Confluence to Azure DevOps Wikis migrator

Product overview

Say hello to the **Confluence to Azure DevOps Wikis migrator** from **Solidify**! With our powerful migration tool, you can easily transfer all of your Confluence pages to Azure DevOps Wikis with just a few easy steps.

Our migrator tool will preserve all of your Confluence content and formatting, and transfer it seamlessly to Azure DevOps Wikis. You will save time, streamline your documentation process, and eliminate the headache of managing multiple platforms. Plus, with our easy-to-use interface, you'll be up and running in minutes.

Don't let managing your documentation hold you back. Move your Confluence pages to Azure DevOps Wikis experience the power of seamless documentation management.

Supported data

Confluence Pages will be migrated to Azure DevOps Wikis in the popular **markdown** format, including the follow data:

- Unlimited amount of Confluence spaces
- Content tree
- Text formatting
    - bold, italic, underline, strikethrough, etc...
    - superscript, subscript
    - Colored text
    - Quotes
    - Lists
    - Action points
    - Code blocks
    - Text alignment (left/right/center)
    - Text indentation
- Images/Attachments
- Tables
- Internal page links
- Links to ADO Work Items (provided that the Jira Migrator has already been run)
- Nested data such as lists inside tables.
- Archived pages
- Page labels
- Comments
- The status macro
- Etc (all basic markdown elements)

Unsupported data

The following data is unsupported as of today:

- Attachments larger than 25MB (see https://learn.microsoft.com/en-us/azure/devops/user-guide/service-limits?view=azure-devops#wiki (https://learn.microsoft.com/en-us/azure/devops/user-guide/service-limits?view=azure-devops#wiki))
- Third party apps and integrations
- Macros
- Layouts (columns and sidebars)
- Page versioning
- Whiteboards
- Template library
- Space and Page permissions
- Page insights, analytics
- Team calendars
- Automations
- Admin controls
- Release tracks
- Security & Compliance
    - Atlassian Access (SSO, SCIM, Active Directory Sync)
    - Audit logs
    - IP allowlisting

Process overview

Here is an overview of the Confluence to Azure DevOps Wiki migration process:

1. **Confluence Export:**
    - Export content from Confluence using Confluence's export features.
    - Ensure the exported data includes pages, attachments, and any other relevant content.
2. **Transformation (Run confluence-ado-migrator.exe):**
    - Run a transformation program (confluence-ado-migrator.exe) on the exported Confluence data.
    - The program converts Confluence-specific HTML content to Markdown.
3. **Azure DevOps Wiki Import (Git Push):**
    - Initialize a local Git repository to store the transformed content.
    - Add the transformed content to the local repository.
    - Commit the changes and push the repository to the Azure DevOps Wiki Git repository.
4. **Verification and Cleanup:**
    - Verify the migrated content in Azure DevOps Wiki to ensure formatting, links, and attachments are correctly imported.
    - Address any discrepancies or issues identified during verification.
5. **Testing and User Validation:**
    - Conduct testing to ensure the migrated content meets expectations and is accessible in Azure DevOps Wiki.
    - Inform and involve relevant users or stakeholders for validation.
    - Provide training or documentation for users transitioning from Confluence to Azure DevOps Wiki.
6. **Cleanup:**
    - (Optional) Use a regex search-replace feature in any modern IDE to scrub the new Wiki repository from unwanted content and formatting.
7. **Finalization:**
    - Once validated, consider finalizing the migration by decommissioning the Confluence instance or adjusting access permissions.
    - Update documentation and communication channels to reflect the new location of documentation in Azure DevOps Wiki.

Argument explanations

Argument	Description	Example
--url	The URL of the Confluence site.	--url "https://example.atlassian.net (https://example.atlassian.net)"
--space	The key of the Confluence space to export. The space key can be found in the URL of the space, for example `https://xyz.atlassian.net/wiki/spaces/<SPACE KEY>/overview`	--space SOLIDIFY
--username	The username or email address used for authentication.	--username "john.doe@solidify.se (mailto:john.doe@solidify.se)"
--password	The API key or password for authentication.	--password "abc123"
--api-version	The Confluence API version to use. (cloud/server)	--api-version "cloud"
--src-path	The path to the local export folder where Confluence data was exported.	--src-path "C:\path\to\local\export\folder\SOLIDIFY"
--out-path	The path to the target folder where the exported data will be stored.	--out-path "C:\path\to\target\folder"
--license-file-path	The path to the license file.	--license-file-path "C:\Temp\license.json"
--wiki-name	Confluence wiki name (as seen in the top-left in the Confluence GUI)	--wiki-name "Solidify Wiki"
--ado-url	ADO URL, including organization/collection.	--ado-url "https://dev.azure.com/solidifydemo (https://dev.azure.com/solidifydemo)"
--ado-project	ADO project name.	--ado-project "ContosoAir"
--ado-pat	ADO personal access token.	--ado-pat "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
--ado-jira-		--ado-jira-id-field

id-field	JiraID field in ADO.	"Custom.JiraReferenceID"
--jira-url	URL to your jira instance.	--jira-url "https://solidifydemo.atlassian.net (https://solidifydemo.atlassian.net)"
--ado-wiki-name	ADO wiki GUID	--ado-wiki-name "ContosoAir.wiki"
--include-page-labels	Should page labels be migrated? (y/n)	--include-page-labels "y"
--pure-markdown	Only write pure markdown elements and no html? Excluding html elements will disable special formatted element such as colored text and content inside table cells. The resulting page will be more human-readable though. (y/n)	--pure-markdown "n"
--log-level	Log level (info/debug/warning/error/critical)	--log-level "info"

Requirements

Hardware requirements/VM Specs

The VM should run on the Standard_B2s Azure service plan (minimum), or equivalent hardware specs:

- Operating System: Windows-based 64bit
    - 2 vCPUs
    - 4 GiB Memory
    - 8 GiB Swap space
    - 128 GiB Storage (SSD/HDD) (or as needed)
    - A stable internet connection

Software requirements

The following software should be installed on the machine:

- Git client (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git (https://git-scm.com/book/en/v2/Getting-Started-Installing-Git))
    - `core.longpaths` enabled in git config (`git config --system core.longpaths true`)
    - system longpaths registry key set (https://learn.microsoft.com/en-gb/windows/win32/fileio/maximum-file-path-limitation?tabs=registry#enable-long-paths-in-windows-10-version-1607-and-later (https://learn.microsoft.com/en-gb/windows/win32/fileio/maximum-file-path-limitation?tabs=registry#enable-long-paths-in-windows-10-version-1607-and-later))
    - Git LFS (optional if you want to bring big attachments into Git LFS) (https://git-lfs.com/ (https://git-lfs.com/))
- Node 15+

Network requirements

The migration occurs in two steps, first exporting from Confluence, then importing to Azure DevOps.

- During the **Confluence export** and **transformation** steps, the machine will need a stable internet connection and access to the **Confluence server** or **Confluence cloud**.
- During the **ADO import** (git push) step, the machine will need a stable internet connection and access to the **ADO Server** or **ADO Services**.

You do not need access to ADO during the export, and similarly you do not need access to Confluence during the import.

If you do not have an internet connection during the export, the tool will not be able to run.

Required permissions

The user running the migration must have:

- **space admin** permission in the Confluence space
- **Contributor** access in the target ADO project

Benchmark

Previous tests have show that a **1 GB Confluence** space takes **less than 20 minutes** to migrate, provided that the recommended system requirements are met, and that all the necessary configuration is in place and that the html exporter has already been run.

[Optional] Requirements for fixing links to ADO Work Items

In order to enable the automatic link mapping functionality for Jira Issues to ADO Work Items, there are a few prerequisite steps:

- Ensure that you have set up a custom text field in Azure DevOps for holding the reference to the original issue key, on each relevant Work Item Type, for example **Custom.JiraReferenceID**. This field can be hidden or removed later.
- Ensure that you have mapped the issue key field in your Jira AzureDevOps Migrator **config.json** file, for example:

```
{
  "source": "key",
  "target": "Custom.JiraReferenceID"
}
```

- Run the Jira AzureDevOps migration
- When you run the Confluence migrator CLI (confluence-ado-migrator.exe), supply the parameter `--ado-jira-id-field Custom.JiraReferenceID`
- You are now set up for using the link mapping functionality!

[Optional] Consolidating confluence space into an existing ADO Wiki hierarchy

The tool can be used to migrate multiple Confluence spaces to a single target, or to a specific existing page hierarchy within an ADO Wiki.

After running the **Confluence to Azure DevOps Wikis migrator** locally to generate the wiki data, you will need to do the following:

- create a new folder (for example: `folderA`) in your wiki inside **Azure DevOps Wikis** (not locally, as this will not set up the required indexes)
- Do a `git export` to get your ADO Wiki files locally
- Paste the files generated by the **Confluence Azure DevOps Wiki migrator** inside `folderA`
- Do a `git commit` and `git push`
- Can it migrate multiple Confluence spaces into a single ADO Wiki?

Step-by-step

Here's a step-by-step guide for using the Confluence to Azure DevOps Wikis migrator:

1. Install node dependencies with the following script: `npm install commander jsdom turndown turndown-plugin-confluence-to-gfm`
2. Export pages from Confluence to a local folder:
    - Log in to your Confluence account and navigate to the space containing the pages you want to migrate.
    - Click on the "Space tools" menu and select "Content Tools."
    - Click on "Export" and select "Export to HTML" from the dropdown menu.
    - Choose the pages you want to export and click "Next."
    - Select "Normal Export" and click "Next."
    - Choose a destination folder on your local machine and click "Export."
    - Wait for the export process to complete.
3. Run our CLI tool to convert the local Confluence resources to an Azure DevOps Wikis-compliant format:
    - Open a Powershell terminal or command prompt and navigate to the directory where the CLI tool is installed.
    - Run the command:

```
.\confluence-ado-migrator.exe `
--url "https://xxx.atlassian.net" `
--space "SOLIDIFY" `
--username "john.doe@solidify.se" `
--password "<confluence api key>" `
--api-version "cloud" `
--src-path "C:\path\to\local\export\folder" `
--out-path "C:\path\to\target\folder" `
--license-file-path "C:\Temp\license.json" `
--wiki-name "Solidify Wiki" `
--ado-url "https://dev.azure.com/solidifydemo" `
--ado-project "ContosoAir" `
--ado-pat "<ADO personal access token>" `
--ado-jira-id-field "Custom.JiraID" `
--jira-url "https://solidifydemo.atlassian.net" `
--ado-wiki-name "ContosoAir.wiki" `
--include-page-labels "y" `
--pure-markdown "n"
```

where `C:\path\to\local\export\folder` is the path to the folder where you exported your Confluence pages and `C:\path\to\target\folder` is the path to the folder where you want to store the converted pages.

- Copy all contents of the folder `devops-migration-confluence\extra-icons` to `$out-path\.images\icons`. The resulting folder structure should look like the following:
    - `$out-path\.images\icons`
        - `\emoticons`
        - `\priorities`

4. Run the following **powershell script** To find all attachments larger than **25 MB**.

```
$filesize=1024*1024*25 # 25GB
$targetPath="C:\path\to\target\folder"
Get-ChildItem $targetPath -Recurse | Where-Object { $_.Length -gt $filesize} | ForEach-Object { Write-Host -NoNewline (Get-Item $_); Write-Host -NoNewline (", size (MB): "); Write-Host($_.Length/1MB) }
```

    - Delete all files listed by the script.

5. Alternative 1: Import to Azure Devops Wiki (as project wiki)
    - In your ADO project, navigate to your project wiki.
    - Create a new blank page if the project wiki is not already populated.
    - Go to `More actions` > `Clone wiki` and copy the wiki clone url.
    - On your local machine, open a terminal or command prompt and navigate to the target folder where your markdown files were generated
    - Run the command `git init` to initialize a new Git repository in the folder.
    - Add everything and commit: `git add .`, `git commit -m "Init"`
    - Add the wiki url that you copied before as a new remote, for example: `git remote add origin https://dev.azure.com/solidifydemo/ContosoAir/_git/ContosoAir.wiki`
    - Push everything with `git push -u origin wikiMaster --all --force` (Warning, this will irreversibly overwrite your current Project Wiki).

6. Alternative 2: Import to Azure Devops Wiki (as code wiki)
    - In your ADO project, create a new git repository where your wiki will reside.
    - On your local machine, open a terminal or command prompt and navigate to the target folder where your markdown files were generated
    - Run the command `git init` to initialize a new Git repository in the folder.
    - Add everything and commit: `git add .`, `git commit -m "Init"`
    - In the terminal or command prompt, run the command `git remote add origin https://dev.azure.com/{organization}/{project}/_git/{wiki name}` where `{organization}` is the name of your Azure DevOps organization.
    - Finally, run the command `git push -u origin master` to push the pages to the new Azure DevOps wiki.
    - In Azure Devops Wikis, create a new Wiki with the "Publish code as wiki". Select the new repo and use "/" as the folder.

Congratulations! You've successfully migrated your Confluence pages to Azure DevOps Wikis using our migrator tool.

Post migration cleanup, how-to with Visual Studio Code

Visual Studio Code (VS Code) provides powerful tools for performing project-wide search and replace operations using regular expressions (regex). This allows you to efficiently find and modify text patterns across multiple files in your project. In this guide, we will the project-wide search and replace operation in VS Code to scrub the new Wiki repository from unwanted content and formatting.

Below is a documentation on performing a project-wide search and replace with regex in Visual Studio Code, starting with opening the `--out-path` folder:

Performing Project-Wide Search and Replace with Regex in Visual Studio Code

Overview

Visual Studio Code (VS Code) provides powerful tools for performing project-wide search and replace operations using regular expressions (regex). This allows you to efficiently find and modify text patterns across multiple files in your project. In this guide, we'll walk through the process of performing a project-wide search and replace operation with regex in VS Code.

Opening the `--out-path` Folder in Visual Studio Code

1. **Open VS Code**:
    - Launch Visual Studio Code on your computer.
2. **Open Folder**:
    - Use the "File" menu or the keyboard shortcut `Ctrl + K` followed by `Ctrl + O` to open the folder containing your project.
3. **Navigate to `--out-path` Folder**:
    - In the VS Code file explorer, navigate to the directory specified by the `--out-path` option. You can do this by clicking on folders in the file explorer until you reach the desired folder.

Performing Project-Wide Search and Replace with Regex

1. **Open Search Panel**:
    - Use the keyboard shortcut `Ctrl + Shift + F` to open the search panel in VS Code.
2. **Enable Regex Search**:
    - Click on the icon with `.*` in the search panel to enable regex search mode. This allows you to use regular expressions in your search queries.
3. **Enter Search Query**:
    - Enter your regex search query in the search input field. You can use regex patterns to match specific text patterns in your project files.
4. **Perform Search**:
    - Press `Enter` to perform the search. VS Code will display a list of search results in the search panel.
5. **Review Search Results**:
    - Review the search results to ensure that they match the text patterns you want to replace.
6. **Perform Replace**:
    - Once you're satisfied with the search results, you can proceed with the replace operation. Click on the "Replace" button next to the search input field to open the replace input field.
7. **Enter Replace Pattern**:
    - Enter your regex replace pattern in the replace input field. You can use capture groups (`$1`, `$2`, etc.) to reference matched groups from your regex search query.
8. **Perform Replace**:
    - Click on the "Replace All" button to perform the replace operation. VS Code will replace all occurrences of the matched text patterns with the specified replace pattern in your project files.

Using regex101.com to Build and Test Regular Expressions

1. **Open regex101.com**:

- Open your web browser and navigate to regex101.com (https://regex101.com/).

2. **Build and Test Regular Expressions**:
    - Use the regex101.com interface to build and test your regular expressions. You can enter sample text and see how your regex pattern matches against it. The website also provides explanations and highlights matched text for easier debugging.
3. **Refine and Test**:
    - Refine your regex pattern as needed based on the test results. Make adjustments until you achieve the desired matching behavior.

Solution for Self service-based migrations

Solidify offers a self-service solution for the Confluence to ADO Wiki Migrator, which enables organizations to automate the migration process of many Confluence spaces, thus saving time and effort.

The self-service solution is powered by Azure Pipelines, and the intention is to set up a Build Agent with the neccessary tooling. The entrypoint for the end users will be an **Azure Pipeline** with some amount of configurability.

The Self Service solution is part of our Professional Services offering. Contact us at support.jira-migrator@solidify.dev (mailto:support.jira-migrator@solidify.dev) for more information.