Continuous Integration and Continuous deployment using Ansible, Git and Jenkins

Presented by Sadia Zaman Nishat
Id : 00 - 30087

Contents

Introduction:

This documentation contains CI/CD automation using ansible, git and jenkins. Continuous Integration (CI) and Continuous Deployment (CD) are software development practices that involve automating the process of building, testing, and deploying applications. Ansible, Git, and Jenkins are commonly used tools in setting up CI/CD pipelines. Ansible is an open-source automation tool that allows to define and manage infrastructure and application deployments in a declarative manner.Git is a distributed version control system that enables collaboration and version management of source code.Jenkins is an open-source automation server that supports CI/CD workflows.

Objectives:

The objective of the project is to build an automated CI/CD pipeline using Jenkins ,git and Ansible for a web application, ensuring efficient and reliable software delivery. The key steps are to achieve an objectives that to set up version control, then configure jenkins , then build process and then implement ci/cd pipeline. By achieving these objectives, we will have a fully automated CI/CD pipeline using Jenkins and Ansible, allowing for efficient and reliable software delivery, continuous integration, and continuous deployment of our web application.

Installing Cantos:

1. Download the Cantos ISO:
   - Visit the official Cantos website (https://www.cantos.org/) and navigate to the download section.
   - Choose the appropriate CentOS version and architecture (e.g., CentOS 7 or CentOS 8).
   - Download the ISO file for the desired version.

2. Install Oracle VM VirtualBox:
   - Download and install Oracle VM VirtualBox from the official website (https://www.virtualbox.org/).
   - Follow the installation wizard to complete the installation process.

3. Create a new virtual machine in VirtualBox:
   - Open Oracle VM VirtualBox Manager.
   - Click on "New" to create a new virtual machine.
   - Give your virtual machine a name (e.g., CentOS) and select the appropriate Type and Version (e.g., Linux and CentOS 7 or CentOS 8).
   - Allocate memory (RAM) for the virtual machine. Choose an amount based on your system resources.
   - Create a virtual hard disk or use an existing one. Allocate storage space based on your requirements.

4. Configure the virtual machine settings:
   - Select the virtual machine you just created and click on "Settings."
   - In the settings window, configure options such as display, storage, network, and other advanced settings as per your needs.

5. Mount the CentOS ISO:
   - In the virtual machine settings window, go to the "Storage" section.
   - Under the "Controller: IDE" tab, click on the "Empty" CD/DVD icon.
   - On the right side, click on the small disk icon and select "Choose a disk file."
   - Locate and select the CentOS ISO file you downloaded in step 1.
   - Click "OK" to save the settings.

6. Install CentOS:
   - Start the virtual machine by clicking on the "Start" button in the Oracle VM VirtualBox Manager.
   - The CentOS installation process should begin. Follow the on-screen instructions to complete the installation.
   - Configure the desired settings during the installation, such as language, keyboard layout, disk partitioning, and user account creation.

7. Complete the installation:
   - After the installation is complete, the virtual machine will restart.
   - Log in to CentOS using the credentials you created during the installation process.

Master and Client node setup for CI/CD project:

Requirements:

Need to install 3 centos…

- one is for master on virtualbox
- other two as client node(webserver and dbserver) on virtualbox.

Master-Node Setup:

1. Configure IP address

- nmtui

2. Install basic necessary packages

- yum install vim curl wget open-vm-tools -y
- Disable selinux security policy
- vim /etc/selinux/config

3. Disable firewalld

- systemctl stop firewalld
- systemctl disable firewalld

4. Setup node hostname

- vim /etc/hostname
  ansible-master.localdomain

5. Setup local dns name resolution

- vim /etc/hosts
  192.168.20.52 ansible-master.localdomain ansible-master
  192.168.20.33 webserver.localdomain webserver
  192.168.20.42 dbserver.localdomain dbserver

6. Setup key base authorized base authentication.

- ssh-keygen

Client1-Node for webserver setup:

1. Setup hostname

- Vi /etc/hostname/
  webserver.localdomain

2. Setup local dns name resoulation

- vim /etc/hosts
  192.168.20.52 ansible-master.localdomain ansible-master
  192.168.20.33 webserver.localdomain webserver
  192.168.20.42 dbserver.localdomain dbserver

3. Update sudoers file for passwordless authentication

- vi /etc/sudoers.d/ansible
  add this line: ansible ALL=(ALL) NOPASSWD: ALL

4. Create .ssh directory at ansible user home
- mkdir .ssh
- cd .ssh
- vim authorized_keys #(copy id_rsa.pub from master-node "ansible" user to all client-node as authorized_keys)

5. Change permission for .ssh directory
- chmod 700 .ssh
- chmod 600 authorized_keys

Client2-Node for dbserver setup:

6. Setup hostname

- Vi /etc/hostname/
  dbserver.localdomain

7. Setup local dns name resoulation

- vim /etc/hosts
  192.168.20.52 ansible-master.localdomain ansible-master
  192.168.20.33 webserver.localdomain webserver
  192.168.20.42 dbserver.localdomain dbserver

8. Update sudoers file for passwordless authentication

- vi /etc/sudoers.d/ansible
  add this line: ansible ALL=(ALL) NOPASSWD: ALL

9. Create .ssh directory at ansible user home
- mkdir .ssh
- cd .ssh
- vim authorized_keys #(copy id_rsa.pub from master-node "ansible" user to all client-node as authorized_keys)

10. Change permission for .ssh directory
- chmod 700 .ssh
- chmod 600 authorized_keys

Check authentication from Master

- ansible all -m ping

To install Ansible on the master VM, we can follow these steps:

Ansible master node:

1.Install Ansible on master node using epel repository.

- yum install epel-release -y
- yum install ansible -y

2. Configure ansible.cfg file
vi /etc/ansible/ansible.cfg

3.configure inventory file
- vi /etc/ansible/hosts

To install jenkins on the master VM, we can follow these steps:

1. Update the system

- sudo yum update
- sudo yum upgrade

2. Install Java Development Kit(JDK):
Jenkins required Java to run. Install OpenJDK using the following command:

- sudo yum install openjdk-11-jdk

3. To install the latest stable version of Jenkins , we have to add the official Jenkins repository to the system. Execute the below commands to add the key and repo.

- yum install wget -y
- sudo wget -O /etc/yum.repos.d/jenkins.repohttps://pkg.jenkins.io/redhatstable/jenkins.repo
- sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io-2023.key

3. Now install, start and enable jenkins by below commands

- yum install jenkins -y
- systemctl start jenkins
- systemctl enable jenkins

To create roles on the master VM, we can follow these steps:

Now we will create an ansible-playbook to install, start mandatory services in node. Firstly we need to create files (httpd_project.yml) and a directory (roles) in the same directory. Under the “roles” directory we created another directory which is called “web” and “db”. Under this “web” and “db” directory we created four directory (vars, tasks, templates, files). We created “main.yml” files in the vars and tasks directory. In the files of “db” directory we created two files “db-load-script.sql” and “my.cnf”.

- pwd
- cd /etc/ansible
- touch project.yml
- cd roles/
- ansible-galaxy init web --offline
- ansible-galaxy init db --offline

```
.
├── project.yml
└── roles
    ├── db
    │   ├── files
    │   │   ├── db-load-script.sql
    │   │   └── my.cnf
    │   ├── tasks
    │   │   └── main.yml
    │   ├── templates
    │   └── vars
    │       └── main.yml
    └── web
        ├── files
        ├── tasks
        │   └── main.yml
        ├── templates
        └── vars
            └── main.yml
```

We have mentioned the below lines in “roles/web/tasks/main.yml”.

```
- name: Installation Services
  yum:
  name:
      - libselinux-python
      - libsemanage-python
      - httpd
      - git
      - php
      - php-mysql
  state: installed
  tags: install
  Note: Installing mandatory services

- name: Start firewalld
  service: name=firewalld state=started enabled=yes
  tags: start firewalld
  Note: start firewall

- name: Insert firewalld rule for httpd
  firewalld: port={{ httpd_port }}/tcp permanent=true state=enabled immediate=yes
  tags: enable httpd port
  Note: enable httpd port in firewall

- name: insert firewalld rule for mysql
  firewalld: port={{ mysql_port }}/tcp permanent=true state=enabled immediate=yes
  tags: enable mysql port
  Note: enable mysql port in firewall

- name: Set index.php as the default page
  replace:
      path: /etc/httpd/conf/httpd.conf
      regexp: 'DirectoryIndex index.html'
      replace: '#DirectoryIndex index.html \nDirectoryIndex index.php'
  tags: rename html file
  Note: Rename html file to php in configuration file


- name: http service state
  service: name=httpd state=started enabled=yes
  tags: httpd start
  Note: Starting httpd service
```

```
- name: Copy the code from repository
  git: repo={{ repository }} dest=/var/www/html/ force=yes
  tags: clone
```
Note: Clone repository

```
- name: replace ip in index.php file
  command: sed -i 's/172.20.1.101/192.168.20.42/g' /var/www/html/index.php
  tags: replace IP
```
Note: Replace the IP

 # We added the below lines in "roles/web/vars/main.yml" file

```
---
httpd_port: 80
mysql_port: 3306
repository:  https://github.com/Nishat792/Ecommerce_Project.git
```
Note: We have mentioned the variable in this file

 # We have mentioned the below lines in "roles/db/tasks/main.yml".

```
- name: Installation Services
  yum:
  name:
      - libselinux-python
      - libsemanage-python
      - mariadb-server
      - MySQL-python
      - php-mysql
  state: installed
  tags: install

- name: Start firewalld
  service: name=firewalld state=started enabled=yes
  tags: start firewalld

- name: Insert firewalld rule for mysql
  firewalld: port={{ mysql_port }}/tcp permanent=true state=enabled immediate=yes
  tags: enable mysql port


- name: Restart firewalld
  service: name=firewalld state=reloaded enabled=yes
  tags: restarted firewalld
```

```
- name: Copy Mysql configuration file
  copy: src=files/my.cnf dest=/etc/my.cnf
  tags: mysql conf copy

- name: Start MariaDB Service
  service: name=mariadb state=started enabled=yes
  tags: start mariadb


- name: Create Application Database
  mysql_db: name={{ dbname }} state=present
  tags: create database

- name: Create Application DB User
  mysql_user: name={{ dbuser }} password={{ dbpassword }} priv=*.*:ALL
host='192.168.20.42' state=present
  tags: create user

- name: Move db-load-script to db host
  copy:
    src: files/db-load-script.sql
    dest: /tmp/db-load-script.sql
  tags: copy sql

- name: Load Inventory Data
  shell: mysql -f < /tmp/db-load-script.sql
  tags: run sql
```

We have mentioned the below lines in “roles/db/vars/main.yml”.

```
---
mysql_port: 3306
dbname: ecomdb
dbuser: ecomuser
dbpassword: ecompassword
```

Now in the “project.yml” file we have added the below lines:

```
---
- name: DB Service
  hosts: dbserver
  roles:
     - db
     - name: Web Service
  hosts: webserver
  roles:
     - web
```

Note: We have provided the host name and roles information in this files. So it will go to “web” and “db” roles and execute the tasks

We have mentioned the below lines in “roles/db/files/db-load-script.sql”

```
GRANT ALL PRIVILEGES ON *.* TO 'ecomuser'@'192.168.20.33' IDENTIFIED BY
'ecompassword' WITH GRANT OPTION;
FLUSH PRIVILEGES;
USE ecomdb;
CREATE TABLE products (id mediumint(8) unsigned NOT NULL auto_increment,Name
varchar(255) default NULL,Price varchar(255) default NULL, ImageUrl varchar(255) default
NULL,PRIMARY KEY (id)) AUTO_INCREMENT=1;
INSERT INTO products (Name,Price,ImageUrl) VALUES ("Bengal crafts","100","c-
1.png"),("Bengal crafts","200","c-2.png"),("Bengal crafts","300","c-3.png"),("Bengal
crafts","50","c-5.png"),("Bengal crafts","90","c6.png"),("Bengal crafts","20","c-
7.png"),("Bengal crafts","80","c-8.png"),("Bengal crafts","150","c-4.png");
```

We have mentioned the below lines in “roles/db/files/db-load-script.sql”

```
[mysqld]
Bind-address=0.0.0.0
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
symbolic-links=0
[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid
!includedir /etc/my.cnf.d
```

Now run ansible-playbook by below command

- ansible-playbook project.yml

To use Jenkins for automatic builds, you can follow these steps:

1. Set up source code repository using github.

2. Create a new Jenkins job:

   - Open the Jenkins web interface by accessing `http://<vm_ip_address>:8080` in a web browser.
   - Click on "New Item" to create a new Jenkins job.
   - Enter a name for your job and select the type of job to create Pipeline project.
   - Click on "OK" to proceed.

3. Configure the job.

4. Save the job configuration.

5. Test the build

6. Configure automatic builds:

- Depending on the build trigger you selected in the job configuration, Jenkins can automatically start builds when specific events occur. For example:
  - Periodic builds: Set the schedule for the job to run at regular intervals.
  - Webhook triggers: Configure your source code repository to send a webhook to Jenkins whenever changes are pushed.
  - SCM polling: Schedule Jenkins to check for changes in the repository at specific intervals.

7. Monitor build results:

By following these steps, you can leverage Jenkins to automatically build your projects based on the configured triggers, ensuring a streamlined and efficient build process.

Definition

Pipeline script

Script ?

```
pipeline {
    agent any

    stages {
        stage('git clone') {
            steps {
                git branch: 'main', url: 'https://github.com/Nishat792/Ecommerce_Project.git'
            }
        }
        stage('ansible playbook run') {
            steps {
                ansiblePlaybook become: true, credentialsId: '2b380ab8-94fc-4bea-be39-65529a2f1abe', disableHostKeyCheck
            }
        }
        stage('test') {
            steps {
                echo "Test"
            }
        }
        stage('deploy') {
            steps {
                echo "deploy successfully"
            }
        }
```

Save Apply

- [ ] GitHub hook trigger for GITScm polling ?
- [x] Poll SCM ?

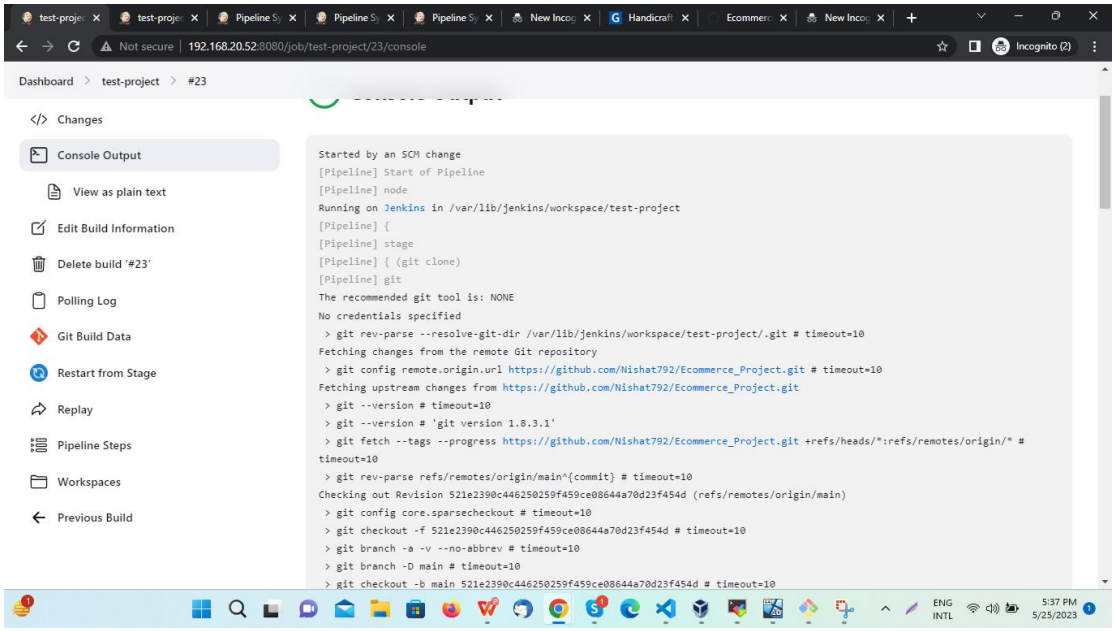
Schedule ?

```
H/2 * * * *
```

Would last have run at Thursday, May 25, 2023 at 11:29:17 PM Eastern Daylight Time; would next run at Thursday, May 25, 2023 at 11:29:17 PM Eastern Daylight Time.

- [ ] Ignore post-commit hooks ?

- [ ] Quiet period ?



Not secure | 192.168.20.52:8080/job/test-project/23/console

Dashboard > test-project > #23

```
Commit message: "image update"
 > git rev-list --no-walk 3a81f53884f5080fdd644eea87f3255023160fed # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (ansible playbook run)
[Pipeline] ansiblePlaybook
[test-project] $ sshpass ******** /usr/bin/ansible-playbook /etc/ansible/project.yml -i /etc/ansible/hosts -b --become-user
root -u root -k

PLAY [DB Service] *************************************************************

TASK [Gathering Facts] ********************************************************
ok: [192.168.20.42]

TASK [db : Installation Services] *********************************************
ok: [192.168.20.42]

TASK [db : Start firewalld] ***************************************************
ok: [192.168.20.42]

TASK [db : Insert firewalld rule for mysql] ***********************************
ok: [192.168.20.42]

TASK [db : Restart firewalld] *************************************************
changed: [192.168.20.42]
```

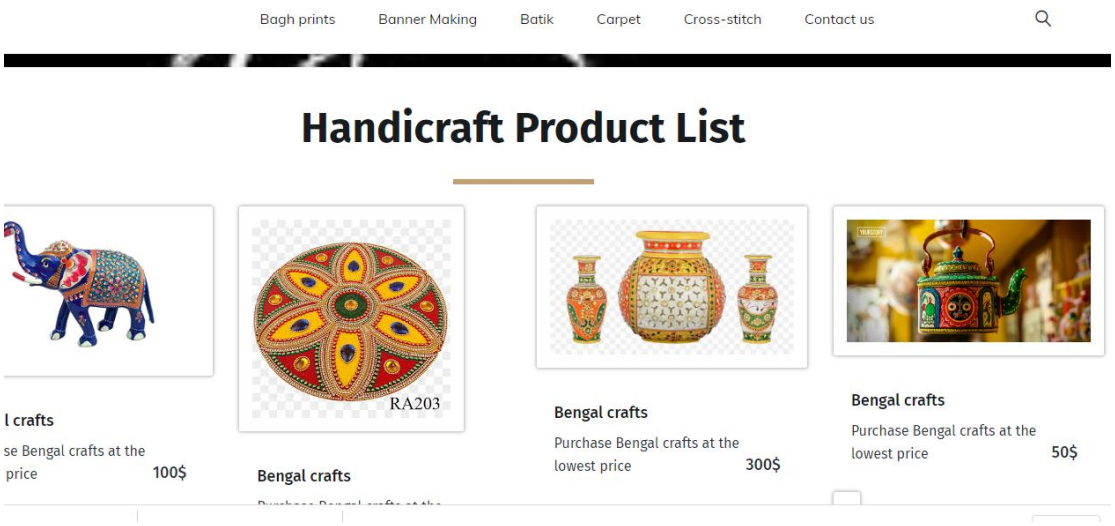
Not secure | 192.168.20.52:8080/job/test-project/23/console

Dashboard > test-project > #23

```
TASK [web : Copy the code from repository] ************************************
changed: [192.168.20.33]

TASK [web : replace ip in index.php file] *************************************
[WARNING]: Consider using the replace, lineinfile or template module rather
than running 'sed'.  If you need to use command because replace, lineinfile or
template is insufficient you can add 'warn: false' to this command task or set
'command_warnings=False' in ansible.cfg to get rid of this message.
changed: [192.168.20.33]

PLAY RECAP ********************************************************************
192.168.20.33              : ok=9    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.20.42              : ok=11   changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.387.3

Lastly, use webserver ip in the browser to see this web pages



This part is about my understanding about playbook:

```
name: Set index.php as the default page
  replace:
      path: /etc/httpd/conf/httpd.conf
      regexp: 'DirectoryIndex index.html'
      replace: '#DirectoryIndex index.html \nDirectoryIndex index.php'
  tags: rename html file
```

- The name parameter provides a descriptive name for the task.
- The ansible.builtin.replace module is used to perform the replacement operation.
- The path parameter specifies the path of the file to modify (/etc/httpd/conf/httpd.conf in this case).
- The regexp parameter defines the regular expression pattern to search for (DirectoryIndex index.html).
- The replace parameter specifies the replacement string (DirectoryIndex index.php).
- The backup parameter is set to yes to create a backup of the original file.
- The tags parameter assigns the rename html file tag to the task, allowing you to selectively run or skip tasks with specific tags.

```
- name: replace ip in index.php file
command: sed -i 's/172.20.1.101/192.168.20.14/g' /var/www/html/index.php
tags: replace IP
```

- The name parameter provides a descriptive name for the task.
- The command module is used to execute the sed command.
- The sed command performs a search and replace operation in the index.php file.
- The -i option is used to edit the file in-place.
- The 's/172.20.1.101/192.168.20.14/g' argument is the search and replace pattern. It replaces all occurrences of 172.20.1.101 with 192.168.20.14 in the file.
- The /var/www/html/index.php argument specifies the path to the index.php file.
- The tags parameter assigns the replace IP tag to the task, allowing you to selectively run or skip tasks with specific tags.

name: Copy Mysql configuration file
copy: src=files/my.cnf dest=/etc/my.cnf
tags: mysql conf copy.

- The name parameter provides a descriptive name for the task.
- The copy module is used to copy files and directories.
- The src parameter specifies the source file path relative to the playbook directory. Make sure to place the my.cnf file in the files/ directory relative to your playbook.
- The dest parameter specifies the destination file path on the target host, which is /etc/my.cnf in this case.
- The tags parameter assigns the mysql conf copy tag to the task, allowing you to selectively run or skip tasks with specific tags.

name: Load Inventory Data
shell: mysql -f < /tmp/db-load-script.sql
tags: run sql

- the shell module is used to execute shell commands on the target host.
- The mysql -f < /tmp/db-load-script.sql command is executed to load the inventory data from the db-load-script.sql file. Adjust the command and file path as per your specific setup.
- The -f option is used with the mysql command to force execution of the SQL file without prompting for confirmation.
- The < symbol is used to redirect the contents of the db-load-script.sql file as input to the mysql command.
- The tags parameter assigns the run sql tag to the task, allowing you to selectively run or skip tasks with specific tags.