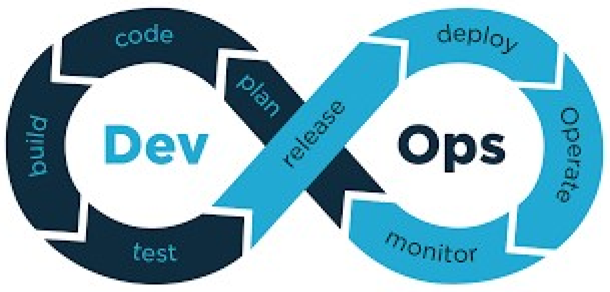
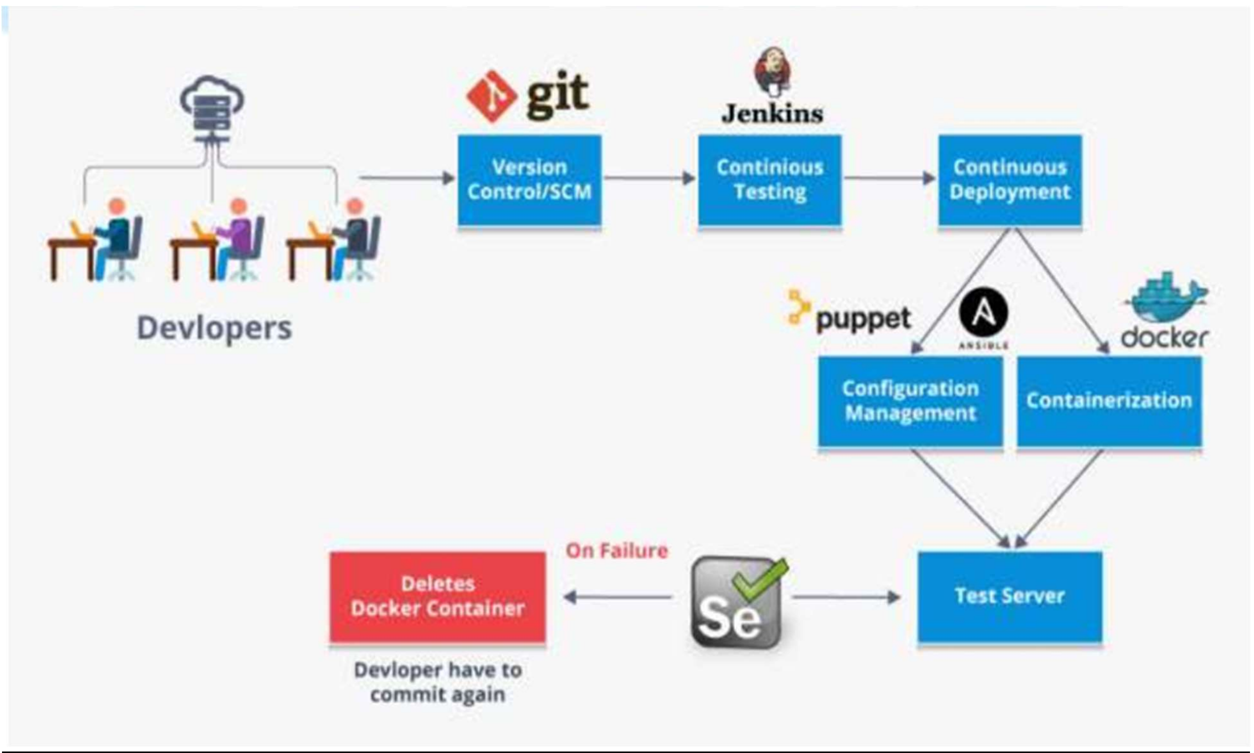


Devops Certification Training - Edureka

Complete CI/CD Pipeline – Certification Project

Vikhyat Dhamija



Problem Statement

Apple Bite Co. is using Cloud for one of their products. The project uses modular components, multiple frameworks and want the components to be developed by different teams or by 3rd party vendors.

The company's goal is to deliver the product updates frequently to production with High quality & Reliability. They also want to accelerate software delivery speed, quality and reduce feedback time between developers and testers.

As development progressed, they are facing multiple problems, because of various technologies involved in the project. Following are the problems:

- Building Complex builds is difficult
- Manual efforts to test various components/modules of the project
- Incremental builds are difficult to manage, test and deploy

To solve these problems, they need to implement Continuous Integration & Continuous Deployment with DevOps using following tools:

Git – For version control for tracking changes in the code files

Jenkins – For continuous integration and continuous deployment

Docker– For deploying containerized applications

Puppet/Ansible- Configuration management tools

Selenium- For automating tests on the deployed web application

Business challenge/requirement:

- As soon as the developer pushes the updated code on the GIT master branch, a new test server should be provisioned with all the required software.
- Post this, the code should be containerized and deployed on the test server.
- The deployment should then be tested using a test automation tool, and if the build is successful, it should be pushed to the prod server.
- All this should happen automatically and should be triggered from a push to the GitHub master branch.

This project will be about how to do deploy code to dev/stage/prod etc., just on a click of button. Link for the sample PHP application: https://github.com/edureka-devops/projCert.git

<u>Steps for executing the solution:</u>

•Use the Master VM for Jenkins, Ansible, Puppet, GIT etc.

•Use the Clean Ubuntu VM image provided in the “Edureka Setup Guide” for Jenkins Slave Node (Test Server)

Solution:

Two AWS EC-2 Instances were used – with one as a Master and one as a slave(Present in stopped state in my AWS account)

•Change the IP address of the VMs accordingly

•Add Build Pipeline Plugin and Post-build task plugin to Jenkins on the master VM

•Install python, open ssh-server, and git on the slave node manually

Solution:

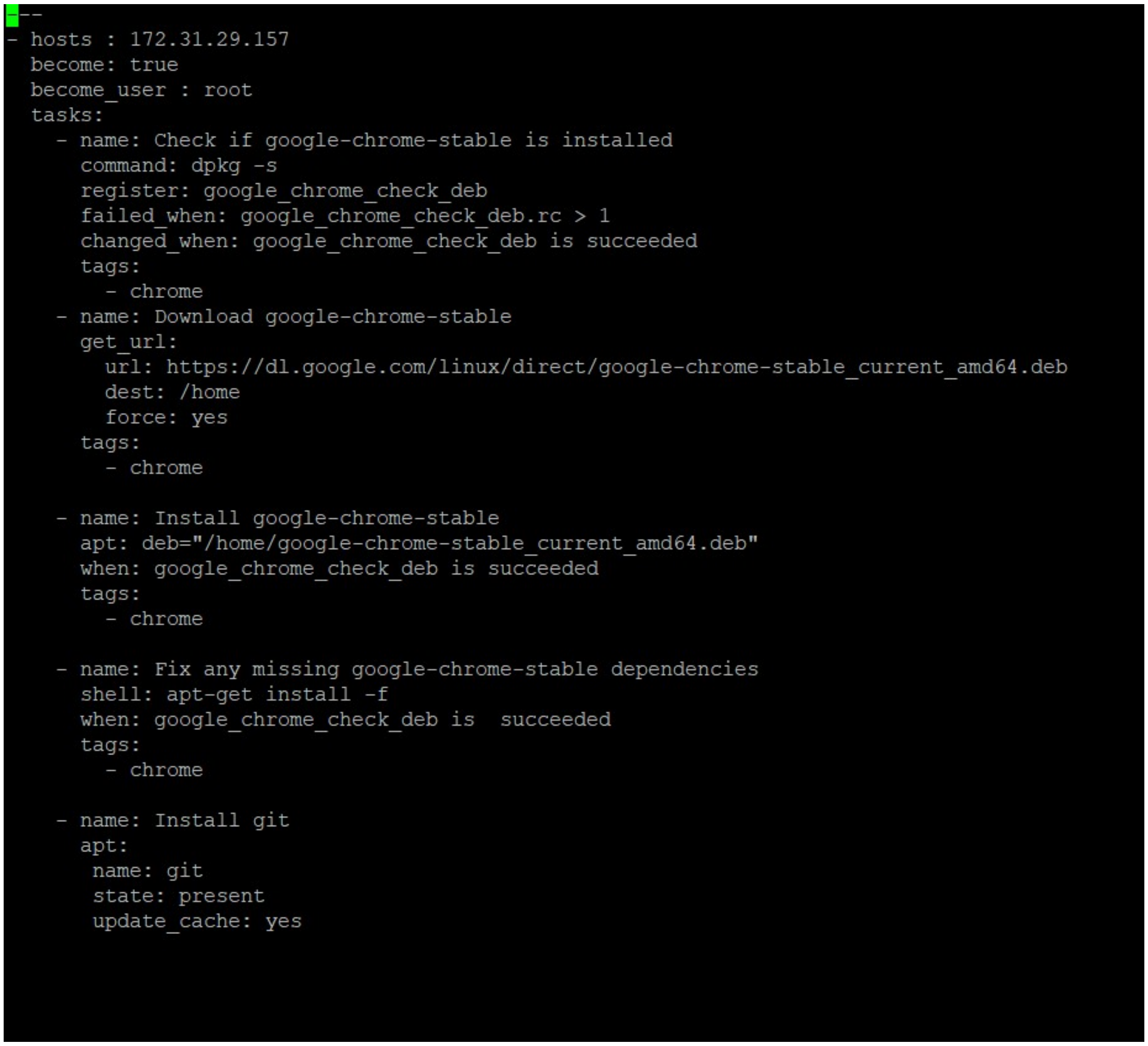
***Two Amazon EC2 instances were configured as Master Slave with Master provided with all the tools Jenkins, Ansible, Puppet, GIT etc*.**

• Set up the necessary tools such as git, chrome driver(selenium), chromium browser(selenium) on the slave node through Ansible

Solution:

Slave Amazon EC2 instances was configured with required tools using ansible:

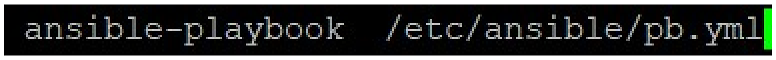
Chrome Browser and Git installed through Ansible Playbook



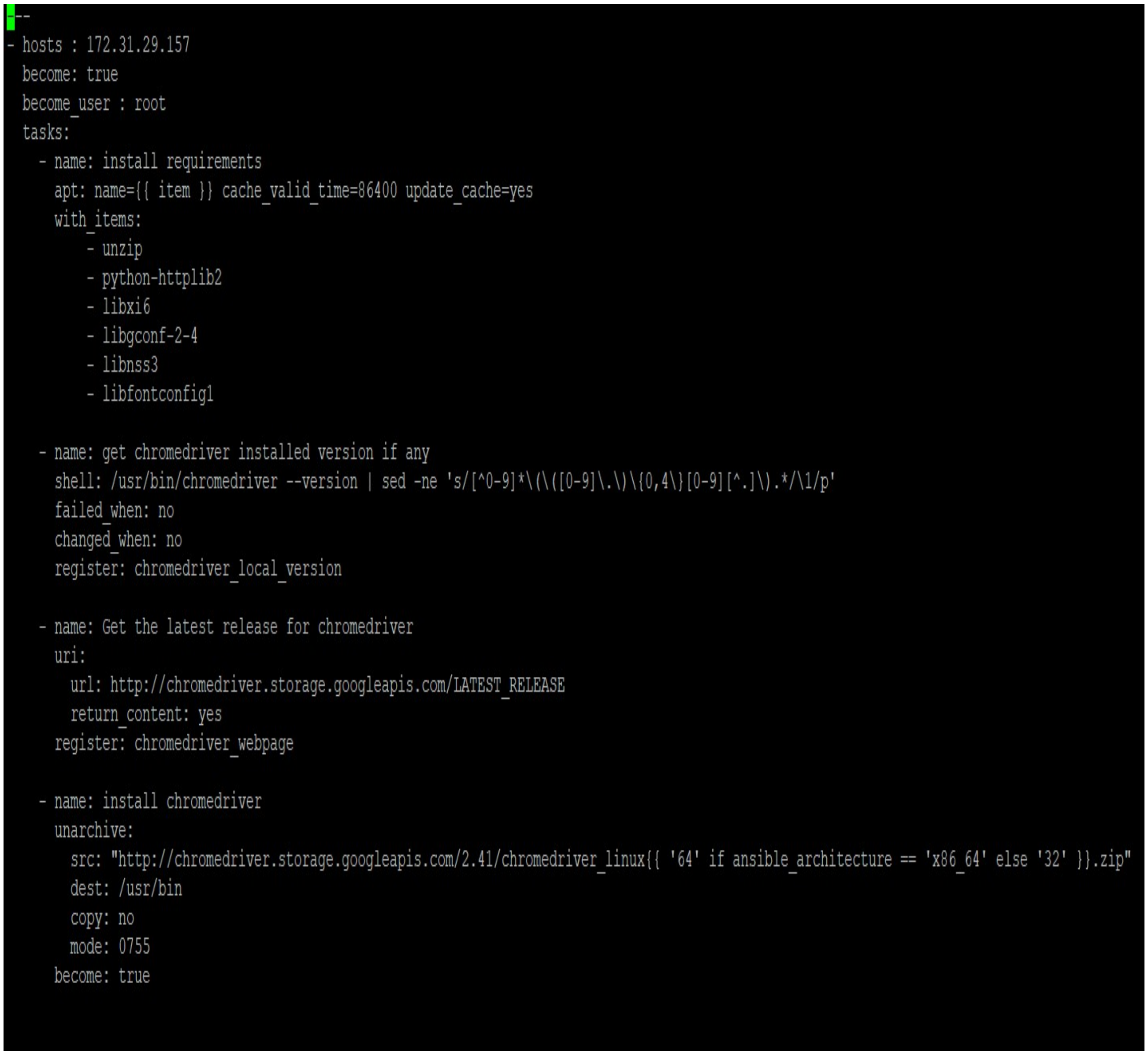
```
---
- hosts : 172.31.29.157
  become: true
  become_user : root
  tasks:
    - name: Check if google-chrome-stable is installed
      command: dpkg -s
      register: google_chrome_check_deb
      failed_when: google_chrome_check_deb.rc > 1
      changed_when: google_chrome_check_deb is succeeded
      tags:
        - chrome
    - name: Download google-chrome-stable
      get_url:
        url: https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
        dest: /home
        force: yes
      tags:
        - chrome

    - name: Install google-chrome-stable
      apt: deb="/home/google-chrome-stable_current_amd64.deb"
      when: google_chrome_check_deb is succeeded
      tags:
        - chrome

    - name: Fix any missing google-chrome-stable dependencies
      shell: apt-get install -f
      when: google_chrome_check_deb is  succeeded
      tags:
        - chrome

    - name: Install git
      apt:
       name: git
       state: present
       update_cache: yes
```

Playbook for the installation of the google-chrome and git

Master:



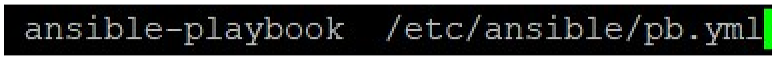
```
ansible-playbook  /etc/ansible/pb.yml
```



```
---
- hosts : 172.31.29.157
  become: true
  become_user : root
  tasks:
    - name: install requirements
      apt: name={{ item }} cache_valid_time=86400 update_cache=yes
      with_items:
          - unzip
          - python-httplib2
          - libxi6
          - libgconf-2-4
          - libnss3
          - libfontconfig1

    - name: get chromedriver installed version if any
      shell: /usr/bin/chromedriver --version | sed -ne 's/[^0-9]*\(\([0-9]\.\)\{0,4\}[0-9][^.]\).*/\1/p'
      failed_when: no
      changed_when: no
      register: chromedriver_local_version

    - name: Get the latest release for chromedriver
      uri:
        url: http://chromedriver.storage.googleapis.com/LATEST_RELEASE
        return_content: yes
      register: chromedriver_webpage

    - name: install chromedriver
      unarchive:
        src: "http://chromedriver.storage.googleapis.com/2.41/chromedriver_linux{{ '64' if ansible_architecture == 'x86_64' else '32' }}.zip"
        dest: /usr/bin
        copy: no
        mode: 0755
      become: true
```

Chrome Driver :

Play book for installation of the chrome Driver for chrome browser



```
ansible-playbook  /etc/ansible/pb.yml
```

Further Steps:

•Use the image Devops edu /webapp and add your PHP website to it using a Docker file

•Create a Selenium Test for your PHP website. It should click on “About” and verify the text written in it. This will conclude the website is deployed and is running fine.

Note that : Selenium Test file has been shown ahead.

•Push the PHP website, Docker file and Selenium JAR to a git repository

Solution:

1. ***https://github.com/edureka-devops/projCert was cloned and then its contents were brought to the local machine.***

    a. ***Gitrepo folder created   mkdir gitrepo***
    b. ***Git init   // To initiate git***
    c. ***Git clone https://github.com/edureka-devops/projCert***

2. ***The contents were moved to the /home/ubuntu local directory and created Dockerfile and Jar file was also present there.***

    a. ***Git init***
    b. ***Git add Dockerfile //adding to the staging area***
    c. ***Git commit -m Dockerfile //adding the docker file to my local server***
    d. ***Git push https://github.com/vikhyat-dhamija/devops master //pushing it to the remote repository***

Note : Jar file was pushed through window machine as eclipse was installed in the window machine and the remote desktop was used for pushing the jar file on the master.

Similarly, the whole website folder containing the whole website was pushed to remote repository,

a. ***Git init***
b. ***Git add website/****
c. ***Git commit -m "website"***
d. ***Git push https://github.com/vikhyat-dhamija/devops master //pushing it to the remote repository***

Similarly, the selenium jar file for testing purpose was pushed to the remote repository.

Automation of Tasks through Jenkins

Below tasks should be automated through Jenkins by creating a pipeline:

1.Install and configure puppet agent on the slave node (Job 1)

Puppet agent is already installed in the slave node/test server already so in the Jenkins job1 named as project_job_1 with two shell commands to be executed in the slave node :

sudo systemctl start puppet ;
sudo systemctl enable puppet ; -- if you want puppet to start on boot also

These commands have made the puppet up and running in the slave.

2.Sign the puppet certificate on master using Jenkins (Job 2)

New job is created with the following command to be executed for the master to sign the certificate send by the slave.

sudo puppet cert sign -a ;

Note we can pass the slave hostname as the parameter to sign the certificate of that slave instead of just passing -a to sign certificates for all.

3.Trigger the puppet agent on test server to install docker (Job 3)

sudo puppet agent –test

Content of the site.pp file for application of the catalog by the slave on trigger:

```
node default{
  class{'dock':}
}

class dock{
  package{'docker':
    ensure => 'present',
  }
  service{'docker':
    ensure => 'running',
    enable => true
  }
}
```

This file leads to the installation of the docker in the slave machine as desired.

4.Pull the PHP website, Docker file and Selenium JAR from your git repo and build and deploy your PHP docker container. After this test the deployment using Selenium JAR file. (Job 4)

```
sudo git init ;
[ -d ./devops_w ] && sudo rm -r ./devops ;
sudo git clone https://github.com/vikhyat-dhamija/devops ;
[ -d ./devops_w ] || mkdir ./devops_w ;
sudo cp -r ./devops ./devops_w ;
sudo docker stop $(sudo docker ps -aq) | wc -l ;
sudo docker build -t webimage ./devops_w/devops ;
sudo docker run --rm -itd -p 32732:80 webimage;
```

sudo java -jar ./devops_w/devops/selenium_test.jar ;

Here in this job in the post build actions the trigger to Project_job_5 was added with Parameterized Trigger (Failed one) which issues trigger to Project_job_5 when Job4 failed.

5.If Job 4 fails, delete the running container on Test Server.

sudo docker stop $(sudo docker ps -aq) | wc -l ;

Trigger in the pipeline was configured with Project_Job_1:

1. Webhook was inserted in the GitHub repository which can be seen in the git repository in the settings section:

   https://github.com/vikhyat-dhamija/devops

Note that the Jenkins URL was inserted wherein json data will be sent with regards to any push and pull in the repository that triggers the pipeline.

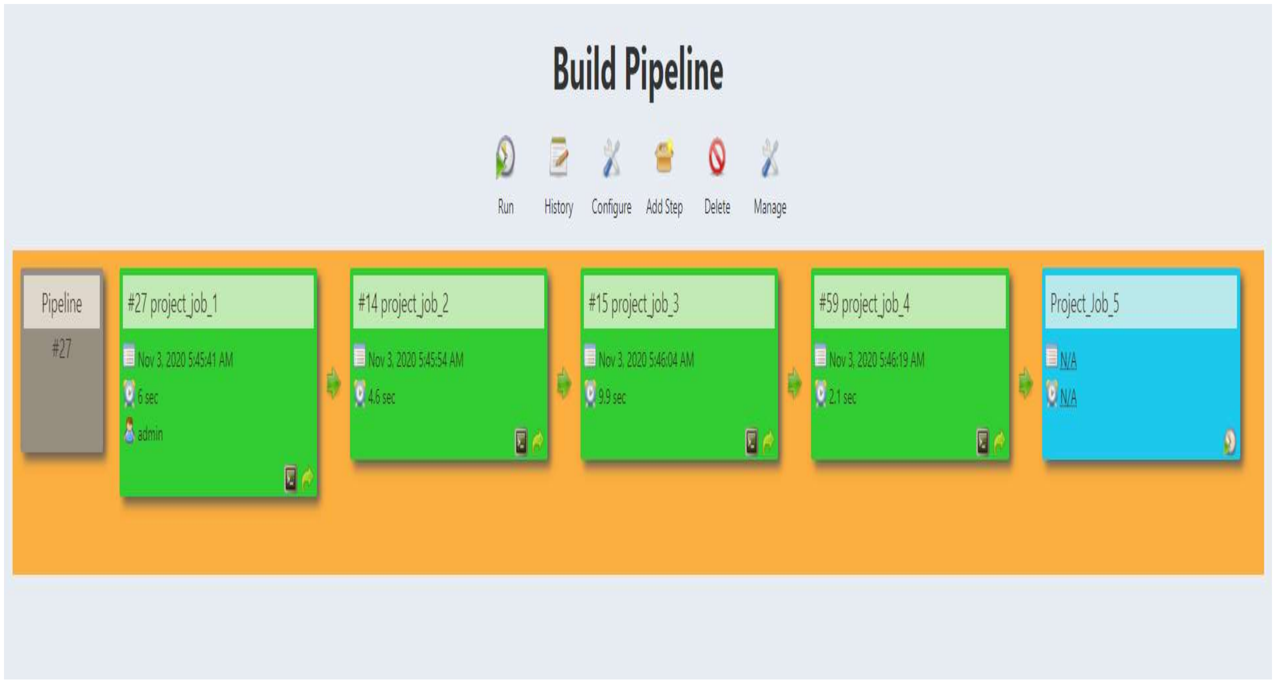
2. Jenkins Job 1 was configured:
   Build Trigger: GithubPollSCM changes
   Build Code: GitHub repository : https://github.com/vikhyat-dhamija/devops

   These were the changes performed over the Job_1 in order to get triggered with GitHub

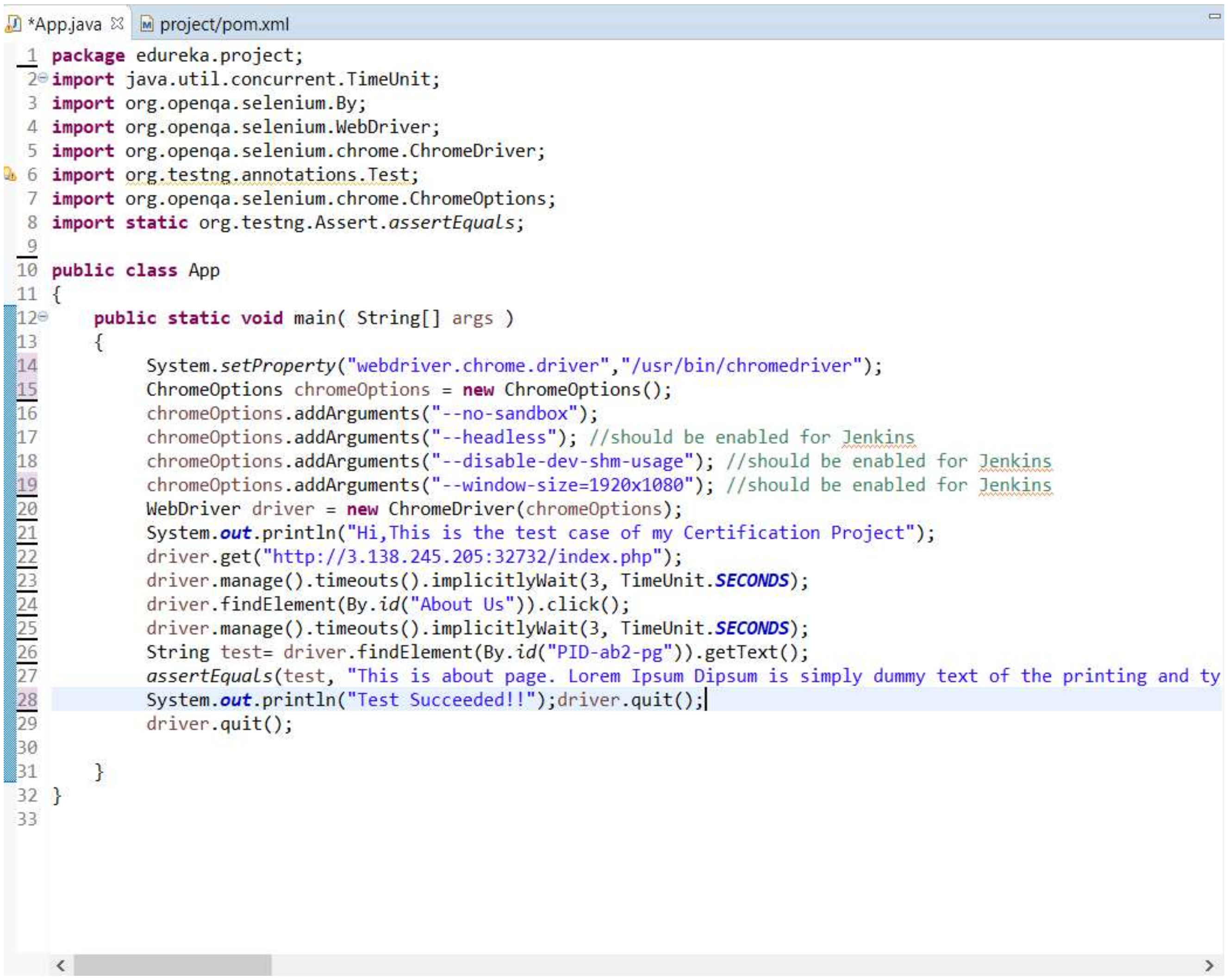
Pipeline was set up:

With each previous job acting as a trigger and all ahead jobs will be put in the post build triggers. Upstream and Downstream Projects (Jobs) were created in each job and the pipeline view is as shown. Last trigger was parametrized trigger from the job4 to job5 when build of job4 is failed. And first one was triggered with GitHub push or pull as described above.

Pipeline Created is as shown :



Selenium Code file contents are as below:



```
package edureka.project;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.openqa.selenium.chrome.ChromeOptions;
import static org.testng.Assert.assertEquals;

public class App
{
    public static void main( String[] args )
    {
        System.setProperty("webdriver.chrome.driver","/usr/bin/chromedriver");
        ChromeOptions chromeOptions = new ChromeOptions();
        chromeOptions.addArguments("--no-sandbox");
        chromeOptions.addArguments("--headless"); //should be enabled for Jenkins
        chromeOptions.addArguments("--disable-dev-shm-usage"); //should be enabled for Jenkins
        chromeOptions.addArguments("--window-size=1920x1080"); //should be enabled for Jenkins
        WebDriver driver = new ChromeDriver(chromeOptions);
        System.out.println("Hi,This is the test case of my Certification Project");
        driver.get("http://3.138.245.205:32732/index.php");
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        driver.findElement(By.id("About Us")).click();
        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        String test= driver.findElement(By.id("PID-ab2-pg")).getText();
        assertEquals(test, "This is about page. Lorem Ipsum Dipsum is simply dummy text of the printing and ty
        System.out.println("Test Succeeded!!");driver.quit();
        driver.quit();

    }
}
```

Comments :

1. About us tab was clicked
2. The contents of the <p> field was matched. If assertion is right then test pass otherwise fail. Hence the whole **Job4 build fails**.
3. Note --- many arguments were added to **chromeOptions** in order to make chrome work on Jenkins like headless start , disable sand box , describing the resolution of the display etc.. Otherwise the selenium test was not running as chrome was not getting started.

Result:

All desired tasks in the given Project were tested successfully.

Attachments:

1. Site.pp File for application of configuration through puppet
2. App.java – Selenium Test File
3. pb.yml and pb2.yml – ansible playbooks
4. Note all other configurations/scripts are clearly mentioned in this document.
5. DockerFile for building image
6. Selenium Jar File

Note 5 and 6 with Project website are there on my Github:

https://github.com/vikhyat-dhamija/devops