Section 3 ~

Implementation:

1. To be started, make sure working host open-stack cloud instance (IP: 128.39.121.131) is running and accessible.
2. Ensure that the Jenkins is installed. Here we install Jenkins inside a docker container and updated the jenkins script to allow running docker commands from inside the container [1]. Below is the command , run as root user:

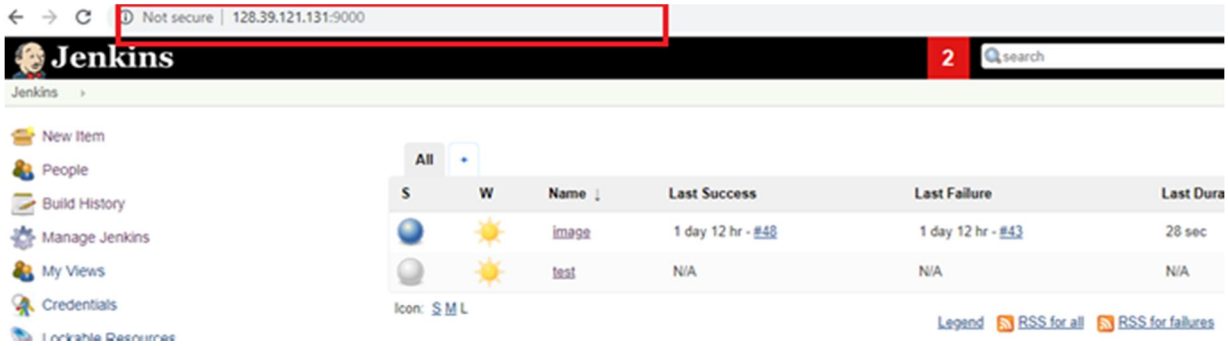
```
#Installing Docker
apt install docker.io

#Create a folder for jenkins_home folder in your root and Pull the official jenkins image from Docker repository
mkdir /var/jenkins_home

# pull jenkins latest stable version
 docker pull jenkins/jenkins:lts

# runs as user root: -u root ,
# runs docker with port binded 9000 : -p 9000:8080
# adds jenkins home : -v jenkins_home:/var/jenkins_home
# adds docker socker : -v /var/run/docker.sock:/var/run/docker.sock
# adds path to docker : -v /usr/bin/docker:/usr/bin/docker
docker run -d -u root -p 9000:8080 -v jenkins_home:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock -v /usr/bin/docker:/usr/bin/docker -v /etc/timezone:/etc/timezone -v /etc/localtime:/etc/localtime --restart always --name=jenkins jenkins/jenkins:lts

# install libraries to allow running of docker commands from inside jenkins container
# ref: https://medium.com/faun/using-docker-in-jenkins-cba6b8070756

docker exec -it -u root jenkins apt-get update && apt-get install -y libltdl7 && rm -rf /var/lib/apt/lists/*

#tests with "docker ps" - The command is successful IF see the docker containers running
docker exec -it -u root jenkins docker ps -a


#check docker ps process whether Jenkins is running or not

docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED
STATUS              PORTS                                NAMES
```

```
732f7ff085df        jenkins/jenkins:lts   "/sbin/tini -- /usr/…"   6 minutes ago
Up 6 minutes        50000/tcp, 0.0.0.0:9000->8080/tcp   Jenkins
```

show password
to see password use the tail command
sudo docker exec jenkins tail -f /var/jenkins_home/secrets/initialAdminPassword
or use cat to see the password
sudo docker exec jenkins cat /var/jenkins_home/secrets/initialAdminPassword
docker exec -it 10a9acee2944 cat /var/jenkins_home/secrets/initialAdminPassword

c22c843a77934b0ca2f74eb3c6248b13

#Login in to Jenkins GUI
Login to Jenkins server http://128.39.121.131:9000/



3. **Code a index.html webpage for website**

Run below command as root user to create index.html page which content the words “Hello World”.

```
#vi index.html
<!DOCTYPE html>
<html>
        <body>

                <h1>Hello World</h1>

        </body>
</html>
:wq!
```

4. **Maintain a Git Code Repository**

Here we use below code repository to maintain source code for HTML code and test script. Need to push the code index.html to git repository

https://git.cs.hioa.no/s340023/awesome-project.git

Git setup procedure command on local machine is given below:

Git global setup

```
git config --global user.name "Asif"
git config --global user.email s340023@hioa.no
```

Add an origin for the repository

```
git remote add origin https://git.cs.hioa.no/s340023/awesome-project.git
```

Push the code to remote repository

```
git add index.html
git commit –m “index.html”
git push –u origin master
```

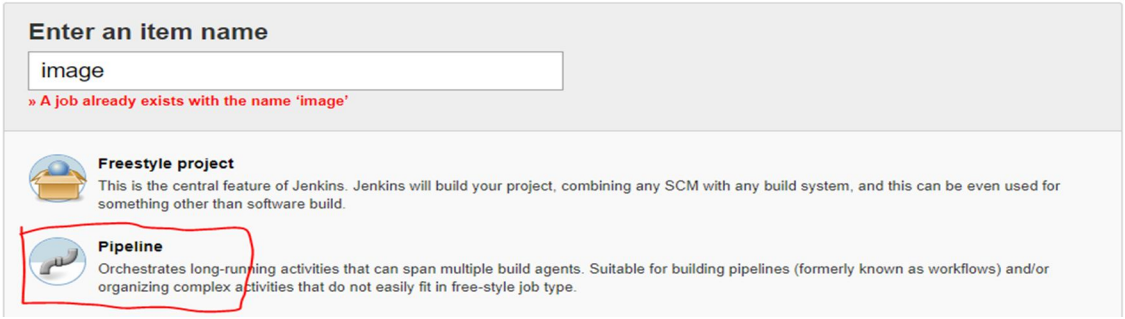
The same git repository also used to maintain code like Jenkinsfile (is a text file that contains the definition of a Jenkins Pipeline and is checked into source control).

Push the code to remote repository

```
git add .
git commit –m “Jenkinsfile_testscript”
git push –u origin master
```

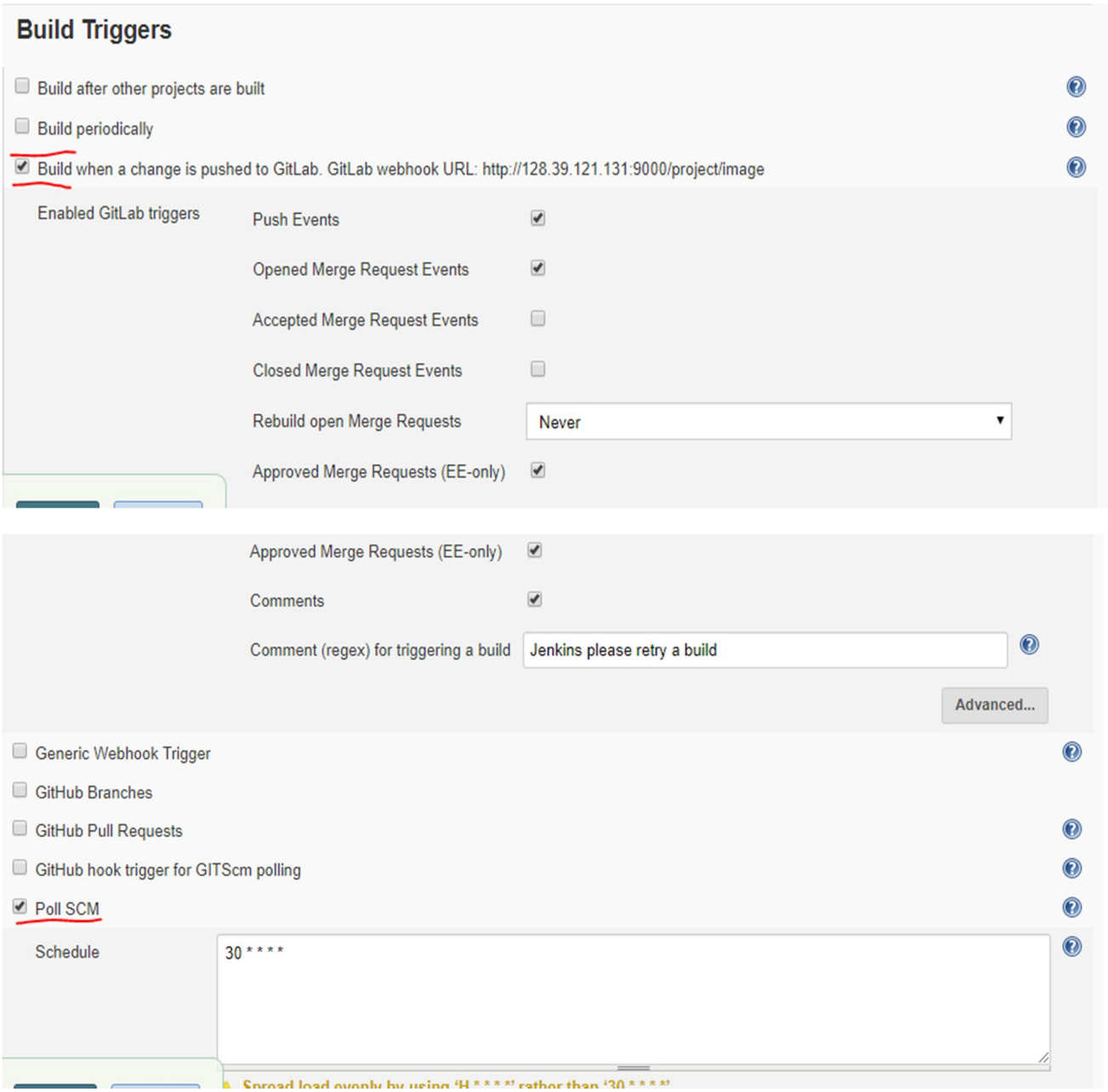
5. Configure Jenkins Pipeline project

Create a new item and select “Pipeline”



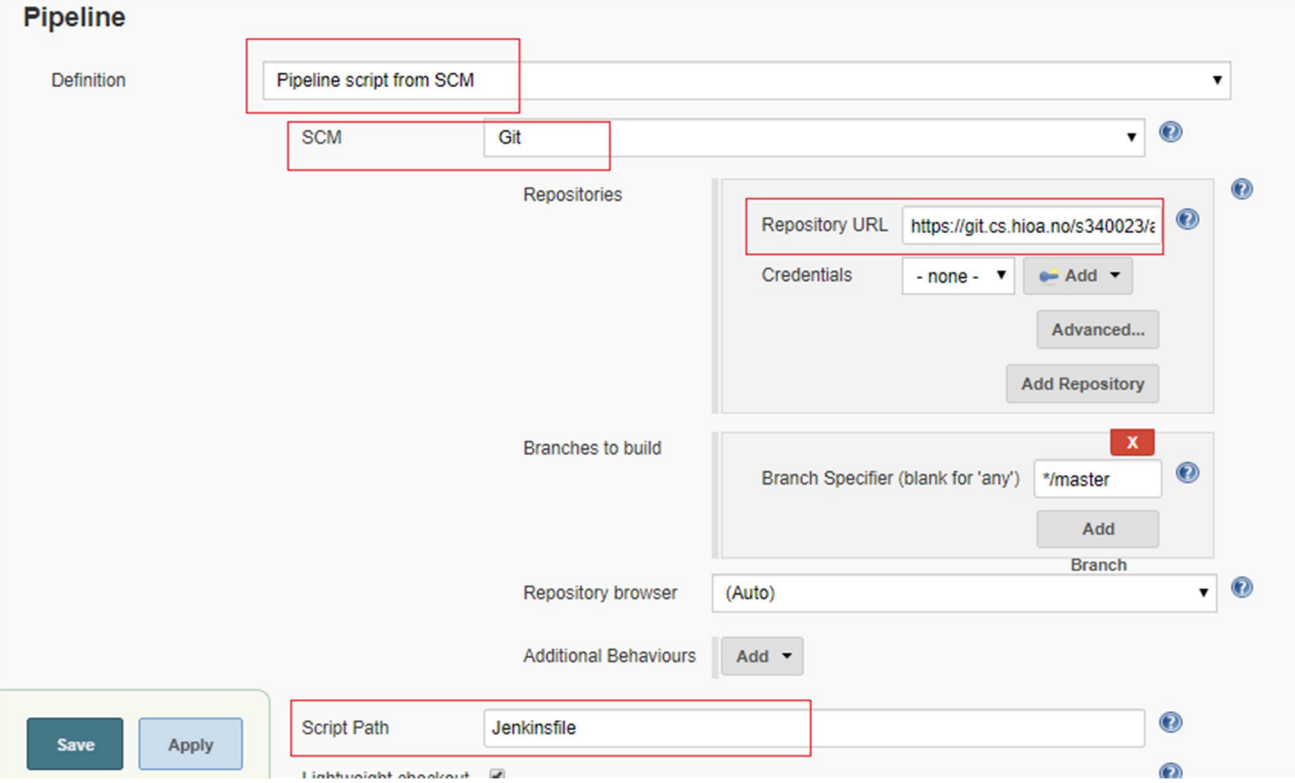
5.1 Configure build triggers

Configure the build triggers like that it get sync with git repository and can pull the source code from there if there is any change. Hence, we configured Poll SCM every 30 minutes and we configure when there is a change is pushed to GitLab it will start build. Screenshot is given below:



5.2 Loading Pipeline Scripts from SCM

Here pipeline scripts has been writing in remote SCM (remote git reposity-https://git.cs.hioa.no/s340023/awesome-project.git) system, and then loading those scripts into Jenkins using the Pipeline Script from SCM option and at remote repository script path file name is "Jenkinsfile" which is the source code control place for testing and deploying scripts. Below is the screenshot of Pipeline script from SCM configuration.



6. **Test the Jenkins Pipeline project**

As per above section Jenkins pipeline configuration, once the code (index.html) pushed at Git repository Jenkins pipeline will detect within 30 minutes and trigger the build process as per pipeline script file ""Jenkinsfile" from remote git repository as below steps**:**

6.1 First create docker file and implement latest code into it and run as web server:

Jenkins Pipeline has a built-in support for interacting with Docker from within a Jenkinsfile. In order to create a Docker image, the Docker Pipeline plugin also provides a build() method for creating a new image, from a Dockerfile in the repository, during a Pipeline run. [2]

As per below "Jenkins" file configuration it will first create a docker container by using command docker.build() and then start docker run and check whether web server is running or not.

6.1.1 Script Jenkinsfile configuration in Git repository to build an image of docker

```
node
{
def customImage = docker.build("web:v1")
sh "sleep 2"
sh "docker run -d -p 80:80 web:v1"
sh "curl http://128.39.121.131:80/"
}
```

6.1.2 Script Dockerfile configuration in same Git repository to create docker container

```
FROM ubuntu:18.04
MAINTAINER s340023@oslomet.no

RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y python

ADD index.html /var/www/html/index.html

EXPOSE 80

CMD /usr/sbin/apache2ctl -D FOREGROUND
```

6.1.3 Verify above part build result

To verify the status of a build, once the build is triggered, we can check for the console output. below is the screenshot of docker image build , docker run and test web server running status.

```
+ docker build -t web:v1 .
Sending build context to Docker daemon  223.2kB

Step 1/8 : FROM ubuntu:18.04
 ---> 2ca708c1c9cc
Step 2/8 : MAINTAINER s340023@oslomet.no
 ---> Using cache
 ---> 418038118ac1
Step 3/8 : RUN apt-get update
 ---> Using cache
 ---> aad1158e328e
Step 4/8 : RUN apt-get install -y apache2
 ---> Using cache
 ---> bb9736082da5
Step 5/8 : RUN apt-get install -y python
 ---> Using cache
 ---> eca95883f9eb
Step 6/8 : ADD index.html /var/www/html/index.html
 ---> 8583171993d4
Step 7/8 : EXPOSE 80
 ---> Running in 8c23f7c8e1c0
Removing intermediate container 8c23f7c8e1c0
 ---> 5079770d0798
Step 8/8 : CMD /usr/sbin/apache2ctl -D FOREGROUND
 ---> Running in 95ba515a65da
Removing intermediate container 95ba515a65da
 ---> 2d7178671c04
Successfully built 2d7178671c04
Successfully tagged web:v1
[Pipeline] sh
+ sleep 2
[Pipeline] sh
+ docker run -d -p 80:80 web:v1
cf580366cbc0127eb71e228365fa4a036c282d56edb1bbc335bc1a4e49cf9f74
```

```
+ curl http://128.39.121.131:80/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    74  100    74    0     0  17502      0 --:--:-- --:--:-- --:--:-- 24666
<!DOCTYPE html>
<html>
        <body>

                <h1>Hello World</h1>

        </body>
</html>
```

6.2 Second it will write the content of the webpage in a file:

As per below "Jenkins" file configuration screenshot secondly it will check particular text "Hello world" in index.html file , this step used a script "text_cut.sh" to read the content of webpage and write its content in a file, details is mentioned downward.

"Jenkins" file configuration screenshot to check text

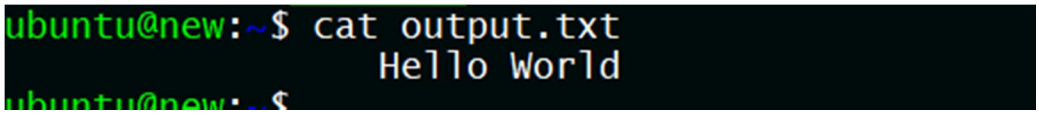
```
node
{
sh "sleep 3"
def customImage = docker.build("web:v1")
sh "sleep 2"
sh "docker run -d -p 80:80 web:v1"
sh "curl http://128.39.121.131:80/"
sh "sleep 2"
sh "chmod 777 text_*.sh"
sh "./text_cut.sh"
sh "sleep 2"
sh "./text_compare_email.py"
}
```

6.2.2 Script "text_cut.sh" script file to check web page content and write that information

Here the script first curl the website then grep the words "Hello world" , if found then it write the entire content of the code which lies in between html tag <h1> & </h1> and write this information to a file called "output.txt"

```
#!/bin/sh
curl http://128.39.121.131:80 | grep -v grep |grep –i "Hello World" | awk -F"<h1>"
'{print$1,$2}' | awk -F"</h1>" '{print$1}' > output.txt
```

screenshot of 'output.txt' file is given below :



6.2.3 Verify the above part build result

Below screens shot is the build console output for script to write webpage content

```
+ ./text_cut.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    74  100    74    0     0  32244      0 --:--:-- --:--:-- --:--:-- 37000
[Pipeline] sh
+ sleep 2
```

6.3 Third it will test the code output and decide whether test success or fail

6.3.1 As per below "Jenkins" file configuration screenshot thirdly it will test the output of the webpge , this step used a python script "text_compare_email.py" to read the content of webpage and decide test success or not and take action accordingly. Details mention downward.

```
node
{
checkout scm
sh "sleep 3"
def customImage = docker.build("web:v1")
sh "sleep 2"
sh "docker run -d -p 80:80 web:v1"
sh "curl http://128.39.121.131:80/"
sh "sleep 2"
sh "chmod 777 text_*.sh"
sh "./text_cut.sh"
sh "sleep 2"
sh "./text_compare_email.py"
}
```

6.3.2 Script "text_compare_email.py" script file explanation

Below is the explanations of the script file and source code given respectively:

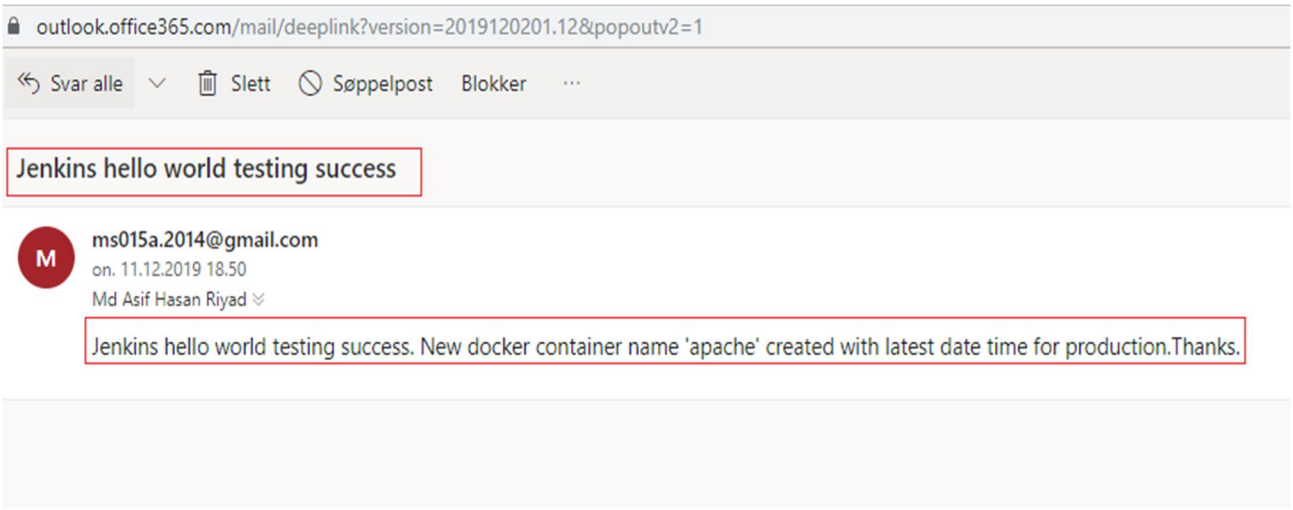
a. At first import SMTPlib & Popen module where The smtplib module defines an SMTP client session object that can be used to send mail. The smtplib modules is useful for communicating with mail servers to send mail . Besides, 'Popen' is Python's subprocess module is one of useful modules in the standard library for system administration to run command in linux

b. Second define the desired word ‘Hello world’ which will be compared with the output of webpage written in ‘output.txt’ file
c. Open the output.txt file as read mode which stores the output of webpage and do a iteration over the file , remove white space before & after the output and save the output in list and close the file
d. run a condition where if list not match with the word ‘Hello world’ then it compose a email and send email to user by mention testing failed
e. else if list output match with words ‘Hello World’ then build a new container from same Git repository from where test webserver was installed to ensure production webserver have latest code. Finally curl the new webpage and check webserver is running properly or not. Here, the new container name is ‘apache’ and the same time an email also sent to user regarding test success and deployment

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
""" The smtplib module defines an SMTP client session object that can be used to send mail. The smtplib modules is useful for communicating with mail servers to send mail"""

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.application import MIMEApplication
import datetime

""" Popen is a Python's subprocess module is one of useful modules in the standard library for system administration to run command in linux """
from subprocess import Popen

import time

""" Word which will be compared with webpage output """
comp="Hello World"

""" Open the output.txt file as read mode  which stores the output of webpage and do a iteration over the file , remove white space before & after the output and save the output in list and close the file """
f=open('output.txt','r')
for i in f:
  a=i.strip()
f.close()

""" run a condition where if list output not match with the word ‘Hello world’ then it compose a email and send email to user  by mention testing failed   """
if (a != comp):
 fromaddr = "ms015a.2014@gmail.com"
 toaddr = "s340023@oslomet.no"
 msg = MIMEMultipart()
 msg['From'] = fromaddr
 msg['To'] = toaddr
```

```
 msg['Subject'] = "Jenkins hello world testing failed.Plz check.Thanks"
 body = "Jenkins hello world testing failed.Plz check.Thanks"
 msg.attach(MIMEText(body))
 server = smtplib.SMTP('smtp.gmail.com', 587)
 server.ehlo()
 server.starttls()
 server.ehlo()
 server.login("ms015a.2014@gmail.com", "password")
 text = msg.as_string()
 server.sendmail(fromaddr, toaddr, text)

""" else if output match with words 'Hello World' then build a new container from
same repository from where test webserver was installed to ensure production
webserver have latest code. Finally curl the new webpage and check webserver is
running properly or not. Here, the new container name is 'apache' and the same time
an email also sent to user regarding test succss and deployment """

else :
   p1=Popen (['docker build -t apache https://git.cs.hioa.no/s340023/awesome-
project.git'],shell=True)
   time.sleep(1)
   p2=Popen (['docker run -d -p 80:80 apache'],shell=True)
   time.sleep(1)
   p4=Popen (['curl http://128.39.121.131:80/'],shell=True)
   fromaddr = "ms015a.2014@gmail.com"
   toaddr = "s340023@oslomet.no"
   msg = MIMEMultipart()
   msg['From'] = fromaddr
   msg['To'] = toaddr
   msg['Subject'] = "Jenkins hello world testing success"
   body = "Jenkins hello world testing success. New docker container name 'apache'
created with latest date time for production.Thanks."
   msg.attach(MIMEText(body))
   server = smtplib.SMTP('smtp.gmail.com', 587)
   server.ehlo()
   server.starttls()
   server.ehlo()
   server.login("ms015a.2014@gmail.com", "password")
   text = msg.as_string()
   server.sendmail(fromaddr, toaddr, text)
```

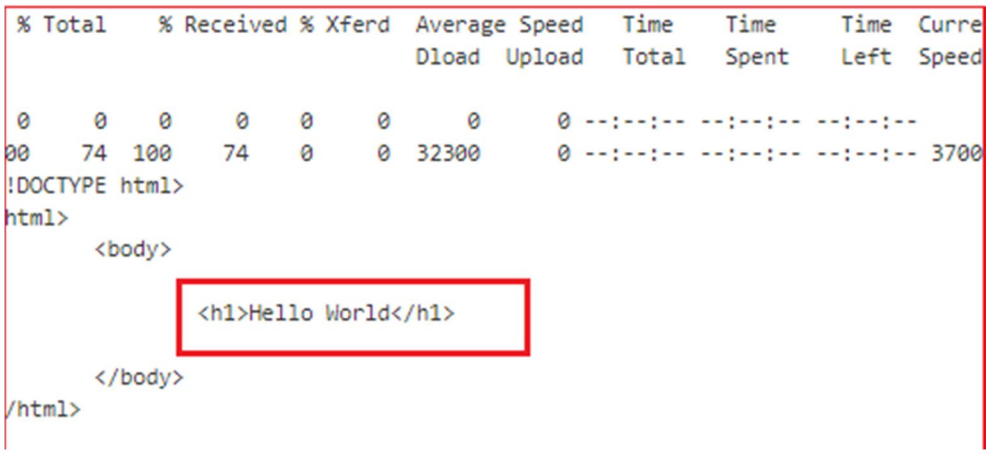
Below is the screenshot of email notification regarding test success:



6.3.3 Verify the test build result

Below screens shot is the build console output for script to perform the test and deployment

```
+ ./text_compare_email.py
Sending build context to Docker daemon   72.7kB
Step 1/8 : FROM ubuntu:18.04
 ---> 2ca708c1c9cc
Step 2/8 : MAINTAINER s340023@oslomet.no
 ---> Using cache
 ---> 418038118ac1
Step 3/8 : RUN apt-get update
 ---> Using cache
 ---> aad1158e328e
Step 4/8 : RUN apt-get install -y apache2
 ---> Using cache
 ---> bb9736082da5
Step 5/8 : RUN apt-get install -y python
 ---> Using cache
 ---> eca95883f9eb
Step 6/8 : ADD index.html /var/www/html/index.html
 ---> Using cache
 ---> 4a6276e9a0bf
Step 7/8 : EXPOSE 80
 ---> Using cache
 ---> 4798ac54d2e5
Step 8/8 : CMD /usr/sbin/apache2ctl -D FOREGROUND
 ---> Using cache
 ---> 90cd005b6f6d
Successfully built 90cd005b6f6d
Successfully tagged apache:latest
```



6.3.4 Action if test failed

Now a test failed scenario will be demonstrated here. In this case, below new index.html code has been pushed to Git repository where it can see that there are two extra special character "<< >>" and a extra word 'Hello' lies instead of just word 'Hello World'

```
<!DOCTYPE html>
<html>
        <body>
                <h1><<Hello World>><Hello></h1>
        </body>
</html>
```

In this case during build process when curl tested webserver then as a output "<<Hello World>><Hello>" saved in 'output.txt' file as showing below screenshots respectively . So, during comparison python script found discrepancy with expected word and found word and testing became failed.

```
[Pipeline] sh
+ curl http://128.39.121.131:80/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100    86  100    86    0     0  33230      0 --:--:-- --:--:-- --:--:-- 43000
<!DOCTYPE html>
<html>
        <body>

                <h1><<Hello World>></Hello></h1>

        </body>
</html>
```

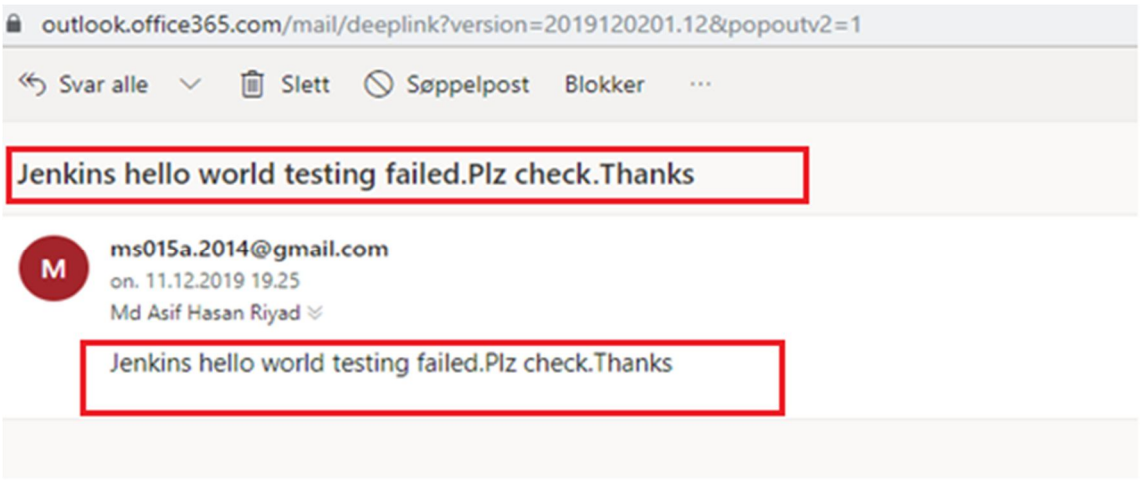
```
ubuntu@new:~$ cat output.txt
<<Hello World>></Hello>
ubuntu@new:~$
```

According to above section explanation and below partial source code; if test failed i.e webpage output content is not match with desired word “Hello World” then an email notification will send to user.

```
""" Word which will be compared with webpage output """
comp="Hello World"
```

```
""" run a condition where if list output not match with the word 'Hello world' then
it compose a email and send email to user  by mention testing failed   """
if (a != comp):
 fromaddr = "ms015a.2014@gmail.com"
 toaddr = "s340023@oslomet.no"
 msg = MIMEMultipart()
 msg['From'] = fromaddr
 msg['To'] = toaddr
 msg['Subject'] = "Jenkins hello world testing failed.Plz check.Thanks"
 body = "Jenkins hello world testing failed.Plz check.Thanks"
 msg.attach(MIMEText(body))
 server = smtplib.SMTP('smtp.gmail.com', 587)
 server.ehlo()
 server.starttls()
 server.ehlo()
 server.login("ms015a.2014@gmail.com", "password")
 text = msg.as_string()
 server.sendmail(fromaddr, toaddr, text)
```

Below is the screen shot of email testing notification regarding test failed:



Section 4 ~

Results

Jenkins triggers build and compilation processes automatically, and notify if something went wrong with the HTML code. So these tests automatically get executed. Besides, Jenkins automatically deploying code to production as a Docker container, if the test is green. This way we can achieve faster build execution, early detection of bugs, increased code quality.

Evaluation

By doing above test it observed that Jenkins triggers build and compilation processes automatically, and notify if something went wrong with the HTML source code. So these tests automatically get executed. Besides, Jenkins automatically deploying code to production as a Docker container, if the test is green. This way we can achieve faster build execution, early detection of bugs, increased code quality. But there are some optimization could be done to ensure more product quality:

1. Further work can be done to find a way to reuse port 80 or other workaround thus a fixed port can be use for website browse during multiple test cases.
2. Future work can be done during test web server creation time when need to implement dynamic method to create container tag & version if multiple test case need to run parallel.
3. Instead of sending email notification, send notification to company instant messaging team or any other group where team person can collaborate with each other regarding test result online.