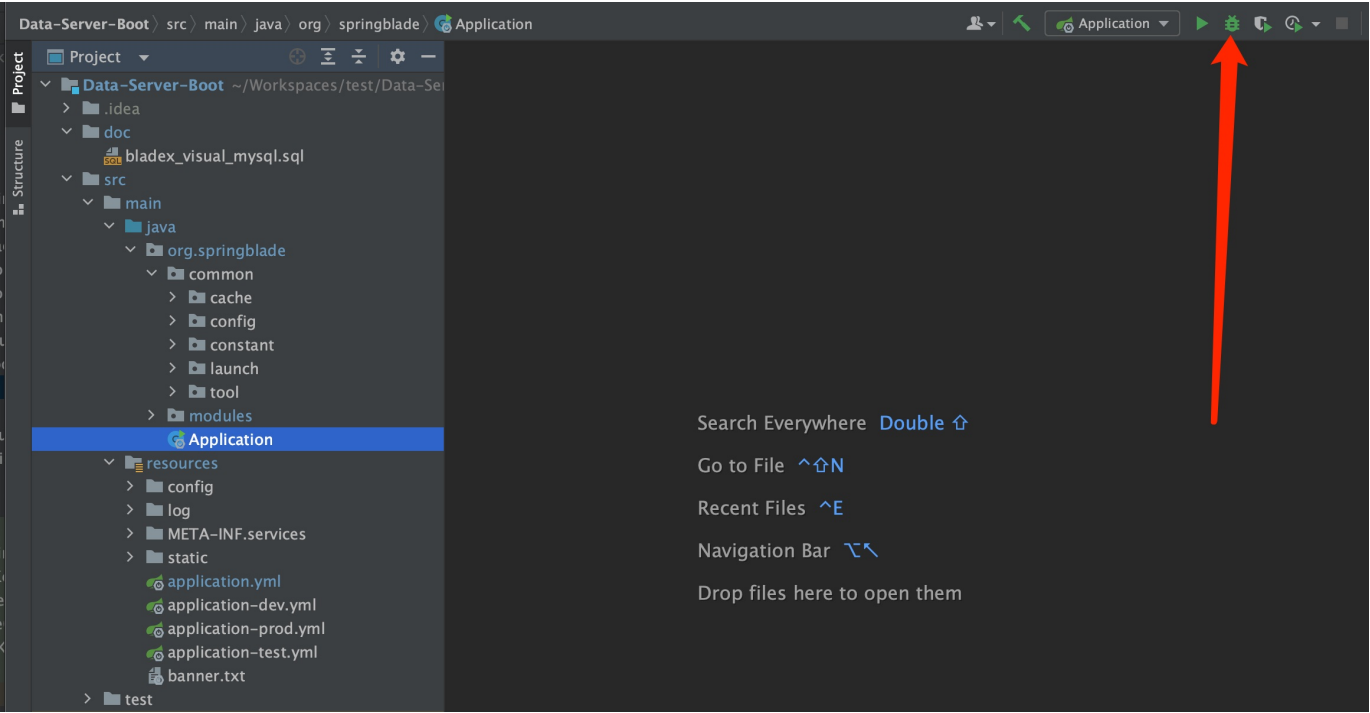
Designed for mission critical environments where both a license and strict SLAs are required. The Enterprise version is fully featured but comes with additional capabilities over the Standard license. To learn more about the features and capabilities available to commercial license holders click [here](#).

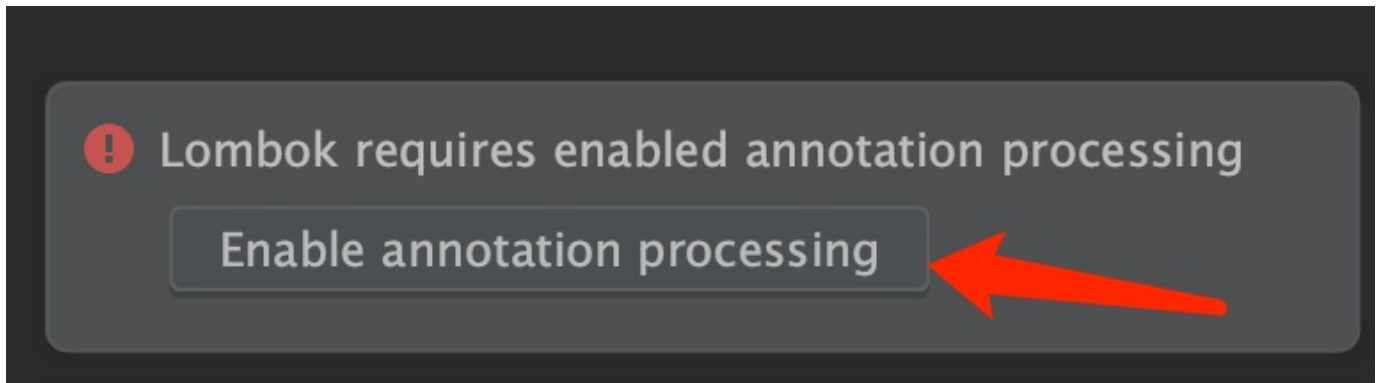
FEATURES

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

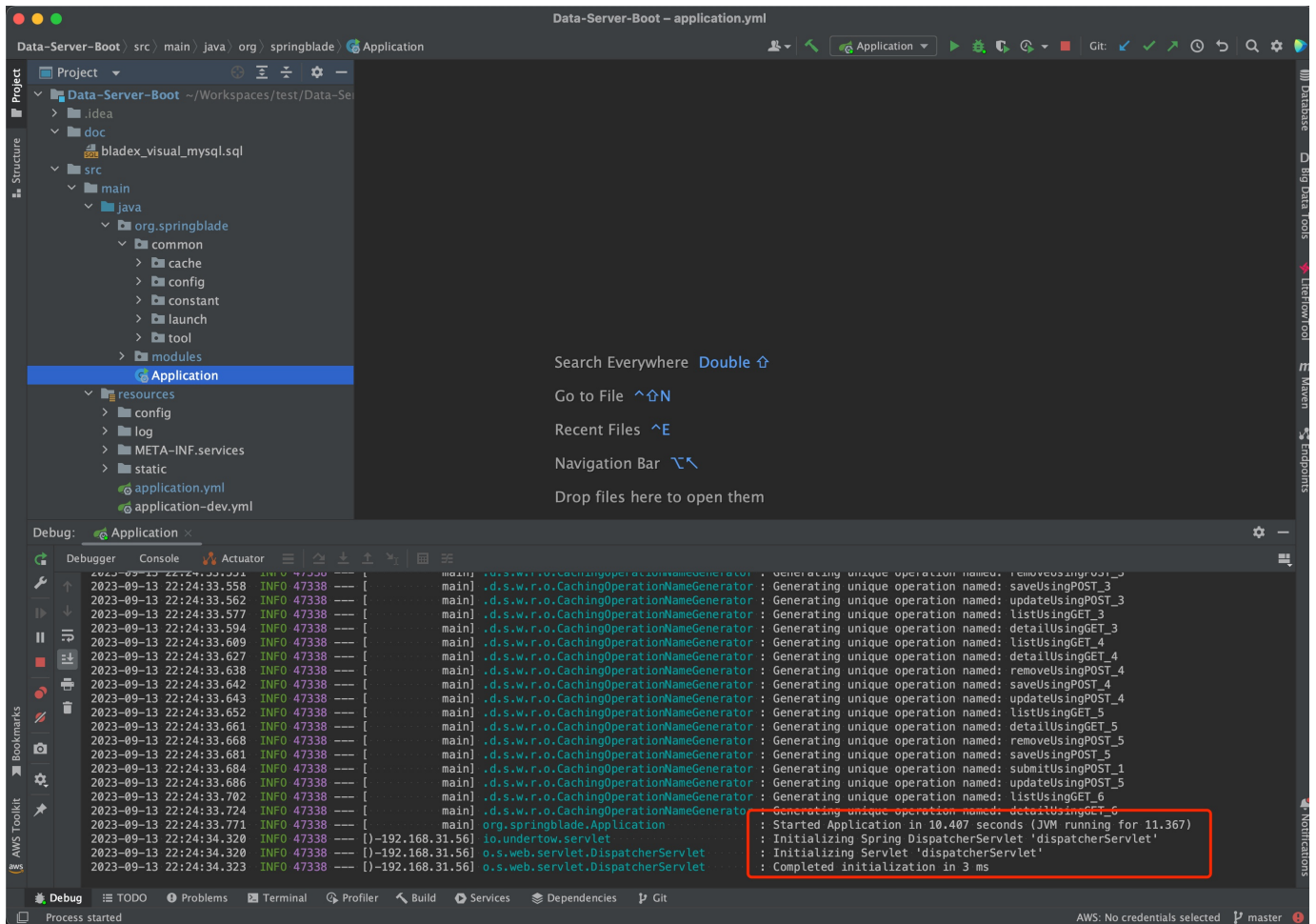
四、工程启动

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

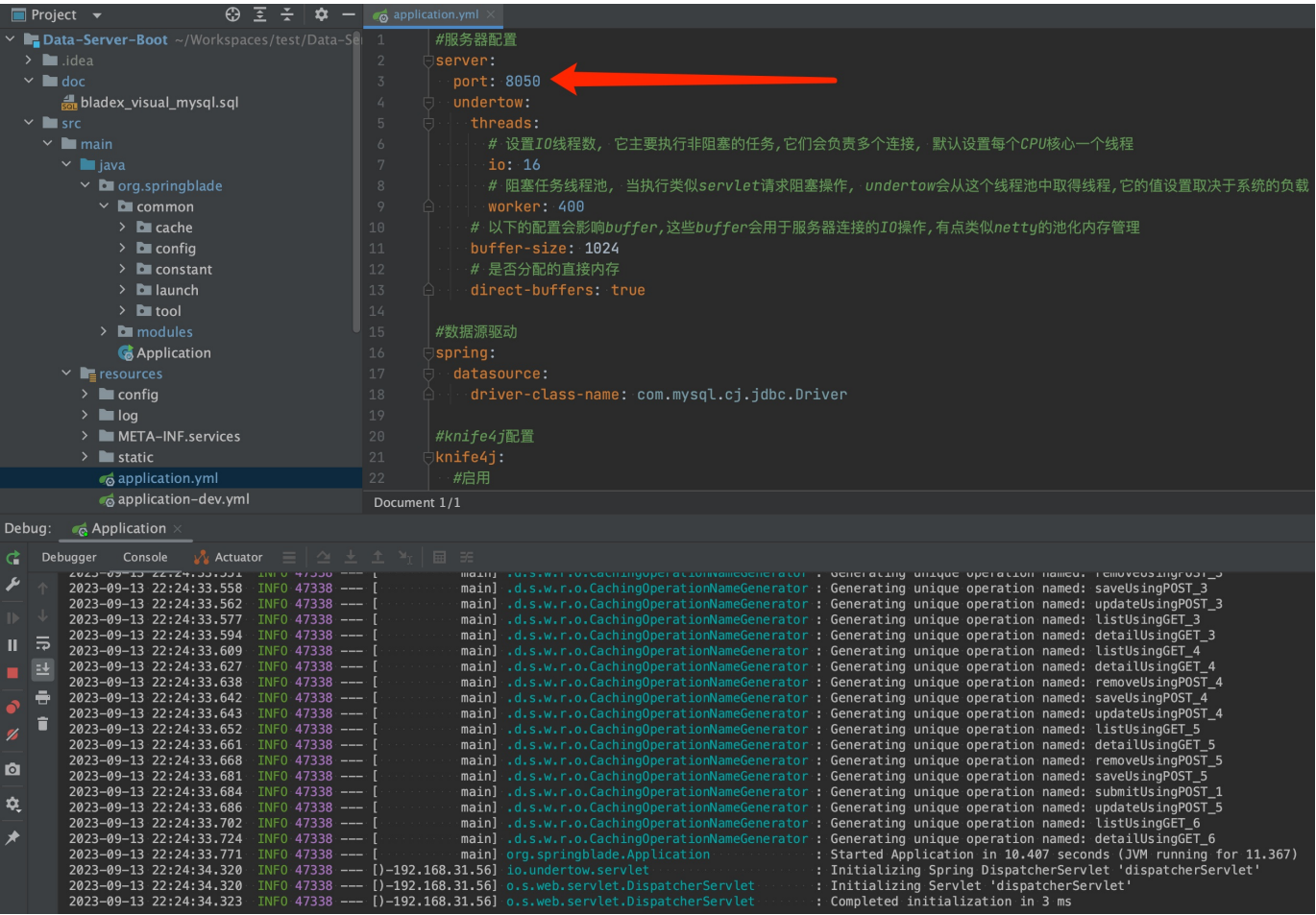




2. 若配置无误，我们就可以看到启动成功的日志



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

→ ↺ 🏠 ⓘ localhost:8050/blade-visual/category/list

```
// 20230913222530
// 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": "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：

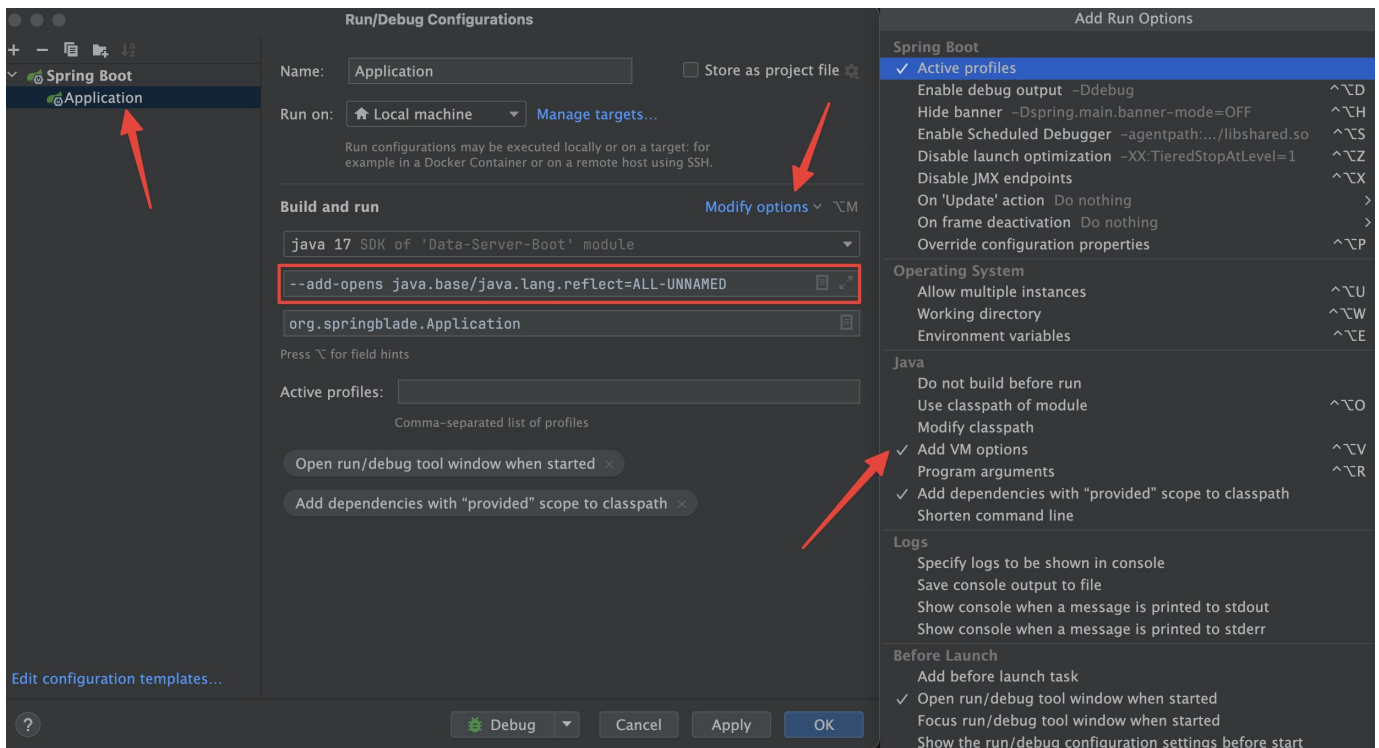
```
Caused by: java.lang.reflect.InaccessibleObjectException: Unable to make field
protected java.lang.reflect.InvocationHandler java.lang.reflect.Proxy.h accessi
ble: module java.base does not "opens java.lang.reflect" to unnamed module @23c
30a20
    at java.base/java.lang.reflect.AccessibleObject.checkCanSetAccessible(Acces
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(BeanWrapper.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(BaseWrapper.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(SqlLogInterceptor.java:46)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy216.query(Unknown Source)
    at org.apache.ibatis.executor.SimpleExecutor.doQuery(SimpleExecutor.java:65)
    at org.apache.ibatis.executor.BaseExecutor.queryFromDatabase(BaseExecutor.java:336)
    at org.apache.ibatis.executor.BaseExecutor.query(BaseExecutor.java:158)
    at com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor.intercept(MybatisPlusInterceptor.java:81)
    at org.apache.ibatis.plugin.Plugin.invoke(Plugin.java:59)
    at jdk.proxy2/jdk.proxy2.$Proxy215.query(Unknown Source)
    at org.apache.ibatis.session.defaults.DefaultSqlSession.selectList(DefaultSqlSession.java:154)
    ... 115 common frames omitted

```

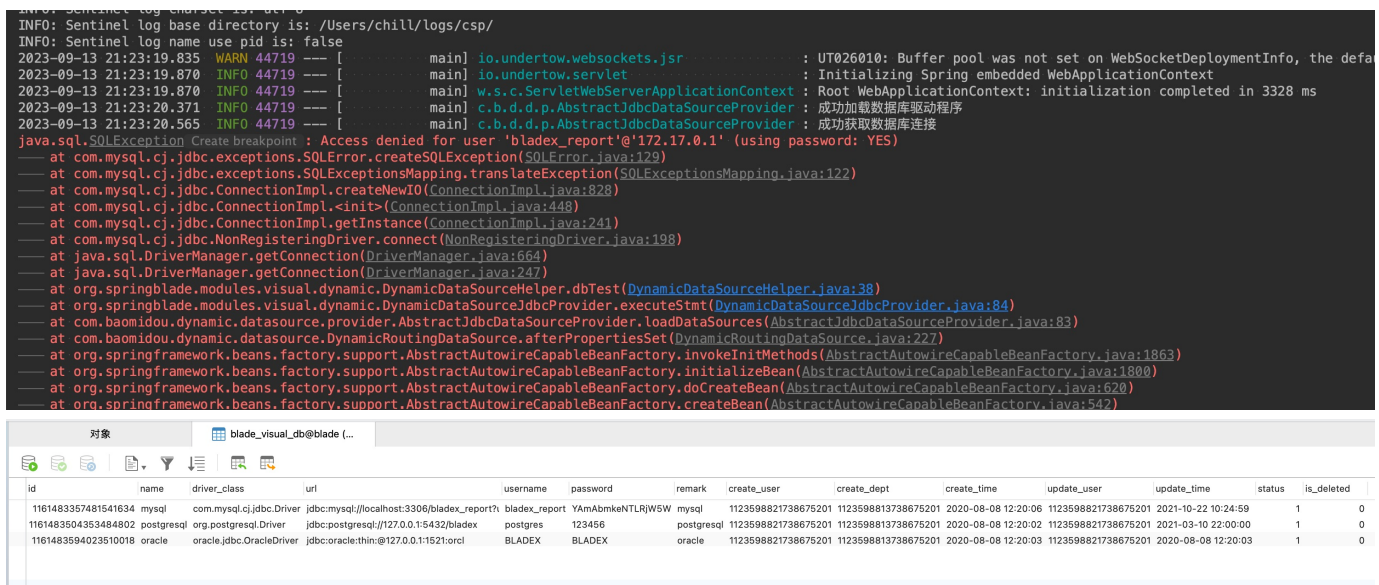
6. 遇到上述情况，在启动类增加命令 `--add-opens java.base/java.lang.reflect=ALL-UNNAMED` 即可



7. JAR 包启动时也需要加入此配置，具体命令如下

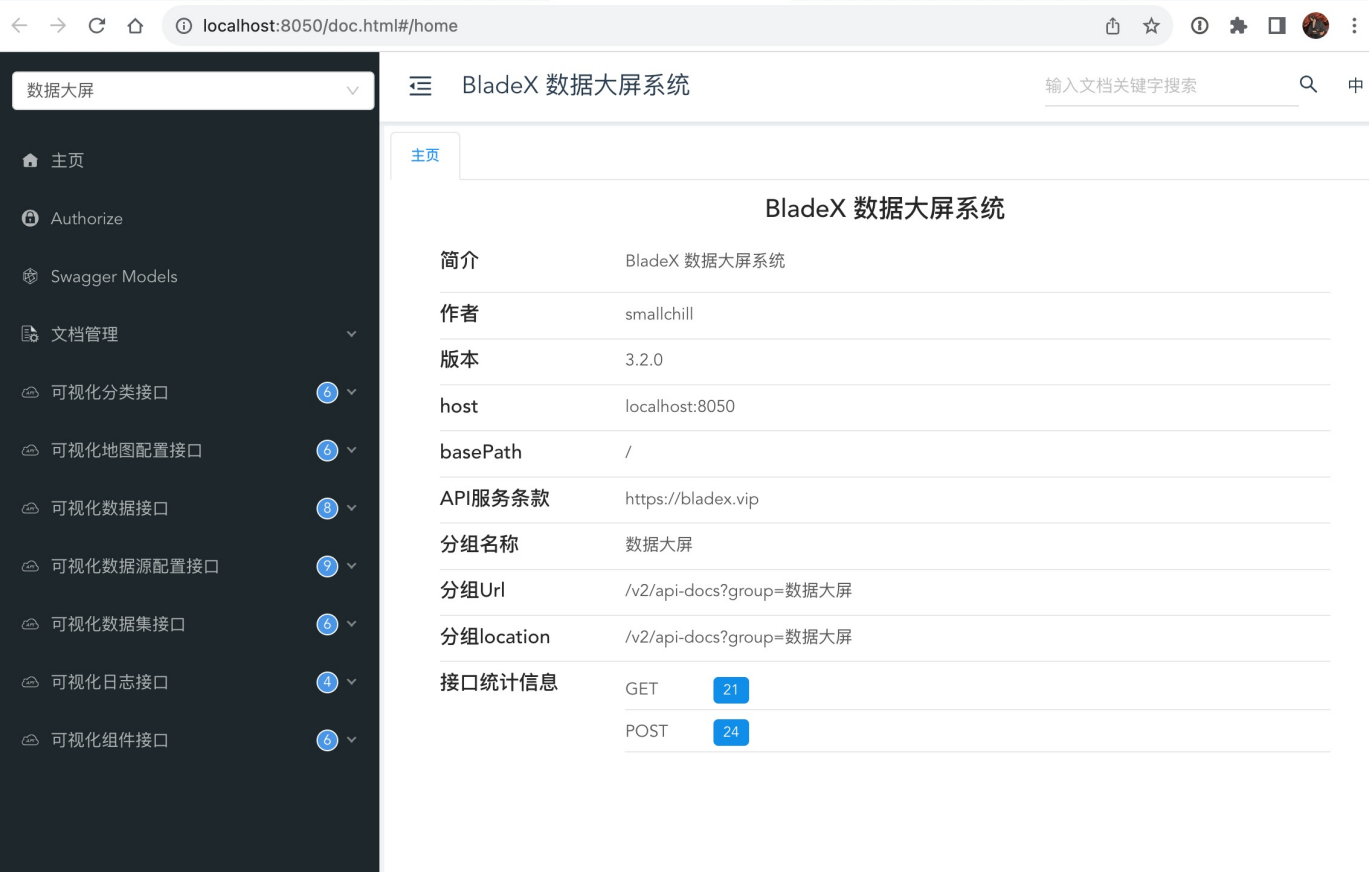
```
java --add-opens java.base/java.lang.reflect=ALL-UNNAMED -jar your-application.jar
```

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

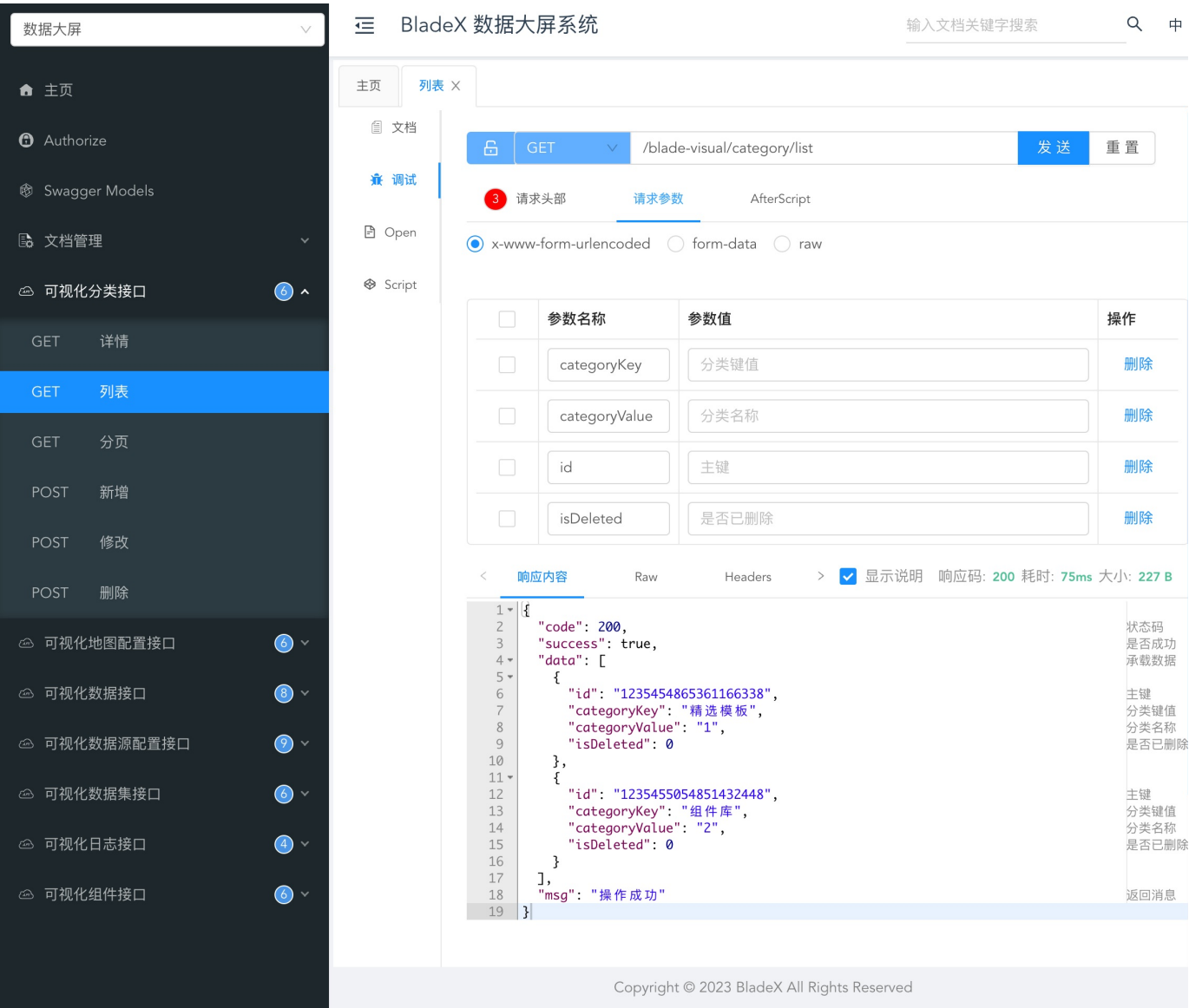
3 请求头部

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

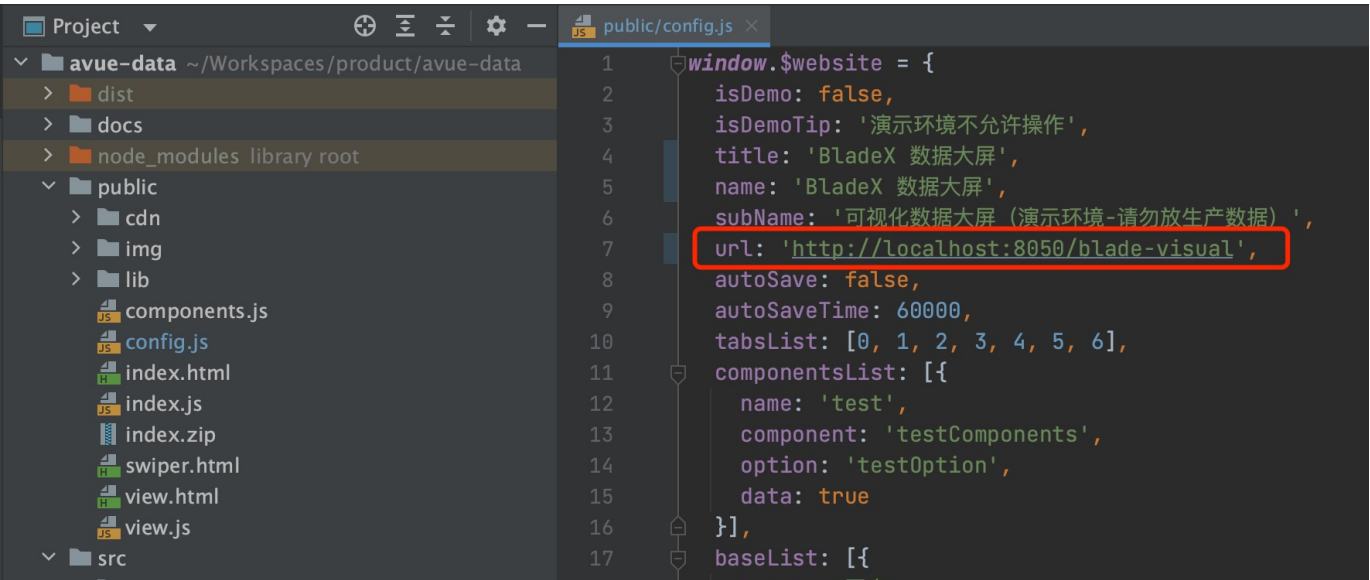
Copyright © 2023 BladeX All Rights Reserved



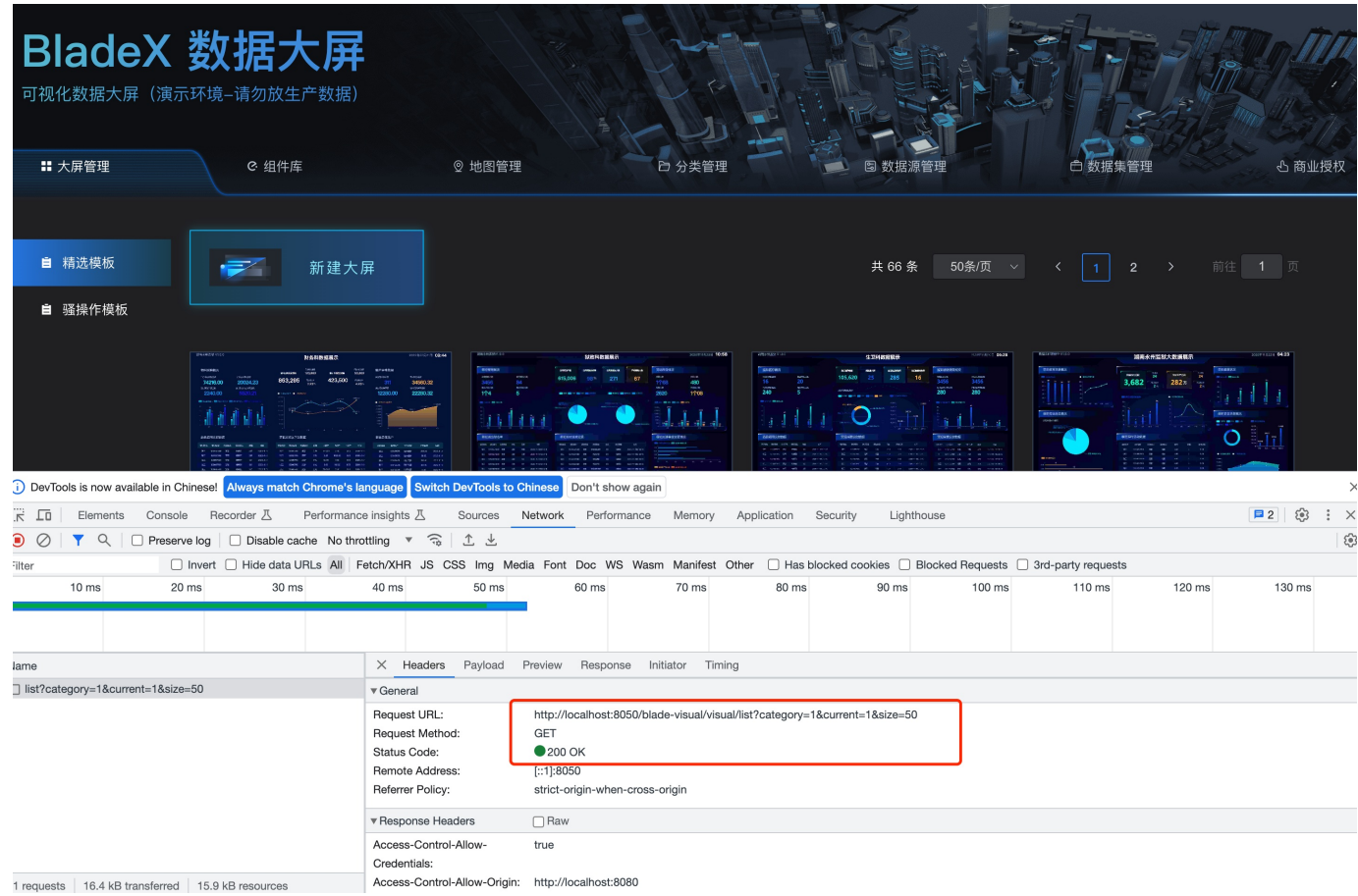
3. 至此后端部署全流程结束

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>



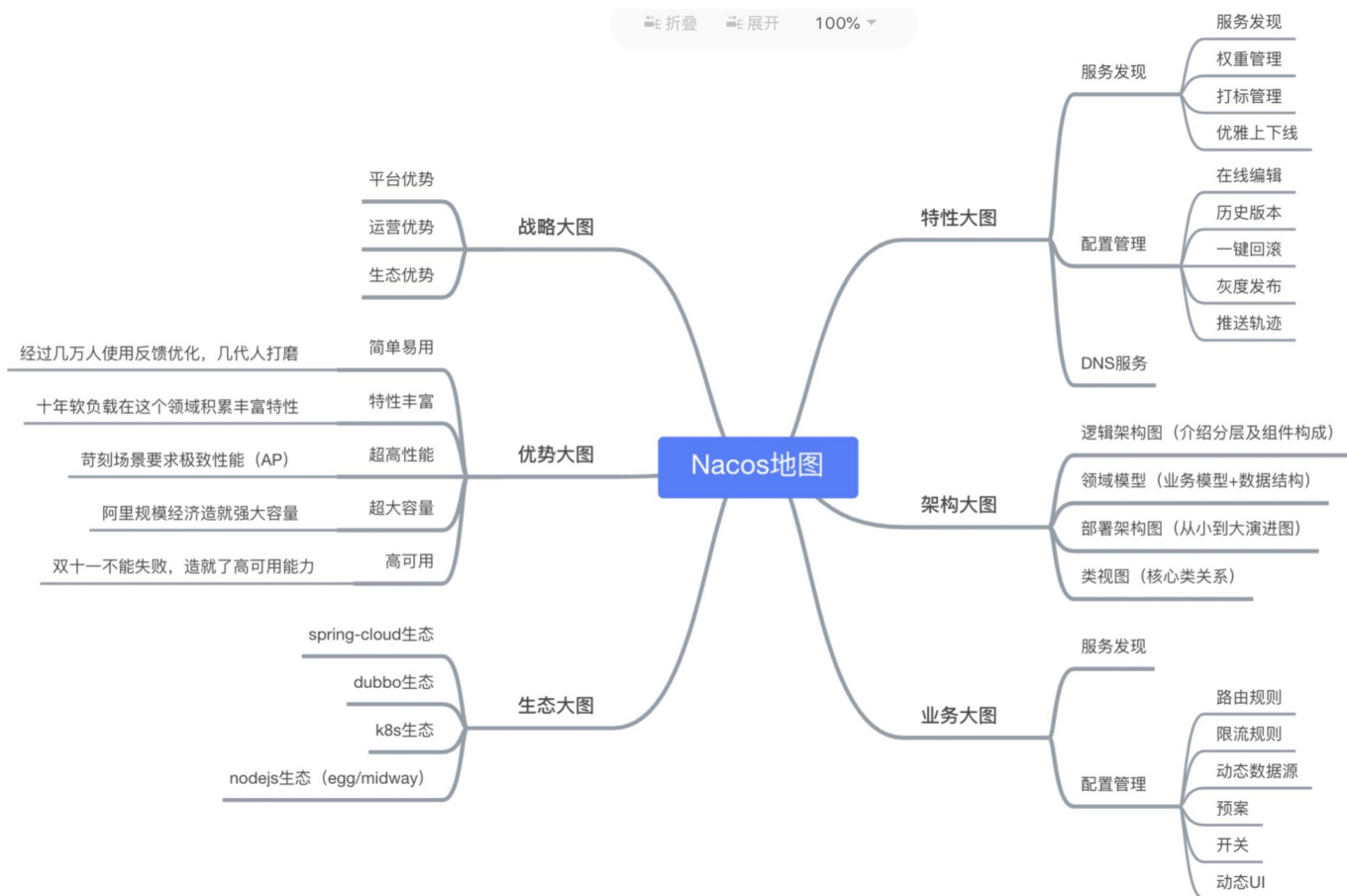
2. 刷新前端大屏可以看到数据获取成功



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>

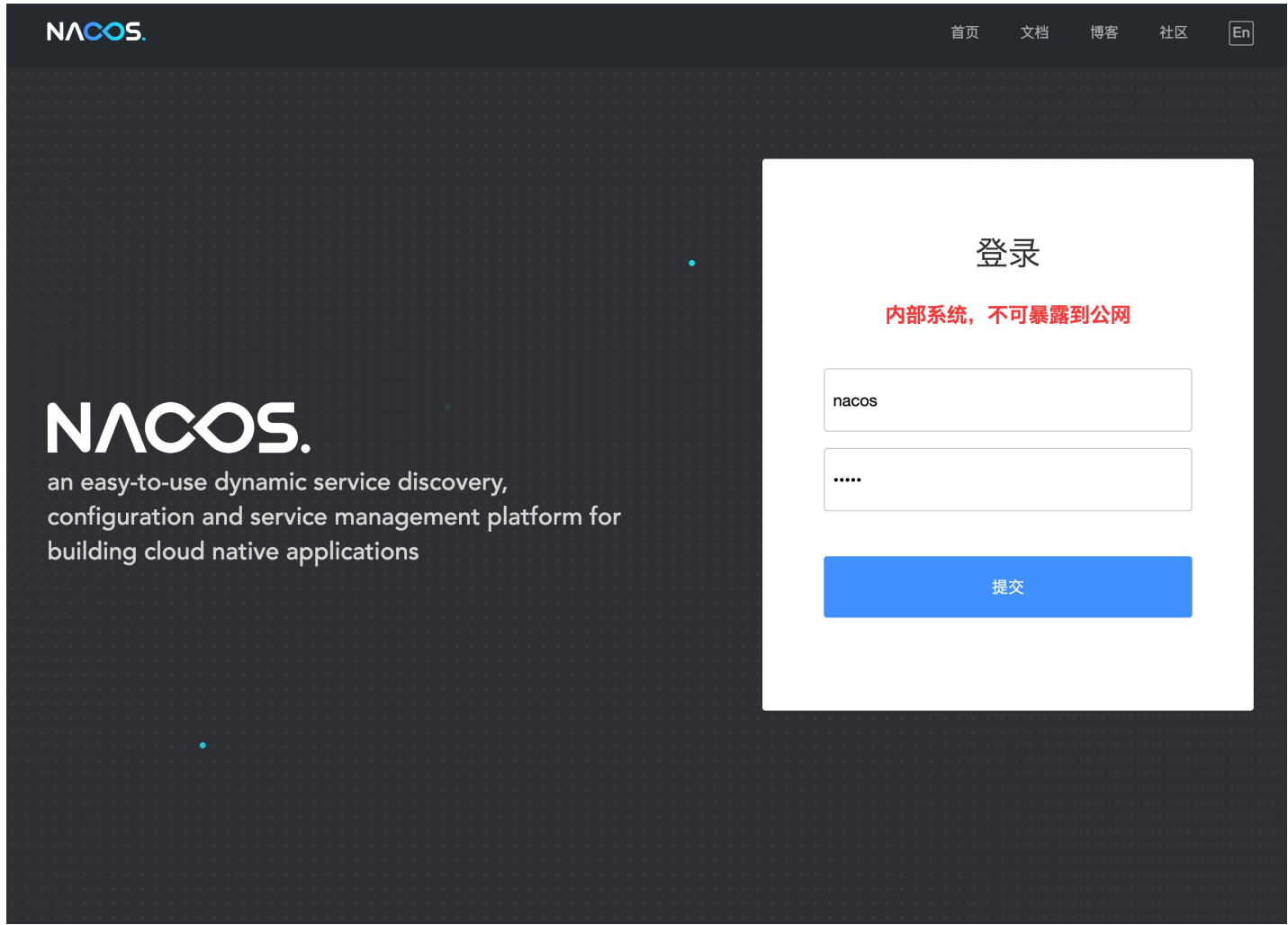
- 帐号密码都为：nacos

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 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

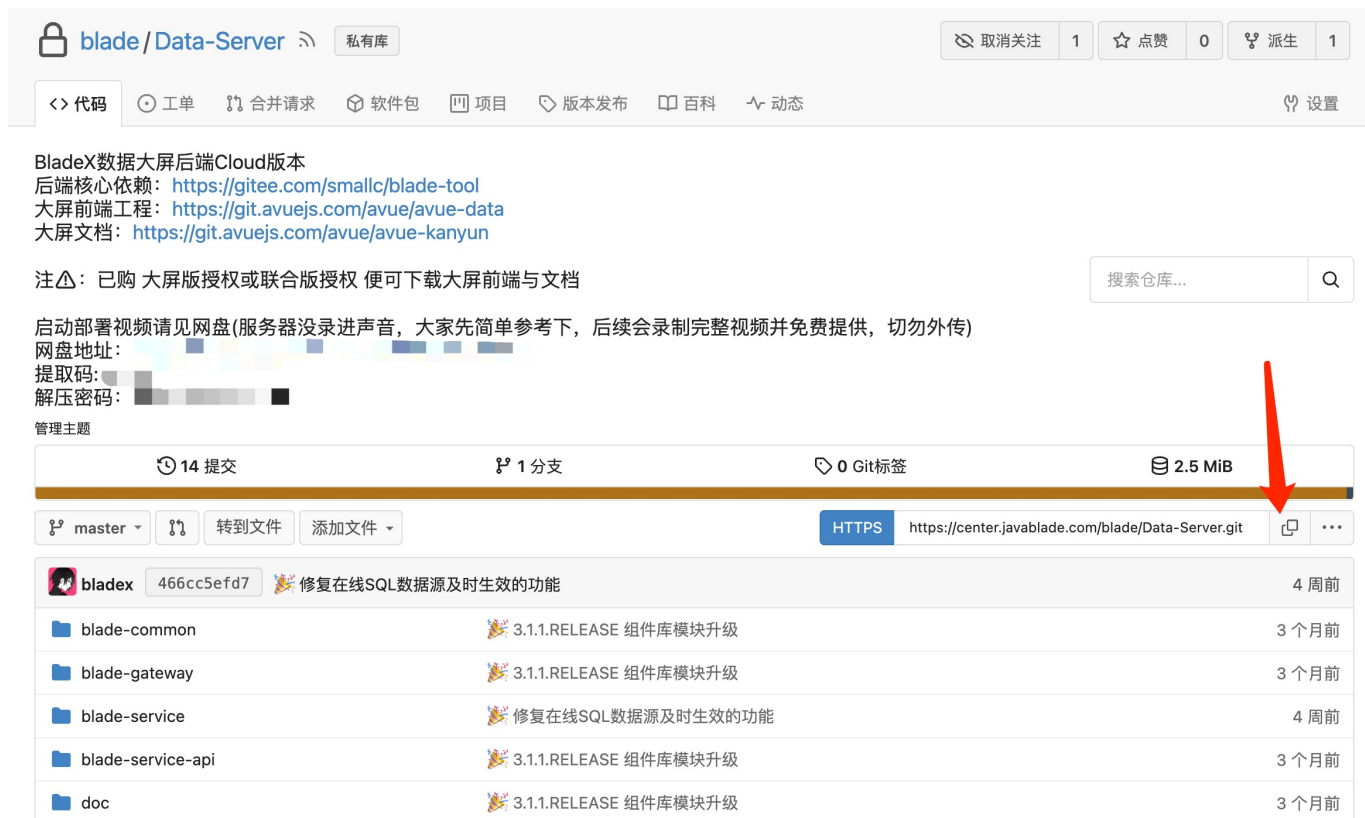
每页显示: 10

< 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



blade / Data-Server 私有库

取消关注 1 点赞 0 派生 1

<> 代码 工单 合并请求 软件包 项目 版本发布 百科 动态 设置

BladeX数据大屏后端Cloud版本
后端核心依赖：<https://gitee.com/smallc/blade-tool>
大屏前端工程：<https://git.avuejs.com/avue/avue-data>
大屏文档：<https://git.avuejs.com/avue/avue-kanyun>

注△：已购 大屏版授权或联合版授权 便可下载大屏前端与文档

启动部署视频请见网盘(服务器没录进声音，大家先简单参考下，后续会录制完整视频并免费提供，切勿外传)
网盘地址：
提取码：
解压密码：

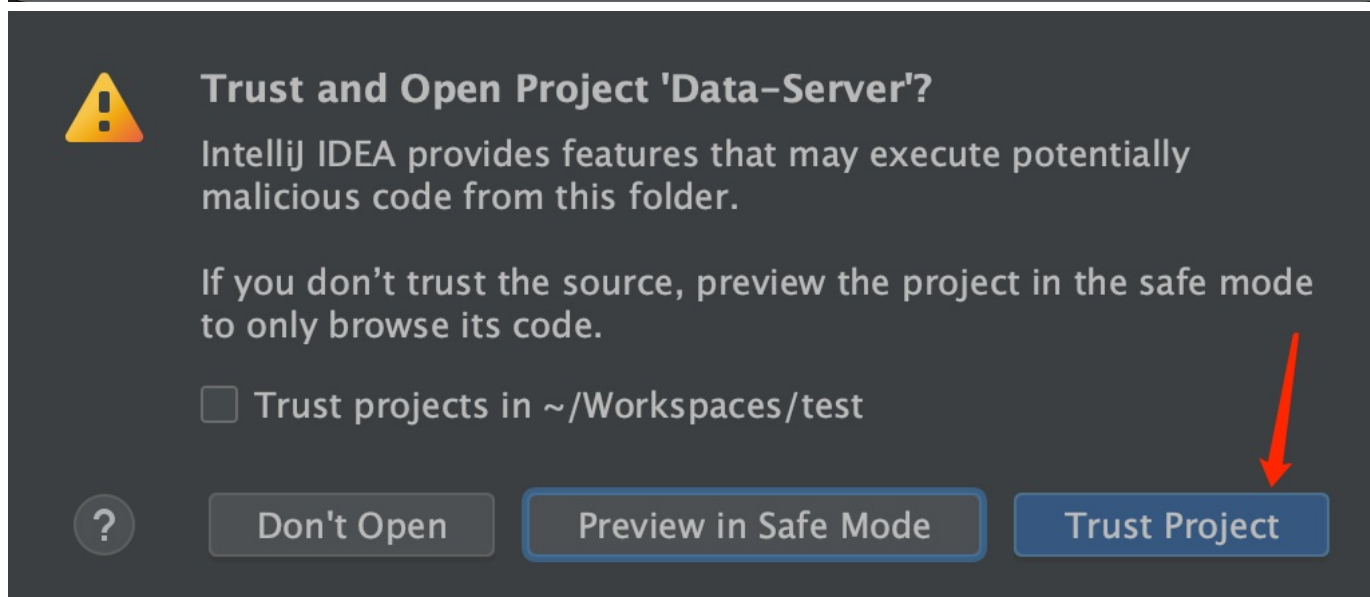
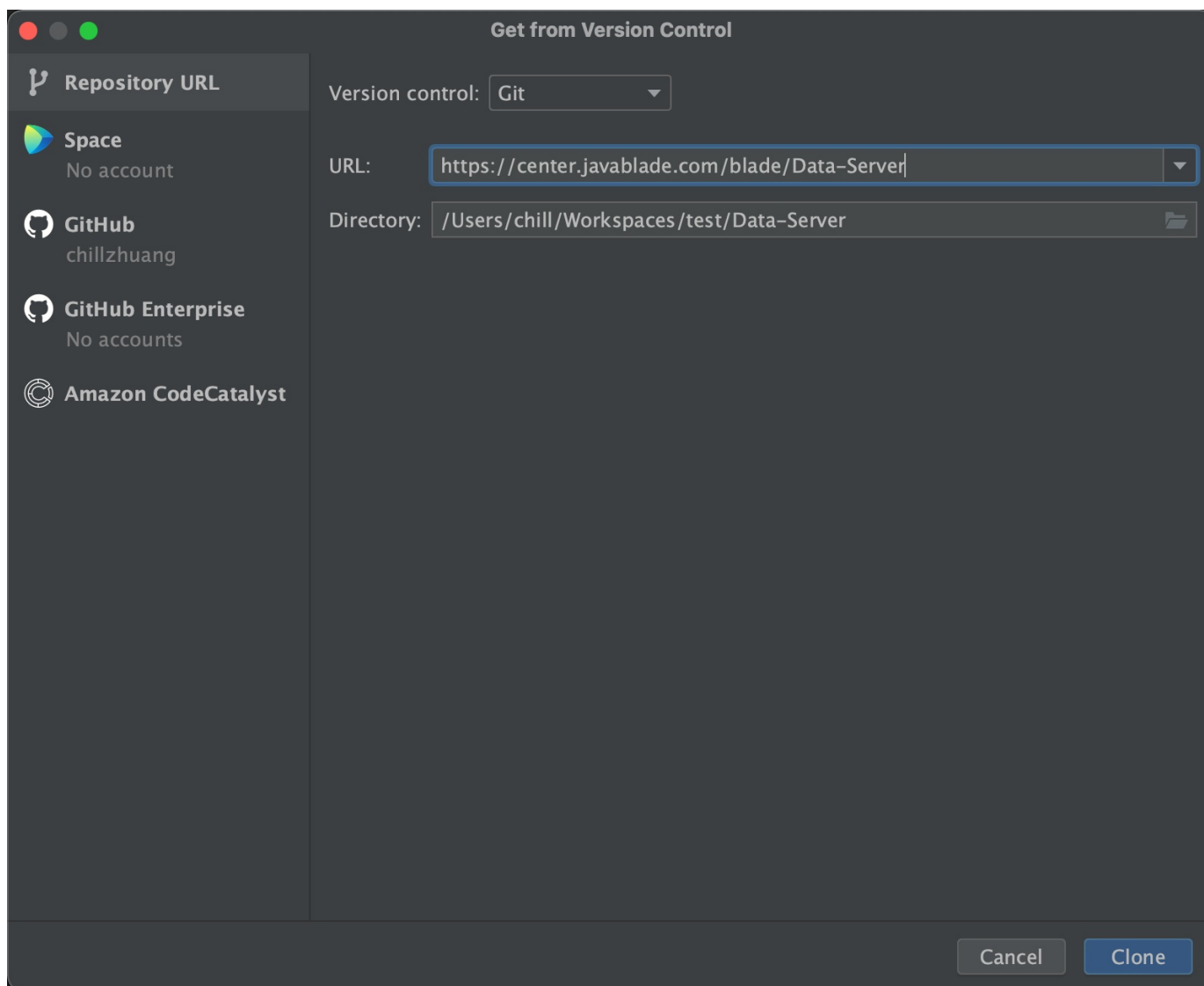
管理主题

14 提交 1 分支 0 Git标签 2.5 MiB

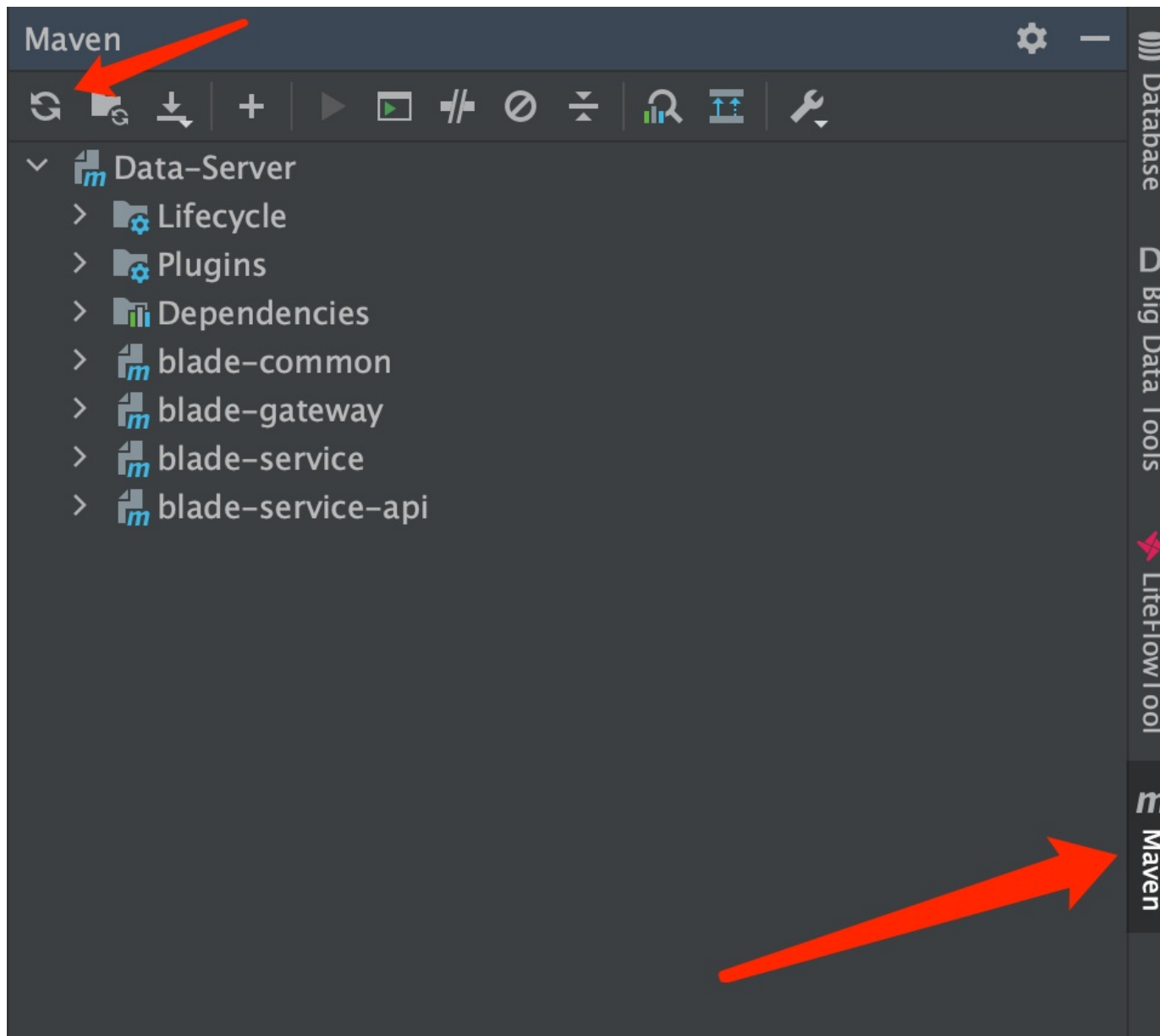
master 转到文件 添加文件 HTTPS <https://center.javablade.com/blade/Data-Server.git>

bladex	466cc5efd7	修复在线SQL数据源及时生效的功能	4 周前
blade-common		3.1.1.RELEASE 组件库模块升级	3 个月前
blade-gateway		3.1.1.RELEASE 组件库模块升级	3 个月前
blade-service		修复在线SQL数据源及时生效的功能	4 周前
blade-service-api		3.1.1.RELEASE 组件库模块升级	3 个月前
doc		3.1.1.RELEASE 组件库模块升级	3 个月前

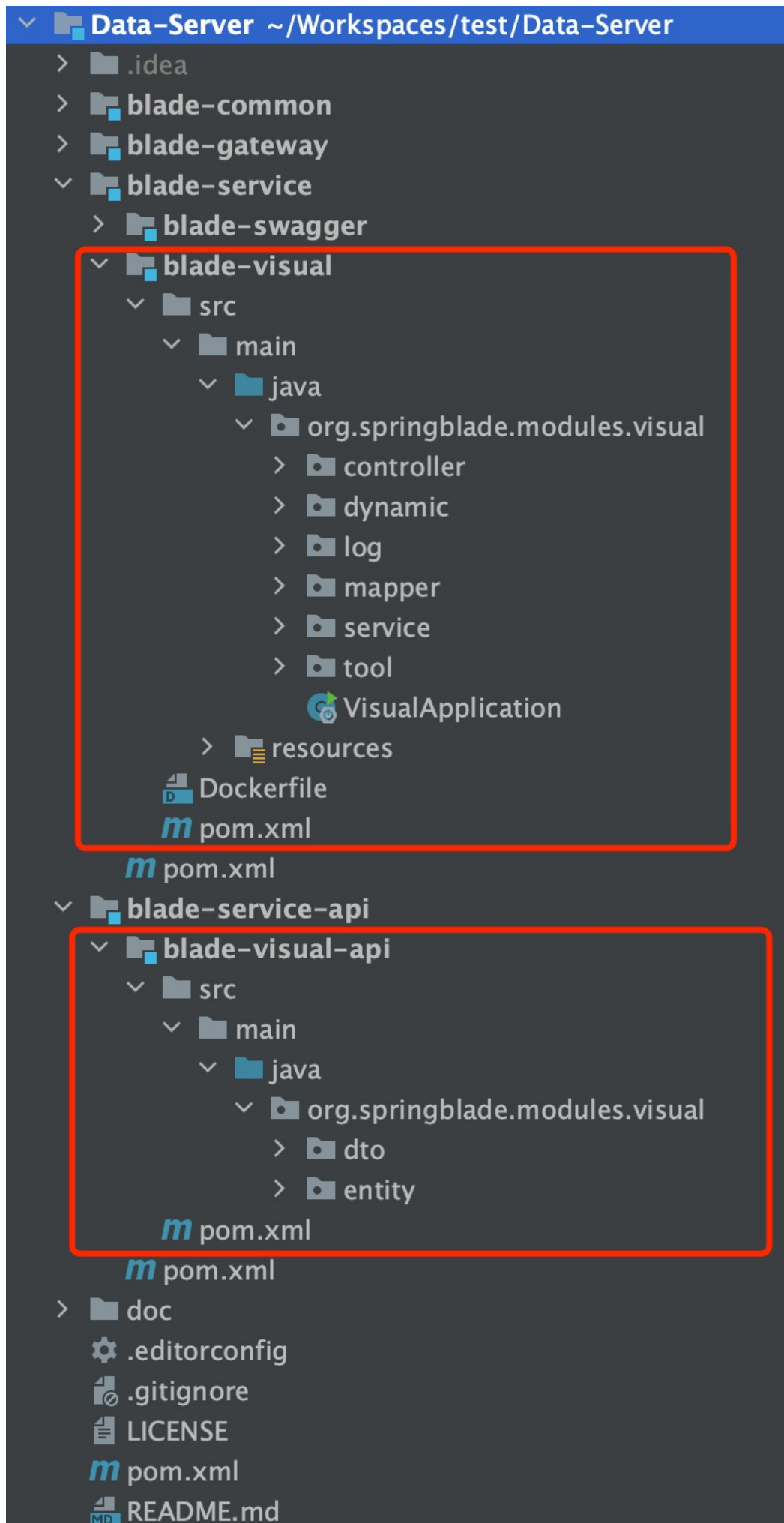
3. 导入工程



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

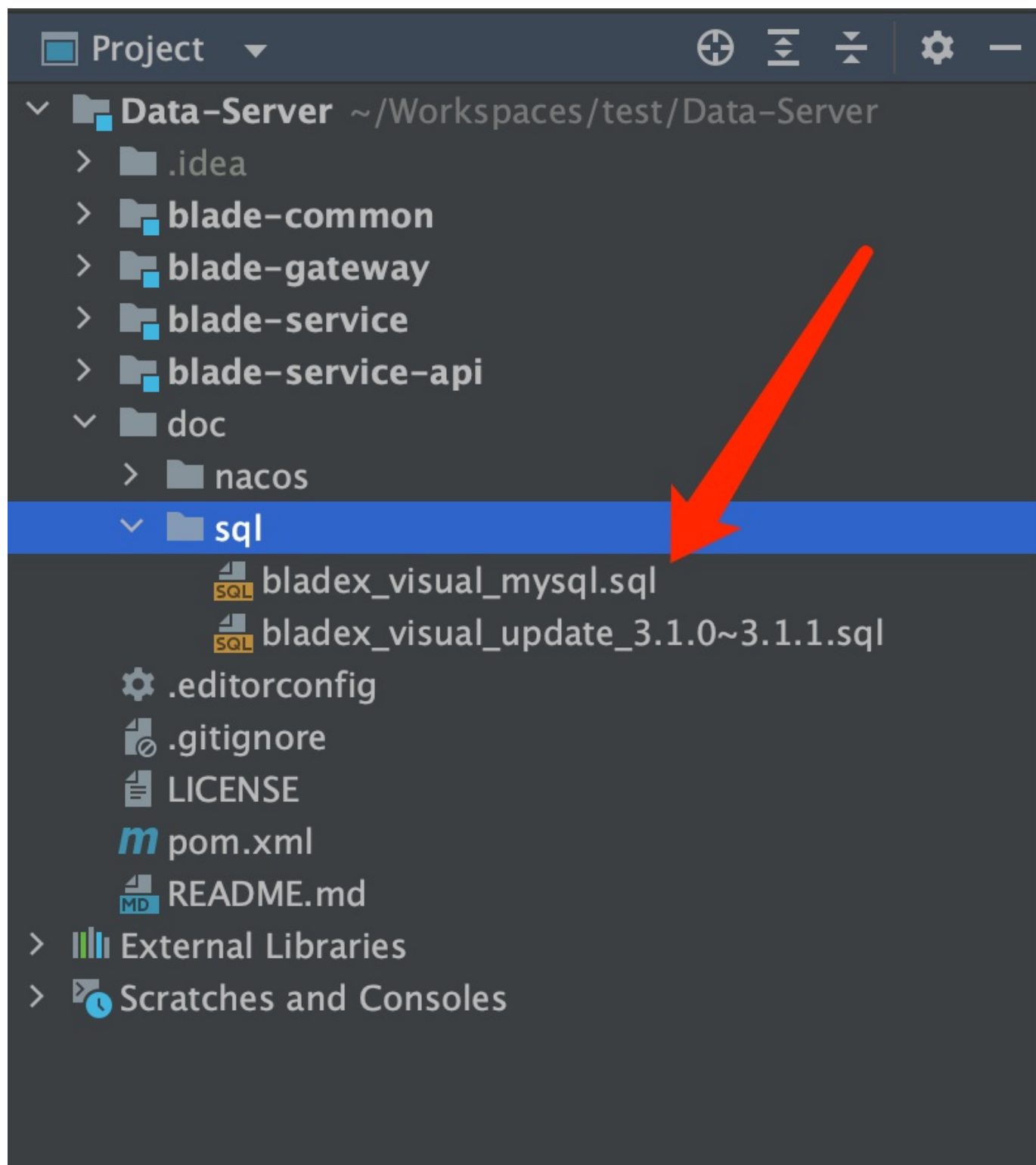


5. 工程构建完毕后，便可以看到核心服务了

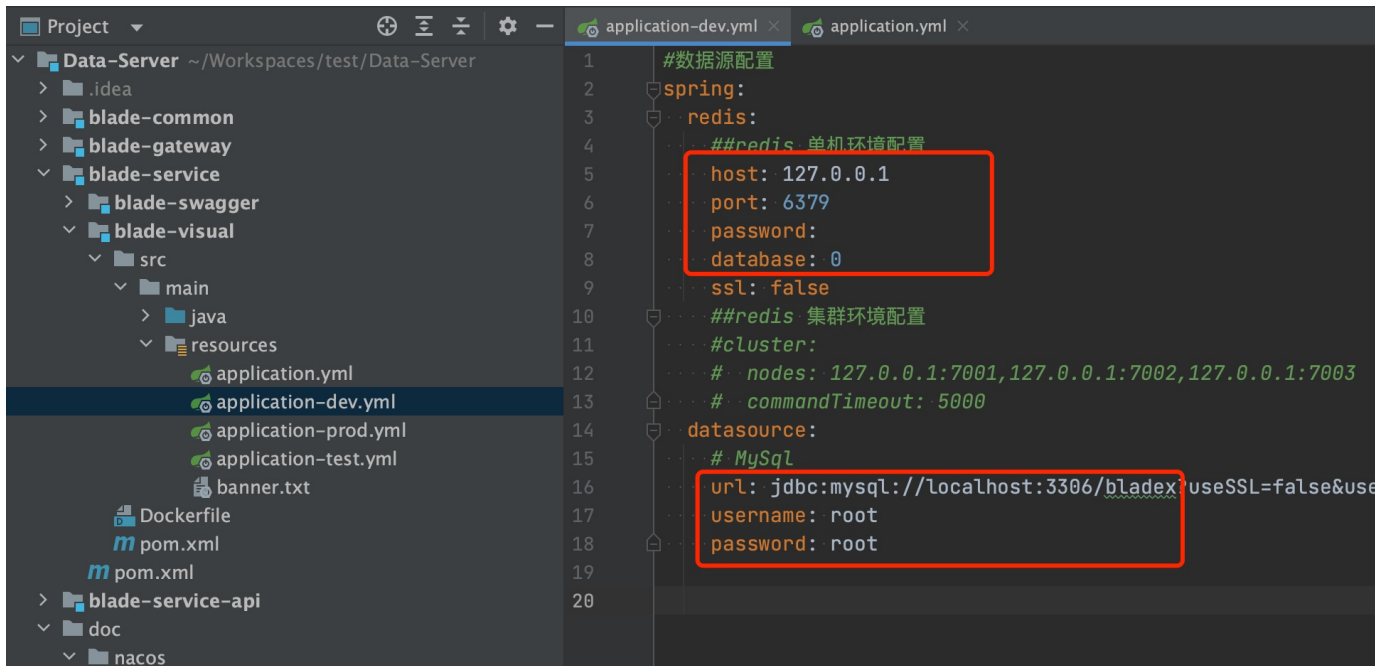


二、数据库配置

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)

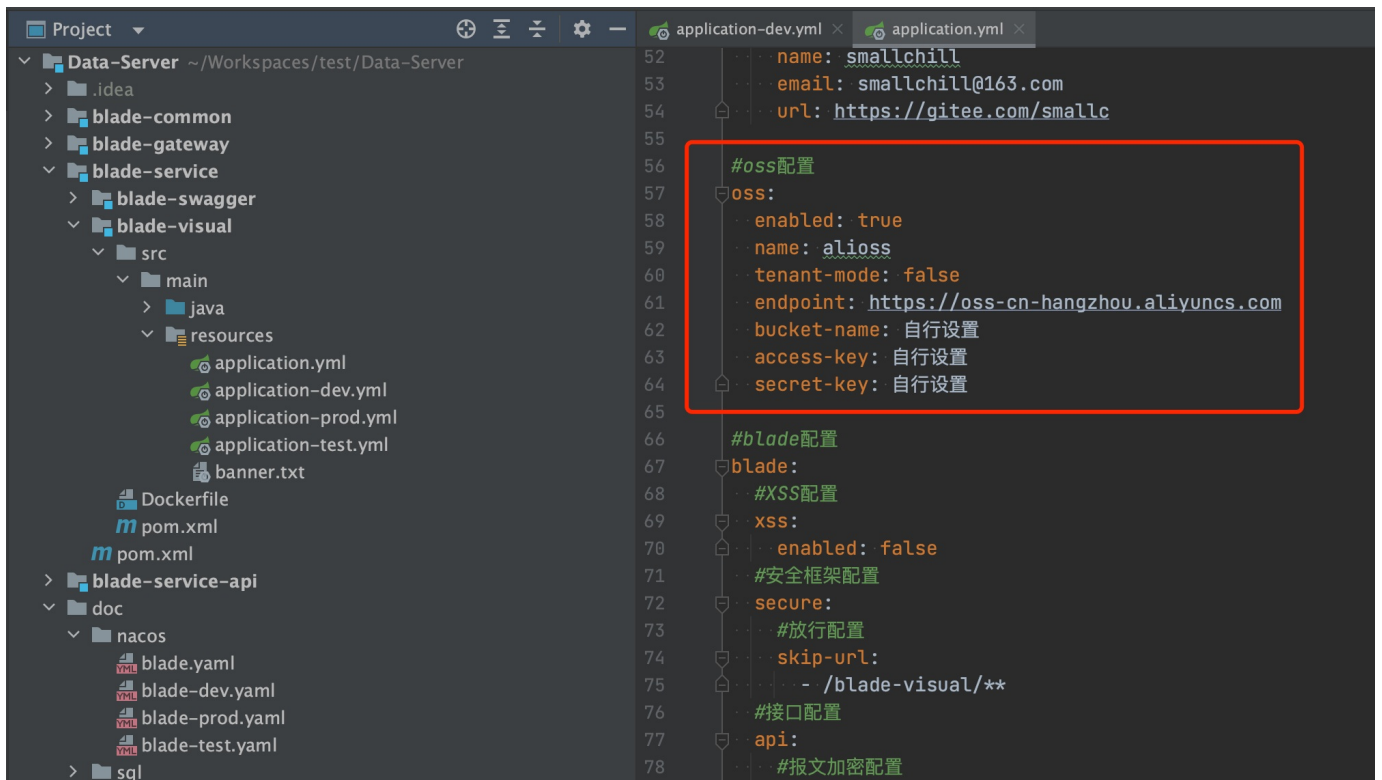


2. 到对应配置文件将数据库信息配置准确



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

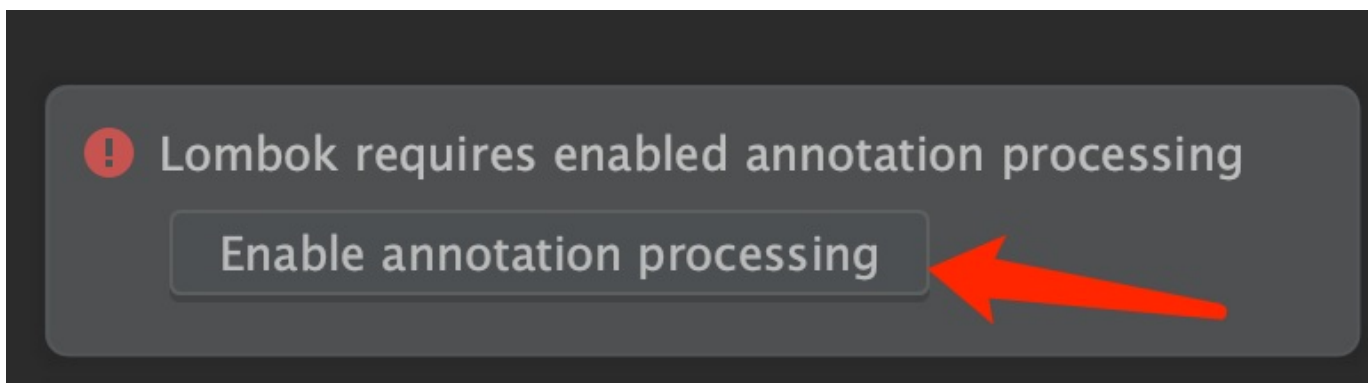
MINIO Customer Login

Product Solutions Docs Blog Resources Partner Pricing Download

License	MINIO AGPL v3	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



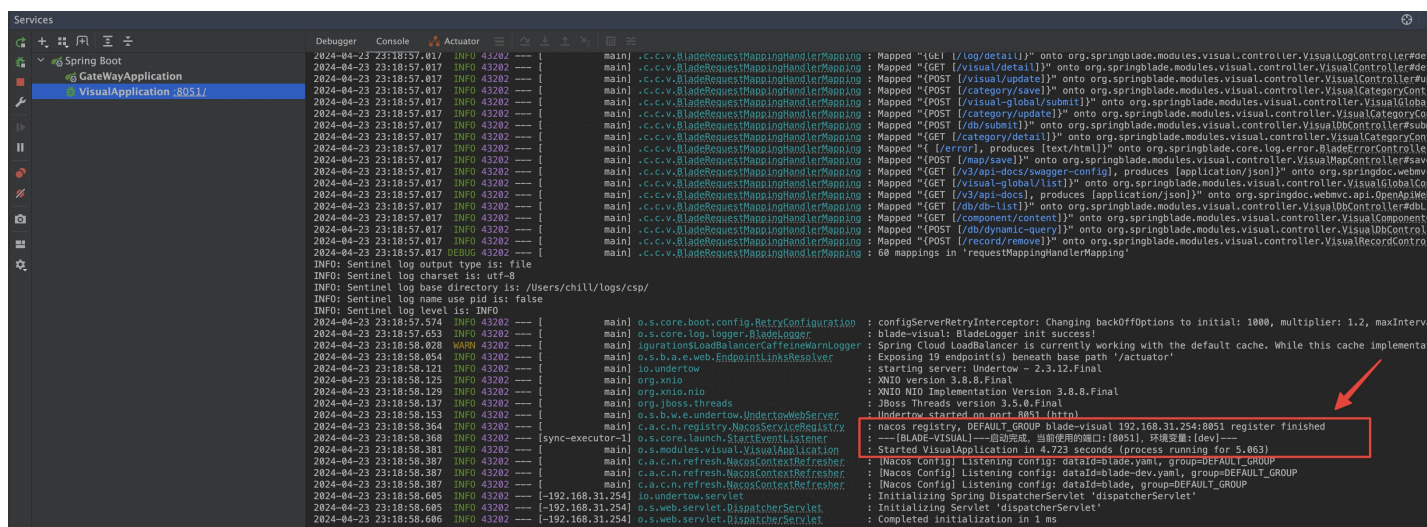
2. 先将nacos配置改为本地配置测试，若配置无误，我们就可以看到如下日志

#数据源配置

```

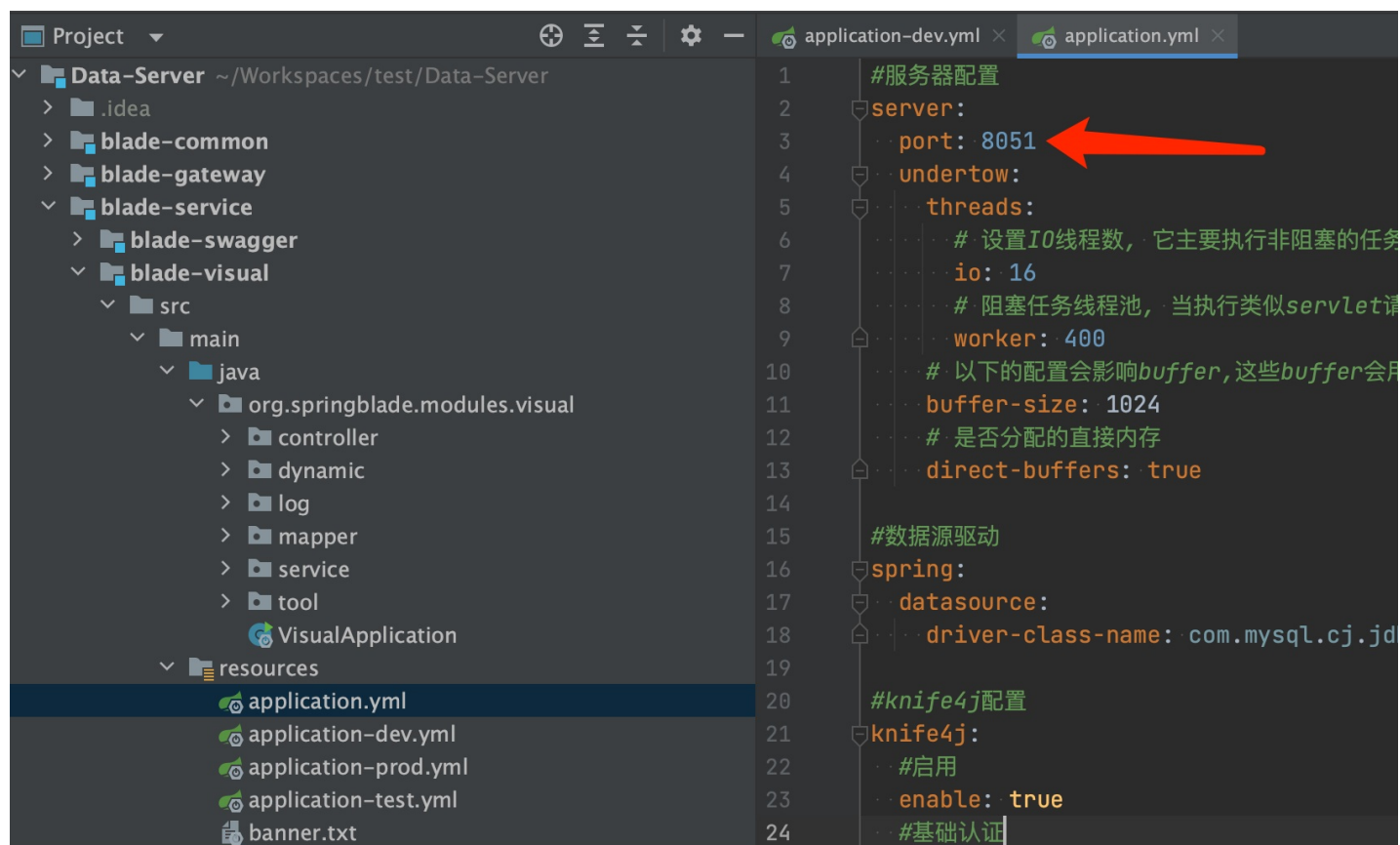
spring:
  data:
    redis:
      ##redis 单机环境配置
      ##将docker脚本部署的redis服务映射为宿主机ip
      ##生产环境推荐使用阿里云高可用redis服务并设置密码
      host: 127.0.0.1
      port: 6379
      password:
      database: 0
      ssl:
        enabled: false
      ##redis 集群环境配置
      #cluster:
      #  nodes: 127.0.0.1:7001,127.0.0.1:7002,127.0.0.1:7003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

→ ↻ 🏠 ⓘ localhost:8051/category/list

```
// 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"
}
```

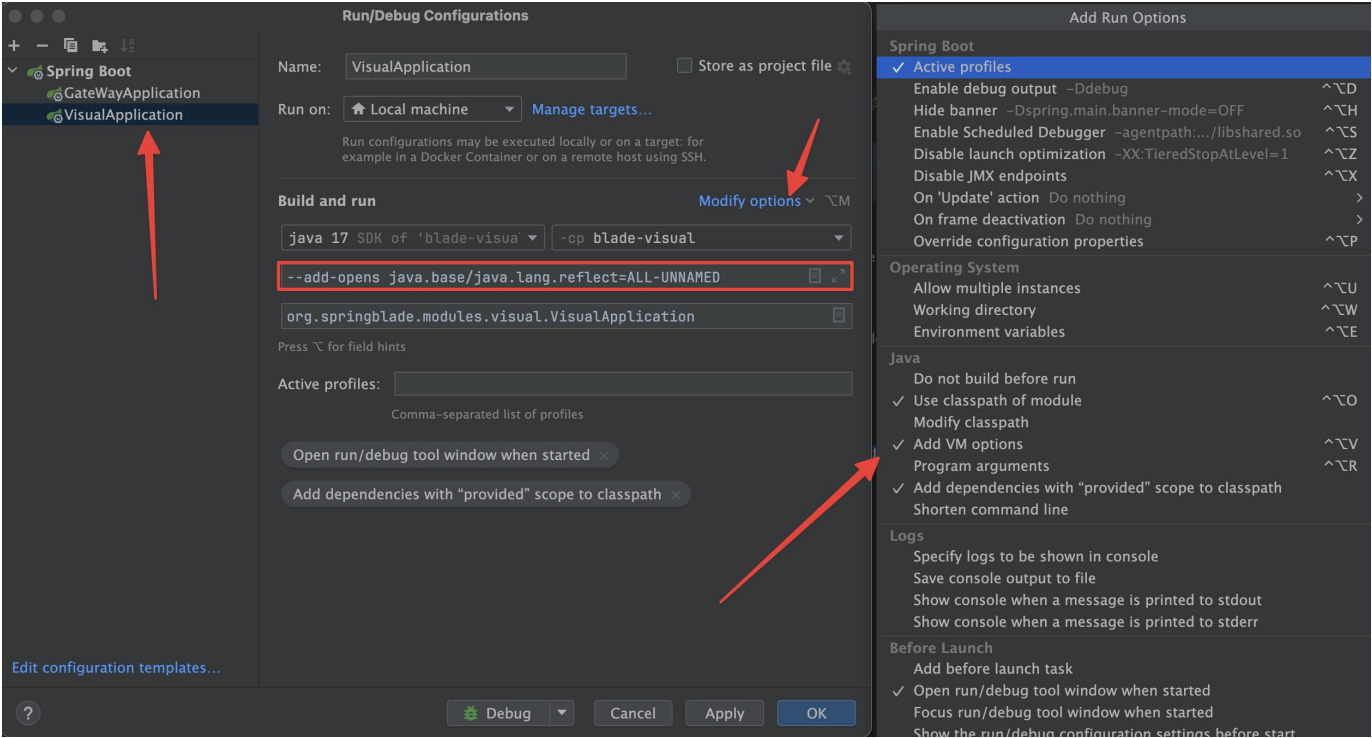
5. 若返回异常，是因为升级JDK17后，Java 模块化系统（Java Module System）的安全限制导致的针对反射等场景有可能会出现问题：

```

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

```
INFO: Sentinel log base directory is: /Users/chill/logs/csp/
INFO: Sentinel log name use pid is: false
2023-09-13 21:23:19.835 WARN 44719 --- [main] io.undertow.websockets.jsr : UT026010: Buffer pool was not set on WebSocketDeploymentInfo, the default pool is deprecated
2023-09-13 21:23:19.870 INFO 44719 --- [main] io.undertow.servlet : Initializing Spring embedded WebApplicationContext
2023-09-13 21:23:19.870 INFO 44719 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3328 ms
2023-09-13 21:23:20.371 INFO 44719 --- [main] c.b.d.d.p.AbstractJdbcDataSourceProvider : 成功加载数据库驱动程序
2023-09-13 21:23:20.565 INFO 44719 --- [main] c.b.d.d.p.AbstractJdbcDataSourceProvider : 成功获取数据库连接
java.sql.SQLException: Create breakpoint : Access denied for user 'bladex_report'@'172.17.0.1' (using password: YES)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:828)
    at com.mysql.cj.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:448)
    at com.mysql.cj.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:241)
    at com.mysql.cj.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:198)
    at java.sql.DriverManager.getConnection(DriverManager.java:664)
    at java.sql.DriverManager.getConnection(DriverManager.java:247)
    at org.springframework.modules.visual.dynamic.DynamicDataSourceHelper.dbTest(DynamicDataSourceHelper.java:38)
    at org.springframework.modules.visual.dynamic.DynamicDataSourceJdbcProvider.executeStmt(DynamicDataSourceJdbcProvider.java:84)
    at com.baomidou.dynamic.datasource.provider.AbstractJdbcDataSourceProvider.loadDataSources(AbstractJdbcDataSourceProvider.java:83)
    at com.baomidou.dynamic.datasource.DynamicRoutingDataSource.afterPropertiesSet(DynamicRoutingDataSource.java:227)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1863)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1800)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:620)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:542)
```

对象 blade_visual_db@blade [...]

id	name	driver_class	url	username	password	remark	create_user	create_dept	create_time	update_user	update_time	status	is_deleted
1161483357461541634	mysql	com.mysql.cj.jdbc.Driver	jdbc:mysql://localhost:3306/blade_report?h	bladex_report	YAmAbmkeNtLRjWSW	mysql	1123598821738675201	1123598813738675201	2020-08-08 12:20:06	1123598821738675201	2021-10-22 10:24:59	1	0
116148350435484802	postgresql	org.postgresql.Driver	jdbc:postgresql://127.0.0.1:5432/blade	postgres	123456	postgresql	1123598821738675201	1123598813738675201	2020-08-08 12:20:02	1123598821738675201	2021-03-10 22:00:00	1	0
1161483594023510018	oracle	oracle.jdbc.OracleDriver	jdbc:oracle:thin:@127.0.0.1:1521/orcl	BLADEX	BLADEX	oracle	1123598821738675201	1123598813738675201	2020-08-08 12:20:03	1123598821738675201	2020-08-08 12:20:03	1	0

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

Project

~/Workspaces/test/Data-Server

blade.yaml

blade-dev.yaml

blade-prod.yaml

blade-test.yaml

sql

.editorconfig

.gitignore

LICENSE

pom.xml

README.md

External Libraries

< 1.8 > /Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/C

Maven: ch.qos.logback:logback-classic:1.2.11

Maven: ch.qos.logback:logback-core:1.2.11

Maven: cn.hutool:hutool-all:5.4.1

Maven: com.alibaba.cloud:spring-cloud-alibaba-commons:202

Maven: com.alibaba.cloud:spring-cloud-alibaba-sentinel-data

Maven: com.alibaba.cloud:spring-cloud-circuitbreaker-sentine

Maven: com.alibaba.cloud:spring-cloud-starter-alibaba-nacos

Maven: com.alibaba.cloud:spring-cloud-starter-alibaba-nacos

blade.yaml

1 #服务器配置

2 server:

3 undertow:

4 # 以下的配置会影响buffer, 这些buffer会用于服务

5 buffer-size: 1024

6 # 是否分配的直接内存

7 direct-buffers: true

8 # 线程配置

9 threads:

10 # 设置IO线程数, 它主要执行非阻塞的任务, 它们

11 io: 16

12 # 阻塞任务线程池, 当执行类似servlet请求阻塞

13 worker: 400

14

15 #spring配置

16 spring:

17 cloud:

18 sentinel:

19 eager: true

20 devtools:

21 restart:

22 log-condition-evaluation-delta: false

23 livereload:

24 port: 23333

25

26 #feign配置

27 feign:

28 sentinel:

NACOS

首页 文档 博客 社区 Nacos企业版 En nacos

NACOS 2.3.1

模式 standalone

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

设置中心

配置管理

public

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

高级查询

导入配置

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-demo-dev.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 >

<

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 ⓘ :

```
79 url: https://gitee.com/smallc
80
81 #blade配置
82 blade:
83   xss:
84     enabled: true
85     skip-url:
86       - /weixin
87   secure:
88     skip-url:
89       - /test/**
90   client:
91     - client-id: sword
92       path-patterns:
93         - /sword/**
94     - client-id: saber
95       path-patterns:
96         - /saber/**
97   tenant:
98     column: tenant_id
99     tables:
100       - blade_notice
101
102
```

发布

返回

NACOS.

首页 文档 博客 社区 Nacos企业版 En nacos

<

新建配置

* Data ID blade-dev.yaml

* Group DEFAULT_GROUP

更多高级选项

描述

配置格式 ☐ TEXT ☐ JSON ☐ XML ☒ YAML ☐ HTML ☐ Properties

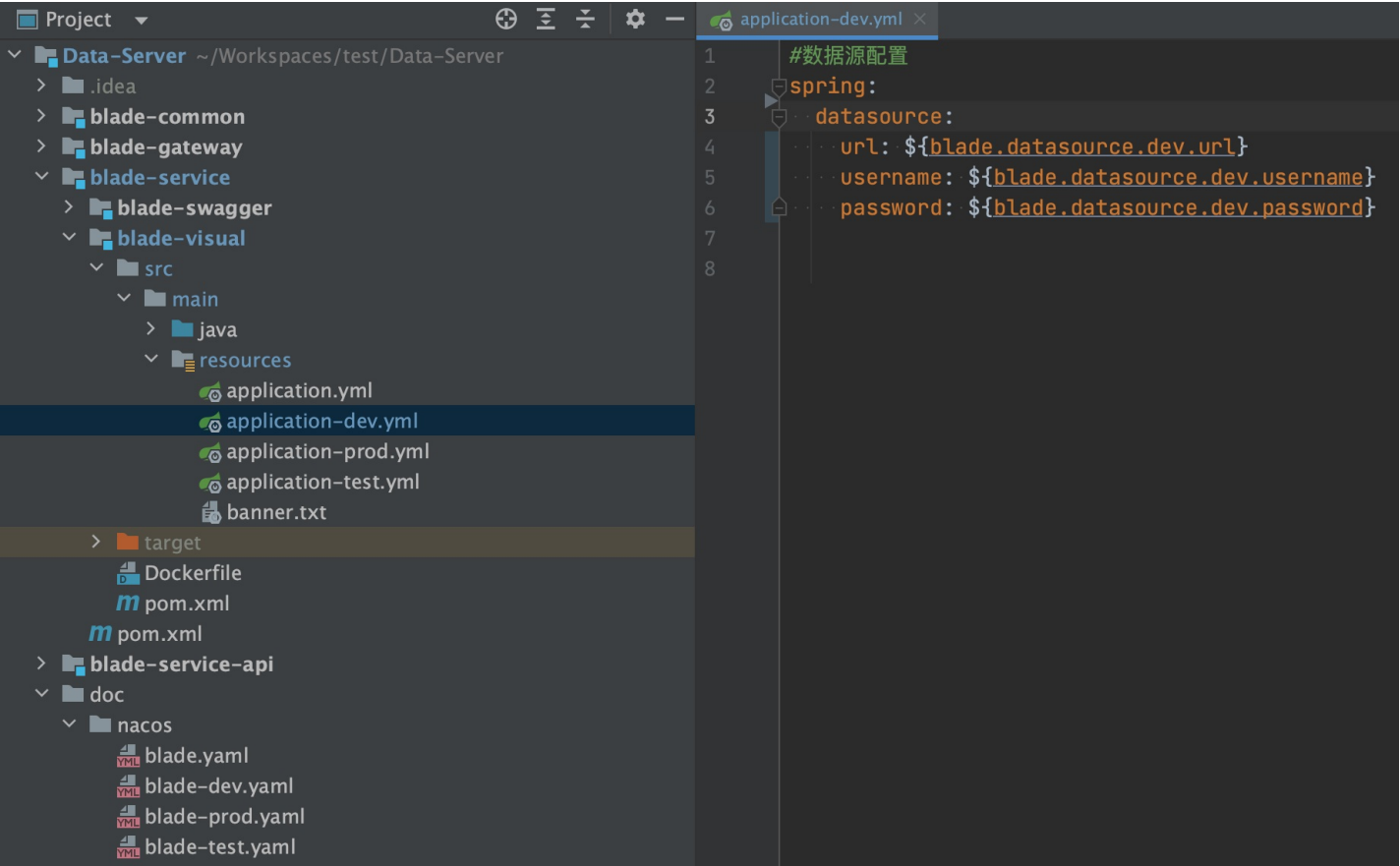
* 配置内容

```
1 #spring配置
2 spring:
3   redis:
4     ##redis 单机环境配置
5     host: 127.0.0.1
6     port: 6379
7     password:
8     database: 0
9     ssl: false
10    ##redis 集群环境配置
11    #cluster:
12    # nodes: 127.0.0.1:7001,127.0.0.1:7002,127.0.0.1:7003
13    # commandTimeout: 5000
14
15  #项目模块集中配置
16 blade:
17   #通用开发生产环境数据库地址(特殊情况可在对应的子工程里配置覆盖)
18   datasource:
19     dev:
20       url: jdbc:mysql://localhost:3306/blade?useSSL=false&useUnicode=true&characterEncod
21       username: root
22       password: root
23
24   ##大屏配置
25   visual:
26     #限流白名单
```

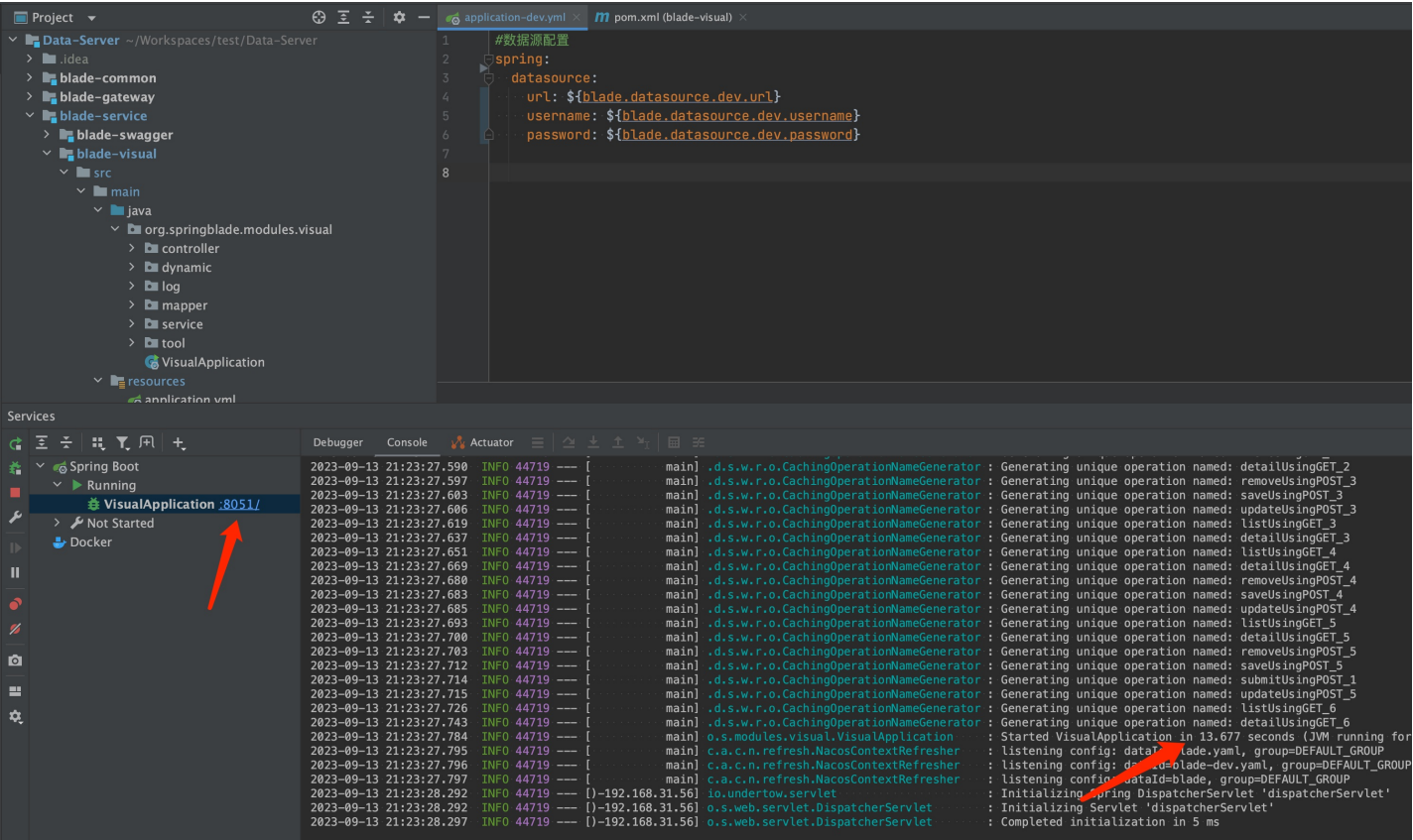
发布 返回

3. 修改blade-visual服务的配置，改为从nacos读取

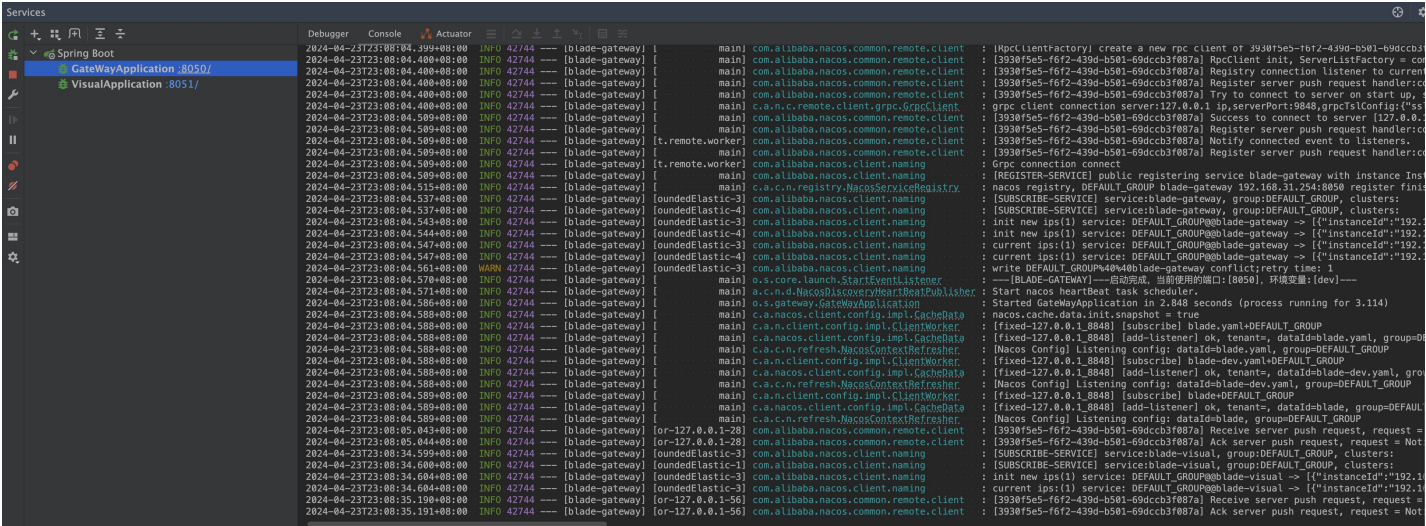
```
spring:
  datasource:
    url: ${blade.datasource.dev.url}
    username: ${blade.datasource.dev.username}
    password: ${blade.datasource.dev.password}
```



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

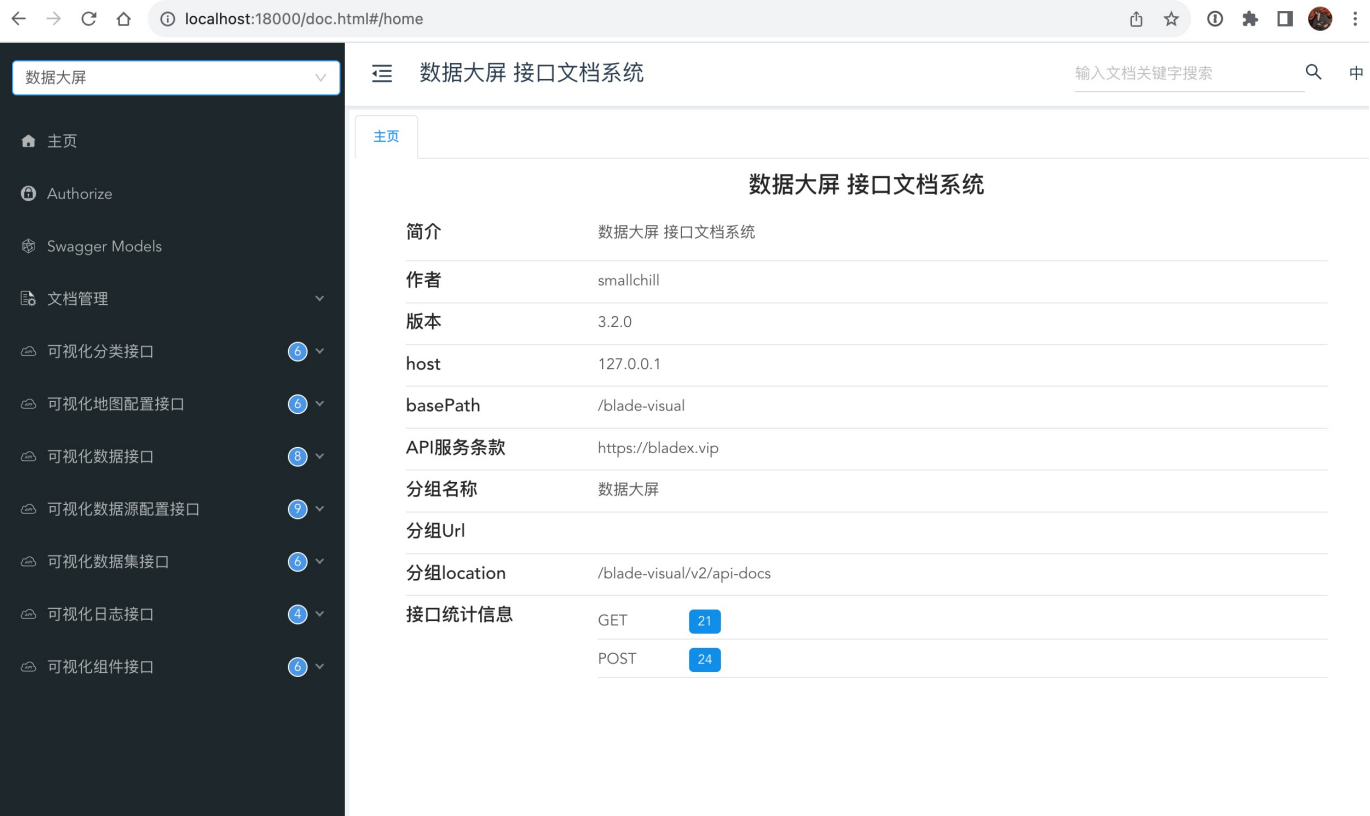
→ ↻ 🏠 ⓘ localhost:8050/blade-visual/category/list

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

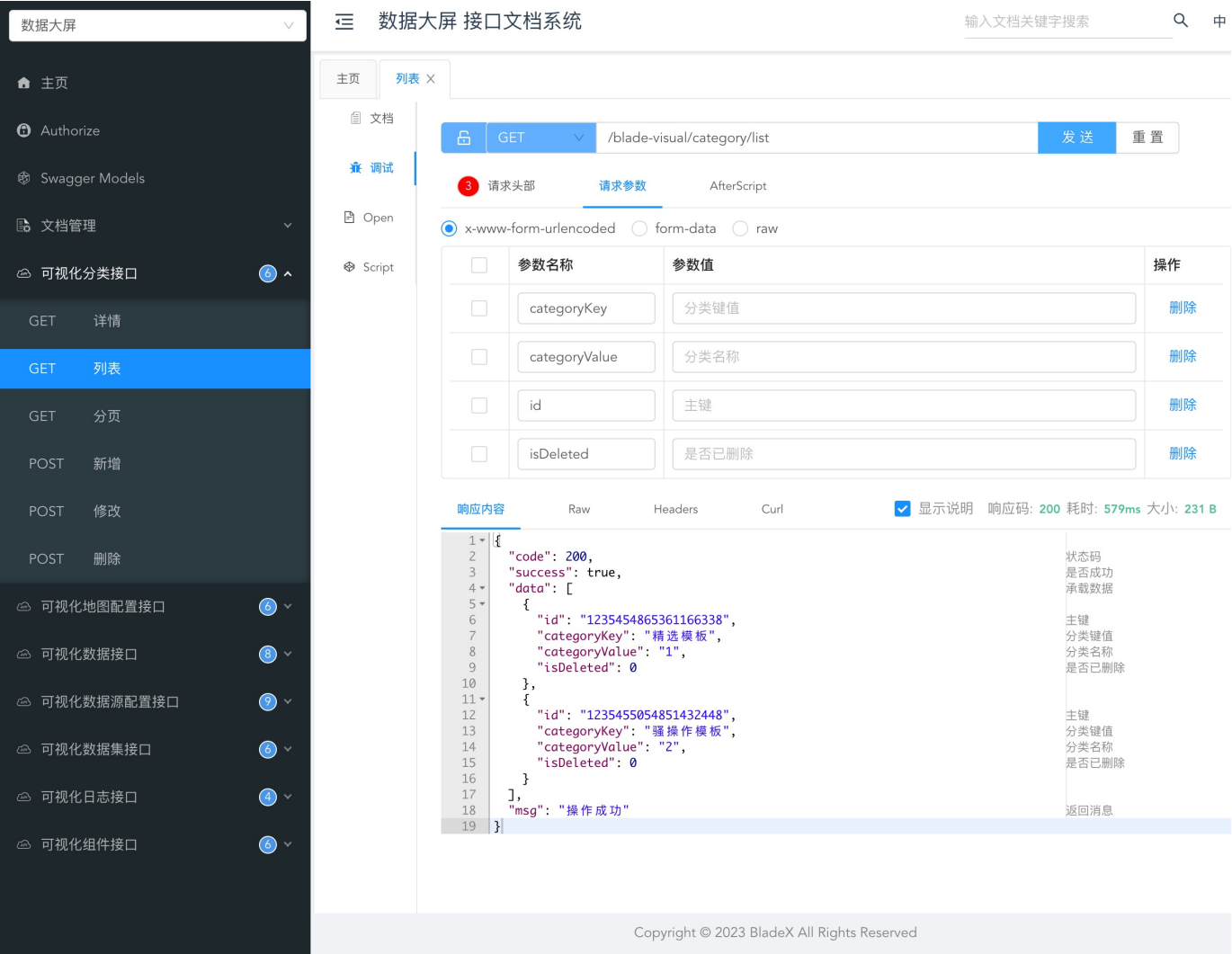
{
  "code": 200,
  "success": true,
  "data": [
    {
      "id": "1235454865361166338",
      "categoryKey": "精选模板",
      "categoryValue": "1",
      "isDeleted": 0
    },
    {
      "id": "1235455054851432448",
      "categoryKey": "骚操作模板",
      "categoryValue": "2",
      "isDeleted": 0
    }
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



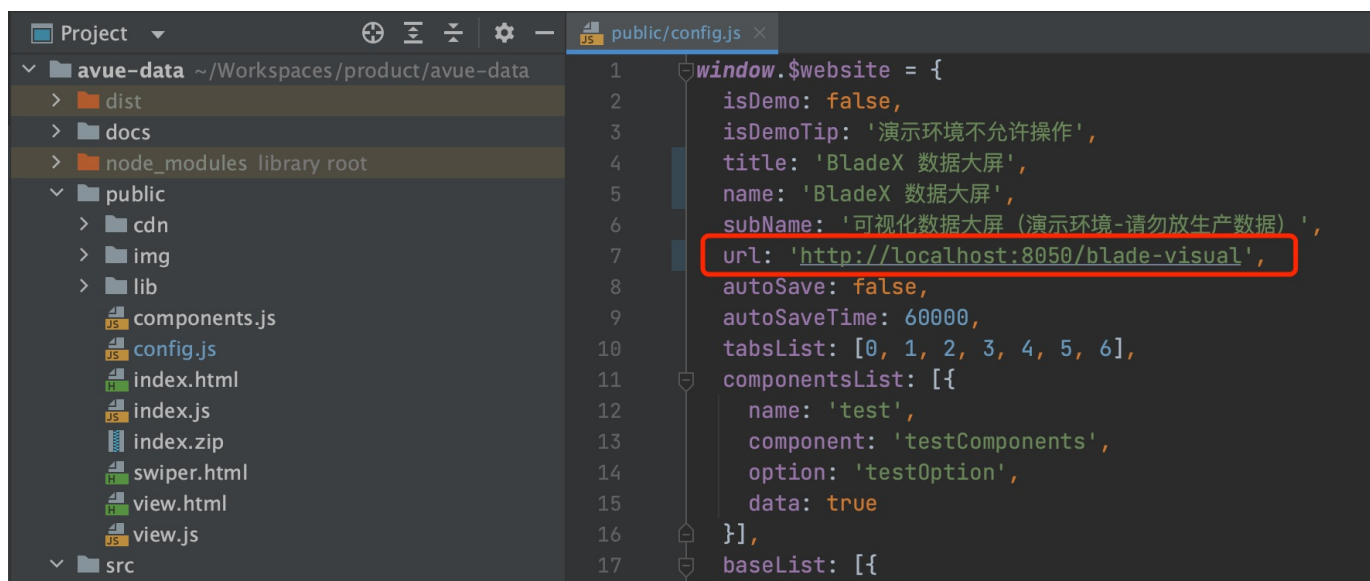
2. 找一个接口点击查看数据正确返回



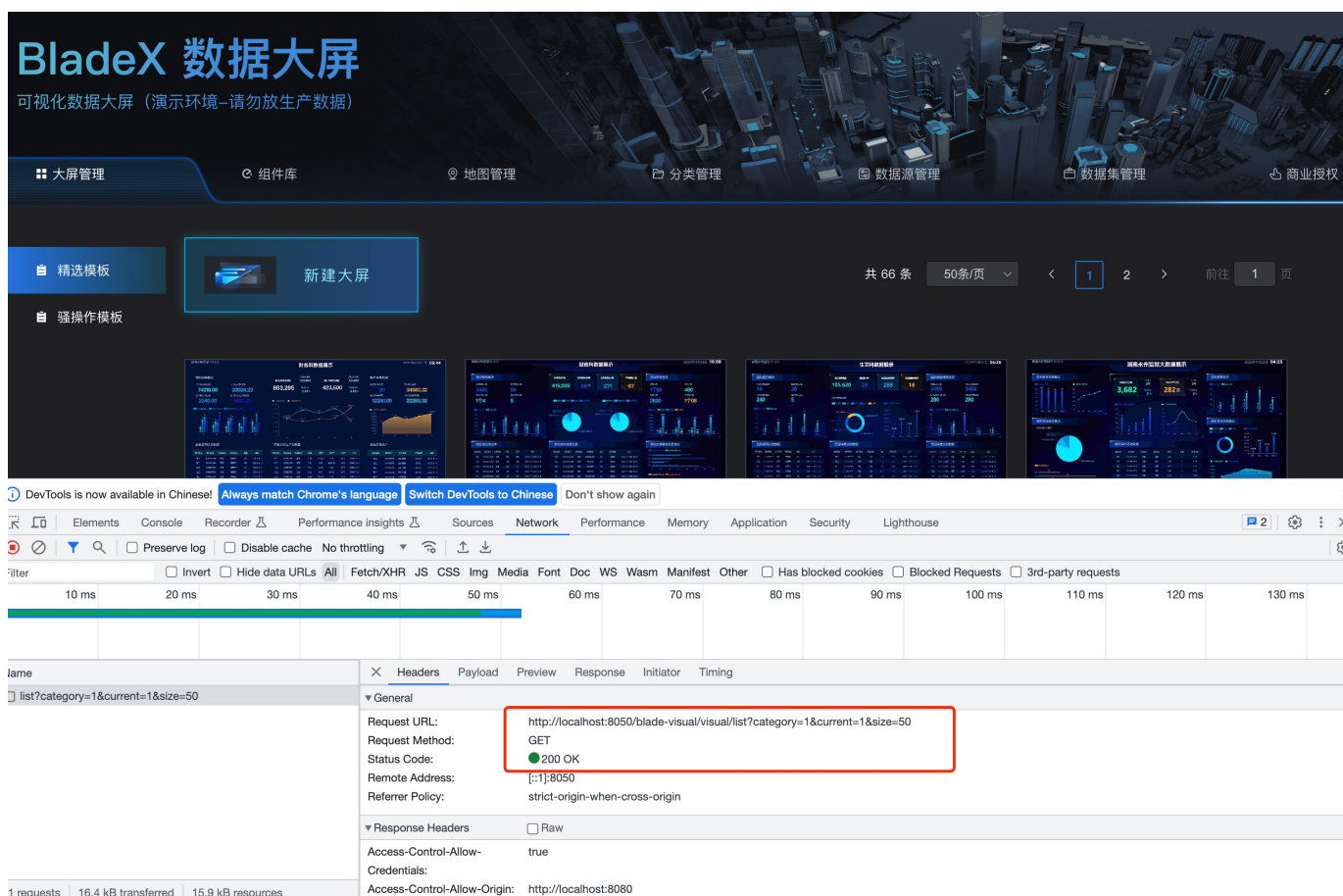
3. 至此后端部署全流程结束

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



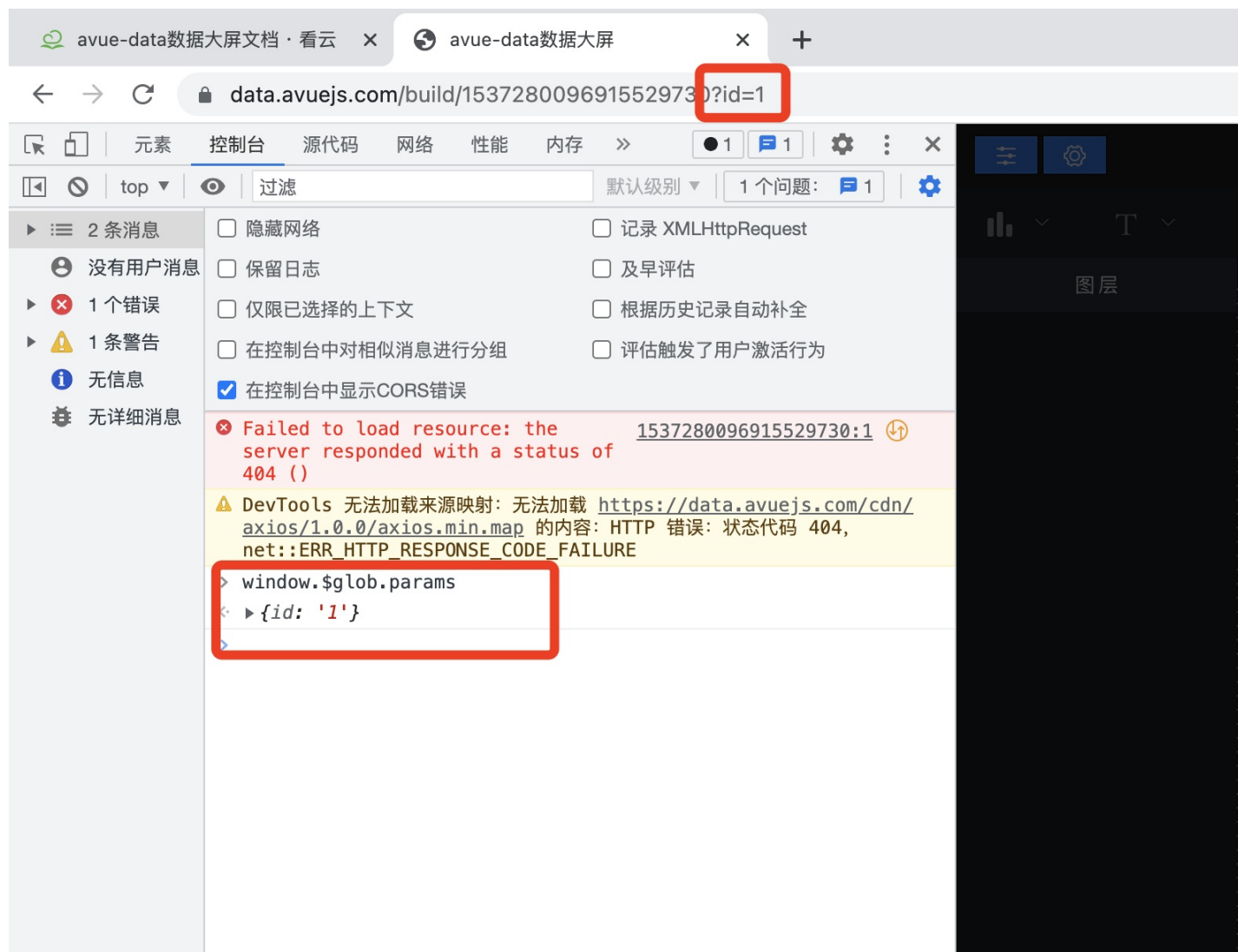
全局变量

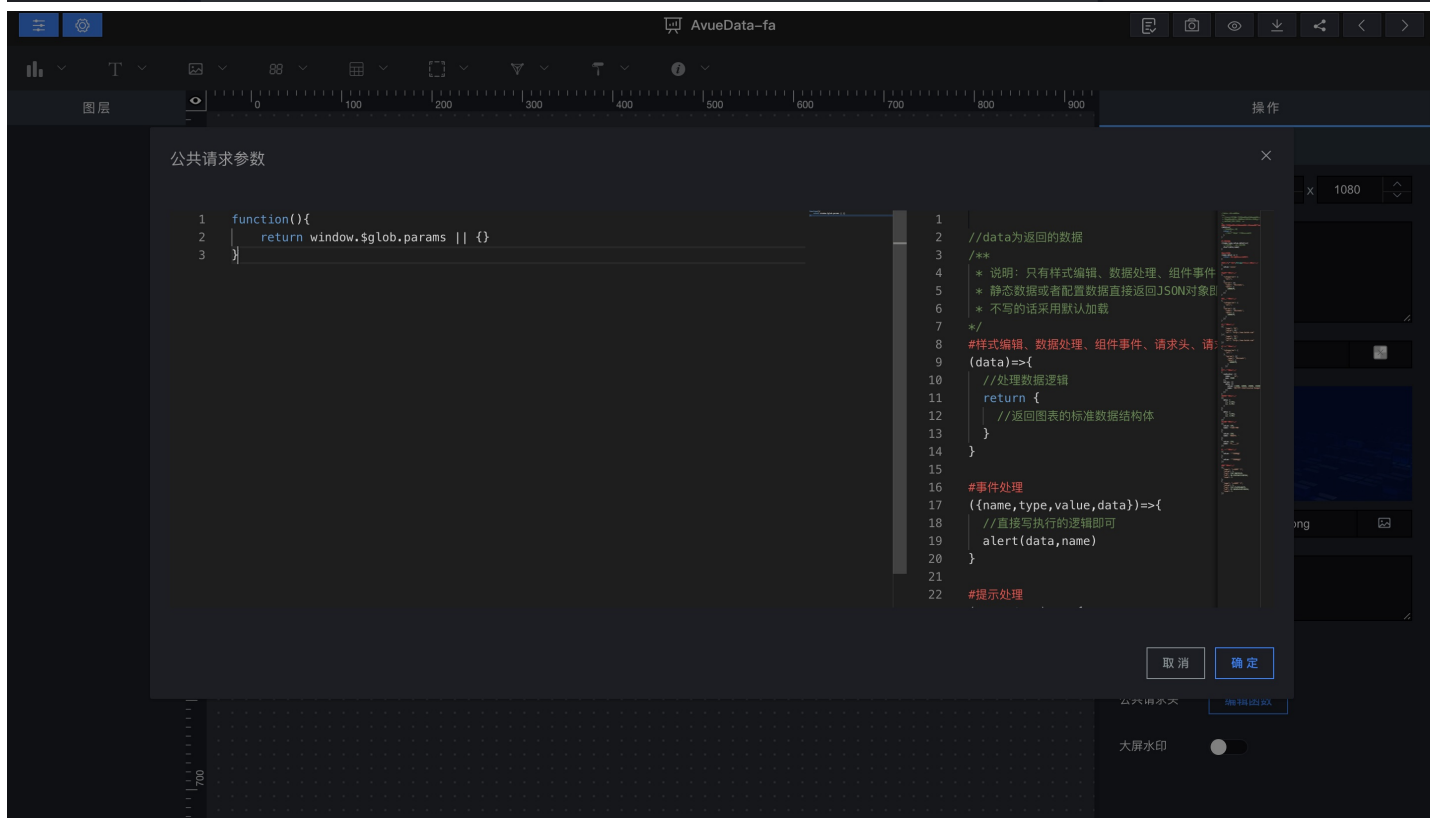
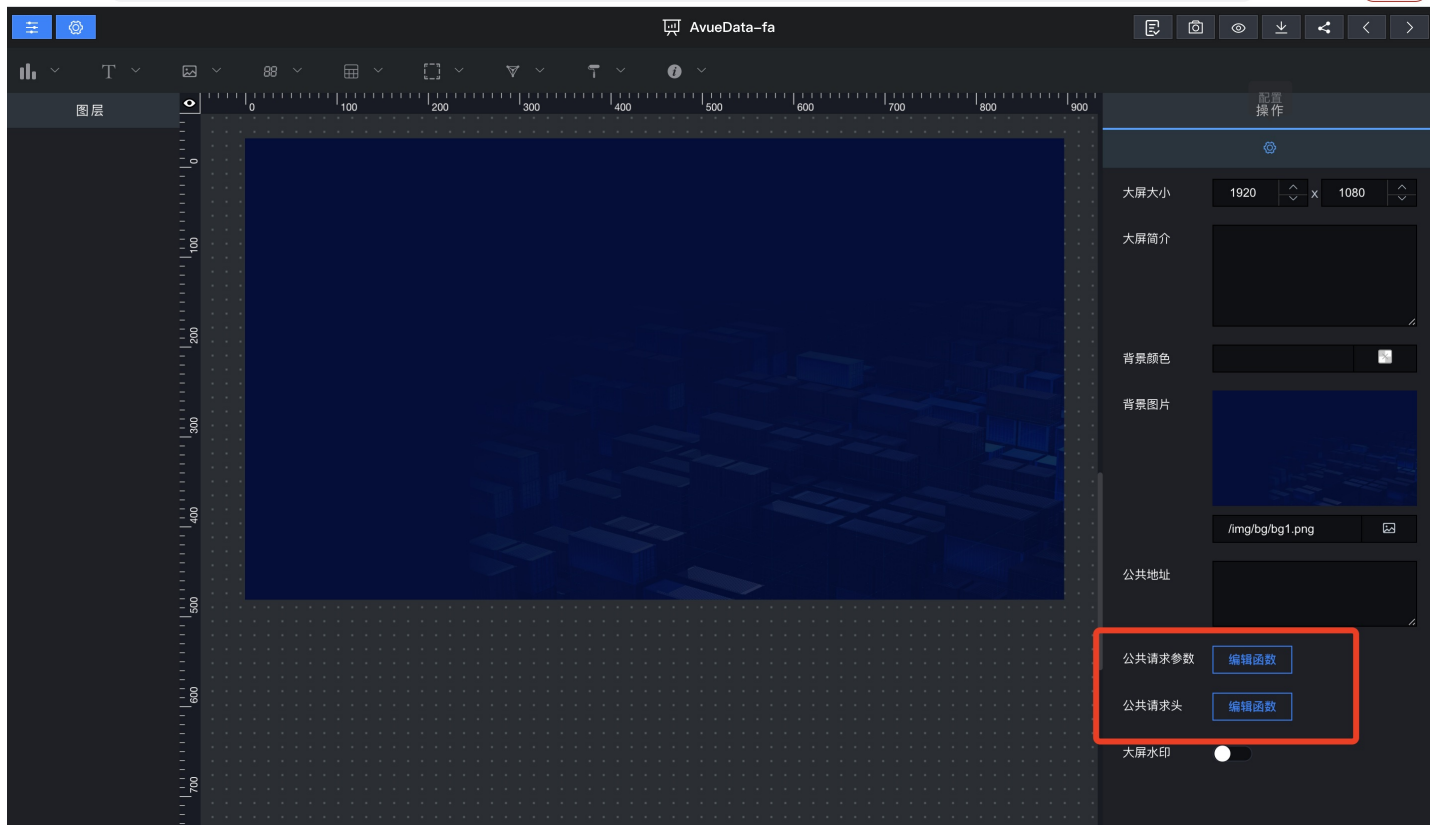
全局变量，它可以作用于axios发送数据中

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



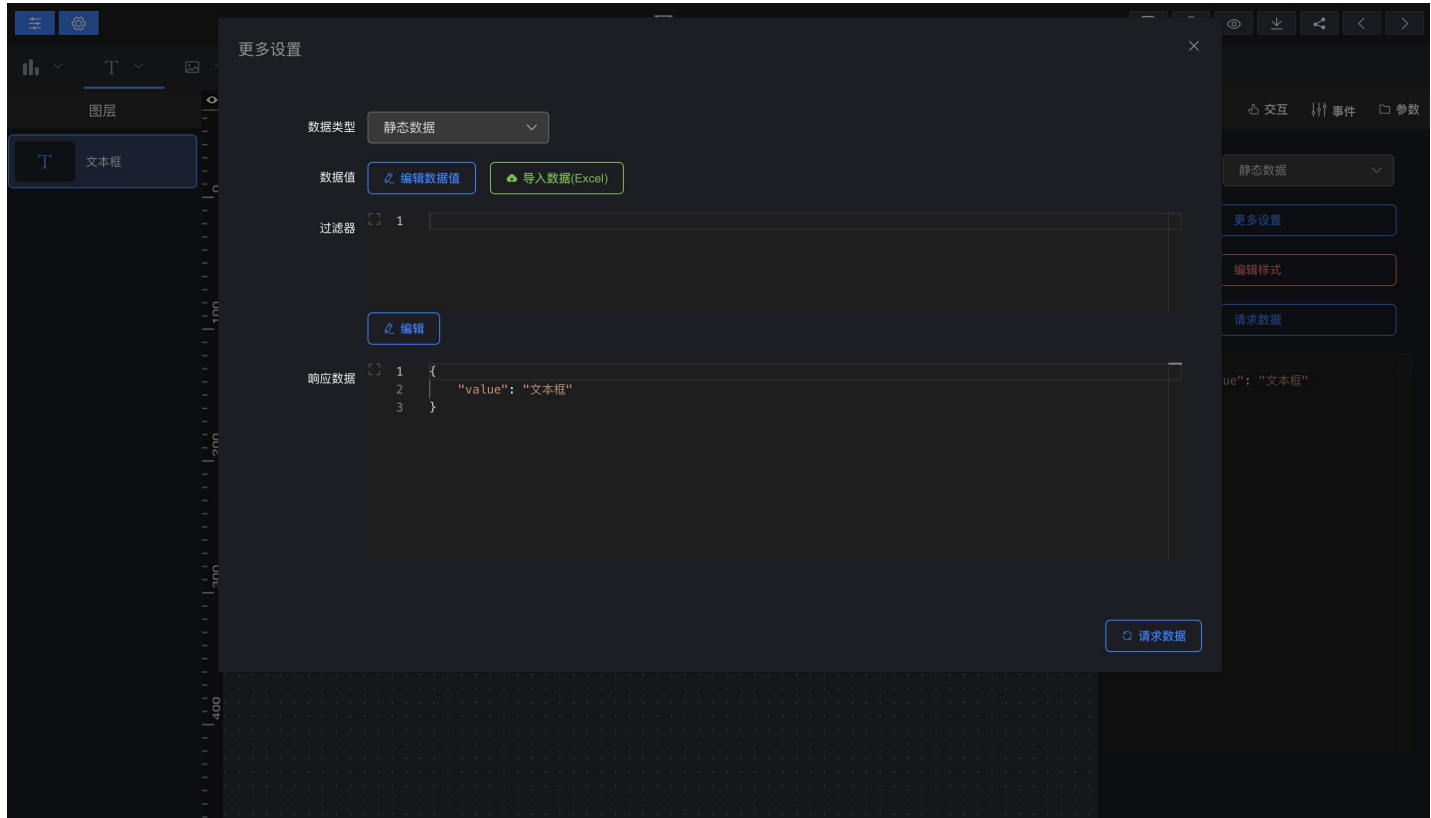


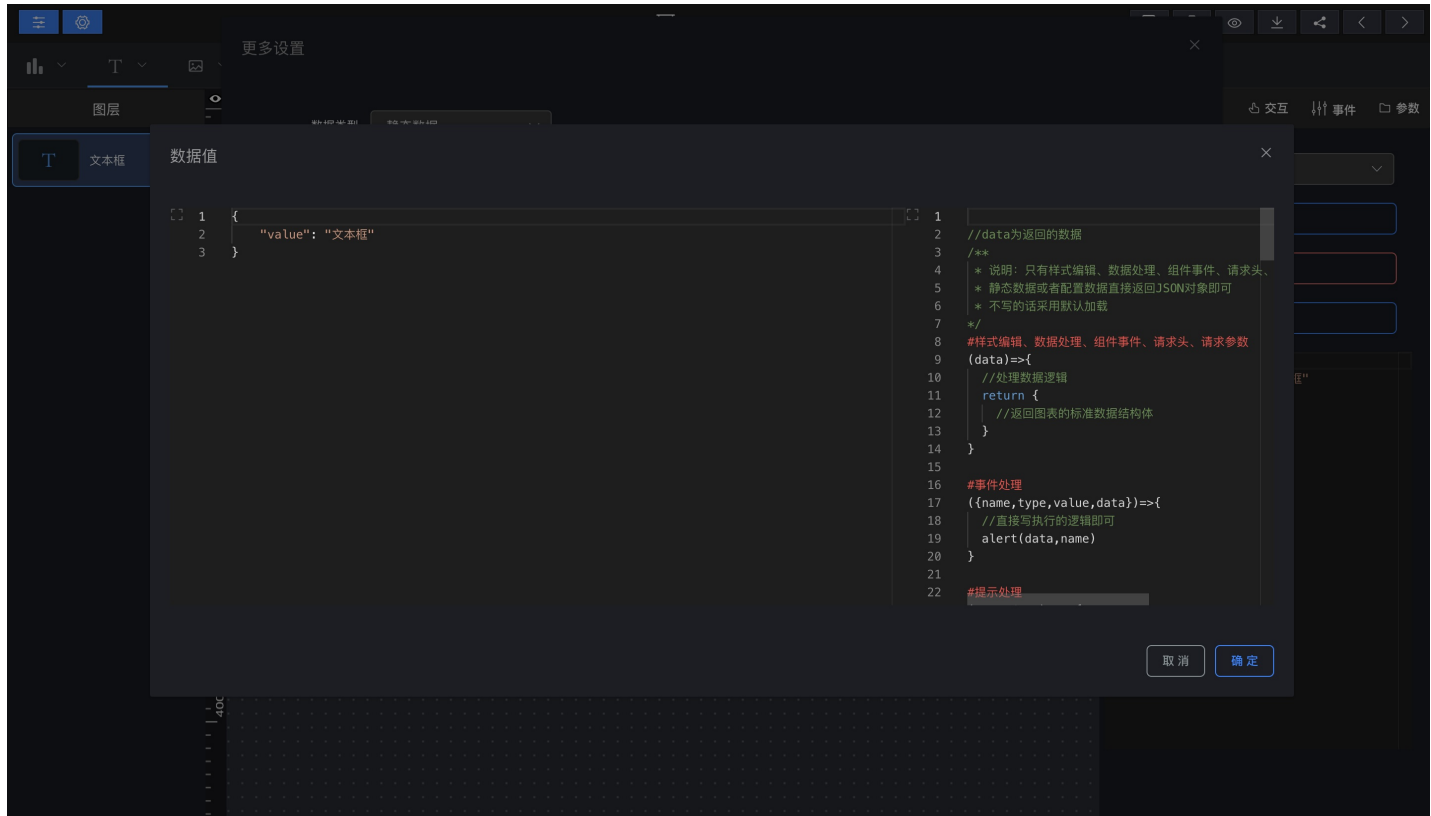
组件数据

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

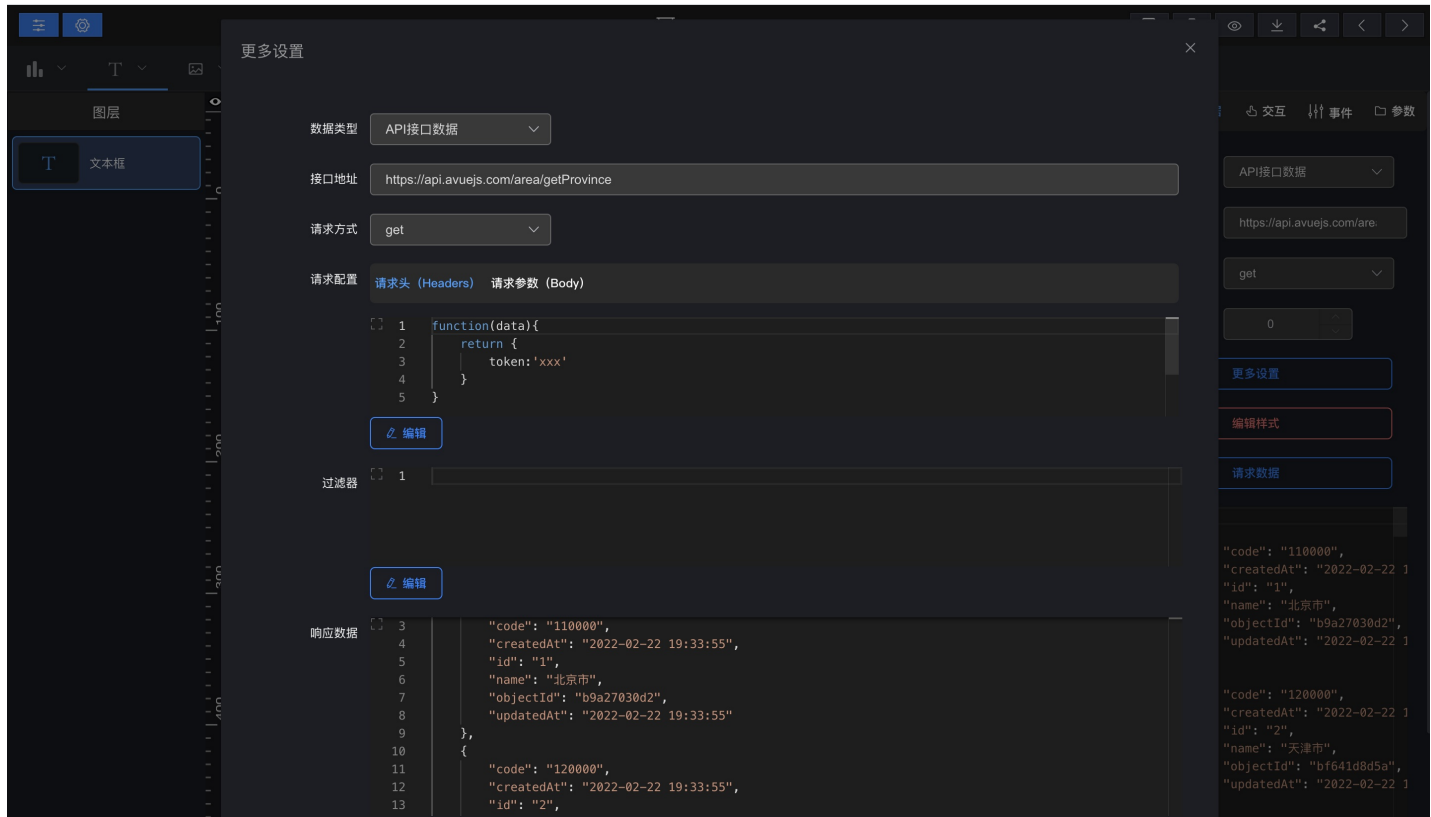
直接配置成组件对应的数据格式





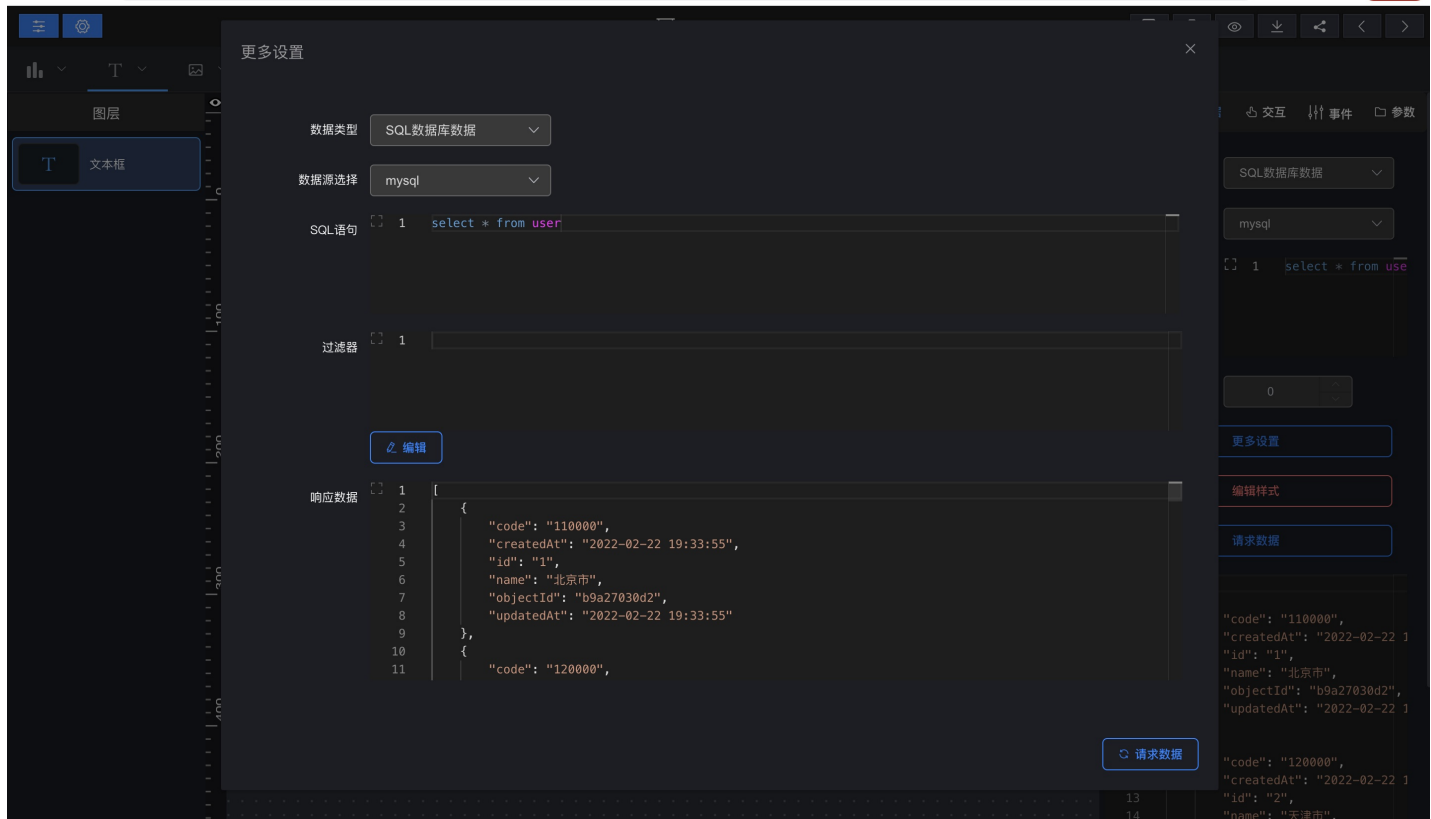
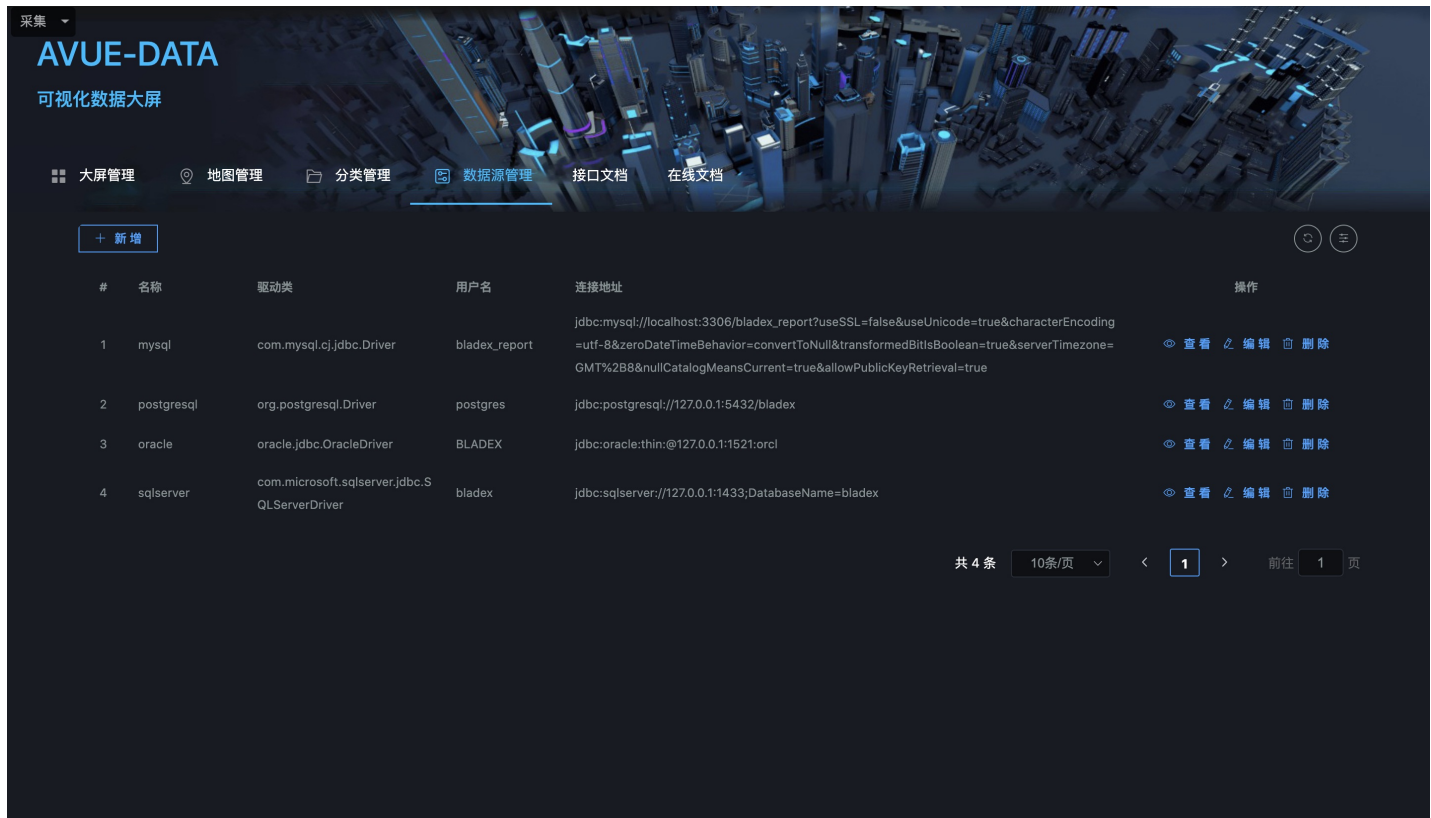
二、API接口

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

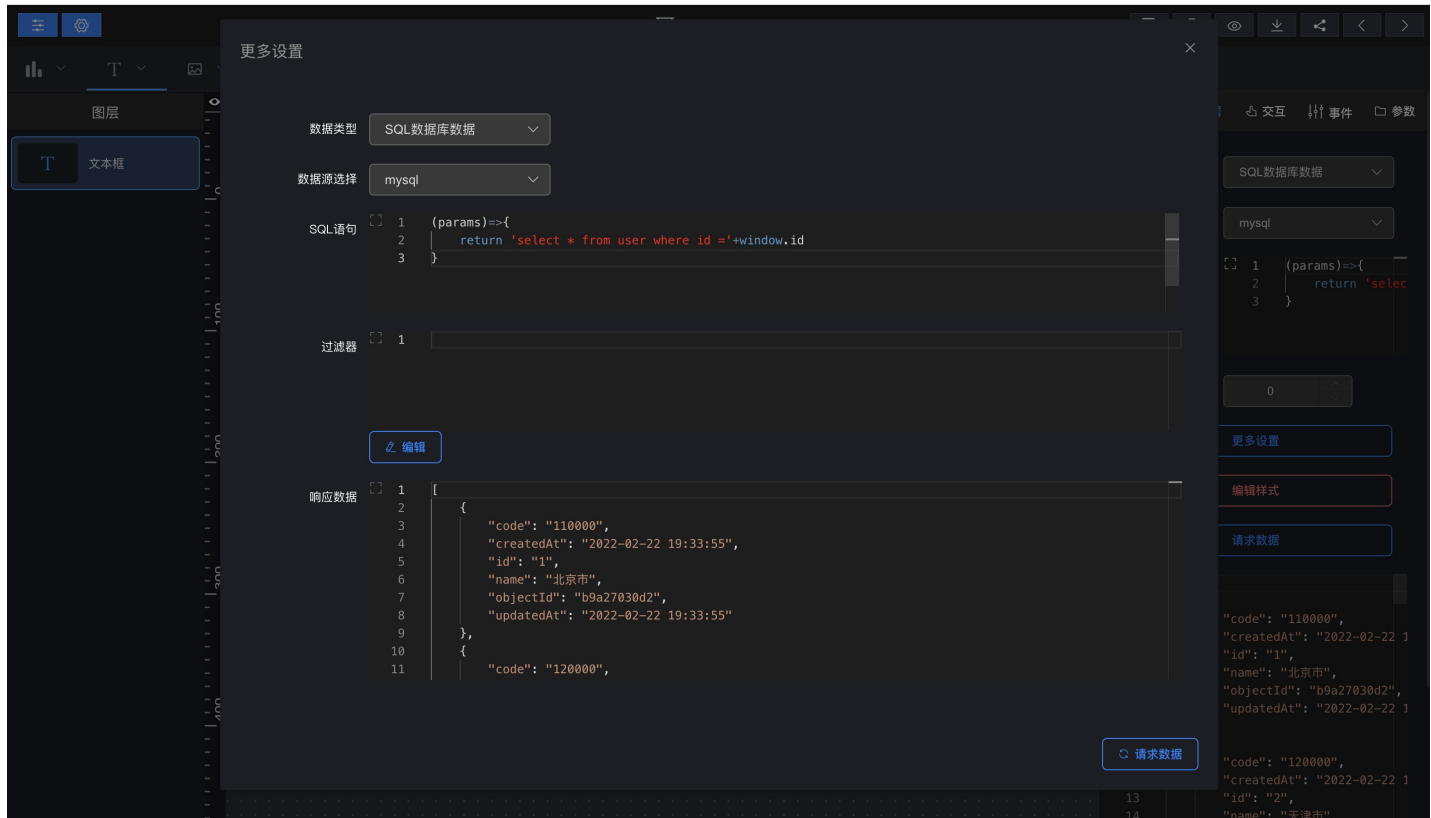


三、SQL数据源

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

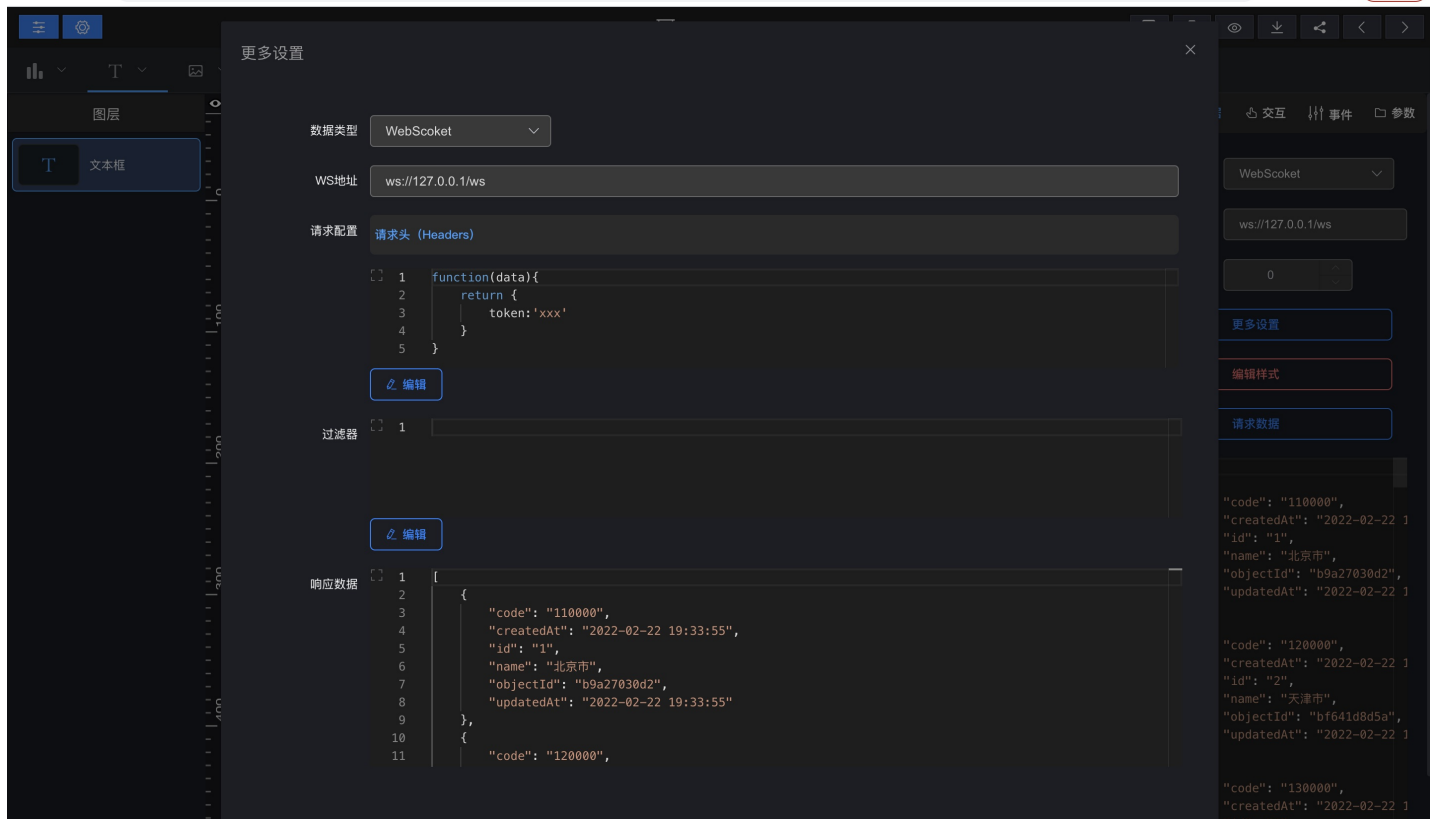


这里的SQL是可以配置成函数形式，这样就可以取各种参数和做一些处理逻辑,返回最终的查询语句



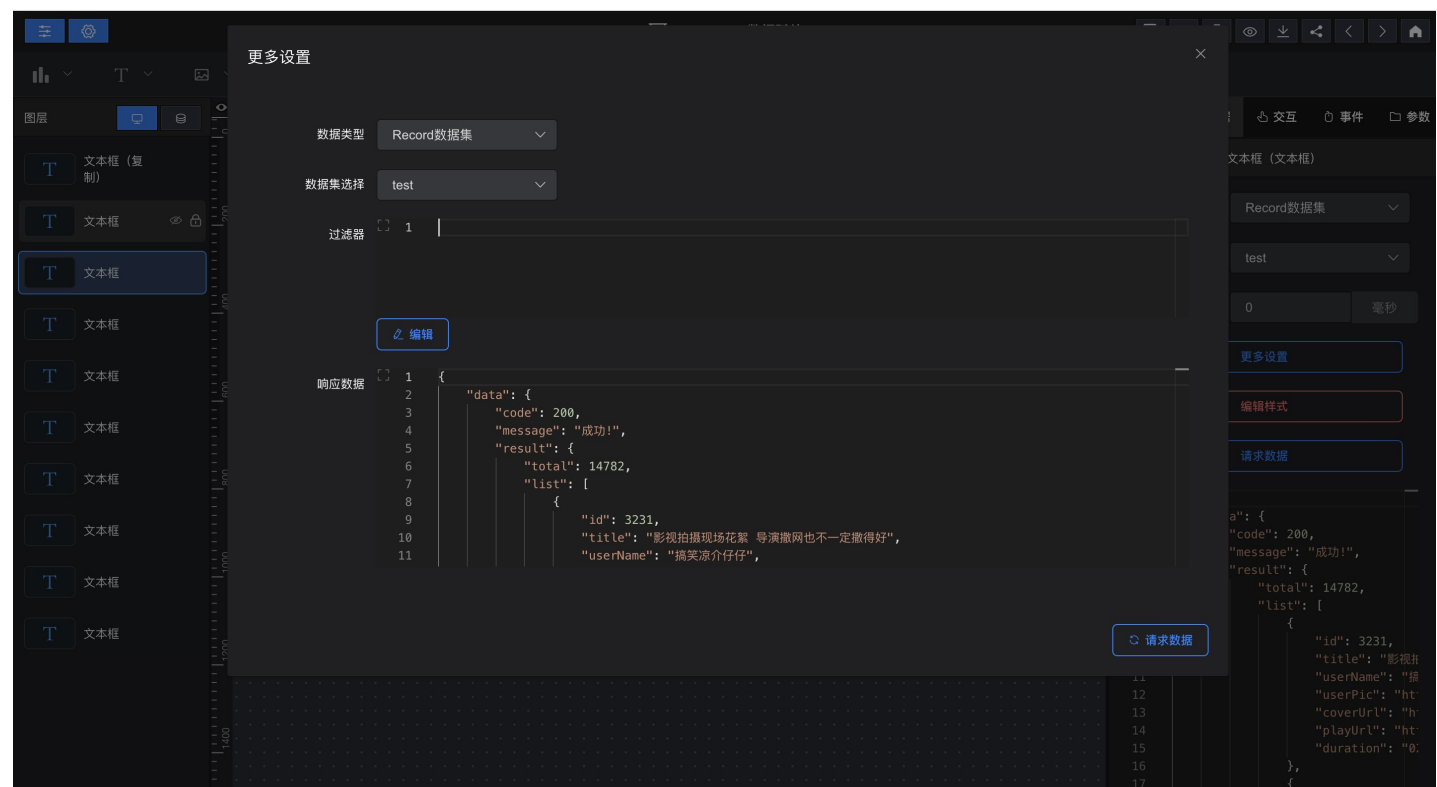
四、Websocket长链接

配置长链接地址返回数据，用法和API类似

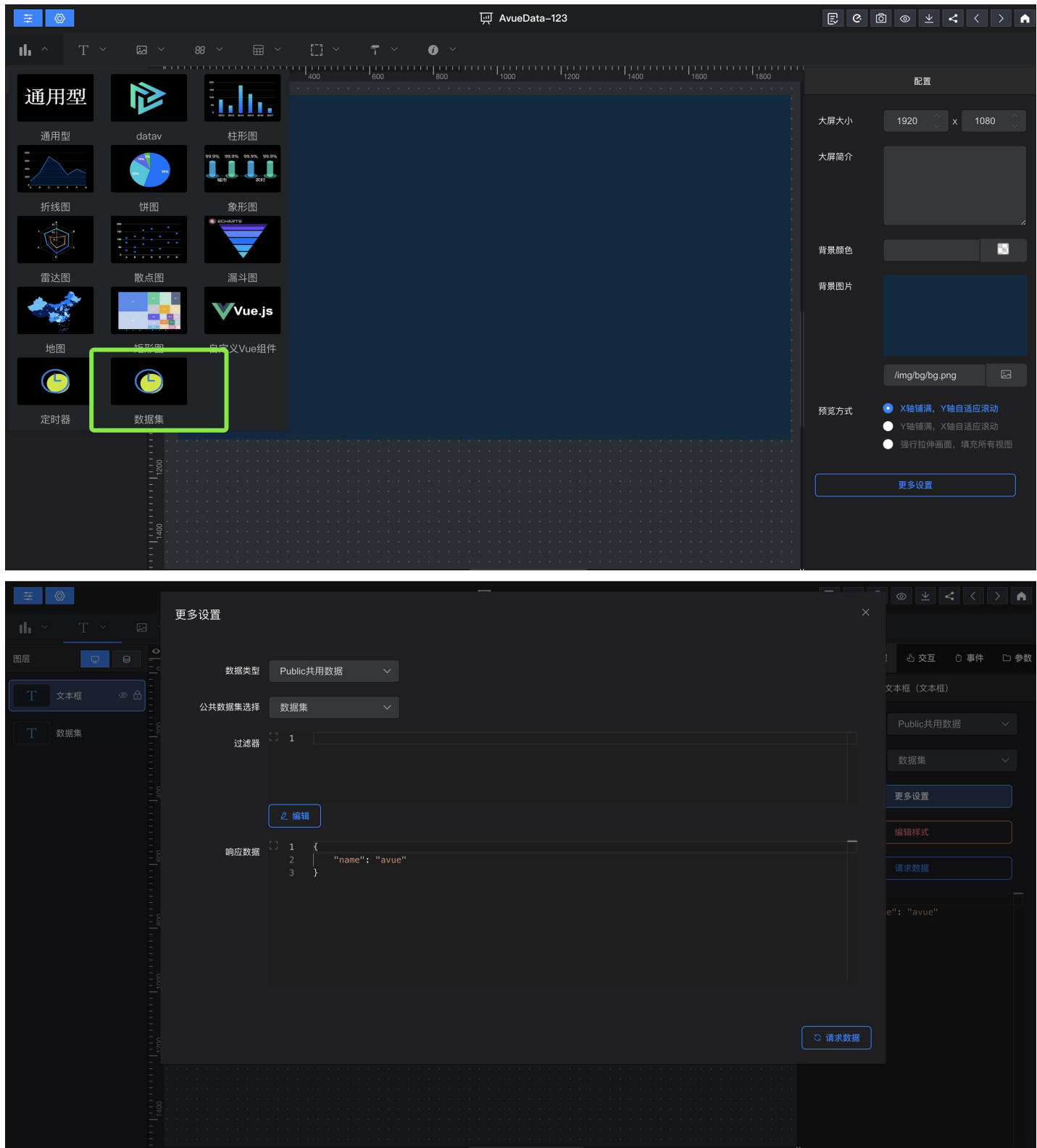


五、Record数据集

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

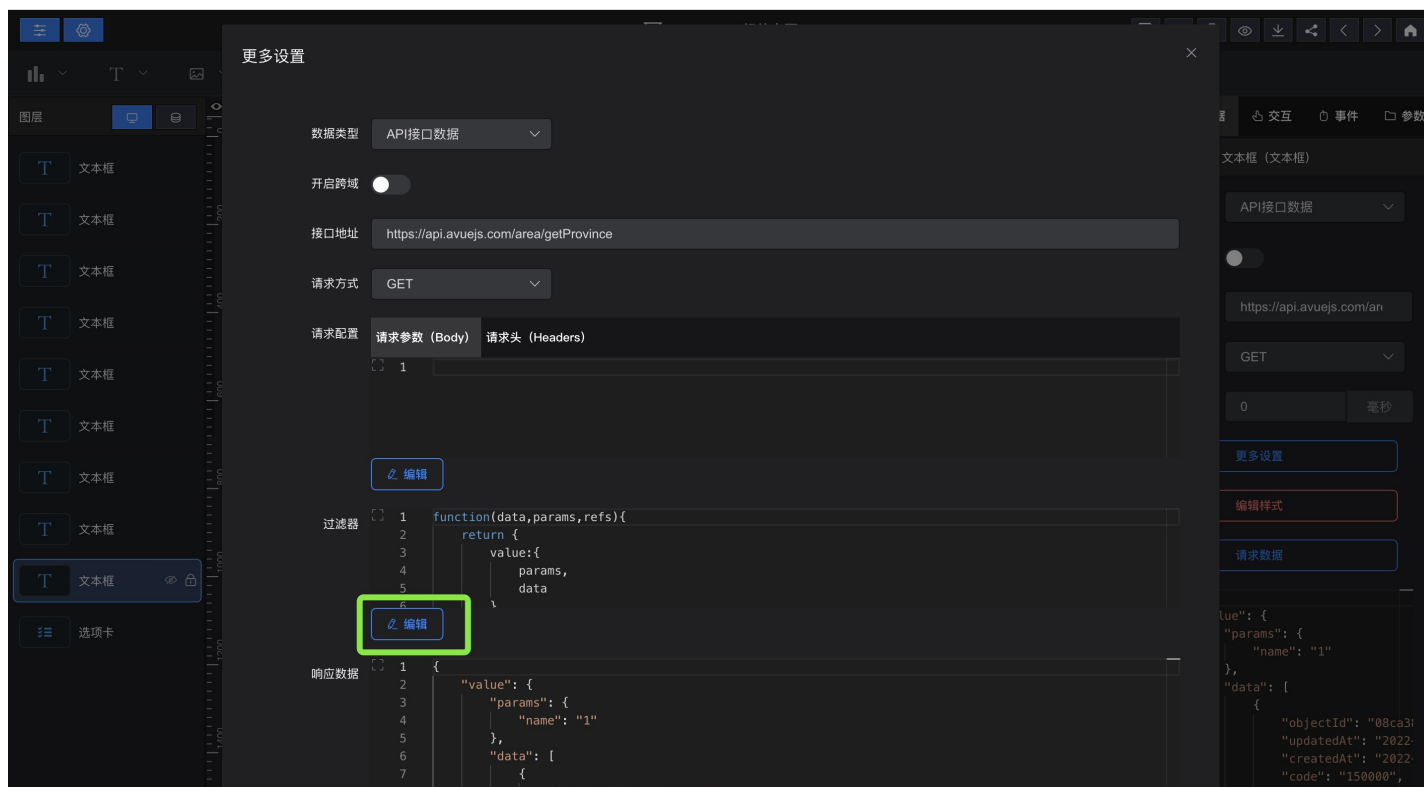


他和Record数据集的用法一样，他是相对于页面的，他需要在页面配置，完后对应的组件去引用，它只会发送一次请求，把数据结果下发到每个组件使用



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。



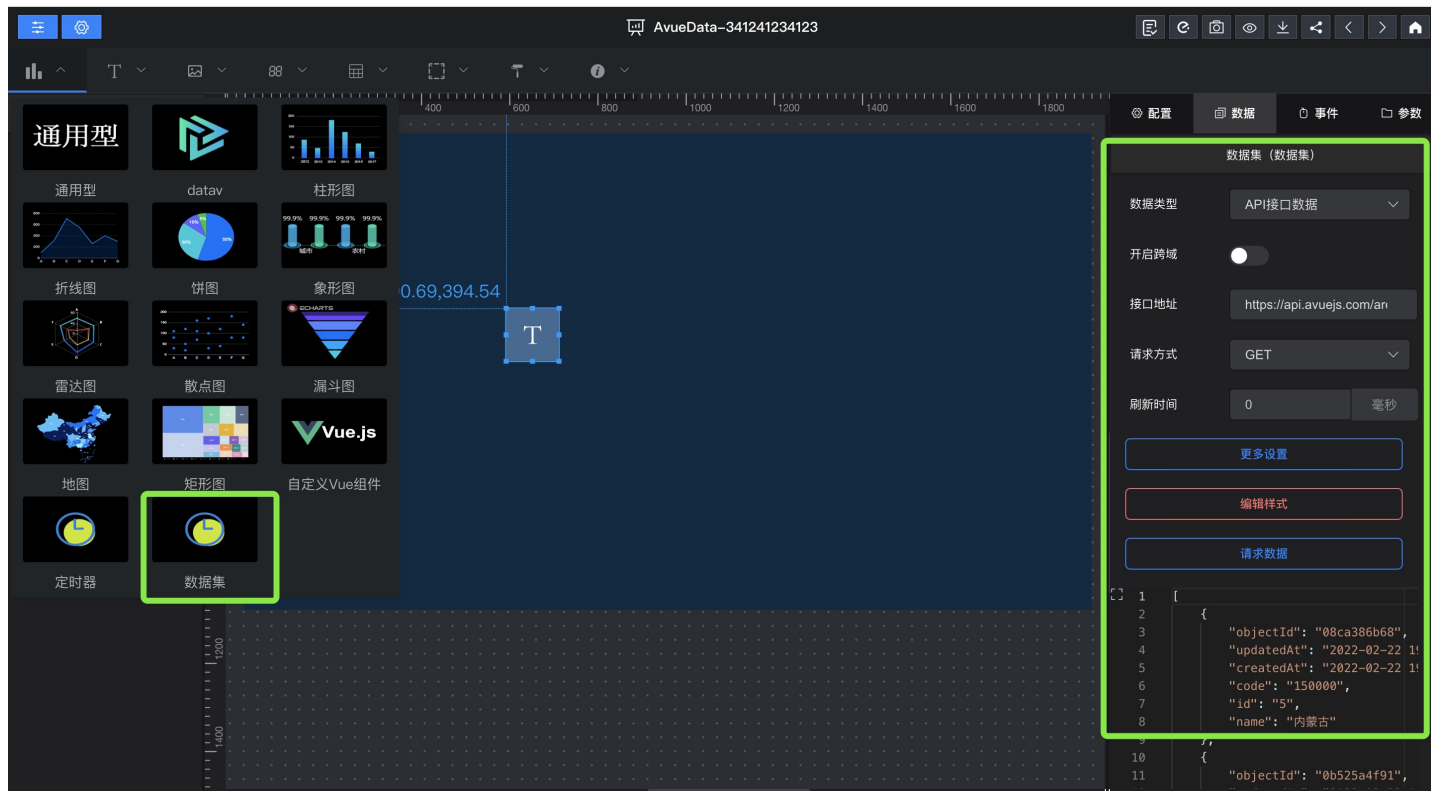
下方的例子是将数据处理成一个数组返回

```
//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]
    }
  }
}
```

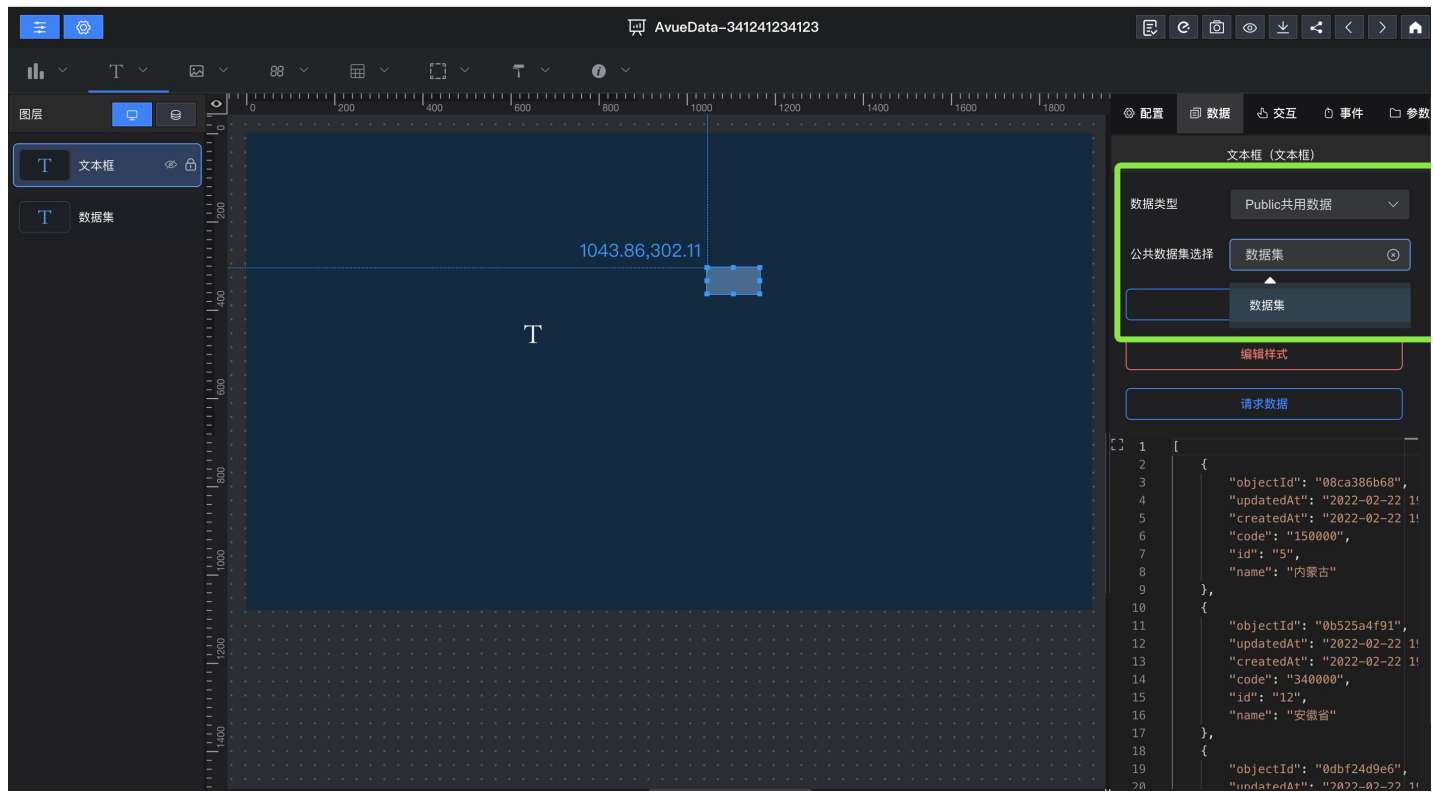
数据集共用

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



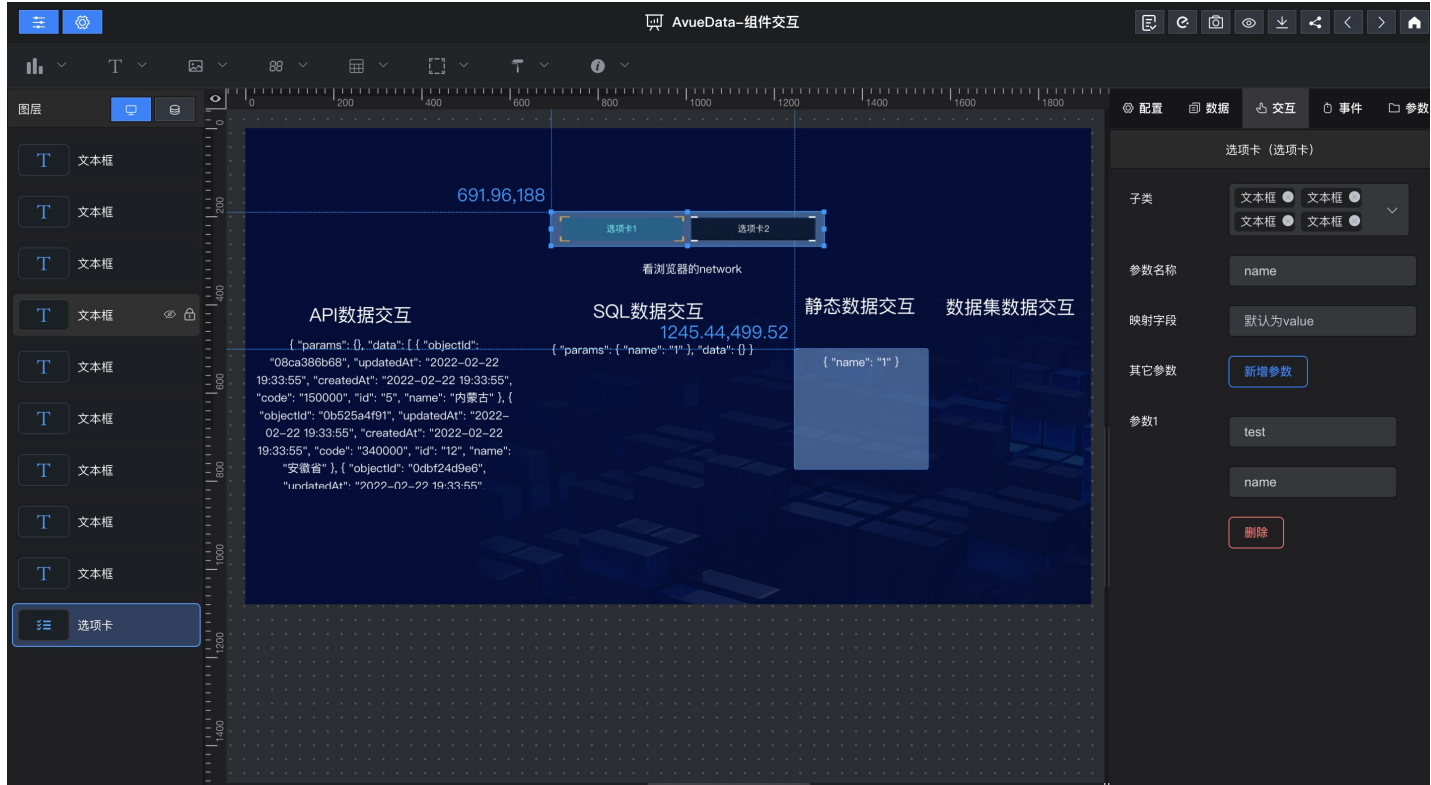
组件参数

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'
}
```

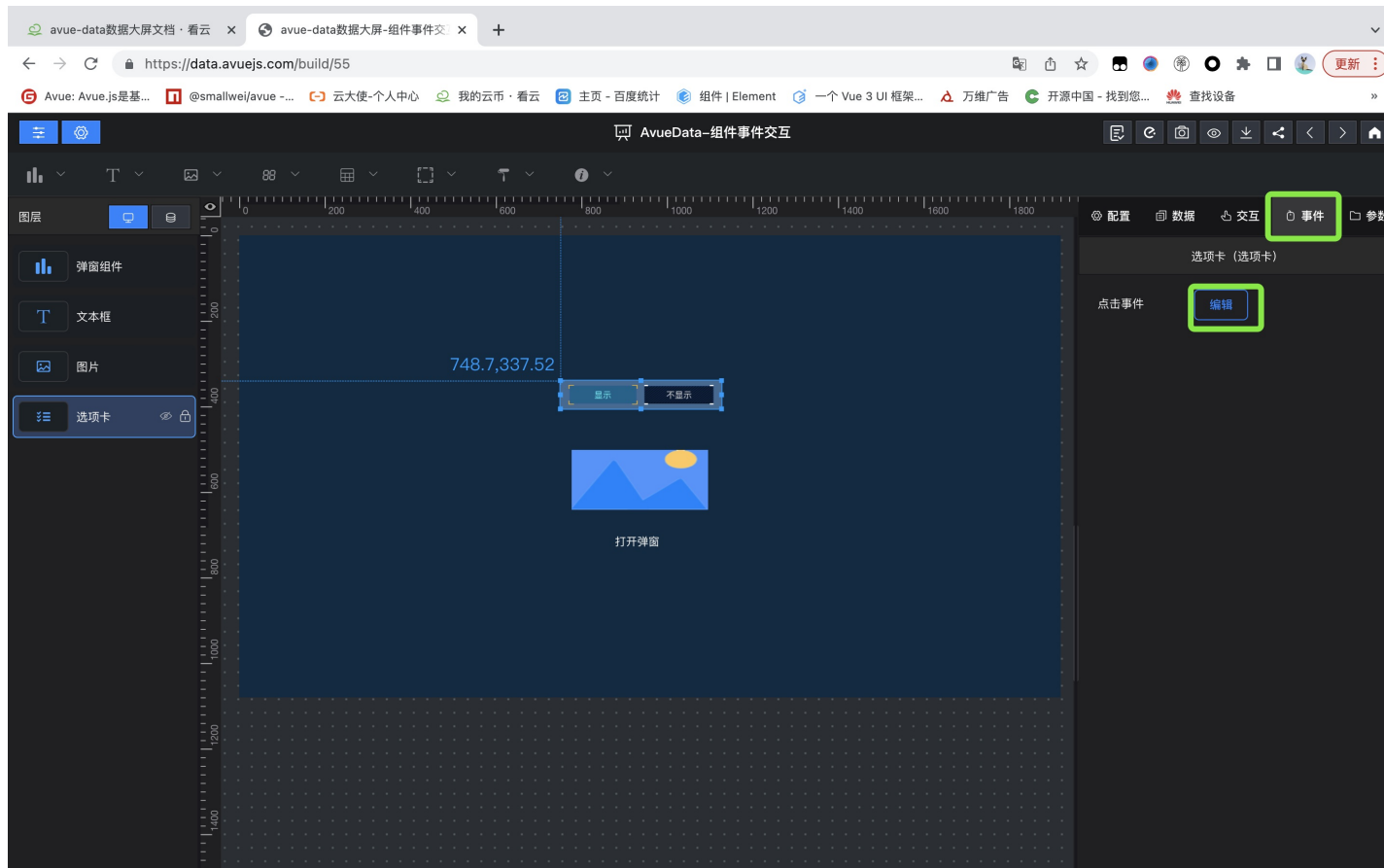
参数名称为test，映射字段为value，传递过去会是test=我是值这样的格式



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

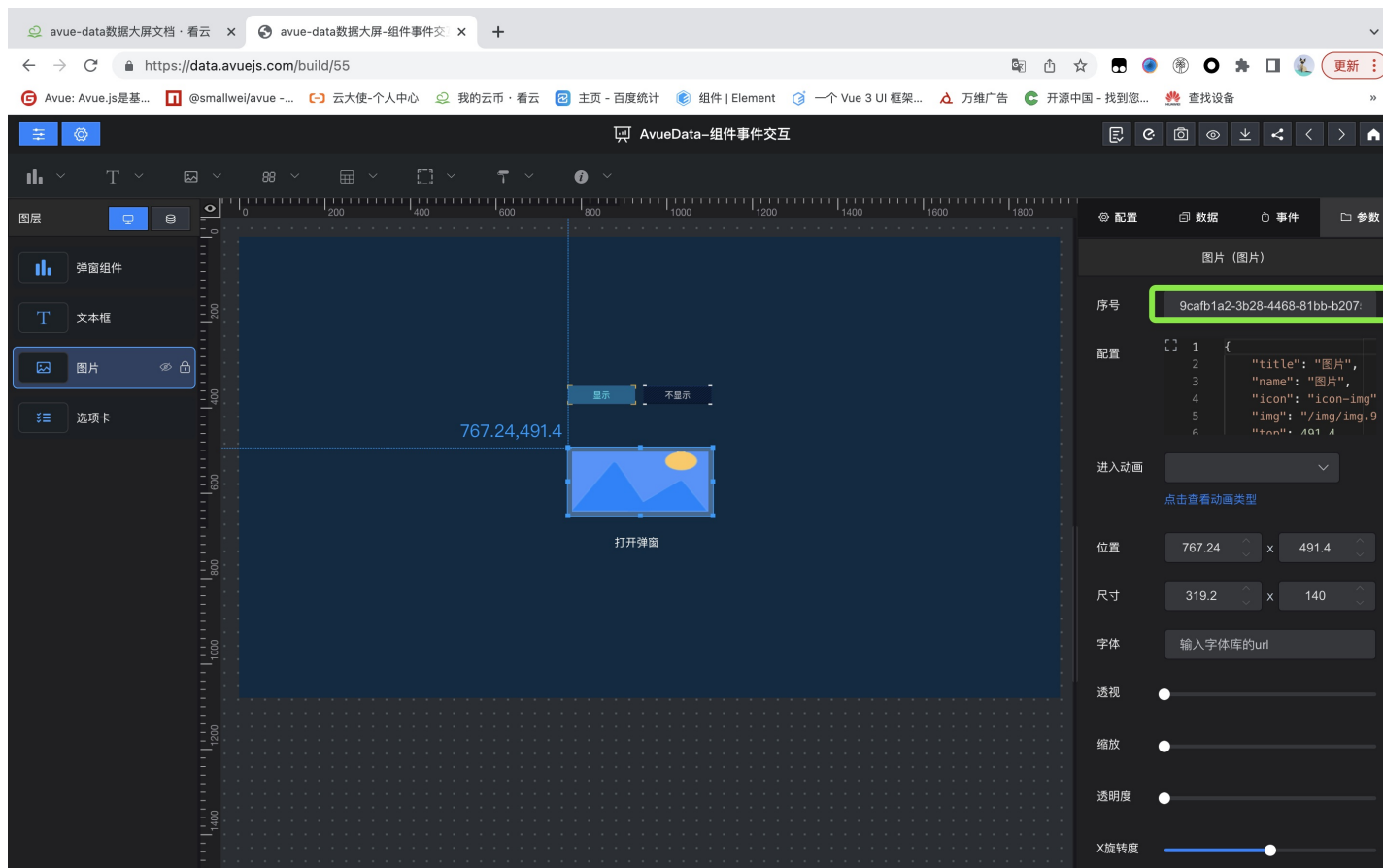


这里可JS处理逻辑，同时也可以拿到window.\$glob全局变量

- 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

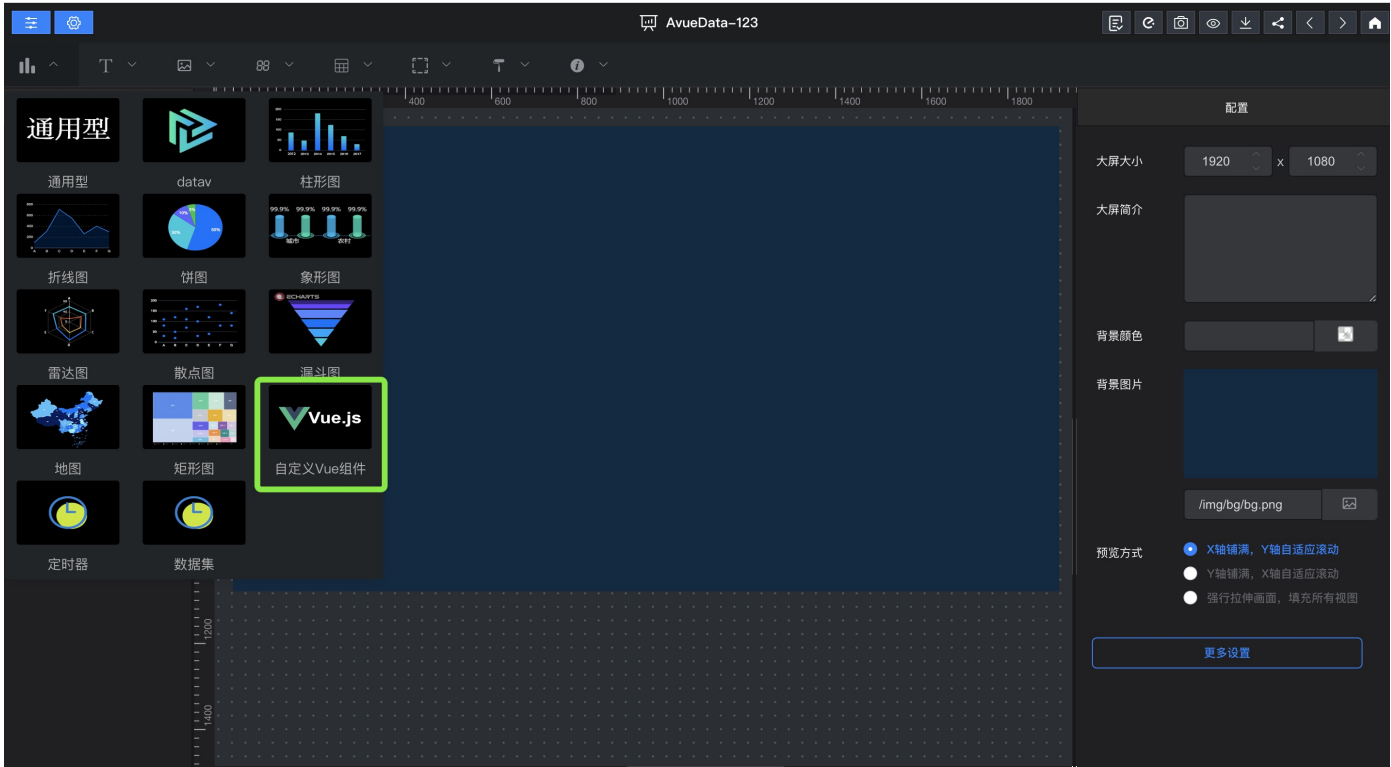


显隐逻辑如下:

```
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').then(()=>{

//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:

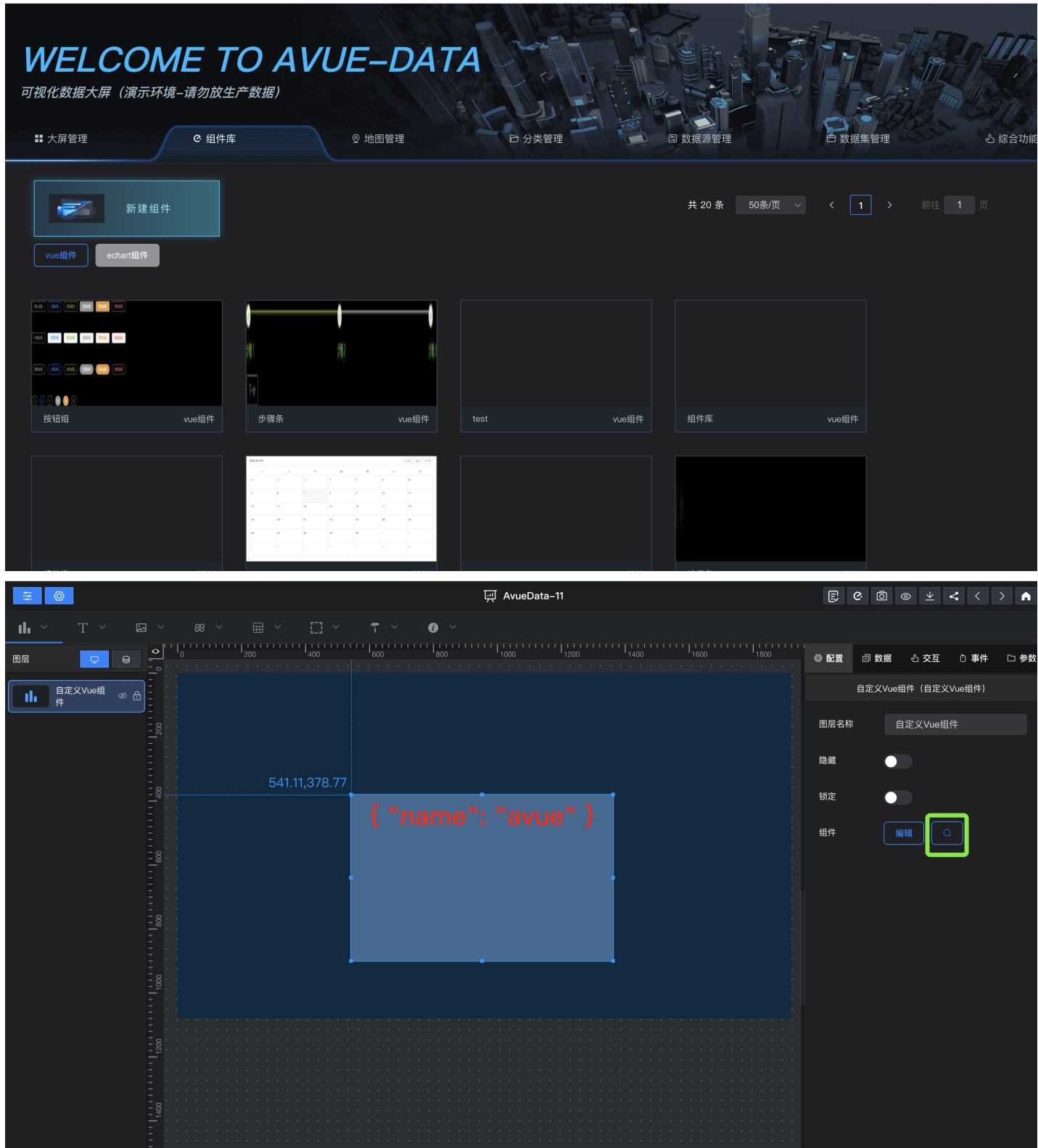
```
<template>
  <div class = "test">
    <el - button type = "success"@ click = "handleClick" > 点击我弹窗 < /el-button>
    <h1>{{dataChart}}<h1>
  </div>
</template>
<script>
```

```

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/jquery.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 >

```

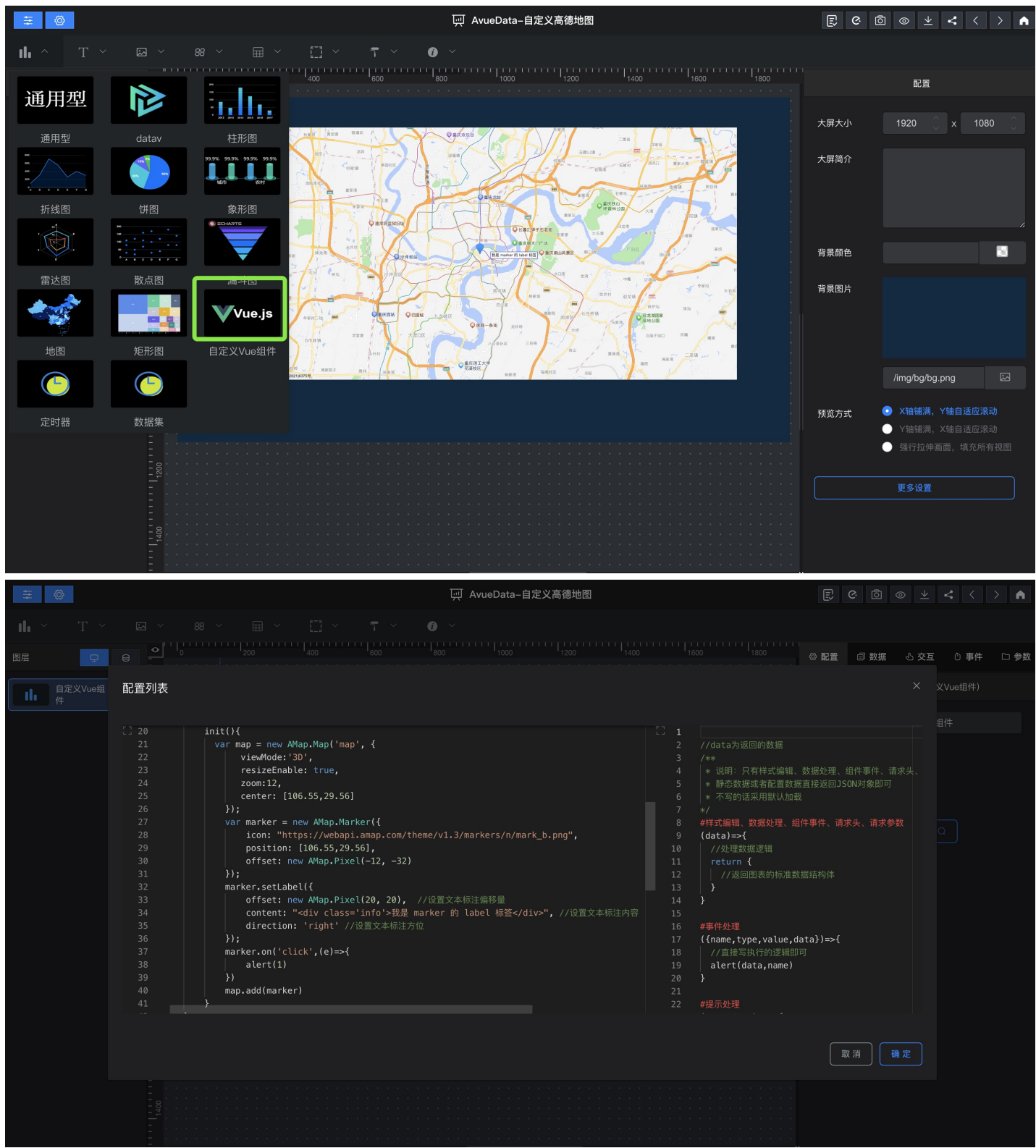
3.同时也可以把常用的组件新增到组件库中，多个大屏去使用



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](#)

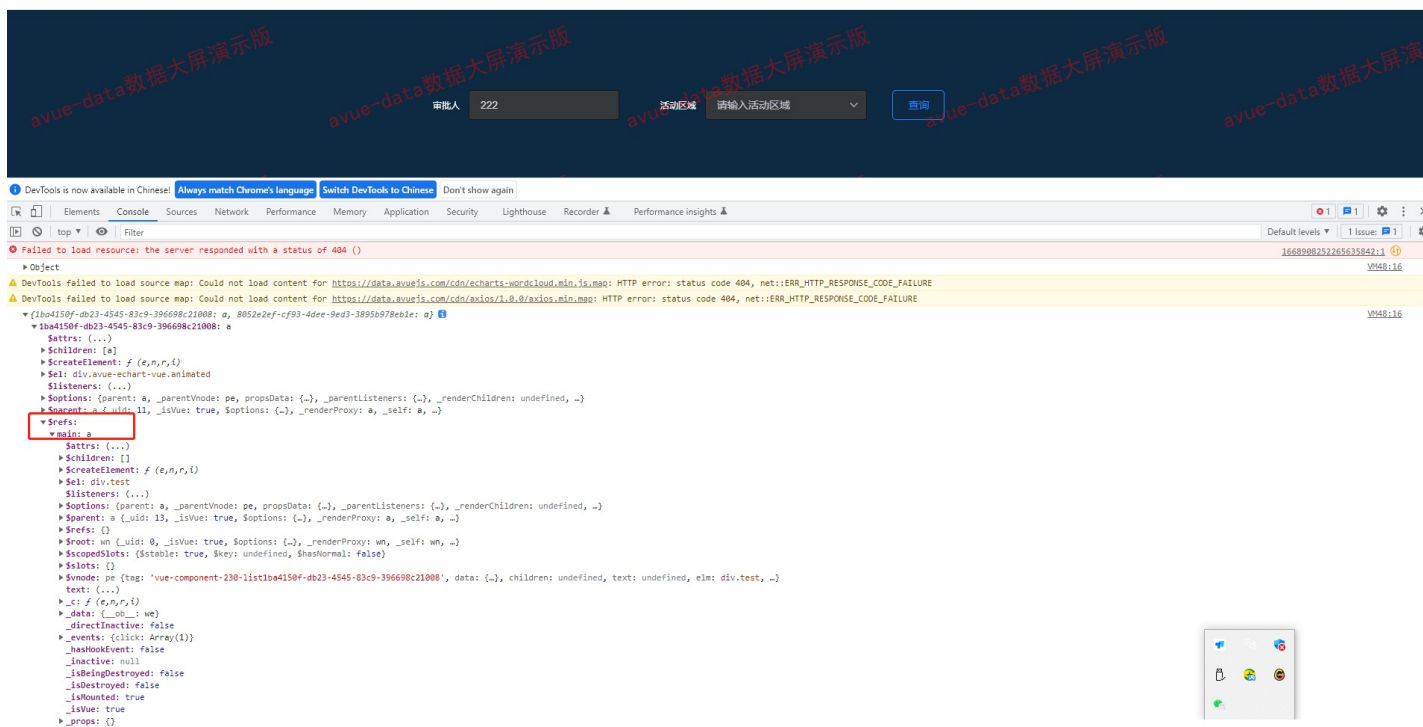
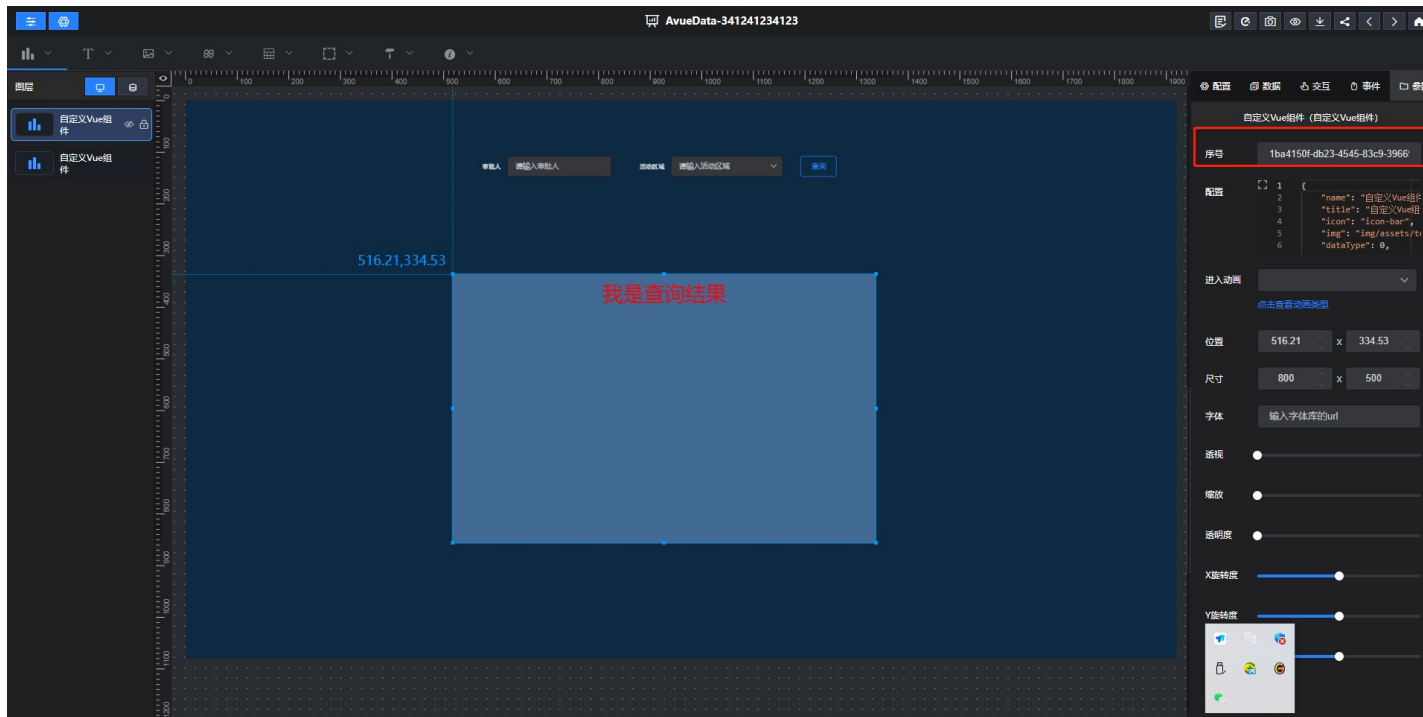
如果想操作其他组件，可以在自定义组件内调用父类this.\$parent.getItemRefs();方法获取其他组件的实例

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>
```

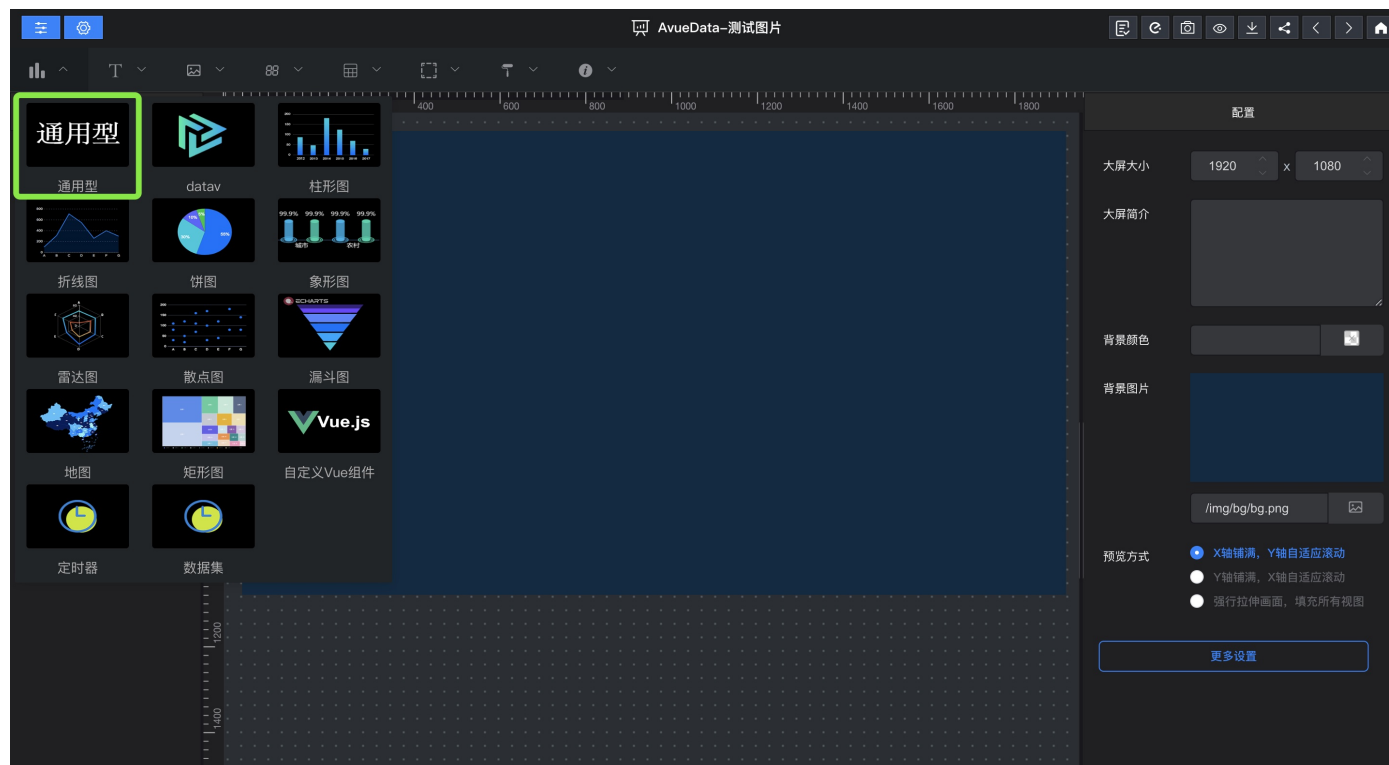
选择其中要传递组件的序号,选择要操作组件内部对象需要获取\$refs下的main

自定义Vue组件之间交互



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').then(()=>{
//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
}
```

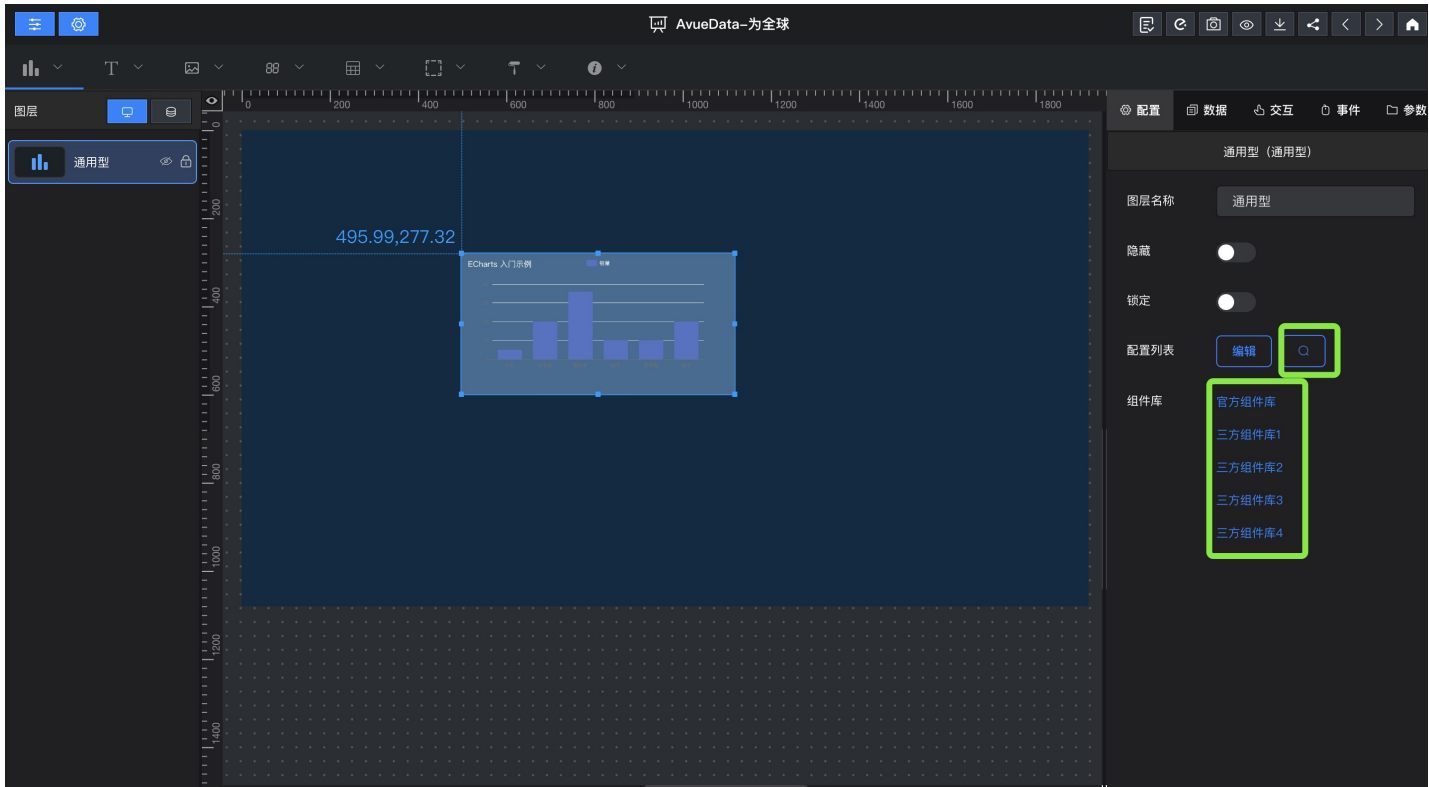
综合例子:

```
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





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'
    }
  }
}
```

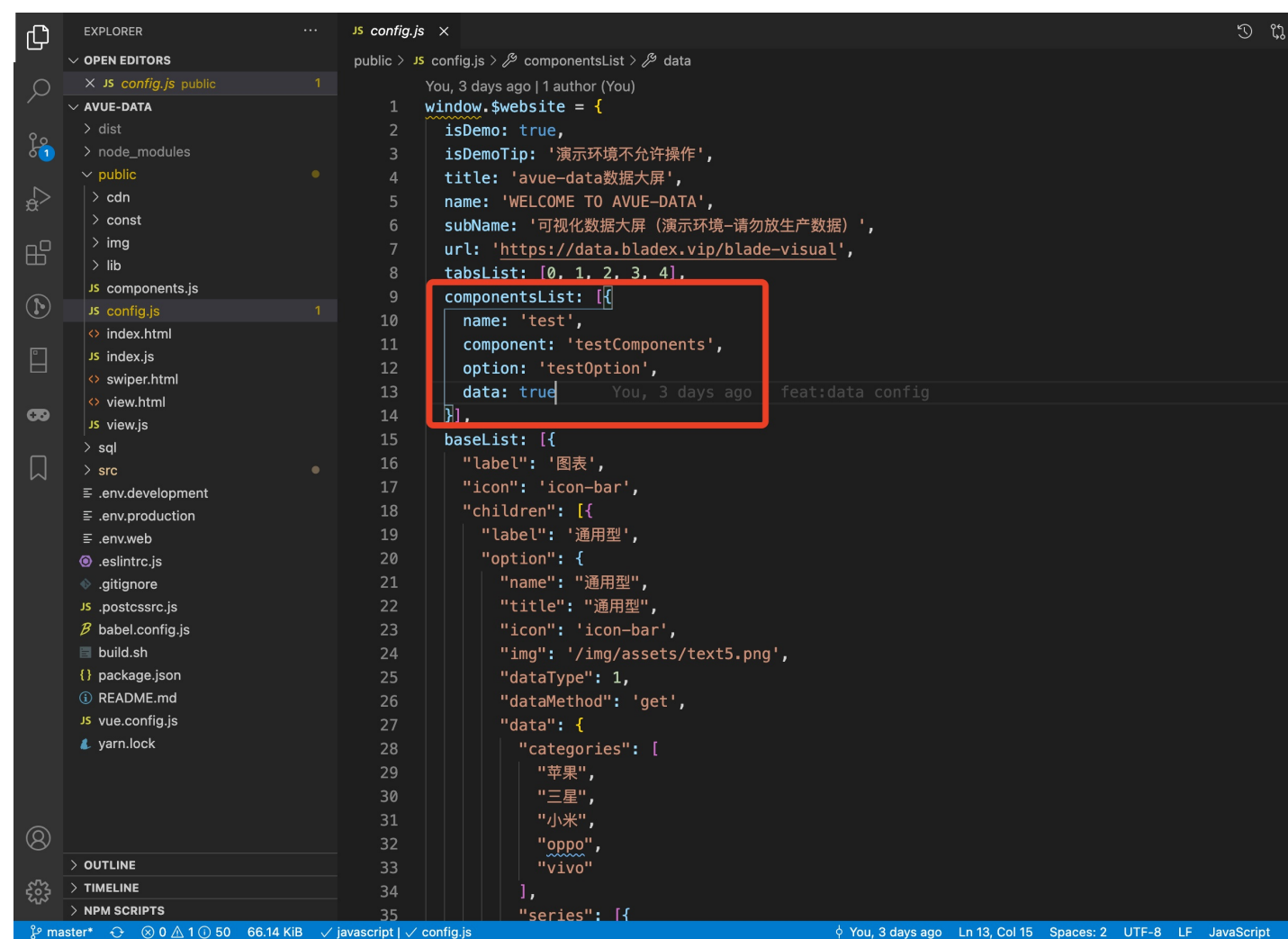
配置模块

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"]
}
```

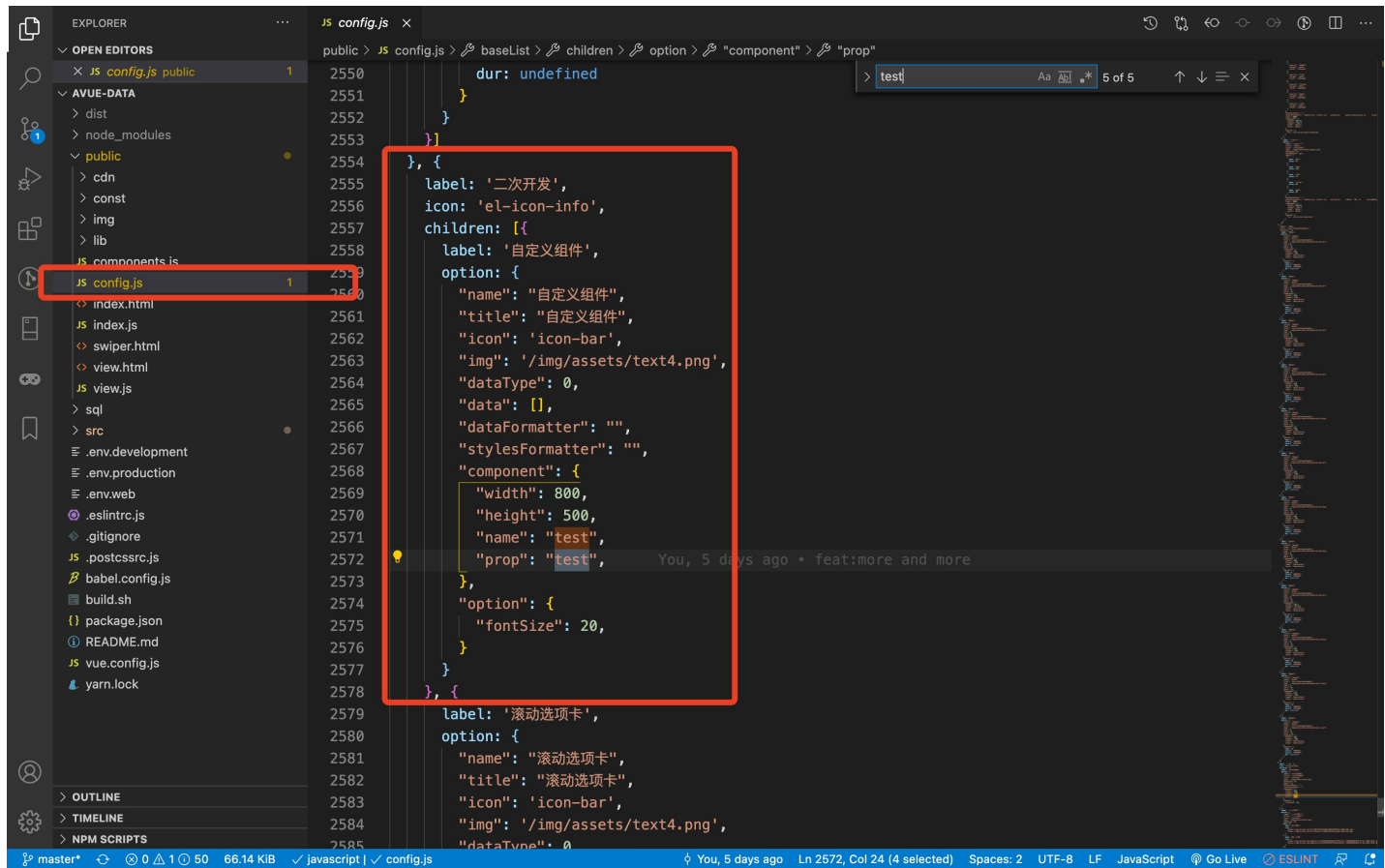
组件配置

配置文件文件路径/public/config.js

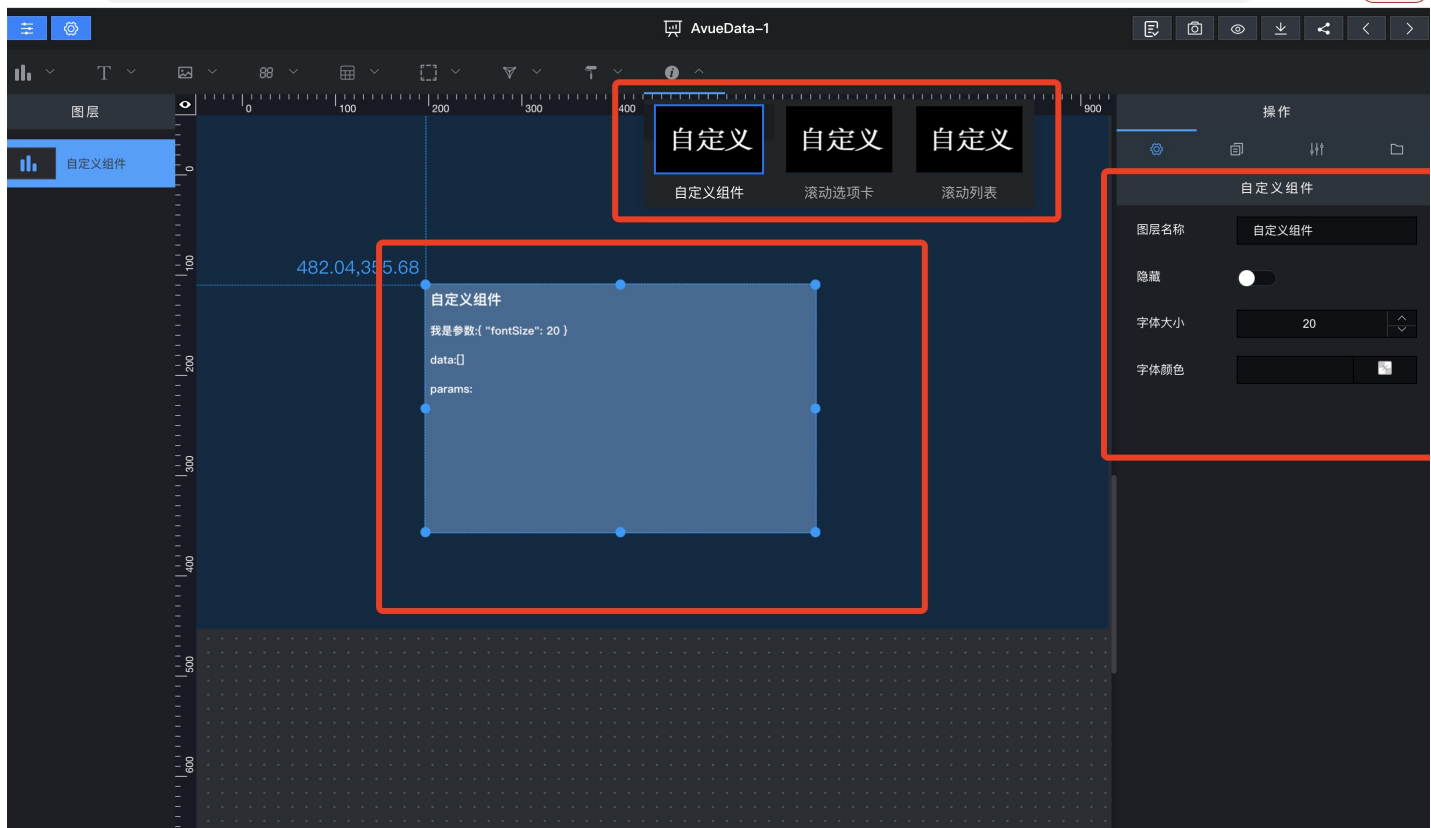


组件引用

配置文件文件路径/public/config.js



最终效果



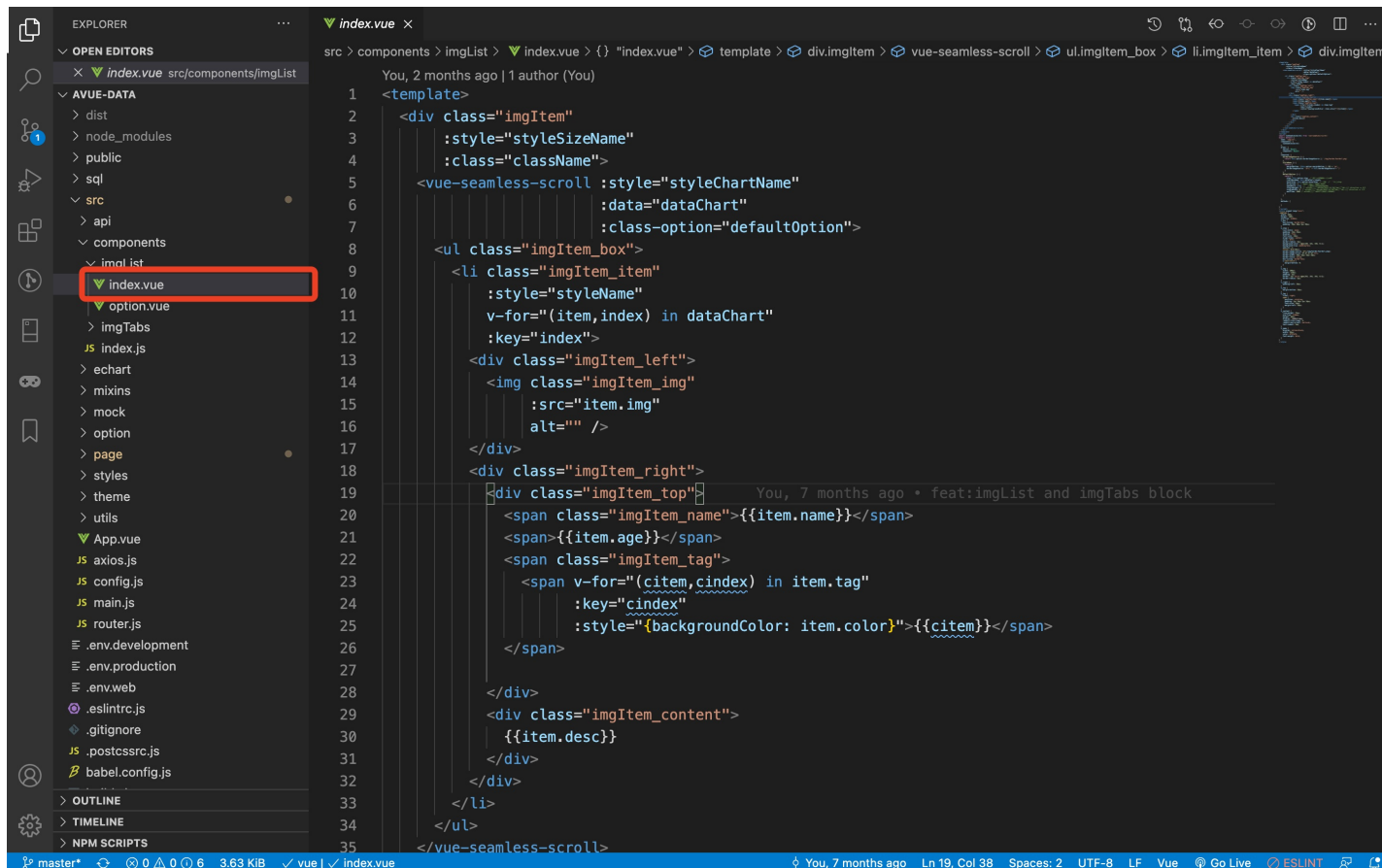
内部组件开发

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

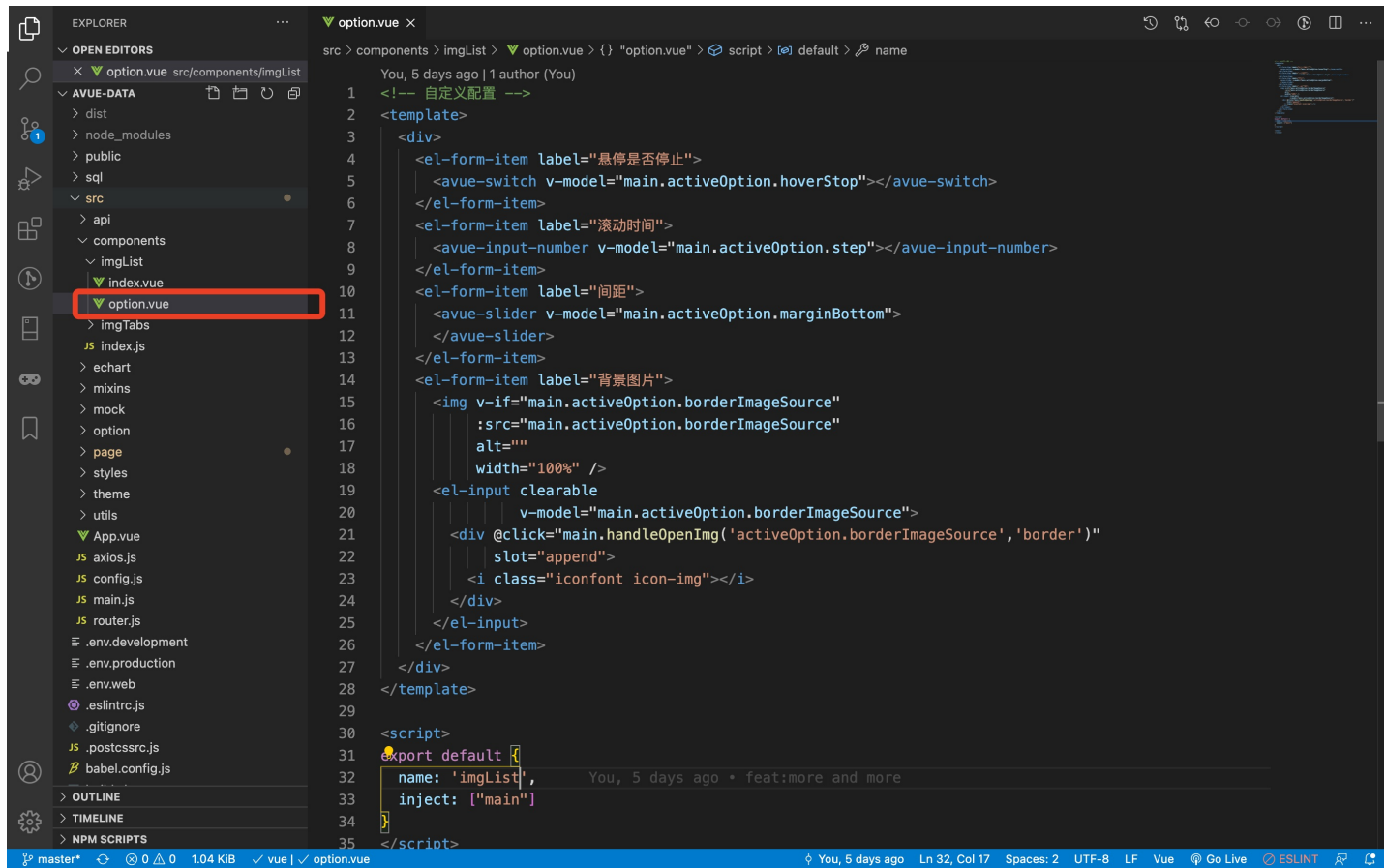
组件模块

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



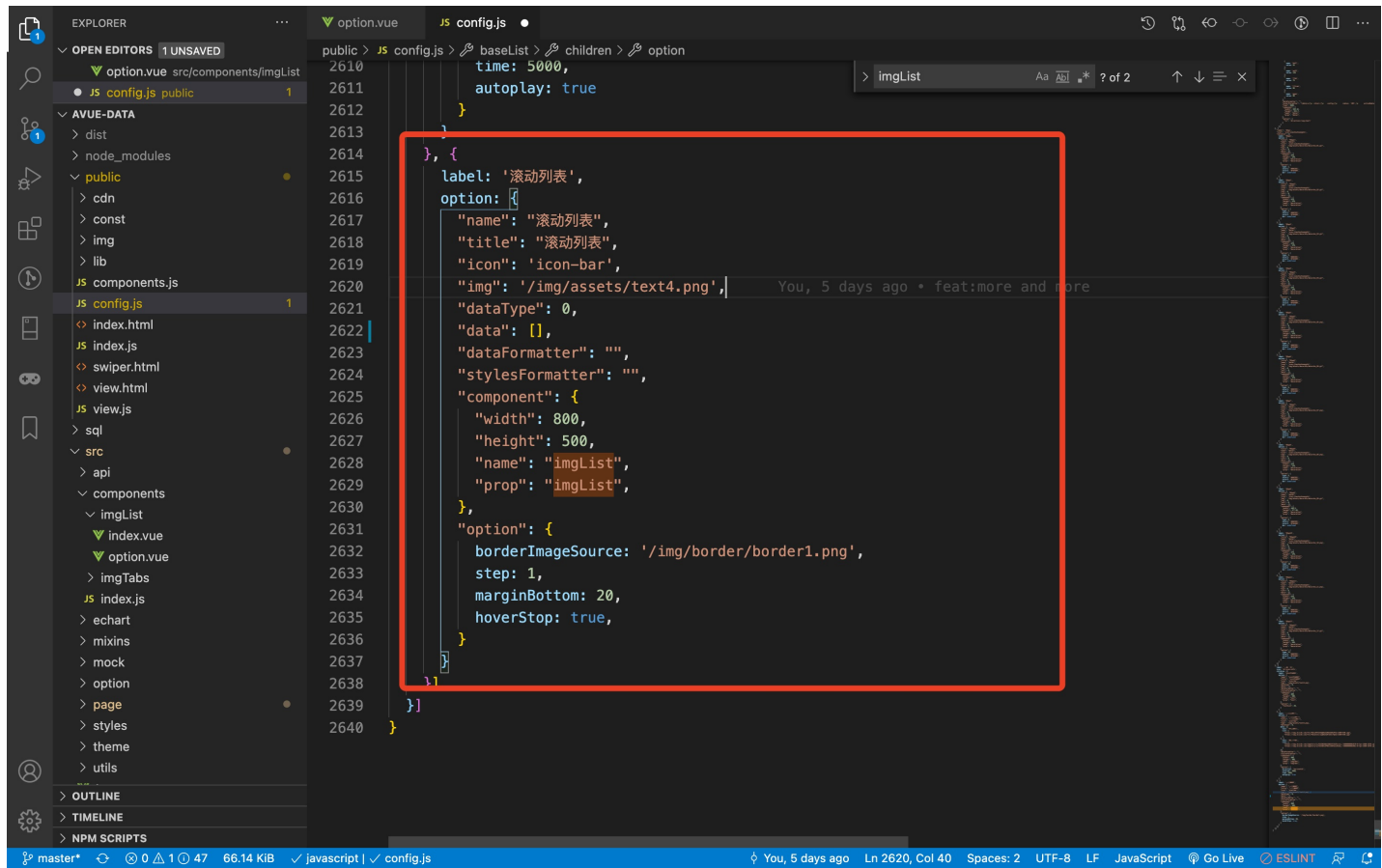
配置模块

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

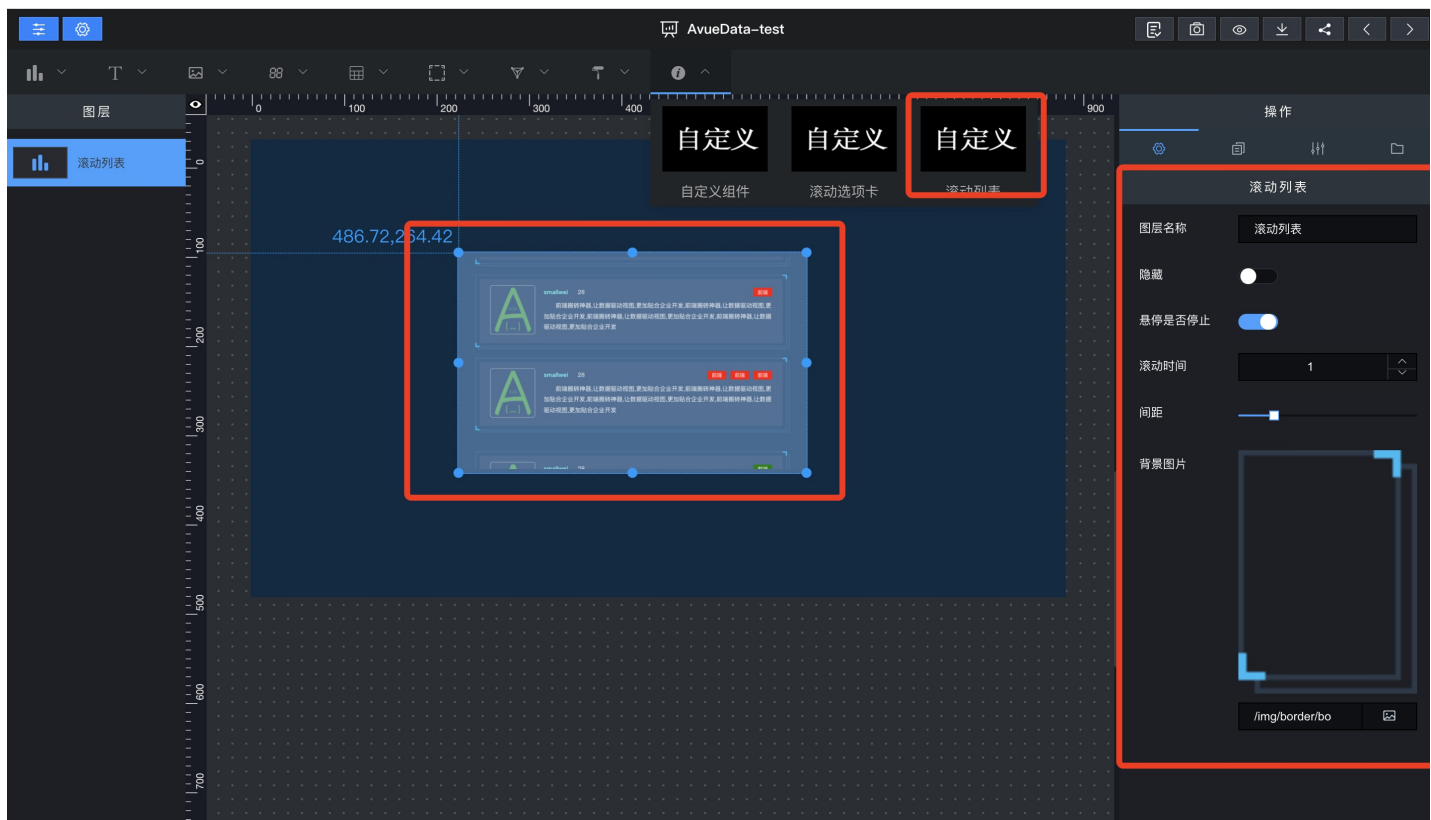


组件引用

配置文件文件路径/public/config.js



最终效果



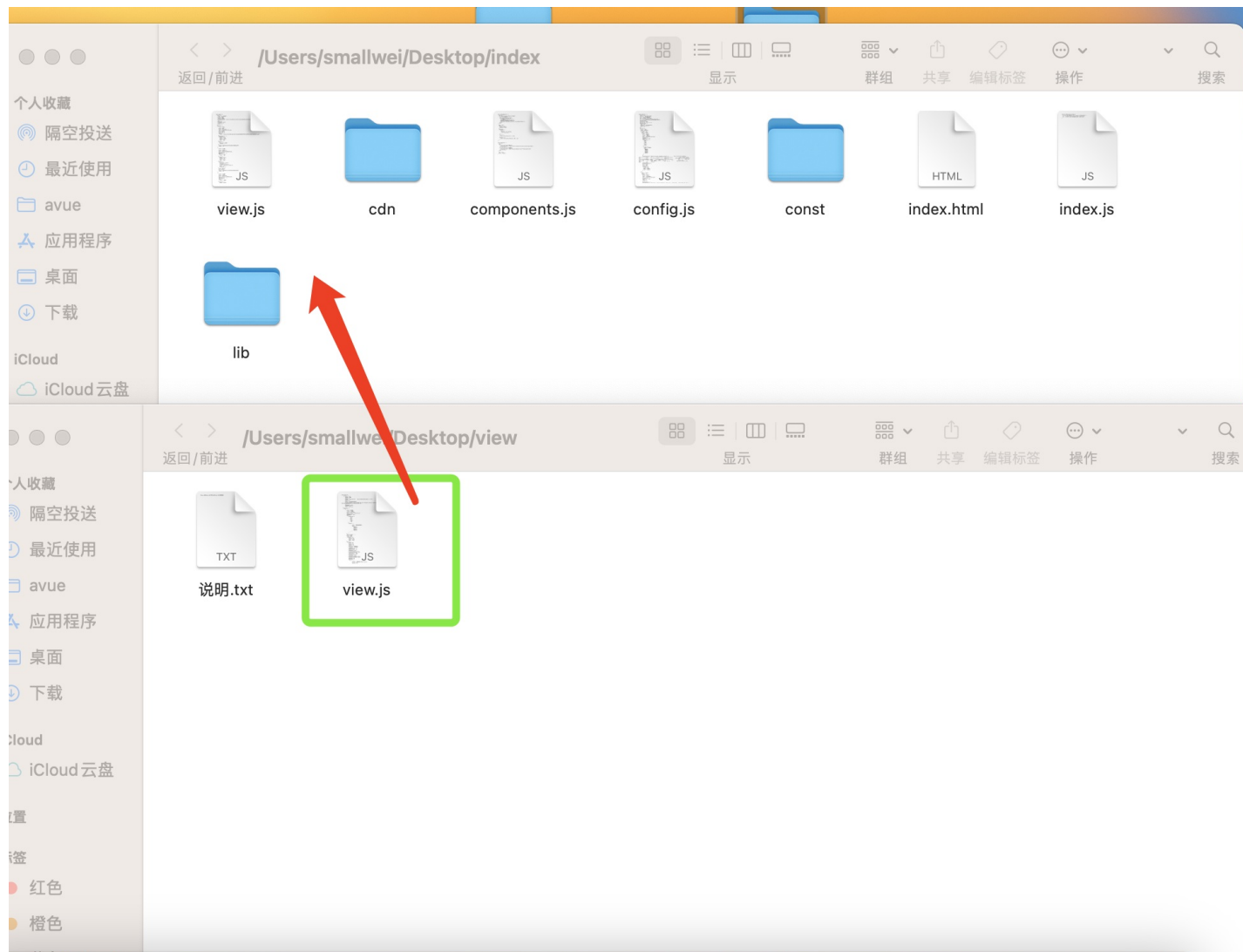
单独大屏导出部署

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](#)

[1.项目打包](#)

[2.Linux部署](#)

[3.Baota部署](#)

Nginx部署

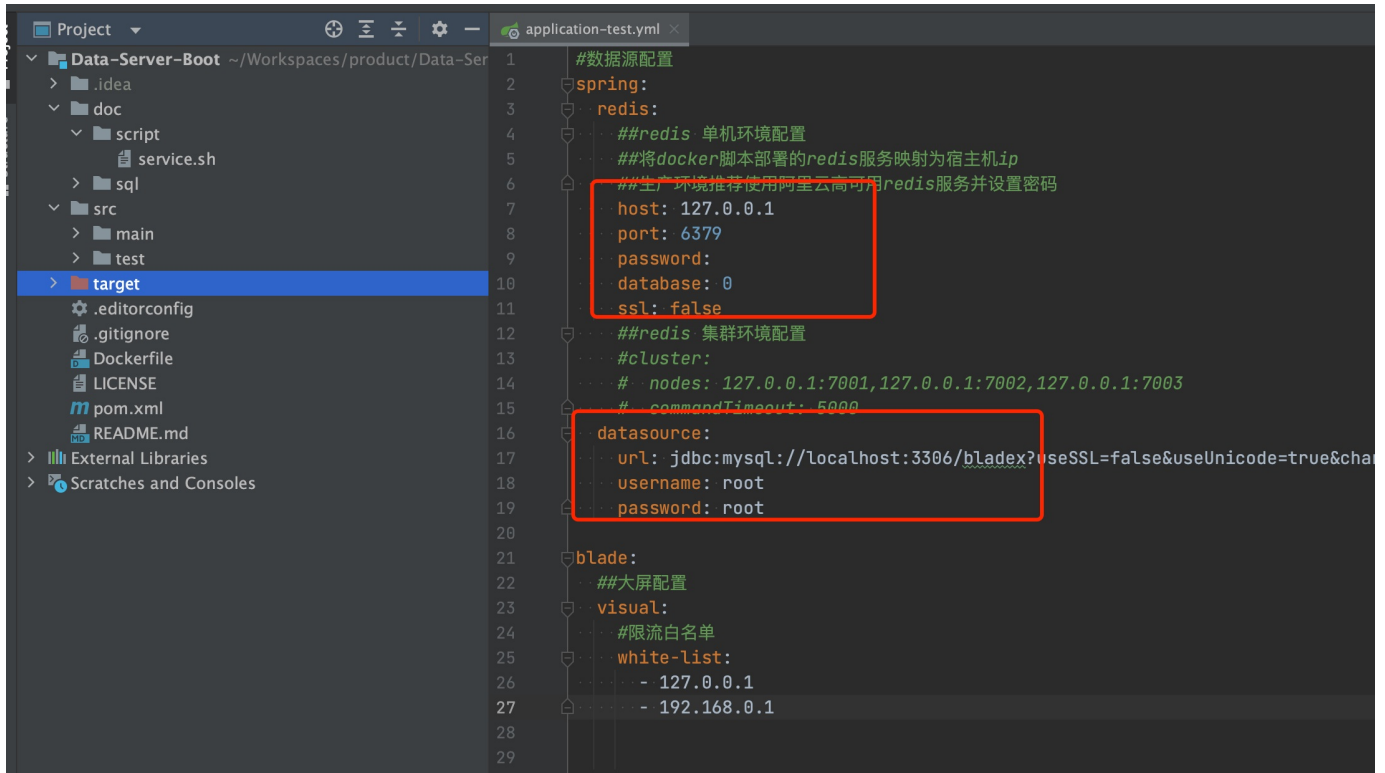
执行npm run build打出对应的目录

```
//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
}
```

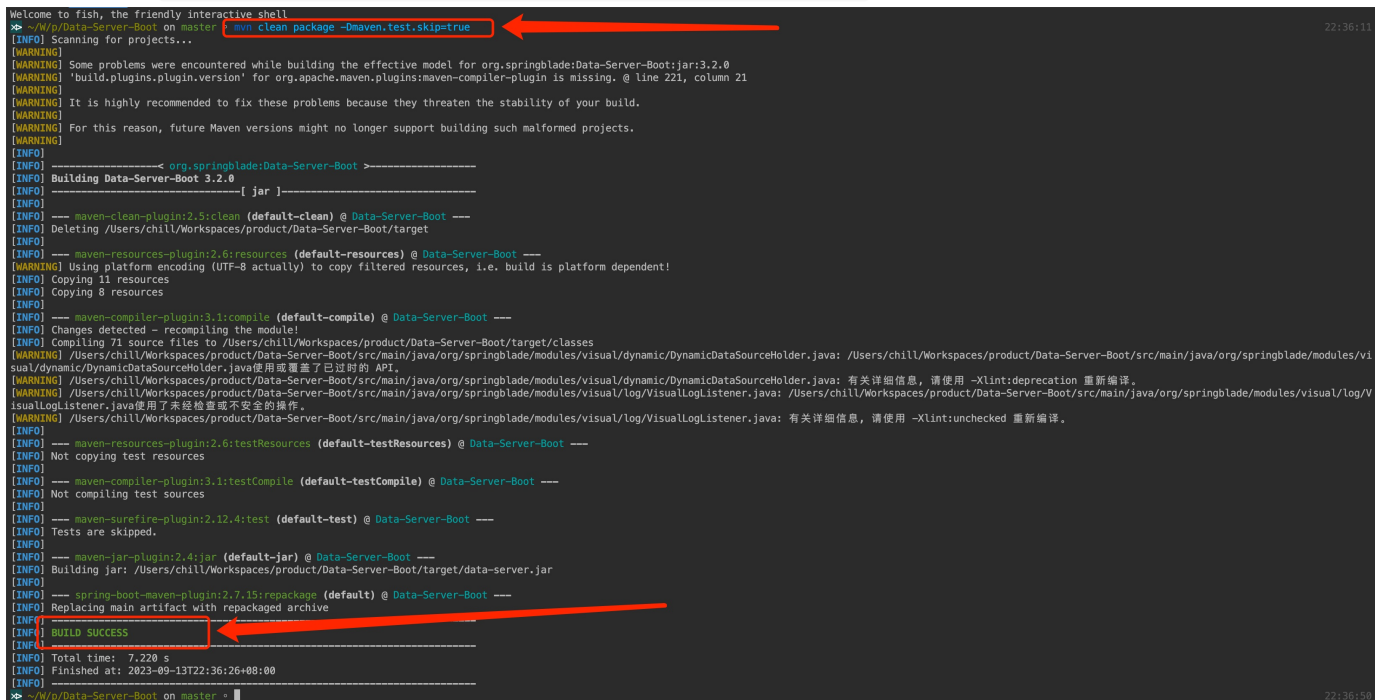
1.项目打包

一、后端打包

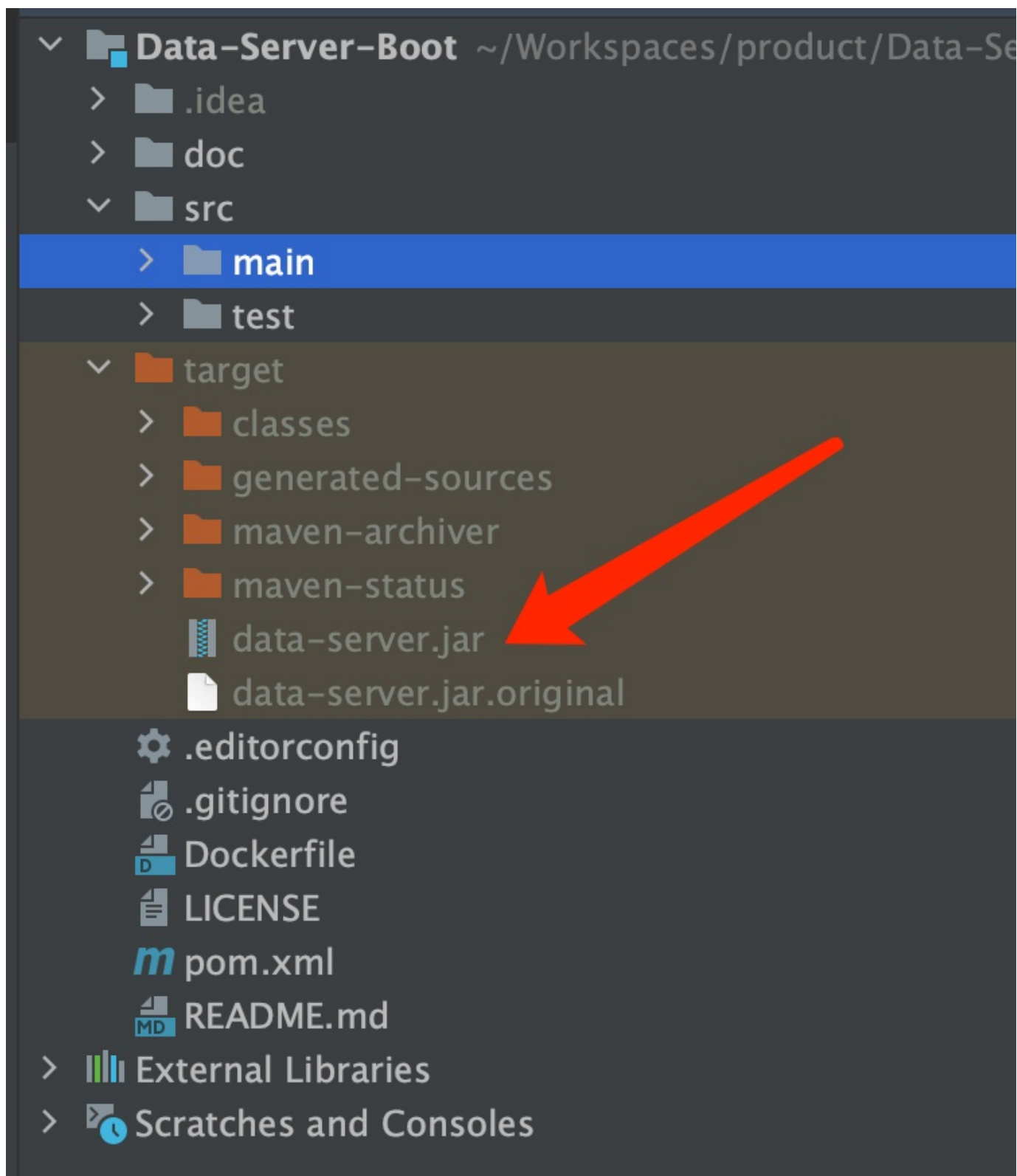
1. 将相关配置放到对应的文件



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

1.项目打包

```
config.js x
1  window.$website = {
2      isDemo: false,
3      isDemoTip: '演示环境不允许操作',
4      title: 'BladeX 数据大屏',
5      name: 'BladeX 数据大屏',
6      subName: '可视化数据大屏 (演示环境-请勿放生产数据)',
7      url: 'http://服务地址:服务端口/blade-visual',
8      autoSave: false,
9      autoSaveTime: 60000,
10     tabsList: [0, 1, 2, 3, 4, 5, 6],
11     componentsList: [{
12         name: 'test',
13         component: 'testComponents',
14         option: 'testOption',
15         data: true
16     }],
```

2. 根目录执行 yarn run build

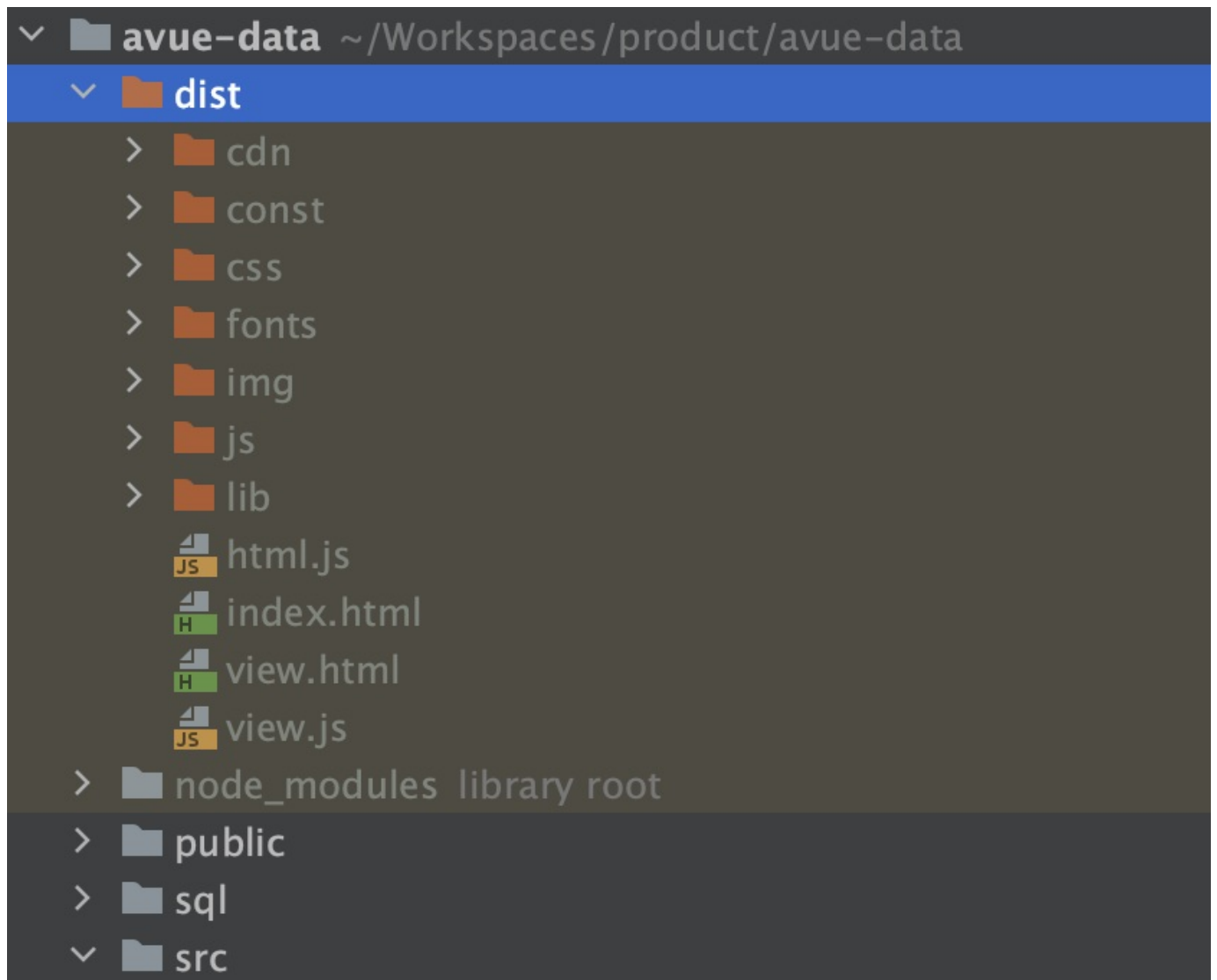
```
Terminal: Local x +
dist/cdn/vue/2.5.2/vue.min.js          91.47 KiB          33.30 KiB
dist/cdn/vue-router/3.0.1/vue-router.min.js 23.98 KiB          8.40 KiB
dist/cdn/axios/1.0.0/axios.min.js       12.67 KiB          4.59 KiB
dist/cdn/vuex/3.1.1/vuex.min.js         11.38 KiB          3.16 KiB
dist/cdn/vuex/2.4.1/vuex.min.js          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



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

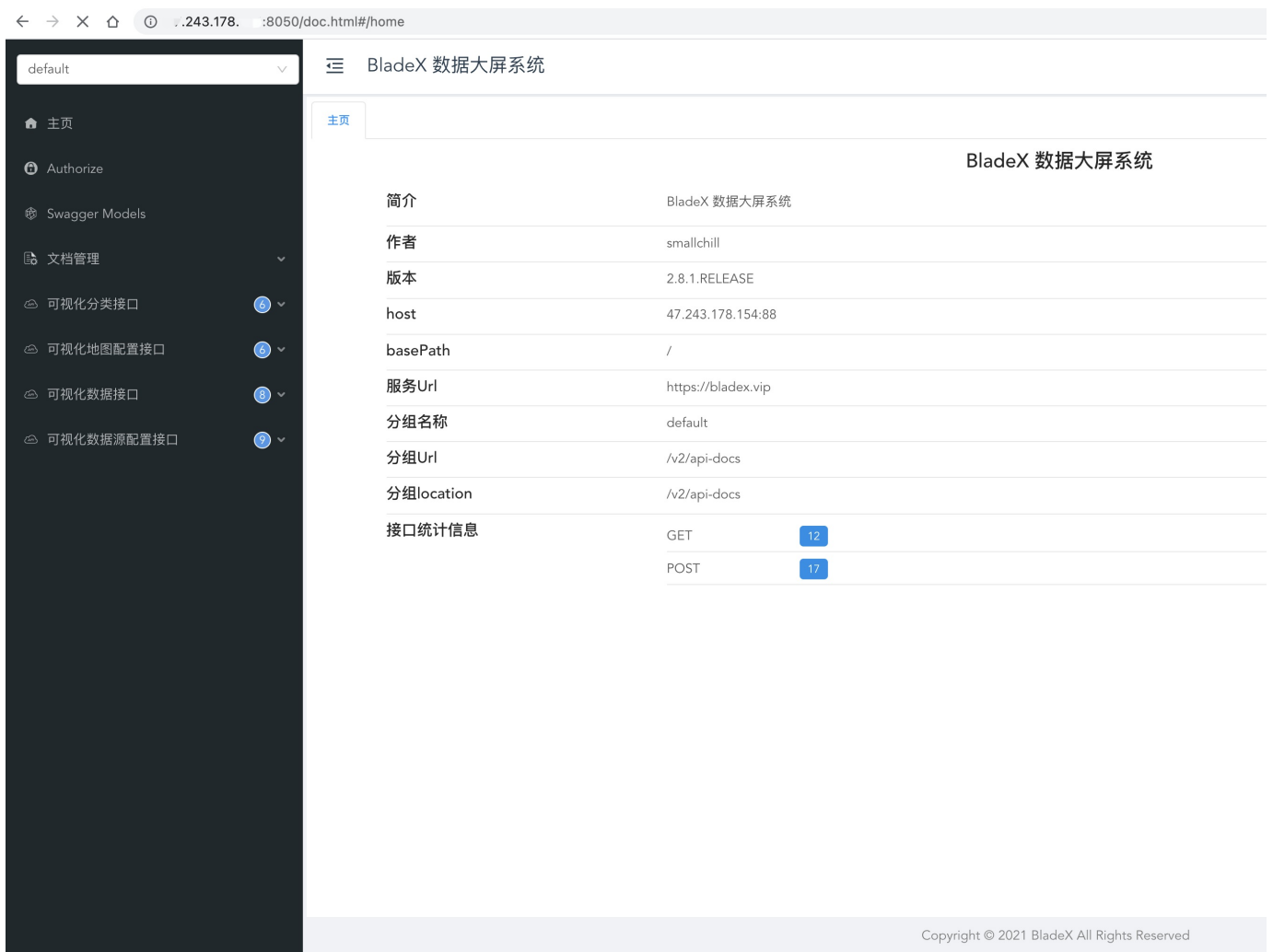
二、后端部署

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

```

autowire-marked constructor flagged as optional - this constructor is effectively required since there is no default constructor to fall back to: public org.springframework.boot.autoconfigure.web.ServerProperties)
2021-06-26 12:04:24.850 INFO 28163 --- [main] o.s.c.log.Logger: blade-visual: BladeLogger init success!
2021-06-26 12:04:25.028 INFO 28163 --- [main] pertySourcedRequestMappingHandlerMapping : Mapped URL path [/v2/api-docs] onto method [springfox.documentation.swagger2.web.Swagger2Controll
umentation(String, HttpServletRequest)]
2021-06-26 12:04:25.353 INFO 28163 --- [main] o.s.s.c.ThreadPoolTaskScheduler : Initializing ExecutorService 'taskScheduler'
2021-06-26 12:04:25.618 INFO 28163 --- [main] d.s.w.p.DocumentationPluginsBootstrapper : Documentation plugins bootstrapped
2021-06-26 12:04:25.622 INFO 28163 --- [main] d.s.w.p.DocumentationPluginsBootstrapper : Found 1 custom documentation plugin(s)
2021-06-26 12:04:25.694 INFO 28163 --- [main] s.d.s.w.s.ApiListingReferenceScanner : Scanning for api listing references
2021-06-26 12:04:25.983 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: removeUsingPOST_1
2021-06-26 12:04:26.010 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: saveUsingPOST_1
2021-06-26 12:04:26.013 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: updateUsingPOST_1
2021-06-26 12:04:26.047 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: listUsingGET_1
2021-06-26 12:04:26.081 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: detailUsingGET_1
2021-06-26 12:04:26.101 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: removeUsingPOST_2
2021-06-26 12:04:26.116 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: saveUsingPOST_2
2021-06-26 12:04:26.120 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: updateUsingPOST_2
2021-06-26 12:04:26.148 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: listUsingGET_2
2021-06-26 12:04:26.170 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: detailUsingGET_2
2021-06-26 12:04:26.184 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: removeUsingPOST_3
2021-06-26 12:04:26.188 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: saveUsingPOST_3
2021-06-26 12:04:26.191 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: updateUsingPOST_3
2021-06-26 12:04:26.209 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: listUsingGET_3
2021-06-26 12:04:26.217 INFO 28163 --- [main] .d.s.w.r.o.CachingOperationNameGenerator : Generating unique operation named: detailUsingGET_3
2021-06-26 12:04:26.273 INFO 28163 --- [main] io.undertow : starting server: Undertow - 2.0.33.Final
2021-06-26 12:04:26.290 INFO 28163 --- [main] org.xnio : XNIO version 3.3.8.Final
2021-06-26 12:04:26.311 INFO 28163 --- [main] org.xnio.nio : XNIO NIO Implementation Version 3.3.8.Final
2021-06-26 12:04:26.418 INFO 28163 --- [main] o.s.b.w.e.u.UndertowServletWebServer : Undertow started on port(s) 8050 (http) with context path ''
2021-06-26 12:04:26.436 INFO 28163 --- [main] o.s.modules.visual.VisualApplication : Started VisualApplication in 13.544 seconds (JVM running for 14.361)
2021-06-26 12:04:26.453 INFO 28163 --- [async-task-1] o.s.c.launch.StartEventListener : ---[BLADE-VISUAL]---启动完成, 当前使用的端口:[8050], 环境变量:[test]---
```

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
```

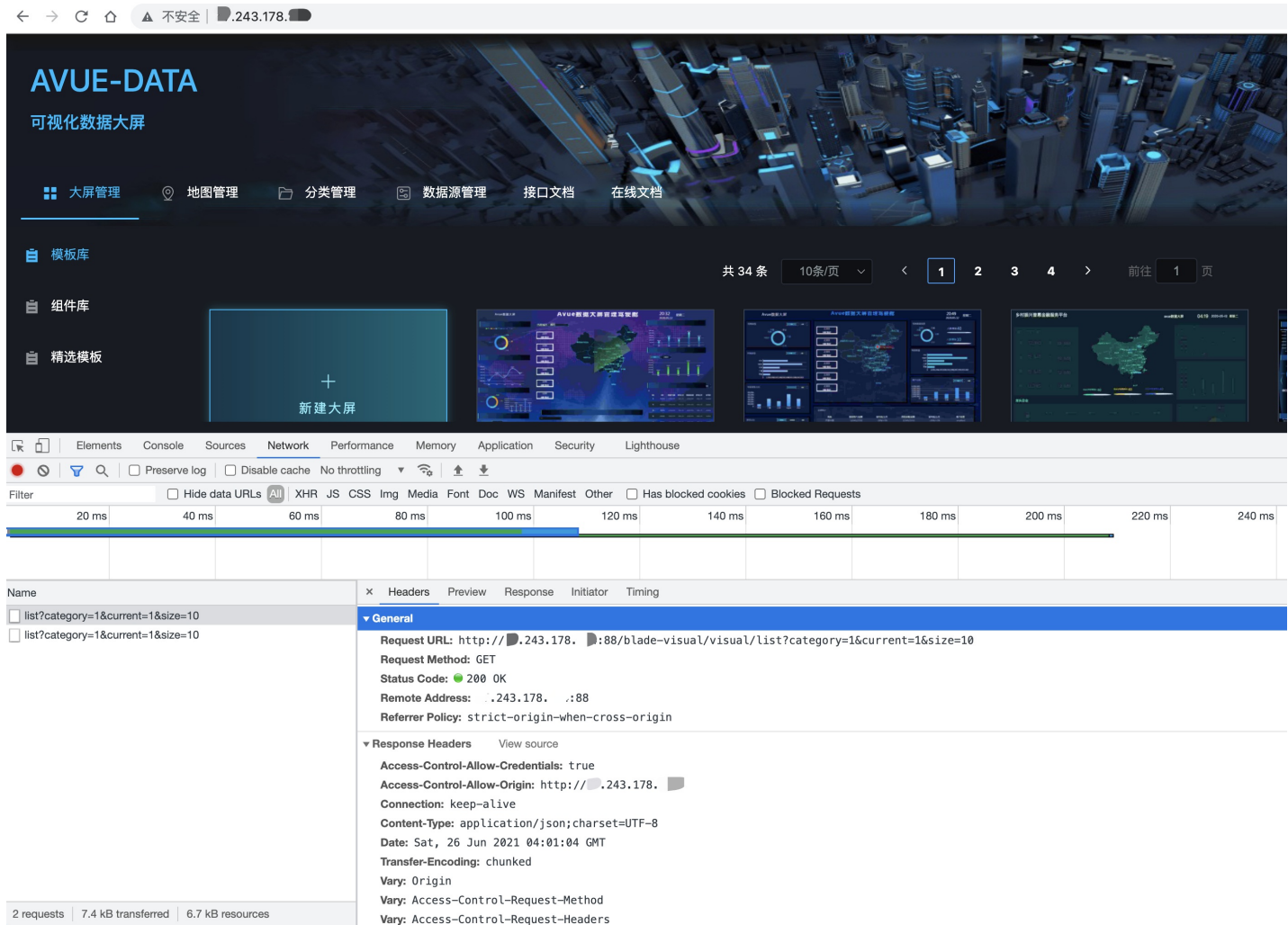
2. 到nginx配置文件进行配置增加如下配置，否则会出现404

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

```
server {  
    listen      80;  
    listen      [::]:80;  
    server_name _;  
    root        /usr/share/nginx/html;  
  
    # Load configuration files for the default server block.  
    include /etc/nginx/default.d/*.conf;  
  
    location / {  
        root /usr/share/nginx/html;  
        index index.html;  
        error_page 404 /index.html;  
    }  
  
    error_page 404 /404.html;  
    location = /404.html {  
    }  
  
    error_page 500 502 503 504 /50x.html;  
    location = /50x.html {  
    }  
}
```

部署在nginx的目录
可自行修改

3. 打开服务器地址，查看结果，可以正常访问，说明部署成功



3.Baota部署

[3.1安装宝塔](#)

[3.2基础部署](#)

[3.3大屏部署](#)

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://download.bt.cn/install/install_6.0.sh && sh install.sh
```

```
Welcome to Alibaba Cloud Elastic Compute Service !

[root@blade-test ~]# yum install -y wget && wget -O install.sh http://download.bt.cn/install/install_6.0.sh && sh install.sh
已加载插件：fastestmirror
Determining fastest mirrors
base                                | 3.6 kB  00:00:00
epel                                | 4.7 kB  00:00:00
extras                              | 2.9 kB  00:00:00
updates                             | 2.9 kB  00:00:00
(1/7): epel/x86_64/group_gz         | 95 kB  00:00:00
(2/7): base/7/x86_64/primary_db     | 6.1 MB  00:00:00
(3/7): extras/7/x86_64/primary_db   | 224 kB  00:00:00
(4/7): epel/x86_64/primary_db       | 6.9 MB  00:00:00
(5/7): base/7/x86_64/group_gz       | 153 kB  00:00:00
(6/7): updates/7/x86_64/primary_db  | 5.6 MB  00:00:00
(7/7): epel/x86_64/updateinfo        | 1.0 MB  00:00:00
软件包 wget-1.14-18.el7_6.1.x86_64 已安装并且是最新版本
无须任何处理
--2021-02-11 12:52:49--  http://download.bt.cn/install/install_6.0.sh
正在解析主机 download.bt.cn (download.bt.cn)... 158.247.208.19, 2001:19f0:7001:54fc:5400:2ff:fe9b:97f9
正在连接 download.bt.cn (download.bt.cn)|158.247.208.19|:80... 已连接。
已发出 HTTP 请求，正在等待响应... 200 OK
长度：25386 (25K) [application/octet-stream]
正在保存至: "install.sh"

100%[=====>] 25,386 ---K/s 用时 0.04s

2021-02-11 12:52:49 (567 KB/s) - 已保存 "install.sh" [25386/25386]
```

```
-----
| The WebPanel URL will be http://SERVER_IP:8888 when installed.
-----
Do you want to install Bt-Panel to the /www directory now?(y/n): y

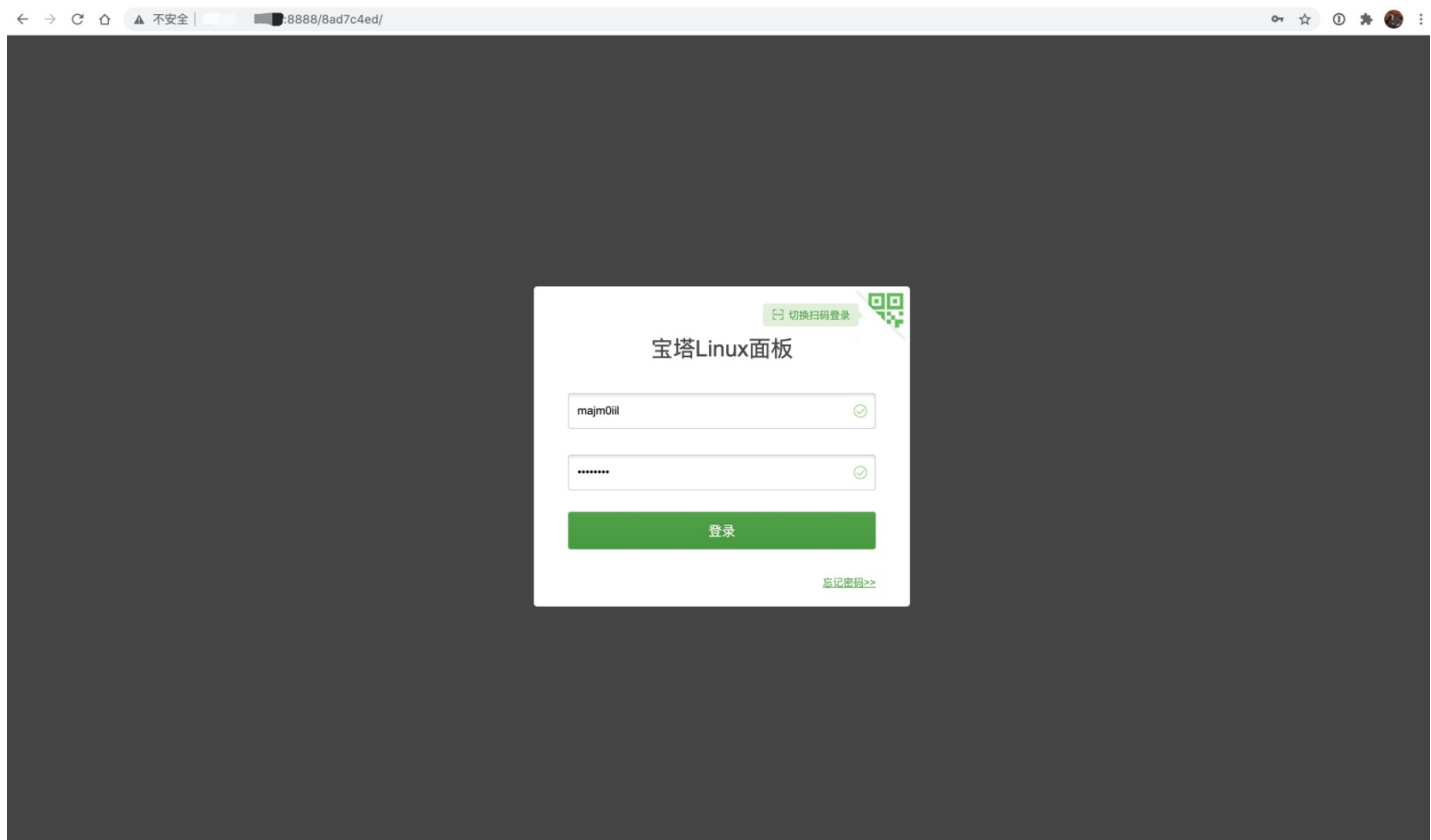
Selected download node...
Download node: http://45.76.53.20
-----
Synchronizing system time...
Thu Feb 11 12:53:00 CST 2021
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
base                                | 3.6 kB  00:00:00
epel                                | 4.7 kB  00:00:00
extras                              | 2.9 kB  00:00:00
updates                             | 2.9 kB  00:00:00
Resolving Dependencies
--> Running transaction check
----> Package ntp.x86_64 0:4.2.6p5-29.el7.centos.2 will be installed
--> Processing Dependency: ntpdate = 4.2.6p5-29.el7.centos.2 for package: ntp-4.2.6p5-29.el7.centos.2.x86_64
--> Processing Dependency: libopts.so.25()(64bit) for package: ntp-4.2.6p5-29.el7.centos.2.x86_64
--> Running transaction check
----> Package autogen-libopts.x86_64 0:5.18-5.el7 will be installed
----> Package ntpdate.x86_64 0:4.2.6p5-29.el7.centos.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

3.1安装宝塔

```
Starting Bt-Panel.... done
Starting Bt-Tasks... done
username: majm0iil
Stopping Bt-Tasks... done
Stopping Bt-Panel... done
Starting Bt-Panel.... done
Starting Bt-Tasks... done
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Package firewall-0.6.3-12.el7.noarch already installed and latest version
Nothing to do
Created symlink from /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service to /usr/lib/systemd/system/firewalld.service.
Created symlink from /etc/systemd/system/multi-user.target.wants/firewalld.service to /usr/lib/systemd/system/firewalld.service.
success
=====
Congratulations! Installed successfully!
=====
外网面板地址: http://:8888/8ad7c4ed
内网面板地址: http://172.31.24.175:8888/8ad7c4ed
username: majm0iil
password: 4ad45c77
If you cannot access the panel,
release the following panel port [8888] in the security group
若无法访问面板, 请检查防火墙/安全组是否有放行面板[8888]端口
=====
Time consumed: 0 Minute!
```

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

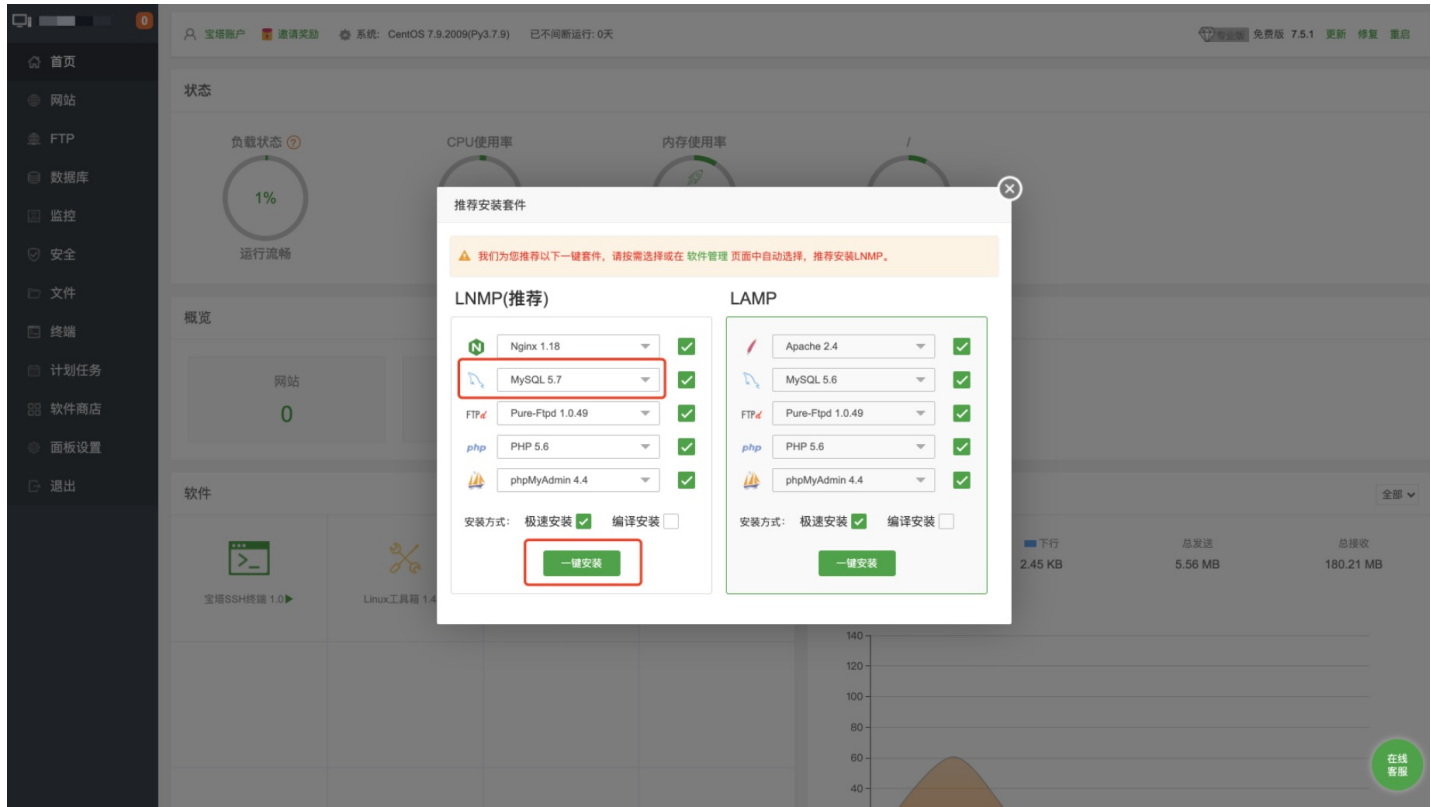


4. 初次登陆也许会相对耗时，此时系统在初始化，稍等片刻就可以看到进入了面板欢迎页，选择同意并进入面板



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

3.1安装宝塔



消息盒子

任务列表(5)

消息列表(0)

执行日志

● 安装[nginx-1.18]

正在安装 -- | 关闭

Upgrade 1 Package (+ 6 Dependent packages)

Total download size: 81 M

Downloading packages:

Delta RPMs disabled because /usr/bin/applydeltarpm not installed.

Total 40 MB/s | 81 MB 00:02

Running transaction check

Running transaction test

● 安装[mysql-5.7]

等待 | 删除

● 安装[pureftpd-1.0.49]

等待 | 删除

● 安装[php-5.6]

等待 | 删除

● 安装[phpmyadmin-4.4]

等待 | 删除

若任务长时间未执行，请尝试在首页点【重启面板】来重置任务队列

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=

×

绑定宝塔官网账号

恭喜您，宝塔面板已经安装成功。

绑定宝塔官网账号，即可开始使用 ?

手机

密码

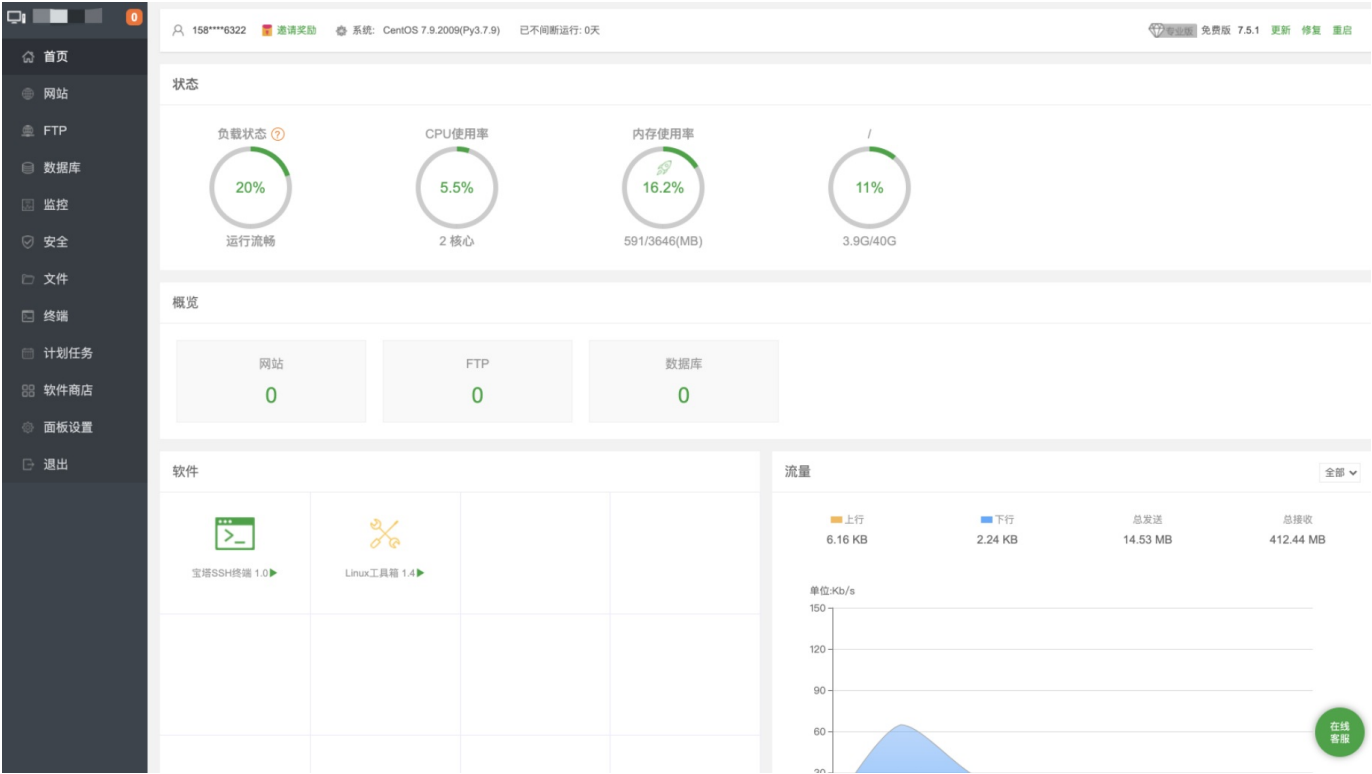
登录

未有账号，免费注册

一个账号可以绑定多台宝塔面板

7. 完整首页，看到这一步就说明可以了

3.1安装宝塔



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，注意到云服务器安全组规则给端口开放访问权限

The screenshot shows the 'Add Site' dialog box in the Baotao Linux Panel. The dialog is titled '添加站点-支持批量建站' (Add Site - Support Batch Site Creation). It has two tabs: '创建站点' (Create Site) and '批量创建' (Batch Create). The '创建站点' tab is active. The form contains the following fields:

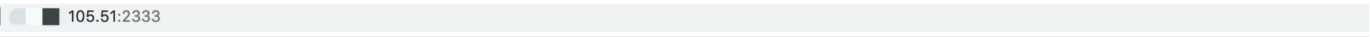
- 域名 (Domain): 105.51:2333
- 备注 (Remarks): _105_51
- 根目录 (Root Directory): /www/wwwroot/.105.51
- FTP: 不创建 (Do not create)
- 数据库 (Database): 不创建 (Do not create)
- PHP版本 (PHP Version): 纯静态 (Pure static)
- 网站分类 (Website Category): 默认分类 (Default category)

 At the bottom, there are '取消' (Cancel) and '提交' (Submit) buttons.

3.1安装宝塔



3. 访问设定的ip和端口，可以看到显示创建成功

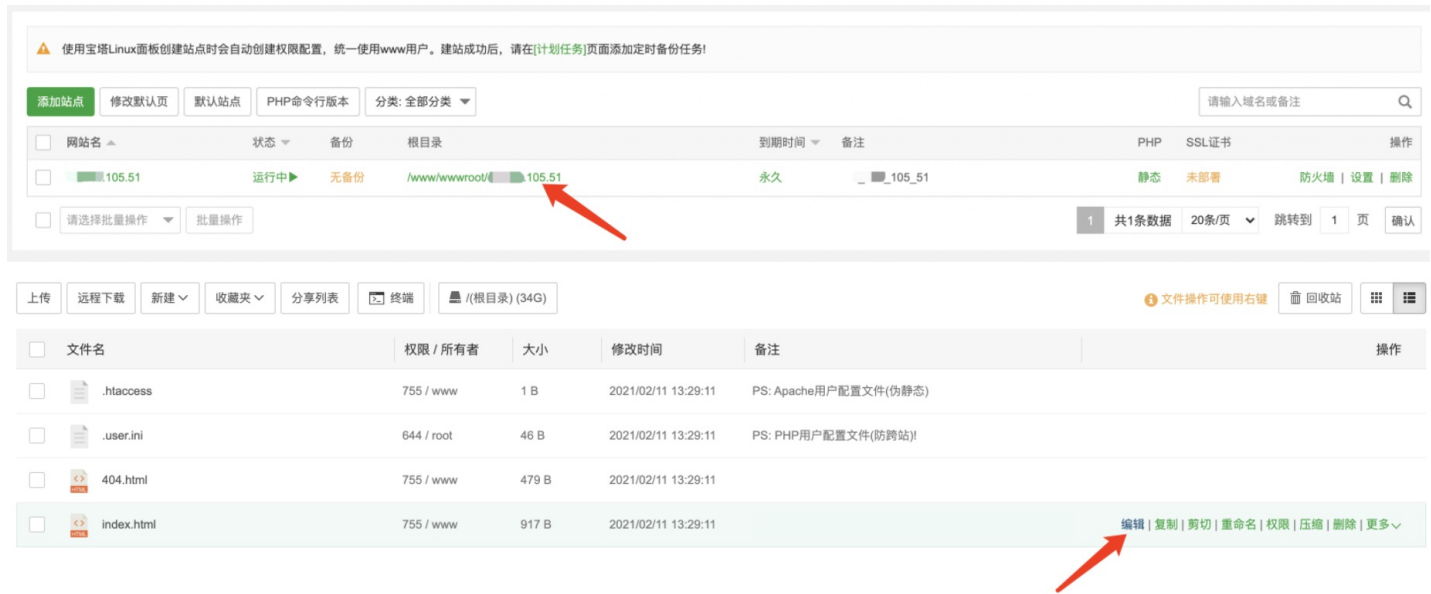


恭喜, 站点创建成功!

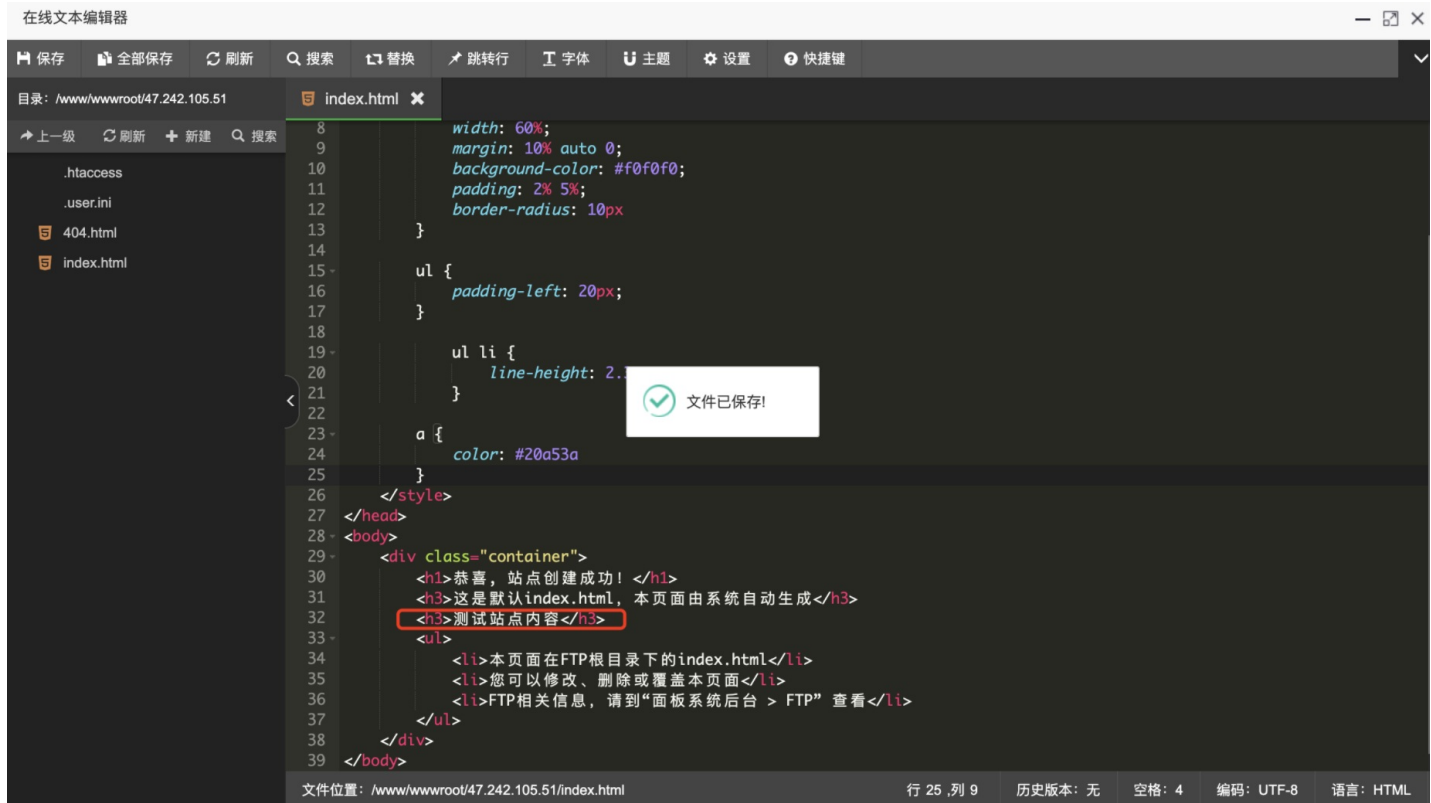
这是默认index.html，本页面由系统自动生成

- 本页面在FTP根目录下的index.html
- 您可以修改、删除或覆盖本页面
- FTP相关信息，请到“面板系统后台 > FTP” 查看

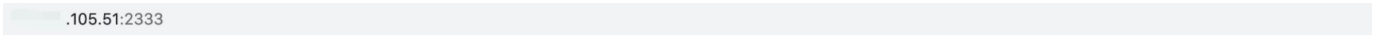
4. 点进根目录，找到index.html，修改一下内容



3.1安装宝塔



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



恭喜, 站点创建成功!

这是默认index.html, 本页面由系统自动生成

测试站点内容

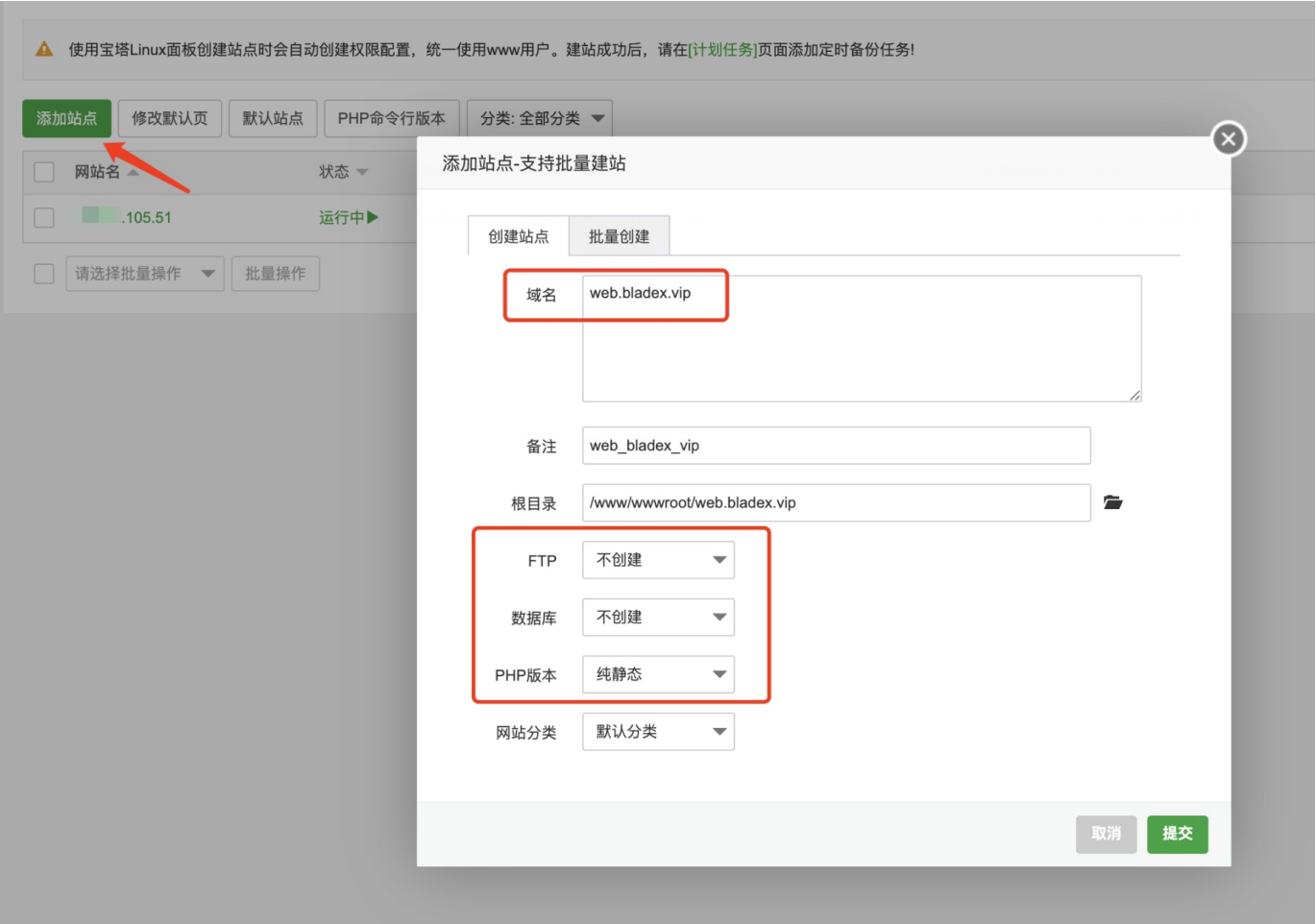
- 本页面在FTP根目录下的index.html
- 您可以修改、删除或覆盖本页面
- FTP相关信息, 请到“面板系统后台 > FTP” 查看

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



添加记录



记录类型：

A- 将域名指向一个IPV4地址

主机记录：

web

.bladex.vip

解析线路：

默认 - 必填！ 未匹配到智能解析线路时，返回【默认】线路设置结果

* 记录值：

.105.51

* TTL：

10 分钟

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

```
web.bladex.vip
```

恭喜, 站点创建成功!

这是默认index.html，本页面由系统自动生成

- 本页面在FTP根目录下的index.html
- 您可以修改、删除或覆盖本页面
- FTP相关信息，请到“面板系统后台 > FTP” 查看

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

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

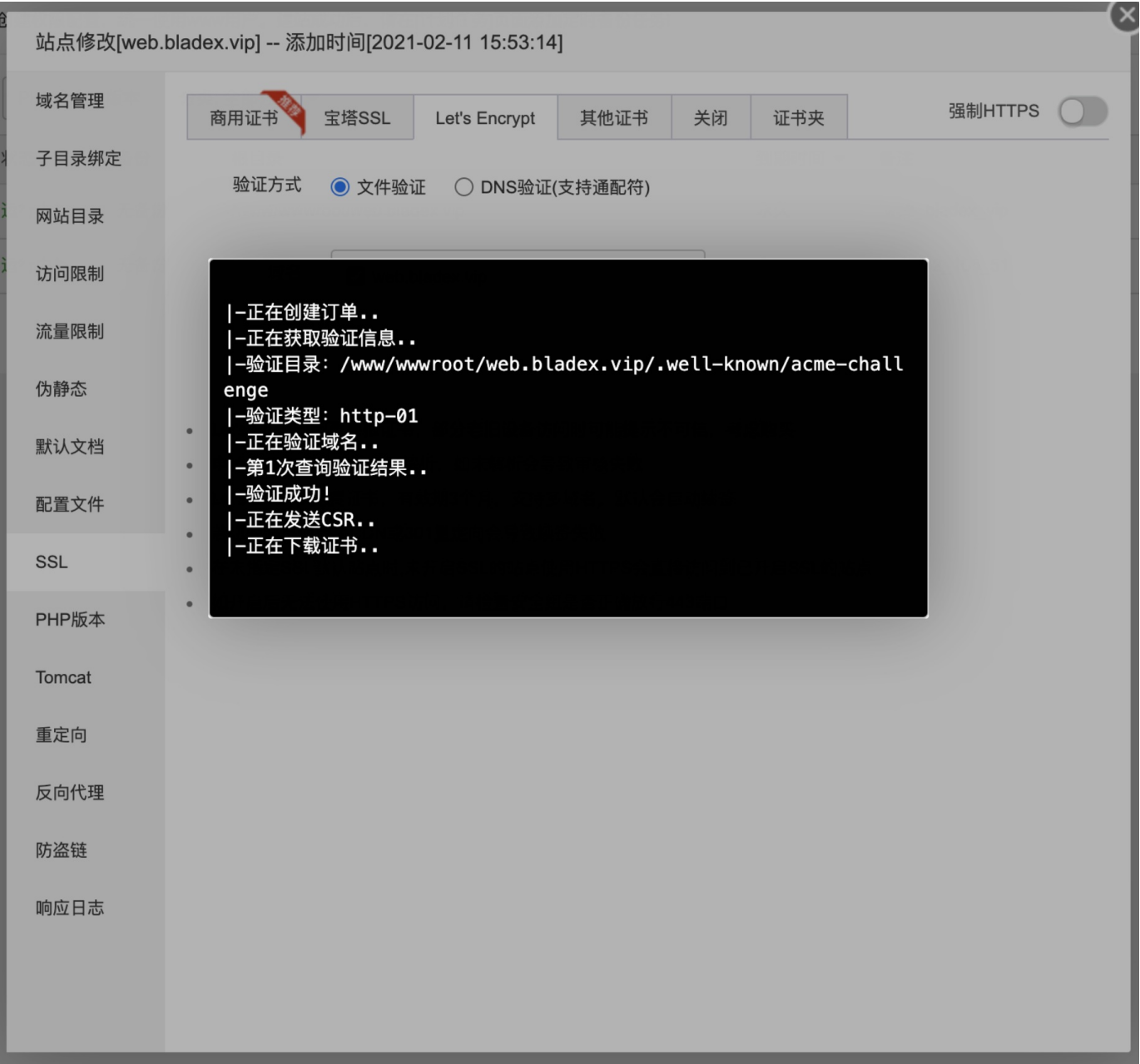
申请之前，请确保域名已解析，如未解析会导致审核失败

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

宝塔SSL

Let's Encrypt

其他证书

关闭

证书夹

强制HTTPS ☒

已部署成功: 将在距离到期时间1个月内尝试自动续签

证书品牌: Let's Encrypt

认证域名: web.bladex.vip

到期时间: 2021-05-12

密钥(KEY)

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgwggg
SkAgEAAoIBAQCg6V1lzkeVwkbI
QjKRoaE1u0tFw4t+MuVrSojDk3oMPs/iDA5jMgNTh
683PP4cg3ZzPGTNx0EZ3mM6
9Ar/sGurfazxiq2H4cONciaKQONWX4kvTCHVIULB
b++j0mjvN3iNnc/goZilx07E
edReSiEIoi6mVc0e/Odzspu5Uf+MiA4/ndu5iL7Wgd
V5RN0EAu4M2zY/s+WwVyoL
6zBMXjnKZyHkuBh39W7jKE4CbYLq9RwWLE9+JdJ
```

证书(PEM格式)

```
-----BEGIN CERTIFICATE-----
MIIFITCCBAmgAwIBAgISBLPT5aosBilioxCHXUHwt
an7MA0GCSqGSIb3DQEBEwUA
MDIxSzAJBgNVBAYTAIVTMRYwFAYDVQQKEw1M
ZXQncyBFbmNyeXB0MQswCQYDVQQD
EwJSMzAeFw0yMTAyMTEwNzAwMzNaFw0yMTA1
MTIwNzAwMzNaMBxkFzAVBgNVBAMT
DndlYi5ibGFkZXgudmlwMIIBIjANBgkqhkiG9w0BAQ
EFAAOCAQ8AMIIBCgKCAQEA
oOldSM5HlcJASEIykaGhNbtLRcOLfjLla0qlw5N6DD
```

关闭SSL

续签

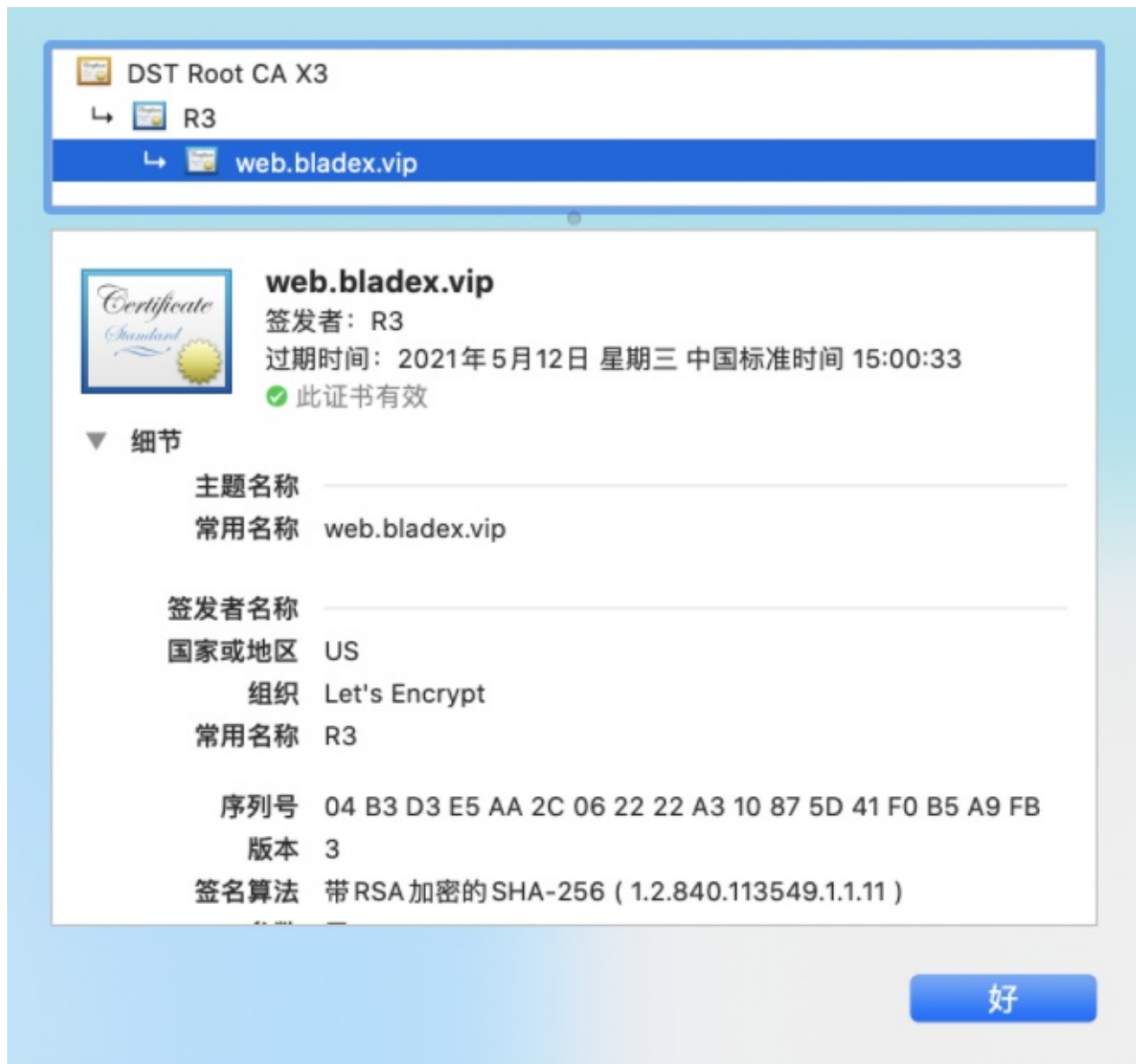
- 已为您自动生成Let's Encrypt免费证书;
- 如需使用其他SSL,请切换其他证书后粘贴您的KEY以及PEM内容, 然后保存即可。
- 如开启后无法使用HTTPS访问, 请检查安全组是否正确放行443端口

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

恭喜, 站点创建成功!

这是默认index.html, 本页面由系统自动生成

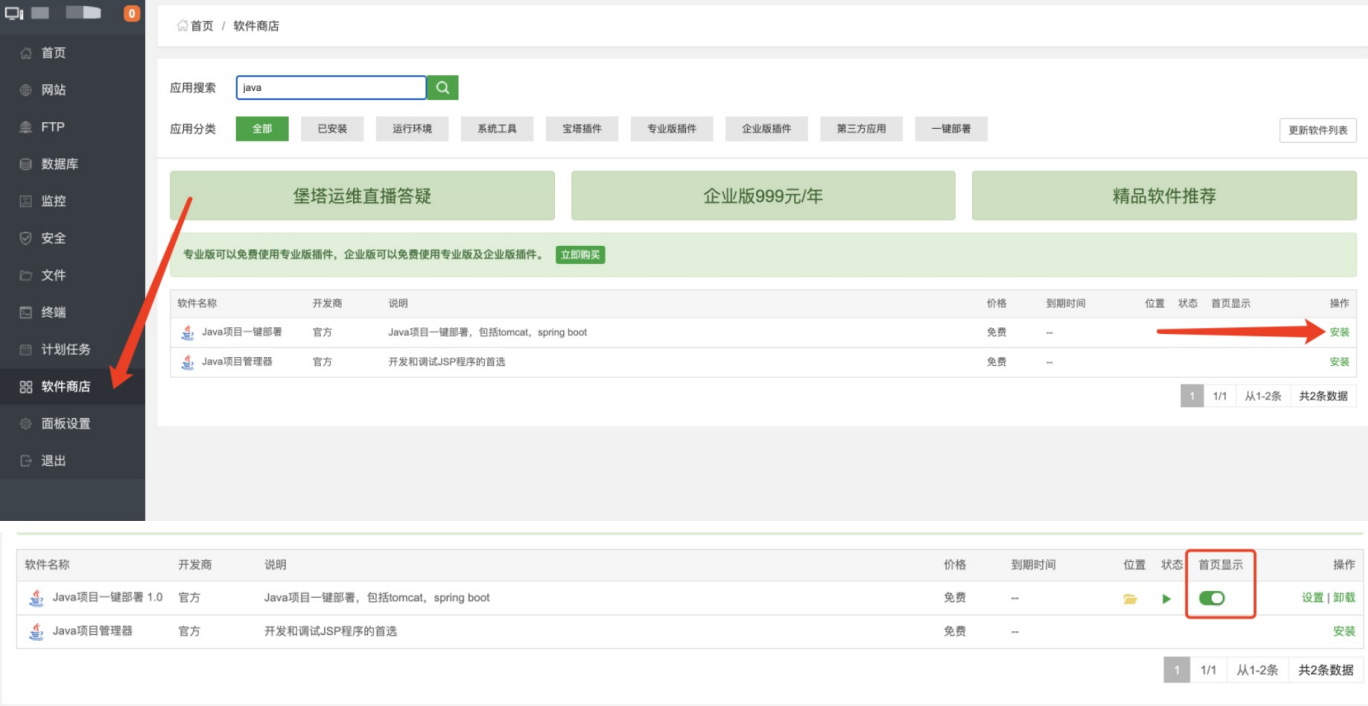
- 本页面在FTP根目录下的index.html
- 您可以修改、删除或覆盖本页面
- FTP相关信息, 请到“面板系统后台 > FTP”查看



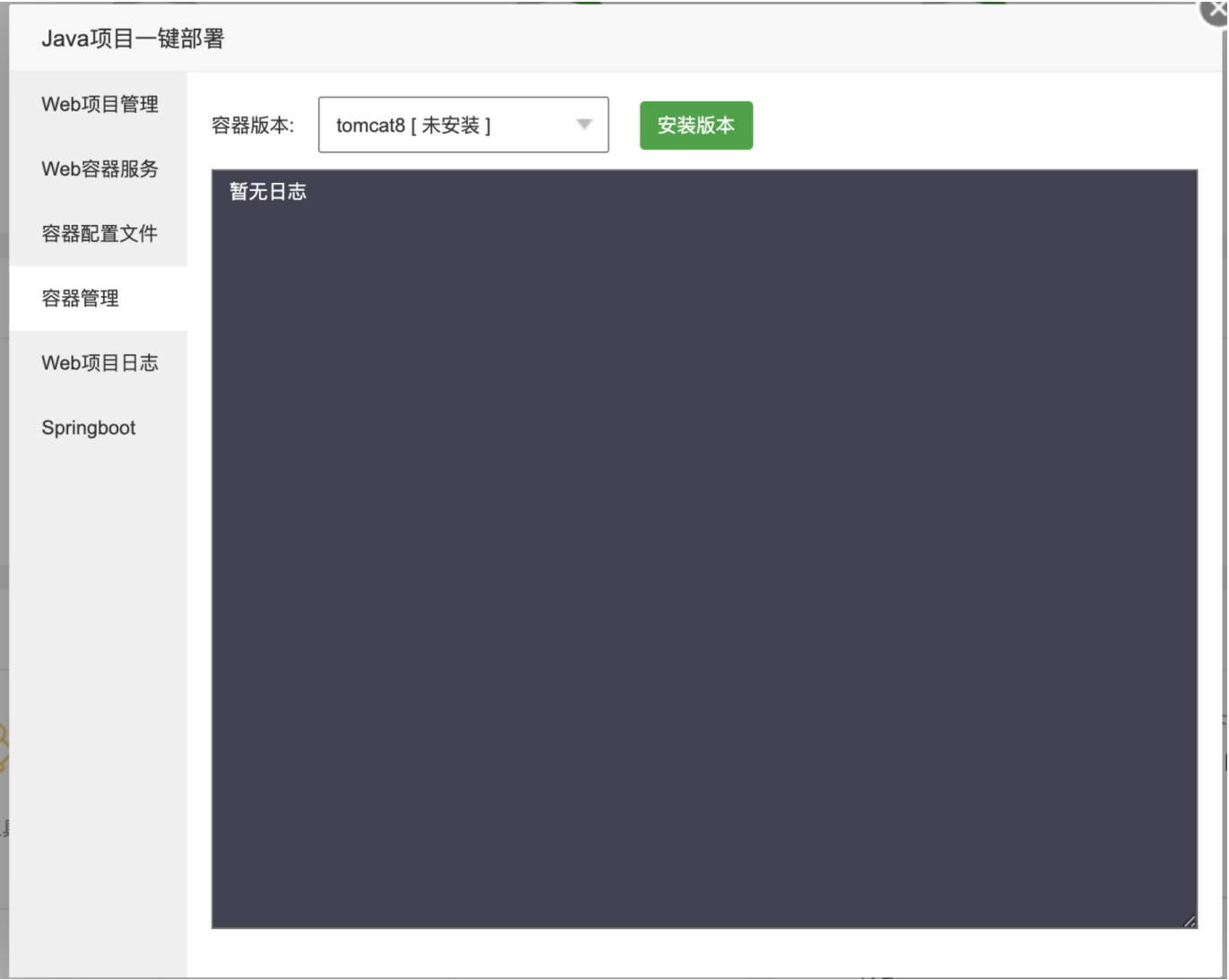
3.2基础部署

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





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，并且不开放至公网访问



Redis

服务

版本切换

性能调整

配置文件

负载状态

设置持久化

bind

127.0.0.1

绑定IP(修改绑定IP可能会存在安全隐患)

port

6379

绑定端口

timeout

0

空闲连接超时时间,0表示不断开

maxclients

10000

最大连接数

databases

16

数据库数量

requirepass

redis密码,留空代表没有设置密码

maxmemory

0

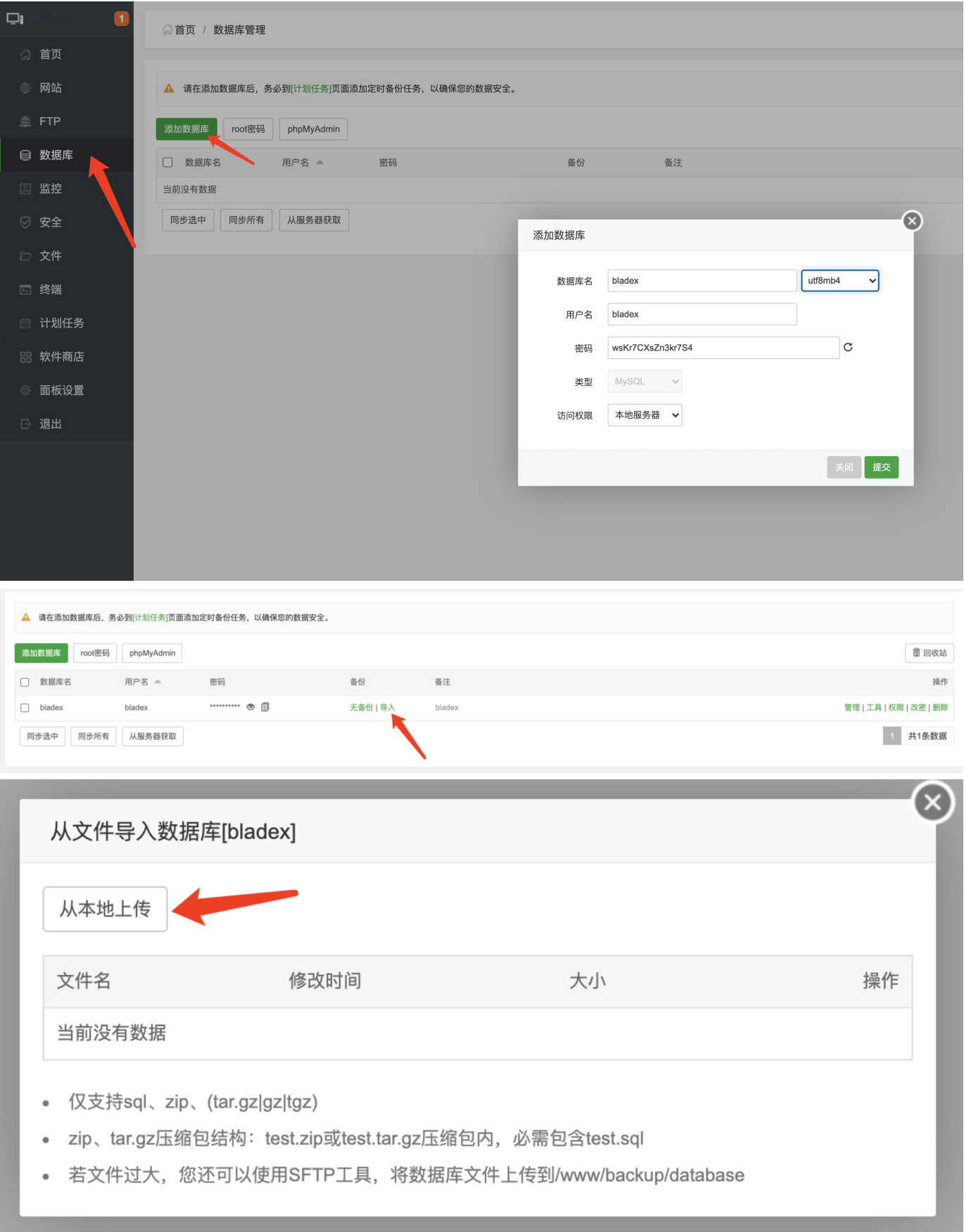
MB,最大使用内存, 0表示不限制

保存

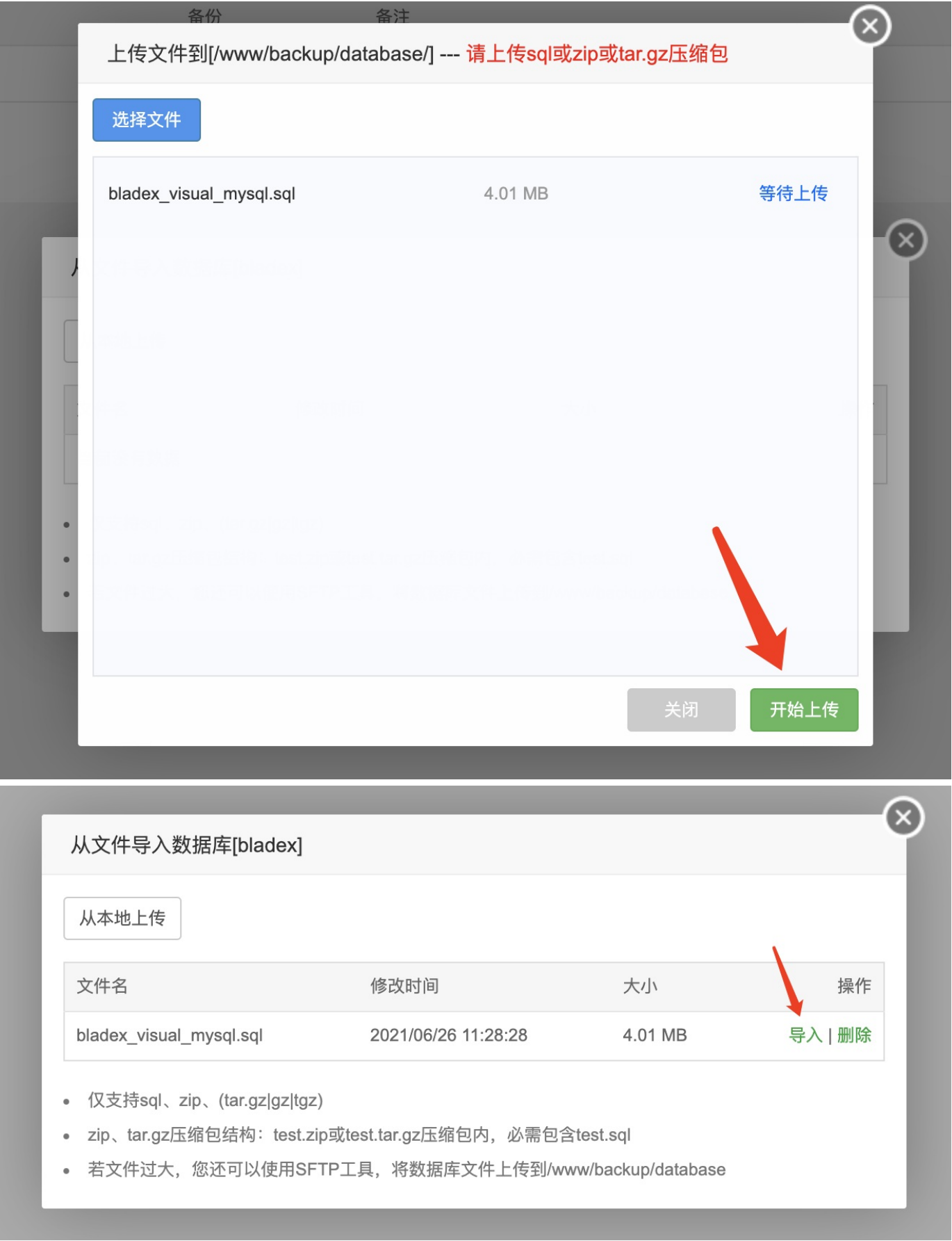
- 修改配置后需要重启redis生效,若您的数据需要持久化请先执行save操作。
- 警告: 请勿开启Redis外网访问权限, 未限制访问IP的情况下极大概率导致服务器被入侵。

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

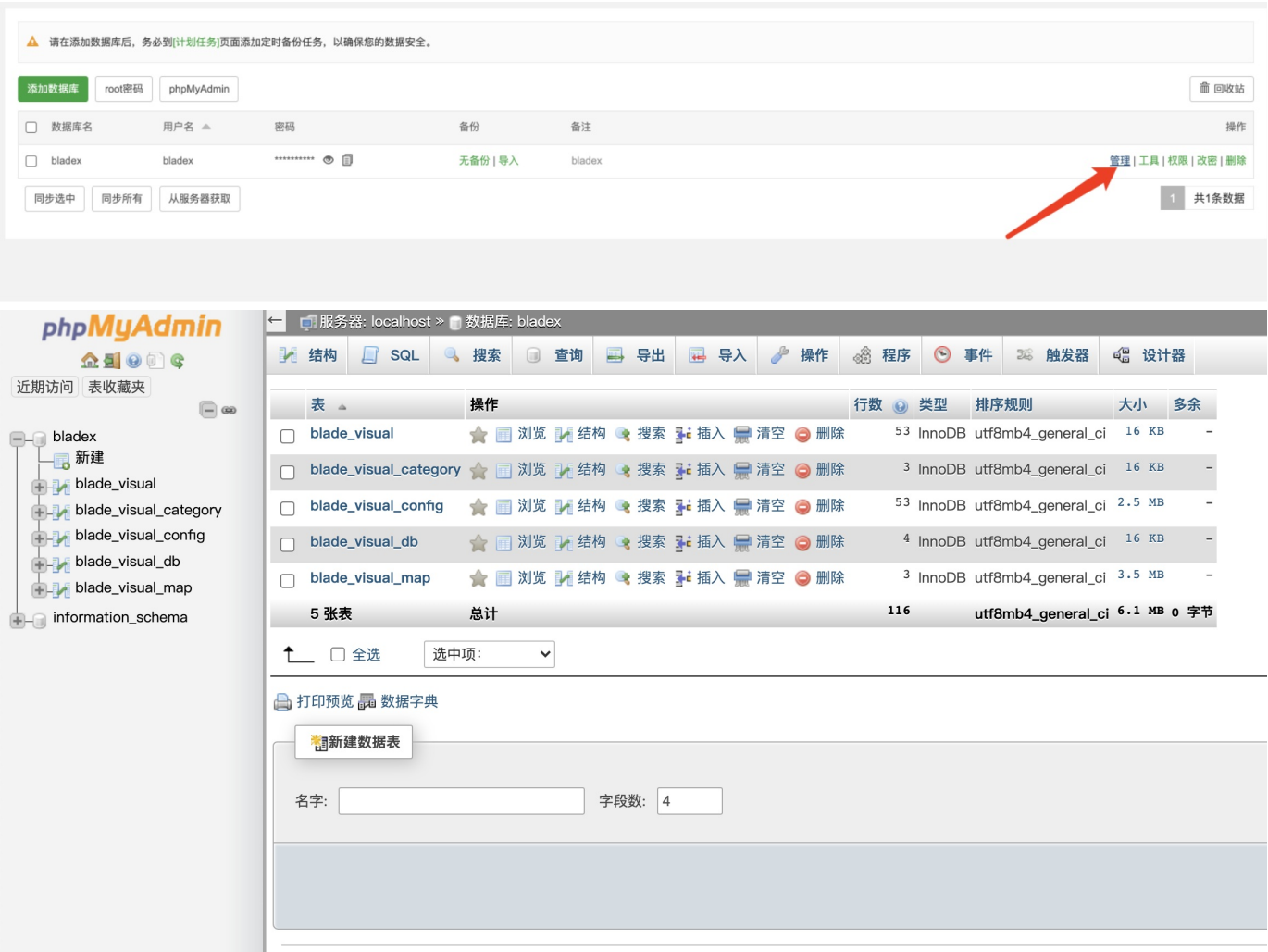








2. 点击管理，查看数据是否导入成功



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

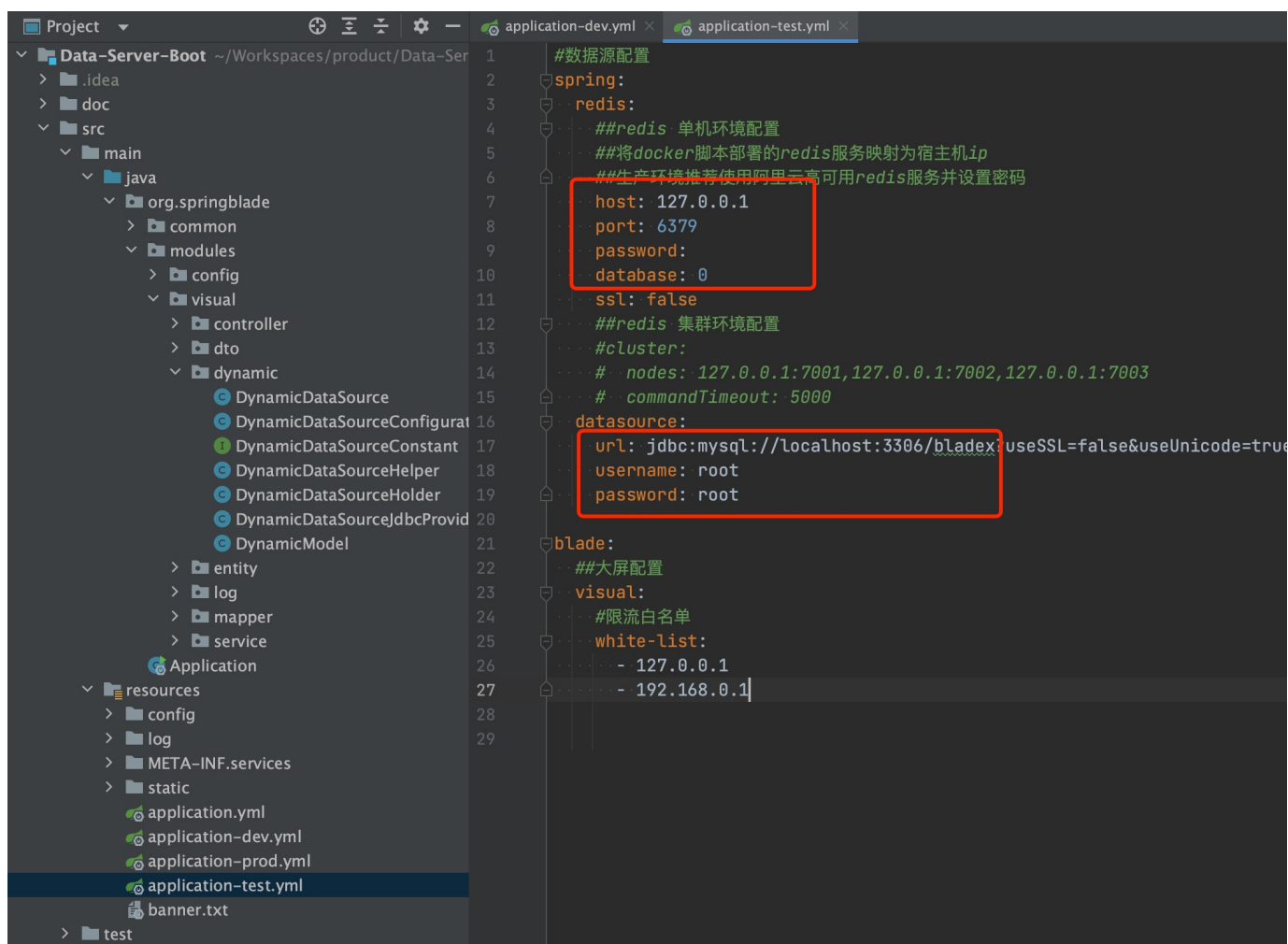


2. 站点访问成功



二、后端部署

1. 将部署好的配置放入application-test.yml



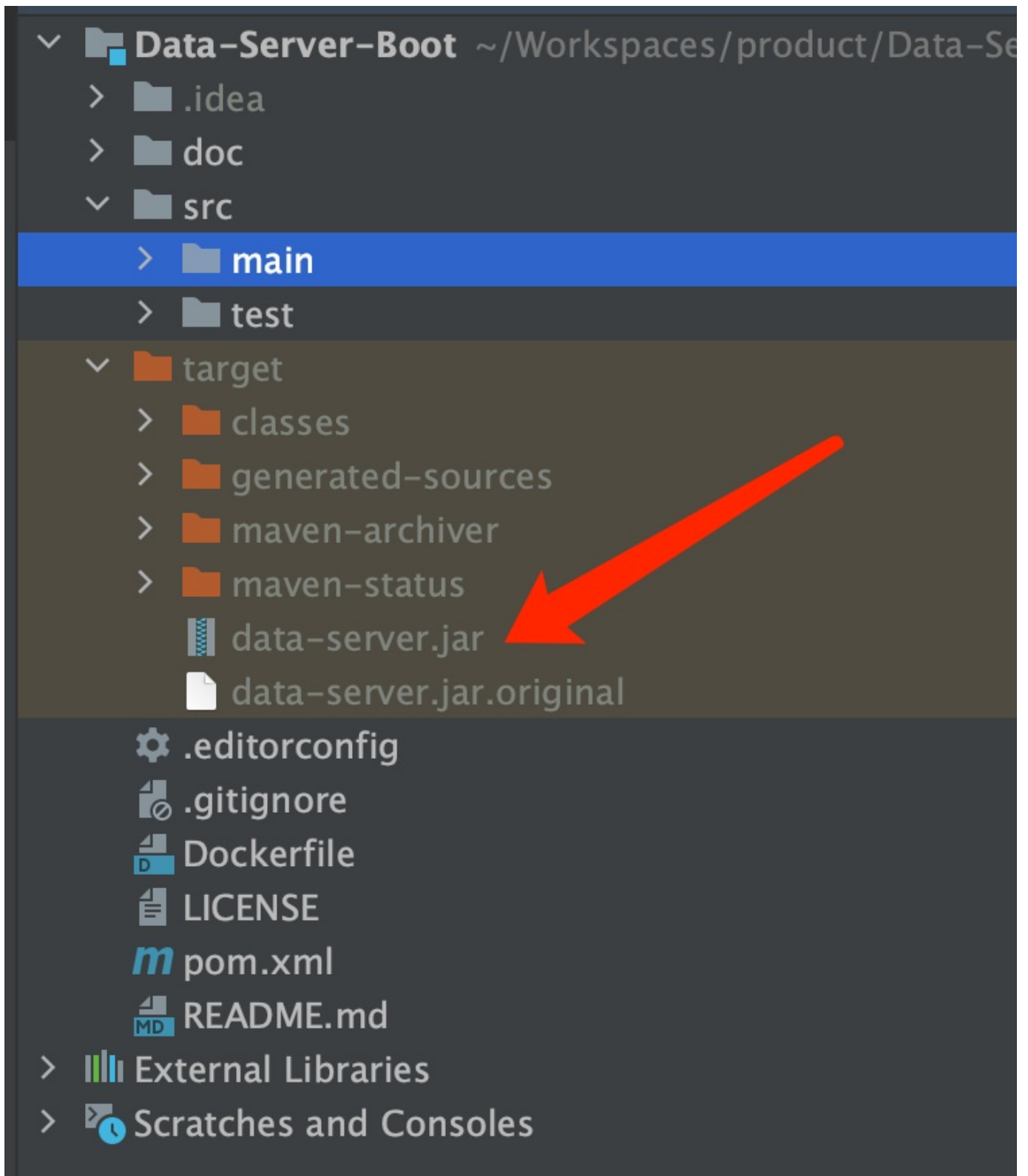
2. 根目录执行 `mvn clean package -Dmaven.test.skip=true`

3.3大屏部署

```
Welcome to fish, the friendly interactive shell
❯ ~/p/Data-Server-Boot on master ❯ mvn clean package -Dmaven.test.skip=true
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.springblade:Data-Server-Boot:jar:3.2.0
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-compiler-plugin is missing. @ line 221, column 21
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO] -----< org.springblade:Data-Server-Boot >-----
[INFO] Building Data-Server-Boot 3.2.0
[INFO] -----[ jar ]-----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ Data-Server-Boot ---
[INFO] Deleting /Users/chill/Workspaces/product/Data-Server-Boot/target
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ Data-Server-Boot ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 11 resources
[INFO] Copying 8 resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ Data-Server-Boot ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 71 source files to /Users/chill/Workspaces/product/Data-Server-Boot/target/classes
[WARNING] /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/dynamic/DynamicDataSourceHolder.java: /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/dynamic/DynamicDataSourceHolder.java 使用或覆盖了已过时的 API。
[WARNING] /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/dynamic/DynamicDataSourceHolder.java: 有关详细信息，请使用 -Xlint:deprecation 重新编译。
[WARNING] /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/log/VisualLogListener.java: /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/log/VisualLogListener.java 使用了未经检查或不安全的操作。
[WARNING] /Users/chill/Workspaces/product/Data-Server-Boot/src/main/java/org/springblade/modules/visual/log/VisualLogListener.java: 有关详细信息，请使用 -Xlint:unchecked 重新编译。
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Data-Server-Boot ---
[INFO] Not copying test resources
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ Data-Server-Boot ---
[INFO] Not compiling test sources
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ Data-Server-Boot ---
[INFO] Tests are skipped.
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ Data-Server-Boot ---
[INFO] Building jar: /Users/chill/Workspaces/product/Data-Server-Boot/target/data-server.jar
[INFO] --- spring-boot-maven-plugin:2.7.15:repackage (default) @ Data-Server-Boot ---
[INFO] Replacing main artifact with repackaged archive
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.220 s
[INFO] Finished at: 2023-09-13T22:36:26+08:00
[INFO] -----
❯ ~/p/Data-Server-Boot on master ❯
```

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



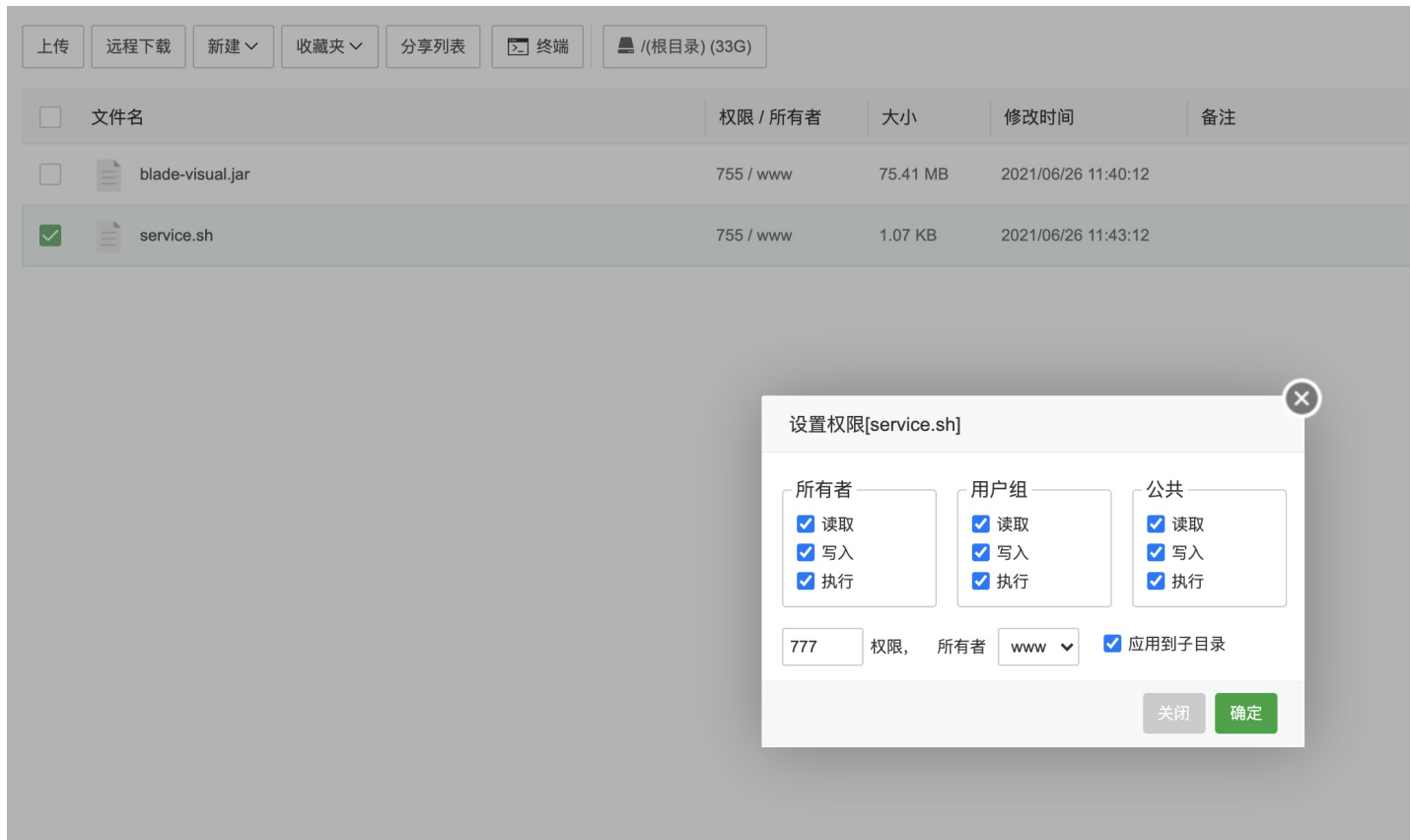
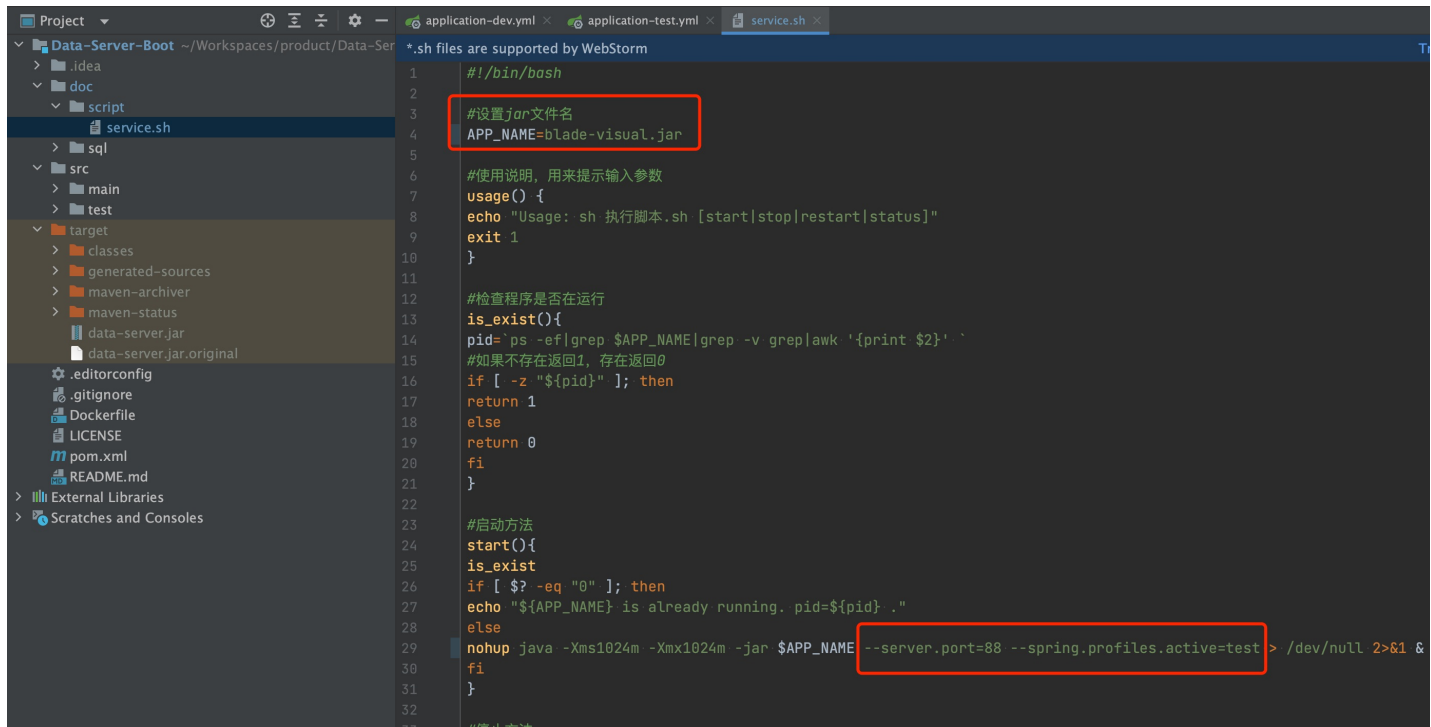
5. 将blade-visual.jar上传至api文件夹



6. 将启动脚本稍作修改也上传至api文件夹并配置好权限

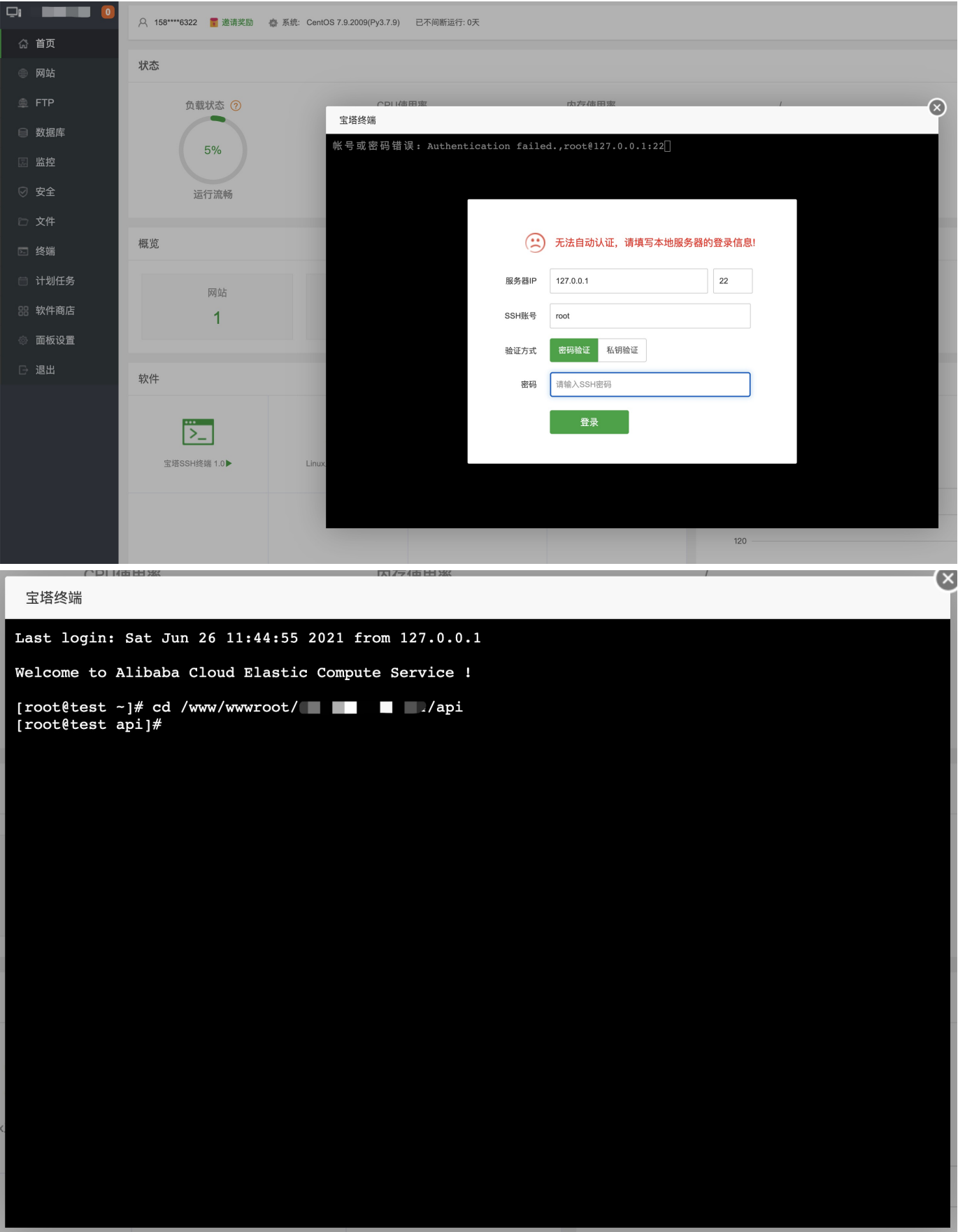
默认端口为8050，若我们需要改成其他端口，比如88，给启动命令加上 `--server.port=88`

为了生效 `application-test.yml`，给启动命令加上 `--spring.profiles.active=test`

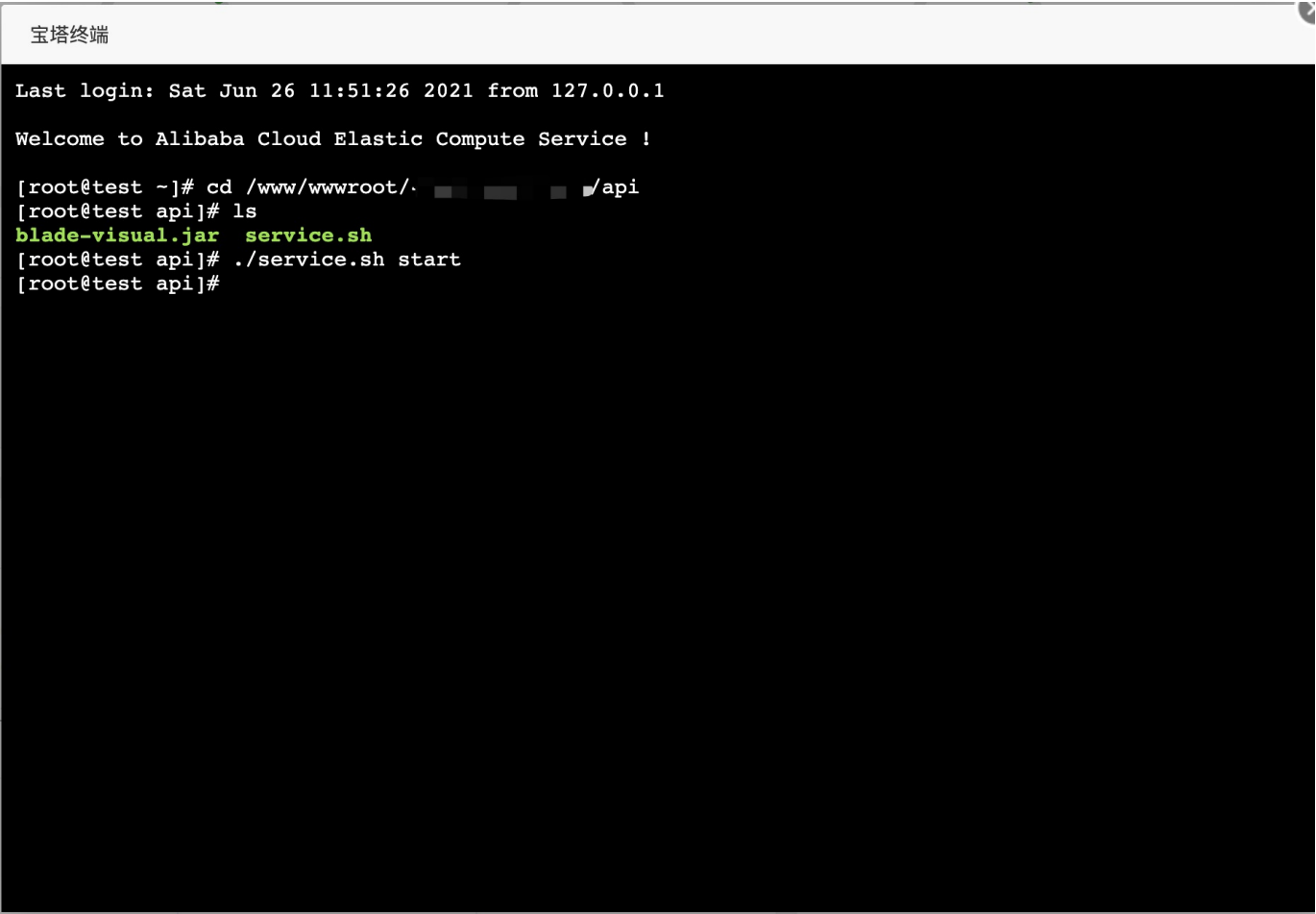


三、后端运行

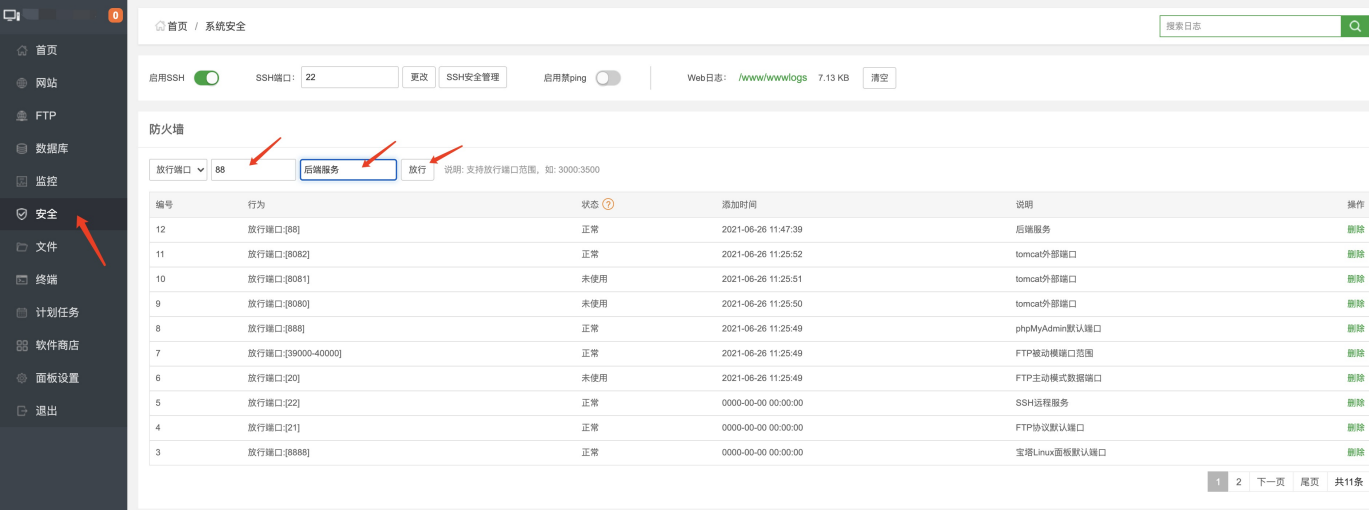
1. 前往首页点击ssh终端并输入服务器密码



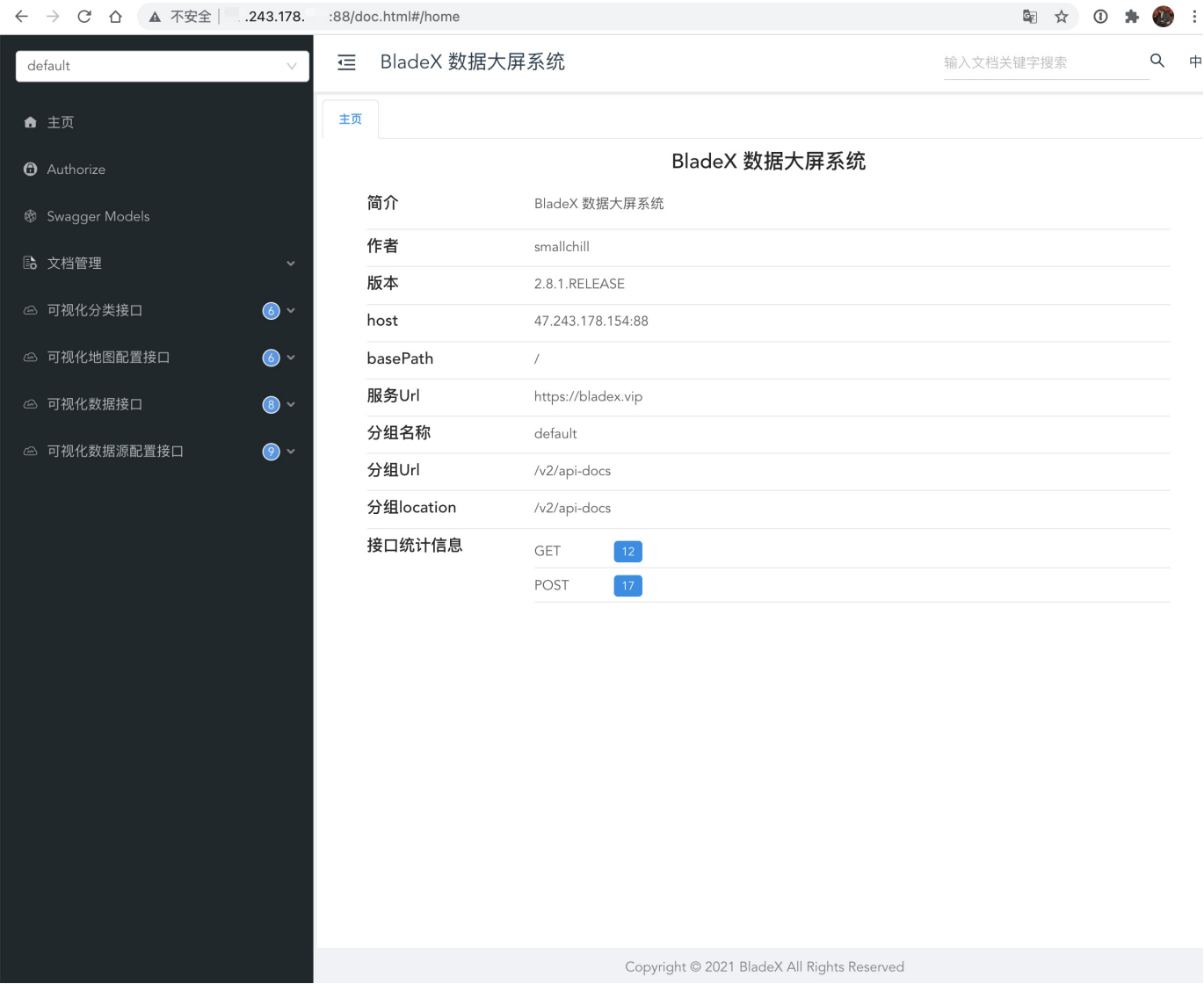
2. 进入目录运行脚本



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

```

1  window.$website = {
2      isDemo: false,
3      isDemoTip: '演示环境不允许操作',
4      title: 'BladeX 数据大屏',
5      name: 'BladeX 数据大屏',
6      subName: '可视化数据大屏 (演示环境-请勿放生产数据)',
7      url: 'http://服务地址:服务端口/blade-visual',
8      autoSave: false,
9      autoSaveTime: 60000,
10     tabsList: [0, 1, 2, 3, 4, 5, 6],
11     componentsList: [{
12         name: 'test',
13         component: 'testComponents',
14         option: 'testOption',
15         data: true
16     }],

```

2. 根目录执行 yarn run build

```

Terminal: Local x +
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

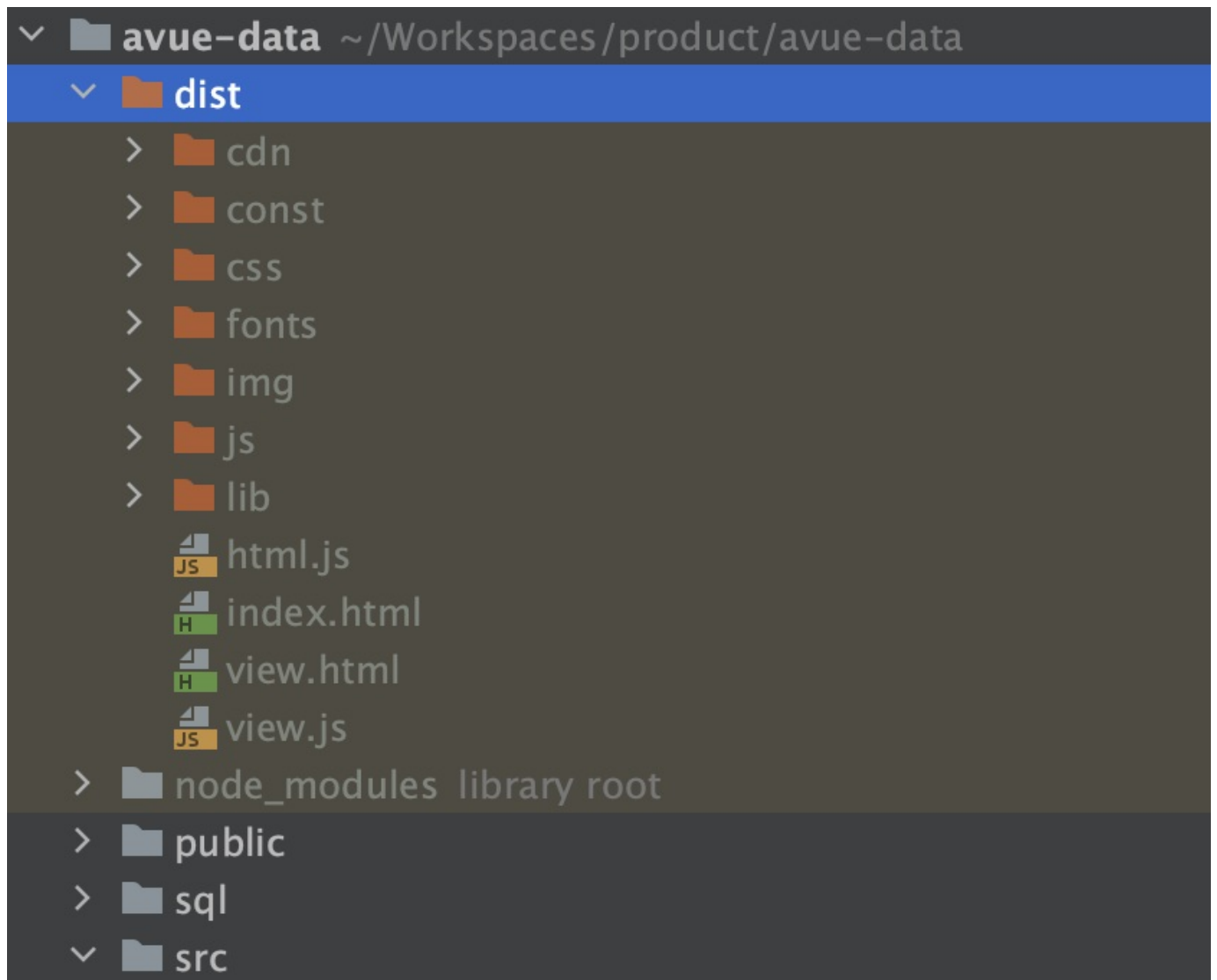
Images and other types of assets omitted.

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 / 所有者
☐	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

3.3大屏部署

☐	文件名	权限 / 所有者	大小	修改时间	备注
☐	api	755 / www	计算	2021/06/26 11:52:09	
☐	cdn	755 / www	计算	2021/06/26 11:55:44	
☐	const	755 / www	计算	2021/06/26 11:55:44	
☐	css	755 / www	计算	2021/06/26 11:55:44	
☐	fonts	755 / www	计算	2021/06/26 11:55:44	
☐	img	755 / www	计算	2021/06/26 11:55:44	
☐	js	755 / www	✓ 已将解压任务添加到消息队列!		
☐	lib	755 / www			
☐	.htaccess	755 / www	1 B	2021/06/26 11:33:36	PS: Apache用户配置文件(伪静态)
☐	.user.ini	644 / root	47 B	2021/06/26 11:33:36	PS: PHP用户配置文件(防跨站)!
☐	404.html	755 / www	479 B	2021/06/26 11:33:36	
☐	caster.zip	755 / www	17.45 MB	2021/06/26 11:56:36	
☐	html.js	755 / www	2.57 KB	2021/06/26 11:55:44	
☐	index.html	755 / www	2.53 KB	2021/06/26 11:55:44	
☐	view.html	755 / www	2.25 KB	2021/06/26 11:55:44	

共8个目录，8个文件，文件大小: 计算

5. 返回站点列表，进行nginx伪静态配置，否则会出现404

⚠ 使用宝塔Linux面板创建站点时会自动创建权限配置，统一使用www用户。建站成功后，请在计划任务页面添加定时备份任务!

添加站点

修改默认页

默认站点

PHP命令行版本

分类: 全部分类

请输入域名或备注

☐	网站名	状态	备份	根目录	到期时间	备注	PHP	SSL证书	操作
☐		运行中	无备份	/www/wwwroot/	永久		静态	未部署	防火墙 设置 删除

☐ 请选择批量操作

批量操作

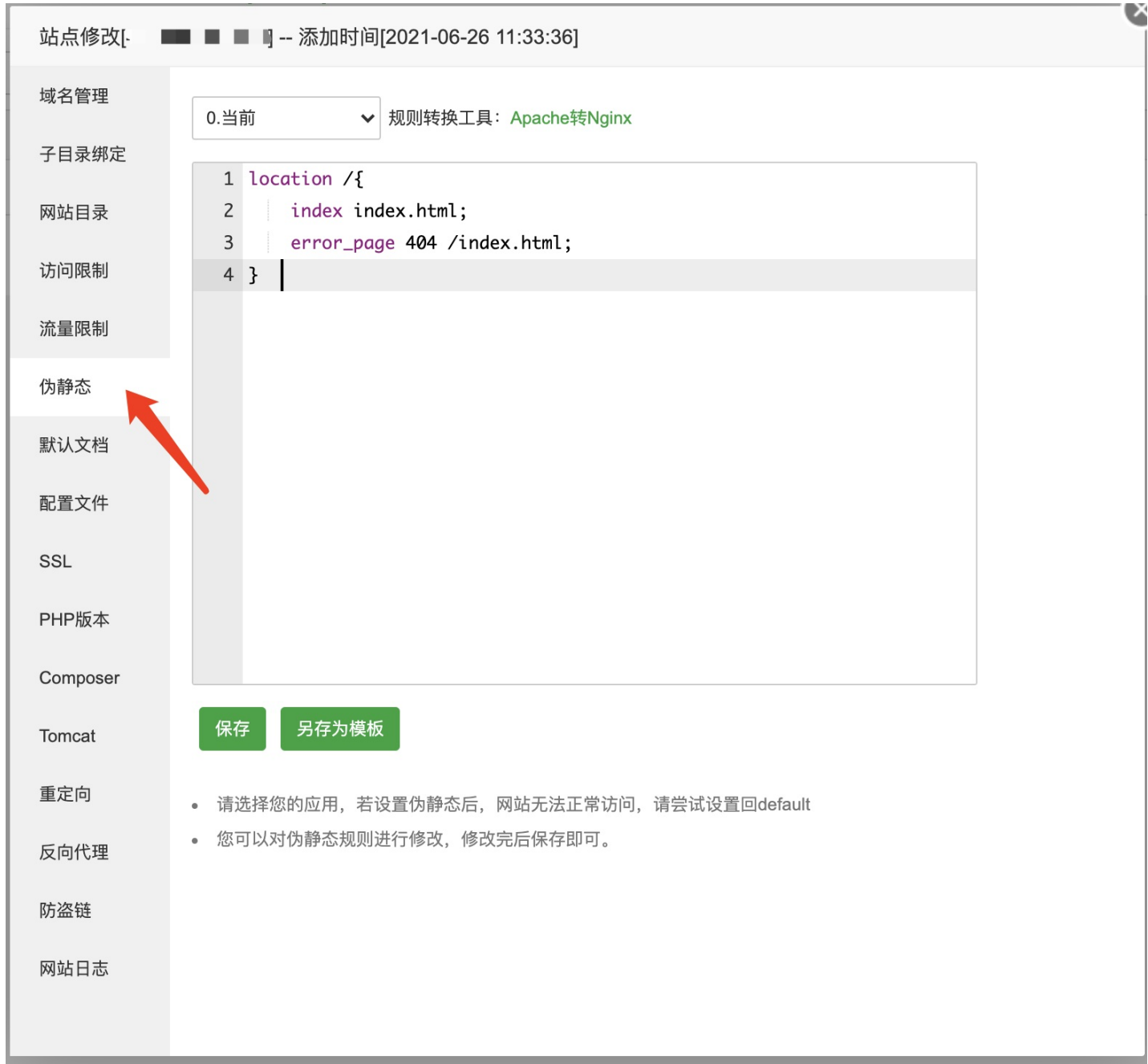
1 共1条 20条/页

跳转到 1 页

确认

6. 加入配置并保存

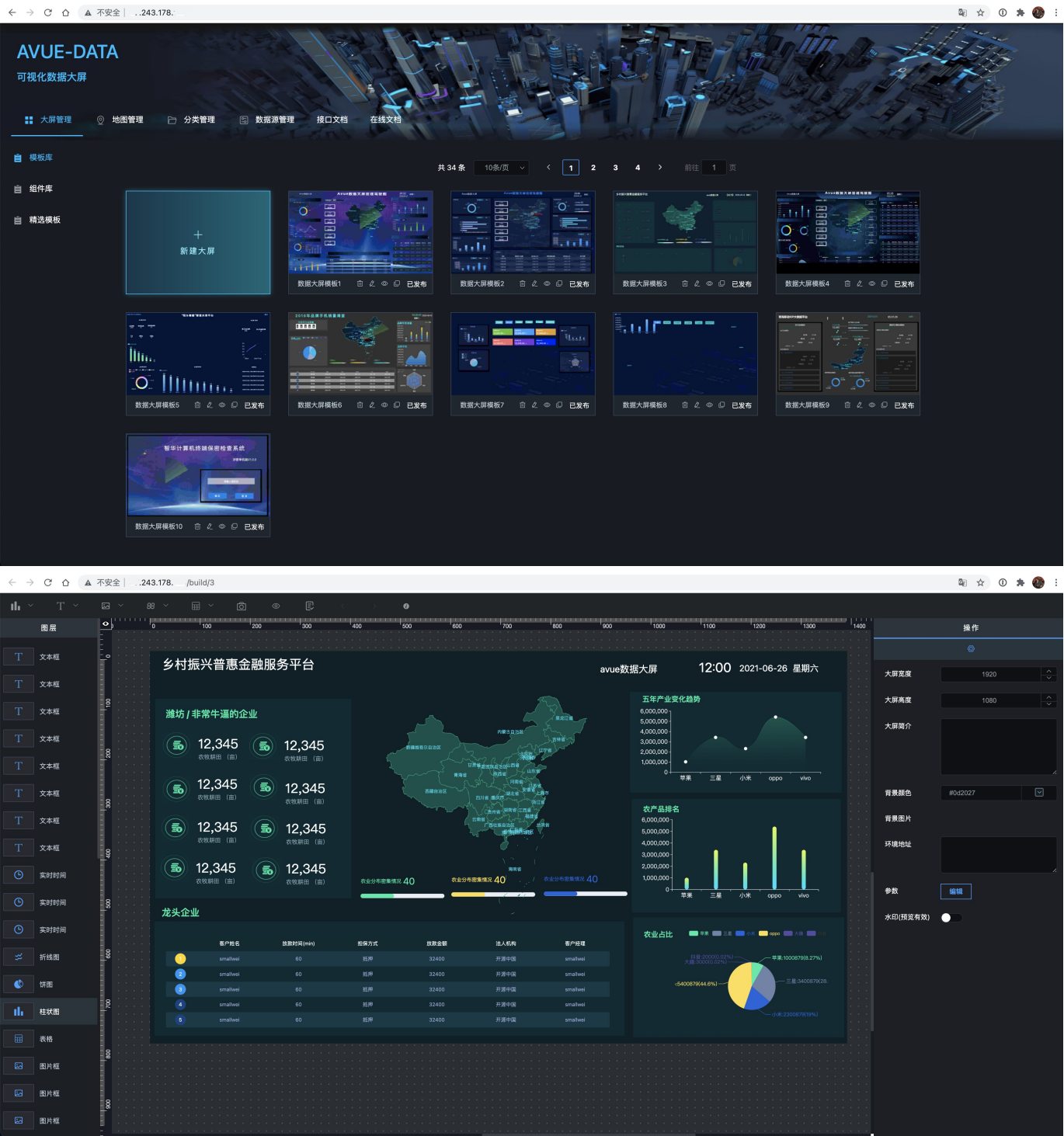
```
location /{
    index index.html;
    error_page 404 /index.html;
}
```



五、前端运行

1. 刷新站点网站，可以看到系统已经部署成功

3.3大屏部署



2. 打开f12，可以看到接口地址也正确无误

3.3大屏部署

