# **Documentation: Exporting Binary Files from DFS (Domino File System) to GitLab**

## **Overview**

This document provides a step-by-step guide to exporting binary files from the Domino File System (DFS) to GitLab. The process involves reading metadata from a JSON-based repository, reconstructing commit histories with the correct binaries, and pushing the final repository to a GitLab remote. Additionally, a CSV file is used to manage project configurations, which should be maintained on GitLab.

## **Requirements**

* Access to DFS where the binary files are stored.
* A GitLab repository for storing the final reconstructed binaries.
* A CSV file containing project configurations (stored in GitLab).
* Access to Kubernetes (if running the script inside a pod).

## **Process Overview**

The export process follows these key steps:

1. **Preparing the Project CSV File**:
   * The CSV file contains essential project information, including the source and target Git URLs, tokens, and project identifiers.
   * This file should be stored in a GitLab repository and regularly updated to reflect current projects.
2. **Cloning the JSON Repository**:
   * Clone the repository containing JSON metadata (the "fake" repository) for each project.
   * This repository provides the commit history and content hashes for reconstructing the true binaries.
3. **Extracting Binary Files from DFS**:
   * For each commit, the content hash in the JSON file is used to locate the corresponding binary file in DFS.
   * The binary may be in a ZIP format or as a standalone file. If in ZIP format, it’s extracted to obtain the actual binary.
4. **Recreating Commits with Binary Files**:
   * Using the JSON repository’s commit history, each commit is reconstructed in a new repository.
   * For each commit, the corresponding binary file is added to the repository, replicating the metadata from the JSON repository.
5. **Pushing the Reconstructed Repository to GitLab**:
   * After reconstruction, the new repository (with real binaries) is pushed to the GitLab target specified in the CSV file.
   * The push operation is done using GitLab access tokens for authentication.
6. **Error Handling and Logging**:
   * If any step fails (e.g., missing binaries or repository access issues), the script logs the error and continues to the next project.
   * This ensures that errors in one project do not stop the entire process.

## **Detailed Steps**

### **Step 1: Configuring the CSV File**

The CSV file must be formatted with the following columns:

* **ProjectSourceID**: Identifier for the project in DFS.
* **GitUrlJson**: GitLab URL of the JSON repository with metadata.
* **GitUrlBinaries**: GitLab URL where the binary repository will be pushed.
* **Token**: GitLab access token with push access to the target repository.

Example of a CSV file:

**Note**: Ensure this CSV file is stored on GitLab for ease of access and versioning.

### **Step 2: Cloning the JSON Repository**

Each project’s JSON repository is cloned using the GitLab URL and access token. The repository contains JSON files with the commit history and content hash information for each file.

python

Copier le code

* clone\_json\_repo(git\_url\_json, token, json\_repo\_dir)

### **Step 3: Extracting Binary Files from DFS**

For each commit, the JSON metadata is read to find the contentHash. Using this hash, the binary file is located and extracted from DFS. If the binary is in a ZIP file, it’s extracted; otherwise, the binary file is copied directly.

python

Copier le code

* extract\_file\_from\_blob(content\_hash, target\_file)

### **Step 4: Recreating Commits with Binary Files**

Using the commit history from the JSON repository, each commit is recreated in a new Git repository. The actual binary file replaces the JSON metadata in each commit, and the original commit metadata (author, date, message) is retained.

python

Copier le code

* recreate\_commit\_with\_binaries(new\_repo, commit, old\_repo)

### **Step 5: Pushing the Reconstructed Repository to GitLab**

After the repository has been reconstructed with binaries, it is pushed to the GitLab repository specified in the CSV file. Authentication is done using the token from the CSV file.

python

Copier le code

* push\_to\_target\_repo(binaries\_repo\_dir, git\_url\_binaries, token)

### **Step 6: Error Handling**

If the target GitLab repository for the binaries does not exist, a message is displayed, and the process moves to the next project in the CSV.

### **Script Execution**

To run the script, ensure that:

* You have access to DFS and the required GitLab repositories.
* You’ve updated the csv\_path variable in the script to point to the CSV file location.

Run the script:

bash

Copier le code

* python3 migration\_git.py

## **Important Considerations**

1. **CSV File Management**: The CSV file must be accurate and up-to-date, as it controls which projects are processed.
2. **DFS Access**: Ensure that the DFS file system is accessible and that you have permissions to read from the blob storage directory.
3. **GitLab Access Tokens**: Tokens must have appropriate permissions to access and push to the GitLab repositories.
4. **Error Handling**: If an error occurs, the script logs it and continues with the next project to ensure full processing of the CSV file.