Continuous Delivery with Jenkins Pipelines (incl. Advanced Topics)

Roman Pickl
23.05.2018

join at Slido.com with #K100
Demo

1. docker run -p 8080:8080 jenkinsci/blueocean (add –p 44444:44444 or any other port to try the ssh linter)
2. Go to http://localhost:8080/
3. Unlock jenkins with initialpw from log file
4. Install suggested plugins (you may have to continue the process in case any plugins are broken and update plugins later on)
5. Create admin user / or continue with admin and initialpw
7. Generate new Pipeline with Github repository
8. Create Jenkinsfile (e.g. https://github.com/rompic/jenkinspipeline)
How did I end up here?

Roman Pickl (@rompic)

• Technical Project Manager @ Elektrobit
• Uses Jenkins since 2012
• Loves CI/CD/DevOps
• Here to learn
Continuous Delivery (CD)

Automated implementation of your system’s build, deploy, test, release process

• Every change results in a build
• Every build is a release candidate
• Delivery can be done at any time, on any environment
→ Make releases a non-event

Deployment Pipeline provides:
• Visibility
• Feedback
• Control

Jez Humble on Continuous Delivery (2012): [https://www.youtube.com/watch?v=sklJuksCRTw](https://www.youtube.com/watch?v=sklJuksCRTw)

Read these books if you want to know more!
Deployment Pipelines (Let’s build it with jenkins)

Jez Humble D. F., Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (2010)

© Elektrobit (EB) 2018
Jenkins

#1 Continuous Integration and Delivery Server

- Created by Kohsuke Kawaguchi
- Initial Release 2005 (Hudson)
- Open Source (MIT License)
- Active and independent community (https://jenkins.io)
- 164,000 active installations
- 1,500+ plugins (!)

- Since 2.0 Pipelines (April 2016) are first class citizens
- Pipeline as Code (Jenkinsfile).
- New User Experience “Blue Ocean” with Blue Ocean Pipeline Editor
- Blue Ocean 1.5 released in April 2018
Welcome Blue Ocean!

<table>
<thead>
<tr>
<th>Jenkins</th>
<th>Pipelines</th>
<th>Administration</th>
<th>Pipelines</th>
<th>New Pipeline</th>
</tr>
</thead>
</table>

Welcome to Jenkins
It's time to create your first Pipeline.

Create a new Pipeline
Create a Pipeline Wizard

Where do you store your code?
- Bitbucket Cloud
- Bitbucket Server
- GitHub
- GitHub Enterprise
- Git

Which organization does the repository belong to?
- rompic

Connect to Github
Jenkins needs an access key to authorize itself with Github.
Create an access key here.

Connect to Github

Choose a repository
Loaded '14' repositories
jenkinspipeline

Create Pipeline
Jenkinsfile

Written in a Groovy DSL
“Jenkinsfile” in top level folder (different path possible since June 2017 https://issues.jenkins-ci.org/browse/JENKINS-34561)

Store in SCM (e.g. GIT) for additional benefits
• Code review/iteration
• Audit trail
• Single source of truth

Supports two syntaxes (can be mixed)
• Declarative pipelines (easier; “new”; 1.0 Feb 2017)
• Scripted pipelines (more powerful)

© Elektrobit (EB) 2018
Scripted vs. Declarative

Scripted Pipelines

```groovy
Jenkinsfile (Scripted Pipeline)
node {
    stage('Example') {
        if (env.BRANCH_NAME == 'master') {
            echo 'I only execute on the master branch'
        } else {
            echo 'I execute elsewhere'
        }
    }
}
```

```groovy
Jenkinsfile (Scripted Pipeline)
node {
    stage('Example') {
        try {
            sh 'exit 1'
        }
        catch (exc) {
            echo 'Something failed, I should sound the klaxons!' throw
        }
    }
}
```

Declarative Pipeline

```groovy
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```
Scripted vs. Declarative

Scripted Pipelines
- imperative programming model
- fully featured programming environment,
- higher flexibility and extensibility
- very few limits
→ for power-users and more complex requirements

Declarative Pipeline
- declarative programming model
- simpler and more opinionated syntax for authoring Jenkins Pipeline.
- Allows for validation and a visual editor
- limits what is available to the user
→ ideal choice for simpler continuous delivery pipelines

Both
- use Groovy
- same Pipeline sub-system underneath
- mostly use same steps
- able to utilize Shared Libraries
→ can be mixed using the script step

See [https://jenkins.io/blog/2017/01/19/converting-conditional-to-pipeline/](https://jenkins.io/blog/2017/01/19/converting-conditional-to-pipeline/) for a more complex example of migrating a freestyle job to a declarative/scripted pipeline.
script Step

- takes a block of Scripted Pipeline & executes that in the Declarative Pipeline
- can provide a useful "escape hatch".
- script blocks of non-trivial size and/or complexity should be moved into Shared Libraries

```groovy
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'

                script {
                    def browsers = ['chrome', 'firefox']
                    for (int i = 0; i < browsers.size(); ++i) {
                        echo "Testing the \$\{browsers[i]\} browser"
                    }
                }
            }
        }
    }
}
```
Settings

pipeline {
  agent none      // don't block an executor for approval
  // see http://bit.ly/2qrz2Ty
  // environment, options, tools, parameters
  // and triggers can also be defined here for the whole pipeline
  triggers { pollSCM('H/5 * * * *') } // poll every 5 mins
  options {
    timeout(time: 60, unit: 'DAYS')
    buildDiscarder(logRotator(numToKeepStr: '30'))
  }
}
Stages

```groovy
stages {
    stage('Build & unit tests') {
        agent any
        tools {
            //this is ignored at top level if agent none is specified.
            // jdk 'Oracle Java 8' (defined in jenkins setup)
        }
        steps {
            echo 'running build and unit tests'
            deleteDir() //delete everything in this workspace
            checkout scm
            // sh './gradlew build'
            // archiveArtifacts
            // stash
        }
    }
    // post {
    //    always{
    //        // publish unit tests
    //        // junit 'path/to/tests/*.xml'
    //    }
    // }
}
```
Parallel Execution

```groovy
stage('Automated Acceptance tests'){
    parallel{ //since declarative pipelines 1.2
        stage('Automated Acceptance tests Firefox'){
            agent any
            steps{
                // unstash
                echo "testing Firefox"
            }
            //publish unit tests (omitted here)
        }
        stage('Automated Acceptance tests chrome'){
            agent any
            steps{
                // unstash
                echo "testing chrome"
            }
            //publish unit tests (omitted here)
        }
    }
}
```
Approval

```java
stage('Deploy to Stage for User acceptance tests') {
    when { branch 'master' } //only offer this option on master
    steps {
        milestone(1)
        timeout(time: 30, unit: 'DAYS') {
            input message: 'Deploy to Stage?', submitter: 'admin,tom.tester,pete.pm'
        }
        milestone(2)
    }
}
```

Deploy to Live / Release omitted here
Post Build Notifications

```yaml
post {
    // feedback on failure (also always, success, unstable, changed, fixed, regression available)
    failure {
        emailext {
            subject: "FAILED: Job '{{$env.JOB_NAME} [{{$env.BUILD_NUMBER}}]}',
            body: """"<p>FAILED: Job '{{$env.JOB_NAME} [{{$env.BUILD_NUMBER}}]}':</p>
            <p>Check console output at &QUOT;<a href='${env.BUILD_URL}'>${env.JOB_NAME} [{{$env.BUILD_NUMBER}}]</a>&QUOT;</p>"
            recipientProviders: [[$class: 'CulpritsRecipientProvider']]
        }
    }
}
```
Blue Ocean 1.5 released 11.04.2018

https://wiki.jenkins.io/display/JENKINS/Blue+Ocean+Plugin

• Latest additions
  – Show the downstream jobs launched with the build step
  – Reorder steps in pipeline editor by drag and drop
  – Pagination of artifacts page

Public roadmap
• https://jenkins.io/projects/blueocean/roadmap/

https://jenkins.io/blog/2018/04/18/blueocean-1-5-0/
© Elektrobit (EB) 2018
# Overview

## Jenkins

<table>
<thead>
<tr>
<th>Status</th>
<th>Run</th>
<th>Commit</th>
<th>Branch</th>
<th>Message</th>
<th>Duration</th>
<th>Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>eCte6f5</td>
<td>master</td>
<td>updated comment on failure modes</td>
<td>33m 7s</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>438f169</td>
<td>master</td>
<td>Branch indexing</td>
<td>6d 13h 36m 21s</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>438f169</td>
<td>dev</td>
<td>Branch indexing</td>
<td>54s</td>
<td>7 days ago</td>
</tr>
</tbody>
</table>
# Approval

## Jenkins Pipeline 2

<table>
<thead>
<tr>
<th>Branch:</th>
<th>master</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit:</td>
<td>e38d68f5</td>
</tr>
</tbody>
</table>

### Changes
- **Pipeline**: Started by an SCM change

### Diagram
![Pipeline Diagram]

### Notes
- **Deploy LIVE? - 20s**
  - Milestone step forces all builds to go through in order
  - Wait for interactive input

### Questions
- **Deploy to Live?**
  - Proceed
  - Abort
Continuous Delivery with Jenkins Pipelines (incl. Advanced Topics)

Detail

jenkinspipeline 2

Branch: master
Commit: ech68f5

38m 46s
a few seconds ago

Changes by no-reply
Started by an SCM change

Release - 2s

- General SCM
- deploying to live and running smoke tests

Automated Acceptance tests
Deploy to Stage
Deploy to Stage
Deploy LIVE?
Release
End
Multi branch support

<table>
<thead>
<tr>
<th>HEALTH</th>
<th>STATUS</th>
<th>BRANCH</th>
<th>COMMIT</th>
<th>LATEST MESSAGE</th>
<th>COMPLETED</th>
</tr>
</thead>
<tbody>
<tr>
<td>☀️</td>
<td>✔️</td>
<td>master</td>
<td>—</td>
<td>Started by an SCM change</td>
<td>4 minutes ago</td>
</tr>
<tr>
<td>☀️</td>
<td>✔️</td>
<td>dev</td>
<td>—</td>
<td>Branch indexing</td>
<td>7 days ago</td>
</tr>
</tbody>
</table>
Multi branch (When)
Pipeline Development and Advanced Tools

- Blue Ocean Pipeline Editor
- Snippet Generator
- Directive Generator (*NEW*)
- Auto-Convert Freestyle Jobs to Jenkins Pipeline
- Replay Feature
- IntelliJ IDEA GDSL – Autocomplete
- Command-line Pipeline Linter
- Jenkins File Runner (*NEW*)
- Unit Testing Jenkins Pipelines
- Shared Libraries
Pipeline Editor

- GitLab currently not supported (planned! [https://issues.jenkins-ci.org/browse/JENKINS-43976](https://issues.jenkins-ci.org/browse/JENKINS-43976))
- Ctrl-S / Cmd-S to open the load save dialog
Snippet generator

http://localhost:8080/pipeline-syntax
Declarative generator

Auto-Convert Freestyle Jobs to Jenkins Pipeline

Plugin to automatically convert Freestyle Jobs to Jenkins Pipeline

https://wiki.jenkins.io/display/JENKINS/Convert+To+Pipeline+Plugin
Replay Feature

Green sub-title

• Allows for quick modifications and execution of an existing (valid!) Pipeline without changing the Pipeline configuration or creating a new commit.

• Once you are satisfied with the changes, you can use Replay to view them again, copy them back to your Pipeline job or Jenkinsfile, and then commit them using your usual engineering processes.
IntelliJ IDEA GDSL - Autocomplete

Green sub-title

- Auto completion of steps for **scripted** pipelines
- Install Groovy Plugin
- Add it as e.g. pipeline.gdsl to your projects src path

See
- [https://st-g.de/2016/08/jenkins-pipeline-autocompletion-in-intellij](https://st-g.de/2016/08/jenkins-pipeline-autocompletion-in-intellij)
- [https://stackoverflow.com/a/41149255/3165782](https://stackoverflow.com/a/41149255/3165782)

for setting it up.
Command-line Pipeline Linter

Validate **Declarative Pipelines** from the cli before actually running it/checking it in.

**Linting via the CLI with SSH**

```
# ssh (Jenkins CLI)
# JENKINS_SSHD_PORT=[sshd port on master]
# JENKINS_HOSTNAME=[Jenkins master hostname]
ssh -p $JENKINS_SSHD_PORT $JENKINS_HOSTNAME declarative-linter < Jenkinsfile
```

Errors encountered validating Jenkinsfile:
```
WorkflowScript: 30: Unknown stage section "step". Starting with version 0.5, steps in a stage must be in a steps block. @ line 30,
  stage ('Test') {
  ^
```
```
WorkflowScript: 30: Nothing to execute within stage "Test" @ line 34, column 5.
  stage ('Test') {
  ^
```

See [https://jenkins.io/doc/book/pipeline/development/#linter](https://jenkins.io/doc/book/pipeline/development/#linter) for details. Remember to enable SSH access, expose a port on your docker container and add ssh key to try this!
jenkinsfile-runner

So i guess we can run now run a job in jenkins which downloads jenkins to run a jenkins job ...

Experiment to package Jenkins pipeline execution as a command line tool.

Use cases include:

• Assist editing and testing Jenkinsfile locally
• Use Jenkins in Function-as-a-Service context
• Integration test shared libraries

• downloads latest Jenkins LTS
• installs plugins as defined by a plugins.txt file
• setup .jenkinsfile-runner directory
• runs Jenkins master headless
• run a single job based on a local Jenkinsfile, then shutdown on completion.

https://github.com/ndeloof/jenkinsfile-runner

© Elektrobit (EB) 2018
Shared Libraries

Green sub-title

Share parts of Pipelines between various projects to reduce redundancies and keep code "DRY".

Functions can than be called from Jenkins files.

See


and

https://jenkins.io/blog/2017/10/02/pipeline-templates-with-shared-libraries/

for more details.
Unit Testing Jenkins Pipelines

- Allows to unit test Pipelines and Shared Libraries before running them in full
- Provides a mock execution environment that can be used to check for expected behavior
- Still quite rough around the edges. (e.g. no support for declarative pipeline yet
  https://github.com/lesfurets/JenkinsPipelineUnit/pull/13)

See:
- https://github.com/lesfurets/JenkinsPipelineUnit
- https://github.com/lesfurets/JenkinsPipelineUnit/blob/master/README.md
- https://issues.jenkins-ci.org/browse/JENKINS-33925
Things missing / Things to come

• Missing:
  – Support for definition of variables in declarative pipelines (see workaround in https://issues.jenkins-ci.org/browse/JENKINS-41335)
  – Keep build forever (https://issues.jenkins-ci.org/browse/JENKINS-39028; workaround via shared lib, change to be released)

• To Come:
  – More editor coverage of declaration syntax
  – GitLab read/write support
  – Jenkins Essentials
  – Project Cheetah https://jenkins.io/blog/2018/02/22/cheetah/
  – For more see https://jenkins.io/projects/blueocean/roadmap/
Continuous Delivery with Jenkins Pipelines (incl. Advanced Topics)

Further references & information I

- Website: https://jenkins.io
- Blog: https://jenkins.io/node/

- Getting Started
  - https://jenkins.io/doc/tutorials/
  - https://jenkins.io/doc/pipeline/steps/
  - https://github.com/jenkinsci/pipeline-model-definition-plugin/wiki/getting%20started
  - https://jenkins.io/blog/2017/05/18/pipeline-dev-tools/
Further references & information II

Docker Files:

• [https://github.com/jenkinsci/docker/blob/master/README.md](https://github.com/jenkinsci/docker/blob/master/README.md)
• [https://hub.docker.com/r/jenkinsci/blueocean/](https://hub.docker.com/r/jenkinsci/blueocean/)
