Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 31-01-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma **ROLL: I011** **DIV: IT1-1**

EXPERIMENT NO. 1

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To understand Version Control System / Source Code Management, install git and create a GitHub account

THEORY:

A Version Control System is a tool you use to track, make, and manage changes to your software code. It's also called source control.

A version control system helps developers store every change they make to a file at different stages so they and their teammates can retrieve those changes at a later time.

There are three types of version control systems, which are:

- Local Version Control Systems
- Centralized Version Control Systems
- Distributed Version Control Systems.

What is a Local Version Control System (LVCS)?

Version Control Systems (VCS) are essential tools in software development, enabling teams to track changes, collaborate efficiently, and maintain code integrity. There are three primary types of VCS: Local, Centralized, and Distributed.

Local Version Control System (LVCS): In an LVCS, developers manage versioning on their individual machines. This involves keeping multiple copies of files in different directories to track changes. While simple, this method is prone to errors and lacks collaboration features, making it less suitable for team environments.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

What is a Centralized Version Control System (CVCS)?

Centralized Version Control System (CVCS): A CVCS uses a central server to store all file versions. Developers commit their changes directly to this central repository. This setup provides a unified view of the project and facilitates collaboration. However, it has a single point of failure; if the server becomes unavailable, developers cannot access the repository or commit changes. Examples of CVCS include Subversion (SVN) and Perforce.

What is a Distributed Version Control System (DVCS)?

Distributed Version Control System (DVCS): In a DVCS, each developer has a complete copy of the repository, including its full history, on their local machine. This allows for offline work and provides redundancy; if a central server fails, any local repository can restore the system. DVCS supports complex workflows and enhances collaboration through features like branching and merging. Git is a prominent example of a DVCS.

What is Git?

Git: Git is a distributed version control system that enables developers to track changes, manage branches, and collaborate on projects efficiently. Its distributed nature allows each user to have a full copy of the repository, facilitating offline work and robust version tracking. Git has become the de facto standard for version control in modern software development.

What is GitHub?

GitHub: GitHub is a web-based platform built upon Git. It provides a user-friendly interface for repository hosting, along with additional features like issue tracking, code reviews, and project management tools. GitHub enhances collaboration by allowing developers to share repositories, contribute to open-source projects, and integrate with various services to streamline the development workflow.

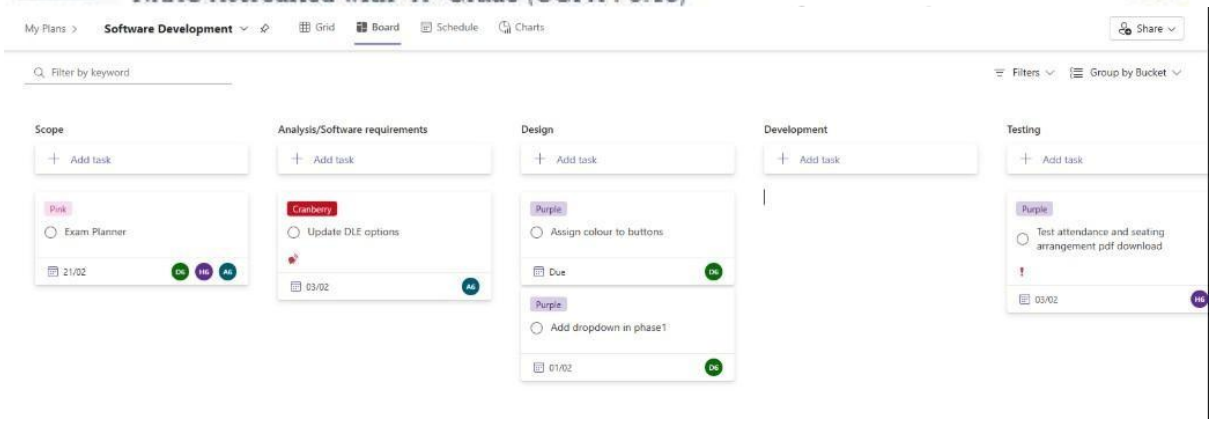
Git Commands(https://git-scm.com/docs-all commands)

Observation:

Kanban Board:

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

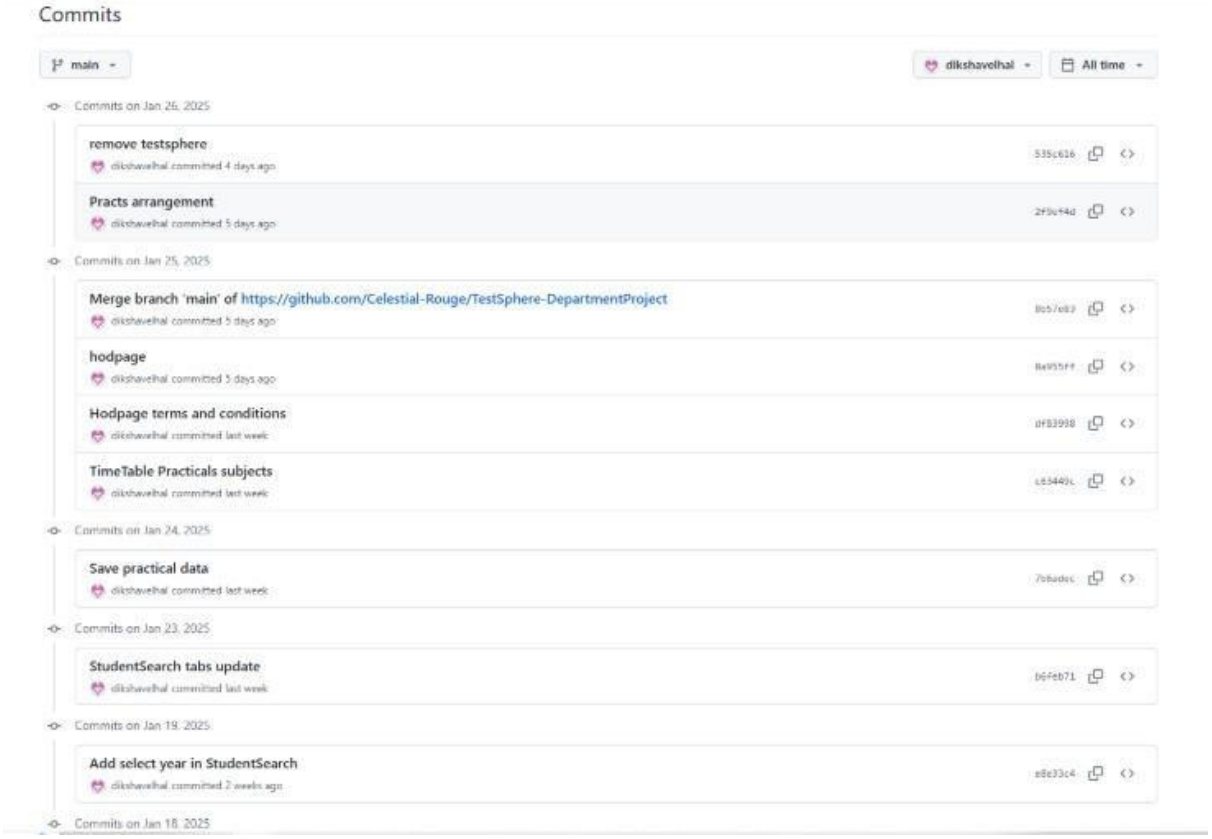


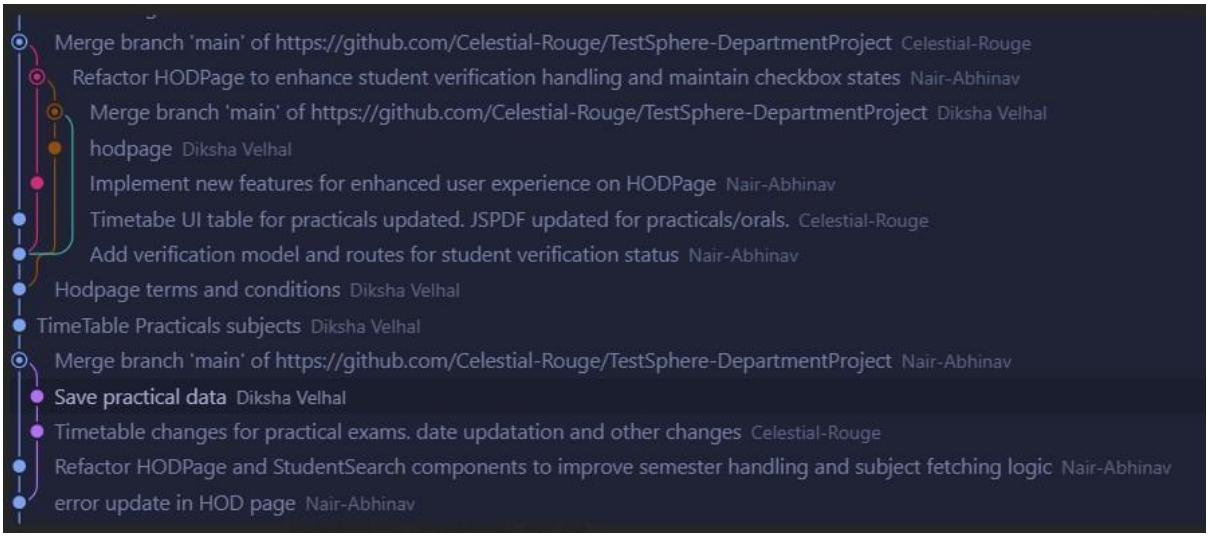
Git Commands:

```
PS C:\Users\djsce.student\Desktop\devops exp\exp1> git clone https://github.com/Nair-Abhinav/restful_apis.git
Cloning into 'restful_apis'...
remote: Enumerating objects: 2159, done.
remote: Compressing objects: 100% (1785/1785), done.
remote: Total 2159 (delta 306), reused 2159 (delta 306), pack-reused 0 (from 0)
Updating files: 100% (1909/1909), done.
fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\djsce.student\Desktop\devops exp\exp1> git checkout main
fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\djsce.student\Desktop\devops exp\exp1> cd .\restful_apis\
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git fetch
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git merge
Already up to date.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git pull
Already up to date.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis>
```

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)





Conclusion:

Academic Year: 2024 - 2025

In this experiment, we understood Version Control System / Source Code Management, install git and create a GitHub account.

References:

1. How to Use Git and GitHub – Version Control Basics for Beginners (freecodecamp.org)
2. Version Control Systems - GeeksforGeeks
3. VCS Program Details - Verra

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 31-01-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma **ROLL: I011** **DIV: IT1-1**

EXPERIMENT NO. 2

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To perform various GIT operations on local and Remote repositories

THEORY:

What is Git?

Git: Git is a distributed version control system that enables developers to track changes, manage branches, and collaborate on projects efficiently. Its distributed nature allows each user to have a full copy of the repository, facilitating offline work and robust version tracking. Git has become the de facto standard for version control in modern software development.

What is GitHub?

GitHub: GitHub is a web-based platform built upon Git. It provides a user-friendly interface for repository hosting, along with additional features like issue tracking, code reviews, and project management tools. GitHub enhances collaboration by allowing developers to share repositories, contribute to open-source projects, and integrate with various services to streamline the development workflow.

Git Commands(https://git-scm.com/docs-all commands)

Git Commands:

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's

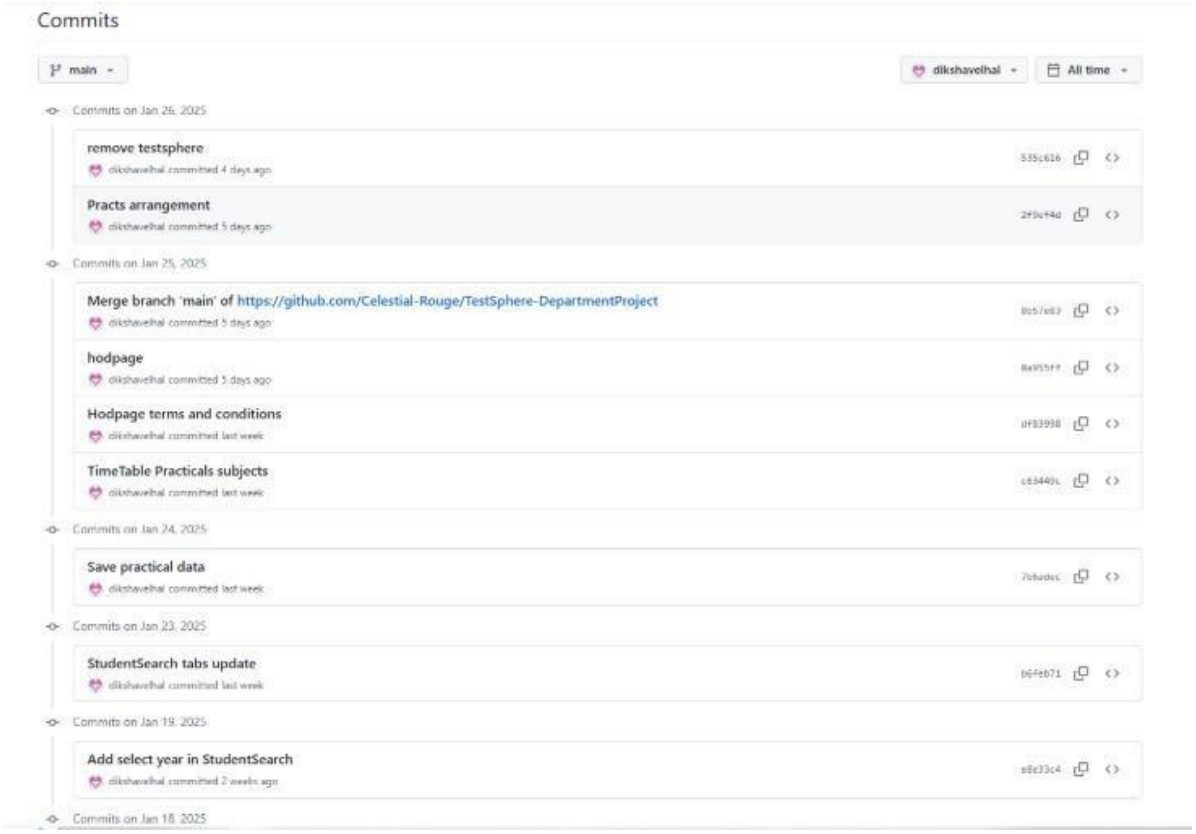
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)

Academic Year: 2024 - 2025

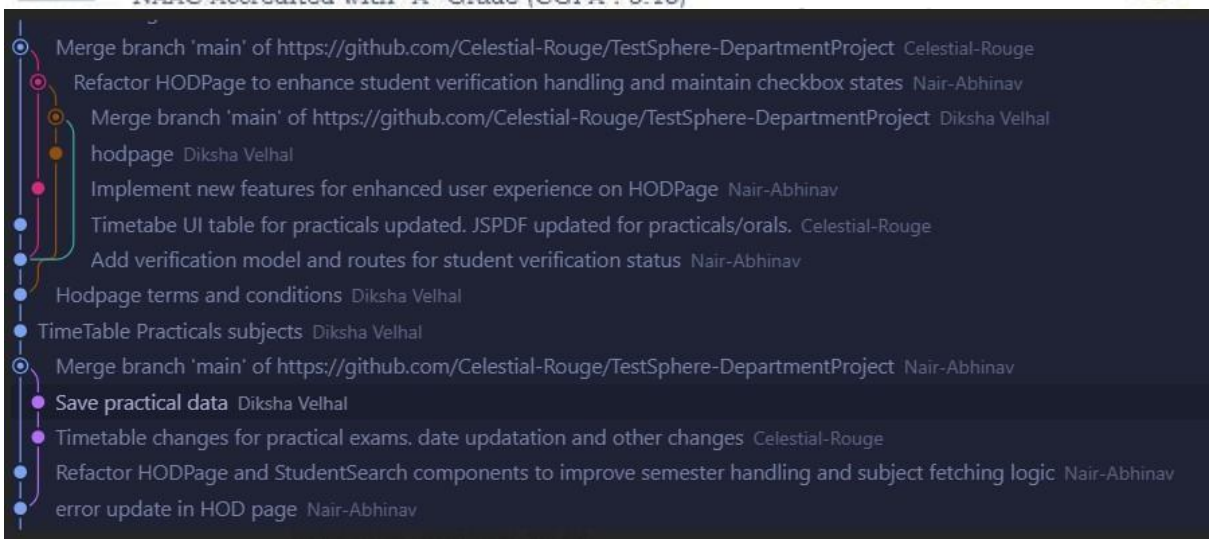
```
PS C:\Users\djsce.student\Desktop\devops exp\exp1> git clone https://github.com/Nair-Abhinav/restful_apis.git
Cloning into 'restful_apis'...
remote: Enumerating objects: 2159, done.
remote: Compressing objects: 100% (1785/1785), done.
remote: Total 2159 (delta 306), reused 2159 (delta 306), pack-reused 0 (from 0)
Updating files: 100% (1909/1909), done.
fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\djsce.student\Desktop\devops exp\exp1> git checkout main
fatal: not a git repository (or any of the parent directories): .git
PS C:\Users\djsce.student\Desktop\devops exp\exp1> cd .\restful_apis\
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git fetch
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git merge
Already up to date.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis> git pull
Already up to date.
PS C:\Users\djsce.student\Desktop\devops exp\exp1\restful_apis>
```



Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)



Academic Year: 2024 - 2025

MINGW64:/c/Users/djsce.student/Desktop/HN/FixIt-Code-Debugging-AI

```
DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git log
commit 3178932081c9415a11ae5cd401878a89994c6645 (HEAD -> dbhr)
Author: DJSCE Student <DJSCE.Student@SVKMGRP.COM>
Date:   Fri Jan 31 09:07:44 2025 +0530

    Tryitout

commit e5e2ac94f16ab6976cba07cc1baa09ba6336aa3d (origin/main, origin/HEAD, main)
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:25:41 2024 +0530

    second commit

commit 7d656b3e8f0774f23928eded6840f12289049bd8
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:20:58 2024 +0530

    first commit

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git push
fatal: The current branch dbhr has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin dbhr


DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git push --set-upstream origin dbhr
fatal: helper error (-1): authentication prompt was canceled
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexec/git-core/git-gui--
Username for 'https://github.com':
DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git fetch

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git checkout -b main
fatal: A branch named 'main' already exists.

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git feth
git: 'feth' is not a git command. See 'git --help'.

The most similar command is
        fetch

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git fetch

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git merger
git: 'merger' is not a git command. See 'git --help'.

The most similar command is
        merge

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git merge
Already up to date.

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git pull
```

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

MINGW64:/c/Users/djsce.student/Desktop/HN/FixIt-Code-Debugging-AI

```
    -C                    copy a branch, even if target exists
    -l, --list            list branch names
    --show-current        show current branch name
    --create-reflog       create the branch's reflog
    --edit-description    edit the description for the branch
    -f, --force           force creation, move/rename, deletion
    --merged <commit>     print only branches that are merged
    --no-merged <commit>  print only branches that are not merged
    --column[=<style>]    list branches in columns
    --sort <key>          field name to sort on
    --points-at <object>  print only branches of the object
    -i, --ignore-case     sorting and filtering are case insensitive
    --format <format>     format to use for the output


DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git branch dbhr

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git checkout dbhr
Switched to branch 'dbhr'

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git status
On branch dbhr
nothing to commit, working tree clean

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git add .

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git status
On branch dbhr
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   GIThelp/hello.txt


DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
$ git commit -m "Tryitout"
[dbhr 3178932] Tryitout
 Committer: DJSCE Student <DJSCE.Student@SVKMGRP.COM>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 2 insertions(+)
 create mode 100644 GIThelp/hello.txt

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (dbhr)
```

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

MINGW64:/c/Users/djsce.student/Desktop/HN/FixIt-Code-Debugging-AI

```
Date:   Fri Sep 6 01:25:41 2024 +0530

    second commit

commit 7d656b3e8f0774f23928eded6840f12289049bd8
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:20:58 2024 +0530

    first commit

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git config list
error: key does not contain a section: list

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git branch -b dbhr

error: unknown switch `b'
usage: git branch [<options>] [-r | -a] [--merged] [--no-merged]
   or: git branch [<options>] [-l] [-f] <branch-name> [<start-point>]
   or: git branch [<options>] [-r] (-d | -D) <branch-name>...
   or: git branch [<options>] (-m | -M) [<old-branch>] <new-branch>
   or: git branch [<options>] (-c | -C) [<old-branch>] <new-branch>
   or: git branch [<options>] [-r | -a] [--points-at]
   or: git branch [<options>] [-r | -a] [--format]

Generic options
    -v, --verbose         show hash and subject, give twice for upstream branch
    -q, --quiet           suppress informational messages
    -t, --track           set up tracking mode (see git-pull(1))
    -u, --set-upstream-to <upstream>
                          change the upstream info
    --unset-upstream      unset the upstream info
    --color[=<when>]      use colored output
    -r, --remotes         act on remote-tracking branches
    --contains <commit>   print only branches that contain the commit
    --no-contains <commit>
                          print only branches that don't contain the commit
    --abbrev[=<n>]        use <n> digits to display object names

Specific git-branch actions:
    -a, --all             list both remote-tracking and local branches
    -d, --delete          delete fully merged branch
    -D                    delete branch (even if not merged)
    -m, --move            move/rename a branch and its reflog
    -M                    move/rename a branch, even if target exists
    -c, --copy            copy a branch and its reflog
    -C                    copy a branch, even if target exists
    -l, --list            list branch names
    --show-current        show current branch name
    --create-reflog       create the branch's reflog
    --edit-description    edit the description for the branch
    -f, --force           force creation, move/rename, deletion
    --merged <commit>     print only branches that are merged
    --no-merged <commit>  print only branches that are not merged
    --column[=<style>]    list branches in columns
    --sort <key>          field name to sort on
    --points-at <object>  print only branches of the object
    -i, --ignore-case     sorting and filtering are case insensitive
    --format <format>     format to use for the output
```

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

MINGW64:/c/Users/djsce.student/Desktop/HN/FixIt-Code-Debugging-AI

```
$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git log
commit e5e2ac94f16ab6976cba07cc1baa09ba6336aa3d (HEAD -> main, origin/main, origin/HEAD)
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:25:41 2024 +0530

    second commit

commit 7d656b3e8f0774f23928eded6840f12289049bd8
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:20:58 2024 +0530

    first commit

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git config list
error: key does not contain a section: list

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git status
On branch main
```

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

MINGW64:/c/Users/djsce.student/Desktop/HN/FixIt-Code-Debugging-AI

```
DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git --helf
unknown option: --helf
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git --help
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
   grep              Print lines matching a pattern
   log               Show commit logs
   show              Show various types of objects
   status            Show the working tree status

grow, mark and tweak your common history
   branch            List, create, or delete branches
   commit            Record changes to the repository
   merge             Join two or more development histories together
   rebase            Reapply commits on top of another base tip
   reset             Reset current HEAD to the specified state
   switch            Switch branches
   tag               Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch             Download objects and refs from another repository
   pull              Fetch from and integrate with another repository or a local branch
   push              Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

DJSCE.Student@MUM0922CPU0316 MINGW64 ~/Desktop/HN/FixIt-Code-Debugging-AI (main)
$ git log
commit e5e2ac94f16ab6976cba07cc1baa09ba6336aa3d (HEAD -> main, origin/main, origin/HEAD)
Author: Dikshant <dikshantbadawadagi@gmail.com>
Date:   Fri Sep 6 01:25:41 2024 +0530

    second commit

commit 7d656b3e8f0774f23928eded6840f12289049bd8
Author: Dikshant <dikshantbadawadagi@gmail.com>
```

Conclusion:

In this experiment, we understood Version Control System / Source Code Management, install git and create a GitHub account.

References:

1. How to Use Git and GitHub – Version Control Basics for Beginners (freecodecamp.org)
2. Version Control Systems - GeeksforGeeks
3. VCS Program Details - Verra

Academic Year: 2024-25

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 06-03-2025**

COURSE NAME: DevOps Laboratory **CLASS:TYBTech**
NAME: Anish Sharma ROLL: I011 DIV: IT1-1 EXPERIMENT NO. 3

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To install Jenkins and perform Java and Python Programs.

THEORY:

Installing Jenkins and Running Java & Python Programs

1. Install Jenkins

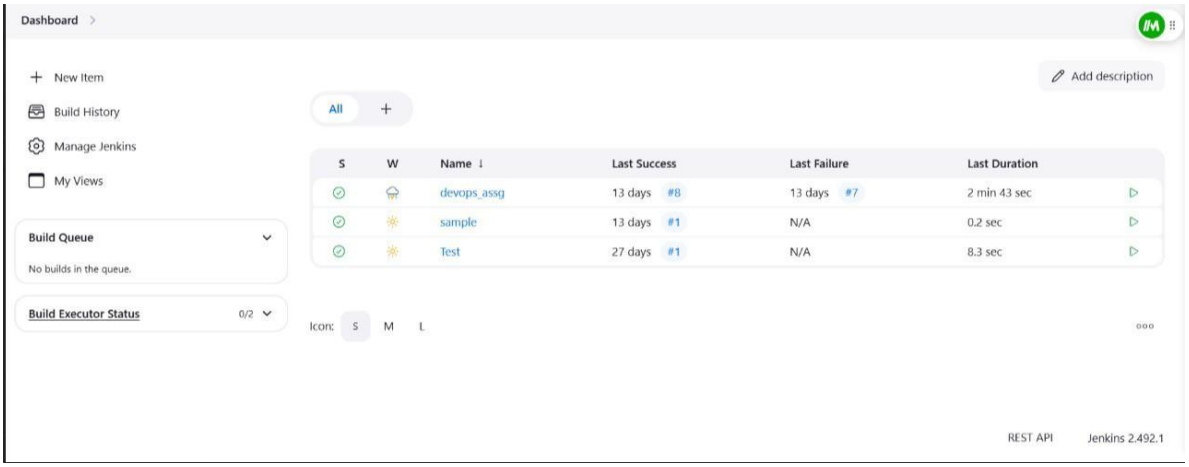
- **Prerequisite:** Install Java (JDK).
  - Windows/Linux/macOS: Install OpenJDK (sudo apt install openjdk-11-jdk for Ubuntu). • **Install Jenkins:**
  - Windows: Download from Jenkins official site.

2. Running Java & Python Programs in Jenkins

- **For Java:** Add a Jenkins job with javac Program.java && java Program
- **For Python:** Use python3 script.py in Jenkins job configuration. Jenkins automates execution, making CI/CD easier.

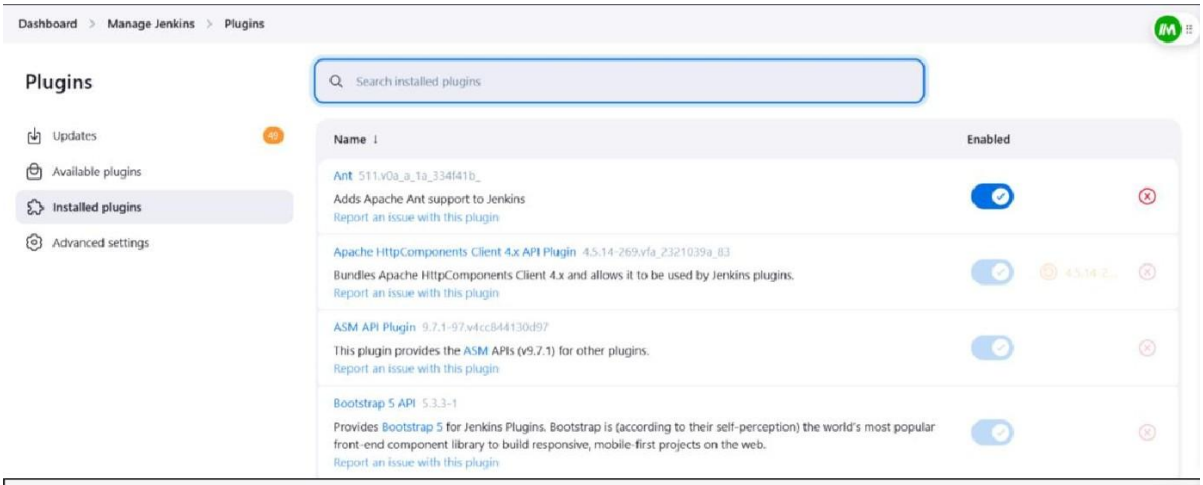
OUTPUT:

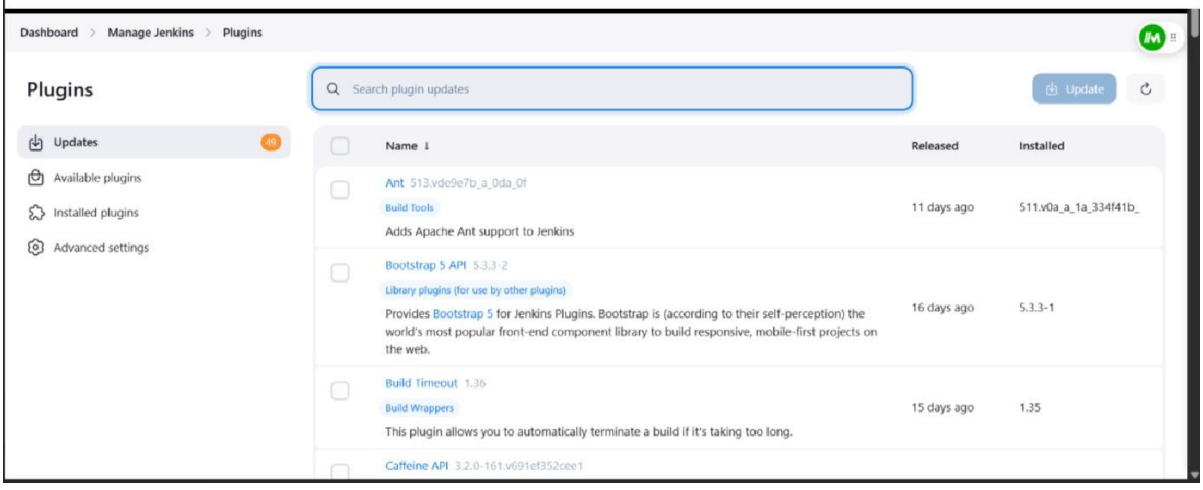
Academic Year: 2024-25



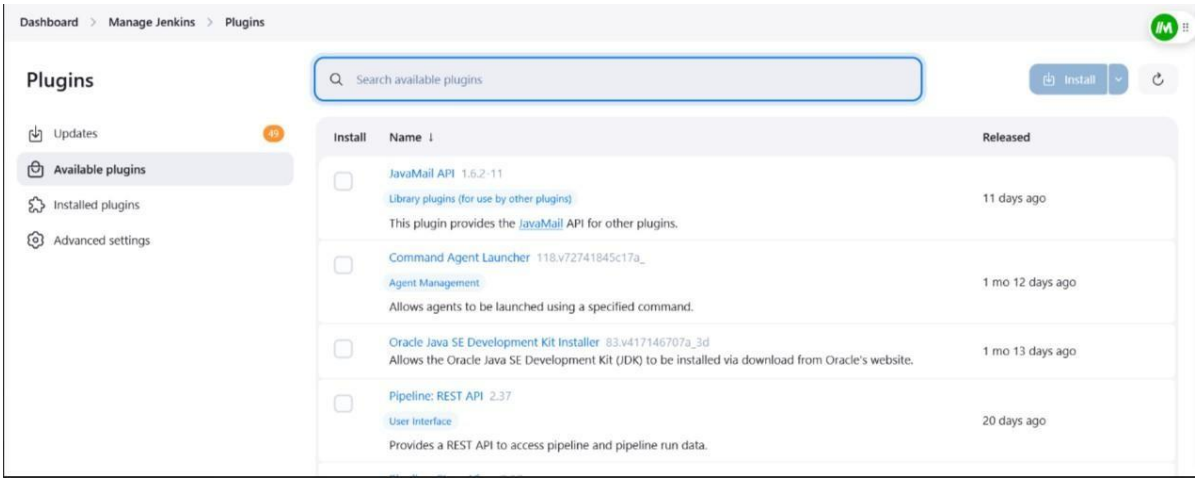
Academic Year: 2024-25

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)





Academic Year: 2024-25



CONCLUSION:

Jenkins is a powerful automation tool that simplifies Continuous Integration and Deployment (CI/CD). By installing Jenkins and configuring it to run Java and Python programs, developers can automate builds, testing, and execution efficiently. This enhances productivity, reduces manual effort, and ensures consistency in software development workflows.

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 31-01-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma ROLL: I011 DIV: IT1-1 EXPERIMENT NO. 4

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To implement the pipeline of jobs using Maven in Jenkins, create a pipeline script to Test and deploy an application.

THEORY:

Implementing a Continuous Integration/Continuous Deployment (CI/CD) pipeline in Jenkins for a Maven-based Java application involves several key steps:

1. **Setting Up Jenkins and Required Tools:**
    - **Jenkins Installation: Ensure Jenkins is installed and running. ○ Maven Integration: Configure Maven in Jenkins by navigating to "Manage Jenkins" > "Global Tool Configuration" and adding a Maven installation.**
    - **Version Control: Integrate your source code repository (e.g., GitHub) with Jenkins.**
2. **Creating the Jenkins Pipeline:**
    - **Pipeline Script (Jenkinsfile): Define your build, test, and deployment stages in a Jenkinsfile stored in your repository.**
    - **Declarative Pipeline Syntax: Utilize Jenkins' declarative pipeline syntax for clarity and maintainability.**
3. **Defining Pipeline Stages:**
    - **Build Stage: Compile the code and package it using Maven.**
    - **Test Stage: Execute unit tests to ensure code quality.**
    - **Deploy Stage: Deploy the application to the desired environment, such as a web server like Apache Tomcat.**

4. **Sample Jenkinsfile: Below is an example of a declarative Jenkins Pipeline script for a Maven project:**

```
groovy
CopyEdit
pipeline {
agent any
tools {
    maven 'Maven' // Assumes 'Maven' is configured in Global Tool Configuration
  } stages
  {
    stage('Checkout') { steps { git
      'https://github.com/your-repo/your-project.git'
      }
    }
    stage('Build') { steps
      {
        sh 'mvn clean package -DskipTests'
      }
    } stage('Test') {
    steps { sh 'mvn
    test'
      } post
      {
        always { junit 'target/surefire-
          reports/*.xml'
```

```
            }
        }
    }
    stage('Deploy') { steps
        {
            // Deployment steps, e.g., copying files to a server sh 'scp
            target/your-app.war user@server:/path/to/deploy/'
        }
    }
  } post {
  cleanup {
        cleanWs()
    }
  }
}
```

Explanation:

- **tools: Specifies the Maven installation to use.**
- **stages: Defines the sequence of stages: Checkout, Build, Test, and Deploy.**
- **post: Contains actions to perform after each stage or the entire pipeline, such as archiving test results or cleaning up the workspace.**

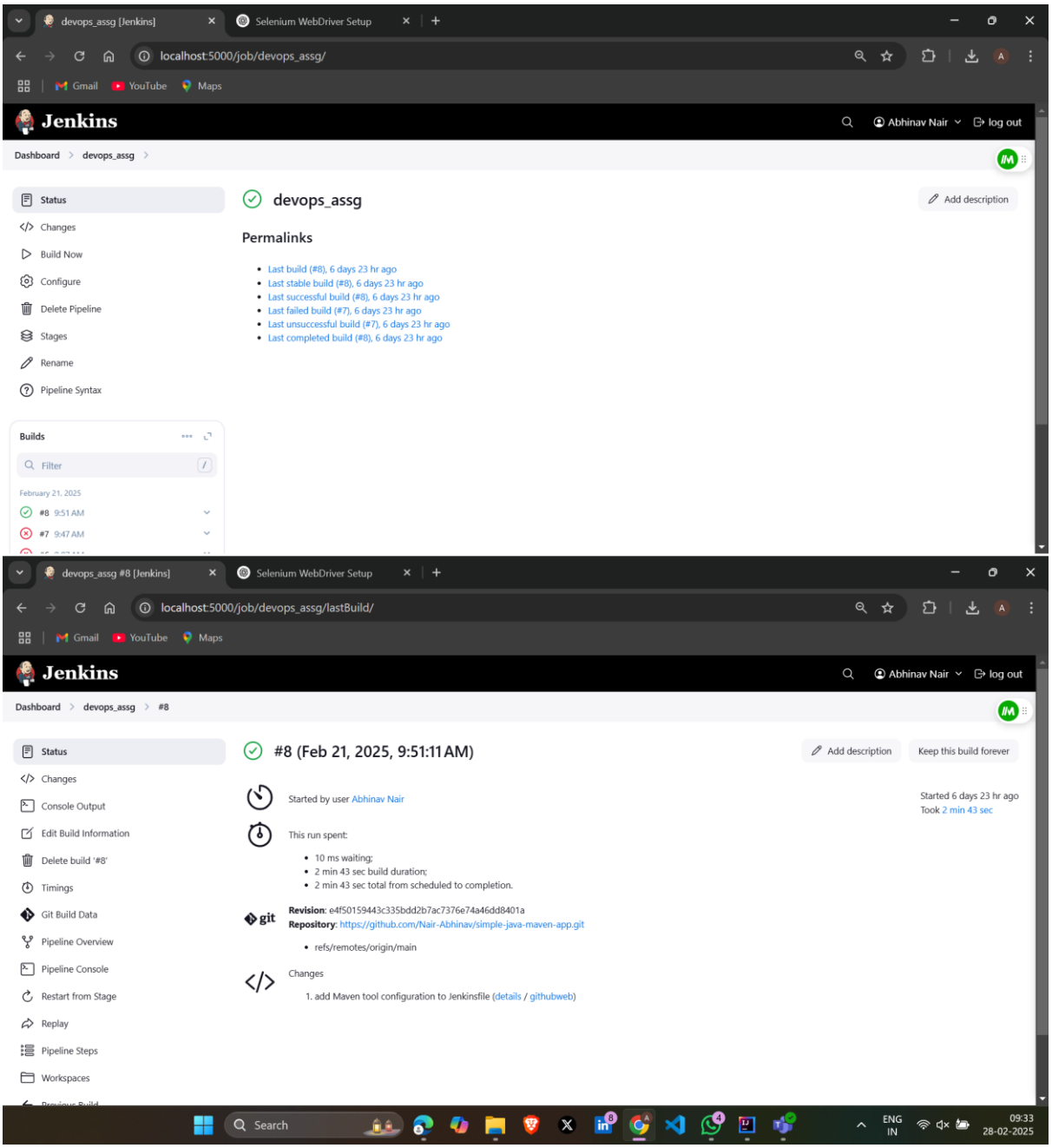
5. Enhancing the Pipeline:

- **Parallel Testing: Run tests in parallel to reduce execution time.**
- **Environment-Specific Deployments: Use parameters to deploy to different environments (development, staging, production).**
- **Notifications: Integrate with communication tools (e.g., email, Slack) to send build and deployment notifications.**

Academic Year: 2024 - 2025

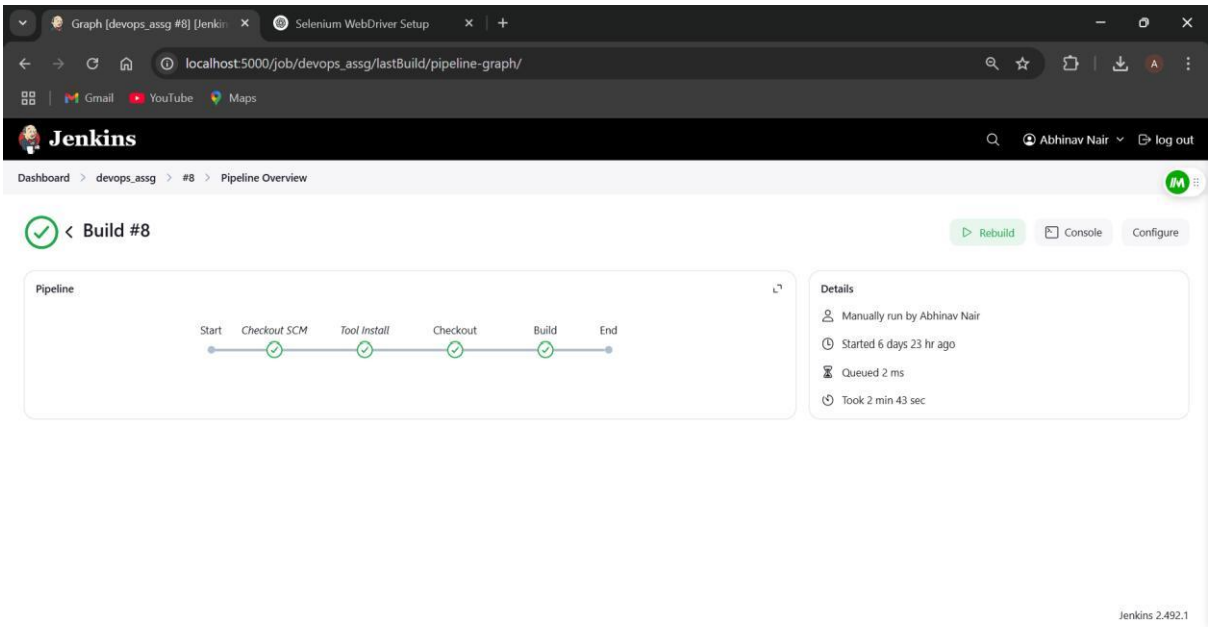
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

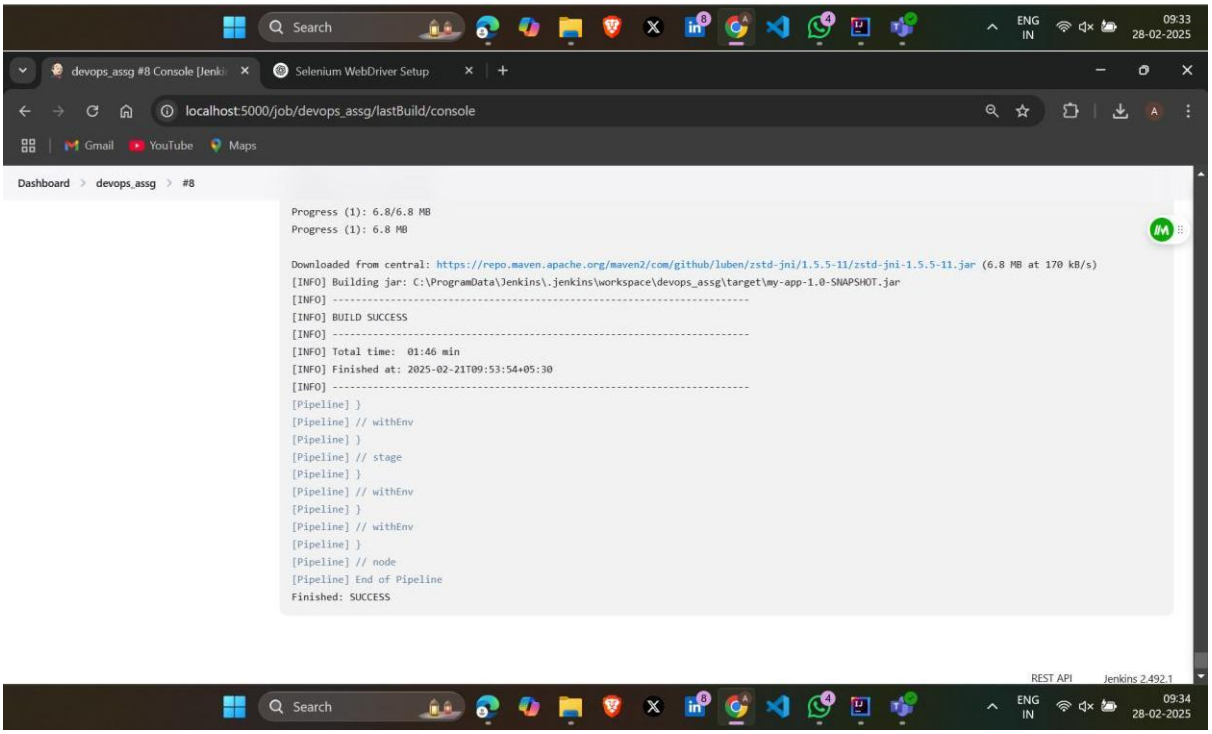
OUTPUT:



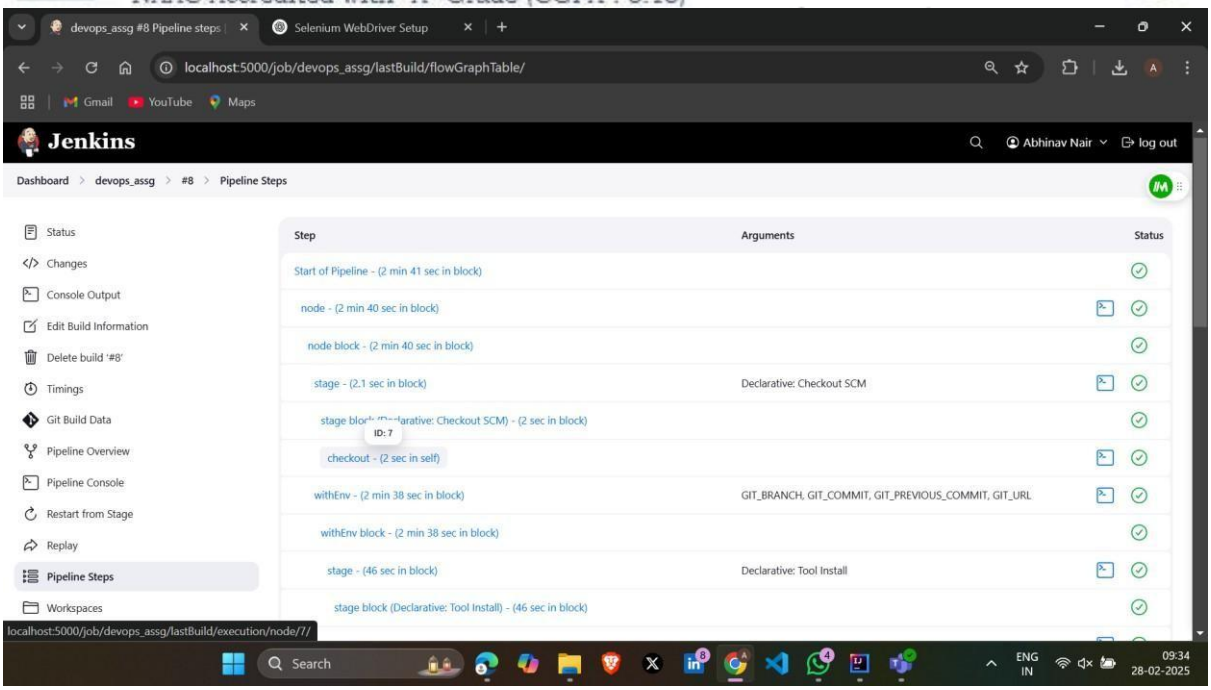
Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)





Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)



Conclusion:

In this experiment, we implemented the pipeline of jobs using Maven in Jenkins, create a pipeline script to Test and deploy an application.

References:

1. How to Use Git and GitHub – Version Control Basics for Beginners (freecodecamp.org)
2. Version Control Systems - GeeksforGeeks
3. VCS Program Details - Verra

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITHN1L1 **DATE: 5/5/2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

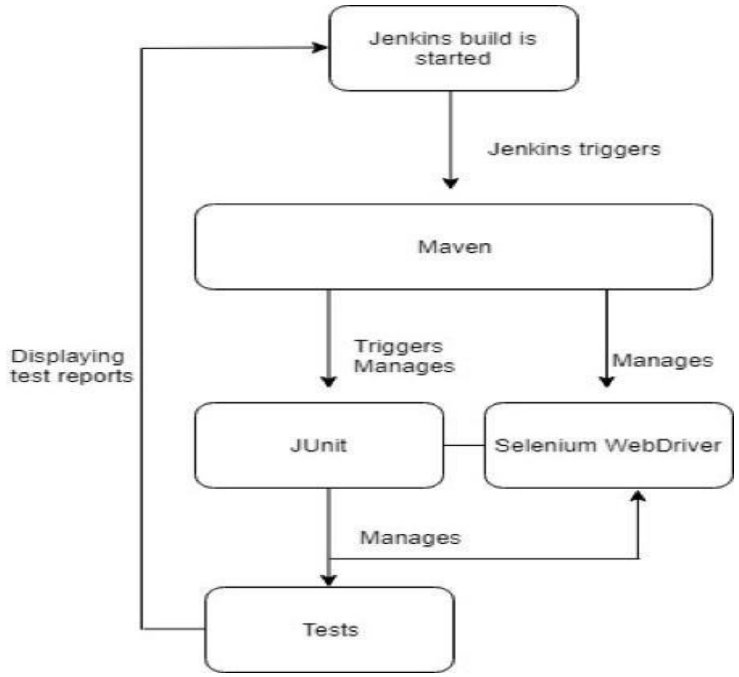
NAME: Anish Sharma DIV: IT1-1 ROLL: I004 EXPERIMENT NO. 5

CO/LO: Apply DevOps principles to meet software development requirements.

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

THEORY:

Let us have a look at the Architecture below diagram depicts the same.



Maven – Set up

Step 1: To setup Maven, download the maven's latest version from Apache depending upon different OS.

Step 2: Unzip the folder and save it on the local disk.

Step 3: Create environment variable for MAVEN_HOME. Follow the below step:

Navigate to System Properties ->Advanced System Setting->Environment Variable ->System Variable ->New ->Add path of Maven folder

Step 4: Edit path variable and provide the bin folder path.

Step 5: Now verify the maven installation using command prompt and don't forget to setup JAVA_HOME
Use mvn –version to verify maven version in the command prompt window.

Maven can be integrated with Eclipse IDE by adding the m2eclipse plugin to Eclipse to facilitate the build process and create pom.xml file.

Adding m2eclipse plugin to Eclipse with following steps
Step 1) In Eclipse IDE, select Help | Install New Software from Eclipse Main Menu. Step 2) On the Install dialog, select Work with and m2e plugin as shown in the following screenshot and select Next and finish installation.

How to configure Eclipse with Maven

With m2e plugin is installed, we now need create Maven project.

Step 1) In Eclipse IDE, create a new project by selecting File | New | Other from Eclipse menu.
Step 2) On the New dialog, select Maven | Maven Project and click Next.

Step 3) On the New Maven Project dialog select the Create a simple project and click next.

Step 4) Enter WebdriverTest in Group Id: and Artifact Id: and click finish.

Step 5) Eclipse will create WebdriverTest Project with following structure.

Executing the Maven Build

Project can be built by both using IDE and command prompt

Using IDE, right click on the POM-Run as Maven build

Same can be done using command prompt. Navigate to project folder where pom.xml lies

Use the below commands to clean, compile and install

Clean: mvn clean Compile: mvn compile
Install: mvn install

Integration of Maven with Jenkins

Jenkins is an open-source automation tool and allows us to continuously deliver the software and continuously making it easier to integrate the changes to the project and obtain a fresh build.

Below are the steps to set up Jenkins

Step 1: Download Jenkins war file from https://Jenkins.io/
Step 2: Place the war file into any location on our system

Step 3: Go to command prompt (windows) and terminal(mac)

- Go to folder where Jenkins.war is
- java – jar jenkins.war

Step 4: Go to browser http://localhost:8080 and Jenkins dashboard should show up
To start Jenkins (standalone) on a different port, use the below command

java –jar Jenkins. War– httpPort=9090

Configure Maven with Jenkins

Maven can be integrated with Jenkins, and this can be achieved by installing the maven2 project plugin for the tool.

1. Navigate to http://localhost:8080
2. Click Manage Jenkins
3. Click Configure System
4. In the Maven installation section-> click Add Maven button ->uncheck install automatically checkbox ->enter Maven_Home->Apply->Save
5. Execute Maven project using Jenkins
6. Go to Jenkins dashboard
7. Click on New item

1. Go to Build Environment->Enter the POM.xml file path of Maven->Enter Goal and options: clean install. 2. Click Apply->Save

Now Maven project is created on Jenkins as below

To execute the Maven project, click on the Build Now button and it will invoke the pom.xml file.

Right click on the build number and click on console output to see the result.

Key Advantages

Below are few advantages of Selenium Maven Jenkins Integration.

- Whenever a change is made in the implementation, the changes are deployed on the test environment and developers are kept informed about the build and test stage results.

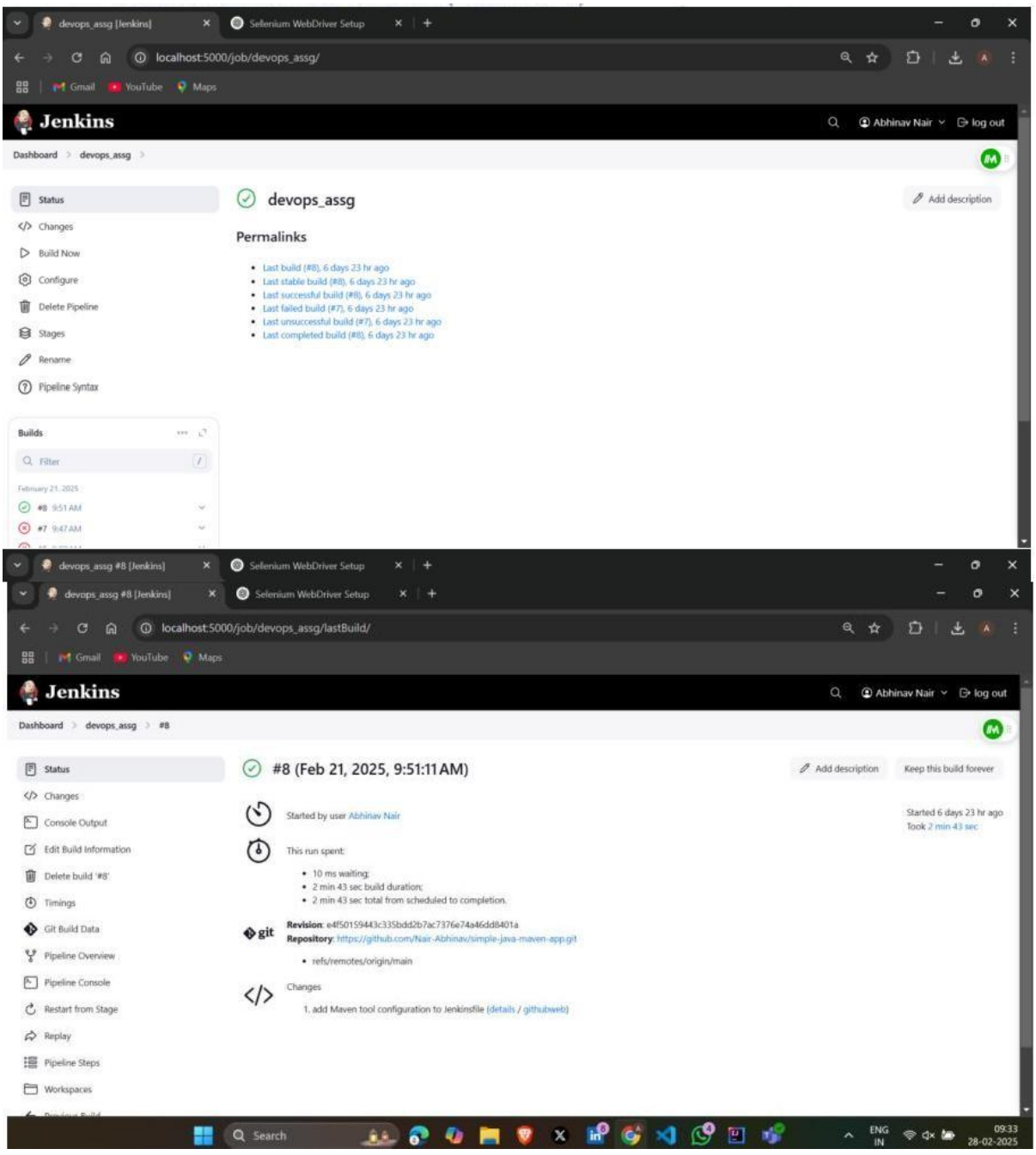
- Maven reduces the overall dependency of the manual download of jar files. Also, the 'Maven Project' type in Jenkins helps in getting started with Selenium Maven.
- The Selenium Maven Jenkins integration is best suited for developing and testing teams distributed across different geographies.

To conclude Maven and Jenkins with Selenium helps to accelerate with minimal required to integrate all the changes and automate the build process for continuous integrated project.

Output:

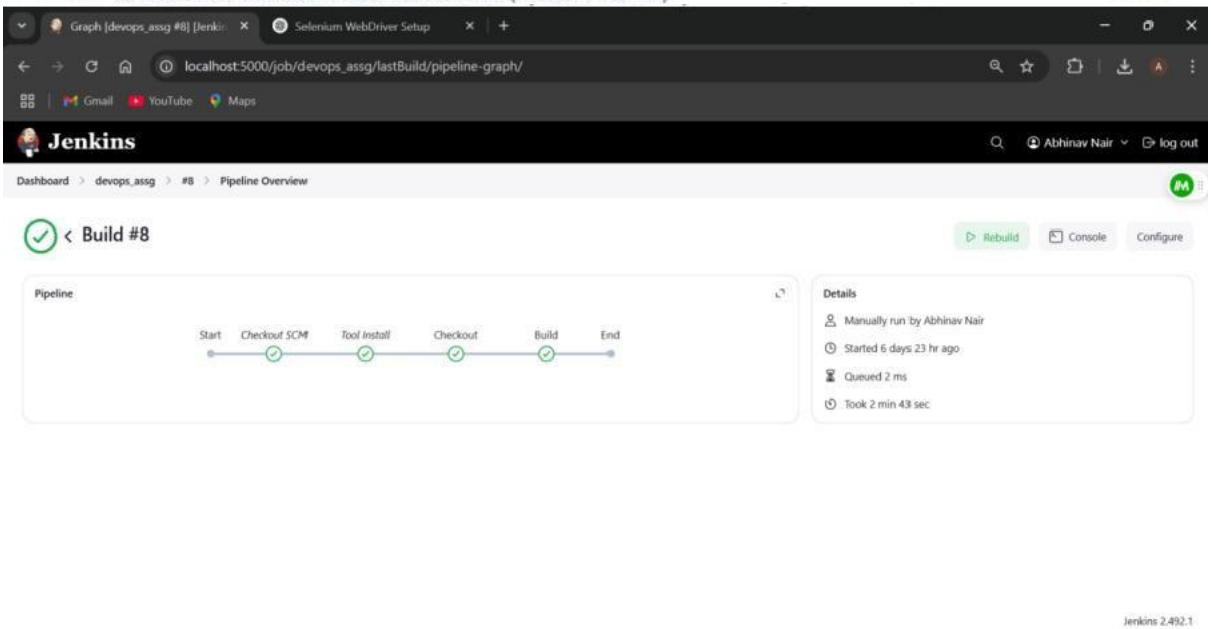
Academic Year: 2024 - 2025

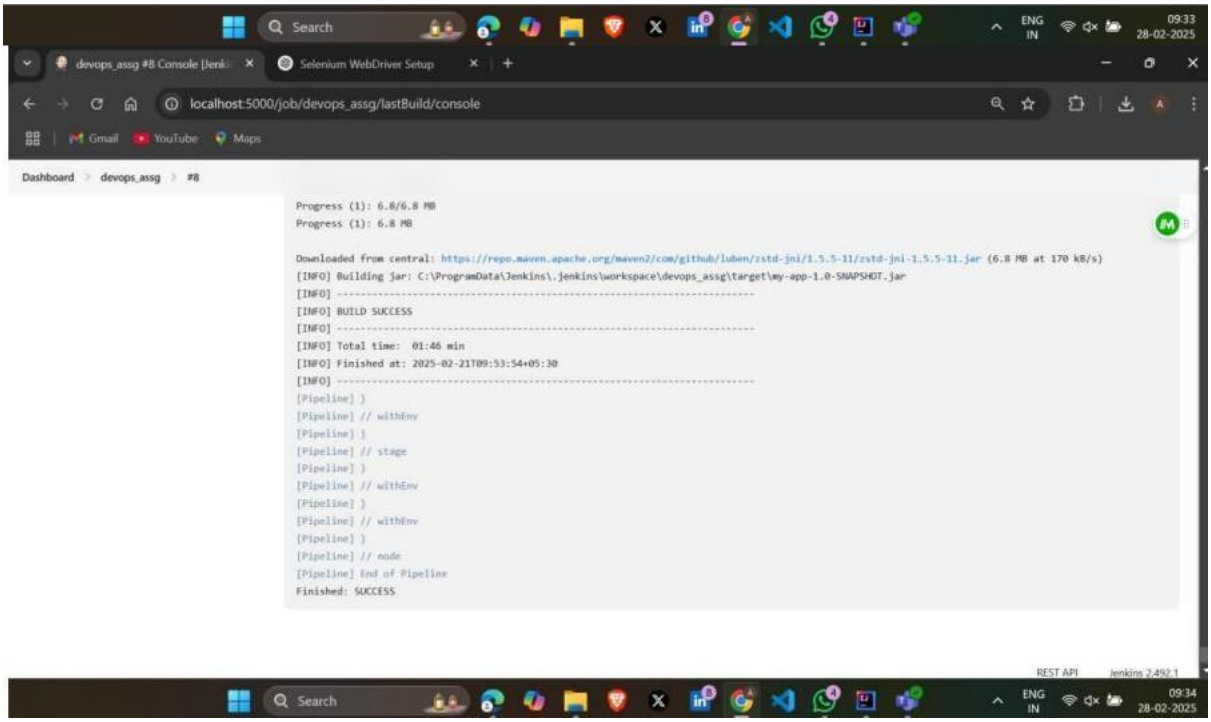
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)



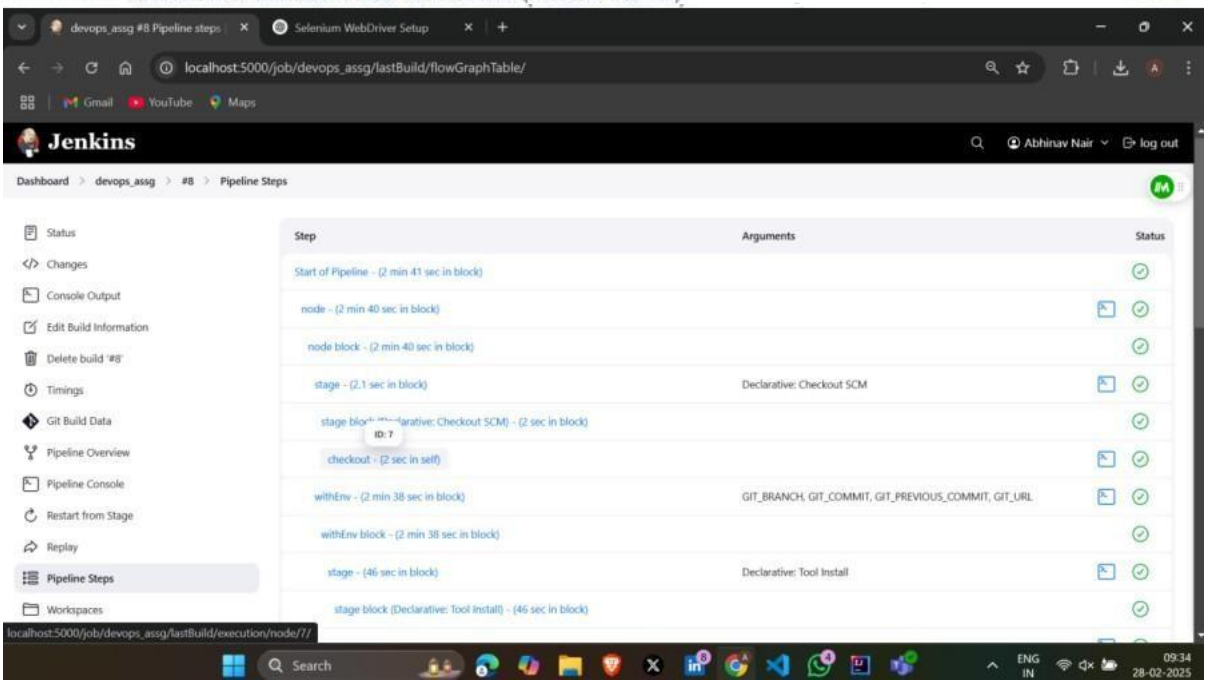
Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)





Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)



Conclusion:

In this experiment, we set and Run Selenium Tests in Jenkins Using Maven.

References:

1. https://www.blazemeter.com/blog/jenkins-selenium
2. https://ww7.q-automations.com/2019/06/01/how-run-selenium-tests-in-jenkins-using-maven/?usid=24&utid=10871809163
3. https://www.winwire.com/blog/how-to-integrate-maven-and-jenkins-with-selenium/
4. https://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html

Academic Year:2024-25

A

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Information Technology

(Academic Year 2024-25)

COURSE CODE: DJS22ITHN1L1 **DATE:11-04-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma

EXP: 06

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

Theory:

Introduction to docker:

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

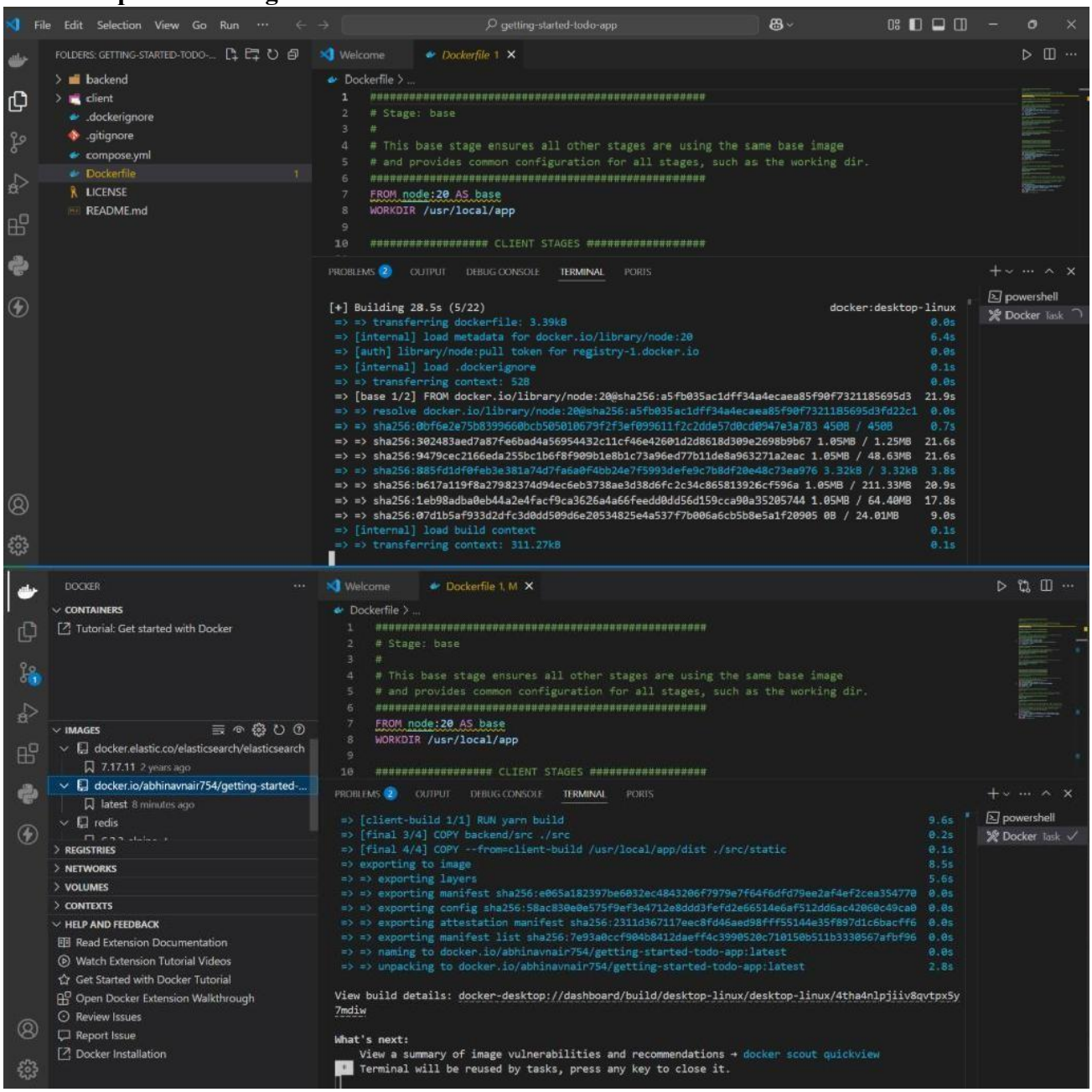
Why is Docker used?

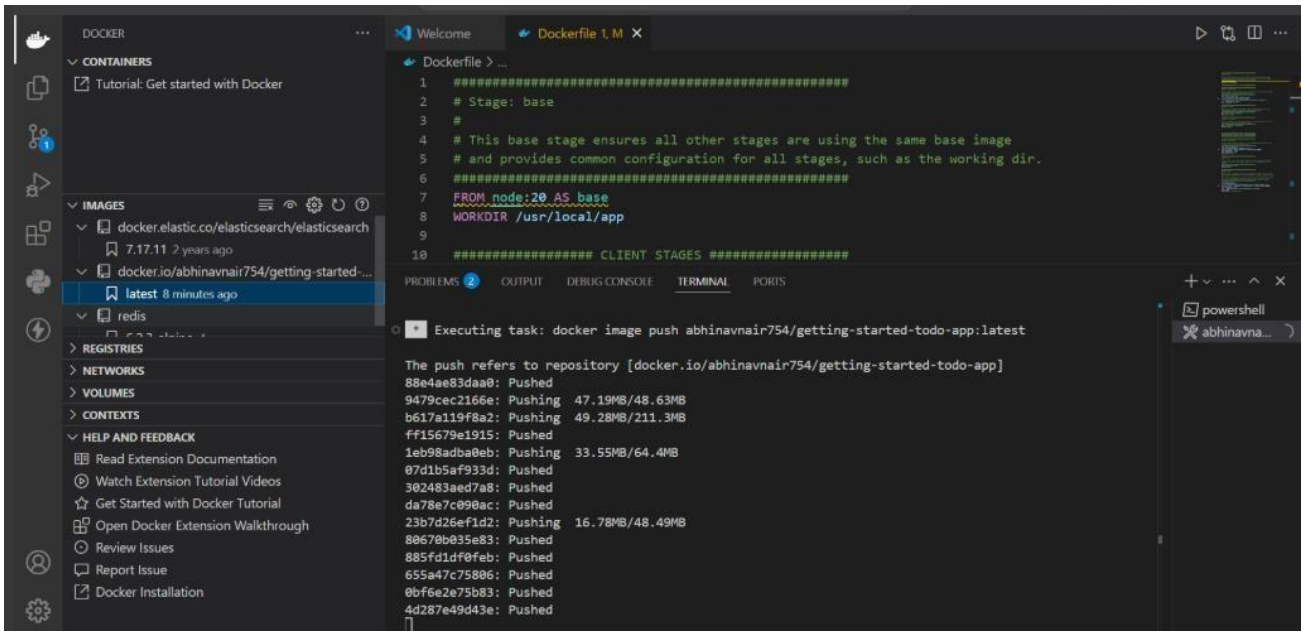
Docker is a basic tool, like git or java, that you should start incorporating into your daily development and ops practices. o Use Docker as version control system for your entire app's operating system o Use Docker when you want to distribute/collaborate on your app's operating system with a team o Use Docker to run your code on your laptop in the same environment as you have on your server (try the building tool) o Use Docker whenever your app needs to go through multiple phases of development (dev/test/qa/prod, try Drone or Shippable, both do Docker CI/CD) o Use Docker with your Chef Cookbooks and Puppet Manifests (remember, Docker doesn't do configuration management) What role docker plays in dev-ops?

Docker is just another tool available to DevOps Engineers, DevOps practitioners, or whatever you want to call them. What Docker does is it encapsulates code and code dependencies in a single unit (a container) that can be run anywhere where the Docker engine is installed. Why is this useful? For multiple reasons; but in terms of CI/CD it can help Engineers separate Configuration from Code, decrease the amount of time spent doing dependency management etc., can use it to scale (with the help of some other tools of course). The list goes on. For example: If I had a single code repository, in my build script I could pull in environment specific dependencies to create a Container that functionally behaves the same in each environment, as I'm building from the same source repository, but it can contain a set of environment specific certificates and configuration files etc. Having said all of that, there really is a great deal you can do to utilize Docker in your CI/CD Pipelines. I think an understanding of what Docker is, and what Docker can do is more

important that a "how to use Docker in your CI/CD" guide. While there are some common patterns out there, it all comes down to the problem(s) you are trying to solve, and certain patterns may not apply to a certain use case.

Build and push the image:





CONCLUSION: In this experiment we build and pushed the image on docker

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY COURSE

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma

EXPERIMENT NO. 7

CO/LO: Apply DevOps principles to meet software development requirements.

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

THEORY:

Docker is a powerful tool that allows developers to containerize applications, making them portable, scalable, and consistent across different environments. One of the core components of Docker is the Dockerfile — a script used to define the steps to build a Docker image. This document will cover the theory of Dockerfile instructions and explain how to build a Docker image for a sample web application.

Dockerfile: A Dockerfile is a text file that contains a series of instructions that Docker uses to build an image. The image is a blueprint for creating containers, which are lightweight, isolated environments where your application will run.

A Dockerfile includes:

- Base images (e.g., operating systems, programming runtimes)
- Application dependencies (e.g., libraries, frameworks)
- Instructions on how to set up the application and environment inside the container
- Exposed ports, volumes, and environment variables needed to run the application

Once the Dockerfile is written, you can use it to build a Docker image using the docker build command.

Dockerfile Instructions:

- **FROM**: Specifies the base image to use. Here, python:3.9-slim is a slim Python image.
- **WORKDIR**: Sets the working directory in the container to /app.
- **COPY**: Copies the contents of your local directory to /app inside the container.
- **RUN**: Installs Python dependencies using pip.

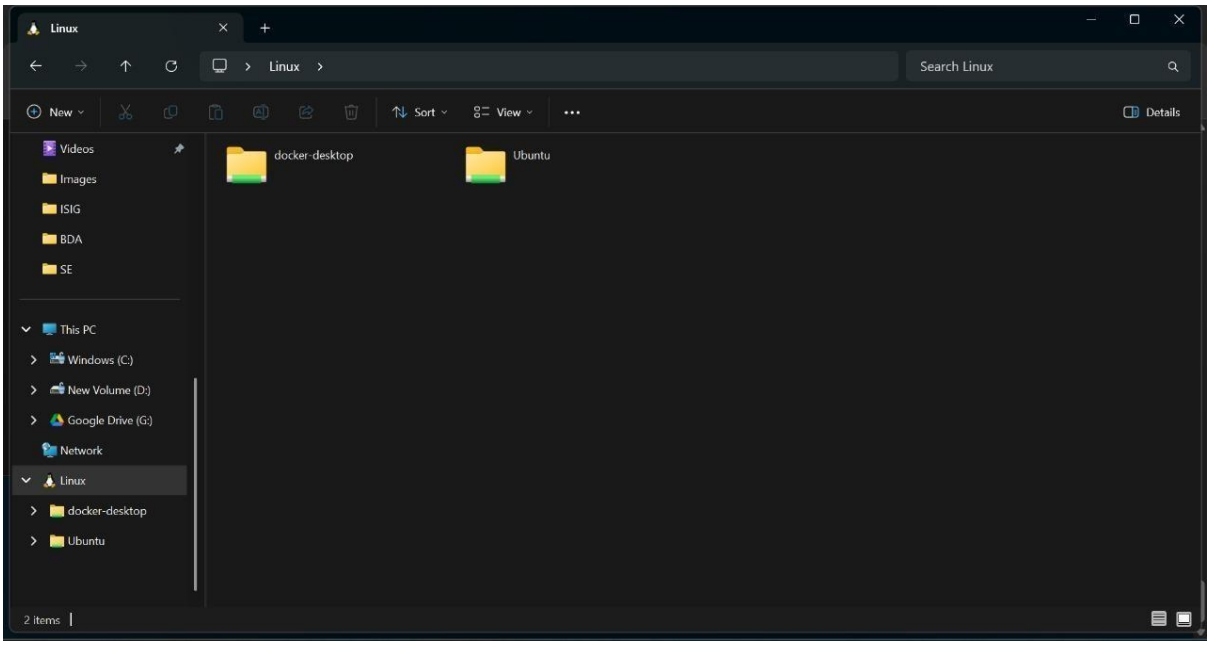
- **EXPOSE**: Informs Docker that the container listens on port 5000.
- **ENV**: Sets environment variables for Flask to work correctly inside the container.
- **CMD**: Specifies the command to run when the container starts. In this case, it starts the Flask application.

Implementation:

Create a Dockerfile to containerize a simple Flask web application. By following these steps, you:

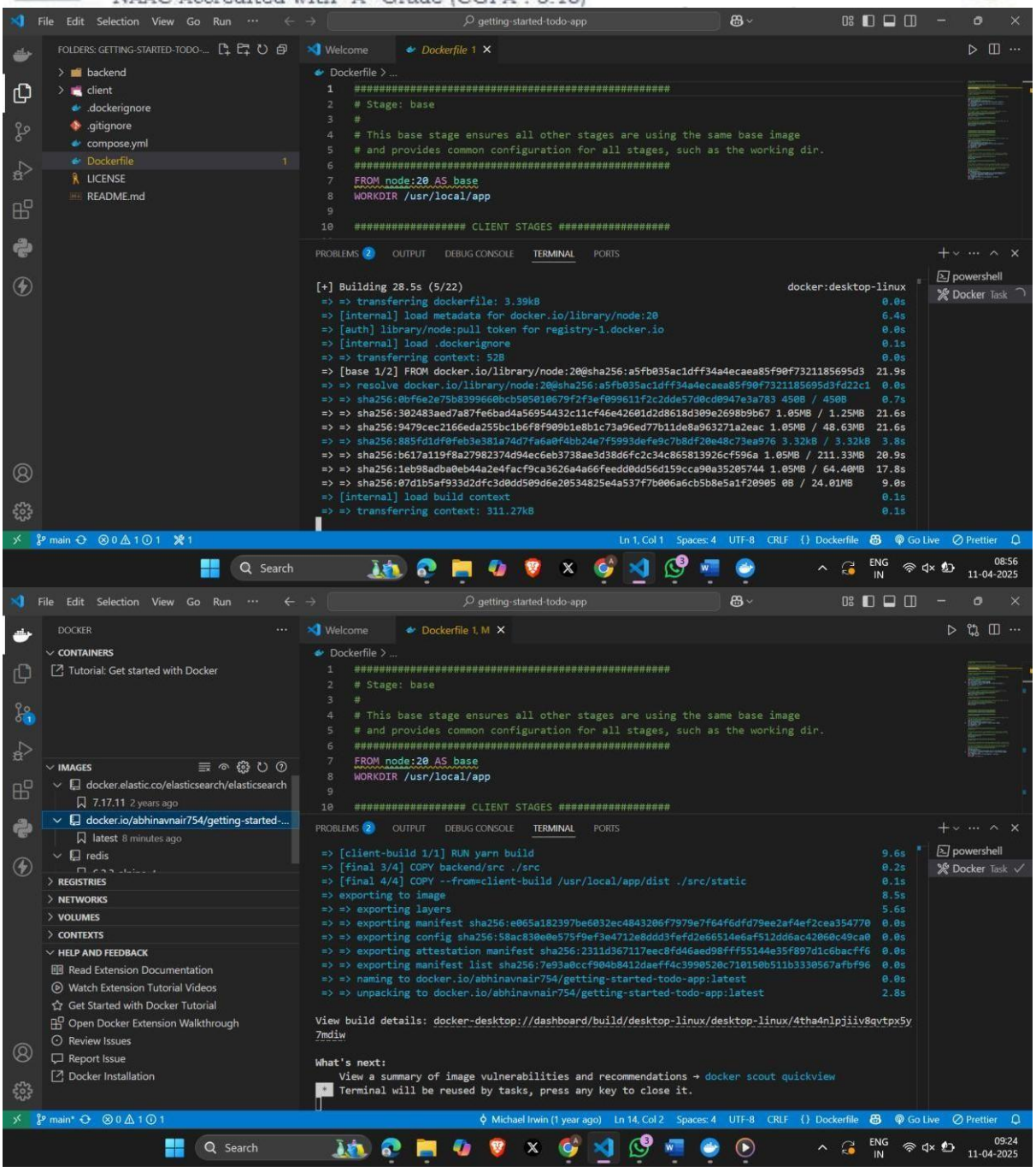
1. Created a Flask app with minimal code.
2. Defined dependencies using requirements.txt.
3. Created a Dockerfile that:
    - Installs Python dependencies. ○ Exposes the port Flask listens to.
    - Runs the Flask app inside the container.
4. Built and ran the Docker image, then accessed the web app in a browser.

Output:



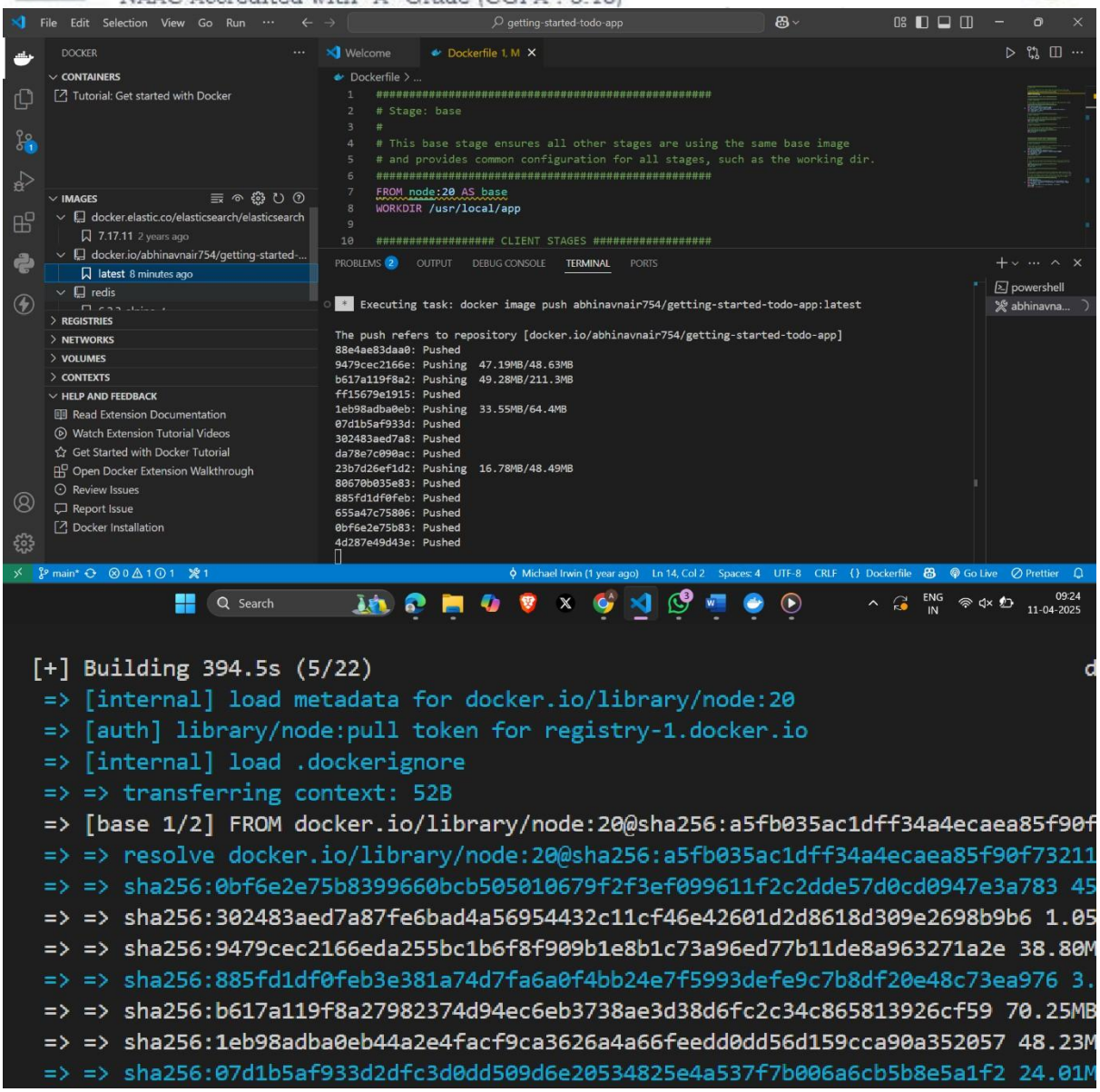
Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

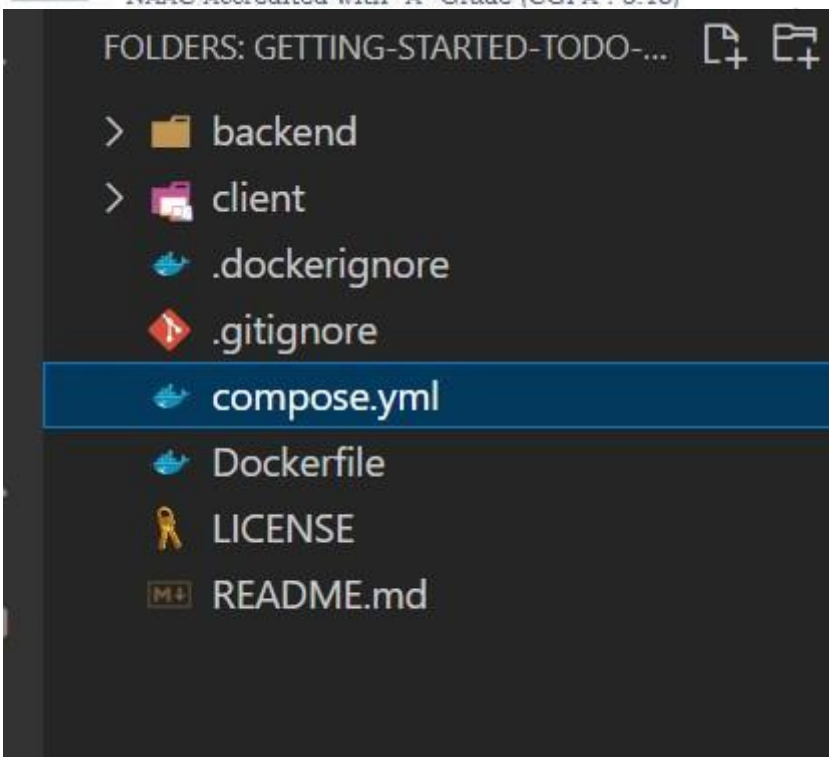


Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)




```
[+] Building 394.5s (5/22)
 => [internal] load metadata for docker.io/library/node:20
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 52B
 => [base 1/2] FROM docker.io/library/node:20@sha256:a5fb035ac1dff34a4ecaea85f90f
 => => resolve docker.io/library/node:20@sha256:a5fb035ac1dff34a4ecaea85f90f73211
 => => sha256:0bf6e2e75b8399660bcb505010679f2f3ef099611f2c2dde57d0cd0947e3a783 45
 => => sha256:302483aed7a87fe6bad4a56954432c11cf46e42601d2d8618d309e2698b9b6 1.05
 => => sha256:9479cec2166eda255bc1b6f8f909b1e8b1c73a96ed77b11de8a963271a2e 38.80M
 => => sha256:885fd1df0feb3e381a74d7fa6a0f4bb24e7f5993defe9c7b8df20e48c73ea976 3.
 => => sha256:b617a119f8a27982374d94ec6eb3738ae3d38d6fc2c34c865813926cf59 70.25MB
 => => sha256:1eb98adba0eb44a2e4facf9ca3626a4a66feedd0dd56d159cca90a352057 48.23M
 => => sha256:07d1b5af933d2dfc3d0dd509d6e20534825e4a537f7b006a6cb5b8e5a1f2 24.01M
```



Conclusion: In this experiment we build and pushed the image on docker

References:

1. https://docs.docker.com/get-started/workshop/02_our_app/
2. https://docs.docker.com/get-started/docker-concepts/building-images/writing-a-dockerfile/
3. https://phoenixnap.com/kb/create-docker-images-with-dockerfile
4. https://www.cherryservers.com/blog/docker-build-command
5. https://kodekloud.com/blog/how-to-build-a-docker-image/

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME: Anish Sharma **DIV: IT1-1** **ROLL: I011**

EXPERIMENT NO. 8

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To Install and Configure Kubernetes-based Configuration Management and Provisioning Tools.

THEORY:

Setting up **Kubernetes-based Configuration Management and Provisioning Tools** on **Windows 10** involves several steps, including installing Kubernetes itself (via Minikube or Docker Desktop), then deploying tools like **Helm**, **Kustomize**, or **Ansible with Kubernetes plugins**.

To install and configure Kubernetes-based configuration management and provisioning tools on Windows 10, you'll primarily use Minikube and kubectl. Minikube provides a single-node Kubernetes cluster for local development, while kubectl is the command-line tool for interacting with the cluster. Helm, a package manager for Kubernetes, is also often used for deploying applications and managing configurations.

1. Install Minikube:

- Download and install Minikube
- Minikube will typically install a virtual machine using Hyper-V, which is a Windows virtualization platform.
- Verify the installation by running minikube status in the command prompt or PowerShell.

2. **Install kubectl**:

- Download and install the kubectl binary for Windows.
- You can use Chocolatey or directly download the binary from the Kubernetes website.
- Ensure kubectl is added to your system's environment variables so you can run it from any directory.

3. Install Helm:

- Helm is used to manage Kubernetes manifests, which are like blueprints for deploying applications.
- Install Helm using Chocolatey: choco install helm.
- Verify the installation by running helm version.

Implementation:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Step 1: Install Required Tools

1. **Install WSL (Windows Subsystem for Linux)**
   - Kubernetes tools run best in a Linux-like environment.
   - Open PowerShell as Administrator and run: powershell wsl --install
   - Reboot and choose a Linux distribution (e.g., Ubuntu) from Microsoft Store.
2. **Install Minikube or Docker Desktop** • Minikube (Recommended for learning):
   - Download Minikube: https://minikube.sigs.k8s.io/docs/start/
   - Install using Chocolatey: choco install minikube Start Minikube: minikube start --driver=docker
   - Docker Desktop (Includes Kubernetes):
   - Download: https://www.docker.com/products/docker-desktop/
   - Enable Kubernetes from Docker Desktop settings.

Step 2: Install Configuration Management Tools

- Helm (Package Manager for Kubernetes) Install Helm on Windows: choco install kubernetes-helm Verify: bash helm version
- Kustomize (YAML Customization Tool)
  Install:
  choco install kustomize Or via kubectl
  plugin: kubectl kustomize ./your-config-folder
- Ansible (for provisioning, not native to Kubernetes) Install inside WSL (Ubuntu):
  sudo apt update && sudo apt install ansible For
  Kubernetes modules, install:
  ansible-galaxy collection install community.kubernetes

Step 3: Configure Your Environment

- Set up kubectl Install kubectl: choco install kubernetes-cli
- Verify: kubectl version --client
  Configure it to talk to Minikube or Docker Desktop Kubernetes cluster:
  kubectl config use-context minikube

Step 4: Test Your Setup Helm
helm repo add bitnami https://charts.bitnami.com/bitnami helm
install my-nginx bitnami/nginx
Kustomize

Create a base deployment YAML and a customization layer. Then run: bash
kubectl apply -k ./kustomize-directory

OUTPUT:

Academic Year: 2024 - 2025


```
[~]$ minikube status
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
[~]$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   21h   v1.17.0
[~]$
```

```
[~]$ minikube start --vm-driver=hyperkit
😄  minikube v1.6.2 on Darwin 10.14.1
✨  Selecting 'hyperkit' driver from user configuration (alternates: [])
💡  Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to
delete this one.
🔄  Starting existing hyperkit VM for "minikube" ...
⌛  Waiting for the host to be provisioned ...
🐳  Preparing Kubernetes v1.17.0 on Docker '19.03.5' ...
🚀  Launching Kubernetes ...
🏄  Done! kubectl is now configured to use "minikube"
[~]$
[~]$ minikube
Minikube is a CLI tool that provisions and manages single-node Kubernetes
clusters optimized for development workflows.

Basic Commands:
  start          Starts a local kubernetes cluster
  status         Gets the status of a local kubernetes cluster
  stop           Stops a running local kubernetes cluster
  delete         Deletes a local kubernetes cluster
  dashboard      Access the kubernetes dashboard running within the minikube
cluster

Images Commands:
  docker-env     Sets up docker env variables; similar to '$(docker-machine
env)'
  cache          Add or delete an image from the local cache.
```

Academic Year: 2024 - 2025


```
[~]$ kubectl
kubectl controls the Kubernetes cluster manager.

 Find more information at:
https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service, deployment or pod and
expose it as a new Kubernetes Service
  run            Run a particular image on the cluster
  set            Set specific features on objects

Basic Commands (Intermediate):
  explain        Documentation of resources
  get            Display one or many resources
  edit           Edit a resource on the server
==> New Formulae
apollo-cli                                eureka
==> Updated Formulae
auditbeat         git-lfs           jetty             monolith          telegraf
borgmatic         h3                logstash          phpstan           terraform-docs
elasticsearch     hlint             metricbeat        serverless        traefik
exploitdb         hub               micronaut         skopeo
[~]$ brew install hyperkit
==> Downloading https://homebrew.bintray.com/bottles/hyperkit-0.20190802.mojave.bottle.ta
Already downloaded: /Users/nanajanashia/Library/Caches/Homebrew/downloads/aa5d075eb83c262
874438ed5d4eb4a76581a740e89feca7057add827cc7da153--hyperkit-0.20190802.mojave.bottle.tar.
gz
==> Pouring hyperkit-0.20190802.mojave.bottle.tar.gz
🍺  /usr/local/Cellar/hyperkit/0.20190802: 5 files, 4.0MB
[~]$ brew install minikube
Updating Homebrew...
==> Installing dependencies for minikube: kubernetes-cli
==> Installing minikube dependency: kubernetes-cli
==> Downloading https://homebrew.bintray.com/bottles/kubernetes-cli-1.17.1.mojave.bottle.
Already downloaded: /Users/nanajanashia/Library/Caches/Homebrew/downloads/da86e6d5a7a1dcd
14832ef280f222da5a75d2980cb597ed79a5975a0e35c4fd2--kubernetes-cli-1.17.1.mojave.bottle.ta
r.gz
==> Pouring kubernetes-cli-1.17.1.mojave.bottle.tar.gz
==> Caveats
```

```
==> Pouring hyperkit-0.20190802.mojave.bottle.tar.gz
🍺  /usr/local/Cellar/hyperkit/0.20190802: 5 files, 4.0MB
[~]$ brew install minikube
Updating Homebrew...
==> Installing dependencies for minikube: kubernetes-cli
==> Installing minikube dependency: kubernetes-cli
==> Downloading https://homebrew.bintray.com/bottles/kubernetes-cli-1.17.1.mojave.bottle.
Already downloaded: /Users/nanajanashia/Library/Caches/Homebrew/downloads/da86e6d5a7a1dcd
14832ef280f222da5a75d2980cb597ed79a5975a0e35c4fd2--kubernetes-cli-1.17.1.mojave.bottle.ta
r.gz
==> Pouring kubernetes-cli-1.17.1.mojave.bottle.tar.gz
==> Caveats
Bash completion has been installed to:
  /usr/local/etc/bash_completion.d

zsh completions have been installed to:
  /usr/local/share/zsh/site-functions
==> Summary
🍺  /usr/local/Cellar/kubernetes-cli/1.17.1: 235 files, 49MB
==> Installing minikube
[~]$ brew update
Updated 3 taps (homebrew/core, homebrew/cask and caskroom/versions).
==> New Formulae
apollo-cli                                eureka
==> Updated Formulae
auditbeat         git-lfs           jetty             monolith          telegraf
borgmatic         h3                logstash          phpstan           terraform-docs
elasticsearch     hlint             metricbeat        serverless        traefik
exploitdb         hub               micronaut         skopeo
[~]$ brew install hyperkit
==> Downloading https://homebrew.bintray.com/bottles/hyperkit-0.20190802.mojave.bottle.ta
Already downloaded: /Users/nanajanashia/Library/Caches/Homebrew/downloads/aa5d075eb83c262
874438ed5d4eb4a76581a740e89feca7057add827cc7da153--hyperkit-0.20190802.mojave.bottle.tar.
gz
==> Pouring hyperkit-0.20190802.mojave.bottle.tar.gz
🍺  /usr/local/Cellar/hyperkit/0.20190802: 5 files, 4.0MB
```

Conclusion:

In this practical, we have explained in detail how to install Kubernetes with Hyper-V. Also, we have tackled what requirements we need, both in terms of the software and hardware. We have explained how to install Hyper-V and Docker on Windows 10.

References:

1. https://www.knowledgehut.com/blog/devops/install-kubernetes-on-windows
2. https://learnk8s.io/installing-docker-kubernetes-windows
3. https://spacelift.io/blog/install-kubernetes
4. https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/
5. https://www.docker.com/products/docker-desktop/

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 8-5-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME : Anish Sharma **DIV: IT1-1** **ROLL: I011**

EXPERIMENT NO. 9

CO/LO: Apply DevOps principles to meet software development requirements.

AIM / OBJECTIVE: To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet /Ansible.

THEORY:

To install and configure pull-based configuration management with Puppet or Ansible on Windows 10, you'll need to set up either the Puppet master or the Ansible control node, then install the appropriate agent on the Windows 10 machine. Puppet uses a pull-based approach where agents periodically check in with the master for updates, while Ansible is typically used in a push-based manner where the control node sends commands to the managed nodes.

Puppet on Windows 10:

1. Install Puppet Master: Download and install the Puppet Master on a server or workstation.
2. Install Puppet Agent: Download and install the Puppet Agent on the Windows 10 machine.
3. Configure Puppet Agent: Configure the agent to point to the Puppet Master server.
4. Create Puppet Modules: Develop Puppet manifests and modules to define the desired state of the Windows 10 system.
5. Apply Puppet Configurations: Run Puppet on the agent to apply the desired state.

Ansible on Windows 10:

1. Install Ansible Control Node: Download and install Ansible on a server or workstation.
2. Install Ansible Agent (Optional): While Ansible can be used in a push-based manner without an agent, you might need to install an agent on Windows 10 for specific functionalities.
3. Configure SSH: Ensure SSH is configured on the Windows 10 machine for Ansible to connect.

4. Create Ansible Playbooks: Develop Ansible playbooks to define the desired state of the Windows 10 system.

5. Run Ansible Playbooks: Run the playbooks from the control node to apply the desired state to the Windows 10 machine.

Implementation:

To install and configure pull-based software configuration management and provisioning tools on Windows 10 using Puppet or Ansible, here's a step-by-step guide for each tool.

OPTION 1: Using Puppet **(Windows-native support)**

Puppet supports a pull-based model where the agent pulls configurations from the Puppet Master (Server).

Step 1: Install Puppet Agent on Windows 10

1. Download Puppet Agent for Windows from the official site:
   - https://puppet.com/download-puppet-enterprise
2. Run the installer and follow the prompts.

Step 2: Configure Puppet Agent

1. Open Command Prompt as Administrator.
2. Run the following command to configure the Puppet agent to connect to the Puppet master:

   puppet config set server <puppet-master-hostname>

3. You can also set the runinterval (optional):

   puppet config set runinterval 30m **Step**

3: Run the Puppet Agent

Run manually for testing:

puppet agent -t

To enable the pull-based service (runs every 30 mins by default):

puppet resource service puppet ensure=running enable=true

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

OPTION 2: Using Ansible **(Control from Windows via WSL)**

Ansible doesn't run natively on Windows, but you can use WSL (Windows Subsystem for Linux) to run it and control Windows/other hosts via WinRM.

Step 1: Install WSL + Ubuntu

Open PowerShell as Admin and run:

```
wsl --install
```

This installs WSL with Ubuntu by default. Restart your machine if prompted.

Step 2: Install Ansible in WSL

Launch Ubuntu from Start menu, then run:

```
sudo apt update sudo apt
install ansible -y
```

Step 3: Configure Ansible to Manage Windows Host

1. Install pywinrm:

```
pip install "pywinrm>=0.3.0"
```

2. Configure WinRM on your Windows machine (you can use a PowerShell script from Ansible's docs or manually enable it):

   ```
   winrm quickconfig
   ```

3. Create an inventory file in WSL:

   ```
   ini
    [windows]
   192.168.1.10

   [windows:vars]
   ansible_user=Administrator
   ansible_password=YourPassword
   ansible_connection=winrm
   ansible_winrm_server_cert_validation=ignore
   ```

4. Test connection:

   ```
   ansible windows -i inventory -m win_ping
   ```

Conclusion:

Installing and configuring pull-based configuration management tools like **Puppet** and **Ansible** on **Windows 10** is achievable with some setup:

- **Puppet** is natively supported on Windows and is ideal for pull-based management. The Puppet Agent can be easily installed, configured, and scheduled to pull configurations from a Puppet Master at regular intervals.
- **Ansible**, while primarily push-based and not natively supported on Windows, can be made to work in a pull-like fashion using `ansible-pull` within **WSL (Windows Subsystem for Linux)**. This method uses Git repositories to manage and deliver configurations.

For a native Windows experience and true pull-based model, **Puppet** is the better fit. However, if you prefer Git-based workflows and already use WSL, **Ansible pull** is a flexible alternative.

References:

1. https://www.puppet.com/why-puppet/use-cases/continuous-configuration-automation
2. https://www.puppet.com/blog/windows-configuration-management
3. https://blog.risingstack.com/getting-started-with-ansible-infrastructure-automation/
4. https://www.futurelearn.com/info/courses/infrastructureascode/0/steps/185965

Academic Year: 2024 - 2025

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITHN1L1 **DATE: 8-05-2025**

COURSE NAME: DevOps Laboratory **CLASS: TY BTech**

NAME : Anish Sharma **DIV: IT1-1** **ROLL: I011**

EXPERIMENT NO. 10

CO/LO: Apply DevOps principles to meet software development requirements.

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

THEORY:

To learn Software Configuration Management (SCM) and provisioning using Puppet, want to follow a structured path focusing on Puppet's key building blocks: Manifests, Modules, Classes, and Functions.

To effectively learn Software Configuration Management and provisioning using Puppet, understanding its key building blocks is crucial: Manifests, Modules, Classes, and Functions. These components work together to automate and centralize infrastructure management.

1. Manifests:

- A manifest is a file containing Puppet code that describes how resources should be configured on target systems.
- It's the core of a Puppet program and defines the desired state of the system.
- Manifests are typically located in the manifests directory of a module.
- They can include various types of resources, such as files, packages, users, services, and more.

2. Modules:

- Modules are the fundamental building blocks of Puppet, offering reusability and shareability.
- They can contain various Puppet elements, including classes, defined types, tasks, functions, resource types, and providers.
- Modules are installed in the Puppet module path.
- They encapsulate specific functionalities, like deploying a web server, configuring databases, or installing software.

3. Classes:

- Classes are reusable blocks of code that can be included in manifests.
- They define a set of related resources and can be used to modularize and organize configurations.

- Classes can be parameterized, allowing for flexible and customized deployments.
- For example, a class could define the configuration for a web server, including installing necessary packages, creating directories, and starting services.
- 4. Functions:
- Functions are reusable blocks of code that perform specific tasks within Puppet code.
- They can be used to manipulate data, perform calculations, or execute other actions.
- Examples include functions for accessing facts, querying databases, or generating output.
- Functions can enhance the flexibility and expressiveness of Puppet code.

Key Concepts for Learning Puppet:

- Puppet Agent:

The software that runs on managed nodes, applying configurations received from the Puppet master.

- Puppet Master:

The central server that stores Puppet code, compiles configurations, and serves them to agents.

- Resources:

The individual items that Puppet manages, such as files, packages, users, and services.

- Configuration Management:

The process of managing and automating the configuration of infrastructure and software.

- Provisioning:

The process of setting up and configuring infrastructure and software resources.

Resources for Learning Puppet:

- Puppet Documentation:

The official Puppet documentation provides comprehensive information about Puppet's features, syntax, and best practices.

- Online Courses:

Several online platforms offer courses on Puppet, such as Udemy, Coursera, and Pluralsight.

- Puppet Labs:

Puppet Labs provides a variety of resources for learning and using Puppet, including tutorials, examples, and support materials.

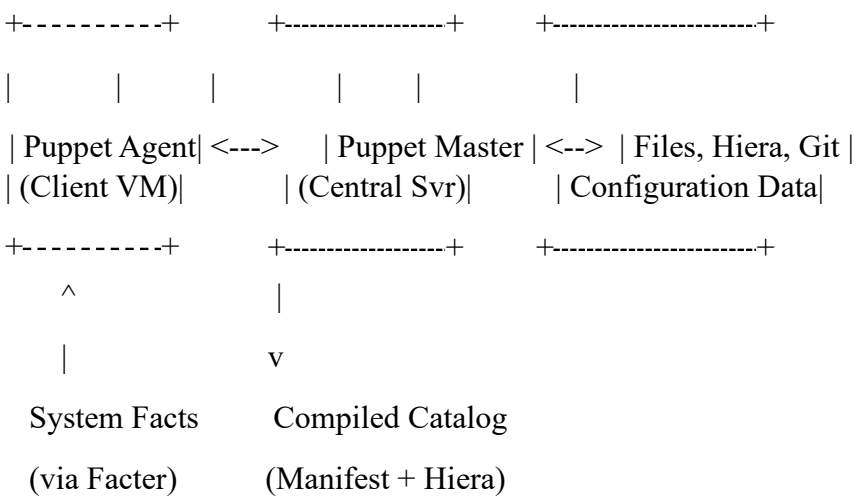
- Community Forums:

Engaging with the Puppet community can provide valuable insights and assistance.

By understanding these core components and utilizing available learning resources, you can effectively learn Puppet and leverage its power for Software Configuration Management and provisioning.

Implementation:

Step 1: Understand the Basics of SCM & Puppet



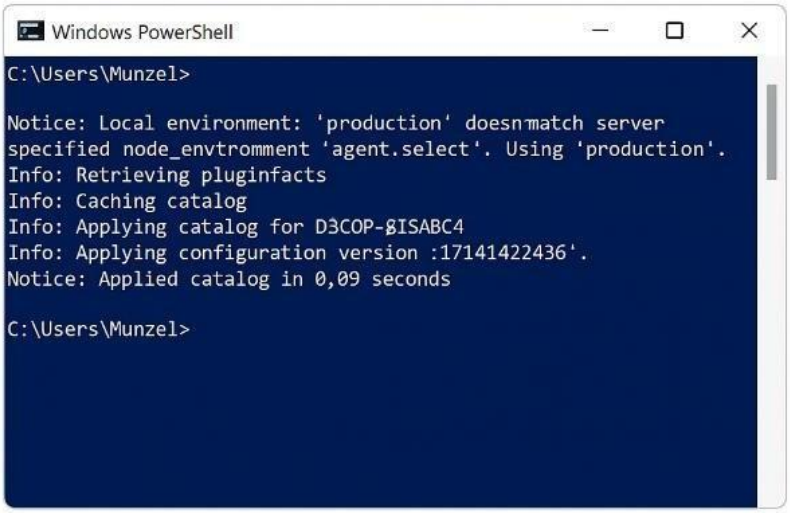
Step 2: Set Up Puppet on Windows 10

Run Puppet in two roles:

- As a master (server) or agent (node).
- On Windows 10, you'll typically set it up in agent mode for practice, or use Puppet Bolt for masterless operations.

Option A: Install Puppet Agent on Windows

1. Download Puppet Agent: Puppet Downloads 2.
   Install it on Windows 10.
3. Configure it to connect to a Puppet Master (Linux VM or cloud).
   - Use puppet.conf to define server settings.
4. Test with:
   puppet agent -t



Option B: Use Puppet Bolt (Recommended for Beginners)

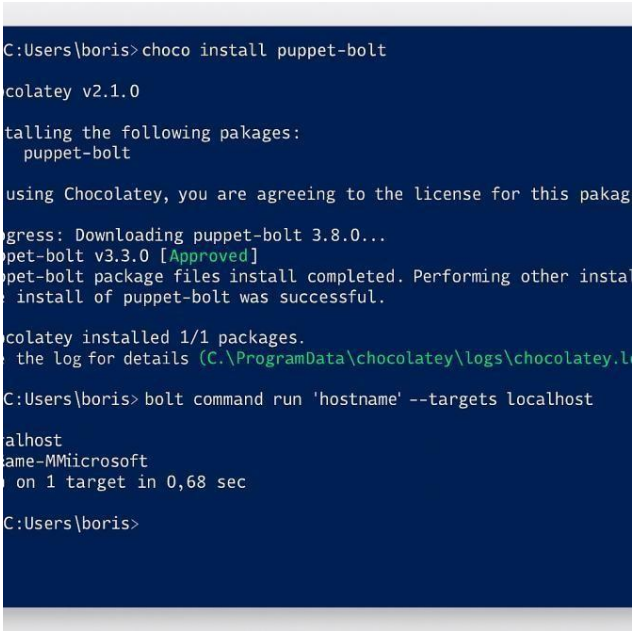
Puppet Bolt is an agentless tool for running Puppet tasks and plans.

1. Install Bolt:
   - Official Bolt Guide
   - Use Chocolatey:

     choco install puppet-bolt

2. Run a sample task:

   bolt command run 'hostname' --targets localhost



Step 3: Learn Puppet DSL & Manifests Learn:

- Puppet resources and manifests
- Classes and modules
- Writing and applying simple Puppet code

Example Manifest:
file { 'C:/example.txt': ensure
=> present,
content => 'Hello from Puppet!', }

Apply it using:

puppet apply .\example.pp

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

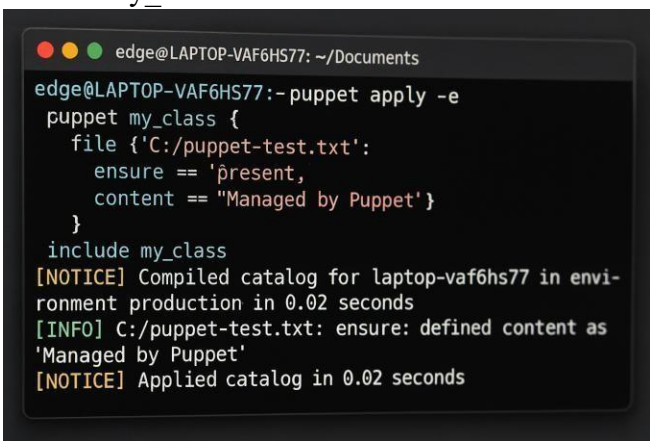
Step 4: Practice with Puppet Blocks

Puppet Blocks (also called code blocks or logical blocks) are how structure Puppet code. Key block types:

- class, define, node, if, case, unless, etc.

Example Block:

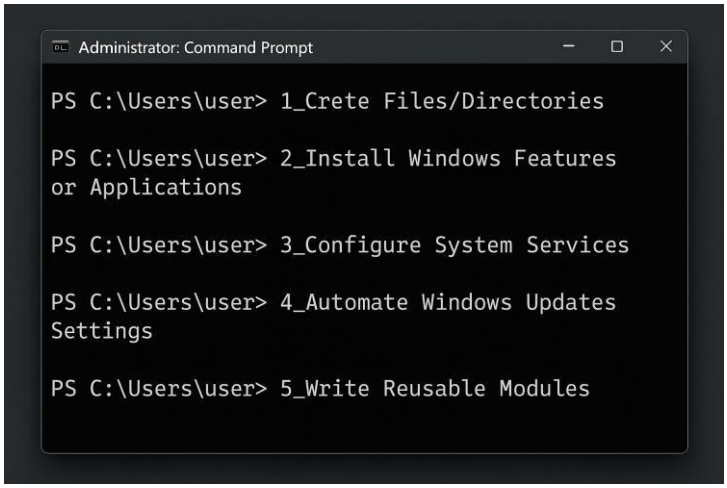
```
class my_class {
  file { 'C:/puppet-test.txt': ensure
    => 'present',
    content => "Managed by Puppet",
  }
}
include my_class
```



Step 5: Projects to Practice

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

1. Create Files/Directories
2. Install Windows Features or Applications
3. Configure System Services
4. Automate Windows Updates Settings
5. Write Reusable Modules



Tools

- Visual Studio Code + Puppet Extension
- PowerShell (for local testing and scripting)
- Chocolatey (to install packages)

Further Learning

- Puppet Forge
- Learn Puppet Course
- GitHub repositories with sample manifests and modules

Conclusion:

Software Configuration Management (SCM) **and** provisioning with Puppet **on** Windows 10 provides strong foundation in automation and DevOps practices. By installing Puppet (or Puppet Bolt), understanding its DSL (Domain Specific Language), and practicing real-world provisioning tasks—like managing files, services, and software—gain hands-on experience in infrastructure automation. Though Puppet is more native to Linux, its support for Windows makes it a valuable tool for cross-platform environments.

References:

1. https://www.tutorialspoint.com/puppet/index.htm
2. https://www.puppet.com/blog/windows-configuration-management
3. https://www.puppet.com/why-puppet/use-cases/continuous-configuration-automation
4. https://www.puppet.com/blog/windows-configuration-management