目 录

前言

高级参数

default.json-文件大小

Docker 文档

前端集成

- https://vitepress.dev/reference/default-theme-home-page
- React
- Vue 2.x
- Vue 3.x

开启token验证

免费镜像

- 免费镜像8.2.0
- 免费镜像8.0.0
- 免费镜像7.5.1.23
- 免费镜像7.1.1.23

如何配置代理

docx 转 pdf

前端SDK

常见错误

字体名称

中文字号

中文字体

文件大小限制

试一试

插件API

- 插件
- 模板替换
- 插件入门
- 如何与插件通信
- 水印

付费

文档生成器（￥499）
文档生成器（￥499）
付费镜像
付费镜像
付费镜像
连接器
付费镜像
付费镜像
付费咨询
特殊功能定制
更新日志
联系方式
8.3版笔记
centos升级和docker升级
onlyoffice8.2-token-golang

前言

离线部署

首推用centos之类的linux系统，其次win11，再次win10
1.首先，安装docker。linux系统见这里。
2.其次，dokcer命令行导入离线包，离线包下载地址：
链接: https://pan.baidu.com/s/1wfUHVDrM1fablcFV301W3Q 提取码: avvp 复制这段内容后打开百度网盘手机App，操作更方便哦

下载好离线包，
3.在docker命令行下用命令 `docker load < 离线包.tar` ，
4.然后再用命令 `docker run -i -t -d -p 9000:80 -e JWT_ENABLED=false --restart=always knoxzhang/oo-ce-docker-license:8.2.0.1`
5.根据这里的方法，将onlyoffice的default.json文件修改，不然就无法下载文件。

6.再次，下载一个文档管理系统，可以用我的engineercms（网盘里的2025***engineercms_publish.zip），或者其他nextcloud之类的支持onlyoffice的。
engineercms下载地址：链接：https://pan.baidu.com/s/1f4zuhoymaHMN_QAEgZLwOg?pwd=upvm
提取码：upvm

7.下载解压后，进入conf文件夹，打开app.conf，修改里面的onlyoffice的2个地址，保存。

```
##### 配置onlyoffice #####
#### 1.onlyoffice的服务ip和端口号，或域名，如http://172.26.32.1:9000;https://zsj.itdos.com/
onlyofficeapi_url = http://192.168.100.37:9002
#### 2.engineercms服务的ip和端口号，或域名，如http://172.26.32.1:8081;https://zsj.itdos.com/
engineercmsapi_url = http://192.168.137.1:8081
#### 3.是否开启onlyoffice token,true or false#####
JWT_ENABLED = true
#### 4.onlyoffice token secret#####
onlyoffice_token_secret = RLP9Qe2Mb8
```

8.win系统，打开cmd命令行窗口进入engineercms目录下直接运行engineercms.exe文件即可。
任何局域网用户用浏览器输入这台电脑的 `ip:端口号` 即可。
可以将conf里的app.conf端口号设置为80，那么访问地址就不用输端口号。
linux系统见readme.md文件。

过程可以查阅下列，有点乱。
https://kchzhang.github.io/ooffice-docs-zh-cn/integration/

关闭token验证

https://gitcode.com/knoxZhang/ooffice-docs-zh-cn

重启系统：
`reboot`

查看当前系统内核版本

`uname -r`
查看日志：
`docker logs -f -t --tail 500 <容器名称/容器 ID>`
一、问题描述
启动docker服务： `sudo systemctl restart docker`
提示：Job for docker.service failed because the control process exited with error code. See “systemctl status docker.service” and “journalctl -xe” for details.
查看服务信息： `systemctl status docker.service`
二、问题解决
1、进入docker目录：cd /etc/docker/
2、修改daemon的类型： `mv daemon.json daemon.conf`
3、重启docker： `systemctl restart docker`

https://www.cnblogs.com/ybli/p/18593341

```
[root@localhost ~]# tee /etc/docker/daemon.json <<EOF
{
    "registry-mirrors" : [
      "https://docker.registry.cyou",
      "https://docker-cf.registry.cyou",
      "https://dockercf.jsdelivr.fyi",
      "https://docker.jsdelivr.fyi",
      "https://dockertest.jsdelivr.fyi",
      "https://mirror.aliyuncs.com",
      "https://dockerproxy.com",
      "https://mirror.baidubce.com",
      "https://docker.m.daocloud.io",
      "https://docker.nju.edu.cn",
      "https://docker.mirrors.sjtug.sjtu.edu.cn",
      "https://docker.mirrors.ustc.edu.cn",
```

```
      "https://mirror.iscas.ac.cn",
      "https://docker.rainbond.cc"
  ]
}
> EOF
{
    "registry-mirrors": [
      "https://docker.registry.cyou",
      "https://docker-cf.registry.cyou",
      "https://dockercf.jsdelivr.fyi",
      "https://docker.jsdelivr.fyi",
      "https://dockertest.jsdelivr.fyi",
      "https://mirror.aliyuncs.com",
      "https://dockerproxy.com",
      "https://mirror.baidubce.com",
      "https://docker.m.daocloud.io",
      "https://docker.nju.edu.cn",
      "https://docker.mirrors.sjtug.sjtu.edu.cn",
      "https://docker.mirrors.ustc.edu.cn",
      "https://mirror.iscas.ac.cn",
      "https://docker.rainbond.cc"
    ]
}
```

重载和重启：好家伙，这时候 systemctl 就能用上了， 这个地方卡了我好久，气死了家人们

```
[root@localhost ~]# systemctl daemon-reload

systemctl restart docker
```

开机自启docker

`systemctl enable docker`

拉取

`docker pull knoxzhang/oo-ce-docker-license:8.2.0.1`

启动

`docker run -i -t -d -p 9000:80 -e JWT_ENABLED=false --restart=always knoxzhang/oo-ce-docker-license:8.2.0.1`

```
# 载入镜像
docker load < onlyoffice.tar
运行镜像，采用关闭token方式运行。变成容器
docker run -i -t -d -p 9000:80 -e JWT_ENABLED=false --restart=always knox
zhang/oo-ce-docker-license:
docker run -i -t -d -p 9000:80 --restart=always fei6onlyoffice/documentse
rver

# 拉取
docker pull knoxzhang/oo-ce-docker-license:8.0.0.1
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always knoxzhang/
oo-ce-docker-license:8.0.0.1
[root@localhost ~]# docker load < oo8.2.0.1.tar
2573e0d81582: Loading layer  80.41MB/80.41MB
67e7b6faf68f: Loading layer  2.005GB/2.005GB
e515863057db: Loading layer   7.68kB/7.68kB
94a45f095275: Loading layer  9.216kB/9.216kB
38135d309352: Loading layer   29.7kB/29.7kB
b25204e29e20: Loading layer  3.072kB/3.072kB
be9a1d63bfc9: Loading layer  1.818GB/1.818GB
b7f761e1695f: Loading layer  151.6MB/151.6MB
065285a5f5e0: Loading layer  161.4MB/161.4MB
Loaded image: knoxzhang/oo-ce-docker-license:8.2.0.1

[root@localhost ~]# docker images
REPOSITORY                          TAG                 IMAGE ID
CREATED             SIZE
knoxzhang/oo-ce-docker-license      8.2.0.1             8132e7048f4f
7 weeks ago         4.16GB
webterminal/magick2                 latest              257423def561
3 years ago         348MB
webterminal/magick                  latest              75ff1e076d70
3 years ago         348MB
fei620onlyoffice/documentserver     latest              79b80c83e29b
3 years ago         4.24GB
[root@localhost ~]# docker run --name=ooffice -i -t -d -p 9001:80 -e JWT_
ENABLED=false --restart=always 813
1ad1be830a5838e46c1d10269ffe30c598e90bdf8c44116c4a38c90f5d39f16c
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND
CREATED             STATUS                  PORTS
```

```
NAMES
1ad1be830a58        813                                  "/app/ds/run-docume
n…"   13 seconds ago      Up 8 seconds              443/tcp, 0.0.0.0:9001-
>80/tcp   ooffice
879eeedd1ddf        fei620onlyoffice/documentserver   "/app/ds/run-docume
n…"   13 months ago       Up 2 months               443/tcp, 0.0.0.0:9000-
>80/tcp   eager_lalande
ea98010d49a2        webterminal/magick                "bash"
3 years ago         Exited (0) 2 years ago
quirky_chatterjee
[root@localhost ~]#
```

列出索引容器

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND
CREATED             STATUS                     PORTS
NAMES
01b845beb144        fei620onlyoffice/documentserver   "/app/ds/run-docume
n…"   2 months ago        Created
silly_mirzakhani
879eeedd1ddf        fei620onlyoffice/documentserver   "/app/ds/run-docume
n…"   13 months ago       Up 2 months               443/tcp, 0.0.0.0:9000-
>80/tcp   eager_lalande
ea98010d49a2        webterminal/magick                "bash"
3 years ago         Exited (0) 2 years ago
quirky_chatterjee
```

删除指定容器

```
[root@localhost ~]# docker rm 01b
01b
```

查看剩余容器

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                                COMMAND
CREATED             STATUS                     PORTS
NAMES
879eeedd1ddf        fei620onlyoffice/documentserver   "/app/ds/run-docume
n…"   13 months ago       Up 2 months               443/tcp, 0.0.0.0:9000-
>80/tcp   eager_lalande
ea98010d49a2        webterminal/magick                "bash"
3 years ago         Exited (0) 2 years ago
```

Linux CentOS7 中 Docker 的安装与升级（在线安装 + 离线安装）
Linux CentOS7 升级内核
QQ交流群：792389659

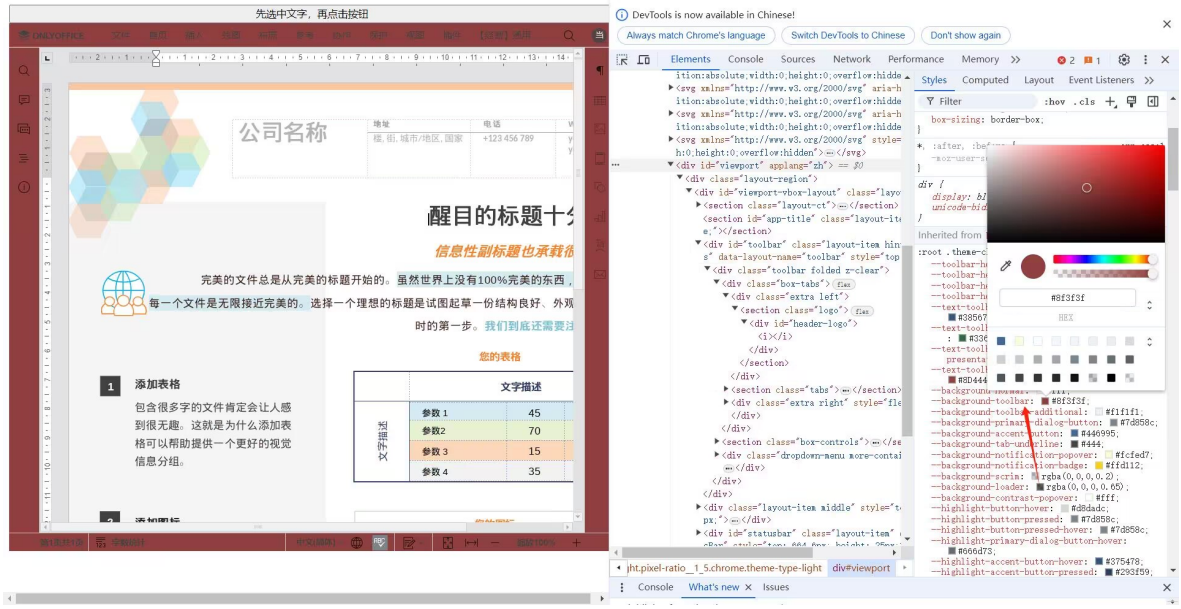
qq-group

如果有帮助到您，可以请作者喝一杯果汁

wxpay

Powered by Froala Editor

–toolbar-header-document: 你要的颜色;这个是改toorbar的背景颜色
/web-apps/apps/common/main/resources/img/header/header-logo_s.svg这个是logo文件地址
你想改啥都在这里面



documentserver/server/DocService

/var/www/onlyoffice/
documentserver/server/FileConverter

然后运行supervisorctl restart all即可

3、增加mobile端编辑功能：

容器：

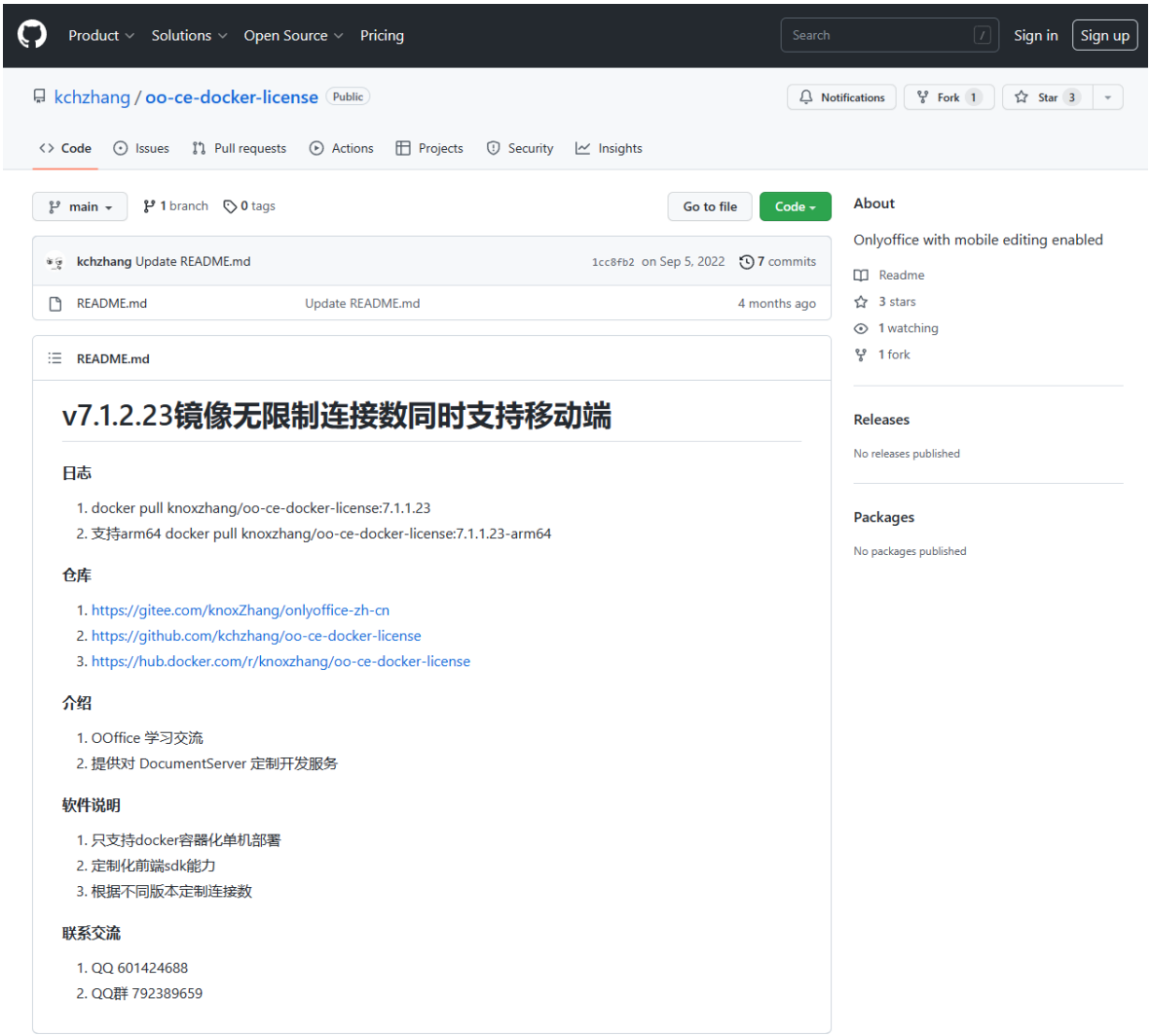
/var/www/onlyoffice/
documentserver/web-apps/apps/
spreadsheeteditor/mobile/dist/js/app.js

/var/www/onlyoffice/
documentserver/web-apps/apps/
documenteditor/mobile/dist/js/app.js

/var/www/onlyoffice/
documentserver/web-apps/apps/
presentationeditor/mobile/dist/js/app.js

三个文件分别找到
isSupportEditFeature=function(){return!1}，去掉！即可

4、修改容器JWT为默认不启动：



https://github.com/wytxer/demo-onlyoffice 这个作者开发的demo对我很有启发，分享一下

Windows装Docker至D盘/其他盘（最新，最准确，直接装）

https://www.cnblogs.com/haima/p/18697360
前言
Docker的默认安装路径为

```
C:\你的用户名\AppData\Local\Docker\wsl
```

这样安装常常会导致C盘爆满。目前现有博客的安装方法往往不能把docker的container和image也装在非C盘。本博客旨在用最简单的方式，把Docker Deskstop的images和container装在D盘中。
安装前，请提前打开WSL2,

步骤1: 下载Docker Desktop安装包
首先，访问官方安装指南下载Docker Desktop的安装包。

步骤2: 准备命令行
确保你已经将Docker Desktop安装包保存到一个易于访问的位置，例如E:\EdgeDownload\Docker Desktop Installer.exe。

进入 cd E:\EdgeDownload\Docker Desktop Installer.exe

步骤3: 修改安装路径
打开命令提示符（CMD）或PowerShell，然后输入以下命令来更改Docker Desktop的安装路径：

```
start /w "" "Docker Desktop Installer.exe" install -accept-license --installation-dir="E:\Program Files\Docker" --wsl-default-data-root="E:\Program Files\Docker\data" --windows-containers-default-data-root="E:\Program Files\Docker"
```

参考文档：
Windows装Docker至D盘/其他盘（最新，最准确，直接装）

```
设置docker-desktop为wsl默认子系统
## 显示所有子系统
D:\Users\Administrator\Downloads>wsl -l -v
  NAME              STATE           VERSION
* Ubuntu-20.04      Running         2
  docker-desktop    Running         2
## 设置docker-desktop为wsl默认子系统
D:\Users\Administrator\Downloads>wsl -s docker-desktop
操作成功完成。

D:\Users\Administrator\Downloads>wsl -l -v
  NAME              STATE           VERSION
* docker-desktop    Running         2
  Ubuntu-20.04      Running         2
```

参考文档：

windows上linux子系统wsl的启用和管理
查看docker版本

```
C:\Users\Administrator>docker --version
Docker version 27.4.0, build bde2b89
```

设置镜像代理

参考文档:
https://www.cnblogs.com/Flat-White/p/17107494.html

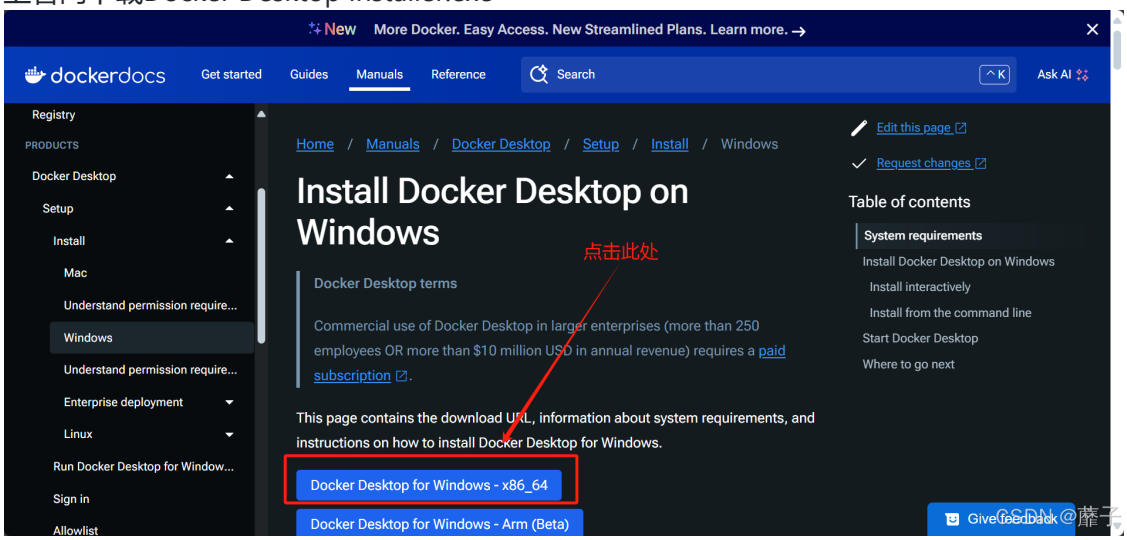
下载镜像
docker pull nginx
docker pull redis:latest
docker pull mysql:5.7.19
参考文档:
https://www.jianshu.com/p/19901ee5ad3c

[Haima的博客] http://www.cnblogs.com/haima/

Windows装Docker至D盘/其他盘（最新，最准确，直接装）

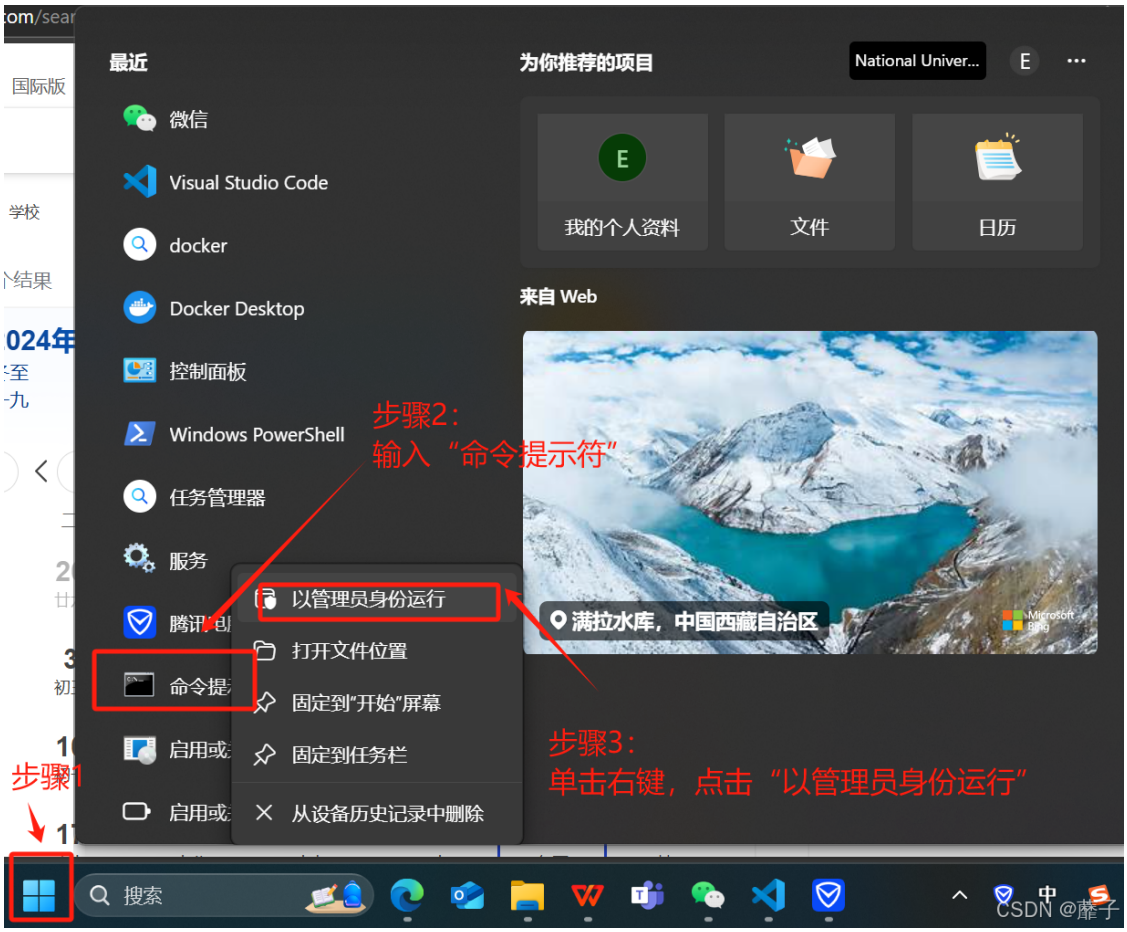
前言
Docker的默认安装路径为
C:\你的用户名\AppData\Local\Docker\wsl
这样安装常常会导致C盘爆满。目前现有博客的安装方法往往不能把docker的container和image也装在非C盘。本博客旨在用最简单的方式，把Docker Deskstop的images和container装在D盘中。
安装前，请提前打开WSL2,
步骤

1. 上官网下载Docker Desktop Installer.exe





2. 以管理员身份打开Windows终端，并转到你下载文件的位置。



输入：cd + 你Docker Desktop Installer.exe下载到的地址。比如，我本机的命令为：

```
cd C:\Users\ASUS\Downloads
```

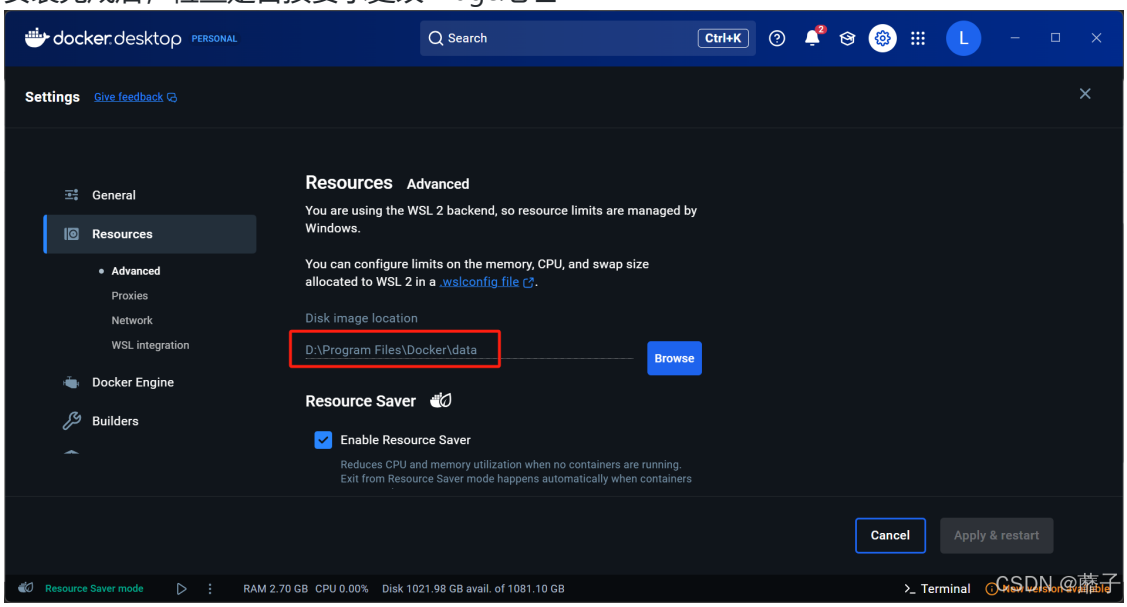


3. 在命令提示符内运行下面代码，并根据你的安装需求做出修改：

```
start /w “” “Docker Desktop Installer.exe” install -accept-license --installation-dir=“D:\Program Files\Docker” --wsl-default-data-root=“D:\Program Files\Docker\data” --windows-containers-default-data-root=“D:\Program Files\Docker”
```

其中，D:\Program Files\Docker 是Docker Desktop 安装程序的安装目录，D:\Program Files\Docker\data是Docker存放是Docker 用于存储镜像、容器、卷等数据的目录。
D:\Program Files\Docker，表示在 Windows 操作系统中，Docker 程序将被安装在 D 盘的 Program Files 文件夹内的一个名为 Docker 的子文件夹中，记得双斜杠。
在运行代码前，一定要提前手动创建好对应文件夹，不然会报错。

4. 安装完成后，检查是否按要求更改image地址



可以看到，docker已经装在D盘目标目录内

版权声明：本文为CSDN博主「蘑子」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
原文链接：https://blog.csdn.net/m0_51290571/article/details/144635357

高级参数

高级参数

logo替换

```
config = {
    "editorConfig": {
        "customization": {
            "logo": {
                "image": "https://example.com/logo.png",
                "imageDark": "https://example.com/dark-logo.png",
                "url": "https://example.com"
            }
        }
    }
};
```

更多参数

https://api.onlyoffice.com/zh/editors/advanced

default.json-文件大小

default.json

位置： `/etc/onlyoffice/documentserver/default.json`

操作实例：
首先将default.json拷贝出来

```
[root@localhost engineercms]# docker cp 826:/etc/onlyoffice/documentserver/default.json /usr/src/engineercms
                                                    Successfully copied 18.9kB to /usr/src/engineercms
```

替换为 `50*1024*1024=524288000`

```
"services": {
        "CoAuthoring": {
            "server": {
                "port": 8000,
                "workerpercpu": 1,
                "mode": "development",
                "limits_tempfile_upload": 524288000,
                "limits_image_size": 26214400,
                "limits_image_download_timeout": {
                    "connectionAndInactivity": "2m",
                    "wholeCycle": "2m"
.....
    "FileConverter": {
        "converter": {
            "maxDownloadBytes": 524288000,
            "downloadTimeout": {
                "connectionAndInactivity": "2m",
                "wholeCycle": "2m"
            },
.....
            "inputLimits": [
                {
                "type": "docx;dotx;docm;dotm",
                "zip": {
                    "uncompressed": "500MB",
```

```
                    "template": "*.xml"
                }
                },
                {
                "type": "xlsx;xltx;xlsm;xltm",
                "zip": {
                    "uncompressed": "500MB",
                    "template": "*.xml"
                }
                },
                {
                "type": "pptx;ppsx;potx;pptm;ppsm;potm",
                "zip": {
                    "uncompressed": "500MB",
                    "template": "*.xml"
                }
                }
```

再拷回容器内

```
[root@localhost engineercms]# docker cp /usr/src/engineercms/default.json 826:/etc/onlyoffice/documentserver/
                                                Successfully copied 18.9kB to 826:/etc/onlyoffice/documentserver/
```

重启容器

```
docker restart 826
```

或者这样操作:
在容器内复制以下脚本执行

```
sed -i -e 's/104857600/10485760000/g' /etc/onlyoffice/documentserver/default.json
sed -i -e 's/50MB/5000MB/g' /etc/onlyoffice/documentserver/default.json
sed -i -e 's/300MB/3000MB/g' /etc/onlyoffice/documentserver/default.json
```

附件

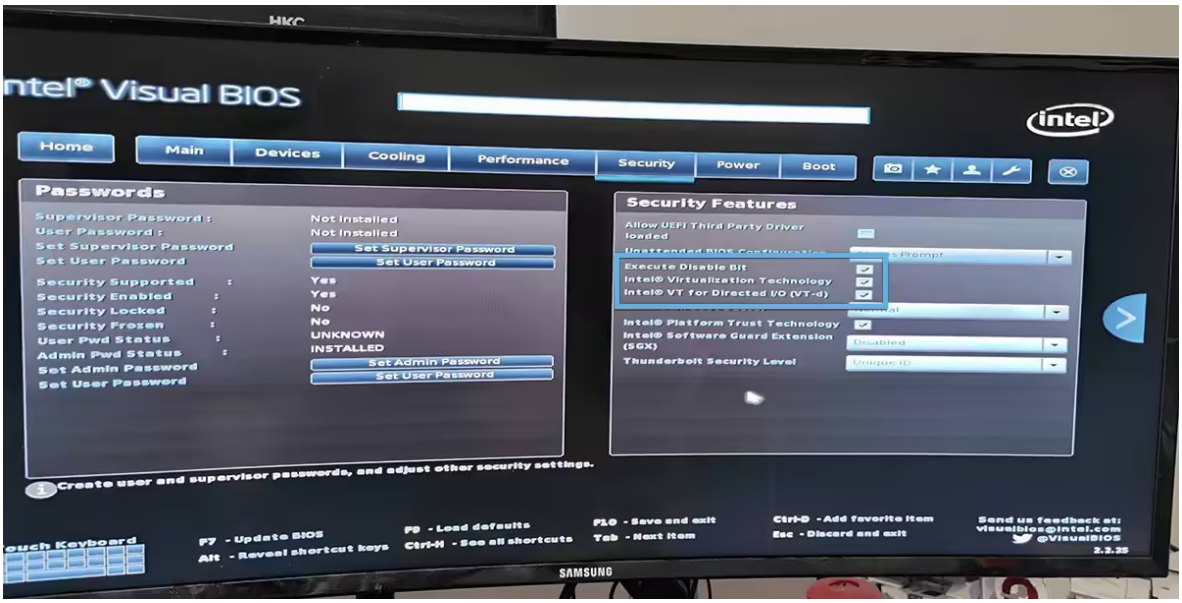
- default.zip

Docker 文档

Docker 文档

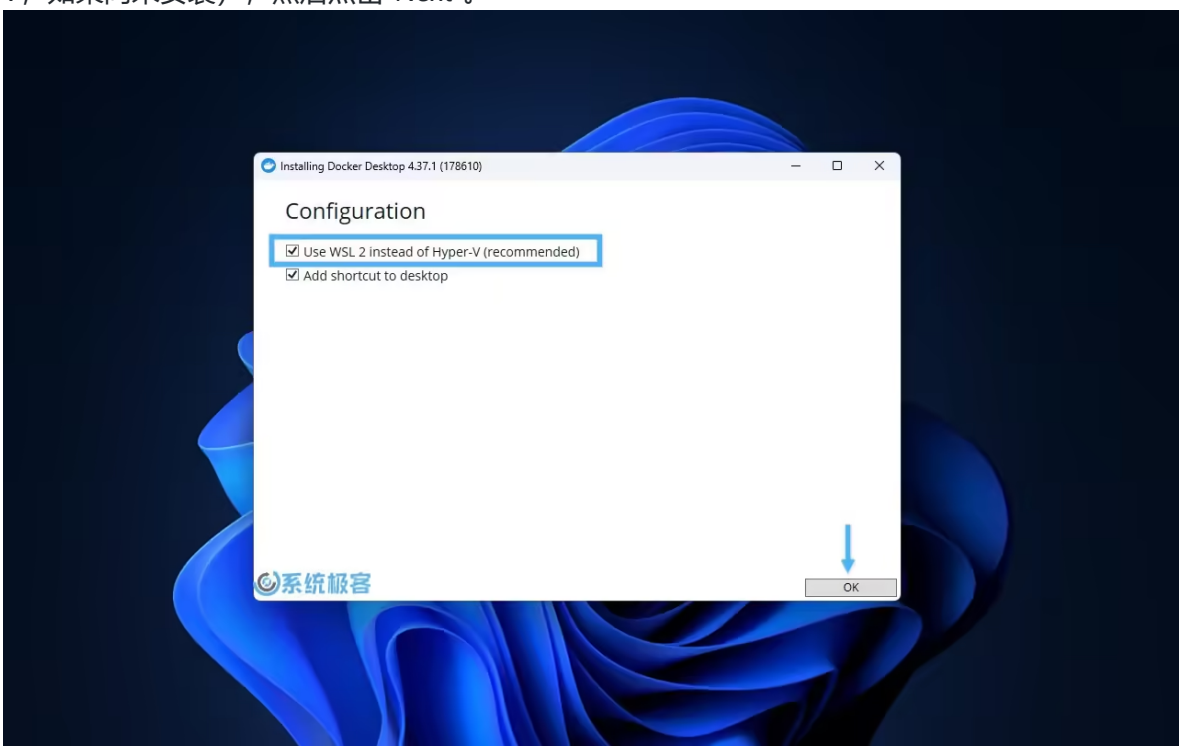
win11安装docker

https://blog.csdn.net/QINGZHID/article/details/144520000
https://zhuanlan.zhihu.com/p/8879067392
https://www.sysgeek.cn/install-docker-desktop-windows-11/
Windows 11 专业版或企业版（家庭版不支持）。
系统必须启用了硬件虚拟化功能。
这通常需要在计算机的 BIOS 或 UEFI 设置中开启，具体的开启方式因计算机型号而异。一般可以在开机时按下对应的按键（如 Del、F2、F10 等）进入 BIOS/UEFI 设置界面，然后找到“Virtualization Technology”或类似选项并将其设置为“Enabled”。
1重启电脑，
按下F2、F10、Delete或Esc进入 BIOS。不同品牌的主板可能有所不同，具体按键可以参考主板说明书或上网搜索。
2根据你的 CPU 型号启用相应的虚拟化选项：
如果是 Intel 处理器，请启用 Intel VT-x
如果是 AMD 处理器，请启用 SVM Mode 或 AMD-V
有些主板可能会统一显示为 CPU Virtualization

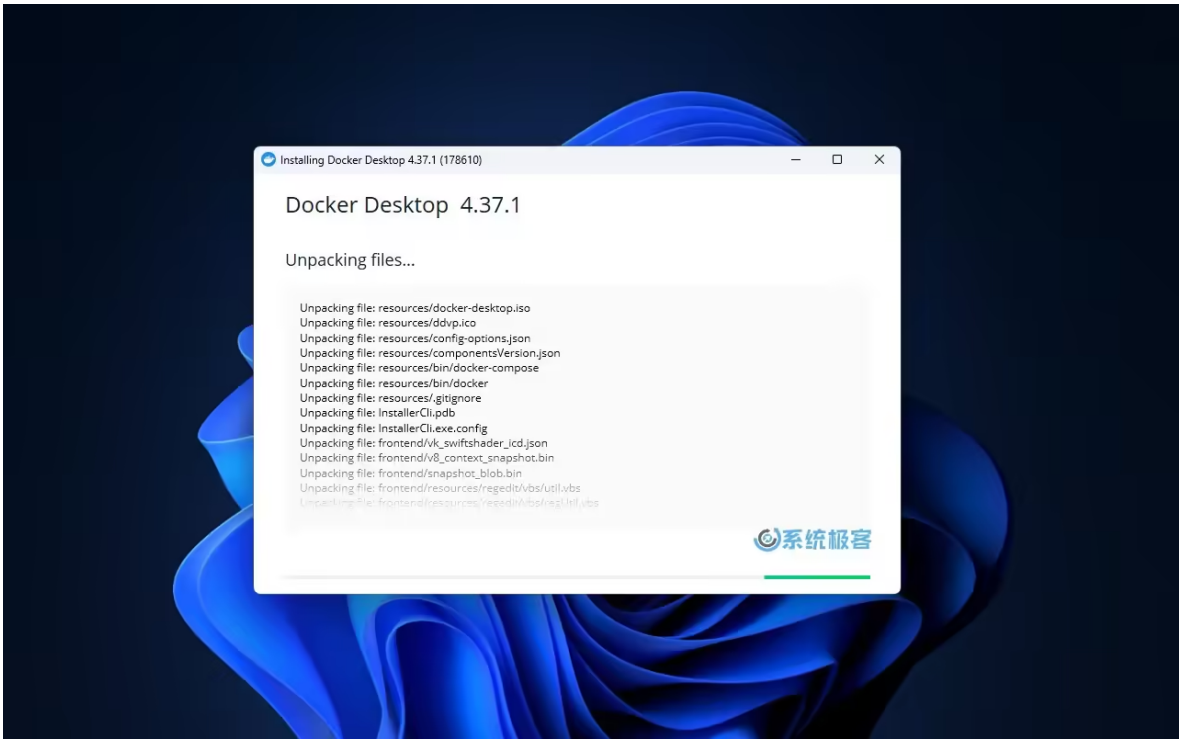


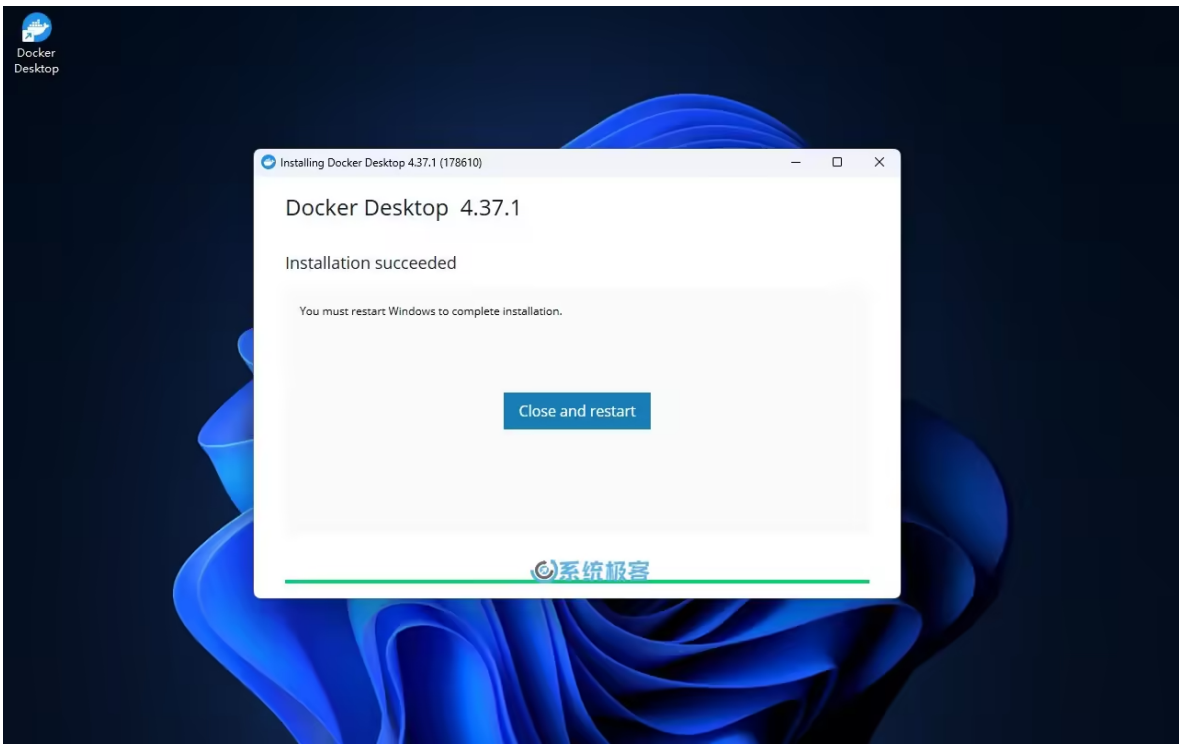
在 BIOS 或 UEFI 中打开硬件虚拟化
最后，记得在退出 BIOS 之前按下F10键保存设置。这样，我们就为 Docker Desktop 配置好了虚拟化环境。
至少拥有 4GB 的系统内存（建议 8GB 或更多以获得更好的性能）。

有足够的磁盘空间用于存储 Docker 镜像、容器以及相关数据。
完成上述系统要求检查后，还需要进行以下准备工作：
确保系统已安装了最新的 Windows 更新。这可以通过点击“开始”菜单，选择“设置”，然后进入“更新和安全”页面，点击“检查更新”按钮来完成。
禁用或卸载可能与 Docker 冲突的软件，如一些虚拟机软件（如 VirtualBox，如果之前安装过）或其他容器化工具。
四、下载 Docker 桌面版
打开浏览器，访问 Docker 官方网站（https://www.docker.com/products/docker-desktop）。
在网站页面上找到适合 Windows 11 的 Docker 桌面版下载链接。通常会有一个明显的“Download for Windows”按钮，点击它开始下载安装程序。
五、安装 Docker 桌面版
下载完成后，找到下载的安装程序文件（一般为.exe 格式），双击运行它。
在安装向导的欢迎页面，点击“Next”按钮继续。
选择安装的目标文件夹。可以使用默认的安装路径，也可以点击“Browse”按钮选择其他磁盘位置，然后点击“Next”。
选择是否创建桌面快捷方式和开始菜单快捷方式，根据个人喜好进行选择后点击“Next”。
在接下来的页面中，会看到关于 Hyper-V 和 Windows 容器的相关设置选项。由于 Windows 11 专业版和企业版支持 Hyper-V，建议选择使用 Hyper-V 作为后端（这将自动安装和配置 Hyper-V，如果尚未安装），然后点击“Next”。



查看安装设置的摘要信息，如果确认无误，点击“Install”按钮开始安装。安装过程可能需要一些时间，期间系统可能会自动重启以完成 Hyper-V 的安装和配置（如果是首次安装 Hyper-V）。





六、配置 Docker 桌面版

安装完成后，Docker 桌面版会自动启动。首次启动时，它会在系统托盘区显示一个小鲸鱼图标。

右键点击该图标，选择“Settings”选项，打开 Docker 设置窗口。

在设置窗口中，可以对 Docker 的各种参数进行配置，如：

General（常规）：可以设置是否在 Windows 启动时自动启动 Docker，以及选择使用的容器运行时（默认是 runc）。

Resources（资源）：配置 Docker 可使用的 CPU 和内存资源比例。可以根据系统的硬件配置和

其他应用的资源需求进行合理调整。例如，如果你的系统同时运行多个大型应用，可以适当减少 Docker 的资源分配，反之则可以增加以提高容器的性能。
Docker Engine（Docker 引擎）：这里可以设置 Docker 引擎的一些高级参数，如镜像仓库地址、容器网络配置等。对于大多数普通用户，使用默认值即可。如果需要使用自定义的镜像仓库或配置特殊的网络模式，可以在这里进行修改。
配置完成后，点击“Apply & Restart”按钮使设置生效并重新启动 Docker 服务。
七、验证安装
打开命令提示符（CMD）或 Windows PowerShell。可以通过在“开始”菜单中搜索“cmd”或“PowerShell”来找到并打开它们。
在命令提示符或 PowerShell 中输入以下命令：
`docker --version`
如果安装成功，将会显示 Docker 的版本信息，例如：
`Docker version 20.10.12, build e91ed57`
接着输入以下命令来运行一个简单的测试容器：
`docker run hello-world`
如果一切正常，将会看到类似于以下的输出：

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:7f0a9f93b4aa3022c3a4c147a449bf11e0941a1fd0bf4a8e6c9408b260
0777c5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctl
y.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs
the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to theDocker client, which sen
t it
    to your terminal.

To try something more ambitious, you can run an image that runs a full Li
nux
distribution as its "rootfs". Try this by running the following:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
```

```
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

这表明 Docker 已经成功安装并且能够正常运行容器。

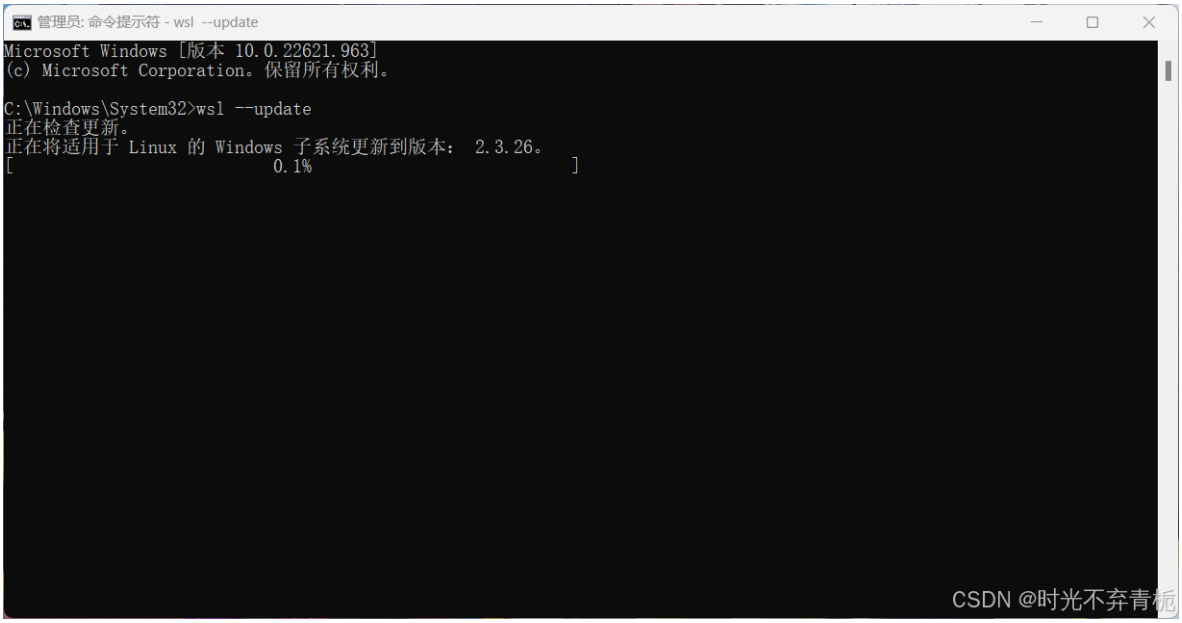
通过以上详细的步骤，你已经在 Windows 11 系统上成功搭建了 Docker 桌面版，并在 WSL2 的支持下，能够更好地利用 Docker 进行容器化应用的开发、测试和部署等工作，享受容器化技术带来的便利与高效。

如果你出现的是这个界面，说明是WSL版本不够新，需要手动安装一下



CSDN @时光不弃青栀

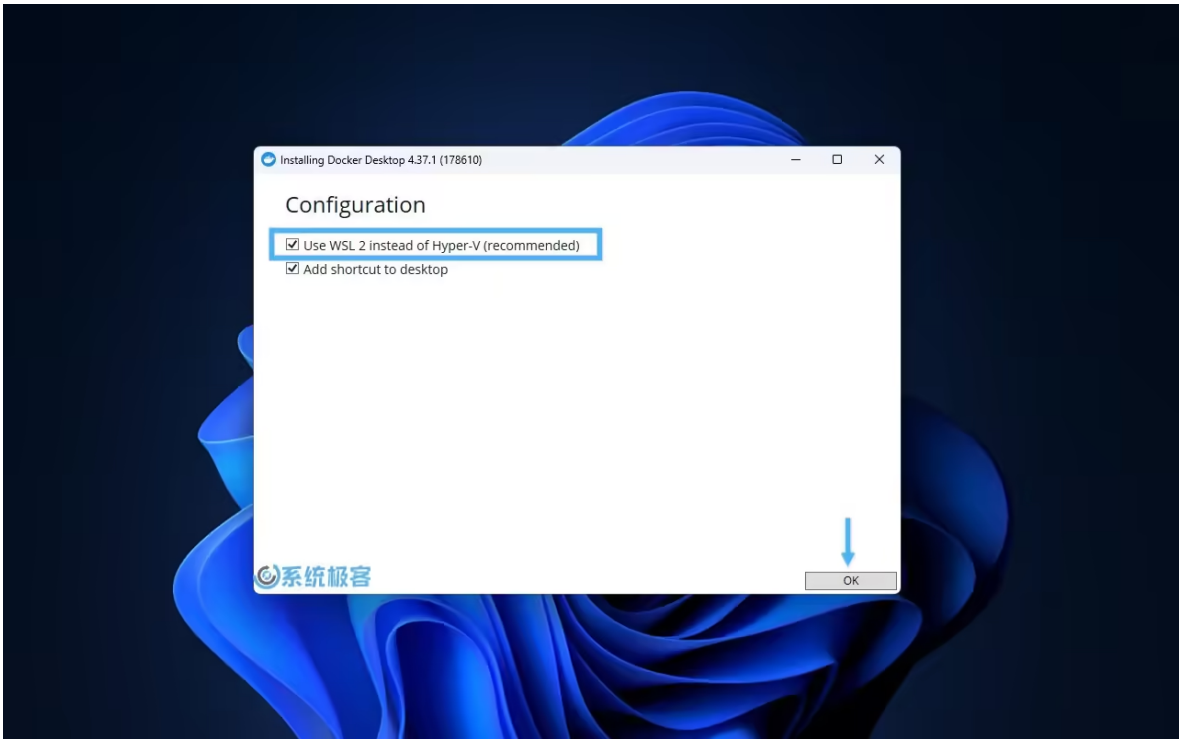
以管理员打开命令行，输入wsl –update进行手动更新，耐心等待成功即可，但是可能会很慢，建议使用我提供的安装包直接安装~



如果输入wsl –update提示的是没有适用的子系统，也可以下载安装包后直接安装后重启电脑再启动docker即可
WSL下载：
夸克网盘链接：https://pan.quark.cn/s/b4c94cc93772
百度网盘链接: https://pan.baidu.com/s/1IkMpdHzhwYFN3ymNVDTmVA?pwd=yijx 提取码: yijx

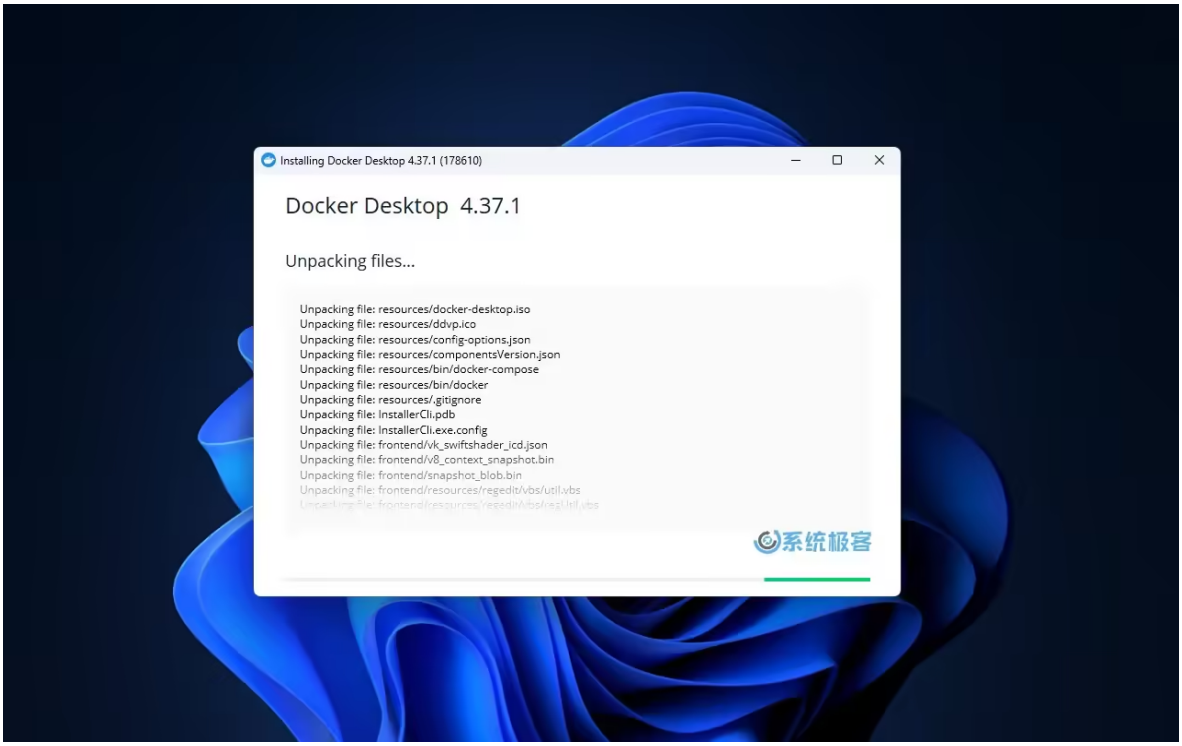
04. 安装 Docker Desktop

1首先，前往 Docker 官网下载「Docker Desktop Installer.exe」安装程序。
2右键点击安装程序，选择「以管理员身份运行」。
3根据你的需求，勾选或取消「Use WSL 2 instead of Hyper-V (recommended)」选项：
勾选：表示使用 WSL 2 后端。
取消：表示使用 Hyper-V 后端。

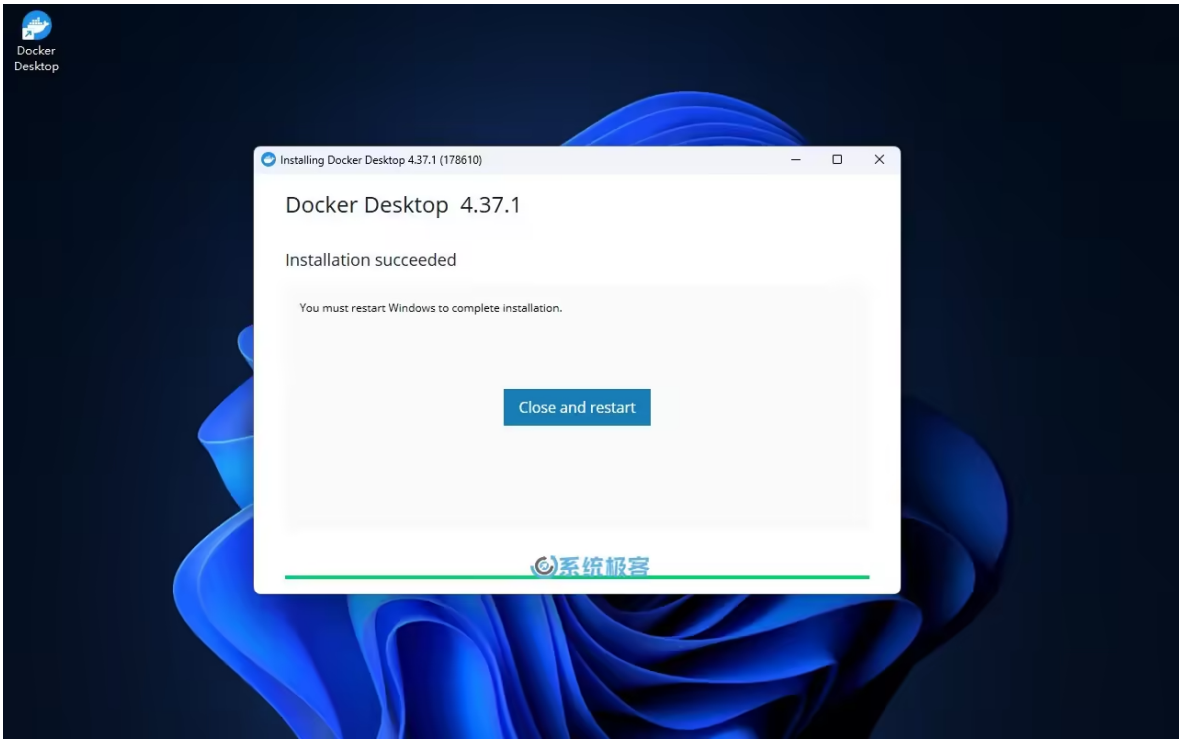


选择是否使用 WSL 2 后端

4点击「OK」开始安装，等待安装完成后，建议重启一下电脑。



安装 Docker Desktop



关闭并重启计算机

05. 初始化 Docker Desktop

1右键点击「开始」菜单，选择「终端管理员」，以管理员权限打开 Windows 终端。

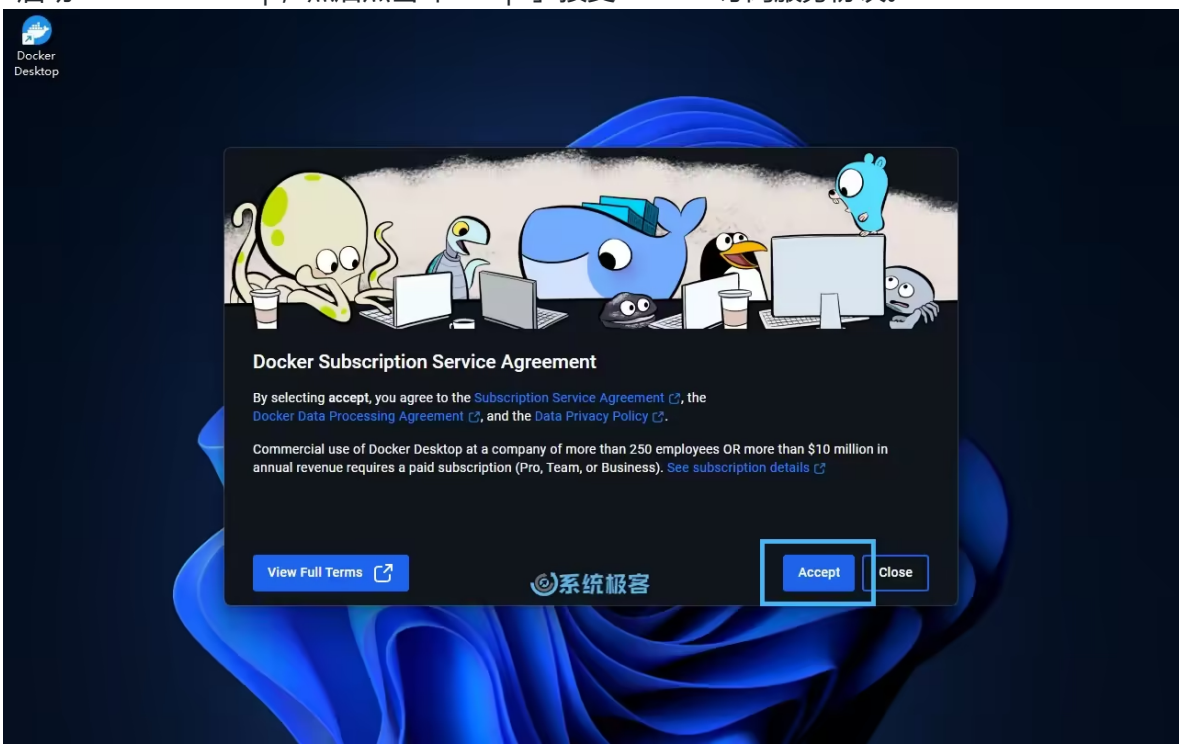
2执行以下命令，先手动更新一下 WSL 版本：

```
wsl --update
```



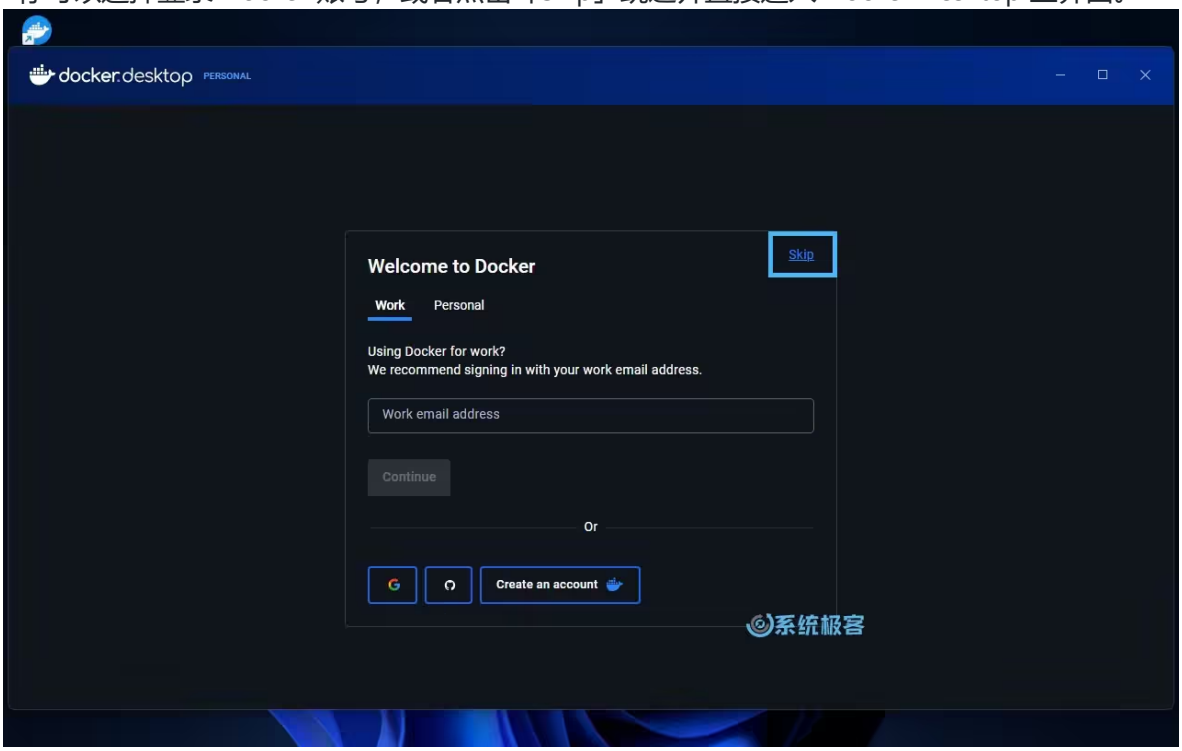
更新 WSL 版本

3启动 Docker Desktop，然后点击「Accept」接受 Docker 订阅服务协议。

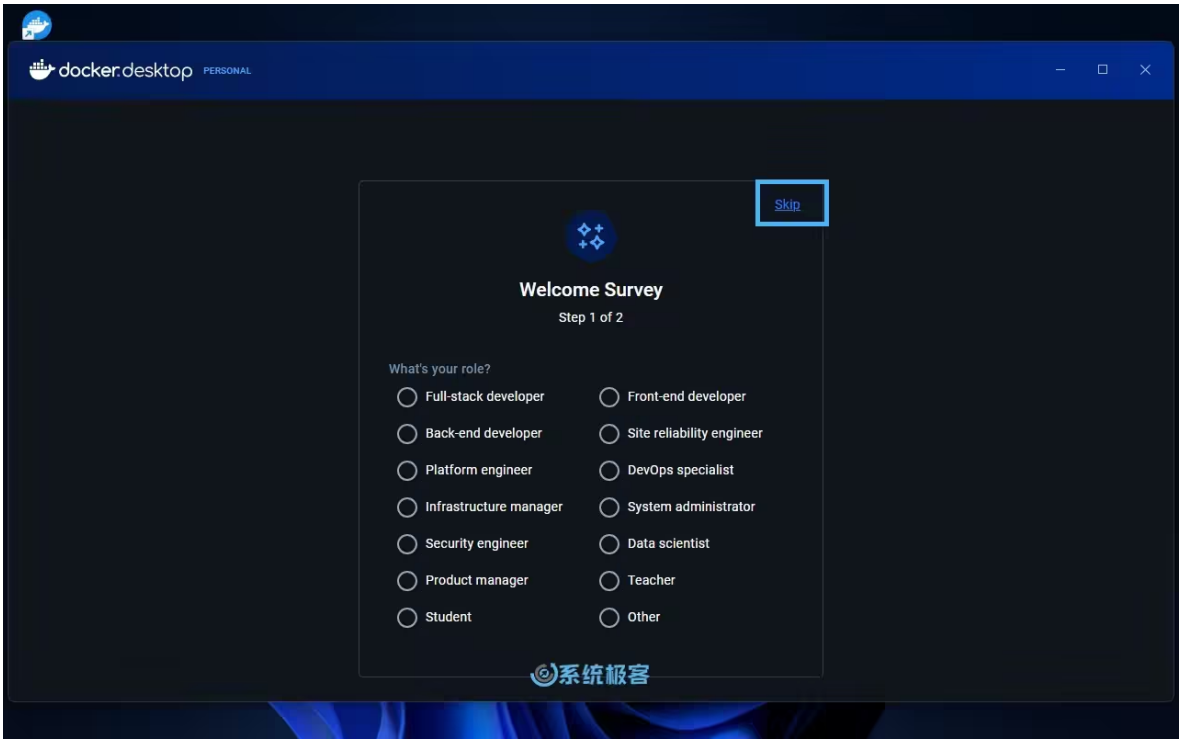


接受 Docker 订阅服务协议

4你可以选择登录 Docker 账号，或者点击「Skip」跳过并直接进入 Docker Desktop 主界面。

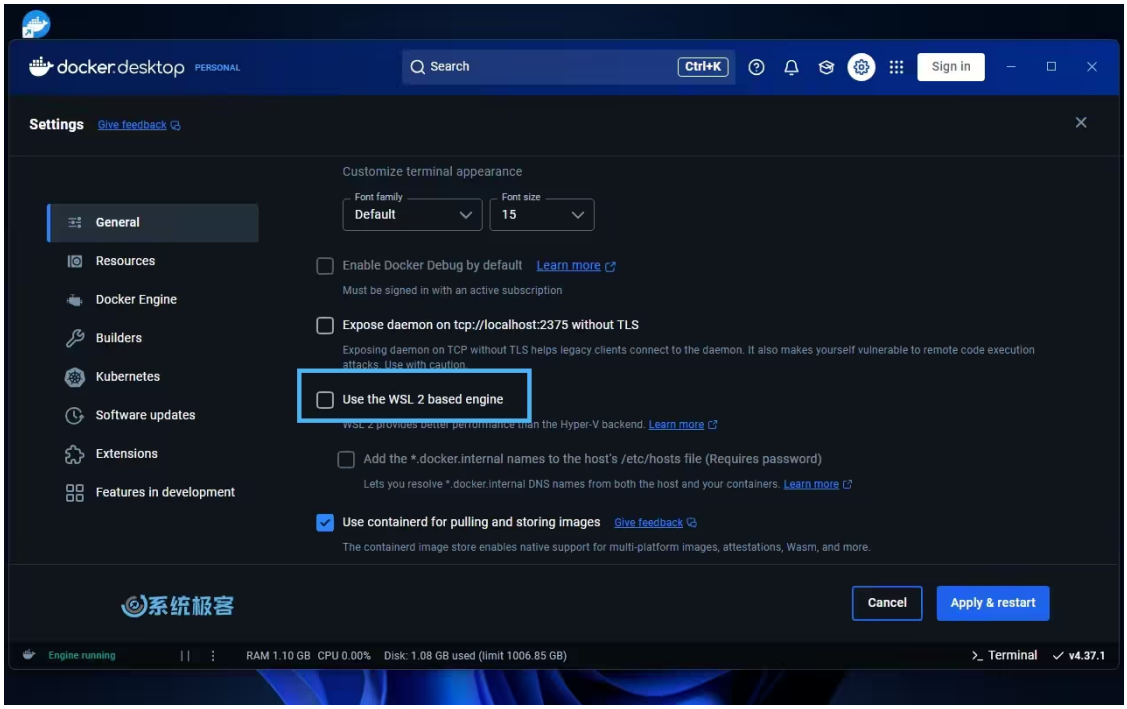


登录 Docker 账户或跳过



选择你的角色或跳过

6. 切换和验证 Docker Desktop 后端
   6.1 切换后端
   你可以根据需要，随时在 WSL 2 和 Hyper-V 后端之间进行切换，操作步骤如下：
   确保 Windows 11 已经安装了后端所需的组件，否则无法从 WSL 2 切换到 Hyper-V。
   1打开 Docker Desktop 管理界面，点击右上角的「设置」图标。
   2在「General」常规选项卡中，勾选（使用）或取消勾选「Use the WSL 2 based engine」选项，以选择是否使用基于 WSL 2 的引擎。
   3点击右下角的「Apply & restart」按钮，重启 Docker Engine。



选择是否使用 WSL 2 引擎

6.2 切换 WSL 发行版关联

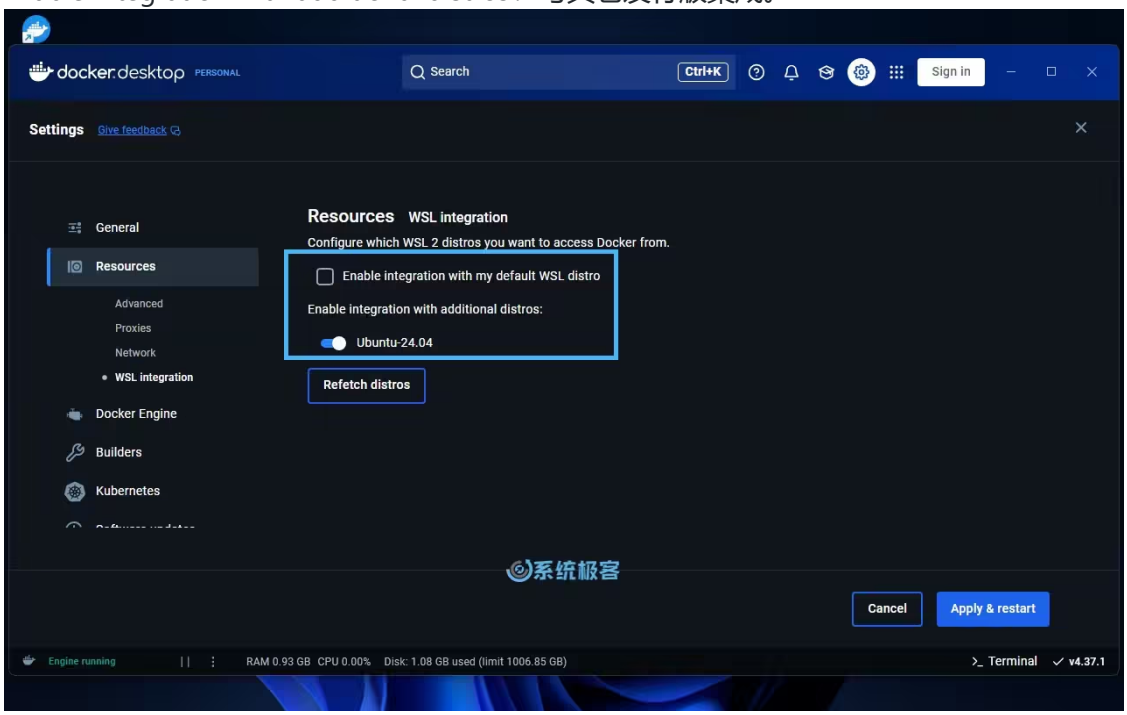
如果你已经在使用某个 WSL 发行版，也可以将它与 Docker 集成：

1打开「设置」>「Resources」>「WSL integration」。

2根据需要，选择要关联的 Linux 发行版。

Enable integration with my default WSL distro：与默认 WSL 发行版成。

Enable integration with additional distros：与其它发行版集成。



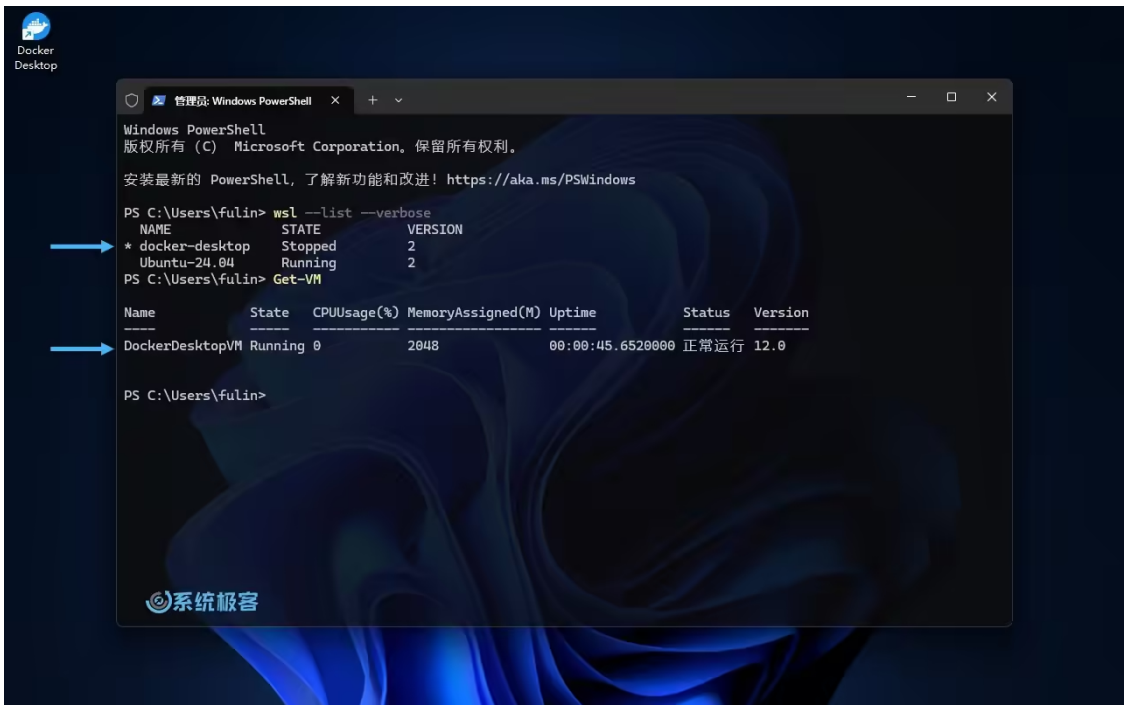
指定要关联的 WSL 发行版

点击右下角的「Apply & restart」按钮，重启 Docker Engine。

6.3 验证后端

在使用 WSL 2 或 Hyper-V 后端时，分别会有一个名为 docker-desktop 的 Linux 子系统（WSL 2 默认创建）或名为DockerDesktopVM的虚拟机在后台运行。你可以使用以下命令进行查看：

```
wsl --list --verbose  # 查看 Linux 子系统的运行状态
Get-VM                # 查看 Hyper-V 虚拟机的运行状态（PowerShell）
```



查看 Docker Desktop 后端 Hyper-V 虚拟机/WSL 子系统

现在，Docker Desktop 已经准备就绪，你可以开始搜索并运行你感兴趣的容器镜像了。

介绍

概要

- 允许您在本地服务器上安装并将在线编辑器与您的 Web 应用程序集成。
- 无限制连接数
- 支持移动端
- 镜像地址

功能性

- 文档编辑器
- 电子表格编辑器
- 演示文稿编辑器

- 移动网络编辑器
- 协作编辑
- 支持所有流行格式：DOC、DOCX、TXT、ODT、RTF、ODP、EPUB、ODS、XLS、XLSX、CSV、PPTX、HTML

服务器配置

- CPU 双核 2 GHz或更高
- 内存 2 GB或更多
- 硬盘至少 40 GB的可用空间
- 内核版本为3.10或更高版本的amd64 Linux 发行版
- 其他要求：
- Docker：版本1.10或更高版本

安装

::: tip 温馨提示
您需要安装最新的 Docker 版本。如果您没有它，请参阅 Docker 网站上的安装部分以了解如何获取它。
:::

安装 Docker 后，运行它并执行以下命令：

```
sudo docker run --name=ooffice -i -t -d -p 8080:80 --restart=always kn
oxzhang/oo-ce-docker-license
```

挂载数据

```
sudo docker run --name=ooffice -i -t -d -p 8080:80 --restart=always \
  -v /app/onlyoffice/DocumentServer/logs:/var/log/onlyoffice  \
  -v /app/onlyoffice/DocumentServer/data:/var/www/onlyoffice/Data  \
  -v /app/onlyoffice/DocumentServer/lib:/var/lib/onlyoffice \
  -v /app/onlyoffice/DocumentServer/db:/var/lib/postgresql \
  knoxzhang/oo-ce-docker-license
```

可用的配置参数

下面是可以使用环境变量设置的参数的完整列表

- **ONLYOFFICE_HTTPS_HSTS_ENABLED** : 用于关闭 HSTS 配置的高级配置选项。仅在使用 SSL 时适用。默认为true
- **ONLYOFFICE_HTTPS_HSTS_MAXAGE** : 用于在 onlyoffice NGINX vHost 配置中设置 HSTS max-age 的高级配置选项。仅在使用 SSL 时适用。默认为31536000
- **SSL_CERTIFICATE_PATH** : 要使用的 SSL 证书的路径。默认为/var/www/onlyoffice/Data/certs/tls.crt
- **SSL_KEY_PATH** : SSL 证书私钥的路径。默认为/var/www/onlyoffice/Data/certs/tls.key
- **SSL_DHPARAM_PATH** : Diffie-Hellman 参数的路径。默认为/var/www/onlyoffice/Data/certs/dhparam.pem
- **SSL_VERIFY_CLIENT** : 启用使用CA_CERTIFICATES_PATH file. 默认为false
- **DB_TYPE** : 数据库类型 支持的值为postgres,mariadb或mysql 默认为postgres
- **DB_HOST** : 运行数据库服务器的主机的 IP 地址或名称
- **DB_PORT** : 数据库服务器端口号
- **DB_NAME** : 要在映像启动时创建的数据库的名称
- **DB_USER** : 具有数据库帐户超级用户权限的新用户名
- **DB_PWD** : 为数据库帐户设置的密码
- **AMQP_URI** : 连接到消息代理服务器的AMQP URI
- **AMQP_TYPE** : 消息代理类型。支持的值为rabbitmq或activemq。默认为rabbitmq
- **REDIS_SERVER_HOST** : 运行 Redis 服务器的主机的 IP 地址或名称
- **REDIS_SERVER_PORT** : Redis 服务器端口号
- **NGINX_WORKER_PROCESSES** : 定义 NGINX 工作进程的数量
- **NGINX_WORKER_CONNECTIONS** : 设置 NGINX 工作进程可以同时打开的最大连接数
- **JWT_ENABLED** : 指定ONLYOFFICE Docs启用 JSON Web 令牌验证。默认为false
- **JWT_SECRET** : 定义密钥以验证对ONLYOFFICE Docs的请求中的 JSON Web 令牌。默认为secret
- **JWT_HEADER** : 定义将用于发送 JSON Web 令牌的 HTTP 标头。默认为Authorization
- **JWT_IN_BODY** : 指定在ONLYOFFICE Docs的请求正文中启用令牌验证。默认为false
- **USE_UNAUTHORIZED_STORAGE** : true 如果您的存储服务器使用自签名证书，则设置为，例如 Nextcloud。默认为false
- **GENERATE_FONTS** : true 每次开始时，重新生成字体列表和字体缩略图等。默认为true
- **METRICS_ENABLED** : 为ONLYOFFICE Docs指定启用 StatsD 默认为false
- **METRICS_HOST** : 定义 StatsD 监听主机。默认为localhost

- **METRICS_PORT** : 定义 StatsD 监听端口。默认为8125
- **METRICS_PREFIX** : 定义后端服务的 StatsD 指标前缀。默认为ds..
- **LETS_ENCRYPT_DOMAIN** : 定义 Let's Encrypt 证书的域
- **LETS_ENCRYPT_MAIL** : 定义 Let's Encrypt 证书的域管理员邮件地址

```
# 启用令牌
sudo docker run --name=ooffice -i -t -d -p 8080:80 --restart=always \
  -e JWT_ENABLED=true \
  knoxzhang/oo-ce-docker-license
```

前端集成

前端集成

HTML 标签

```
<div id="placeholder"></div>
<script type="text/javascript" src="https://documentserver/web-apps/apps/api/documents/api.js"></script>
```

可变参数的页面代码

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

config 是一个对象

```
config = {
    "document": {
        "fileType": "docx",
        "key": "Khirz6zTPdfd7",
        "title": "Example Document Title.docx",
        "url": "https://example.com/url-to-example-document.docx"
    },
    "documentType": "word",
    "editorConfig": {
        "callbackUrl": "https://example.com/url-to-callback.ashx"
    }
};
```

完整示例

```
<!DOCTYPE html>
<html style="height: 100%;">
<head>
```

```
    <title>ONLYOFFICE Api Documentation</title>
</head>
<body style="height: 100%; margin: 0;">

    <div id="placeholder" style="height: 100%"></div>
    <script type="text/javascript" src="https://doc.onlyoffice.com/web-aps/apps/api/documents/api.js"></script>

    <script type="text/javascript">

        window.docEditor = new DocsAPI.DocEditor("placeholder",
            {
                "document": {
                    "fileType": "docx",
                    "key": "apiwhec97b7f2-9f8b-4785-a68a-439de62e670c",
                    "permissions": {},
                    "title": "Example Document Title.docx",
                    "url": "https:\/\/d2nlctn12v279m.cloudfront.net\/assets\/docs\/samples\/demo.docx"
                },
                "documentType": "word", "editorConfig": {
                    "callbackUrl": "https:\/\/api.onlyoffice.com\/editors\/callback",
                    "customization": {
                        "anonymous": {
                            "request": false
                        }
                    }
                },
                "height": "100%",
                "token": "ew0KICAidHlwIjogIkpXVCIsDQogICJhbGciOiAiSFMyNTYiDQp9.ew0KICAiZG9jdW1lbnQiOiB7DQogICAgImZpbGVUeXBlIjogImRvY3giLA0KICAgICJrZXkiOiAiYXBpd2hlYzk3YjdmMi05ZjhiLTQ3ODUtYTY4YS00MzlkZTYyZTY3MGMiLA0KICAgICJwZXJtaXNzaW9ucyI6IHt9LA0KICAgICJ0aXRsZSI6ICJFeGFtcGxlIERvY3VtZW50IFRpdGxlLmRvY3giLA0KICAgICJ1cmwiOiAiaHR0cHM6Ly9kMm5sY3RuMTJ2Mjc5bS5jbG91ZGZyb250Lm5ldC9hc3NldHMvZG9jcy9zYW1wbGVzL2RlbW8uZG9jeCINCiAgfSwNCiAgImRvY3VtZW50VHlwZSI6ICJ3b3JkIiwNCiAgImVkaXRvckNvbmZpZyI6IHsNCiAgICAiY2FsbGJhY2tVcmwiOiAiaHR0cHM6Ly9hcGkub25seW9mZmljZS5jb20vZWRpdG9ycy9jYWxsYmFjayIsDQogICAgImN1c3RvbWl6YXRpb24iOiB7DQogICAgICAiYW5vbnltb3VzIjogew0KICAgICAgICAicmVxdWVzdCI6IGZhbHNlDQogICAgICB9DQogICAgfQ0KICB9LA0KICAiaGVpZ2h0IjogIjEwMCUiLA0KICAid2lkdGgiOiAiMTAwJSINCn0.pVqCERdbkcvbl6s8W-0k8QngLRCYNyhW0IB8i7JxWwk",
                "width": "100%"
            }
        );
```

```
    </script>
</body>
</html>
```

https://vitepress.dev/reference/default-theme-home-page

https://vitepress.dev/reference/default-theme-home-page

layout: home

hero:
name: “ONLYOFFICE”
text: “为开发人员”

tagline: 简单、强大、快速、好用

actions:

```
- theme: brand
  text: 快速查看
  link: /integration/
# - theme: alt
#   text: API Examples
#   link: /api-examples
```

features:

- title: 文档

  details: 集成、服务部署、字体

  link: /integration/
- title: 基础镜像

  details: 免费下载、无限制

  link: /paid-custom-free/8.0.0
- title: Pro 镜像

  details: 无限制、移动端、连接器、文档生成器

  link: /paid-custom-image/8.0.1/

徽标


Docker Pulls

NPM Downloads by package author

NPM Downloads by package author

人数

React

React

安装

```
npm install --save @onlyoffice/document-editor-react
```

or

```
yarn add @onlyoffice/document-editor-react -S
```

使用示例

```
import { DocumentEditor } from "@onlyoffice/document-editor-react";
var onDocumentReady = function (event) {
    console.log("Document is loaded");
};

<DocumentEditor
  id="docxEditor"
  documentServerUrl="http://documentserver/"
  config={{
      "document": {
          "fileType": "docx",
      "key": "Khirz6zTPdfd7",
      "title": "Example Document Title.docx",
      "url": "https://example.com/url-to-example-document.docx"
    },
    "documentType": "word",
    "editorConfig": {
        "callbackUrl": "https://example.com/url-to-callback.ashx"
    }
  }}
  events_onDocumentReady={onDocumentReady} />
```

Props

名称	类型	默认	必填	描述
`id`	string	null	true	组件唯一
`documentServerUrl`	string	null	true	文档服务
`config`	object	null	true	用于打开
`document_fileType`	string	null	false	文件类型
`document_title`	string	null	false	文件名称
`documentType`	string	null	false	文档类型
`height`	string	null	false	窗口高度
`type`	string	null	false	终端类型
`width`	string	null	false	窗口宽度
`events_onAppReady`	(event: object) => void	null	false	应用初始
`events_onDocumentStateChange`	(event: object) => void	null	false	文档修改
`events_onMetaChange`	(event: object) => void	null	false	通过met
`events_onDocumentReady`	(event: object) => void	null	false	文档加载
`events_onInfo`	(event: object) => void	null	false	应用程序
`events_onWarning`	(event: object) => void	null	false	出现警告

名称	类型	默认	必填	描述
events_onError	(event: object) => void	null	false	出现错误
events_onRequestSharingSettings	(event: object) => void	null	false	点击置管
events_onRequestRename	(event: object) => void	null	false	点击重命
events_onMakeActionLink	(event: object) => void	null	false	点击书签
events_onRequestInsertImage	(event: object) => void	null	false	点击插入
events_onRequestSaveAs	(event: object) => void	null	false	点击另存
events_onRequestMailMergeRecipients	(event: object) => void	null	false	点击选择
events_onRequestCompareFile	(event: object) => void	null	false	点击文档
events_onRequestEditRights	(event: object) => void	null	false	点击编辑
events_onRequestHistory	(event: object) => void	null	false	点击历史
events_onRequestHistoryClose	(event: object) => void	null	false	关闭历史

名称	类型	默认	必填	描述
`events_onRequestHistoryData`	(event: object) => void	null	false	点击特定
`events_onRequestRestore`	(event: object) => void	null	false	点击恢复

Vue 2.x

Vue 2.x

安装

```
npm install --save document-editor-vue
```

or

```
yarn add document-editor-vue -S
```

使用示例

```
<template>
  <DocumentEditor
    id="docEditor"
    :documentServerUrl="documentServerUrl"
    :config="config"
    :events_onDocumentReady="onDocumentReady"
    />
</template>

<script>
import { DocumentEditor } from "document-editor-vue";

export default {
  name: 'ExampleComponent',
  components: {
    DocumentEditor
  },
  data() {
    return {
      documentServerUrl:"http://10.10.31.157:7008/web-apps/apps/api/documents/api.js",
      config: {
        document: {
          fileType: "docx",
```

```
          key: "Khirz6zTPdfd7",
          title: "Example Document Title.docx",
          url: "https://example.com/url-to-example-document.docx"
        },
        documentType: "word",
        editorConfig: {
          callbackUrl: "https://example.com/url-to-callback.ashx"
        }
      }
    }
  },
  methods: {
    onDocumentReady() {
      console.log("Document is loaded");
    }
  },
};
</script>
```

Props

名称	类型	默认	必填	描述
`id`	string	null	true	组件唯一
`documentServerUrl`	string	null	true	文档服务
`config`	object	null	true	用于打开
`document_fileType`	string	null	false	文件类型
`document_title`	string	null	false	文件名称
`documentType`	string	null	false	文档类型
`height`	string	null	false	窗口高度
`type`	string	null	false	终端类型
`width`	string	null	false	窗口宽度
`events_onAppReady`	(event: object) => void	null	false	应用初始

名称	类型	默认	必填	描述
`events_onDocumentStateChange`	(event: object) => void	null	false	文档修改
`events_onMetaChange`	(event: object) => void	null	false	通过me
`events_onDocumentReady`	(event: object) => void	null	false	文档加载
`events_onInfo`	(event: object) => void	null	false	应用程序
`events_onWarning`	(event: object) => void	null	false	出现警告
`events_onError`	(event: object) => void	null	false	出现错误
`events_onRequestSharingSettings`	(event: object) => void	null	false	点击置管
`events_onRequestRename`	(event: object) => void	null	false	点击重命
`events_onMakeActionLink`	(event: object) => void	null	false	点击书签
`events_onRequestInsertImage`	(event: object) => void	null	false	点击插入
`events_onRequestSaveAs`	(event: object) => void	null	false	点击另存

名称	类型	默认	必填	描述
`events_onRequestMailMergeRecipients`	(event: object) => void	null	false	点击选择
`events_onRequestCompareFile`	(event: object) => void	null	false	点击文档
`events_onRequestEditRights`	(event: object) => void	null	false	点击编辑
`events_onRequestHistory`	(event: object) => void	null	false	点击历史
`events_onRequestHistoryClose`	(event: object) => void	null	false	关闭历史
`events_onRequestHistoryData`	(event: object) => void	null	false	点击特定
`events_onRequestRestore`	(event: object) => void	null	false	点击恢复

Vue 3.x

Vue 3.x

安装

```
npm install --save @onlyoffice/document-editor-vue
```

or

```
yarn add @onlyoffice/document-editor-vue -S
```

使用示例

```
<template>
  <DocumentEditor
    id="docEditor"
    documentServerUrl="http://documentserver/"
    :config="config"
    :events_onDocumentReady="onDocumentReady"
    />
</template>

<script lang="ts">
import { defineComponent } from 'vue';
import { DocumentEditor } from "@onlyoffice/document-editor-vue";

export default defineComponent({
  name: 'ExampleComponent',
  components: {
    DocumentEditor
  },
  data() {
    return {
      config: {
        document: {
          fileType: "docx",
          key: "Khirz6zTPdfd7",
```

```
          title: "Example Document Title.docx",
          url: "https://example.com/url-to-example-document.docx"
        },
        documentType: "word",
        editorConfig: {
          callbackUrl: "https://example.com/url-to-callback.ashx"
        }
      }
    }
  },
  methods: {
    onDocumentReady() {
      console.log("Document is loaded");
    }
  },
});
</script>
```

Props

名称	类型	默认	必填	描述
`id`	string	null	true	组件唯一
`documentServerUrl`	string	null	true	文档服务
`config`	object	null	true	用于打开
`document_fileType`	string	null	false	文件类型
`document_title`	string	null	false	文件名称
`documentType`	string	null	false	文档类型
`height`	string	null	false	窗口高度
`type`	string	null	false	终端类型
`width`	string	null	false	窗口宽度
`events_onAppReady`	(event: object) => void	null	false	应用初始

名称	类型	默认	必填	描述
`events_onDocumentStateChange`	(event: object) => void	null	false	文档修改
`events_onMetaChange`	(event: object) => void	null	false	通过me
`events_onDocumentReady`	(event: object) => void	null	false	文档加载
`events_onInfo`	(event: object) => void	null	false	应用程序
`events_onWarning`	(event: object) => void	null	false	出现警告
`events_onError`	(event: object) => void	null	false	出现错误
`events_onRequestSharingSettings`	(event: object) => void	null	false	点击置管
`events_onRequestRename`	(event: object) => void	null	false	点击重命
`events_onMakeActionLink`	(event: object) => void	null	false	点击书签
`events_onRequestInsertImage`	(event: object) => void	null	false	点击插入
`events_onRequestSaveAs`	(event: object) => void	null	false	点击另存

名称	类型	默认	必填	描述
`events_onRequestMailMergeRecipients`	(event: object) => void	null	false	点击选择
`events_onRequestCompareFile`	(event: object) => void	null	false	点击文档
`events_onRequestEditRights`	(event: object) => void	null	false	点击编辑
`events_onRequestHistory`	(event: object) => void	null	false	点击历史
`events_onRequestHistoryClose`	(event: object) => void	null	false	关闭历史
`events_onRequestHistoryData`	(event: object) => void	null	false	点击特定
`events_onRequestRestore`	(event: object) => void	null	false	点击恢复

开启token验证

开启token验证

```
 docker run --name=ooffice -i -t -d -p 8080:80 -e JWT_ENABLED=true --rest
art=always knoxzhang/oo-ce-docker-license
```

::: tip 温馨提示

1. 7.1 版本之后 生成的 token的入参与前端集成的 config 格式要严格保持一致
2. 7.2 版本之后 默认开启 jwt 验证，去关闭
   :::

例如

config

```
    {
        "document": {
            "fileType": "docx",
            "key": "apiwhec97b7f2-9f8b-4785-a68a-439de62e670c",
            "permissions": {},
            "title": "Example Document Title.docx",
            "url": "https:\/\/d2nlctn12v279m.cloudfront.net\/assets\/docs
\/samples\/demo.docx"
        },
        "documentType": "word",
        "editorConfig": {
            "callbackUrl": "https:\/\/api.onlyoffice.com\/editors\/callba
ck",
            "customization": {
                "anonymous": {
                    "request": false
                }
            }
        },
        "height": "100%",
        "token": "ew0KICAidHlwIjogIkpXVCIsDQogICJhbGciOiAiSFMyNTYiDQp9.ew
0KICAiZG9jdW1lbnQiOiB7DQogICAgImZpbGVUeXBlIjogImRvY3giLA0KICAgICJrZXkiOiA
iYXBpd2hlYzk3YjdmMi05ZjhiLTQ3ODUtYTY4YS00MzlkZTYyZTY3MGMiLA0KICAgICJwZXJt
aXNzaW9ucyI6IHt9LA0KICAgICJ0aXRsZSI6ICJFeGFtcGxlIERvY3VtZW50IFRpdGxlLmRvY
```

```
3giLA0KICAgICJ1cmwiOiAiaHR0cHM6Ly9kMm5sY3RuMTJ2Mjc5bS5jbG91ZGZyb250Lm5ldC
9hc3NldHMvZG9jcy9zYW1wbGVzL2RlbW8uZG9jeCINCiAgfSwNCiAgImRvY3VtZW50VHlwZSI
6ICJ3b3JkIiwNCiAgImVkaXRvckNvbmZpZyI6IHsNCiAgICAiY2FsbGJhY2tVcmwiOiAiaHR0
cHM6Ly9hcGkub25seW9mZmljZS5jb20vZWRpdG9ycy9jYWxsYmFjayIsDQogICAgImN1c3Rvb
Wl6YXRpb24iOiB7DQogICAgICAiYW5vbnltb3VzIjogew0KICAgICAgICAicmVxdWVzdCI6IG
ZhbHNlDQogICAgICB9DQogICAgfQ0KICB9LA0KICAgICAgICAiaGVpZ2h0IjogIjEwMCUiLA0KICAid2l
kdGgiOiAiMTAwJSINCn0.pVqCERdbkcvbl6s8W-0k8QngLRCYNyhW0IB8i7JxWwk",
        "width": "100%"
    }
```

token

```
 {
        "document": {
            "fileType": "docx",
            "key": "apiwhec97b7f2-9f8b-4785-a68a-439de62e670c",
            "permissions": {},
            "title": "Example Document Title.docx",
            "url": "https:\/\/d2nlctn12v279m.cloudfront.net\/assets\/docs
\/samples\/demo.docx"
        },
        "documentType": "word",
        "editorConfig": {
            "callbackUrl": "https:\/\/api.onlyoffice.com\/editors\/callba
ck",
            "customization": {
                "anonymous": {
                    "request": false
                }
            }
        },
        "height": "100%",
        "width": "100%"
    }
```

关闭token验证

载入镜像
`docker load < onlyoffice.tar`
运行镜像，变成容器

```
docker run --name=ooffice -i -t -d -p 9000:80 -e JWT_ENABLED=false --rest
art=always knoxzhang/oo-ce-docker-license
```

免费镜像

免费镜像

base 镜像 （￥ 0.00）

::: tip 温馨提示 基于 7.1.1.23 镜像制作 :::

包含功能

1. 去除 20 链接数限制  online-users 点击放大
2. 移动端可编辑  mobile 点击放大

联系方式

QQ：601424688

下载

去下载

免费镜像8.2.0

#docker

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license:8.2.0.1
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always knoxzhang/
oo-ce-docker-license:8.2.0.1
```

#百度网盘

```
链接：https://pan.baidu.com/s/1wfUHVDrM1fablcFV301W3Q 提取码：avvp 复制这段
内容后打开百度网盘手机App，操作更方便哦
```

去下载

免费镜像8.0.0

免费镜像

base 镜像 （￥ 0.00）

::: tip 温馨提示
基于 8.0.0 镜像制作
:::

包含功能

1. 去除 20 链接数限制
    online-users
   点击放大

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license:8.0.0.1
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always knoxzhang/
oo-ce-docker-license:8.0.0.1
```

去下载

联系方式

QQ：601424688

免费镜像7.5.1.23

免费镜像

base 镜像 （￥ 0.00）

::: tip 温馨提示
基于 7.5.1 镜像制作
:::

包含功能

1. 去除 20 链接数限制
    online-users
   点击放大

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license:7.5.1.23
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always knoxzhang/
oo-ce-docker-license:7.5.1.23
```

去下载

联系方式

QQ：601424688

免费镜像7.1.1.23

免费镜像

base 镜像 （￥ 0.00）

::: tip 温馨提示
基于 7.1.1.23 镜像制作
:::

包含功能

1. 去除 20 链接数限制
    online-users
   点击放大

2. 移动端可编辑
    mobile
   点击放大

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license:7.1.1.23
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always -e JWT_ENA
BLED=false knoxzhang/oo-ce-docker-license:7.1.1.23
```

去下载

联系方式

QQ：601424688

如何配置代理

如何配置代理

如何使用

1. 安装 NGINX
2. 删除默认配置文件 `/etc/nginx/conf.d/default.conf` 、 `/etc/nginx/sites-enabled/default`
3. 复制配置文件放到 `/etc/nginx/conf.d/` 路径下
4. 重启 NGINX `sudo service nginx reload`

常见场景

当您需要简单的将流量重定向到指定服务器时，使用此方案：
`default.conf` ：配置文件

`backendserver-address` ：指定服务器地址或者服务名

```
#Use this example for proxy traffic to the document server running at 'backendserver-address'.

upstream docservice {
  server backendserver-address;
}

map $http_host $this_host {
    "" $host;
    default $http_host;
}

map $http_x_forwarded_proto $the_scheme {
     default $http_x_forwarded_proto;
     "" $scheme;
}

map $http_x_forwarded_host $the_host {
    default $http_x_forwarded_host;
    "" $this_host;
```

```
}

map $http_upgrade $proxy_connection {
  default upgrade;
  "" close;
}

proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $proxy_connection;
proxy_set_header X-Forwarded-Host $the_host;
proxy_set_header X-Forwarded-Proto $the_scheme;
proxy_set_header Host            $proxy_host;
proxy_set_header X-Real-IP       $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;


server {
  listen 0.0.0.0:80;
  listen [::]:80 default_server;
  server_tokens off;

  location / {
    proxy_pass http://docservice;
    proxy_http_version 1.1;
  }
}
```

代理 HTTPS 到 HTTP

`default.conf` ：配置文件

`backendserver-address` ：指定服务器地址或者服务名

```
# Use this example for proxy HTTPS traffic to the document server running
at 'backendserver-address'.
# Replace {{SSL_CERTIFICATE_PATH}} with the path to the ssl certificate f
ile
# Replace {{SSL_KEY_PATH}} with the path to the ssl private key file

upstream docservice {
  server backendserver-address;
}
```

```
map $http_host $this_host {
    "" $host;
    default $http_host;
}

map $http_x_forwarded_proto $the_scheme {
     default $http_x_forwarded_proto;
     "" $scheme;
}

map $http_x_forwarded_host $the_host {
    default $http_x_forwarded_host;
    "" $this_host;
}

map $http_upgrade $proxy_connection {
  default upgrade;
  "" close;
}

proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $proxy_connection;
proxy_set_header X-Forwarded-Host $the_host;
proxy_set_header X-Forwarded-Proto $the_scheme;
proxy_set_header Host            $proxy_host;
proxy_set_header X-Real-IP       $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

## Normal HTTP host
server {
  listen 0.0.0.0:80;
  listen [::]:80 default_server;
  server_name _;
  server_tokens off;

  ## Redirects all traffic to the HTTPS host
  return 301 https://$server_name:443$request_uri;
}

server {
  listen 0.0.0.0:443 ssl;
  listen [::]:443 ssl default_server;
  server_tokens off;
  root /usr/share/nginx/html;
```

```
  ## Strong SSL Security
  ## https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html
  ssl on;
  ssl_certificate {{SSL_CERTIFICATE_PATH}};
  ssl_certificate_key {{SSL_KEY_PATH}};
  ssl_verify_client off;

  ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";

  ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
  ssl_session_cache  builtin:1000  shared:SSL:10m;

  ssl_prefer_server_ciphers   on;

  ## [Optional] Before enabling Strict-Transport-Security headers, ensure
your server is properly configured for SSL.
  ## This directive informs the browser to always use HTTPS. For more inf
o see:
  ## - https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-T
ransport-Security
  # add_header Strict-Transport-Security "max-age=31536000; includeSubDom
ains" always;
  # add_header X-Frame-Options SAMEORIGIN;
  add_header X-Content-Type-Options nosniff;

  ## [Optional] If your certficate has OCSP, enable OCSP stapling to redu
ce the overhead and latency of running SSL.
  ## Replace with your ssl_trusted_certificate. For more info see:
  ## - https://medium.com/devops-programming/4445f4862461
  ## - https://www.ruby-forum.com/topic/4419319
  ## - https://www.digitalocean.com/community/tutorials/how-to-configure-
ocsp-stapling-on-apache-and-nginx
  # ssl_stapling on;
  # ssl_stapling_verify on;
  # ssl_trusted_certificate /etc/nginx/ssl/stapling.trusted.crt;
  # resolver 208.67.222.222 208.67.222.220 valid=300s; # Can change to yo
ur DNS resolver if desired
  # resolver_timeout 10s;

  ## [Optional] Generate a stronger DHE parameter:
  ##   cd /etc/ssl/certs
  ##   sudo openssl dhparam -out dhparam.pem 4096
  ##
  # ssl_dhparam /etc/ssl/certs/dhparam.pem;
```

```
  location / {
    proxy_pass http://docservice;
    proxy_http_version 1.1;
  }
}
```

代理虚拟路径

当您需要将主机上的某个目录映射到 Web 服务器中指定的路径时，这是一个合适的场景：

`default.conf` ：配置文件

`backendserver-address` ：指定服务器地址或者服务名

```
#Use this example for the proxy document server running at 'backendserver
-address'
# into the virtual directory 'documentserver-virtual-path'.

upstream docservice {
  server backendserver-address;
}

map $http_x_forwarded_proto $the_scheme {
     default $http_x_forwarded_proto;
     "" $scheme;
}

map $http_x_forwarded_host $the_host {
    default $http_x_forwarded_host;
    "" $host:$server_port;
}

map $http_upgrade $proxy_connection {
  default upgrade;
  "" close;
}

proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $proxy_connection;
proxy_set_header X-Forwarded-Host $the_host/documentserver-virtual-path;
proxy_set_header X-Forwarded-Proto $the_scheme;
proxy_set_header Host            $proxy_host;
proxy_set_header X-Real-IP       $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
server {
  listen 0.0.0.0:80;
  listen [::]:80 default_server;
  server_tokens off;

  location /documentserver-virtual-path/ {
    proxy_pass http://docservice/;
    proxy_http_version 1.1;
  }
}
```

与应用集成 (非虚拟路径)

假设你已经有一个应用地址为 `https://www.A.com` ，同时想把编辑器部署在这个域名下，这是一个合适的场景：

`https://www.A.com/7.4.0-163/web-apps/apps/api/documents/api.js`

```
# backendserver-address 换成你的服务名或者地址
upstream docservice {
  server backendserver-address;
}

map $http_x_forwarded_proto $the_scheme {
     default $http_x_forwarded_proto;
     "" $scheme;
}

map $http_x_forwarded_host $the_host {
    default $http_x_forwarded_host;
    "" $host;
}

map $http_upgrade $proxy_connection {
  default upgrade;
  "" close;
}

proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection $proxy_connection;
proxy_set_header X-Forwarded-Host $the_host;
proxy_set_header X-Forwarded-Proto $the_scheme;
```

```
proxy_set_header Host              $proxy_host;
proxy_set_header X-Real-IP         $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;


server {
  listen 0.0.0.0:80;
  listen [::]:80 default_server;
  server_tokens off;

  # 你的 office 版本号
  location /7.4.0-163/ {
    proxy_pass http://docservice;
    proxy_http_version 1.1;
  }

  location /cache/ {
    proxy_pass http://docservice;
    proxy_http_version 1.1;
  }
}
```

docx 转 pdf

docx 转 pdf

地址：POST http://documentserver/ConvertService.ashx

```
{
    "async": false,
    "filetype": "docx",
    "key": "Khirz6zTPdfd7",
    "outputtype": "pdf",
    "title": "Example Document Title.docx",
    "url": "https://example.com/url-to-example-document.docx"
}
```

Response

```
{
  "fileUrl": "http://example.com/cache/files/conv_3917555110101_pdf/output.pdf/Example%20Document%20Title.pdf?md5=Yi-k3hvnnp_30xfBQ9gcFg&expires=1663401660&disposition=attachment&filename=Example%20Document%20Title.pdf",
  "percent": 100,
  "endConvert": true
}
```

错误码	描述
-1	未知错误
-2	转换超时错误
-3	转换错误
-4	下载要转换的文档文件时出错
-5	密码错误
-6	访问转换结果数据库时出错
-7	输入错误

错误码	描述
-8	令牌无效

指令服务

强制保存

地址：POST http://documentserver/coauthoring/CommandService.ashx

```
{
    "c": "forcesave",
    "key": "Khirz6zTPdfd7",
    "userdata": "sample userdata"
}
```

Response

```
{
    "error": 0,
    "key": "Khirz6zTPdfd7"
}
```

踢出用户

```
{
    "c": "drop",
    "key": "Khirz6zTPdfd7",
    "users": [ "6d5a81d0" ]
}
```

Response

```
{
    "error": 0,
    "key": "Khirz6zTPdfd7"
}
```

前端SDK

前端SDK

初始化后，可调用sdk能力

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

拒绝编辑

```
docEditor.denyEditingRights("hello,world");
```

销毁文档对象

```
docEditor.destroyEditor();
```

下载编辑后的文件

```
docEditor.downloadAs(格式);
```

::: tip 温馨提示
只有在存在onDownloadAs事件 时才能调用该方法。文档编辑服务异步创建文档并使用参数中的链接触发onDownloadAs事件。
:::

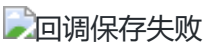
事件

https://api.onlyoffice.com/zh/editors/methods

常见错误

常见错误

无法保存

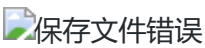


编辑器加载了"无法保存文档"消息。

文档编辑服务 无法连接到 editorConfig.callbackUrl 地址的 文档存储服务。

检查 回调处理程序 是否正常工作。 文档存储服务 必须返回 {"error": 0} 作为响应。

文件错误



编辑器加载了"保存文件时发生错误"消息。

文档编辑服务 无法连接到 web-apps 与 server 层使用 sdk 不一致造成脚本执行错误。

检查 web-apps 层 与 server 层 是否使用相同的 sdk。

文件版本已更新



编辑器加载了"文件版本已更改。 将重新加载页面"消息。

文档编辑服务 无法打开之前编辑和保存过的文件来进行编辑。

不要忘记，每次编辑和保存文档时，都必须重新生成 document.key。

下载失败



在编辑器加载过程中会显示"下载失败"消息。

文档编辑服务 无法上传文件进行编辑。

检查 document.url 中指定的文件的链接是否正确。 该链接必须可以从 文档编辑服务访问。

令牌无效

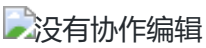


编辑器加载了"文档安全令牌没有正确生成。 请联系您的文档服务器管理员"消息。

文档编辑服务 请求加密过的 签名。

检查 令牌 是否正确。 令牌必须根据 JWT（JSON Web 令牌）标准 生成并存在于 ONLYOFFICE 文档服务器 配置中。

没有协作编辑



当不同用户打开文档进行编辑时，无法进行共同编辑。

文档编辑服务 打开 两个不同 的文件进行编辑。

检查 document.key 值是否一致以便能够共同编辑同一个文档。 key 值在保存后必须改变，不同文档必须不同，并且在共同编辑同一个文档时必须相同。

文件版本打不开



文档编辑服务 无法打开文件版本。

检查setHistoryData方法中的changesUrl链接是否与previous.url参数对应。

字体名称

字体名称

制作中文字体

1. 下载 High-Logic FontCreator
2. 导入系统字体
    foont
3. 修改名称
    font
4. 导出
    font

已有中文字体

1. 已经制作好的中文字体
   - 链接: https://pan.baidu.com/s/1dn6Vv1KF1KCh69efmojsDg
   - 提取码: 2gft

```
宋体
仿宋
幼圆
楷体
等线
隶书
黑体
小标宋
新宋体
华文中宋
华文仿宋
华文宋体
华文彩云
华文新魏
华文楷体
华文琥珀
```

```
华文细黑
华文行楷
华文隶书
微软雅黑
方正姚体
方正舒体
仿宋_GB2312
楷体_GB2312
方正仿宋简体
方正黑体简体
方正小标宋简体
```

中文字号

中文字号

方案一

1. 查看容器ID

```
docker ps
```

2. 进入容器

```
docker exec -it <容器ID> /bin/bash
```

3. 添加中文字号

```
sed -i "s/{value:8,displayValue:\"8\"}/{value:42,displayValue:\"初号\"},{value:36,displayValue:\"小初\"},{value:26,displayValue:\"一号\"},{value:24,displayValue:\"小一\"},{value:22,displayValue:\"二号\"},{value:18,displayValue:\"小二\"},{value:16,displayValue:\"三号\"},{value:15,displayValue:\"小三\"},{value:14,displayValue:\"四号\"},{value:12,displayValue:\"小四\"},{value:10.5,displayValue:\"五号\"},{value:9,displayValue:\"小五\"},{value:7.5,displayValue:\"六号\"},{value:6.5,displayValue:\"小六\"},{value:5.5,displayValue:\"七号\"},{value:5,displayValue:\"八号\"},{value:8,displayValue:\"8\"}/g" `grep -rwl --include="*.js" "{value:8,displayValue:\"8\"}" /var/www/onlyoffice/documentserver/web-apps/apps`
```

4. 固定字体列表高度
   ` sh
   sed -i “s/{cls:"input-group-nr",menuStyle:"min-width: 55px;"/{cls:"input-group-nr",menuStyle:"min-width: 55px;height: 500px;"/g” `grep -rwl --include="*.js" "{value:8,displayValue:\"8\"}" /var/www/onlyoffice/documentserver/web-apps/apps`

```
5. 重启容器
```sh
docker restart <容器ID>
```
```

6. 制作新镜像

```
docker commit -p -a "Knox" -m "zh-cn" <容器ID> ooffice:v1
```

中文字体

中文字体

原文链接：https://blog.csdn.net/weixin_44876263/article/details/145629622
详细方案

1. 删除原生字体文件
   OnlyOffice的容器中可能自带一些原生的字体文件，首先需要删除这些文件，以避免与新字体发生冲突。
   操作步骤：
   登录到运行OnlyOffice的服务器。
   使用Docker命令进入OnlyOffice容器，并删除原有的字体文件：

```
sudo docker exec $DOCKER_NAME rm -rf /var/www/onlyoffice/documentserver/core-fonts
```

   这一步骤将会删除OnlyOffice编辑器中的原生字体文件。
2. 创建字体目录
   在删除原有字体文件后，我们需要为新的字体文件创建一个目录，用于存放我们替换的字体。
   操作步骤：
   创建字体目录：

```
sudo docker exec $DOCKER_NAME mkdir /var/www/onlyoffice/documentserver/core-fonts
```

   这样，我们就为新的字体文件准备好了一个存放的目录。
3. 复制字体文件到容器中
   接下来，我们需要将常见的字体文件复制到容器中的指定目录。这些字体可以是任何你需要的常见字体文件，如Arial、Calibri、Times New Roman等。



CSDN @老友@

   操作步骤：
   使用Docker命令将字体文件从宿主机复制到容器内。假设字体文件保存在 $FONTS 目录下，以下是具体命令：

```
sudo docker cp $FONTS/AlibabaPuHui.woff $DOCKER_NAME:/usr/share/fonts/truetype/custom
sudo docker cp $FONTS/Aptos.woff $DOCKER_NAME:/usr/share/fonts/truetyp
```

```
e/custom
sudo docker cp $FONTS/Arial.woff $DOCKER_NAME:/usr/share/fonts/truetyp
e/custom
```

在这里，$FONTS 是你存放字体文件的目录，$DOCKER_NAME 是你的OnlyOffice Docker容器的名称。每个字体文件都需要手动复制到容器内的 /usr/share/fonts/truetype/custom 目录。

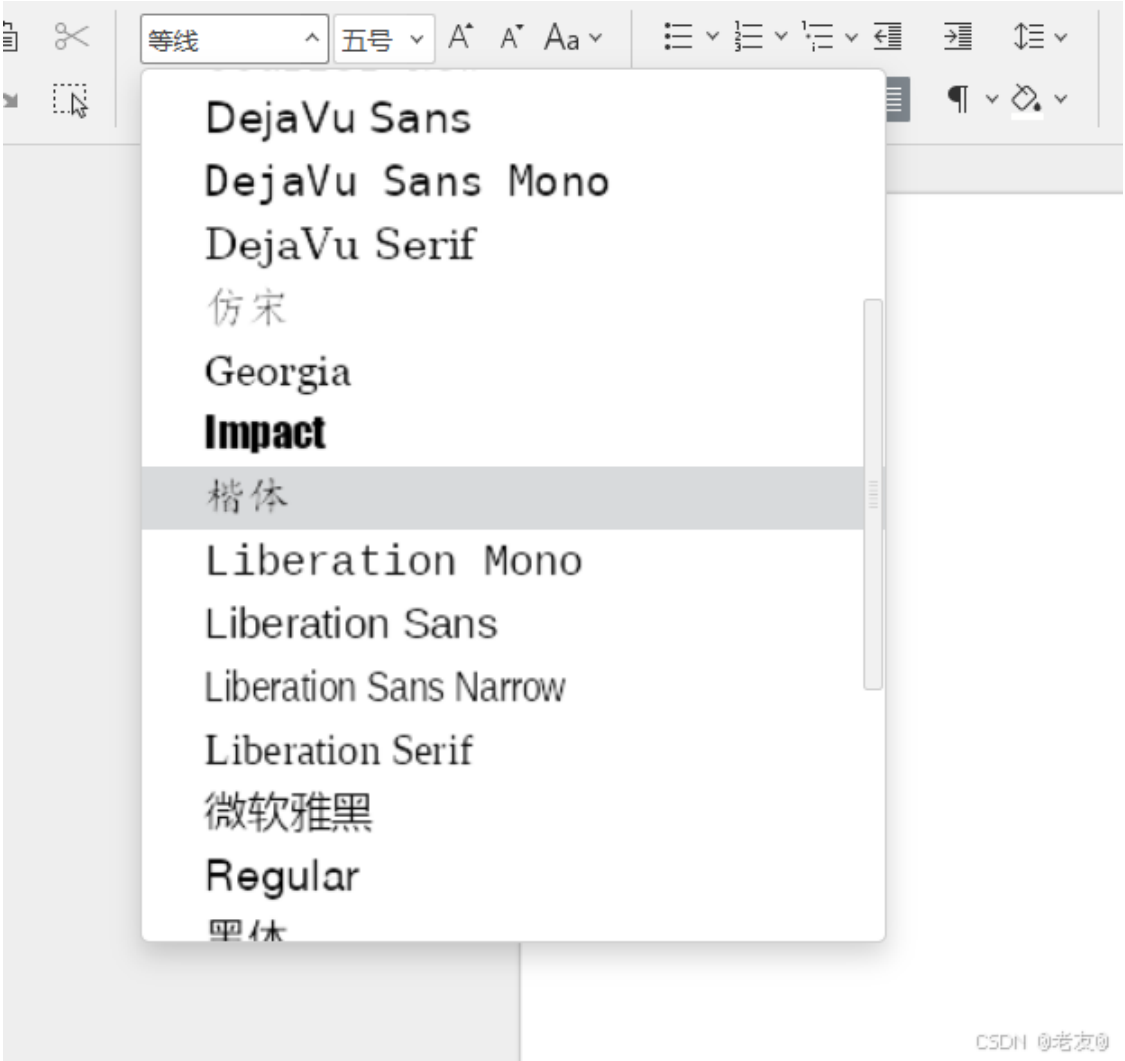
4. 执行字体更新脚本
   完成字体文件复制后，我们需要执行OnlyOffice的字体更新脚本，这样系统才能识别并加载新的字体。
   操作步骤：
   执行字体更新命令：

```
sudo docker exec $DOCKER_NAME /usr/bin/documentserver-generate-allfont
s.sh
```

5. 重新启动容器
   修改完字体后，需要重启OnlyOffice容器以使更改生效。
   操作步骤：
   使用以下命令重新启动容器：

```
sudo docker restart $DOCKER_NAME
```

重新启动容器后，新的字体配置将会生效。



注意事项
字体版权：使用新的字体时，请确保拥有字体文件的合法使用权。
字体格式：确保你使用的字体文件格式（如 .woff、.ttf）与OnlyOffice兼容，通常 .woff 格式是推荐的字体格式。
备份：在进行任何更改之前，最好备份现有配置文件，以防万一出现问题。

方案一

1. 把从 windows 字体目录拷贝的文件 cp 到镜像内

```
docker cp /root/fonts/ <容器ID>:/usr/share/fonts/
```

::: tip 温馨提示
拷贝之前先把中文字体属性名称改成中文,否则编辑器字体列表不会显示中文名称。

例如：【宋体】会显示成 【simsun】

修改中文字体名称
:::

::: tip 温馨提示
如果您觉得麻烦可以下载现有的字体 去下载
:::

1. 进入容器

```
docker ps
docker exec -it <容器ID> /bin/bash
```

3. 执行命令

```
/usr/bin/documentserver-generate-allfonts.sh
```

4. 把容器保存为新镜像

```
docker commit -p -a "Knox" -m "fonts" <容器ID> ooffice:v1
```

::: tip 温馨提示
如果新增字体不显示，尝试清理浏览器缓存，因为你可能读取的是浏览器缓存
:::

方案二

1. 用 Dockerfile 重新制作一个新的镜像
2. Dockerfile 文件示例

```
FROM knoxzhang/oo-ce-docker-license
COPY fonts/ /usr/share/fonts/
```

3. 制作镜像

```
docker build -t ooffice:v1 .
```

文件大小限制

Onlyoffice文件大小限制

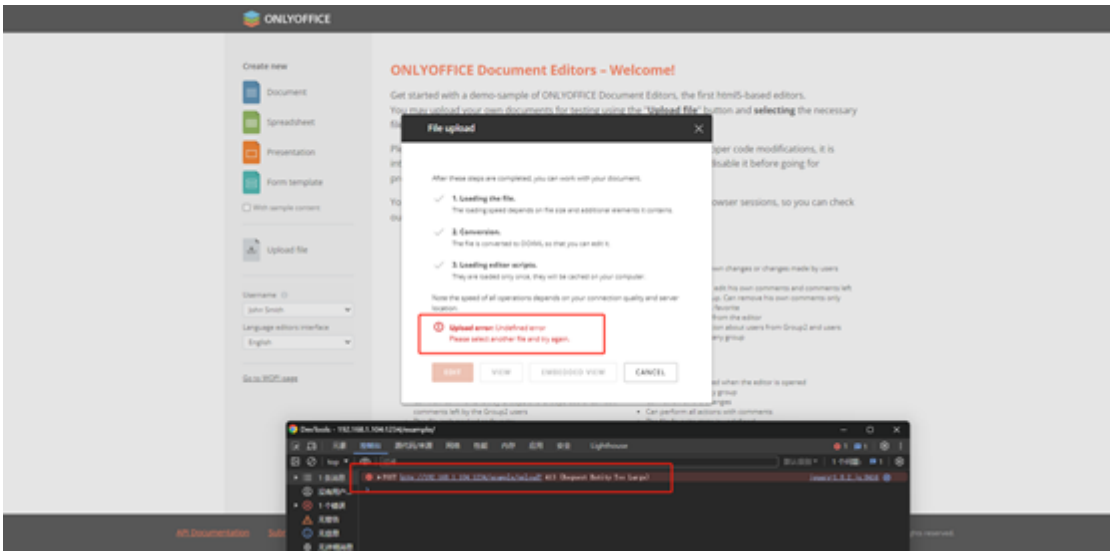
1 前言

Onlyoffice默认初始文件上线为100MB，超过100MB的文件在example中会无法上传。打开文件的时候会报错“文件超出大小限制”
文件越大，onlyoffice效率会越低，所以尽量将大小设置在1GB内，即1000MB
注意，所有配置规则修改后一定要重启四大服务。所有配置规则修改后只对新上传的文件才有效。

2 Example报错

2.1 413 (Request Entity Too Large)

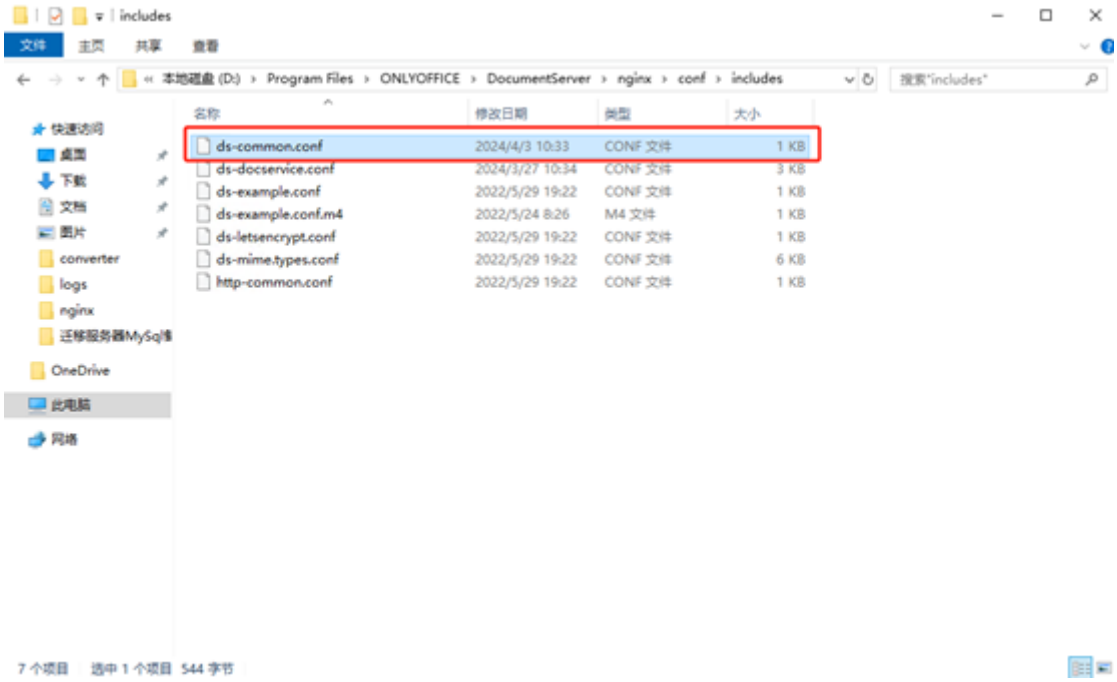
2.1.1 错误描述



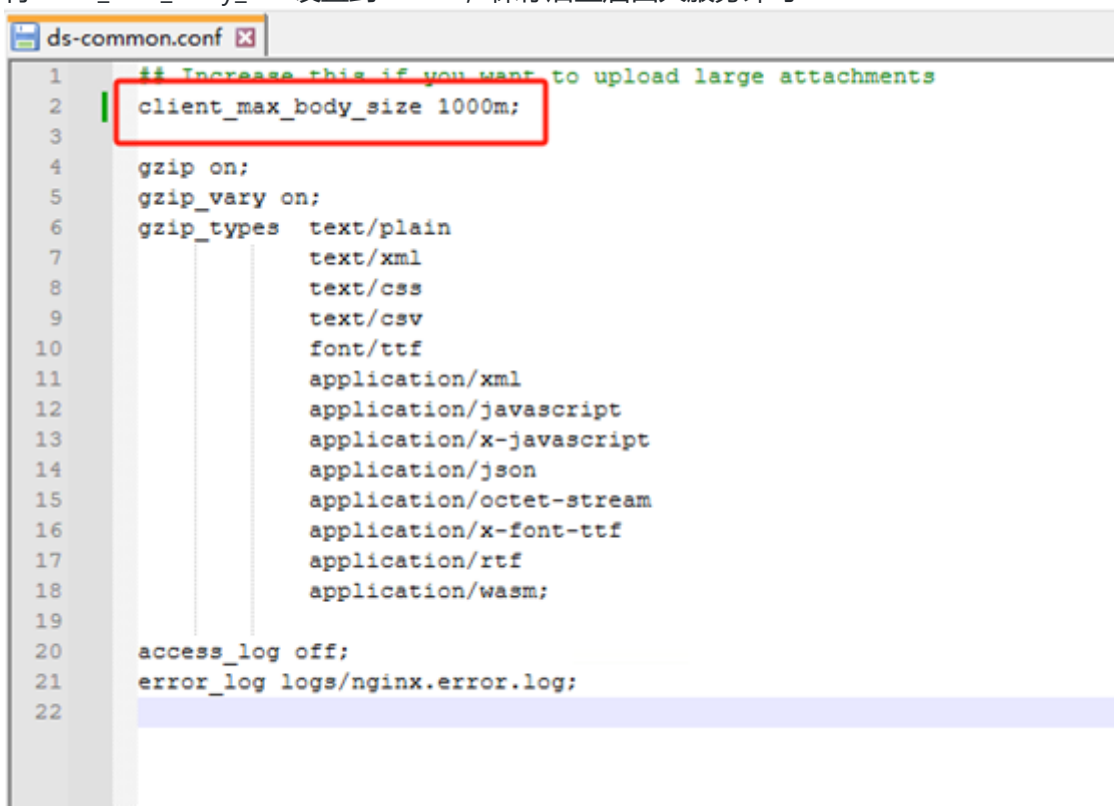
带413状态码的请求实体过大
这种情况通常发生在请求文件过大（不包含参数）的情况下。客户端发送的请求实体超过服务器所能处理的最大限制，服务器会返回413状态码提示请求实体过大。

2.1.2 解决方案

1. 定位到目录“ONLYOFFICE\DocumentServer\nginx\conf\includes”，打开“ds-common.conf”

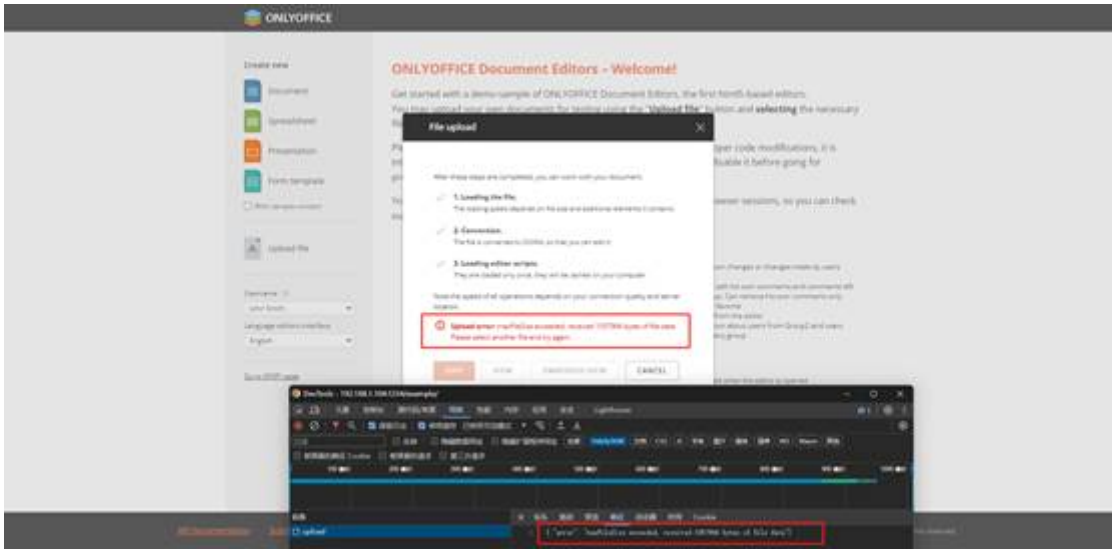


2. 将client_max_body_size设置到1000M，保存后重启四大服务即可



2.2 MaxFileSize exceeded

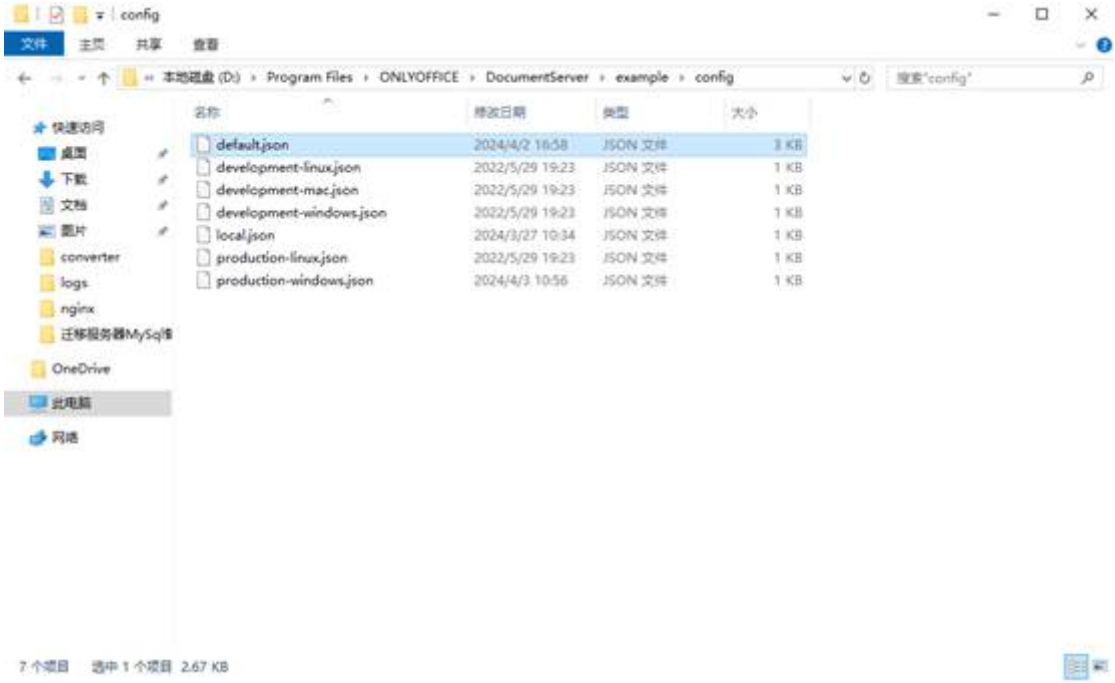
2.2.1 错误描述



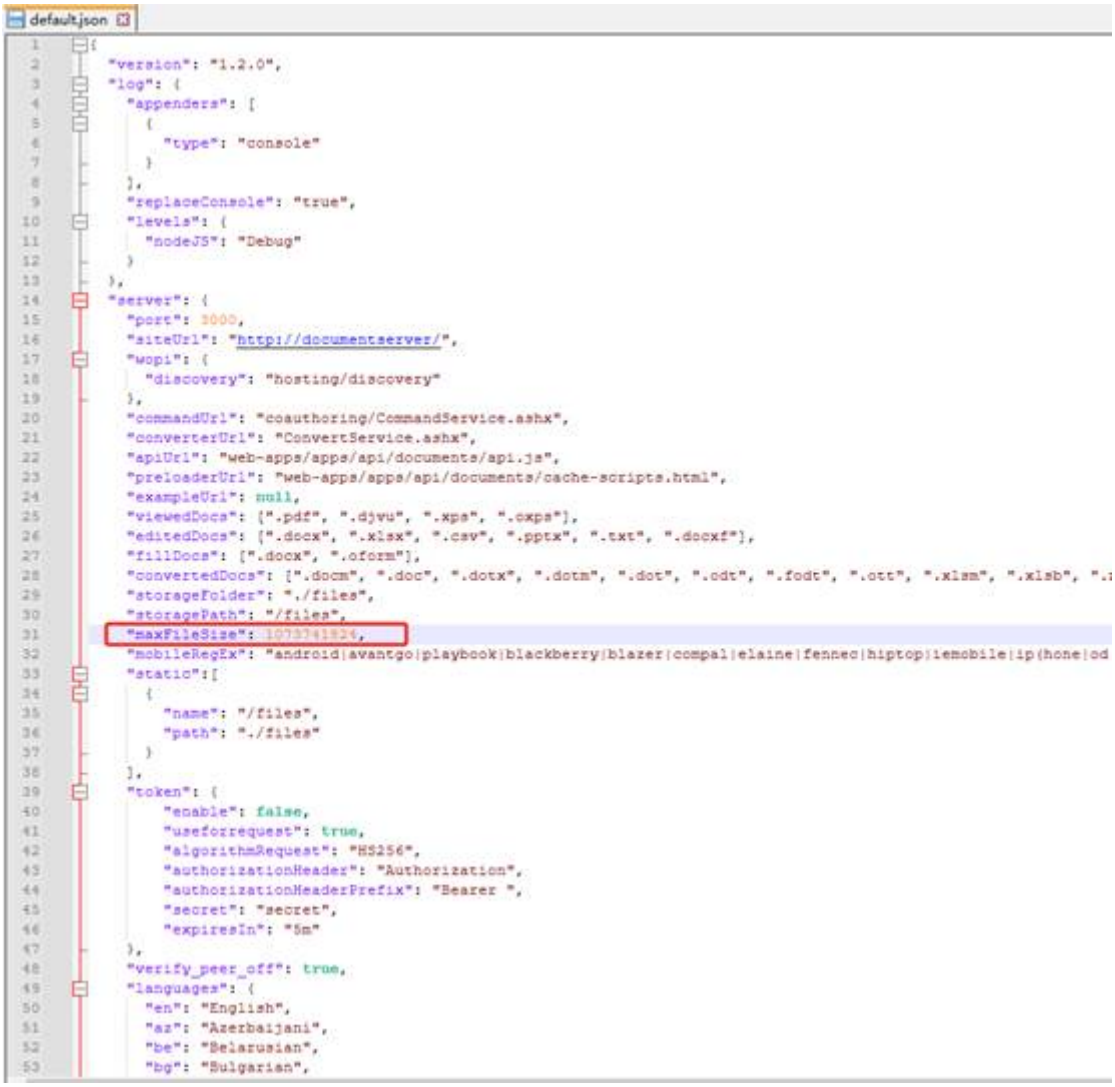
上传文件大小超出可最大限制

2.2.2 解决方案

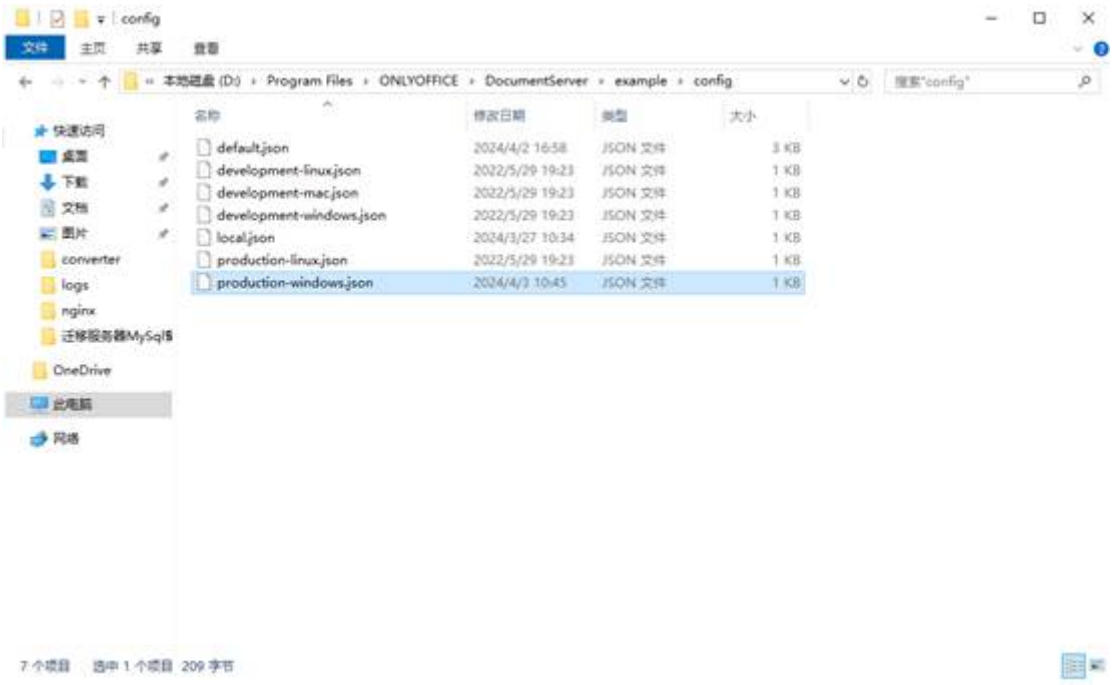
3. 定位到目录“ONLYOFFICE\DocumentServer\example\config”，打开“default.json”



4. 修改“maxFileSize”值为“1043471824”，表示最大上线为1000M



5. 定位到目录“ONLYOFFICE\DocumentServer\example\config”，打开“production-windows.json”



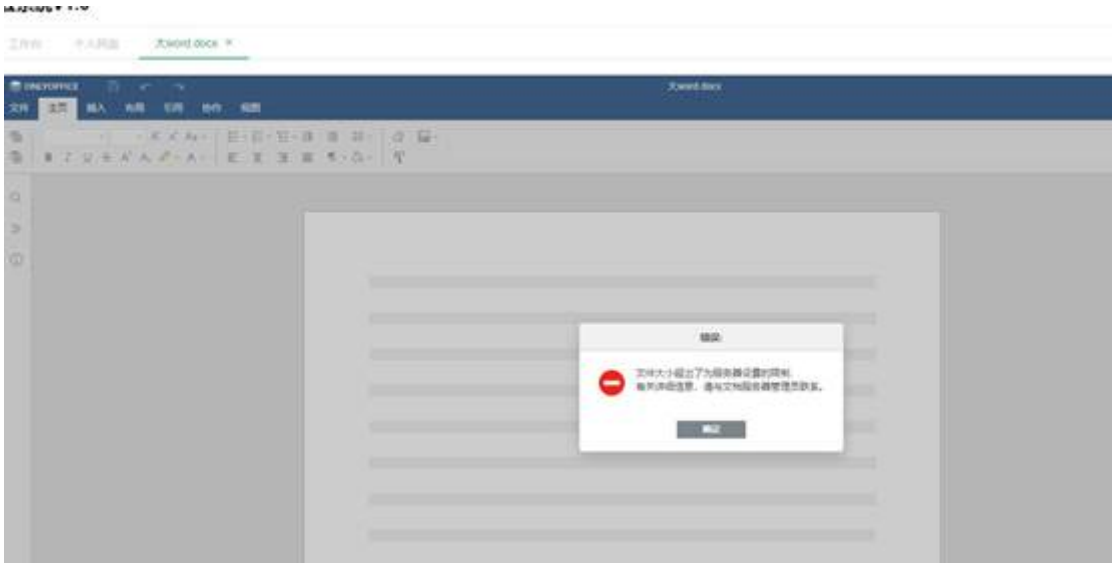
6. 修改“maxFileSize”值为“1043471824”，表示最大上线为1000M



3 DocumentServer报错

3.1 文件大小限制

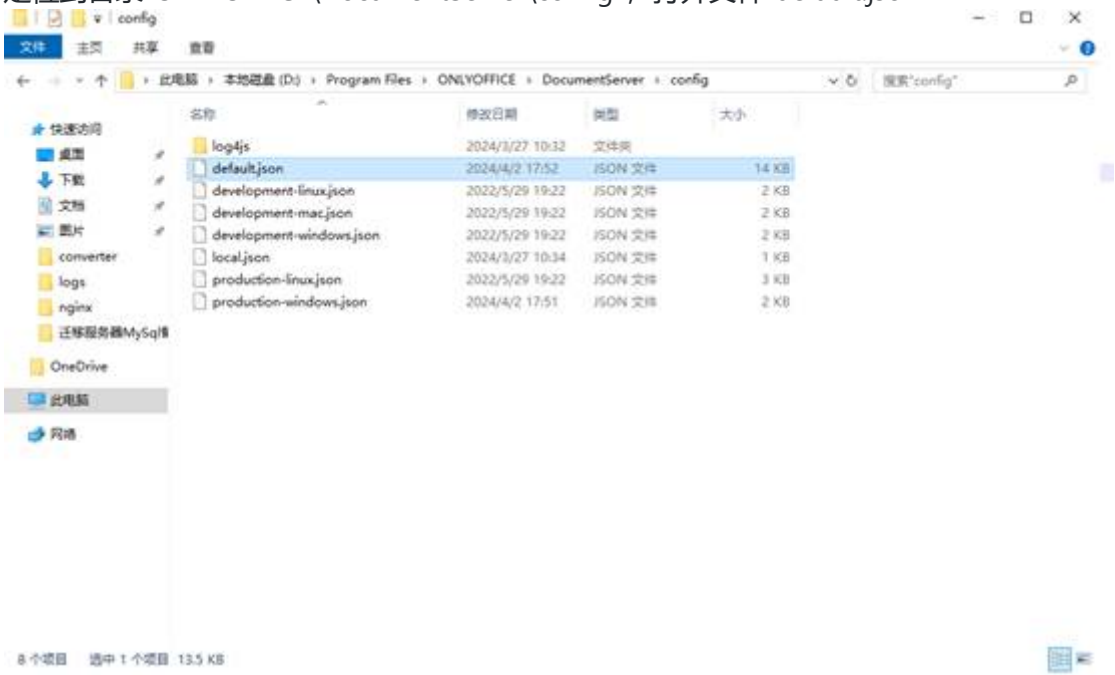
3.1.1 错误描述



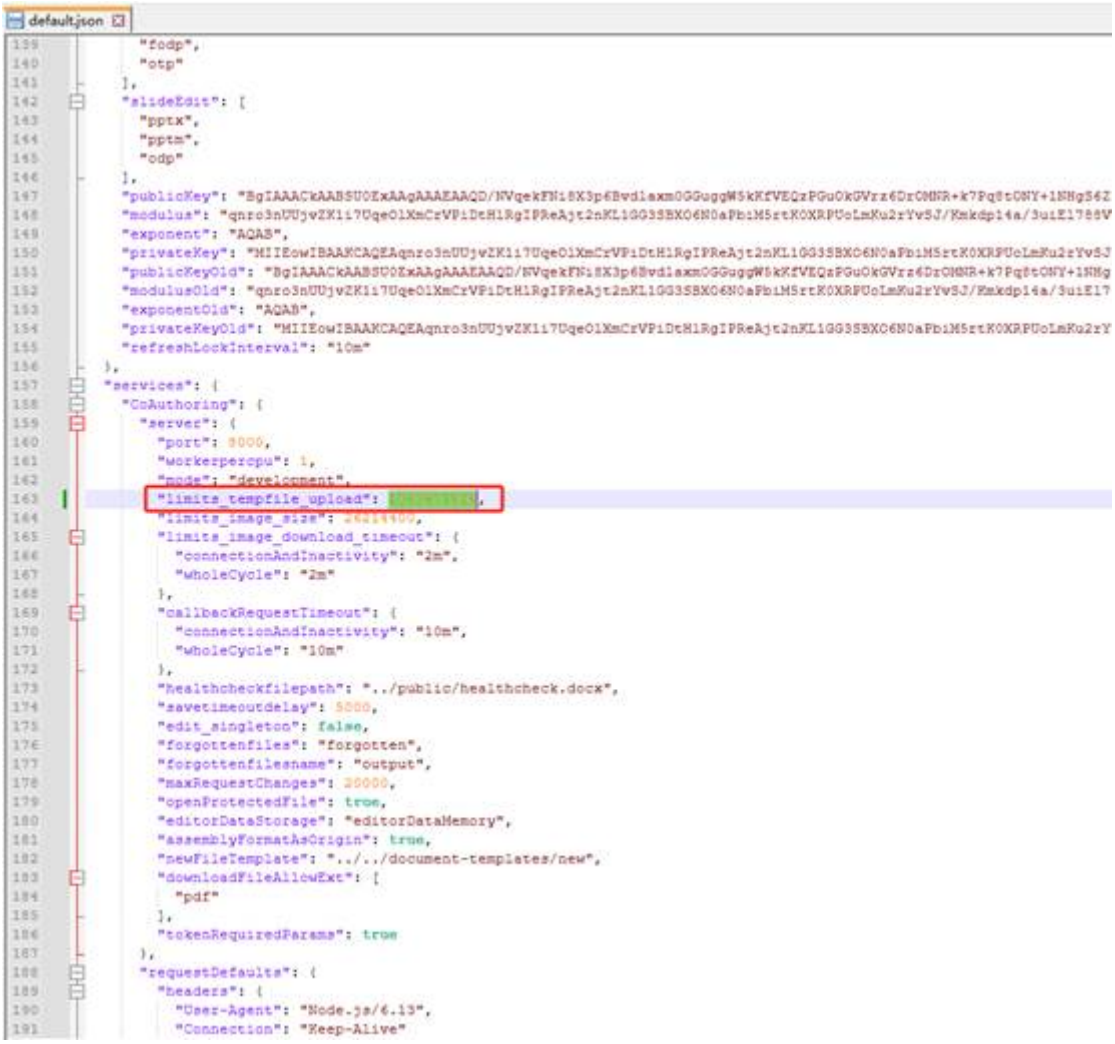
打开文档提示“文件大小超出了为服务器设置的限制”

3.1.2 解决方案

7. 定位到目录“ONLYOFFICE\DocumentServer\config”，打开文件“default.json”



8. 修改“limits_tempfile_upload”值为“1043471824”



9. 修改“maxDownloadBytes”值为“1043471824”，修改“uncompressed”值为“1000MB”

default.json

```
352       "packageType": 0
353     },
354     "FileConverter": {
355       "converter": {
356         "maxDownloadBytes": 1043471824,
357         "downloadTimeout": {
358           "connectionAndInactivity": "2m",
359           "wholeCycle": "2m"
360         },
361         "downloadAttemptMaxCount": 3,
362         "downloadAttemptDelay": 1000,
363         "maxprocesscount": 1,
364         "fontDir": "null",
365         "presentationThemesDir": "null",
366         "x2tPath": "null",
367         "docbuilderPath": "null",
368         "docbuilderAllFontsPath": "null",
369         "docbuilderCoreFontsPath": "",
370         "args": "",
371         "spawnOptions": {},
372         "errorfiles": "",
373         "streamWriterBufferSize": 8388608,
374         "maxRedeliveredCount": 2,
375         "inputLimits": [
376           {
377             "type": "docx;dotx;docm;dotm",
378             "zip": {
379               "uncompressed": "1000MB",
380               "template": "*.xml"
381             }
382           },
383           {
384             "type": "xlsx;xltx;xlsm;xltm",
385             "zip": {
386               "uncompressed": "1000MB",
387               "template": "*.xml"
388             }
389           },
390           {
391             "type": "pptx;ppsx;potx;pptm;ppsm;potm",
392             "zip": {
393               "uncompressed": "1000MB",
394               "template": "*.xml"
395             }
396           }
397         ]
398       }
399     }
400   }
401
```

试一试

试一试

–	文档	电子表格	演示文稿
编辑	DOCX	XLSX	PPTX

插件API

https://mp.weixin.qq.com/s?__biz=MzI2MjUyNzkyNw==&mid=2247503905&idx=1&sn=20cbdf930e217417b604559a93712e8d&chksm=ea4b5ef9dd3cd7ef3f539a6f50f06353e0fbf59852e8b2590480c9ad8dfc690bd0488d50d694&mpshare=1&scene=22&srcid=01259h0TuqZOtUEsUTYX2DKp&sharer_sharetime=1674708141464&sharer_shareid=02e7a23f26ced4faeb5fcb359d794ee8#rd
如何为自己的 ONLYOFFICE 开发创建一个功能插件
那么对于一个开源的软件，最值得我们程序员入手学习的就是功能扩展开发了，这里就开始讲一讲，如何为自己的 ONLYOFFICE 开发创建一个功能插件。开发一个 ONLYOFFICE 插件需要如下几个基本步骤概念：

准备
编码开发
调试插件的风格样式
本地化
准备发布
测试
公开发布
第一步：准备
开发一个插件并且要发布在 ONLYOFFICE 官方的插件市场上，你需要有一个GitHub[1]账号，注册一个新账号或登陆你自己的账号。
将 ONLYOFFICE 官方插件市场源码文件仓库新开一份分叉到你的账号下：*https://github.com/ONLYOFFICE/onlyoffice.github.io*.
图片
将会得到你自己的一份源码文件仓库，地址类似于这样：*https://github.com/YOUR-USERNAME/ONLYOFFICE/onlyoffice.github.io*.

图片
创建你自己的 GitHub Pages 网站，方便之后在线测试你的插件，可以参考官方GitHub Pages 快速入门[2]。
把代码 clone 到自己的开发用的计算机上
图片
图片
图片
图片

在路径onlyoffice.github.io/sdkjs-plugins/content下添加一个文件夹开发自己的插件，比如起名为HelloWorld，这里我并没有真的打算开发一个有什么特定的我需要的功能而 ONLYOFFICE 本身没有提供的功能，因为 ONLYOFFICE 基本上提供了完备的所有办公功能，所以自己决定只是照搬官网的 Hello World 这个最简单的开发测试用例来举例子。
图片
第二步：编码开发

开发你的插件，遵循这里介绍的架构[3]开发，该文件夹下必须包含如下三个文件：*config.json*[4]、*index.html*[5]、*pluginCode.js*[6]

图片
需要使用插件 API 的方法 method[7]和事件 event[8]来编写实现插件功能。

在学习入门阶段，为了最简化问题，就开发一个 Hello World 功能来测试，并且不准备最终提交 ONLYOFFICE 官方插件市场。最终开发实现的，这个功能很简单，怎么开发我不管，就是如下图所示

图片
我点击这个测试用插件的按钮，自动插入 Hello World。为了更简化开发，不用手写一行代码，直接打开官方提供的 Hello World 代码库。

直接把下载的 hello world 示例文件夹的内容拷贝进自己的 hello world 文件夹内即可。

第三步：调试插件的风格样式
添加 ONLYOFFICE 的Style 风格 Sheet[9]到index.html文件，调试你的插件的风格样式

在config.json文件中的翻译如下：

图片
这是一个按钮，按钮上的 ok 这个字符串如果需要翻译的话，新建一个对象为textLocale，其内仅添加一种中文既zh，内容为确认

第五步：准备发布
创建一个关于窗口，包括一段简短的说明、版本号、开发者或开发商信息、官网链接等。在config.json文件中创建**关于**窗口的详情在这里[11]
图片
准备图标，置于resources文件夹内，在*config.json*文件中创建配置图标的详情在这里[12]

注意总共需要准备 8 个图标文件，四种不同的大小比例：125%、150%、175%、200%以及正常和黑暗两种模式。

不要忘了readme说明文件，说明功能特征、安装卸载、使用方法、以及其它任何你想告知使用者的信息，推荐为 markdown 格式。

第六步：测试
准备好上述所有文件后可以推送到你的远程代码仓库上，就可以用你的在线版和本地版 ONLYOFFICE 测试功能了。

在桌面版 ONLYOFFICE 编辑器[13]中测试
在线测试之前先要完全在本地测试一遍，在配置文件config.json中找到插件的唯一 GUID 识别代号：

图片
在 Windows 系统下 ONLYOFFICE 安装路径下面的 plugins 插件集中存放的路径d:\Program

Files\ONLYOFFICE\DesktopEditors\editors\sdkjs-plugins下创建一个文件夹，以该 GUID 命名

图片
然后将开发的该插件的文件夹内的所有文件拷贝到新建的该路径下

图片
就可以打开本地版 ONLYOFFICE 编辑器查看效果了

图片
可以用 debug 模式启动本地 ONLYOFFICE 编辑器，启动时添加参数–ascdesktop-support-debug-info，根据你电脑所使用的操作系统的版本，参考这个帮助页来开启 debug 模式运行[14]

图片
用 debug 模式可以调试自己开发出来的 bug，毕竟不写 bug 的程序员不是好程序员，在 debug 模式打开 ONLYOFFICE 桌面编辑器后，可以在任何操作过程中按下键盘快捷键 F1 键来启动 debug 工具如下图所示：

图片
看到这个画面，web 网页开发工程师就心里踏实了，这不就是最熟悉的 debug 工具功能吗，这里就可以查看调试自己所开发的任意代码了。

在线版 ONLYOFFICE 编辑器中测试
本地测试通过后再推送到网络上，在网页浏览器里面测试在线 ONLYOFFICE 版本使用新开发的插件的功能。

如果有自己团队架设了 ONLYOFFICE 文档服务器，那就安装到自己的文档服务器上测试使用，ONLYOFFICE 文档服务器端安装测试新开发的插件的操作说明在这里[15]。但是我更建议在部署到文档服务器之前，先安装到网页浏览器里面测试插件功能，这样有几个好处：你没有自己的文档服务器可以用 ONLYOFFICE 的官方免费个人在线版来测试，官方在线 ONLYOFFICE 或者团队的 ONLYOFFICE 文档服务器不是你能去操作服务器端安装插件的，在浏览器里面不管连接哪一家的文档服务器都能使用自己开发的新插件。

首先别忘了把自己开发的新插件推送到自己 Github 账户下分叉出来的onlyoffice.github.io这个代码库上，在网页上查看确认自己写的 HelloWorld 确实存在了

图片
然后打开本地文件onlyoffice.github.io/store/plugin-dev/extension/inject.js，向参数 URL_TO_PLUGIN添加你开发的插件在 GitHub 上的网络路径

var URL_TO_PLUGIN = “https://YOUR-USERNAME.github.io/onlyoffice.github.io/sdkjs-plugins/content/helloworld/"
这个地址需要根据自己的 Page 修改，然而按照手册的这个地址自己反复测试也不通过，最后自己采用了比较简单粗暴的方法，直接把这个helloworld文件夹整体拷贝到自己的 Pages 的根路径下反倒是能够最终测试成功，

图片
修改inject.js文件内容：

图片
修改好的本地文件夹onlyoffice.github.io/store/plugin-dev/extension就是浏览器的一个新插件了，安装到浏览器[16]上，比如一个新插件开发完成了，再开发另一个新插件，则不要忘了再修改onlyoffice.github.io/store/plugin-dev/extension/inject.js这个文件指向新插件的 GitHub 的网络路径，不要忘了在浏览器里面卸载重新安装这个插件

图片
比如在 Chrome 浏览器中手动安装插件，选择本地安装从onlyoffice.github.io/store/plugin-dev/extension文件夹安装，浏览器安装好这个 ONLYOFFICE 插件开发专用插件之后，就可以打开在线版 ONLYOFFICE 文档，比如ONLYOFFICE 免费个人版[17]，官方免费个人版只能安装官方插件市场的插件，但是通过这个浏览器插件，就可以测试自己开发的插件了，就可以在插件栏中发现已经自动加载了自己新开发的插件了：

图片
在插件管理器[18]中找到该插件

注意要在这里的插件管理器、我的已安装插件界面里面切换正常和黑夜模式，来观察图标设计是否清晰辨认，如有必要返回上述有关步骤修改设计

图片
第七步：公开发布
如果想把你开发的插件贡献到 ONLYOFFICE 的官方插件市场上，从你的 fork 分支向官方的https://github.com/ONLYOFFICE/onlyoffice.github.io发起一个pull request[19]，功能正常的话，你的 pull request 将被批准并上线官方插件市场。

最后，你可以讨论插件的开发、请求新功能、或者报告 bug 等等，只需要在marketplace repository 插件市场库[20]中提交一个 issue 提案。

参考文献
[1]GitHub: https://github.com
[2]GitHub Pages 快速入门: https://docs.github.com/zh/pages/quickstart
[3]这里介绍的架构: https://api.onlyoffice.com/plugin/structure
[4]config.json: https://api.onlyoffice.com/plugin/config
[5]index.html: https://api.onlyoffice.com/plugin/indexhtml
[6]pluginCode.js: https://api.onlyoffice.com/plugin/code
[7]方法method: https://api.onlyoffice.com/plugin/plugin
[8]事件event: https://api.onlyoffice.com/plugin/events
[9]Style风格Sheet: https://api.onlyoffice.com/plugin/styles
[10]翻译的细节可以参考这个帮助: https://api.onlyoffice.com/plugin/localization
[11]创建关于窗口的详情在这里: https://api.onlyoffice.com/plugin/variations
[12]在config.json文件中创建配置图标的详情在这里: https://api.onlyoffice.com/plugin/icons
[13]桌面版ONLYOFFICE编辑器: https://api.onlyoffice.com/plugin/installation/desktop

[14]参考这个帮助页来开启debug模式运行: https://api.onlyoffice.com/desktop/debugging
[15]ONLYOFFICE文档服务器端安装测试新开发的插件的操作说明在这里:
https://api.onlyoffice.com/plugin/installation/onpremises#plugin-manager
[16]浏览器: https://api.onlyoffice.com/plugin/installation/cloud#upload-extensions
[17]ONLYOFFICE免费个人版: personal.onlyoffice.com
[18]插件管理器: https://api.onlyoffice.com/plugin/installation/onpremises#plugin-manager
[19]pull request: https://github.com/ONLYOFFICE/onlyoffice.github.io/pulls
[20]marketplace repository插件市场库:
https://github.com/ONLYOFFICE/onlyoffice.github.io/issues
[21]Getting started with plugins: https://api.onlyoffice.com/plugin/gettingstarted
[22]Hello World: https://api.onlyoffice.com/plugin/example/helloworld
图片 第四步：本地化 创建translations文件夹，为每一个语言创建一个json文件，具体本地化翻译的细节可以参考这个帮助[10]。

插件

插件

示例

```
(function (window, undefined) {
  window.Asc.plugin.init = function () {
    var sdkApi = window.parent.DE.getController("Main").api;
    console.log("可以调用的sdkapi", sdkApi);
    window.parent.addEventListener(
      "message",
      function (res) {
        var data = { type: 1 };
        try {
          data = JSON.parse(res.data);
        } catch (error) {}
        switch (data.type) {
          case 2:
            var currentPage = sdkApi.getCurrentPage();
            break;
          case 3:
            var currentPage = sdkApi.goToPage(1);
            break;

          default:
            break;
        }
      },
      false
    );

    var params = JSON.stringify({ type: 1, message: "加载完成" });
    window.top.postMessage(params, "*");
  };
})(window, undefined);
```

下载

示例代码下载

模板替换

模板替换

字段插入替换

1. 插入字段
   
   放大查看

2. 替换字段
   
   放大查看

3. 下载

   插件例子代码下载

   ::: tip 温馨提示
   通信:如果想在上层应用调用插件里面的方法可以尝试使用 postMessage 更多介绍
   :::

```
/*
* A 窗口的域名是<http://example.com:8080>，以下是 A 窗口的 script 标签下的代码：
*/

var popup = window.open(...popup details...);

// 如果弹出框没有被阻止且加载完成

// 这行语句没有发送信息出去，即使假设当前页面没有改变 location（因为 targetOrigin 设置不对）
popup.postMessage("The user is 'bob' and the password is 'secret'",
                  "https://secure.example.net");

// 假设当前页面没有改变 location，这条语句会成功添加 message 到发送队列中去（targetOrigin 设置对了）
popup.postMessage("hello there!", "http://example.org");

function receiveMessage(event)
```

```
{
// 我们能相信信息的发送者吗?(也许这个发送者和我们最初打开的不是同一个页面).
if (event.origin !== "http://example.org")
    return;

// event.source 是我们通过 window.open 打开的弹出页面 popup
// event.data 是 popup 发送给当前页面的消息 "hi there yourself!  the sec
ret response is: rheeeeet!"
}
window.addEventListener("message", receiveMessage, false);
```

插件入门

插件入门

1. 创建工程名称 example 下载例子

```
.
│  config.json
│  index.html
│  README.md
│
├─resources
│  ├─dark
│  │      icon.png
│  │      icon@1.25x.png
│  │      icon@1.5x.png
│  │      icon@1.75x.png
│  │      icon@2x.png
│  │
│  └─light
│          icon.png
│          icon@1.25x.png
│          icon@1.5x.png
│          icon@1.75x.png
│          icon@2x.png
│
└─scripts
        code.js
```

2. 编写代码
    - 在 code.js 编辑你的业务逻辑
3. 部署到镜像

```
docker cp /example <容器ID>:/var/www/onlyoffice/documentserver/sdkjs-plugins/
```

4. 制作新镜像

```
docker commit -p -a "Knox" -m "plugins" <容器ID> ooffice:v1
```

::: tip 温馨提示
在前端集成 config 中开启插件属性

```
    {
        "editorConfig": {
            "customization": {
                "plugins": true
            }
        },
        ...
    }
```

:::

如何与插件通信

如何与插件通信

同源场景下（同域名、端口）

在文件 `code.js` 暴露 window 对象给顶级窗口

```
window.top.sonWin = window;
window.helloFun = function () {
  console.log("hello world");
};
```

在应用中使用插件内的方法

```
sonWin.helloFun();
// hello world
```

如何代理到指定服务器

::: tip 温馨提示

以 NGINX 为例

查看详细配置
:::

水印

水印

示例



放大查看



放大查看

下载

水印例子代码下载

付费

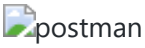
文档生成器（ ￥499 ）

文档生成器（￥499)

镜像部署


builder
点击放大

使用


postman
点击放大

下载

```
# 拉取
docker pull knoxzhang/document-builder:7.5.1.24
```

联系方式

QQ：601424688

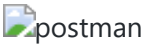
文档生成器（ ￥499 ）

文档生成器（￥499）

镜像部署


点击放大

使用


点击放大

下载

```
# 拉取
docker pull knoxzhang/document-builder:8.0.0.1
```

联系方式

QQ：601424688

付费镜像

付费镜像

pro 镜像 （￥ 499）

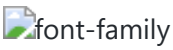
1. 去除 20 链接数限制


点击放大

2. 移动端可编辑


点击放大

3. 增加中文字体


点击放大
不限以上字体

4. 增加中文字号


点击放大

5. 开启多核线程


点击放大

6. 连接器 (自动化 API)*

```
Asc.scope.text = "Hello world!";
var connector = docEditor.createConnector();
connector.callCommand(function() {
    var oDocument = Api.GetDocument();
    var oParagraph = Api.CreateParagraph();
    oParagraph.AddText(Asc.scope.text);
    oDocument.InsertContent([oParagraph]);
}, function() { console.log("callback command"); });
```

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license-pro:7.5.1.23
```

联系方式

QQ：601424688

付费镜像

付费镜像

pro 镜像 （￥ 299.00）

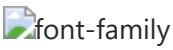
1. 去除 20 链接数限制


点击放大

2. 移动端可编辑


点击放大

3. 增加中文字体


点击放大
不限以上字体

4. 增加中文字号


点击放大

5. 开启多核线程


点击放大

6. 移动端 cell 文件设置可配置隐藏


点击放大

7. 新增 sdkjsapi (新)

可以调用 `sdk` 的所有 `api` 方法，包含范围（pc、移动端，Documents（docx）、Spreadsheets（xlsx）、Presentations(pptx)）

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license-pro:7.1.1.26
```

联系方式

QQ：601424688

付费镜像

付费镜像

pro 镜像 （￥ 499）

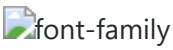
1. 去除 20 链接数限制


点击放大

2. 移动端可编辑


点击放大

3. 增加中文字体


点击放大
不限以上字体

4. 增加中文字号


点击放大

5. 连接器 (自动化 API)*

```
Asc.scope.text = "Hello world!";
var connector = docEditor.createConnector();
connector.callCommand(function() {
    var oDocument = Api.GetDocument();
    var oParagraph = Api.CreateParagraph();
    oParagraph.AddText(Asc.scope.text);
```

```
    oDocument.InsertContent([oParagraph]);
}, function() { console.log("callback command"); });
```

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license-pro:8.0.1.33
```

联系方式

QQ：601424688

连接器

连接器

连接器 是一个允许从外部源编辑文本文档、电子表格、演示文稿和可填写表单的类

结合 `BUILDER.API` 能发挥更强大的能力

自定义右键菜单

光标坐标&被选中的文字

水印

表格操作

书签操作

内容控件操作

无刷新切换`编辑/预览`状态

获取文件链接

更多更能开发中

付费镜像

付费镜像

pro 镜像 （￥ 499）

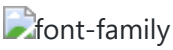
1. 去除 20 链接数限制


点击放大

2. 移动端可编辑


点击放大

3. 增加中文字体


点击放大
不限以上字体

4. 增加中文字号


点击放大

5. 连接器 (自动化 API)*

```
Asc.scope.text = "Hello world!";
var connector = docEditor.createConnector();
connector.callCommand(function() {
    var oDocument = Api.GetDocument();
    var oParagraph = Api.CreateParagraph();
    oParagraph.AddText(Asc.scope.text);
```

```
    oDocument.InsertContent([oParagraph]);
}, function() { console.log("callback command"); });
```

下载

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license-pro:8.0.0.1
```

联系方式

QQ：601424688

付费镜像

付费镜像

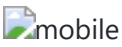
pro 镜像 （￥ 299.00）

::: tip 温馨提示
基于 7.1.1.23 镜像制作
:::

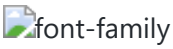
1. 去除 20 链接数限制


点击放大

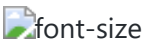
2. 移动端可编辑


点击放大

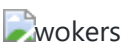
3. 增加中文字体


点击放大
不限以上字体

4. 增加中文字号


点击放大

5. 开启多核线程


点击放大

6. 移动端 cell 文件设置可配置隐藏


cell
点击放大

7. 新增 sdkjsapi (新)

可以调用 `sdk` 的所有 `api` 方法，包含范围（pc、移动端，Documents（docx）、Spreadsheets（xlsx）、Presentations(pptx)）

联系方式

QQ：601424688

付费咨询

付费咨询

疑难解答

- 根据您的业务需要寻求帮助，如果您需要可以通过下面的方式联系，谢谢

联系方式

QQ：601424688

特殊功能定制

特殊功能定制

支持定制服务

- 空格字符显示下划线（￥598）
  
  field-underline

  放大查看

- 可以根据您的业务来定制功能，如果您需要可以通过下面的方式联系，谢谢

联系方式

QQ：601424688

更新日志

更新日志

7.2.0版本

- 默认情况下启用随机密钥的JWT
- 针对较小屏幕的顶部工具栏优化
- 添加了选择“对比暗”或“系统默认”界面主题的功能
- “视图”选项卡中的更多选项。“视图”标签在注释和视图模式下可用
- 重做标题行中的图标
- 能够从标题重命名文件
- 重做设置页
- “关于”页中的确切内部版本号
- 新的实时查看器-能够查看查看模式的更改（DMS应支持）
- 统计页面将显示Live viewer的统计信息
- 新的接口语言-pt pt、zh TW、eu ES、ms MY、hy AM
- 重做颜色选择组件
- 禁用替代菜单的新选项
- 完全重新设计了文档内的搜索
- 特殊粘贴的新热键
- 将“剪切”和“全选”按钮添加到“复制 `Paste”旁边的工具栏`

- 字体引擎的主要改进（针对孟加拉语或僧伽罗语等语言）（仅在文档编辑器和演示文稿编辑器中）
- 能够将表格作为OLE对象插入
- 支持将图像作为项目符号列表，并能够使用它们
- EMF和WMF文件渲染的主要改进
- 全新的插件市场

7.1.0版本

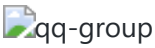
- 支持ARM版本
- 用于插入形状的新菜单（带有最近使用的列表）
- 能够编辑选定形状的点
- 能够打开新的图表类型：金字塔、条形（金字塔）、垂直和水平圆柱体、垂直和垂直圆锥体
- 能够将选定图像裁剪为形状
- 按组对左侧边栏上的注释进行排序
- 能够在输入文件保护密码时查看该密码
- 支持SmartArt对象，无需转换为一组对象
- 新的界面语言：加利戈语/加利西亚语、阿塞拜疆语
- 连接丢失并正在恢复时会显示通知
- 渐变填充图标显示所选颜色

联系方式

联系方式

QQ：601424688

QQ交流群：792389659



资源链接

镜像仓库：https://hub.docker.com/r/knoxzhang/oo-ce-docker-license

服务能力

- 支持定制服务

8.3版笔记

本版特性
基于docker镜像制作，最新版本号: 8.3.0
设置OnlyOffice自动强制保存（8.1以后版本新增，每隔2小时强制保存）
默认OnlyOffice 只会在用户关闭页面后才会向后端发送文件保存请求。 如果用户长时间未关闭页面，则用户对文件的更改将不会及时保存（有可能丢失正在编辑的数据）。
开启多核心，速度更快（进一步优化，打开速度提升50%）。
关闭地址过滤，导入镜像后可以完美通过IP局域网运行（在7.4以上版本默认无法通过IP访问）
完美解锁手机端浏览和编辑（无需设置手机UA为电脑模式）
添加中文字体，加入了常用的十几种字体
支持http/https快速部署（默认开启ssl非常繁琐）
最大支持1G文档在线编辑（7.5以上新增）（默认30M）。
移除logo图标，协作编辑更专注（8.3新增）。
安装docker
已经安装过docker的直接跳过

```
#检测是否已经安装
docker -v
docker
```

如果能够显示版本号则已经安装。

Centos 8.x安装步骤

```
#安装docker容器,并添加阿里云私有加速镜像
wget -O install-docker.sh https://yun.ittel.cn:1443/f/55914227a93d4c50b3ce/?dl=1 && sh install-docker.sh
Ubuntu 20.x/22.x安装步骤
#安装docker容器,并添加阿里云私有加速镜像
wget -O install-docker.sh https://yun.ittel.cn:1443/f/bb8ecd197bed4521a59d/?dl=1 && sudo bash install-docker.sh
```

下载导入镜像（重要）
7.2.X存在致命缺陷（特定情况会提示无法保存），对于运行7.2.X的，建议更新到7.5.0以上版本。
运行onlyoffice7.3及以上镜像，linux内核版本需要大于3.10，Docker版本 >= 21（一般centos7.6以上、ubuntu20以上都符合）。
对于系统比较老，推荐使用7.1.1（比如群辉系统）
onlyoffice 7.1.1 社区版去除限制版镜像下载：
https://pan.quark.cn/s/0194fc51712d

onlyoffice 8.3.0最新版本文末下载(点此前往)

什么是连接器?
连接器 是一个允许从外部源编辑文本文档、电子表格、演示文稿和可填写表单的类， 使用高级"自动化API" 处理文档。

要创建连接器，请使用文档编辑器对象的createConnector方法：

```
var connector = docEditor.createConnector()
onlyoffice-connector
```

运行容器-http方式（二选一）
docker运行下载onlyoffice，此处设置onlyoffice监听端口为8889（按实际情况修改）

```
#--restart=always表示每次开机，只要当docker被设置为开机启动，此onlyoffice就会自动运行
#-d 表示后台运行
#-i:  交互式操作。
#-t:  终端。
#-p 8889:80  宿主机的8889端口映射到容器的80端口
# ittel/onlyoffice:8.1.0zh-cn 改成实际的容器名称
#运行onlyoffice 8.1.0
docker run -i -t -d -e JWT_ENABLED=false --restart=always \
-p 8889:80 ittel/onlyoffice:8.1.0zh-cn
```

访问下 `http://IP:8889` 如果出现欢迎页面，则安装成功

运行容器-https方式（二选一）
1、上传SSL证书
将SSL证书.key文件和.crt文件改名为 `onlyoffice.key` 和 `onlyoffice.crt`
如果没有.crt则把.pem改名为onlyoffice.crt

新建证书目录

```
[root@ecs-a869 yum.repos.d]# mkdir -pv /home/certs
将key和crt上传至/home/certs目录下，并检查是否上传
[root@ecs-a869 yum.repos.d]# cd /home/certs
[root@ecs-a869 certs]# ls
onlyoffice.crt  onlyoffice.key
```

2、配置docker容器里nginx

#下载配置模板文件
`wget -O /home/certs/ds.conf`

`https://yun.ittel.cn:1443/f/1a877fe2e9794a2fb5a7/?dl=1`

```
#编辑配置文件
[root@ecs-a869 yum.repos.d]#vi /home/certs/ds.conf
修改域名为实际域名，其他地方不需要修改
include /etc/nginx/includes/http-common.conf;
server {
  listen 0.0.0.0:80;
  listen [::]:80 default_server;
  server_tokens off;

  include /etc/nginx/includes/ds-*.conf;
}
server {
    listen 443 ssl;
    ssl_certificate /mnt/certs/onlyoffice.crt;
    ssl_certificate_key /mnt/certs/onlyoffice.key;
    server_name yun.ittel.cn;  #改成自己的域名
.....
......
```

3、docker运行下载onlyoffice，此处设置onlyoffice监听端口为8889

```
#根据实际情况修改端口即可，这里使用的是8889端口
docker run -i -t -d --restart=always -e JWT_ENABLED=false \
-e USE_UNAUTHORIZED_STORAGE=true \
-p 8889:443 -v /home/certs:/mnt/certs \
-v /home/certs/ds.conf:/etc/nginx/conf.d/ds.conf \
ittel/onlyoffice:7.6.0zh-cn
#--restart=always表示每次开机，只要当docker被设置为开机启动，此onlyoffice就会自动运行
#-v 指挂载主机目到容器中
#关闭证书校验，针对自签证书
#-e USE_UNAUTHORIZED_STORAGE=true
# ittel/onlyoffice:7.6.0zh-cn 换成实际魔改容器名称
```

4、SSL访问测试

访问 `https://IP或者域名:8889/welcome/`

后续可在nextcloud，ZZoffice，SeaFile 结合使用在线编辑

效果图
主界面：

支持多人同时编辑

编译添加了常用几种中文字体

问题排查(点击展开)
修改文件限制大小
onlyoffice默认可以打开30M左右文件，超过会提示文件大小限制，可以按照以下方法增大（魔改镜像已默认修改）。

修改容器default.json，目录：

容器下的： `/etc/onlyoffice/documentserver/default.json`

可以增大下面几个标记的数值，最后记得重启容器。

页面提示下载失败，容器日志提示 is not allowed. Because, It is private IP address.
查看容器日志提示：

```
nodeJS - error
downloadFile:url=http://xxx:8080/Products/Files/HttpHandlers/filehandler.ashx?action=stream&fileid=28&stream_auth=42443xxxXJ56L2TBSE&X-REWRITER-URL=https%3a%2f%2fxxx%3a443;attempt=3;code:undefined;connect:undefined
Error: DNS lookup 192.168.1.41(family:4, host:onlyoffice) is not allowed. Because, It is private IP address.
```

问题分析
在onlyoffice 7.4中新增了一项安全功能，如果局域网IP访问，必须开启JWT。（在魔改版镜像中已默认关闭）

开发者回复

解决方法：
直接使用上面编译好的魔改镜像或者老老实实开启jwt。

添加自定义字体库
很多同学收藏了很多字体，想加入onlyoffice在线编辑器里，这里讲述下基于上述魔改镜像添加自定义字体的方法。

备注：字体不是越多越好，一个字体文件小则500KB，大的要20M，字体太多会导致文档加载时间变长（特别是在带宽很小情况下）。
1、把需要添加的字体库包传到服务器上
这里也整理了一份常用的中文字体库，同学们按需下载即可，也可以把你的字体库发我，我会放入下面的共享里。
下载完后上传到服务器上面。
中文常用字体库下载

2、把字体拷贝到镜像
查看容器ID方法，第一行字符串就是ID值

```
docker ps|grep onlyoffice
```

这里以添加方正小标宋字体为例

a、上传字体到容器内（这里无欢不爱把字体上传到服务器的/home/fonts/目录下）

格式：
```
docker cp /home/fonts/* <容器ID>:/usr/share/fonts/
```

#这里执行
```
docker cp /home/fonts/* 81d98fa20029:/usr/share/fonts/
```
3、生成字体文件
格式：

```
docker exec -i <容器ID> /usr/bin/documentserver-generate-allfonts.sh
```

执行，等待执行完毕即可

```
docker exec -i 81d98fa20029 /usr/bin/documentserver-generate-allfonts.sh
```

4、验证是否添加成功
刷新文档，验证是否有刚添加的字体，如果没有显示，建议清理下浏览器缓存。

Docker映射端口后，浏览器无法访问欢迎界面
创建Docker的onlyoffice容器正常，也已经起来，防火墙也开放了对应端口，但是浏览器就是无法访问。

原因与解决方法：
谷歌浏览器、火狐浏览器、新版Edge浏览器预留了部分端口，避免使用下面的端口号。

```
端口    原因
1    tcpmux
7    echo
9    discard
11    systat
13    daytime
15    netstat
17    qotd
19    chargen
20    ftp data
21    ftp access
22    ssh
23    telnet
```

```
25    smtp
37    time
42    name
43    nicname
53    domain
77    priv-rjs
79    finger
87    ttylink
95    supdup
101    hostriame
102    iso-tsap
103    gppitnp
104    acr-nema
109    pop2
110    pop3
111    sunrpc
113    auth
115    sftp
117    uucp-path
119    nntp
123    NTP
135    loc-srv /epmap
139    netbios
143    imap2
179    BGP
389    ldap
465    smtp+ssl
512    print / exec
513    login
514    shell
515    printer
526    tempo
530    courier
531    chat
532    netnews
540    uucp
556    remotefs
563    nntp+ssl
587    stmp?
601    ??
636    ldap+ssl
993    ldap+ssl
995    pop3+ssl
2049    nfs
3659    apple-sasl / PasswordServer
```

```
4045    lockd
6000    X11
6665    Alternate IRC [Apple addition]
6666    Alternate IRC [Apple addition]
6667    Standard IRC [Apple addition]
6668    Alternate IRC [Apple addition]
6669    Alternate IRC [Apple addition]
```

在线打开office文档，提示无法保存和下载
问题现象：
如果onlyoffice容器和seafile、nextcloud不在同一主机，在线编辑会提示下载失败。

原因分析：
docker容器内默认是无法访问外网的，如果容器和seafile等服务不在同一主机，则容器需要主动联网。但是linux默认策略是不会转发包的，所以需要开启主机NAT转发。

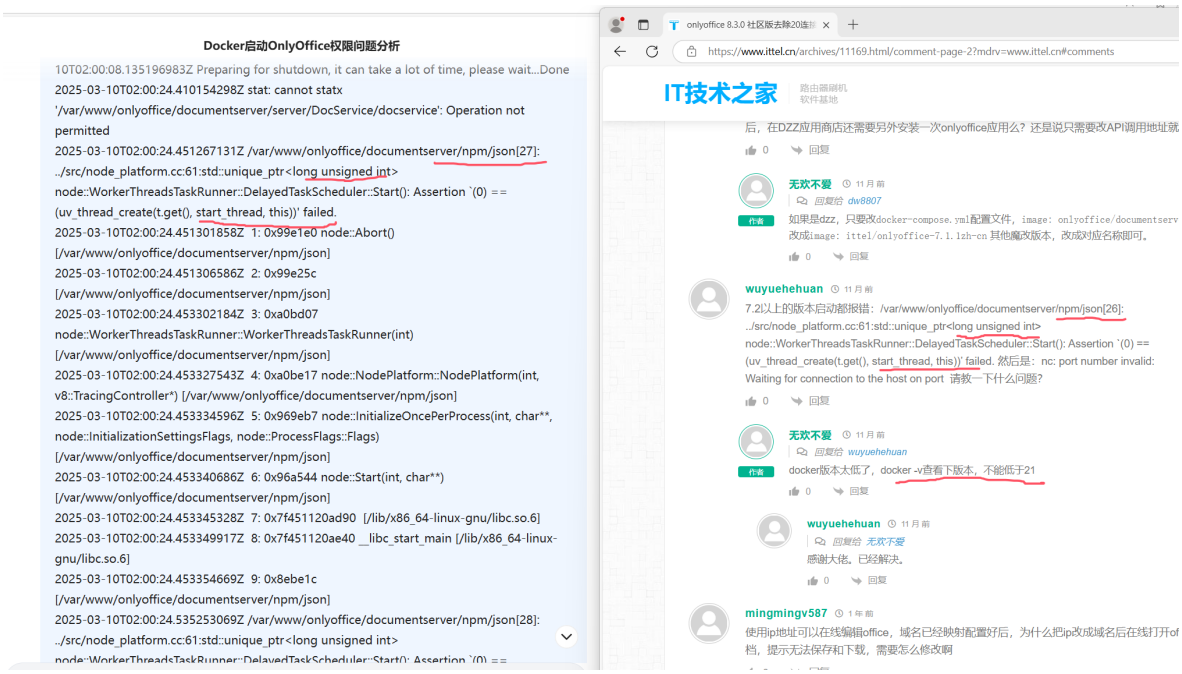
解决办法：

```
#编辑系统设置
vi /etc/sysctl.conf
#添加以下内容到sysctl.conf
#添加完毕记得 :wq 保存
net.ipv4.ip_forward = 1
#最后重启主机
```

Onlyoffice 魔改Docker镜像下载
运行onlyoffice7.3及以上镜像，linux内核版本需要大于3.10，Docker版本 >= 20（一般centos7.6以上、ubuntu20以上都符合）。
对于系统比较老，推荐使用7.1.1（比如群辉系统）

centos升级和docker升级

缘由：因为8.2版OO在docker下运行，无法接入api，用命令查看，发现提示以下错误，网友博主提示是因为docker版本低，那么只好升级docker，升级docker就要升级centos，升级centos后，删除已有docker，最好做一下镜像的备份，如果容器有修改，将容器导出为镜像保存。删除docker后，直接安装docker出错，需要清理，再安装。本文就是这么个过程。



首先，centos的升级有下面3种，我们选择第二种。
一、yum安装；二、安装指定内核版本rpm；三、源码安装。本文是第二种方式，rpm安装。

1 手动升级centos7内核（版本自行选择，亲测内核下载链接有效）

lianglove7

一、下载centos内核rpm安装包

查半天也没查出5.10.222是什么版本，反正能用最新的docker。
1.内核下载地址：https://dl.lamp.sh/kernel/el7/

```
wget https://dl.lamp.sh/kernel/el7/kernel-ml-5.10.222-1.el7.x86_64.rpm
wget https://dl.lamp.sh/kernel/el7/kernel-ml-devel-5.10.222-1.el7.x86_64.rpm
```

二、安装RPM包

```
yum localinstall -y kernel-ml-5.10.222-1.el7.x86_64.rpm kernel-ml-devel-5.10.222-1.el7.x86_64.rpm
#安装完毕后查看系统可用启动内核
awk -F\' '$1=="menuentry " {print  $2}' /etc/grub2.cfg
```

三、修改默认启动用的内核版本

默认启动项由/etc/default/grub中的GRUB_DEFAULT控制，可使用grub2-editenv list查看：

```
[root@localhost ~]# grub2-editenv list
saved_entry=CentOS Linux (3.10.0-1160.el7.x86_64) 7 (Core)
```

通过grub2-editenv list查看当前使用的内核，通过grub2-set-default命令修改启动用内核（通过内核名称设置）

```
[root@localhost ~]# grub2-editenv list
saved_entry=CentOS Linux (3.10.0-1160.el7.x86_64) 7 (Core)
[root@localhost ~]# grub2-set-default 'CentOS Linux (5.10.222-1.el7.x86_64) 7 (Core)'
[root@localhost ~]# grub2-editenv list
saved_entry=CentOS Linux (5.10.222-1.el7.x86_64) 7 (Core)
```

重启并查看内核版本

```
[root@localhost ~]# reboot
#重启后使用命令查看当前内核版本
[root@localhost ~]# uname -r
5.10.222-1.el7.x86_64
```

原文链接：https://blog.csdn.net/lianglove7/article/details/143734553

四、实际操作过程记录

```
[root@localhost src]# wget https://dl.lamp.sh/kernel/el7/kernel-ml-5.10.222-1.el7.x86_64.rpm
--2025-03-11 10:38:33--  https://dl.lamp.sh/kernel/el7/kernel-ml-5.10.222-1.el7.x86_64.rpm
正在解析主机 dl.lamp.sh (dl.lamp.sh)... 192.99.37.188, 2607:5300:60:4cbc::
```

```
1
正在连接 dl.lamp.sh (dl.lamp.sh)|192.99.37.188|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：57507844 (55M) [application/x-rpm]
正在保存至：“kernel-ml-5.10.222-1.el7.x86_64.rpm”

100%[==========================================================================
==========================================================>] 57,507,844  3.
24MB/s 用时 18s

2025-03-11 10:38:55 (3.10 MB/s) - 已保存 “kernel-ml-5.10.222-1.el7.x86_64.
rpm” [57507844/57507844])

[root@localhost src]#
[root@localhost src]# wget https://dl.lamp.sh/kernel/el7/kernel-ml-devel-
5.10.222-1.el7.x86_64.rpm
--2025-03-11 10:39:16--  https://dl.lamp.sh/kernel/el7/kernel-ml-devel-5.
10.222-1.el7.x86_64.rpm
正在解析主机 dl.lamp.sh (dl.lamp.sh)... 192.99.37.188, 2607:5300:60:4cbc::
1
正在连接 dl.lamp.sh (dl.lamp.sh)|192.99.37.188|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度：14178892 (14M) [application/x-rpm]
正在保存至：“kernel-ml-devel-5.10.222-1.el7.x86_64.rpm”

100%[==========================================================================
==========================================================>] 14,178,892  2.
66MB/s 用时 6.0s

2025-03-11 10:39:23 (2.24 MB/s) - 已保存 “kernel-ml-devel-5.10.222-1.el7.x
86_64.rpm” [14178892/14178892])

[root@localhost src]# yum localinstall -y kernel-ml-5.10.222-1.el7.x86_6
4.rpm kernel-ml-devel-5.10.222-1.el7.x86_64.rpm
已加载插件：fastestmirror, langpacks
正在检查 kernel-ml-5.10.222-1.el7.x86_64.rpm: kernel-ml-5.10.222-1.el7.x86
_64
kernel-ml-5.10.222-1.el7.x86_64.rpm 将被安装
正在检查 kernel-ml-devel-5.10.222-1.el7.x86_64.rpm: kernel-ml-devel-5.10.2
22-1.el7.x86_64
kernel-ml-devel-5.10.222-1.el7.x86_64.rpm 将被安装
正在解决依赖关系
--> 正在检查事务
---> 软件包 kernel-ml.x86_64.0.5.10.222-1.el7 将被 安装
---> 软件包 kernel-ml-devel.x86_64.0.5.10.222-1.el7 将被 安装
```

```
--> 解决依赖关系完成
Could not retrieve mirrorlist http://mirrorlist.centos.org/?release=7&arc
h=x86_64&repo=os&infra=stock error was
14: curl#6 - "Could not resolve host: mirrorlist.centos.org; 未知的名称或
服务"
http://mirrors.163.com/centos/7.9.2009/os/x86_64/repodata/repomd.xml: [Er
rno 14] HTTP Error 404 - Not Found
正在尝试其它镜像。
To address this issue please refer to the below wiki article

https://wiki.centos.org/yum-errors

If above article doesn't help to resolve this issue please use https://bu
gs.centos.org/.

base/7/x86_64
| 3.6 kB  00:00:00
docker-ce-stable/x86_64
| 3.5 kB  00:00:00
docker-ce-stable/x86_64/primary_db
| 152 kB  00:00:00
epel/x86_64/metalink
| 5.1 kB  00:00:00
Could not retrieve mirrorlist http://mirrorlist.centos.org/?release=7&arc
h=x86_64&repo=extras&infra=stock error was
14: curl#6 - "Could not resolve host: mirrorlist.centos.org; 未知的名称或
服务"
http://mirrors.tuna.tsinghua.edu.cn/centos/7.9.2009/extras/x86_64/repodat
a/repomd.xml: [Errno 14] HTTP Error 404 - Not Found
正在尝试其它镜像。
extras/7/x86_64
| 2.9 kB  00:00:00
Could not retrieve mirrorlist http://mirrorlist.centos.org/?release=7&arc
h=x86_64&repo=updates&infra=stock error was
14: curl#6 - "Could not resolve host: mirrorlist.centos.org; 未知的名称或
服务"
updates/7/x86_64
| 2.9 kB  00:00:00

依赖关系解决

================================================================================
================================================================================
====================
 Package                                架构                                版本
```

```
源                                                                    大小
==========================================================================
==========================================================================
====================
正在安装:
 kernel-ml                          x86_64                    5.10.222-
1.el7                      /kernel-ml-5.10.222-1.el7.x86_64
256 M
 kernel-ml-devel                    x86_64                    5.10.222-
1.el7                      /kernel-ml-devel-5.10.222-1.el7.x86_64
53 M

事务概要
==========================================================================
==========================================================================
====================
安装  2 软件包

总计: 308 M
安装大小: 308 M
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  正在安装    : kernel-ml-devel-5.10.222-1.el7.x86_64
1/2
  正在安装    : kernel-ml-5.10.222-1.el7.x86_64
2/2
  验证中      : kernel-ml-5.10.222-1.el7.x86_64
1/2
  验证中      : kernel-ml-devel-5.10.222-1.el7.x86_64
2/2

已安装:
  kernel-ml.x86_64 0:5.10.222-1.el7
kernel-ml-devel.x86_64 0:5.10.222-1.el7

完毕!
[root@localhost src]#
[root@localhost src]# awk -F\' '$1=="menuentry " {print  $2}' /etc/grub2.
cfg
CentOS Linux (5.10.222-1.el7.x86_64) 7 (Core)
CentOS Linux (3.10.0-862.el7.x86_64) 7 (Core)
CentOS Linux (0-rescue-a9324175f378456fbe6bda50a0e45a75) 7 (Core)
```

```
[root@localhost src]# grub2-editenv list
saved_entry=CentOS Linux (3.10.0-862.el7.x86_64) 7 (Core)
[root@localhost src]# grub2-set-default 'CentOS Linux (5.10.222-1.el7.x86_64) 7 (Core)'
[root@localhost src]# grub2-editenv list
saved_entry=CentOS Linux (5.10.222-1.el7.x86_64) 7 (Core)
[root@localhost src]# reboot
```

https://www.cnblogs.com/guuyoog/p/17666449.html

2 解决yum报错：错误：Invalid version flag: if

1. 备份默认的yum：

```
mv /etc/yum.repos.d /etc/yum.repos.d.backup
```

2. 设置新的yum目录：

```
mkdir /etc/yum.repos.d
```

3. 下载阿里yum配置到该目录中，用wget，进行下载

```
wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
```

4. 重建缓存

```
yum clean all
yum makecache
```

5. 升级所有包

```
yum upgrade -y
```

下面这2步没做

`yum -y update` ：升级所有包同时，也升级软件和系统内核；（时间比较久）

`yum -y upgrade` ：只升级所有包，不升级软件和系统内核，软件和内核保持原样

原文链接：https://blog.csdn.net/weixin_56781779/article/details/132258636

3 CentOS安装Docker

Docker CE 支持 64 位版本 CentOS 7，并且要求内核版本不低于 3.10， CentOS 7 满足最低内核的要求，所以我们在CentOS 7安装Docker。

3.1.卸载（可选）

如果之前安装过旧版本的Docker，可以使用下面命令卸载：

```
yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-selinux \
                  docker-engine-selinux \
                  docker-engine \
                  docker-ce
```

3.2.安装docker

首先需要大家虚拟机联网，安装yum工具

```
yum install -y yum-utils \
           device-mapper-persistent-data \
           lvm2 --skip-broken
```

然后更新本地镜像源：

```
# 设置docker镜像源
yum-config-manager \
    --add-repo \
    https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
sed -i 's/download.docker.com/mirrors.aliyun.com\/docker-ce/g' /etc/yum.repos.d/docker-ce.repo
yum makecache fast
```

然后输入命令：

```
yum install -y docker-ce
```

docker-ce为社区免费版本。稍等片刻，docker即可安装成功。

3.3.启动docker

Docker应用需要用到各种端口，逐一去修改防火墙设置。非常麻烦，因此建议大家直接关闭防火墙!
启动docker前，一定要关闭防火墙后！！
启动docker前，一定要关闭防火墙后！！
启动docker前，一定要关闭防火墙后！！

```
# 关闭防火墙
`systemctl stop firewalld`
# 禁止开机启动防火墙
`systemctl disable firewalld`
查看docker是否启动
`docker images`
```

通过命令启动docker：

```
systemctl start docker  # 启动docker服务
systemctl stop docker  # 停止docker服务
systemctl restart docker  # 重启docker服务
```

如果启动时出现错误
提示：Job for docker.service failed because the control process exited with error code. See “systemctl status docker.service” and “journalctl -xe” for details.

```
# 进入docker目录
`cd /etc/docker/`
# 修改daemon的类型
`mv daemon.json daemon.conf`
# 重启docker
`systemctl restart docker`
然后输入命令，可以查看docker版本：
`docker -v`
```

3.4.配置镜像加速

阿里云
docker官方镜像仓库网速较差，我们需要设置国内镜像服务：
参考阿里云的镜像加速文档：

https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors
腾讯云
配置使用腾讯云 Docker 镜像源加速镜像下载（可跳过）
执行以下命令，打开 /etc/docker/daemon.json 配置文件
`vim /etc/docker/daemon.json`
按 i 切换至编辑模式，添加以下内容，并保存。
`{ "registry-mirrors": [ "https://mirror.ccs.tencentyun.com" ] }`
重启Docker `systemctl restart docker`
我的安装过程如下：

```
替代:
  caribou.x86_64 0:0.4.21-1.el7                                 caribou-gtk2
-module.x86_64 0:0.4.21-1.el7       caribou-gtk3-module.x86_64 0:0.4.21-
1.el7
  gnome-dictionary-libs.x86_64 0:3.20.0-1.el7                   gnome-tweak-
tool.noarch 0:3.22.0-1.el7          grub2.x86_64 1:2.02-0.65.el7.centos.2
  grub2-tools.x86_64 1:2.02-0.65.el7.centos.2                   iwl7265-firm
ware.noarch 0:22.0.7.0-62.el7       mesa-libwayland-egl.x86_64 0:17.2.3-
8.20171019.el7
  pyatspi.noarch 0:2.20.3-1.el7                                 python2-cari
bou.noarch 0:0.4.21-1.el7           urw-fonts.noarch 0:2.4-16.el7
  webkitgtk4-plugin-process-gtk2.x86_64 0:2.16.6-6.el7

完毕!
[root@localhost ~]# sudo yum install -y yum-utils
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
软件包 yum-utils-1.1.31-54.el7_8.noarch 已安装并且是最新版本
无须任何处理
[root@localhost ~]# sudo yum-config-manager --add-repo https://download.d
ocker.com/linux/centos/docker-ce.repo
已加载插件：fastestmirror, langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to
/etc/yum.repos.d/docker-ce.repo
Could not fetch/save url https://download.docker.com/linux/centos/docker-
ce.repo to file /etc/yum.repos.d/docker-ce.repo: [Errno 14] curl#35 - "TC
P connection reset by peer"
[root@localhost ~]# sudo yum install docker-ce docker-ce-cli containerd.i
o -y
已加载插件：fastestmirror, langpacks
```

```
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
没有可用软件包 docker-ce。
没有可用软件包 docker-ce-cli。
没有可用软件包 containerd.io。
错误：无须任何处理

[root@localhost ~]# sudo yum install -y yum-utils device-mapper-persisten
t-data lvm2
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
软件包 yum-utils-1.1.31-54.el7_8.noarch 已安装并且是最新版本
软件包 device-mapper-persistent-data-0.8.5-3.el7_9.2.x86_64 已安装并且是最
新版本
软件包 7:lvm2-2.02.187-6.el7_9.5.x86_64 已安装并且是最新版本
无须任何处理
[root@localhost ~]# sudo yum-config-manager --add-repo https://download.d
ocker.com/linux/centos/docker-ce.repo
已加载插件：fastestmirror, langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to
/etc/yum.repos.d/docker-ce.repo
Could not fetch/save url https://download.docker.com/linux/centos/docker-
ce.repo to file /etc/yum.repos.d/docker-ce.repo: [Errno 14] curl#35 - "TC
P connection reset by peer"
[root@localhost ~]# docker
bash: docker: 未找到命令...

[root@localhost ~]# yum-config-manager \
>     --add-repo \
>     https://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo

sed -i 's/download.docker.com/mirrors.aliyun.com\/docker-ce/g' /etc/yum.r
epos.d/docker-ce.repo

yum makecache fast
已加载插件：fastestmirror, langpacks
adding repo from: https://mirrors.aliyun.com/docker-ce/linux/centos/docke
r-ce.repo
grabbing file https://mirrors.aliyun.com/docker-ce/linux/centos/docker-c
```

```
e.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo

[root@localhost ~]# sed -i 's/download.docker.com/mirrors.aliyun.com\/doc
ker-ce/g' /etc/yum.repos.d/docker-ce.repo

[root@localhost ~]# yum makecache fast
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
base
| 3.6 kB  00:00:00
docker-ce-stable
| 3.5 kB  00:00:00
extras
| 2.9 kB  00:00:00
updates
| 2.9 kB  00:00:00
元数据缓存已建立
[root@localhost ~]# yum install -y docker-ce
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
正在解决依赖关系
--> 正在检查事务
---> 软件包 docker-ce.x86_64.3.26.1.4-1.el7 将被 安装
--> 正在处理依赖关系 container-selinux >= 2:2.74，它被软件包 3:docker-ce-26.
1.4-1.el7.x86_64 需要
--> 正在处理依赖关系 containerd.io >= 1.6.24，它被软件包 3:docker-ce-26.1.4-
1.el7.x86_64 需要
--> 正在处理依赖关系 docker-ce-cli，它被软件包 3:docker-ce-26.1.4-1.el7.x86_
64 需要
--> 正在处理依赖关系 docker-ce-rootless-extras，它被软件包 3:docker-ce-26.1.
4-1.el7.x86_64 需要
--> 正在检查事务
---> 软件包 container-selinux.noarch.2.2.119.2-1.911c772.el7_8 将被 安装
---> 软件包 containerd.io.x86_64.0.1.6.33-3.1.el7 将被 安装
---> 软件包 docker-ce-cli.x86_64.1.26.1.4-1.el7 将被 安装
--> 正在处理依赖关系 docker-buildx-plugin，它被软件包 1:docker-ce-cli-26.1.4
-1.el7.x86_64 需要
--> 正在处理依赖关系 docker-compose-plugin，它被软件包 1:docker-ce-cli-26.1.
```

```
4-1.el7.x86_64 需要
---> 软件包 docker-ce-rootless-extras.x86_64.0.26.1.4-1.el7 将被 安装
--> 正在处理依赖关系 fuse-overlayfs >= 0.7，它被软件包 docker-ce-rootless-ex
tras-26.1.4-1.el7.x86_64 需要
--> 正在处理依赖关系 slirp4netns >= 0.4，它被软件包 docker-ce-rootless-extra
s-26.1.4-1.el7.x86_64 需要
--> 正在检查事务
---> 软件包 docker-buildx-plugin.x86_64.0.0.14.1-1.el7 将被 安装
---> 软件包 docker-compose-plugin.x86_64.0.2.27.1-1.el7 将被 安装
---> 软件包 fuse-overlayfs.x86_64.0.0.7.2-6.el7_8 将被 安装
--> 正在处理依赖关系 libfuse3.so.3(FUSE_3.2)(64bit)，它被软件包 fuse-overlay
fs-0.7.2-6.el7_8.x86_64 需要
--> 正在处理依赖关系 libfuse3.so.3(FUSE_3.0)(64bit)，它被软件包 fuse-overlay
fs-0.7.2-6.el7_8.x86_64 需要
--> 正在处理依赖关系 libfuse3.so.3()(64bit)，它被软件包 fuse-overlayfs-0.7.2
-6.el7_8.x86_64 需要
---> 软件包 slirp4netns.x86_64.0.0.4.3-4.el7_8 将被 安装
--> 正在检查事务
---> 软件包 fuse3-libs.x86_64.0.3.6.1-4.el7 将被 安装
--> 解决依赖关系完成

依赖关系解决

================================================================================
================================================================================
====================
 Package                                    架构
版本                                        源
大小
================================================================================
================================================================================
====================
正在安装:
 docker-ce                                  x86_64
3:26.1.4-1.el7                              docker-ce-stable
27 M
为依赖而安装:
 container-selinux                          noarch
2:2.119.2-1.911c772.el7_8                   extras
40 k
 containerd.io                              x86_64
1.6.33-3.1.el7                              docker-ce-stable
35 M
 docker-buildx-plugin                       x86_64
0.14.1-1.el7                                docker-ce-stable
```

```
14 M
 docker-ce-cli                                    x86_64
1:26.1.4-1.el7                                     docker-ce-stable
15 M
 docker-ce-rootless-extras                        x86_64
26.1.4-1.el7                                       docker-ce-stable
9.4 M
 docker-compose-plugin                            x86_64
2.27.1-1.el7                                       docker-ce-stable
13 M
 fuse-overlayfs                                   x86_64
0.7.2-6.el7_8                                      extras
54 k
 fuse3-libs                                       x86_64
3.6.1-4.el7                                        extras
82 k
 slirp4netns                                      x86_64
0.4.3-4.el7_8                                      extras
81 k

事务概要
================================================================================
================================================================================
====================
安装  1 软件包 (+9 依赖软件包)

总下载量：114 M
安装大小：401 M
Downloading packages:
No Presto metadata available for docker-ce-stable
(1/10): container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm
|  40 kB  00:00:04
(2/10): docker-buildx-plugin-0.14.1-1.el7.x86_64.rpm
|  14 MB  00:00:12
(3/10): containerd.io-1.6.33-3.1.el7.x86_64.rpm
|  35 MB  00:00:25
(4/10): docker-ce-26.1.4-1.el7.x86_64.rpm
|  27 MB  00:00:15
(5/10): docker-ce-rootless-extras-26.1.4-1.el7.x86_64.rpm
| 9.4 MB  00:00:05
(6/10): docker-ce-cli-26.1.4-1.el7.x86_64.rpm
|  15 MB  00:00:08
(7/10): fuse-overlayfs-0.7.2-6.el7_8.x86_64.rpm
|  54 kB  00:00:03
(8/10): fuse3-libs-3.6.1-4.el7.x86_64.rpm
```

```
|  82 kB  00:00:03
(9/10): slirp4netns-0.4.3-4.el7_8.x86_64.rpm
|  81 kB  00:00:00
(10/10): docker-compose-plugin-2.27.1-1.el7.x86_64.rpm
|  13 MB  00:00:05
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------
总计
2.9 MB/s | 114 MB  00:00:38
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  正在安装    : 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
1/10
  正在安装    : containerd.io-1.6.33-3.1.el7.x86_64
2/10
  正在安装    : docker-buildx-plugin-0.14.1-1.el7.x86_64
3/10
  正在安装    : slirp4netns-0.4.3-4.el7_8.x86_64
4/10
  正在安装    : fuse3-libs-3.6.1-4.el7.x86_64
5/10
  正在安装    : fuse-overlayfs-0.7.2-6.el7_8.x86_64
6/10
  正在安装    : docker-compose-plugin-2.27.1-1.el7.x86_64
7/10
  正在安装    : 1:docker-ce-cli-26.1.4-1.el7.x86_64
8/10
  正在安装    : docker-ce-rootless-extras-26.1.4-1.el7.x86_64
9/10
  正在安装    : 3:docker-ce-26.1.4-1.el7.x86_64
10/10
  验证中      : docker-compose-plugin-2.27.1-1.el7.x86_64
1/10
  验证中      : fuse3-libs-3.6.1-4.el7.x86_64
2/10
  验证中      : fuse-overlayfs-0.7.2-6.el7_8.x86_64
3/10
  验证中      : slirp4netns-0.4.3-4.el7_8.x86_64
4/10
  验证中      : 2:container-selinux-2.119.2-1.911c772.el7_8.noarch
5/10
  验证中      : containerd.io-1.6.33-3.1.el7.x86_64
```

```
6/10
  验证中      : 3:docker-ce-26.1.4-1.el7.x86_64
7/10
  验证中      : 1:docker-ce-cli-26.1.4-1.el7.x86_64
8/10
  验证中      : docker-ce-rootless-extras-26.1.4-1.el7.x86_64
9/10
  验证中      : docker-buildx-plugin-0.14.1-1.el7.x86_64
10/10

已安装:
  docker-ce.x86_64 3:26.1.4-1.el7

作为依赖被安装:
  container-selinux.noarch 2:2.119.2-1.911c772.el7_8        containerd.i
o.x86_64 0:1.6.33-3.1.el7                   docker-buildx-plugin.x86_64 0:
0.14.1-1.el7
  docker-ce-cli.x86_64 1:26.1.4-1.el7                       docker-ce-roo
tless-extras.x86_64 0:26.1.4-1.el7        docker-compose-plugin.x86_64 0:
2.27.1-1.el7
  fuse-overlayfs.x86_64 0:0.7.2-6.el7_8                     fuse3-libs.x8
6_64 0:3.6.1-4.el7                        slirp4netns.x86_64 0:0.4.3-4.el
7_8

完毕!
[root@localhost ~]# yum install -y docker-ce
已加载插件: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirrors.aliyun.com
 * extras: mirrors.aliyun.com
 * updates: mirrors.aliyun.com
软件包 3:docker-ce-26.1.4-1.el7.x86_64 已安装并且是最新版本
无须任何处理
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.ser
vice.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.ser
vice.
[root@localhost ~]# docker images
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is th
e docker daemon running?
[root@localhost ~]# systemctl start docker
```

4 协作页面提示下载失败

页面打开文档协作，提示 `下载失败` ，是因为OO里的default文件需要更改允许下载私有ip服务器上的文件

`Error: DNS lookup 192.168.100.37(family:undefined, host:undefined) is not allowed. Because, It is private IP address.`

https://www.cnblogs.com/wxwgk/p/17982561

https://blog.csdn.net/m0_53401243/article/details/133869439

解决方式：需要编辑配置文件允许私有ip通过：

编辑docker中 `/etc/onlyoffice/documentserver/default.json` 里的内容：

搜索并修改以下字段为 `true` ：

```
"request-filtering-agent" : {
    "allowPrivateIPAddress": true,
    "allowMetaIPAddress": true
},
```

也可以将文件复制出来再进行编辑：

docker cp onlyoffice:/etc/onlyoffice/documentserver/default.json ./

vim default.json

docker cp ./default.json onlyoffice:/etc/onlyoffice/documentserver/default.json

更改后，需要重启容器后恢复正常。

具体操作步骤详见下面

```
[root@localhost engineercms]# docker logs -f 826
Generating WOPI private key...Done
Generating WOPI public key...Done
[ OK ]rting PostgreSQL 14 database server
[ OK ]rting RabbitMQ Messaging Server rabbitmq-server
psql:/var/www/onlyoffice/documentserver/server/schema/postgresql/removetbl.sql:4: NOTICE:  table "doc_callbacks" does not exist, skipping
psql:/var/www/onlyoffice/documentserver/server/schema/postgresql/removetbl.sql:9: NOTICE:  function merge_db(pg_catalog.varchar,int2,int8,pg_catalog.timestamp,pg_catalog.varchar,int8,int8) does not exist, skipping
psql:/var/www/onlyoffice/documentserver/server/schema/postgresql/removetbl.sql:10: NOTICE:  function merge_db(pg_catalog.varchar,int2,int4,pg_catalog.timestamp,pg_catalog.varchar,int4,int4) does not exist, skipping
psql:/var/www/onlyoffice/documentserver/server/schema/postgresql/removetbl.sql:11: NOTICE:  function merge_db(pg_catalog.varchar,int2,int4,pg_catalog.timestamp,int4,int4,text,text) does not exist, skipping
Starting supervisor: supervisord.
```

```
[ OK ]rting periodic command scheduler cron
[ OK ]rting nginx nginx
Generating AllFonts.js, please wait...Done
Generating presentation themes, please wait...Done
Generating js caches, please wait...Done
/usr/bin/documentserver-generate-allfonts.sh: 65: /var/www/onlyoffice/doc
umentserver/sdkjs/slide/themes/themes.js.gz: not found
ds:docservice: stopped
ds:docservice: started
ds:converter: stopped
ds:converter: started
[ OK ]oading nginx configuration nginx
Installing plugins, please wait...Done
[ OK ]oading nginx configuration nginx
JWT is enabled by default. A random secret is generated automatically. Ru
n the command "docker exec 826ce03a8378 sudo documentserver-jwt-status.s
h" to get information about JWT.
==> /var/log/onlyoffice/documentserver/converter/err.log <==

==> /var/log/onlyoffice/documentserver/converter/out.log <==
[2025-03-11T06:16:58.020] [WARN] [localhost] [docId] [userId] nodeJS - nu
m of CPUs: 4; availableParallelism: undefined
[2025-03-11T06:16:58.035] [WARN] [localhost] [docId] [userId] nodeJS - up
date cluster with 1 workers
[2025-03-11T06:16:58.065] [WARN] [localhost] [docId] [userId] nodeJS - wo
rker 786 started.
[2025-03-11T06:16:58.079] [WARN] [localhost] [docId] [userId] nodeJS - nu
m of CPUs: 4; availableParallelism: undefined
[2025-03-11T06:16:58.080] [WARN] [localhost] [docId] [userId] nodeJS - up
date cluster with 1 workers
[2025-03-11T06:17:51.380] [WARN] [localhost] [docId] [userId] nodeJS - nu
m of CPUs: 4; availableParallelism: undefined
[2025-03-11T06:17:51.384] [WARN] [localhost] [docId] [userId] nodeJS - up
date cluster with 1 workers
[2025-03-11T06:17:51.393] [WARN] [localhost] [docId] [userId] nodeJS - wo
rker 909 started.
[2025-03-11T06:17:51.394] [WARN] [localhost] [docId] [userId] nodeJS - nu
m of CPUs: 4; availableParallelism: undefined
[2025-03-11T06:17:51.395] [WARN] [localhost] [docId] [userId] nodeJS - up
date cluster with 1 workers

==> /var/log/onlyoffice/documentserver/docservice/err.log <==

==> /var/log/onlyoffice/documentserver/docservice/out.log <==
[2025-03-11T06:16:59.495] [WARN] [localhost] [docId] [userId] nodeJS - Ex
```

```
press server starting...
[2025-03-11T06:16:59.497] [WARN] [localhost] [docId] [userId] nodeJS - Se
t services.CoAuthoring.token.enable.browser, services.CoAuthoring.token.e
nable.request.inbox, services.CoAuthoring.token.enable.request.outbox in
the Document Server config to prevent an unauthorized access to your docu
ments and the substitution of important parameters in Document Server req
uests.
[2025-03-11T06:16:59.498] [WARN] [localhost] [docId] [userId] nodeJS - Fa
iled to subscribe to plugin folder updates. When changing the list of plu
gins, you must restart the server. https://nodejs.org/docs/latest/api/fs.
html#fs_availability
[2025-03-11T06:16:59.694] [WARN] [localhost] [docId] [userId] nodeJS - Ex
press server listening on port 8000 in production-linux mode. Version: 8.
2.0. Build: 1
[2025-03-11T06:17:50.544] [WARN] [localhost] [docId] [userId] nodeJS - Ex
press server starting...
[2025-03-11T06:17:50.546] [WARN] [localhost] [docId] [userId] nodeJS - Se
t services.CoAuthoring.token.enable.browser, services.CoAuthoring.token.e
nable.request.inbox, services.CoAuthoring.token.enable.request.outbox in
the Document Server config to prevent an unauthorized access to your docu
ments and the substitution of important parameters in Document Server req
uests.
[2025-03-11T06:17:50.547] [WARN] [localhost] [docId] [userId] nodeJS - Fa
iled to subscribe to plugin folder updates. When changing the list of plu
gins, you must restart the server. https://nodejs.org/docs/latest/api/fs.
html#fs_availability
[2025-03-11T06:17:50.694] [WARN] [localhost] [docId] [userId] nodeJS - Ex
press server listening on port 8000 in production-linux mode. Version: 8.
2.0. Build: 1

==> /var/log/onlyoffice/documentserver/nginx.error.log <==
2025/03/11 06:31:17 [error] 10008#10008: *5 open() "/var/www/onlyoffice/d
ocumentserver/web-apps/vendor/socketio/socket.io.min.js.map" failed (2: N
o such file or directory), client: 192.168.9.13, server: , request: "GET
/8.2.0-fe3258694c051949279b3285420cfca5/web-apps/vendor/socketio/socket.i
o.min.js.map HTTP/1.1", host: "192.168.100.37:9002"

==> /var/log/onlyoffice/documentserver/converter/out.log <==
[2025-03-11T06:31:22.771] [ERROR] [localhost] [1741224439356100011] [0] n
odeJS - error downloadFile:url=http://192.168.100.37/attachment/onlyoffic
e/广州东部水源工程文件领号登记v9.docx?hotqinsessionid=;attempt=1;code:null;c
onnect:null Error: DNS lookup 192.168.100.37(family:undefined, host:undef
ined) is not allowed. Because, It is private IP address.
    at validateIPAddress (/snapshot/server/Common/node_modules/request-fi
ltering-agent/lib/request-filtering-agent.js:71:20)
```

```
    at Socket.<anonymous> (/snapshot/server/Common/node_modules/request-filtering-agent/lib/request-filtering-agent.js:125:29)
    at Object.onceWrapper (node:events:641:28)
    at Socket.emit (node:events:539:35)
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1173:10)
[2025-03-11T06:31:23.780] [ERROR] [localhost] [1741224439356100011] [0] nodeJS - error downloadFile:url=http://192.168.100.37/attachment/onlyoffice/广州东部水源工程文件领号登记v9.docx?hotqinsessionid=;attempt=2;code:null;connect:null Error: DNS lookup 192.168.100.37(family:undefined, host:undefined) is not allowed. Because, It is private IP address.

[root@localhost engineercms]# docker exec -it 826 /bin/bash
root@826ce03a8378:/# ls
app         bin   dev  home  lib32  libx32  mnt  proc  run   sqlplus.zip  sys  usr
basic.zip   boot  etc  lib   lib64  media   opt  root  sbin  srv          tmp  var
root@826ce03a8378:/# cd etc
root@826ce03a8378:/etc# cd onlyoffice
root@826ce03a8378:/etc/onlyoffice# cd documentserver
root@826ce03a8378:/etc/onlyoffice/documentserver# ls
default.json            development-mac.json      local.json  logrotate
production-linux.json
development-linux.json  development-windows.json  log4js      nginx
production-windows.json
root@826ce03a8378:/etc/onlyoffice/documentserver# exit
exit
[root@localhost engineercms]# docker cp 826:/etc/onlyoffice/documentserver/default.json ./
                                                    Successfully copied 18.9kB to /usr/src/engineercms/./
[root@localhost engineercms]# docker cp ./default.json 826:/etc/onlyoffice/documentserver/default.json
                                                  Successfully copied 18.9kB to 826:/etc/onlyoffice/documentserver/default.json
[root@localhost engineercms]# docker restart 826
826
[root@localhost engineercms]#
```

5 服务端接口调整

从5.3到现在的8.3，OnlyOffice的API接口最大的变化是documentType，变成了word，cell，slide和pdf，其他可以不动，但是要禁止token。否则要在服务端开发好token。前端协作页面，

如果带有历史版本对比功能的，要调整，接口应该也发生了变化，如果不改变，则导致word协作页面无法编辑，excel不受影响。

onlyoffice8.2-token-golang

参考：
https://blog.csdn.net/qq_36437991/article/details/136255926

前端提交给服务端生成token并返回

1 思路：

根据官方文档，生成token的内容（即document，editorConfig这些页面上的json格式代码）要在前端一起打包通过body发送到服务端，服务端收到json数据，golang语言需要解析为结构体，然后将结构体作为“荷载”payload ，生成token，官方的说明：
https://api.onlyoffice.com/docs/docs-api/additional-api/signature/browser/
The payload for the JWT token in the JSON format must have the same structure as the config. Please note that starting from version 7.1, the parameter list to be signed will be strictly regulated. Don’t forget to add all the parameters listed below to your signature.

2 前端：

页面代码中的{{}}是beego框架下golang服务端传递给页面的变量。这些怎么弄，见我的开源代码吧

```
const config = {
    "document": {
      "fileType": "{{.fileType}}",
      "key": "{{.Key}}", //"Khirz6zTPdfd7"
      "title": "{{.Doc.FileName}}",
      "url":"http:\/\/192.168.100.37/attachment/onlyoffice/测试v3.docx"
      // "url": "{{.Engineercmsapi_url}}/attachment/onlyoffice/{{.Doc.FileName}}?hotqinsessionid={{.Sessionid}}",
    },
    "documentType": "{{.documentType}}",
    "editorConfig": {
      "callbackUrl": "{{.Engineercmsapi_url}}/url-to-callback?id={{.Doc.Id}}",
      "customization": {
        "uiTheme": "theme-dark",
        "unit": "cm",
        "wordHeadingsColor": "#00ff00",
        "zoom": 100,
      },
```

```
      "user": {
        "id": {{.Uid }},
        "name": "{{.Username}}"
      },
      "lang": "zh-CN", //"en-US",
      "mode": {{.Mode }}, //"view",//edit
      "region": "zh-CN",
    },
    "height": "100%",
    "type": {{.Type }}, //"desktop",embedded,mobile访问文档的平台类型 网页嵌入
    "width": "100%"
  }
  // window.docEditor = new DocsAPI.DocEditor("placeholder", config);

  $(function () {
    const configJsonStr = JSON.stringify(config);
    $.ajax({
      type: "POST",
      url: "/v1/onlyoffice/jwtencode",
      contentType: "application/json",
      data: JSON.stringify({
        "jsonStr":configJsonStr
      }),
      dataType: "json",
      success: function (data) {
        console.log("成功")
        console.log(data)
        if(data.jwt){
          config.token = data.jwt;
        }
        var docEditor = new DocsAPI.DocEditor("placeholder", config);
      },
      error: function (err) {
        console.error(err);
      }
    })
  })
```

3 服务端：

服务端golang beego框架，配合生成token返回给前端

```
type onlyofficeJwtManager struct {
    key []byte
}

func (j onlyofficeJwtManager) Sign(payload interface {
    Valid() error
}) (string, error) {
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, payload)
    ss, err := token.SignedString(j.key)

    if err != nil {
        return "", errors.New("could not generate a new jwt")
    }

    return ss, nil
}

type OnlyofficeClaims struct {
    Document     OnlyDocument     `json:"document"`
    DocumentType string           `json:"documentType"`
    EditorConfig OnlyEditorConfig `json:"editorConfig"`
    Height       string           `json:"height"`
    Type         string           `json:"type"`
    Width        string           `json:"width"`
    jwt.RegisteredClaims
}

type OnlyDocument struct {
    FileType string `json:"fileType"`
    Key      string `json:"key"`
    Title    string `json:"title"`
    Url      string `json:"url"`
}

type OnlyEditorConfig struct {
    CallbackUrl   string     `json:"callbackUrl"`
    Customization OnlyCustom `json:"customization"`
    User          OnlyUser   `json:"user"`
    Lang          string     `json:"lang"`
    Mode          string     `json:"mode"`
    Region        string     `json:"region"`
}

type OnlyCustom struct {
    UiTheme           string `json:"uiTheme"`
```

```
    Unit              string `json:"unit"`
    WordHeadingsColor string `json:"wordHeadingsColor"`
    Zoom              int    `json:"zoom"`
}

type OnlyUser struct {
    Id   string `json:"id"`
    Name string `json:"name"`
}

// @Title post onlyoffce jwtencode token
// @Description post onlyoffice jwtencode token
// @Success 200 {object} models.Ollyoffice
// @Failure 400 Invalid page supplied
// @Failure 404 office not found
// @router /jwtencode [post]
// 生成token
func (c *OnlyController) JwtEncode() {
    content := c.Ctx.Input.RequestBody
    var onlyofficeconfig OnlyofficeClaims
    err := json.Unmarshal(content, &onlyofficeconfig)
    if err != nil {
        logs.Error(err)
    }
    logs.Info(onlyofficeconfig) // {"document":{"fileType":"doc","key":"1
558683150313208200","title":"xml.doc","url":"http://192.168.100.37/attach
ment/onlyoffice/测试v3.docx"},"documentType":"word","editorConfig":{"callb
ackUrl":"http://192.168.137.1:8081/url-to-callback?id=10","customizatio
n":{"uiTheme":"theme-dark","unit":"cm","wordHeadingsColor":"#00ff00","zoo
m":100},"user":{"id":"0","name":"127.0.0.1"},"lang":"zh-CN","mode":"edi
t","region":"zh-CN"},"height":"100%","type":"desktop","width":"100%"}
    jwt_enabled, err := web.AppConfig.String("JWT_ENABLED")
    if err != nil {
        logs.Error(err)
    }
    if jwt_enabled == "true" {
        jwt_secret, err := web.AppConfig.String("onlyoffice_token_secre
t")
        if err != nil {
            logs.Error(err)
        }
        key := onlyofficeJwtManager{[]byte(jwt_secret)}
        logs.Info(key)
        token, err := key.Sign(onlyofficeconfig)
        if err != nil {
```

```
            logs.Error(err)
        }
        c.Data["json"] = map[string]interface{}{"state": "SUCCESS", "inf
o": "SUCCESS", "jwt": token}
        c.ServeJSON()
        return
    }
    c.Data["json"] = map[string]interface{}{"state": "SUCCESS", "info":
"SUCCESS", "jwt": onlyofficeconfig}
    c.ServeJSON()
}
```

4 callbackUrl

似乎最后一个关闭编辑后，oo服务器会像文档下载服务器callbackUrl传出token，需要将token解析吗？？官方文件：
https://api.onlyoffice.com/docs/docs-api/additional-api/signature/request/token-in-body/

```
Outgoing requests
Request to "callbackUrl" address by document editing service when the las
t user closed the document for editing without changes
Validation is performed for outgoing requests to "callbackUrl" address by
document editing service.

Sample parameters of request to "callbackUrl" address by document editing
service when the last user closed the document for editing without change
s:

{
  "key": "Khirz6zTPdfd7",
  "status": 4
}

Sample of request to "callbackUrl" address by document editing service wh
en the last user closed the document for editing without changes:

POST /url-to-callback.ashx HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJrZXkiOiJLaGlyejZ6VF
BkZmQ3Iiwic3RhdHVzIjo0fQ.gCyNKPpg6ISAnhvFQmRiY6BRqG6WPcEGgnK79hREdkU"
```

```
}

Where the example.com is the name of the server where document manager an
d document storage service are installed. See the How it works section to
find out more on ONLYOFFICE Docs service client-server interactions.

When performing the GET requests from ONLYOFFICE Docs an authorization he
ader with the token is added.
```

```
# 拉取
docker pull knoxzhang/oo-ce-docker-license:8.2.0.1
# 启动
docker run --name=ooffice -i -t -d -p 8080:80 --restart=always knoxzhang/
oo-ce-docker-license:8.2.0.1

链接：https://pan.baidu.com/s/1wfUHVDrM1fablcFV301W3Q 提取码：avvp 复制这段
内容后打开百度网盘手机App，操作更方便哦
```