Intro to Jenkins Job Builder

Eric Ball

Thanh Ha (zxiiro)

Outline

- Basic JJB CLI usage
- Project & Job Template Definition
- Variables
- Macros
- Common issues & Best Practices

Intent is to go over the very basics of JJB and a few common uses

Main components of a Jenkins Job

- Properties
- Parameters
  - Appears as environment variables in shell
- Wrappers
- Triggers
- SCM
- Builders
- Publishers

What is Jenkins Job Builder

- Basically a YAML to XML conversion tool
- YAML is a human friendly configuration format
- Supports powerful features like templating

- Jenkins jobs are stored as XML files on Jenkins
- Uses Jenkins Rest API to push XML files to Jenkins

JJB Command line utility

- Used to upload jobs to a Jenkins Server
- Configuration file via jenkins.ini and the -c flag
- Test command generates XML
- Update command pushes patches to Jenkins

What is Global JJB?

- Provides macros and job templates
- Many jobs can be implemented directly, simply filling in parameters.
- Custom or advanced jobs can still use parts of it (e.g. macros, builders, etc.)

Global JJB is included in the ci-management repo as a submodule. LF versions the global-jjb repo, with the intention that projects utilizing it can check out particular tags as versions.

https://github.com/lfit/releng-global-jjb

https://docs.releng.linuxfoundation.org/projects/global-jjb/en/latest/index.html

lftools

- lftools is a suite of tools developed internally by the LF Release Engineering team
- These tools handle CI tasks such as:
  - Deploying files to Nexus servers
  - Sign and release staging repositories in Nexus
  - Clean up Docker repos

Lftools is installed via the global-jjb builder “lf-infra-pre-build”. The version to be used is hardcoded in that macro, so that changes to lftools do not affect Jenkins jobs unless a global-jjb upgrade is completed.

Development and Release of Global JJB and lftools

- There are no staff dedicated specifically to global-jjb or lftools.
- Once a week (generally Thursdays), we will release any changes that have been made.
  - Since there are not new changes every week, there are frequently weeks with no new release.
  - When a new version of either suite is created, an announcement is sent to lf-releng@lists.linuxfoundation.org (sign up at https://lists.linuxfoundation.org/mailman/listinfo/lf-releng)
- Release notes are published for both suites
  - https://docs.releng.linuxfoundation.org/projects/global-jjb/en/latest/release-notes.html
  - https://docs.releng.linuxfoundation.org/projects/lftools/en/latest/release-notes.html

JJB Test Command

- Useful to do a quick test to ensure correct syntax
- Can detect issues like undeclared variables
- -c <file> flag to pass configuration file (-c jenkins.ini)
- -l DEBUG set debug mode for logs
- -o <dir> flag to redirect xml output to files
- Parameter order matters!!!

jenkins-jobs --conf jenkins.ini test jjb.yaml

jenkins-jobs -l DEBUG test jjb.yaml

jenkins-jobs test -o /tmp/jjb-test jjb.yaml

JJB Update Command

- Typically used to push individual jobs to a Sandbox system
- --conf <file> flag to pass configuration file (-c jenkins.ini)
- Optional final parameter [job-name]
- Supports wildcards if shell escaped eg. **'*-merge-*'**
- Parameter order matters!!!

jenkins-jobs --conf jenkins.ini update jjb.yaml

jenkins-jobs update jjb.yaml [job-name]

JJB Configuration File

```
[job_builder]
ignore_cache=True
keep_descriptions=False
include_path=.
recursive=True

[jenkins]
user=#USERNAME#
password=#API_TOKEN#
url=https://jenkins.opendaylight.org/sandbox
query_plugins_info=False
```

Basic “Job” Definition

- Defines a Jenkins job
- Very static
- Does not support variables

```
- job:
    name: job-name
```

Do not bother with “job” definition.

Basic “Project” and “Job-Template” Definition

- Basic constructs of a JJB job
- Supports variables
- Re-usable
- **Job-template** defines job
- **Project** realizes job

Basic “Project” and “Job-Template” Definition Example

```
- job-template:
    name: '{project-name}-jjb-verify-{stream}'

- project:
    name: ciman-jjb-verify
    jobs:
        - '{project-name}-jjb-verify-{stream}'
    project-name: ciman
    stream: master
```

JJB “Job Template” variables in job name

- Project creates a job for every variable value combo in a job template name
- Can nest variables to pass different values to different jobs
- JJB will not parse variables if you go 2+ levels deep

JJB Project with multiple name definitions (creates 4 jobs)

```
- project:
    name: ciman-maven

    project-name: ciman
    jobs:
        - '{project-name}-jjb-verify-{stream}'

    stream:

      - nitrogen

      - carbon
    project-name:

      - aaa

      - odlparent
```

JJB Project nested variables (creates 2 jobs)

```
- project:
    name: ciman-maven

    project-name: ciman
    jobs:
        - '{project-name}-jjb-verify-{stream}'
    project-name:

      - aaa:

          stream: nitrogen

      - odlparent

          stream: carbon
```

JJB “Defaults” definition

- Defaults are variables that are used if missing in project definition
- Usually used for system / infra level configuration

```
- defaults:
    name: global
    stream: master

- project:
    name: ciman-jjb-verify

    project-name: ciman
    jobs:
        - '{project-name}-jjb-verify-{stream}'

- job-template:
    name: '{project-name}-jjb-verify-{stream}'
```

JJB “Macro” definition

- Basically a template for a specific configuration section
- Useful for repeating common patterns
- Useful for simplifying a complex definition

JJB “Macro” definition example

```
- builder:
    name: lf-infra-create-netrc
    # Macro to create a ~/.netrc file from a Maven settings.xml
    # Parameters:
    #     {server-id} The id of a server as defined in settings.xml
    builders:
      - inject:
          properties-content: 'SERVER_ID={server-id}'
      - shell: !include-raw-escape: ../shell/create-netrc.sh


 - job-template:

    name: '{project-name}-jjb-verify-{stream}'

    builders:

      - lf-infra-create-netrc:

          server-id: logs
```

JJB “Shell” and !include-raw-escape

- Used to include a shell script defined elsewhere in the jjb repo
- !include-raw exists but recommend to not use it
- !include-raw-escape accepts a list of scripts

```
builders:
  - shell: !include-raw-escape:
      - script1.sh
      - script2.sh
```

```
builders:
  - shell: !include-raw-escape: script1.sh
  - shell: !include-raw-escape: script2.sh
```

Common Issues

- Error: {variable} missing to format '{variable}'
  - A **job-template** or **macro** is expecting to be passed variable but it is not defined
- Over nesting of variables (JJB only parses 1 level deep)
- Mixing {jjb-var} and shell ${SHELL_VAR} style variables
  - !include-raw: will parse **{curly-braces}** as JJB Variables
  - ${SHELL_VAR} will need the curly braces double escaped **${{SHELL_VAR}}**
  - Recommend to only use **${SHELL_VAR}** to keep things simple
- !include-raw-escape has a bug where if no {curl-brace-variables} are declared then it will double up curly braces braking shell scripts. In this case use !include-raw

Best Practices

- Make generally reusable templates and / or macros
- Document templates (see global-jjb for examples)
- Stream & Branch variables are useful for when code-names are involved
- DO NOT use {jjb-variables} in shell scripts. Prefer the “inject” and !include-raw-escape pattern instead (or build parameters).
- JJB has excellent documentation
  - https://docs.openstack.org/infra/jenkins-job-builder/

https://lf-releng-tools.readthedocs.io/en/latest/best-practices.html#jenkins-job-builder

Resources

Eric Ball

eball@linuxfoundation.org

Jenkins Job Builder Docs

https://docs.openstack.org/infra/jenkins-job-builder/

LF Best Practices

https://lf-releng-tools.readthedocs.io/en/latest/best-practices.html#jenkins-job-builder

Global JJB

https://github.com/lfit/releng-global-jjb

IRC Channel

#openstack-jjb @ irc.freenode.net