

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

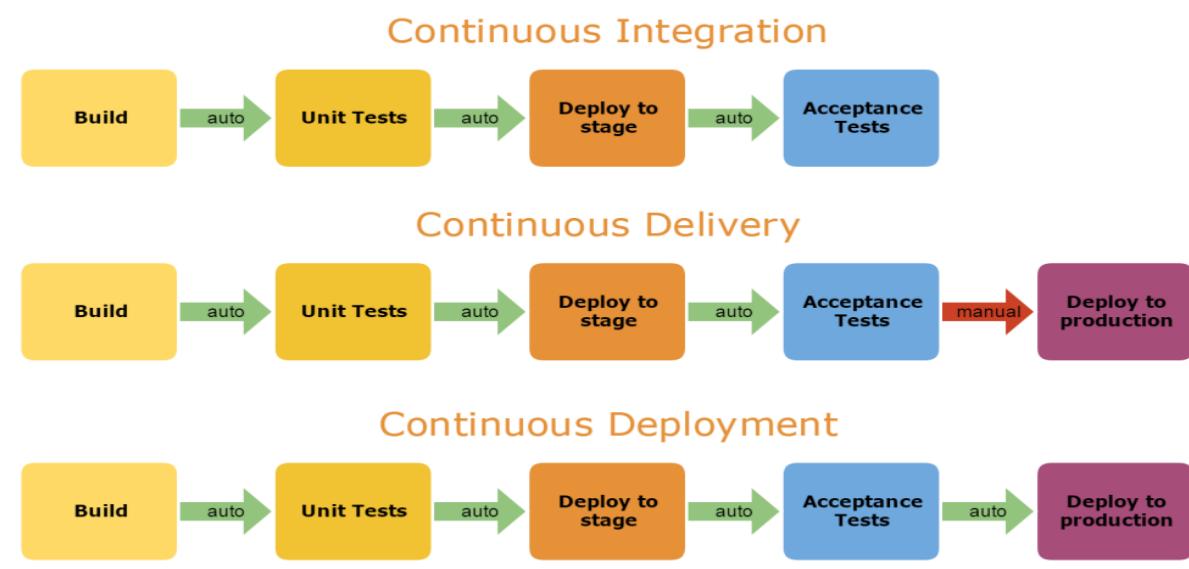
What is Continuous Deployment (C.D)?

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.

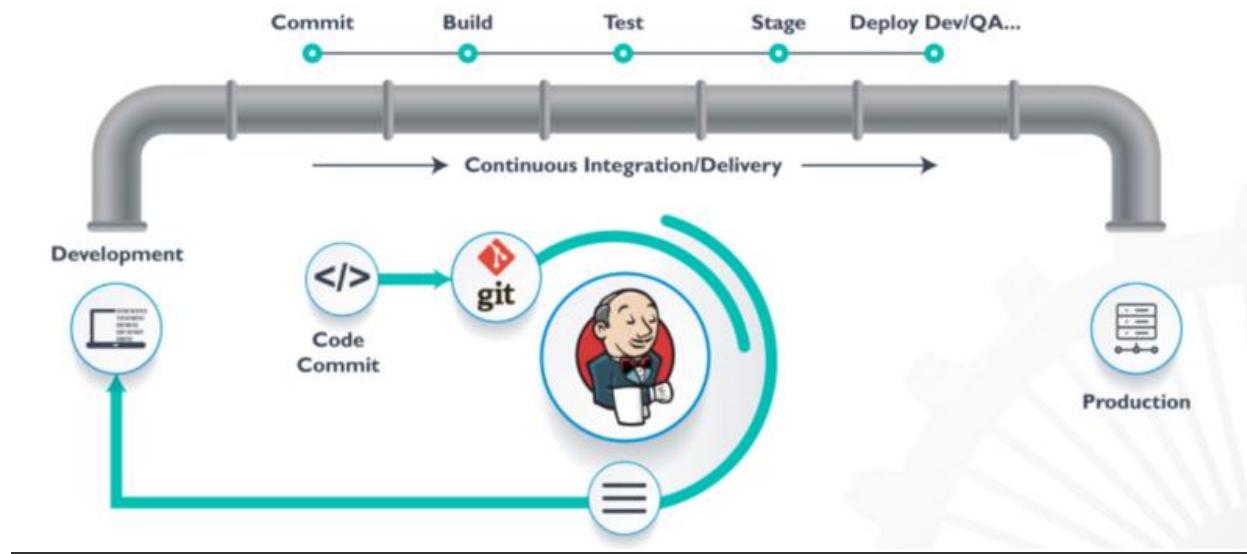
A software engineering approach, CD delivers software functionalities frequently through automated deployments.

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:

<u>Jenkins</u>	<u>Git</u>	<u>Maven</u>	<u>SonarQube</u>	<u>Selenium</u>
<u>Checkstyle</u>	<u>AWS EC2</u>	<u>Nexus3</u>	<u>Ansible</u>	<u>Apache Tomcat</u>

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)

The screenshot shows the Jenkins dashboard with the following details:

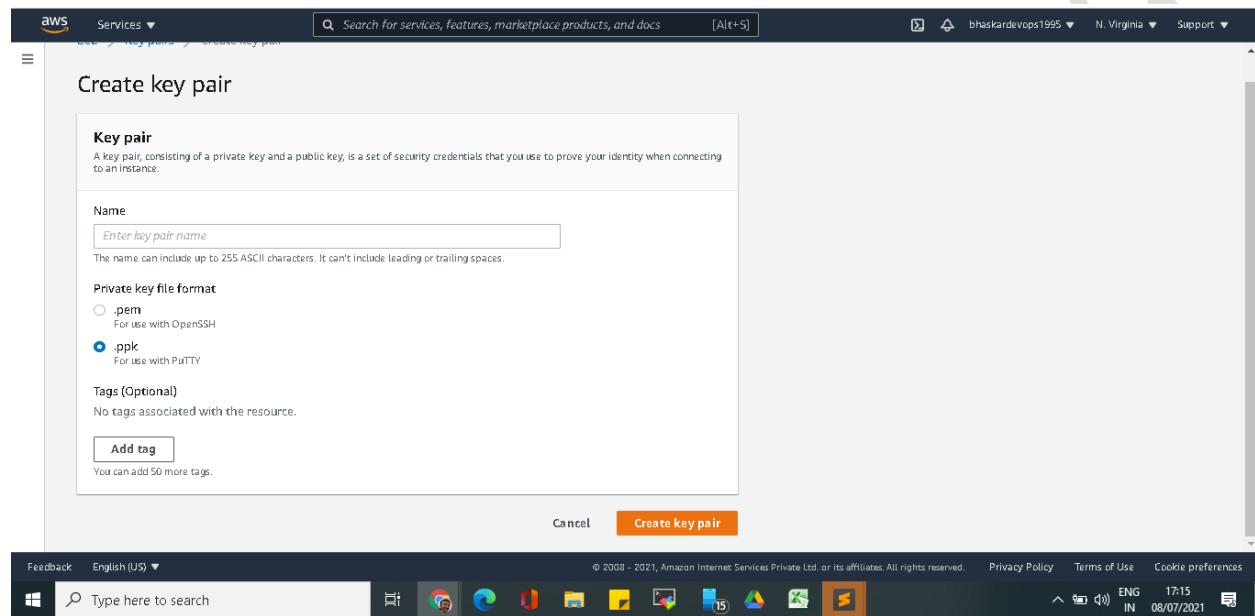
- Dashboard [Jenkins]**: The title bar indicates the browser is not secure.
- Toolbar**: Includes links for Google Docs, Reports - Billing, Devops, IAS, Python, Cloud, Courses List - Digitalocean, Webmail Login, Edit PDF Foxit Phan, The Hindu, and Other favorites.
- User Information**: Shows admin logged in.
- Left Sidebar**: Includes links for New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. A "Build Queue" section states "No builds in the queue".
- Central Content Area**: A table displays six Jenkins jobs:

S	W	Name	Last Success	Last Failure	Last Duration	# Issues
✓	⌚	CI-jenkins-Build1	8 hr 10 min - #6	N/A	8.3 sec	-
✓	⚡	Code_Analysis	1 hr 21 min - #13	8 hr 39 min - #7	8.5 sec	96
✓	⌚	Deploy-to-Nexus	8 min 1 sec - #3	N/A	0.91 sec	-
✓	⌚	Integration	1 hr 22 min - #8	N/A	13 sec	-
✓	☁️	Sonarscanner-CodeAnalysis	1 hr 9 min - #4	1 hr 12 min - #3	2 min 0 sec	-
✓	⌚	Test	1 hr 22 min - #12	N/A	11 sec	-
- Bottom Navigation**: Includes a legend for icons, links for Atom feed for all, Atom feed for failures, Atom feed for just latest builds, and a "5 M L" link.

Stage -1

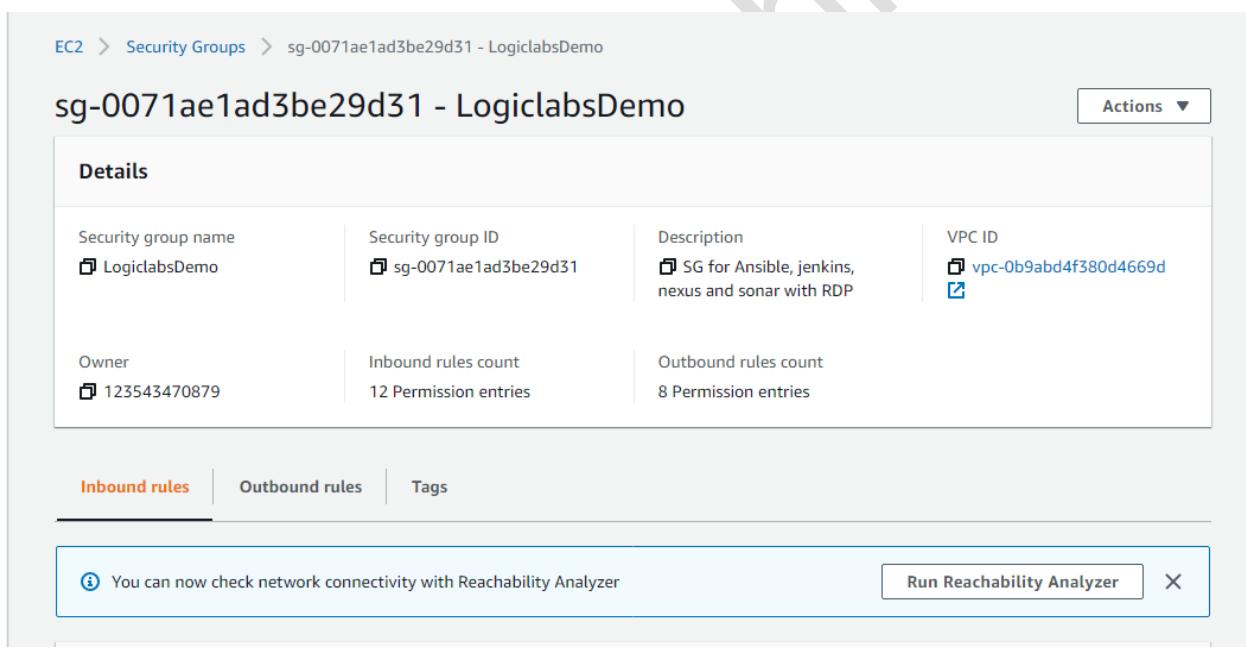
Key pair setup (if not already created)

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)



A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

6

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-0cdd0a133279ffe3a	All traffic	All	All	Custom ▼	<input type="text" value="49.204.177.219/32"/> personal machine Delete
sgr-00a3e8fdcfcd23c39d	MySQL/Aurora	TCP	3306	Custom ▼	<input type="text" value="sg-0071ae1ad3be29d31"/> for DB (optional) Delete
sgr-0321f695f3a1f5079	All traffic	All	All	Custom ▼	<input type="text" value="124.123.188.241/32"/> jenkins server Delete
sgr-0b92243232c535685	All traffic	All	All	Custom ▼	<input type="text" value="sg-0071ae1ad3be29d31"/> This security group -allow i Delete
sgr-07746d4ab330be6f5	All traffic	All	All	Custom ▼	<input type="text" value="13.233.63.130/32"/> Selenium server Delete

[Add rule](#)

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

Instances (1/7) Info						C	Connect	Instance state ▼	Actions ▲	Launch instances ▼
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status ch					
<input checked="" type="checkbox"/>	Jenkins-server	i-0b0172e37eb604e73	Stopped Start Stop	t2.small	-					
<input type="checkbox"/>	sonarqube-server	i-0f0ae2f0808b79701	Stopped Start Stop	t2.medium	-					
<input type="checkbox"/>	Nexus-server	i-06c990075e179b649	Stopped Start Stop	t2.medium	-					
<input type="checkbox"/>	app01-staging	i-0801d0c8ec2054a13	Stopped Start Stop	t2.medium	-					
<input type="checkbox"/>	windows-server	i-0c707a773e258efb0	Stopped Start Stop	t2.micro	-					
<input type="checkbox"/>	be01-staging	i-00b5b4321e2429623	Stopped Start Stop	t2.micro	-					

- Connect
- View details
- Manage instance state
- Instance settings
- Networking
- Security [▼](#)
- Get Windows password
- Image and templates
- Modify IAM role
- Monitor and troubleshoot

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

The screenshot shows the AWS EC2 'Change security groups' interface. At the top, there's a navigation bar with links like 'Services', 'Console Home', 'AWS Cost Explorer', and 'Support'. Below that, the breadcrumb navigation shows 'EC2 > Instances > i-0b0172e57eb604e73 > Change security groups'. The main content area has two sections: 'Instance details' (showing Instance ID: i-0b0172e57eb604e73 (Jenkins-server) and Network interface ID: eni-09418580e17da0472) and 'Associated security groups' (listing Security group name: LogilabDemo and Security group ID: sg-0071ae1ad3be29d31). There are buttons for 'Add security group' and 'Remove'. At the bottom right are 'Cancel' and 'Save' buttons.

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/Logilabs-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.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

9

The screenshot shows a GitHub repository page for 'gnsharma530 / Logilabs-cidemo'. The repository is public and has 40 branches and 0 tags. The commit history for the 'cd-ansible-jenkins' branch is displayed, showing 121 commits from 27 days ago. The commits include changes to backend.sh, ansible, src, and userdata files, along with Jenkinsfile and README.md. The repository has 65 forks and 0 stars. It also includes sections for About, Releases, and Packages.

File	Description	Time
backend.sh	modified	8953ee6 27 days ago
ansible	Commented application.properties file deployment section	2 years ago
src	changed rmq details in application.properties file	2 years ago
userdata	backend.sh modified	27 days ago
Jenkinsfile	Testing new Jenkinsfile	2 years ago
README.md	testing git poll from jenkins	2 years ago
pom.xml	pom and settings.xml	2 years ago
settings.xml	pom and settings.xml	2 years ago

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:

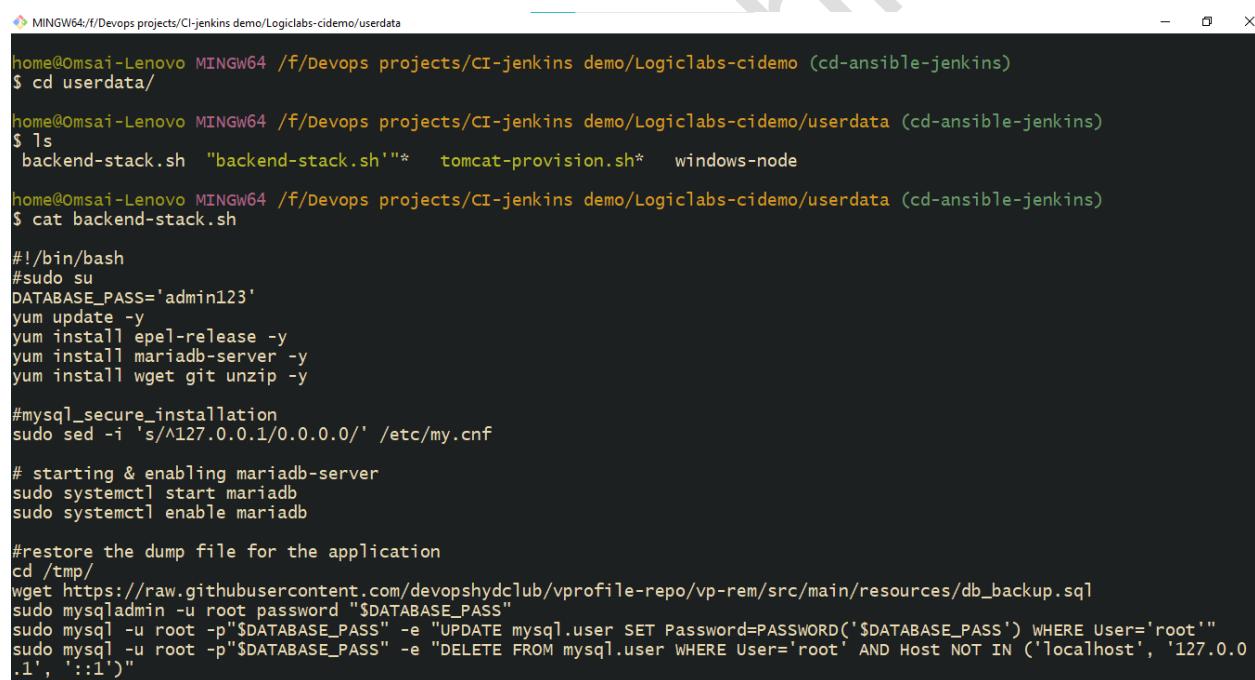
Instance Type: t2.micro

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.



```
MINGW64:/f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata
$ cd userdata/
MINGW64:/f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (cd-ansible-jenkins)
$ ls
backend-stack.sh  "backend-stack.sh"/*  tomcat-provision.sh*  windows-node
MINGW64:/f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata (cd-ansible-jenkins)
$ cat backend-stack.sh
#!/bin/bash
#sudo su
DATABASE_PASS='admin123'
yum update -y
yum install epel-release -y
yum install mariadb-server -y
yum install wget git unzip -y

#mysql_secure_installation
sudo sed -i 's/^127.0.0.1/0.0.0.0/' /etc/my.cnf

# starting & enabling mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb

#restore the dump file for the application
cd /tmp/
wget https://raw.githubusercontent.com/devopshydrus/vprofile-repo/vp-rem/src/main/resources/db_backup.sql
sudo mysqladmin -u root password "$DATABASE_PASS"
sudo mysql -u root -p"$DATABASE_PASS" -e "UPDATE mysql.user SET Password=PASSWORD('$DATABASE_PASS') WHERE User='root'"
sudo mysql -u root -p"$DATABASE_PASS" -e "DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1', '::1')"
```

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

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

11

Step 3: Configure Instance Details

Credit specification Unlimited Additional charges may apply

File systems Add file system Create new file system

Advanced Details

Enclave Enable

Metadata accessible Enabled

Metadata version V1 and V2 (token optional)

Metadata token response hop limit 1

User data As text As file Input is already base64 encoded

```
#!/bin/bash
sudo apt update
sudo apt install openjdk-8-jdk -y
sudo apt install maven git wget unzip -y
wget -q -O /tmp/jenkins-key https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

Cancel Previous Review and Launch Next: Add Storage

- 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
```

```
[centos@ip-172-31-44-64 ~]
[centos@ip-172-31-44-64 ~]$ sudo systemctl status mariadb
● mariadb.service - MariaDB database server
  Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
  Active: active (running) since Thu 2022-03-24 11:44:00 UTC; 6min ago
    Process: 1038 ExecStartPost=/usr/libexec/mariadb-wait-ready $MAINPID (code=exited, status=0/SUCCESS)
   Main PID: 1037 (mysqld_safe)
  CGroup: /system.slice/mariadb.service
         └─1037 /bin/sh /usr/bin/mysqld_safe --basedir=/usr
           ├─1337 /usr/libexec/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --log-error=/var/log/mariadb/mariadb.log --pid-fil...
Mar 24 11:43:56 ip-172-31-44-64.ap-south-1.compute.internal systemd[1]: Starting MariaDB database server...
Mar 24 11:43:56 ip-172-31-44-64.ap-south-1.compute.internal mariadb[962]: Database MariaDB is probably initialized in /var/lib/mysql already, no...done.
Mar 24 11:43:57 ip-172-31-44-64.ap-south-1.compute.internal mysqld_safe[1037]: 220324 11:43:57 mysqld_safe Logging to '/var/log/mariadb/mariadb.log'.
Mar 24 11:43:58 ip-172-31-44-64.ap-south-1.compute.internal mysqld_safe[1037]: 220324 11:43:58 mysqld_safe Starting mysqld daemon with dataBases from /var/lib/mysql
Mar 24 11:44:00 ip-172-31-44-64.ap-south-1.compute.internal systemd[1]: Started MariaDB database server.
Hint: Some lines were ellipsized, use -l to show in full.
[centos@ip-172-31-44-64 ~]$ sudo systemctl status memcached
● memcached.service - Memcached
  Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; vendor preset: disabled)
  Active: active (running) since Thu 2022-03-24 11:43:56 UTC; 7min ago
    Main PID: 979 (memcached)
   CGroup: /system.slice/memcached.service
         └─979 /usr/bin/memcached -u memcached -p 11211 -m 64 -c 1024

Mar 24 11:43:56 ip-172-31-44-64.ap-south-1.compute.internal systemd[1]: Started Memcached.
[centos@ip-172-31-44-64 ~]$ sudo systemctl status rabbitmq-server
● rabbitmq-server.service - RabbitMQ broker
  Loaded: loaded (/usr/lib/systemd/system/rabbitmq-server.service; enabled; vendor preset: disabled)
  Active: active (running) since Thu 2022-03-24 11:44:04 UTC; 7min ago
    Main PID: 964 (beam.smp)
   Status: "Initialized"
   CGroup: /system.slice/rabbitmq-server.service
         └─964 /usr/lib/erlang/erts-11.2/bin/beam.smp -W w -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcbs 30 -P 1048576 -t 5000000 -stbt db -zdb...
Mar 24 11:44:04 ip-172-31-44-64.ap-south-1.compute.internal rabbitmq-server[964]: TLS Library: OpenSSL - OpenSSL 1.0.2k-fips 26 Jan 2017
Mar 24 11:44:04 ip-172-31-44-64.ap-south-1.compute.internal rabbitmq-server[964]: Doc guides: https://rabbitmq.com/documentation.html
Mar 24 11:44:04 ip-172-31-44-64.ap-south-1.compute.internal rabbitmq-server[964]: Support: https://rabbitmq.com/contact.html
Mar 24 11:44:04 ip-172-31-44-64.ap-south-1.compute.internal rabbitmq-server[964]: Tutorials: https://rabbitmq.com/getstarted.html
```

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

Step4.2.1: (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 :

Windows server creation:

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:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
Nexus-server	i-06c90075e179b649	Stopped	t2.medium	-	No alarms	ap-south-1b
app01-staging	i-0801d0c8ec2054a13	Stopped	t2.micro	-	No alarms	ap-south-1b
windows-server	i-0c707a773e258efb0	Running	t2.medium	-	No alarms	ap-south-1b
be01-staging	i-00b5b4321e2429623	Stopped	t2.micro	2/2 checks passed	No alarms	ap-south-1a
app01-prod	i-006c996a84b900f7a	Stopped	t2.micro	-	No alarms	ap-south-1a

Instance: i-0c707a773e258efb0 (windows-server)

Platform	AMI ID	Monitoring
windows	ami-0a4a4775bdbcc44e58	disabled
Platform details	AMI name	Termination protection
Windows	Windows_Server-2019-English-Full-Base-2022.02.10	Disabled
Launch time	AMI location	Lifecycle
Fri Mar 25 2022 17:19:40 GMT+0530 (India)	amazon/Windows_Server-2019-English-Full-Base-	normal

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

Connect to instance [Info](#)

Connect to your instance i-0c707a773e258efb0 (windows-server) using any of these options

Session Manager | **RDP client** | EC2 Serial Console

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

[Download remote desktop file](#)

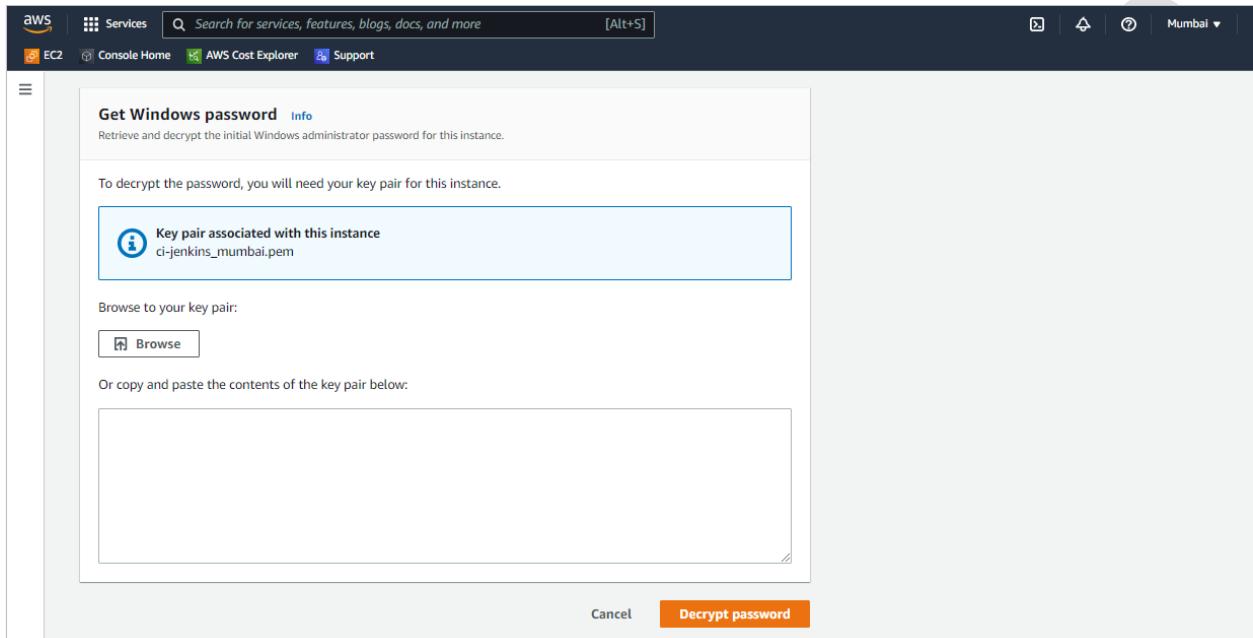
When prompted, connect to your instance using the following details:

Private IP	User name
172.31.2.101	Administrator
Password	Get password

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

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:

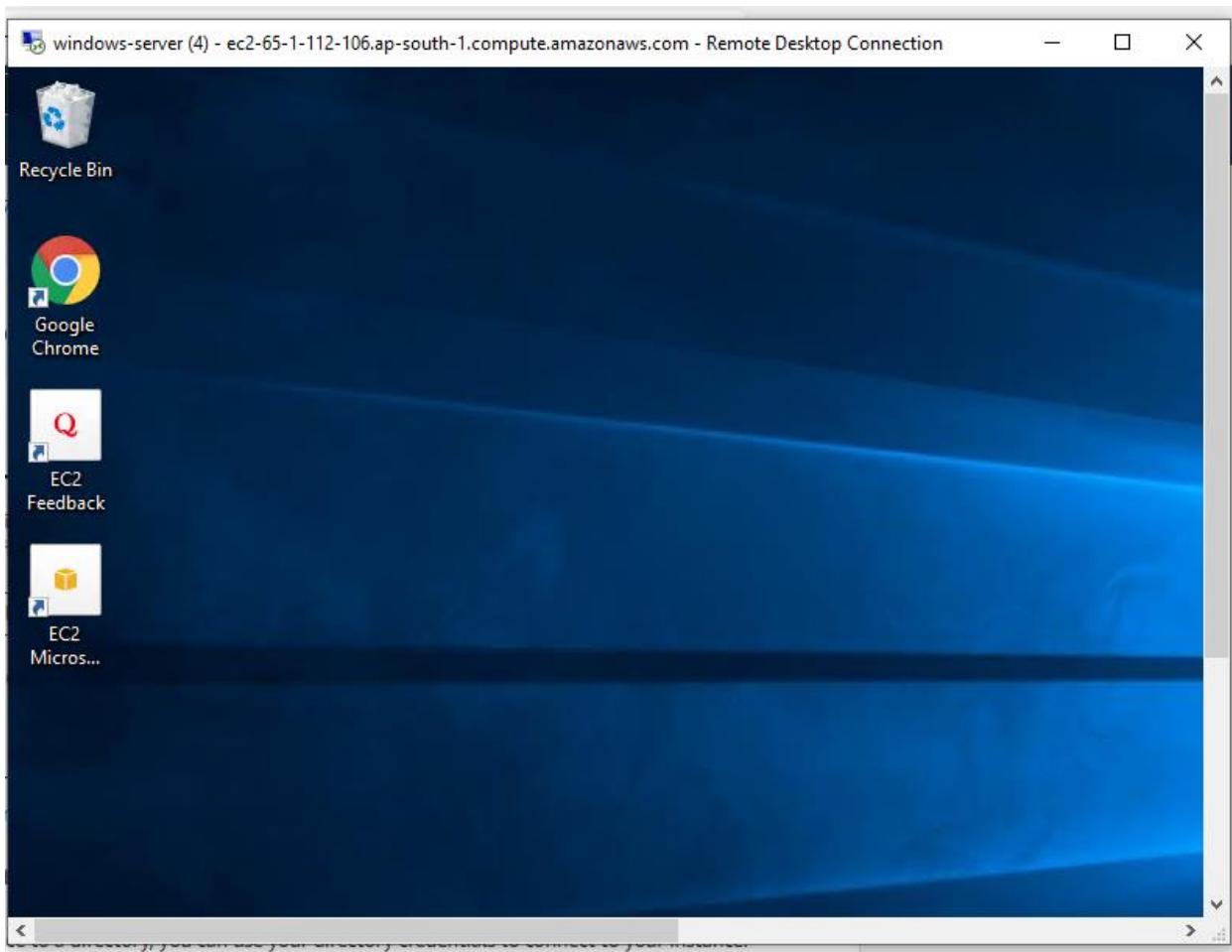
1. Keep the private key of the keypair handy. (it is the file with .pem extension)



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
-

The screenshot shows the Jenkins Dashboard. At the top, there is a red box highlighting several security vulnerabilities:

- Build Pipeline Plugin 1.5.8: Stored XSS vulnerability
- Git plugin 4.8.1: Stored XSS vulnerability
- Pipeline: Groovy 2.9.3: OS command execution vulnerabilities in Pipeline-related plugins, Sensitive information disclosure, Vulnerabilities in multiple Pipeline-related plugins allow reading arbitrary files on the controller

Below the vulnerabilities, there are sections for System Configuration, Security, and Manage Plugins.

System Configuration

- Configure System: Configure global settings and paths.
- Global Tool Configuration: Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins. There are updates available.

Security

- Configure Global Security: Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials: Configure credentials.
- Configure Credential Providers: Configure the credential providers and types.
- Manage Users: Create/delete/modify users that can log in to this Jenkins.
- In-process Script Approval: Allows a Jenkins administrator to review proposed scripts written in Groovy.

The screenshot shows the Jenkins Global Tool Configuration page for Ansible. The URL is `3.111.167.81:8080/configureTools/`.

Maven

Maven installations: Add Maven (button). List of Maven installations on this system.

Ansible

Ansible installations: Add Ansible (button). List of Ansible installations on this system.

Ansible configuration:

- Name: ansible2
- Path to ansible executables directory: /usr/bin
- Install automatically

Buttons: Add Ansible, Save, Apply, Delete Ansible.

Jenkins version: Jenkins 2.289.3

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

The screenshot shows the Jenkins Global credentials (unrestricted) interface. At the top, there's a navigation bar with 'Dashboard', 'Credentials', 'System', and 'Global credentials (unrestricted)'. Below the navigation, there are two buttons: 'Back to credential domains' and 'Add Credentials'. The main title 'Global credentials (unrestricted)' is displayed with a castle icon. A sub-header says 'Credentials that should be available irrespective of domain specification to requirements'. There is some very small, illegible text at the bottom of the page.

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

The screenshot shows the Jenkins credential configuration for 'ubuntu (app-stage-ssh-login)'. The URL is 'Dashboard > Credentials > System > Global credentials (unrestricted) > ubuntu (app-stage-ssh-login)'. On the left, there's a sidebar with 'Back to Global credentials (unrestricted)', 'Update', 'Delete', and 'Move' buttons. The main form has fields for 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'ID' ('app-stage-ssh-login'), 'Description' ('app-stage-ssh-login'), 'Username' ('ubuntu'), and 'Private Key'. Under 'Private Key', there's a radio button for 'Enter directly' and a text area labeled 'Key' containing a concealed value. A 'Replace' button is also visible. At the bottom, there's a 'Save' button.

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 previous 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/

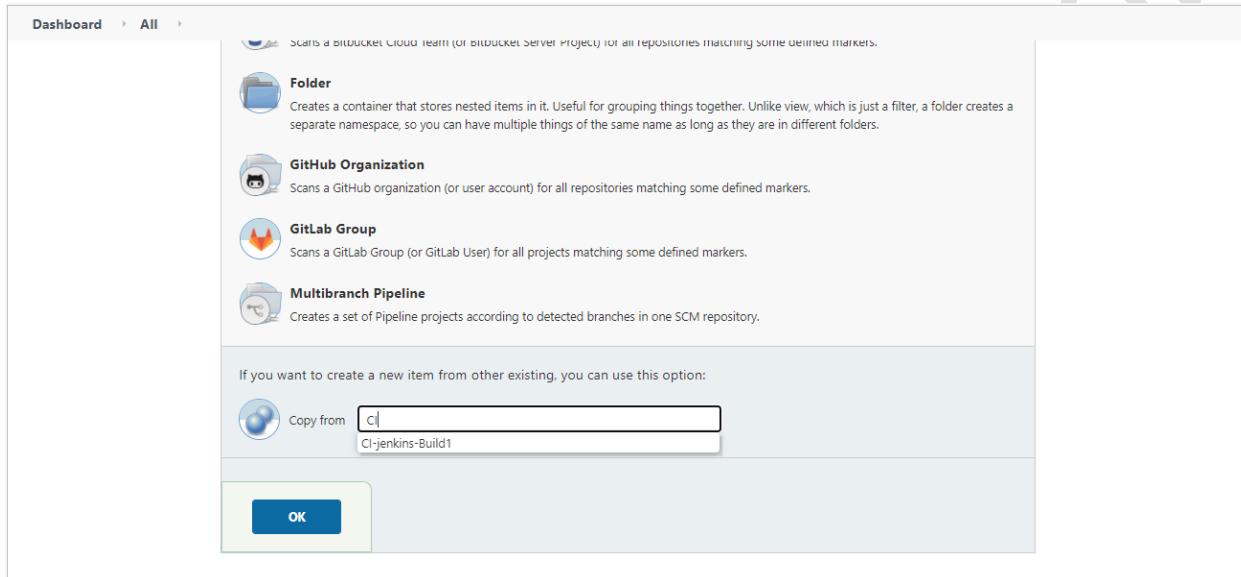
  rescue:
    - name: Start tomcat svc
      service:
        name: tomcat
        state: started

  - 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.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

21

The screenshot shows the Jenkins configuration interface for a job named 'Deploy-to-Staging-Ansible'. The 'General' tab is selected. In the 'Post-build Actions' section, there is a text area containing the description: 'This job is to deploy the artifact from nexus to web-server using Ansible'. Below this, under 'Job Notifications', there is a 'Notification Endpoints' section with a 'Save' button. Under 'Office 365 Connector', there is a 'Notification webhooks' section with a 'Save' button. At the bottom of the page are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins configuration interface for the same job. The 'Office 365 Connector' tab is selected. In the 'Post-build Actions' section, the 'This project is parameterized' checkbox is checked. A dashed box highlights the 'String Parameter' section where two parameters are defined: 'TIME' and 'ID'. Both parameters have their 'Name' fields set to 'TIME' and 'ID' respectively. The 'Save' and 'Apply' buttons are visible at the bottom.

The screenshot shows the configuration of a Jenkins parameter. It is a 'String Parameter' with the following fields:

- Name:** ID
- Default Value:** (empty)
- Description:** (empty)
- [Plain text] Preview:** (empty)
- Trim the string
- Add Parameter** dropdown menu:
 - Throttle builds
 - Disable this project
 - Execute concurrent builds if necessary
 - Restrict where this project can be run
- Advanced...** button

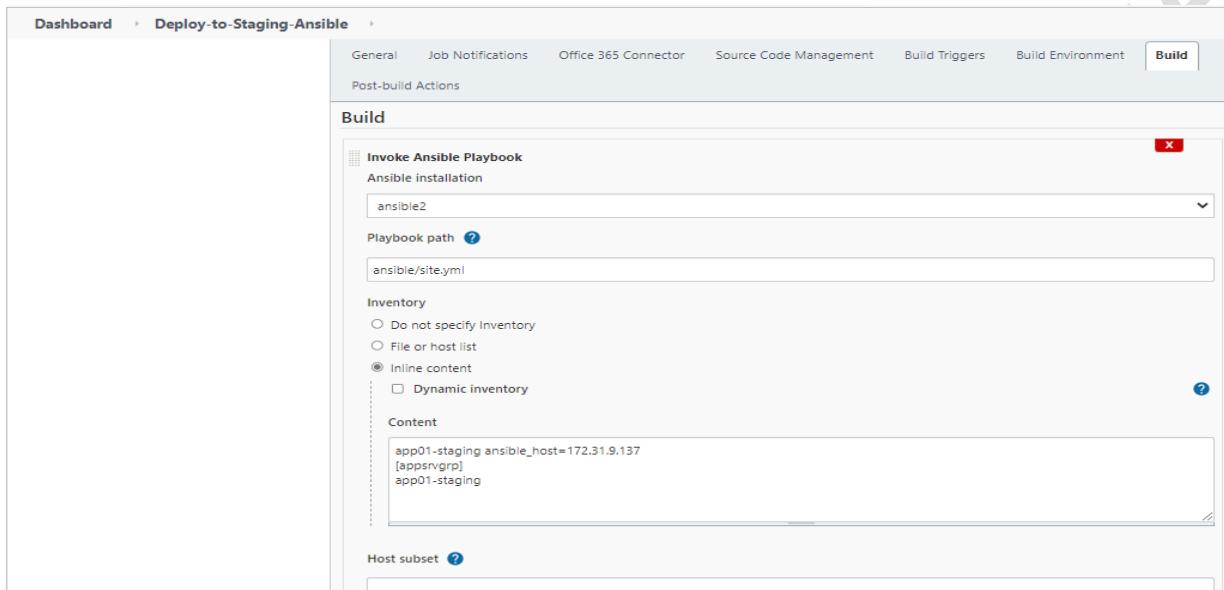
The screenshot shows the Jenkins Source Code Management configuration for a Git repository:

- Source Code Management:** Git
- Repositories:**
 - Repository URL:** https://github.com/gnsharma530/Logiclabs-cidemo.git
 - Credentials:** - none - (dropdown) | Add (button)
 - Advanced...** button
 - Add Repository** button
- Branches to build:**
 - Branch Specifier (blank for 'any'):** */cd-ansible-jenkins
 - Add Branch** button

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.

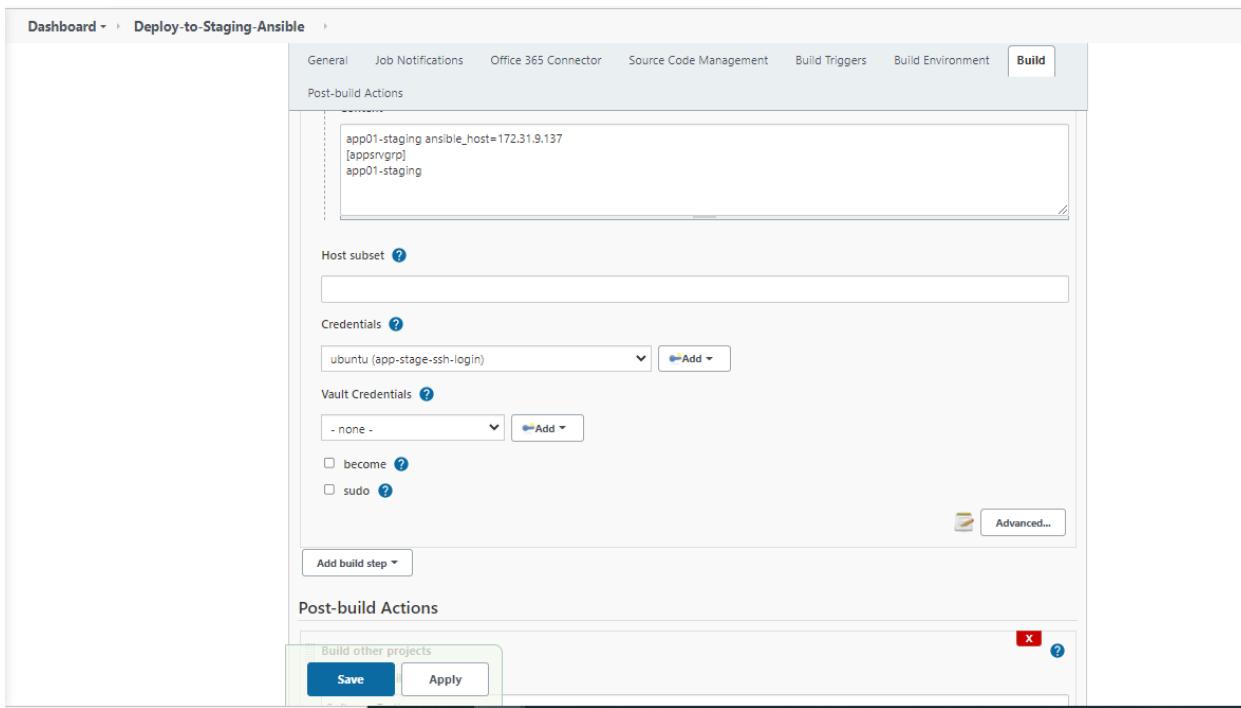


Enter the data in the “content” setting as given below :

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

[appsvrgrp]

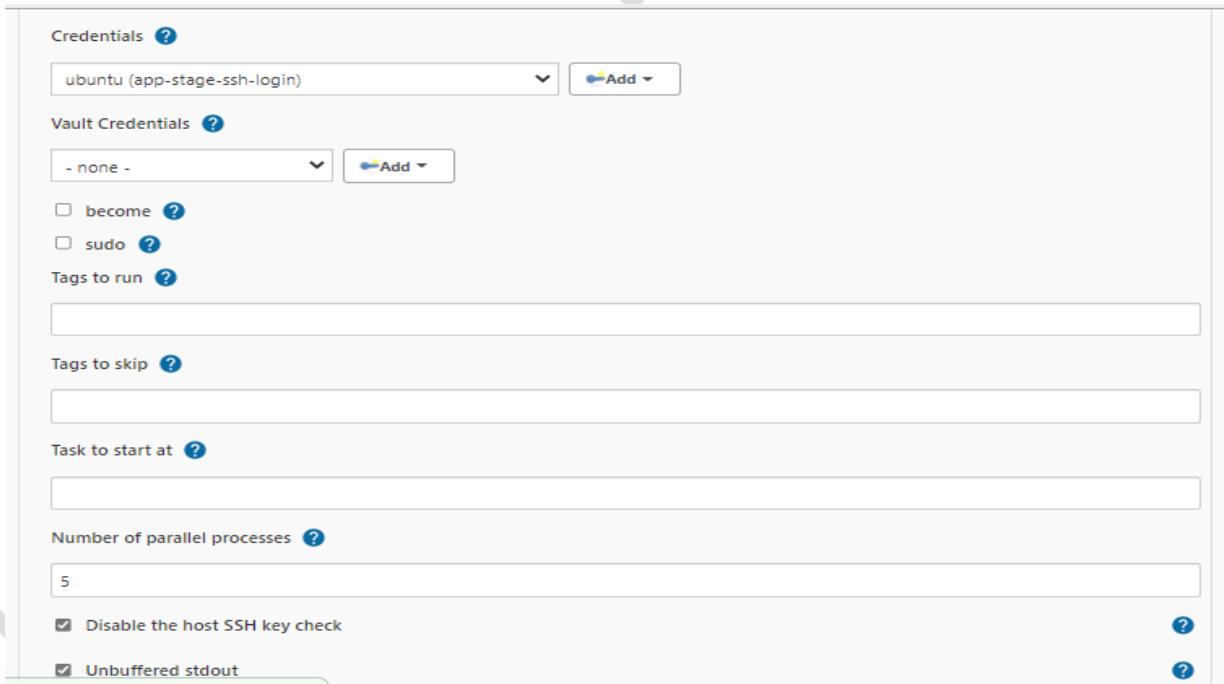
app01-staging (This is the host name of web-server given while creating)



The screenshot shows the Jenkins configuration interface for a job named 'Deploy-to-Staging-Ansible'. The 'Build' tab is active. In the 'Post-build Actions' section, there is a 'Host subset' field which contains the text 'app01-staging ansible_host=172.31.9.137 [appsvngrp] app01-staging'. Below this is a 'Credentials' dropdown set to 'ubuntu (app-stage-ssh-login)'. There are also fields for 'Vault Credentials' (set to 'none'), 'become' (unchecked), and 'sudo' (unchecked). At the bottom of the 'Post-build Actions' section, there is a 'Save' button.

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:



The screenshot shows the 'Advanced...' tab configuration for the build step. It includes sections for 'Credentials' (set to 'ubuntu (app-stage-ssh-login)'), 'Vault Credentials' (set to 'none'), 'become' (unchecked), 'sudo' (unchecked), 'Tags to run' (empty), 'Tags to skip' (empty), 'Task to start at' (empty), 'Number of parallel processes' (set to 5), and two checked checkboxes: 'Disable the host SSH key check' and 'Unbuffered stdout'.

Enter the extra variables as given below:

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

25

Number of parallel processes [?](#)

5

Disable the host SSH key check [?](#)

Unbuffered stdout [?](#)

Colorized stdout [?](#)

Extra Variables

Key	USER	X
Value	admin	
<input type="checkbox"/> Hidden variable in build log		
Key	PASS	X
Value	admin	
<input checked="" type="checkbox"/> Hidden variable in build log		

Key	nexusip	X
Value	172.31.8.182	
<input type="checkbox"/> Hidden variable in build log		
Key	reponame	X
Value	logiclabs-release	
<input type="checkbox"/> Hidden variable in build log		
Key	groupid	X
Value	QA	
<input type="checkbox"/> Hidden variable in build log		
Key	time	X
Value	\$TIME	

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

The screenshot shows the Jenkins build configuration page. Under the 'Environment' section, there are four entries:

- Key: time, Value: \$TIME, with a checked checkbox for 'Hidden variable in build log'.
- Key: build, Value: \$ID, with a checked checkbox for 'Hidden variable in build log'.
- Key: vprofile_version, Value: \$TIME-\$ID.war, with a checked checkbox for 'Hidden variable in build log'.
- An 'Add Extra Variable' button and an 'Additional parameters' field.

Below the environment section, there is a 'Post-build Actions' section which is currently empty.

S.No	Key	Value
1	USER	admin
2	PASS	admin
3	nexusip	Xx.xx.xx.xx (your nexus server ip)
4	reponame	Logicleabs-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:

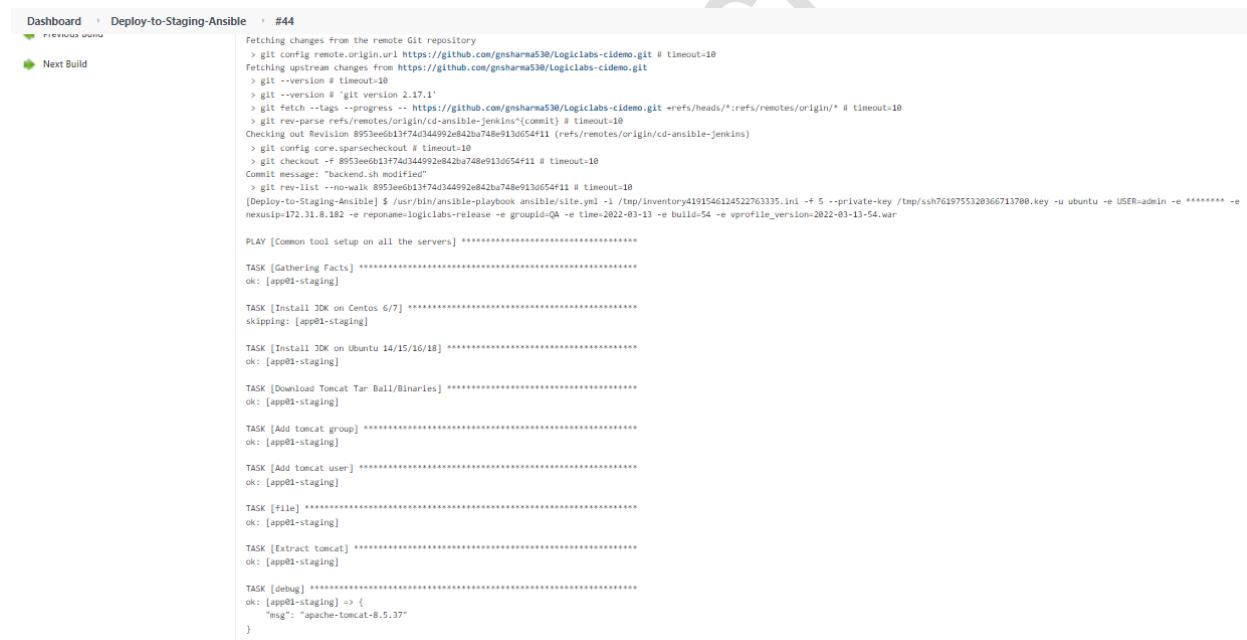
- \$ which python #if python is installed, it would return it's version.

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.



The screenshot shows the Jenkins log for the 'Deploy-to-Staging-Ansible' job, build #44. The log output is as follows:

```

Dashboard > Deploy-to-Staging-Ansible > #44
  Previous build
  Next Build

Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/gnsharma530/Logiclabs-cidemo.git # timeout=10
Fetching upstream changes from https://github.com/gnsharma530/Logiclabs-cidemo.git
> git -version # timeout=10
> git -version # 'git version 2.17.1'
> git fetch -tags --progress -- https://github.com/gnsharma530/Logiclabs-cidemo.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/cd-ansible-jenkins^{commit} # timeout=10
Checking out Revision 8953eed013f74d344992e842ba748e913d054f11 (refs/remotes/origin/cd-ansible-jenkins)
> git config core.sparsecheckout # timeout=10
> git checkout -f 8953eed013f74d344992e842ba748e913d054f11 # timeout=10
Commit message: "backend.sh modified"
> git rev-list --no-walk 8953eed013f74d344992e842ba748e913d054f11 # timeout=10
[Deploy-to-Staging-Ansible] $ /usr/bin/ansible-playbook ansible/site.yml -i /tmp/inventory4191546124522763335.ini -f 5 --private-key /tmp/ssh7619755320366713700.key -u ubuntu -e USER=admin -e ***** -e nexusip=172.31.8.182 -e repename=logiclabs-release -e groupid=QA -e time=2022-01-13 -e build=54 -e vprofile_version=2022-01-13-54.war

PLAY [Common tool setup on all the servers] ****
TASK [Gathering Facts] ****
ok: [app01-staging]

TASK [Install JDK on Centos 6/7] ****
skipping: [app01-staging]

TASK [Install JDK on Ubuntu 14/15/16/18] ****
ok: [app01-staging]

TASK [Download Tomcat Tar Ball/Binaries] ****
ok: [app01-staging]

TASK [Add tomcat group] ****
ok: [app01-staging]

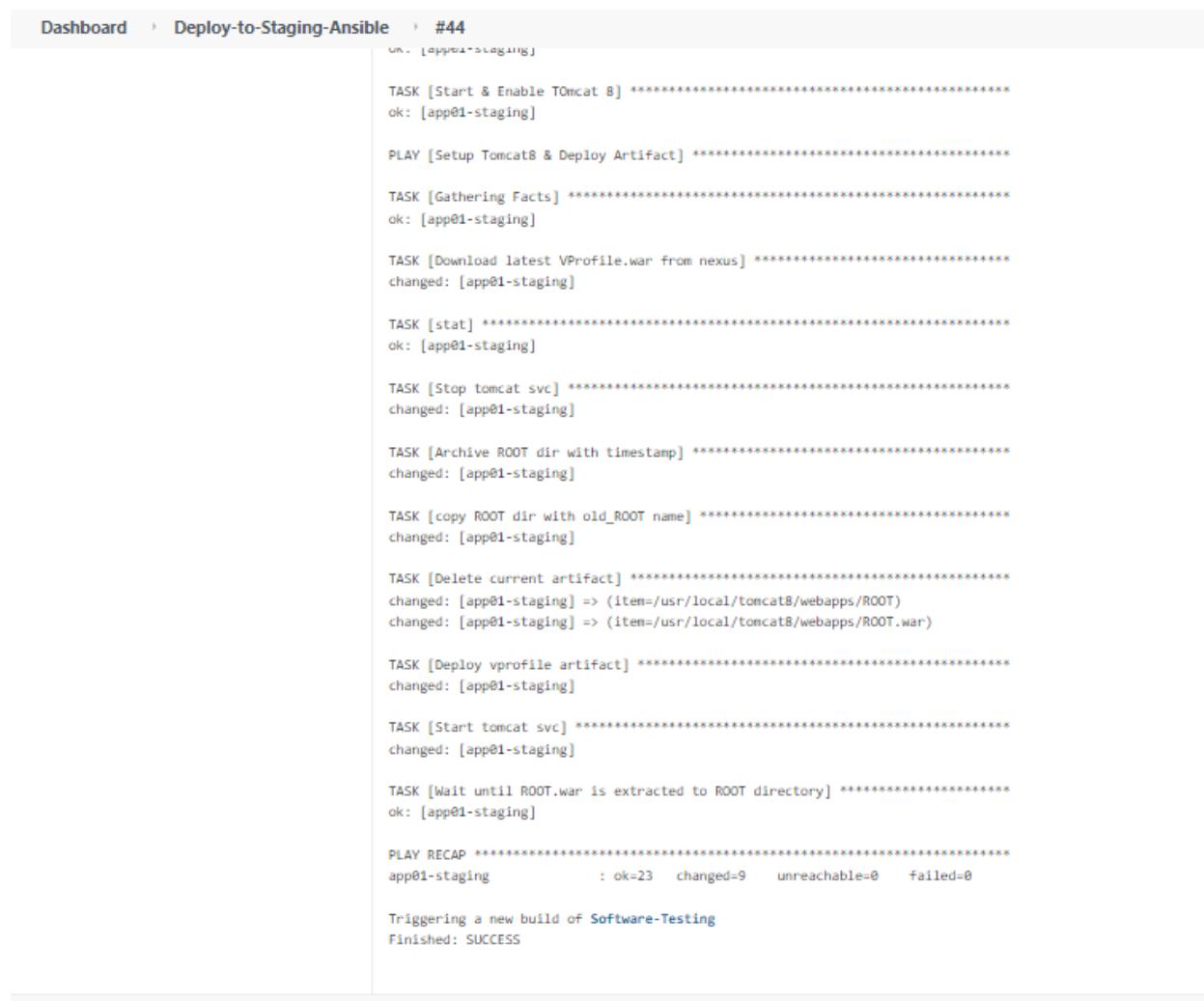
TASK [Add tomcat user] ****
ok: [app01-staging]

TASK [File] ****
ok: [app01-staging]

TASK [Extract tomcat] ****
ok: [app01-staging]

TASK [debug] ****
ok: [app01-staging] => {
    "msg": "apache-tomcat-8.5.37"
}

```



The screenshot shows the Jenkins job log for 'Deploy-to-Staging-Ansible' #44. The log displays the execution of an Ansible playbook. The tasks include starting and enabling Tomcat 8, gathering facts, downloading the latest VProfile.war from Nexus, stopping the tomcat service, archiving the ROOT directory with a timestamp, copying the ROOT directory with the old_ROOT name, deleting the current artifact, deploying the vprofile artifact, starting the tomcat service, and waiting until ROOT.war is extracted to the ROOT directory. The final output shows a PLAY RECAP with 23 successful tasks, 9 changed tasks, 0 unreachable hosts, and 0 failed tasks. The job is triggered by a new build of 'Software-Testing' and is finished successfully.

```

Dashboard > Deploy-to-Staging-Ansible > #44
on app01-staging

TASK [Start & Enable Tomcat 8] *****
ok: [app01-staging]

PLAY [Setup Tomcat8 & Deploy Artifact] *****

TASK [Gathering Facts] *****
ok: [app01-staging]

TASK [Download latest VProfile.war from nexus] *****
changed: [app01-staging]

TASK [stat] *****
ok: [app01-staging]

TASK [Stop tomcat svc] *****
changed: [app01-staging]

TASK [Archive ROOT dir with timestamp] *****
changed: [app01-staging]

TASK [copy ROOT dir with old_ROOT name] *****
changed: [app01-staging]

TASK [Delete current artifact] *****
changed: [app01-staging] => (item=/usr/local/tomcat8/webapps/ROOT)
changed: [app01-staging] => (item=/usr/local/tomcat8/webapps/ROOT.war)

TASK [Deploy vprofile artifact] *****
changed: [app01-staging]

TASK [Start tomcat svc] *****
changed: [app01-staging]

TASK [Wait until ROOT.war is extracted to ROOT directory] *****
ok: [app01-staging]

PLAY RECAP *****
app01-staging : ok=23    changed=9    unreachable=0    failed=0

Triggering a new build of Software-Testing
Finished: SUCCESS

```

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.

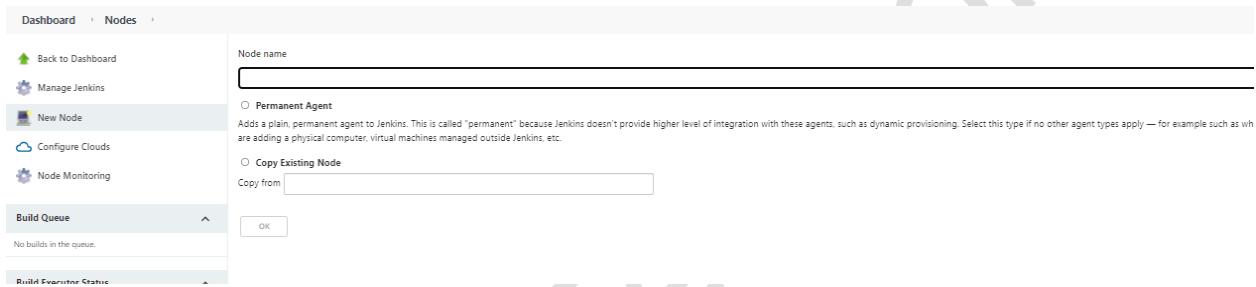
Step 6: Configuring and Running Selenium Automated Test cases

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



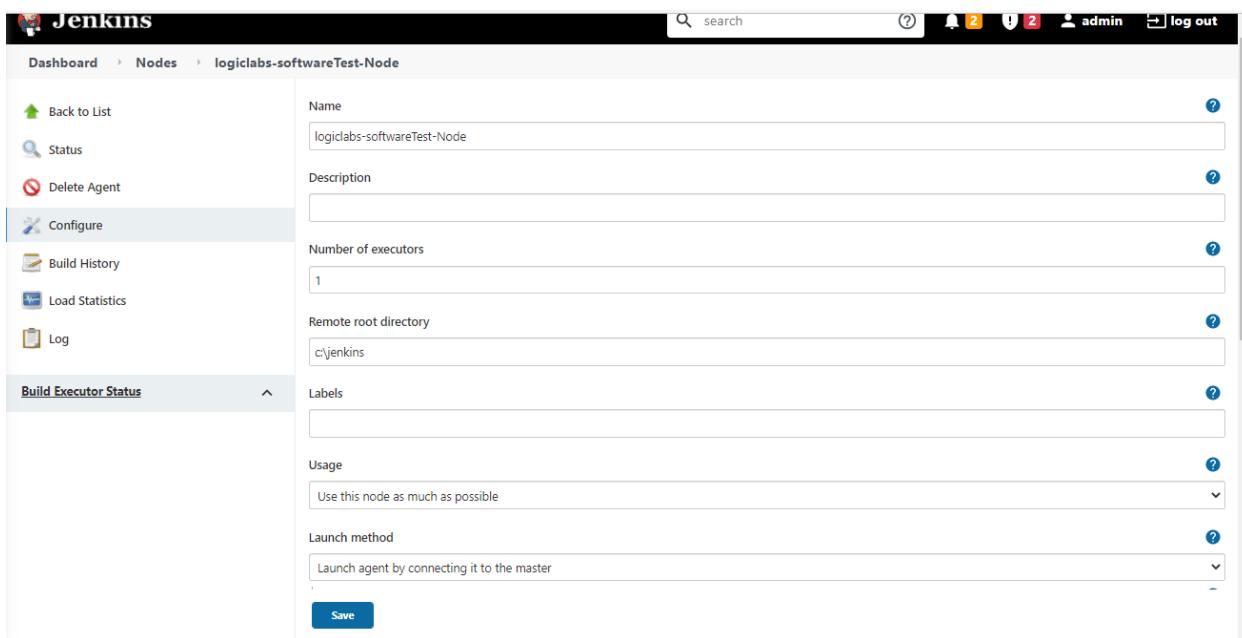
Give the name logiclabs-softwareTest-Node and select permanent agent and select ok.

Step 6.2

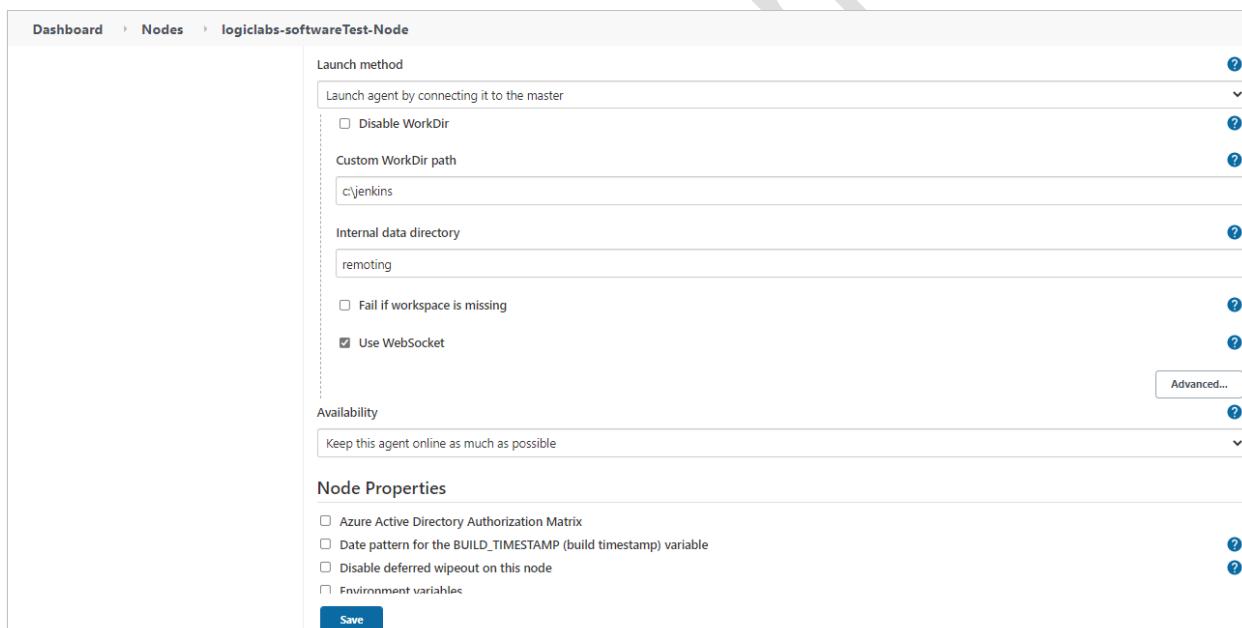
Configure the node as given below:

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

30



The screenshot shows the Jenkins 'Configure' page for a node named 'logiclabs-softwareTest-Node'. The left sidebar includes links for Back to List, Status, Delete Agent, and Log. The main configuration area includes fields for Name (logiclabs-softwareTest-Node), Description, Number of executors (1), Remote root directory (c:\jenkins), Labels, Usage (Use this node as much as possible), Launch method (Launch agent by connecting it to the master), and a Save button.



This screenshot shows an expanded view of the Jenkins node configuration for 'logiclabs-softwareTest-Node'. It includes the 'Launch method' section with 'Launch agent by connecting it to the master' selected, and options for 'Disable WorkDir' (unchecked), 'Custom WorkDir path' (c:\jenkins), 'Internal data directory' (remoting), 'Fail if workspace is missing' (unchecked), and 'Use WebSocket' (checked). There is also an 'Advanced...' button. Below this is the 'Availability' section with the 'Keep this agent online as much as possible' option. The 'Node Properties' section contains checkboxes for 'Azure Active Directory Authorization Matrix', 'Date pattern for the BUILD_TIMESTAMP (build timestamp) variable', 'Disable deferred wipeout on this node', and 'Environment variables', all of which are unchecked. A final 'Save' button is at the bottom.

Add the node.

Step 6.3 Connect the node with the master

Agent logiclabs-softwareTest-Node

Connect agent to Jenkins one of these ways:

- Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://172.31.32.213:8080/computer/logiclabs-softwareTest-Node/jenkins-agent.jnlp -secret 19ba45357e2b4a880b0bafc0e3dd3bd927e891da2d916d345c173b13bd07cefb -workDir "c:\jenkins"
```

Run from agent command line, with the secret stored in a file:

```
echo 19ba45357e2b4a880b0bafc0e3dd3bd927e891da2d916d345c173b13bd07cefb > secret-file
java -jar agent.jar -jnlpUrl http://172.31.32.213:8080/computer/logiclabs-softwareTest-Node/jenkins-agent.jnlp -secret @secret-file -workDir "c:\jenkins"
```

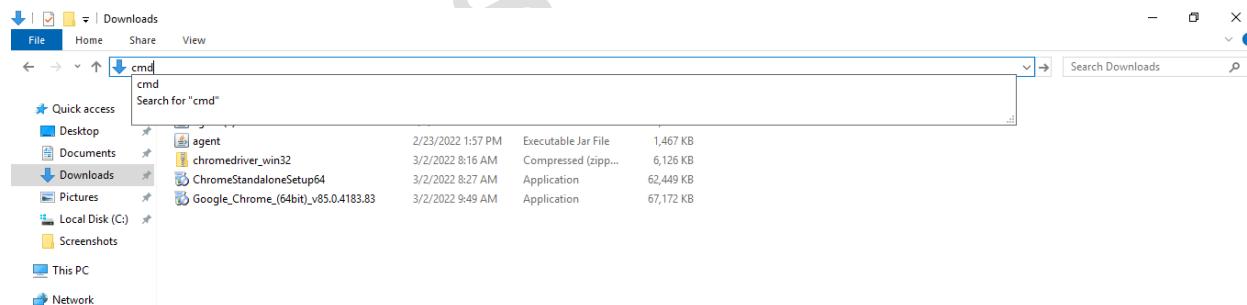
Initially the node will be offline and we need to configure it to bring it online.

- 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.

- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://172.31.32.213:8080/computer/logiclabs-softwareTest-Node/jenkins-agent.jnlp -secret 19ba45357e2b4a880b0bafc0e3dd3bd927e891da2d916d345c173b13bd07cefb -workDir "c:\jenkins"
```

Open command prompt from downloads folder (open downloads folder, press Alt + d → clear the contents and enter cmd and press enter)

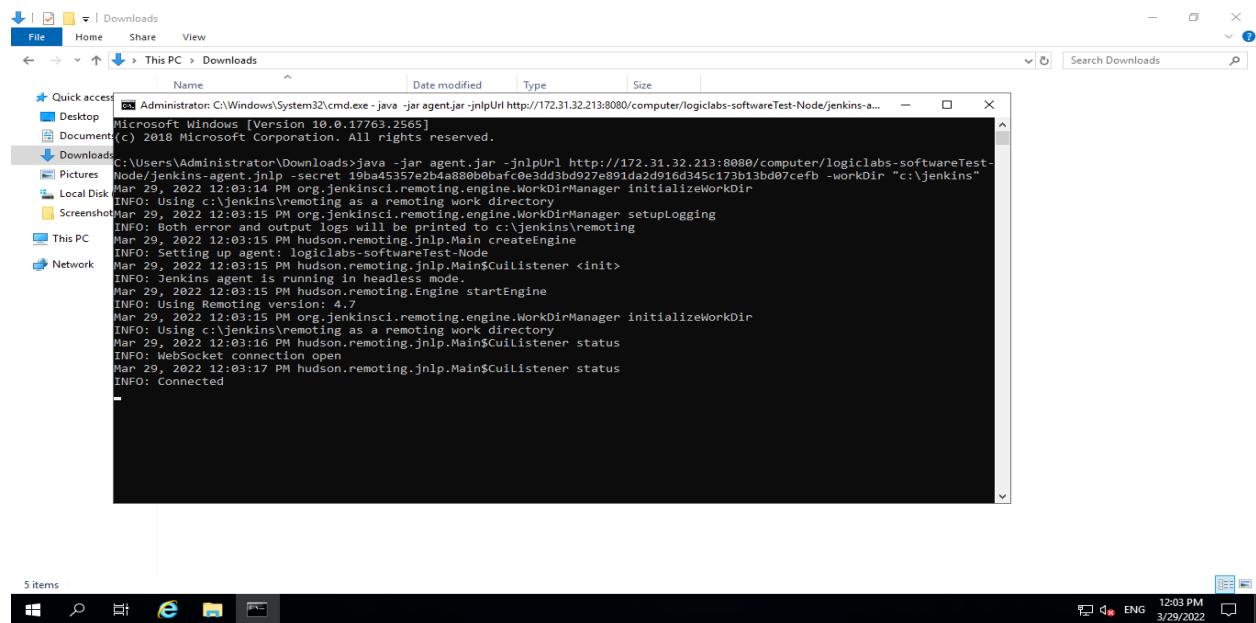


Run the command given below the Run from agent command line prompt.

For eg:

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

19ba45357e2b4a880b0bafc0e3dd3bd927e891da2d916d345c173b13bd07cef
b -workDir "c:\jenkins"



Now your node should look online.

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Dashboard', 'Nodes', and the specific node 'logiclabs-softwareTest-Node'. Below the navigation, there's a search bar and some user icons. The main content area displays the node details: 'Agent logiclabs-softwareTest-Node' and 'Agent is connected.'. There are also links for 'Back to List' and 'Status'. A button at the top right says 'Mark this node temporarily offline'.

Step 6.4 Create Jenkins job to run Selenium software test cases

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

33

The screenshot shows the Jenkins configuration interface for the 'Software-Testing' job. The 'General' tab is selected. Under 'Post-build Actions', there is a 'Description' text area and a 'Date and Time Pattern' field set to 'yyyy-MM-dd'. Other options like 'Discard old builds' and 'GitHub project' are available. The 'Job Notifications' section includes a 'Notification Endpoints' button. The top navigation bar shows 'Dashboard > Software-Testing' and various Jenkins status indicators.

The screenshot shows the Jenkins configuration interface for the 'Software-Testing' job, specifically the 'Office 365 Connector' tab. Under 'Post-build Actions', there is a checkbox for 'This project is parameterized', which is checked. A 'String Parameter' section is expanded, showing fields for 'Name' (set to 'TIME'), 'Default Value', and 'Description'. There is also a 'Trim the string' checkbox. The top navigation bar shows 'Dashboard > Software-Testing' and various Jenkins status indicators.

The screenshot shows the Jenkins project configuration interface. The top navigation bar includes tabs for General, Job Notifications, Office 365 Connector (which is selected), Source Code Management, Build Triggers, Build Environment, and Build. Below the tabs, there is a section titled "Post-build Actions". Under "Post-build Actions", there is a "String Parameter" configuration. The "String Parameter" section contains fields for Name (set to "ID"), Default Value (empty), and Description (empty). There is also a "[Plain text] Preview" area with a checkbox for "Trim the string". Below this, there is a "Label Expression" dropdown menu set to "logiclabs-softwareTest-Node". To the right of the "Label Expression" dropdown, there are five question mark icons. At the bottom of the "String Parameter" section, there is an "Add Parameter" button with a dropdown arrow.

String Parameter

Name ?
ID

Default Value ?

Description ?

[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 logiclabs-softwareTest-Node

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

The screenshot shows the Jenkins project configuration interface. The top navigation bar includes tabs for General, Job Notifications, Office 365 Connector, Source Code Management (which is selected and highlighted in blue), Build Triggers, Build Environment, and Build. The main content area is divided into sections: Post-build Actions, Label Expression, Source Code Management, and Advanced options.

Post-build Actions: Contains checkboxes for Throttle builds, Disable this project, Execute concurrent builds if necessary, and Restrict where this project can be run. The 'Restrict where this project can be run' checkbox is checked.

Label Expression: A text input field contains the label `logilabs-softwareTest-Node`. Below it, a message states: "Label logilabs-softwareTest-Node matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list." An "Advanced..." button is located at the bottom right of this section.

Source Code Management: This section is expanded, showing the following configuration:

- Source Code Management type: Git (radio button selected)
- Repository URL: `https://github.com/gnsharma530/Logilabs-cidemo.git`
- Credentials: A dropdown menu shows "- none -" and an "Add" button.

An "Advanced..." button is located at the bottom right of the Source Code Management section.

Change the branch to */seleniumAutoScripts.

The screenshot shows the Jenkins project configuration interface for a specific job. The top navigation bar includes tabs for General, Job Notifications, Office 365 Connector, Source Code Management (which is selected), Build Triggers, Build Environment, and Build. Below the tabs, there are sections for Post-build Actions, Branches to build, Repository browser, and Additional Behaviours. Under Build Triggers, a list of triggers is shown. Under Build Environment, several environment-related checkboxes are listed.

Source Code Management

Post-build Actions

Branches to build

Branch Specifier (blank for 'any')
*/seleniumAutoScripts

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

The screenshot shows the Jenkins 'Build' configuration page. In the 'Goals' section, the command is set to 'clean test -Dsurefire.suiteXmlFiles=testng.xml -Durl=http://172.31.9.137:8080//login -Dusr=admin_vp -Dpass=admin_vp -DsShotPath=C:\jenkins'. In the 'Post-build Actions' section, there is a 'Build Triggers' item.

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/branch-heads/85@{#1})  
server did not provide any stacktrace information)
```

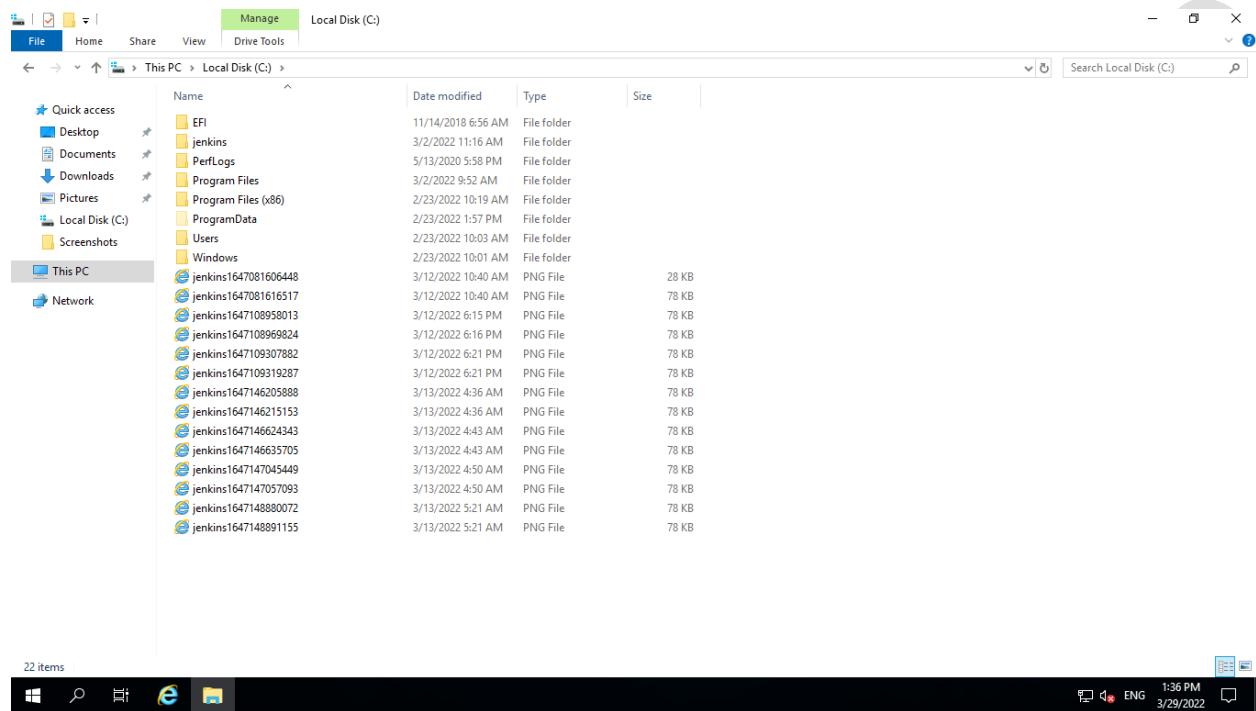
a. **Solution:** if windows server chrome version is latest then downgrade to exactly version 85 and disable auto update. Google to know how to downgrade the chrome version and disable updates.

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

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

39

The screenshot shows the Jenkins General configuration page. The top navigation bar includes tabs for General, Job Notifications, Office 365 Connector, Source Code Management, Build Triggers, Build Environment, and Build. The General tab is selected. The Post-build Actions section contains a text area with the placeholder "First Build Job". Below this is a link "[Plain text] Preview". Under the Job Notifications section, there is a "Notification Endpoints" section with an "Add Endpoint" button. The Office 365 Connector section includes a "Notification webhooks" section with an "Add Webhook" button.

The screenshot shows the Jenkins Office 365 Connector configuration page. The top navigation bar includes tabs for General, Job Notifications, Office 365 Connector, Source Code Management, Build Triggers, Build Environment, and Build. The Office 365 Connector tab is selected. The Post-build Actions section lists several options: "Permission to Copy Artifact", "This build requires lockable resources", "This project is parameterized", "Throttle builds", "Disable this project", "Execute concurrent builds if necessary", and "Restrict where this project can be run" (with the checkbox checked). Below this is a "Label Expression" field containing "master". A note states "Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list." An "Advanced..." button is also present. The Source Code Management section shows "Git" selected as the provider, with a "Repository URL" field containing "https://github.com/gnsharma530/Logiclabs-cidemo.git". The Credentials section is partially visible at the bottom.

The screenshot shows the 'Source Code Management' configuration page for a Jenkins job. The 'Git' option is selected under 'Post-build Actions'. The 'Repository URL' is set to `https://github.com/gnsharma530/Logiclabs-cidemo.git`. Under 'Credentials', there is a dropdown menu set to '- none -' and an 'Add' button. To the right are 'Advanced...' and 'Add Repository' buttons. The 'Branches to build' section contains a 'Branch Specifier' field with the value `*cd-ansible-jenkins`, which has a red 'X' icon and a question mark icon. An 'Add Branch' button is also present. The 'Repository browser' section is at the bottom with a question mark icon.

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` [X](#) [?](#)

[Add Branch](#)

Repository browser [?](#)

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

Build

Execute shell

Command

```
cat << EOT > src/main/resources/application.properties
#JDBC Configuration for Database Connection
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://172.31.44.64:3306/accounts?useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull
jdbc.username=admin
jdbc.password=admin123

#Memcached Configuration For Active and StandBy Host
#for Active Host
memcached.active.host=172.31.44.64
memcached.active.port=11211
#for StandBy Host
memcached.standBy.host=127.0.0.2
memcached.standBy.port=11211

#RabbitMq Configuration
rabbitmq.address=172.31.44.64
rabbitmq.port=5672
rabbitmq.username=test
rabbitmq.password=test

#Elasticsearch Configuration
elasticsearch.host =172.31.44.64
elasticsearch.port =9300
elasticsearch.cluster=vprofile
elasticsearch.node=vprofilenode
EOT
```

[See the list of available environment variables](#)

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

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

42

The screenshot shows the 'Build' configuration tab for a Jenkins job. It includes sections for Goals (with 'install -DskipTests'), POM (empty), Properties (environment variables like SNAP-REPO, NEXUS-USER, etc.), JVM Options (empty), and Settings file (empty). A large watermark 'LogicLabs' is diagonally across the page.

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

The screenshot shows the Jenkins job configuration interface for a Maven build. The top navigation bar includes General, Job Notifications, Office 365 Connector, Source Code Management, Build Triggers, Build Environment, and Build tabs. The Build tab is selected. Under Post-build Actions, there is a section for Maven settings:

```
KLLCLAB-MCPO=logiclabs-release  
CENTRAL-REPO=logiclabs-maven-central  
NEXUS-GRP-REPO=logiclabs-maven-group  
NEXUSIP=172.31.8.182  
NEXUSPORT=8081
```

Below this, there are sections for JVM Options, Inject build variables, and Use private Maven repository. Under Settings file, the dropdown is set to "Settings file in filesystem" and the file path is "settings.xml". Under Global Settings file, the dropdown is set to "Use default maven global settings". A "Add build step" button is located below the settings section. The bottom section is titled "Post-build Actions" and contains a "Archive the artifacts" step, which has a "Save" button and an "Apply" button. A red "X" icon and a blue question mark icon are located in the top right corner of this section.

* make sure to change the Nexus server ip.

The screenshot shows the Jenkins configuration interface for a build job. The top navigation bar includes General, Job Notifications, Office 365 Connector, Source Code Management, Build Triggers, Build Environment, and Build tabs. The Post-build Actions tab is selected, indicated by a highlighted border. A sub-section titled "Post-build Actions" contains two main configurations:

- Archive the artifacts**:
 - Files to archive: `**/*.war`
 - Excludes: (empty field)
 - Checkboxes:
 - Do not fail build if archiving returns nothing
 - Archive artifacts only if build is successful
 - Fingerprint all archived artifacts
 - Use default excludes
 - Treat include and exclude patterns as case sensitive
 - Follow symbolic links
- Build other projects**:
 - Projects to build: `Test`
 - Radio buttons:
 - Trigger only if build is stable
 - Trigger even if the build is unstable
 - Trigger even if the build fails

At the bottom of the configuration area are "Save" and "Apply" buttons.

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

The screenshot shows the Jenkins configuration interface for the 'Test' job. The 'Source Code Management' tab is selected. Under 'Branches to build', the 'Branch Specifier' field contains the value '/cd-ansible-jenkins'. Below this, the 'Repository browser' dropdown is set to '(Auto)'. Under 'Build Triggers', there are four checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', and 'Build when a change is pushed to BitBucket'. At the bottom of the page, there are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration interface for a job named 'Test'. The 'Build' tab is selected. Under 'Post-build Actions', there is a 'Goals' section with a text input field containing 'test' and an 'Advanced...' button. Below it is an 'Add build step ▾' button. The 'Post-build Actions' section contains a 'Build other projects' section for 'Integration'. It includes a 'Projects to build' dropdown set to 'Integration', and three trigger options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. An 'Add post-build action ▾' button is present. At the bottom are 'Save' and 'Apply' buttons. The footer indicates 'REST API' and 'Jenkins 2.289.3'.

Step 7.3: Configuring Integration job for C.D

The screenshot shows the Jenkins configuration interface for a job named 'Integration'. The 'Source Code Management' tab is selected. Under 'Source Code Management', the 'Git' option is selected. The 'Repositories' section shows a 'Repository URL' input field with the value 'https://github.com/gnsharma530/Logiclabs-cidemo.git'. The 'Credentials' section shows a dropdown menu with '- none -' and an 'Add' button. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' input field with the value '/cd-ansible-jenkins'. An 'Add Branch' button is present. At the bottom are 'Save' and 'Apply' buttons. The footer indicates 'REST API' and 'Jenkins 2.289.3'.

The image contains two screenshots of Jenkins job configuration pages. The top screenshot shows the 'Build' section of a Jenkins job. It includes fields for 'Goals' (set to 'verify -DskipUnitTests'), 'POM' (empty), 'Properties' (containing environment variables like SNAP-REPO, NEXUS-USER, NEXUS-PASS, RELEASE-REPO, CENTRAL-REPO, and NEXUS-GRP-REPO), and 'JVM Options' (empty). The bottom screenshot shows the 'Post-build Actions' section, which includes a 'Build other projects' step for 'Code_Analysis'. The 'Projects to build' field contains 'Code_Analysis'. There are three radio button options under 'Trigger only if build is stable': 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom of the page are 'Save' and 'Apply' buttons.

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

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

48

The screenshot shows the Jenkins job configuration for a project named "CI-Jenkins project sonarscanner codeAnalysis".

General Tab:

- Description: CI-Jenkins project sonarscanner codeAnalysis
- [Plain text] Preview
- Post-build Actions:
 - Enable project-based security
 - Change date pattern for the BUILD_TIMESTAMP (build timestamp) variable
 - Discard old builds
 - GitHub project

Job Notifications Tab:

- General Job Notifications Office 365 Connector Source Code Management Build Triggers Build Environment Build
- Post-build Actions
- Advanced...

Source Code Management Tab:

- None (radio button)
- Git (radio button)
- Repositories
 - Repository URL: https://github.com/gnsharma530/Logilabs-cidemo.git
 - Credentials: - none - (dropdown) 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

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=logilabs
sonar.projectName=logilabs-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

IVM Options

The screenshot shows the Jenkins 'Post-build Actions' configuration page. At the top, there is a button labeled 'Add build step ▾'. Below it, the 'Post-build Actions' section is titled 'Build other projects'. It includes a 'Projects to build' field containing 'Deploy-to-Nexus' and three radio button options for triggering: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. To the right of this section are a red 'X' icon and a blue question mark icon. Below this is another section titled 'Quality Gates Sonarqube Plugin'. It has a 'Project Key' field with 'logilabs' entered, a dropdown for 'Job status when analysis fails' set to 'UNSTABLE', and a 'Save' button at the bottom. A large watermark 'Logilabs' is visible across the page.

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.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

51

SonarQube Quality Gates configuration page. The 'Conditions' section shows a single condition: 'Bugs' is greater than 10. The 'Projects' section filters by 'With' logicclabs-repo and logicclabs.

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

Jenkins 'Deploy-to-Nexus' configuration page. The 'General' tab includes a description of 'deploy artifact' and several build triggers and notifications.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

52

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Nexus' job. The 'Office 365 Connector' tab is selected. Under 'Notification webhooks', there is an 'Add Webhook' button. A list of checkboxes includes: 'Permission to Copy Artifact', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', 'Execute concurrent builds if necessary', and 'Restrict where this project can be run'. An 'Advanced...' button is located at the bottom right. On the left, a sidebar lists various Jenkins features like Pipeline, Git, and Artifacts.

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Nexus' job. The 'Source Code Management' tab is selected. It shows the following configuration: 'None' is selected for the source code management provider. Under 'Build Triggers', the 'Build when a change is pushed to BitBucket' option is checked. At the bottom, there are 'Save' and 'Apply' buttons. The sidebar on the left is identical to the previous screenshot.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

53

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The 'Build' tab is selected. Under the 'Copy artifacts from another project' section, 'Project name' is set to 'CI-jenkins-Build1' and 'Which build' is set to 'Latest successful build'. The 'Artifacts to copy' field contains '**/*.war'. The 'Target directory' field is empty. The 'Parameter filters' field is also empty. There are checkboxes for 'Flatten directories', 'Optional', and 'Fingerprint Artifacts'. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The 'Nexus Details' tab is selected. Under the 'Nexus artifact uploader' section, 'Nexus Version' is set to 'NEXUS3', 'Protocol' is set to 'HTTP', and 'Nexus URL' is set to '172.31.8.182:8081'. The 'Credentials' dropdown shows 'admin/******** (nexus/login)'. The 'GroupId' field is 'QA', the 'Version' field is '\$BUILD_ID', and the 'Repository' dropdown is empty. At the bottom are 'Save' and 'Apply' buttons.

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

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

54

The screenshot shows the 'Deploy-to-Nexus' configuration page in Jenkins. The 'Nexus Details' tab is selected. The 'Version' field contains '\$BUILD_ID'. The 'Repository' field is set to 'logiclabs-release'. Under 'Artifacts', there is one entry for 'Artifact' with 'Artifactid' '\$BUILD_TIMESTAMP', 'Type' 'war', and 'Classifier' empty. The 'File' field contains 'target/profile-v2.war'. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the 'Post-build Actions' section of the 'Deploy-to-Nexus' configuration. It includes an 'Add' button and a 'Build Triggers' section with 'Projects to build' set to 'Deploy-to-Staging-Ansible' and 'Trigger when build is' set to 'Stable'. There is also a 'Trigger build without parameters' checkbox. Below that is a 'Predefined parameters' section with 'Parameters' TIME=\$BUILD_TIMESTAMP ID=\$BUILD_ID. At the bottom are 'Save', 'Apply', and 'Advanced...' buttons.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

55

The screenshot shows the Jenkins configuration interface for a job named 'Deploy-to-Nexus'. The 'Post-build Actions' section is active. It contains two main sections: 'Trigger parameterized build on other projects' and 'Predefined parameters'. Under 'Trigger parameterized build on other projects', there is a 'Build Triggers' section with a dropdown set to 'Stable' and a checkbox for 'Trigger build without parameters'. Under 'Predefined parameters', there is a 'Parameters' section with the entries 'TIME=\$BUILD_TIMESTAMP' and 'ID=\$BUILD_ID'. At the bottom are 'Save' and 'Apply' buttons.

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.

The screenshot shows the 'Post-build Actions' configuration page in Jenkins. Under the 'Build Triggers' section, there is a 'Trigger parameterized build on other projects' action. It specifies 'Deploy-to-Prod-Ansible' as the project to build and 'Stable' as the condition. A checkbox for 'Trigger build without parameters' is unchecked. Below this, under 'Predefined parameters', there is a 'Parameters' section containing 'TIME=\$BUILD_TIMESTAMP' and 'ID=\$BUILD_ID'. A checkbox for 'Use TextParameterValue if property contains newline' is also present. At the bottom right of the configuration area, there is a 'Add trigger...' button.

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,

The screenshot shows the Jenkins interface for build #54 of the 'Software-Testing' project. The left sidebar has 'Console Output' selected. The main content area is titled 'Console Output' with a green checkmark icon. It displays the command-line output of the build process, which includes several upstream project triggers and a series of git commands for cloning and checking out code from GitHub. The output ends with a command to run Maven tests.

```
Started by upstream project "Deploy-to-Staging-Ansible" build number 44
originally caused by:
Started by upstream project "Deploy-to-Nexus" build number 54
originally caused by:
Started by upstream project "Sonarscanner-CodeAnalysis" build number 56
originally caused by:
Started by upstream project "Code_Analysis" build number 58
originally caused by:
Started by upstream project "Integration" build number 53
originally caused by:
Started by upstream project "Test" build number 57
originally caused by:
Started by upstream project "CI-jenkins-Build1" build number 60
originally caused by:
Started by user admin
Running as SYSTEM
Building remotely on Logiclabs-softwareTest-Node in workspace c:\jenkins\workspace\Software-Testing
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir c:\\jenkins\\workspace\\Software-Testing\\.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/gnsharma530/Logiclabs-cidemo.git # timeout=10
Fetching upstream changes from https://github.com/gnsharma530/Logiclabs-cidemo.git
> git --version # timeout=10
> git --version # 'git version 2.35.1.windows.2'
> git fetch --tags --progress -- https://github.com/gnsharma530/Logiclabs-cidemo.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse 'refs/remotes/origin/seleniumAutoScripts{commit}' # timeout=10
Checking out Revision a9e01631d70e3933ad3a222e87f7d41ae9d64776 (refs/remotes/origin/seleniumAutoScripts)
> git config core.sparsecheckout # timeout=10
> git checkout -f a9e01631d70e3933ad3a222e87f7d41ae9d64776 # timeout=10
Commit message: "latest chrome driver"
> git rev-list --no-walk a9e01631d70e3933ad3a222e87f7d41ae9d64776 # timeout=10
[Software-Testing] $ cmd.exe /C "mvn clean test -Dsurefire.suiteXmlFiles=testng.xml -Durl=http://172.31.9.137:8080/login -Dusr=admin vp -Dpass=admin vp -
```

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.

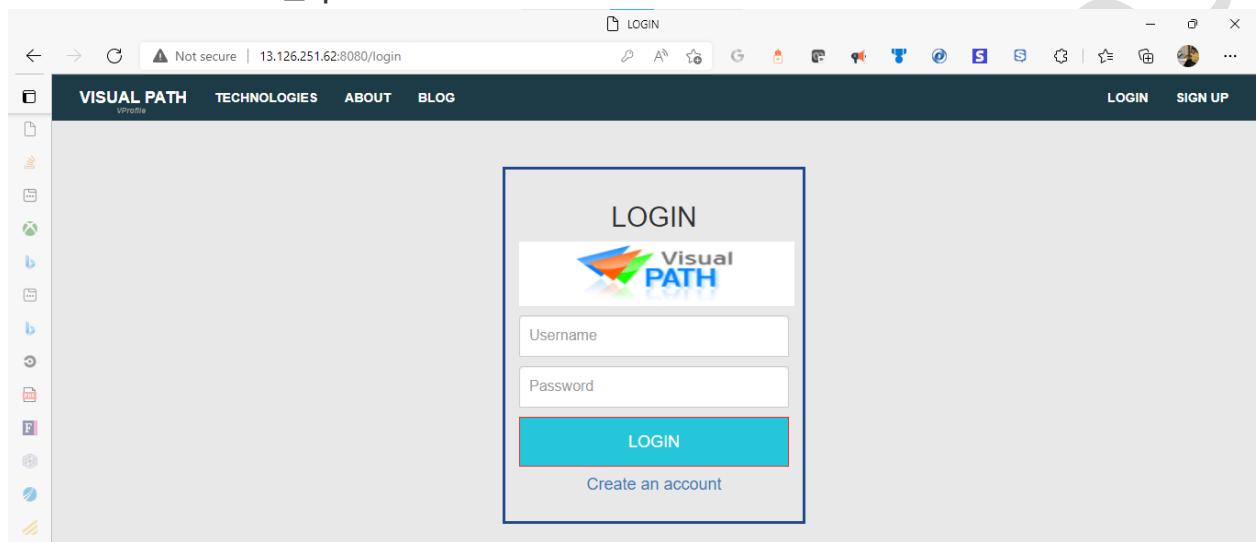
CI jenkins Build → UnitTest→ Integration Test→ CodeAnalysis → SonarScanner

Code Analysis → Deploy to Nexus → Deploy-to-staging-Ansible → Software Test (Selenium)

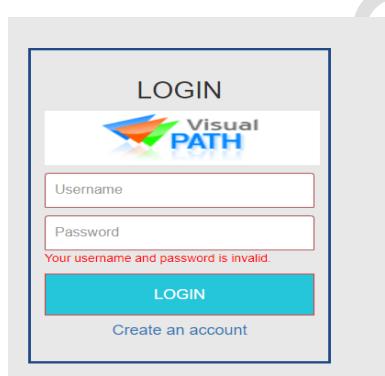
Verifying the Jobs status:

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.

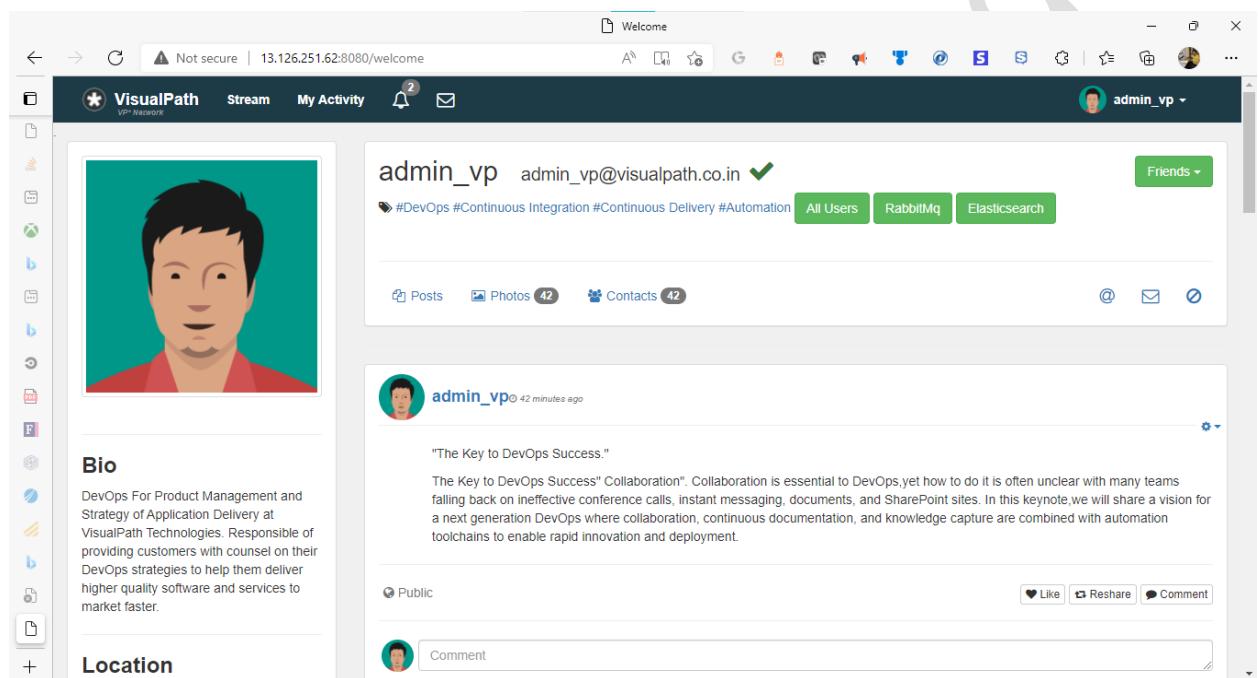
- Open the ec2 console and select web-server, copy the public ip of the server.
- Try logging in with the credentials given below:
- Username: admin_vp
- Password: admin_vp



*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.

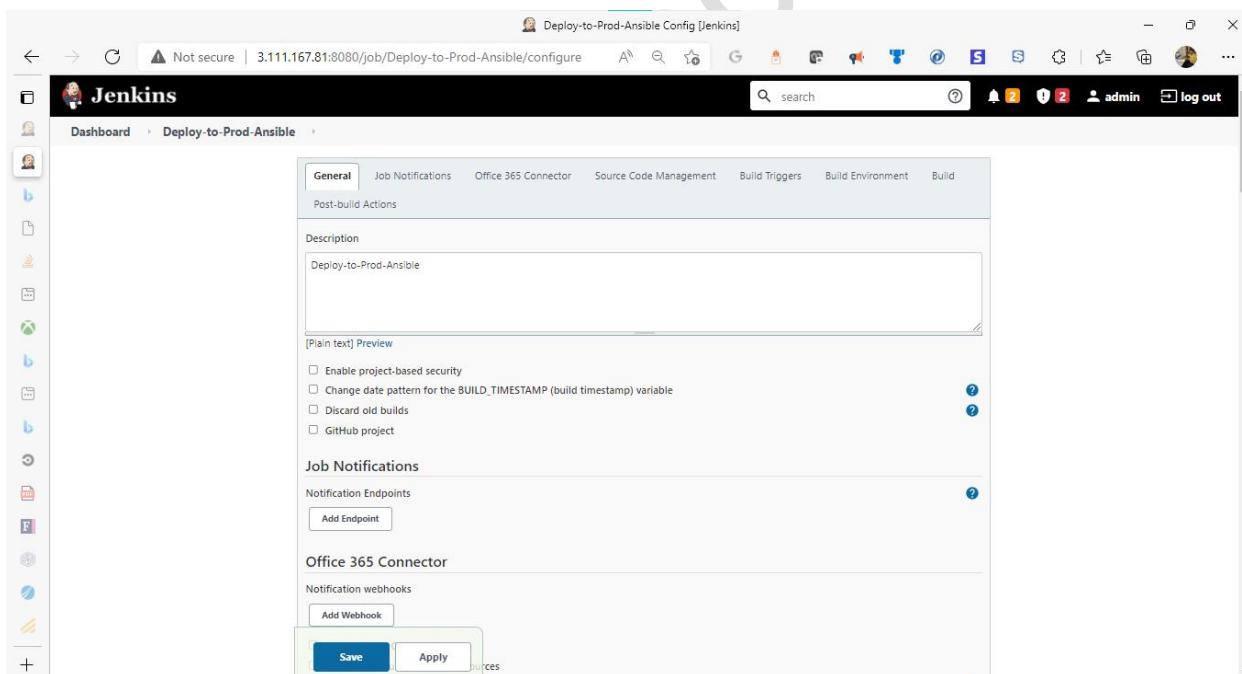
Step 9.1 Creation of Prod server

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.**

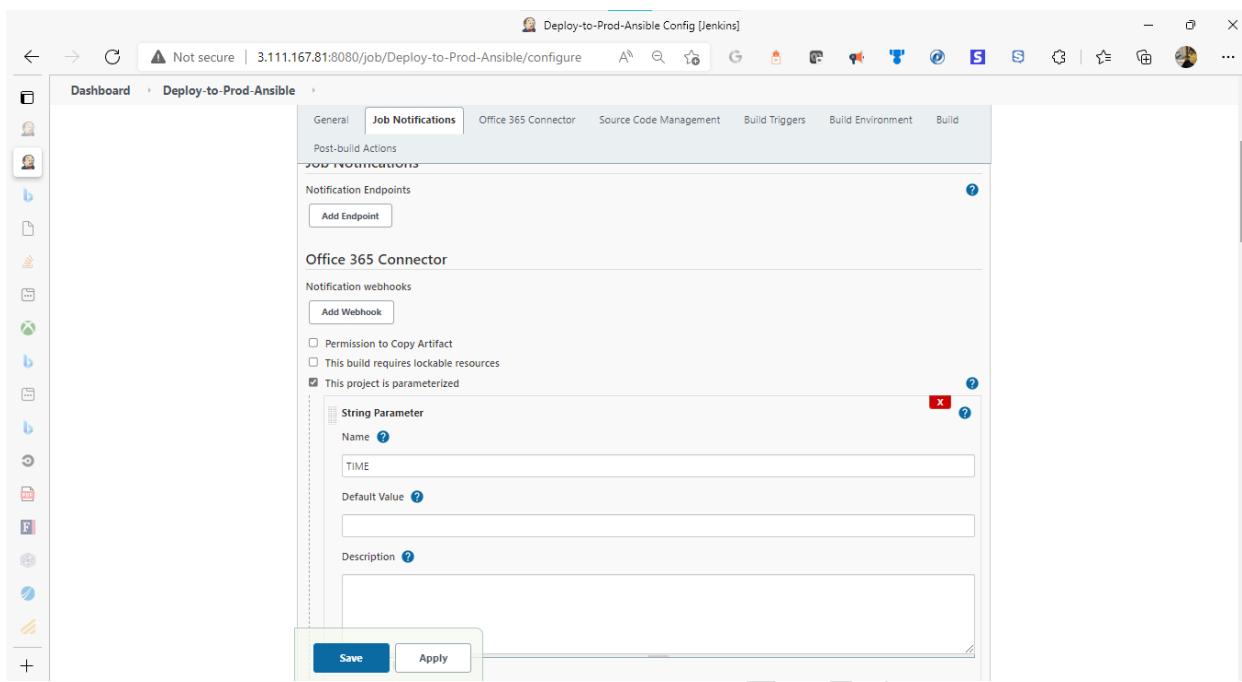
Step 9.2 Configuring Deploy-to-prod-Ansible job

- 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:

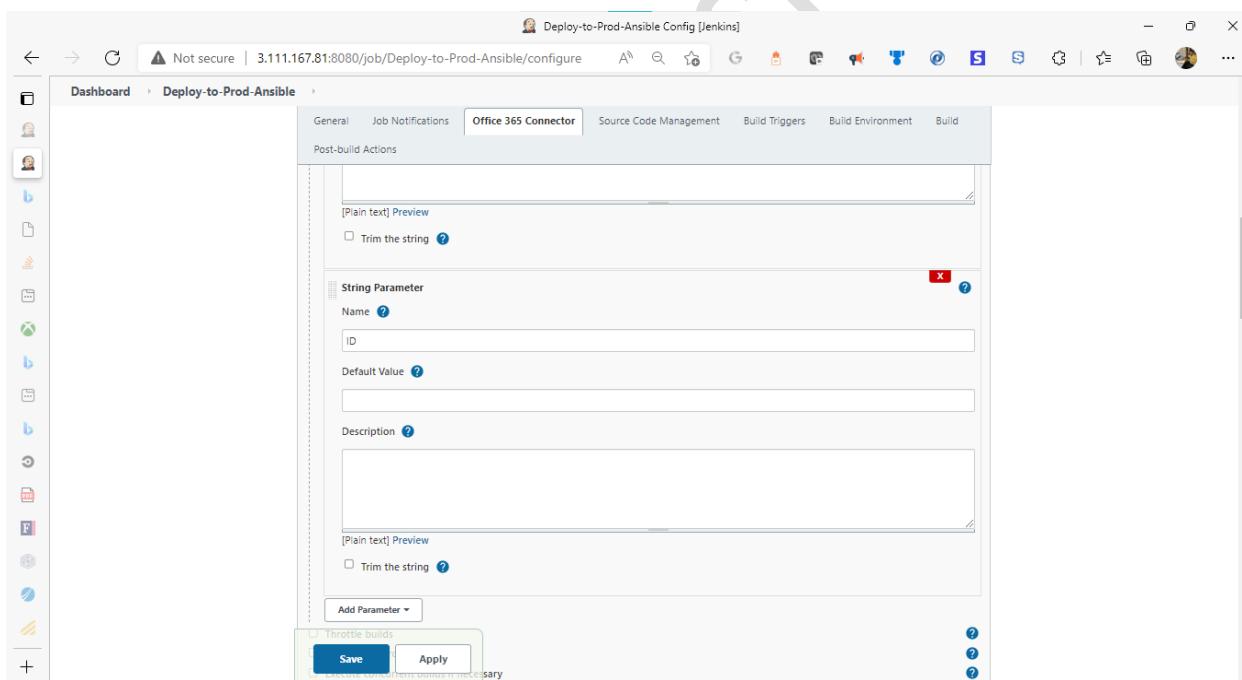


A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

61



The screenshot shows the Jenkins job configuration interface for the 'Deploy-to-Prod-Ansible' job. The 'Job Notifications' tab is selected. Under the 'Post-build Actions' section, there is a 'Notification Endpoints' panel with an 'Add Endpoint' button. Below it is the 'Office 365 Connector' panel, which includes sections for 'Notification webhooks' (with an 'Add Webhook' button) and configuration for this project (checkboxes for 'Permission to Copy Artifact', 'This build requires lockable resources', and 'This project is parameterized'). A 'String Parameter' is defined with the name 'TIME'. At the bottom are 'Save' and 'Apply' buttons.



The screenshot shows the Jenkins job configuration interface for the 'Deploy-to-Prod-Ansible' job. The 'Post-build Actions' tab is selected. It displays two 'String Parameter' definitions: one named 'TIME' and another named 'ID'. Each parameter has fields for 'Default Value' and 'Description'. Below each parameter is a '[Plain text] Preview' section with a 'Trim the string' checkbox. At the bottom are 'Add Parameter', 'Save', and 'Apply' buttons.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

62

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Prod-Ansible' job. The 'Post-build Actions' section is active. It contains a 'Plain text' preview field with the placeholder '[Plain text] Preview' and a checkbox for 'Trim the string'. Below this is a list of build-related checkboxes: 'Throttle builds', 'Disable this project', 'Execute concurrent builds if necessary', and 'Restrict where this project can be run' (which is checked). A 'Label Expression' field is set to 'master'. A note below states: 'Label master matches 1 node. Permissions or other restrictions provided by plugins may further reduce that list.' There is also an 'Advanced...' button. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Prod-Ansible' job. The 'Source Code Management' tab is active. It shows the 'Git' repository selected. Under 'Repositories', the 'Repository URL' is set to 'https://github.com/gnsharma530/Logilabs-cidemo.git'. A 'Save' and 'Apply' button are at the bottom.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

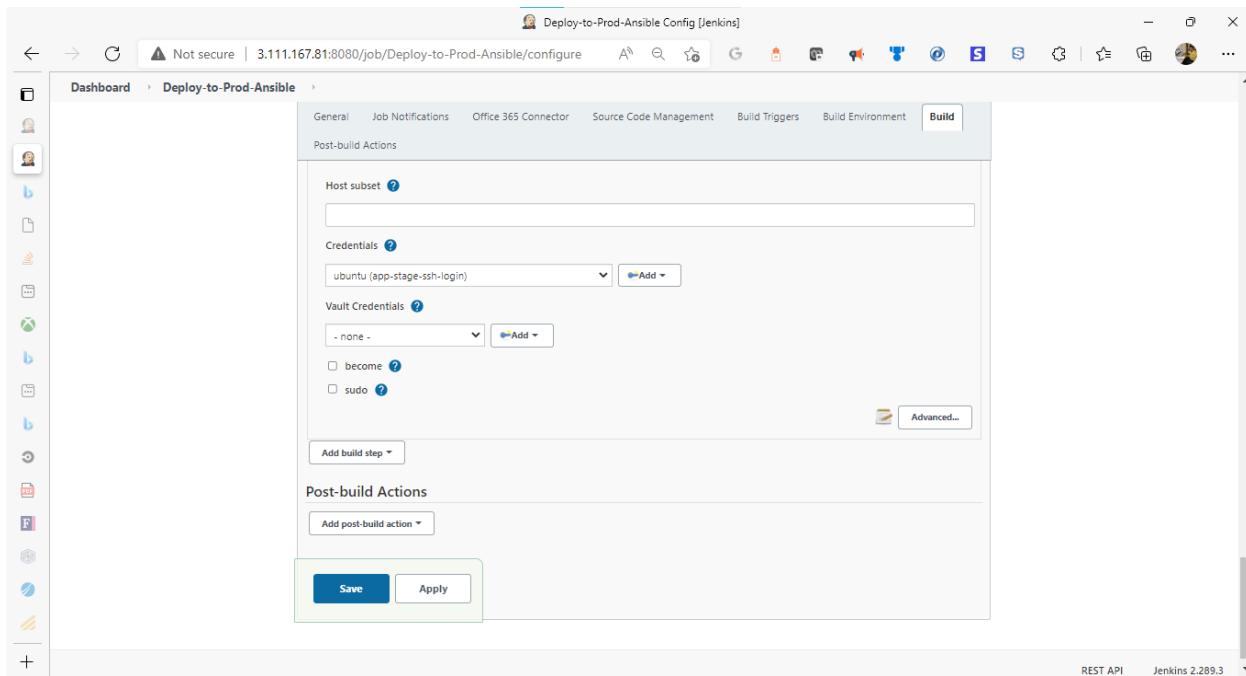
63

The screenshot shows the Jenkins configuration interface for the 'Deploy-to-Prod-Ansible' job. The 'Source Code Management' tab is selected, showing options for Repository browser (set to 'Auto') and Additional Behaviours (Mercurial). The 'Build Triggers' section includes checkboxes for triggering builds from scripts, after other projects, periodically, or via BitBucket/GitHub webhooks. The 'Build Environment' section includes checkboxes for workspace cleanup, secret text usage, build abort, timestamps, and Azure Key Vault binding. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the 'Build' configuration for the 'Deploy-to-Prod-Ansible' job. The 'Invoke Ansible Playbook' step is selected. Under 'Ansible installation', 'ansible2' is chosen. The 'Playbook path' is set to 'ansible/site.yml'. Under 'Inventory', 'Inline content' is selected, with 'Dynamic inventory' as an option. In the 'Content' section, the following YAML code is defined:

```
app01-prod ansible_host=172.31.39.0
[appsvrgrp]
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.

A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

65

The screenshot shows the AWS EC2 Management Console interface. The left sidebar has sections for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-options like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), and Images. The main content area displays a table of instances:

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 +

Below the table, a specific instance is selected: **Instance: i-006c996a84b900f7a (app01-prod)**. The details page shows the following information:

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
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 Running	Public IPv4 DNS 13.127.177.48				

The screenshot shows a web browser window with the URL [Not secure | 13.127.177.48:8080/login](http://13.127.177.48:8080/login). The page title is "VISUAL PATH". The content area contains a login form with the following fields:

- Logo: Visual PATH
- Text: LOGIN
- Text input field: Username
- Text input field: Password
- Text button: LOGIN
- Text link: Create an account

microsoftcertificate.zip ^

Show all X

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.

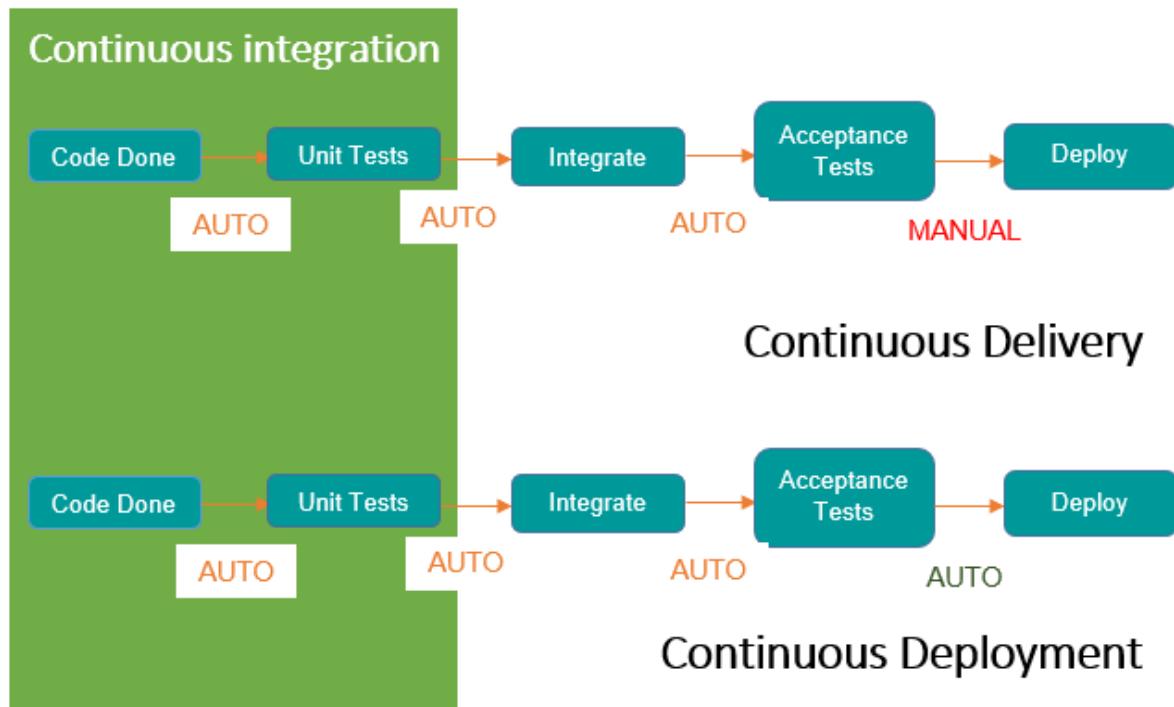
In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches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



LogicV



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.

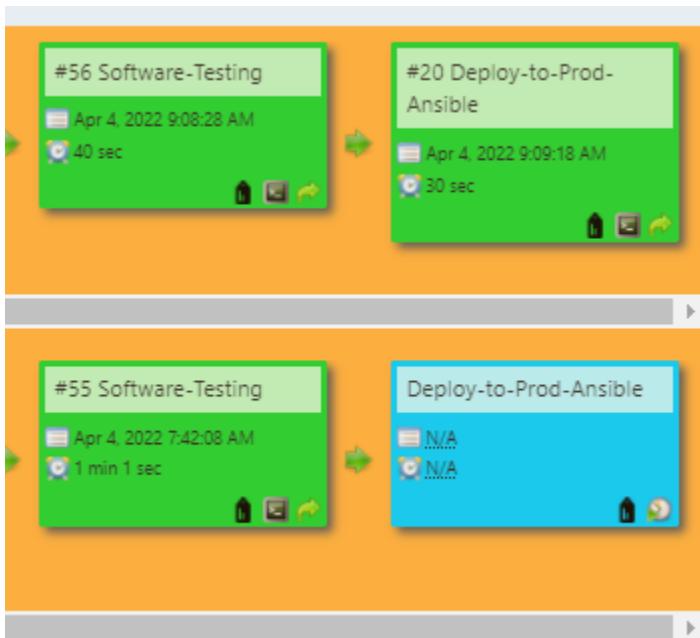
A Project Demonstration on Continuous Deployment using Jenkins, Ansible and Nexus.

68

The screenshot shows the Jenkins configuration interface for the 'Software-Testing' job. The 'Post-build Actions' tab is selected. Under 'Post-build Actions', there is a section for 'Build other projects (manual step)' with a dropdown menu labeled 'Deploy-to-Prod-Ansible'. Below this, there is a 'Predefined parameters' section with a text input field containing 'TIME=\$BUILD_TIMESTAMP' and 'ID=\$BUILD_ID'. At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

* Change your prod server name and IP address.

The screenshot shows the Jenkins dashboard for the 'logilabs-Continuous-Delivery' project. The main area is titled 'Build Pipeline' and displays a series of green cards representing build steps. The steps are arranged in two rows. The top row includes: 'Pipeline #63 Integration' (status: green, last run: Apr 4, 2022 9:06:03 AM), '#61 Code_Analysis' (status: green, last run: Apr 4, 2022 9:08:28 AM, duration: 15 sec), '#59 Sonarscanner-CodeAnalysis' (status: green, last run: Apr 4, 2022 9:08:40 AM, duration: 44 sec), '#57 Deploy-to-Nexus' (status: green, last run: Apr 4, 2022 9:07:33 AM, duration: 13 sec), '#47 Deploy-to-Staging-Ansible' (status: green, last run: Apr 4, 2022 9:07:53 AM, duration: 23 sec), '#56 Software-Testing' (status: green, last run: Apr 4, 2022 9:08:28 AM, duration: 40 sec), and '#20 Deploy-to-Prod-Ansible' (status: green, last run: Apr 4, 2022 9:09:18 AM, duration: 20 sec). The bottom row includes: 'Pipeline #62 Integration' (status: grey, last run: N/A), '#60 Code_Analysis' (status: green, last run: Apr 4, 2022 9:48:38 AM, duration: 0 sec), '#58 Sonarscanner-CodeAnalysis' (status: green, last run: Apr 4, 2022 9:48:59 AM, duration: 57 sec), '#56 Deploy-to-Nexus' (status: green, last run: Apr 4, 2022 9:50:03 AM, duration: 48 sec), '#46 Deploy-to-Staging-Ansible' (status: green, last run: Apr 4, 2022 9:50:13 AM, duration: 28 sec), '#55 Software-Testing' (status: green, last run: Apr 4, 2022 7:42:08 AM, duration: 1 min 1 sec), and 'Deploy-to-Prod-Ansible' (status: blue, last run: N/A). Above the pipeline cards, there are buttons for 'Run', 'History', 'Configure', 'Add Step', 'Delete', and 'Manage'.



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

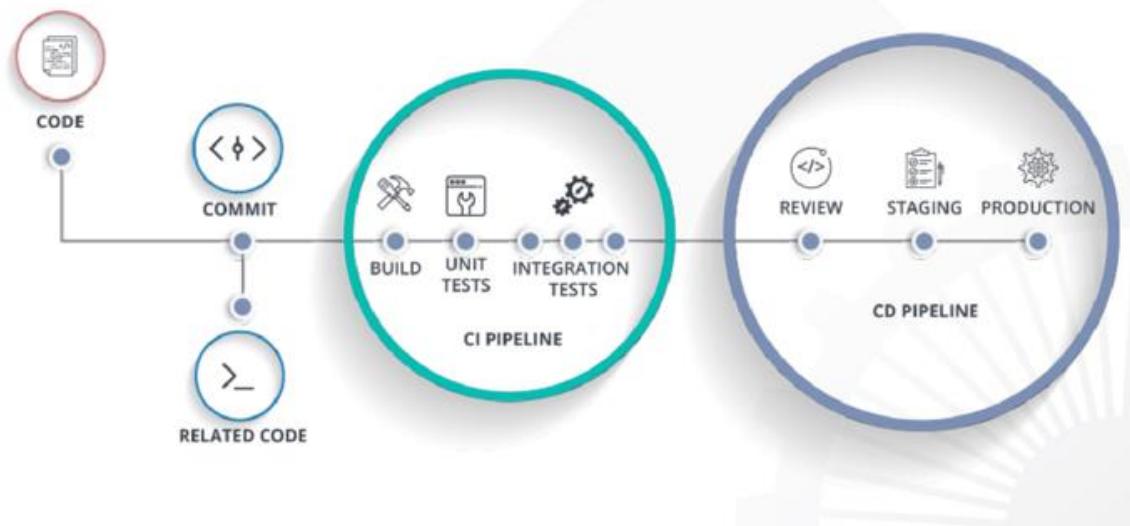
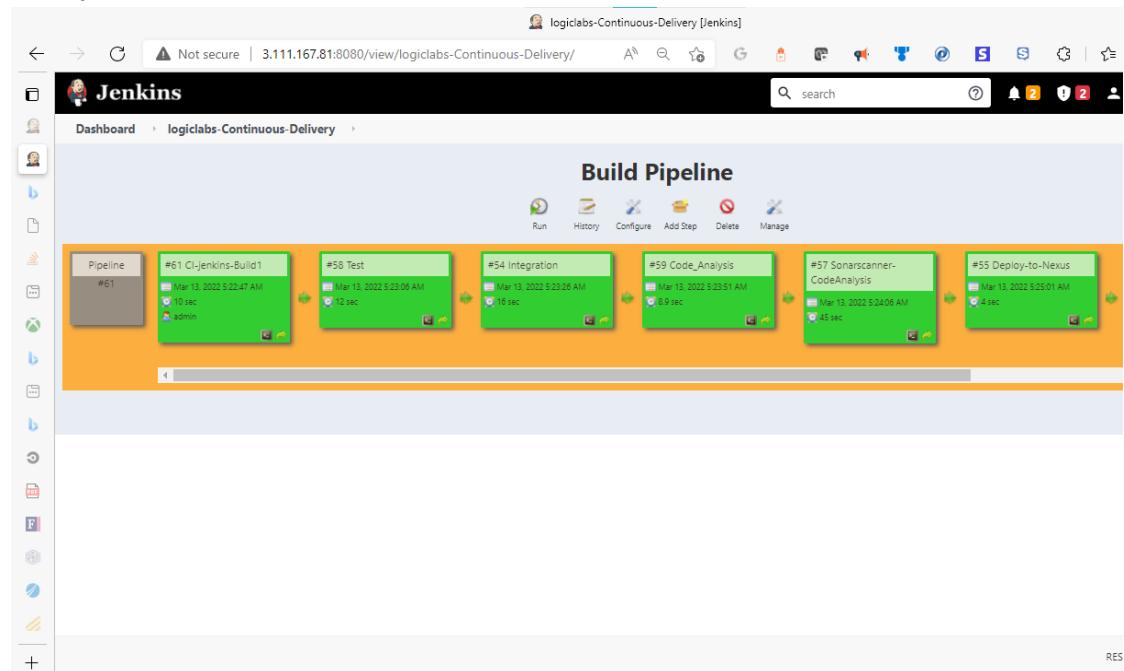
Manual procedure:

The screenshot shows the Jenkins dashboard with the 'Deploy-to-Prod-Ansible' project listed in the 'All' section. The project status is green, indicating success. A context menu is open over the project row, showing options: Changes, Workspace, Build with Parameters, Configure, Delete Project, and Rename. The 'Configure' option is highlighted. Other projects listed include CI-jenkins-Build1, Code_Analysis, continuous delivery test, Deploy-to-Nexus, and Test.

S	W	Name	Last Success	Last Failure	Last Duration	# Issues
		CI-jenkins-Build1	59 min - #62	1 mo 1 day - #50	13 sec	-
		Code_Analysis	58 min - #60	1 mo 2 days - #39	10 sec	706
		continuous delivery test	N/A	N/A	N/A	-
		Deploy-to-Nexus	57 min - #56	N/A	4.6 sec	-
		Deploy-to-Prod-Ansible	22 days - #19	1 mo 1 day - #11	21 sec	-
			56 min - #46	22 days - #45	26 sec	-
			58 min - #55	N/A	18 sec	-
			4 min 54 sec - #55	22 days - #49	1 min 1 sec	-
			58 min - #58	1 mo 1 day - #47	57 sec	-
		Test	59 min - #59	N/A	13 sec	-

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.

LogicLabs Technologies