

[illegible]

Name	DevOps Pipeline: Nginx
URL	https://attackdefense.com/challengedetails?cid=2070
Type	Pipeline Basics: Software

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

DevOps practices are to combine software development (Dev) and IT operations (Ops) in order to improve the delivery process. DevOps pipelines are chained tasks and components that run in a sequence to cover different phases of software compilation, packaging, automated testing, and test deployment.

In this lab, we have a simple DevOps pipeline for Nginx. The pipeline consists of the following components (and tasks):

- Kali machine (For pulling, modifying, and pushing the code)
- GitLab server (For hosting code)
- Jenkins server (For integrating). Different phases and components used:
 - Building Binary: make
 - Test Deployment: Ansible
 - Dynamic Testing: Selenium
- Test server (For test deployment)

Objective: Run the pipeline and observe/understand the DevOps process!

Instructions:

- The GitLab server is reachable with the name 'gitlab'
- Gitlab credentials:

Username	Password
root	welcome123

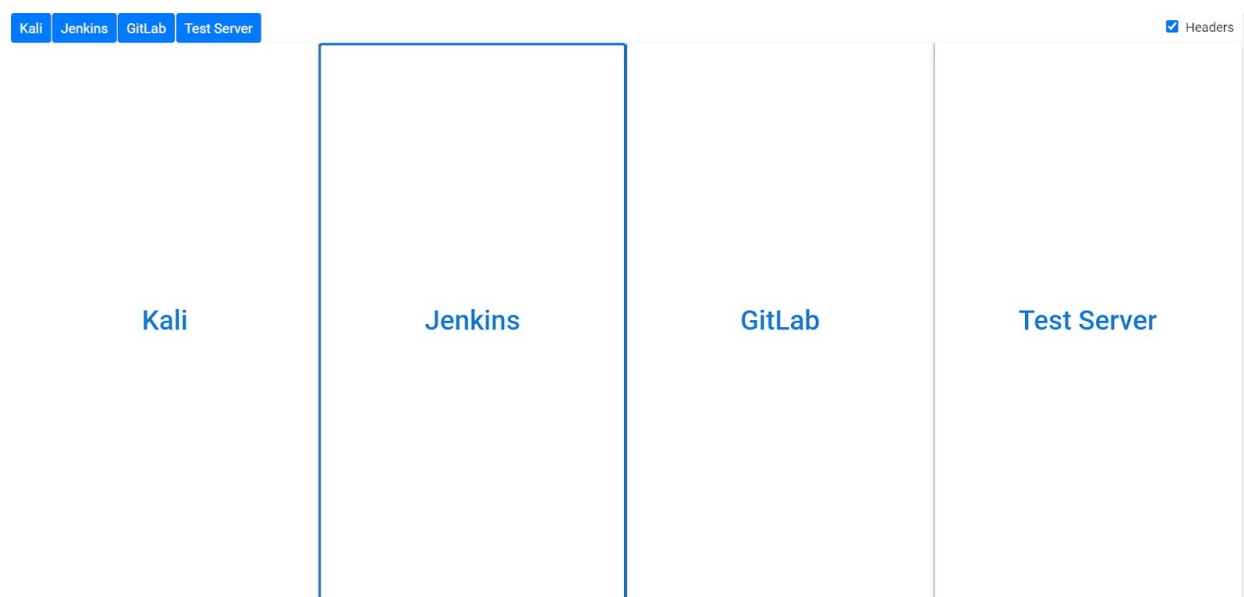
- The Jenkins server is reachable with the name 'jenkins'
- Jenkins credentials:

Username	Password
admin	welcome123

- The test deployment server is reachable by the name "test-server"

Lab Setup

On starting the lab, the following interface will be accessible to the user.



On choosing the first panel, **KALI CLI** will open in a new tab

```
root@kali-cli:~#
```

On selecting the second panel, **Jenkins** will open in a new tab



Welcome to Jenkins!

Sign in

☐ Keep me signed in

On selecting the middle panel, a web UI of **Gitlab** will open in a new tab.

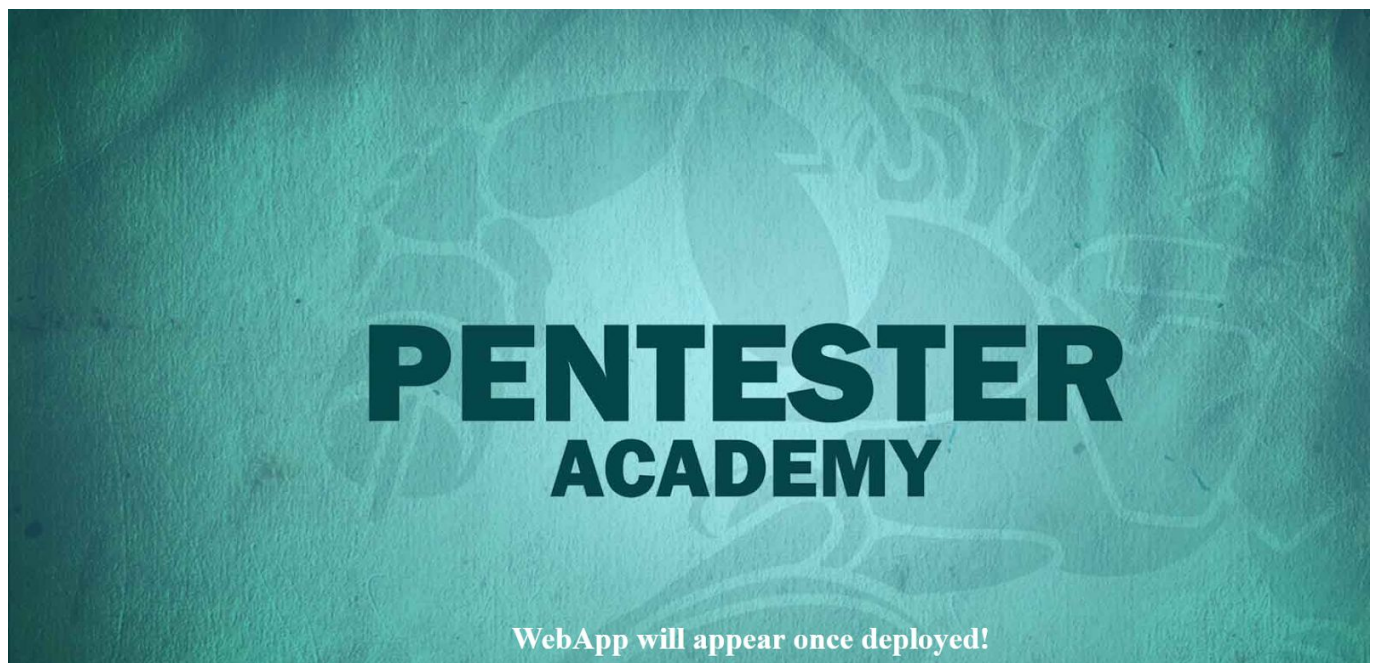
GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Sign in	Register
Username or email	
<input type="text"/>	
Password	
<input type="password"/>	
<input type="checkbox"/> Remember me	Forgot your password?
<input type="button" value="Sign in"/>	

And on selecting the right panel, a web UI of **Test Server** will open in a new tab.



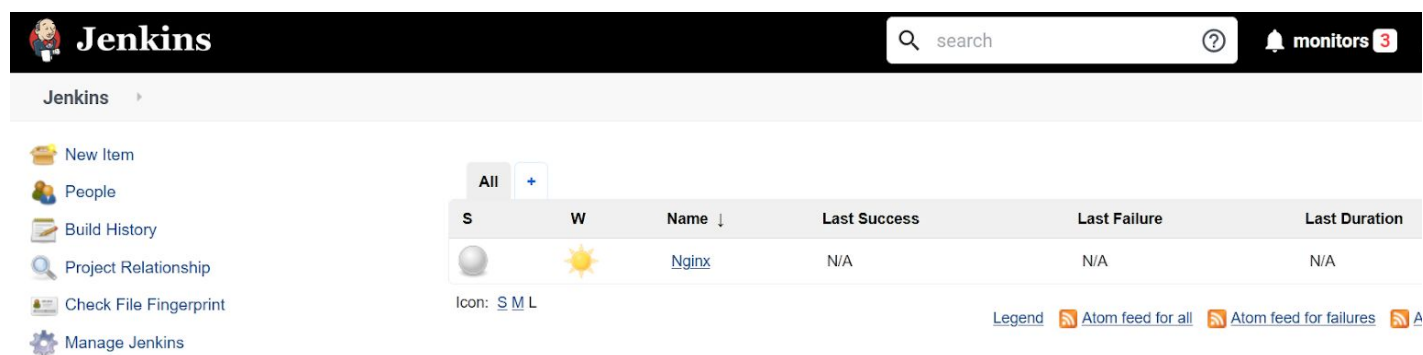
The page will reload until the test-server has started running the web service at port 80

Solution

Step 1: Login into the Jenkins, The credentials are provided in the challenge description.

Credentials:

- **Username:** admin
- **Password:** welcome123

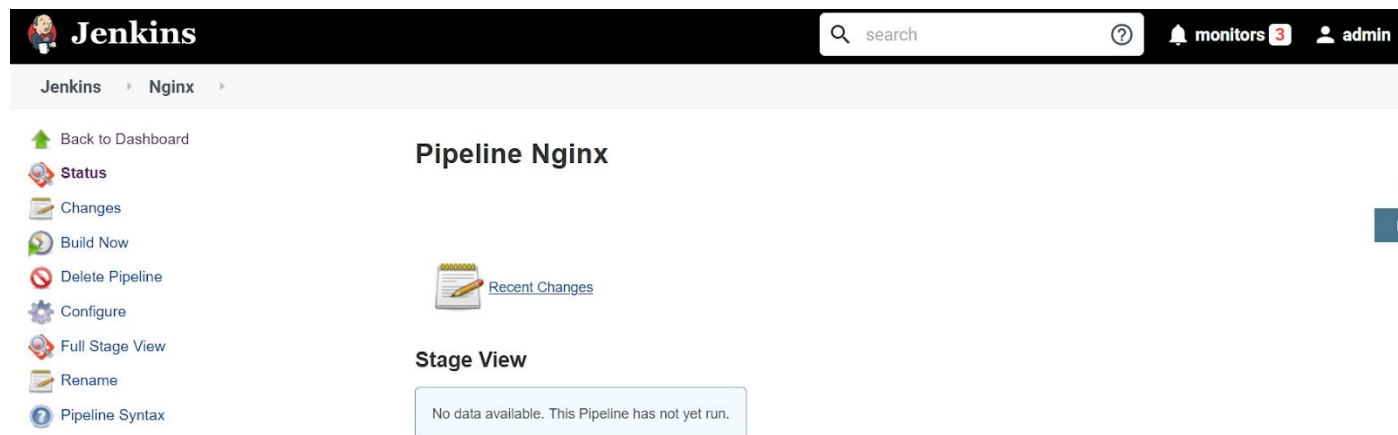


The screenshot shows the Jenkins dashboard. The top navigation bar includes the Jenkins logo, a search bar, a help icon, and a 'monitors 3' notification. The left sidebar contains links for 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', and 'Manage Jenkins'. The main content area displays a table of jobs. The table has columns for 'S' (Status), 'W' (Weather icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. A single job named 'Nginx' is listed with a status of 'S' (represented by a sphere icon) and a weather icon of a sun. Below the table, there are links for 'Icon: S M L' and 'Legend'. On the right, there are links for 'Atom feed for all' and 'Atom feed for failures'.

S	W	Name ↓	Last Success	Last Failure	Last Duration
		Nginx	N/A	N/A	N/A

There is only one Job (Nginx) available in the Jenkins instance.

Step 2: Click on the “Nginx” job.



The screenshot shows the Jenkins 'Pipeline Nginx' page. The top navigation bar includes the Jenkins logo, a search bar, a help icon, a 'monitors 3' notification, and a user profile icon labeled 'admin'. The left sidebar contains links for 'Back to Dashboard', 'Status', 'Changes', 'Build Now', 'Delete Pipeline', 'Configure', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. The main content area is titled 'Pipeline Nginx' and features a 'Recent Changes' section with a notepad icon. Below this, there is a 'Stage View' section with a message: 'No data available. This Pipeline has not yet run.'

This page is for “Pipeline Nginx” job, The Pipeline is appended in front of the Job name because this is a “Pipeline” type job in which it accepts a ‘Jenkinsfile’ which has all the commands and configuration of the pipeline.

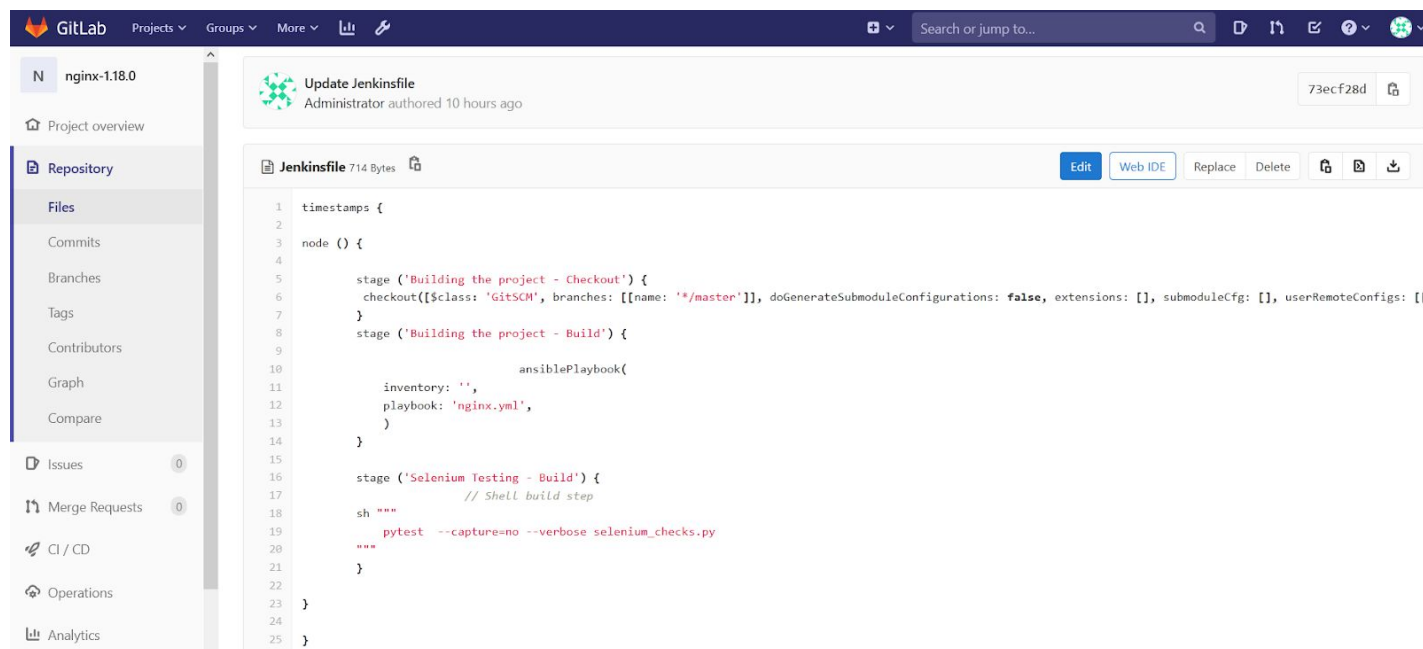
Step 3: Click on the “Configure” option to check the configuration of the Job.

This is the 'General' configuration tab for a Jenkins job. It features a 'Description' text area with a '[Plain text] Preview' link below it. A list of checkboxes is on the left: 'Discard old builds', 'GitHub project', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', and 'Execute concurrent builds if necessary'. On the right side of the tab, there are five circular help icons. An 'Advanced...' button is located at the bottom right.This is the 'Pipeline' configuration tab for a Jenkins job. The 'Definition' dropdown is set to 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git'. Under 'Repositories', the 'Repository URL' is 'http://gitlab/root/nginx-1.18.0.git', 'Credentials' is '- none -', and there are 'Add' and 'Advanced...' buttons. Below this, the 'Branches to build' section shows a 'Branch Specifier (blank for \'any\')' set to '*/*master' with an 'Add Branch' button. The 'Repository browser' is set to '(Auto)'. At the bottom, 'Additional Behaviours' has an 'Add' button, and 'Script Path' is set to 'Jenkinsfile'.

The “Pipeline” sections accept Jenkinsfile directly or a source such as Gitlab where the code and Jenkinsfile are stored for the project. The code is hosted on GitLab instance at this path “http://gitlab/root/nginx-1.18.0.git”



Step 3: Open the project on Gitlab and check the Jenkinsfile to build the pipeline.



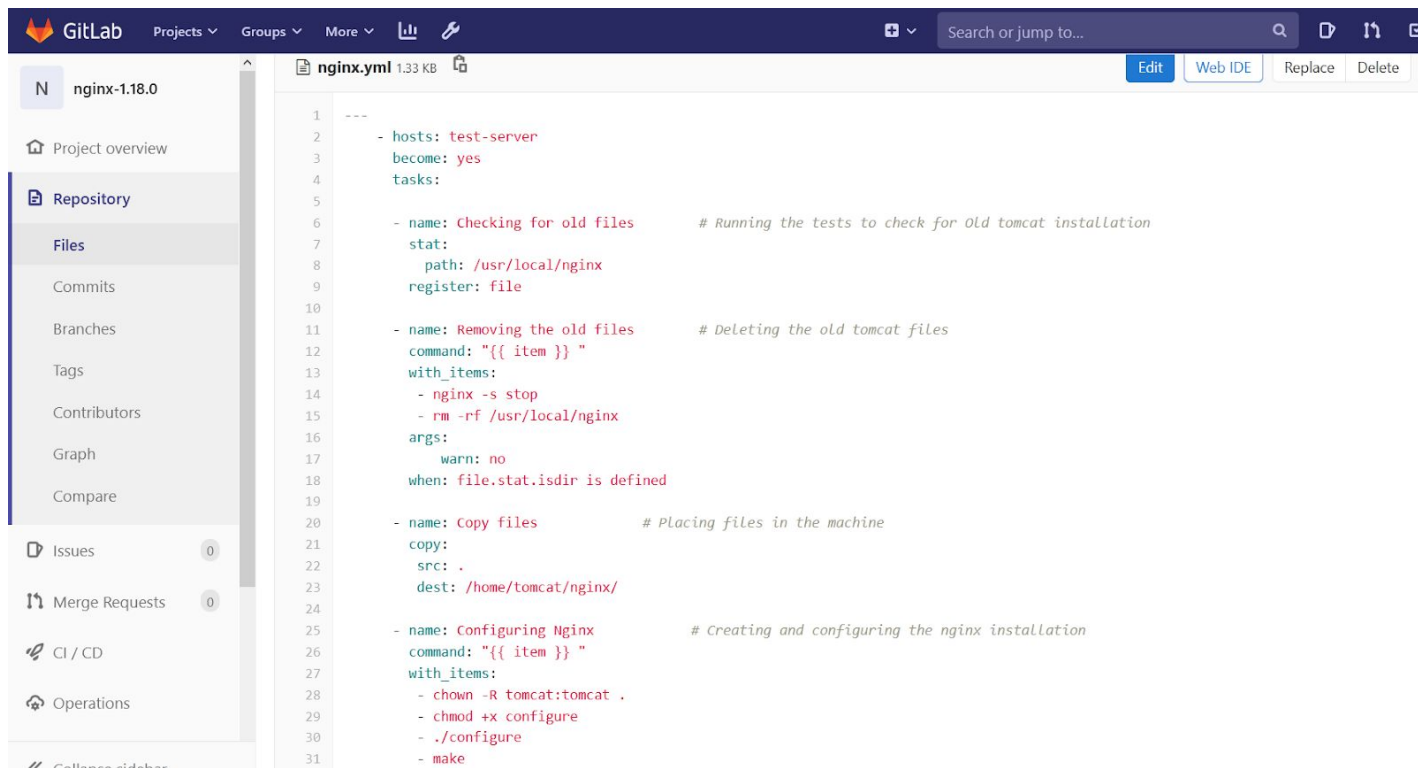
The file includes ‘stages’ that are a collection of steps, Each step performs a function which is explained below:-

Jenkinsfile Stages:

- **Building the project - Checkout:** In this stage, the git repository will be checked for any updates or commits. If commits are found in the repository then the new files will be fetched from remote repository.
- **Building the project - Build:** In this stage, the ansible will initiate the installation of Nginx on the remote server (test-server).
- **Selenium Testing - Build:** In this stage, the Jenkins will start checks on the newly deployed server to verify if the installation was successful or not.

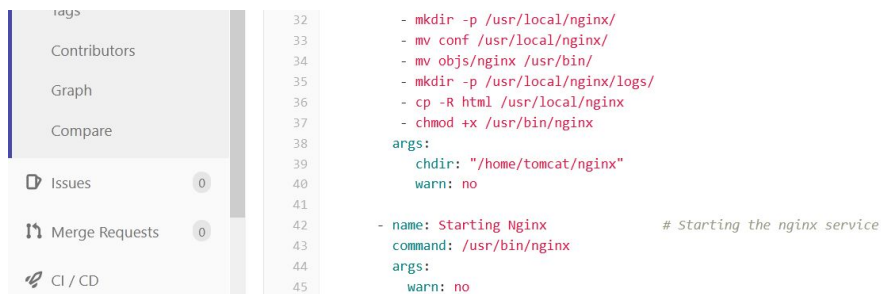
Note: The code for ansible (nginx.yml) and selenium testing (selenium_checks.py) are stored in the Gitlab repository itself.

Step 4: Check the Ansible configuration of the Nginx installation. The nginx.yml can be found in the root directory of the project.



The screenshot displays the GitLab web interface for a project named 'nginx-1.18.0'. The left sidebar shows the 'Repository' section with links to Files, Commits, Branches, Tags, Contributors, Graph, and Compare. The main content area shows the 'nginx.yml' file, which is 1.33 KB in size. The file content is as follows:

```
1 ---
2 - hosts: test-server
3   become: yes
4   tasks:
5
6     - name: Checking for old files          # Running the tests to check for Old tomcat installation
7       stat:
8         path: /usr/local/nginx
9       register: file
10
11    - name: Removing the old files          # Deleting the old tomcat files
12      command: "{{ item }}"
13      with_items:
14        - nginx -s stop
15        - rm -rf /usr/local/nginx
16      args:
17        warn: no
18      when: file.stat.isdir is defined
19
20    - name: Copy files                      # Placing files in the machine
21      copy:
22        src: .
23        dest: /home/tomcat/nginx/
24
25    - name: Configuring Nginx              # Creating and configuring the nginx installation
26      command: "{{ item }}"
27      with_items:
28        - chown -R tomcat:tomcat .
29        - chmod +x configure
30        - ./configure
31        - make
```



The image shows a screenshot of an Ansible configuration file. On the left, there is a sidebar with navigation links: 'Contributors', 'Graph', 'Compare', 'Issues' (with a count of 0), 'Merge Requests' (with a count of 0), and 'CI / CD'. The main area displays a list of tasks for installing Nginx, with line numbers 32 through 45 visible. The tasks include creating directories, moving files, copying HTML files, and starting the Nginx service.

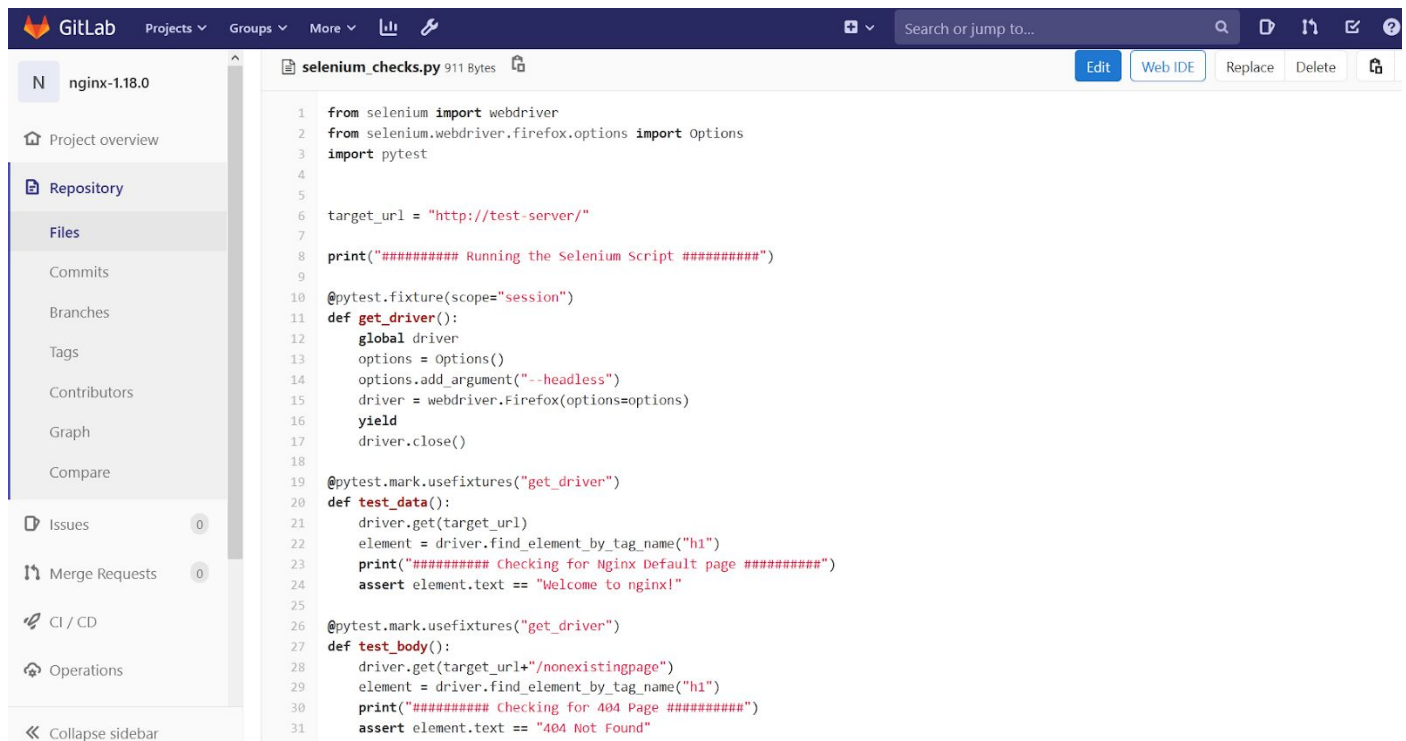
```
32 - mkdir -p /usr/local/nginx/
33 - mv conf /usr/local/nginx/
34 - mv objs/nginx /usr/bin/
35 - mkdir -p /usr/local/nginx/logs/
36 - cp -R html /usr/local/nginx
37 - chmod +x /usr/bin/nginx
38 args:
39   chdir: "/home/tomcat/nginx"
40   warn: no
41
42 - name: Starting Nginx # Starting the nginx service
43   command: /usr/bin/nginx
44   args:
45     warn: no
```

In the Ansible configuration provided above performs the following tasks:

Tasks:

- Check for the old Nginx installation on the server
- Remove the old installation from test-server
- Copy the new files into the server
- Install the Nginx into the remote server (test-server)
- Start the Nginx server.

Step 5: Check the Selenium tests. The tests will be used to determine if the Nginx is successfully installed or not.



```
1 from selenium import webdriver
2 from selenium.webdriver.firefox.options import Options
3 import pytest
4
5
6 target_url = "http://test-server/"
7
8 print("##### Running the Selenium Script #####")
9
10 @pytest.fixture(scope="session")
11 def get_driver():
12     global driver
13     options = Options()
14     options.add_argument("--headless")
15     driver = webdriver.Firefox(options=options)
16     yield
17     driver.close()
18
19 @pytest.mark.usefixtures("get_driver")
20 def test_data():
21     driver.get(target_url)
22     element = driver.find_element_by_tag_name("h1")
23     print("##### Checking for Nginx Default page #####")
24     assert element.text == "Welcome to nginx!"
25
26 @pytest.mark.usefixtures("get_driver")
27 def test_body():
28     driver.get(target_url+"/nonexistingpage")
29     element = driver.find_element_by_tag_name("h1")
30     print("##### Checking for 404 Page #####")
31     assert element.text == "404 Not Found"
```

Tasks:

- Open the index file and find "welcome to nginx!" in the source code.
- Open any non-existing page (i.e nonexistingpage) and find the string "404 not Found" which comes in the Nginx errors when the file does not exist.

Performing these tests on the target machine will ensure the Server is up and running on the test-server

Pipeline Execution

Step 1: Navigate to the Pipeline tab.

Defender Labs | Nginx [Jenkins] | selenium_checks.py - master

https://9bb241ae7cdb8571ncvyo9pwti2uip0ydu03nux6t.old-stager.attackdefensecloudlabs.com/job/Nginx/

Jenkins search ? monitors 3 admin

Jenkins > Nginx

Back to Dashboard

Status

Changes

Build Now

Delete Pipeline

Configure

Full Stage View

Rename

Pipeline Syntax

Build History trend

find

Atom feed for all Atom feed for failures

Pipeline Nginx

Recent Changes

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Step 2: Click on the “Build Now” button to start the Pipeline.

Jenkins search ? monitors 3 admin

Jenkins > Nginx

Back to Dashboard

Status

Changes

Build Now

Delete Pipeline

Configure

Full Stage View

Rename

Pipeline Syntax

Build History trend

find

Atom feed for all Atom feed for failures

Pipeline Nginx

Recent Changes

Stage View

Average stage times:	
Building the project - Checkout	582ms
Building the project - Build	1s

#1 Sep 23, 2020, 3:57 PM

Atom feed for all Atom feed for failures

#1 Sep 23, 2020, 21:27 No Changes

The page will automatically update and show the latest build information about the test-server.

The image shows the Jenkins web interface for a pipeline named 'Nginx'. The top navigation bar includes the Jenkins logo, a search bar, and a user profile 'admin'. The left sidebar contains various actions like 'Back to Dashboard', 'Status', 'Changes', 'Build Now', 'Delete Pipeline', 'Configure', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. The main content area is titled 'Pipeline Nginx' and shows a 'Recent Changes' section with a 'Stage View' table. The 'Build History' section shows a single build from Sep 23, 2020, at 3:57 PM. The 'Stage View' table displays the following data:

Stage	Building the project - Checkout	Building the project - Build	Selenium Testing - Build
Average stage times:	582ms	3min 30s	8s
Average full run time: ~3min 41s	582ms	3min 30s	8s

Below the table, there is a 'Permalinks' section and a 'Build History' section with a search bar and a list of builds.

The pipeline completed the execution successfully.

Step 3: Check the Test server.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

The default page of Nginx is displayed on the test-server which means the installation of Nginx was successful.

Learning

Working of a simple DevOps pipeline consisting of different components.