**[onlyoffice](https://www.onlyoffice.com/)在线编辑私有化部署**

### 1、安装准备

如果是局域网，部署的电脑拥有固定ip（下文叫宿主机，切记）。电脑能打开cup的虚拟功能。如果部署到云主机，道理是一样的。

环境是win7以上，win10。Win10区别就是安装的docker软件不同而已（win10不要安装docker toolbox了），win10不需要设置端口转发。

linux就更简单了。

第一步：确定电脑打开了cpu虚拟打开。https://jingyan.baidu.com/article/22fe7ced3b4c003002617f17.html

第二步：下载和安装docker toolbox（win7）/docker（win10），默认安装即可。下载地址[DaoCloud | Docker 极速下载](http://get.daocloud.io/)，下载支持旧版windows的docker toolbox。win10就直接下载docker了。切记，win7下安装完成docker后，启动docker的时候断网，启动后可以联网。win10安装完docker后在cmd里执行docker命令。

第三步：启动docker后在里面拉取Onlyoffice Document Server 的Docker镜像，然后运行即可。

下面过程看起来繁琐，等你理解后其实就2步：安装docker，拉取镜像（或导入离线包），然后运行。再就是在文档管理平台上配置一下IP和端口。

### 2、拉取ONLYOFFICE Document Server 的Docker镜像

拉取镜像：启动docker，在里面输入命令docker pull onlyoffice/documentserver:( 版本号)

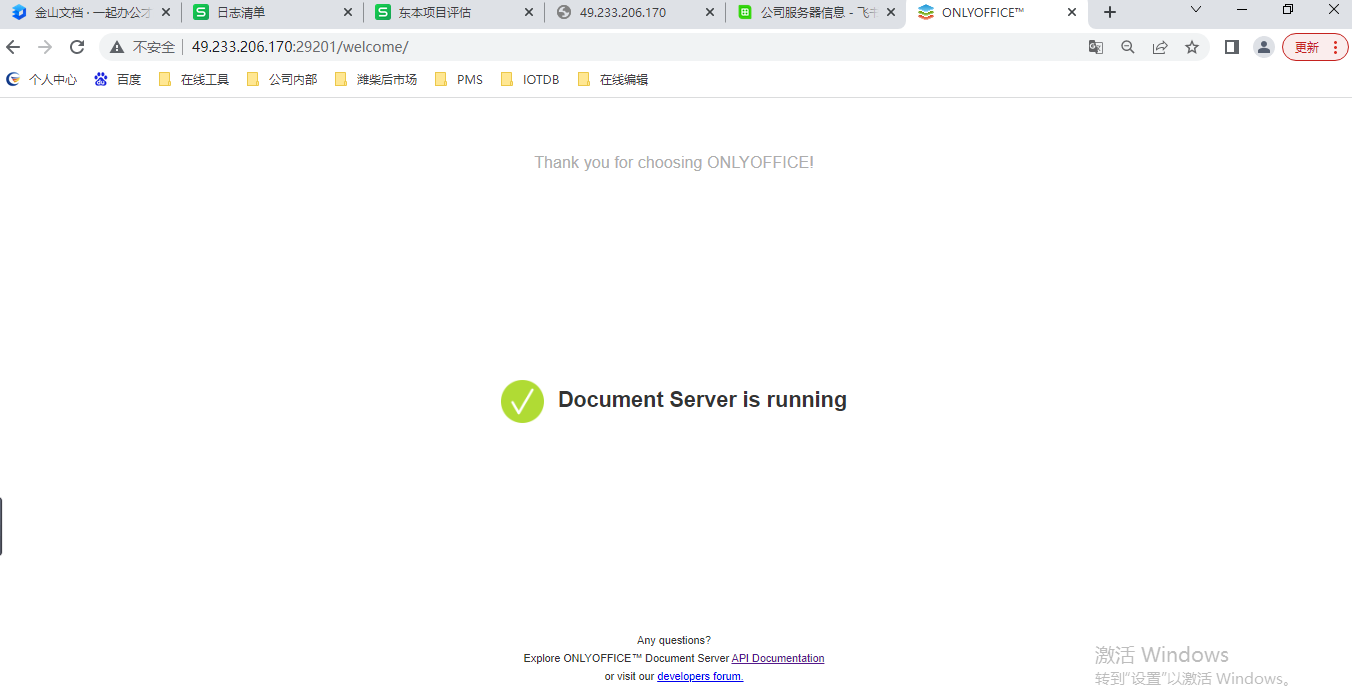
（注意：win7下的docker要断网启动！！切记，启动后再联网。如果已经运行了onlyoffice document server， 下次开机，重新启动docker后，不需要再运行onlyoffice的，当你启动docker后，OO已经启动了）

### 3、启动——映射至9000端口（运行镜像后变成容器）

docker run -i -t -d -p 9000:80 onlyoffice/documentserver

如果是导入的镜像，上面的命令中onlyoffice/documentserver用镜像id代替，查阅镜像用docker images命令。

运行后就可以通过http://192.168.99.100:9000访问OO服务的欢迎界面了，运行后的我们称之为容器。



### 4、Demo详解

小Demo已经上传到码云：<https://gitee.com/lei223/onlyofficeDemo>

1.Controller类详解

@Slf4j

@Controller

public class FileController {

@Value("${files.savePath}")

private String filePath;

@Value("${files.docservice.url.site}")

private String officeUrl;

@Value("${files.docservice.url.command}")

private String officeCommand;

@Autowired

private DocumentService documentService;

@Autowired

private FileUploadService uploadService;

@Autowired

RestTemplate restTemplate;

@ResponseBody

@PostMapping(value = "upload")

public ResponseEntity<Object> upload(@RequestParam("file") MultipartFile file, HttpServletRequest request) throws Exception {

if (file.isEmpty()) {

throw new Exception("上传文件不能为空");

}

FileUpload upload = new FileUpload();

String fileName = file.getOriginalFilename();

// if (!fileName.endsWith("xls") && !fileName.endsWith("xlsx")) {

// throw new Exception("请上传Excel文件");

// }

//更新保存文件信息到数据库

FileUtil.saveFile(file.getInputStream(), filePath + file.getOriginalFilename());

// System.out.println(fileName);

upload.setUpload\_date(new Date());

System.out.println(".".indexOf(fileName));

System.out.println(fileName.length());

upload.setFile\_type(fileName.substring(fileName.indexOf(".")));

upload.setFile\_path(filePath);

upload.setFile\_name(file.getOriginalFilename());

upload.setFile\_size(file.getSize());

uploadService.save(upload);

//操作人

// String operator=request.getAttribute(StrUtil.USER\_WORKNUMBER).toString();

// xxxService.saveUploadCkdExecl(file,operator);

return new ResponseEntity<Object>("上传成功", HttpStatus.OK);

}

/\*\*

\* \

\* 查询所有上传文档信息接口

\*

\* @return

\*/

@GetMapping("/filelist")

public ResponseEntity<Object> listFile() {

return new ResponseEntity<Object>(uploadService.list(), HttpStatus.OK);

}

// public ResponseEntity<Object> rview(){

//

// }

/\*\*

\* 下载文档接口

\* @param name

\* @param response

\*/

@GetMapping("/download")

public void download(String name, HttpServletResponse response) {

try {

FileUtil.downLoadFile(name,filePath,response);

} catch (IOException e) {

e.printStackTrace();

}

}

@GetMapping("/edit")

public String editDocFile(@RequestParam String name,String userName,String userId,Model model) {

String path = filePath+name;

// String name = "cc.docx";

Document document = documentService.getDocument(documentService.buildDocument(path, name));

model.addAttribute("document", document);

// 如果该格式不支持编辑，则返回预览页面

if (!documentService.canEdit(document)) {

return "/demo";

}

model.addAttribute("documentEditParam", documentService.buildDocumentEditParam(userId, userName,name));

return "/editor";

}

/\*\*

\* 编辑文档时回调接口

\* @param request

\* @param response

\* @throws IOException

\*/

@RequestMapping("/callback")

public void saveDocumentFile(HttpServletRequest request, HttpServletResponse response) throws IOException {

//处理编辑回调逻辑

callBack(request, response);

}

/\*\*

\*

\* @return

\*/

@GetMapping("/editStatus")

public ResponseEntity<Object> getDoucmentEditStatus(String name) throws ParseException {

String url = officeUrl+officeCommand;

Map<String,String> map = new HashMap<String,String>();

map.put("c", "forcesave");

String docFileMd5 = Md5Utils.getFileMd5(new File(filePath+name));

if (StringUtils.isBlank(docFileMd5)) {

throw new DocumentException(ErrorCodeEnum.DOC\_FILE\_MD5\_ERROR);

}

String pathShortMd5 = Md5Utils.md5(filePath + name);

String nameShortMd5 = Md5Utils.md5(name);

Hashids hashids = new Hashids(DocumentConstants.HASH\_KEY);

// (将路径字符串短md5值 + 名称字符串短md5值) ==> 再转成短id形式 ==> 作为文档的key（暂且认为是不会重复的）

String key = hashids.encodeHex(String.format("%s%s%s", docFileMd5,pathShortMd5, nameShortMd5));

map.put("key", key);

map.put("userdata", "sample userdata");

JSONObject obj = (JSONObject) new JSONParser().parse(FileUtil.editStatus(url, JSON.toJSONString(map)));

return new ResponseEntity<Object>(obj, HttpStatus.OK);

}

/\*\*

\* 处理在线编辑文档的逻辑

\* @param request

\* @param response

\* @throws IOException

\*/

private void callBack(HttpServletRequest request, HttpServletResponse response) throws IOException {

PrintWriter writer = null;

JSONObject jsonObj = null;

System.out.println("===saveeditedfile------------");

try {

writer = response.getWriter();

Scanner scanner = new Scanner(request.getInputStream()).useDelimiter("\\A");

String body = scanner.hasNext() ? scanner.next() : "";

jsonObj = (JSONObject) new JSONParser().parse(body);

System.out.println(jsonObj);

System.out.println("===saveeditedfile:" + jsonObj.get("status"));

/\*

0 - no document with the key identifier could be found,

1 - document is being edited,

2 - document is ready for saving,

3 - document saving error has occurred,

4 - document is closed with no changes,

6 - document is being edited, but the current document state is saved,

7 - error has occurred while force saving the document.

\* \*/

if ((long) jsonObj.get("status") == 2) {

FileUtil.callBackSaveDocument(jsonObj,filePath,request, response);

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

/\*

\* status = 1，我们给onlyoffice的服务返回{"error":"0"}的信息，这样onlyoffice会认为回调接口是没问题的，这样就可以在线编辑文档了，否则的话会弹出窗口说明

\* 在线编辑还没有关闭，前端有人下载文档时，强制保存最新内容 当status 是6时说明有人在编辑时下载文档

\* \*/

System.out.println(jsonObj.get("status"));

if ((long) jsonObj.get("status") == 6) {

//处理当文档正在编辑为关闭时，下载文档

if (((String)jsonObj.get("userdata")).equals("sample userdata")){

FileUtil.callBackSaveDocument(jsonObj,filePath,request, response);

}

System.out.println("====保存失败:");

writer.write("{\"error\":1}");

} else {

//执行删除编辑时下载保存的文件:

FileUtil.deleteTempFile(filePath,request.getParameter("fileName"));

writer.write("{\"error\":0}");

}

}

}

1》/edit 此接口是返回给前端onlyoffice的编辑界面并打开要编辑的文件

2》/download 下载文件接口，onlyoffice需要的接口之一，当将文档参数传入到onlyoffice服务器时需要一个download 的url 来让onlyoffice将文件下载到服务器配置参数如下：



3》/callback 接口 此接口是onlyoffice服务器对于你这服务的回调接口，当打开文档时，编辑完成后保存文档，或是正在编辑时，下载文档调用onlyoffice的强制保存命令之后都会回调这个接口。

4》/editStatus 获取文档的编辑状态，用于编辑时下载文档，如果用户下载的文档正在被编辑，就可以先调用此接口，向onlyoffice服务器发送强制保存的命令，然后会调用callback接口进行保存。

以上的四个接口是，demo对于onlyoffice服务器的大部分的集成逻辑都在这几个接口的实现细节里。当然还有另外的，上传文件接口和提供文件列表查询的接口。

2.下边说下修改配置 然后运行起来。

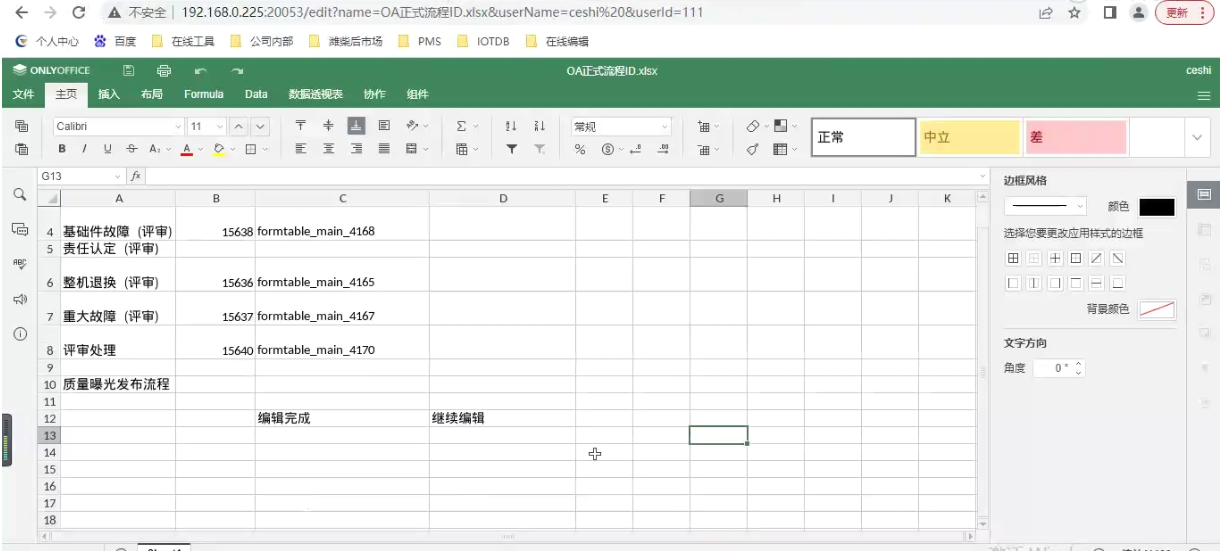
先说前端配置。将请求url中的ip加端口都替换成你本地服务的ip+端口



修改 java工程中的url配置：url的配置全部在application.yml中：



3.示例效果:



### 5、前端组件集成属性详解

function initDoc(key, url, fileType, title, model, callbackUrl) {

let config = {

"document": {

"documentType": "text",

"width": "100%", //打开窗口宽度

"height": "100%", //打开窗口高度

"fileType": fileType, //文档类型

"key": key, //定义用于服务识别文档的唯一文档标识符。每次编辑和保存文档时，都必须重新生成密钥。长度限制为128个符号。

"title": title, //为查看或编辑的文档定义所需的文件名，该文件名也将在下载文档时用作文件名。长度限制为128个符号。

"url": url, //定义存储原始查看或编辑的文档的绝对URL

"info": {

"owner": "王重阳", //文件创建者名称

"sharingSettings": [ //文件对应用户的操作权限配置

{

"permissions": "Full Access", // 完全操作权限-Full Access,只读权限-Read Only 拒绝访问-Deny Access

"user": "林朝英" //有次权限的用户

},

{

"permissions": "Read Only",

"user": "周伯通"

},

],

"uploaded": "2010-07-07 3:46 PM" //文件创建时间

},

//文档权限参数

"permissions": {

"edit": true, //（文件是否可以编辑，false时文件不可编辑）

"fillForms": true, //定义是否能在文档中填充表单

"print": true, //定义文档是否能打印

"review": false, //第一是否显示审阅文档菜单

"comment": true, //定义是否可以注释文档。如果注释权限设置为“ true”，则文档侧栏将包含“注释”菜单选项；只有将mode参数设置为edit时才生效，默认值与edit参数的值一致。

"copy": true, //是否允许您将内容复制到剪贴板。默认值为true。

"download": true, //定义是否可以下载文档或仅在线查看或编辑文档。如果下载权限设置为“false”下载为菜单选项将没有。默认值为true。

"modifyContentControl": true, //定义是否可以更改内容控件设置。仅当mode参数设置为edit时，内容控件修改才可用于文档编辑器。默认值为true。

"modifyFilter": true, //定义过滤器是否可以全局应用（true）影响所有其他用户，或局部应用（false），即仅适用于当前用户。如果将mode参数设置为edit，则过滤器修改仅对电子表格编辑器可用。默认值为true。

}

},

// type: "embedded",

//打开文档类型

// text对应各种文档类型(.doc, .docm, .docx, .dot, .dotm, .dotx, .epub, .fodt, .htm, .html, .mht, .odt, .ott, .pdf, .rtf, .txt, .djvu, .xps)

//spreadsheet对应表格类型(.csv, .fods, .ods, .ots, .xls, .xlsm, .xlsx, .xlt, .xltm, .xltx)

//presentation对应PPT类型(.fodp, .odp, .otp, .pot, .potm, .potx, .pps, .ppsm, .ppsx, .ppt, .pptm, .pptx)

"editorConfig": { //编辑配置

"createUrl": "http://docServer:port/url-to-create-document/", //指定创建文件的页面,添加该配置后文档服务器插件才会显示新建文件按钮

"mode": model, //文档操作模式 view 视图模式不可编辑 edit 编辑模式可编辑文档

"callbackUrl": callbackUrl, //保存文件时的回调地址

"lang": "zh-CN", //语言环境

"customization": { //定制部分允许自定义编辑器界面，使其看起来像您的其他产品，并更改是否存在其他按钮，链接，更改徽标和编辑者所有者详细信息。

"help": false, //定义是显示还是隐藏“帮助”菜单按钮。默认值为true。

"hideRightMenu": false, //定义在第一次加载时是显示还是隐藏右侧菜单。默认值为false。

"autosave": false, //定义是启用还是禁用“自动保存”菜单选项。请注意，如果您在菜单中更改此选项，它将被保存到浏览器的localStorage中。默认值为true。

"forcesave": true, //定义保存按钮是否显示默认false

"chat": false, //定义“聊天”菜单按钮是显示还是隐藏；请注意，如果您隐藏“聊天”按钮，则相应的聊天功能也将被禁用。默认值为true。

"commentAuthorOnly": false, //定义用户是否只能编辑和删除他的评论。默认值为false。

"comments": false, //定义是显示还是隐藏“注释”菜单按钮；请注意，如果您隐藏“评论”按钮，则相应的评论功能将仅可用于查看，评论的添加和编辑将不可用。默认值为true。

"compactHeader": false, //定义是否将菜单栏放在在徽标旁边使界面更加紧凑默认false。

"compactToolbar": false, //定义显示的顶部工具栏类型是完整（false）还是紧凑true。默认值为false 多余菜单将在右侧折叠点击显示。

"compatibleFeatures": false, //定义仅与OOXML格式兼容的功能的使用。例如，不要在整个文档上使用注释。默认值为false。

"macros": false, //定义是否将运行文档宏以及可用的宏设置。默认值为true。

"macrosMode": "warn", //定义是否将运行文档宏。可以采用以下值： disable -根本不运行；enable -自动运行所有宏；warn -警告宏并请求允许运行。默认值为original。

"plugins": false, //定义是否将启动插件并可用。默认值为true。

"showReviewChanges": false, //定义在加载编辑器时是否自动显示或隐藏审阅更改面板。默认值为false。

"spellcheck": false, //定义在加载编辑器时是否自动打开或关闭拼写检查器。拼写检查器仅适用于文档编辑器和演示文稿编辑器。默认值为true。

"toolbarNoTabs": false, //定义是突出显示顶部工具栏选项卡样式。默认值为false。

"unit": "cm", //定义在标尺和对话框中使用的度量单位。可以采用以下值：cm -厘米，pt-点，inch -英寸。默认值为厘米（cm）。

"zoom": 100, //定义以百分比为单位的文档显示缩放值。可以取大于0的值。对于文本文档和演示文稿，可以将此参数设置为-1（使文档适合页面选项）或-2（使文档页面宽度适合编辑器页面）。默认值为100。

"customer": { //关于 文档编辑器的显示信息

"address": "My City, 123a-45", //有权访问编辑或编辑作者的公司或个人的邮政地址，

"info": "Some additional information", //有关您希望其他人认识的公司或个人的一些其他信息，

"logo": "https://example.com/logo-big.png", //图片徽标的路径（此文件没有特别建议，但是如果使用透明背景的.png格式会更好）。图片必须具有以下尺寸：432x70，

"mail": "john@example.com", //有权访问编辑者或编辑者的公司或个人的电子邮件地址

"name": "欧阳锋", //该公司或个人的谁可以访问编辑或编辑作者，名称

"www": "example.com" //以上公司或个人的家庭网站地址，

},

"feedback": { //反馈配置信息

"url": "https://example.com", //单击“反馈和支持”菜单按钮时将打开的网站地址的绝对URL ，

"visible": false //显示或隐藏“反馈和支持”菜单按钮，

},

"goback": { //定义“打开文件位置”菜单按钮和右上角按钮的设置。该对象具有以下参数：

"blank": true, //在新的浏览器选项卡/窗口（如果值设置为true）或当前选项卡（如果值设置为false）中打开网站。默认值为true，

"requestClose": false, //定义如果单击“打开文件位置”按钮，则调用events.onRequestClose事件，而不是打开浏览器选项卡或窗口。默认值为false，

"text": "Open file location", //将在“打开文件位置”菜单按钮和右上角按钮（即，而不是“转到文档”）上显示的文本，

"url": "https://example.com" //单击“打开文件位置”菜单按钮时将打开的网站地址的绝对URL ，

},

"logo": {

"image": "https://example.com/logo.png", //图像文件的路径，用于在普通工作模式下显示（即，在所有编辑器的查看和编辑模式下）。图片必须具有以下尺寸：172x40，

"imageEmbedded": "https://example.com/logo\_em.png", //用于以嵌入式模式显示的图像文件的路径（请参阅config部分以了解如何定义嵌入式文档类型）。图片必须具有以下尺寸：248x40，

"url": "https://www.baidu.com" //某人单击徽标图像时将使用的绝对URL（可用于转到您的网站等）。保留为空字符串或null以使徽标不可单击，

},

},

"user": { //用户信息

"id": "admin", //用户ID

"name": "操作员" //用户全名称

},

"embedded": { //Embedded部分仅适用于嵌入式文档类型（请参阅config部分以了解如何定义嵌入式文档类型）。它允许更改设置，这些设置定义嵌入式模式下按钮的行为。

"embedUrl": "https://example.com/embedded?doc=exampledocument1.docx", //定义文档的绝对URL，以作为嵌入到网页中的文档的源文件

"fullscreenUrl": "https://example.com/embedded?doc=exampledocument1.docx#fullscreen", //定义将以全屏模式打开的文档的绝对URL。

"saveUrl": "https://example.com/download?doc=exampledocument1.docx", //定义允许将文档保存到用户个人计算机上的绝对URL。

"shareUrl": "https://example.com/view?doc=exampledocument1.docx", //定义允许其他用户共享此文档的绝对URL。

"toolbarDocked": "top" //定义嵌入式查看器工具栏的位置，可以为top或bottom。

}

},

"events": { //事件配置

// onAppReady,//-将应用程序加载到浏览器时调用的函数。

// onCollaborativeChanges //-当文档由其他用户在严格共同编辑模式下共同编辑时调用的函数。

// onDocumentReady,//-将应用程序加载到浏览器时调用的函数。

// onDocumentStateChange,//-修改文档时调用的函数。这就是所谓的使用参数：{真正的“数据”}在当前用户编辑文档以及与参数：{“数据”：假}在当前用户的更改发送到文档编辑服务。

// onDownloadAs,//-调用downloadAs方法时，使用指向已编辑文件的绝对URL调用的函数。在data参数中发送要下载的文档的绝对URL 。

// onError,//-发生错误或其他特定事件时调用的函数。错误消息在data参数中发送。==

// onInfo,//-应用程序打开文件时调用的函数。该模式在data.mode参数中发送。可以查看或编辑。

// onMetaChange,//-通过meta命令更改文档的元信息时调用的函数。文档名称通过data.title参数发送。

// onOutdatedVersion,//-使用旧的document.key值打开文档进行编辑时，显示错误后调用的函数，该值用于编辑先前的文档版本并已成功保存。调用此事件时，必须使用新的document.key重新初始化编辑器。

// onReady,//-将应用程序加载到浏览器时调用的函数。自从5.0版本不推荐使用，请使用onAppReady代替

// onRequestClose,//-结束编辑器的工作并且必须关闭编辑器时调用的函数。==

// onRequestCompareFile,//-用户尝试通过单击“存储中的文档”按钮来选择要比较的文档时调用的函数。要选择要比较的文档，必须调用setRevisedFile方法。如果未声明该方法，则不会显示“来自存储的文档”按钮。

// onRequestCreateNew,//-用户尝试通过单击“新建”按钮来创建文档时调用的函数。使用此方法代替createUrl字段。如果未声明该方法且未指定createUrl，则将不会显示“创建新”按钮。

// onRequestEditRights,//-用户尝试通过单击“编辑文档”按钮尝试将文档从视图切换到编辑模式时调用的函数。调用该函数时，必须在编辑模式下再次初始化编辑器。如果未声明该方法，则不会显示“编辑”按钮。

// onRequestHistory,//-用户尝试通过单击“版本历史记录”按钮显示文档版本历史记录时调用的函数。要显示文档版本历史，您必须调用refreshHistory方法。如果未声明该方法和onRequestHistoryData方法，则不会显示“版本历史记录”按钮。

// onRequestHistoryClose,//-当用户尝试通过单击“关闭历史记录”按钮来查看文档版本历史记录时，试图调用该文档时调用的函数。调用该函数时，必须在编辑模式下再次初始化编辑器。如果未声明该方法，则不会显示“关闭历史记录”按钮。

// onRequestHistoryData,//-用户尝试单击文档版本历史记录中的特定文档版本时调用的函数。

// onRequestInsertImage,//-用户尝试通过单击“保存图像”按钮插入图像时调用的函数。图像插入的类型在参数data.c中指定。

// onRequestRename,//-用户尝试通过单击“重命名...”按钮重命名文件时调用的函数。

// onRequestRestore,//-用户单击版本历史记录中的“还原”按钮来还原文件版本时调用的函数。

// onRequestSaveAs,//-用户尝试通过单击“另存为...”按钮保存文件时调用的函数。文档的标题和要下载的文档的绝对URL在data参数中发送。如果未声明该方法，则不会显示“另存为...”按钮。

// onRequestSharingSettings,//-用户单击“更改访问权限”按钮来管理文档访问权限时调用的函数。必须调用setSharingSettings方法来更新有关允许与其他用户共享文档的设置的信息。如果未声明该方法，则不会显示“更改访问权限”按钮。

// onRequestUsers,//-评论者可以选择要在评论中提及的其他用户时调用的函数。要设置用户列表，必须调用setUsers方法。

// onWarning,//-发生警告时调用的函数。警告消息在data参数中发送。

// "onDocumentStateChange": function() {

// }, //文档改变后的回调

//"onDocumentReady" : onDocumentReady, //文档初始化准备好后的回调

},

};

var docEditor = new DocsAPI.DocEditor("placeholder", config);

}