**SEPM EXPERIMENT 5**

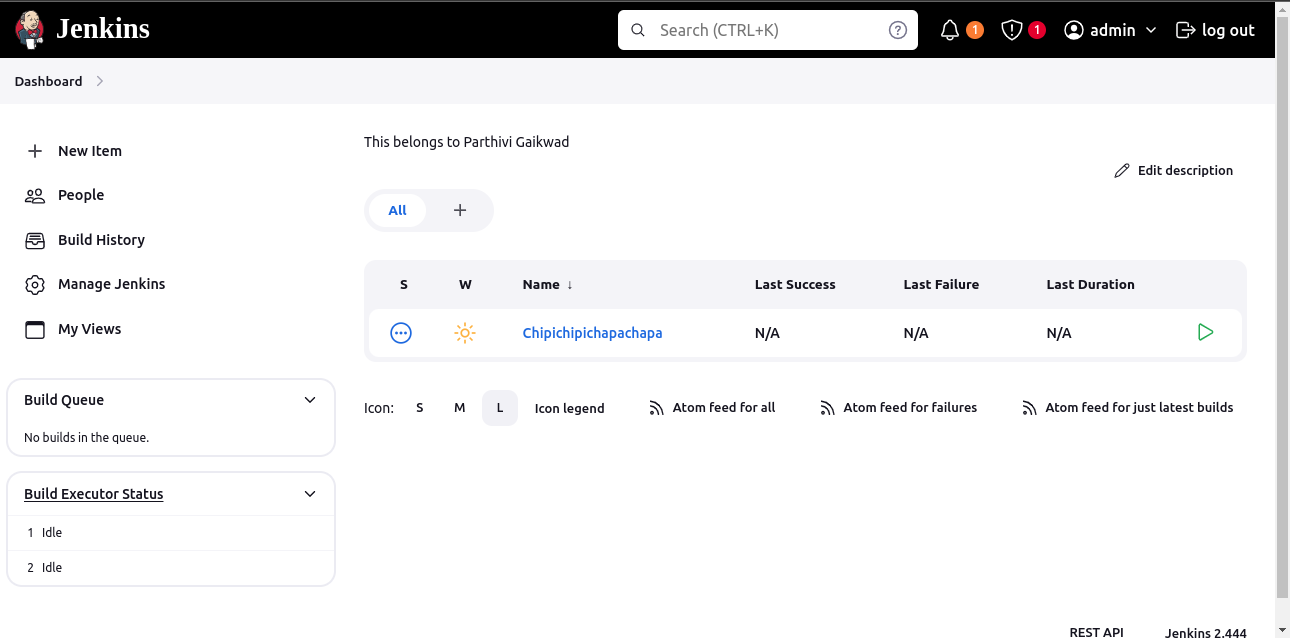
***Aim:*** To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

***Theory:***

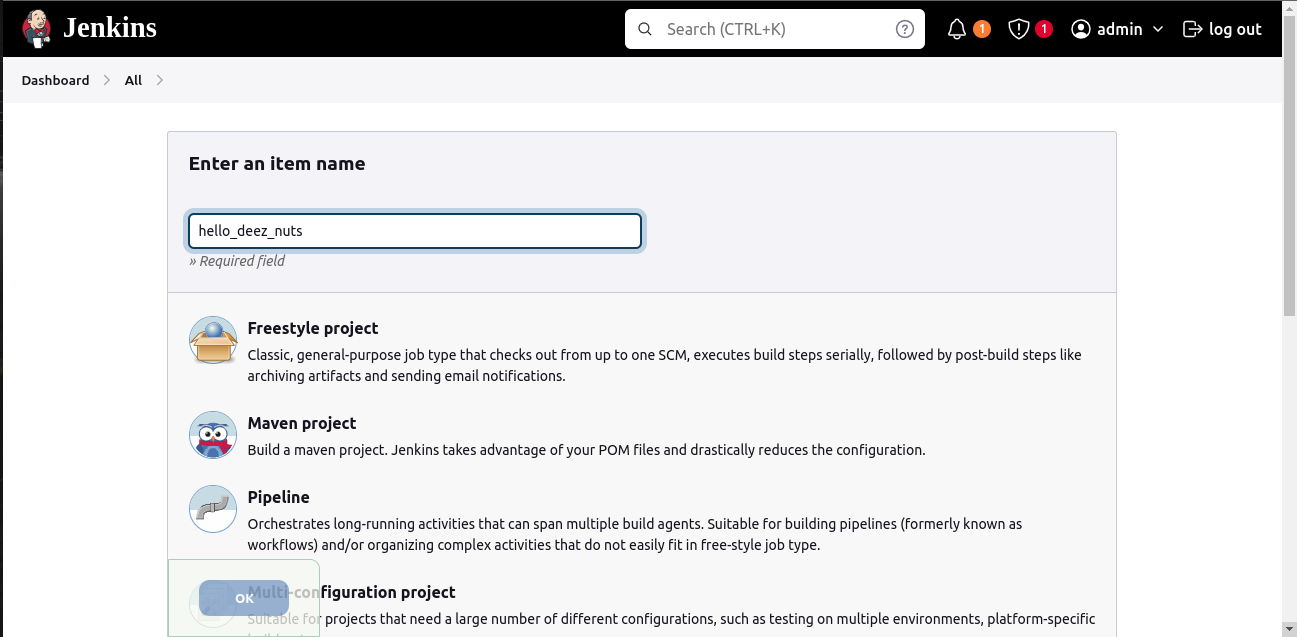
1. A pipeline of jobs, in the context of software development and automation, refers to a series of interconnected tasks or stages that are executed sequentially to automate the software delivery process.
2. Each job in the pipeline performs a specific set of actions, such as compiling code, running tests, deploying applications, and generating reports.
3. The output of one job serves as the input for the next job in the pipeline, creating a continuous flow of actions that streamline the software development lifecycle.
4. Benefits of Pipeline of Jobs:
5. Automation: Enables the automation of the software delivery process, reducing manual intervention and human errors.
6. Continuous Integration: Facilitates continuous integration by building, testing, and deploying applications in a streamlined manner.
7. Visibility and Traceability: Provides visibility into the status of each job and stage, offering traceability and insights into the build process.
8. Efficiency: Helps in improving efficiency by optimizing the build, test, and deployment workflows.
9. Consistency: Ensures consistency in the software delivery process, leading to more reliable releases.
10. Key Components of a Pipeline of Jobs:
11. Jobs: Each individual task or stage in the pipeline is represented by a job. These jobs can include building the application, running tests, deploying to servers, etc.
12. Stages: Jobs are organized into stages, which represent logical divisions in the pipeline workflow. Common stages include build, test, deploy, and post-build actions.
13. Dependencies: Jobs within a pipeline can have dependencies on each other, ensuring that a job runs only after its dependent job(s) have successfully completed.
14. Triggers: Pipelines can be triggered manually, on a schedule, or automatically based on events such as code commits to a version control system.
15. Parallel Execution: Jobs within a stage or across stages can run in parallel to optimize the pipeline's performance.
16. Visualization: Jenkins provides visual representations of pipeline status and progress, allowing users to monitor the execution of jobs and stages.

***Output:***

Step 1: Click on New Item

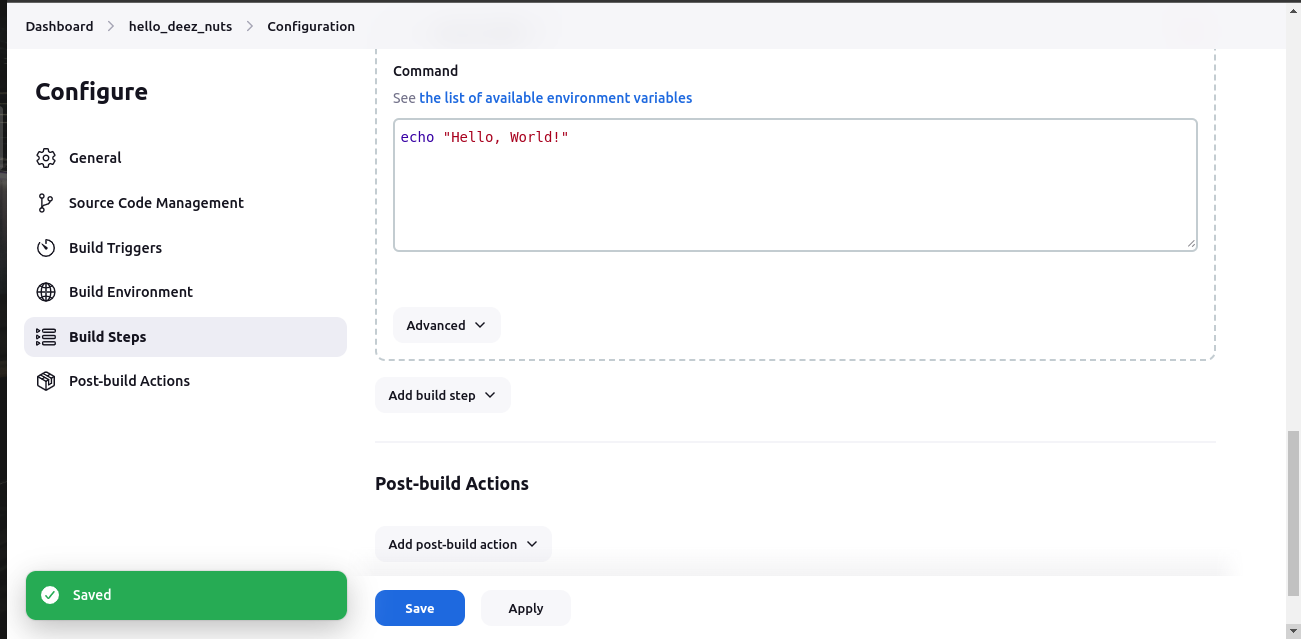


Step 2: Create a freestyle project

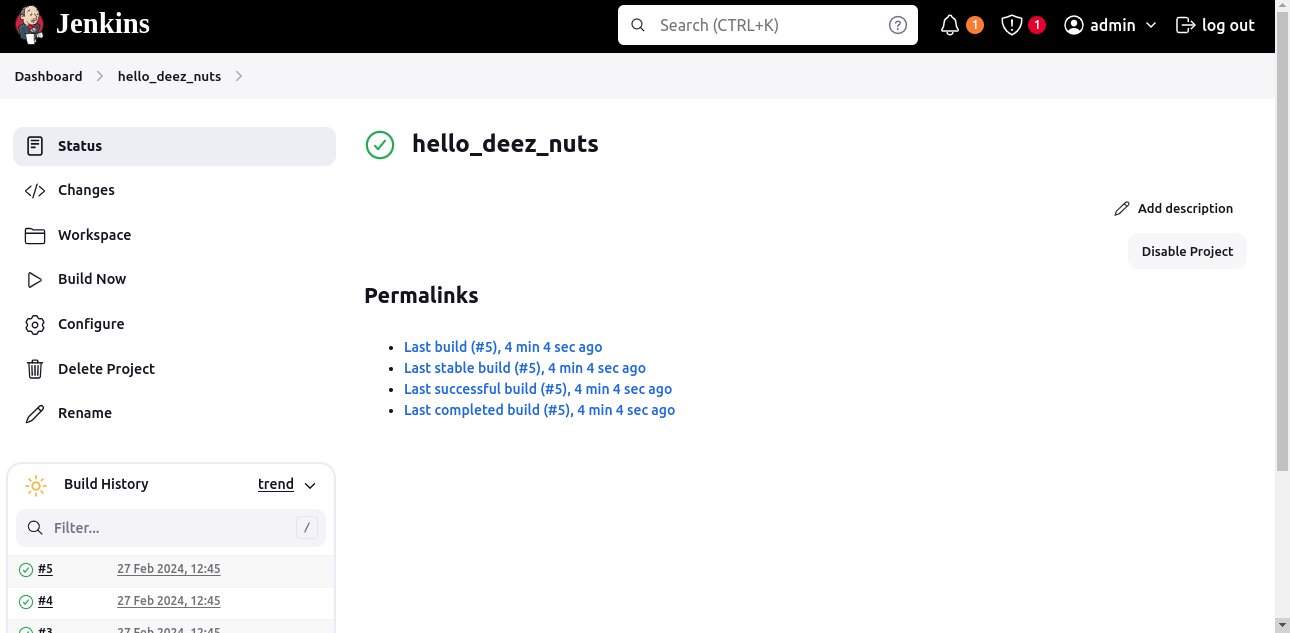


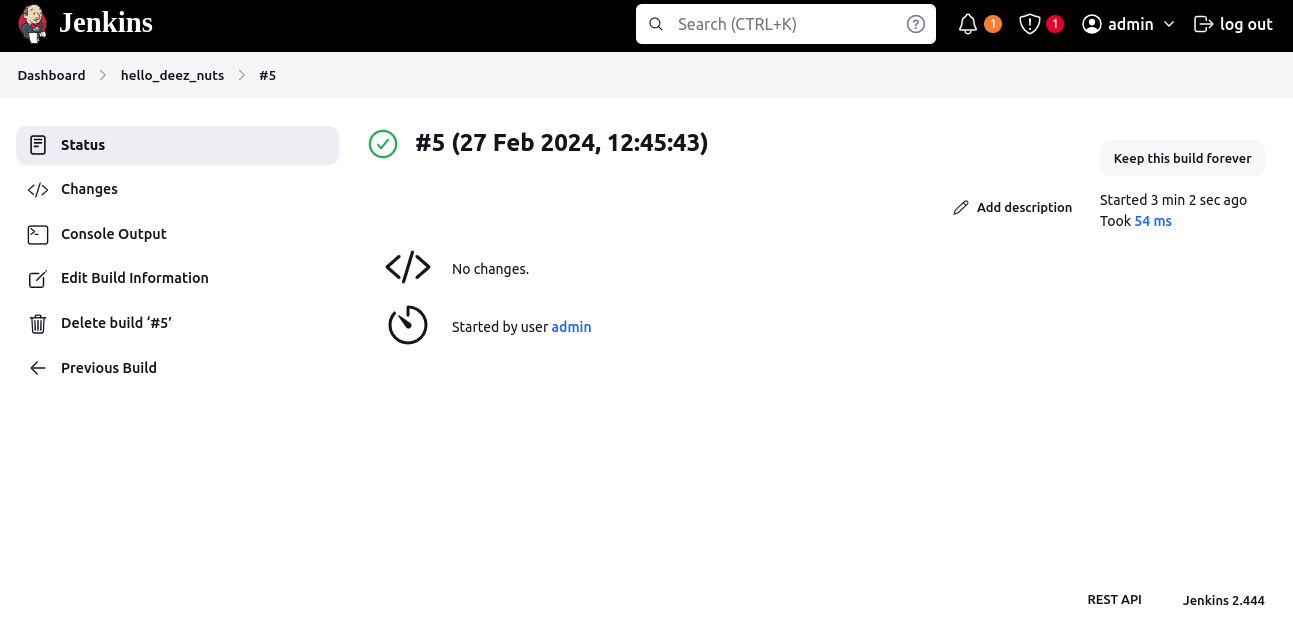
Step 3: Go to Configuration >> Build Steps >> Add build step >> Execute Shell

Write code and click on “Save” and “Apply”

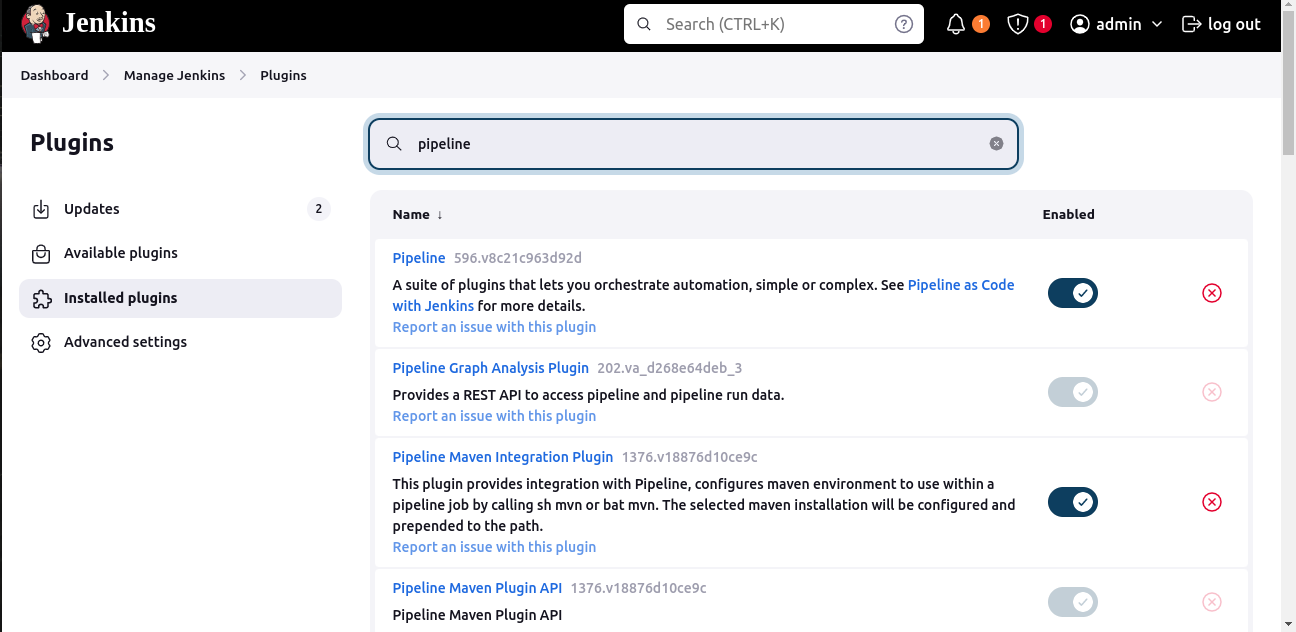


Step 4: Check the status of the job, by clicking on Permalinks

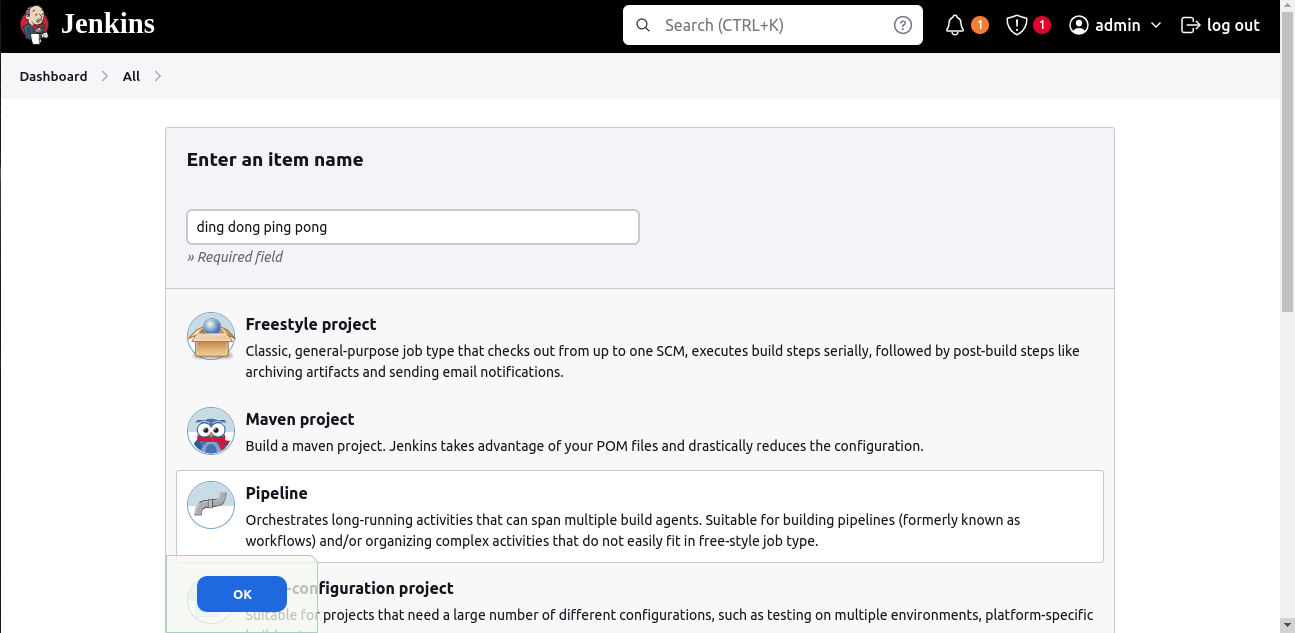




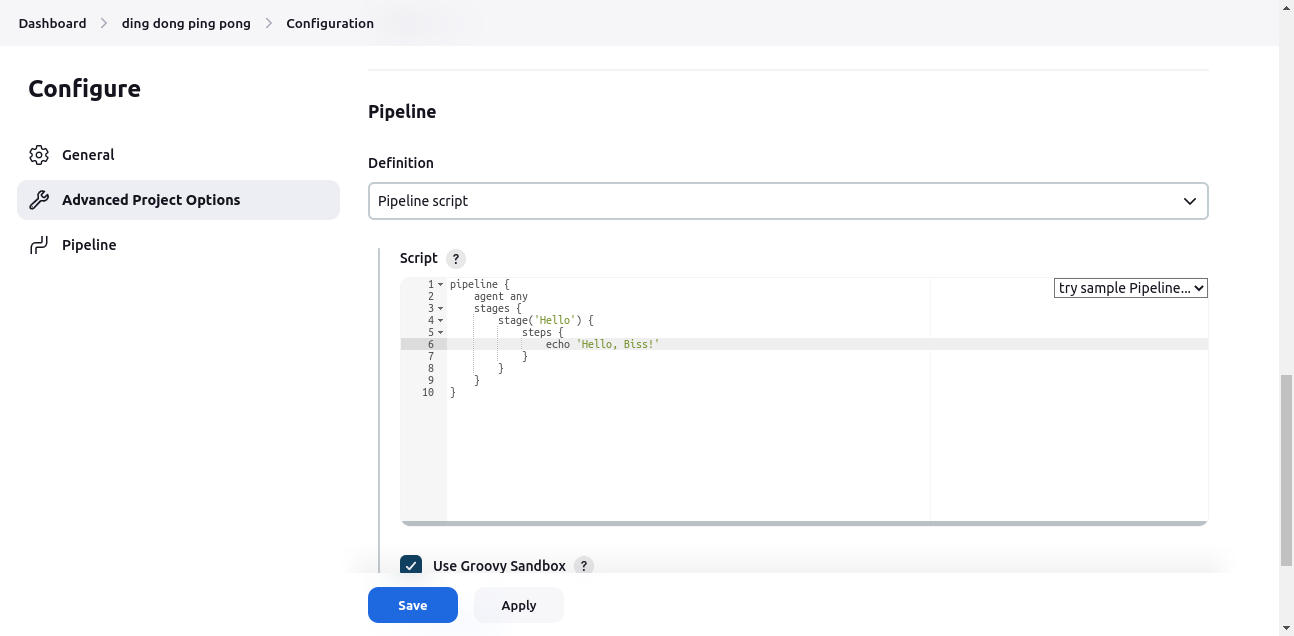
Step 5: For creating Manage Jenkins >> Plugins >> Available Plugins >> Search “Pipleline” >> Install



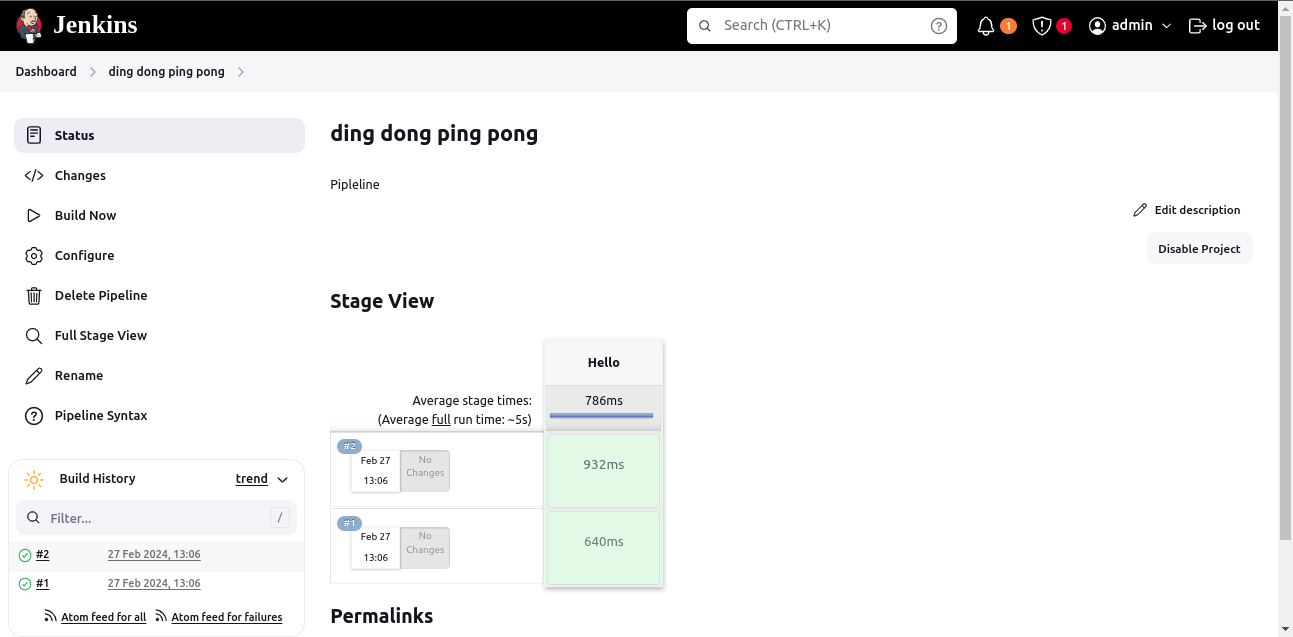
Step 6: After installation, go to Dashboard and create a new item, as mentioned in Step 1. Make sure to select Pipeline as type of project, not Freestyle.

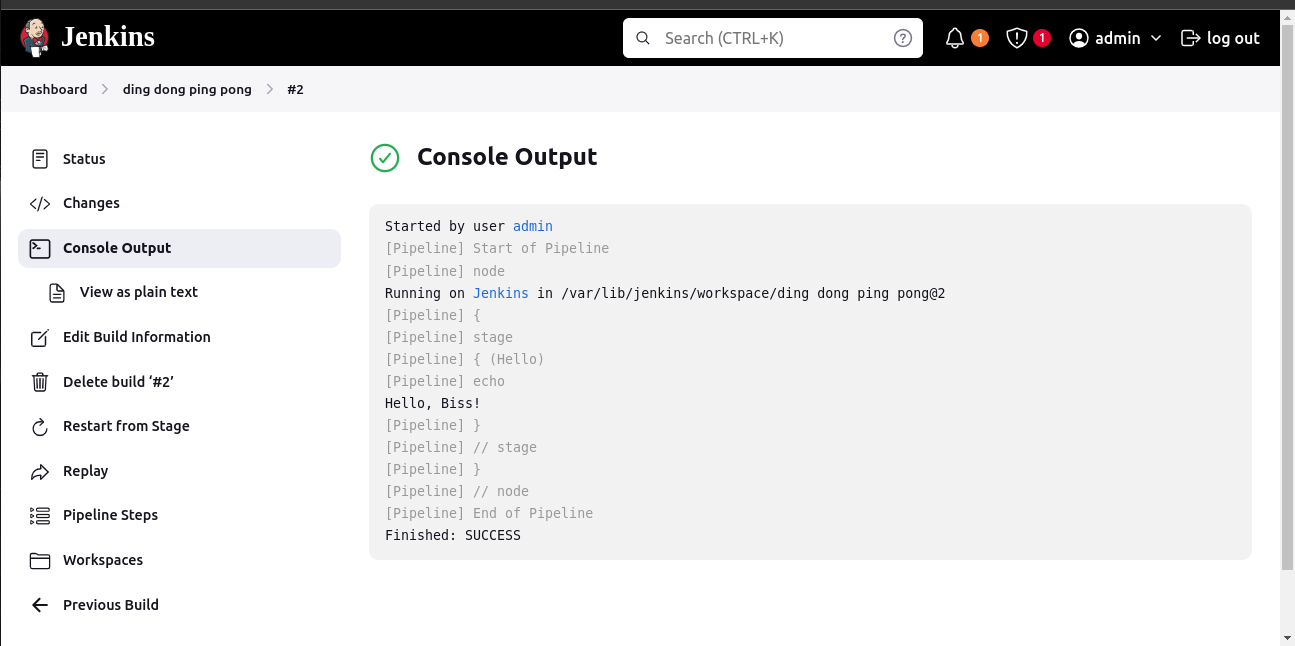


Step 7: After selecting Pipleline and clicking OK, Scroll down to Advanced Project options and insert your own script for Pipleline. Click Apply and Save.



Step 8: Go to build now and wait till you set the status of your pipeline. You can go to Console Output to see the output.





**Conclusion:** By creating a pipeline script, we automated the entire CI/CD process, ensuring efficient testing and deployment of the application.