KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)

Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH,
Narayanguda, Hyderabad, Telangana – 500029

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

LAB RECORD

SOFTWARE ENGINEERING LAB

B. Tech. III YEAR I SEM (KR23)
ACADEMIC YEAR
2025-26

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad
Narayanguda, Hyderabad, Telangana – 500029

Certificate

This is to certify that following is a Bonafide Record of the workbook task done by _____bearing Roll No_____ of _____

Branch of _____ year B. Tech. Course in the_____

Subject during the Academic year _____& _____under our supervision.

Number of week tasks completed: _____

Signature of Staff Member Incharge Signature of Head of the Dept.

Signature of Internal Examiner Signature of External Examiner

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(AN AUTONOMOUS INSTITUTION)
Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad
Narayanguda, Hyderabad, Telangana – 500029

Daily Laboratory Assessment Sheet

Name of the Lab: Student Name:
Branch & Section: HT. No:

Sr. No.	Name of the Experiment	Date	Observation Marks (5M)	Record Marks (5M)	Viva Voice Marks(10M)	Total Marks (20M)	Signature of Faculty
1	Software Installation & SRS Document	25-07-2025					
2	Exploring git local and remote commands	29-07-2025					
3	Collaborative coding using git	05-08-2025					
4	Build and package Java and Web applications using Maven	13-08-2025					
5	Docker CLI commands	19-08-2025					
6	Docker	26-08-2025					
7	Creating a Multi-Module MavenProject	02-09-2025					
8	Jenkins Automation	16-09-2025					
9	Pipeline Creation using script	23-09-2025					
10	minikibe and nagios	28-10-2025					
11	Jenkins-CI/CD	04-11-2025					
12.	Creation of virtual machine for Ubuntu OS and Deploying the web application	11-11-2025					
	Total						

Faculty Incharge

INDEX

Experiment-1 : Software Installation & SRS Document

a. Abstract

This project focuses on building a smart Question-and-Answer (Q&A) system that can read documents and answer user questions based on the content in those documents — just like asking a knowledgeable assistant. The system supports different types of documents such as PDFs, Word files, text files, web pages, and notes. Once these documents are uploaded, the system breaks them into meaningful parts and converts them into semantic embeddings, which means it stores the meaning of the content, not just the words. These are saved in a special type of searchable storage called a vector database. When a user asks a question, the system first finds the most relevant parts of the documents and then uses Google Gemini, a powerful AI model, to understand the question and generate a clear, accurate answer. The system also uses a technique called Retrieval-Augmented Generation (RAG) — which means it combines document search with intelligent answer generation. A key feature of this system is memory. It remembers what the user asked before (short-term memory) and can even remember past conversations (long-term memory), which helps it give more personalized and meaningful answers over time. The system also learns and improves continuously by collecting feedback, correcting errors, and testing different answer styles. It is tested using standard question sets like SQuAD 2.0 and CoQA to make sure it's accurate, fast, and reliable. The project is built using FastAPI or Flask for managing the system's backend and Streamlit or React to design the user interface. Overall, this system transforms the boring task of document search into a smart, helpful, and interactive experience, useful for students, teachers, researchers, and anyone who works with a lot of documents.

a. Functional Requirements

1. **Document Upload and Processing**:
    - Support uploading of PDFs, Word documents (.doc, .docx), text files (.txt), web pages (via URL), and notes.
    - Segment documents into meaningful chunks for processing.
    - Store embeddings in a vector database (e.g., Pinecone, Weaviate, or FAISS).

2. **Question Handling**:
    - Process questions using natural language understanding (via Gemini or similar AI model).
    - Retrieve relevant document chunks from the vector database based on semantic similarity.

3. **Memory and Context Management**:
    - Maintain short-term memory to track the context of a single conversation session.
    - Store long-term memory to recall past interactions and user preferences across sessions.

4. **Feedback and Improvement**:
    - Allow users to provide feedback on answer quality (e.g., thumbs up/down, comments).
    - Log feedback and errors for continuous learning and model fine-tuning.

5. **Evaluation and Testing**:
    - Evaluate system performance using standard Q&A datasets (e.g., SQuAD 2.0, CoQA).
    - Provide metrics for accuracy, response time, and relevance of answers.

6. **User Interface**:
    - Provide an intuitive interface for document upload, question input, and answer display.
    - Display retrieved document chunks alongside answers for transparency.
    - Allow users to view conversation history and manage uploaded documents.

b. Non-Functional Requirements

1. **Usability**:
    - Ensure the interface is user-friendly, with clear instructions and minimal learning curve.
    - Support responsive design for access on desktops, tablets, and mobile devices.

2. **Performance**:
    - Process document uploads and generate embeddings within 5 sec for files up to 10 MB.
    - Retrieve relevant document chunks and generate answers within 2 seconds per query.

3. **Security**:
    - Encrypt uploaded documents and user data during transmission and storage.
    - Implement user authentication and role access control to protect sensitive docs.

4. **Reliability**:
    - Achieve 99.9% system uptime, excluding scheduled maintenance.
    - Handle errors gracefully, providing meaningful messages to users

5. **Scalability**:
    - Support scaling to handle increased document volumes and user traffic.
    - Allow integration of additional document types or AI models in the future.

c. User Identification

1. **Students**:
    - Use the system to extract answers from study materials, research papers, or notes for assignments and exam preparation.

2. **Teachers**:
    - Utilize the system to create teaching materials or answer student queries based on course documents.

3. **Researchers**:
    - Query large volumes of academic papers or reports to extract specific information quickly.

4. **Administrators**:

    - Manage system settings, user access, and document storage (for enterprise deployments).

d. Workflow of Each User

Below are detailed workflows for each identified user type, outlining step-by-step interactions with the smart Q&A system. Workflows incorporate document upload, processing, question handling, memory management, feedback, and UI features as per the functional requirements.

Students

1. **Onboarding & Document Upload**: Student logs in, uploads study materials (PDF notes, Word assignments, .txt summaries, or web URLs of research articles). System extracts text, chunks into paragraphs/sections, generates semantic embeddings, and stores in vector database. Processing completes in 2sec.
2. **Question Session Initiation**: Student navigates to the intuitive React interface, starts a new conversation session. Short-term memory activates to track context within the session.
3. **Query Submission**: Types a question (e.g., "What is the main theorem in Chapter 3 of my uploaded PDF?"). System uses Google Gemini for NLU, similarity from vector DB, applies RAG to generate concise answer. Answer displays in <2 sec, with source chunks shown for transparency. Long-term memory recalls past sessions for personalized context.
4. **Feedback & Improvement**: Rates answer (thumbs up/down) or adds comments. Feedback logs for system learning; student views conversation history.
5. **Session End & Management**: Views/manages uploaded documents; logs out. System evaluates performance internally using datasets like SQuAD 2.0.

Teachers

1. **Onboarding & Document Upload**: Teacher authenticates, uploads course materials (PDF slides, Word lesson plans, .txt syllabi, web URLs of resources, or notes). System processes and stores embeddings securely (encrypted, role-based access).
2. **Question Session Initiation**: Opens interface, starts session for creating materials or student support. Long-term memory recalls past interactions (e.g., frequent student query patterns).
3. **Query Submission**: Inputs question (e.g., "Summarize key points from the uploaded syllabus for a quiz"). RAG retrieves chunks, Gemini generates detailed/conversational answer with sources displayed.
4. **Feedback & Improvement**: Provides feedback to fine-tune (e.g., prefers detailed styles); reviews history to manage documents.
5. **Session End & Management**: Exports answers for teaching aids; system scales for multiple concurrent sessions.

Researchers

1. **Onboarding & Document Upload**: Researcher logs in with secure authentication, uploads large volumes (PDF papers, .docx reports, .txt datasets, web URLs). System chunks, embeds, and stores; handles scalability for high volumes.
2. **Question Session Initiation**: Starts session; long-term memory recalls prior research queries across sessions.
3. **Query Submission**: Asks complex question (e.g., "Compare findings on Topic X from all uploaded papers"). Semantic retrieval + RAG generates accurate, context-rich answer in <2 seconds, with transparent sources.

4. **Feedback & Improvement**: Submits feedback/comments for error correction; logs support continuous improvement.

5. **Session End & Management**: Manages document library; views history for ongoing research.

Administrators

1. **Onboarding & Access**: Admin logs in with elevated roles, accesses system settings (no personal document upload unless needed).

2. **Management Initiation**: Views dashboard for user access, document storage, and performance metrics.

3. **Monitoring & Queries**: Optionally uploads test docs; runs evaluation queries using SQuAD 2.0/CoQA datasets to check accuracy/response time.

4. **Configuration & Interaction**: Adjusts settings (e.g., user roles, A/B tests); reviews logs for feedback/errors. System handles gracefully with 99.9% uptime.

5. **Feedback & Scaling**: Aggregates user feedback for fine-tuning; scales infrastructure for traffic.

6. **Session End & Oversight**: Generates reports; ensures maintainability via modular code/documentation.

e. Use Cases

Students

- **Exam Preparation**: Uploads textbook PDFs and notes; asks "Solve this sample question from Chapter 5" → Gets step-by-step answer with sources;
- **Assignment Research**: Inputs web URL of an article; queries specifics → RAG pulls relevant chunks; long-term memory recalls similar past assignments for personalized summaries.
- **Quick Revision**: Asks comparative questions across multiple .txt files → Fast retrieval (<2 sec); feedback improves accuracy over sessions.

Teachers

- **Material Creation**: Uploads Word lesson plans; asks "Generate quiz questions from Section 2" → Detailed answers with sources; A/B tests conversational vs. formal styles.
- **Student Query Support**: Shares session with students; answers based on course PDFs → Context from long-term memory (e.g., common misconceptions); transparent chunks build trust.
- **Grading Aid**: Queries student-submitted .docx → Extracts key points; feedback logs

Researchers

- **Literature Review**: Uploads dozens of PDF papers; asks "Synthesize trends in AI ethics from 2020-2025 docs" → RAG aggregates from vector DB; memory recalls prior queries for evolving insights.
- **Data Extraction**: Web URLs + .txt datasets; precise queries like "Extract methodology from Paper X" → Accurate, cited responses; evaluates via CoQA metrics internally.
- **Collaboration**: Shares document library; follow-ups in sessions → Scalable for large volumes.

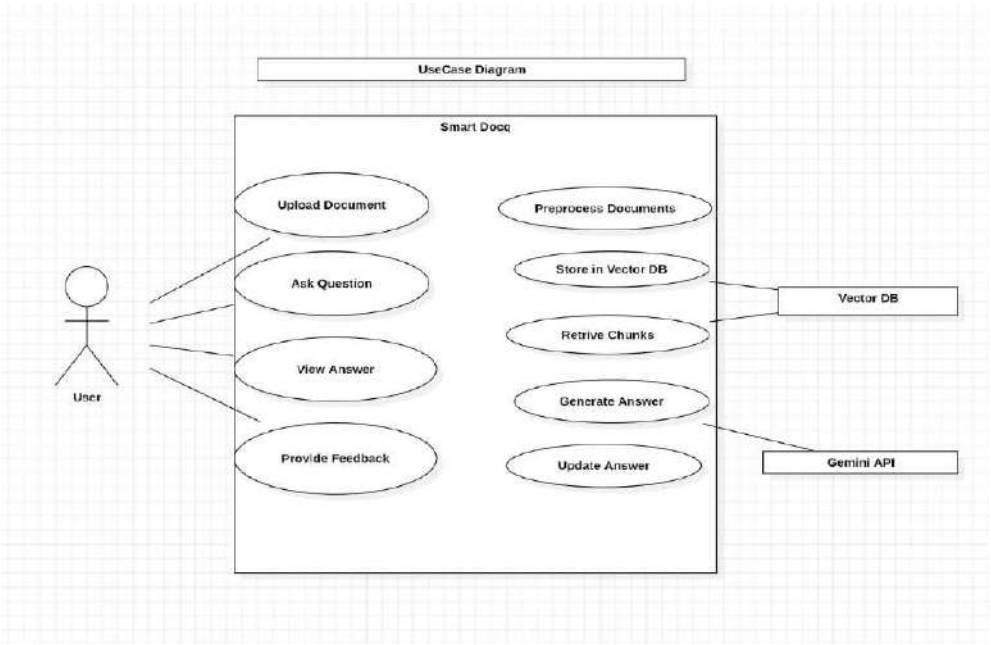
Administrators

- **System Monitoring**: Runs test queries on sample docs; views metrics (accuracy from SQuAD 2.0, uptime); A/B tests new AI integrations.
- **User Management**: Revokes access for sensitive docs; reviews feedback logs for global improvements; ensures scalability during peak usage.

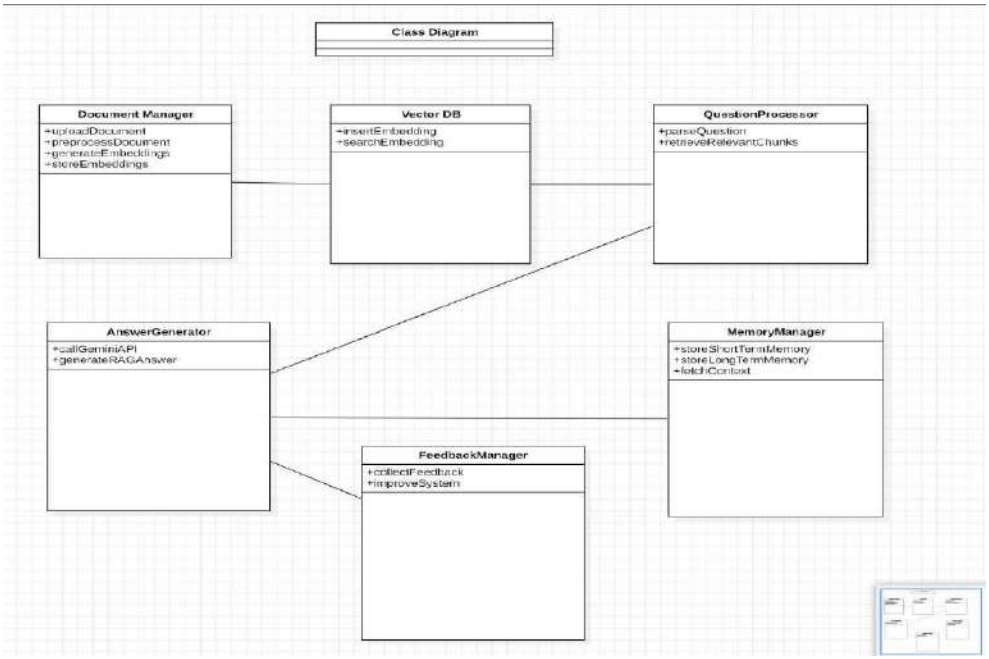
- **Compliance Audit**: Queries system logs; verifies encryption/privacy; manages document deletion for GDPR compliance.

f. UML Diagrams

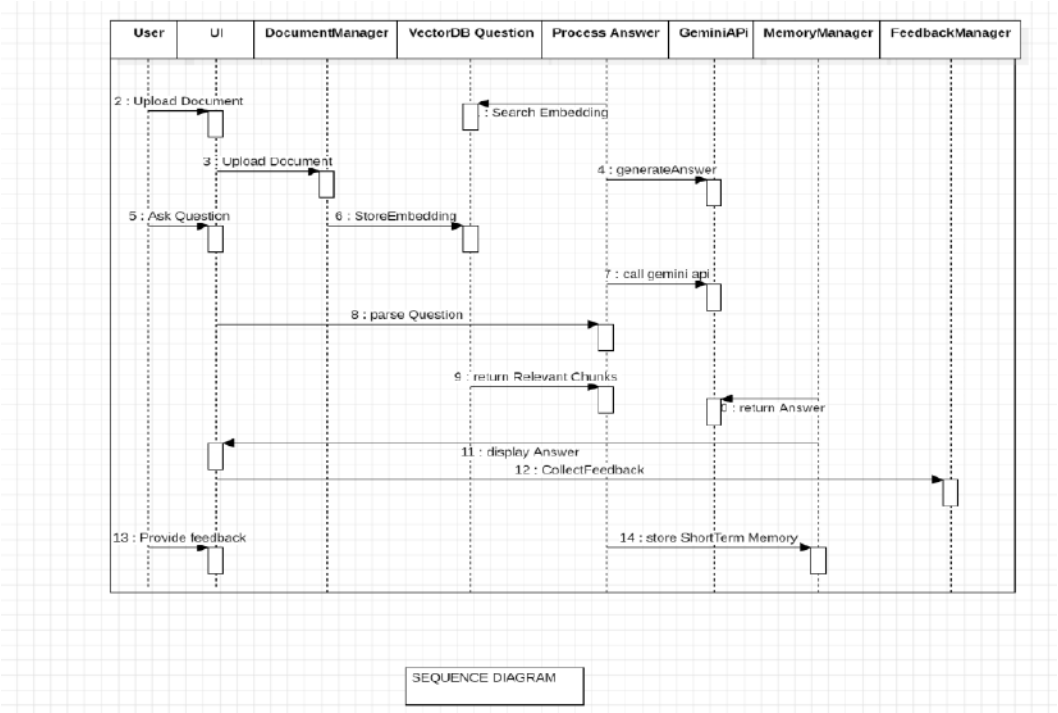
Use Case Diagram



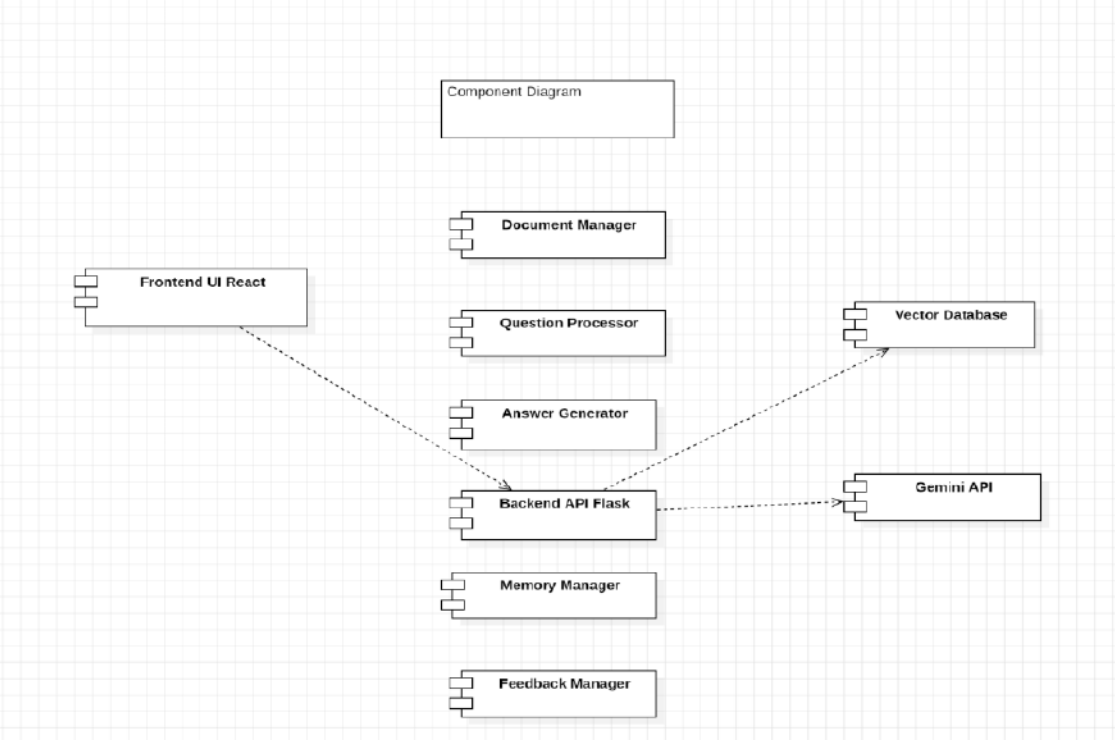
Class Diagram



Sequence Diagram



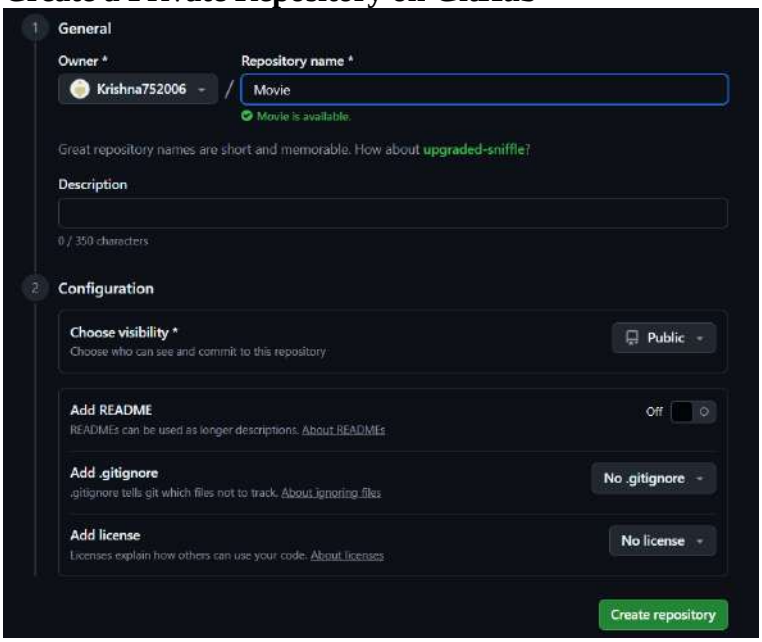
Component Diagram



Experiment-2 : Exploring git local and remote commands on the multi-folder project

a. Pushing multi-folder project into private repository (by student).

1. **Create a Private Repository on GitHub**



2. **Initialize Git in Your Local Project (if not already done)**

```
C:\Movie>git init
Initialized empty Git repository in C:/Movie/.git/
```

3. **Add the Remote Repository (in bash)**

```
C:\Movie>git remote add origin https://github.com/HARIVIGNESHRAO/Movie.git
```

4. **Add and Commit Your Files(in bash)**

```
C:\Movie>git add .
```

b. Students must explore all listed git commands on the multi-folder project in local and remote repository.

1. git version

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git version
git version 2.47.1.windows.1
```

2. git config: Configures Git settings. Commonly used to set up user information

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git config --global user.email "harivigneshrao151@gmail.com"
```

3. **git config --list**: Displays all the Git configurations for the current user.

```
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=harivigneshrao151@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

2. Repository Management

- **git init**: Initializes a new Git repository in the current directory

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git init
Reinitialized existing Git repository in C:/Movie/.git/
```

- **git clone**: Creates a copy of an existing Git repository from a remote source (e.g., GitHub) to your local machine

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git clone https://github.com/savram674/Movie-Ticket.git
Cloning into 'Movie-Ticket'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 10.43 KiB | 395.00 KiB/s, done.
```

- **git remote –v:** To see the remote repository that is connected to your local repository

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git remote -v
origin  https://github.com/HARIVIGNESHRAO/Movie.git (fetch)
origin  https://github.com/HARIVIGNESHRAO/Movie.git (push)
```

- **git remote add :** To add a new remote repository to your local repository

```
fs@DESKTOP-GGREQK6 MINGW64 /c/Movie (master)
$ git remote add origin https://github.com/HARIVIGNESHRAO/Movie.git
error: remote origin already exists.
```

git push –u origin main

```
C:\Movies>git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 10.42 KiB | 2.60 MiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/HARIVIGNESHRAO/Movie.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

git pull origin main

```
C:\Movies>git pull origin main
From https://github.com/HARIVIGNESHRAO/Movie
 * branch            main       -> FETCH_HEAD
Already up to date.
```

c. Students must explore all git commands on given scenario-based question

Scenario based questions on basic git commands

1. You made changes to one file in the above project but haven't staged them yet. You realize they were a mistake. What Git command will you use to discard the local changes?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git restore file.txt
```

2. You accidentally ran git add file1.txt (note instead of file.txt consider one file from above project), but you're not ready to commit yet. How do you remove it from the staging area without losing the changes?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git add file1.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git status
On branch main
Your branch is ahead of 'origin1/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file1.txt


91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git reset file1.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git status
On branch main
Your branch is ahead of 'origin1/main' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt

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

3. You made a commit but typed the wrong commit message. You haven't pushed it yet. How do you fix it?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ notepad file1.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git add .

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git commit -m "added file.txt"
[main ba62c66] added file.txt
 1 file changed, 1 insertion(+), 1 deletion(-)

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git commit --amend -m "added file1.txt"
[main c1ab6b2] added file1.txt
 Date: Wed Jul 30 12:11:24 2025 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
```

4. You want to view the commit history of the current branch in a readable format. What Git command should you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git log
commit c1ab6b284134d937e9beb479e4e1ca5db5a953d1 (HEAD -> main)
Author: Harshitha-Macha <machaharshitha3@gmail.com>
Date:   Wed Jul 30 12:11:24 2025 +0530

    added file1.txt

commit d19996cba51da5980005b5af4b045d1d7c37ce54
Author: Harshitha-Macha <machaharshitha3@gmail.com>
Date:   Wed Jul 30 12:04:03 2025 +0530

    added file.txt

commit 343d053cc4d3c2cbef26ebd92fb4b038ed1aaf54
Author: Harshitha-Macha <machaharshitha3@gmail.com>
Date:   Tue Jul 29 14:39:49 2025 +0530

    2nd commit

commit 959067f8957d8ed563efb05e4e33882d8fd810cb (origin1/main)
Author: Harshitha-Macha <machaharshitha3@gmail.com>
Date:   Tue Jul 29 14:24:54 2025 +0530

    first commit
```

5. After cloning a repo, how can you set your name and email globally for all Git repositories?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git config --global user.name "Harshitha-Macha"

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git config --global user.email "machaharshitha3@gmail.com"
```

6. You've made some edits to your files but haven't staged them. How can you view the changes?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git diff
diff --git a/file.txt b/file.txt
index 80939b1..725d44b 100644
--- a/file.txt
+++ b/file.txt
@@ -1,3 +1,4 @@
 Name Harshitha
 Rno 23bd1a6633
 Year 3
+Section B
```

7. You're on the main branch but need to switch to feature/login. What command do you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git checkout -b feature/login
Switched to a new branch 'feature/login'

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature/login)
$ git branch
* feature/login
  main
```

8. You deleted the feature-ui branch by mistake. You haven't pushed the deletion. How ?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature-ui)
$ git checkout -d feature-ui
M       file.txt
HEAD is now at c1ab6b2 added file1.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ git reflog
c1ab6b2 (HEAD, main, feature/login, feature-ui) HEAD@{0}: checkout: moving from feature-ui to feature-u
i
c1ab6b2 (HEAD, main, feature/login, feature-ui) HEAD@{1}: checkout: moving from feature/login to featur
e-ui
c1ab6b2 (HEAD, main, feature/login, feature-ui) HEAD@{2}: checkout: moving from main to feature/login
c1ab6b2 (HEAD, main, feature/login, feature-ui) HEAD@{3}: commit (amend): added file1.txt
ba62c66 HEAD@{4}: commit: added file.txt
d19996c HEAD@{5}: commit: added file.txt
343d053 HEAD@{6}: commit: 2nd commit
959067f (origin1/main) HEAD@{7}: commit (initial): first commit

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ ^C

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ git reflog c1ab6b2

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ git branch
* (HEAD detached at refs/heads/feature-ui)
  feature-ui
  feature/login
  main
```

9. You're on the main branch and want to start working on a new feature called search-filter. What command do you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git checkout -b search-filter
```

10. You committed a file containing an API key and want to completely remove it from the repo's history. What should you do?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git filter-branch --force --index-filter "git rm --cached --ignore-unmatch apikeys.txt" --prune-empty
 --tag-name-filter cat -- --all
WARNING: git-filter-branch has a glut of gotchas generating mangled history
         rewrites.  Hit Ctrl-C before proceeding to abort, then use an
         alternative filtering tool such as 'git filter-repo'
         (https://github.com/newren/git-filter-repo/) instead.  See the
         filter-branch manual page for more details; to squelch this warning,
         set FILTER_BRANCH_SQUELCH_WARNING=1.
Proceeding with filter-branch...

Rewrite aff33ed2da6ced7eaabc86e7f278b3bc0fb1d40a (5/6) (2 seconds passed, remaining 0 predicted)
WARNING: Ref 'refs/heads/feature-ui' is unchanged
WARNING: Ref 'refs/heads/feature/login' is unchanged
Ref 'refs/heads/main' was rewritten
Ref 'refs/heads/search-filter' was rewritten
WARNING: Ref 'refs/remotes/origin1/feature/login' is unchanged
Ref 'refs/remotes/origin1/main' was rewritten
```

11. You want to see all the branches that exist both locally and on the remote. What command?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch -a
  feature-ui
  feature/login
* main
  search-filter
  remotes/origin1/feature-ui
  remotes/origin1/feature/login
  remotes/origin1/main
  remotes/origin1/search-filter
```

12. You tried to merge two branches and Git reported a conflict in app.js. What are the general steps to resolve it?

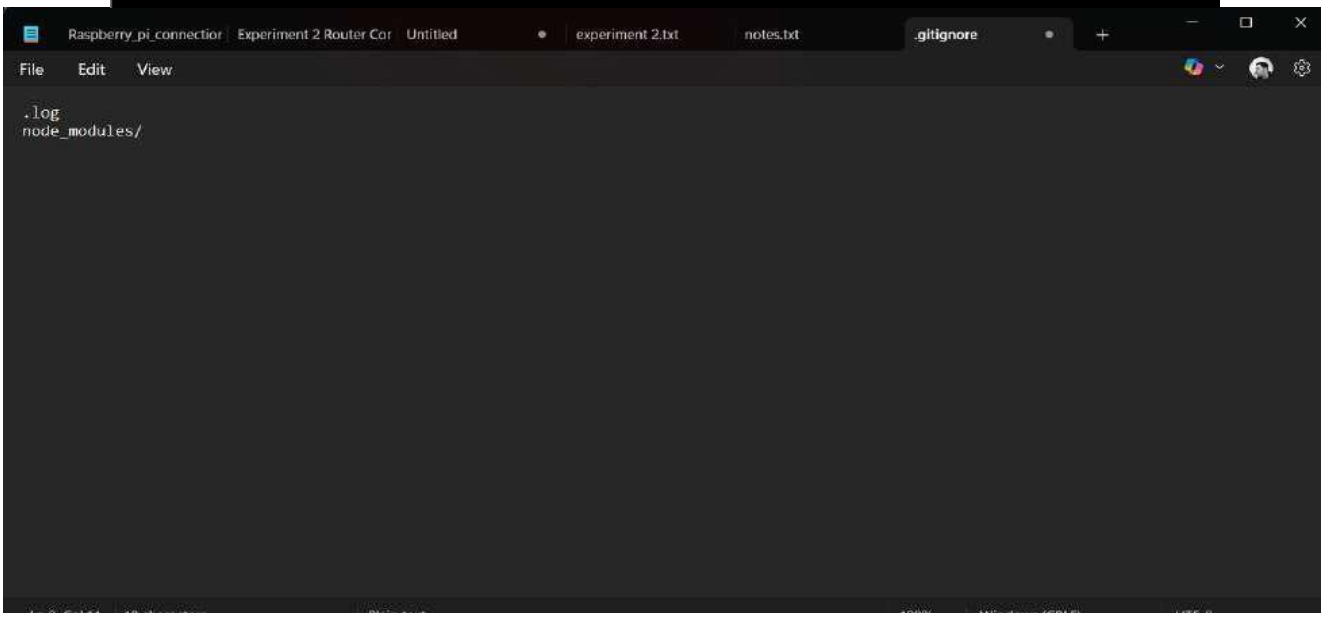
Ans) Open conflicted file(file.txt)

Edit and remove conflict markers(<<<<<<<,=======,>>>>>>>)

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git merge feature/login
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main|MERGING)
$ notepad file.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main|MERGING)
$ git add file.txt

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main|MERGING)
$ git status
On branch main
Your branch is ahead of 'origin1/main' by 1 commit.
  (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)


91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main|MERGING)
$ git commit -m "resolved conflict"
[main e71a73a] resolved conflict
```

13. You don't want Git to track changes to .log files or node_modules/. How do you achieve this?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ touch .gitignore
```



14. You're investigating a bug and want to know who last changed line 25 in script.py. What command do you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git blame file1.txt
b55c5b02 (Macha Harshitha 2025-07-30 12:26:03 +0530 1) hellooo friends!!

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git blame -L 25,25 file1.txt
fatal: file file1.txt has only 1 line
```

15. You're in the middle of working on a file but need to switch branches quickly. What do you do to save your work?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git stash
Saved working directory and index state WIP on main: e71a73a resolved conflict

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git checkout feature/login
Switched to branch 'feature/login'
Your branch is ahead of 'origin1/feature/login' by 1 commit.
  (use "git push" to publish your local commits)
```

16. You previously ran git stash and now want to restore those changes. What command do you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git stash pop
On branch main
Your branch is up to date with 'origin1/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (a55157222955c76fc3b9c1268f8bd6cd4061ac1b)
```

17. The feature/test branch is no longer needed. How do you delete it locally?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature/test)
$ git checkout -d feature/test
M       file1.txt
HEAD is now at e71a73a resolved conflict
```

18. You created a branch `feature-experiment`, made some changes, but now realize the code is no longer needed. You want to delete it, even though it hasn't been merged. What command will you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch -D feature-experiment
Deleted branch feature-experiment (was e71a73a).
```

19. You want to delete `bugfix-footer`, but you're unsure if it's been merged. What should you do before deleting it?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch --merged
  feature/login
  feature/signup
  feature/test
* main
  search-filter
```

Scenario based questions on working with remote repository using Git commands

1. Specify the git command when you're starting work on a project and need to clone a remote repository to your local machine.

```
$ git clone https://github.com/RitiClub/Riti-backend.git
Cloning into 'Riti-backend'...
remote: Enumerating objects: 308, done.
remote: Counting objects: 100% (308/308), done.
remote: Compressing objects: 100% (252/252), done.
remote: Total 308 (delta 72), reused 274 (delta 53), pack-reused 0
Receiving objects: 100% (308/308), 1.05 MiB | 9.24 MiB/s, done.
Resolving deltas: 100% (72/72), done.
```

2. Specify the git command if you want to see the remotes that are connected to your local repository.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote -v
origin1 https://github.com/Harshitha-Macha/harshitha2907.git (fetch)
origin1 https://github.com/Harshitha-Macha/harshitha2907.git (push)
```

3. Specify the git command if you want to add a new remote repository to your local repository.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote add origin1 https://github.com/Harshitha-Macha/harshitha2907.git
error: remote origin1 already exists.
```

4. Specify the git command to remove a remote repository from your local configuration:

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote remove origin1
```

5. Specify the git command if you want to rename an existing remote:

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote rename origin harshitha
```

6. Specify the git command to push your local commits to a remote repository.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git push origin main
Everything up-to-date
```

7. Specify the git command to when pushing for the first time and want to set the remote branch you are pushing to.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git push --set-upstream origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date
```

8. Specify the git command to change the URL of a remote (e.g., after changing the remote repository address).

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote set-url origin https://github.com/Harshitha-Macha/Tabulax.git
```

9. Specify the git command if you want to list all branches on the remote repository.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch -r
  origin/feature-ui
  origin/feature/login
  origin/main
  origin/search-filter
```

10. Specify the git command if a branch was deleted on the remote but still shows up locally.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git fetch -p
```

11. Specify the git command to fetch a specific branch from a remote.

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git fetch origin feature-ui
From https://github.com/Harshitha-Macha/harshitha2907
 * branch            feature-ui -> FETCH_HEAD
```

12. Specify the git command if you want to rebase your local branch onto a remote branch (this can be useful to keep your history linear)

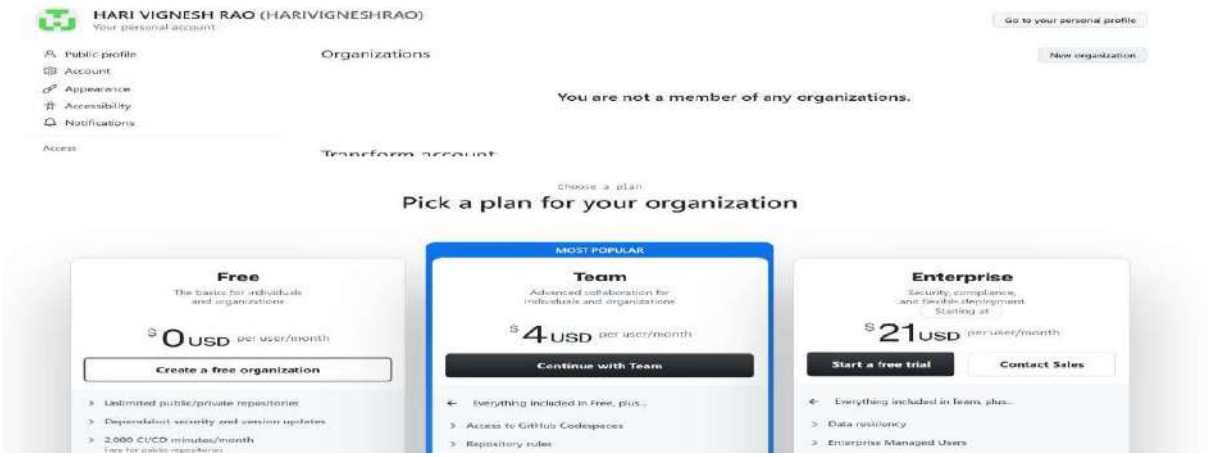
```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git fetch origin

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git rebase origin/main
Current branch main is up to date.
```

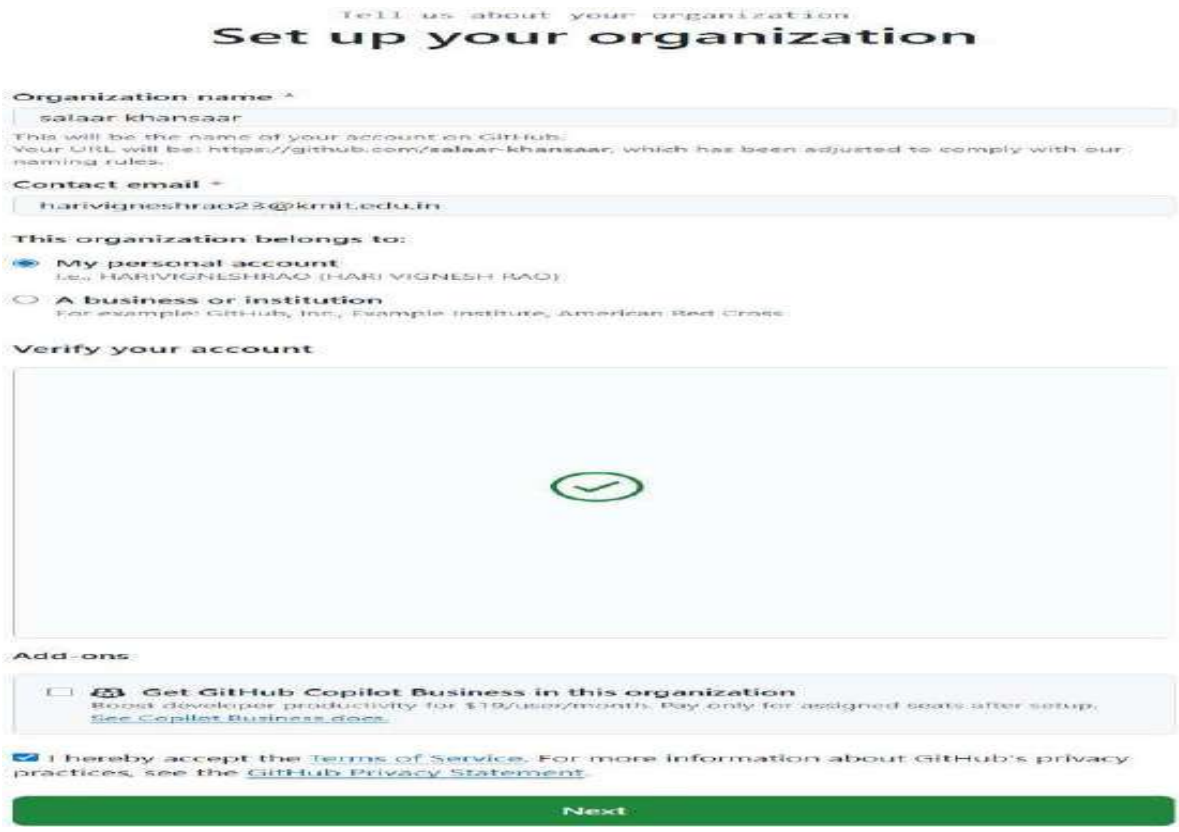
Experiment-3 : Collaborative coding using git

a. To work on collaborative coding by:
b. Creating Organization.

Organization - Organizations are shared accounts where businesses and open-source projects can collaborate across many projects at once. Steps to facilitate collaborative work by creating organization are:

Step 1: Click on New organization in Git hub. lick on Create a free organization.





Step 2: Set up your organization by entering the details like name, associated email, account

verification, etc. Click on Next

Step 3: Complete the setup by

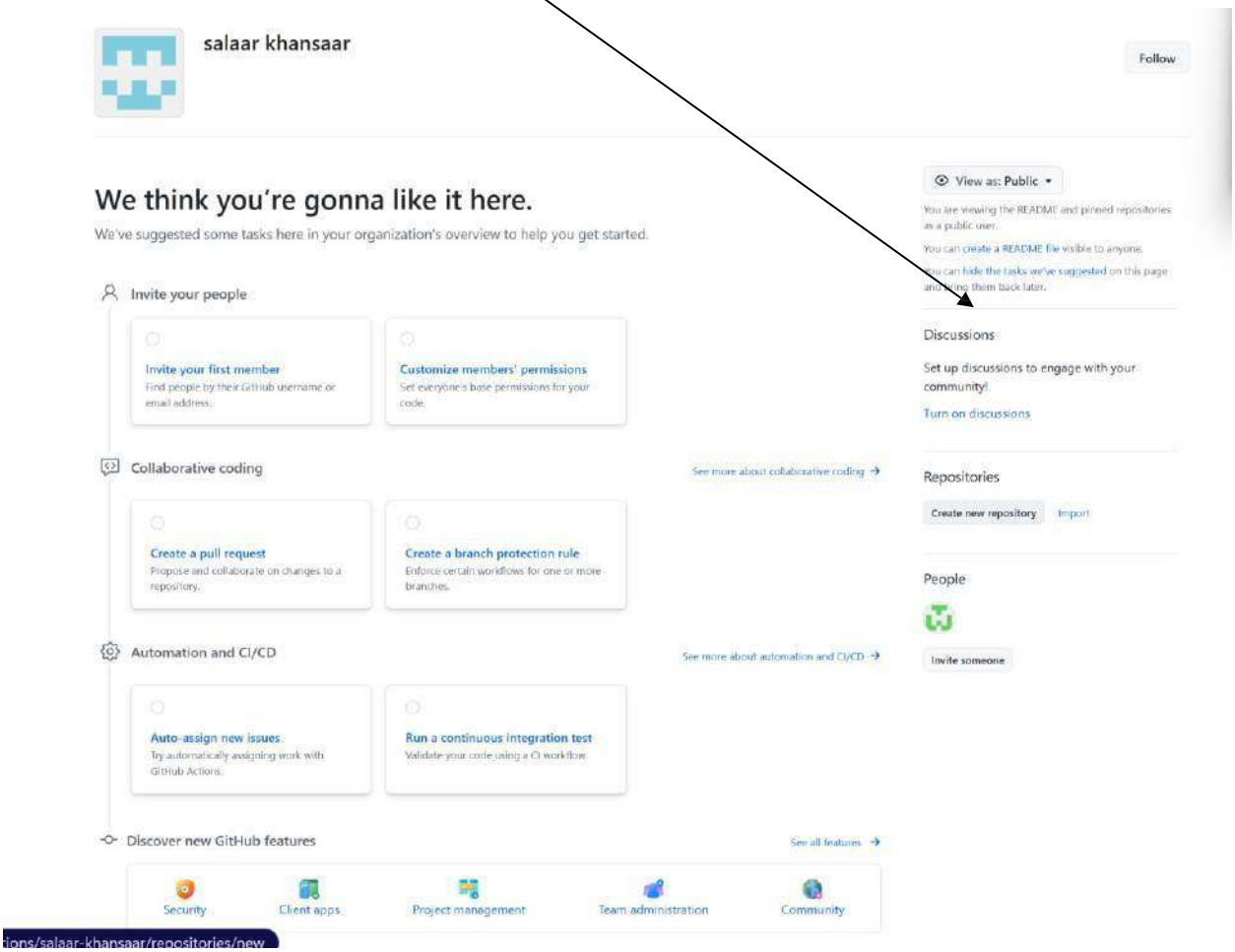
- ✓ adding members to the group.
- ✓ Now, the invitation goes to other collaborator and they must accept.
- ✓ Next click on complete setup
- ✓ Goto Github and click on

Settings → Member privileges → Base permissions → select write → change base permission to "Write"

Next under,

Projects base permissions →select Write → change base permission to "Write"

Step 4: Create the remote repository for storing the project related files in Organization. This repository is accessible to every member of the team as per the permissions given.



Note: The repository can be made private so that it is accessible only to the group members rather than being in a public domain.
Now all the members of the team can contribute to the development of the project and the different files with all the versions and modification notices will be available in the respective repositories and is accessible to all the members

c. Coordinating with others through a shared repository

Steps for Collaborator 1 are: (collaborator-1 has repository that he wants to share with collaborator-2)

- Go to Settings > collaborators
- Now in Manage Access click on Add people
- Now you will be able to see ONE collaborator added
- After this invitation goes to collaborator 2 and they need to accept it.

Steps for Collaborator 2 are:

1. Set Up Git and GitHub
In git bash

git config --global user.name "Your Name"
git config --global user.email you@example.com

Note : run $ git remote -v
If origin git@github.com:OrgYOG/commonREPO.git (fetch)
origin git@github.com:OrgYOG/commonREPO.git (push)
then

$ git remote –v // Note now, not connected to any remote

2. Goto Git hub and copy url of shared repository by collaborator 1

- Copy url (option 1)

3. Clone the Repository
Get a local copy of the repo:

In git bash
git clone https://github.com/username/repository-name.git
cd repository-name

4. Create a Branch for Your Work
Always work on a separate branch:
In git bash

git checkout -b feature/your-feature-name

5. Make Changes and Commit
Edit files, then stage and commit your changes:
In git bash to existing file / new file

git add .
git commit -m "Add a clear and descriptive commit message"

6. Push Your Branch to GitHub

➔ In git bash

➔ git push origin feature/your-feature-name

7. Keep Your Branch Updated (has to be done by owner collaborator 1)

Before making new changes:

In git bash

```
git checkout main
git pull origin main
git checkout feature/your-new-feature
git merge main
```

Exercise on Fork

- If you're contributing to an existing project: (in public repository say of Person1)
- Note: Person 1means other public repository already exit that is pushed by person 1. Person 2 is you who want to contribute.

Note: Following steps needed if you forked from any public repository to contribute to Person 1

1. Go to: `https://github.com/Person1/awesome-project` (of Person 1)
2. Click the "Fork" button at the top-right of the page.
3. GitHub will create a copy of that repository in your account (say Person2):

https://github.com/your-username/awesome-project

as

https://github.com/Person2/awesome-project

4. Clone your forked repo: in bash (by Person 2)
   - git clone https://github.com/your-username/awesome-project
   - Make changes.
   - git add .
   - git Commit
   - git push origin main

1. Create or a "pull request" back to the original repo (`othername/awesome-project) to suggest your changes. (by Person 2)

   - Click at top "Pull request" or click on at top or click on repo "awesome-project" then click on Contribute at top then click on "open pull request"
   - Add title and Add a description if you want. Below it shows what changes you made to file
   - Click on Create pull request

Now you see on page message "NO conflicts with base branch"

6. By Person 1
   - Pull requests 1 appears at top in git hub account of Person 1
   - Clicking on "Pull requests" shows:

Commit by Person 2. Click on this commit message.

It redirect to repository (say awesome-project)

- Click on “File changed 1” to see changes made by Person 2
- Click on “Review Changes”, add review and click on “submit review”
- Click on Merge pull request
- Confirm merge

Note: It displays “Pull request Successfully merged and closed” to Person 1

- Now go to Code and see changes have been merged to the file.
- In commit it shows: Commit by Person 2
  Merge commit by Person 1

7. Merge the Pull Request (has to be done by both collaborators)
Once approved:
- The repo owner can click Merge pull request

8. Keep Your Branch Updated (has to be done by Person 1 and Person 2)
Before making new changes:
In git bash

```
git checkout main
git pull origin main
git checkout feature/your-new-feature
git merge main
cd repository-name
```

Steps to Resolve a Conflict
1. See Which Files Are in Conflict (Say f1.txt)
In git bash
git status
Conflicted files will be marked like:
In git bash
both modified: (Say f1.txt)

1. **Open the File and Look for Conflict Markers by clicking on**

Click on **Resolve Conflict**

Git will insert conflict markers in the file like this:

```
def greet():
<<<<<<< HEAD
   print("Hello from First collaborator ")
=======
   print("Hello from second collaborator ")
>>>>>>> feature/second collaborator branch
```

This shows the two conflicting changes:
- HEAD is your version
- The section below ======= is the other branch’s version or second collaborator branch version

3. Mark the Conflict as Resolved

Once you've edited and saved the file:

- ✓ git add f1.txt

4. Complete the Merge or Pull

- ✓ git commit # Only needed if you're merging

Or if you're rebasing:

- ✓ git rebase --continue

If You Want to Abort the Merge

If it gets messy and you want to cancel:
git merge --abort

Here below screen shot shows conflict raised when collaborator 1 wanted to merge changes to same line that already collaborator 2 added the line of code at that line in same file but in their own branch.

Tips to Avoid Conflicts

- Pull the latest changes before starting work:
  In git bash
  git pull origin main
-
- Communicate with your team
- Work in small branches with short-lived changes

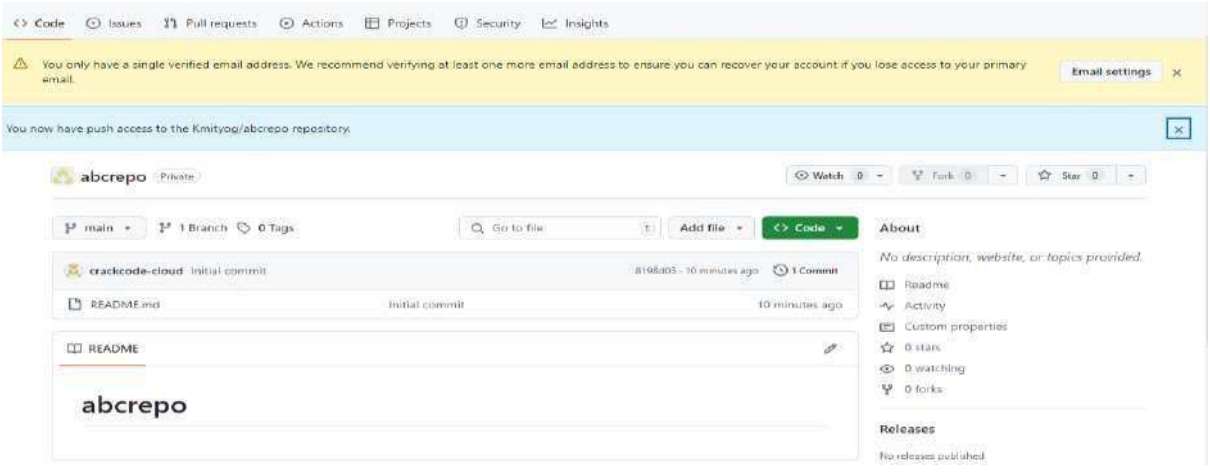
Example :

Merge Conflict – Changes made to a file by two different users leads to merge conflict as below.

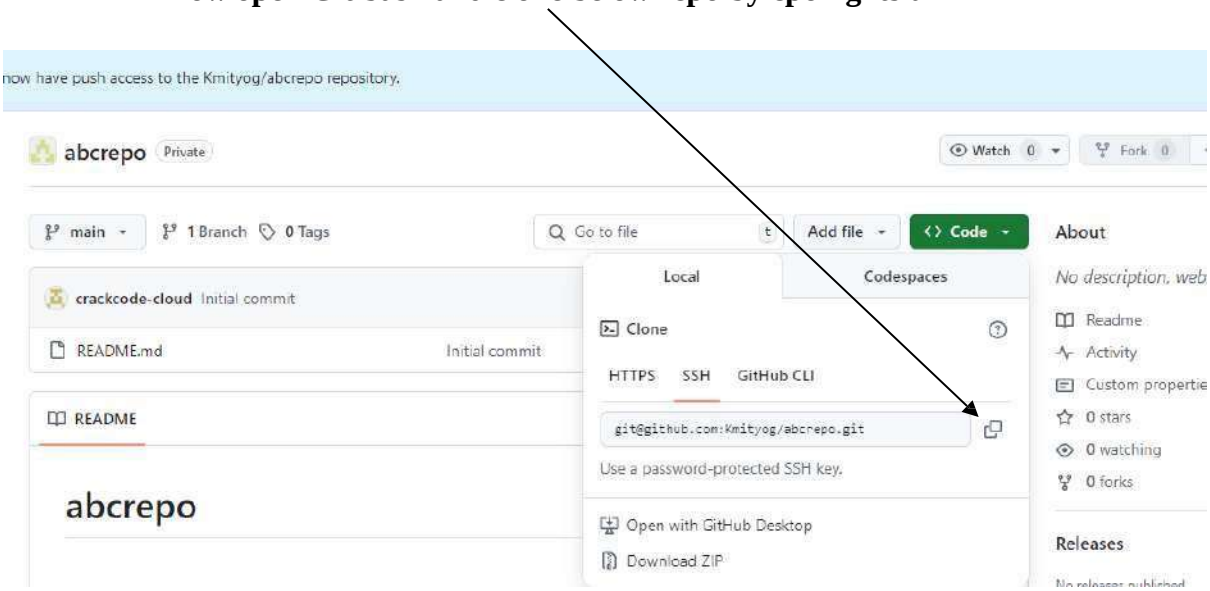
I am collaborator 2, now

➔ **Accept invitation from collaborator 1**

➔ **Next goes to abcrepo**



- ✓ **Now open Git bash and clone below repo by cpoing its url**



Step 1: first connect to this abcrepo to local repository by clone

```
User@Dell3542 MINGW64 ~ (master)
$ git clone git@github.com:Kmityog/abcrepo.git
Cloning into 'abcrepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

User@Dell3542 MINGW64 ~ (master)
```

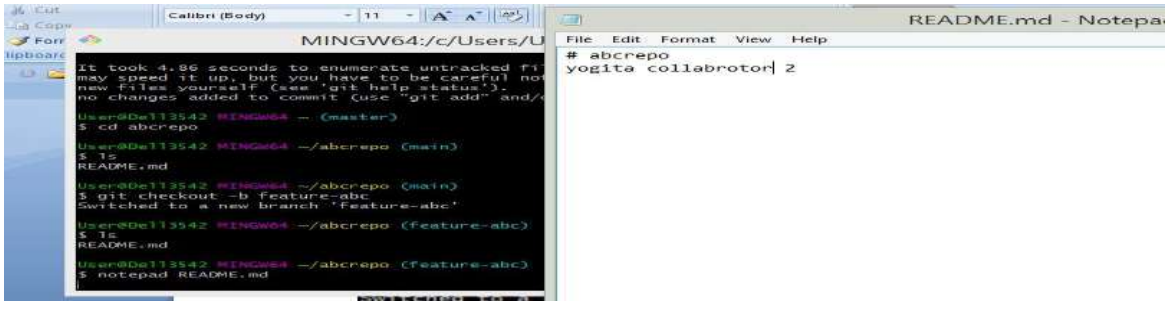
- ✓ **Now change directory to abcrepo**

```
User@Dell3542 MINGW64 ~ (master)
$ cd abcrepo

User@Dell3542 MINGW64 ~/abcrepo (main)
$ ls
README.md

User@Dell3542 MINGW64 ~/abcrepo (main)
$
```

- ✓ **Switch to branch feature-abc**

```
User@Dell3542 MINGW64 ~/abcrepo (main)
$ git checkout -b feature-abc
Switched to a new branch 'feature-abc'

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ ls
README.md

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$
```

- ✓ **Now add line in README.md file**



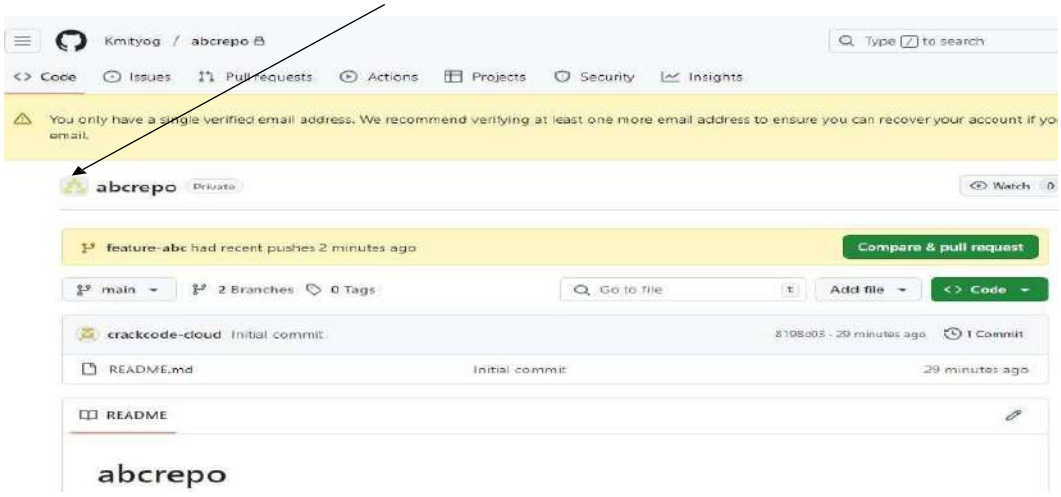
✓ **Now git add, commit**

```
User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git status
On branch feature-abc
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git add .

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git commit -m "1st commit"
[feature-abc aadcb9b] 1st commit
 1 file changed, 2 insertions(+), 1 deletion(-)

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$
```

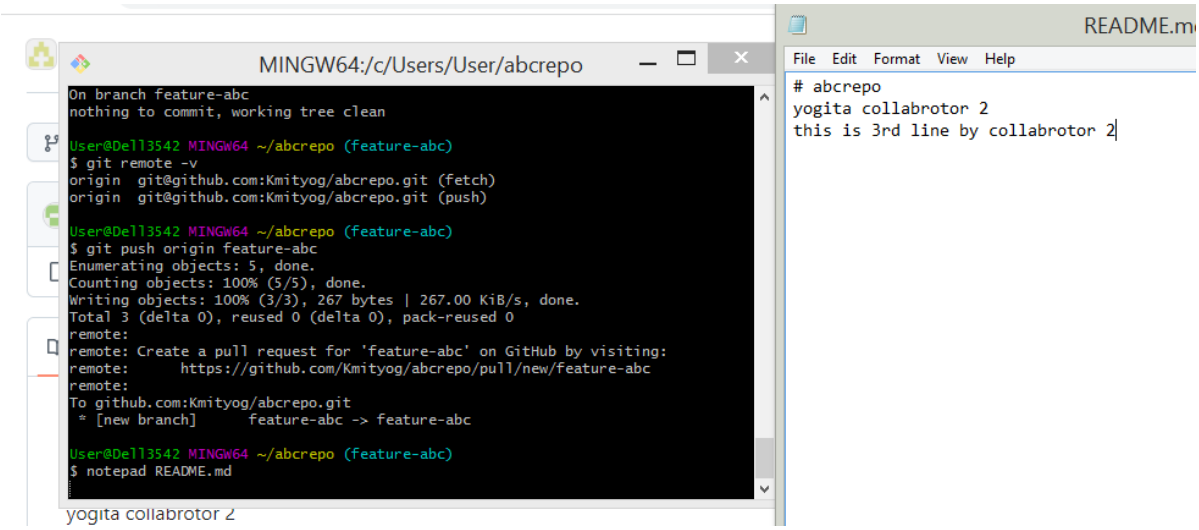
✓ **Now see status and push to remote branch feature-abc**

```
User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git status
On branch feature-abc
nothing to commit, working tree clean

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git remote -v
origin  git@github.com:Kmityog/abcrepo.git (fetch)
origin  git@github.com:Kmityog/abcrepo.git (push)

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git push origin feature-abc
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 267 bytes | 267.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature-abc' on GitHub by visiting:
remote:      https://github.com/Kmityog/abcrepo/pull/new/feature-abc
remote:
To github.com:Kmityog/abcrepo.git
 * [new branch]      feature-abc -> feature-abc

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$
```

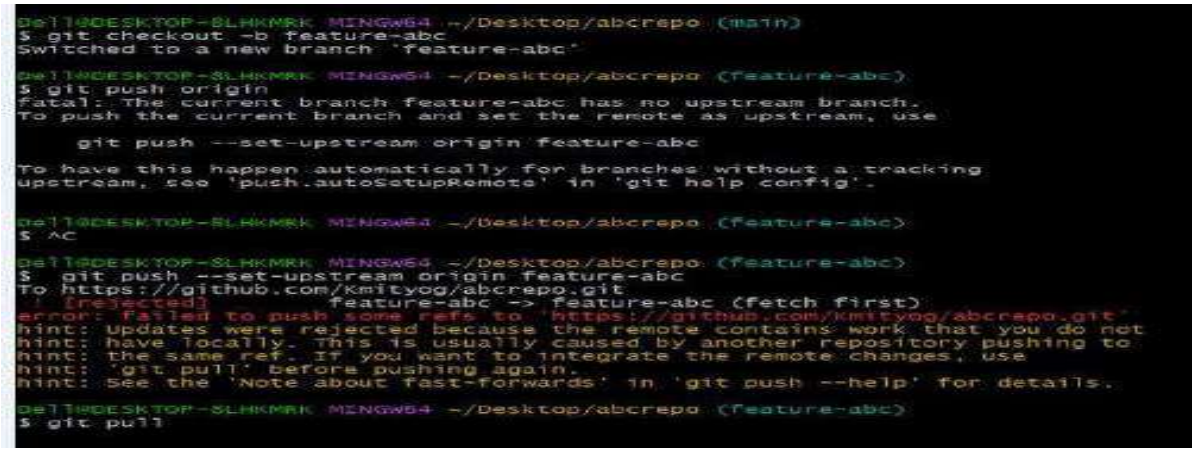
✓ **Now go to Github and refresh or click on abcrepo, you can see**



Step 2: Now collaborator 2 adding line in README.md, save, git add, commit, push. **No conflicts.**



Step 3: Also collaborator 1 adding line in README.md at same line collaborator 2 added, then git add, commit, push gives **merge conflict.**



Now open file, Edit the File to Fix the Conflict by keeping either or both collaborator changes and remove conflict markers. Add file, commit, and push.

d. To create and apply patch.

A patch in Git is a text file that represents changes between two sets of files, or commits. It's essentially the output of the git diff command, packaged in a format that can be applied to another set of files.

Steps:

1. Goto github and take public project url
2. Clone into git bash
3. Now open file and edit it, add, commit

```
User@Dell3542 MINGW64 ~/my_project/Desktop (master)
$ cd crack

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ ls
cracku.txt

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ notepad cracku.txt

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ git add .

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ git commit -m "commit from cloned person"
[master 9b3943a] commit from cloned person
 1 file changed, 2 insertions(+), 1 deletion(-)

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$
```

4. Run git log

```
User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ git log
commit 9b3943a9d67bf90a15800e8c396b48d49c564339 (HEAD -> master)
Author: Yogita <agyogita@gmail.com>
Date:   Thu Jul 31 02:15:11 2025 +0530

    commit from cloned person

commit 686b0a755049b4321e993912a182cee4d2226aa7 (origin/master, origin/HEAD)
Author: shradha-2022 <shradhacg@gmail.com>
Date:   Wed Jul 30 09:44:59 2025 +0530

    first

commit d0795ccd0f03fc53097b7a80bef7a396a046526e
Author: crackcode-cloud <shradhacg@gmail.com>
Date:   Wed Jul 30 09:30:43 2025 +0530

    firat commit

User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
```

5. Create patch with commit hash code as:

   git format-patch -1 commit-hash-code

```
User@Dell3542 MINGW64 ~/my_project/Desktop/crack (master)
$ git format-patch -1 9b3943a9d67bf90a15800e8c396b48d49c564339
0001-commit-from-cloned-person.patch
```

This creates 0001-commit-from-cloned-person.patch file.

- Example:

  git format-patch -1 abc1234
- This creates a file like `0001-Your-commit-message.patch`

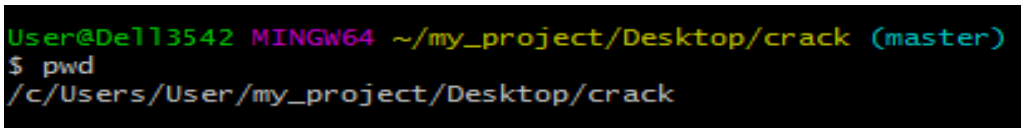
To create patches for the **last 3 commits**: `git format-patch -3`

or

Create patches from multiple commits

git format-patch <base_commit>..HEAD

6. Get the path of above patch file and email or what's up to the remote owner of file.



7. Once the remote owner gets the patch file 0001-commit-from-cloned-person.patch next the owner need to apply patch file in his git bash using git command below:

Syntax:

git apply my-changes.patch

Or, if it's a patch made by `git format-patch`, use:

git am 0001-Your-commit-message.patch

- ✓ git apply 0001-Your-commit-message.patch

Experiment-4: Build and package Java and Web applications using Maven

a. Understand the structure and lifecycle of a Maven project.

Maven standardizes the project structure for both Java and web-based applications. src/

├── main/

│ ├── java/ → Application source code

│ └── resources/ → Configuration files like config.properties

└── test/

├── java/ → Unit test source code

└── resources/ → Test resources

The compiled files and reports are generated in the **target/** directory after a successful build.

b. Build and package Java and Web applications using Maven.

----mvn clean install

- clean: Deletes the previous build (target/ folder)
- install: Builds, tests, and installs the package to local .m2 repository After execution:
- target/ folder contains compiled .class files and the final .jar or .war
- Artifact is stored in:
  ~/.m2/repository/groupId/artifactId/version/

Creation of Maven Java Project

Step 1. Open Eclipse IDE

└── 1.1. Launch Eclipse workspace

Step 2. Install Maven Plugin (if not installed)

└── 2.1. Go to "Help" in the top menu

└── 2.1.1. Click "Eclipse Marketplace"

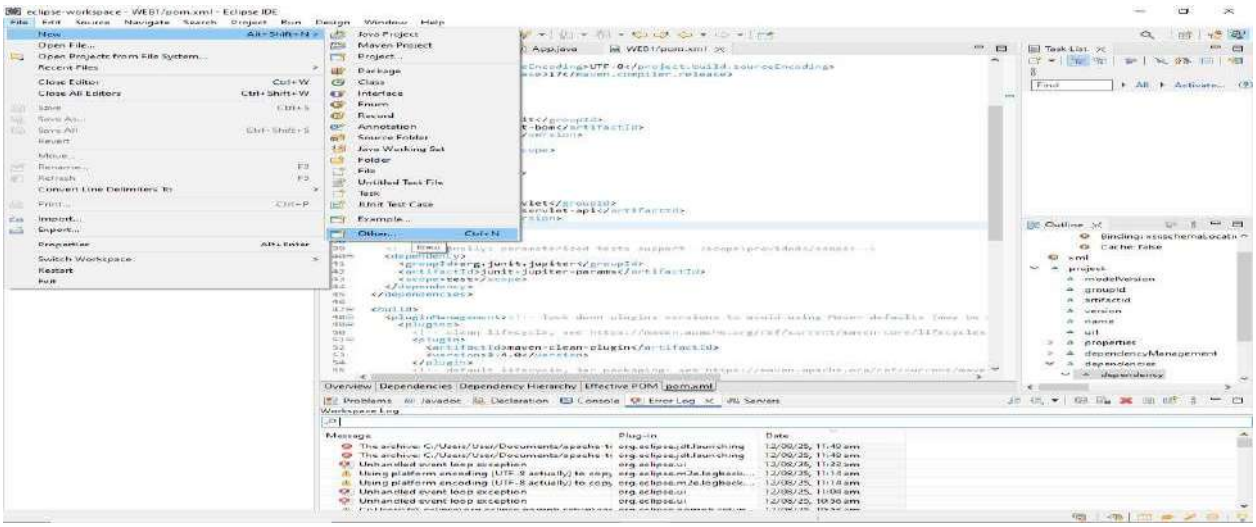
└── **2.1.2. Search for "Maven Integration for Eclipse"**

└── **2.1.3. Install the plugin if not already installed**

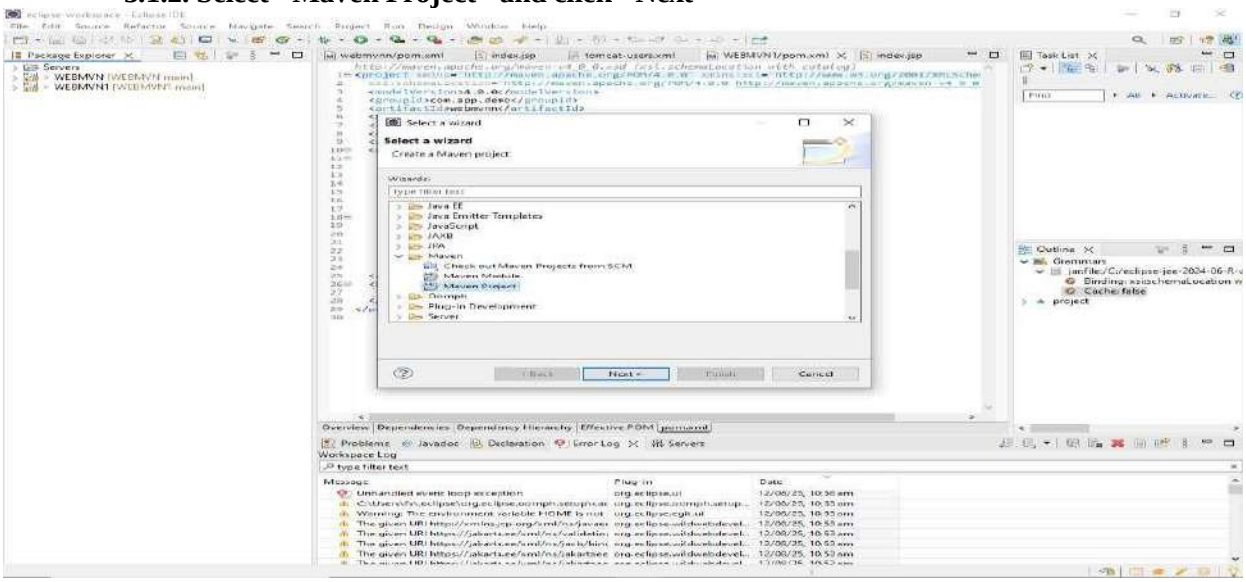
Step 3. Create a New Maven Project

└── **3.1. File -> New -> Project...**

└── **3.1.1. Expand "Maven"**



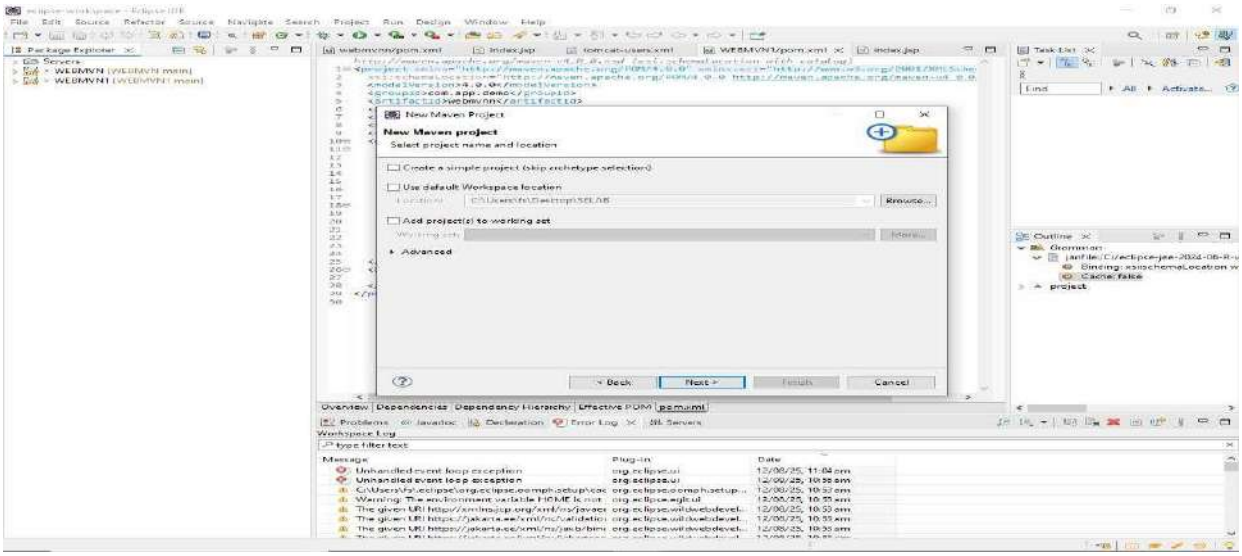
└── **3.1.2. Select "Maven Project" and click "Next"**



Step 4. Set Project Configuration

└── **4.1. Select workspace location (default or custom)**

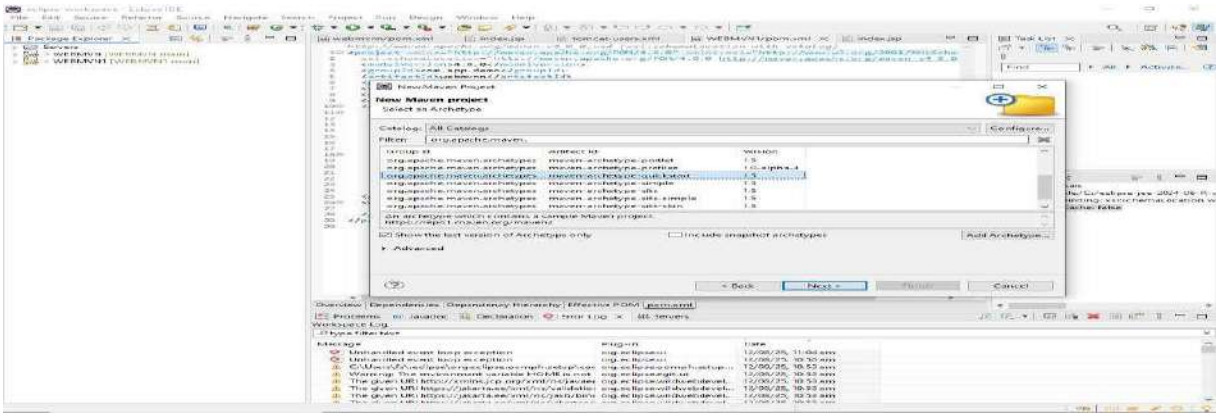
└── **4.2. Click "Next"**



Step 5. Choose Maven Archetype

└── 5.1. Select an archetype(e.g "org.apache.maven.archetypes -> maven-archetype-quickstart 1.4 ")

└── 5.2. Click "Next"



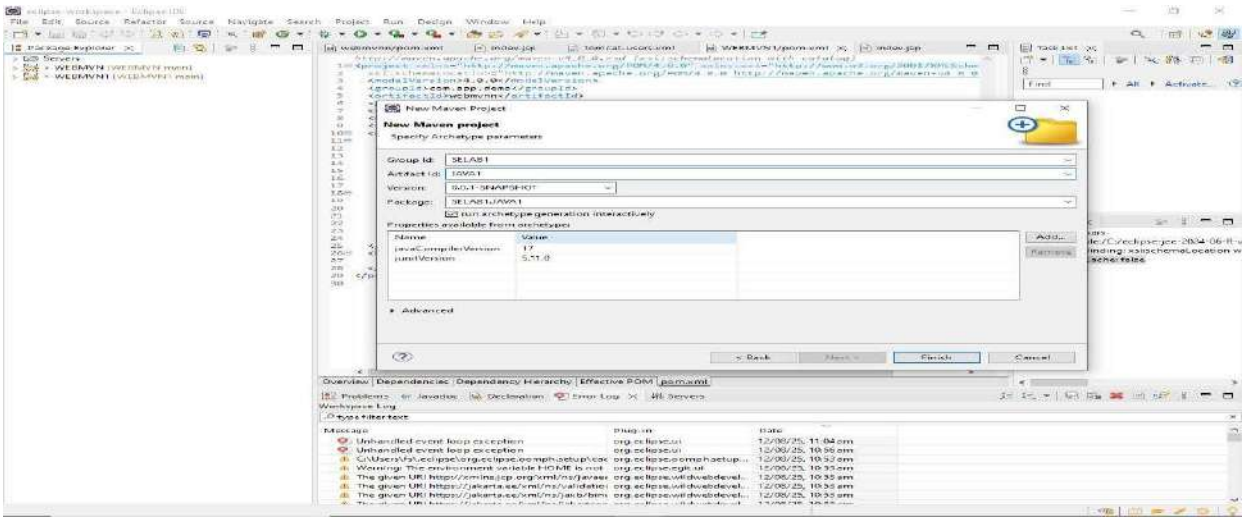
Step 6. Define Project Metadata

└── 6.1. Group ID: (e.g., com.example)

└── 6.2. Artifact ID: (e.g., my-maven-project)

└── 6.3. Version: (default is usually fine)

└── 6.4. Click "Finish"



In Console, artifacts are grouped. When prompted with Y/N, type 'Y'.

Step 7. Maven Project Created

└── 7.1. Project structure is generated with a standard Maven layout

└── 7.2. Includes:

└── src/main/java (for Java source code)

└── src/test/java (for test code)

└── pom.xml (Maven configuration file)

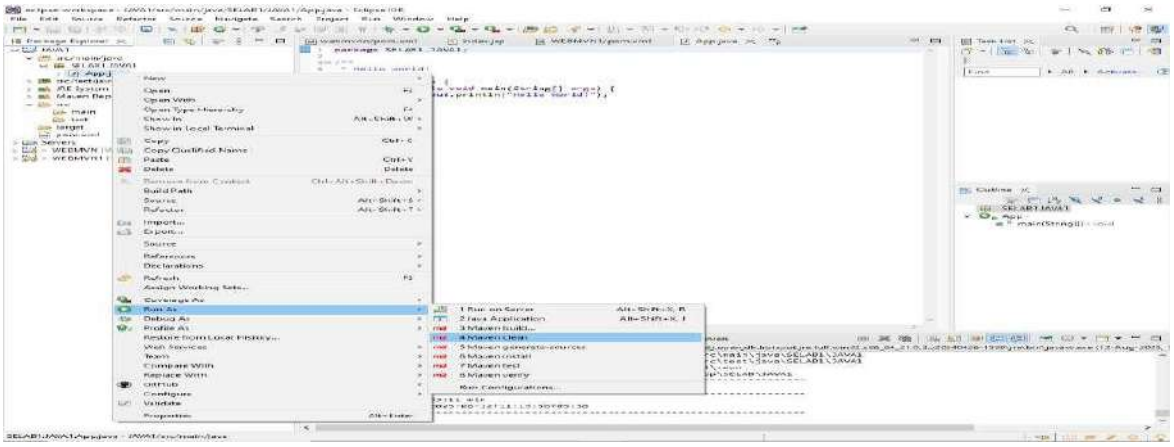
Step 8. Update Project Settings (if needed)

└── 8.1. Right-click on the project -> Maven -> Update Project...

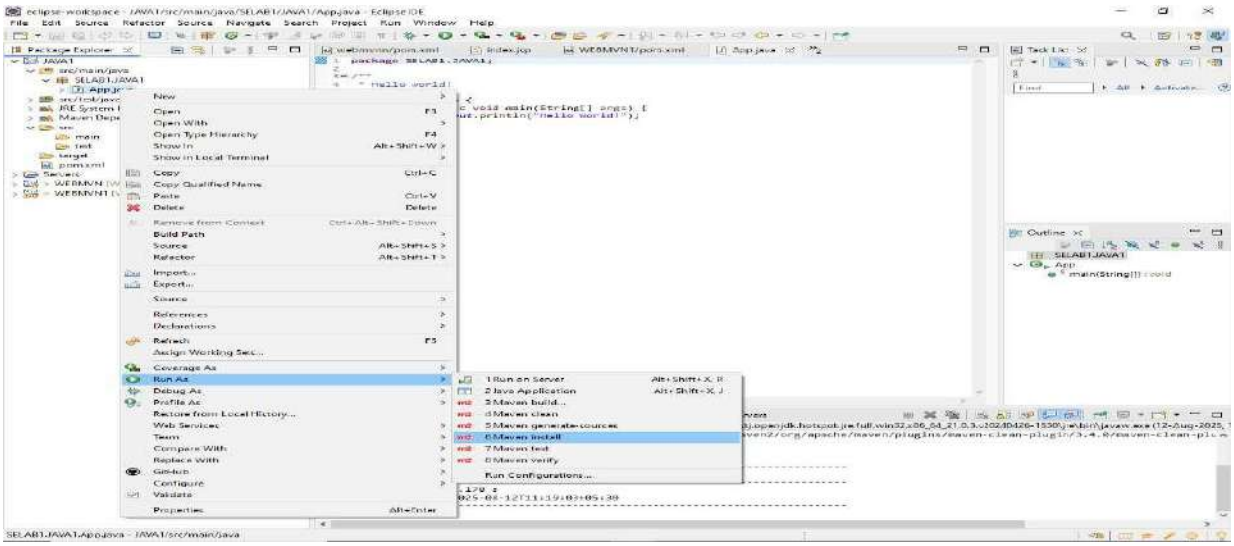
└── 8.2. Ensure dependencies are up to date

Step 9. Build and Run Maven Project

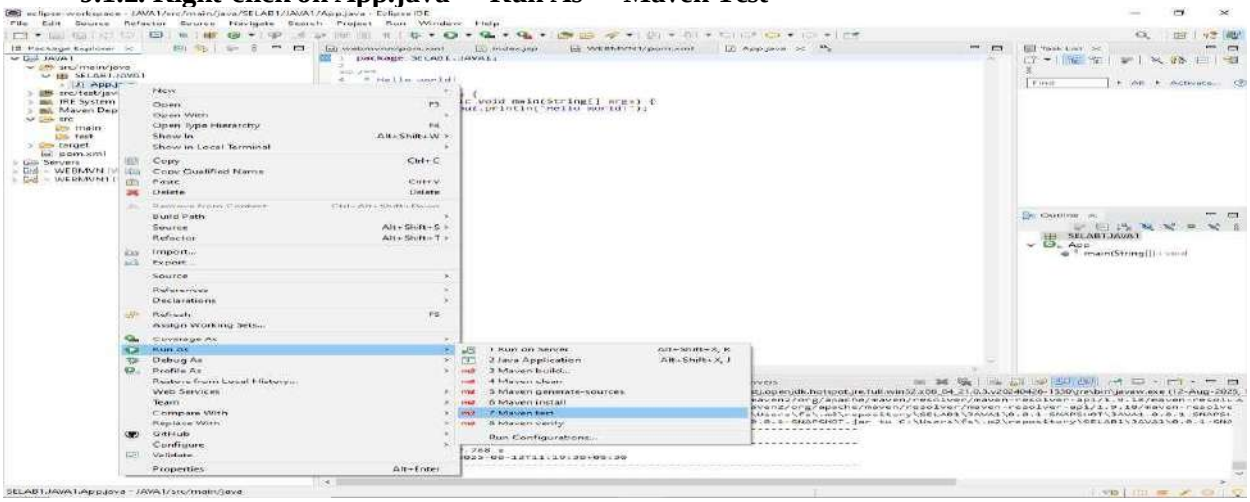
└── 9.1. Right-click on App.java -> Run As -> Maven Clean



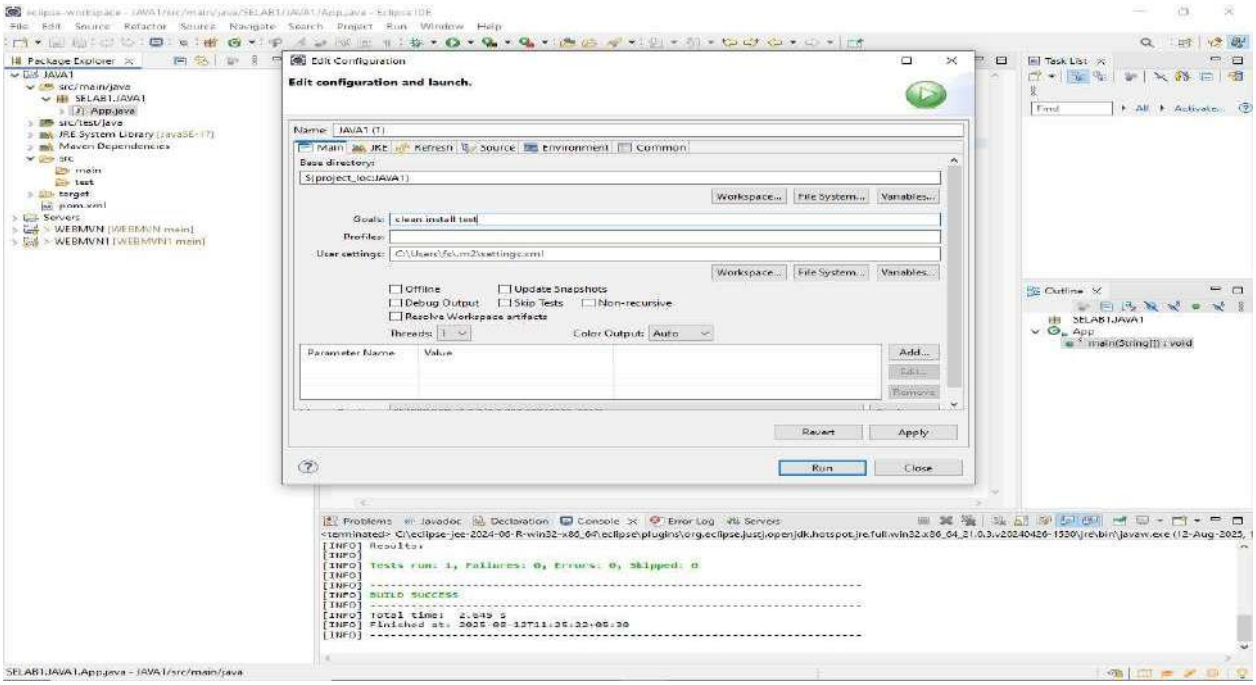
└── **9.1.1. Right-click on App.java -> Run As -> Maven Install**



└── **9.1.2. Right-click on App.java -> Run As -> Maven Test**



└── **9.1.3. Right-click on App.java -> Run As -> Maven Build**



Step 10. In the Maven Build dialog:

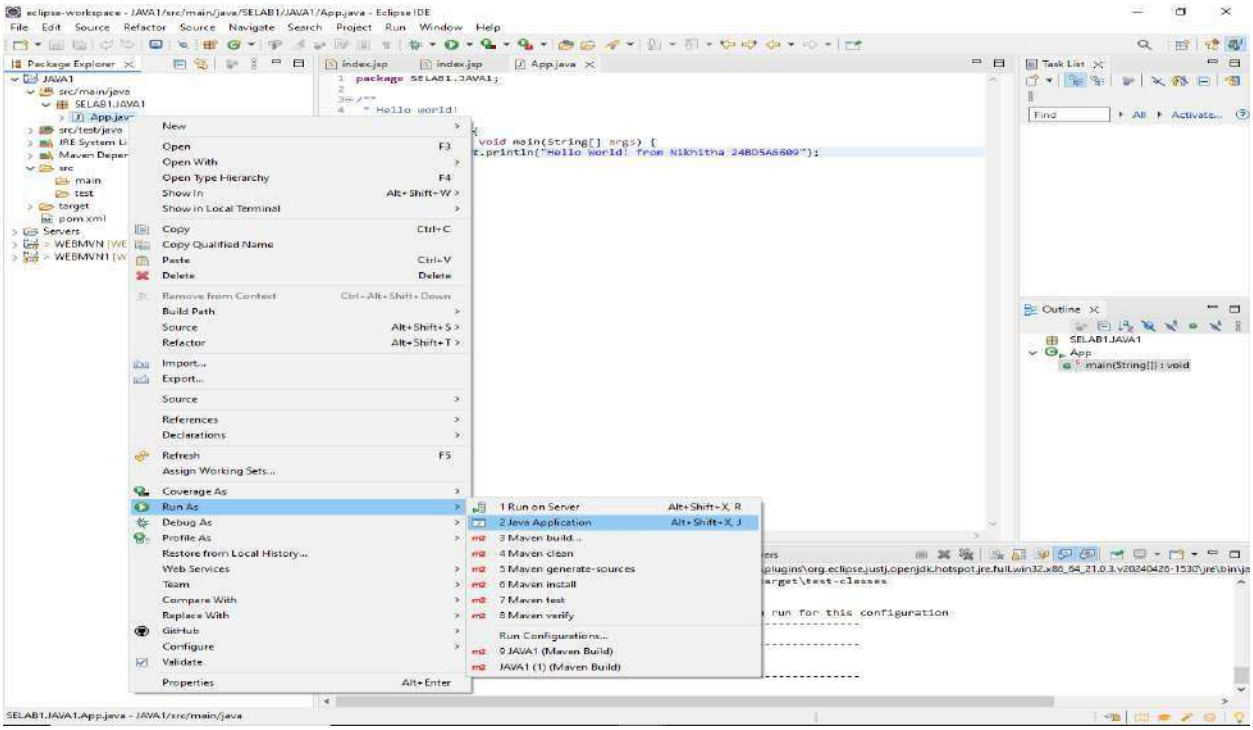
└── Enter Goals: clean install test

└── Click on Apply -> Click on Run

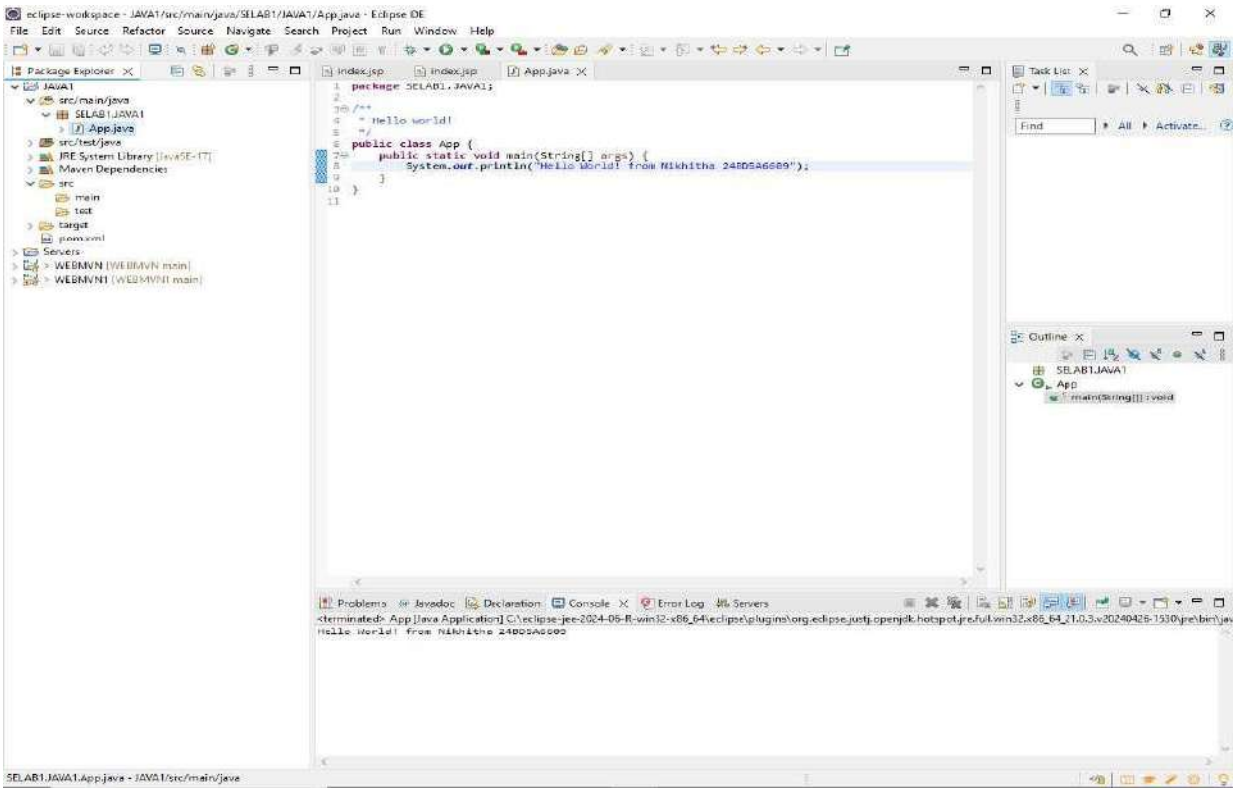
Step 11. Check console for BUILD SUCCESS message.

Step 12. Run the application:

└── Right-click on App.java -> Run As -> Java Application



└── **Output: "Hello World" displayed.**



Creation of Maven web Java Project

Step 1: Open Eclipse

└── 1.1 Launch Eclipse IDE.

└── 1.2 Select or create a workspace.

Step 2: Create a New Maven Project

└── 2.1. File -> New -> Project...

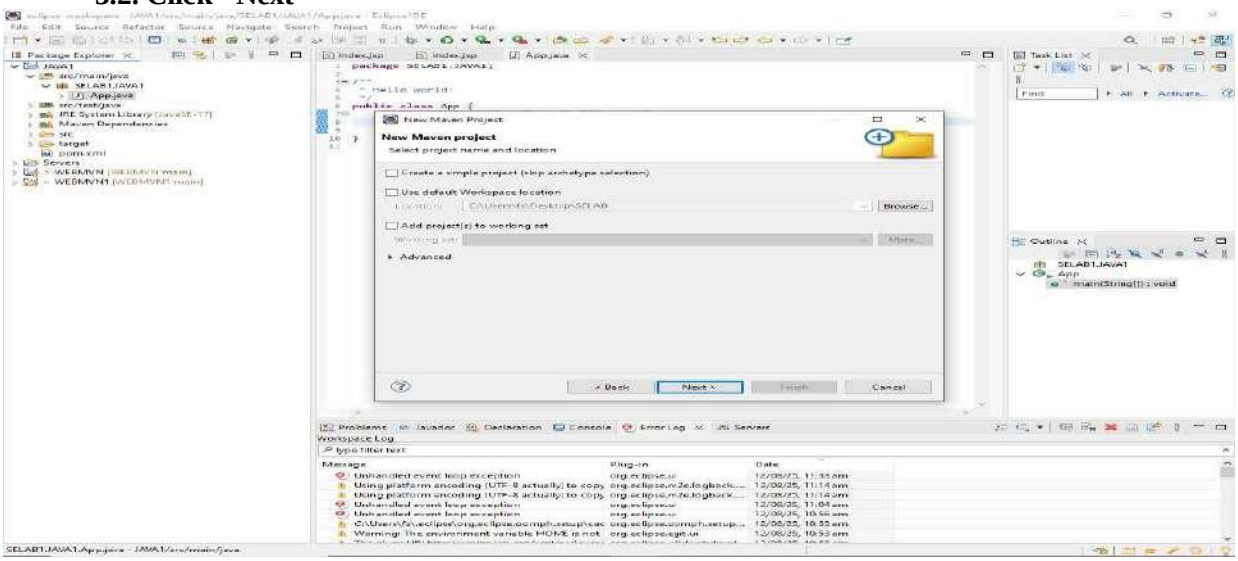
└── 2.1.1. Expand "Maven"

└── 2.1.2. Select "Maven Project" and click "Next"

Step 3: Choose Maven Archetype

└── 3.1. Select an archetype(e.g "'org.apache.maven.archetypes' -> 'maven-archetype-webapp' 1.4 ")

└── 3.2. Click "Next"

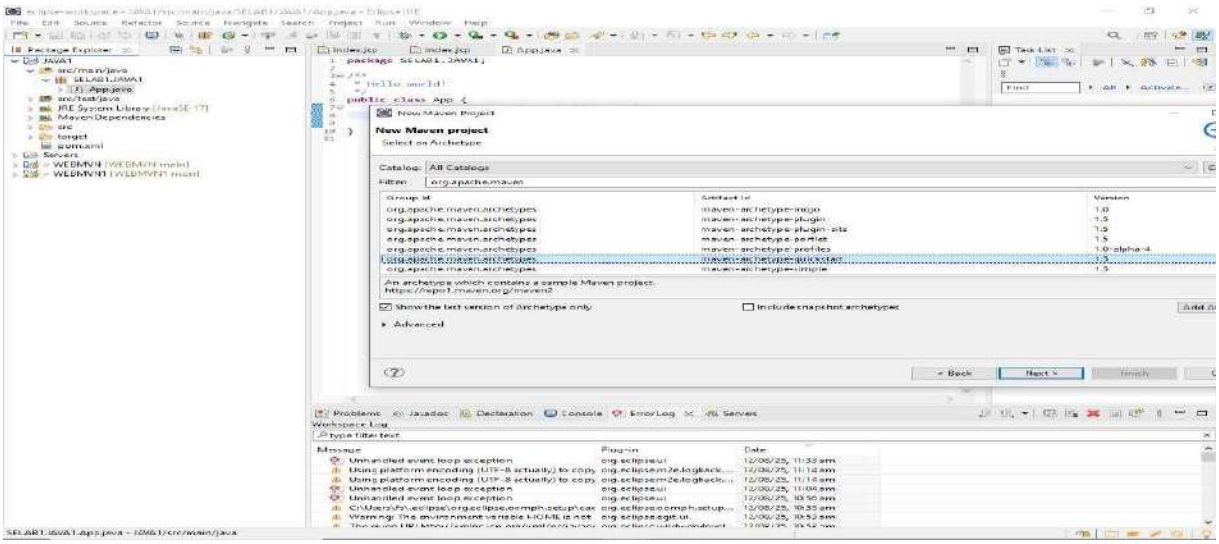


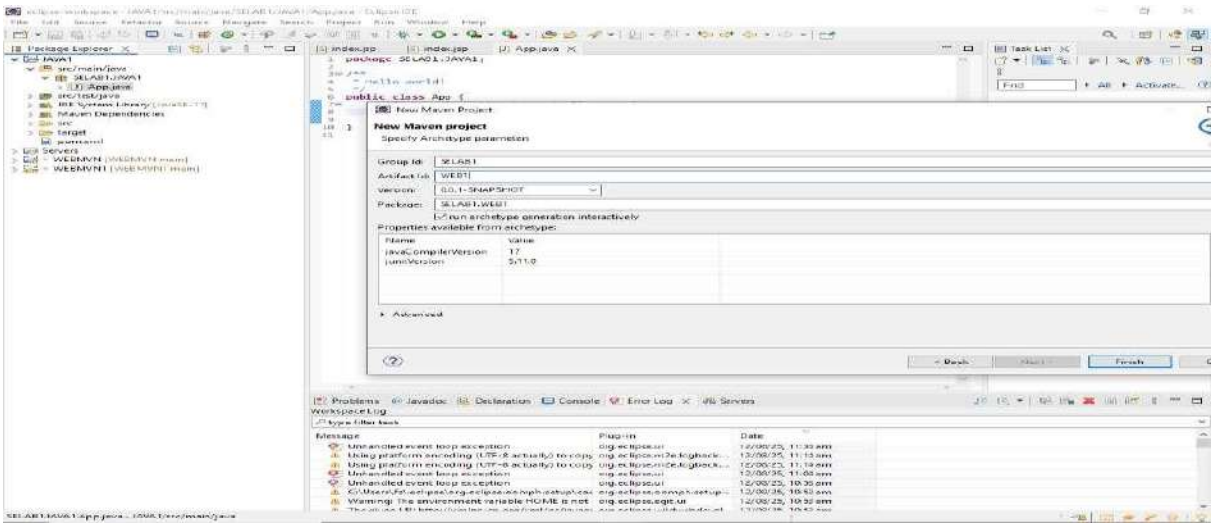
Step 4: Configure the Maven Project

└── 4.1 Group Id: Enter a group ID (e.g., com.example).

└── 4.2 Artifact Id: Enter an artifact ID (e.g., my-web-app).

└── 4.3 Click **Finish to create the project.**

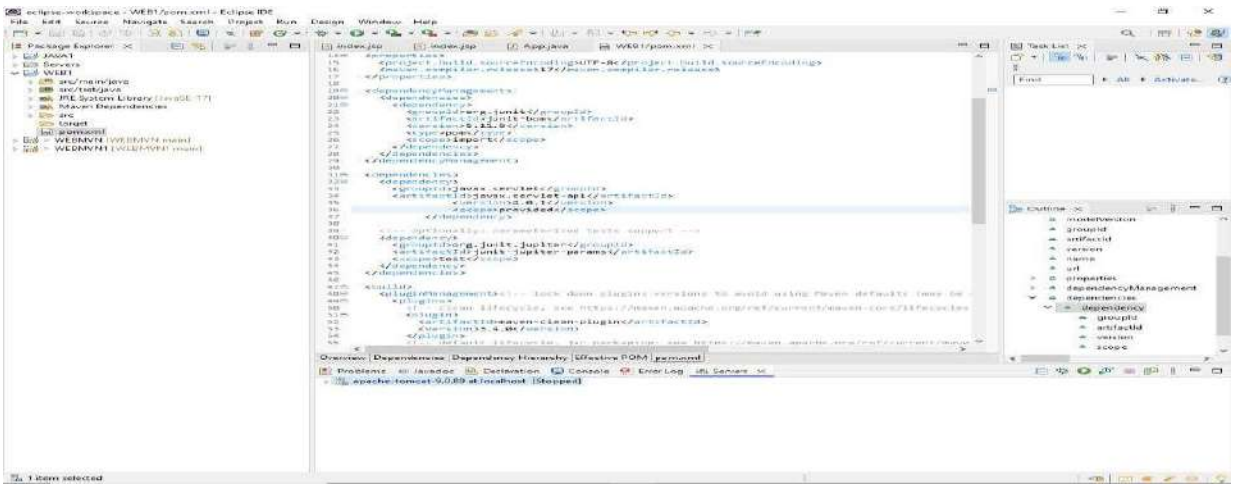




Step 5: Add Maven Dependencies

└── 5.1 Open the **pom.xml file in the Maven project.**

└── 5.2 Add the necessary dependencies for your web project (e.g., Servlet, JSP):



Go to browser -> Open mvnrepository.com

Search for 'Java Servlet API' -> Select the latest version.

Copy the dependency code -> Paste it in MavenWeb's pom.xml under the target folder

Step 6:-. Configure server:

└── Window -> Show View -> Servers

└── Add server -> Select Tomcat v9.0 server -> Click Next

└── Configure server options (e.g., ports, server location).

Step 7:-. Modify 'tomcat-users.xml':

└── Add role and user details under <tomcat-users> tag.

Step 8:. Build the project:

└── Right-click on index.jsp -> Run As -> Maven Clean

└── Right-click on index.jsp -> Run As -> Maven Install

└── Right-click on index.jsp -> Run As -> Maven Test

└── Right-click on index.jsp -> Run As -> Maven Build

Step 9. In the Maven Build dialog:

└── Enter Goals: clean install test

└── Click on Apply -> Click on Run

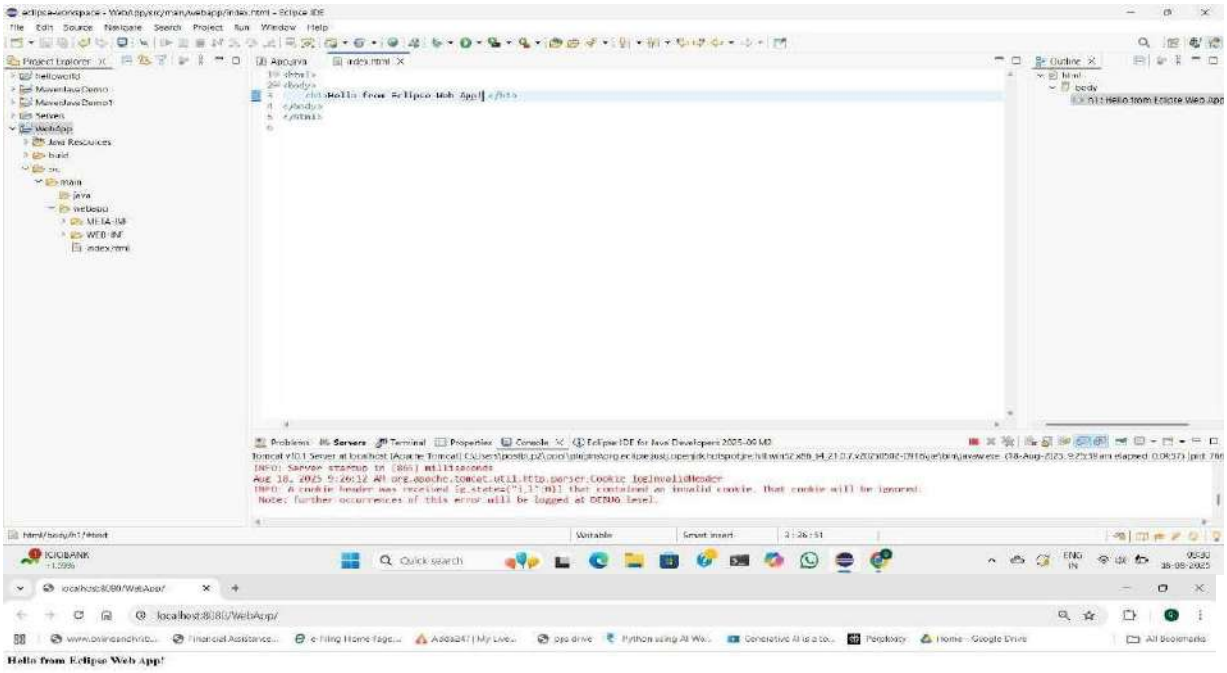
Step 10. Check console for BUILD SUCCESS message.

Step 11. Run the application:

└── Right-click on index.jsp -> Run As -> Run on Server

└── Select the Tomcat server -> Click on Finish

Step 12. Output: "Hello World" webpage displayed.



c. Add dependencies using pom.xml, compile and test using plugins.

Add a Dependency (e.g., Gson):
After running:

mvn clean install

Check:

- Gson JAR is downloaded to **.m2/repository/com/google/code/gson/gson/**
- If version is wrong or not found → **BUILD FAILURE** with dependency resolution error.

d. Resolve errors and conflicts arising from dependency mismatches.

- Typos in version numbers or missing repositories cause `BUILD FAILURE`
- Maven shows precise error in console
- Fix the dependency tag → re-run `mvn clean install`

e. Work with parent and multi-module Maven projects.

Structure:

parent/

├── pom.xml (packaging: pom)
├── core/
│ └── pom.xml
└── web/
└── pom.xml

- Parent pom.xml defines <modules> and common dependencies
- Each submodule builds independently into its target/ directory

e. Generate executable JARs and deployable WARs using Maven.

Executable JAR

Add to pom.xml:

<build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-jar-plugin</artifactId> <configuration> <archive>
<manifest> <mainClass>com.example.Main</mainClass>
</manifest> </archive> </configuration> </plugin> </plugins>
</build>

Run:

mvn package
java -jar target/myapp.jar

Executable WAR

Create a Maven web project with structure:
src/main/webapp/
└── WEB-INF/web.xml
Add:
<packaging>war</packaging>
Command:
mvn package
Generates target/mywebapp.war → deploy on Tomcat server.

Experiment- 5: Docker CLI commands

a. Learn how to pull, run, stop, start, remove, and inspect containers and images.

Docker CLI Commands with redis

Step 1: Pull the redis Image

Command: docker pull redis

```
PS C:\Users\RAMM> docker pull redis
Using default tag: latest
latest: Pulling from library/redis
b1badc6e5066: Download complete
6f34ca96de3f: Download complete
63edde0222ea: Download complete
4f4fb700ef54: Download complete
e51f60c71d8e: Download complete
f22261c94fe4: Download complete
311d9cf20af5: Download complete
Digest: sha256:cc2dfb8f5151da2684b4a09bd04b567f92d07591d91980eb3eca21df07e12760
Status: Downloaded newer image for redis:latest
```

Step 2: Run a Redis Container Command: docker run --name my- redis -d redis

```
PS C:\Users\RAMM> docker run --name my-redis -d redis
f7c205294842bed66bcec3261758e9302730e6740709701556c0b0db8199d2de
PS C:\Users\RAMM> docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
redis         latest    cc2dfb8f5151   12 hours ago    200MB
ubuntu        latest    b59d21599a2b   2 months ago    117MB
<none>        <none>    6015f66923d7   3 months ago    117MB
<none>        <none>    1e622c5f073b   4 months ago    117MB
<none>        <none>    72297848456d   6 months ago    117MB
PS C:\Users\RAMM> docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
redis         latest    cc2dfb8f5151   12 hours ago    200MB
ubuntu        latest    b59d21599a2b   2 months ago    117MB
<none>        <none>    6015f66923d7   3 months ago    117MB
<none>        <none>    1e622c5f073b   4 months ago    117MB
<none>        <none>    72297848456d   6 months ago    117MB
```

What It Does:

Creates and starts a container named my-redis from the redis image.

The -d flag runs the container in the background.

Step 3: Check Running Containers

Command: docker ps

```
PS C:\Users\RAMM> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\RAMM> docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED         STATUS                      PORTS     NAMES
e3018a7f3450   ubuntu         "/bin/bash"   2 months ago    Exited (255) 6 minutes ago            strange_mendeleev
2337d2755d16   ubuntu         "/bin/bash"   2 months ago    Exited (255) 2 months ago             stoic_wing
229f60b860f3   ubuntu         "/bin/bash"   2 months ago    Exited (255) 2 months ago             inspiring_blackwell
75a9f03a7d3c   ubuntu         "/bin/bash"   2 months ago    Exited (255) 2 months ago             hardcore_goldberg
06821945c7b6   6015f66923d7   "/bin/bash"   3 months ago    Exited (255) 2 months ago             awesome_wescoff
a42940212930   1e622c5f073b   "/bin/bash"   4 months ago    Exited (255) 3 months ago             affectionate_ramanujan
3980dbbf3512   1e622c5f073b   "/bin/bash"   4 months ago    Exited (255) 3 months ago             interesting_mccarthy
e8a4452f558c   1e622c5f073b   "/bin/bash"   4 months ago    Exited (255) 3 months ago             hungry_wilbur
2efe79d0b058   72297848456d   "/bin/bash"   4 months ago    Exited (255) 4 months ago             nifty_jepsen
d8d24f05424f   72297848456d   "/bin/bash"   6 months ago    Exited (255) 5 months ago             jolly_ellis
a719ed346bd9   72297848456d   "/bin/bash"   6 months ago    Exited (255) 6 months ago             hardcore_sammet
```

What It Does: Lists all running containers. **Step 4:** Access Redis Command:
docker exec -it my-redis redis-cli

Opens the Redis command-line tool (redis-cli) inside the container.

```
PS C:\Users\RAMM> docker exec -it my-redis redis-cli
127.0.0.1:6379> SET name "KMIT"
OK
127.0.0.1:6379> get name
"KMIT"
127.0.0.1:6379> exit
```

Step 5: Stop the Redis Container

Command: docker stop my-redis

```
PS C:\Users\RAMM> docker stop my-redis
my-redis
PS C:\Users\RAMM> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS     NAMES
f7c205294842   redis          "docker-entrypoint.s…"   11 minutes ago   Exited (0) 8 seconds ago              my-redis
e3018a7f3450   ubuntu         "/bin/bash"              2 months ago     Exited (255) 39 minutes ago           strange_mendeleev
2337d2755d16   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago             stoic_wing
229f60b860f3   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago             inspiring_blackwell
75a9f03a7d3c   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago             hardcore_goldberg
06821945c7b6   6015f66923d7   "/bin/bash"              3 months ago     Exited (255) 2 months ago             awesome_wescoff
a42940212930   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago             affectionate_ramanuj
an
3980dbbf3512   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago             interesting_mccarthy
e8a4452f558c   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago             hungry_wilbur
2efe79d0b058   72297848456d   "/bin/bash"              4 months ago     Exited (255) 4 months ago             nifty_jepsen
d8d24f05424f   72297848456d   "/bin/bash"              6 months ago     Exited (255) 5 months ago             jolly_ellis
a719ed346bd9   72297848456d   "/bin/bash"              6 months ago     Exited (255) 6 months ago             hardcore_sammet
```

What It Does: Stops the Redis container but doesn't delete it.

Step 6: Restart the Redis Container

Command: docker start my-redis

```
PS C:\Users\RAMM> docker start my-redis
my-redis
PS C:\Users\RAMM> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                      PORTS      NAMES
f7c205294842   redis          "docker-entrypoint.s…"   12 minutes ago   Up 12 seconds               6379/tcp   my-redis
e3018a7f3450   ubuntu         "/bin/bash"              2 months ago     Exited (255) 40 minutes ago            strange_mendeleev
2337d2755d16   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago              stoic_wing
229f60b860f3   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago              inspiring_blackwell
75a9f03a7d3c   ubuntu         "/bin/bash"              2 months ago     Exited (255) 2 months ago              hardcore_goldberg
06821945c7b6   6015f66923d7   "/bin/bash"              3 months ago     Exited (255) 2 months ago              awesome_wescoff
a42940212930   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago              affectionate_ramanuj
an
3980dbbf3512   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago              interesting_mccarthy
e8a4452f558c   1e622c5f073b   "/bin/bash"              4 months ago     Exited (255) 3 months ago              hungry_wilbur
2efe79d0b058   72297848456d   "/bin/bash"              4 months ago     Exited (255) 4 months ago              nifty_jepsen
d8d24f05424f   72297848456d   "/bin/bash"              6 months ago     Exited (255) 5 months ago              jolly_ellis
a719ed346bd9   72297848456d   "/bin/bash"              6 months ago     Exited (255) 6 months ago              hardcore_sammet
```

What It Does: Restarts the stopped container. **Step 7:** Remove the Redis Container Command:
docker rm my-redis
What It Does: Deletes the container permanently.

```
PS C:\Users\RAMM> docker rm my-redis
Error response from daemon: cannot remove container "/my-redis": container is running: stop the container before removing or force r
emove
PS C:\Users\RAMM> docker stop my-redis
my-redis
PS C:\Users\RAMM> docker rm my-redis
my-redis
PS C:\Users\RAMM> docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED          STATUS                      PORTS     NAMES
e3018a7f3450   ubuntu         "/bin/bash"   2 months ago     Exited (255) 43 minutes ago           strange_mendeleev
2337d2755d16   ubuntu         "/bin/bash"   2 months ago     Exited (255) 2 months ago             stoic_wing
229f60b860f3   ubuntu         "/bin/bash"   2 months ago     Exited (255) 2 months ago             inspiring_blackwell
75a9f03a7d3c   ubuntu         "/bin/bash"   2 months ago     Exited (255) 2 months ago             hardcore_goldberg
06821945c7b6   6015f66923d7   "/bin/bash"   3 months ago     Exited (255) 2 months ago             awesome_wescoff
a42940212930   1e622c5f073b   "/bin/bash"   4 months ago     Exited (255) 3 months ago             affectionate_ramanujan
3980dbbf3512   1e622c5f073b   "/bin/bash"   4 months ago     Exited (255) 3 months ago             interesting_mccarthy
e8a4452f558c   1e622c5f073b   "/bin/bash"   4 months ago     Exited (255) 3 months ago             hungry_wilbur
2efe79d0b058   72297848456d   "/bin/bash"   4 months ago     Exited (255) 4 months ago             nifty_jepsen
d8d24f05424f   72297848456d   "/bin/bash"   6 months ago     Exited (255) 5 months ago             jolly_ellis
a719ed346bd9   72297848456d   "/bin/bash"   6 months ago     Exited (255) 6 months ago             hardcore_sammet
```

Step 8: Remove the Redis Image

Command: docker rmi redis

```
PS C:\Users\RAMM> docker rmi redis
Untagged: redis:latest
Deleted: sha256:cc2dfb8f5151da2684b4a09bd04b567f92d07591d91980eb3eca21df07e12760
PS C:\Users\RAMM> docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    b59d21599a2b   2 months ago   117MB
<none>       <none>    6015f66923d7   3 months ago   117MB
<none>       <none>    1e622c5f073b   4 months ago   117MB
<none>       <none>    72297848456d   6 months ago   117MB
PS C:\Users\RAMM>
```

What It Does: Deletes the Redis image from your local system.

b. Gain the ability to create, monitor, and troubleshoot running containers.

docker run --name my-redis -d redis

```
PS C:\Users\RAMM> docker run --name my-redis -d redis
f7c205294842bed66bcec3261758e9302730e6740709701556c0b0db8199d2de
PS C:\Users\RAMM> docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    cc2dfb8f5151   12 hours ago    200MB
ubuntu       latest    b59d21599a2b   2 months ago    117MB
<none>       <none>    6015f66923d7   3 months ago    117MB
<none>       <none>    1e622c5f073b   4 months ago    117MB
<none>       <none>    72297848456d   6 months ago    117MB
PS C:\Users\RAMM> docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
redis        latest    cc2dfb8f5151   12 hours ago    200MB
ubuntu       latest    b59d21599a2b   2 months ago    117MB
<none>       <none>    6015f66923d7   3 months ago    117MB
<none>       <none>    1e622c5f073b   4 months ago    117MB
<none>       <none>    72297848456d   6 months ago    117MB
```

What It Does:

Creates and starts a container named my-redis from the redis image.

docker ps

```
PS C:\Users\RAMM> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Users\RAMM> docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED        STATUS                      PORTS     NAMES
e3018a7f3450   ubuntu         "/bin/bash"   2 months ago   Exited (255) 6 minutes ago            strange_mendeleev
2337d2755d16   ubuntu         "/bin/bash"   2 months ago   Exited (255) 2 months ago             stoic_wing
229f60b860f3   ubuntu         "/bin/bash"   2 months ago   Exited (255) 2 months ago             inspiring_blackwell
75a9f03a7d3c   ubuntu         "/bin/bash"   2 months ago   Exited (255) 2 months ago             hardcore_goldberg
06821945c7b6   6015f66923d7   "/bin/bash"   3 months ago   Exited (255) 2 months ago             awesome_wescoff
a42940212930   1e622c5f073b   "/bin/bash"   4 months ago   Exited (255) 3 months ago             affectionate_ramanujan
3980dbbf3512   1e622c5f073b   "/bin/bash"   4 months ago   Exited (255) 3 months ago             interesting_mccarthy
e8a4452f558c   1e622c5f073b   "/bin/bash"   4 months ago   Exited (255) 3 months ago             hungry_wilbur
2efe79d0b058   72297848456d   "/bin/bash"   4 months ago   Exited (255) 4 months ago             nifty_jepsen
d8d24f05424f   72297848456d   "/bin/bash"   6 months ago   Exited (255) 5 months ago             jolly_ellis
a719ed346bd9   72297848456d   "/bin/bash"   6 months ago   Exited (255) 6 months ago             hardcore_sammet
```

What It Does: Lists all running containers. docker logs my-redis

```
C:\Users\hariv>docker logs my-redis
Starting Redis Server
1:C 16 Nov 2025 12:46:44.997 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 16 Nov 2025 12:46:44.997 * Redis version=8.2.3, bits=64, commit=00000000, modified=1, pid=1, just started
1:C 16 Nov 2025 12:46:44.997 * Configuration loaded
1:M 16 Nov 2025 12:46:44.997 * monotonic clock: POSIX clock_gettime
1:M 16 Nov 2025 12:46:44.998 * Running mode=standalone, port=6379.
1:M 16 Nov 2025 12:46:44.998 * <bf> RedisBloom version 8.2.8 (Git=unknown)
1:M 16 Nov 2025 12:46:44.998 * <bf> Registering configuration options: [
1:M 16 Nov 2025 12:46:44.998 * <bf>      { bf-error-rate       :      0.01 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { bf-initial-size     :       100 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { bf-expansion-factor :         2 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { cf-bucket-size      :         2 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { cf-initial-size     :      1024 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { cf-max-iterations   :        20 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { cf-expansion-factor :         1 }
1:M 16 Nov 2025 12:46:44.998 * <bf>      { cf-max-expansions   :        32 }
1:M 16 Nov 2025 12:46:44.998 * <bf> ]
1:M 16 Nov 2025 12:46:44.998 * Module 'bf' loaded from /usr/local/lib/redis/modules//redisbloom.so
1:M 16 Nov 2025 12:46:45.001 * <search> Redis version found by RedisSearch : 8.2.3 - oss
1:M 16 Nov 2025 12:46:45.001 * <search> RediSearch version 8.2.5 (Git=222ad3b)
1:M 16 Nov 2025 12:46:45.001 * <search> Low level api version 1 initialized successfully
```

c. Configure and manage networks for container communication.

List Existing Networks
docker network ls

```
C:\Users\hariv>docker network ls
NETWORK ID     NAME                           DRIVER    SCOPE
31c5fdb6949f   bridge                         bridge    local
b60ee1e9bf6f   campusmgmtsystem_default       bridge    local
2d827f414a7b   host                           host      local
90d954b8c778   minikube                       bridge    local
c48cfac87cab   none                           null      local
```

Create a Custom Network

docker network create mynet

```
C:\Users\hariv>docker network create mynet
7ffdff9784dec90ae36d58ed2371e96b5241c256392b8ed3fb8f9035fd4574c4
```

Run Containers in the Same Network
docker run -d --name redis-server --network=mynet redis

```
C:\Users\hariv>docker run -d --name redis-server --network=mynet redis
757253d9cb146e2161fd82f610f34a7b523eedf259fa8468754ec85277e45089
```

d. Create and manage persistent storage for containers.

Create a Docker Volume

docker volume create mydata

```
C:\Users\hariv>docker volume create mydata
mydata
```

List All Volumes

docker volume ls

```
C:\Users\hariv>docker volume ls
DRIVER    VOLUME NAME
local     1e526237a6616ae2a4ae61b4ac6ecd9b04fd479f3168c57374fcce5c500b6c49
local     6b9c92ba1695220cb182570ddaea4f94ba038598956b2cd223a083ad238f6470
local     6ba3da0e8a7aab18c73298458baf821cc86da5b1dc1e3313bc1e6c5639bb5937
local     8af097ca3272d55f7704c66b144729a67e0217003c0c7b8c694a882d8f1ddb3d
local     8e0e832a836d6311c2e67cff7636aedf408e7fde647c0ff5a87c16e5c4f9f2f9
local     20b68914fdffb22538e4a1bede685b8400a377599c5efb2955343b49fe31cc22
local     63d0e9591ece8e9bed661a2de13c225ff2e295b8d60142947dd29a8c568a25ab
local     281aa4561ead21b125a3abc7de451b867cb0a673f8677925b0a9a7a0dc28ee2c
local     317f86e9bb0fe8fb7af0b92a816d6f503a0c51db049a83200f12e645d871c3a1
local     81405fef6d241a1e81a7f6ee9ffa59ced97dcdf6ea0c2b06c63821f09d5fa58e
local     146231b502f4aa5b9fae6a6083f3d90eb353e25c6637d8aad1e20dd8a45ef1d3
local     a1d8fbd026b1ca352ebeae5310c72cab3b97d5f436aa301bd252515cc305570f
local     ae5cc679d8b0b290f53715d89024ed6ea7d826badda15fe7f92ae919e6b59e88
local     af8cb48263f4155f53e1c58b40134753a4d06e16d6cafc2e808859c733bebe14
local     c6e224923a574186212a02278d1ea65d4d915a29e15d7f28bf0f0f4b91aee02a
local     campusmgmtsystem_mongo_data
local     campusmgmtsystem_mysql_data
local     campusmgmtsystem_postgress
local     d012abeca927d7c33433c191992ea247ec0c60a8b0ed2458e26c069820fcabf3
local     ec363566b7a1b40074ea5aa0f9ff4110474a503eb40dee078d391fd7d13b923b
local     f209dc2712c34650f256496d42e5a126725c4f6c6e64ace85cafc9392c0bdfdd
local     minikube
local     mydata
```

Inspect a Volume

docker volume inspect mydata

```
C:\Users\hariv>docker volume inspect mydata
[
    {
        "CreatedAt": "2025-11-16T12:54:18Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/mydata/_data",
        "Name": "mydata",
        "Options": null,
        "Scope": "local"
    }
]
```

Use a Volume in a Container

docker run -d --name my-reds -v mydata:/data redis

```
C:\Users\hariv>docker run -d --name my-reds -v mydata:/data redis
5b3a76f680866c43b8adad68bac6477aa6d6e769c8e41df1fca27a1499aebaa7
```

e. Learn how to list, remove, and manage images efficiently.

List Docker Images

docker images

Remove Docker Images
docker rmi image-name

```
C:\Users\hariv>docker rmi hello-world
Untagged: hello-world:latest
Deleted: sha256:a0dfb02aac212703bfcb339d77d47ec32c8706ff250850ecc0e19c8737b18567
```

Pull the redis Image

docker pull redis

```
PS C:\Users\RAMM> docker pull redis
Using default tag: latest
latest: Pulling from library/redis
b1badc6e5066: Download complete
6f34ca96de3f: Download complete
63edde0222ea: Download complete
4f4fb700ef54: Download complete
e51f60c71d8e: Download complete
f22261c94fe4: Download complete
311d9cf20af5: Download complete
Digest: sha256:cc2dfb8f5151da2684b4a09bd04b567f92d07591d91980eb3eca21df07e12760
Status: Downloaded newer image for redis:latest
```

Experiment-6: Docker

a. Learn how to define and run multiple interdependent services (e.g., web server, database) in a single configuration file.

I. Create a new folder compose-lab

Inside it, create a file docker-compose.yml with the following content:

```
version: "3.9"
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"

  db:
    image: postgres:15
    environment:
    POSTGRES_USER: demo
      POSTGRES_PASSWORD: demo
      POSTGRES_DB: demo_db
```

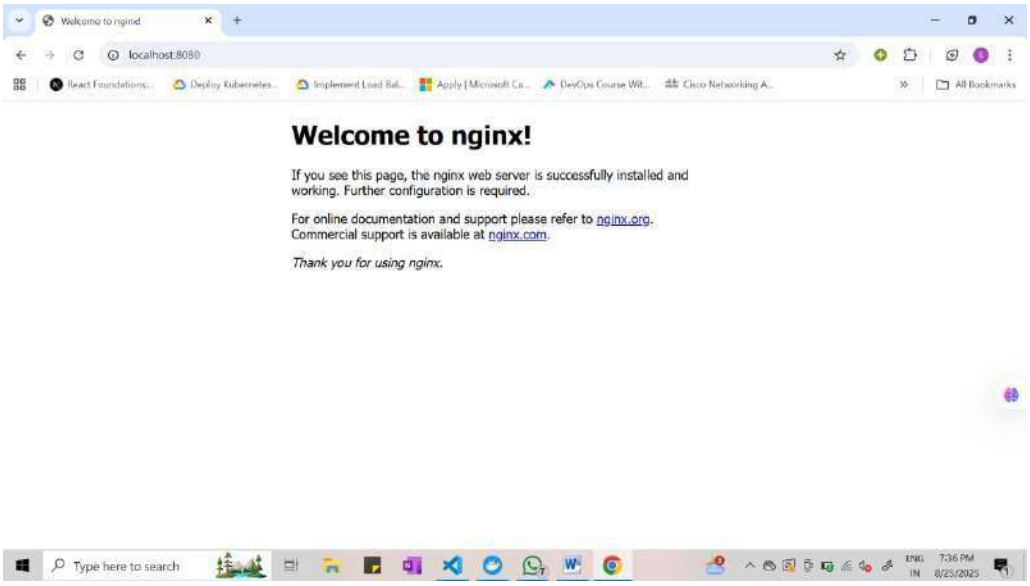
II. Run the setup:

```
docker compose up -d
```

III. Open your browser and visit: http://localhost:8080.

IV. Expected Output:

Nginx welcome page is displayed.
db container runs in the background.



b. Gain skills in writing and interpreting docker-compose.yml files for service setup.

I. Modify docker-compose.yml to add a Redis cache:

// Note add these lines in above code in Services:

```
redis:
  image: redis:alpine
```

II. Add a depends_on so web waits for Redis:

```
web:
  image: nginx:latest
  ports:
    - "8080:80"
  depends_on:
    - redis
```

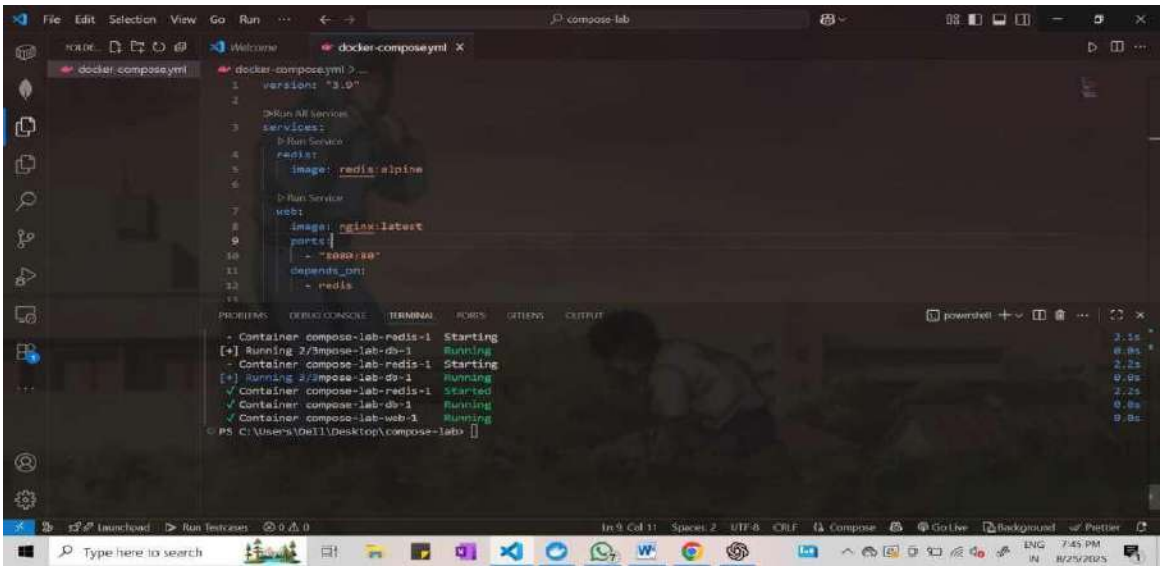
III. Restart the setup:

docker compose up -d

docker compose ps

IV. Expected Output:

Three services (web, db, redis) are listed as running.



c. Deploy the same setup across different machines without manual configuration.

I. Zip your compose-lab folder.

Transfer it to another machine with Docker Compose installed.

II. Run:

docker compose up -d
Check that Nginx and Postgres work there as well.

III. **Expected Output:**
IV. The same services run on the new machine without changes.
Now:

All three services (redis, web, db) are connected via the custom network app-net.
Postgres persists data using db-data volume.
web depends on both redis and db.

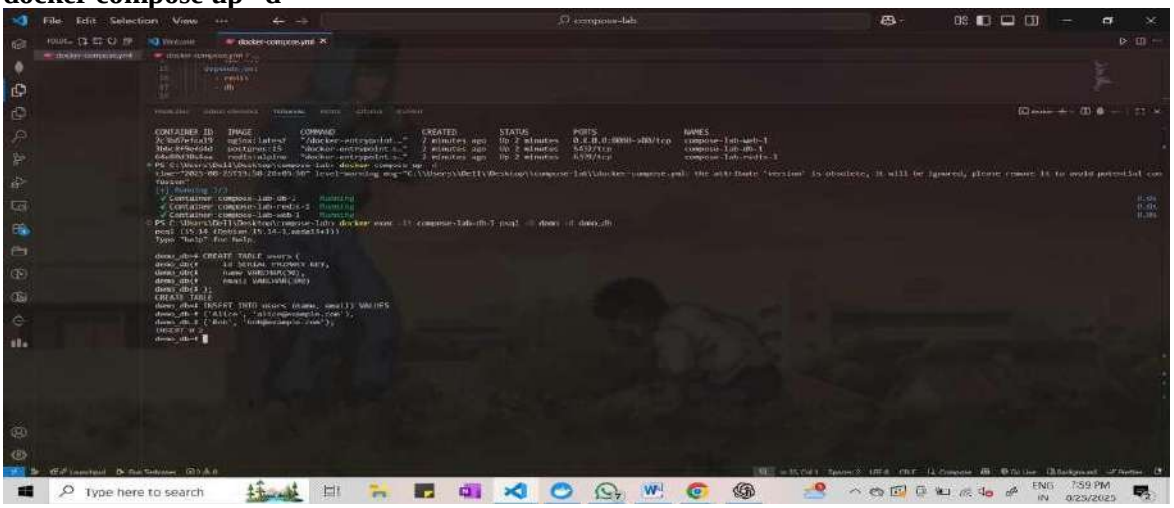
d. Configure container networking and persistent storage within Compose.

I. Update your docker-compose.yml to add a custom network and volume:

```
networks:
  app-net:

volumes:
  db-data:

services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    networks:
      - app-net
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
    POSTGRES_USER: demo
      POSTGRES_PASSWORD: demo
      POSTGRES_DB: demo_db
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - app-net
```

II. Run:

docker compose up –d



Note:

You already have Postgres running in your docker-compose.yml.
Now, to insert some data into Postgres, you have two main options:

Option 1: Using psql inside the running container

1. Start your containers:

```
sh
docker-compose up -d
```

2. Open a shell inside the Postgres container:

```
sh
docker exec -it <container_name> psql -U demo -d demo_db
```

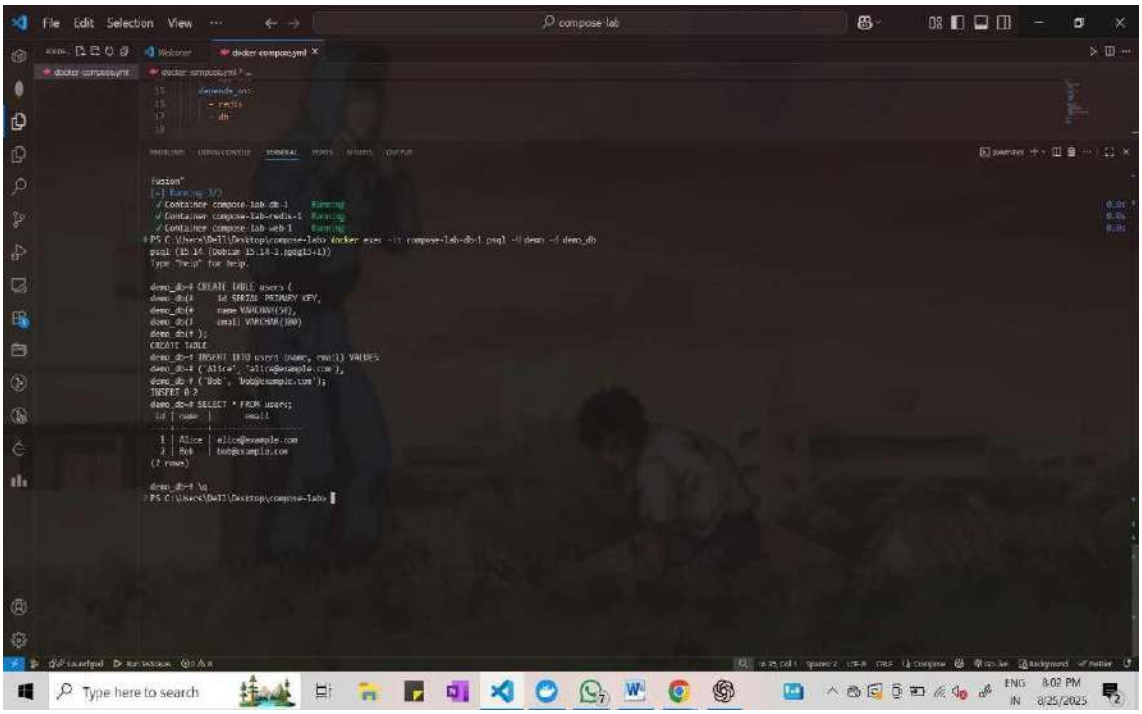
Replace <container_name> with your Postgres container name (you can check with docker ps). As here docker exec -it compose-lab-db-1 psql -U demo -d demo_db

3. Run SQL commands inside psql:

```
sql
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);
INSERT INTO users (name, email) VALUES
('Alice', 'alice@example.com'),
('Bob', 'bob@example.com');
```

4. Verify data:

```
sql
SELECT * FROM users;
```



III. Insert some data into Postgres (optional with psql).
IV. Remove containers:
docker compose down
V. Start again:
docker compose up -d
VI. Expected Output:
Database data persists across restarts.
Services communicate via the app-net network using service names.

e. Reduce setup time and enable faster iteration during application development.

I. Create a simple Flask app in app.py:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")

def home():

    return "Hello from Flask + Docker!"

if __name__== "__main__":

    app.run(host="0.0.0.0", port=5000)
```

II. Add a Dockerfile in the same folder:

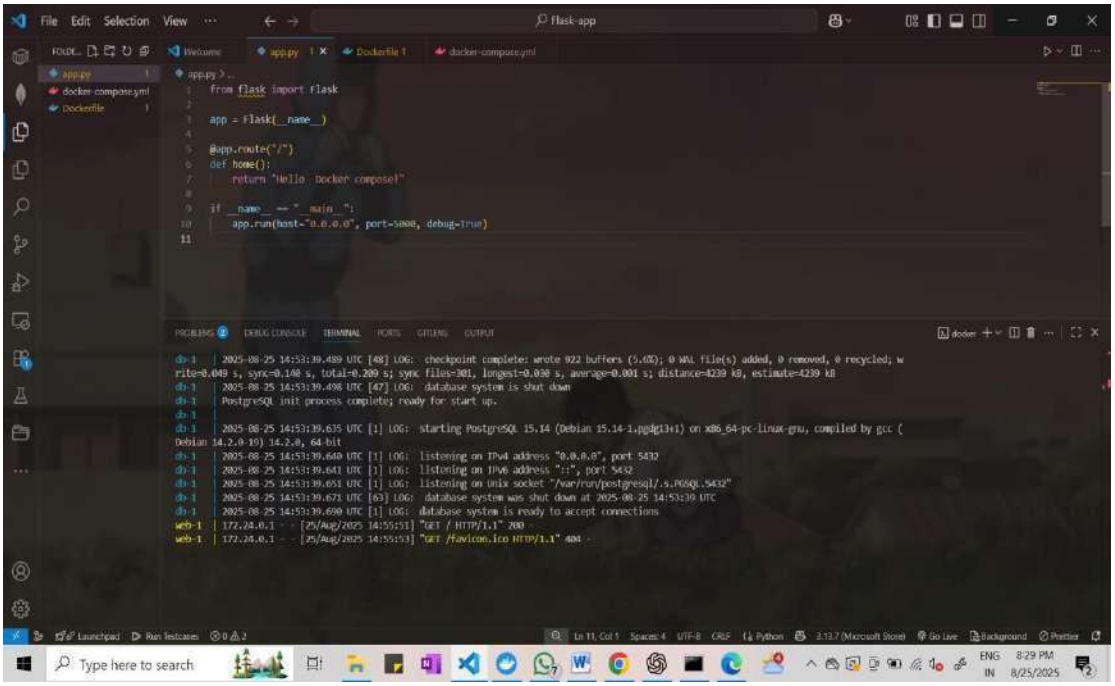
```
FROM python:3.10-slim
```

WORKDIR /app

COPY app.py /app/

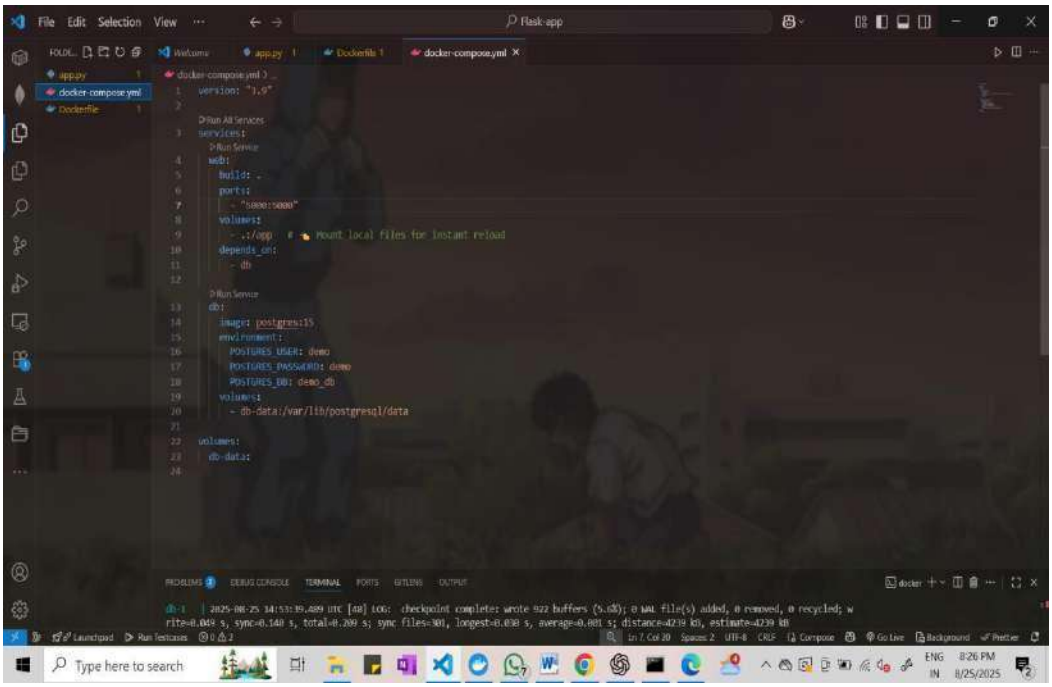
RUN pip install flask

CMD ["python", "app.py"]



III. Update docker-compose.yml:

```
  web:
    build: .
    ports:
      - "5000:5000"
    depends_on:
      - db
```

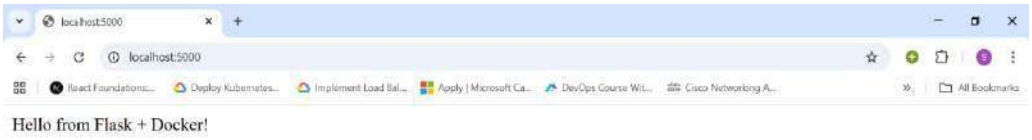


IV. Run:

docker compose up --build

Visit http://localhost:5000.

Change the return text in app.py (e.g., "Hello Docker Compose!").

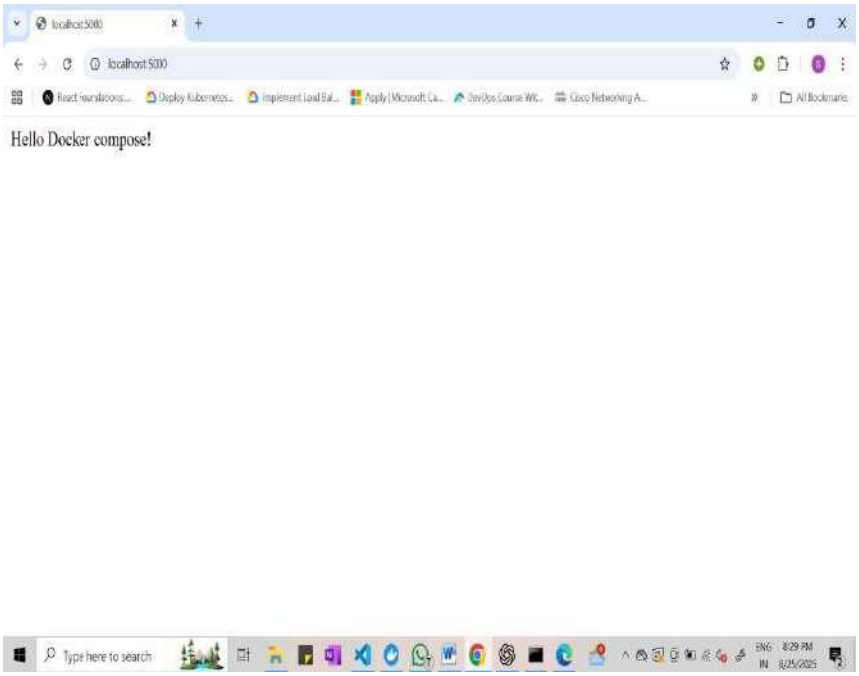


V. Rebuild:

docker compose up –build // not needed

VI. Expected Output:

New message appears instantly after rebuild.



Experiment-7: Creating a Multi-Module Maven Project

a. Build and package Java and Web applications using Maven.

----mvn clean install

- clean: Deletes the previous build (target/ folder)
- install: Builds, tests, and installs the package to local .m2 repository After

execution:

- target/ folder contains compiled .class files and the final .jar or .war
- Artifact is stored in:
  ~/.m2/repository/groupId/artifactId/version/

Creation of Maven Java Project

Step 1. Open Eclipse IDE

└── 1.1. Launch Eclipse workspace

Step 2. Install Maven Plugin (if not installed)

└── 2.1. Go to "Help" in the top menu

└── 2.1.1. Click "Eclipse Marketplace"

└── 2.1.2. Search for "Maven Integration for Eclipse"

└── 2.1.3. Install the plugin if not already installed

Step 3. Create a New Maven Project

└── 3.1. File -> New -> Project...

└── 3.1.1. Expand "Maven"

└── 3.1.2. Select "Maven Project" and click "Next"

Step 4. Set Project Configuration

└── 4.1. Select workspace location (default or custom)

└── 4.2. Click "Next"

Step 5. Choose Maven Archetype

└── 5.1. Select an archetype(e.g "org.apache.maven.archetypes -> maven-archetype-quickstart 1.4 ")

└── 5.2. Click "Next"

Step 6. Define Project Metadata

└── 6.1. Group ID: (e.g., com.example)

└── 6.2. Artifact ID: (e.g., my-maven-project)

└── 6.3. Version: (default is usually fine)

└── 6.4. Click "Finish"

In Console, artifacts are grouped. When prompted with Y/N, type 'Y'. Step 7.

Maven Project Created

└── 7.1. Project structure is generated with a standard Maven layout

└── 7.2. Includes:

└── src/main/java (for Java source code)

└── src/test/java (for test code)

└── pom.xml (Maven configuration file)

Step 8. Update Project Settings (if needed)

└── 8.1. Right-click on the project -> Maven -> Update Project...

└── 8.2. Ensure dependencies are up to date

Step 9. Build and Run Maven Project

└── 9.1. Right-click on App.java -> Run As -> Maven Clean

└── 9.1.1. Right-click on App.java -> Run As -> Maven Install

└── 9.1.2. Right-click on App.java -> Run As -> Maven Test

└── 9.1.3. Right-click on App.java -> Run As -> Maven Build Step 10.

In the Maven Build dialog:

└── Enter Goals: clean install test

└── Click on Apply -> Click on Run

Step 11. Check console for BUILD SUCCESS message. Step 12.

Run the application:

└── Right-click on App.java -> Run As -> Java Application

└── Output: "Hello World" displayed.

Creation of Maven web Java Project

Step 1: Open Eclipse

└── 1.1 Launch Eclipse IDE.

└── 1.2 Select or create a workspace.

Step 2: Create a New Maven Project

└── 2.1. File -> New -> Project...

└── 2.1.1. Expand "Maven"

└── 2.1.2. Select "Maven Project" and click "Next"

Step 3: Choose Maven Archetype

└── 3.1. Select an archetype(e.g "'org.apache.maven.archetypes' -> 'maven-archetype- webapp' 1.4 ")

└── 3.2. Click "Next"

Step 4: Configure the Maven Project

└── 4.1 Group Id: Enter a group ID (e.g., com.example).

└── 4.2 Artifact Id: Enter an artifact ID (e.g., my-web-app).

└── 4.3 Click **Finish** to create the project.

Step 5: Add Maven Dependencies

└── 5.1 Open the **pom.xml** file in the Maven project.

└── 5.2 Add the necessary dependencies for your web project (e.g., Servlet, JSP):

Go to browser -> Open mvnrepository.com

Search for 'Java Servlet API' -> Select the latest version.

Copy the dependency code -> Paste it in MavenWeb's pom.xml under the target folder

Step 6:-. Configure server:

└── Window -> Show View -> Servers
└── Add server -> Select Tomcat v9.0 server -> Click Next

└── Configure server options (e.g., ports, server location).

Step 7:-. Modify 'tomcat-users.xml':

└── Add role and user details under <tomcat-users> tag.

Step 8:. Build the project:

└── Right-click on index.jsp -> Run As -> Maven Clean

└── Right-click on index.jsp -> Run As -> Maven Install

└── Right-click on index.jsp -> Run As -> Maven Test

└── Right-click on index.jsp -> Run As -> Maven Build Step 9.

In the Maven Build dialog:

└── Enter Goals: clean install test

└── Click on Apply -> Click on Run

Step 10. Check console for BUILD SUCCESS message. Step 11.

Run the application:

└── Right-click on index.jsp -> Run As -> Run on Server

└── Select the Tomcat server -> Click on Finish Step

12. Output: "Hello World" webpage displayed.

Note:-Now push yours Maven java project and Maven Web Project into your github

b. Add dependencies using pom.xml, compile and test using plugins.

Add a Dependency (e.g., Gson):

In pom.xml:

<dependency> <groupId>com.google.code.gson</groupId> <artifactId>gson</artifactId> <version>2.10</version> </dependency>

After running:

mvn clean install

Check:

- Gson JAR is downloaded to .m2/repository/com/google/code/gson/gson/
- If version is wrong or not found → BUILD FAILURE with dependency resolution error.

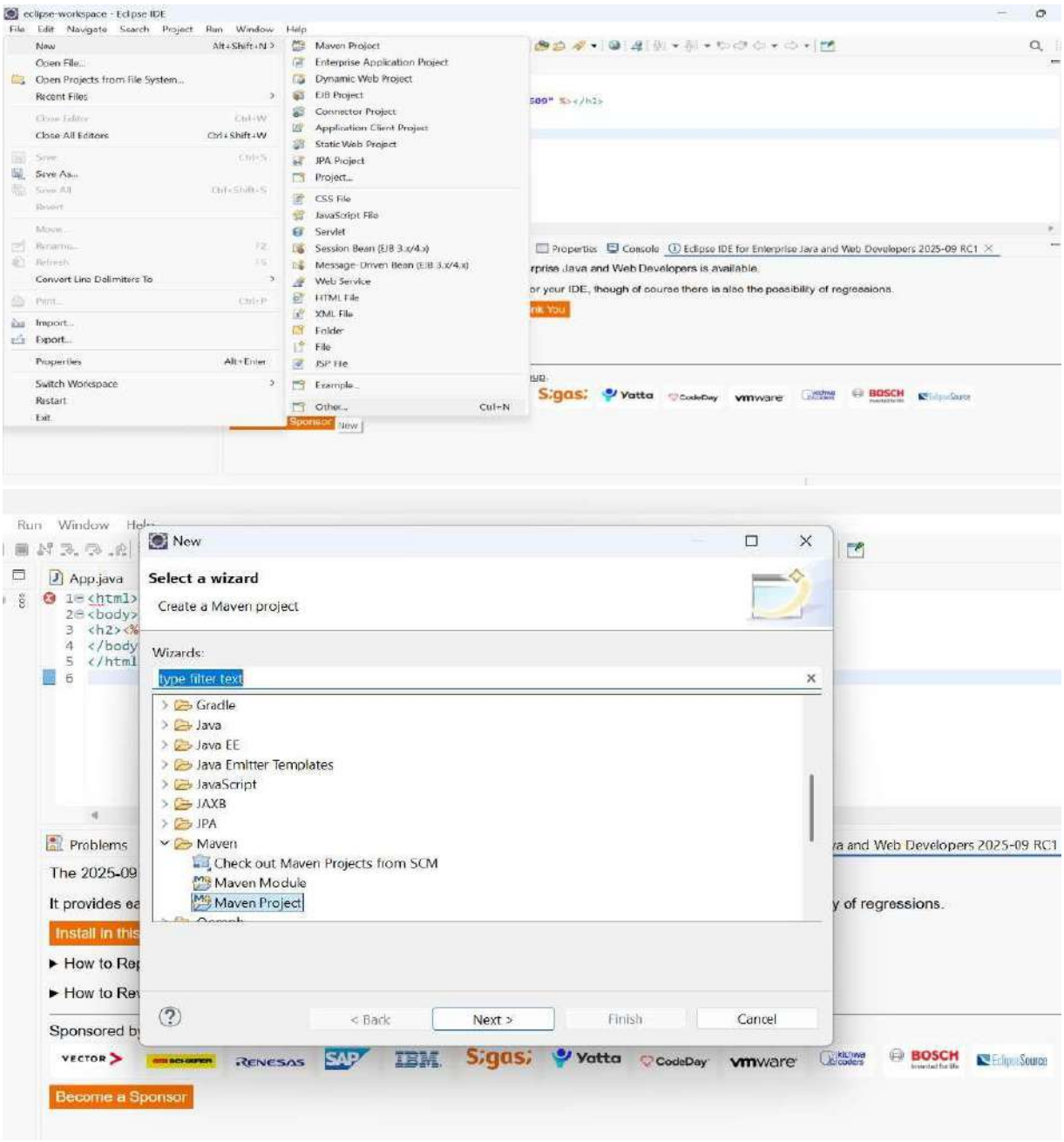
c. Resolve errors and conflicts arising from dependency mismatches.

- Typos in version numbers or missing repositories cause BUILD FAILURE
- Maven shows precise error in console
- Fix the dependency tag → re-run mvn clean install

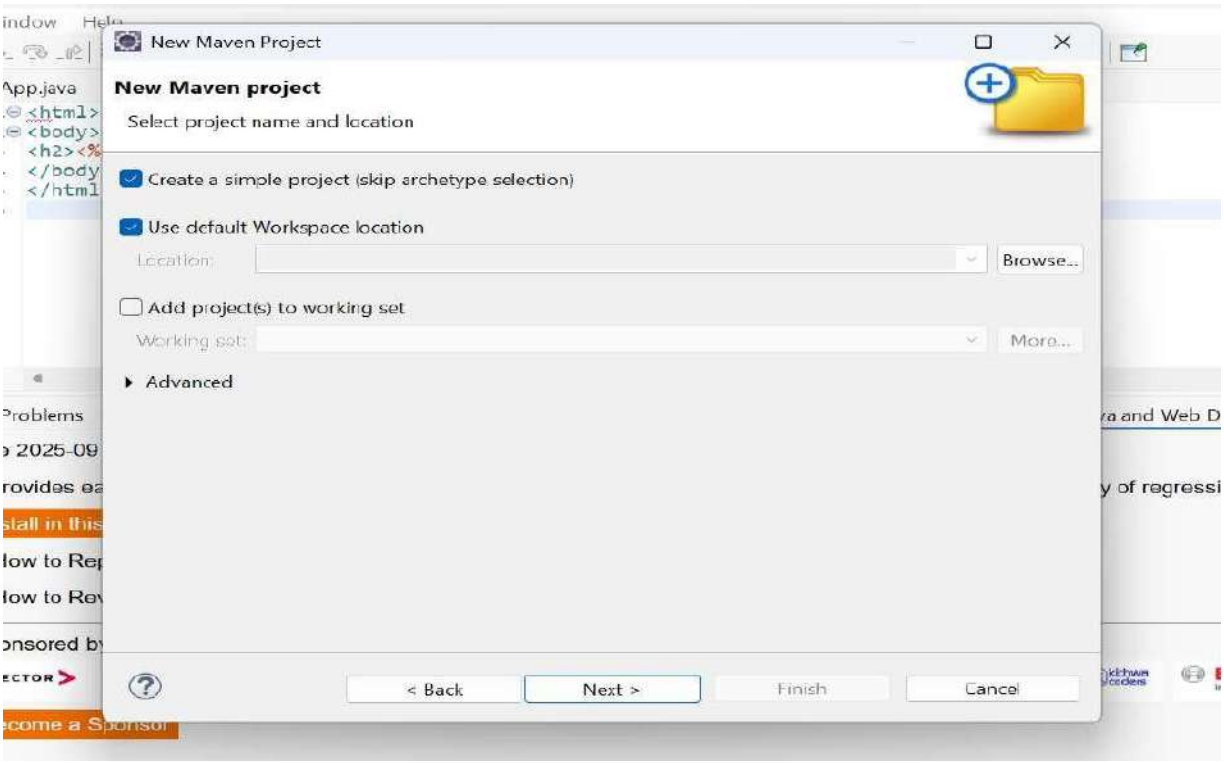
d. Work with parent and multi-module Maven projects.

Creating a Multi-Module Maven Project:

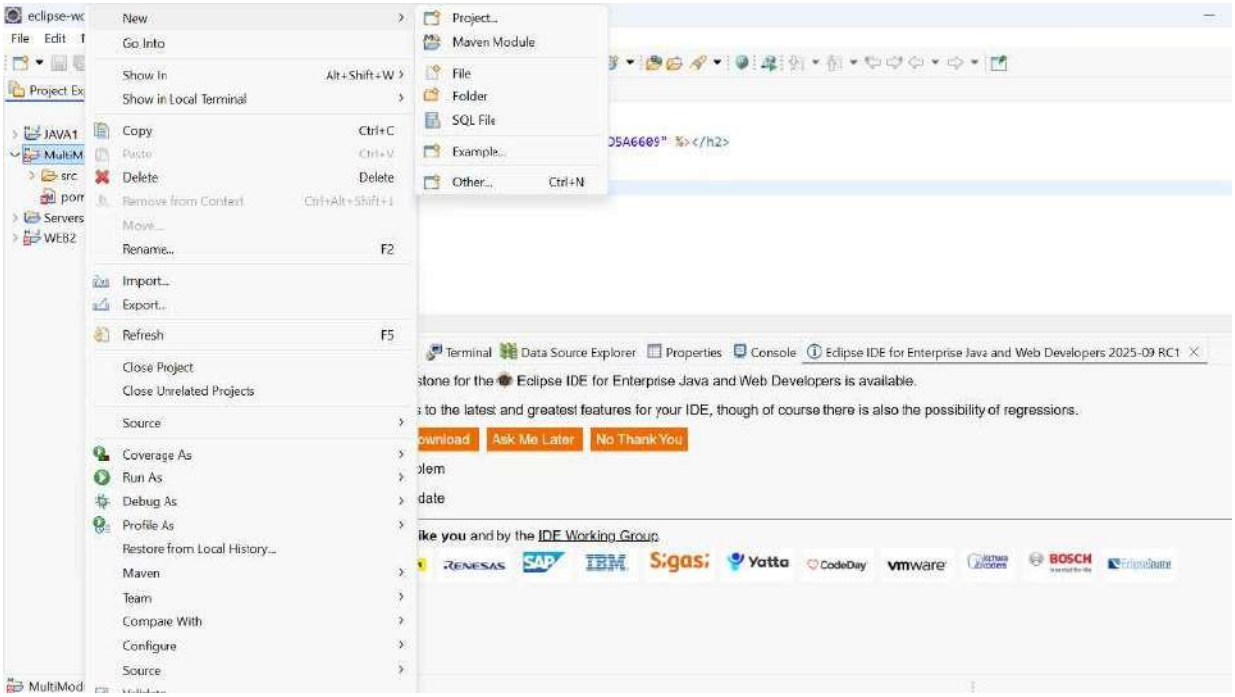
1. Open eclipse-> file-> new -> other -> Maven -> MavenProject ->click on next.



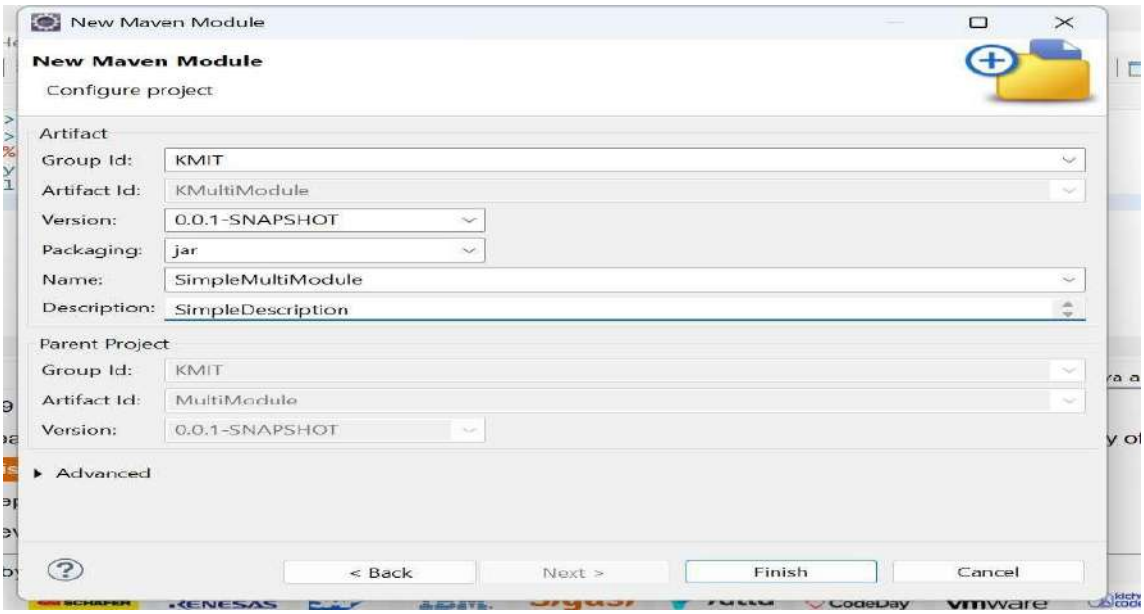
2. Check the first option: Create a Simple project(skip archetype selection) and click on next.



3. Give groupID: KMIT, artifact ID: MultiModule, in the packaging tab select the drop down and opt for pom.
4. Give Name: Multimodule creation and Description: Sample multimodule with parent child hierarchy and click on next.
5. Now the Multimodule folder is created and displayed in the project explorer on the left. Right click on the folder and opt new -> mavenModule

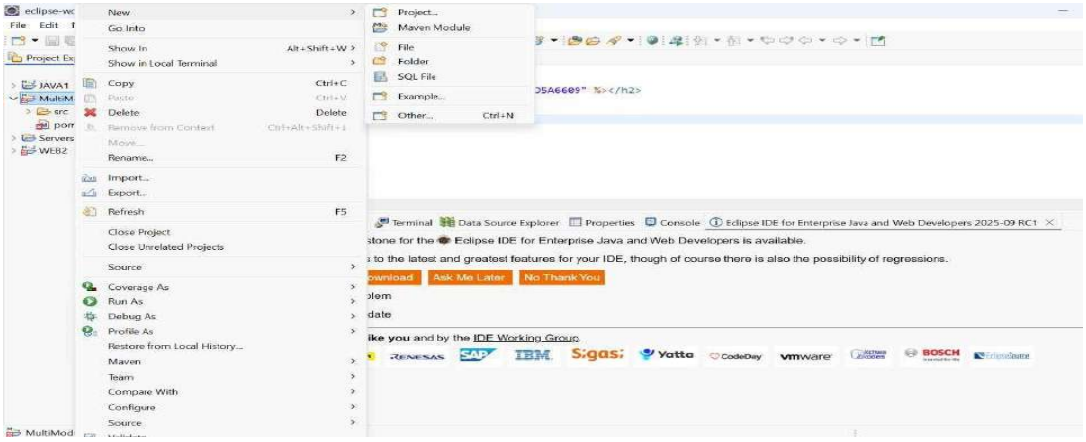


6. Check the option Create a Simple project(skip archetype selection, give the artifactID: MultiModuleChild1 and click on next and finish.

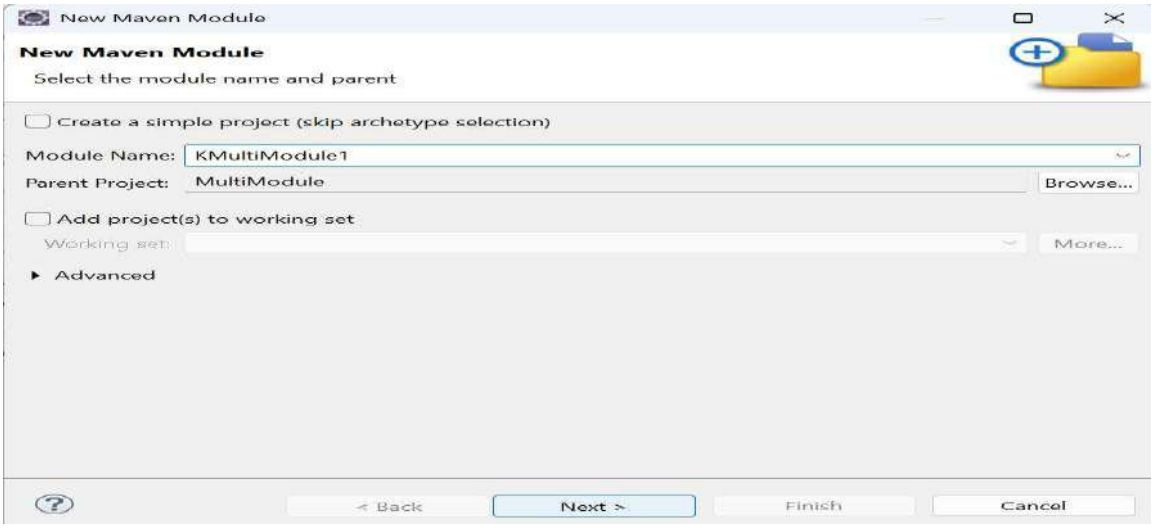


7. Now you can see, the module1 we created is displayed in the project explorer with a separate pom.xml file.

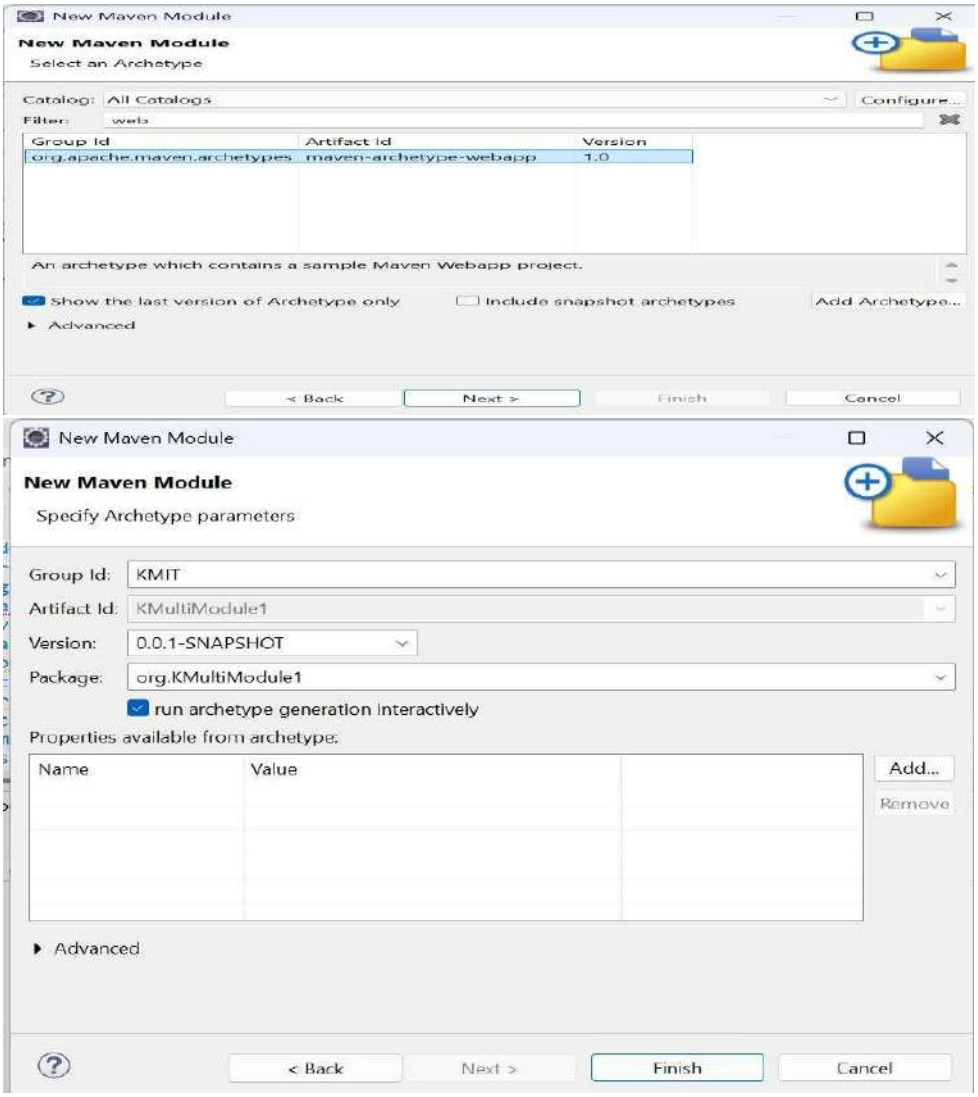
8. . Right click on the MultiModule folder in the project explorer and opt new -> mavenModule



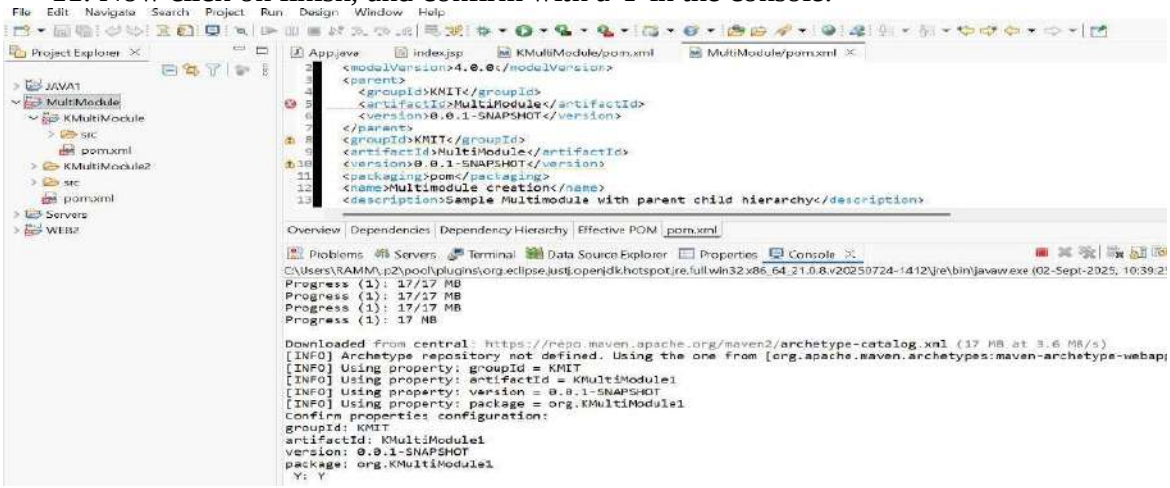
9. Give the artifactID: MultiModuleChild2 and click on next.



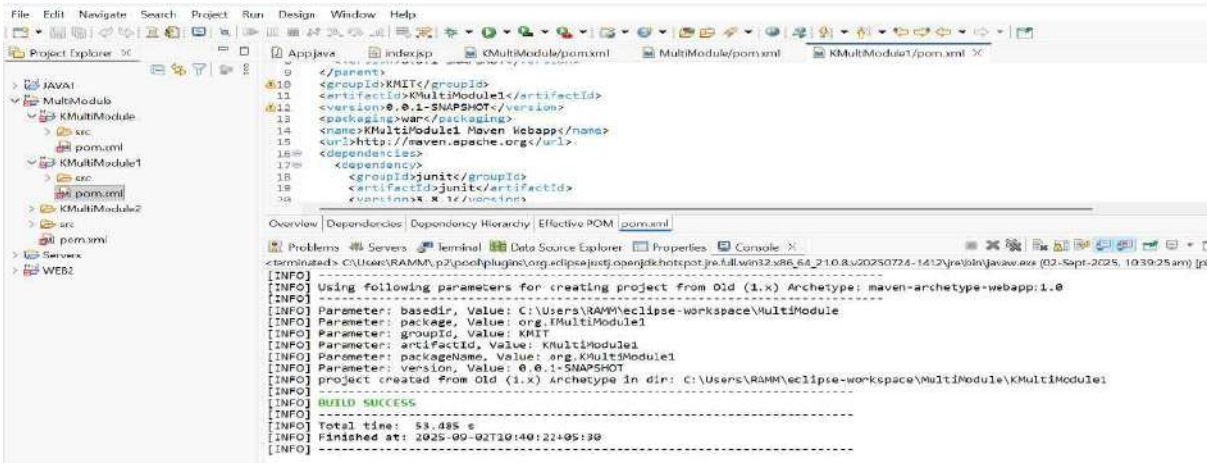
10. In filters search for maven-archetype-webapp, select the version 1.1/1.5 and click on next.



11. Now click on finish, and confirm with a Y in the console.



12. Now you can watch the new child is been added to the parent project.



Note: *Childs or Modules can be added to the projects who have the packaging type as pom. If we try to create a module within child1 it is not possible since its packaging is opted to JAR.*

Linking two modules in a Parent Project:
Go to the module u want to connect. And add a dependency to other module on which the current module depends. For example if my child2 is dependent on child1. I have to change the dependencies in the pom.xml by adding a new dependency of child1.

Steps to add dependency:

1. Goto pom.xml of child2.

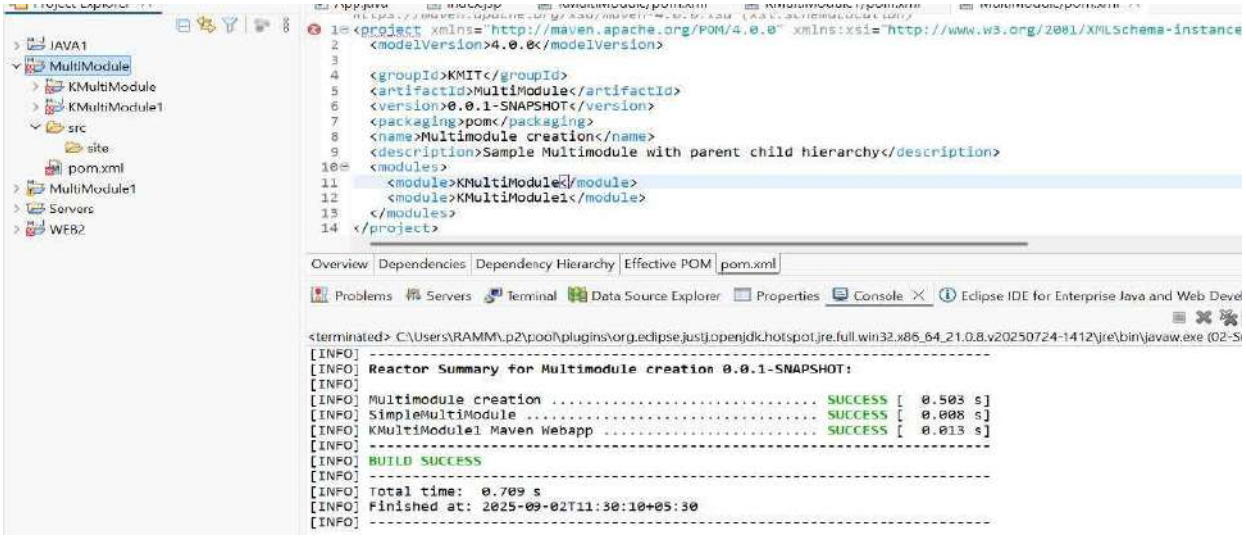
2. Navigate to dependencies tag. Update the dependencies with child1 dependency as given below, and save pom.xml after update.

<dependency> <groupId>KMIT</groupId> <artifactId>MultimoduleChild1</artifactId><version>0.0.1-SNAPSHOT</version> </dependency>

3. Now build the child2 by right clicking it and opting run as->maven build.

4. The build will be a *failure.* Because we haven't build the parent project and the child project. Child1 is dependent on parent. And Child2 is dependent on child2. SO we have to build ass the previous module first and then build the current child2.

5. After building parent and child1, build child2 now the build will be a *Success*



e. Generate executable JARs and deployable WARs using Maven

Executable JAR

Add to pom.xml:

<build> <plugins> <plugin> <groupId>org.apache.maven.plugins</groupId> <artifactId>maven-jar-plugin</artifactId> <configuration> <archive> <manifest> <mainClass>com.example.Main</mainClass> </manifest> </archive> </configuration> </plugin> </plugins> </build>

Run:

mvn package

java -jar target/myapp.jar

Executable WAR

Create a Maven web project with structure:

src/main/webapp/

└── WEB-INF/web.xml

Add:

<packaging>war</packaging>

Command:

mvn package

Generates target/mywebapp.war → deploy on Tomcat server.

Experiment-8: Jenkins Automation

a. Hands-on practice on manual creation of Jenkins pipeline using Maven projects from Github

Maven Java Automation Steps:

Step 1: Open Jenkins (localhost:8080)

├── Click on "New Item" (left side menu

Step 2: Create Freestyle Project (e.g., MavenJava_Build)

├── Enter project name (e.g., MavenJava_Build)

├── Click "OK"

└── **Configure the project:**

├── **Description**: "Java Build demo"

├── **Source Code Management**:

└── Git repository URL: [GitMavenJava repo URL]

├── **Branches to build**: */Main or */master

└── **Build Steps**:

├── **Add Build Step** -> "Invoke top-level Maven targets"

└── Maven version: MAVEN_HOME

└── Goals: clean

├── **Add Build Step** -> "Invoke top-level Maven targets"

└── Maven version: MAVEN_HOME

└── Goals: install

└── **Post-build Actions**:

├── Add Post Build Action -> "Archive the artifacts"

└── Files to archive: **/*

├── Add Post Build Action -> "Build other projects"

└── Projects to build: MavenJava_Test

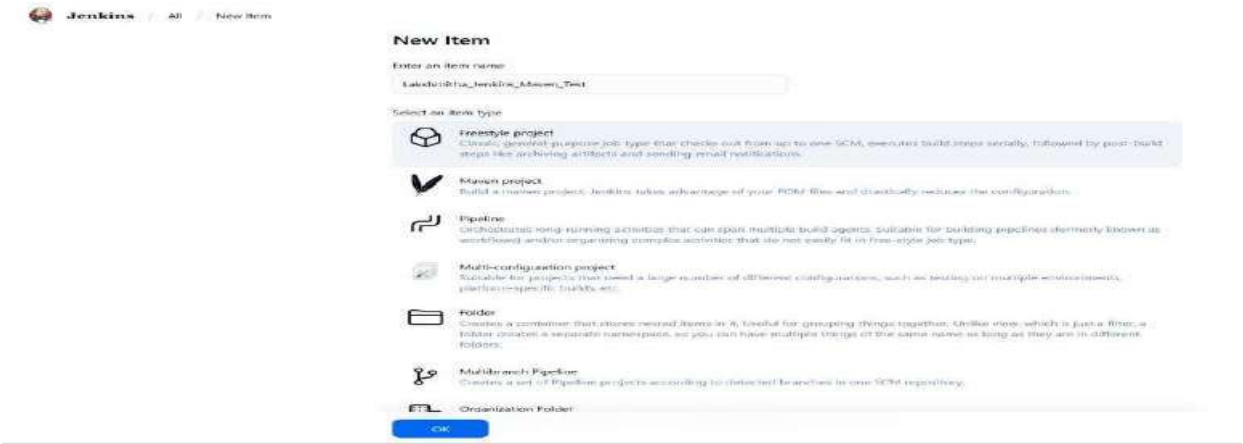
└── Trigger: Only if build is stable

└── Apply and Save

└── **Step 3: Create Freestyle Project (e.g., MavenJava_Test)**

├── Enter project name (e.g., MavenJava_Test)

├── Click "OK"



└── **Configure the project**:

├── **Description**: "Test demo"

├── **Build Environment**:

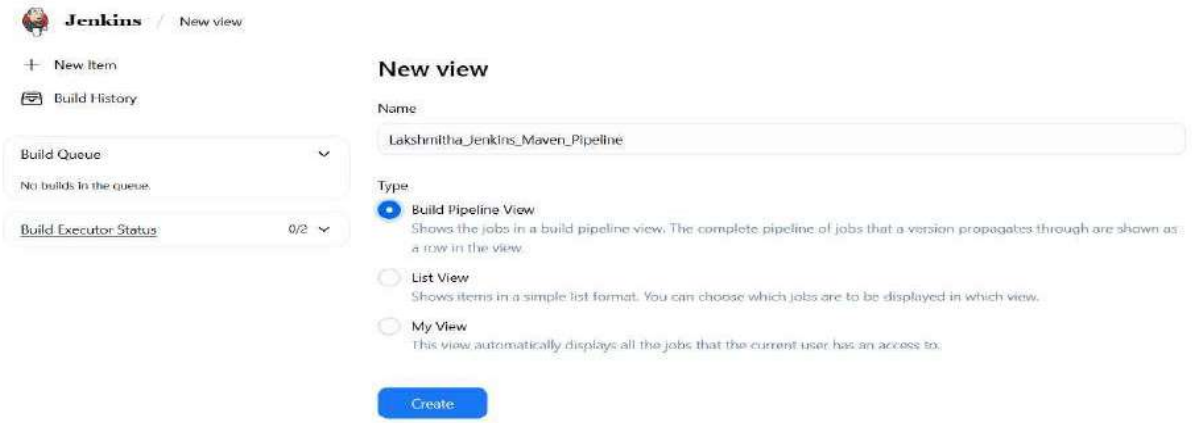
└── Check: "Delete the workspace before build starts"

├── **Add Build Step** -> "Copy artifacts from another project"

└── Project name: MavenJava_Build

└── Build: Stable build only // tick at this

└── Artifacts to copy: **/*



├── **Add Build Step** -> "Invoke top-level Maven targets"

└── Maven version: MAVEN_HOME

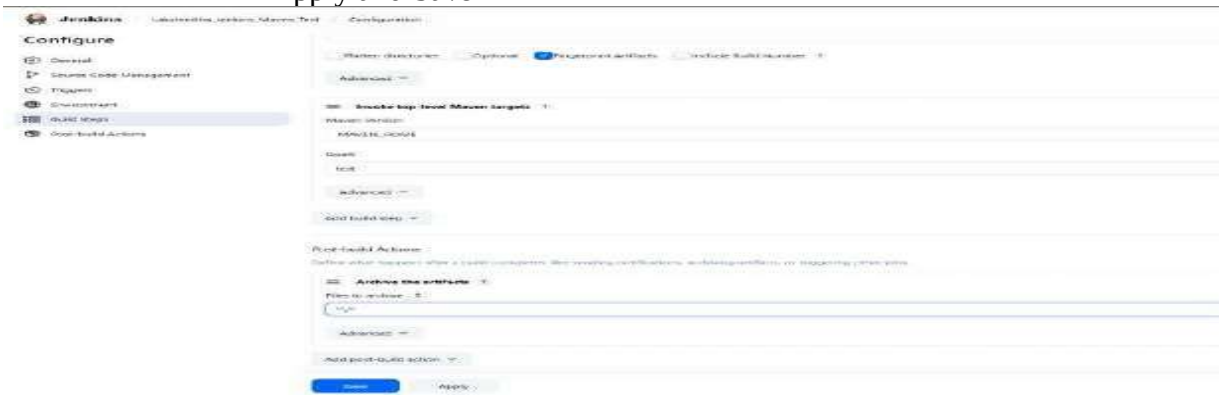
└── Goals: test

└── Post-build Actions:

├── **Add Post Build Action** -> "Archive the artifacts"

└── Files to archive: **/*

└── Apply and Save



└── **Step 4**: Create Pipeline View for Maven Java project

├── Click "+" beside "All" on the dashboard

├── Enter name: MavenJava_Pipeline

├── **Select "Build pipeline view"** **// tick here**

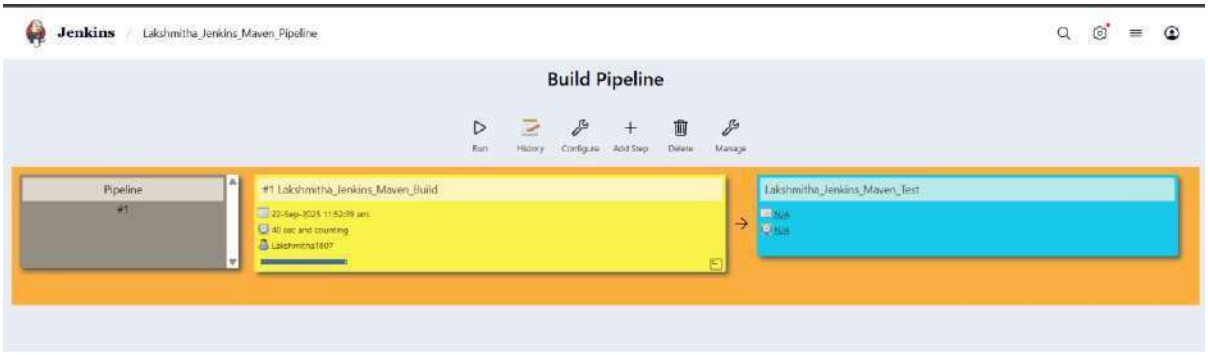
|--- **create**

└── **Pipeline Flow**:

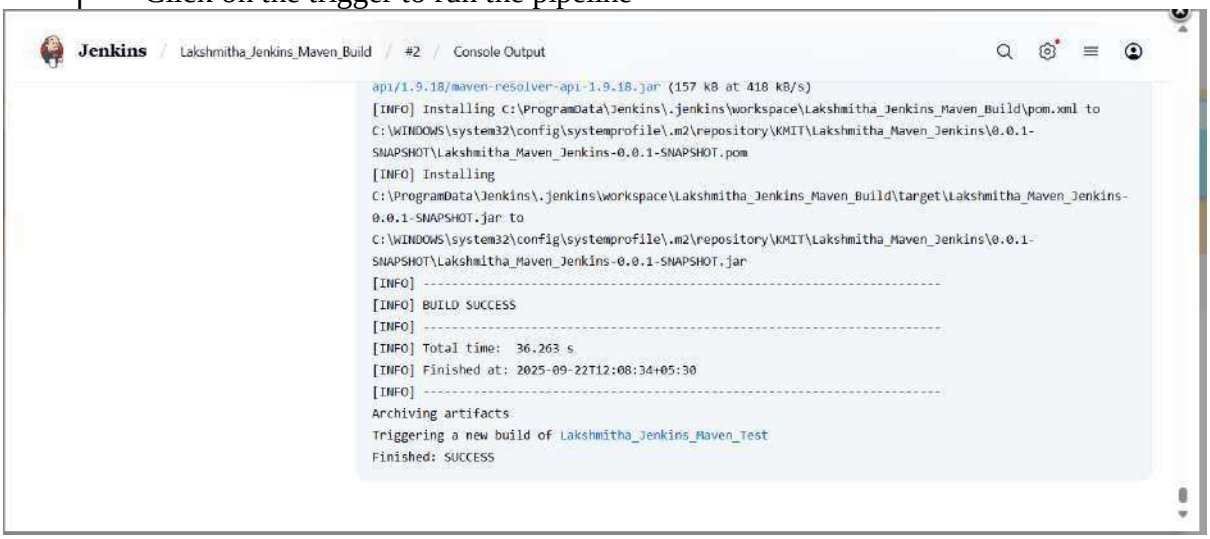
├── **Layout**: Based on upstream/downstream relationship

├── Initial job: MavenJava_Build
└── Apply and Save OK

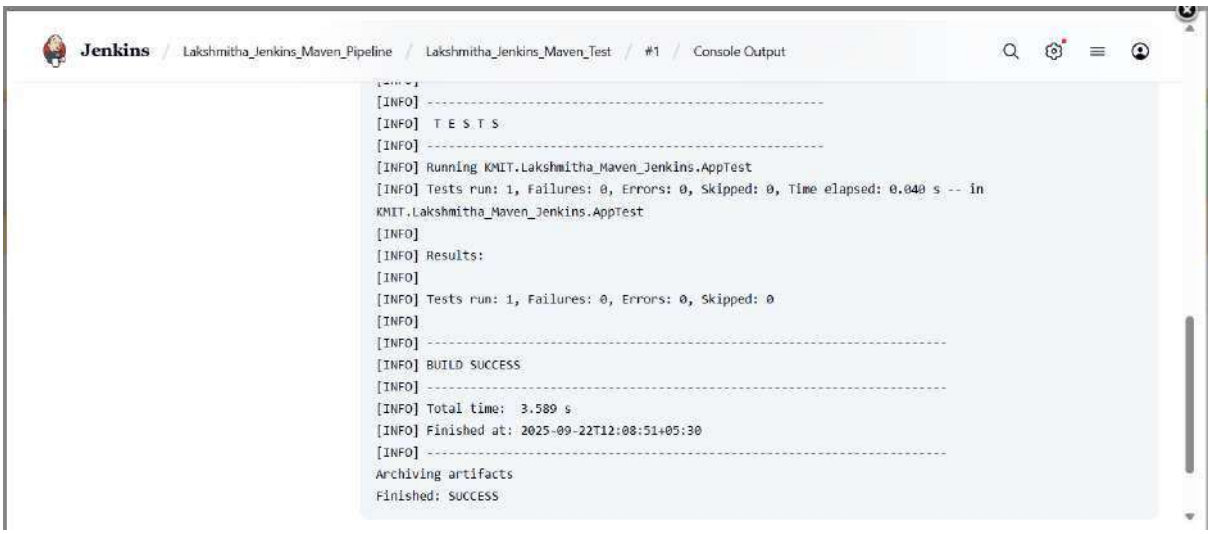
└── **Step 5**: Run the Pipeline and Check Output



├── Click on the trigger to run the pipeline



├── click on the small black box to open the console to check if the build is success

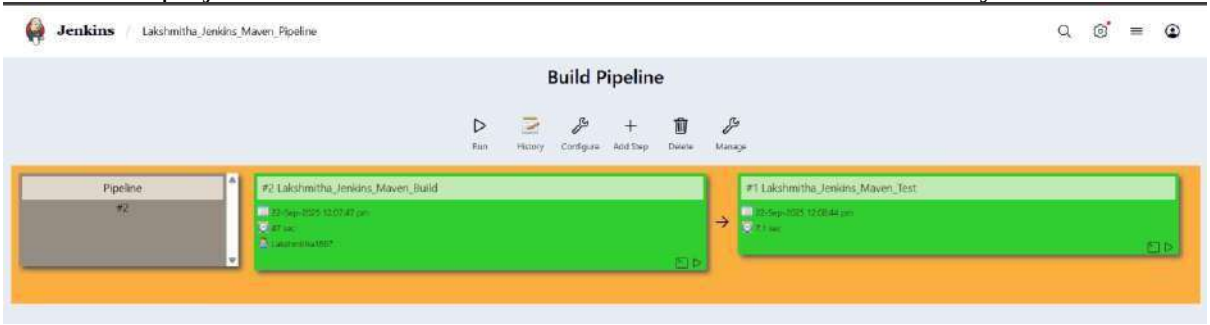


Note :

1. If build is success and the test project is also automatically triggered with name

   "MavenJava_Test"

2. The pipeline is successful if it is in green color as shown ->check the console of the test project
3. The test project is successful and all the artifacts are archived successfully



b. Create the job and build the pipeline for maven-java and maven-web project

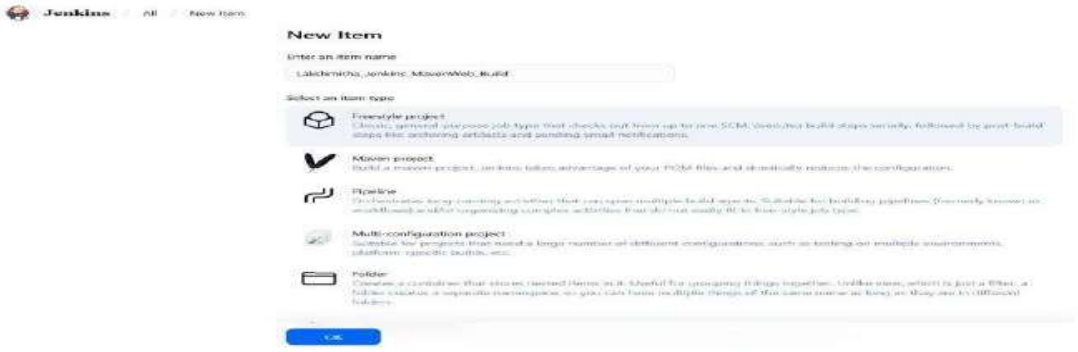
└── **Step 1:** Open Jenkins (localhost:8080)

├── Click on "New Item" (left side menu)

└── **Step 2**: Create Freestyle Project (e.g., MavenWeb_Build)

├── Enter project name (e.g., MavenWeb_Build)

├── Click "OK"



└── **Configure the project**:

├── **Description**: "Web Build demo"

├── **Source Code Management:**

└── Git repository URL: [GitMavenWeb repo URL]

├── *Branches to build*: */Main or master

└── **Build Steps**:

├── **Add Build Step** -> "Invoke top-level Maven targets"

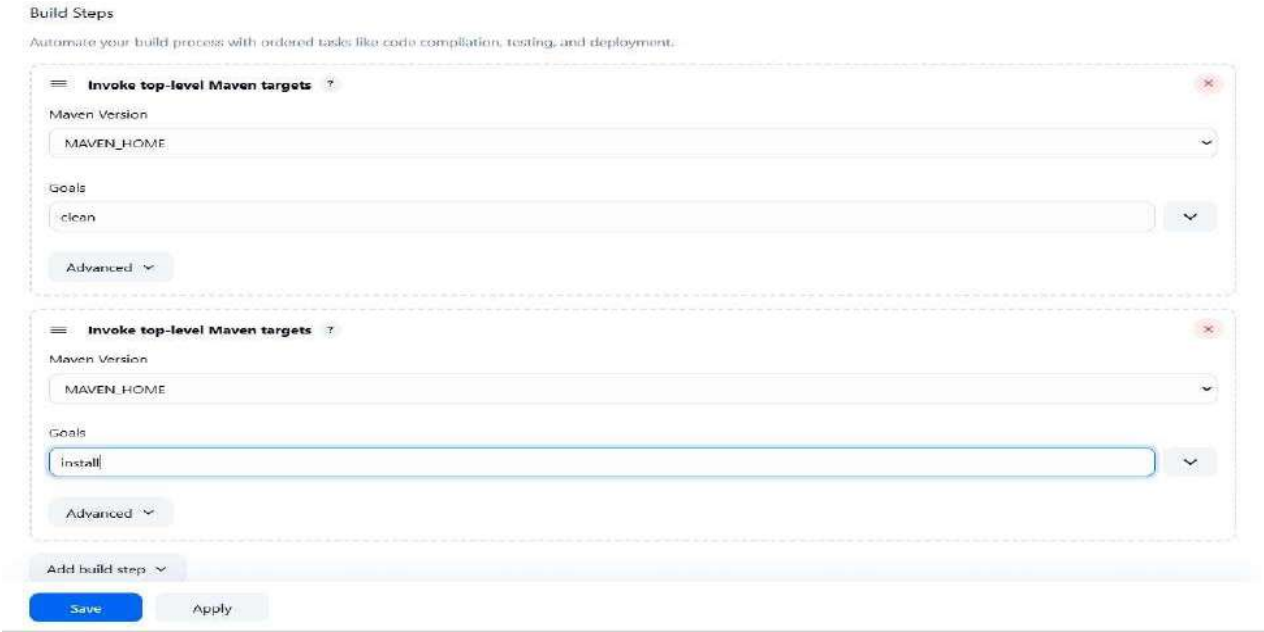
└── Maven version: MAVEN_HOME

└── Goals: clean

├── **Add Build Step** -> "Invoke top-level Maven targets"

└── Maven version: MAVEN_HOME

└── Goals: install



└── **Post-build Actions**:

├── **Add Post Build Action** -> "Archive the artifacts"

└── Files to archive: **/*

├── **Add Post Build Action** -> "Build other projects"

└── Projects to build: MavenWeb_Test

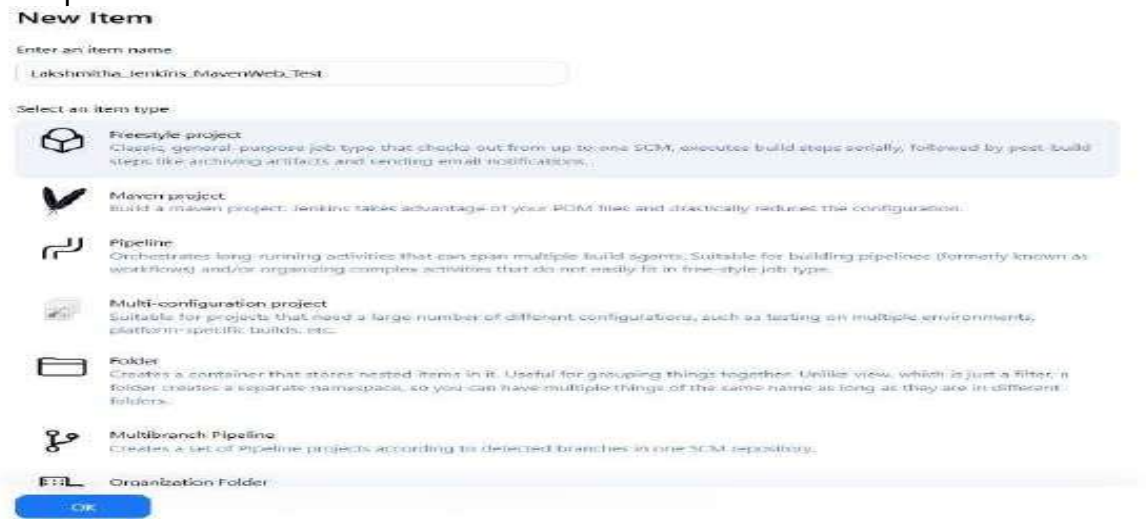
└── Trigger: Only if build is stable

└── Apply and Save

└── **Step 3**: Create Freestyle Project (e.g., MavenWeb_Test)

├── Enter project name (e.g., MavenWeb_Test)

├── Click "OK"



└── **Configure the project:**

├── **Description:** "Test demo"

├── **Build Environment**:

└── Check: "Delete the workspace before build starts"

├── **Add Build Step** -> "Copy artifacts from another project"

└── Project name: MavenWeb_Build

└── Build: Stable build only

└── Artifacts to copy: **/*

├── **Add Build Step** -> "Invoke top-level Maven targets"

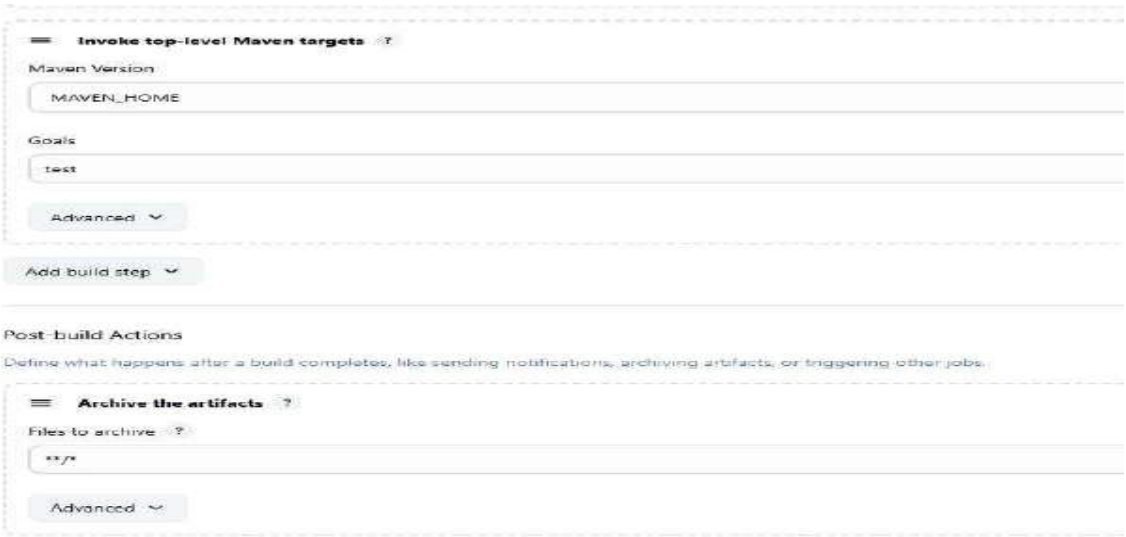
└── Maven version: MAVEN_HOME

└── Goals: test

└── **Post-build Actions**:

├── **Add Post Build Action** -> "Archive the artifacts"

└── Files to archive: **/*



├── **Add Post Build Action** -> "Build other projects"

└── Projects to build: MavenWeb_Deploy

└── Apply and Save

└── **Step 4**: Create Freestyle Project (e.g., MavenWeb_Deploy)

├── Enter project name (e.g., MavenWeb_Deploy)

├── Click "OK"

└── **Configure the projec**t:

├── **Description**: "Web Code Deployment"

├── **Build Environment**:

└── Check: "Delete the workspace before build starts"

├── **Add Build Step** -> "Copy artifacts from another project"

└── Project name: MavenWeb_Test

└── Build: Stable build only

└── Artifacts to copy: **/*

└── **Post-build Actions**:

├── **Add Post Build Action** -> "Deploy WAR/EAR to a container"

└── WAR/EAR File: **/*.war

└── Context path: Webpath

└── Add container -> Tomcat 9.x remote

└── Credentials: Username: admin, Password: 1234

── Tomcat URL: https://localhost:8085/

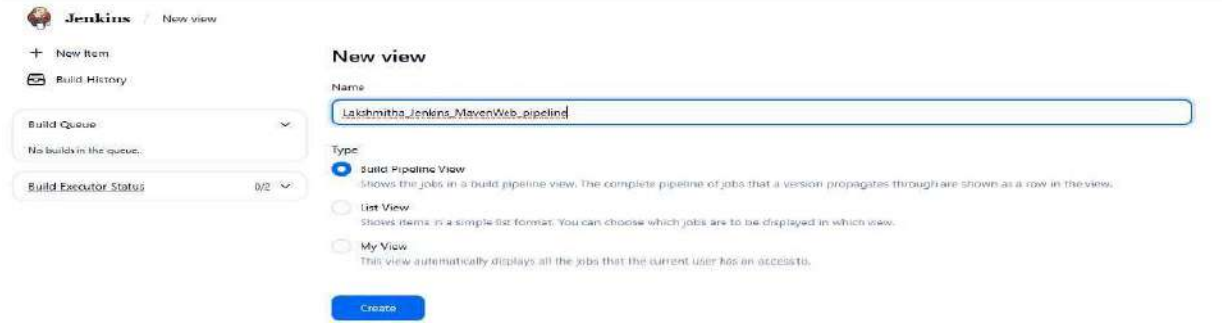
└── Apply and Save

└── **Step 5**: Create Pipeline View for MavenWeb

├── Click "+" beside "All" on the dashboard

├── Enter name: MavenWeb_Pipeline

├── **Select "Build pipeline view"**



└── **Pipeline Flow**:

├── **Layout**: Based on upstream/downstream relationship

├── Initial job: MavenWeb_Build

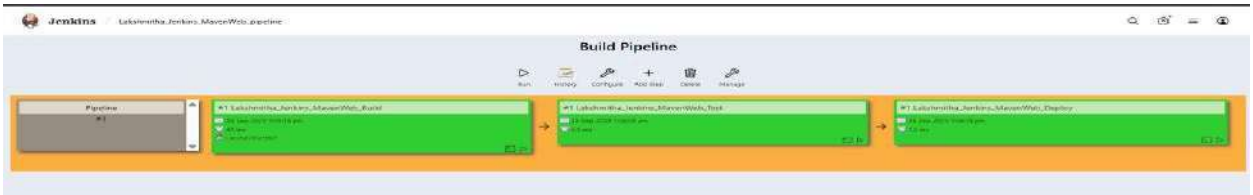
└── Apply and Save

└── **Step 6**: Run the Pipeline and Check Output

├── Click on the trigger **"RUN"** to run the pipeline

Note:

1. After Click on Run -> click on the small black box to open the console to check if the build is success
2. Now we see all the build has success if it appears in green color



├── Open Tomcat homepage in another tab

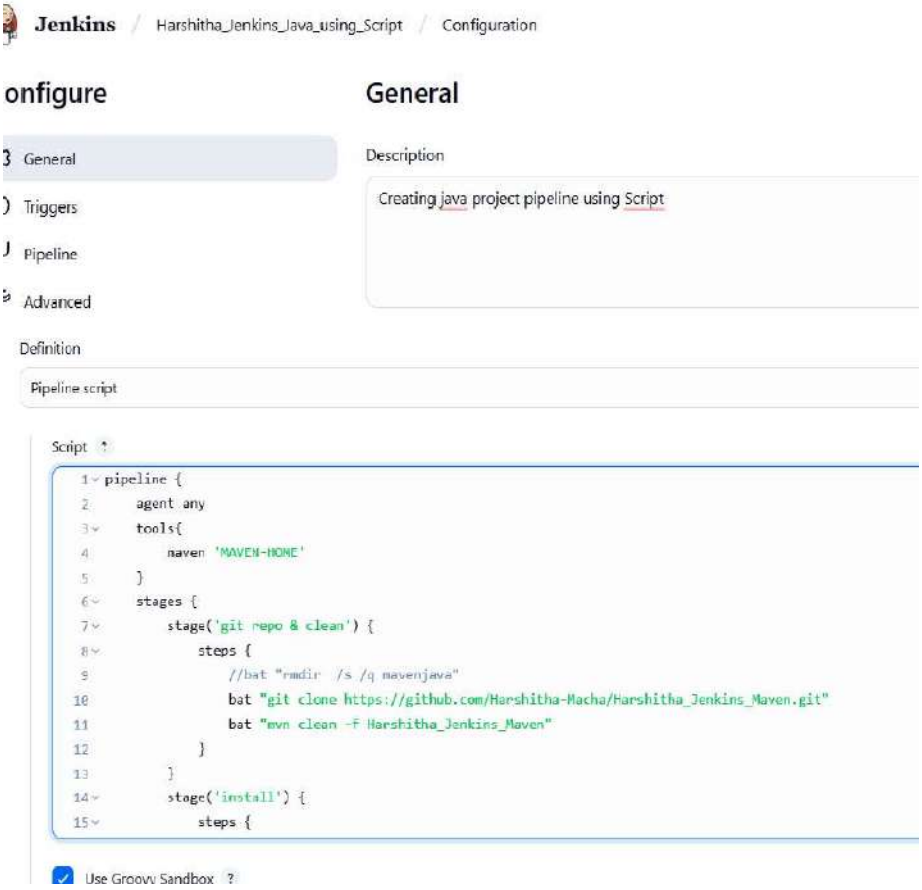
├── Click on the "/webpath" option under the manager app

Note:

1. It ask for user credentials for login ,provide the credentials of tomcat.
2. It provide the page with out project name which is highlighted.
3. After clicking on our project we can see output.

Experiment-9: Pipeline Creation using script

a. Evaluation of Jenkins pipeline.



Procedure

General :

Description :- provide the description of the project

Build triggers: here we can provide with build triggers of you choice

Advance project option:

Definition:

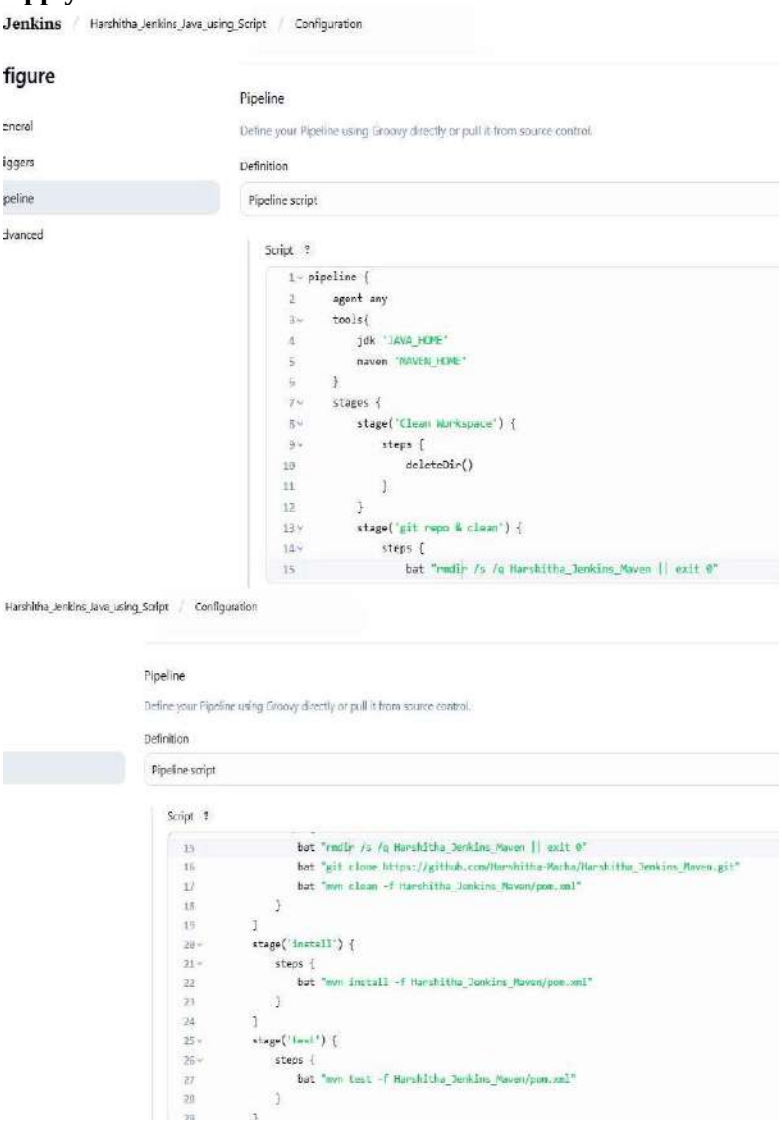
Choose pipeline script

CODE:

```
pipeline {
   agent any
   tools{
      jdk 'JAVA_HOME'
      maven 'MAVEN_HOME'
   }
   stages {
      stage('git repo & clean') {
         steps {
            bat "rmdir /s /q Harshitha_Jenkins_Maven || exit 0"
            bat "git clone https://github.com/Harshitha-Macha/Harshitha_Jenkins_Maven.git"
            bat "mvn clean -f Harshitha_Jenkins_Maven/pom.xml"
         }
      }
```

```
        stage('install') {
            steps {
                bat "mvn install -f Harshitha_Jenkins_Maven/pom.xml"
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f Harshitha_Jenkins_Maven/pom.xml"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f Harshitha_Jenkins_Maven/pom.xml"
            }
        }
    }
}
```

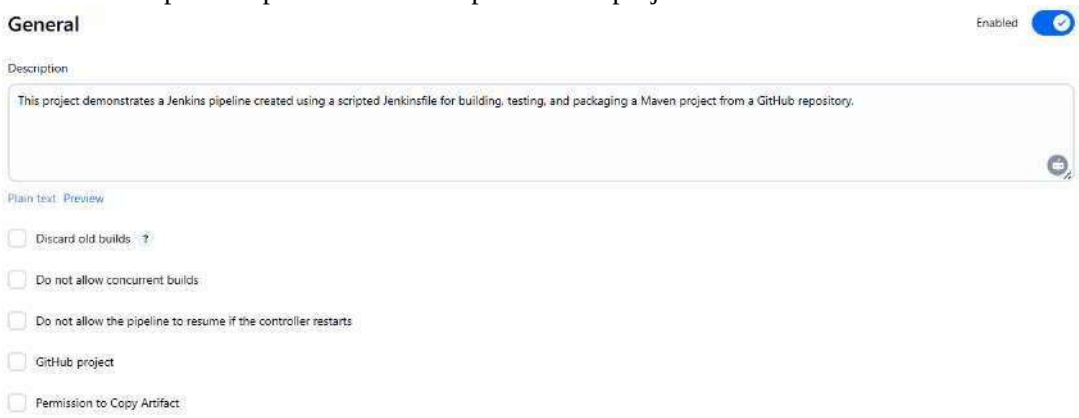
Apply and save



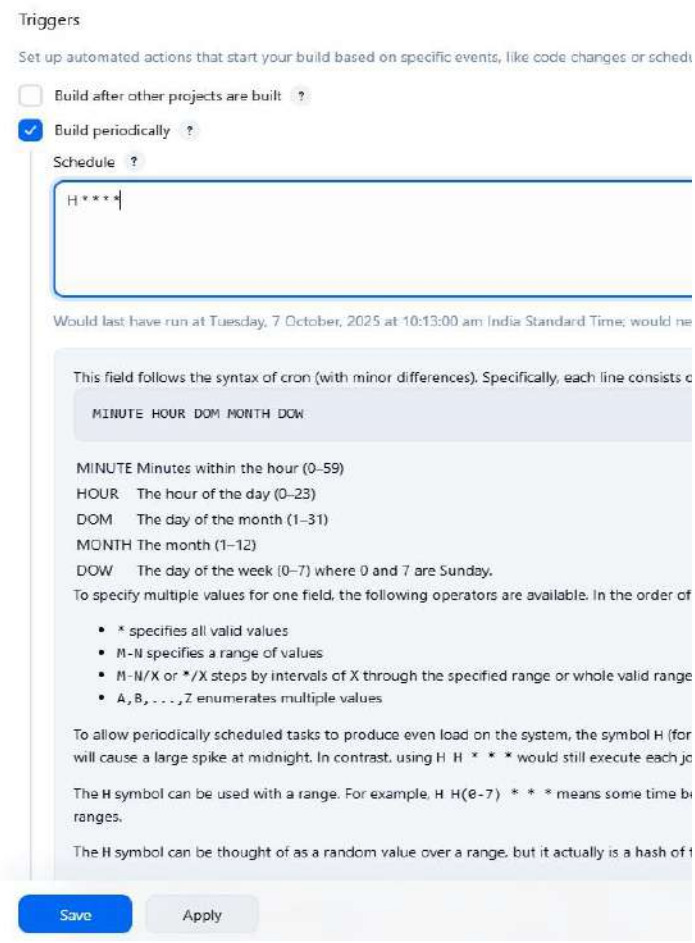
b. WORKING ON BUILD TRIGGERS FOR LAST JENKINS PIPILINE

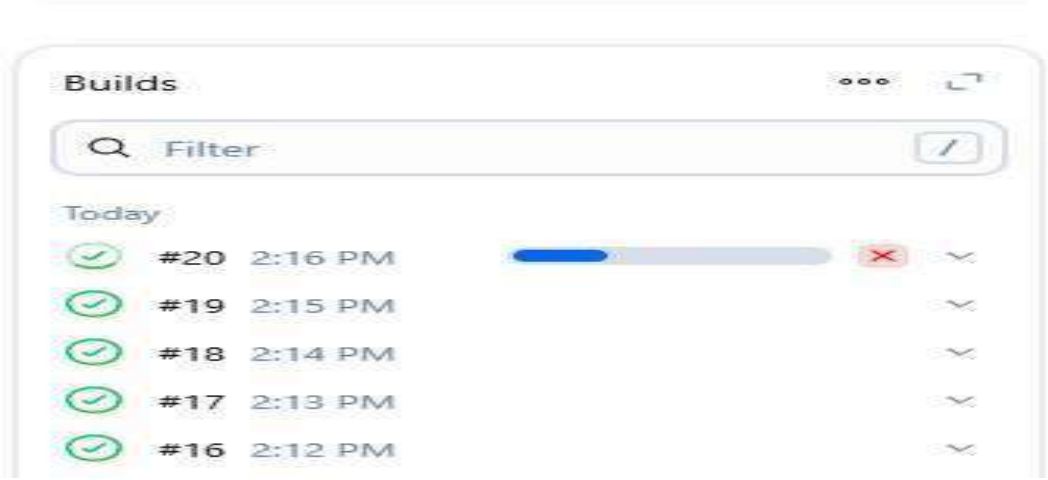
Procedure
General :

Description :- provide the description of the project



Build triggers: here we can provide with build triggers of you choice





Advance project option:

Definition:

Choose pipeline script

Here code for pipeline -script

Copy the code there

```
pipeline {
    agent any
    tools{
        maven 'MAVEN-HOME'
    }
    stages {
        stage('git repo & clean') {
            steps {
                //bat "rmdir  /s /q mavenjava"
                bat "git clone provide your github link"
                bat "mvn clean -f mavenjava"
            }
        }
        stage('install') {
            steps {
                bat "mvn install -f mavenjava" #project name#
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f mavenjava"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f mavenjava"
            }
        }
    }
}
```

Apply and save

c. Hands-on practice on creation of scripted Jenkins pipeline.

1. Your manager asks you to clean the workspace before building — which stage in this pipeline takes care of it?

   A) The “Git Repo & Clean” stage handles workspace cleaning. It can include a command like bat "rmdir /s /q Lakshmitha-Maven-JavaProject || exit 0" to delete old project files before cloning.

2. The GitHub repository link is missing in the script — where exactly do you provide it?

   A) Inside “Git Repo & Clean” stage, in line:
   bat " git clone https://github.com/LakshmithaReddy1807/Lakshmitha-Maven-JavaProject.git"

3. If Maven is not configured globally in Jenkins, which section of the pipeline will fail first?

   A) The pipeline will fail at the tools section during initialization, since tools { maven 'MAVEN-HOME' } depends on a globally configured Maven installation.

4. A teammate complains the pipeline is not creating .war files — which stage is responsible?

   A) The “Package” stage is responsible for creating .jar or .war files using mvn package.

5. Your test cases failed, but the pipeline still continued — how will Jenkins behave in the test stage?

   A) By default, Jenkins marks the Test stage as failed but continues to later stages unless error or failFast conditions are used.
   You can enforce a stop using: bat "mvn test -f project/pom.xml"
   error("Stopping pipeline since tests failed.")

6. Instead of running nightly builds, you want this pipeline to trigger only when GitHub changes occur — where will you configure it?

   A) In the Build Triggers section of the project configuration, select “GitHub hook trigger for GITScm polling”.

7. If you replace mvn clean with mvn compile, what difference will it make to the project build?

   A) mvn clean deletes previous build artifacts, ensuring a fresh build.
   mvn compile only compiles source code and doesn’t clean or remove old files.

8. The project folder name is not mavenjava but studentapp — which parts of the script must you edit?

   A) Every Maven command path and folder reference:
   mvn clean -f studentapp/pom.xml, mvn install -f studentapp/pom.xml, etc.

9. If the install stage fails, will the test and package stages still run in this pipeline?

   A) No. Jenkins Declarative Pipelines stop executing subsequent stages if any previous stage fails by default.

10. A student asks where to add deployment steps to Tomcat after packaging — which is the best place in this script?

    A) Add a new “Deploy” stage after the Package stage, for example:
    stage('Deploy') {

```
        steps {
            bat "mvn tomcat7:deploy -f project/pom.xml"
        }
    }
```

11. If GitHub credentials are private, how can you secure them in the git repo & clean stage?

    A) Store credentials in Jenkins using Credentials Manager, then use:
    withCredentials([usernamePassword(credentialsId: 'git-creds', usernameVariable: 'USER', passwordVariable: 'PASS')])
    {
    bat "git clone https://${USER}:${PASS}@github.com/username/repo.git"
    }

12. In Windows, the script uses bat commands — what change would you make if Jenkins runs on Linux?

    A) Replace all bat commands with sh commands.

13. Your Java file Hello.java must be compiled every time code is committed — how will you add this step?

    A) Add a pull step before building: bat "git -C Lakshmitha-Maven-JavaProject pull"

or delete and re-clone the repository each run.

14. Every evening at 6 PM your pipeline should run automatically — how can you set this in Jenkins?

    A) Under Build Triggers → Build periodically, enter the CRON expression: H 18 * * *

15. You only want to package the project if compilation succeeds — how would you connect the stages?

    A) In Declarative Pipelines, this is automatic — each stage depends on the previous one.
    But you can also use:
    when { success() }
    in the package stage to ensure it runs only on successful compilation.

16. Your professor wants a .jar file generated for submission — what pipeline stage will you add?

    A) The Package stage handles .jar creation using: mvn package

17. A Git clone step is failing due to wrong credentials — how would you secure and use them in your script?

    A) Store credentials in Jenkins Credentials Manager, then access them securely:
    withCredentials([usernamePassword(credentialsId: 'git-creds', usernameVariable: 'USER', passwordVariable: 'PASS')]) {
        bat "git clone https://${USER}:${PASS}@github.com/username/repo.git"
    }

18. A teammate wants to see the build number inside console logs — how do you print it in the pipeline?

    A) Use the Jenkins environment variable BUILD_NUMBER:
    echo "Running Build Number: ${env.BUILD_NUMBER}"

Experiment-10: Working with minikube and Nagios

a. Hands-on practice of creating, running and scaling pods in minikube.

Minikube Automation Steps

Step 1: Start Minikube Cluster

- Open your terminal and run the command:

```
minikube start
```

Step 2: Create and Manage Deployment

1. **Create an application in Kubernetes:**

   - Command:

kubectl create deployment mynginx --image=nginx
if already created then kubectl set image deployment/myngnix nginx=nginx:latest

```
PS C:\WINDOWS\system32> kubectl set image deployment/mynginx nginx=nginx:latest
>>
deployment.apps/mynginx image updated
PS C:\WINDOWS\system32> kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mynginx   1/1     1            1           11m
```

   - Verify the deployment using: Kubernetes responds by showing you a list that includes the names of your deployment groups

```
kubectl get deployments
```

     - Ensure `mynginx` appears in the output.

Check the following commands:

- kubectl get pods

```
PS C:\WINDOWS\system32> kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mynginx-5d6b79544c-pjg5c    1/1     Running   0          53s
```

- kubectl describe pods

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  94s   default-scheduler  Successfully assigned default/mynginx-5d6b79544c-pjg5c to minikube
  Normal  Pulling    93s   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     90s   kubelet            Successfully pulled image "nginx:latest" in 2.855s (2.855s includin
159974475 bytes.
  Normal  Created    90s   kubelet            Created container: nginx
  Normal  Started    90s   kubelet            Started container nginx
```

2. **Expose Deployment as a Service**:
   - Command:
     `kubectl expose deployment mynginx --type=NodePort --port=80 --target-port=80`

```
PS C:\WINDOWS\system32> kubectl expose deployment mynginx --type=NodePort --port=80 --target-port=80
service/mynginx exposed
```

Step 3: Scale the Deployment
Command:Scales the Nginx deployment to 4 replicas (pods).

```
kubectl scale deployment mynginx --replicas=4
```

```
PS C:\WINDOWS\system32> kubectl scale deployment mynginx --replicas=4
deployment.apps/mynginx scaled
```

```
kubectl get service myngnix
```

```
PS C:\WINDOWS\system32> kubectl get svc
>>
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        51m
mynginx      NodePort    10.111.250.201  <none>        80:31149/TCP   100s
```

b.

c. Running Nginx server on specified port number by explaining the Nginx monitoring tool

Access the Nginx App

1. **Using Port Forwarding**:
   - Command:
     `kubectl port-forward svc/mynginx 8081:80`

```
PS C:\WINDOWS\system32> kubectl port-forward svc/mynginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
Handling connection for 8081
```

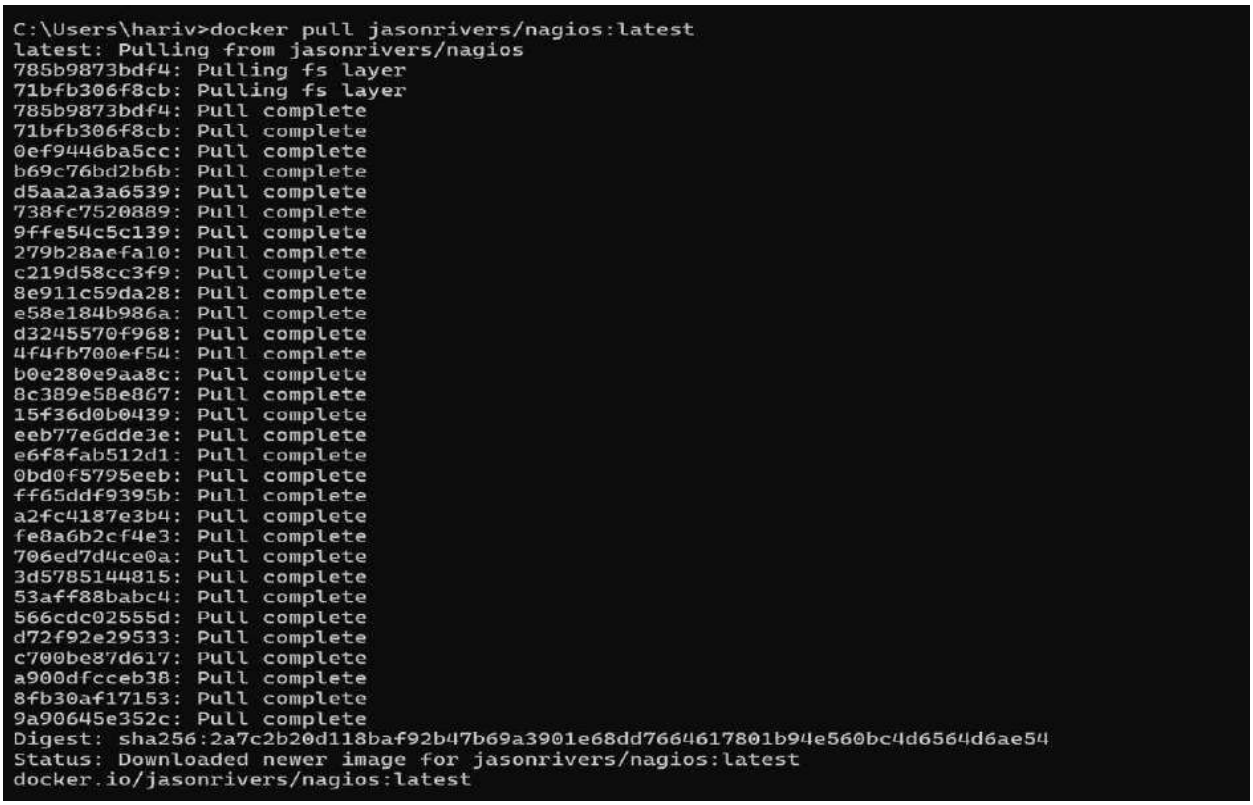
Access the app via http://localhost:8081.

d. Running Nagios server and Understanding the Monitoring tool using Docker

Step 1: Pull the Nagios Docker Image

- Open a terminal and run:

```
docker pull jasonrivers/nagios:latest
```



Step 2: Run Nagios

- Command:

```
docker run --name nagiosdemo -p 8888:80 jasonrivers/nagios:latest
```

```
C:\Users\hariv>docker run --name nagiosdemo -p 8888:80 jasonrivers/nagios:latest
Adding password for user nagiosadmin
chown: warning: '.' should be ':': 'nagios.nagios'
Started runsvdir, PID is 13
checking permissions for nagios & nagiosgraph
rsyslogd: [origin software="rsyslogd" swVersion="8.2312.0" x-pid="20" x-info="https://www.rsyslog.com"] start

Nagios Core 4.5.7
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2024-10-24
License: GPL

Website: https://www.nagios.org
Nagios 4.5.7 starting... (PID=27)
Local time is Tue Oct 14 15:29:16 UTC 2025
nagios: Nagios 4.5.7 starting... (PID=27)
nagios: Local time is Tue Oct 14 15:29:16 UTC 2025
nagios: LOG VERSION: 2.0
wproc: Successfully registered manager as @wproc with query handler
nagios: qh: Socket '/opt/nagios/var/rw/nagios.qh' successfully initialized
nagios: qh: core query handler registered
nagios: qh: echo service query handler registered
nagios: qh: help for the query handler registered
nagios: wproc: Successfully registered manager as @wproc with query handler
wproc: Registry request: name=Core Worker 42;pid=42
wproc: Registry request: name=Core Worker 46;pid=46
nagios: wproc: Registry request: name=Core Worker 42;pid=42
wproc: Registry request: name=Core Worker 45;pid=45
nagios: wproc: Registry request: name=Core Worker 46;pid=46
nagios: wproc: Registry request: name=Core Worker 45;pid=45
wproc: Registry request: name=Core Worker 44;pid=44
wproc: Registry request: name=Core Worker 43;pid=43
```

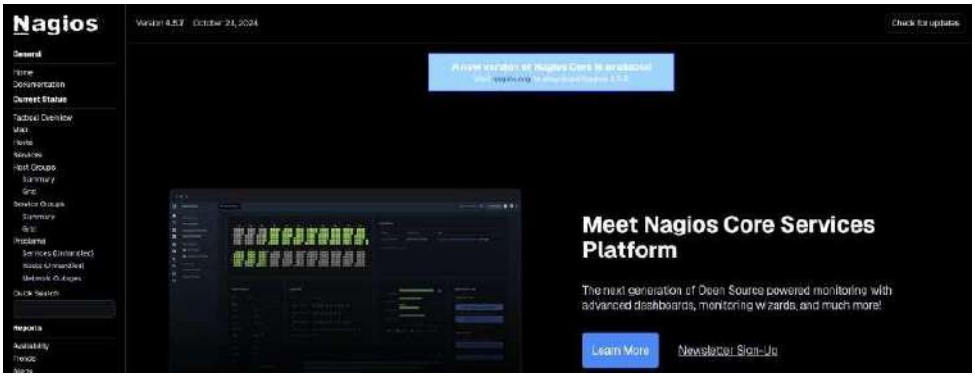
Step 3: Access Nagios Dashboard

- Open your browser and navigate to:

  `http://localhost:8888`

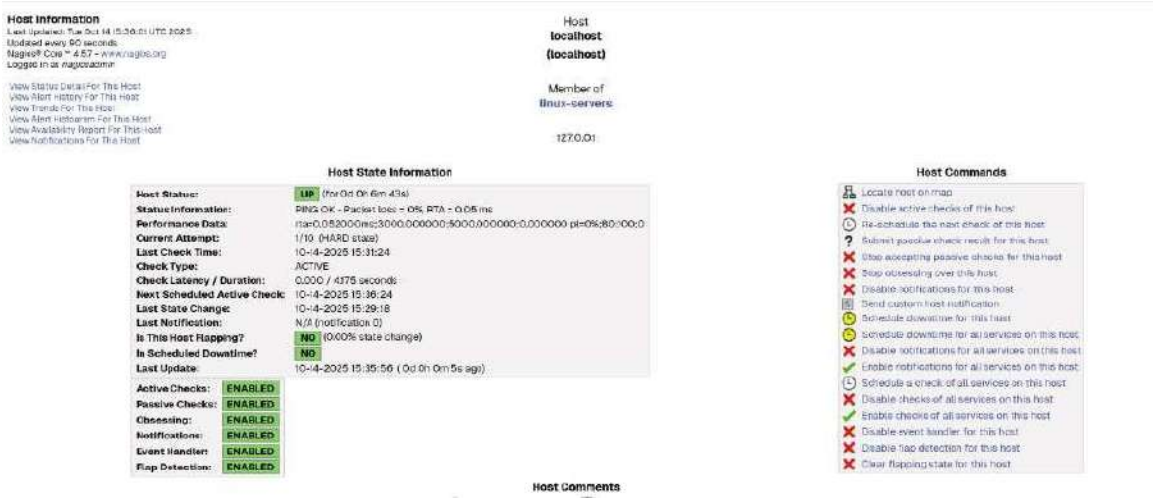
  - **Login Credentials**:
    - Username: `nagiosadmin`
    - Password: `nagios`

  - Once logged in, explore:
    - Hosts: View systems being monitored.
    - Services: Check tasks being monitored (e.g., CPU usage).
    - Alerts: Access recent notifications.

- 

Step 4: Monitoring Host Details

1. **Navigate to the Host Information Page**:
    - Select a host from the **Hosts** menu.
2. **Key Details**:
    - Host Status: Indicates if the system is UP or DOWN.
    - Metrics: View CPU usage, memory status, and network activity.
    - Actions: Reschedule checks, disable notifications, or schedule downtime.



Step 5: Stop and Remove Nagios

1. **Stop the Container**:
    - Command:

```
docker stop nagiosdemo
```

```
C:\Users\hariv>docker stop nagiosdemo
nagiosdemo
```

2. **Delete the Container**:
    - Command:

```
docker rm nagiosdemo
```

3. **Remove the Image** (Optional):
    - List images:

```
docker images
```

    - Delete the Nagios image:

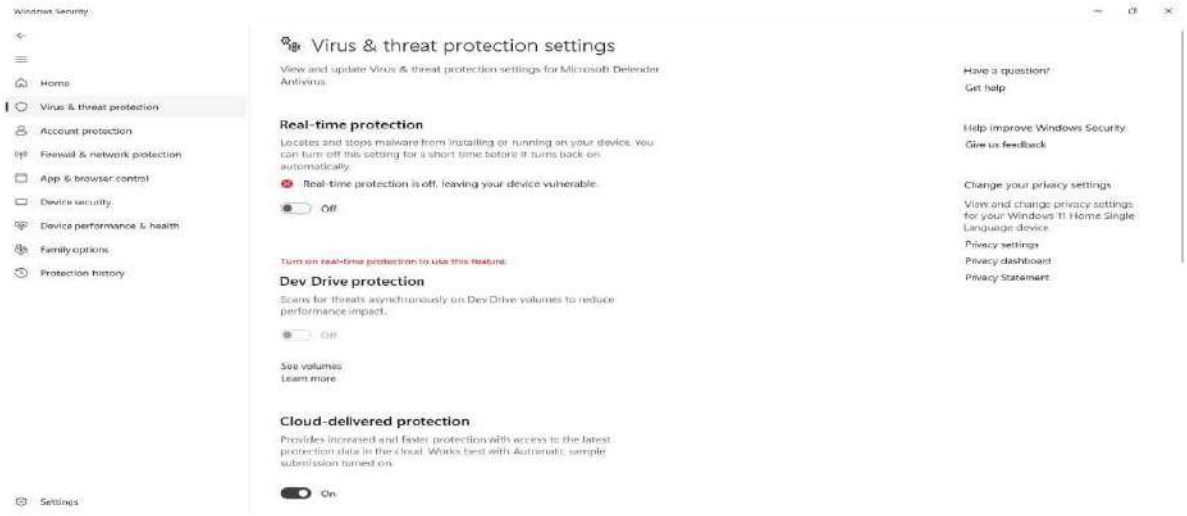
```
docker rmijasonrivers/nagios:latest
```

4. Observe the docker containers in DockerHub, we can see the latest Nagios Installed running on port:8888

Experiment-11: Jenkins-CI/CD

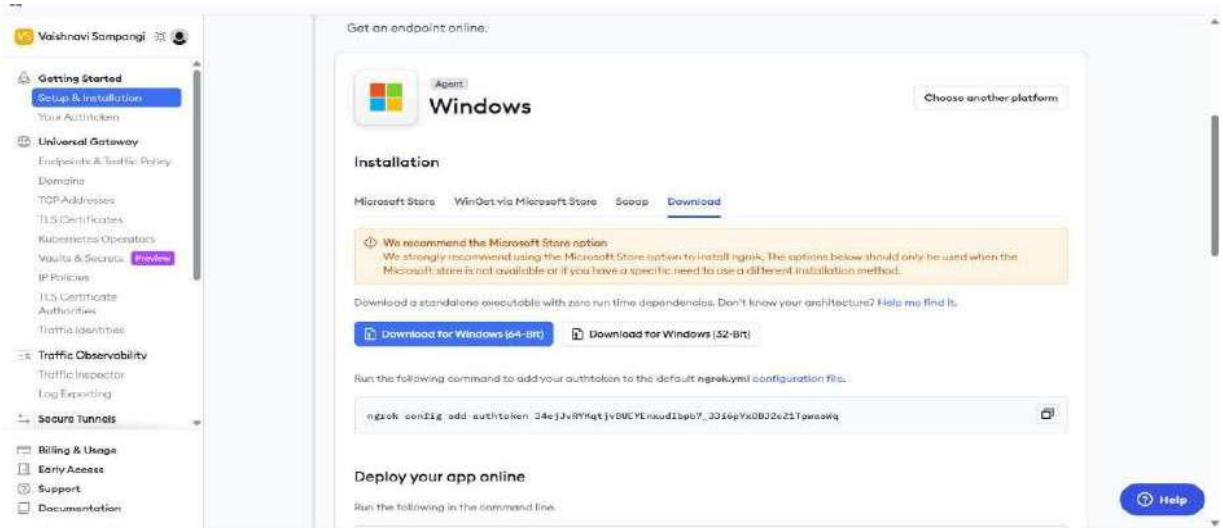
a. CI-Continuous Integration using Webhooks.

Step-1: To install ngroks Go->settings->privacy and security -> windows security -> off antivirus ->

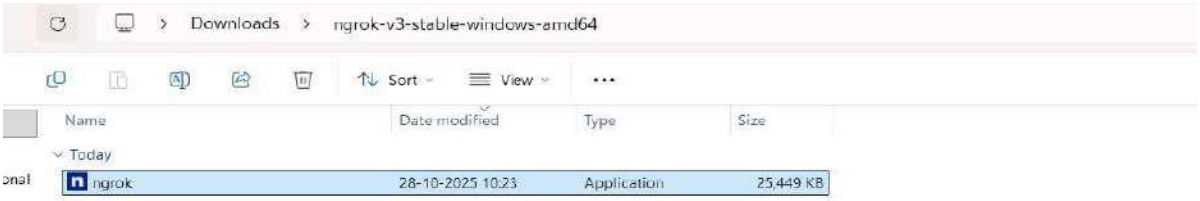


Step-2:Go -> https://ngrok.com and signup by giving your name ,email and password of atleast 10 charcaters

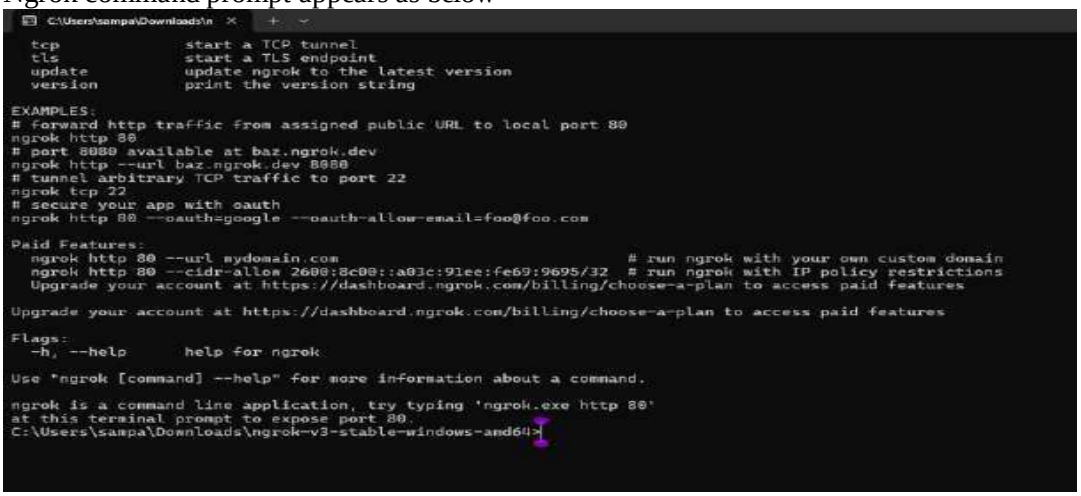
Step-3: After sign in up ,it will show below screen with your name in the top left .now click on download for windows (64Bit) to download ngork



Step4:After downloading ,Extract the file and click on ngrok.exe



Ngrok command prompt appears as below



Step-5:Connect Your ngrok Account (optional but useful)

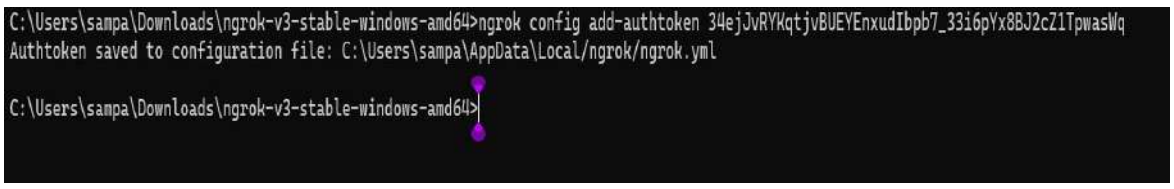
- Go to ngrok gives you an auth token.
- Then go to your Authtoken click here
- Copy your Authtoken

CREATE AUTHENTICATOR [https://dashboard.ngrok.com/get-started/your-authtoken]

Run this command in ngrok command prompt:(replace <your_token> with yours):

ngrok config add-authtoken <your_token> // syntax:

Example command



Step-6

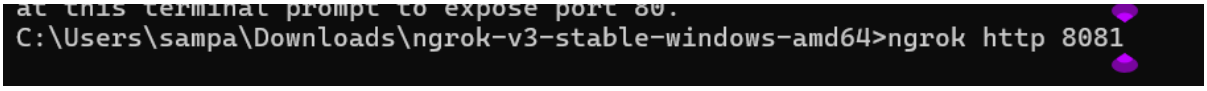
Start a Tunnel for Jenkins

- Check on which port is your Jenkins running . for this give in browers or url localhost:8081

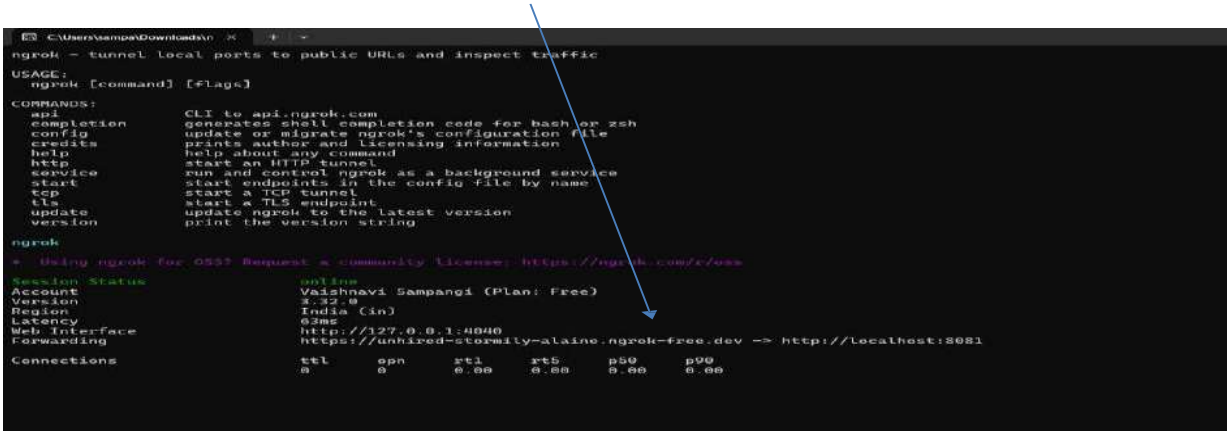
For me Jenkins is running on 8081

- Go to ngrok command prompt and type below command
- ngork http 8081 //Always use this command to start a tunnel for jenkins .

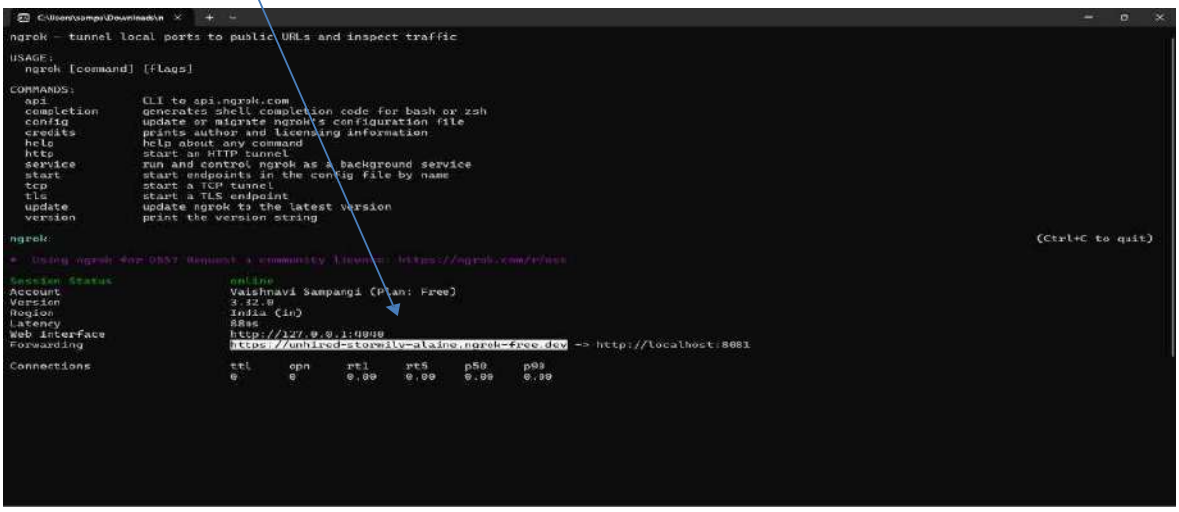
Type in ngrok command prompt:



Next it shows this public jenkins URL generated by ngrok that can be pasted into github repo for Webhooks.



Copy this URL only highlighted part

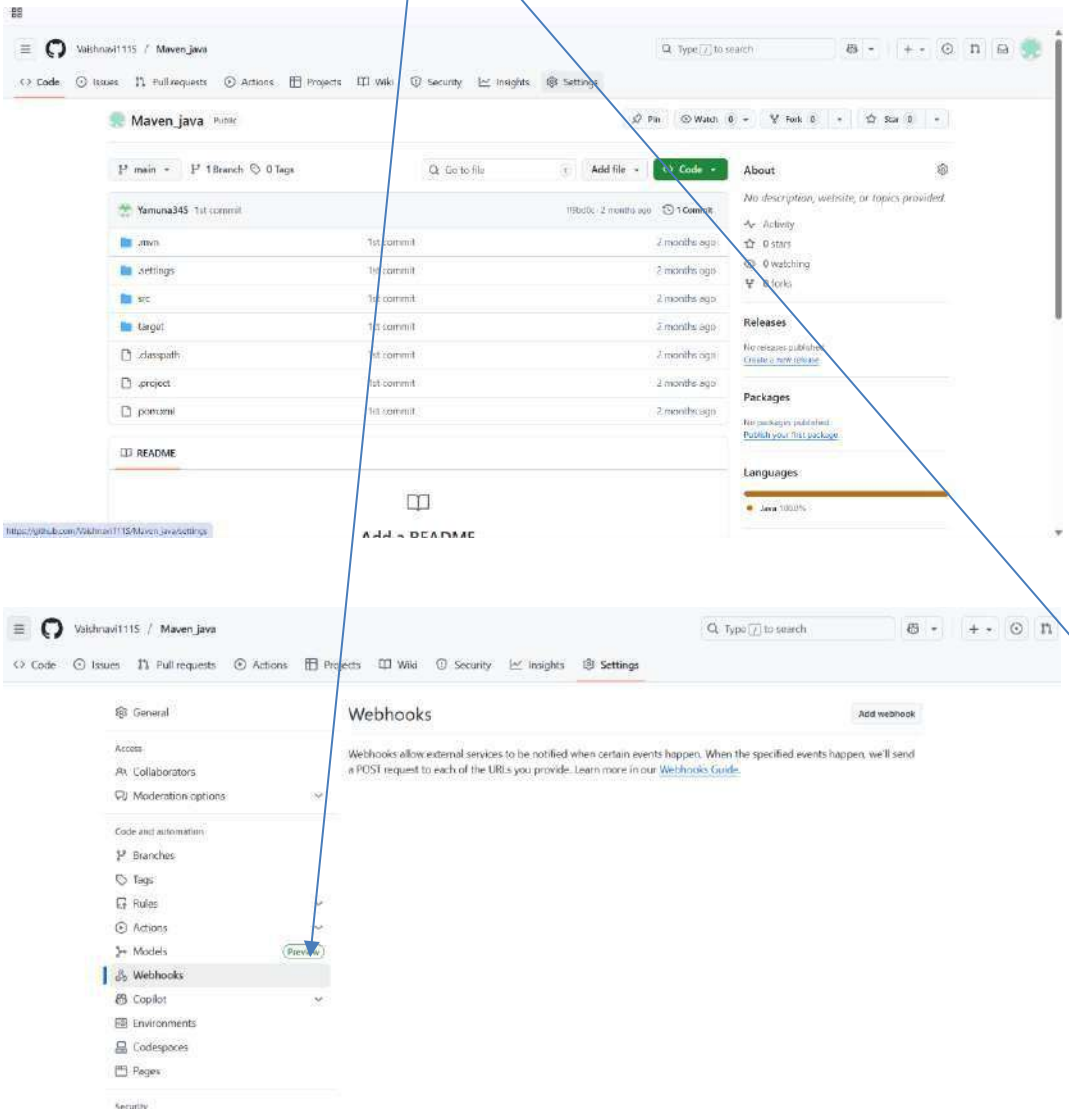


Step-7: Configure Webhook in GitHub

1. Go to your GitHub repository.
2. Navigate to Settings → Webhooks.
3. Click “Add webhook”.
4. In the Payload URL field:
    - Enter the Jenkins webhook URL in the format: http://<jenkins-server-url>/github-webhook/

Ex: https://unhired-stormily-alaine.ngrok-free.dev/github-webhook/

Note: If Jenkins is running on localhost, GitHub cannot access it directly

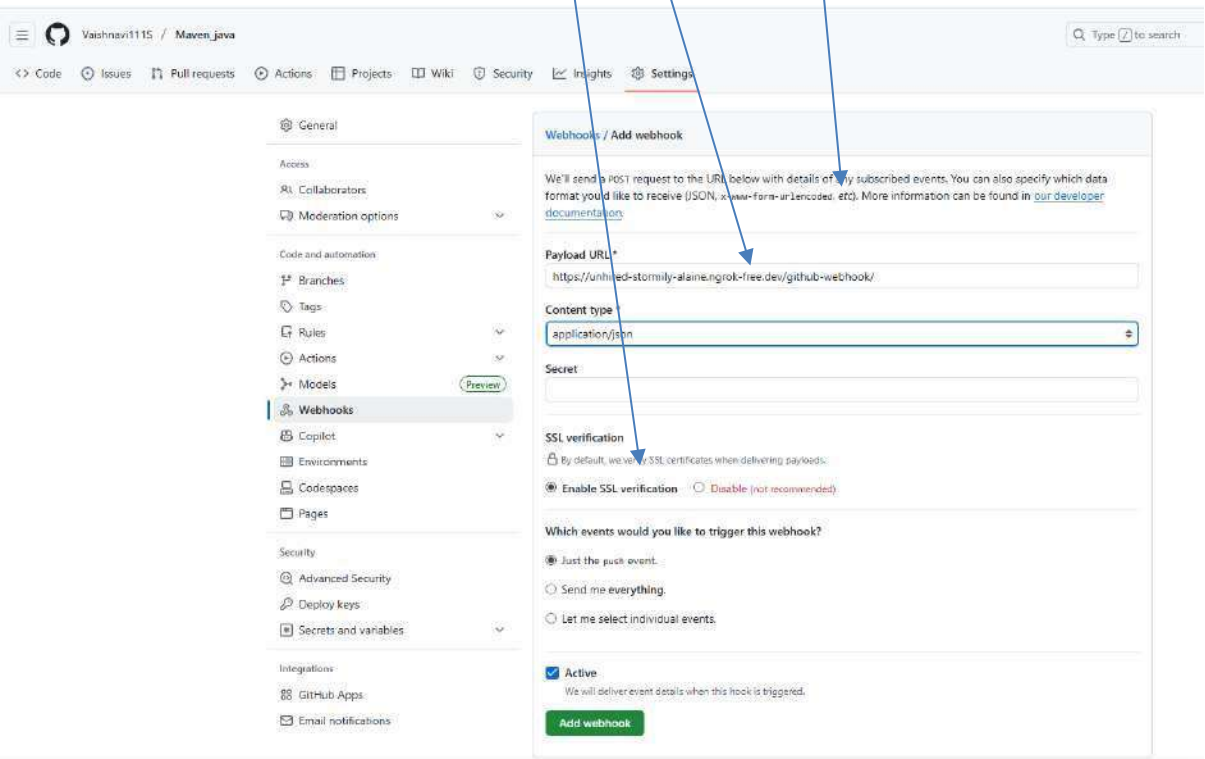


Step-8:

- Add url https://unhired-stormily-alaine.ngrok-free.dev/github-webhook/
- Set content Type to application/json
- Under “Which events would you like to trigger this webhook?”, select:

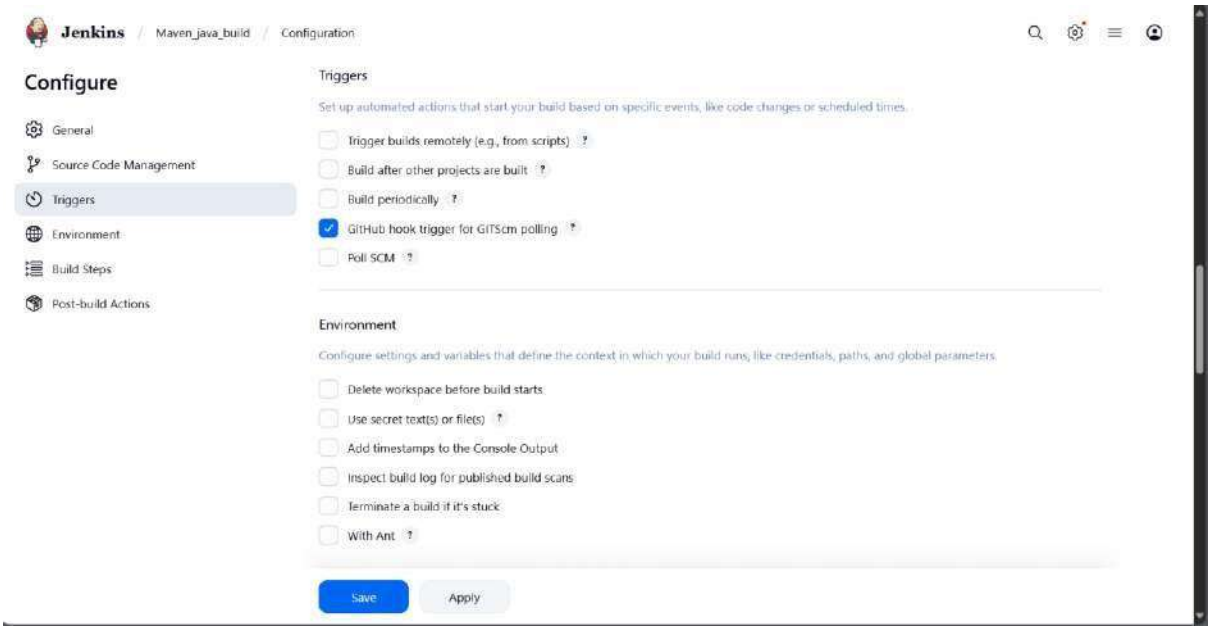
Just the push event.

- Click “Add webhook” to save.



Step 10: Configure Jenkins to Accept GitHub Webhooks

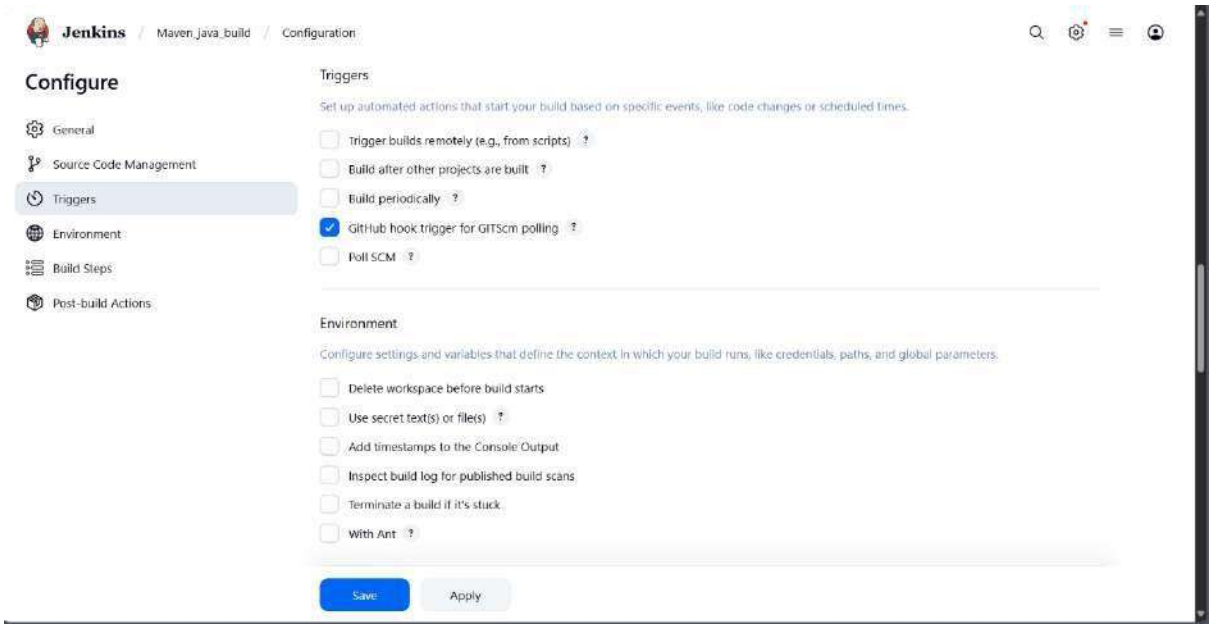
1. Open Jenkins Dashboard.
2. Select the job (freestyle or pipeline) you've already created.
3. Click Configure.
4. Scroll down to the Build Triggers section.
5. Check the box: ✅ GitHub hook trigger for GITScm polling

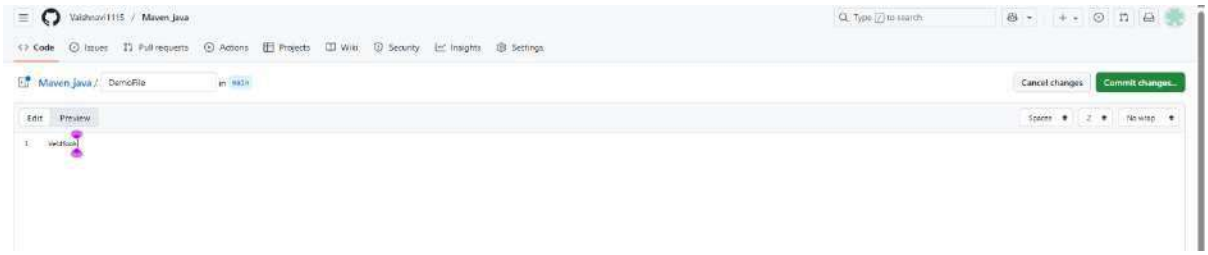


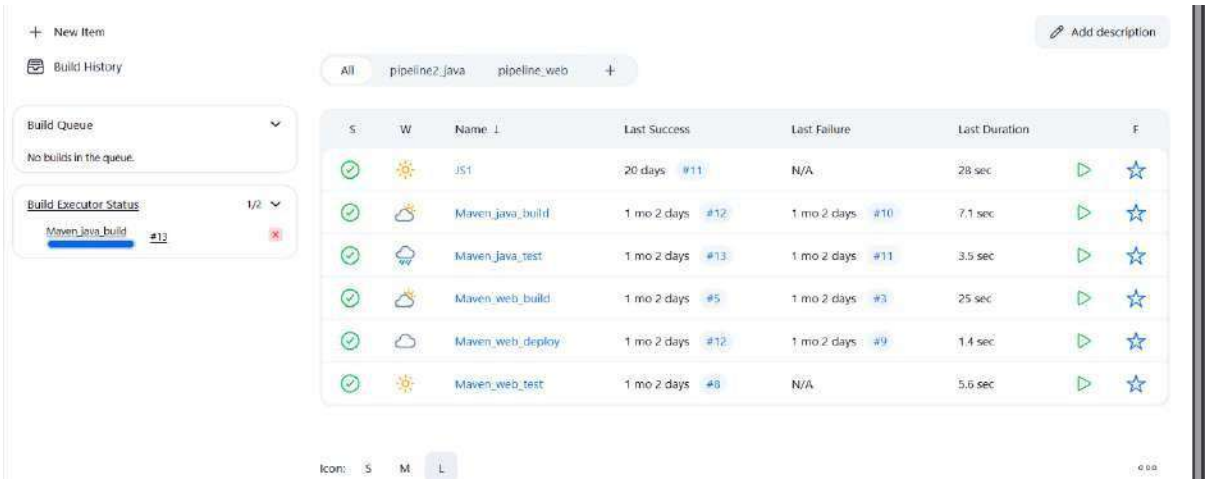
6. Click Save.

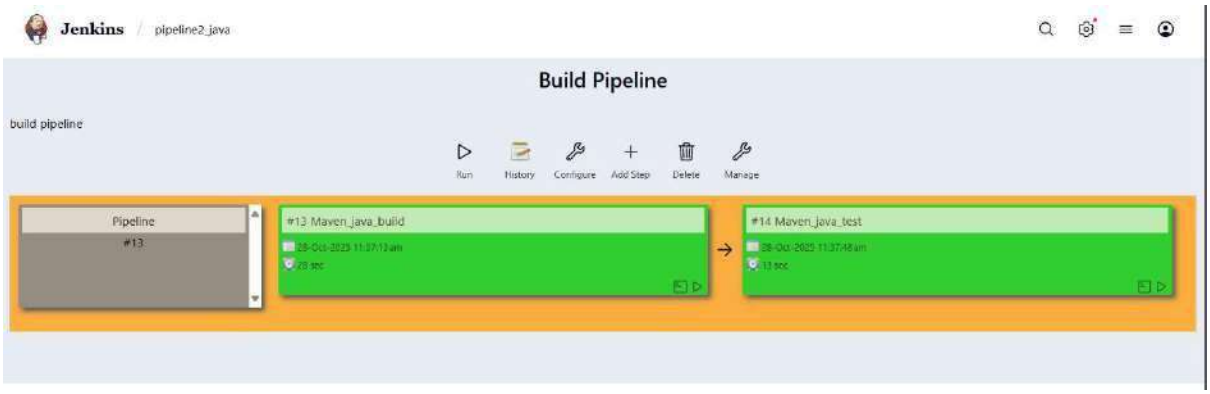
Step 11: Test the Setup

1. Make any code update in your local repo and push it to GitHub.
2. Once pushed, GitHub will trigger the webhook.
3. Jenkins will automatically detect the change and start the build pipeline.









---

outcome

- You've successfully connected GitHub and Jenkins using webhooks.
- Every time you push code to GitHub, Jenkins will automatically start building your project without manual intervention.

b. Sending E-mail Notification on Build Failure or success

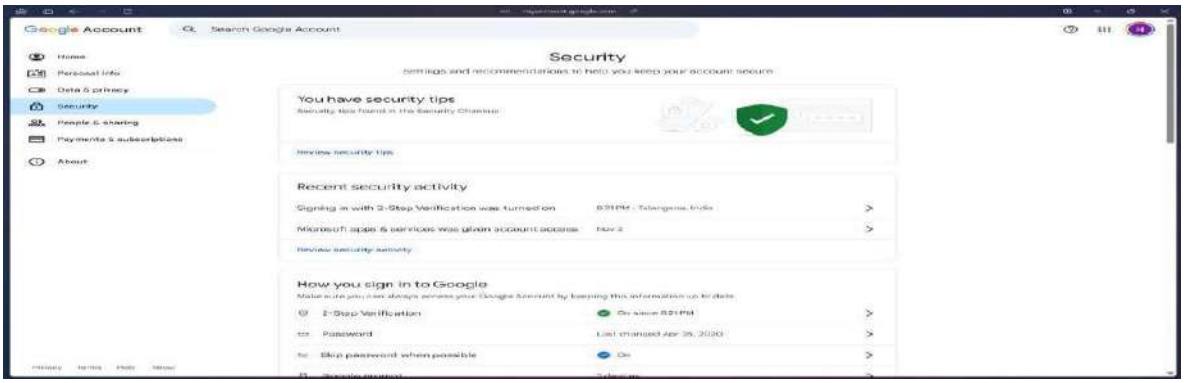
Creation of app password

1. Gmail: Enable App Password (for 2-Step Verification)

i. Go to: https://myaccount.google.com

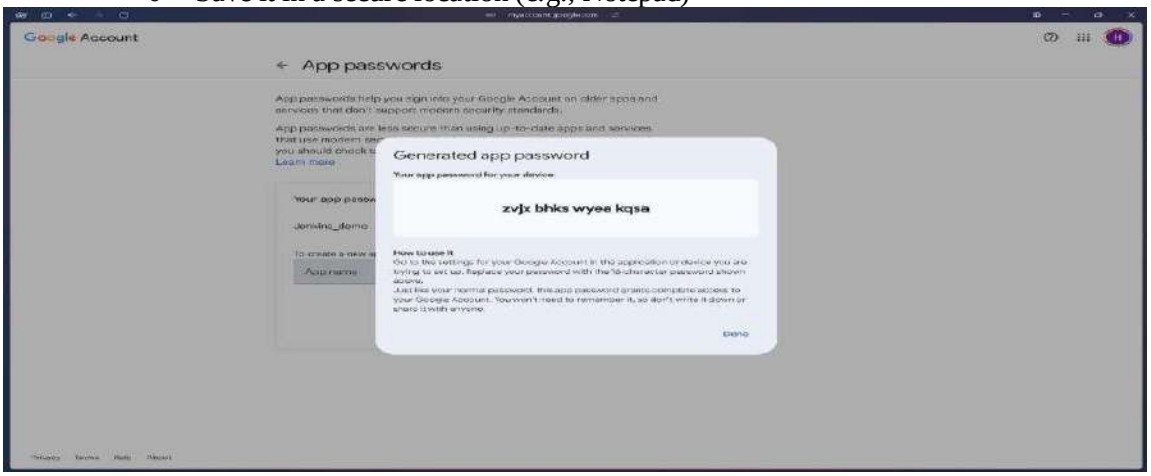
ii. Enable 2-Step Verification

- Navigate to:
    - Security → 2-Step Verification
    - Turn it ON
    - Complete the OTP verification process (via phone/email)



iii. Generate App Password for Jenkins

- Go to:
    - Security → App passwords
- Select:
    - App: Other (Custom name)
    - Name: Jenkins-Demo
- Click Generate
- Copy the 16-digit app password
    - Save it in a secure location (e.g., Notepad)



2. Jenkins Plugin Installation

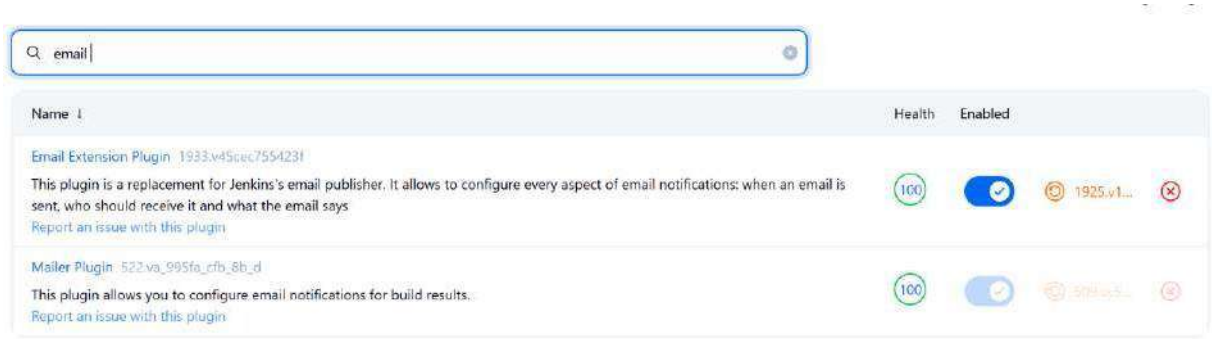
i. Open Jenkins Dashboard

ii. Navigate to:

- Manage Jenkins → Manage Plugins

iii. Install Plugin:

- Search for and install:
    - Email Extension Plugin



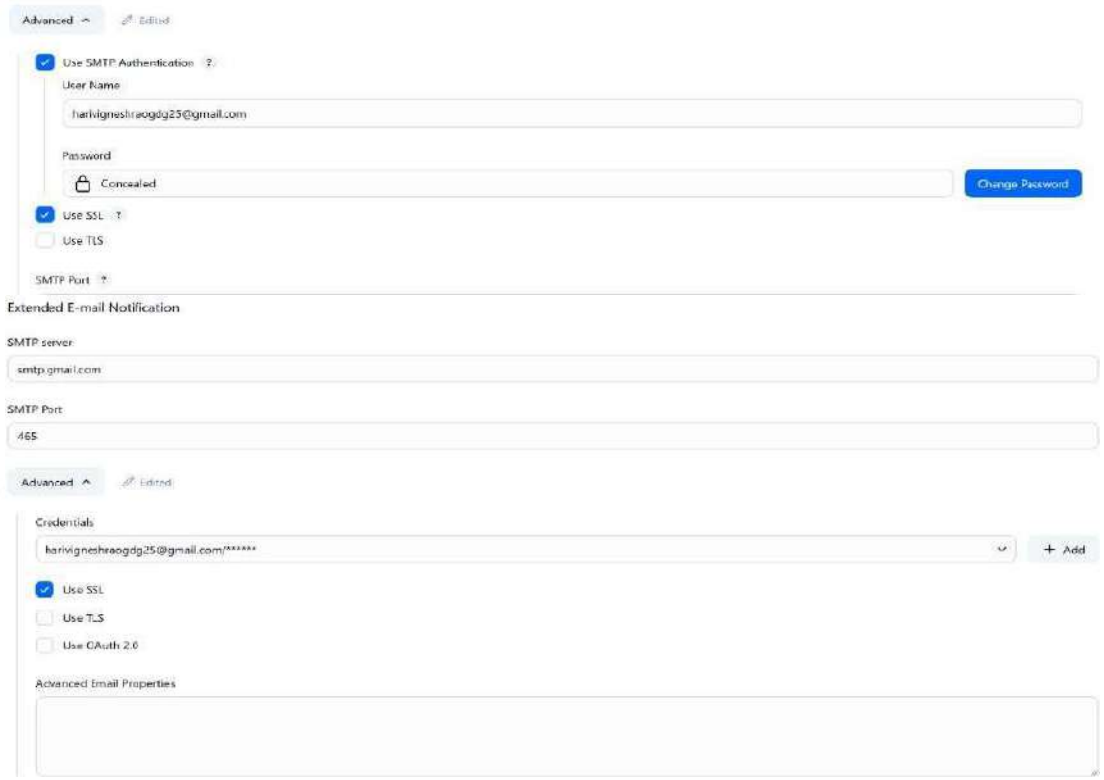
3. Configure Jenkins Global Email Settings

i. Go to:

- Manage Jenkins → Configure System

A. E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
Use SMTP Auth	■ Enabled
User Name	Your Gmail ID (e.g., archanareddykmit@gmail.com)
Password	Paste the 16-digit App Password
Use SSL	■ Enabled
SMTP Port	465
Reply-To Address	Your Gmail ID (same as above)



Test Configuration

- Click: Test configuration by sending test e-mail
- Provide a valid email address to receive a test mail
- ■ Should receive email from Jenkins

B. Extended E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
SMTP Port	465
Use SSL	■ Enabled
Credentials	Add Gmail ID and App Password as Jenkins credentials
Default Content Type	text/html or leave default
Default Recipients	Leave empty or provide default emails
Triggers	Select as per needs (e.g., Failure)

4. Configure Email Notifications for a Jenkins Job

i. Go to:

- Jenkins → Select a Job → Configure

ii. In the Post-build Actions section:

- Click: Add post-build action → Editable Email Notification

A. Fill in the fields:

Field	Value
Project Recipient List	Add recipient email addresses (comma-separated)

Field	Value
Content Type	Default (text/plain) or text/html
Triggers	Select events (e.g., Failure, Success, etc.)
Attachments	(Optional) Add logs, reports, etc.

iii. Click Save

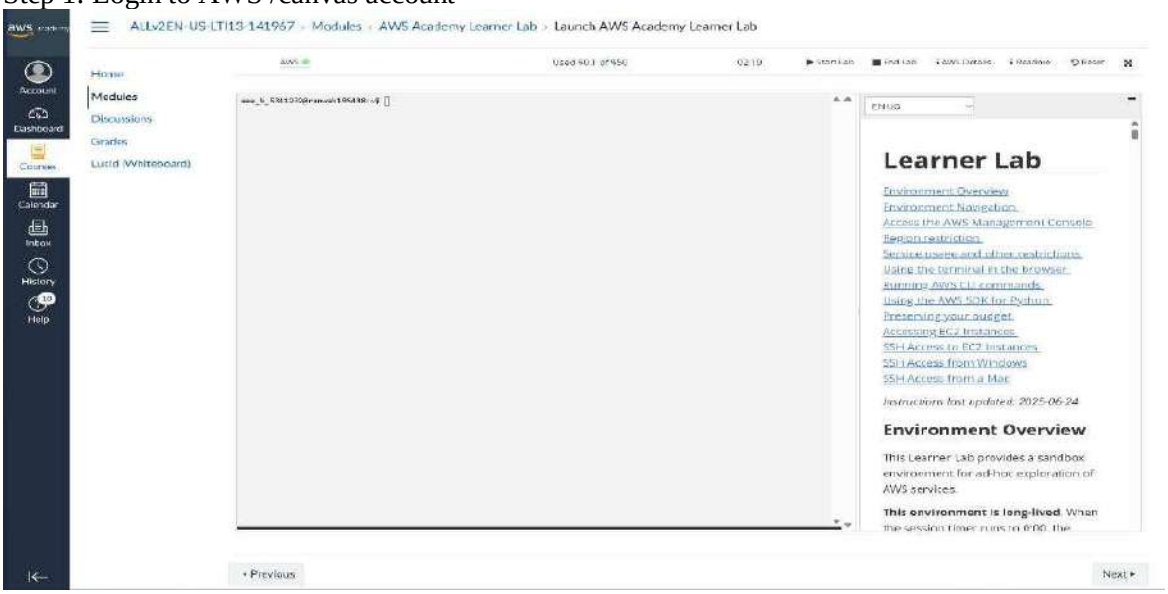
Now your Jenkins job is set up to send email notifications based on the build status!

Experiment-12: Creation of virtual machine for Ubuntu OS and Deploying the web application

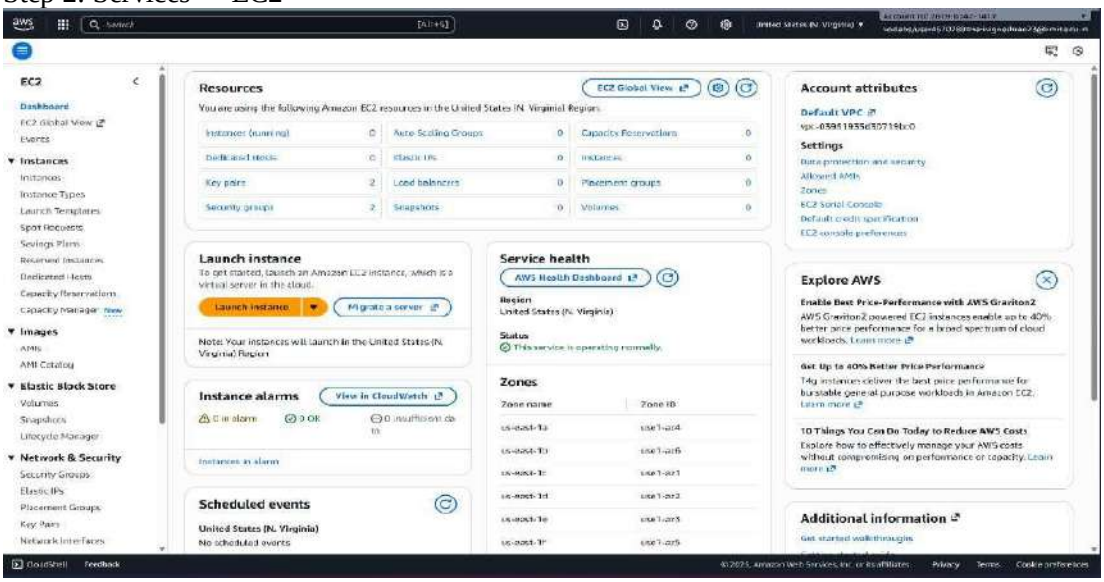
1. Creation of virtual machine

Ex : Launch ubuntu instnace

Step 1: Login to AWS /canvas account



Step 2: Services -- EC2

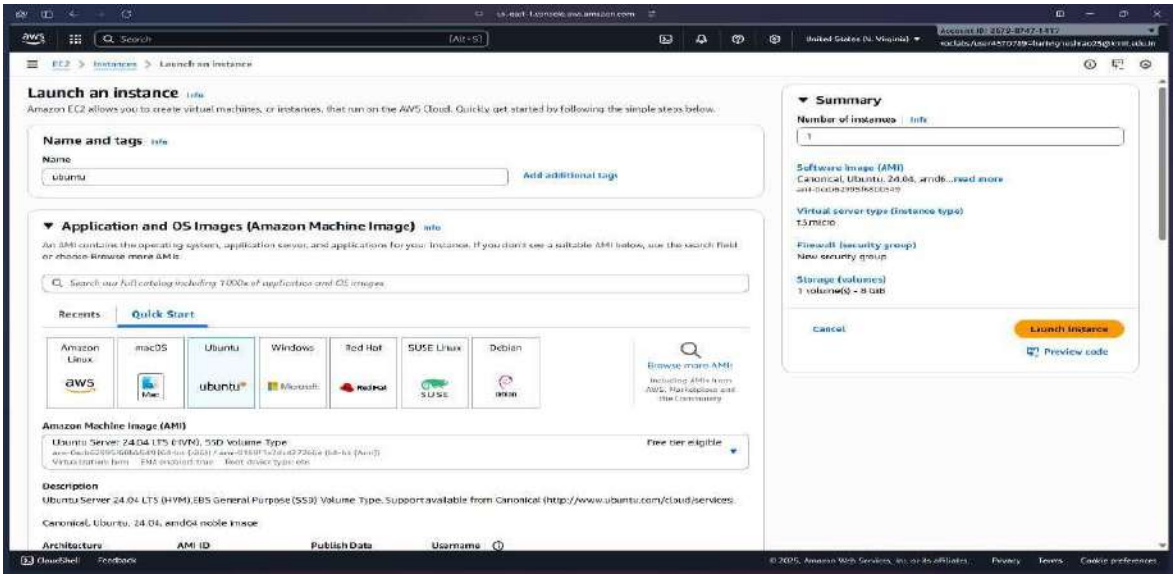


Step 3: Choose region which is near ?

Services -- EC2 --- Launch Instance

Stage 1 --Name (Giving name to the machine) ubuntu

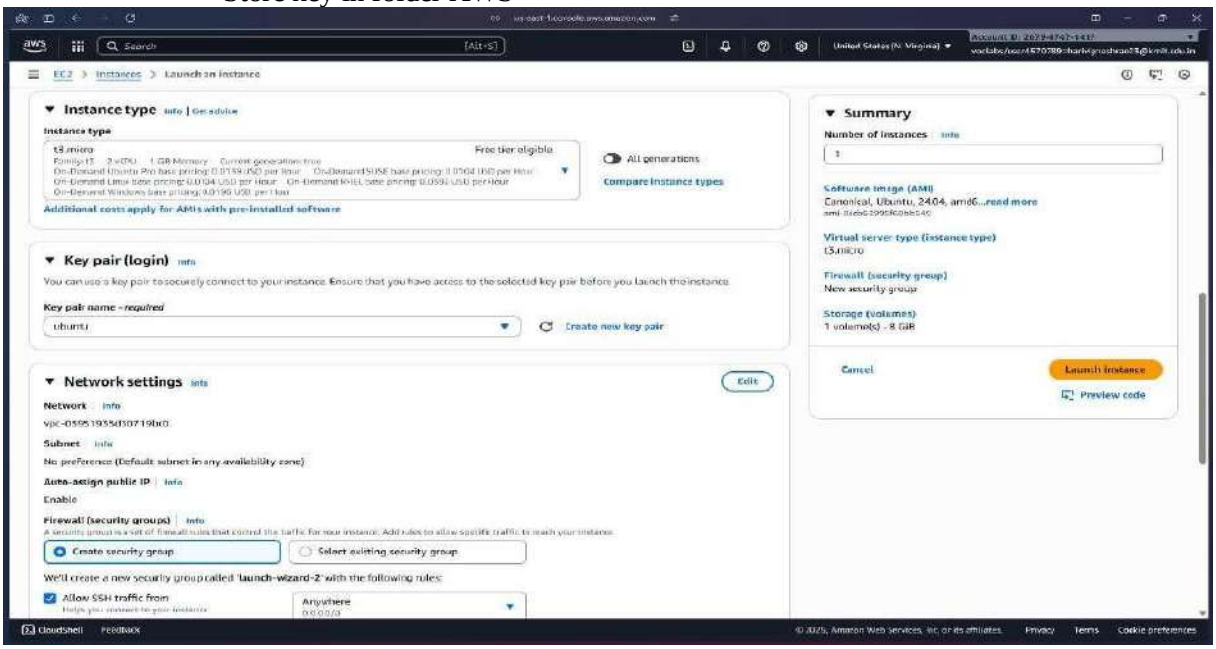
Stage 2 -- Select AMI (Note: Select free tier eligible) ubuntu server



Stage 3 -- Architecture as 64-bit

Stage 4 -- Instance type------ t2.micro(default 1 CPU,1 GB RAM)

Stage 5 -- Create a new keypair---a keypair will downloaded with extension .pem
Store key in folder AWS



Stage 6 -- Network Setting ----Create Security group -- (It deals with ports)
(Note for understanding We have 0 to 65535 ports. Every port is dedicated to special purpose)

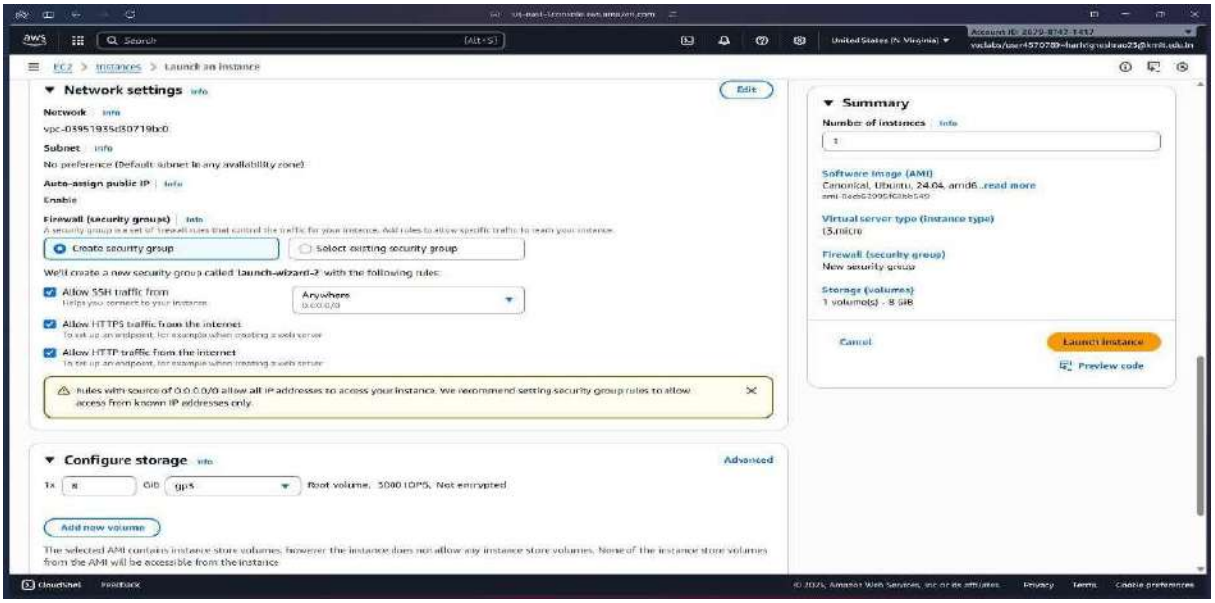
Do this step : HERE select http and https

Stage 7 -- Storage - 8GB (Observation - we have root - it is same as C Drive)

Stage 8---- click on launch instance

Stage 9: Number of instances--- 1

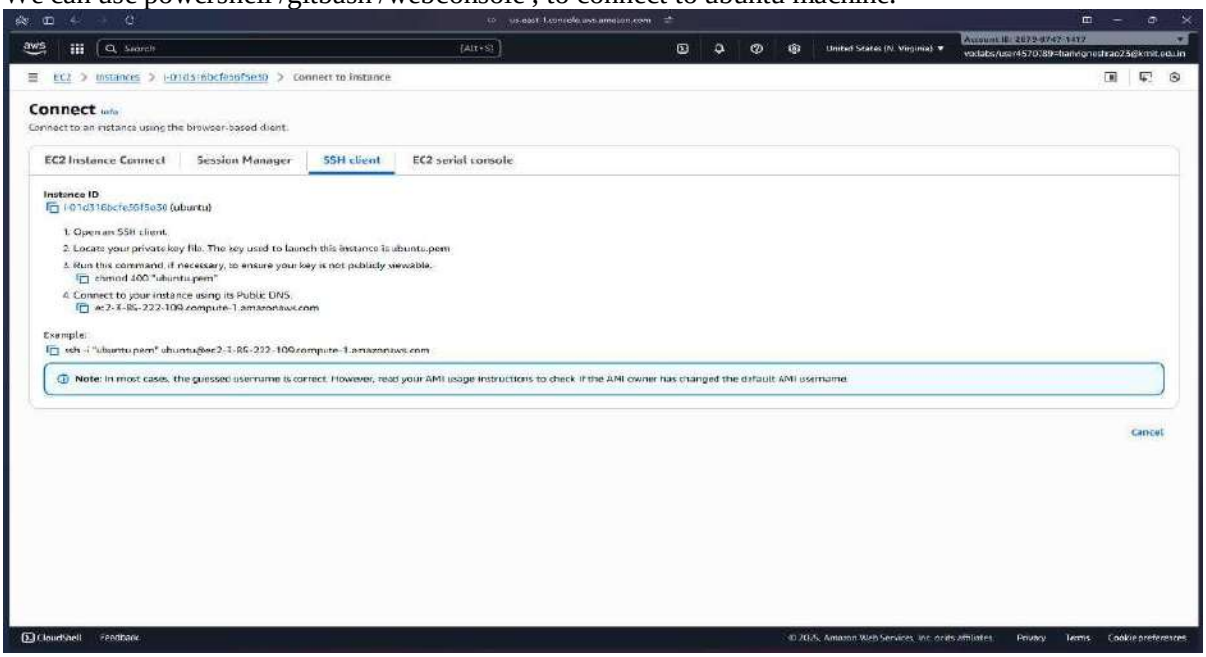
Observation - One machines created



Do this step:---once it is created select that instance and click on connect

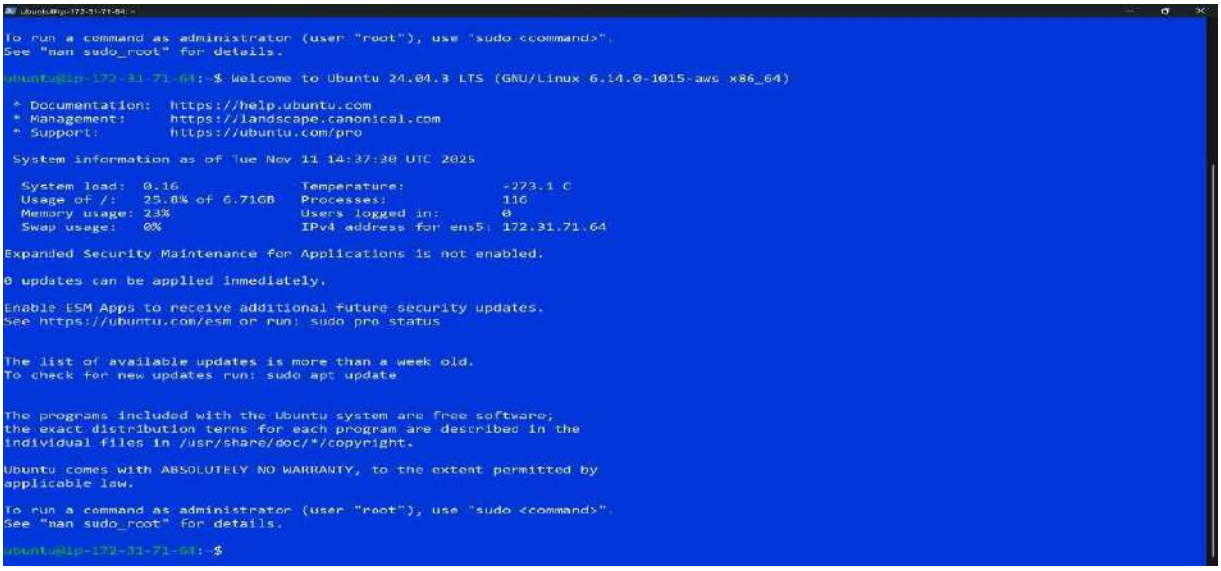
Here copy the ssh – i command from SSH client connect tab

We can use powershell /gitbash /webconsole , to connect to ubuntu machine.



NOTE:- cd path of AWS folder // change path

To connect to above terminals we need to go into the path of the keypair.and paste the
ssh -i command from the aws console

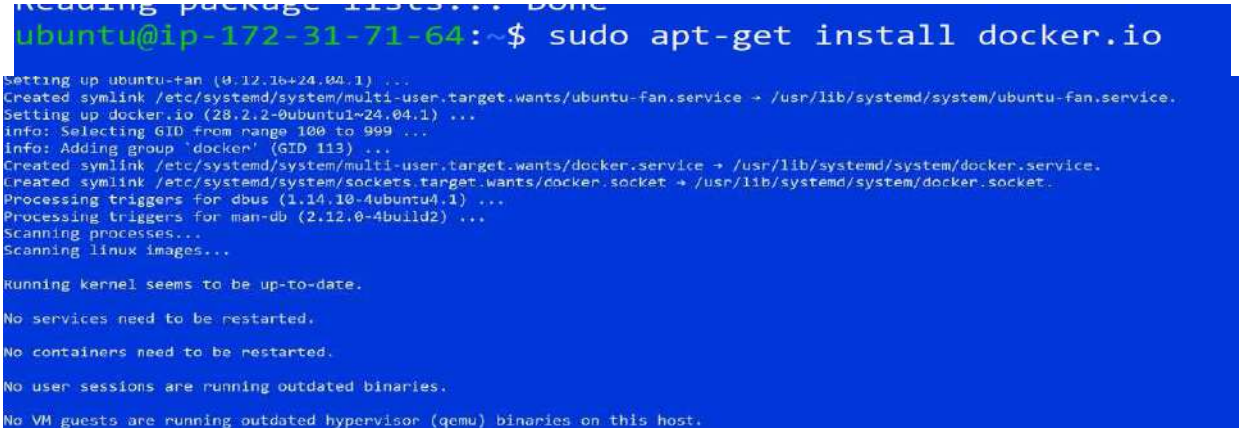


2.Deploying the web application using Nginx and Tomcat servers

.Clone the application from github, Write the Dockerfile
once connected to instance

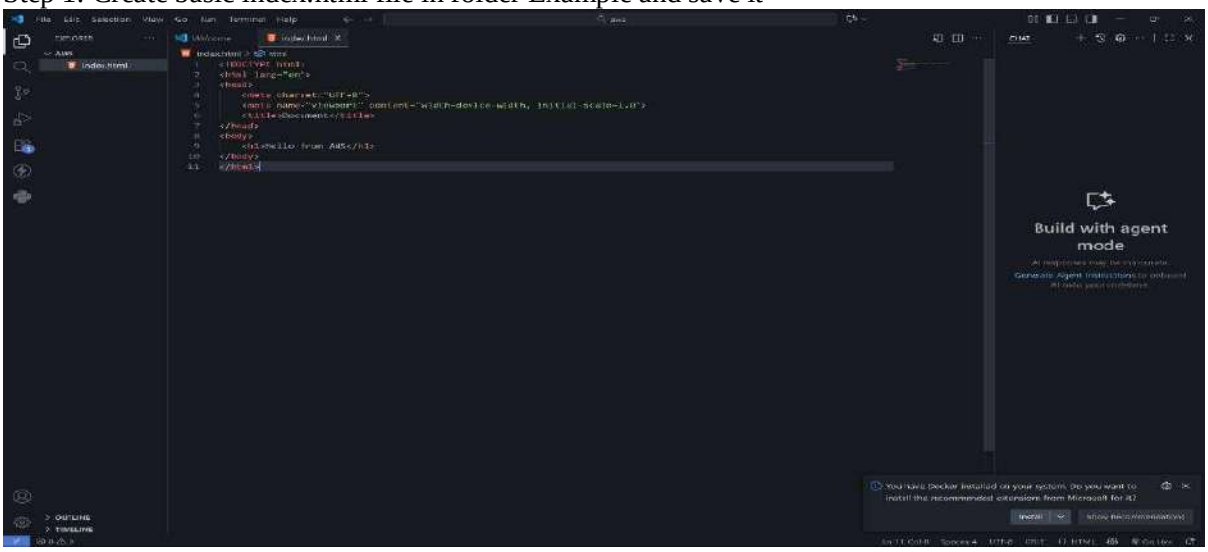
Step 1:- install the docker
install docker ---apt-get update

- apt-get install docker.io



```
ubuntu@ip-172-31-71-64:~$ sudo apt-get install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (7.2-2ubuntu0.1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
ubuntu@ip-172-31-71-64:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
```

Step 1: Create basic index.html file in folder Example and save it



step 2:- git clone <paste the github link of web project>

step 3:- navigate to the aws

```
ubuntu@ip-172-31-71-64:~$ git clone https://github.com/HARIVIGNESHRAO/aws.git
Cloning into 'aws'...
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 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
ubuntu@ip-172-31-71-64:~$ cd aws
ubuntu@ip-172-31-71-64:~/aws$
```

create Dockerfile in above command prompt in ububtu ie in power shell

Nano Dockerfile

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Step : Build docker image by executing the following command

sudo docker build –t mywebapp .

```
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  63.49kB
Step 1/2 : FROM nginx:alpine
alpine: Pulling from library/nginx
2d35ebdb57d9: Pulling fs layer
8f6a6833e95d: Pulling fs layer
194fa24e147d: Pulling fs layer
3eaba6cd10a3: Pulling fs layer
df413d6ebdc8: Pulling fs layer
d9a55dab5954: Pulling fs layer
ff8a36d5502a: Pulling fs layer
bdabb0d44271: Pulling fs layer
d9a55dab5954: Waiting
ff8a36d5502a: Waiting
bdabb0d44271: Waiting
3eaba6cd10a3: Waiting
df413d6ebdc8: Waiting
194fa24e147d: Verifying Checksum
194fa24e147d: Download complete
2d35ebdb57d9: Verifying Checksum
2d35ebdb57d9: Download complete
8f6a6833e95d: Verifying Checksum
8f6a6833e95d: Download complete
3eaba6cd10a3: Verifying Checksum
3eaba6cd10a3: Download complete
d9a55dab5954: Verifying Checksum
d9a55dab5954: Download complete
df413d6ebdc8: Verifying Checksum
df413d6ebdc8: Download complete
ff8a36d5502a: Verifying Checksum
ff8a36d5502a: Download complete
2d35ebdb57d9: Pull complete
bdabb0d44271: Verifying Checksum
bdabb0d44271: Download complete
8f6a6833e95d: Pull complete
```

Thus image is created

Step : run the image and map it to port 80

sudo docker run –d –p 80:80 mywebapp

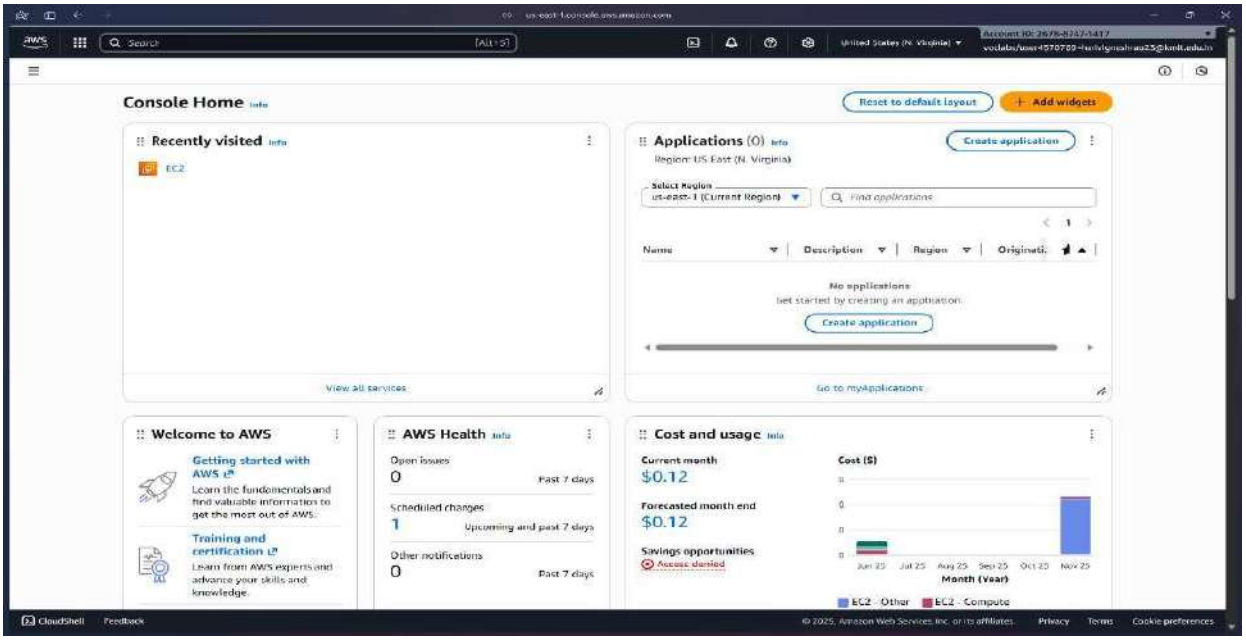
```
ubuntu@ip-172-31-71-64:~/aws$ sudo docker run -d -p 80:80 index
ff2a6ece16299261ae62cdceeee6f643247f8a0d1316736aefb512e9942ea4e4
```

2 MAVEN WEB PROJECT DEPLOYMENT IN AWS

Click on start lab

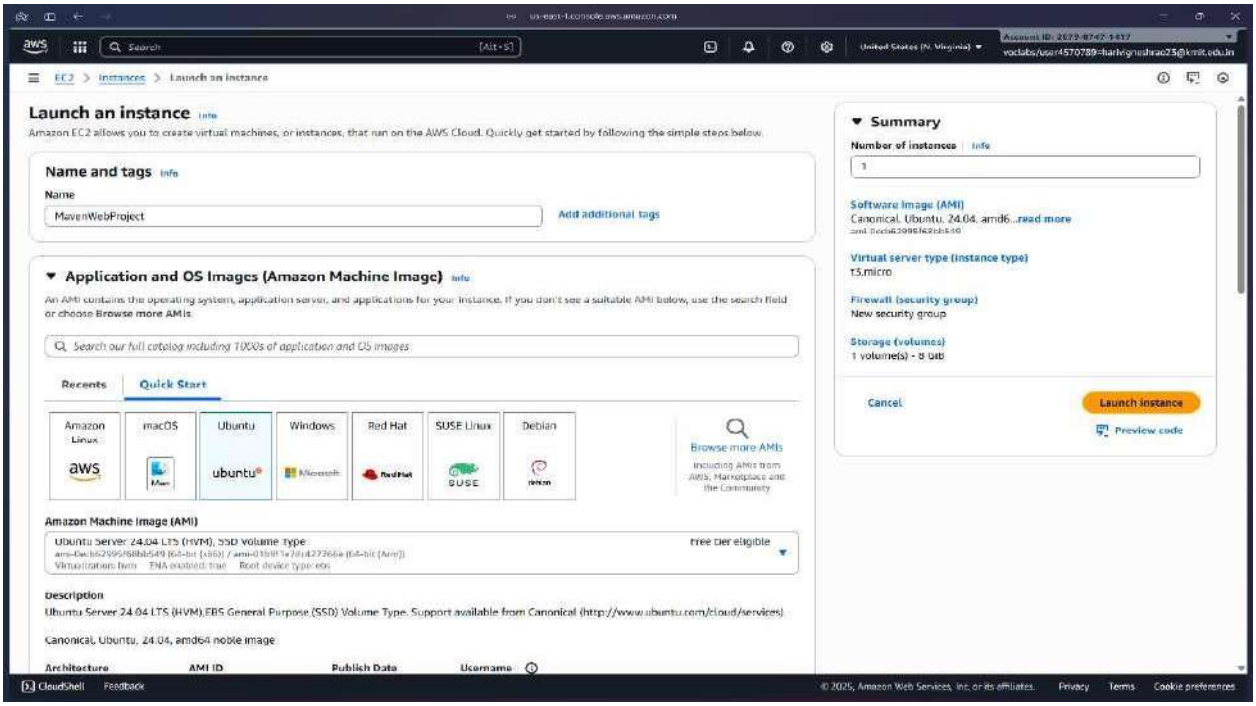
Click on AWS

Click on EC2

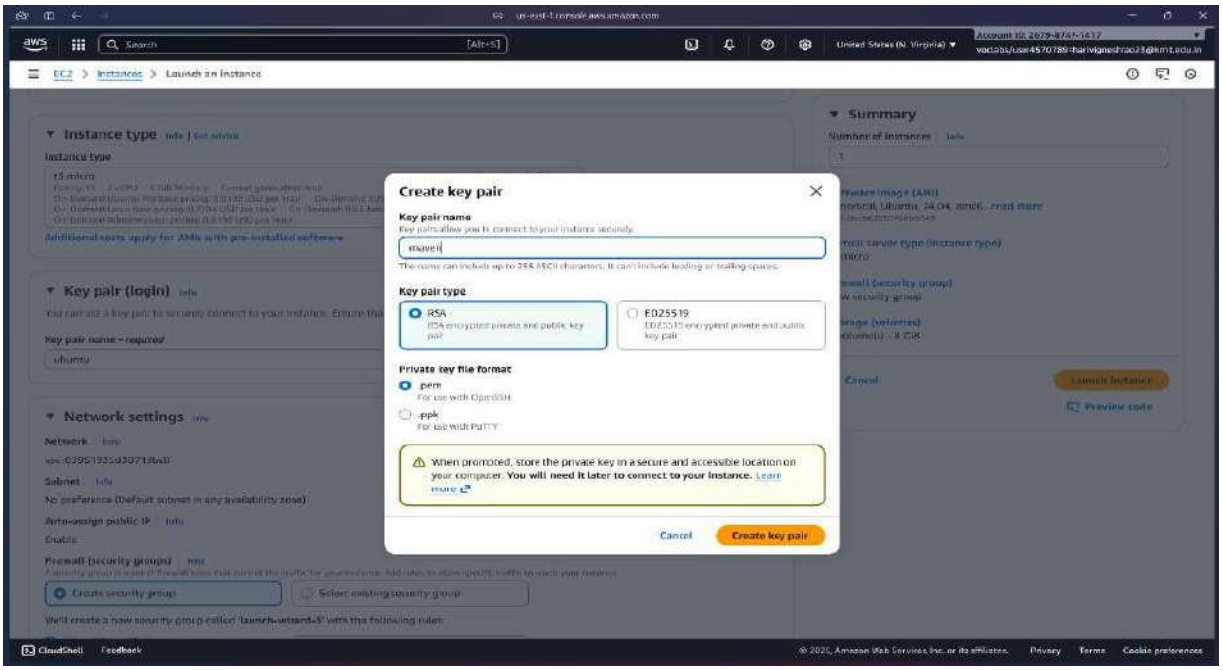


Click on launch instance

Give name



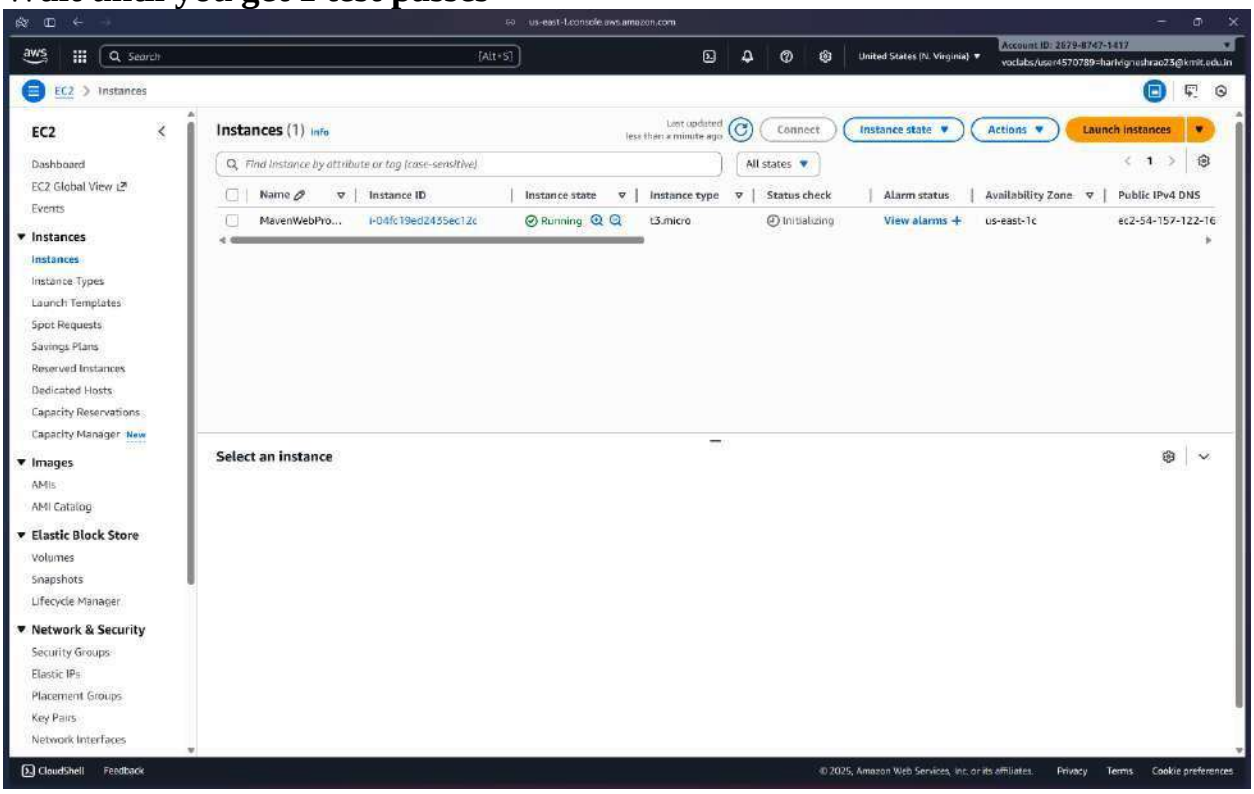
Create new key



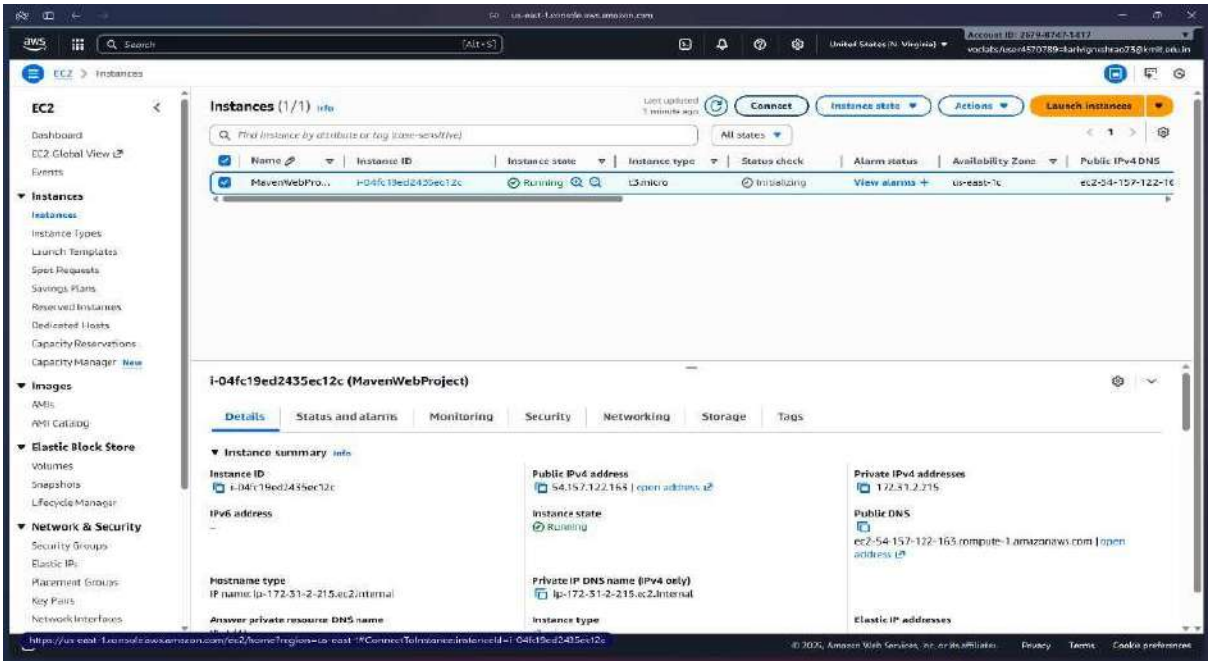
Check all the boxes under network and click on launch instance

Click on instances

Wait until you get 1 test passes



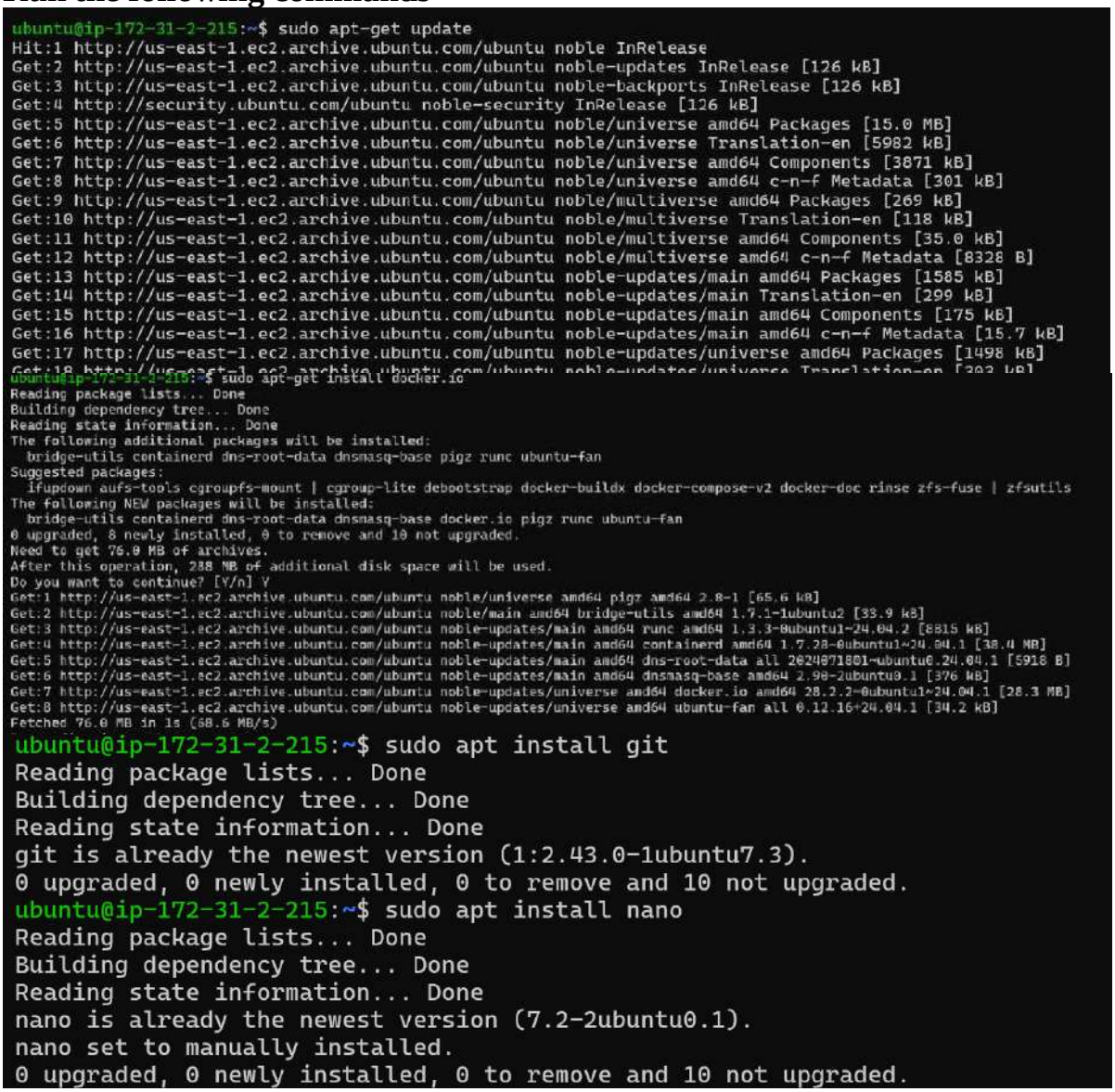
Click on connect after checking the box



Copy ssh command
Open the terminal, Navigate to the path
Run the ssh command

```
The authenticity of host 'ec2-54-157-122-163.compute-1.amazonaws.com (54.157.122.163)' can't be established.
ED25519 key fingerprint is SHA256:VcP9UsM9WrBkWXUewS8RX22A771IaJf3GPMW9YNfnOM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-157-122-163.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Wed Nov 12 01:12:48 UTC 2025

  System load:  0.08              Temperature:           -273.1 C
  Usage of /:   25.9% of 6.71GB   Processes:             110
  Memory usage: 22%               Users logged in:       0
  Swap usage:   0%                IPv4 address for ens5: 172.31.2.215

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-2-215:~$
```

Run the following commands



Open MAVEN WEB PROJECT REPO in github and copy http link
Clone the repository

```
ubuntu@ip-172-31-2-215:~$ git clone https://github.com/HARIVIGNESHRAO/se.git
Cloning into 'se'...
remote: Enumerating objects: 45, done.
remote: Counting objects: 100% (45/45), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 45 (delta 12), reused 45 (delta 12), pack-reused 0 (from 0)
Receiving objects: 100% (45/45), 4.04 MiB | 53.67 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

If your repository is in main run following

```
ubuntu@ip-172-31-2-215:~/se$ nano Dockerfile
```

```
  GNU nano 7.2
FROM tomcat:9-jdk11
COPY target/*.war /usr/local/tomcat/webapps
```

Build docker image

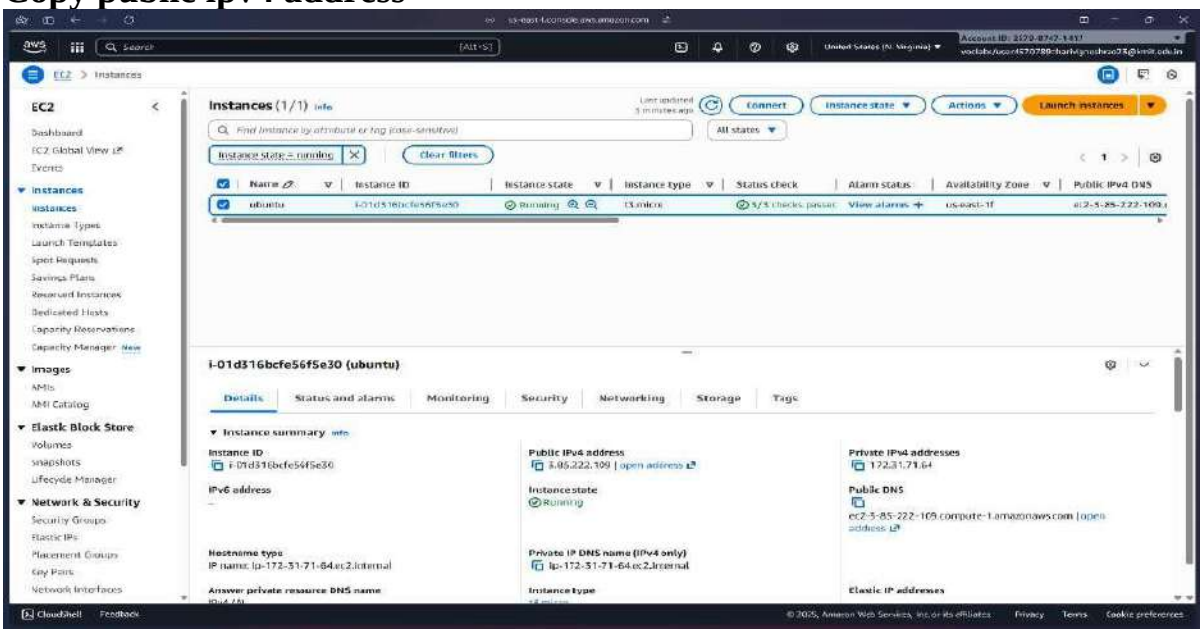
```
ubuntu@ip-172-31-2-215:~/se$ sudo docker build -t maven .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  8.546MB
Step 1/2 : FROM tomcat:9-jdk11
9-jdk11: Pulling from library/tomcat
4b3ffd8ccb52: Pulling fs layer
a4b8822e712a: Pulling fs layer
305f2a30fb03: Pulling fs layer
ac56e5323a09: Pulling fs layer
e46c95531a6b: Pulling fs layer
c3224bb24617: Pulling fs layer
4f4fb700ef54: Pulling fs layer
5c56d23d6b20: Pulling fs layer
c3224bb24617: Waiting
4f4fb700ef54: Waiting
5c56d23d6b20: Waiting
ac56e5323a09: Waiting
e46c95531a6b: Waiting
a4b8822e712a: Verifying Checksum
a4b8822e712a: Download complete
4b3ffd8ccb52: Verifying Checksum
4b3ffd8ccb52: Download complete
ac56e5323a09: Verifying Checksum
ac56e5323a09: Download complete
e46c95531a6b: Verifying Checksum
e46c95531a6b: Download complete
c3224bb24617: Verifying Checksum
c3224bb24617: Download complete
4f4fb700ef54: Verifying Checksum
```

Run the container

```
ubuntu@ip-172-31-2-215:~/se$ sudo docker run -d -p 80:8080 maven
c4936d663ab9e4f4d1bcd4f19b9be6ea4c31bf056109e743a40e10d93afb0795
```

3. Accessing it publicly
Go to instances and click on instance here

Copy public ipv4 address



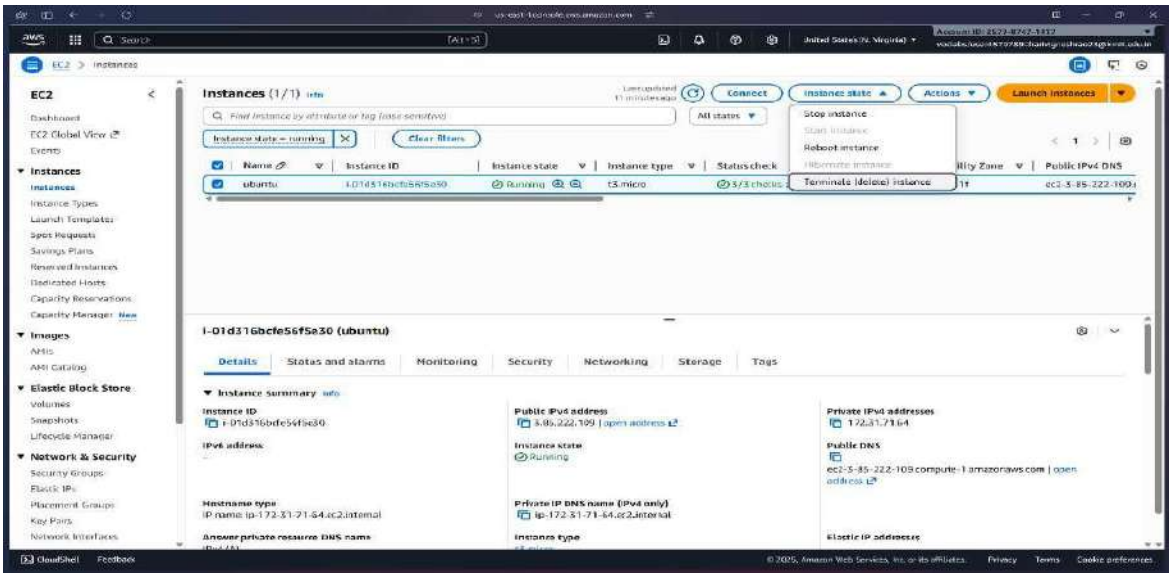
Paste it in the browser to get below output
Note : if https :__won't work then type just http:____

Step: stop container

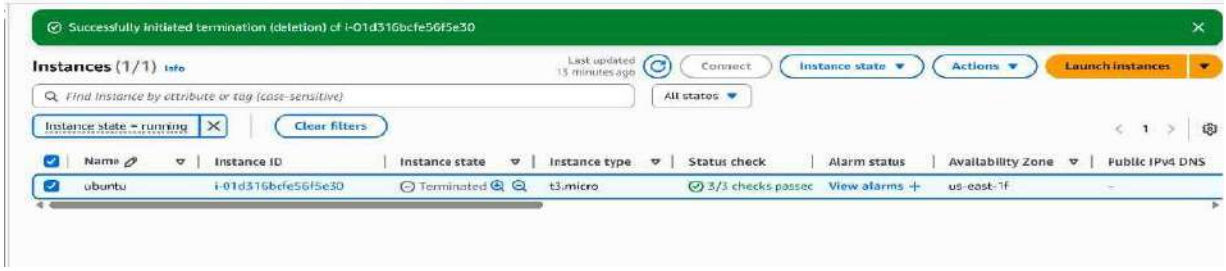
```
ubuntu@ip-172-31-71-64:~/aws$ sudo docker stop ff2a6ece1629
ff2a6ece1629
```

Run the following command to stop the container

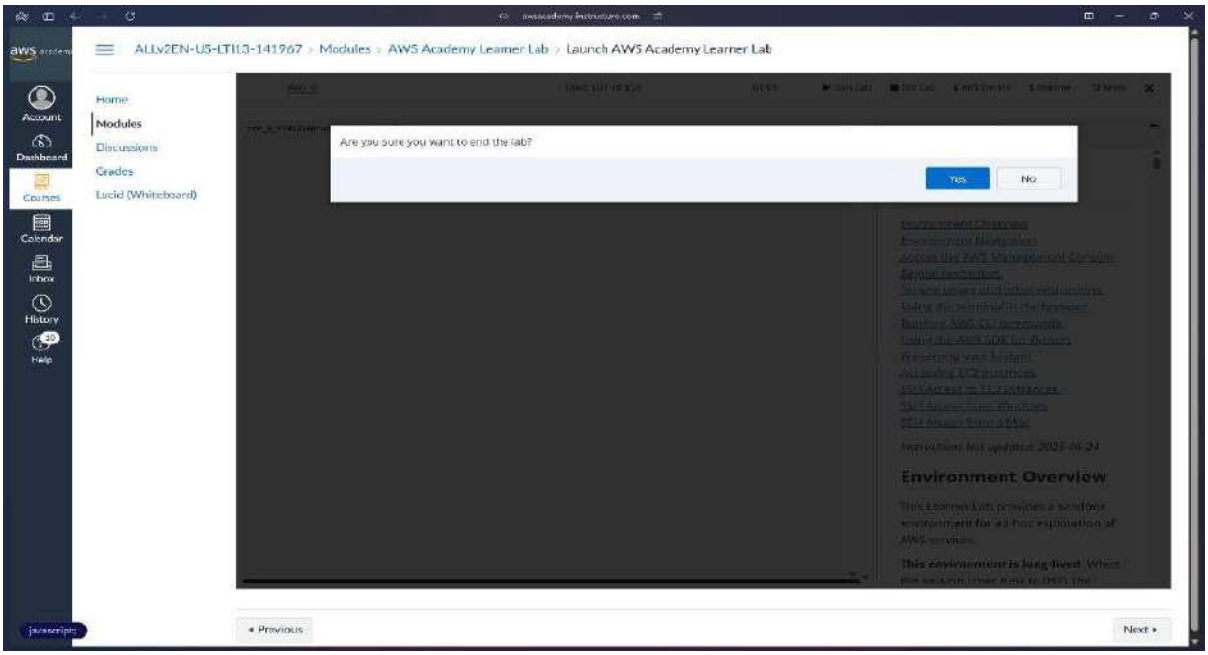
i. **sudo docker ps**
ii. **sudo docker stop <container-id>**
iii. **Go back to instances and click on option instance. Once load as below, then Click on Instance state and select terminate instances**



Click on Terminated (deleted)

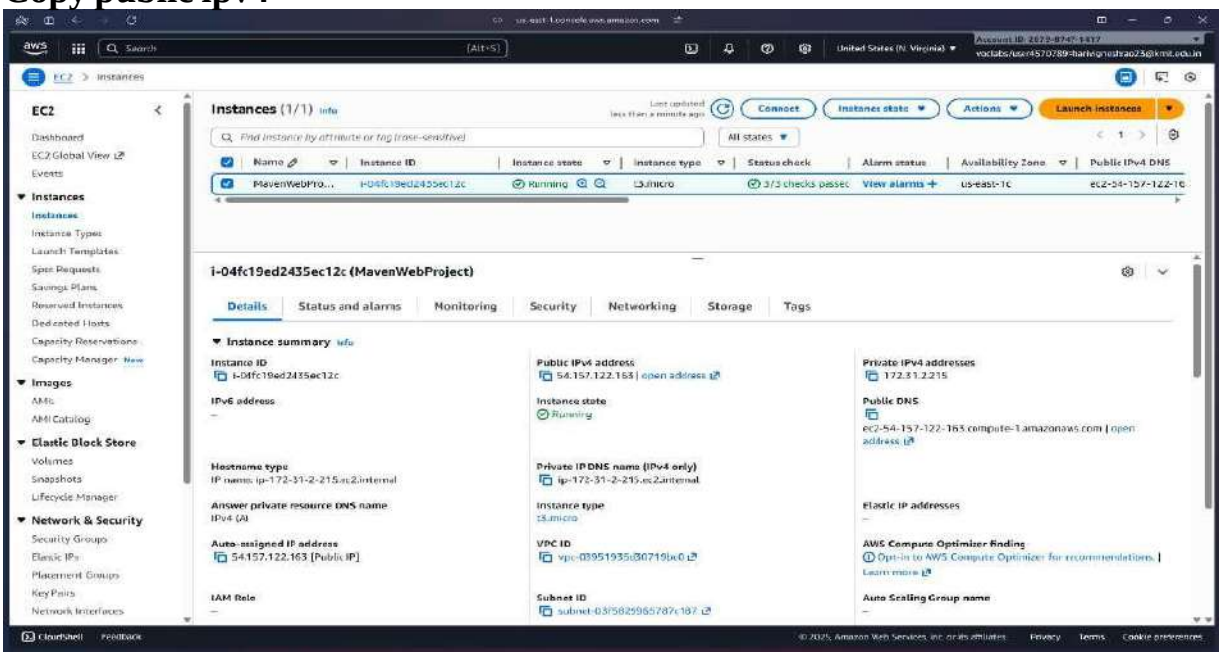


iv. Now click on End lab



Maven Project

Copy public ipv4

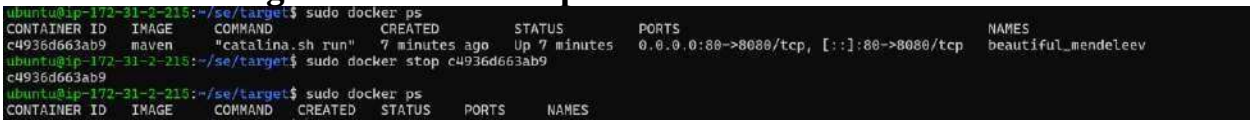


Enter this url in browser and click enter

And refresh the page to check whether the web page has loaded or not

If it is loaded you have successfully deployed you maven web project In your ec2 instance

Run the following commands to stop the container



Terminate your instance

End your lab