Institute of Systems Science, National University of Singapore

# Practical Language Processing System User Guide

Food Search & Customer Analytics


**TEAM MEMBERS**

Alfred Tay Wenjie
Li Xinlin
Wang Zilong
Xu JiaChen

MASTER OF TECHNOLOGY

# Contents

1.0 Abstract ..... 1

2.0 Deployment Structure ..... 1

3.0 Dependencies ..... 2

4.0 Key files of Deployment ..... 3

5.0 Deployment Process ..... 5

# 1.0 Abstract

This document is about how to deploy the Food Search & Customer Analytics. Considering the system's architecture and connection between every component, our team choose the docker as our system deployment environment. Using a docker can further improve the portability of our system and can be deployed, so long as has the docker and python3.6 environment any server can only through a line command can complete the deploying and running of the whole system, at the same time because it is based on the docker and docker - compose deployment, each function module is encapsulated as a docker container, so you can compile each module of the enclosed update, improve the maintainability and expansibility of the system

# 2.0 Deployment Structure

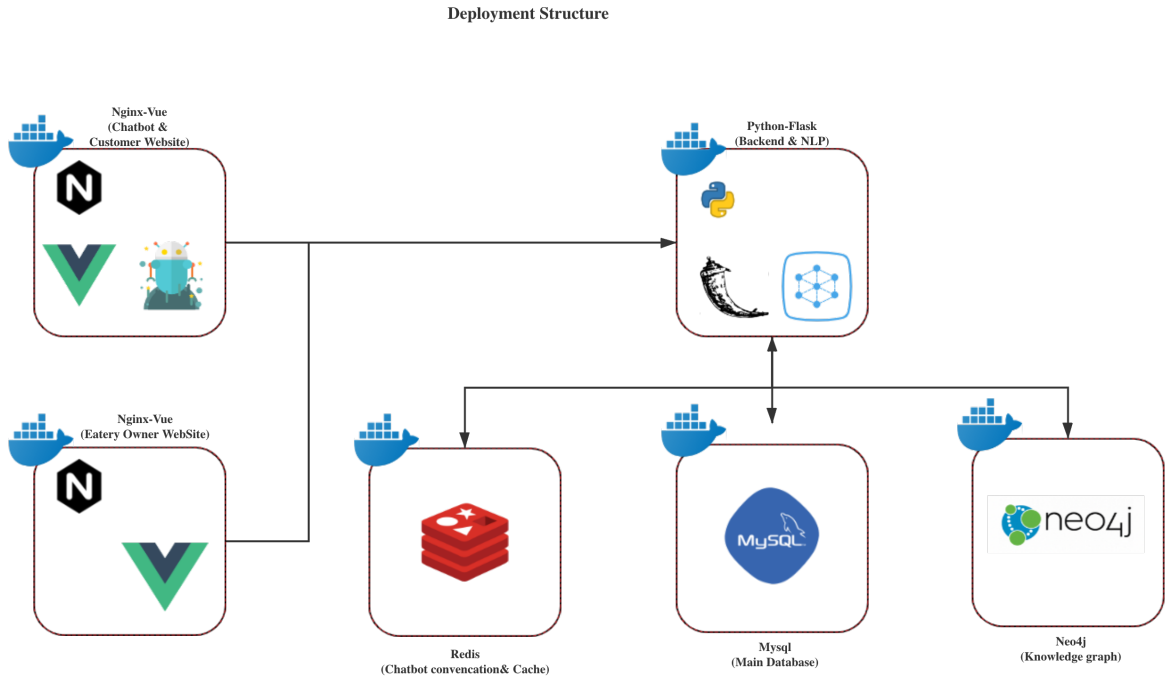


Figure 1 Deployment Structure

In the deployment structure of this system, our team use four docker images to package and deploy components.

Nginx-Vue: this is a docker images based on official Nginx docker image. We use the Nginx to proxy our Single-Page Application based on Vue and we use two Nginx-Vue containers to deploy the frontend for customers and the frontend for eatery owners respectively.

Python-flask: this is a docker images based on official python3.6 docker image. For offline deployment, our team pre-install all the python requirements of the backend and packaged it to be a new docker images -- Python-flask. And because there are several machine learning, deep learning and knowledge

graph components in the system, the image for the backend also has their dependence, like deeppavlov, Keras, Tensorflow and spacy.

MySQL: this is a docker images based on official MySQL 5.6 docker images. To prevent garbled code problems, our team modified the setting of MySQL and packaged it to be a new docker image –- fofshsf/mysql-utf8.

Redis: this is a docker images of official Redis. Redis in the system is mainly responsible for the conversation data storage and query.

Neo4j: this is a docker images of official Neo4j 3.5.14, which is very good and convenient knowledge graph components. Neo4j in the system is mainly responsible for the graph creation, storage and query.

## 3.0 Dependencies

Because the system has multiple machine learning and deep learning models, it has high requirements for deployment and running environments

Local Machine or Virtual Machine:

CPU Cores: 2 cores

Minimum Memory: 10 G

Recommend Memory: 16 G

Docker Environment:  version: 18.09.0

Python environment:  version: Python3.6

## 4.0 Key files of Deployment

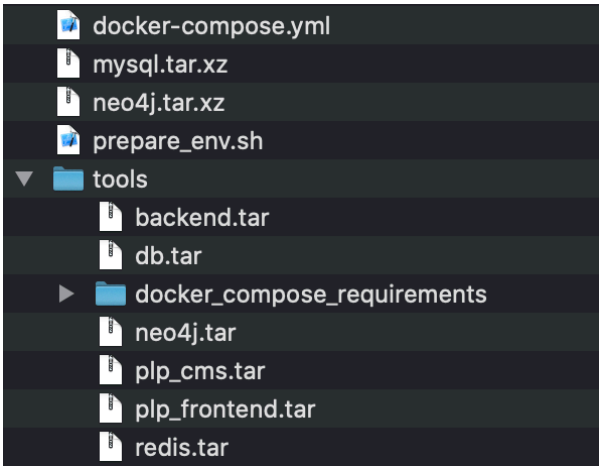


Figure 2 Files of Deployment

The docker-compose.yml, prepare_env.sh, all the files in the tools folder are the most important files in the deployment package. Mysql.tar.xz and Neo4j.tar.xz contain all the data the system needs for the deployment.

The docker-compose.xml is to control our internal system's start and stop, which needs the python3.6 environment.

```
version: '2.1'
services:
  mysql4ca:
    image: tofshsf/mysql-utf8
    ports:
      - "3306:3306"
    volumes:
      - "./mysql/data:/var/lib/mysql"
      - "./mysql/conf/my.cnf:/etc/mysql/my.cnf"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    restart: always

  redis4ca:
    image: redis
    ports:
      - "6379:6379"
    restart: always

  neo4j4ca:
    image: neo4j:3.5.14
    ports:
      - "7474:7474"
    volumes:
      - "./neo4j/data:/data"
      - "./neo4j/logs:/logs"
      - "./neo4j/import:/var/lib/neo4j/import"
      - "./neo4j/plugins:/plugins"
    environment:
      NEO4J_AUTH: neo4j/test
    restart: always

  backend:
    image: plp_backend_prod
    ports:
      - "5000:5000"
    depends_on:
      - mysql4ca
      - redis4ca
      - neo4j4ca
    mem_limit: 10G
    memswap_limit: 10G
    oom_kill_disable: true
    restart: always

  frontend:
    image: plp_frontend
    restart: always
    ports:
      - "8080:80"
    depends_on:
      - backend

  cms:
    image: plp_cms
    restart: always
    ports:
      - "8081:80"
    depends_on:
      - backend
```

Figure 3 docker-compose.yml

The prepare_env.sh is the shell script to prepare the environment of deployment and start our system.

```
#!/bin/sh

echo "loading docker images"
docker load < ./tools/db.tar
docker load < ./tools/plp_frontend.tar
docker load < ./tools/plp_cms.tar
docker load < ./tools/neo4j.tar
docker load < ./tools/redis.tar
docker load < ./tools/backend.tar

echo "unzip data"
tar -xvf mysql.tar.xz
tar -xvf neo4j.tar.xz

echo "install docker-compose"
pip install docker-compose -f ./tools/docker_compose_requirements/requirements.txt

docker-compose down
docker-compose up
```

Figure 4 prepare_env.sh

The tools folder contains all the required python libraries by docker-compose and all the docker images our system needs. This folder is very prepared for the offline deployment.

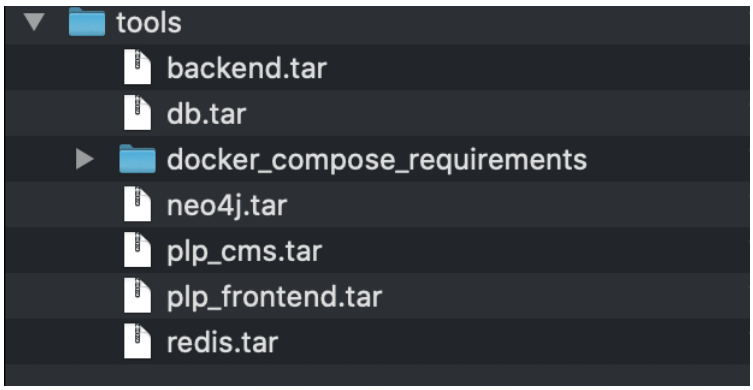


Figure 5 tools folder

## 5.0 Deployment Process

For the deployment, our team use the ISS-VM provided by Sam, which is for the docker and python3.6 environment, if you have your own server with docker and python3.6, you can pass the step of VM download.

If you have trouble with the deployment process, you can see the deployment video (https://youtu.be/HmR8BYN2fGQ) or you can email with any member in the teams we are very happy to help you!

1. Please download and install iss-vm according to this document https://github.com/telescopeuser/iss-vm/blob/master/User%20Guide%20for%20iss-vm.pdf

   Double Click the iss-vm-v17 to open the virtual machine

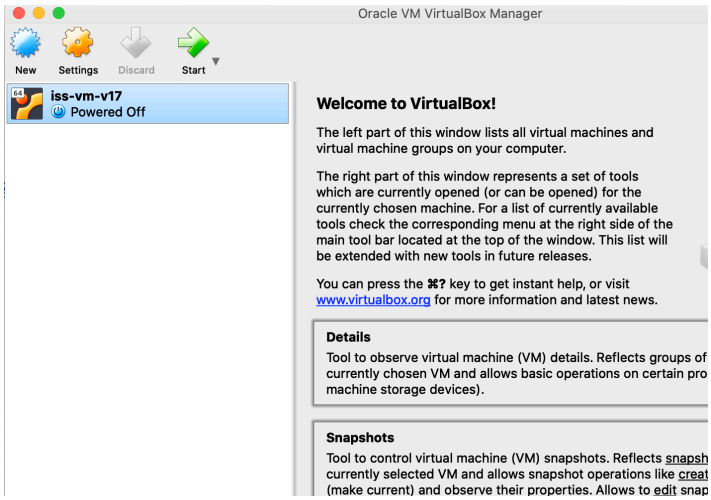


Figure 6 Install ISS-VM

2. Please download the system's docker Deployment Package (It is on the Google Drive, because it is too big for the GitHub)

https://drive.google.com/open?id=1dQ185hNQqPTvm8oeAsfmCbopUZcS4ZCA

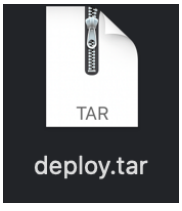


Figure 7 deploy.tar

3. Right click on VM desktop and click `Open a Terminal` to Open a Terminal
4. Please use `mv /home/iss-user/Downloads/deploy.tar /home/iss-user/Desktop` command to move the deployment package to the Desktop

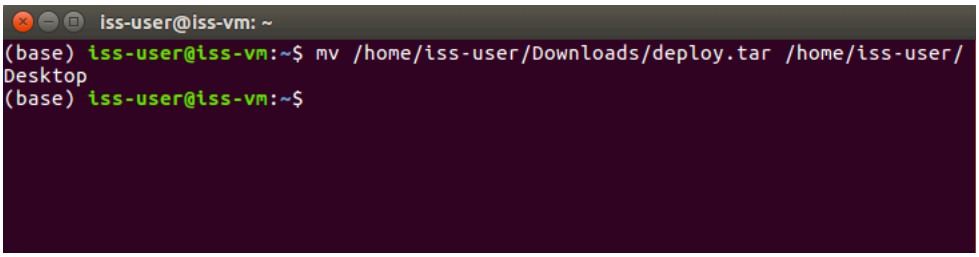


Figure 8 mv /home/iss-user/Downloads/deploy.tar /home/iss-user/Desktop

5. Please use `cd /home/iss-user/Desktop` command to change current location

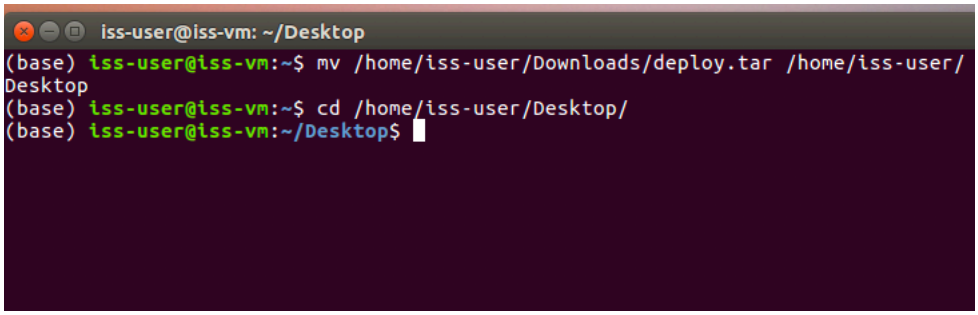


Figure 9 cd /home/iss-user

6. Please use `tar -xvf deploy.tar` to extract the deployment file

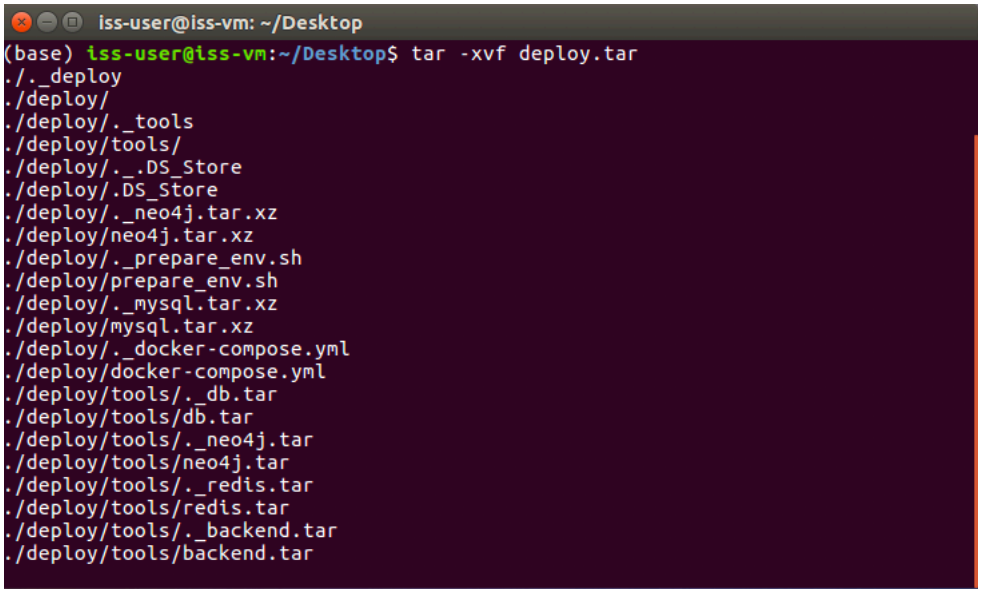


Figure 10 tar -xvf Deploy.tar

7. Please use `cd deploy` to enter the extracted deployment file

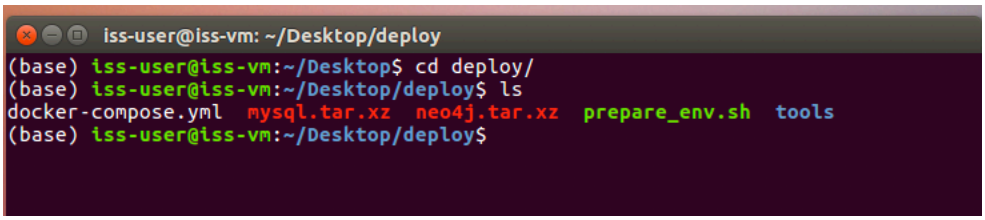


Figure 11 cd Deploy/ITAS

8. Please use `bash prepare_env.sh` command to prepare the environment and start the system

It will take about 5-8 minutes depending on the system to prepare the docker and python environment, then start the system. Because the system contains some very large models, please wait patiently for it.

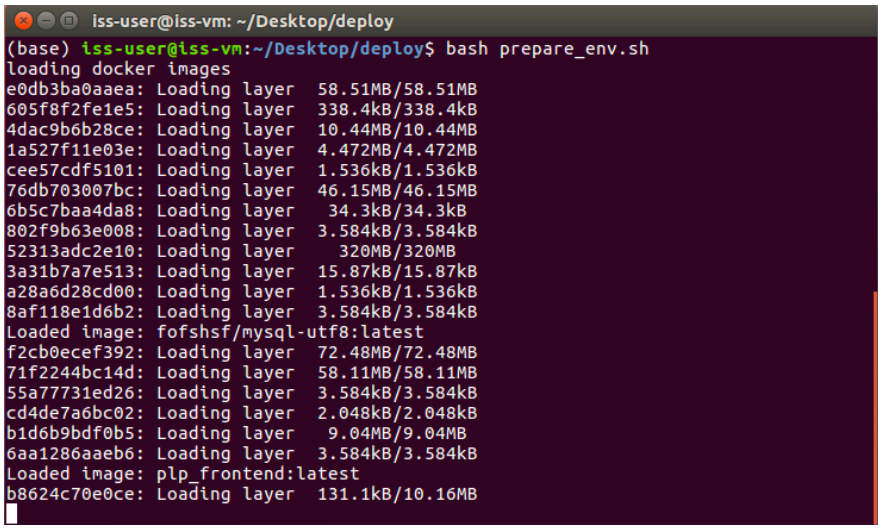


Figure 12 bash prepare_env.sh

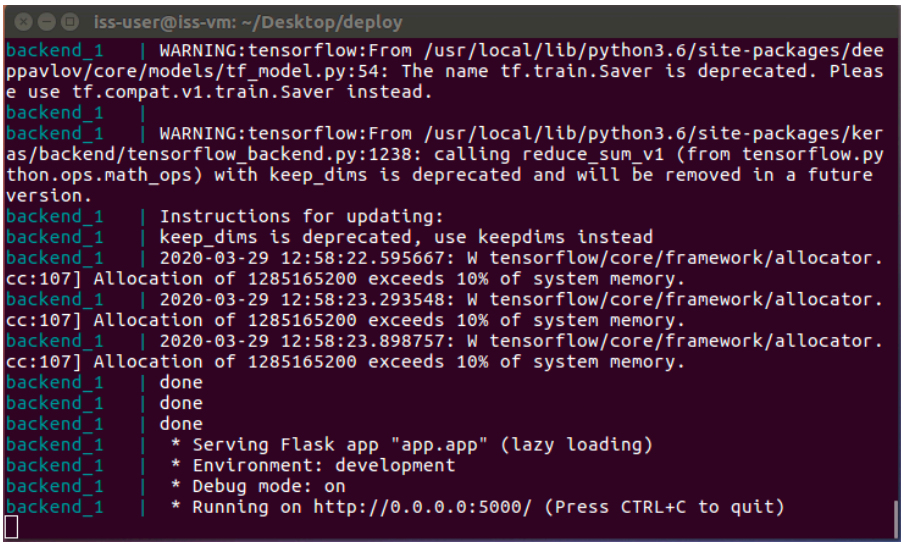


Figure 13 deployment done

If you can see the above screenshot, the system has started. Congratulation! Now you can open the Chrome and go to our two website.

localhost:8080 Customer Website

localhost:8081 Eatery Owner Website

Please refer to the User Guide of the system to use the system.