Configuring Azure DevOps Integration for World Papers

Introduction

Azure DevOps is a powerful platform for managing software development, enabling seamless integration with GitHub for tracking work items, automating tasks, and setting up CI/CD pipelines. In this blog post, we will walk through the **complete setup process** for integrating Azure DevOps with GitHub, troubleshooting common issues, and ensuring a smooth workflow.

Step 1: Authenticate with Azure DevOps CLI

Before we begin configuring Azure DevOps, we need to authenticate with the **Azure CLI** to gain access to our projects.

Authenticate via Device Code (Recommended)

If you don't have an active Azure subscription, use:

```
az login --use-device-code
```

Steps: 1. Open **Microsoft Device Login** in your browser. 2. Enter the code provided in the terminal. 3. Complete authentication with your Microsoft account.

Allow Tenant-Level Access (If No Subscriptions Found)

If the above command fails with **"No subscriptions found"**, use:

```
az login --allow-no-subscriptions --use-device-code
```

This grants access to DevOps without requiring an Azure subscription.

Verify Authentication

Once logged in, check your account details:

```
az account show
```

If this returns **your Azure account information**, authentication was successful.

Step 2: Set Up Azure DevOps Organization & Project Defaults

To integrate with GitHub, we need to configure our **organization and project defaults**.

Retrieve Your Azure DevOps Organizations

Azure DevOps organizations are not listed by default in the CLI. Run this **REST API request** to find yours:

```
az rest --method get --uri "https://app.vssps.visualstudio.com/_apis/accounts?api-version=6.
```

Look for the `"accountName"` field, which represents your organization name.

Set Your Organization as Default

Once you have the correct organization, set it in the CLI:

```
az devops configure --defaults organization=https://dev.azure.com/YOUR_ORG_NAME
```

Verify the configuration:

```
az devops configure --list
```

List All Projects in Your Organization

Now, retrieve all projects within your DevOps organization:

```
az devops project list --output table
```

If you don't see your project, check Step 3: Troubleshooting.

Set Your Project as Default

Once you find the correct project name, set it:

```
az devops configure --defaults project="World Papers"
```

Now, Azure DevOps CLI will automatically use this project for all commands.

Step 3: Troubleshooting Common Issues

1. Project Not Found (TF200016)

If you receive the error:

```
TF200016: The following project does not exist: World Papers.
```

Fix: - Run:
`bash az devops project list --output table` and **use the exact**

project name. - Ensure you're in the **correct organization** by running:
`bash  az devops configure --list`

2. Work Item Linking Policy Fails

If applying a **work-item linking policy** fails with:

`--project must be specified.`

Fix: Ensure the project name is explicitly provided:

```
az repos policy work-item-linking create --organization https://dev.azure.com/YOUR_ORG_NAME
```

3. Cannot Retrieve Repository ID

If your script fails to find the repository ID, run:

```
az repos list --organization "https://dev.azure.com/YOUR_ORG_NAME" --project "World Papers"
```

Copy the **repository ID** and manually update your script.

Step 4: Automate Azure DevOps Configuration with a Script

To automate the configuration process, use the following **Bash script**:

```
#!/bin/bash

# Set variables
ORG_URL="https://dev.azure.com/CloudLasso"
PROJECT_NAME="World Papers"
BRANCH="main"
REPO_ID="ee1fa234-65a8-4264-a867-cba56bf30f5d"

# Check if repository ID is set
if [ -z "$REPO_ID" ]; then
    echo "  Error: Repository ID not found. Ensure the repository exists in Azure DevOps."
    exit 1
fi

echo "  Repository ID for world-papers: $REPO_ID"

# Apply work-item linking policy
az repos policy work-item-linking create --organization $ORG_URL --project "$PROJECT_NAME" -

# Verify if policy is applied
echo "  Verifying policy..."
az repos policy list --organization $ORG_URL --project "$PROJECT_NAME" --repository-id $REPO
```

```
echo "  Work item linking policy successfully applied!"
```

Save the script as **configure_devops.sh**, then run:

```
bash configure_devops.sh
```

Conclusion

By following these steps, you have successfully **configured Azure DevOps**, connected it with **GitHub**, and automated the setup with a script. If you encounter further issues, check permissions and ensure your **organization and project names are correct**.

What we accomplished: - Authenticated Azure DevOps CLI. - Configured organization & project defaults. - Troubleshot common DevOps errors. - Automated DevOps setup with a script.

Now, Azure DevOps is ready for managing work items, tracking issues, and automating CI/CD workflows for **World Papers**!