1. Global Environment Variables

When you define environment variables globally in Jenkins (via `Manage Jenkins` > `Configure System`), they are available to all jobs and pipelines running on the Jenkins instance. These variables are fetched by Jenkins and injected into the pipeline's environment.

How It Works:

- **Configuration**: You define global environment variables in the Jenkins global configuration page.
- **Pipeline Access**: In the `Jenkinsfile`, you can access these variables directly by their names. Jenkins resolves these names to their values during runtime.

Example: If you define `MAVEN_HOME` globally:

- Name: `MAVEN_HOME`
- Value: `/usr/local/maven`

In the `Jenkinsfile`:

```
sh "${env.MAVEN_HOME}/bin/mvn clean install"
```

Jenkins replaces `${env.MAVEN_HOME}` with `/usr/local/maven`.

2. Environment Variables in Pipeline Job Configuration

When you define environment variables within a specific pipeline job's configuration:

- **Configuration**: You add environment variables under the job configuration or in the pipeline's `environment` block.
- **Pipeline Access**: These variables are available only to that specific job or pipeline.

Example in Pipeline Job Configuration: If you define `DEPLOY_SERVER` and `DEPLOY_PATH` in the pipeline configuration:

- Name: `DEPLOY_SERVER`
- Value: `your-server`
- Name: `DEPLOY_PATH`
- Value: `/path/to/deploy`

In the `Jenkinsfile`:

```
sh "scp target/${env.APP_NAME}.jar
user@${env.DEPLOY_SERVER}:${env.DEPLOY_PATH}"
```

Multi-Stage Pipeline

1. **Stages**: Logical divisions within the pipeline that group related steps or tasks.
2. **Steps**: Individual commands or scripts executed within a stage.
3. **Agents**: Define where the pipeline or specific stages will run.
4. **Environment**: Variables that are available during the pipeline execution.
5. **Post Actions**: Actions that run after a stage or pipeline completes, such as cleanup or notifications.
6. **Triggers**: Conditions that start the pipeline, like changes in the source code repository.

Step 4: Define the Stages and Steps

```
pipeline {
    agent any

    environment {
        // Define environment variables if needed
        EXAMPLE_VAR = 'value'
    }

    stages {
        stage('Build') {
            steps {
                echo 'Building...'
                // Add build steps here, e.g., compile code
                sh 'make build'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing...'
                // Add test steps here, e.g., run unit tests
                sh 'make test'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
                // Add deployment steps here, e.g., deploy to
staging environment
                sh 'make deploy'
            }
        }
    }

    post {
        always {
            // Steps that always run after the pipeline finishes
            echo 'Cleaning up...'
            sh 'make clean'
        }
        success {
            // Steps that run only if the pipeline succeeds
            echo 'Pipeline succeeded!'
        }
        failure {
            // Steps that run only if the pipeline fails
            echo 'Pipeline failed!'
        }
    }
}
```

Breakdown

- **Pipeline Block**: The main container for the pipeline script.
- **Agent Block**: Specifies where the pipeline or specific stages will run. `any` means it can run on any available agent.
- **Environment Block**: Defines environment variables that will be available during the pipeline execution.
- **Stages Block**: Contains individual stages (`Build`, `Test`, `Deploy`), each with its own set of steps.
- **Steps Block**: Contains the commands or scripts to be executed within a stage.
- **Post Block**: Contains actions that run after the pipeline completes. You can define different post actions for `always`, `success`, and `failure` scenarios.