nFlows Installation Manual - Docker

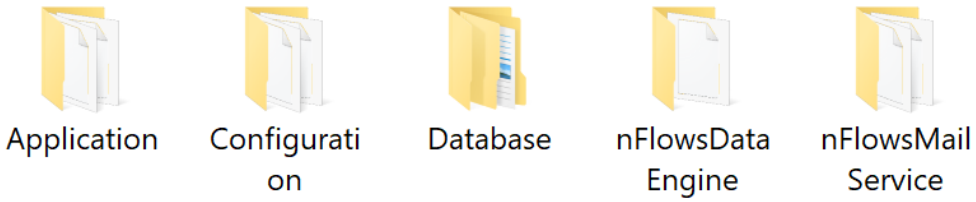
Table of Contents

1. Pre - Requisites

- Hope you have received the installation package from Stradegi.
- The user used for deployment needs to have below access privileges

S. No	Access Requirement	Description of Access	RWE (Read/Write/Execute)	Remarks
1	chmod	Able to change access for the files / folders for the mounted volume and release package path	RWE	This command will help to deploy the latest release in the mounted volume and configuration file change.
2	Secure File Access	Able to login and download the release package from Secure File repository of NTRS	R	
3	Docker Commands	Able to execute all docker commands mentioned in the document	RWE	

- Please download the Release package from the SecureFile.ntrs link.
- Extract the package named **nFlows_Suite(V4.0.140.1).zip.**
- You will find the folders named Application, Database, Configuration, nFlowsDataEngine and nFlowsMailService as shown in the below screenshot.



2. Neo4j Installation

2.1. Pre-Requisites

- Open the **Database** folder.
- Navigate to **Neo4jDB** folder inside Database folder.
- You can find the below list of files in Neo4jDB Folder.

Folder Name	Folder/File Name	Type
Neo4jDB	Neo4jDockerFile	File
	Neo4j	Folder

2.2. Building Docker Image

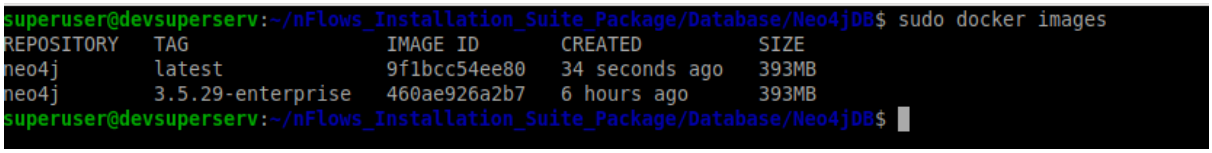
- Open Terminal in **Neo4jDB** Folder.
- Run the below command (including the last period) in terminal to build the neo4j image.

docker image build -t neo4j -f Neo4jDockerFile .

- Once the above command is executed, please verify whether the image is built using the below command.

docker images

- You can find that the neo4j image is built as shown in the below screenshot.



2.3. Spinning up Neo4j Container

- Before running the container, we can create a volume for neo4j.
- Create a folder named **nFlows_DB** in your desired location.
- Copy the folder named **Neo4j** from the package (mentioned in 2.1) and paste inside **nFlows_DB** as shown in below screenshot.
- Give full access to **Neo4j** folder by opening the terminal in the same path and enter the below command.

sudo chmod -R 777 .

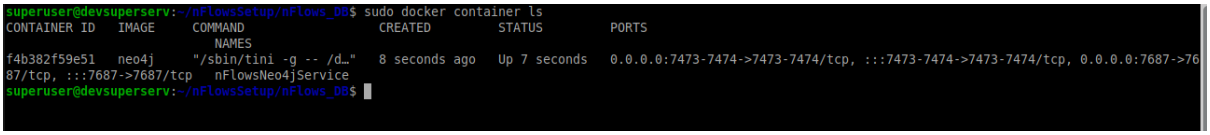
- We can use this **Neo4j** folder as our **volume for neo4j container**.
- Copy the path till Neo4j folder.
- In the below command paste the above copied path in place of <MOUNTED_VOLUME> and provide password in place of <PASSWORD>.

```
docker run -it -d -p 7474:7474 -p 7687:7687 -p 7473:7473 -e
NEO4J_AUTH=neo4j/<PASSWORD> -v <MOUNTED_VOLUME>/conf:/home -v
<MOUNTED_VOLUME>/data:/data -v <MOUNTED_VOLUME>/import:/import -v
<MOUNTED_VOLUME>/logs:/logs -v <MOUNTED_VOLUME>/plugins:/plugins --
name nFlowsNeo4jService neo4j
```

- Run the above command in terminal to create Neo4j container.
- Verify whether the container is up by running the below command

docker container ls

- You can find that neo4j has been running with the container name **nFlowsNeo4jService.**



- Please make note of Neo4j Password replaced in the above command as you may need this while setting up the application.

2.4. Neo4j Config File Movement

- Navigate to Neo4j DB Mounted Volume/conf folder in the server.
- Open Terminal in that folder.
- Replace with your container name in the below command and execute it in Terminal.

  docker cp <CONTAINER_NAME>:/var/lib/neo4j/conf/neo4j.conf ./
- Find **neo4j.conf** file will be present in the current folder. Open the **neo4j.conf** file in any text editor and find the below line.

  #dbms.security.allow_csv_import_from_file_urls=true
- If hash (#) is present in front of the above line, then remove the hash (#) for enabling to allow the CSV load and save the file.
- Replace with Neo4j container name in the below command and execute it in Terminal.

  docker exec -it -t <CONTAINER_NAME> neo4j-admin memrec
- Open neo4j.conf file in any text editor and find the below line.

  dbms.memory.heap.max_size

  dbms.memory.pagecache.size
- Based upon the recommended memory settings, change the neo4j.conf file respectively and save it.
- Memory settings will be denoted as MB that needs to be changed to GB while changing it in neo4j.conf file.
- For Example, if the value is mentioned as 8000m then it needs to be updated as 8g in neo4j.conf file.
- Open Terminal in the current folder and replace the container name in the below command to move the neo4j.conf file into the container.

  docker cp neo4j.conf <CONTAINER_NAME>:/var/lib/neo4j/conf
- Restart the Neo4j DB Container using the below command.

  docker container restart <CONTAINER_NAME>

3. MongoDB Installation

3.1. Pre-Requisites

- Open the **Database** folder.
- Navigate to **MongoDB** folder inside Database folder.
- You can find the below list of files in MongoDB Folder.

Folder Name	Folder/File Name	Type
MongoDB	MongoDBDockerFile	File
	nFlows	Folder

3.2. Building Docker Image

- Open Terminal in **MongoDB** Folder.

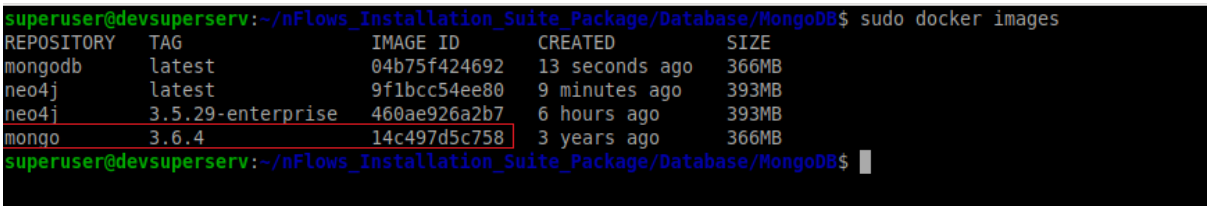
- Run the below command (including the last period) in terminal to build the mongo image.

docker build -t mongodb -f MongoDBDockerFile .

- Once the above command is executed, please verify whether the image is built using the below command.

docker images

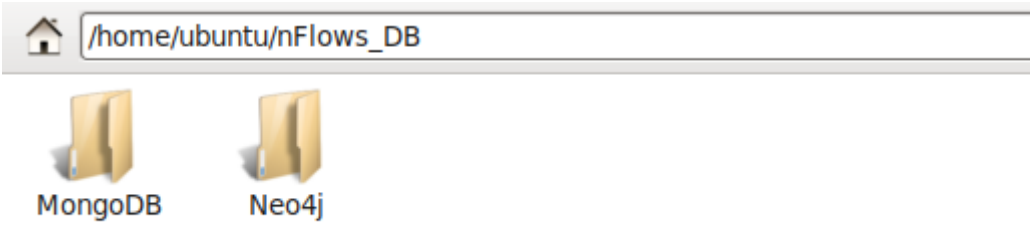
- You can find that the mongo image is built as shown in below screenshot.



3.3. Spinning up MongoDB Container

- Before running the container, can create a volume for MongoDB.
- Open the folder named **nFlows_DB** which you have created in the previous section 2.3.
- Create a folder named **MongoDB** inside **nFlows_DB** as shown in below screenshot.
- Give full access to **MongoDB** folder by opening the terminal in the same path and enter the below command.

sudo chmod -R 777 .



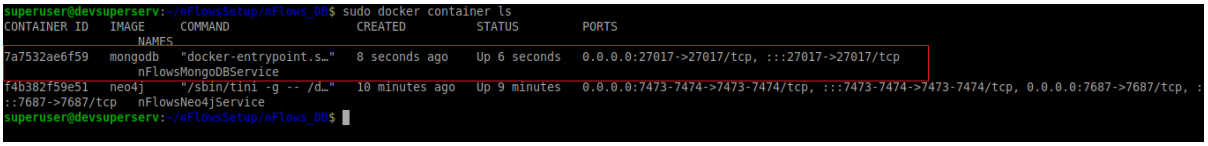
- We can use this **MongoDB** folder as our **volume for MongoDB container**.
- Copy the path till MongoDB folder.
- In the below command paste the above copied path in place of <MOUNTED_VOLUME> and provide username and password in place of respective <ROOT_USER> and <ROOT_PASSWORD>.

```
docker run -it -d -p 27017:27017 -v <MOUNTED_VOLUME>/data:/data/db -
e MONGO_INITDB_ROOT_USERNAME=<ROOT_USER> -e
MONGO_INITDB_ROOT_PASSWORD=<ROOT_PASSWORD> --name
nFlowsMongoDBService mongodb
```

- Run the above command in terminal to create MongoDB container.
- Verify whether the container is up by running the below command

docker container ls

- You can find that mongodb has been running with the container name **nFlowsMongoDBService.**



3.4. Restoring DB

- Navigate to MongoDB folder as mentioned in **Section 3.1**.
- To restore nFlows DB in MongoDB, need to move nFlows DB folder inside the container.
- Open the terminal in the current folder and enter the below command. You can find the container name as mentioned in end of **Section 3.3**

```
docker cp nFlows <CONTAINER_NAME>:/backup/
```

- After moving the DB backup to container, replace <CONTAINER_NAME>, <ROOT_USER> and <ROOT_PASSWORD> with the username and password that you have setup in **Section 3.3** in the below command to restore the DB.

```
docker exec <CONTAINER_NAME> sh -c 'mongorestore  --
authenticationDatabase admin -d nFlows -u <ROOT_USER> -p
<ROOT_PASSWORD> backup/nFlows'
```

3.5. Setting Credentials to nFlows DB.

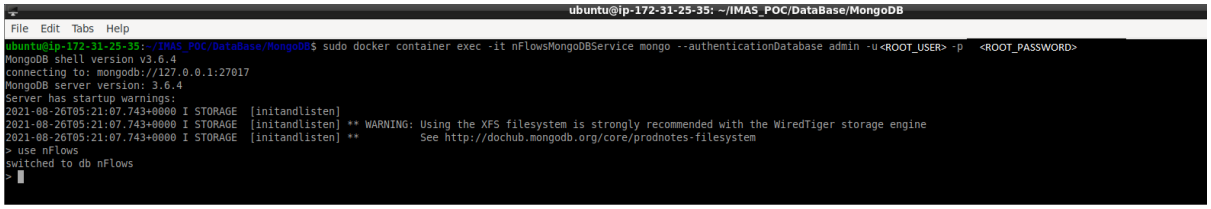
- Replace <CONTAINER_NAME>, <ROOT_USER> and <ROOT_PASSWORD> with the username and password that you have setup in **Section 3.3** in the below command and execute it in the Terminal to open the Mongo Shell.

```
docker container exec -it <CONTAINER_NAME> mongo --
authenticationDatabase admin -u <ROOT_USER> -p <ROOT_PASSWORD>
```

- In Mongo Shell use the below command to switch to nFlows DB.

  use nFlows

- You can find that nFlows DB has been switched as shown in below screenshot.



- Now enter the below command by replacing <USERNAME> and <PASSWORD> which will be assigned as the authorization credentials for nFlows DB.

  db.createUser({
  user:'<USERNAME>',
  pwd:'<PASSWORD>',
  roles:[{role:'readWrite',db:'nFlows'}]})

```
> use nFlows
switched to db nFlows
> db.createUser({
... user:'admin',
... pwd:'nFlows123',
... roles:[{role:'readWrite',db:'nFlows'}]})
Successfully added user: {
        "user" : "admin",
        "roles" : [
                {
                        "role" : "readWrite",
                        "db" : "nFlows"
                }
        ]
}
>
```

- To check whether the credentials have been setup without any failure, use the below command in mongoshell.

`db.auth('<USERNAME>','<PASSWORD>')`

```
> db.auth('admin','nFlows123')
1
>
```

4. Application Installation

4.1. Pre-Requisites

- Open the **Application** folder.
- You can find the below list of files in Application Folder.
- Keystore file and password necessary for the step 4.2 (This step only needed for HTTPS Configuration)

Folder Name	Folder/File Name	Type
Application	nFlowsAppDockerFile	File
	nFlows.war	File
	server.xml	File
	setenv.sh	File

4.2. SSL Configuration for Public HTTPS Setup

- If SSL needs to be configured for cloud HTTPS, then follow Section 4.2
- Open the **Application** folder and open **server.xml** in edit mode.
- Search for nFlows SSL Configuration in server.xml.
- Replace the **keystore file name** in place of **REPLACE_KEYSTORE_FILE_NAME** and **keystore password** in place of REPLACE_KEYSTORE_PASSWORD in **server.xml**.

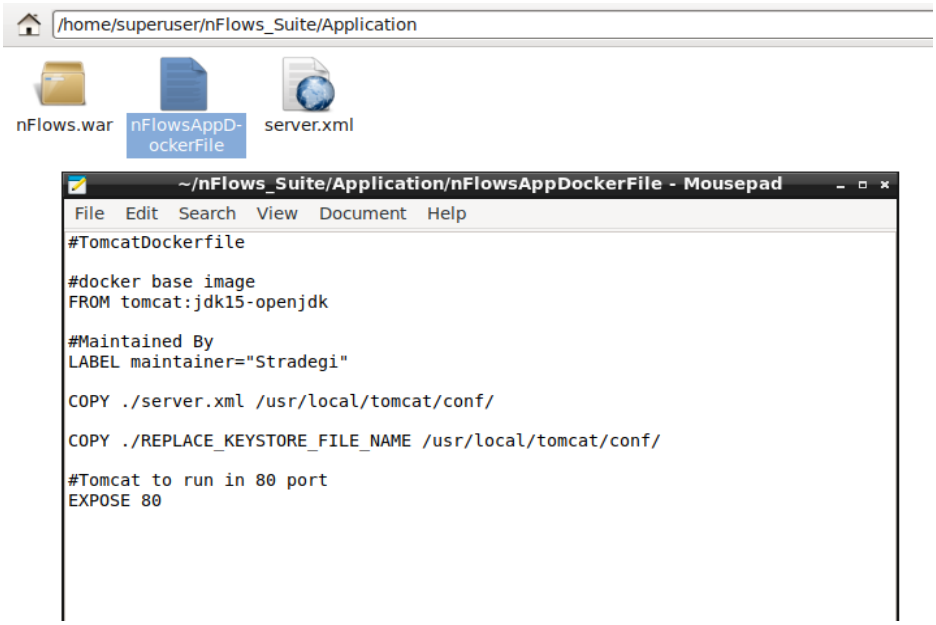
```
            connectionTimeout="20000"
            redirectPort="8443" />
    -->
<!--Start nFlows SSL Configuration -->
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true" keystoreFile="conf/REPLACE_KEYSTORE_FILE_NAME"
           keystorePass="REPLACE_KEYSTORE_PASSWORD" clientAuth="false" sslProtocol="TLS"/>
<!--End nFlows SSL Configuration -->
    <!-- Define an SSL/TLS HTTP/1.1 Connector on port 8443
         This connector uses the NIO implementation. The default
         SSLImplementation will depend on the presence of the APR/native
```

4.3. SSL Configuration for Intranet HTTPS Setup

- For configuring SSL for Intranet HTTPS, then kindly follow the document name **"Linux Red Hat server -HTTPS.docx"** which was already shared with NT.

4.4. Building Docker Image

- Open Terminal in **Application** Folder.
- Copy the **keystore file** and paste in **Application** Folder.
- Now open **nFlowsAppDockerFile** in **edit mode** and replace the **keystore file name** in place of **REPLACE_KEYSTORE_FILE_NAME** as shown in below screenshot.



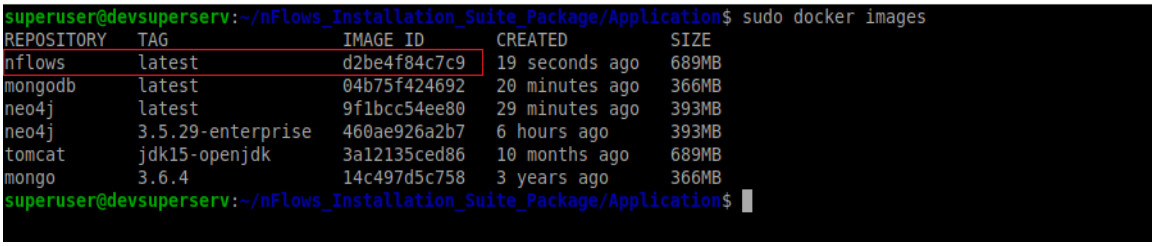
- Save and close the file.
- Run the below command in terminal to build the nFlows Application image.

docker build -t nflows -f nFlowsAppDockerFile .

- Once the above command is executed, please verify whether the image is built using the below command.

docker images

- You can find that the nflows image is built as shown in below screenshot. There might be more than below number of rows found, if all the installation on the single server.



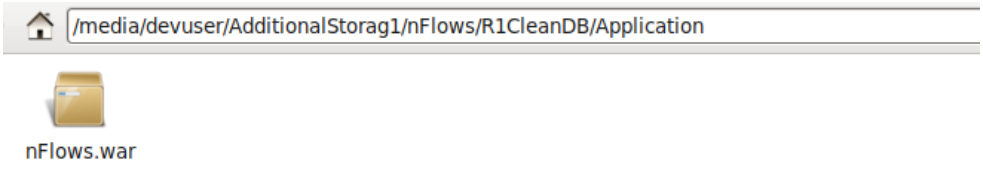
4.5. Creating a Custom Bridge Network

- A custom bridge network must be created since our app container and highcharts container (Section 7) must run on same network.
- Run the below command in terminal to create a bridge network by replacing the name of the network in place of <**NETWORK_NAME>**.
- <NETWORK_NAME> can be any user defined name.

```
docker network create -d bridge <NETWORK_NAME>
```

4.6. Spinning up App Container

- Before running the container, create a volume for nFlows Application.
- Create a folder named **Application** in your desired location.
- Copy **nFlows** WAR file from Application package (mentioned in section 4.1) and paste it in the above folder as show in below screenshot.



- If you want to customize the application name, then you need rename the nFlows war file name according to your customized application
- Give full access to **Application** folder by opening the terminal in the same path and enter the below command.

sudo chmod -R 777 .

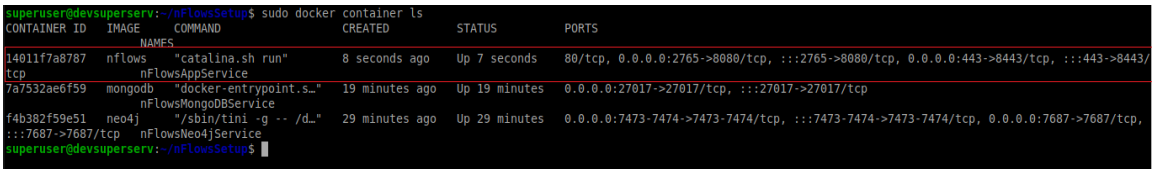
- We can use this folder as our **volume for nFlows App container**.
- Copy the path till Application folder.
- In the below command paste the above copied path in place of <MOUNTED_VOLUME> and Network name in place of <NETWORK_NAME> which was created in **Section 4.4**.

```
docker run -it -d -p 80:8080 -p 443:8443 --network=<NETWORK_NAME> -v
<MOUNTED_VOLUME>:/usr/local/tomcat/webapps --name nFlowsAppService
nflows
```

- Run the above command in terminal to create nFlows Application container.
- Verify whether the container is up by running the below command

docker container ls

- You can find that nflows app has been running with the container name **nFlowsAppService.**



- Stop the above container as of now since we need to generate Config File which will be covered in further sections.
- You can stop the above container using the below command

docker container stop nFlowsAppService

5. Config File Generation

5.1. Pre-Requisites

- Open the **Configuration** folder.
- You can find the below list of files in Configuration Folder.

Folder Name	File Name	Type
Configuration	nFlowsUtility.jar	File
	GenerateConfigFile.sh	File

5.2. Generation of Config File

- Open the **Configuration** folder.
- You can find a script file named **GenerateConfigFile.sh.**
- Open the above file in edit mode as shown in below screenshot.

- Edit the values as per the instructions given below

TOMCAT Configuration

APP_NAME=nFlows (if you have customized the application name as mentioned in the section 4.5, you need to change this property accordingly)
APP_URL=https://<IP OR HOST_NAME> (If you are not using SSL configuration, then **https to be changed to http)**
PATH=/usr/local/tomcat/webapps/nFlows/ (don't change this default location)
PORT=443 (If you are not using HTTPS then, Port should be changed to 80)

NEO4J Configuration

NEO4J_HOST=< IP OR HOST_NAME >
NEO4J_PORT=<NEO4J_BOLT> (Given in Section 2.3)
NEO4J_USER=<NEO4J_USER> (Given in Section 2.3)
NEO4J_PWD=<NEO4J_PASS> (Given in Section 2.3)

MONGODB Configuration

MONGODB_HOST=< IP OR HOST_NAME >
MONGODB _PORT=<MONGO_PORT>
MONGODB _USER=<MONGO_USER_WHICH_IS_CREATED_FOR_nFLOWS_DB> (For Username Refer Section 3.5)
MONGODB _PWD=<MONGO_PASS_WHICH_IS_CREATED_FOR_nFLOWS_DB > (For Password Refer Section 3.5)
MONGODB_NAME=nFlows

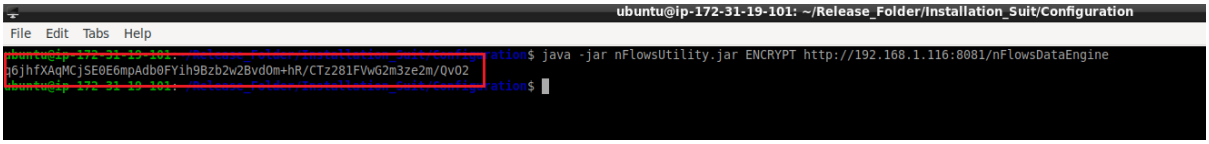
- Once the above configurations have been set save the script file.
- Open the Terminal in current folder. Execute the below command.

```
bash GenerateConfigFile.sh
```

- Once the script file is executed you will find a file named **AppConfig.Properties** generated.
- Open Terminal in the current path.
- In the below command replace <IP_OR_DNS_SERVER> and execute it in terminal.

java -jar nFlowsUtility.jar ENCRYPT https://<IP_OR_DNS_SERVER>:8081/nFlowsDataEngine

- Copy encrypted value from the terminal which is shown as below in the screenshot.

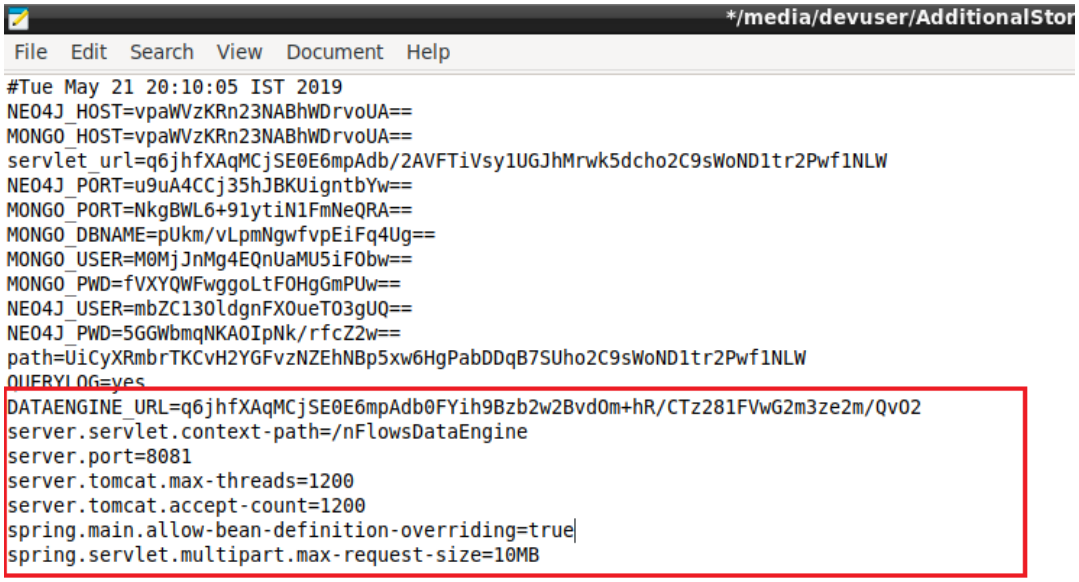


- Open AppConfig.Properties file which was present in the current folder.
- Type the below command in that file and replace <VALUE> which is copied from the terminal.

DATAENGINE_URL=<VALUE>

- And paste the below line in the same file as shown in the below screenshot.

server.servlet.context-path=/nFlowsDataEngine
server.port=8081
server.tomcat.max-threads=1200
server.tomcat.accept-count=1200
spring.main.allow-bean-definition-overriding=true
spring.servlet.multipart.max-request-size=10MB



- Save the file
- Copy this Property File and paste it inside the below path where you have mounted the nFlows Application. (refer section 4.6)

  <YOUR_APP_MOUNTED_VOLUME>/Application/nFlows/WEB-INF/classes

- Start the nFlowsAppService container using the below command

  docker container start nFlowsAppService

6. Mail Service Installation

6.1. Pre-Requisites

- Open the nFlowsMailService folder.
- You can find the below list of files in nFlowsMailService folder.

Folder Name	Folder/File Name	Type
nFlowsMailService	nFlowsMailService.jar Config nFlowsMailServiceDockerFile	File Folder File

6.2. Building Docker Image

- Open Terminal in nFlowsMailService Folder.
- Run the below command (including the last period) in terminal to build the nFlowsMailService image.

  docker build -t nflowsmailservice -f nFlowsMailServiceDockerFile .

- Once the above command is executed, please verify whether the image is built using the below command.

  docker images

- You can find that the nflowsmailservice image is built as shown in the below screenshot.

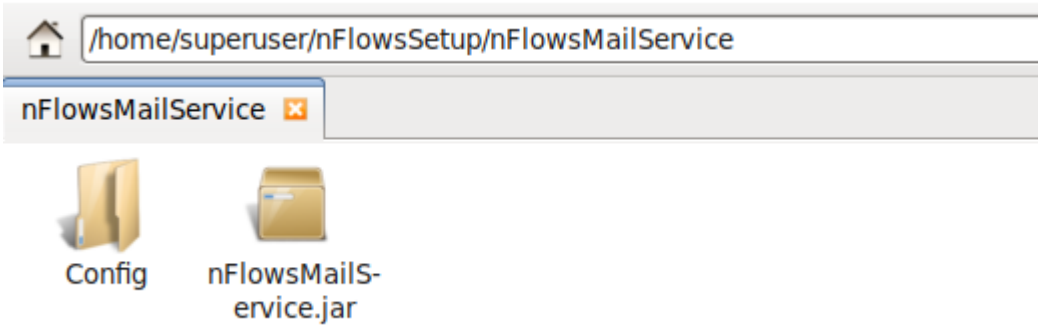
```
superuser@devsuperserv:~/nFlows_Installation_Suite_Package/nFlowsMailService$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nflowsmailservice   latest              693df029cc8b        27 seconds ago      486MB
nflows              latest              d2be4f84c7c9        About an hour ago   689MB
mongodb             latest              04b75f424692        About an hour ago   366MB
neo4j               latest              9f1bcc54ee80        2 hours ago         393MB
neo4j               3.5.29-enterprise   460ae926a2b7        7 hours ago         393MB
openjdk             15.0                bae9931e822b        10 months ago       486MB
tomcat              jdk15-openjdk       3a12135ced86        10 months ago       689MB
mongo               3.6.4               14c497d5c758        3 years ago         366MB
superuser@devsuperserv:~/nFlows_Installation_Suite_Package/nFlowsMailService$
```

6.3. Spinning up nFlowsMailService Container

- Before running the container, we can create a volume for nFlowsMailService.
- Create a folder named nFlowsMailService in your desired location.
- Copy nFlowsMailService.jar and Config folder and paste inside nFlowsMailService.
- Give full access to nFlowsMailService folder by opening the terminal in the same path and enter the below command.

  sudo chmod -R 777 .

- Copy **AppConfig.Properties** file which is generated in **Section 5.2** and paste inside Config folder.



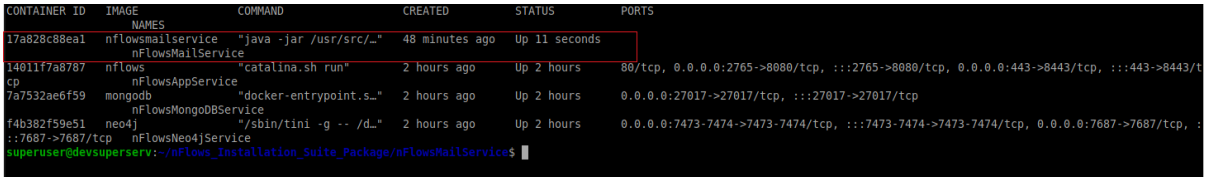
- We can use this **nFlowsMailService** folder as our volume for **nFlowsMailService Container**.
- Copy the path till nFlowsMailService folder.
- In the below command paste the above copied path in place of <MOUNTED_VOLUME> and <NETWORK_NAME> which is created in Section 4.4.

  docker run -it -d --network=<NETWORK_NAME> -v <MOUNTED_VOLUME>:/usr/src/nFlowsMailService --name nFlowsMailService nflowsmailservice

- Run the above command in terminal to create nFlowsMailService container.
- Verify whether the container is up by running the below command

  docker container ls

- You can find that container has been running with the container name **nFlowsMailService.**



7. DataEngine Service Deployment

7.1. Pre-Requisites

- Navigate to **nFlowsDataEngine** folder.
- You can find the below list of folder/files in **nFlowsDataEngine** Folder.

Folder Name	Folder/File Name
nFlowsDataEngine	nFlowsDataEngine.war

7.2. Mounted Volume Creation

- Copy **nFlowsDataEngine** Folder and paste it in any desired server location.
- Use this **nFlowsDataEngine** folder as **mounted volume for nFlows Data Engine container**.

7.3. Spinning up nFlowsDataEngine Container

- Give full access to **nFlowsDataEngine** folder which was created using Section 7.2 by opening the terminal in the same path and enter the below command.

```
sudo chmod -R 777 .
```

- Copy the path till nFlowsDataEngine folder.
- If nFlowsDataEngine service is deploying in different server then follow the steps mentioned in section 4.2 to section 4.4 to create nFlows Image.
- In the below command paste the above copied path in place of <MOUNTED_VOLUME> and provide the network name and nFlows image name in place of <NETWORK_NAME> and <nFlows_IMAGE_NAME> respectively.

```
docker run -it -d -p 8081:8080 --network=<NETWORK_NAME> -v
<MOUNTED_VOLUME>:/usr/local/tomcat/webapps --name
nFlowsR1DataLoadEngine <nFlows_IMAGE_NAME>
```

- Run the above command in terminal to create **nFlowsR1DataLoadEngine** container.
- Verify whether the container is up by running the below command

docker container ls

- You can find that nFlowsR1DataLoadEngine has been running with the container name **nFlowsR1DataLoadEngine.**
- Now stop the container with the below command

docker container stop nFlowsR1DataLoadEngine

- Give full access to **nFlowsDataEngine** folder which was created using Section 7.2 by opening the terminal in the same path and enter the below command.

```
sudo chmod -R 777 .
```

- Copy **AppConfig.Properties** file which is generated in **Section 5.2.**

- Paste AppConfig.Properties file into the below path of nFlowsDataEngine **<MOUNTED_VOLUME>/nFlowsDataEngine/WEB-INF/classes** folder.
- And rename the file **AppConfig.Properties** into **application.properties**
- Now start the container using below command.

  docker container start nFlowsR1DataLoadEngine

8. Installation Verification

- Make sure all the containers are up.
- Enter the below URL in your browser to login into the application.

  For HTTP:

  http://<IP OR HOST_NAME>/nFlows/login/login.jsp

  For HTTPS:

  https://<IP OR HOST_NAME>/nFlows/login/login.jsp