Building a (resumable and extensible) DSL with Apache Groovy

Jesse Glick
CloudBees, Inc.

Introduction

About Me

- Longtime Jenkins core contributor
- Primary developer on Jenkins Pipeline

© 2016 CloudBees, Inc. All Rights Reserved.

Meet Jenkins Pipeline

- A new “project type” in Jenkins.
- Defines an implicit series of behaviors as an explicit series of stages, implemented in *code.*
- Generally checked into source control as a `Jenkinsfile`.
- Resumability and durability of the pipeline state.
- Easy to extend DSL for end-users and plugin authors alike.

© 2016 CloudBees, Inc. All Rights Reserved.

Meet Jenkins Pipeline

```
node {
    stage('Build') { sh 'mvn -B clean package' }
    stage('Test') { sh 'mvn verify' }
    stage('Deploy') { sh 'mvn release' }
}
```

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

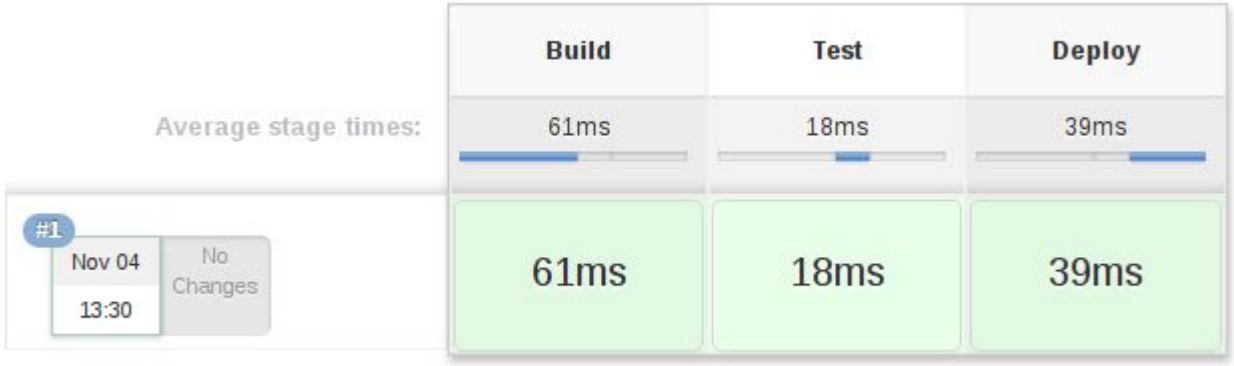
Meet Jenkins Pipeline

```
def container

stage('Build Container') {

    container = docker.build('pipeline-demo:apacheconeu')

}

stage('Verify Container') {

    container.inside { sh './self-test.sh'  }

}
```

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

Meet Jenkins Pipeline



© 2016 CloudBees, Inc. All Rights Reserved.

Meet Jenkins Pipeline

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

Design Requirements

Technology Constraints

- Must run on the Java Virtual Machine
  - Groovy was already familiar from other Jenkins scripting features
  - But must be able to restrict access to Jenkins internals
- Compatible with Jenkins domain concepts
  - Plugins working with “nodes”, “workspaces”, &c. can be migrated naturally
- Allow the creation of a domain-specific language (DSL)

© 2016 CloudBees, Inc. All Rights Reserved.

Desired Features

- A DSL which end-users can understand/get started with relative ease
  - Enable modeling control flow in a single script instead of multiple job configurations
- A DSL which plugin developers can understand/extend with relative ease
  - Support new “steps” via existing Extension Point mechanisms used by Jenkins plugins
- Pause and resume execution
  - Survive a Jenkins master restart

© 2016 CloudBees, Inc. All Rights Reserved.

Why create a DSL?

- Easier to model a continuous delivery pipeline “as code”
  - Developers tend to express complex concepts efficiently in source code
  - Easy to express continuous delivery logic using imperative programming constructs
  - Describing a pipeline in pseudo-code would look a lot like the Pipeline DSL
- Easily understood metaphors for extensibility
  - New “steps” provided by plugins logically integrate into a Pipeline script

© 2016 CloudBees, Inc. All Rights Reserved.

Touring the Implementation

Continuation Passing Style

- All Groovy methods calls, loops, &c. translated to “continuations”
  - Uses stock compiler with a `CompilationCustomizer`
  - Special exception type `CpsCallableInvocation` denotes transfer of control
- The Jenkins build runs an interpreter loop
  - CPS-transformed methods may call “native” Java/Groovy functions, or “steps”
- Currently the only implementation of “Pipeline engine” extension point

cloudbees.
© 2016 CloudBees, Inc. All Rights Reserved.

Serialization of program state

- Program state saved periodically from interpreter
  - When Jenkins restarts, interpreter loop resumes running where it left off
- Local variables/values must be `java.io.Serializable`
  - Unless inside a `@NonCPS` (“native”) method
- Uses JBoss Marshalling River for features not in Java serialization
  - Extension point to replace references to “live” model objects with “pickles”

© 2016 CloudBees, Inc. All Rights Reserved.

```groovy
@Library('testInParallel') _

stage('Sources') {
  node {
    checkout scm
    stash name: 'sources', excludes: 'Jenkinsfile,target/'
  }
}

stage('Testing') {
  testInParallel(count(5), 'inclusions.txt', 'exclusions.txt', 'target/surefire-reports/TEST-*.xml', 'maven:3.3.9-jdk-8', {
    unstash 'sources'
  }, {
    configFileProvider([configFile(fileId: 'jenkins-mirror', variable: 'SETTINGS')]) {
      sh 'mvn -s $SETTINGS -B clean test -Dmaven.test.failure.ignore'
    }
  })
}
```

restart here (×5)

```groovy
def call(parallelism, inclusionsFile, exclusionsFile, results, image, prepare, run) {
  def splits = splitTests parallelism: parallelism, generateInclusions: true
  def branches = [:]
  for (int i = 0; i < splits.size(); i++) {
    def num = i
    def split = splits[num]
    branches["split${num}"] = {
      stage("Test Section #${num + 1}") {
        docker.image(image).inside {
          stage('Preparation') {
            prepare()
            writeFile file: (split.includes ? inclusionsFile : exclusionsFile), text: split.list.join("\n")
            writeFile file: (split.includes ? exclusionsFile : inclusionsFile), text: ''
          }
          stage('Main') {
            run()
          }
          stage('Reporting') {
            junit results
          }
        }
      }
    }
  }
  parallel branches
}
```

running closure

© 2016 CloudBees, Inc. All Rights Reserved.

Thread behavior

- Build runs in at most one native thread
  - From a thread pool, so zero native resources consumed when sleeping
- `parallel` step (fork + join) uses coöperative multitasking
- All Groovy code runs on Jenkins master
  - “Real work” is done in external processes, typically on remote agents
  - `node {…}` block merely sets a connection protocol for nested `sh`/`bat` steps
- Block-scoped steps may pass information via dynamic scope
  - Example: environment variables

© 2016 CloudBees, Inc. All Rights Reserved.

Thread Dump

```
Thread #6
        at DSL.parallel(Native Method)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:25)
        at WorkflowScript.run(WorkflowScript:11)
        at DSL.stage(Native Method)
        at WorkflowScript.run(WorkflowScript:10)
Thread #76
        at DSL.sh(awaiting process completion in /var/jenkins_home/mock-agents/mock-agent-4/workspace/pipeline/master@tmp/durable-cb6fce04 on mock-agent-4 (pid: 7))
        at WorkflowScript.run(WorkflowScript:15)
        at DSL.wrap(Native Method)
        at WorkflowScript.run(WorkflowScript:14)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:16)
        at DSL.stage(Native Method)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:15)
        at org.jenkinsci.plugins.docker.workflow.Docker$Image.inside(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci,
        at DSL.withDockerContainer(Native Method)
        at org.jenkinsci.plugins.docker.workflow.Docker$Image.inside(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci,
        at org.jenkinsci.plugins.docker.workflow.Docker.node(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci/plugins,
        at DSL.node(running on mock-agent-4)
        at org.jenkinsci.plugins.docker.workflow.Docker.node(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci/plugins,
        at org.jenkinsci.plugins.docker.workflow.Docker$Image.inside(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci,
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:9)
        at DSL.stage(Native Method)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:8)
Thread #66
        at DSL.sh(awaiting process completion in /var/jenkins_home/mock-agents/mock-agent-3/workspace/pipeline/master@tmp/durable-4d0fa6af on mock-agent-3 (pid: 7))
        at WorkflowScript.run(WorkflowScript:15)
        at DSL.wrap(Native Method)
        at WorkflowScript.run(WorkflowScript:14)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:16)
        at DSL.stage(Native Method)
        at testInParallel.call(/var/jenkins_home/jobs/pipeline/branches/master/builds/2/libs/testInParallel/vars/testInParallel.groovy:15)
        at org.jenkinsci.plugins.docker.workflow.Docker$Image.inside(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci,
        at DSL.withDockerContainer(Native Method)
        at org.jenkinsci.plugins.docker.workflow.Docker$Image.inside(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci,
        at org.jenkinsci.plugins.docker.workflow.Docker.node(jar:file:/var/jenkins_home/plugins/docker-workflow/WEB-INF/lib/docker-workflow.jar!/org/jenkinsci/plugins,
        at DSL.node(running on mock-agent-3)
```

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

Script security

- Do not want scripts making arbitrary Java API calls or accessing local system
  - Yet some trusted users *should* be able to access Jenkins internal APIs
- “Sandbox”: another `CompilationCustomizer` to insert security checks
  - Before every method/constructor/field access
  - Implementation shared with several other Groovy-based features in Jenkins
- Stock whitelist in product, plus per-site additions
- Libraries configured by an administrator are trusted

© 2016 CloudBees, Inc. All Rights Reserved.

Extension by plugins

- `Step` extension point permits any plugin to add a new “built-in” function
- Can take named parameters
  - polymorphic structures & lists
  - optional “block” (`Closure`)
- `StepExecution` can return immediately, or start something then go to sleep
  - Asynchronous execution terminated with a callback: result object, or exception
- Blocks may be run 0+ times and given context (e.g., a console highlighter)
- Work in progress: `StepExecution` implemented in Groovy
  - Can call other steps, which may be asynchronous
  - Handy for aggregating lower-level steps into a convenient wrapper
- Arbitrary DSLs also possible
  - but, GUI & tool support is weaker

© 2016 CloudBees, Inc. All Rights Reserved.

Groovy libraries

- Reusable code via Pipeline library system
  - Global libraries configured by administrators
  - Per-folder libraries configured by team
  - Or config-free: `@Library('github.com/cloudbeers/multibranch-demo-lib') _`
- Specify version in SCM (`@1.3`, `@abc1234`) or float (`@master`)
- Define class libraries: `src/org/myorg/jenkins/Lib.groovy`
- Or variables/functions: `src/utils.groovy`
- Global libraries may use “Grape” system to load anything in Maven Central
  - `@Grab('com.google.guava:guava:19.0') import com.google.common.base.CharMatcher`

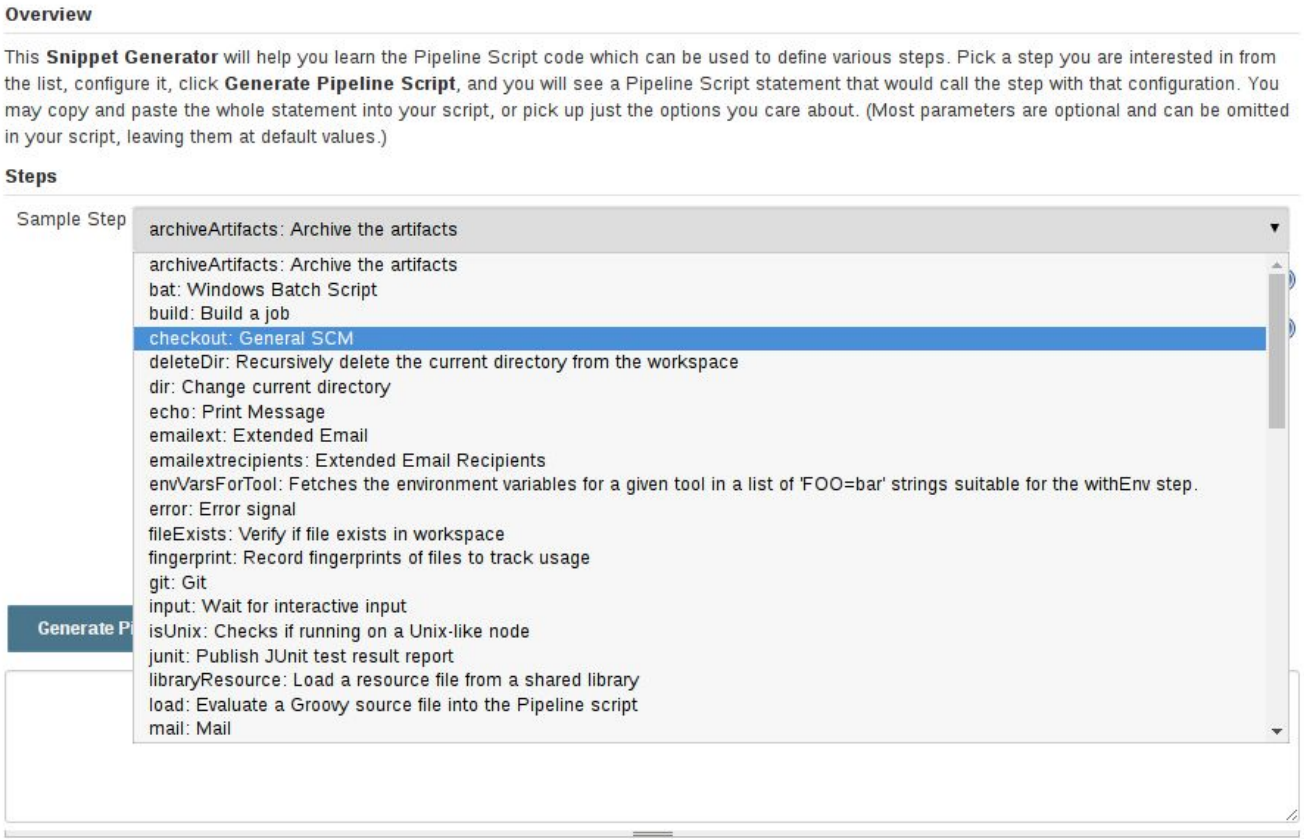
cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

Auto-generated documentation

- Extension point for plugins to provide built-in help
- Structure of step parameters introspected
- In-product help accepts configuration forms similar to rest of Jenkins

cloudbees.
© 2016 CloudBees, Inc. All Rights Reserved.



Snippet Generator

© 2016 CloudBees, Inc. All Rights Reserved.

Sample Step: configFileProvider: Provide Configuration files

Make globally configured files available in you local workspace. All files configured via the config-file-provider plugin are available an can be referenced.

(from Config File Provider Plugin)

Managed Files

File: Jenkins Mirror

Target:

Variable: SETTINGS

Replace Tokens

Add file

Generate Groovy

```
configFileProvider([configFile(fileId: 'jenkins-mirror', variable: 'SETTINGS')]) {
    // some block
}
```

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.

Pipeline Steps Reference

The following plugins offer Pipeline-compatible steps. Each plugin link offers more information about the parameters for each step.

ansible
- ansiblePlaybook: Invoke an ansible playbook

artifactory
- ArtifactoryGradleBuild: run Artifactory gradle
- ArtifactoryMavenBuild: run Artifactory maven
- artifactoryDownload: Download artifacts
- artifactoryPromoteBuild: Promote build
- artifactoryUpload: Upload artifacts
- collectEnv: Collect environment variables and system properties
- dockerPullStep: Artifactory docker pull
- dockerPushStep: Artifactory docker push
- getArtifactoryServer: Get Artifactory server from Jenkins config
- newArtifactoryServer: Returns new Artifactory server
- newBuildInfo: New buildInfo
- newGradleBuild: New Artifactory gradle executor
- newMavenBuild: New Artifactory maven
- publishBuildInfo: Publish build Info to Artifactory

Pipeline Step Reference

jenkins.io/doc/pipeline/steps

© 2016 CloudBees, Inc. All Rights Reserved.

The Good,
The Bad,
The Groovy

Useful Groovy features

- Smooth integration of Java APIs
- Flexible syntax (named vs. positional parameters, closures, …)
- `CompilationCustomizer`

cloudbees.
© 2016 CloudBees, Inc. All Rights Reserved.

CPS & Sandbox vs. Groovy Challenges

- `DefaultGroovyMethods` helpers taking a `Closure` do not work
  - `[1, 2, 3].each {x -> sh "make world${x}"}` → FAIL
- Most java.util.Iterator implementations are not Serializable
  - `for (x in [1, 2, 3]) {sh "make world${x}"}` → OK (special-cased)
  - `for (x in [1, 2, 3, 4, 5].subList(0, 3)) {sh "make world${x}"}` → FAIL
- No CPS translation possible for a constructor
- Finding the actual call site for whitelist lookup is really hard
  - `GroovyObject.getProperty`, coercions, `GString`, `Closure.delegate`, `curry`, …
- More exotic language constructs not yet translated in CPS
  - Tuple assignment, spread operator, method pointer, `obj as Interface`, …
- Summary: Groovy is far more complex than initially realized
  - and CPS transformation is hard to develop & debug

© 2016 CloudBees, Inc. All Rights Reserved.

Groovy runtime challenges/roadblocks

- Leaks, leaks, leaks
  - Groovy is full of caches which do not let go of class loaders
  - Java has a few, too
  - `SoftReferences` get cleared…eventually (after your heap is already huge)
  - so Pipeline resorts to tricks to unset fields
- Compilation is expensive
  - not just syntactic parsing, lots of class loading to resolve symbols
  - not currently being cached—under consideration

© 2016 CloudBees, Inc. All Rights Reserved.

Future Development

Declarative Pipeline

- Easier way to write common pipelines
- Friendly to GUI editors
- Lintable
- Escape to `script {...}`

```
pipeline {
    agent docker:"java"
    stages {
        stage("build") {
            steps {
                sh 'mvn clean install -Dmaven.test.failure.ignore=true'
            }
        }
    }
    postBuild {
        always {
            archive "target/**/*"
            junit 'target/surefire-reports/*.xml'
        }
    }
}
```


© 2016 CloudBees, Inc. All Rights Reserved.

Questions

Resources

- jenkins.io/doc
- @jenkinsci
- github.com/jenkinsci/pipeline-plugin
- github.com/jenkinsci/workflow-cps-plugin
- github.com/jenkinsci/pipeline-model-definition-plugin

cloudbees

© 2016 CloudBees, Inc. All Rights Reserved.