Branching Strategy for Azure DevOps

Contents

Existing Git Deploy Problems

What are the existing problems and how the proposed solution solves them

- Problems with active test code getting published to Test/Production without the developer ready for the promotion
    - Causes
        - The ARM process is taking all pipelines, datasets, global parameters, data flows, etc. into one Json file. This process promotes all new work into the main branch.
        - The ARM Json file is published to Test and Production
    - Solutions
        - Dev ADF will have its collaboration sourced from Dev_Release. Test/Prod will receive changes from ARM template deployment.
        - Developers will merge from their Feature Branches to Dev, Test and Main allowing only their work to move to higher environments
        - Main branch now contains only production code
- Checking into main is required to be able to link ADF pipelines to Control-M
    - Causes
        - ADF only displays objects from the Publish Branch. This branch only merges with the collaboration branch (Main)
    - Solutions
        - Dev, Test and Production publish from different collaboration branches
            - Developers can create Pull Request without being concerned with taking other developers code to higher environments before it is ready to move

Branching Strategy for Azure DevOps

Git Branches

1. Main Branch

- **Purpose**: **Main** branch represents the production-ready code.
- **Access**: Only Pull Request (PRs) approvals from both Team Leads and Release Managers will allow code changes to this branch.

2. Environment Release Branch

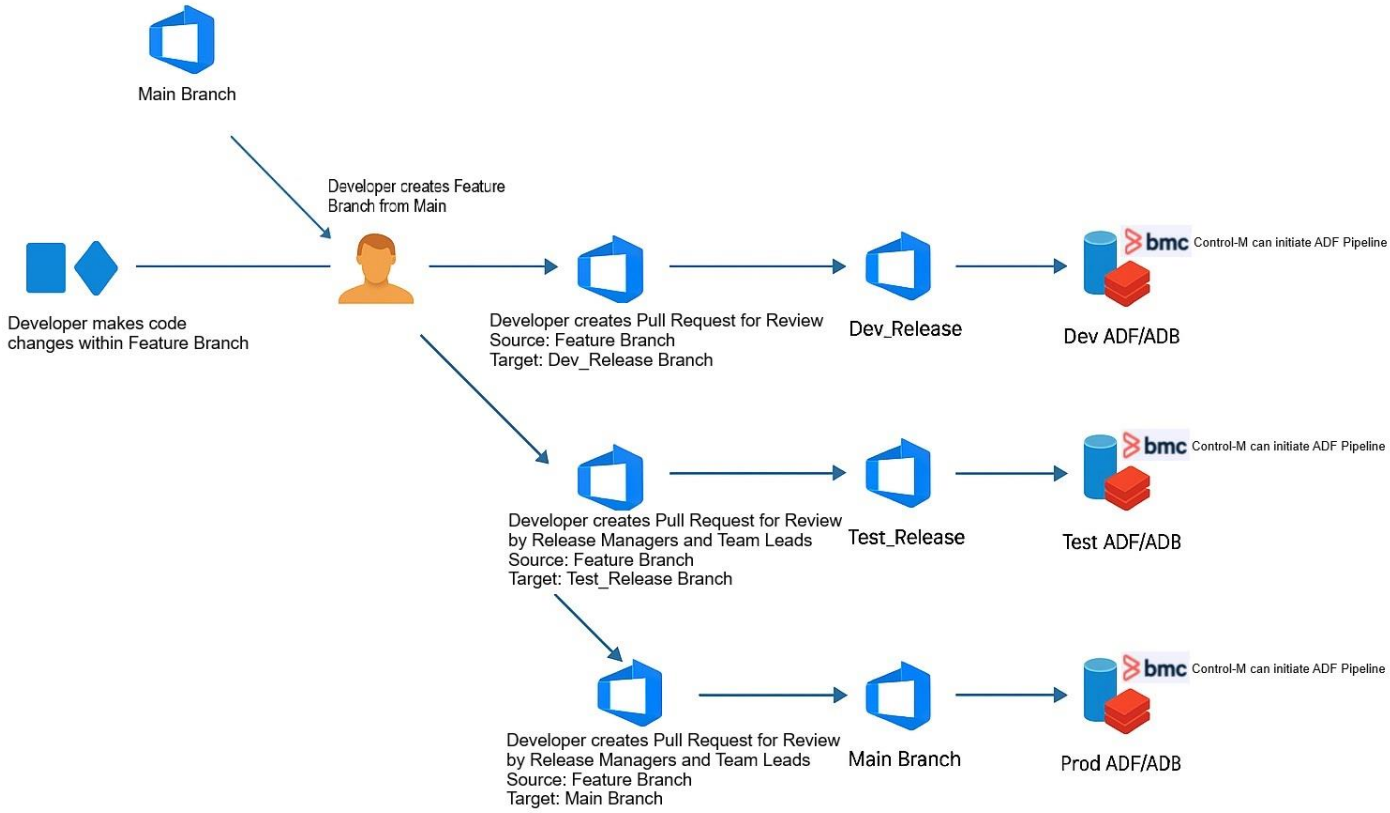
- **Purpose**: Dev/Test branches will be used as collaboration branches. They are the source for adf_publish which displays objects to ADF and ADF web services. ADB will also use Dev/Test Branches as part of the deployment cycle.
- **Access**: Team Leads manage PRs within **Dev_Release** Branch and both Release Managers and Team Leads manage PRs within **Test_Release** Branch.

3. Feature Branches

- **Purpose**: Developers create feature branches from the `main` branch to work on new features or bug fixes.
- **Naming Convention**: Use descriptive names like ***`{feature number/poc/test}_{team abbreviation}_{employee number}_{short_description}`***.

Branching Strategy for Azure DevOps

Workflow



1. **Create Feature Branch**
    - **Action**: A team member creates a feature branch from the Main branch
    - **Note:** A developer at any moment has access to merge Feature Branch changes to Main. This can only be done through a pull request and would require approval.
    - **Responsibility:** Developer is responsible for creating feature branch from Main
2. **Develop Feature**
    - **Action**: The developer works on the feature in the feature branch
    - **Responsibility:** Developer is responsible for completing development
3. **Merge Feature Branch into DEV_Release Branch**
    - **Action**: Create a Pull Request to merge changes from Feature Branch into Dev_Release Branch

- **Responsibility:** Developer is responsible for creating pull requests. Team Lead is responsible for validating requests and approving pull requests. Any issues identified by Team Lead must be resolved by the Developer.
- **Process**: This will trigger a process to publish changes to Dev ADF allowing for control-m jobs to trigger new or changed pipelines.
- **Approval**: Approver confirms commits are related to change, Dev_Release Branch is properly selected, and code is making the proper changes
- **Defect Found**: Changes are made in Feature Branch and a new Pull Request is created to merge into Dev_Release Branch

4. **Merge Feature Branch into Test_Release Branch**
    - **Action**: Create a Pull Request to merge changes from Feature Branch into Test_Release Branch
    - **Responsibility:** Developer is responsible for creating pull requests. Release Managers and Team Leads are responsible for validating requests and approving pull requests. Any issues identified by Release Managers or Team Leads must be resolved by the Developer.
    - **Process:** This will trigger a process to publish changes to Test ADF allowing for control-m jobs to trigger new or changed pipelines.
    - **Approval**: Approver confirms commits are related to change, Test_Release Branch is properly selected, and code is making the proper changes
    - **Defect Found**: Changes are made in Feature Branch and a new Pull Request is created to merge into Dev_Release Branch
5. **Merge Feature Branch into Main Branch**
    - **Action**: Create a Pull Request to merge changes from Feature Branch into Main Branch
    - **Responsibility:** Developer is responsible for creating pull requests. Release Managers and Team Leads are responsible for validating requests and approving pull requests. Any issues identified by Release Managers or Team Leads must be resolved by the Developer.
    - **Process:** This will trigger a process to publish changes to Prod ADF allowing for control-m jobs to trigger new or changed pipelines.
    - **Approval**: Approver confirms commits are related to change, Main Branch is properly selected, and code is making the proper changes
    - **Defect Found**: Changes are made in Feature Branch and a new Pull Request is created to merge into Dev_Release Branch

Branching Strategy for Azure DevOps

Create Feature Branch within Azure Data Factory

1. **Open Azure Data Factory Studio**
    - Navigate to the Azure portal and open your Azure Data Factory instance.
    - Click on the "Author" tab to access the authoring environment.
2. **Navigate to the Branch Dropdown**
    - In the "Author" tab, locate the branch dropdown at the top of the page.
    - Click on the dropdown to see the list of branches.
3. **Create a New Feature Branch**
    - Click on the "+ New Branch" button in the branch dropdown.
    - Enter a name for your new feature branch (e.g., *{feature number/poc/test}_{team abbreviation}_{employee number}_{short_description})*.
    - Select the main branch as the source branch for your new feature branch.
    - Click "Create" to create the new feature branch.
4. **Switch to the Feature Branch**
    - After creating the feature branch, select it from the branch dropdown to switch to it.
    - You can now make changes and develop new features within this branch.

Create Pull Request from Azure Data Factory

1. **Open Azure Data Factory Studio**
    - Navigate to the Azure portal and open your Azure Data Factory instance.
    - Click on the "Author" tab to access the authoring environment.
2. **Switch to the Feature Branch**
    - In the "Author" tab, locate the branch dropdown at the top of the page.
    - Select the feature branch you want to merge into the {*Dev Environment Branch, Tst Environment Branch or Main*} branch.
3. **Commit Changes**

- Make sure all your changes are committed to the feature branch.
- Click on the "Save" button which will create a commit including all your changes.

4. **Create a Pull Request**
    - Go to the branch dropdown and select "Create pull request."
    - This action will take you to Azure DevOps (*ADO*).
    - In the ADO interface, fill in the details for the pull request:
        - **Title**: *{feature number/poc/test}_{team abbreviation}_{employee number}_{target environment dev environment branch/tst Environment branch/main}{short_description}*.
        - **Description**: Add a detailed description of the changes.
        - **Reviewers**: Add reviewers who will review the pull request.
    - Select the {*Dev Environment Branch, Tst Environment Branch or Main*} branch as the target branch for the pull request.
5. **Review and Approve**
    - Reviewers will receive a notification to review the pull request.
    - They can add comments, request changes, or approve the pull request.
6. **Merge the Pull Request**
    - Once the pull request is approved, you can merge it into the {*Dev Environment Branch, Tst Environment Branch or Main*} branch.
    - Click on the "Complete" button in the ADO interface. Click on Complete merge leaving Post-completion options as default when target is Dev/Test branches. Delete *{branch}* after merging can be selecting if merging to Main.

Steps to Recover a Deleted Branch

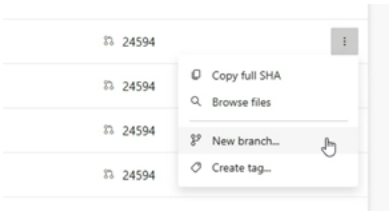
If example branch 550887_Trailblazers_e42554_DeleteBranch_Dev_Environment_Branch was deleted, then here are the steps to instantiate it again.

The commit of the pull request can be found and a new feature branch containing the original feature code can be created.

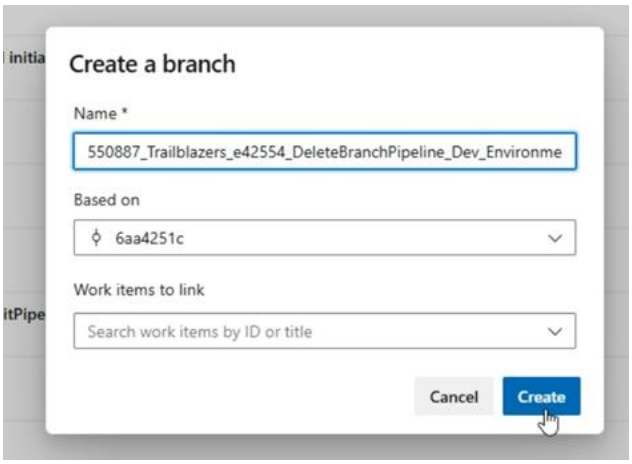
Branching Strategy for Azure DevOps

Go to the commits of the Environment branch that the feature branch was merged into. Then identify the last commit of the pull request, but not the commit merging the feature branch to the Environment branch. It should be a blue dot.

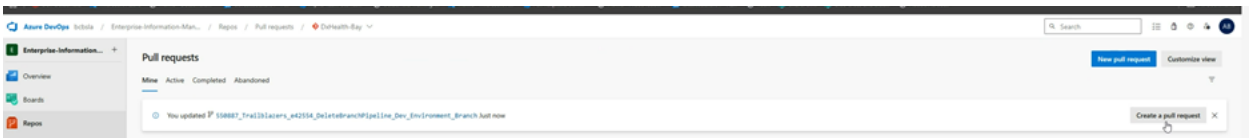
Click on the ellipsis of the commit. Fix image with drawing program.



Click on the “New Branch” drop down item. Rename the new feature branch to the following the pull request using the naming convention.



Now go to the pull requests on the left panel, and create another pull request. This time you will target the higher environment.



Branching Strategy for Azure DevOps