
# Jenkins



## Introduction to Jenkins

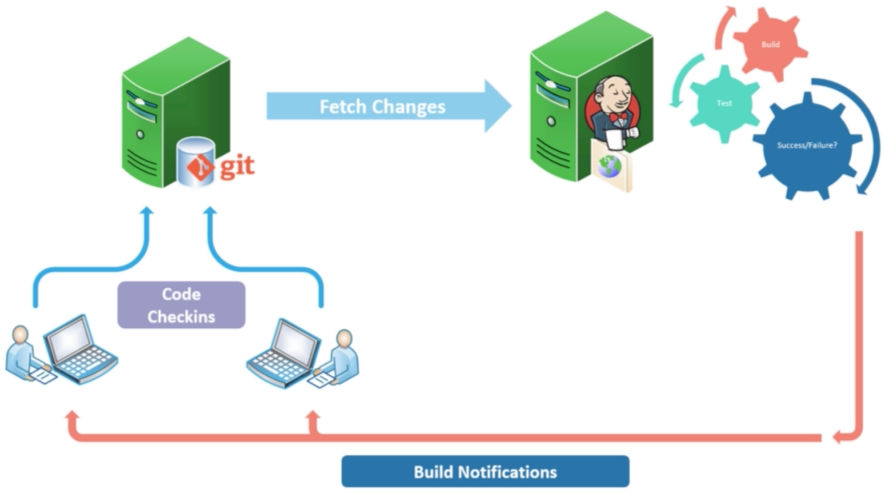
Before get into Jenkins let's try to understand the famous terminolgy CI/CD

> What is Continuous Integration, Delivery and Deployment

- **Continuous Integration (CI)** is the practice which helps to merge all developer working copies to the central shared location.
- **Continuous Delivery (CD)** is the approach which helps teams to produce in short cycles, Ensures that software developed is more reliable. The main objective of CD is building, testing and releasing software faster and more frequent.
- **Continuous Deployment (CD)** Once the tests have been validated on the dev environment, it must be put into production. Continuous deployment, therefore, consists of automating deployment actions that were previously performed manually.

> Let's checkout how this works in Practical.

- As part of software development one or more developers used to develop the code regularly and push it to common source code repository (GIT)
- Then CI server (Jenkins) will fetches this code from GIT and then run the Build process which produces the bundle of softwares.
- These artifacts are stored in and central repository which is famously know Artifactory.
- Then CI server might also run several autoamted tests (Unit & Integration tests) on artifacts which could valuate the build.
- CI server will also might send the notification to developers about the sucess and failure of build.
- This cycle will continue to happen in an automated fashion once after every single commit done by developers in Git.




> Why this approach becomes so popular these days ?

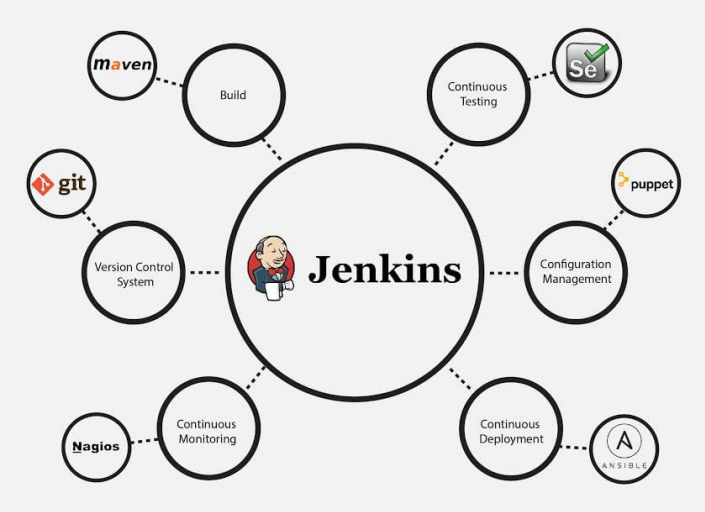
- Reduced risk on doing things manually
- Increased confidence about your product
- Better quality code
- Ready to ship code
- Time to market
- Reduced cost

> Tools available to perform CI/CD operations in market

- Bamboo
- Travis CI
- Cruise Control
- Jenkins

# What is Jenkins and why to use it?

- Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes.
- Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.
- It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.
- Jenkins achieves  Integration with the help of plugins. Plugins allow the integration of Various DevOps tools.
- If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven, Ansible & Terraform.

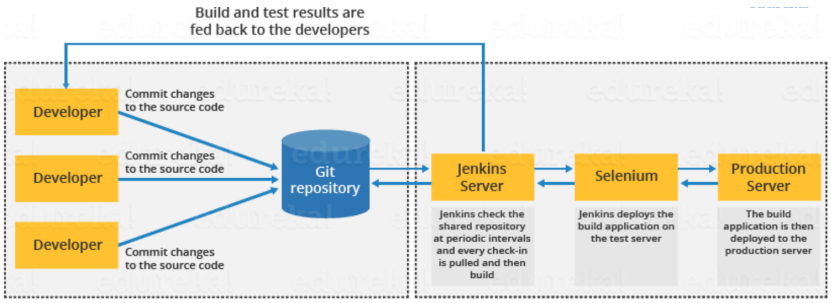



## Advantages of Jenkins include

- It is an open-source tool with great community support.
- It is easy to install.
- It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share it with the community.
- It is free of cost.
- It is built with Java and hence, it is portable to all the major platforms.

## Let's see how Jenkins will help to do achive the tasks seemlessly.

- Let me  first explain to you a generic flow diagram of Continuous Integration with Jenkins so that it becomes self-explanatory.




- First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If built is successful, then Jenkins deploys the built in the test server.
- After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
- It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

# Jenkins Install & Architecture

## Centos

- Install Java

```
yum install java-1.8.0-openjdk-devel
```

- Add Jenkins Software Repository

```
wget –O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo
```

The system will reach out to the Jenkins server and download the location of the repository to your system. It should display /etc/yum.repos.d/jenkins.repo saved.

- Next, import the GPG key to ensure your software is legitimate.

```
rpm ––import https://pkg.jenkins.io/redhat/jenkins.io.key
```

- Install Jenkins

```
yum install jenkins
```

- Start Jenkins service

```
systemctl start jenkins
```

## Ubuntu

- Update

```
apt update
```

- Install Java

```
apt install java
```

- Install Jenkins
- First, add the repository key to the system

```
wget -q -O - http://pkg.jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
```

- When the key is added, the system will return OK.
- Next, append the Debian package repository address to the server's sources.list.

```
sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

- Install Jenkins

```
apt install jenkins
```

- Start Jenkins

```
Starting Jenkins
```

# Jenkins dashboard overview

Once after installation lets take and overview of below things in Jenkins dashboard

- Plugins
- Login
- User details
- Jobs
- Credentials

# Create standard job which can integrate Terraform & deploy HTML file

## Objective

Create standard jenkins job to deploy EC2 instance using Terraform

## Prerequsites

- One container with Jenkins & Terraform installed.
- Better to have shared volume.
- AWS cli installed and AWS credentials configured.

## Actions

- Create Terraform configuration file to create simple EC2 instance.
- Makesure security group is attached
- Makesure Pem key is attached
- Use provisioner to install apache and copy HTML file.
- Create standard Jenkins job and do this from Jenkins portal.
- After successfull job execution - should be able to access the html file from url "hostname:80"

## Tasks to be done in machine

- Since all the tasks will be done from jenkins user let's ensure jenkins user has full previlage.
- To check jenkins user

```
id jenkins
```

- Ensure Jenkins user has shell login access.
- Ensure line is similar to below mentioned line in the file.

```
vi /etc/passwd
jenkins:x:997:994:Jenkins Automation Server:/var/lib/jenkins:/bin/bash
```

- Adding user into sudo list for full permission.

- First install sudoers package (yum install sudo)
- Check if sudoers file is available
- Uncomment line (%wheel ALL=(ALL) NOPASSWD: ALL)
- Comment line (%wheel ALL=(ALL) ALL)
- Add jenkins user in wheel group.
- Configure aws credentails in jenkins home directory

```
yum install sudo
cat /etc/sudoers
%wheel ALL=(ALL)       NOPASSWD: ALL
%wheel ALL=(ALL)       ALL
usermod -aG wheel jenkins
aws configure
```

> Actions in Jenkins dashboard

- Create simple jenkins freestyle job and add terraform path in build area.

```
cd /data/devops/Jenkins-workspace/Example_0/ec2_creation && /usr/local/bin/terraform apply -auto-approve
```

# Create standard job which can integrate Terraform & deploy HTML file [Import choices]

> Actions in Jenkins dashboard

- Create simple jenkins freestyle job and add terraform path in build area.

```
if [ "$Action" == "Create" ]; then
echo "Creating instance"
cd /data/devops/Jenkins-workspace/Example_0/ec2_creation && /usr/local/bin/terraform apply -auto-approve
else
echo "Deleting instance"
cd /data/devops/Jenkins-workspace/Example_0/ec2_creation && /usr/local/bin/terraform destroy -auto-approve
fi
```

# Create standard job which can integrate Terraform & Ansible

> Action in Jenkins dashboad

- Create simple jenkins freestyle job and add terraform path in build area.

```
# Preparing Ansible investory file
echo "Preparing ansible inventory file"
sudo rm -f /etc/ansible/hosts
sudo touch /etc/ansible/hosts
sudo chmod 666 /etc/ansible/hosts

if [ "$Action" == "Create" ]; then
# Executing Terraform configuration file
echo "Creating instance"
cd /data/devops/Jenkins-workspace/Example-2/ec2_creation && /usr/local/bin/terraform apply -auto-approve && /usr/local/bin/terraform o
sleep 30
# Executing Ansible
echo "Executing playbook"
sudo /usr/local/bin/ansible-playbook /data/devops/Jenkins-workspace/Example-2/playbook/apache.yml --key-file "/data/devops/Jenkins-wor
else
```

```
echo "Deleting instance"
cd /data/devops/Jenkins-workspace/Example-2/ec2_creation && /usr/local/bin/terraform destroy -auto-approve
fi
```

# Create groovy pipeline which can integrate Terraform & Ansible

> Action in Jenkins dashboad

- Install all blue ocean plugins
- Create pipeline job add all the codes in build area which executes Example_3

```
pipeline {
  agent any
    stages {
      stage("Prepare Inventory") {
        steps {
          script {
            sh '''
            echo "Preparing ansible inventory file"
            sudo rm -f /etc/ansible/hosts
            sudo touch /etc/ansible/hosts
            sudo chmod 666 /etc/ansible/hosts
            '''
          }
        }
      }

      stage("Terraform") {
        steps {
          script {
            sh '''
            echo "Creating instance"
            cd /data/devops/Jenkins-workspace/Example_3/ec2_creation && /usr/local/bin/terraform apply -auto-approve && /usr/local/bin
            '''
            }
          }
        }

      stage("Ansible") {
        steps {
          script {
            sh '''
            echo "Executing playbook"
            sleep 120
            sudo /usr/local/bin/ansible-playbook /data/devops/Jenkins-workspace/Example_3/playbook/apache.yml --key-file "/data/devops
            '''
            }
          }
        }
      }
    }
```

# Create groovy pipeline which can integrate Terraform & Ansible [Import choices]

> Action in Jenkins dashboad

- Install all blue ocean plugins
- Create pipeline job add all the codes in build area which executes Example_3

```
pipeline {
  agent any
    stages {
```

```
    stage("Generic Inputs") {
      input {
        message "Select what task you want to perform ?"
        parameters {
          booleanParam(defaultValue: 'false', description: 'Create', name: 'create')
          booleanParam(defaultValue: 'false', description: 'Destroy', name: 'destroy')
          }
        }
        steps {
        script {
          env_create = "${create}"
          env_destroy = "${destroy}"
          }
        }
      }

    stage("Prepare Inventory") {
      steps {
        script {
          if (env_create == 'true') {
          sh '''
          echo "Preparing ansible inventory file"
          sudo rm -f /etc/ansible/hosts
          sudo touch /etc/ansible/hosts
          sudo chmod 666 /etc/ansible/hosts
          '''
          }
        }
      }
    }

    stage("Terraform") {
      steps {
        script {
          if (env_create == 'true') {
          sh '''
          echo "Creating instance"
          cd /data/devops/Jenkins-workspace/Example_4/ec2_creation && /usr/local/bin/terraform apply -auto-approve && /usr/local/bin
          '''
          }
        }
      }
    }


    stage("Ansible") {
      steps {
        script {
          if (env_create == 'true') {
          sh '''
          echo "Executing playbook"
          sleep 120
          sudo /usr/local/bin/ansible-playbook /data/devops/Jenkins-workspace/Example_4/playbook/apache.yml --key-file "/data/devops
          '''
          }
        }
      }
    }

    stage("Destroy") {
      steps {
        script {
          if (env_destroy == 'true') {
          sh '''
          echo "Destroy instance"
          cd /data/devops/Jenkins-workspace/Example_4/ec2_creation && /usr/local/bin/terraform destroy -auto-approve
          '''
          }
        }
      }
    }
  }
}
```