

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.
- Extract text content from uploaded documents.
- Segment documents into meaningful chunks (e.g., paragraphs or sections) for processing.
- Convert text chunks into semantic embeddings using a pre-trained language model.
- Store embeddings in a vector database (e.g., Pinecone, Weaviate, or FAISS).

2. Question Handling:

- Accept user questions via a text input interface.
- Process questions using natural language understanding (via Google Gemini or similar AI model).
- Retrieve relevant document chunks from the vector database based on semantic similarity.
- Generate accurate, concise, and contextually relevant answers using RAG.

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.
- Use memory to provide personalized and contextually coherent responses.

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.
- Experiment with different answer styles (e.g., concise, detailed, or conversational) based on user preferences.

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.
- Support A/B testing for different answer generation approaches.

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 document.

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 seconds for files up to 10 MB.
- Retrieve relevant document chunks and generate answers within 2 seconds per query.
- Handle up to 100 concurrent users without significant performance degradation.

3. Security:

- Encrypt uploaded documents and user data during transmission and storage.
- Implement user authentication and role-based access control to protect sensitive documents.
- Ensure compliance with data privacy regulations (e.g., GDPR, CCPA).

4. Reliability:

- Achieve 99.9% system uptime, excluding scheduled maintenance.
- Handle errors gracefully, providing meaningful messages to users (e.g., unsupported file types).

5. Scalability:

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

6. Maintainability:

- Use modular code design for easy updates and debugging.
- Include comprehensive documentation for developers and users.

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. Professionals:

- Leverage the system to analyze business documents, contracts, or reports for decision-making.

5. 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

- Onboarding & Document Upload:** Student logs in (or uses guest mode), 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 (e.g., Pinecone). Processing completes in <5 seconds for files ≤10 MB.
- Question Session Initiation:** Student navigates to the intuitive Streamlit/React interface, starts a new conversation session. Short-term memory activates to track context within the session.
- 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, retrieves relevant chunks via semantic similarity from vector DB, applies RAG to generate concise answer. Answer displays in <2 seconds, with source chunks shown for transparency. Long-term memory recalls past sessions (e.g., prior questions on the same notes) for personalized context.
- Follow-up & Interaction:** Asks follow-ups (e.g., "Explain it simply"). Short-term memory maintains conversation flow; system adapts style based on stored preferences (e.g., concise for exams).
- Feedback & Improvement:** Rates answer (thumbs up/down) or adds comments. Feedback logs for system learning; student views conversation history.
- Session End & Management:** Views/manages uploaded documents; logs out. System evaluates performance internally using datasets like SQuAD 2.0.

Teachers

- 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. **Follow-up & Interaction:** Refines with follow-ups (e.g., "Make it student-friendly"). A/B testing experiments with styles; short-term memory ensures coherence.
5. **Feedback & Improvement:** Provides feedback to fine-tune (e.g., prefers detailed styles); reviews history to manage documents.
6. **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. **Follow-up & Interaction:** Drills down (e.g., "Cite specific sections"). Memory personalizes (e.g., recalls preferred detailed style).
5. **Feedback & Improvement:** Submits feedback/comments for error correction; logs support continuous improvement.
6. **Session End & Management:** Manages document library; views history for ongoing research.

Professionals

1. **Onboarding & Document Upload:** Professional authenticates (enterprise role-based), uploads business docs (PDF contracts, Word reports, .txt memos, web URLs). Encrypted processing and storage ensure compliance (GDPR/CCPA).
2. **Question Session Initiation:** Initiates session for decision-making; long-term memory tracks preferences (e.g., concise for quick insights).
3. **Query Submission:** Queries (e.g., "What are the risk clauses in the uploaded contract?"). RAG delivers relevant, concise answer with chunks.
4. **Follow-up & Interaction:** Follows up (e.g., "Highlight implications"). Short-term memory maintains context.
5. **Feedback & Improvement:** Rates for style experimentation; feedback refines system.
6. **Session End & Management:** Securely manages/deletes documents; supports 100+ concurrent users.

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

Below are practical use cases tailored to each user type, demonstrating real-world applications of the system's features (RAG, memory, feedback, multi-format support, UI transparency, and evaluation).

Students

- **Exam Preparation:** Uploads textbook PDFs and notes; asks "Solve this sample question from Chapter 5" → Gets step-by-step answer with sources; follow-ups use short-term memory for clarification; rates for concise style preference.
- **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 refine for educational tone.

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 (role-based); follow-ups in sessions → Scalable for large volumes.

Professionals

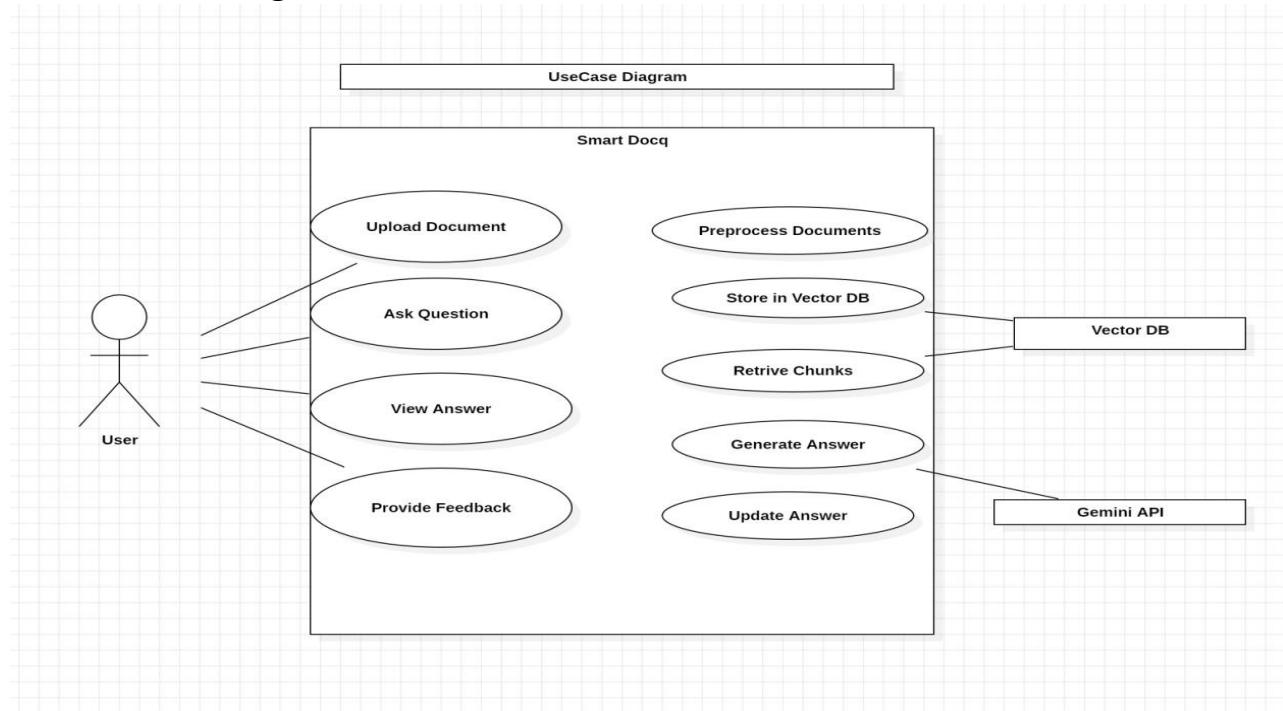
- **Contract Analysis:** Uploads PDF/Word contracts; asks "Identify termination clauses" → Concise RAG answer with highlights; short-term memory for negotiation follow-ups.
- **Report Insights:** .txt memos + web reports; decision queries like "Compare Q3 sales data" → Personalized based on preferences; secure encryption for sensitive data.
- **Meeting Prep:** Quick questions on uploaded docs → <2 sec responses; feedback for business-concise style.

Administrators

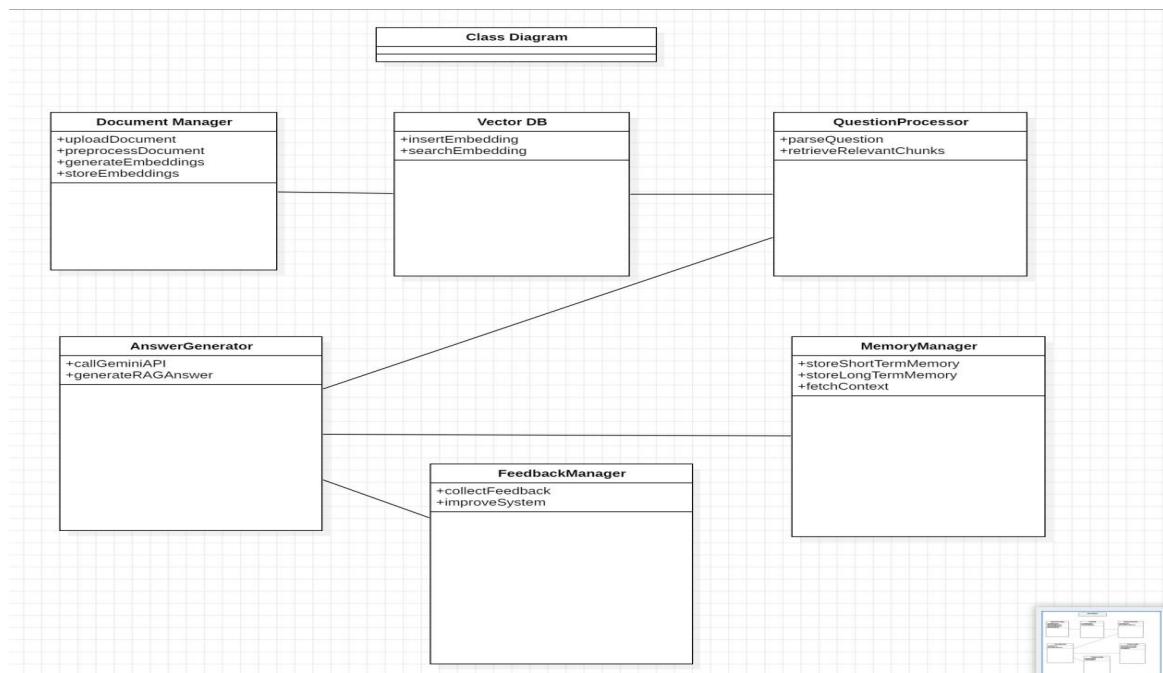
- **System Monitoring:** Runs test queries on sample docs; views metrics (accuracy from SQuAD 2.0, uptime); A/B tests new AI integrations.
- **User Management:** Revokes access for sensitive docs; reviews feedback logs for global improvements; ensures scalability during peak usage.
- **Compliance Audit:** Queries system logs; verifies encryption/privacy; manages document deletion for GDPR compliance.

f. UML Diagrams

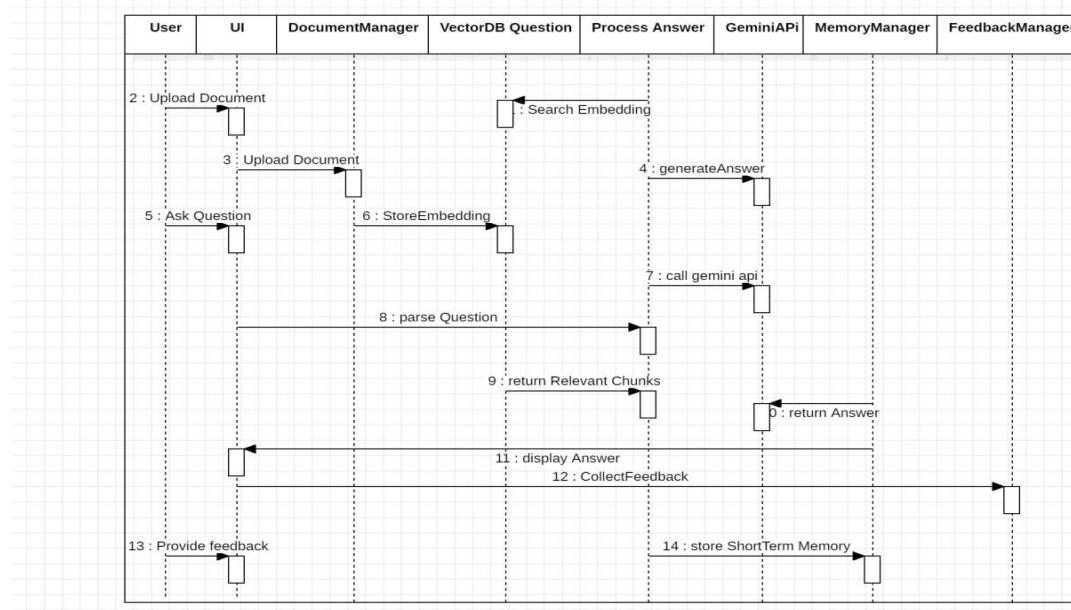
Use Case Diagram



Class Diagram

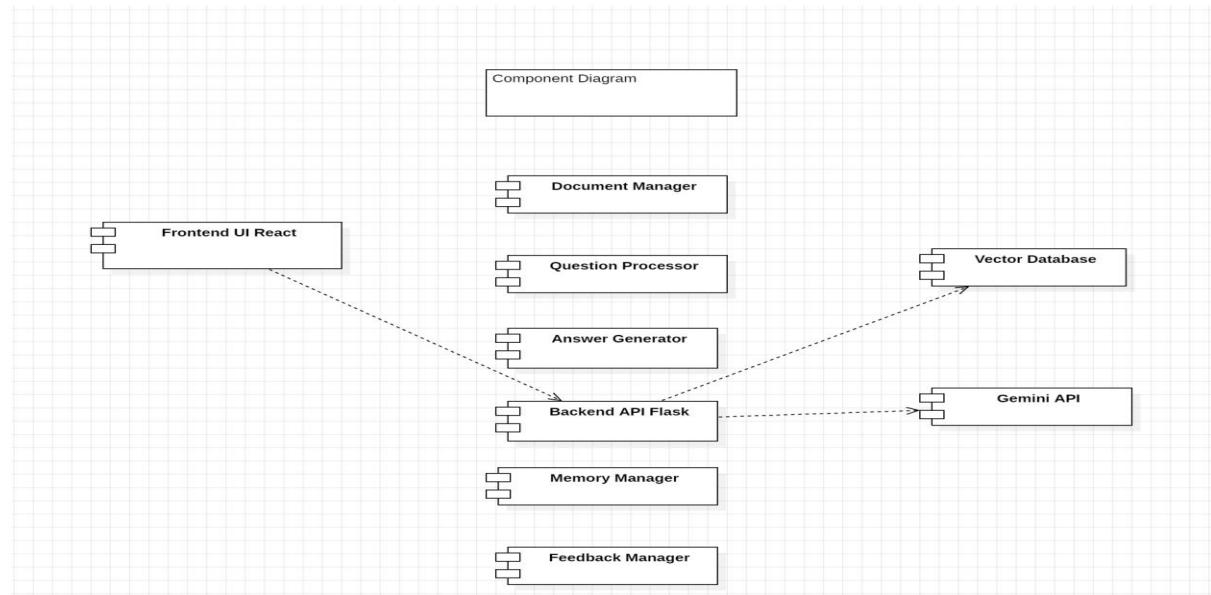


SequenceDiagram



SEQUENCE DIAGRAM

Component Diagram



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

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

1. Create a Private Repository on GitHub

Create a new repository [Try the new experience](#)

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * HARIVIGNESHRAO / Repository name * Movie Movie is available.

Great repository names are short and memorable. Need inspiration? How about [glowing-meme](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.
 Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore
Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license
License: None

① You are creating a private repository in your personal account.

[Create repository](#)

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@DESgit remote -vMINGW64 /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.
```

- 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?
Ans)

```
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?

Ans)

```
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 'origin/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 'origin/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?

Ans)

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

Ans) first commit

```

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

Ans)

```

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?

Ans

```

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 ?

Ans)

```

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 (origin/main) HEAD@{7}: commit (initial): first commit

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

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've made some commits locally and now want to upload them to the remote repository. What do you run? Ans)

```

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ git remote -v
origin1 https://github.com/Harshitha-Macha/harshitha2907.git (fetch)
origin1 https://github.com/Harshitha-Macha/harshitha2907.git (push)

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((c1ab6b2...))
$ git push -u origin1 feature/login
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 788 bytes | 394.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/login' on GitHub by visiting:
remote:     https://github.com/Harshitha-Macha/harshitha2907/pull/new/feature/login
remote:
To https://github.com/Harshitha-Macha/harshitha2907.git
 * [new branch]      feature/login -> feature/login
branch 'feature/login' set up to track 'origin1/feature/login'.

```

10. How can you fetch the latest changes from the remote repository without merging them automatically?

Ans)

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 934 bytes | 3.00 KiB/s, done.
From https://github.com/Harshitha-Macha/harshitha2907
  c1ab6b2..208deeb main      -> origin/main
```

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

Ans)

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

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

Ans)

```
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 aff33ed2da6ced7eabc86e7f278b3bc0fb1d40a (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/origin/main/feature/login' is unchanged
Ref 'refs/remotes/origin/main' was rewritten
```

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

Ans)

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

14. You're on main and want to merge the completed feature/signup branch into it. What command do you use?

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature/signup)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git merge feature/signup
Already up to date.
```

15. 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 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

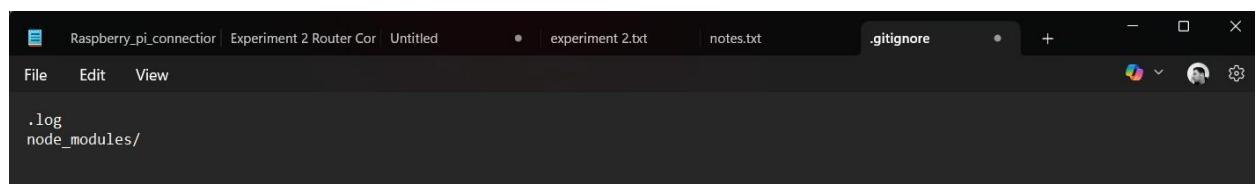
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

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

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

Ans) Add to .gitignore

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



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

Ans)

```

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

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

```

18. 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 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

```

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

Ans)

```

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

```

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

Ans)

```

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

```

21. You just merged the feature-ui branch into main. You want to clean up your local branches. How do you safely delete feature-ui?

Ans)

```

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha ((e71a73a...))
$ git branch -d feature-ui
Deleted branch feature-ui (was c1ab6b2).

```

22. 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?

Ans)

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

23. You're currently on the `feature-login` branch and want to delete `feature-ui`. What must you ensure before running the delete command?

Ans)

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature-ui)
$ git checkout feature/login
Switched to branch 'feature/login'
Your branch is up to date with 'origin1/feature/login'.

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (feature/login)
$ git branch -d feature-ui
Deleted branch feature-ui (was 4ebd3db).
```

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

Ans)

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

25. You finished working on `feature-a`, `feature-b`, and `feature-c`. All have been merged into `main`. How do you delete them in one command?

Ans)

```
branch
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch --merged main | grep feature-a |xargs git branch -d
Deleted branch feature-a (was 9d4848c).

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch --merged main | grep feature-b feature-c |xargs git branch -d
grep: feature-c: No such file or directory
fatal: branch name required

91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git branch --merged main | grep feature- |xargs git branch -d
Deleted branch feature-b (was 9d4848c).
Deleted branch feature-c (was 9d4848c).
```

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

3. Specify the git command if you want to add a new remote repository to your local repository.
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 fetch updates from the remote repository but not merge them into your local branch.

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

7. Specify the git command to pull the latest changes from the remote repository and merge them into your local branch.
8. Specify the git command to push your local commits to a remote repository.

```
919 91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git push origin main
$ gEverything up-to-date
branch 'main' set up to track 'origin/main'.
Everything up-to-date
```

9. Specify the git command to when pushing for the first time and want to set the remote branch you are pushing to.
10. 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
```

11. 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
```

12. 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
```

13. 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
```

14. Specify the git command if you want to see detailed information about a remote repository.

Ans)

15. 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)

Ans)

```
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.
```

```
91949@Harshitha-Macha MINGW64 ~/Desktop/SE/Harshitha (main)
$ git remote show origin
* remote origin
  Fetch URL: https://github.com/Harshitha-Macha/harshitha2907.git
  Push URL: https://github.com/Harshitha-Macha/harshitha2907.git
  HEAD branch: main
  Remote branches:
    feature-ui      tracked
    feature/login   tracked
    main            tracked
    search-filter   tracked
  Local branches configured for 'git pull':
    feature/login merges with remote feature/login
    main          merges with remote main
  Local refs configured for 'git push':
    feature/login pushes to feature/login (up to date)
    main          pushes to main        (up to date)
    search-filter pushes to search-filter (up to date)
```

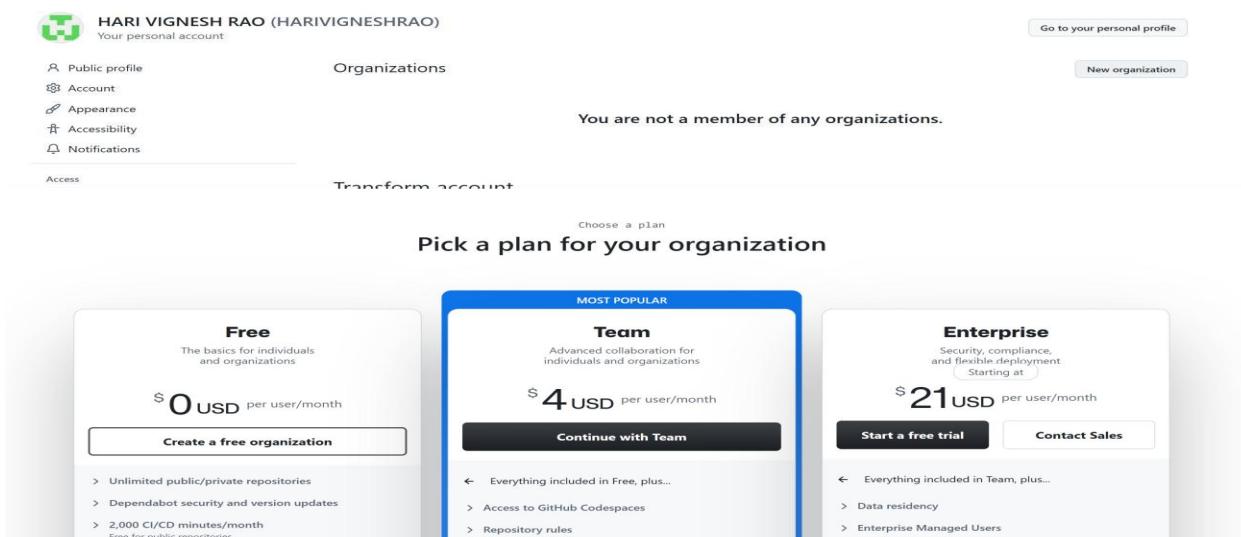
Experiment-3 : Collaborative coding using git

- 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. Owners and administrators can manage member access to the organization's data and projects with sophisticated security and administrative features. Steps to facilitate collaborative work by creating organization are:

Step 1: Click on New organization in Git hub

Click 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

Tell us about your organization

Set up your organization

Organization name *

salaar khansaar

This will be the name of your account on GitHub.
Your URL will be: <https://github.com/salaar-khansaar>, which has been adjusted to with our naming rules.

Contact email *

harivigneshrao23@kmit.edu.in

This organization belongs to:

My personal account
i.e., HARIVIGNESHRAO (HARI RAO)

A business or institution
For example: GitHub, Example American Red Cross

Verify your account

Add-ons

0 @} Get GitHub Copilot Business in this organization
Boost developer productivity for \$19/user/month. Pay only for assigned seats after setup.
[See Copilot docs.](#)

0 I hereby accept the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#).

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

The screenshot shows a GitHub organization profile for 'salaar khansaar'. The top navigation bar includes 'Follow' and a dropdown for 'View as: Public'. Below the header, a message says 'We think you're gonna like it here.' with a note about suggested tasks. The main content area is divided into several sections: 'Invite your people' (with 'Invite your first member' and 'Customize members' permissions'), 'Collaborative coding' (with 'Create a pull request' and 'Create a branch protection rule'), 'Automation and CI/CD' (with 'Auto-assign new issues' and 'Run a continuous integration test'), and 'Discover new GitHub features' (with links to Security, Client apps, Project management, Team administration, and Community). A large arrow points from the text 'This repository is accessible to every member of the team as per the permissions given' towards the 'Customize members' permissions' section.

Create a new repository
 Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
 Required fields are marked with an asterisk (*).

1 General

Owner *  **salaar-khansaar** / Repository name *  commonREPO is available.

Great repository names are short and memorable. How about **effective-octo-goggles**?

Description 0 / 350 characters

2 Configuration

Choose visibility *  Private

Add README READMEs can be used as longer descriptions. [About READMEs](#) Off

Add .gitignore .gitignore tells git which files not to track. [About ignoring files](#) No .gitignore

Add license Licenses explain how others can use your code. [About licenses](#) No license

3 Jumpstart your project with Copilot (optional) 🤖
 Tell Copilot what you want to build in this repository. After creation, Copilot will open a pull request with generated files - such as a basic app, starter code, or other features you describe - then request your review when it's ready. [About Copilot coding agent](#)

Prompt
 Describe what you want Copilot to build
 0 / 500 characters

Add README READMEs can be used as longer descriptions. [About READMEs](#) Off

Add .gitignore .gitignore tells git which files not to track. [About ignoring files](#) No .gitignore

Add license Licenses explain how others can use your code. [About licenses](#) No license

3 Jumpstart your project with Copilot (optional) 🤖
 Tell Copilot what you want to build in this repository. After creation, Copilot will open a pull request with generated files - such as a basic app, starter code, or other features you describe - then request your review when it's ready. [About Copilot coding agent](#)

Prompt
 Describe what you want Copilot to build
 0 / 500 characters

Create repository

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 1 means other public repository already exists that is pushed by person 1. Person 2 is who wants 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
```

d. To resolve conflicts when collaborating on same part of code.

Resolving conflicts when collaborating with GitHub (or Git in general) is a normal part of team development. Conflicts happen when two people make changes to the same part of the code, and Git doesn't know which version to keep.

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. Edit the File to Fix the Conflict

- ✓ Choose one version or combine them:

```
def greet():
    print("Hello from First collaborator ")
    print("Hello from second collaborator ")
```

- ✓ Then **delete** all the **conflict markers** (<<<<<, =====, >>>>>).

4. Mark the Conflict as Resolved

Once you've edited and saved the file:

- ✓ git add f1.txt

5. 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)
$ |
```

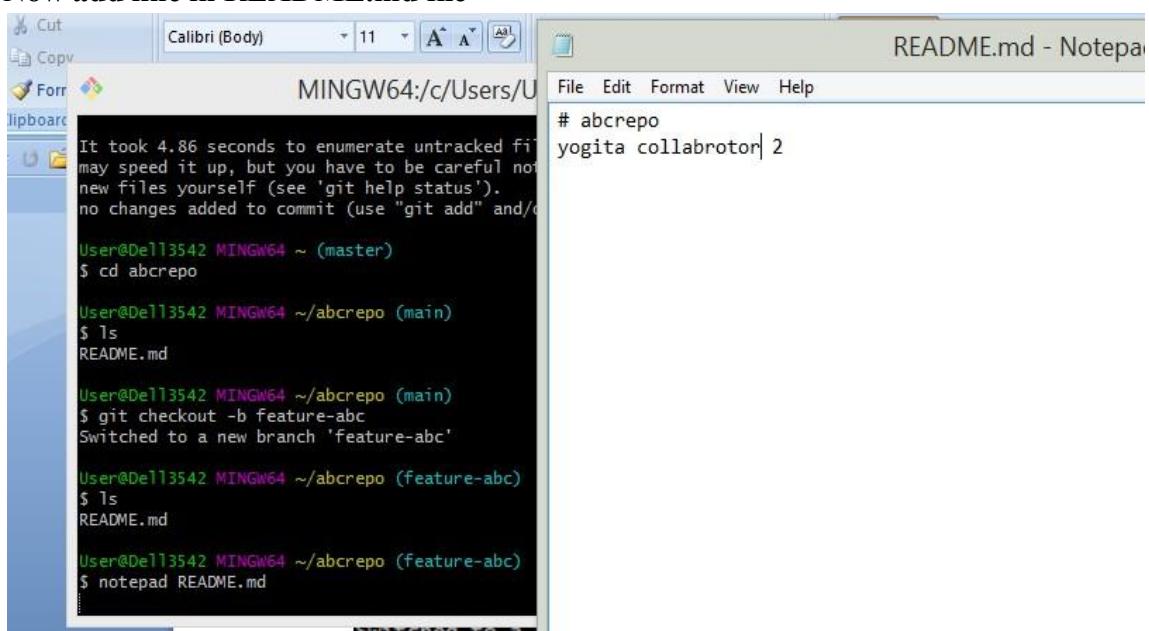
- ✓ Switch to branch feature-abc

```
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)
$ |
```

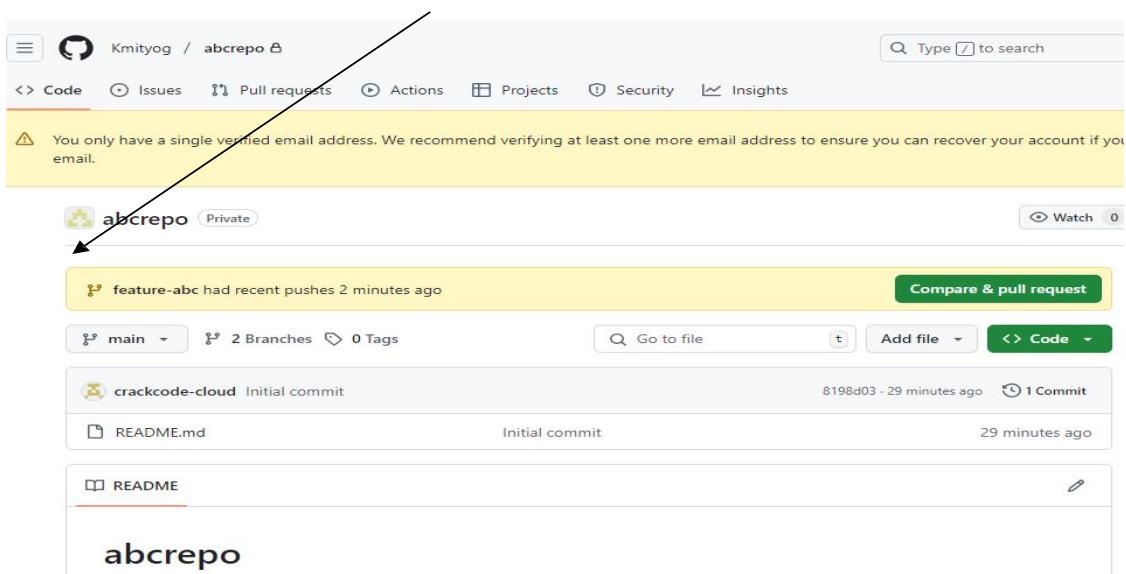
✓ Now see status and push to remote branch feature-abc

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

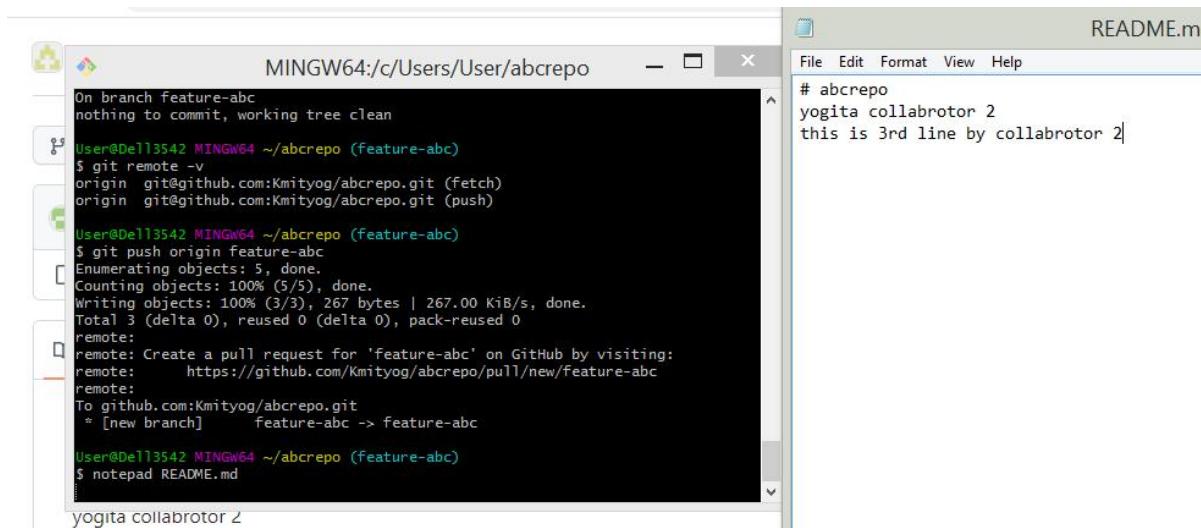
User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git remote -v
origin  git@github.com:Kmityog/abcrepo.git (fetch)
origin  git@github.com:Kmityog/abcrepo.git (push)

User@Dell3542 MINGW64 ~/abcrepo (feature-abc)
$ git push origin feature-abc
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (5/5), 267 bytes | 267.00 KiB/s, done.
Total 5 (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)
$ |
```



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



The terminal window shows the command history for a user named 'yogita collaborator 2'. It includes a 'git remote -v' command, a 'git push' command that successfully pushes changes to the 'feature-abc' branch, and a note to create a pull request. The code editor window shows the 'README.md' file with a new line: '# abcrepo' followed by 'yogita collaborator 2' and 'this is 3rd line by collaborator 2'.

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



The terminal window shows a user named 'Dell' attempting to push changes to the 'feature-abc' branch. The push fails with a 'fatal' error message stating that the current branch has no upstream branch. It provides instructions to use 'git push --set-upstream origin feature-abc'. The user then runs 'git push --set-upstream origin feature-abc' and receives a rejection message from GitHub, indicating that the push failed because the remote contains work that the user does not have locally. It suggests using 'git pull' before pushing again. Finally, the user runs 'git pull'.

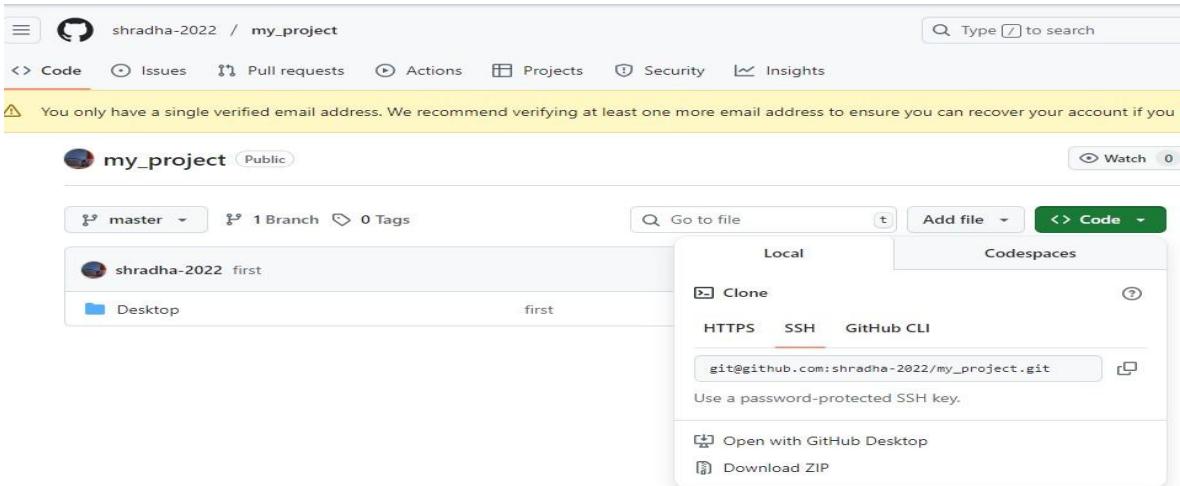
Now open file, Edit the File to Fix the Conflict by keeping either or both collaborator changes and remove conflict markers. Add file, commit, and push.

e. 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

```
User@Dell3542 MINGW64 ~ (master)
$ git clone git@github.com:shradha-2022/my_project.git
Cloning into 'my_project'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 10 (delta 0), reused 10 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (10/10), done.

User@Dell3542 MINGW64 ~ (master)
$ cd my_project

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

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

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

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

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 <yogita@gmail.com>
Date:   Thu Jul 31 02:15:11 2025 +0530

    commit from cloned person

commit 686b0a755049b4321e993912a182cee4d2226aa? (origin/master, origin/HEAD)
Author: shradha-2022 <shradhacg@gmail.com>
Date:   Wed Jul 30 09:44:59 2025 +0530

    first

commit d095cc0f03fc53097b7a80bef7a396a046526e
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.

Note: git format-patch -1 <commit-hash>

- -1 means "create a patch from 1 commit"
- This creates a .patch file for a single commit.
- 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.

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

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"

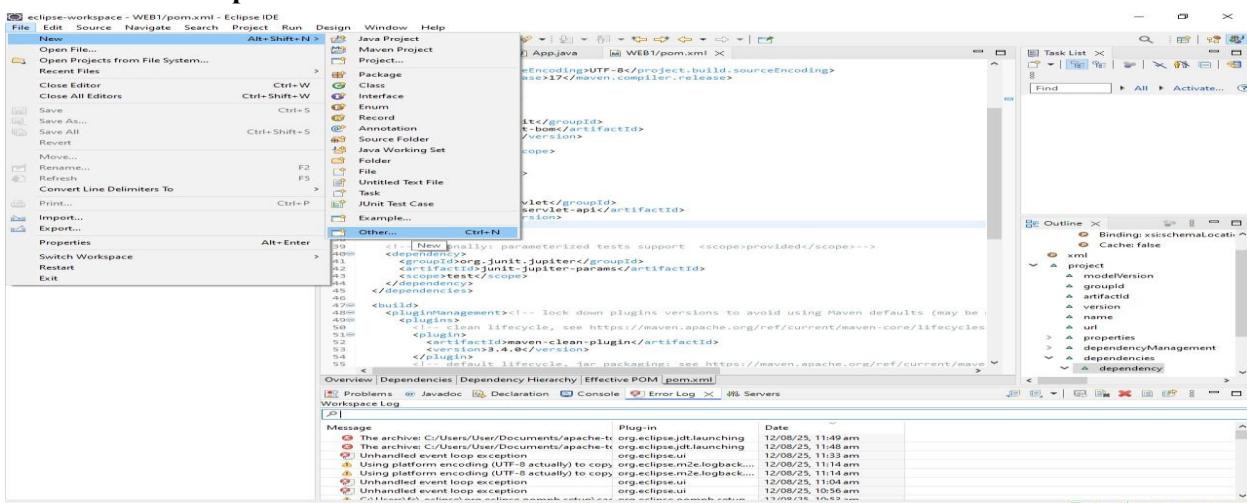
└─ 2.1.2. Search for "Maven Integration for Eclipse"

└─ 2.1.3. Install the plugin if not already installed

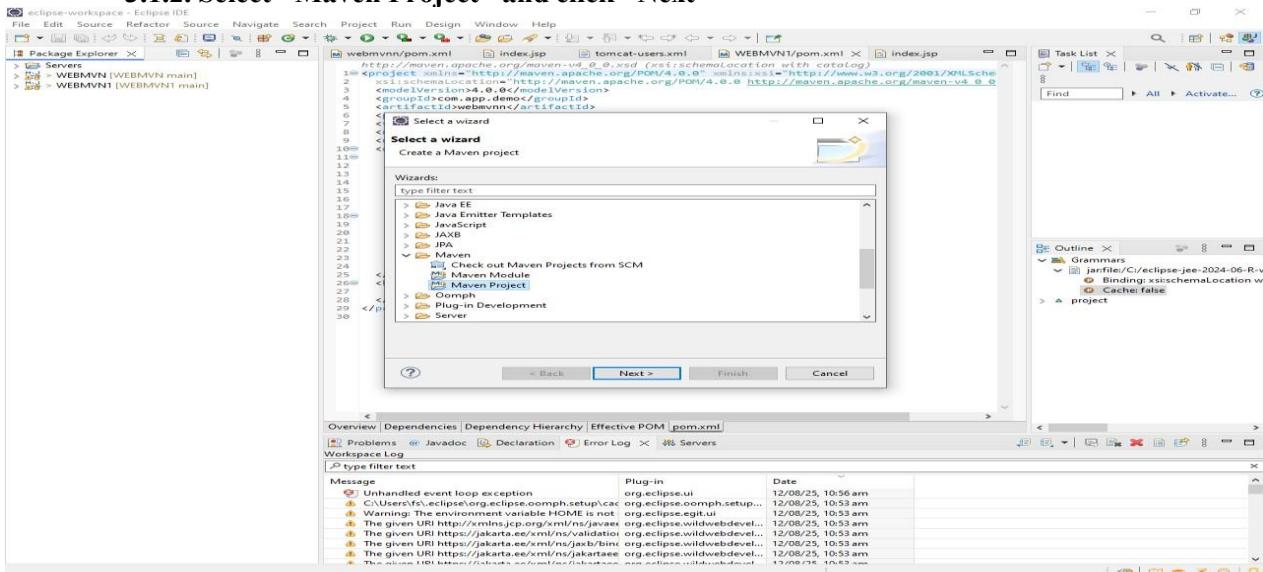
Step 3. Create a New Maven Project

└─ 3.1. File -> New -> Project...

└─ 3.1.1. Expand "Maven"



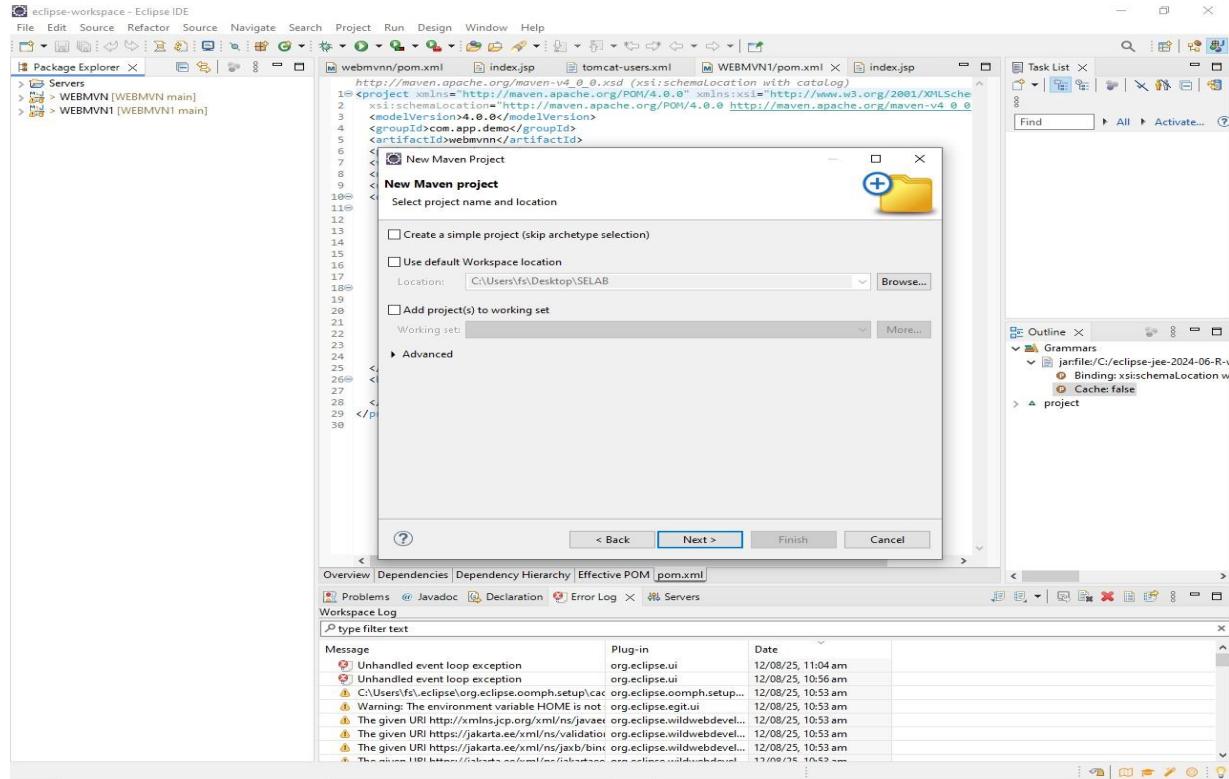
└─ 3.1.2. Select "Maven Project" and click "Next"



Step 4. Set Project Configuration

4.1. Select workspace location (default or custom)

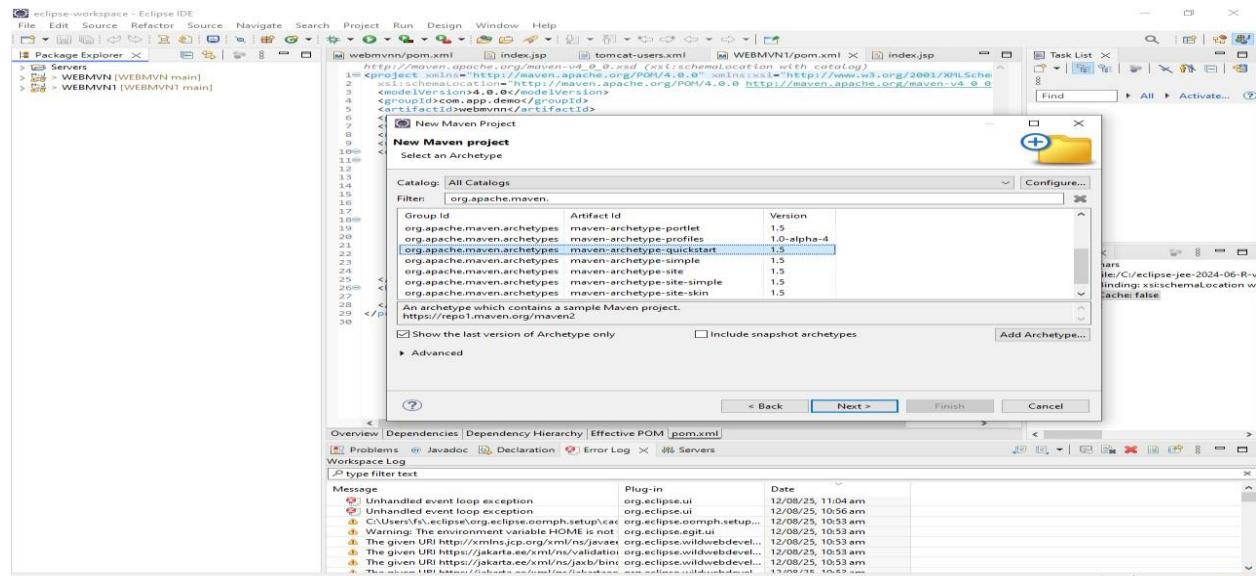
4.2. Click "Next"



Step 5. Choose Maven Archetype

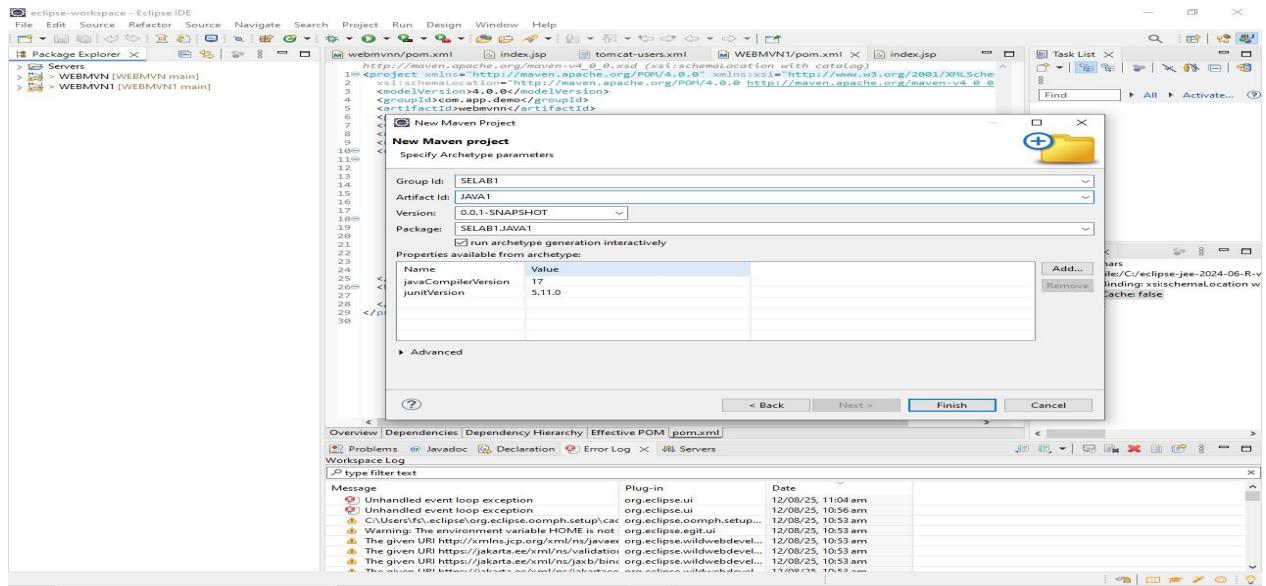
5.1. Select an archetype(e.g "org.apache.maven.archetypes -> maven-archetype-quickstart 1.4 ")

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)

The screenshot shows the Eclipse IDE interface. In the center, there is a code editor window displaying the following Java code:

```

1 package SELAB1.JAVA1;
2
3 public class App {
4     public static void main(String[] args) {
5         System.out.println("Hello world!");
6     }
7 }

```

Below the code editor is a terminal window showing Maven build logs:

```

[terminated] C:\eclipse-jee-2024-06-R-win32-x86_64\plugins\org.eclipse.justmyjava\hotspot\jre\full\win32\x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (12-Aug-2025, 1
[WARNINGS] Don't override file C:\Users\fa\Desktop\SELAB1\JAVA1\src\main\Java\SELAB1\JAVA1\index.jsp
[WARNINGS] Don't override file C:\Users\fa\Desktop\SELAB1\JAVA1\src\main\Java\SELAB1\JAVA1\WebMVN1\pom.xml
[INFO] Project created at 2025-08-12T11:13:56+05:30
[INFO] BUILD SUCCESS
[INFO] Total time: 03:11 min
[INFO] Finished at: 2025-08-12T11:13:56+05:30
[INFO]

```

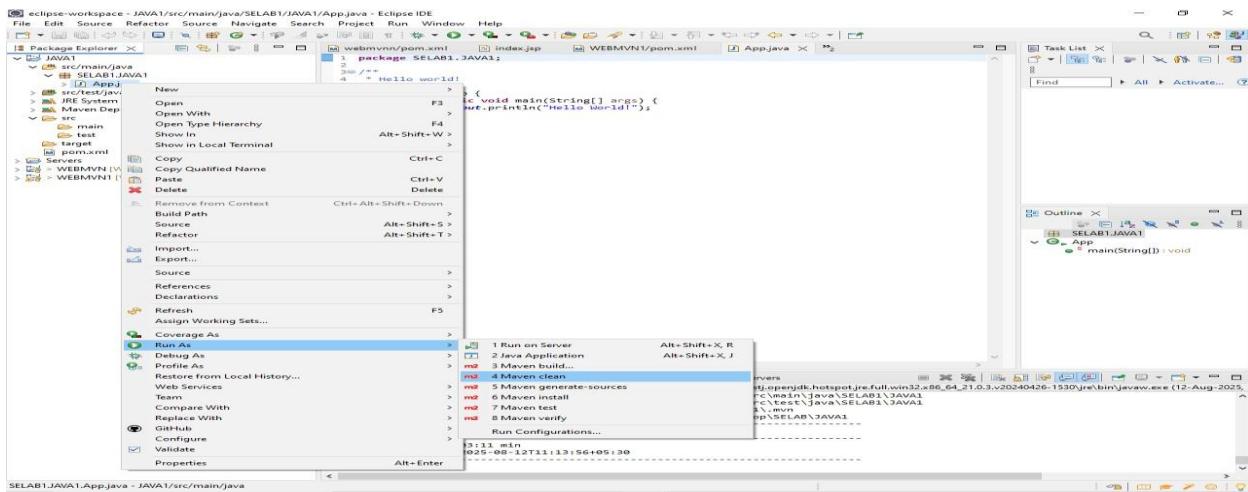
Step 8. Update Project Settings (if needed)

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

└─ 8.2. Ensure dependencies are up to date

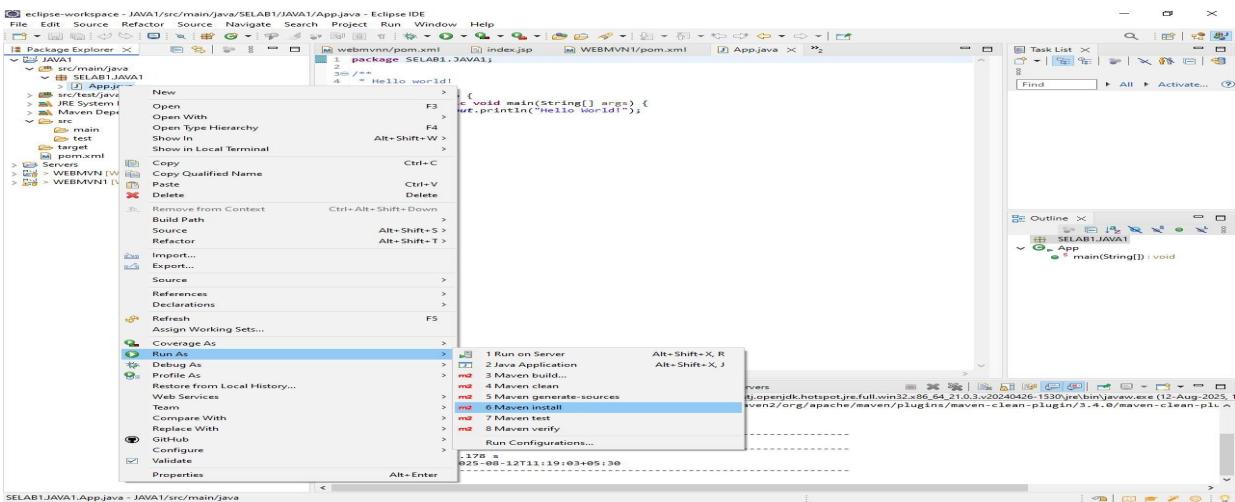
Step 9. Build and Run Maven Project

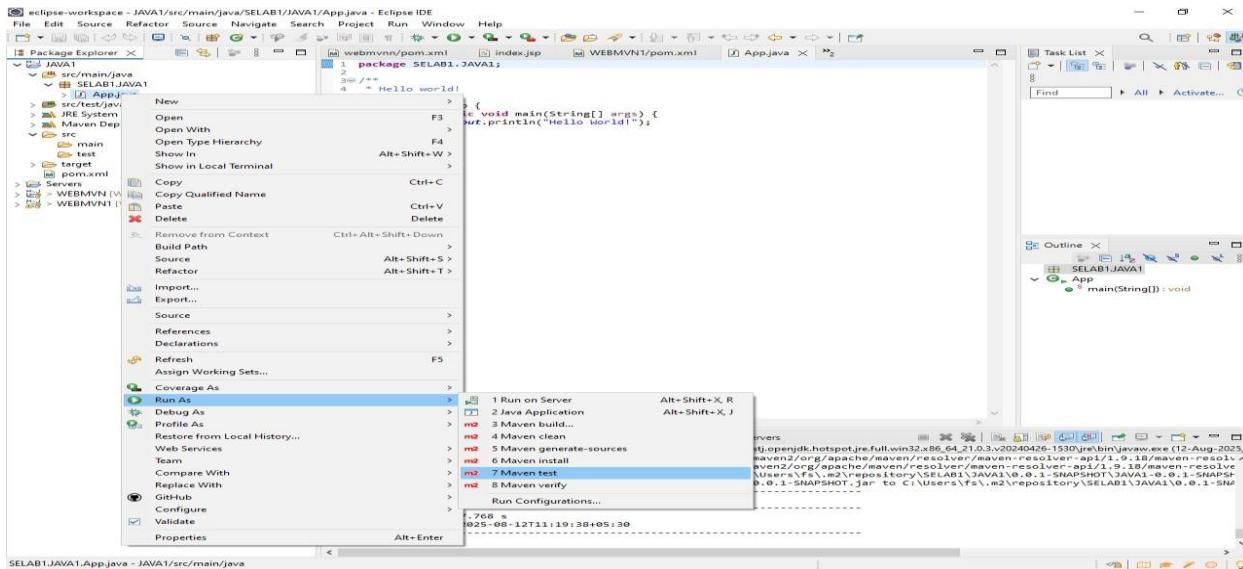
└─ 9.1. Right-click on App.java -> Run As -> Maven Clean



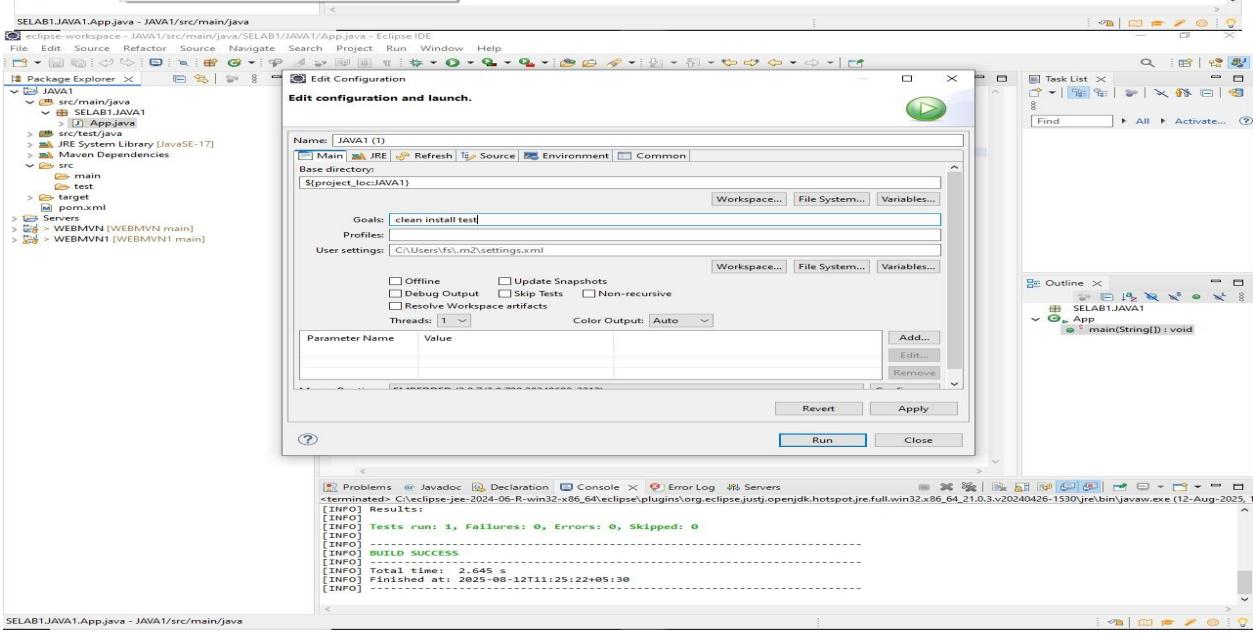
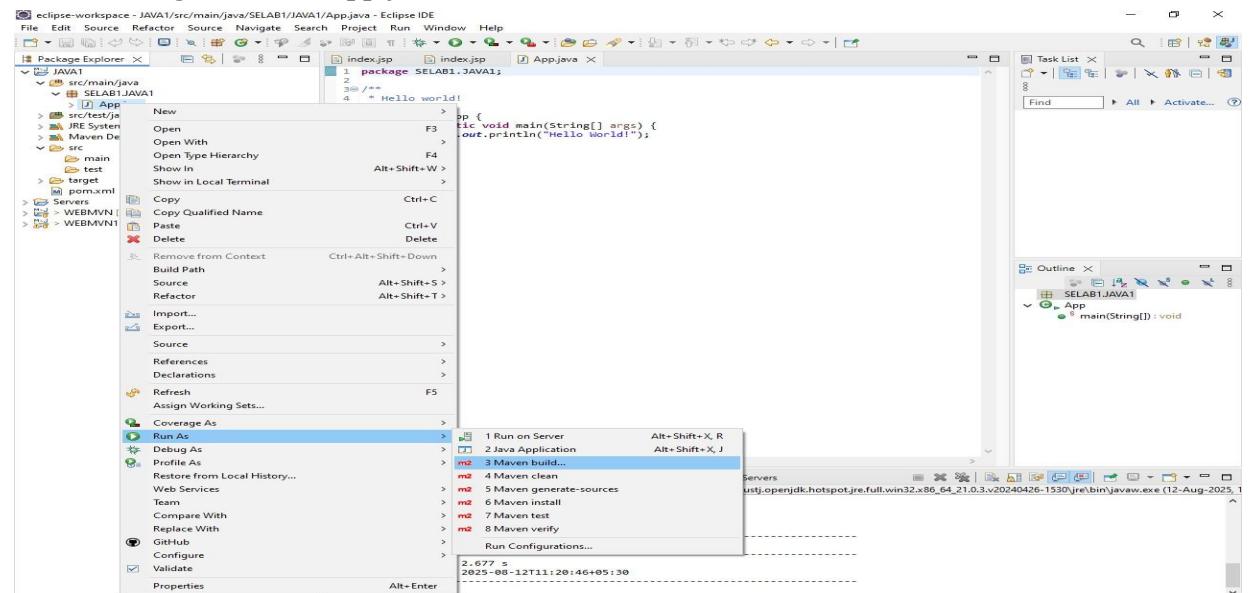
└─ 9.1.1. Right-click on App.java -> Run As -> Maven Install

└─ 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:

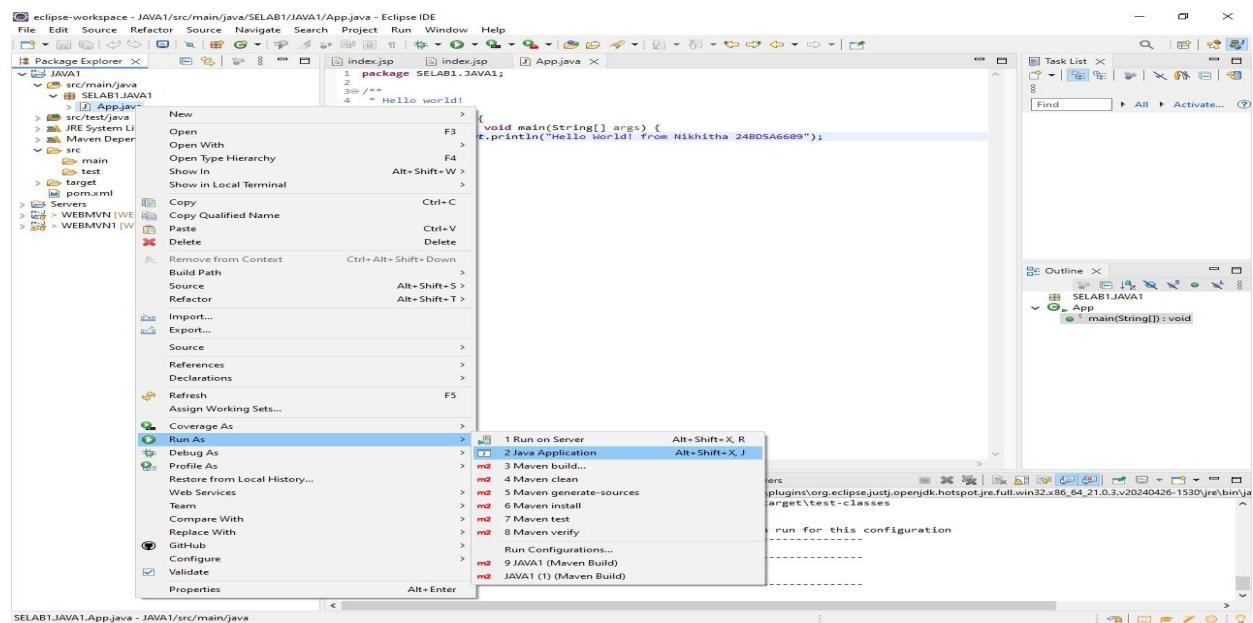
└─ Enter Goals: clean install test

└─ Click on Apply -> Click on Run

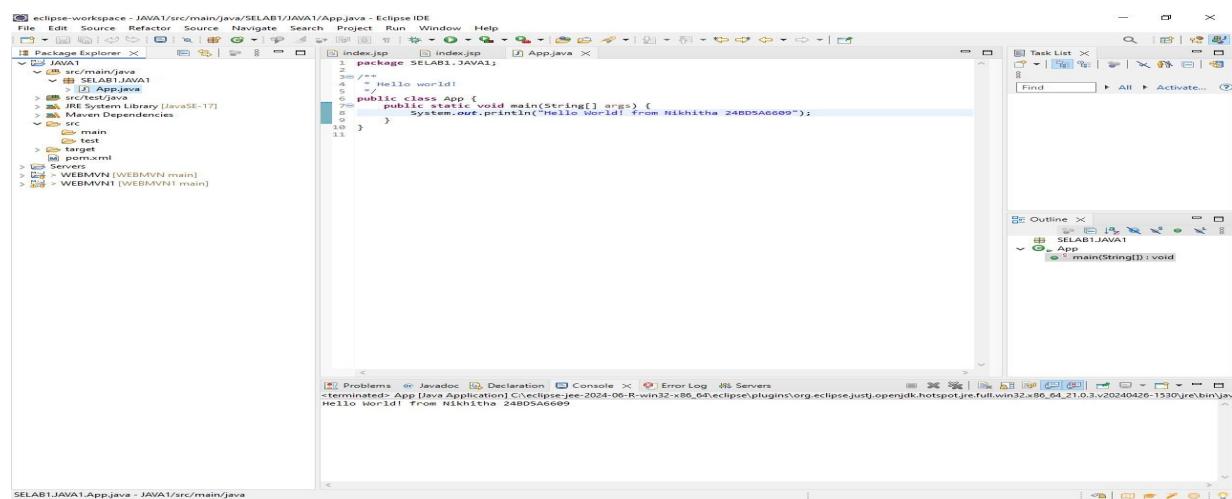
Step 11. Check console for BUILD SUCCESS message.

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

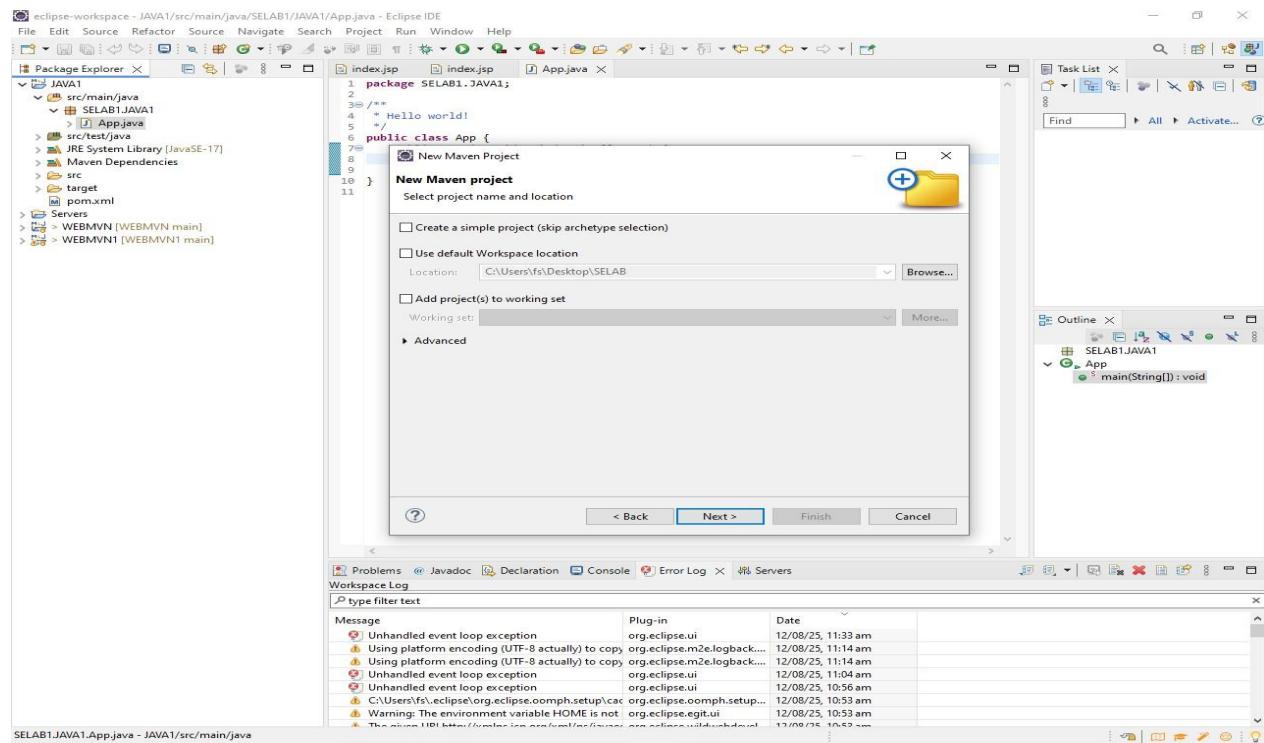
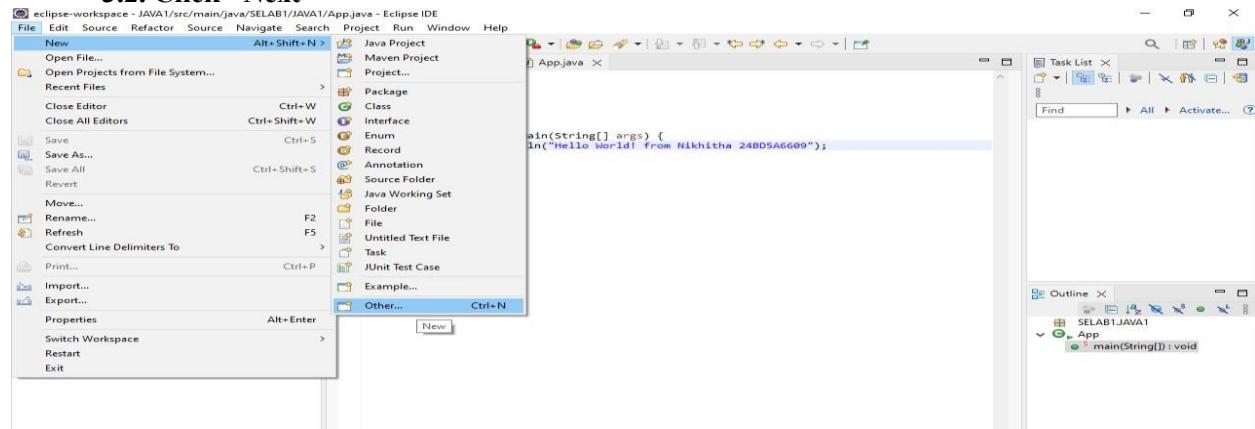
- └─ 2.1. File -> New -> Project...
- └─ 2.1.1. Expand "Maven"
- └─ 2.1.2. Select "Maven Project" and click "Next"

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"

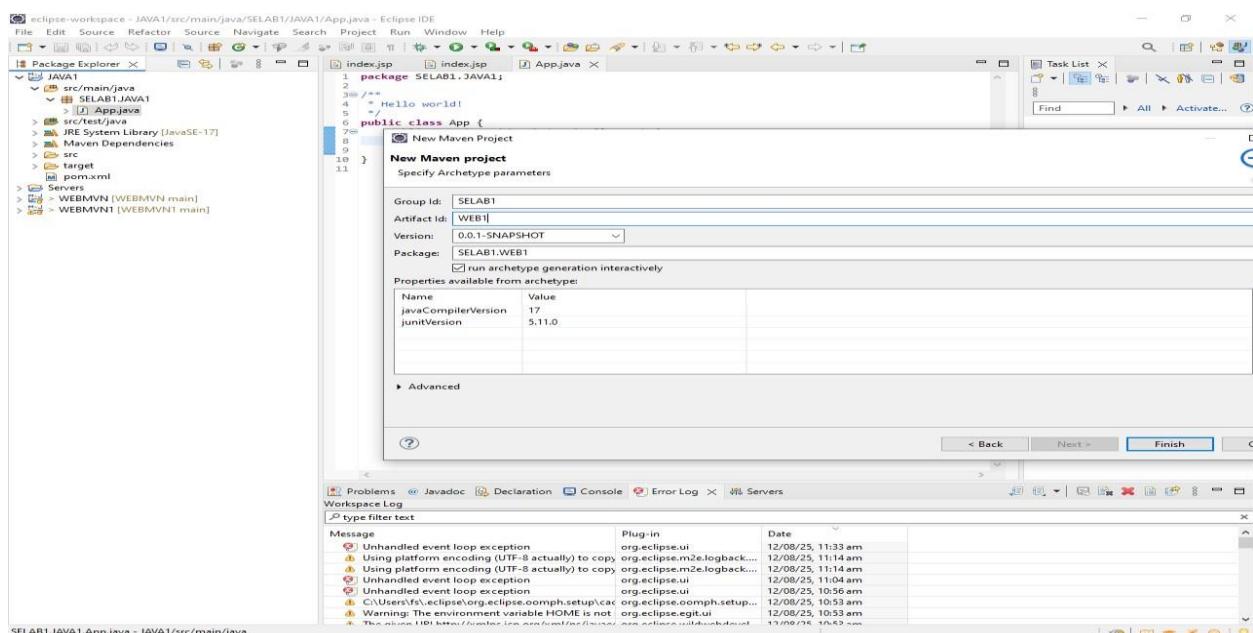
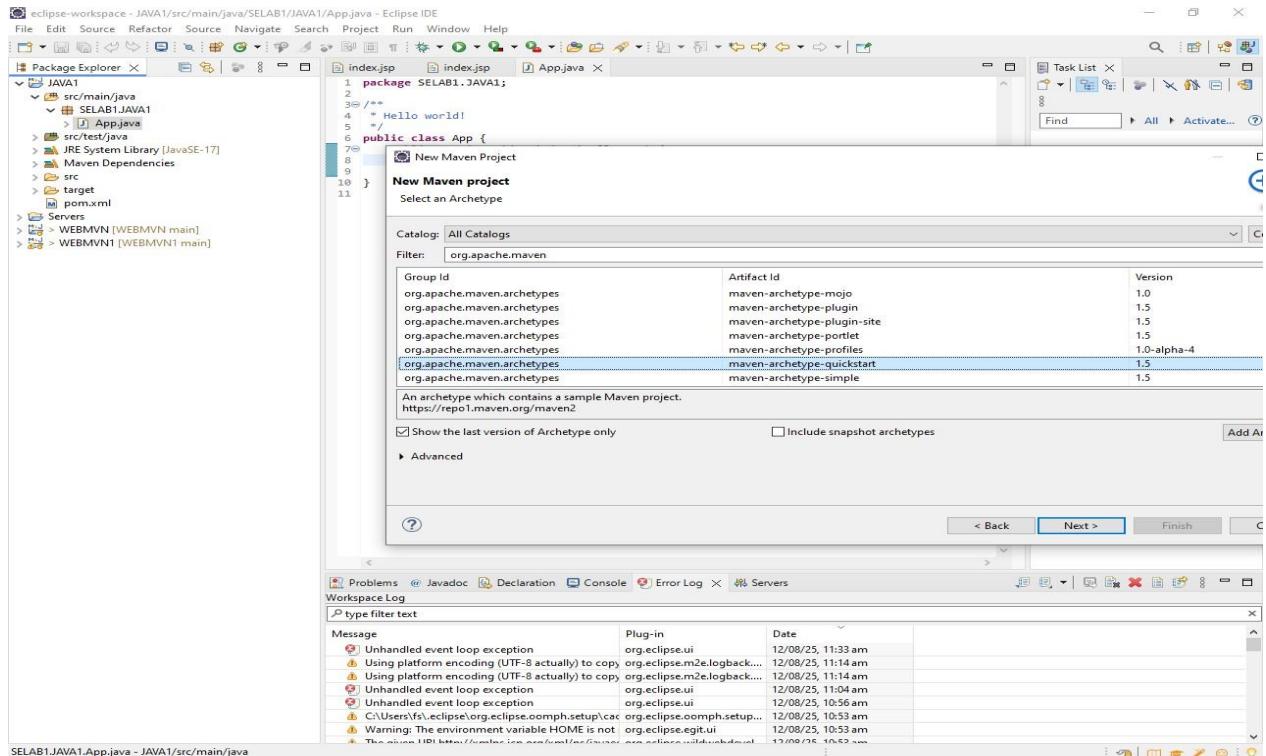


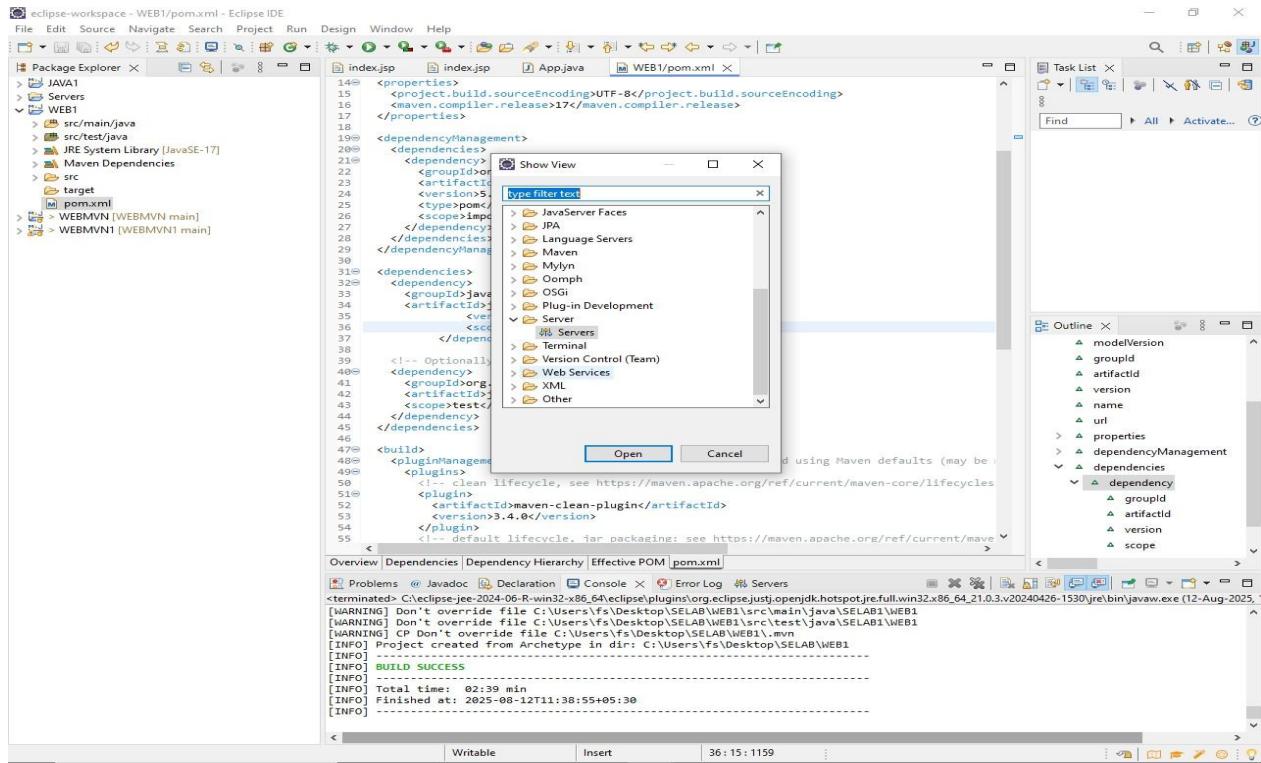
Step 4: Configure the Maven Project

└─ 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.

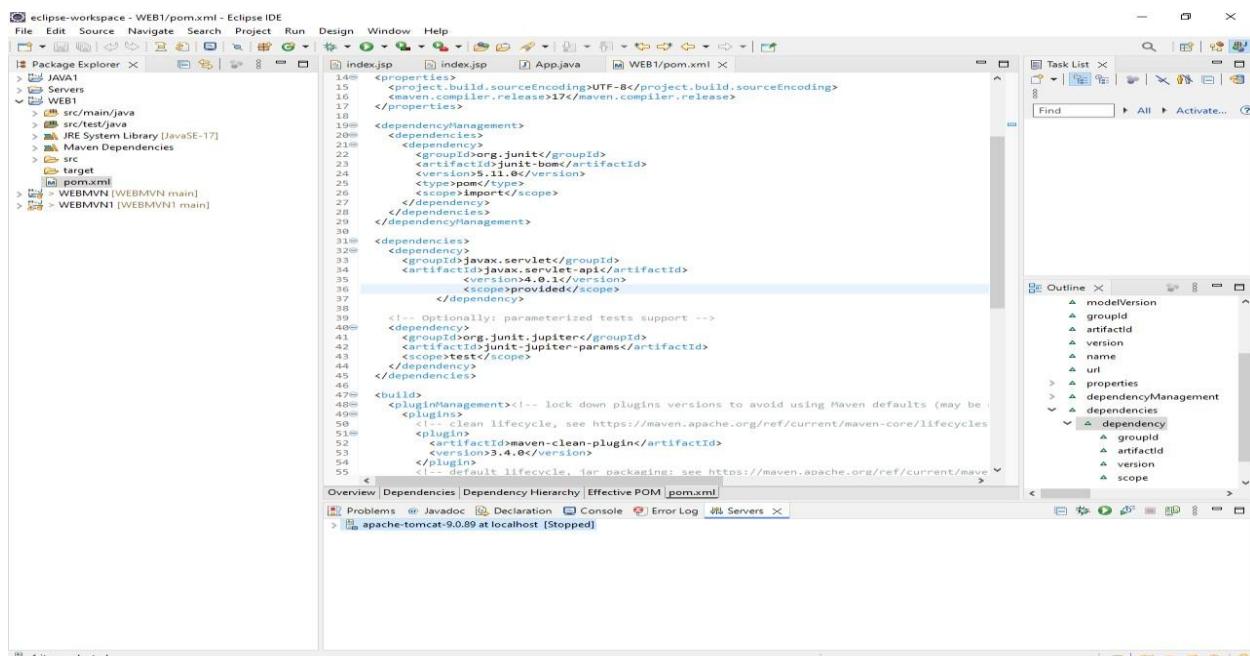




Step 5: Add Maven Dependencies

└─ 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

Example:

```xml

```
<dependency>
 <groupId>javax.servlet</groupId>
 <artifactId>javax.servlet-api</artifactId>
 <version>4.0.1</version>
 <scope>provided</scope>
</dependency>
````
```

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

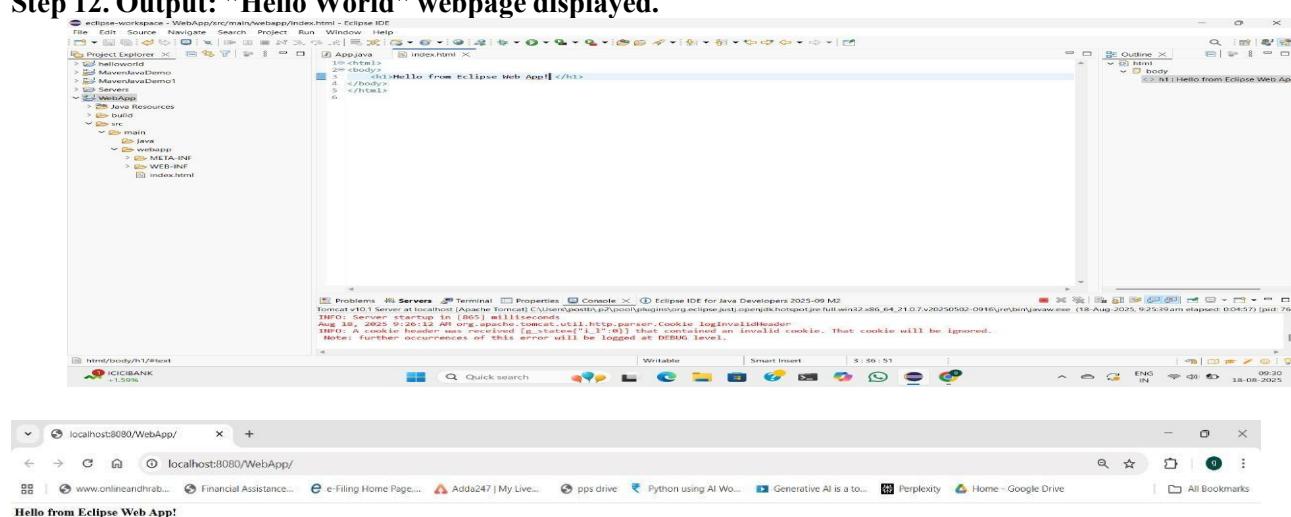
Step 10. Check console for BUILD SUCCESS message.

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):

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.

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

f. 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
f7c205294842bed6bcec3261758e9302730e6740709701556c0b0db8199d2de
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.

Example Redis Commands:

```
127.0.0.1:6379> SET name "Alice"
```

```
OK
```

```
127.0.0.1:6379> GET name
```

```
"Alice"
```

```
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
n
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
2efe79db058 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
n
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
2efe79db058 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 remove
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
2337d2755d16        ubuntu              "/bin/bash"         2 months ago       Exited (255) 2 months ago
229f60b860f3        ubuntu              "/bin/bash"         2 months ago       Exited (255) 2 months ago
75a9f03a7d3c        ubuntu              "/bin/bash"         2 months ago       Exited (255) 2 months ago
06821945c7b6        6015f66923d7      "/bin/bash"         3 months ago       Exited (255) 2 months ago
a42940212930        1e622c5f073b      "/bin/bash"         4 months ago       Exited (255) 3 months ago
3980dbbf3512        1e622c5f073b      "/bin/bash"         4 months ago       Exited (255) 3 months ago
e8a4452f558c        1e622c5f073b      "/bin/bash"         4 months ago       Exited (255) 3 months ago
2ef79d0b058        72297848456d      "/bin/bash"         4 months ago       Exited (255) 4 months ago
d8d24f05424f        72297848456d      "/bin/bash"         6 months ago       Exited (255) 5 months ago
a719ed346bd9        72297848456d      "/bin/bash"         6 months ago       Exited (255) 6 months ago
strange_mendeleev
stoic_wing
inspiring_blackwell
hardcore_goldberg
awesome_wescoff
affectionate_ramanujan
interesting_mcCarthy
hungry_wilbur
nifty_jepsen
jolly_ellis
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
f7c205294842bed66bce3c3261758e9302730e6740709701556c0b0db8199d2de
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

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e301a7f3450	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
2efe79db058	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 * o000o000o000o Redis is starting o000o000o000o
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> RedisSearch version 8.2.5 (Git=222ad3b)
1:M 16 Nov 2025 12:46:45.001 * <search> Low level api version 1 initialized successfully
```

What It Does:

troubleshoot the container

c. Configure and manage networks for container communication.

List Existing Networks

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  
7ffdf9784dec90ae36d58ed2371e96b5241c256392b8ed3fb8f9035fd4574c4
```

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
```

Inspect a Network

```
docker network inspect mynet
```

```
C:\Users\hariv>docker network inspect mynet  
[  
  {  
    "Name": "mynet",  
    "Id": "7ffdf9784dec90ae36d58ed2371e96b5241c256392b8ed3fb8f9035fd4574c4",  
    "Created": "2025-11-16T12:50:38.619762201Z",  
    "Scope": "local",  
    "Driver": "bridge",  
    "EnableIPv4": true,  
    "EnableIPv6": false,  
    "IPAM": {  
      "Driver": "default",  
      "Options": {},  
      "Config": [  
        {  
          "Subnet": "172.19.0.0/16",  
          "Gateway": "172.19.0.1"  
        }  
      ]  
    },  
    "Internal": false,  
    "Attachable": false,  
    "Ingress": false,  
    "ConfigFrom": {  
      "Network": ""  
    },  
    "ConfigOnly": false,  
    "Containers": {  
      "757253d9cb146e2161fd82f610f34a7b523eedf259fa8468754ec85277e45089": {  
        "Name": "redis-server",  
        "EndpointID": "fb68a16ecf11cd5d182b30ab891e5bc94b175065808633d38664fde613e17c0",  
        "MacAddress": "8a:e7:b9:35:f2:9c",  
        "IPv4Address": "172.19.0.2/16",  
        "IPv6Address": ""  
      }  
    },  
    "Options": {  
      "com.docker.network.enable_ipv4": "true",  
      "com.docker.network.enable_ipv6": "false"  
    },  
    "Labels": {}  
  }  
]
```

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      63d0e9591ece8e9bed661a2de13c225f2e295b8d60142947dd29a8c568a25ab
local      281aa4561ead21b125a3abc7de451b867cb0a673f8677925b0a9a7a0dc28ee2c
local      317f86e9bb0fe8fb7af0b92a816d6f503a0c51db049a83200f12e645d871c3a1
local      81405fef6d241a1e81a7f6ee9ffa59ced97dcdf6ea0c2b06c63821f09d5fa58e
local      146231b502f4aa5b9fae6a6083f3d90eb353e25c6637d8aad1e20dd8a45ef1d3
local      a1d8fb026b1ca352ebeae5310c72cab3b97d5f436aa301bd252515cc305570f
local      ae5cc679d8b0b290f53715d89024ed6ea7d826badda15fe7f92ae919e6b59e88
local      af8cb48263f4155f53e1c58b40134753a4d06e16d6cafc2e808859c733bebe14
local      c6e224923a574186212a02278d1ea65d4d915a29e15d7f28bf0f0f4b91aeee02a
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-redis -v mydata:/data redis

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

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

List Docker Images

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
redis	latest	5c7c0445ed86	12 days ago	200MB
hari151/mavenapp	latest	adb0df0a789f	2 months ago	713MB
mywebapp	latest	adb0df0a789f	2 months ago	713MB
hari151/csimage	latest	e40475ea9774	2 months ago	642MB
csimage	latest	e40475ea9774	2 months ago	642MB
gcr.io/k8s-minikube/kicbase	<none>	7171c97a5162	2 months ago	1.84GB
gcr.io/k8s-minikube/kicbase	v0.0.48	41454ef774d0	2 months ago	1.84GB
mysql	8	6e60ad6d61d8	2 months ago	1.07GB
postgres	15	1cd9dd548427	2 months ago	628MB
mongo	6	4bf2adba7807	2 months ago	1.05GB
my-nginx-image	latest	1bab28abb812	2 months ago	79.4MB
nginx	alpine	42a516af16b8	3 months ago	79.3MB
nginx	latest	d5f28ef21aab	3 months ago	279MB
hello-world	latest	a0dfb02aac21	3 months ago	20.3kB
os-2.1	latest	e05918bcd094	5 months ago	1.61GB
<none>	<none>	8474330a392b	5 months ago	1.61GB
<none>	<none>	41ef677d0d15	5 months ago	1.61GB
<none>	<none>	dcb6cca5e9e0	5 months ago	1.61GB
<none>	<none>	ea0207c5da39	5 months ago	1.61GB
<none>	<none>	dabbc152cb2e	5 months ago	1.61GB
<none>	<none>	6e45e8eac41e	5 months ago	1.61GB
<none>	<none>	00f6d912f1e5	6 months ago	1.61GB
<none>	<none>	ef38734b3cbc	6 months ago	1.61GB
<none>	<none>	93dbb2f2f2bc	6 months ago	1.61GB
<none>	<none>	8c40e1e2f2b0	6 months ago	1.61GB
<none>	<none>	362f7ed30fab	6 months ago	1.61GB
<none>	<none>	54cc762d89d2	7 months ago	1.6GB
<none>	<none>	ab661d4e1f19	7 months ago	1.6GB
os-1	latest	c0e6da14ea0b	9 months ago	1.58GB
my-app	latest	36a014b90968	9 months ago	196MB
task	latest	36a014b90968	9 months ago	196MB
ubunt-1	latest	51cfbf730c88	9 months ago	803MB
ubuntu-1	latest	013064897341	9 months ago	489MB
ubuntu	latest	72297848456d	9 months ago	117MB
jasonrivers/nagios	latest	2a7c2b20d118	12 months ago	1.36GB

\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
```

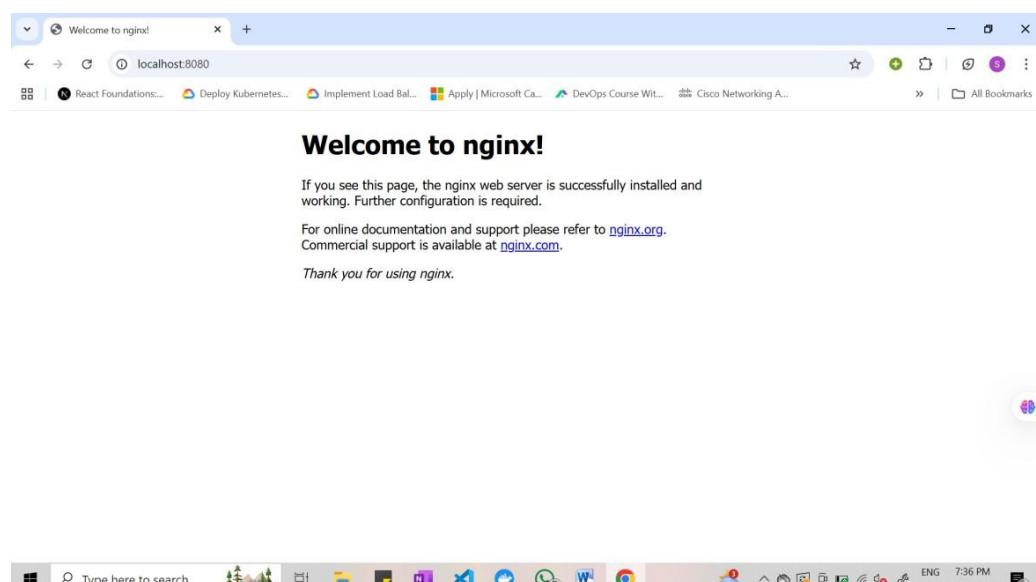
II. Run the setup:

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.

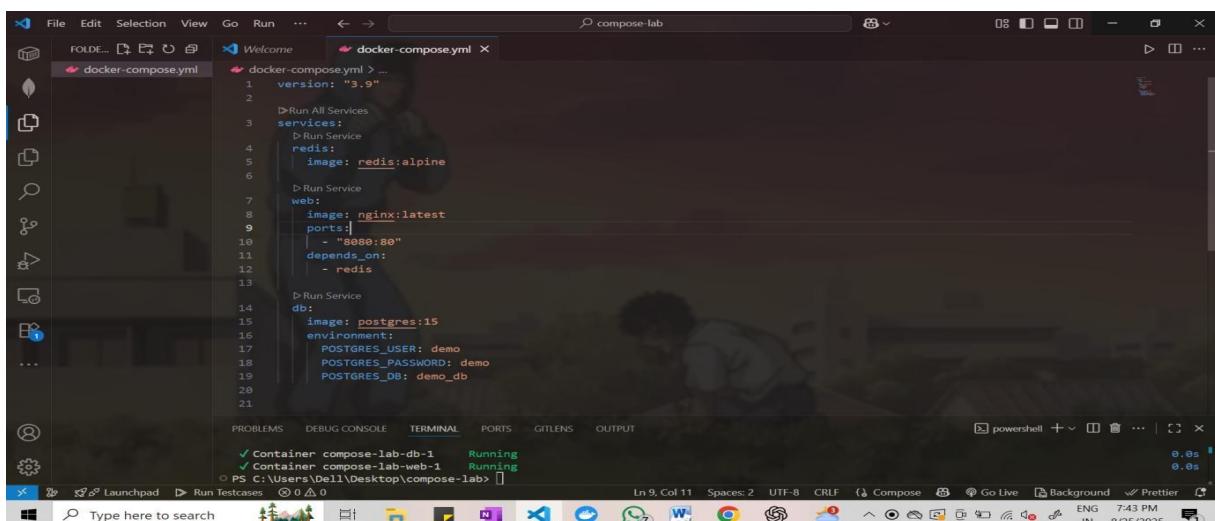
I. Modify docker-compose.yml to add a Redis cache:

// 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
```



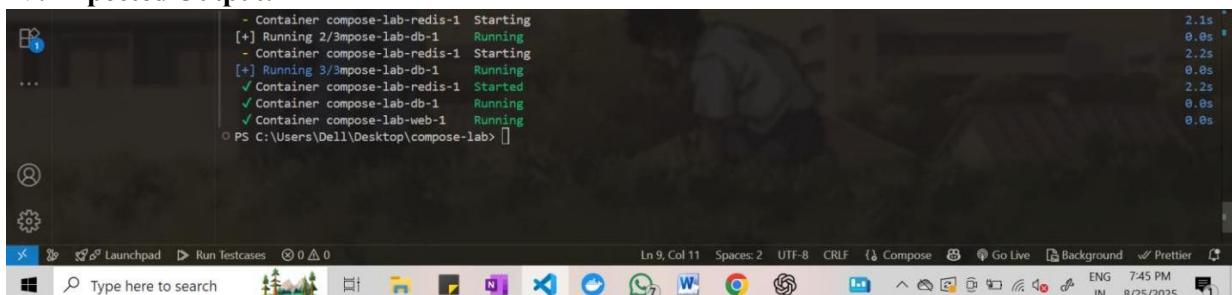
```
version: "3.9"  
services:  
  redis:  
    image: redis:alpine  
  web:  
    image: nginx:latest  
    ports:  
      - "8080:80"  
    depends_on:  
      - redis  
  db:  
    image: postgres:15  
    environment:  
      POSTGRES_USER: demo  
      POSTGRES_PASSWORD: demo  
      POSTGRES_DB: demo_db
```

III. Restart the setup:

docker compose up -d

docker compose ps

IV. Expected Output:



```
- Container compose-lab-redis-1 Starting  
[+] Running 2/3compose-lab-db-1  
- Container compose-lab-redis-1 Starting  
[+] Running 3/3compose-lab-db-1  
✓ Container compose-lab-redis-1 Started  
✓ Container compose-lab-db-1 Running  
✓ Container compose-lab-web-1 Running
```

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:

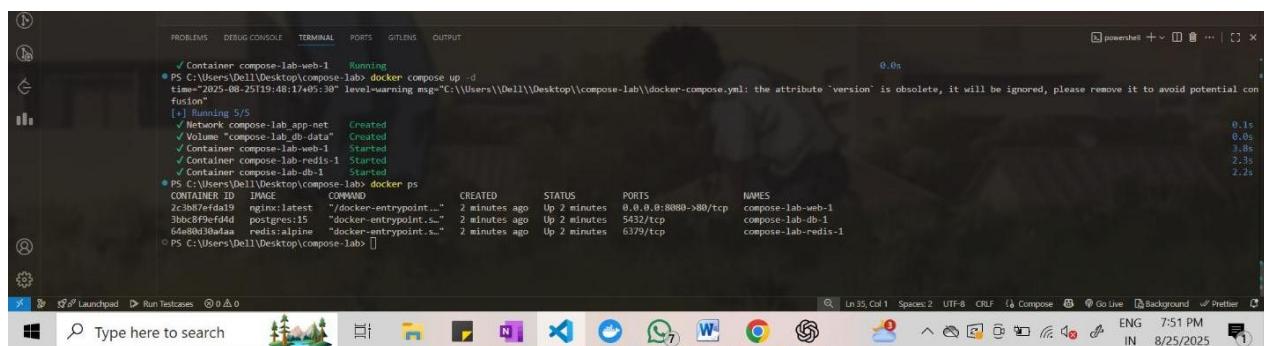
docker compose up -d

Check that Nginx and Postgres work there as well.

III. Expected Output:

The same services run on the new machine without changes.

Now:



```
PS C:\Users\DeLL\Desktop\compose-labs> docker compose up -d
[+] Running 5/5
  ✓ Container compose-lab-web-1   Running
  ✓ Network compose-lab_app-net  Created
  ✓ Volume "compose-lab_db-data" Created
  ✓ Container compose-lab-web-1   Started
  ✓ Container compose-lab-redis-1 Started
  ✓ Container compose-lab-db-1   Started
PS C:\Users\DeLL\Desktop\compose-labs> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
2c3b7efda19        nginx:latest       "/docker-entrypoint..."   2 minutes ago     Up 2 minutes      0.0.0.0:8080->80/tcp   compose-lab-web-1
3b0c89fcfd4d       postgres:15        "docker-entrypoint..."   2 minutes ago     Up 2 minutes      5432/tcp           compose-lab-db-1
6de8bd30eada       redis:alpine        "docker-entrypoint..."   2 minutes ago     Up 2 minutes      6379/tcp           compose-lab-redis-1
PS C:\Users\DeLL\Desktop\compose-labs>
```

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



A screenshot of a Windows desktop environment. At the bottom, the taskbar shows the Start button, a search bar with placeholder text "Type here to search", and several pinned icons for Microsoft Edge, File Explorer, Task View, and others. Above the taskbar, the Start menu is open, displaying a list of recently used applications and system links.

The main window is a terminal or code editor titled "compose-lab". It displays the contents of a "docker-compose.yml" file. The file defines a stack with three services: "db" (PostgreSQL 15), "redis" (Redis 6.2), and "web" (Nginx 1.15). The "web" service depends on "db" and "redis". The terminal shows the output of running "docker compose up -d", indicating successful container creation. Below the terminal, a command-line interface (CLI) session is shown, where the user runs "psql" inside the "db" container to create a "users" table and insert two rows of data (Alice and Bob).

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

IV. (optional with psql).

V. Remove containers:

docker compose down

VI. Start again:

docker compose up -d

VII. Expected Output:

Database data persists across restarts.

Services communicate via the app-net network using service names.



```
fusion"
[+] Running 3/3
  ✓ Container compose-lab-db-1  Running
  ✓ Container compose-lab-redis-1  Running
  ✓ Container compose-lab-web-1  Running
● PS C:\Users\DATA\Desktop\compose-lab: docker exec -it compose-lab-db-1 psql -U demo -d demo_db
psql (15.14 (Debian 15.14-1.pgdg13+1))
Type "help" for help.

demo_db=# CREATE TABLE users (
demo_db(#   id SERIAL PRIMARY KEY,
demo_db(#   name VARCHAR(50),
demo_db(#   email VARCHAR(100)
demo_db(# );
CREATE TABLE
demo_db=# INSERT INTO users (name, email) VALUES
demo_db# ('Alice', 'alice@example.com'),
demo_db# ('Bob', 'bob@example.com');
demo_db# SELECT * FROM users;
 id | name |      email
----+-----+
 1 | Alice | alice@example.com
 2 | Bob  | bob@example.com
(2 rows)

demo_db# \q
PS C:\Users\DATA\Desktop\compose-lab>
```



```
[+] Running 4/4
  ✓ Network compose-lab.app-net  Created
  ✓ Container compose-lab-redis-1  Started
  ✓ Container compose-lab-db-1  Started
  ✓ Container compose-lab-web-1  Started
● PS C:\Users\DATA\Desktop\compose-lab: docker ps
CONTAINER ID        IMAGE               COMMAND             STATUS              PORTS               NAMES
ac10f359f010        "postgres:15"       "postgres"         About a minute ago   Up About a minute   5432/tcp            compose-lab-db-1
82e0607fb0         "redis:5.0"        "redis"           About a minute ago   Up About a minute   6379/tcp            compose-lab-redis-1
02c0e0333a         "nginx:1.23.3"     "nginx"          About a minute ago   Up About a minute   80/tcp, 443/tcp   compose-lab-web-1
psql (15.14 (Debian 15.14-1.pgdg13+1))
Type "help" for help.

demo_db=# select * from users
demo_db# ^
ERROR: syntax error at or near "select"
LINE 1: select * from users
demo_db# ^

demo_db=# select * from users;
ERROR: syntax error at or near "select"
LINE 1: select * from users;
demo_db# ^

demo_db# select * from users;
 id | name |      email
----+-----+
 1 | Alice | alice@example.com
 2 | Bob  | bob@example.com
(2 rows)
```

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"]

```

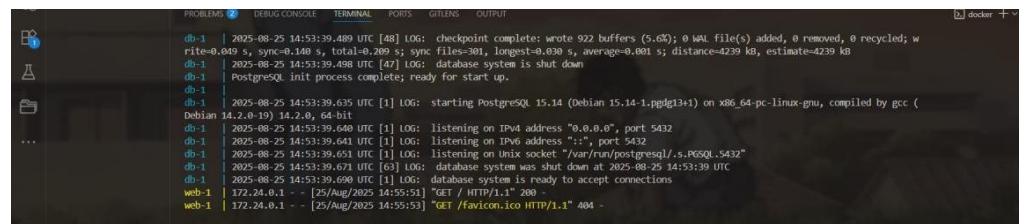
III. Update docker-compose.yml:

web:

```

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

```

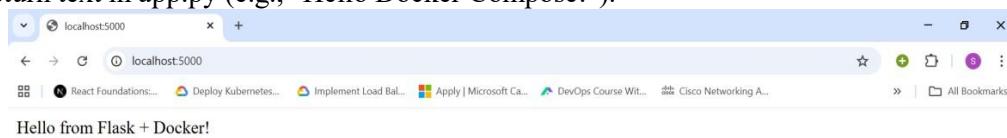


IV. Run:

docker compose up --build

Visit <http://localhost:5000>.

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

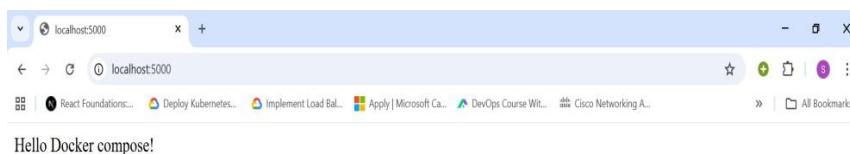


V. Rebuild:

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

└─ Example:

```
'''xml
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
...
'''
```

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.

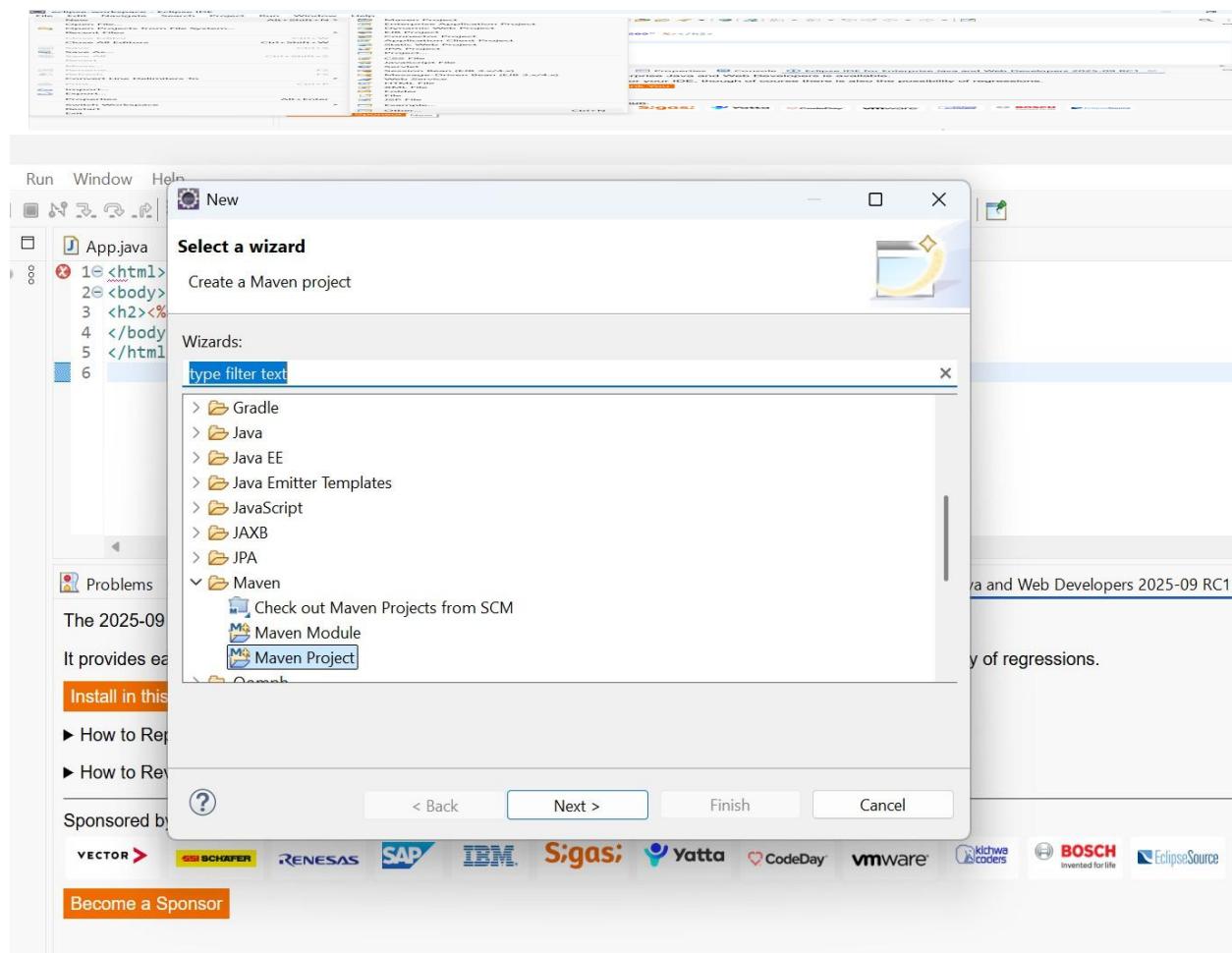
c. Resolve errors and conflicts arising from dependency mismatches.

- 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

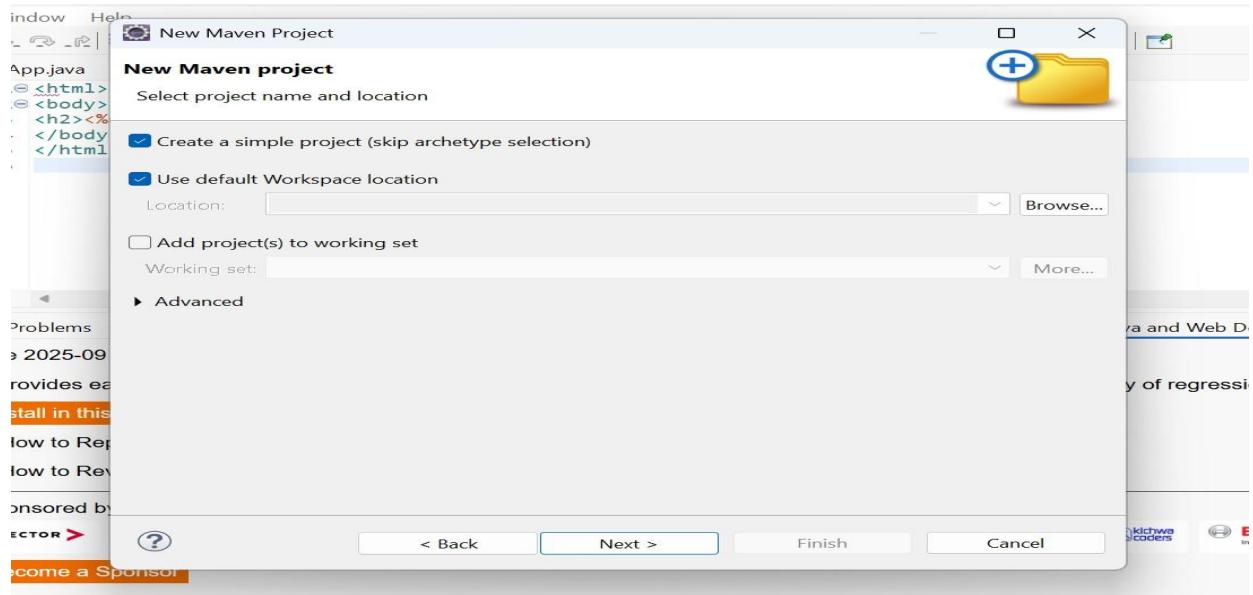
d. Work with parent and multi-module Maven projects.

Creating a Multi-Module Maven Project:

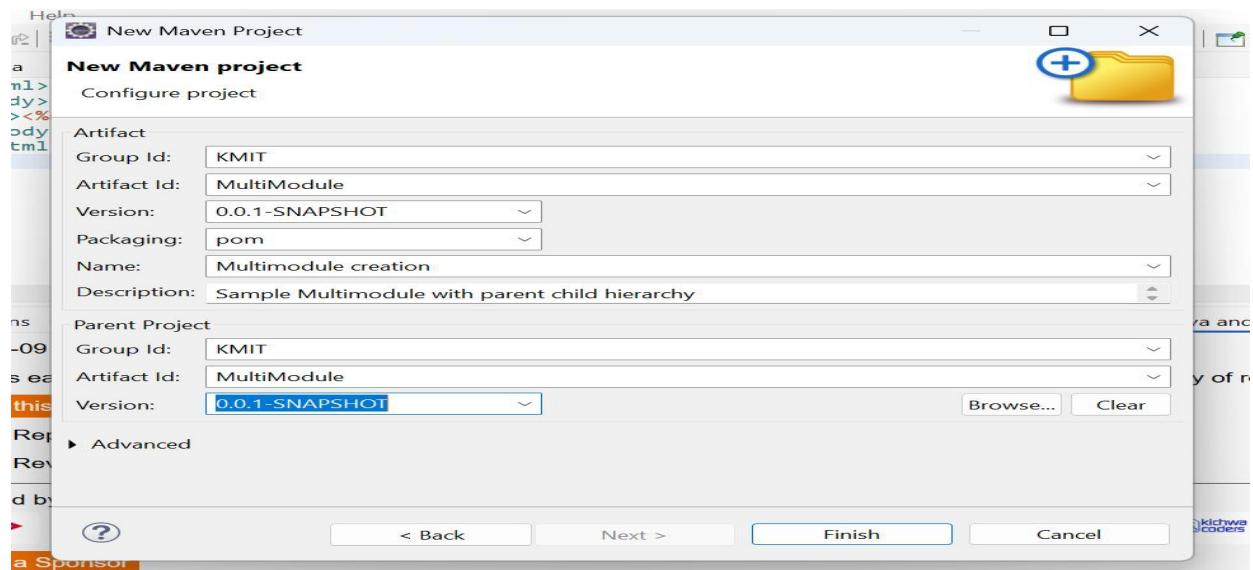
1. Open eclipse-> file-> new -> other -> Maven -> MavenProject ->click on next.



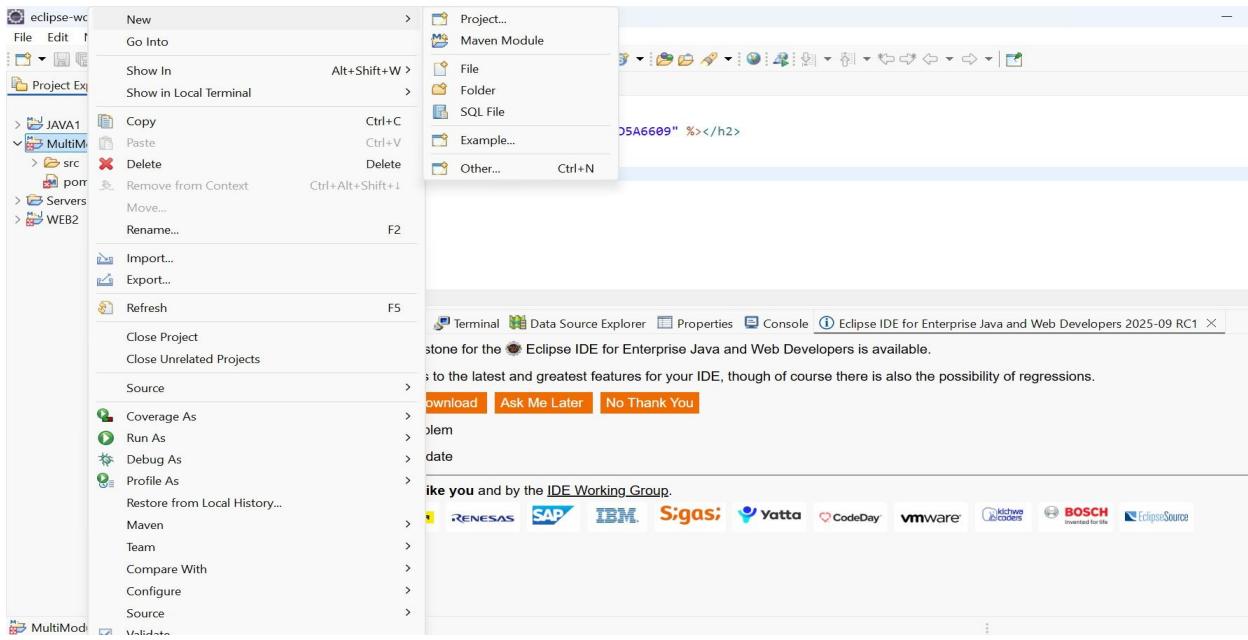
- Check the first option: Create a Simple project(skip archetype selection) and click on next.



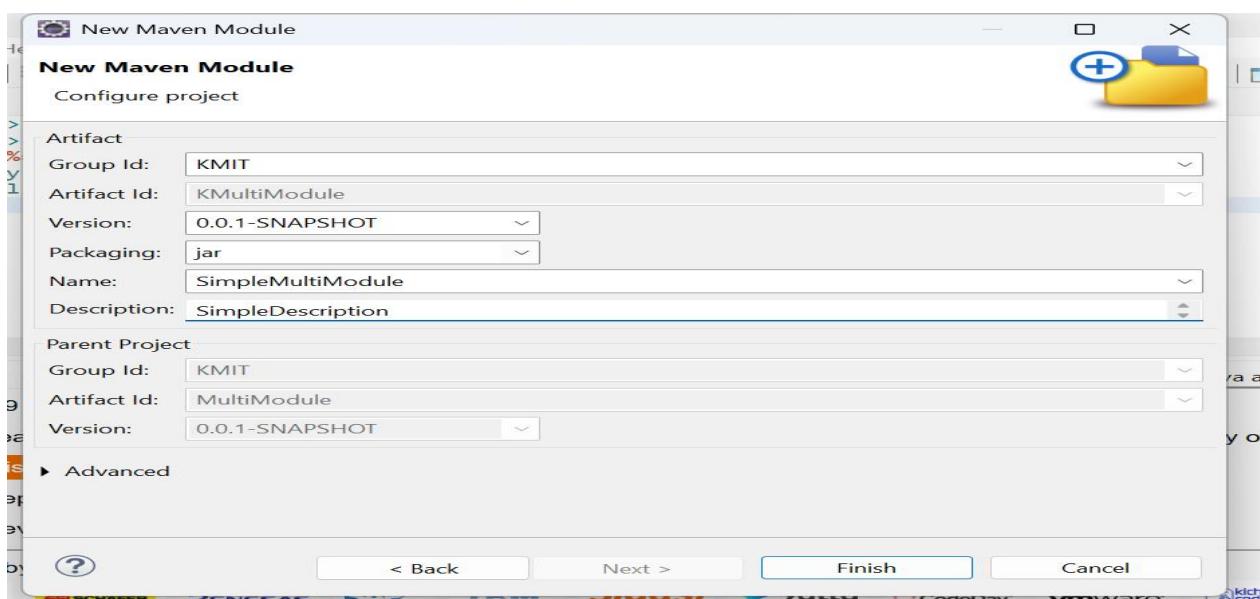
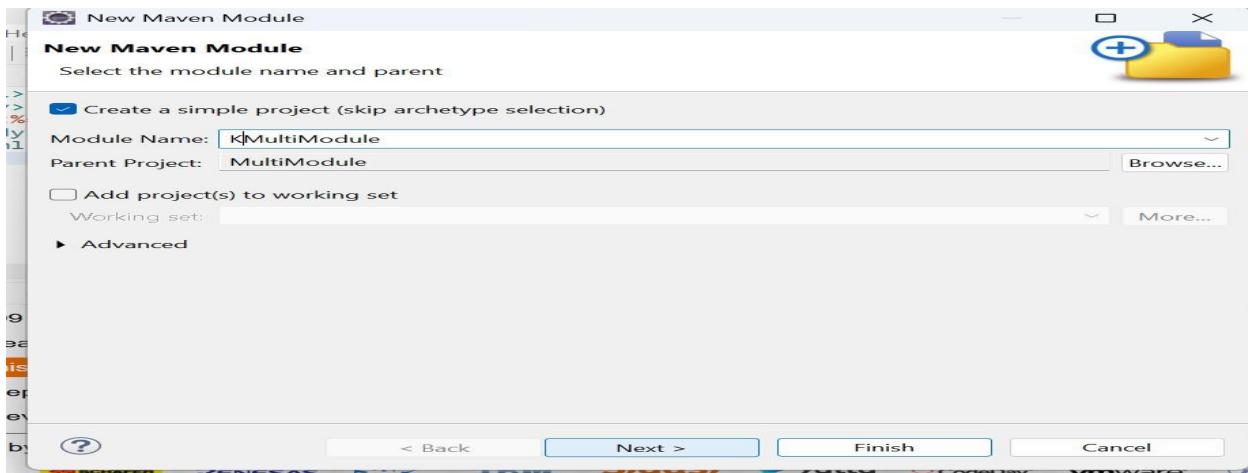
- Give groupID: KMIT, artifact ID: MultiModule, in the packaging tab select the drop down and opt for pom.
- Give Name: Multimodule creation and Description: Sample multimodule with parent child hierarchy and click on next.



- 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.

The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - KMultiModule/pom.xml - Eclipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Design, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left shows a Java project named "JAVA1" and a Maven MultiModule project named "MultiModule" containing "src" and "pom.xml". The central editor area displays the content of "KMultiModule/pom.xml". The XML code is as follows:

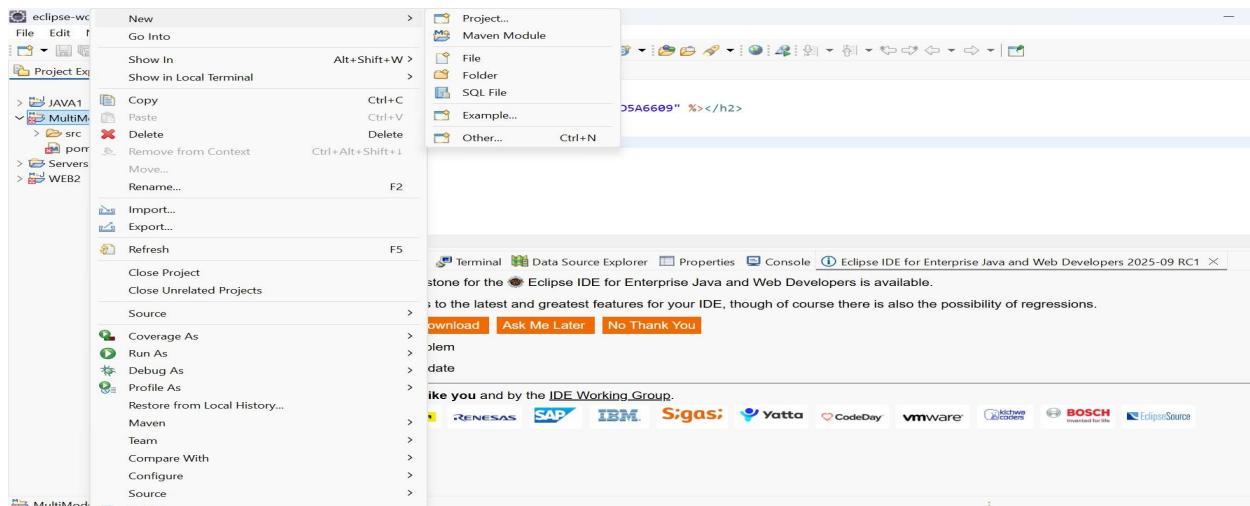
```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>KMIT</groupId>
    <artifactId>MultiModule</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <artifactId>KMultiModule</artifactId>
  <name>SimpleMultiModule</name>
  <description>SimpleDescription</description>
</project>

```

The status bar at the bottom shows tabs for Overview, Dependencies, Dependency Hierarchy, Effective POM, and pom.xml.

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



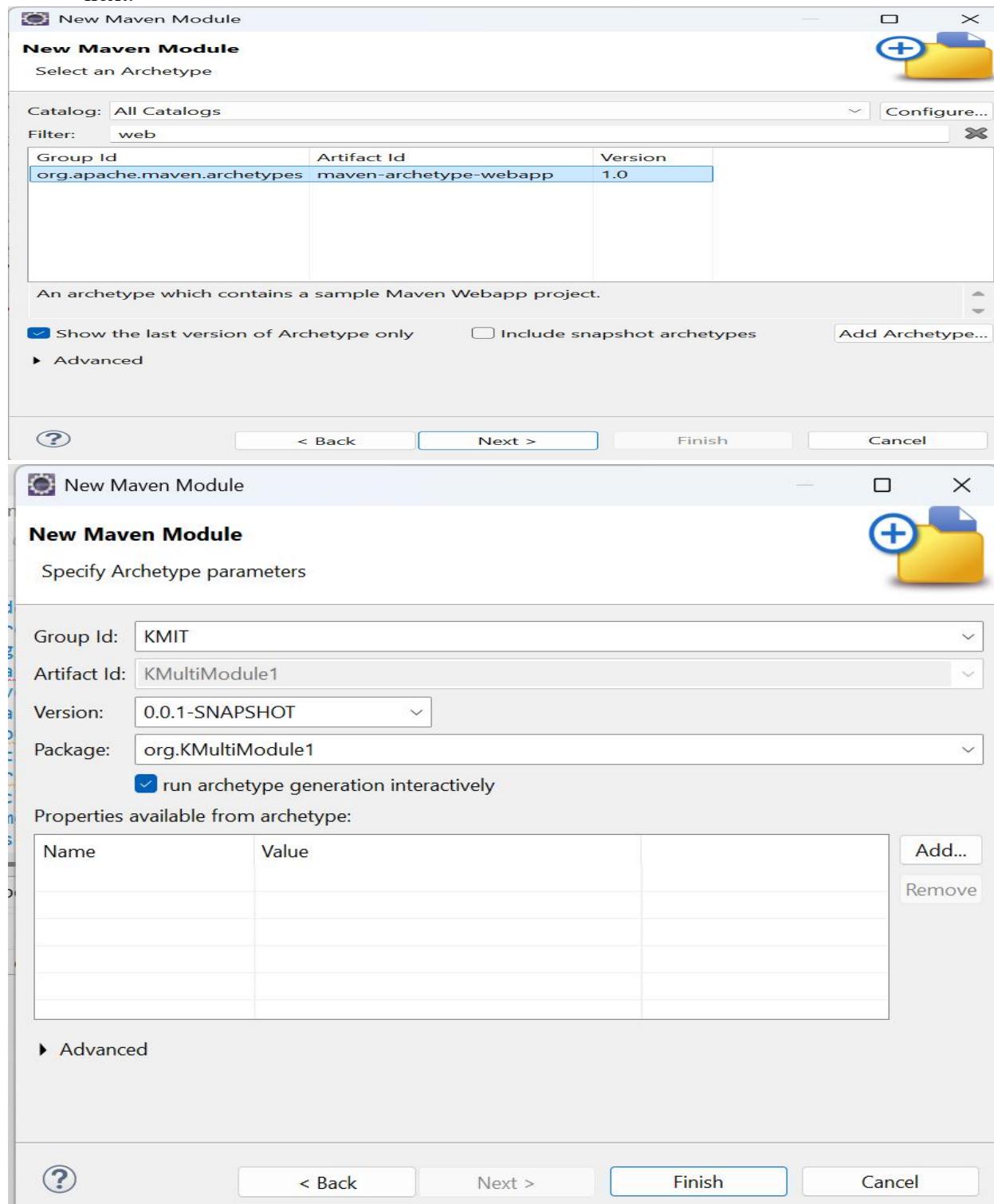
9. Give the artifactID: MultiModuleChild2 and click on next.

The screenshot shows the "New Maven Module" dialog box. The title bar says "New Maven Module". The main area is titled "New Maven Module" and says "Select the module name and parent". It contains the following fields:

- Create a simple project (skip archetype selection)
- Module Name:
- Parent Project:
- Add project(s) to working set
Working set:
-

At the bottom are buttons for "?", "< Back", "Next >" (highlighted in blue), "Finish", and "Cancel".

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.

```

<modelVersion>4.0.0</modelVersion>
<parent>
    <groupId>KMIT</groupId>
    <artifactId>MultiModule</artifactId>
    <version>0.0.1-SNAPSHOT</version>
</parent>
<groupId>KMIT</groupId>
<artifactId>MultiModule</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
<name>Multimodule creation</name>
<description>Sample Multimodule with parent child hierarchy</description>

```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Problems Servers Terminal Data Source Explorer Properties Console

C:\Users\RAMM\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (02-Sept-2025, 10:39:21)
Progress (1): 17/17 MB
Progress (1): 17/17 MB
Progress (1): 17/17 MB
Progress (1): 17 MB

Downloaded from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml (17 MB at 3.6 MB/s)
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-archetype-webapp]
[INFO] Using property: groupId = KMIT
[INFO] Using property: artifactId = KMultiModule1
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = org.KMultiModule1
Confirm properties configuration:
groupId: KMIT
artifactId: KMultiModule1
version: 0.0.1-SNAPSHOT
package: org.KMultiModule1
Y: Y

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

```

</parent>
<groupId>KMIT</groupId>
<artifactId>KMultiModule1</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<name>KMultiModule1 Maven Webapp</name>
<url>http://maven.apache.org</url>
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
    
```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Problems Servers Terminal Data Source Explorer Properties Console

<terminated> C:\Users\RAMM\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (02-Sept-2025, 10:39:25 am) [pi]

[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: C:/Users/RAMM/eclipse-workspace/MultiModule
[INFO] Parameter: package, Value: org.KMultiModule1
[INFO] Parameter: groupId, Value: KMIT
[INFO] Parameter: artifactId, Value: KMultiModule1
[INFO] Parameter: packageName, Value: org.KMultiModule1
[INFO] Parameter: version, Value: 0.0.1-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:/Users/RAMM/eclipse-workspace/MultiModule/KMultiModule1
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 53.485 s
[INFO] Finished at: 2025-09-02T10:40:22+05:30
[INFO] -----

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>

```

```

<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.

```

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

```

```

[INFO] Scanning for projects...
[INFO] BUILD FAILURE
[INFO] Total time: 0.145 s
[INFO] Finished at: 2025-09-02T11:27:31+05:30
[INFO]
[ERROR] No goals have been specified for this build. You must specify a valid lifecycle phase or a goal in the format
[ERROR] [ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1 http://cwiki.apache.org/confluence/display/MAVEN/NoGoalSpecifiedException

```

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

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://www.w3.org/2001/XMLSchema-instance">
  <groupId>KMIT</groupId>
  <artifactId>MultiModule</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>Multimodule creation</name>
  <description>Sample Multimodule with parent child hierarchy</description>
  <modules>
    <module>KMultimodule</module>
    <module>KMultiModule1</module>
  </modules>
</project>

```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

Problems | Servers | Terminal | Data Source Explorer | Properties | Console | Eclipse IDE for Enterprise Java and Web Development

```

<terminated> C:\Users\RAMM\p2\pool\plugins\org.eclipse.jst.jdt.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (02-SubProcess)
[INFO] -----
[INFO] Reactor Summary for Multimodule creation 0.0.1-SNAPSHOT:
[INFO]
[INFO] Multimodule creation ..... SUCCESS [ 0.503 s]
[INFO] SimpleMultiModule ..... SUCCESS [ 0.008 s]
[INFO] KMultiModule1 Maven Webapp ..... SUCCESS [ 0.013 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.709 s
[INFO] Finished at: 2025-09-02T11:30:10+05:30
[INFO] -----

```

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>

```

```
</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)

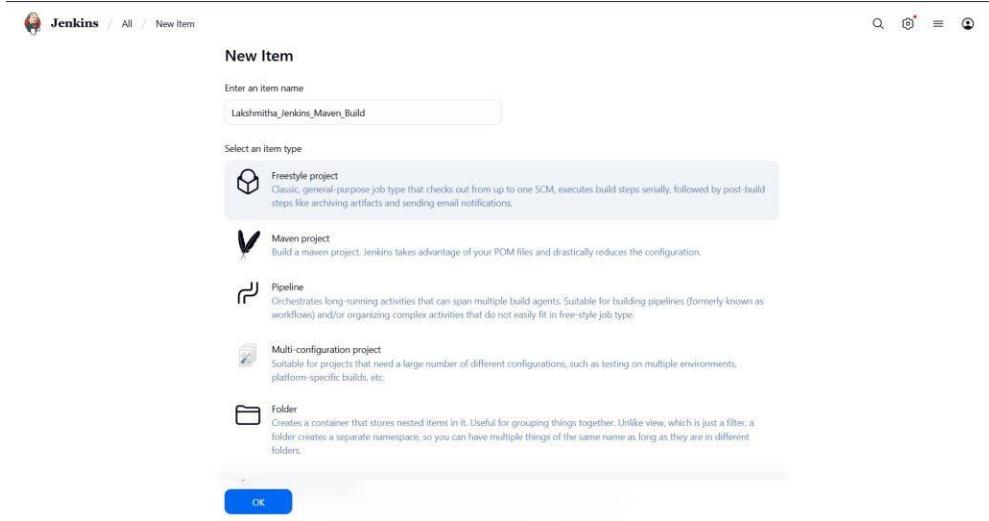
 └─ Click on "New Item" (left side menu)



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

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Advanced

Add build step

Save Apply

- └─ 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

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Advanced

Archive the artifacts

Files to archive

**/*

Advanced

Build other projects

Projects to build

Lakshmitha_Jenkins_Maven_Test

No such project 'Lakshmitha_Jenkins_Maven_Test'. Did you mean 'Lakshmitha_Jenkins_Build'?

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

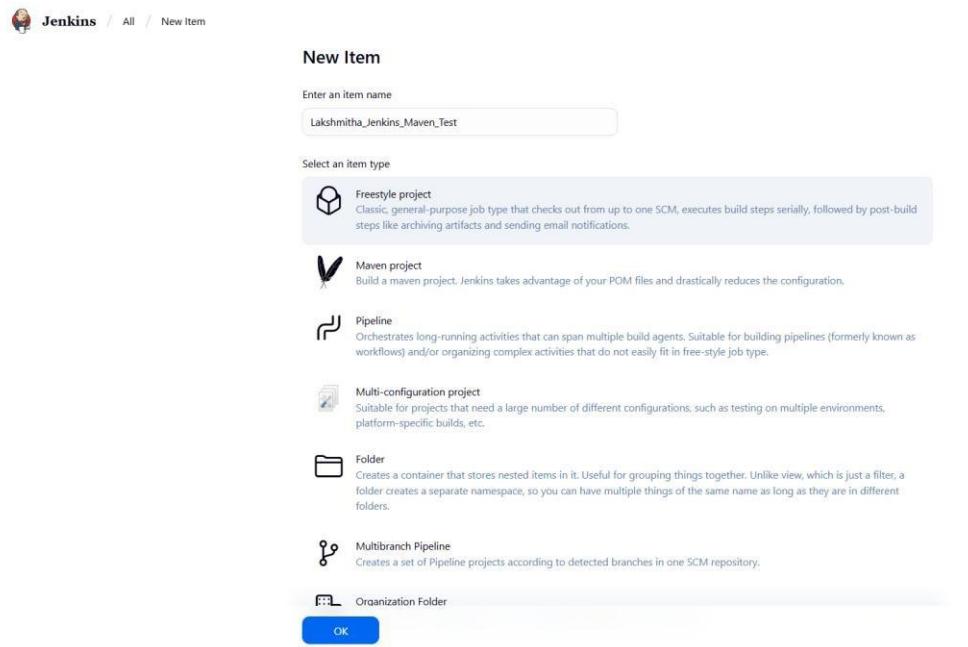
Add post-build action

Save Apply

└─ 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: **/*

The screenshot shows the Jenkins 'Configure' screen for the 'Lakshmitha_Jenkins_Maven_Test' project. The 'Build Steps' section is expanded, showing a 'Copy artifacts from another project' step. It is configured to copy from 'Project name: T' (set to 'Lakshmitha_Jenkins_Maven_Build'), 'Which build: Latest successful build', 'Stable build only' (checked), and 'Artifacts to copy: **/*'. Other sections like 'Environment' and 'Post-build Actions' are also visible.

└─ 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

The screenshot shows the Jenkins configuration page for a job named 'Lakshmitha_Jenkins_Maven_Test'. Under the 'Build Steps' section, there is a step titled 'Invoke top-level Maven targets' with 'MAVEN_HOME' set as the Maven Version and 'test' as the Goals. Under the 'Post-build Actions' section, there is an action titled 'Archive the artifacts' with '**/*' set as the Files to archive. At the bottom of the page are 'Save' and 'Apply' buttons.

└─ 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

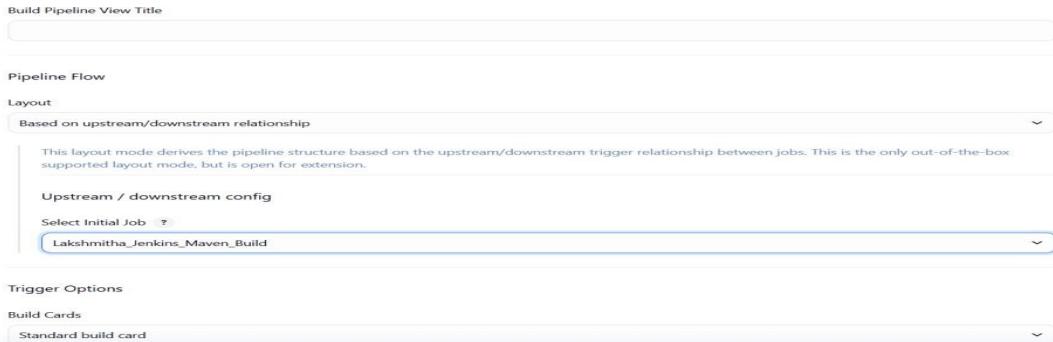
The screenshot shows the Jenkins dashboard with a 'New view' dialog open. The 'Name' field is filled with 'Lakshmitha_Jenkins_Maven_Pipeline'. The 'Type' section has 'Build Pipeline View' selected, which is described as showing jobs in a build pipeline view. There are also options for 'List View' and 'My View'. At the bottom of the dialog is a 'Create' button.

└─ Pipeline Flow:

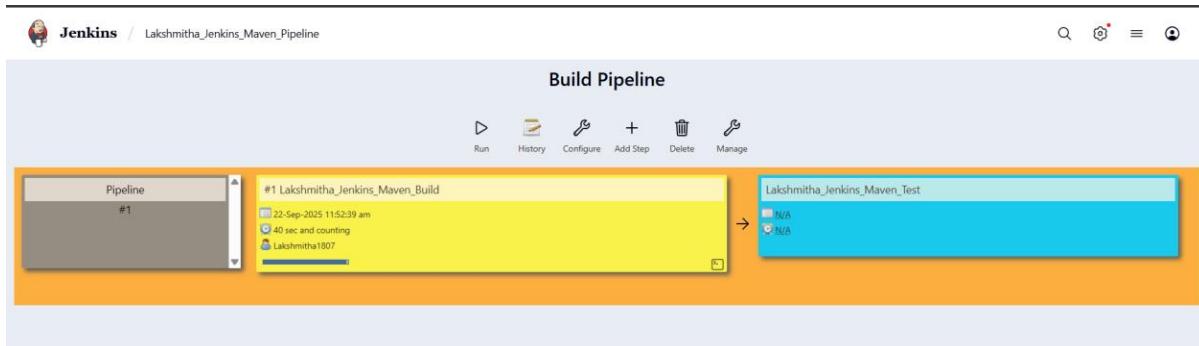
└─ **Layout:** Based on upstream/downstream relationship

└─ Initial job: MavenJava_Build

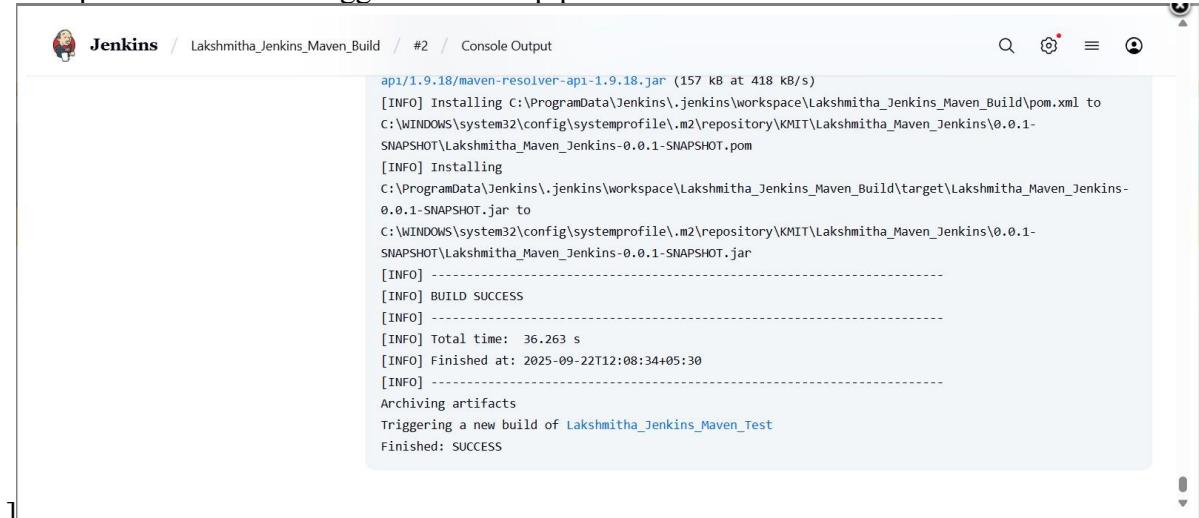
└─ Apply and Save OK



└─ Step 5: Run the Pipeline and Check Output



└─ Click on the trigger to run the pipeline



```
api/1.9.18/maven-resolver-api-1.9.18.jar (157 KB at 418 kB/s)
[INFO] Installing C:\ProgramData\Jenkins\.jenkins\workspace\Lakshmitha_Jenkins_Maven_Build\pom.xml to C:\WINDOWS\system32\config\systemprofile\.m2\repository\KMIT\Lakshmitha_Maven_Jenkins\0.0.1-SNAPSHOT\Lakshmitha_Maven_Jenkins-0.0.1-SNAPSHOT.pom
[INFO] Installing C:\ProgramData\Jenkins\.jenkins\workspace\Lakshmitha_Jenkins_Maven_Build\target\Lakshmitha_Maven_Jenkins-0.0.1-SNAPSHOT.jar to C:\WINDOWS\system32\config\systemprofile\.m2\repository\KMIT\Lakshmitha_Maven_Jenkins\0.0.1-SNAPSHOT\Lakshmitha_Maven_Jenkins-0.0.1-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 36.263 s
[INFO] Finished at: 2025-09-22T12:08:34+05:30
[INFO] -----
Archiving artifacts
Triggering a new build of Lakshmitha_Jenkins_Maven_Test
Finished: SUCCESS
```

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

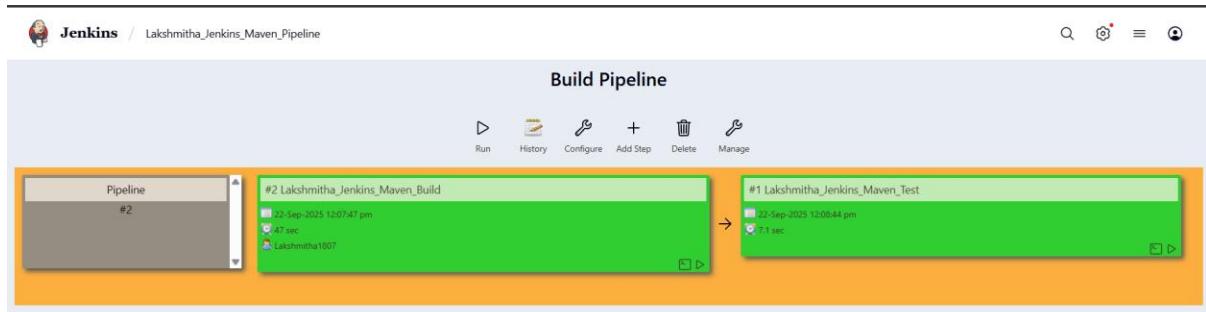
```

[INFO] -----
[INFO] T E S T
[INFO] -----
[INFO] Running KMIT.Lakshmitha_Maven_Jenkins.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.040 s -- in
KMIT.Lakshmitha_Maven_Jenkins.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  3.589 s
[INFO] Finished at: 2025-09-22T12:08:51+05:30
[INFO] -----
Archiving artifacts
Finished: 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"

Build Queue

Jenkins / All / New Item

New Item

Enter an item name

Lakshmitha_Jenkins_MavenWeb_Build

Select an item type

- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

└─ Configure the project:

 └─ Description: "Web Build demo"

 └─ Source Code Management:

 └─ Git repository URL: [GitMavenWeb repo URL]

 └─ Branches to build: */Main or master

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

None

Git ?

Repositories ?

Repository URL ?

https://github.com/LakshmithaReddy1807/Lakshmitha-Maven-Java-WebProject.git

Credentials ?

LakshmithaReddy1807/*****

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Save Apply

└─ 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

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

The screenshot shows a build configuration interface with two build steps. Each step is a card with a header, input fields for Maven Version and Goals, and an 'Advanced' dropdown. The first step's Goals field contains 'clean'. The second step's Goals field contains 'install'. Both steps use 'MAVEN_HOME' as the Maven Version. At the bottom, there are 'Save' and 'Apply' buttons.

 ≡ **Invoke top-level Maven targets** ? ×

Maven Version
MAVEN_HOME

Goals
clean

Advanced ▾

 ≡ **Invoke top-level Maven targets** ? ×

Maven Version
MAVEN_HOME

Goals
install

Advanced ▾

Add build step ▾

Save Apply

└─ 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

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

The screenshot shows the Jenkins post-build actions configuration interface. It includes sections for 'Archive the artifacts' (with a pattern of '**/*'), 'Build other projects' (with a project named 'Lakshmitha_Jenkins_MavenWeb_Test' and a note about a misspelling), and an 'Add post-build action' dropdown. At the bottom are 'Save' and 'Apply' buttons.

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

- ├─ Enter project name (e.g., MavenWeb_Test)
- ├─ Click "OK"

New Item

The screenshot shows the Jenkins 'New Item' dialog. It has fields for 'Enter an item name' (containing 'Lakshmitha_Jenkins_MavenWeb_Test') and 'Select an item type'. A list of project types is shown:

- Freestyle project: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder: Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline: Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder

An 'OK' button is at the bottom.

└─ 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: **/*

Environment
Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

Delete workspace before build starts
 Advanced

Use secret text(s) or file(s)
 Add timestamps to the Console Output
 Inspect build log for published build scans
 Terminate a build if it's stuck
 With Ant

Build Steps
Automate your build process with ordered tasks like code compilation, testing, and deployment.

Copy artifacts from another project

Project name: Lakshmitha_Jenkins_MavenWeb_Build

Which build: Latest successful build
 Stable build only

Artifacts to copy: **/*

Artifacts not to copy:

Save **Apply**

 └─ 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: **/*

The screenshot shows the Jenkins configuration interface for a job. At the top, there's a section for "Invoke top-level Maven targets" with fields for "MAVEN_HOME" and "Goals" set to "test". Below this is a "Post-build Actions" section titled "Archive the artifacts" with a "Files to archive" field containing "**/*".

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

 └─ Projects to build: MavenWeb_Deploy

 └─ Apply and Save

The screenshot shows the "Build other projects" configuration. It lists "Projects to build" as "Lakshmitha_Jenkins_MavenWeb_Deploy". A warning message says "No such project 'Lakshmitha_Jenkins_MavenWeb_Deploy'. Did you mean 'Lakshmitha_Jenkins_MavenWeb_Build'?". There are three trigger options: "Trigger only if build is stable" (selected), "Trigger even if the build is unstable", and "Trigger even if the build fails". Below are "Add post-build action" and "Save/Apply" buttons.

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

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

 └─ Click "OK"

New Item

The screenshot shows the "New Item" dialog. In the "Enter an item name" field, "Lakshmitha_Jenkins_MavenWeb_Deploy" is typed. Under "Select an item type", "Freestyle project" is selected, described as a "Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications." Other options shown include "Maven project", "Pipeline", "Multi-configuration project", "Folder", "Multibranch Pipeline", and "Organization Folder". At the bottom, there's an "OK" button.

└─ Configure the project:

└─ **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: **/*

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

Delete workspace before build starts

Advanced

Use secret text(s) or file(s) ?

Add timestamps to the Console Output

Inspect build log for published build scans

Terminate a build if it's stuck

With Ant ?

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Copy artifacts from another project

 Project name ?

 Which build ?

Stable build only

 Artifacts to copy ?

 Save Apply

└─ **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"

Jenkins / New view

+ New Item

Build History

New view

Name: Lakshmitha_Jenkins_MavenWeb_pipeline

Type: Build Pipeline View

Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

List View

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View

This view automatically displays all the jobs that the current user has an access to.

Create

- └─ **Pipeline Flow:**
- └─ **Layout:** Based on upstream/downstream relationship
 - └─ Initial job: MavenWeb_Build
 - └─ Apply and Save

Edit View

Name: Lakshmitha_Jenkins_MavenWeb_pipeline

Description: Describe the purpose of this view.

Build Pipeline View Title

Pipeline Flow

Layout: Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger extension.

Upstream / downstream config

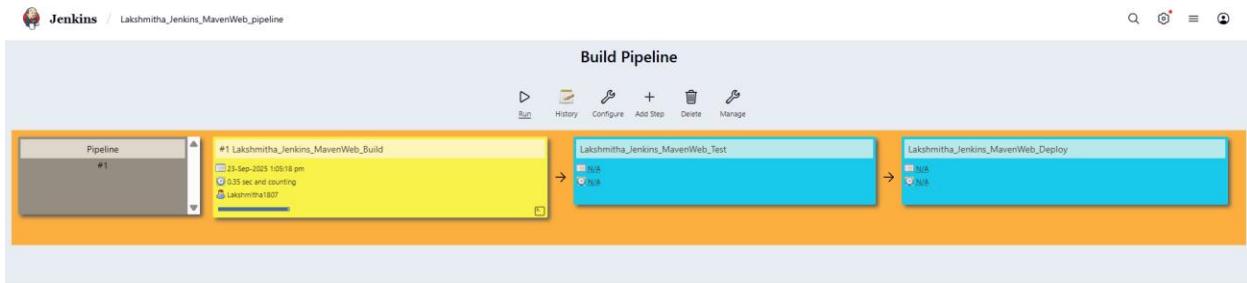
Select Initial Job: ?

Lakshmitha_Jenkins_MavenWeb_Build

Save Apply

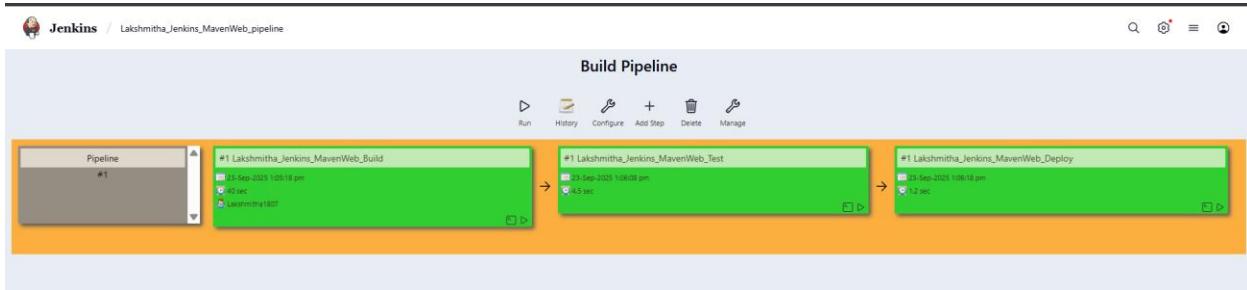
└─ **Step 6:** Run the Pipeline and Check Output

—— 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



o

—— 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.

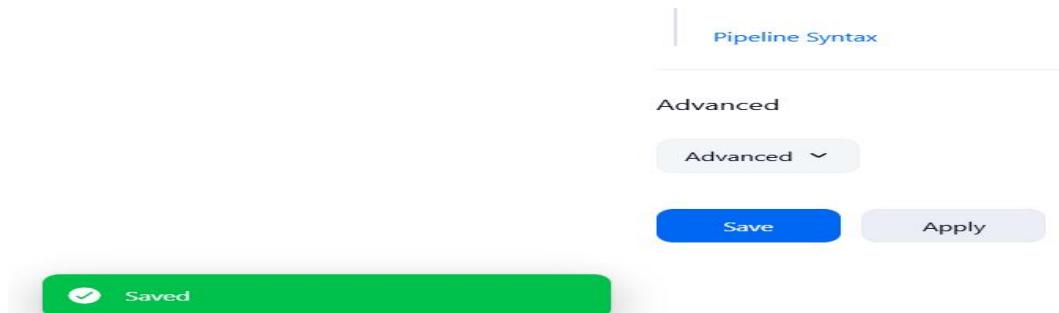
The screenshot shows the Jenkins 'New Item' creation dialog. On the left, there's a sidebar with 'Jenkins' logo, 'New Item' button, and 'Build History'. The main area has a search bar 'Enter an item name' with 'Lakshmitha_Jenkins_Java_using_script' typed in. Below it, 'Select an item type' dropdown is open, showing several options: 'Freestyle project' (selected), 'Maven project', 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. Each option has a brief description. At the bottom right of the dialog is an 'OK' button.

The screenshot shows the Jenkins Pipeline configuration dialog for the item 'Harshitha_Jenkins_Java_using_Script'. The left sidebar has tabs for 'General', 'Triggers', 'Pipeline', and 'Advanced', with 'General' selected. Under 'Definition', 'Pipeline script' is chosen. The main area shows a 'General' tab with a 'Description' field containing 'Creating java project pipeline using Script'. Below it is a 'Script' code editor with the following Groovy code:

```
1~ pipeline {
2    agent any
3    tools{
4        maven 'MAVEN-HOME'
5    }
6    stages {
7        stage('git repo & clean') {
8            steps {
9                //bat "rmdir /s /q mavenjava"
10               bat "git clone https://github.com/Harshitha-Macha/Harshitha_Jenkins_Maven.git"
11               bat "mvn clean -f Harshitha_Jenkins_Maven"
12            }
13        }
14        stage('install') {
15            steps {

```

At the bottom left of the dialog is a checked checkbox 'Use Groovy Sandbox'.



Procedure

General :

Description :- provide the description of the project

Build triggers: here we can provide with build triggers of your 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

figure

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

General

Triggers

Definition

Pipeline script

Advanced

Script ?

```
1 v pipeline {
2     agent any
3 v     tools{
4         jdk 'JAVA_HOME'
5         maven 'MAVEN_HOME'
6     }
7 v     stages {
8 v         stage('Clean Workspace') {
9 v             steps {
10                 deleteDir()
11             }
12         }
13 v         stage('git repo & clean') {
14 v             steps {
15                 bat "rmdir /s /q Marshitha_Jenkins_Maven || exit 0"
16             }
17         }
18     }
19 }
```

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
15      bat "rmdir /s /q Harshitha_Jenkins_Maven || exit 0"
16      bat "git clone https://github.com/Harshitha-Macha/Harshitha_Jenkins_Maven.git"
17      bat "mvn clean -f Harshitha_Jenkins_Maven/pom.xml"
18    }
19  }
20  stage('install') {
21    steps {
22      bat "mvn install -f Harshitha_Jenkins_Maven/pom.xml"
23    }
24  }
25  stage('test') {
26    steps {
27      bat "mvn test -f Harshitha_Jenkins_Maven/pom.xml"
28    }
29  }
```

 #6 (Oct 7, 2025, 10:47:07 AM)

 Started by user Macha Harshitha

ion

 This run spent:

- 11 ms waiting;
- 31 sec build duration;
- 31 sec total from scheduled to completion.

 No changes.

```
[INFO] Running KMIT.Harshitha_Jenkins_Maven.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ Harshitha_Jenkins_Maven ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  3.485 s
[INFO] Finished at: 2025-10-07T10:47:38+05:30
[INFO] -----
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

b. WORKING ON BUILD TRIGGERS FOR LAST JENKINS PIPILINE



+ New Item

 Build History

Build Queue

No builds in the queue.

New Item

Enter an item name

Lakshmitha_Jenkins_Java_using_script

Select an item type



Freestyle project

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



Maven project

Build a maven project. Jenkins takes advantage of your POM files



Pipeline

Orchestrates long-running activities that can span multiple build workflows) and/or organizing complex activities that do not easily fit into a single build step.



Multi-configuration project

Suitable for projects that need a large number of different config platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for group folder creates a separate namespace, so you can have multiple them folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches



Organization Folder

Creates a set of multibranch project subfolders by scanning for re

If you want to create a new item from other existing, you can use this option:

OK

Procedure

General :

Description :- provide the description of the project

General

Enabled

Description

This project demonstrates a Jenkins pipeline created using a scripted Jenkinsfile for building, testing, and packaging a Maven project from a GitHub repository.

Plain text [Preview](#)

Discard old builds ?

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Permission to Copy Artifact

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

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built ?

Build periodically ?

Schedule ?

H * * * *

Would last have run at Tuesday, 7 October, 2025 at 10:13:00 am India Standard Time; would next run at Tuesday, 7 October, 2025 at 10:13:00 am India Standard Time.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)

HOUR The hour of the day (0–23)

DOM The day of the month (1–31)

MONTH The month (1–12)

DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of increasing specificity:

- * specifies all valid values
- M-N specifies a range of values
- M-N/X or */X steps by intervals of X through the specified range or whole valid range
- A,B,...,Z enumerates multiple values

To allow periodically scheduled tasks to produce even load on the system, the symbol H (for hour) will cause a large spike at midnight. In contrast, using H H * * * would still execute each job once per hour.

The H symbol can be used with a range. For example, H H(0-7) * * * means some time between 0 and 7 hours past the hour.

The H symbol can be thought of as a random value over a range, but it actually is a hash of the current time.

Builds

Filter

Today

#20 2:16 PM

#19 2:15 PM

#18 2:14 PM

#17 2:13 PM

#16 2:12 PM

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"
        }
      }
    }
  }
```

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script

Script ?

```
1~ pipeline {
2~   agent any
3~   tools {
4~     maven 'MAVEN_HOME'
5~   }
6~   stages {
7~     stage('Clean Workspace') {
8~       steps {
9~         deleteDir()
10~      }
11~    }
12~    stage('Git Repo & Clean') {
13~      steps {
14~        bat "rmdir /s /q Lakshmitha-Maven-JavaProject || exit 0"
15~        bat "git clone https://github.com/LakshmithaReddy1807/Lakshmitha-Maven-JavaProject.git"
16~      }
17~    }
18~  }
19~}
```

Use Groovy Sandbox

Pipeline Syntax

Advanced

Advanced ▾

Save

Apply

```

11      }
12  v     stage('Git Repo & Clean') {
13  v       steps {
14         bat "rmdir /s /q Lakshmitha-Maven-JavaProject || exit 0"
15         bat "git clone https://github.com/LakshmithaReddy1807/Lakshmitha-Maven-JavaProject.git"
16         bat "mvn clean -f Lakshmitha-Maven-JavaProject"
17     }
18   }
19
20 v   stage('Install') {
21 v     steps {
22       bat "mvn install -f Lakshmitha-Maven-JavaProject"
23     }
24   }
25
26 v   stage('Test') {
27 v     steps {
28       bat "mvn test -f Lakshmitha-Maven-JavaProject"
29     }
30   }
31
32 v   stage('Package') {
33 v     steps {
34       bat "mvn package -f Lakshmitha-Maven-JavaProject"
35     }
36   }
37 }
38 }
```

Apply and save

Console Output

```

Started by user Bade Lakshmitha Reddy
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\Lakshmitha_Jenkins_Jav
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
```

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. Why is tools { maven 'MAVEN-HOME' } used in this pipeline?

- A) It tells Jenkins to automatically install and use the configured Maven tool from Manage Jenkins → Global Tool Configuration, ensuring Maven is available for build commands.
12. 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"
}
```
13. Which Jenkins plugin is needed to recognize the pipeline { ... } structure in this script?
- A) The Pipeline Plugin (also called Workflow Plugin) must be installed.
14. 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.
15. How will you modify the pipeline to stop execution if the GitHub clone command fails?  
 Your project lead asks you to print a welcome message in Jenkins to confirm the pipeline is working — how would you script it?
- A) Add a condition or use returnStatus:
- ```
script {
  def status = bat(script: "git clone https://github.com/...git", returnStatus: true)
  if (status != 0) {
    error("Git clone failed — stopping pipeline.")
  }
}
```
16. A teammate forgot to pull the latest GitHub code before building; how can you fix this in your pipeline?
- A) Add a simple echo step at the beginning:
- ```
stage('Welcome') {
 steps {
 echo "Welcome to Jenkins Pipeline — Build Started Successfully!"
 }
}
```

17. 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.

18. Tests should stop the pipeline if they fail — where will you put the mvn test command?

A) Add a Compile stage after cloning:

```
stage('Compile') {
 steps {
 bat "mvn compile -f project/pom.xml"
 }
}
```

19. 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 \* \* \*

20. 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.

21. 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

22. 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"
}
```

23. 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

- Create an application in Kubernetes:
  - Command:

```
kubectl create deployment mynginx --image=nginx
if already created then
kubectl set image deployment/mynginx 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: node.kubernetes.io/unreachable:op-exists-for-300s
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 including
159974475 bytes.
Normal Created 90s kubelet Created container: nginx
Normal Started 90s kubelet Started container nginx
PS C:\WINDOWS\system32>
```

## 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
```

```
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. 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
```

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

- Access the app via <http://localhost:8081>.

### c. 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
```

```
C:\Users\hariv>docker pull jasonrivers/nagios:latest
latest: Pulling from jasonrivers/nagios
785b9873bdf4: Pulling fs layer
71bfb306f8cb: Pulling fs layer
785b9873bdf4: Pull complete
71bfb306f8cb: Pull complete
0ef9446ba5cc: Pull complete
b69c76bd2b6b: Pull complete
d5aa2a3a6539: Pull complete
738fc7520889: Pull complete
9ffe54c5c139: Pull complete
279b28aeafa10: Pull complete
c219d58cc3f9: Pull complete
8e911c59da28: Pull complete
e58e184b986a: Pull complete
d3245570f968: Pull complete
4f4fb700ef54: Pull complete
b0e280e9aa8c: Pull complete
8c389e58e867: Pull complete
15f36d0b0439: Pull complete
eeb77e6dde3e: Pull complete
e6f8fab512d1: Pull complete
0bd0f5795eeb: Pull complete
ff65ddf9395b: Pull complete
a2fc4187e3b4: Pull complete
fe8a6b2cf4e3: Pull complete
706ed7d4ce0a: Pull complete
3d5785144815: Pull complete
53aff88babc4: Pull complete
566cdc02555d: Pull complete
d72f92e29533: Pull complete
c700be87d617: Pull complete
a900dfcceeb38: Pull complete
8fb30af17153: Pull complete
9a90645e352c: Pull complete
Digest: sha256:2a7c2b20d118baf92b47b69a3901e68dd7664617801b94e560bc4d6564d6ae54
Status: Downloaded newer image for jasonrivers/nagios:latest
docker.io/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.

The screenshot shows the Nagios Core 4.5.7 Host Information page for the host **localhost**. The top navigation bar includes links for View Status Detail For This Host, View Alert History For This Host, View Trends For This Host, View Alert Histogram For This Host, View Availability Report For This Host, and View Notifications For This Host. The main content area is divided into several sections:

- Host Information**: Last Updated: Tue Oct 14 15:36:01 UTC 2025, Updated every 90 seconds, Nagios® Core™ 4.5.7 - www.nagios.org, Logged in as nagiosadmin.
- Host**: **localhost** ([localhost](#))
- Member of**: [linux-servers](#)
- Address**: 127.0.0.1
- Host State Information**: Status: **UP** (for 0d 0h 6m 43s). Status Information: PING OK - Packet loss = 0%, RTA = 0.05 ms. Performance Data: rta=0.052000ms;3000.000000;5000.000000;0.000000 pl=0%;80;100;0. Current Attempt: 1/10 (HARD state). Last Check Time: 10-14-2025 15:31:24. Check Type: ACTIVE. Check Latency / Duration: 0.000 / 4375 seconds. Next Scheduled Active Check: 10-14-2025 15:36:24. Last State Change: 10-14-2025 15:29:18. Last Notification: N/A (notification 0). Is This Host Flapping?: **NO** (0.00% state change). In Scheduled Downtime?: **NO**. Last Update: 10-14-2025 15:35:56 (0d 0h 0m 5s ago).
- Host Commands**: A list of actions including Locate host on map, Disable active checks of this host, Re-schedule the next check of this host, Submit passive check result for this host, Stop accepting passive checks for this host, Stop obsessing over this host, Disable notifications for this host, Send custom host notification, Schedule downtime for this host, Schedule downtime for all services on this host, Disable notifications for all services on this host, Enable notifications for all services on this host, Schedule a check of all services on this host, Disable checks of all services on this host, Enable checks of all services on this host, Disable event handler for this host, Disable flap detection for this host, and Clear flapping state for this host.

## 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:

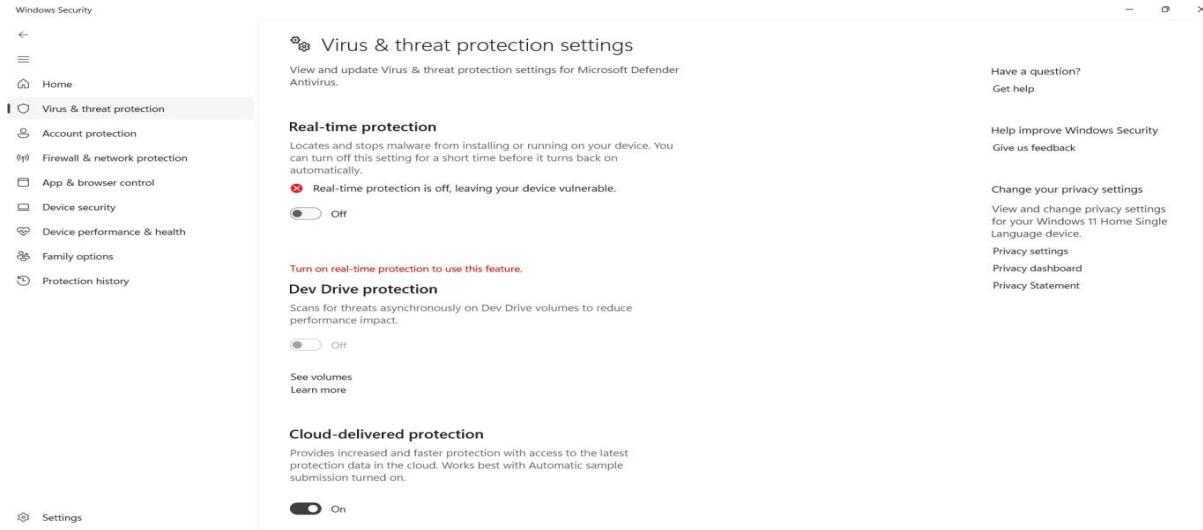
```
docker rmijasonrivers/nagios:latest
```

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.

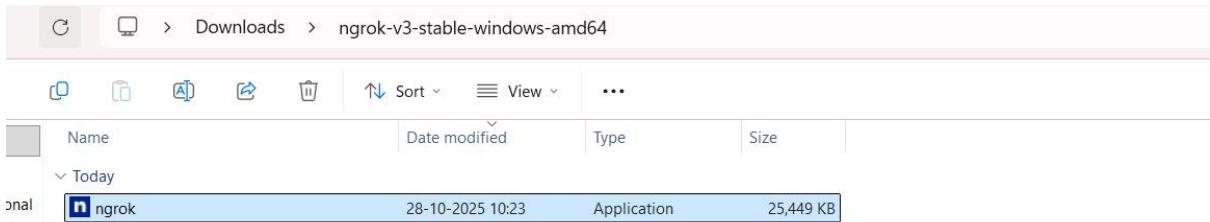
Step-1: To install ngroks Go->settings->privacy and security -> windows security -> off antivirus ->



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

A screenshot of the ngrok website. At the top, there's a navigation bar with links for Platform, Use cases, Blog, Resources, Docs, Pricing, Get ngrok, and a sign-in button. The main content area has a purple header 'Download ngrok' with the tagline 'ngrok is your app's front door—and the fastest way to put anything on the internet.' Below this, there's a 'Windows' section with a 'Setup &amp; Installation' link. A sidebar on the left shows sections like Getting Started, Universal Gateway, Traffic Observability, and Billing &amp; Usage. The main content area also includes a 'Deploy your app online' section with a command-line input field containing 'ngrok config add-authToken 34ejJvRYKqtjvBUEYEnxudIbpb7\_33i6pYx8BJ2cZ1TpwasWq'.

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



Ngrok command prompt appears as below

```
C:\Users\sample\Downloads> +
tcp start a TCP tunnel
tls start a TLS endpoint
update update ngrok to the latest version
version print the version string

EXAMPLES:
forward http traffic from assigned public URL to local port 80
ngrok http 80
port 8080 available at baz.ngrok.dev
ngrok http --url baz.ngrok.dev 8080
tunnel arbitrary TCP traffic to port 22
ngrok tcp 22
secure your app with oauth
ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com

Paid Features:
ngrok http 80 --url mydomain.com # run ngrok with your own custom domain
ngrok http 80 --cidr-allow 2600:8c00::a03c:fe69:9695/32 # run ngrok with IP policy restrictions
Upgrade your account at https://dashboard.ngrok.com/billing/choose-a-plan to access paid features

Upgrade your account at https://dashboard.ngrok.com/billing/choose-a-plan to access paid features

Flags:
-h, --help help for ngrok

Use "ngrok [command] --help" for more information about a command.

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\sample\Downloads\ngrok-v3-stable-windows-amd64>
```

#### 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

VS Vaishnavi Sampangi

Getting Started

Your Authtoken

Universal Gateway

Add your Authtoken to the ngrok agent using any of these methods:

Service Users

Command Line

Configuration File

Help

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

```
ngrok config add-authtoken 34ejJvRYKqtjvBUEYEnxudIpb7_33i6pYx8BJ2cZ1TpwasWq
```

```
C:\Users\sample\Downloads\ngrok-v3-stable-windows-amd64>ngrok config add-authtoken 34ejJvRYKqtjvBUEYEnxudIpb7_33i6pYx8BJ2cZ1TpwasWq
Authtoken saved to configuration file: C:\Users\sample\AppData\Local/ngrok/ngrok.yml

C:\Users\sample\Downloads\ngrok-v3-stable-windows-amd64>
```

Step-6

Start a Tunnel for Jenkins

- Check on which port is your Jenkins running . for this give in browser or url localhost:8081  
For me Jenkins is running on 8081
  - Go to ngrok command prompt and type below command
  - ngrok http 8081 //Always use this command to start a tunnel for jenkins .

Type in ngrok command prompt:

```
at this terminal prompt to expose port 80.
C:\Users\sample\Downloads\ngrok-v3-stable-windows-amd64>ngrok http 8081
```

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

```
C:\Users\sample\Downloads\ngn > + v
ngrok - tunnel local ports to public URLs and inspect traffic

USAGE:
 ngrok [command] [flags]

COMMANDS:
 api CLI to api.ngrok.com
 completion generates shell completion code for bash or zsh
 config update or migrate ngrok's configuration file
 credits prints author and licensing information
 help help about any command
 http start an HTTP tunnel
 service run and control ngrok as a background service
 start start endpoints in the config file by name
 tcp start a TCP tunnel
 tls start a TLS endpoint
 update update ngrok to the latest version
 version print the version string

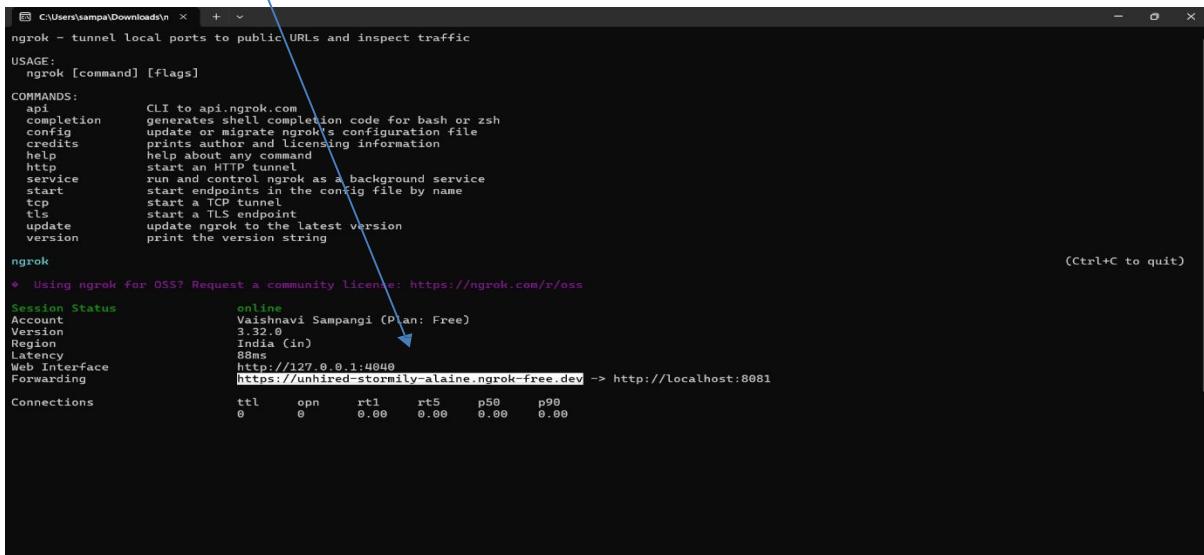
ngrok

* Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss

Session Status online
Account Vaishnavi Sampangi (Plan: Free)
Version 3.32.0
Region India (in)
Latency 63ms
Web Interface http://127.0.0.1:4040
Forwarding https://unhired-stormily-alaine.ngrok-free.dev -> http://localhost:8081

Connections ttl opn rt1 rt5 p50 p90
 0 0 0.00 0.00 0.00 0.00
```

Copy this URL only highlighted part



```
C:\Users\sample\Downloads\ngrok - tunnel local ports to public URLs and inspect traffic
USAGE:
 ngrok [command] [flags]

COMMANDS:
 api CLI to api.ngrok.com
 completion generates shell completion code for bash or zsh
 config update or migrate ngrok's configuration file
 credits prints author and licensing information
 help help about any command
 http start an HTTP tunnel
 service run ngrok as a background service
 start start endpoints in the config file by name
 tcp start a TCP tunnel
 tls start a TLS endpoint
 update update ngrok to the latest version
 version print the version string

 (Ctrl+C to quit)

Session Status
 Account: Vaishnavi Sampangi (Plan: Free)
 Version: 3.32.0
 Region: India (in)
 Latency: 88ms
 Web Interface: http://127.0.0.1:4040
 Forwarding: https://unhired-stormily-alaine.ngrok-free.dev -> http://localhost:8081

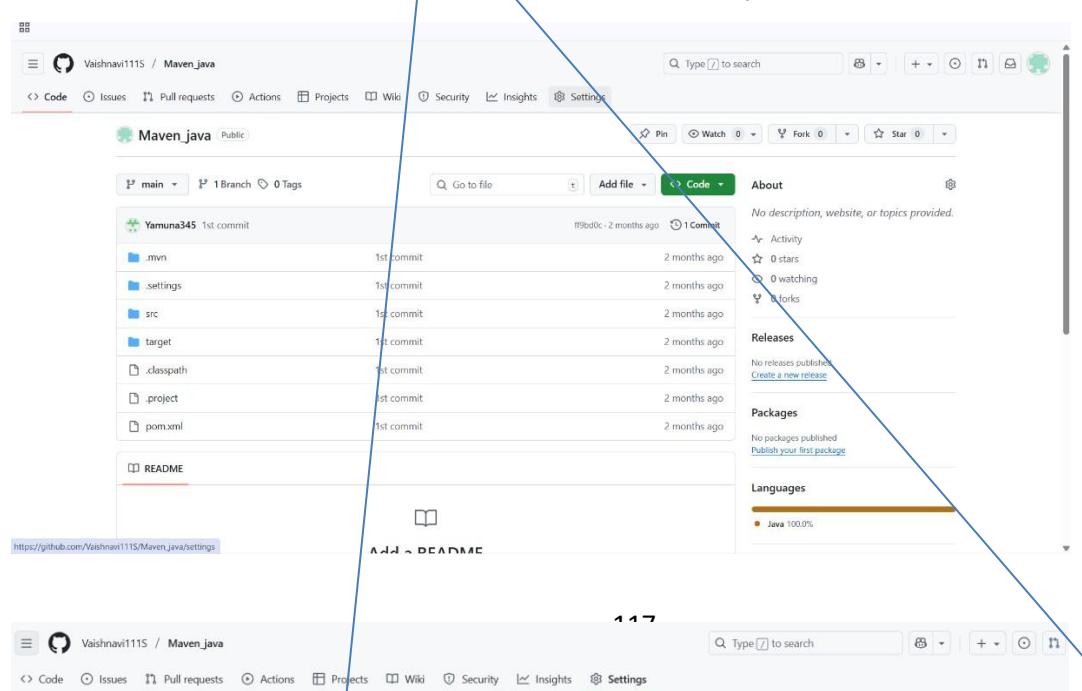
Connections ttl opn rt1 rt5 p50 p90
 0 0 0.00 0.00 0.00 0.00 0.00
```

### 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.

The screenshot shows the GitHub settings interface for a repository named 'Maven\_java'. The left sidebar has 'Webhooks' selected under the 'General' section. The right pane shows the 'Webhooks / Add webhook' configuration page. Arrows from the list items point to the corresponding fields: the URL field contains 'https://unhired-stormily-alaine.ngrok-free.dev/github-webhook/' (with a red arrow), the Content type dropdown is set to 'application/json' (with a blue arrow), and the 'Just the push event.' radio button is selected under 'Which events would you like to trigger this webhook?' (with a green arrow). A large red arrow points to the 'Add webhook' button at the bottom right of the form.

## Step 10: Configure Jenkins to Accept GitHub Webhooks

1. Open Jenkins Dashboard.
2. Select the job (freestyle or pipeline) you've already created.

The screenshot shows the Jenkins dashboard with several jobs listed:

| S | W  | Name             | Last Success    | Last Failure    | Last Duration | F   |
|---|----|------------------|-----------------|-----------------|---------------|-----|
| ✓ | ☀️ | JS1              | 20 days #11     | N/A             | 28 sec        | ▶ ⭐ |
| ✓ | ☁️ | Maven_java_build | 1 mo 2 days #12 | 1 mo 2 days #10 | 7.1 sec       | ▶ ⭐ |
| ✓ | 🌧️ | Maven_java_test  | 1 mo 2 days #13 | 1 mo 2 days #11 | 3.5 sec       | ▶ ⭐ |
| ✓ | ☁️ | Maven_web_build  | 1 mo 2 days #5  | 1 mo 2 days #3  | 25 sec        | ▶ ⭐ |
| ✓ | ☁️ | Maven_web_deploy | 1 mo 2 days #12 | 1 mo 2 days #9  | 1.4 sec       | ▶ ⭐ |
| ✓ | ☀️ | Maven_web_test   | 1 mo 2 days #8  | N/A             | 5.6 sec       | ▶ ⭐ |

3. Click Configure.
4. Scroll down to the Build Triggers section.
5. Check the box:  GitHub hook trigger for GITScm polling

The screenshot shows the configuration page for the 'Maven\_java\_build' job. The 'Triggers' section is active, displaying the following options:

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

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.

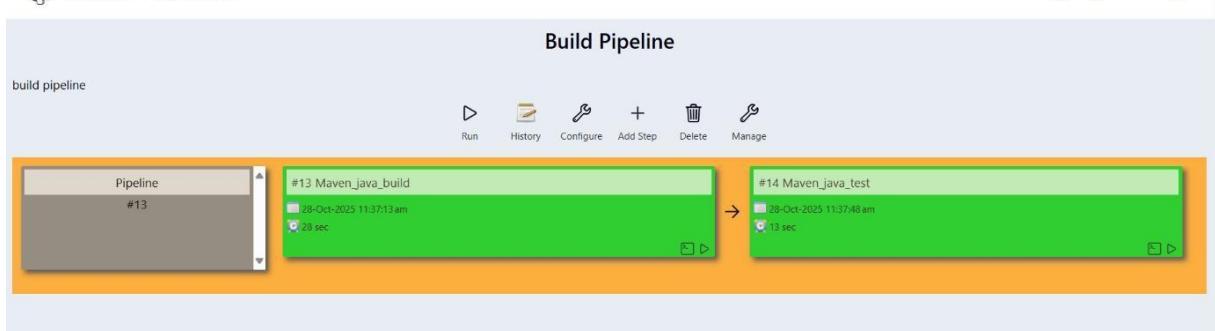
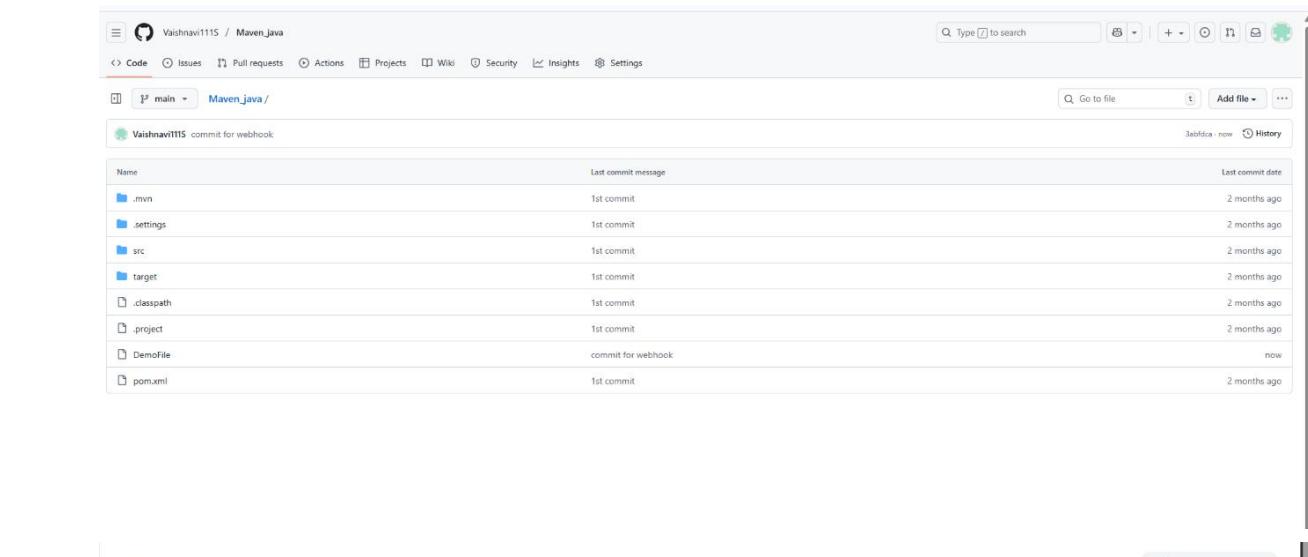
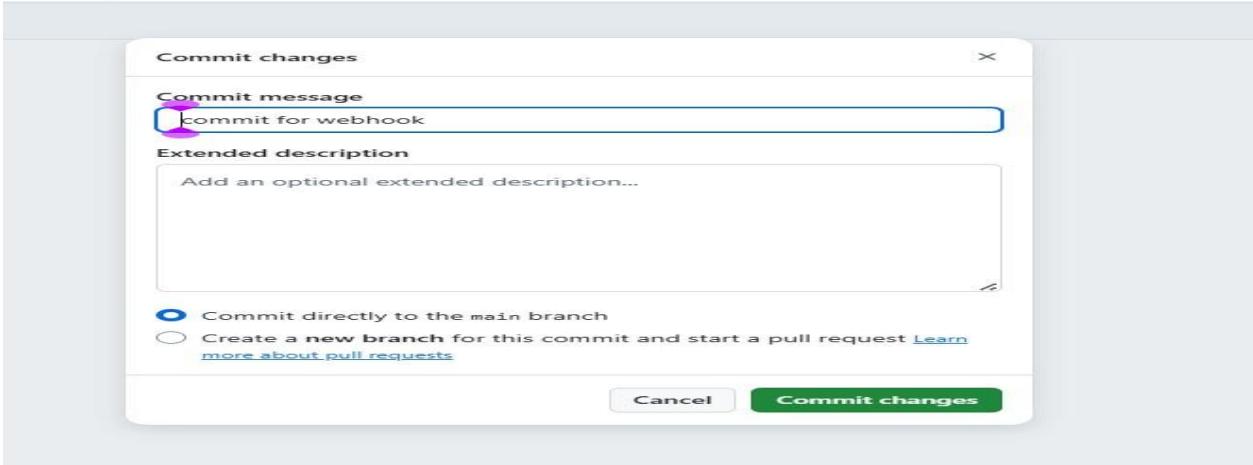
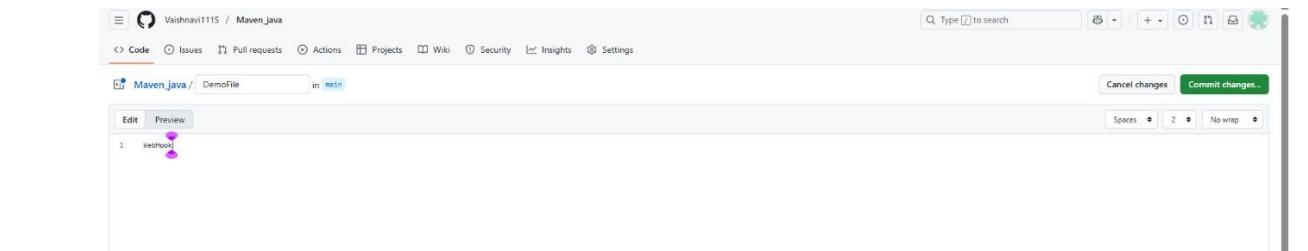
The screenshot shows the Jenkins configuration interface for a project named "Maven\_java\_build". The left sidebar has tabs for General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The "Triggers" tab is selected. Under "Triggers", there are several options: "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", "Build periodically", "GitHub hook trigger for GITScm polling" (which is checked), and "Poll SCM". Below the triggers is the "Environment" section, which contains settings for workspace deletion, secret text usage, timestamps, build log inspection, stuck build termination, and Ant usage. At the bottom are "Save" and "Apply" buttons.

The screenshot shows the Jenkins dashboard. On the left, there are two collapsed sections: "Build Queue" (No builds in the queue) and "Build Executor Status" (0/2). The main area displays a table of active builds:

| S | W | Name ↓           | Last Success    | Last Failure    | Last Duration | F |
|---|---|------------------|-----------------|-----------------|---------------|---|
|   |   | JS1              | 20 days #11     | N/A             | 28 sec        |   |
|   |   | Maven_java_build | 1 mo 2 days #12 | 1 mo 2 days #10 | 7.1 sec       |   |
|   |   | Maven_java_test  | 1 mo 2 days #13 | 1 mo 2 days #11 | 3.5 sec       |   |
|   |   | Maven_web_build  | 1 mo 2 days #5  | 1 mo 2 days #3  | 25 sec        |   |
|   |   | Maven_web_deploy | 1 mo 2 days #12 | 1 mo 2 days #9  | 1.4 sec       |   |
|   |   | Maven_web_test   | 1 mo 2 days #8  | N/A             | 5.6 sec       |   |

At the bottom, there are icons for S, M, L, and a search bar with placeholder text "Type / to search".

The screenshot shows a GitHub repository named "Maven\_java". The "Code" tab is selected. A file named "demo of webhook" is shown with its content: "1 demo of webhook". There are "Edit" and "Preview" buttons at the top of the code editor. At the bottom right, there are "Cancel changes" and "Commit changes..." buttons, along with a "Spaces" dropdown set to "2" and "No wrap" option.



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

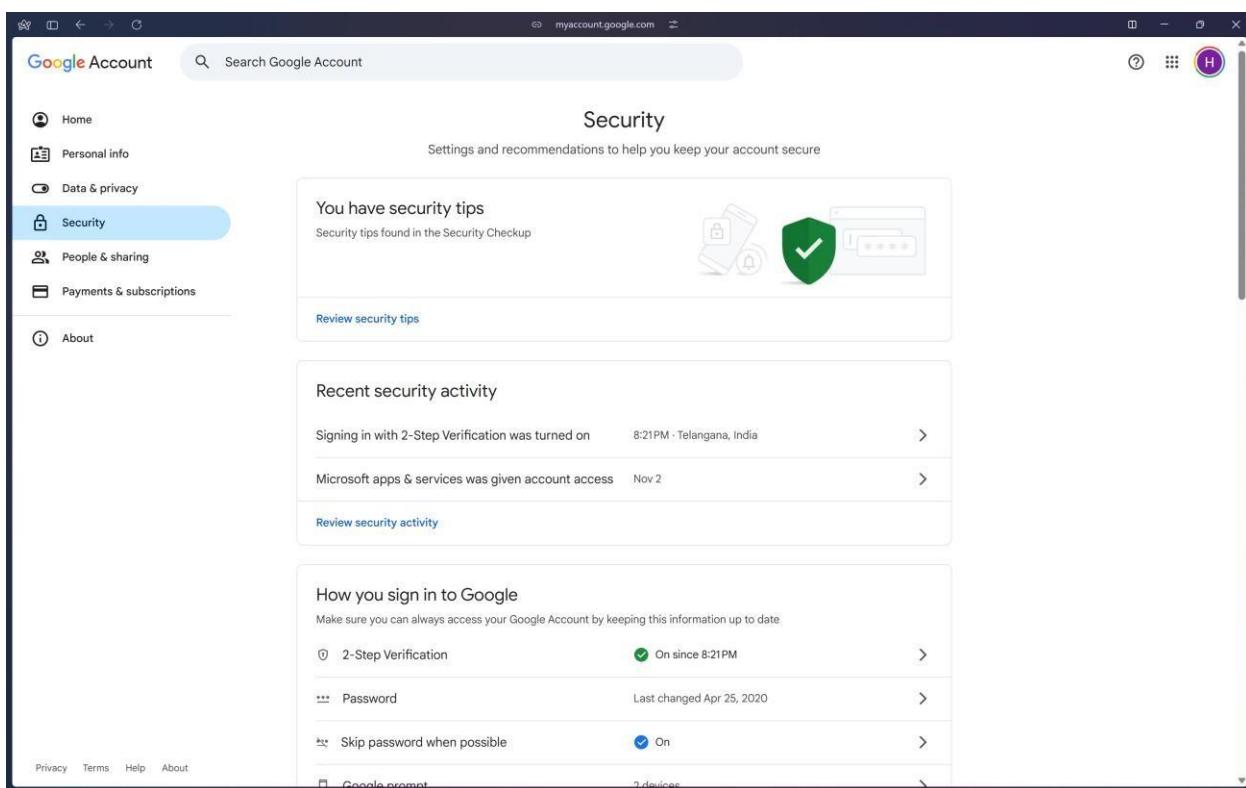
Creation of app password

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

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

ii. Enable 2-Step Verification

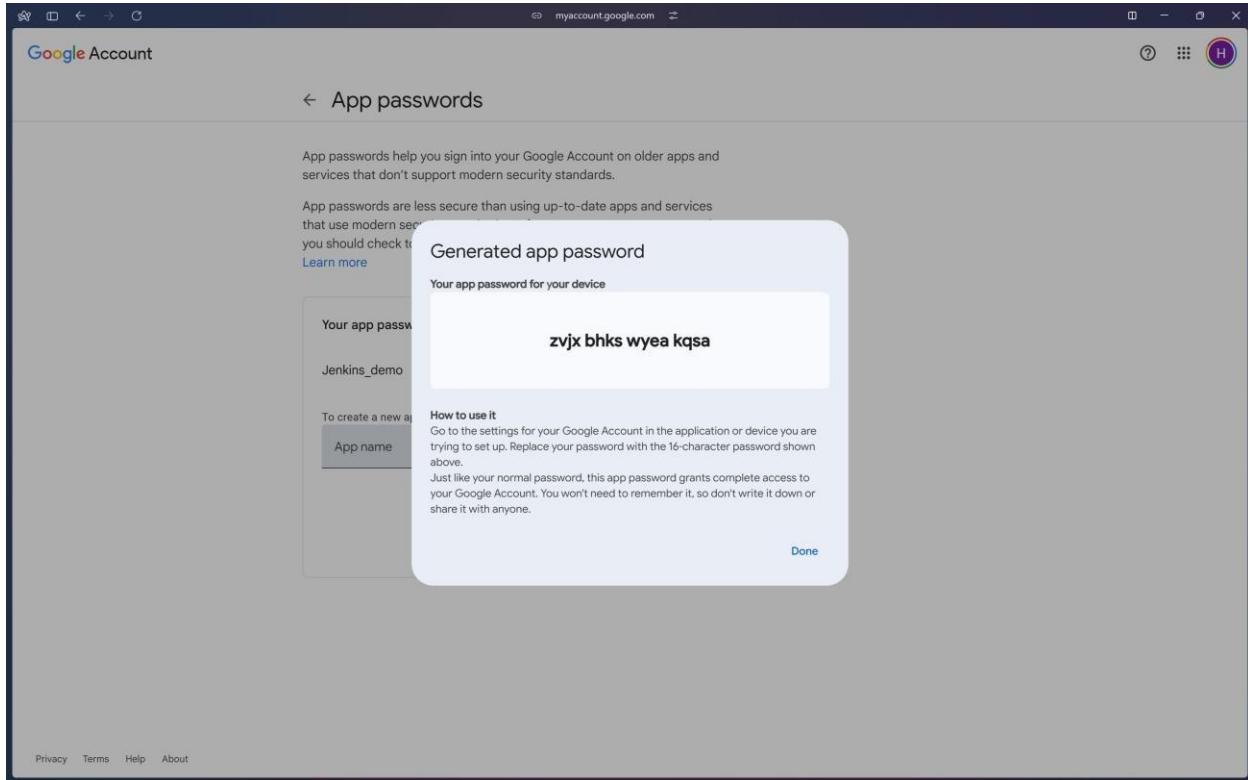
- Navigate to:
  - Security → 2-Step Verification
  - Turn it ON
  - Complete the OTP verification process (via phone/email)



### 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

The screenshot shows the Jenkins Manage Plugins page. The search bar at the top contains the text "email". The results table lists two plugins:

| Name                                      | Health            | Enabled              |
|-------------------------------------------|-------------------|----------------------|
| Email Extension Plugin 1933.v45cec755423f | <span>100%</span> | <span>Enabled</span> |
| Mailer Plugin 522.va_995fa_cfb_8b_d       | <span>100%</span> | <span>Enabled</span> |

Both plugins have green status indicators and blue "Enabled" buttons. The "Email Extension Plugin" row includes a link to its GitHub repository: "Report an issue with this 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 | <input checked="" type="checkbox"/> Enabled      |
| User Name     | Your Gmail ID (e.g., archanareddykmit@gmail.com) |
| Password      | Paste the 16-digit App Password                  |
| Use SSL       | <input checked="" type="checkbox"/> Enabled      |
| SMTP Port     | 465                                              |

The screenshot shows the Jenkins configuration interface for the 'Email' section. It includes fields for User Name, Password, Use SSL, SMTP Port, Reply-To Address, Charset, and a checkbox for testing by sending a test email.

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              | <input checked="" type="checkbox"/> 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

Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

Ex : Launch ubuntu instance

Step 1: Login to AWS /canvas account

The screenshot shows the AWS Academy Learner Lab interface. On the left is a sidebar with options like Home, Modules, Discussions, Grades, and Lucid (Whiteboard). The main area has a terminal window showing a session on a Ubuntu instance. To the right is a sidebar titled 'Learner Lab' containing links to various AWS services and best practices. A timer at the top indicates 02:19 and a usage bar shows 'Used \$0.1 of \$50'. Navigation buttons for 'Previous' and 'Next' are at the bottom.

Step 2: Services -- EC2

The screenshot shows the AWS EC2 service dashboard. The left sidebar includes sections for Dashboard, Instances, Images, Elastic Block Store, Network & Security, and more. The main area displays 'Resources' for the United States (N. Virginia) Region, showing counts for Instances (running), Auto Scaling Groups, Capacity Reservations, Dedicated Hosts, Elastic IPs, Instances, Key pairs, Load balancers, Placement groups, Security groups, Snapshots, and Volumes. It also features a 'Launch instance' button and a 'Service health' card. Other cards include 'Explore AWS' (mentioning Best Price-Performance with AWS Graviton2), 'Additional information' (Get started walkthroughs), and 'Account attributes' (Default VPC, Settings, Data protection and security, Allowed AMIs, Zones, EC2 Serial Console, Default credit specification, EC2 console preferences).

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

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, the 'Name' field is filled with 'ubuntu'. In the 'Software Image (AMI)' section, 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' is selected, noted as 'Free tier eligible'. The 'Virtual server type (instance type)' is set to 't3.micro'. The 'Launch instance' button is highlighted in orange.

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 download with extension .pem  
Store key in folder AWS

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Instance type' step, 't2.micro' is selected. In the 'Key pair (login)' step, a new key pair named 'ubuntu' is being created. The 'Network settings' step is partially visible. The 'Launch instance' button is highlighted in orange.

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

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Network settings' section, under 'Subnet', it says 'No preference (Default subnet in any availability zone)'. Under 'Firewall (security groups)', there are three checkboxes: 'Allow SSH traffic from anywhere', 'Allow HTTPS traffic from the internet', and 'Allow HTTP traffic from the internet'. The 'Allow SSH traffic from anywhere' checkbox is selected. In the 'Configure storage' section, it shows 1x 8 GiB gp3 Root volume, 3000 IOPS, Not encrypted. The 'Advanced' button is visible. On the right side, the 'Summary' section shows 'Number of instances' set to 1, 'Software Image (AMI)' as Canonical, Ubuntu, 24.04, amd64, and 'Virtual server type (instance type)' as t3.micro. The 'Launch instance' button is highlighted.

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.

The screenshot shows the 'Connect' tab for the instance. It includes instructions for opening an SSH client, locating the private key file (ubuntu.pem), running chmod 400 on it, and connecting to the Public DNS (ec2-3-85-222-109.compute-1.amazonaws.com). An example command is provided: ssh -i "ubuntu.pem" ubuntu@ec2-3-85-222-109.compute-1.amazonaws.com. A note at the bottom states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

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

```
ubuntu@ip-172-31-71-64:~$ To run a command as administrator (user "root"), use "sudo <command>". See "man sudo_root" for details.

ubuntu@ip-172-31-71-64:~$ 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 Tue Nov 11 14:37:30 UTC 2025

System load: 0.16 Temperature: -273.1 C
Usage of /: 25.8% of 6.71GB Processes: 116
Memory usage: 23% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 172.31.71.64

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-172-31-71-64:~$
```

## 2. Deploying the web application using Nginx and Tomcat server

.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

```
Reading package lists... done
ubuntu@ip-172-31-71-64:~$ sudo apt-get install docker.io
```

```
Setting up ubuntu-fan (0.12.16+24.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /usr/lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (28.2.2-0ubuntu1~24.04.1) ...
info: Selecting GID from range 100 to 999 ...
info: Adding group `docker' (GID 113) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /usr/lib/systemd/system/docker.socket.
Processing triggers for dbus (1.14.10-4ubuntu4.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

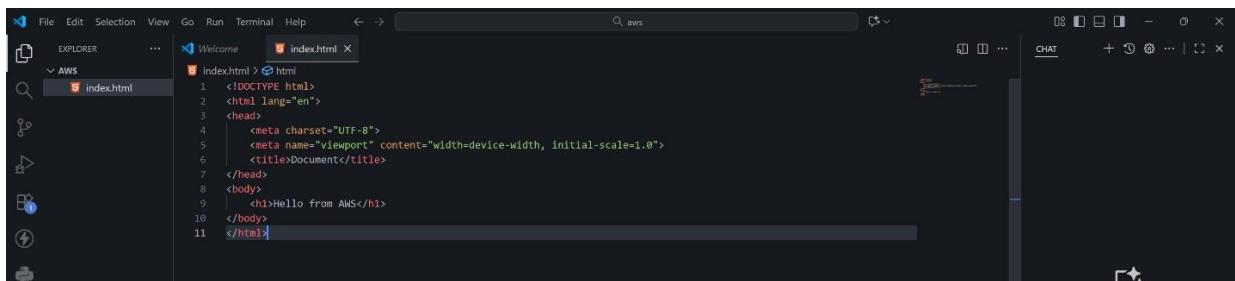
No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

```

apt-get install nano
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 ubutu 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

The screenshot shows the AWS Academy Learner Lab interface. On the left, there's a sidebar with options like Home, Modules, Discussions, Grades, and Lucid (Whiteboard). The main area has tabs for AWS and a terminal window showing command-line output. On the right, there's a sidebar titled "Learner Lab" with sections for Environment Overview, Environment Navigation, and various AWS services like EC2, S3, and Lambda. A message at the bottom says "Instructions last updated: 2025-06-24".

Click on AWS

Click on EC2

The screenshot shows the AWS Cloud Home page. It features a "Recently visited" section with "EC2" highlighted. Below it are four main cards: "Welcome to AWS" (with links to Getting started with AWS and Training and certification), "AWS Health" (showing 0 open issues, 1 scheduled change, and 0 other notifications), and "Cost and usage" (showing current month cost of \$0.12, forecasted end-of-month cost of \$0.12, and a bar chart for savings opportunities). At the bottom, there are links for CloudShell, Feedback, and cookie preferences.

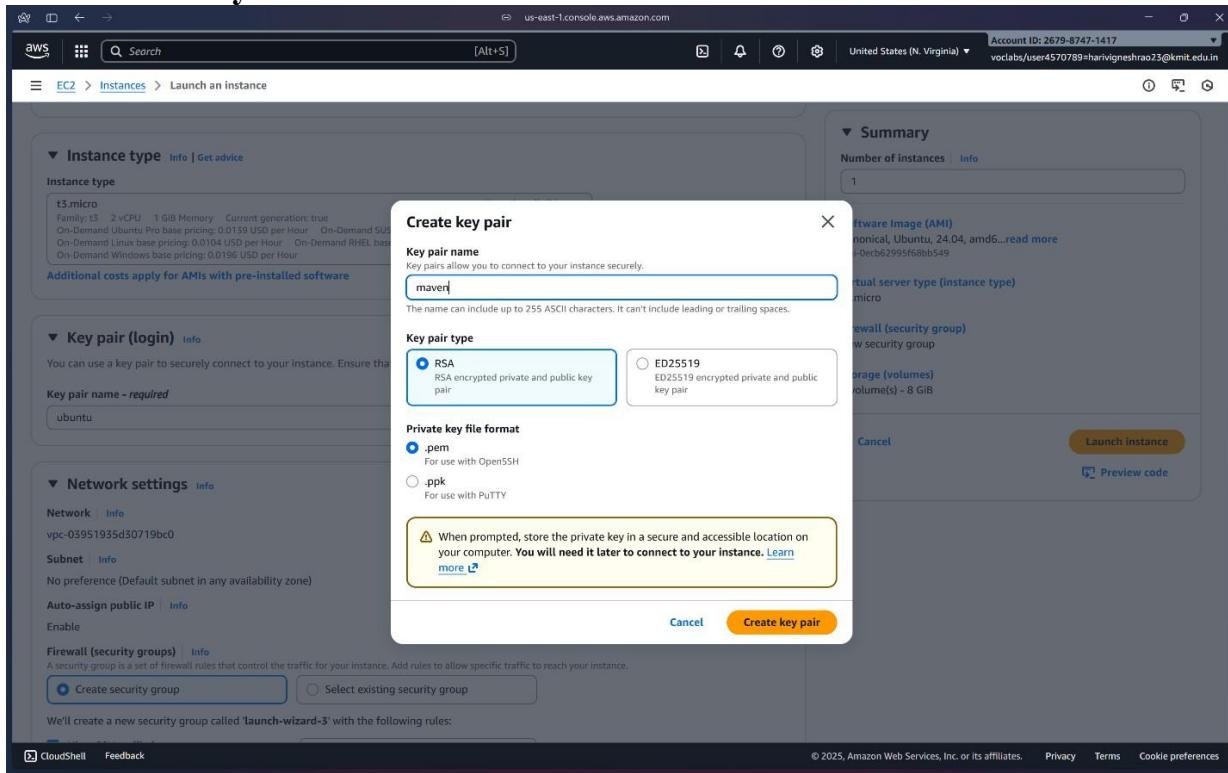
## Click on launch instance

The screenshot shows the AWS EC2 dashboard with the 'Instances' section selected. In the center, there's a 'Launch instance' button. To its right, under 'Service health', it says 'This service is operating normally'. Below that is a table for 'Zones' showing six entries: us-east-1a through us-east-1f, each associated with zone ID use1-az4.

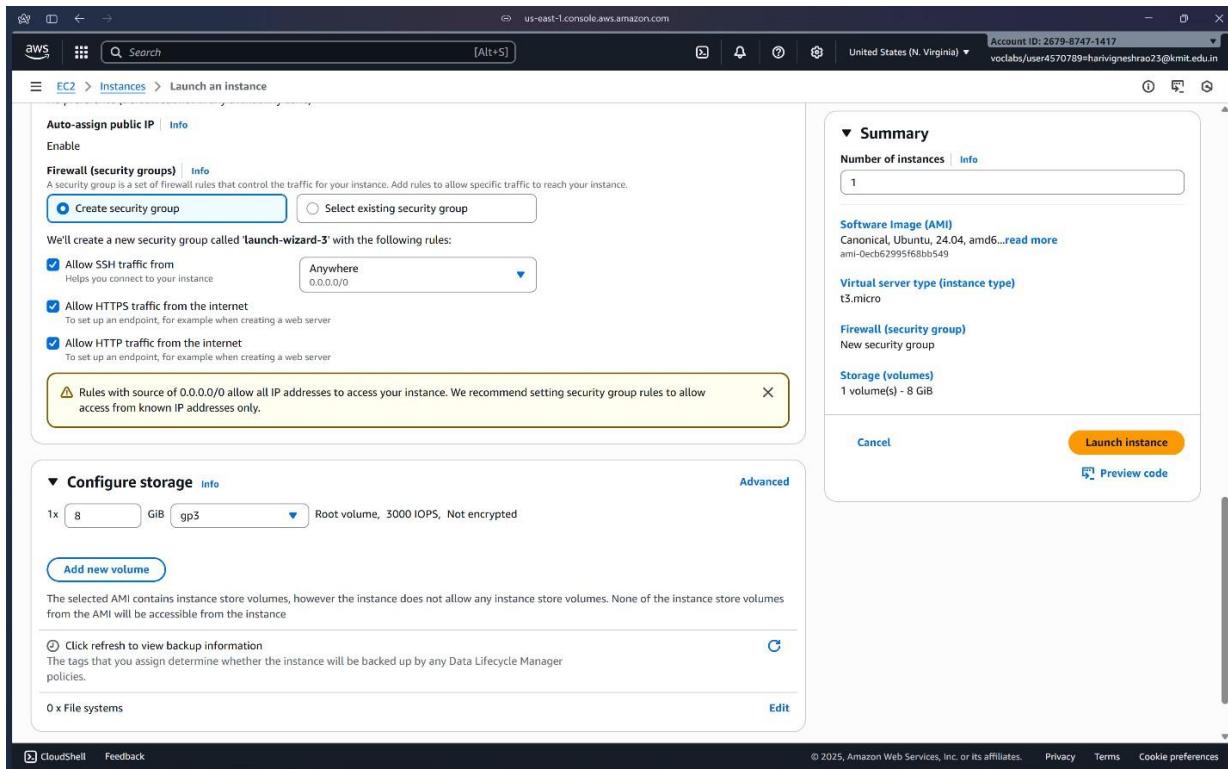
## Give name

This screenshot shows the 'Launch an instance' wizard. Step 1 is 'Name and tags'. It has a text input field with 'MavenWebProject' and a 'Add additional tags' button. Step 2, 'Application and OS Images (Amazon Machine Image)', shows a search bar and a grid of recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. A note says 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' is selected. Step 3, 'Summary', shows 'Number of instances' set to 1. Other summary details include the software image (Canonical, Ubuntu, 24.04, amd64), virtual server type (t3.micro), and storage (1 volume(s) - 8 GiB). A large orange 'Launch instance' button is at the bottom.

## Create new key



Check all the boxes under network and click on launch instance



## Click on instances

The screenshot shows the AWS EC2 Instances launch log page. At the top, there is a green success message: "Successfully initiated launch of instance (i-04fc19ed2435ec12c)". Below this, there is a "Launch log" section. Under "Next Steps", there are eight cards:

- Create billing usage alerts**: To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds. [Create billing alerts](#)
- Connect to your instance**: Once your instance is running, log into it from your local computer. [Connect to instance](#)
- Connect an RDS database**: Configure the connection between an EC2 instance and a database to allow traffic flow between them. [Connect an RDS database](#)
- Create EBS snapshot policy**: Create a policy that automates the creation, retention, and deletion of EBS snapshots. [Create EBS snapshot policy](#)
- Manage detailed monitoring**: Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period. [Manage detailed monitoring](#)
- Create Load Balancer**: Create an application, network gateway or classic Elastic Load Balancer. [Create Load Balancer](#)
- Create AWS budget**: AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location. [Create AWS budget](#)
- Manage CloudWatch alarms**: Create or update Amazon CloudWatch alarms for the instance. [Manage CloudWatch alarms](#)

At the bottom, there is a "View all Instances" button.

## Wait until you get 1 test passes

The screenshot shows the AWS EC2 Instances page. The left sidebar includes sections for EC2, Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, and Network Interfaces.

The main content area displays the "Instances (1) Info" table:

| Name           | Instance ID         | Instance state | Instance type | Status check | Alarm status                | Availability Zone | Public IPv4 DNS   |
|----------------|---------------------|----------------|---------------|--------------|-----------------------------|-------------------|-------------------|
| MavenWebPro... | i-04fc19ed2435ec12c | Running        | t3.micro      | Initializing | <a href="#">View alarms</a> | us-east-1c        | ec2-54-157-122-16 |

Below the table, there is a "Select an instance" dropdown menu.

## Click on connect after checking the box

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under the 'Instances' section, the 'Instances' tab is selected. In the main content area, there is a table titled 'Instances (1/1) Info'. A single row is listed: 'MavenWebProject' (Instance ID: i-04fc19ed2435ec12c), which is 'Running' (Instance type: t3.micro) and 'Initializing' (Status check). The 'Connect' button is visible in the top right of the table header. Below the table, a detailed view for the instance 'i-04fc19ed2435ec12c (MavenWebProject)' is shown, including sections for Instance summary, Public IP, Private IP, and Elastic IP addresses.

## Copy ssh command

The screenshot shows the 'Connect' dialog box for the instance 'i-04fc19ed2435ec12c'. The 'SSH client' tab is selected. The 'Instance ID' field contains 'i-04fc19ed2435ec12c (MavenWebProject)'. Below it, a numbered list provides instructions for connecting via SSH:

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is `ubuntu.pem`.
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
     `chmod 400 "ubuntu.pem"`
4. Connect to your instance using its Public DNS:  
     `ec2-54-157-122-163.compute-1.amazonaws.com`

Below the instructions, an 'Example:' section shows the command: `ssh -i "ubuntu.pem" ubuntu@ec2-54-157-122-163.compute-1.amazonaws.com`. A note at the bottom states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

## Open the terminal, Navigate to the path

```
PS C:\Users\hariv> cd Downloads
PS C:\Users\hariv\Downloads>
```

## Run the ssh command

```
PS C:\Users\hariv\Downloads> ssh -i "ubuntu.pem" ubuntu@ec2-54-157-122-163.compute-1.amazonaws.com
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:VcP9UsM9WrBkWXUewsS8RX22A771IaJf3GPMW9YNfnOM.
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

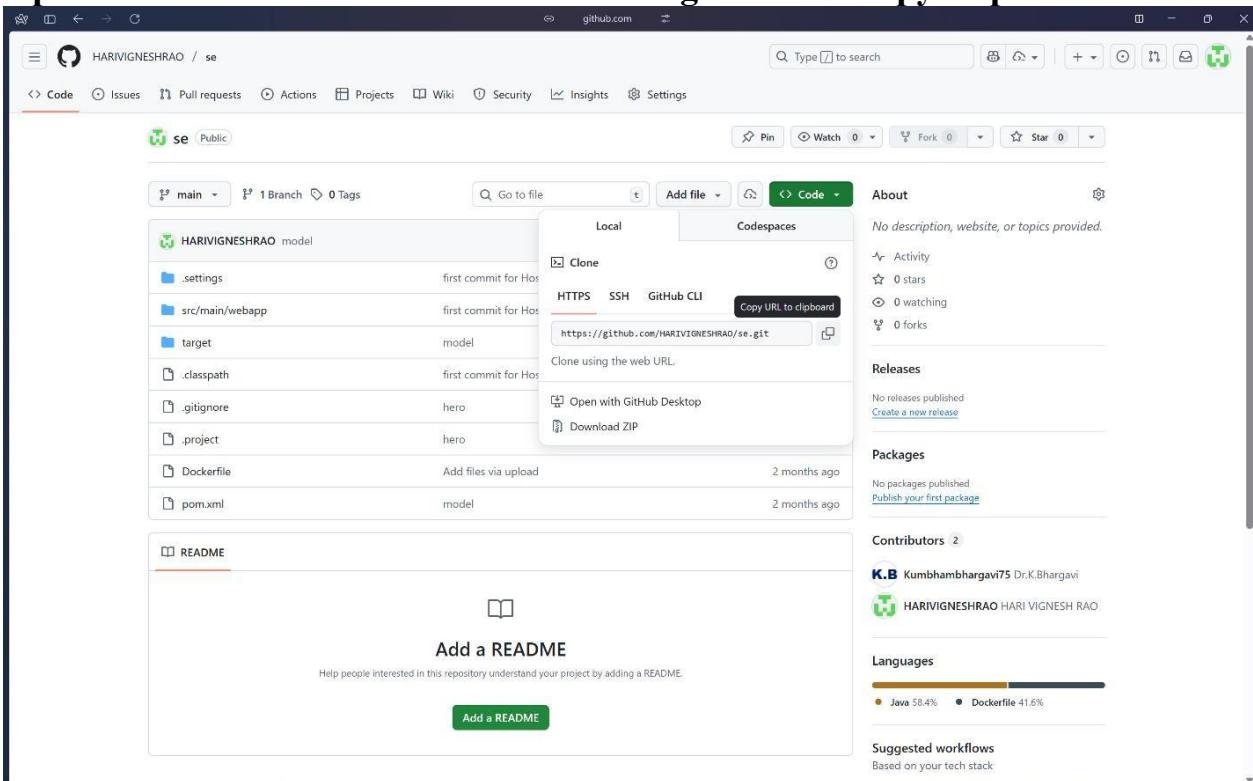
```
ubuntu@ip-172-31-2-215:~$ sudo apt-get update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1585 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [299 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.7 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1498 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [303 kB]
```

```

ubuntu@ip-172-31-2-215:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 10 not upgraded.
Need to get 76.0 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.3.3-0ubuntu1~24.04.2 [8815 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.28-0ubuntu1~24.04.1 [38.4 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ubuntu0.24.04.1 [5918 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dnsmasq-base amd64 2.90-2ubuntu0.1 [376 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 28.2.2-0ubuntu1~24.04.1 [28.3 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-fan all 0.12.16+24.04.1 [34.2 kB]
Fetched 76.0 MB in 1s (68.6 MB/s)
ubuntu@ip-172-31-2-215:~$ sudo apt 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).
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
ubuntu@ip-172-31-2-215:~$ sudo apt 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.

```

## 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:~$ ls
se
ubuntu@ip-172-31-2-215:~$ cd se
ubuntu@ip-172-31-2-215:~/se$ ls
Dockerfile pom.xml src target
```

```
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
4f4fb700ef54: Download complete
```

Run the  
container

### 3. Accessing it publicly

Go to instances and click on instance here

Copy public ipv4 address

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various EC2-related options like Dashboard, Instances, AMIs, and Elastic Block Store. The main area shows a table of instances. One instance is selected, labeled 'ubuntu' with the ID 'i-01d316bcfe56f5e30'. The details pane below shows the instance summary, including its public IPv4 address (3.85.222.109), private IP (172.31.71.64), and other metadata like instance type (t3.micro) and security group (us-east-1f).

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>

```
ubuntu@ip-172-31-71-64:~/aws$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

- iii. Go back to instances and click on option instance. Once load as below, then Click on Instance state and select terminate instances

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, CloudShell, and Feedback.

The main content area displays 'Instances (1/1) Info'. It shows a single instance named 'ubuntu' with the following details:

- Instance ID:** i-01d316bcfe56f5e30
- Instance state:** Running
- Instance type:** t3.micro
- Status check:** 3/3 checks passed
- Public IPv4 DNS:** ec2-3-85-222-109.x
- Private IP address:** 172.31.71.64
- Public IP address:** 3.85.222.109
- Instance type:** t3.micro
- Network interface:** ip-172-31-71-64.ec2.internal
- Private IP DNS name (IPv4 only):** ip-172-31-71-64.ec2.internal
- Public DNS:** ec2-3-85-222-109.compute-1.amazonaws.com
- Elastic IP addresses:** None

At the bottom right of the main content area, there are links for '2025, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

## Click on Terminated (deleted)

This screenshot shows the same EC2 Instances page after the instance has been terminated. The status bar at the top indicates a successful termination.

The main content area shows the same instance details as before, but with a different state:

- Instance state:** Terminated
- Status check:** 3/3 checks passed
- Public IPv4 DNS:** -

## iv. Now click on End lab

This screenshot shows the AWS Academy Learner Lab interface. On the left, there's a sidebar with 'Account', 'Dashboard', 'Courses' (selected), and 'Calendar'. The main content area shows a lab environment with a progress bar indicating '\$0.1 of \$50' used over 01:09. A modal dialog box in the center asks 'Are you sure you want to end the lab?' with 'Yes' and 'No' buttons.

## Maven Project

### Copy public ipv4

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links for EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces). The main content area shows 'Instances (1/1) Info'. A table lists one instance: Name: MavenWebProject, Instance ID: i-04fc19ed2435ec12c, Instance state: Running, Instance type: t3.micro, Status check: 3/3 checks passed, Alarm status: None, Availability Zone: us-east-1c, Public IPv4 DNS: ec2-54-157-122-16. Below the table, the instance details for 'i-04fc19ed2435ec12c (MavenWebProject)' are shown under the 'Details' tab. It includes sections for Instance summary, Public IPv4 address (54.157.122.163), Private IPv4 addresses (172.31.2.215), Public DNS (ec2-54-157-122-163.compute-1.amazonaws.com), and other configuration details like VPC ID, Subnet ID, and IAM Role.

Enter this url in browser and click enter

<http://54.157.122.163/HospitalMgmtSystem/>

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

The screenshot shows a web browser window with the URL 54.157.122.163/HospitalMgmtSystem/ in the address bar. The page content is a simple welcome message: 'Welcome to HospitalMgmtSystem !'.

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

Run the following commands to stop the container

```
ubuntu@ip-172-31-2-215:~/se/target$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
c4936d663ab9 maven "catalina.sh run" 7 minutes ago Up 7 minutes 0.0.0.0:80->8080/tcp, [::]:80->8080/tcp NAMES
beautiful_mendeleev
ubuntu@ip-172-31-2-215:~/se/target$ sudo docker stop c4936d663ab9
c4936d663ab9
ubuntu@ip-172-31-2-215:~/se/target$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
```

## Terminate your instance

The screenshot shows the AWS EC2 Instances page. A single instance, `i-04fc19ed2435ec12c` (MavenWebProject), is listed as `Running`. The `t3.micro` instance type is shown. The `Public IPv4 address` is `54.157.122.163`. The `Private IP address` is `172.31.2.215`. The `VPC ID` is `vpc-03951935d30719bc0`. The `Subnet ID` is `subnet-03f5829965787c187`. The `Instance state` is `Running`. The `Actions` dropdown menu is open, and the `Terminate (delete) instance` option is highlighted.

The screenshot shows the AWS EC2 Instances page with the same instance details. A confirmation dialog box titled "Terminate (delete) instance" is displayed. It contains a warning message: "On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost." Below this, it asks, "Are you sure you want to terminate these instances?" There are two input fields: "Instance ID" (set to `i-04fc19ed2435ec12c (MavenWebProject)`) and "Termination protection" (set to `Disabled`). A note states: "To confirm that you want to delete the instances, choose the terminate button below. Instances with termination protection enabled will not be terminated. Terminating the instance cannot be undone." At the bottom, there is a checkbox for "Skip OS shutdown" (unchecked) and a large orange "Terminate (delete)" button. The right side of the dialog shows the instance details again.

## End your lab

The screenshot shows the AWS Academy Learner Lab interface. The top navigation bar includes "AWS", "Used \$0.1 of \$50", "03:01", "Start Lab", "End Lab" (which is highlighted in orange), "AWS Details", "Readme", and "Reset". On the left, there's a sidebar with "Account", "Dashboard", "Courses" (highlighted in black), and "Lucid (Whiteboard)". The main area shows a terminal window with the command `eee_w_5341232@runweb195748:~$`. To the right is a "Learner Lab" section with a "Environment Overview" panel.