



# Major Project: Simulation of SIP with Automation

[JENKINS DOCUMENTATION]

HAFAH HUSSAIN AHAMED

## Table of Contents


<b>Installation of Jenkins.....</b>	<b>2</b>
<b>Set up of Jenkins .....</b>	<b>2</b>
<b>Creation of Pipeline.....</b>	<b>3</b>
<b>How was it implemented?.....</b>	<b>5</b>
1. <i>Dockerize the MEAN Stack Application .....</i>	5
i. <i>About Docker: .....</i>	5
ii. <i>To dockerize the Angular application: .....</i>	5
iii. <i>To dockerize the express server api backend:.....</i>	6
iiii. <i>Dockerize whole application with Docker-compose: .....</i>	6
2. <i>Dockerize the Python Application .....</i>	7
i. <i>Create python folder .....</i>	7
<b>3.     <i>Implementing Build Failure/Success using MailGun .....</i></b>	<b>7</b>

## Installation of Jenkins


### 1) Windows


Go to : <https://www.jenkins.io/download/>

Just Click Windows to download and then install

**Jenkins**  [Blog](#) [Success Stories](#) [Documentation](#)

2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section.
4. You may also want to verify the package you downloaded. [Learn more about verifying](#)

 **Download Jenkins 2.387.3 LTS for:**

Generic Java package (.war) SHA-256: f40374910de94c9c1aafbd0fd190156e7f6afad6dc1534e1b55d20e125156be5 
Docker
Kubernetes
Ubuntu/Debian
Red Hat/Fedora/Alma/Rocky/CentOS
Windows

### 2) MacOS

Go to: <https://www.jenkins.io/download/>

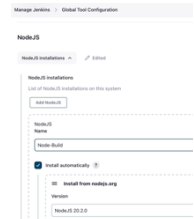
Just click MacOS, use the commands in the homebrew terminal to install jenkins.

## Set up of Jenkins

- 1) Follow the guidelines to adding initialadminpassword based on directory
- 2) Set up the Jenkins user and configure jenkins url as the default
- 3) Download and install the suggested plugins
- 4) Manage Jenkins -> Manage Plugins -> NodeJs Plugin, Docker Plugin, Docker-Compose Plugin, Blue Ocean Plugin.

## Creation of Pipeline

- 1) Create New Item as type as pipeline
- 2) If you are using windows, use the pipeline from SCM and connect to the Demo Branch, where there is a JenkinsFile.
- 3) If you are using macOS, use the pipeline syntax instead, and copy and paste from github and change the bat commands to shell commands
- 4) Comment the parts on DockerRegistry {} (Still working on it as due to Jenkins Upgrades – someparts need to change)
- 5) Configurations needed :
  - Node Js Installation Name to use in the Pipeline Syntax
    - Name: Node-Build
    - Version: Latest Version / Default
  - Go to Configure System -> Extended Email Notification
    - Under Default Recipients – Input an email that is not a gmail account
    - (Gmail does not have this feature : Less Secure application access must be ON in your account)
  - Create a docker credential using username & password



## Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	credentials	***** (docker credentials)

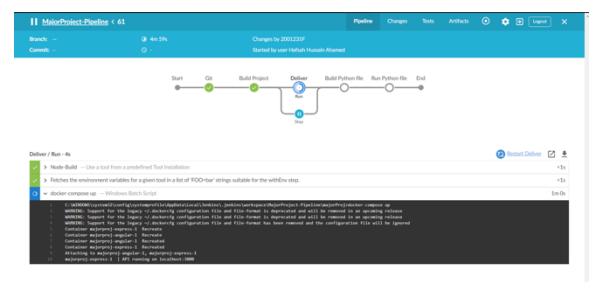
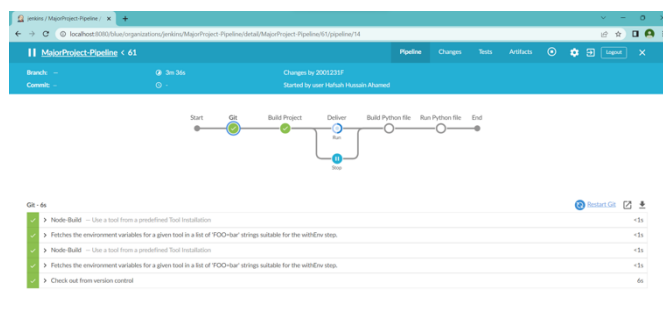
- 6) Download Docker Desktop & Create an account, Make sure the docker is running in the background before running the pipeline

**Note: If Docker Application is not open in the background, it will cause the jenkins pipeline to error**

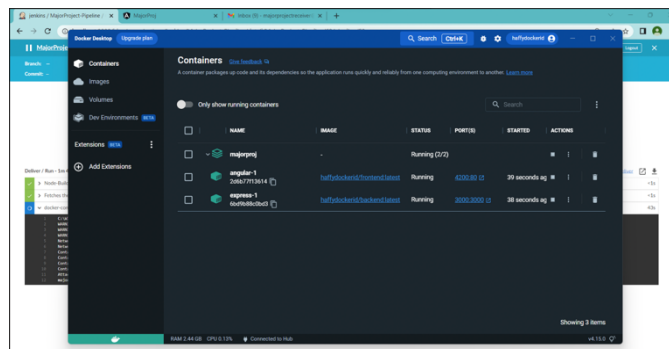
## ⊗ Console Output

```
+ docker-compose build
no valid drivers found: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

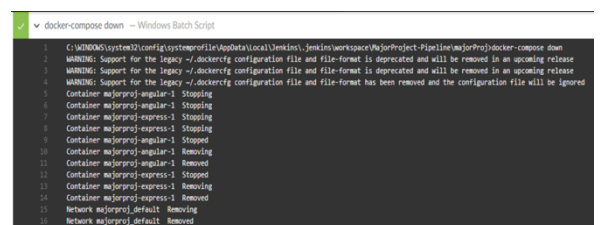
- 7) Run the Pipeline -> Open Blue Ocean



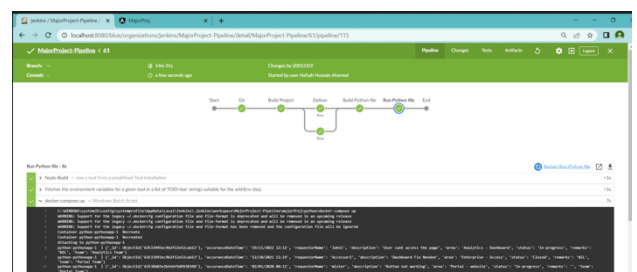
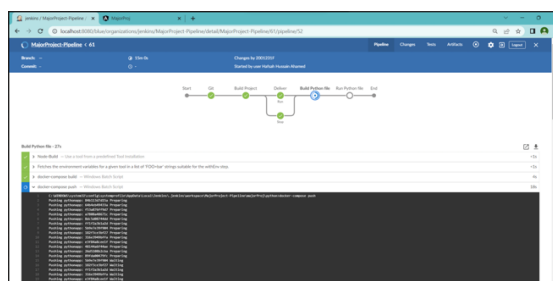
8) Open the Docker Desktop and you can see that the containers are running:



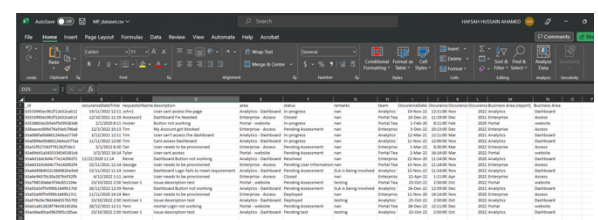
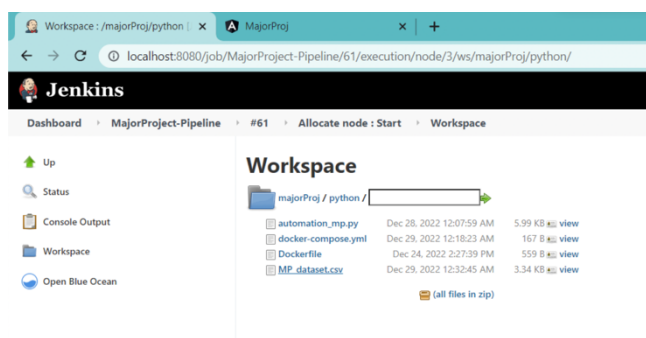
9) Abort the server once interacting with user management system (<http://localhost:4200>). Once aborting the server -> it will cause the docker to stop the running containers and remove them.



10) Build & Run the python file to clean the dataset and make it consistent to prepare for visualizations



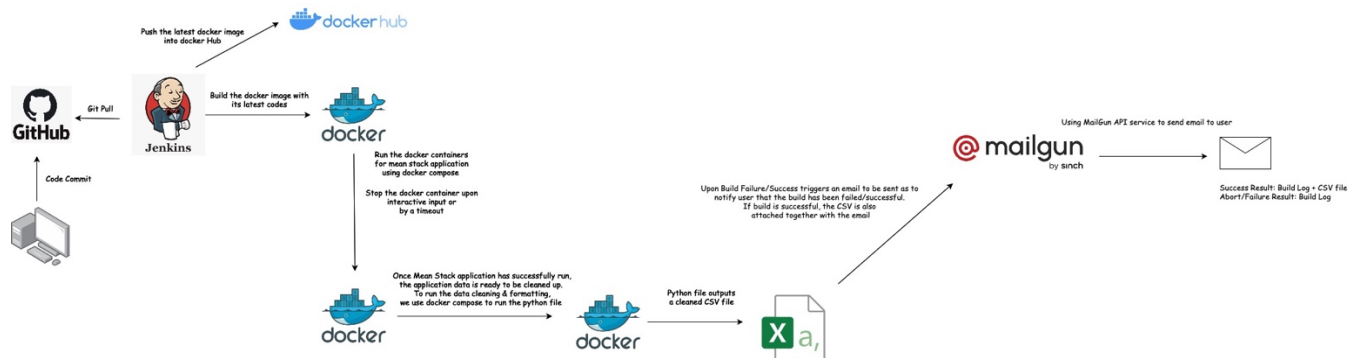
11) It automatically saves the csv file as an artifact into the workspace in jenkins.



----- Once the dataset is ready, you can get the tableau ready for visualization. -----

## How was it implemented?

- Why was docker used? And the purpose for it?



### 1. Dockerize the MEAN Stack Application

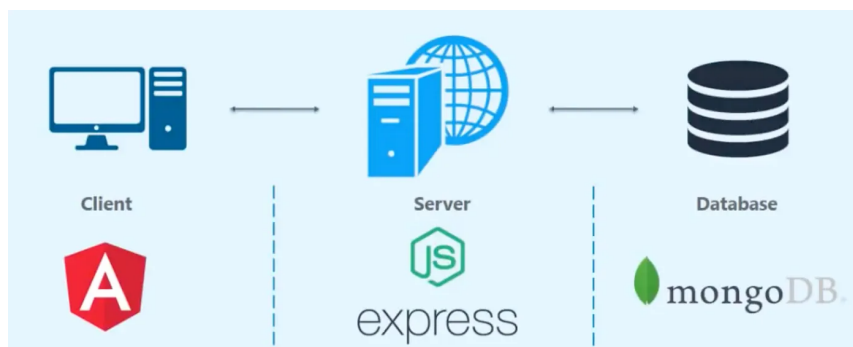
#### i. About Docker:

Docker allows to run application inside containers.

- These containers contains and stores all the information it needs to run that application e.g code, runtime, environments of the application, dependencies etc.
- Since the containers is package with all the environment, anyone who needs to run the application, will not have the trouble to install anything except the docker desktop application. Once that is installed, everything is taken care of.

How I containerized the application?

- Build an angular app in one container and point it to an expressJs API in another container.



#### ii. To dockerize the Angular application:

##### 1. A DockerFile is needed

- o A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image

Inside the docker image, the commands we will use are:

- 1) Node JS Env
- 2) Dependencies inside the package.json file (includes all needed for the app e.g angular cli..]

- 3) Copy all the files in directory except the node module & dist folders.
- 4) Run the angular project in production mode
- 5) For Routing purposes: Copy the dist folder content from the WORKDIR to the nginx html

By doing number 4, allows the project work to run in production mode deployed in nginx server(localhost:80)

*iii. To dockerize the express server api backend:*

1. A package.json file is needed
2. A dockerfile is needed

Inside the package.json,

1. We will need to include all the dependencies needed in the api.js file/server.js file.  
E.g.: Axios , bcrypt, body-parser, bootstrap, chromedriver, jsonwebtoken, express...
2. Add in the scripts to run for build and start
3. Add the name as server

Inside the docker image, the commands we will use are:

1. Node JS Env
2. Install Dependencies form the package.json file
3. Copy all the files except the files requested in the dockerignore file
4. Expose port as 3000
5. Start Server with the command line node server.js

Note at this point, there are 2 containers that is needed to run. And for each container to run, it involves 2 commands for each to build and run the container. Hence to run the whole applications efficiently and easily, docker-compose will be better.

*iiii. Dockerize whole application with Docker-compose:*

Docker Compose:

- Helps to connect & run multiple containers with docker
- With Compose, use a compose file to configure your application's services
- Then using a SINGLE command, you create and start all the services based on the configurations in the file

Create a Docker-compose file in the root directory

- In this file, we need to tell which containers to build by specifying the directories of the docker file and the ports for each of the containers.

To simply run the containers based on 2 images, simply run:

To build the environment: docker-compose build

To run the application: docker-compose up

Problem/Disadvantage of using this:

Anytime we make changes to either the angular app/the express app, we need to run docker-compose build and then docker-compose up to run the application -> **in a way it is fine for my application as Jenkins will do the building and running for us.**

If you don't have the codes and you want to simply run the application from the dockerhub then follow the below:

### Download Desktop Docker Application

Use the commandline/terminal to run the following commands:

```
docker pull haffydockerid/frontend:latest
```

```
docker pull haffydockerid/backend:latest
```

If you would like to specify the ports:

```
docker run -p 3000:3000 haffydockerid/backend
```

```
docker run -d haffydockerid/frontend -p 4200:80
```

else, you can just, cd to the folder that contains the docker compose file and open terminal

```
docker-compose build
```

```
docker-compose up
```

## 2. Dockerize the Python Application

### i. Create python folder

- Export the IPYNB notebook to .py file and save as automation\_mp.py file
- Dockerfile is needed
- Docker compose file is needed

Inside the docker image:

1. Python Env
2. Copy the automation\_mp.py file into the container directory
3. Install the packages needed for the python py file to run
4. Run the python file

- The reason behind why the docker compose file is needed is because this python file outputs a csv file that needs to be stored in the current directory hence volume in the docker-compose file

## 3. Implementing Build Failure/Success using MailGun

