Network Management and Automation

Lab 9

CI/CD with Jenkins

University of Colorado Boulder

Network Engineering Program

Professor Levi Perigo, Ph.D.

Summary

As the network scale increases, so does the complexity of the code that manages, monitors, and configures the network. The CI/CD tools make it easier to continuously integrate and deploy Infrastructure-as-Code (IaC) in a cost-efficient manner, with higher product quality, and shorter release cycles. As the name suggests, one of the benefits of CI/CD tools is the Continuous Deployment of networks. The purpose of this lab is to understand the DevOps cycle by learning Jenkins, integrating it with GitHub, and configuring Jenkins Pipeline to continuously deploy networks via code. The other purpose of this lab is to use unit tests to make sure your code is meeting the pre-defined requirements, which in turn, indicates your code is ready for production. The objectives of this lab are to be used as guidelines, and additional exploration by the student is strongly encouraged.

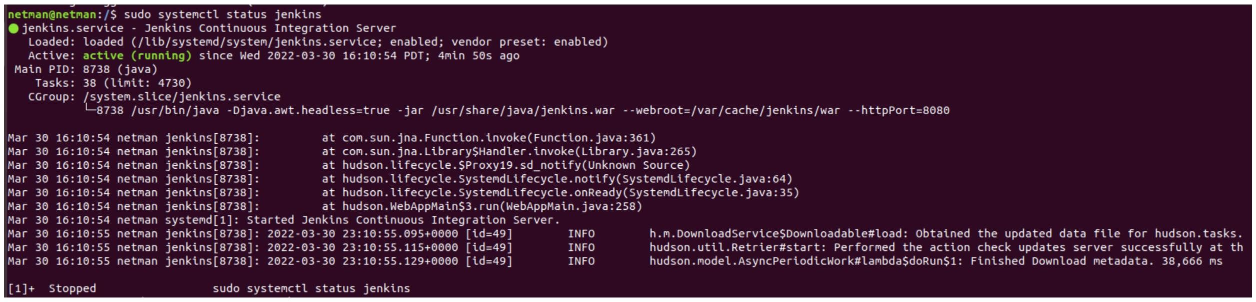
Objectives

1. Learn about Jenkins functionalities
2. Learn how to use version control (GitHub)
3. Learn how Jenkins and GitHub work together
4. Learn how to write Jenkins Pipeline
5. Learn how to write a clean and readable code that follows the PEP8 style
6. Learn how to write Unit Tests
7. Learn how to run Jenkins jobs (both push and pull methods)

Objective 1 – Getting started

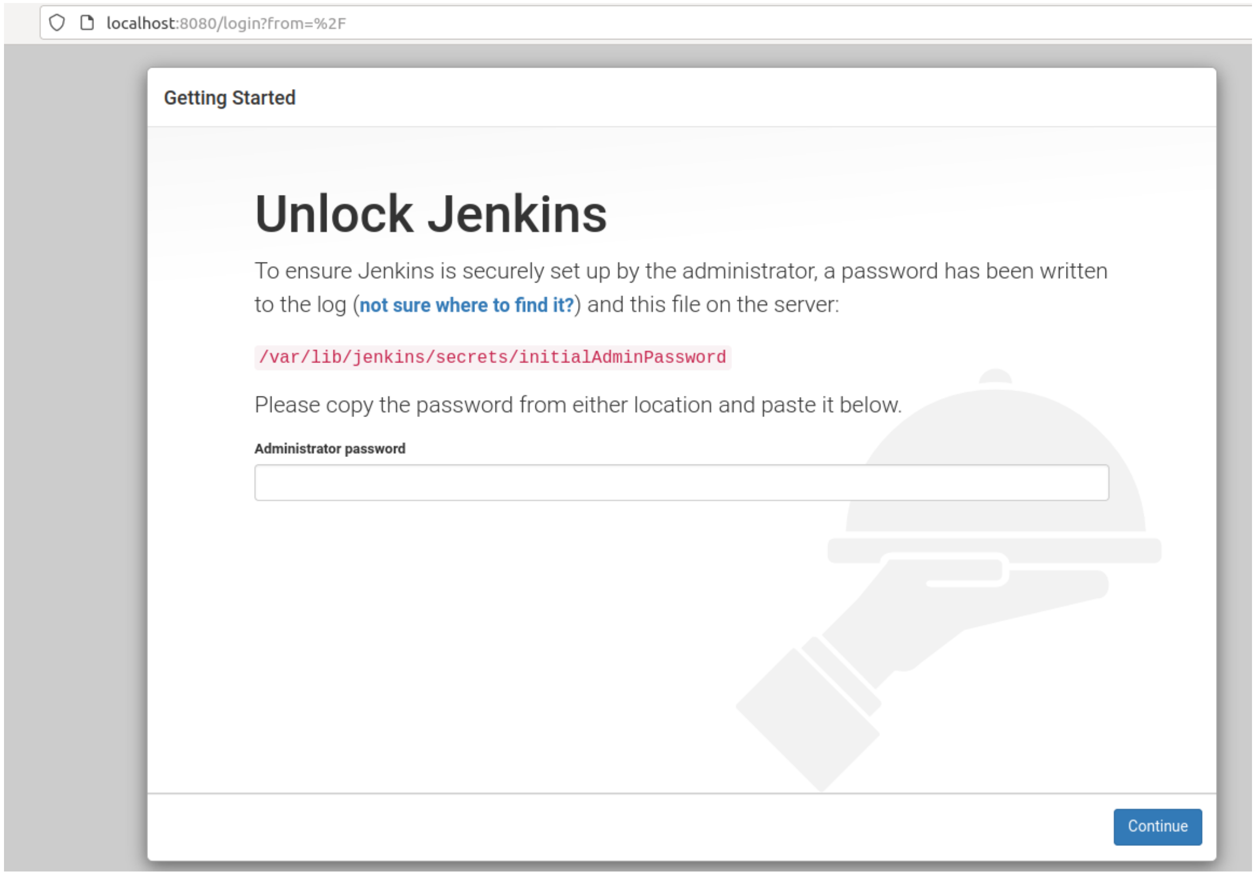
1. Install Jenkins on the NetMan VM or on your local system. Just make sure you have connectivity to GNS3 devices. Provide relevant screenshots of Jenkins status and Jenkins console. **[10 points]**

   Status:



```
netman@netman:/$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-03-30 16:10:54 PDT; 4min 50s ago
 Main PID: 8738 (java)
    Tasks: 38 (limit: 4730)
   CGroup: /system.slice/jenkins.service
           └─8738 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Mar 30 16:10:54 netman jenkins[8738]:         at com.sun.jna.Function.invoke(Function.java:361)
Mar 30 16:10:54 netman jenkins[8738]:         at com.sun.jna.Library$Handler.invoke(Library.java:265)
Mar 30 16:10:54 netman jenkins[8738]:         at hudson.lifecycle.$Proxy19.sd_notify(Unknown Source)
Mar 30 16:10:54 netman jenkins[8738]:         at hudson.lifecycle.SystemdLifecycle.notify(SystemdLifecycle.java:64)
Mar 30 16:10:54 netman jenkins[8738]:         at hudson.lifecycle.SystemdLifecycle.onReady(SystemdLifecycle.java:35)
Mar 30 16:10:54 netman jenkins[8738]:         at hudson.WebAppMain$3.run(WebAppMain.java:258)
Mar 30 16:10:54 netman systemd[1]: Started Jenkins Continuous Integration Server.
Mar 30 16:10:55 netman jenkins[8738]: 2022-03-30 23:10:55.095+0000 [id=49]        INFO        h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.
Mar 30 16:10:55 netman jenkins[8738]: 2022-03-30 23:10:55.115+0000 [id=49]        INFO        hudson.util.Retrier#start: Performed the action check updates server successfully at th
Mar 30 16:10:55 netman jenkins[8738]: 2022-03-30 23:10:55.129+0000 [id=49]        INFO        hudson.model.AsyncPeriodicWork#lambda$doRun$1: Finished Download metadata. 38,666 ms

[1]+  Stopped                 sudo systemctl status jenkins
```

   Console:



localhost:8080/login?from=%2F

Getting Started

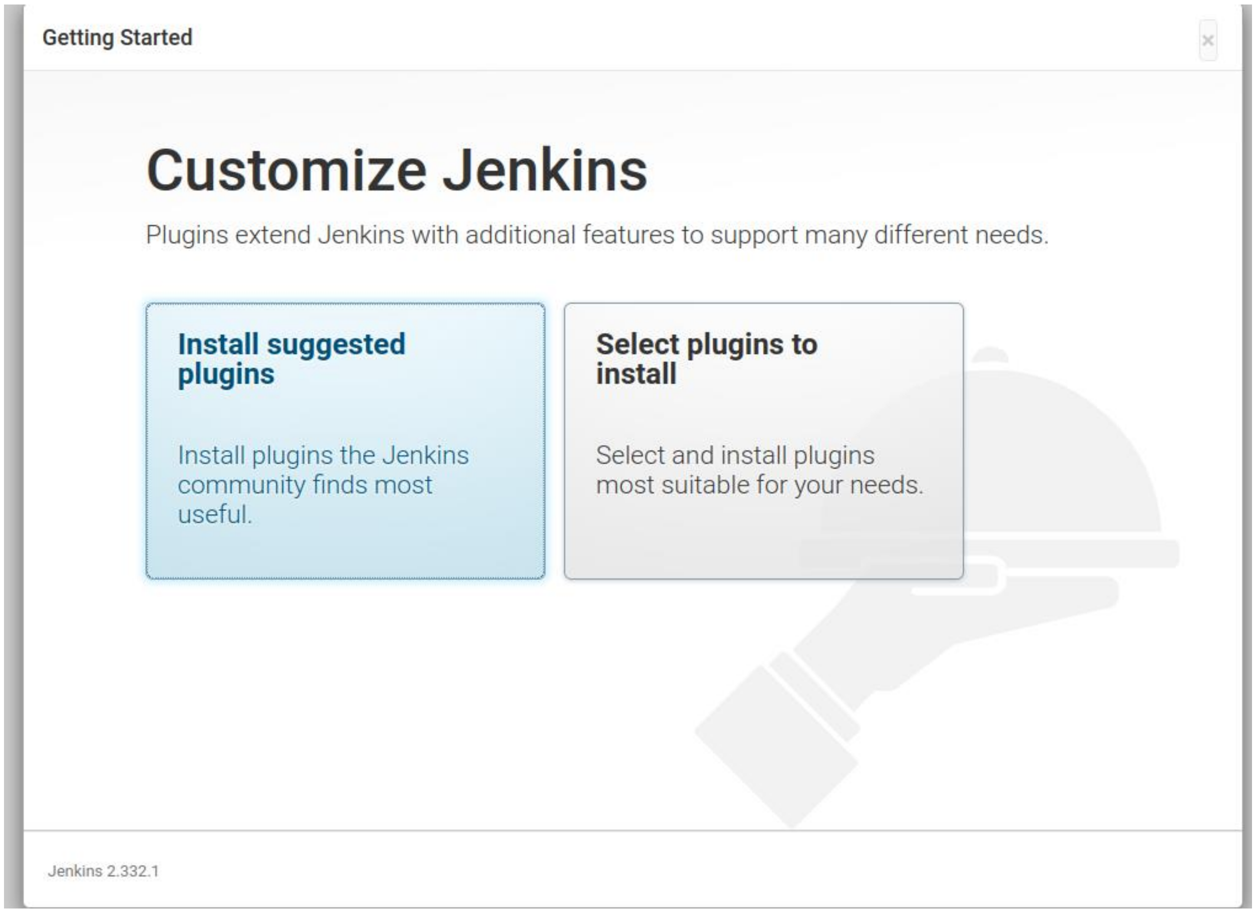
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password


Getting Started

Customize Jenkins

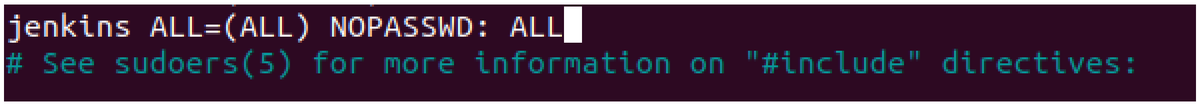
Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.332.1

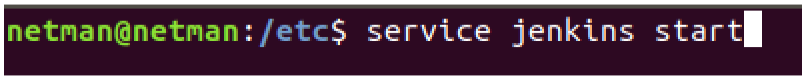
2. Provide Jenkins super-user permission to avoid using 'sudo' for the upcoming objectives.

   Provide a screenshot. **[5 points]**

   I made changes to the /etc/sudoers file and added below line:

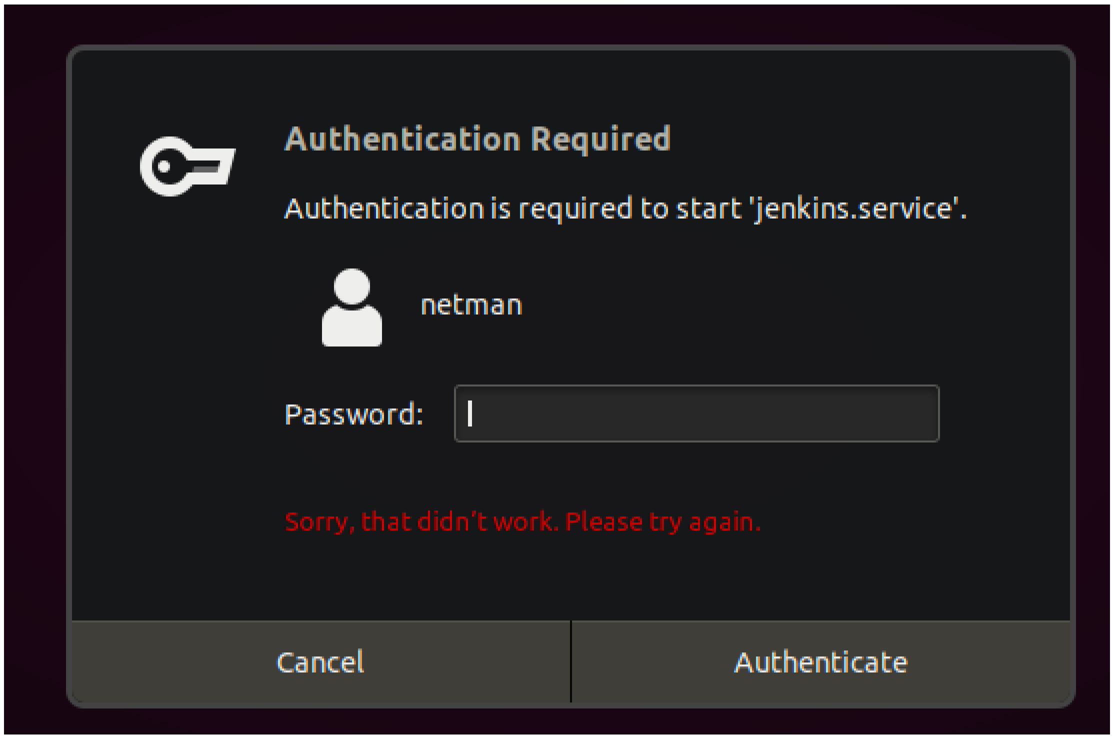


```
jenkins ALL=(ALL) NOPASSWD: ALL
# See sudoers(5) for more information on "#include" directives:
```



```
netman@netman:/etc$ service jenkins start
```

   Without using sudo I was getting the screen:



Authentication Required

Authentication is required to start 'jenkins.service'.

netman

Password:

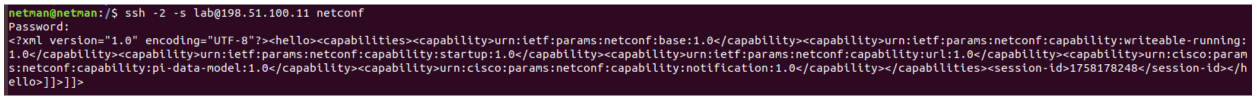
Sorry, that didn't work. Please try again.

Cancel

Authenticate

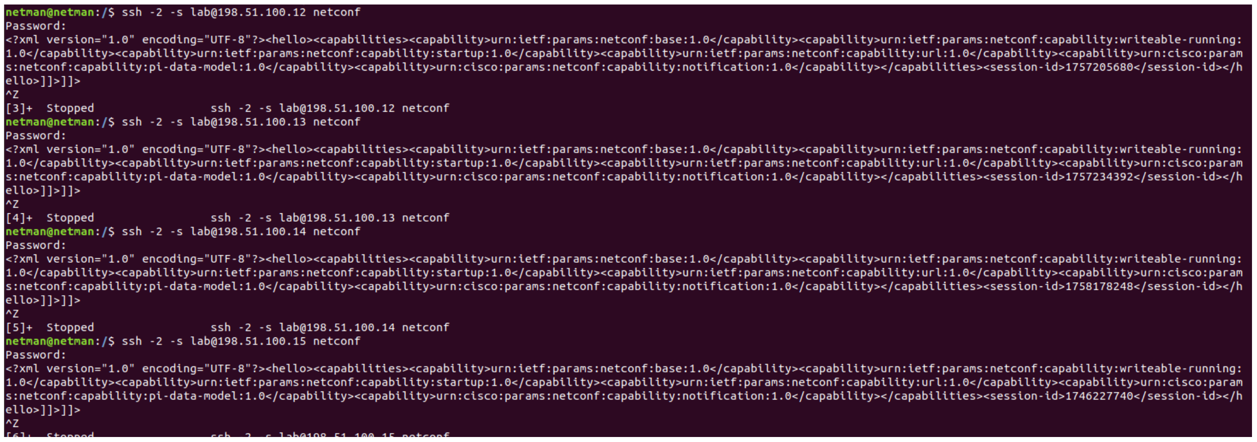
3. Create a GNS3 topology provided for objective 2 in the previous Netconf lab. Configure Management IP addresses and NETCONF over SSH on all routers for the upcoming objectives. Paste relevant screenshots **[10 points]**

   Netconf over SSH on R1:



```
netman@netman:/$ ssh -2 -s lab@198.51.100.11 netconf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello><capabilities><capability>urn:ietf:params:netconf:base:1.0</capability><capability>urn:ietf:params:netconf:capability:writeable-running:
1.0</capability><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><capability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:cisco:param
s:netconf:capability:pi-data-model:1.0</capability><capability>urn:cisco:params:netconf:capability:notification:1.0</capability></capabilities><session-id>1758178248</session-id></h
ello>]]>]]>
```

   Netconf over SSH on R2, R3, R4, and R5:



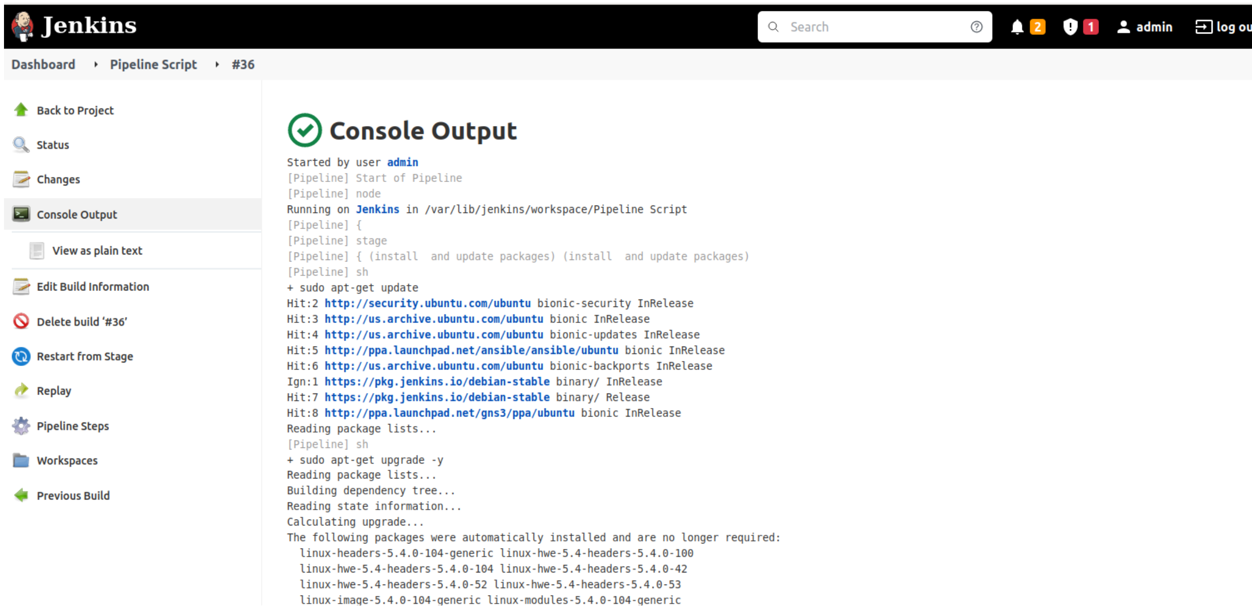
```
netman@netman:/$ ssh -2 -s lab@198.51.100.12 netconf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello><capabilities><capability>urn:ietf:params:netconf:base:1.0</capability><capability>urn:ietf:params:netconf:capability:writeable-running:
1.0</capability><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><capability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:cisco:param
s:netconf:capability:pi-data-model:1.0</capability><capability>urn:cisco:params:netconf:capability:notification:1.0</capability></capabilities><session-id>1757205680</session-id></h
ello>]]>]]>
^Z
[3]+  Stopped                 ssh -2 -s lab@198.51.100.12 netconf
netman@netman:/$ ssh -2 -s lab@198.51.100.13 netconf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello><capabilities><capability>urn:ietf:params:netconf:base:1.0</capability><capability>urn:ietf:params:netconf:capability:writeable-running:
1.0</capability><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><capability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:cisco:param
s:netconf:capability:pi-data-model:1.0</capability><capability>urn:cisco:params:netconf:capability:notification:1.0</capability></capabilities><session-id>1757234392</session-id></h
ello>]]>]]>
^Z
[4]+  Stopped                 ssh -2 -s lab@198.51.100.13 netconf
netman@netman:/$ ssh -2 -s lab@198.51.100.14 netconf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello><capabilities><capability>urn:ietf:params:netconf:base:1.0</capability><capability>urn:ietf:params:netconf:capability:writeable-running:
1.0</capability><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><capability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:cisco:param
s:netconf:capability:pi-data-model:1.0</capability><capability>urn:cisco:params:netconf:capability:notification:1.0</capability></capabilities><session-id>1758178248</session-id></h
ello>]]>]]>
^Z
[5]+  Stopped                 ssh -2 -s lab@198.51.100.14 netconf
netman@netman:/$ ssh -2 -s lab@198.51.100.15 netconf
Password:
<?xml version="1.0" encoding="UTF-8"?><hello><capabilities><capability>urn:ietf:params:netconf:base:1.0</capability><capability>urn:ietf:params:netconf:capability:writeable-running:
1.0</capability><capability>urn:ietf:params:netconf:capability:startup:1.0</capability><capability>urn:ietf:params:netconf:capability:url:1.0</capability><capability>urn:cisco:param
s:netconf:capability:pi-data-model:1.0</capability><capability>urn:cisco:params:netconf:capability:notification:1.0</capability></capabilities><session-id>1746227740</session-id></h
ello>]]>]]>
^Z
[6]+  Stopped                 ssh -2 -s lab@198.51.100.15 netconf
```

4. You will be using the script 'netman_netconf_obj2.py' and 'info.csv' provided to you (they are present in this directory) for the upcoming objective 2.2. (This script achieves objectives mentioned in objective 2 of the previous Netconf lab)

Objective 2 – Creating Jenkinsfile for the pipeline

We will use Jenkinsfile to build our pipeline. The requirements of Jenkinsfile are:

1) It should trigger the Jenkins jobs whenever we push code to the repository. **[10 points]**
   Obj 2.1 is done by doing obj 3.9 where the part with ngrok triggers a build whenever we push the code to the repository.
2) It should have 4 stages (You can add additional stages without compromising the requirements):
   I. **Stage 1: Update/Install packages in the NetMan VM/laptop** (These libraries are used in the script 'netman_netconf_obj2.py')
   Install the below libraries if they are not present already.
   nccclient, pandas, ipaddress, netaddr, prettytable
   [10 points]
   Successfully updated and installed all the above said packages.



Console Output

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Pipeline Script
[Pipeline] {
[Pipeline] stage
[Pipeline] { (install  and update packages) (install  and update packages)
[Pipeline] sh
+ sudo apt-get update
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:8 http://ppa.launchpad.net/gns3/ppa/ubuntu bionic InRelease
Reading package lists...
[Pipeline] sh
+ sudo apt-get upgrade -y
Reading package lists...
Building dependency tree...
Reading state information...
Calculating upgrade...
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-104-generic linux-hwe-5.4-headers-5.4.0-100
  linux-hwe-5.4-headers-5.4.0-104 linux-hwe-5.4-headers-5.4.0-42
  linux-hwe-5.4-headers-5.4.0-52 linux-hwe-5.4-headers-5.4.0-53
  linux-image-5.4.0-104-generic linux-modules-5.4.0-104-generic
```

Pipeline script:

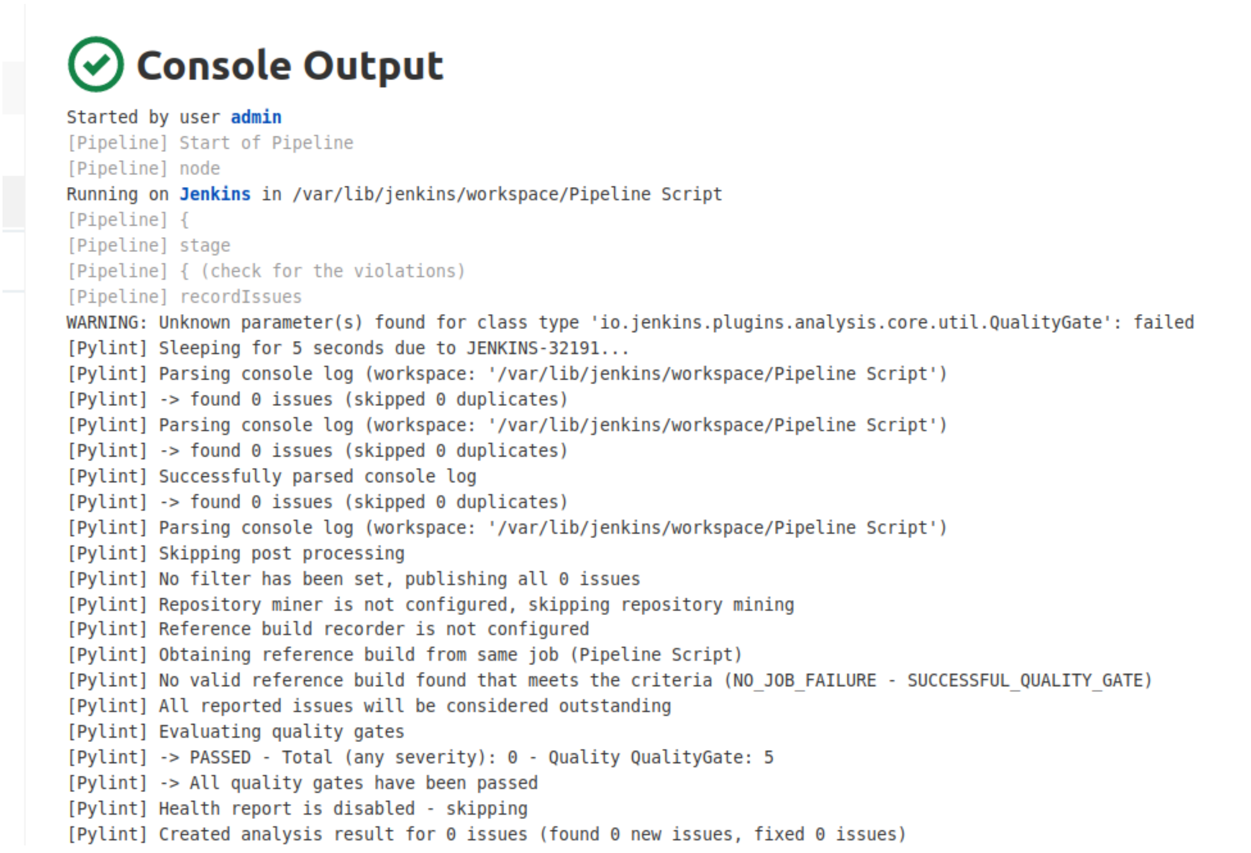
```
pipeline {
    agent any
    stages {
        stage('install  and update packages') {
            steps {
                sh 'sudo apt-get update'
                sh 'sudo apt-get upgrade -y'
                sh 'python3 -m pip install --upgrade pip'
                sh 'python3 -m pip install ncclient pandas ipaddress netaddr prettytable'
            }
        }
    }
}
```

II. **Stage 2: Checking and fixing violations** (Checking if application code complies with PEP8 code style)

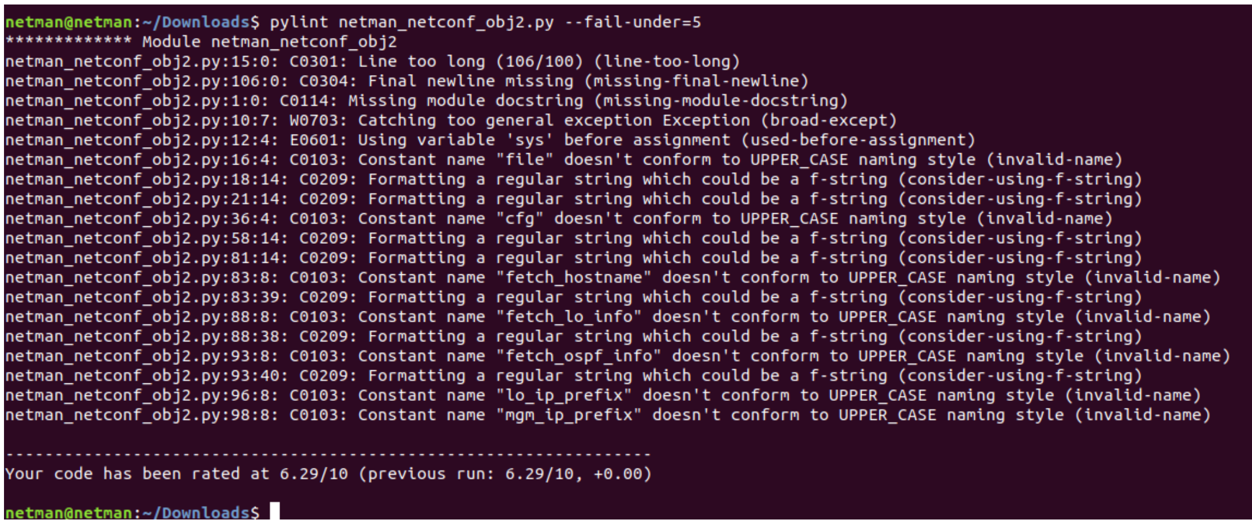
It should report the check-style violations using pylint for the script 'netman_netconf_obj2.py' and in case of violations, the pipeline should fail. Set the quality gate to 5. On failure, violations should be fixed before proceeding further. **[25 points]**

Pipeline Script:

```
stage('check for the violations') {
    steps {
        sh 'python3 -m pip install pylint'
        sh 'pylint netman_netconf_obj2.py --fail-under=5'
    }
}
```



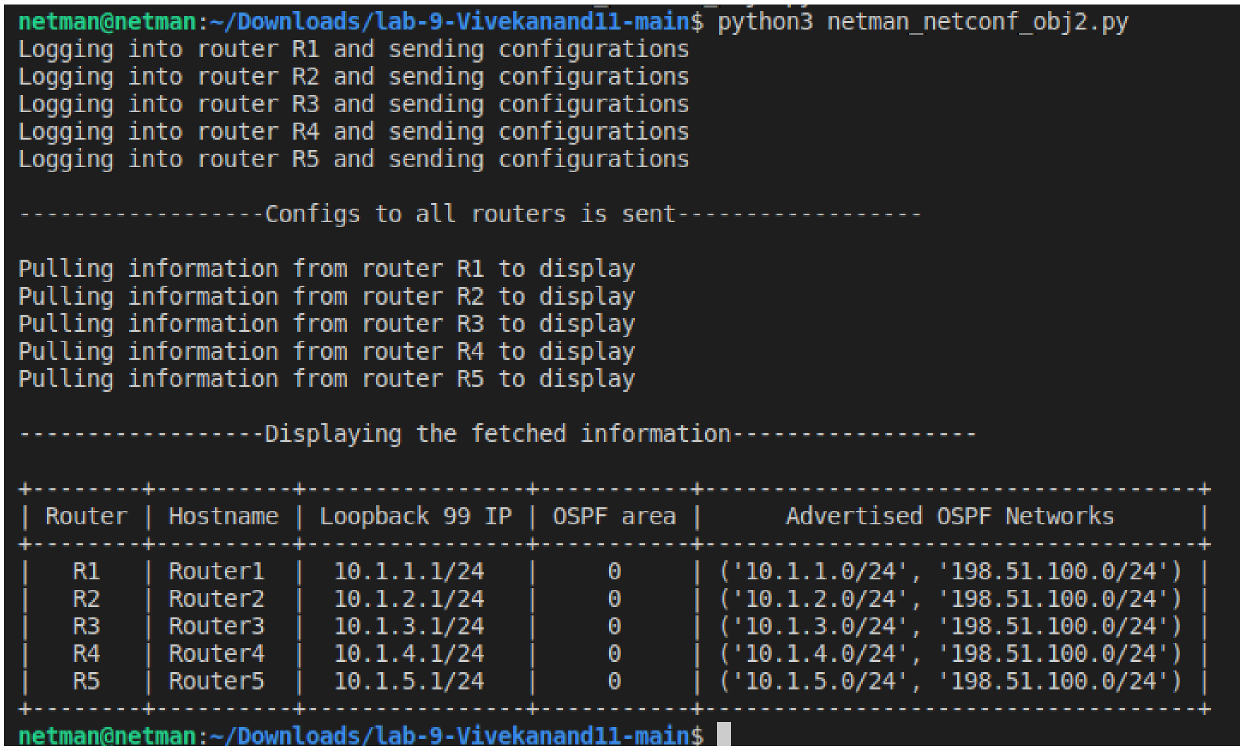
```
Console Output
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Pipeline Script
[Pipeline] {
[Pipeline] stage
[Pipeline] { (check for the violations)
[Pipeline] recordIssues
WARNING: Unknown parameter(s) found for class type 'io.jenkins.plugins.analysis.core.util.QualityGate': failed
[Pylint] Sleeping for 5 seconds due to JENKINS-32191...
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Successfully parsed console log
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] Skipping post processing
[Pylint] No filter has been set, publishing all 0 issues
[Pylint] Repository miner is not configured, skipping repository mining
[Pylint] Reference build recorder is not configured
[Pylint] Obtaining reference build from same job (Pipeline Script)
[Pylint] No valid reference build found that meets the criteria (NO_JOB_FAILURE - SUCCESSFUL_QUALITY_GATE)
[Pylint] All reported issues will be considered outstanding
[Pylint] Evaluating quality gates
[Pylint] -> PASSED - Total (any severity): 0 - Quality QualityGate: 5
[Pylint] -> All quality gates have been passed
[Pylint] Health report is disabled - skipping
[Pylint] Created analysis result for 0 issues (found 0 new issues, fixed 0 issues)
```



```
netman@netman:~/Downloads$ pylint netman_netconf_obj2.py --fail-under=5
************* Module netman_netconf_obj2
netman_netconf_obj2.py:15:0: C0301: Line too long (106/100) (line-too-long)
netman_netconf_obj2.py:106:0: C0304: Final newline missing (missing-final-newline)
netman_netconf_obj2.py:1:0: C0114: Missing module docstring (missing-module-docstring)
netman_netconf_obj2.py:10:7: W0703: Catching too general exception Exception (broad-except)
netman_netconf_obj2.py:12:4: E0601: Using variable 'sys' before assignment (used-before-assignment)
netman_netconf_obj2.py:16:4: C0103: Constant name "file" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:18:14: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:21:14: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:36:4: C0103: Constant name "cfg" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:58:14: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:81:14: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:83:8: C0103: Constant name "fetch_hostname" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:83:39: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:88:8: C0103: Constant name "fetch_lo_info" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:88:38: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:93:8: C0103: Constant name "fetch_ospf_info" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:93:40: C0209: Formatting a regular string which could be a f-string (consider-using-f-string)
netman_netconf_obj2.py:96:8: C0103: Constant name "lo_ip_prefix" doesn't conform to UPPER_CASE naming style (invalid-name)
netman_netconf_obj2.py:98:8: C0103: Constant name "mgm_ip_prefix" doesn't conform to UPPER_CASE naming style (invalid-name)

------------------------------------------------------------------
Your code has been rated at 6.29/10 (previous run: 6.29/10, +0.00)

netman@netman:~/Downloads$ 
```

III. **Stage 3: Running the application** (Executing the application code to be tested in the next objective for functionality)

It should run the application code 'netman_netconf_obj2.py'. **[5 points]**



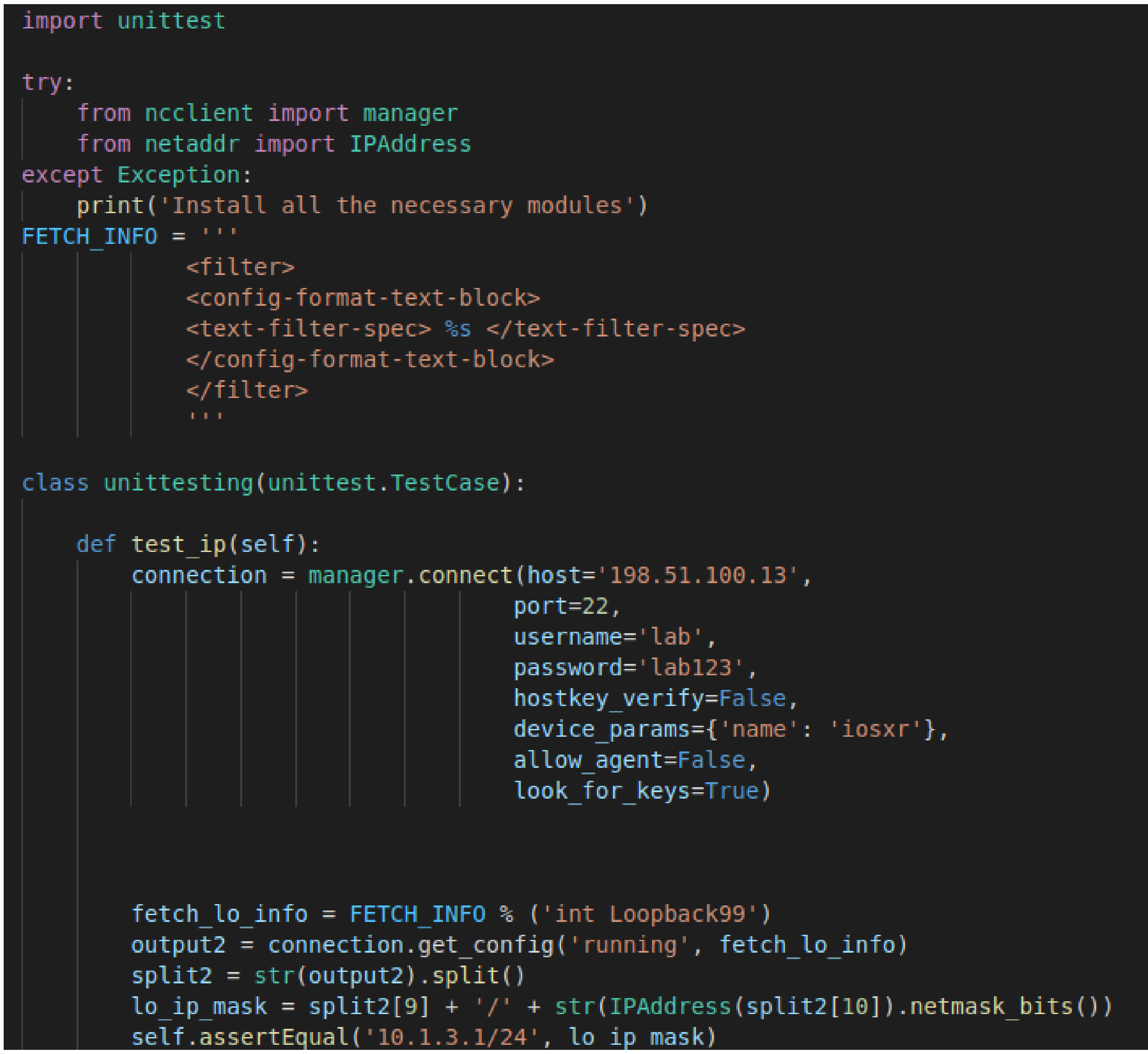
```
netman@netman:~/Downloads/lab-9-Vivekanand11-main$ python3 netman_netconf_obj2.py
Logging into router R1 and sending configurations
Logging into router R2 and sending configurations
Logging into router R3 and sending configurations
Logging into router R4 and sending configurations
Logging into router R5 and sending configurations

------------------Configs to all routers is sent-----------------

Pulling information from router R1 to display
Pulling information from router R2 to display
Pulling information from router R3 to display
Pulling information from router R4 to display
Pulling information from router R5 to display

------------------Displaying the fetched information-----------------

+--------+----------+----------------+-----------+-----------------------------------+
| Router | Hostname | Loopback 99 IP | OSPF area |      Advertised OSPF Networks     |
+--------+----------+----------------+-----------+-----------------------------------+
|   R1   | Router1  |  10.1.1.1/24   |     0     | ('10.1.1.0/24', '198.51.100.0/24') |
|   R2   | Router2  |  10.1.2.1/24   |     0     | ('10.1.2.0/24', '198.51.100.0/24') |
|   R3   | Router3  |  10.1.3.1/24   |     0     | ('10.1.3.0/24', '198.51.100.0/24') |
|   R4   | Router4  |  10.1.4.1/24   |     0     | ('10.1.4.0/24', '198.51.100.0/24') |
|   R5   | Router5  |  10.1.5.1/24   |     0     | ('10.1.5.0/24', '198.51.100.0/24') |
+--------+----------+----------------+-----------+-----------------------------------+
netman@netman:~/Downloads/lab-9-Vivekanand11-main$
```

IV. **Stage 4: Unit test** (Testing if the application is meeting the requirements)

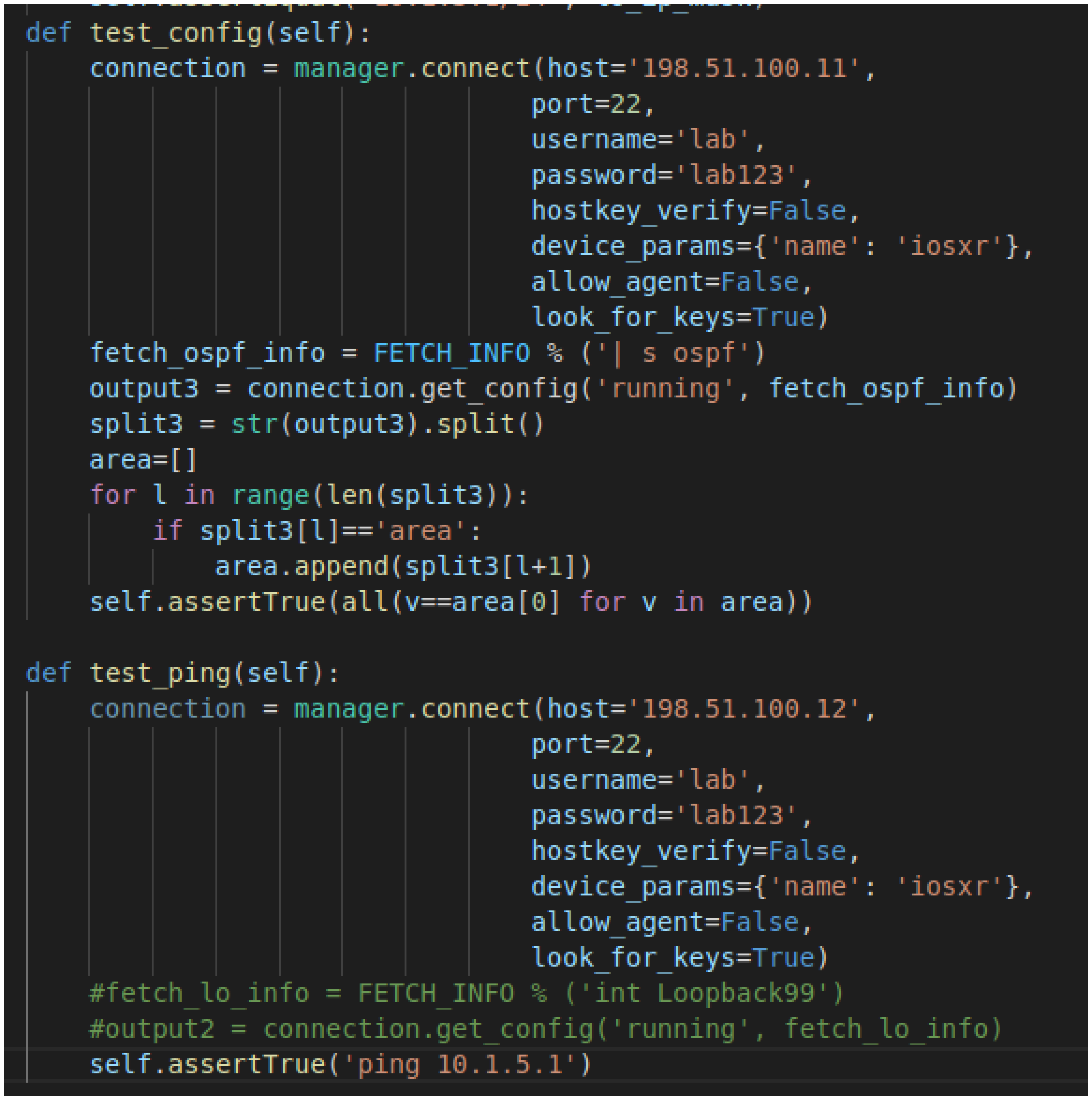
Write unit tests using Python's unittest framework to test the following scenarios:

1) If the IP address of loopback 99 on router 3 is 10.1.3.1/24
2) If R1 is configured only for a single area
3) If a ping from router 2's loopback to router 5's loopback is successful

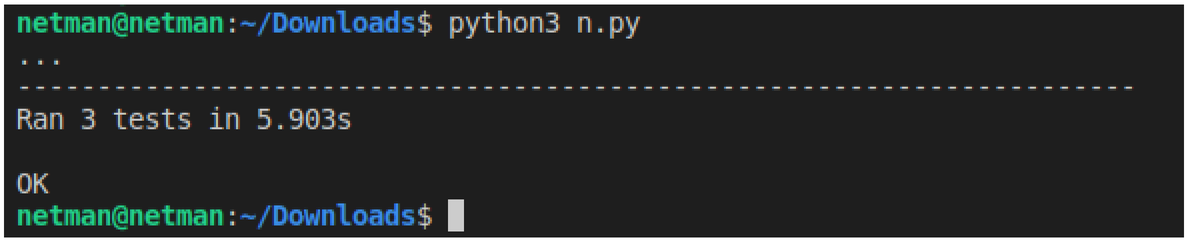
[30 points]



```python
import unittest

try:
    from ncclient import manager
    from netaddr import IPAddress
except Exception:
    print('Install all the necessary modules')
FETCH_INFO = '''
            <filter>
            <config-format-text-block>
            <text-filter-spec> %s </text-filter-spec>
            </config-format-text-block>
            </filter>
            '''

class unittesting(unittest.TestCase):

    def test_ip(self):
        connection = manager.connect(host='198.51.100.13',
                                    port=22,
                                    username='lab',
                                    password='lab123',
                                    hostkey_verify=False,
                                    device_params={'name': 'iosxr'},
                                    allow_agent=False,
                                    look_for_keys=True)



        fetch_lo_info = FETCH_INFO % ('int Loopback99')
        output2 = connection.get_config('running', fetch_lo_info)
        split2 = str(output2).split()
        lo_ip_mask = split2[9] + '/' + str(IPAddress(split2[10]).netmask_bits())
        self.assertEqual('10.1.3.1/24', lo_ip_mask)
```

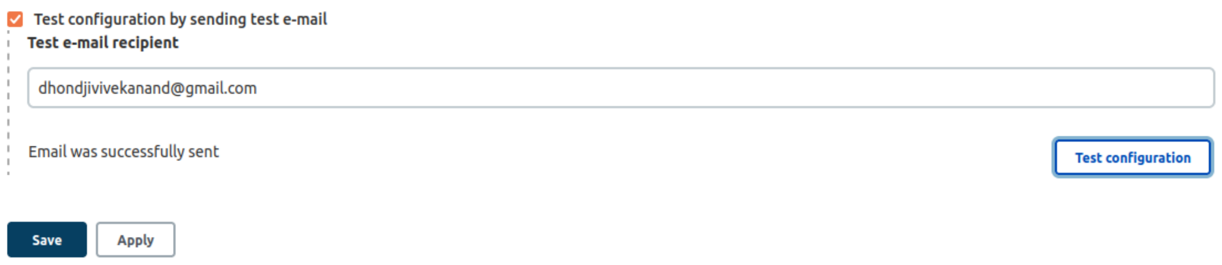


```
def test_config(self):
    connection = manager.connect(host='198.51.100.11',
                                 port=22,
                                 username='lab',
                                 password='lab123',
                                 hostkey_verify=False,
                                 device_params={'name': 'iosxr'},
                                 allow_agent=False,
                                 look_for_keys=True)
    fetch_ospf_info = FETCH_INFO % ('| s ospf')
    output3 = connection.get_config('running', fetch_ospf_info)
    split3 = str(output3).split()
    area=[]
    for l in range(len(split3)):
        if split3[l]=='area':
            area.append(split3[l+1])
    self.assertTrue(all(v==area[0] for v in area))

def test_ping(self):
    connection = manager.connect(host='198.51.100.12',
                                 port=22,
                                 username='lab',
                                 password='lab123',
                                 hostkey_verify=False,
                                 device_params={'name': 'iosxr'},
                                 allow_agent=False,
                                 look_for_keys=True)
    #fetch_lo_info = FETCH_INFO % ('int Loopback99')
    #output2 = connection.get_config('running', fetch_lo_info)
    self.assertTrue('ping 10.1.5.1')
```

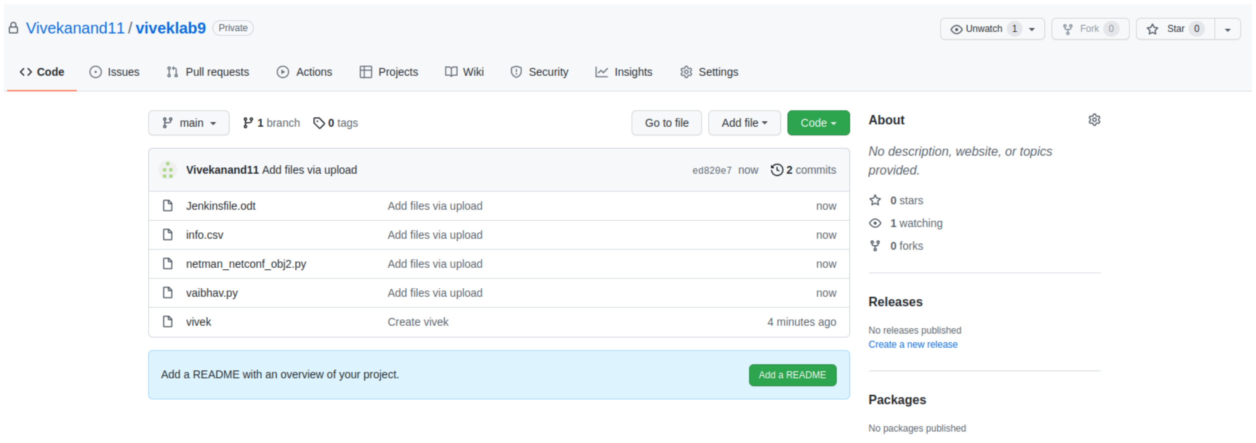


```
netman@netman:~/Downloads$ python3 n.py
...
----------------------------------------------------------------------
Ran 3 tests in 5.903s

OK
netman@netman:~/Downloads$ 
```

3) The post action should send an email from Jenkin's admin to you informing the build success/failure. This is explained in detail in objective 4 (below) **[5 points]**


Test configuration by sending test e-mail
Test e-mail recipient
dhondjivivekanand@gmail.com
Email was successfully sent

4) Create a private repository in GitHub with the Jenkinsfile, the Unit test script, and the provided script along with the CSV file. Provide a relevant screenshot. **[5 points]**


Vivekanand11 / viveklab9 Private

Vivekanand11 Add files via upload ed820e7 now 2 commits

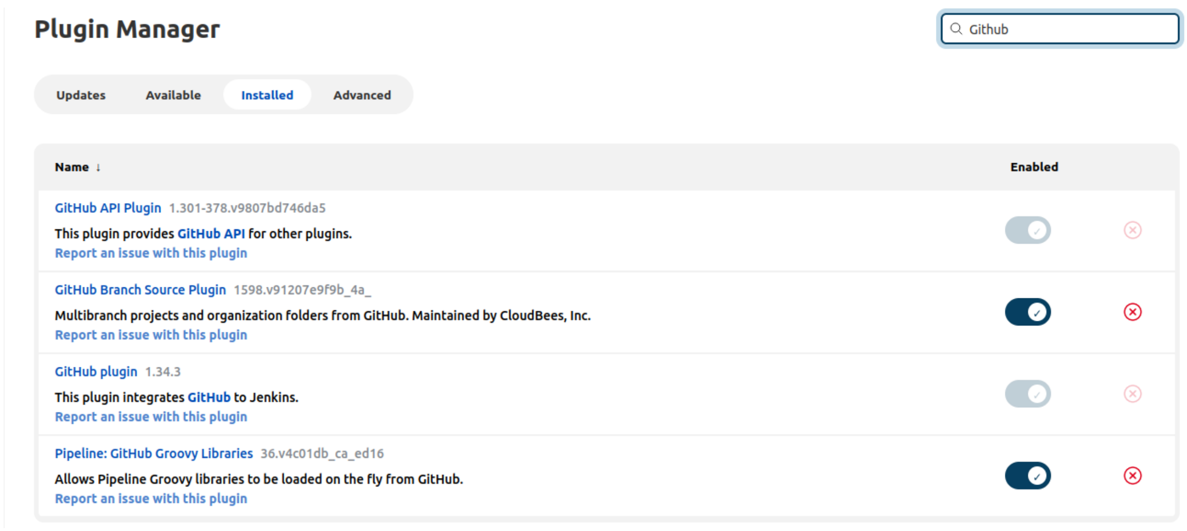
Jenkinsfile.odt	Add files via upload	now
info.csv	Add files via upload	now
netman_netconf_obj2.py	Add files via upload	now
vaibhav.py	Add files via upload	now
vivek	Create vivek	4 minutes ago

Add a README with an overview of your project.

About
No description, website, or topics provided.
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published

Objective 3 – Adding GitHub project to the pipeline and configuring GitHub webhook

1. Install GitHub plugin in Jenkins. Provide a screenshot showing that it is installed. **[3 points]**

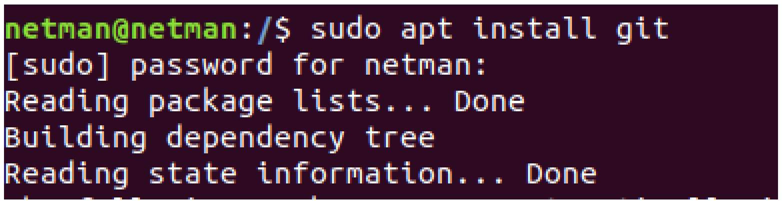

Plugin Manager

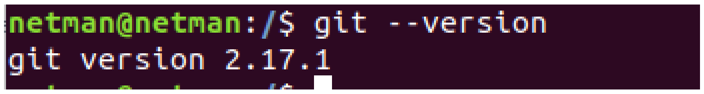
Github

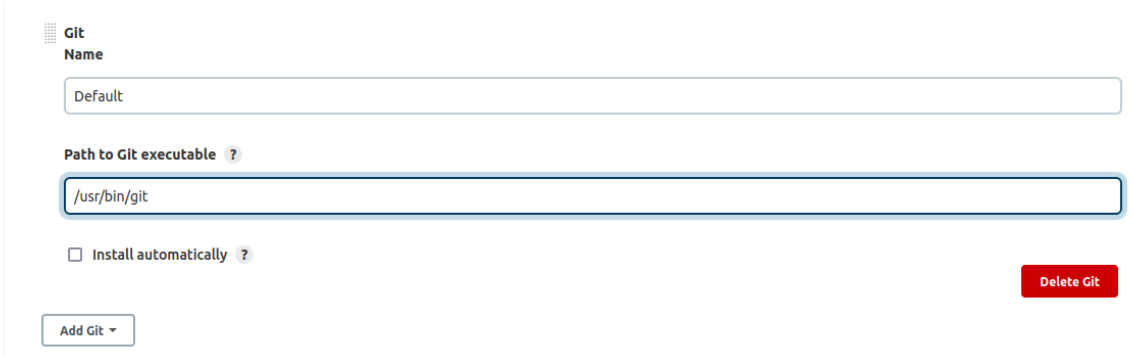
Updates Available Installed Advanced

Name ↓	Enabled
GitHub API Plugin 1.301-378.v9807bd746da5 This plugin provides GitHub API for other plugins. Report an issue with this plugin	✓
GitHub Branch Source Plugin 1598.v91207e9f9b_4a_ Multibranch projects and organization folders from GitHub. Maintained by CloudBees, Inc. Report an issue with this plugin	✓
GitHub plugin 1.34.3 This plugin integrates GitHub to Jenkins. Report an issue with this plugin	✓
Pipeline: GitHub Groovy Libraries 36.v4c01db_ca_ed16 Allows Pipeline Groovy libraries to be loaded on the fly from GitHub. Report an issue with this plugin	✓

2. Install Git in the NetMan VM/laptop if it is not installed already and set the path to executable in 'Manage Jenkins'-> 'Global Tool Configuration'->'Git'. Provide a screenshot. **[5 points]**


```
netman@netman:/$ sudo apt install git
[sudo] password for netman:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

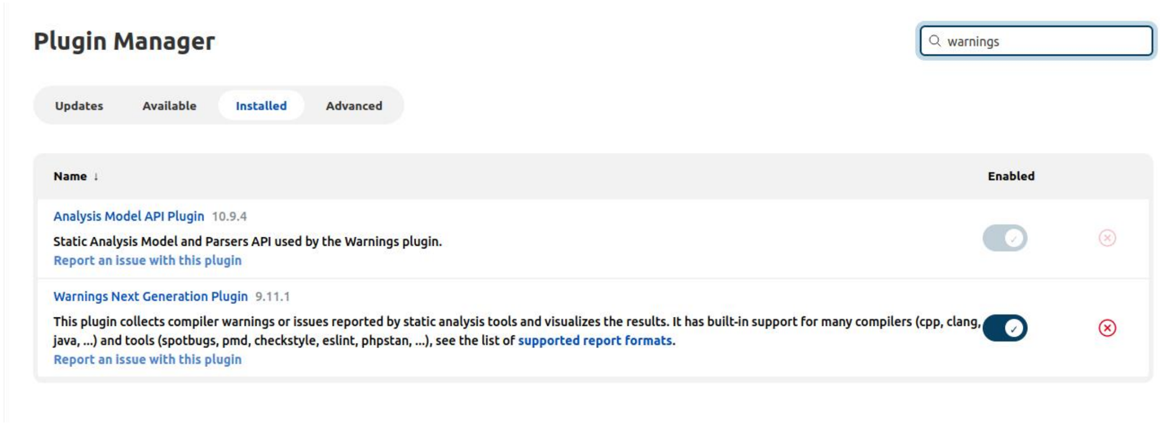

```
netman@netman:/$ git --version
git version 2.17.1
```


Git
Name
Default

Path to Git executable
/usr/bin/git

☐ Install automatically

3. Install 'Warnings Next Generation' plugin in Jenkins. Provide a screenshot showing that it is installed. **[2 points]**



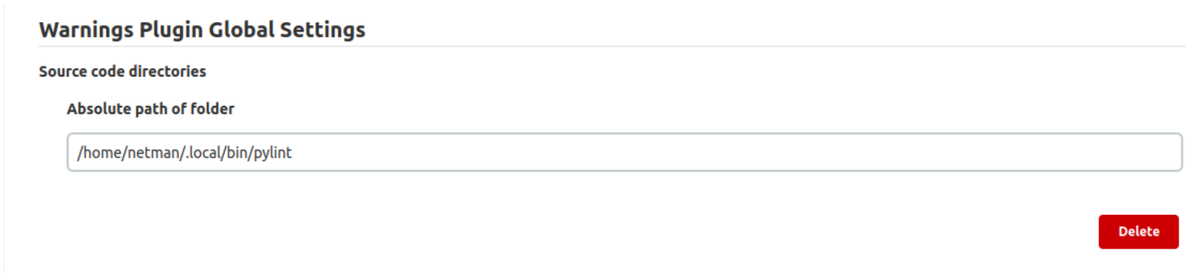
Plugin Manager

warnings

Updates | Available | Installed | Advanced

Name ↓	Enabled
Analysis Model API Plugin 10.9.4 Static Analysis Model and Parsers API used by the Warnings plugin. Report an issue with this plugin	✓
Warnings Next Generation Plugin 9.11.1 This plugin collects compiler warnings or issues reported by static analysis tools and visualizes the results. It has built-in support for many compilers (cpp, clang, java, ...) and tools (spotbugs, pmd, checkstyle, eslint, phpstan, ...), see the list of supported report formats. Report an issue with this plugin	✓

4. Install pylint in NetMan VM/laptop using pip3 and set the path to executable in ‘Manage Jenkins’ -> ‘Configure System’ -> ‘Warnings Next Generation Plugin Global Settings’. Provide a screenshot. **[5 points]**

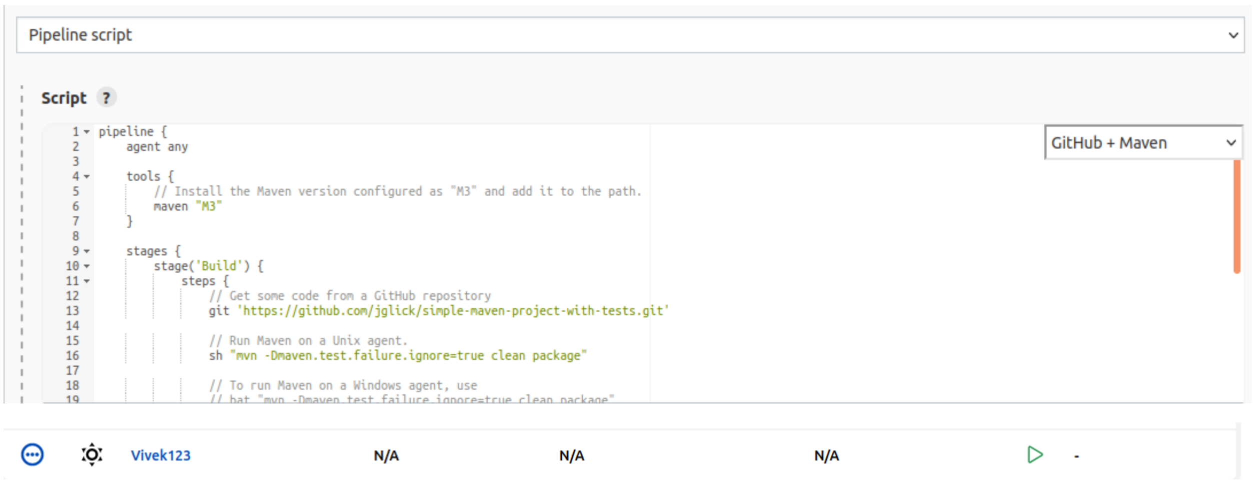


Warnings Plugin Global Settings

Source code directories

Absolute path of folder: /home/netman/.local/bin/pylint

5. Create a pipeline project from Jenkins Dashboard. Provide a screenshot. **[5 points]**



Pipeline script

Script ?

GitHub + Maven

```
pipeline {
    agent any

    tools {
        // Install the Maven version configured as "M3" and add it to the path.
        maven "M3"
    }

    stages {
        stage('Build') {
            steps {
                // Get some code from a GitHub repository
                git 'https://github.com/jglick/simple-maven-project-with-tests.git'

                // Run Maven on a Unix agent.
                sh "mvn -Dmaven.test.failure.ignore=true clean package"

                // To run Maven on a Windows agent, use
                // bat "mvn -Dmaven.test.failure.ignore=true clean package"
```

Vivek123	N/A	N/A	N/A	▷	-

6. Go to your pipeline project. Under ‘General’ tab, enter your project URL.
7. In ‘Build Triggers’ tab, enable ‘Github hook trigger for GITScm polling’.
8. In ‘Pipeline’ tab:

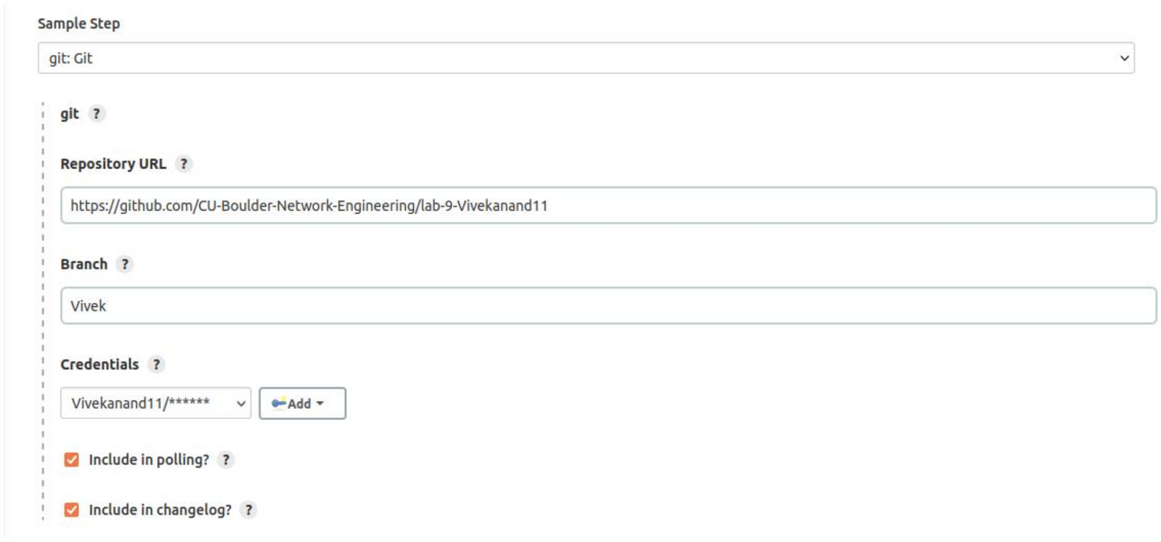
I. Select 'Definition' as Pipeline script from SCM

II. Select 'SCM' as Git

III. Provide repository URL along with a method of credentials (of your choice).

IV. Enter the branch name in 'Branches to Build'

Save the changes and provide relevant screenshots. **[10 points]**



Sample Step

git: Git

git

Repository URL

https://github.com/CU-Boulder-Network-Engineering/lab-9-Vivekanand11

Branch

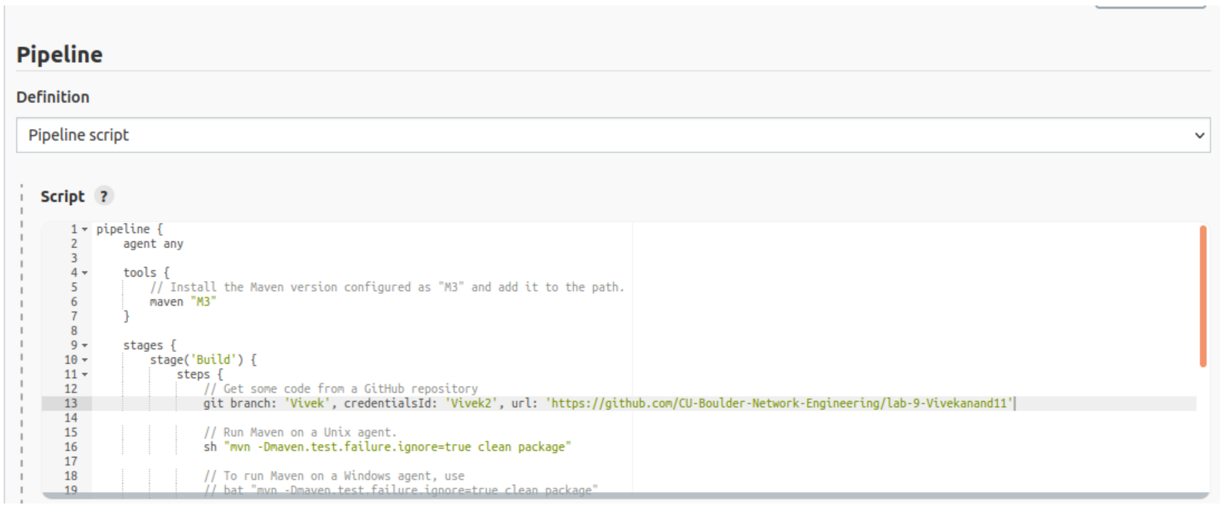
Vivek

Credentials

Vivekanand11/******

Include in polling? ✓

Include in changelog? ✓



Pipeline

Definition

Pipeline script

Script

```
pipeline {
    agent any

    tools {
        // Install the Maven version configured as "M3" and add it to the path.
        maven "M3"
    }

    stages {
        stage('Build') {
            steps {
                // Get some code from a GitHub repository
                git branch: 'Vivek', credentialsId: 'Vivek2', url: 'https://github.com/CU-Boulder-Network-Engineering/lab-9-Vivekanand11'

                // Run Maven on a Unix agent.
                sh "mvn -Dmaven.test.failure.ignore=true clean package"

                // To run Maven on a Windows agent, use
                // bat "mvn -Dmaven.test.failure.ignore=true clean package"
```

9. Configure GitHub Webhook:

GitHub integration with Jenkins needs a public URL for the webhook. Use ngrok to create a third-party tunnel for the same.

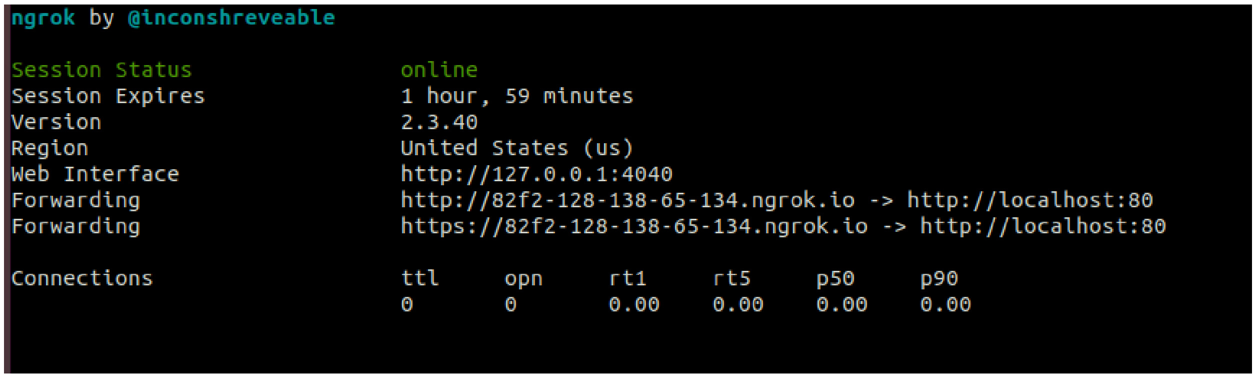
Go to your GitHub repository settings and in 'webhook' tab, provide:

I. Newly generated URL in 'Payload URL' field

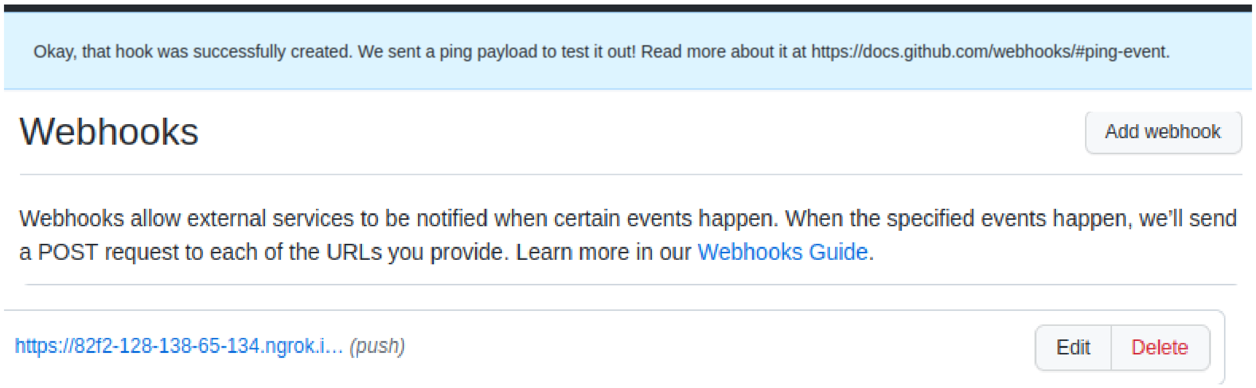
II. 'Content Type' as application/x-www-form-urlencoded.

III. Enable it for ‘push’ event.

Provide a screenshot. **[10 points]**



```
ngrok by @inconshreveable

Session Status                online
Session Expires               1 hour, 59 minutes
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://82f2-128-138-65-134.ngrok.io -> http://localhost:80
Forwarding                    https://82f2-128-138-65-134.ngrok.io -> http://localhost:80

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

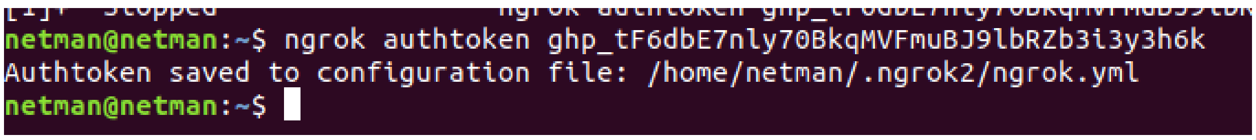


Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://docs.github.com/webhooks/#ping-event.

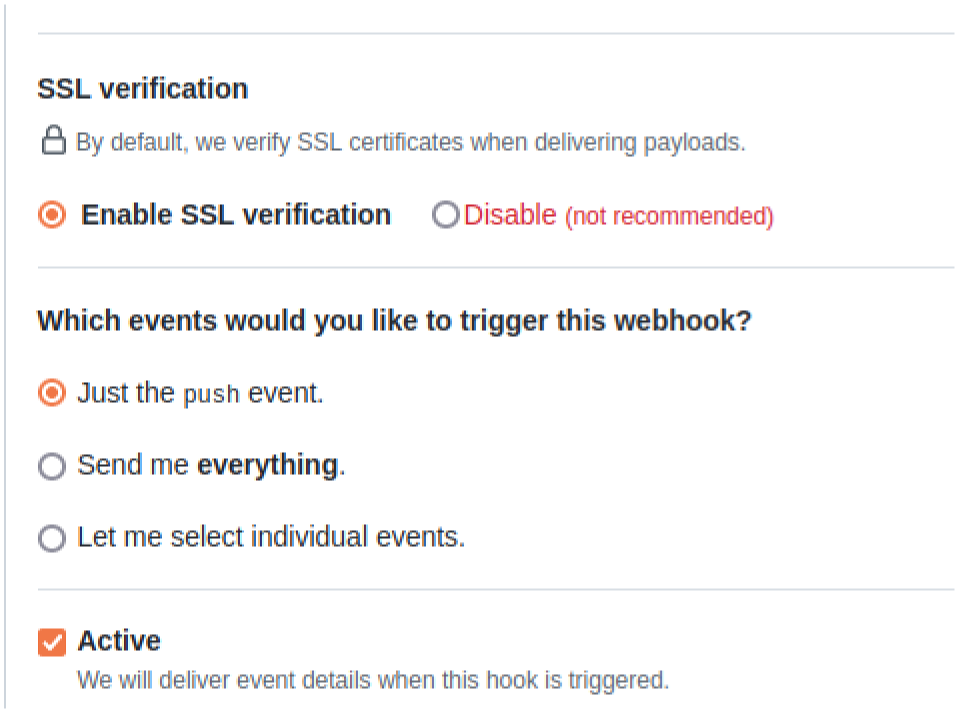
Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

https://82f2-128-138-65-134.ngrok.i… (push)



```
[1]+  Stopped                 ngrok authtoken ghp_tF6dbE7nly70BkqMVFmuBJ9lbR
netman@netman:~$ ngrok authtoken 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
Authtoken saved to configuration file: /home/netman/.ngrok2/ngrok.yml
netman@netman:~$ 
```


SSL verification

By default, we verify SSL certificates when delivering payloads.

[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?

[✓ Just the push event.]

Send me everything.

Let me select individual events.

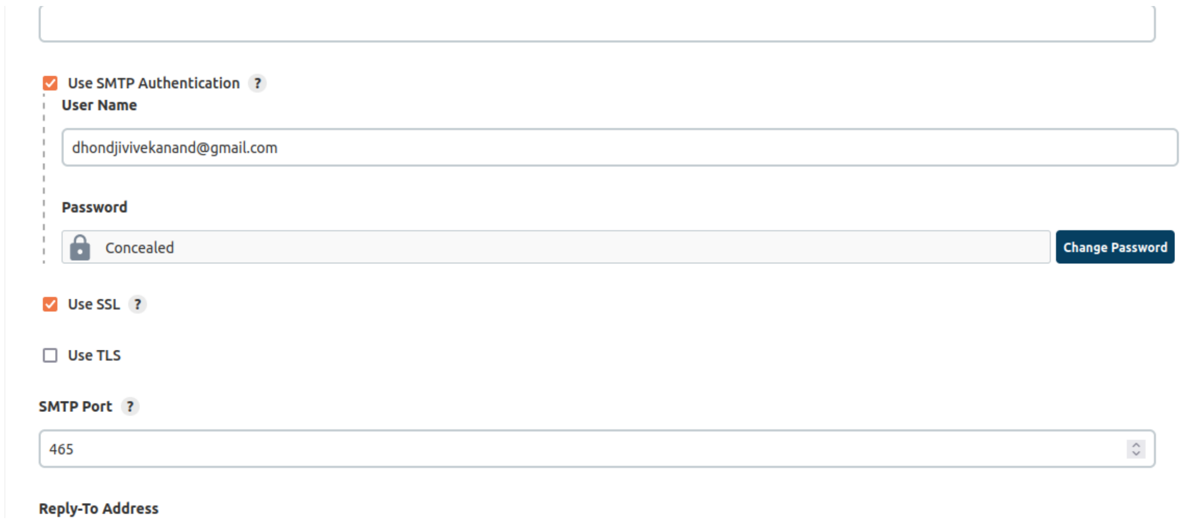
[✓] Active

We will deliver event details when this hook is triggered.

Note: Do not use this URL to access Jenkins (it does not work properly for this purpose), use this just to integrate GitHub with Jenkins

Objective 4 – Configure E-mail notifications

1. Go to manage Jenkins -> Configure system -> Configure E-mail notification and extended E-mail notifications. Provide a screenshot of configurations **[10 points]**


[✓] Use SMTP Authentication

User Name: dhondjivivekanand@gmail.com

Password: Concealed

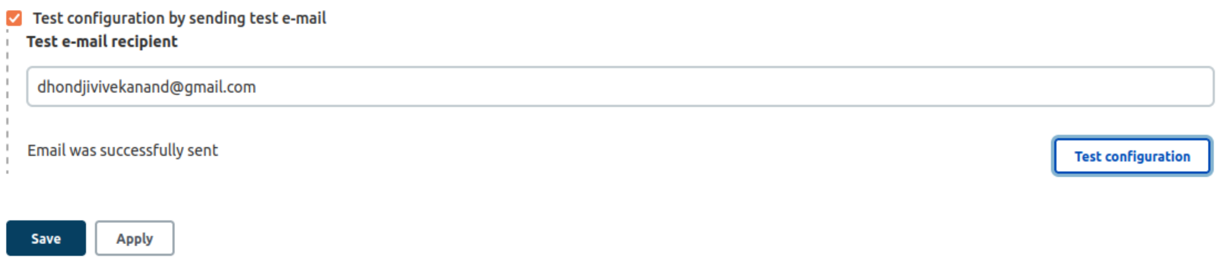
[✓] Use SSL

[ ] Use TLS

SMTP Port: 465

Reply-To Address

2. Hit ‘Test configuration by sending test e-mail’ in E-mail notifications and provide screenshot of receiving an email. **[2 points]**

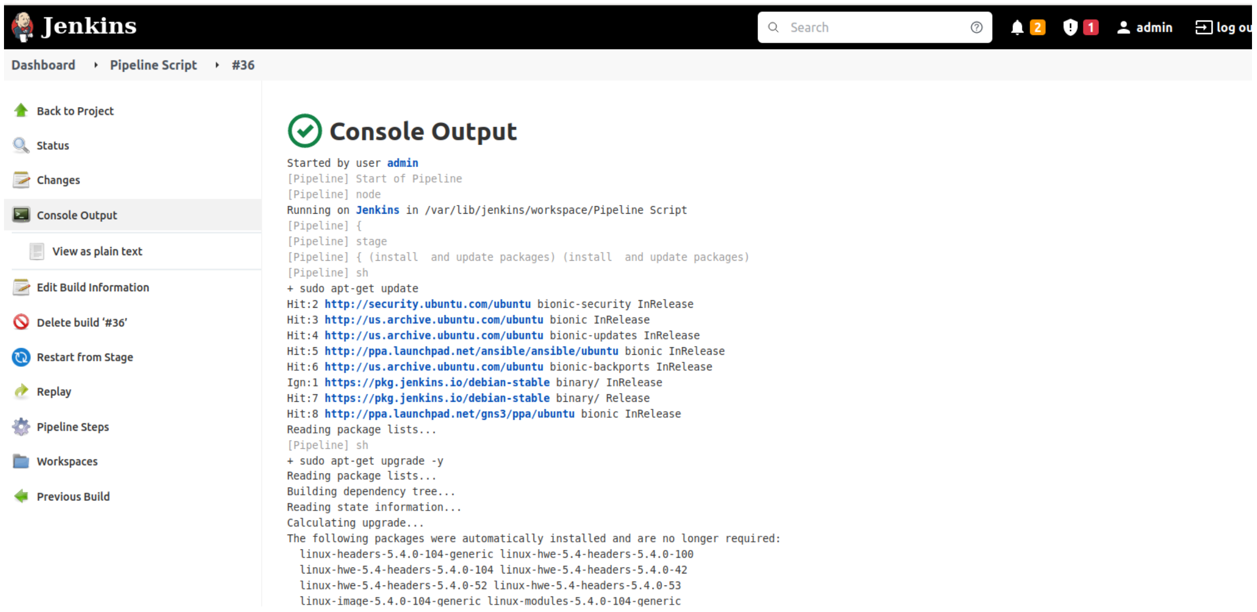


Test configuration by sending test e-mail
Test e-mail recipient
dhondjivivekanand@gmail.com
Email was successfully sent

Objective 5 – Running Jenkins Job

1. Go to Jenkins and hit ‘build now’. GitHub webhook needs one build to be run manually first to trigger the builds later automatically. Provide a screenshot of console output and a SUCCESSFUL build. If your build fails, work to fix it. **[10 points]**

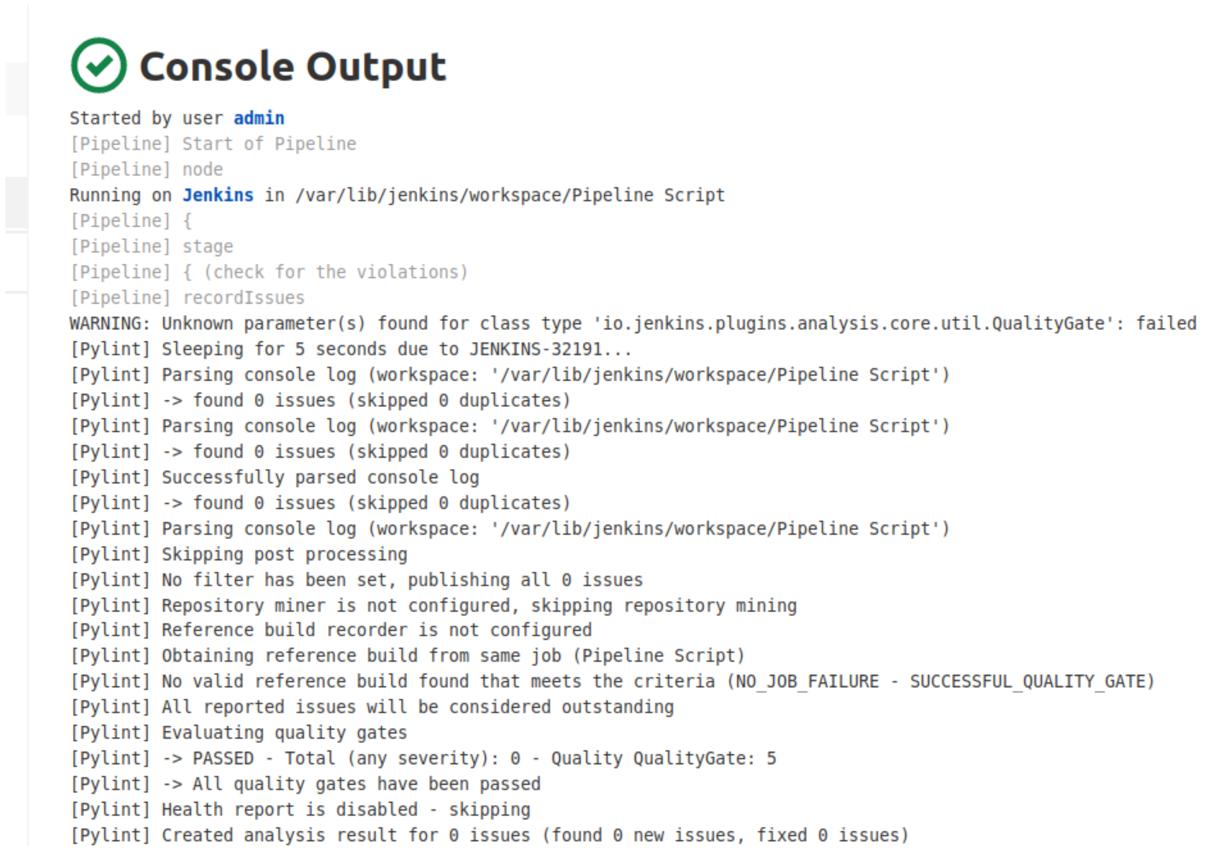
Packages build:



Console Output

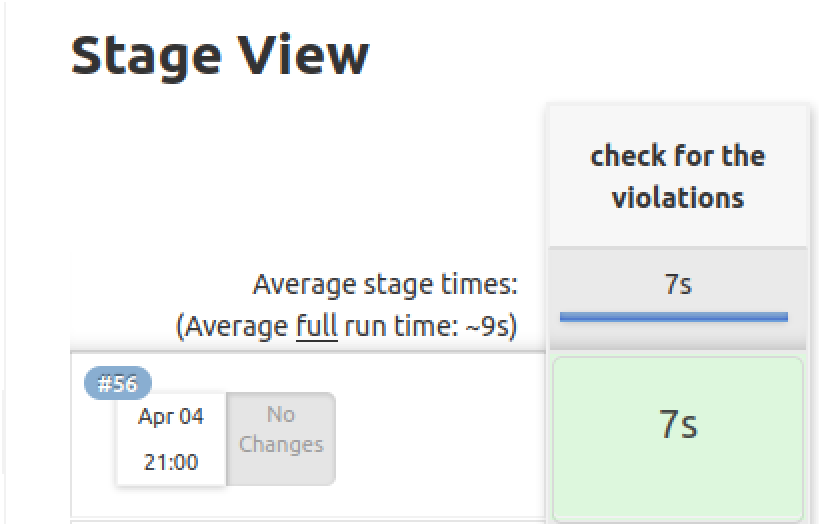
```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Pipeline Script
[Pipeline] {
[Pipeline] stage
[Pipeline] { (install  and update packages) (install  and update packages)
[Pipeline] sh
+ sudo apt-get update
Hit:2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://ppa.launchpad.net/ansible/ansible/ubuntu bionic InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Ign:1 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:8 http://ppa.launchpad.net/gns3/ppa/ubuntu bionic InRelease
Reading package lists...
[Pipeline] sh
+ sudo apt-get upgrade -y
Reading package lists...
Building dependency tree...
Reading state information...
Calculating upgrade...
The following packages were automatically installed and are no longer required:
  linux-headers-5.4.0-104-generic linux-hwe-5.4-headers-5.4.0-100
  linux-hwe-5.4-headers-5.4.0-104 linux-hwe-5.4-headers-5.4.0-42
  linux-hwe-5.4-headers-5.4.0-52 linux-hwe-5.4-headers-5.4.0-53
  linux-image-5.4.0-104-generic linux-modules-5.4.0-104-generic
```

Violations Check:



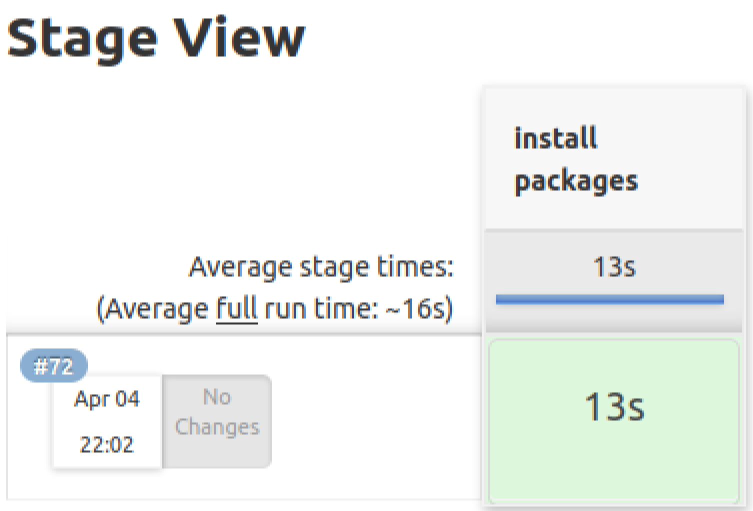
Console Output

```
Started by user admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Pipeline Script
[Pipeline] {
[Pipeline] stage
[Pipeline] { (check for the violations)
[Pipeline] recordIssues
WARNING: Unknown parameter(s) found for class type 'io.jenkins.plugins.analysis.core.util.QualityGate': failed
[Pylint] Sleeping for 5 seconds due to JENKINS-32191...
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Successfully parsed console log
[Pylint] -> found 0 issues (skipped 0 duplicates)
[Pylint] Parsing console log (workspace: '/var/lib/jenkins/workspace/Pipeline Script')
[Pylint] Skipping post processing
[Pylint] No filter has been set, publishing all 0 issues
[Pylint] Repository miner is not configured, skipping repository mining
[Pylint] Reference build recorder is not configured
[Pylint] Obtaining reference build from same job (Pipeline Script)
[Pylint] No valid reference build found that meets the criteria (NO_JOB_FAILURE - SUCCESSFUL_QUALITY_GATE)
[Pylint] All reported issues will be considered outstanding
[Pylint] Evaluating quality gates
[Pylint] -> PASSED - Total (any severity): 0 - Quality QualityGate: 5
[Pylint] -> All quality gates have been passed
[Pylint] Health report is disabled - skipping
[Pylint] Created analysis result for 0 issues (found 0 new issues, fixed 0 issues)
```



Stage View

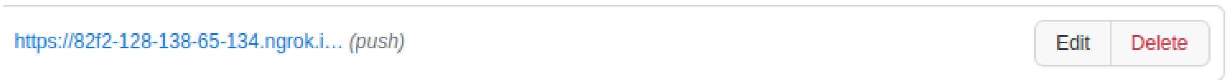
	check for the violations
Average stage times: (Average full run time: ~9s)	7s
#56 Apr 04 21:00 No Changes	7s



Stage View

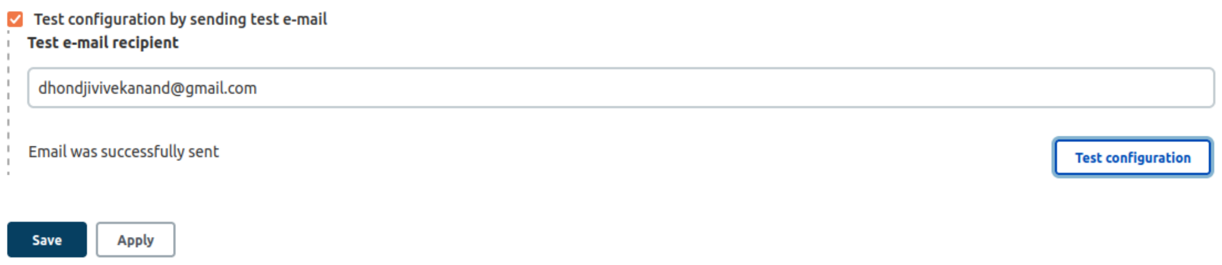
	install packages
Average stage times: (Average full run time: ~16s)	13s
#72 Apr 04 22:02 No Changes	13s

2. Now, make a small change in any file. Commit it to GitHub. The job should start automatically. Show the console output of your next Jenkins job succeeding. The console output should show 'started by GitHub push…' **[10 points]**



https://82f2-128-138-65-134.ngrok.i… *(push)*

3. Provide a screenshot of receiving an email about a SUCCESSFUL build. **[10 points]**


☑ Test configuration by sending test e-mail

Test e-mail recipient

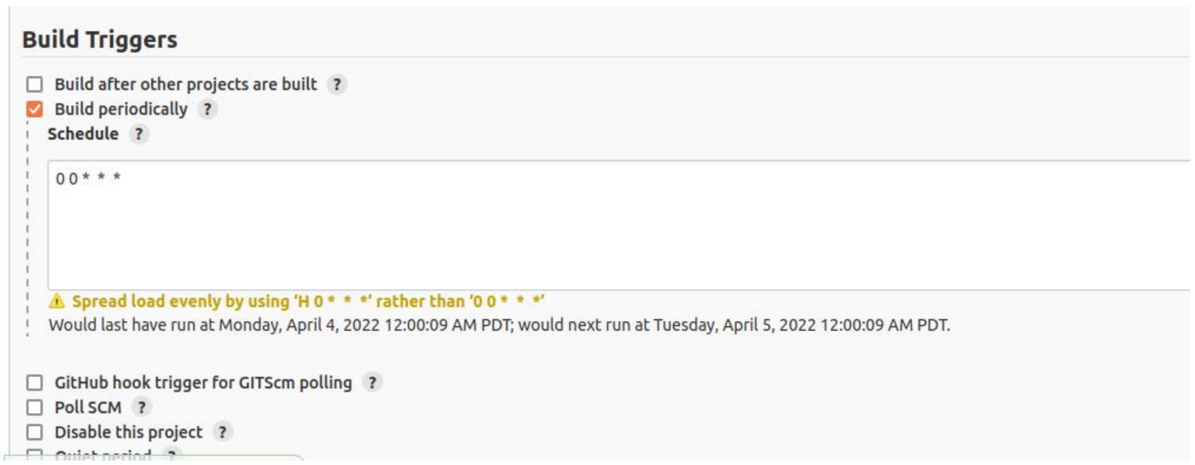
dhondjivivekanand@gmail.com

Email was successfully sent

Objective 6 – Schedule Nightly Jobs

Configure Jenkins to run GitHub jobs every night.

1. Provide a relevant screenshot. **[10 points]**


Build Triggers

☐ Build after other projects are built

☑ Build periodically

Schedule

```
0 0 * * *
```

⚠ Spread load evenly by using 'H 0 * * *' rather than '0 0 * * *'

Would last have run at Monday, April 4, 2022 12:00:09 AM PDT; would next run at Tuesday, April 5, 2022 12:00:09 AM PDT.

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Disable this project

2. Briefly explain the significance of this. **[5 points]**

   The beauty of doing this is that if we run the github jobs every night since no developer is working on any task at night however through emails he will be notified about the bugs and when he gets up in the morning he will see the bugs received through emails and can fix it. Therefore, this helps in continuous integration, continuous deployment.

Report Questions [15 points]

1. Do you think Jenkins is actively used in the industry? Why?

   Jenkins is actively used as the open-source server. It provides hundreds of plugins to support build, deploy and automate project which in turns helps in continuous integration and build automation. The basic functionality of Jenkins is to execute a predefined list of tasks i.e. as soon as there is a change in repository, triggers a pipeline that executes different jobs and performs the tasks.

2. What is another CI/CD tool you would like to learn and why?

   I would learn GitLab because of the following features of it:

   a. Provides a single source of truth and scalability.
   b. View, create and manage codes through branching tools.
   c. Helps in automation and shortens the release and delivery of applications.

3. Where else could you run your Jenkins jobs?

   We can run it on local host. By default, Jenkins uses port 8080. Hit localhost 8080 in the browser and it redirects you to the login page.

4. Explain the use of Unit Testing in Jenkins.

   Unit testing is important in Jenkins because it ensures that the changes made by developers to the projects and functions are as per the requirements. Unit testing is done to make sure whether the particular unit/software is functioning as per required/desired.

5. What other tests can be carried out before an application deployment?

   Some other tests that are carried out before an application deployment are manually verifying functionality, performing end-to-end testing. Checking it in the test environment (GNS3) then in the lab environment and then in the production environment.

Total Points ________ / 226