

统信办公云盘

V7 部署安装手册

【2024 年版】



统信软件技术有限公司
UnionTech Software Technology Co., Ltd.

一 前言

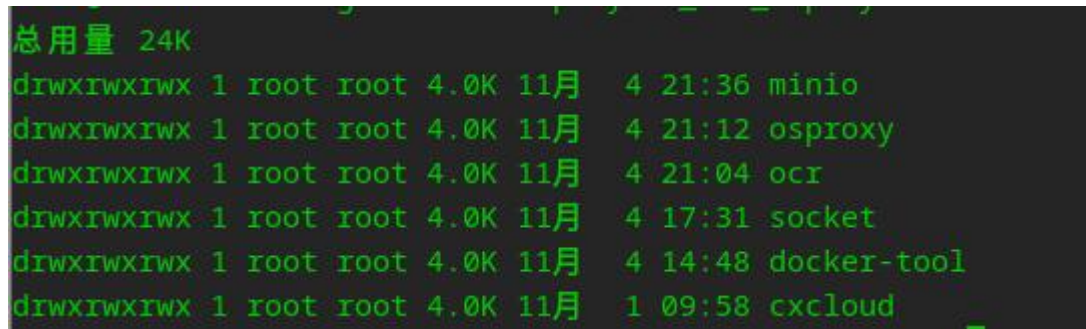
V7 单机部署包 `cx-7.x.x-x.tar.gz` 是一个压缩文件，部署前需进行解压。7.x.x 为版本号，-x 为平台名称，如：`cx-7.1.0-x64.tar.gz`，为 `x86_64` 平台的 7.1.0 版本部署包。

解压部署包

```
tar xvf cx-7.1.0-x64.tar.gz
```

```
cd deploy
```

解压后的目录结构如下：



```
总用量 24K
drwxrwxrwx 1 root root 4.0K 11月 4 21:36 minio
drwxrwxrwx 1 root root 4.0K 11月 4 21:12 osproxy
drwxrwxrwx 1 root root 4.0K 11月 4 21:04 ocr
drwxrwxrwx 1 root root 4.0K 11月 4 17:31 socket
drwxrwxrwx 1 root root 4.0K 11月 4 14:48 docker-tool
drwxrwxrwx 1 root root 4.0K 11月 1 09:58 cxcloud
```

各个子目录如下：

```

root@localhost:/home/mode/changxie/deploy# ll *
-rwxrwxrwx 1 root root 1462887 Nov  5 21:31 畅写文档v7单机部署说明.docx*

cxcloud:
total 8
dr-xr-xr-x 4 root root 4096 Nov  7 10:26 compose/
dr-xr-xr-x 2 root root 4096 Nov  5 21:16 images/

docker-tool:
total 11948
drwxrwxr-x 2 mode mode    4096 Nov  7 11:35 centos-repo/
drwxrwxrwx 2 root root    4096 Nov  4 13:43 docker/
-rwxrwxrwx 1 root root 12218968 Mar 13  2024 docker-compose*
-rwxrwxrwx 1 root root    1114 Nov  4 14:48 docker.service*

minio:
total 405524
-rwxrwxrwx 1 root root      498 Nov  4 16:32 docker-compose.yaml*
-rwxrwxrwx 1 root root 415240704 Oct 30 10:45 minio-x64.tar*

ocr:
total 1703692
-rwxrwxrwx 1 root root      767 Nov  4 15:19 docker-compose.yml*
-rwxrwxrwx 1 root root 1744561152 Oct 29 16:32 ocr-x64.tar*

osproxy:
total 1280152
-rwxrwxrwx 1 root root    2552 Nov  6 14:32 config_default.yaml*
-rwxrwxrwx 1 root root     795 Nov  5 17:40 docker-compose.yaml*
-rwxrwxrwx 1 root root 1310844928 Oct 30 17:45 go-server-x64.tar*
-rwxrwxrwx 1 root root    1625 Nov  6 14:33 nginx.conf*
-rwxrwxrwx 1 root root    5461 Oct 30 14:24 osproxy闕 | 讲發存疑.md*

socket:
total 1101944
-rwxrwxrwx 1 root root     568 Nov  5 16:22 app.conf*
-rwxrwxrwx 1 root root     743 Nov  5 16:32 docker-compose.yaml*
-rwxrwxrwx 1 root root    1582 Nov  6 12:50 nginx.conf*
-rwxrwxrwx 1 root root 1128366592 Oct 30 11:29 socket-server-x64.tar*
-rwxrwxrwx 1 root root    2670 Oct 30 14:24 socket闕 | 讲發存疑.md*
root@localhost:/home/mode/changxie/deploy#

```

进入解压目录开始部署。

二 安装前设置环境

1 关防火墙

centos 系统

```
sudo systemctl disable --now firewalld
```

ubuntu 系统

```
sudo ufw disable
```

2 关闭安全策略

centos 系统

```
setenforce 0
```

```
sed -i 's/^SELINUX=enforcing$/SELINUX=disabled/' /etc/selinux/config
```

ubuntu 系统 无需设置

3 查看当前服务器 IP

```
hostname -I
```

```
root@uos-PC: /home/uos/deploy/cxcloud/compose# hostname -I  
192.168.230.150 172.18.0.1 172.17.0.1 172.21.0.1 172.23.0.1
```

记住机器实际 IP，后续组件部署中改 IP 地址时要用到它。

三 安装镜像管理软件

1 docker-tool 目录

进入目录

```
root@localhost:/home/mode/deploy# cd docker-tool/  
root@localhost:/home/mode/deploy/docker-tool#
```

2 离线安装

#将 docker 文件复制到系统目录

```
sudo cp docker/* /usr/bin/
```

#将 docker 注册为系统服务

```
sudo cp docker.service /etc/systemd/system/docker.service
```

#重新加载配置文件

```
sudo systemctl daemon-reload
```

#启动 Docker 服务

```
systemctl start docker
```

#查看启动状态

```
systemctl status docker
```

```
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/docker-tool# systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/etc/systemd/system/docker.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2024-11-04 14:47:38 CST; 4min 0s ago  
     Docs: https://docs.docker.com  
   Main PID: 8603 (dockerd)  
    Tasks: 38 (limit: 19080)  
   Memory: 23.3M  
   CGroup: /system.slice/docker.service  
           └─8603 /usr/bin/dockerd --selinux-enabled=false -H unix://  
             └─8614 docker-containerd --config /var/run/docker/containerd/containerd.toml
```

#设置 docker 为开机自启

```
systemctl enable docker
```

```
[root@localhost docker-tool]# systemctl enable docker  
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /etc/systemd/system/docker.service.
```

#查看 docker 版本

docker version

```
root@uos-PC: /mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/docker-tool# docker version
Client:
 Version:           18.06.3-ce
 API version:       1.38
 Go version:        go1.10.4
 Git commit:        d7080c1
 Built:             Wed Feb 20 02:24:22 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.3-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       d7080c1
  Built:            Wed Feb 20 02:25:33 2019
  OS/Arch:          linux/amd64
  Experimental:     false
```

#####

#配置 docker-compose

```
sudo cp docker-compose /usr/bin/
```

```
chmod +x /usr/bin/docker-compose
```

#查看版本号

```
root@uos-PC: /mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/docker-tool# docker-compose version
docker-compose version 1.27.4, build 40524192
docker-py version: 4.3.1
CPython version: 3.7.7
OpenSSL version: OpenSSL 1.1.0l 10 Sep 2019
```

如果显示上述提示，说明服务启动完成。

3 在线安装

如果手动离线安装有兼容性问题，且服务器能连接公网则可以进行在线源安装。

A apt 源安装

```
sudo update -y
```

```
sudo apt install docker.io -y          #安装 docker
```

```
sudo apt install docker-compose -y #安装 docker-compose
```

B yum 源安装

若安装源不好使，可以更新成阿里云的安装源，并追加一个 docker 的镜像库，如下图

```
root@localhost:/home/mode/changxie/deploy/docker-tool/centos-repo# ll
total 8
-rw-rw-r-- 1 mode mode 2523 Aug  4 2022 CentOS-Base.repo
-rw-rw-r-- 1 mode mode 1919 Nov  7 11:22 docker-ce.repo
root@localhost:/home/mode/changxie/deploy/docker-tool/centos-repo#
```

```
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak
```

#备份

```
cp CentOS-Base.repo /etc/yum.repos.d/ #替换成阿里云安装源
```

```
cp docker-ce.repo /etc/yum.repos.d/ #添加 docker 安装源
```

```
yum clean all #重建镜像缓存
```

```
yum makecache
```

在线安装 docker

#安装 docker，高版本会自带 compose，无需安装 docker-compose

```
yum install docker-ce docker-ce-cli containerd.io -y
```

#启动 docker 服务，并检查下服务是否启动成功

```
systemctl start docker
```

```
docker ps #能出现红框中的表头，说明启动完成
```

```
root@localhost:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@localhost:~#
```

#查看版本

```
docker -v
```

```
[root@localhost yum.repos.d]# docker -v
Docker version 26.1.4, build 5650f9b
[root@localhost yum.repos.d]#
```

#查看 compose 版本

docker compose version

```
[root@localhost yum.repos.d]# docker compose version
Docker Compose version v2.27.1
[root@localhost yum.repos.d]#
```

至此，centos 环境的 docker 环境安装完成。后述章节中，通过 yml 文件进行部署时，docker-compose 命令要替换成 docker compose 进行部署。

四 OCR 组件

注：ocr 服务对 cpu 有要求，需查看机器 CPU 是否支持 avx2 指令集。

查看方式：

```
lscpu | grep avx2
```

```
[root@localhost ocr]# lscpu | grep avx2
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nonstop_tsc eagerfpu pni pclmulqdq sse3 fma cx16 pcid sse4_1 sse4_2 xzavic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 inopcid rtm mpx rdseed adx smap clflushopt xsaveopt xsavecsg arat
```

如果显示 上图有 Flag 结果则支持。否则不支持。不支持则跳过本步骤安装。

1 进入 ocr 目录

```
root@localhost:/home/mode/deploy# cd ocr/
root@localhost:/home/mode/deploy/ocr# ll
total 1703684
-rw-r--r-- 1 mode mode      878 Oct 29 16:33 docker-compose.yml
-rw----- 1 mode mode 1744561152 Oct 29 16:32 ocr-x64.tar*
root@localhost:/home/mode/deploy/ocr#
```

2 加载 ocr 镜像

docker load < ocr-x.tar #x 为平台名称，如 x64, arm, arm64, SW 等


```

b6f63823f8c99: Loading layer [=====>] 84.18MB/84.18MB
0a49aba3fd52: Loading layer [=====>] 3.395MB/3.395MB
c0f8c3cb14e1: Loading layer [=====>] 30.53MB/30.53MB
195a4622ada2: Loading layer [=====>] 4.608kB/4.608kB
acc5a7b3e8d6: Loading layer [=====>] 12.08MB/12.08MB
c7e1874db306: Loading layer [=====>] 2.048kB/2.048kB
93932edb47e6: Loading layer [=====>] 3.072kB/3.072kB
0fde5d320448: Loading layer [=====>] 233MB/233MB
6ad0195b8716: Loading layer [=====>] 1.31GB/1.31GB
c38bfb7b11ea: Loading layer [=====>] 71.51MB/71.51MB
9cbde3fe2e8e: Loading layer [=====>] 15.87kB/15.87kB
7d666eadadc7: Loading layer [=====>] 51.71kB/51.71kB
942ce12833a1: Loading layer [=====>] 64kB/64kB
f8f6f2f13f9a: Loading layer [=====>] 3.072kB/3.072kB
Loaded image: ocr:m8

```

3 部署组件

查看 yml 文件，根据部署场景修改参数 mem_limit，若端口无冲突则保持端口不动。

vim docker-compose.yml

```

version: "3"

services:
  PaddleOCR:
    #build: .
    container_name: paddle_ocr_api
    image: ocr:m8
    mem_limit: 8000m # 限制容器内存约 8G，该值必需小于系统实际内存
    # 值，否则会因内存吃满造成宕机。无需求则不改
    environment:
      - TZ=Asia/Shanghai
      - OCR_LANGUAGE=ch
      - MAX_WORKERS=1
    #   - PACKAGES_DIR=/usr/local/lib/python3.10/dist-packages
      - PACKAGES_DIR=/usr/local/lib/python3.10/site-packages
    ports:
      - "8000:8000" # 自定义服务暴露端口，8000 为 FastAPI 默认端口，不做修改，
    # 只能改前面的 8000,不要忘了引号。无冲突可不用修改端口
    restart: unless-stopped
    shm_size: '200m'

```

```
volumes:
  - ./data:/root/.paddleocr
#  - ./routers:/app/routers
#  - ./utils:/app/utils
#  - ./models:/app/models
  - ./log:/app/log
```

若有报错:

```
ERROR: The Compose file './docker-compose.yml' is invalid because:
Unsupported config option for services.PaddleOCR: 'mem_limit'
```

请在上述 yml 文件中注释掉 “ mem_limit: 500m”

启动 ocr 服务

```
docker-compose up -d
```

查看状态是否正常

```
docker-compose ps
```

Name	Command	State	Ports
paddle_ocr_api	uvicorn main:app --host 0.0.0.0	Up	0.0.0.0:8000->8000/tcp

部署完成后，ocr 服务端口为：8000 (该端口后续配置中要用到)

4 转换失败提示

注意，OCR 会逐渐消耗内存，可以通过如下命令查看当前 OCR 运行状态:

```
docker stats paddle_ocr_api
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
7bf584a20e2b	paddle_ocr_api	0.15%	654.2MiB / 7.812GiB	8.18%	1.13kB / 0B	159MB / 148kB	12

如果 CPU 或 MEM 接近 100%，客户端在做 OCR 识别时，有可能会提示转换失败，如下图。



此时，可以重启该镜像，如果需要增大内存，可修改 yml 中的参数，再重启。

#重启镜像

```
docker-compose down
```

```
docker-compose up -d
```

五 Socket 组件

1 进入 socket 目录

```
root@localhost:/home/mode/deploy/socket# ll
total 1101944
-rw-r--r-- 1 mode mode      624 Oct 30 10:42 app.conf
-rw-r--r-- 1 mode mode      810 Oct 30 10:42 docker-compose.yml
-rw-r--r-- 1 mode mode     1580 Oct 30 10:42 nginx.conf
-rw-r--r-- 1 mode mode 1128366592 Oct 30 11:29 socket-server.tar
-rw-r--r-- 1 mode mode     2670 Oct 30 14:24 socket関 | 讲辦存霖.md
root@localhost:/home/mode/deploy/socket#
```

2 加载 socker 镜像

docker load < socket-server-x.tar #x 为平台名称, 如 x64, arm, arm64, SW 等

```
1dae5147cd29: Loading layer [=====>] 121.4MB/121.4MB
bcd354c940e1: Loading layer [=====>] 49.61MB/49.61MB
9f843c569746: Loading layer [=====>] 181.5MB/181.5MB
206456f70e8e: Loading layer [=====>] 261.9MB/261.9MB
b1d525823ba4: Loading layer [=====>] 254.6MB/254.6MB
c2563f941150: Loading layer [=====>] 3.584kB/3.584kB
5f70bf18a086: Loading layer [=====>] 1.024kB/1.024kB
1048a5169952: Loading layer [=====>] 2.048kB/2.048kB
67dc67e16d4e: Loading layer [=====>] 158.2kB/158.2kB
f35fc4e3fea4: Loading layer [=====>] 259.2MB/259.2MB
Loaded image: socket-server:latest
```

加载 nginx 镜像

docker load < ../cxcloud/images/nginx_1.24_x64.tar

```
1cbe4b54fa88: Loading layer [=====>] 84.01MB/84.01MB
ba30fdf499c0: Loading layer [=====>] 62.55MB/62.55MB
f38466315ca3: Loading layer [=====>] 3.584kB/3.584kB
8825820b6a3: Loading layer [=====>] 4.608kB/4.608kB
6654ec4b6b7: Loading layer [=====>] 3.584kB/3.584kB
1c7ef6a65288: Loading layer [=====>] 7.168kB/7.168kB
loaded image: nginx:1.24.0
```

3 部署组件

打开 nginx.conf

vim nginx.conf

```
http {
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile    on;

    keepalive_timeout  65;
    # 替换成你的服务地址
    upstream back_server {
        server 192.168.230.150:8080;
    }
    server {
        listen 9999;
        proxy_read_timeout 600s;
        proxy_max_temp_file_size 40960m;
        gzip on;
    }
}
```

根据场景改IP，不动端口

根据部署场景修改 IP 地址，保存后启动服务

```
docker-compose up -d
```

查看状态是否正常

```
docker-compose ps
```

```
root@localhost:/home/mode/changxie/deploy/socket# docker-compose ps
-----
Name                Command              State      Ports
-----
socket-nginx        /docker-entrypoint.sh nginx ...   up        80/tcp, 0.0.0.0:9997->9999/tcp, :::9997->9999/tcp
socket-server       /app/main            up        0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
```

服务对外访问接口： 9997 (该端口后续配置中要用到)

该服务组件的功能是否正常启用，请见第七单的第 6 节说明。

六 minio 组件

1 进行 minio 目录

```
root@localhost:/home/mode/deploy# cd minio/
root@localhost:/home/mode/deploy/minio# ll
total 405520
-rw-r--r-- 1 mode mode      495 Oct 30 10:47 docker-compose.yaml
-rw----- 1 mode mode 415240704 Oct 30 10:45 minio.tar
root@localhost:/home/mode/deploy/minio#
```

2 加载 minio 镜像

docker load < minio-x.tar #x 为平台名称, 如 x64, arm, arm64, SW 等

```
744c86b54390: Loading layer [=====>] 104.1MB/104.1MB
1323fbff4dd: Loading layer [=====>] 20.48kB/20.48kB
9a5123a464dc: Loading layer [=====>] 3.584kB/3.584kB
9e9eecf9e95d: Loading layer [=====>] 3.584kB/3.584kB
6088fcbd6a76: Loading layer [=====>] 1.724MB/1.724MB
678ce496e457: Loading layer [=====>] 36.86kB/36.86kB
50f383b04a07: Loading layer [=====>] 309.3MB/309.3MB
Loaded image: minio/minio:latest
```

3 部署组件

```
docker-compose up -d
```

查看状态

```
docker-compose ps
```

```
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/minio# docker-compose ps
-----
Name                Command                                State      Ports
-----
osproxy_minio      /usr/bin/docker-entrypoint ...        Up         0.0.0.0:9100->9000/tcp, 0.0.0.0:9001->9001/tcp
```

应用访问地址:

IP:9001 系统 Web 交互界面, 可创建存储桶

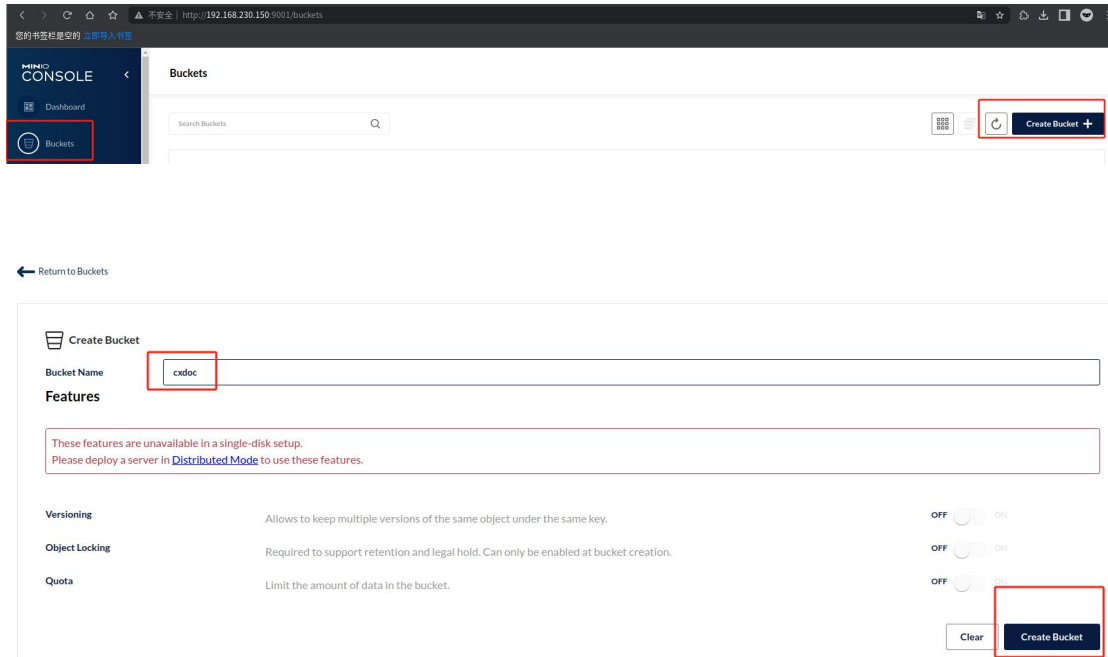
9100 系统对外访问接口 (该端口后续配置中要用到)

4 创建存储桶

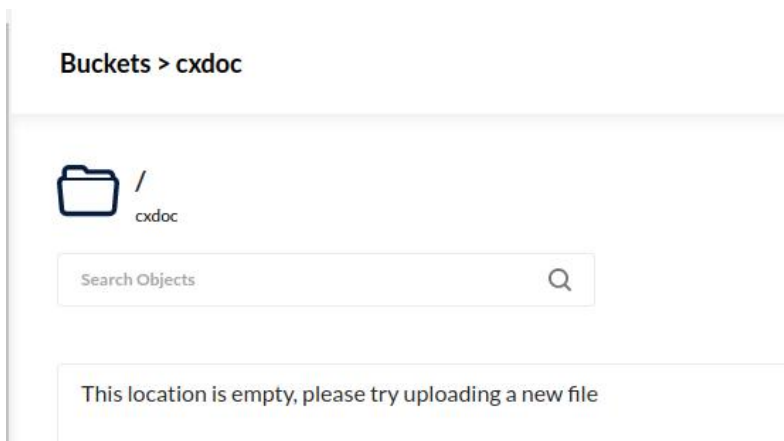
打开浏览器, 输入 IP:9001。用户名 uosdrive@2024 密码: uosdrive@2024 进行登录。

A 创建云盘对象存储桶

如下图, 创建存储桶为 **cxdoc** (该名称后续配置中要用到)

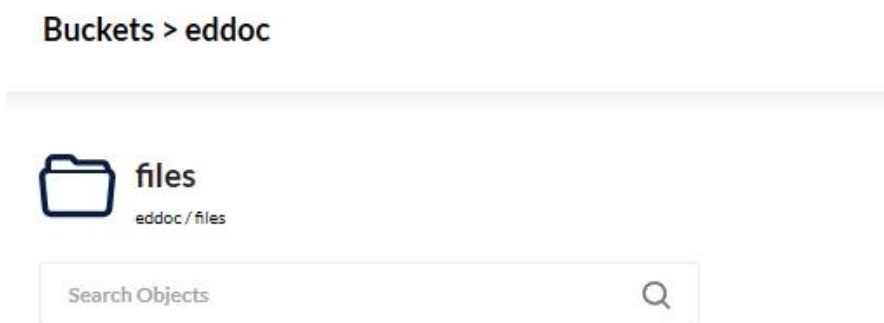


如下图，为创建成功



B 创建编辑器对象存储桶

如下图，创建存储桶为 **eddoc** (该名称后续配置中要用到)



七 统信办公云盘

1 进入 cxcloud 目录

```
root@localhost:/home/mode/deploy# cd cxcloud/
root@localhost:/home/mode/deploy/cxcloud# ll
total 8
drwxr-xr-x 3 root root 4096 Oct 31 21:44 compose/
drwxr-xr-x 2 mode mode 4096 Oct 31 21:36 images/
root@localhost:/home/mode/deploy/cxcloud#
```

2 加载镜像

```
cd images
```

```
for i in `ls *.tar` ; do docker load -i $i ; done
```

```
root@localhost:/home/mode/deploy/cxcloud/images# for i in `ls *.tar` ; do docker load -i $i ; done
Loaded image: registry.cn-beijing.aliyuncs.com/changxie/changxie:6.7.4
Loaded image: registry.cn-beijing.aliyuncs.com/changxie/changxielaborate:cloud-test
Loaded image: registry.cn-beijing.aliyuncs.com/changxie/elasticsearch-changxie:7.6.0
Loaded image: mysql:5.7
Loaded image: nginx:1.24.0
Loaded image: redis:latest
root@localhost:/home/mode/deploy/cxcloud/images#
```

3 部署组件

进入 compose 目录

```
cd ../compose
```

```
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/cxcloud/images# cd ../compose/
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/cxcloud/compose# ls
docker-compose.yml install_cron.sh nginx.conf objectstore
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/cxcloud/compose#
```

运行前置脚本

```
sh install_cron.sh
```

```
setting configuration.....
vm.max_map_count=262144
vm.max_map_count = 262144
Redirecting to /bin/systemctl reload cron.service
Failed to reload cron.service: Unit not found.
Redirecting to /bin/systemctl reload crond.service
setting OK.....
```

打开 yml 文件，根据部署场景修改 IP，若系统端口无冲突则保持端口不动

```
vim docker-compose.yml
```



```

- CXCLOUD_MOBILE_REGISTER_PRIVATE_WEB_URL=
- CXCLOUD_GO_PROXY_PASS=osproxy
- CXCLOUD_SYNFILE_API_URL=http://10.12.17.228/osproxy
- CXCLOUD_SYNFILE_API_LOCAL_URL=http://10.12.17.228/osproxy
- CXCLOUD_SOCKET_API_URL=http://10.12.17.228:9997/
- CXCLOUD_OCR_URL=http://10.12.17.228:8000
- REDIS_HOST=redis
- REDIS_PORT=6379
#- REDIS_PASSWORD=
- MYSQL_DATABASE=jianxiecloud
- MYSQL_USER=jianxiecloud
- MYSQL_PASSWORD=qwe!123
- MYSQL_ROOT_PASSWORD=qwe!123
- MYSQL_HOST=db
changxieoffice-document-server:
  container_name: changxieoffice-document-server
  #build: ../Docker-DocumentServer
  image: registry.cn-beijing.aliyuncs.com/changxie/changxie:6.7.4
  stdin_open: true
  tty: true
  privileged: true
  restart: always
  links:
    - db
  networks:
    - changxieoffice
  expose:
    - '80'
  environment:
    - AUTO_SAVE_ENABLE=true
    - OBJECT_STORAGE_ENABLED=true
    - STORAGE_NAME=storage-minio
    - STORAGE_REGION=local
    - STORAGE_ENDPOINT=10.12.17.228
    - STORAGE_PORT=9100
    - STORAGE_BUCKETNAME=eddoc
    - STORAGE_FOLDERNAME=files
    - STORAGE_URLEXPIRES=60480000
    - STORAGE_ID=uosdrive@2024
    - STORAGE_KEY=uosdrive@2024
    - STORAGE_USERREQUESTTOGETURL=false
    - STORAGE_USESIGNEDURL=true
    - STORAGE_SSEENABLED=false
    - STORAGE_S3FORCEPATHSTYLE=true
    - STORAGE_EXTENALHOST=10.12.17.228:9100
  volumes:

```

根据场景修改IP地址

查看 nginx.conf 文件，根据部署场景修改 IP, 若系统无冲突则端口不动

```

http {
    upstream back_server {
        # server 124.222.198.8:8888;
        server 192.168.230.150:8888;#修改 osproxy 地址
    }
    upstream backend {
        server app-server:9000;
        keepalive 8;#optimizeTest
    }
    upstream backend_esproxy {
        server app-server:9000;
        keepalive 8;#optimizeTest
    }
    #test-modify-time
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

```

```
location ^~/osproxy/ {
    rewrite /osproxy/(.*) /$1 break;
    proxy_pass http://back_server/api/;
    proxy_set_header    X-Forwarded-Proto $scheme;
    proxy_set_header    X-Real-IP      $remote_addr;
    proxy_read_timeout  3600s;
}

location /os-socket-proxy/ {
    rewrite ^/os-socket-proxy/(.*) /$1 break;
    proxy_pass http://192.168.230.150:9997;#修改 socker 地址

    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

location /doc{
    #rewrite /doc/doc/(.*) /$1 break;
    proxy_pass http://changxieoffice-document-server;
    proxy_redirect    off;
}
```

启动服务

```
docker-compose up -d
```

查看状态

```
docker-compose ps
```

```
root@uos-PC:~/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/cxcloud/compose# docker-compose ps
-----
Name                                Command                                State                                Ports
-----
app-server                          /entrypoint.sh php-fpm                Up                                80/tcp, 9000/tcp, 9001/tcp, 9502/tcp
changxieoffice-document-server     /app/ds/run-document-server.sh        Up                                15672/tcp, 443/tcp, 5432/tcp, 5672/tcp, 6379/tcp, 80/tcp, 8000/tcp
elasticsearch_nc                   /usr/local/bin/docker-entr ...        Up                                127.0.0.1:9200->9200/tcp, 9300/tcp
mysql                                docker-entrypoint.sh mysql ...        Up                                3306/tcp, 33060/tcp
nginx-server                        /docker-entrypoint.sh nginx ...       Up                                0.0.0.0:80->80/tcp
redis                               docker-entrypoint.sh redis ...        Up                                6379/tcp
```

查看是否启动完成

```
docker logs -f app-server
```

```
[2024-11-04 19:53:55] [CXINS] - set env configuration finished.....
[2024-11-04 19:53:55] [CXINS] - *****
[2024-11-04 19:53:55] [CXINS] - *                               *
[2024-11-04 19:53:55] [CXINS] - *               installation and configration finished               *
[2024-11-04 19:53:55] [CXINS] - *                               *
[2024-11-04 19:53:55] [CXINS] - *****
[04-Nov-2024 19:53:55] NOTICE: fpm is running, pid 1
[04-Nov-2024 19:53:55] NOTICE: ready to handle connections
```

如图红框中所示，云盘服务已启动完成

4 设置数据存储模式

进入 objectstore 目录，修改如下文件

```
vim s3-minio-m8.json
```

```
1.
2.  {
3.      "system": {
4.          "objectstore": {
5.              "class": "\\OC\\Files\\ObjectStore\\S3",
6.              "arguments": {
7.                  "hostname": "192.168.230.150", #根据实际 IP 填写
8.                  "bucket": "cxdoc", #此处为 minio 中创建的存储桶名
9.                  "region": "default",
10.                 "key": "uosdrive@2024",
11.                 "secret": "uosdrive@2024",
12.                 "port": "9100",
13.                 "use_ssl": false,
14.                 "autocreate": false,
15.                 "use_path_style": true,
16.                 "legacy_path": true
17.             }
18.         }
19.     }
20. }
```

导入参数

```
./objectstore_set.sh s3-minio-m8.json
```

```
root@localhost:/home/mode/changxie/deploy/cxcloud/compose/objectstore# ./objectstore_set.sh s3-minio-m8.json
Config successfully imported from:
```

如图所示，已导入成功。

5 使用系统

打开浏览器，输入部署机器的 IP，出现登录界面，用如下初始管理员登录。

账号: admin 密码: qwe!123

登录后需要授权系统，可联系产品商务获取。



6 验证 Socket 服务是否启动

登录系统后，提示未授权，此时按下 F12，进入浏览器调试页面，进行如图操作，进入 WS 列表。再按下 F5 刷新页面，看是否有如图消息显示。有消息则说明 socket 服务运行良好。



八 OSProxy 组件

1 进入 osproxy 目录

```
root@localhost:/home/mode/deploy# cd osproxy/
root@localhost:/home/mode/deploy/osproxy# ll
total 1280148
-rw-r--r-- 1 mode mode      2556 Oct 30 10:27 config_default.yaml
-rw-r--r-- 1 mode mode      1288 Oct 30 10:27 docker-compose.yaml
-rw-r--r-- 1 mode mode 1310844928 Oct 30 17:45 go-server.tar
-rw-r--r-- 1 mode mode      1624 Oct 30 10:27 nginx.conf
-rw-r--r-- 1 mode mode      5461 Oct 30 14:24 osproxy 讲璇存霖.md
root@localhost:/home/mode/deploy/osproxy#
```

2 加载镜像

docker load < go-server-x.tar #x 为平台名称, 如 x64, arm, arm64, SW 等

```
8e396a1aad50: Loading layer [====>] 129.3MB/129.3MB
9d49e0bc68a4: Loading layer [====>] 11.31MB/11.31MB
dc8e1d8b53e9: Loading layer [====>] 19.31MB/19.31MB
11829b3be9c0: Loading layer [====>] 156.6MB/156.6MB
680d1a3501ba: Loading layer [====>] 230.6MB/230.6MB
7349b2fc8799: Loading layer [====>] 253.5MB/253.5MB
d1f3854396a8: Loading layer [====>] 3.072kB/3.072kB
ff8501cec0bb: Loading layer [====>] 2.048kB/2.048kB
5eae4af34d26: Loading layer [====>] 40.36MB/40.36MB
6a27730eedb: Loading layer [====>] 469.8MB/469.8MB
Loaded image: go-server:latest
```

加载 nginx 镜像

docker load < ../cxcloud/images/nginx_1.24_x64.tar (第五章节已加载则跳过)

```
root@uos-PC:/mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/osproxy# docker load < ../cxcloud/images/nginx_1.24_x64.tar
8cbe4b54fa88: Loading layer [====>] 84.01MB/84.01MB
0a30fdf499c0: Loading layer [====>] 62.55MB/62.55MB
538466315ca3: Loading layer [====>] 3.584kB/3.584kB
68825820b6a3: Loading layer [====>] 4.608kB/4.608kB
56654ec4b6b7: Loading layer [====>] 3.584kB/3.584kB
dc7ef6a65288: Loading layer [====>] 7.168kB/7.168kB
Loaded image: nginx:1.24.0
```

3 部署组件

查看 yml 文件, 根据部署场景修改 IP 地址, 若系统端口无冲突则保持端口不动

vim config_default.yaml

```
app:
  env: prod
  port: 8888 # 服务端口
  app_name: osproxy # 服务名称
  app_url: http://192.168.230.150
  os_prefix: "urn:oid:"
  bucket: "cxdoc"

log:
  level: info # 日志等级
  root_dir: ./storage/logs # 日志根目录
  filename: app.log # 日志文件名称
  format: json # 写入格式 可选json
  show_line: true # 是否显示调用行
  max_backups: 3 # 旧文件的最大个数
  max_size: 500 # 日志文件最大大小 (MB)
  max_age: 28 # 旧文件的最大保留天数
  compress: true # 是否压缩
  enable_file: true # 是否启用日志文件

database:
  - db_name: default
    driver: mysql # 数据库驱动
    host: 192.168.230.150 # 服务地址
    port: 3306 # 端口
    database: jianxiecloud # 数据库名称
    username: jianxiecloud # 用户名
    password: qwel123 # 密码
    charset: utf8mb4 # 编码格式
    max_idle_conns: 10 # 空闲连接池中连接的最大数量
    max_open_conns: 100 # 打开数据库连接的最大数量
    log_mode: info # 日志级别
    enable_lg_log: false # 是否开启自定义日志
    enable_file_log_writer: false # 是否打印到日志文件
    log_filename: sql.log # 日志文件名称

redis:
  host: 192.168.230.150 # 服务地址
  port: 6379 # 服务端口
  db: 0 # 库选择
  #password: # 密码

minio:
  endpoint: 192.168.230.150:9100 # 服务地址
  access_key_id: uosdrive@2024 # 用户名
  secret_access_key: uosdrive@2024 # 密码
  use_ssl: false # 密码
  enabled: true # 是否启用

cos:
  appid: 12***63195 # 唯一标识
```

查看 nginx.conf 文件，根据部署场景修改 IP 地址，若系统端口无冲突则保持端口不动

```
vim nginx.conf
```

```
http {
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile    on;

    keepalive_timeout 65;
    # 替换成你的服务地址
    upstream back_server {
        server 192.168.230.150:8888;
    }
    server {
        listen 80;
        proxy_read_timeout 600s;
    }
}
```

启动 osproxy 服务

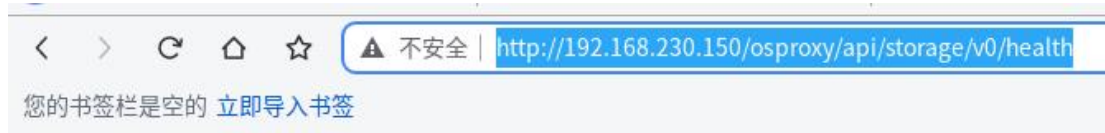
```
docker-compose up -d
```

注意查看状态是否正常

```
docker-compose ps
```

```
root@uos-PC: /mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/osproxy# docker-compose ps
Name                Command                  State      Ports
-----
osproxy_nginx       /docker-entrypoint.sh nginx ...   Up        0.0.0.0:9999->80/tcp
storage             /app/main                Up        0.0.0.0:8888->8888/tcp
root@uos-PC: /mnt/hgfs/nc/cx-deploy/cx_x64_deploy/m8/osproxy#
```

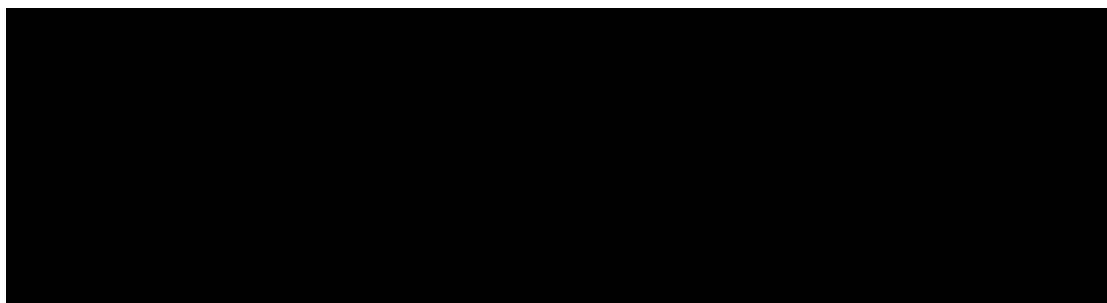
测试地址: <http://IP/osproxy/api/storage/v0/health>, 如下图则已启动成功



```
{\"code\":0,\"msg\":\"ok\",\"data\":\"Health!\"}
```

九 硬件配置清单

【附】 办公云盘硬件单机配置清单



办公云盘及依赖组件服务器配置要求(100 用户)

服务器配置	架构 (CPU)	网卡	系统盘	数据盘	数量	应用	操作系统	应用提供方	用户数量/并发
内存: 128G CPU:32 核	Intel 或 AMD	万兆网卡 (10000M)	200G SSD 盘	5T SSD (5 年存储 周期)	1	畅写文档(Cloud、 OSProxy 及 Editor)	CentOS7.4 以 上、统信服务器 操作系统 V20	厂商	100 用户(15 并发 量) 用户数量与并发量 按照 15%进行折 算
						Elasticsearch 7.6.0		客户自行准 备	
						MYSQL 5.7		客户自行准 备	
						REDIS 5.0 以上		客户自行准 备	
						RabbitMQ 3.7 以上		客户自行准 备	
内存: 92G CPU: 12 核 显卡: 24G		万兆网卡 (10000M)	200G SSD 盘	500G SSD	1	OCR 服务(选配)	厂商		

办公云盘及依赖组件服务器配置要求(200 用户)									
服务器配置	架构 (CPU)	网卡	系统盘	数据盘	数量	应用	操作系统	应用提供方	用户数量/并发
内存: 92G CPU:32 核	Intel 或 AMD	万兆网卡 (10000M)	200G SSD	6T SSD (5 年存储 周期)	1	畅写文档(Cloud、 OSProxy 及 Editor)	CentOS7.4 以 上、统信服务器 操作系统 V20	厂商	200 用户(30 并发 量) 用户数量与并发量 按照 15%进行折 算
内存: 32G CPU:16 核			200G SSD	2T SSD	1	Elasticsearch 7.6.0		客户自行准 备	
内存: 32G CPU:16 核			200G SSD	500G SSD	1	MYSQL 5.7 REDIS 5.0 以上 RabbitMQ 3.7 以上			
内存: 92G CPU: 12 核 显卡: 24G			200G SSD	500G SSD	1	OCR 服务		厂商	

办公云盘及依赖组件服务器配置要求(1000 用户)

服务器配置	架构(CPU)	网卡	系统盘	数据盘	数量	应用	操作系统	应用提供方	用户数量/并发
内存: 64G CPU:32 核	Intel 或 AMD	万兆网卡 (10000M)	200G SSD	20T SSD	1	畅写文档(Cloud、 OSProxy)	CentOS7.4 以 上、统信服务器 操作系统 V20	厂商	1000 用户(150 并发 量) 用户数量与并发量按 照 15%进行折算
内存: 128G CPU:64 核			200G SSD	4T SSD	1	畅写在线(Editor)		厂商	
内存: 32G CPU:16 核			200G SSD	2T SSD	1	Elasticsearch 7.6.0		客户自行准 备	
内存: 32G CPU:16 核			200G SSD	500G SSD	1	MYSQL 5.7			
						REDIS 5.0 以上			
内存: 92G CPU: 12 核 显卡: 24G	200G SSD	500G SSD	1	OCR 服务	厂商				

办公云盘及依赖组件服务器配置要求(2000 用户)

服务器配置	架构(CPU)	网卡	系统盘	数据盘	数量	应用	操作系统	应用提供方	用户数量/并发
内存: 128G CPU:64 核	Intel 或 AMD	万兆网卡 (10000M)	200G SSD	20T SSD	1	畅写文档(Cloud、 OSProxy)	CentOS7.4 以 上、统信服务器 操作系统 V20	厂商	2000 用户(300 并发 量) 用户数量与并发量按 照 15%进行折算
内存: 128G CPU:64 核			200G SSD	4T SSD	1	畅写在线(Editor)		厂商	
内存: 32G CPU:16 核			200G SSD	2T SSD	1	Elasticsearch 7.6.0		客户自行准 备	
内存: 64G CPU:16 核			200G SSD	500G SSD	1	MYSQL 5.7			
						REDIS 5.0 以上			
内存: 92G CPU: 12 核 显卡: 24G	200G SSD	500G SSD	1	OCR 服务	厂商				