DOCKER IMAGE BUILDING FOR CAN DEPLOYMENT

Cognitive Assistant for Networks (CAN) Release 6.0

JANUARY 18, 2023

AVANSEUS TECHNOLOGY PVT. LTD.

REVISION HISTORY

Version	Date	Change description	Created By	Updated by	Reviewed by
V5.5	Aug, 2021	Initial Release	Umesh/Naveen	Sandeep Singh	Chiranjib
V6.0	Jan, 2023	Release 6.0	Neha/Naveen	Raksha N	Chiranjib

Table of Contents

DOCKER BUILDING IMAGES

1. Focus

This document focuses on building an image of CAN application and all other supporting applications. This document is mainly for the Avanseus team who builds the Docker image before distributing. Docker images created for CAN applications to run are:

1. MongoDB image
2. OpenDJ LDAP image
3. VBI python module image
4. CAN application image (CAN Module)
5. CAS application image (CAS Module)
6. Prediction Controller image
7. Prediction Worker Image
8. Batch handler image
9. Record processor image
10. Python processor image
11. RTSP image
12. BCXP image
13. ThreeGPP image
14. Ansible processor image

2. Prerequisites

Private Docker Registry server for storing and distributing the docker images.

3. Image Creation

Note: For automatically compiling and building all the applications (CAN, CAS, PredictionWorker & PredictionController), creating docker builds and pushing them to desired Docker registry repository, please refer the document "**CAN 6.0 Jenkins Auto Build for Applications**".

3.1 MongoDB Image

The DockerHub repository, by default, provides a MongoDB image. It is used to create the setup.

Command to download MongoDB-4.4 image:

```
$sudo docker pull mongo:4.4
```

Command to push the built image to the private docker registry:

```
$sudo docker tag mongo:4.4 <IP/DOMAIN>/<PATH>/mongo:v1
$sudo docker push <IP/DOMAIN>/<PATH>/mongo:v1
```

IP/DOMAIN: Domain name or IP address of the registry server.

PATH: Path where all the docker images will be stored.

3.2 OpenDJ LDAP Image

User will find all the files required for creating an OPENDJLDAP image in kubernetes_resources/DOCKER_FILES// folder of the GIT repository.

Command to build the OPENDJ LDAP:

```
$cd kubernetes_resources/DOCKER_FILES/ OPENDJLDAP/
```

```
$sudo docker build . -t avanseuscontainer.com/release/6.0/ldap:v1 --build-arg
VERSION=4.4.14
```

Command for pushing the built image to the private docker registry is as follows:

```
$sudo docker tag ldap:4.4 <IP/DOMAIN>/<PATH>/ldap:v1

$sudo docker push <IP/DOMAIN>/<PATH>/ldap:v1
```

3.3 VBI Python Module Image

- Download and use the quickstart package for building VBI images from this link. Un-tar and user will find a directory with important artifacts needed for building VBI images.
- Go into the “VBI” directory of the quickstart package. Update the latest python script file in the directory with name “docker_socket_server_json.py”.

Command to build the image:

```
$sudo docker build . -t vbi:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag vbi:v1 <IP/DOMAIN>/<PATH>/vbi:v1

$sudo docker push <IP/DOMAIN>/<PATH>/vbi:v1
```

3.4 CAN Application Image (CAN Module)

User will find all the files required for creating a CAN image in kubernetes_resources/DOCKER_FILES/CAN/ folder of the GIT repository. Go to kubernetes_resources/DOCKER_FILES/CAN/.

- Add the newly built CAN war (CAN.war) file in this folder.
- Update the nasmout folder contents with the contents of the folder MasterTables/nasPath/ of the GIT repository.

```
$cp -r MasterTables/nasPath/* kubernetes_resources/DOCKER_FILES/CAN/nasmount/
```

Command to build the image:

```
$cd kubernetes_resources/DOCKER_FILES/CAN/

$sudo docker build . -t can:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag can:v1 <IP/DOMAIN>/<PATH>/can:v1

$sudo docker push <IP/DOMAIN>/<PATH>/can:v1
```

3.5 CAS Application Image (CAS Module)

- User will find all the files required for creating a CAS image in kubernetes_resources/DOCKER_FILES/CAS/ folder of the GIT repository. Go to kubernetes_resources/DOCKER_FILES/CAS/.

- Add the newly built CAS war (CAS.war) file in this directory.

Command to build the image:

```
$cd kubernetes_resources/DOCKER_FILES/CAS/
$sudo docker build . -t cas:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag cas:v1 <IP/DOMAIN>/<PATH>/cas:v1
$sudo docker push <IP/DOMAIN>/<PATH>/cas:v1
```

3.6 Prediction Controller Image

- User will find all the files required for creating a PredictionController image in kubernetes_resources/DOCKER_FILES/PredictionController/ folder of the GIT repository. Go to kubernetes_resources/DOCKER_FILES/PredictionController/.
- Add the newly built Prediction Controller war (PredictionController.war) file in this directory.

Command to build the image:

```
$cd kubernetes_resources/DOCKER_FILES/PredictionController/
$sudo docker build . -t predictioncontroller:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag predictioncontroller:v1 <IP/DOMAIN>/<PATH>/predictioncontroller:v1
$sudo docker push <IP/DOMAIN>/<PATH>/predictioncontroller:v1
```

3.7 Prediction Worker Image

Prerequisite: For building the CPP executable binary file of PredictionWorker, install C++ 11 version. Use “make clean” and “make” commands to clean and build the CPP executable file. If C++ 11 version is not installed in the local system for building the PredictionWorker, refer the document “**CAN 6.0 Jenkins Auto Build for Applications**”.

- User will find all the files required for creating a PredictionWorker image in kubernetes_resources/DOCKER_FILES/PredictionWorker/ folder of the GIT repository. Go to kubernetes_resources/DOCKER_FILES/PredictionWorker/.
- Update the latest build of the PredictionWorker i.e., universal file.

Command to build the image:

```
$cd kubernetes_resources/DOCKER_FILES/PredictionWorker/
$sudo docker build . -t predictionworker:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag predictionworker:v1 <IP/DOMAIN>/<PATH>/predictionworker:v1
$sudo docker push <IP/DOMAIN>/<PATH>/predictionworker:v1
```

3.8 BatchHandler Image

- User will find all the files required for creating a BatchHandler image in kubernetes_resources/DOCKER_FILES/BatchHandler/ folder of the GIT repository.
- Add the newly built BatchHandler jar (BatchHandler.jar) file in this folder.

Command to build the BatchHandler image:

```
$cd kubernetes_resources/DOCKER_FILES/BatchHandler/
$sudo docker build . -t batchhandler:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag batchhandler:v1 <IP/DOMAIN>/<PATH>/batchhandler:v1
$sudo docker push <IP/DOMAIN>/<PATH>/batchhandler:v1
```

3.9 RecordProcessor Image

- User will find all the files required for creating a RecordProcessor image in kubernetes_resources/DOCKER_FILES/RecordProcesssor/ folder of the GIT repository.
- Add the newly built RecordProcessor jar (RecordProcessor.jar) file in this folder.

Command to build the RecordProcessor:

```
$cd kubernetes_resources/DOCKER_FILES/RecordProcessor/
$sudo docker build . -t recordprocessor:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag recordprocessor:v1 <IP/DOMAIN>/<PATH>/recordprocessor:v1
$sudo docker push <IP/DOMAIN>/<PATH>/recordprocessor:v1
```

3.10 PythonProcessor Image

- User will find all the files required for creating a PythonProcessor image in kubernetes_resources/DOCKER_FILES/PythonProcessor/ folder of the GIT repository.
- Copy the python code files from PythonProcessor-server/Python-server/python/ directory in the local repository to kubernetes_resources/DOCKER_FILES/PythonProcessor/ folder of the GIT repository.

Command to build the PythonProcessor:

```
$cd kubernetes_resources/DOCKER_FILES/pythonprocessor/
$ sudo docker build . -t pythonprocessor:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag pythonprocessor:v1 <IP/DOMAIN>/<PATH>/pythonprocessor:v1
$sudo docker push <IP/DOMAIN>/<PATH>/pythonprocessor:v1
```

3.11 RTSP Image

- User will find all the files required for creating a RTSP image in kubernetes_resources/DOCKER_FILES/RTSP/ folder of the GIT repository.
- Add the newly built RTSP jar (RTSP.jar) file in this folder.

Command to build the RTSP:

```
$cd kubernetes_resources/DOCKER_FILES/RTSP/
$sudo docker build . -t rtsp:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag rtsp:v1 <IP/DOMAIN>/<PATH>/rtsp:v1
$sudo docker push <IP/DOMAIN>/<PATH>/rtsp:v1
```

3.12 BCXP Image

- User will find all the files required for creating a Bcxp image in kubernetes_resources/DOCKER_FILES/Bcxp/ folder of the GIT repository.
- Add the newly built Bcxp war (Bcxp.war) file in this folder.

Command to build the BCXP:

```
$cd kubernetes_resources/DOCKER_FILES/Bcxp/
$sudo docker build . -t bcxp:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag bcxp:v1 <IP/DOMAIN>/<PATH>/bcxp:v1
$sudo docker push <IP/DOMAIN>/<PATH>/bcxp:v1
```

3.13 ThreeGPP Image

- User will find all the files required for creating a ThreeGPP image in kubernetes_resources/DOCKER_FILES/ThreeGPP folder of the GIT repository.
- Add the newly built ThreeGPP jar (ThreeGPP.jar) file in this folder.

Command to build the ThreeGPP:

```
$cd kubernetes_resources/DOCKER_FILES/ ThreeGPP/
$ sudo docker build . -t threegpp:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag threegpp:v1 <IP/DOMAIN>/<PATH>/threegpp:v1
$sudo docker push <IP/DOMAIN>/<PATH>/threegpp:v1
```

3.14 AnsibleProcessor Image

- User will find all the files required for creating an AnsibleProcessor image in kubernetes_resources/DOCKER_FILES/AnsibleProcessor/ folder of the GIT repository.
- Add the newly built AnsibleProcessor jar (AnsibleProcessor.jar) file in this folder.

Command to build the BatchHandler image:

```
$cd kubernetes_resources/DOCKER_FILES/AnsibleProcessor/
$sudo docker build . -t ansibleprocessor:v1
```

Command to push the built image to the private docker registry:

```
$sudo docker tag ansibleprocessor:v1 <IP/DOMAIN>/<PATH>/ansibleprocessor:v1
$sudo docker push <IP/DOMAIN>/<PATH>/ansibleprocessor:v1
```