**Drawbacks in the Current Design**

1. **Branch Explosion and Complexity**:
   * Maintaining separate branches (dev, test, stg, prod) increases the overhead of managing and synchronizing changes across environments.
   * Potential risks of branch drift, where code in one branch diverges significantly from others.
2. **Risk of Inconsistent Configurations**:
   * Environment-specific code in branches can lead to inconsistencies between environments.
   * Bugs or behavior discrepancies might surface in production if test/staging environments are not identical.
3. **Manual and Error-Prone Promotions**:
   * Merging changes to the next environment branch requires manual effort or custom scripting, increasing the chances of human error.
   * It is harder to track whether a specific commit has been promoted to production.
4. **Limited Observability and Rollback Options**:
   * Tracking changes across branches makes it harder to observe what version of the DAGs or configurations is running in each environment.
   * Rolling back to a stable state can be error-prone and time-consuming without a proper deployment strategy.
5. **Environment-Specific Logic**:
   * If the repository branches are used to store environment-specific DAGs or configurations, it violates the principle of "immutable infrastructure" and creates fragility in deployments.

**Challenges in the Current Scenario**

1. **Delayed Promotions**: UAT in staging blocks production deployments for other teams.
2. **Environment Contention**: Multiple teams using the same staging environment introduce risks of untested or unvalidated changes affecting each other.
3. **Dependency Coupling**: Teams are dependent on each other's UAT timelines, reducing agility.
4. **Lack of Change Isolation**: No mechanism to isolate team-specific changes during UAT.
5. **Single Deployment Target**: The shared Airflow instance means changes promoted by one team affect all teams.

**Suggested Approach for DAG Promotion Pipeline**

**Overview**

We are adopting a streamlined promotion pipeline where the main branch is used for code promotion, and each environment (Dev, Test, Staging, Prod) has its own configuration file for DAG scope management. The workflow is designed to minimize manual intervention and ensure a smooth, permission-based promotion from one environment to the next.

**Workflow Scenarios**

1. **New DAG with Scope Till Prod**

* **Action**: A completely new DAG is added to the repository, and it’s defined with a scope to run in all environments (Dev, Test, Staging, Prod).
* **Config Update**: The respective environment-specific configuration files (dev, test, stg, prod)

1. **Existing DAG with Scope Till Prod**

* **Action**: An existing DAG is updated and needs to be promoted across all environments.
* **Config Update**: The environment-specific config files are updated to include the DAG name.

1. **Existing DAG in Prod with Scope Till Staging**

* **Action**: If an existing DAG in production needs testing in staging (UAT-like testing), a new version of the DAG is created (e.g., dag\_uat) and configured with the necessary scope (up to Staging).
* **Config Update**:
  + Update the config files for Staging and possibly Test to include dag\_uat.
  + If promotion to Prod is needed:
    - **Option 1**: Merge changes from dag\_uat to the original dag (if changes are small).
    - **Option 2**: Delete the existing dag and rename dag\_uat to dag (if the changes are more substantial).
    - Add the DAG name (dag) to the prod/promote\_dag.config

**Important Note:**

Since there is no dedicated UAT environment in the current setup, Staging is used to simulate UAT. As such, this approach is **logical testing** rather than real user acceptance testing. **The workflow should be flagged for thorough PR review** because it introduces the risk of changes in Staging directly impacting production, especially when dag\_uat is promoted to dag.

**Changes to Azure DevOps Pipeline for New Approach**

**Current Pipeline Setup:**

1. The entire codebase (DAGs, plugins, etc.) from the Airflow repository is bundled and deployed to specific environments (Dev, Test, Staging, Prod).
2. There is a single deployment process that pushes everything to the target environment, without considering which DAGs are specifically configured to run in that environment.

**Updated Pipeline Setup:**

The updated approach introduces environment-specific config files that control which DAGs should be deployed to each environment. Below are the changes to make this process more flexible and streamlined:

**Key Changes in the Pipeline:**

1. **Config-Driven DAG Selection:**
   * The pipeline now uses the environment-specific configuration files (promote\_dag.config) to determine which DAGs to deploy.
2. **Environment-Specific Deployment:**
   * Instead of deploying the entire Airflow repo, only the DAGs listed in the configuration file for the respective environment are copied to the Airflow DAG directory.
3. **Validation & Testing:**
   * Added validation for the presence of DAG files in the config and post-deployment testing to ensure that the DAGs are properly deployed and function in the target environment.