Pipeline as Code

Jenkins 2.0 and Pipeline
Continuous Delivery

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time. It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

https://en.wikipedia.org/wiki/Continuous_delivery
CD with Jenkins 1
Steps to CD in Jenkins

- Ensuring reproducible builds
- Sharing build artifacts throughout the pipeline
- Choosing the right granularity for each job
- Parallelizing and joining jobs
- Gates and approvals
- Visualizing the pipeline
- Organizing and securing jobs
- Good practice: versioning your Jenkins configuration

http://www.infoq.com/articles/orch-pipelines-jenkins
CD with Jenkins circa 2014

http://www.infoq.com/articles/orch-pipelines-jenkins
Static Code Quality Analysis Job

- **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: [ ]

- **Configuration**
Approval
Visualizing the Pipeline

Build Pipeline Plugin

Delivery Pipeline Plugin
SCM Sync Configuration Plugin

Job Configuration Difference

```
15  <hudson.tasks.BatchFile>
16   <command>@echo off
17     echo Deploying to functional test environment for iOS...
18     ping -n 11 127.0.0.1 >null
19   echo Done</command>
20  </hudson.tasks.BatchFile>
21  </builders>
```

SCM Sync configuration

- None
- Subversion
- Git

Repository URL: git@github.com:xbialabs/jenkins-config.git

Never bother me with commit messages

Display SCM Sync Status

Commit message pattern

Manual synchronization includes

Reload config from SCM
Jenkins Pipeline
Pipeline as Code

- Introduce “pipeline” as a new type in Jenkins
- Codify an implicit series of stages into an explicit Jenkinsfile in your source repository
- Resumability/durability of the pipeline state
- Extend the DSL with your own steps
- DRY - Reusable components and flows

github.com/jenkinsci/pipeline-examples
Pipeline DSL

```groovy
node('docker') {
    checkout scm

    /* Grab the abbreviated SHA1 of our pipeline’s commit.*/
    sh 'git rev-parse HEAD > GIT_COMMIT'
    def shortCommit = readFile('GIT_COMMIT').take(6)

    stage 'Build'
    def image = docker.build("jenkinsciinfra/bind:build-${shortCommit}")

    stage 'Deploy'
    image.push()
}
```
Pipeline DSL

DSL Reference

Steps

- archive: Archive artifacts
- bat: Windows Batch Script
- build: Build a job
- checkout: General SCM

This is a special step that allows you to run checkouts using any configuration options offered by any Pipeline-compatible SCM plugin. To use a concrete SCM implementation, just install the corresponding plugin and check if it is shown in the list below. Then select the SCM to use from the dropdown list and configure it as needed.

Any other specific steps to run checkouts (like svn or git) are simplistic options of this step.

scm

Nested choice of objects

$class: 'GitSCM'

$userRemoteConfigs

Specify the repository to track. This can be a URL or a local file path. Note that for super-projects (repositories with submodules), only a local file path or a complete URL is valid. The following are examples of valid git URLs:

- ssh://git@gitlab.com:9000/istio.git
- git@github.com:github/istio.git (short notation for ssh protocol)
- ssh://user@otherhost.com:/repos/istio.git (to access the repos/istio git repository in the user's home directory)
- https://github.com/istio/istio.git
- git@github.com:github/istio.git

If the repository is a super-project, the location from which to clone submodules is dependent on whether the repository is bare or non-bare (i.e. has a working directory):

- If the super-project is bare, the location of the submodules will be taken from gitmodules.
- If the super-project is not bare, it is assumed that the repository has each of its submodules cloned and checked out.
# Pipeline Stage View

## Stage View

### Average stage times:
(Average full run time: ~30s)

<table>
<thead>
<tr>
<th>Run ID</th>
<th>Date</th>
<th>Time</th>
<th>Commits</th>
<th>Build (s)</th>
<th>Deploy (s)</th>
<th>Test (s)</th>
<th>Promote (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#20</td>
<td>Mar 03</td>
<td>16:11</td>
<td>2</td>
<td>2</td>
<td>17</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>#25</td>
<td>Mar 03</td>
<td>13:11</td>
<td>2</td>
<td>2</td>
<td>16</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>#24</td>
<td>Mar 03</td>
<td>10:12</td>
<td>2</td>
<td>2</td>
<td>16</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>#23</td>
<td>Mar 03</td>
<td>07:11</td>
<td>2</td>
<td>2</td>
<td>16</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>#22</td>
<td>Mar 03</td>
<td>04:11</td>
<td>2</td>
<td>2</td>
<td>18</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>#21</td>
<td>Mar 03</td>
<td>01:11</td>
<td>2</td>
<td>2</td>
<td>17</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Pipeline Stage View

<table>
<thead>
<tr>
<th>Build docker images</th>
<th>Unit Test ubuntu-python27</th>
<th>Unit Test centos6-python26</th>
<th>Unit Test debian-wheezy using python-3.4.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4s</td>
<td>31s</td>
<td>12s</td>
<td>31s</td>
</tr>
</tbody>
</table>

Average stage times: (Average full run time: ~1 min 19s)

- #140: Mar 02 08:51, No Changes
  - Build docker images: 5s
  - Unit Test ubuntu-python27: 32s (failed)

- #139: Mar 02 08:17, No Changes
  - Build docker images: 4s
  - Unit Test ubuntu-python27: 30s
  - Unit Test centos6-python26: 12s
  - Unit Test debian-wheezy using python-3.4.3: 31s

- #138: Mar 02 00:22, No Changes
  - Build docker images: 4s
  - Unit Test ubuntu-python27: 30s
  - Unit Test centos6-python26: 12s
  - Unit Test debian-wheezy using python-3.4.3: 31s

- #137: Mar 01 21:42, No Changes
  - Build docker images: 4s
  - Unit Test ubuntu-python27: 30s
  - Unit Test centos6-python26: 12s
  - Unit Test debian-wheezy using python-3.4.3: 31s
**Durable Task Plugin**

Library offering an extension point for processes which can run outside of Jenkins yet be monitored.

### Plugin Information

<table>
<thead>
<tr>
<th>Plugin ID</th>
<th>durable-task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest Release</td>
<td>1.9 (archives)</td>
</tr>
<tr>
<td>Latest Release Date</td>
<td>Mar 24, 2016</td>
</tr>
<tr>
<td>Required Core Dependencies</td>
<td>1.609.1</td>
</tr>
</tbody>
</table>

### Changes

- In Latest Release
- Since Latest Release

### Source Code

<table>
<thead>
<tr>
<th>Issue Tracking</th>
<th>Pull Requests</th>
<th>Maintainer(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Github</td>
<td>Open Issues</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pull Requests</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Usage

<table>
<thead>
<tr>
<th>durable-task - installations</th>
</tr>
</thead>
<tbody>
<tr>
<td>03 04 05 06 07 08 09 10 11 12 01 02</td>
</tr>
<tr>
<td>0 1000 2000 3000 4000 5000 6000 7000</td>
</tr>
</tbody>
</table>

### Installations

| 2015-Mar 2401 |
| 2015-Apr 2684 |
| 2015-May 2916 |
| 2015-Jun 3467 |
| 2015-Jul 4034 |
| 2015-Aug 4324 |
| 2015-Sep 4805 |
| 2015-Oct 6455 |
| 2015-Nov 8054 |
| 2015-Dec 9683 |
| 2016-Jan 7998 |
| 2016-Feb 7791 |

Dicontinues no direct features on its own but can be used by other feature plugins. [Example](#)

**Blog post**

### Changelog

- [View changelog](#)
Extensibility

➢ CPS Global Library
➢ Remote Loader Plugin
➢ ‘load’ step
➢ Plugins
CD with Jenkins circa 2014

http://www.infoq.com/articles/orch-pipelines-jenkins
Pipeline

- Ensuring reproducible builds
- Sharing build artifacts throughout the pipeline
- Choosing the right granularity for each job
- Parallelizing and joining jobs
- Gates and approvals
- Visualizing the pipeline
- Organizing and securing jobs
- Good practice: versioning your Jenkins configuration