SCM MIGRATION

Contents

INTRODUCTION:

SCM (Source Code Management) migration refers to the process of transitioning from one SCM system to another. This could involve switching from one version control system to another, such as moving from Subversion (SVN) to Git, or it could involve changing the hosting platform or repository provider, such as moving from an on-premises solution to a cloud-based service like GitHub or Bitbucket.

The importance of SCM migration can vary depending on the specific context and goals of the organization, but here are some common reasons why SCM migration is considered important:

1. **Modernization and feature enhancements: Newer SCM systems often provide advanced features and capabilities that can improve the efficiency, productivity, and collaboration of development teams. Migration allows organizations to take advantage of these modern features and enhancements.**

2. **Distributed and scalable workflows: Distributed version control systems like Git offer decentralized workflows, which allow for better collaboration, branching, and merging capabilities. Migrating to such systems can facilitate more flexible and scalable development processes.**

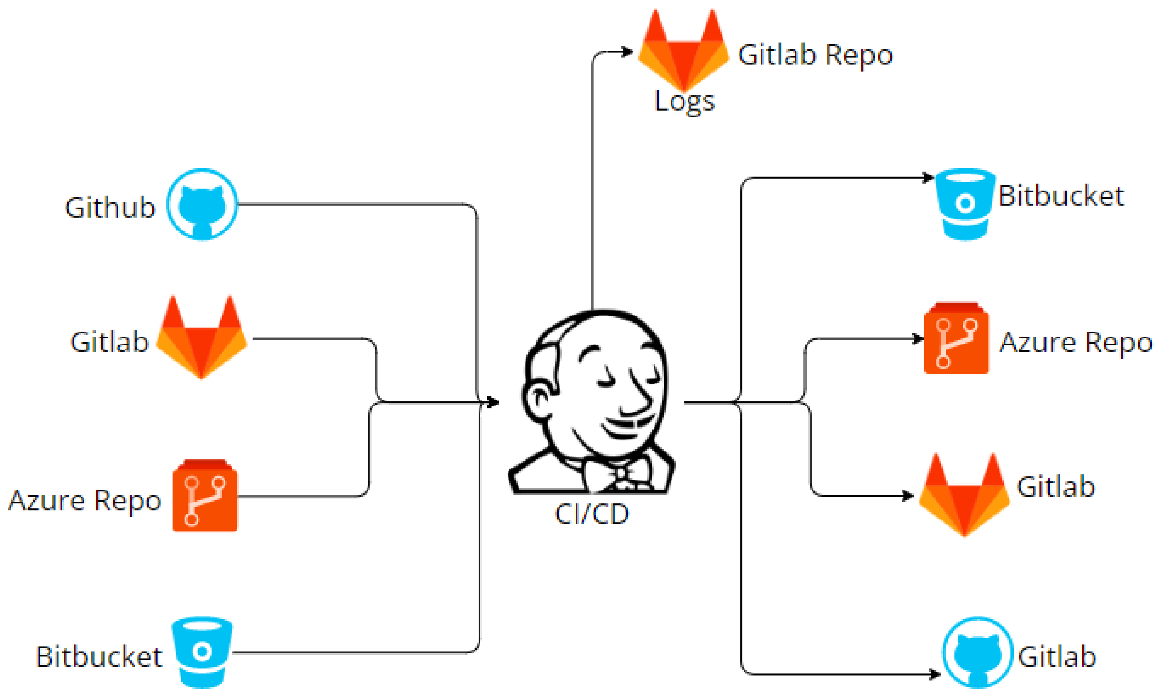
3. **Ecosystem and tooling support: Popular SCM.**

Documentation – Raw Notes

Prerequisite:

1. **Any CICD tool (preferably Jenkins)**
2. **Source Code management for migration.**
3. **PAT token for respective SCM.**

Backend management:



Project Code Repository:

You can find the code at https://github.com/vinayhegde105/SCM-Migration-DEMO.git

Jenkin Server URL: **http://34.125.169.254:8080/**

Test Credentials:

Username: tek

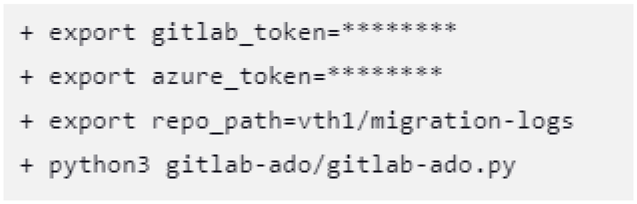
Password: tek@321

Installation Steps:

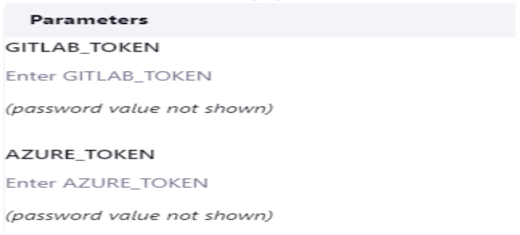
1. Login to your Jenkins Server.
2. Click on + New Item
3. Name your Project and create “PIPELINE” Project.
4. Click on Pipeline in the left menu.
5. Select Definition as “Pipeline script from SCM”
6. Clone the code to local from above mentioned address.
7. Edit “.xlsx” file to your requirements and push to SCM.
8. Paste the address of Code repository.
9. Select your branch.
10. Jenkinsfile will be on each folder and reference the Jenkinsfile.
11. Build the job.
12. Provide the necessary parameter/input to the job and click on build.
13. Wait for the pipeline to get complete.
14. You can find the logs on your Gitlab repository which you have mention during the input.
15. Verify the branch count, commit count, Merge request etc. of source repo and target repo.

Key Points:

1. Easy to Set-up. Just takes few clicks to end to end pipeline.
2. No credentials showed anywhere during or after the execution of the pipeline.

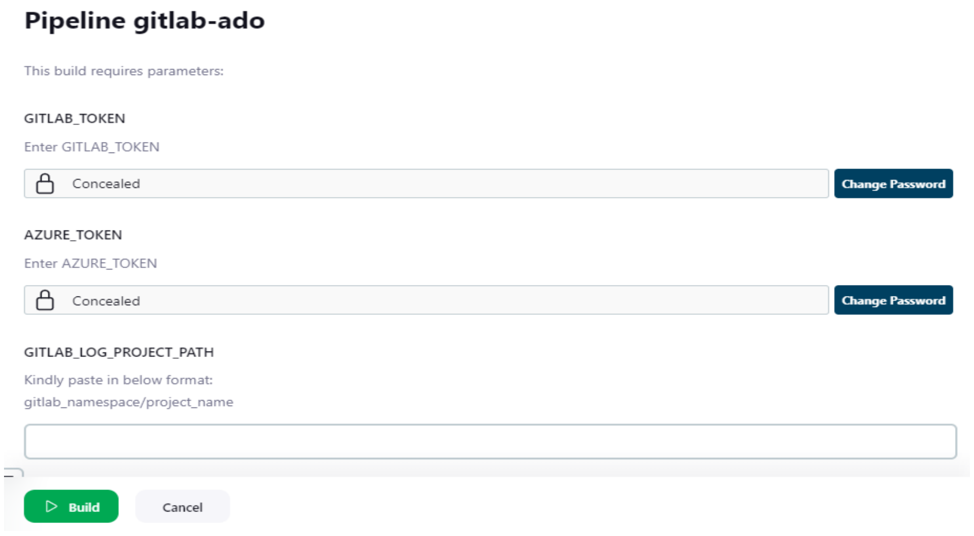


```
+ export gitlab_token=********
+ export azure_token=********
+ export repo_path=vth1/migration-logs
+ python3 gitlab-ado/gitlab-ado.py
```



Parameters
GITLAB_TOKEN
Enter GITLAB_TOKEN
(password value not shown)
AZURE_TOKEN
Enter AZURE_TOKEN
(password value not shown)

3. Parameter View



Pipeline gitlab-ado

This build requires parameters:

GITLAB_TOKEN
Enter GITLAB_TOKEN
Concealed

AZURE_TOKEN
Enter AZURE_TOKEN
Concealed

GITLAB_LOG_PROJECT_PATH
Kindly paste in below format:
gitlab_namespace/project_name