How to Configure Jenkins for CI/CD to Docker

Developer Guide

Contents

[Introduction: 2](#_Toc515903592)

[Getting Started on Jenkins Central: 2](#_Toc515903593)

[Setting Up: 2](#_Toc515903594)

[Configuring a Pipeline: 3](#_Toc515903595)

[How to create a simple webhook on github to trigger your Pipeline: 5](#_Toc515903596)

[How to Create a Jenkinsfile: 6](#_Toc515903597)

[How it works: 7](#_Toc515903598)

[Best Practices: 7](#_Toc515903599)

[Reference Section: 8](#_Toc515903600)

[Known Issues: 8](#_Toc515903601)

[Revision History: 8](#_Toc515903602)

# Introduction:

If you’re looking for instructions on how to setup Jenkins to automatically build and push Docker images using Optum’s enterprise tools, you’ve come to the right place.

This guide assumes:

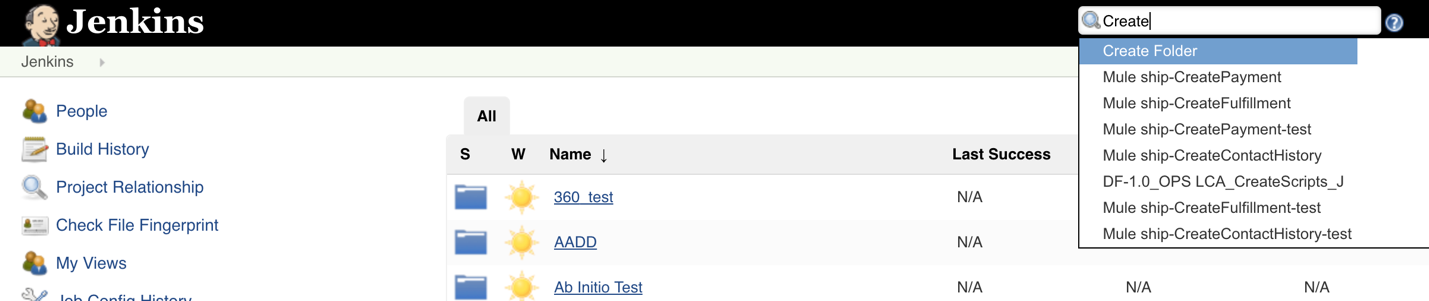
* You are using Optum’s Jenkins Central: <https://jenkins.optum.com/central/>
* You already understand how to build & push Docker images, but you want to automate this process using standard CI/CD methods via Jenkins.

# Getting Started on Jenkins Central:

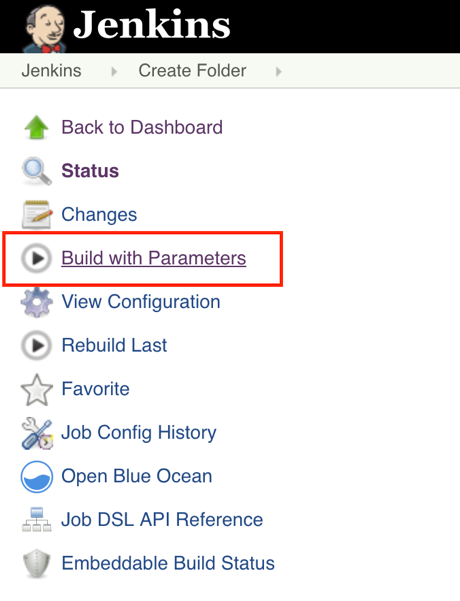
This section will quickly show you how to setup your Jenkins folder and configure a pipeline from scratch.

## Setting Up:

1. Navigate to Jenkins Central (<https://jenkins.optum.com/central/>) and search ‘Create Folder’.



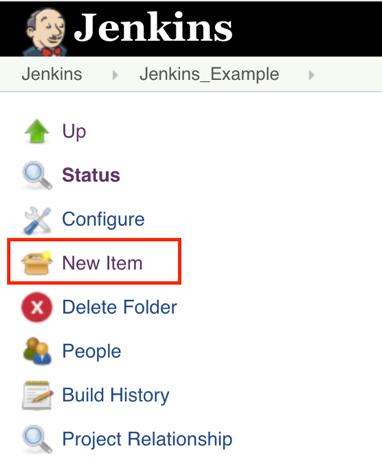
1. On the left side, click ‘Build with Parameters’ and enter in a name for your folder.



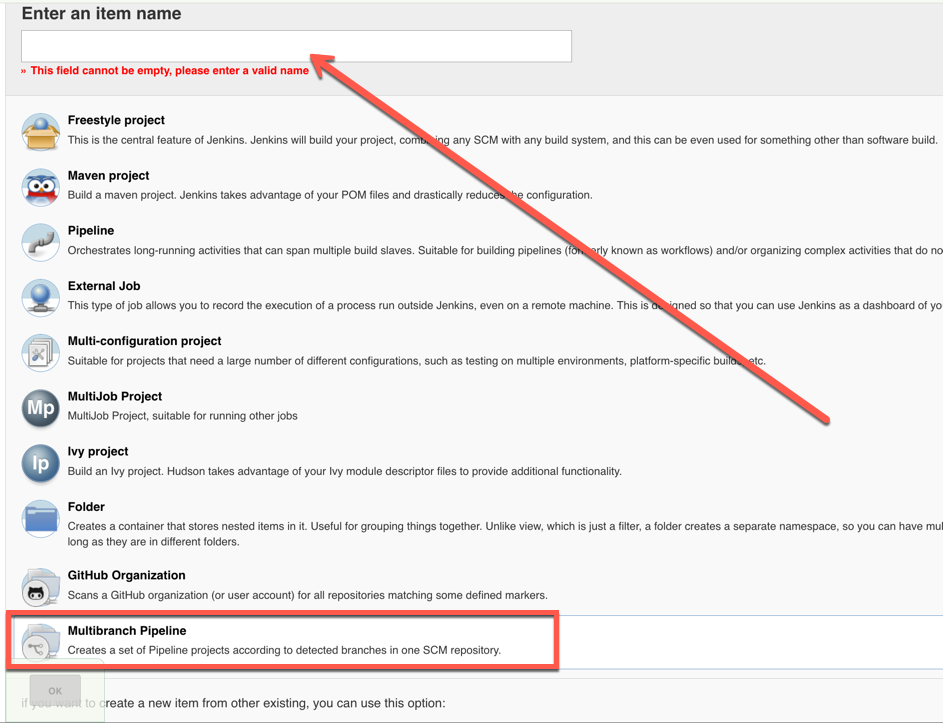
1. You have successfully setup your Jenkins folder. In the search box again, search for your folder name and go to it.

## Configuring a Pipeline:

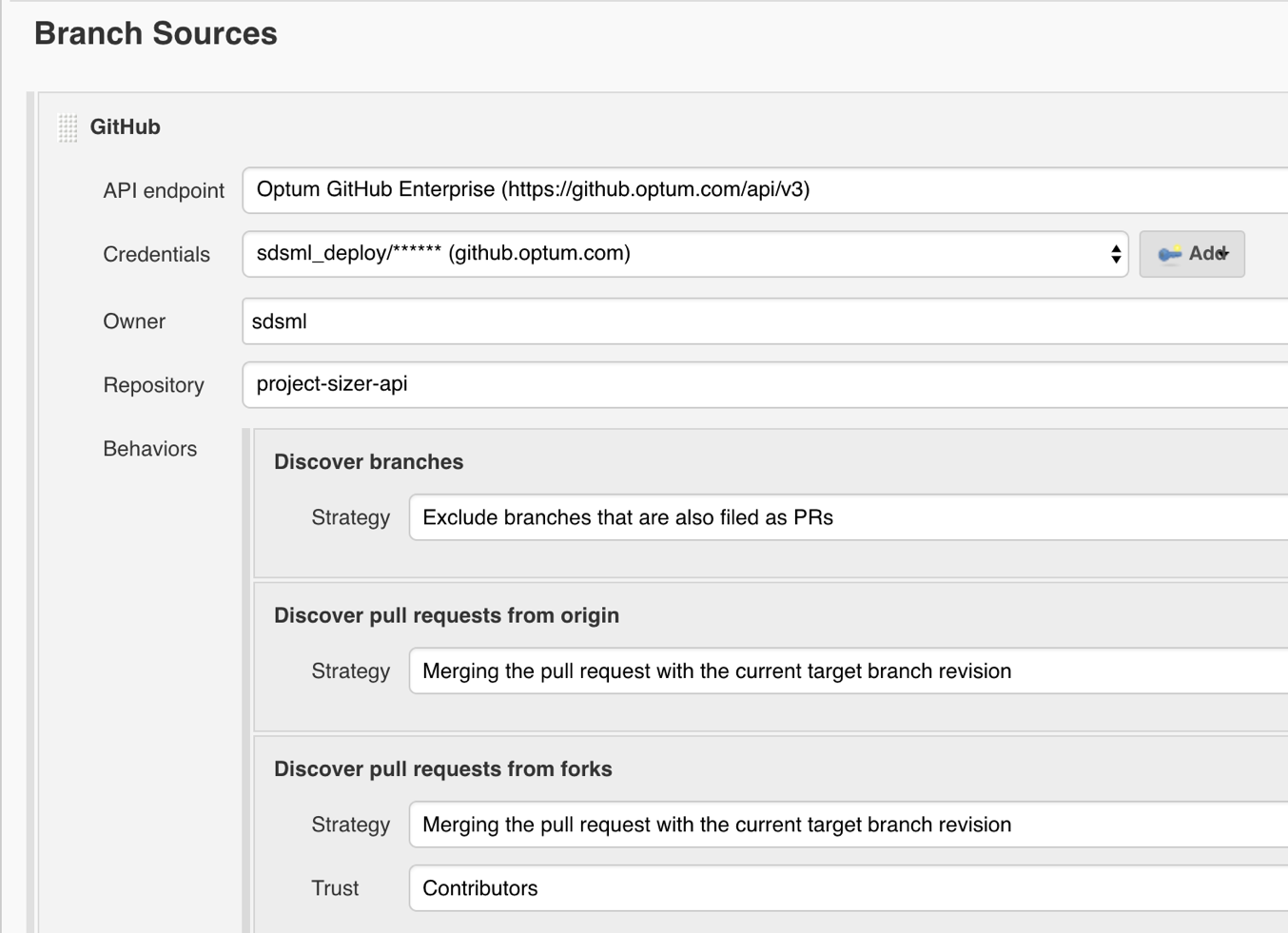
1. From your Jenkins folder on the left side, click ‘New Item’.



1. Select ‘Multibranch Pipeline’ and enter a descriptive name.



1. Under ‘Branch Sources’, enter your project repository and credentials.
   1. Please reference this guide for how to setup credentials on Jenkins: <https://hubconnect.uhg.com/docs/DOC-126725>



1. Under ‘Build Configuration’, specify that your build will be configured using a Jenkinsfile. We will be leveraging the JPAC (Jenkins Pipeline as Code) library, which is an internal open source library built by Optum to do Jenkins CI.

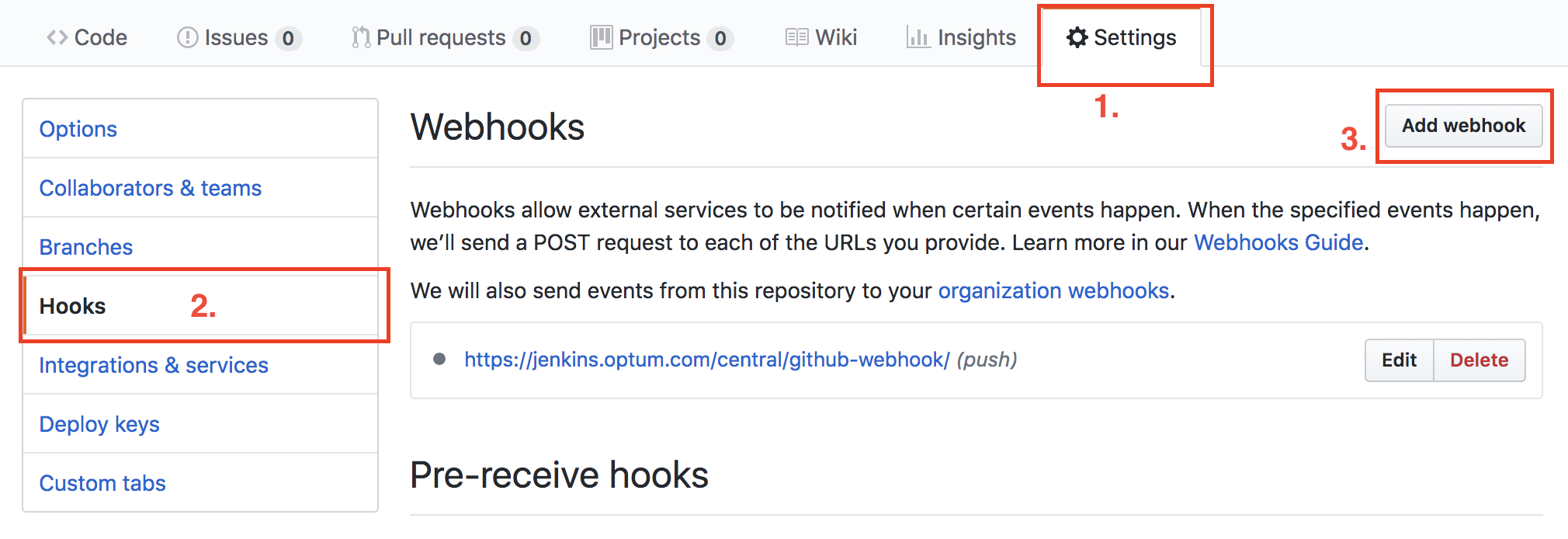


1. Click ‘Save’. Now you have successfully configured your Jenkins pipeline.

# **How to create a simple webhook on github to trigger your Pipeline**:

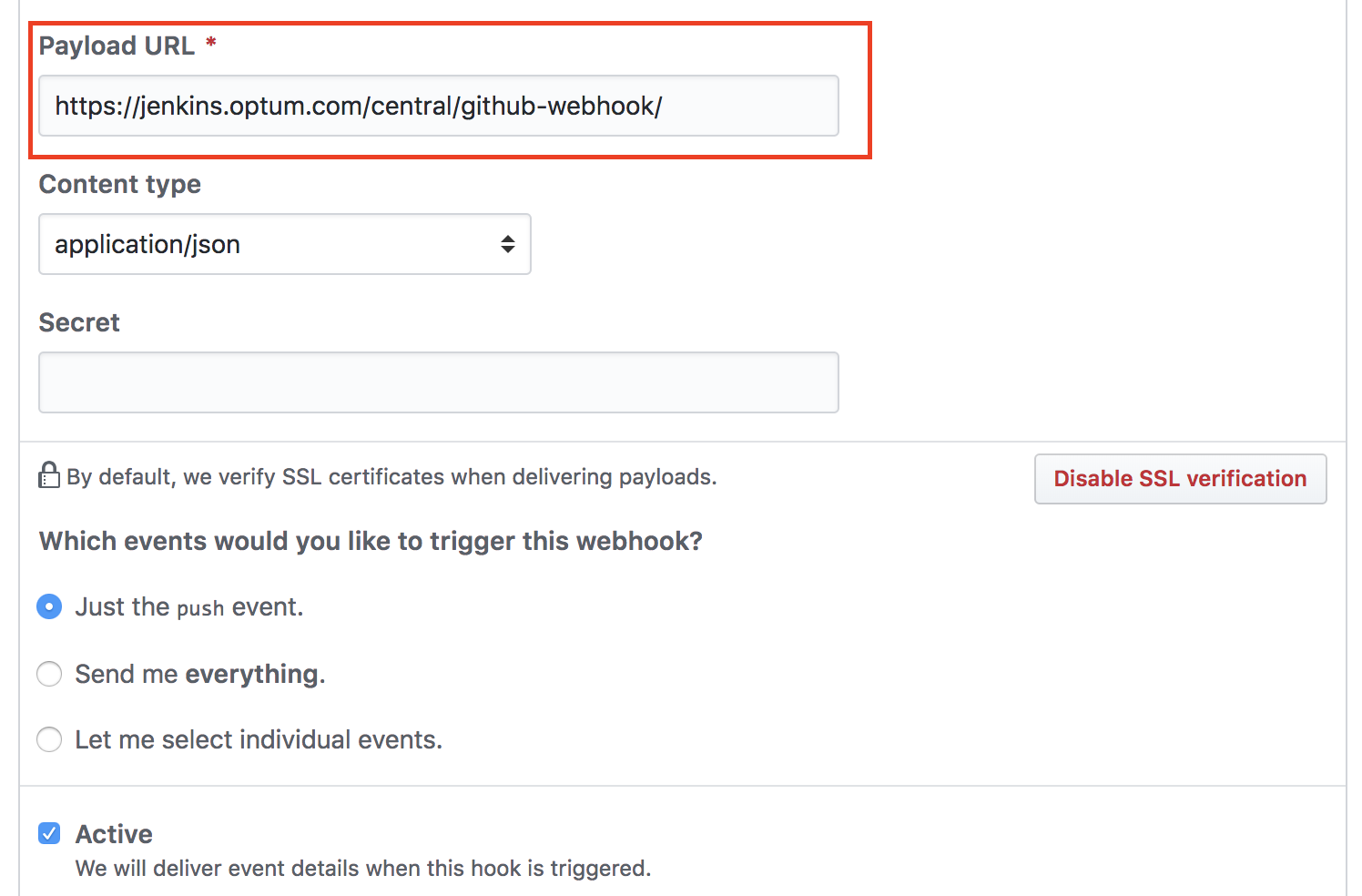
This section will quickly help you enable webhook on your github repository so that Jenkins can start listening to events and trigger automated builds accordingly.

* + - 1. On your git repository page, navigate to ‘Settings’.
      2. On the left side, click ‘Hooks’.
      3. On the right side, click ‘Add webhook’.



1. Use this payload: <https://jenkins.optum.com/central/github-webhook/>

Jenkins central will automatically know to direct the incoming events to your folder since you have specified your ‘Branch Sources’ already.



1. Click ‘Add webhook’ and you are done setting up the webhook.

# **How to Create a Jenkinsfile**:

This section will focus on creating a Jenkinsfile specifically for building a Docker image and pushing it to Dockerhub.

* + - 1. A Jenkinsfile is essentially a groovy script that uses an open source library within Optum called JPaC (Jenkins Pipeline as Code).

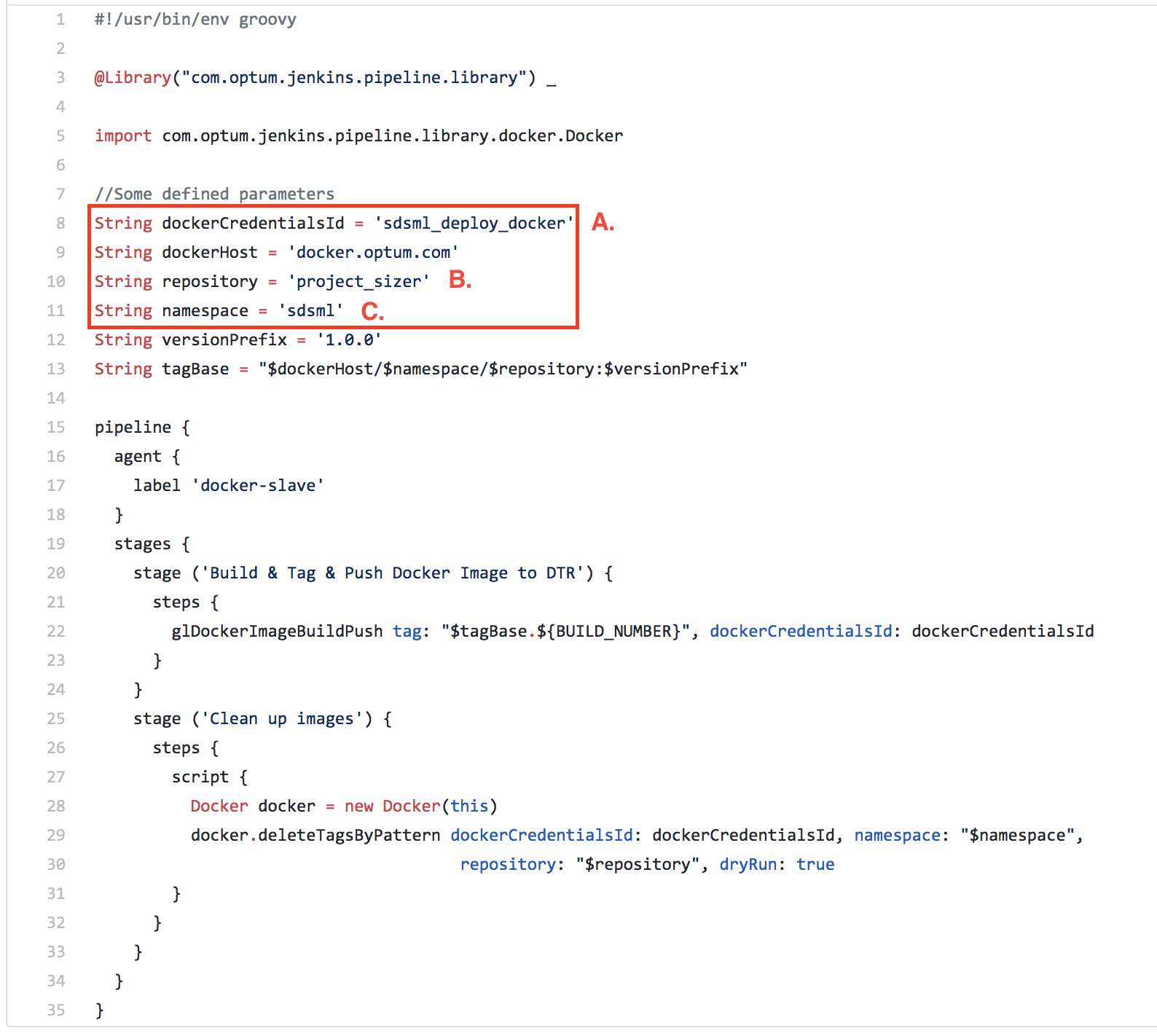
1. The open source code can be found here:

<https://github.optum.com/jenkins-pipelines/global-pipeline-library>

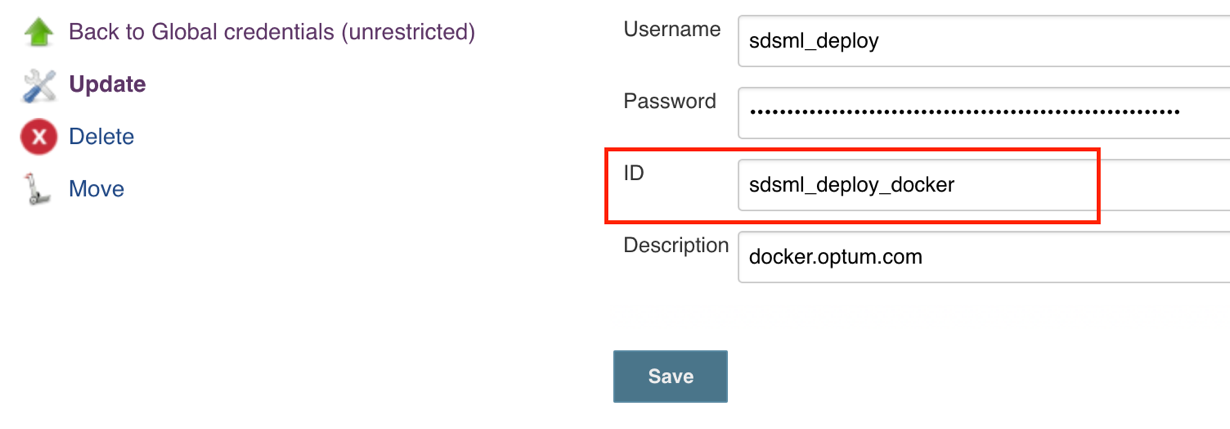
1. Here is a template Jenkinsfile that is used for building & pushing Docker image:

<https://github.optum.com/jenkins-pipelines/example-dockerImageBuild-deployToDTR/blob/master/Jenkinsfile>

* + - 1. Use the template and fill in the information like below:



1. This is the exact ID you used when setting up your docker Credentials on Jenkins.



1. The name of your docker repository.
2. The organization or namespace that your docker repository is located in.
   * + 1. Save this file as ‘Jenkinsfile’ with no extension and commit it to your git repository alongside your Dockerfile or pom.xml.

# How it works:

Your Jenkins pipeline will listen to events from your git repository. When there is a push event from your git repository, the pipeline will check to see which branch has changed and if that branch has a Jenkinsfile. A corresponding build will be triggered for that specific branch using the Jenkinsfile it has found.

* Any branch without a Jenkinsfile will not get built.

# Best Practices:

1. Avoid using plugins on Jenkins Central as they are outdated.
2. Keep the name of the file 'Jenkinsfile' for simplicity and consistency.
3. Join the Optum Flowdock group called ‘JPaC’. This group will have many who can assist and answer your questions regarding Jenkins. <https://www.flowdock.com/app/uhg/jenkins-pipeline-as-code>

# Reference Section:

JPaC Open Source Code

<https://github.optum.com/jenkins-pipelines/global-pipeline-library>

Jenkins Pipeline as Code Homepage

<https://hubconnect.uhg.com/docs/DOC-129470>

Step-By-Step Guide: Maven Build, Sonar Scan and Artifactory Deploy

<https://hubconnect.uhg.com/docs/DOC-126587>

# Known Issues:*[Record any issues here, including issues found during the Architecture Review process. If there are remediation plans, include information about them in the Details section. In the severity column, indicate low, medium or high. Low severity issues are informational and do not require a specific time frame for remediation. Medium issues require a remediation plan and a timeframe to be in place before the SAD can be approved. High Severity issues will prevent the SAD from being approved until the item is remediated or moved to a medium severity. The Owner is the person who will follow up on the issue to ensure it gets the attention it needs. It is not necessarily the person who will actually do the work of remediation. The status should be either open or closed. Closed items are not removed from the SAD to provide a history of issues and work across multiple releases of the SAD. ]*

| # | Issue | Severity | Status | Date  Closed | Issue Details |
| --- | --- | --- | --- | --- | --- |
| 1 | *Password with special characters will leak in build logs* | *Medium* | OPEN |  | Using special characters in your credentials may cause issues and your password will be visible to others in your build history log. This issue has been reported to Jenkins Central on December 2017. |
| 2 | *Other Known Issues* | *NA* | NA | *NA* | Please see: <https://hubconnect.uhg.com/docs/DOC-74835> |

# Revision History:

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 6/04/2018 | 0.1 | Initial version of the document. | Triet Pham |