**Newt Global Consulting, LLC**

1255 Corporate Dr | Suite # 200 | Irving TX 75038

Phone: +1 972 887 3165 | Fax: +1 214 260 6051

**Table of Contents**

[1. Introduction 1](#_Toc470035073)

[2. Prerequisites 1](#_Toc464906263)

[3. Creating an API Token in Jira 1](#_Toc67956288)

[4. Performing Discovery 1](#_Toc130768186)

[5. Migration Process 1](#_Toc805763987)

[6. Post-Migration Tasks 1](#_Toc1550187963)

[7. Reconciliation: 1](#_Toc1730445244)

[7. Limitations 1](#_Toc1119504483)

# JIRA TO AZURE DEVOPS MIGRATION

# 1. Introduction

Migrating from Jira to Azure DevOps can streamline your development workflow, improve collaboration, and centralize your project management. This documentation provides a comprehensive guide to help you migrate your data from Jira to Azure DevOps seamlessly.

# 2. Prerequisites

Before starting the migration process, ensure you have the following:

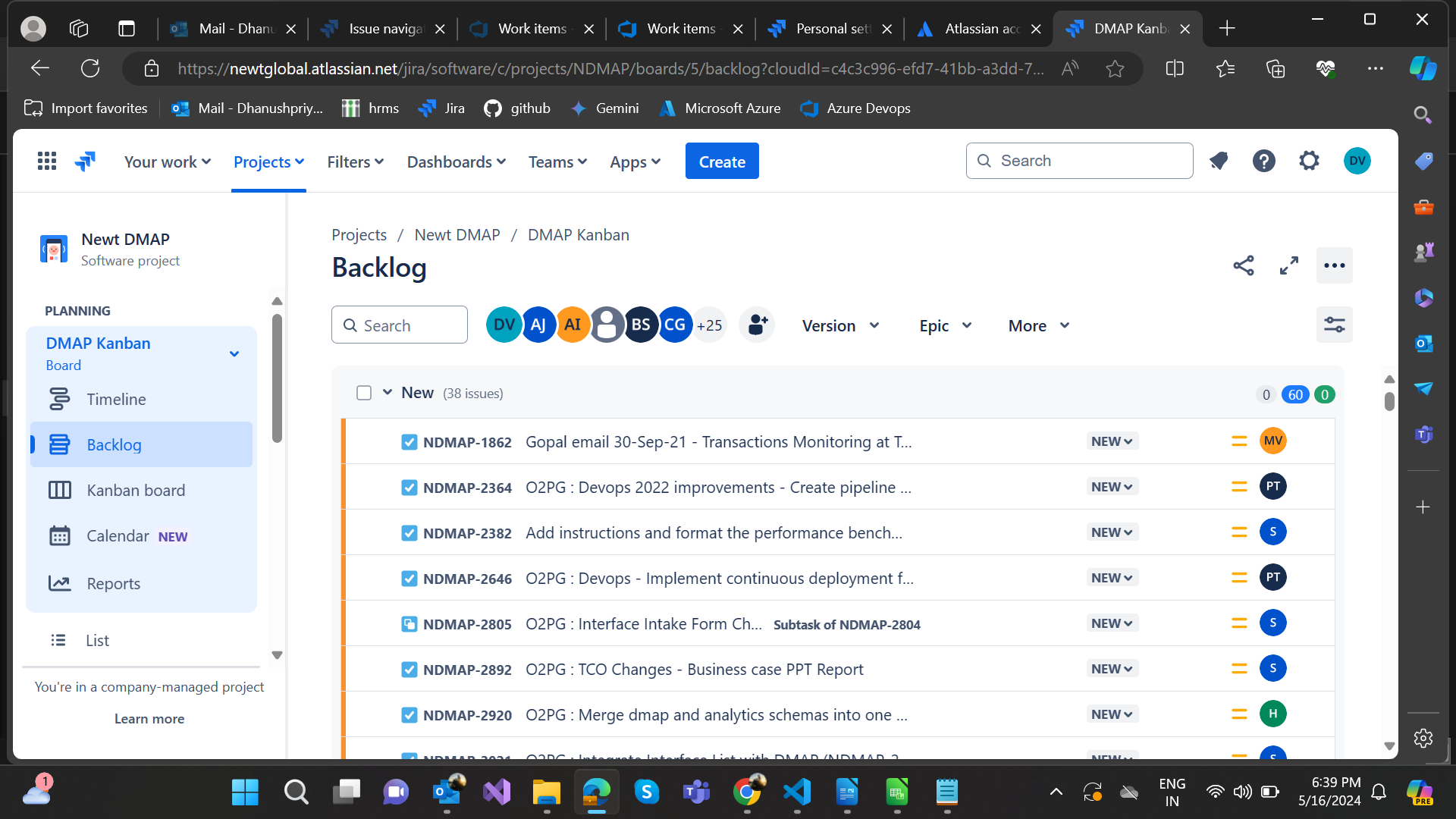
* Administrative access to both Jira and Azure DevOps.
* Basic understanding of Jira and Azure DevOps terminologies and workflows.
* Access to the Jira API and Azure DevOps API.
* Excel or similar spreadsheet software to manage project details.
* A migration tool that supports Jira to Azure DevOps migration

# 3. Creating an API Token in Jira

To migrate data from Jira, you need to create an API token. Follow these steps to create one:

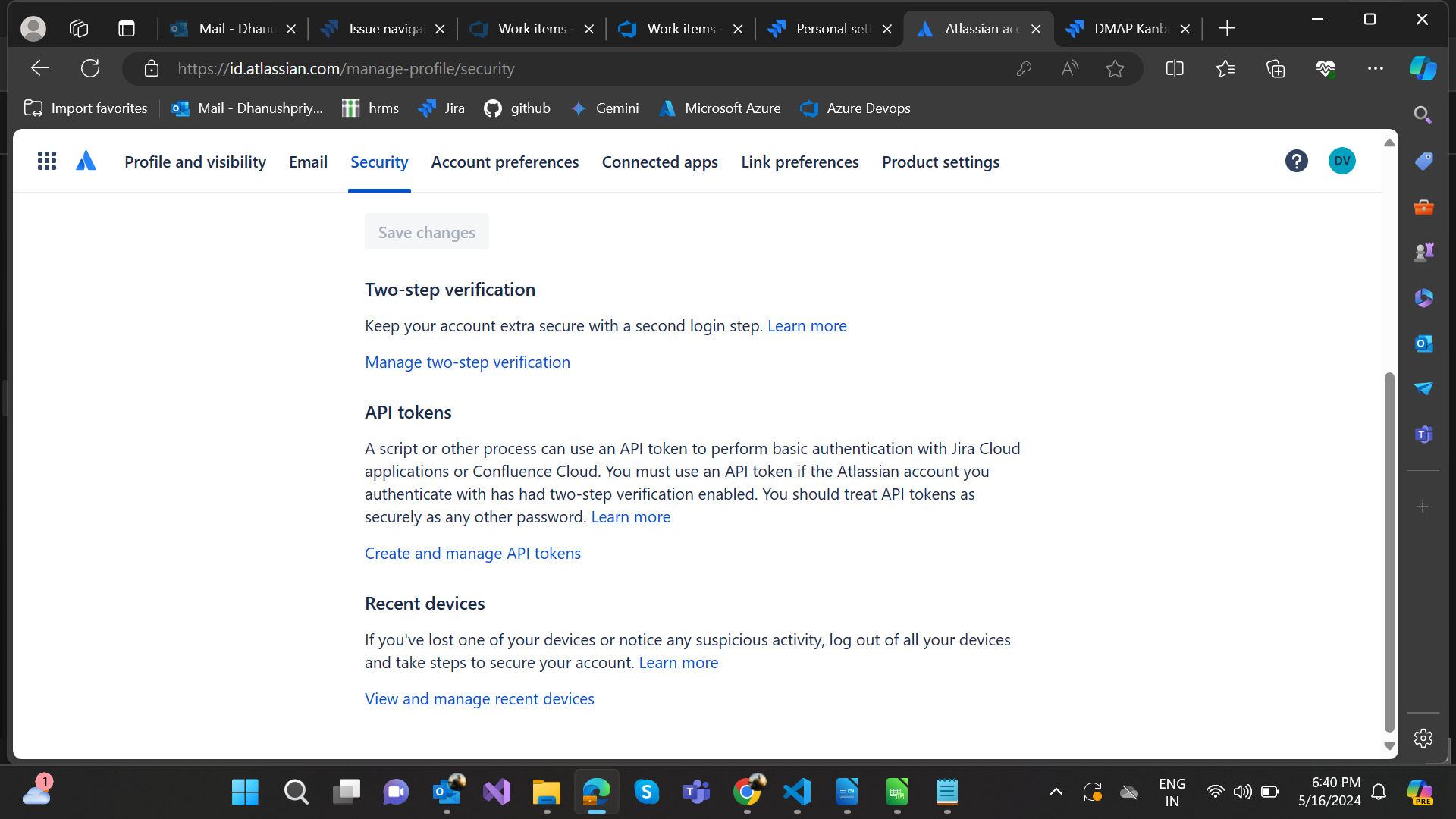
Log in to Jira:

* Go to your Jira account.



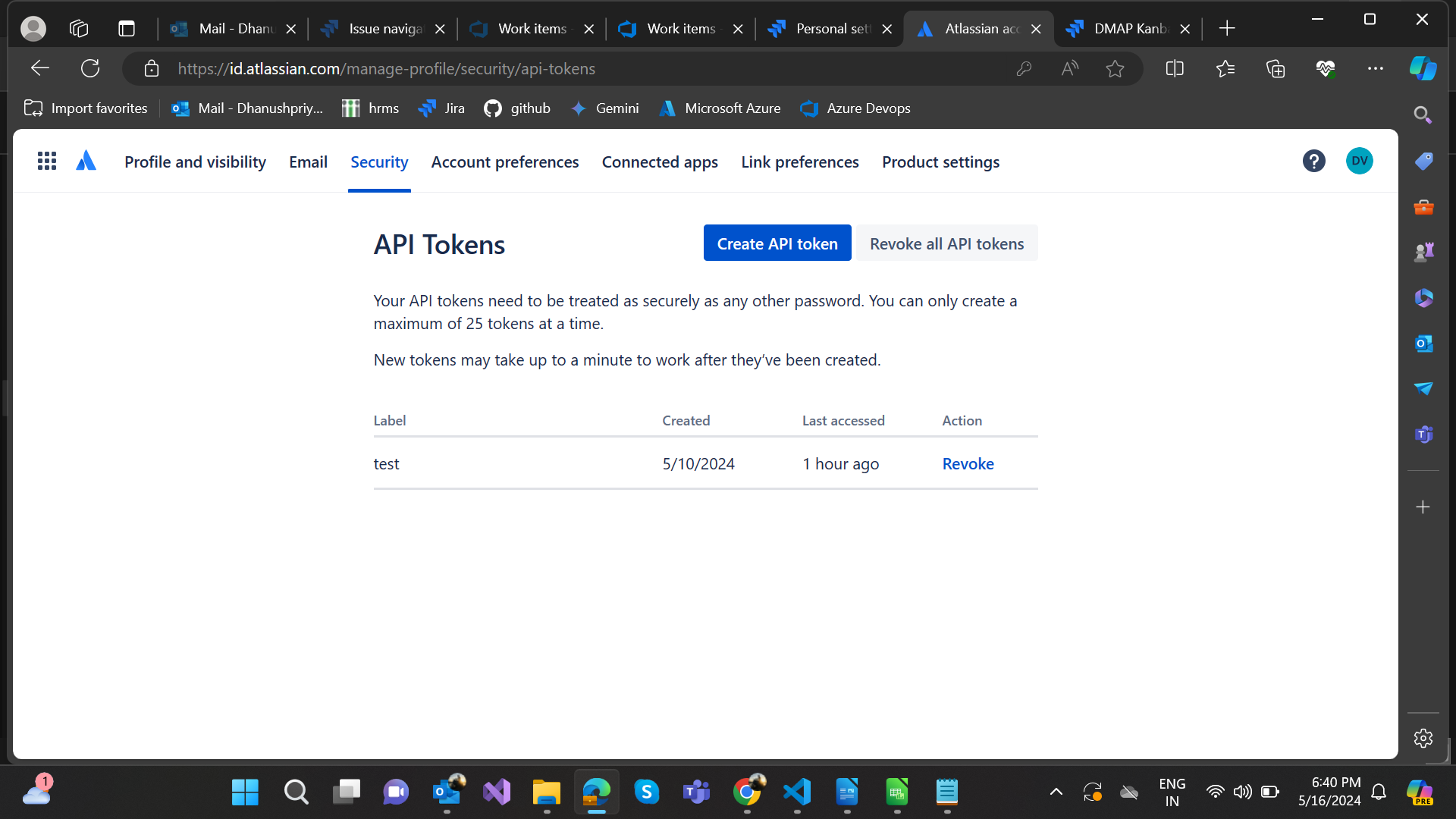
Access Account Settings:

* Click on your profile icon and select "Account settings."



Create API Token:

* Navigate to the "Security" section.
* Click on "Create and manage API tokens."
* Click the "Create API token" button.
* Give your token a label (e.g., "Azure DevOps Migration").
* Click "Create."
* Store it securely as you will need it for the migration process.



# 4. Performing Discovery

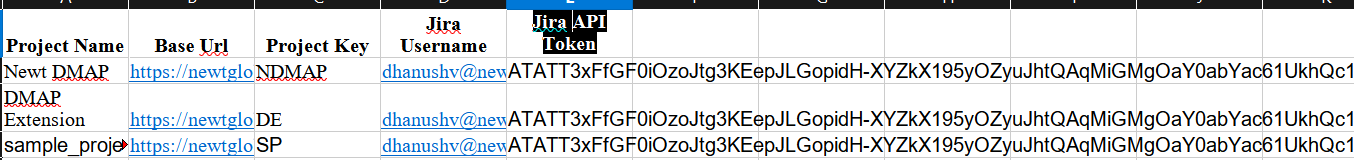
Setting Up the Discovery Process:

Discovery involves gathering all relevant project details from Jira. This helps in planning the migration effectively.

Preparing the Project Details:

1. Create an Excel Sheet:

* Prepare an Excel sheet to list all projects you want to migrate.
* Include columns such as Project Name, Project Key, Base URL Jira Username and Jira API Token.



2. Fill in Project Details:

* Populate the Excel sheet with details of each project from Jira.

Running the Discovery Script:

1. Prepare the Python Script:

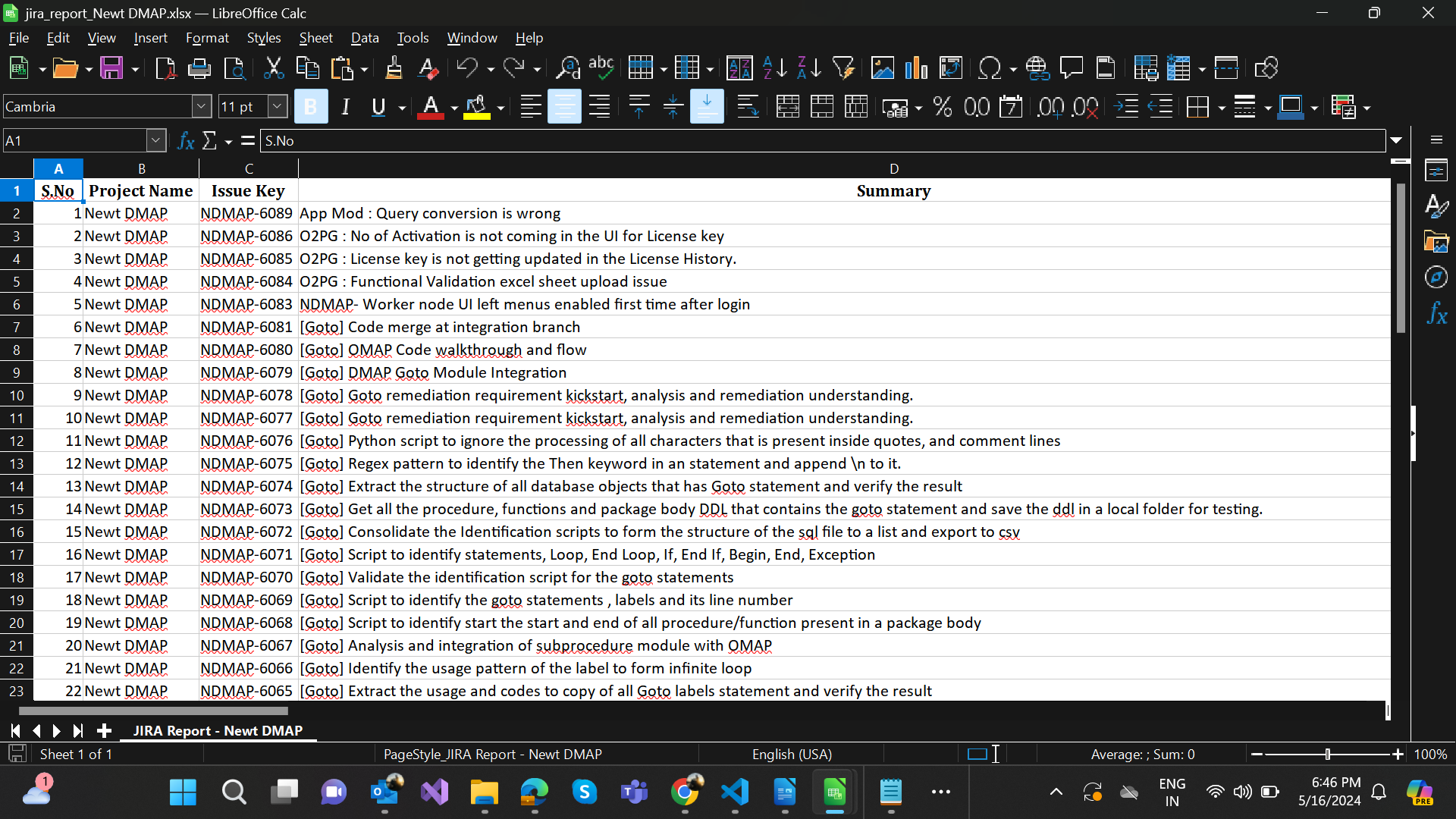
* Ensure you have a Python script capable of extracting data from Jira. The script should use the Jira REST API to fetch project details.

1. Run the Script:

* Execute the script using the API token created earlier.
* The script will gather details such as key, summary, description, assignee, reporter, time Zone, time estimate, time spent, due date, and created date for each issue.

1. Generate the Report:

* The script will output these details into an Excel sheet.



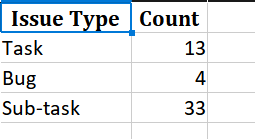
Analyzing the Discovery Report:

Review the Report:

* Check the generated Excel sheet for completeness.
* Ensure it contains all required fields: key, summary, description, assignee, reporter, time zone, time estimate, time spent, due date, and created date.

Validate the Data:

* Compare the data with Jira to ensure accuracy.
* Note any discrepancies and correct them.



# 5. Migration Process

Understand Entity Mapping:

* Map Jira entities (e.g., Issues, Epics, Tasks) to corresponding Azure DevOps entities (e.g., Work Items, Features).

Jira to Azure DevOps Work Item Migration Tool:

Download the Tool:

* Obtain the Jira to Azure DevOps Work Item Migration Tool from the below link.
* Download the latest version

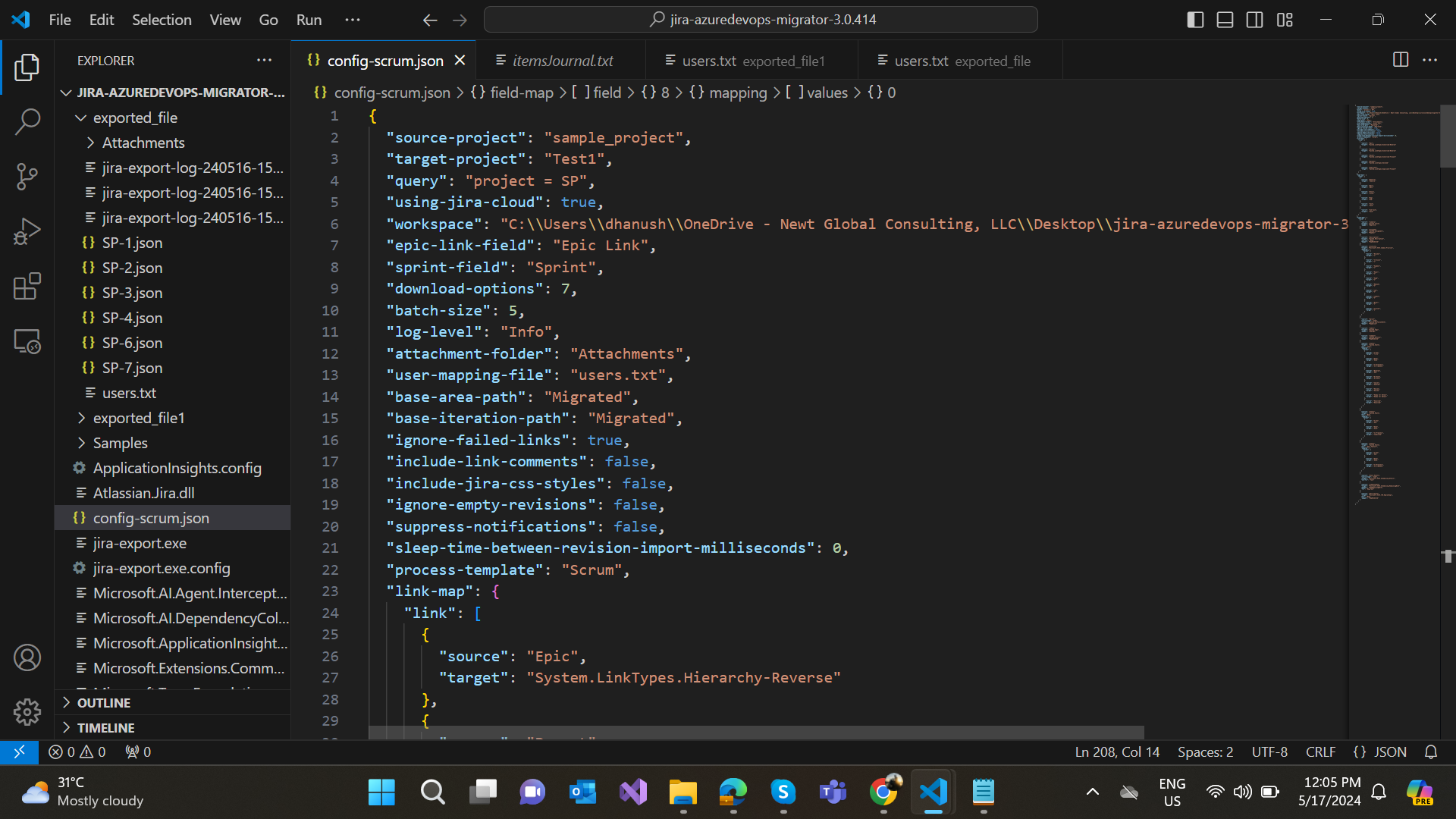
<https://github.com/solidify/jira-azuredevops-migrator/releases>

Demo Document: <https://solidify.dev/blog/jira-azure-devops-migration>



Edit the Config File:

* Since we want to migrate our Jira sample project to a Test project in Azure DevOps, we're going to use the sample configuration file that is based on the Scrum template in Azure DevOps.
* The sample configuration file is placed in the folder Samples and it's called config-scrum.json .
* This configuration says that we’re going to export all items in the Jira project with short name "Sample\_Project" and migrate them to an Azure DevOps project called "Test".
* The configuration has also mapped standard Jira items to their respective Azure DevOps work item types based on the Scrum template.



Edit the Users File:

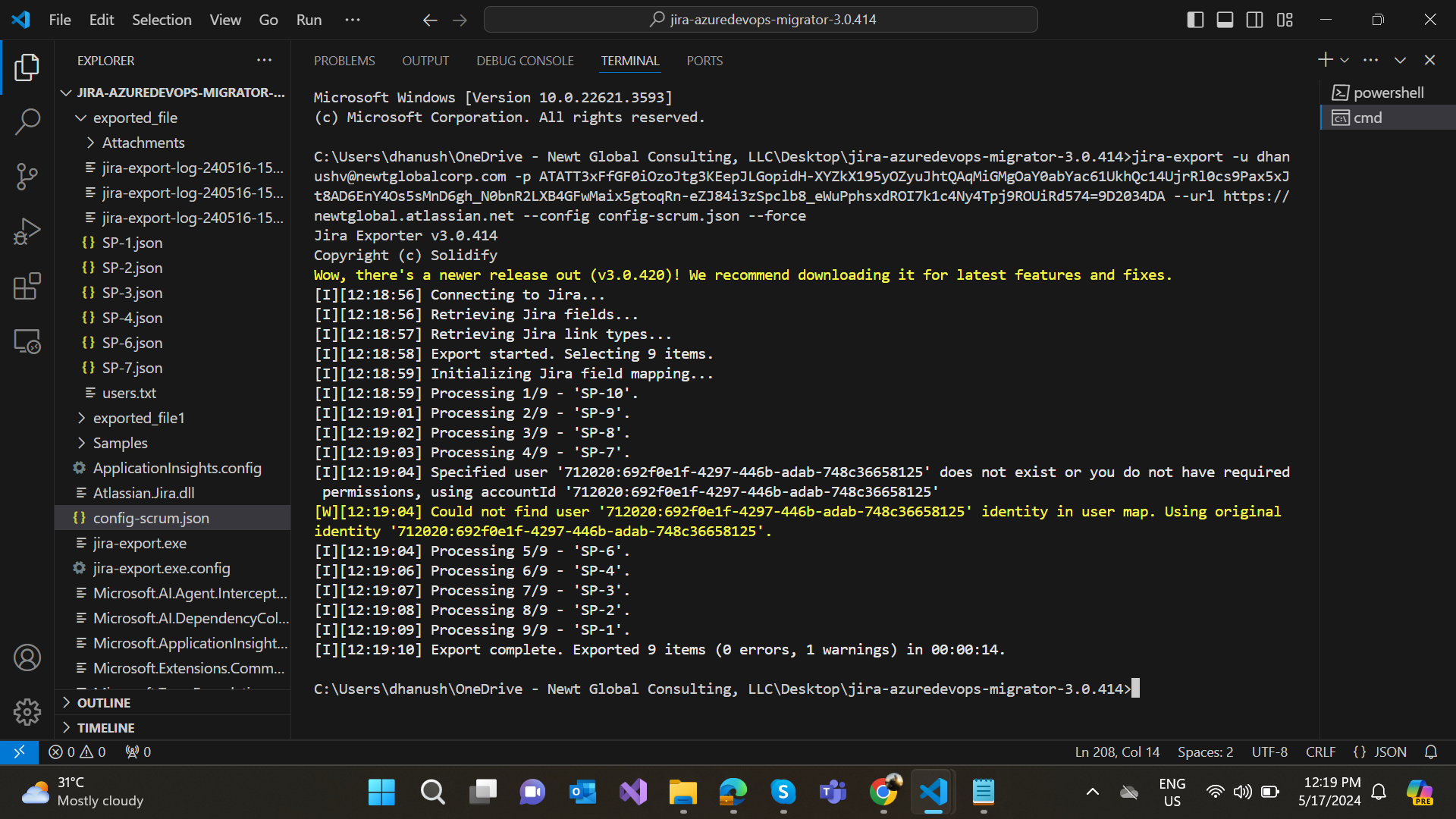
* To migrate users and assign the new work items in Azure DevOps to the same user as the original task had in Jira, we need to add a text file in the root that would look something like this:
* [Some.JiraUser@domain.com=Some.AzureDevOpsUser@domain.com](mailto:Some.JiraUser@domain.com=Some.AzureDevOpsUser@domain.com)

Export Json files from Jira:

* Open a cli of your choice and navigate to where the extracted zip file exists. In my case I'll open a Command Prompt and write: cd C:\Temp\jira-azuredevops-migrator-2.0.2
* Make sure your updated config-scrum.json and users.txt are in the root
* Collect the access keys you need for Jira:
* Identify the migration account (username and password) to access Jira
* Get the URL to Jira and the name of the source project
* Run the export tool by typing the following command:

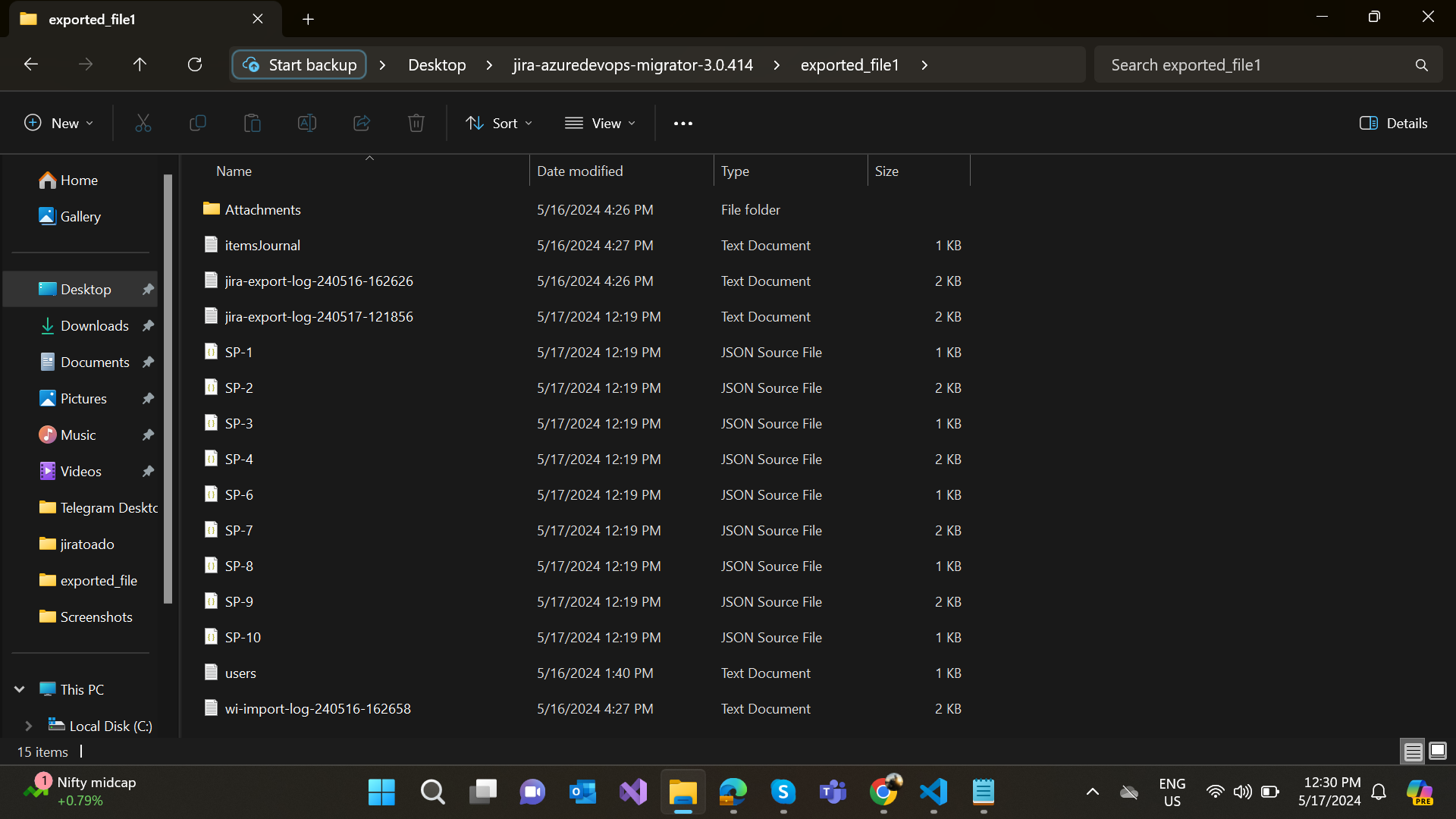
Eg:

jira-export -u jiraaccount@some.domain -p Jira-API-token --url https://my.jira.url --config config-scrum.json --force

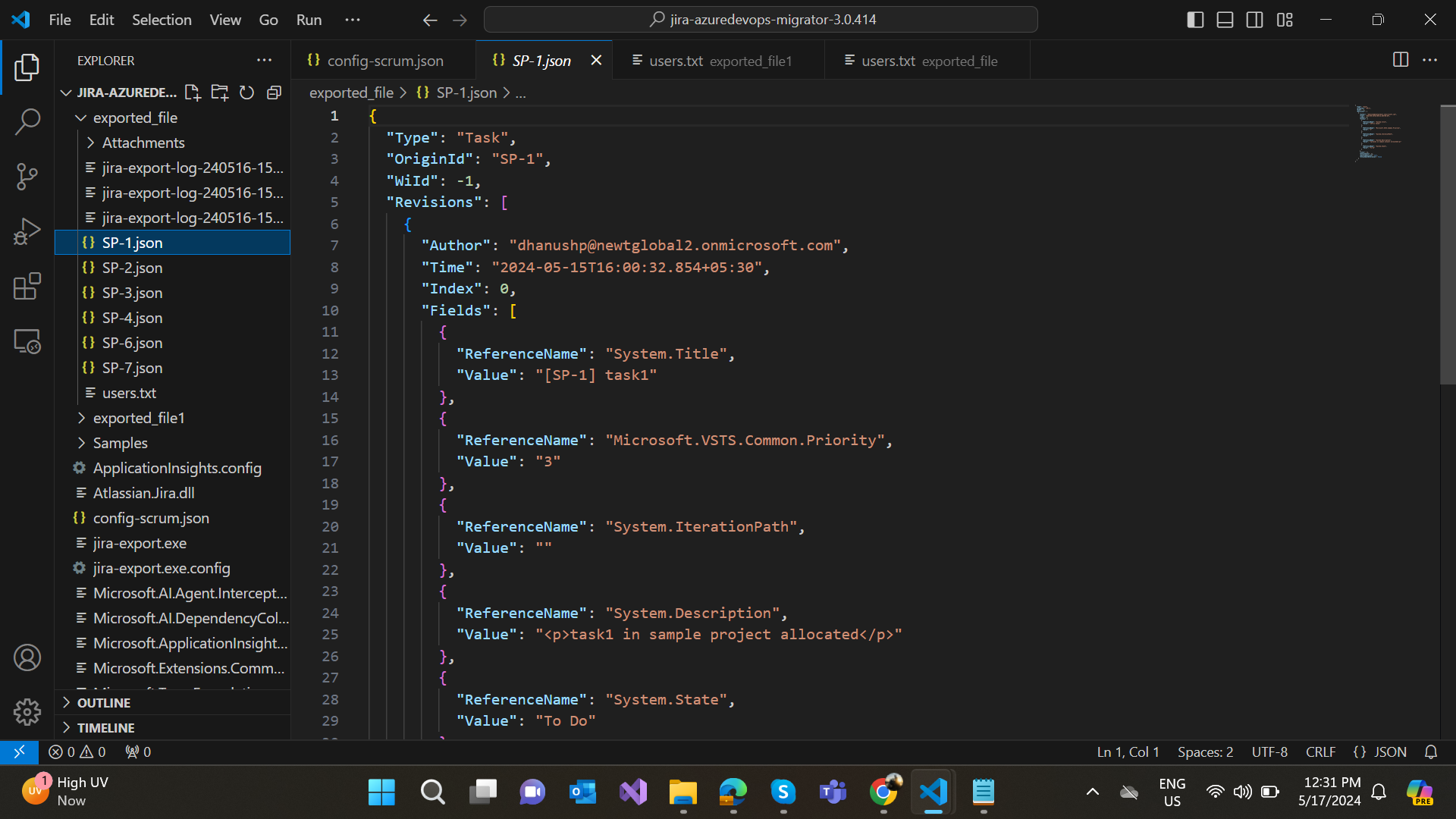


Exporting from Jira:

* Review the export logfile (created for reference and troubleshooting)
* Done. The exported items are now in the workspace folder specified in the configuration and it looks like this:



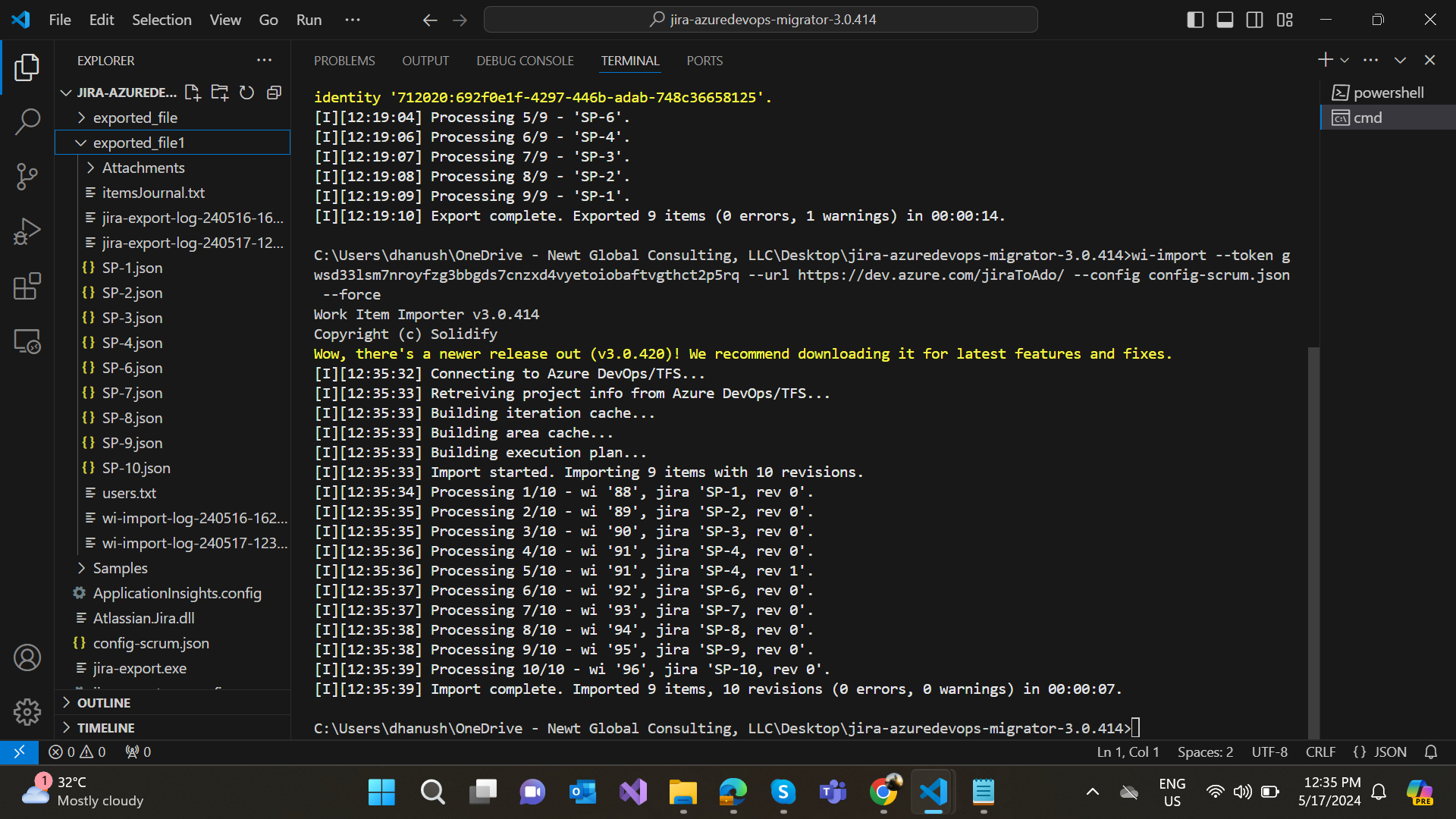
Exported Json file:



Import to Azure DevOps:

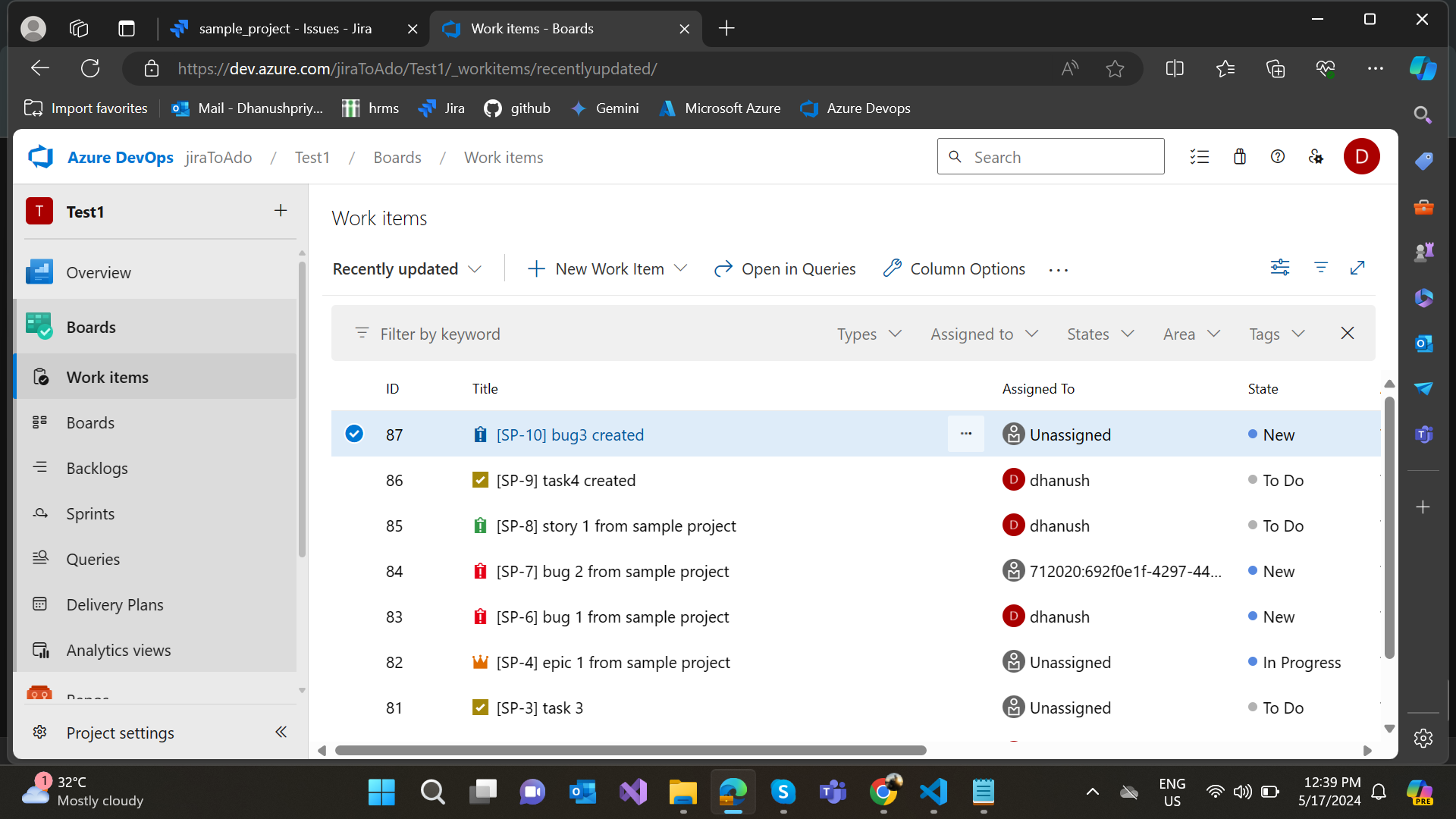
* + Collect the access keys you need for Azure DevOps
  + Get the URL and name of the target Azure DevOps organization
  + Acquire a personal access token for the organization here with at least work item read/write permissions
  + Get the name of the target project
  + Run the import tool by typing the following command:

wi-import --token myAccessToken --url https://my.azuredevops.url --config config-scrum.json --force



Post-Migration Verification:

* Check that all data has been migrated accurately.
* Compare the number of tasks, issues, etc., in Azure DevOps with the discovery report.



# 6. Post-Migration Tasks

## Discovering Users:

* After migrating the work items, it is essential to discover and verify all users associated with the projects to ensure they have been correctly mapped and included in the new system.

1. Run User Discovery:

◦ Fetch all users from Jira and gather details such as project, display name, email, and account ID.

◦ Store the user details in a structured format (e.g., JSON or CSV file).

2. Validate Users:

◦ Review the collected user data for completeness and accuracy.

◦ Ensure that all users are listed with their corresponding project, display name, email, and account ID.

## Performing Discovery on Azure DevOps:

* To ensure that the migration was successful, perform a similar discovery on Azure DevOps to fetch all work items and their details, and then compare them with the Jira data.

1. Fetch Work Items from Azure DevOps:

◦ Retrieve all work items from the Azure DevOps project, including details such as name, description, assignee, created date, and state.

◦ Store the work item details in a structured format (e.g., JSON or CSV file).

2. Generate Statistics:

◦ Compile statistics on the work items from both Jira and Azure DevOps.

◦ Include metrics such as the total number of work items, types of work items (e.g., tasks, bugs), and status distributions.

# 7. Reconciliation:

* Reconciliation is the process of verifying that the migration was successful by comparing the data in Jira and Azure DevOps.

1. Store Data in a Database:

◦ Import the fetched data from Jira and Azure DevOps into a relational database for comparison.

◦ Create tables for Jira users, Jira work items, and Azure DevOps work items.

2. Compare Work Items:

◦ Execute queries to compare the number of work items in Jira and Azure DevOps.

◦ Check that each work item from Jira exists in Azure DevOps with matching details (e.g., name, description, assignee).

3. Generate Reconciliation Report:

◦ Create a report that includes a summary of the comparison.

◦ Add a column to indicate whether each work item matched between Jira and Azure DevOps.

◦ Include additional columns for any discrepancies found during the comparison.

4. Review and Validate:

◦ Examine the reconciliation report to ensure that all work items have been accurately migrated.

◦ Investigate and resolve any discrepancies.

# 7. Limitations

* While the migration process is designed to be comprehensive, there are some limitations:

1. API Rate Limits:

◦ Jira API limits may restrict the number of issues fetched in a single request. Typically, only 50 issues can be fetched per request using a single API token.

◦ You may need to paginate requests to handle larger datasets.

2. Data Fidelity:

◦ Certain custom fields or plugins in Jira may not have direct equivalents in Azure DevOps, requiring manual adjustment or custom scripts to handle such cases.

3. User Mapping:

◦ Users must be manually mapped between Jira and Azure DevOps. Ensure that usernames and email addresses are correctly aligned to avoid assignment issues.

4. Work Item Relationships:

◦ Complex relationships between work items, such as linked issues or dependencies, might not be fully supported by all migration tools and may require additional handling.