ATTACK DEFENSE

by PentesterAcademy

Name	DevOps Pipeline: Java WebApp
URL	attackdefense.com/challengedetails?cid=2064
Type	Pipeline Basics: Web Applications

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 a sample Java-based web application. 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 all parts: building/testing project with maven, deploying with Ansible, and dynamic testing with 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:

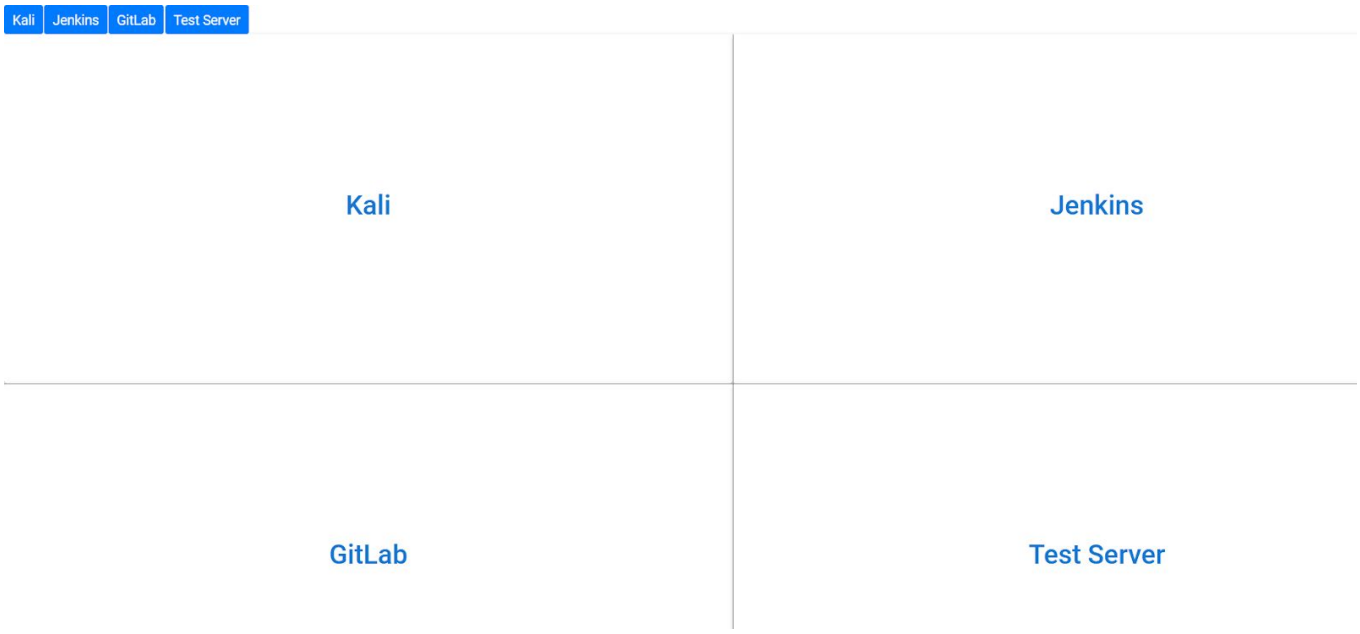
Username	Password
admin	welcome123

- The test deployment server is reachable by the name "test-server"
- Test server SSH credentials:

Username	Password
tomcat	password1

Lab Setup

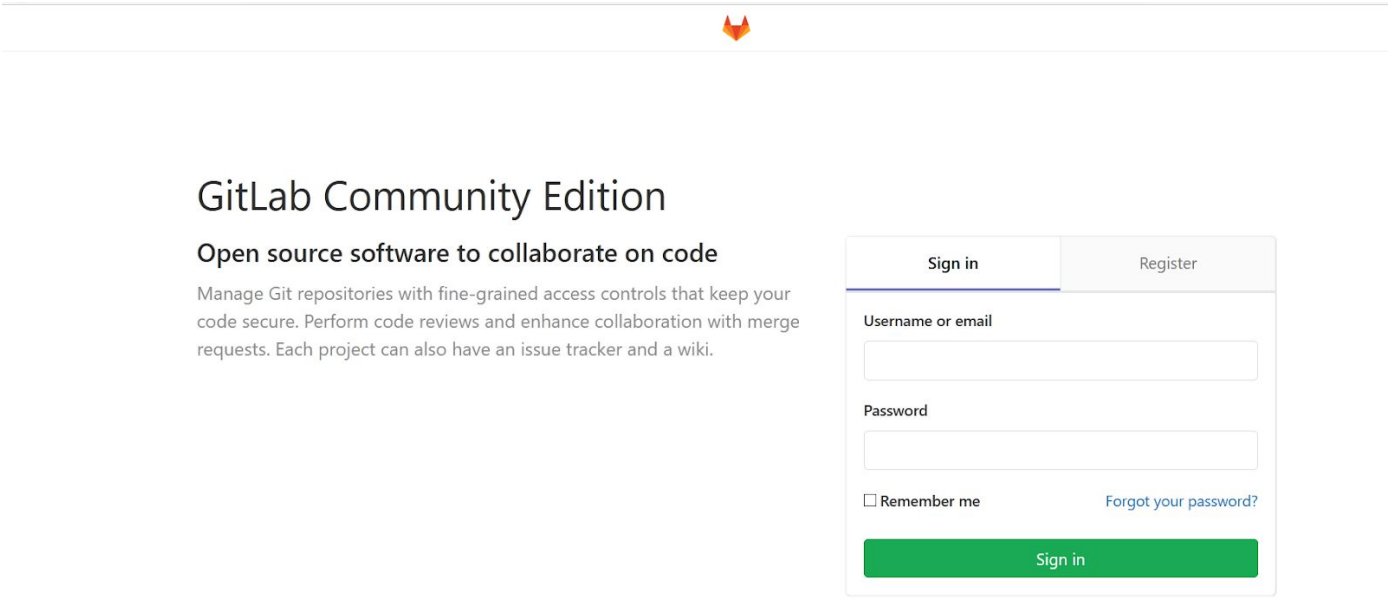
On starting the lab, the following interface will be accessible to the user.



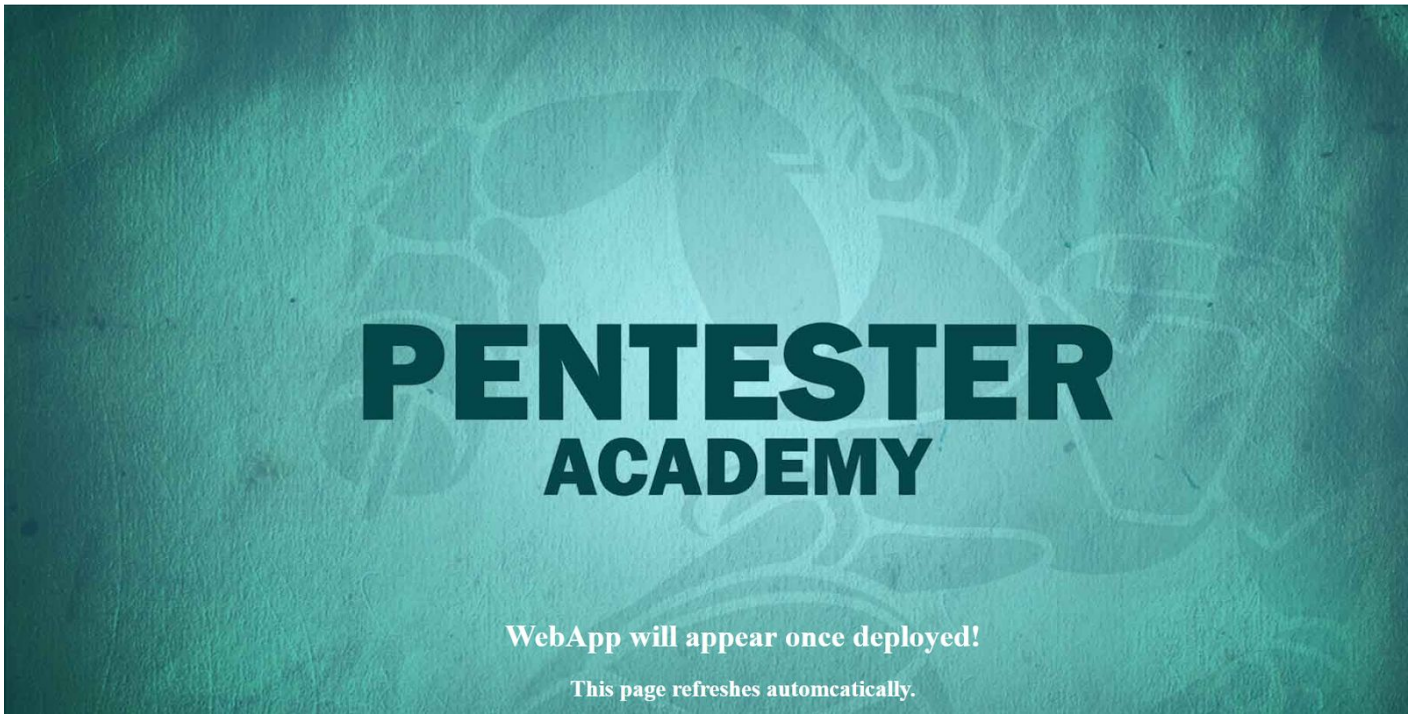
On choosing (clicking the text in the center) top left panel, **KALI CLI** will open in a new tab

Similarly on selecting the top right panel, a web UI of **Jenkins** will open in a new tab.

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



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



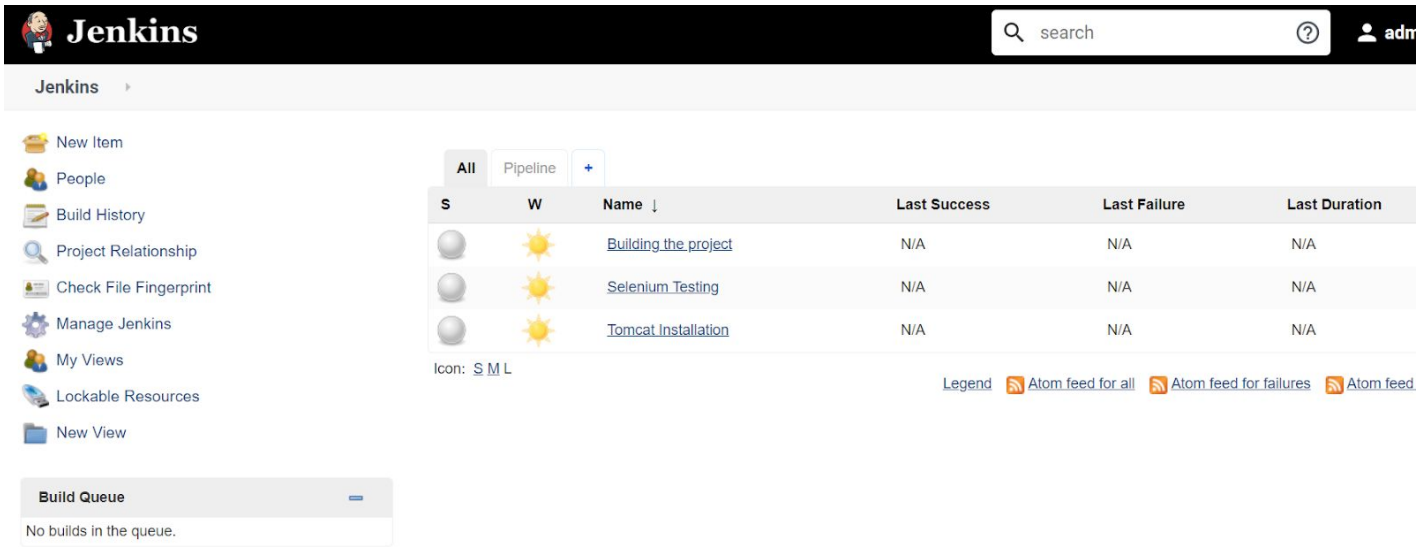
The page will reload until the test-server has started running the web service at port 8080

Solution

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

Credentials:

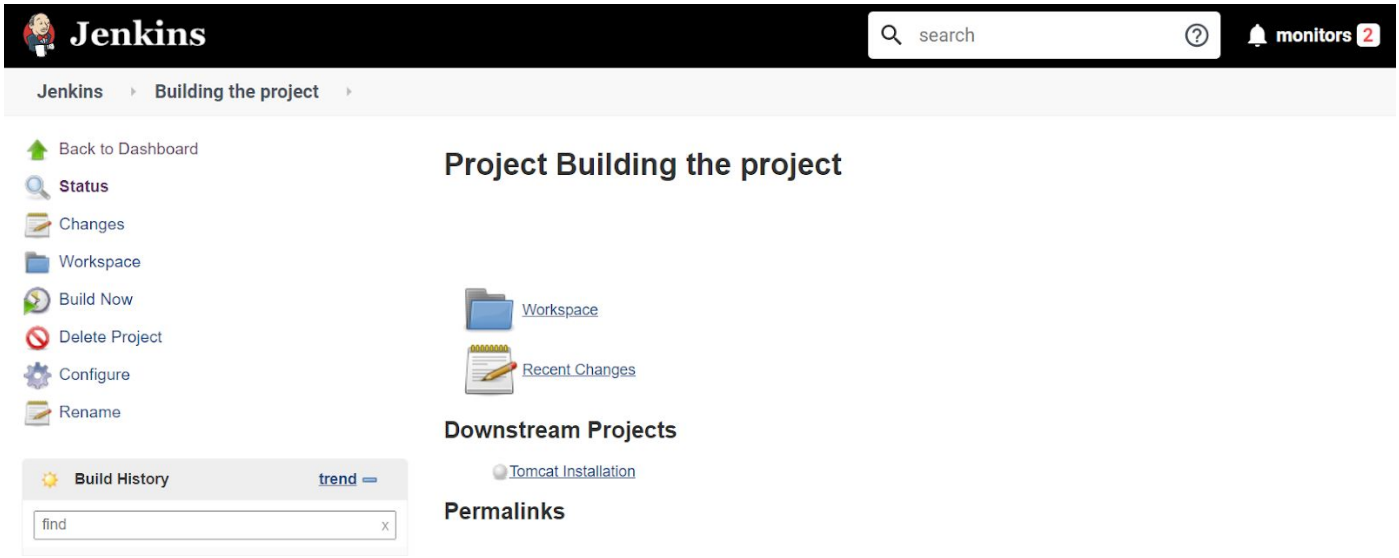
- **Username:** admin
- **Password:** welcome123



There are 3 jobs present in the Jenkins Interface, We will take one job at a time to study.

Job 1: Building the Project

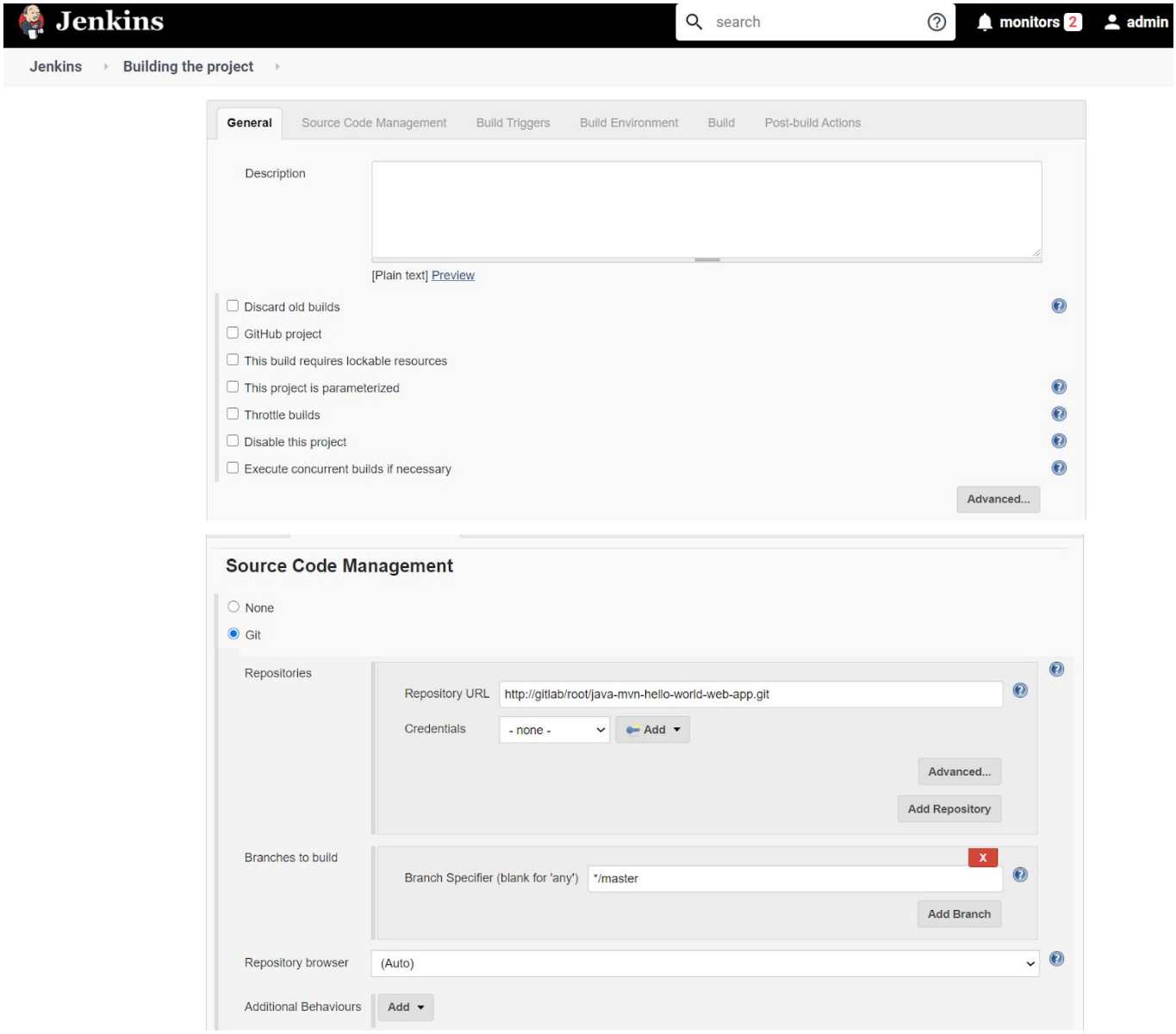
Step 1: Click on the “Building the project” job.



This page is for “Building the Project” job, from here the user can configure, Build, Delete the Job.

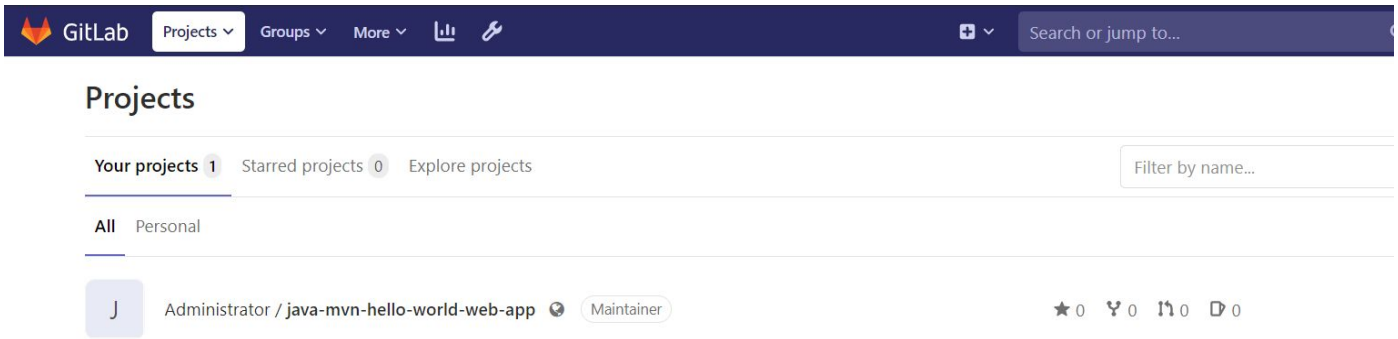
The workspace is a personal directory of this job where all the files which get downloaded during the build phase will reside.

Step 2: Click on the “Configure” option to check the configuration of the Job.

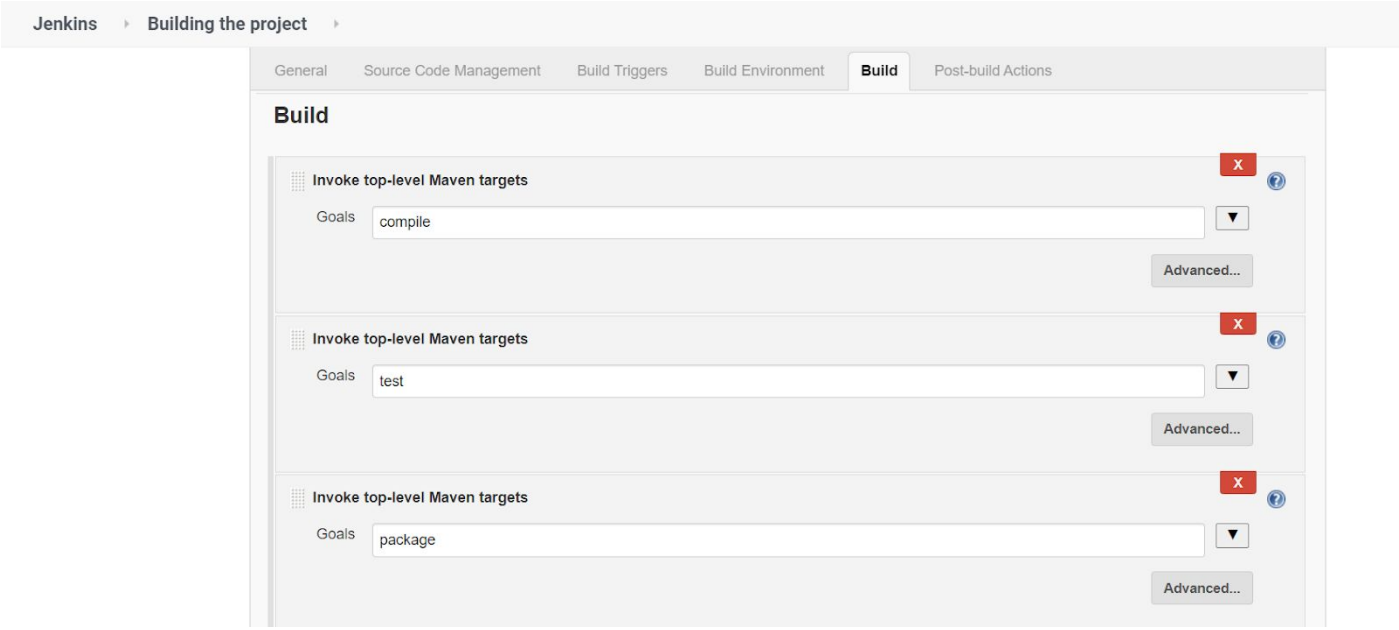


The Source Code Management (SCM) is the repository where the source code of the application is stored. This option will tell Jenkins to pull the code from the provided repository,

Which in this case is “http://gitlab/root/java-mvn-hello-world-web-app.git”, This is the repository which is stored in the Gitlab.



Step 3: Scroll down to check more options for the Job configuration



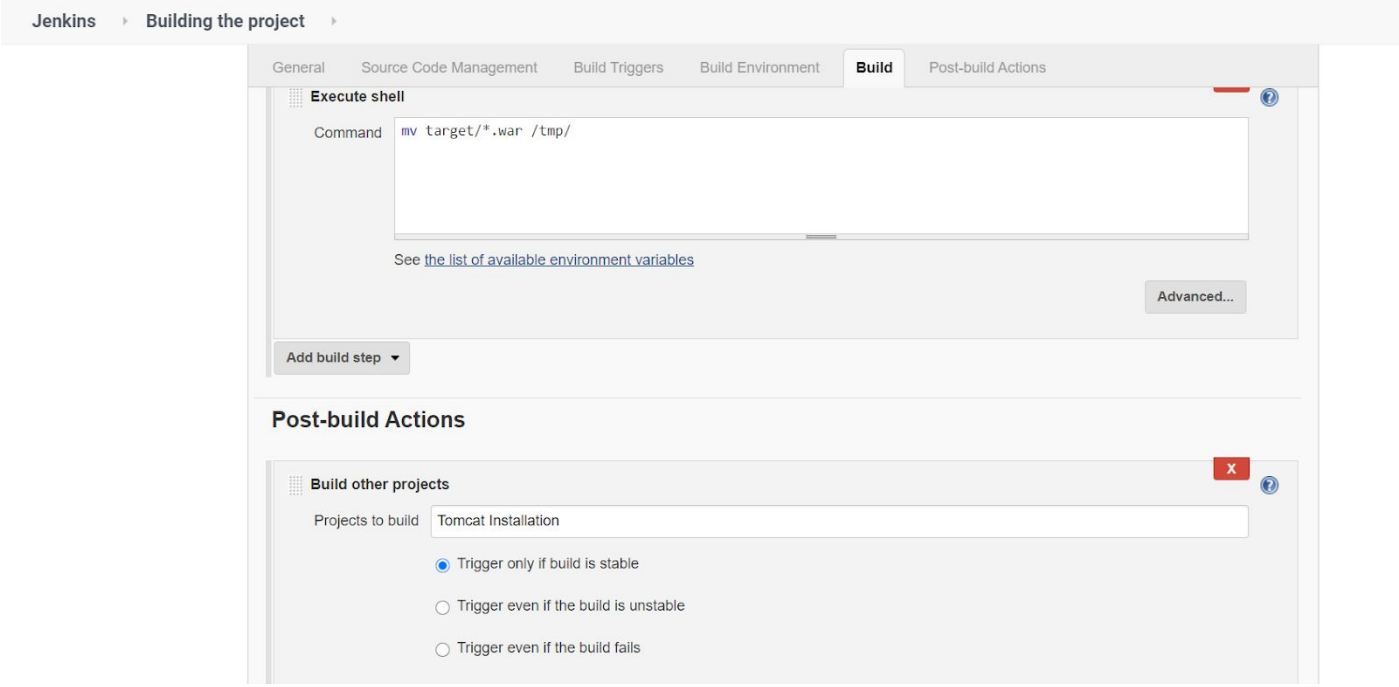
This phase is the “Build” phase where the application gets compiled or packages. In this case, we are using maven for compilation hence the maven plugin is installed and Goals are passed.

Goals are the tasks which are used to build and manage an application. Here, we have assigned 3 tasks such as compile, test, package.

Tasks:

- **Compile:** To compile the source code of the application
- **Test:** Testing the compiled source code using unit testing frameworks.
- **Package:** Package the compiled source code into a distributable format like JAR, WAR

Step 4: Checking the Shell commands and post-build actions of the job.

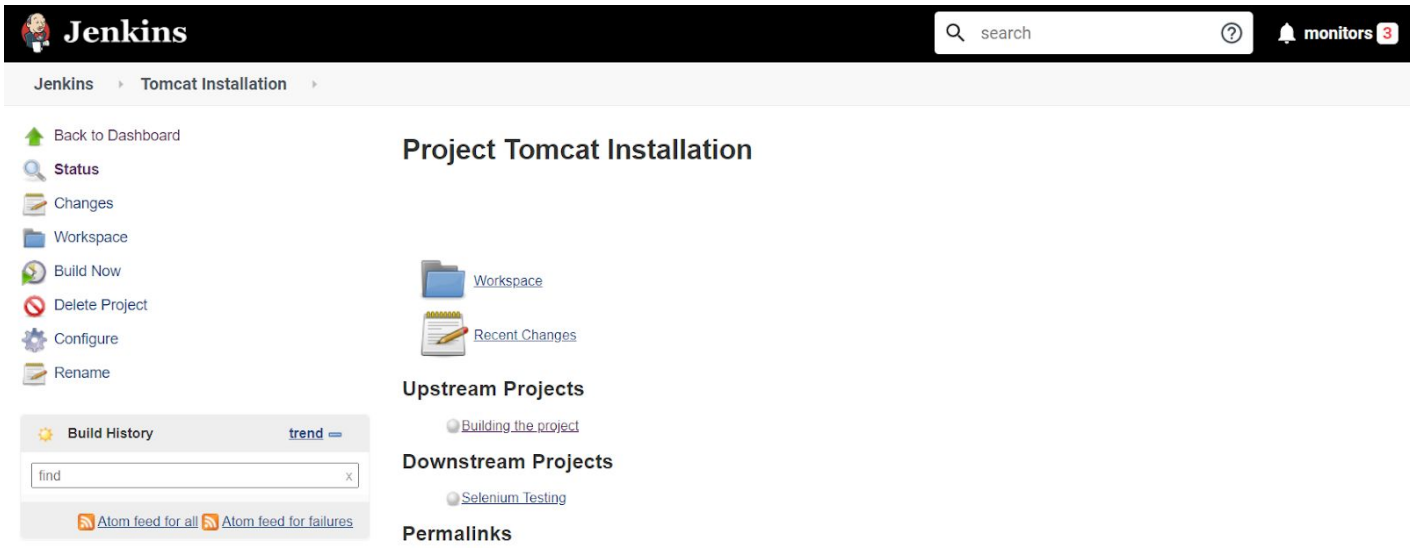


The “Execute Shell” section let the Jenkins execute certain sets of commands while building the project. In this case, The war file which is packaged by the maven will be moved to the temp (/tmp) directory.

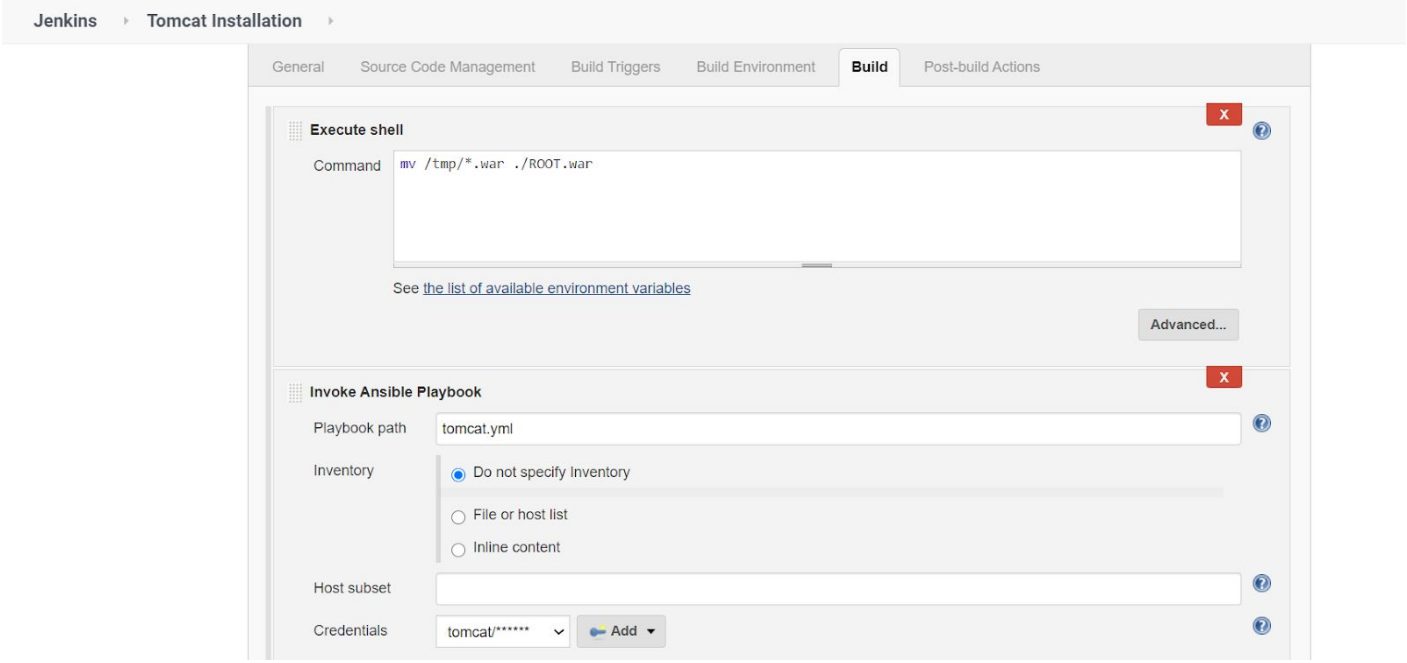
Post Build Actions: These are the steps which get executed after performing the Build phase. Here, the action is to execute another Job (Tomcat Installation) after finishing this one.

Job 2: Tomcat Installation

Step 1: Click on the “Tomcat Installation” job.



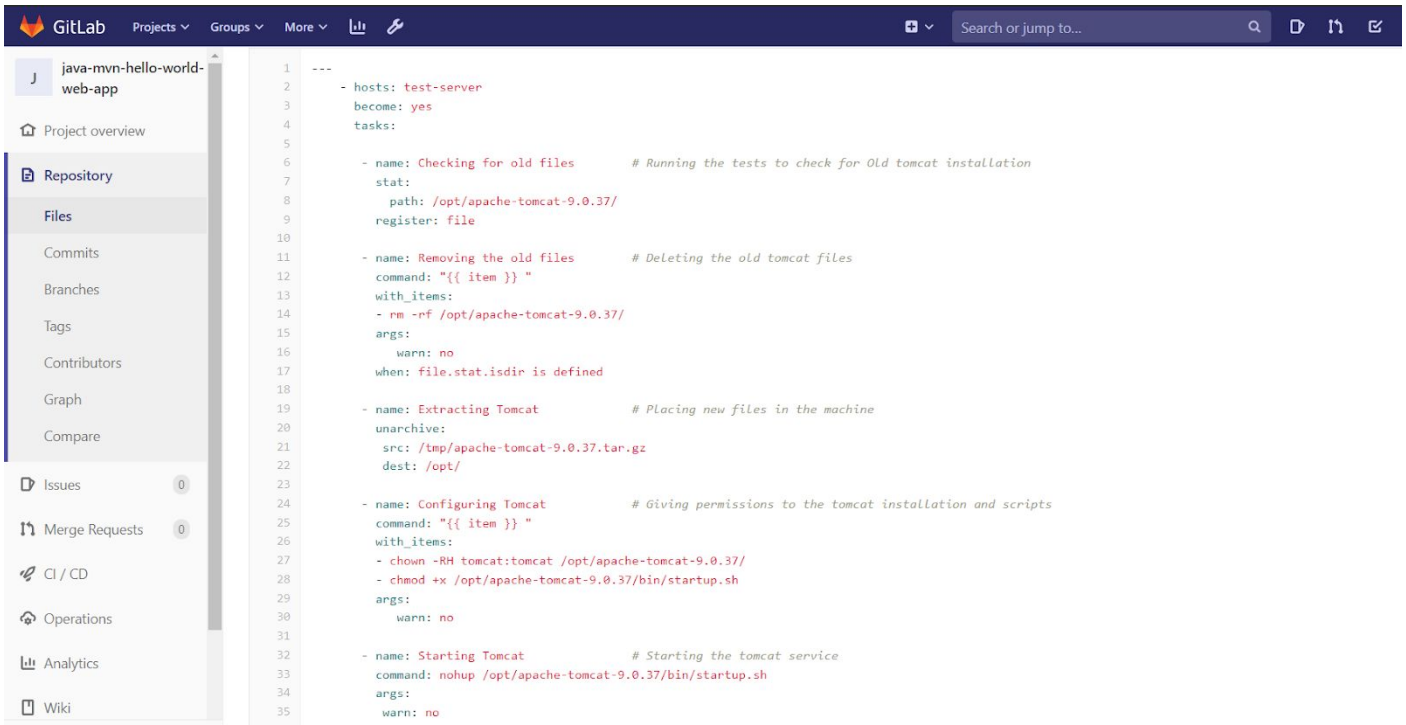
Step 2: Click on the “Configure” option to check the configuration of the Job.



The Shell command will rename any file ending with ".war" extension to ROOT.war.

The Ansible Plugin will load the "tomcat.yml" as the configuration to install tomcat on test-server

Step 3: Check the tomcat.yml file on Github repository.

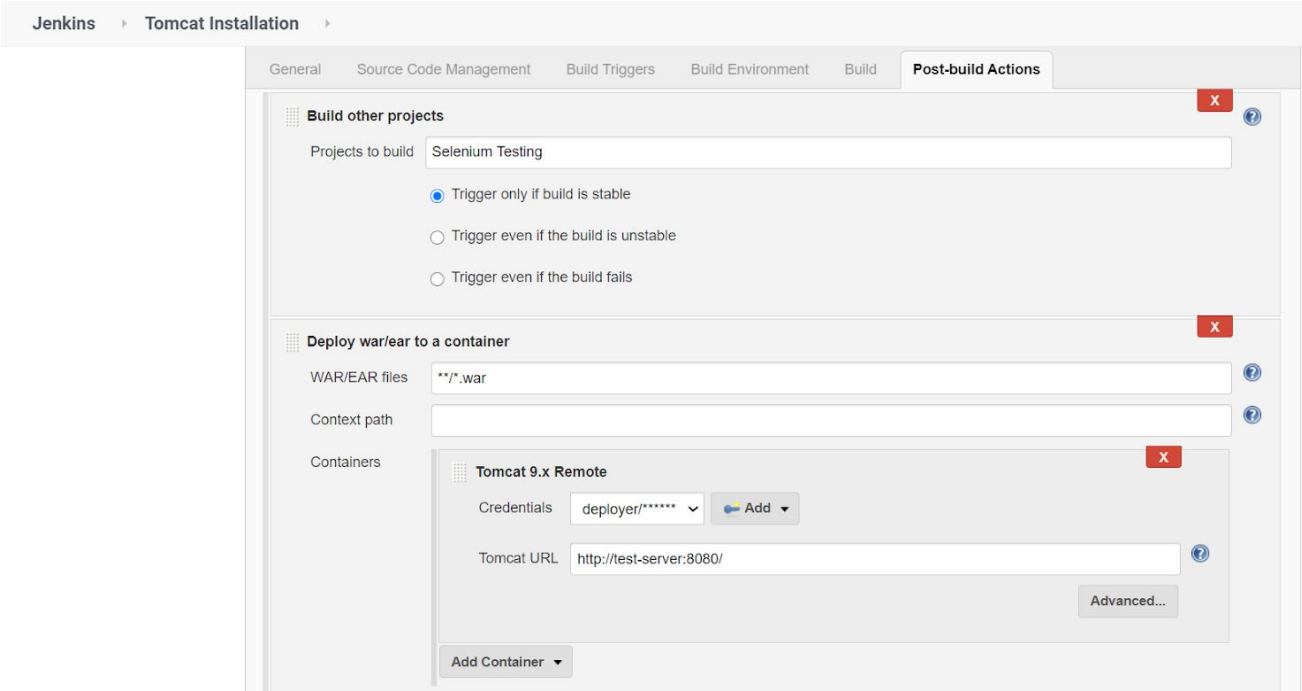



```
---
  - hosts: test-server
    become: yes
    tasks:

     - name: Checking for old files        # Running the tests to check for Old tomcat installation
       stat:
         path: /opt/apache-tomcat-9.0.37/
       register: file

     - name: Removing the old files        # Deleting the old tomcat files
       command: "{{ item }} "
       with_items:
       - rm -rf /opt/apache-tomcat-9.0.37/
       args:
          warn: no
       when: file.stat.isdir is defined

     - name: Extracting Tomcat             # Placing new files in the machine
       unarchive:
        src: /tmp/apache-tomcat-9.0.37.tar.gz
        dest: /opt/

     - name: Configuring Tomcat            # Giving permissions to the tomcat installation and scripts
       command: "{{ item }} "
       with_items:
       - chown -RH tomcat:tomcat /opt/apache-tomcat-9.0.37/
       - chmod +x /opt/apache-tomcat-9.0.37/bin/startup.sh
       args:
          warn: no

     - name: Starting Tomcat               # Starting the tomcat service
       command: nohup /opt/apache-tomcat-9.0.37/bin/startup.sh
       args:
        warn: no
```

According to the configuration, These steps will be performed by Ansible:

- Check the old configuration of tomcat
- Remove the old files from the test-server
- Copy and extract the new files from Jenkins server
- Assign permissions to the tomcat directory for "tomcat" user
- Start the tomcat server

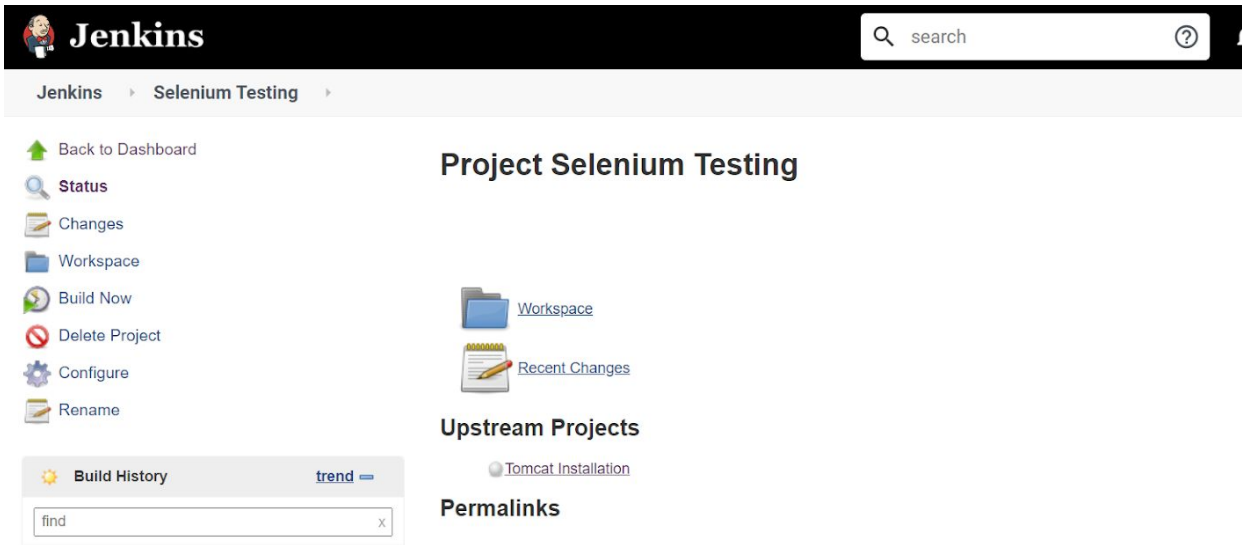
Step 4: Check the Post Build Section from the Job configuration



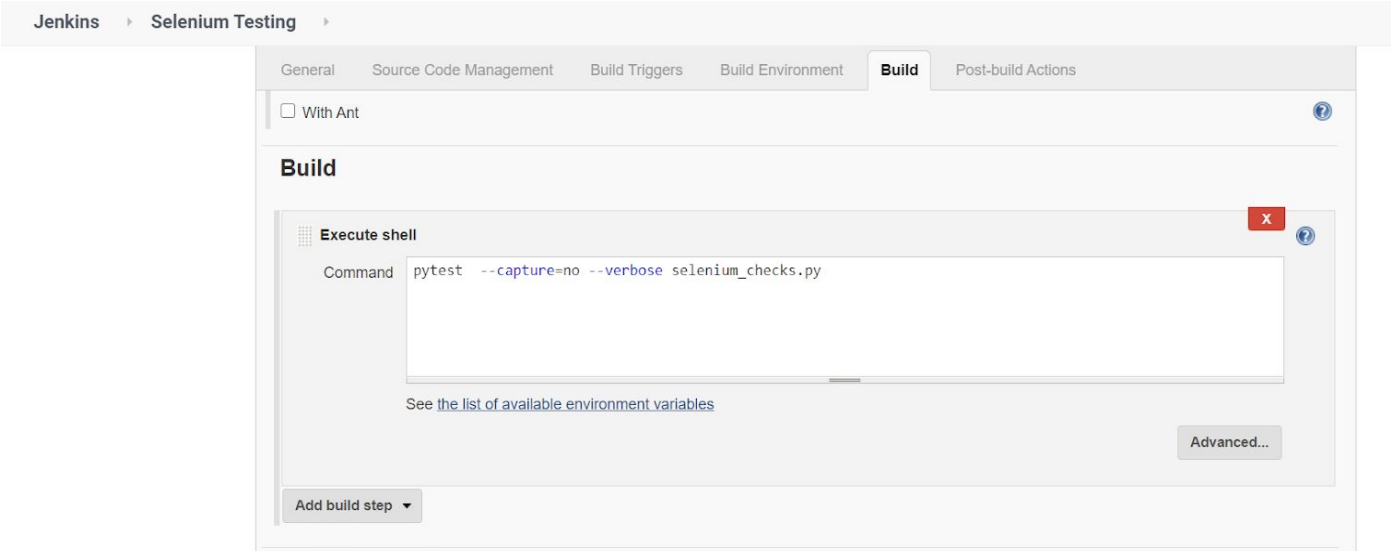
The “Deployer war/ear to a container” plugin will deploy the ROOT.war to the tomcat server and install the web application. The next Job which gets executed is “Selenium Testing”

Job 3: Selenium Testing

Step 1: Click on the “Selenium Testing” job.



Step 2: Click on the “Configure” option to check the configuration of the Job.



The “selenium_checks.py” contains test cases which will perform tests on the web application.

Step 3: Check the selenium_checks.py file in the gitlab instance

GitLab Projects Groups More

java-mvn-hello-world-web-app

Project overview

Repository

Files

Commits

Branches

Tags

Contributors

Graph

Compare

Issues 0

Merge Requests 0

CI / CD

Operations

selenium_checks.py 889 Bytes

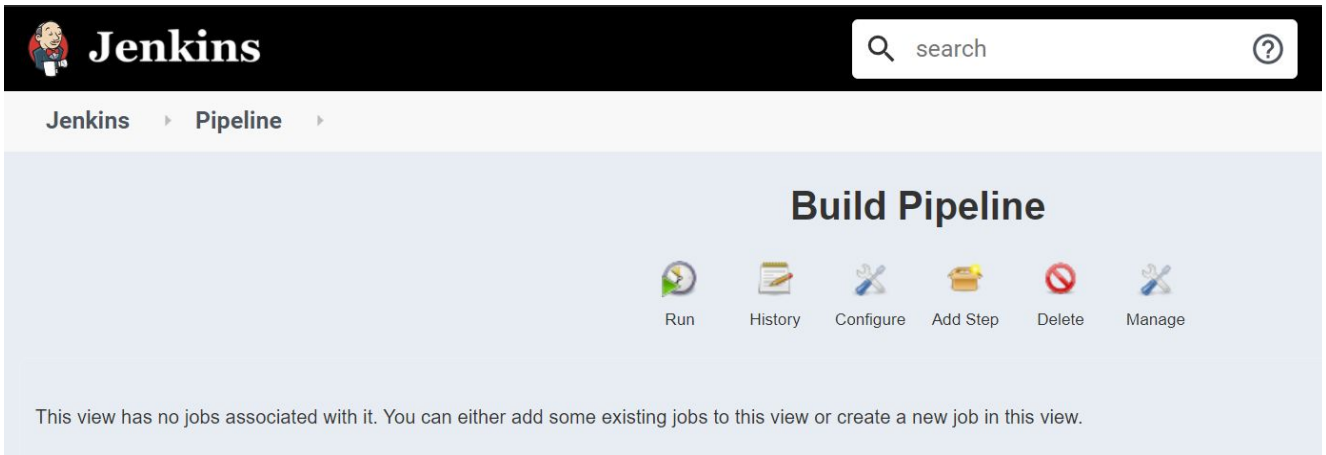
```python
from selenium import webdriver
from selenium.webdriver.firefox.options import Options
import pytest


target_url = "http://test-server:8080/"

print("########## Running the Selenium Script ##########")

@pytest.fixture(scope="session")
def get_driver():
    global driver
    options = Options()
    options.add_argument("--headless")
    driver = webdriver.Firefox(options=options)
    yield
    driver.close()

@pytest.mark.usefixtures("get_driver")
def test_data():
    driver.get(target_url)
    element = driver.find_element_by_tag_name("h2")
    print("########## Checking for Hello World on the page ##########")
    assert element.text == "Hello World!"

@pytest.mark.usefixtures("get_driver")
def test_body():
    element = driver.find_element_by_tag_name("p")
    print("########## Checking for Body content ##########")
```

The tests will do the following tasks:

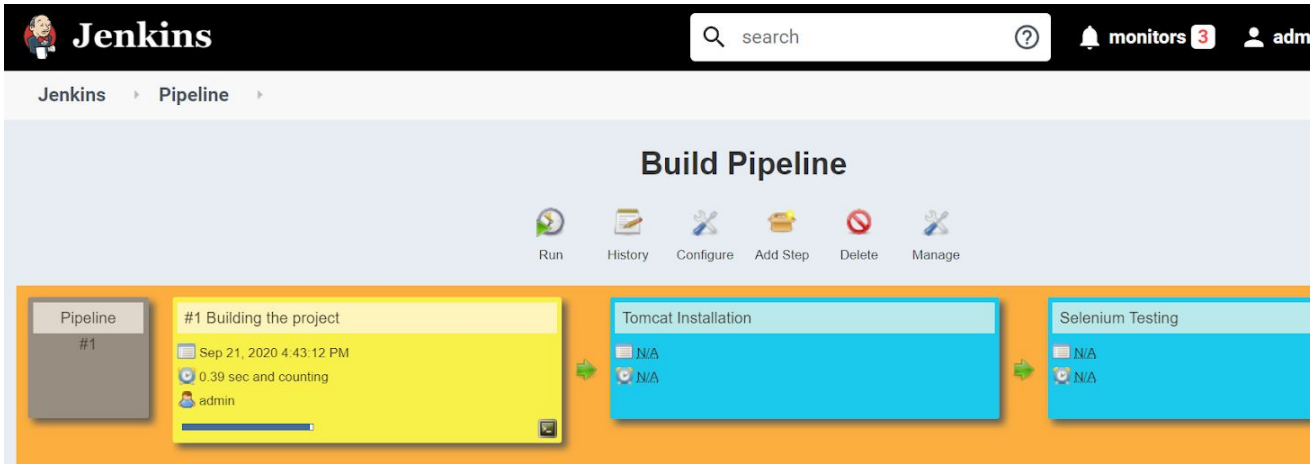
- Check if the “Hello World!” text exists in the h2 tag.
- Check if the “This is a sample application” exists under the ‘p’ tag

Pipeline Execution

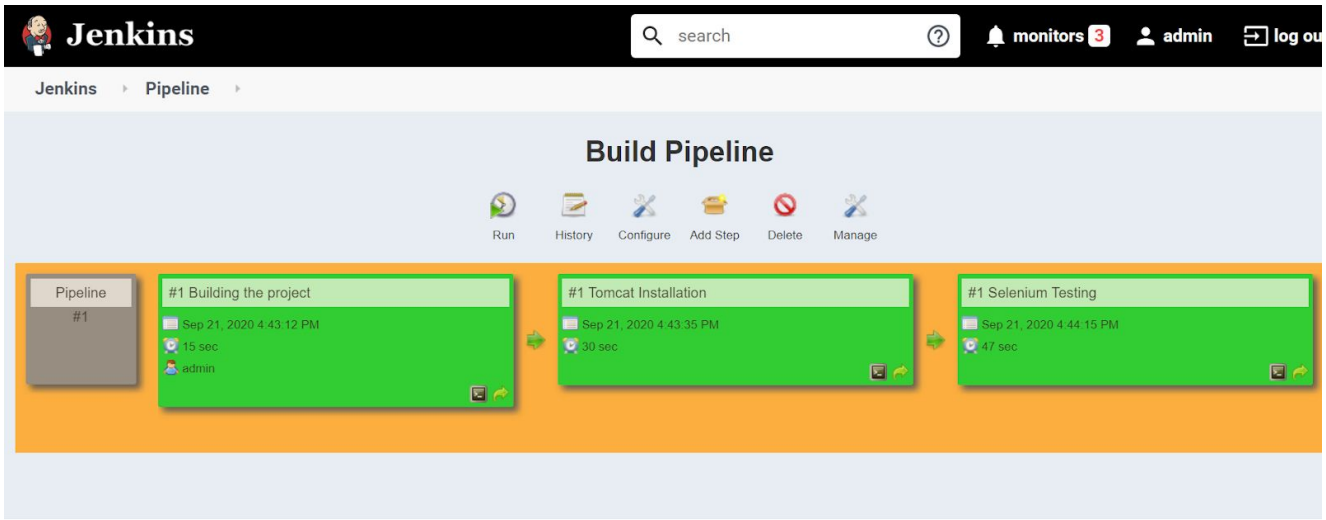
Step 1: Navigate to the Pipeline tab.



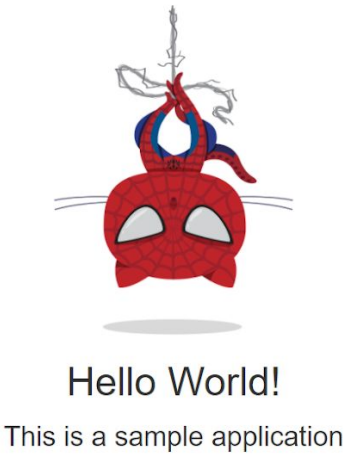
Step 2: Click on the Run button to start the Pipeline.



Reload the page to see the recent changes in the pipeline



Step 3: Navigate to the deployed website.



Learning

Working of a simple DevOps pipeline consisting of different components.