Jenkins
User Conference

Jenkins 101
Getting Started With Jenkins!

Lorelei McCollum
IBM Verse Test Architect

Getting started with Jenkins

- Install
- Build
- Design First
- Configuring Jenkins jobs
- Parameterize
- Workspace
- Jenkins Nodes
- Security
- Source Control for jobs
- Groovy
- Plugins

Installing Jenkins

- Easy instructions to follow for whatever operating system you choose on Jenkins-CI.org
  - Supports Various Linux/Unix installations as well as Windows

- https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins

Jenkins “Build”

- What is a Jenkins “Build”?
- Not necessarily a product build
- **Build is to a Jenkins job as a main method is to java class**

```
public static void main(String[] args) {

}
```

3 Main Parts to a Jenkins Job

Pre Build

Build

Post Build

- Job allows for custom configuration of all these steps via various plugins
- Need to understand the “Config” of a job before you create one
- Understand what the job can do for you!
- Plugins can contribute anywhere! They determine what actions you have

Jenkins
User Conference

Examples of Build Actions

#jenkinsconf

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
- Invoke top-level Maven targets
- LA Job Executor
- Process xUnit test result report
- Trigger a remote parameterized job
- Trigger/call builds on other projects
- Windows PowerShell

Jenkins
User Conference

Examples of Post Build actions

#jenkinsconf

- Flexible publish
- Publish Checkstyle analysis results
- Publish FindBugs analysis results
- Scan for compiler warnings
- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Console output (build log) parsing
- Publish Clover Coverage Report
- Publish Cobertura Coverage Report
- Publish JUnit test result report
- Publish Javadoc
- Publish Performance test result report
- Publish xUnit test result report
- Record Emma coverage report
- Record fingerprints of files to track usage
- Report Violations
- Git Publisher
- Allow broken build claiming
- Build other projects (manual step)
- ClearCase UCM
- Deploy war/ear to a container
- E-mail Notification
- Editable Email Notification
- Publish Coverage / Complexity Scatter Plot
- Trigger parameterized build on other projects
- Delete workspace when build is done

Add post-build action

Design First

- Re-Configuring jobs can be painful and tedious
- Design jobs so they are easy to update/modify
- What could change in your pipeline over time, think ahead!
- Understand the Jenkins job and all the pre/post build options
- Start/Stop/Restart pipelines from any step
- Debugging your pipeline
- Use source control for storing all aspects of your pipeline (Extended Choice Parameter)
- Start small, then expand, use naming conventions (Nested View Plugin)
- Design out your pipeline ahead of time, then implement in Jenkins

Configuring Jenkins Jobs

- Anything command line - Jenkins can execute
- Don't hard-code things that could change (Extended Choice Parameter Plugin)
- Security – who can build and configure (Matrix Authorization Strategy Plugin)
- Jenkins job results (xUnit Plugin)
- Post processing of results (Publish jUnit/xUnit results, Email-ext plugin)
- Use the workspace to your benefit
- Create and pass environment variables (Environment Injector Plugin)
- There may be a better solution than using “blocking jobs”
- How to maintain your jobs/slave nodes/pipeline?

Parameterize

- Allows Job re-use
- Easy to maintain and update
- Think about how often jobs could change
- Think about your entire pipeline
- What happens when you have a new release/version of your product?
- Params change, use source control for them (Extended Choice Parameter Plugin)
- Can help you to scale and run more in parallel without the added maintenance of duplicate jobs (Multi-job Plugin)

How do you get a “build” with parameters

#jenkinsconf

This build is parameterized

Add Parameter

- Boolean Parameter
- Build selector for Copy Artifact
- CVS Symbolic Name Parameter
- Choice Parameter
- Extended Choice Parameter
- File Parameter
- Git Parameter
- Label
- List Subversion tags (and more)
- Node
- Password Parameter
- Patch file as a parameter
- Run Parameter
- String Parameter
- Text Parameter

Extended Choice Parameter

Name	
Description	
Parameter Type	Single Select
Value	
Property File	
Property Key	
Default Value	
Default Property File	
Default Property Key	
Quote Value	☐
Number of Visible Items	
Delimiter	

Extended Choice Parameter

Example of using properties file that is pulled from source
Jenkins job that builds your “config/param project” on a schedule to a “custom workspace”

Extended Choice Parameter

Name	Browser to Install
Description	Browser to Install on the Machine
Parameter Type	Single Select
Value	
Property File	/var/lib/jenkins/jobs/Admin Update Pipeline Config/workspace/jenkins-config/grte/admin/jobparams.properties
Property Key	browserinstall
Default Value	
Default Property File	/var/lib/jenkins/jobs/Admin Update Pipeline Config/workspace/jenkins-config/grte/admin/jobparams.properties
Default Property Key	browserinstall_default
Quote Value	☐
Number of Visible Items	5
Delimiter	,

Naming Conventions

- Job Names –
  - Think about what different types of jobs you will have an see if you can adopt a naming convention. Spaces in job names may make remote calling of this job hard
- Parameter Names –
  - Think about how the job will be called. Spaces in your parameter names can make remote calling of this job hard
- Build Name –
  - Can set this to use environment variables to be easily readable by user
- Spaces can make your life harder, if you don't need them, don't use them
  - You_can_use_this_style_instead and will make remote API's easier to use

Jenkins Workspace

#jenkinsconf

- 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
- Save job state
- If Jenkins is restarted, jobs currently executing are lost
- Can force jobs to use a custom workspace

Use custom workspace

- Concurrent jobs get a “@<int> type workspace name if more than one are going at the same time

Execute concurrent builds if necessary

Jenkins
User Conference

Jenkins Nodes

#jenkinsconf

- Easy to create in Manage Jenkins
- Term Executors refers to how many jobs can be running concurently on that node
- Can choose how you want them connected to Jenkins
- JNLP for windows works well
  - Can set up as service or use a batch file that gets called on startup
- SSH works well for unix/linux
- Can use Jenkins API or groovy to create nodes dynamically
- Best Practice: Limit how many executors you give your master

Jenkins
User Conference

Creating a new “Dumb Node”

#jenkinsconf

Name: Test Node

Description:

# of executors: 1

Remote FS root:

Remote directory is mandatory

Labels:

Usage: Utilize this node as much as possible

Launch method: Launch slave agents on Unix machines via SSH

Host:

Credentials: root (WCA Servers root users) Add

Advanced...

Availability: Keep this slave on-line as much as possible

Node Properties

- Environment variables
- Prepare jobs environment
- Tool Locations

Save

Connecting & Usage of your new node

- Choose your connect method

Launch method

Launch slave agents on Unix machines via SSH

Launch slave agents on Unix machines via SSH
Launch slave agents via Java Web Start
Launch slave via execution of command on the Master
Let Jenkins control this Windows slave as a Windows service

- Choose how the slave is used by the jobs

Usage

Leave this node for tied jobs only

Utilize this node as much as possible
Leave this node for tied jobs only

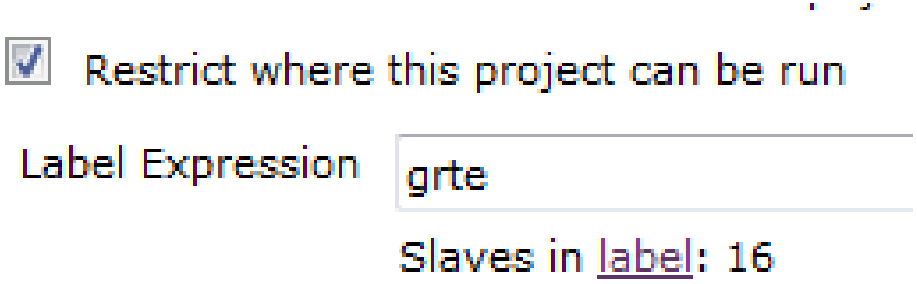
- If you don't specify it will default to “utilizing this node as much as possible”

How do you decide which jobs go to which nodes?

- If don't have to specify, Jenkins will choose a node with executors available

- Can specify in the job itself

  Choose label or node name



- Best to use naming standards and assign “labels” to your nodes to group or categorize them

- When you create nodes you can determine what can run on them as well

Plugins for Job Node Assignment

- Various plugins allow for UI options to choose the “label” or “Node” to run the job against
  (NodeLabel Parameter Plugin)

- Groovy script can decide as well based on user parameters
  (Groovy Label Assignment Plugin)

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";
}
```

NodeLabel Parameter Plugin - Node

Add Parameter

Node

Name

Default nodes

```
master
DEVELOPMENT_GRTE
DEVELOPMENT_GRTE2
DEVELOPMENT_GRTE3
DEVELOPMENT_INOTES
```

The nodes used when job gets triggered by anything else then manually

Possible nodes

```
ALL (no restriction)
master
DEVELOPMENT_GRTE
DEVELOPMENT_GRTE2
DEVELOPMENT_GRTE3
```

The nodes available for selection when job gets triggered manually

- Run next build only if build succeeds
- Run next build only if build succeeds or is unstable
- Run next build regardless of build result
- Allow multi node selection for concurrent builds
- Disallow multi node selection when triggering build manually

In case of multi node selection, should the next build on the next node be triggered only for successful builds, etc.?

Node eligibility: All Nodes

Description

NodeLabel Parameter Plugin - Label

Label

Name

Default Value

☐ Run on all nodes matching the label

Description

Jenkins
User Conference

Security Of Jenkins

#jenkinsconf

- Different ways you can lock down your Jenkins server
- Think about who you want to have access
- How you control your access and ensure the right people have access
- Look at using groups to control access
  - **JenkinsRead** – can only read and see and search and login
  - **JenkinsDevelopment** – can create new jobs, configure jobs, create nodes
  - **JenkinsAdmin** – Administer the Jenkins server
- Global Security trumps job based security

Matrix Authorization Strategy Plugin – global level

Jenkins User Conference

#jenkinsconf

User/group	Overall					Credentials				
	Administer	Read	RunScripts	UploadPlugins	ConfigureUpdateCenter	Create	Update	View	Delete	ManageDomains
JenkinsAdmin	✓									
JenkinsDev		✓	✓			✓	✓	✓	✓	✓
JenkinsRead		✓								
Anonymous		✓								

Slave						Job							Run		View				SCM
Configure	Delete	Create	Disconnect	Connect	Build	Create	Delete	Configure	Read	Discover	Build	Workspace	Cancel	Delete	Update	Create	Delete	Configure	Read	Tag

- Global trumps project based
- Groups allow for adding/removing people easily
- There are a lot of options, having Admins manually adding people can get tedious
- Lock down Anonymous!

Matrix Authorization Strategy Plugin – job level

Jenkins User Conference

#jenkinsconf

☑ Enable project-based security

☐ Block inheritance of global authorization matrix

User/group	Job							Run		SCM
	Delete	Configure	Read	Discover	Build	Workspace	Cancel	Delete	Update	Tag
Anonymous	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

User/group to add: [ ] Add

- Teams can set up their own groups to control access to their jobs
- Makes it easy to add/remove people without updating the job itself
- Teams own their jobs, don't need Admin to update access

Jenkins
User Conference

Source Control for Jobs

#jenkinsconf

- Can store your job config, scripts, shell scripts,batch files and artifacts your job needs in source and pull from source at job runtime
- Should try not to put script code directly into the jenkins job and instead reference a file in the workspace
  - There are exceptions to this, when you don't need to put a script in source
- Can use the source control integration to deploy these artifacts at build time to the workspace and then use them to execute your build
- Having a “jenkins-config” allows you to track and make changes to your jenkins jobs!
- Set up your jobs source code management section to pull code from this repository

Implementing source control for jenkins job scripts

Create a jenkins job to build this source project:
Name: Admin Update Pipeline Config

Source Code Management

- ○ None
- ○ CVS
- ○ CVS Projectset
- ○ ClearCase UCM
- ○ Git
- ◉ Rational Team Concert (RTC)
- ☐ Override global RTC repository connection (https://swgjazz.ibm.com:8010/jazz)
- ○ Use a build definition for better RTC integration
- ◉ Just accept and fetch from a build workspace

Build workspace: Sadmin Pipeline Workspace

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitHub
- ☐ GitHub Pull Request Builder
- ☑ Poll SCM

Schedule: H/15 * * * *

Add build step

Post-build Actions

Archive the artifacts

Files to archive: jenkins-config/,jenkins-config/grte/admin/vmwarescripts/,jenkins-config/grte/admin/graphingtool/

Pull the scripts from source in your other jenkins jobs

Jenkins User Conference

#jenkinsconf

Now to use these scripts directly from source

Copy artifacts from another project

Project name: Admin Update Pipeline Config

Which build: Latest successful build

☐ Stable build only

Artifacts to copy: **/production/**, **/cdl/**,**/grte/admin/svt/**

Artifacts not to copy:

Target directory:

Parameter filters:

☐ Flatten directories ☐ Optional ☑ Fingerprint Artifacts

Execute shell

Command

```
#!/bin/sh
echo $Node

sh "${WORKSPACE}/jenkins-config/grte/admin/production/prodseq.sh"
```

See the list of available environment variables

Pulling job parameter options from source!

Extended Choice Parameter

Name: Operating System

Description: Operating System to reserve for the run

(•) Simple Parameter Types

Parameter Type: Single Select

Number of Visible Items: 5

Delimiter: ,

Quote Value: [ ]

Choose Source for Value

( ) Value

Value:

(•) Property File

/var/lib/jenkins/jobs/Admin Update Pipeline Config/workspace/jenkins-config/grte/admin/jobparams.properties

Property Key: operatingsystem

Pull groovy scripts directly from source!

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");
```

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

Some useful groovy snippets!

Most of these scripts will require some imports such as:

```
import jenkins.model.*
import hudson.model.*
import hudson.slaves.*
```

```
//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


```
///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)

        }

    }
}
```

Create Reports and links for your jobs

- Can create quick links and email reports for the job that ran
- Use Groovy to loop through subbuilds and their artifacts and results and expose the data you want
- Use Groovy Postbuild plugin or Groovy system scripts

Groovy Quick Links with Groovy Postbuild Plugin

Jenkins User Conference

#jenkinsconf

```
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")
}

//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")
    }
}
```

Groovy Quick Links with Groovy System Script

Assumes you already created a build variable and that you have a function similar to previous page that creates this GroovyPosbuildSummaryAction object

```
import org.jvnet.hudson.plugins.groovypostbuild.GroovyPostbuildSummaryAction

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

Come see my session “Getting Groovy with Jenkins” to learn more about the fun groovy scripts you can create!

PLUGINS!

Build Time-out Plugin

Abort the build if it's stuck

Time-out strategy

Absolute

Timeout minutes 1440

Elastic

Likely stuck

No Activity

Time-out variable

Set a build timeout environment variable

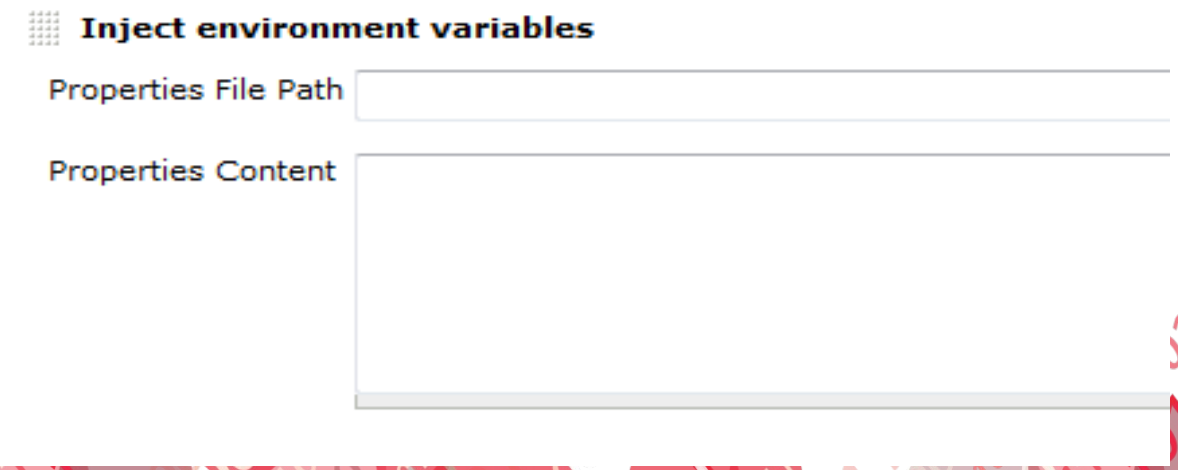
Time-out actions

Abort the build

Add action

Environment Injector Plugin

- What happens in a build step is scoped to that step
- Not obvious how to pass environment params across build steps
- This plugin lets you do this
- In one build step, write out parameters or things you want to save to a properties file in the WORKSPACE
- Then in the next build step use the Inject environment variables option

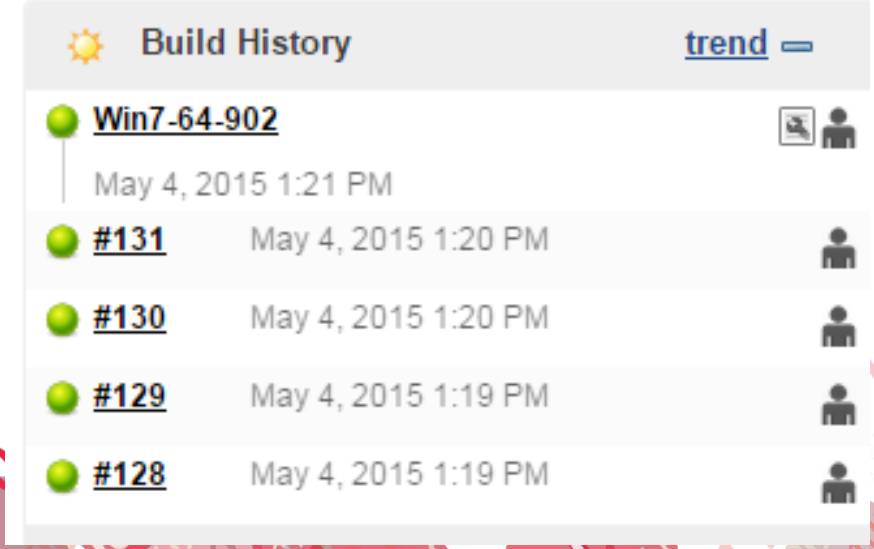


Build-Name Setter Plugin

Set Build Name

Build Name ${ENV,var="OperatingSystem"}-${ENV,var="Stream"}

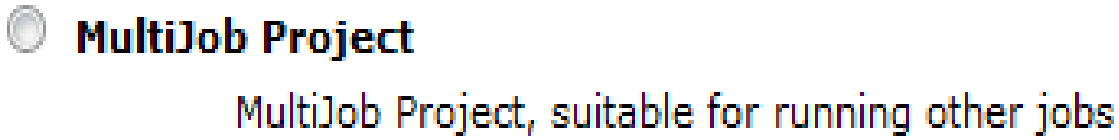
- Operating System & Stream are build parameters to this job
- Changes what you see in the build history!
- Supports environment variables and parameters
- Especially helpful for builds with parameters



Multi Job Plugin

Phases are sequential whilst jobs inside each Phase are parallel

When creating new Jenkins job you have an option to create MultiJob project.



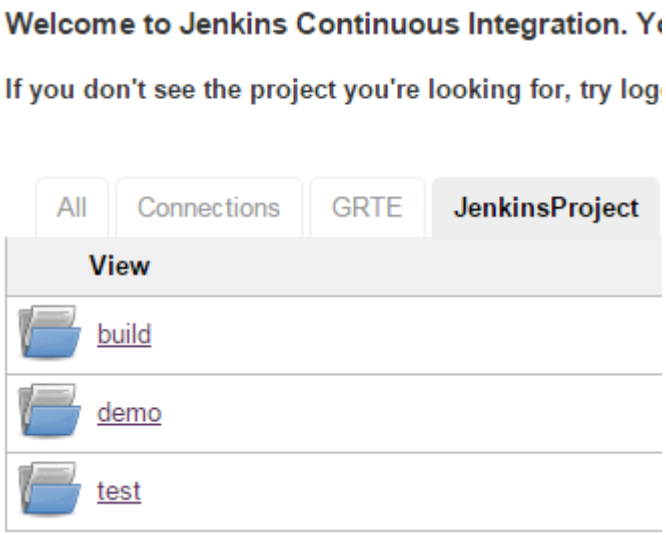
This job can define in the Build section phases that contains one job or more.

- All jobs belong to one phase will be executed in parallel (if there are enough executors on the node)

- All jobs in phase 2 will be executed only after jobs in phase 1 are completed etc.

Nested View Plugin

- Nested Folders within a view
- Hide views and jobs from other areas



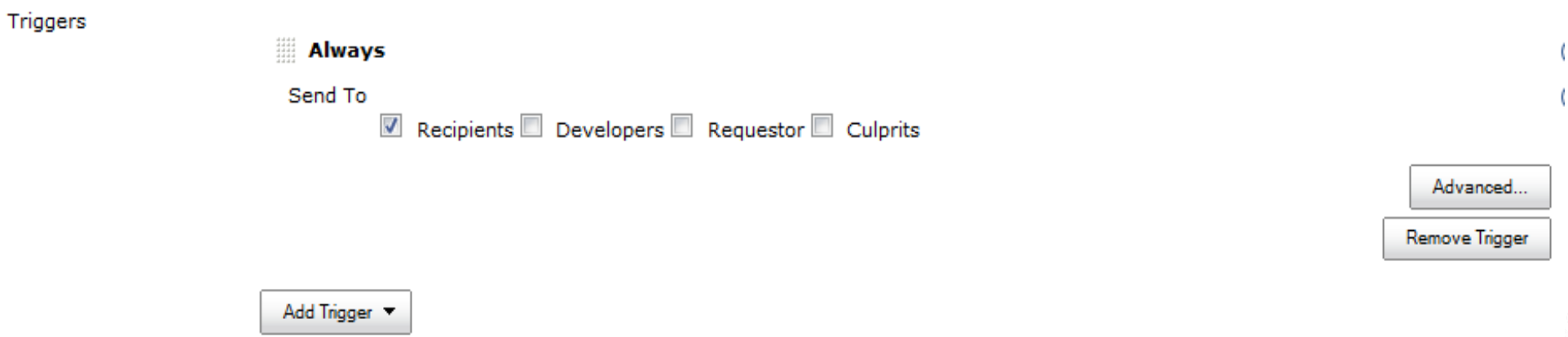
Nested View Plugin – click into a “nested folder”

Email-ext plugin

- Advanced settings gives you the ability to execute some groovy code to modify the email based on the build that executed

- Can use environment variables in the subject and body of the email to be sent

- Set up triggers to determine when to send the email and to who



Email-ext plugin

Post-build Actions

Editable Email Notification

Disable Extended Email Publisher ☐

Allows the user to disable the publisher, while maintaining the settings

Project Recipient List

lmccollu@us.ibm.com

Comma-separated list of email address that should receive notifications for this project.

Project Reply-To List

lmccollu@us.ibm.com

Comma-separated list of email address that should be in the Reply-To header for this project.

Content Type

Plain Text (text/plain)

Default Subject

${ENV,var="Build"} BVT Results on ${ENV,var="Operating System"} for ${ENV,var="Action"}

Default Content

```
Jenkins Results for ${ENV,var="Stream"}
Operating System ${ENV,var="Operating System"}
Action taken on stream was: ${ENV,var="Action"}

Jenkins URL to GRTE Debug for this run: ${BUILD_URL}
Jenkins Build Number: ${ENV,var="BUILD_NUMBER"}.txt

Result of GRTE Job: ${BUILD_STATUS}
Here is the contents of the BVT_Results.txt

${FILE,path="$BUILD_NUMBER.txt"}
```

Email-ext plugin

advanced (can use a script in source too!)

#jenkinsconf

Jenkins User Conference

Pre-send Script

```
def config = new HashMap()
// job vars
def buildMap = build.getBuildVariables()
config.putAll(buildMap)

// build/environmental variables
def envVarsMap = build.parent.builds[0].properties.get("envVars")
config.putAll(envVarsMap)

def pathtoworkspace = config.get("WORKSPACE")
def number = config.get("BUILD_NUMBER")
def xpdbuild = config.get("Build")
def stream = config.get("Stream")
def os = config.get("Operating System")
def action = config.get("Action")
logger.println("Working with build: " + number + ".txt")

def results = build.getWorkspace().child(number + ".txt").readToString()
if(!results.contains("PASSED")){
logger.println("Something wrong with results on workspace")
logger.println(results)
}
def pass = 0
def fail = 0
def skip = 0
def lines = results.split("\n")
for (int i = 0; i < lines.size(); i++) {
if(lines[i].contains("PASSED")){
    logger.println lines[i]
    pass = lines[i].split("PASSED")[0].trim()
  }else if(lines[i].contains("FAILED")){
    logger.println lines[i]
    fail = lines[i].split("FAILED")[0].trim()
  }else if(lines[i].contains("SKIPPED")){
   logger.println lines[i]
   skip = lines[i].split("SKIP")[0].trim()
  }
}

logger.println "PASS: " + pass
logger.println "FAILED: " + fail
logger.println "SKIPPED: " + skip

msg.setSubject(xpdbuild + " on " + os + " BVT Results: PASSED: " + pass + " FAILED: " + fail + " SKIPPED: " + skip)
```

Useful Plugins Covered in Presentation

- Build-Name Setter Plugin
- Conditional Build Step Plugin
- Downstream Build View Plugin
- Environment Injector Plugin
- Extended Choice Parameter Plugin
- Multi Job Plugin
- Groovy Plugin
- Parameterized Trigger Plugin
- Groovy Label Assignment Plugin
- Nested View Plugin
- Run Condition Plugin
- Show Build parameters Plugin
- View Job Filters Plugin
- Groovy PostBuild Plugin
- Junit Plugin
- Xunit Plugin
- Claim Plugin
- EnvInject Plugin
- TeamConcert Plugin
- Build User Vars Plugin
- Log Parser Plugin
- Flexible Publish Plugin
- Build-time-out Plugin
- Copy Artifact Plugin
- NodeLabel Parameter Plugin
- Email-ext plugin

Stuck? Need Help?

#jenkinsconf

- 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

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.