**Grafana使用说明**

# 服务启动

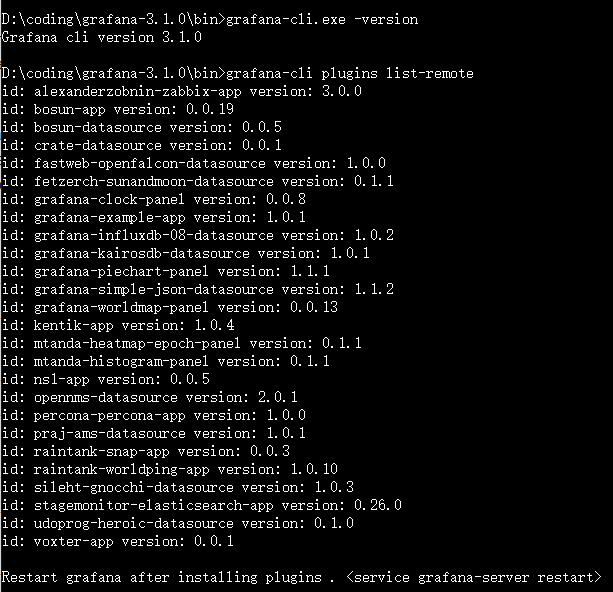
下载：<http://grafana.org/>

安装：<http://docs.grafana.org/installation/windows/>



# 插件管理

<http://docs.grafana.org/plugins/installation/>

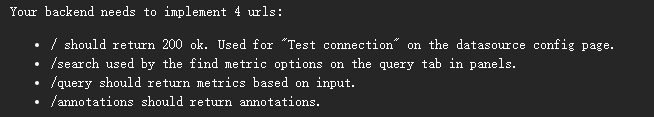


## 安装实现grafana-simple-json-datasource

<https://grafana.net/plugins/grafana-simple-json-datasource>

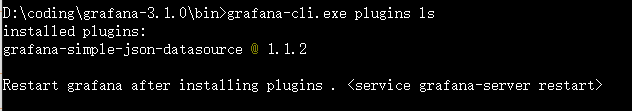
grafana-cli plugins install grafana-simple-json-datasource

介绍了数据源安装必须条件



https://grafana.net/plugins/grafana-simple-json-datasource/installation





### 调用 / 的Request Header(添加数据源时触发)

GET /rest/ HTTP/1.1

Host: 127.0.0.1:8080

Connection: keep-alive

Accept: application/json, text/plain, \*/\*

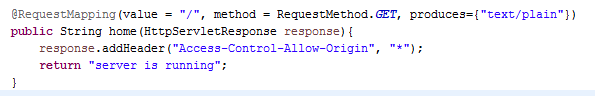
Origin: http://127.0.0.1:8000

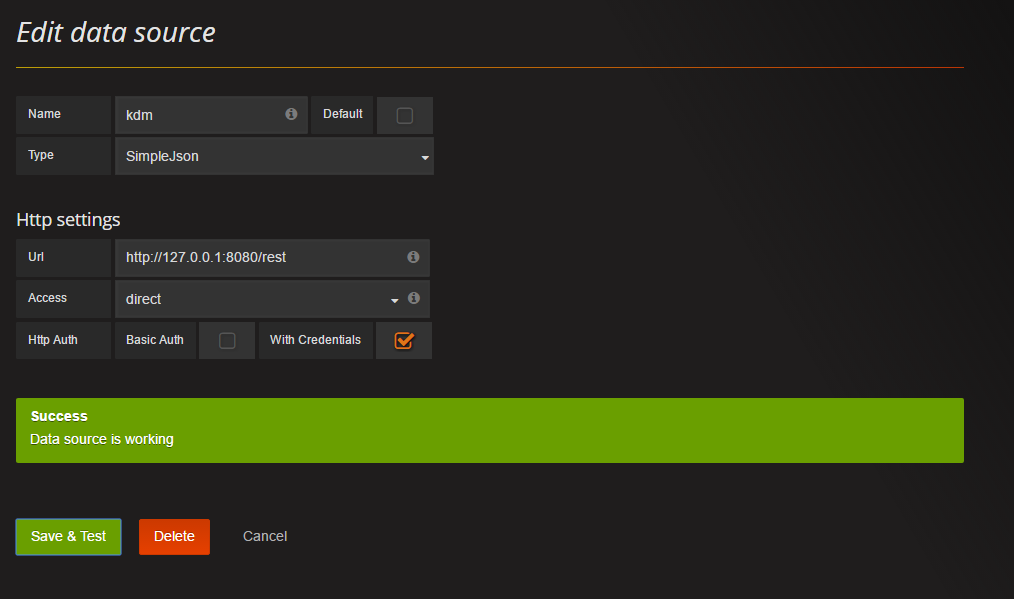
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36

Referer: http://127.0.0.1:8000/datasources/edit/2

Accept-Encoding: gzip, deflate, sdch

Accept-Language: zh-CN,zh;q=0.8





**在本地使用spring boot搭建了一个rest server**

### Panel 发送的query Request Header(刷新panel或验证query时触发)

OPTIONS /rest/query HTTP/1.1

Host: 127.0.0.1:8080

Connection: keep-alive

Access-Control-Request-Method: POST

Origin: http://127.0.0.1:8000

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36

Access-Control-Request-Headers: accept, content-type

Accept: \*/\*

Referer: http://127.0.0.1:8000/dashboard/db/my\_dashboard?panelId=1&fullscreen&edit

Accept-Encoding: gzip, deflate, sdch

Accept-Language: zh-CN,zh;q=0.8

为什么会发送OPTIONS请求？

**百度解释**：因为跨域，浏览器也会为了试探服务器是否会接受请求，先发送一个 options 请求。即便是服务器允许程序跨域访问，若不支持 options 请求，请求也会死掉

**自我理解：**实际上可以理解grafana页面实际发送的还是POST请求，若后台将这个接口设置成method = RequestMethod.POST，那么还是失败，因为POST失败；所以在加入一个过滤器，统一设置支持跨域，支持POST, GET, OPTIONS, DELETE方式请求

### Panel 发送的search Request Header（panel 添加query时触发）

POST /rest/search HTTP/1.1

Host: 127.0.0.1:8080

Connection: keep-alive

Content-Length: 38

Accept: application/json, text/plain, \*/\*

Origin: http://127.0.0.1:8000

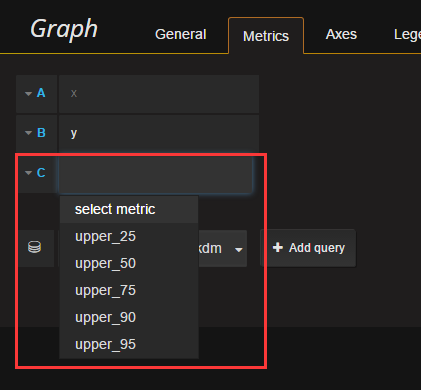
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36

Content-Type: application/json

Referer: http://127.0.0.1:8000/dashboard/db/my\_dashboard?panelId=1&fullscreen&edit&from=1469599986647&to=1469599987191&editorTab=Metrics

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.8



**调用接口后显示标签列表**

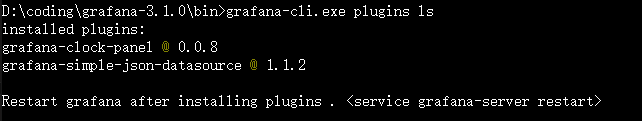
## 安装时钟插件

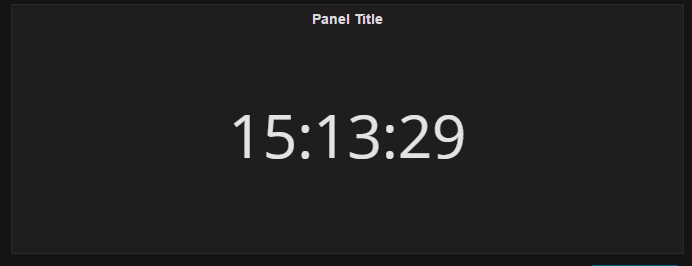
直接安装：<https://grafana.net/plugins/grafana-clock-panel/installation>

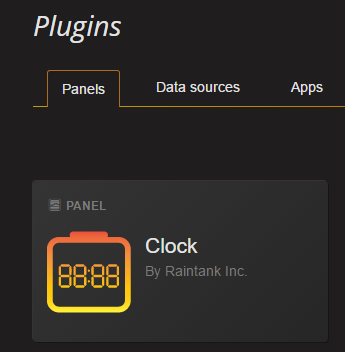
Git下载：<https://github.com/grafana/clock-panel.git>

复制grafana-clock-panel\dist整个文件下的内容至public\app\plugins\panel\xxx，注意文件夹名称xxx必须与plugin.json中id属性一致，重启服务

注意：源码存在错误，public\app\plugins\panel\grafana-clock-panel\clock\_ctrl.js，第112行，对应Options选项的页面对应有误







# 插件开发

## 环境准备

<http://www.cnblogs.com/sybboy/p/4831635.html>

### nodeJs环境

<https://nodejs.org/en/>

<https://nodejs.org/dist/latest-v4.x/docs/api/>

### npm

<https://docs.npmjs.com/>

NPM的全称是Node Package Manager，是随同NodeJS一起安装的包管理和分发工具，它很方便让JavaScript开发者下载、安装、上传以及管理已经安装的包。

npm是一个node包管理和分发工具，已经成为了非官方的发布node模块（包）的标准。有了npm，可以很快的找到特定服务要使用的包，进行下载、安装以及管理已经安装的包。

<http://www.tuicool.com/articles/VNZ7Vb>

npm config get prefix //获取当前设置的目录

npm install xxx -g 时， 模块将被下载安装到【全局目录】中。

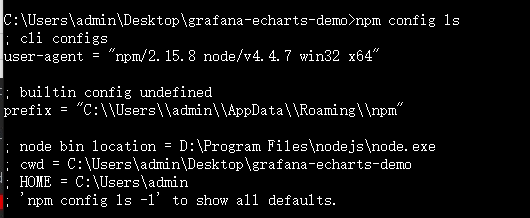
【全局目录】通过 npm config set prefix "目录路径" 来设置，通过 npm config get prefix 来获取当前设置的目录

npm install xxx ，则是将模块下载到当前命令行所在目录，运行目录会增加一个node\_modules\xxx的文件夹

**npm set global=true** //设定安装模式，设定之后npm 不带-g参数也能全局安装，推荐

npm get global //可以查看当前使用的安装模式，默认false

npm set global=false 无效：使用npm config ls看一下global的设置及文件，如果是用户.npmrc文件，找到它删除后就可以了



npm的包安装分为本地安装（local）、全局安装（global）两种，从敲的命令行来看，差别只是有没有-g而已。格式如下：

npm install 包名 # 本地安装

npm install -g 包名 # 全局安装

具体区别：

1、安装模块包位置：

1）全局安装时，会把模块下载到“全局目录”中，【全局目录】通过 npm config set prefix "目录路径" 来设置。通过 npm config get prefix 来获取当前设置的目录。

2）本地安装是，会把模块下载到当前命令行所在目录，c:\123>npm install xxx 将会安装到 c:\123\node\_modules\xxx

这种方式显然是不好的，所以一般都会使用全局安装方式统一安装的一个目录中去,这样既方便管理、结构清晰还可以重复利用。

2、调用方式：

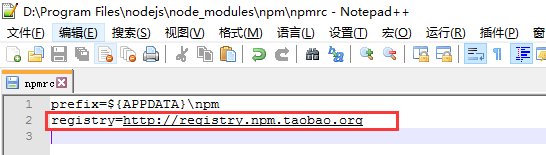
1）全局的安装是供命令行(command line)使用的，比如grunt，全局安装的方式是没有办法用require调用包的；

注：使用全局模式安装的包并不能直接在 JavaScript 文件中用 require 获得，因为 require 不会搜索 /usr/local/lib/node\_modules/。

2）本地安装可以直接通过require()的方式引入；var moduleName = require('moduleName')

注：可以通过使用npm set global=true/false来设定安装模式，npm get global可以查看当前使用的安装模式。

设置淘宝镜像地址，加速模块下载：



### grunt

JS自动化工具。对于需要反复重复的任务，例如压缩（minification）、编译、单元测试、linting等；Gruntfile.js为其配置文件

<https://www.npmjs.com/package/load-grunt-tasks>

<http://www.gruntjs.net/getting-started>

<http://www.cnblogs.com/wangfupeng1988/p/4561993.html>

grunt插件仓库：<http://www.gruntjs.net/plugins>

<http://caibaojian.com/grunt.html>

<http://www.cnblogs.com/chyingp/p/npm-install-difference-between-local-global.html>

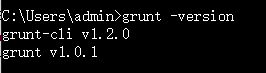
<http://blog.csdn.net/hellochenlu/article/details/49099655>

注意目录：

C:\Users\admin\AppData\Roaming\npm

C:\Users\admin\AppData\Roaming\npm-cache





### Grafana插件开发文档

<http://grafana.org/blog/2016/04/08/clock-plugin-p1.html>

## Gruntfile.js

Grunt打包配置文件

<http://www.gruntjs.net/sample-gruntfile>

## package.json

指定当前项目所需的模块

注：devDependencies里面的参数，指定了项目依赖的Grunt和Grunt插件版本。其中"~0.7.0"代表安装该插件的某个特定版本，如果只需安装最新版本，可以改成"\*"。

## module.js、module\_ctrl.js

module\_ctrl.js hold most of our plugin’s code. This ctrl file is imported into module.js and exported as ModuleCtrl

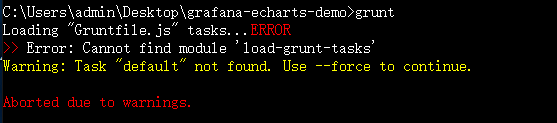
## plugin.json

Plugin json is the same concept as the package.json file for an npm package. Grafana looks for plugin.json files when scanning the plugins directory.

The important fields are the first three, especially the id. The convention for the id is [github username/org]-[plugin name]-[plugin type] and it has to be unique.

## 打包

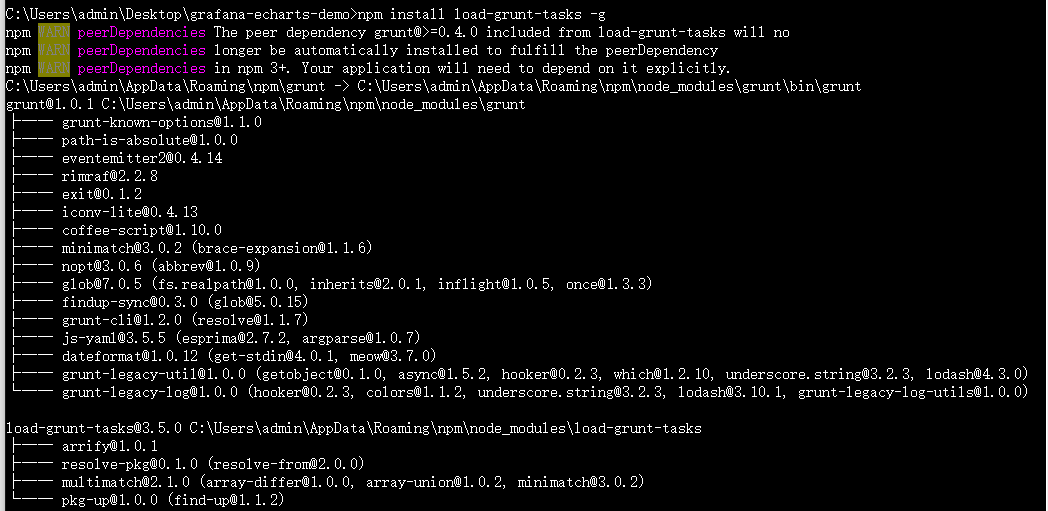
注意以下命令均在项目根目录下运行，根目录：C:\Users\admin\Desktop\grafana-echarts-demo



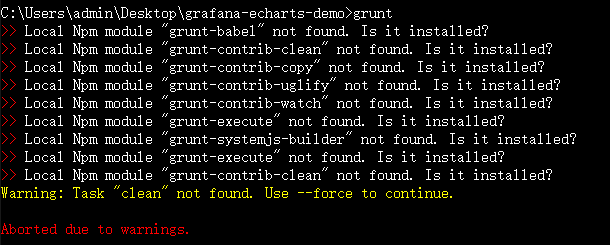
**插件未安装报错截图**

### npm install load-grunt-tasks

安装load-grunt-tasks插件，注意**不需要**全局安装



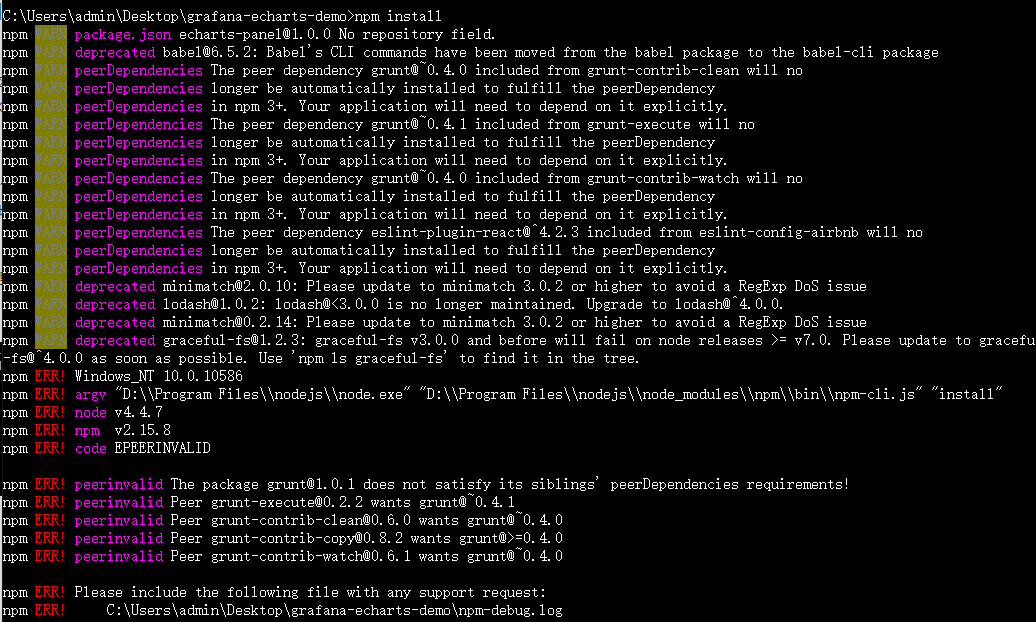
**安装load-grunt-tasks，注意图中是全局安装-g**



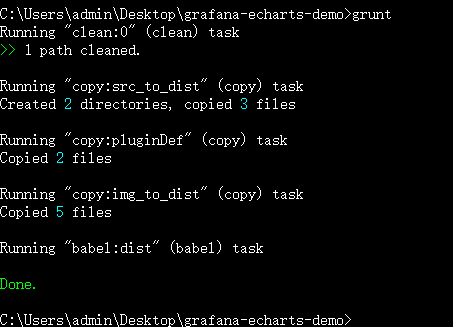
npm install grunt --save-dev //在当前目录安装grunt的同时，顺便把grunt保存为这个目录的开发依赖项

### npm install //批量安装插件

根目录下回创建一个node\_modules文件夹



### 打包成功，请查看dist目录



### 编码

Lodashjs：<http://lodashjs.com/docs/>

### 使用第三方js库

* **增加ECharts 3 图表库**

grafana-3.1.0\public\vendor目录新增echarts文件夹，将需要拷贝的js放入其中

技术关键字：ES6[ECMAScript 6]

<http://www.tuicool.com/articles/7BnM7bb>

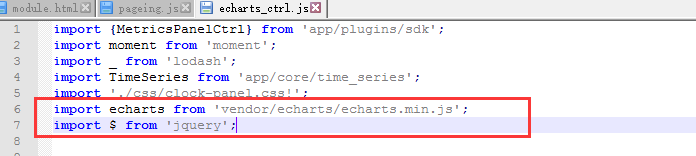
import \_ from 'lodash'; //等价于 import {default as \_} from "lodash";引入lodash.js，并别名\_

import $ from 'jquery';

import 'jquery.flot';

import 'jquery.flot.pie';

* **在module.ctrl.js中加入引用**



**为什么必须带.js?**

### 模块展示事件

* **Constructor方法**

this.events.on('panel-initialized', this.doEvent.bind(this, "panel-initialized"));

this.events.on('init-edit-mode', this.doEvent.bind(this, "init-edit-mode"));//编辑

this.events.on('render', this.onRender.bind(this)); //编辑后调用;给予;递交;表达

this.events.on('data-received', this.onDataReceived.bind(this));

this.events.on('data-error', this.onDataError.bind(this));

this.events.on('data-snapshot-load', this.doEvent.bind(this, 'data-snapshot-load'));

this.events.on('panel-teardown', this.doEvent.bind(this, "panel-teardown")); //卸载

* **link事件**

this.events.on('render', () => {

const $panelContainer = elem.find('.panel-container');

if (this.panel.bgColor) {

$panelContainer.css('background-color', this.panel.bgColor);

} else {

$panelContainer.css('background-color', '');

}

});

* **其他事件**

可在页面中使用AngularJS执行事件，数据绑定

### 模块编辑事件

onInitEditMode() {

this.addEditorTab('Options', 'public/plugins/grafana-piechart-panel/editor.html', 2);

this.unitFormats = kbn.getUnitFormats();

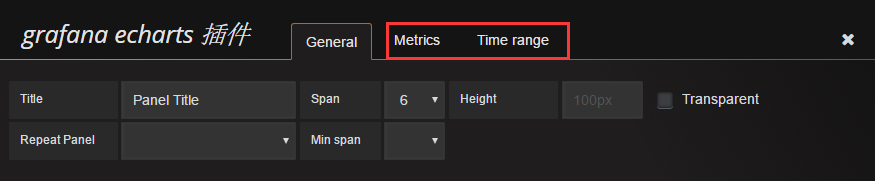
}

ng-style="{height: ctrl.height}"

# 插件怎么进行数据接入？

### 引入数据控件接口

import {MetricsPanelCtrl} from 'app/plugins/sdk';



### 定义自控件属性

var panelDefaults = {

legend: {

show: false, // disable/enable legend

values: true

}, //是否显示图例

maxDataPoints: 5, //限制显示多少个标签

targets: [], //标签

nullPointMode: 'connected',

legendType: 'Under graph', //图例样式

aliasColors: {},

format: 'short', //数据格式

valueName: 'avg' //取数方式

};

**panelDefaults定义了自定义控件展示的基本属性，附加到this.panel上，在控件展示时可根据这些属性显示控件；最重要的，panelDefaults的属性，可用于数据传递（实例化）；**

\_.defaults(this.panel, panelDefaults); //将panelDefaults属性附加到this.panel上

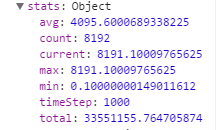
### 语法解释

* Lodash.js .map 循环dataList数据，并进行seriesHandler方法处理

this.series = dataList.map(this.seriesHandler.bind(this));

* Grafanna series.getFlotPairs

series.flotpairs = series.getFlotPairs(this.panel.nullPointMode);

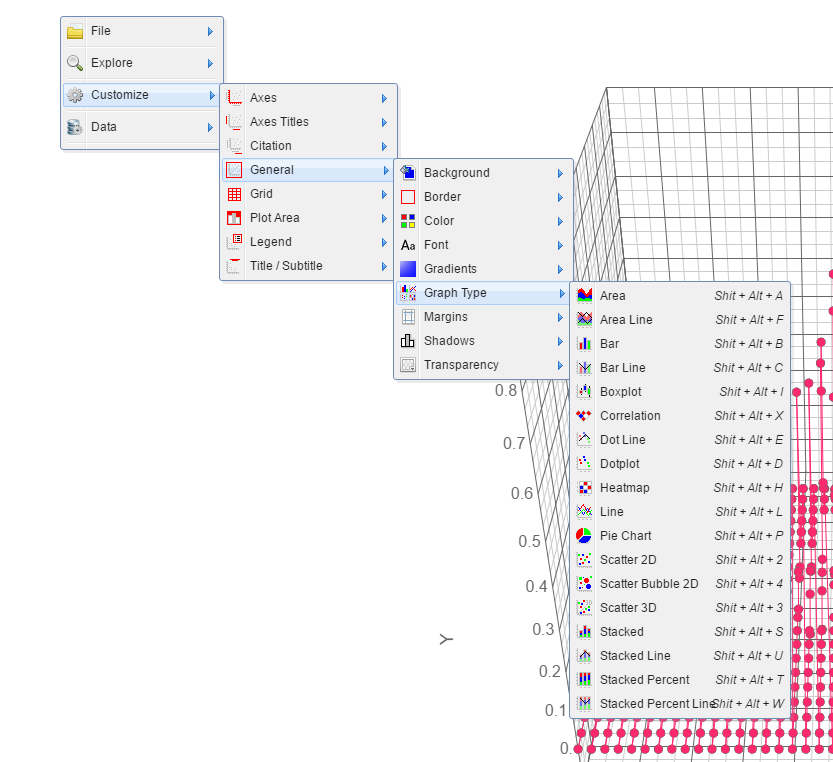


# CanvasXpress插件

插件首页：<http://canvasxpress.org/html/index.html>

### 插件使用

var cX = new CanvasXpress(target, data, config, events);



### 插件配置

"background":"rgb(31,29,29)", //背景色

"canvasBox":"true", //面板边框，默认false

"plotBox":"true", //表格边框，默认false

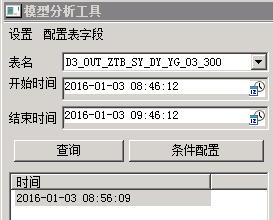
disableEvents //关闭所有事件，默认false

xRotate //起始度数旋转图在X轴

### 数据源分析

dss：<https://192.168.0.194:9443/carbon/admin/login.jsp>

API：<https://docs.wso2.com/display/DAS310/Analytics+REST+API+Guide>

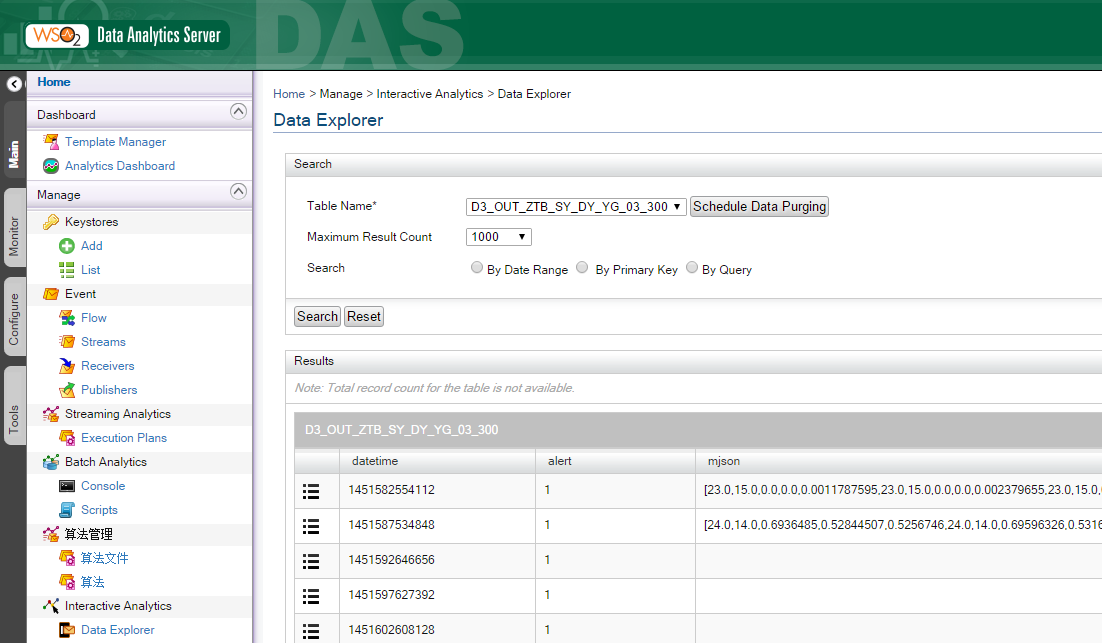


表名：D3\_OUT\_ZTB\_SY\_DY\_YG\_03\_300，model字段

时间段：2016-01-03 08:56:09

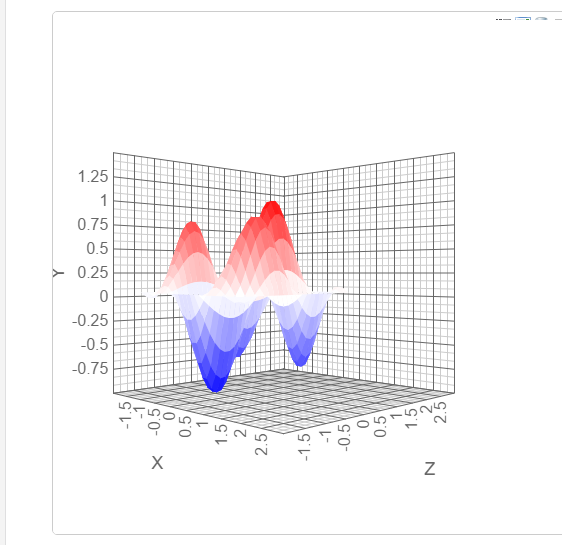
Google consle : +new Date('2016-01-03 08:56:09') => 1451782569984

接口查询：{\"tableName\":\"D3\_OUT\_ZTB\_SY\_DY\_YG\_03\_300\", \"query\":\"datetime : [1451782569000 TO 1451782569999]\",\"sortBy\":[{\"field\":\"datetime\", \"sortType\":\"DESC\"}], \"start\":0, \"count\":5}



### Scatter3D对象分析

展示效果：<http://canvasxpress.org/html/scatter3d.html>



y对象分析：

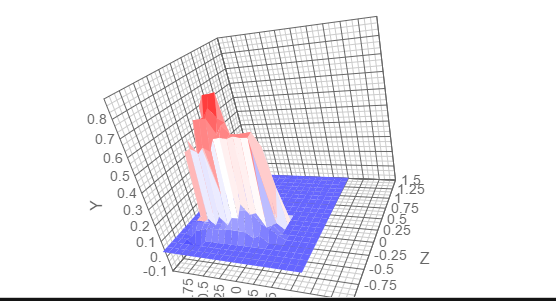
var：一维数组，数据标签数组

smps：["X","Y","Z"]

data：1\*3维矩阵[[1,2,3],[1,2,3]]

一维数组长度=data矩阵数据个数

### 结果



# Plot插件

<https://plot.ly/javascript/getting-started/>

<https://plot.ly/nodejs/getting-started/>

<https://plot.ly/javascript/>

<https://plot.ly/javascript/3d-surface-plots/>

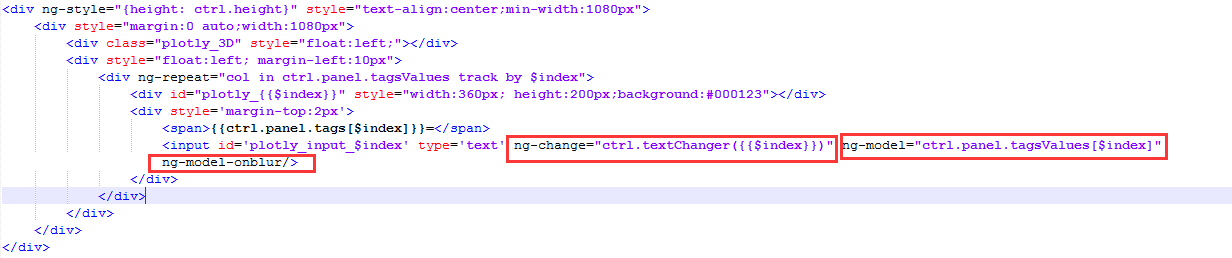
### DSS数据源

//time : 2016-04-30 10:16:33 1461982593000

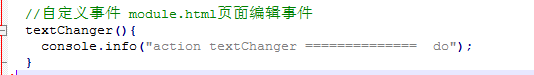
query = "{\"tableName\":\"D3\_OUT\_ZTB\_SY\_DY\_YG\_03\_300\", \"query\":\"datetime : [1461982593000 TO 1461982593999]\",\"sortBy\":[{\"field\":\"datetime\", \"sortType\":\"DESC\"}], \"start\":0, \"count\":5}";

### 页面编辑事件

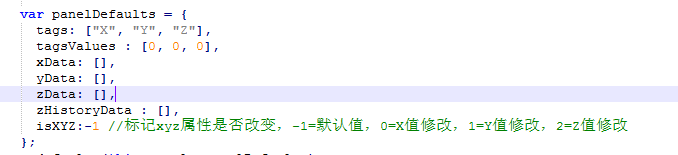
module.html，利用AngularJS绑定数据以及事件



plotly\_ctrl.js Constructor方法定义事件



plotly\_ctrl.js 定义model属性



# 数据源开发

数据源插件是面向任何数据库通信的HTTP插件。插件转换数据库数据为时间序列数据，任何grafana面板可以显示

<http://docs.grafana.org/plugins/datasources/>

<https://github.com/grafana/simple-json-datasource>

### module.js对象原型导出

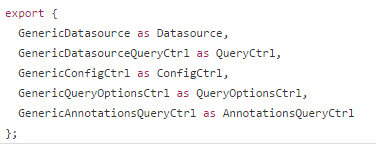
Datasource (Required)

QueryCtrl (Required)

ConfigCtrl (Required)

QueryOptionsCtrl

AnnotationsQueryCtrl



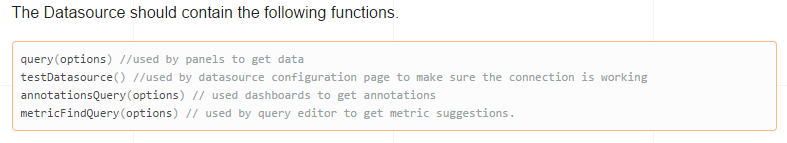
### plugin.json

"metrics": true,

"annotations": false,

设置插件可以提供什么样的数据。至少有一个是真的

### 接口实现



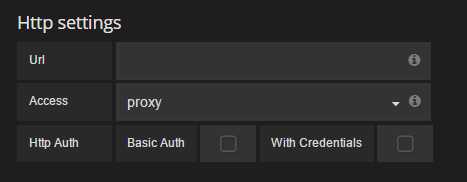
### 数据源配置ConfigCtrl

* 分析simple-json module.js



simple-json js代码里面找不到关于GenericConfigCtrl的实现，然后就直接导出了，但配置了页面partials/config.html

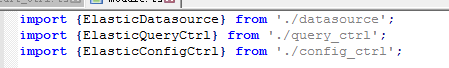




**数据源添加页面**

**=》查阅github grafana关于Elasticsearch、InfluxDB数据源等有特殊页面的配置，仿照UI，添加自定义数据源必要元素**

* 分析elasticsearch module.js



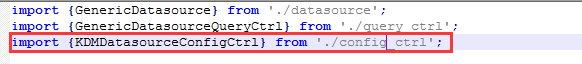
* 其他

源页面地址：public\app\features\plugins\partials\ds\_edit.html

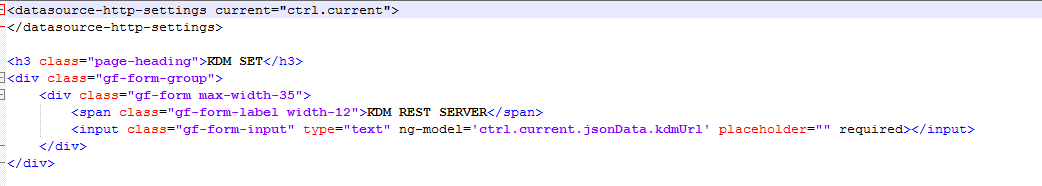
按钮事件js：public\app\features\plugins\ds\_edit\_ctrl.ts

实现自定义数据源步骤：

1. module.js文件

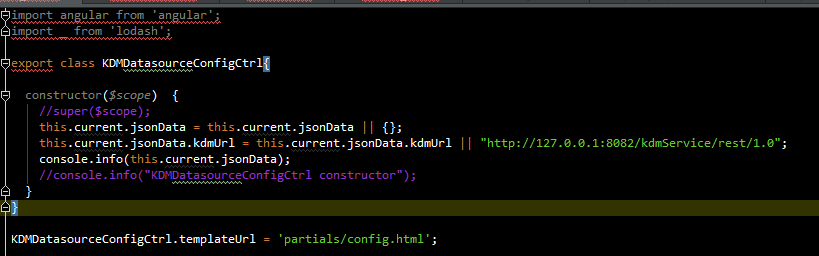


1. 页面config.html



1. config\_ctrl.js

给系统内置对象this.current.jsonData绑定一个kdmUrl属性，并赋予默认值，并在页面进行展示；如下代码，如果kdmUrl不存在，则使用默认地址（127.0.0.1），实现添加修改功能



1. 保存测试datasource.js

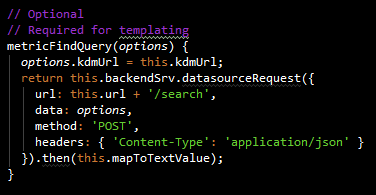


### 编辑接口search

实现标签查询功能

页面：partials/query.editor.html

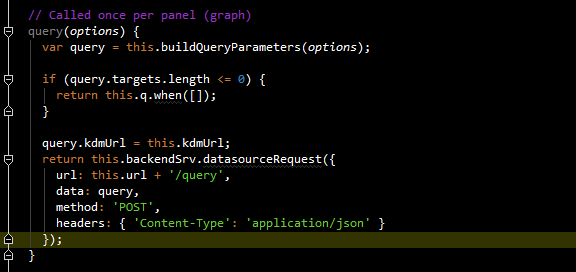
页面：query.options.html



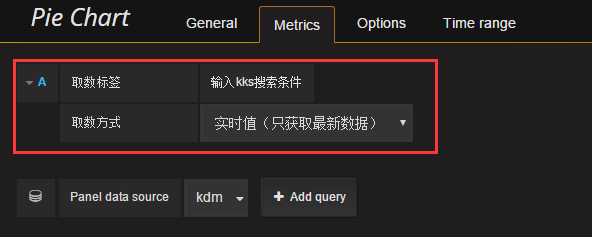
**实现标签搜索功能datasource.js**

### 查询接口Query

实现数据查询功能



**实现数据查询功能datasource.js**



**数据源绑定：query\_ctrl.js**

# Grafana样式

fa-arrow-up 上

fa-arrow-down 下

fa-remove X[删除]

fa-search 查询

fa-check 

fa-forward 

fa-folder-open 