CREATING A SEQUENTIAL PIPELINE IN JENKINS

Managing Sequential Job Execution with User prompts

1. Continuous Integration and Continuous Delivery (CI/CD):

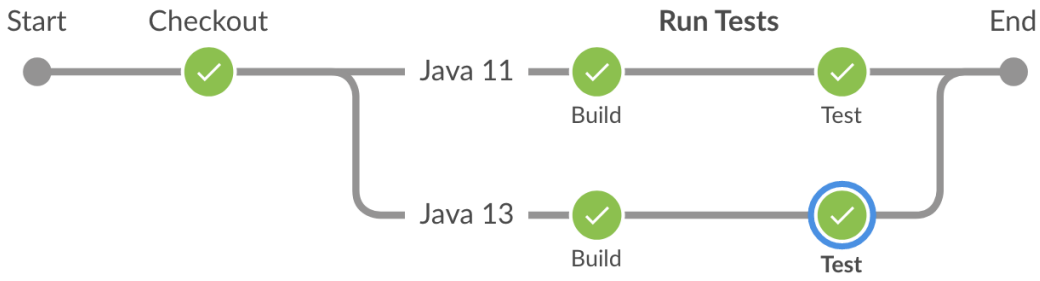
- **Continuous Integration (CI)** is a software development practice where developers frequently integrate code into a shared repository. Each integration is automatically verified by building the application and running automated tests.
- **Continuous Delivery (CD)** is an extension of CI, ensuring that the software can be reliably released at any time. It includes automated deployment processes, allowing code changes to be automatically tested and prepared for production release.

2. Jenkins as a CI/CD Tool:

- **Jenkins** is an open-source automation server used for building, testing, and deploying applications. It enables developers to automate the software development process, ensuring a fast and reliable workflow.
- **Pipelines** in Jenkins allow you to define your CI/CD process as code, enabling version control and easy sharing of the build and deployment process.

3. Jenkins Pipeline Syntax:

- **Declarative Pipelines** provide a simplified syntax for defining CI/CD processes. The structure is intuitive, allowing you to define stages, steps, and the overall flow of the pipeline.



Guideline for Creating a Sequential Jenkins Pipeline on Windows

---

Step 1: Install Jenkins on Windows

1. **Download Jenkins:**
    - Go to the Jenkins download page.
    - Download the Windows installer (e.g., jenkins.war or .msi file).
2. **Install Jenkins:**
    - If you downloaded the .msi file, double-click it to run the installer.
    - Follow the prompts to complete the installation.
    - If you downloaded the .war file, you can run it using the command line:

java -jar jenkins.war

3. **Start Jenkins:**
    - After installation, Jenkins will typically run at http://localhost:8080.
    - Open a web browser and navigate to this address.
4. **Unlock Jenkins:**
    - When prompted, retrieve the initial admin password from:

C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword

    - Copy the password and paste it into the web interface.
5. **Complete Setup:**
    - Follow the on-screen instructions to complete the setup, including installing suggested plugins and creating an admin user.

Step 2: Create Jobs for the Pipeline

1. **Create Job 1:**
    - From the Jenkins dashboard, click on **"New Item"**.
    - Enter the name (e.g., Job1) and select **"Freestyle project"**.
    - Click **"OK"**.
    - Under **Build**, choose **"Windows Batch Command"**.
    - Enter a simple command (e.g., echo Job 1 completed).

- Click **"Save"**.

2. **Create Job 2:**
   - Repeat the steps to create a second job (e.g., Job2).
   - Use a command like echo Job 2 completed.
   - Click **"Save"**.
3. **Create Job 3:**
   - Create a third job (e.g., Job3).
   - Use a command like echo Job 3 completed.
   - Click **"Save"**.

Step 3: Create the Pipeline Job

1. **Create a New Pipeline Job:**
   - Click on **"New Item"**.
   - Enter a name for your pipeline (e.g., SequentialPipeline).
   - Select **"Pipeline"** and click **"OK"**.
2. **Configure the Pipeline:**
   - Scroll down to the **Pipeline** section.
   - In the **Definition** dropdown, select **"Pipeline script"**.

Step 4: Write the Pipeline Script

1. **Define the Pipeline Script:**
   - Enter the following script in the **Pipeline** field:

```
pipeline {
    agent any
    stages {
        stage('Job 1') {
            steps {
                script {
```

```
            echo "Starting Job 1..."
        }
        build job: 'Job 1', wait: true
        script {
            echo "Job 1 completed successfully."
        }
        // Wait for user input to continue
        input message: 'Proceed to Job 2?'
    }
}
stage('Job 2') {
    steps {
        script {
            echo "Starting Job 2..."
        }
        build job: 'Job 2', wait: true
        script {
            echo "Job 2 completed successfully."
        }
        // Wait for user input to continue
        input message: 'Proceed to Job 3?'
    }
}
stage('Job 3') {
    steps {
        script {
            echo "Starting Job 3..."
        }
```

```
            build job: 'Job 3', wait: true
            script {
                echo "Job 3 completed successfully."
            }
            // Final message
            echo "All jobs completed."
        }
    }
  }
}
```

2. **Save the Pipeline:**
    - Click on **"Save"**.

Step 5: Execute the Pipeline

1. **Run the Pipeline:**
    - From the Jenkins dashboard, click on your SequentialPipeline job.
    - Click on **"Build Now"**.
2. **Monitor Execution:**
    - Click on the build number in the **Build History**.
    - After each job completes, you will see a prompt asking for confirmation to proceed to the next job. Click **"Proceed"** to move to the next stage.

Step 6: Verify Results

1. **Check Console Output:**
    - Verify that the output confirms that each job ran sequentially, and observe the messages indicating the start and completion of each job.
2. **Completion Message:**
    - At the end of the pipeline, you should see the final message indicating that all jobs have been completed.

Console Output

Download | Copy | View as plain text

```
Started by user DivyaM
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\Demo
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Job 1)
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Starting Job 1...
[Pipeline] }
[Pipeline] // script
[Pipeline] build (Building Job 1)
Scheduling project: Job 1
Starting building: Job 1 #4
Build Job 1 #4 completed: SUCCESS
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Job 1 completed successfully.
[Pipeline] }
[Pipeline] // script
[Pipeline] input
Proceed to Job 2?
Proceed or Abort
Approved by DivyaM
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Job 2)
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Starting Job 2...
[Pipeline] }
[Pipeline] // script
[Pipeline] build (Building Job 2)
Scheduling project: Job 2
Starting building: Job 2 #3
Build Job 2 #3 completed: SUCCESS
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Job 2 completed successfully.
[Pipeline] }
[Pipeline] // script
[Pipeline] input
Proceed to Job 3?
Proceed or Abort
Approved by DivyaM
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Job 3)
[Pipeline] script
[Pipeline] {
```

```
Starting Job 3...
[Pipeline] }
[Pipeline] // script
[Pipeline] build (Building Job 3)
Scheduling project: Job 3
Starting building: Job 3 #3
Build Job 3 #3 completed: SUCCESS
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Job 3 completed successfully.
[Pipeline] }
[Pipeline] // script
[Pipeline] echo
All jobs completed.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```