# **Experiment-4**

Ashish Patil

D11AD 46

**Aim**:-To understand Continuous Integration, install and configure Jenkins to

setup a build Job.

**Theory:-**

Jenkins is a popular open-source automation server used primarily for continuous integration and continuous delivery (CI/CD) pipelines. It allows developers to automate the process of building, testing, and deploying software applications. Below are some key theoretical concepts related to Jenkins:

**1. Continuous Integration (CI):**

CI is a software development practice where team members integrate their code changes into a shared repository frequently, preferably several times a day. Each integration is verified by automated builds and tests, allowing teams to detect and fix problems early in the development process. Jenkins plays a crucial role in CI by automatically triggering builds whenever new code is committed to the repository.

**2. Continuous Delivery (CD):**

CD extends CI by automating the deployment process, allowing teams to release software updates quickly, reliably, and frequently. Jenkins facilitates CD by automating various tasks such as compiling source code, running tests, and deploying applications to different environments (e.g., development, staging, production).

**3. Pipeline as Code:**

Jenkins Pipeline is a suite of plugins that enables teams to define their CI/CD pipelines as code, allowing them to version control, review, and iterate on pipeline definitions just like any other software code. This approach brings several benefits, including better visibility, repeatability, and maintainability of pipelines.

**4. Master-Slave Architecture:**

Jenkins follows a master-slave architecture where the master server coordinates and schedules build jobs, while the slave nodes execute these jobs. This distributed architecture allows Jenkins to scale horizontally by adding additional slave nodes to handle increased workload and parallelize builds across multiple machines.

**5. Plugins:**

Jenkins provides a vast ecosystem of plugins that extend its functionality and integrate with various third-party tools and services. Plugins cover a wide range of functionalities such as version control systems (e.g., Git, SVN), build tools (e.g., Maven, Gradle), testing frameworks, deployment platforms, and notification services.

**6. Job Configuration:**

In Jenkins, a job represents a single task, such as building a project or running a test suite. Job configuration involves specifying parameters such as source code repository, build triggers, build steps, post-build actions, and notification settings. Jenkins provides a web-based interface for configuring and managing jobs.

**7. Build Triggers:**

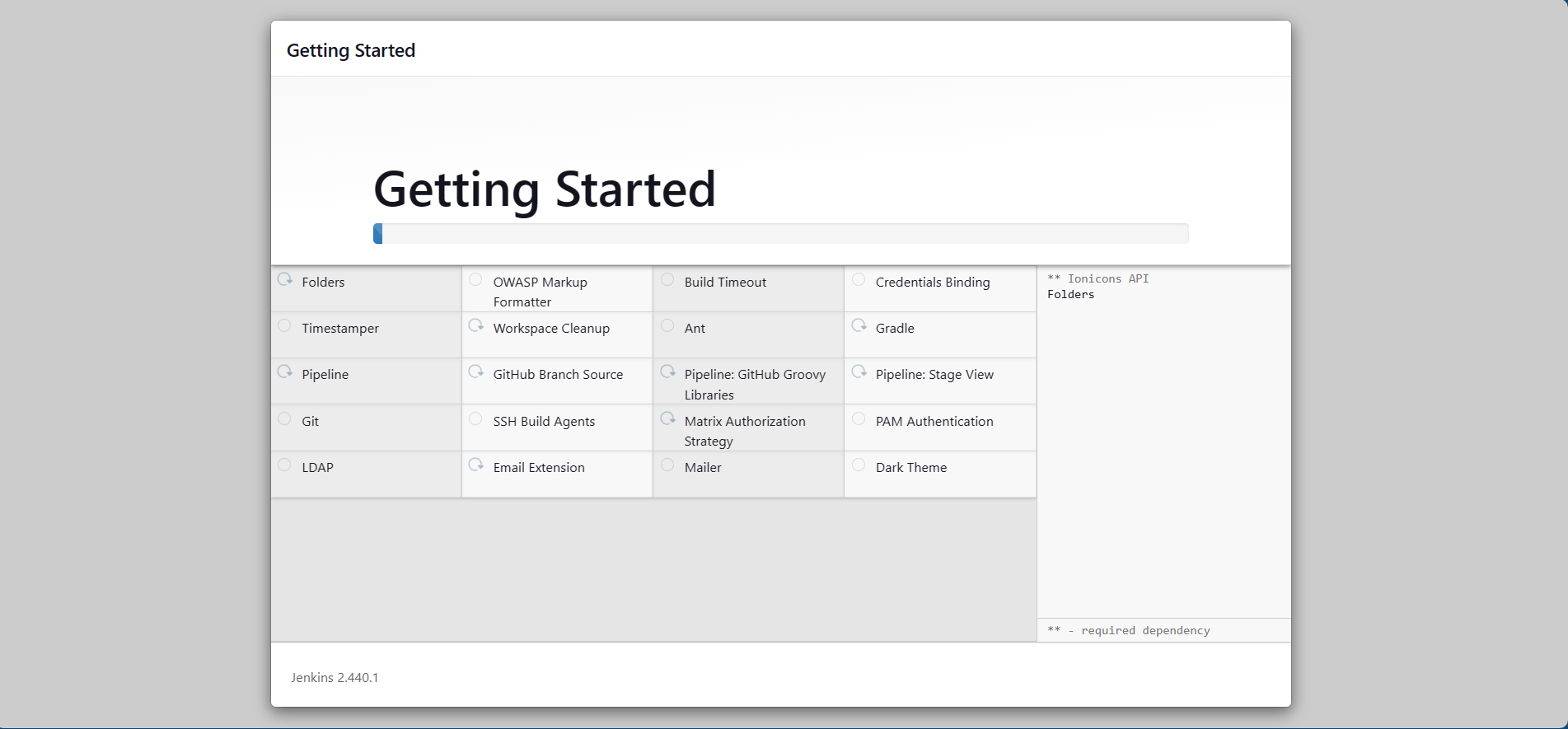
Jenkins supports various triggers to initiate build jobs automatically. Common triggers include SCM polling (polling the source code repository for changes), webhook notifications, scheduled builds, and manual triggers.

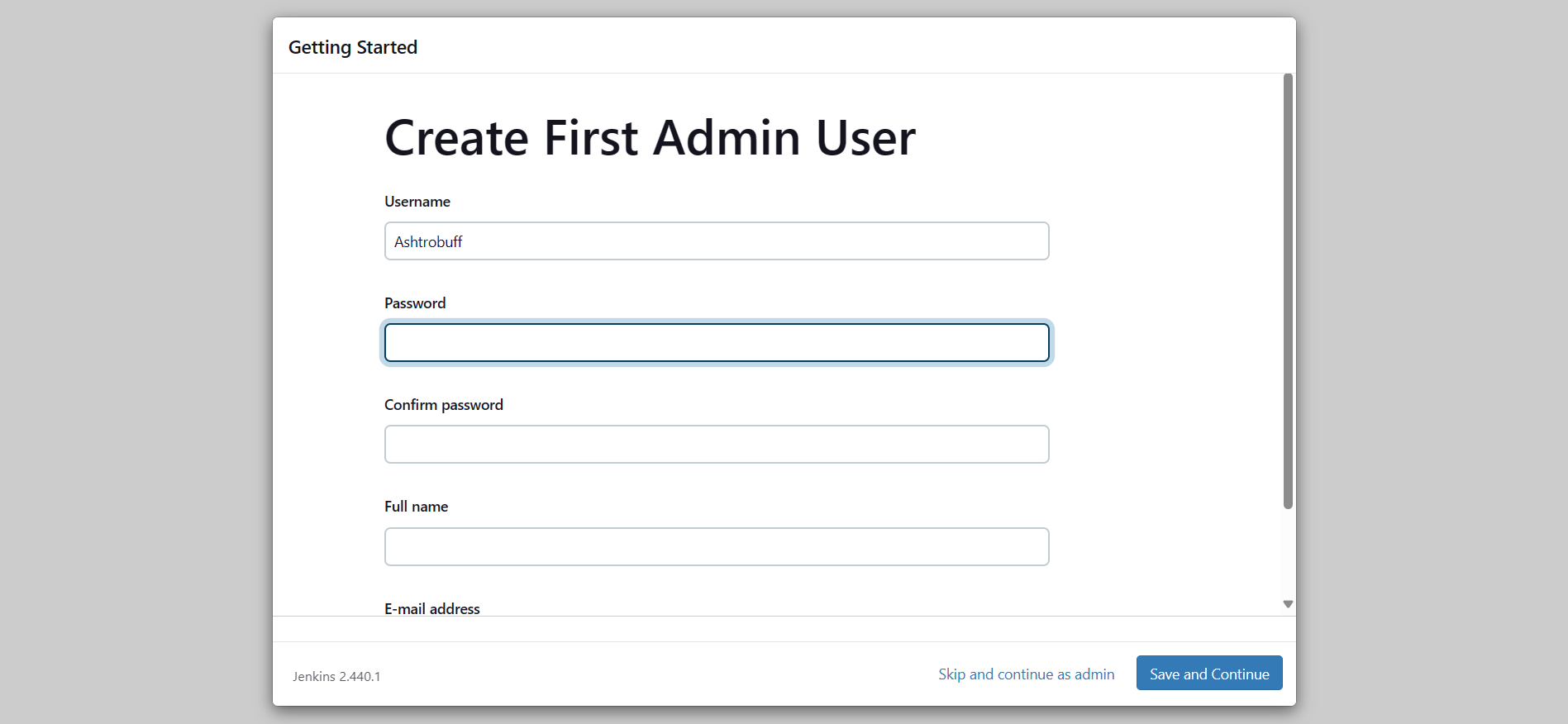
**8. Artifact Management:**

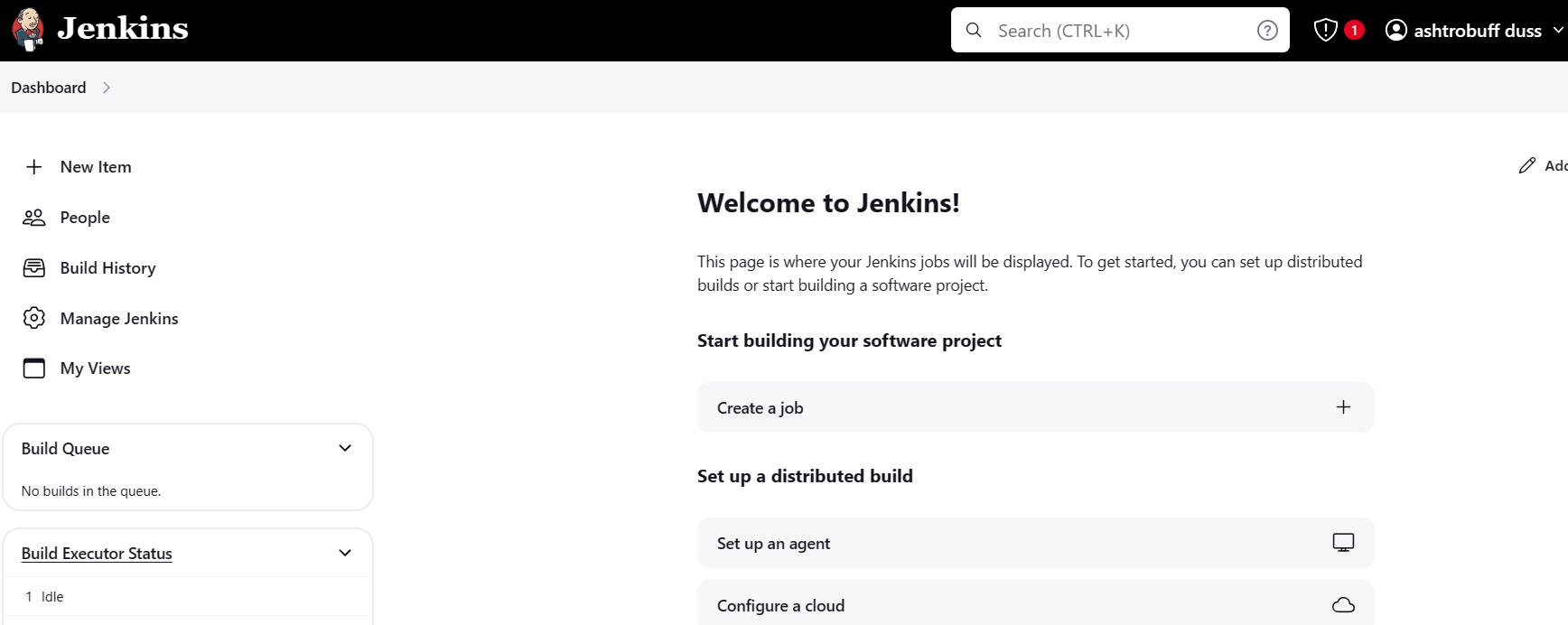
Jenkins allows teams to manage build artifacts (e.g., compiled binaries, test reports) produced during the build process. Artifacts can be archived, published, and retrieved from Jenkins, ensuring traceability and reproducibility of builds.

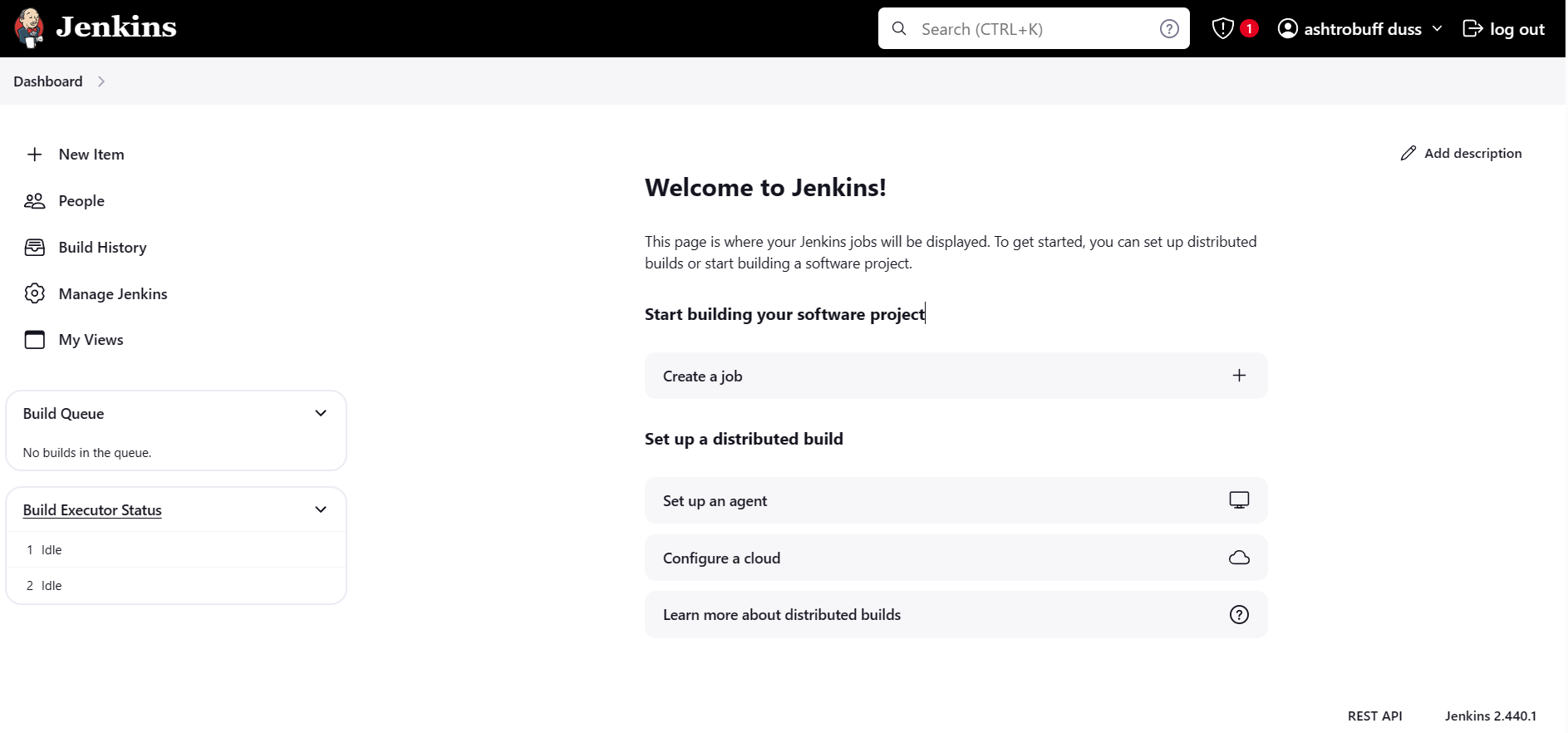
Understanding these theoretical concepts is crucial for effectively utilizing Jenkins in CI/CD workflows and optimizing the software development process.

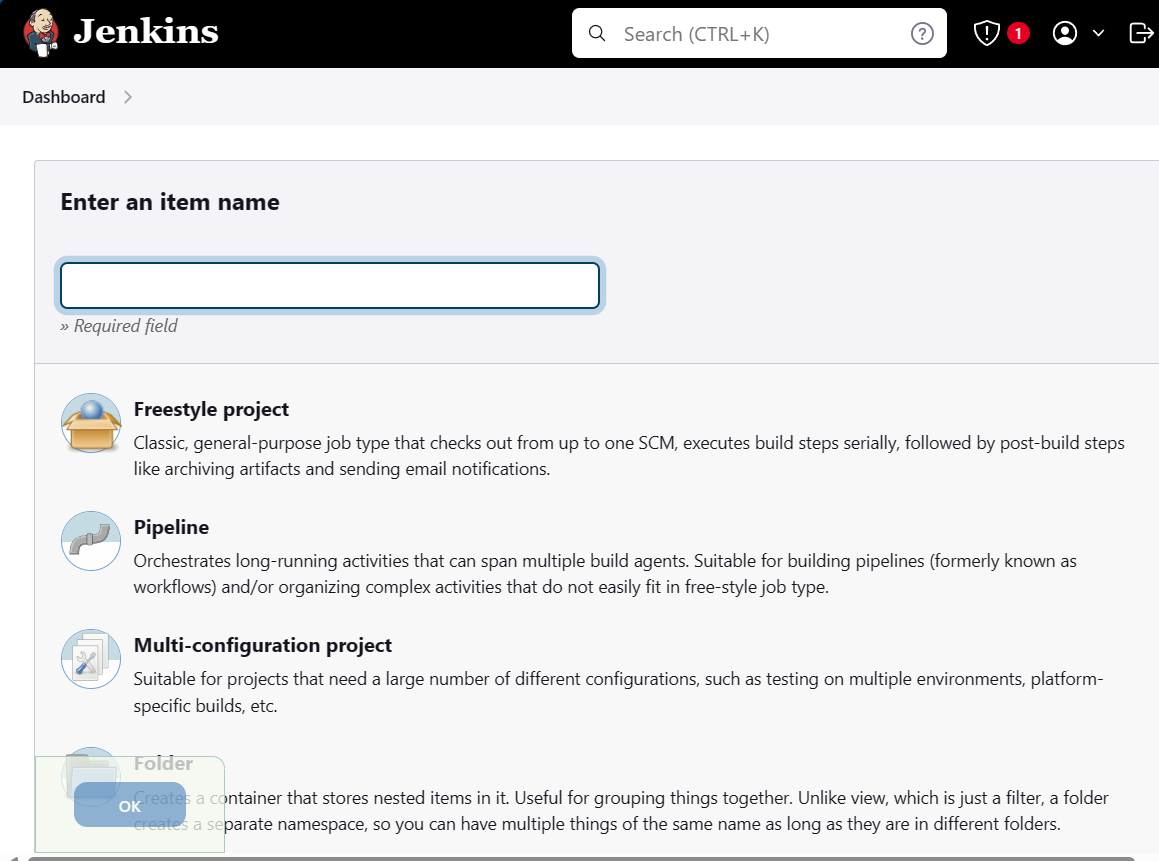
**Setting up a build job in jenkins:-**

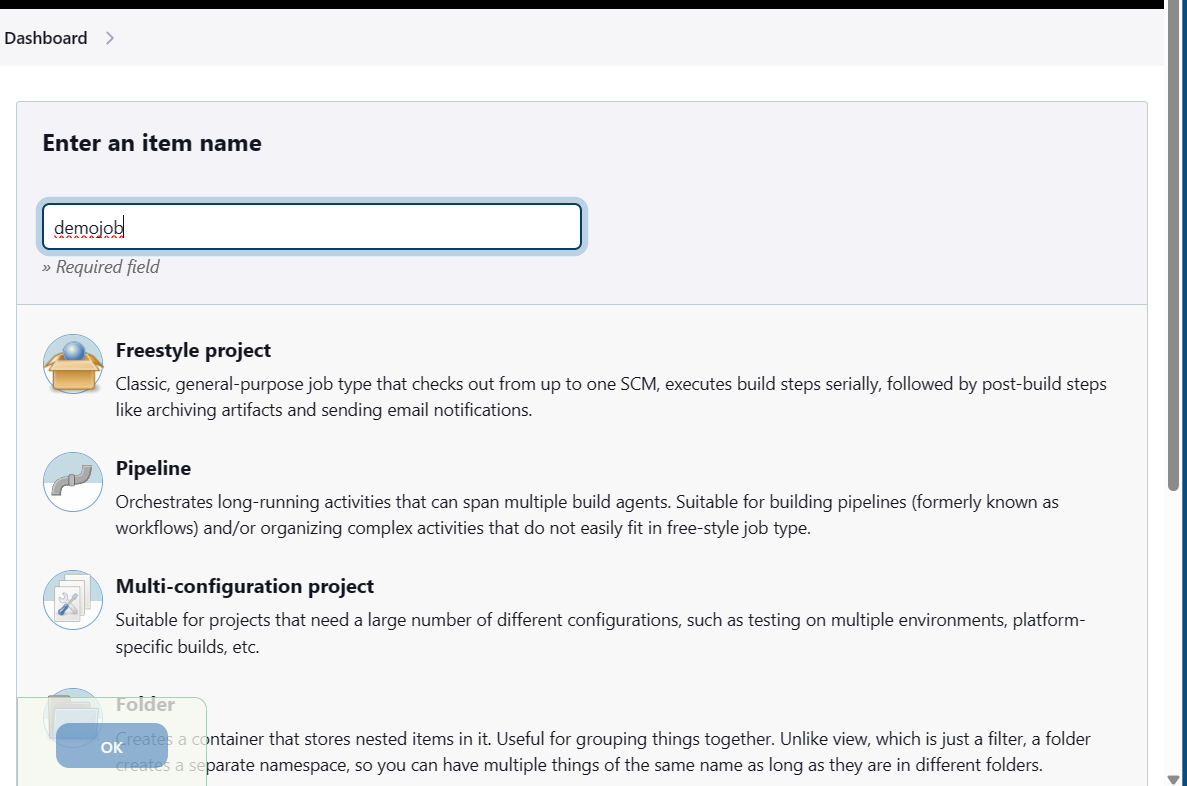
****

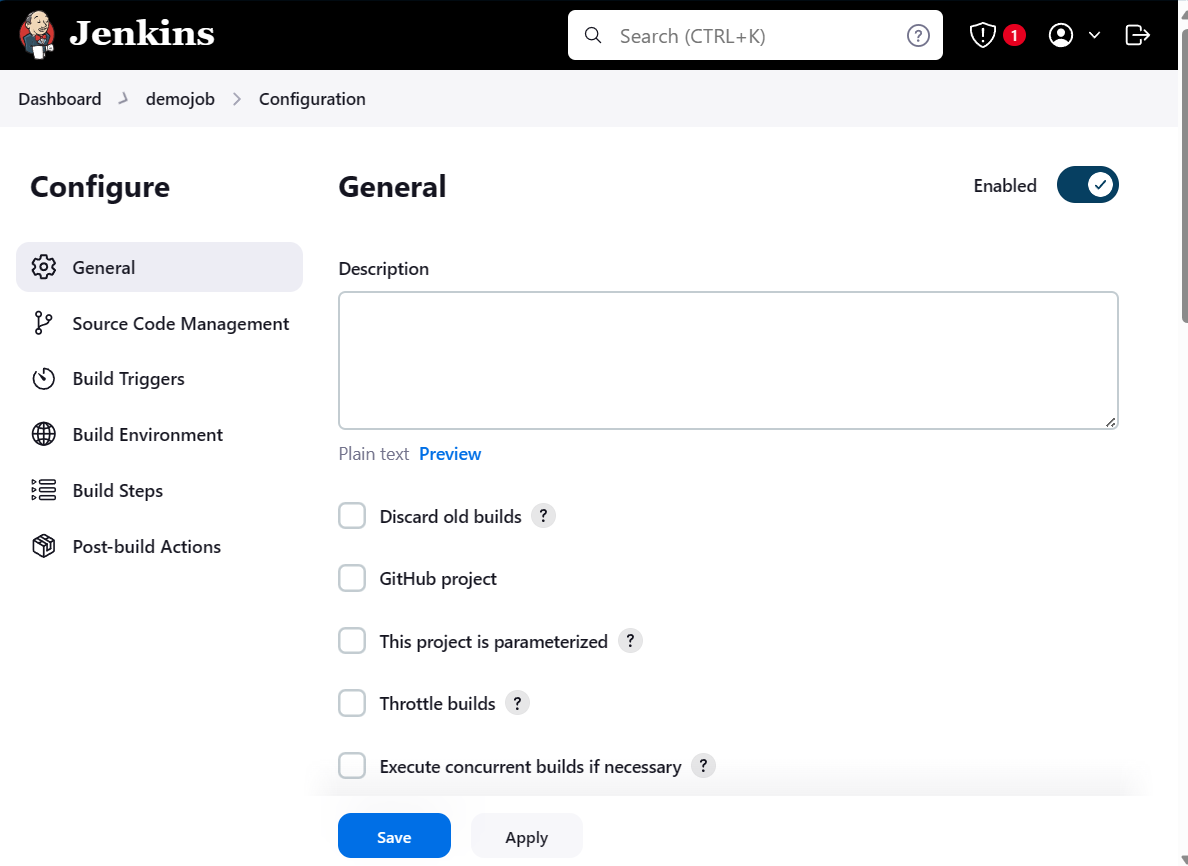
****

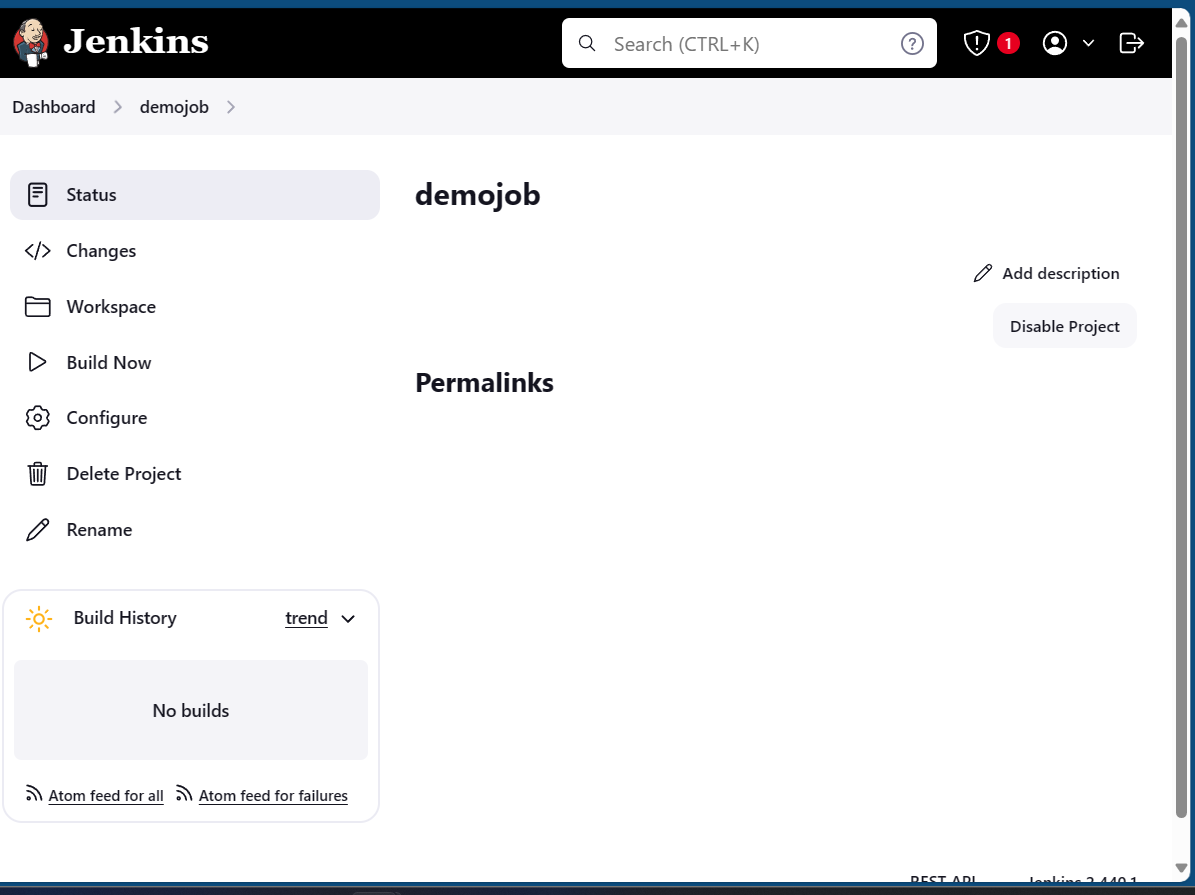
****

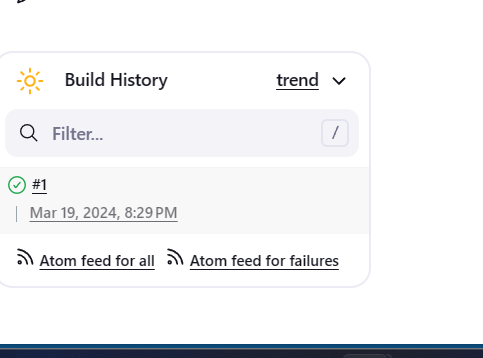
****

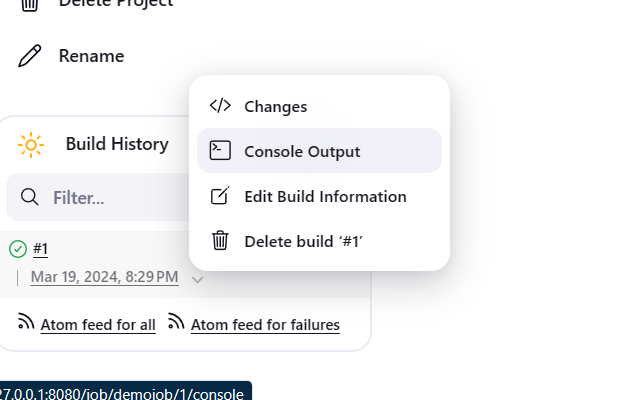
****

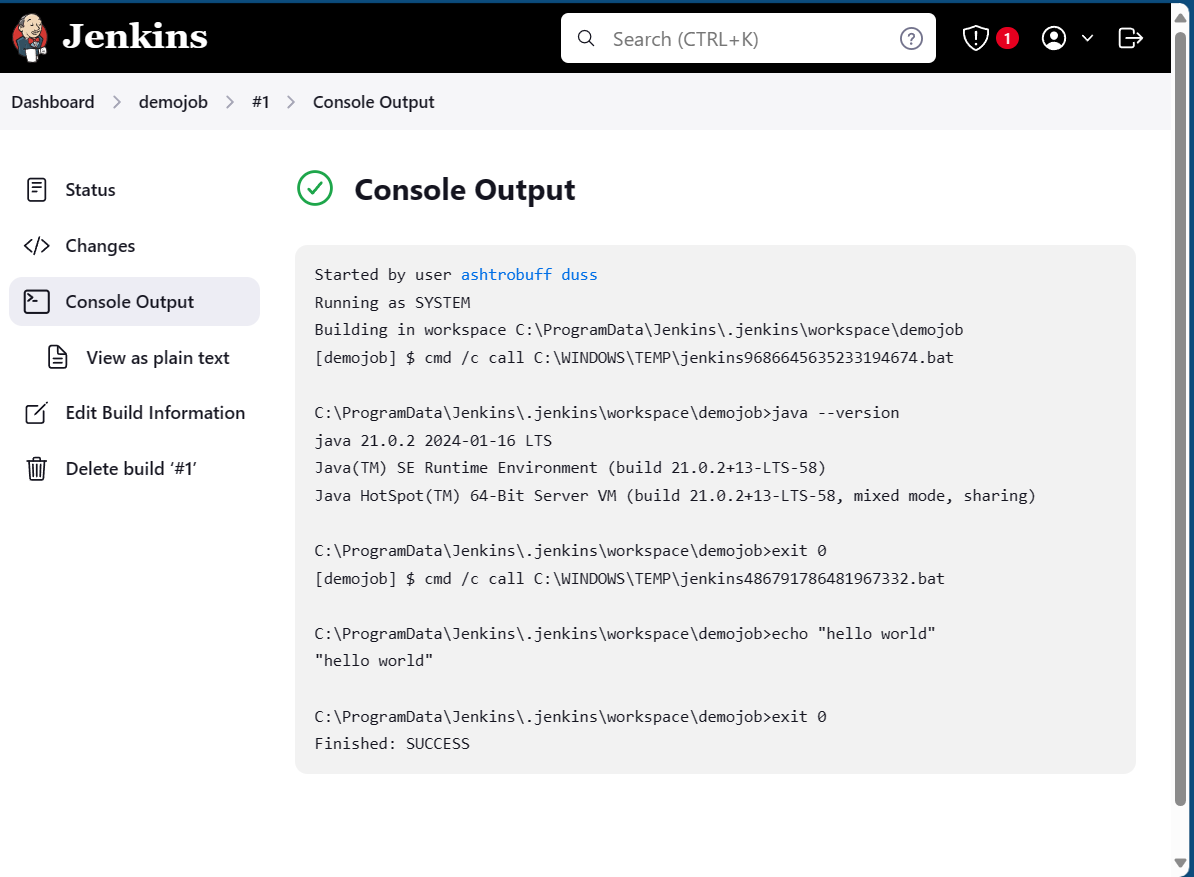
****

****

****

****

****

****

Conclusion : Thus we have successfully setup jenkins on a local machine and then setup our first build job using jenkins.