**Research Document for merging two Jenkins CI/CD pipeline**

Integrating two Jenkins pipelines can be achieved in several ways depending on your specific requirements and environment. Here are some common approaches:

* Pipeline Trigger:

One pipeline can trigger the execution of another pipeline using the build step or Jenkins API.

For example, you can use the build step within a Jenkins pipeline to start another pipeline with parameters.

* Multibranch Pipeline:

If both pipelines are related and share common configurations, you can organize them under a multibranch pipeline.

Multibranch pipelines automatically detect branches or repositories and execute the pipeline defined within them.

* Pipeline Steps:

Jenkins provides steps to interact with other pipelines or jobs, such as build, build job, or trigger remote job.

These steps allow one pipeline to trigger another and pass parameters or receive information back from the downstream pipeline.

* Shared Libraries:

If there are common functions or steps used across pipelines, you can create a shared library.

Shared libraries allow you to define reusable code and share it among multiple pipelines, thus promoting code reusability and maintainability.

* Upstream/Downstream Projects:

Jenkins allows you to define upstream and downstream relationships between projects.

You can configure one pipeline as a downstream project triggered by the completion of another upstream pipeline.

* Webhooks and Event-driven Integration:

Configure webhooks or use event-driven triggers to initiate one pipeline based on events in another system or pipeline completion.

This approach allows for real-time integration and automation based on external events.

* Plugins:

Jenkins has numerous plugins available that extend its functionality.

Explore plugins that facilitate pipeline integration, such as the Pipeline Shared Groovy Libraries Plugin or the Parameterized Trigger Plugin.

**Approved Method:**

Multibranch Pipeline

**Procedure:**

Pipeline ‘Studio Ghibli’ on Jenkins was merged with ‘frontend’ pipeline.

This task involves merging two existing pipelines to streamline the development and deployment process. The first pipeline currently handles GitHub checkout, Node.js application dependency installation, project building, and SonarQube analysis. The second pipeline is responsible for building a Docker image and uploading it to Docker Hub.