Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Datta Meghe College of Engineering, Airoli, Navi Mumbai

DEPARTMENT OF INFORMATION TECHNOLOGY

Institute Vision : **To create value - based technocrats to fit in the world of work and research**

Institute Mission : **To adapt the best practices for creating competent human beings to work in the world of technology and research.**

Department Vision : **To develop and foster students for successful careers in the dynamic field of Information Technology.**

Department Mission :

M1:	**To create and disseminate knowledge through research, teaching & learning and to enhance society in meaningful and sustainable ways.**
M2:	**To impart a suitable environment for students and staff to showcase innovative ideas in the field of IT.**
M3:	**To bridge the curriculum gap by facilitating effective interaction among industry and Staff/Students.**

Program Educational Objectives (PEO)

PEO 1	**Develop proficiency as an IT technocrat with an ability to solve a wide range of computational problems in industry, government, or other work environments.**
PEO 2	**Attain the ability to adapt quickly to new environments and technologies, assimilate new information, and work in multi-disciplinary areas with a strong focus on innovation and entrepreneurship.**
PEO 3	**Prepare graduates with the ability of life-long learning to innovate in ever-changing global economic and technological environments of the current era.**
PEO 4	**Possess the ability to function ethically and responsibly with good cultural values and integrity to apply the best principles and practices of Information Technology towards the society.**

Program Specific Outcomes (PSO)

PSO1	**Apply Core Information Technology knowledge to develop stable and secure IT system**
PSO2	**Design, IT infrastructures for an enterprise using concepts of best practices in Information Technology and security domain.**
PSO3	**Ability to work in multidisciplinary IT enabled projects for industry and society by adapting latest trends and technologies like Analytics, Blockchain, Cloud, Data science.**

Datta Meghe College of Engineering, Airoli

Department of Information Technology

Course Name: DevOPs Lab **(R-19)**

Course Code: ITL503

Year of Study: 2021-22 **Semester:** V

Course Outcomes

ITL503.1	To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements
ITL503.2	To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub
ITL503.3	To understand the importance of Jenkins to Build and deploy Software Applications on server environment
ITL503.4	Understand the importance of Selenium and Jenkins to test Software Applications
ITL503.5	To understand concept of containerization and Analyze the Containerization of OS images and deployment of applications over Docker
ITL503.6	To Synthesize software configuration and provisioning using Ansible.

DATTA MEGHE COLLEGE OF ENGINEERING

AIROLI, NAVI MUMBAI - 400708

C E R T I F I C A T E

This is to certify that Mr / ~~Miss~~ Athang Milind Patil of TE Class B Roll No. 05

Subject Devops Lab has performed the experiments /

Sheets mentioned in the index, in the premises of this institution.

Dr. Sujata Kolhe	Dr. Sujata Kolhe	Dr. Sudhir D. Sawarkar
Practical Incharge	**Head of Dept.**	**Principal**

Date 21/10/2021

Examined on

Examiner 1 _____ Examiner 2 _____

Datta Meghe College of Engineering, Airoli, Navi Mumbai	DEPARTMENT OF INFORMATION TECHNOLOGY
	List of Experiments **Subject: Devops Lab** **Code: ITL503**

Sr.No.	Name of the Experiment	LOs Covered	Page No.	Date	Signature
1.	To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.	LO1	2-7	13.07.21	
2.	To understand Version Control System / Source Code Management, install git and create a GitHub account.	LO1 & LO2	8-15	27.07.21	
3.	To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet	LO1 & LO2	16-22	27.07.21	
4.	To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job.	LO1 & LO3	23-34	03.08.21	
5.	To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.	LO1 & LO3	35-44	03.08.21	
6.	To understand Jenkins master- slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.	LO1 & LO3	45-50	10.08.21	
7.	To Setup and Run Selenium Tests in Jenkins Using Maven.	LO1 , LO3 & LO4	51-53	17.08.21	
8.	To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.	LO1 & LO5	54-59	07.09.21	
9.	To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.	LO1 & LO5	60-61	21.09.21	
10.	To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet.	LO1 & LO6	62-67	28.09.21	
11.	To learn Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)	LO1 & LO6	68-76	05.10.21	

SR NO.	Name of Experiment	CO Covered	Page No.	Date	Signature
1	Assignment No . 1	LO1-LO6	77-80	19.08.21	
2	Assignment No . 2	LO1-LO6	81-87	01.10.21	

EXPERIMENT No. 1

Aim: Introduction to Devops

Theory:

Evolution of software Development over the years

i) Waterfall Methodology.

The evolution of software development life cycle starts in mid 20th century. The software was developed by code & fix techniques which included two steps 1) write some code 2) fix the problems of this code However in 1956, the experience recognizes the problems with more large software developement, and then a model stagewise was originated. Royce (1970) proposed the waterfall model in order to avoid the difficult nature of "code & fix" approach. He proposed the construction of a prototype & involvement of various several phase. In various researches the waterfall model is classified as a traditional methodology. The central concept of waterfall model is classified integrated verification & validation of the result by the customer in order to complete a certain phase.

The waterfall methodology are mainly quality management & documentation. The waterfall methodology describes a sequential proceeding strategy. also covering all phases from requirement to operation while not explicity covering maintences & disassemb-y.

This allows a single team to handle the entire application lifestyle, from development to testing, deployment & operations

Devops can also be defined as a sequence of development of IT operations with better communication & collaboration

Devops helps to increase organizations speeds to deliver application & services

* Why devops ?

We need to understand why we need the Devops over the other methods

I. The operations & development team worked incomplete isolations

II. After the design build, the testing & deployment are performed respectively that's why they consumed more time than actual build cycles

III] Without the use of devops, the team members are spending a large amount of time on designing, testing & deploying insted of building the project.

IV] Manual code deployment leads to human errors in production.

* Why Devops is important ?

Devops is important because its a software development & operations approach that enables faster development of new products & easier maintances of existance deployment

2. Agile Methodology:

Agile Methodology are a new host of methodologies that claim to overcome the limitations of traditional plan drivers (SDMs).

Agile Methodology meaning a practice that promotes continuous iteration of developement & testing throughout the software development lifecycle of the project.

The manifesto states that agile development should focus on four core values.

i) Individuals & interactions over processes
ii) working software over comprehensive documentation.
iii) Customer collaboration over contract negotiation.
iv) Responding to change over following a plan. Most studies reported that agile development practice are easy to ddopt & work well.

Waterfall

|Design| |code| |Test| |Deploy|

Agile

|Design| |code| |Test| |code|Test|□|code|Test|Deploy|

Agile with continous deployment.

|Design|□□□□□□□□□□□□□□□□□□□□

Agile methodology is one of the simplest & effective processes to turn a vision for a business need into software solutions. Agile is term used to describe software development approches that employ continual planning, learning, improvement, team collaboration & early delivery.

* What is Devops?

The main reason behind Devops popularity is that it allows enterprises to create & improve products at a faster place than traditional software development methods

* Benefits of Devops:-

Technical Benefits:-

i) Continuous software delivery
ii) less complexity to manage
iii) Fastest resolution of solutions.

Cultural Benefits:-

Happier, more productive teams
Higher employee engagement
Greater professional development opportunities.

Business Benefits:-

Faster delivery of features
More stable operative environment
Improved communication & collaboration
More time to innovate

Conclusion:-

Hence, we successfully studied about DevOps.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 2

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM :- To study and practice GIT commands for version control.

THEORY :-

Git is a free and open source distributed version control system designed to handle everything fromsmall to very large projects with speed and effciency. Git is easy to learn and has a tiny footprintwith lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, andClearCase with features like cheap local branching, convenient staging areas, and multiple workfows.

Some of the basic operations in Git are:

1.Initialize
2.Add
3.Commit
4.Pull
5.Push

Some advanced Git operations are:

1.Branching

2. Merging

3. Rebasing

Installation of GIT

1)In windows, download GIT from https://git-scm.com/ and perform the straightforward installation.

2) In Ubuntu, install GIT using $sudo apt install git, Confrm the version after installation $git –version

Once installation is done, open the terminal in Ubuntu and perform the following steps or in

windows Right click and select Git bash here.

To perform version control, let us create a directory dvcs (Distributed version control system)

and change directory to dvcs.

$ mkdir git-dvcs
$ cd git-dvcs/
Now check the user information using

$ git confg –global

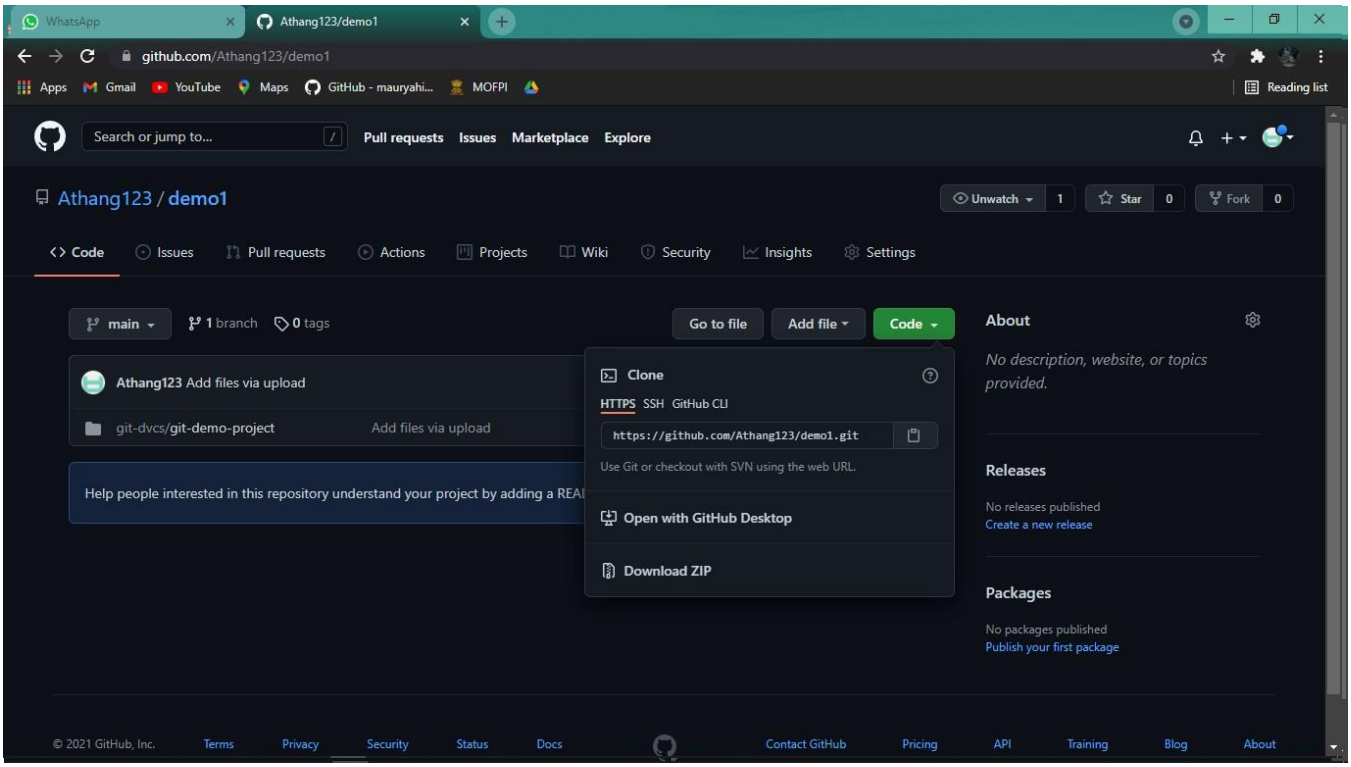
As there are no users defned, let us defne it using following two commands

$ git confg --global user.name "bhushan"

$ git confg --global user.email "bhushan,jadhav1@gmail.com" Now, check the list of users

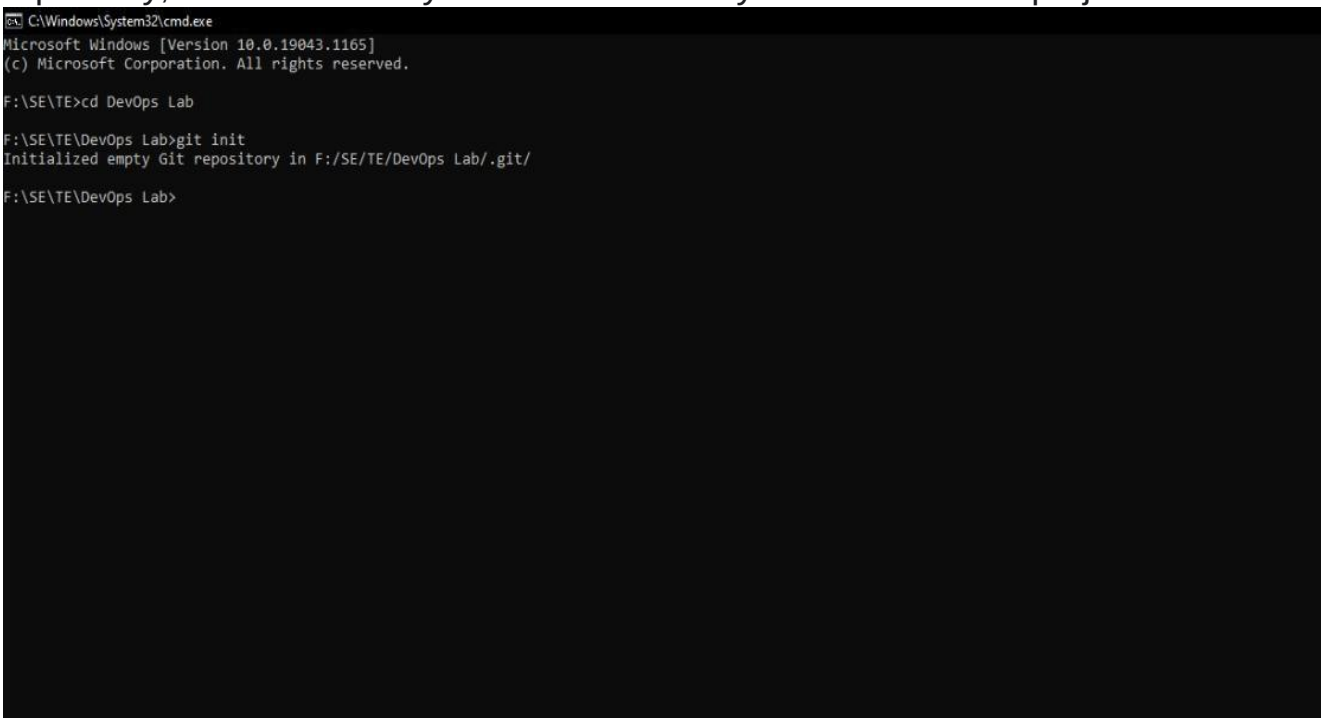
$ git confg --global --list user.name=bhushan
user.email=bhushan.jadhav1@gmail.com

GETTING STARTED :

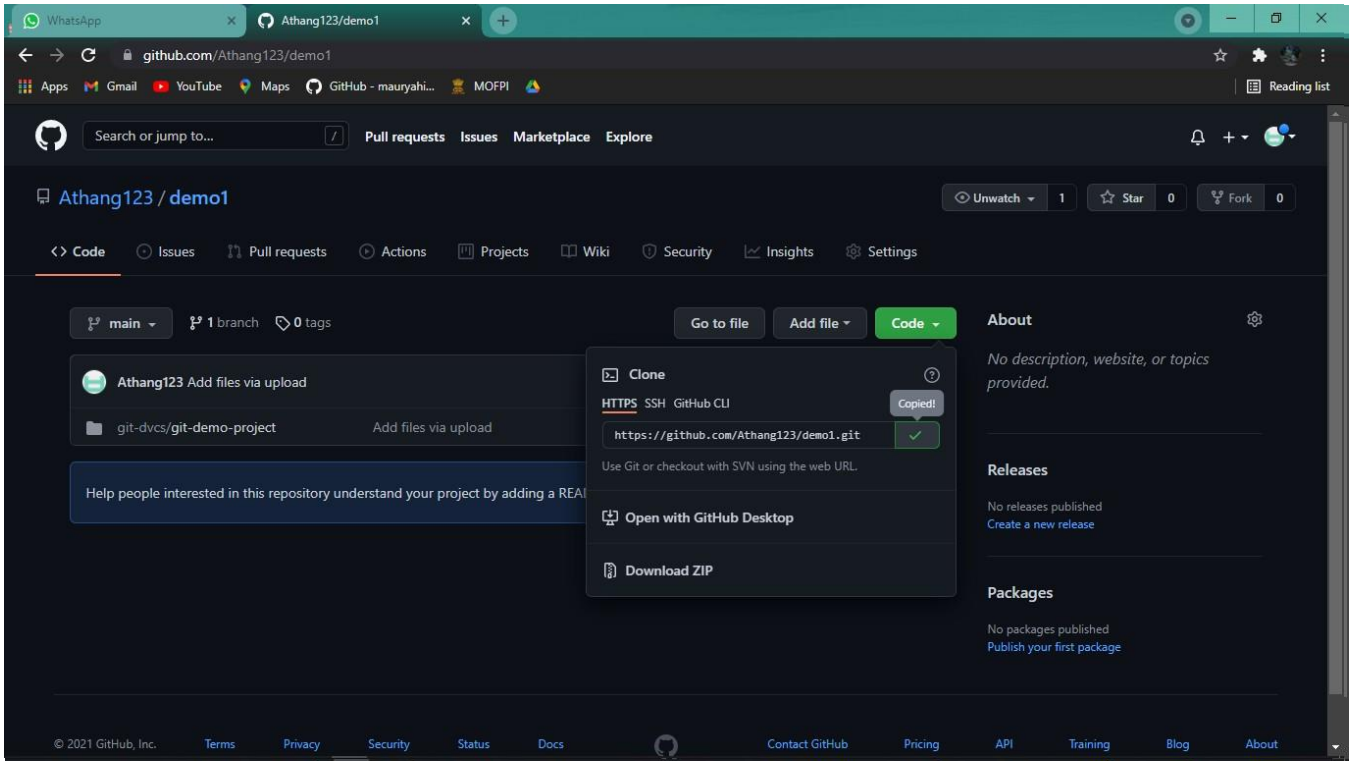


Commands:

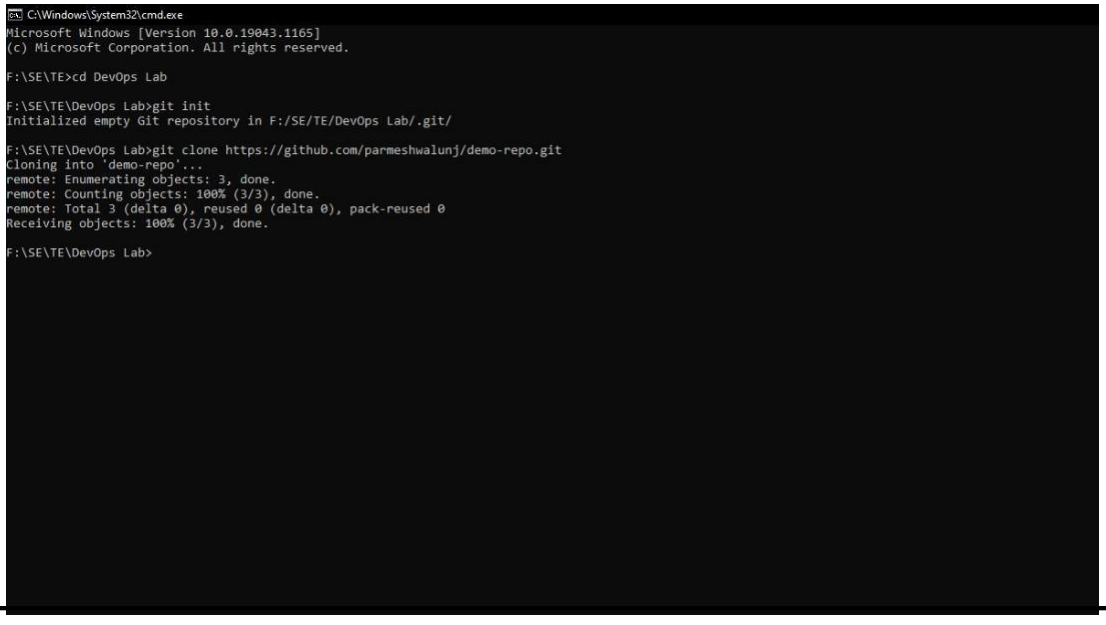
git init: The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialized repository, so this is usually the frst command you'll run in a new project.



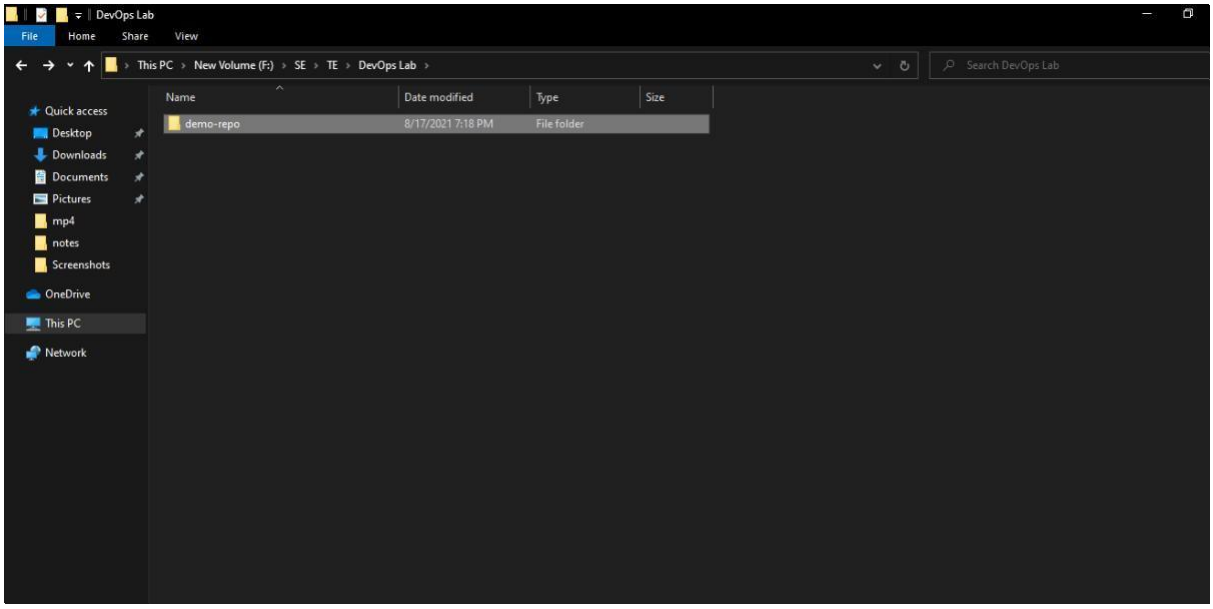
git clone: git clone is a Git command line utility which is used to target an existing repository and create a clone, or copy of the target repository. ... Cloning a local or remote repository. Cloning a bare repository. Using shallow options to partially clone repositories. Git URL syntax and supported protocols.



Step 2: Paste it after git clone & hit enter



Step 3: Repo folder is created in ur localhost

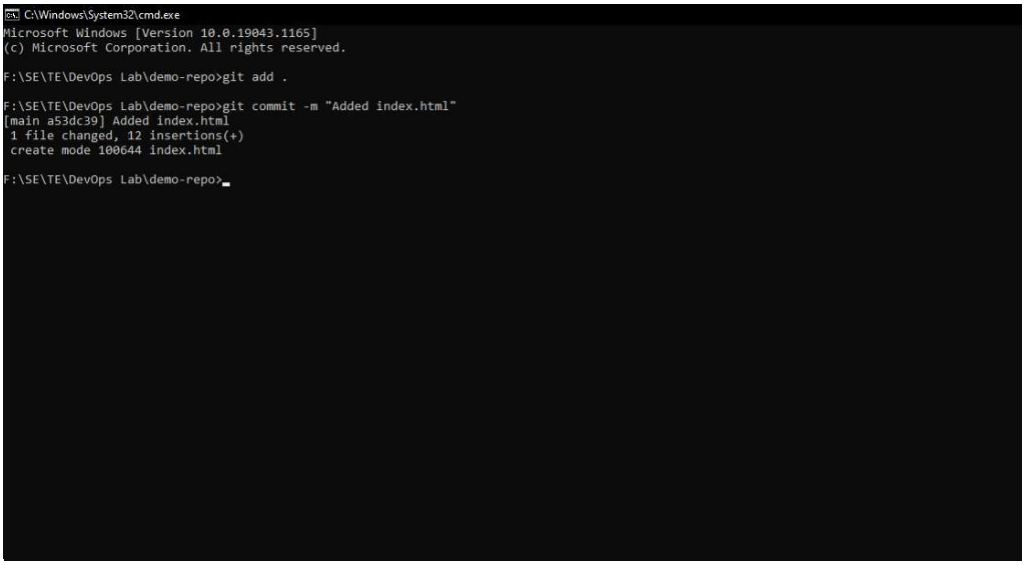


git add .

The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular fle in the next commit. However, git add doesn't really affect the repository in any signifcant way—changes are not actually recorded until you run git commit.

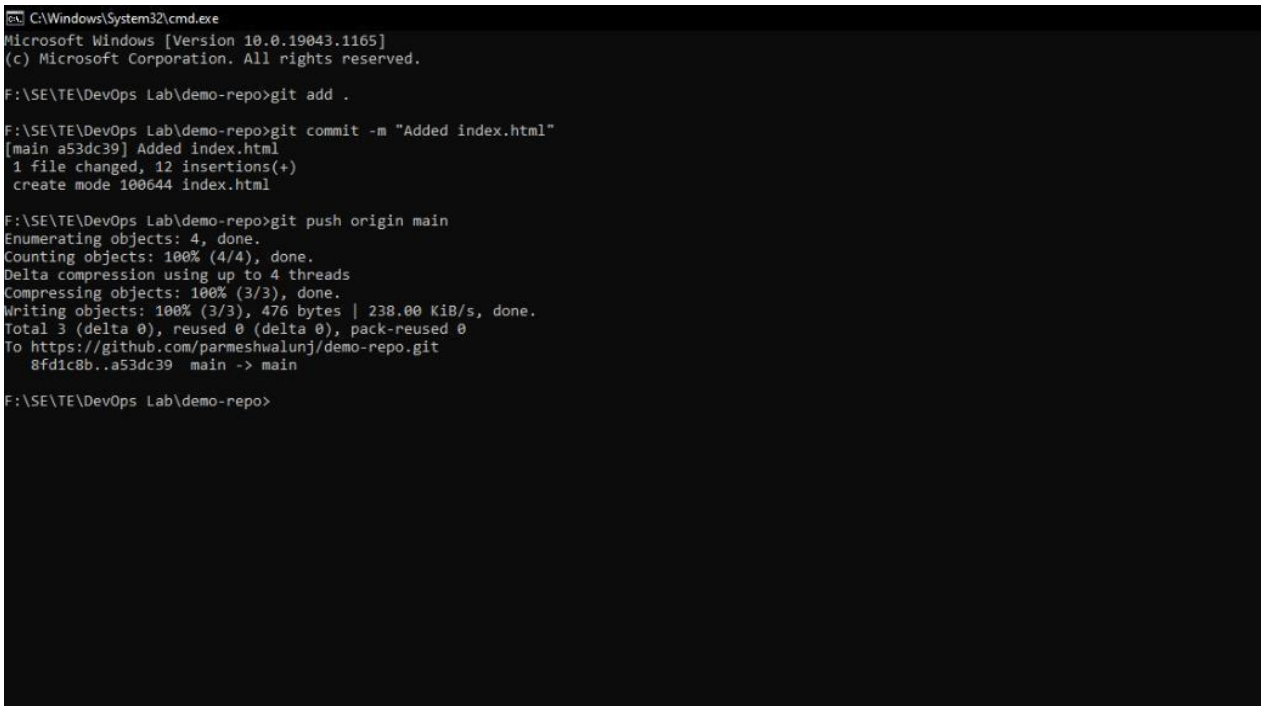
git commit -m “message goes here”

The "commit" command is used to save your changes to the local repository. ... Using the "git commit" command only saves a new commit object in the local Git repository. Exchanging commits has to be performed manually and explicitly (with the "git fetch", "git pull", and "git push" commands).



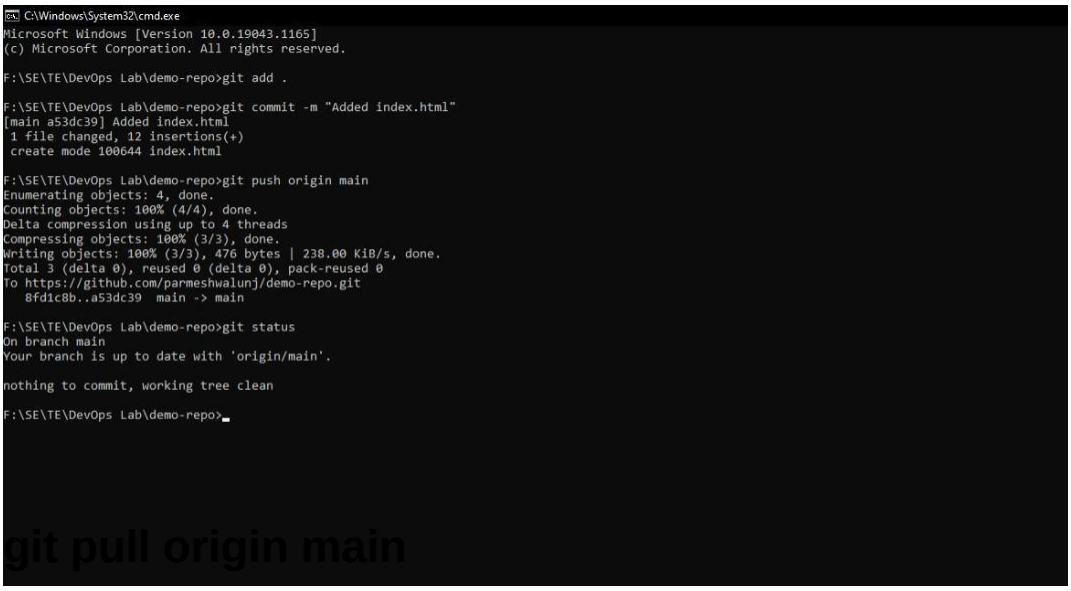
git push origin main

The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch , but whereas fetching imports commits to local branches, pushing exports commits to remote branches.



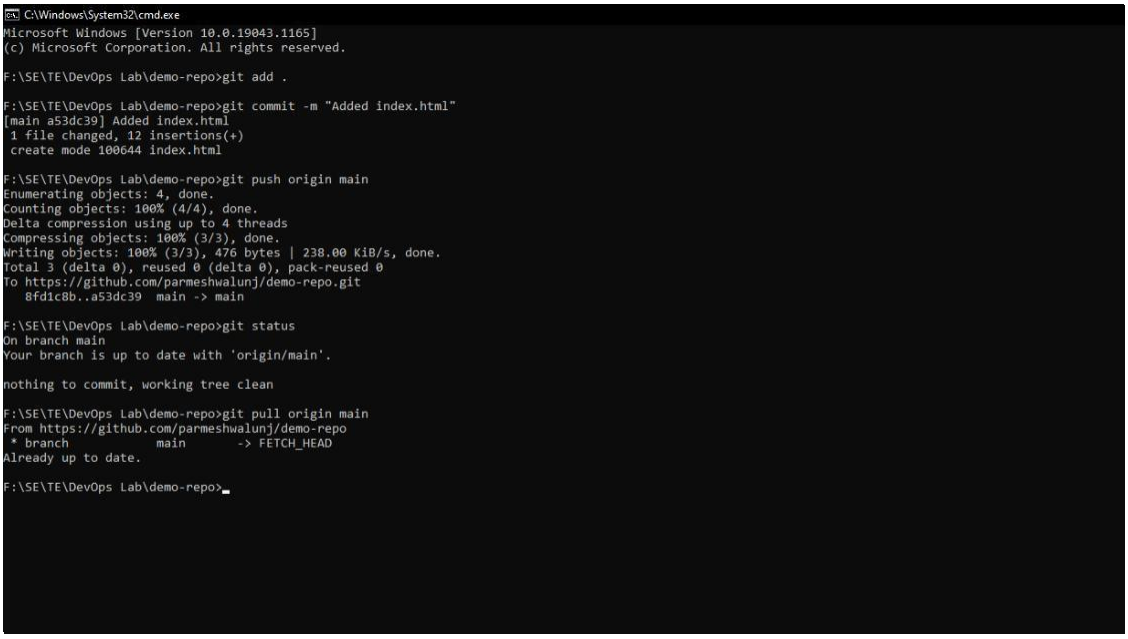
git status

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which fles aren't being tracked by Git.



The term pull is used to receive data from GitHub. It fetches and merges changes from the remote server to your working directory. The git pull command is used to pull a repository.Pull request is a process for a developer to notify team members that they have completed a feature. Once their feature

branch is ready, the developer fles a pull request via their remote server account.



Conclusion:- Hence, basic commands of git version control was studied properly

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO . 3

NAME :Athang Milind Patil
BRANCH : B1
ROLL NO: 5

AIM :- a)To practice/execute shell programs using Jenkins. b)To practice/execute parameterised Java programs using Jenkins.

THEORY :-

Jenkins is an open-source server that is written entirely in Java. It lets you execute a series of actions to achieve the continuous integration process, that too in an automated fashion.
This CI server runs in servlet containers such as Apache Tomcat. Jenkins facilitates continuous integration and continuous delivery in software projects by automating parts related to build, test, and deployment. This makes it easy for developers to continuously work on the betterment of the product by integrating changes to the project. Jenkins automates the software builds in a continuous manner and lets the developers know about the errors at an early stage. A strong Jenkins community is one of the prime reasons for its popularity. Jenkins is not only extensible but also has a thriving plugin ecosystem. Some of the possible steps that can be performed using Jenkins are:

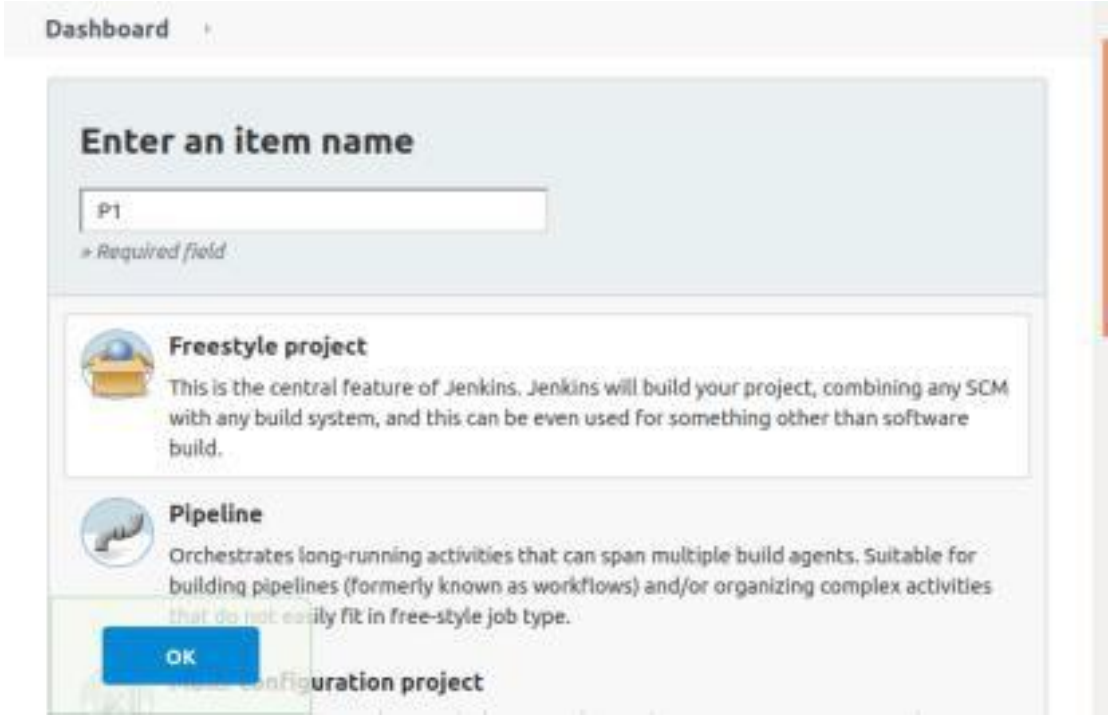
- Software build using build systems such as Gradle, Maven, and more.

· Automation testing using test frameworks such as Nose2, PyTest, Robot, Selenium, and more.

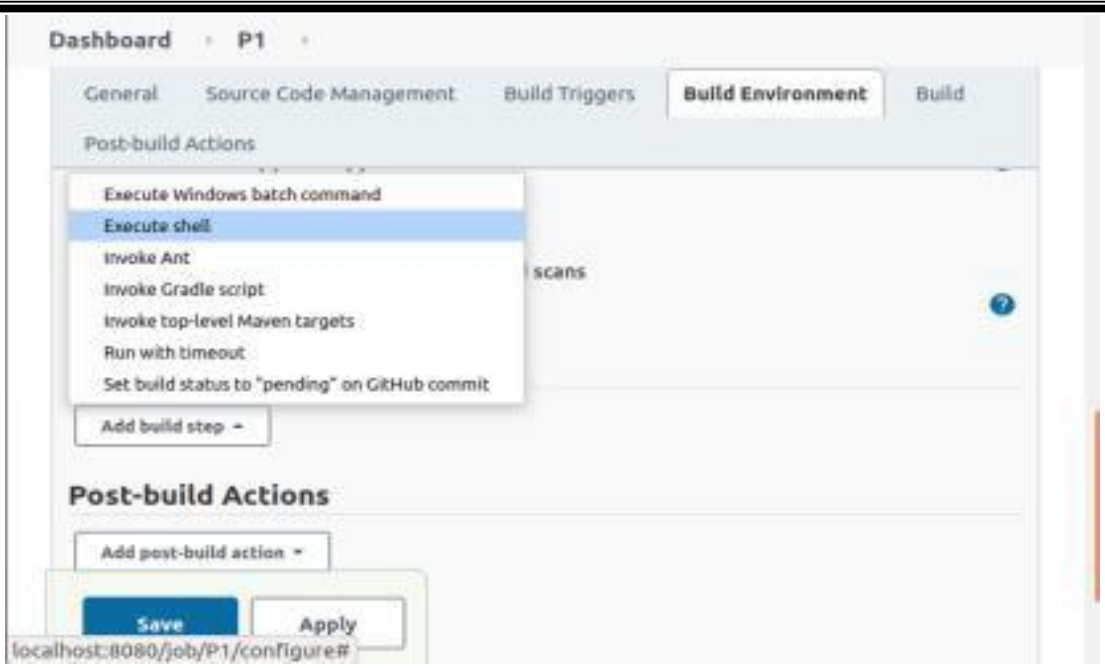
TO PRACTISE/EXECUTE SHELL PROGRAMS USING JENKINS

Step 1 : Click on Create new jobs

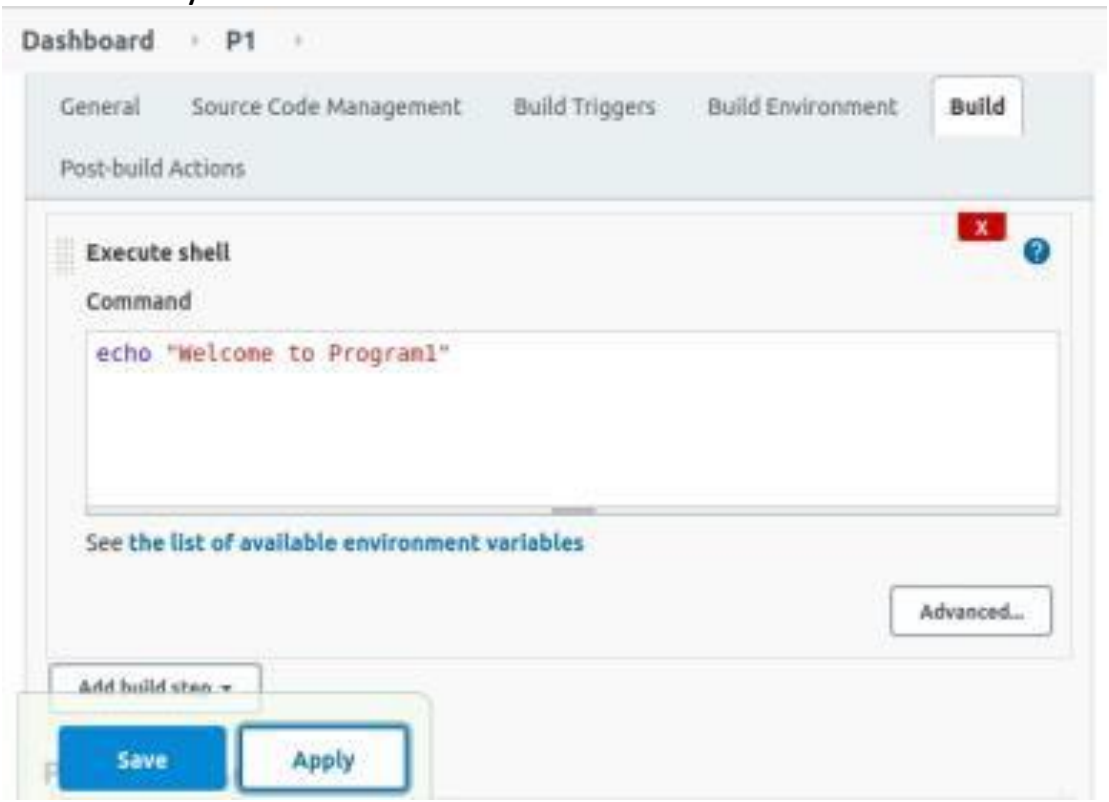
Step 2 : Give a name to project as “P1”, select Option “Free style project” and click on OK button



Step 3 : To run simple shell scripts on Jenkins click on Build option select the Execute script from dropdown menu



Step 4 : Write a simple shell command and click on apply followed by save button

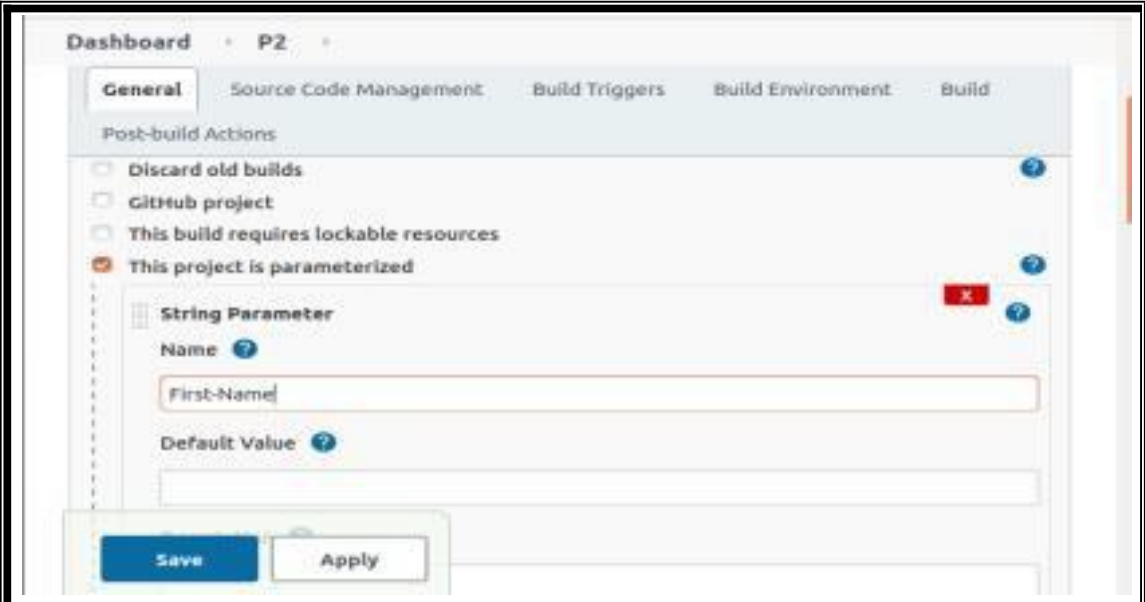


Step 5 : Click on first build “1” followed by console output to see the output

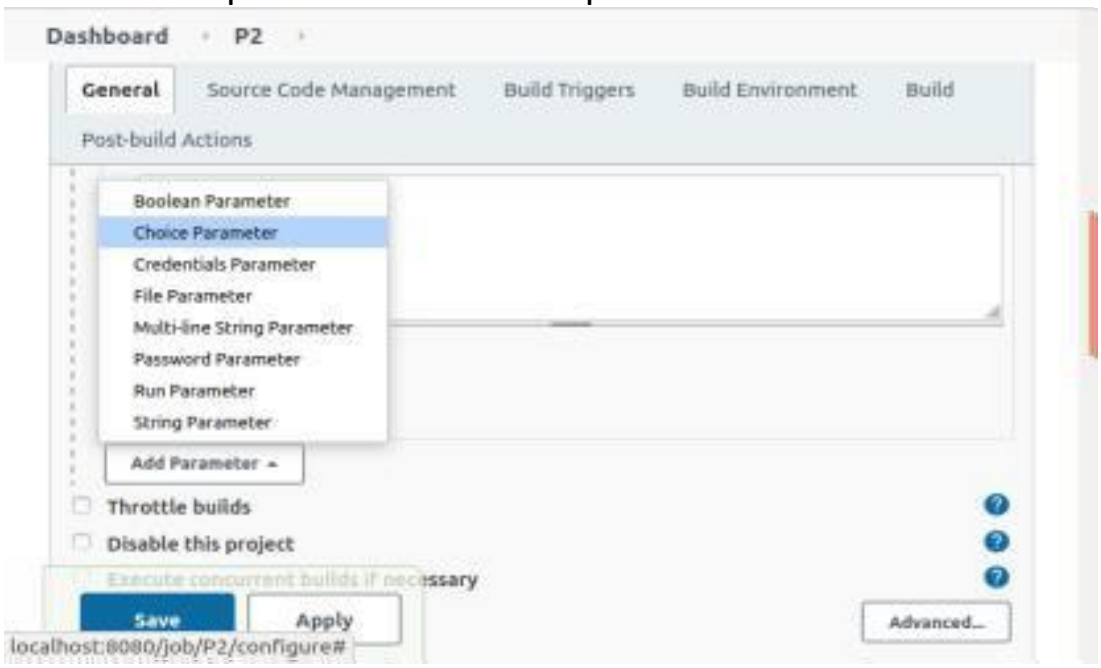
PRACTISE/EXECUTE PARAMETERISED JAVA PROGRAMS USING JENKINS

Step 1 : Create a freestyle project P2 in Jenkins

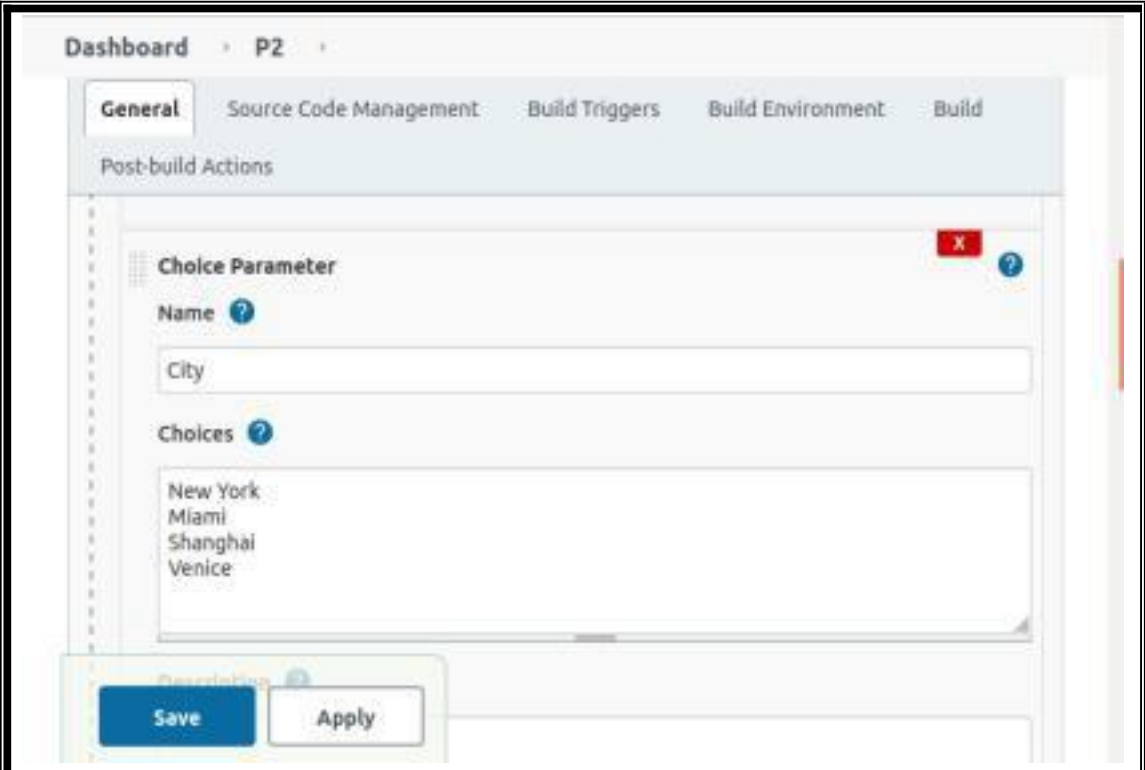
Step 2 : Click on general menu and select option this project is parameterize. Select String parameter and specify name as “First-Name”



Step 3 : Click on add parameter and select choice parameter. Take second parameter as choice parameter



Step 4 : Specify name as “City” and add the choices in each line



Step 5 : Click on build with parameters and specify the values

Console Output

```
Started by user Nikita Ravindra Patil
Running as SYSTEM
Building in workspace C:\Users\Nikita\.jenkins\workspace\P2
Finished: SUCCESS
```

CONCLUSION :- Hence we can conclude that we have learned and implemented shell programs and parametrized Java programs using Jenkins.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 4

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM :- To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job.

THEORY :- DEVOPS practices reflect the idea of continuous improvement and automation. Many practices focus on one or more development cycle phases. These practices include:

Continuous development:-This practice spans the planning and coding phases of the DevOps lifecycle. Version-control mechanisms might be involved.

Continuous testing:-This practice incorporates automated, prescheduled, continued code tests as application code is being written or updated. Such tests can speed the delivery of code to production.

Continuous integration (CI):- This practice brings configuration management (CM) tools together with other test and development tools to track how much of the code being developed is ready for production. It involves rapid feedback between testing and development to quickly identify and resolve code issues.

Continuous delivery:-This practice automates the delivery of code changes, after testing, to a preproduction or staging environment. An staff member might then decide to promote such code changes into production.

Continuous deployment (CD):- Similar to continuous delivery, this practice automates the release of new or changed code into production. A company doing continuous deployment might release code or feature changes several times per day. The use of container technologies, such as Docker and Kubernetes, can enable continuous deployment by helping to maintain consistency of the code across different deployment platforms and environments.

Continuous monitoring:-This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it. A feedback loop that reports on bugs or issues then makes its way back to development.

Infrastructure as code:-This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release. Developers add infrastructure “code” from within their existing development tools. For example, developers might create a storage volume on demand from Docker, Kubernetes, or OpenShift. This practice also allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations.

Jenkins is an open source automation server. With Jenkins, organizations can accelerate the software development process by automating it. Jenkins manages and controls software delivery processes throughout the entire lifecycle, including build, document, test, package, stage, deployment, static code analysis and much more. You can set up Jenkins to watch for any code changes in places like GitHub, Bitbucket or GitLab and automatically do a build a with tools like Maven and Gradle. You can utilize container technology such as Docker and Kubernetes, initiate tests and then take actions like rolling back or rolling forward in production.

OUTPUT :

JENKINS INSTALLATION WINDOWS

The simplest way to install Jenkins on Windows is to use the Jenkins Windows installer. That program will install Jenkins as a service using a 64 bit JVM chosen by the user. Keep in mind that to run Jenkins as a service, the account that runs Jenkins must have permission to login as a service.

Step 1: Setup wizard

Step 2: Select destination folder

Step 3: Service logon credentials

Step 4: Port selection

Step 5: Select Java home directory

Step 6: Custom setup Step 7: Install Jenkins

Step 8: Finish Jenkins installation

Post-installation setup wizard

After downloading, installing and running Jenkins, the postinstallation setup wizard begins.

This setup wizard takes you through a few quick "one-off" steps to unlock Jenkins, customize it with plugins and create the first administrator user through which you can continue accessing Jenkins

Unlocking Jenkins

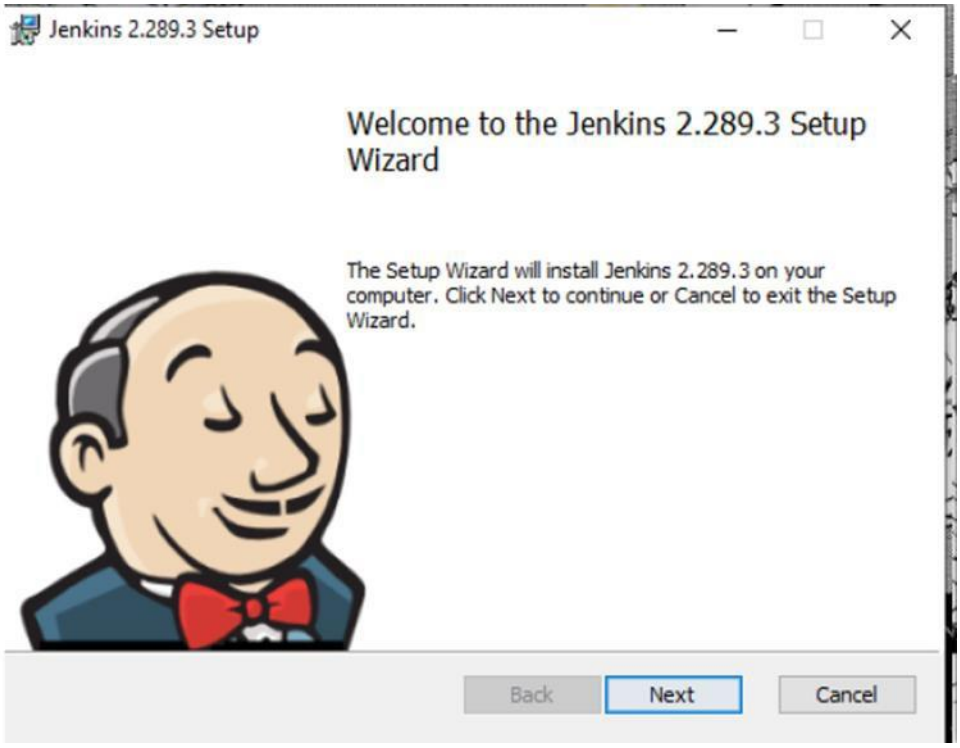
When you first access a new Jenkins instance, you are asked to unlock it using an automatically-generated password.

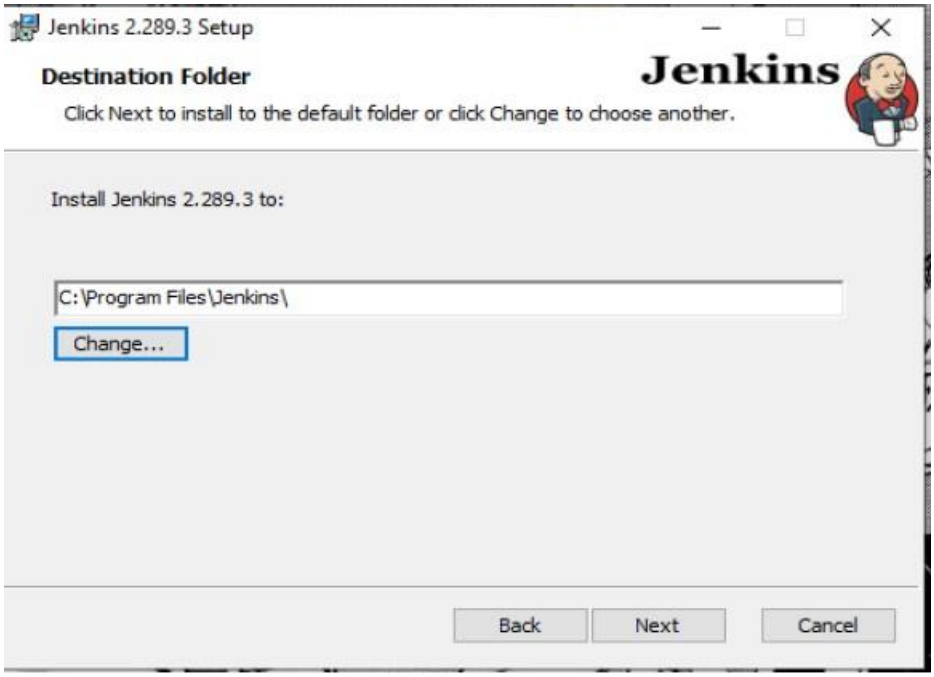
Browse to http://localhost:8080 (or whichever port you configured for Jenkins when installing it) and wait until the Unlock Jenkins page appears. The initial Administrator password should be found under the Jenkins installation path (set at Step 2 in Jenkins Installation).

MAVEN INSTALLATION

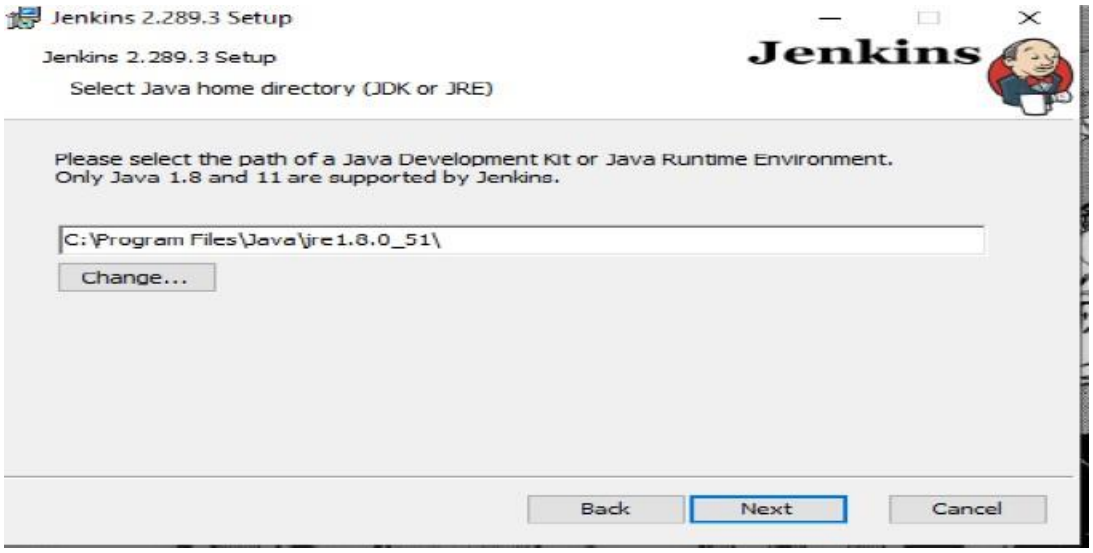
Maven is a powerful project management and comprehension tool that provides complete build life cycle framework to assist developers. It is based on the concept of a POM (Project Object Model) that includes project information and configuration information for Maven such as construction directory, source directory, test source directory, dependency, Goals, plugins etc.

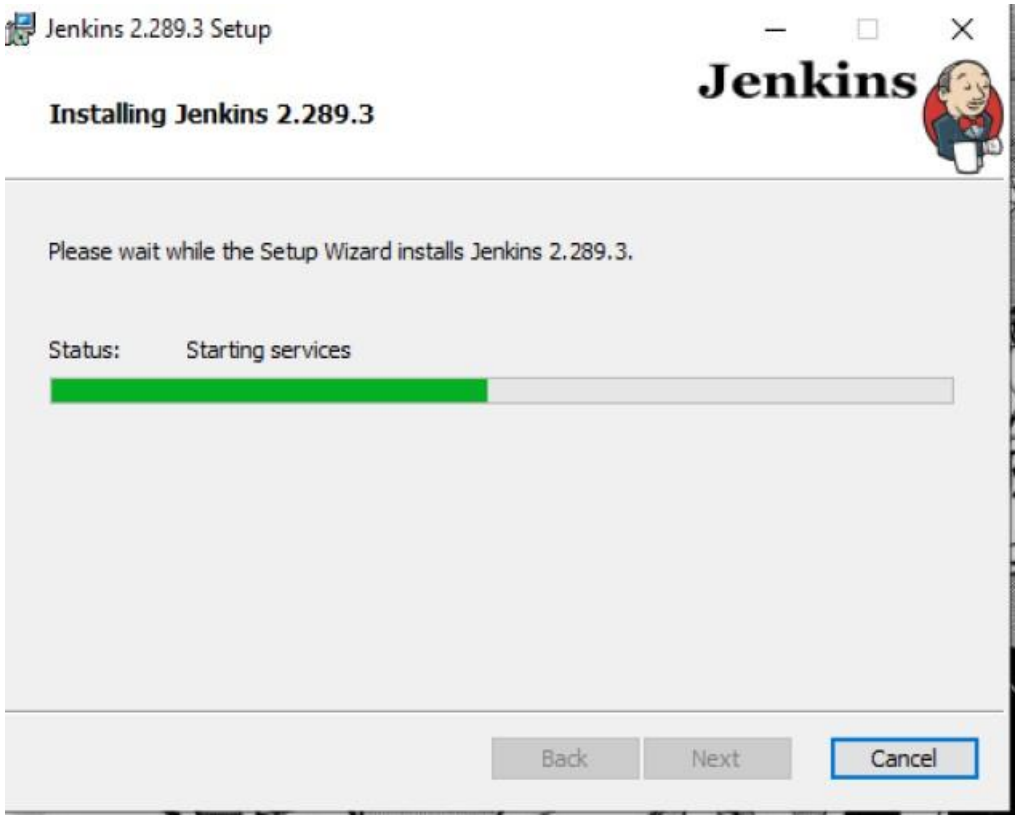
EXECUTION: JENKINS INSTALLATION AND SETUP

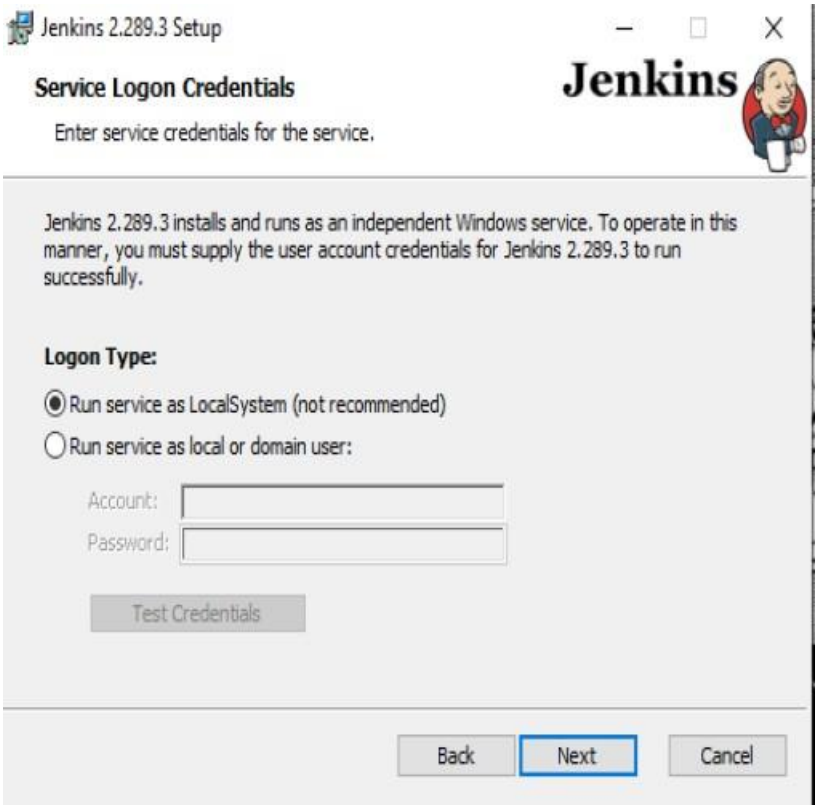


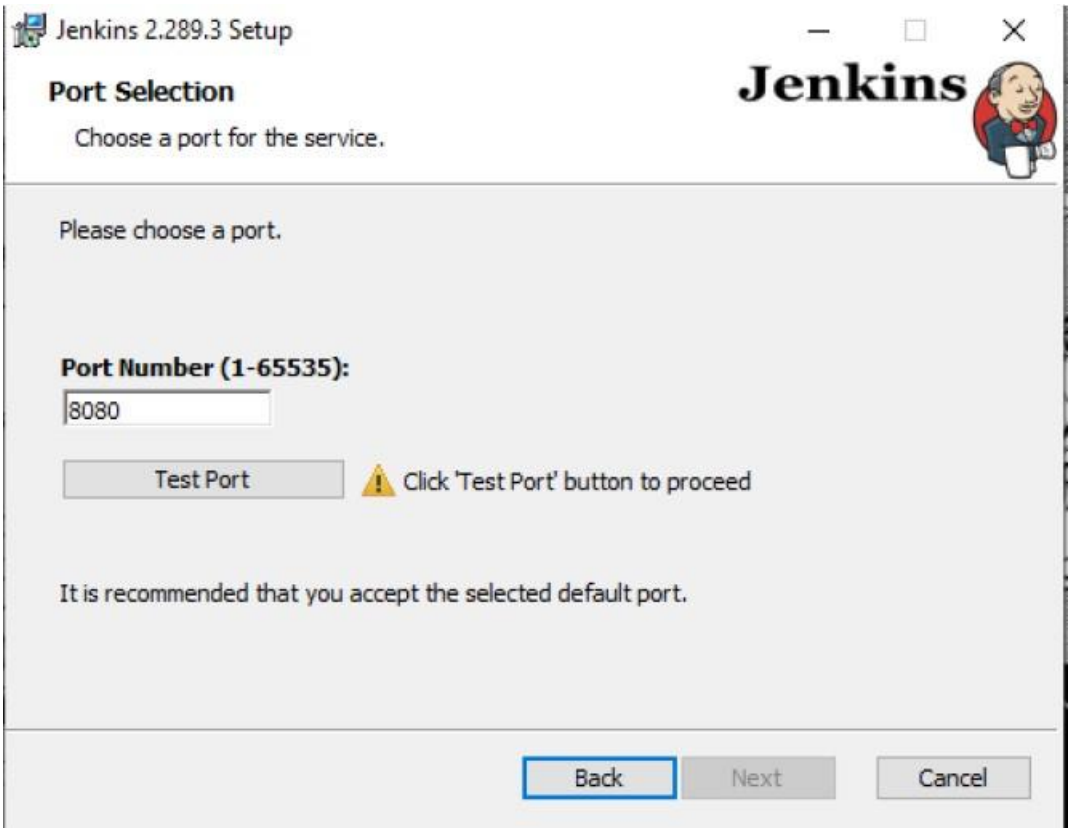

Jenkins 2.289.3 Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another.
Jenkins
Install Jenkins 2.289.3 to:
C:\Program Files\Jenkins\
Change...
Back
Next
Cancel


Jenkins 2.289.3 Setup
Completed the Jenkins 2.289.3 Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel

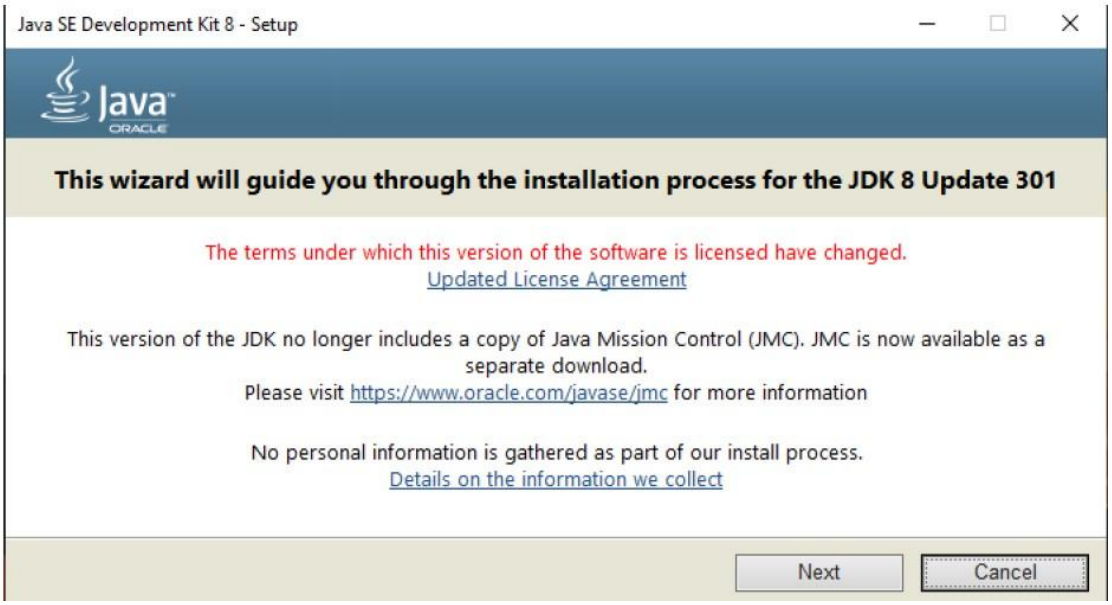

Jenkins 2.289.3 Setup
Jenkins 2.289.3 Setup
Select Java home directory (JDK or JRE)
Jenkins
Please select the path of a Java Development Kit or Java Runtime Environment.
Only Java 1.8 and 11 are supported by Jenkins.
C:\Program Files\Java\jre1.8.0_51\
Change...
Back
Next
Cancel

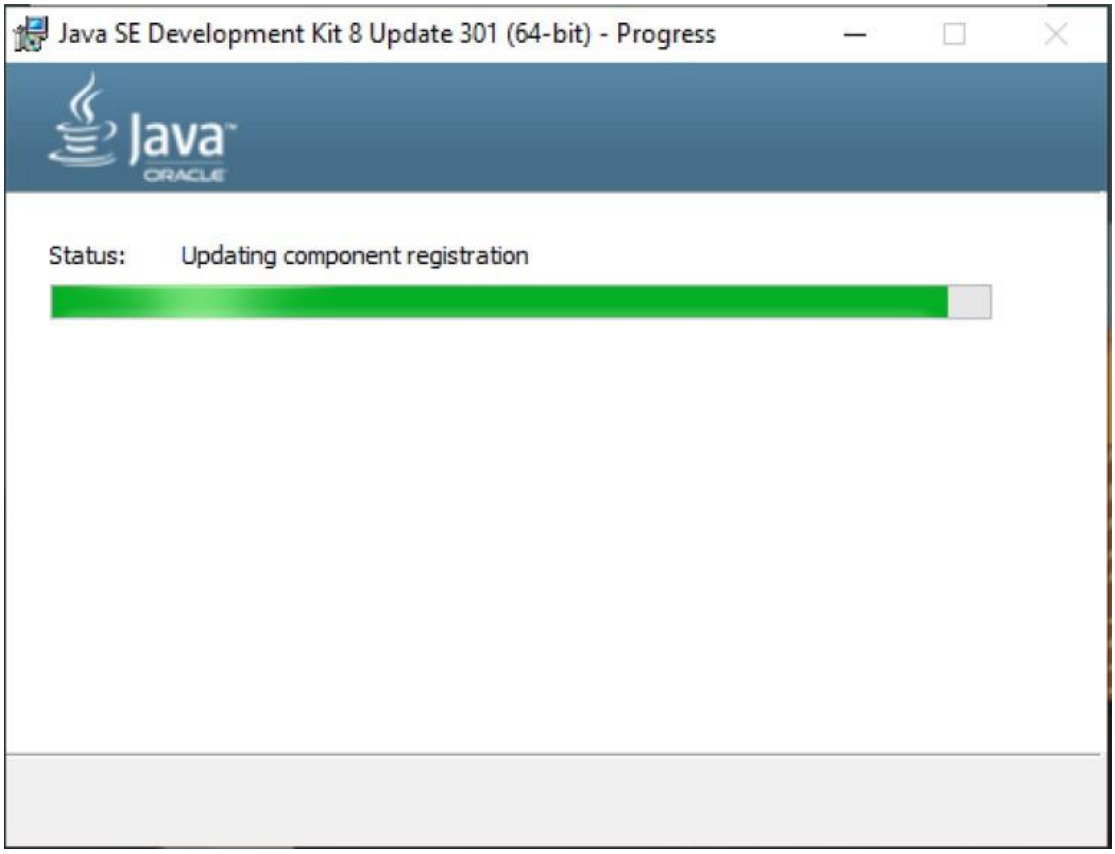

Jenkins 2.289.3 Setup
Jenkins
Installing Jenkins 2.289.3
Please wait while the Setup Wizard installs Jenkins 2.289.3.
Status: Starting services
Back
Next
Cancel

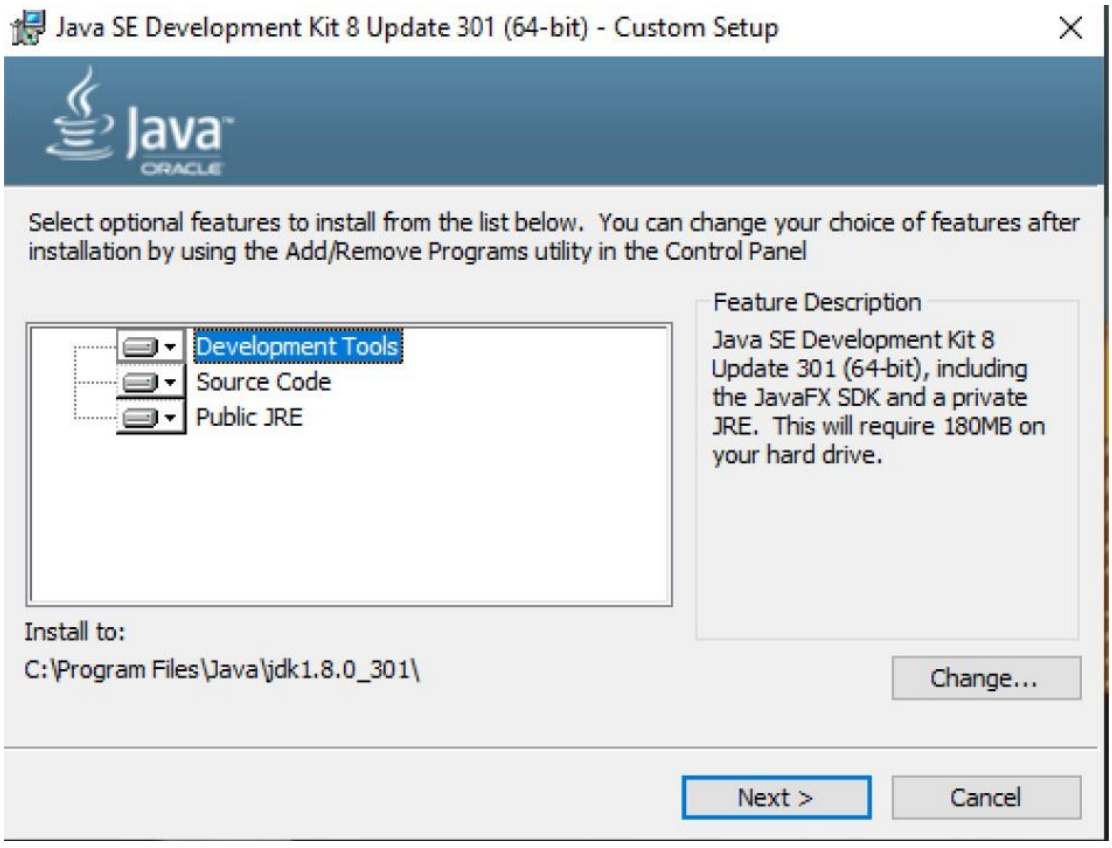

Jenkins 2.289.3 Setup
Service Logon Credentials
Enter service credentials for the service.
Jenkins
Jenkins 2.289.3 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.289.3 to run successfully.
Logon Type:
Run service as LocalSystem (not recommended)
Run service as local or domain user:
Account:
Password:
Test Credentials
Back
Next
Cancel


Jenkins 2.289.3 Setup
Port Selection
Choose a port for the service.
Jenkins
Please choose a port.
Port Number (1-65535):
8080
Test Port
Click 'Test Port' button to proceed
It is recommended that you accept the selected default port.
Back
Next
Cancel

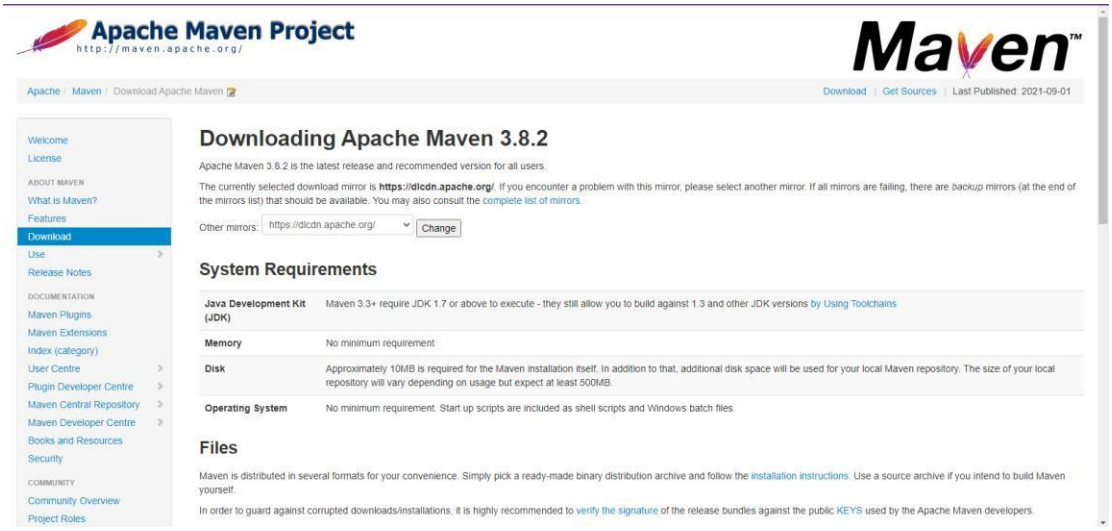
EXECUTION- JAVA JDK 8 KIT INSTALLATION

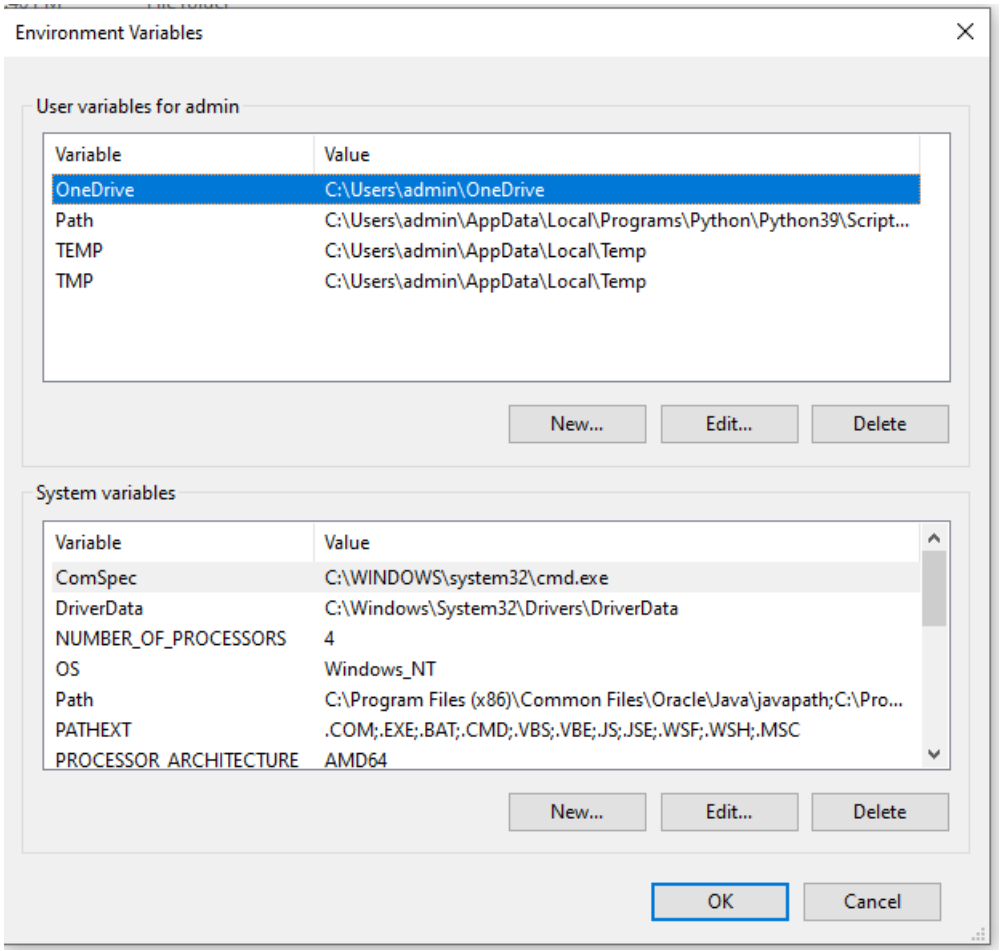


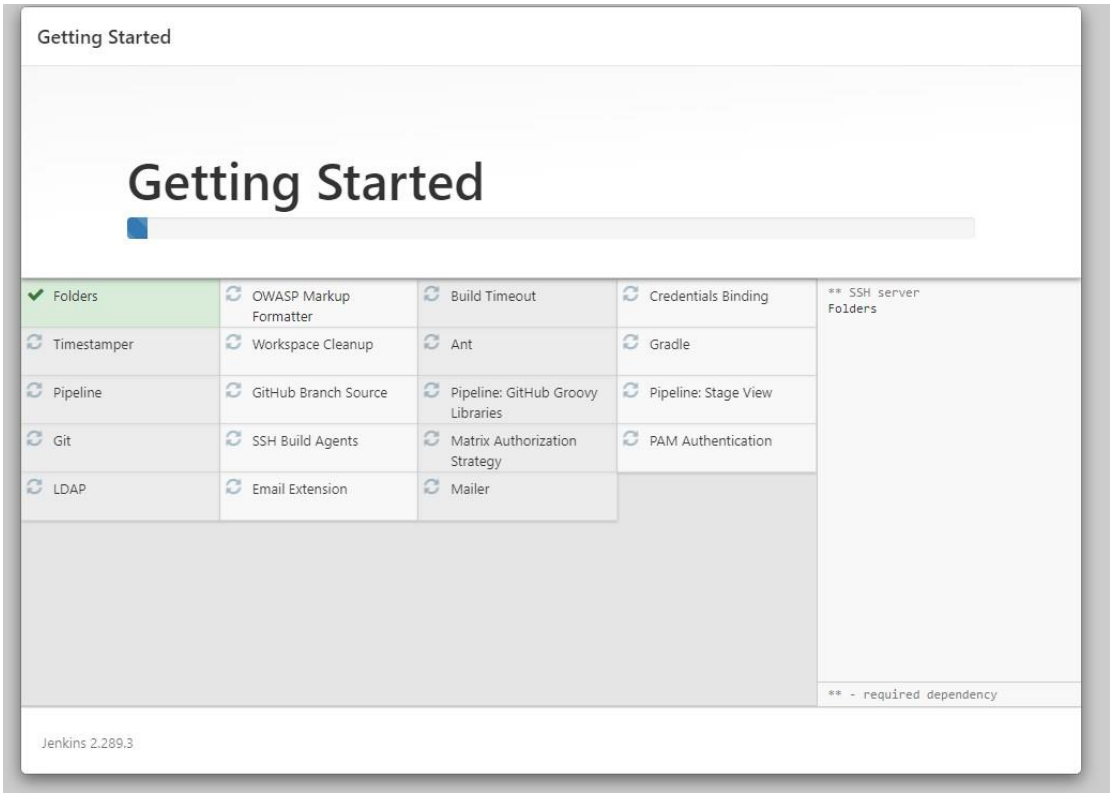

Java SE Development Kit 8 Update 301 (64-bit) - Progress
Java
ORACLE
Status:
Updating component registration

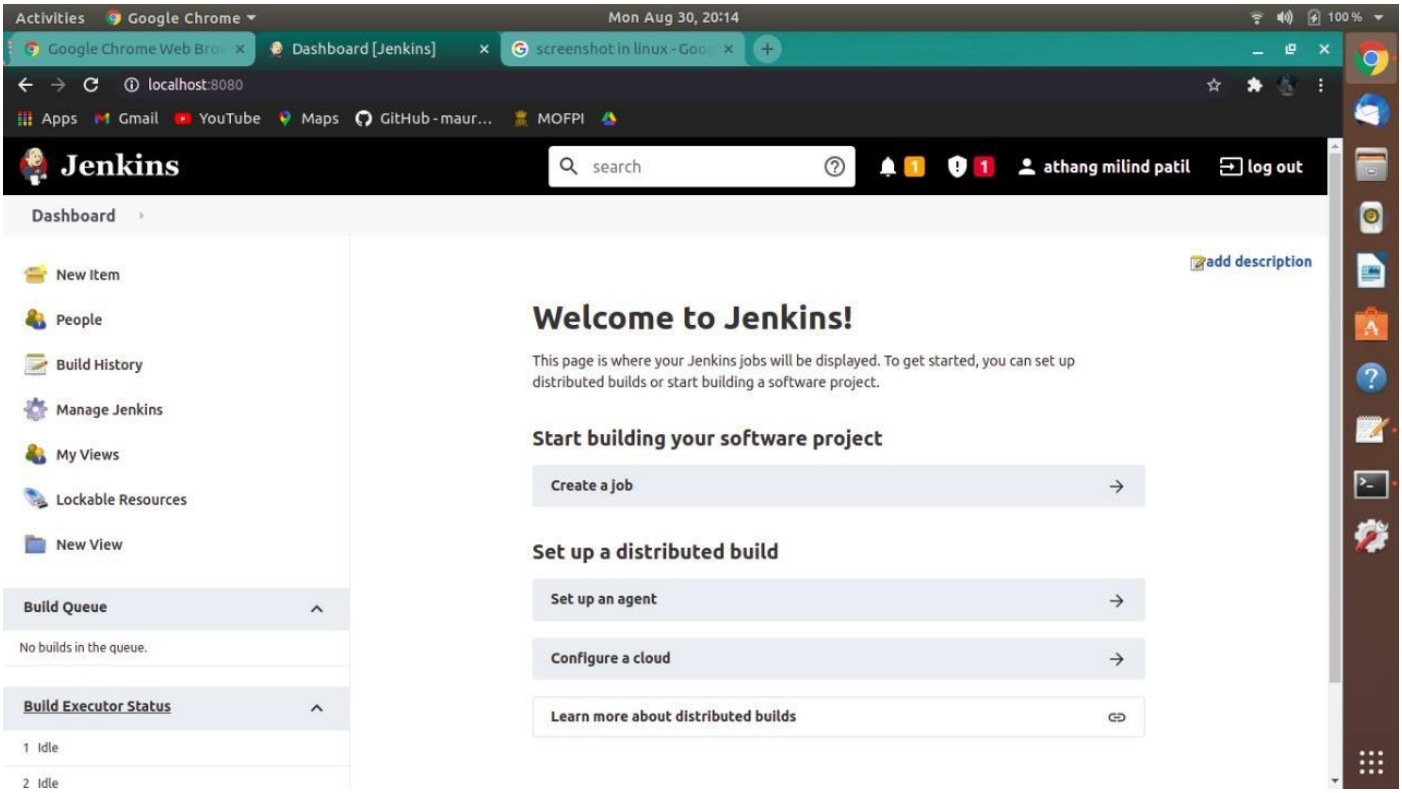

Java SE Development Kit 8 Update 301 (64-bit) - Custom Setup
Java
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Source Code
Public JRE
Feature Description
Java SE Development Kit 8 Update 301 (64-bit), including the JavaFX SDK and a private JRE. This will require 180MB on your hard drive.
Install to:
C:\Program Files\Java\jdk1.8.0_301\
Change...
Next >
Cancel

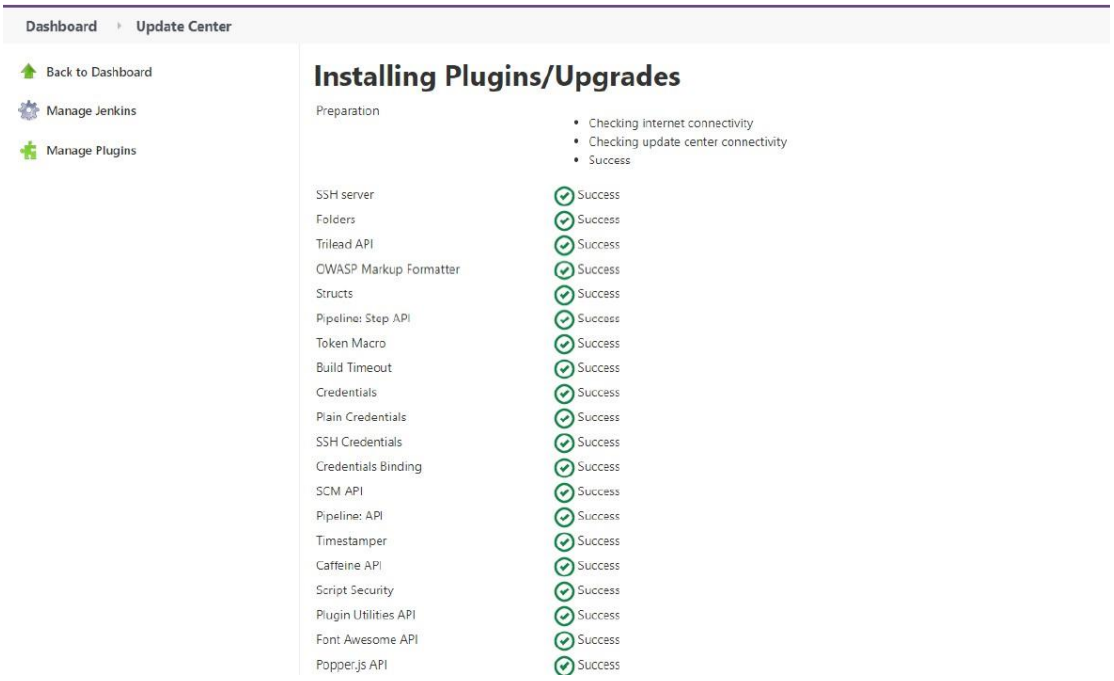
EXECUTION- MAVEN INSTALLATION AND SEETINT SET VARIABLES AND PATH

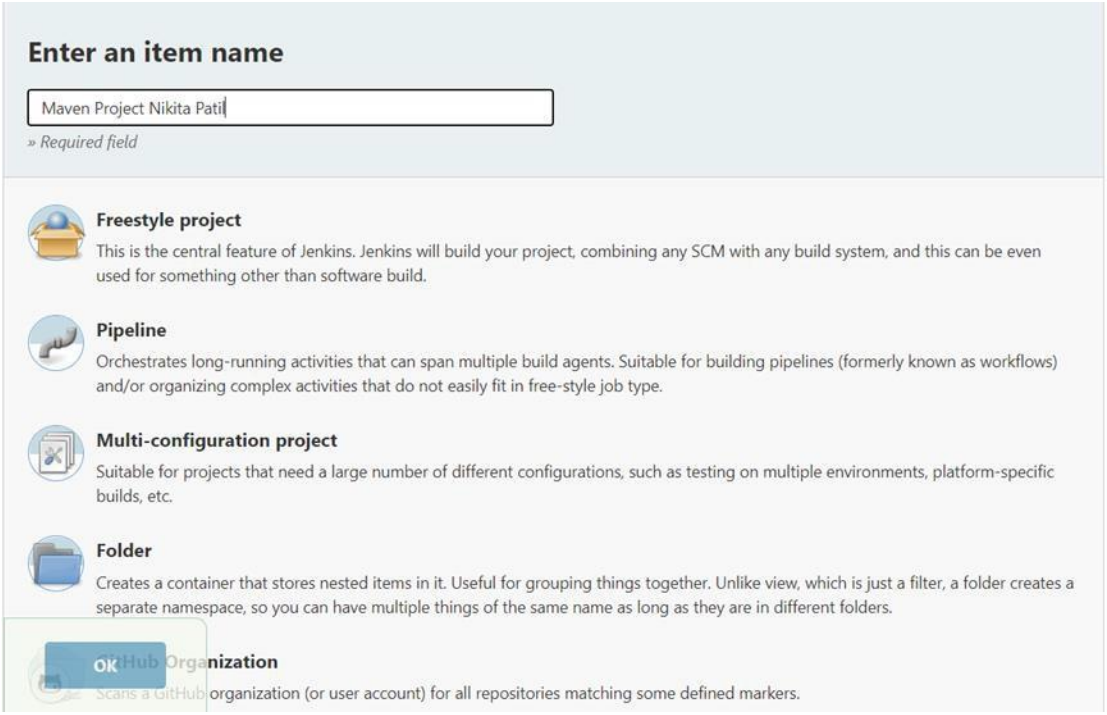


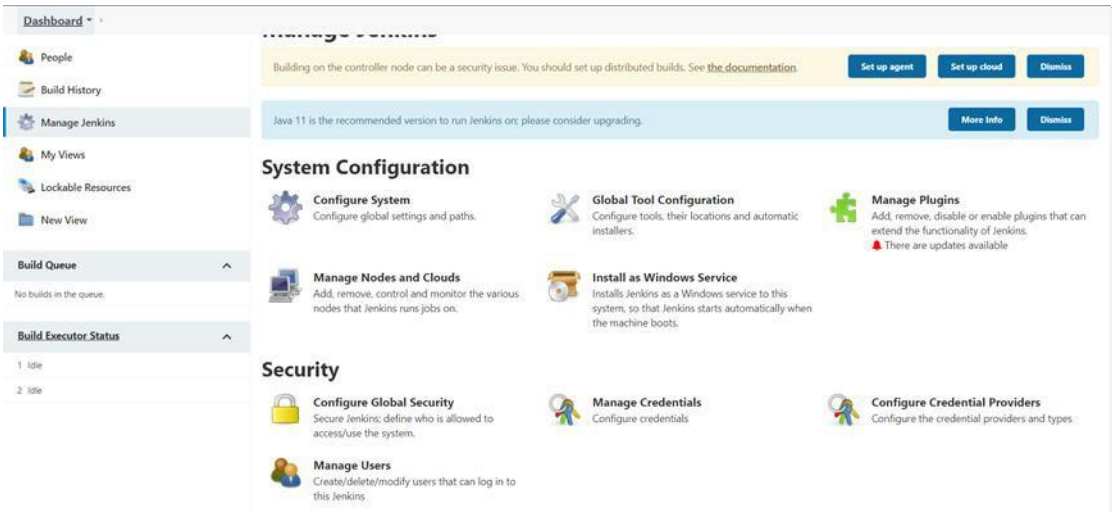



Getting Started
Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** SSH server
Folders
** - required dependency
Jenkins 2.289.3


Activities
Google Chrome
Mon Aug 30, 20:14
Dashboard [Jenkins]
localhost:8080
Apps
Gmail
YouTube
Maps
GitHub - maur...
MOFPI
Jenkins
search
athang milind patil
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Lockable Resources
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
add description
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
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds


Dashboard
Update Center
Back to Dashboard
Manage Jenkins
Manage Plugins
Installing Plugins/Upgrades
Preparation
Checking internet connectivity
Checking update center connectivity
Success
SSH server Success
Folders Success
Trilead API Success
OWASP Markup Formatter Success
Structs Success
Pipeline: Step API Success
Token Macro Success
Build Timeout Success
Credentials Success
Plain Credentials Success
SSH Credentials Success
Credentials Binding Success
SCM API Success
Pipeline: API Success
Timestamper Success
Caffeine API Success
Script Security Success
Plugin Utilities API Success
Font Awesome API Success
Popper.js API Success


Enter an item name
Maven Project Nikita Patil
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
OK
nization
organization (or user account) for all repositories matching some defined markers.



CONCLUSION :- Successfully Installed and setup Jenkins 2.303.1and integrated it with MAVEN 3.8.2 and created a Job.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 5

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM :- To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

THEORY :-

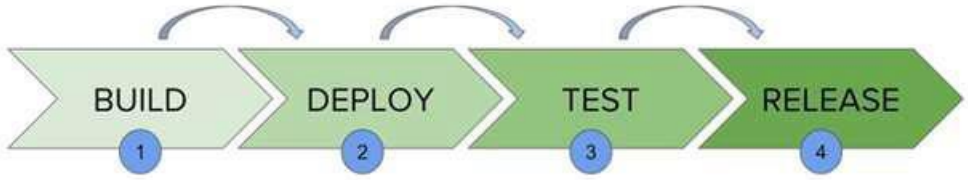
Jenkins Pipeline

In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.

It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.
In other words, a Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow. A pipeline has an extensible automation server for creating simple or even complex delivery pipelines "as code", via DSL (Domain-specific language).

Continuous Delivery Pipeline:

In a Jenkins Pipeline, every job has some sort of dependency on at least one or more jobs or events.



The above diagram represents a continuous delivery pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release. These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

A continuous delivery pipeline is an automated expression to show your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its manner to being released. It also involves developing the software in a repeatable and reliable manner, and progression of the built software through multiple stages of testing and deployment.

Jenkins - Build Jobs

a) Go to New Items->Give a project name in “Item name” field->select Maven project->click OK

b) Now configure the job

-Provide the description

-In Source Code Management, there are options for CVS project, Git etc, select the one which is required

– In Build Triggers, there are multiple options like “Build when a change

is pushed to GitHub", "Poll SCM", "Build whenever a SNAPSHOT dependency is built" etc, select the required one

– Give the path of your pom.xml file in Build Root POM

– Give "Goals and options" take a use case where the requirement is to install the code then give "clean install"

c) Configure the job as in the screenshot and don't forget to save

d) In the same way, create another job(4503_deploy)

e) Suppose the requirement is creating two jobs, one(4502_deploy) for deploying the code on author instance 4502 and if it's built is successful than it should run its downstream job(4503_deploy) for deploying code on publishing instance.

f) Now configure the job "4502_deploy", In Post Steps select "Run only if build succeeds"

g) In Post Build Action, add post build action and give the job name(4503_deploy) in "Projects to build

Step 6 – Build the jobs

Now build the job "4502_deploy", on successful completion "4503_deploy" will trigger automatically and hopefully, jobs will be successful!!!

Following these steps above, you will be able to configure your Maven project over Jenkins.

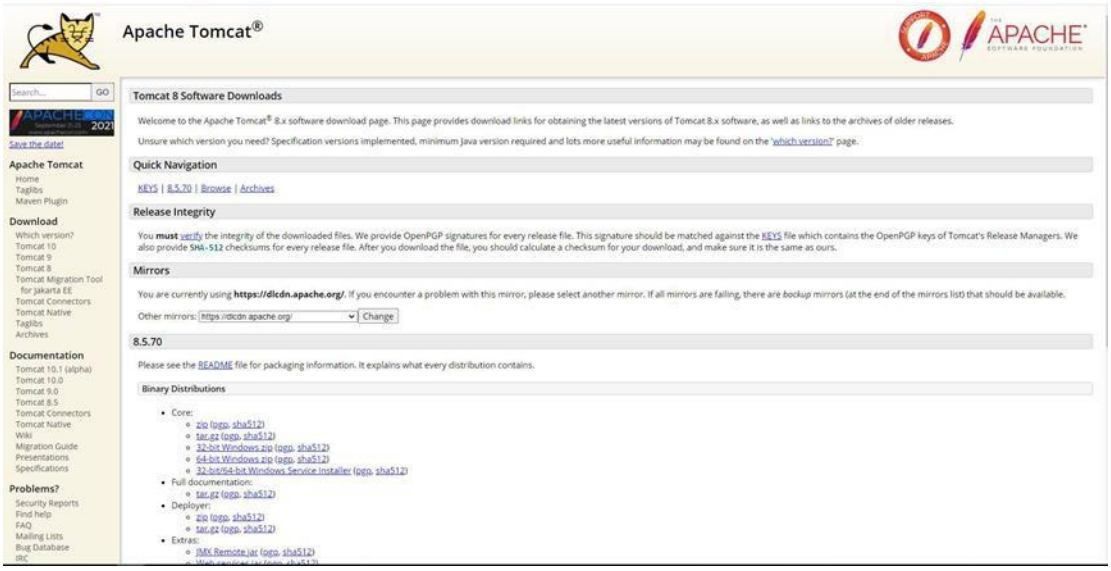
Tomcat Server :-

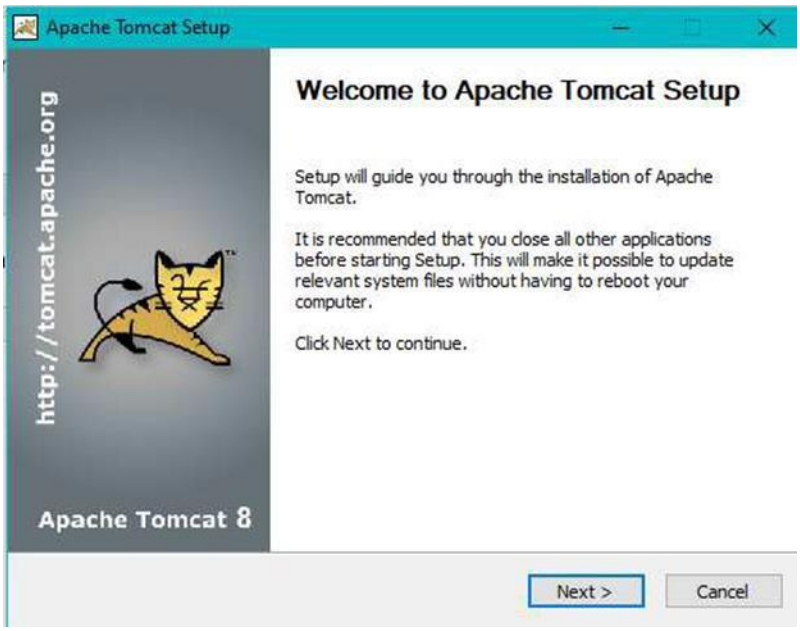
It is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages and last but not least, the Java Servlet. The complete name of Tomcat is "Apache Tomcat" it was developed in an open, participatory environment and released in 1998 for the very first time. It began as

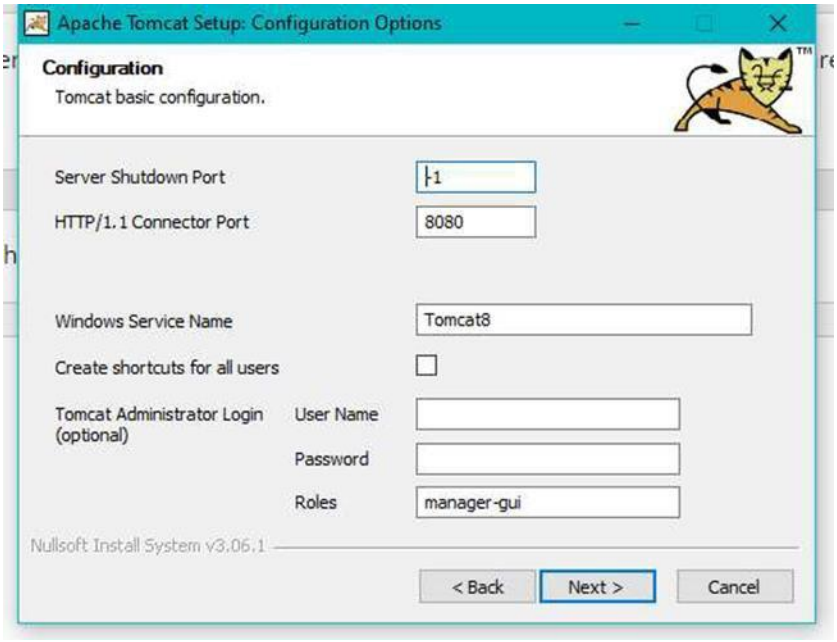
the reference implementation for the very first Java-Server Pages and the Java Servlet API. However, it no longer works as the reference implementation for both of these technologies, but it is considered as the first choice among the users even after that. It is still one of the most widely used java-sever due to several capabilities such as good extensibility, proven core engine, and well-test and durable. Here we used the term "servlet" many times, so what is java servlet; it is a kind of software that enables the webserver to handle the dynamic(javabased) content using the Http protocols.

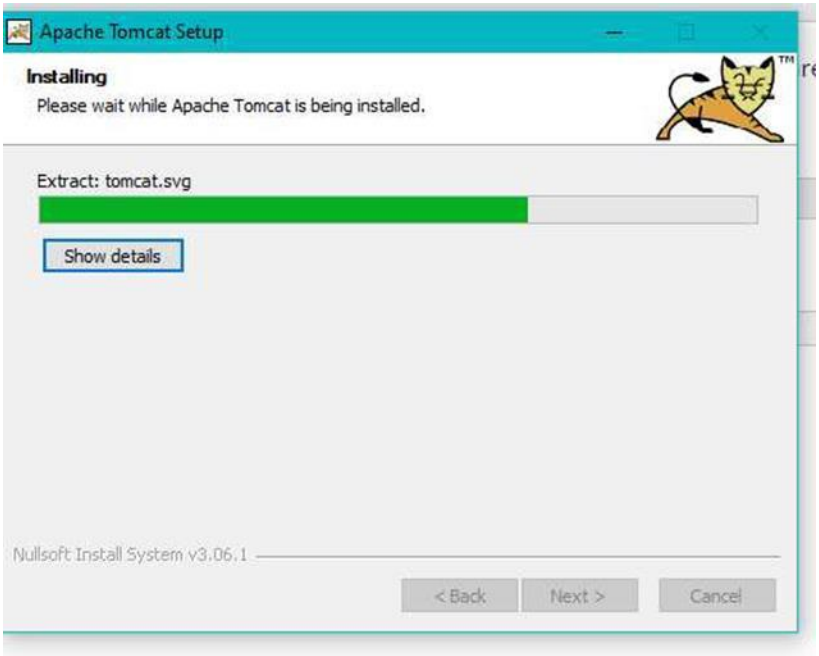
Deploying Steps :-

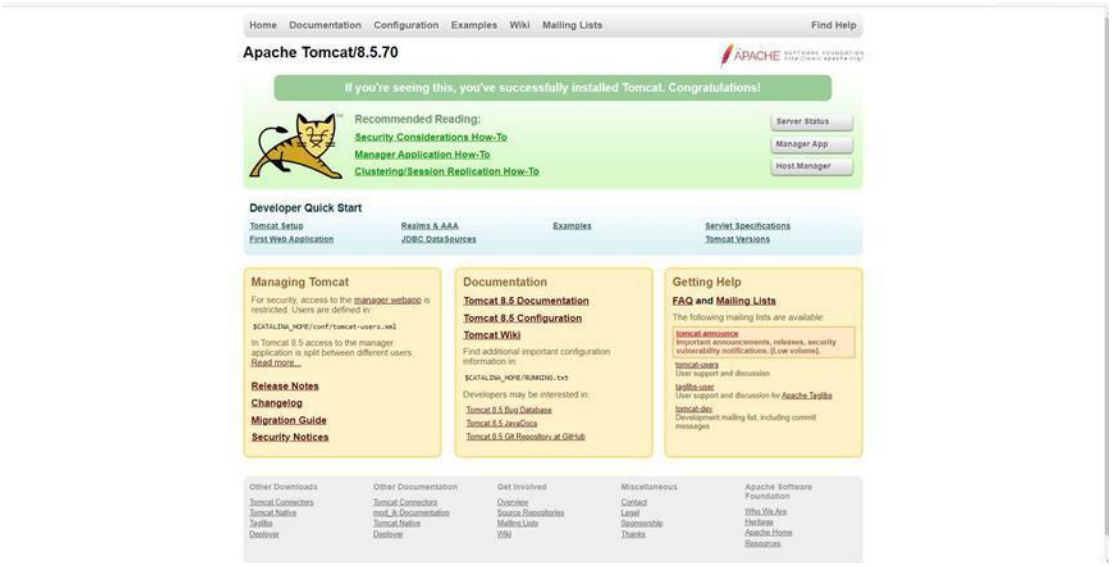
1. Install Deploy to Container Plugin.
2. Create and Configure a Maven Job with Source Code Management (Github)
3. Configure the Post-build Action and Specify the Tomcat Server Details
4. Build Jenkins Job
5. Testing the Application


Apache Tomcat®
Tomcat 8 Software Downloads
Welcome to the Apache Tomcat® 8.x software download page. This page provides download links for obtaining the latest versions of Tomcat 8.x software, as well as links to the archives of older releases.
Unsure which version you need? Specification versions implemented, minimum Java version required and lots more useful information may be found on the 'which version?' page.
Quick Navigation
KEYS | 8.5.70 | Browse | Archives
Release Integrity
You must verify the integrity of the downloaded files. We provide OpenPGP signatures for every release file. This signature should be matched against the KEYS file which contains the OpenPGP keys of Tomcat's Release Managers. We also provide SHA-512 checksums for every release file. After you download the file, you should calculate a checksum for your download, and make sure it is the same as ours.
Mirrors
You are currently using https://dlcdn.apache.org/. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are backup mirrors (at the end of the mirrors list) that should be available.
Other mirrors:
Change
8.5.70
Please see the README file for packaging information. It explains what every distribution contains.
Binary Distributions
Core:
zip (pgp, sha512)
tar.gz (pgp, sha512)
32-bit Windows zip (pgp, sha512)
64-bit Windows zip (pgp, sha512)
32-bit/64-bit Windows Service Installer (pgp, sha512)
Full documentation:
tar.gz (pgp, sha512)
Deployer:
zip (pgp, sha512)
tar.gz (pgp, sha512)
Extras:
JMX Remote jar (pgp, sha512)

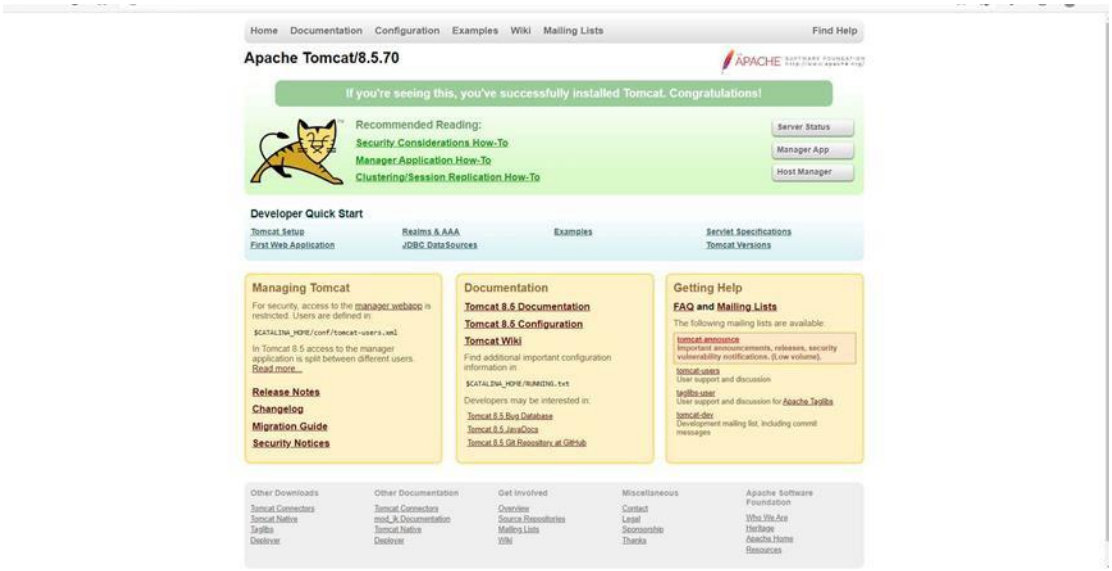

Apache Tomcat Setup
Welcome to Apache Tomcat Setup
Setup will guide you through the installation of Apache Tomcat.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
http://tomcat.apache.org
Apache Tomcat 8
Next >
Cancel

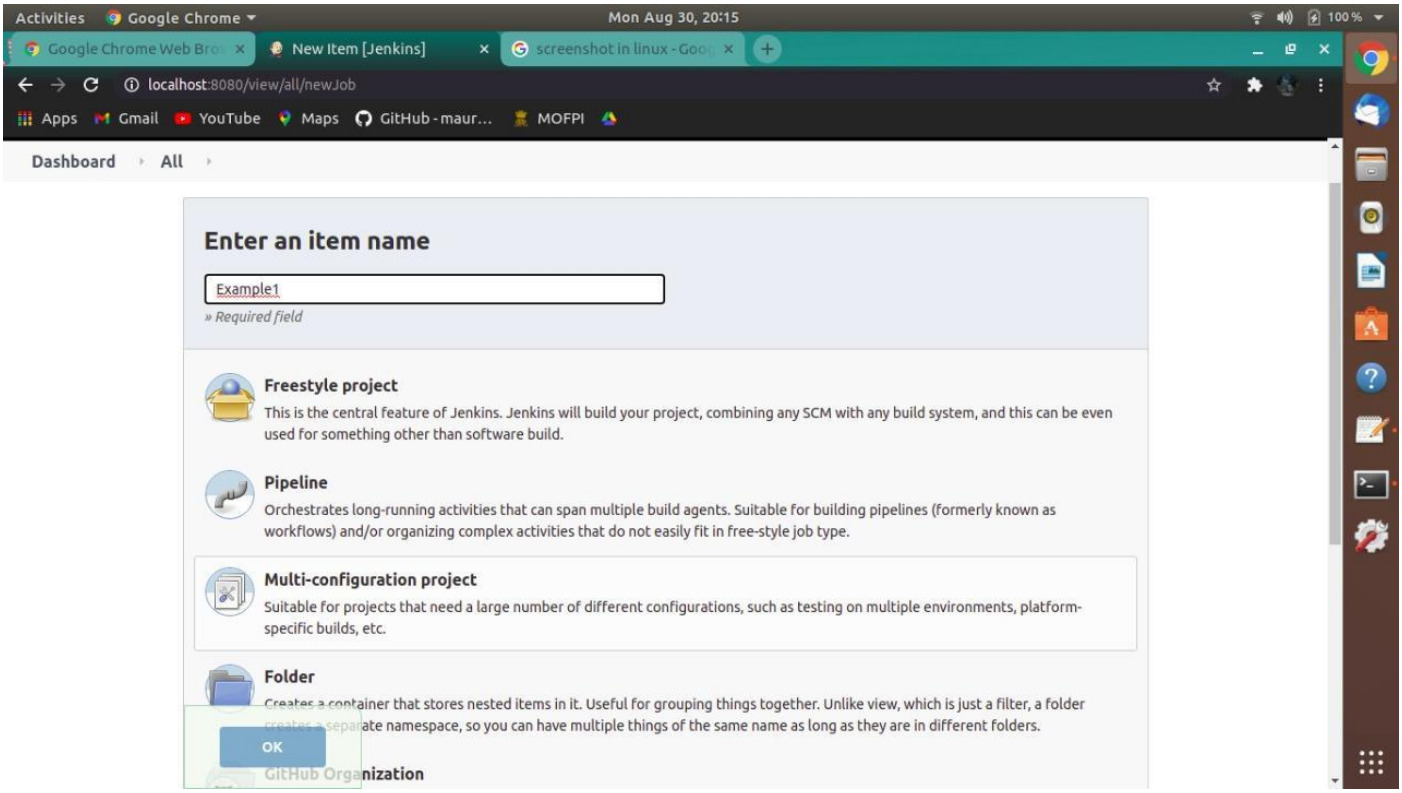

Apache Tomcat Setup: Configuration Options
Configuration
Tomcat basic configuration.
Server Shutdown Port
-1
HTTP/1.1 Connector Port
8080
Windows Service Name
Tomcat8
Create shortcuts for all users
Tomcat Administrator Login (optional)
User Name
Password
Roles
manager-gui
Nullsoft Install System v3.06.1
< Back
Next >
Cancel

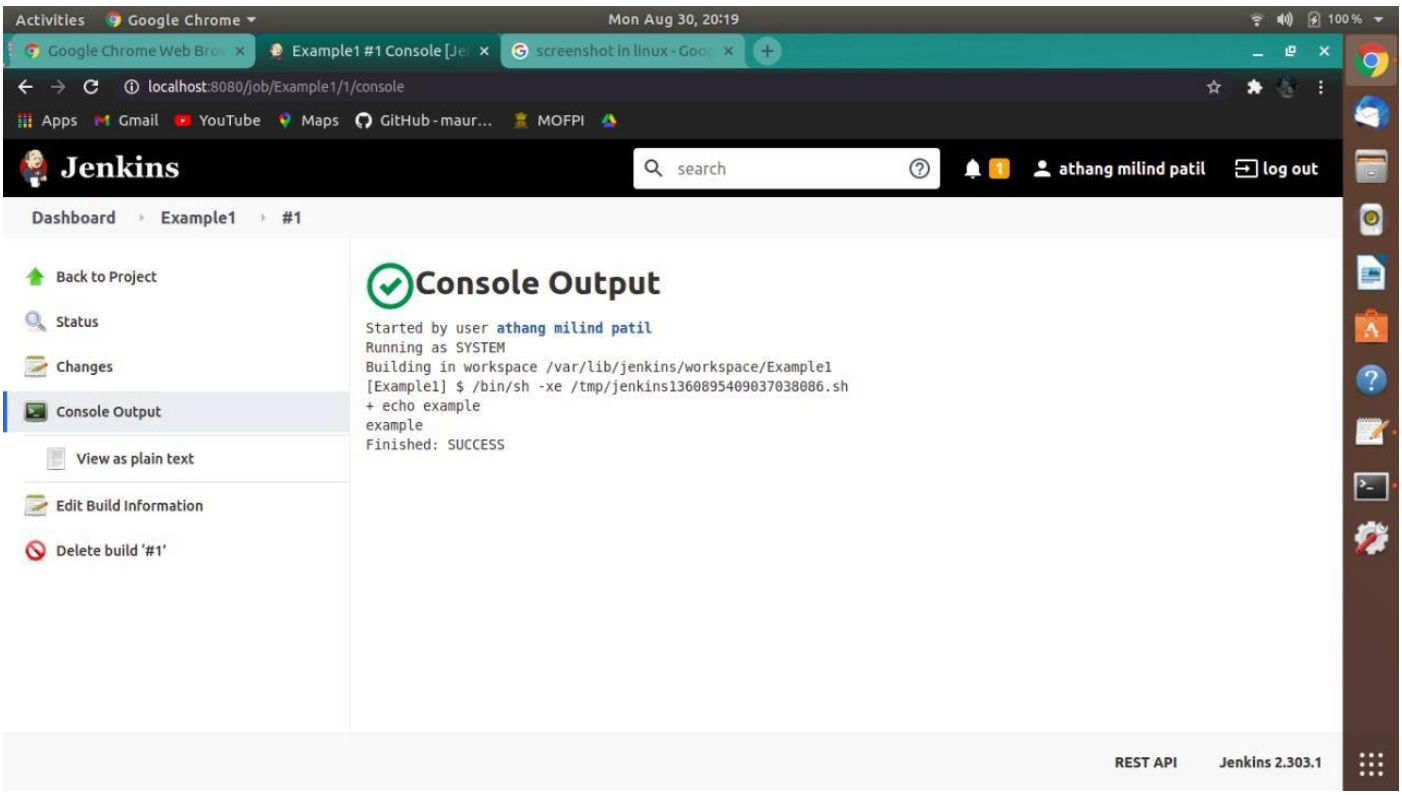

Apache Tomcat Setup
Installing
Please wait while Apache Tomcat is being installed.
Extract: tomcat.svg
Show details
Nullsoft Install System v3.06.1
< Back
Next >
Cancel

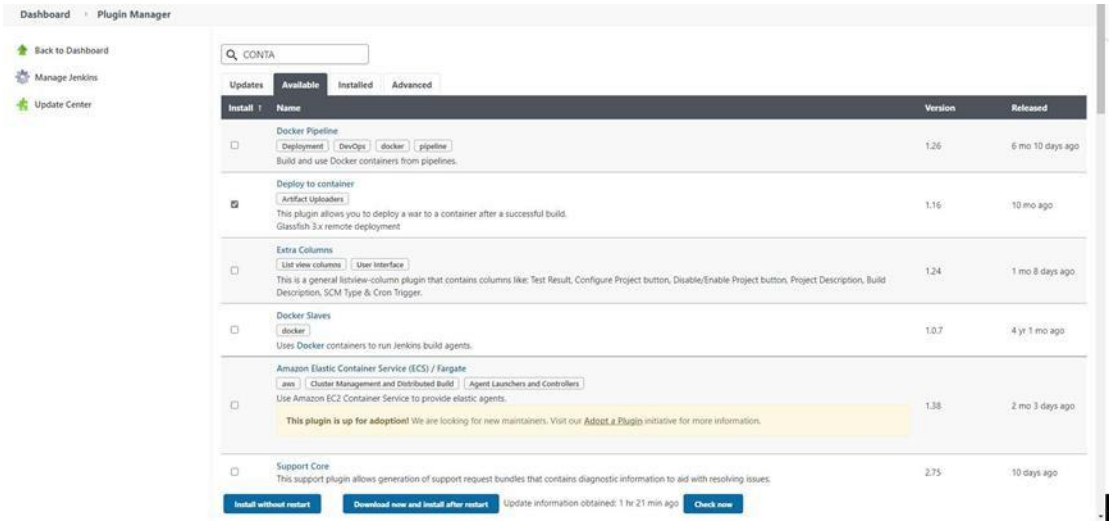


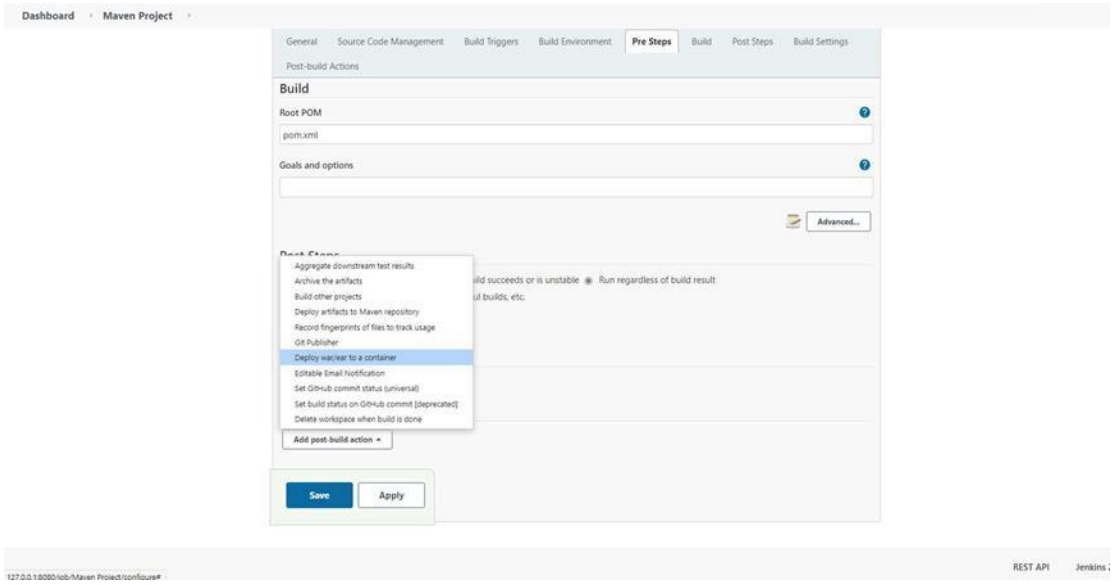
Execution - Setup Pipeline

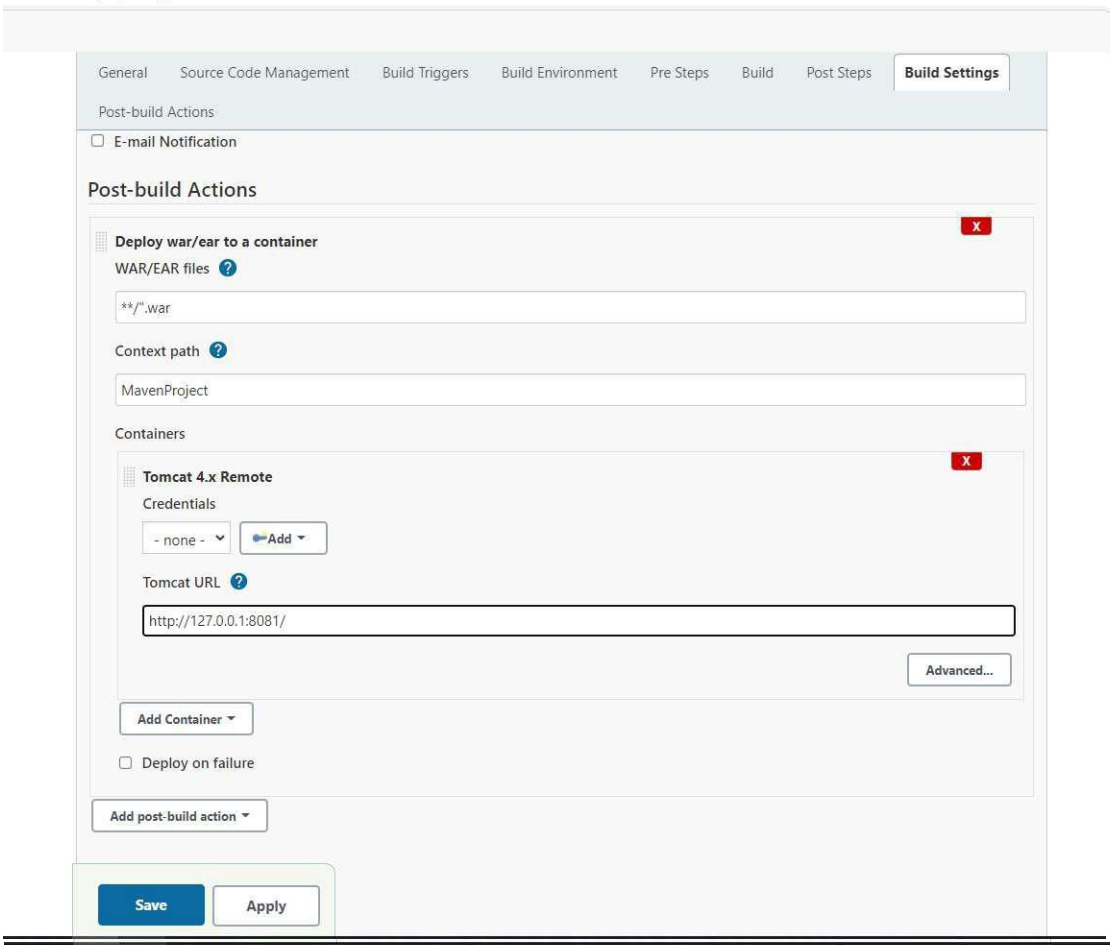



Activities
Google Chrome
Mon Aug 30, 20:15
New Item [Jenkins]
screenshot in linux - Goo
localhost:8080/view/all/newJob
Apps
Gmail
YouTube
Maps
GitHub - maur...
MOFPI
Dashboard
All
Enter an item name
Example1
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder
OK


Activities
Google Chrome
Mon Aug 30, 20:19
Example1 #1 Console [Je
screenshot in linux - Goo
localhost:8080/job/Example1/1/console
Apps
Gmail
YouTube
Maps
GitHub - maur...
MOFPI
Jenkins
search
athang milind patil
log out
Dashboard
Example1
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Console Output
Started by user athang milind patil
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Example1
[Example1] $ /bin/sh -xe /tmp/jenkins1360895409037038086.sh
+ echo example
example
Finished: SUCCESS
REST API
Jenkins 2.303.1


Dashboard › Plugin Manager
Back to Dashboard
Manage Jenkins
Update Center
CONTA
Updates
Available
Installed
Advanced
Install ↑
Name
Version
Released
Docker Pipeline
Deployment
DevOps
docker
pipeline
Build and use Docker containers from pipelines.
1.26
6 mo 10 days ago
Deploy to container
Artifact Uploaders
This plugin allows you to deploy a war to a container after a successful build.
Glassfish 3.x remote deployment
1.16
10 mo ago
Extra Columns
List view columns
User Interface
This is a general listview-column plugin that contains columns like: Test Result, Configure Project button, Disable/Enable Project button, Project Description, Build Description, SCM Type & Cron Trigger.
1.24
1 mo 8 days ago
Docker Slaves
docker
Uses Docker containers to run Jenkins build agents.
1.0.7
4 yr 1 mo ago
Amazon Elastic Container Service (ECS) / Fargate
aws
Cluster Management and Distributed Build
Agent Launchers and Controllers
Use Amazon EC2 Container Service to provide elastic agents.
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
1.38
2 mo 3 days ago
Support Core
This support plugin allows generation of support request bundles that contains diagnostic information to aid with resolving issues.
2.75
10 days ago
Install without restart
Download now and install after restart
Update information obtained: 1 hr 21 min ago
Check now


Dashboard › Maven Project
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build
Root POM
pom.xml
Goals and options
Advanced...
Aggregate downstream test results
Archive the artifacts
Build other projects
Deploy artifacts to Maven repository
Record fingerprints of files to track usage
Git Publisher
Deploy war/ear to a container
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
Add post-build action
Save
Apply
REST API
Jenkins



CONCLUSION :- Successfully Installed Tomcat Server and configured it and Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 6

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

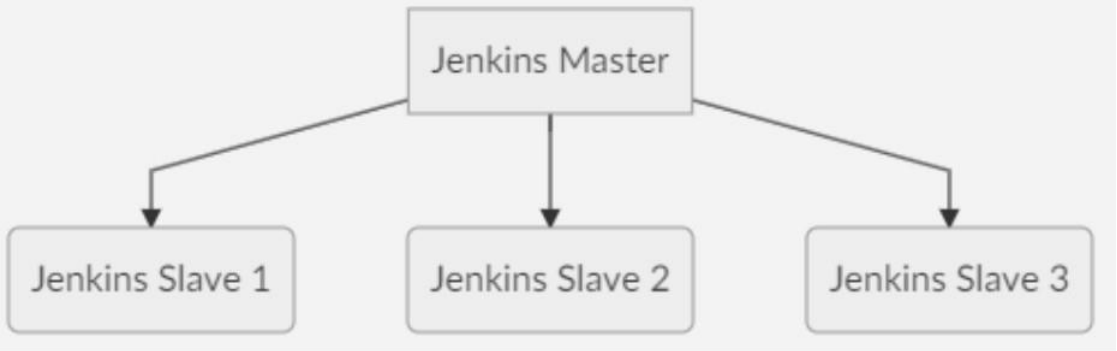
AIM : To understand Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

THEORY :-

Understanding the master and slave architecture

A standalone Jenkins instance can grow fairly quickly into a disk-munching, CPU-eating monster. To prevent this from happening, we can scale Jenkins by implementing a slave node architecture, which can help us offload some of the responsibilities of the master Jenkins instance. Let's clarify this concept. A Jenkins slave node is simply a device configured to act as an automation executor on behalf of the master. The Jenkins master simply represents the base installation of Jenkins. The master will continue to perform basic operations and serve the user interface, while the slaves do the heavy lifting.

This distributed computing model will allow the Jenkins master to remain responsive to users, while offloading automation execution to the connected slave(s). To illustrate the concept of a master, and slave mode architecture let's look at an example. Figure 2-1 shows a Jenkins master and three slave nodes of varying OS types:



The Jenkins master acts to schedule the jobs and assign slaves and send builds to slaves to execute the jobs.

It will also monitor the slave state (offline or online) and getting back the build result responses from slaves and the display build results on the console output. The workload of building jobs is delegated to multiple slaves.

Steps to Configure Jenkins Master and Slave Nodes

1) Click on Manage Jenkins in the left corner on the Jenkins dashboard.
2) Click on Manage Nodes.
3) Select New Node and enter the name of the node in the Node Name field.
4) Select Permanent Agent and click the OK button. Initially, you will get only one option, "Permanent Agent." Once you have one or more slaves you will get the "Copy Existing Node" option.
5) Enter the required information.
6) Enter the Hostname in the Host field.
7) Select the Add button to add credentials. and click Jenkins.
8) Enter Username, Password, ID, and Description.
9) Select the dropdown menu to add credentials in the Credentials field.
10) Select the next dropdown to add the Host Key Verification Strategy under Non verifying Verification Strategy.

11) Select Keep this agent online as much as possible in the Availability field.

Creating a Freestyle Project and Running on The Slave Machine

1) Click on **Save** and it will redirect to job's view page

2) On the left pane, click the **Build Now** button to execute your Pipeline**.**

3) We can verify the history of the executed build under the Build History by clicking the build number.

4) Click on the build number and select Console Output. Here you can see the executed job in the remote host and output.

Creating a Pipeline and Running on The Slave Machine

1) Click **New Item** in the top left corner on the dashboard.

2) Enter the name of your project in the **Enter an item name** field, and select the **Pipeline** project, and click **OK** button.

3) Enter **Description** (optional).

4) Go to the **Pipeline** section, make sure the **Definition** field has the **Pipeline script** option selected.

5) Copy and paste the following declarative Pipeline **script** into a script field.

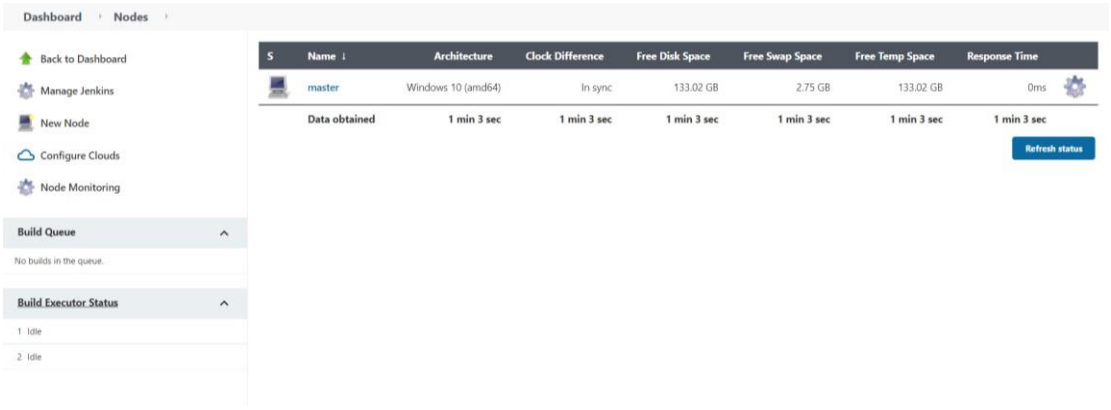
6) Click on **Save**, it will redirect to the Pipeline view page.

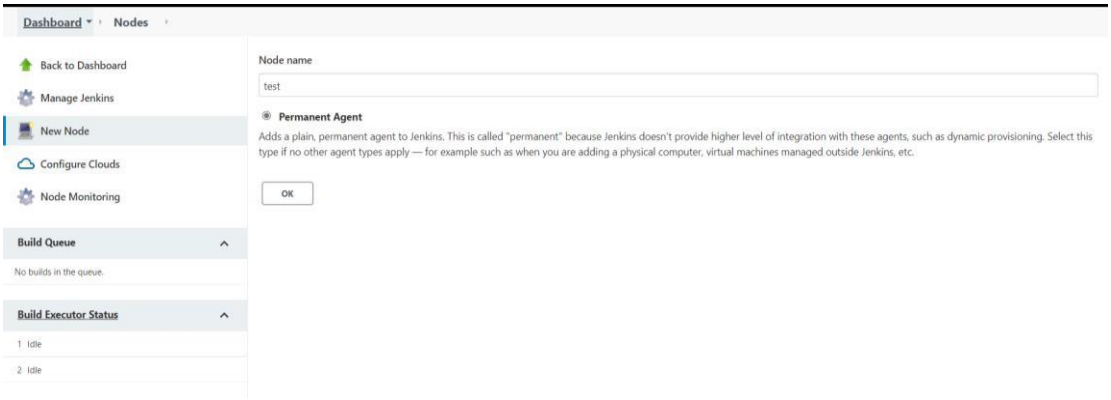
7) On the left pane, click the **Build Now** button to execute your Pipeline.

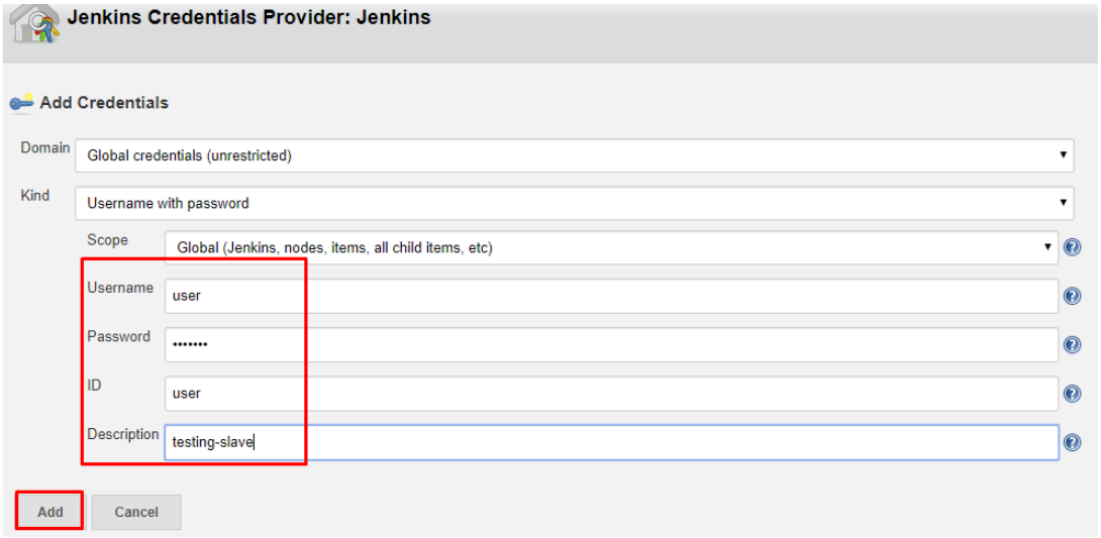
8) After Pipeline execution is completed, the Pipeline view will be as shown below.

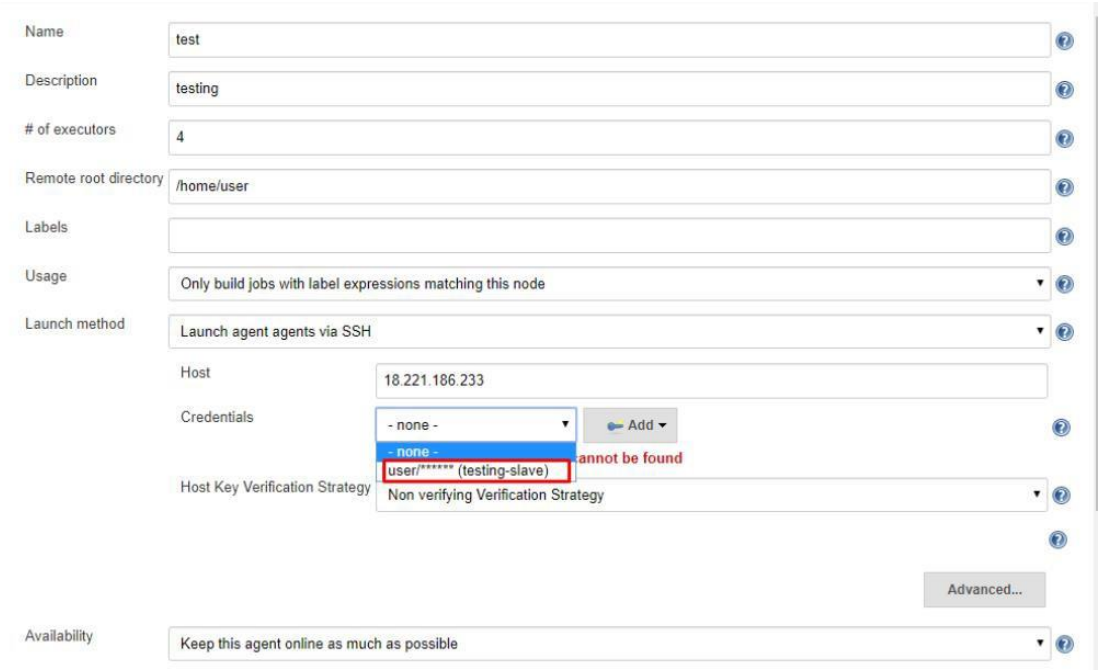
9) We can verify the history of executed build under the **Build History** by clicking the build number.

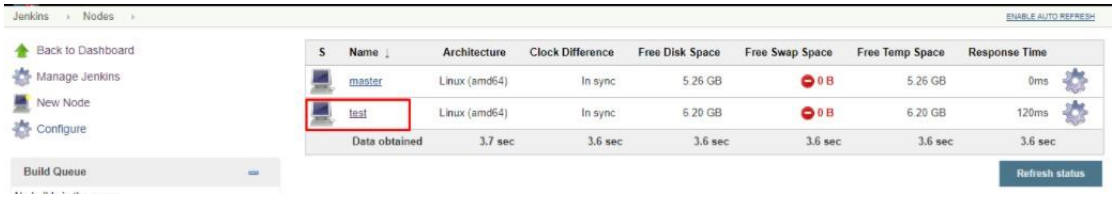
10) Click on build number and select **Console Output**. Here you can see that the pipeline ran on a slave machine**.**

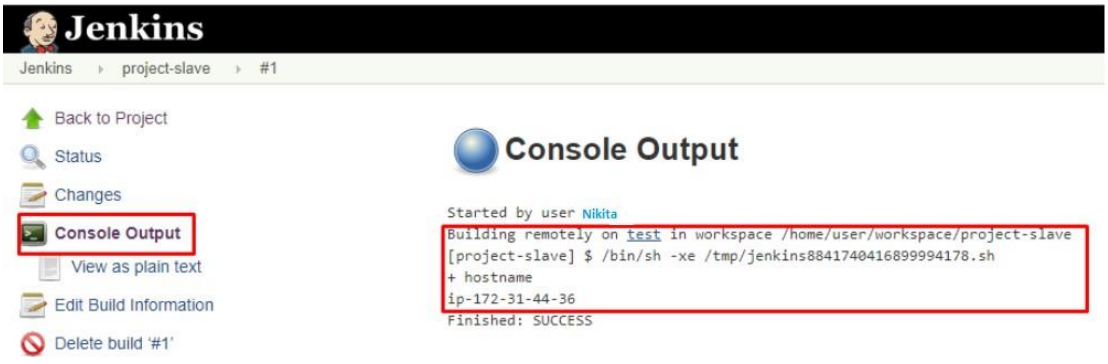

Dashboard
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
S
Name ↓
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
master
Windows 10 (amd64)
In sync
133.02 GB
2.75 GB
133.02 GB
0ms
Data obtained
1 min 3 sec
1 min 3 sec
1 min 3 sec
1 min 3 sec
1 min 3 sec
1 min 3 sec
Refresh status

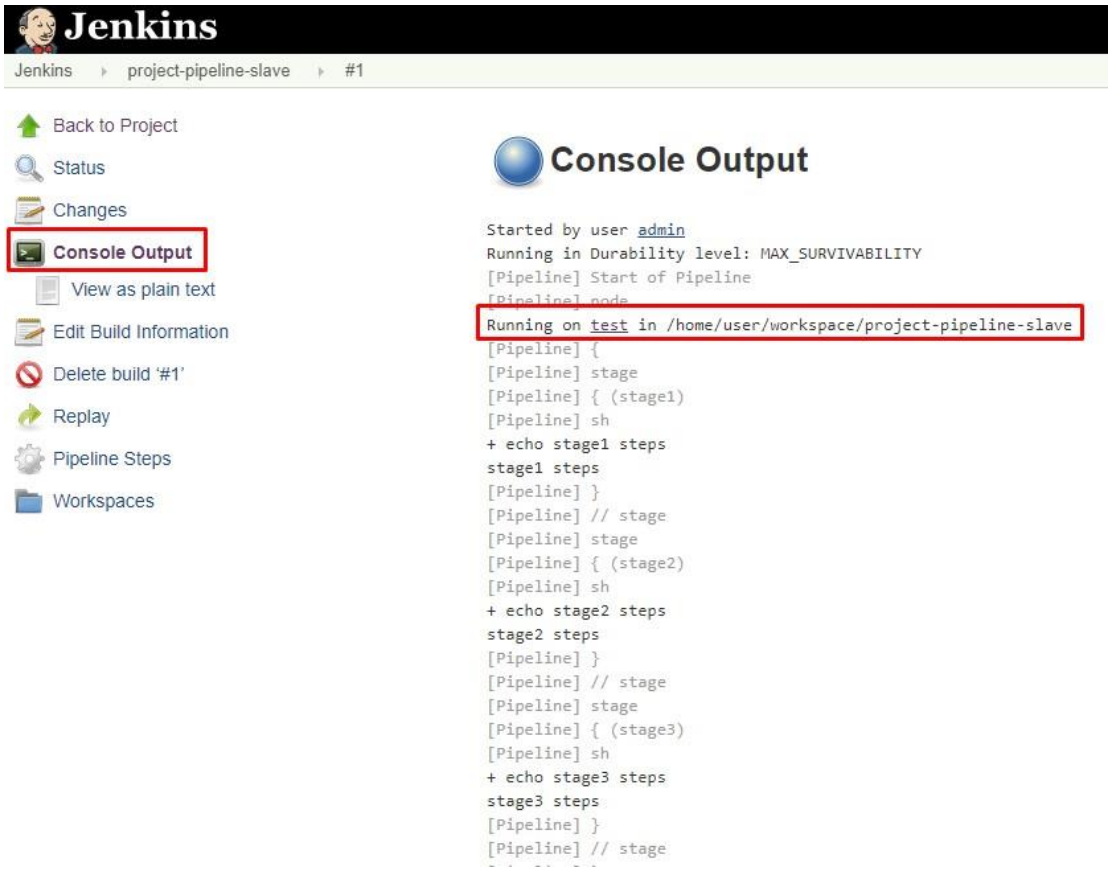

Dashboard
Nodes
Back to Dashboard
Manage Jenkins
New Node
Configure Clouds
Node Monitoring
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Node name
test
Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
OK


Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
Username with password
Scope
Global (Jenkins, nodes, items, all child items, etc)
Username
user
Password
•••••••
ID
user
Description
testing-slave
Add
Cancel


Name
test
Description
testing
# of executors
4
Remote root directory
/home/user
Labels
Usage
Only build jobs with label expressions matching this node
Launch method
Launch agent agents via SSH
Host
18.221.186.233
Credentials
- none -
Add
- none -
user/****** (testing-slave)
annot be found
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced...
Availability
Keep this agent online as much as possible


Jenkins
Nodes
ENABLE AUTO REFRESH
Back to Dashboard
Manage Jenkins
New Node
Configure
Build Queue
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
master
Linux (amd64)
In sync
5.26 GB
0 B
5.26 GB
0ms
test
Linux (amd64)
In sync
6.20 GB
0 B
6.20 GB
120ms
Data obtained
3.7 sec
3.6 sec
3.6 sec
3.6 sec
3.6 sec
3.6 sec
Refresh status


Jenkins
Jenkins
project-slave
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Console Output
Started by user Nikita
Building remotely on test in workspace /home/user/workspace/project-slave
[project-slave] $ /bin/sh -xe /tmp/jenkins8841740416899994178.sh
+ hostname
ip-172-31-44-36
Finished: SUCCESS



CONCLUSION :- Successfully understand abd implemented Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 7

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM :- To Setup and Run Selenium Tests in Jenkins Using Maven.

THEORY :-

Jenkins

Jenkins is the leading open-source continuous integration tool developed by Hudson lab. It is cross-platform and can be used on Windows, Linux, Mac OS and Solaris environments. Jenkins is written in Java.it has taken the place as one of the best opensource tools that allow continuous integration and build management

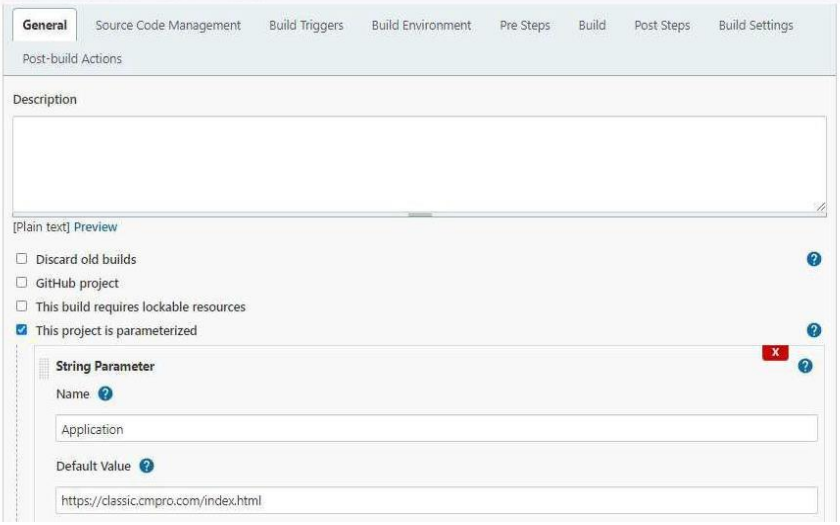
Maven

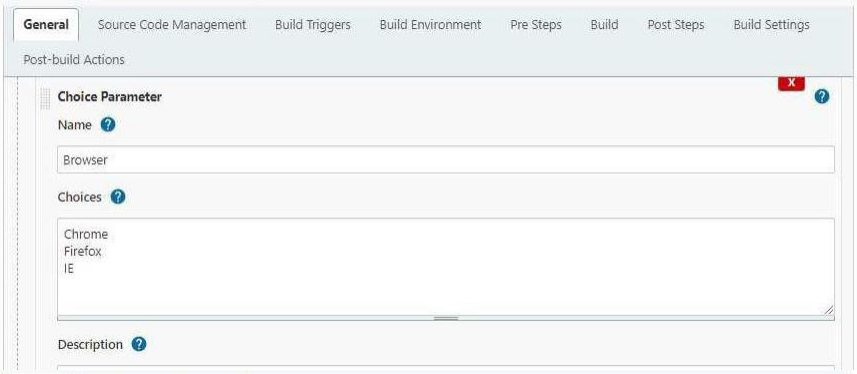
Maven is a powerful project / build management tool, based on the concept of a POM (Project Object Model) that includes project information and configuration information for Maven such as construction directory, source directory, dependency, test source directory, Goals, plugins, etc

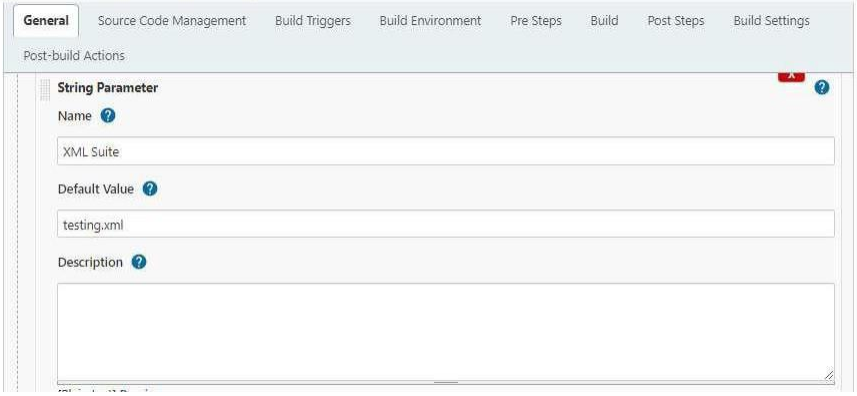
Why Jenkins and Selenium?

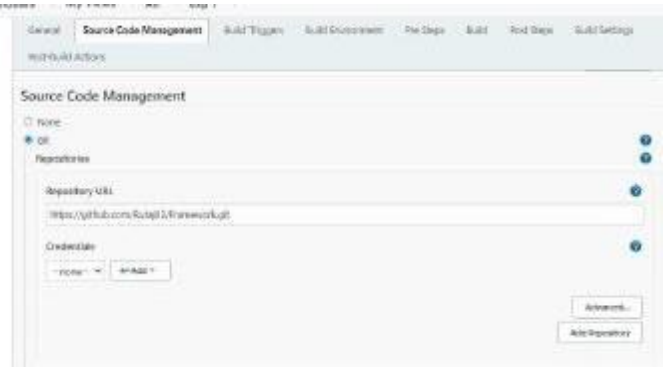
- Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass.
- Jenkins can schedule your tests to run at specific time.
- You can save the execution history and Test Reports.
- Jenkins supports Maven for building and Testing a project in continuous integration.

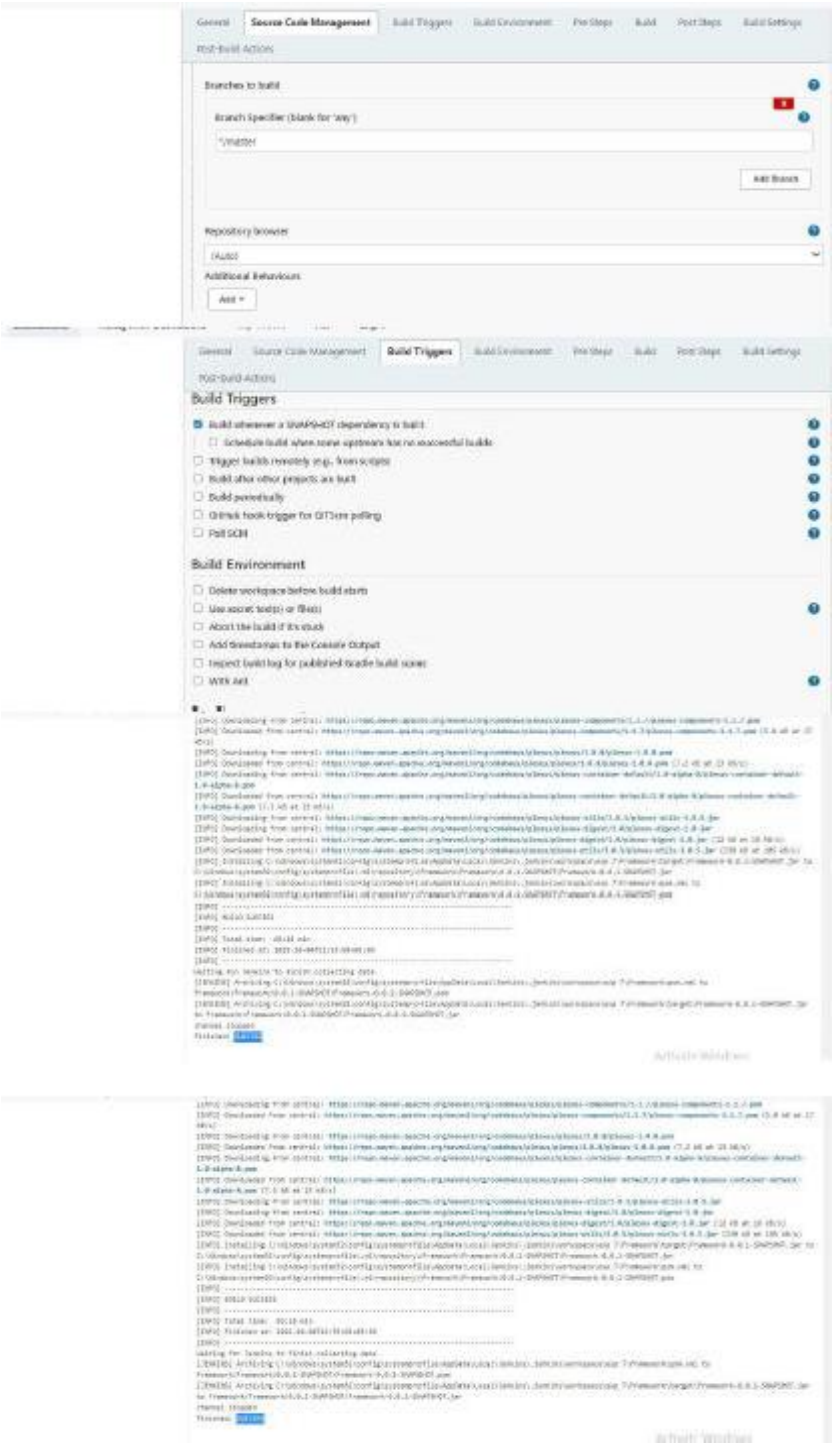
To setup and run selenium tests in Jenkins using maven:


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Description
[Plain text] Preview
Discard old builds
GitHub project
This build requires lockable resources
This project is parameterized
String Parameter
Name
Application
Default Value
https://classic.cmpro.com/index.html


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Choice Parameter
Name
Browser
Choices
Chrome
Firefox
IE
Description


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
String Parameter
Name
XML Suite
Default Value
testing.xml
Description


Source Code Management



CONCLUSION :- Hence, we successfully learn and perform about Selenium Tests in Jenkins using Maven.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 8

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

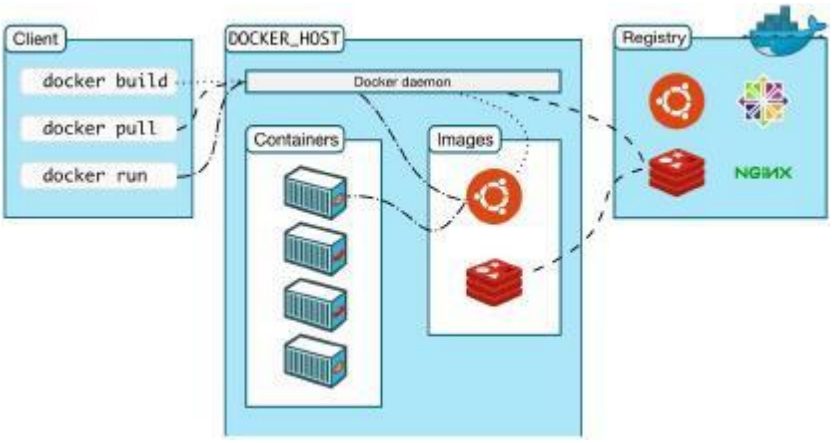
AIM :- To understand Docker Architecture and Container LifeCycle, install Docker and execute docker commands to manage images and interact with containers.

THEORY :-

Docker is an open platform for developing, shipping, andrunning applications. Docker enables you to separate yourapplications from your infrastructure so you can deliver software quickly. With Docker, you can manage yourinfrastructure in the same ways you manage your applications.By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantlyreduce the delay between writing code and running it inproduction.

Docker architecture

Docker uses a client-server architecture. The Docker client talksto the Docker daemon, which does the heavy lifting of building,running, and distributing your Docker containers. The Dockerclient and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. TheDocker client and daemon communicate using a REST API, overUNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers



The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such

as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker registries

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

Docker objects

When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

Images

An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run. You might create your own images or you might only use those created by others and published in a registry. To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.

Containers

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

```
$ docker version
Client:
 Version:      18.03.1-ce
 API version:  1.37
 Go version:   go1.9.2
 Git commit:   9ee9f40
 Built:        Thu Apr 26 07:12:25 2018
 OS/Arch:      linux/amd64
 Experimental: false
 Orchestrator: swarm

Server:
 Engine:
  Version:      18.03.0-ce
  API version:  1.37 (minimum version 1.12)
  Go version:   go1.10
  Git commit:   0520e24302
  Built:        Fri Mar 23 01:48:12 2018
  OS/Arch:      linux/amd64
  Experimental: false
$
```

```
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID,
 head over to https://hub.docker.com to create one.
Username: nikitapatil
Password:
Login Succeeded
$
```

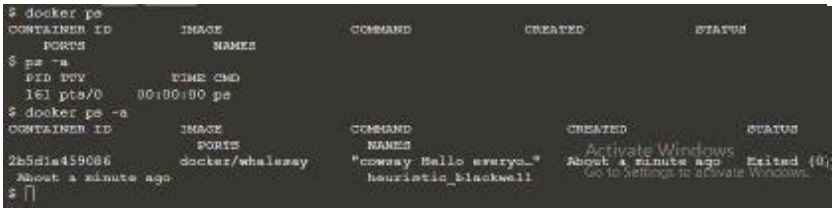
Docker search apache:

```
$ docker search apache
NAME                                DESCRIPTION                                     STARS
      OFFICIAL            AUTOMATED
httpd                               The Apache HTTP Server Project                  3704
      [OK]
tomcat                              Apache Tomcat is an open source implementati…   3145
      [OK]
maven                               Apache Maven is a software project managemen…   1365
      [OK]
apache/airflow                      Apache Airflow                                  289
apache/nifi                                                                         227
                    [OK]
apache/zeppelin                     Apache Zeppelin                                 150
                    [OK]
eboraas/apache-php                  PHP on Apache (with SSL/TLS support), built …   144
                    [OK]
eboraas/apache                      Apache (with SSL/TLS support), built on Debi…   92
                    [OK]
apacheignite/ignite                 Apache Ignite - Distributed Database            78
                    [OK]
apache/skywalking-oap-server        Apache SkyWalking OAP Server                    76
apache/superset                     Apache Superset                                 72
nimmis/apache-php5                  This is docker images of Ubuntu 14.04 LTS wi…   70
                    [OK]
apachepulsar/pulsar                 Apache Pulsar - Distributed pub/sub messagin…   49
apache/nifi-registry                Unofficial convenience binaries for Apache N…   30
linuxserver/apache                  An Apache container, brought to you by Linux…   28
```

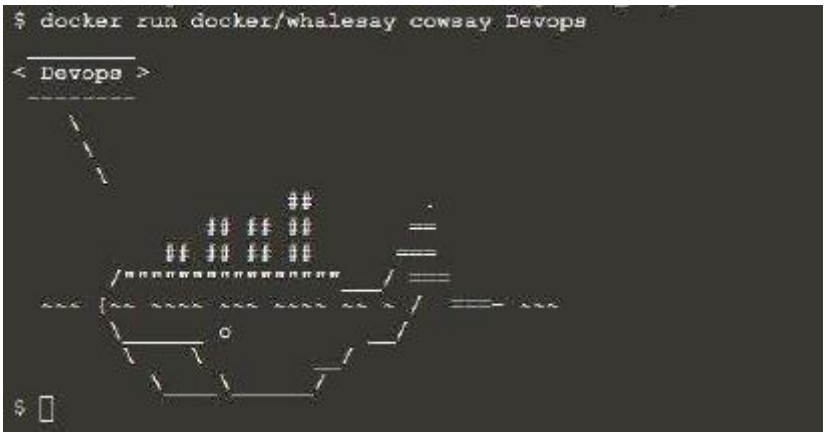
Docker search redis:

Docker run docker/whalesay cowsay Hello everyone:

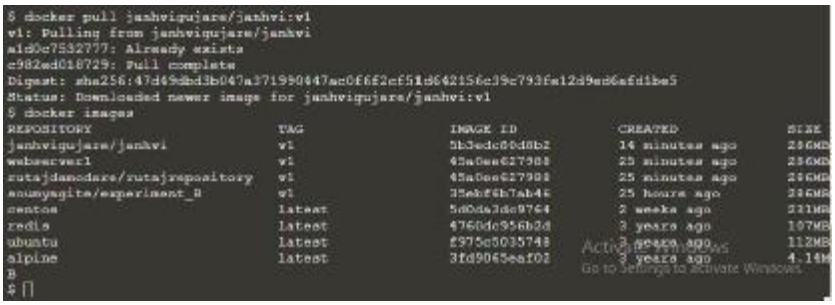
Docker ps/ docker ps -a:

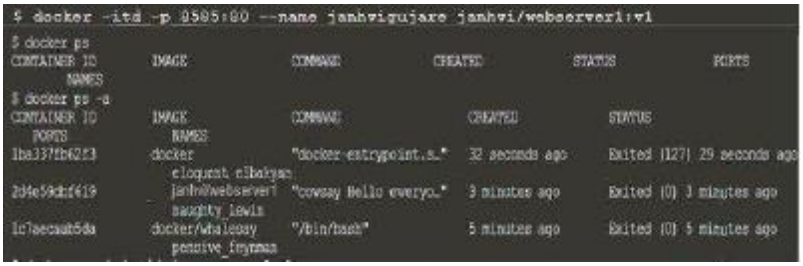


Docker run docker/whalesay cowsay DevOps:


$ docker run docker/whalesay cowsay Devops
< Devops >
$

Docker pull:


$ docker pull janhvigujare/janhvi:v1
v1: Pulling from janhvigujare/janhvi
$ docker images
janhvigujare/janhvi v1
webserver1 v1
rutajdamodare/rutajrepository v1
anumyngita/experiment_8 v1
centos latest
redis latest
ubuntu latest
alpine latest


$ docker -itd -p 8585:80 --name janhvigujare janhvi/webserver1:v1
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
1ba337fb62f3 docker "docker-entrypoint.s…" 32 seconds ago Exited (127) 29 seconds ago
2d4e59dbf619 janhv/webserver1 "cowsay Hello everyo…" 3 minutes ago Exited (0) 3 minutes ago
1c7aecaab5da docker/whalesay "/bin/bash" 5 minutes ago Exited (0) 5 minutes ago


Connecting to Port 8585
Display a different port

CONCLUSION :- Hence, we understood Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 9

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM :- To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.

THEORY :-

Docker:

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

Usage:

The docker build command builds an image from a Dockerfile and a context. The build's context is the set of files at a specified location PATH or URL. The PATH is a directory on your local filesystem. The URL is a Git repository location.

The build context is processed recursively. So, a PATH includes any subdirectories and the URL includes the repository and its submodules. This example shows a build command that uses the current directory (.) as build context:

$ docker build :

The build is run by the Docker daemon, not by the CLI. The first thing a build process does is send the entire context (recursively) to the daemon. In most cases, it's best to start with an empty directory as context and keep your Dockerfile in that directory. Add only the files needed for building the Dockerfile.

To use a file in the build context, the Dockerfile refers to the file specified in an instruction, for example, a COPY instruction. To increase the build's performance, exclude files and directories by adding a .dockerignore file to the context directory. For information about how to create a .dockerignore file see the documentation on this page.

Traditionally, the Dockerfile is called Dockerfile and located in the root of the context. You use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

$ docker build -f /path/to/a/Dockerfile .

You can specify a repository and tag at which to save the new image if the build succeeds:

$ docker build -t shykes/myapp .

To tag the image into multiple repositories after the build, add multiple -t parameters when you run the build command:

$ docker build -t shykes/myapp:1.0.2 -t

shykes/myapp:latest .

Before the Docker daemon runs the instructions in the Dockerfile, it performs a preliminary validation of the Dockerfile and returns an error if the syntax is incorrect:$ docker build -t test/myapp .

Syntax:

syntax=[remote image reference]

OUTPUT :-

CONCLUSION: Hence we successfully built an image for a web application using docker.

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

EXPERIMENT NO 10

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

AIM : To install and configure Pull based software configuration Management and provisioning tools using Puppet.

THEORY :

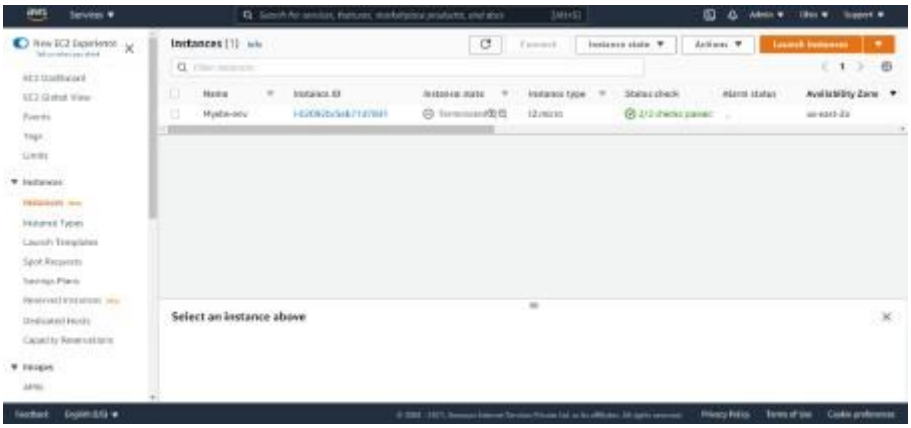
In computing, Puppet is a software configuration management tool which includes its own declarative language to describe system configuration. It is a model-driven solution that requires limited programming knowledge to use.

Puppet is produced by Puppet Inc founded by Lukes Kanies in 2005. Puppet's primary product, Puppet Enterprise, is a commercially supported version of its open-source product. Puppet's automation software uses Puppet's declarative language to manage various stages of the IT infrastructure lifecycle, including the provisioning, patching, configuration and management of operating system and application components across enterprise data centers and cloud infrastructures.

Built as cross-platform software, Puppet and Puppet enterprise operate on multiple Unix-like systems and has Microsoft Windows support.

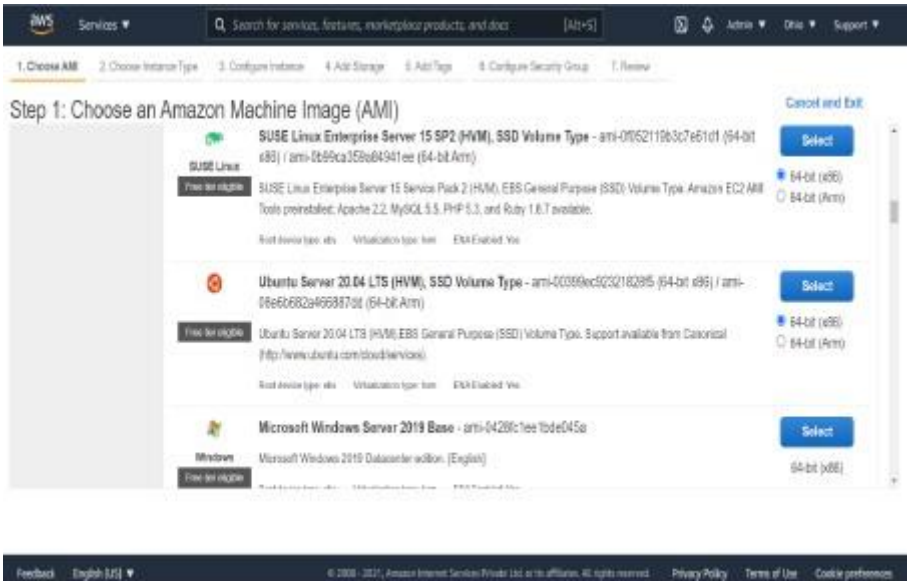
CONFIGURE PULL BASED SOFTWARE CONFIGURATION MANAGEMENT

Step 1 : Log into AWS account

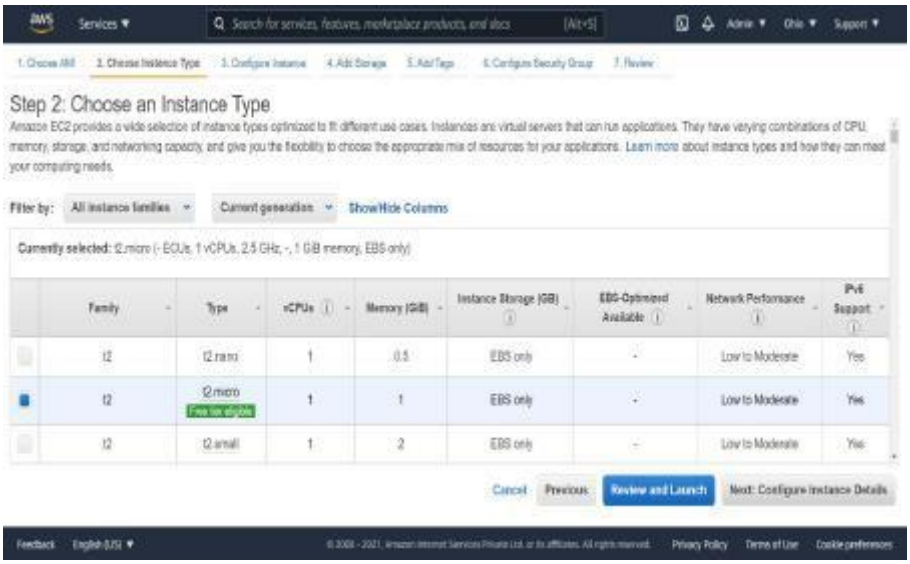


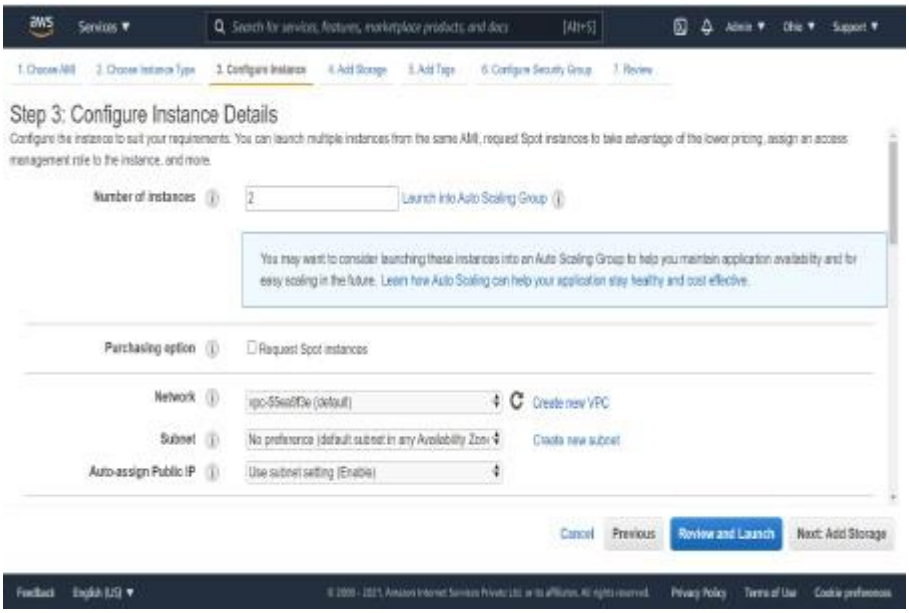
Step 2 : Goto EC2 services and create a new instance

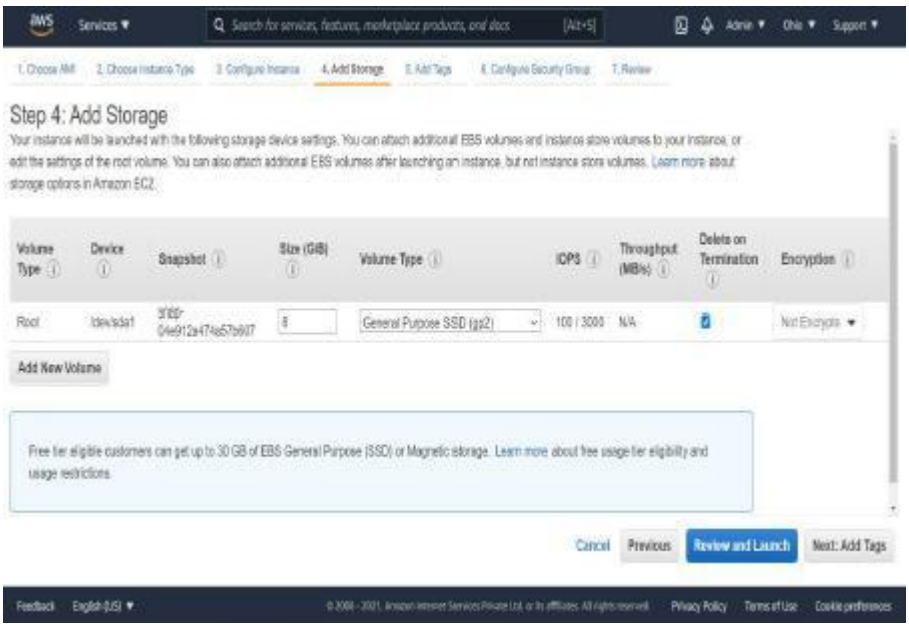
Step 3 : Proceed step by step as shown below to create a new instance



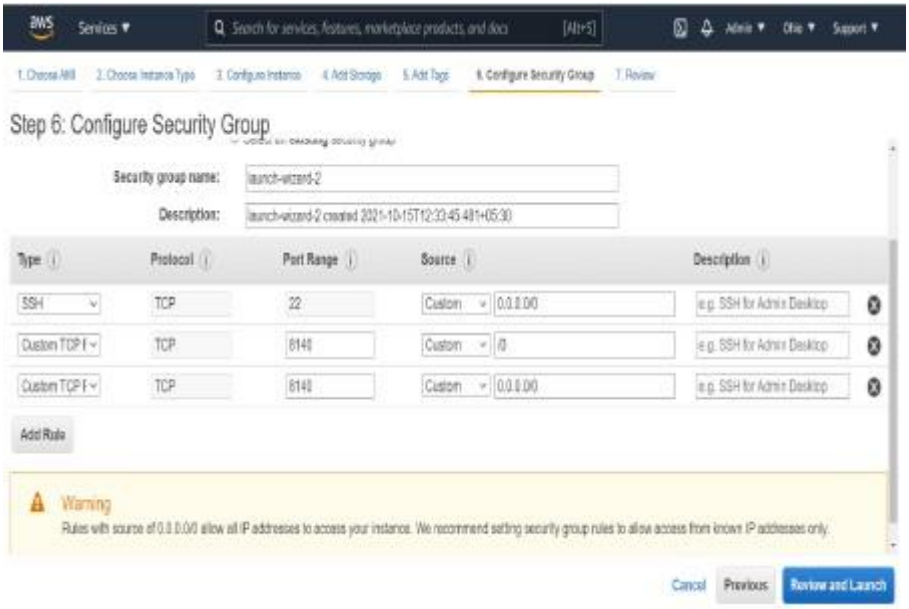
2

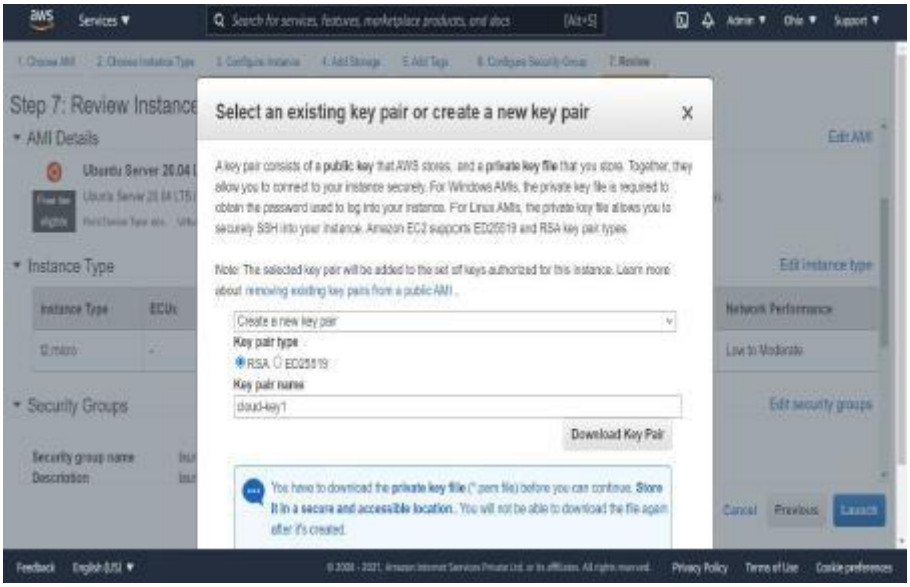

Step 2: Choose an Instance Type
Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.
Filter by: All instance families Current generation Show/Hide Columns
Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)
Family Type vCPUs Memory (GiB) Instance Storage (GB) EBS-Optimized Available Network Performance IPv6 Support
t2 t2.nano 1 0.5 EBS only - Low to Moderate Yes
t2 t2.micro Free tier eligible 1 1 EBS only - Low to Moderate Yes
t2 t2.small 1 2 EBS only - Low to Moderate Yes
Cancel Previous Review and Launch Next: Configure Instance Details


Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances 2 Launch into Auto Scaling Group
You may want to consider launching these instances into an Auto Scaling Group to help you maintain application availability and for easy scaling in the future. Learn how Auto Scaling can help your application stay healthy and cost effective.
Purchasing option Request Spot instances
Network vpc-55ea8f3e (default) Create new VPC
Subnet No preference (default subnet in any Availability Zone) Create new subnet
Auto-assign Public IP Use subnet setting (Enable)
Cancel Previous Review and Launch Next: Add Storage

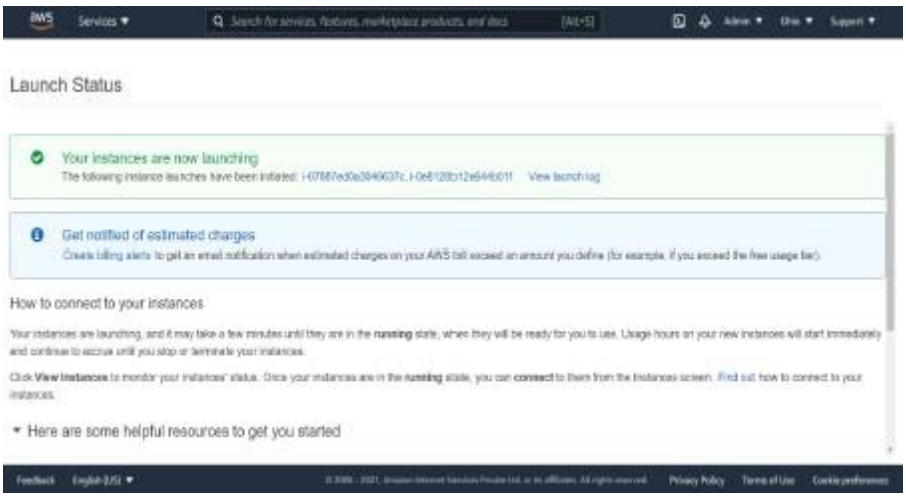

Step 4: Add Storage
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.
Volume Type Device Snapshot Size (GiB) Volume Type IOPS Throughput (MB/s) Delete on Termination Encryption
Root /dev/sda1 snap-04e912a474a57b937 8 General Purpose SSD (gp2) 100 / 3000 N/A Not Encrypted
Add New Volume
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.
Cancel Previous Review and Launch Next: Add Tags

3



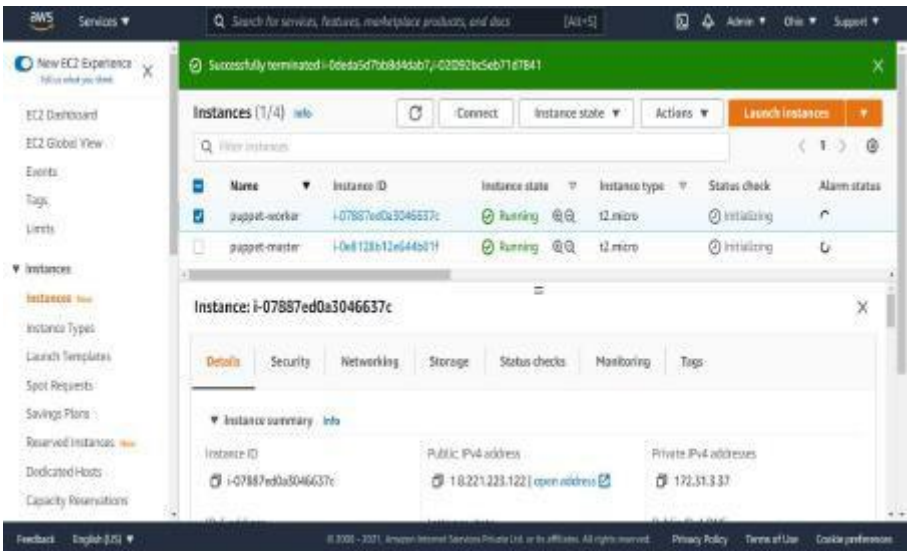


Step 4 : Check whether the newly created instances are successfully launched or not

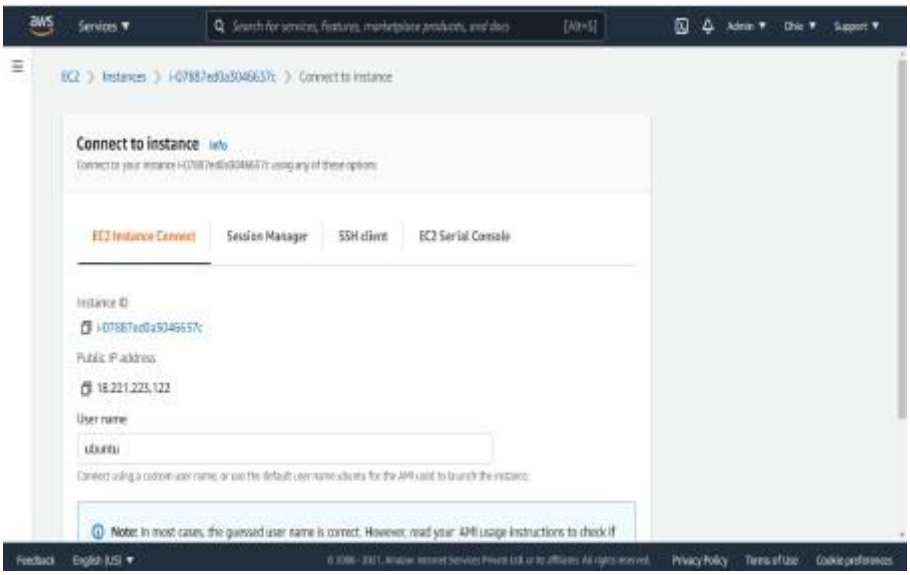


4

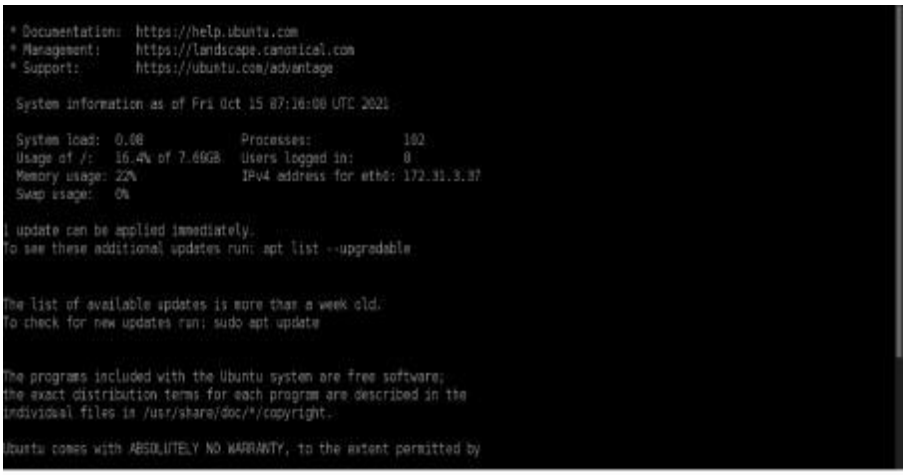
Step 5 : Name the instances as puppet-master and puppet-worker



Step 6 : Connect with the instance created recently



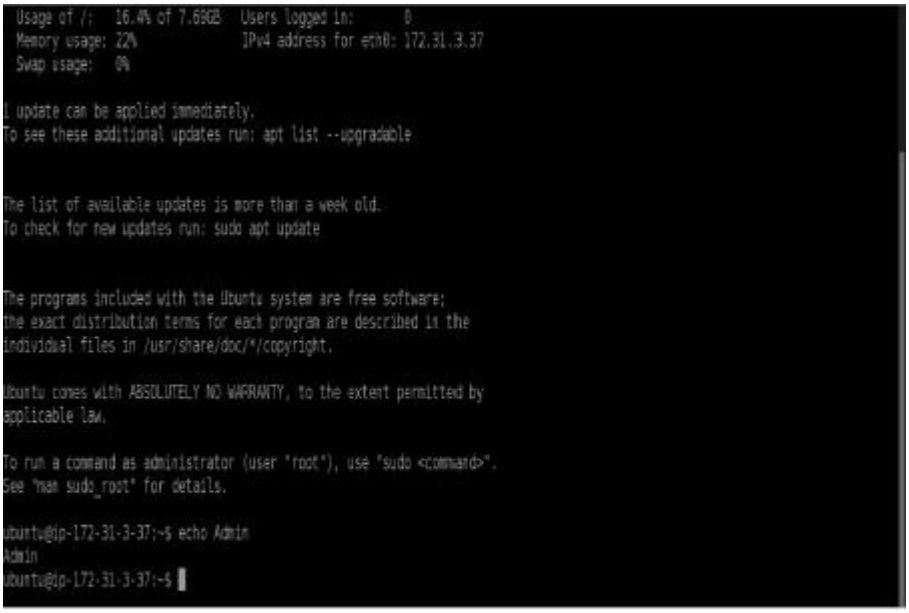
Step 7 : Type echo “Username” to see the output



i-07887ed0a3046637c (puppet-worker)

Public IPs: 18.221.223.122 Private IPs: 172.31.3.37

5



CONCLUSION : Hence we can conclude that we have learned and Pull based software configuration Management and provisioning tools using Puppet.

DATTA MEGHE COLLEGE OF ENGINEEERING

DEVOPS LAB

EXPERIMENT NO . 11

NAME : ATHANG MILIND PATIL
DIV :B1
ROLL NO : 05

AIM :- To learn Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)

THEORY :-

Puppet Manifest

In puppet, all the programs are written in Ruby programming language and added with an extension of .pp is known as manifests. The full form of .pp is the puppet program. Manifest files are puppet programs. This is used to manage the target host system. All the puppet programs follow the puppet coding style.

We can use a set of different kinds of resources in any manifest, which is grouped by definition and class.

Puppet manifest also supports the conditional statement. The default manifest file is available in the /etc/puppet/manifests/site.pp location.

Manifest Components

Puppet manifest has the following components:

Files: Files are the plain text files that can be directly deployed on your puppet clients. Such as yum.conf, httpd.con, etc. Resources: Resources are the elements that we need to evaluate or change. Resources can be packages, files, etc. Templates: This is used to create configuration files on nodes and which we can reuse later.

Nodes: Block of code where all the information and definition related to the client are defined here.

Classes: Classes are used to group different types of resources. Writing Manifests

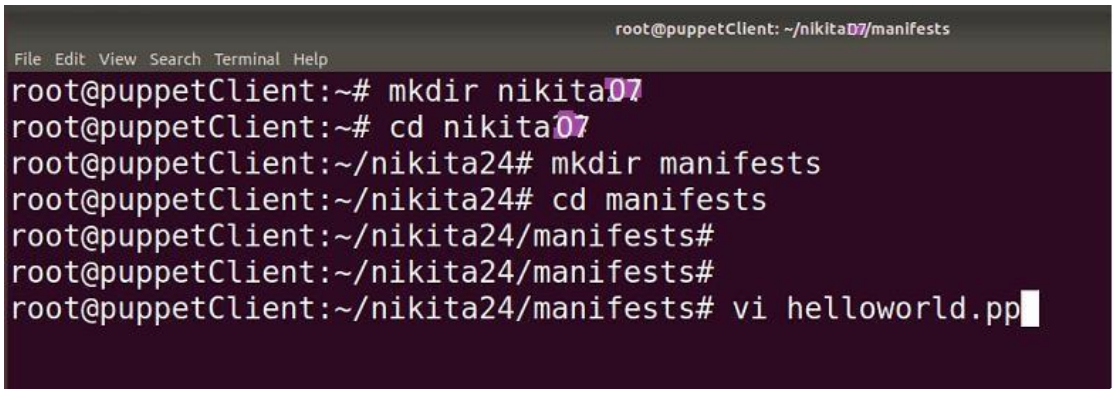
Working with Variables

Puppet provides many in-built variables that we can use in the manifest. As well as we can create our own variable to define in puppet manifest. Puppet provides different types of variables. Some frequently used variables are strings or an array of string.

Create a manifest and write the above script in that manifest. For this first, you have to switch to the root user.

nikita@puppetClient:~$ sudo su

root@puppetClient:~# vi helloworld.pp



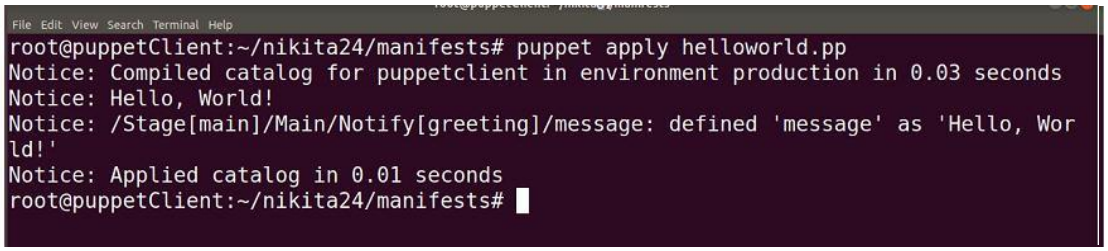
Applying a Manifest

The main quality of the puppet is the ease of testing your code. In this, to work on puppet manifests, there is no need to configure complicated testing environments.

To apply the manifest, puppet uses apply command, which tells puppet to apply a single puppet manifest:

puppet apply helloworld.pp

Notice: Compiled catalog for puppetclient in environment prod
uction in 0.03 seconds
Notice: Hello, World!
Notice: /Stage[main]/Main/Notify[greeting]/message:
defined 'message' as 'Hello, World!'
Notice: Applied catalog run in 0.01 seconds



Puppet has successfully applied the manifest. This process done by many steps:

- Compile or build the Puppet catalog from the manifest.
- Employes dependency and ordering information to decide the evaluation order.
- Evaluates the desired resource to decide if changes should be implemented.
- Create, update, or remove the resource- A message will generate in the notice.
- Gives verbose output about the catalog application.

Puppet Modules

Puppet Module is a collection of files, classes, templates, and resources. Each module handles a specific task in your infrastructure, such as installing and configuring a piece of software.

Since modules allow you to divide your code into multiple manifests, it is very helpful in organizing your puppet code.

Modules are the reusable and shareable units in the puppet. Modules must be installed in the puppet modulepath. And the modulepath is /etc/puppet/modules directory.

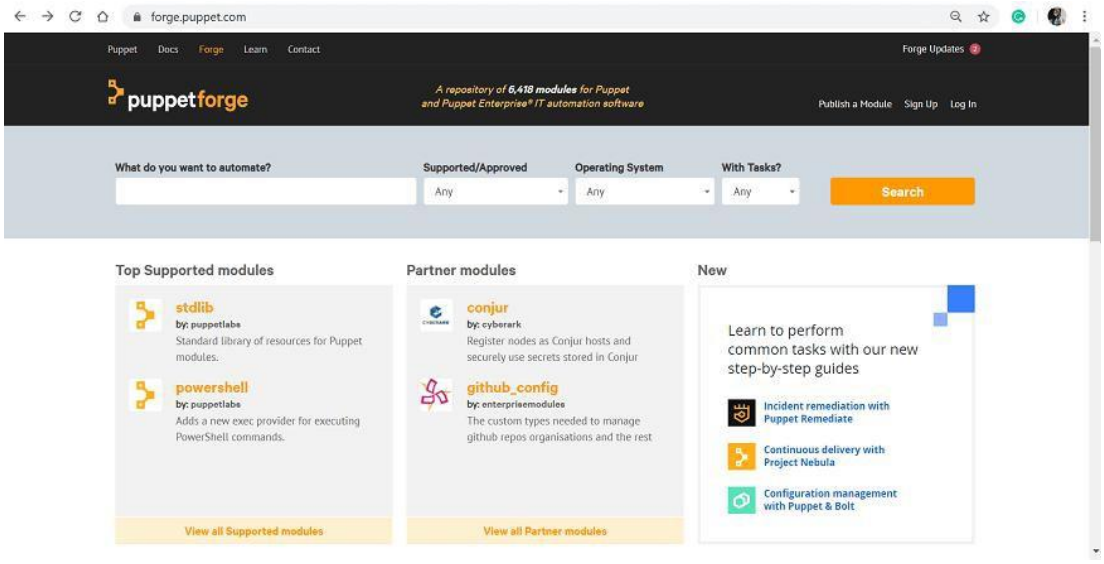
Installation of Puppet Module

The open-source puppet has many pre-existing modules. These modules are written and developed by the puppet communities. Anyone can update pre-existing modules. These are the in-built, public modules that anyone can download, install, and use it. There are over 6,000+ pre-existing modules available in the Puppet Forge.

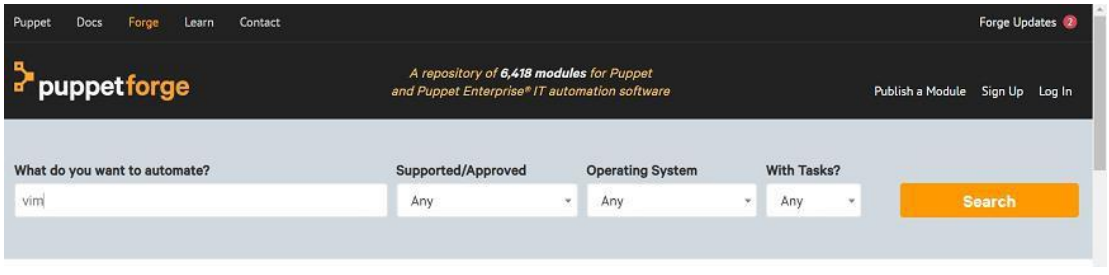
Let's see the steps to download and install these pre-existing puppet modules.

Here, we are going to download the vim module from Puppet Forge. Vim is available in a package that is a free and open-source text editor in Linux operating system.

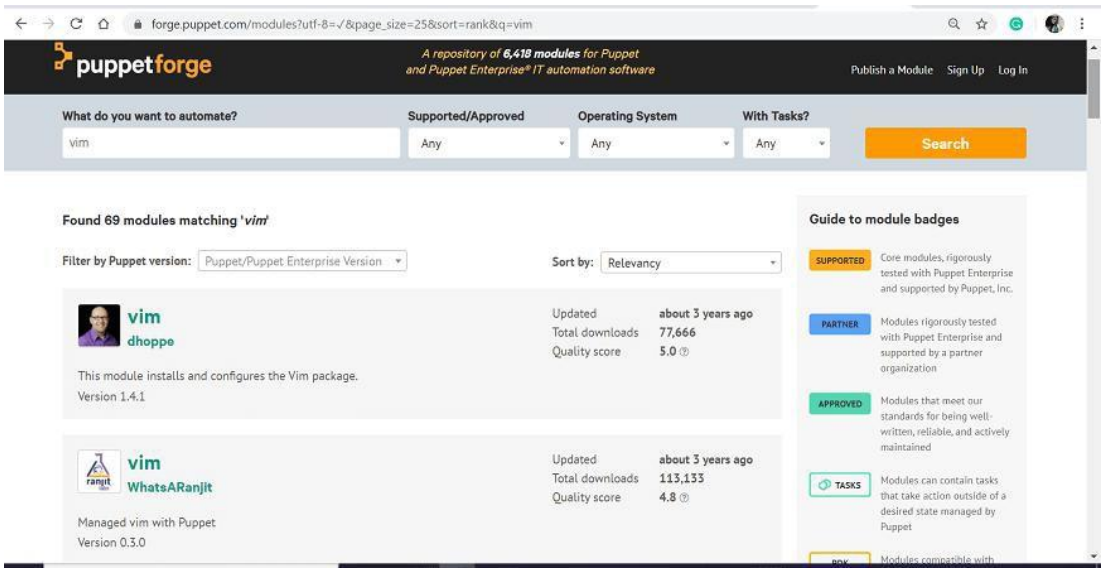
Step 1: Click here to open the official Puppet Forge page. When you click on the link, it will display the following page:



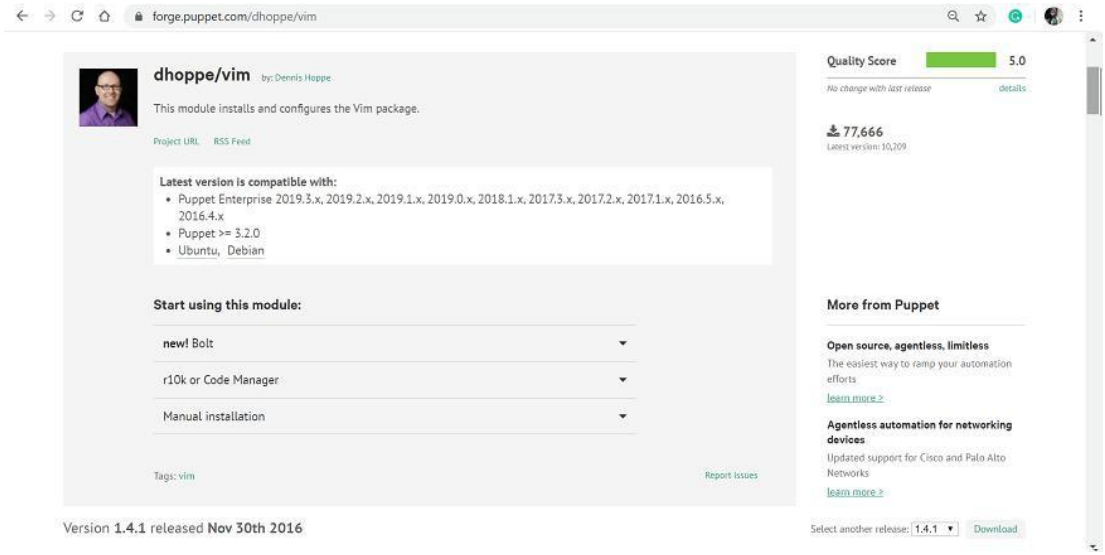
Step 2: In the search bar, enter the name of the module that you want to download. Here, we are going to download vim module:



Step 3: When you click on the Search button, it will display multiple results. Choose the appropriate one. To decide your selection, you can click on the individual module to see complete details. In our case, we will use dhoppe vim.



Step 4: To download the module, click on the download button from the right side of the page, and we will get the module in tarball format.



Step 5: Once the download finish, execute the following command to install a module from the tarball:

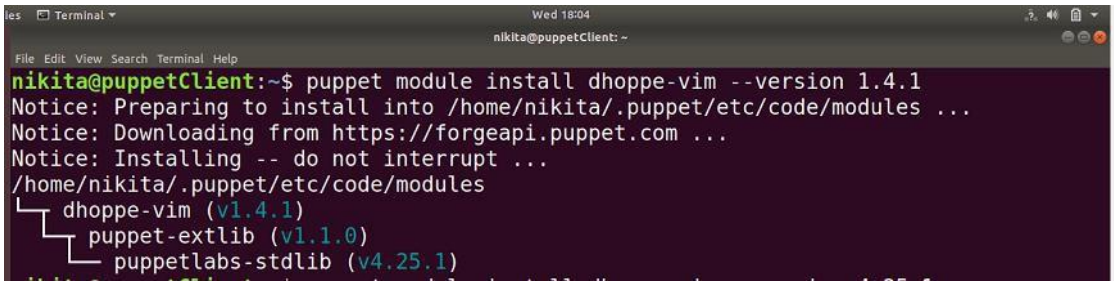
puppet module install /path/dhoppe-vim-1.4.1.tar.gz

In the above command, 'path' is the path of the directory where your tarball is saved.

We can also install the puppet modules online.

To download and install the module from the puppet module tool, execute the following command:

puppet module install dhoppe-vim --version 1.4.1



Puppet Classes

Puppet classes are the set of puppet resources that are grouped together as a single unit. Classes are used to model the fundamental aspects of the node. Puppet uses classes to make the structure reusable and organized. Classes can only be evaluated once per node.

Classes are described within the manifest file, located inside Puppet modules. The main reason for using a class is to decrease the duplication of the same code inside any manifest file or other puppet code.

Defining a Class

Before using a class, we have to define it, which is done by the class keyword, the name of the class, curly braces, and a set of code. This part of the code does not automatically evaluate the code.

Syntax:

```
class my_class {
  ... puppet code ...
```

```
}
```

Declaring a Class

The declaration part of a class evaluates the code in the class and applies all its resources. This part of the code actually does something.

Syntax:

```
class my_class {
  ... puppet code ...
}
include my_class


class unix {
   file {
      '/etc/passwd':
      owner => 'superuser',
      group => 'superuser',
      mode => 644;
      '/etc/shadow':
      owner => 'nikita',
      group  =>  'nikita',
      mode => 440;
   }
}
```

Let's see another simple example which is similar to the above example:

```
class unix {
   file {
      '/etc/passwd':
      owner => 'superuser',
      group  =>  'superuser',
      mode => 644;
   }
```

```
  file {'/etc/shadow':
    owner => 'nikita',
    group  => 'nikita',
    mode => 440;
  }
}
```

Parameterized Class

Parameters are used to allow a class to request external data. If a class has to configure itself to data other than facts, the data will typically be inserted by a parameter into the class.

Let's see one example:

```
class windows_ntp (
  String $server = 'time.windows.com',
) {
  registry::value { 'NtpServer':
    key  => 'HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Services\W32Time\Parameters',
    data => "${server},0x9",
  }
  service { 'w32time':
    ensure => running,
    enable => true,
  }
}
```

In the above example, we created a class windows_ntp, which grouped together a registry resource and a service resource to configure the Windows time service. windows_ntp class accepts the time server address as a parameter named $server.

Puppet Function

As we know, Puppet uses Ruby programming language, and like other programming languages, Ruby also supports the function. We can write the functions in Ruby language and can be

distributed in puppet modules. Puppet provides two different types of functions:

statement: This type of function did not return any value and used to perform standalone tasks. It can be used to import Puppet modules in the new manifest file.

rvalue: In Puppet, if you want to define a function with their return type, then you can use rvalue functions. rvalue can only be used when the statement needs a value, like a case statement or assignment. A function can only take two parameters as an argument.

Function	Description	Syntax	Type
abs	It will return the absolute value of a Numeric value.	abs(Numeric($str_val))	rvalue
alert	This function is used for logging a message on the server at level alert.	alert(Any *$values)	statement
break	Breaks an innermost iteration as if an end of input was found. This function doesn't accept any argument.	break()	statement
binary_file	This function loads a binary file from a module or file system and returns binary content.	binary_file(String $path)	rvalue
call	Used to call an arbitrary function by name.	call(String $function_name, Any *$arguments, Optional[Callable] &$block)	statement
create_resources	This function is used to change a hash into a collection of resources and inserts them into the catalog.	create_resources($type, $resources)	statement
debug	Logs a message on the server at level debug.	debug(Any *$values)	statement

CONCLUSION :- Hence ,we successfully learn Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)

DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

ASSIGNMENT NO 1

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

Case study on DevOps implementation in real world. (For example, Students can create an docker image of any project of their choice).

DevOps is a software development practice that promotes collaboration between development and operations, resulting in faster and more reliable software delivery. Commonly referred to as a culture, DevOps connects people, process, and technology to deliver continuous value.

The software development process can be a highly manual process, resulting in a significant number of code errors. Development and operations teams can often be out of sync, which can slow software delivery and disappoint business stakeholders. DevOps creates efficiency across all tasks involved in the development, deployment and maintenance of software.

Connecting development and operations leads to increased visibility, more accurate requirements, improved communication and faster time to market.

A Docker container image is a standalone, lightweight package that can be executed and contains all the requirements you need to run an application, such as: code, runtime, libraries, and settings. The image can then be pushed to a container registry and pulled to your server to run as a container. Build a custom Docker image with Dockerfile.

Creating a Docker image from scratch using a Dockerfile for your app. Later on in this tutorial, we'll explain how to use prebuilt Docker images with your ASP.NET Core web app.

Create a Dockerfile in your project folder

The Dockerfile is essentially a recipe for creating your Docker image and is added to the project's root. It includes the necessary commands for a user to build an image when executing docker build. For additional details to better understand the commands within the file, please refer to the Dockerfile reference.

Dockerfile

```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base

WORKDIR /app

EXPOSE 80

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build WORKDIR /src COPY

["DockerDemo.csproj", ""]

RUN dotnet restore "./DockerDemo.csproj"

COPY . .

WORKDIR "/src/."

RUN dotnet build "DockerDemo.csproj" -c Release -o

/app/build

FROM build AS publish

RUN dotnet publish "DockerDemo.csproj" -c Release -o

/app/publish FROM

base AS final

WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "DockerDemo.dll”]
```

Create a .dockerignore file

Similar to a .gitignore file a .dockerignore file allows you to mention a list of files or directories that you want to ignore while building the Docker image. This is essential because it helps reduce the size of an image and speeds up the docker building process.

```
.dockerignore
class=""&gt; **/.classpath
**/.dockerignore
**/.env
**/.git
**/.gitignore
**/.project
**/.settings
**/.toolstarget
**/.vs
**/.vscode
```

**/*.*proj.user
**/*.dbmdl
**/*.jfm
**/azds.yaml
**/bin
**/charts
**/docker-compose*
**/Dockerfile*
**/node_modules
**/npm-debug.log

**/obj
**/secrets.dev.yaml
**/values.dev.yaml
LICENSE
README.md

Build Docker image and start container

The docker build builds a Docker image from the Dockerfile and a “context”. The context is a set of files located in a specified PATH or URL.

Open the terminal or command prompt and navigate to your project folder. Use the following command to build your Docker image:

docker build -t dockerdemo .

This returns the status of your build.

Create and run container

The docker run command creates a new container and runs the Docker image.

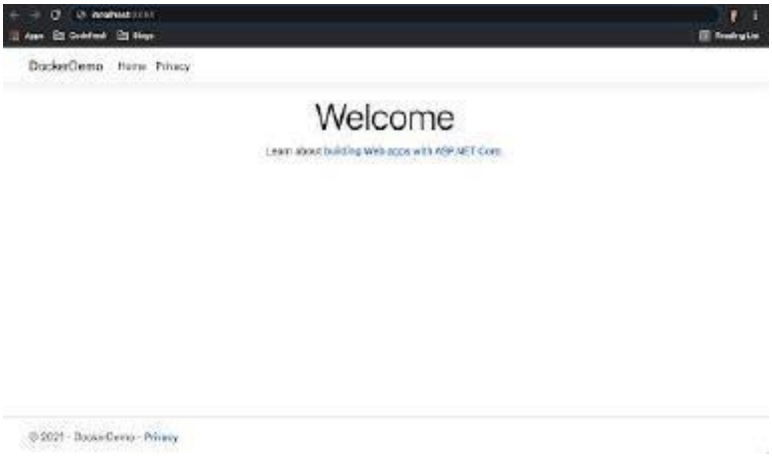
Open the terminal or command prompt and use the following command to run your Docker image:

docker run -d -p 8080:80 --name myapp dockerdemo

Check that the container was created and is running with the command:

docker ps

Lastly, go to http://localhost:8080/ to access your app in a web browser.



DATTA MEGHE COLLEGE OF ENGINEERING

DEVOPS LAB

ASSIGNMENT NO 2

NAME : ATHANG MILIND PATIL

DIV :B1

ROLL NO : 05

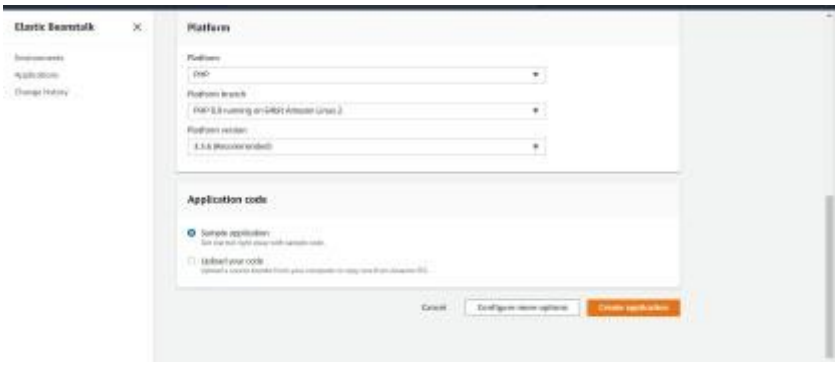
Implement AWS code Pipeline

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated. You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

Step1: Create a deployment environment

Your continuous deployment pipeline will need a target environment containing virtual servers, or Amazon EC2 instances, where it will deploy sample code. You will prepare this environment before creating the pipeline.

1) To simplify the process of setting up and configuring EC2 instances for this tutorial, you will spin up a sample environment using AWS Elastic Beanstalk. Elastic Beanstalk lets you easily host web applications without needing to launch, configure, or operate virtual servers on your own.
2) It automatically provisions and operates the infrastructure (e.g. virtual servers, load balancers, etc.) and provides the application stack (e.g. OS, language and framework, web and application server, etc.) for you.
3) 2) Name your web app and choose PHP from the drop-down menu(or any other language you are interested in) and then click Create Application.



3) Elastic Beanstalk will begin creating a sample environment for you to deploy your application to. It will create an Amazon EC2 instance, a security group, an Auto Scaling group, an Amazon S3 bucket, Amazon CloudWatch alarms, and a domain name for your application.

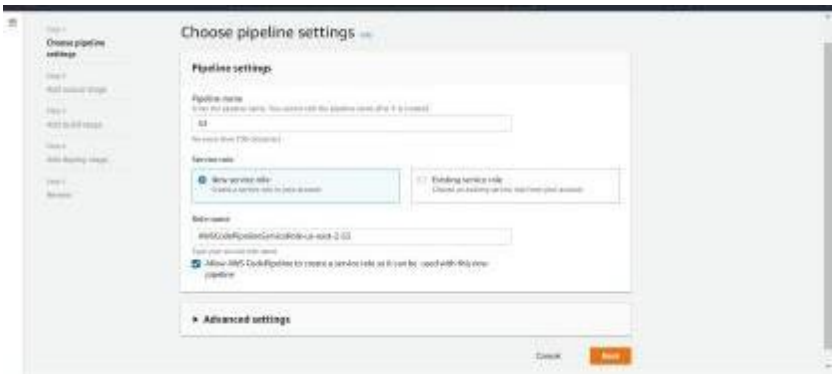
Note: This will take several minutes to complete.

Step2: Get a copy of the sample code

In this step, you will retrieve a copy of the sample app's code and choose a source to host the code.

The pipeline takes code from the source and then performs actions on it. You can use one of three options as your source: a GitHub repository, an Amazon S3 bucket, or an

AWS CodeCommit repository. Select your preference and follow the steps below:

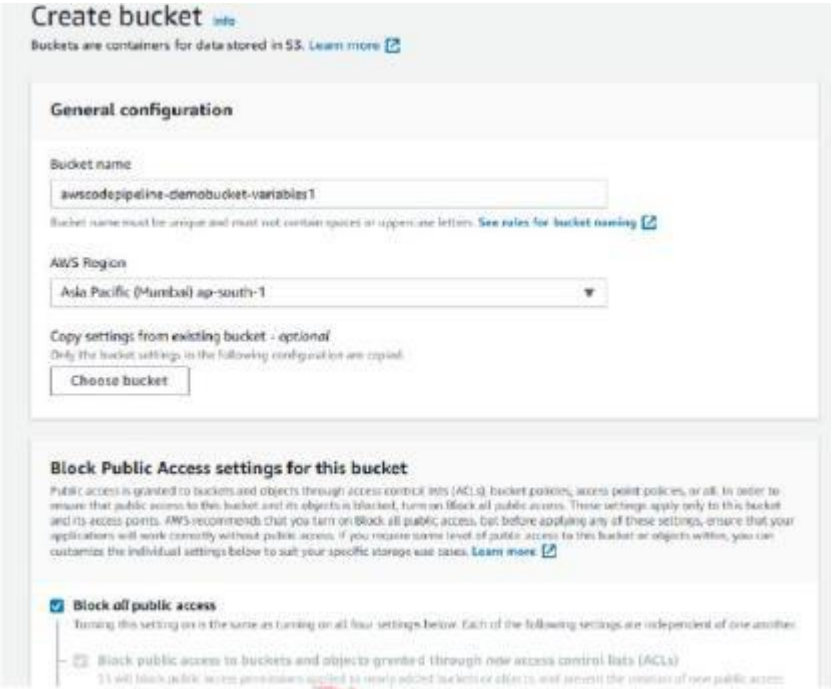


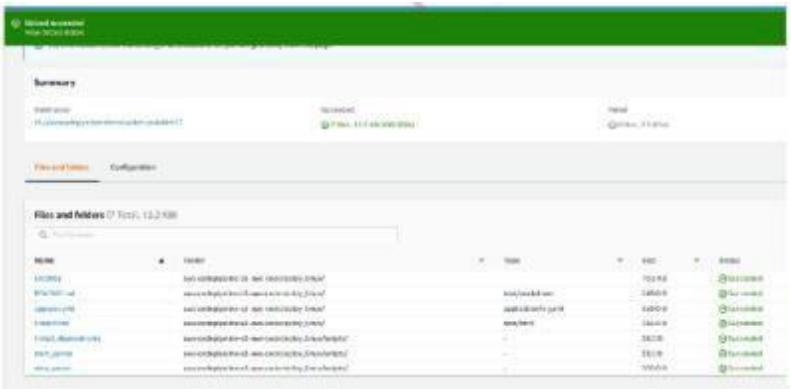
a. If you plan to use Amazon S3 as your source, you will retrieve the sample code from the AWS GitHub repository, save it to your computer, and upload it to an Amazon S3 bucket.

• Visit our GitHub repository containing the sample code at https://github.com/imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0
• Click the dist folder.
b. Save the source files to your computer:
• Click the file named aws-codepipeline-s3-awscodedeploy_ linux.zip
• Click View Raw.
• Save the sample file to your local computer.
c. open the Amazon S3 console and create your Amazon S3 bucket:
• Click Create Bucket
• Bucket Name: type a unique name for your bucket, such as awscodepipeline-demobucketvariables.
All bucket names in Amazon S3 must be unique, so use one of your own, not one with the name shown in the example.
• Region: In the drop-down, select the region where you will create your pipeline, such as ap-South-1
• Click Create.

d.The console displays the newly created bucket, which is empty.
• Click Properties.
• Expand Versioning and select Enable Versioning. When versioning is enabled, Amazon S3 saves every version of every object in the bucket.
e. You will now upload the sample code to the Amazon S3 bucket:
• Click Upload.
• Follow the on-screen directions to upload the .zip file containing the sample code you

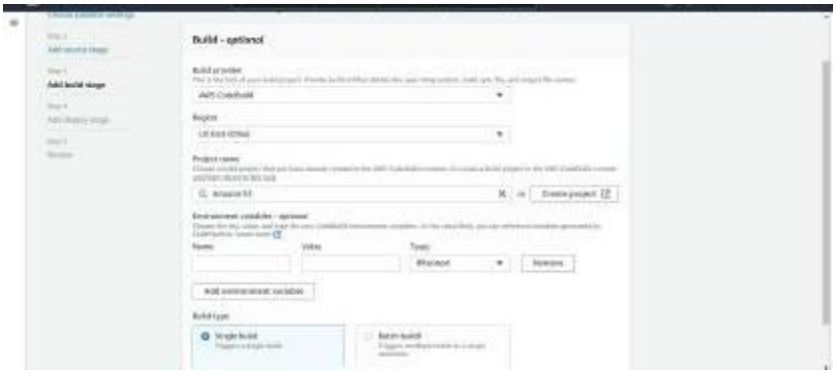
downloaded from GitHub





Step3: Create your Pipeline

In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment. A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this we will skip the build stage. Goto Pipeline again and create it



Step 4 : Deploy Stage

- Deployment provider:Click AWS Elastic Beanstalk.
- Application name: MYEBS.
- Environment name: Click Myebs-env.

After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline perform each action. To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change from No executions yet to In Progress, and then to either Succeeded or Failed. The pipeline should complete the first run within a few minutes.

Now go to your EBS environment and click on the URL to view the sample website you deployed.

Step 5: Commit a change and then update your app

In this step, you will revise the sample code and commit the change to your repository. CodePipeline will detect your updated sample code and then automatically initiate deploying it to your EC2 instance via Elastic Beanstalk.

Note that the sample web page you deployed refers to AWS CodeDeploy, a service that automates code deployments. In CodePipeline, CodeDeploy is an alternative to using Elastic Beanstalk for deployment actions. Let's update the sample code so that it correctly states that you deployed the

sample using Elastic Beanstalk.

a. Visit your own copy of the repository that you forked in GitHub.
b. Update the webpage by copying and pasting the following text on line 30:
c. Commit the change to your repository.
d. Return to your pipeline in the CodePipeline console. In a few minutes, you should see the Source change to blue, indicating that the pipeline has detected the changes you made to your source repository. Once this occurs, it will automatically move the updated code to Elastic Beanstalk.

• After the pipeline status displays Succeeded, in the status area for the Beta stage, click AWS Elastic Beanstalk.
e. The AWS Elastic Beanstalk console opens with the details of the deployment. Select the environment you created earlier. And click the URL again from EBS environment again.

Step 6: Clean up your resources

To avoid future charges, you will delete all the resources you launched throughout this tutorial, which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.
a. First, you will delete your pipeline:
• In the pipeline view, click Edit.
• Click Delete.
• Type in the name of your pipeline and click Delete.
b. Second, delete your Elastic Beanstalk application:
• Visit the Elastic Beanstalk console.
• Click Actions.
• Then click Terminate Environment.
You have successfully created an automated software release pipeline using AWS CodePipeline!
Using CodePipeline, you created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk. Your pipeline will

automatically deploy your code every time there is a code change.