Jenkins
User Conference

Getting Groovy with Jenkins

Lorelei McCollum
IBM Verse Test Architect
lmccollu@us.ibm.com

Groovy!

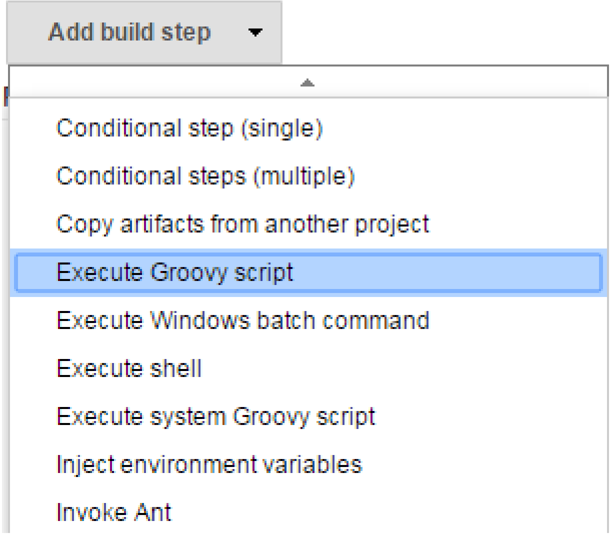
- What is Groovy?
- How to leverage in your jenkins jobs
- Pull groovy scripts directly from source!
- Administration Jobs with Groovy
- Fun Quick Links and Emails with Groovy
- Jenkins Workspace and Groovy scripts
- Groovy Label Assignment Plugin

Groovy

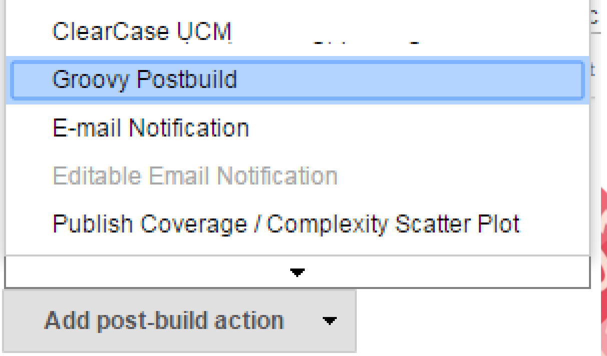
- Is an agile and dynamic language for the Java Virtual Machine
- Is a shortened syntax, looks a lot like java, and understands java syntax
- Jenkins jobs can execute groovy scripts (Groovy plugin)
- You can do a lot with your pipeline dynamically with groovy scripts
- Groovy Script vs System Groovy Script
  - The plain "Groovy Script" is run in a forked JVM, on the slave where the build is run. It's the basically the same as running the "groovy" command and pass in the script.
  - The system groovy script, OTOH, runs inside the Jenkins master's JVM. Thus it will have access to all the internal objects of Jenkins, so you can use this to alter the state of Jenkins

How to use?

- Build Step->
  - Groovy Script or a system Groovy script


Add build step
Conditional step (single)
Conditional steps (multiple)
Copy artifacts from another project
Execute Groovy script
Execute Windows batch command
Execute shell
Execute system Groovy script
Inject environment variables
Invoke Ant

- Postbuild Step->
  - Groovy postbuild


ClearCase UCM
Groovy Postbuild
E-mail Notification
Editable Email Notification
Publish Coverage / Complexity Scatter Plot
Add post-build action

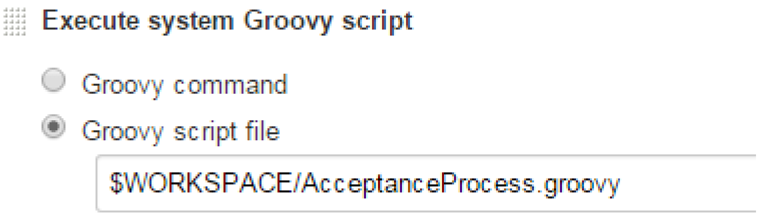
Where to store the script?

- Can put it in source and pull it from source
- Or you can put the script directly into the command box that is provided

Pull groovy scripts directly from source!

#jenkinsconf

Call the scripts directly from Admin Update Pipeline Config

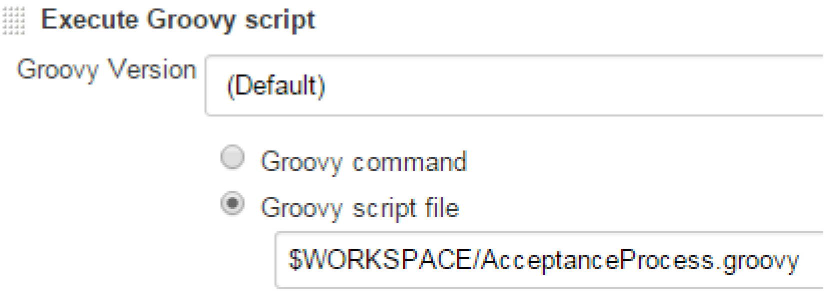


Execute system Groovy script

○ Groovy command

● Groovy script file

$WORKSPACE/AcceptanceProcess.groovy



Execute Groovy script

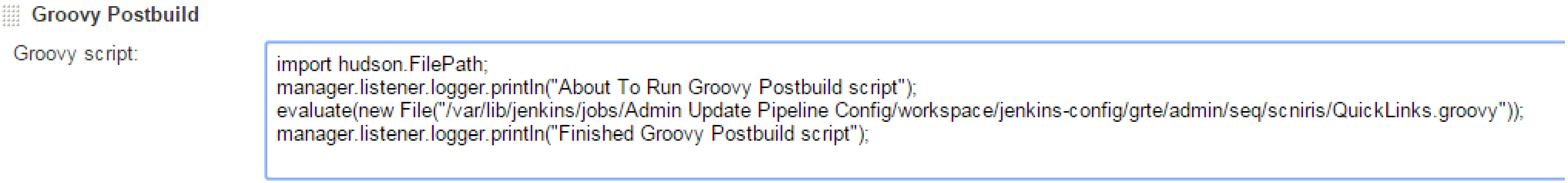
Groovy Version: (Default)

○ Groovy command

● Groovy script file

$WORKSPACE/AcceptanceProcess.groovy

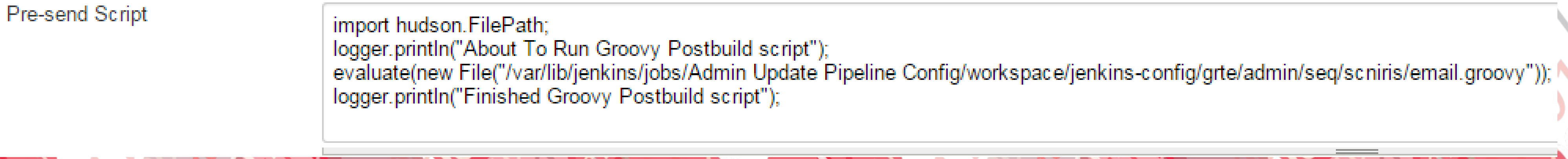
Groovy Postbuild runs on the master and same with the Email-ext pre-send script



Groovy Postbuild

Groovy script:

```
import hudson.FilePath;
manager.listener.logger.println("About To Run Groovy Postbuild script");
evaluate(new File("/var/lib/jenkins/jobs/Admin Update Pipeline Config/workspace/jenkins-config/grte/admin/seq/scniris/QuickLinks.groovy"));
manager.listener.logger.println("Finished Groovy Postbuild script");
```



Pre-send Script

```
import hudson.FilePath;
logger.println("About To Run Groovy Postbuild script");
evaluate(new File("/var/lib/jenkins/jobs/Admin Update Pipeline Config/workspace/jenkins-config/grte/admin/seq/scniris/email.groovy"));
logger.println("Finished Groovy Postbuild script");
```

Fun with Groovy

- Showing Snippets of Groovy scripts
- Most will have various import statements like below, left off the slides, but these are all required

```
import com.tikal.jenkins.plugins.multijob.MultiJobBuild.SubBuild;
import org.jvnet.hudson.plugins.groovypostbuild.GroovyPostbuildSummaryAction
import org.jvnet.hudson.plugins.groovypostbuild.GroovyPostbuildAction
import hudson.model.*
import com.tikal.jenkins.plugins.multijob.*;
import hudson.*;
import hudson.slaves.*;
import hudson.tasks.*;
import jenkins.model.Jenkins
import hudson.plugins.copyartifact.SpecificBuildSelector
import hudson.plugins.copyartifact.CopyArtifact
import hudson.model.AbstractBuild
import hudson.Launcher
import hudson.model.BuildListener
import hudson.FilePath
import groovy.io.FileType
import jenkins.util.VirtualFile;
```

Get ENV variables of the Build

#jenkinsconf

Groovy Script/System Groovy Script

```
def config = new HashMap()
def thr = Thread.currentThread()
def build = thr?.executable
def buildMap = build.getBuildVariables()
config.putAll(buildMap)
def envVarsMap = build.parent.builds[0].properties.get("envVars")
config.putAll(envVarsMap)
def jobName = config.get("JOB_NAME")
def jobURL = build.envVars.BUILD_URL
```

Groovy PostBuild Plugin

```
def bvtresult=manager.envVars["Mail1_BUILD_RESULT"]
```

Groovy Email Pre-send Script

```
def mail1result=build.envVars["MAIL1_BUILD_RESULT"]
def jobURL = build.envVars.BUILD_URL
```

Fun examples for Administrative jobs

#jenkinsconf

Jenkins User Conference

- Poll status of Jenkins nodes
- Restart Jenkins nodes
- Toggle Jenkins nodes offline/online
- Pin the build (mark build as keep forever)
- Create nodes on the fly
- Delete nodes

Poll status of Jenkins Nodes

```
//function to poll how many jobs running on a given slave
def polljobs(node){
    for (slave in jenkins.model.Jenkins.instance.slaves) {
        if (slave.name.equals(node)){
            return slave.getComputer().countBusy()

        }
    }
    return -1
}
```

slave & slave.getComputer() has all sorts of methods you can access off it – google the javadoc!

```
///function for determing if a node is online
def isonline(node){
    for (slave in jenkins.model.Jenkins.instance.slaves)
{
        if (slave.name.equals(node)){
            return slave.getComputer().isOnline()
        }
    }
    return false
}
```

Groovy Snippet to restart a node

#jenkinsconf

```
///function to restart a node
def restart(node){
    for (slave in jenkins.model.Jenkins.instance.slaves) {
        if (slave.name.equals(node)){
            def channel = slave.getComputer().getChannel()
            RemotingDiagnostics.executeGroovy( """
                if (Functions.isWindows()) {
                'shutdown /r /t 10 /c "Restarting after Jenkins test
                completed"'.execute()
                } else {
                "sudo reboot".execute()
                }
            """, channel)

        }

    }
}
```

Groovy Snippet to Toggle a node offline or online

#jenkinsconf

```groovy
class OfflineMessage extends org.jvnet.localizer.Localizable {
    def message
    OfflineMessage() {
        super(null, null, [])
        def time = new Date().format("HH:mm MM/dd/yy z",TimeZone.getTimeZone("EST"))
        this.message = "Toggling the node offline:" + time
    }
    String toString() {
        this.message
    }
    String toString(java.util.Locale l) {
        toString()
    }
}
def togglenode(node,status){
    println "Toggling the node: " + node
    for (slave in jenkins.model.Jenkins.instance.slaves) {
        if (slave.name.equals(node)){
            println "Setting ${node} temporarily ${status}"
            if(status.equals('offline')){
                slave.getComputer().setTemporarilyOffline(true,
                    SimpleOfflineCause.create(new OfflineMessage()))
            }else{
                slave.getComputer().setTemporarilyOffline(false,
                    SimpleOfflineCause.create(new OnlineMessage()))
            }
        }
    }
}
```

Jenkins
User Conference

Pin The Build

#jenkinsconf

```
def getJobs() {
  def hi = hudson.model.Hudson.instance
  return hi.getItems(hudson.model.Job)
}


def getBuildJob(String jobNam) {
  def buildJob = null
  def jobs = getJobs()
  (jobs).each { job ->
    if (job.displayName == jobNam) {
      println("Found")
      println("Exiting job search")
      buildJob = job
      return buildJob
    }
  }
  return buildJob
}
```

```
def getRunToMark(hudson.model.Job job, String
buildNum) {
  def runToMark = null
  if(job != null) {
    (job.getBuilds()).each { run ->
      if (String.valueOf(run.number) ==
buildNum) {
        println("Found")
        println("Exiting build search")
        runToMark = run
        return runToMark
      }
    }
  }
  return runToMark
}
```

Pin the Build - continued

```
def config = new HashMap()
def thr = Thread.currentThread()
def build = thr?.executable
def buildMap = build.getBuildVariables()
def envVarsMap = build.parent.builds[0].properties.get("envVars")
config.putAll(envVarsMap)
config.putAll(build)
String nameOfJob = config.get("JobName")
String numberOfBuild = config.get("BuildNumber")
if((nameOfJob != "") && (numberOfBuild != "")) {
    def runToMark = getBuildRunToMark(getBuildJob(nameOfJob), numberOfBuild)
    if(runToMark != null) {
      runToMark.keepLog(true)
    }
    else {
      println("ERROR : Script could not find matching build")
    }
  }
}else {
    println("ERROR : One of the input parameter value is not set")
}
```

Create a Node

```
def nodename = args[0];
def nodelabel = args[1];
def os = args[2];
def nodedesc = args[3];
def getJNLPSecret(nodename){
    for (slave in jenkins.model.Jenkins.instance.slaves) {
        if (slave.name.equals(nodename)){
            def secret = slave.getComputer().getJnlpMac();
            println "Secret=" + secret
            return;
        }
    }
    println "Secret=Error"
    return;
}
if(os.equals("windows")){
    Jenkins.instance.addNode(
        new DumbSlave(nodename,nodedesc,"C:\\Jenkins","1",
        Node.Mode.EXCLUSIVE,nodelabel,new JNLPLauncher(),
        new RetentionStrategy.Always(),new LinkedList()))
    getJNLPSecret(nodename);
}else {
    println "OS: " + os + " is not supported at this time";
}
```

Delete a Node

#jenkinsconf

```
node_name_value=args[0]
nodelabel=args[1]

for (slave in jenkins.model.Jenkins.instance.slaves) {
    labelName = slave.getLabelString()
    if (labelName.contains("grte")) {
        if (slave.name.equals(node_name_value)){
            println "Delete ${node_name_value}"
            slave.getComputer().doDoDelete();
        }
    }
}
```

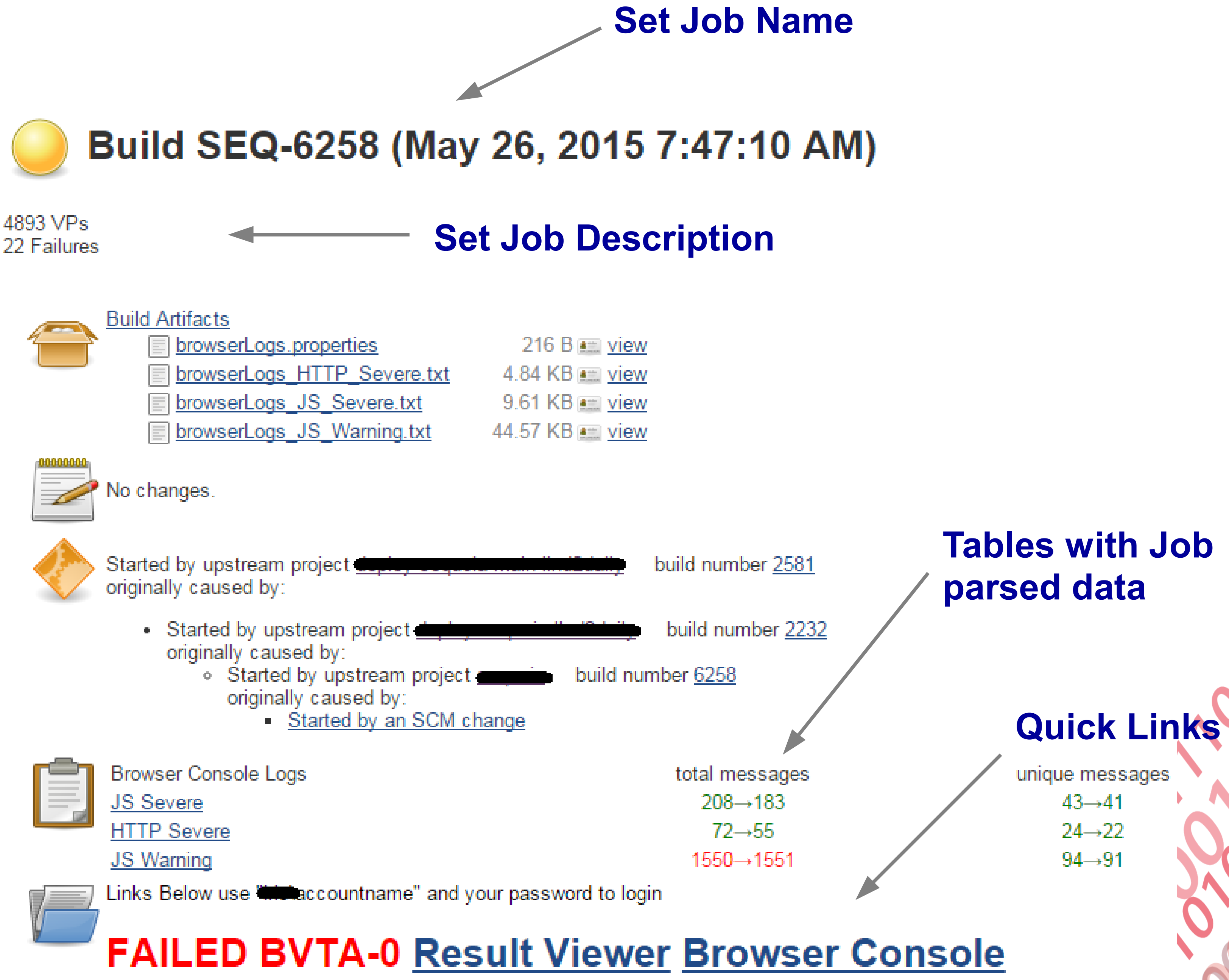
Fun Examples for jenkins builds

- Customize Emails
- Customized QuickLinks
- Set Job Display Name
- Set Job Description
- Parse Sub Job Results and Create Tables

Jenkins User Conference

What are these things?

#jenkinsconf



Set Job Name

Build SEQ-6258 (May 26, 2015 7:47:10 AM)

4893 VPs
22 Failures

Set Job Description

Build Artifacts

browserLogs.properties	216 B	view
browserLogs_HTTP_Severe.txt	4.84 KB	view
browserLogs_JS_Severe.txt	9.61 KB	view
browserLogs_JS_Warning.txt	44.57 KB	view

No changes.

Started by upstream project [redacted] build number 2581
originally caused by:

- Started by upstream project [redacted] build number 2232
  originally caused by:
  - Started by upstream project [redacted] build number 6258
    originally caused by:
    - Started by an SCM change

Tables with Job parsed data

Quick Links

Browser Console Logs	total messages	unique messages
JS Severe	208→183	43→41
HTTP Severe	72→55	24→22
JS Warning	1550→1551	94→91

Links Below use "[redacted]accountname" and your password to login

FAILED BVTA-0 Result Viewer Browser Console

Emails

```
def mail1result=build.envVars["MAIL1_BUILD_RESULT"]
def mail2result=build.envVars["MAIL2_BUILD_RESULT"]
def mail3result=build.envVars["MAIL3_BUILD_RESULT"]
def testResult = build.testResultAction
def subject = "FAILED NO VPs"
def sMessage = ""
def sFailedList = ""
if(testResult != null){
    def total = testResult.totalCount
    def fail = testResult.result.failCount
    if(fail == 0){
        subject =   " PASSED  |  " + total + " VPs"
    }else{
        subject = " FAILED  |  " + fail + " out of " + total
    }

    List listFailed = testResult.result.failedTests
    for(x=0; x<listFailed.size();x++){
        def current = listFailed.get(x)
        def sItem = "Failed: " + current.getClassName() +
            " Reason: " + current.getDisplayName()
        sFailedList = sFailedList + sItem + "<br>"
    }
}
```

Jenkins job has three sub jobs that run via Multi Job plugin
Called: Mail1, Mail2, Mail3
They report test results via Junit plugin

Emails-continued

```
def emailLink(sMessage,result,link,text) {
    if(link != null && !link.equals("")){
            if(result.equals("SUCCESS")){
                sMessage = sMessage + "<b><font color=\"passed\">${text} PASSED:
</font></b><a href=\"${link}\">Log Viewer </a> <br>"
            }else{
                sMessage = sMessage + "<b><font color=\"red\">${text} FAILED:
</font></b><a href=\"${link}\">Log Viewer </a> <br>"
            }
    }else{
        sMessage = sMessage + "<b><font color=\"red\">${text} No Results
</font></b><br>"
    }
    return sMessage
}
```

Emails-continued

```
msg.setSubject(build.envVars.StreamPrefix + "-" +build.envVars.MyBuild + " | "
+build.envVars.JOB_NAME + " | " + subject)
sMessage = emailLink(sMessage,mail1result,build.envVars.ResultsLinkMail1,"Mail1")
sMessage = emailLink(sMessage,mail2result,build.envVars.ResultsLinkMail2,"Mail2")
sMessage = emailLink(sMessage,mail3result,build.envVars.ResultsLinkMail3,"Mail3")
def sTotalMessage = "VerseAlive SCNIris Job results: <br>Jenkins job: "+
build.envVars.BUILD_URL + "<br><br><br>" + sMessage + "<br>" + sFailedList;
msg.setContent(sTotalMessage, 'text/html')
```

All of this goes into a groovy script which is invoked via the pre-send script in the Email-ext post build step

Emails-continued

- Could also do this dynamically instead of hardcoding the variable names for the results

```
public String createEmail(String subject, String jobName, MultiJobBuild build){
 String sMessage = "";
 if(jobName != null){
  MultiJobProject job = Jenkins.instance.getItem(jobName);
  if(job != null){
   MultiJobBuild multiJobBuild =build;
   if (multiJobBuild != null){
    List<SubBuild> subBuilds = multiJobBuild.getSubBuilds();
    for (SubBuild subBuild : subBuilds) {
     String name = subBuild.getJobName();
     String sVarName = name.replace("-","_").toUpperCase();
     String sBuildNumber = sVarName + "_BUILD_NUMBER";
     String sBuildResult = sVarName + "_BUILD_RESULT";
     def buildNbr = build.getEnvironment().get(sBuildNumber)
     def buildResult = build.getEnvironment().get(sBuildResult)
     if(buildNbr != null && buildResult != null && !buildResult.equals("FAILURE")){
       Build bsubBuild = getBuild(name,Integer.parseInt(buildNbr));
       if(bsubBuild != null){
        VirtualFile htmlResultsLinkFile = getArtifact(bsubBuild, "link.txt");
        if (htmlResultsLinkFile != null){
         String logViewerURL = readFirstLine(htmlResultsLinkFile);
         if (logViewerURL != null){
          sMessage = sMessage + emailLink(buildResult,logViewerURL,shortName)
         }else{
          ...

        ...return sMessage
```

QuickLinks – Groovy Postbuild

```
def bvtresult=manager.envVars["Mail1_BUILD_RESULT"]


//logs the quick link on the jenkins build page
def logQuickLink(result,link,text) {
    if(link != null){
        if(!result.equals("SUCCESS")){
            manager.createSummary("folder.gif").appendText("<h1>FAILED ${text} <a href=\
{link}\">Result Viewer</a></h1>", false, false, false, "red")
        }
    }else{
        manager.createSummary("error.gif").appendText("<h1>${text}  No Results </h1>",
false, false, false, "red")
    }
}
logQuickLink(bvtresult,manager.envVars["ResultsLinkMail1"],"Mail1")
```

QuickLinks- Groovy System Script

#jenkinsconf

- Could also do this dynamically instead of hardcoding the variable names for the results

```
public GroovyPostbuildSummaryAction logQuickLink(String result,String link,String text)
{
    if(link != null && link != "NotSet" && !link.equals("")){
      if(!result.equals("SUCCESS")){
        String console=link.replace("logViewer.html","browserConsoleLogs.txt")
        GroovyPostbuildSummaryAction action =
            new GroovyPostbuildSummaryAction("folder.gif");
         action.appendText("<h1>FAILED " + text + " <a href=\"" + link + "\">Result
Viewer</a>  <a href=\"" + console + "\">Browser Console</a> </h1>", false, false, false,
"red")
        return action;
      }else{
       ....

 GroovyPostbuildSummaryAction action=
     logQuickLink(buildResult,logViewerURL,shortName,listener);
 if(action != null){
        build.getActions().add(action);
 }
```

Set Job DisplayName

Groovy System Script

```
def buildMe = Thread.currentThread().executable
buildMe.setDisplayName("${buildprefix}-${Mybuild}")
```

Groovy PostBuild

```
manager.build.setDisplayName("${buildprefix}-${Mybuild}")
```

Assumes you have the envVars defined

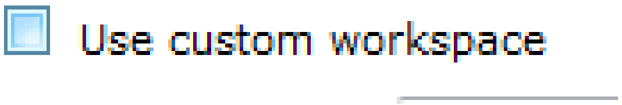
Set Job Description

- Groovy Postbuild Plugin

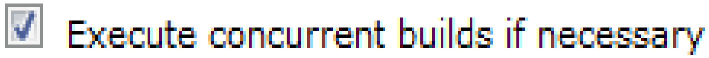
```
def testResult = manager.build.testResultAction
if(testResult != null){
    testResult = manager.build.testResultAction.result
    def total = testResult.totalCount
    def fail = testResult.failCount
    manager.build.setDescription("${total} VPs <br> ${fail} Failures")
}
```

Jenkins Workspace

- All Jenkins jobs have their “own” workspace on the node they run on
- Artifacts get stored for a given build and copied into the workspace
- Use the workspace to your benefit, write out to it in your scripts
- Save job state
- If Jenkins is restarted, jobs currently executing are lost
- Can force jobs to use a custom workspace
- Concurrent jobs get a “@<int> type workspace name if more than one are going at the same time

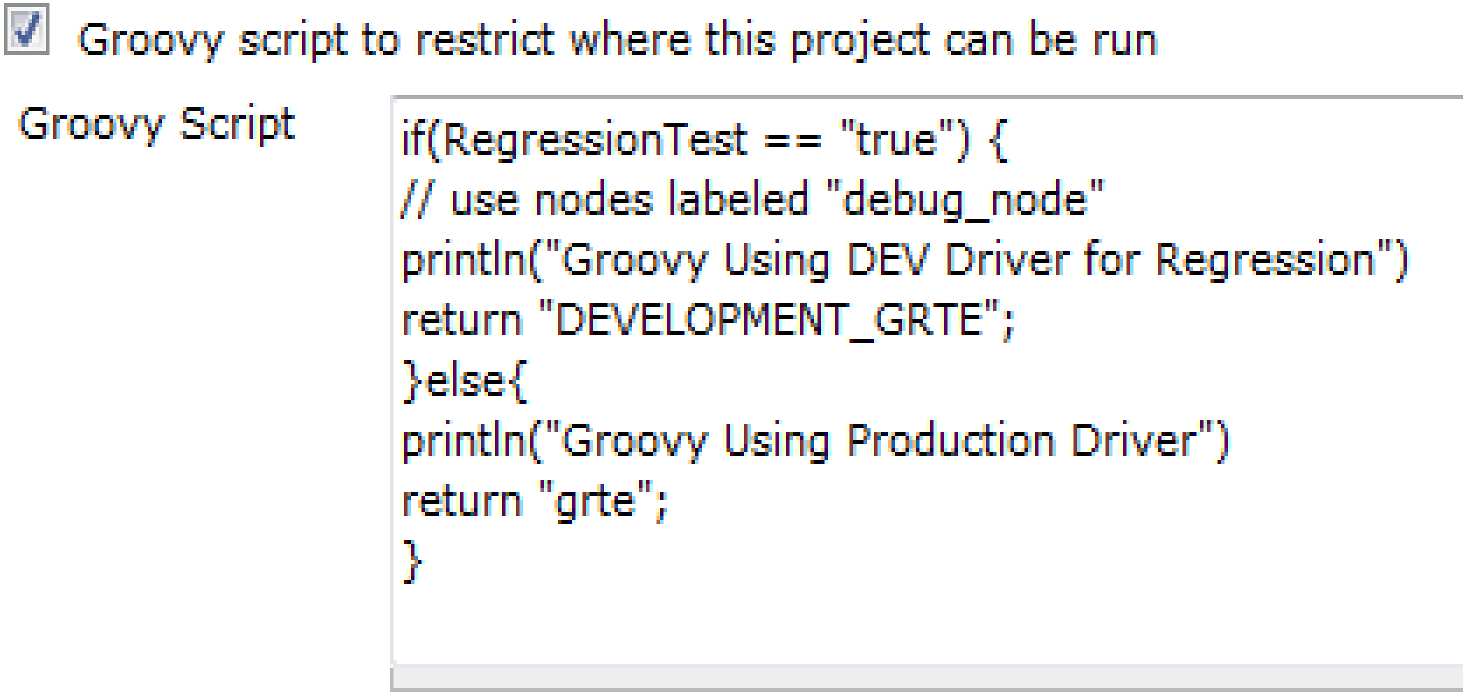


□ Use custom workspace



☑ Execute concurrent builds if necessary

Groovy Label Assignment Plugin



☑ Groovy script to restrict where this project can be run

Groovy Script

```
if(RegressionTest == "true") {
// use nodes labeled "debug_node"
println("Groovy Using DEV Driver for Regression")
return "DEVELOPMENT_GRTE";
}else{
println("Groovy Using Production Driver")
return "grte";
}
```

Useful Plugins Covered in Presentation

- Build-Name Setter Plugin
- Groovy Plugin
- Groovy Label Assignment Plugin
- Groovy PostBuild Plugin
- Junit Plugin
- Xunit Plugin
- Email-ext plugin

Stuck? Need Help?

- Google is your friend, especially stackoverflow
- Jenkins book by O'Reilly
- Forums, communities, use social platforms
- Chances are other people have had your question and may have a solution
- Tons of plugins out there
- Create your own plugin, or help contribute to enhance/fix existing plugins
- Report bugs/issues
- Helpful Links
  - http://jenkins-ci.org/
  - http://www.cloudbees.com/jenkins

Jenkins
User Conference

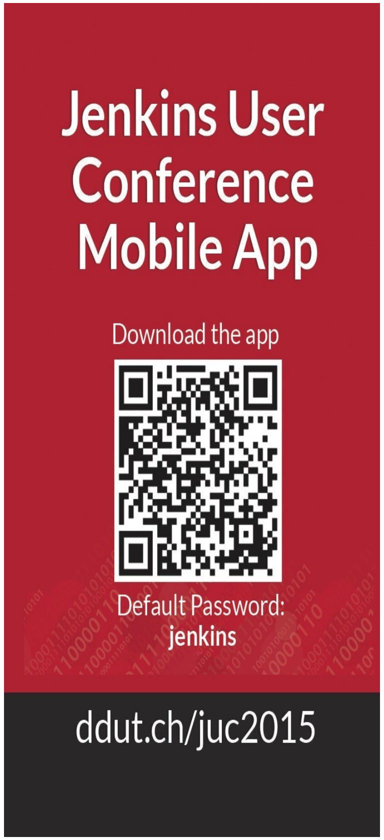
#jenkinsconf

Questions?

Please Share Your Feedback

#jenkinsconf

- Did you find this session valuable?
- Please share your thoughts in the Jenkins User Conference Mobile App.
- Find the session in the app and click on the feedback area.



Jenkins User Conference Mobile App

Download the app

Default Password: jenkins

ddut.ch/juc2015