**Multi-Branch Pipeline Report**

**Contents**

[**1.** **Background** 3](#_Toc169092018)

[**2.** **New Plugins** 4](#_Toc169092019)

[**3.** **Configuration Setting** 5](#_Toc169092020)

[**4.** **Notable Characteristics** 11](#_Toc169092021)

[**5.** **Behaviors** 12](#_Toc169092022)

[**5.1.** **Discover branches** 12](#_Toc169092023)

[**5.2.** **Discover pull requests from origin** 12](#_Toc169092024)

[**6.** **Conclusion** 16](#_Toc169092025)

1. **Background**

: The single branch pipeline handles all the build requests so, tons of build history will make it hard to keep track of the project building result. However, each branch can have its own pipeline in a muti-branch pipeline, so it can make more organized building environments and easy to manage or monitor the system.

For example, we can display only the branches that are processing PR in the muti-branch pipeline dashboard, it will be discussed in ‘5.1. Discover branches’ content.

However, we decided it is not an appropriate time to apply a multi-branch pipeline to the VARLAB pipeline based on two reasons.

**1. Lack of capacity**

: One of the advantages of the multi-branch pipeline is that it can build multiple jobs associated with each branch simultaneously.

However, currently, our Jenkins can handle only 3 jobs in one time, so we can not fully use the characteristic of the multi-branch pipeline.

So, it seems it is premature to apply a multi-branch pipeline to the current working environment.

**2. Sigle Pipeline**

: Currently, DXL teams are using only two pipelines, PR and Deployment, with a single logic of streamlining.

So, even though a multi-branch pipeline is applied to Jenkins, we will need to use only one pipeline.

Therefore, it is better to discuss using a multi-branch pipeline after we face the situation to extend the current pipeline.

1. **New Plugins**

* **Bitbucket Branch Source Plugin**

**[(Behaviours) Discover pull requests from origin – Build Staragy]**

: This setting determines how PRs (Pull Requests) are explored and processed in Jenkins' Multi-Branch Pipeline settings.

The reason this setting is relevant to builds is that depending on how to navigate the PR, Jenkins determines which code to pull and build.

* **Option 1**. **Merging the pull request with the current target branch revision**

**(Not Recommended)**

: This option discovers the PR as if it has already been merged with the target branch.

>> The build and test process occurs using the state after the PR is merged into the target branch, helping preemptively resolve any merge conflicts or integration issues.

* **Option 2. The current pull request revision**

: This option discovers the PR and checks out the latest revision of the corresponding branch.

>> Basically, it is the same building process as a single-branch pipeline

* **Option 3. Both the current pull request revision and the pull request merged with the current target branch revision (Not Recommended)**

: This option discovers both branches which are merged with the target branch and the latest revision of the corresponding branch. Both projects will be built in the same time.

* **Remote Jenkinsfile Provider**

: In the multi-branch pipeline, we need an extra plug-in to pull the Jenkins file from the different repository. The ‘Remote Jenkinsfile Provider’ is the plugin to get the Jenkins file and other scripts from outside of the PR repository.

Basically, a multi-branch pipeline tries to find the Jenkins file inside the working directory in the in multi-branch pipeline building time.

1. **Configuration Setting**

: Each step to set up a multi-branch is included in this content. Creating a multi-branch pipeline, environment set-up, and managing credentials will be discussed in detail.

1. **Create New Project**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Select the ‘New Item’ to add the new project for the muti-branch pipeline.

1. **Create Multi-Branch Pipeline Project**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Enter the name of the project(folder) name and select the ‘Multibranch Pipeline’ for the type of project.

1. **Set the ‘Branch Sources’ to Bitbucket**

텍스트, 스크린샷, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

1. **Create the ‘App Passwords’ in the Bitbucket**

**(Optional – No need to do this if there is an app password credential already)**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

>> Go to the ‘Personal Bitbucket Settings’

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

>> Go to the ‘App passwords’ and select the “Create app password” button.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

>> Enter the app password name at the ‘Lable’ and check two options.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

>> Copy the access code

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

>> Now, go to the Jenkins > Dashboard > Mange Jenkins > Credential

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

>> Select the global

>> However, if you can see the green-labeled credential, skip this step and just use that.

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

>> Click the “+ Add Credential”

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

>> You can see the new credential is created

1. **Set the credential and select a repository**

텍스트, 스크린샷, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

>> 1. Select the validated credential

>> 2. Make sure the owner should be the “**varlab**” 텍스트, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

>> 3. If you enter the Owner, Jenkins will automatically detect the available repositories in the Bitbucket.

>> If Jenkins does not show the list of Bitbucket repositories, double-check the credential. (Suggest just go to the previous step and create a new app password if you skipped it)

1. **Set Behavior (Discover branches)**

**– Recommend to choose just one 3.6 or 3.7**

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

>> Select the strategy how to detect and display the branches

>> Each strategy is explained in ‘5.1. Discover branches’

>> However, select the ‘Only branches that are filed as PRs’ as a default

1. **Set Behavior (Discover pull requests from origin)**

**- Recommend to choose just one 3.6 or 3.7**

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

>> Select the build strategy

>> Each Strategy is explained in the 5.2. Discover pull requests from origin

>> However, select the ‘The current pull request revision’ as a default

1. **Build Configuration Setting**

스크린샷, 텍스트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

>> Select the ‘by Remote Jenkinsfile Provider Plugin’

>> This plug-in allows us to use the Jenkinsfile from a different directory.

>> Enter the relative path of the Jenkins file

텍스트, 스크린샷, 번호, 평행이(가) 표시된 사진

자동 생성된 설명

>> Select the ‘Git’ as a Jenkinsfile SCM

>> Copy and past the git repository which includes the Jenkins file

>> Select the valid credential to access to the repository

>> Enter the name of the branch you want to get the Jenkins file (e.g. main)

1. **Manage the Webhook**

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

>> Go to the Repository settings > Webhook > Add webhook

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

>> Enter the name of the webhook and the URL with the ‘trigger token’

>> You can use anything you want for the trigger token, but just being aware of it will be used in Jenkins to trigger the specific job for the request.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

>> Check the trigger options

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

>> Go to multi-branch pipeline folder > credential > Stores scoped to NAME\_OF-FOLDER > Global credential

>> Add the Credential likewise explained in 3.4 Create the ‘App Passwords’ in the Bitbucket

>> Make sure the name of the credential should ‘trigger-token’

>> Make sure the secret text should be matched with the Webhooks’ trigger token

1. **Notable Characteristics**
2. **Working Directory in the VM**

: In a single branch pipeline, the Jenkins file and other script files exist in the same working directory, so no need to navigate the scripting file to run a build.

However, in a multi-branch pipeline, the ‘Remote Jenkinsfile Provider’ creates the script folder outside of the working directory like the below image.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

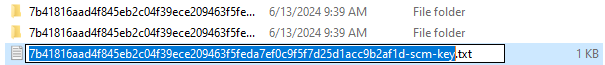
자동 생성된 설명

So, we need to navigate to the directory that actually has scripting files when we want to call other scripting files.

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Additionally, the strange hash code, 7b41816aad4f845eb2c04…, comes from the SCM hash key. It is hard-coded inside of the multi-branch pipeline Jenkins file and it was reported at the <https://varlab-dev.atlassian.net/browse/CORE-713>



1. **Jenkins hook**

: When a multi-branch pipeline has been set up completely, the ‘Jenkins hook’ will be automatically created by the multi-branch pipeline plugin like the below image.

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

This webhook will trigger a build one more time, so we want to disable this webhook.

>> Firstly, make sure the ‘Webhook to Jenkins for Bitbucket’ option is disabled like the below image.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

>> And **inactivate** the ‘Jenkins hook’ Do not delete it. It won’t act as you expect.

>> If you delete it, it will regenerated by the Jenkins plugin and cause the same problem.

**텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명**

1. **Behaviors**

: In a multi-branch pipeline we can’t change each branch configuration setting, so we need to add a specific behavior if we want a certain feature to each job.

1. **Discover branches**

: Jenkins will search and display each branch based on this condition.

* **Strategy 1. Exclude branches that are also filed as PRs**

: Jenkins will discover all branches except PR branches.

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* **Strategy 2. Only branches that are also filed as PRs**

: Jenkins will discover the branches that create PR.

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

* **Strategy 3. All branches**

: Jenkins will discover all the branches in the selected repository.

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

자동 생성된 설명

1. **Discover pull requests from origin**

* **Strategy 1. Merging the pull request with the current target branch revision**

: Discover each pull request once with the discovered revision corresponding to the result of merging with the current revision of the target branch

>> When this strategy is selected, Jenkins will create a two-build job the one named as a ‘branch name’ and another as a ‘PR-Number’.

>> So, when Jenkins is triggered by the Bitbucket Webhook(Create or Update), it triggers two jobs. This concept will be discussed below with screenshots.

**[Branch Build – dashboard feature]**

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

: This is the screenshot of the job in a multi-branch pipeline. The name of the job is the same name as the branch and it can not be changed.

The branches will be detected depending on the strategy of the “Discover branches” behavior. For example, Jenkins can detect only the branches that create PR.

**[Barnch Build – process example]**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

: In this job, it builds the project in the current branch environment using a Jenkins file.

**[Pull Request Build – dashboard feature]**

텍스트, 폰트, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

: If the multi-branch pipeline is using a “Merging the pull request with the current target branch revision” strategy, Jenkins tries to merge the PR branch to the destination branch and builds that project.

[**Pull Request Build – process example**]

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

: This screenshot shows the process by which Jenkins tries to merge the destination branch(main) to the PR branch(Multi-Branch-Pipeline-Test), and the merge conflict occurs while it tries to check out again to the main branch after the merge.

Like this example, this build strategy can detect merge conflicts or some problems in a project that can occur after the merge.

* **Strategy 2. The current pull request revision**

: It builds the PR branch only.

* **Strategy 3. Both the current pull request revision and the pull request merged with the current target branch revision**

: It builds two project in a one time which are one merged with the destination branch and another just a PR branch.

1. **Conclusion**

As mentioned in the ‘Background,’ a multi-branch pipeline is not a necessary option for our pipeline now.

However, it can be considerable in a situation where extending the pipeline or setting up a different pipeline for each branch in the same project.

In that situation, researching more about the behaviors and strategies we can use in a multi-branch pipeline.