



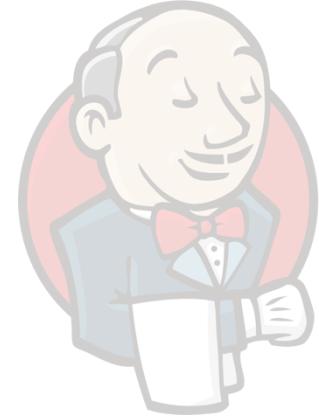
Workflow in Jenkins

Jesse Glick
CloudBees

June 18, 2014

#jenkinsconf

What people are trying to do



- continuous deployment in stages
- run part of build with a temporary server
- blue/green deployment with auto commit/abort
- parallel tests with automatic sharding
- retrying validated merges
- “matrix” builds with per-combination history
- automatic per-branch jobs (à la Literate plugin)
- submit tasks to batch job system
- crazy stuff mentioned in Scalability Summit

Orchestration: what we need



- **complex pipelines** involving multiple stages
- **non-sequential logic** such as loops and forks
- **long-running builds** must survive outages
- **interaction with humans** including pauses, input
- **restartable builds** in case of a transient error
- **reusable definitions** to avoid duplication
- **comprehensible scripts** with one clear definition

Job chaining: what we had (1/2)

Copy artifacts from another project

Project name: Basic Build and Package

Which build: Specific build

Build number: \$ARTIFACT_BUILD_NUMBER

Artifacts to copy: my-build-artifact.txt

Target directory:

Parameter filters:

Flatten directories Optional Fingerprint Artifacts

Trigger parameterized build on other projects

Build Triggers

- Projects to build: MyProject/Dev/Static Code Quality Analysis
- Trigger when build is: Stable
- Trigger build without parameters:

Predefined parameters

Parameters: ARTIFACT_BUILD_NUMBER=\$BUILD_NUMBER

Join Trigger

Trigger even if some downstream projects are unstable

Projects to build once, after all downstream projects have finished:

Run post-build actions at join

Post-Join Actions

Trigger parameterized build on other projects

Build Triggers

- Projects to build: MyProject/QA/Func Tests
- Trigger when build is: Stable
- Trigger build without parameters:

Current build parameters

Promotion process

Name: Approved by Business

Icon: Gold star

Restrict where this promotion process can be run

Criteria

Only when manually approved

Approvers: Bobbi

Approval Parameters:

Promote immediately once the build is complete

Promote immediately once the build is complete based on build parameters

When the following downstream projects build successfully

When the following upstream promotions are promoted

Promotion names: Validated by Pipeline

Actions

Keep Build Forever

Trigger parameterized build on other projects

Build Triggers

- Projects to build: MyProject/Biz/Deploy to Prod
- Trigger when build is: Stable
- Trigger build without parameters:

Predefined parameters

Parameters: ARTIFACT_BUILD_NUMBER=\$PROMOTED_NUMBER

Promotion process

Name: Validated by Pipeline

Icon: Gold empty star

Restrict where this promotion process can be run

Criteria

Only when manually approved

Promote immediately once the build is complete

Promote immediately once the build is complete based on build parameters

When the following downstream projects build successfully

Names: Static Code Quality Analysis, MyProject/QA/Perf Tests, MyProject/QA/Regr Tests

Trigger even if the build is unstable



Build Flow plugin: what we had (2/2)

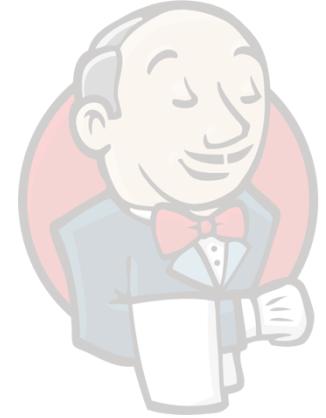


```
b = build("upstream")
build("downstream", /*parameter*/ which: b.build.number)
```

- did have scriptability and extensibility
- did not address configuration sprawl
 - “meat” of builds still had to be in regular jobs
- disjointed view of what really ran
- no ability to survive restarts
- almost good enough but could not go further

Workflow: the one-pager

```
with.node('linux') {  
    git(url: 'git://server/myapp.git')  
    sh('mvn clean package')  
    archive('target/myapp.war')  
    stage('Test')  
    parallel({  
        sh('mvn -Psometests test')  
    }, {  
        sh('mvn -Pothertests test')  
    })  
    input('OK to deploy?')  
    stage(value: 'Deploy', concurrency: 1)  
    sh('mvn deploy')  
}
```



Key features

- entire flow is one concise Groovy script
 - for-loops, try-finally, fork-join, &c.
- can restart Jenkins while flow is running
- allocate slave nodes and workspaces
 - as many as you like, when you like
- *stages* throttle concurrency of builds
- human input/approval integrated into flow
- standard project concepts: SCM, artifacts, ...



Project setup

- one workflow is defined as a job
- single script for all steps
- build triggers & parameters like regular projects
- SCM, publishing, &c. all part of script
- Each build shown using regular Jenkins view
- Graphical visualizations of actual build possible
 - (not of job definition; could be too dynamic)



Resumption of Groovy flows

- transformed to continuation-passing style
- custom interpreter of Groovy
- state of program saved at each point
- variables serialized and restored after restart
- *pickles*: extensible object replacements
 - slaves reallocated, workspaces relocked



Resumed builds to the user

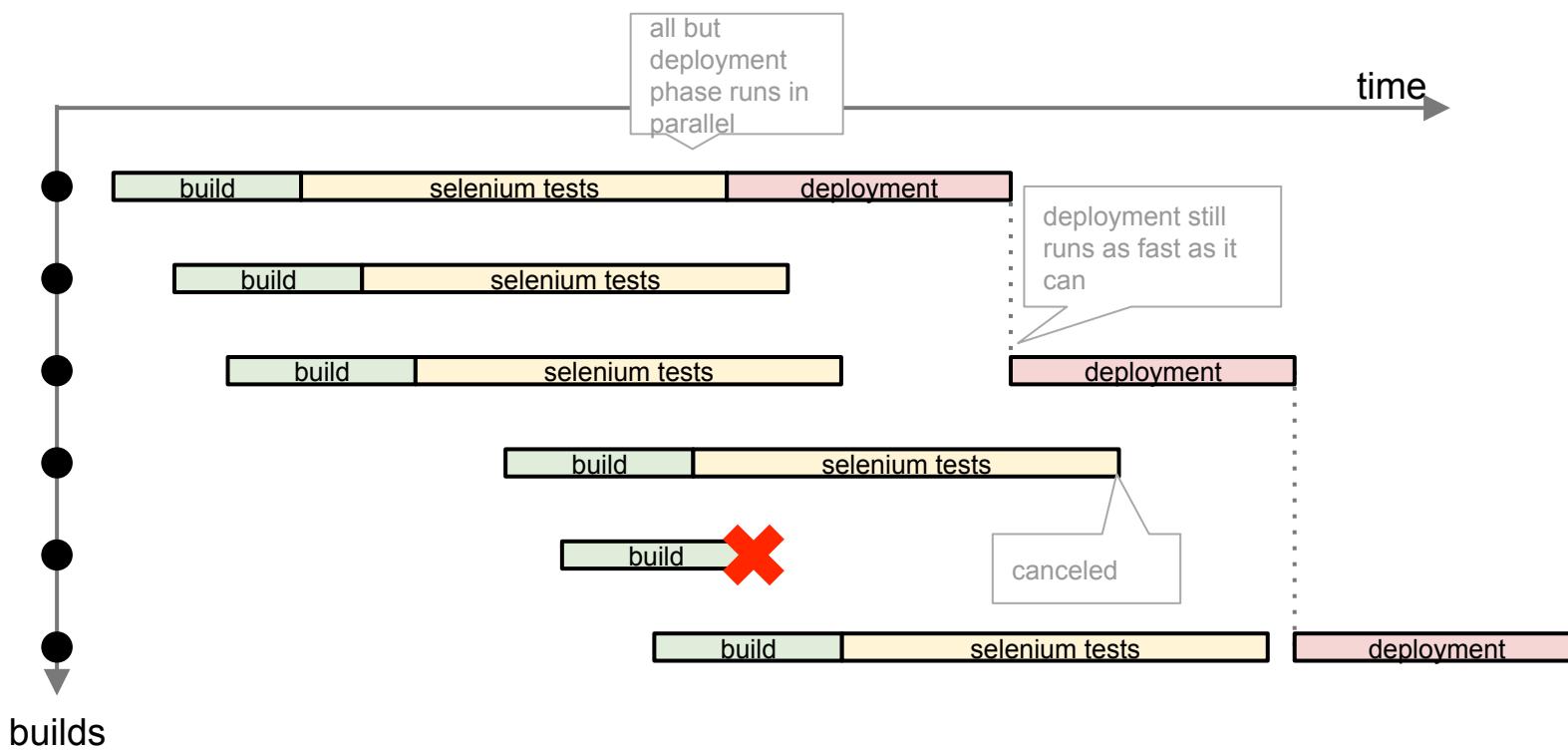
- it “just works”
- loops, methods, closures, &c.
- (Serializable) local variables restored too
- shell-like steps survive restart
 - reconnection of slave, too
- Jenkins Enterprise: resume from checkpoint
 - can pick up artifacts from original build
 - no need to rerun earlier expensive steps



Stages

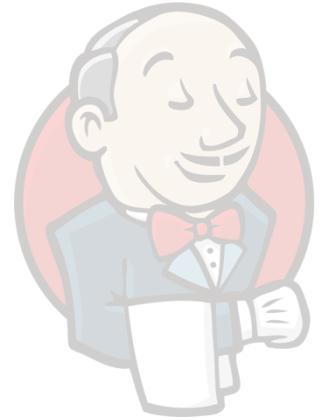


- special semaphore: only newest build may wait
- kudos to James Nord for the idea (in Build Flow)



Demo

simple CD pipeline



Design: overall

- suite of Jenkins plugins
 - Jenkins Enterprise may add checkpoints, &c.
- pluggable flow definition & execution engine
 - Groovy CPS is recommended choice
 - STM (proof of concept)
 - Activiti or other BPMN should be possible

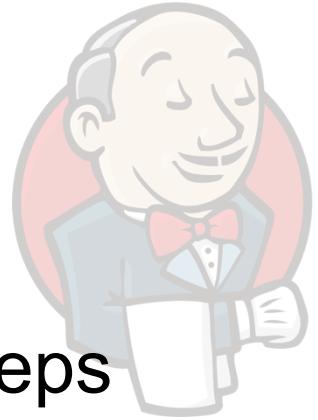


Design: flows

- persistent record of execution
- directed acyclic graph of *nodes*
- some nodes represent one *step*
- others indicate block start/stop structure
- nodes may have associated metadata
 - console log fragment contributes to main log
- pluggable visualizations for different views



Design: steps



- standalone API for asynchronous build steps
- *context* serves as an identifier & callback
 - also offers logger, build, workspace, &c.
- support for block-structured steps
 - invoke body 0+ times with revised context
- standard step for “durable” shell/batch scripts
- standard steps for SCMs (git, svn, hg)
 - >1 SCM per build possible

Design: interoperability



- run on existing Jenkins slaves
 - no elastic cloud support (yet?)
- SCM plugins supported with modest changes
 - changelog, polling, commit trigger
- coming soon: existing build steps & publishers
- coming soon: trigger existing jobs
- standard build history, artifacts
- needs ongoing core changes (currently 1.568+)
 - features factored out of standard projects

Still to come

- more build steps
- workspace management
- Cancel button
- robustness, polished UI
- Groovy sandbox
- load libraries, or script from SCM
- open for contributions!



Status

- github.com/jenkinsci/workflow-plugin
- 0.1-beta-1 binaries on experimental UC
- requires Jenkins 1.568+ today
- fundamentals all work now
- aiming for 1.0 this year
- considered strategic by CloudBees



Thank You To Our Sponsors

Platinum



Gold



MidVisionTM
Release the innovation



Silver

SOASTA
Test Faster. Release Sooner.

ELIASSEN GROUP
Agile Consulting Services

KENSHO

 **SERENA**TM

 **assembla**

