JENKINS SESSION 5

GROOVY SCRIPTING AND SHARED LIBRARIES

HRISHIKESH MOHAN

INTRO

- Understand Groovy's role in Jenkins and basic scripting syntax.
- Learn to use Groovy in pipelines and the Script Console.
- Create and configure Jenkins shared libraries for code reusability.
- Implement practical examples to standardize CI/CD workflows.

GROOVY IN JENKINS

- Groovy is a dynamic scripting language that runs on the JVM, used extensively in Jenkins for pipeline DSL.
- Why Groovy? It's beginner-friendly, integrates seamlessly with Java, and powers Jenkins' Pipeline as Code.
- Key features: Dynamic typing, closures, DSL support—ideal for automating Jenkins tasks without full Java complexity.
- Use case: Writing custom pipeline steps or admin scripts.

GROOVY IN JENKINS

- Groovy is a dynamic scripting language that runs on the JVM, used extensively in Jenkins for pipeline DSL.
- Why Groovy? It's beginner-friendly, integrates seamlessly with Java, and powers Jenkins' Pipeline as Code.
- Key features: Dynamic typing, closures, DSL support—ideal for automating Jenkins tasks without full Java complexity.
- Use case: Writing custom pipeline steps or admin scripts.

SYNTAX AND USAGE

- Variables: def x = 10 (dynamic) or typed like int y = 20.
- Control structures: If-else, loops (for, while), and closures { println it }.
- Strings: Single quotes for literals, double for interpolation ("Hello ${name}").

```
def greet(name) {
    println "Hello, ${name}!"

greet("Jenkins User")
```

- Practical use: Custom functions for build notifications or file manipulations in Jenkins.

GROOVY IN JENKINS PIPELINES

- Pipelines use Groovy-based DSL for declarative/scripted syntax.
- Declarative with Groovy logic: Use script blocks for custom code inside stages.
- Use case: Conditional builds, like if (env.BRANCH_NAME == 'main') { deploy() }.

JENKINS CONSOLE (CLI)

- Access via Manage Jenkins → Script Console for running ad-hoc Groovy.
- Useful for: Querying Jenkins objects, bulk updates, or troubleshooting.
- Example: List all jobs—*Jenkins.instance.getAllItems(Job.class).each { println it.name }.*
- Security note: Restrict access; use for admin tasks like disabling jobs programmatically.
- Practical use: Automate plugin installations or user management.

SHARED LIBRARIES

- Collections of reusable Groovy scripts stored in a Git repo, loaded into pipelines.
- Benefits: Reduce duplication, enforce standards, share code across teams.
- Components: Global variables (vars/), classes (src/), resources.
- Use case: Standard deploy functions used in multiple project pipelines.

```
(root)
├── src                     # Groovy source files (classes)
│   └── org
│       └── foo
│           └── Bar.groovy
├── vars                    # Global variables/functions
│   ├── deploy.groovy       # Reusable function
│   └── deploy.txt          # Documentation (optional)
└── resources               # Static files (e.g., templates)
```

SETUP A SIMPLE SHARED LIBRARY

- WORKING SESSION

vars/greet.groovy

```
def call(String name) {
    echo "Hello, ${name}! Welcome to Jenkins."
}
```

THANK YOU