**Databricks Deployment Workflow - Case Study**

**Case Study: Bundle Deployment, Tagging, and Rollback**

**Background**

A data engineering team maintains a Databricks project with multiple environments: dev, test, and prod. The team follows a CI/CD process using bundle.yml to manage deployments. Their goal is to deploy version v1.0 to Test, validate it, and if needed, roll back to version v0.9.

**Scenario**

The team has just completed several new features and bug fixes. After unit and integration testing in dev, they decide to promote the bundle to the test environment for further validation.

**Step 1: Deploy Bundle to Test**

**Action:**

databricks bundle deploy --environment test

**Outcome:**

* Notebooks, jobs, and configs are pushed to the test Databricks workspace.
* Integration tests run automatically.
* Performance metrics and output data are verified by QA engineers.

**Observation:**  
All test cases pass, but performance is slightly degraded in one job.

**Step 2: Tag Release v1.0**

**Action:**

git tag -a v1.0.0 -m "Release v1.0.0 for Test deployment"

git push origin v1.0.0

**Outcome:**

* v1.0.0 tag is created in GitHub.
* CI pipeline triggers artefact packaging for v1.0.0.

**Observation:**  
The tagged version is now a reference point for potential promotion to production.

**Step 3: Rollback to v0.9**

After monitoring v1.0.0 in Test for a week, the performance degradation is deemed unacceptable. The decision is made to roll back to the previous stable version v0.9.0.

**Action:**

git checkout v0.9.0

databricks bundle deploy --environment test

**Outcome:**

* Test environment reverts to the v0.9.0 state.
* Job performance returns to acceptable levels.

**Observation:**  
Rollback process is smooth because of proper tagging and deployment discipline.

**Lessons Learned**

1. **Tagging is critical**: It ensures that exact code states are easily redeployable.
2. **Test environment validation**: Prevents flawed releases from reaching production.
3. **Rollback readiness**: Having a tested rollback process minimizes downtime and performance risks.

**Best Practices Highlighted**

* Always validate in test before production.
* Use semantic versioning for tags (vMAJOR.MINOR.PATCH).
* Maintain artefact retention for all release tags.
* Monitor KPIs during testing to inform go/no-go decisions