**TFS Git to GitHub Repository Migration Script Documentation**

**Overview**

This PowerShell script automates the migration of repositories from **TFS Git** (Team Foundation Server Git) to **GitHub**. It reads project and repository details from a JSON file, creates corresponding repositories in a GitHub organization, clones TFS Git repositories locally using Git, and pushes them to GitHub.

**Prerequisites**

1. **Personal Access Tokens (PATs)**

You need two PATs:

* TFS PAT: Should have access to the source TFS Git repositories with scope for repository read and project read.
* GitHub PAT: Should have scopes like repo and admin:repo\_hook to create and push to GitHub repositories.

Use environment variables or secret vaults to store tokens securely instead of hardcoding them.

1. **Git Installation**

Ensure Git is installed and accessible from the system’s PATH.

To verify run below command in a terminal

git –version

1. **JSON File**

A JSON configuration file is required, e.g., projectRepoDetailsTFS2017.json, containing the project and repository mapping details such as:

[

{

"projectName": "InventoryApp",

"repoName": "InventoryGitRepo"

}

]

This file is used to determine which TFS Git repositories to clone and migrate.

**Script Workflow**

**1. Configuration**

In this step, we need to set up key variables required for the script to function:

* **TFS PAT ($tfsPat)**: Which is used to authenticate API and Git operations against your TFS 2017+ server.
* **GitHub PAT ($githubPat)**: Which is used to authenticate with GitHub for repository creation and pushing code.
* **$ProjectListJsonPath**: Path to a JSON file that contains a list of TFS Git repositories and project names you plan to migrate.
* **$downloadedReposFolder**: A local folder where the script will store cloned TFS repositories before pushing them to GitHub.

This configuration section should be updated before each run to reflect your target environment and input files.

**2. Read and Parse JSON**

The script reads the JSON file defined in $ProjectListJsonPath and converts it into a PowerShell object:

* $projectData = Get-Content $ProjectListJsonPath | ConvertFrom-Json
* This loads the list of repositories into memory for processing.

**3. Environment Setup**

To authenticate with TFS for Git clone operations, the script encodes the TFS PAT:

* Sets the base64-encoded TFS PAT for the Authorization header.
* Ensure the $downloadedReposFolder exists or creates it.

**4. Process Projects and Repositories**

For each entry in the JSON file, the tfsgit-to-github.ps1 script:

* Constructs the TFS Git clone URL.
* Sanitizes the repository name if it contains spaces or special characters to make it GitHub-friendly (e.g., replacing spaces with hyphens).
* Defines a folder path for where the local repository will be cloned.

This setup allows each repository to be processed independently and ensures consistency in folder structure.

**5. Create GitHub Repository**

Before pushing any code, the script creates a corresponding GitHub repository using GitHub’s REST API which includes

* The new repository name.
* Visibility settings (Internal).
* Optional parameters like description and auto-initialization.

**6. Clone TFS Git Repository**

The script uses git clone --mirror to create a full mirror clone of the TFS Git repository:

* git clone --mirror $sourceUrl $repoFolderPath

Key benefits of using --mirror:

* Clones **all refs**, including branches, tags, and remotes.
* Preserves the complete Git history and structure.

This step downloads the entire repository into a local folder where it can be reconfigured and pushed to GitHub

**7. Push to GitHub**

Once cloned, the script sets the new GitHub remote:

* git remote set-url origin https://github.com/org/repo.git

Then push everything:

* git push --mirror

This ensures that:

* All branches and tags are preserved.
* The GitHub repo becomes an exact replica of the TFS Git repo.

**8. Cleanup**

Once the migration is successful, the script may optionally clean up by deleting the local repository folder to save disk space. Alternatively, you can keep the folder for audit, backup, or retry purposes.