Configuration as Code

Adoption of the Job DSL Plugin at Netflix

Justin Ryan <jryan@netflix.com>
@quidryan
http://www.slideshare.net/quidryan

Who Are We

- Engineering Tools
  - Justin Ryan / jryan@netflix.com / @quidryan
  - Curt Patrick / cpatrick@netflix.com
- Freedom & Responsibility

NETFLIX

Situation

- No single flow that builds everything
- Every team has their own jobs
- Each branch needs multiple jobs

Problem

- Lots of copying of jobs
- Subtle fields hidden in Advanced button
- No ability to mimic another team
- Change in UI can be slow

NETFLIX

Alternatives

- Template Project Plugin from Cloudbees
- Job Generator Plugin
- Literate Plugin
- etc...

NETFLIX

Infrastructure As Code

"The end goal of infrastructure as code is to perform as many infrastructure tasks as possible programmatically. Key word is automation."

(http://www.somic.org/2012/09/28/concise-introduction-to-infrastructure-as-code/)

NETFLIX

Configuration As Code

- Netflix - We understand the value of scaling and automation
- Jobs should be responsible citizens
- Team can own their scripts

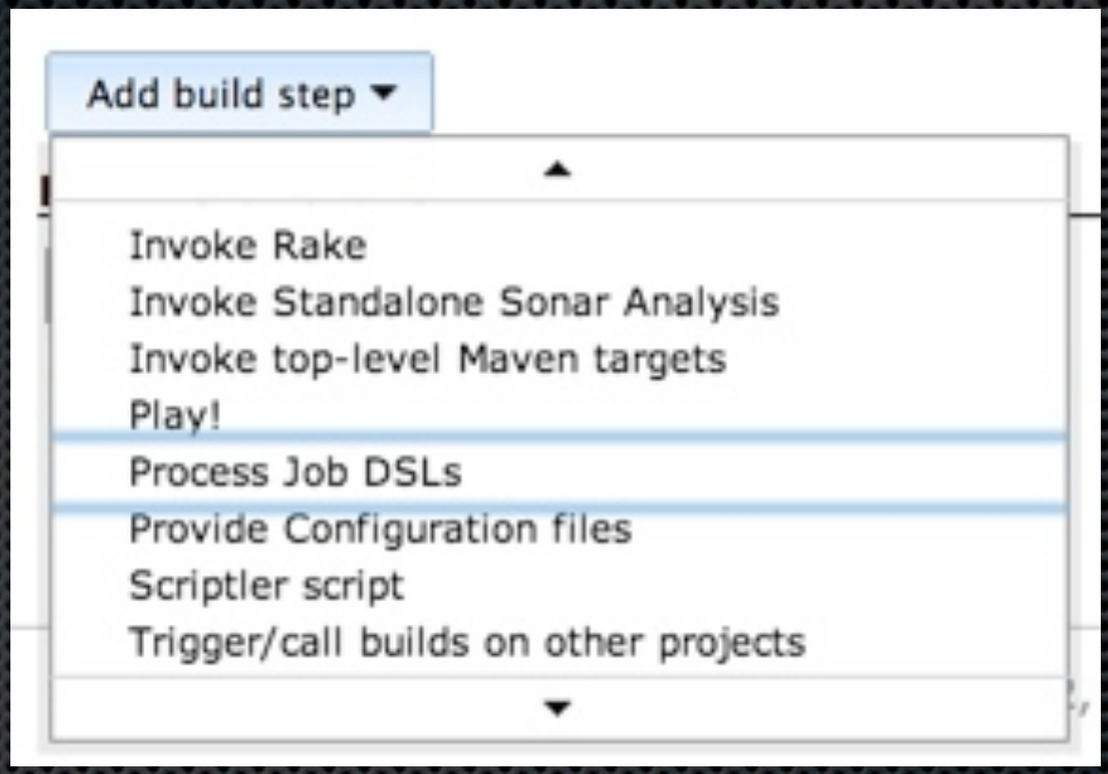
NETFLIX

Job DSL Plugin

- Create a Free-form project
- Add a “Process Job DSL” build step
- Enter Groovy script using a Jenkins DSL
- Run job

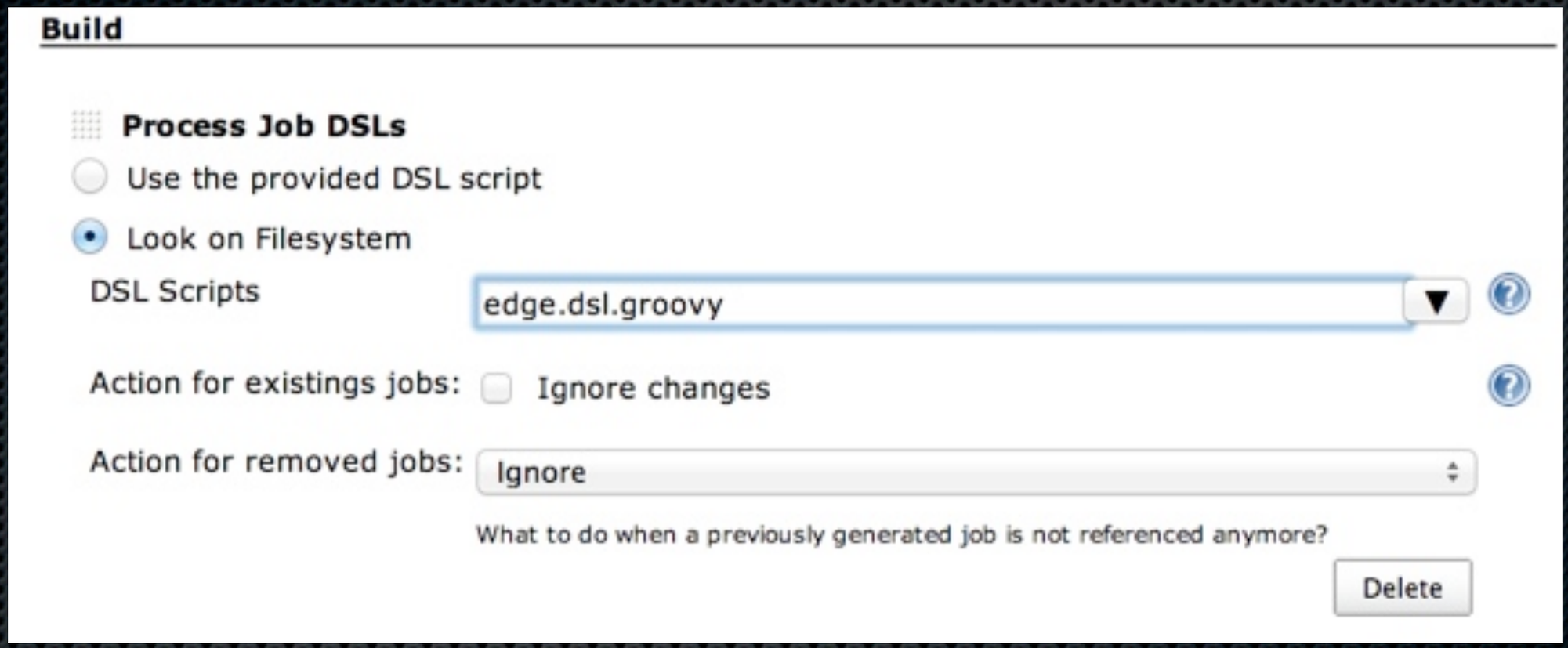
NETFLIX

Step 2. Add Build Step



NETFLIX

Step 3. Specify Script



NETFLIX

Step 4. Run

NETFLIX

Job DSL Language

- Groovy
- One additional method, “job”
- Documentation in Github Wiki
  - https://github.com/jenkinsci/job-dsl-plugin/wiki
  - Job DSL Commands
  - Job Reference Page

Single Job from Github

```
job {
  name "RxJava-Test”
  scm {
    github(“git@github.com:Netflix/RxJava.git”)
  }
  steps {
    gradle("test")
  }
}
```

NETFLIX

Leveraging Groovy

```
import groovy.json.JsonSlurper

def project = 'quidryan/aws-sdk-test'
def github = 'https://api.github.com'
def api = new URL("${github}/repos/${project}/branches")
def branches = new JsonSlurper().parse(api.newReader())
branches.each {
    def branchName = it.name
    job {
        name "${project}-${branchName}".replaceAll('/','-')
        scm {
            git("git://github.com/${project}.git", branchName)
        }
        steps {
            maven("test -Dproject.name=${project}/${branchName}")
        }
    }
}
```

NETFLIX

Disadvantages

- Learning Curve
- Can get complicated
- Not all plugins exist
- Possible problems with plugin version skew

The Project

- 408 installs at last check
- Active Google Group, with 78 members
- Regular Google Hangout
- Constant stream of contributors
- Core contributors - myself, Daniel Spilker, Stefan Wolf, Andrew Harmel-Law

NETFLIX

Use Case - Opower

"Grab json blobs, which define pipeline, from an internal Rails app then loop through with Job DSL. I now have only one job to set up manually. So, in theory, I set up one job and I end up with 600 jobs. Throw each set into a folder and you've got yourself a red/green dashboard."

James Levinson <james.levinson@opower.com>

NETFLIX

```
while(it.hasNext()) {
    Object a = it.next();
    def job_name = "${task}_"+ a.name.toUpperCase()
    println "> Processing ${job_name}..."
    def addresses = config.getAddressesForArtifact(a.name)
    println "addresses: ${addresses}"
    all_string = all_string.concat("${job_name}, ")
    scm_url = "svn+ssh://svn.opower.com/opt/svnroot/main/clients/${a.name}/trunk"

    /* START DSL */
    job(type: Maven) {
        name "${job_name}"
        description Config.managed_by + Config.add_your_email
        logRotator (-1,10,-1,-1)
        label 'javaapps'
        scm { svn "${scm_url}" }
        goals '-U clean deploy'
        triggers { scm '*/15 * * * *' }
        publishers {
            mailer addresses, true, false
            buildDescription "\\[INFO\\] Uploading repository metadata for: \\'snapshot com.[^\\
        }
        configure { project ->
            project / mavenName << 'maven-2.2.1'
        }
    }
    /* END DSL */
    counter++
}
println "Processed " + counter + " ${bucket}.";
```

Use Case - Chicago Trading

"I use the Job DSL to create gerrit build and release build jobs, parameterized on the git repository. Then I use the Build Flow plugin to orchestrate those jobs"

Alan Beale <alan.beale@chicagotrading.com>

NETFLIX

Netflix Conventions

- Utilities methods to create common jobs
- Groovy Category to enhance “job” object
- Published as a jar

NETFLIX

Merge Jobs

```
import netflix.Merge

Merge.nfMergeJobs( [jobFactory: this, securityGroup: ‘nbs’,
    base: '//depot/IT_Engineering/NBS/nbsrules',
    projectPrefix: 'BILLING-rules'] )

Merge.nfMergeJobs( [jobFactory: this, securityGroup: ‘nbs’,
    base: '//depot/IT_Engineering/NBS/Phoenix',
    projectPrefix: 'BILLING-phoenix'] )
```

NETFLIX

Merge Jobs

- BILLING-rules-dev-force-to-test
- BILLING-rules-test-force-to-prod
- BILLING-rules-prod-merge-to-test
- BILLING-rules-test-force-to-dev
- BILLING-phoenix-dev-force-to-test
- BILLING-phoenix-test-force-to-prod
- BILLING-phoenix-prod-merge-to-test
- BILLING-phoenix-test-force-to-dev

NETFLIX

Jobs as Objects

```
import netflix.*
def defaults = [...]
def jobs = [‘dev’,’test’,’prod’].collectMany {[
    CBFQuickBuild(defaults + [branch:branch]),
    CBFQualityBuild(defaults + [branch:branch]),
    nfBakeJob(defaults + [branch:branch])
]}

jobs.each {
    it.publishers {
        extendedEmail('jryan@netflix.com', 'Oops') {
            trigger(triggerName: 'StillUnstable',
                    subject: 'Subject',
                    recipientList:'RecipientList',
                    sendToDevelopers: true,
                    sendToRequester: true,
                    includeCulprits: true,
                    sendToRecipientList: false)
    }
}
```

NETFLIX

Extra features

- Environment variables
- Queue jobs to run
- Read files workspace
- Configure blocks
- @Grab
- @Category

@Category

```
use(netflix.Conventions) {
  def myjob = job {
    name ‘BuildFromP4’
  }
  myjob.addPerforce("//depot/webapps/astrid/...")
}
```

NETFLIX

Good Practice

- Respect “source of truth”
- All scripts go in version control
- Use configure block if needed
- Trigger “seed” job on SCM changes

NETFLIX

Future of project

- Plugins provide DSL methods
- Produce DSL from real job
- Support for utility classes
- Produce jobs from build, e.g. Gradle or Maven

Thank You To Our Sponsors

Platinum

CloudBees

Gold

ZEROTURNAROUND

appvance

black diamond SOFTWARE

New Relic.

JenkinsMobi Mobile ALM

XebiaLabs the deployment automation company

BrightRoll

JFrog

Silver

AppDynamics

Confreaks

LIFERAY

SOASTA Test Faster. Release Sooner.

NETFLIX

Thank You

We're hiring.

Justin Ryan <jryan@netflix.com>
@quidryan
http://www.slideshare.net/quidryan

NETFLIX