1. 硬件要求：CPU 8核以上， 硬盘1T 以上， GPU内存32G，最好配置2个GPU以上。
2. 宿主机安装Linux Ubuntu 16版本操作系统

查看ubuntu操作系统版本：

lsb\_release -a

uname -a

1. 宿主机安装GPU驱动，确保 nvidia-smi 命令可用。
2. 在宿主机安装docker
3. sudo apt-get update
4. sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common
5. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. sudo apt install docker-ce
2. 安装nvidia-docker

# If you have nvidia-docker 1.0 installed: we need to remove it and all existing GPU containers

docker volume ls -q -f driver=nvidia-docker | xargs -r -I{} -n1 docker ps -q -a -f volume={} | xargs -r docker rm -f

sudo apt-get purge -y nvidia-docker

# Add the package repositories

curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \

sudo apt-key add -

distribution=$(. /etc/os-release;echo $ID$VERSION\_ID)

curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | \

sudo tee /etc/apt/sources.list.d/nvidia-docker.list

sudo apt-get update

# Install nvidia-docker2 and reload the Docker daemon configuration

sudo apt-get install -y nvidia-docker2

sudo pkill -SIGHUP dockerd

# Test nvidia-smi with the latest official CUDA image

docker run --runtime=nvidia --rm nvidia/cuda:10.0-base nvidia-smi

1. 在宿主机上安装mysql数据库

sudo apt-get install mysql-server

sudo apt install mysql-client

sudo apt install libmysqlclient-dev

找到默认用户及密码：

sudo cat /etc/mysql/debian.cnf

登录：

mysql -u debian-sys-maint -p

更改root密码及访问策略：

use mysql;

update user set authentication\_string=password("pcl123456") where user="root";

grant all privileges on \*.\* to 'root'@'%' identified by 'pcl123456' with grant option;

update user set plugin="mysql\_native\_password";

flush privileges;

设置mysql允许远程访问：

sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

注释掉 bind-address = 127.0.0.1

然后保存。

重新启动mysql

sudo service mysql restart

连接上mysql数据库，创建labelsystem库（CREATE SCHEMA `test\_schema` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;），一定要使用utf8mb4字符集，并执行db目录下的labelsystem\_2.sql脚本。

1. 恢复docker并在docker中运行:
2. 从文件中加载docker，docker镜像目前比较大，可以联系 [zouap@pcl.ac.cn索取。](mailto:zouap@pcl.ac.cn索取。)也可以直接从百度网盘下载：

（

百度网盘链接：https://pan.baidu.com/s/1Pvy7TKNb4UQsGnSUJhb0-Q

提取码：1234

共18个文件，下载之后，需要将这些文件合并成一个，才是最终的镜像。合并的Java代码，见本文档目录下的java类）

sudo docker load --input label\_base\_v0604.tar

(2)从镜像中启动一个容器，注意 8000端口要映射：

sudo docker run --runtime=nvidia --name mm\_prj1 -p 8000:8000 --shm-size 4G -i -t -v /data/label:/data 5d9daad6749d /bin/bash

1. 进入docker后，进入minio目录，运行minio，其中/data/minio/data/表示minio数据存储地址，一般映射在宿主机某个目录，见docker run中的-v参数.执行以下命令：

export MINIO\_ACCESS\_KEY=minio

export MINIO\_SECRET\_KEY=miniominio

nohup ./minio server /data/minio/data/ &

执行完成后，查看minio输出结果，如下所示：

Attempting encryption of all config, IAM users and policies on MinIO backend

Endpoint: http://172.17.0.2:9000 http://127.0.0.1:9000

Browser Access:

http://172.17.0.2:9000 http://127.0.0.1:9000

标红的部分，需要配置在下面第（4）步中application-runtime.properties中minio.url属性中。

1. 进入javaapp目录：

拷贝代码jar目录下的两个文件

labelSystem-0.0.1-SNAPSHOT.jar

application-runtime.properties

到javaapp目录中。

修改application-runtime.properties中下面标红的参数为自己设置的参数：

spring.datasource.url=jdbc:mysql://192.168.62.129:3306/labelsystem?serverTimezone=CTT&useUnicode=true&characterEncoding=utf8&useSSL=false

spring.datasource.username=root

spring.datasource.password=pcl123456

这个是minio配置变量，使用第(3)步的minio运行结果进行替换。

minio.url= <http://192.168.62.129:9000/>

这个配置文件中这4行用来决定服务对外提供https访问，如果注释掉这4行，则是http访问。

server.ssl.key-store=classpath:keystore.p12

server.ssl.key-store-password=Pcl123456

server.ssl.keyStoreType=PKCS12

server.ssl.keyAlias=pcl

然后运行：

nohup java -jar labelSystem-0.0.1-SNAPSHOT.jar &

1. 在宿主机上安装nginx反向代理

sudo apt-get install nginx

配置nginx:

cd /etc/nginx/

sudo vim nginx.conf

在此文件中的http节点下添加：

server {

listen 80;

server\_name ip;

client\_max\_body\_size 10000m;

client\_body\_buffer\_size 8m;

location ^~ /minio/ {

proxy\_pass <https://ip:8000/minio/;>

proxy\_set\_header Host $host;

}

location ^~ /dcm/ {

proxy\_pass <https://ip:8000/dcm/;>

proxy\_set\_header Host $host;

}

location ^~ /api/ {

proxy\_pass <https://ip:8000/api/;>

proxy\_set\_header Host $host;

}

location / {

add\_header Access-Control-Allow-Origin \*;

add\_header Access-Control-Allow-Methods 'GET, POST, OPTIONS, PATCH';

add\_header Access-Control-Allow-Headers 'DNT,X-Mx-ReqToken,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Authorization';

root /home/label/javaapp/web/labeling/web-dl;

index index.html;

}

}

修改其中的ip及root节点中标红的字符串为正确的本地环境数据。

7086833b9ae0de63fc2e238864c4c9d

其中，在第6步java 正常启动之后，查看labelsystem.log，检查tomcat的启动方式是http还是https，并查看端口。 然后根据显示的信息修改proxy\_pass中的内容。

8、在宿主机上安装前端页面

（1）将web目录下文件拷贝到宿主机的某个目录下（如：/home/pcl/web），然后在nginx.conf配置文件中增加服务器配置，可以参见nginx搭建一节。

（2）查看宿主机的IP地址，修改 web\labeling\web-dl\js\func.js 文件中getIp方法返回的IP地址，与nginx中配置的ip地址一致。

（3）重新启动nginx服务。

（4）使用浏览器，用 http://宿主机的IP地址 访问，并进行测试。用户名及密码：

LabelSystem01 / pcl123456