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~~ Parmar Jay Rohit

Of ___TE_____Class _____ IT-B_____Roll No. _3_____

Subject _____DevOps_____has performed the experiments /

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

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

Date _____

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	1-5	13/7/21	
2.	To understand Version Control System / Source Code Management, install git and create a GitHub account.	LO1 & LO2	6-20	27/7/2021	
3.	To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet	LO1 & LO2	21-29	27/7/2021	
4.	To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job.	LO1 & LO3	30-43	3/8/2021	
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	44-52	3/8/2021	
6.	To understand Jenkins master- slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.	LO1 & LO3	53-59	10/8/2021	
7.	To Setup and Run Selenium Tests in Jenkins Using Maven.	LO1 , LO3 & LO4	60-69	17/8/2021	
8.	To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.	LO1 & LO5	70-79	7/9/2021	
9.	To learn Dockerfile instructions, build an image for a sample web application using Dockerfile.	LO1 & LO5	80-84	21/9/2021	
10.	To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet.	LO1 & LO6	85-95	28/9/2021	
11.	To learn Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function)	LO1 & LO6	96-113	5/10/2021	

12.	Assignment no.1	CO1	114-118	19/8/2021	
13.	Assignment no.2	CO2	119-128	1/10/2021	

Expt. no. 1

Page No.
Date

Aim : Introduction to DevOps

Theory :

- Evolution of Software Development over years.

i) Waterfall Methodology

The evolution of Software Development life cycle starts in the mid twentieth century. The software was developed by code & fix technique 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 development & then a model stagewise was originated.

Royee (1970) proposed the waterfall Methodology in order to avoid the difficult nature of "code & fix" approach He proposed the construction of a prototype, & involvement of the users in 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 maintances & disassembly.

2) Agile methodology :-

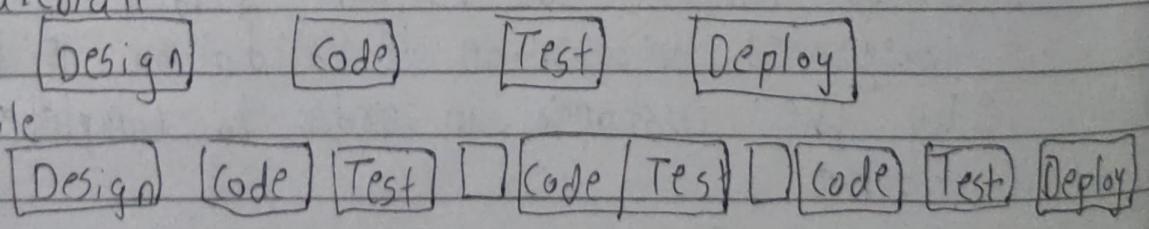
Agile methodology are a new host of methodologies that claim to overcome the limitations of traditional plan driven SDMs.

Agile means a practice that promotes continous iteration of development & testing throughout the software development lifecycle of the project.

The manifesto states that agile development should Focus on Four core values.

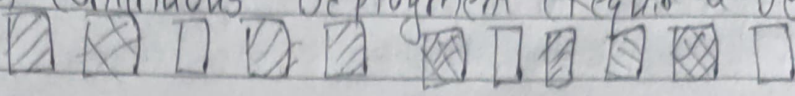
(i) Individuals & interactions over processes & tools
(ii) Working software over comprehensive documentation
(iii) Customer collaboration over contract negotiation
(iv) Responding to change over following a plan.

most studies reported that agile development pratice are easy to adopt & work well.



Page No.
Date

Agile With Continuous Deployment Require a Devops

Design 

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

- What is DevOps?

① Devops is a set of practice that combines software development (Dev) & information-technology operations (ops) which aims to shorten the system development life cycle & provide continuous delivery with high software quality.

② This allows a single team to handle the entire application lifestyle, from development to testing, deployment & operations.

③ Devops helps you to reduce the disconnection between software developers quality assurance (QA) engineers & system administrator.

④ Devops can also be defined as a sequence of development & IT operations with better communication & collaboration Devops helps to increase organisation.

Page No.
Date

Speeds to deliver applications & services

- Why devops ?
we need to understand why we need the Devops over the other methods.

① The operations & development team worked incomplete isolations.
② After the design build the testing & deployment are performed respectively. That's why they consumed more time than actual build cycles.
③ Without the use of Devops, the team members are spending a large amount of time on designing, testing & deploying instead of building the project.
④ Manual code deployment leads to human errors in production.

- Why DevOps is Important ?

① Devops is important because it's a Software development & operations approach that enables faster development of new products & easier maintainances of existance development.
② Devops is no more than a set of processes that co-ordinate to unity development team & processes to complement Software development.
③ The main reason behind Devops popularity is that it allows enterprises to execute & improve products at a faster place than traditional software development methods.

Page No.
Date

- Benefits of Devops :-

Technical Benefits :-

(i) Continuous software delivery
(ii) less complexity to manage.
(iii) Faster resolution of problems.

Cultural Benefits :-

(i) Happier & more productive teams.
(ii) Higher employee engagement
(iii) Greater professional development opportunies.

Business Benefits :-

(i) Faster delivery of features
(ii) More stable operating environment
(iii) Improved communication & collaboration
(iv) More time to innovate.

Conclusion :- We can conclude that we have completed studying Introduction to Devops.

EXPERIMENT NO. 2

Aim:

To study and practice GIT commands for version control.

Theory:

Git is one of the most popular version control systems. It is a distributed version control system. Changes do not have to be committed to the same central repository, which would require that every person working on the project to access that central repository and download the latest code in order to save changes.

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

The following diagram depict the all supported operations in GIT

Installation of GIT

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

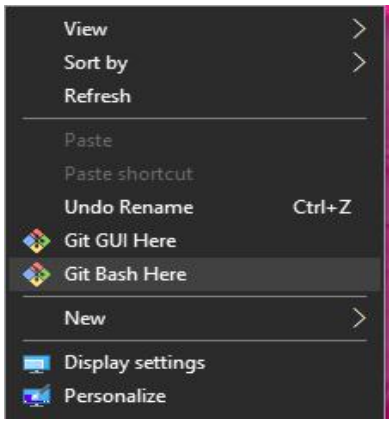
straightforward installation.

2) In Ubuntu, install GIT using $sudo apt install git, Confirm 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.



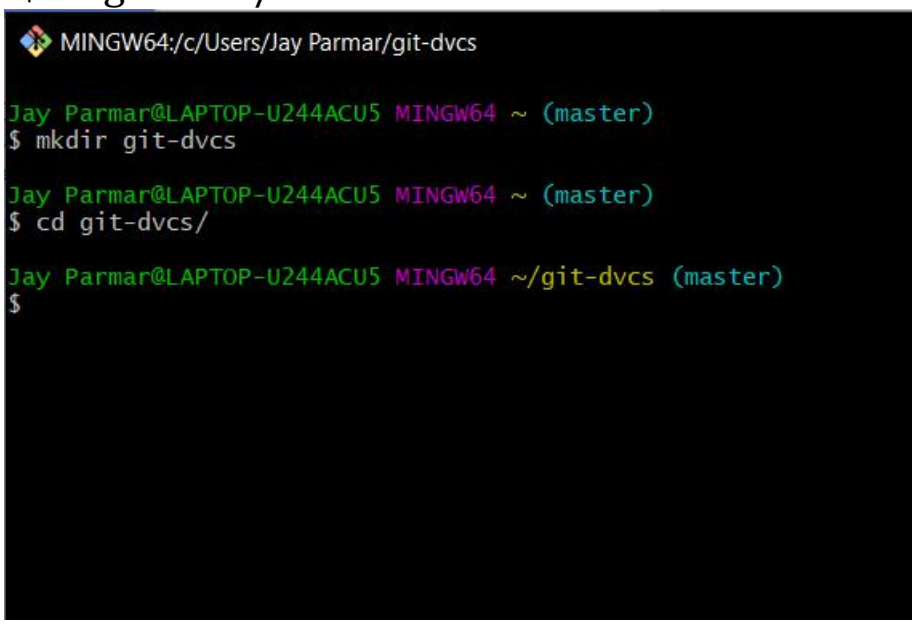
Version Control

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

and change directory to dvcs.

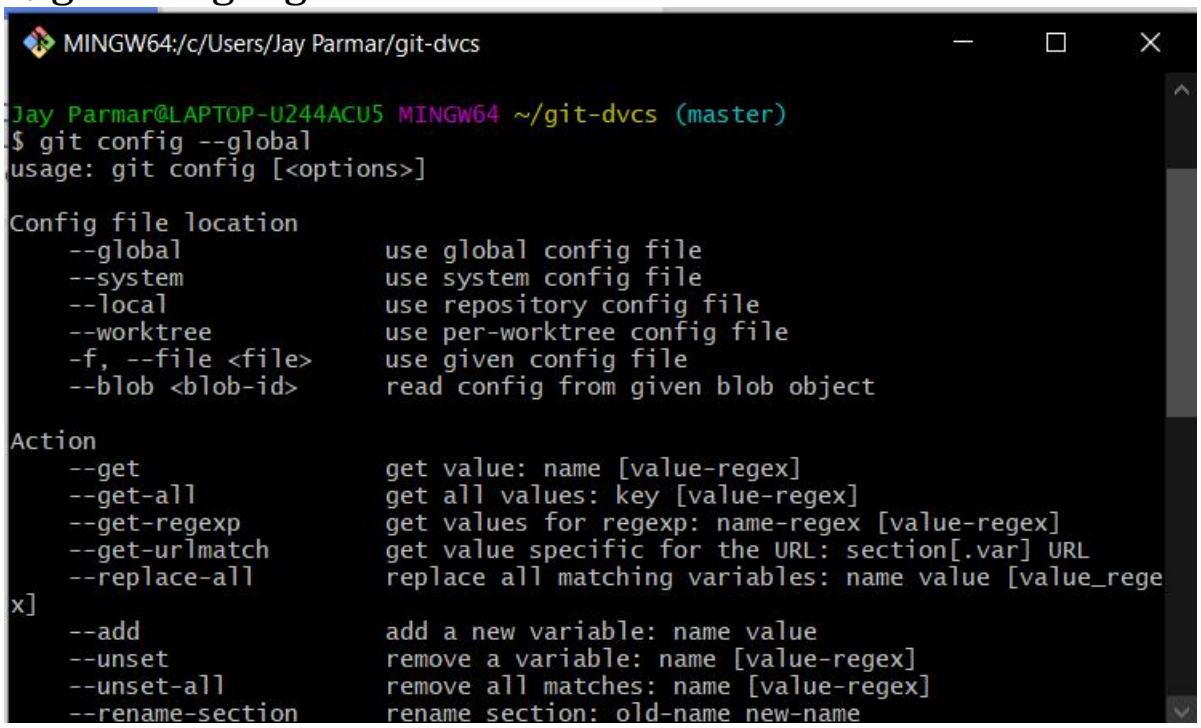
$ mkdir git-dvcs

$ cd git-dvcs/



Now check the user information using

$ git config –-global



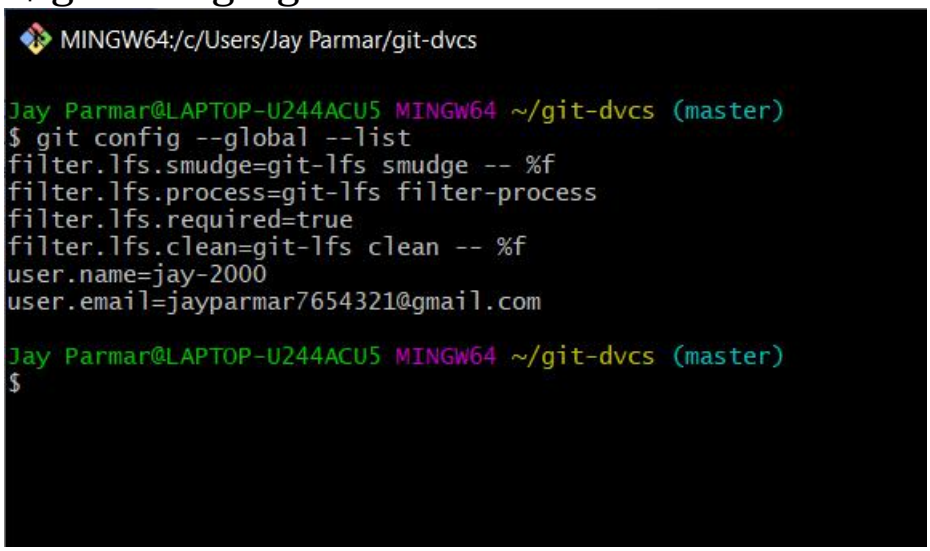
As there are no users defined, let us define it using following two commands

$ git config --global user.name "jay-2000"

$ git config --global user.email “jayparmar7654321@gmail.com”

Now, check the list of users

$ git config --global –list



Let us create a repository for version control named ”git-demo-project”

$ mkdir git-demo-project

$ cd git-demo-project/

Now, initialize the repository using following command

$ git init


Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs (master)
$ mkdir git-demo-project
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs (master)
$ cd git-demo-project/
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git init
Initialized empty Git repository in C:/Users/Jay Parmar/git-dvcs/git-demo-projec
t/.git/
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$

Now, let us add some files inside our repository “git-demo-project”

To add files in index and staging area, add command is used along with dot (. Dot means

current directory)

$ git add .

Index and staging area

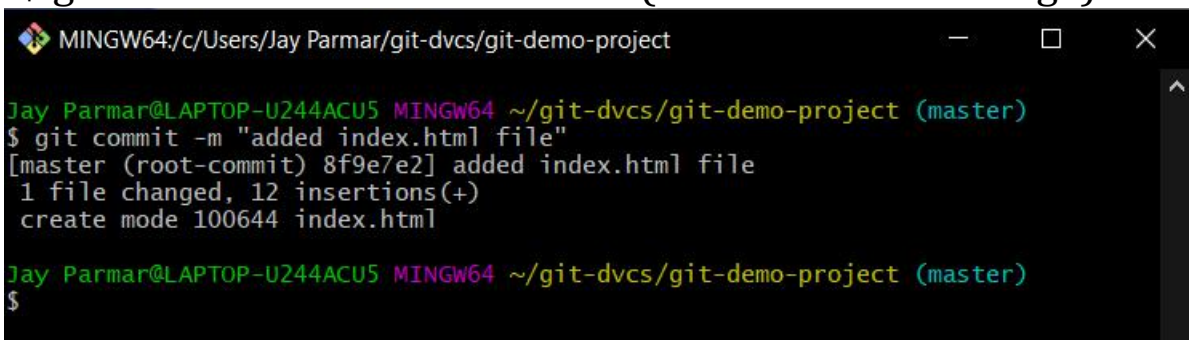
To check the status of repository, use

$ git status

Which will show you some untrack files, so untracks files can be tracked using commit command.

Now, let us commit the changes

$ git commit -m "First Commit" (#here -m for message)



Added index.html in our directory

$ touch teststatus

```
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ touch teststatus

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        teststatus

nothing added to commit but untracked files present (use "git add" to track)

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

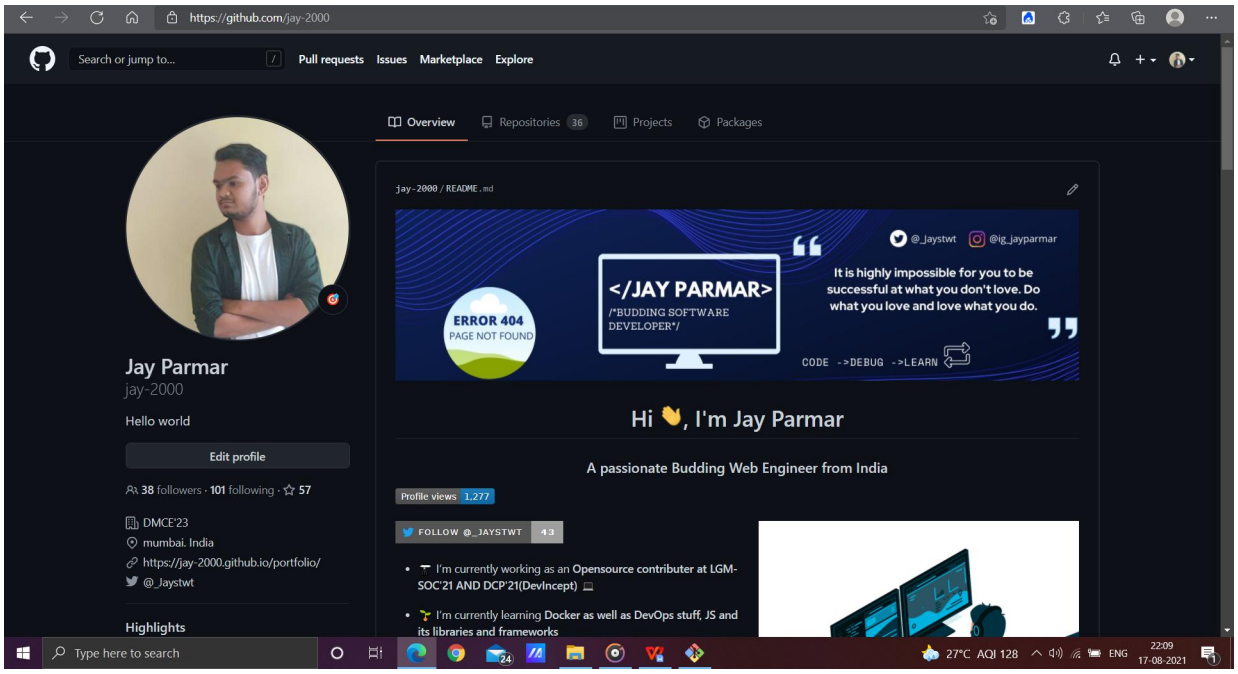
History of Commits

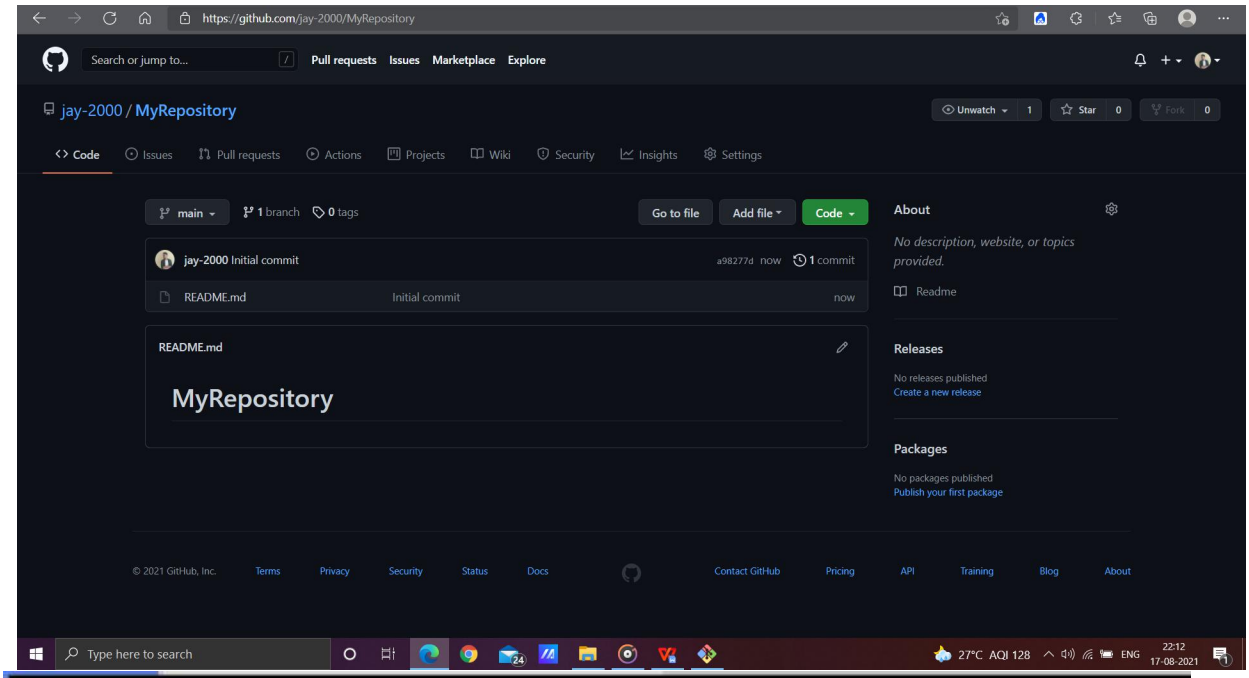
$ git log

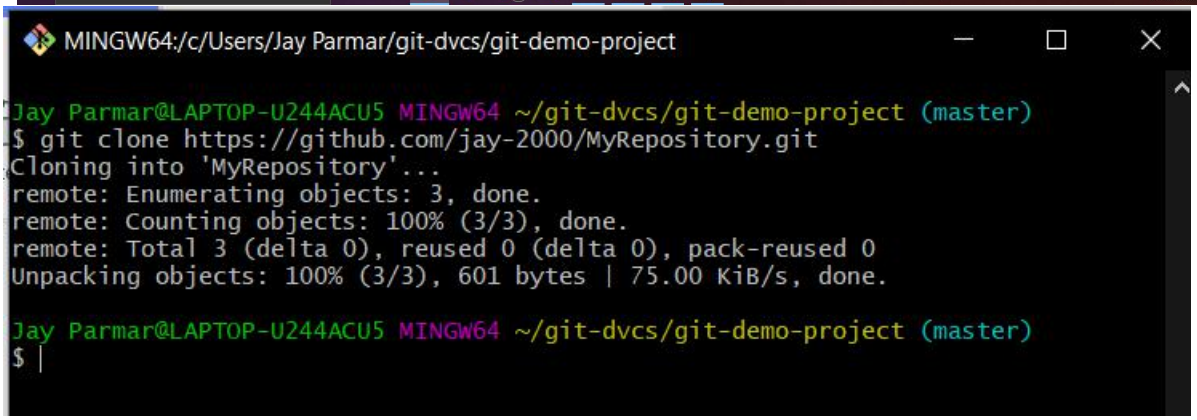
```
MINGW64:/c/Users/Jay Parmar/git-dvcs/git-demo-project

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git log
commit 8f9e7e2db2fb604f654875fe9a2dc30d0313c5c3 (HEAD -> master)
Author: jay-2000 <jayparmar7654321@gmail.com>
Date:   Tue Aug 17 22:03:37 2021 +0530

    added index.html file

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

Now Create a Repository on github.com. Open github.com→ create an account→After login

Select New repository from the menu.


https://github.com/jay-2000
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Overview
Repositories 36
Projects
Packages
Jay Parmar
jay-2000
Hello world
Edit profile
38 followers · 101 following · 57
DMCE'23
mumbai, India
https://jay-2000.github.io/portfolio/
@_Jaystwt
Highlights
jay-2000 / README.md
@_Jaystwt
@ig_jayparmar
ERROR 404
PAGE NOT FOUND
</JAY PARMAR>
/*BUDDING SOFTWARE DEVELOPER*/
It is highly impossible for you to be successful at what you don't love. Do what you love and love what you do.
CODE ->DEBUG ->LEARN
Hi 👋, I'm Jay Parmar
A passionate Budding Web Engineer from India
Profile views 1,277
FOLLOW @_JAYSTWT 43
I'm currently working as an Opensource contributer at LGM-SOC'21 AND DCP'21(DevIncept)
I'm currently learning Docker as well as DevOps stuff, JS and its libraries and frameworks
Type here to search
27°C AQI 128
ENG
22:09
17-08-2021

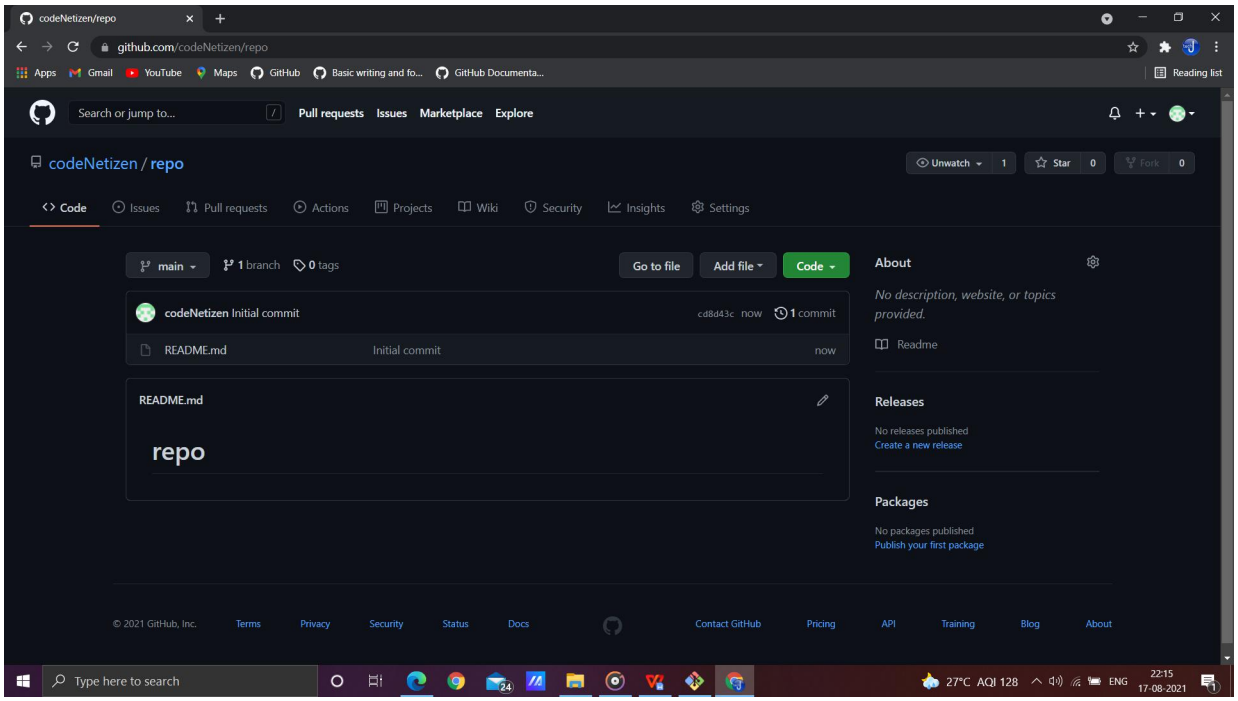


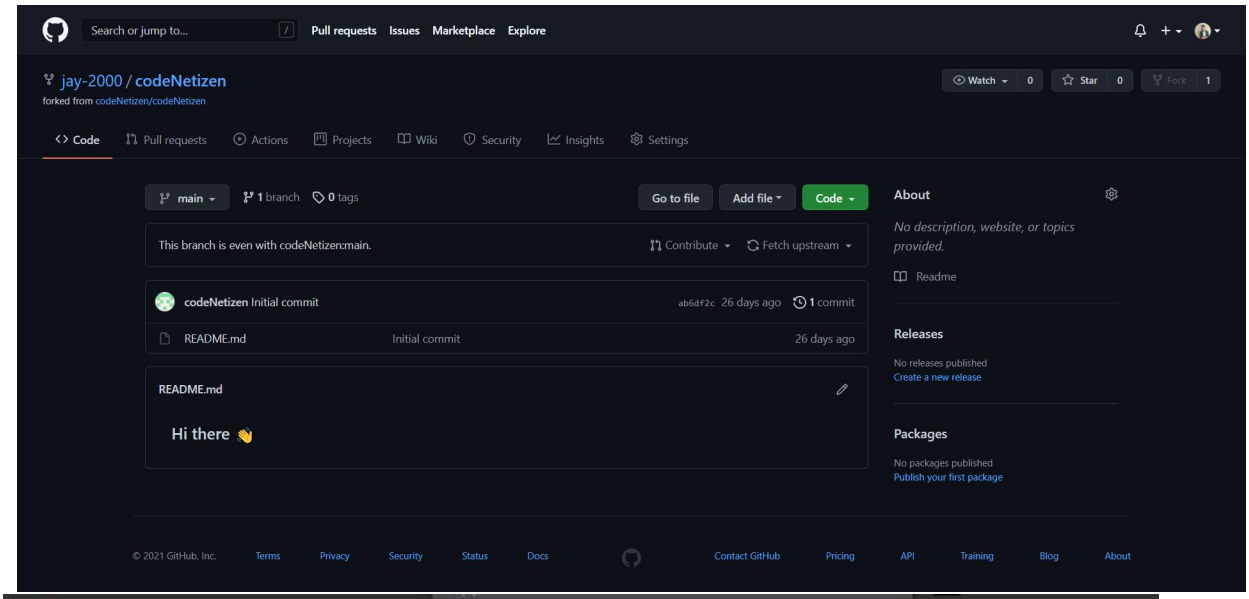


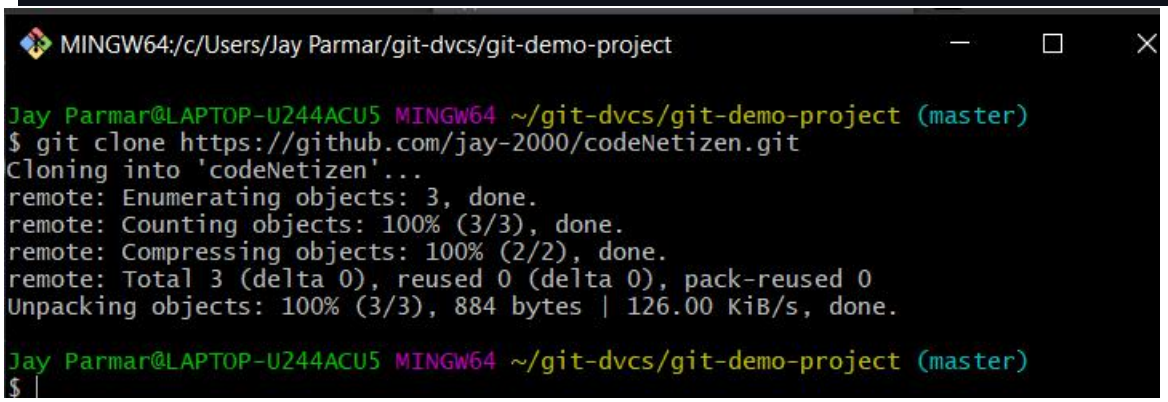
Now fork the repository (Sharing with other users who wants to contribute).

Login with another account→Copy and Paste URL of repository→then just click on fork to clone

to others account.


codeNetizen/repo
github.com/codeNetizen/repo
Search or jump to...
Pull requests
Issues
Marketplace
Explore
codeNetizen / repo
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
codeNetizen Initial commit
cd8d43c now
1 commit
README.md
Initial commit
now
README.md
repo
About
No description, website, or topics provided.
Readme
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2021 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About
Type here to search
27°C AQI 128
ENG
22:15
17-08-2021





Pull and Push Processes

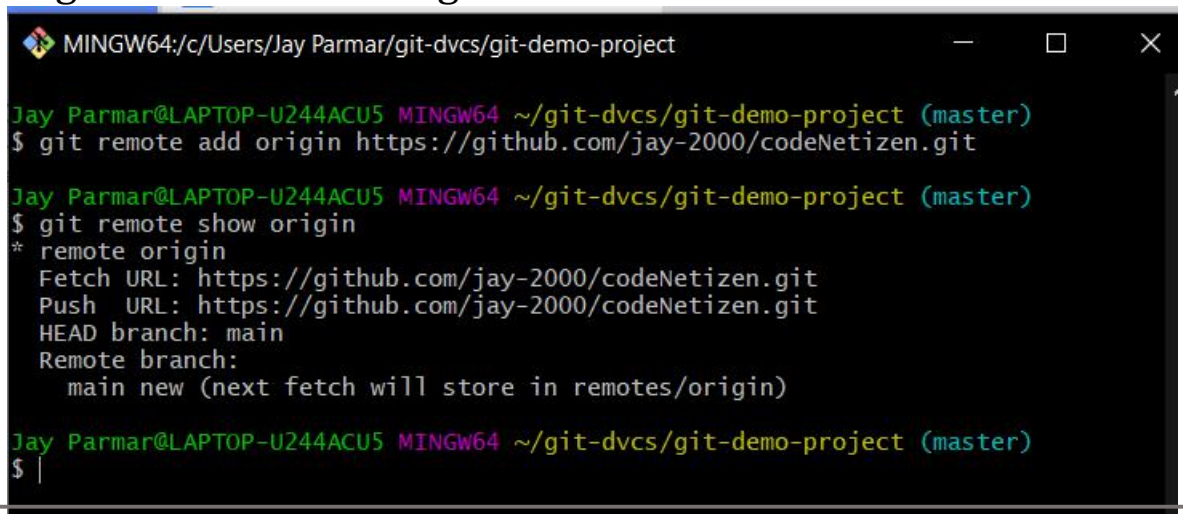
Push → Push changes to Web repository

Pull → Pull changes to Local repository

1) Push command to remote reference origin master

$ git remote add origin https://github.com/bhushanjadhav1/siesworkshop.git

$ git remote show origin



```
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git pull https://github.com/jay-2000/codeNetizen.git
warning: no common commits
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 884 bytes | 68.00 KiB/s, done.
From https://github.com/jay-2000/codeNetizen
 * branch            HEAD       -> FETCH_HEAD
fatal: refusing to merge unrelated histories

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
```

MINGW64:/c/Users/Jay Parmar/git-dvcs/git-demo-project

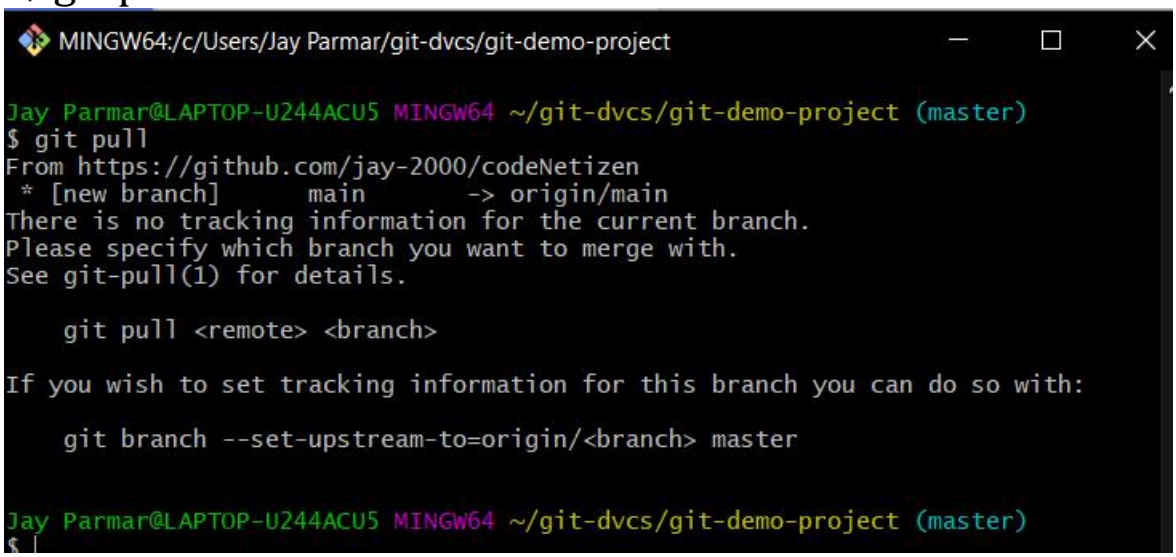
```
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        MyRepository/
        codeNetizen/
        teststatus

nothing added to commit but untracked files present (use "git add" to track)

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

MINGW64:/c/Users/Jay Parmar/git-dvcs/git-demo-project

```
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git remote -v
origin  https://github.com/jay-2000/codeNetizen.git (fetch)
origin  https://github.com/jay-2000/codeNetizen.git (push)

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

```
Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$ git push -v origin master
Pushing to https://github.com/jay-2000/codeNetizen.git
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 400 bytes | 400.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
POST git-receive-pack (563 bytes)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/jay-2000/codeNetizen/pull/new/master
remote:
To https://github.com/jay-2000/codeNetizen.git
 * [new branch]      master -> master
updating local tracking ref 'refs/remotes/origin/master'

Jay Parmar@LAPTOP-U244ACU5 MINGW64 ~/git-dvcs/git-demo-project (master)
$
```

$ git pull



Conclusion:

Hence studied and practiced GIT commands for version control.

EXPERIMENT NO : 03

AIM : To practice/execute shell programs and parameterized 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.

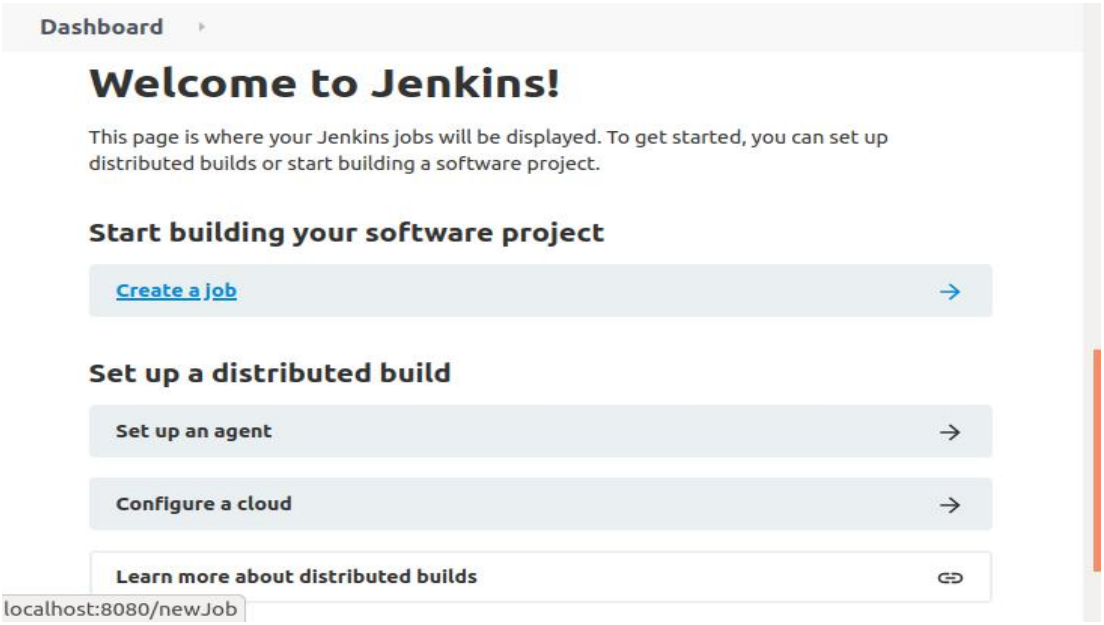
Some of the possible steps that can be performed using Jenkins are:

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

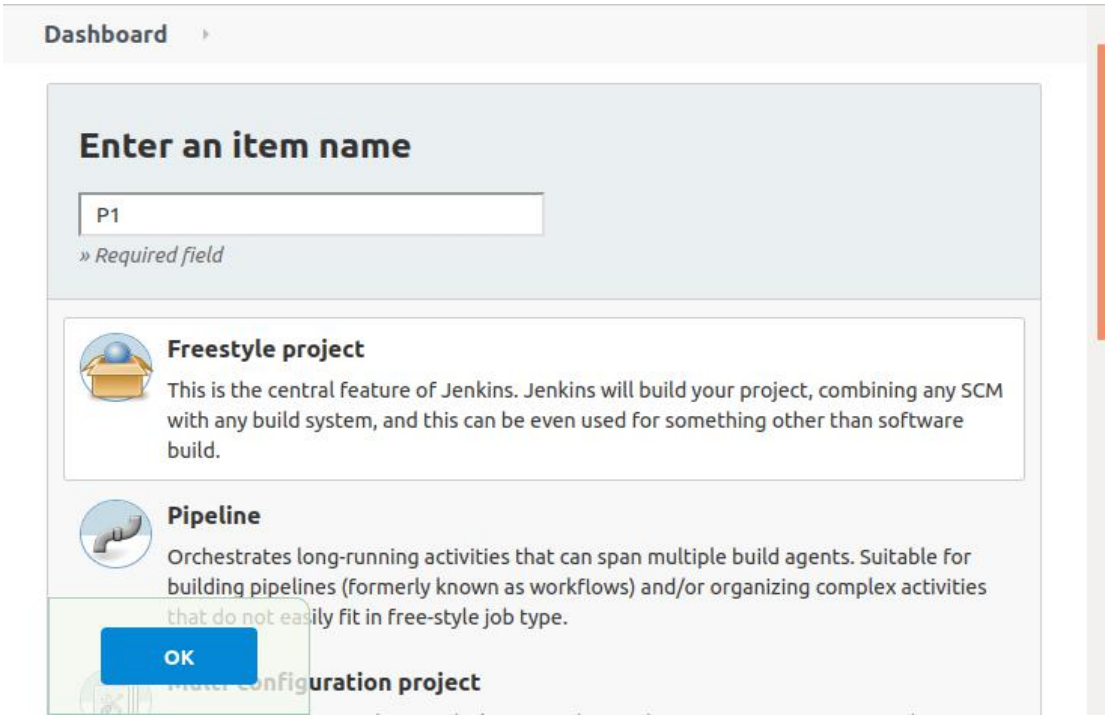
- Automation testing using test frameworks such as Nose2, PyTest, Robot, Selenium, and more.

TO PRACTISE/EXECUTE SHELL PROGRAMS USING JENKINS

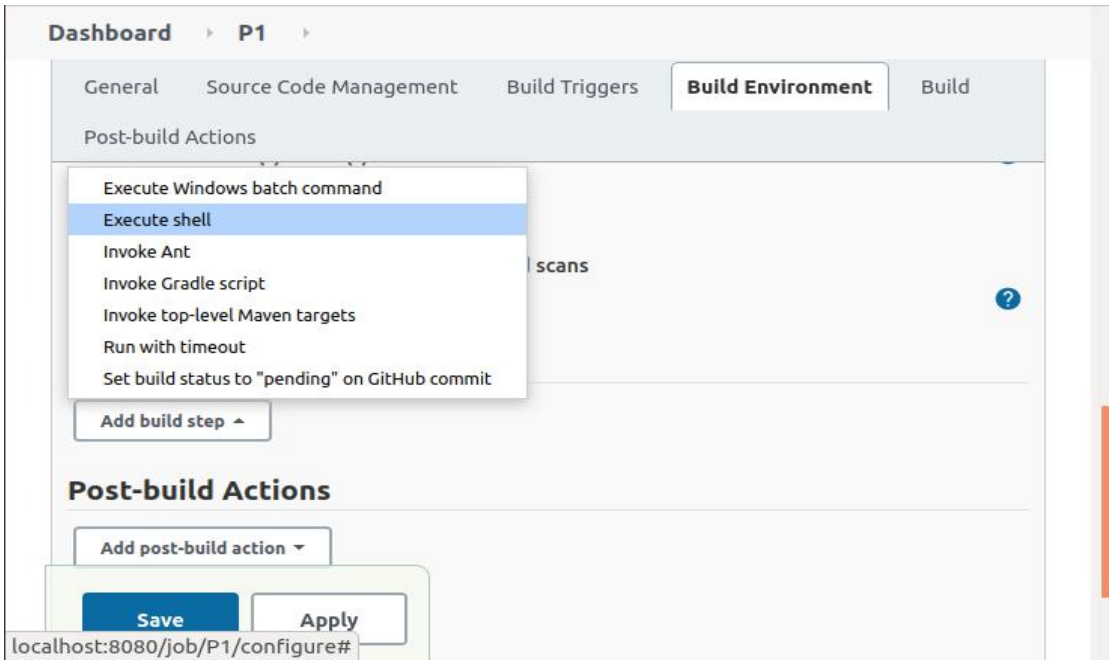
Step 1 : Click on Create new jobs



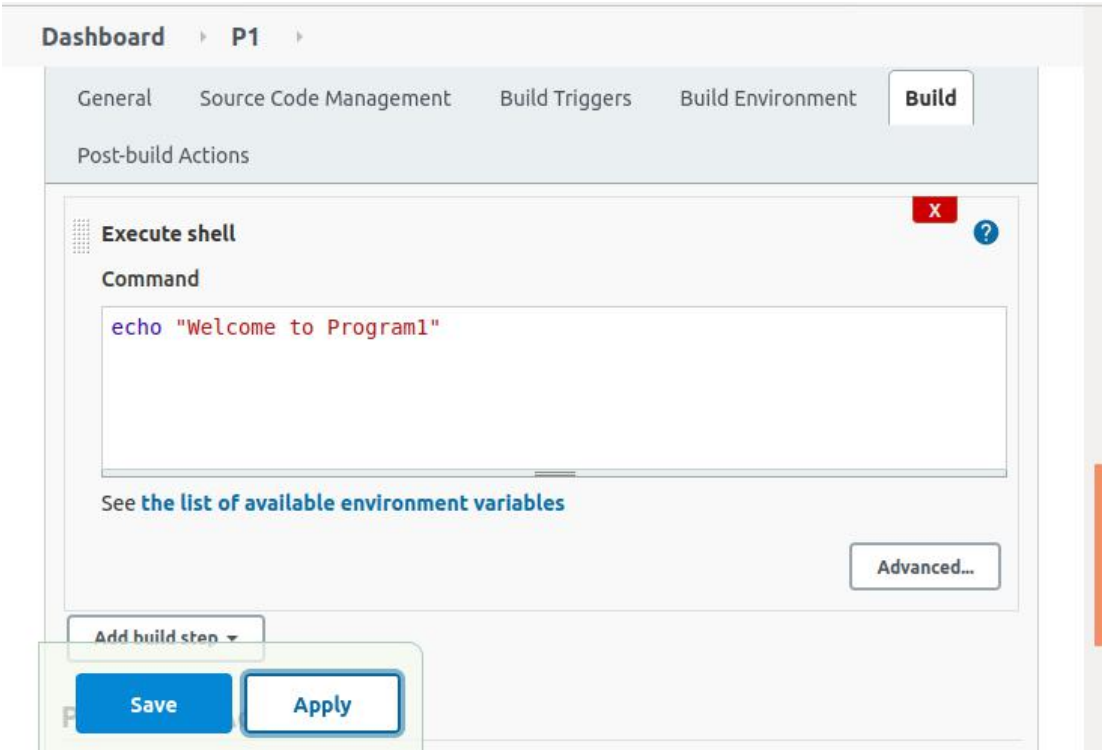
Step 2 : Give a name to project as “P1”, select Option “Free style project” and click on OK button



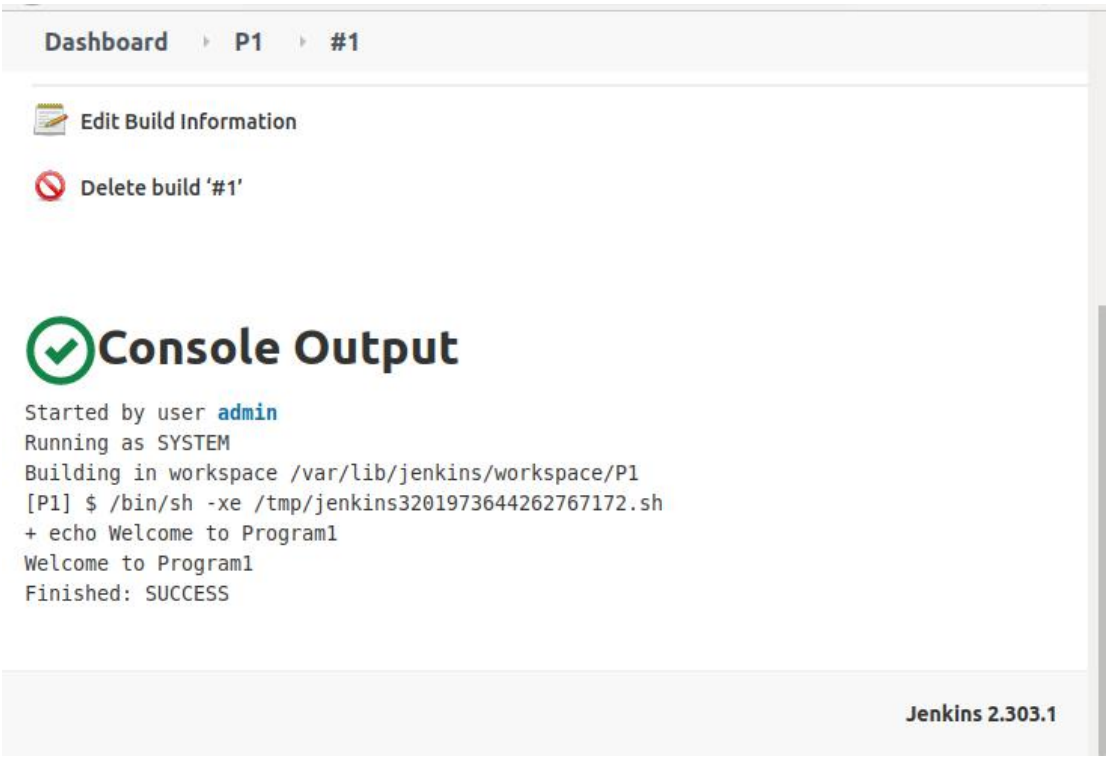
Step 3 : To run simple shell scripts on Jenkins click on Build option select the Execute script from dropdown menu



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

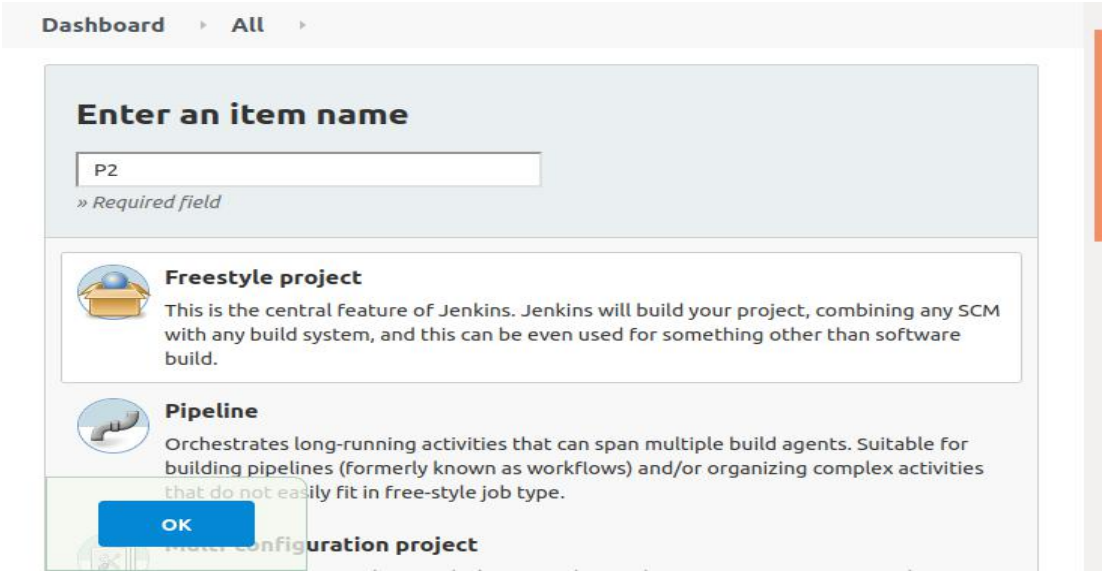


Step 5 : Click on first build “1” followed by console output to see the output

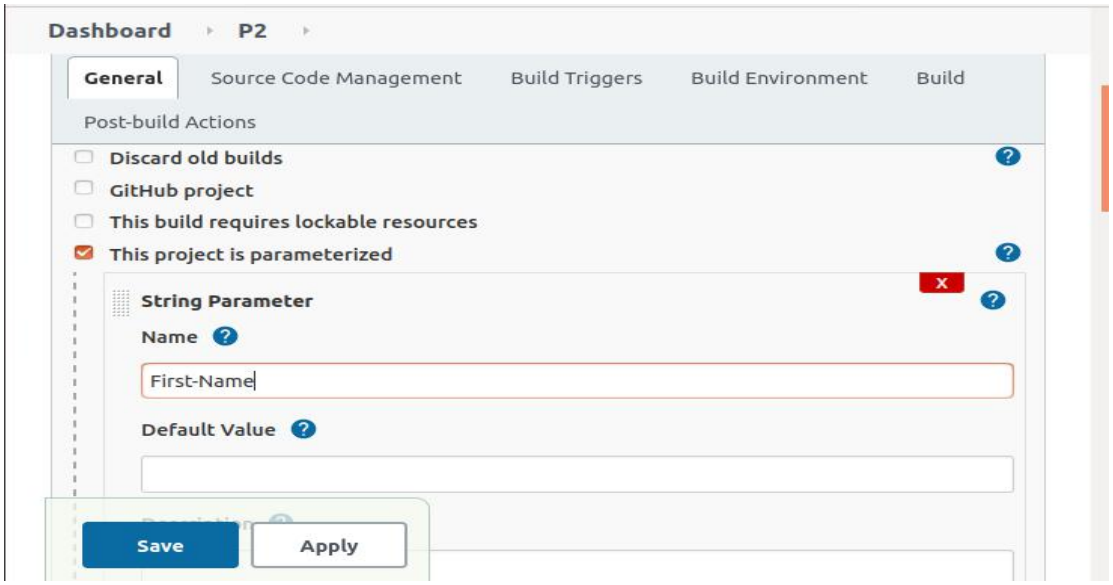


TO PRACTISE/EXECUTE PARAMETERISED JAVA PROGRAMS USING JENKINS

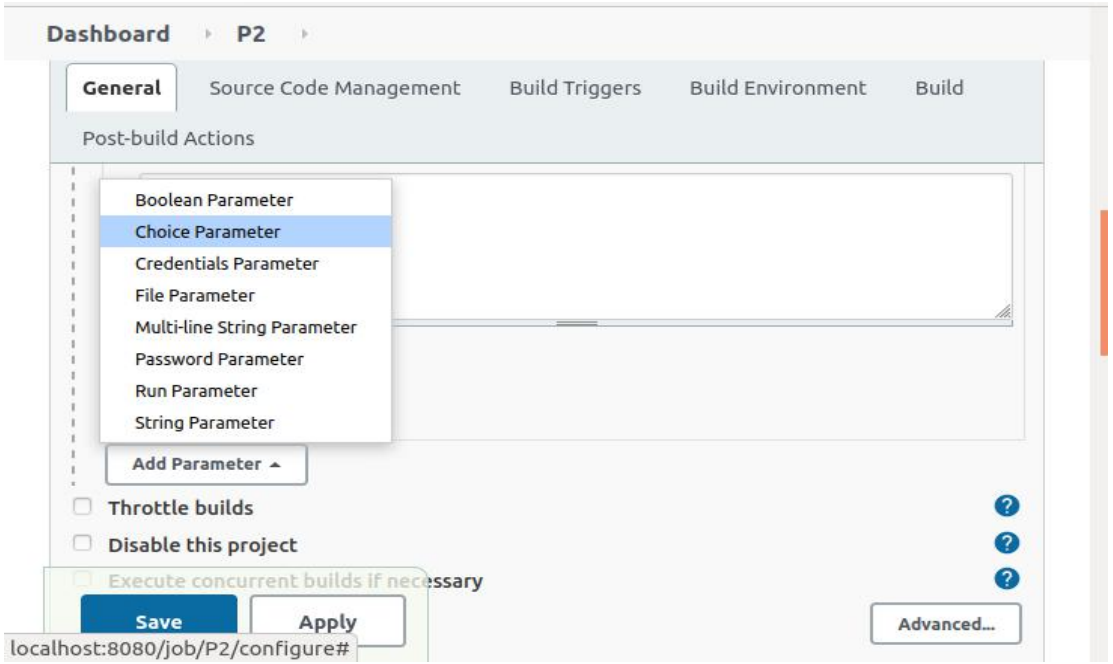
Step 1 : Create a freestyle project P2 in Jenkins



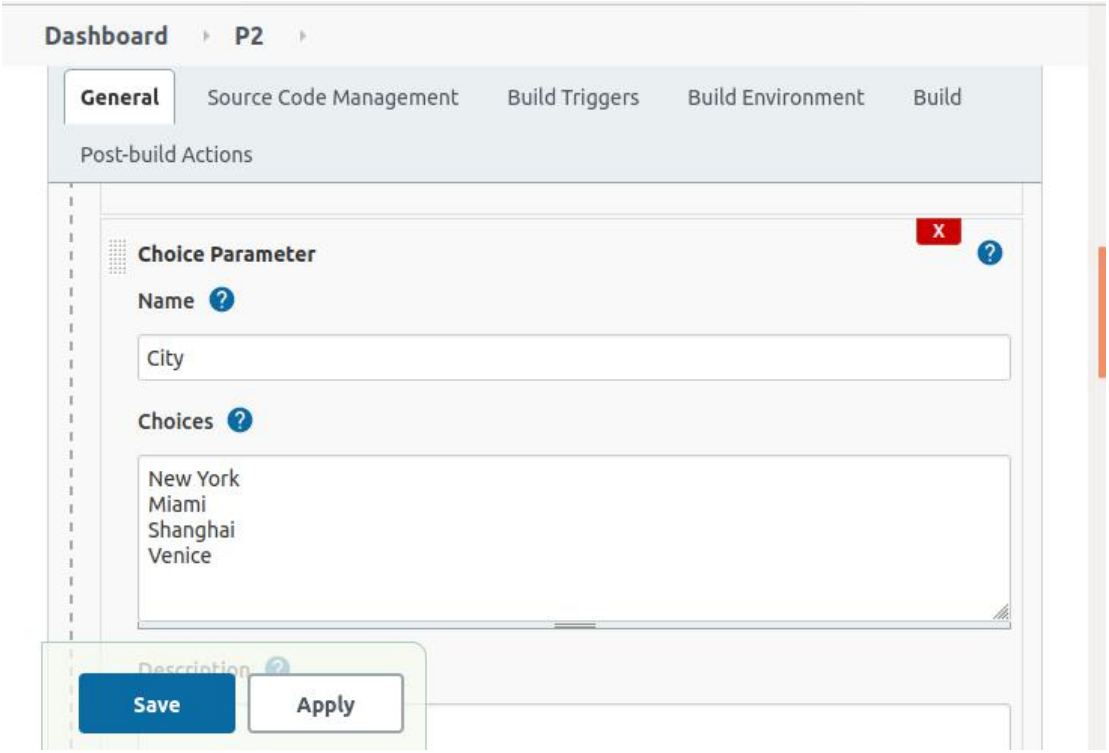
Step 2 : Click on general menu and select option this project is parameterize. Select String parameter and specify name as “First-Name”



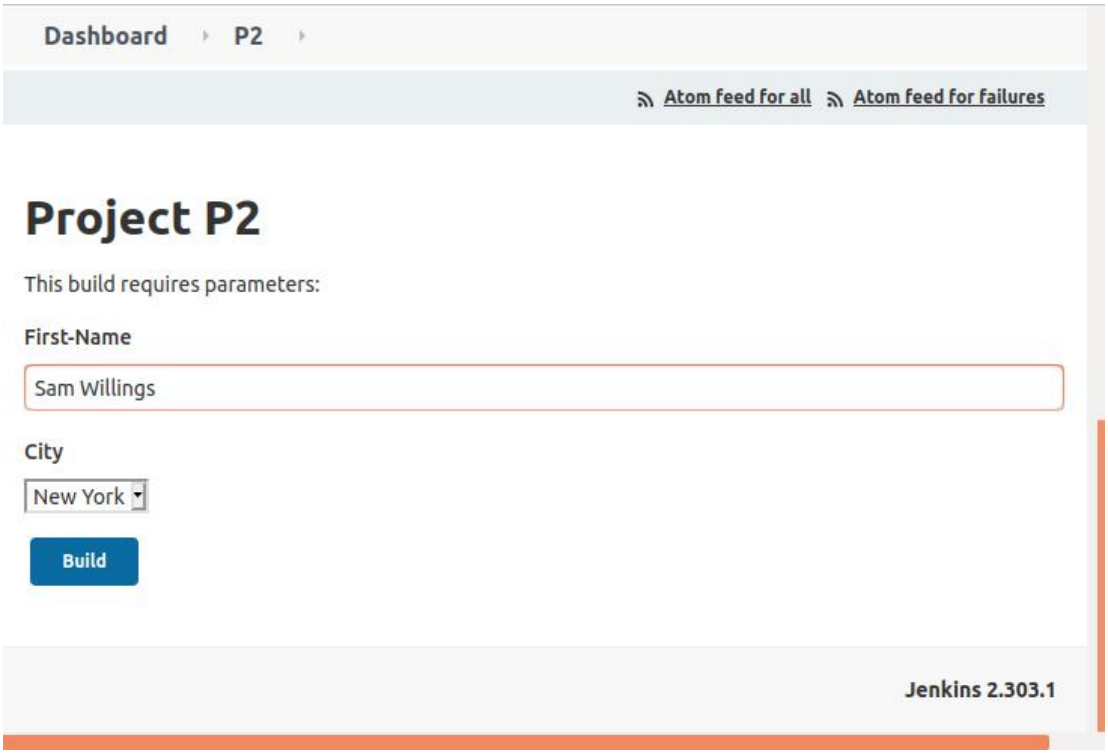
Step 3 : Click on add parameter and select choice parameter. Take second parameter as choice parameter



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



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



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

EXPERIMENT NO : 04

AIM : To understand continuous integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build job

THEORY :

MAVEN

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta project.

ANT

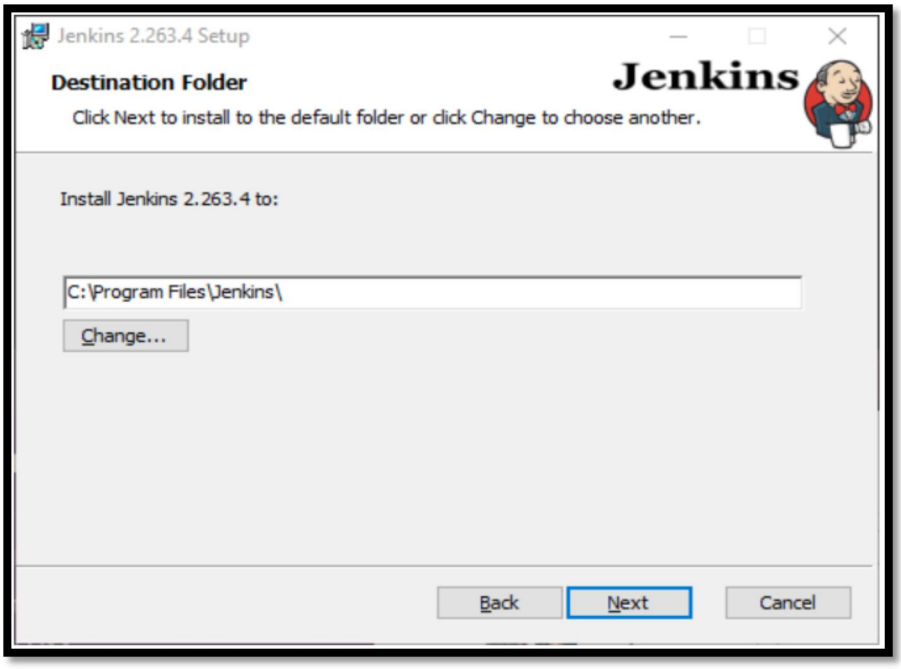
Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extensions points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non-Java applications, for instance C or C++ applications.

STEPS FOR INSTALLING AND SETUP OF JENKINS:

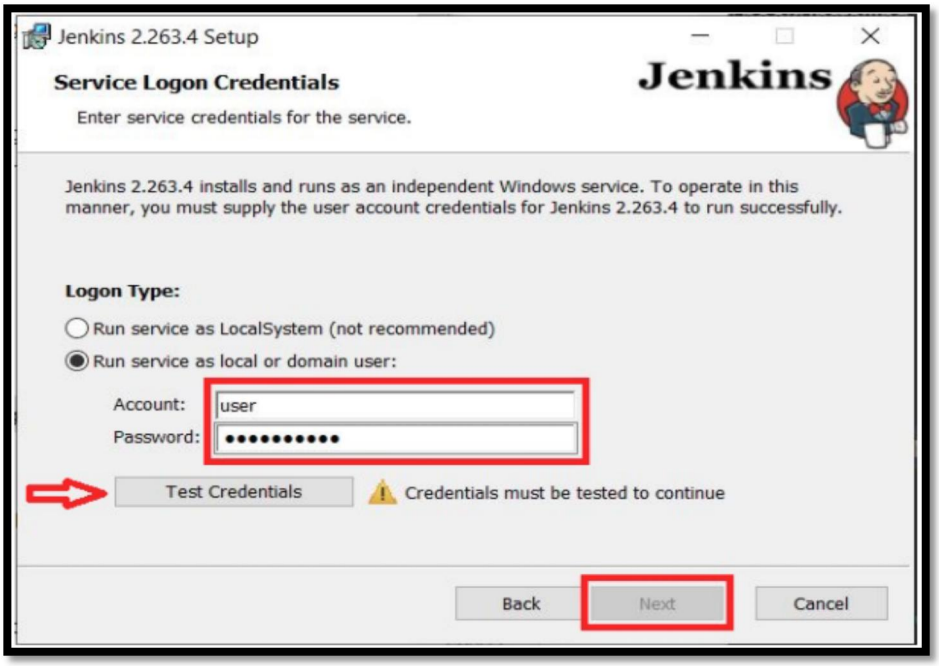
Installing Steps:

Step 1:

Step 2:



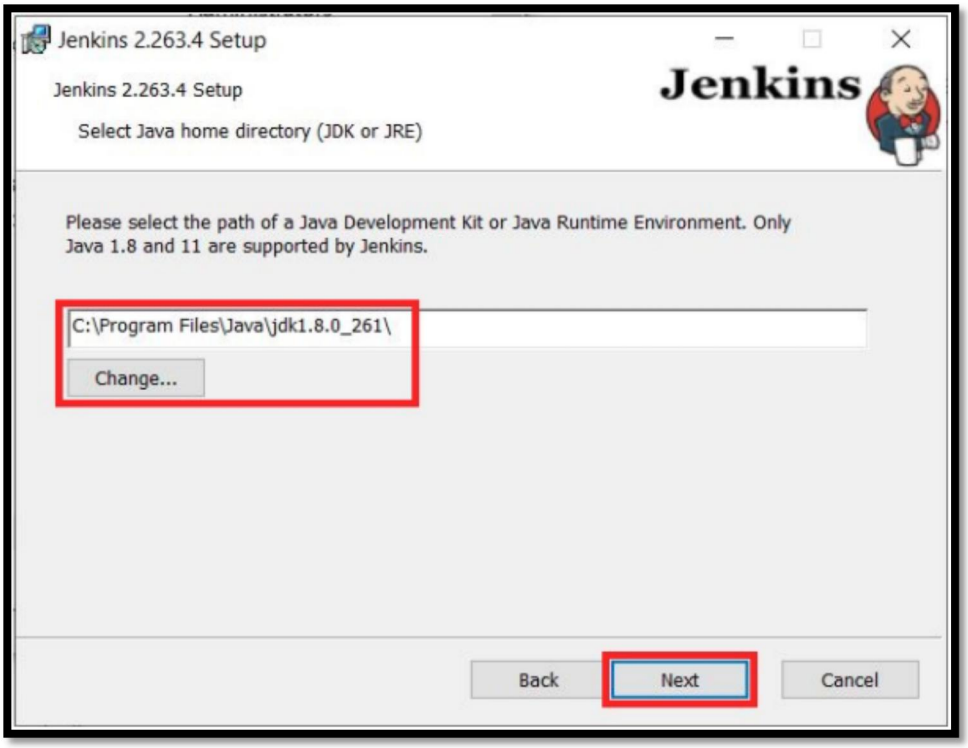
Step 3:


Jenkins 2.263.4 Setup
Service Logon Credentials
Enter service credentials for the service.
Jenkins
Jenkins 2.263.4 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.263.4 to run successfully.
Logon Type:
Run service as LocalSystem (not recommended)
Run service as local or domain user:
Account:
user
Password:
Test Credentials
Credentials must be tested to continue
Back
Next
Cancel

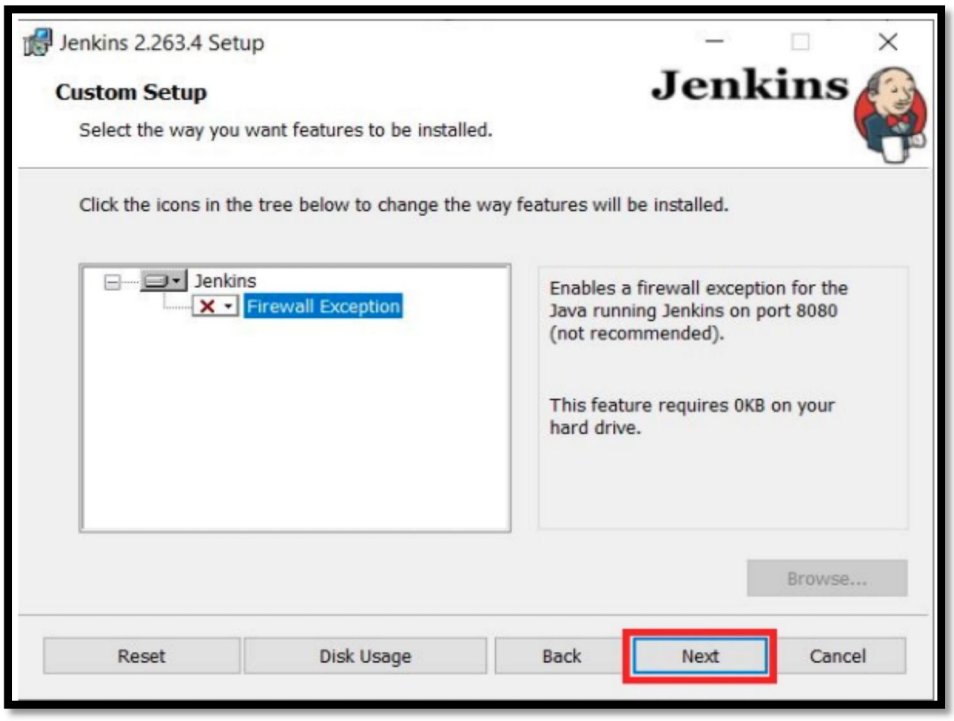
Step 4:


Jenkins 2.263.4 Setup
Port Selection
Choose a port for the service.
Jenkins
Please choose a port.
Port Number (1-65535):
8080
Test Port
It is recommended that you accept the selected default port.
Back
Next
Cancel

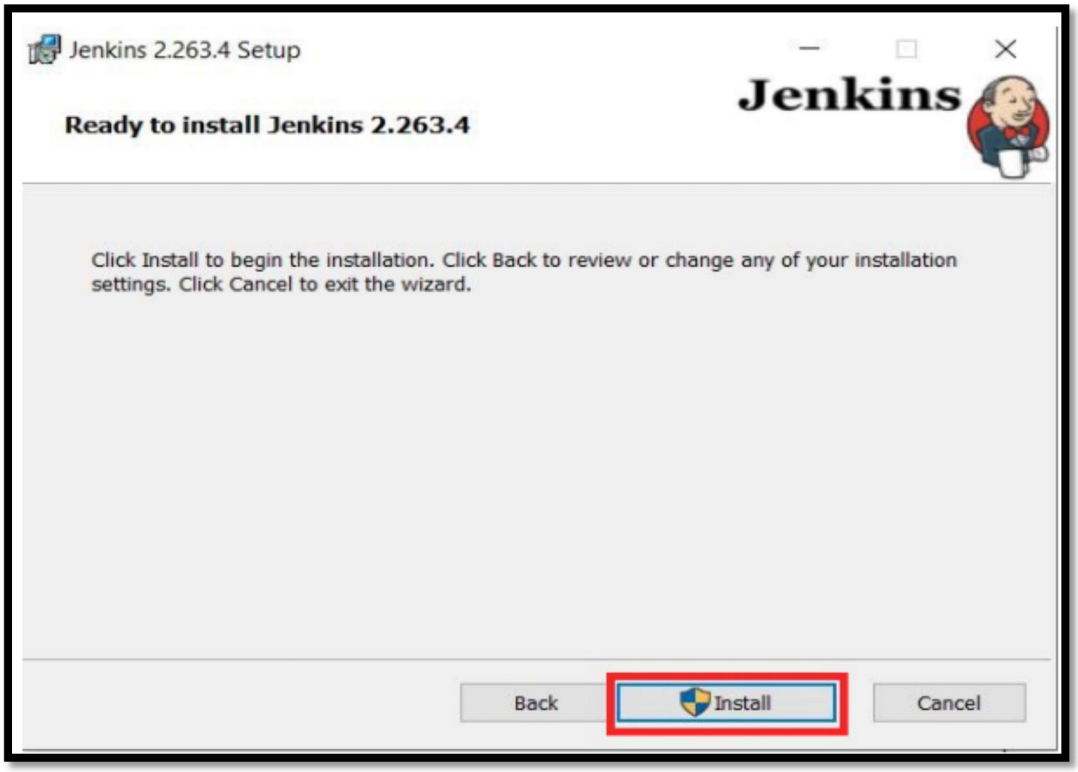
Step 5:


Jenkins 2.263.4 Setup
Jenkins 2.263.4 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\jdk1.8.0_261\
Change...
Back
Next
Cancel

Step 6:

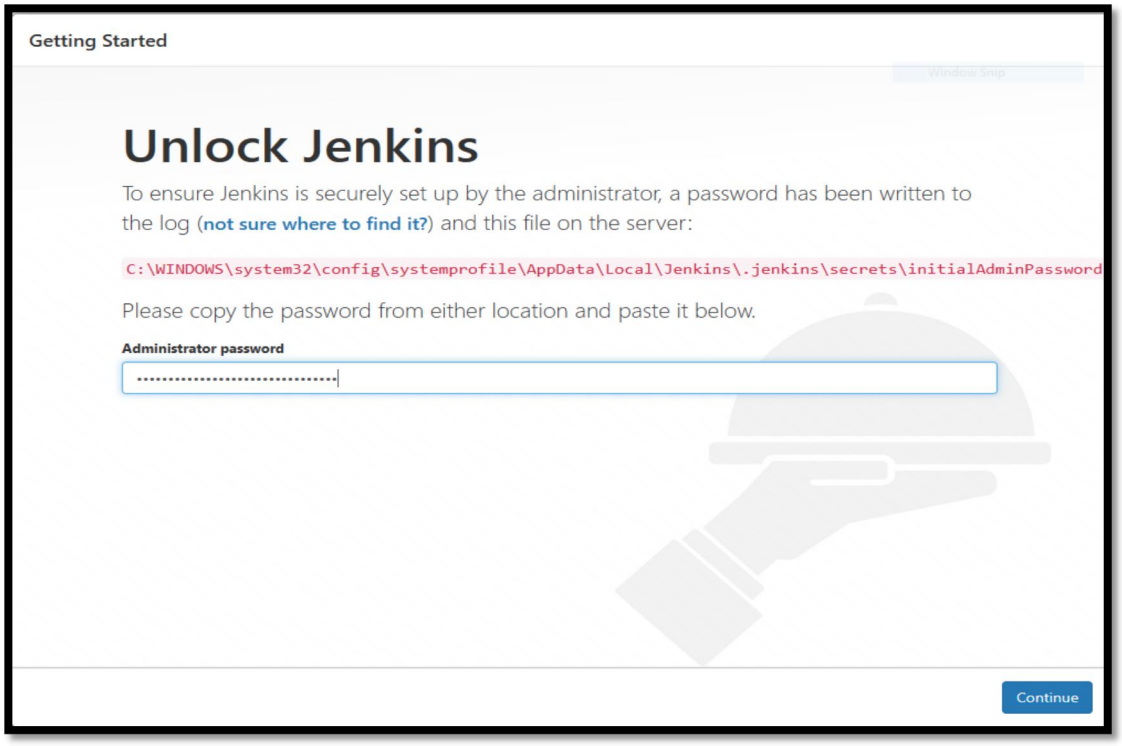

Jenkins 2.263.4 Setup
Custom Setup
Select the way you want features to be installed.
Jenkins
Click the icons in the tree below to change the way features will be installed.
Jenkins
Firewall Exception
Enables a firewall exception for the Java running Jenkins on port 8080 (not recommended).
This feature requires 0KB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel

Step 7:

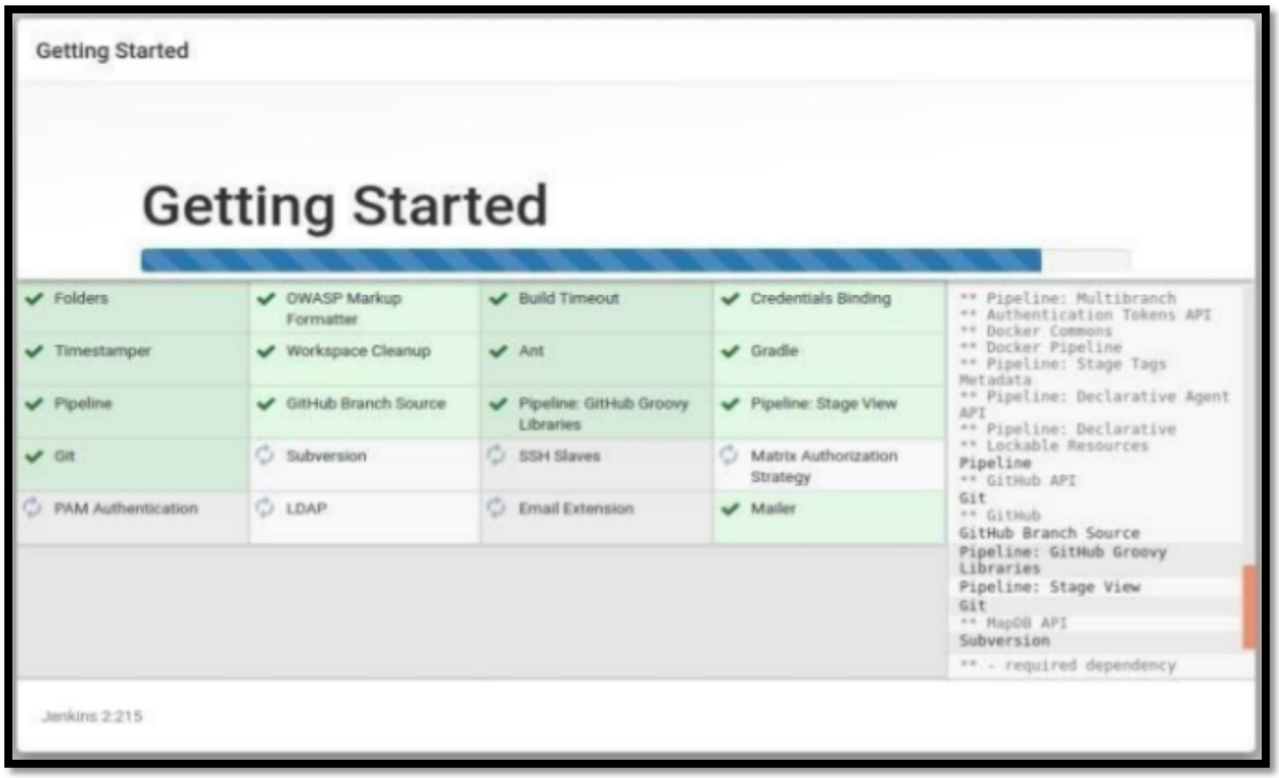

Jenkins 2.263.4 Setup
Jenkins
Ready to install Jenkins 2.263.4
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel


Jenkins 2.263.4 Setup
Jenkins
Installing Jenkins 2.263.4
Please wait while the Setup Wizard installs Jenkins 2.263.4.
Status:
Back
Next
Cancel

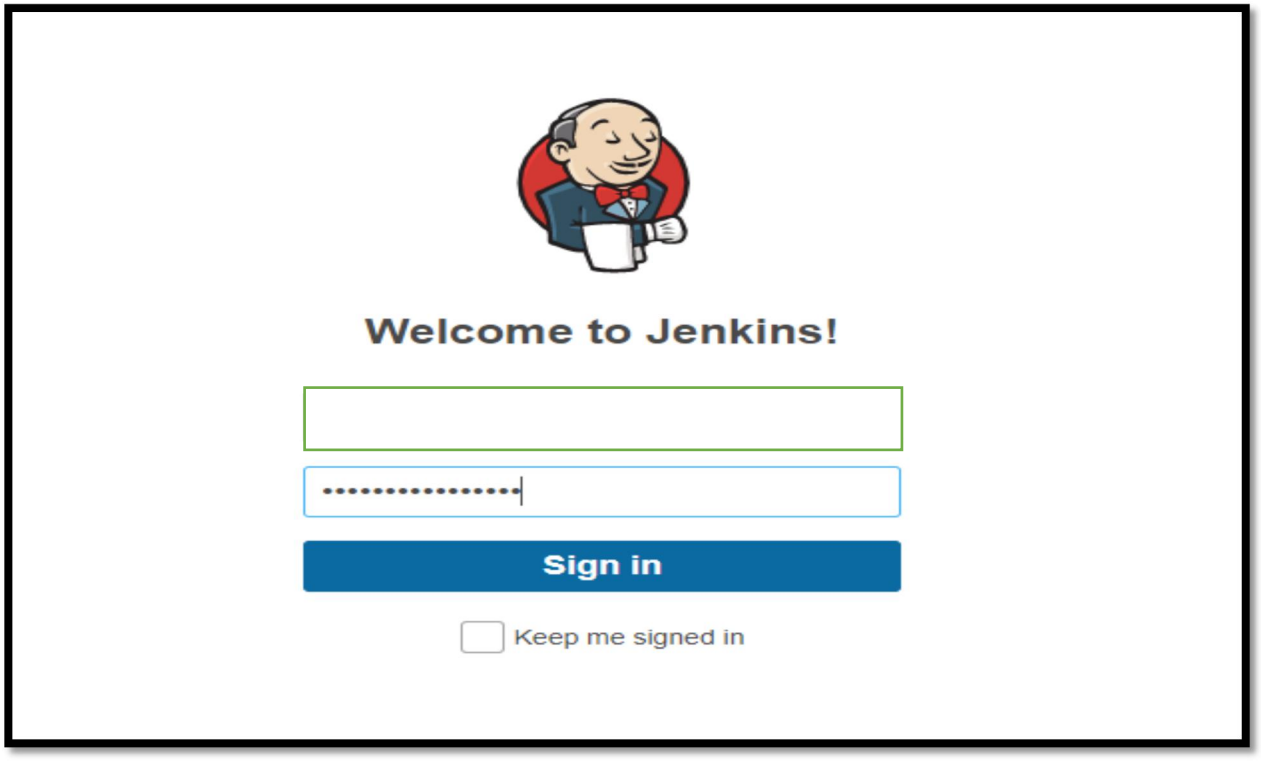
UNLOCKING JENKINS



CUSTOMIZE JENKINS WITH PLUGINS


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
Subversion
SSH Slaves
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** Pipeline: Multibranch
** Authentication Tokens API
** Docker Commons
** Docker Pipeline
** Pipeline: Stage Tags Metadata
** Pipeline: Declarative Agent API
** Pipeline: Declarative
** Lockable Resources
Pipeline
** GitHub API
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
** MapDB API
Subversion
** - required dependency
Jenkins 2.215

USER LOGIN:


Welcome to Jenkins!
Sign in
Keep me signed in

MANAGE JENKINS:

ADD JDK:

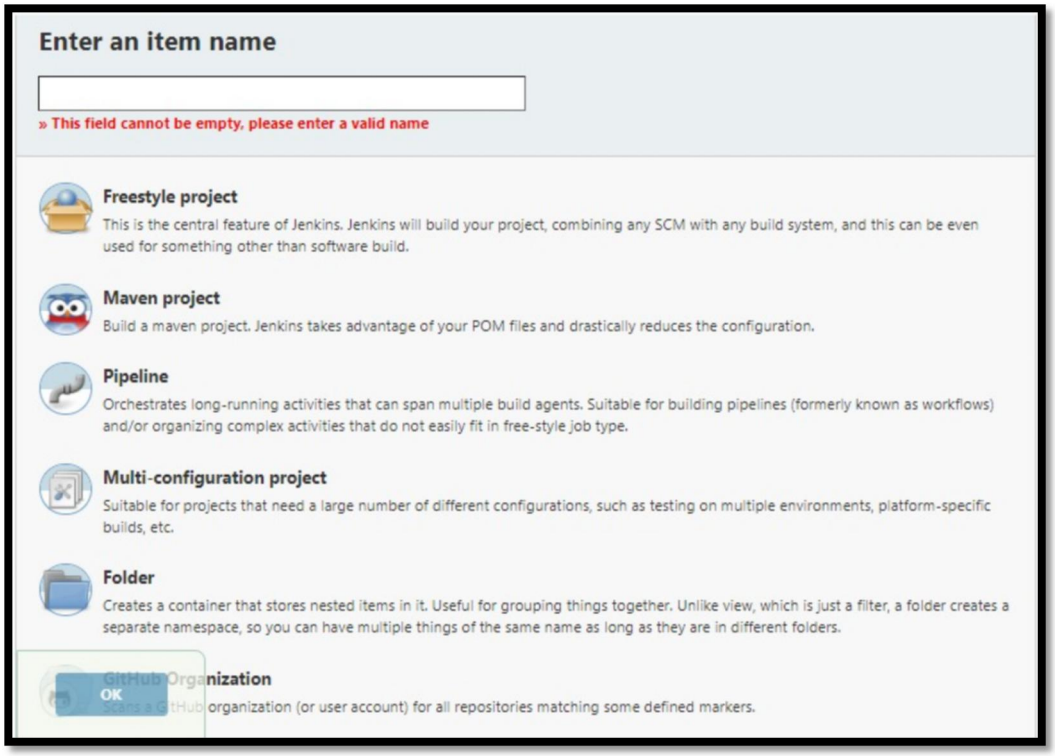
ADD GIT:

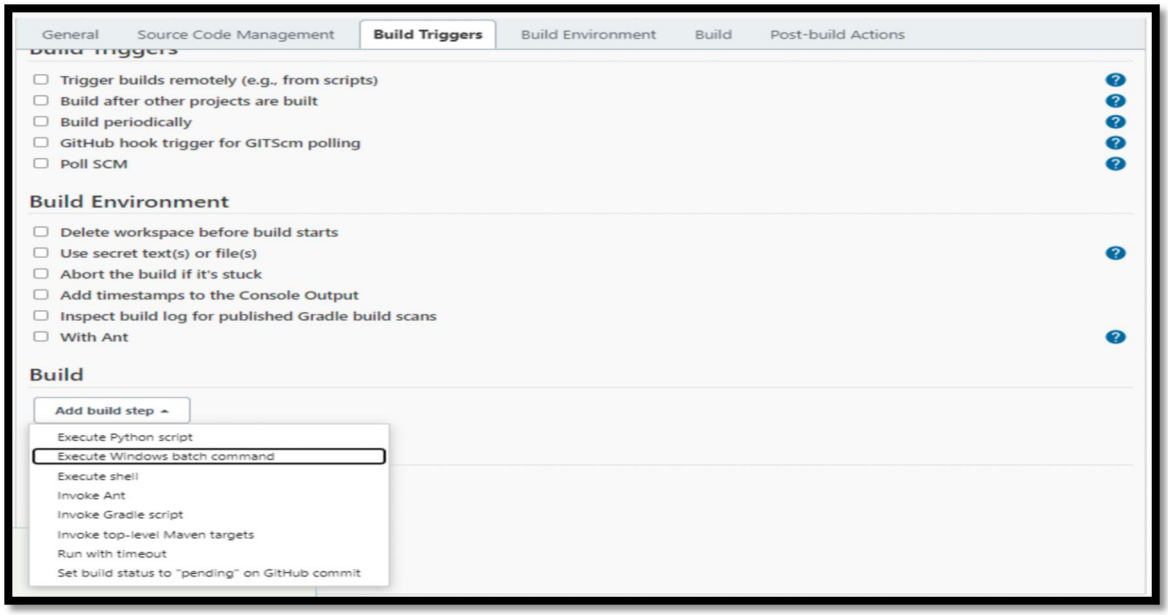
ADD GRADLE:

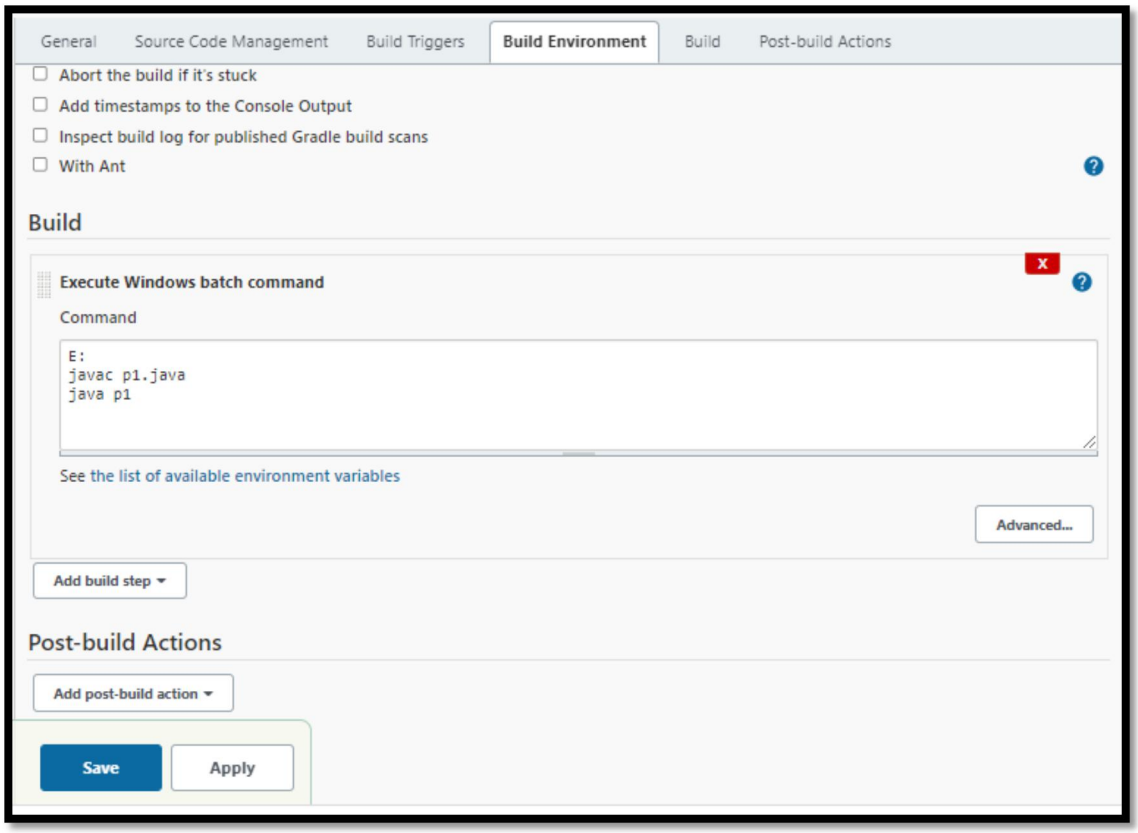
ADD ANT:

ADD MAVEN:

A NEW BULID JOB IN JENKINS:






General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Abort the build if it's stuck
Add timestamps to the Console Output
Inspect build log for published Gradle build scans
With Ant
Build
Execute Windows batch command
Command
E:
javac p1.java
java p1
See the list of available environment variables
Advanced...
Add build step
Post-build Actions
Add post-build action
Save
Apply


Dashboard
demo
Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Build History
trend
find
Atom feed for all
Atom feed for failures
Project demo
add description
Disable Project
Workspace
Recent Changes
Permalinks

CONCLUSION : Hence, we understood Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to setup a build Job.

EXPERIMENT NO. 5

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

Jenkins Pipeline is a combination of plugins that supports integration and implementation of continuous delivery pipelines. It has an extensible automation server to create simple and complex delivery pipelines as code via pipeline DSL. A Pipeline is a group of events interlinked with each other in a sequence.

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

The benefits of using Jenkins are:

- You can create pipelines automatically for all branches and execute pull requests with just one Jenkins.
- You can review your Jenkins code on the pipeline
- You can audit your Jenkins pipeline
- This is the singular source for your pipeline and can be modified by multiple users.

Maven Jenkins

Maven is used to define project structure, dependencies, build, and test management.

Using pom. xml(Maven) you can configure dependencies needed for building testing and running code.

Maven automatically downloads the necessary files from the repository while building the project.

Purpose of Maven :

- A maven is a build tool designed to manage dependencies and the software lifecycle. It is also designed to work with plugins that allow users to add other tasks to the standard compile, test, package, install, deploy tasks.
- Jenkins is designed for the purpose of implementing Continuous Integration (CI). It checks code out of a repository, builds and packages it, and sends it out to a server for testing – automatically. Jenkins can use Maven as its build tool.

Gradle Jenkins :

Gradle is managed as another tool inside Jenkins (the same way as Ant or Maven), including support for automatic installation and a new build step is provided to execute Gradle tasks.

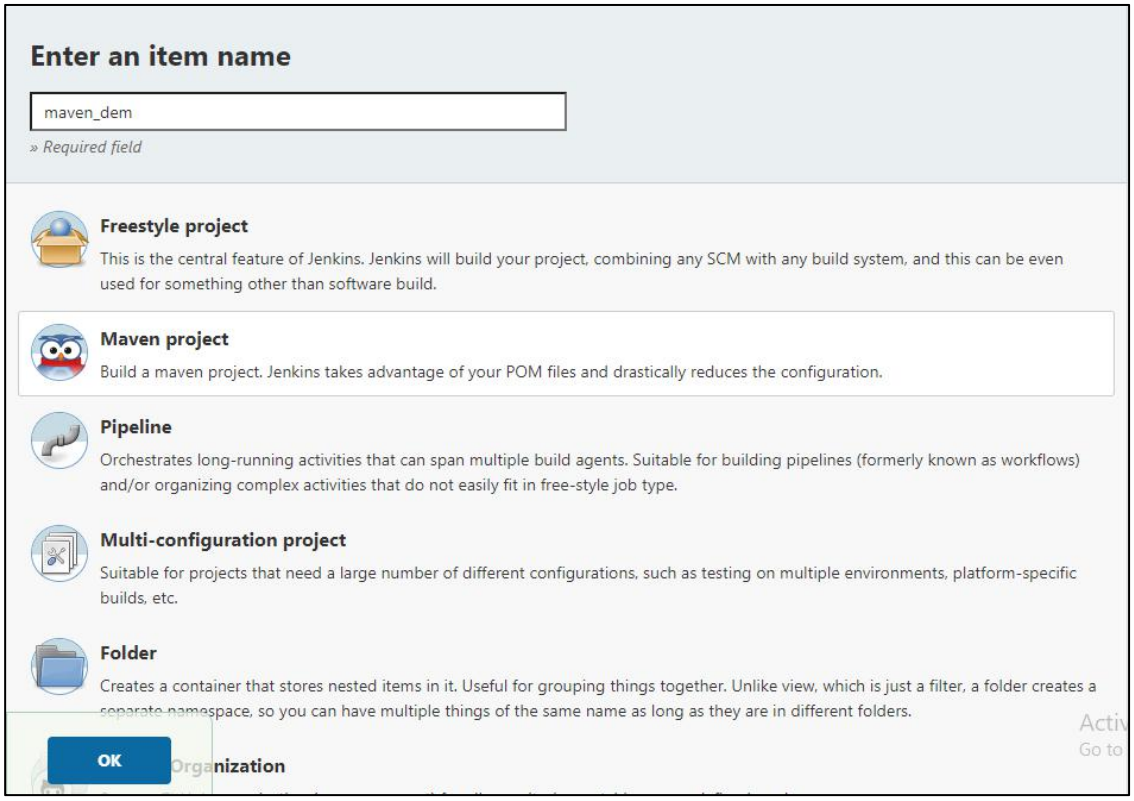
It also allows detecting Build Scans in arbitrary console logs, for Maven and Gradle builds and display them in the Jenkins UI. It is a powerful build tool for the JVM.

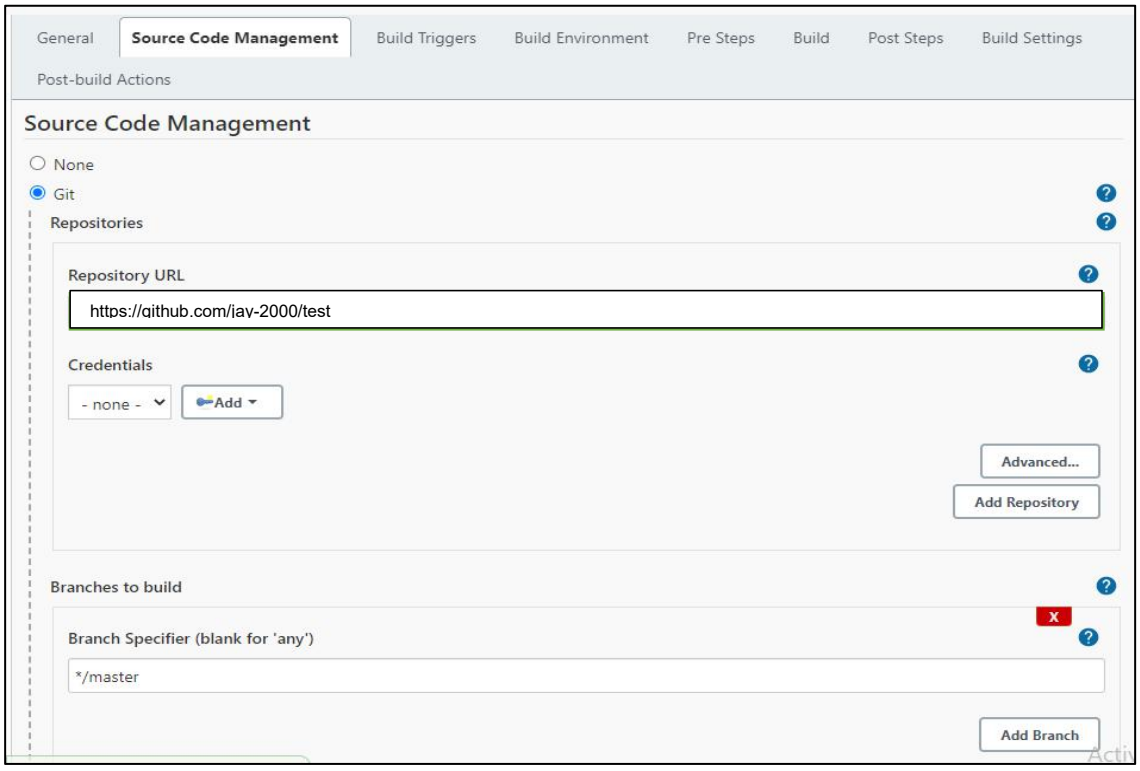
It primarily focuses on build automation and supports multi-language development.

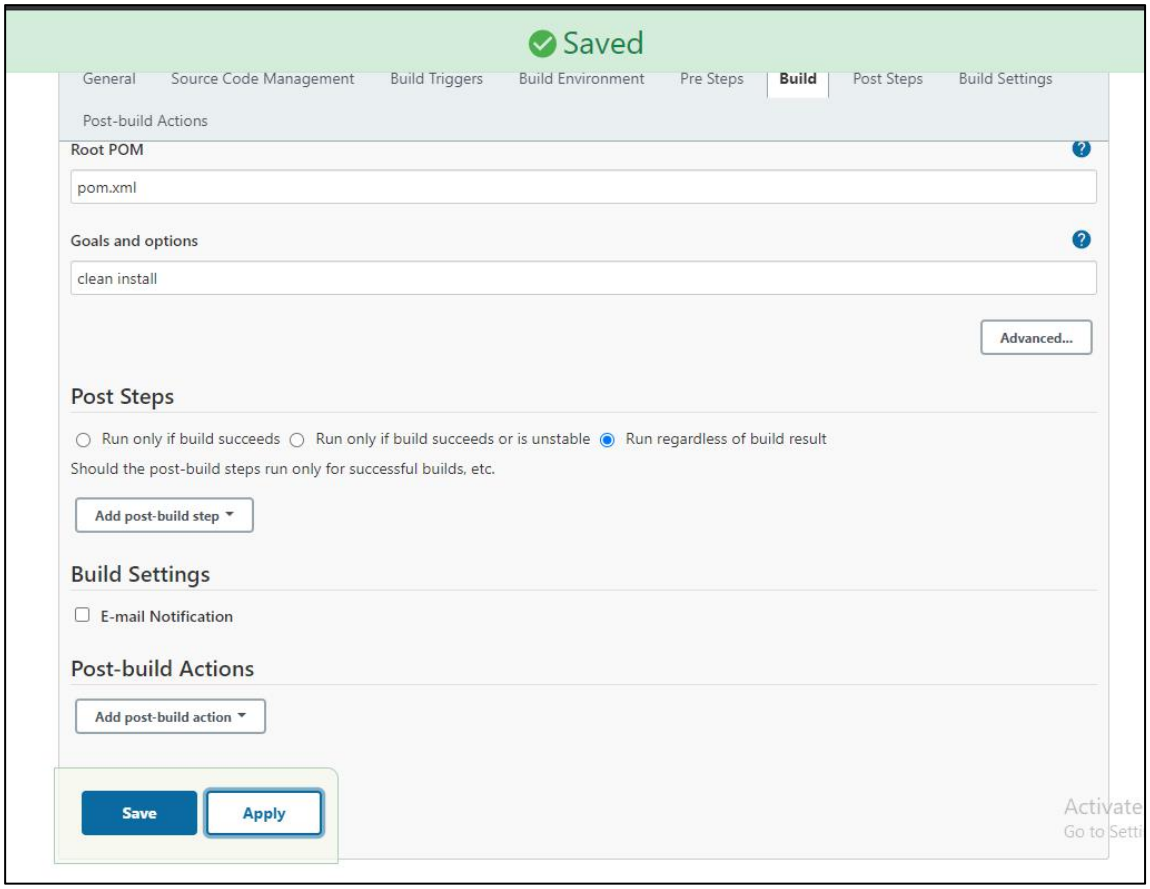
If we are building, testing, publishing, and deploying software on any platform, Gradle provides a flexible model to support the entire development lifecycle from compiling and deploying the project.

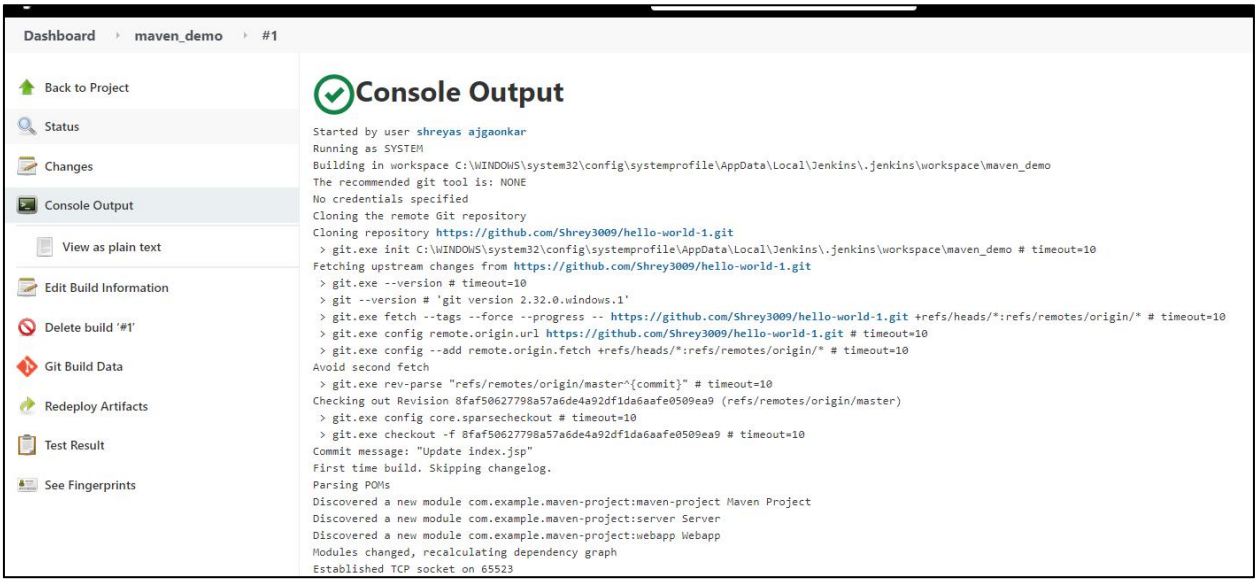
OUTPUT :

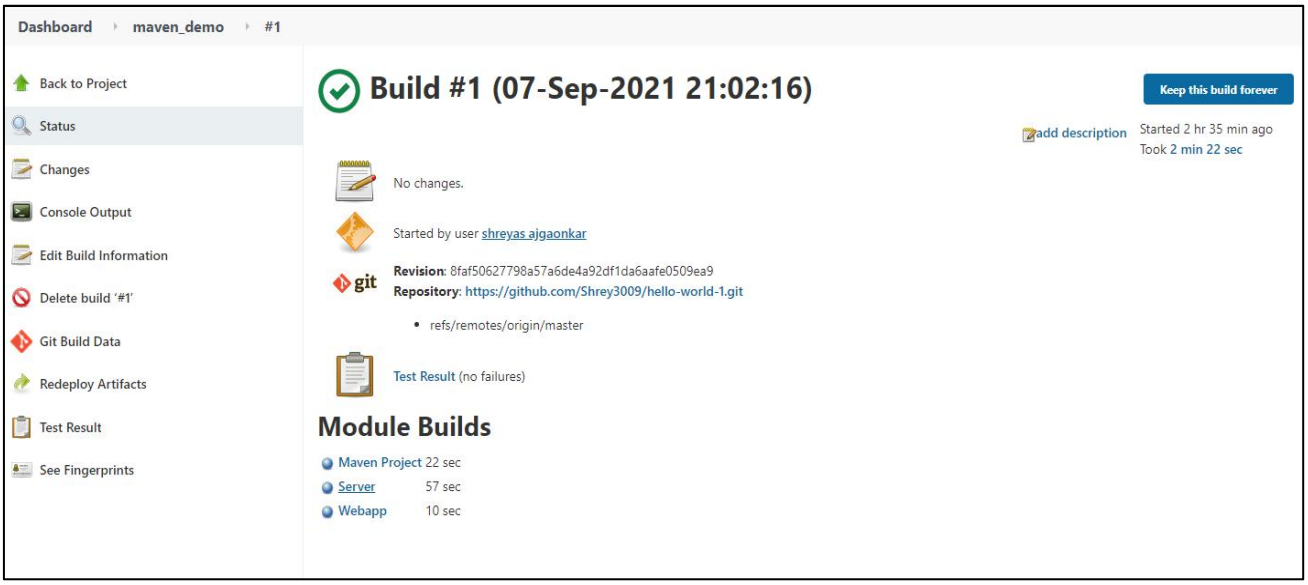
Installing Maven in Jenkins :



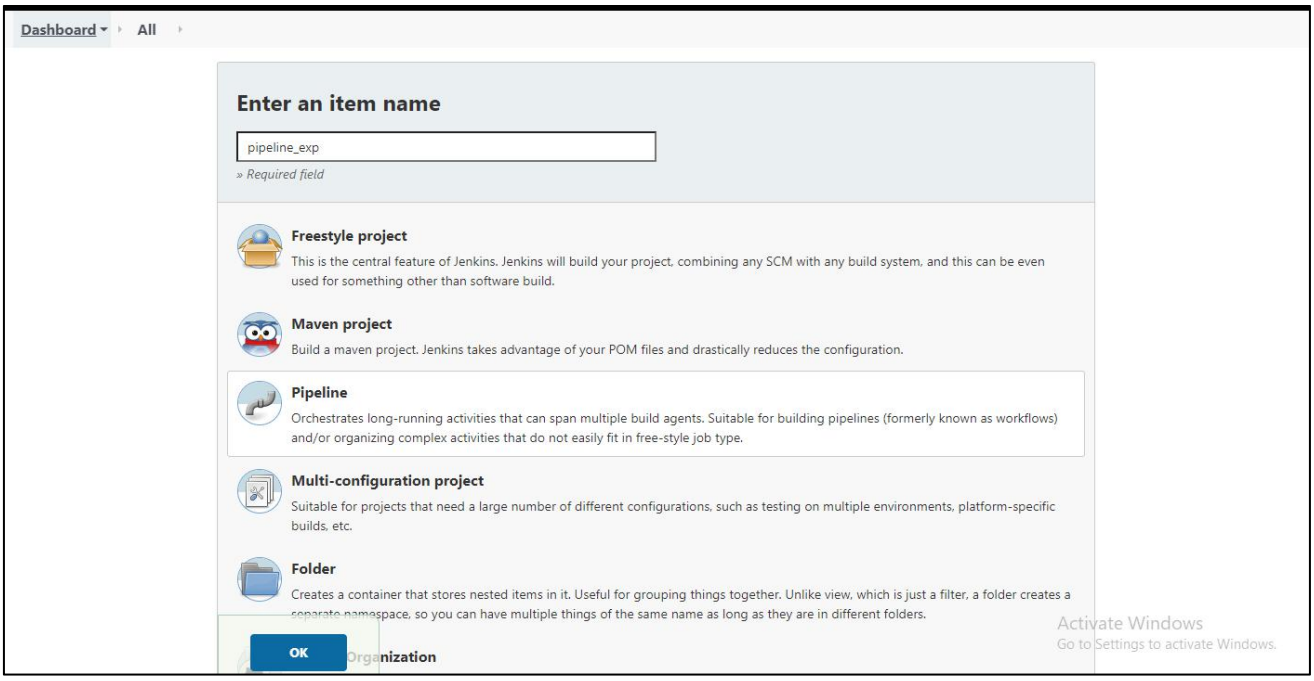



Saved
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Root POM
pom.xml
Goals and options
clean install
Advanced...
Post Steps
Run only if build succeeds
Run only if build succeeds or is unstable
Run regardless of build result
Should the post-build steps run only for successful builds, etc.
Add post-build step
Build Settings
E-mail Notification
Post-build Actions
Add post-build action
Save
Apply


Dashboard
maven_demo
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Git Build Data
Redeploy Artifacts
Test Result
See Fingerprints
Console Output
Started by user shreyas ajgaonkar
Running as SYSTEM
Building in workspace C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\maven_demo
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/Shrey3009/hello-world-1.git
> git.exe init C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\maven_demo # timeout=10
Fetching upstream changes from https://github.com/Shrey3009/hello-world-1.git
> git.exe --version # timeout=10
> git --version # 'git version 2.32.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/Shrey3009/hello-world-1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/Shrey3009/hello-world-1.git # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 8faf50627798a57a6de4a92df1da6aafe0509ea9 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 8faf50627798a57a6de4a92df1da6aafe0509ea9 # timeout=10
Commit message: "Update index.jsp"
First time build. Skipping changelog.
Parsing POMs
Discovered a new module com.example.maven-project:maven-project Maven Project
Discovered a new module com.example.maven-project:server Server
Discovered a new module com.example.maven-project:webapp Webapp
Modules changed, recalculating dependency graph
Established TCP socket on 65523



Pipeline of jobs in Jenkins :



General | Build Triggers | Advanced Project Options | Pipeline

Description

pipeline example

[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [ ] GitHub project
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterised
- [ ] Throttle builds

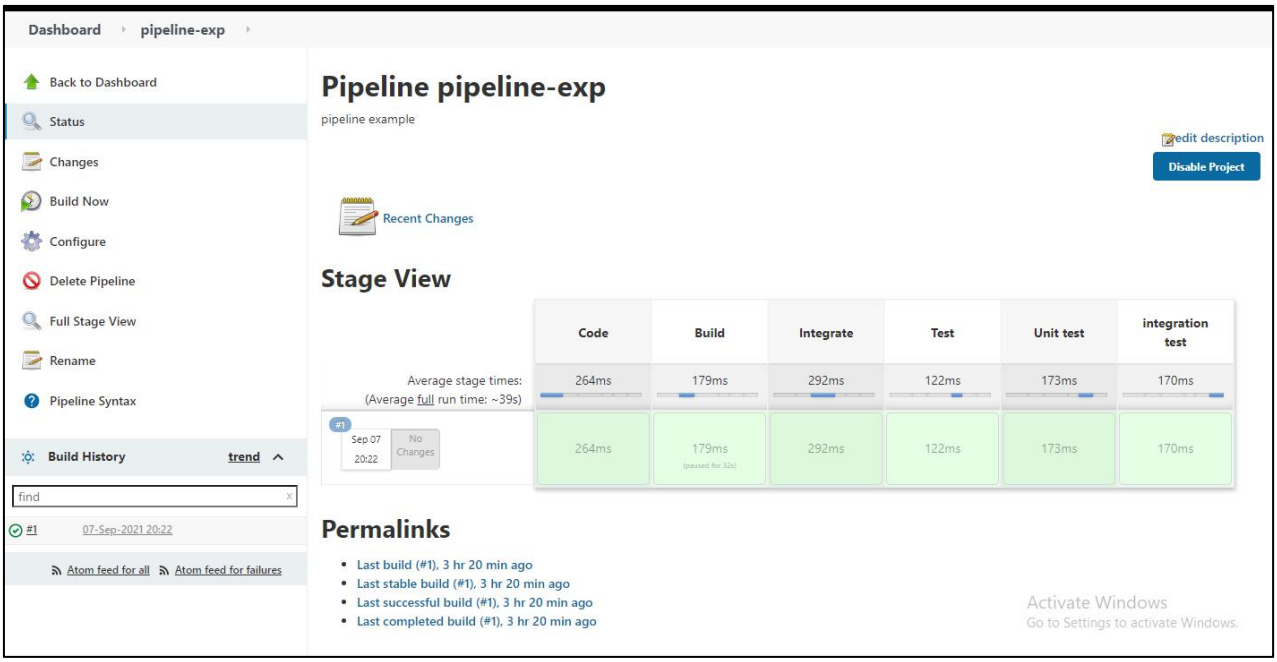
General | Build Triggers | Advanced Project Options | **Pipeline**

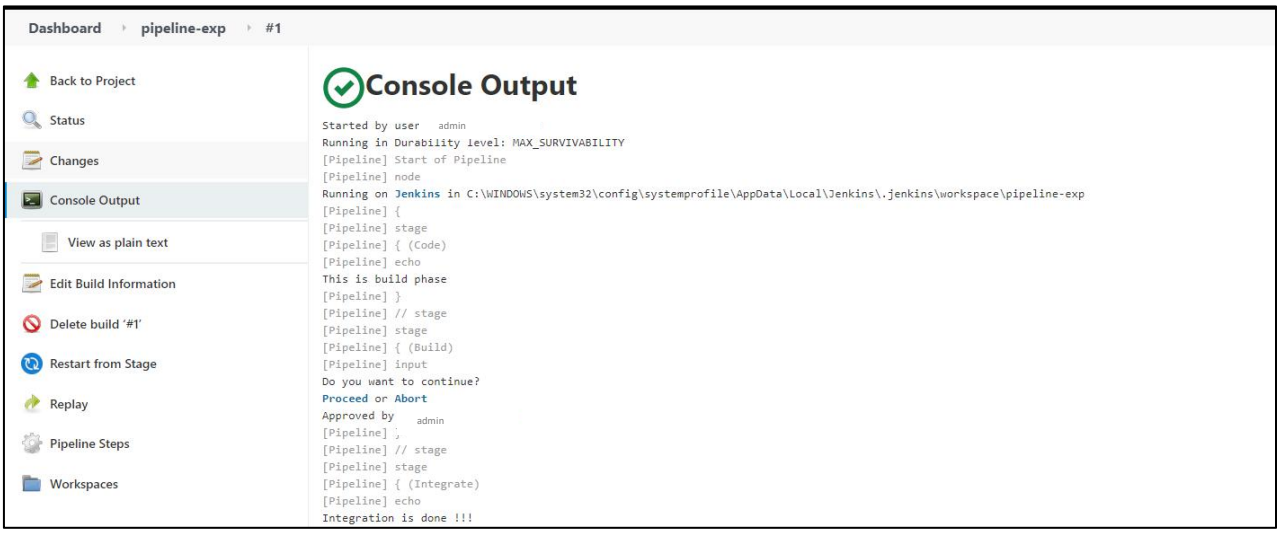
Definition

Pipeline script

Script

```
pipeline {
    agent any

    stages {
        stage('Code') {
            steps{
                    echo 'This is build phase'
            }

        }
        stage('Build'){
            steps{
                input('Do you want to continue?')
            }
        }

        stage('Integrate')  {
            when{
                not{
                    branch "master"
                }
            }
            steps{
                echo "Integration is done !!!"
            }
        }
        stage('Test'){
            parallel{
            stage('Unit test'){
                steps{
                        echo"test done"
                }
            }
           stage('integration test'){
               steps{
```


Dashboard
pipeline-exp
Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
Build History
trend
find
#1
07-Sep-2021 20:22
Atom feed for all
Atom feed for failures
Pipeline pipeline-exp
pipeline example
edit description
Disable Project
Recent Changes
Stage View
Code
Build
Integrate
Test
Unit test
integration test
Average stage times:
(Average full run time: ~39s)
264ms
179ms
292ms
122ms
173ms
170ms
#1
Sep 07
20:22
No Changes
Permalinks
Last build (#1), 3 hr 20 min ago
Last stable build (#1), 3 hr 20 min ago
Last successful build (#1), 3 hr 20 min ago
Last completed build (#1), 3 hr 20 min ago
Activate Windows
Go to Settings to activate Windows.

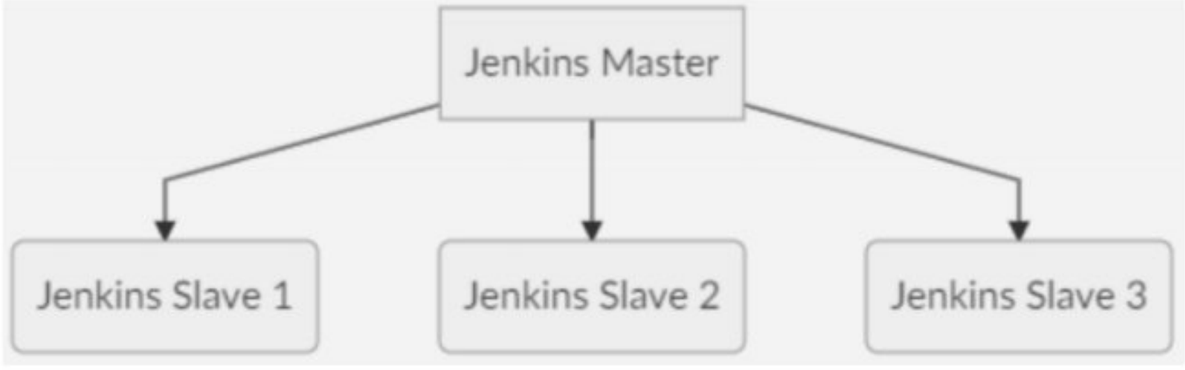

Dashboard
pipeline-exp
#1
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#1'
Restart from Stage
Replay
Pipeline Steps
Workspaces
Console Output
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\WINDOWS\system32\config\systemprofile\AppData\Local\Jenkins\.jenkins\workspace\pipeline-exp
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Code)
[Pipeline] echo
This is build phase
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] input
Do you want to continue?
Proceed or Abort
Approved by admin
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Integrate)
[Pipeline] echo
Integration is done !!!

CONCLUSION : Thus we successfully studied about pipeline of jobs using Maven / Gradle / Ant in Jenkins, created a pipeline script to Test and deploy an application over the tomcat server.

Experiment no -06

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, lD, 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.

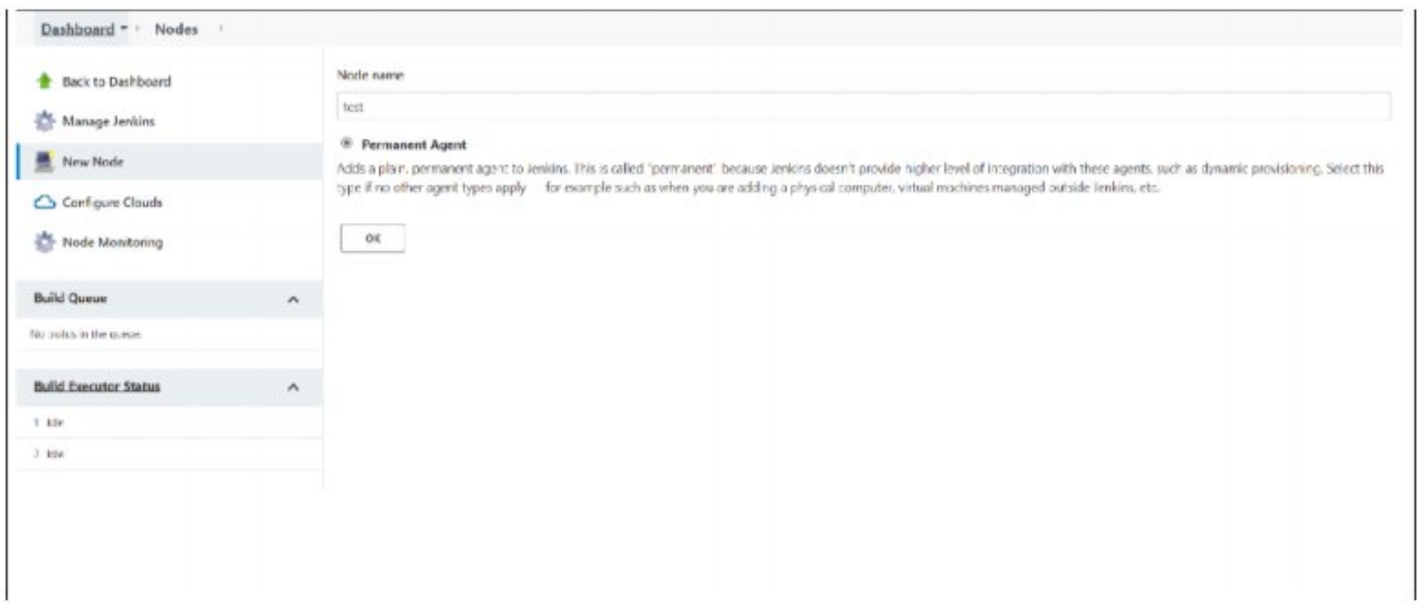
Creating a Pipeline and Running on The Slave Machine

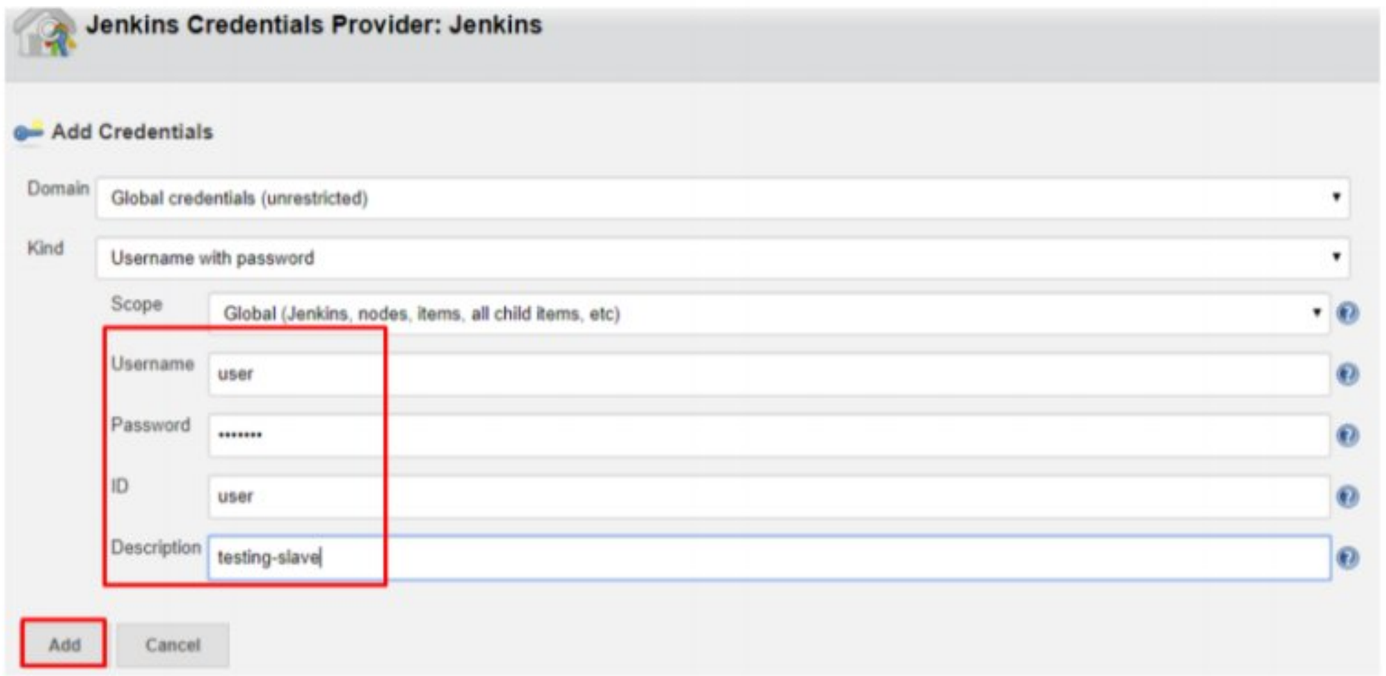
1) Click New Item in the top left corner on the dashboard.

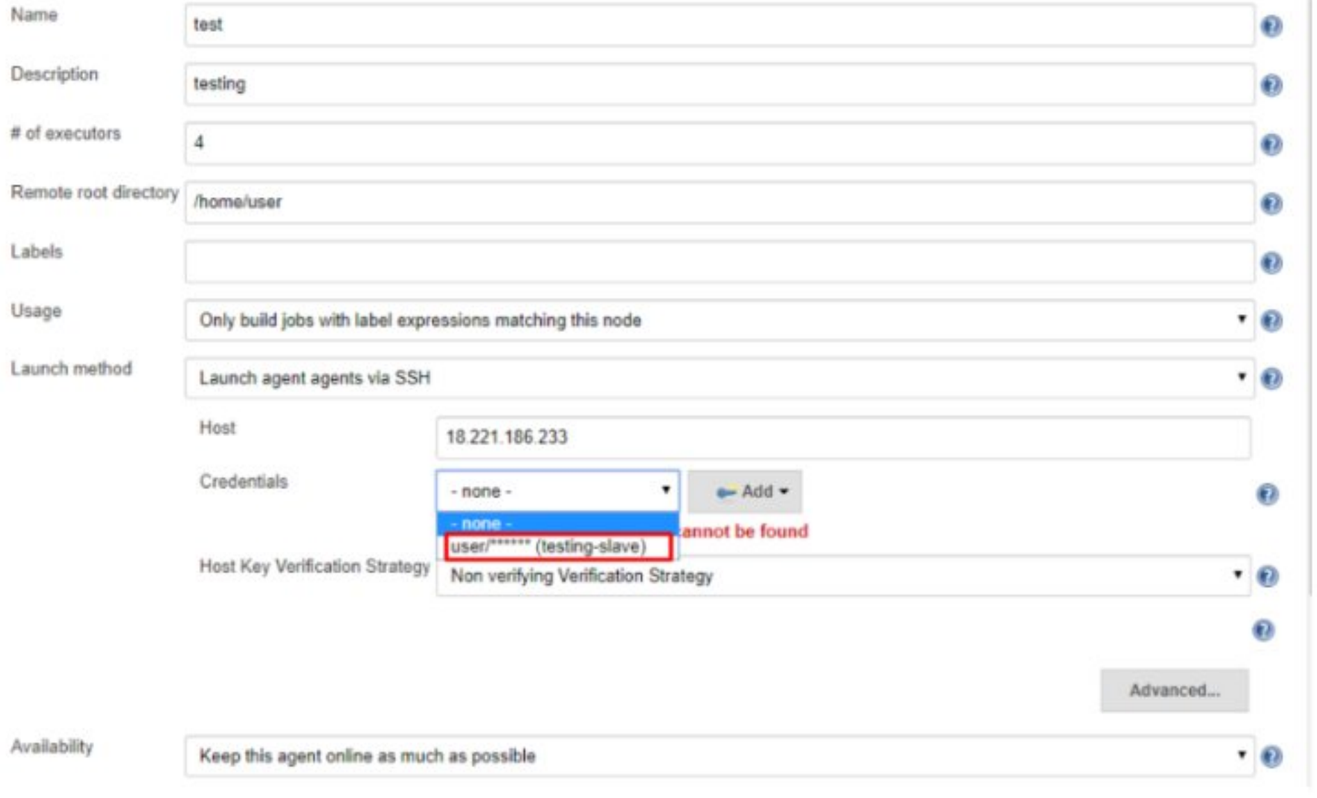
2) Enter the name of your project in the Enter an item name field, and select the Pipeline project, and click OK button.

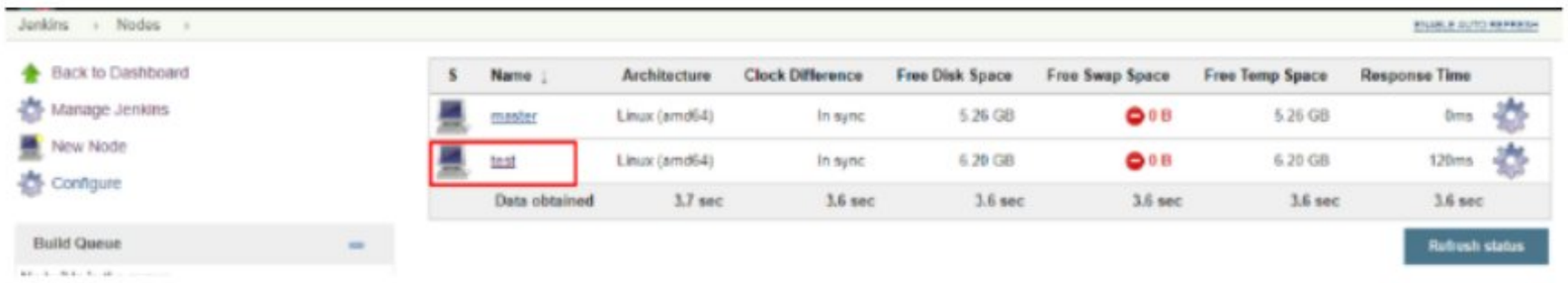
3) Enter Description (optional).

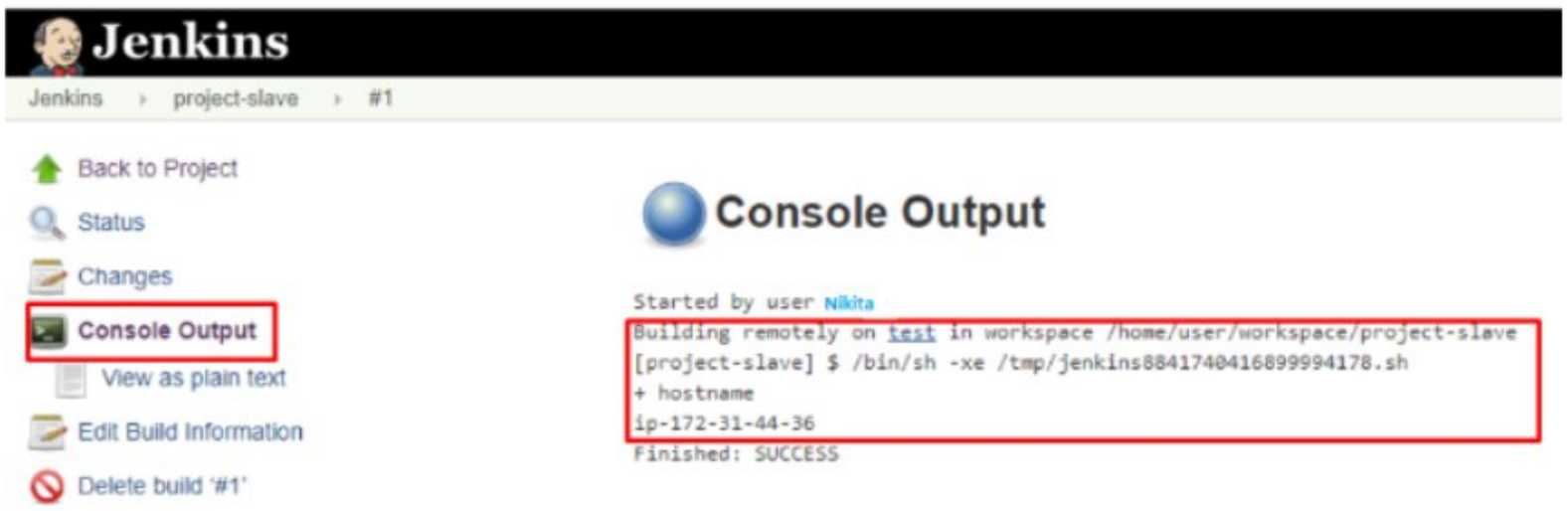
4) Go to the Pipeline section, make sure the Definition field has
the Pipeline script option selected.
5) Copy and paste the following declarative Pipeline script into a script field.
6) Click on Save, it will redirect to the Pipeline view page.
7) On the left pane, click the Build Now button to execute 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.

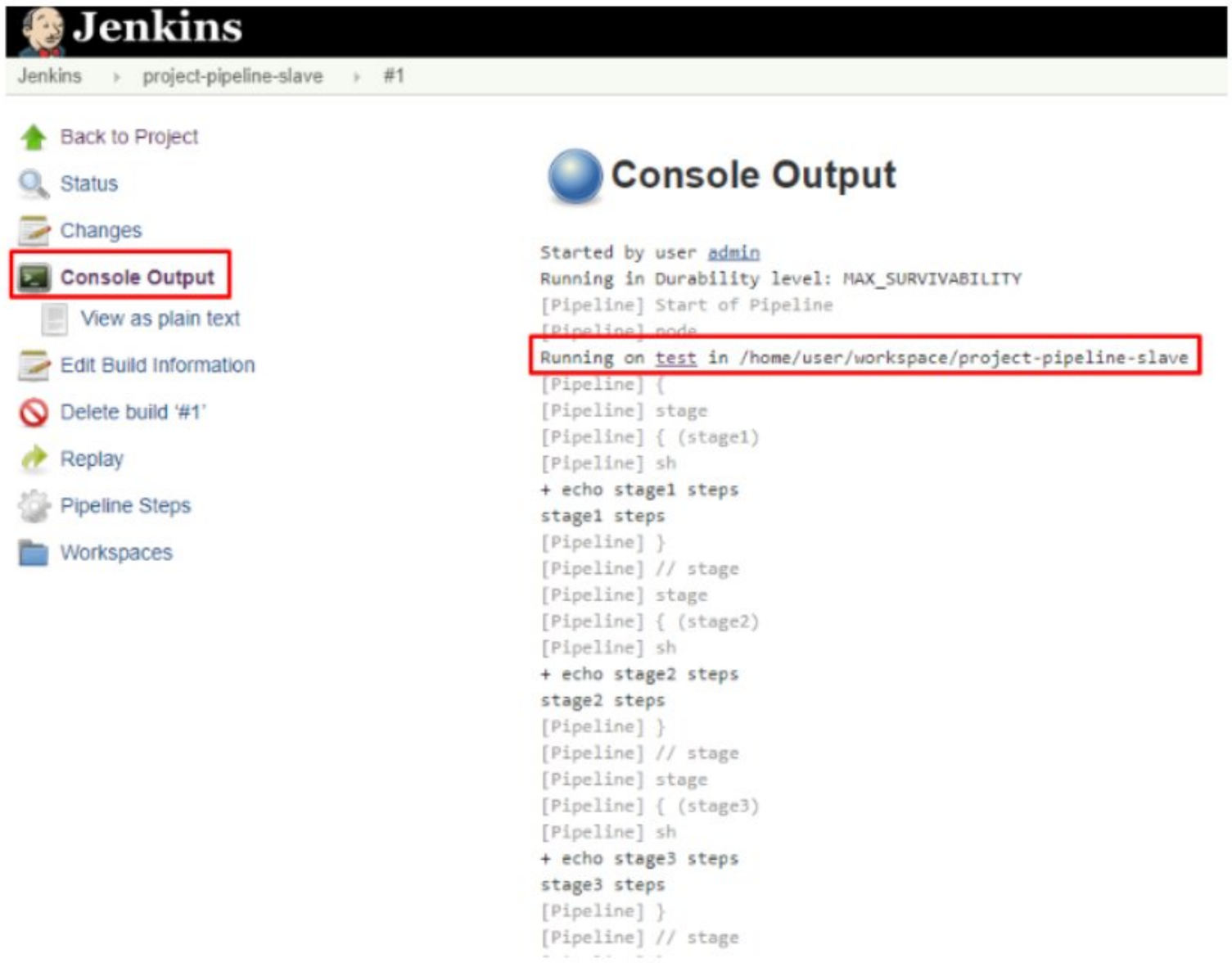












Conclusion: Successfully understand abd implemented Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

Experiment no. 7

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. Jenkin's chief usage is to monitor any job which can be SVN checkout, cron or any application states. It fires pre-configured actions when a particular step occurs in jobs.

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.

Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefits

Apache Maven provides support for managing the full lifecycle of a test project.

- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.
- Maven automatically downloads the necessary files from the repository while building the project.

Why Jenkins and Selenium?

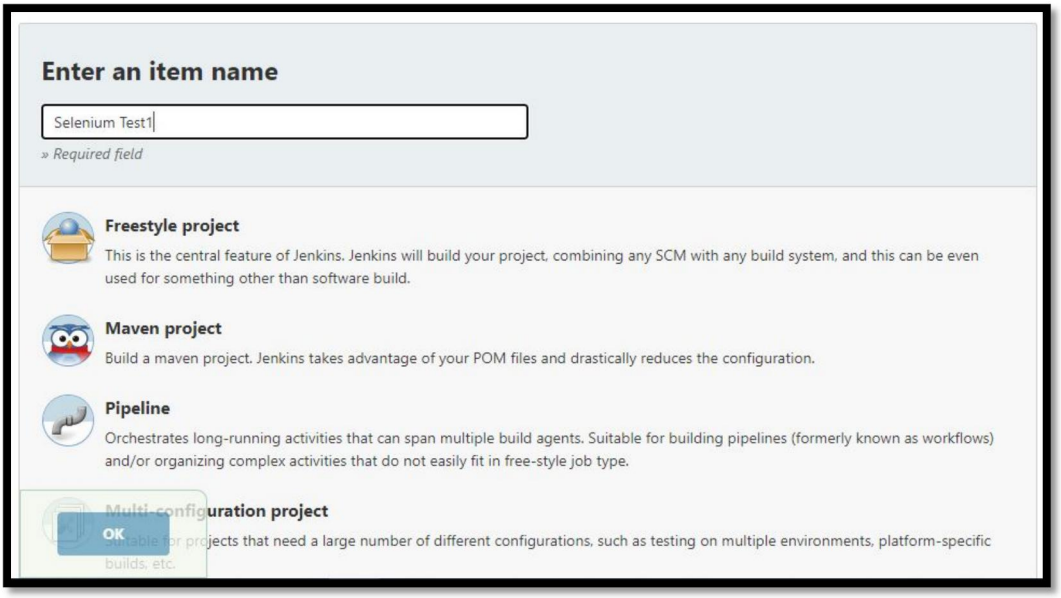
Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass.

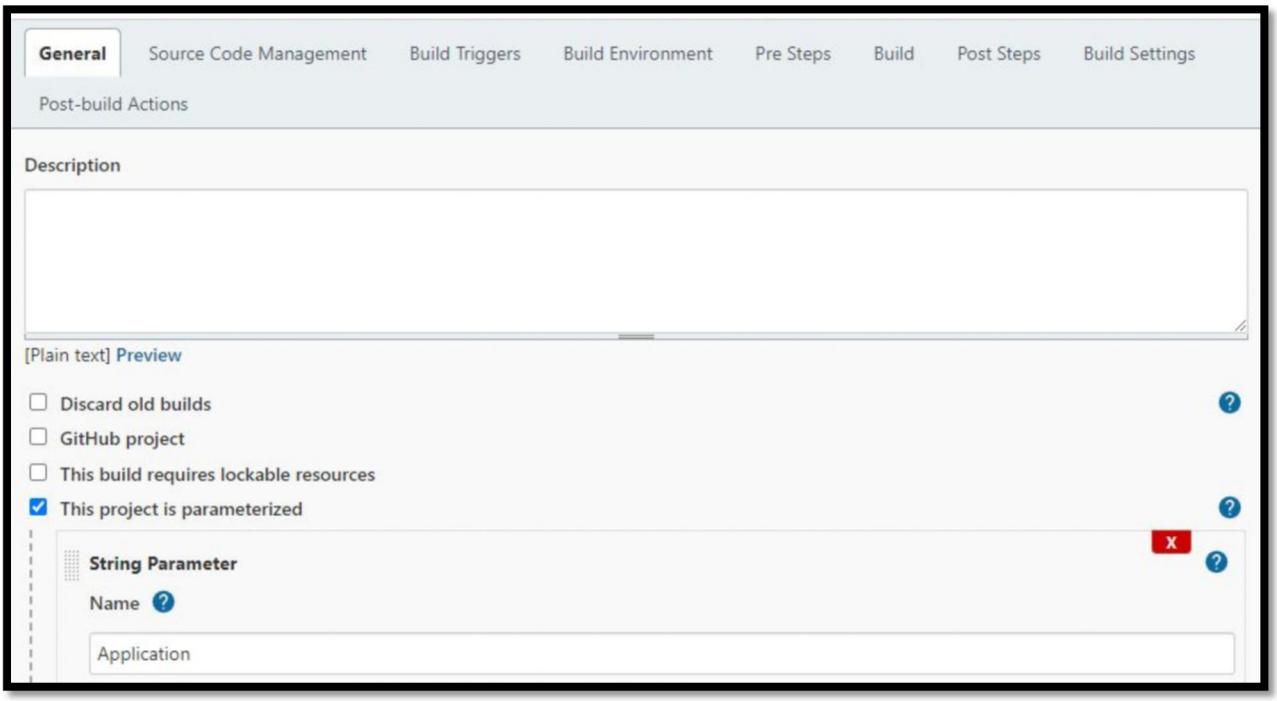
- Jenkins can schedule your tests to run at specific time.
- You can save the execution history and Test Reports.
- Jenkins supports Maven for building and Testing a project in continuous integration.

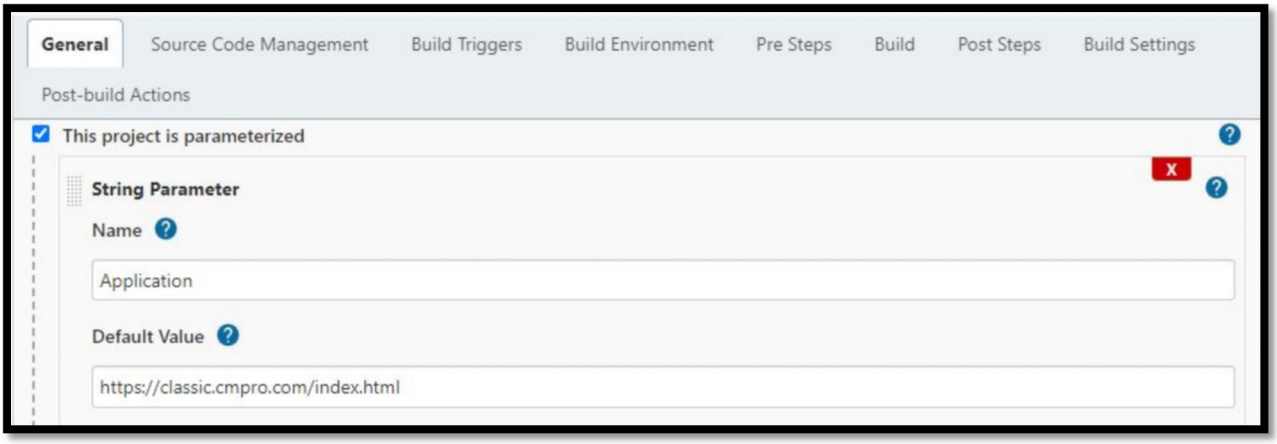
Advantages of Using Maven and Jenkins with selenium:

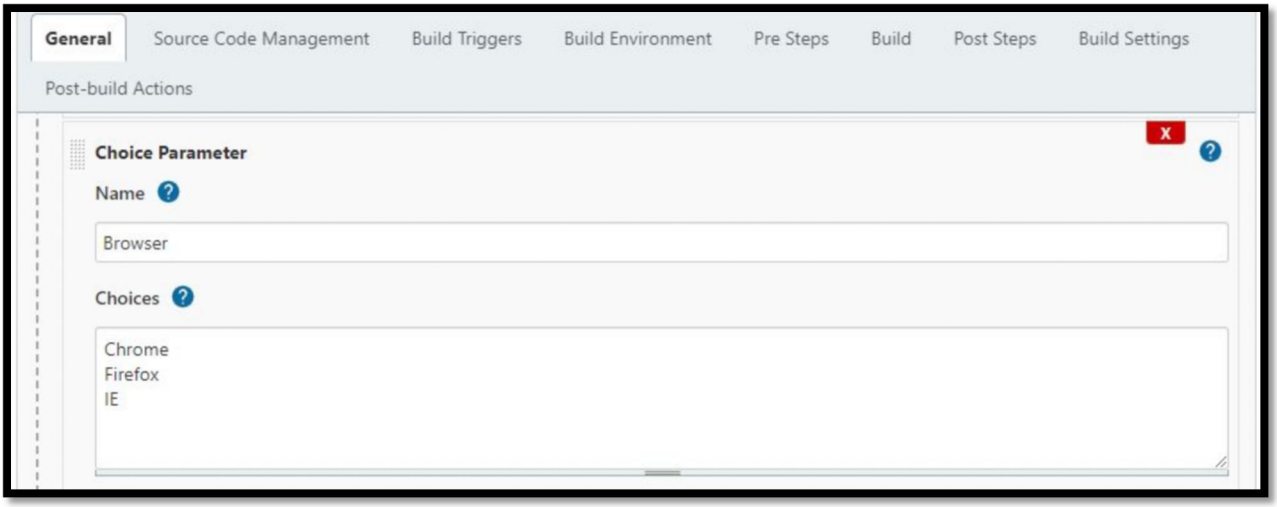
- Jenkins provides a way to do smoke testing for every time the code changes and deployed to a new environment. It will make sure that the code is running properly.
- You cans schedule your test cases with Jenkins so that if regression suite takes almost 6-7 hours to run then you can have nightly build run so that by the time you reach office it will be done.
- Jenkins server will act as a common server for client as well as technical people to logon to it and see all test reports and test execution history.
- Maven in turn reduces dependency on hard coding of jars.
- Maven can make the build process very easy.
- Also, at some time if you need to update the jars with a specific number, you don't need to go to Build path and add that particular jar. You can just change the number of version in pom.xml and it will be done.
- In a team where people are distributed across geographical locations it is easy to share artefact id and version id to clone the project rather than sharing on a common drive.

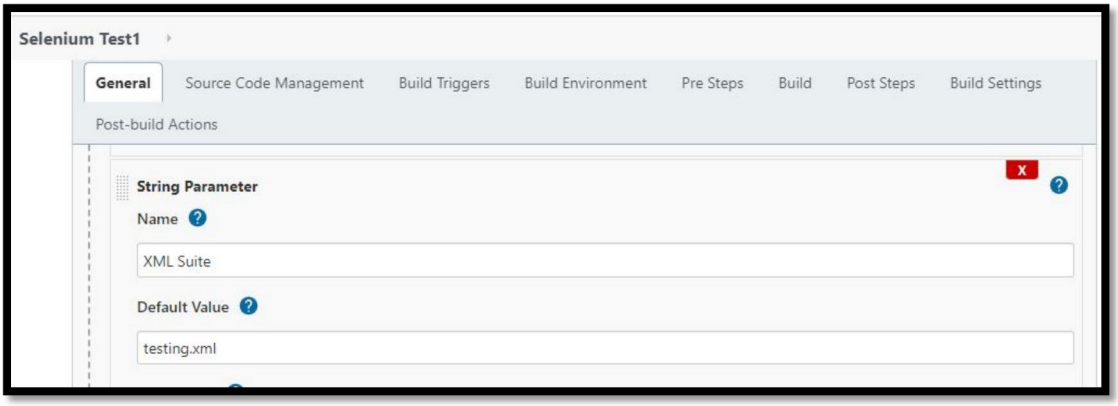
To setup and run selenium tests in Jenkins using maven:

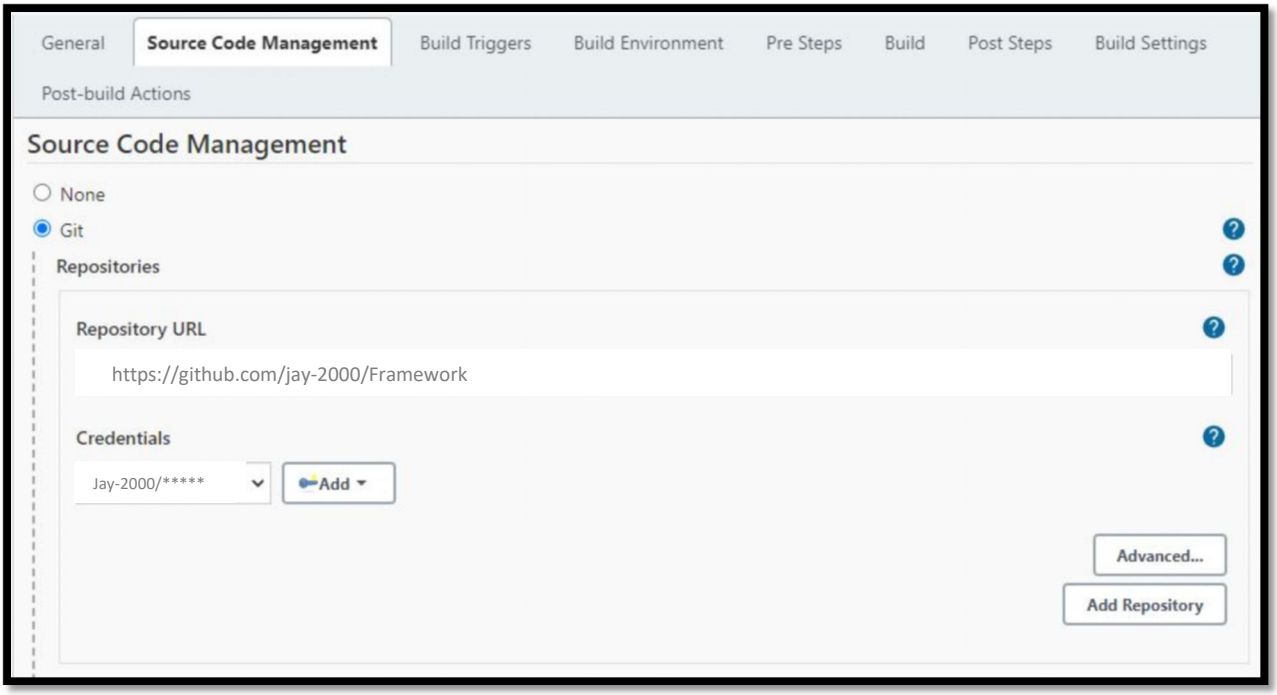

Enter an item name
Selenium Test1
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
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.
OK

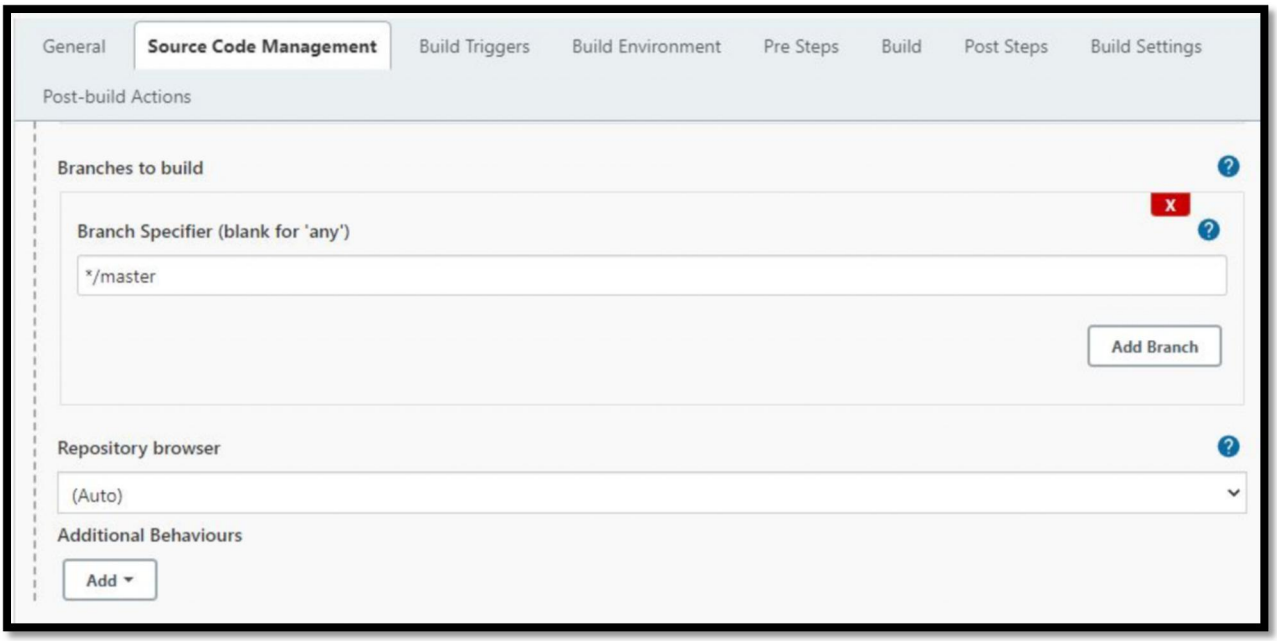

General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Description
[Plain text] Preview
Discard old builds
GitHub project
This build requires lockable resources
This project is parameterized
String Parameter
Name
Application

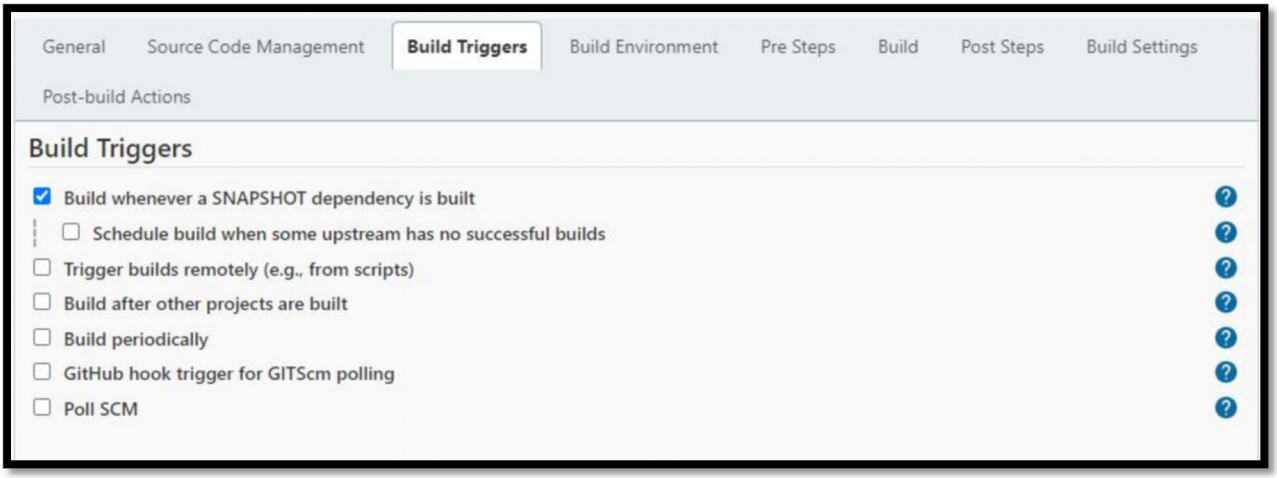

General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
This project is parameterized
String Parameter
Name
Application
Default Value
https://classic.cmpro.com/index.html

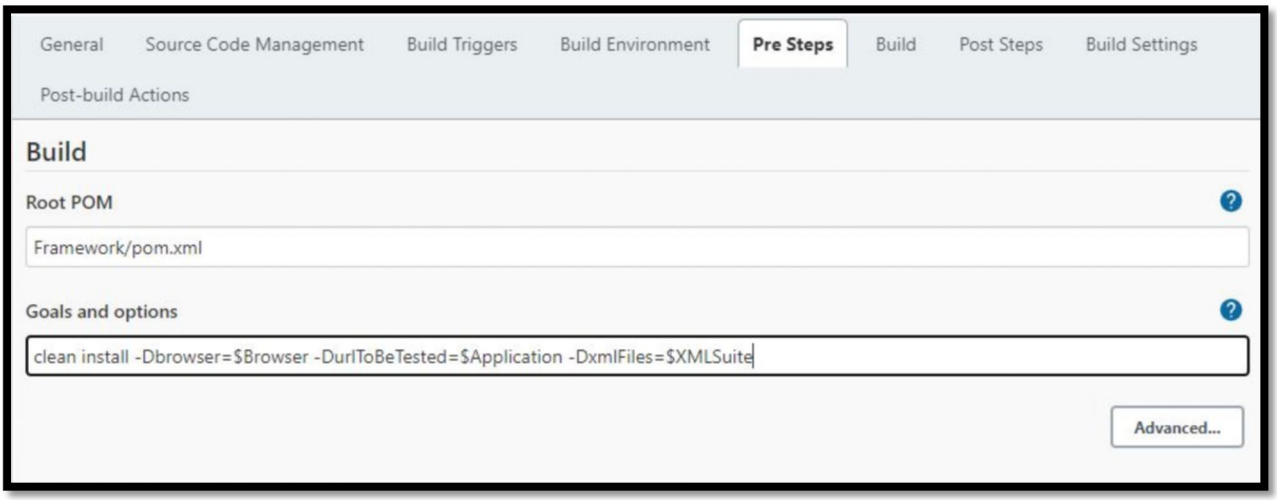

General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Choice Parameter
Name
Browser
Choices
Chrome
Firefox
IE

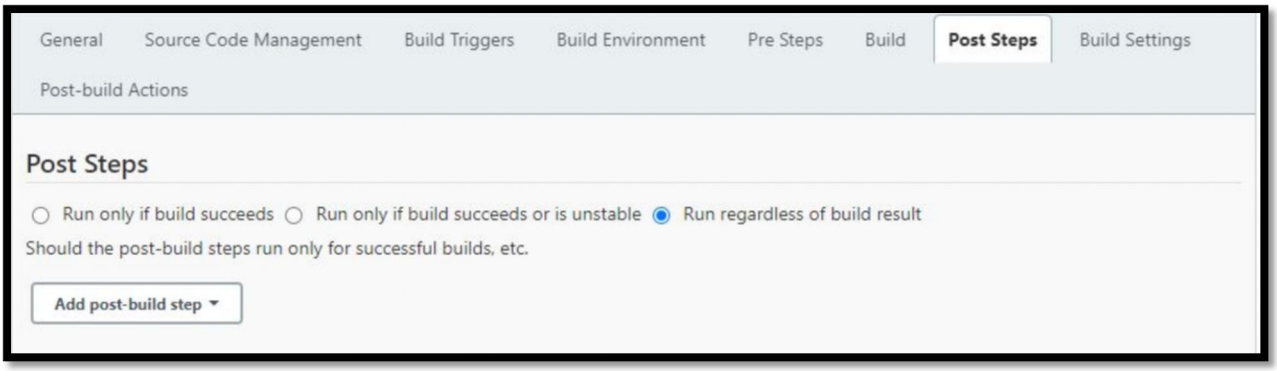

Selenium Test1
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
String Parameter
Name
XML Suite
Default Value
testing.xml


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/jay-2000/Framework
Credentials
Jay-2000/*****
Add
Advanced...
Add Repository


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build Triggers
Build whenever a SNAPSHOT dependency is built
Schedule build when some upstream has no successful builds
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build
Root POM
Framework/pom.xml
Goals and options
clean install -Dbrowser=$Browser -DurlToBeTested=$Application -DxmlFiles=$XMLSuite
Advanced...


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Post Steps
Run only if build succeeds
Run only if build succeeds or is unstable
Run regardless of build result
Should the post-build steps run only for successful builds, etc.
Add post-build step

```
Running as SYSTEM
Building on master in workspace C:\Users\jmd\.jenkins\workspace\Selenium Test1
The recommended git tool is: NONE
using credential 4c5e73f4-3de4-403b-8126-ad1721b3da23
 > git.exe rev-parse --resolve-git-dir C:\Users\jmd\.jenkins\workspace\Selenium Test1\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Aashiyana1/Framework # timeout=10
Fetching upstream changes from https://github.com/Aashiyana1/Framework
 > git.exe --version # timeout=10
 > git --version # 'git version 2.29.2.windows.3'
using GIT_ASKPASS to set credentials
 > git.exe fetch --tags --force --progress -- https://github.com/Aashiyana1/Framework +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 46c807957a75eeaa3754a9765115c799448e074a (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 46c807957a75eeaa3754a9765115c799448e074a # timeout=10
Commit message: "a"
First time build. Skipping changelog.
Parsing POMs
Discovered a new module Framework:Framework Framework
Modules changed, recalculating dependency graph
Established TCP socket on 49825
[Framework] $ java -cp C:\Users\jmd\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven35-agent-
```

```
project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

Conclusion:

Thus, we successfully learn and perform about Selenium Tests in Jenkins using Maven.

EXPERIMENT NO. : 08

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

THEORY :

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.

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.

DOCKER COMMANDS

1. docker version : Show the Docker version information

```
$ 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
```

2. docker search apache : Search images with name apache

```
$ docker search apache
NAME                              DESCRIPTION                                     STARS
      OFFICIAL            AUTOMATED
httpd                             The Apache HTTP Server Project                  3710
      [OK]
tomcat                            Apache Tomcat is an open source implementati…   3148
      [OK]
maven                             Apache Maven is a software project managemen…   1268
      [OK]
apache/airflow                    Apache Airflow                                  290
apache/nifi                       Unofficial convenience binaries and Docker i…   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                                 73
apachepulsar/pulsar               Apache Pulsar - Distributed pub/sub messagin…   49
apache/nifi-registry              Unofficial convenience binaries for Apache N…   30
apache/druid                      Apache Druid                                    28
```

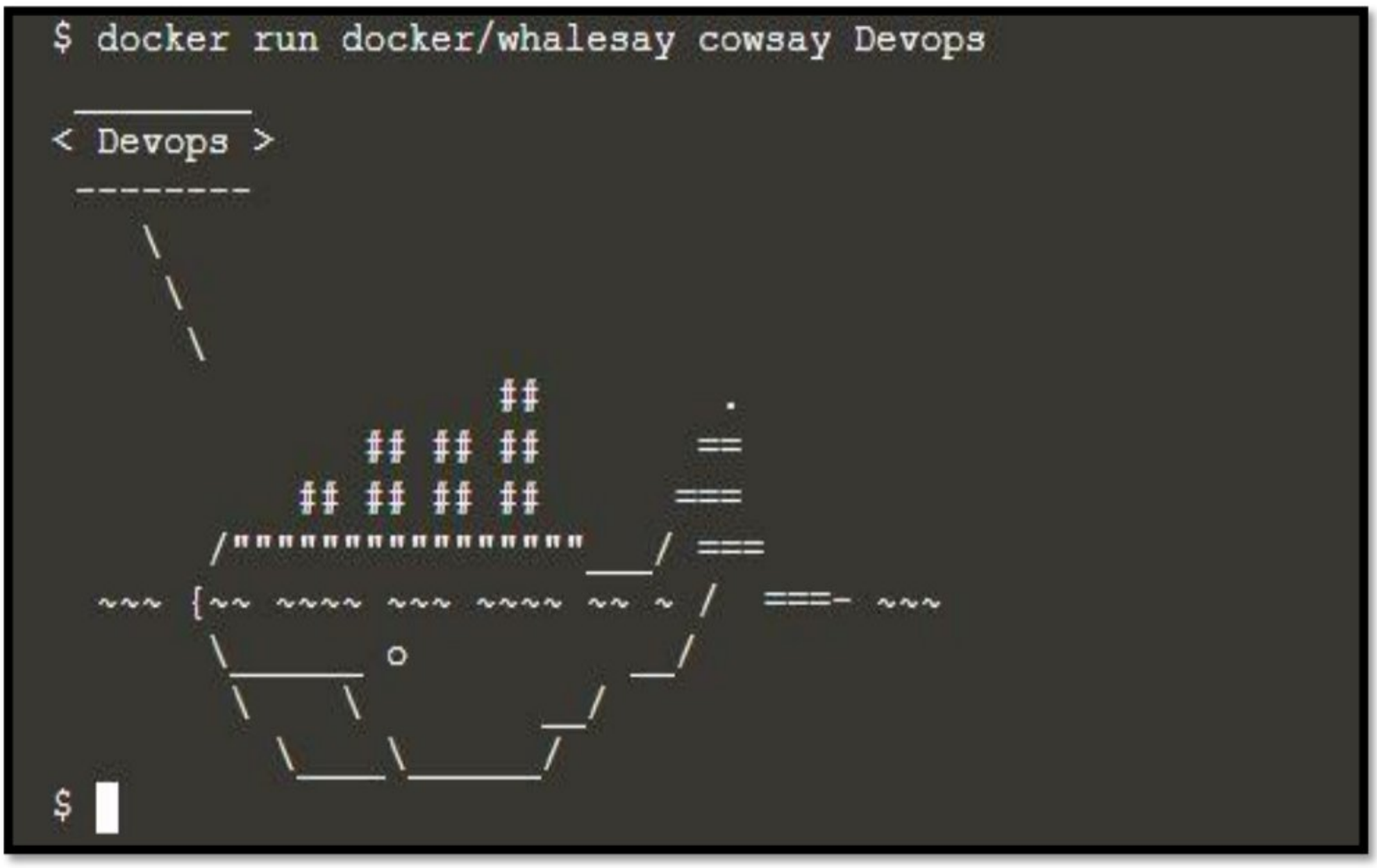
3. docker search redis : Search images with name redis

```
$ docker search redis
NAME                                DESCRIPTION                                     STARS
     OFFICIAL            AUTOMATED
redis                               Redis is an open source key-value store that…   10015
     [OK]
sameersbn/redis                                                                     83
                         [OK]
grokzen/redis-cluster               Redis cluster 3.0, 3.2, 4.0, 5.0, 6.0, 6.2      79

rediscommander/redis-commander      Alpine image for redis-commander - Redis man…   66
                         [OK]
redislabs/redisearch                Redis With the RedisSearch module pre-loaded…   39
redislabs/redisinsight              RedisInsight - The GUI for Redis                35
redislabs/redis                     Clustered in-memory database engine compatib…   31
oliver006/redis_exporter             Prometheus Exporter for Redis Metrics. Supp…   30
redislabs/rejson                    RedisJSON - Enhanced JSON data type processi…   27
arm32v7/redis                       Redis is an open source key-value store that…   25
redislabs/redisgraph                A graph database module for Redis               16
                         [OK]
redislabs/redismod                  An automated build of redismod - latest Redi…   15
                         [OK]
arm64v8/redis                       Redis is an open source key-value store that…   15
```

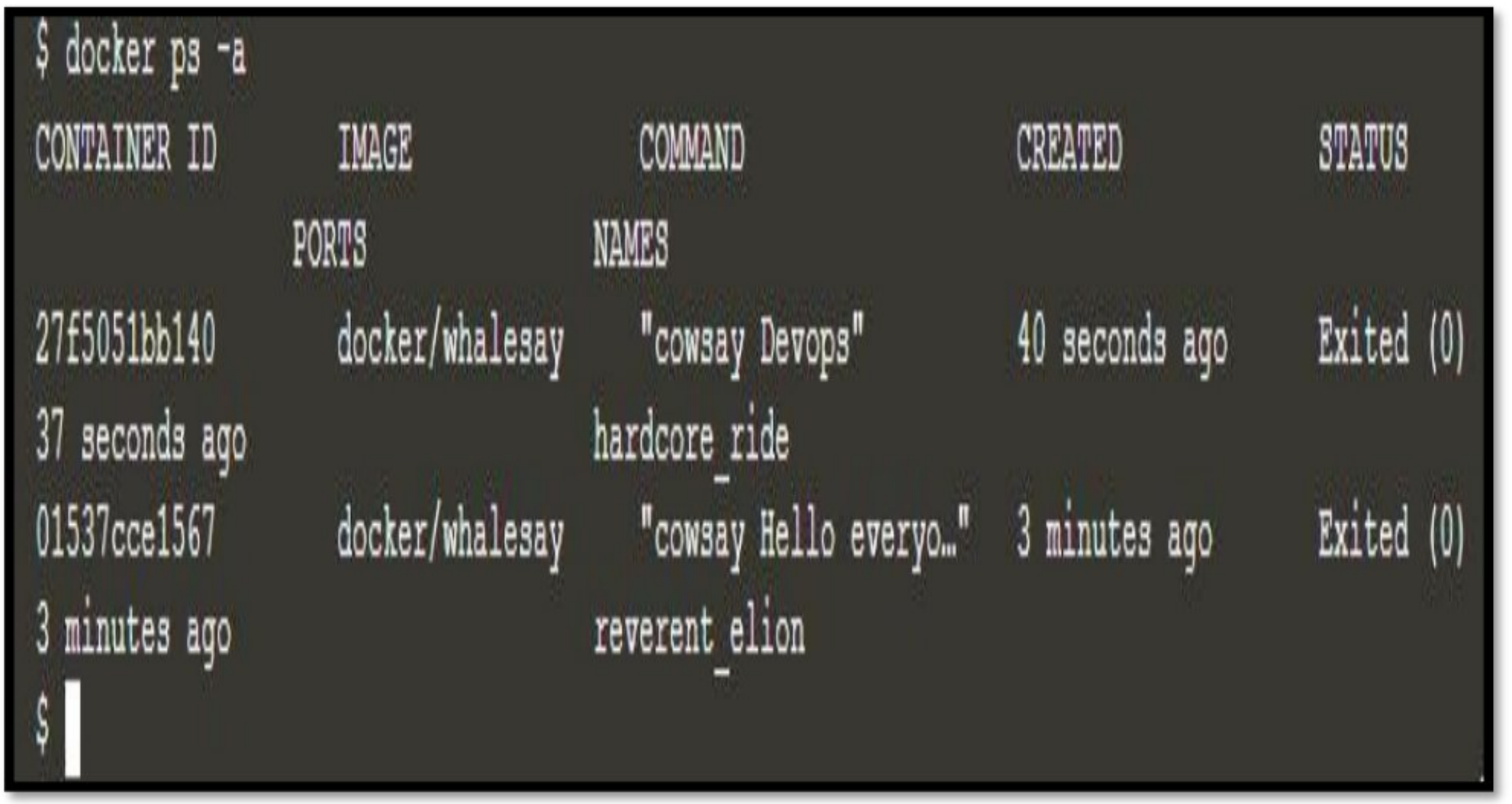
4. docker run docker/whalesay cowsay Hello everyone

```
$ docker run docker/whalesay cowsay Hello everyone
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 _____________
< Hello everyone >
 -------------
    \
     \
      \
                    ##         .
              ## ## ##        ==
           ## ## ## ##       ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
$
```

5. docker run docker/wehalesay cowsay DevOps



6. docker ps / docker ps -a : To list the containers



7. docker rm : To remove one or more containers

```
$ docker rm 27f5051bb140 01537cce1567
27f5051bb140
01537cce1567
$
```

8. docker rmi docker/whalesay : To remove container

```
$ docker rmi docker/whalesay
Untagged: docker/whalesay:latest
Untagged: docker/whalesay@sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f32
b
Deleted: sha256:6b362a9f73eb8c33b48c95f4fcce1b6637fc25646728cf7fb0679b2da273c3f4
Deleted: sha256:34dd66b3cb4467517d0c5c7dbe320b84539fbb58bc21702d2f749a5c932b3a38
Deleted: sha256:52f57e48814ed1bb08a651ef7f91f191db3680212a96b7f318bff0904fed2e65
Deleted: sha256:72915b616c0db6345e52a2c536de38e29208d945889eecef01d0fef0ed207ce8
Deleted: sha256:4ee0c1e90444c9b56880381aff6455f149c92c9a29c3774919632ded4f728d6b
Deleted: sha256:86ac1c0970bf5ea1bf482edb0ba83dbc88fefb1ac431d3020f134691d749d9a6
Deleted: sha256:5c4ac45a28f91f851b66af332a452cba25bd74a811f7e3884ed8723570ad6bc8
Deleted: sha256:088f9eb16f16713e449903f7edb4016084de8234d73a45b1882cf29b1f753a5a
Deleted: sha256:799115b9fdd1511e8af8a8a3c8b450d81aa842bbf3c9f88e9126d264b232c598
Deleted: sha256:3549adbf614379d5c33ef0c5c6486a0d3f577ba3341f573be91b4ba1d8c60ce4
Deleted: sha256:1154ba695078d29ea6c4e1adb55c463959cd77509adf09710e2315827d66271a
$
```

9. docker images : To remove images

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis               latest              4760dc956b2d        3 years ago         107MB
ubuntu              latest              f975c5035748        3 years ago         112MB
alpine              latest              3fd9065eaf02        3 years ago         4.14MB
$
```

10. docker pull centos:latest : To download centos

```
$ docker pull centos:latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              latest              5d0da3dc9764        3 weeks ago         231MB
redis               latest              4760dc956b2d        3 years ago         107MB
ubuntu              latest              f975c5035748        3 years ago         112MB
alpine              latest              3fd9065eaf02        3 years ago         4.14MB
$
```

11. docker run -it -name webserverdemo centos:latest

```
$ docker run -it --name webserverdemo centos:latest
[root@ab4ab0dfe8ff /]# yum install httpd -y
Failed to set locale, defaulting to C.UTF-8
CentOS Linux 8 - AppStream                      16 MB/s | 9.3 MB     00:00
CentOS Linux 8 - BaseOS                         23 MB/s | 7.5 MB     00:00
CentOS Linux 8 - Extras                         28 kB/s |  10 kB     00:00
Dependencies resolved.
================================================================================
 Package               Arch       Version                        Repository     Size
================================================================================
Installing:
 httpd                 x86_64     2.4.37-39.module_el8.4.0+778+c970deab     appstream     1.4 M
Installing dependencies:
 apr                   x86_64     1.6.3-11.el8                   appstream     125 k
 apr-util              x86_64     1.6.1-6.el8                    appstream     105 k
 brotli                x86_64     1.0.6-3.el8                    baseos        323 k
 centos-logos-httpd    noarch     85.8-1.el8                     baseos         75 k
 httpd-filesystem      noarch     2.4.37-39.module_el8.4.0+778+c970deab     appstream      38 k
 httpd-tools           x86_64     2.4.37-39.module_el8.4.0+778+c970deab     appstream     106 k
 mailcap               noarch     2.1.48-3.el8                   baseos         39 k
 mod_http2             x86_64     1.15.7-3.module_el8.4.0+778+c970deab      appstream     154 k
Installing weak dependencies:
 apr-util-bdb          x86_64     1.6.1-6.el8                    appstream      25 k
 apr-util-openssl      x86_64     1.6.1-6.el8                    appstream      27 k
Enabling module streams:
 httpd                            2.4

Transaction Summary
```

```
Transaction Summary
================================================================================
Install  11 Packages

Total download size: 2.4 M
Installed size: 7.1 M
Downloading Packages:
CentOS Linux 8 - BaseOS      204% [=============================================================] 2.1
(1/11): apr-util-bdb-1.6.1-6.el8.x86_64.rpm                 685 kB/s |  25 kB     00:00
(2/11): apr-util-1.6.1-6.el8.x86_64.rpm                     2.0 MB/s | 105 kB     00:00
(3/11): apr-util-openssl-1.6.1-6.el8.x86_64.rpm             1.7 MB/s |  27 kB     00:00
(4/11): apr-1.6.3-11.el8.x86_64.rpm                         2.1 MB/s | 125 kB     00:00
(5/11): httpd-filesystem-2.4.37-39.module_el8.4.0+778+c970deab.n 2.7 MB/s |  38 kB     00:00
(6/11): httpd-tools-2.4.37-39.module_el8.4.0+778+c970deab.x86_64 7.0 MB/s | 106 kB     00:00
(7/11): mod_http2-1.15.7-3.module_el8.4.0+778+c970deab.x86_64.rp 5.7 MB/s | 154 kB     00:00
(8/11): httpd-2.4.37-39.module_el8.4.0+778+c970deab.x86_64.rpm   22 MB/s | 1.4 MB     00:00
(9/11): brotli-1.0.6-3.el8.x86_64.rpm                       3.3 MB/s | 323 kB     00:00
(10/11): centos-logos-httpd-85.8-1.el8.noarch.rpm           973 kB/s |  75 kB     00:00
(11/11): mailcap-2.1.48-3.el8.noarch.rpm                    616 kB/s |  39 kB     00:00
--------------------------------------------------------------------------------
Total                                                       4.2 MB/s | 2.4 MB     00:00
warning: /var/cache/dnf/appstream-02e86d1c976ab532/packages/apr-1.6.3-11.el8.x86_64.rpm: Header V
3 RSA/SHA256 Signature, key ID 8483c65d: NOKEY
CentOS Linux 8 - AppStream                                  1.6 MB/s | 1.6 kB     00:00
Importing GPG key 0x8483C65D:
 Userid     : "CentOS (CentOS Official Signing Key) <security@centos.org>"
 Fingerprint: 99DB 70FA E1D7 CE22 7FB6 4882 05B5 55B3 8483 C65D
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-centosofficial
Key imported successfully
```

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : apr-1.6.3-11.el8.x86_64                                1/11
  Running scriptlet: apr-1.6.3-11.el8.x86_64                                1/11
  Installing       : apr-util-bdb-1.6.1-6.el8.x86_64                        2/11
  Installing       : apr-util-openssl-1.6.1-6.el8.x86_64                    3/11
  Installing       : apr-util-1.6.1-6.el8.x86_64                            4/11
  Running scriptlet: apr-util-1.6.1-6.el8.x86_64                            4/11
  Installing       : httpd-tools-2.4.37-39.module_el8.4.0+778+c970deab.x86_64 5/11
  Installing       : mailcap-2.1.48-3.el8.noarch                            6/11
  Installing       : centos-logos-httpd-85.8-1.el8.noarch                   7/11
  Installing       : brotli-1.0.6-3.el8.x86_64                              8/11
  Running scriptlet: httpd-filesystem-2.4.37-39.module_el8.4.0+778+c970deab.noarch 9/11
  Installing       : httpd-filesystem-2.4.37-39.module_el8.4.0+778+c970deab.noarch 9/11
  Installing       : mod_http2-1.15.7-3.module_el8.4.0+778+c970deab.x86_64  10/11
  Installing       : httpd-2.4.37-39.module_el8.4.0+778+c970deab.x86_64     11/11
  Running scriptlet: httpd-2.4.37-39.module_el8.4.0+778+c970deab.x86_64     11/11
  Verifying        : apr-1.6.3-11.el8.x86_64                                1/11
  Verifying        : apr-util-1.6.1-6.el8.x86_64                            2/11
  Verifying        : apr-util-bdb-1.6.1-6.el8.x86_64                        3/11
  Verifying        : apr-util-openssl-1.6.1-6.el8.x86_64                    4/11
  Verifying        : httpd-2.4.37-39.module_el8.4.0+778+c970deab.x86_64     5/11
  Verifying        : httpd-filesystem-2.4.37-39.module_el8.4.0+778+c970deab.noarch 6/11
  Verifying        : httpd-tools-2.4.37-39.module_el8.4.0+778+c970deab.x86_64 7/11
  Verifying        : mod_http2-1.15.7-3.module_el8.4.0+778+c970deab.x86_64  8/11
  Verifying        : brotli-1.0.6-3.el8.x86_64                              9/11
  Verifying        : centos-logos-httpd-85.8-1.el8.noarch                  10/11
  Verifying        : mailcap-2.1.48-3.el8.noarch                           11/11

Installed:
  apr-1.6.3-11.el8.x86_64
  apr-util-1.6.1-6.el8.x86_64
  apr-util-bdb-1.6.1-6.el8.x86_64
  apr-util-openssl-1.6.1-6.el8.x86_64
  brotli-1.0.6-3.el8.x86_64
  centos-logos-httpd-85.8-1.el8.noarch
  httpd-2.4.37-39.module_el8.4.0+778+c970deab.x86_64
  httpd-filesystem-2.4.37-39.module_el8.4.0+778+c970deab.noarch
  httpd-tools-2.4.37-39.module_el8.4.0+778+c970deab.x86_64
  mailcap-2.1.48-3.el8.noarch
  mod_http2-1.15.7-3.module_el8.4.0+778+c970deab.x86_64

Complete!
[root@ab4ab0dfe8ff /]#
```

CONCLUSION : Hence we can conclude that we have learned and implemented Docker Architecture and Container Life Cycle and executed docker commands to manage images and interact with containers.

Experiment no. 9

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

Theory:

DOCKER

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software,
libraries and configuration files; they can communicate with each other through well-defined channels.

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.

FROM: This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.
Ex. FROM ubuntu LABEL We can add labels to an image to organize images of our project. We need to use LABEL instruction to set label for the image. Ex. LABEL vendorl = "JavaTpoint".

RUN: RUN instruction allows you to install your application and packages requited for it. It executes any commands on top of the current image and creates a new layer by committing the results. Often you will find multiple RUN instructions in a Dockerfile.

CMD: CMD instruction allows you to set a default command, which will be executed only when you run container without specifying a command. If Docker container runs with a command, the default command will be ignored. If Dockerfile has more than one CMD instruction, all but last CMD instructions are ignored.

Rules: The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon. If source is a directory, the entire contents of the directory are copied including filesystem metadata.

WORKDIR: The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.

Output:

app.js pg.dockerfile

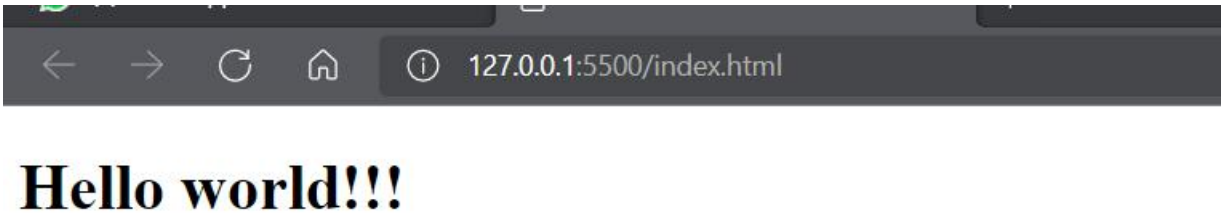
pg.dockerfile > ...

```
1  FROM node:alpine
2  COPY . /app
3  WORKDIR /app
4  CMD node app.js
```

```
$ docker run docker/whalesay cowsay Hello everyone
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 _______________
< Hello everyone >
 ---------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \     \        __/
          \____\______/
$
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
    PORTS               NAMES
$ ps -a
    PID TTY          TIME CMD
    155 pts/0    00:00:00 ps
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
                    PORTS               NAMES
01537cce1567        docker/whalesay     "cowsay Hello everyo…"   About a minute ago   Exited (0)
 About a minute ago                     reverent_elion
$
```

Html:



Conclusion:
Hence we have successfully executed and implemented to build an image for a sample web application using Dockerfile.

Experiment no 10

Aim: To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet.

Theory:

Puppet :

Puppet is an open source software configuration management and deployment tool. It's most commonly used on Linux and Windows to pull the strings on multiple application servers at once. But you can also use Puppet on several platforms, including IBM mainframes, Cisco switches, and Mac OS servers.

Like other DevOps programs, Puppet does more than automate system administration. It changes the human workflow, and enables developers and system administrators to work together. Programmers can write, test, and launch applications without waiting on Ops staff to deliver the resources needed.

For example, Microsoft and Puppet recently partnered with the RISCO Group, an Israeli security project company, to create a Puppet and Azure Resource Manager-powered self-service web portal. The result, says Ido Vapner, RISCO Group's head of DevSecOps and technology, is a development workflow that enables the company to "provision an entire environment with a single click" instead of it taking "a week to build a new environment."

Puppet Versions :

Puppet comes in two versions:-

1. Open Source Puppet-

It is a basic version of Puppet configuration management tool, which is also known as Open Source Puppet. It is available directly from Puppet's website and is licensed under the Apache 2.0 system.

2. Puppet Enterprise-

Commercial version that offers features like compliance reporting, orchestration, role based access control, GUI,API, and command-line tools for effective management of nodes.

Functions:

1) Puppet allows you to define distinct configurations for every host.
2) The tool allows you to continuously monitor servers to confirm whether the required configuration exists or not and it is not altered. If the config is changed, Puppet tool will revert to the pre-defined configuration on the host.
3) It also provides control over all the configured system, so a centralized change gets automatically effected.
4) It is also used as deployment tools as it automatically deploys software to the system. It implements the infrastructure as a code because policies and configuration are written as code.

How Puppet Works?

Puppet is based on a Pull deployment model, where the agent nodes check in regularly after every 1800 seconds with the master node to see if anything needs to be updated in the agent. If anything needs to be updated the agent pulls the necessary puppet codes from the master and performs required actions.

Puppet Blocks:

Puppet provides the flexibility to integrate Reports with third-party tools using Puppet APIs.

Four types of Puppet building blocks are

- Resources
- Classes
- Manifest
- Modules

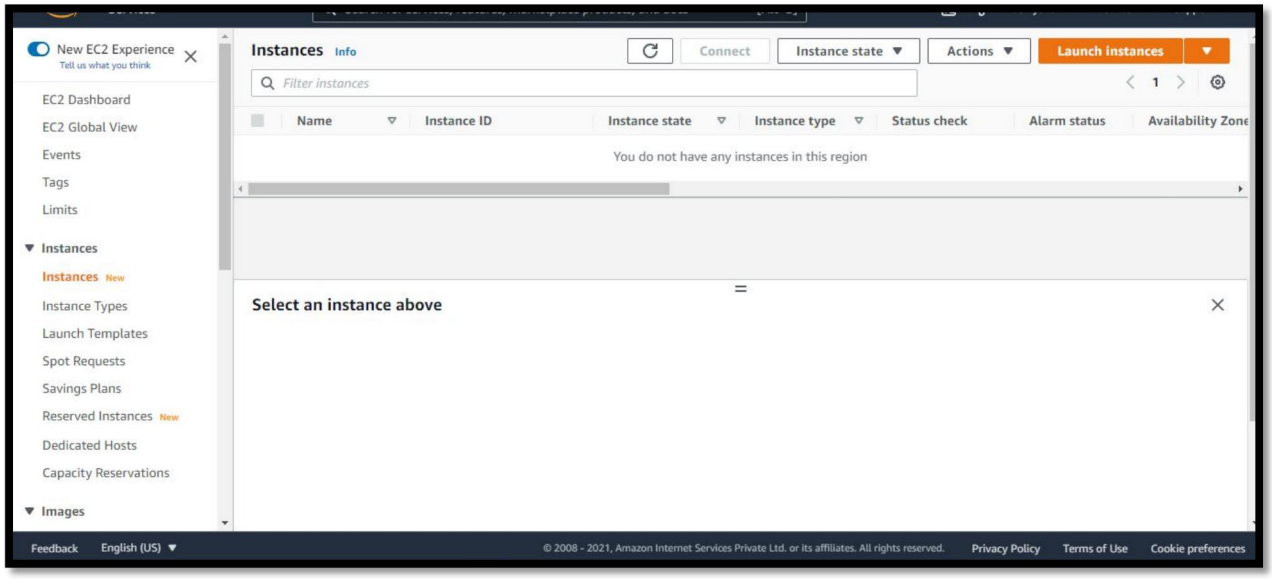
Configuration Management?

Configuration management is the process of maintaining software and computer systems (for example servers, storage, networks) in a known, desired and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes.

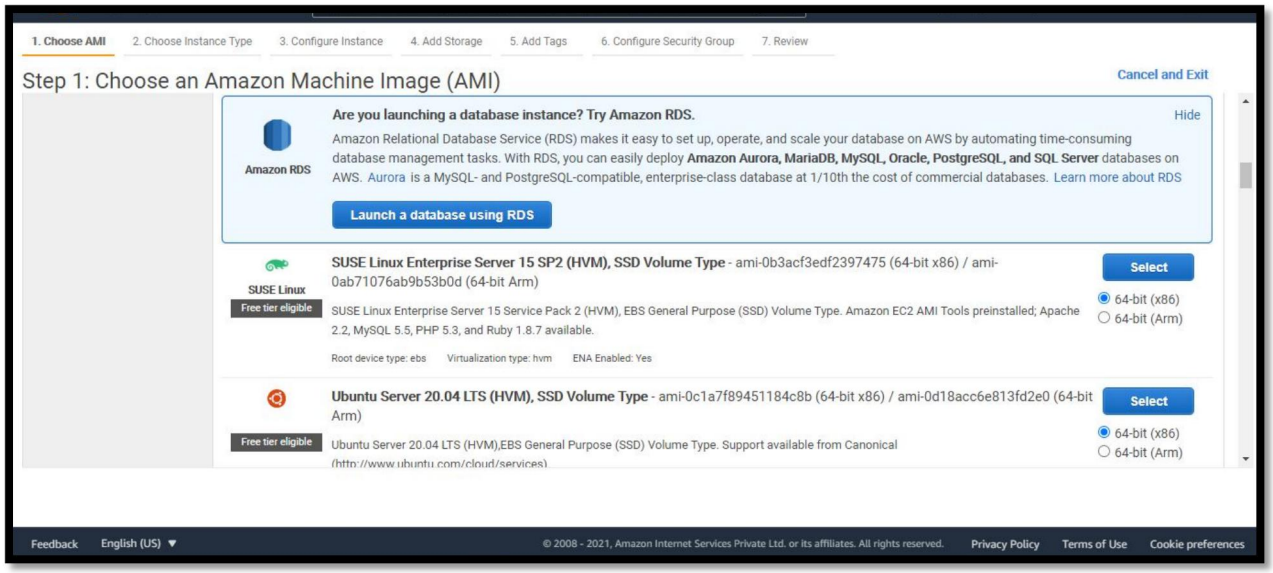
System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate this task, by writing scripts.

However, it is a difficult job when they are working on a massive infrastructure.

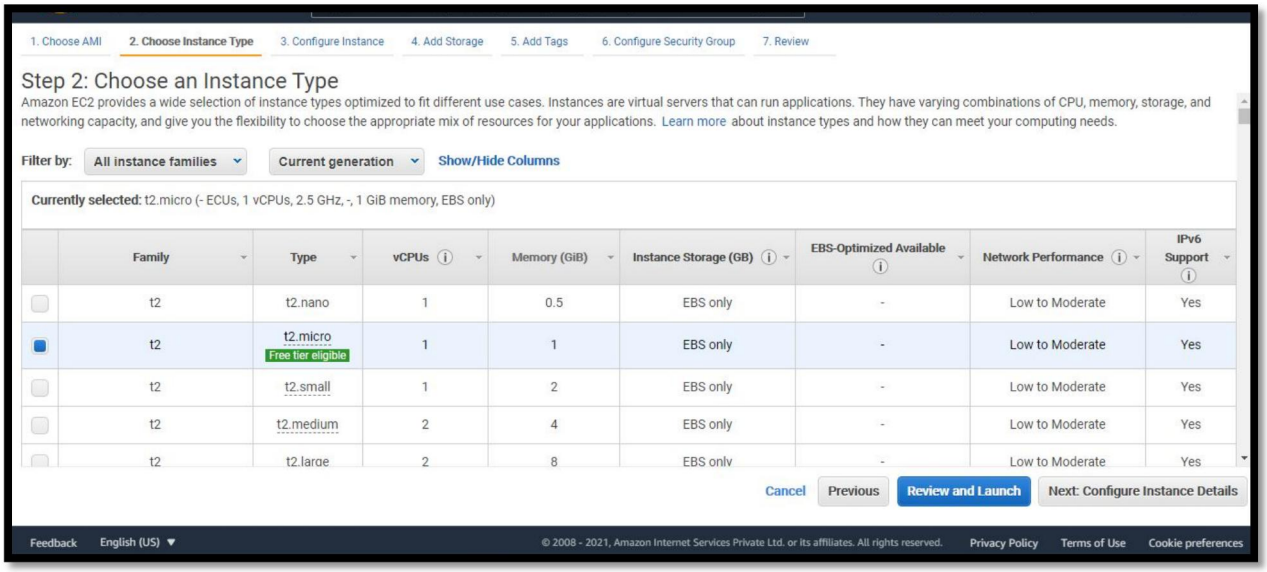
Step 1:


New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
Tags
Limits
Instances
Instances New
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances New
Dedicated Hosts
Capacity Reservations
Images
Instances Info
Connect
Instance state
Actions
Launch instances
Filter instances
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
You do not have any instances in this region
Select an instance above
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

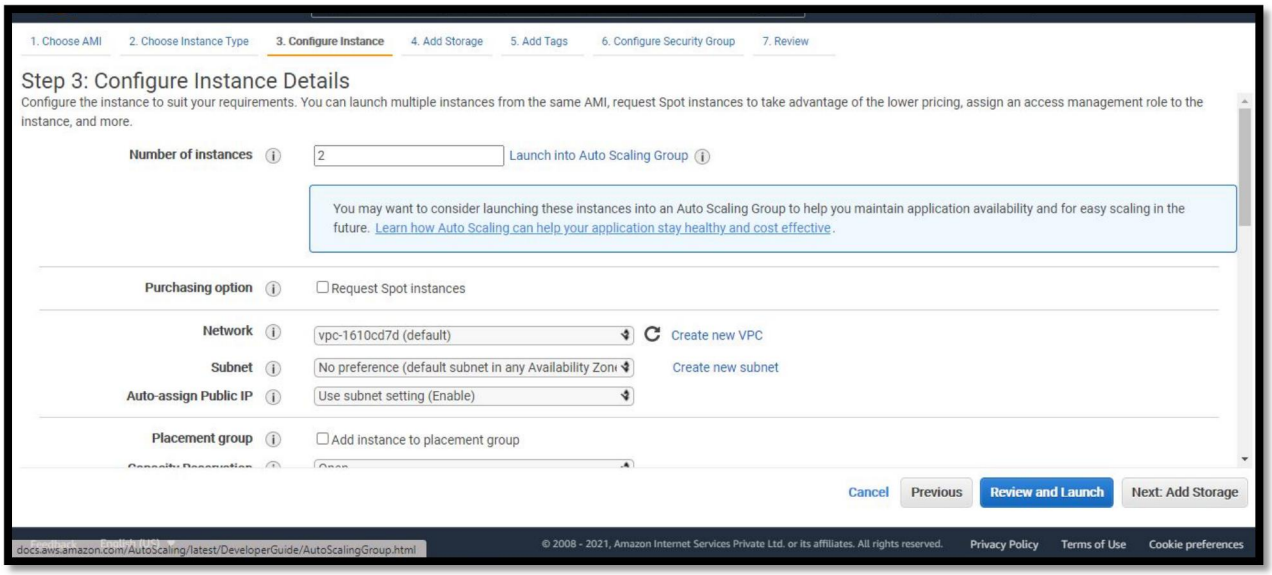
Step 2:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
Are you launching a database instance? Try Amazon RDS.
Hide
Amazon RDS
Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale your database on AWS by automating time-consuming database management tasks. With RDS, you can easily deploy Amazon Aurora, MariaDB, MySQL, Oracle, PostgreSQL, and SQL Server databases on AWS. Aurora is a MySQL- and PostgreSQL-compatible, enterprise-class database at 1/10th the cost of commercial databases. Learn more about RDS
Launch a database using RDS
SUSE Linux
Free tier eligible
SUSE Linux Enterprise Server 15 SP2 (HVM), SSD Volume Type - ami-0b3acf3edf2397475 (64-bit x86) / ami-0ab71076ab9b53b0d (64-bit Arm)
Select
64-bit (x86)
64-bit (Arm)
SUSE Linux Enterprise Server 15 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available.
Root device type: ebs
Virtualization type: hvm
ENA Enabled: Yes
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0c1a7f89451184c8b (64-bit x86) / ami-0d18acc6e813fd2e0 (64-bit Arm)
Select
Free tier eligible
Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical
64-bit (x86)
64-bit (Arm)
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

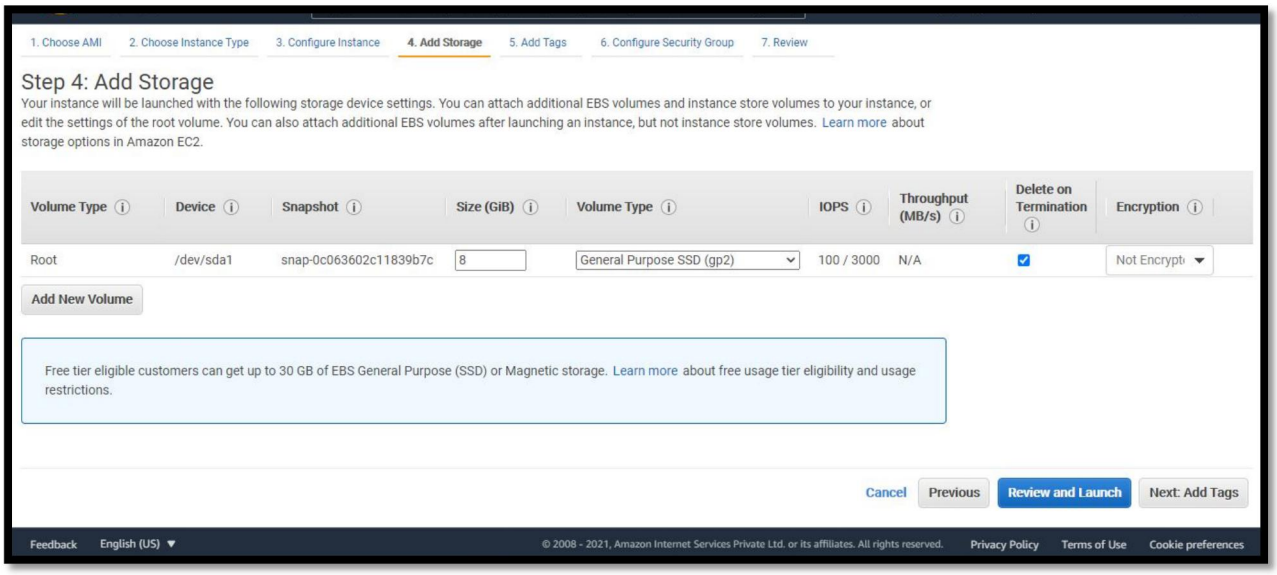
Step 3:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
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
t2 t2.medium 2 4 EBS only - Low to Moderate Yes
t2 t2.large 2 8 EBS only - Low to Moderate Yes
Cancel
Previous
Review and Launch
Next: Configure Instance Details
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

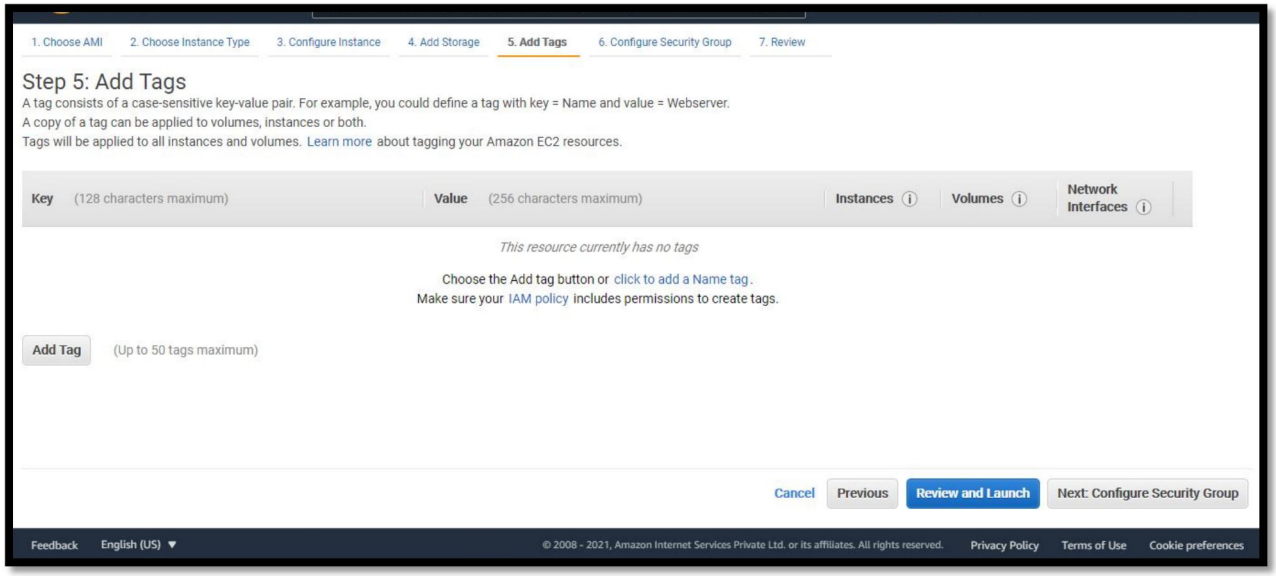
Step 4:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
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-1610cd7d (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zone
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
Placement group
Add instance to placement group
Cancel
Previous
Review and Launch
Next: Add Storage
docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroup.html
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

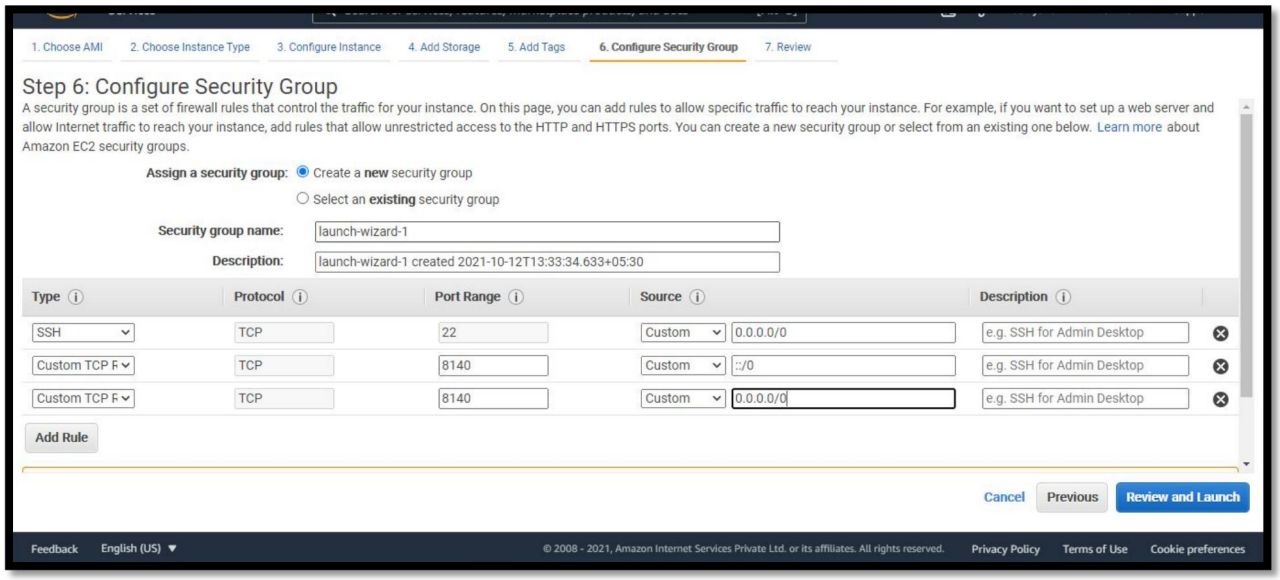
Step 5:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
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-0c063602c11839b7c
8
General Purpose SSD (gp2)
100 / 3000
N/A
Not Encrypt
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
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

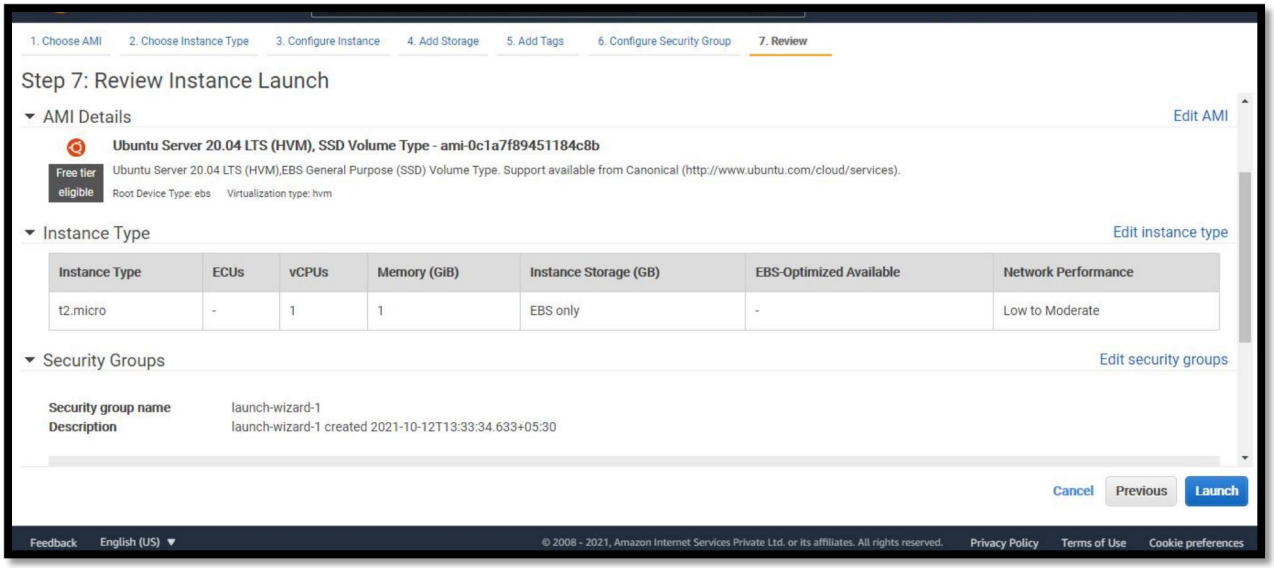
Step 6:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 5: Add Tags
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.
Key (128 characters maximum)
Value (256 characters maximum)
Instances
Volumes
Network Interfaces
This resource currently has no tags
Choose the Add tag button or click to add a Name tag.
Make sure your IAM policy includes permissions to create tags.
Add Tag
(Up to 50 tags maximum)
Cancel
Previous
Review and Launch
Next: Configure Security Group
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

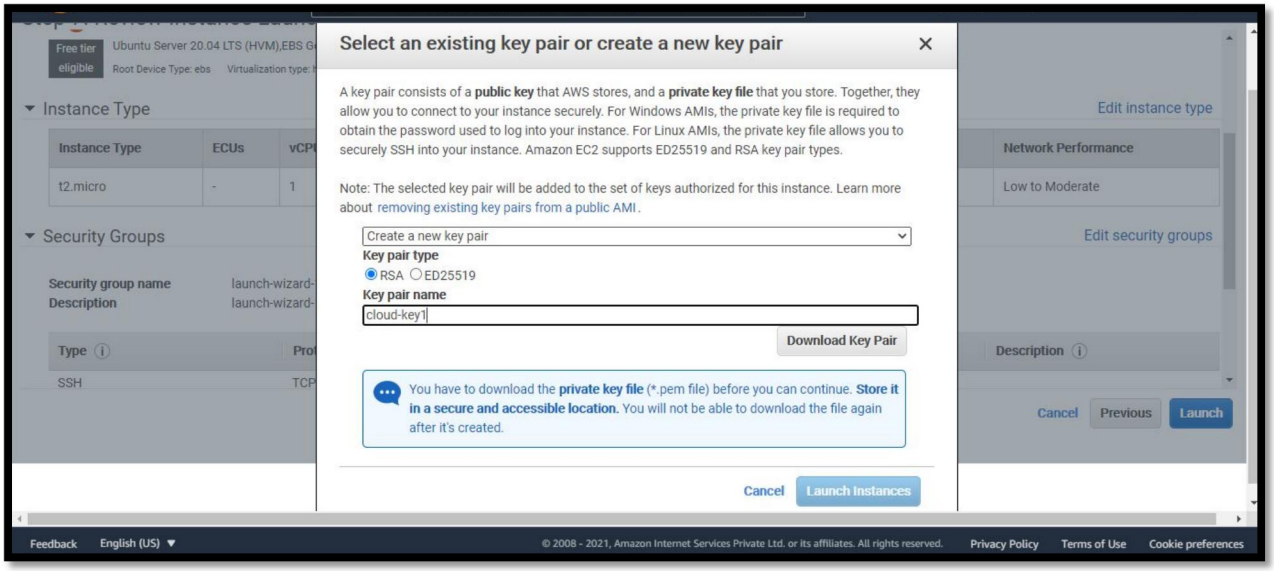
Step 7:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 6: Configure Security Group
A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.
Assign a security group: Create a new security group
Select an existing security group
Security group name: launch-wizard-1
Description: launch-wizard-1 created 2021-10-12T13:33:34.633+05:30
Type
Protocol
Port Range
Source
Description
SSH
TCP
22
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
Custom TCP R
TCP
8140
Custom
::/0
e.g. SSH for Admin Desktop
Custom TCP R
TCP
8140
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
Add Rule
Cancel
Previous
Review and Launch
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

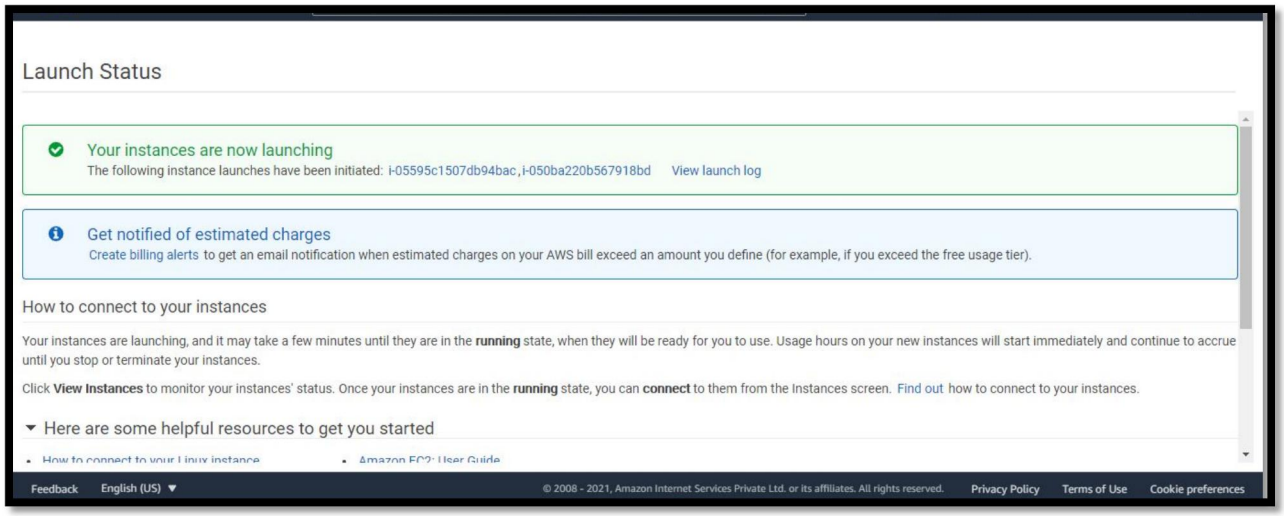
Step 8:


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
AMI Details
Edit AMI
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0c1a7f89451184c8b
Free tier eligible
Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root Device Type: ebs
Virtualization type: hvm
Instance Type
Edit instance type
Instance Type
ECUs
vCPUs
Memory (GiB)
Instance Storage (GB)
EBS-Optimized Available
Network Performance
t2.micro
-
1
1
EBS only
-
Low to Moderate
Security Groups
Edit security groups
Security group name
launch-wizard-1
Description
launch-wizard-1 created 2021-10-12T13:33:34.633+05:30
Cancel
Previous
Launch
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

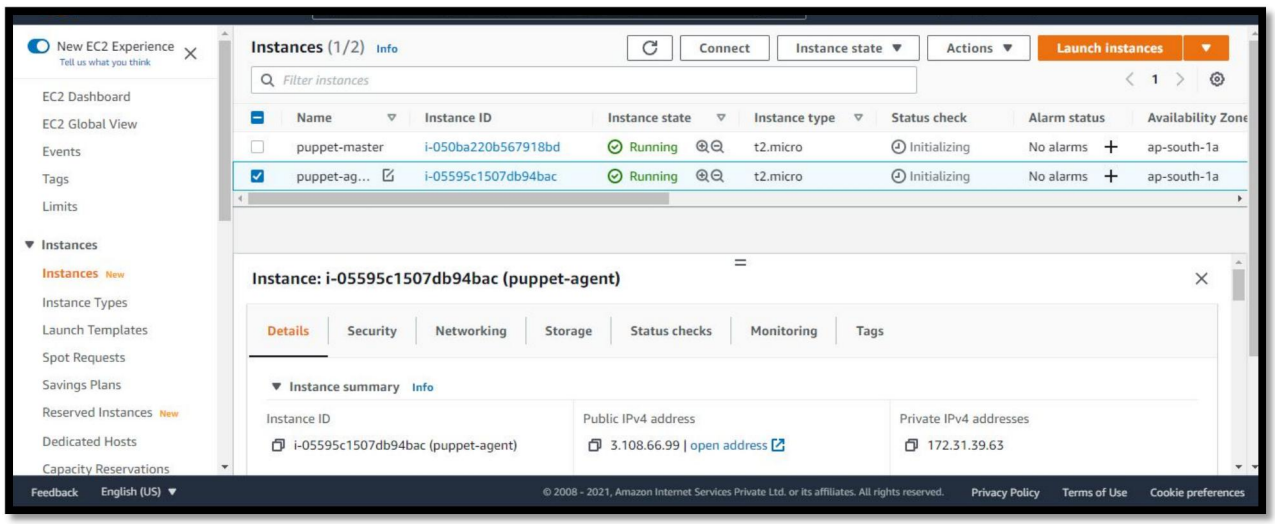
Step 9:


Select an existing key pair or create a new key pair
A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types.
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.
Create a new key pair
Key pair type
RSA
ED25519
Key pair name
cloud-key1
Download Key Pair
You have to download the private key file (*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.
Cancel
Launch Instances
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

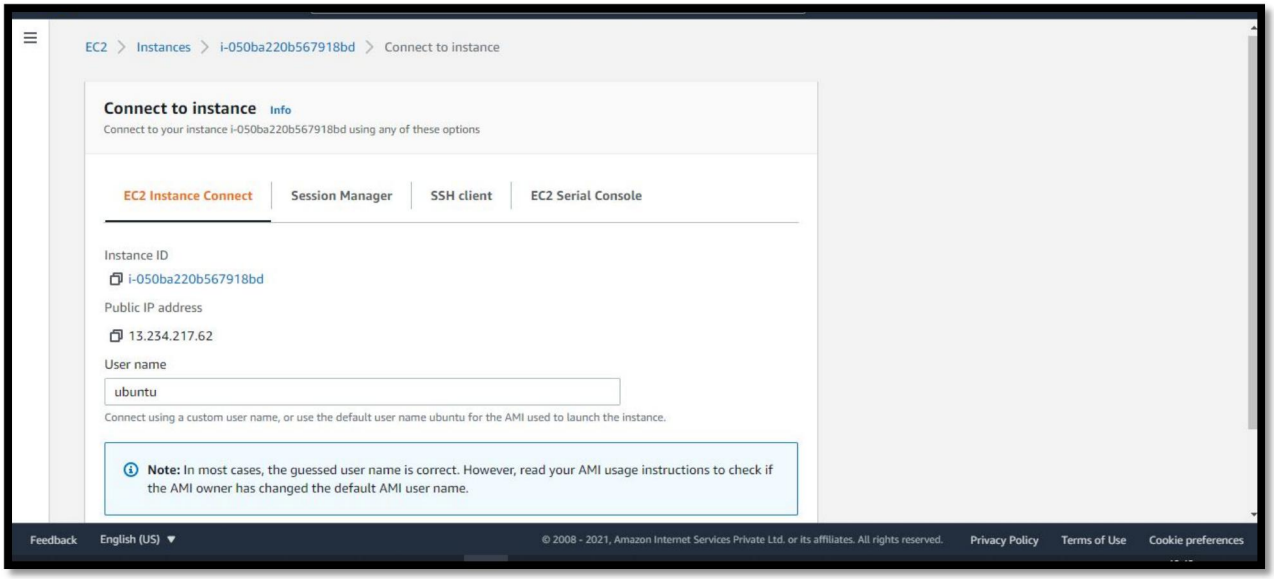
Step 10:



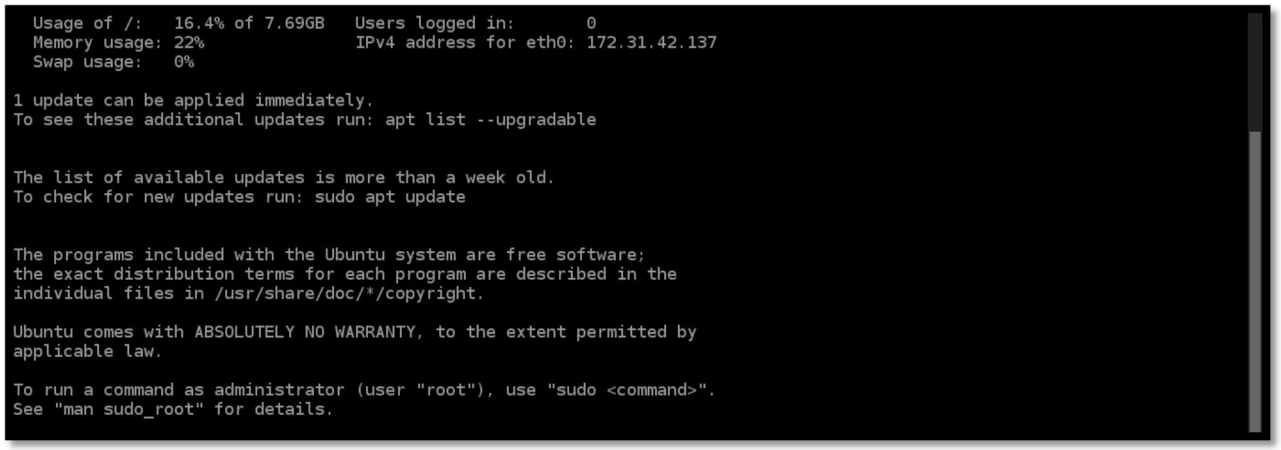
Step 11:



Step 12:


EC2 > Instances > i-050ba220b567918bd > Connect to instance
Connect to instance Info
Connect to your instance i-050ba220b567918bd using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 Serial Console
Instance ID
i-050ba220b567918bd
Public IP address
13.234.217.62
User name
ubuntu
Connect using a custom user name, or use the default user name ubuntu for the AMI used to launch the instance.
Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

Step 13:



Conclusion:

Hence, We Successfully installed and Configured Pull based Software Configuration Management and provisioning tools using Puppet.

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

Theory :

What is Configuration Management?

Configuration management is the process of maintaining software and computer systems (for example servers, storage, networks) in a known, desired and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes.System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate this task, by writing scripts.However, it is a difficult job when they are working on a massive infrastructure. The Configuration Management tool like a Puppet was introduced to resolve such issues.

What is Puppet?

Puppet is a system management tool for centralizing and automating the configuration management process. Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across the whole infrastructure of an organization.

Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby and uses its unique **D**omain **S**pecific **L**anguage (DSL) to describe system configuration.

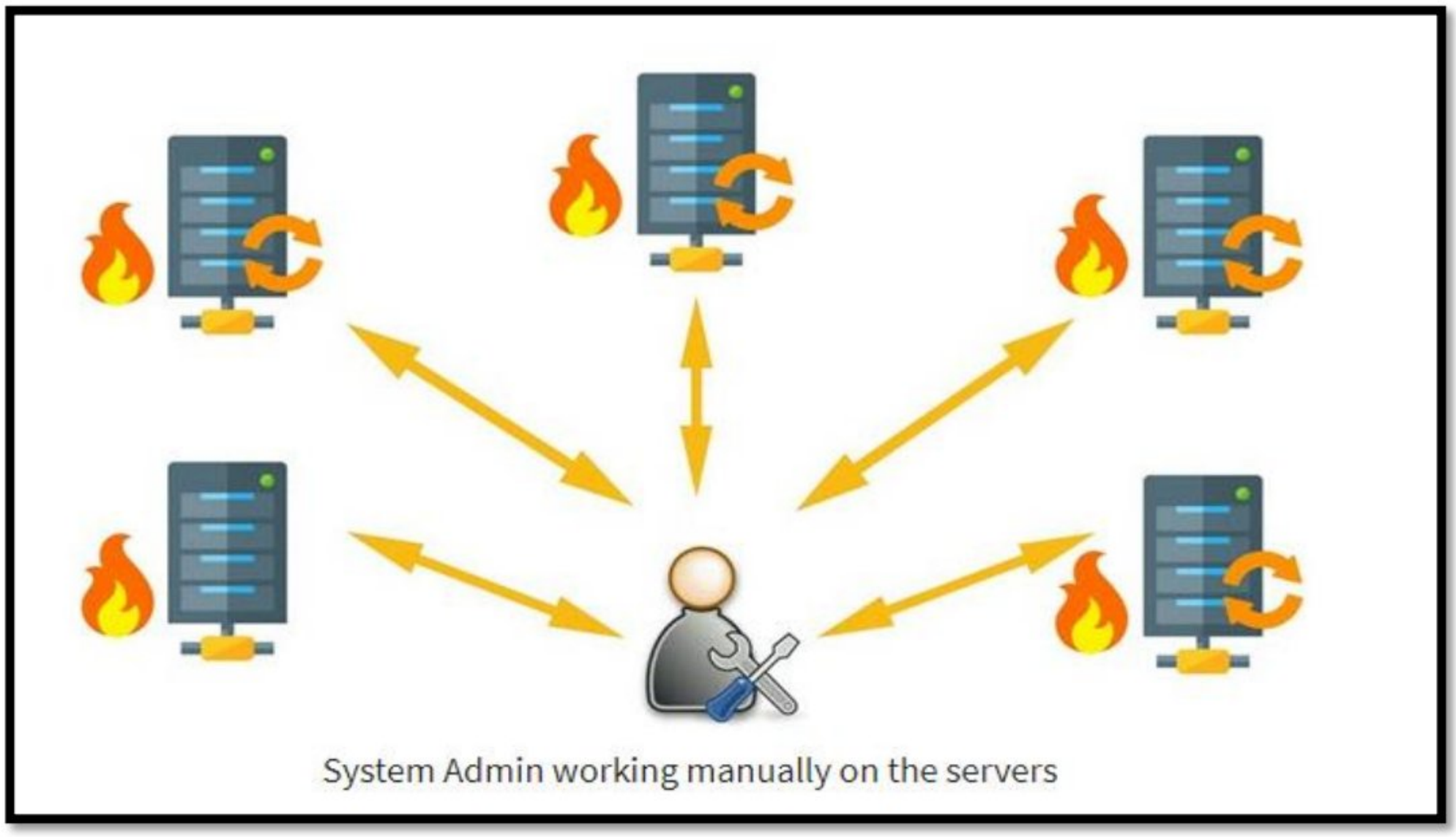
What are the Puppet versions?

Puppet comes in two versions:

- **Open Source Puppet**: It is a basic version of Puppet configuration management tool, which is also known as Open Source Puppet. It is available directly from Puppet's website and is licensed under the Apache 2.0 system.
- **Puppet Enterprise**: Commercial version that offers features like compliance reporting, orchestration, role-based access control, GUI, API, and command-line tools for effective management of nodes.

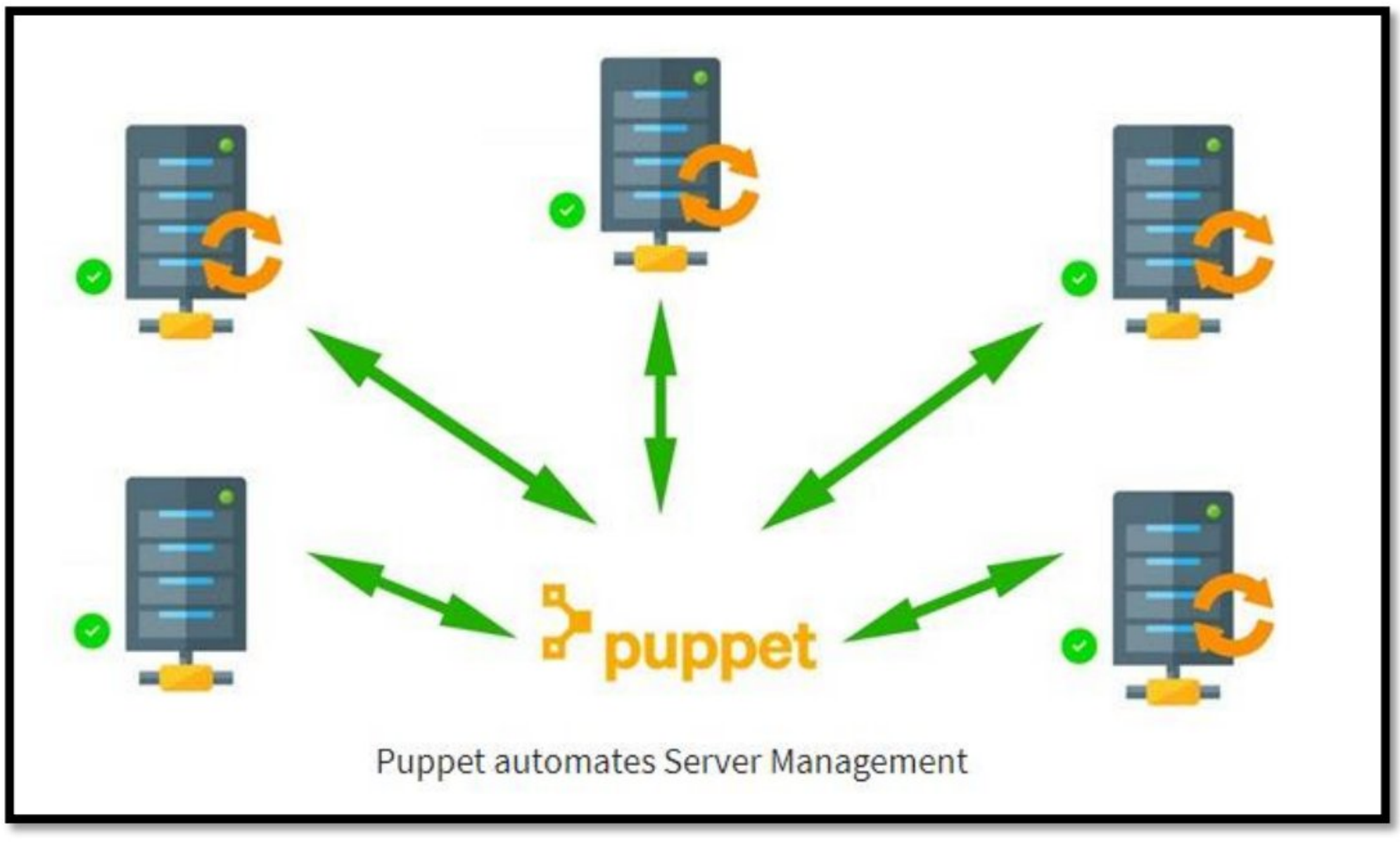
What Puppet can do?

For example, you have an infrastructure with about 100 servers. As a system admin, it's your role to ensure that all these servers are always up to date and running with full functionality.



System Admin working manually on the servers

To do this, you can use Puppet, which allows you to write a simple code which can be deployed automatically on these servers. This reduces the human effort and makes the development process fast and effective.



Puppet automates Server Management

Puppet performs the following functions:

- Puppet allows you to define distinct configurations for every host.
- The tool allows you to continuously monitor servers to confirm whether the required configuration exists or not and it is not altered. If the config is changed, Puppet tool will revert to the pre-defined configuration on the host.
- It also provides control over all the configured system, so a centralized change gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements the **infrastructure as a code because policies and configurations are written as code.**

Puppet DSL and Programming Paradigms

Before we learn Puppet DSL, let's understand programming paradigms:
A programming paradigm is a style you are using in computer programming.

Four types of paradigms are:

Imperative.

Declarative.

Functional (which is considered a subset of that declarative paradigm)

Object-oriented.

We will focus on Imperative and Declarative.

Imperative Paradigms:

This programming paradigm expresses the logic of a computation(What to do) and describes its control flow(How to do)
Example:
Assume you are going to your office, you book a cab and start giving step by step directions to the driver until you reach the office. Specifying what to do and how to do is an imperative style.

Declarative Paradigms:

This programming paradigm expresses the logic of a computation(What to do) without describing its control flow(How to do)
Example:
Assume you are going to your office, you book Uber cab and specify the final destination (Office). Specifying what to do not how to do is a declarative style.

Paradigm	What to do	How to do
Imperative	Yes	Yes
Declarative	Yes	No

uppet uses a declarative programming paradigm
Puppet uses a declarative programming approach.
Example: Create a user on the system:
It can be done using Imperative programming pattern by a shell script: Here we specify how to create the user and what commands to use on the operating system.

```
#!/bin/bash
# Script to add a user to Linux System
If [ $(id -u) -eq 0 ]; then
    $username=puppetuser
    read -s -p "Enter password : " password
    Egrep "^$username" /etc/passwd >/dev/null
    If [ $? -eq 0 ]; then
        echo "$username exists!"
        exit 1
    else
        useradd -m -p $password $username
        [ $? -eq 0 ] && echo "User has been added to the system!" || echo "Failed to add a user!"
    fi
```

However, it can be done using Declarative programming pattern with only a few lines of puppet code, Puppet domain specific language (DSL), and still achieve the same result.

```
user { "puppetuser":
    ensure => "present",
}
```

Deployment models of configuration management tools

There are two deployment models for configuration management tools :

- Push-based deployment model: initiated by a master node.
- Pull-based deployment model: initiated by agents.

Push-based deployment model:

In this deployment model master server pushes the configurations and software to the individual agents. After verifying a secure connection, the master runs commands remotely on the agents. For example, Ansible and Salt Stack.

Pull-based deployment model.

In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software and then configure themselves accordingly — for example, Puppet and Chef.

How Puppet works?

Puppet is based on a Pull deployment model, where the agent nodes check in regularly after every **1800** seconds with the master node to see if anything needs to be updated in the agent. If anything needs to be updated the agent pulls the necessary puppet codes from the master and performs required actions.

Let's explain it by an example:

Example: Master – Agent Setup:

The Master:

A Linux based machine with Puppet master software installed on it. It is responsible for maintaining configurations in the form of puppet codes. The master node can only be Linux.

The Agents:

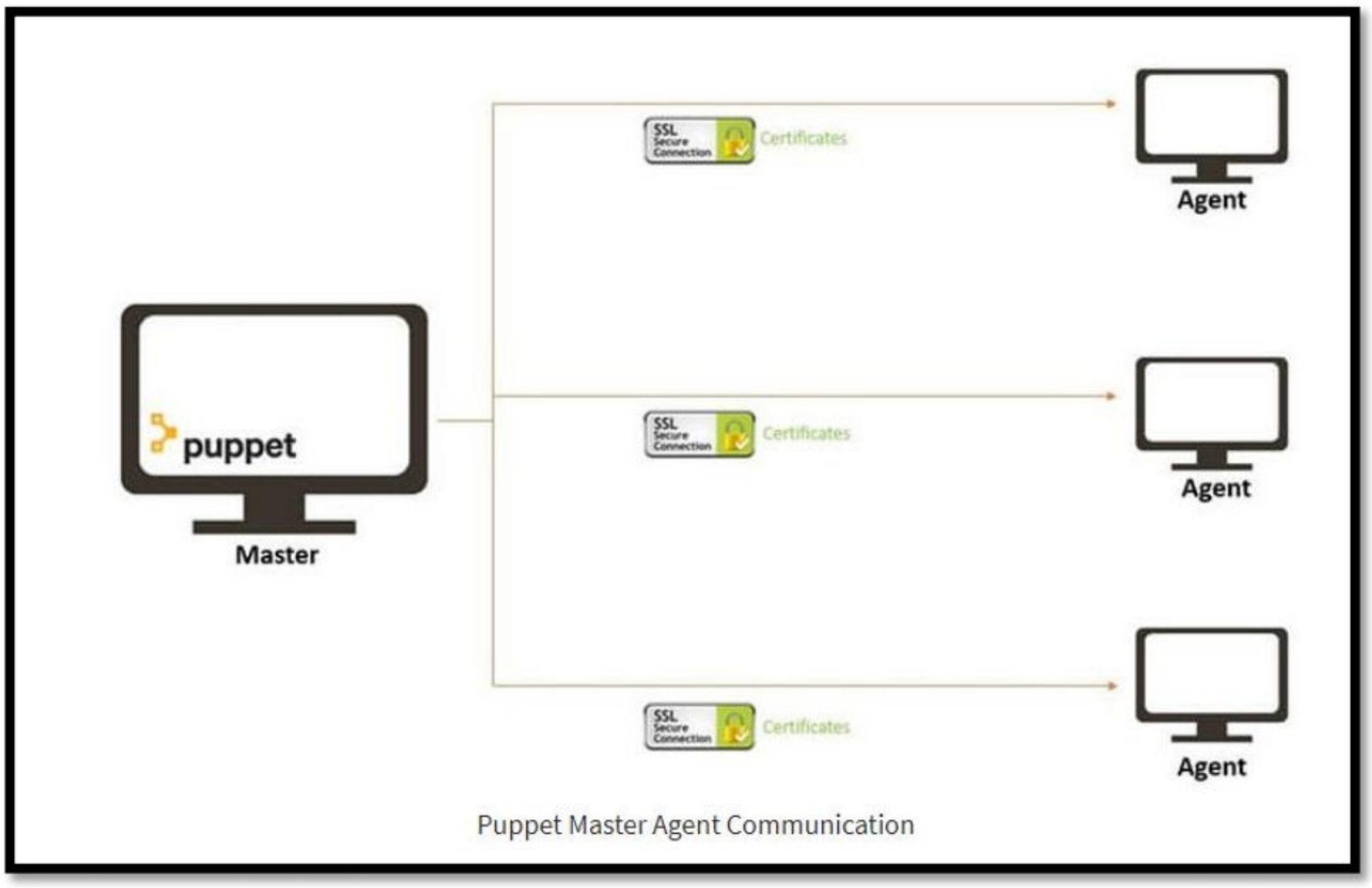
The target machines managed by a puppet with the puppet agent software installed on them.

The agent can be configured on any supported operating system such as Linux or Windows or Solaris or Mac OS.

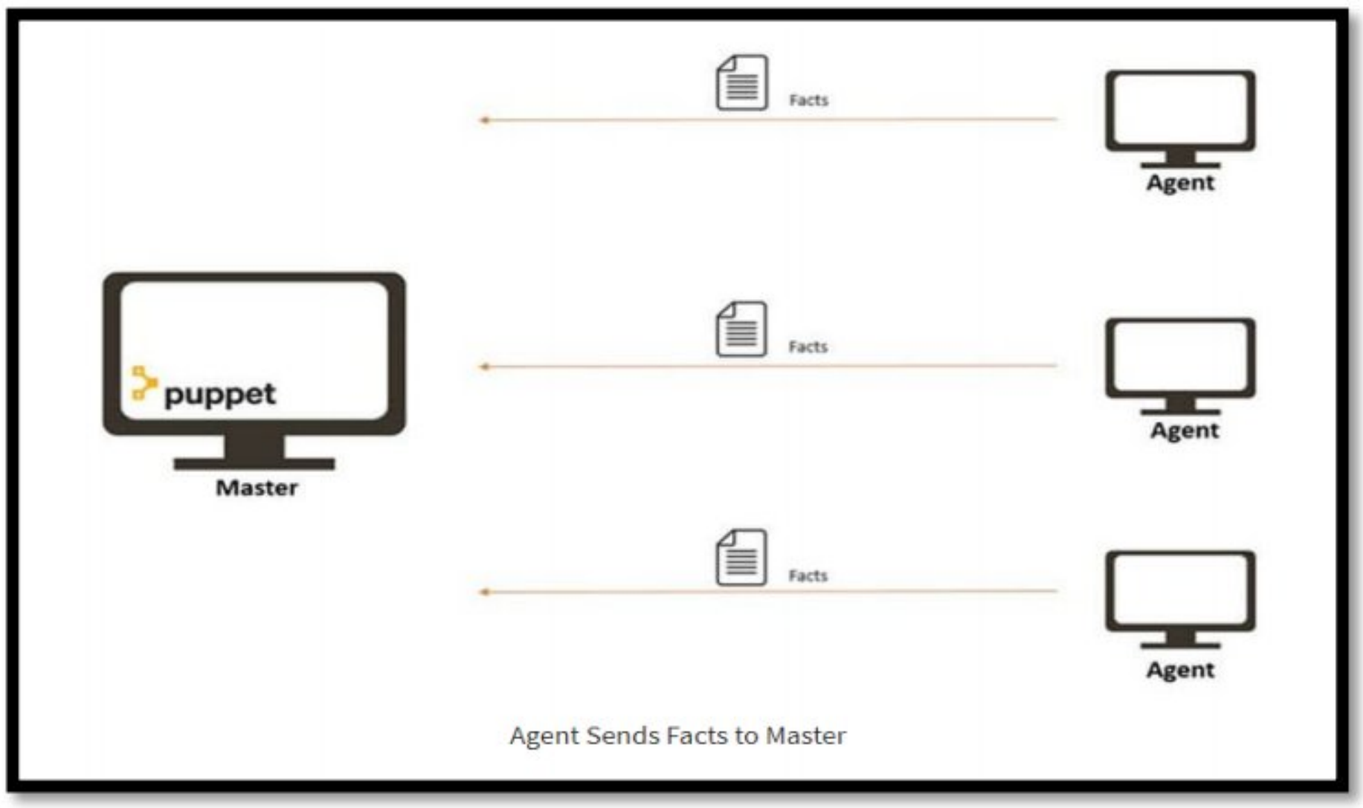
The communication between master and agent is established through secure certificates.

Communication between the Master and the Agent:

Step 1) Once the connectivity is established between the agent and the master, the Puppet agent sends the data about its state to the Puppet master server. These are called Facts: This information includes the hostname, kernel details, IP address, file name details, etc.…

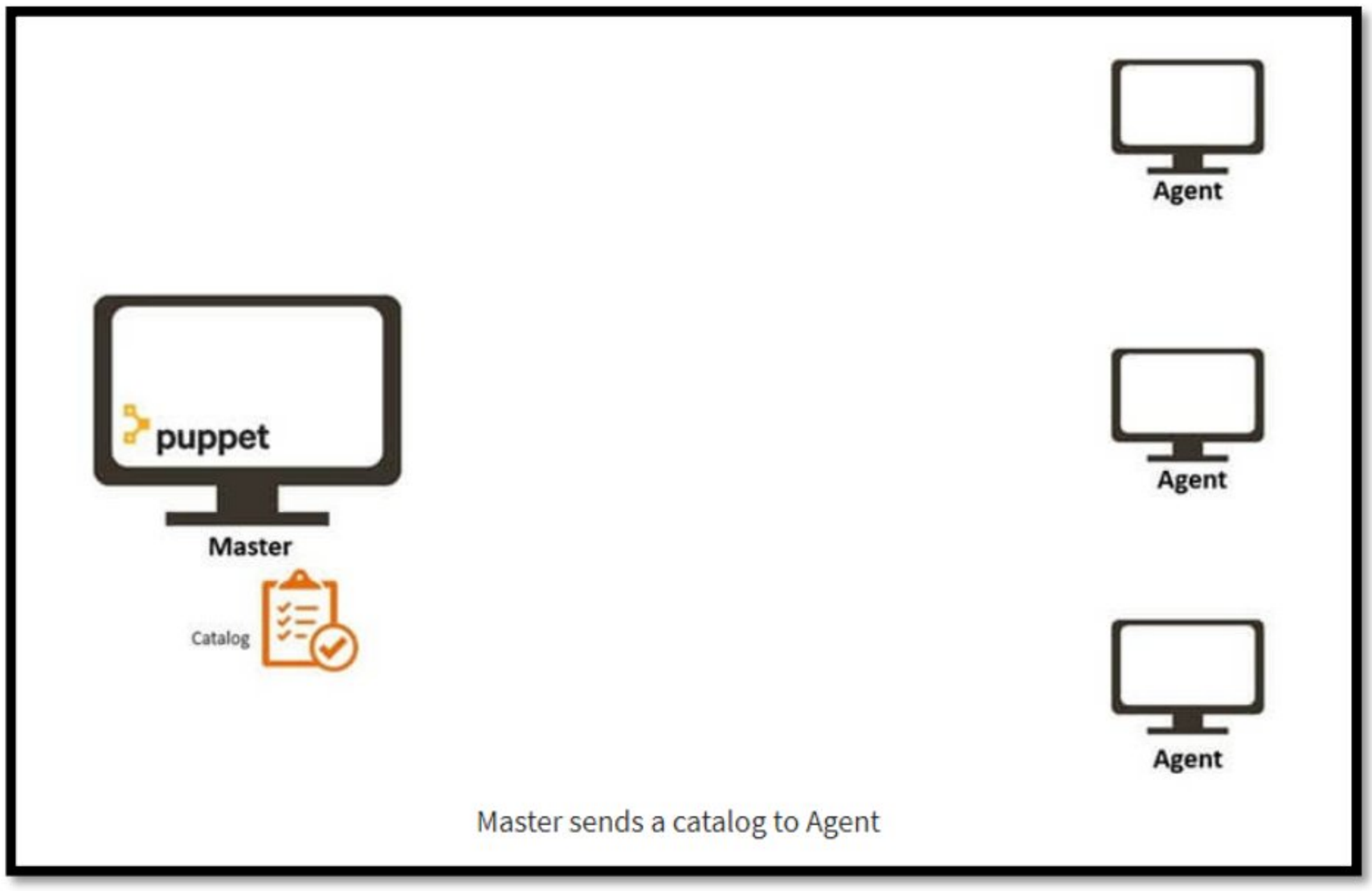


Puppet Master Agent Communication

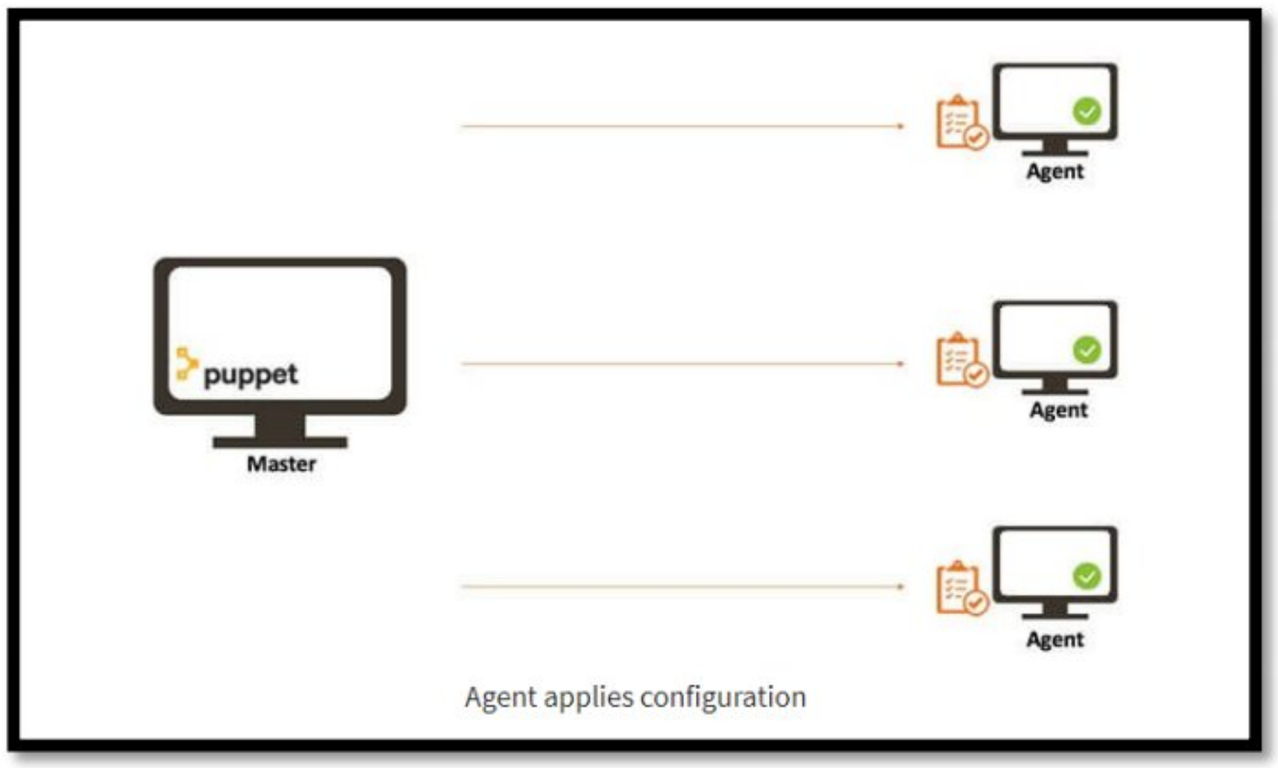


Agent Sends Facts to Master

Step 2) Puppet Master uses this data and compiles a list with the configuration to be applied to the agent. This list of configuration to be performed on an agent is known as a **catalog.** This could be changed such as package installation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.



Step 3) The agent uses this list of configuration to apply any required configuration changes on the node.
In case there are no drifts in the configuration, Agent does not perform any configuration changes and leaves the node to run with the same configuration.



Step 4) Once it is done the node reports back to puppet master indicating that the configuration has been applied and completed.

Puppet Blocks

Puppet provides the flexibility to integrate Reports with third-party tools using Puppet APIs.

Four types of Puppet building blocks are

1. Resources
2. Classes
3. Manifest
4. Modules

Puppet Resources:

Puppet Resources are the building blocks of Puppet.

Resources are the **inbuilt functions** that run at the back end to perform the required operations in puppet.

Puppet Classes:

A combination of different resources can be grouped together into a single unit called class.

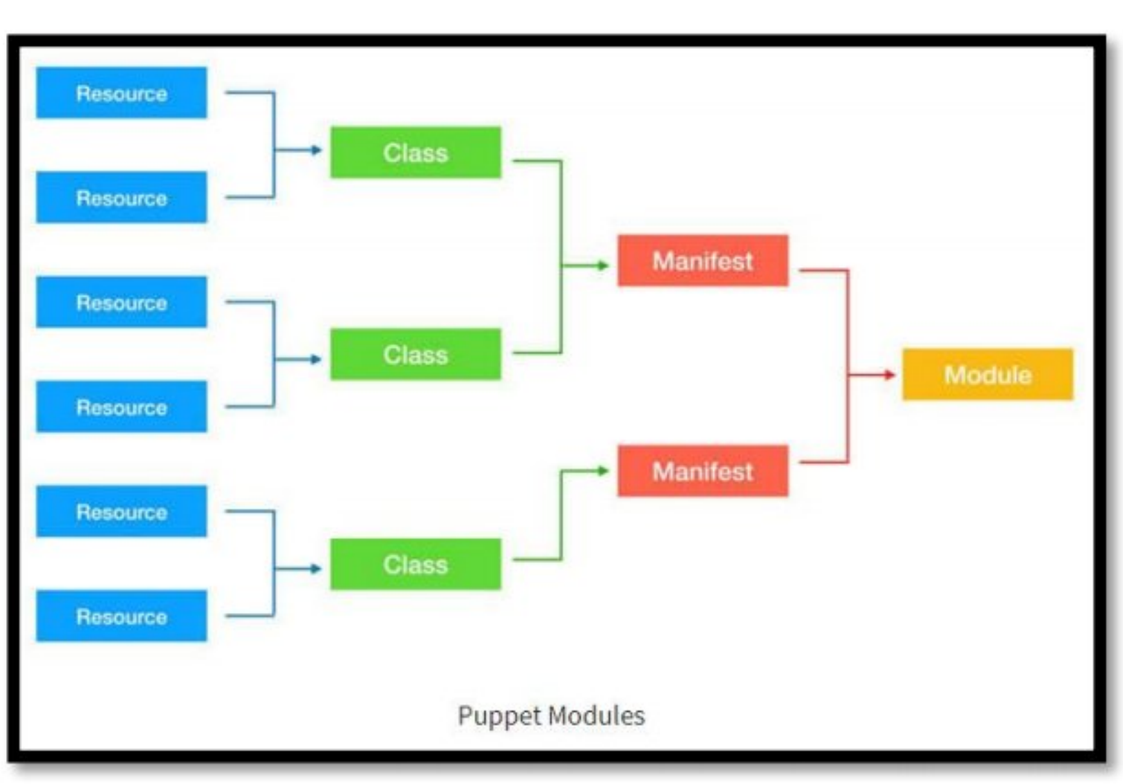
Puppet Manifest:

Manifest is a directory containing puppet DSL files. Those files have a .pp extension. The .pp extension stands for puppet program. The puppet code consists of definitions or declarations of Puppet Classes.

Puppet Modules:

Modules are a collection of files and directories such as Manifests, Class definitions. They are the re-usable and sharable units in Puppet.

For example, the MySQL module to install and configure MySQL or the Jenkins module to manage Jenkins, etc..



Puppet Modules

Types of Puppet resources

In general, a system consists of files, users, services, processes, packages, etc. In Puppet, these are called resources. Resources are the fundamental building blocks in

Puppet. All the operations on puppet agents are performed with the help of puppet resources.

Puppet resources are the readymade tools that are used to perform various tasks and operations on any supported platform. We can use a single puppet resource to perform a specific task, or we can use multiple puppet resources together to perform some complex application configurations deployments.

Resources can have different types. Puppet uses **resources** and **resource types** in order to describe a system's configuration.

There are three kinds of resource types:

1. Puppet core or built-in resource types.
2. Puppet defined resource types.
3. Puppet custom resource types.

Puppet core or built-in resource types:

Core or built-in resource types are the pre-built puppet resource types shipped with puppet software. All of the core or built-in Puppet resource types are written and maintained by Puppet team.

Puppet defined resource types:

Defined resource types are lightweight resource types written in Puppet declarative language using a combination of existing resource types.

Puppet custom resource types:

Custom resource types are completely customized resource types written in Ruby. command to display a list of Puppet relevant subcommands:

```
Puppet --help
```

```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet --help

Usage: puppet <subcommand> [options] <action> [options]

Available subcommands:

  Common:
    agent              The puppet agent daemon
    apply              Apply Puppet manifests locally
    config             Interact with Puppet's settings.
    help               Display Puppet help.
    lookup             Interactive Hiera lookup
    module             Creates, installs and searches for modules on the Puppet Forge.
    resource           The resource abstraction layer shell
```

In our case, we are interested in the subcommand "**resource**" which we will use to find the information about inbuilt puppet resource types.
In the terminal, type any of the following commands to display a list of **actions** associated with the puppet subcommand "**resource**":

```
Puppet help resource
Puppet resource --help
```

```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet help resource

puppet-resource(8) -- The resource abstraction layer shell
========

SYNOPSIS
--------
Uses the Puppet RAL to directly interact with the system.


USAGE
-----
puppet resource [-h|--help] [-d|--debug] [-v|--verbose] [-e|--edit]
  [-p|--param <parameter>] [-t|--types] [-y|--to_yaml] <type>
  [<name>] [<attribute>=<value> ...]
```

In this case, we have the **resource** as subcommand and **–types** as action.
Puppet has 49 inbuilt core resource types.
In the terminal, type the following command to display a list of available inbuilt puppet resource types:

```
puppet resource –types
```

```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet resource --types
augeas
cron
exec
file
filebucket
group
host
mount
notify
package
resources
schedule
scheduled_task
selboolean
selmodule
service
ssh_authorized_key
sshkey
stage
tidy
user
whit
yumrepo
zfs
```

Each type supports a list of **attributes**. These attributes provide a detailed description that Puppet uses to manage the resource.
To find out all the attributes associated with the puppet resource type, use the following command:
puppet describe <resource type name>
Parameters will list all the available attributes for that resource type.
puppet describe package

```
File  Edit  View  Search  Terminal  Help
[osboxes@puppetselfcontained ~]$ puppet describe package

package
=======
Manage packages.  There is a basic dichotomy in package
support right now:  Some package types (such as yum and apt) can
retrieve their own package files, while others (such as rpm and sun)
cannot.  For those package formats that cannot retrieve their own files,
you can use the `source` parameter to point to the correct file.
Puppet will automatically guess the packaging format that you are
using based on the platform you are on, but you can override it
using the `provider` parameter; each provider defines what it
requires in order to function, and you must meet those requirements
to use a given provider.
You can declare multiple package resources with the same `name`, as long
as they specify different providers and have unique titles.
Note that you must use the _title_ to make a reference to a package
resource; `Package[<NAME>]` is not a synonym for `Package[<TITLE>]` like
it is for many other resource types.
**Autorequires:** If Puppet is managing the files specified as a
package's `adminfile`, `responsefile`, or `source`, the package
resource will autorequire those files.


Parameters
----------
```

```
File  Edit  View  Search  Terminal  Help


Parameters
----------

- **adminfile**
    A file containing package defaults for installing packages.
    This attribute is only used on Solaris. Its value should be a path to a
    local file stored on the target system. Solaris's package tools expect
    either an absolute file path or a relative path to a file in
    `/var/sadm/install/admin`.
    The value of `adminfile` will be passed directly to the `pkgadd` or
    `pkgrm` command with the `-a <ADMINFILE>` option.

- **allow_virtual**
    Specifies if virtual package names are allowed for install and
    uninstall.
    Valid values are `true`, `false`, `yes`, `no`.
    Requires features virtual_packages.

- **allowcdrom**
    Tells apt to allow cdrom sources in the sources.list file.
    Normally apt will bail if you try this.
    Valid values are `true`, `false`.

- **category**
```

It's hard for a new person to understand and relate many unmanaged puppet code files. This is where we need some grouping to tie together operations. The aim is to solve a single problem, such as all operations required to configure ssh on a server or ntp service or a complete web server or database server from scratch.

What are Puppet Classes?

Puppet classes are the collection of puppet resources bundled together as a single unit.

Puppet introduced classes to make the structure re-usable and organized.

First, we need to define a class using class definition syntax; classes must be unique and can be declared only once with the same name:

```
class <class-name> {
<Resource declarations>
}
```

Example:

```
class ntpconfig {
file {
"/etc/ntp.conf":
ensure=> "present", content=> "server 0.centos.pool.ntp.org iburst\n",
}
}
```

So far we have only defined the class, but we have not used it anywhere. Meaning this code that we have written will never get executed unless we declare this class elsewhere.

Class Declaration

To use a defined class in code, use the **include** keyword.

```
class ntpconfig {
file {
"/etc/ntp.conf":
ensure=> "present",
content=> "server 0.centos.pool.ntp.org iburst\n",
}
}
include ntpconfig
```

Demo install NTP

First, make sure the NTP package is not already present on the server, the following command will return nothing if the telnet is not present on the server:

```
rpm -qa | grep -i ntp
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained etc]# rpm -qa | grep -i ntp
ntp-4.2.6p5-28.el7.centos.x86_64
python-ntplib-0.3.2-1.el7.noarch
fontpackages-filesystem-1.44-8.el7.noarch
ntpdate-4.2.6p5-28.el7.centos.x86_64
[root@puppetselfcontained etc]#
```

As we can see, the NTP package is already present on the server. Let's remove the existing NTP package:

```
yum remove ntp
```

After removing the package, ensure that ntp.conf file is not existing:

```
ls -lrt /etc/ntp.conf
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained etc]# ls -lrt /etc/ntp.conf
ls: cannot access /etc/ntp.conf: No such file or directory
[root@puppetselfcontained etc]#
```

Verify the ntp service does not exist by running the following command:

```
systemctl status ntp
```

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# systemctl status ntpd
Unit ntpd.service could not be found.
[root@puppetselfcontained demo]#
```

Create a new .pp file to save the code. From the command line:

```
vi demontp.pp
```

Change to insert mode by pressing i from the keyboard.
code to create a new file:

```
# Class Definition
class ntpconfig {
    # Installing NTP Package
  package {"ntp":
    ensure=> "present",
    }
    # Configuring NTP configuration file
  file {"/etc/ntp.conf":
    ensure=> "present",
    content=> "server 0.centos.pool.ntp.org iburst\n",
    }
    # Starting NTP services
  service {"ntpd":
    ensure=> "running",
    }
}
```

After done with editing : press esc
To save the file, press :wq!
Next step is to **check** whether the code has any syntax errors. Execute the following command:
puppet parser validate demontp.pp
Make sure that you switched to the **root** to be able to complete the test without any error, by executing the command :
su root
Test is the next step in the code creation process. Execute the following command to perform a smoke test:
Puppet applies demontp.pp --noop
Last step is to **run** the puppet in real mode and verify the output.
puppet apply demontp.pp

Puppet didn't perform anything because the demo class was just **defined** but not **declared**.
So, until you declare the puppet class, the code will not get applied.
Let's **declare** the demo class inside the same code using **include class name** at the end of the code:

```
# Class Definition
class ntpconfig {
    # Installing NTP Package
  package {"ntp":
    ensure=> "present",
    }
    # Configuring NTP configuration file
  file {"/etc/ntp.conf":
    ensure=> "present",
    content=> "server 0.centos.pool.ntp.org iburst\n",
    }
    # Starting NTP services
  service {"ntpd":
    ensure=> "running",
    }
}

# Class Declaration
include ntpconfig
```

Again **check** whether the code has any syntax errors. Execute the following command:
puppet parser validate demontp.pp
Make sure that you switched to the **root** to be able to complete the test without any error, by executing the command :
su root
Testing is the next step in the code creation process. Execute the following command to perform a smoke test:
Puppet apply demontp.pp --noop
Last step is to **run** the puppet in real mode and verify the output.
puppet apply demontp.pp
This time the code gets applied because the class was defined and then declared.

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# puppet apply demontp.pp
Notice: Compiled catalog for puppetselfcontained.example.com in environment production
in 0.69 seconds
Notice: /Stage[main]/Ntpconfig/Package[ntp]/ensure: created
Notice: /Stage[main]/Ntpconfig/File[/etc/ntp.conf]/content: content changed '{md5}dc9e5
754ad2bb6f6c32b954c04431d0a' to '{md5}83fd3914bd6bb10f2b717c277d995b75'
Notice: /Stage[main]/Ntpconfig/Service[ntpd]/ensure: ensure changed 'stopped' to 'runni
ng'
Notice: Applied catalog in 1.88 seconds
[root@puppetselfcontained demo]#
```

Ensure that ntp.conf is now existing:
ls -lrt /etc/ntp.conf
Verify the ntp service has been started by running the following command:
systemctl status ntpd

```
File  Edit  View  Search  Terminal  Help
[root@puppetselfcontained demo]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; disabled; vendor preset: disab
led)
   Active: active (running) since Sat 2019-03-23 04:14:35 EDT; 15s ago
  Process: 21487 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SU
CCESS)
```

Conclusion:
Hence, We Successfully Studied Software Configuration Management and provisioning using Puppet Blocks(Manifest, Modules, Classes, Function).

Assignment no. 01

Q. Case study on DevOps implementation in real world.

CUSTOMER

The Customer is a US multi-business enterprise operating in a range of industries, such as fashion retail, hospitality, and restaurant.

CHALLENGE

The Customer had an IT infrastructure consisting of more than 50 servers with a range of integrated systems: an ecommerce website, CRM, a data warehouse, and others.

The number of the Customer's clients was rapidly increasing together with their needs, so the Customer needed to frequently update their applications to keep their clients satisfied. To meet these and other business needs, the Customer needed their IT infrastructure properly managed, regularly enhanced without critical operational errors and system failures, and continuously monitored to make sure their web services were highly available.

SOLUTION

To solve the Customer's infrastructure management tasks, ScienceSoft assigned a team of DevOps practitioners with the expertise in system administration and software development. By combining the development (Dev) and operations (Ops) specialists' efforts, ScienceSoft aimed at significantly accelerating the delivery of new software features, fixes, and updates in alignment with the Customer's business objectives.

The Customer's IT infrastructure included three key elements integrated with each other: an ecommerce website, CRM, and data warehouse. To ensure the faultless performance of the integrated systems, ScienceSoft set up and managed the tools for load balancing, infrastructure monitoring and log management. To manage the Customer's ecommerce website, ScienceSoft's DevOps practitioners implemented the continuous integration and continuous delivery (CI/CD) pipeline.

CI/CD PIPELINE IMPLEMENTATION

Using Stash as a source code repository and Jenkins as a CI/CD automation tool, DevOps engineers designed a CI/CD pipeline to accelerate the processes of developing, testing and releasing the updates and bug fixes for the Customer's web applications based on Pimcore, Magento, Akeneo, etc. ScienceSoft's team applied

Docker to containerize the Customer's web applications and used the Distributed Cloud Operating System (DC/OS) based on the Apache Mesos distributed systems kernel to manage the containers. With the use of Ansible, ScienceSoft's team managed applications configurations.
With CI/CD pipeline implementation, ScienceSoft's DevOps practitioners aligned the development, test and stage environments with the production environment to eliminate the differences between them, and automated performance testing of web applications.

LOAD BALANCING AND HTTP CACHING

For proper load balancing, network traffic distribution and HTTP caching across the Customer's IT infrastructure, ScienceSoft's DevOps engineers used the following tools:
o Nginx – as a web server processing users' requests.
o Traefik – as a load balancer responding to users' requests and directing them to the right resource.
o HAProxy – as a load balancer distributing the requests across the Customer's multiple servers while ensuring their high availability.
o Varnish – as an HTTP reverse proxy cache accelerating web servers' response to the users' requests of the Customer's clients.

IT INFRASTRUCTURE MONITORING AND LOG MANAGEMENT

To monitor disk usage, RAM and CPU consumption, DevOps engineers set up and applied Zabbix. ScienceSoft's team used Blackbox Exporter to monitor web services availability. To collect the data on web services availability from Blackbox Exporter and send alert notifications on the web services issues, ScienceSoft's DevOps engineers used Prometheus. For constant container performance monitoring, ScienceSoft's DevOps engineers applied Container Advisor.

For convenient web application log management, ScienceSoft's team applied the ELK stack: Logstash (L) parsed web application logs, sent them to Elasticsearch (E) where the logs were collected and filtered, and Kibana (K) worked as a visualization tool allowing for using the data stored in Elasticsearch to build charts and graphs with analytics results.

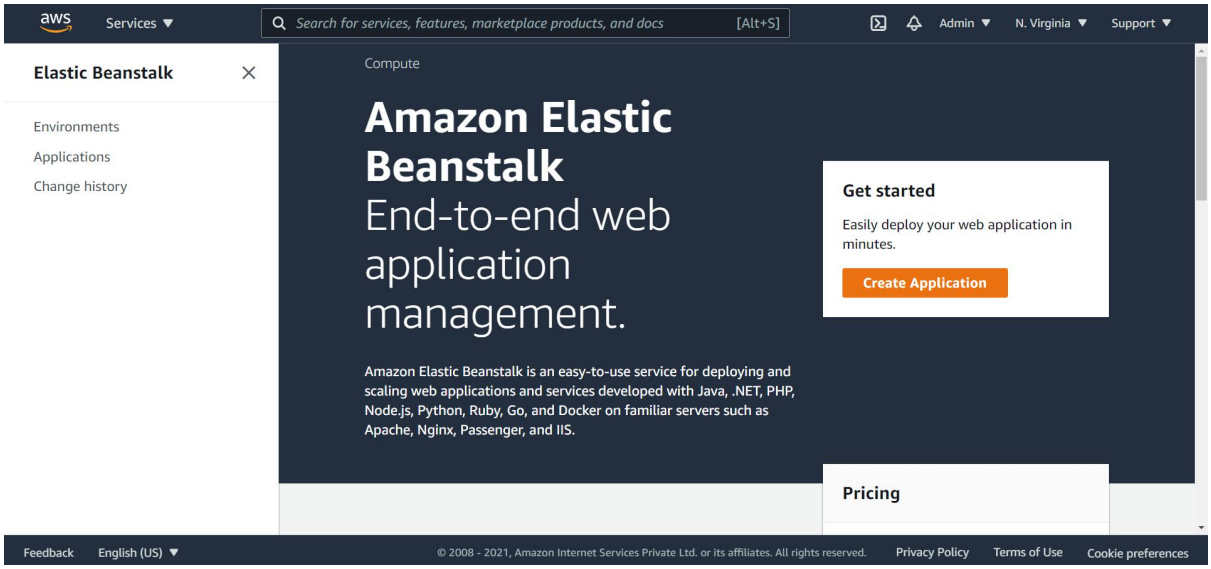
RESULTS

With the DevOps practices that ScienceSoft introduced and applied, the Customer got the proper management of their IT infrastructure and benefited from the high availability of the IT infrastructure. As a result of ScienceSoft's DevOps engineers' work, the Customer got the possibility to enhance the ecommerce solution frequently without disruptions to the business process.

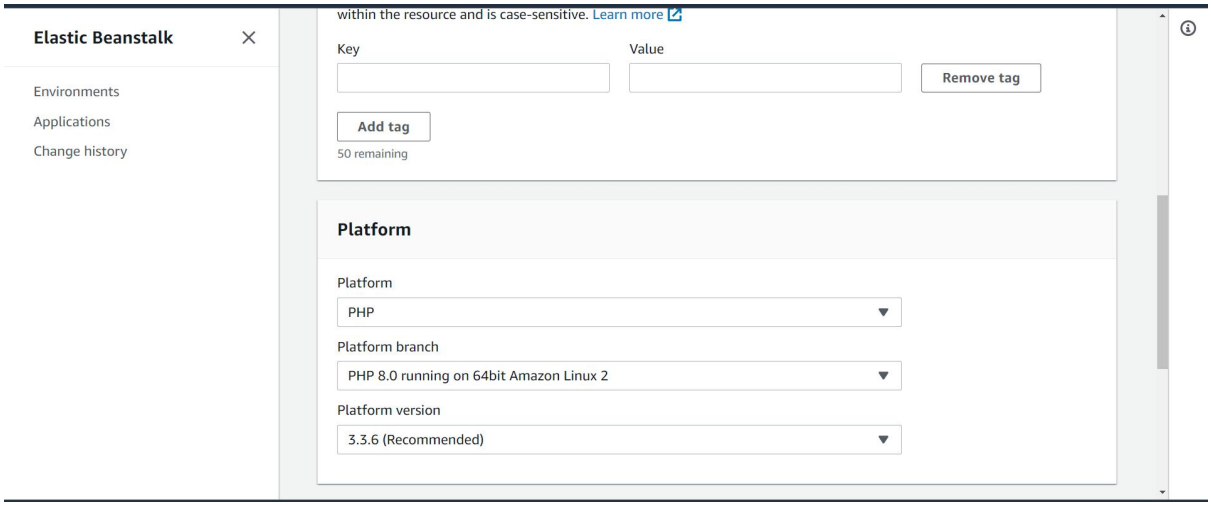
Assignment no. 02

IMPLEMENT AWS CodePipeline

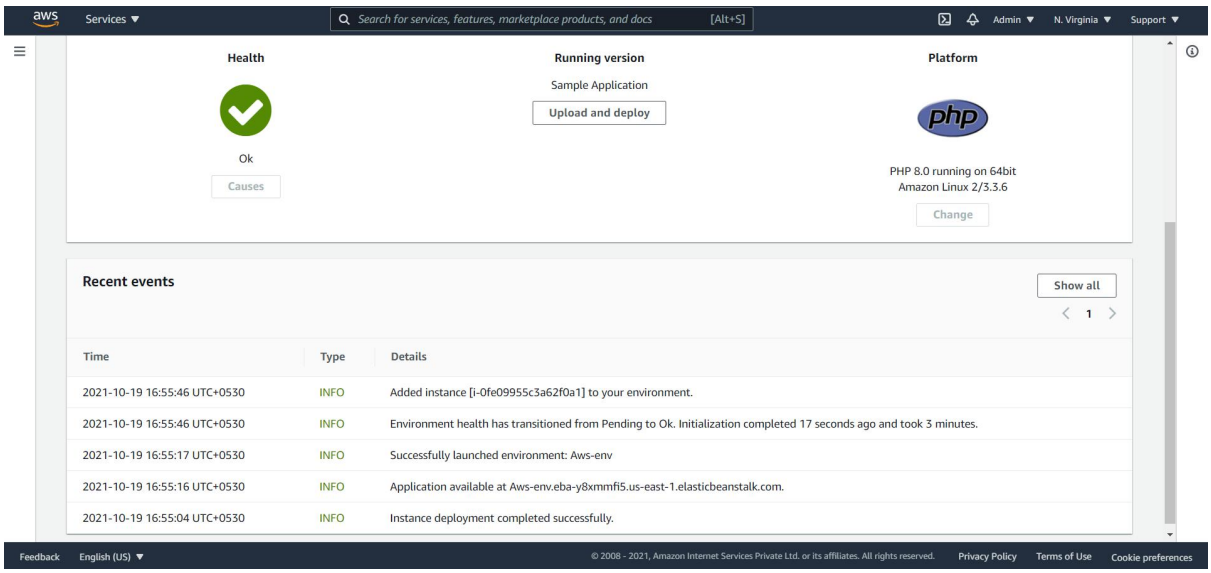
Step 1: Create a deployment environment using AWS EWS Beanstalk



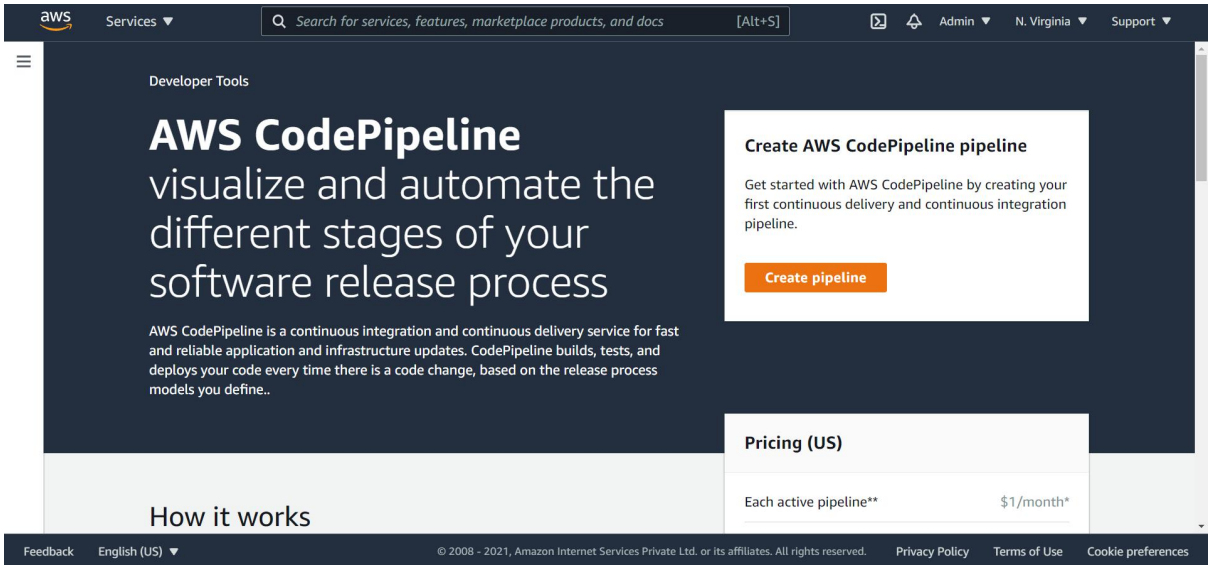
Step 2: Choose PHP from the drop-down menu and then click Launch Now



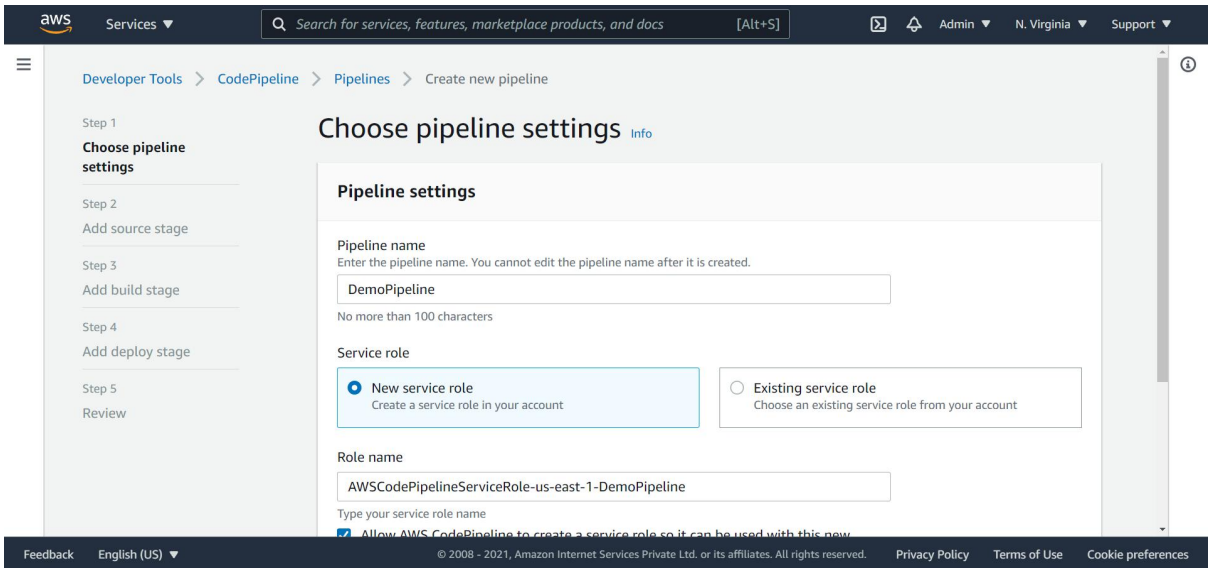
Step 3: Elastic Beanstalk has created sample environment to deploy your application



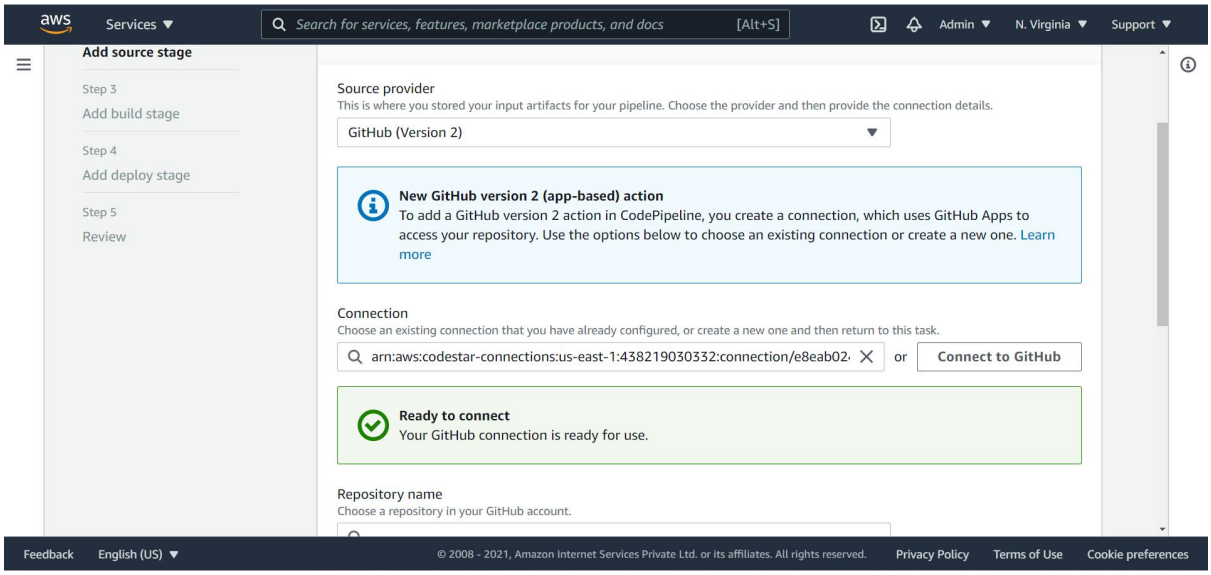
Step 4: Create a pipeline using AWS CodePipeline Console



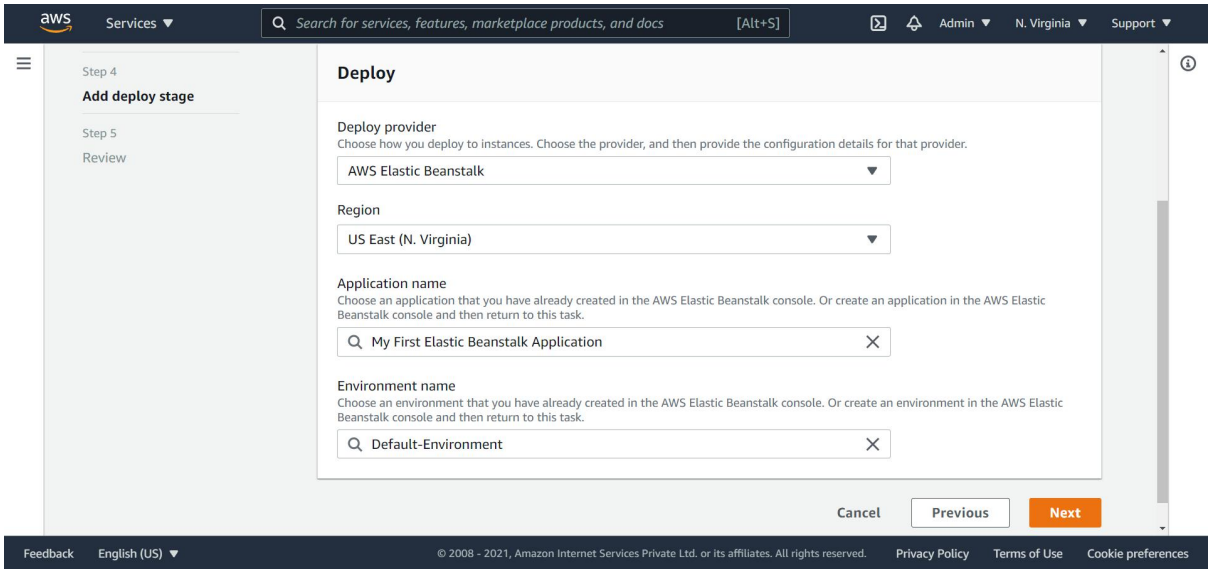
Step 6: Give name to pipeline as DemoPipeline and click on Next Step



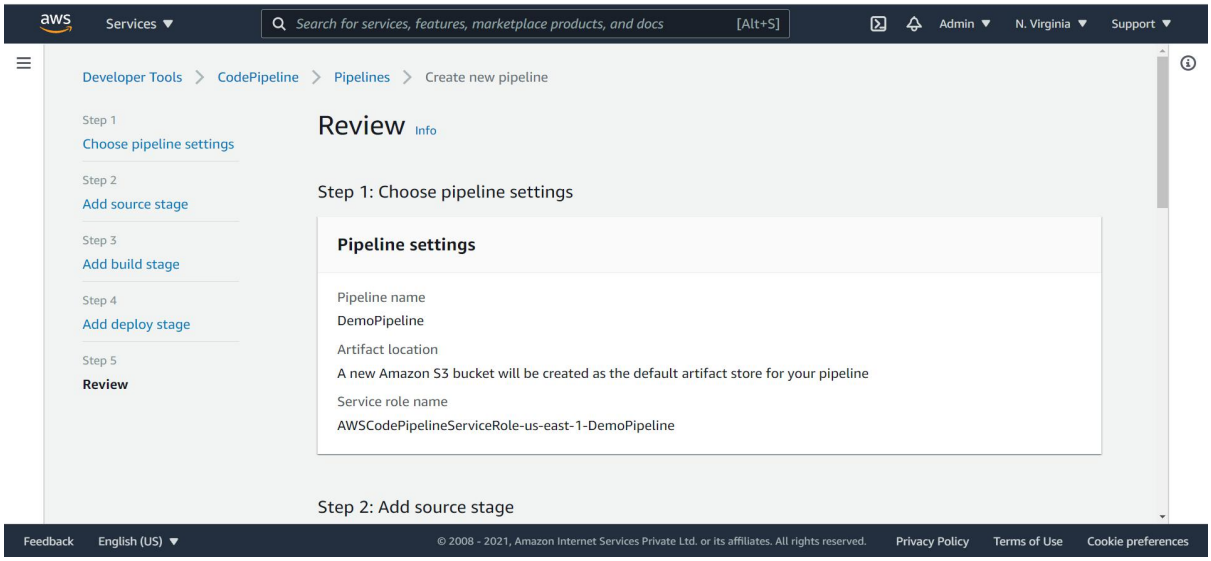
Step 7: Select GitHub as Source provider and click on Connect to GitHub in Connect to GitHub section

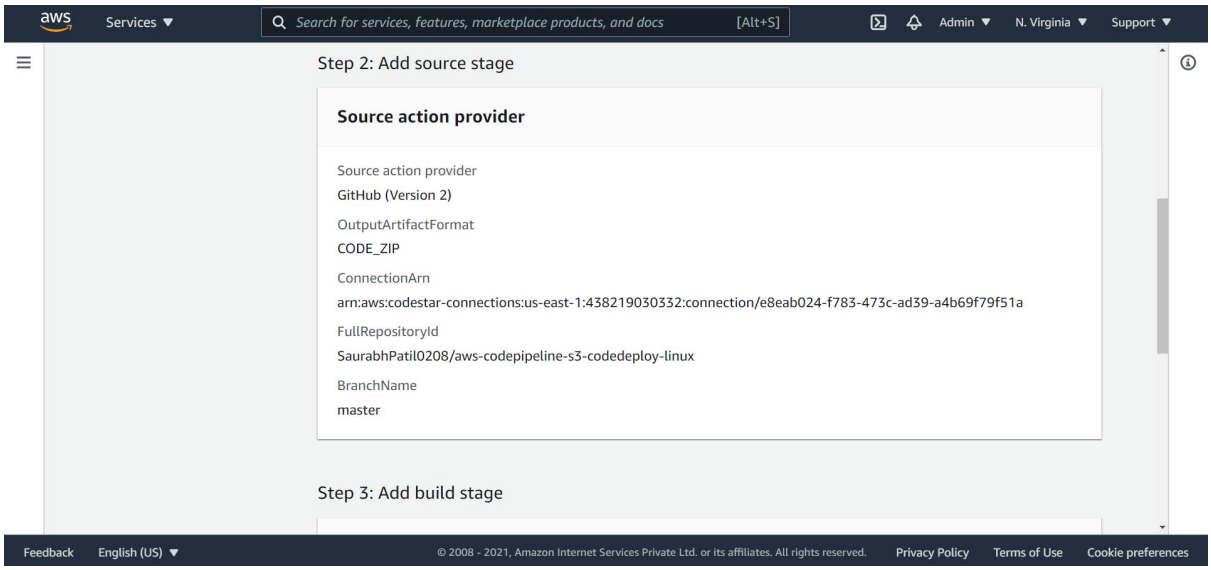


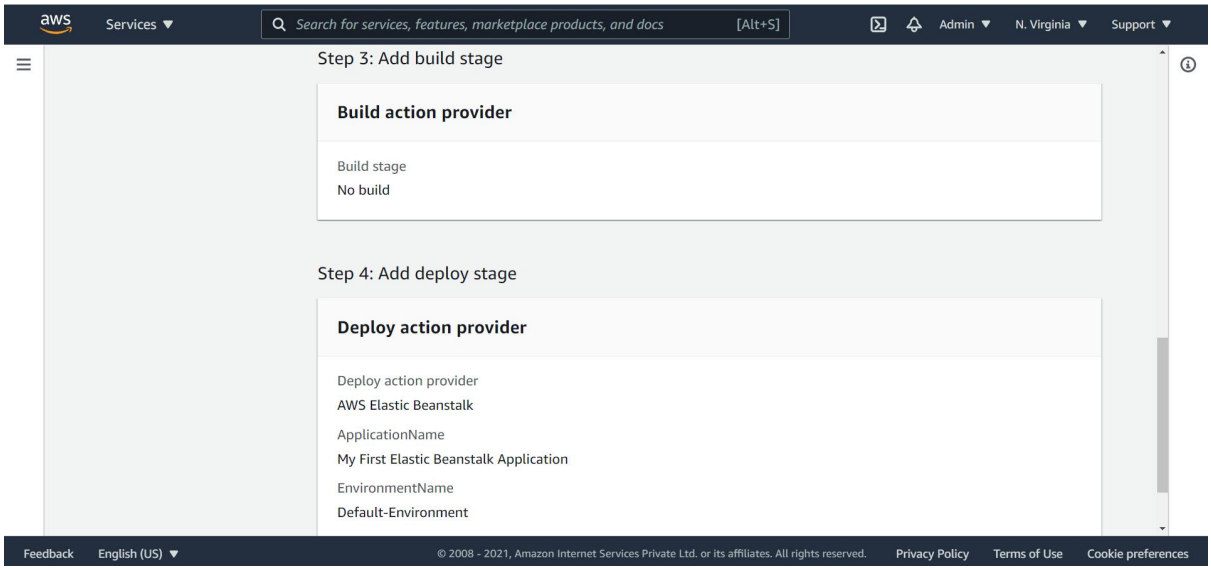
Step 8: In Deploy page choose AWS Elastic Beanstalk as Deploy provider and give Application and Environment name of your choice

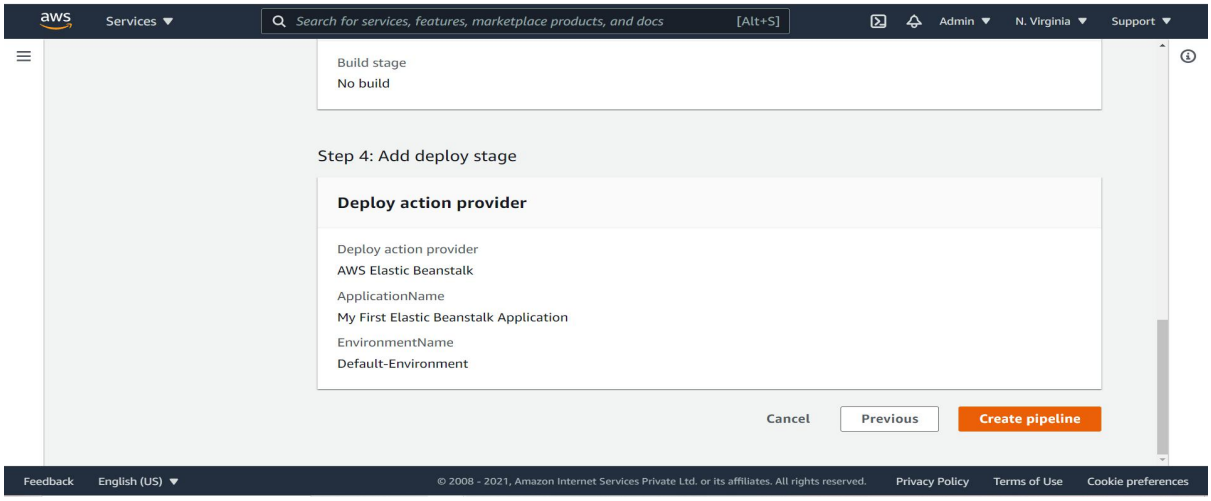


Step 10: In Review page, review the information and click Create Pipeline

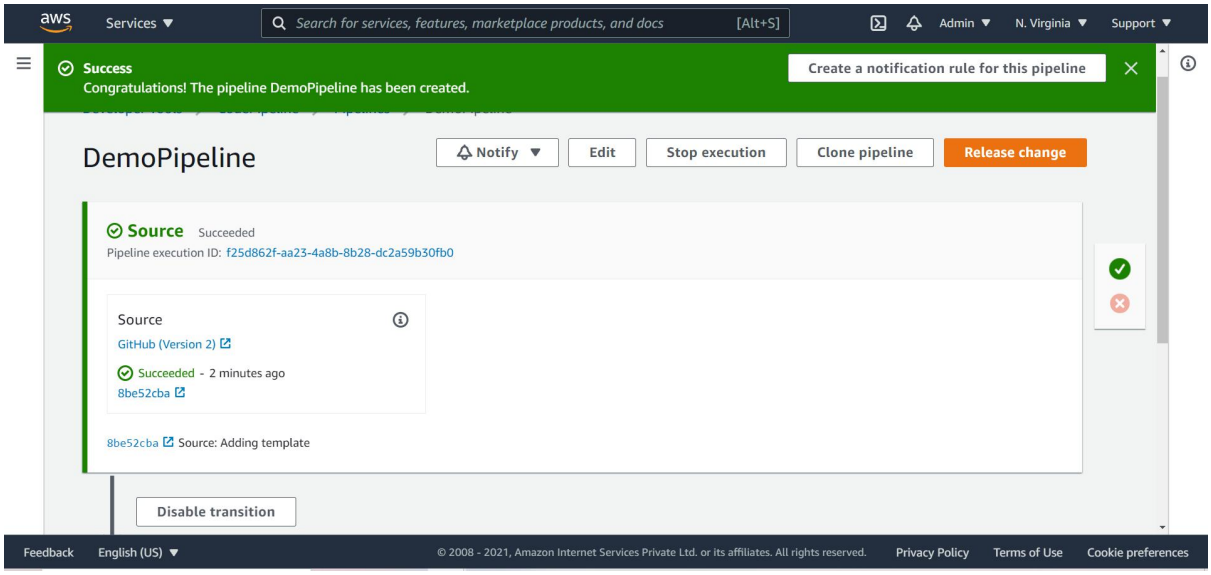


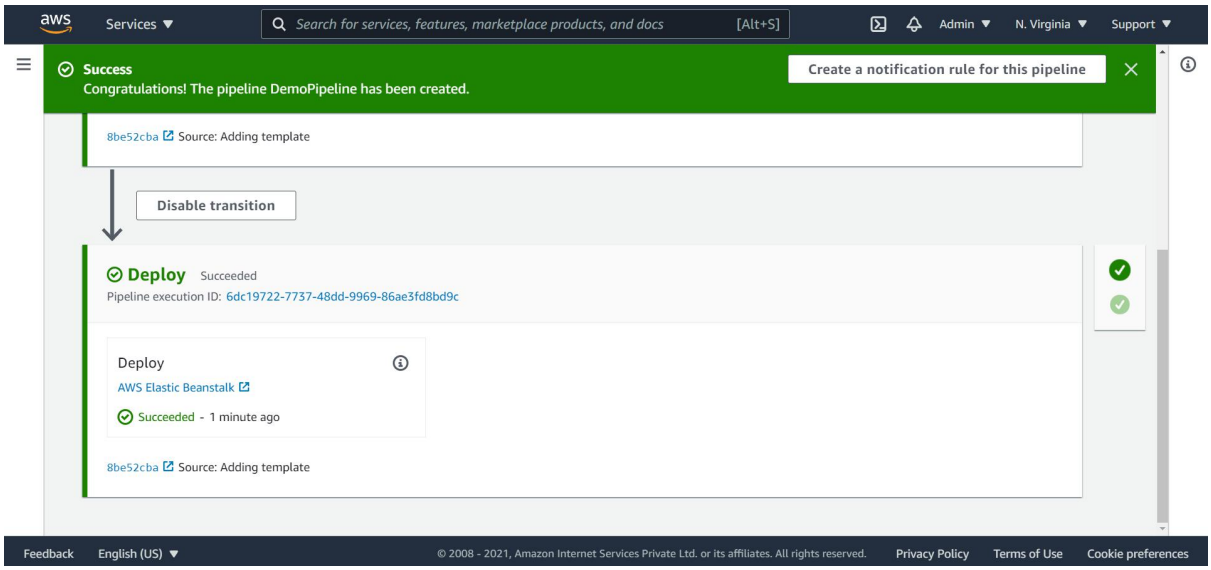




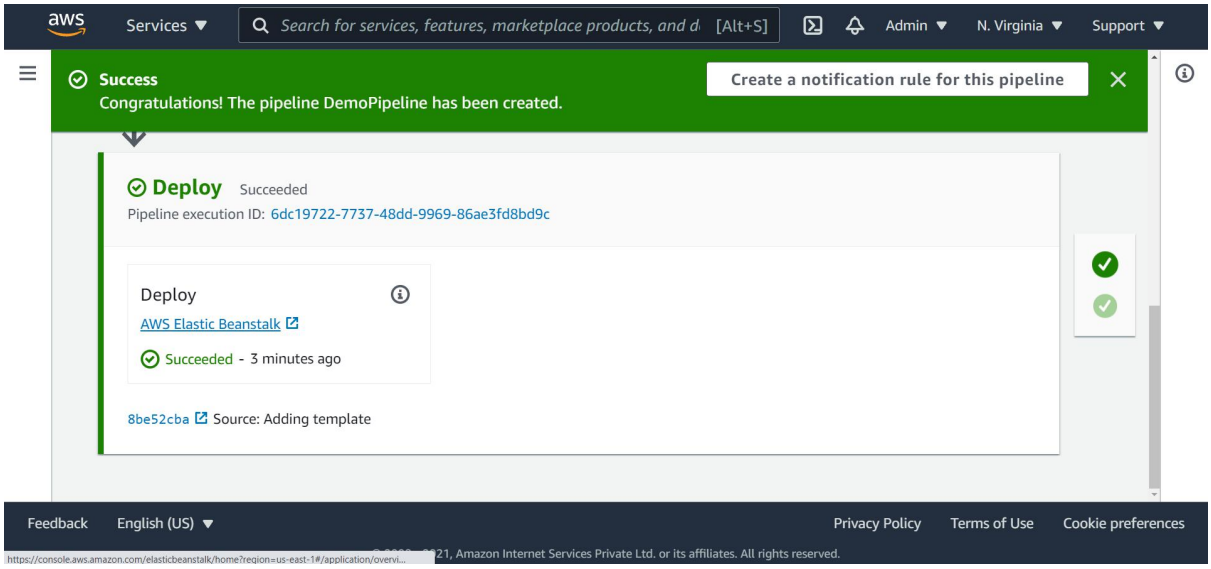


Step 10: Check the status of Pipeline created

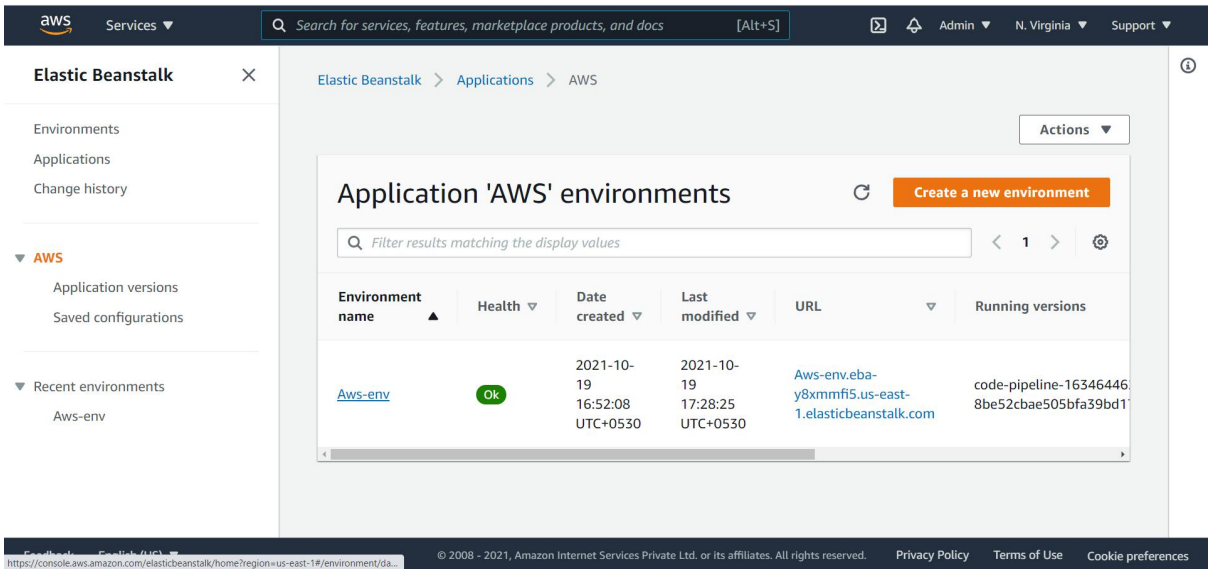




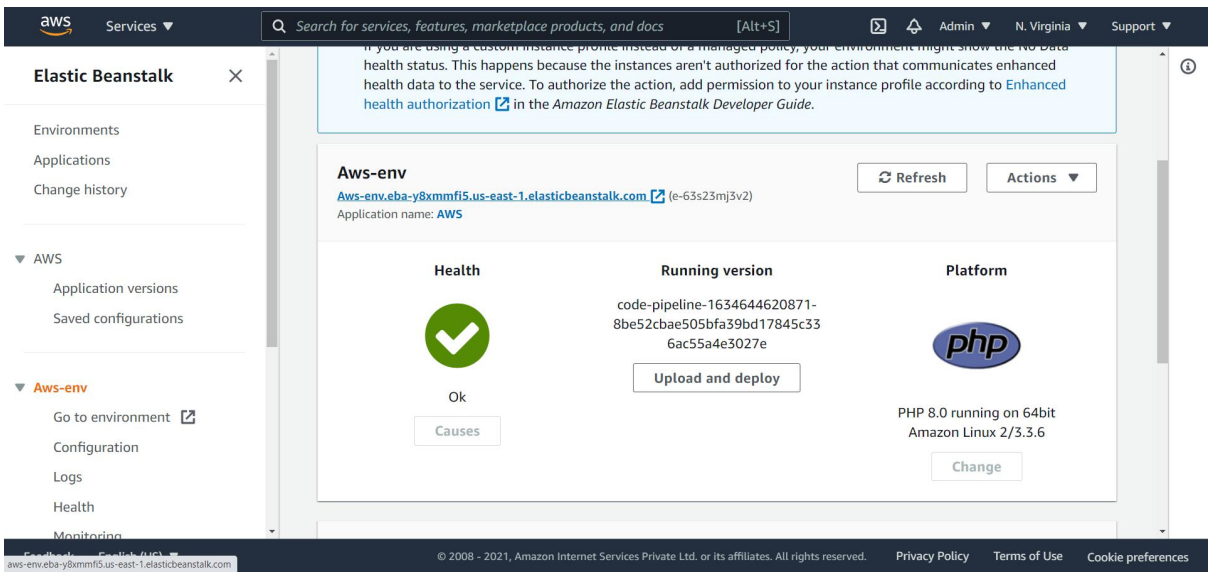
Step 11: Click AWS Elastic Beanstalk in status area of Deploy stage



Step 12: Click on the environment you created earlier



Step 13: Click on the URL below the created environment to deploy your sample website



Congratulations!

You have successfully created a pipeline that retrieved this source application from an Amazon S3 bucket and deployed it to three Amazon EC2 instances using AWS CodeDeploy.

For next steps, read the AWS CodePipeline Documentation.