A Project Demonstration on Continuous Deployment/Delivery (C.D) using Jenkins and Ansible

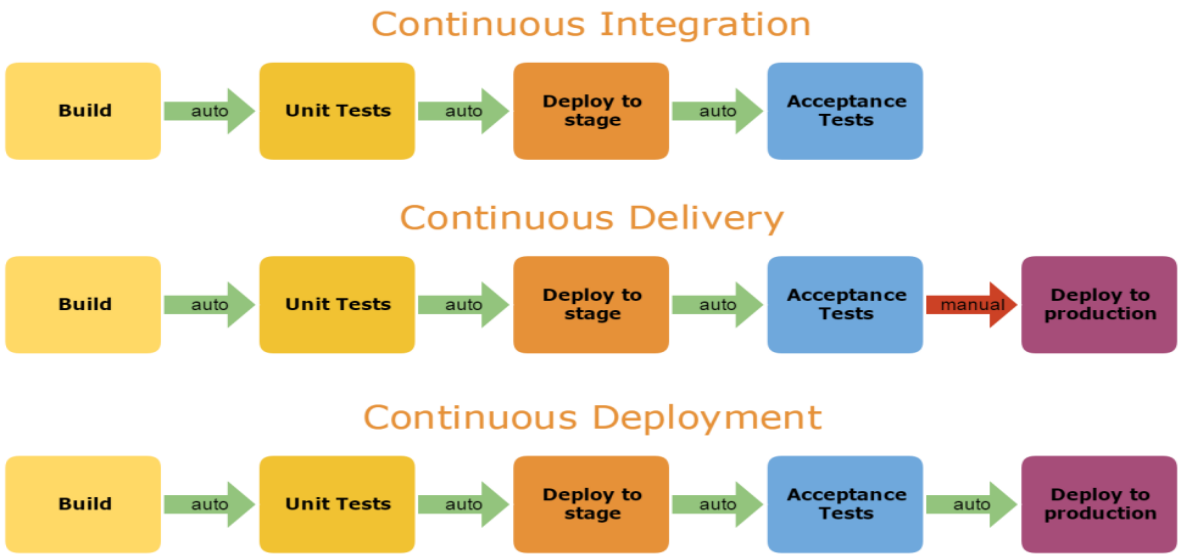
What is Continuous Deployment (C.D)?

Continuous Deployment (CD) is a software release process, using automated testing to validate that all changes to a codebase are accurate and ready to be deployed autonomously to a production environment. This software release cycle has progressed and advanced over recent years.

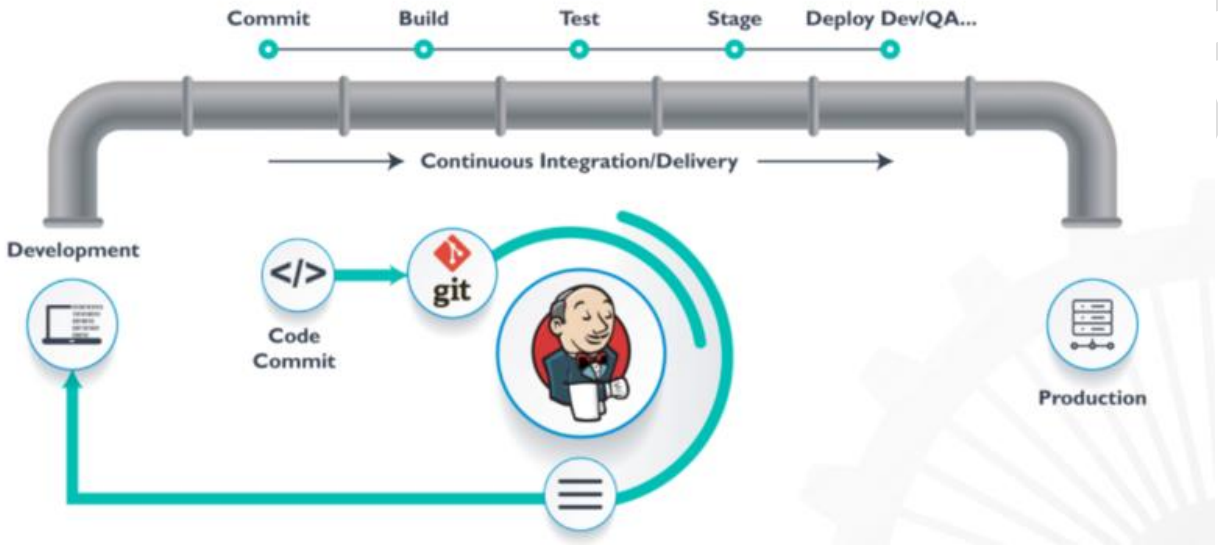
A software engineering approach, CD delivers software functionalities frequently through automated deployments.

As a software release process, CD automatically tests, verifies, and pushes new pieces of code into a production environment if they pass the automated tests.

Process of C.D:-



Continuous Deployment WorkFlow:-



Pre-requisites:

1. Basic understanding of Jenkins
2. Basic knowledge of EC2 instances and linux administration
3. Awareness of Tools such as Nexus, Git,SonarQube and Ansible

Tools used:

Jenkins	Git	Maven	SonarQube	Selenium
Checkstyle	AWS EC2	Nexus3	Ansible	Apache Tomcat

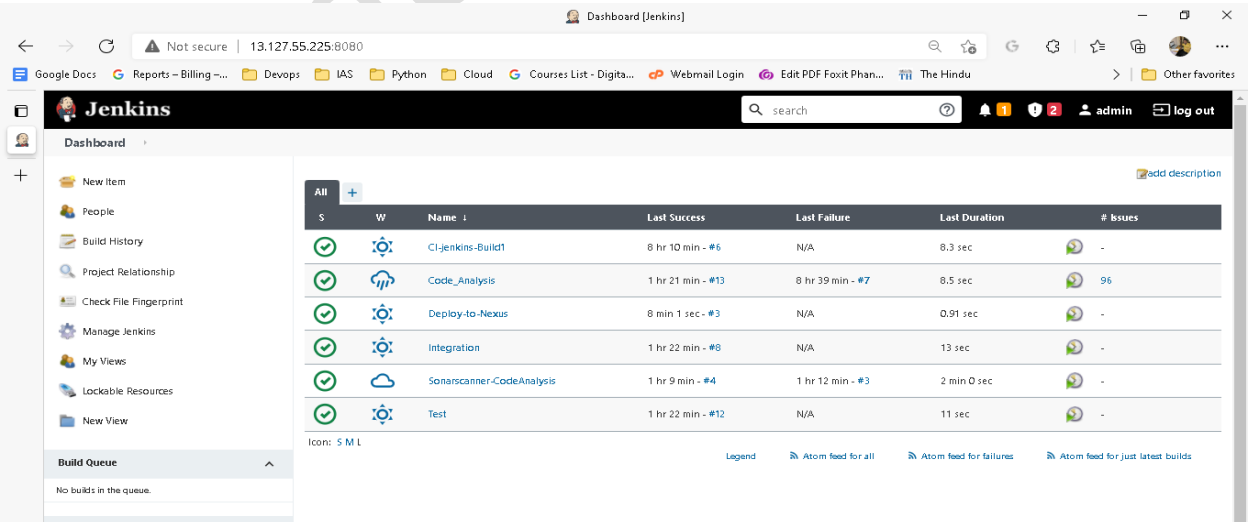
Note: Inorder to Setup the Continuous Delivery pipeline, we need to have the Continuous Integration setup ready with us. If not, It can be set up by referring to the below link:
https://bit.ly/CI-Jenkins-Final-Document

Continuous Integration pipeline Setup

(Ref: https://bit.ly/CI-Jenkins-Final-Document)

Firstly, Implement the CI pipeline setup using the document reference, once completed, you should be ready with the following setup:

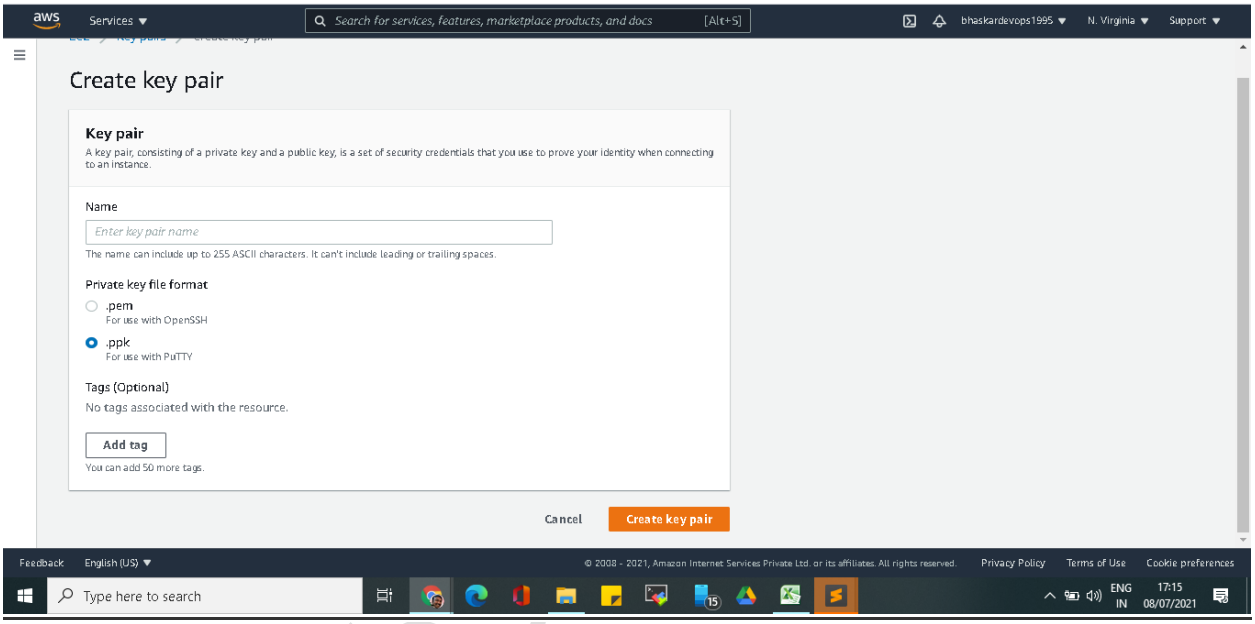
1. Valid AWS account to create infrastructure for CD pipeline
2. Necessary Security Groups. If you are fresher/ new to security Groups, for simplicity create a single security group and include two inbound rules : 1. Allow all traffic within the security group 2. Allow all traffic from your ip.(your ip changes occasionally so make sure to update it periodically)
3. Servers: Jenkins, SonarQube, Nexus
4. Jenkins configured and integrated with remaining services
5. Jenkins jobs for Build, Test, Integration Test, Code Analysis, SonarQube code Analysis(optional), Deploy to Nexus.. (names might vary, focus is on the functionality)



Stage -1

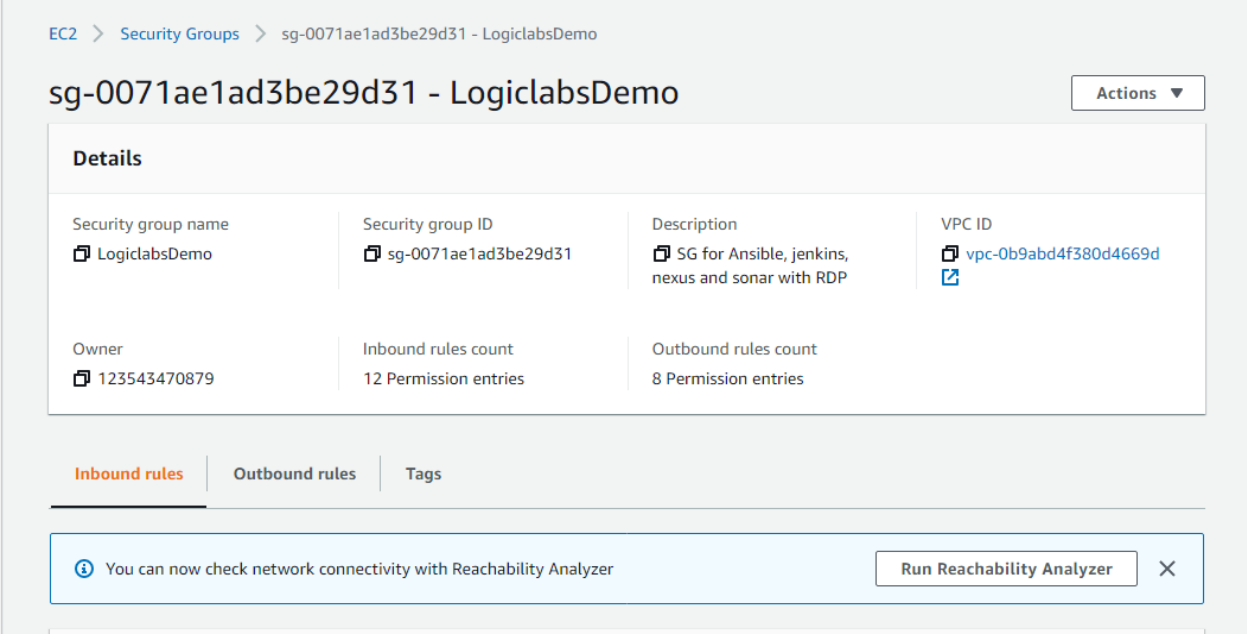
Key pair setup (if not already created)

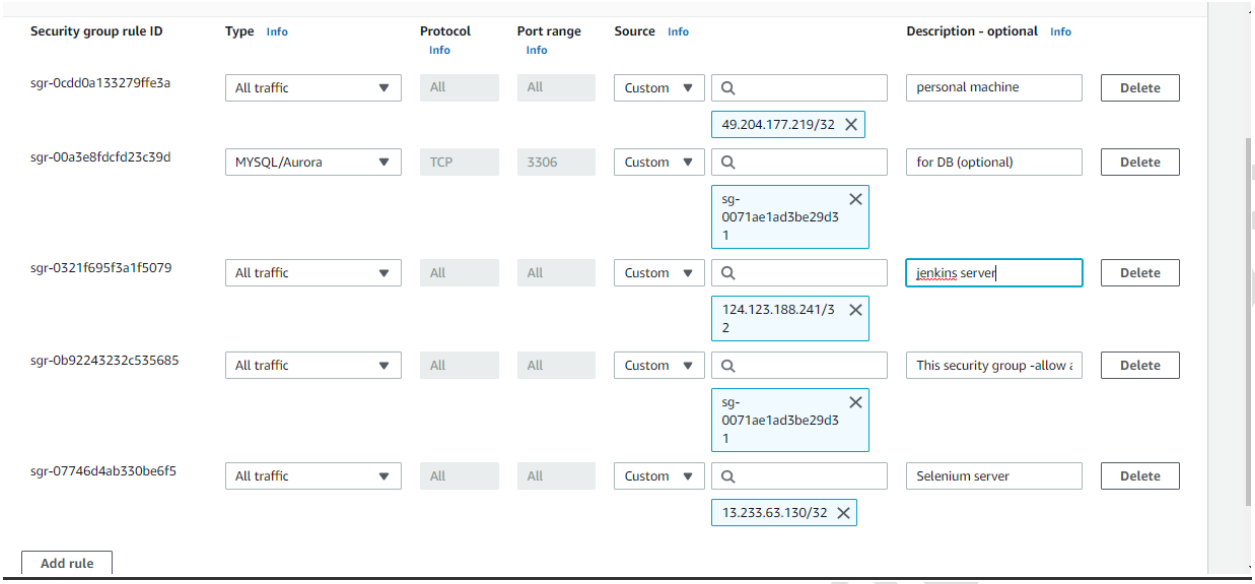
Login to the AWS management console → EC2 service→ Keypair and Create a new keypair for the purpose of creating EC2 instances



Security Groups setup

1. Login to AWS console-> Ec2-> Security Groups
2. Create three Security Groups (S.Gs) one for each- Jenkins, Sonar and Nexus
3. Edit inbound rules for each security group as per the screenshot below and leave outbound as it is.
4. If unclear about inbound rules, for the practice purpose, create a single security group and include two inbound rules : 1. Allow all traffic within the security group 2. Allow all traffic from your ip.(your ip changes occasionally so make sure to update it periodically)

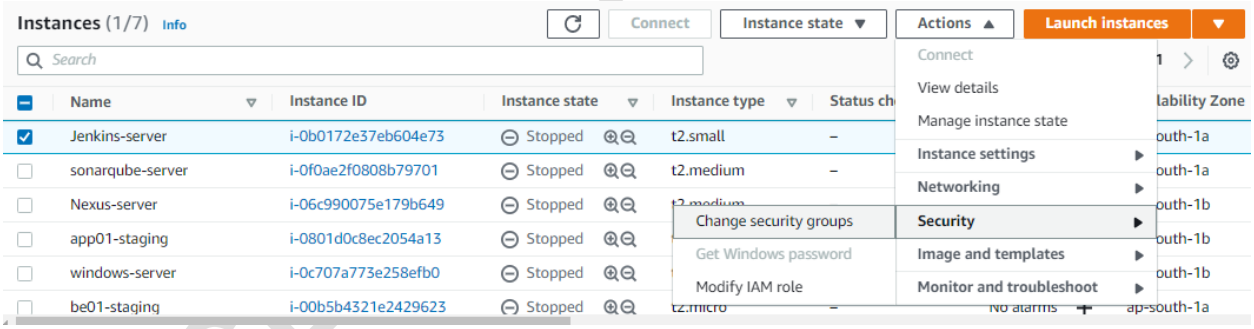




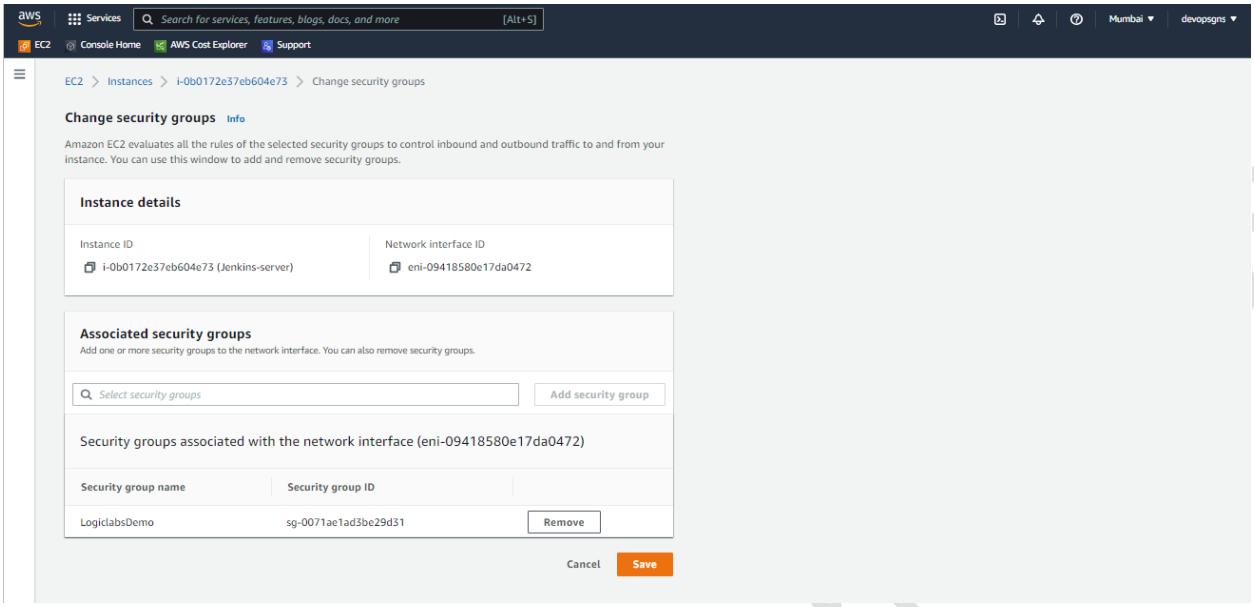
Configuring Security Group for earlier setup:

Configure other servers earlier created for C.I demo, to link to our common security group, I.e., Jenkins server, Nexus server and sonar server.
For linking go to the servers follow the below steps:

1. Go to Instance→ Actions→security → Change security Group



2. Select the common security group and add the security group and save.



Server setup for C.D

We need to create 3 more servers for Continuous Delivery setup:

1. Web server - linux to host our web-application
2. Windows server - for Selenium Tests
3. Backend server - linux for Db, Cache and MQ

(4. Web-server for Production - Optional)

Note: Ansible will be configured within the Jenkins server

Step1: clone source code and user data from repository

https://github.com/gnsharma530/Logiclabs-cidemo.git

Step2: checkout to cd-ansible-jenkins branch and open userdata folder:

Commands to checkout branch:

git checkout cd-ansible-jenkins

cd userdata

Step3 : observe the files in it

```
home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ ls
Jenkinsfile  README.md  ansible/  ible-jenkins  pom.xml  settings.xml  src/  target/  userdata/

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ ls
Jenkinsfile  README.md  ansible/  ible-jenkins  pom.xml  settings.xml  src/  target/  userdata/

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ cd userdata/

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata (cd-ansible-jenkins)
$ ls
 backend-stack.sh  "backend-stack.sh'"*   tomcat-provision.sh*   windows-node

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata (cd-ansible-jenkins)
$
```

User data folder contains the script required for bringing up the server with necessary configurations without manually configuring everything.

step4 : Server Creation:

open the AWS console to create the ec2 instances for Servers

AWS console→ ec2→ create instance

Choice of Instances:

Windows :t2.medium

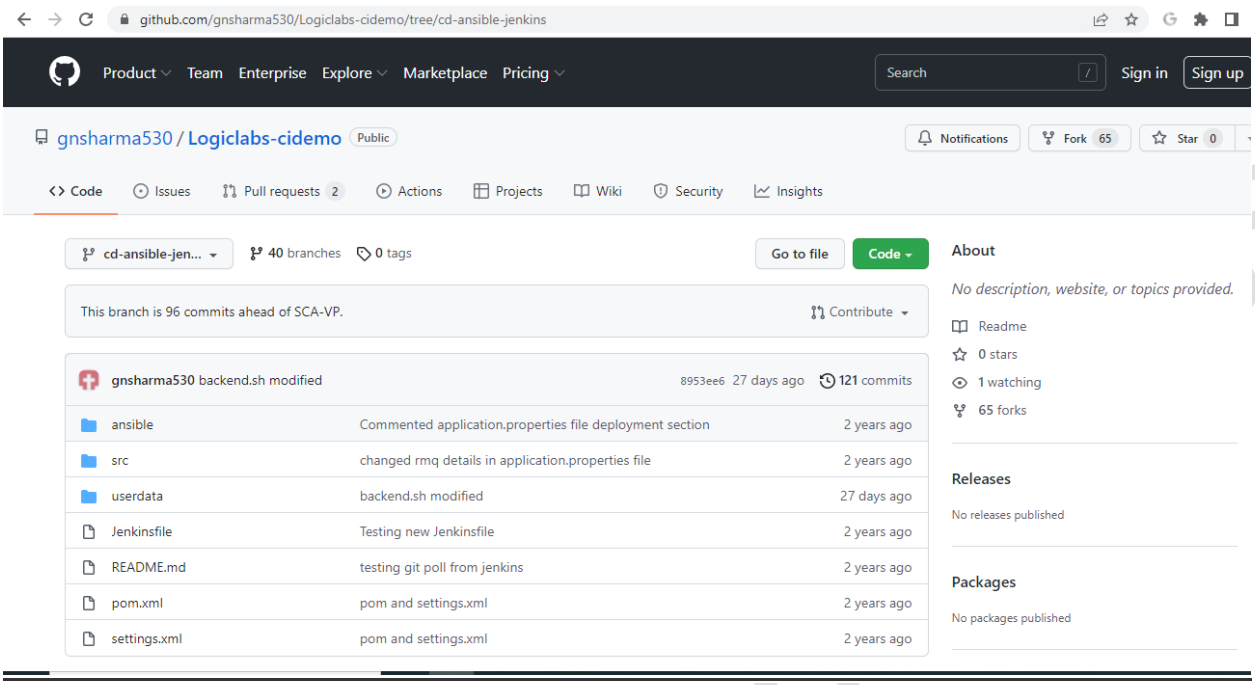
Web-server :t2.micro

Backend server: t2.micro

If you want to replicate prod web server also, you need another ec2 instance.

-Select the relevant security groups

-paste the proper user data from the user data folder of the repository.



Step 4.1:

Web server creation : OS→ UBUNTU

Instance type → t2 micro

use below settings while creating web-server:

Name: app01-staging
Os: ubuntu 18 LTS
Instance type: t2.micro

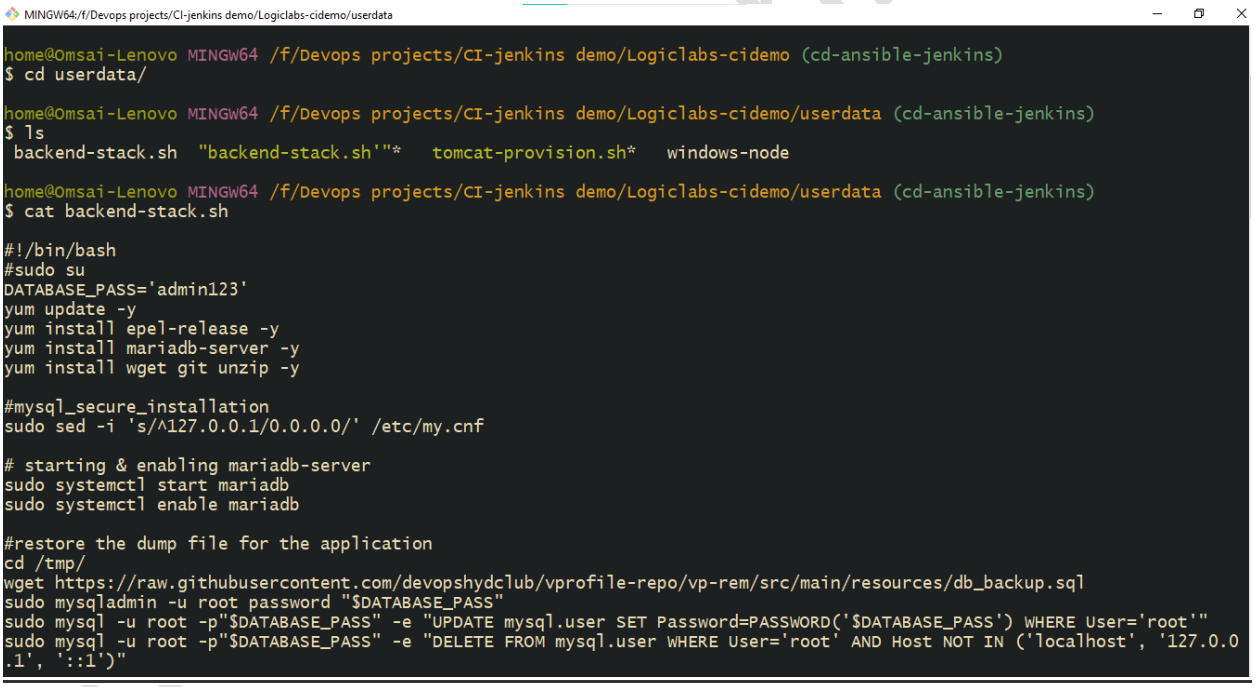
- Leave the user data blank for web-server, as Ansible will be installing and configuring Tomcat.
- Link the created security group which was created before.
- Apply the keypair used for ssh

After creation of vm, check whether you are able to ssh into the machine.

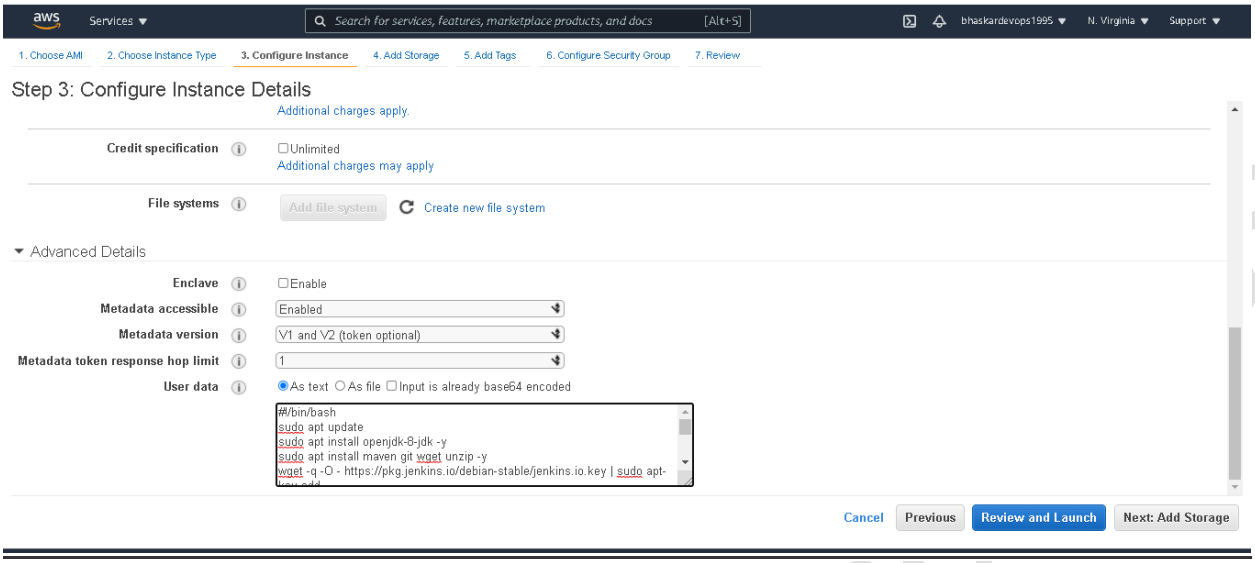
Step 4.2 :

Backed server creation:

Instance Type: t2.micro
OS: CentOs
Name: be01-staging
Userdata:
In the folder where you have cloned the repository, go to the user data folder and open backed-stack.sh there.



Copy the contents and paste in the user-data space in the instance creation page.



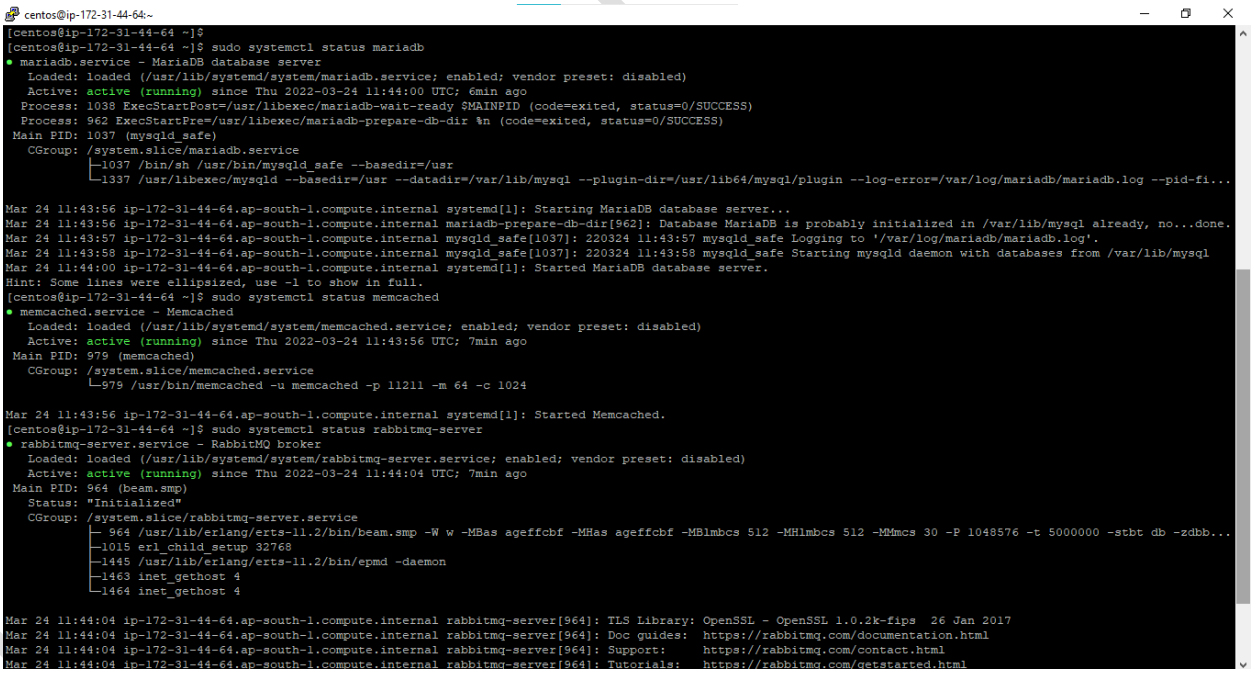
- Configure Security group and keypair and create vm.

Log into the backend vm using ssh and run the following commands to verify the installation of 3 services - Maraidb, Rabbitmq and Memcached.

$ sudo systemctl status mariadb

$ sudo systemctl status rabbitmq-server

$ sudo systemctl status memcached



<u>Note: If you cannot see one or more services not active(running) please perform the below steps.</u>**

<u>Step4.2.1: (</u>use this when none of them are configured properly)

1. Ssh in to the be01-staging machine and create a file with the name backend-stack.sh using command below
2. $ vi backend-stack.sh
3. Copy paste the content of user data into the file and save it.
4. Assign execution permissions
5. $ sudo chmod +x backend-stack.sh
6. Run the shell script
7. $ ./backend-stack.sh
8. Verify the installation using the systemctl commands.

This will install the services in the backend vm.
*Note: if you face any issues with installation, troubleshoot individual service.

Step4.2.2: (use this If one or more services are running but remaining are not running)

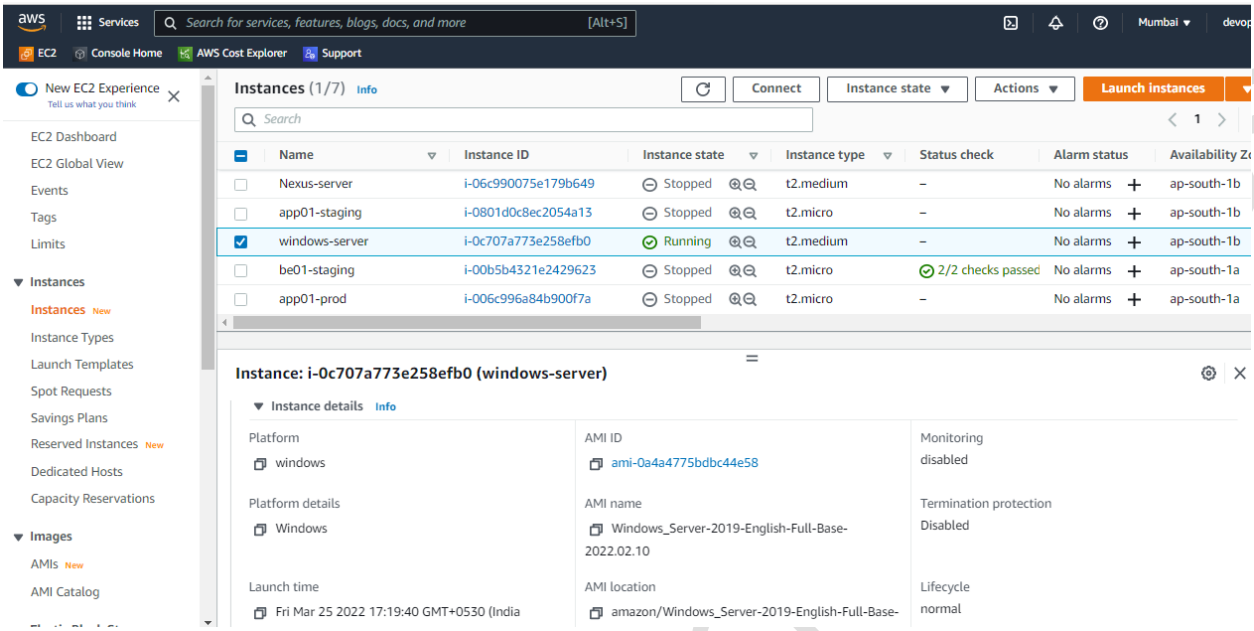
1. Execute the commands from the backend-stack.sh manually to install the 3 services (mariadb-server, Rabbitmq, memcached)
2. Verify installation using systemctl commands

Step 4.3 :
<u>Windows server creation:</u>

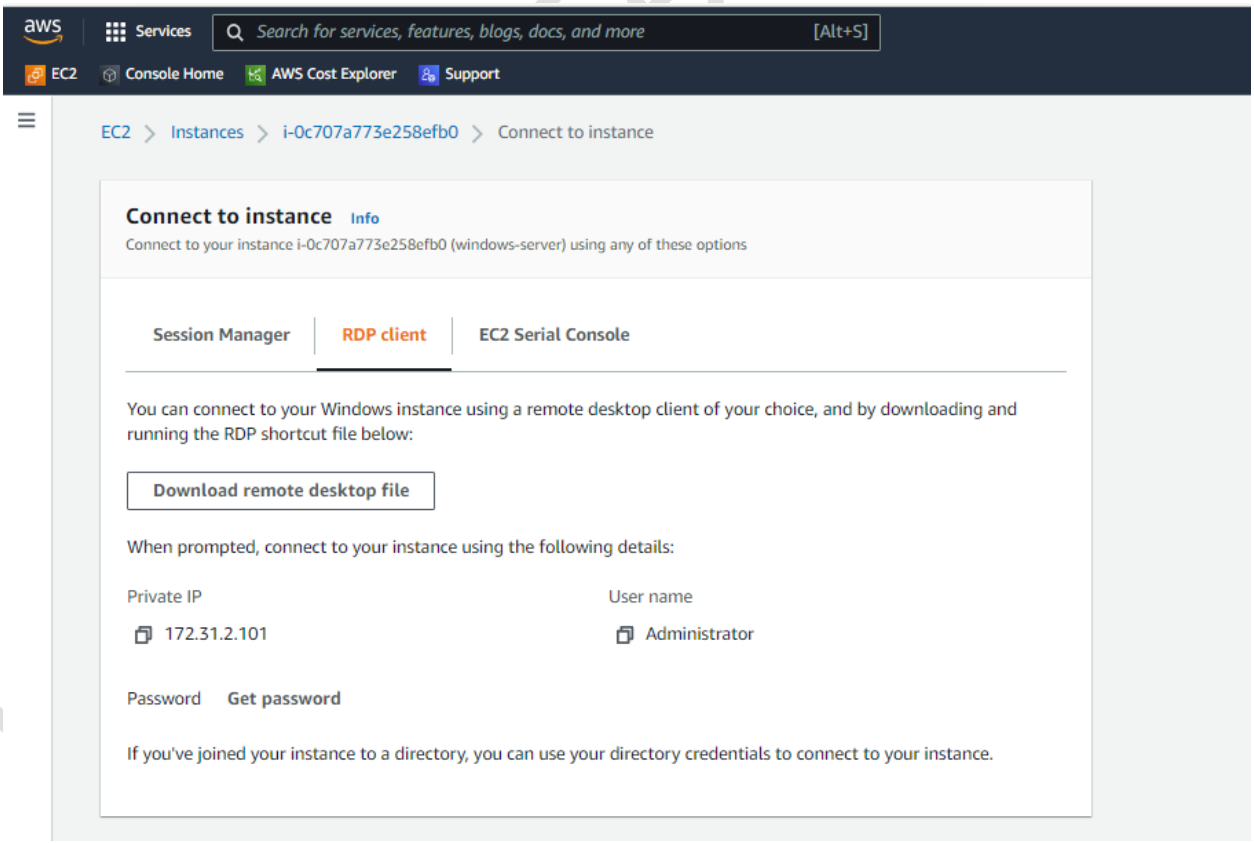
Instance Type: t2.medium
OS: Windows server 2019
Name: windows-server
Userdata:
In the folder where you have cloned the repository, go to the user data folder and open windows-node there. Copy the contents and paste in the user-data space in the instance creation page.
Select the security group and keypair.

Give the username and complete the setup of vm.

How to connect to Windows-server:

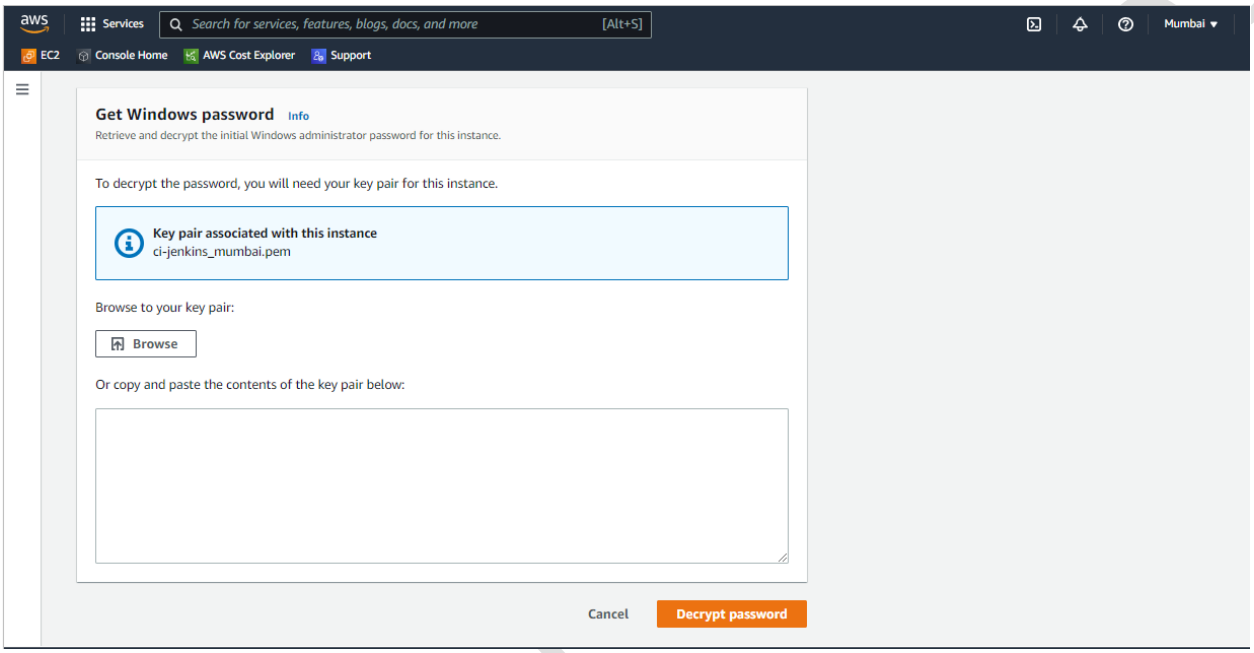


After the server is on, press on the connect option on the top menu bar. Select the RDP client as the choice of connection.



Select the Download remote desktop file. A rdp file will be downloaded. Before you open the file, first get the password to login to windows server. For that follow the below steps:

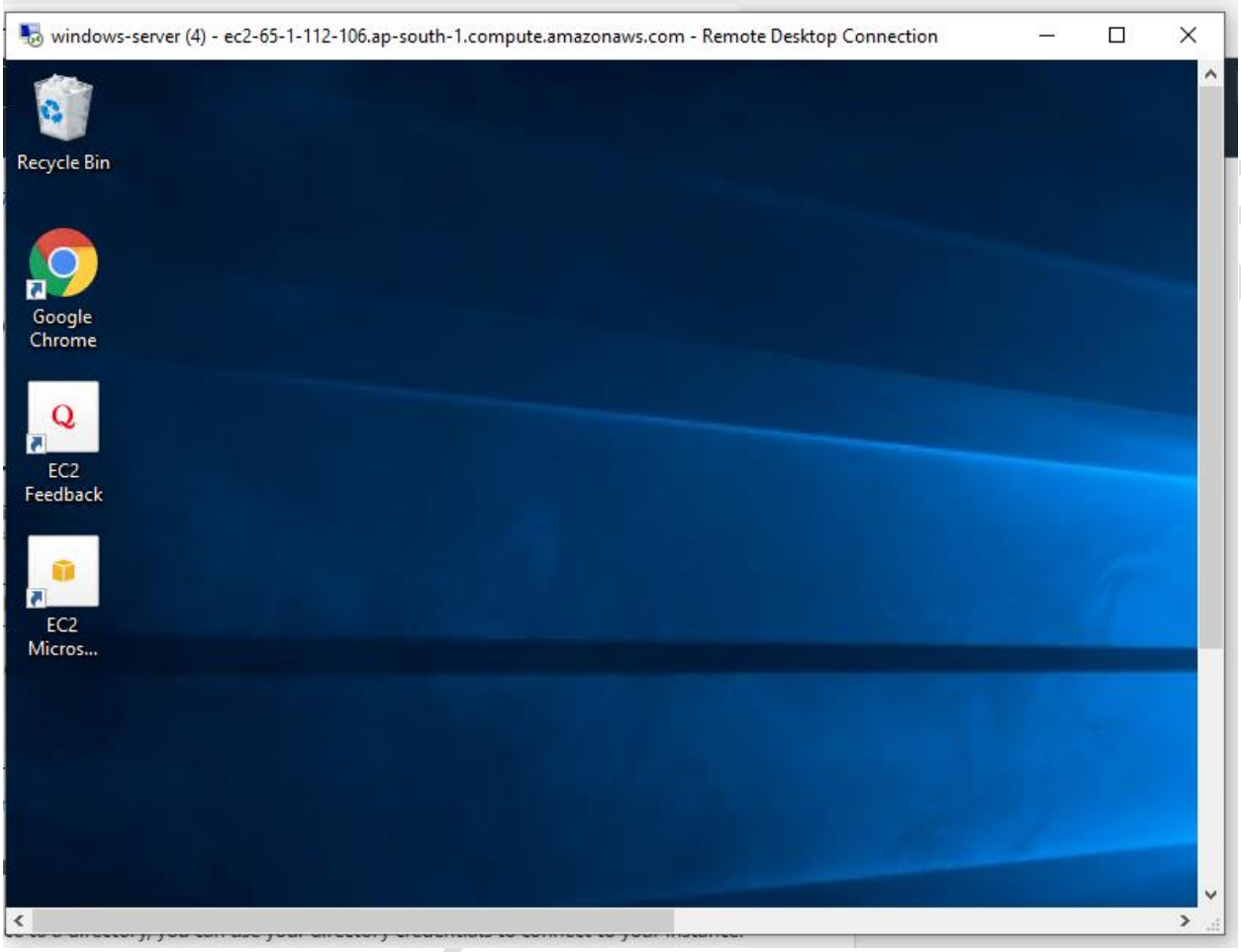
1. Keep the private key of the keypair handy. (it is the file with .pem extension)



2. Select the pem file to upload or paste the contents manually and password will be shown. Please copy and keep it safe.

Now login to the windows server using the earlier downloaded RDP file and enter the saved password.

You should be able to see the desktop page with windows chrome installed.



Step 5 Configuring the Ansible on Jenkins

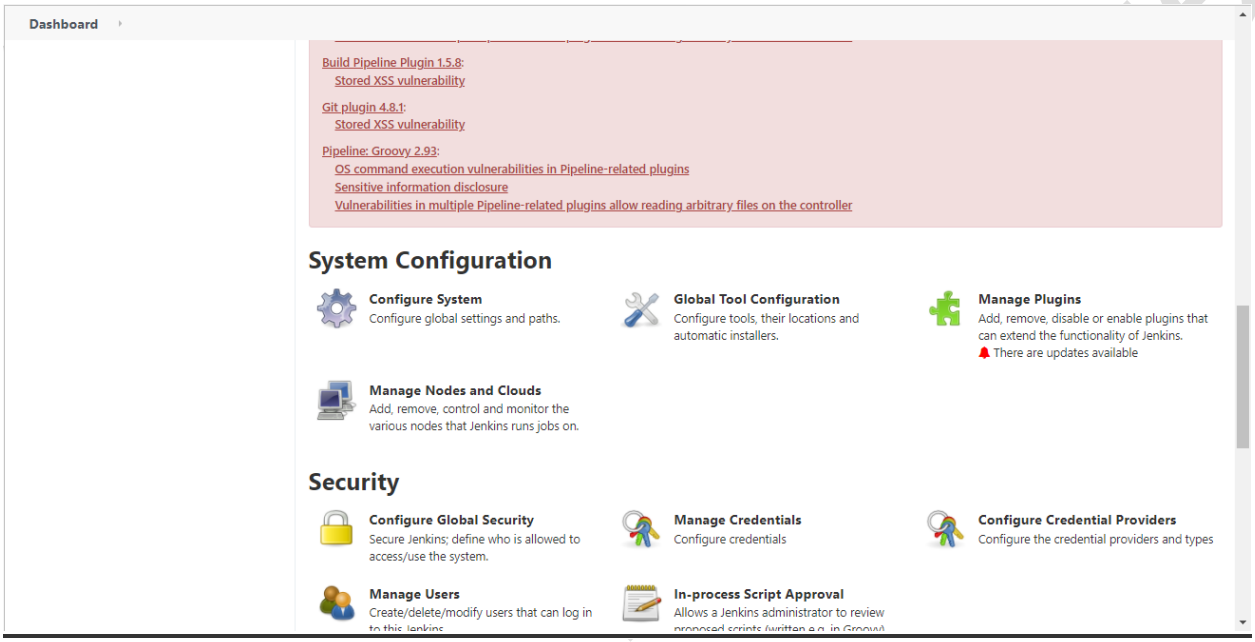
Step 5.1 Ansible installation on jenkins server

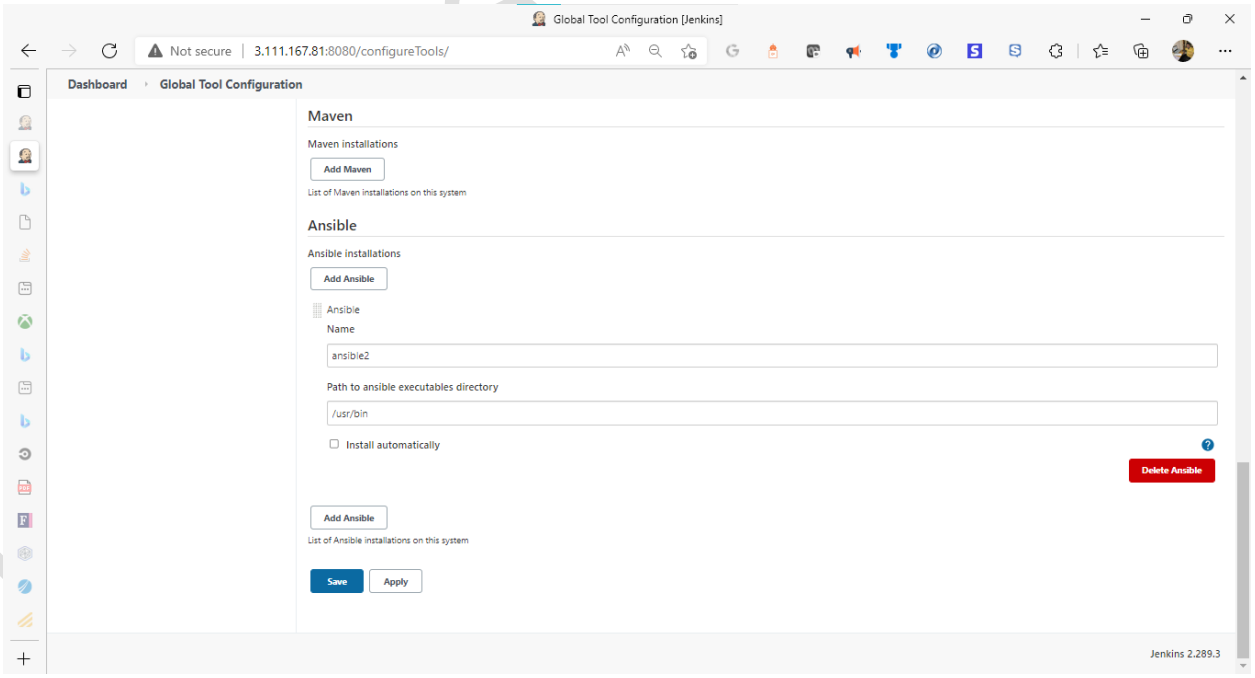
- We will be installing Ansible on jenkins as creating a new ansible server is not necessary for demonstration purposes.
- Login to jenkins server using ssh
- Use the below commands to install ansible on Jenkins server:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

- Check the ansible installation using $ ansible --version

- Open browser and Login to the jenkins and install ansible plugin
- Go to jenkins→manage → plugins→ search for Ansible and click on install without restart
- Go to Global tool configuration and give path for Ansible
-

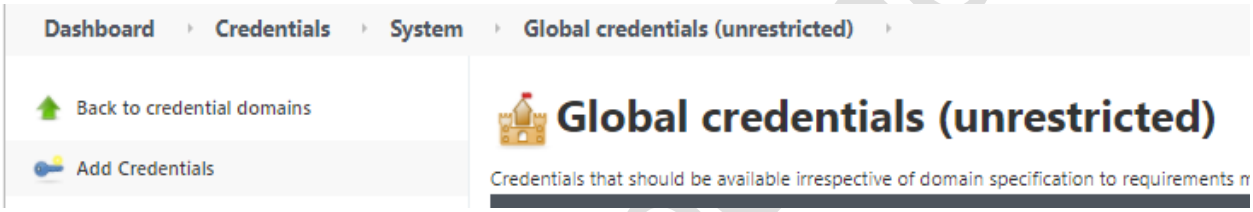




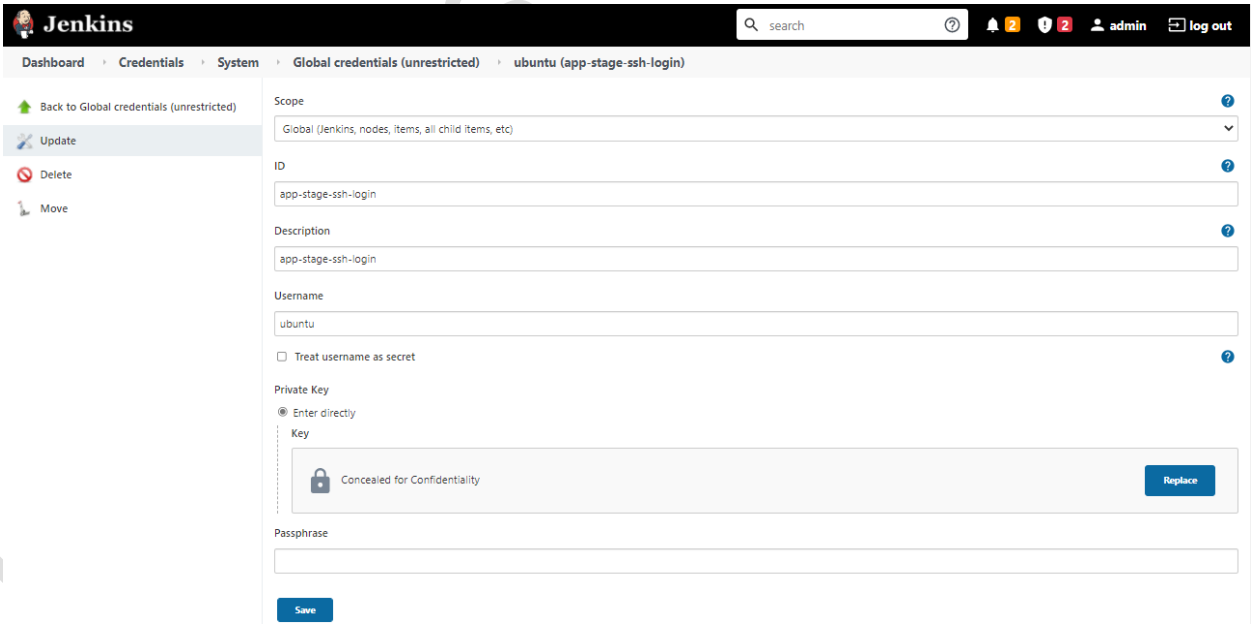
Step 5.2: Configuring Jenkins credentials for Ansible to access web-server

Ansible will connect to hosts using ssh. For this we need to configure the web-server ssh private key in jenkins credentials manager. Later we will create a job where we access these credentials for deployment of artifact into web server.

- Jenkins home page → manage jenkins → Credentials manager → System → Global credentials and select create new credentials



Configure ssh credentials as follows(keep the private key you used to create the web-server handy)



Save the credentials.

Step 5.3: Web-server setup using Ansible:

We have created a web-server named app01-staging. In this step we will configure Tomcat service in the webserver.

Open the source code folder downloaded from the earlier steps and open the folder Ansible.

```
home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ ls
Jenkinsfile  README.md  ansible/  ible-jenkins  pom.xml  settings.xml  src/  target/  userdata/

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ cd ansible

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/ansible (cd-ansible-jenkins)
$ ls
ansible.cfg  site.yml  templates/  tomcat_setup.yml  vpro-app-setup.yml
```

Brief explanation of Ansible playbooks:

Open the file tomcat_setup.yml to view the playbook used for tomcat setup.

```
- name: Install JDK on Ubuntu 14/15/16/18
  apt:
    name: openjdk-8-jdk
    state: present
    update_cache: yes
  when: ansible_distribution == 'Ubuntu'
```

The above block of code from playbook, installs the necessary tools like jdk in the webserver.

```
- name: Setup tomcat SVC file on ubuntu 16 and 18
  template:
     src: templates/ubuntu16-svcfile.j2
     dest: /etc/systemd/system/tomcat.service
     mode: "a+x"
  when: ansible_distribution == 'Ubuntu' and ansible_distribution_major_version >= '16'
```

The above code set up tomcat service file after downloading it as tarball. Finally, the below part of playbook starts the configured tomcat service.

```
- name: Start & Enable TOmcat 8
  service:
    name: tomcat
    state: started
    enabled: yes
```

Similarly, another playbook helps in deploying the artifact into the tomcat directory.

Open the playbook, vpro-app-setup.yml

The below playbook code downloads the latest artifact from nexus repository.

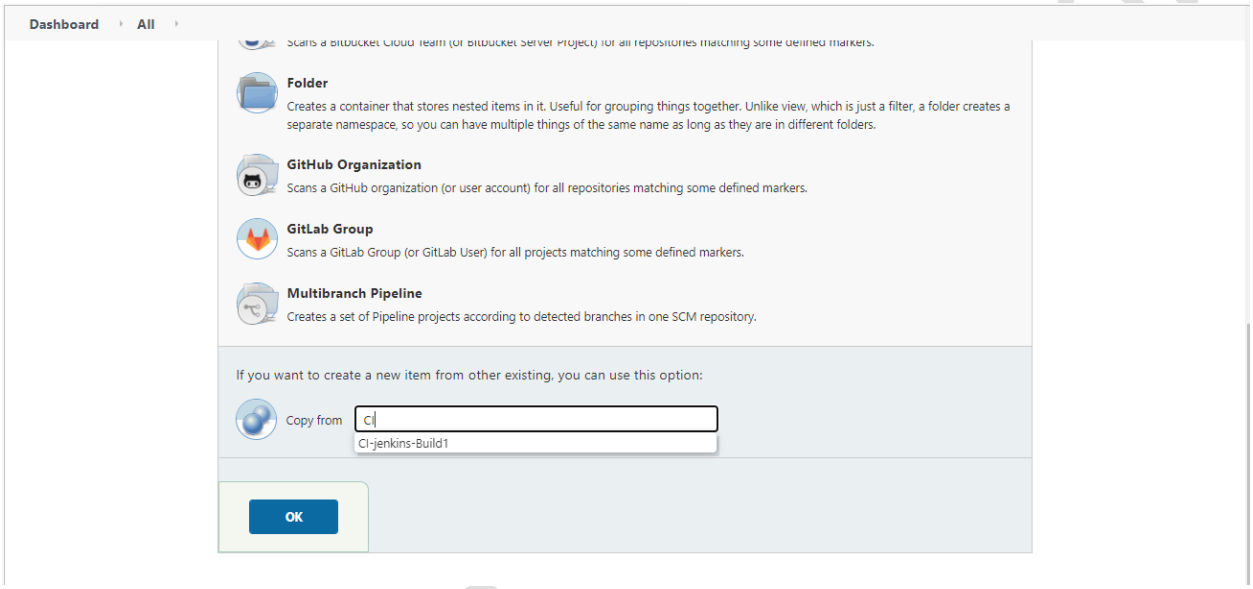
```
    - name: Download latest VProfile.war from nexus
      get_url:
        url: "http://{{USER}}:{{PASS}}@{{nexusip}}:8081/repository/{{reponame}}/{{groupid}}/{{time}}/{{build}}/{{vprofile
_version}}"
        dest: "/tmp/vproapp-{{vprofile_version}}"
      register: wardeploy
      tags:
       - deploy

    - stat:
        path: /usr/local/tomcat8/webapps/ROOT
      register: artifact_stat
      tags:
       - deploy
```

```
      - name: Try deploy artifact else restore from previos old_ROOT
        block:
         - name: Deploy vprofile artifact
           copy:
             src: "/tmp/vproapp-{{vprofile_version}}"
             dest: /usr/local/tomcat8/webapps/ROOT.war
             remote_src: yes
           register: deploy_info
           tags:
            - deploy
        rescue:
          - shell: cp -r old_ROOT ROOT
            args:
              chdir: /usr/local/tomcat8/webapps/

      rescue:
       - name: Start tomcat svc
         service:
          name: tomcat
          state: started

    - name: Start tomcat svc
      service:
        name: tomcat
        state: started
      when: deploy_info.changed
      tags:
       - deploy
```

The above part helps in deploying the downloaded artifact and starts the tomcat service.

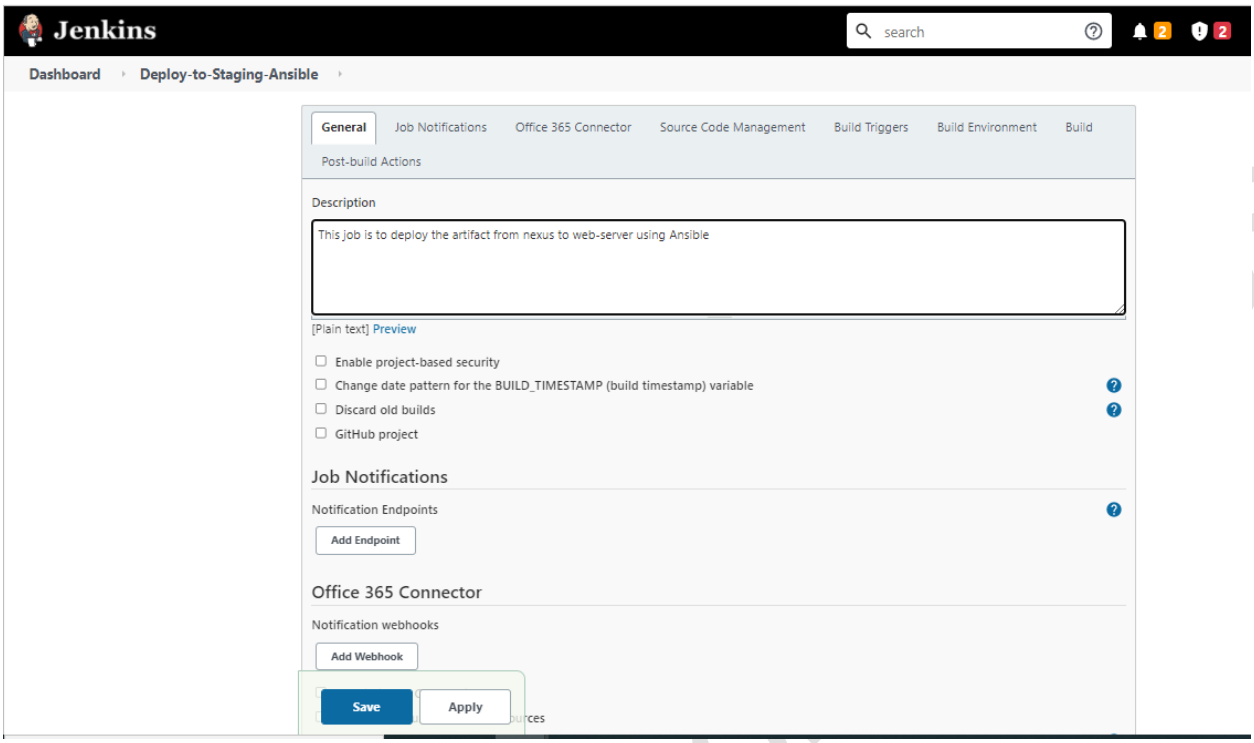
Step 5.4: Configuring Jenkins Job for deployment to staging

From jenkins home page, click on New Item→ Give a name, Deploy-to-Staging-Ansible and select free style project and select a job to copy settings from Build job as shown below.

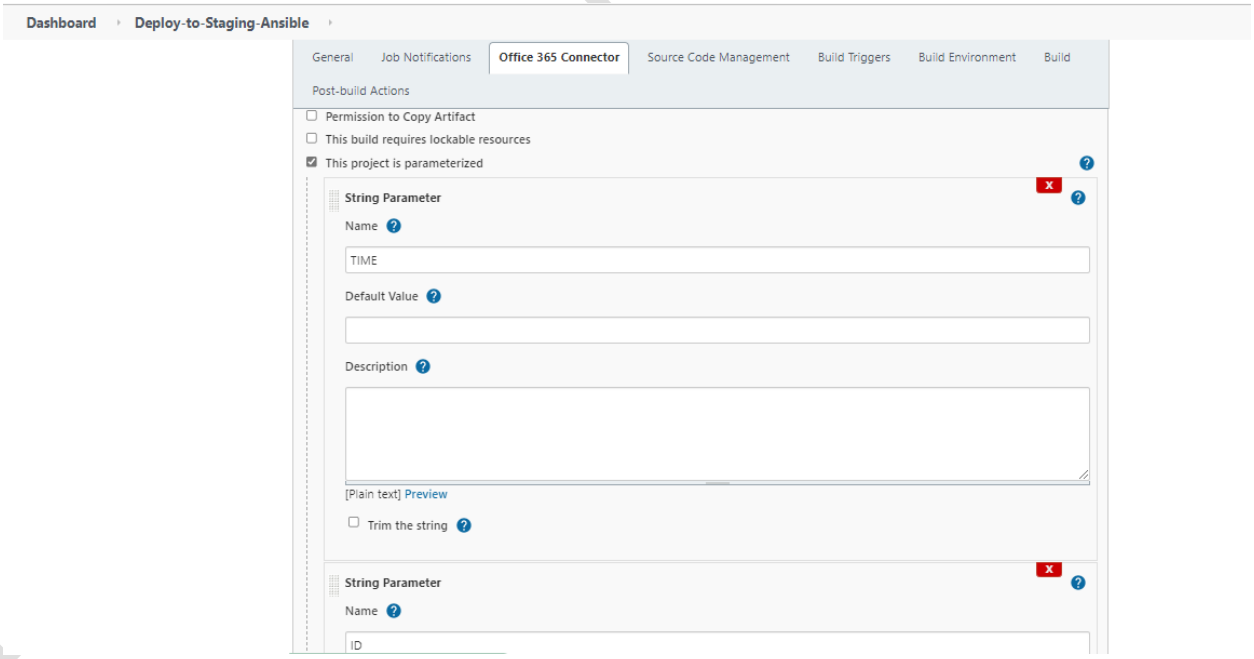


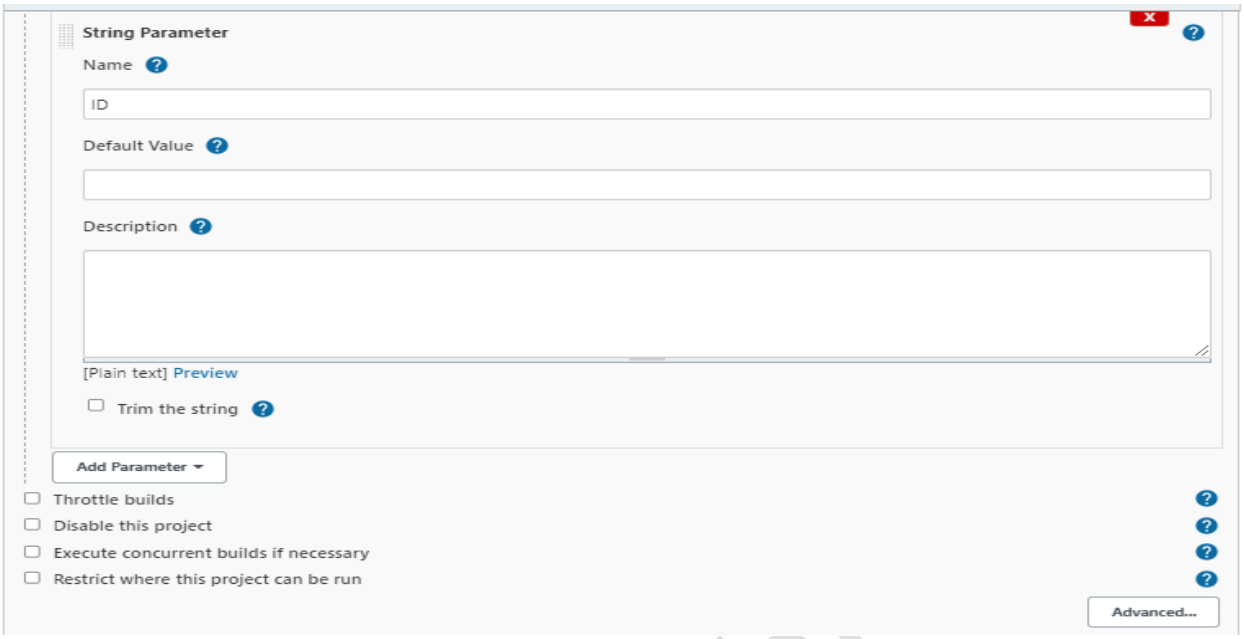
After creating the job configure it as given below:

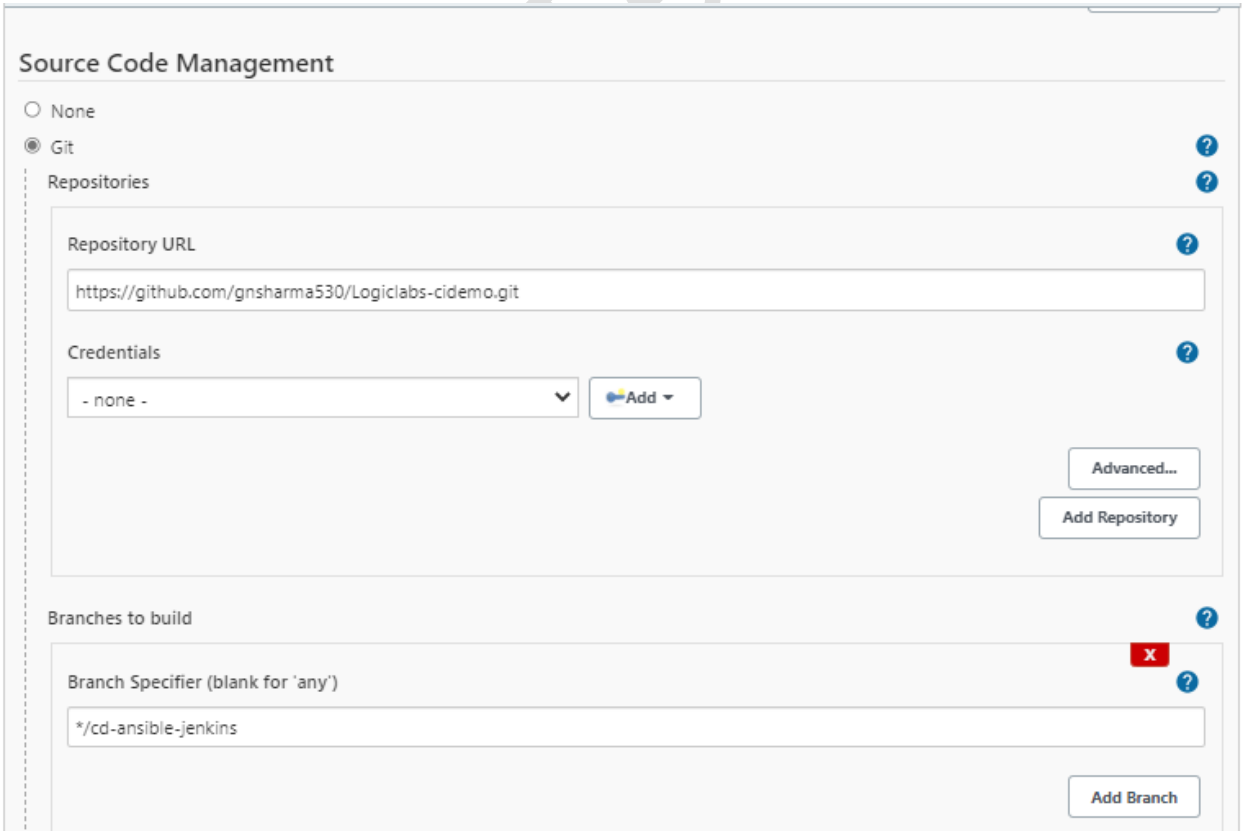
- Remove unnecessary build steps such as execute shell, invoke top level maven targets and we don't want to archive any artifacts in post-build actions hence, delete the unnecessary details and configure the job as exactly given below.



Select *"This project is parameterized"* option in General and select *string parameter* option and give the two parameters called TIME and ID as below.

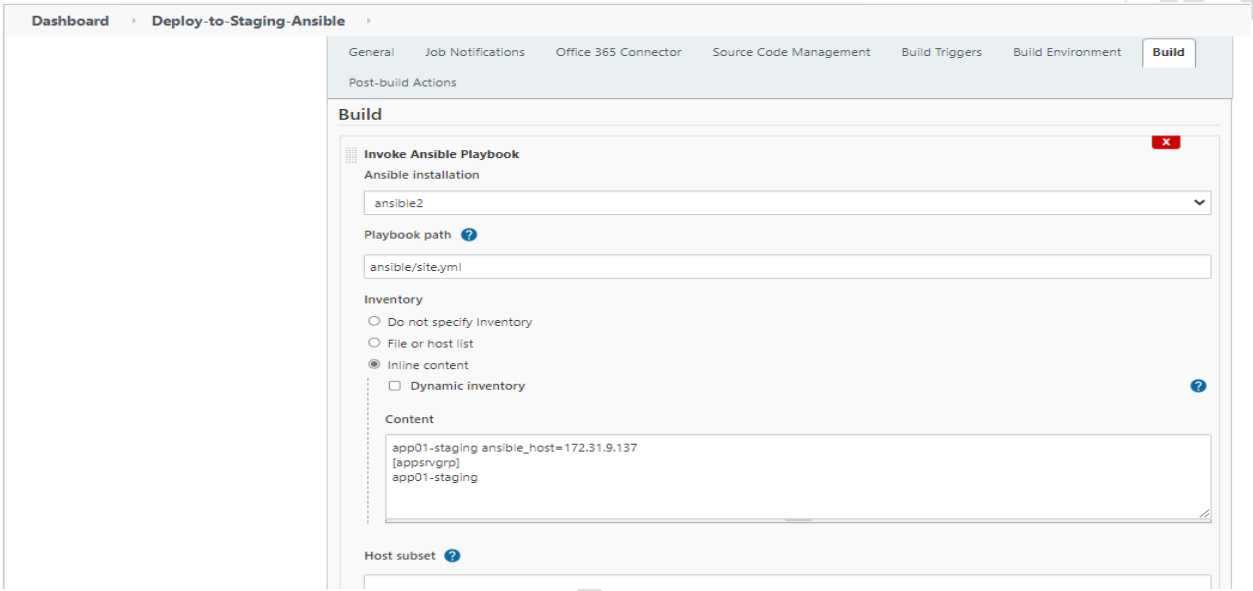






Observe the change in the branch from where Jenkins pulls source code.

Leave Build Triggers and Build Environment as default. Proceed to Build stage and do the configuration as shown below:
In the Build step, select *invoke ansible playbook* option and fill the configuration settings as shown below.

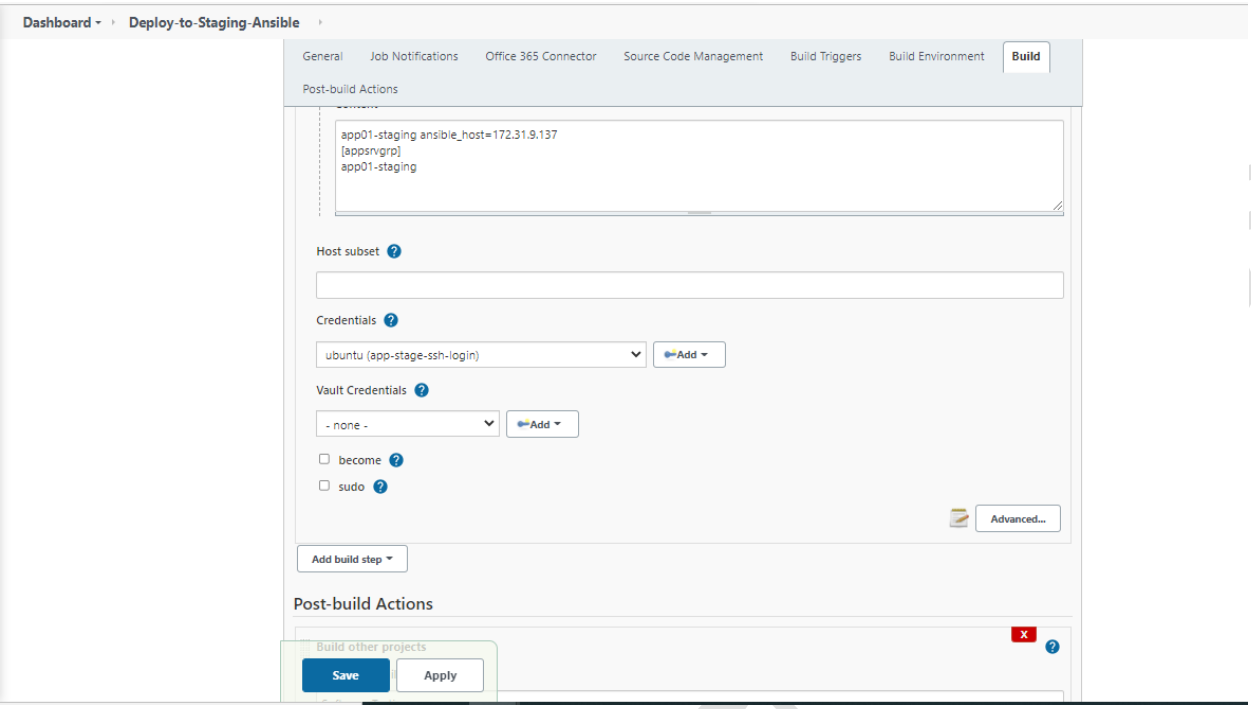


Enter the data in the “content” setting as given below :

app01-staging ansible_host= xx.xx.xx (replace this ip with your web-servers internal ip)

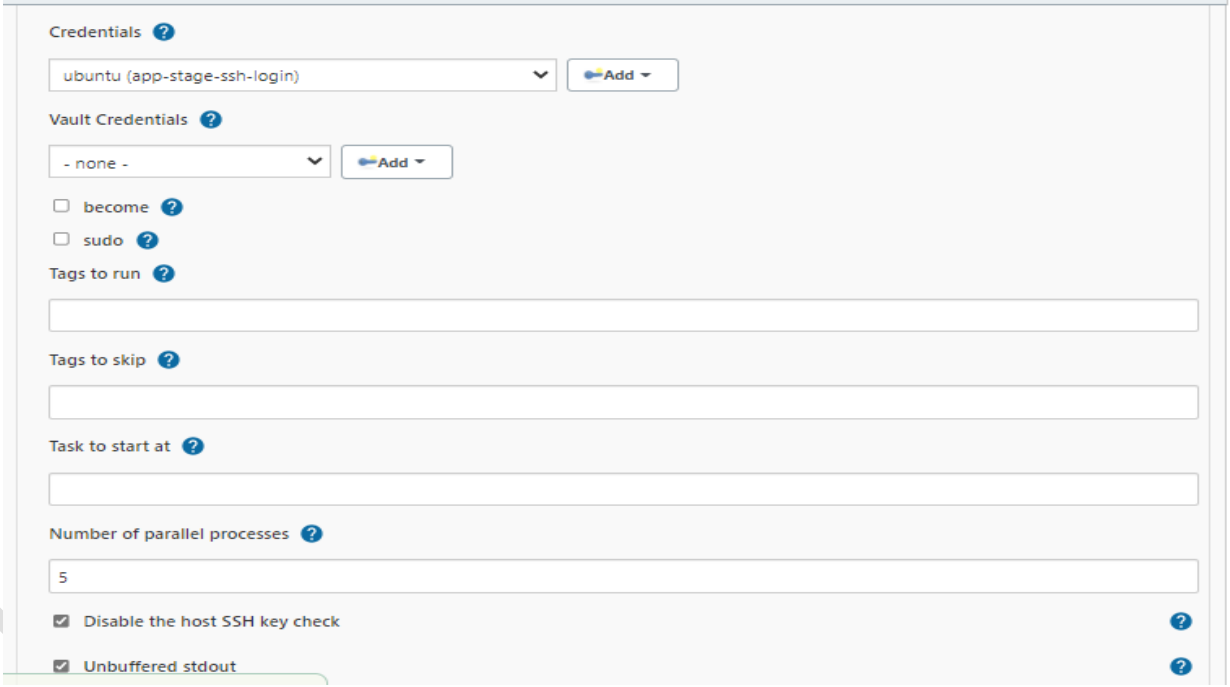
[appsrvgrp]

app01-staging (This is the host name of web-server given while creating)

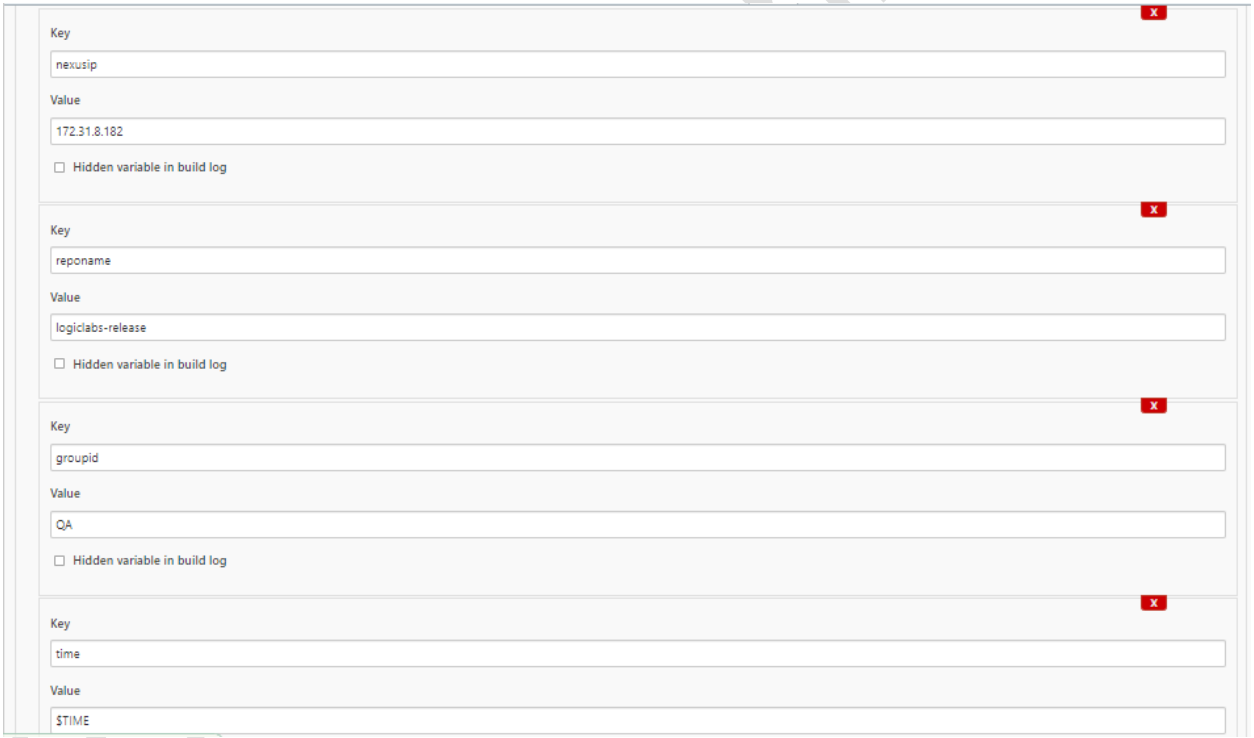


While configuring the credentials for host connection, use the credentials created in the previous step(Step 5.2).

Click on Advanced tab in the build step and fill the settings as given below:



Enter the extra variables as given below:



* change the nexus ip setting and give your nexus server internal ip.

S.No	Key	Value
1	USER	admin
2	PASS	admin
3	nexusip	Xx.xx.xx.xx (your nexus server ip)
4	reponame	Logiclabs-release (your nexus release repo name created during CI demo)
5	groupid	QA
6	time	$TIME
7	build	$ID
8	vprofile_version	$TIME-$ID.war

Remove the post-build Actions setting and save the job.

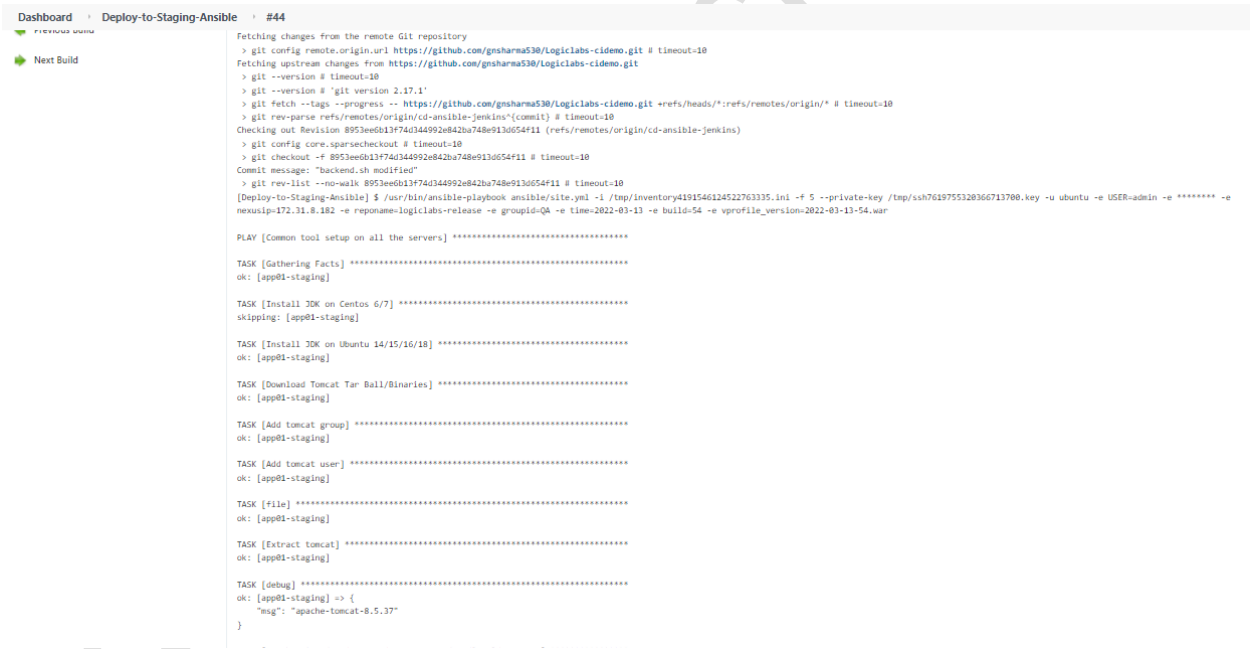
Before you run this job, make sure you login to web-server and check whether python is installed. Use the blow steps:

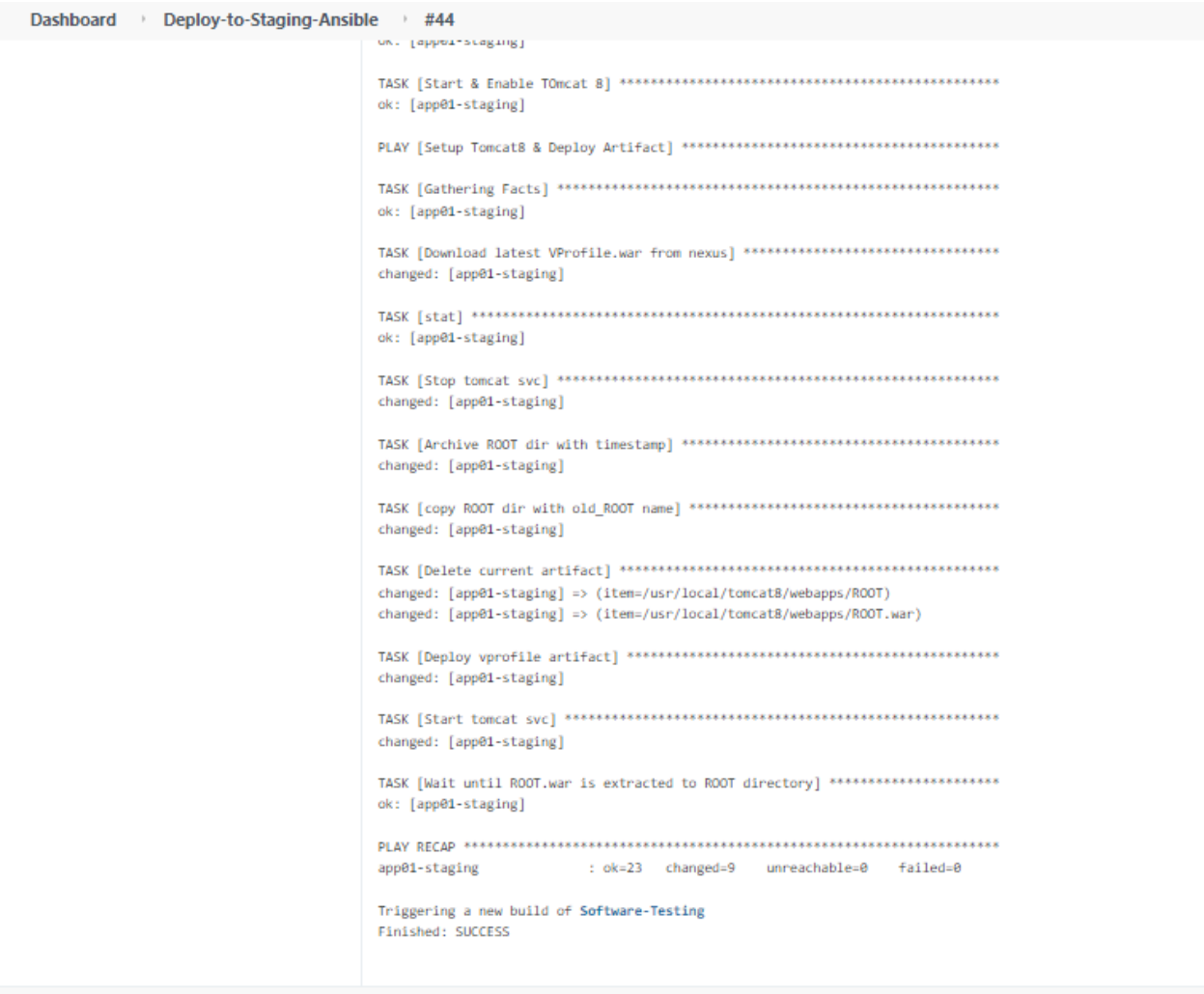
- $ which python #if python is installed, it would return it's version

  If it says python not found, install with the below command,
- $ sudo apt install python

Check the version of python again and if installed correctly, exit out of the vm.

Now run the Deploy-to-staging-Ansible job we have created earlier and check the console output for progress.





If the job is not successful, troubleshoot using the error messages.
* Its practically not possible to anticipate every scenario where the job fails, hence be prepared to google and sort things out.
* Generally a job can fail when python is not installed in web-server or web-server is not reachable. Check security group settings and also all the configurations given in jenkins job carefully.

* As this job depends on Deploy-to-Nexus Job, ensure that nexus server is up and artifacts are getting uploaded to repository properly.

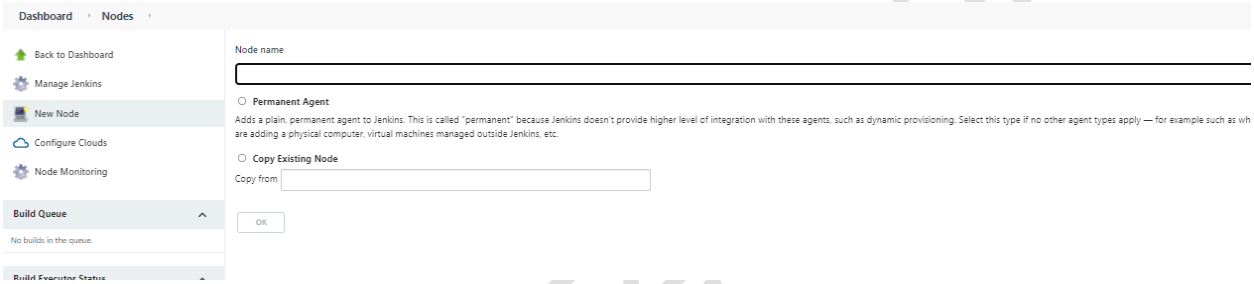
Step 6: Configuring and Running Selenium Automated Test cases

Selenium Test cases require windows server to be created.

- You should have created a windows server in the earlier steps.
- We need to add this windows server as a node slave to jenkins master. This is necessary to make sure that selenium software test cases job only run on windows node.

Step 6.1: Adding windows node to jenkins:

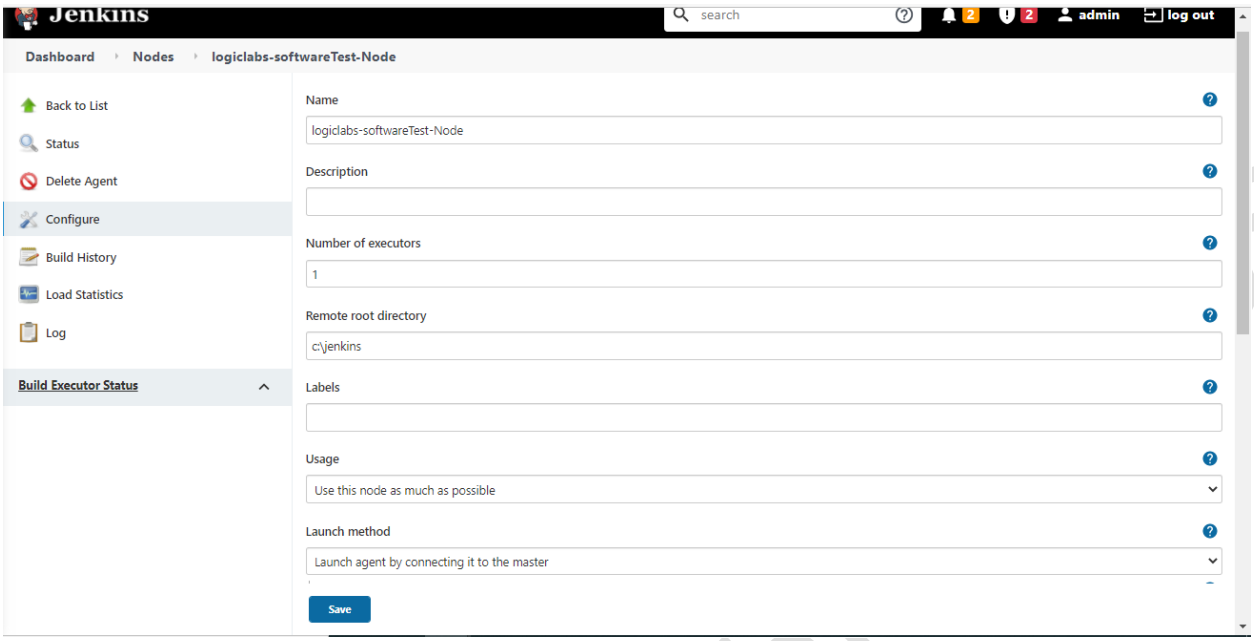
Go to Jenkins Home → manage Jenkins → manage Nodes and cloud → New Node

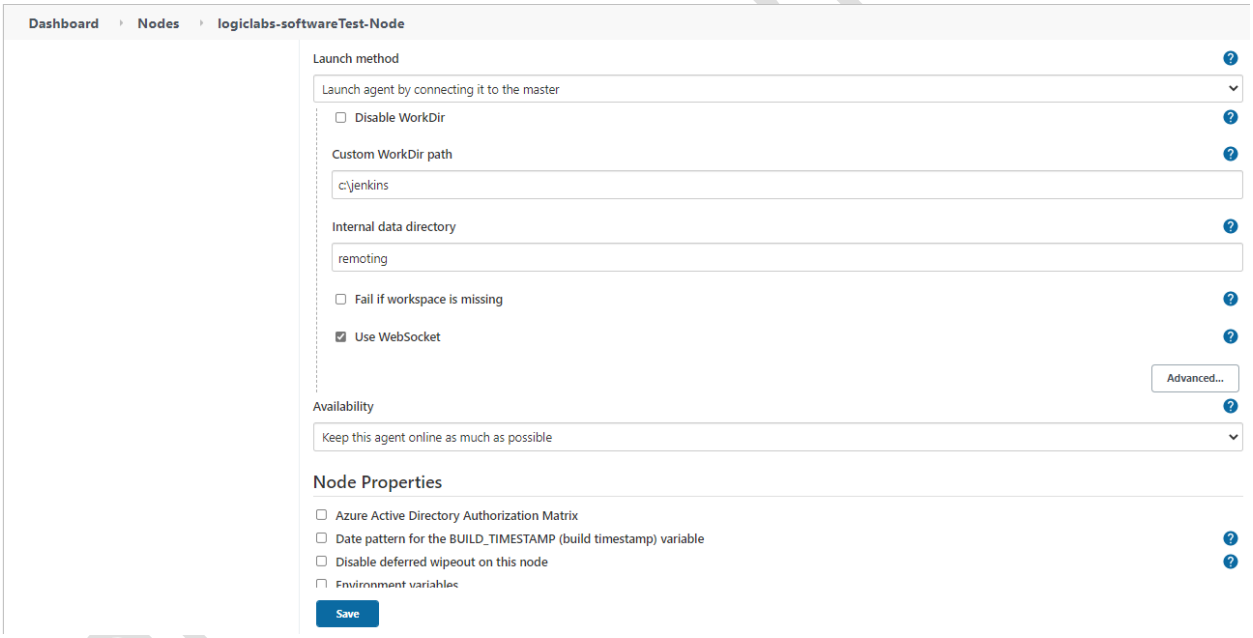


Give the name logiclabs-softwareTest-Node and select permanent agent and select ok.

Step 6.2

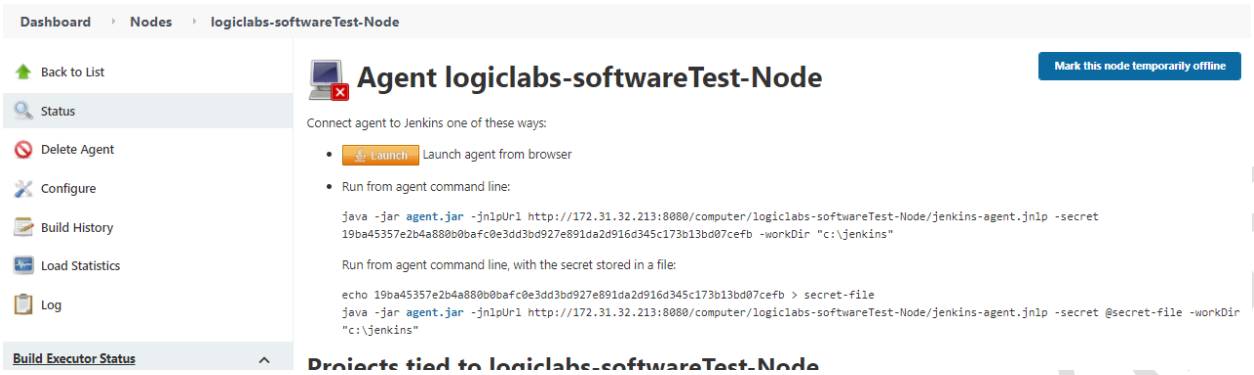
Configure the node as given below:





Add the node.

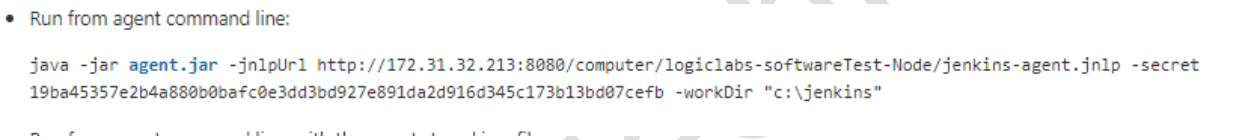
Step 6.3 Connect the node with the master



Initially the node will be offline and we need to configure it to bring it online.

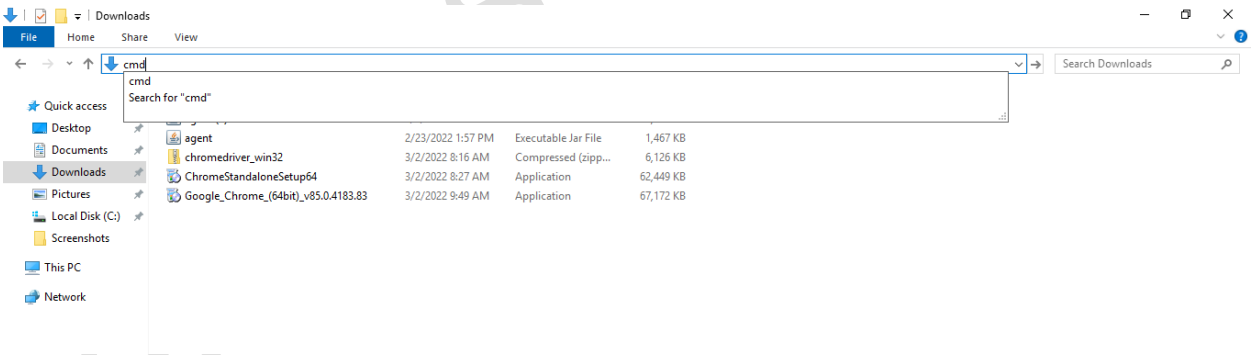
- Login to the windows slave server and open the jenkins url. Go to the manage nodes and cloud you should be able to see the node.

- Under Run agent from command line option, click on agent.jar. It will download into to the downloads folder of your server.



Open command prompt from downloads folder (open downloads folder, press

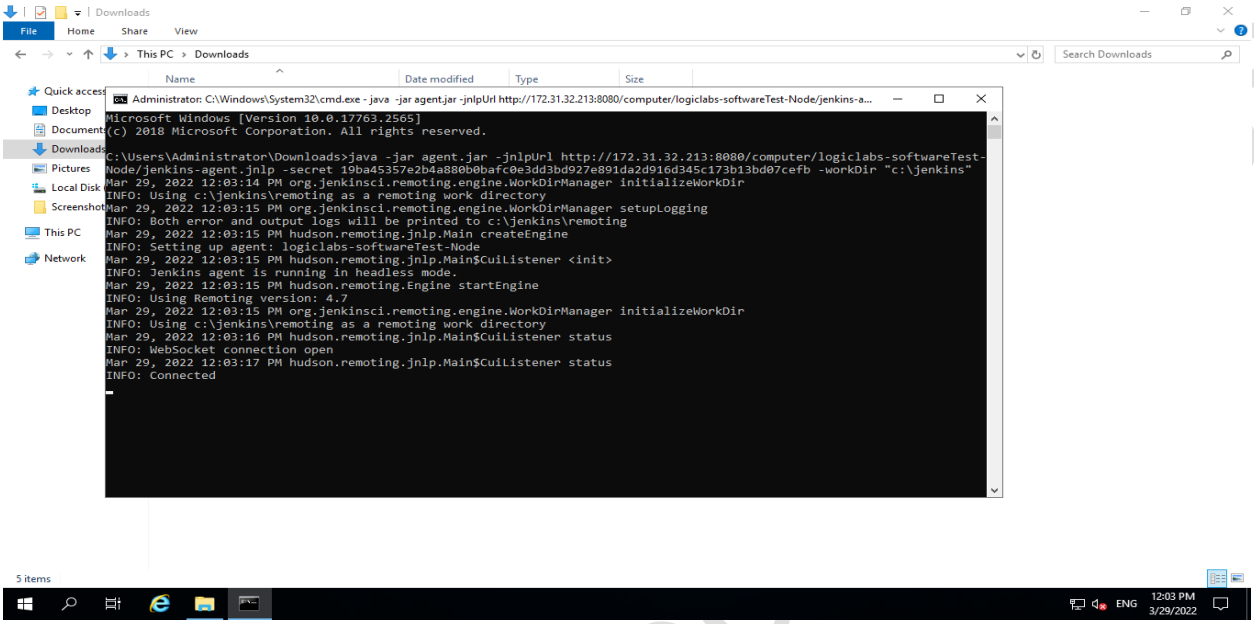
Alt + d → clear the contents and enter cmd and press enter)



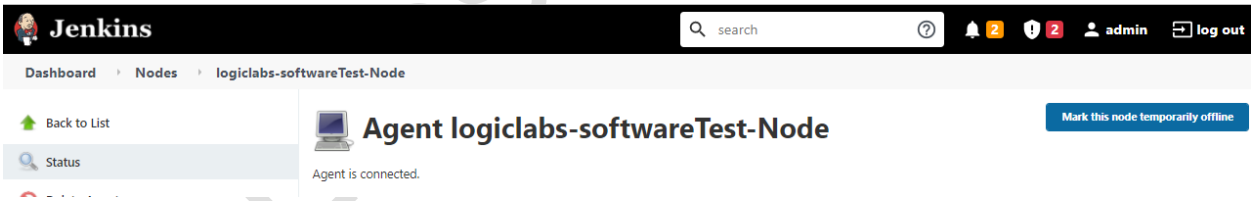
Run the command given below the Run from agent command line prompt.

For eg:
java -jar agent.jar -jnlpUrl http://172.31.32.213:8080/computer/logiclabs-softwareTest-Node/jenkins-agent.jnlp -secret

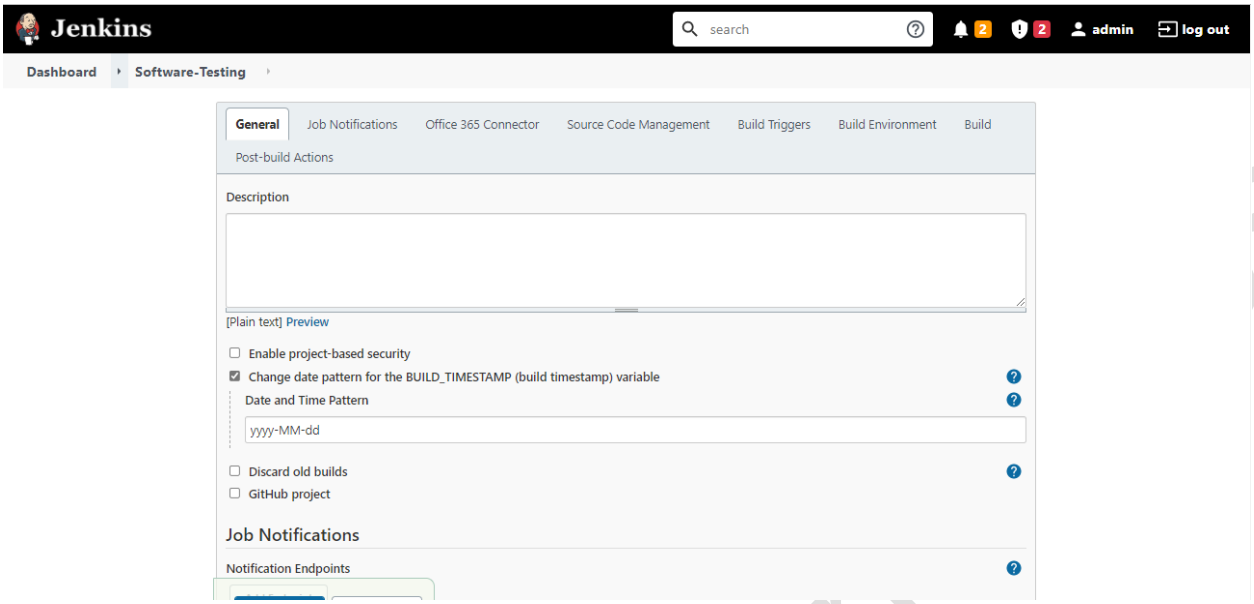
19ba45357e2b4a880b0bafc0e3dd3bd927e891da2d916d345c173b13bd07cefb -workDir "c:\jenkins"

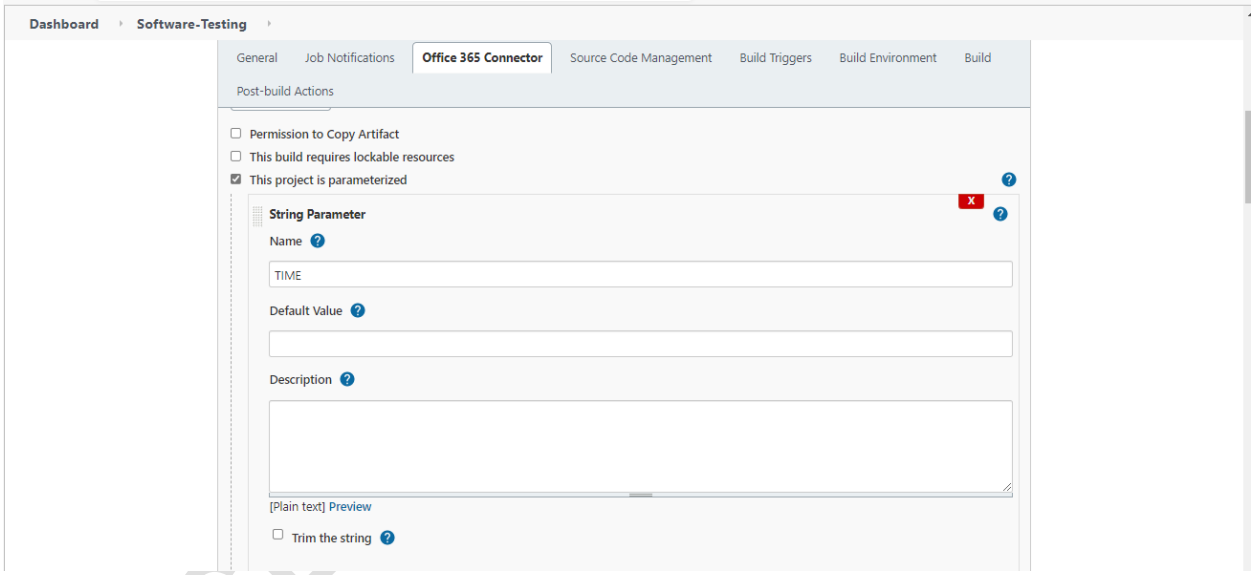


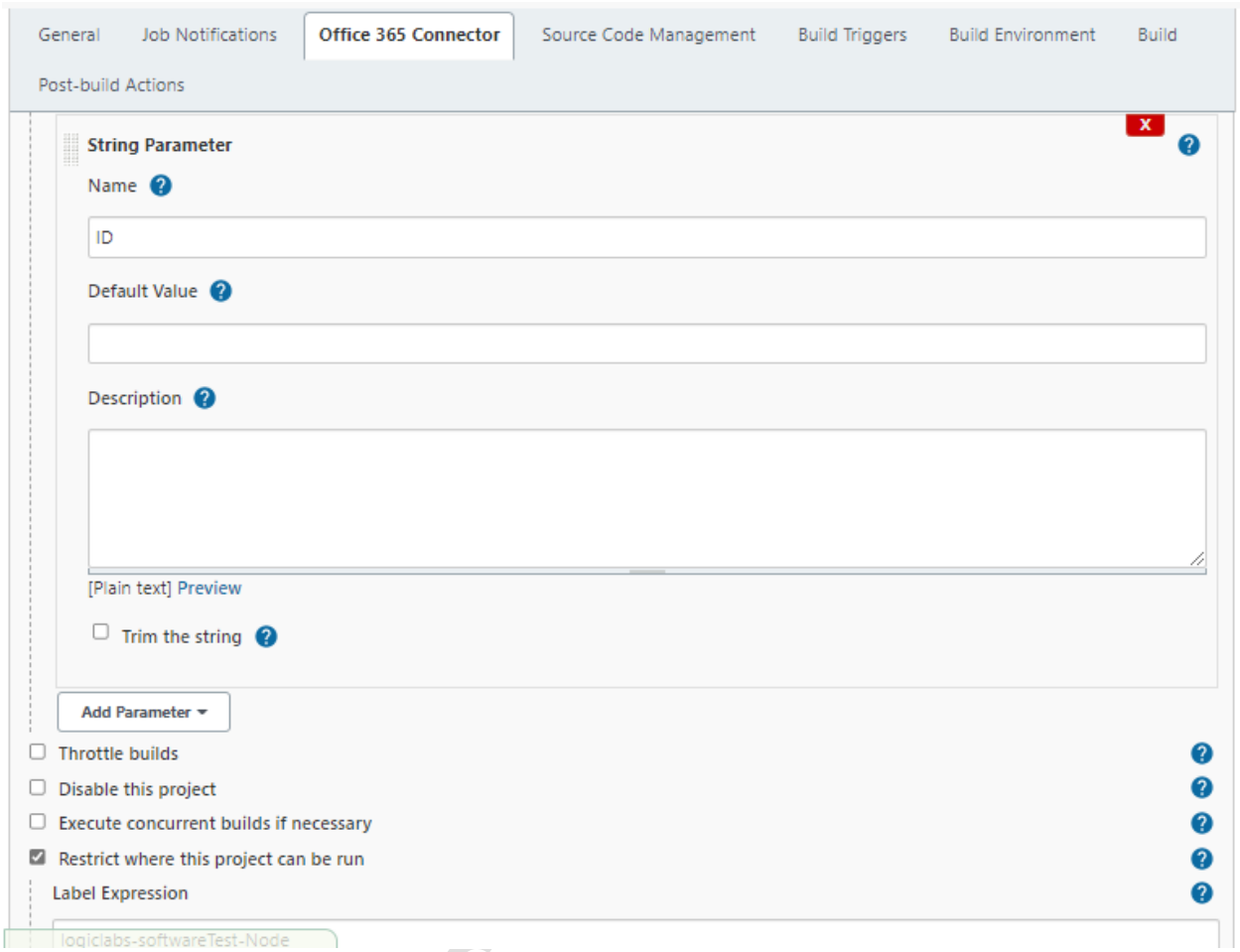
Now your node should look online.



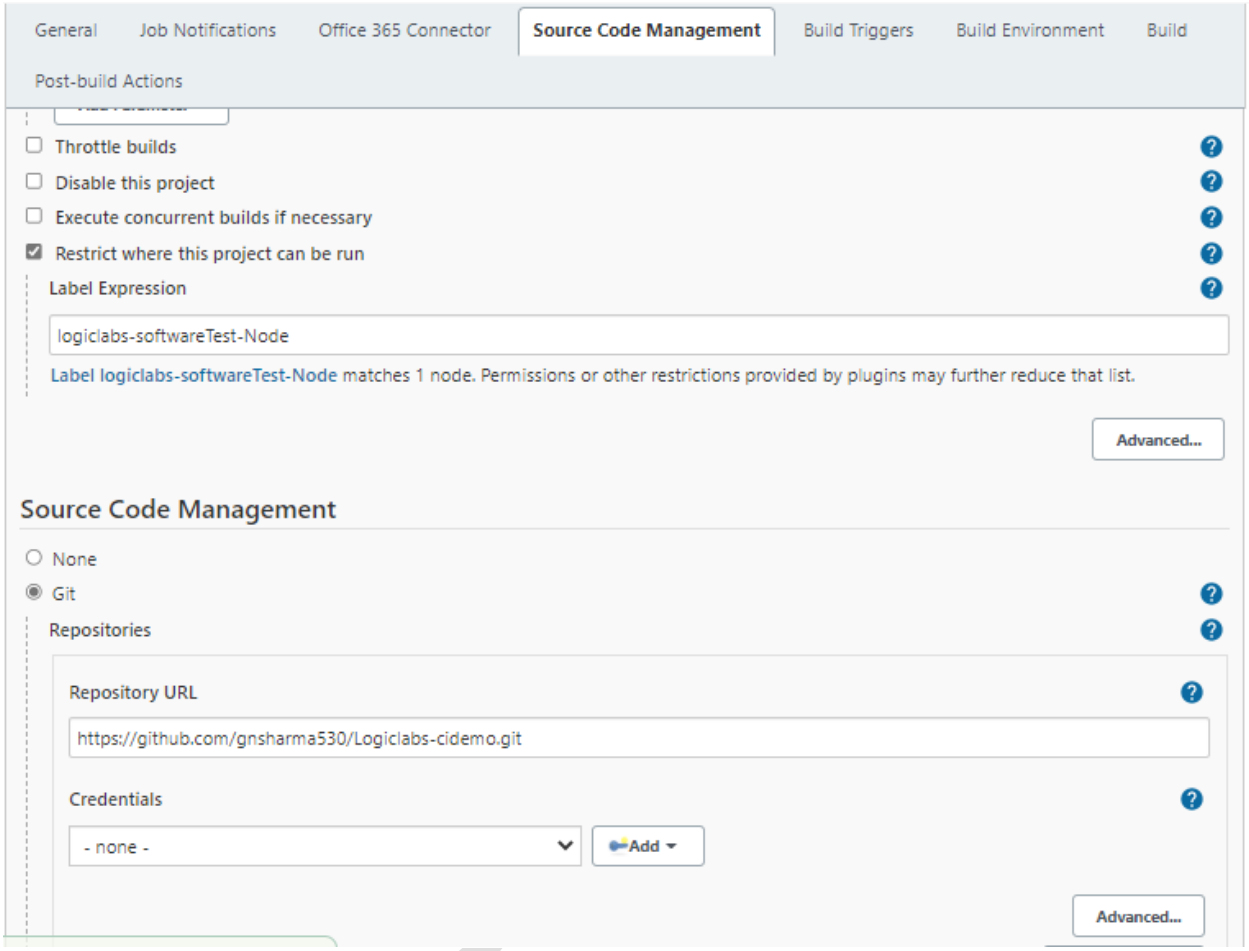
Step 6.4 Create Jenkins job to run Selenium software test cases


Jenkins
search
admin
log out
Dashboard
Software-Testing
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
[Plain text] Preview
Enable project-based security
Change date pattern for the BUILD_TIMESTAMP (build timestamp) variable
Date and Time Pattern
yyyy-MM-dd
Discard old builds
GitHub project
Job Notifications
Notification Endpoints

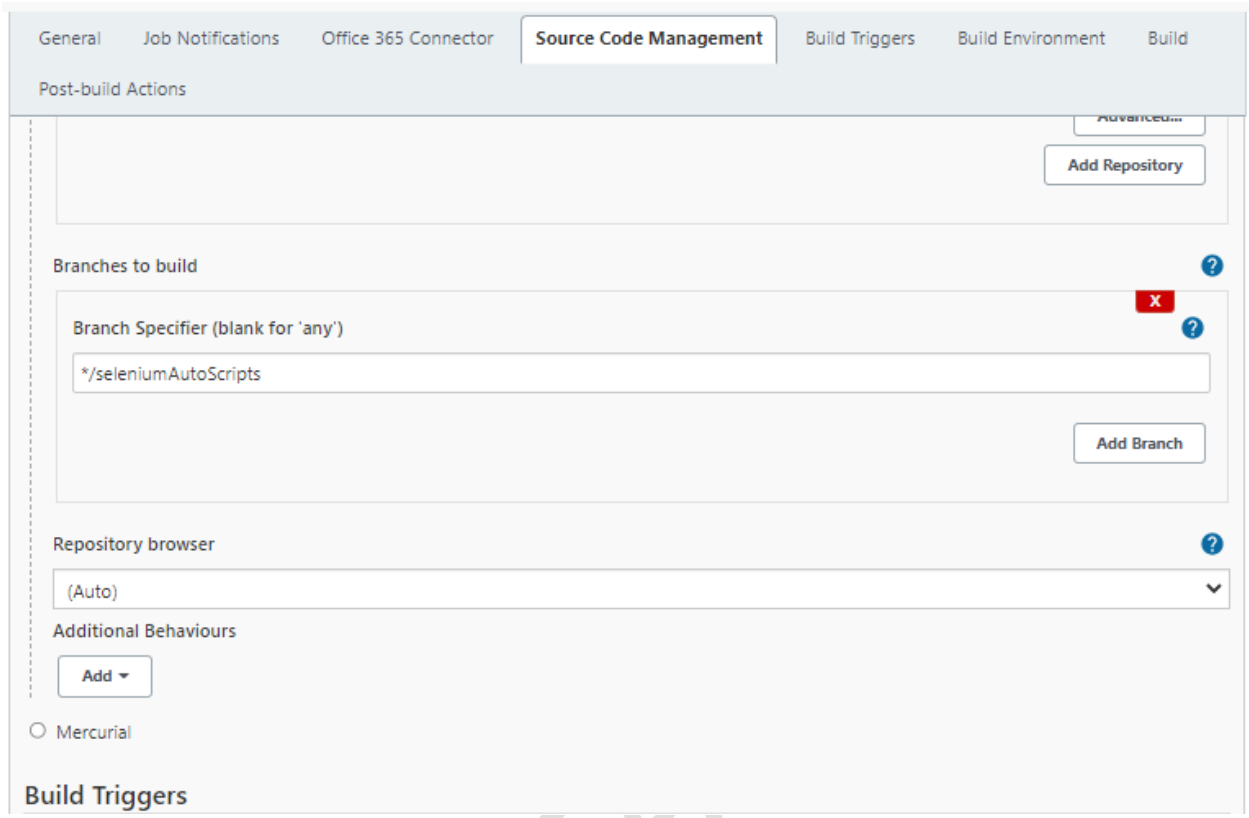

Dashboard
Software-Testing
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Permission to Copy Artifact
This build requires lockable resources
This project is parameterized
String Parameter
Name
TIME
Default Value
Description
[Plain text] Preview
Trim the string

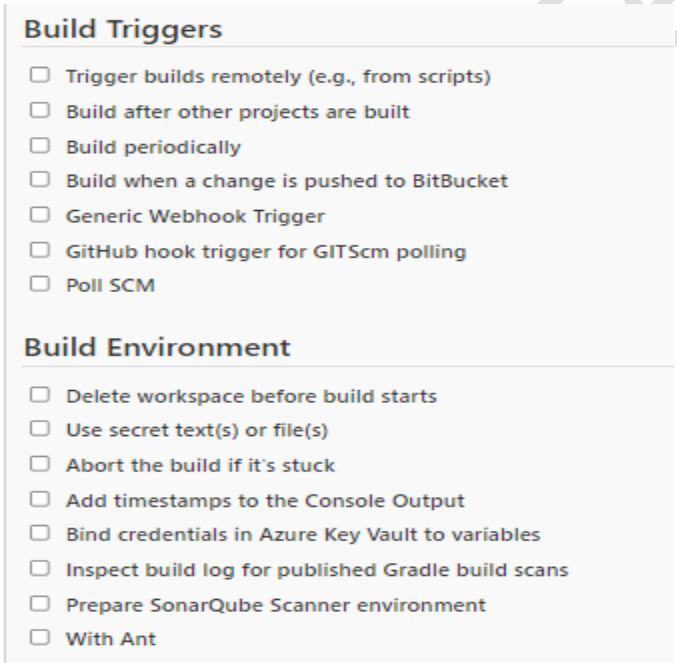


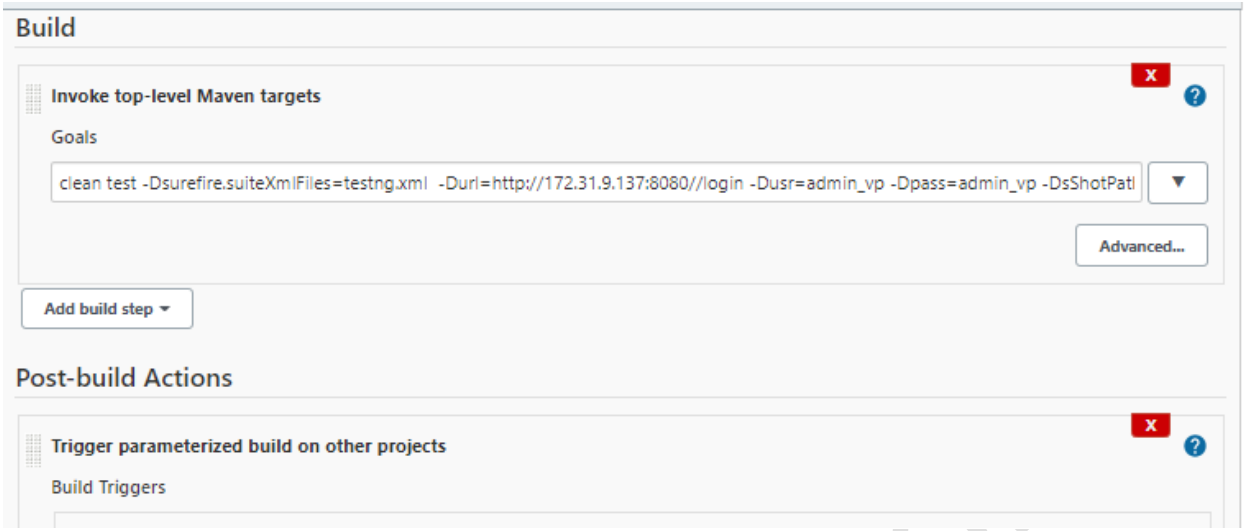
Enable restrict where this project can be run and select windows node.



Change the branch to */seleniumAutoScripts.


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/seleniumAutoScripts
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Mercurial
Build Triggers


Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Build when a change is pushed to BitBucket
Generic Webhook Trigger
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Bind credentials in Azure Key Vault to variables
Inspect build log for published Gradle build scans
Prepare SonarQube Scanner environment
With Ant



Use the below Maven goal in the build section:

clean test -Dsurefire.suiteXmlFiles=testng.xml -Durl=http://*172.31.9.137*:8080//login -Dusr=admin_vp -Dpass=admin_vp -DsShotPath=C:\\jenkins

Replace the ip with your windows server internal ip. The user name and password are the web-application hosted on staging server.

Leave the post-build actions as of now.

Test the job and check whether Selenium tests are getting executed or not.

** **Note: The most common errors and troubleshooting steps are given below:**

1. **Chrome version 85 related error:**

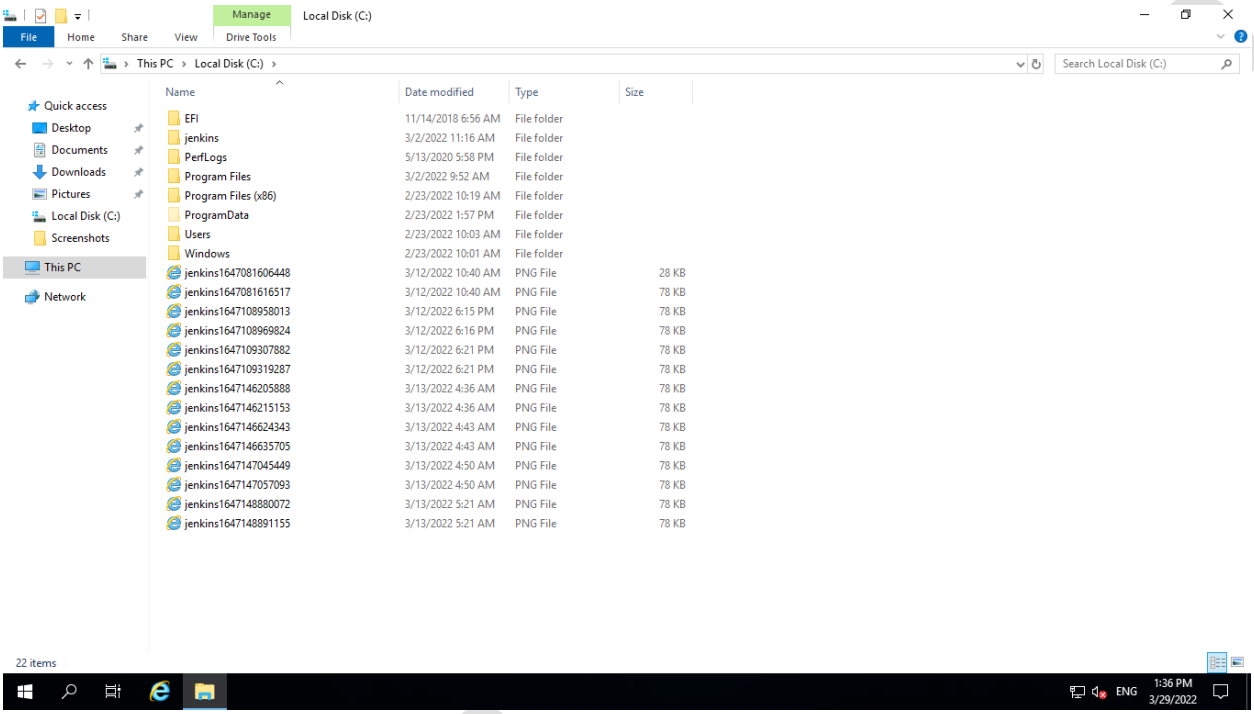
```
session not created: This version of ChromeDriver only supports Chrome version 85
  (Driver info: chromedriver=85.0.4183.87 (cd6713ebf92fa1cacc0f1a598df280093af0c5d7-refs/bran
server did not provide any stacktrace information)
```

   a. **Solution:** if windows server chrome version is latest then downgrade to exactly version 85 and disable auto update. Google to know how to downgrade the chrome version and disable updates.
2. **Ansible not able to reach Windows**

a. **Check security groups**

3.

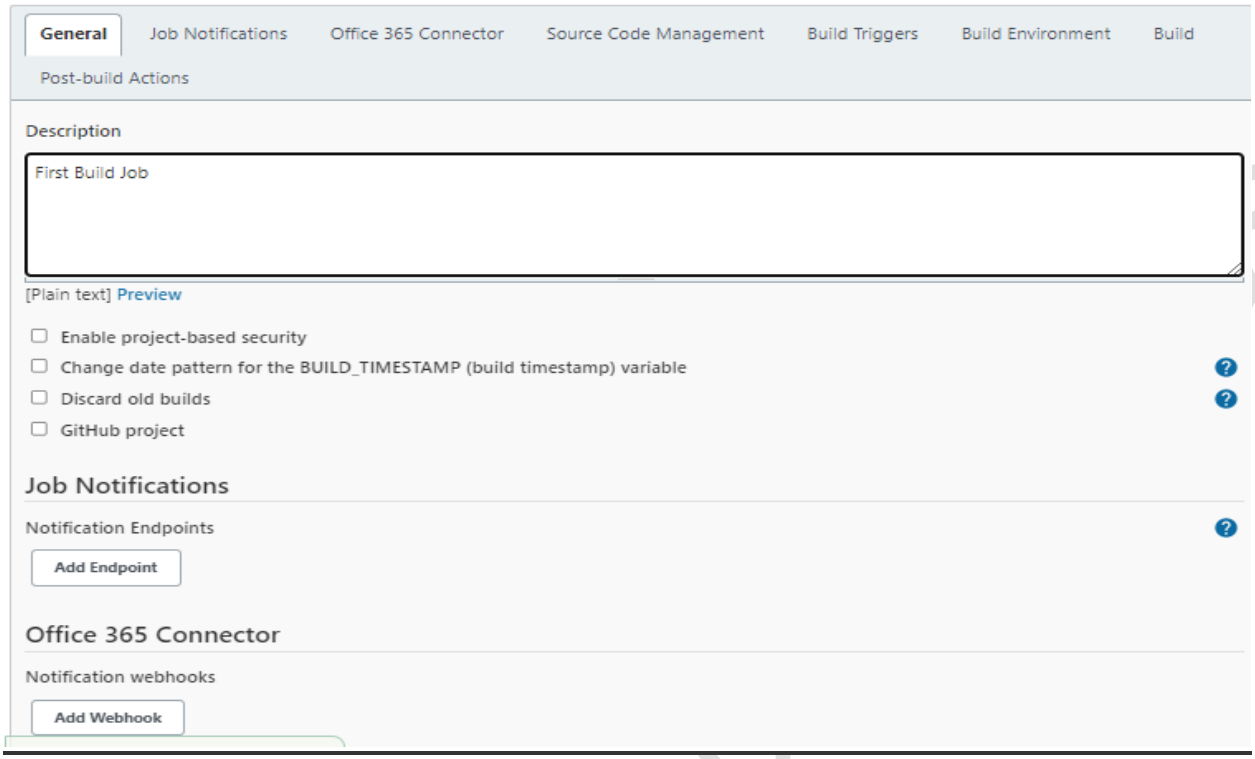
Verify whether test cases are running properly by logging in to windows server and check whether screenshots are saved at the C drive.

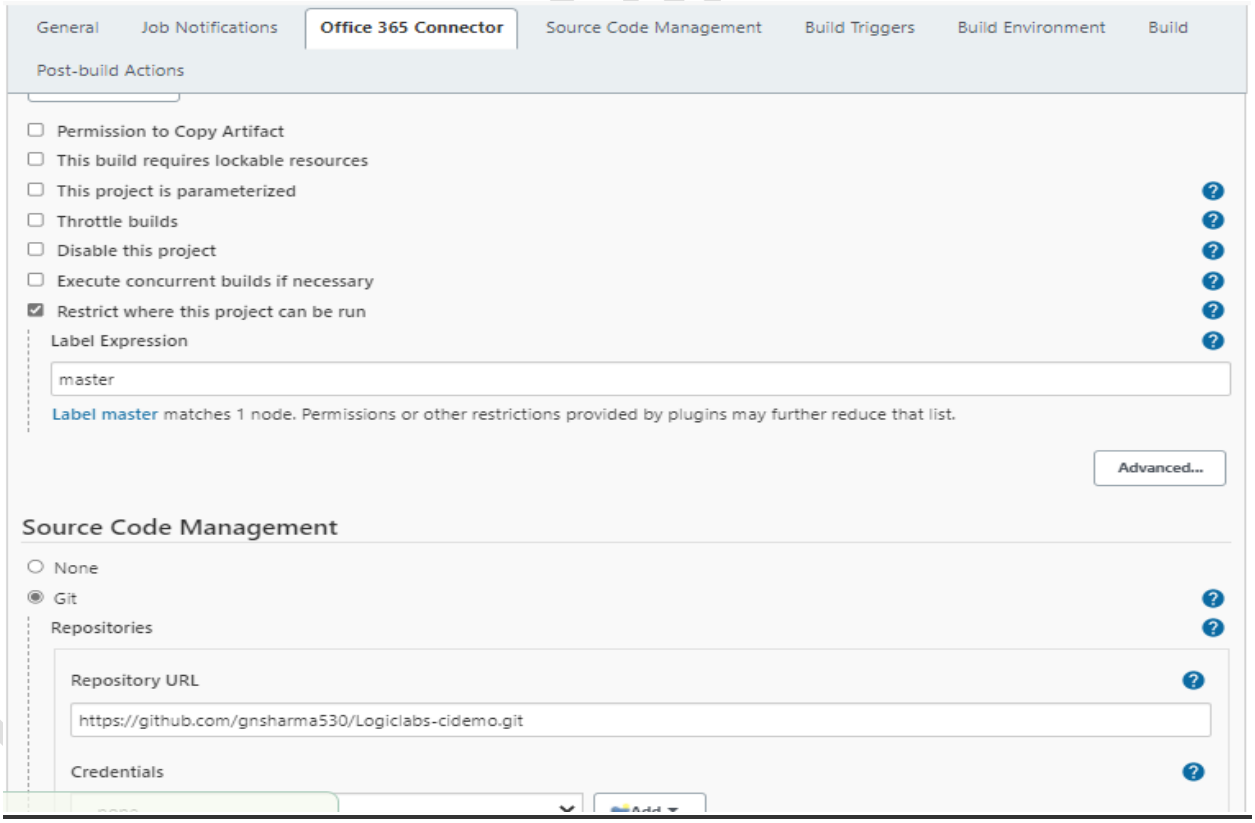


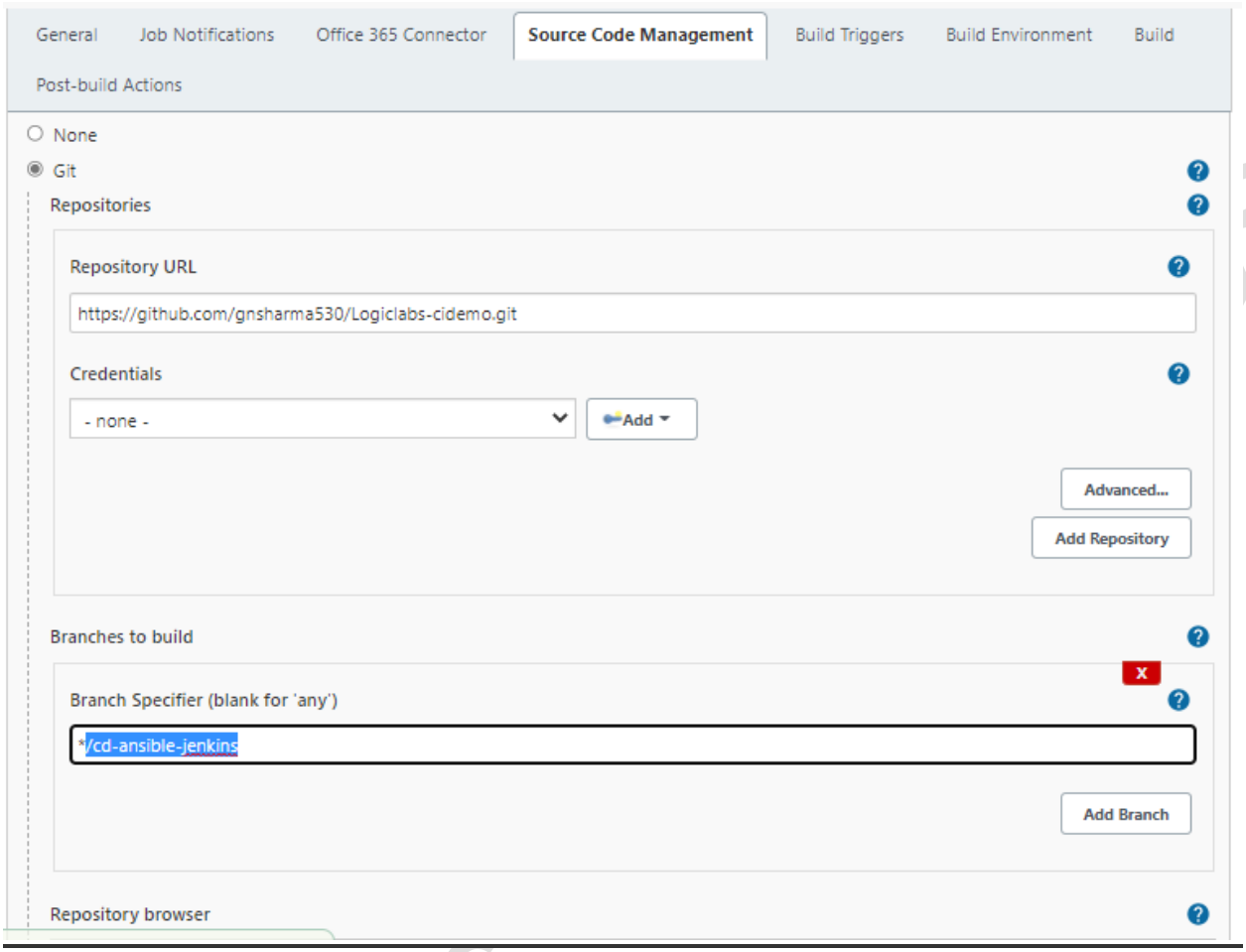
Step -7 Configuring other Jenkins jobs for Continuous Delivery

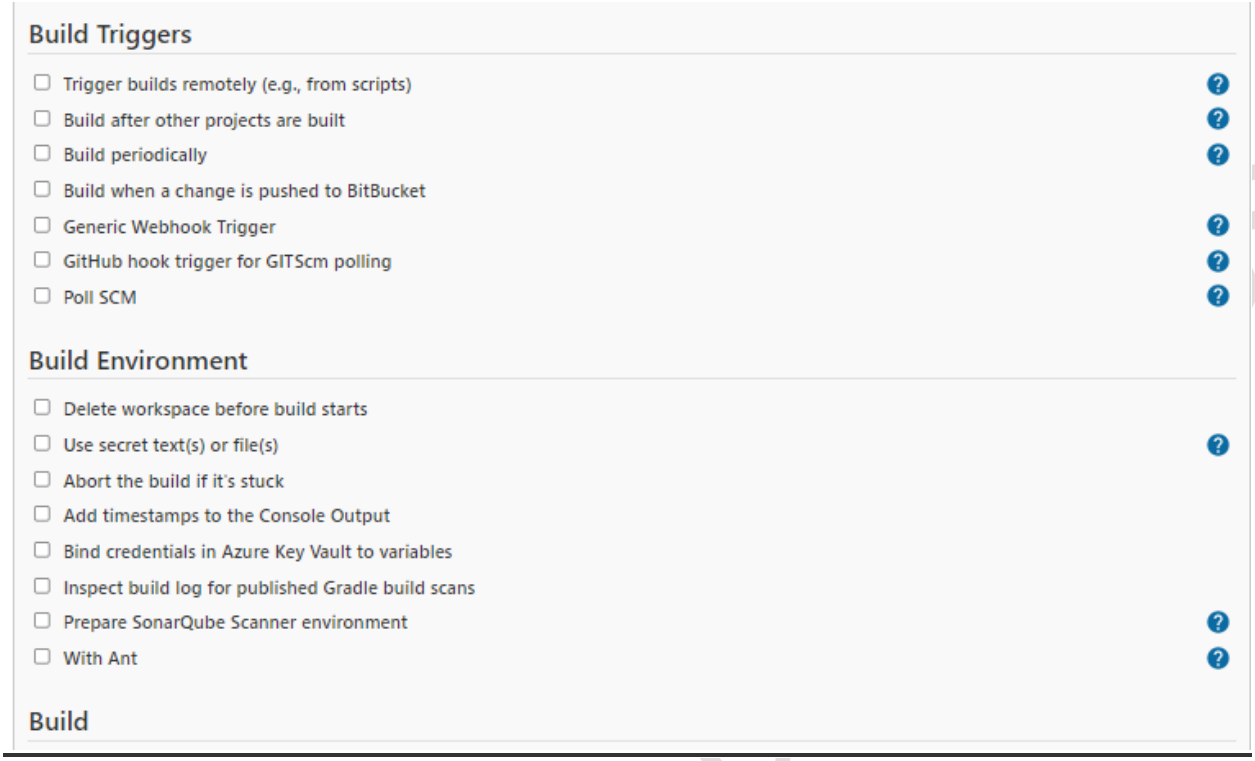
We have used the “ci-jenkins” branch to build jenkins jobs in the Continuous Integration demonstration.
However in order to build our Continuous Delivery/ Continuous Deployment we need to build from cd-ansible-jenkins branch.

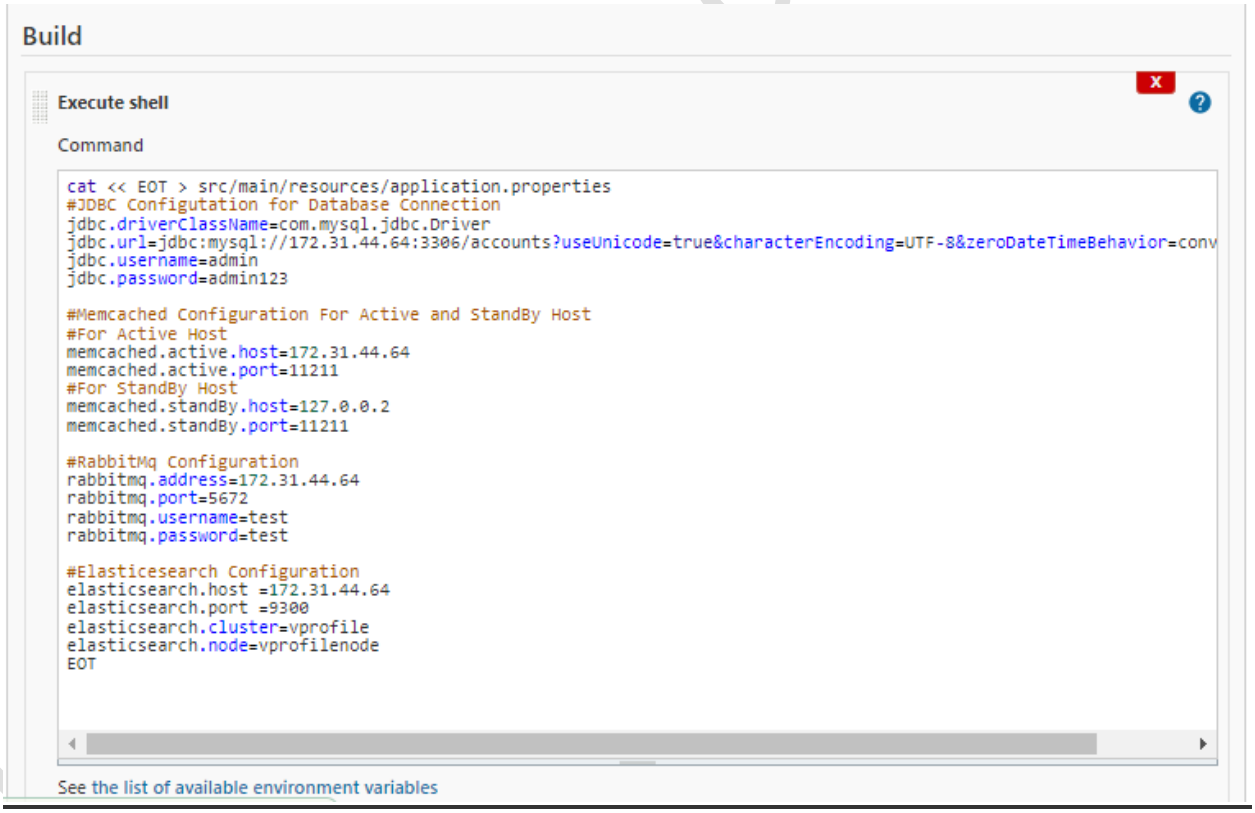
Step 7.1 Configuring Jenkins Build job for **Continuous Delivery**


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
First Build Job
[Plain text] Preview
Enable project-based security
Change date pattern for the BUILD_TIMESTAMP (build timestamp) variable
Discard old builds
GitHub project
Job Notifications
Notification Endpoints
Add Endpoint
Office 365 Connector
Notification webhooks
Add Webhook

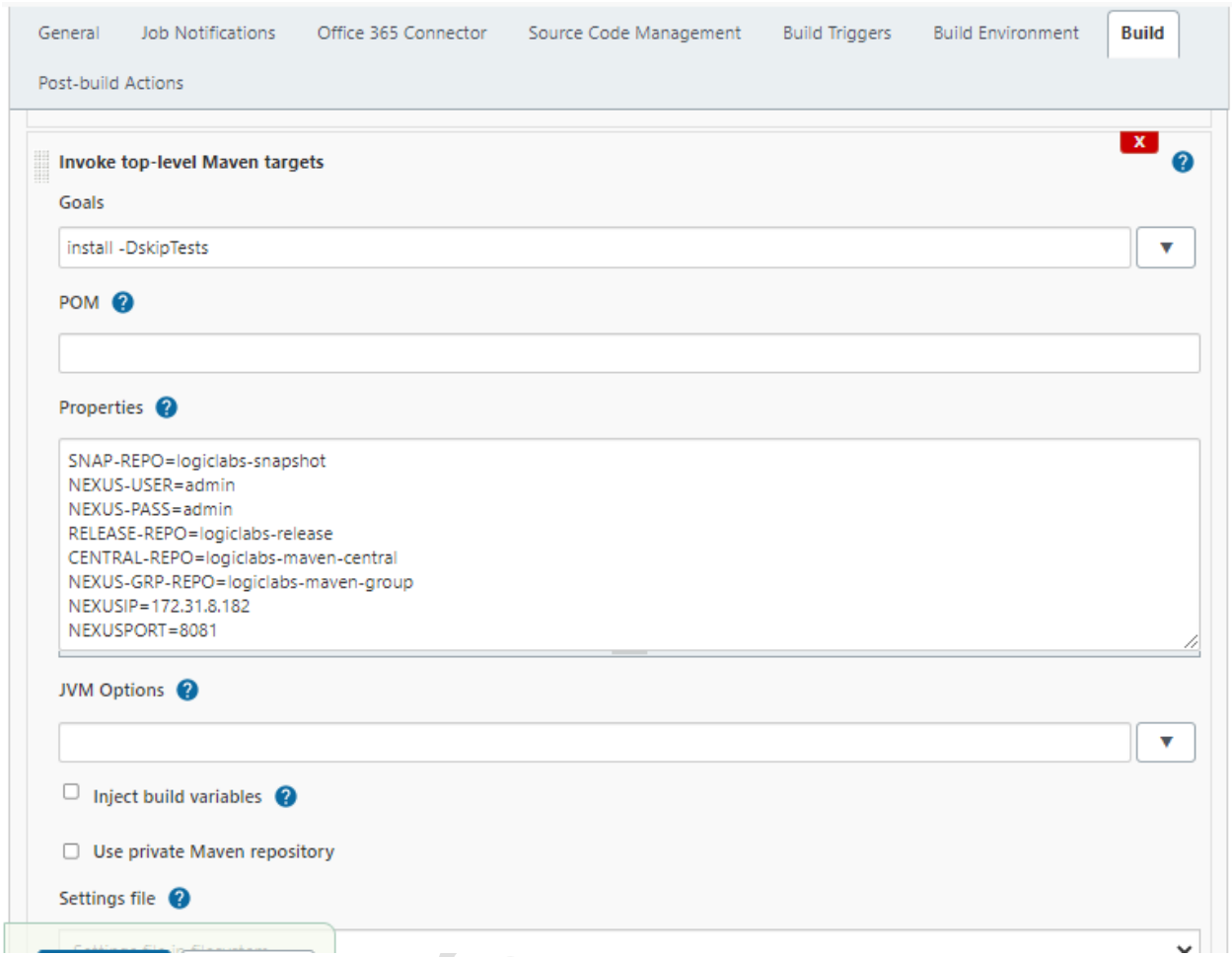

General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Permission to Copy Artifact
This build requires lockable resources
This project is parameterized
Throttle builds
Disable this project
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
master
Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.
Advanced...
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/gnsharma530/Logiclabs-cidemo.git
Credentials

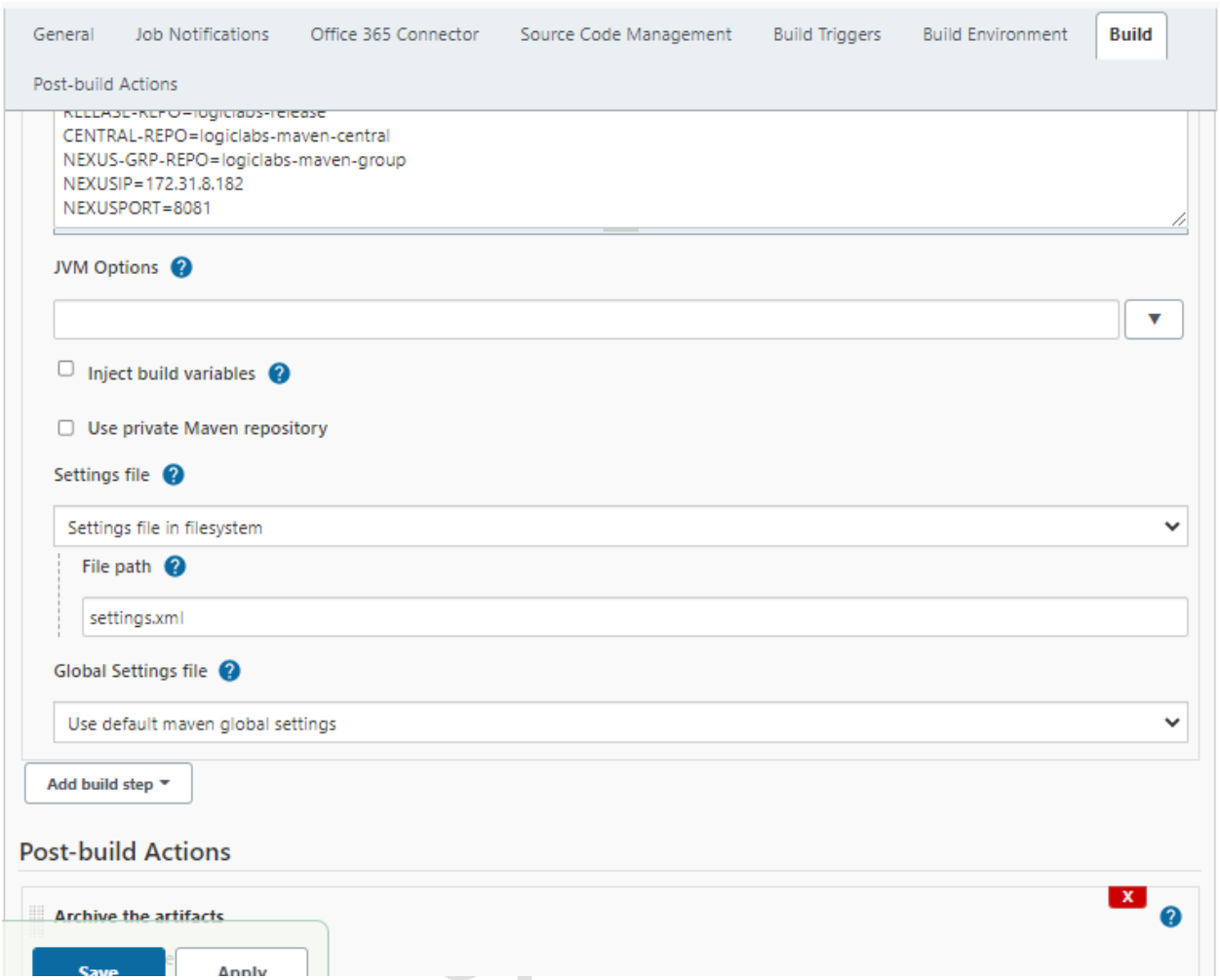

General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
None
Git
Repositories
Repository URL
https://github.com/gnsharma530/Logiclabs-cidemo.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/cd-ansible-jenkins
Add Branch
Repository browser



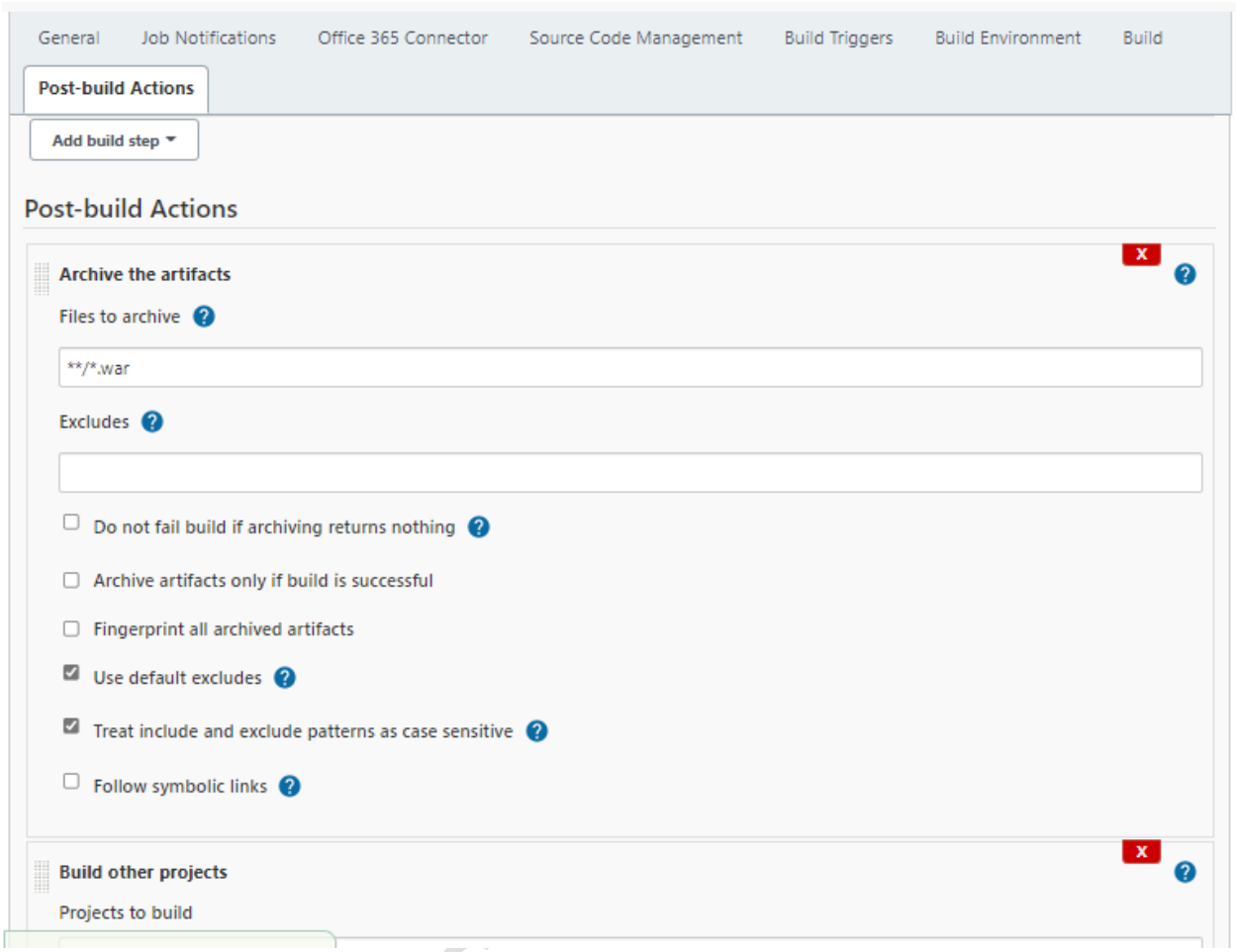


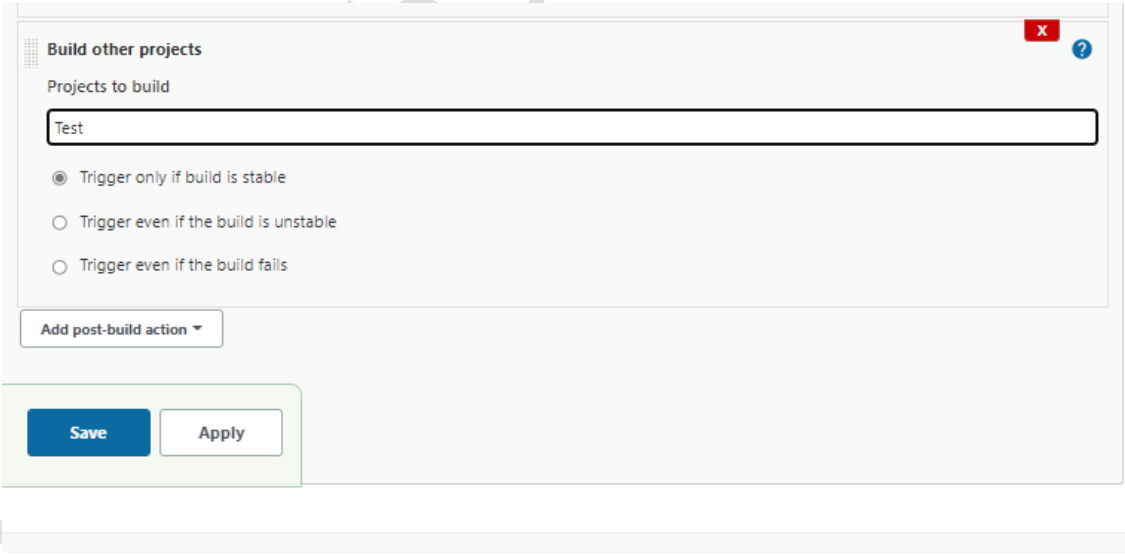
*The shell command text and explanation is provided after job configuration.


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Invoke top-level Maven targets
Goals
install -DskipTests
POM
Properties
SNAP-REPO=logiclabs-snapshot
NEXUS-USER=admin
NEXUS-PASS=admin
RELEASE-REPO=logiclabs-release
CENTRAL-REPO=logiclabs-maven-central
NEXUS-GRP-REPO=logiclabs-maven-group
NEXUSIP=172.31.8.182
NEXUSPORT=8081
JVM Options
Inject build variables
Use private Maven repository
Settings file



* make sure to change the Nexus server ip.





Save the Build job with post build action to run the next job “Test”.

Configuring Application properties (which was copied in Build- shell execution step) for web application:

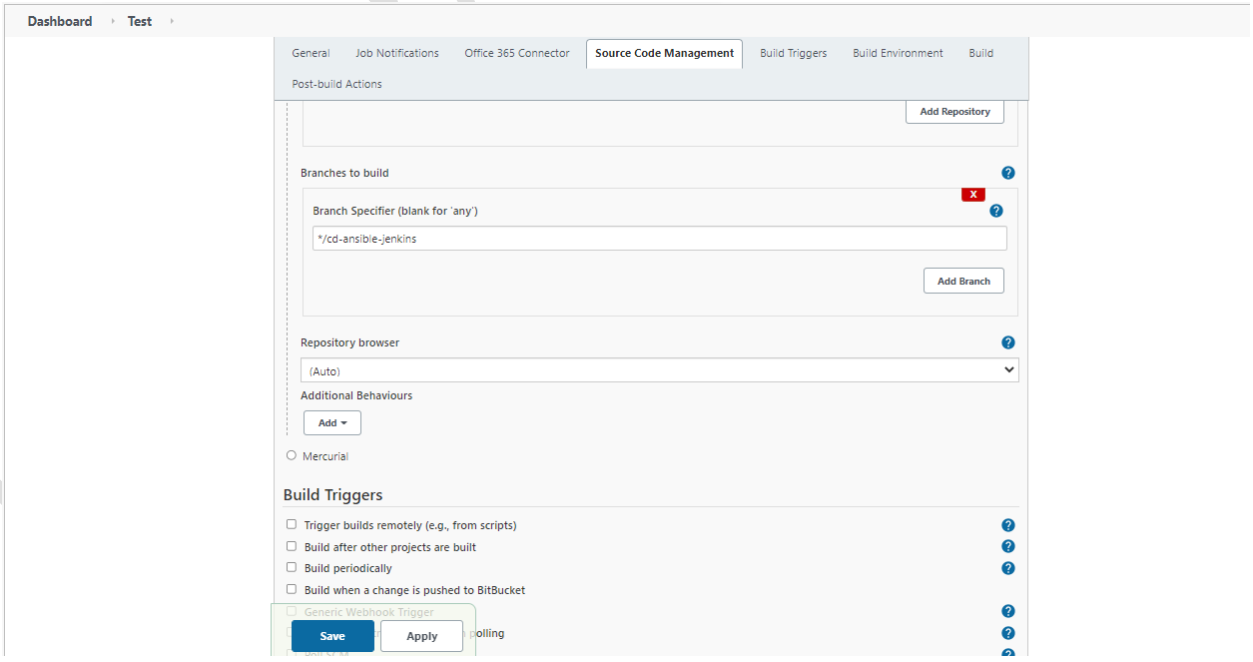
- In order for a web-application to communicate with Database, cache and message queue service like RabbitMQ, it needs some place where the details are stored.
- These details are stored in a file called application.properties file.
- In the Build step we have added the command shell that contains the details of the backend server internal ip, because the backend server is configured with Mariadb-server, Memcached and RabbitMQ .
- By building the source code with application.properties file, after deployment the artifact web-server can communicate with the backend-server.

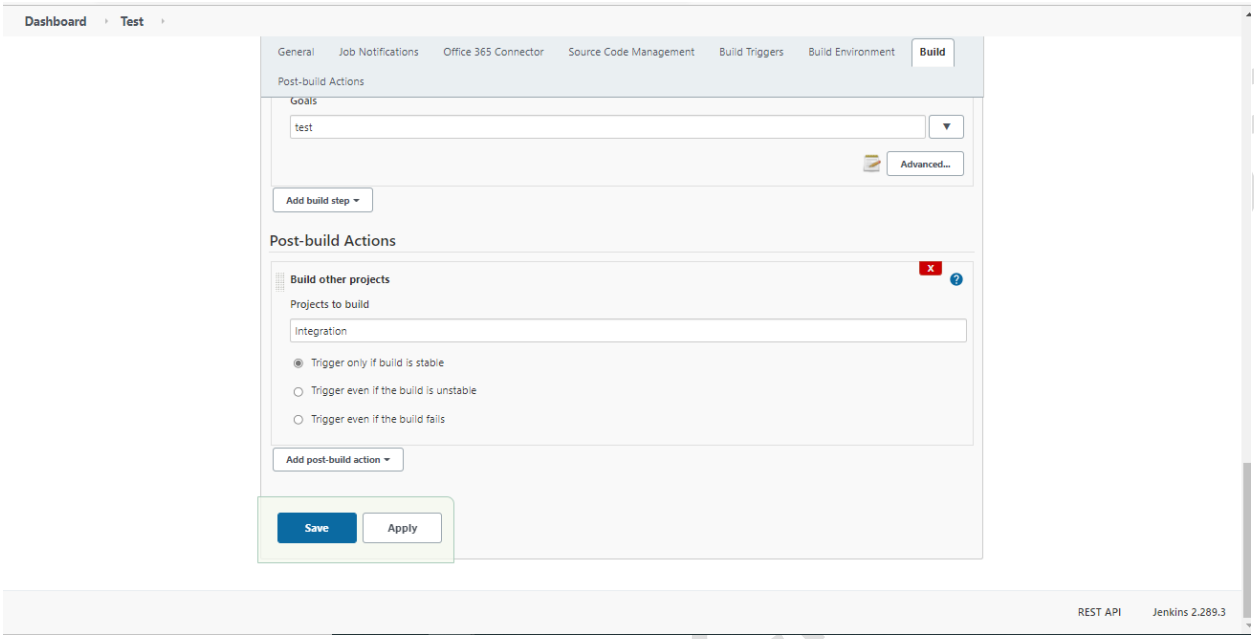
Steps to copy the application.properties file.

Open the cd-ansible-jenkins cloned repository folder and enter the folder src/main/resources. Open the application.properties file and copy the contents and paste in the execute shell command space.

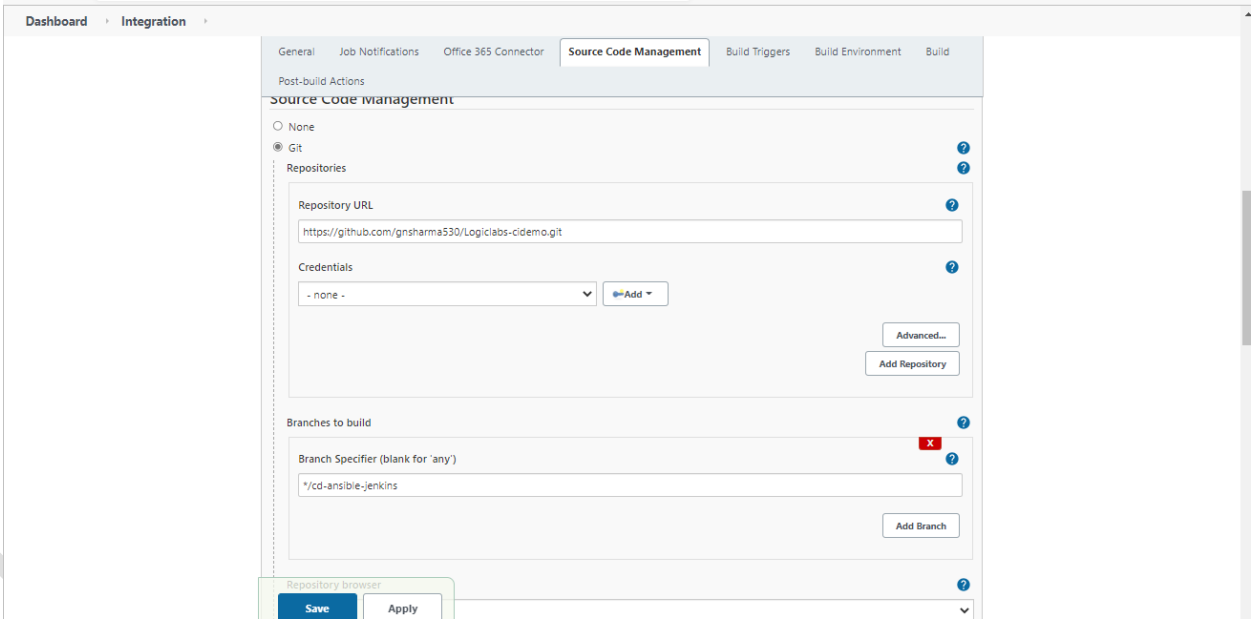
Make sure to change the ip address to your backend server ip.

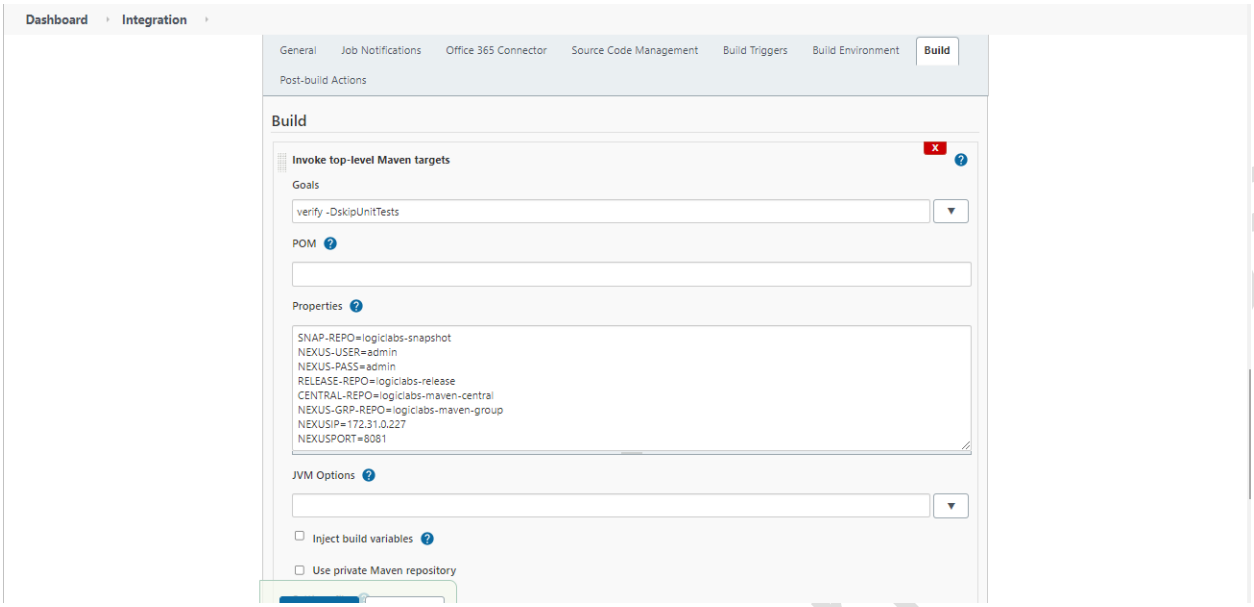
Step 7.2 : Configuring Test Job for C.D

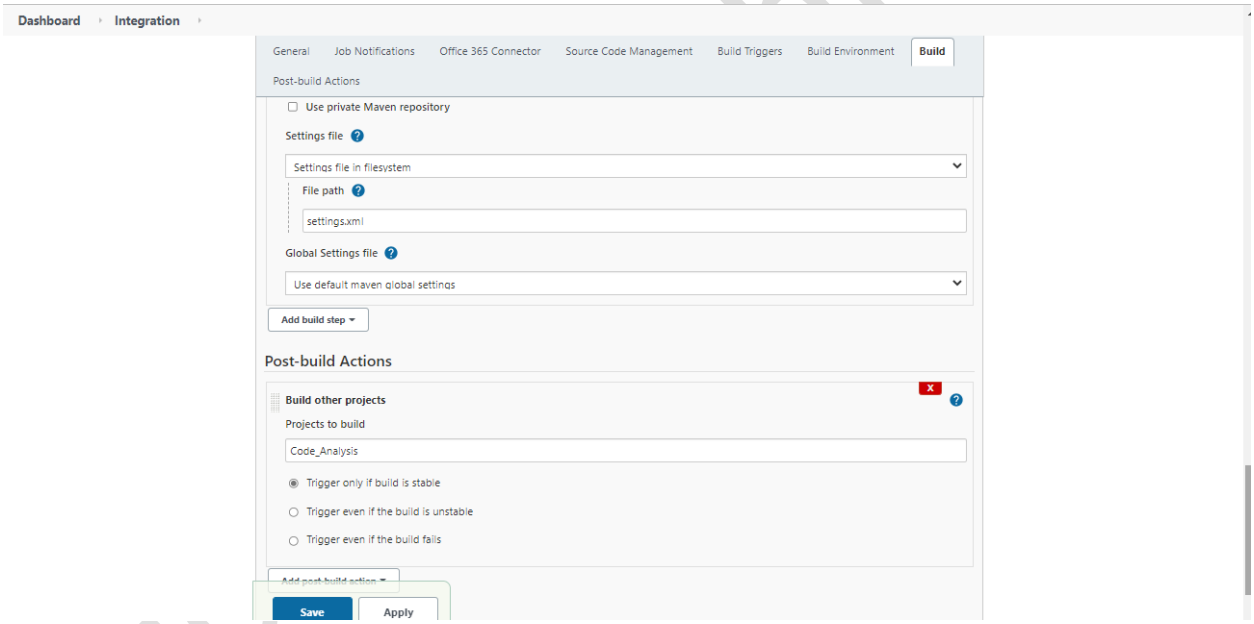




Step 7.3: Configuring Integration job for C.D



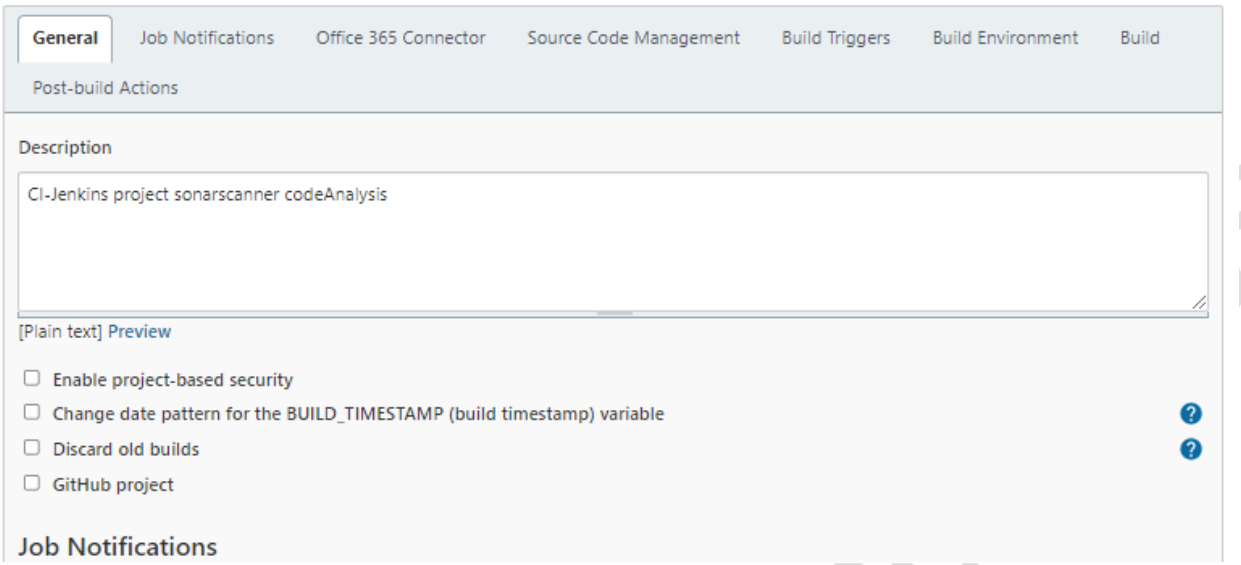


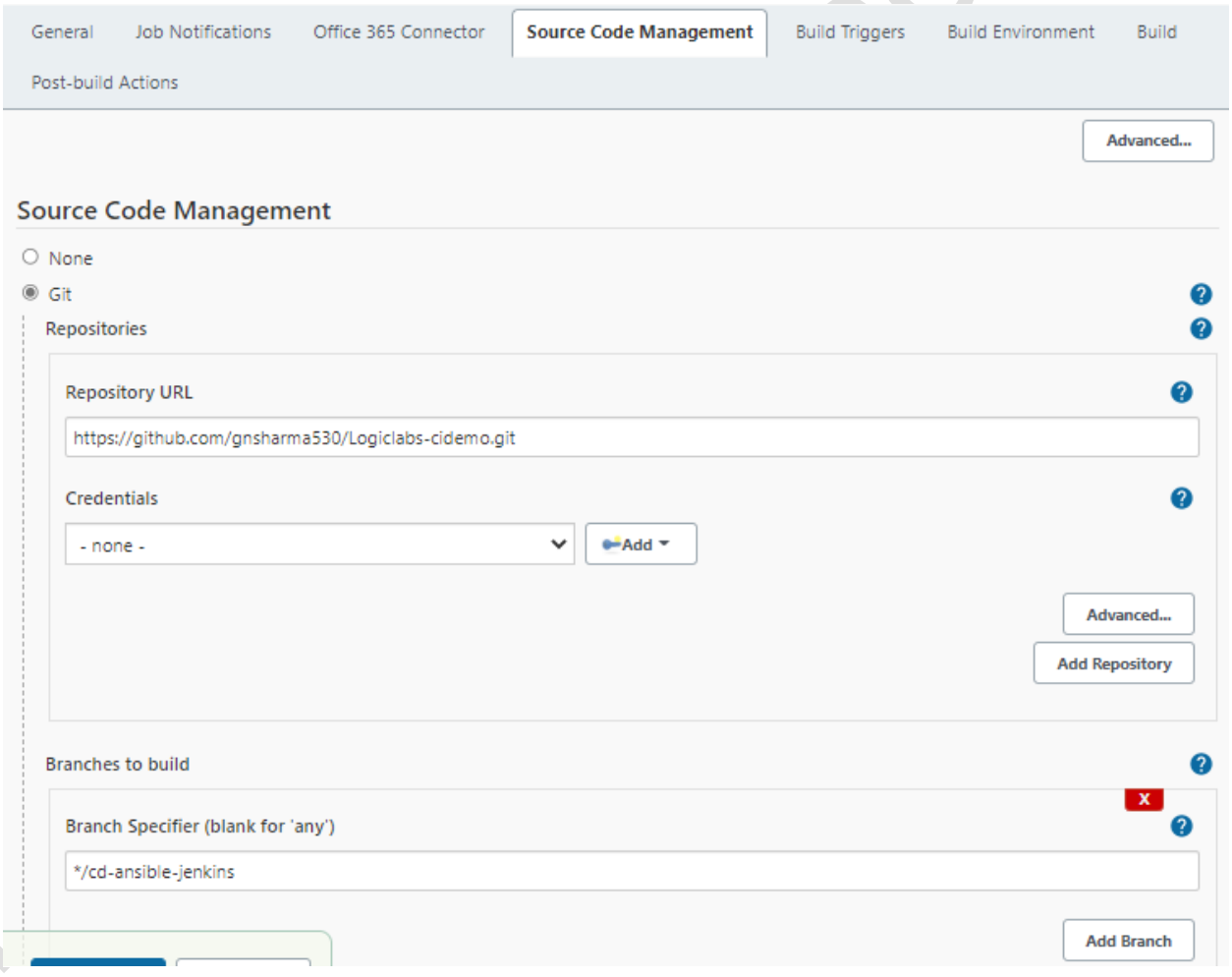


Step 7.4: Configuring Code Analysis job for C.D

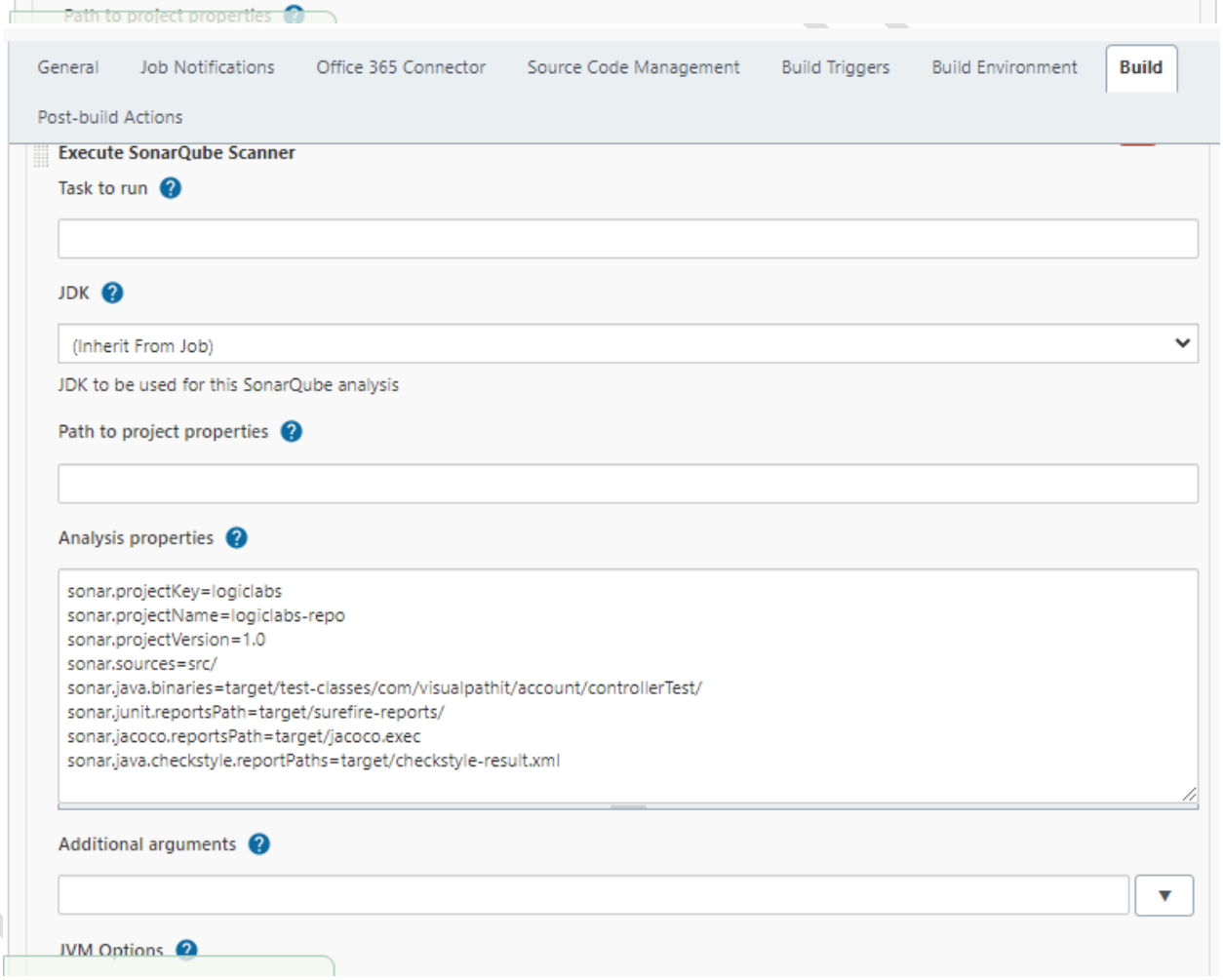
Change the branch specifier in source code management step to cd-ansible-jenkins, as you did in other jobs

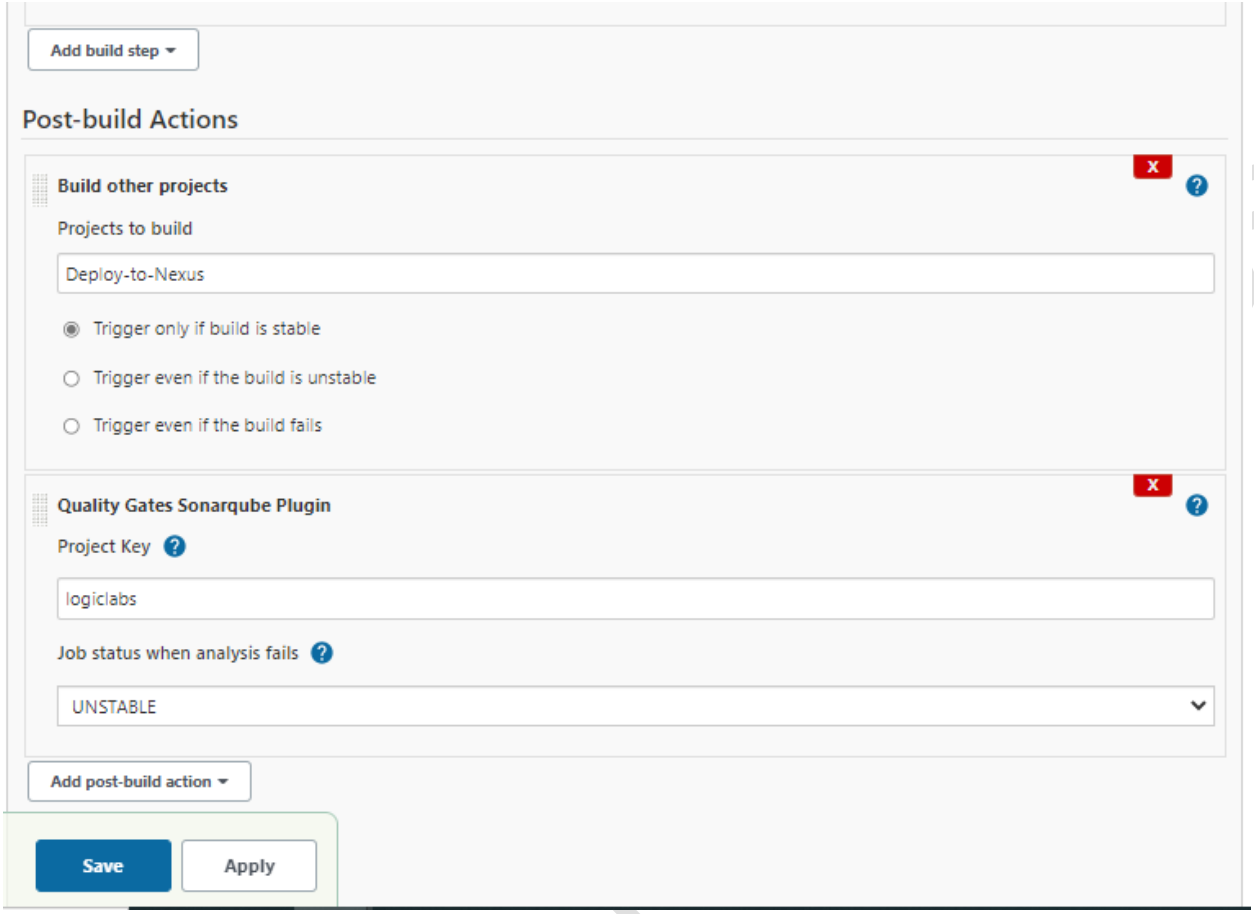
Step 7.4 Sonar Scanner code Analysis


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Description
CI-Jenkins project sonarscanner codeAnalysis
[Plain text] Preview
Enable project-based security
Change date pattern for the BUILD_TIMESTAMP (build timestamp) variable
Discard old builds
GitHub project
Job Notifications


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Advanced...
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/gnsharma530/Logiclabs-cidemo.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/cd-ansible-jenkins
Add Branch


Build
Invoke top-level Maven targets
Goals
install
Advanced...
Execute SonarQube Scanner
Task to run
JDK
(Inherit From Job)
JDK to be used for this SonarQube analysis
Path to project properties


General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Execute SonarQube Scanner
Task to run
JDK
(Inherit From Job)
JDK to be used for this SonarQube analysis
Path to project properties
Analysis properties
sonar.projectKey=logiclabs
sonar.projectName=logiclabs-repo
sonar.projectVersion=1.0
sonar.sources=src/
sonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/
sonar.junit.reportsPath=target/surefire-reports/
sonar.jacoco.reportsPath=target/jacoco.exec
sonar.java.checkstyle.reportPaths=target/checkstyle-result.xml
Additional arguments
JVM Options

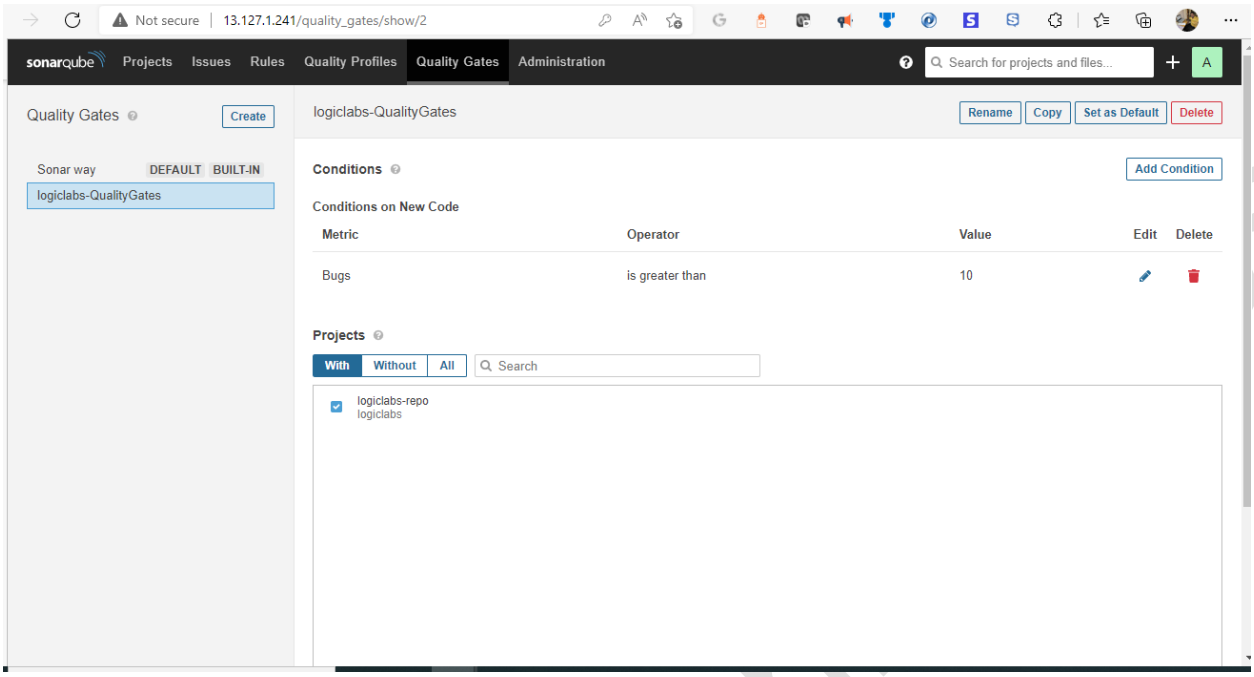


The Quality Gates sonarqube plugin has to be installed in jenkins in order to get that option in post-build action.

The project key value can be obtained from the Sonarqube server.

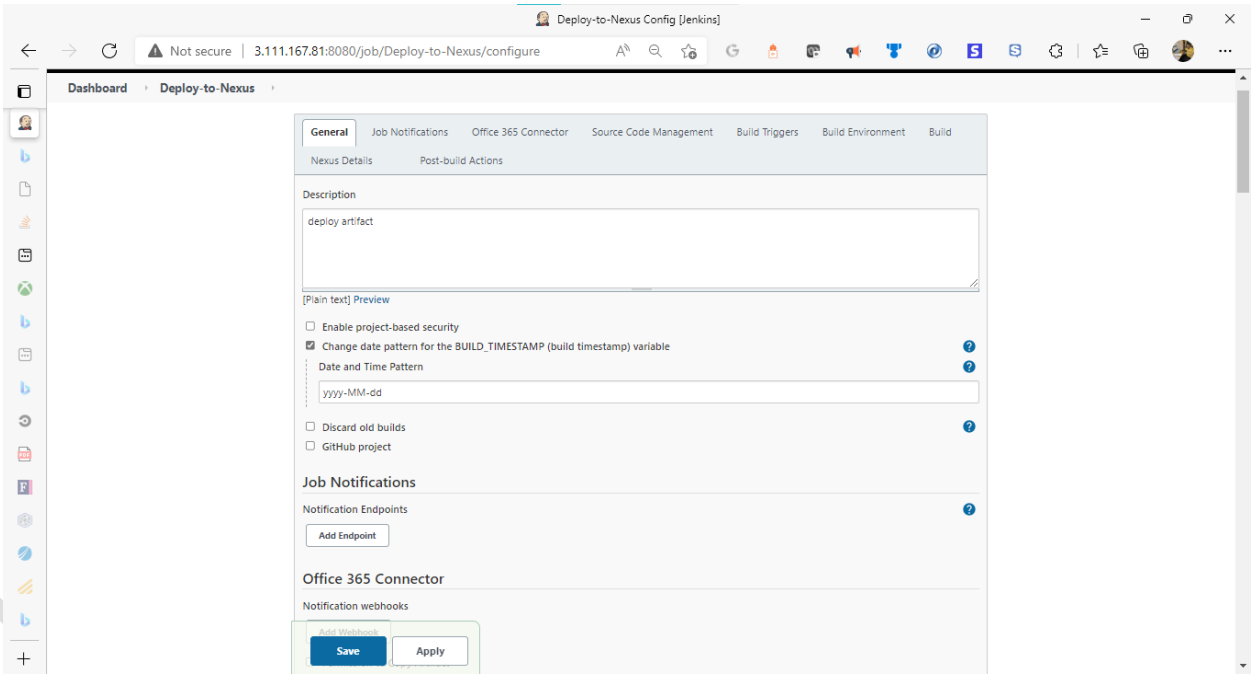
Open SonarQube server to verify the project quality gates settings:

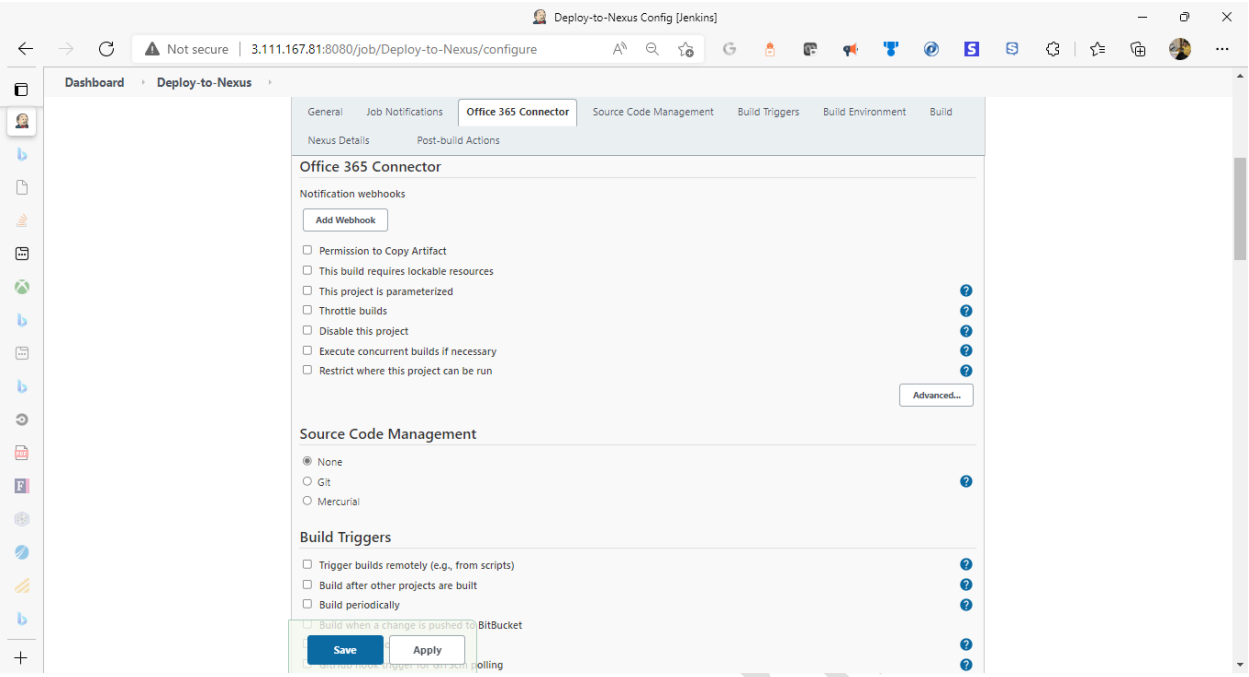
* Sonar job is optional and not a necessary ingredient for the CI/CD process.

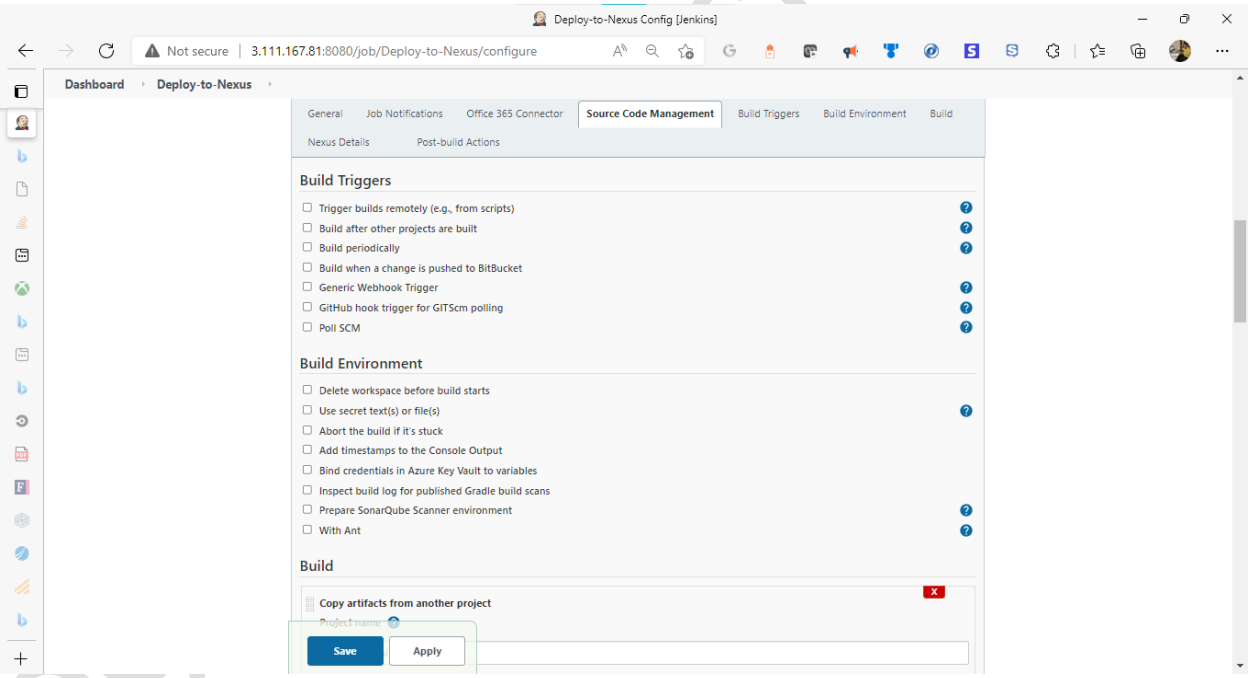


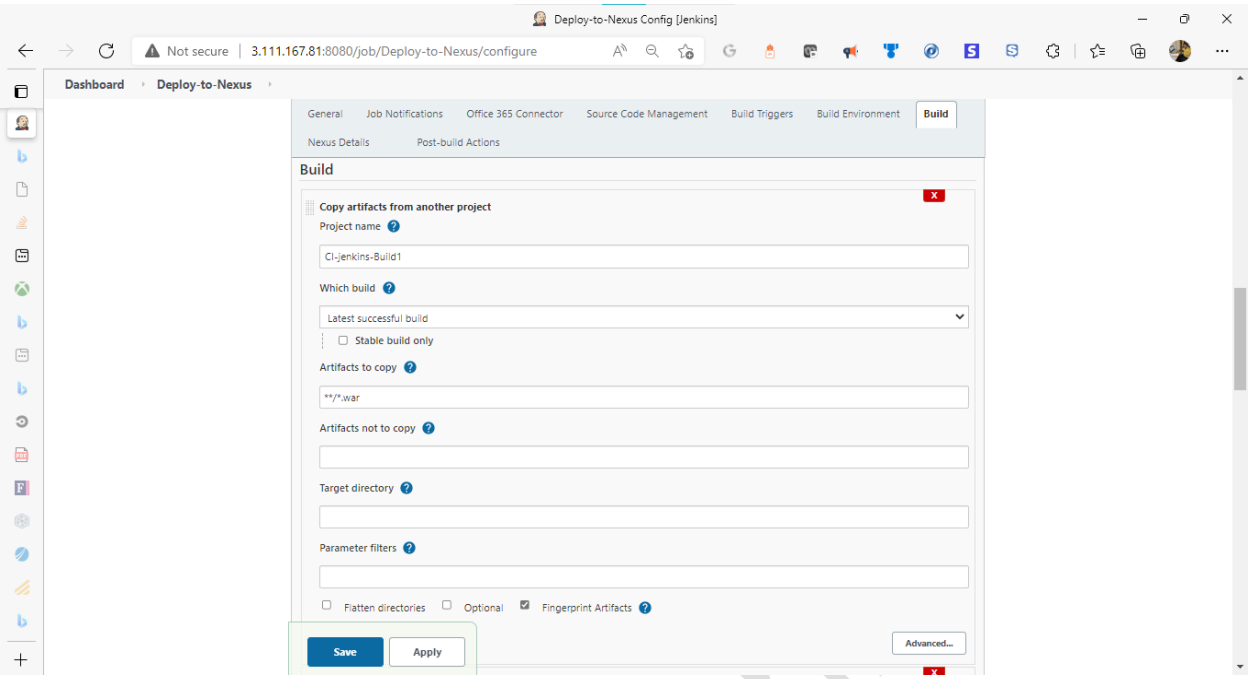
Verify the project key name in the sonar portal. If different rename accordingly

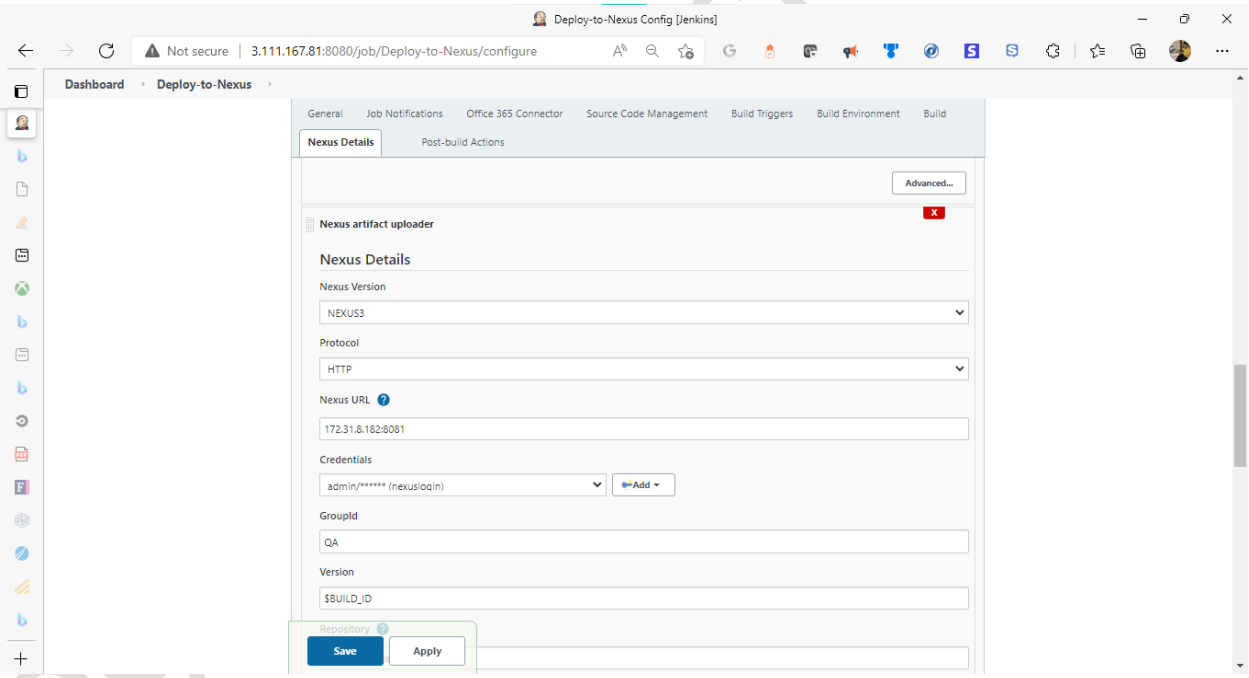
Step 7.4: Configure the job Deploy to Nexus for C.D



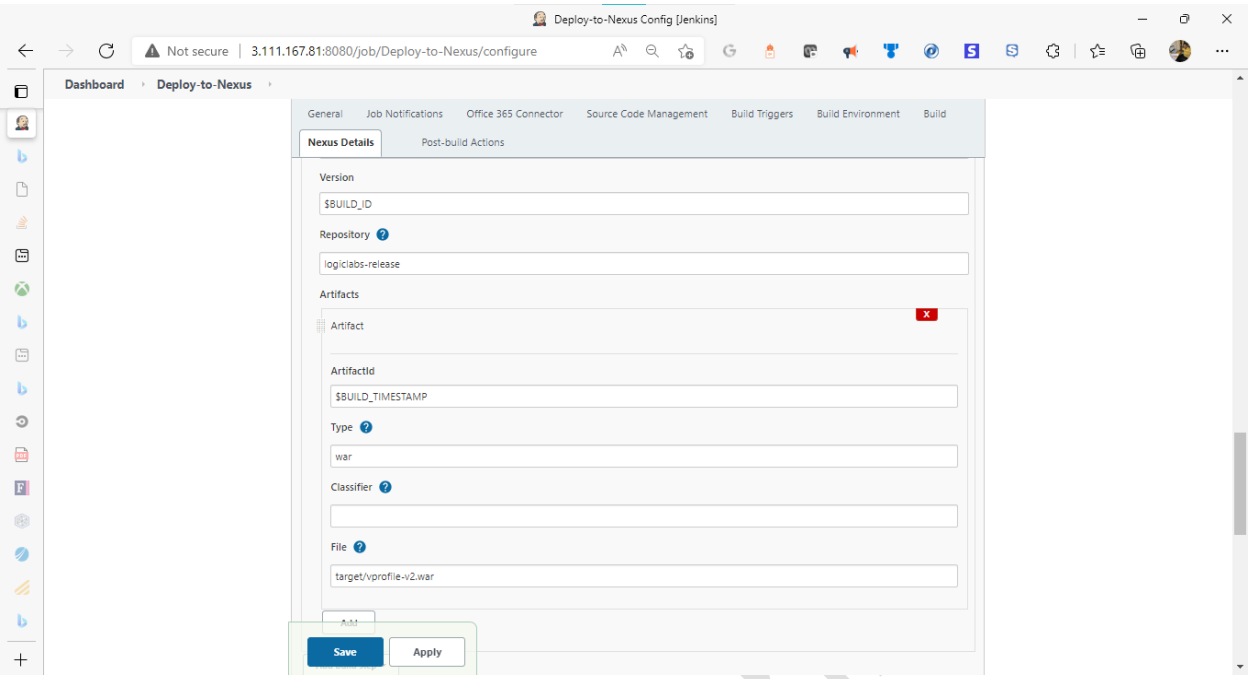

Deploy-to-Nexus Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Nexus/configure
Dashboard
Deploy-to-Nexus
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Nexus Details
Post-build Actions
Office 365 Connector
Notification webhooks
Add Webhook
Permission to Copy Artifact
This build requires lockable resources
This project is parameterized
Throttle builds
Disable this project
Execute concurrent builds if necessary
Restrict where this project can be run
Advanced...
Source Code Management
None
Git
Mercurial
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Build when a change is pushed to BitBucket
Save
Apply

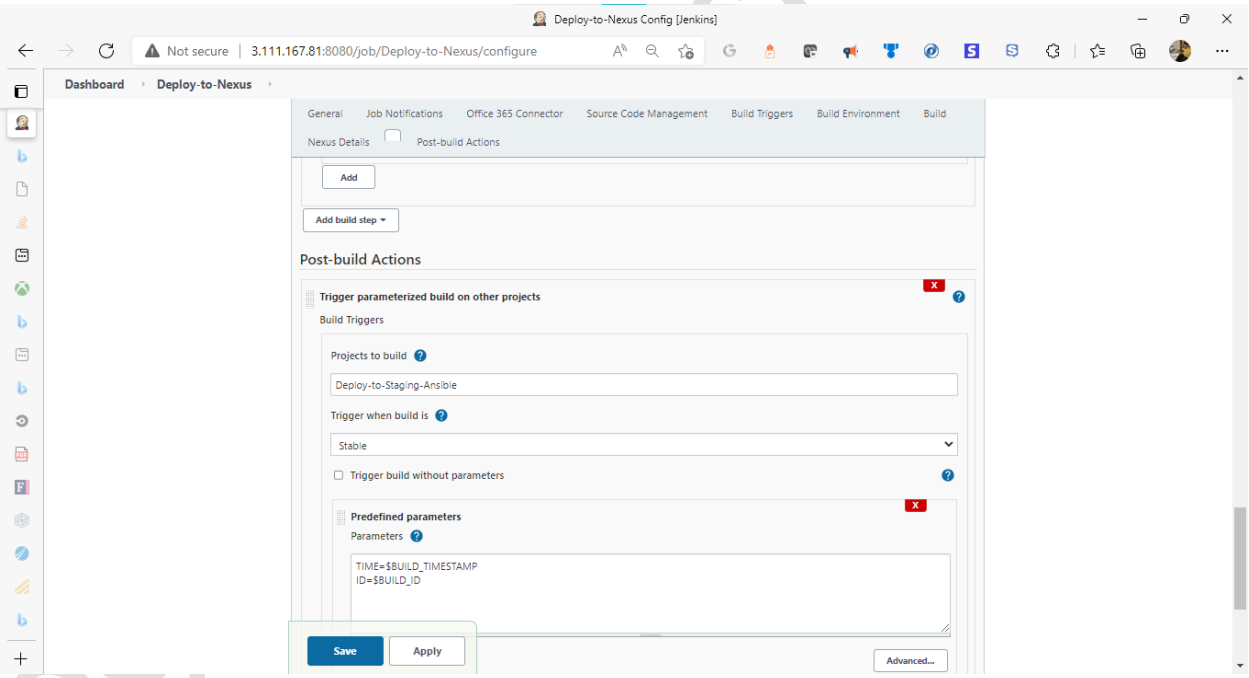

Deploy-to-Nexus Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Nexus/configure
Dashboard
Deploy-to-Nexus
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Nexus Details
Post-build Actions
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Build when a change is pushed to BitBucket
Generic Webhook Trigger
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Bind credentials in Azure Key Vault to variables
Inspect build log for published Gradle build scans
Prepare SonarQube Scanner environment
With Ant
Build
Copy artifacts from another project
Save
Apply

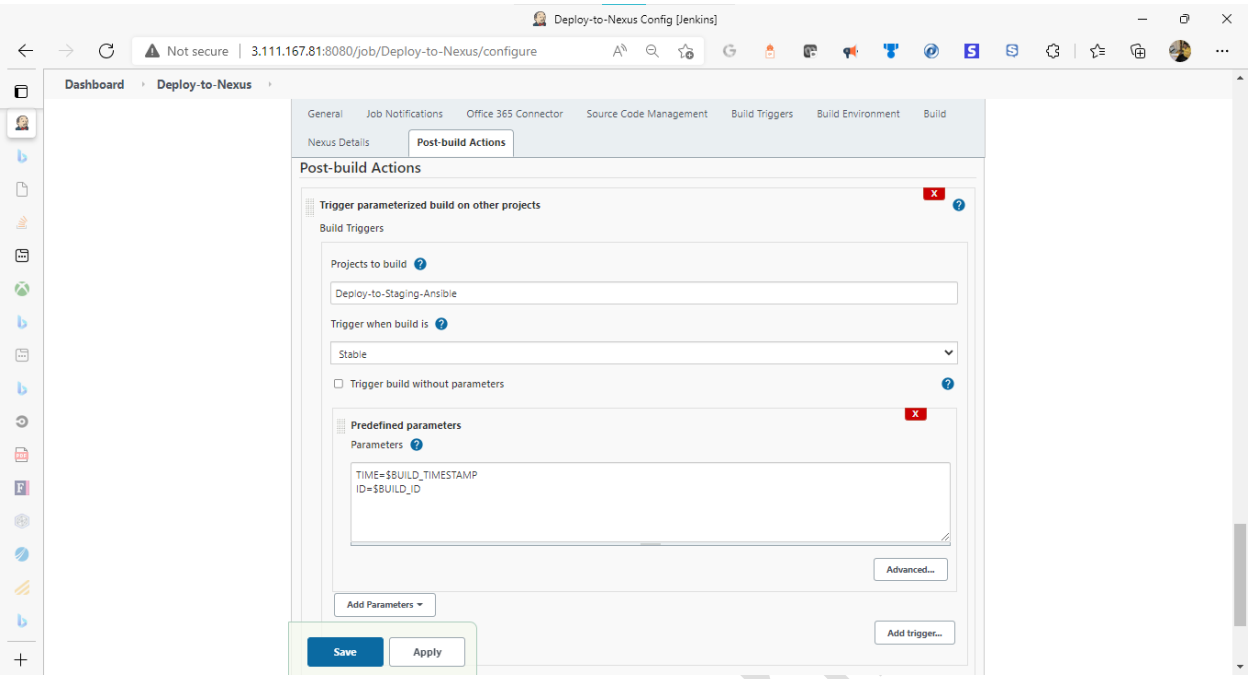




Use your nexus server ip and credentials created earlier during CI setup.

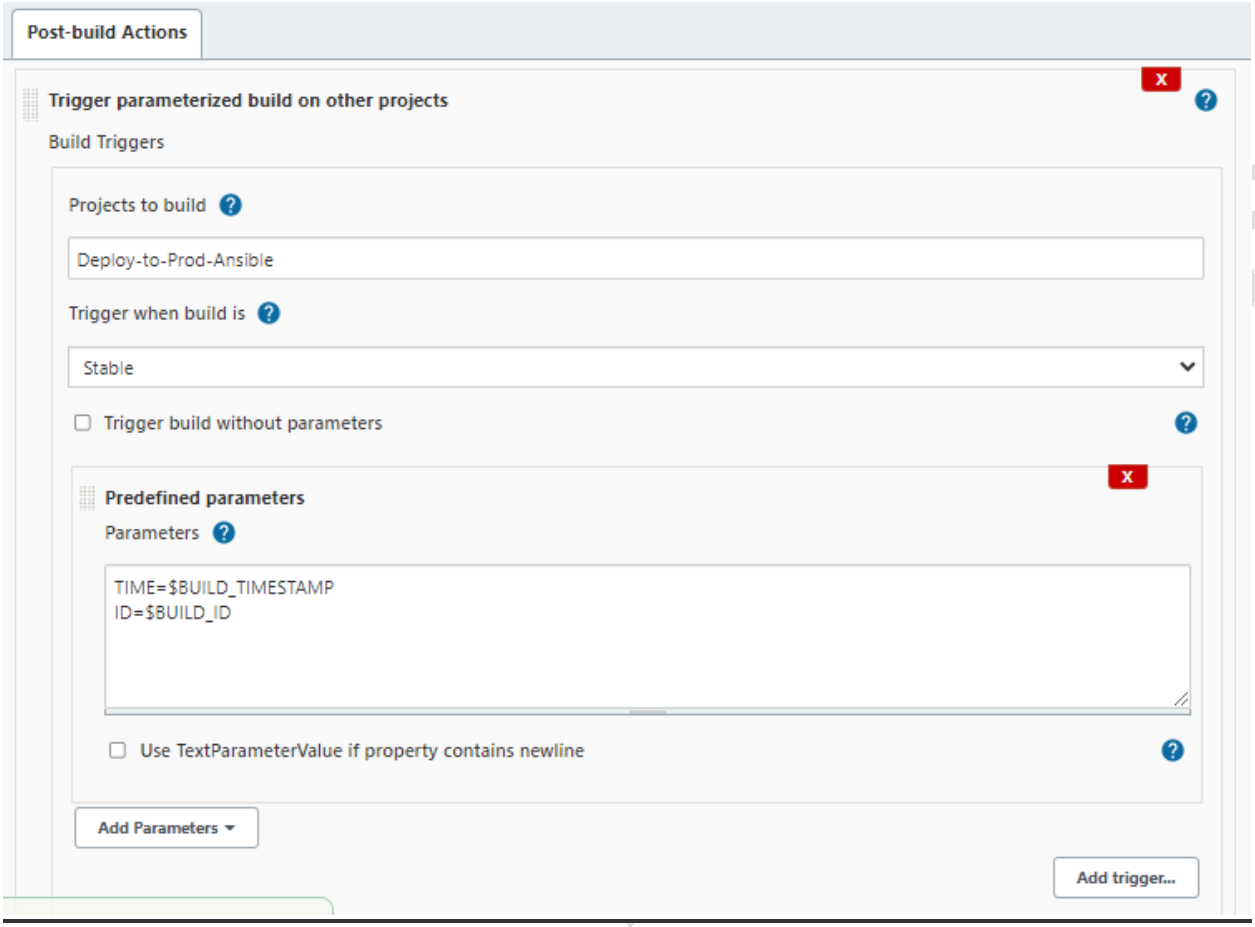

Deploy-to-Nexus Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Nexus/configure
Dashboard
Deploy-to-Nexus
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Nexus Details
Post-build Actions
Version
$BUILD_ID
Repository
logiclabs-release
Artifacts
Artifact
ArtifactId
$BUILD_TIMESTAMP
Type
war
Classifier
File
target/vprofile-v2.war
Add
Save
Apply


Deploy-to-Nexus Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Nexus/configure
Dashboard
Deploy-to-Nexus
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Nexus Details
Post-build Actions
Add
Add build step
Post-build Actions
Trigger parameterized build on other projects
Build Triggers
Projects to build
Deploy-to-Staging-Ansible
Trigger when build is
Stable
Trigger build without parameters
Predefined parameters
Parameters
TIME=$BUILD_TIMESTAMP
ID=$BUILD_ID
Save
Apply
Advanced...



Save the job with above settings and change any custom values such as names and ip addresses.

*Note: parameters are case sensitive, use exact values.



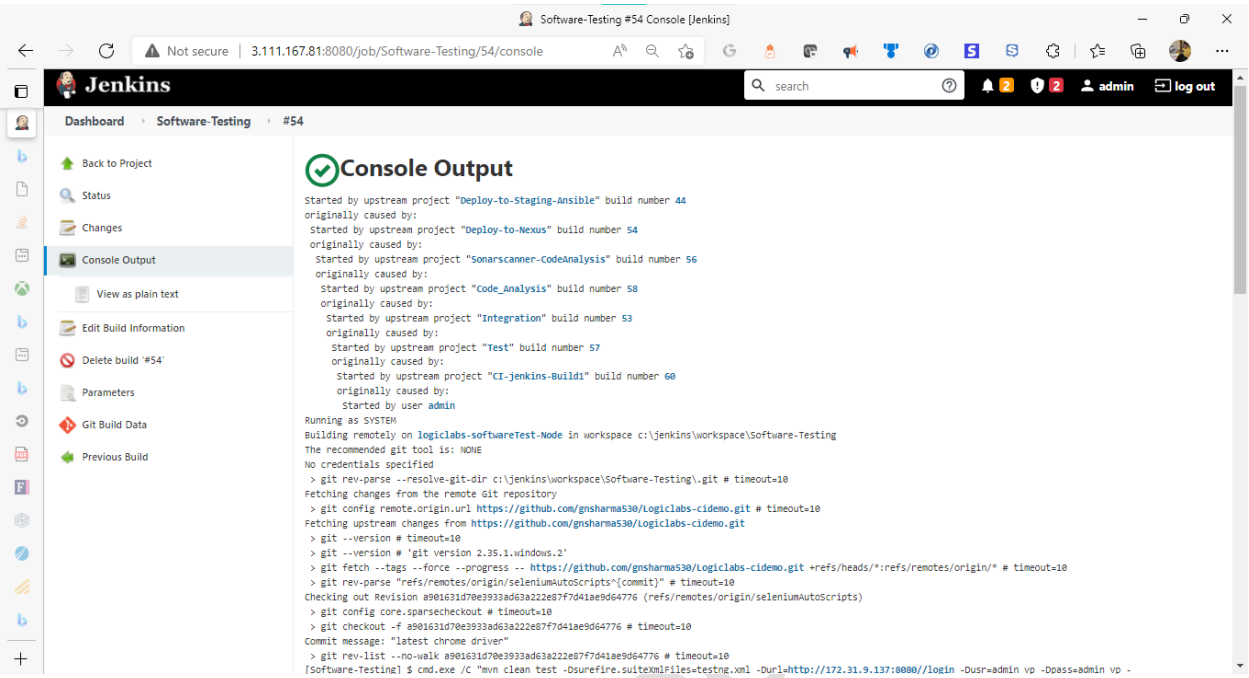
Step 8: Verifying Jenkins jobs Integration

Now that we have the jobs required for Continuous Delivery, let us integrate all into a single pipeline.

This is accomplished by using **Post-build** actions step, where we tag the subsequent job to be run after the present job.

So, till now, we have configured all the jobs with upstream and downstream jobs.

Your jenkins job pipeline console output should look like this by now,



When you start the build job, it will trigger the rest of the jobs automatically(only if the previous job succeeds).

Verify that all the jobs are properly configured for downstream jobs/upstream jobs.
Please make sure the pipeline flow is as shown below.

CI jenkins Build → UnitTest→ Integration Test→ CodeAnalysis → SonarScanner

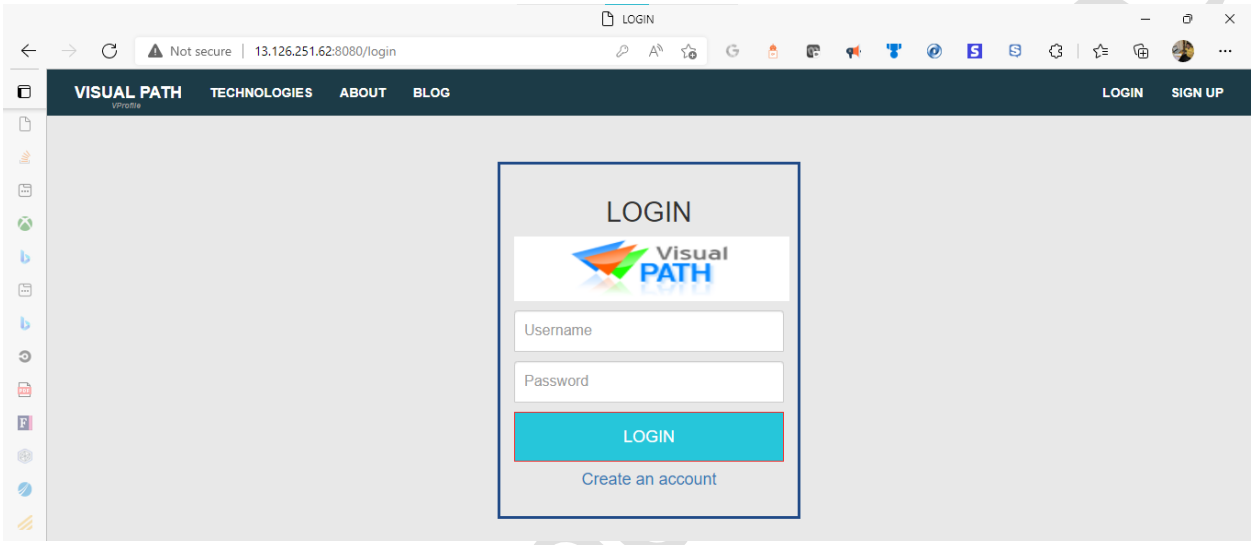
Code Analysis → Deploy to Nexus → Deploy-to-staging-Ansible → Software

Test (Selenium)

Verifying the Jobs status:

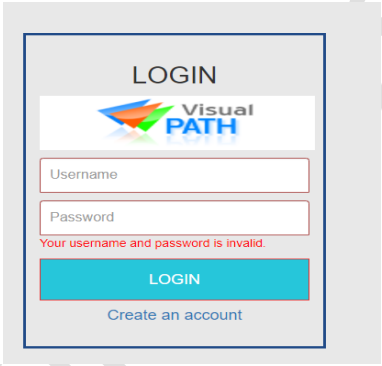
Inorder to verify all the jobs are running properly, we need to check whether our Web-server (app-01-staging) is working or not using the below steps.

- Open the ec2 console and select web-server, copy the public ip of the server.
- Try logging in with the credentials given below:
- Username: admin_vp
- Password: admin_vp
-

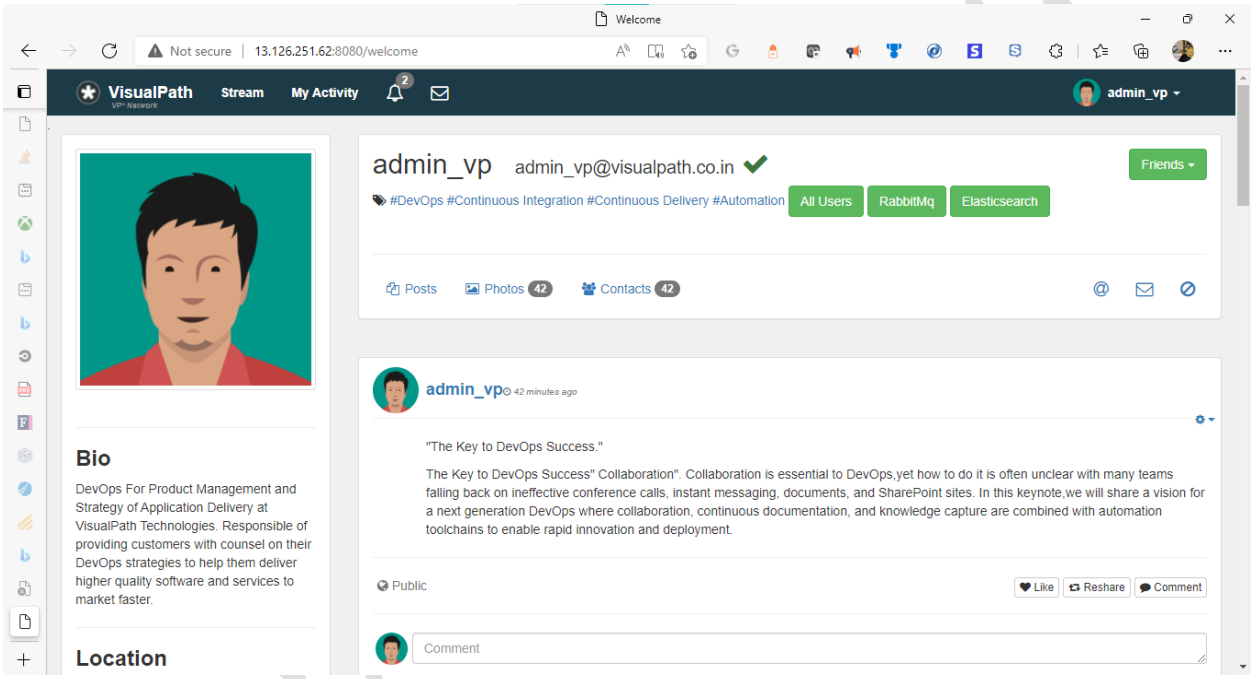


*Note if the error like “your username and password is invalid”, follow below troubleshooting steps:

-



- Check if there are any typo errors in username and password
- Check if the backend server is configured correctly with all the services like mariadb, Rabbitmq and memcached are working properly
- Enable dB port (3306) within the backend server using firewall using ufw or other command
- Similarly enable the port 3306 within the web-server port.
- Restart the firewalld service and check logging in to the web application.



Step 9: Continuous Deployment configuration

After successful working of staging environment, usually there are rigorous tests like, UAT, Sanity Tests etc., are conducted on the Staging/pre-prod environment.

Once signoff is obtained from the necessary stakeholders, the artifact is moved to production environment using the below steps.

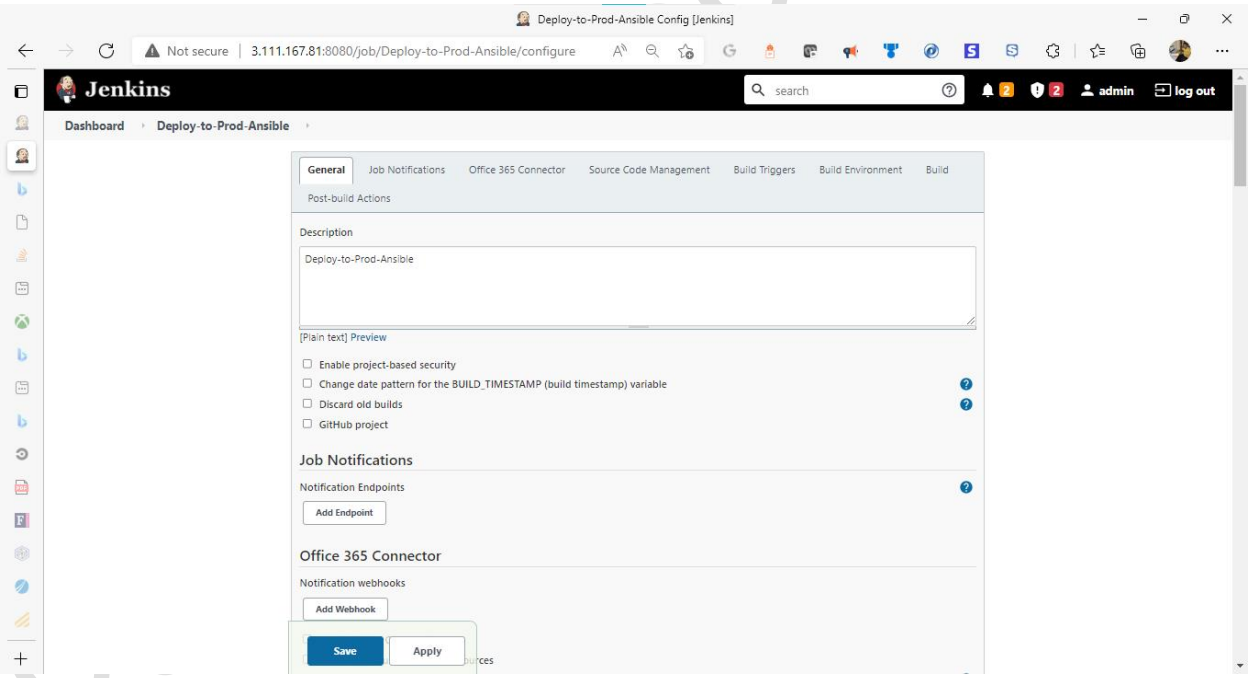
Step 9.1 Creation of Prod server

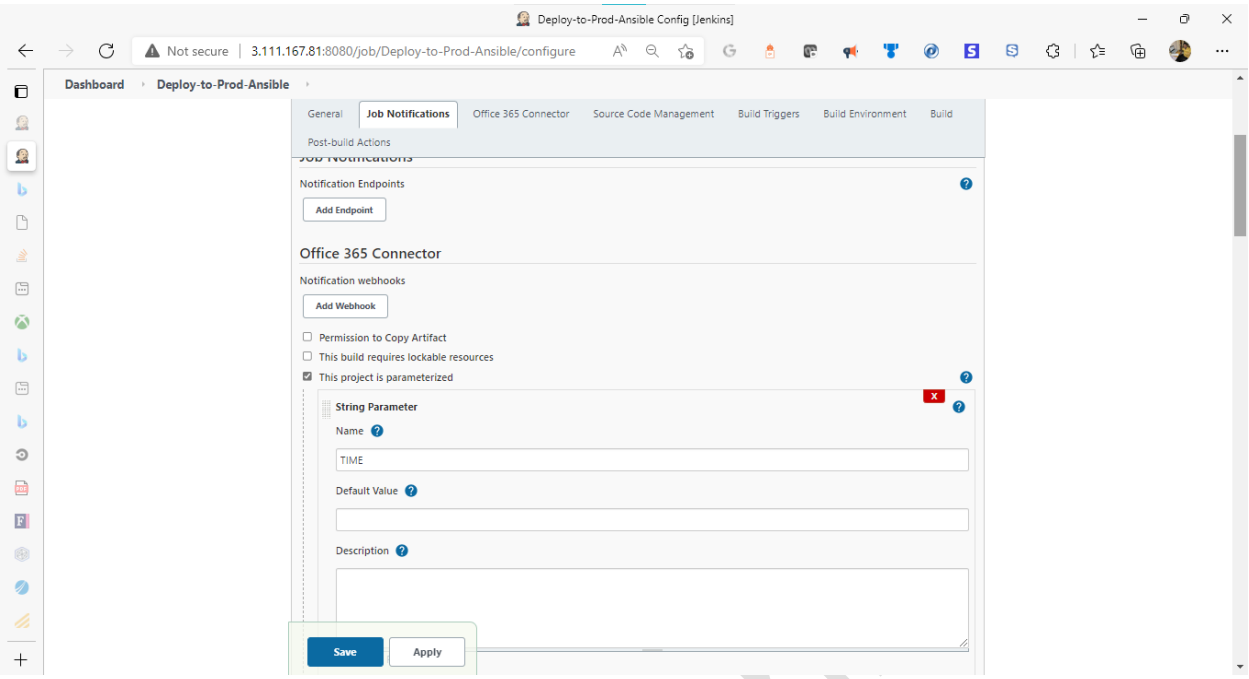
Create a server similar to staging server(app01-staging) and name it app01-prod.copy the internal ip for use in the later steps while configuring jenkins job.

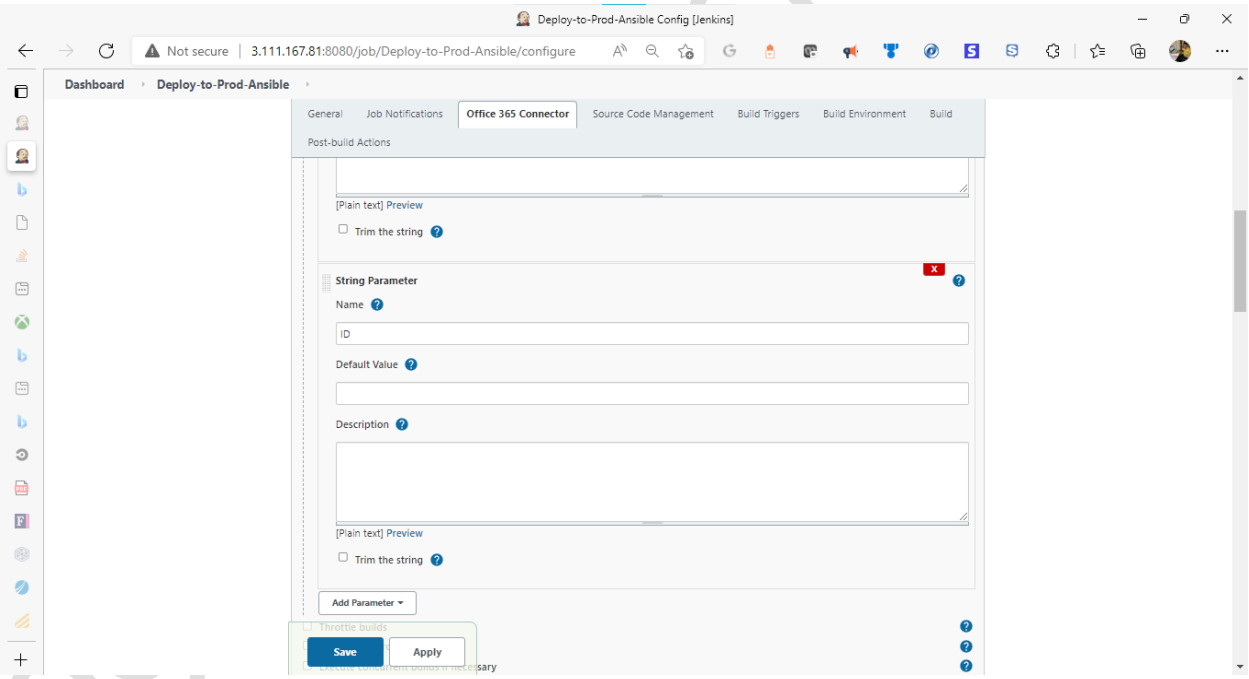
** **if you face cpu core limit issue in your AWS account, you can mail to the AWS support for increasing resource quota. However its a time taking process, so you can just use the same staging server also in the Deploy-to-prod job for this Demonstration purpose.**

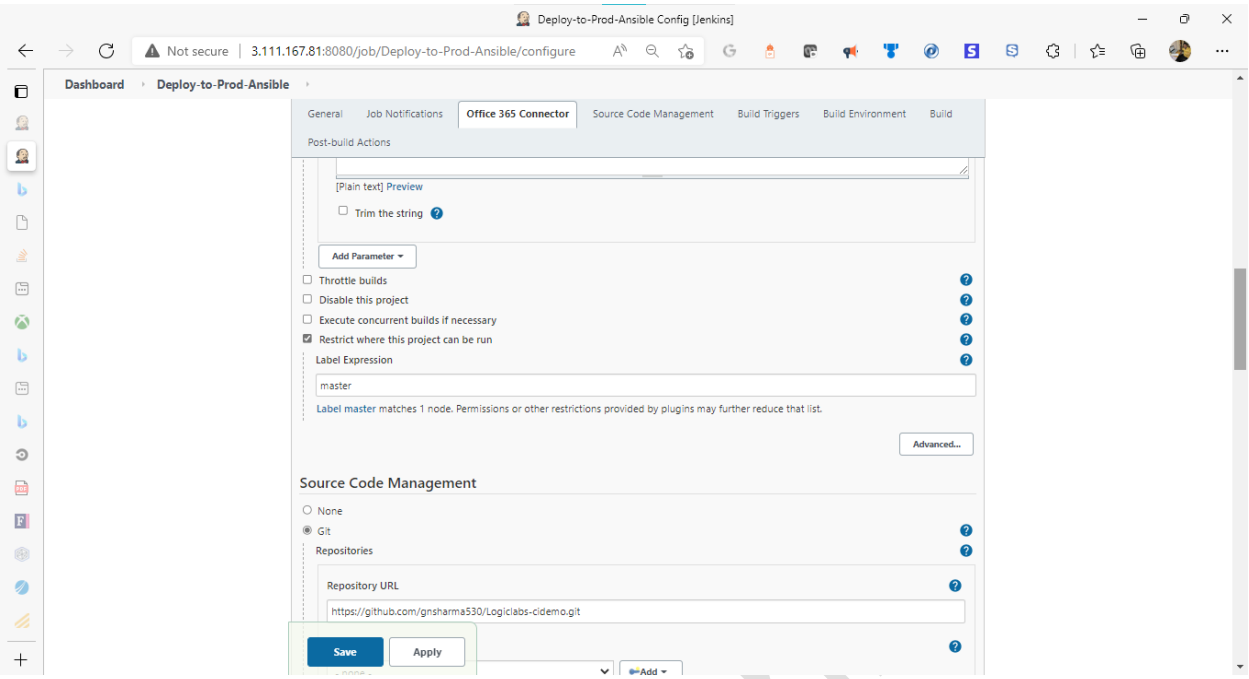
Step 9.2 Configuring Deploy-to-prod-Ansible job

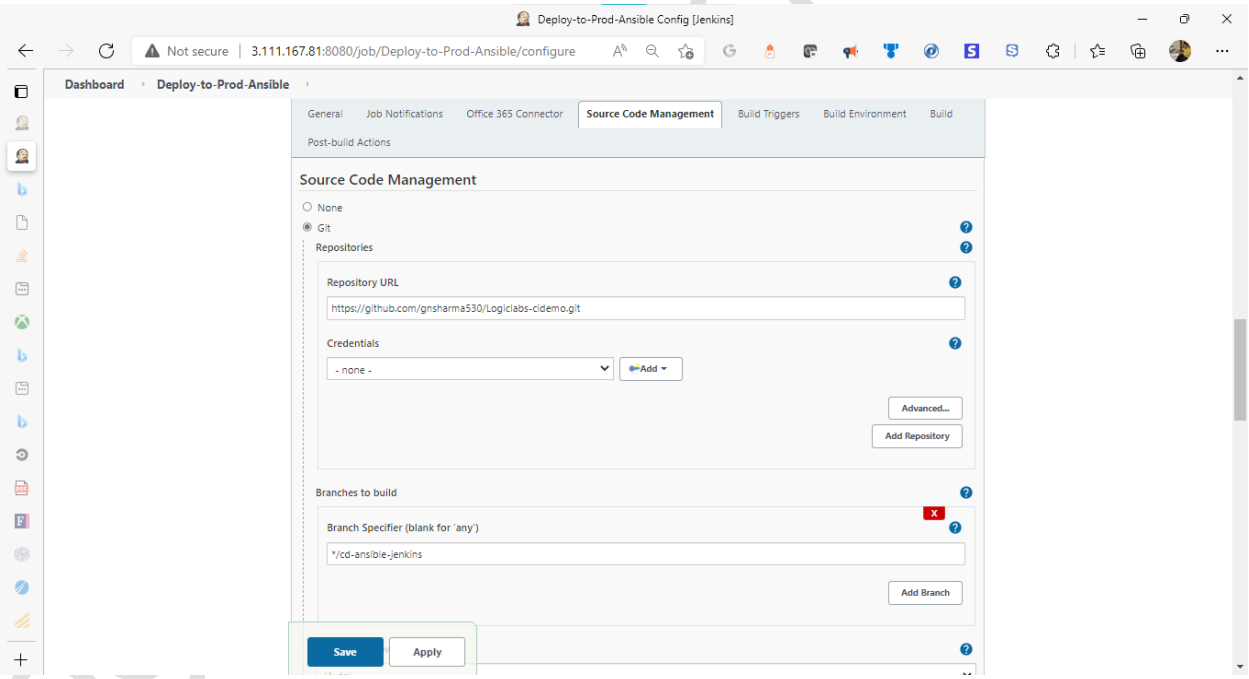
- After properly configuring the software-test (selenium) job i.e., step 6.4, follow the below steps
- Create a new jenkins free style job and copy from Deploy-to-staging job, with the following configuration:

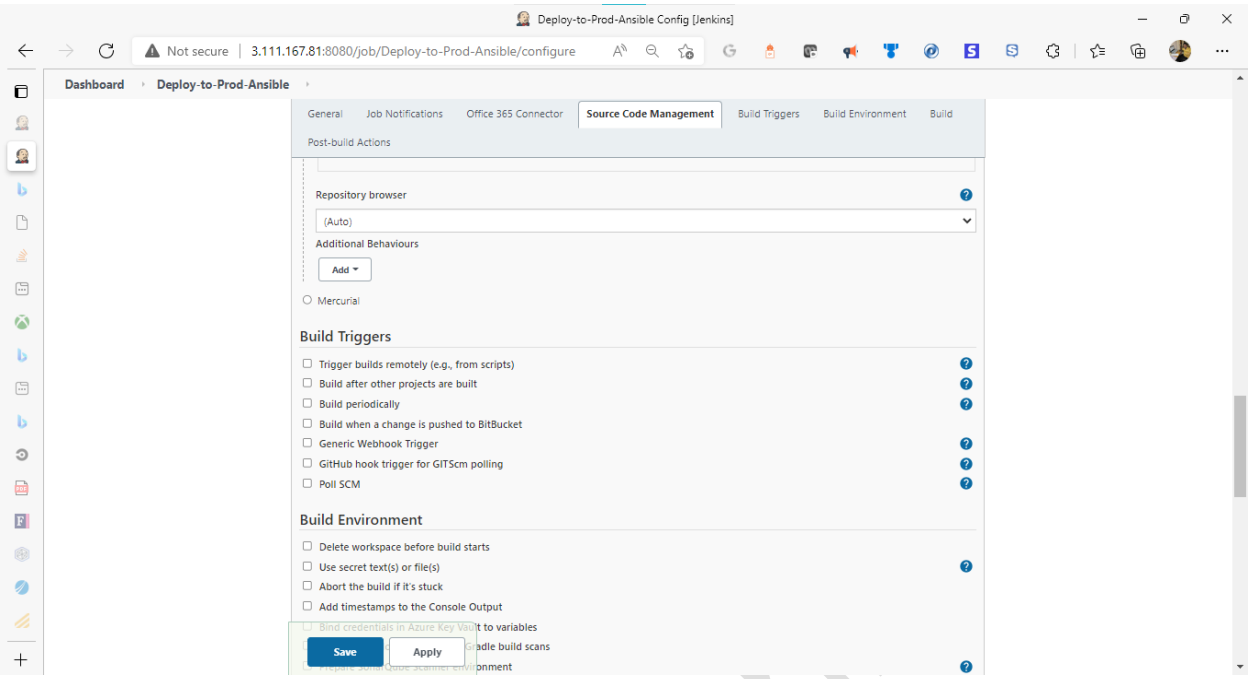


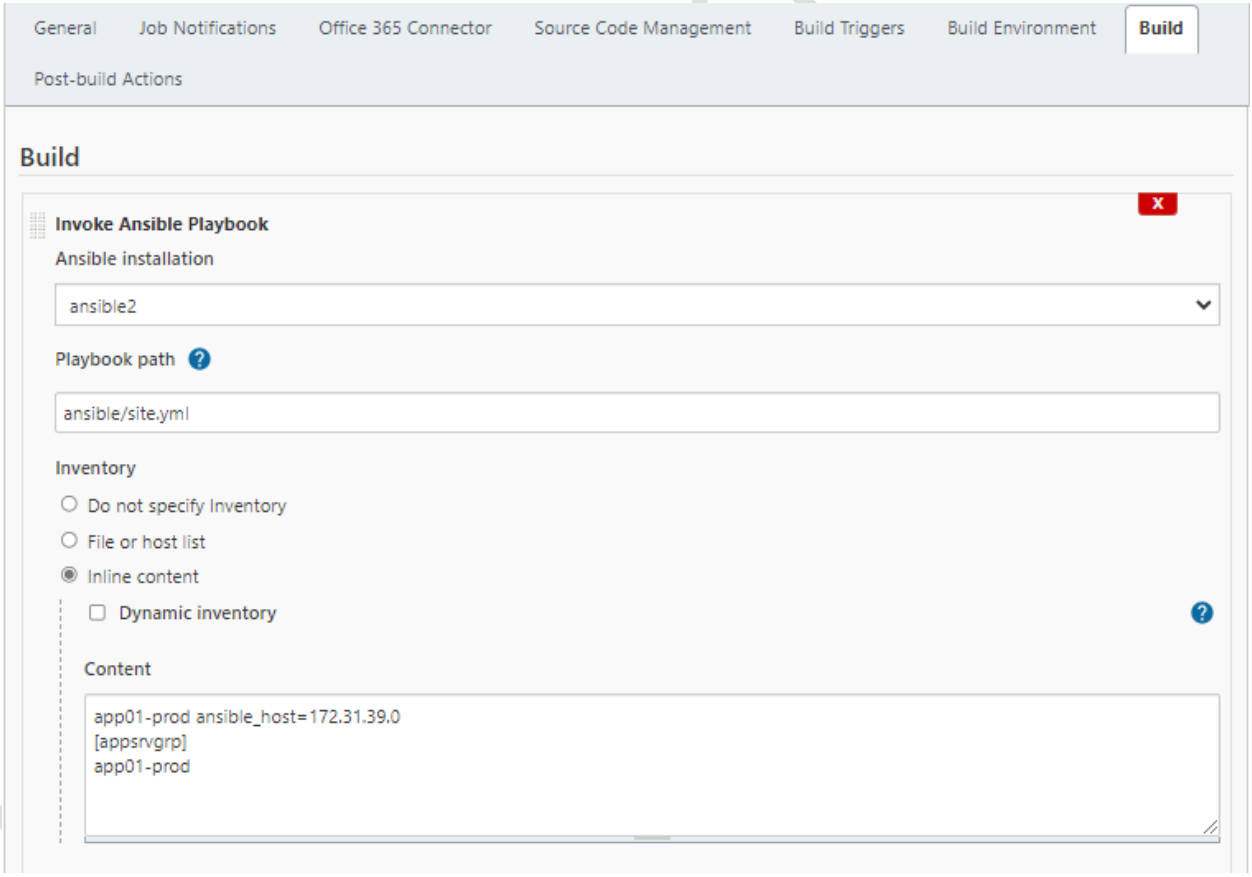

Deploy-to-Prod-Ansible Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Prod-Ansible/configure
Dashboard
Deploy-to-Prod-Ansible
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Notification Endpoints
Add Endpoint
Office 365 Connector
Notification webhooks
Add Webhook
Permission to Copy Artifact
This build requires lockable resources
This project is parameterized
String Parameter
Name
TIME
Default Value
Description
Save
Apply


Deploy-to-Prod-Ansible Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Prod-Ansible/configure
Dashboard
Deploy-to-Prod-Ansible
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
[Plain text] Preview
Trim the string
String Parameter
Name
ID
Default Value
Description
[Plain text] Preview
Trim the string
Add Parameter
Save
Apply

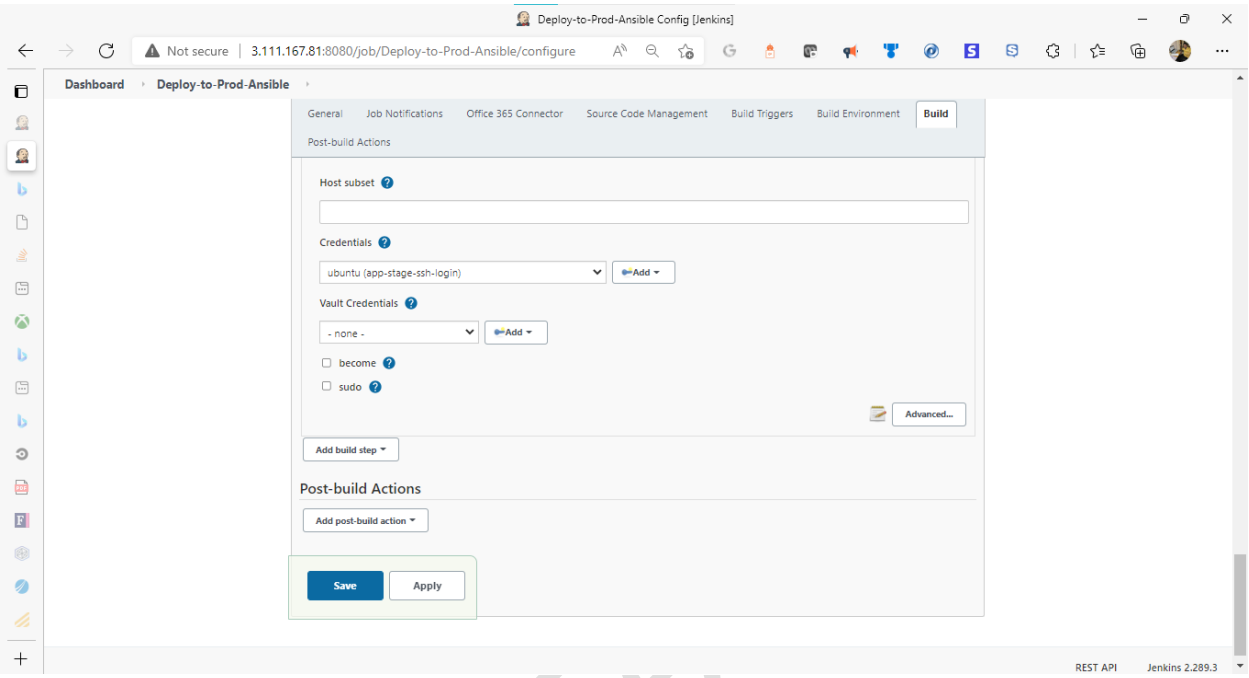

Deploy-to-Prod-Ansible Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Prod-Ansible/configure
Dashboard
Deploy-to-Prod-Ansible
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
[Plain text] Preview
Trim the string
Add Parameter
Throttle builds
Disable this project
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
master
Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.
Advanced...
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/gnsharma530/Logiclabs-cidemo.git
Save
Apply


Deploy-to-Prod-Ansible Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Prod-Ansible/configure
Dashboard
Deploy-to-Prod-Ansible
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/gnsharma530/Logiclabs-cidemo.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/cd-ansible-jenkins
Add Branch
Save
Apply

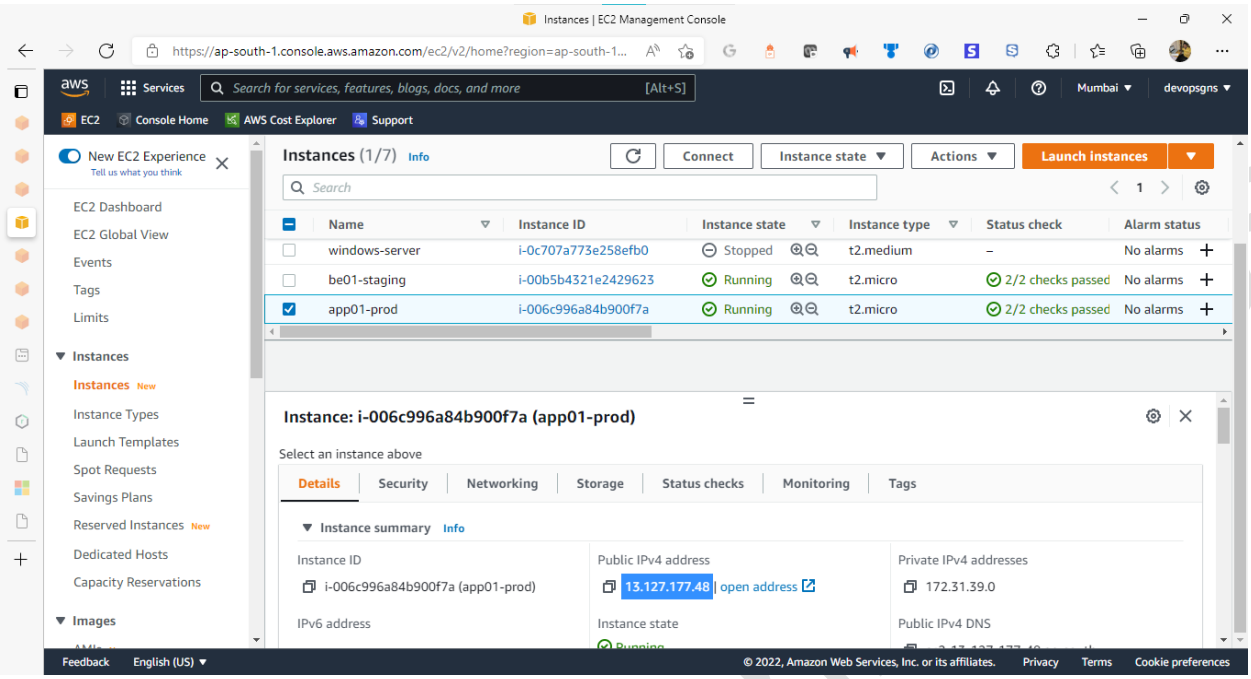

Deploy-to-Prod-Ansible Config [Jenkins]
Not secure | 3.111.167.81:8080/job/Deploy-to-Prod-Ansible/configure
Dashboard
Deploy-to-Prod-Ansible
General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Repository browser
(Auto)
Additional Behaviours
Add
Mercurial
Build Triggers
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
Build when a change is pushed to BitBucket
Generic Webhook Trigger
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
Save
Apply

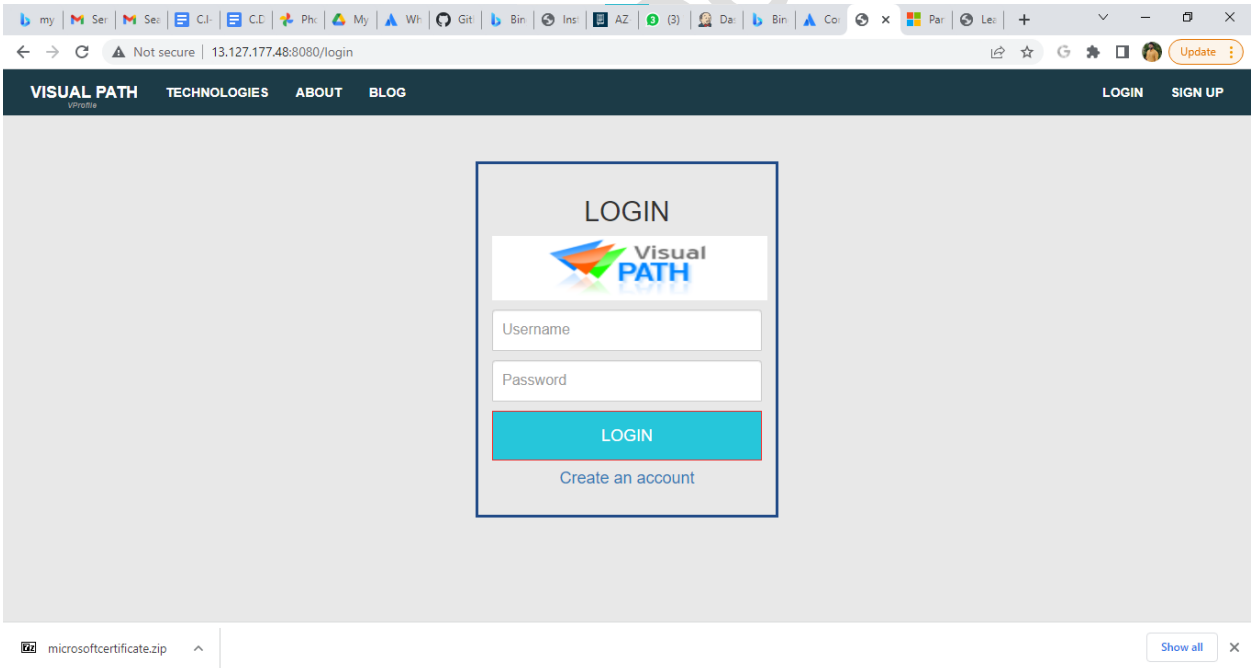

General
Job Notifications
Office 365 Connector
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Build
Invoke Ansible Playbook
Ansible installation
ansible2
Playbook path
ansible/site.yml
Inventory
Do not specify Inventory
File or host list
Inline content
Dynamic inventory
Content
app01-prod ansible_host=172.31.39.0
[appsrvgrp]
app01-prod

Observe the changes made to the Build step in the inventory content. Replace the name of the server and ip of your production server.



Complete the configuring of Deployment-to-Prod-Ansible job. Test the deployment into the production server by logging into the web-application using the prod server public ip.


Instances | EC2 Management Console
https://ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1...
Search for services, features, blogs, docs, and more [Alt+S]
Mumbai
devopsgns
EC2 Console Home AWS Cost Explorer Support
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
Instances (1/7) Info
Connect
Instance state
Actions
Launch instances
Search
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
windows-server i-0c707a773e258efb0 Stopped t2.medium – No alarms
be01-staging i-00b5b4321e2429623 Running t2.micro 2/2 checks passed No alarms
app01-prod i-006c996a84b900f7a Running t2.micro 2/2 checks passed No alarms
Instance: i-006c996a84b900f7a (app01-prod)
Select an instance above
Details
Security
Networking
Storage
Status checks
Monitoring
Tags
Instance summary Info
Instance ID
i-006c996a84b900f7a (app01-prod)
Public IPv4 address
13.127.177.48 | open address
Private IPv4 addresses
172.31.39.0
IPv6 address
Instance state
Public IPv4 DNS
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Not secure 13.127.177.48:8080/login
Update
VISUAL PATH
TECHNOLOGIES
ABOUT
BLOG
LOGIN
SIGN UP
LOGIN
Visual PATH
Username
Password
LOGIN
Create an account
microsoftcertificate.zip
Show all

Step 10: Continuous Delivery Vs Continuous Deployment

Continuous delivery

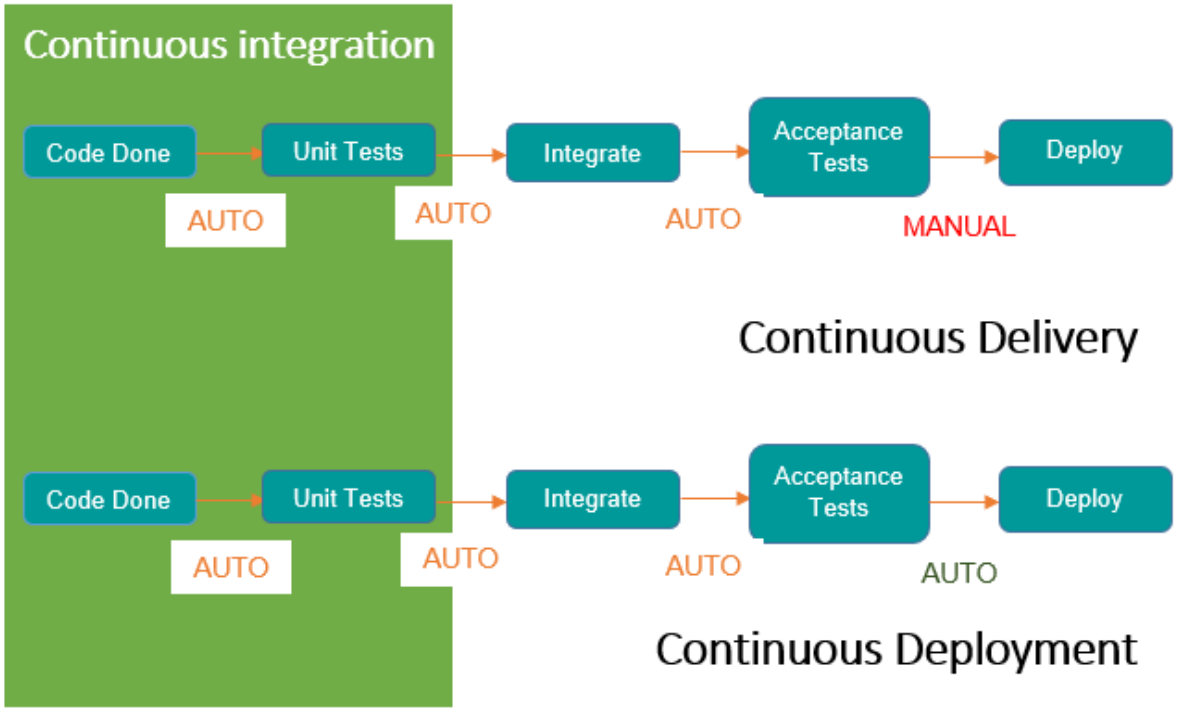
Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.

This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

Continuous deployment

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.



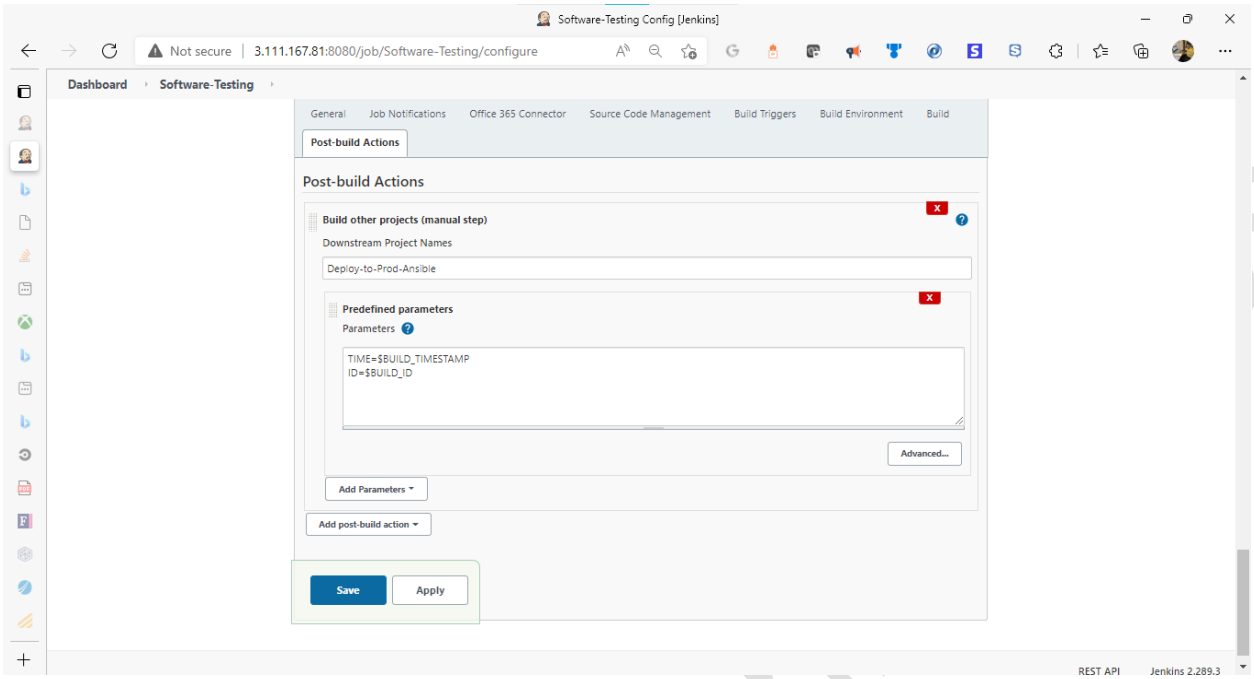
The difference between Continuous Delivery and Deployment is the automated deployment of artifact into the production server, that makes application go live.

We have demonstrated how to perform continuous deployment in the above steps.
In this step we will look into enabling manual approval for deployment into production so that we demonstrate continuous Delivery.

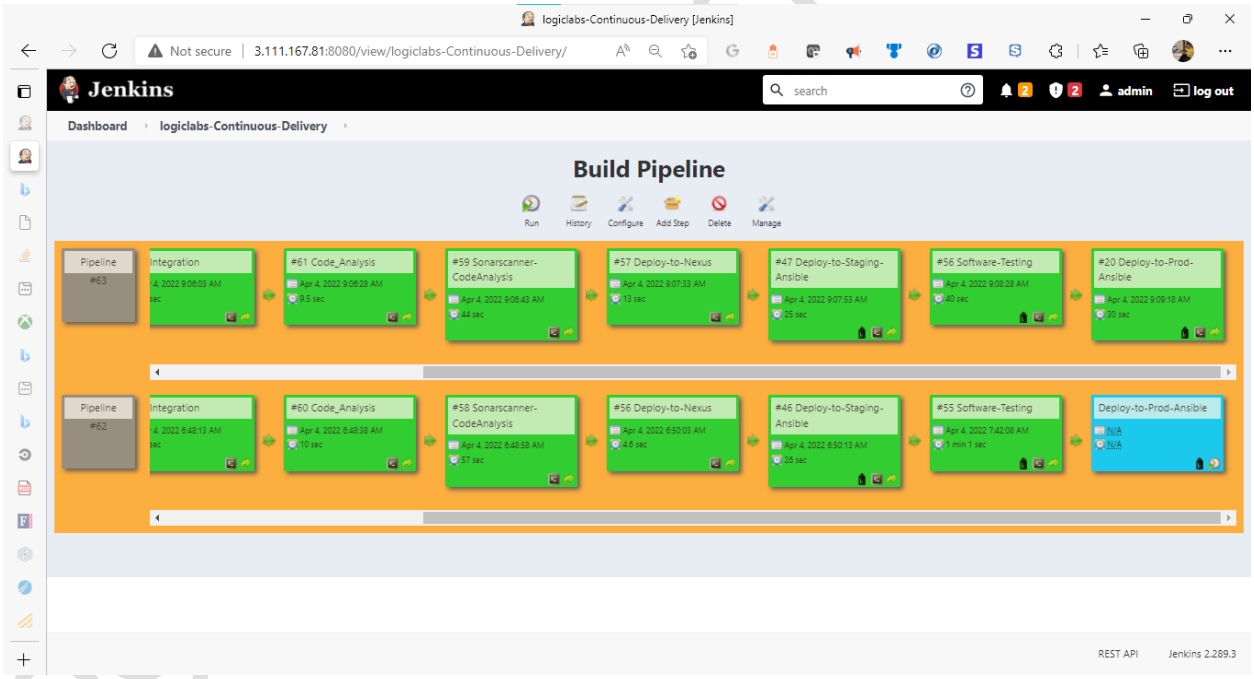
Go to Software testing job and make the changes to enable manual triggering of Deploy-to-prod job.

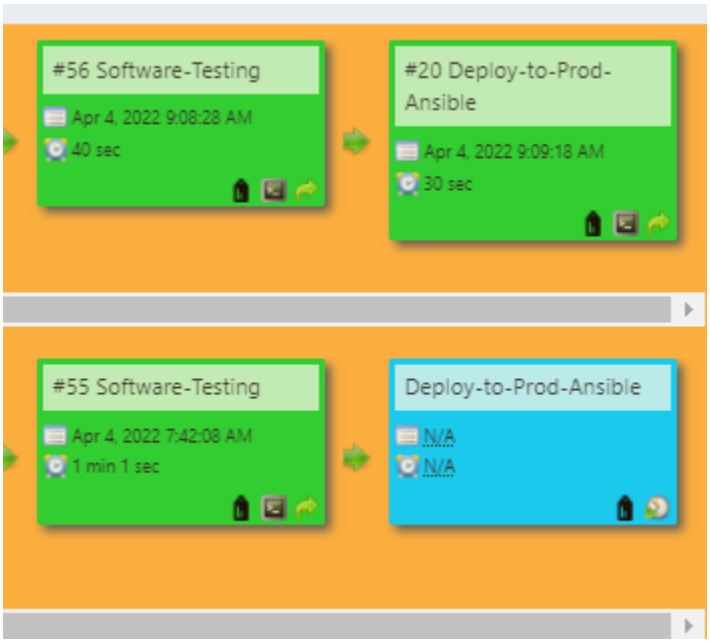
Go to Software-Testing job → Post build Actions → select Build other projects

(Manual step) and give the parameters as given below.



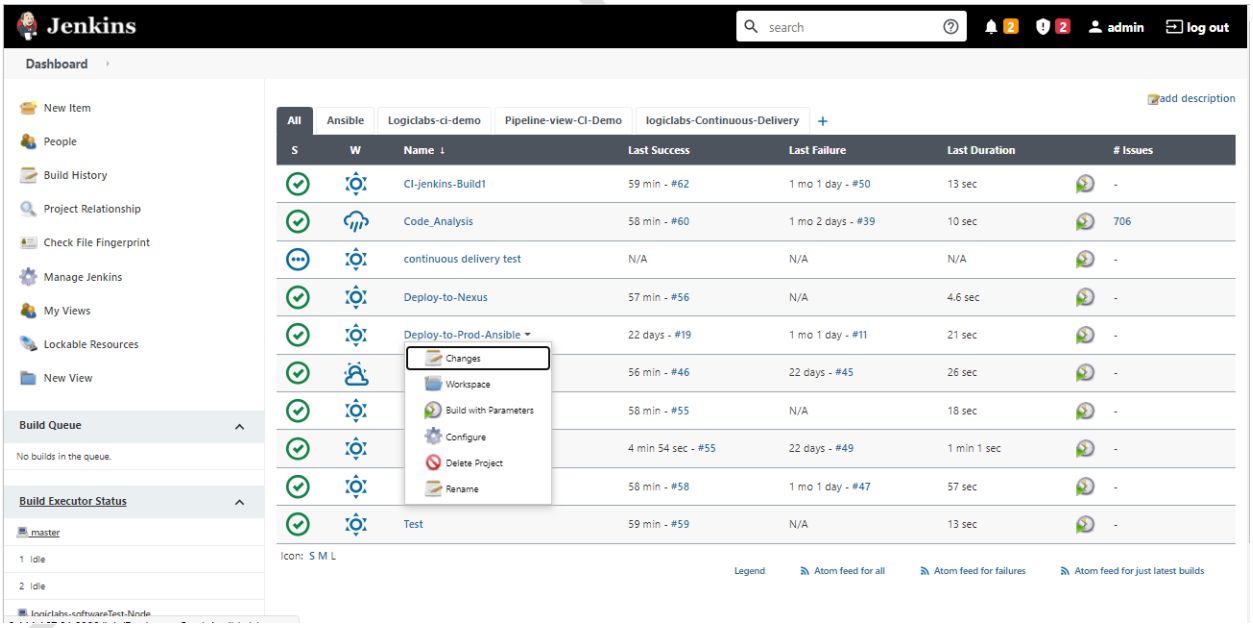
* Change your prod server name and IP address.





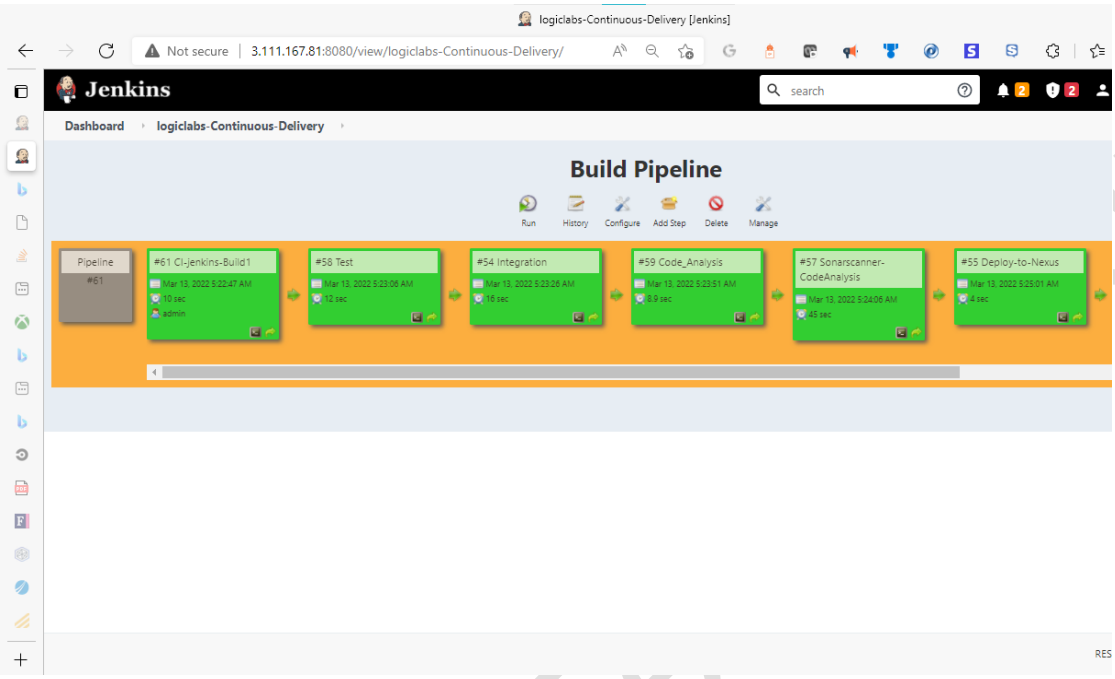
Observe that deploy to prod (below pipeline) has not automatically triggered and it has to be manually triggered,because continuous Delivery has been performed.

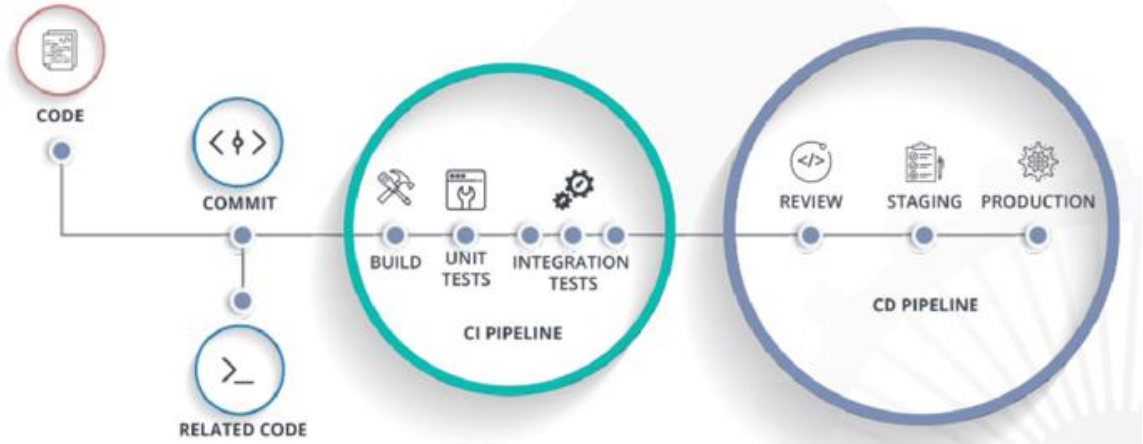
Manual procedure:



Conclusion:

After following the above steps you will be ready with a complete C.I/C.D setup .





With this demonstration, you have learnt how to build an end to end C.I and C.D setup using jenkins and Ansible. We have witnessed how a C.I. pipeline can be built using tools such as jenkins, sonarQube and Nexus. We have observed how the source code commit by developer triggers a pipeline which starts with build job followed by Unit test, Integration test, Code analysis using

SonarQube scanner and a well tested artifact is uploaded into Nexus repository.

We have seen how to setup and configure a web-server using Ansible. After that we have seen backend server setup using user-data scripts. We have performed automated testing using Selenium by setting up a Windows server. Finally we have observed how to configure a pipeline for Continuous Delivery or Deployment using jenkins.

---***---