Groovy in Jenkins

Ioannis K. Moutsatsos

Repurposing Jenkins for Life Sciences Data Pipelining

Who Am I?

- Research scientist at local pharmaceutical company
- Software engineer
- Open Source advocate and contributor
  - Biouno.org, Jenkins.org
- Educator
  - Have taught graduate level courses at Brandeis University
  - Currently teaching a Groovy programming course at work
- Blogger (occasional)
  - http://imoutsatsos.blogspot.com/
  - http://biouno.org/blog
- Crafter, hobbyist and maker (Arduino,RasPi)

@ioannismou

Ioannis Moutsatsos

An Overview

Introduction

- What I do and how
- What makes Jenkins attractive for life-science use
- The BioUno Project
- Jenkins integration in a life-sciences HPC environment

Groovy in Jenkins

- Fundamentals
- Modes of operation
- Security
- Plugins and Usage
- Groovy Scriptlets.... everywhere!

Take home examples

Using **Jenkins as an artifact repository**

- Storing and re-using artifacts

Creating consistent **build reports**

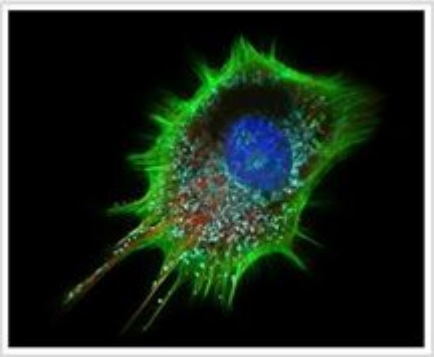
- Configuration and data sources

Creating **interactive Jenkins interfaces**

- Jenkins Active Choices plugins

High Throughput Screening: HTS

A high throughput drug discovery process



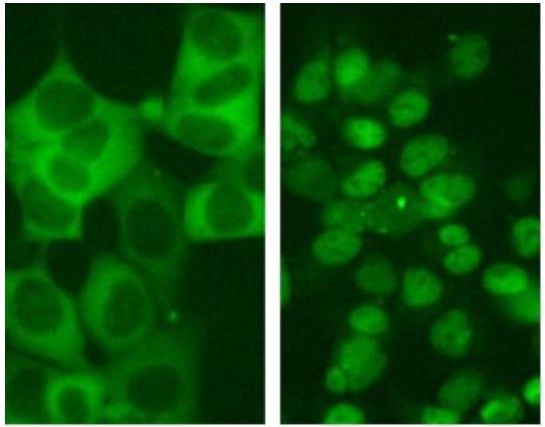
The cell

- One of the smallest reaction vessels
- Potentially contains all of the drug targets the pharmaceutical industry may want

My Current Focus: High Content Screening

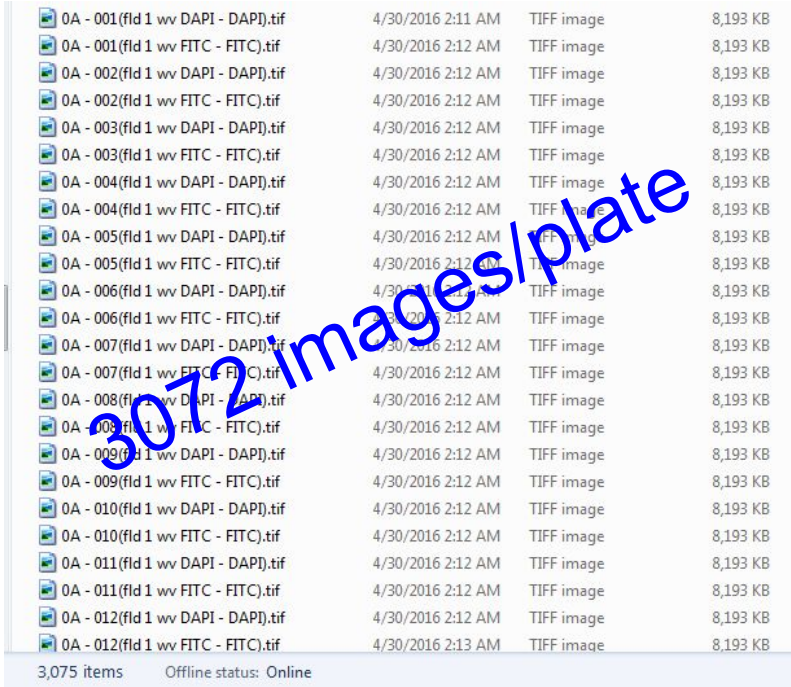
Use cellular imaging to identify new medicines

- Develop infrastructure to process large numbers of cellular images
- Develop imaging pipelines for identifying and quantifying image features
- Develop user friendly software for data processing and data review
- Use multi-parametric analysis methods to analyze image feature measurements (in the hundreds)



HEK293/tGFP-hGR cell line. The glucocorticoid receptor (labelled with a green fluorescence) moves from the cell cytoplasm to the nucleus. Image analysis allows us to measure this translocation.

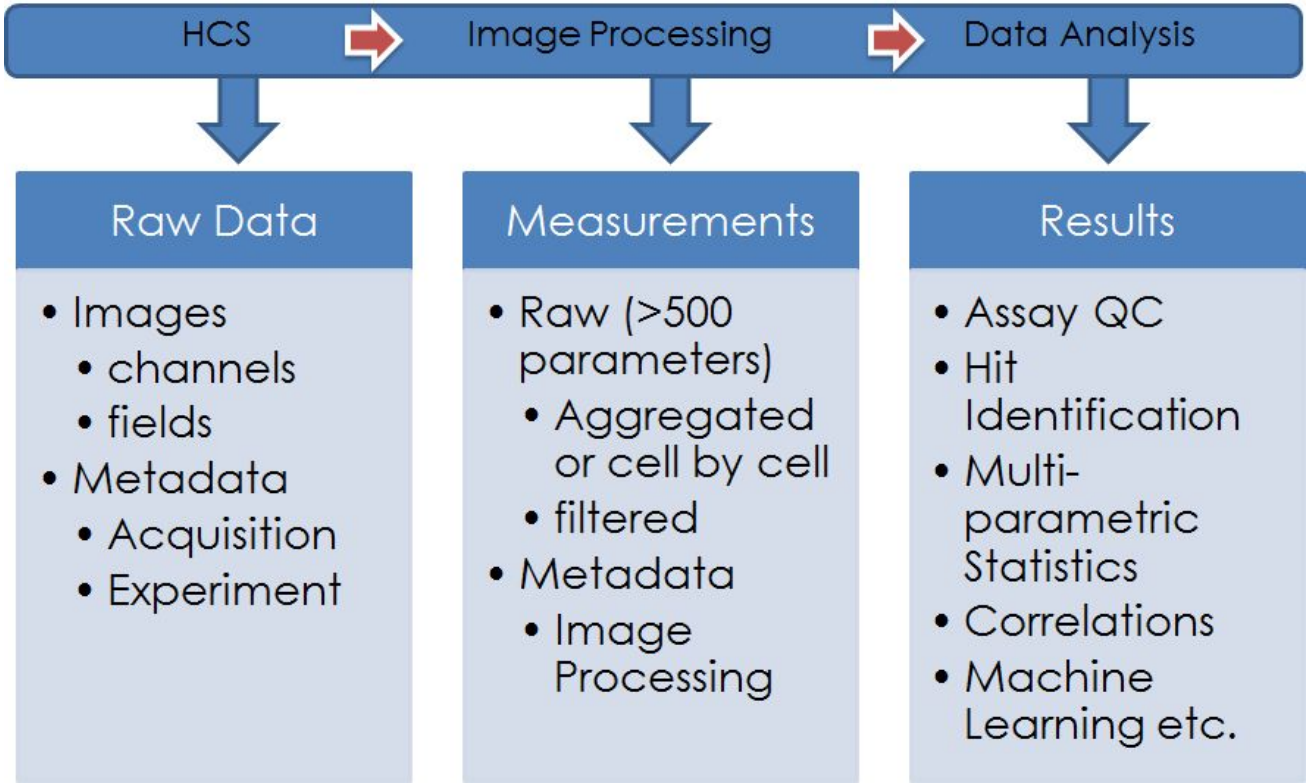
High Content Screening: Analysis Input



Structure of a typical image acquisition run in a screening assay

- Image files (typically TIF format)
- Multiple images per assay well
  - One per fluorescent marker used
- Typically 1536 wells/assay plate
- 50-100 assay plates/run

High Content Screening



My Current GoTo Tools

Image Analysis

- CellProfiler
- Icy
- Ilastik

Scripting & Workflow

- Groovy scripting
- Jenkins
- Linux cluster (for scaling up)

Data Analysis

- R-graphics & statistics
- H2 in memory database

Jenkins

From DevOps to Life-Sciences

Continuous Integration and Jenkins-CI

Continuous Integration (CI)

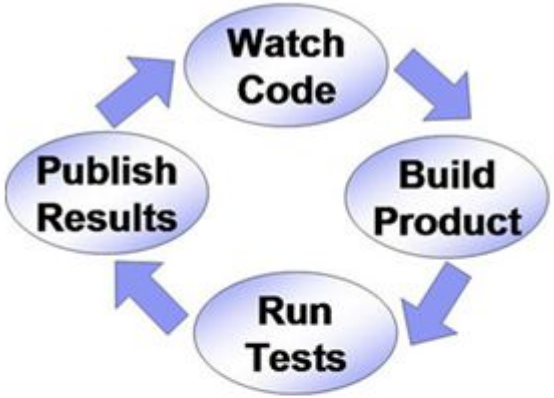
- A software development best practice for creating and testing executable code and software documentation

Jenkins-CI

- An open source continuous integration server

Who uses Jenkins-CI

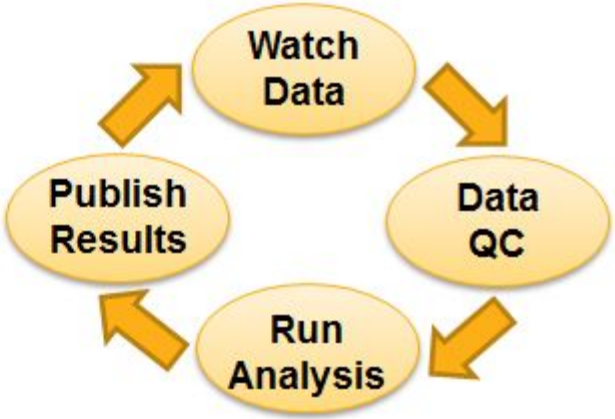
- DevOps teams throughout the industry



Jenkins

An extendable open source continuous integration server

Why use Jenkins for Life Science Applications?



Continuous Integration resembles typical Scientific Data Processing/Analysis

Why use Jenkins for Life Science Applications?

Jenkins-CI is

- Free and open-source
- Platform independent, and language agnostic
- Modular, expandable (over 1000 plugins), scalable
- Well-supported

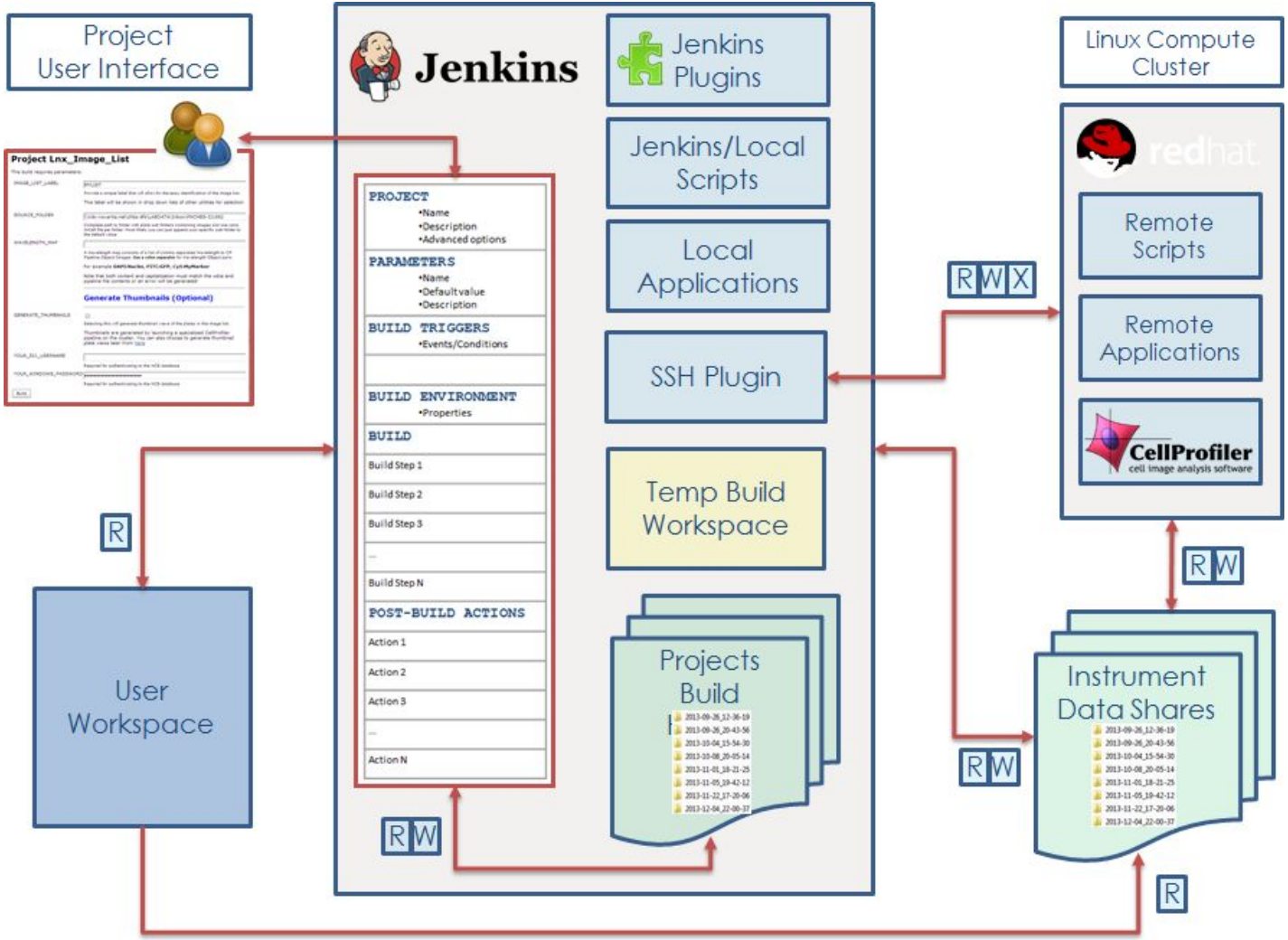
Jenkins-CI can serve as

- Web-portal for a variety of utilities, applications, and computational tools of interest to life-scientists
- An integration platform for a variety of bio/chem informatics packages
- Reproducible workflow platform
- Data management platform
- Collaboration platform

Jenkins Workflows=Data Processing Pipelines

Jobs can be chained to create modular pipelines

Jenkins integration in the HPC scientific environment



Groovy

Integration with Jenkins

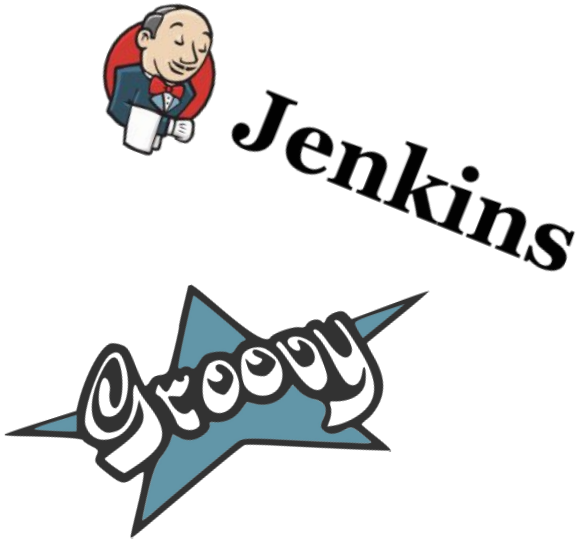
This section covers the basics of using Groovy in Jenkins.

I also demonstrate a generic way for adapting existing command line interface Groovy scripts to a Jenkins job with a web interface.

Groovy In Jenkins: The basics

Groovy Plugin supports Groovy script execution in **build** and **post-build** steps of Jenkins jobs

- Groovy scripts can run in a ‘forked’ JVM
- System Groovy scripts
  - Run in Jenkins JVM
  - Can control Jenkins
  - Can get internal Jenkins information, such as build parameters, artifacts etc.



Groovy in Jenkins: Security

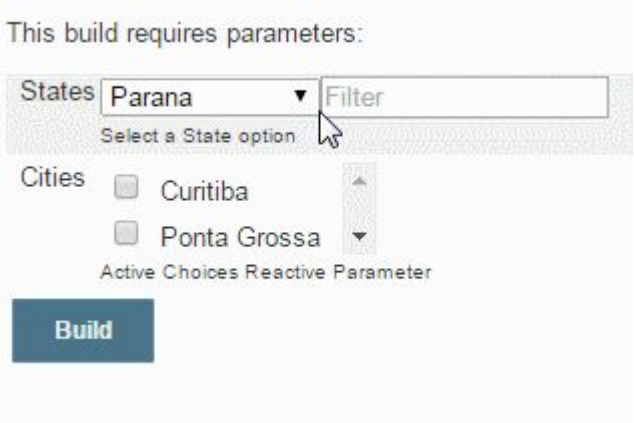
Script Security Plugin supports:

- Script **approval** via a globally maintained list of approved scripts
  - Scripts authored by admin auto-approved
  - Scripts authored by others require admin approval
- Groovy **sandboxing**
  - Groovy scripts can be run without approval so long as they limit themselves to operations considered inherently safe
  - Script Security Plugin provides a small default whitelist, and integrating plugins may add operations to that list

Plugins that use Groovy scripts

User Interface Plugins

- Active Choices Plugin
- Extended Choice Parameter Plugin



Build Steps Plugins

- Groovy Post-build Plugin
- Pipeline Groovy Plugin

Use Groovy Scripts as Build Steps

Any existing Groovy script can be easily reused as a Jenkins build step using the Groovy Plugin.

Here, I focus on scripts that were designed to be run from a command line interface (**CLI Groovy scripts**) by passing command line arguments

Script command line arguments can be captured as **Jenkins build parameters** and then used in a Groovy build step.

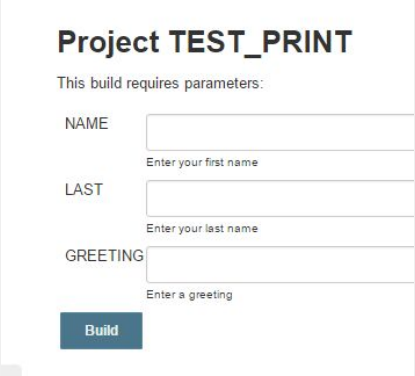
See here on how to use the `CliBuilder` class in Groovy : http://mrhaki.blogspot.com/2009/09/groovy-goodness-parsing-commandline.html

Even without a CLI interface any groovy script can accept arguments from the command line using an implicit **`args`** String array

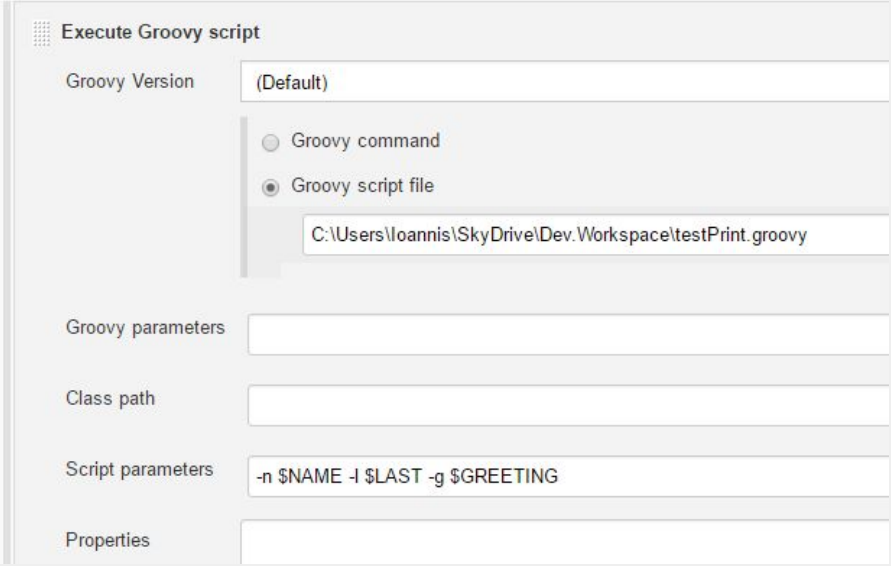
CLI Groovy Used as Jenkins Build Step

Example Groovy CLI and the corresponding Jenkins build form for collecting the required CLI arguments

```
C:\Users\Ioannis\SkyDrive\Dev.Workspace>groovy testPrint.groovy -h
error: Missing required options: n, l
usage: testPrint.groovy -n -l -g
 -g,--greet <greeting>   greeting
 -h,--help               Show usage information
 -l,--last <last>        last name to greet
 -n,--name <name>        name to greet
```



Jenkins Groovy build step configuration. Note the use of Jenkins build parameters as script arguments



Adapting CLI Groovy Scripts for Jenkins

Goal and Benefits

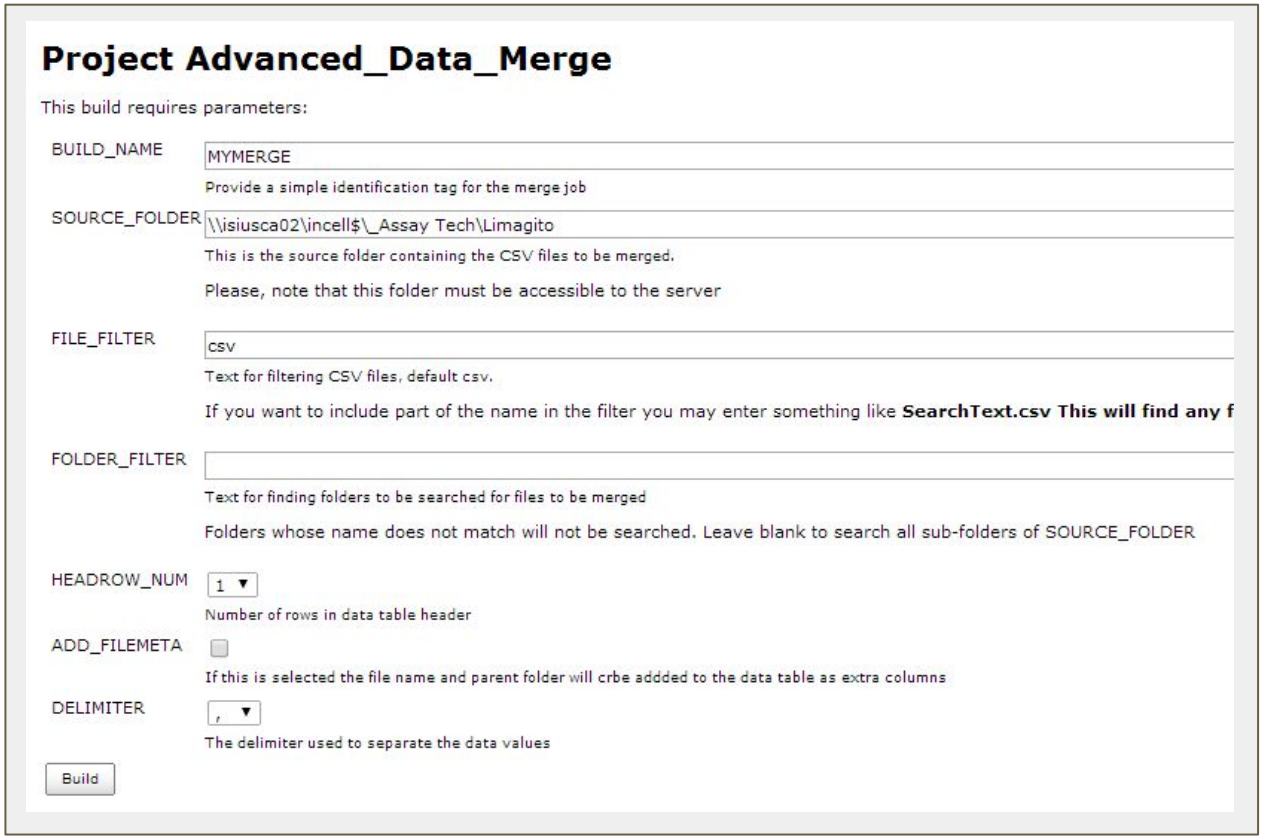
- Adapt any Groovy script with a **command line interface** so that it can be **executed on Jenkins**
  - See CLI script example on the right
- Users run script remotely
  - Nothing to install locally
- Script usability increases by providing a Web-UI for the script
  - Default values, on-line help etc

```
usage: advanceDataMerge.groovy -sd[fprmch]
 -c,--delimiter <value delimiter>      delimiter for data values, default:
                                       ,(comma)
 -d,--destination <destination file>   file where joined data will be
                                       written
 -f,--filter <file filter>             string for filtering files,
                                       default: csv
 -h,--help                             script merges data from multiple
                                       csv files to a new single file
 -m,--fileMeta                         includes file and folder names as
                                       extra columns
 -p,--parentFilter <parent filter>     string for filtering parent folder,
                                       default: all
 -r,--headerRows <header rows>         number of header rows, default: 1
 -s,--source <csv folder>              folder with separate csv files
```

Adapting CLI Groovy Scripts for Jenkins

Here is an example of the Jenkins Web-UI we can provide for the CLI Groovy script on the previous slide

- Note that the job parameter names will be exposed as **environment variables** during the build process



Adapting CLI Groovy Adaptor Script Example

Note the **helper method,** `getAdvanceDataMergeCLI`, that builds the argument list to the script command

- The required command line options and their values are formatted, validated and stored as a **`String[]`**
- A `String[]` is required for passing the argument list to the original CLI Groovy script

advanceDataMerge_build.groovy | genImageListThumbs_build.gro

```
 * The script gets parameterized from the environment variables
 *
 * Below are the CLI options for the main script
     h longOpt: 'help', 'script merges data from multiple csv files to a new single file'
     s longOpt: 'source', args:1, argName: 'csv folder', 'folder with separate csv files', r
     f longOpt: 'filter', args:1, argName: 'file filter', 'string for filtering files, defau
     p longOpt: 'parentFilter', args:1, argName: 'parent filter', 'string for filtering pare
     r longOpt: 'headerRows', args:1, argName: 'header rows', 'number of header rows, defaul
     m longOpt: 'fileMeta', args:0, argName: 'file metadata', 'includes file and folder name
     d longOpt: 'destination', args:1, argName: 'destination file', 'file where joined data
     c longOpt: 'delimiter', args:1, argName: 'value delimiter', 'delimiter for data values,
 */

def env = System.getenv()
def options= new HashMap()

options.s=env['SOURCE_FOLDER'].trim().replace('\\','/')
options.f=env['FILE_FILTER'].trim()
options.p=env['FOLDER_FILTER'].trim()
options.r=env['HEADROW_NUM'].toInteger()
options.m=env['ADD_FILEMETA'].toBoolean()
options.d="${env['WORKSPACE'].replace('\\','/')}/Merged.csv"
options.c=env['DELIMITER']

def thisAdvancedDataMerge= new advanceDataMerge()
println getAdvanceDataMergeCLI(options)
//now we call the main script with the args array
thisAdvancedDataMerge.main(getAdvanceDataMergeCLI(options))
/*
method to build the cli args String [] required for the advanceDataMerge script
Note the casting of the list to String[] required by the script main(String[]) method call
 */
def getAdvanceDataMergeCLI(options){
    def args=[] //args is defined as a list but will be returned as a String[]
//    The following are required options-Exit if user did not provide folder
    (options.s!='')?System.exit(1):args. add("-s${options.s}")
     args.add("-d${options.d}")

    //now we use a set of ternary conditionals to review/set the CLI options from the UI
    (options.f!='') ?args.add("-f${options.f}"):''
    (options.p!='') ?args.add("-p${options.p}"):''
    (options.r>1) ?args.add("-r${options.r}"):''
    (options.m) ?args.add('-m'):''
    (options.c!=',') ?args.add("-c${options.c}"):''
    return args as String[]
}
```

Adapting CLI Groovy
Adjust the classpath

Finally, we

Add the classpath of the original CLI script to the helper script's Groovy parameters

Now the original script can be used as a build step in Jenkins using the Groovy Plugin

- Configuration of build step shown on the right

Build

Execute Groovy script

Groovy Version: jenkinsGroovy

Groovy command

Groovy script file: C:\CWorkspace\Jenkins\IMAGE_DATA\Processing\advanceDataMerge_build.groovy

Groovy parameters: -cp C:\CWorkspace\JENKINS\IMAGE_DATA\Processing

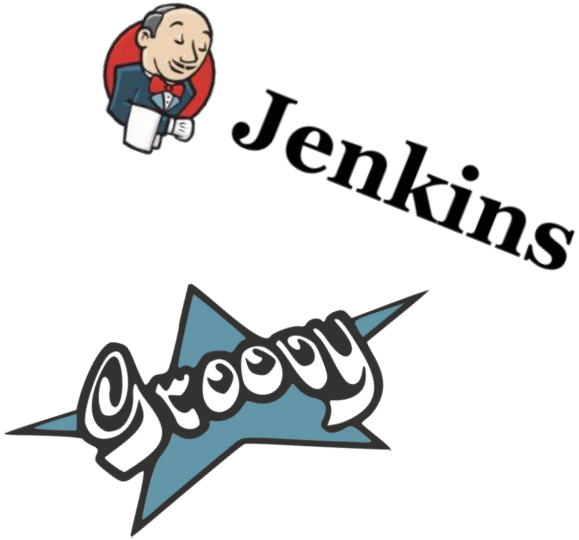
Class path:

Script parameters:

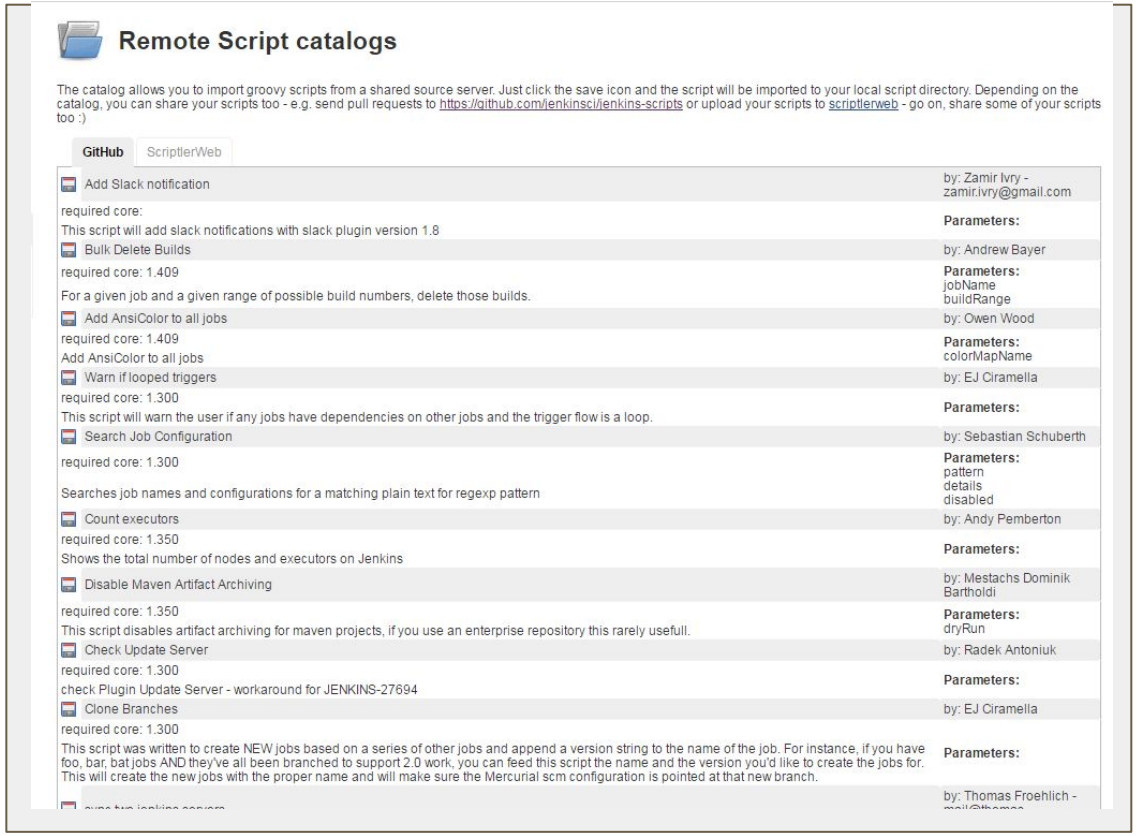
Groovy In Jenkins: Scriptler

Scriptler Plugin allows usage of **Groovy scripts** from a Jenkins managed **script catalog**

- Scripts are better organized and managed
- A Scriptlet web catalog allows script sharing
- Script security and parameterization are well defined



The Remote Scripts Catalog



Remote Script catalogs

The catalog allows you to import groovy scripts from a shared source server. Just click the save icon and the script will be imported to your local script directory. Depending on the catalog, you can share your scripts too - e.g. send pull requests to https://github.com/jenkinsci/jenkins-scripts or upload your scripts to scriptlerweb - go on, share some of your scripts too :)

GitHub | ScriptlerWeb

Add Slack notification required core: This script will add slack notifications with slack plugin version 1.8	by: Zamir Ivry - zamir.ivry@gmail.com **Parameters:**
Bulk Delete Builds required core: 1.409 For a given job and a given range of possible build numbers, delete those builds.	by: Andrew Bayer **Parameters:** jobName buildRange
Add AnsiColor to all jobs required core: 1.409 Add AnsiColor to all jobs	by: Owen Wood **Parameters:** colorMapName
Warn if looped triggers required core: 1.300 This script will warn the user if any jobs have dependencies on other jobs and the trigger flow is a loop.	by: EJ Ciramella **Parameters:**
Search Job Configuration required core: 1.300 Searches job names and configurations for a matching plain text for regexp pattern	by: Sebastian Schuberth **Parameters:** pattern details disabled
Count executors required core: 1.350 Shows the total number of nodes and executors on Jenkins	by: Andy Pemberton **Parameters:**
Disable Maven Artifact Archiving required core: 1.350 This script disables artifact archiving for maven projects, if you use an enterprise repository this rarely usefull.	by: Mestachs Dominik Bartholdi **Parameters:** dryRun
Check Update Server required core: 1.300 check Plugin Update Server - workaround for JENKINS-27694	by: Radek Antoniuk **Parameters:**
Clone Branches required core: 1.300 This script was written to create NEW jobs based on a series of other jobs and append a version string to the name of the job. For instance, if you have foo, bar, bat jobs AND they've all been branched to support 2.0 work, you can feed this script the name and the version you'd like to create the jobs for. This will create the new jobs with the proper name and will make sure the Mercurial scm configuration is pointed at that new branch.	by: EJ Ciramella **Parameters:**
	by: Thomas Froehlich -

- The Jenkins **Remote Script Catalog** is available to all Jenkins installations
- It provides many useful ‘system’ scripts for discovery, maintenance and administration
- DevOps oriented

Configuring a Jenkins Groovy Scriptler

Jenkins ▸ Scriptler

Edit Script

Id: AC_helper_detectFileType.groovy

if changed, the script gets copied

Name: AC_helper_detectFileType

Comment: Identifies a file extension and returns a fileType label

Permission ☑

Allow execution by user with RunScripts permission

Scriptler code @github

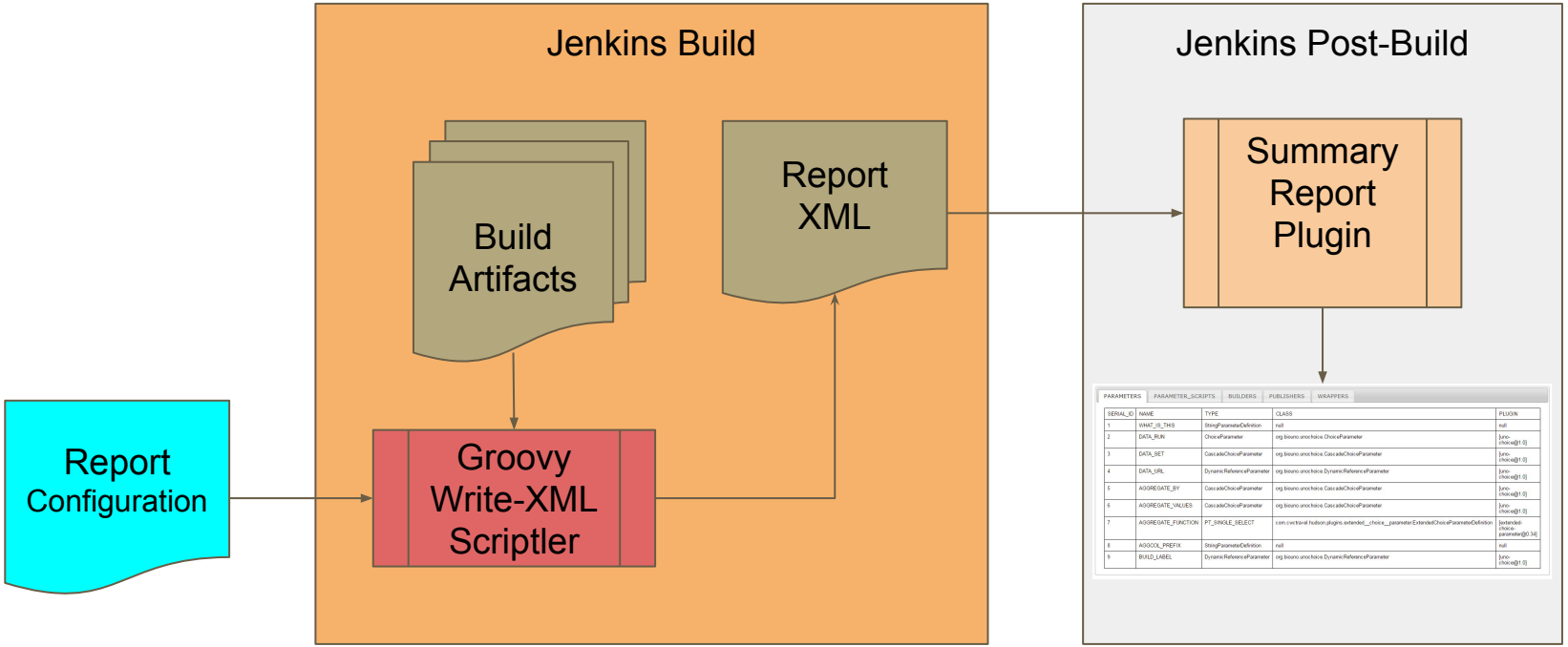
By including a script 'META'' header, a scriptlet can be 'auto-discovered' when placed in the scriptler/scripts folder in Jenkins home

Build report

Dynamic Consistency through Configuration

We discuss a general way of creating attractive build reports by adapting the flexibility of the XML Summary Report with the convenience of a Groovy generated XML report template

Data Flow for Report Generation



SERIAL_ID	NAME	TYPE	CLASS	PLUGIN
1	WHAT_IS_THIS	StringParameterDefinition	null	null
2	DATA_RUN	ChoiceParameter	org.biourno.uroc.choice.ChoiceParameter	juno-choice@1.0]
3	DATA_SET	CascadeChoiceParameter	org.biourno.uroc.choice.CascadeChoiceParameter	juno-choice@1.0]
4	DATA_URL	DynamicReferenceParameter	org.biourno.uroc.choice.DynamicReferenceParameter	juno-choice@1.0]
5	AGGREGATE_BY	CascadeChoiceParameter	org.biourno.uroc.choice.CascadeChoiceParameter	juno-choice@1.0]
6	AGGREGATE_VALUES	CascadeChoiceParameter	org.biourno.uroc.choice.CascadeChoiceParameter	juno-choice@1.0]
7	AGGREGATE_FUNCTION	PT_SINGLE_SELECT	com.cwctravel.hudson.plugins.extended_choice.__parameter.ExtendedChoiceParameterDefinition	[extended-choice-parameter@0.34]
8	AGGCOL_PREFIX	StringParameterDefinition	null	null
9	BUILD_LABEL	DynamicReferenceParameter	org.biourno.uroc.choice.DynamicReferenceParameter	juno-choice@1.0]

Example Report Configuration

```
# A properties file for report of UTIL_CONFIG_PARSER
# 4/19/2016

summary.properties=none
report.style=tab

tab.header=SCM,PARAMETERS,PARAMETER_SCRIPTS,BUILDERS,PUBLISHERS,WRAPPERS

field.key.color=black
field.value.color=blue

content.SCM=table
table.data.SCM=pscmProps.csv


content.PARAMETERS=table
table.data.PARAMETERS=paramProps.csv
table.header.PARAMETERS=SERIAL_ID,NAME,TYPE,CLASS,PLUGIN

content.PARAMETER_SCRIPTS=table
table.data.PARAMETER_SCRIPTS=paramProps.csv
table.header.PARAMETER_SCRIPTS=SERIAL_ID,NAME,TYPE,SCRIPTLET,SCRIPTLET_LINK,CODE_LINK,PLUGIN

content.BUILDERS=table
table.data.BUILDERS=builderProps.csv

content.PUBLISHERS=table
table.data.PUBLISHERS=publisherProps.csv

content.WRAPPERS=table
table.data.WRAPPERS=wrapperProps.csv
```

Report Configuration

- Defines report layout properties
  - Tabs
  - Colors
  - Max range of data to display
  - Size of graphics
- Defines data sources
  - Delimited files
  - Property files
- Defines data selectors-Query Criteria
  - Columns
  - Rows
  - Properties

Summary Report: Example 1

Build OS_CELLPROFILER_JENKINS (Jun 20, 2016 10:07:05 PM)

Build Artifacts

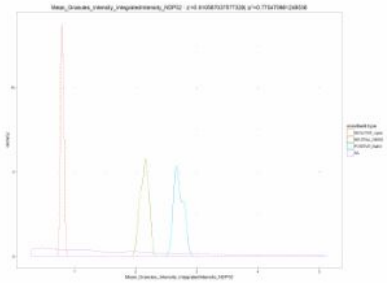
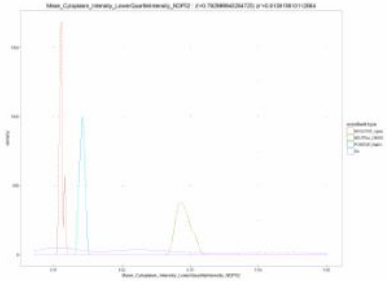
builderProps.csv	1.35 KB	view
config.xml	17.66 KB	view
paramProps.csv	807 B	view
publisherProps.csv	121 B	view
wrapperProps.csv	135 B	view
writeXmlSummary.xml	15.33 KB	view

No changes.

Started by user Ioannis Moutsatsos

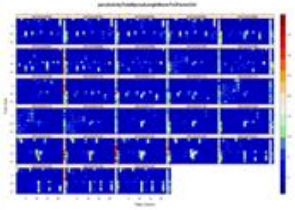
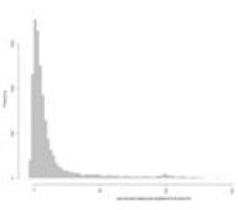
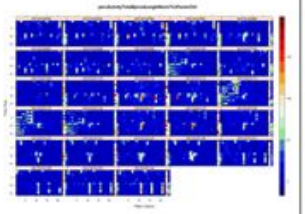
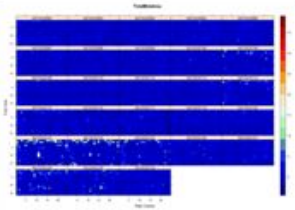
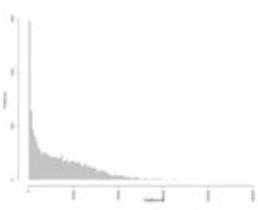
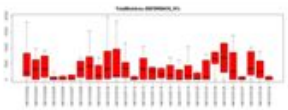
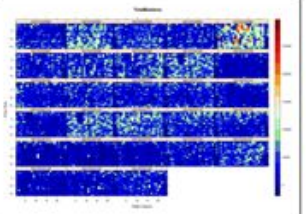
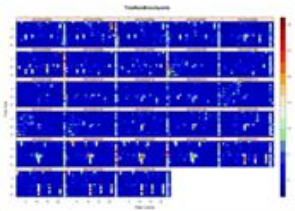
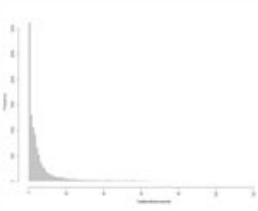
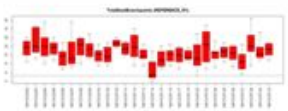
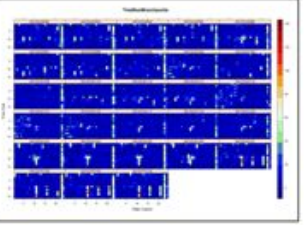
Summary Report: Example 2

PARAMETER.KERNEL.DENSITY | ACTIONS

QC_GRAPH	DataColumn	z.prime ↑	z.robust.prime
	Mean_Granules_Intensity_IntegratedIntensity_NDP52	0.810567037577328	0.770470961249536
	Mean_Cytoplasm_Intensity_LowerQuartileIntensity_NDP52	0.792999940284725	0.813815610112684
	Mean_Granules_Intensity_IntegratedIntensityEdge_NDP52	0.79097527579581	0.740298250030525

Summary Report: Example 3

GRAPH.COMPARISON | ACTIONS

DataColumn	PP_LOESS_NORMALIZATION_11	PP_DATASET_HISTOGRAM_25	PP_DATASET_QC_24	PP_DATASET_QCHEATMAP_11
percActivityTotalSproutLenghtNormTo3FactorCtrl				
TotalBlobArea				
TotalNumBranchpoints				

A Dynamic Build UI

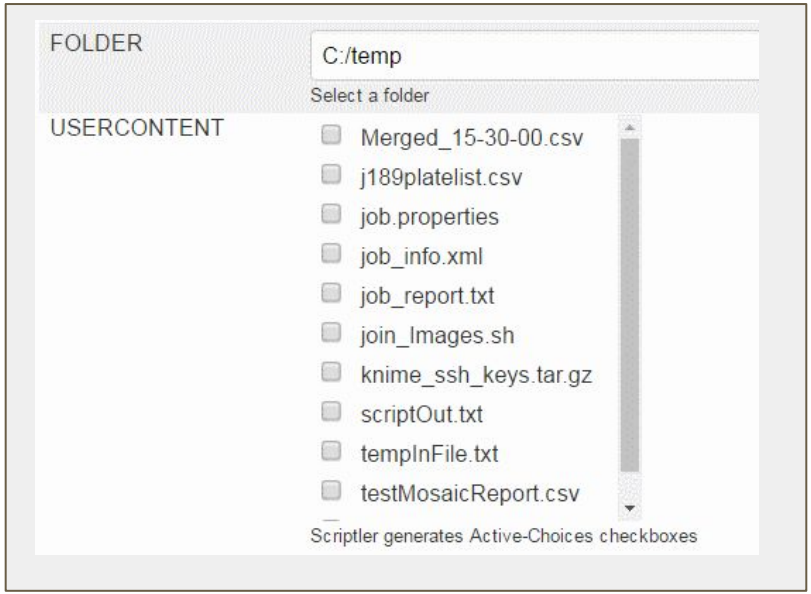
Active Choices

We discuss the Active Choices plugin, which provides dynamic and cascading build parameters via Groovy Scripts returning lists, maps or dynamic HTML

Groovy in Jenkins: Build UI & parameters

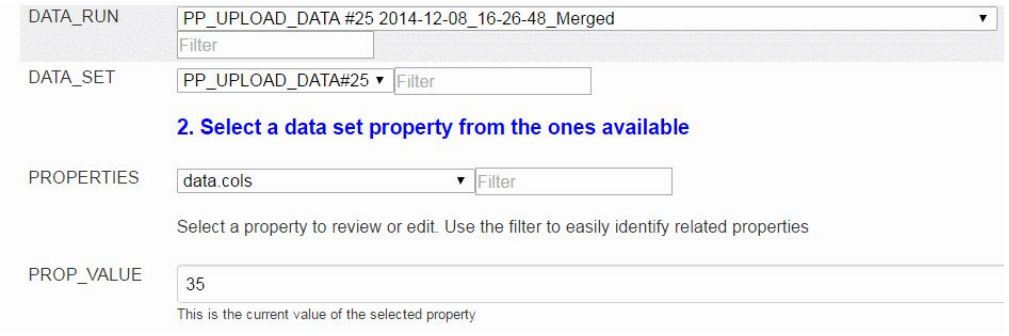
Active Choices and other plugins use Groovy/Scriptlet plugin to extend Jenkins functionality in several areas

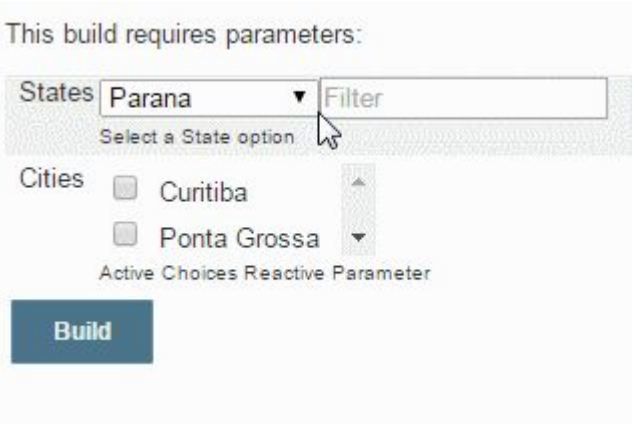
- Job UI/parameter plugins generate dynamic parameters with Groovy scripts
- Examples
  - Active Choices (contributed by BioUno)
  - Extended Choice parameter



USERCONTENT is an example Active Choice **cascading parameter** that dynamically generates file check box options from a scriptler upon changes in the FOLDER parameter

Other examples





Re-Using Build Artifacts

Jenkins as an object repository

We discuss ways in which build artifacts from one Jenkins project can be referenced and re-used as input to other Jenkins projects. This is a typical scenario in data analysis workflows/pipelines

CellProfiler Image Analysis: Overview

Example: Want to configure CellProfiler execution as a Jenkins job. How do I provide the required input image processing instructions (imaging pipeline)and image list to Cellprofiler?

Image processing Instructions

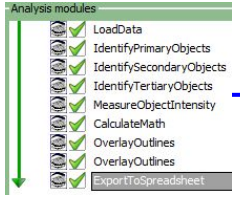


Image Sets

Image_FileName_mCherry	Image_FileName_DNA	Image_FileName_GFP
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...
\\sisiusc02.na.novartis.net...	\\sisiusc02.na.novartis...	\\sisiusc02.na.novar...

CellProfiler
cell image analysis software

Nuclei Intensity_MaxIntensity_GFP	Nuclei Intensity_MeanIntensityEdge_GFP	Nuclei Intensity_MeanIntensity_GFP	Nuclei Intensity_MedianIntensity_GFP	Nuclei Intensity_MinIntensityEdge_GFP	Nuclei Intensity_MinIntensity_GFP	Nuclei Intensity_StdIntensityEdge_GFP
0.04664683	0.006170534	0.017489066	0.012344548	0.003631647	0.003631647	0.001446654
0.032074463	0.00505514	0.011642756	0.0081178	0.003555352	0.003555352	0.000750723
0.021164263	0.005492656	0.010624631	0.00982681	0.003707942	0.003707942	0.000940625
0.019775692	0.005353831	0.009164943	0.008826246	0.003723201	0.003677424	0.000939074
0.022786979	0.009912294	0.014347105	0.014175631	0.00608835	0.00608835	0.003467233
0.032440681	0.0079314	0.01621422	0.017578393	0.00436408	0.003952087	0.00413396
0.022583352	0.006742668	0.010969913	0.010330358	0.003662165	0.003524834	0.00102081
0.022552835	0.005744618	0.01209347	0.012451362	0.003555352	0.003555352	0.000926871
0.042236973	0.007074873	0.020960123	0.021820402	0.005432212	0.005371176	0.001252574
0.029863979	0.004953841	0.012587891	0.009452964	0.00328069	0.00328069	0.000914862
0.00608835	0.004228981	0.004205293	0.004226749	-0.00296025	0.002563516	0.00043695
0.016327154	0.005238961	0.009635143	0.009811551	0.002517739	0.002517739	0.001610524
0.005598543	0.003626967	0.003655402	0.003662165	0.00250248	0.002441444	0.000413185
0.005798428	0.004157254	0.004170794	0.004180972	-0.00250248	0.00250248	0.000436837
0.006515603	0.00420632	0.004186953	0.004196231	0.003036545	0.002304112	0.000451144
0.005081255	0.003725654	0.003719411	0.003723201	0.00250248	0.002410925	0.000420831
0.00578317	0.00424704	0.004198501	0.004226749	0.002777142	0.002777142	0.000454746
0.005279602	0.003767219	0.003751621	0.003753719	0.002456703	0.002456703	0.000433736
0.005081255	0.003566214	0.003573766	0.003570611	-0.00205507	0.002258335	0.000404095
0.005676356	0.004073144	0.004079126	0.004089418	0.002670329	0.002319371	0.000423776

INPUT	Image Processing	Mutli-Parametric Measurements

Providing input for Jenkins builds

- A typical Jenkins build usually **checks code out of a code repository**
  - This is a **dedicated initial phase** for almost ALL DevOps related builds

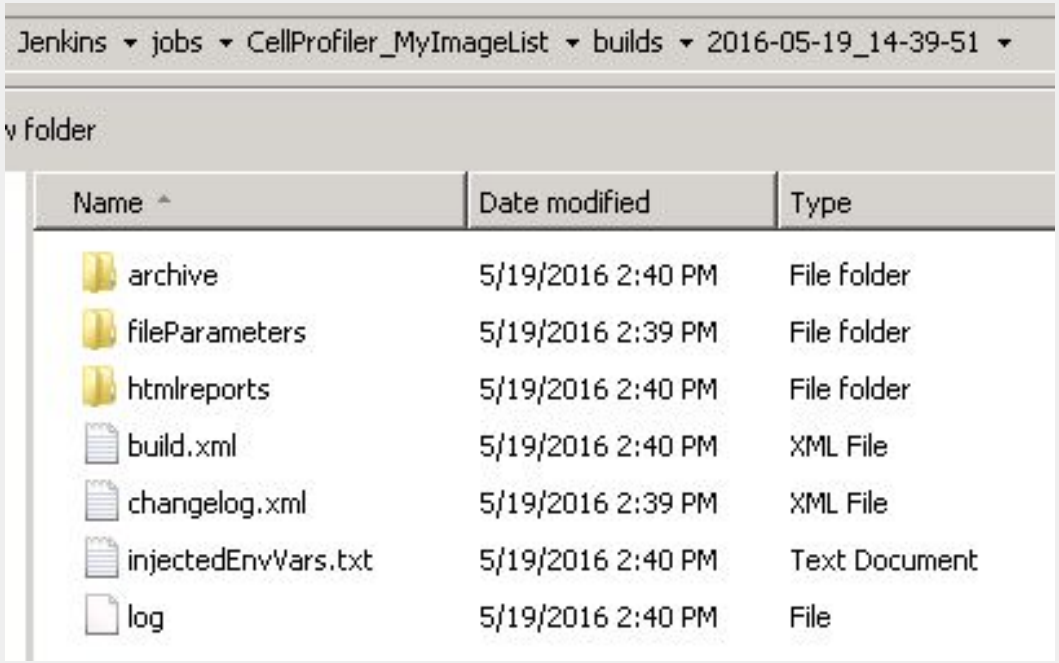
Typical Source Code Configuration in Jenkins

Scientists do not store data and files in code repositories. What options do we have?

- Upload data from user's desktop
  - Jenkins has a 'file parameter' type that allows users to upload data directly
- Read data from network shares
  - Somewhat 'cludgy' to access and browse
- Store data on Jenkins as build artifacts.
  - In this case **Jenkins acts as the data repository**

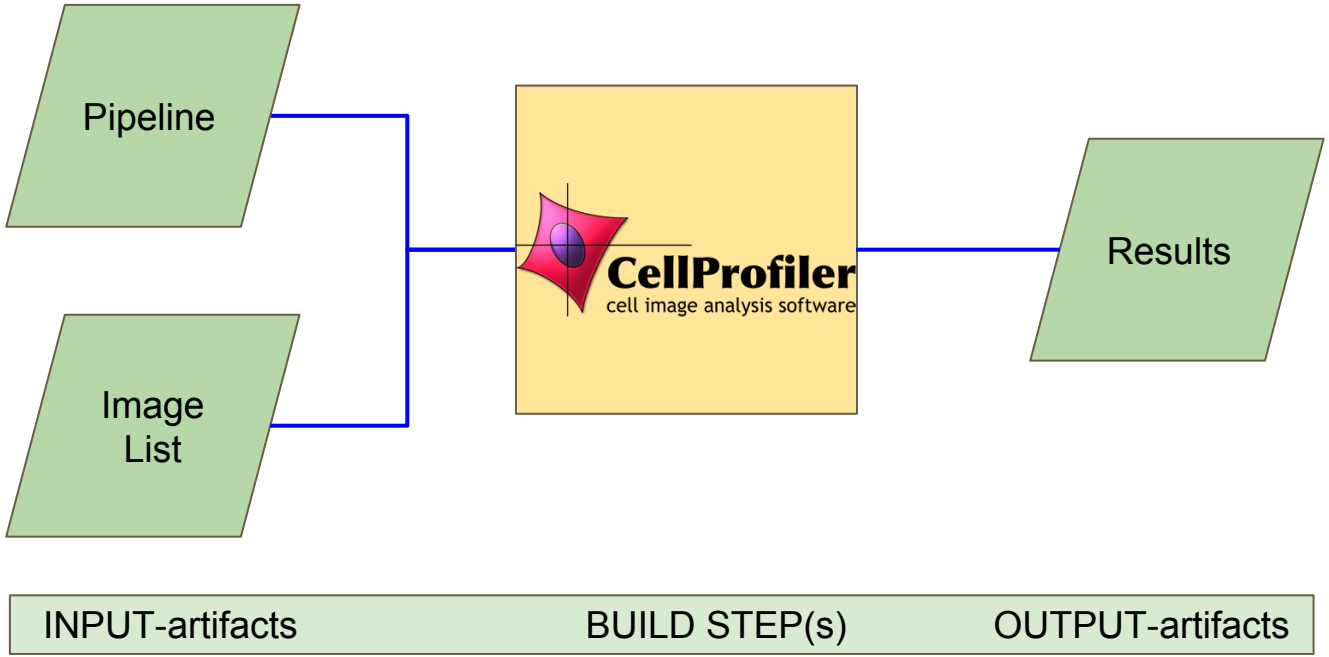
A Jenkins repository: Reusing build artifacts

- Jenkins **build artifacts** are archived on the file system in a build-specific folder
  - Typical example shown on the right
- Build **metadata** are stored in a build-specific file in xml format
  - The build.xml file
- Jenkins **does not use a database**
  - Instead, on startup it parses the build.xml files and builds an in memory model of the builds that can be accessed via the **Jenkins Java API.**



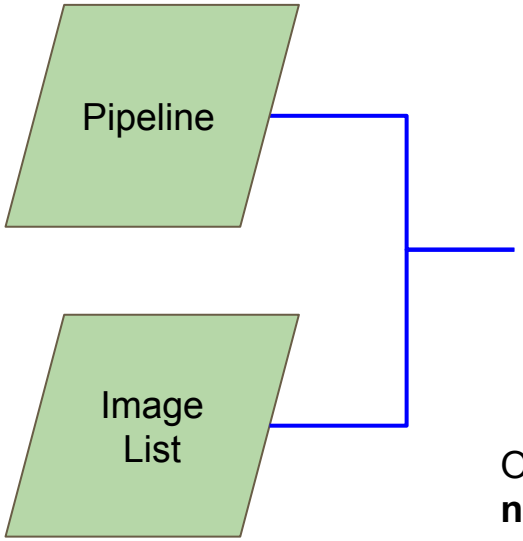
Typical structure of a Jenkins build folder

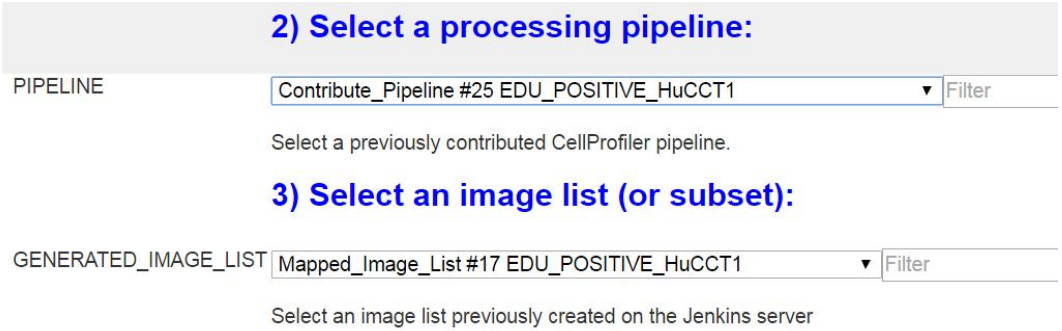
Wrapping CellProfiler in a Jenkins Build



Selecting input on the build UI

Using Active Choice parameters, **Pipeline** and **Image List** artifacts from previous builds can be referenced and used as input in this project





Options for the required input parameters display **references (build names) of previous builds.**

- **Active Choice parameters display build references** using the UC_helper_GetBuildsByNumAsMap scriptlet (see slide 42)

Get a Jenkins build reference: ‘Run Type’ Plus

- A Jenkins ‘Run Type’ parameter provides a reference to a previous build.
  - However it has limited capabilities. For example it can reference ony one project
- An Active Choice parameter using the `UC_helper_GetBuildsByNumAsMap` provides a build reference with advanced options
  - It can select from more than one jobs
  - It can select a range of builds by number
  - It can select builds by status
    - SUCCESS status by default

Parameters

Name:	vSearchSpace	Default:	JOB_NAME1,JOB_NAME2,ETC
Name:	f	Default:	1
Name:	l	Default:	

UC_helper_GetBuildsByNumAsMap @ github

```
/*** BEGIN META {
  "name" : "UC_helper_GetBuildsByNumAsMap",
  "comment" : "Returns a map of builds within numbered constraints. Use default Java API format for key",
  "parameters" : [ 'vSearchSpace','f','l'],
  "core": "1.593",
  "authors" : [
    { name : "Ioannis Moutsatsos" }
  ]
} END META**/

/*
j: Required; A comma delimited list of Job names
f      : Required; lowest range limit
l      : Required; upper range limit or null for all
*/
import jenkins.model.*
def options= new HashMap()
def uc_key='' //uno-choice key: stored in parameter
def uc_value='' //uno-choice value: displayed to user
def buildSet=[]
  def buildSetHr=[:] //a map with human readable value
  options.'f'=f // first build number to include
  options.'l'=l //last build number to include

jobNames=vSearchSpace.split(',')
jobNames.each{ job->
  job=job.trim()
jenkins.model.Jenkins.instance.getItem(job).getBuilds().each{
  println it.number.toInteger()
  println it.result
  println "\n"
  if (options.l!=''){
     if (it.number.toInteger()>=options.f.toInteger() && it.number.toInteger()<options.l.toInteger() && it
    buildSet.add(it)
  } }else{
      if (it.number.toInteger()>=options.f.toInteger()&& it.result.toString()=='SUCCESS'){
    buildSet.add(it)
  }
      }
}
} //end each JobNames


    buildSet.each{
      uc_key="${it.project.name}#${it.number}" //it.toString()
      uc_value="#${it.number} ${it.getDisplayName()}"

    buildSetHr.put(uc_key,uc_value)
    }
return buildSetHr
```

Get build artifacts: AC_React_ArtifactCollector

Display a list of build artifacts from a specified build of a Jenkins project.

- The scriptlet returns a map in the form of `artifact_URL=artifact_Name`.
- You can filter the artifacts by extension
- The user makes selections on human-readable artifact names
- The Active Choice selected artifacts are returned as URLs to the artifacts
- Note the format of the vBuildRef parameter JOBNAME#BUILD_NUMBER

AC_React_ArtifactCollector @github

Parameters

Name: vBuildRef Default: JobName#30

Name: vXtension Default: csv

Script

```
/*** BEGIN META {
  "name" : "AC_React_ArtifactCollector",
  "comment" : "Creates an extension filtered map of artifacts from a selected job build",
  "parameters" : [ 'vBuildRef','vXtension'],
  "core": "1.596",
  "authors" : [
    { name : "Ioannis K. Moutsatsos" }
  ]
} END META**/
import hudson.model.*
  xtension=vXtension
  j_project=vBuildRef.split('#')[0]
  j_build_no=vBuildRef.split('#')[1]
def choices=[:]
def job = hudson.model.Hudson.instance.getItem(j_project)
def build=job.getBuildByNumber(j_build_no.toInteger())
buildURL="${jenkins.model.Jenkins.instance.getRootUrl()}job/$j_project/$j_build_no"
artifact= build.getArtifacts()
   artifact.each{
    choices.put("${buildURL}/artifact/$it" as String,"$it" as String)
   }
return choices.findAll{key,value->value.endsWith(".$xtension")}
```

The BioUno Project

Using Jenkins and other software engineering practices to life-science bioinformatics

The BioUno Open Source Project

While thinking and working on Jenkins and Groovy in 2012, I came to realize that at least one more person was thinking along the same lines!

- **Bruno Kinoshita** had founded the BioUno FOS project
  - Bruno is a contributor to several Open Source projects
  - He has developed several bioinformatic plugins for Jenkins
  - He has used Jenkins in processing life-science data

In 2013, I approached him with a proposal for a new plugin.

- This was to become the Active Choices plugin
  - Bruno asked me to join the BioUno contributors
  - In 2015 BioUno participated in the Mozilla Open Science Hackathon and developed a new figshare plugin

BioUno: Jenkins bioinformatic plugins

Grid computing	
PBS Plug-in	This plug-ins lets you use Jenkins to control a Torque PBS cluster. More information about it in the PBS Plug-in GitHub Wiki
Phylogenetics	
FigTree Plug-in	Integrates FigTree and Jenkins.
MrBayes Plug-in	Integrates MrBayes and Jenkins.
Genetic Analysis	
CLUMPP Plug-in	Integrates CLUMPP and Jenkins.
Distruct Plug-in	Integrates Distruct and Jenkins.
Structure Plug-in	Integrates Structure and Jenkins.
Structure Harvester Plug-in	Integrates Structure Harvester and Jenkins.
Chemical Structures	
Jmol Plug-in	Renders Jmol molecules in the build page, using Jmol Javascript library
UI	
Active Choices Plugin	A Jenkins UI plugin for selecting one or multiple options for a job parameter. The UI control can be rendered in a variety of ways including dynamic HTML
Image Gallery Plug-in	This plug-in reads a job workspace and collects images to produce an image gallery using colorbox lightbox Javascript library.
Misc	
figshare Plug-in	This plug-in integrates figshare and Jenkins, using figshare API v1 with OAuth 1.0 and the Credentials Plug-in.
R Plug-in	A simple plug-in to invoke R interpreter and execute an R script.

BioUno Plugins support:

- Grid computing
- Phylogenetics
- Genetic analysis
- Chemical structures
- User interface
- Visualization
- Statistical computing and graphics
- Collaboration

Groovy in Jenkins: Pipeline as Code

The new **Pipeline Plugin** introduces simple or complex **build orchestration via scripting**

- A **pipeline is a Groovy script** that tells Jenkins what to do when your Pipeline is run
- The community uptake on this is rapid as this approach addresses several important workflow automation issues
  - Can support complex, real-world, CD Pipelines
  - Is Resilient
  - Is Pausable
  - Is Efficient
  - Is Visualized: Pipeline StageView dashboards

Acknowledgements

Thank you for ideas, implementation, collaboration, and support!

Jenkins

- Kohsuke Kawaguchi
- Jesse Glick

BioUno

- Bruno Kinoshita

Boston Groovy/Grails/Spring Boot Meetup

- Tucker-organizer
- Michael Kerry-our host!

References

Quick list of references from slides

Discussed Sites/Plugins

Jenkins

- https://jenkins.io/index.html

Biouno

- http://biouno.org/
- http://biouno.org/blog/

Github: (Ioannis)

- https://github.com/imoutsatsos

Recording of this presentation

- https://youtu.be/ajj1s0TBBnM

Jenkins Plugins Discussed

- https://wiki.jenkins-ci.org/display/JENKINS/Groovy+plugin
- https://wiki.jenkins-ci.org/display/JENKINS/Groovy+Postbuild+Plugin
- https://wiki.jenkins-ci.org/display/JENKINS/Script+Security+Plugin
- https://wiki.jenkins-ci.org/display/JENKINS/Summary+Display+Plugin
- https://wiki.jenkins-ci.org/display/JENKINS/Active+Choices+Plugin