DOCKER IMAGE BUILDING FOR CAN DEPLOYMENT

Cognitive Assistant for Networks (CAN) Release 5.5

AUGUST 4, 2021

AVANSEUS TECHNOLOGY PVT. LTD.

REVISION HISTORY

Version	Date	Change description	Created by	Updated by	Reviewed by
V 5.5	Aug, 2021	Initial Release	Umesh	Sandeep Singh	Chiranjib

Table of Contents

1. Focus

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

1. MongoDB image
2. LDAP image (With OpenDS and one default user)
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

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 documentation “**CAN 5.5 Jenkins Auto Build for Applications**”.

3.1. MongoDB Image

The DockerHub repository, by default, provides a MongoDB image. We will use that to create the setup.

Command to download MongoDB-4.4 image:

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

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

```
$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. LDAP Image

You will find all the files required for creating a LDAP image in ***kubernetes_resources/DOCKER_FILES/LDAP/*** folder of the GIT repository. The LDAP package already consists of a LDAP user whose details would be shared separately.

Build the LDAP image by running the following command:

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

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

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

3.3. VBI Python Module Image

Please download and use the ***quickstart package*** for building VBI images from this link. Un-tar and you 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".

Build the image by running the following command:

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

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

```
$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)

You 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/
```

Build the image by running the following command:

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

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

```
$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)

You 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.

Build the image by running the following commands:

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

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

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

3.6. Prediction Controller Image

You 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.

Build the image by running the following command:

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

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

```
$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 we need to have C++ 11 version installed. Use “make clean” and “make” commands to clean and build the CPP executable file. If you don't have C++ 11 version installed in the local system for building the PredictionWorker, please follow the document “**CAN 5.5 Jenkins Auto Build for Applications**”.

You 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.

Build the image by running the following command:

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

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

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