算法开发服务器（镜像/容器）管理

一、当前服务器资源列表

外网ip	端口	内网ip	用户名	密码
58.250.166.10	22223	192.168.6.113	root	***
58.250.166.10	22227	192.168.6.117	root	***
58.250.166.10	22228	192.168.6.118	root	***
58.250.166.10	22224	192.168.6.121	developer	***

二、使用基础镜像，启动容器【服务器开发版】示例：

1、镜像使用说明：

1-1、目的：

为了使算法开发与运维工作相对独立，提高算法人员在环境运维的效率。

Dockerfile 文件【已入库】：

~~1.Dockerfile-cuda11.2_u18.06_torch1.9【弃用】~~

2.Dockerfile-cuda11.7_u18.06_torch1.13

基础镜像：在各个服务器上都会有一个基础镜像：docker images

```
1 ~~nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04~~
2 ~~nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v2~~
3 nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3
```

1-2、服务器算法基础环境：

基础环境：**cv_env**

建议项目的不同开发人员可以克隆该环境进行定制。

1-3、项目镜像【算法不参与管理】：

各项目在实施过程中的镜像，在base基础镜像上扩展而来。

项目镜像由运维管理，算法开发人员迭代算法和模型过程中有需要新增的依赖包同步给系统和运维。安装过程中出问题再协助解决。

2、命令行启动容器示例【--name 容器名称 -p 端口映射】：

加载镜像【镜像包会提供在服务器对应目录】：

```
docker load -i Ind_Vision_Base.tar
docker load -i Ind_Vision_Base_v2.tar
docker load -i Ind_Vision_Base_v3.tar
# docker images
nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04
nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v2
nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3
```

创建容器：

```
nvidia-docker run -dit --name cv_algo -p 8322:22 -p 8330-8399:8330-8399 -v /etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v /var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v /data/algorithm/cv_algo/dataset/public:/root/dataset/public -v /data/algorithm/cv_algo/dataset/cv_algo:/root/dataset/cv_algo -v /data/algorithm/cv_algo/project/cv_algo:/root/project/cv_algo -v /data/algorithm/cv_algo/shared:/root/shared -v /data/algorithm/cv_algo/common/pretrained/_.torch:/root/.torch -v /data/algorithm/cv_algo/common/pretrained/_.cache:/root/.cache -v /dev/shm:/dev/shm --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04

nvidia-docker run -dit --name cv_algo -p 8322:22 -p 8330-8399:8330-8399 -v /etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v /var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v /data/algorithm/cv_algo/dataset/public:/root/dataset/public -v /data/algorithm/cv_algo/dataset/cv_algo:/root/dataset/cv_algo -v /data/algorithm/cv_algo/project/cv_algo:/root/project/cv_algo -v /data/algorithm/cv_algo/shared:/root/shared -v /data/algorithm/cv_algo/common/pretrained/_.torch:/root/.torch -v /data/algorithm/cv_algo/common/pretrained/_.cache:/root/.cache -v /dev/shm:/dev/shm --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v2

# 注意挂载路径；目录命名规则：
nvidia-docker run -dit --name test_algo -p 7322:22 -p 7330-7399:7330-7399 -v /etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v
```

```
/var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v
/data/algorithm/gtcv_algo/dataset/public:/root/dataset/public -v
/data/algorithm/gtcv_algo/dataset/test_algo:/root/dataset/test_algo -v
/data/algorithm/gtcv_algo/project/test_algo:/root/project/test_algo -v
/data/algorithm/gtcv_algo/shared:/root/shared -v
/data/algorithm/gtcv_algo/common/pretrained/_.torch:/root/.torch -v
/data/algorithm/gtcv_algo/common/pretrained/_.cache:/root/.cache -v
/dev/shm:/dev/shm --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3

nvidia-docker run -dit --name test_algo -p 7322:22 -p 7330-7399:7330-7399 -v
/etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v
/var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v
/data03/algorithm/gtcv_algo/dataset/public:/root/dataset/public -v
/data03/algorithm/gtcv_algo/dataset/test_algo:/root/dataset/test_algo -v
/data03/algorithm/gtcv_algo/project/test_algo:/root/project/test_algo -v
/data03/algorithm/gtcv_algo/shared:/root/shared -v
/data03/algorithm/gtcv_algo/common/pretrained/_.torch:/root/.torch -v
/data03/algorithm/gtcv_algo/common/pretrained/_.cache:/root/.cache -v
/dev/shm:/dev/shm --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3

# or 主目录 /data/algorithm/
docker run -dit --name test_algo -p 7322:22 -p 7330-7399:7330-7399 -v
/etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v
/var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v
/data/algorithm/dataset/public:/root/dataset/public -v
/data/algorithm/dataset/test_algo:/root/dataset/test_algo -v
/data/algorithm/project/test_algo:/root/project/test_algo -v
/data/algorithm/shared:/root/shared -v
/data/algorithm/common/pretrained/_.torch:/root/.torch -v
/data/algorithm/common/pretrained/_.cache:/root/.cache -v /dev/shm:/dev/shm --
gpus all --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3

主目录 /data03/algorithm/
docker run -dit --name test_algo -p 7322:22 -p 7330-7399:7330-7399 -v
/etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v
/var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v
/data03/algorithm/dataset/public:/root/dataset/public -v
/data03/algorithm/dataset/test_algo:/root/dataset/test_algo -v
/data03/algorithm/project/test_algo:/root/project/test_algo -v
/data03/algorithm/shared:/root/shared -v
/data03/algorithm/common/pretrained/_.torch:/root/.torch -v
/data03/algorithm/common/pretrained/_.cache:/root/.cache -v /dev/shm:/dev/shm -
-gpus all --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v3
```

目录管理：

1、每台服务器磁盘，搭建为共享文件系统；挂载为统一主目录：比如 /mnt/data; 或者 /data/ ; 算法主目录设置：可直接 /mnt/data；或者/data;【因当前服务器目录比较乱，示例的统一目录为：/data/algorithm/】

2、主目录下分为四个分目录：

-dataset: 存放项目数据集，子目录按照工厂、项目进行分级管理

```
-dataset
|--test_algo # 项目1, 每个项目对应一个容器
|--smt # 项目2: smt项目, 下面子目录可按照俊宇smt的数据集管理分级管理。
|   |-- zhaoxing
|   |-- fushikang
|   |-- xinghua
|   |__ gaoshengda
|__northglass # 项目3: 北玻项目

```

-project: 存放项目代码

```
-project
|--test_algo # 项目1, 每个项目对应一个容器
|--smt # 项目2: smt项目
|   |-- 产线1
|   |-- 产线2
|   |-- 产线3
|   |__ 产线4
|__northglass # 项目3: 北玻项目

```

-shared: 共享交换目录， 非重要数据，如开源数据集（空间不足时可删除）

-common: 算法通用路径， pretrained, 预训练模型，目前有 .cache, .torch 等，将在 docker启动时将挂载在docker 内。

3、其他相关命令

```
# docker 修改后保存为新的镜像： docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
```

```
示例：docker commit 46a4561f6d49 nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v2
# 导出镜像包
示例：docker save -o *.tar nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-v2
# 其他
docker ps：列出正在运行的容器列表。
docker start <容器ID或容器名>：启动已停止的容器。
docker stop <容器ID或容器名>：停止运行中的容器。
docker restart <容器ID或容器名>：重启容器。
docker rm <容器ID或容器名>：删除容器。
docker logs <容器ID或容器名>：查看容器的日志输出。
docker exec -it <容器ID或容器名> <命令>：在运行中的容器中执行命令
```

4、容器使用说明

4-1、docker容器：

每个容器 SSH默认用户密码：root / getech

容器基础环境：cv_env

建议同一个项目的不同开发人员，可克隆该环境进行定制，如 cv_env_jy, cv_env_yj

```
ssh -p xx22 root@ip
映射端口：xx22
SSH 默认用户密码：root/getech
jupyter可用端口范围：xx30-xx99
代码：/root/project/aaa/bbb/ccc（个人代码，要求放在此路径下）
数据：/root/dataset/aaa/bbb/ccc（数据，要求放在此路径下）
```

4-2、ssh进入docker内

```
# 如果未启动ssh server
docker exec -it c_name bash
service ssh start
```

4-3、工业算法虚拟环境：

```
conda deactivate && conda activate cv_env
（可以修改~/.bashrc实现）
```

4-4、jupyter使用：

```
# 配置jupyter密码：
jupyter notebook password
# 在~/目录下，启动jupyter服务：
nohup jupyter-notebook --no-browser --ip 0.0.0.0 --port 7340--allow-root > jupyter.nohub.out &
```

三、使用基础镜像、启动容器【项目部署版】示例：

1、命令行启动容器示例【--name 容器名称 -p 端口映射】：

加载镜像：

```
docker load -i  bp_algo.tar
```

创建容器：

```
nvidia-docker run -dit --name cv_algo_v1 -p 8322:22 -p 2502:8502 -v /etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v /var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v /data/algorithm/cv_algo/dataset/public:/root/dataset/public -v /data/algorithm/cv_algo/dataset/cv_algo:/root/dataset/cv_algo -v /data/algorithm/cv_algo/project/cv_algo:/root/project/cv_algo -v /data/algorithm/cv_algo/shared:/root/shared -v /data/algorithm/cv_algo/common/pretrained/_.torch:/root/.torch -v /data/algorithm/cv_algo/common/pretrained/_.cache:/root/.cache -v /dev/shm:/dev/shm --privileged nvidia/cuda:11.2.2-cudnn8-devel-ubuntu18.04-v1
```

2、docker-compose方式启动容器：

./docker-compose.yml文件, 常用命令如下：

```
version: '3.3'
services:
```

```
  algorithma_sub1:
    image: nvidia/cuda:11.2.2-cudnn8-devel-ubuntu18.04-v1
    restart: always
    container_name: cv_algo_v1
    ports:
      - "2502:8502"
      - "2322:22"
    command: ["bash",
"/root/project/cv_algo/bpglass/bp_glass/bp_glass_algo/algo_start.sh"]
    deploy:
      resources:
        reservations:
          devices:
          - driver: nvidia
            capabilities: [gpu]
    volumes:
      - /etc/localtime:/etc/localtime
      - /var/wdcvlm:/var/wdcvlm
      - /var/tscvlm:/var/tscvlm
      - /etc/machine-id:/etc/machine-id
      - /data/algorithm/cv_algo/dataset/public:/root/dataset/public
      - /data/algorithm/cv_algo/dataset/cv_algo:/root/dataset/cv_algo
      - /data/algorithm/cv_algo/project/cv_algo:/root/project/cv_algo
      - /data/algorithm/cv_algo/shared:/root/shared
      - /data/algorithm/cv_algo/common/pretrained/_.torch:/root/.torch
      - /data/algorithm/cv_algo/common/pretrained/_.cache:/root/.cache
      - /dev/shm:/dev/shm
    networks:
      ox_smt:
        ipv4_address: 172.32.0.14
networks:
  ox_smt:
    ipam:
      config:
        - subnet: 172.32.0.0/24

```

示例：当前北玻项目（bp_algo）：

```
version: '3.3'
services:
  algorithma_sub1:
    image: nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-20230724
    restart: always
    container_name: bp_algo
```

```
    ports:
      - "2502:8502"
      - "2322:22"
    command: ["bash",
"/root/project/bp_algo/bpglass/project/northglass/bp_glass_algo_s1/algo_start.s
h"]
    deploy:
      resources:
        reservations:
          devices:
          - driver: nvidia
            capabilities: [gpu]
    volumes:
      - /etc/localtime:/etc/localtime
      - /var/wdcvlm:/var/wdcvlm
      - /var/tscvlm:/var/tscvlm
      - /etc/machine-id:/etc/machine-id
      - /usr/data:/usr/data
      - /usr/algorithm:/usr/algorithm
      - /usr/algorithm/bp_algo/dataset/public:/root/dataset/public
      - /usr/algorithm/bp_algo/dataset/bp_algo:/root/dataset/bp_algo
      - /usr/algorithm/bp_algo/project/bp_algo:/root/project/bp_algo
      - /usr/algorithm/bp_algo/shared:/root/shared
      - /usr/algorithm/bp_algo/common/pretrained/_.torch:/root/.torch
      - /usr/algorithm/bp_algo/common/pretrained/_.cache:/root/.cache
      - /dev/shm:/dev/shm
    networks:
      ox_smt:
        ipv4_address: 172.32.0.14
networks:
  ox_smt:
    ipam:
      config:
        - subnet: 172.32.0.0/24

```

algo_start.sh：

```
/root/conda/envs/cv_env/bin/python
/root/project/bp_algo/bpglass/project/northglass/bp_glass_algo_s2/algo_service.
py

```

```
# 构建和启动容器 这个命令会根据 docker-compose.yml 文件中的配置构建并启动所有的服务容器。
docker-compose up
# 启动容器（后台模式）
docker-compose up -d
# 停止容器：
docker-compose down
# 查看容器状态：
docker-compose ps
# 查看日志
docker-compose logs
```

3、其他相关命令

```
# docker 修改后保存为新的镜像： docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]
docker commit 46a4561f6d49 nvidia/cuda:11.2.2-cudnn8-devel-ubuntu18.04-v1
# 导出镜像包
docker save -o *.tar nvidia/cuda:11.2.2-cudnn8-devel-ubuntu18.04-v1
# 其他
docker ps: 列出正在运行的容器列表。
docker start <容器ID或容器名>: 启动已停止的容器。
docker stop <容器ID或容器名>: 停止运行中的容器。
docker restart <容器ID或容器名>: 重启容器。
docker rm <容器ID或容器名>: 删除容器。
docker logs <容器ID或容器名>: 查看容器的日志输出。
docker exec -it <容器ID或容器名> <命令>: 在运行中的容器中执行命令
```

4、example （北坡项目示例）

工控机上每次算法有依赖更新（环境/算法包）：

1、提供对应的whl包： 算法人员提供 算法和需要新增的环境whl包和算法包，发布到工控机。由算法人员远程安装/无法远程（无网）提供文档由现场人员安装。

~~镜像保存: 项目稳定，最后交付的时候，由运维人员commit当前的容器保存新的tag镜像。~~

2、镜像保存：当前系统与算法的交互方式，依赖包更新只能由算法人员更新，为了保证运维环境管理的一致性和环境的稳定性，基础镜像统一使用 Ind_Vision_Base.tar，该镜像包含基本所有的工业算法的环境。以前的项目不需要更新，依旧按照以前项目方案。即算法人员不需要发镜像到现场，Ind_Vision_Base.tar运维会加载在部署工控机。

目的：

1、Ind_Vision_Base.tar 算法研发进行大版本迭代，交付运维管理。

2、算法开发团队保持开发环境的一致性，方便环境维护和项目对接。

3、算法开发人员在部署时以及每次有新的算法包和依赖包更新时，不需要进行类似镜像包的大文件的拷贝传输。

4-1、有依赖包更新, 更新流程：

如需要更新安装gtcv算法库的whl包（gtcv-0.0.1-py3-none-any.whl ），将依赖包放在docker映射目录下，手动 doker run启动容器：

```
docker run -dit --name bp_algo -p 2502:8502 -p 2322:22 \
-v /etc/localtime:/etc/localtime -v /var/wdcvlm:/var/wdcvlm -v
/var/tscvlm:/var/tscvlm -v /etc/machine-id:/etc/machine-id -v
/usr/data:/usr/data -v
/usr/algorithm:/usr/algorithm -v
/usr/algorithm/bp_algo/dataset/public:/root/dataset/public -v
/usr/algorithm/bp_algo/dataset/bp_algo:/root/dataset/bp_algo -v
/usr/algorithm/bp_algo/project/bp_algo:/root/project/bp_algo -v
/usr/algorithm/bp_algo/shared:/root/shared -v
/usr/algorithm/bp_algo/common/pretrained/_.torch:/root/.torch -v
/usr/algorithm/bp_algo/common/pretrained/_.cache:/root/.cache -v
/dev/shm:/dev/shm \
 --gpus all \
 --privileged nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04
```

进入容器安装对应依赖包：

```
docker exec -it bp_algo bash
pip install *.whl
```

Commit 保存为新的镜像（添加时间节点注释）：

```
docker commit f2050e481331 nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-20230724
```

修改docker-compose文件的images, docker-compose管理容器和启动服务：

```
version: '3.3'
services:
  algorithma_sub1:
```

```
    image: nvidia/cuda:11.7-cudnn8-devel-ubuntu18.04-20230724
    restart: always
    container_name: bp_algo
    ports:
      - "2502:8502"
      - "2322:22"
    command: ["bash",
"/root/project/bp_algo/bpglass/project/northglass/bp_glass_algo/algo_start.sh"]
    deploy:
      resources:
        reservations:
          devices:
          - driver: nvidia
            capabilities: [gpu]
    volumes:
      - /etc/localtime:/etc/localtime
      - /var/wdcvlm:/var/wdcvlm
      - /var/tscvlm:/var/tscvlm
      - /etc/machine-id:/etc/machine-id
      - /usr/data:/usr/data
      - /usr/algorithm:/usr/algorithm
      - /usr/algorithm/bp_algo/dataset/public:/root/dataset/public
      - /usr/algorithm/bp_algo/dataset/bp_algo:/root/dataset/bp_algo
      - /usr/algorithm/bp_algo/project/bp_algo:/root/project/bp_algo
      - /usr/algorithm/bp_algo/shared:/root/shared
      - /usr/algorithm/bp_algo/common/pretrained/_.torch:/root/.torch
      - /usr/algorithm/bp_algo/common/pretrained/_.cache:/root/.cache
      - /dev/shm:/dev/shm
    networks:
      ox_smt:
        ipv4_address: 172.32.0.14
networks:
  ox_smt:
    ipam:
      config:
        - subnet: 172.32.0.0/24

```