7 More Habits of Highly Scalable Jenkins Administrators

Kristin Whetstone

Introduction

- Software Engineer at CloudBees
- CloudBees Jenkins Advisor
  - Automatically scan and offer insights to your Jenkins instance
  - Free service, open source plugin
- Email - kwhetstone@cloudbees.com
- GitHub - kwhetstone
- Twitter - @lighteningdrake

Seven Habits of Highly Effective Jenkins Users

Check it out on YouTube!

Seven Habits of Highly Scalable Jenkins Administrators

Jenkins 2

Jenkins 2

- Released April of 2016
- Includes:
  - New secure defaults
  - Pipeline as Code front and center
  - Improved on-boarding experience
- 2.x based releases are all the Jenkins project supports

Use the latest Jenkins LTS

Habit #1

Use the latest Jenkins LTS

- Long-Term Support for Jenkins is ~3 months
- Newer plugins require newer core features:
    - Blue Ocean requires 2.7.1 or later
- Security updates only applied to latest weekly and current LTS
    - jenkinsci-advisories@googlegroups.com

Dockerized master with LTS

- Use a Dockerized master to stay up to date:
  - `jenkinsci/jenkins:lts`
  - `jenkinsci/jenkins:lts-alpine`
- Guaranteed to be up to date and supported.
- Uses OpenJDK 8 for the runtime
  - Java 7 support is going away soon!
- Production ready
  - ci.jenkins.io is a Dockerized master

Dockerized master with LTS

- Can be a drop-in upgrade on Linux hosts.

```
docker run -p 8080:8080 \
    -u $(id -u jenkins):$(id -g jenkins) \
    -v /var/lib/jenkins:/var/jenkins_home \
    jenkinsci/jenkins:lts-alpine
```

Resources

- jenkins.io/doc/upgrade-guide
- jenkins.io/changelog-stable
- jenkins.io/download

Use Jenkins Pipeline

Habit #2

Use Jenkins Pipeline

- Pipeline as Code enables teams for self-service
- Modern tooling in Jenkins increasingly built atop Pipeline
    - Blue Ocean
    - Org Folders
- Automatically register projects
    - GitHub Organization Folder
    - Bitbucket Team/Project Folder

Use Jenkins Pipeline

- Durable
  - No more waiting for jobs to finish to restart the master
  - Agents continue working while the master restarts

Scripted Pipeline

- Groovy
- A power-tool for experienced users/administrators
- Used for creating Shared Libraries

Scripted Pipeline

```
node('docker') {
    stage('Build') {
        docker.image('maven:3-alpine').inside {
            sh 'mvn'
        }
    }
}
```

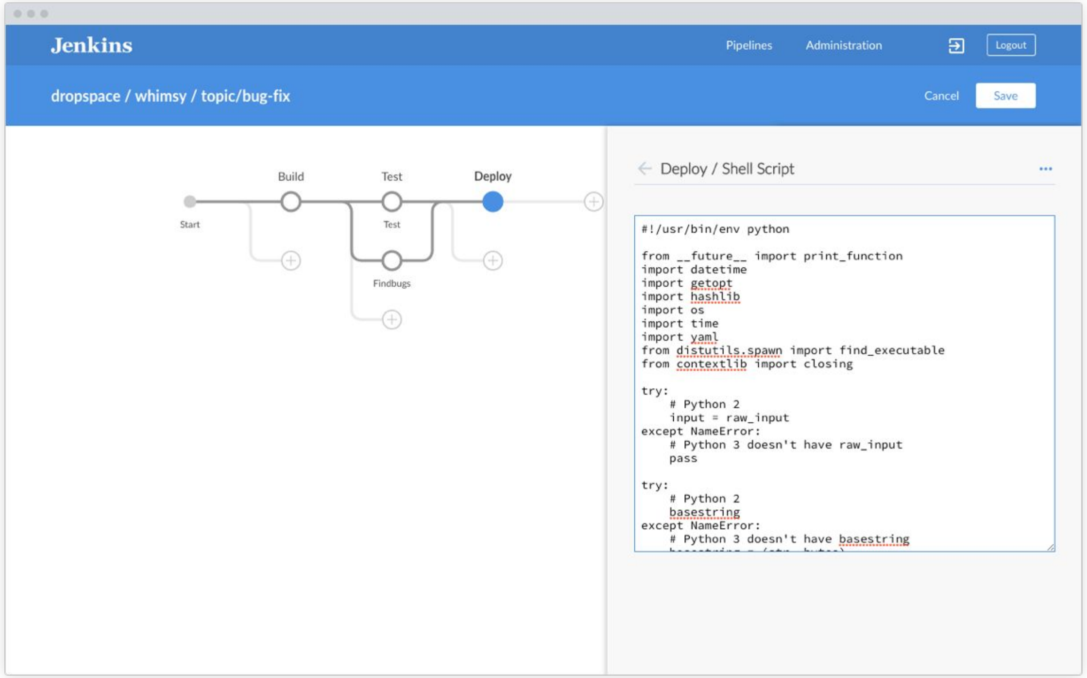
Declarative Pipeline

- Easy to verify, write, and understand
- Intentionally restricted
  - Can be validated before execution using the `declarative-linter` CLI command
- Supported by the Blue Ocean Pipeline Editor

Declarative Pipeline

```
pipeline {
    agent { docker { image 'maven:3-alpine' } }
    stages {
        stage('Build') {
            steps {
                sh 'mvn'
            }
        }
    }
}
```

Blue Ocean Pipeline Editor



```
#!/usr/bin/env python

from __future__ import print_function
import datetime
import getopt
import hashlib
import os
import time
import yaml
from distutils.spawn import find_executable
from contextlib import closing

try:
    # Python 2
    input = raw_input
except NameError:
    # Python 3 doesn't have raw_input
    pass

try:
    # Python 2
    basestring
except NameError:
    # Python 3 doesn't have basestring
```

@lighteningdrake #ATO2017

Key Differences / Tips

- Default to Declarative Pipeline
  - unless you have a **strong** reason not to
- Declarative Pipeline can be validated without being run
  - CLI command: `declarative-linter`
- Declarative Pipelines with too much logic are a "smell"
  - Consider refactoring parts into a Shared Library
- Shared Libraries use Scripted Pipeline
  - Knowledge in the organization of Scripted is useful

Plugins to consider

- Pipeline plugin
- Blue Ocean plugin
- Blue Ocean Pipeline Editor plugin
- GitHub Branch Source / Bitbucket Branch Source

jenkins.io/doc/book/pipeline

Scale with Shared Libraries

Habit #3

Scale with Shared Libraries

- Build organization-specific DSL additions
- Prevents Jenkinsfiles from being clever
- Reduces copy/paste, increases consistency
    - Having common steps "owned" by the Shared Services team encourages creation of reliable continuous delivery pipelines

Scale with Shared Libraries

```
/* vars/runMaven.groovy */

def call(Map params = [:]) {

    def targets = params.targets ?: ''

    node('docker') {

        checkout scm

        docker.image('maven:3-alpine').inside {

            sh "mvn ${targets}"

            junit '**/target/**/surefire-reports/*.xml'

        }

    }

}
```

Scale with Shared Libraries

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                runMaven(targets='clean package')
            }
        }
    }
}
```

Scale with Shared Libraries

```
/**
 * Simple wrapper step for building a plugin
 */
def call(Map params = [:]) {
    def platforms = params.containsKey('platforms') ? params.platforms : ['linux', 'windows']
    def jdkVersions = params.containsKey('jdkVersions') ? params.jdkVersions : [8]
    def jenkinsVersions = params.containsKey('jenkinsVersions') ? params.jenkinsVersions : [null]
    def repo = params.containsKey('repo') ? params.repo : null
    def failFast = params.containsKey('failFast') ? params.failFast : true
    Map tasks = [failFast: failFast]
    for (int i = 0; i < platforms.size(); ++i) {
        for (int j = 0; j < jdkVersions.size(); ++j) {
            for (int k = 0; k < jenkinsVersions.size(); ++k) {
                String label = platforms[i]
                String jdk = jdkVersions[j]
                String jenkinsVersion = jenkinsVersions[k]
                String stageIdentifier = "${label}-${jdk}${jenkinsVersion ? '-' + jenkinsVersion : ''}"

                tasks[stageIdentifier] = {
                    node(label) {
                        boolean isMaven

                        stage("Checkout (${stageIdentifier})") {
                            if (env.BRANCH_NAME) {
                                timestamps {
                                    checkout scm
                                }
                            }
                            else if ((env.BRANCH_NAME == null) && (repo)) {
                                timestamps {
                                    git repo
                                }
                            }
                            else {
                                error 'buildPlugin must be used as part of a Multibranch Pipeline *or*
a `repo` argument must be provided'
                            }
                            isMaven = fileExists('pom.xml')
                                                                                    }
```

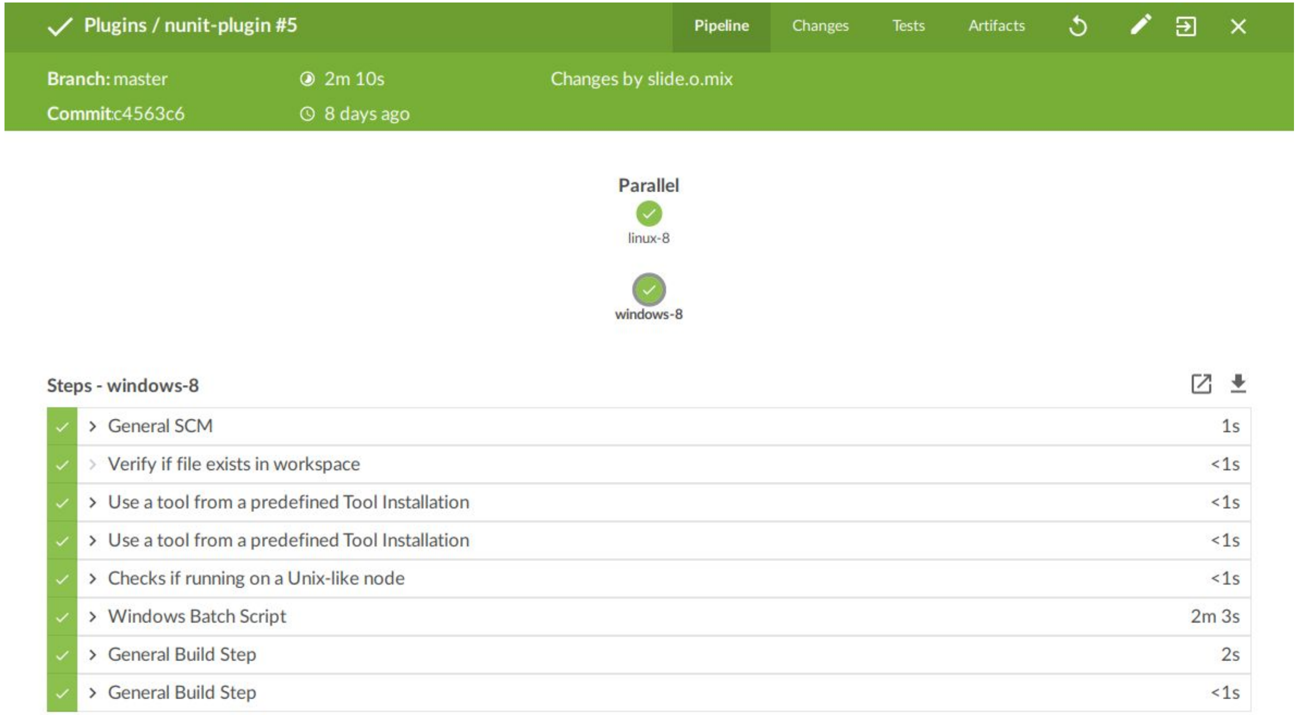
```
String command
if (isMaven) {
    List<String> mavenOptions = [
            '--batch-mode',
            '--errors',
            '--update-snapshots',
            '-Dmaven.test.failure.ignore=true',
            "-DskipAfterFailureCount=${failFast}",
    ]
    if (jenkinsVersion) {
        mavenOptions += "-Djenkins.version=${jenkinsVersion}"
    }
    command = "mvn ${mavenOptions.join(' ')} clean install"
    env << "PATH+MAVEN=${tool 'mvn'}/bin"
} else {
    List<String> gradleOptions = [
            '--no-daemon',
            'cleanTest',
            'build',
    ]
    command = "gradlew ${gradleOptions.join(' ')}"
    if (isUnix()) {
        command = "./" + command
    }
}

withEnv(env) {
    if (isUnix()) {
        timestamps {
            sh command
        }
    }
    else {
        timestamps {
            bat command
        }
    }
}
```

```
                        stage("Archive (${stag
                            String testReports
                            String artifacts
                            if (isMaven) {
                                testReports =
                                artifacts = '*
                            } else {
                                testReports =
                                artifacts = '*
                            }

                            timestamps {
                                junit testRepo
                                if (failFast &
                                    error 'The
                                }
                                archiveArtifac
                                        fi
                            }
                        }
                    }
                }
            }
        }
    }

    /* If we cannot complete in 60 minutes, we
     * isn't free!
     */
    timeout(60) {
        return parallel(tasks)
    }
}
```

Scale with Shared Libraries

```
/* Jenkinsfile */
buildPlugin()


/* A different Jenkinsfile */
buildPlugin(platforms: ['linux'], jdkVersions: [7, 8])
```

Scale with Shared Libraries



✓ Plugins / nunit-plugin #5

Pipeline | Changes | Tests | Artifacts

Branch: master
Commit: c4563c6
2m 10s
8 days ago
Changes by slide.o.mix

Steps - windows-8

✓	General SCM	1s
✓	Verify if file exists in workspace	<1s
✓	Use a tool from a predefined Tool Installation	<1s
✓	Use a tool from a predefined Tool Installation	<1s
✓	Checks if running on a Unix-like node	<1s
✓	Windows Batch Script	2m 3s
✓	General Build Step	2s
✓	General Build Step	<1s

Plugins to consider

- Pipeline plugin

jenkins.io/doc/book/pipeline/shared-libraries

Examples:

- github.com/jenkins-infra/pipeline-library
- github.com/docker/jenkins-pipeline-scripts

Self-service with containers

Habit #4

Self-service with containers

- Enables developers to "choose their own adventure"
- Teams can use their own project-specific system requirements
    - E.g. Native libraries can live in a container, instead of rolled out to the Jenkins environment
- Easy for teams to choose "side-car" containers for datastores (DBs, caches, etc)

Self-service with containers

```
pipeline {
    agent { docker { image 'maven:3-alpine' } }
    stages {
        stage('Build') {
            steps {
                sh 'mvn'
            }
        }
    }
}
```

Plugins to consider

- Pipeline plugin
- Docker Pipeline plugin

Make Agents Elastic

Habit #5

Make Agents Elastic

- Great cost/benefit ratio
- Agents always spin up in a clean state
  - Messy jobs and Pipelines aren't Your Problem™
- More capacity, means faster feedback cycles
  - Pipelines can easily run across many agents
- Agents deleted when not in use, means less wasted money

Parallel Pipelines

```
/* assume the "Build" stage completed */
parallel(unit: {
     stage('Unit Testing') {
          node('linux') { sh 'make check' }
     }
},
scan: {
     stage('Static Analysis') {
          node('linux') { sh 'make scan' }
     }
})
```

Elastic Agents Tips

- Most cloud-provider plugins allow for a "retention period"
  - Set the retention period ~2-4x spin-up time
  - Experiment to find the sweet spot for "peak" load
- Investigate VM image creation tools for reduced spin-up
  - Packer
- Create "general purpose" VMs
  - Avoid creating VM images or templates for each specific project

Plugins to consider

- EC2 Agents plugin
- Azure VM Agents plugin
- EC2 Spot Fleet plugin
- JClouds plugin

Reduce Permissions Increase Auditability

Habit #6

Reduce Permissions

- Pipeline as Code means developers needn't have Job Configure access
- Docker-based self-service Pipelines require no special agent permissions to execute
- Reducing write-access reduces opportunity for shadow IT and misconfiguration

Increase Auditability

- Use a configuration management tool, or other scripting tools for orchestration of the master.
- Groovy scripts can be run on boot or via CLI
    - `JENKINS_HOME/init.groovy.d/*.groovy`
    - Runs after plugins are loaded.
- Practically anything in Jenkins be configured with Groovy
    - It is not terribly user-friendly.

Plugins to consider

- Matrix Authorization Strategy plugin
- Job Config History plugin
  - Can cause issues at scale
- "**Mastering the Jenkins Script Console**" with Sam Gleske from Jenkins World 2017

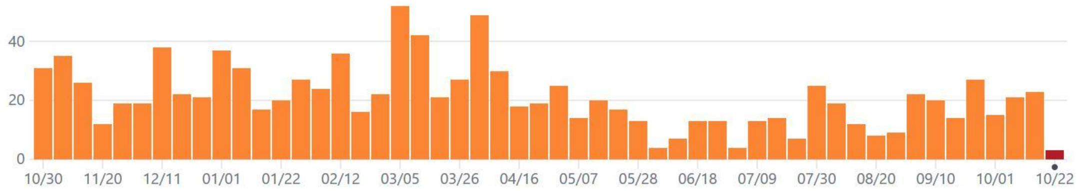
github.com/jenkinsci/jenkins-scripts

Participate in Jenkins

Habit #7

Participate in Jenkins

- **Lots** of best practices and tips shared in JAMs, on the mailing lists, IRC, Stack Overflow, etc
    - Next Triangle JAM: Nov 8
- A number of plugins are up for adoption
- Contributors define the future of the project



Participate in Jenkins

- jenkins.io/participate
- Blog: jenkins.io
- @jenkinsci
- github.com/jenkinsci
- Adopt a Plugin:

  wiki.jenkins-ci.org/display/JENKINS/Adopt+a+Plugin
- jenkinsci-dev@googlegroups.com

Summary

1. Use the latest Jenkins LTS
2. Use Jenkins Pipeline
3. Scale with shared libraries
4. Self-service with containers
5. Make agents elastic
6. Reduce permissions, increase auditability
7. Participate in Jenkins

Bonus: CloudBees Jenkins Advisor!

Questions?

Seven Habits of Highly Scalable Jenkins Administrators