**零售供应链中台基础系统 运行指南**

Retail Supply Chain Eco-System Operation Guide

系统分为前端架构和后台两个部分, 以下指令都是基于ubuntu linux 16.04LTS

The system is comprised of two parts: the front-end architecture and the backend. The following commands are based on **ubuntu linux 16.04LTS**.

**前端**

The Front-end

前端使用yarn编译, 由于项目庞大, 必须设置额外的两个参数nodejs参数

The front-end is compiled with **yarn**. Due to the huge size of the project, you must set two additional nodejs parameters.

* NODE\_OPTIONS=--max-old-space-size=10230，增加编译内容， 或者安装并且下载 increase-memory-limit
* PUPPETEER\_SKIP\_CHROMIUM\_DOWNLOAD=1，不下载chrome防止下载时间过长
* NODE\_OPTIONS=--max-old-space-size=10230，increase the compiled content, or install and download increase-memory-limit
* PUPPETEER\_SKIP\_CHROMIUM\_DOWNLOAD=1，don't download chrome to avoid taking too long to download

然后执行

cd retailscm-biz-suite/bizui/ && yarn install && yarn build

Then execute command:

cd retailscm-biz-suite/bizui/ && yarn install && yarn build

下载时间随网络情况而定，编译时间大约从300秒到700秒，此步骤需要一颗强劲的CPU

The download time varies depending on the network, and the compilation time will take about 300 seconds to 700 seconds. This step requires a high performance CPU.

在bizui目录下面的dist目录就会有需要部署的所有的js文件和其他文件，可以部署到任何地方,使用CDN对响应速度帮助很大

In the dist directory under the bizui directory, there will be all the js files and other files that need to be deployed. It can be deployed anywhere. Using CDNs can greatly help the response speed.

**后端**

Backend

后端有反向代理服务器ngnix，servlet容器Resin或者Tomcat（后期换成Spring Boot），数据库服务器MySQL，缓存服务器Redis组成，消息服务器kafka，多层次权限管理需要图数据库arrangodb，外部email服务器，阿里云短信服务器，OSS服务器，极光app消息push服务器，区块链超级账本fabric节点。

Backend has reverse proxy server ngnix, servlet container Resin or Tomcat (late for Spring Boot), database server MySQL, cache server Redis, message server kafka, multi-level rights management needs graph database arrangodb, external email server, Ali cloud SMS server, OSS server, Aurora app message push server, blockchain HyperLedger fabric node.

**下载Resin**

<https://caucho.com/products/resin/download/3-1/gpl>

Download **Resin**

<https://caucho.com/products/resin/download/3-1/gpl>

**安装docker,并且利用国内镜像加速**

sudo curl -sSL https://get.daocloud.io/docker | sh

curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s http://84763bc6.m.daocloud.io

sudo groupadd docker

sudo usermod -aG docker $USER

Install docker and accelerate it by domestic mirror image

sudo curl -sSL https://get.daocloud.io/docker | sh

curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s http://84763bc6.m.daocloud.io

sudo groupadd docker

sudo usermod -aG docker $USER

**安装MYSQL和Redis**

Install **MYSQL** and **Redis**

基本系统运行需要redis和mysql，均通过docker安装，命令如下

docker run -e MYSQL\_ROOT\_PASSWORD=0254891276 -p 3306:3306 --name demo\_db mysql:5.7

docker run -d --name demo\_redis -p 6379:6379 redis

The system needs redis and mysql to run, both are installed by docker, the command is as follows:

docker run -e MYSQL\_ROOT\_PASSWORD=0254891276 -p 3306:3306 --name demo\_db mysql:5.7

docker run -d --name demo\_redis -p 6379:6379 redis

请注意，mysql5.7默认的字符集不是utf8mb4, 需求修改相关配置来支持utf8mb4

Please note that the default character set of mysql5.7 is not utf8mb4, needs to modify the configuration to support utf8mb4.

容器内文件/etc/mysql/my.cnf内容

[client]

default-character-set = utf8mb4

[mysql]

default-character-set = utf8mb4

[mysqld]

character-set-client-handshake = FALSE

character-set-server = utf8mb4

collation-server = utf8mb4\_unicode\_ci

time\_zone =+08:00

In-container file /etc/mysql/my.cnf

[client]

default-character-set = utf8mb4

[mysql]

default-character-set = utf8mb4

[mysqld]

character-set-client-handshake = FALSE

character-set-server = utf8mb4

collation-server = utf8mb4\_unicode\_ci

time\_zone =+08:00

Redis很简单，运行就是了

**Redis** is very simple, just run it.

**编译**

Compilation

java项目使用gradle来编译，为了快速开发， 我们只是把java文件编译成class，其他的目录结构保持不变，建议把输出目录直接设置为 classes并且使用resin的开发模式，这样，当class发生变更的时候，Resin会自动重新装载新的类，无需重新编译和启动，开发体验和写PHP类似。

The java project uses gradle to compile. For fast development, we just compile the java file into class, and the other directory structure remains unchanged. It is recommended to set the output directory directly to classes and use the development mode of resin, so that when the class changes, Resin will automatically reload the new class without recompiling and starting. The development experience is similar to writing PHP.

使用最新的gradle 5.1， sdk install gradle 5.1

cd retailscm-biz-suite/bizcore&& gradle classes

Use the latest gradle 5.1, sdk install gradle 5.1

cd retailscm-biz-suite/bizcore&& gradle classes

这个过程大约在10多秒到20秒这样得到编译后的classes，在WEB-INF/classes

It will take about 10 seconds to 20 seconds to get the compiled classes, which is in the directory WEB-INF/classes

**配置nginx**

Configuring **nginx**

这一步非常简单，拷贝这个文件到 /etc/nginx/sites-enabled/demo, 然后 service ngnix start

This step is very simple, copy this file to /etc/nginx/sites-enabled/demo, then service ngnix start

server {

gzip on;

gzip\_disable "msie6";

gzip\_comp\_level 6;

gzip\_min\_length 1100;

gzip\_buffers 16 8k;

gzip\_proxied any;

gzip\_types

text/plain

text/css

text/js

text/xml

text/javascript

application/javascript

application/x-javascript

application/json

application/xml

application/rss+xml

image/svg+xml;

# SSL configuration

#

# listen 443 ssl default\_server;

# listen [::]:443 ssl default\_server;

#

# Note: You should disable gzip for SSL traffic.

# See: https://bugs.debian.org/773332

#

# Read up on ssl\_ciphers to ensure a secure configuration.

# See: https://bugs.debian.org/765782

#

# Self signed certs generated by the ssl-cert package

# Don't use them in a production server!

#

# include snippets/snakeoil.conf;

root /var/www/html;

# Add index.php to the list if you are using PHP

index index.html index.htm index.nginx-debian.html;

server\_name demo.doublechaintech.com;

location / {

proxy\_pass http://127.0.0.1:8080;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Host $host;

proxy\_set\_header X-Forwarded-Server $host;

proxy\_set\_header X-Forwarded-Port 80;

proxy\_set\_header X-Forwarded-Proto http;

}

listen 443 ssl; # managed by Certbot

ssl\_certificate /etc/letsencrypt/live/demo.doublechaintech.com/fullchain.pem; # managed by Certbot

ssl\_certificate\_key /etc/letsencrypt/live/demo.doublechaintech.com/privkey.pem; # managed by Certbot

include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}

**使用Tomcat容器**

Use the **Tomcat** Container

需要需改bizcore/WEB-INF/web.xml路径匹配规则，其他不用动

The bizcore/WEB-INF/web.xml path matching rule needs to be modified, and nothing else needs to be changed