**Name :- Komal N. Mhetre**
**Role:- ( DevOps Engineer )**
**Tast:- Publish the artifacts through the Jenkins pipeline**

# Jenkins

- **Introduction :-**

Jenkins is an open-source automation server widely used for building, deploying, and automating projects. Jenkins support many applications. It is famous open-source tool.

Jenkins is a CI-CD.

CI-CD is a continuous your cycle workflow and deployment.

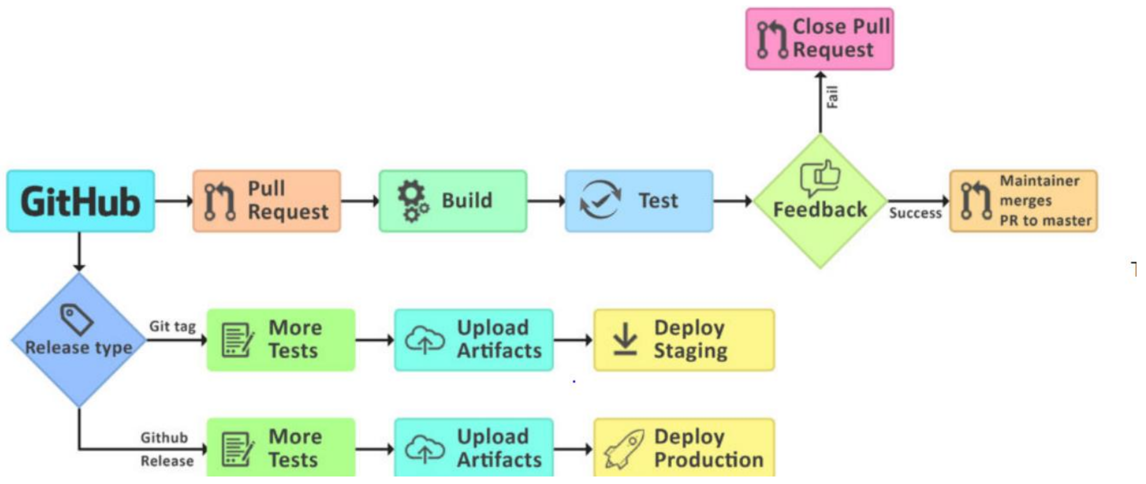
To automate the software and development workflow and deploy better quality code more obtain using the continuous and Iterativeprocesses to build, test and deploy helps avoid works and code failure.

Jenkins can integrate with various version control systems, build tools, and testing frameworks, making it a versatile tool for automating software development workflows.

- **Continuous Integration workflow**

Continuous integration is a software development procedure where each applied change invokes an automated build test. The process ensures the code integrates into a working executable form without bugs.

1. **Integrated** – All changes up until that point are combined into the project
2. **Built** – The code is compiled into an executable or package
3. **Tested** – Automated test suites are run
4. **Archived** – Versioned and stored so it can be distributed as is, if desired
5. **Deployed** – Loaded onto a system where the developers can interact with it



## Continuous Integration: Code

### Entry Points

Developers start coding based on the tasks and requirements defined in the plan.

### Tools

Github Gitlab BitBucket

**Why:** These platforms facilitate collaborative coding, diff tracking, branching, and merging, enhancing code quality and accessibility

### Exit Point

Once the code is written and reviewed, it's committed to a version control system. This triggers the transition to the 'build' stage.
Transition takes place because the required features or bug fixes have been coded and now need to be built into an executable program.

## Continuous Integration: Build

### Entry Points

Starts as soon as a new commit is made in the version control system.

### Tools

Maven Gradle Travis CI

Why: These tools optimize the transformation of code into a deployable format and enable automated build creation, aiding in containerization and continuous integration.

### Exit Point

The build process compiles the code, checks dependencies, and creates an executable artifact. The completion of this process triggers the 'test' phase.
Transition occurs as the code needs to be tested for issues before it is released.

## Continuous Deployment: Artifacts

### Entry Points

When ever build is completed. Artifacts will be Stored safely

### Tools

JFrog Docker hub Nexus

Why: All of your artifacts are managed in a single universal artifact repository manager that delivers scale, reliability, and stability of automation and crucial systems while eliminating bottlenecks. Authentication integrations such as SAML, SCIM, LDAP, OAuth, etc.

### Exit Point

Artifacts pulled from the build section.
It uniquely stores artifacts using checksum-based storage. A file that is uploaded to Artifactory, first has its SHA1 checksum calculated, and is then renamed to its checksum.

## Continuous Deployment: Test

### Entry Points

Start of the testing process on the build artifact.

### Tools

Sonar Qube JUnit k6

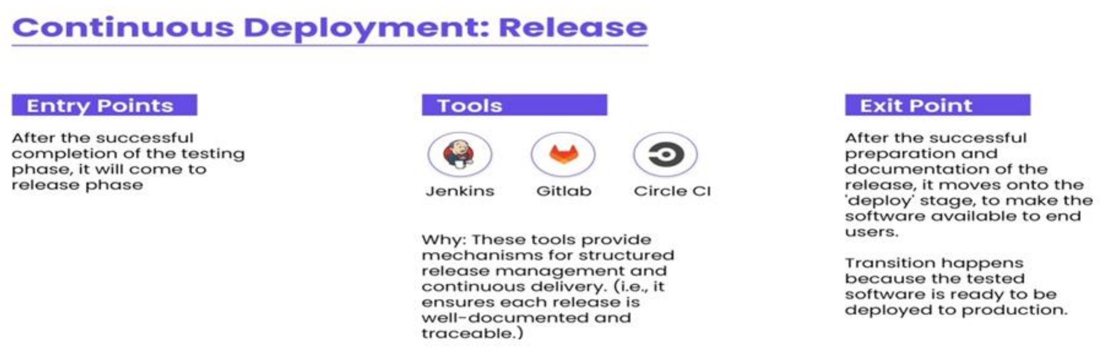
Why: These tools facilitate comprehensive testing capabilities for ensuring feature integrity and performance.

### Exit Point

Artifacts will be pulled for further testing process.

After the testing process, if the tests are successful, it transitions to the 'release' stage. If the test fails, it goes back to the 'code' stage for bug fixing.

Transition takes place to prepare the successful build for release.



## Continuous Deployment: Release

**Entry Points**

After the successful completion of the testing phase, it will come to release phase

**Tools**

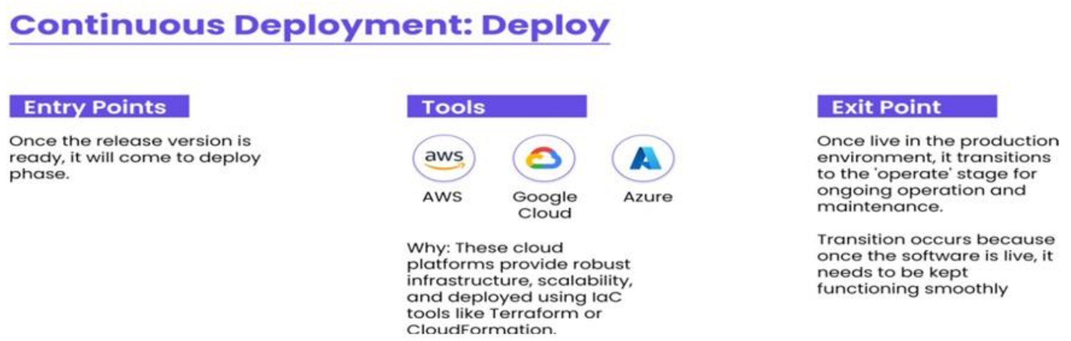
Jenkins
Gitlab
Circle CI

Why: These tools provide mechanisms for structured release management and continuous delivery. (i.e., it ensures each release is well-documented and traceable.)

**Exit Point**

After the successful preparation and documentation of the release, it moves onto the 'deploy' stage, to make the software available to end users.

Transition happens because the tested software is ready to be deployed to production.



## Continuous Deployment: Deploy

**Entry Points**

Once the release version is ready, it will come to deploy phase.

**Tools**

AWS
Google Cloud
Azure

Why: These cloud platforms provide robust infrastructure, scalability, and deployed using IaC tools like Terraform or CloudFormation.

**Exit Point**

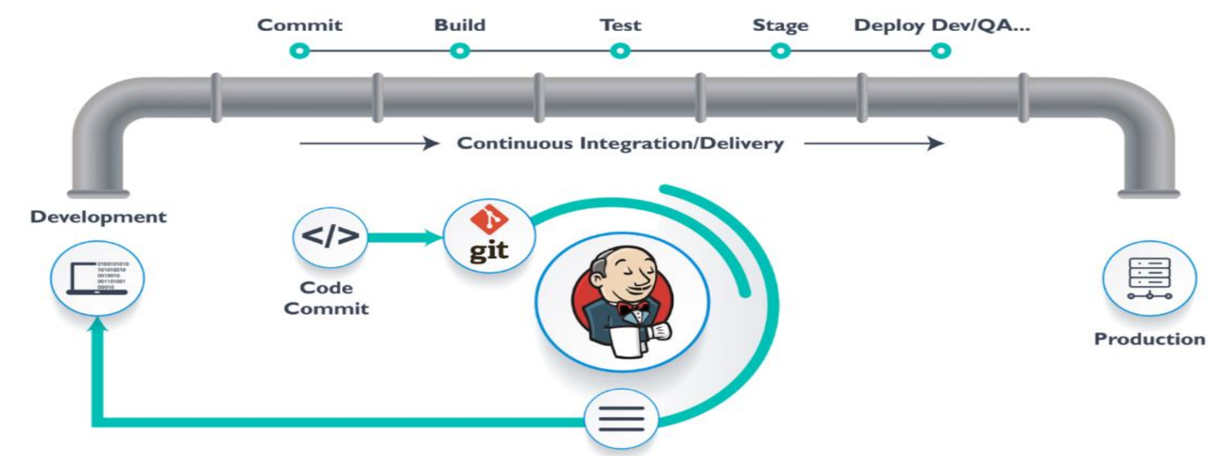
Once live in the production environment, it transitions to the 'operate' stage for ongoing operation and maintenance.

Transition occurs because once the software is live, it needs to be kept functioning smoothly

- **Continuous Integration Tools**
  - ➢ Code Repositories : SVN, Mercurial, Git
  - ➢ Continuous Build Systems : Jenkins, Bamboo, Cruise Control
  - ➢ Test Frameworks : JUnit, Cucumber, CppUnit
  - ➢ Artifact Repositories : Nexus, Artifactory, Archiva

- **Jenkins Tool Workflow and Key Features of Jenkins**
  - ➢ It will Generate test reports
  - ➢ Jenkins can Integrate with many different Version Control Systems
  - ➢ Jenkins will Deploys directly to production or test environments and many more

test, package, stage, deploy, static analysis, and much more.

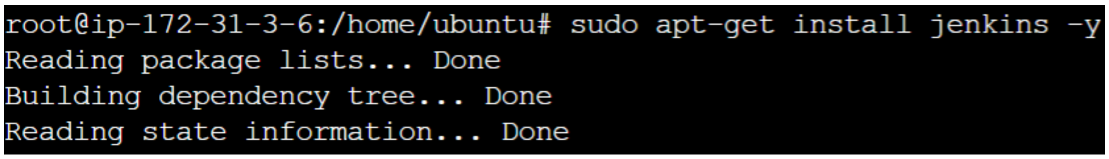


- **Jenkins Installation**

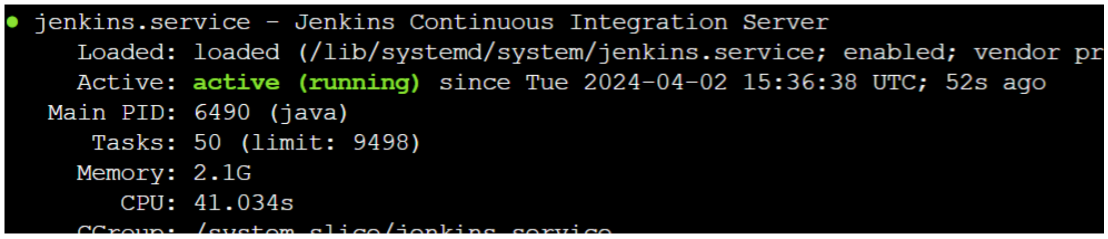
   1. Install jdk java file to run the maven and Jenkins And then install their packages before installing the jenkins.

**Below are the commands to installation**

a. sudo apt-get update
b. sudo apt install openjdk-17-jdk
c. curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key| sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
d. echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
e. sudo apt-get update
f. sudo apt-get install jenkins
g. sudo systemctl enable jenkins
h. sudo systemctl start jenkins
i. sudo systemctl status jenkins



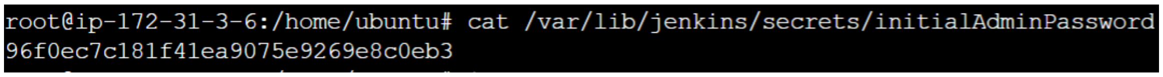
```
root@ip-172-31-3-6:/home/ubuntu# sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```



```
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor pr
     Active: active (running) since Tue 2024-04-02 15:36:38 UTC; 52s ago
   Main PID: 6490 (java)
      Tasks: 50 (limit: 9498)
     Memory: 2.1G
        CPU: 41.034s
```

2. Then access Jenkins in a Web Browser. Using default port number IP:8080

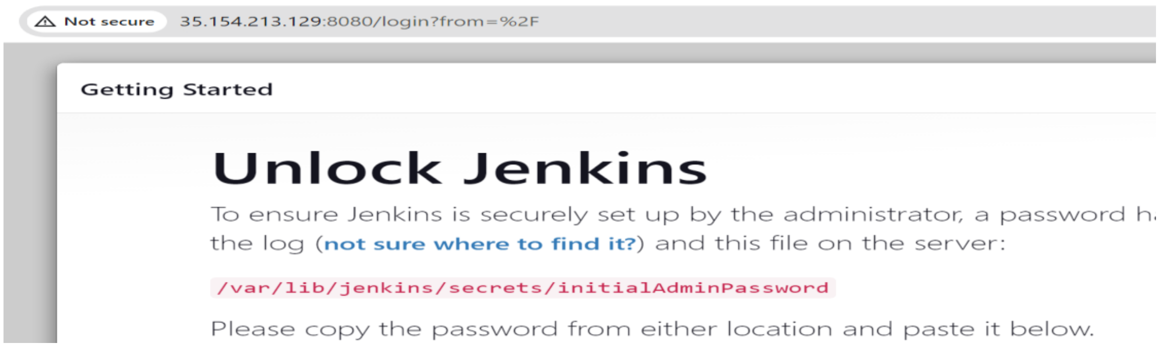
3. If port 8080 is not functioning for Jenkins, navigate to the server, choose the security settings edit the inbond rule, and include port 8080. Save the modification and the access it through the browser, it should be operational now.



```
root@ip-172-31-3-6:/home/ubuntu# cat /var/lib/jenkins/secrets/initialAdminPassword
96f0ec7c181f41ea9075e9269e8c0eb3
```

4. Copy the public IP of your running instance paste in the browser with 8080 port number allow all required specifications.

   http://localhost:8080



Not secure 35.154.213.129:8080/login?from=%2F

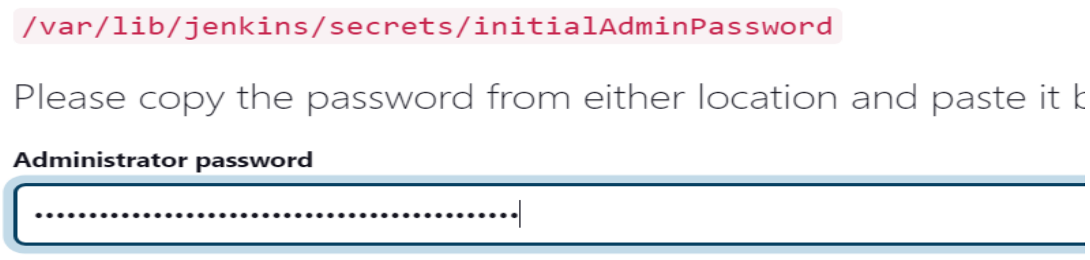
Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password h
the log (not sure where to find it?) and this file on the server:

/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it below.

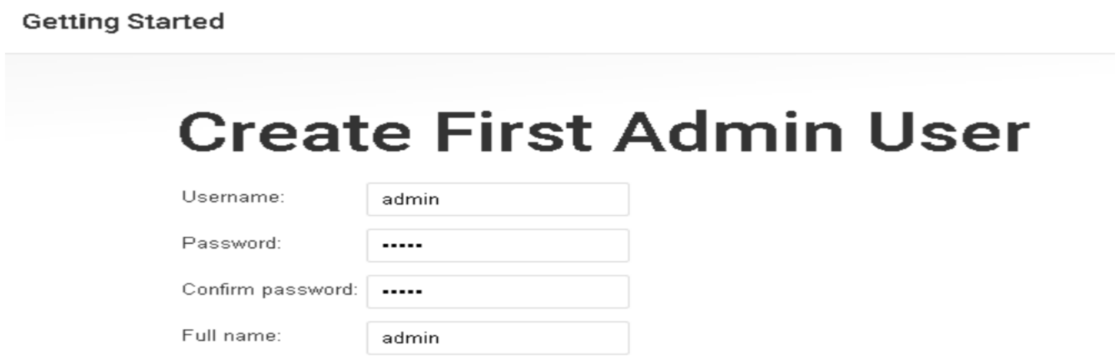


/var/lib/jenkins/secrets/initialAdminPassword

Please copy the password from either location and paste it b

Administrator password: ••••••••••••••••••••••••••••••••••••••••••••••••

5. Next it will takes you to 'Create First Admin User', provide details & finish

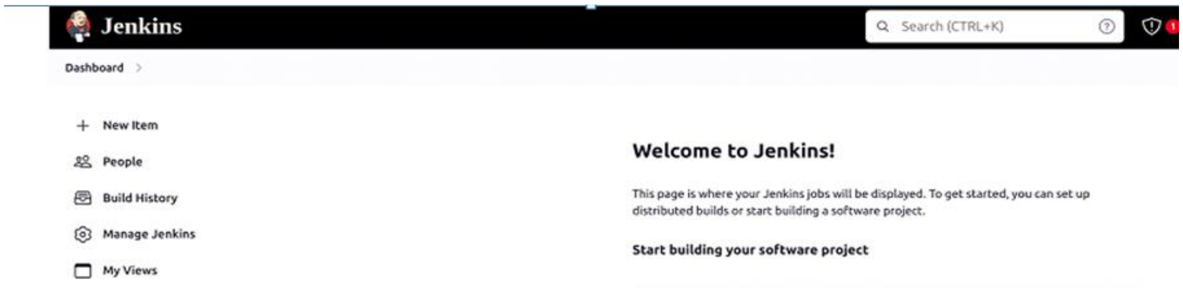


Getting Started

Create First Admin User

| | |
|---|---|
| Username: | admin |
| Password: | ••••• |
| Confirm password: | ••••• |
| Full name: | admin |

6. It will open homepage as below.


Jenkins

Search (CTRL+K)

Dashboard >

New Item

People

Build History

Manage Jenkins

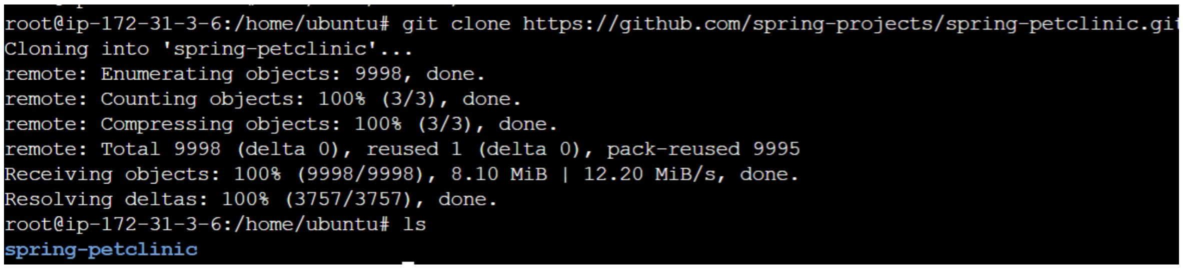
My Views

**Welcome to Jenkins!**

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

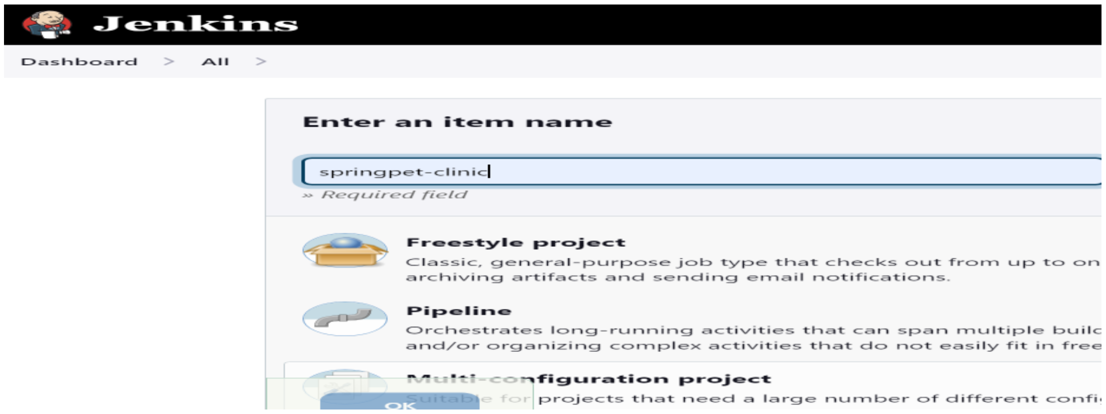
**Start building your software project**

Using the following command to clone the repository.


```
root@ip-172-31-3-6:/home/ubuntu# git clone https://github.com/spring-projects/spring-petclinic.gi
Cloning into 'spring-petclinic'...
remote: Enumerating objects: 9998, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 9998 (delta 0), reused 1 (delta 0), pack-reused 9995
Receiving objects: 100% (9998/9998), 8.10 MiB | 12.20 MiB/s, done.
Resolving deltas: 100% (3757/3757), done.
root@ip-172-31-3-6:/home/ubuntu# ls
spring-petclinic
```

7. See below process for creation of pipeline-

   Click on new item in the dashboard enter a item name, click on pipeline and ok.


Jenkins

Dashboard > All >

**Enter an item name**

springpet-clinic

» Required field

**Freestyle project**

Classic, general-purpose job type that checks out from up to on archiving artifacts and sending email notifications.

**Pipeline**

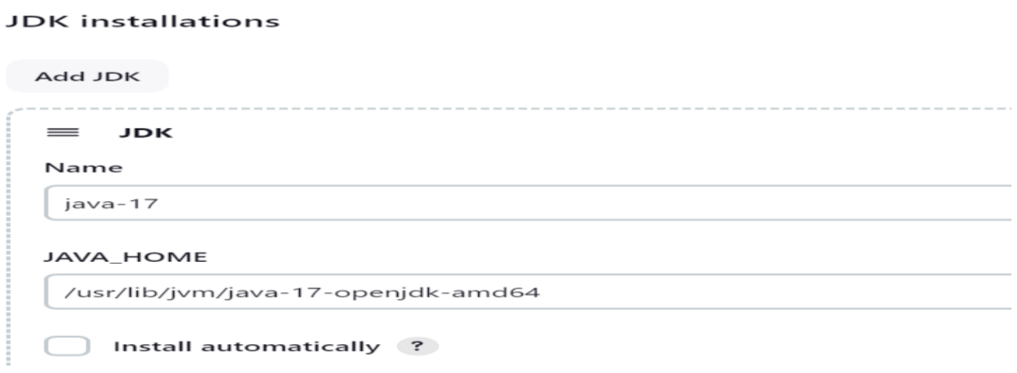
Orchestrates long-running activities that can span multiple builc and/or organizing complex activities that do not easily fit in free

**Multi-configuration project**

Suitable for projects that need a large number of different confi

OK

**Configure the JDK and Maven in Jenkins**


JDK installations

Add JDK

JDK

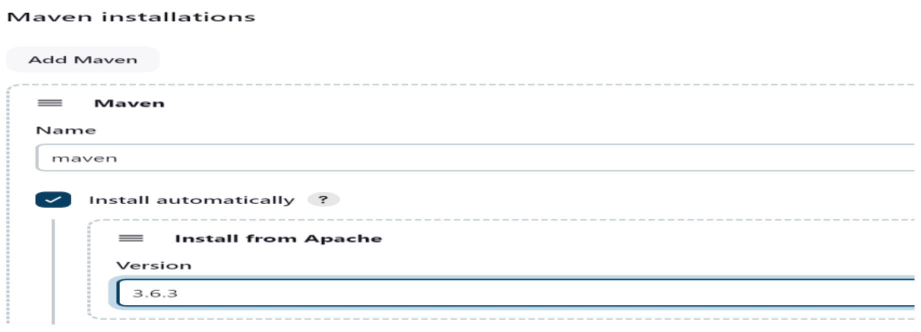
Name

java-17

JAVA_HOME

/usr/lib/jvm/java-17-openjdk-amd64

Install automatically ?



Maven installations

Maven

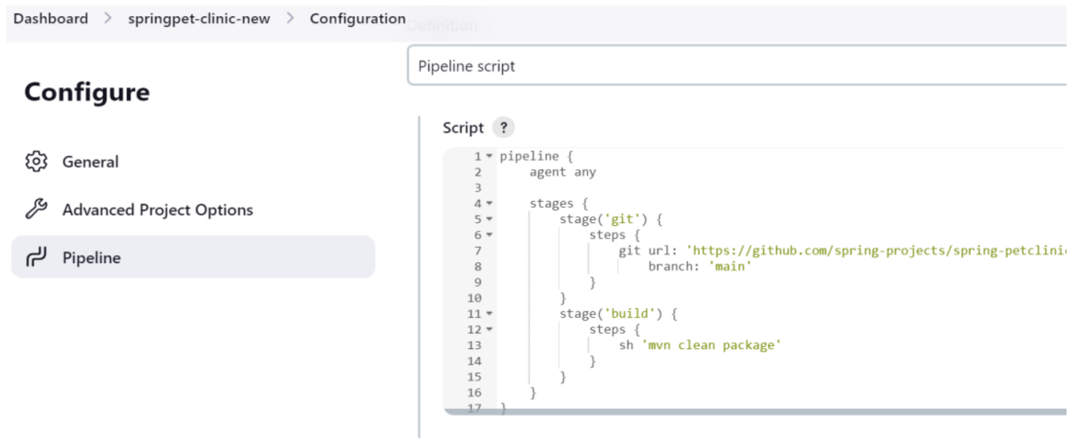
Name: maven

Install automatically [✓]

Install from Apache

Version: 3.6.3

8. Write a pipeline and build the package. After that save and click on build now then check the console output

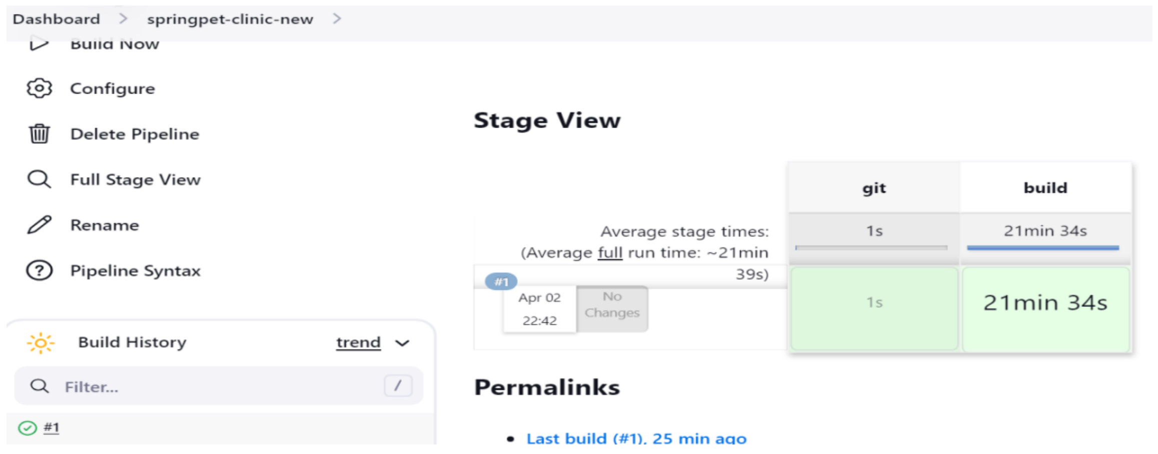


Pipeline script

Script

```
pipeline {
    agent any

    stages {
        stage('git') {
            steps {
                git url: 'https://github.com/spring-projects/spring-petclini
                    branch: 'main'
            }
        }
        stage('build') {
            steps {
                sh 'mvn clean package'
            }
        }
    }
}
```



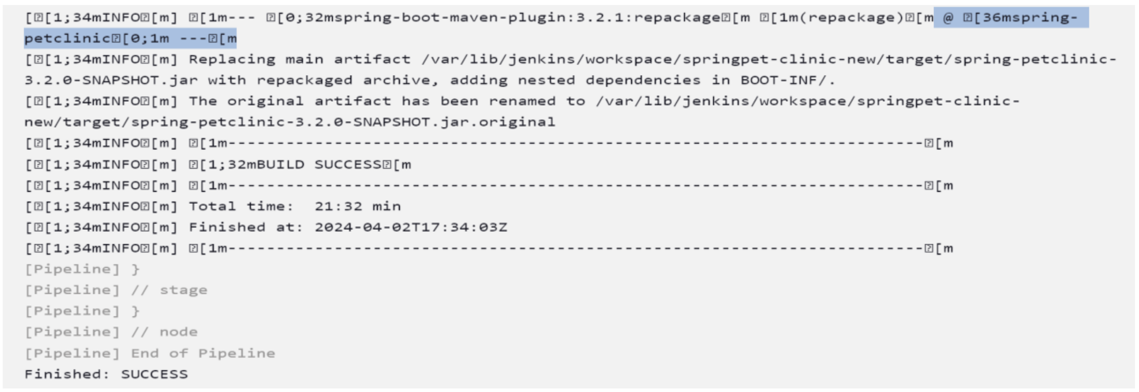
Stage View

| | git | build |
|---|---|---|
| Average stage times: (Average full run time: ~21min 39s) | 1s | 21min 34s |
| #1 Apr 02 22:42 No Changes | 1s | 21min 34s |

Permalinks

- Last build (#1), 25 min ago

9. Save and run the pipline and see the result.



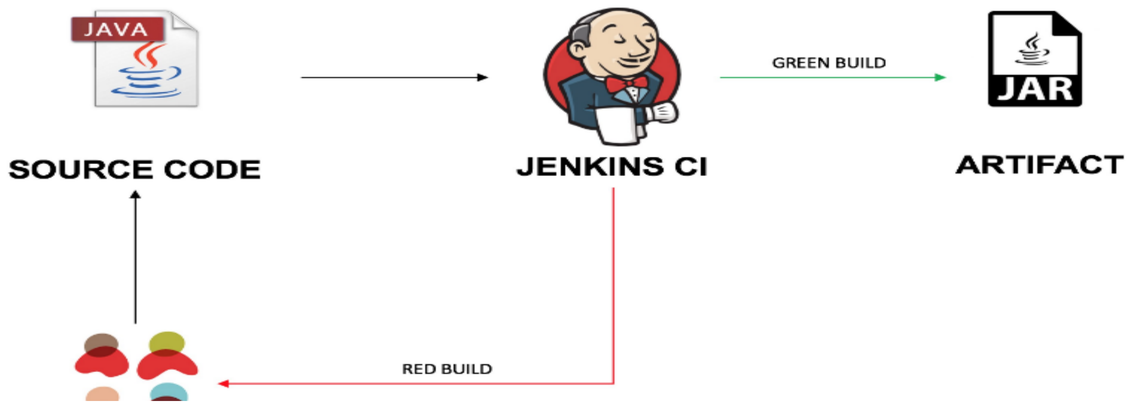
```
[⍰[1;34mINFO⍰[m] ⍰[1m--- ⍰[0;32mspring-boot-maven-plugin:3.2.1:repackage⍰[m ⍰[1m(repackage)⍰[m @ ⍰[36mspring-
petclinic⍰[0;1m ---⍰[m
[⍰[1;34mINFO⍰[m] Replacing main artifact /var/lib/jenkins/workspace/springpet-clinic-new/target/spring-petclinic-
3.2.0-SNAPSHOT.jar with repackaged archive, adding nested dependencies in BOOT-INF/.
[⍰[1;34mINFO⍰[m] The original artifact has been renamed to /var/lib/jenkins/workspace/springpet-clinic-
new/target/spring-petclinic-3.2.0-SNAPSHOT.jar.original
[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
[⍰[1;34mINFO⍰[m] ⍰[1;32mBUILD SUCCESS⍰[m
[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
[⍰[1;34mINFO⍰[m] Total time:  21:32 min
[⍰[1;34mINFO⍰[m] Finished at: 2024-04-02T17:34:03Z
[⍰[1;34mINFO⍰[m] ⍰[1m------------------------------------------------------------------------⍰[m
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

**The role of artifacts will be significant in this stage now.**

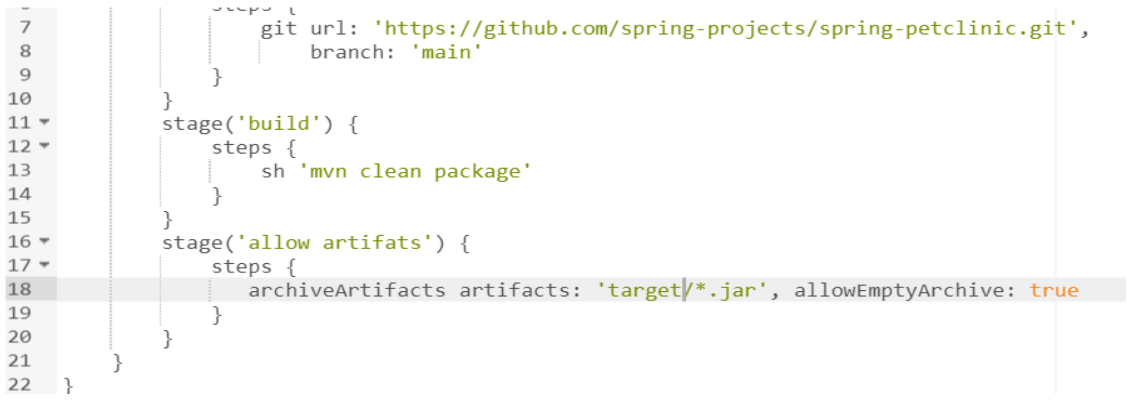
In Jenkins, an artifact refers to any file or collection of files generated during the build process. These files could be compiled code, documentation, test results, or any other output produced by the build job.

When a Jenkins job executes, it typically generates artifacts as a result of the build process. These artifacts can be archived and made available for further use, such as deployment, testing, or as a record of the build. Archiving artifacts allows Jenkins users to access and download them later, either manually or as part of downstream jobs.

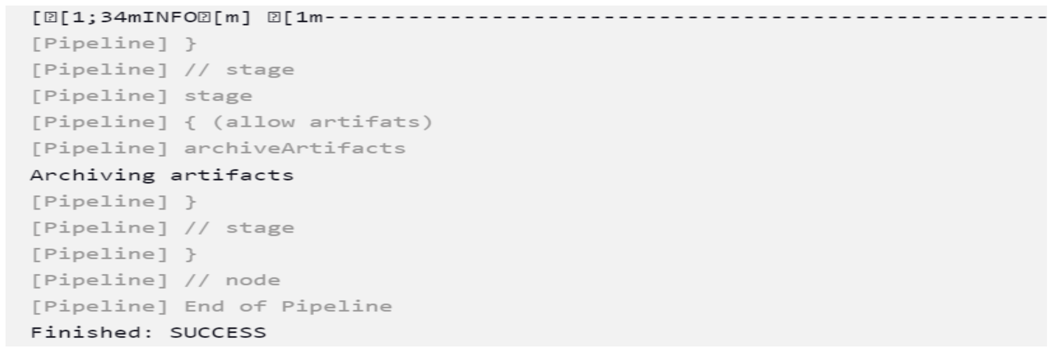
Artifact management is an essential aspect of continuous integration and continuous delivery (CI/CD) pipelines, enabling teams to share and distribute the outputs of their builds efficiently. Jenkins provides built-in support for archiving and managing artifacts through its user interface and scripting capabilities.



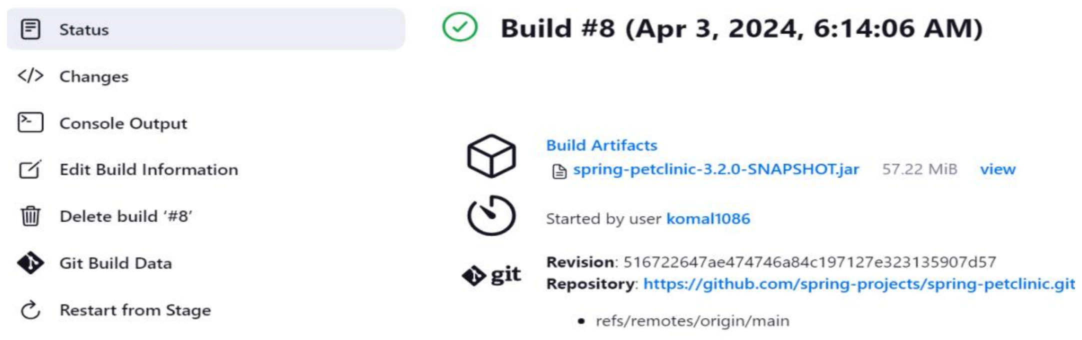
10. See below pipeline to build package and publish the artifacts.



```
                git url: 'https://github.com/spring-projects/spring-petclinic.git',
                    branch: 'main'
            }
        }
        stage('build') {
            steps {
                sh 'mvn clean package'
            }
        }
        stage('allow artifats') {
            steps {
               archiveArtifacts artifacts: 'target/*.jar', allowEmptyArchive: true
            }
        }
    }
}
```



```
[⍰[1;34mINFO⍰[m] ⍰[1m-----------------------------------------------------
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (allow artifats)
[Pipeline] archiveArtifacts
Archiving artifacts
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



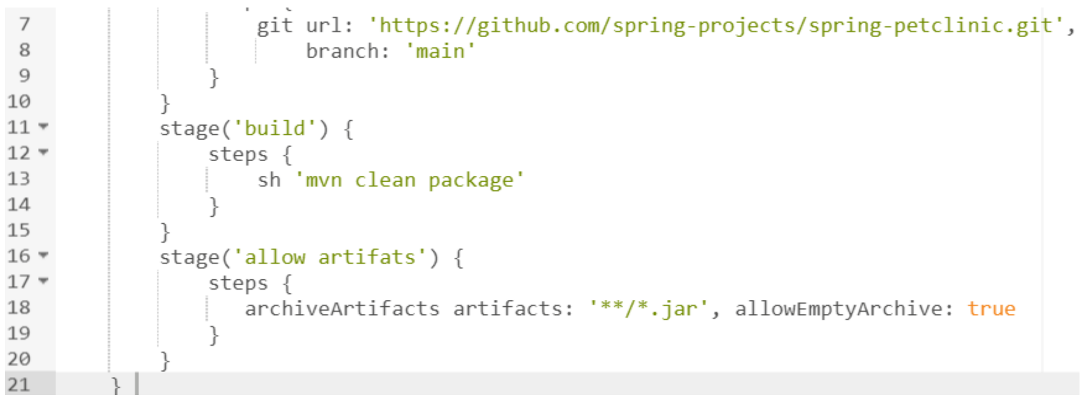
Build #8 (Apr 3, 2024, 6:14:06 AM)

Build Artifacts
spring-petclinic-3.2.0-SNAPSHOT.jar 57.22 MiB view

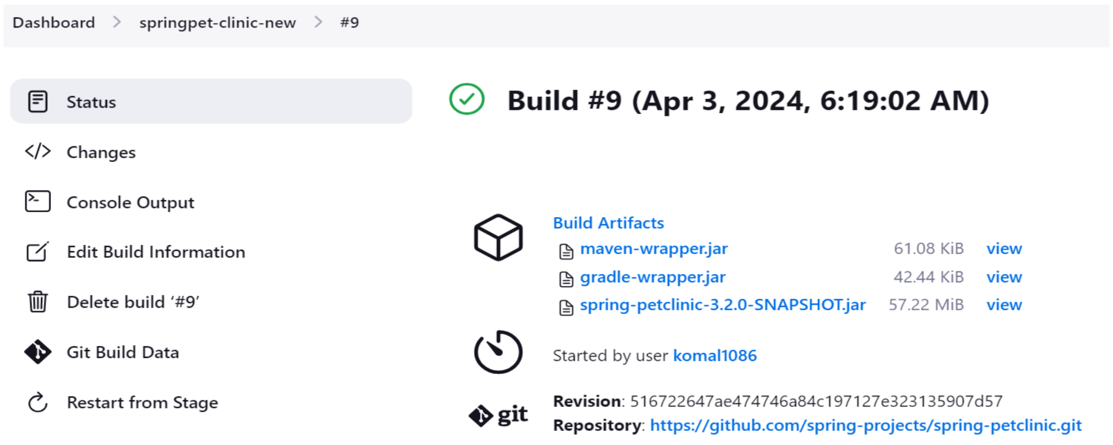
Started by user komal1086

**Revision**: 516722647ae474746a84c197127e323135907d57
**Repository**: https://github.com/spring-projects/spring-petclinic.git

- refs/remotes/origin/main



```
            git url: 'https://github.com/spring-projects/spring-petclinic.git',
                branch: 'main'
            }
        }
    stage('build') {
        steps {
            sh 'mvn clean package'
        }
    }
    stage('allow artifats') {
        steps {
           archiveArtifacts artifacts: '**/*.jar', allowEmptyArchive: true
        }
    }
}
```



Dashboard > springpet-clinic-new > #9

## Build #9 (Apr 3, 2024, 6:19:02 AM)

Build Artifacts

| | | |
|---|---|---|
| maven-wrapper.jar | 61.08 KiB | view |
| gradle-wrapper.jar | 42.44 KiB | view |
| spring-petclinic-3.2.0-SNAPSHOT.jar | 57.22 MiB | view |

Started by user komal1086

**Revision**: 516722647ae474746a84c197127e323135907d57
**Repository**: https://github.com/spring-projects/spring-petclinic.git