



**VIT®**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

---

**NAME:** K. Manoj Kumar

**REGISTER NUMBER:** 23MIS0159

**COURSE NAME:** Agile Development Process and DevOps Lab

**COURSE CODE:** ISWE406P

**SLOT:** L51+L52

## **TASK 1: Jenkins Familiarization Objective:**

**Understand Jenkins UI and basic navigation**

**Tasks:**

**1. Open Jenkins Dashboard in browser**

**2. Identify:**

**o Dashboard**

**o Manage Jenkins**

**o New Item**

**o Build History**

**3. Check Jenkins version**

**Expected Output:**

**Screenshot or note of Jenkins version**

**DASHBOARD, Jenkins Version:**

The screenshot shows the Jenkins dashboard at the URL `localhost:8085`. The top navigation bar includes links for Home, Help, and Log Out. On the left, there are links for New Item, Build History, and a dropdown for Build Queue which shows "No builds in the queue". Below this is a section for Build Executor Status indicating "0 of 2 executors busy". The main content area displays a table of build items. The columns are labeled S, W, Name, Last Success, Last Failure, and Last Duration. The data rows are:

S	W	Name	Last Success	Last Failure	Last Duration
Green circle	Yellow star	demo1	37 min #4	N/A	66 ms
Green circle	Yellow star	GIT	38 min #5	N/A	1.6 sec
Green circle	Yellow star	GIT DEMO	3 days 3 hr #1	N/A	1.7 sec
Green circle	Yellow star	Task1	38 min #6	N/A	1.9 sec

At the bottom, there are icons for S, M, L, and a REST API link.

## NEWITEMS:

The screenshot shows the Jenkins 'New Item' creation interface. At the top, there's a header with the Jenkins logo and a search bar. Below it, a sub-header says 'New Item'. A text input field is labeled 'Enter an item name' with the value 'agilelab'. A section titled 'Select an item type' lists several options with icons: 'Freestyle project' (classic general purpose job type), 'Pipeline' (orchestrates long-running activities), 'Multi-configuration project' (suitable for large numbers of configurations), 'Folder' (creates a container for nested items), 'Multibranch Pipeline' (creates a set of Pipeline projects for detected branches), and 'Organization Folder' (creates a set of multibranch project subfolders). Below this is a note: 'If you want to create a new item from other existing, you can use this option:'. A 'Copy from' input field contains 'Type to autocomplete...' and a blue 'OK' button.

## Build History:

The screenshot shows the Jenkins 'Build History' page. The left sidebar has a 'Build History' tab selected. Underneath, it shows 'Build Queue' (empty) and 'Build Executor Status' (0 of 2 executors busy). The main area is titled 'Build History of Jenkins' and displays a table of build logs. The columns are 'Build', 'Time Since', and 'Status'. The builds listed are: demo1 #4, GIT #5, Task1 #6, demo1 #3, GIT #4, Task1 #5, demo1 #2, GIT #3, Task1 #4, GIT #2, Task1 #3, Task1 #2, GIT DEMO #1, GIT #1, and demo1 #1. All builds are marked as 'stable' and completed within 42 minutes.

## MANAGE JENKINS:

The screenshot shows the Jenkins 'Manage Jenkins' configuration page. The top navigation bar includes 'Manage Jenkins', 'Search settings', 'Set up agent', 'Set up slave', and 'Docker'. The main content area is divided into several sections: 'System Configuration' (System, Clouds, Appearance), 'Security' (Security), 'Status Information' (System information, System Log, Load Statistics, About Jenkins), 'Troubleshooting' (Manage Old Data), 'Tools and Actions' (Release Configuration from Disk, Jenkins CLI, Script Console, Prepare for Shutdown), and a footer note: 'Jenkins 2.520.3'.

## TASK 2: Create First Freestyle Job

**Objective:**

Create and run a Jenkins job

**Tasks:**

### 1. Create a Freestyle project named Hello-Jenkins

New Item

Enter an item name  
Hello-Jenkins

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.
- 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**  
Creates a set of multibranch project subfolders by scanning for repositories.

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

Copy from  
Type to autocomplete

OK

### 2. Add a description

Jenkins / Hello-Jenkins / Configuration

Configure General Enabled

General

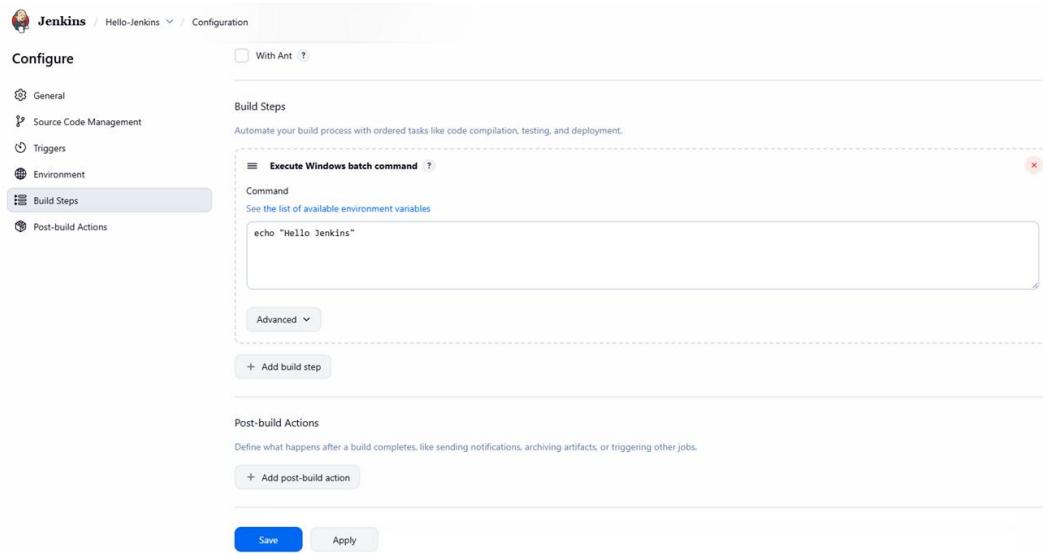
Description  
Hello-Jenkins freestyle project

Plain text Preview

Discard old builds ?  
 GitHub project  
 This project is parameterized ?  
 Throttle builds ?  
 Execute concurrent builds if necessary ?

Advanced ▾

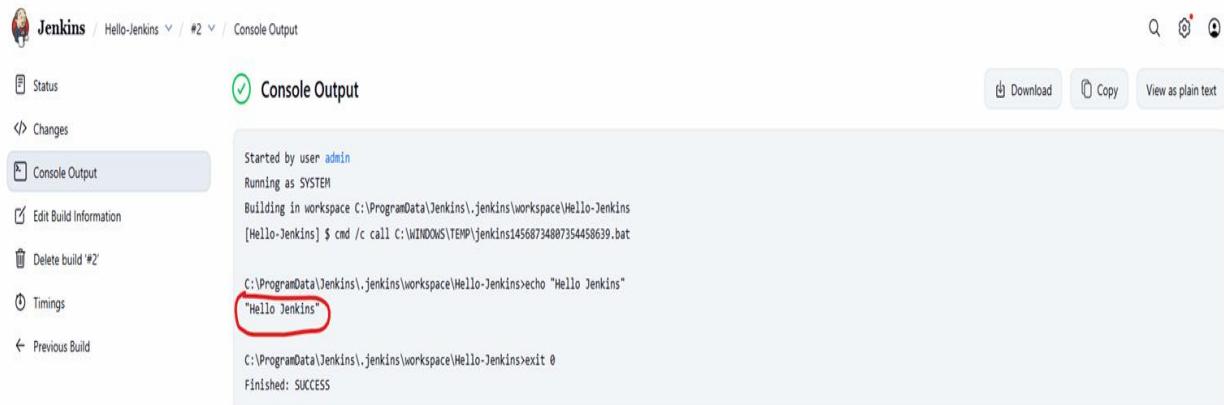
### 3. Add build step: o Execute shell / Windows batch command o Print "Hello Jenkins"



The screenshot shows the Jenkins job configuration page for 'Hello-Jenkins'. The 'Build Steps' section is active, displaying a single step: 'Execute Windows batch command'. The command entered is 'echo "Hello Jenkins"'. Below this, there are 'Post-build Actions' and buttons for 'Save' and 'Apply'.

### 4. Build the job manually

**Expected Output: Console output showing message**



The screenshot shows the Jenkins console output for build #2. The output window is titled 'Console Output' and shows the command 'echo "Hello Jenkins"' being executed, with the output 'Hello Jenkins' highlighted and circled in red. Other visible text includes the build start information and the final 'Finished: SUCCESS' message.

```
Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins
[Hello-Jenkins] $ cmd /c call C:\WINDOWS\TEMP\jenkins14568734807354458639.bat
C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins>echo "Hello Jenkins"
Hello Jenkins
C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins>exit 0
Finished: SUCCESS
```

## TASK 3: Jenkins Workspace & Commands

### Objective: Understand workspace usage

Tasks:

#### 1. Navigate to job workspace

The screenshot shows the Jenkins workspace interface for the 'Hello-Jenkins' project. At the top, there are links for Status, Changes, and Workspace. The 'Workspace' link is highlighted. Below it, there's a 'Wipe Out Current Workspace' button, a 'Build Now' button, a 'Configure' link, a 'Delete Project' link, and a 'Rename' link. A 'Builds' section shows two builds: #2 at 4:24PM and #1 at 4:22PM. The URL in the browser address bar is `localhost:8085/job>Hello-Jenkins/ws/`.

#### 2. Create a text file using build step

The screenshot shows the Jenkins configuration page for the 'Hello-Jenkins' project. Under the 'Build Steps' section, there are two steps: 'Command' and 'Execute Windows batch command'. The 'Command' step contains the command `echo "Hello Jenkins"`. The 'Execute Windows batch command' step contains the following script:

```
@echo off
echo ## Build Status ## > status.txt
echo File created inside workspace >> status.txt
echo Timestamp: %date% %time% >> status.txt
echo.
echo --- Displaying file contents ---
type status.txt
```

Both steps have an 'Advanced' dropdown menu. A '+ Add build step' button is located at the bottom left.

#### 3. Display file contents in console

The screenshot shows the Jenkins console output page for the 'Hello-Jenkins' project. The 'Console Output' tab is selected. The output shows the execution of the build steps:

```
Started by user admin
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins
[Hello-Jenkins] $ cmd /c call C:\WINDOWS\TEMP\jenkins1285915517784394429.bat

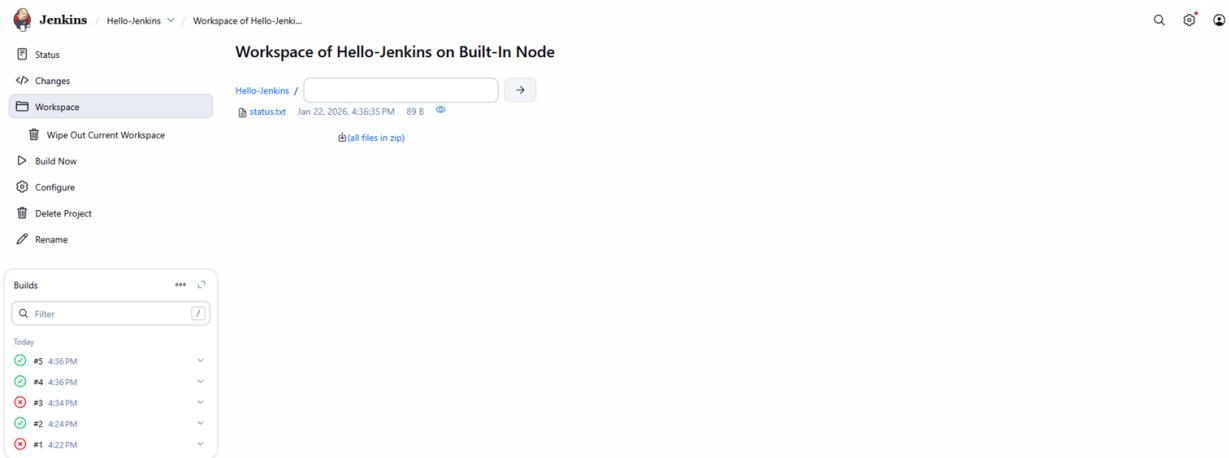
C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins>echo "Hello Jenkins"
"Hello Jenkins"

C:\ProgramData\Jenkins\jenkins\workspace\Hello-Jenkins>exit 0
[Hello-Jenkins] $ cmd /c call C:\WINDOWS\TEMP\jenkins18336715499502381890.bat

--- Displaying file contents ---
## Build Status ##
File created inside workspace
Timestamp: 22-01-2026 16:36:35.43
Finished: SUCCESS
```

At the top right, there are buttons for 'Download', 'Copy', and 'View as plain text'.

## Expected Output: File created inside workspace

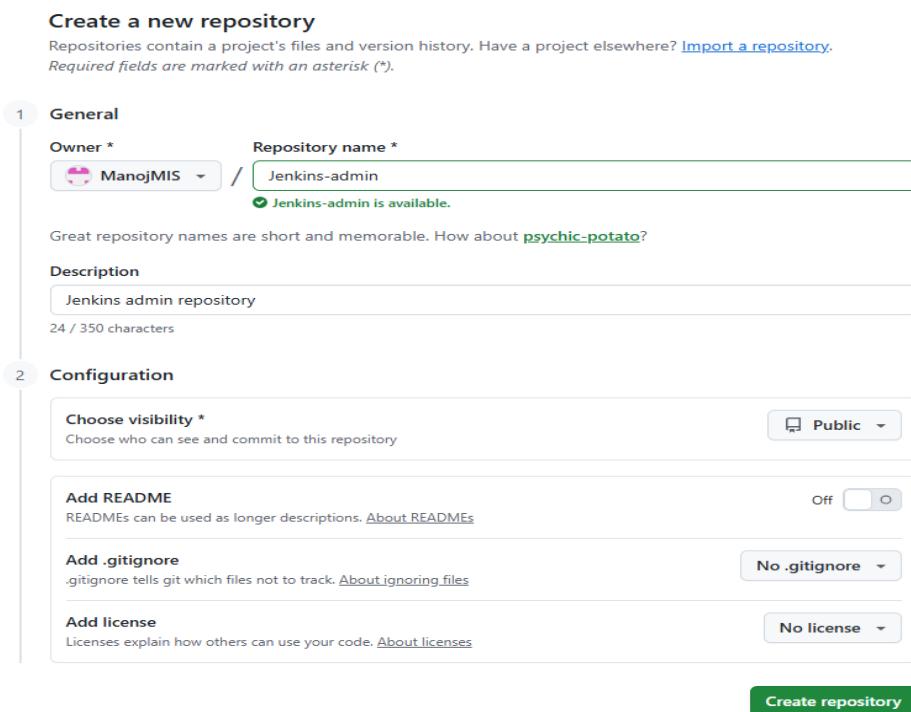


The screenshot shows the Jenkins interface for the 'Hello-Jenkins' project. The top navigation bar includes links for Jenkins, Hello-Jenkins, and Workspace of Hello-Jenkins. Below the navigation is a search bar and a refresh button. The main content area is titled 'Workspace of Hello-Jenkins on Built-In Node'. It displays a file named 'status.txt' with a size of 89 B, last modified on Jan 22, 2026, at 4:36:35 PM. A link to download the file as a zip is provided. On the left, there's a sidebar with options like Status, Changes, Workspace (which is selected), Wipe Out Current Workspace, Build Now, Configure, Delete Project, and Rename. Below the sidebar is a 'Builds' section showing a list of recent builds with their numbers, times, and statuses.

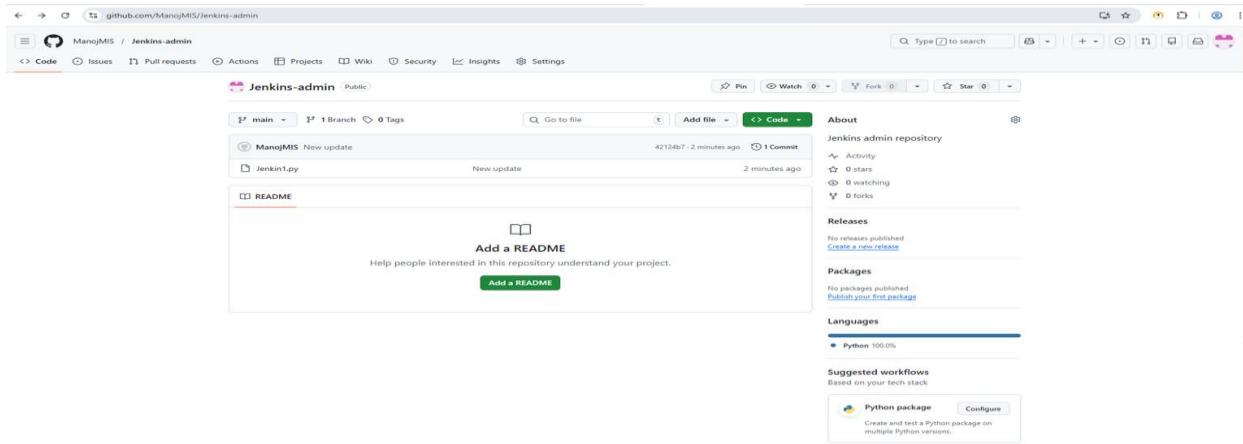
## TASK 4: Git Integration

### Objective: Integrate Jenkins with GitHub

#### Tasks: 1. Create a GitHub repository with sample code



The screenshot shows the GitHub 'Create a new repository' form. The first step, 'General', is active. It asks for the 'Owner' (set to 'ManojMIS') and 'Repository name' (set to 'Jenkins-admin'). A note says 'Jenkins-admin is available.' Below these fields is a suggestion for a repository name: 'psychic-potato?'. The 'Description' field contains 'Jenkins admin repository' with a character count of 24 / 350 characters. The second step, 'Configuration', is shown below. It includes options for 'Choose visibility' (set to 'Public'), 'Add README' (disabled), 'Add .gitignore' (disabled), and 'Add license' (disabled). At the bottom right is a green 'Create repository' button.



## 2. Configure Git in Jenkins

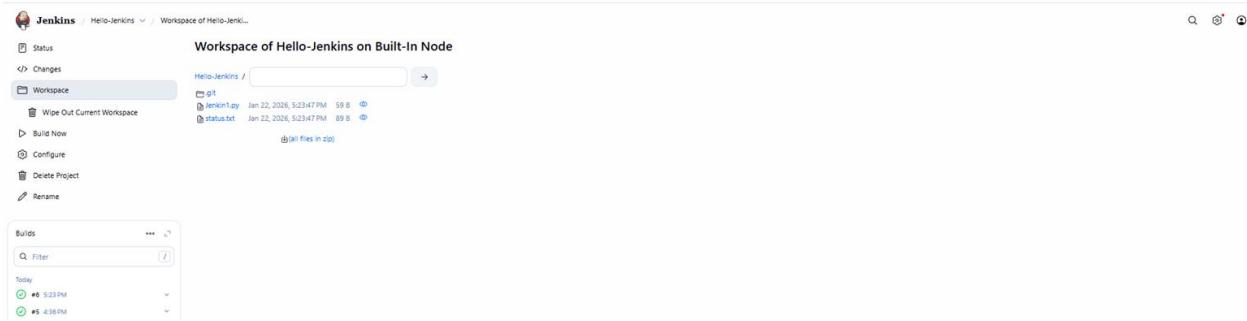
The screenshot shows the Jenkins configuration interface under 'Source Code Management'. It shows a 'Git' repository selected with 'Repository URL' set to 'https://github.com/ManojMIS/jenkins-admin.git'. The 'Branches to build' section shows '\*main'. The 'Repository browser' is set to '(Auto)'.

## 3. Add Git repository URL in job

The screenshot shows the Jenkins job configuration under 'Build Steps'. It shows three 'Execute Windows batch command' steps. The first step has the command 'echo "Hello Jenkins"'. The second step has a complex command involving timestamping and file creation. The third step has the command 'echo "Jenkins GIT integration"'. An '+ Add build step' button is visible at the bottom.

#### 4. Build and verify code checkout

**Expected Output:** Source code visible in workspace



The screenshot shows the Jenkins interface for a project named 'Hello-Jenkins'. On the left, there's a sidebar with options like Status, Changes, Workspace (which is currently selected), Wipe Out Current Workspace, Build Now, Configure, Delete Project, and Rename. The main area is titled 'Workspace of Hello-Jenkins on Built-In Node' and shows a list of files: 'g1.txt' (Jan 22, 2026, 5:23:47 PM, 59 B), 'Jenkins.log' (Jan 22, 2026, 5:23:47 PM, 89 B), and 'status.txt' (Jan 22, 2026, 5:23:47 PM, 89 B). Below this is a 'Builds' section with a 'Filter' input field and a dropdown menu set to 'Today'. It lists two builds: one at 5:23PM and another at 4:38PM.

#### TASK 5: Poll SCM Trigger

**Objective:** Automatically trigger builds on code change

**Tasks:** 1. Enable Poll SCM

2. Set schedule: \* \* \* \* \*

3. Modify GitHub file and commit

4. Observe automatic build **Expected Output:** • Build triggered without manual action

#### TASK 6: Parameterized Build

**Objective:** Use parameters in Jenkins job

**Tasks:** 1. Enable parameterized build

2. Add String parameter USERNAME

3. Print parameter value in build step **Expected Output:** • Console output showing parameter value

#### TASK 7: Java Build Using Jenkins

**Objective:** Compile Java program using Jenkins

**Tasks:** 1. Create simple Hello.java

2. Compile using javac

3. Run Java program **Expected Output:** • Java output in console

## **TASK 8: Archive Artifacts**

**Objective:** Store build outputs

- Tasks:**
1. Generate .class or .jar file
  2. Archive artifacts in post-build action

3. Download artifact from Jenkins UI **Expected Output:** • Artifact available for download

## **TASK 9: Users & Roles**

**Objective:** Manage Jenkins users

- Tasks:**
1. Create two users
  2. Assign read-only permission to one user
  3. Assign build permission to another user **Expected Output:** • Permission differences verified

## **TASK 10: Simple Jenkins Pipeline**

**Objective:** Create basic pipeline

- Tasks:**
1. Create Pipeline job
  2. Write pipeline with stages: o Checkout o Build o Test
  3. Run pipeline **Expected Output:** • Pipeline stage view

## **TASK 11: Jenkinsfile from Git**

**Objective:** Pipeline as Code

- Tasks:**
1. Create Jenkinsfile in Git repo
  2. Configure pipeline from SCM
  3. Trigger build **Expected Output:** • Pipeline executed from Git

## **TASK 12: Post-Build Actions**

**Objective:** Handle build result

- Tasks:**
1. Add post section

2. Print message on success/failure
- Expected Output: • Appropriate message displayed

**TASK 13: Trigger Job from Another Job**

**Objective:** Job chaining

**Tasks:** 1. Create Job-A and Job-B

3. Configure Job-B to trigger after Job-A
- Expected Output: • Job-B triggered automatically

**TASK 14: Workspace Cleanup**

**Objective:** Manage disk usage

**Tasks:** 1. Install Workspace Cleanup plugin

2. Clean workspace before build
- Expected Output: • Workspace cleared before execution

**TASK**

**15: Mini CI Project**

**Objective:** Implement basic CI flow

**Tasks:** 1. Git commit → Jenkins build

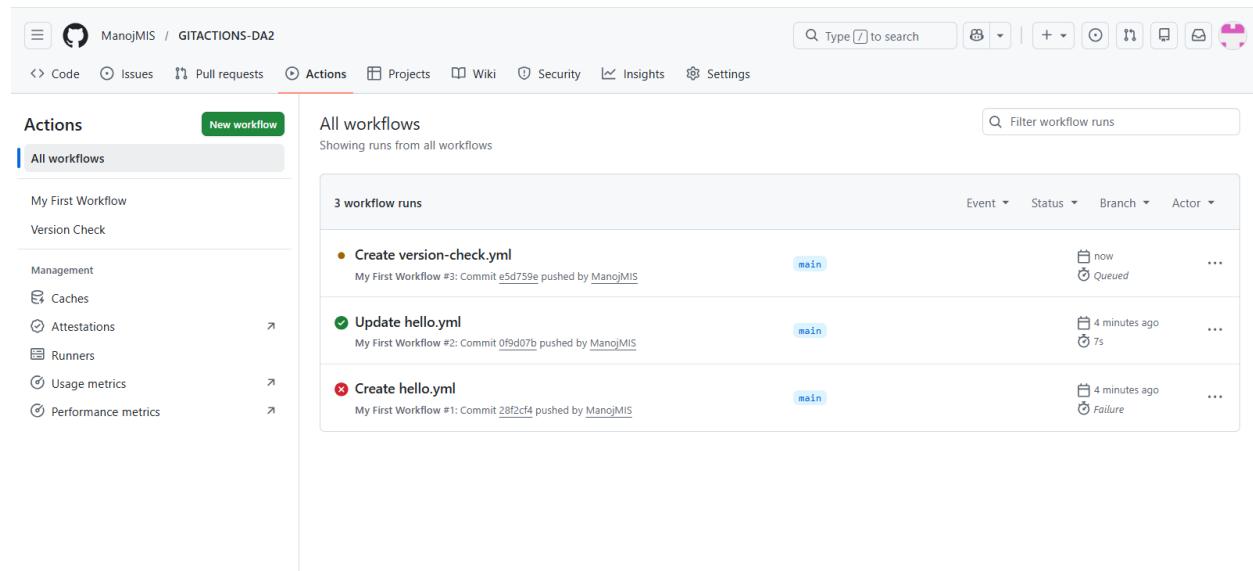
2. Compile code

3. Archive artifacts

4. Fail build on error
- Expected Output: • Automated CI pipeline

## TOOL:GIT ACTIONS

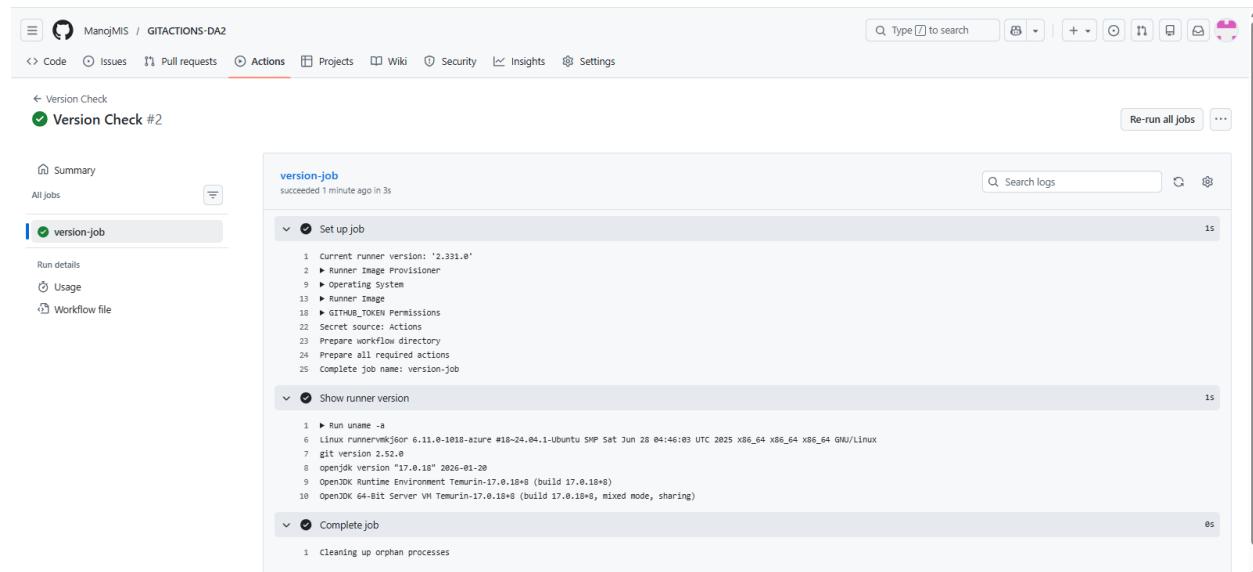
### 1.)GIT ACTIONS DASBOARD:



The screenshot shows the GitHub Actions dashboard for the repository "GITACTIONS-DA2". The left sidebar has a "Actions" section with a "New workflow" button and a "All workflows" tab selected. The main area displays "All workflows" with "Showing runs from all workflows". There are three workflow runs listed:

- Create version-check.yml**: Status: now, Actor: Queued
- Update hello.yml**: Status: 4 minutes ago, Duration: 7s
- Create hello.yml**: Status: 4 minutes ago, Actor: Failure

### VERSION AND OUTPUT:



The screenshot shows the GitHub Actions job details for "Version Check #2". The left sidebar has a "Version Check" section with a "Re-run all jobs" button. The main area shows the "version-job" step with its logs:

**version-job**  
succeeded 1 minute ago in 3s

**Set up job**

```
1 Current runner version: '2.331.0'
2 ▶ Runner Image Provisioner
9 ▶ Operating System
13 ▶ Runner Image
18 ▶ GITHUB_TOKEN: Permissions
22 Secret source: Actions
23 Prepare workflow directory
24 Prepare all required actions
25 Complete job name: version-job
```

**Show runner version**

```
1 ▶ Run uname -a
6 Linux runnervekjk6qr 6.11.0-1018-azure #18-24.04.1-Ubuntu SMP Sat Jun 28 04:46:03 UTC 2025 x86_64 x86_64 GNU/Linux
7 git version 2.52.0
8 openjdk version "17.0.18" 2026-01-20
9 OpenJDK Runtime Environment Temurin-17.0.18+8 (build 17.0.18+8)
10 OpenJDK 64-Bit Server VM Temurin-17.0.18+8 (build 17.0.18+8, mixed mode, sharing)
```

**Complete job**

```
1 Cleaning up orphan processes
```

## 2.)First Freestyle job created and Output:

The screenshot shows the GitHub Actions interface for a workflow named 'Create hello1.yml #4'. The workflow has one job named 'demo' which succeeded in 4 seconds. The steps in the job are: Set up job, Run echo "Hello World" (which includes a step to run echo "Hello World" and a step to echo "Hello World"), and Complete job.

## 3.)Text File Created:

The screenshot shows the GitHub Actions workflow editor for a repository named 'GITACCTIONS-DA2'. The workflow file 'workspace1.yml' is open in the main editor area. The code defines a workflow with a single job named 'hello1-job' that runs on 'ubuntu-latest' and contains steps to create a file 'test.txt' with the content 'This is workspace file', and to list files in the workspace. The right sidebar shows the Marketplace for Actions with several featured actions listed.

```
1 name: Workspace Task 3
2
3 on: workflow_dispatch
4
5 jobs:
6   hello1-job:
7     runs-on: ubuntu-latest
8     steps:
9       - name: Create file
10         run: echo "This is workspace file" > test.txt
11
12       - name: Show file content
13         run: cat test.txt
14
15       - name: List files
16         run: ls -l
```

## Job Workspace:

The screenshot shows the Jenkins Job Management interface. On the left, a sidebar lists various Jenkins jobs and management tasks. The main area displays a table of 'All workflow runs' with the following data:

Run	Description	Status	Event	Actor
Workspace Task 3 #1	Workspace Task 3 #1: Manually run by ManojMIS	main	2 minutes ago	7s
Hello Jenkins Equivalent #2	Hello Jenkins Equivalent #2: Manually run by ManojMIS	main	5 minutes ago	7s
Hello Jenkins Equivalent #1	Hello Jenkins Equivalent #1: Manually run by ManojMIS	main	6 minutes ago	8s
Create workspace.yml	Workspace Demo #5: Commit 64377bd pushed by ManojMIS	main	14 minutes ago	9s
Create hello1.yml	Workspace Demo #4: Commit 78b4035 pushed by ManojMIS	main	15 minutes ago	7s
Version Check #2	Version Check #2: Manually run by ManojMIS	main	18 minutes ago	7s

## OUTPUT:

The screenshot shows the details of a specific workflow run. The left sidebar shows the job summary and run details. The main area displays the workflow steps and their logs:

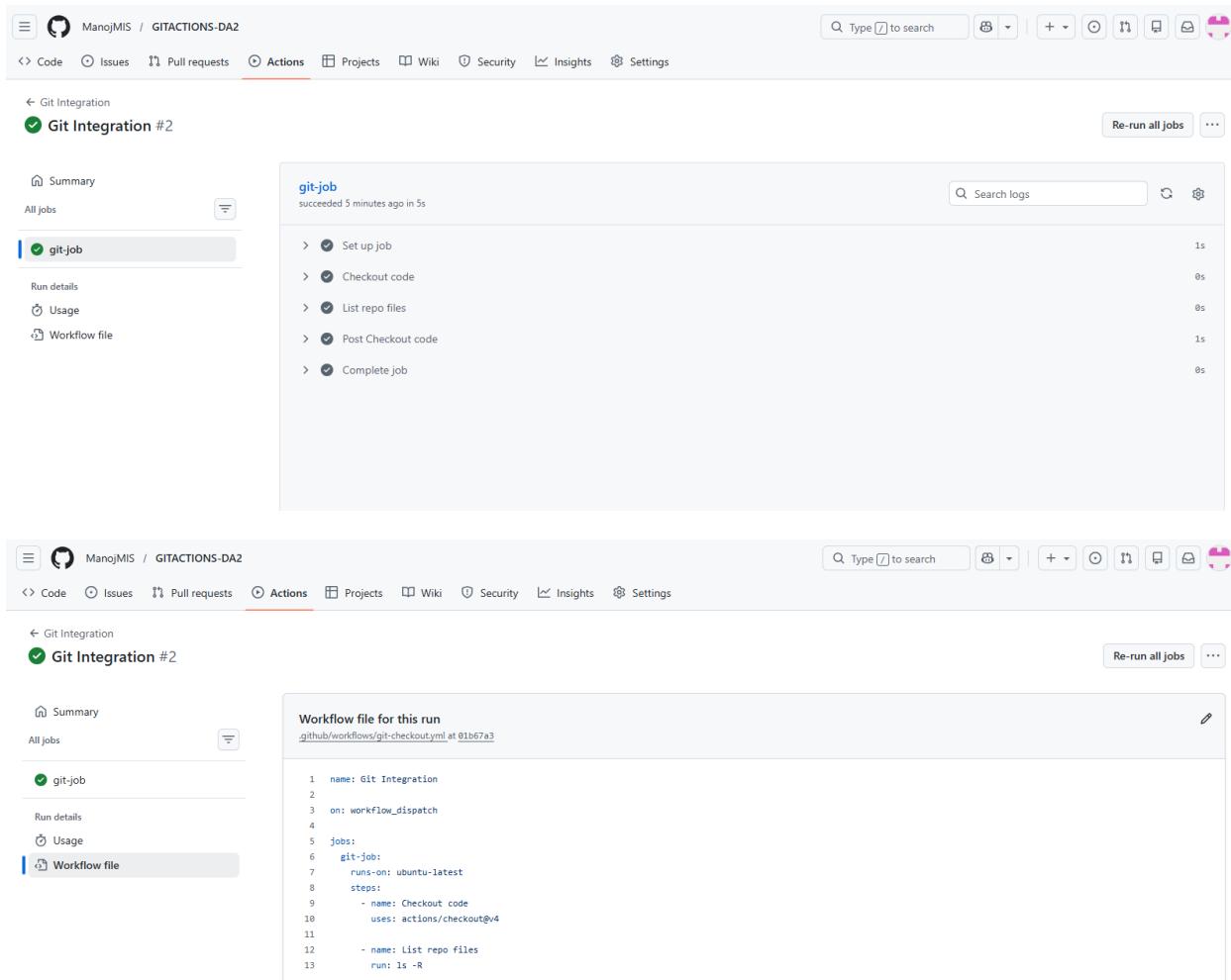
- workspace-job
  - Set up job
  - Create file
    - Run echo "This is workspace file" > test.txt
  - Show file content
    - Run cat test.txt
    - This is workspace file
  - List files
    - Run ls -l
    - total 4
    - rw-r--r-- 1 runner runner 23 Feb 4 14:11 test.txt
  - Complete job

## 4.)Git Integration Workflow created:

The screenshot shows the Jenkins Job Management interface. The sidebar lists Jenkins jobs and management tasks. The main area displays a table of '2 workflow runs' for the 'Git Integration' job, with the following data:

Run	Description	Status	Event	Actor
Git Integration #2	Git Integration #2: Manually run by ManojMIS	main	now	8s
Git Integration #1	Git Integration #1: Manually run by ManojMIS	main	now	8s

## OUTPUT:



The screenshot shows two views of a GitHub Actions workflow named "git-job".

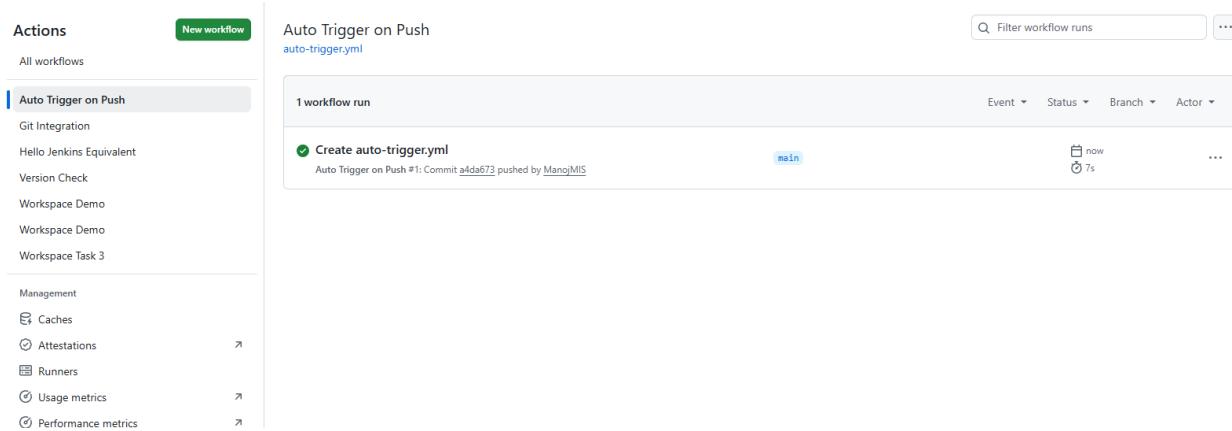
**Workflow Logs:** The top section displays the execution history of the "git-job" run. It shows the following steps and their durations:

- Set up job: 1s
- Checkout code: 0s
- List repo files: 0s
- Post Checkout code: 1s
- Complete job: 0s

**Workflow File:** The bottom section shows the YAML configuration for the workflow:

```
name: Git Integration
on: workflow_dispatch
jobs:
  git-job:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4
      - name: List repo files
        run: ls -R
```

## 5.)Auto Trigger Workflow created:



The screenshot shows the GitHub Actions interface for managing auto-triggered workflows.

**Left Sidebar:** A navigation menu with the following sections:

- Actions
- New workflow
- All workflows
- Auto Trigger on Push
- Git Integration
- Hello Jenkins Equivalent
- Version Check
- Workspace Demo
- Workspace Demo
- Workspace Task 3
- Management
- Caches
- Attestations
- Runners
- Usage metrics
- Performance metrics

**Right Panel:** The "Auto Trigger on Push" section shows a single workflow run for the "auto-trigger.yml" file. The run was triggered by a push to the "main" branch and completed successfully 7 seconds ago.

Event	Status	Branch	Actor
now	Success	main	ManojMIS

**Log:**

```
Create auto-trigger.yml
Auto Trigger on Push #1: Commit g4da673 pushed by ManojMIS
```

## OUTPUT:

The screenshot shows the GitHub Actions interface for a repository named 'GITACCTIONS-DA2'. The 'Actions' tab is selected. A workflow named 'Create auto-trigger.yml #1' is shown, which has just succeeded. The logs for the 'build-job' step are displayed, detailing the execution of various actions and commands. The logs show the creation of the commit 'auto-trigger.yml' and its push to the 'main' branch.

```
1 ► Run git log -1
4 commit a4da67379e0e8e195da0f7c1387737a9c1f021ff
5 Author: ManojMIS <manojkumar.k2023@itstudent.ac.in>
6 Date:   Wed Feb 4 19:48:12 2026 +0530
7
8     Create auto-trigger.yml

10 > Set up job
11 > Run actions/checkout@v4
12 > Show commit message
13
14 1 ► Run git log -1
15 4 commit a4da67379e0e8e195da0f7c1387737a9c1f021ff
16 5 Author: ManojMIS <manojkumar.k2023@itstudent.ac.in>
17 6 Date:   Wed Feb 4 19:48:12 2026 +0530
18 7
19 8     Create auto-trigger.yml

20 > Post Run actions/checkout@v4
21 > Complete job
```

## 6.) Parameterized Workflow Created:

The screenshot shows the GitHub Actions interface for the same repository. The 'Actions' tab is selected, displaying a list of workflows. One workflow, 'Auto Trigger on Push', is highlighted. The 'auto-trigger.yml' file is shown to have triggered two workflow runs. The first run, 'Create params.yml', was triggered by a push to the 'main' branch and completed successfully. The second run, 'Create auto-trigger.yml', was triggered by a push to the 'main' branch and completed successfully.

## OUTPUT:

The screenshot shows the GitHub Actions interface for the repository. The 'Actions' tab is selected, and a workflow named 'Parameterized Workflow #2' is shown. The 'param-job' step is highlighted. The logs for this job show it successfully printed the username 'Hello K.Manoj Kumar 23mis0159'.

```
1 ► Run echo "Hello K.Manoj Kumar 23mis0159"
2 Hello K.Manoj Kumar 23mis0159

3 > Set up job
4 > Print username
5
6 1 ► Run echo "Hello K.Manoj Kumar 23mis0159"
7 2 Hello K.Manoj Kumar 23mis0159

8 > Complete job
```

## 7.)Hello.java FILE:

The screenshot shows the GitHub interface for the repository `ManojMIS / GITACCTIONS-DA2`. The `main` branch is selected. A commit message `Create Hello.java` is shown, which includes three files: `.github/workflows`, `Create params.yml`, and `Hello.java`. The `Hello.java` file was created "now".

## JAVA workflow created:

The screenshot shows the GitHub Actions interface for the repository. The `Actions` tab is selected. Two workflow runs are listed: `Create java-build.yml` (status: `Queued`) and `Create Hello.java` (status: `1 minute ago`). The `Create Hello.java` run was triggered by a push to the `main` branch.

## OUTPUT:

The screenshot shows the GitHub Actions interface for the `Java Build #1` run. The `java-job` step is selected. The log output shows the execution of a Java program that prints "Hello from Java GitHub Actions!".

```
java-job
succeeded now in 7s

> ⚡ Set up job
> ⚡ Run actions/checkout@v4
> ⚡ Setup Java
> ⚡ Compile Java file
> ⚡ Run Java program
  1 ► Run java Hello
  7 Hello from Java GitHub Actions!
> ⚡ Post Setup Java
> ⚡ Post Run actions/checkout@v4
> ⚡ Complete job
```

## 8.)Artifacts workflow Created:

The screenshot shows the GitHub Actions interface for the repository 'ManojMIS / GITACCTIONS-DA2'. The 'Actions' tab is selected. On the left, under 'All workflows', there is a list of actions: Archive Artifacts, Auto Trigger on Push, Git Integration, and Hello Jenkins Equivalent. The main area displays the 'Create artifacts.yml' workflow run, which was triggered by an 'Auto Trigger on Push' event. The run is in progress, indicated by a green circle icon. The status bar shows 'main' and a timestamp of '1 minute ago'. A log entry indicates the artifact was uploaded successfully.

## OUTPUT:

This screenshot shows the detailed logs for the 'artifact-job' step of the 'Create artifacts.yml' workflow. The job has succeeded now in 6 seconds. The logs show the execution of several steps: Set up job, Run actions/checkout@v4, Compile Java, and Upload artifact. The final log entry details the successful upload of an artifact named 'Artifact compiled-class.zip' with a size of 492 bytes and an ID of 5375402547.

## 10.)

## Pipeline workflow created:

The screenshot shows the GitHub Actions interface for the repository 'ManojMIS / GITACCTIONS-DA2'. The 'Actions' tab is selected. On the left, under 'All workflows', there is a list of actions: Archive Artifacts, Auto Trigger on Push, Git Integration, Hello Jenkins Equivalent, Java Build, Parameterized Workflow, Simple Pipeline, Version Check, Workspace Demo, and Show more workflows... (with 'Version Check' highlighted by a yellow oval). The main area displays the 'Create pipeline.yml' workflow run, which was triggered by an 'Auto Trigger on Push' event. The run is completed, indicated by a green circle icon. The status bar shows 'main' and a timestamp of 'now'. A log entry indicates the pipeline was successfully triggered.

## OUTPUT:

The screenshot shows the GitHub Pipeline interface for a repository named 'Simple Pipeline'. The pipeline has one job, 'pipeline-job', which has completed successfully. The pipeline consists of several stages: 'Checkout Stage', 'Build Stage', 'Test Stage', 'Post Checkout Stage', and 'Complete job'. Each stage shows the command run, its duration, and the status. The 'Checkout Stage' took 1s and ran commands related to cloning the repository. The 'Build Stage' took 8s and ran 'echo "Building project..."'. The 'Test Stage' took 8s and ran 'echo "Running tests..."'. The 'Post Checkout Stage' and 'Complete job' stages both took 0s.

11.)

## Pipeline from GIT workflow created:

The screenshot shows the GitHub Actions page for the repository 'ManojMIS / GITACTIONS-DA2'. The 'Actions' tab is selected. On the left, there is a sidebar with various workflow templates: 'Archive Artifacts', 'Auto Trigger on Push', 'Git Integration', 'Hello Jenkins Equivalent', 'Java Build', 'Parameterized Workflow', 'Pipeline from Git' (which is highlighted with a yellow circle), 'Simple Pipeline' (also highlighted with a yellow circle), 'Version Check', 'Workspace Demo', 'Workspace Demo', and 'Workspace Task 3'. The main area displays a list of 'All workflows' with 26 workflow runs. The runs are listed in descending order of age. The first two runs are 'Create from-git.yml' triggered by auto-pushes, both are in a 'Queued' state. The third run is 'Simple Pipeline' triggered manually, completed 7s ago. The fourth run is 'Create pipeline.yml' triggered by auto-push, completed 8s ago. The fifth run is 'Archive Artifacts', completed 11 minutes ago.

## OUTPUT:

The screenshot shows the GitHub Actions interface for a pipeline named 'job1'. The pipeline status is 'succeeded 1 minute ago in 4s'. The steps listed are: Set up job (1s), Run actions/checkout@v4 (0s), Run echo "Pipeline executed from repo YAML" (0s), Post Run actions/checkout@v4 (0s), and Complete job (0s). A search bar for logs is present at the top right.

12.)

## Post-build workflow created:

The screenshot shows the GitHub Actions workflows page. On the left, under 'Actions', the 'Post Build Actions' workflow is highlighted with a yellow circle. The main area displays a list of 28 workflow runs, including 'Create post-build.yml', 'Create from-git.yml', and 'Simple Pipeline', all in progress or recently completed.

## OUTPUT:

The screenshot shows the GitHub Actions interface for a pipeline named 'post-job'. The pipeline status is 'succeeded now in 4s'. The steps listed are: Set up job (0s), Build Step (0s), Success Message (0s), Failure Message (0s), and Complete job (0s). A search bar for logs is present at the top right.

13.)

### Chaining workflow created:

The screenshot shows the GitHub Actions interface for the repository 'ManojMIS / GITACTIONS-DA2'. The left sidebar has a 'Job Chaining' option highlighted with a yellow box. The main area displays 'All workflows' and '34 workflow runs'. The runs are listed as follows:

- Create chaining.yml (Pipeline from Git #4: Commit c8922ef pushed by ManojMIS) - Status: now, Queued
- Create chaining.yml (Auto Trigger on Push #10: Commit c8922ef pushed by ManojMIS) - Status: now, Queued
- Create chaining.yml (Pipeline from Git #3: Commit 63cc2d7 pushed by ManojMIS) - Status: now, 8s
- Create chaining.yml (Auto Trigger on Push #9: Commit 63cc2d7 pushed by ManojMIS) - Status: now, 7s
- Post Build Actions (Post Build Actions #2: Manually run by ManojMIS) - Status: 2 minutes ago, 7s

### OUTPUT:

#### JOB-A:

The screenshot shows the GitHub Actions interface for the repository 'ManojMIS / GITACTIONS-DA2'. The left sidebar shows 'Job Chaining' and 'Job Chaining #1' selected. The main area shows the details for 'jobA':

- Summary: succeeded now in 4s
- Set up job: 0s
- Run echo "Job A running..." (with log entries 1 and 4): 0s
- Complete job: 1s

#### JOB-B:

The screenshot shows the GitHub Actions interface for the repository 'ManojMIS / GITACTIONS-DA2'. The left sidebar shows 'Job Chaining' and 'Job Chaining #1' selected. The main area shows the details for 'jobB':

- Summary: succeeded now in 4s
- Set up job: 1s
- Run echo "Job B triggered after Job A" (with log entries 1 and 4): 0s
- Complete job: 0s

14.)

## Cleanup Workflow created:

The screenshot shows the GitHub Actions interface. On the left, a sidebar lists various workflow templates: Archive Artifacts, Auto Trigger on Push, Git Integration, Hello Jenkins Equivalent, Java Build, Job Chaining, Parameterized Workflow, Pipeline from Git, Post Build Actions, Simple Pipeline, Version Check (which is highlighted with a yellow circle), Workspace Cleanup (also highlighted with a yellow circle), Workspace Demo, and Workspace Task 3. The 'Actions' tab is selected. In the main area, the 'All workflows' section displays a table of 37 workflow runs. The columns include Event (e.g., now, 9s ago), Status (green checkmark), Branch (main), and Actor (ManojMIS). The rows list workflows like 'Create cleanup.yml', 'Job Chaining', 'Create chaining.yml', and 'Create temp file'. A search bar at the top right says 'Filter workflow runs'.

## OUTPUT:

The screenshot shows the details of a workflow run named 'cleanup-job'. The left sidebar shows 'All jobs' and the current job 'cleanup-job' is selected. The main area displays the workflow steps: 'Set up job' (1s), 'Run actions/checkout@v4' (0s), 'Create temp file' (0s), 'Run echo "temp data" > temp.txt' (1s), 'Clean workspace' (0s), 'Run rm -rf \*' (0s), 'Confirm cleanup' (0s), 'Run ls -la' (4 total 16 lines of output), 'Post Run actions/checkout@v4' (1s), and 'Complete job' (0s). A 'Search logs' bar is at the top right.

15.)

### CI workflow created:

The screenshot shows the GitHub Actions interface for the repository "ManojMIS / GITACCTIONS-DA2". The left sidebar lists various actions, with "Mini CI Project" highlighted and circled in yellow. The main area displays a table of "All workflows" with 41 runs. The columns include Event, Status, Branch, and Actor. The runs listed are:

- Create mini-ci.yml (main, now, Queued)
- Create mini-ci.yml (main, now, Queued)
- Create mini-ci.yml (main, now, Queued)
- Workspace Cleanup (main, 1 minute ago, 7s)
- Create cleanup.yml (main, 2 minutes ago)

### OUTPUT:

The screenshot shows the GitHub Actions job details for the "Create mini-ci.yml #1" job. The left sidebar shows "All jobs" and "ci-job" is selected. The main area displays the job log titled "ci-job" which succeeded now in 6s. The log entries are:

- > Set up job (2s)
- > Checkout code (0s)
- > Setup Java (0s)
- > Compile Java (1s)
- > Run Java (0s)
- > Upload Artifact (1s)
- > Post Setup Java (0s)
- > Post Checkout code (0s)
- > Complete job (0s)