

Online UI/UX Hackfest

Migrating Docs to jenkins.io
(May 26, 2020)



#jenkinsIsTheWay

Agenda

- Goals of documentation migration
- Collecting, transforming, and publishing from wiki to jenkins.io
- Demo
- Wiki information categories
- Prioritizing the migration efforts
- Q&A

Slides: See the Gitter chat
<https://gitter.im/jenkinsci/docs>

Questions and Feedback

Online

- Zoom Q&A
- Hackfest office hours
- Docs office hours

Offline

- Gitter: [jenkinsci/docs](https://gitter.im/jenkinsci/docs)
- Mailing lists

User Documentation

<https://www.jenkins.io/events/online-hackfest/2020-uiux/#user-documentation>

Docs Wiki Migration

- Much useful content still in the wiki to rework & reuse
- Wiki exporter - <https://jenkins-wiki-exporter.jenkins.io/>
- GitHub issues - <https://bit.ly/wiki-migration>


Pipeline CPS method mismatches

Created by Unknown User (jglick), last modified by Unknown User (indifire) on Aug 27, 2019

Introduction

Jenkins Pipeline uses a library called Groovy CPS to run Pipeline scripts. While Pipeline uses the Groovy parser and compiler, unlike a regular Groovy environment it runs most of the program inside a special interpreter. This uses a continuation-passing style (CPS) transform to turn your code into a version that can save its current state to disk (a file called `program.dat` inside your build directory) and continue running even after Jenkins has restarted. (You can get some more technical background on the plugin page and the library page.)

While the CPS transform is usually transparent to users, there are limitations to what Groovy language constructs can be supported, and in some circumstances it can lead to counterintuitive behavior.

 **JENKINS-34344** - Running asynchronous code inside a `@NonCPS` method should fail cleanly **RESOLVED**
makes the runtime try to detect the most common mistake: calling CPS-transformed code from non-CPS-transformed code. The following kinds of things are CPS-transformed:

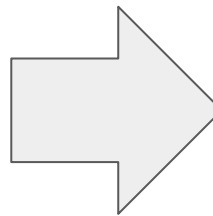
- Almost all of the Pipeline script you write (including in libraries).
- Most Pipeline steps, including all those which take a block.

The following kinds of things are *not* CPS-transformed:

- Compiled Java bytecode, including
 - the Java Platform
 - Jenkins core and plugins
 - the runtime for the Groovy language

Table of Contents

- 1 Introduction
 - 1.1 Table of Contents
- 2 Common problems and solutions
 - 2.1 Use of Pipeline steps from `@NonCPS`
 - 2.2 Calling non-CPS-transformed methods with CPS-transformed arguments
 - 2.3 Constructors
 - 2.4 Overrides of non-CPS-transformed methods
 - 2.5 Closures inside `GString`
- 3 False Positives
 - 3.1 Direct invocation of closures stored in object fields or maps



User Handbook

- User Handbook overview
- Installing Jenkins
- Using Jenkins
- **Pipeline**
 - Getting started with Pipeline
 - Using a `Jenkinsfile`
 - Running Pipelines
 - Branches and Pull Requests
 - Using Docker with Pipeline
 - Extending with Shared Libraries
 - Pipeline Development Tools
 - Pipeline Syntax
 - Pipeline Best Practices
 - Scaling Pipelines
- Blue Ocean
- Managing Jenkins
- System Administration
- Scaling Jenkins
- Appendix
- Glossary

Tutorials

- Guided Tour
- Jenkins Pipeline
- Using Build Tools

Resources

- Pipeline Syntax reference
- Pipeline Steps reference

Pipeline

This chapter covers all recommended aspects of Jenkins Pipeline functionality, including how to:

- **get started with Pipeline** - covers how to define a Jenkins Pipeline (i.e. your `Pipeline`) through Blue Ocean, through the classic UI or in SCM,
- **create and use a `Jenkinsfile`** - covers use-case scenarios on how to craft and construct your `Jenkinsfile`,
- **work with branches and pull requests**,
- **use Docker with Pipeline** - covers how Jenkins can invoke Docker containers on agents/nodes (from a `Jenkinsfile`) to build your Pipeline projects,
- **extend Pipeline with shared libraries**,
- **use different development tools** to facilitate the creation of your Pipeline, and
- **work with Pipeline syntax** - this page is a comprehensive reference of all Declarative Pipeline syntax.

For an overview of content in the Jenkins User Handbook, see [User Handbook overview](#).

What is Jenkins Pipeline?

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

Chapter Sub-Sections

[Getting started with Pipeline](#)
[Using a `Jenkinsfile`](#)
[Running Pipelines](#)
[Branches and Pull Requests](#)
[Using Docker with Pipeline](#)
[Extending with Shared Libraries](#)
[Pipeline Development Tools](#)
[Pipeline Syntax](#)
[Pipeline Best Practices](#)
[Scaling Pipelines](#)

Table of Contents

[What is Jenkins Pipeline?](#)
 [Declarative versus Scripted Pipeline syntax](#)
[Why Pipeline?](#)
[Pipeline concepts](#)
 [Pipeline](#)
 [Node](#)
 [Stage](#)
 [Step](#)
[Pipeline syntax overview](#)
 [Declarative Pipeline fundamentals](#)
 [Scripted Pipeline fundamentals](#)
[Pipeline example](#)



Jenkins

Choosing a Page to Migrate

[GitHub issues](#) for top 50 pages

- [wiki-migration label](#)

[Wiki conversion sheet](#) for all accessed pages

- Many wiki pages still to triage
- Many plugins still to convert to documentation as code

Demo

Categorizing Wiki Pages

- Using Jenkins
 - [GitHub project](#)
 - [Chapter](#) on jenkins.io
- Managing Jenkins
 - [Chapter](#) on jenkins.io
- Administering Jenkins
 - [GitHub project](#)
 - [Chapter](#) on jenkins.io

Prioritizing the Migration

- Work on things that interest you
- [GitHub issues](#) for top 50 pages
- [Wiki conversion sheet](#) for all accessed pages

User Documentation Track. Links

- All project ideas
 - <https://www.jenkins.io/events/online-hackfest/2020-uiux/#user-documentation>
- jenkins.io
 - Issues: [here](#)
 - Newcomer-friendly Issues: [here](#)
- Plugin issue links
 - All issues: [Jira](#), [GitHub Issues](#)
 - Newcomer-friendly issues: [Jira](#), [GitHub Issues](#)

Questions?



Have fun!

#JenkinsIsTheWay