**environment {}**

In a Jenkins Pipeline, the environment {} block is used to define environment variables that are accessible throughout the pipeline or within a specific stage. These variables can be used to store configuration values, credentials, or any other data that needs to be reused.

Here’s a **detailed explanation** with an example:

**🔧 Syntax and Usage of environment {}**

The environment block can be placed:

* At the **pipeline level** (to apply to all stages)
* Inside a **stage** (to apply only to that stage)

**✅ Example: Basic Pipeline with environment {}**

pipeline {

    agent any

    environment {

        // Global environment variables

        APP\_NAME = 'MyApp'

        DEPLOY\_ENV = 'staging'

        BUILD\_NUMBER = "${env.BUILD\_ID}" // Using built-in env variable

    }

    stages {

        stage('Build') {

            steps {

                echo "Building ${APP\_NAME} for ${DEPLOY\_ENV}"

                echo "Build number is ${BUILD\_NUMBER}"

            }

        }

        stage('Test') {

            environment {

                // Stage-specific environment variable

                TEST\_ENV = 'unit'

            }

            steps {

                echo "Running tests in ${TEST\_ENV} environment"

            }

        }

    }

}

**🧠 Key Points**

* **Global vs Stage-Specific**: Variables defined at the top level are available in all stages. Variables inside a stage override or add to the global ones.
* **Using Built-in Variables**: Jenkins provides built-in environment variables like BUILD\_ID, JOB\_NAME, WORKSPACE, etc.
* **String Interpolation**: You can use ${} to reference other variables.
* **Security**: For sensitive data like passwords or tokens, use Jenkins **Credentials Binding Plugin** instead of hardcoding them.

**🔐 Using Credentials Securely**

environment {

    // Use credentials securely

    GITHUB\_TOKEN = credentials('github-token-id')

}

This pulls a credential stored in Jenkins (with ID github-token-id) and makes it available as an environment variable.

**Params**

In a Jenkins Pipeline, the params object is used to define and access **parameters** that can be passed to the pipeline when it's triggered. These parameters allow users to customize the behavior of the pipeline without modifying the code.

**🧩 Where params is Used**

* Defined in the parameters {} block at the top of the pipeline.
* Accessed using params.<PARAM\_NAME> in the pipeline logic.

**✅ Example: Using params in a Declarative Pipeline**

pipeline {

    agent any

    parameters {

        string(name: 'BRANCH\_NAME', defaultValue: 'main', description: 'Git branch to build')

        booleanParam(name: 'RUN\_TESTS', defaultValue: true, description: 'Run unit tests?')

        choice(name: 'DEPLOY\_ENV', choices: ['dev', 'staging', 'prod'], description: 'Deployment environment')

    }

    stages {

        stage('Checkout') {

            steps {

                echo "Checking out branch: ${params.BRANCH\_NAME}"

                // git branch: params.BRANCH\_NAME, url: 'https://github.com/your-repo.git'

            }

        }

        stage('Test') {

            when {

                expression { return params.RUN\_TESTS }

            }

            steps {

                echo "Running tests..."

                // sh 'npm test'

            }

        }

        stage('Deploy') {

            steps {

                echo "Deploying to ${params.DEPLOY\_ENV} environment"

                // deploy logic here

            }

        }

    }

}

**🧠 Types of Parameters You Can Use**

| **Type** | **Syntax Example** |
| --- | --- |
| string | string(name: 'BRANCH', defaultValue: 'main', description: 'Branch to build') |
| booleanParam | booleanParam(name: 'RUN\_TESTS', defaultValue: true, description: 'Run tests?') |
| choice | choice(name: 'ENV', choices: ['dev', 'staging', 'prod'], description: 'Environment') |
| text | text(name: 'NOTES', defaultValue: '', description: 'Release notes') |
| password | password(name: 'SECRET', defaultValue: '', description: 'Secret token') |
| file | file(name: 'CONFIG\_FILE', description: 'Upload config file') |

**withEnv(['VAR=value'])**

In Jenkins **Scripted Pipelines**, withEnv(['VAR=value']) is used to **temporarily set environment variables** for a block of steps. It’s a way to inject environment variables into the shell or script execution context.

**🧪 Example: Using withEnv in a Scripted Pipeline**

node {

    stage('Build') {

        withEnv(['APP\_ENV=production', 'VERSION=1.2.3']) {

            echo "Environment: ${env.APP\_ENV}"

            echo "Version: ${env.VERSION}"

            sh 'echo Deploying version $VERSION to $APP\_ENV'

        }

    }

}

**🔍 How It Works**

* withEnv takes a **list of strings**, each in the format 'KEY=value'.
* These variables are **only available inside the block**.
* Outside the block, they are not accessible.

**🧠 When to Use withEnv**

* When you need **temporary environment variables** for a specific step or stage.
* When you want to **override global environment variables** temporarily.
* When using **Scripted Pipelines**, as environment {} is for Declarative Pipelines.

**✅ Comparison with environment {}**

| **Feature** | **environment {} (Declarative)** | **withEnv([]) (Scripted)** |
| --- | --- | --- |
| Scope | Pipeline or stage | Specific block |
| Syntax | Declarative | Scripted |
| Use case | Global or stage-level vars | Temporary/local vars |
| Example | APP\_ENV = 'prod' | 'APP\_ENV=prod' |

**withCredentials()**

In Jenkins Pipelines, withCredentials() is a **step** used to securely access credentials stored in Jenkins' **Credentials Manager**. It temporarily sets environment variables with the credential values, which can then be used in shell scripts, tools, or other steps.

**🔐 Why Use withCredentials()?**

* Keeps secrets like API tokens, passwords, SSH keys, etc., **secure and hidden**.
* Prevents accidental exposure in logs.
* Supports multiple credential types (username/password, secret text, SSH keys, etc.).

**✅ Basic Example: Secret Text**

pipeline {

    agent any

    stages {

        stage('Use Secret') {

            steps {

                withCredentials([string(credentialsId: 'my-secret-token', variable: 'TOKEN')]) {

                    sh 'echo "Using token: $TOKEN"'

                    // Use $TOKEN in your scripts or tools

                }

            }

        }

    }

}

* credentialsId: The ID of the credential stored in Jenkins.
* variable: The name of the environment variable to expose.

**🔑 Other Credential Types**

| **Credential Type** | **Syntax Example** |
| --- | --- |
| **Secret Text** | string(credentialsId: 'id', variable: 'VAR') |
| **Username/Password** | usernamePassword(credentialsId: 'id', usernameVariable: 'USER', passwordVariable: 'PASS') |
| **SSH Key** | sshUserPrivateKey(credentialsId: 'id', keyFileVariable: 'KEY', usernameVariable: 'USER') |
| **File** | file(credentialsId: 'id', variable: 'FILE') |

**🧪 Example: Username and Password**

withCredentials([usernamePassword(credentialsId: 'my-creds', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {

    sh 'curl -u $USERNAME:$PASSWORD https://example.com/api'

}

Or

pipeline {

    agent any

    stages {

        stage('Clone Git Repository') {

            steps {

                script {

                    withCredentials([usernamePassword(

                        credentialsId: 'git-creds',

                        usernameVariable: 'GIT\_USER',

                        passwordVariable: 'GIT\_PASS'

                    )]) {

                        // Clone using HTTPS with embedded credentials

                        sh '''

                            echo "Cloning repository with credentials..."

                            git clone https://$GIT\_USER:$GIT\_PASS@github.com/your-username/your-private-repo.git

                        '''

                    }

                }

            }

        }

    }

}

**🧠 Best Practices**

* Never echo secrets directly in logs.
* Use withCredentials only in the scope where the secret is needed.
* Store credentials in Jenkins > **Manage Jenkins > Credentials**.