

unit Barcade
NO: 03
C-IT

DevOps

Datly

09.

Experiment no: 5

Aim: to build the pipeline jobs using Maven / Gradle / Ant in Jenkins & create a pipeline to test & deploy application.

pre-requisites:

- 1) ensure jenkins is installed & running

- 2) Have your applications source code hosted in a version control system like Git.

Theory:- DevOps professionals mostly work with pipelines because pipelines can automate the processes like building, testing & deploying this application. with the help of continuous integration / continuous deployment (CI/CD) pipeline we can automate the whole process which will increase productivity.

In computing, a pipeline is a set of stages or processes linked together to form a processing system.

CI means whenever new code is committed to remote repositories like Github.

CI will continuously build, tested & merged into shared repository . CD meanr automating the further stages of pipeline automatically or manually deploying the application / code to different environments

step 1) open Jenkins by typing localhost:8080 on chrome

The screenshot shows the Jenkins dashboard with two projects listed:

Name	Last Success	Last Failure	Last Duration
MavenProject	5 min 4 sec ago	7 min 1 sec ago	4.2 sec
Project	10 days ago	N/A	83 ms

step 2) click on project compile & then click on configure.

The screenshot shows the Jenkins dashboard with the 'Configure' option highlighted for the 'Project Compile' project under the 'Changes' section.

3) Add Repo URL

The screenshot shows the Jenkins configuration page for the 'Project Compile' project, specifically the 'Source Code Management' section. A red arrow points to the 'Add' button in the 'Repositories' table, which is currently empty.

4) Go in Build environment & select invoke step-> maven integratr.

The screenshot shows the Jenkins configuration interface for a project named "Project Compile". In the "Build Steps" section, a dropdown menu is open under "Add build step". The options listed are: Execute Windows batch command, Execute shell, Invoke Ant, Invoke Gradle script, Invoke top-level Maven targets, Run with timeout, and Set build status to "Pending" on GitHub commit. The "Invoke top-level Maven targets" option is highlighted.

5) set goals or compile & apply & save.

The screenshot shows the Jenkins configuration interface for the same project. The "Build Steps" section now displays a configuration for "Invoke top-level Maven targets". It includes fields for "Maven Version" (set to "MyMaven") and "Goals" (set to "compile"). Below this, there is an "Advanced" section and a "Post-build Actions" section which is currently empty. At the bottom, there are "Save" and "Apply" buttons.

6) go to dashboard

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with links like "New Item", "People", "Build History", "Project Relationships", "Check File Fingerprint", "Manage Jenkins", and "My Views". The main area is titled "Build History" and lists three projects: "MavenProject", "Project Compile", and "Project". Each entry shows the last success time, last failure time, and last duration. A red arrow points from the handwritten note "6) go to dashboard" towards the "Build History" section. At the bottom of the dashboard, there are sections for "Build Queue" and "Build Executor Status".

7) create dev item then formed project test.

The screenshot shows the Jenkins dashboard with a "New Item (Blank)" dialog open. The title bar says "Dashboard > All > New Item (Blank)". The main area has a search bar "Search (CTRL + S)" and a user "admin". The dialog is titled "Enter an item name" with a text input field containing "Project Test". Below the input are several project types listed with icons: "Freestyle project", "Maven project", "Pipeline", and "Configuration project". A note for "Freestyle project" states: "This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software builds." A note for "Maven project" states: "Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration." A note for "Pipeline" states: "Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows)." A note for "Configuration project" states: "Projects that need a large number of different configurations, such as testing on multiple environments, platform-specific configurations, or multiple build variants." At the bottom right of the dialog are buttons for "Save" and "Cancel".

8) Enter the repo URL

The screenshot shows the Jenkins configuration page for a "Project Test" job. The title bar says "Project Test Configuration (Jenkins)" and the address bar shows "localhost:8080/job/ProjectTest/configure". The left sidebar shows "Configure", "General", "Source Code Management", "Build Triggers", "Build Environment", "Build Steps", and "Post-build Actions". The "Source Code Management" section is expanded, showing "None" selected. A "Git" option is also present. The "Repositories" section contains a "Repository URL" input field with the value "https://github.com/anjudevplm/MavenBuild.git" and a "Credentials" dropdown set to "none". Buttons for "Save" and "Apply" are at the bottom.

9) set the goals of test

The screenshot shows the Jenkins configuration page for a "Project Test" job. The title bar says "Project Test Configuration (Jenkins)" and the address bar shows "localhost:8080/job/ProjectTest/configure". The left sidebar shows "Configure", "General", "Source Code Management", "Build Triggers", "Build Environment", "Build Steps", and "Post-build Actions". The "Source Code Management" section is expanded, showing "With Ant" selected. The "Build Steps" section contains a "Invoke top-level Maven targets" step. Under "Maven Version", "Maven" is selected. Under "Goals", "test" is entered. Buttons for "Save" and "Apply" are at the bottom.

10) Go back to dashboard

The screenshot shows the Jenkins dashboard with several project cards:

- MavenProject: Last Success 17 min ago, Last Failure 19 min ago, Last Duration 4.2 sec.
- Project Compile: Last Success 4 min 39 sec ago, Last Failure N/A, Last Duration 5.6 sec.
- Project Test: Last Success 21 sec ago, Last Failure N/A, Last Duration 5.6 sec.
- Project1: Last Success 13 days ago, Last Failure N/A, Last Duration 83 ms.

Build Executor Status: 1 idle, 5 busy, 1 available.

11) Go to workspace

The screenshot shows the workspace for the "Project Test" job. It displays a list of files and their modification history:

- git (modified Sep 4, 2023, 3:48:18 PM)
- github-workflows (modified Sep 4, 2023, 3:48:18 PM)
- src (modified Sep 4, 2023, 3:48:18 PM)
- target (modified Sep 4, 2023, 3:48:18 PM)
- dependency (modified Sep 4, 2023, 3:48:18 PM)
- Dockerfile (modified Sep 4, 2023, 3:48:18 PM)
- Jenkinsfile (modified Sep 4, 2023, 3:48:18 PM)
- pom.xml (modified Sep 4, 2023, 3:48:18 PM)
- README.md (modified Sep 4, 2023, 3:48:18 PM)
- sonar-project.properties (modified Sep 4, 2023, 3:48:18 PM)

12) Go to target

The screenshot shows the workspace for the "Project Test" job, specifically the "target" directory. It lists the following files:

- classes/com/geekcamp/mintube
- generated-sources/annotations
- generated-test-sources/test-annotations
- javac-outputs/even-compiler-plugin
- javac-report
- test-classes/com/geekcamp/mintube

13) Go to search reports

The screenshot shows the Jenkins interface for a project named "Project Test". On the left, there's a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, Rename, Build History, and Filter builds. The main area is titled "Workspace of Project Test on Built-In Node". It lists two build entries: "com.geekcap.vmturbo.ThingTest" (Build 1, 2023-09-04 11:42 PM) and "TEST-com.geekcap.vmturbo.ThingTest.xml" (Build 2, 2023-09-04 11:42 PM). Below the builds, there's a link to "all files in zip". At the bottom, there are filters for "All tests" and "All tests for failures".

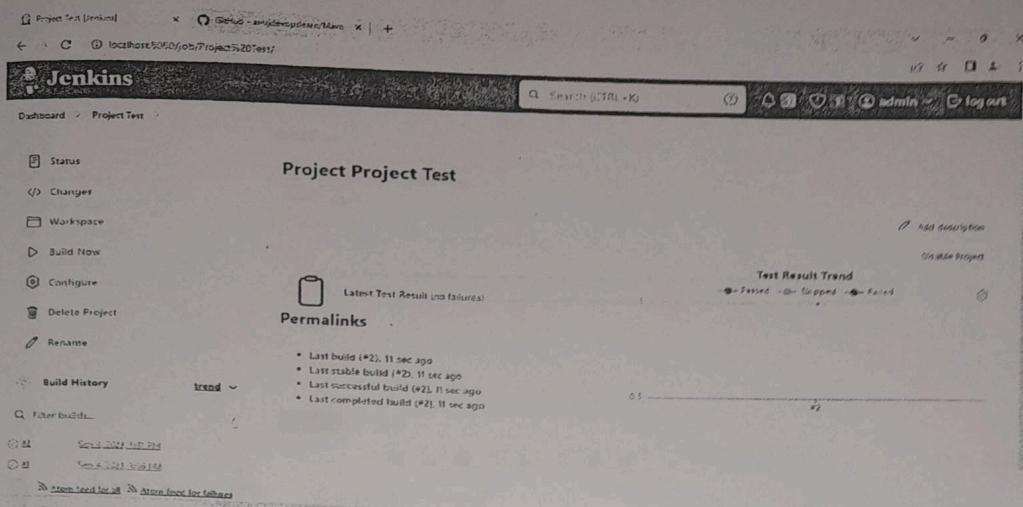
14) open the XML file

The screenshot shows a browser window displaying the XML test report for "com.geekcap.vmturbo.ThingTest". The page title is "ThingTest". The content includes a summary: "Tests run: 1, Failures: 0, Errors: 0, Skipped: 0. Time elapsed: 0.030 s ... in com.geekcap.vmturbo.ThingTest".

15) open the configuration & enter XML file

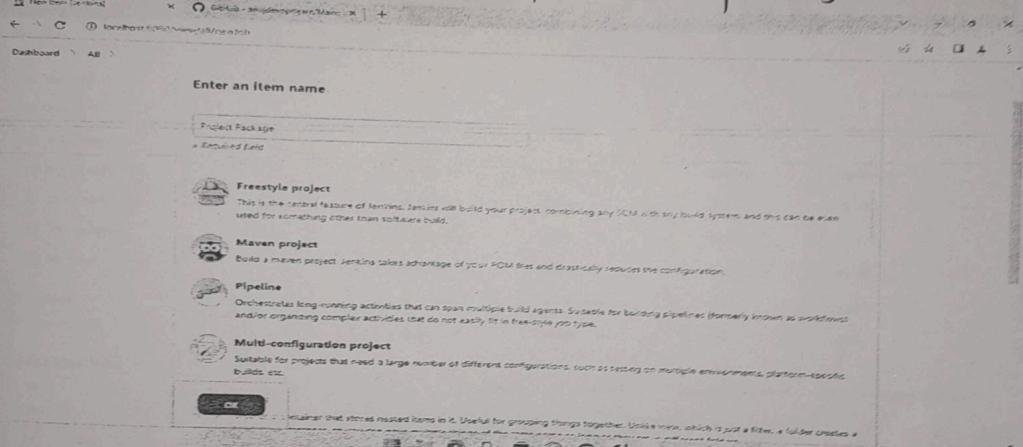
The screenshot shows the Jenkins configuration screen for "Project Test Config [Jenkins]". The left sidebar has sections for General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. Under "Post-build Actions", there's a section for "Publish JUnit test result report". It shows a dropdown menu set to "Test report XMLs" and a field containing "target/referer/report.xml". There are also options for "Keep all the properties" and "Health report amplification factor". At the bottom, there are "Save" and "Apply" buttons.

16) go to project test status



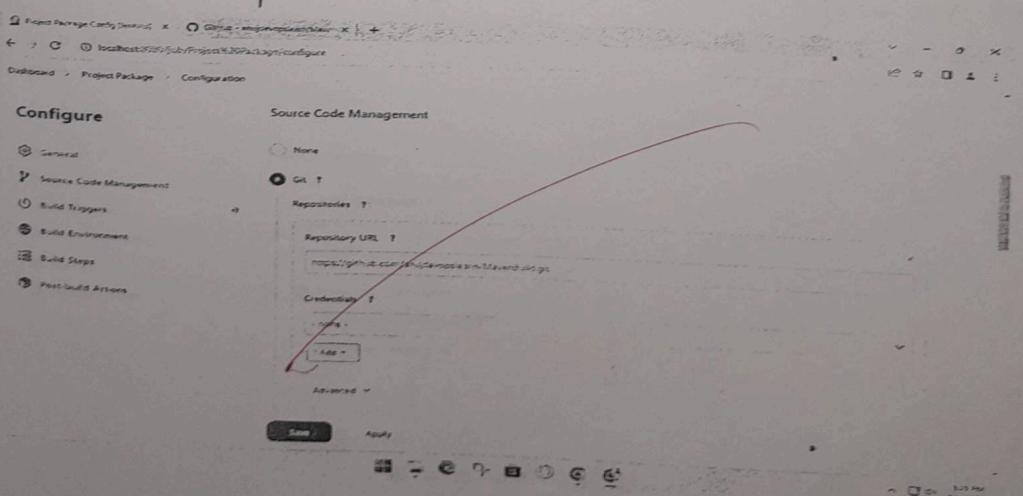
The screenshot shows the Jenkins interface for the 'Project Test' job. On the left, there's a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, Rename, and Build History. The main area is titled 'Project Test' and displays a 'Latest Test Result (no failures)' icon. Below it is a 'Test Result Trend' chart showing a single data point from 11 seconds ago. A 'Permalinks' section lists recent builds. At the bottom, there are links for Atom feed and RSS feed.

17) create new item named project packages.



The screenshot shows the 'New Item' creation screen. The title bar says 'New Item (General)'. The main area is titled 'Enter an item name' with a placeholder 'Project Package'. Below it, there's a 'Required Fields' section. Under 'Project Type', there are four options: 'Freestyle project' (selected), 'Maven project', 'Pipeline', and 'Multi-configuration project'. Each option has a brief description. At the bottom right is a 'Create' button.

18) Enter the repo URL



The screenshot shows the 'Project Package Configuration' screen. The title bar says 'Project Package Configuration'. The left sidebar has sections: General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area is titled 'Source Code Management' and shows a 'None' option. Below it is a 'Git' section with a 'Repositories' tab. A red arrow points from the 'Configure' section of the sidebar to the 'Repository URL' field, which contains 'https://github.com/username/repo.git'. There are 'Save' and 'Apply' buttons at the bottom.

19) set goals as packages

The screenshot shows the Jenkins configuration interface for a project named "Project Package". Under the "Build Steps" section, there is one step: "Invoke top-level Maven targets" with "Goals" set to "package". Other tabs like General, Source Code Management, Build Triggers, Build Environment, and Post-build Actions are visible on the left.

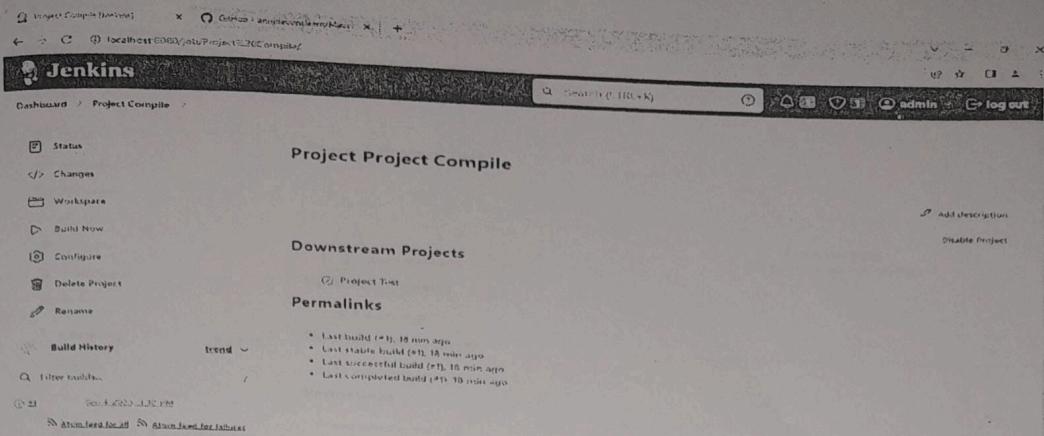
20) go back to dashboard

The dashboard lists several projects: MavenProject, Project Compile, Project Package, Project Test, and Project. Project Package has the most recent success at 9:4 sec ago. Project Test has the longest duration at 4 min 24 sec. Project has the shortest duration at 83 ms.

21) Go to configuration of set project to build project test

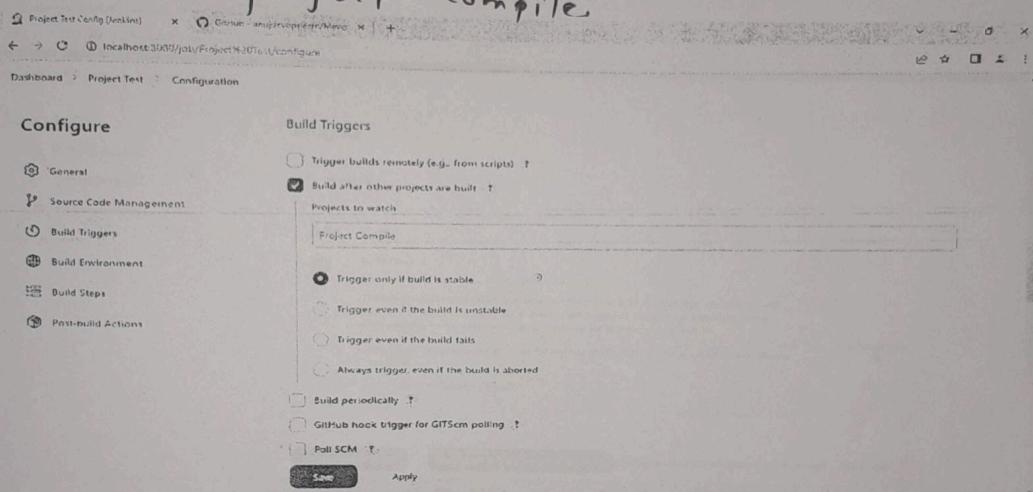
The configuration screen for "Project Compile" shows a "Post-build Actions" section with a "Build other projects" step. It is configured to trigger only if the build is stable. The "Projects to build" dropdown is set to "Project Test". Other tabs like General, Source Code Management, Build Triggers, Build Environment, and Build Steps are visible on the left.

23) go to status



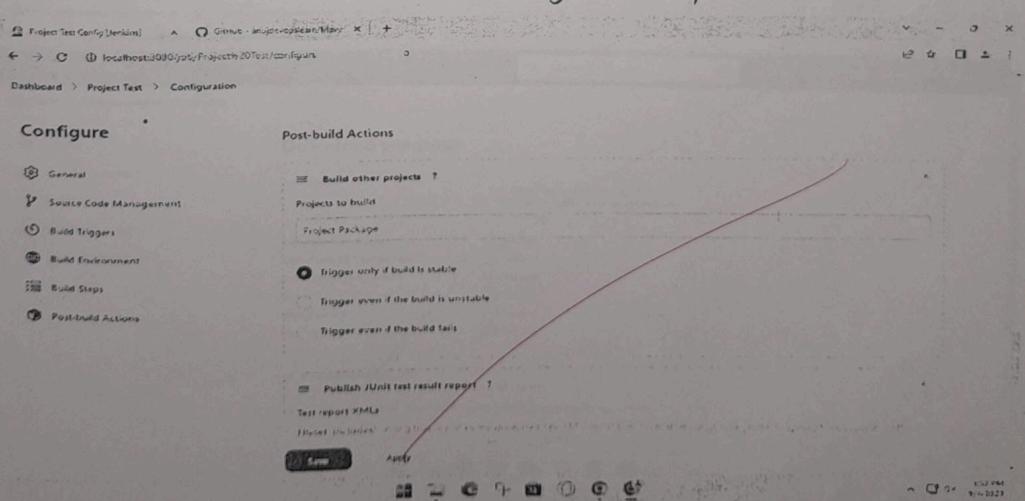
The screenshot shows the Jenkins interface for the 'Project Compile' project. On the left, there's a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. Below that is a 'Build History' section with a dropdown menu. A 'Downstream Projects' section lists 'Project Test' with a 'Project Test' link. To the right, there's a 'Permalinks' section with a 'Last build' link. At the top right, there are 'Add description' and 'Disable project' buttons.

23) go to project test configuration & set project to watch as project compile



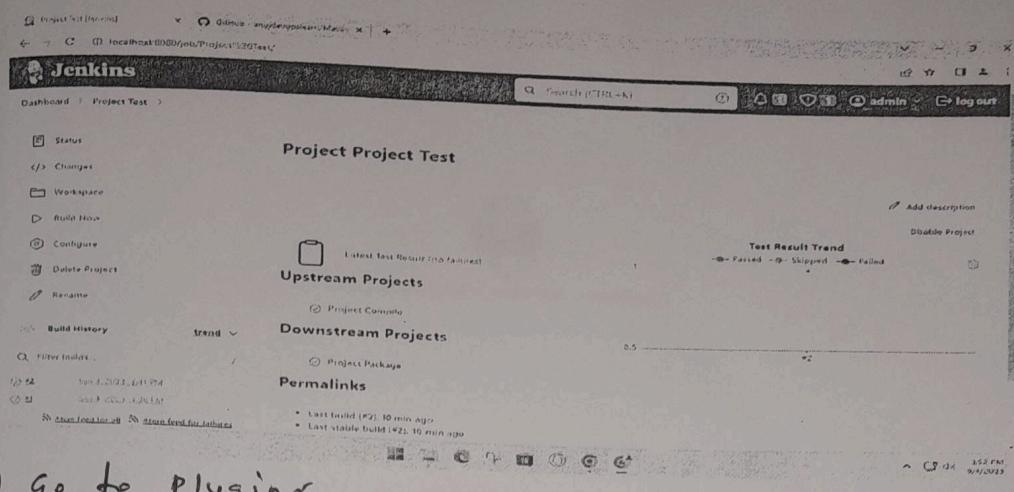
The screenshot shows the 'Configuration' page for the 'Project Test' project in Jenkins. On the left, there's a sidebar with General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Build Triggers' section is expanded, showing several trigger options: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built' (which is checked), 'Projects to watch' (set to 'Project Compile'), and four radio button options for triggering based on build stability. There are also checkboxes for 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. At the bottom are 'Save' and 'Apply' buttons.

24) project to build is project package.



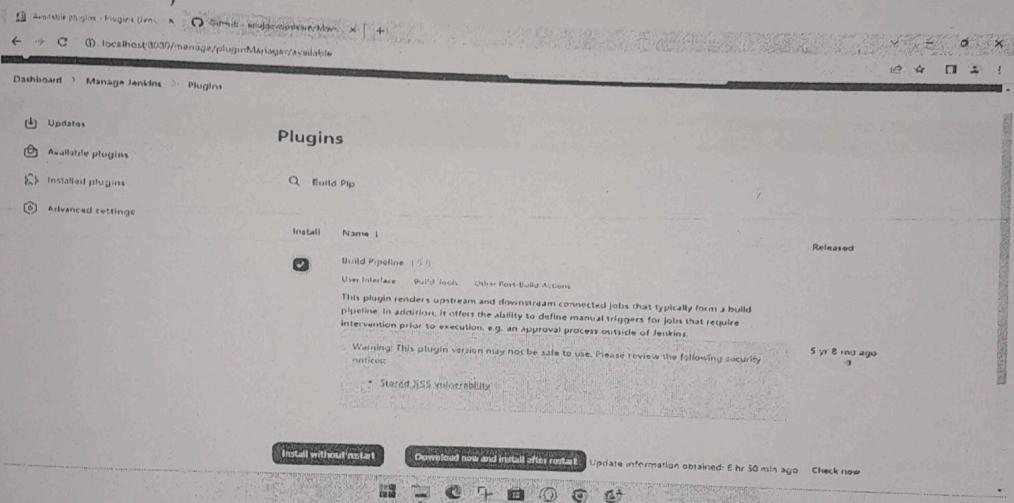
The screenshot shows the 'Configuration' page for the 'Project Test' project in Jenkins, specifically focusing on the 'Post-build Actions' section. On the left, there's a sidebar with General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is expanded, showing two main sections: 'Build other projects' and 'Publish JUnit test result report'. Under 'Build other projects', there's a 'Projects to build' field set to 'Project Package'. Under 'Publish JUnit test result report', there's a 'Test report XMLs' field with a 'Selected' dropdown. At the bottom are 'Save' and 'Apply' buttons.

25) Go to status



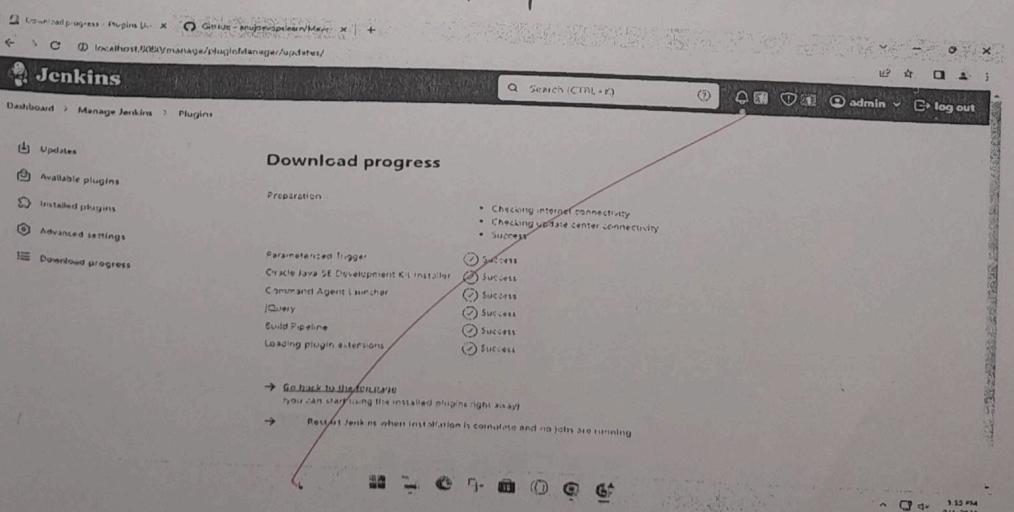
Screenshot of the Jenkins Project Test status page. The page shows the project's status, build history, upstream projects, downstream projects, and permalinks. A note indicates no stable build found for failure.

26) Go to Plugins



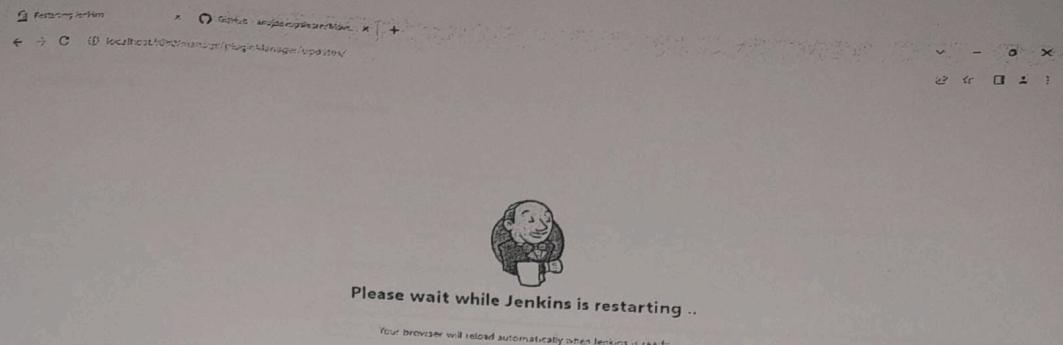
Screenshot of the Jenkins Manage Jenkins > Plugins page. It lists available plugins, including Build Pipeline, and provides options to install or download them.

27) Install without restart



Screenshot of the Jenkins Manage Jenkins > Plugins > Download progress page. It shows the download progress of various plugins, with most marked as successful.

29) Jenkins is restarting



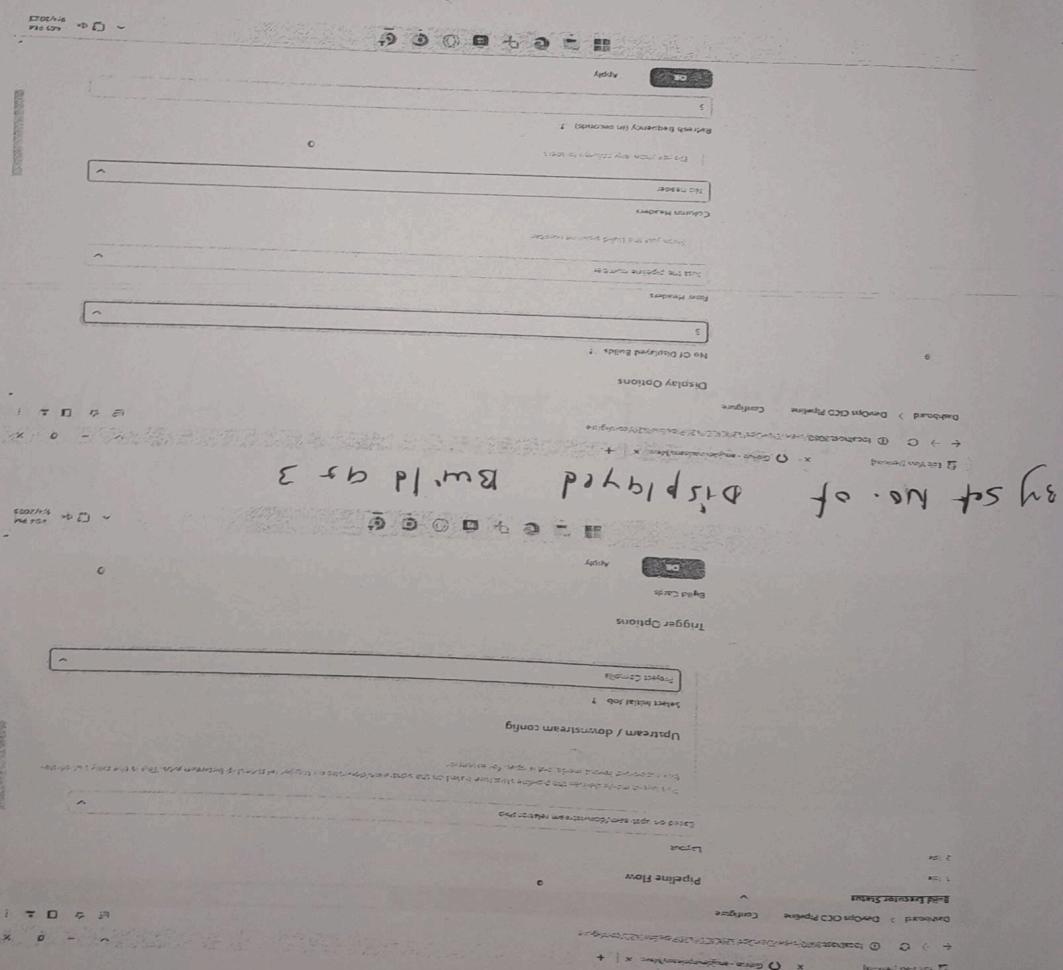
29) go to dashboard

A screenshot of the Jenkins dashboard. On the left, there's a sidebar with links like "New Item", "People", "Build History", "Project Relationship", "Check File Fingerprint", "Manage Jenkins", and "My Views". The main area shows five projects: "MavenProject", "Project Compile", "Project Package", "Project Test", and "Project". Each project has a status icon, a name, the last success time, the last failure time, and the last duration.

30) Name the view as DevOps CI/CD pipeline.

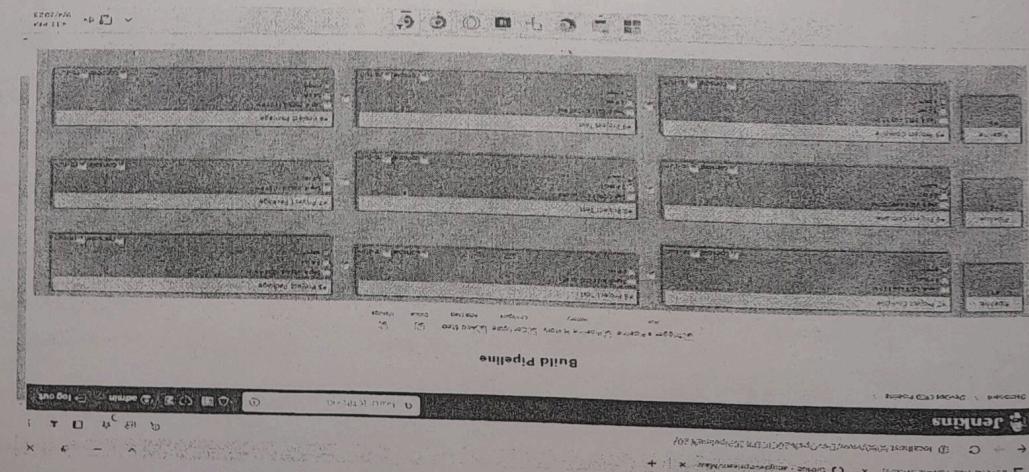
A screenshot of the "New view" creation page in Jenkins. The left sidebar is identical to the dashboard. The main area has a title "New view" and a text input field containing "DevOps CI/CD Pipeline". Below it, there's a section titled "Type" with a radio button selected for "Build Pipeline View". A red arrow points from the handwritten note "30) Name the view as DevOps CI/CD pipeline." to this section. There are also descriptions for "List View" and "My View". At the bottom, there's a "Create" button.

81) go to configurtion of unit project compile



11c

a pipeline to test and deploy application.



33) Pipeline is built