# MapGo-1.3.1 部署文档

## 一、资源

资源包下载地址：\\192.168.0.180\软件\GIS\MapGo -> MapGo1.3.1.zip

注：

* 该地址为公司内部window共享资源，打开方式请参考 [window系统怎么访问其他电脑共享文件夹](http://www.win7zhijia.cn/jiaocheng/win7_25204.html)
* 打开共享文件夹时需要公司域账户登录

## 二、基础环境要求

* 操作系统：window 7 或者 window 10
* Node版本：11+（长期支持版本）

## 三、部署步骤

### 1. 下载资源包

资源包中主要包含两个程序：

* Nginx 程序文件

 ├── nginx-1.18.0 // Nginx
 │ ├── conf // Nginx 配置文件夹
 │ │ ├── nginx.conf // Nginx 主要配置文件（端口、代理等）
 │ │ └── ...
 │ ├── html // 默认资源文件夹（即默认ip:port/ 指向的路径）
 │ │ ├── visualmap // 平台代码文件夹
 │ │ └── ...
 │ └── ...

* MapGo 程序文件

 ├── visualmap // 地图可视化配置平台程序
 │ ├── apps // 用户应用配置文件夹
 │ │ ├── user // 用户名文件，保存该用户的应用配置
 │ │ │ ├── 2D // 二维应用配置文件夹
 │ │ │ │ ├── applicationName // 应用名称文件夹，保存某个应用的具体配置
 │ │ │ │ │ ├── config.json // 地图配置文件（底图、微件等配置）
 │ │ │ │ │ ├── interactions.js // 开发者代码文件
 │ │ │ │ │ ├── LayerManager // 图层列表配置文件夹
 │ │ │ │ │ │ └── config.json // 具体图层配置
 │ │ │ │ │ ├── LegendManager // 图例配置文件夹
 │ │ │ │ │ │ └── config.json // 图例具体配置
 │ │ │ │ │ └── TimeLineManager // 时间轴配置文件夹
 │ │ │ │ │ └── config.json // 时间轴具体配置
 │ │ │ │ └── ...
 │ │ │ └── ...
 │ │ └── ...
 │ ├── gis\_framework // 二维地图框架
 │ ├── icons // 图标库
 │ ├── public // 模板库
 │ │ ├── echarts\_template // Echarts 模板保存位置
 │ │ │ ├── images // 模板展示效果图片库
 │ │ │ │ └── ... // 模板展示效果图片（以模板id为文件名称）
 │ │ │ └── template.json // 模板具体配置
 │ │ ├── html\_template // HTML 模板保存位置
 │ │ │ ├── template\_id // 某个模板所能使用的图片
 │ │ │ │ └── ... // 具体图片
 │ │ │ └── template.json // 模板具体配置
 │ ├── services // 平台后台代码
 │ │ ├── mapapi.js // node 主要代码
 │ │ └── ...
 │ ├── config.js // 平台配置文件（设置后台端口、公告等）
 │ └── ...

注： 具体文件目录请看附录一

### 2. Node部署

注：如果没有 Node 环境，请先根据要求安装 Node。

* 在 services 目录下启动 cmd , 运行 npm install 安装依赖。
* 打开 mapapi.js , 设置 Node 和 WebSocket 的端口
* 在文档中寻找如下代码并修改
* // node服务的端口，修改时可只改此全局变量
* const port = 3007;
* // websocket服务的端口
* const wsport = 3006

注： 默认为 3007 和 3006 ，如果有端口占用，请选择其他端口

* 设置第三方接口（主要是为了统一平台登录账户）

注：

* 默认选择了平台组最新开发的用户管理平台的接口，一般情况下只需把 IP 和 port 修改为项目本地的即可
	+ 关于用户管理平台接口详情请看 services -> readme.md 和直接咨询公司平台组
	+ 如果不打算使用第三方接口，请保证登录接口和获取所有用户接口不会访问第三方接口，以防出现问题，修改接口请参考 [附录二](#_附录二)

请在文档代码中查找以下代码并修改

// 最新平台统一登录接口

const userManagerURL = 'http://192.168.0.140:7070/ou'; //用户管理平台地址

自己设置第三方接口的步骤请查看 [附录二](#_附录二)。

* 运行并测试

运行：在 services 目录下，启动 cmd ，运行命令 node mapapi.js

测试接口：http://localhost:3007/mapapi/icons

成功结果：

注：如果您在上步中修改了 Node 的接口，请把 3007 修改为对应的端口号

失败的解决办法：根据 cmd 中的提示信息，查找后解决

### 3. Nginx 部署

注：

1. 资源包中默认提供了 Nginx 文件，并且默认配置了相关代理
2. 如果您不想使用提供的 Nginx , 您把下方三个代理和 visualmap 文件夹正确的配置和部署到您的 Nginx 中即可
3. 以下代理均为默认 Node 和 WebSocket 默认端口，有需要修改为对应的即可
4. 因为 Nginx 和 Node 是在同一台服务器上，所以可以使用 localhost，如果不是，请改为具体 IP

Nginx 端口设置

在 nginx-1.18.0 -> conf -> nginx.conf 文件中找到 listen 配置项

# http -> server -> listen

server {

   # Nginx 端口设置，默认为10229,端口占用修改即可

   listen       10229;

   server\_name  localhost;

   charset utf-8;

   server\_tokens off;

   #charset koi8-r;

   # ...

}

#### Node 服务代理

# 代理 MapGo 平台后端 Node 接口

location /mapapi/ {

   proxy\_pass http://localhost:3007/mapapi/; # 3007 为 mapapi.js 默认后台端口

   #proxy\_connect\_timeout 300s;

   # 获取IP代理路径

   proxy\_set\_header Host $http\_host;

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

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

   proxy\_set\_header X-Forwarded-Proto $scheme;

}

#### WebScokt 代理

# 代理 MapGo 后端 WebSocket 接口

location /wss {

   proxy\_pass http://localhost:3006/ws; #3006 为 mapapi.js 监听的 WebSocket 端口

   #proxy\_connect\_timeout 300s;

   proxy\_http\_version 1.1;

   proxy\_set\_header Upgrade \$http\_upgrade;

   proxy\_set\_header Connection "upgrade";

   proxy\_set\_header Host \$host;

}

#### 地图代理

# MapGo 后端提供的框架请求代理，应用于 OGC 图层请求

location /visualmap/proxy {

   proxy\_pass http://localhost:3007/mapapi/proxy/;

   add\_header 'Access-Control-Allow-Origin' '\*'; #允许来自所有的访问地址

   add\_header 'Access-Control-Allow-Credentials' 'true';

   add\_header 'Access-Control-Allow-Methods' '\*'; #允许所有的方法

   add\_header 'Access-Control-Allow-Headers' 'Content-Type,\*';

}

#### 运行

在 nginx-1.18.0 目录下，启动 cmd，运行命令 start nginx

#### 测试

在浏览器地址栏中访问 <http://localhost:10229/mapapi/icons>

返回结果和 <http://localhost:3007/mapapi/icons> 一致即可

注：端口有修改请自行改为相应的即可

### 4. 平台配置

将 visualmap 下的 config.js 的 HOST、nodeHOST、webSocketHOST、proxyURL 这四个配置修改为 Nginx 代理之后的地址

const appConfig = {

   HOST: '121.46.19.47:10229', // 项目部署的 ip + port

   nodeHOST: '121.46.19.47:10229', // node服务的 ip + port

   webSocketHOST: '121.46.19.47:10229', // webSocket服务的 ip + port

   enableDownload: false, // 是否启用下载功能

   proxyURL: 'http://121.46.19.47:10229/visualmap/proxy', // GIS框架的代理地址

   isUpdateProxyURL: true, // 是否每次都更新配置文件中的代理地址

   enableVerificationCode: false, // 是否启用验证码

   isGISFrameBuild: true, // GIS框架是否打包

   // 全局替换接口中的某些字段（一般是迁移的时候，地址有统一变动时用，注意：应用只有再次保存的时候，地址才会真的被修改）

   // 替换范围：1.框架配置文件（代理地址不参与修改）2.图层 3.图例 4.开发者代码

   enableURLReplaceStr: {

      enable: false, // 是否启用

      replaceStr: '', // 要替换的字符串

      replacedStr: '', // 替换后的字符串

   },

   helpDocLink: 'http://121.46.19.2:20724/MapGo\_doc/', // 可视化平台帮助文档地址

   noticeOption: { // 公告配置

      nable: false,

      notices: [

         {

            content: '公告示例',

            startTime: '2021-01-05 11:00', // 公告开始提示时间

            endTime: '2021-01-08 18:30', // 公告停止提示时间

         }

      ],

   },

};

### 5.平台测试运行

确保 Nginx 和 Node 正常运行，访问 [http://localhost:10229/visualmap/#/Login](http://localhost:10229/visualmap/%22%20%5Cl%20%22/Login)

输入正确的账号密码能进应用列表界面即可





## 四、注意事项

* 平台升级时，采用的是文件替换的方式，所以您的自定义代码需要自行替换调试

## 五、意见反馈

如果您有任何关于平台部署的疑问，请直接联系思路 GIS 组，我们将会研究解决您的问题并及时更新部署文档来方便您的使用，感谢您的支持，谢谢！

## 附录

### 附录一

#### 配置平台包

#### 地图框架包

#### 图层样式模板

#### 图标库

#### Node 服务

#### 其他

完整资源目录



### 附录二

设置第三方接口的步骤如下：

#### 接口准备

* 第三方登录接口
* 第三方获取所有用户的接口

#### 接口返回要求

{

    status: 1, // 1:验证成功 2: 验证失败 3：其他

    loginInfo: { // 必须存在

        loginName: 'admin', // 用户ID（失败或者其他时为空即可）

        nickName: '管理员', // 昵称（失败或者其他时为空即可）

    },

    msg: '登录成功' // 描述信息（登录成功，密码错误等）

}

* 获取所有用户接口返回要求
* [
* {
* USER\_NAME: 员工名称（王小红）
* LOGIN\_NAME：员工用户名（wangxh）
* }
* ]

注：这里说的要求是指 Node 向前端返回时要符合的格式。

#### 修改登录接口

在文档中搜索 /mapapi/login ,可跳转到登录接口代码处，内容基本如下：

/\*\*

 \* 登录接口

 \*/

app.post('/mapapi/login', (req, res) => {

  let {

    loginName,

    password

  } = req.body;

  password = decode(password);

  // 修改返回内容的编码为utf-8

  res.writeHead(200, {

    'Content-Type': 'text/json;charset=utf-8'

  });

  // 编写自己的代码

});

自己请求第三方接口并按**要求**返回相关数据

#### 修改获取所有用户接口

在文档中搜索 /mapapi/login ,可跳转到登录接口代码处，内容基本如下：

/\*\*

 \* 获取所有用户

 \*/

app.get('/mapapi/users', (req, res) => {

  const rootPath = path.join(root, 'apps');

  console.log(rootPath);

  res.writeHead(200, {

    'Content-Type': 'text/json;charset=utf-8'

  });

  // 编写自己的代码

});

自己请求第三方接口并按**要求**返回相关数据

在 Node 中可以目前支持两种 HTTP 请求的方式：

* Node 自带的 [http.request 模块](http://nodejs.cn/api/http.html%22%20%5Cl%20%22http_http_request_options_callback)
* [axios](https://www.cnblogs.com/JasmineLily/p/11157701.html)

注：两种请求在代码中都有实现