Day 4 notes

Azure Data Factory integrates with Azure Repos (Git) to enable collaborative development. You can manage multiple branches to organize development work effectively.

Default Branches in ADF

- **Collaboration Branch**: The main branch used for deploying and sharing changes. Commonly named main or master.
- **Feature/Development Branches**: Individual branches where developers work on specific features, bugs, or enhancements.

2. Key Concepts in ADF Branching

Collaboration Branch

- The default branch where all approved and tested changes are merged.
- Represents the source of truth for deployments to production.
- Configurable in ADF under "Git configuration."

Feature Branches

- Created to develop features, resolve issues, or experiment with ideas without affecting the collaboration branch.
- Example: feature/add-pipeline, bugfix/fix-dataset-issue.

---

3. Typical Workflow

1. **Create a Feature Branch**:
    - In Azure Repos, create a new branch for your work.
    - Example command: git checkout -b feature/new-feature.
2. **Develop in the Feature Branch**:
    - Use ADF UI to create or edit pipelines, datasets, linked services, or triggers in the feature branch.
    - The changes are saved to the branch in Azure Repos.
3. **Testing**:
    - Debug and test your changes within the feature branch.
    - Use integration testing environments to ensure stability.
4. **Merge Changes**:
    - Create a Pull Request (PR) in Azure Repos to merge changes from the feature branch into the collaboration branch.

    - Review, approve, and resolve conflicts before merging.
5. **Publish to Live**:
    - Once merged, **publish the ADF changes** from the collaboration branch. This will deploy artifacts (pipelines, datasets, triggers) to the Data Factory service.

---

4. Branching Best Practices

1. **Use Meaningful Branch Names**:
    - Follow naming conventions such as feature/, bugfix/, or hotfix/.
    - Example: feature/add-sales-data-pipeline.
2. **Small, Focused Changes**:
    - Keep feature branches focused on specific changes for easier review and conflict resolution.
3. **Avoid Direct Edits in the Collaboration Branch**:
    - Always create feature branches for development to maintain collaboration branch integrity.
4. **Automate Pull Requests**:
    - Set up policies for PRs in Azure Repos to require approvals, code reviews, and successful builds before merging.
5. **Branch Cleanup**:
    - Delete feature branches after merging to keep the repository clean.

---

5. Integration with CI/CD

- Use Azure Pipelines to trigger builds and deployments based on branch policies.
- Example: Deploy changes only from the main branch to production after merging and publishing in ADF.

---

6. Managing Multiple Environments

Branching can be combined with environment-specific configurations:

- Use separate branches for dev, staging, and prod environments.
- Example:
    - dev branch for development and testing.
    - staging branch for pre-production validation.
    - main branch for production deployments.

---

7. Git Integration in ADF

To enable Git in ADF:

1. Go to **Manage > Git Configuration** in the ADF portal.
2. Connect to Azure Repos and configure:
    - Collaboration branch.
    - Root folder for ADF artifacts.
3. Commit and push changes to Azure Repos directly from the ADF UI.