DevOps & Releases

Last updated by | Daniels, Steve | Sep 30, 2025 at 8:51 PM GMT+5:30

Contents

Controls

Change Control Process

- IQUW_IT Change Management Policy v3.6.docx

Deployment Pipeline Controls

All deployment pipelines must implement the following controls:

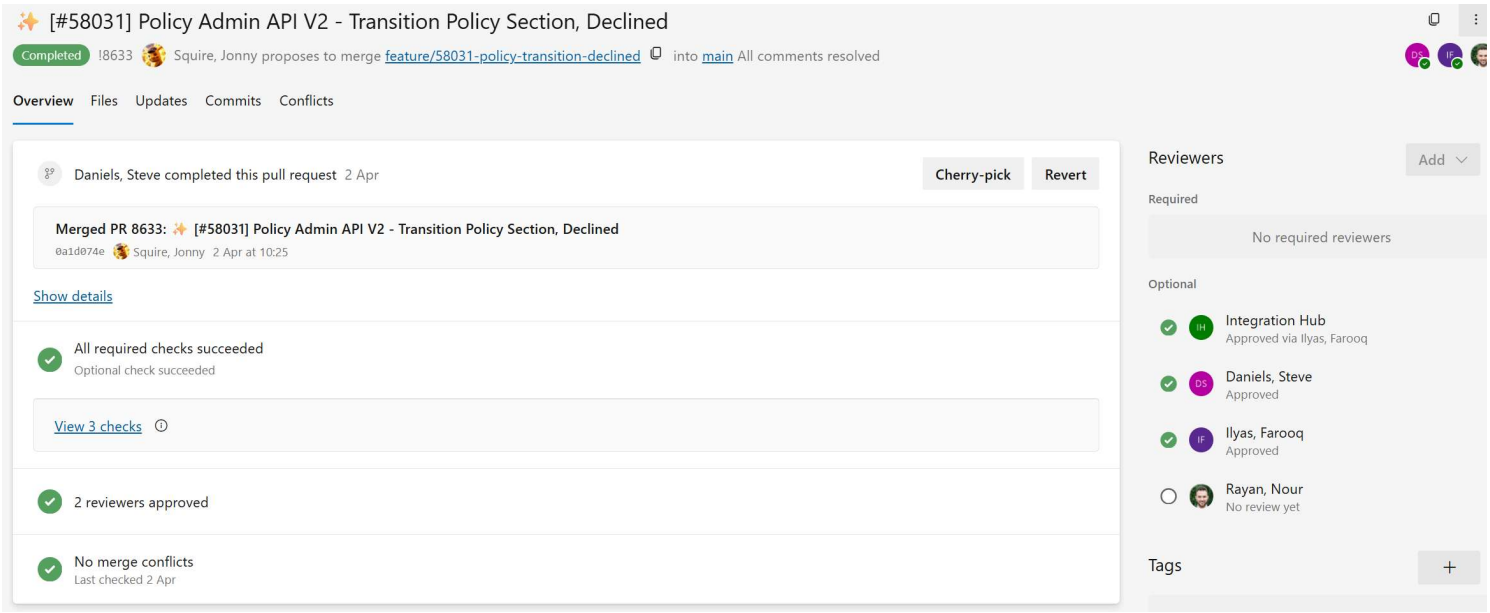
- Prior to deploying to any environment the app must check:
  - Build successfully
  - All Tests pass
  - All Linting pass
  - All Code scanning checks pass
- Deployment to non-production environments must be approved by at least one engineer

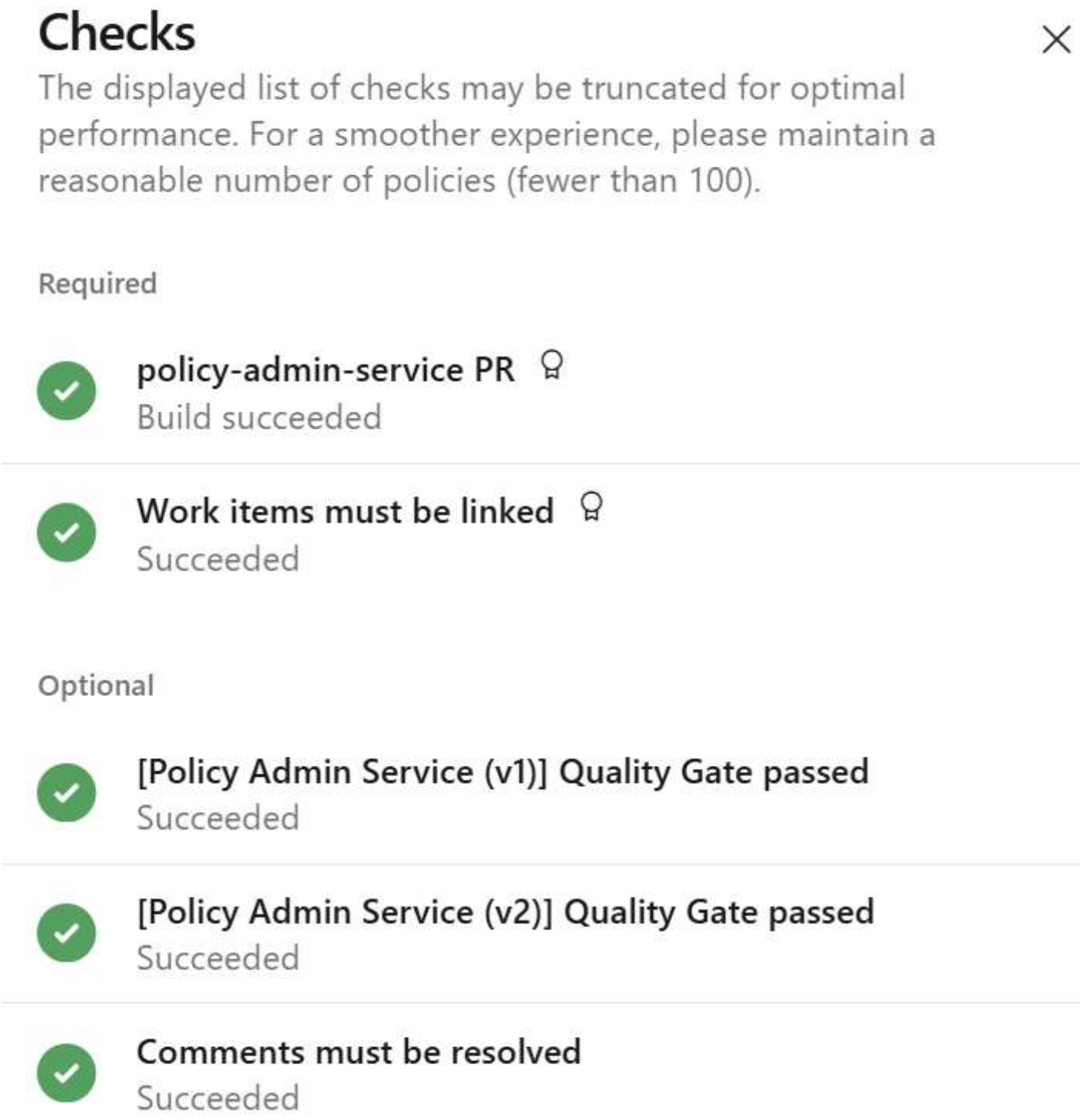
- Deployment to pre-production/production environments cannot be approved by the user who initiated the deployment
- Deployments to production require cross team approval (i.e. Tester + Developer + Platform) with at least 1 approval from a minimum of 2 teams

Pull Request Controls

All code repositories must implement the following controls

- All changes to any `main` branch must be made via a Pull Request
- Pull Request cannot be merged unless peer reviewed by at least one other engineer
- Pull Requests cannot be merged until all Security & Testing checks pass
- Pull Requests cannot be merged if any outstanding comments or discussions made on a given Pull Request have been resolved
- Pull Requests must have a work item linked
- All changes must meet the relevant Coding Standards





Pipelines

We use Azure DevOps pipelines to handle releases and other automated tasks such as PR requests or health checks

Vulnerability Scanning (Dependabot)

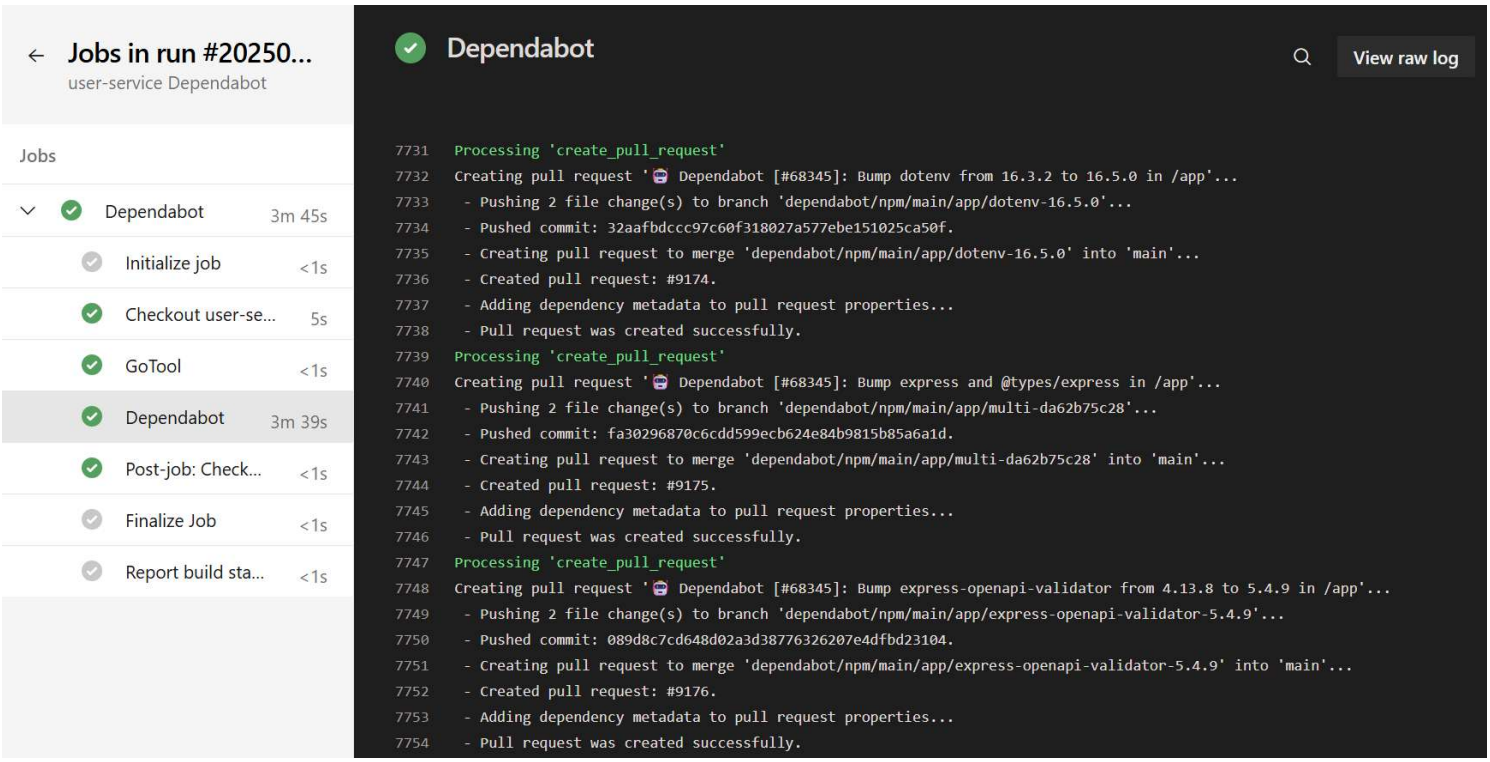
See: Dependabot Pipelines

Work items:

- 48313 **Vulnerabilities scanning (iHub)** | Done
- 70762 **Vulnerabilities Scanning (Python)** | Done
- 67525 **Vulnerabilities Scanning (Azure Functions)** | Done
- 70763 **Vulnerabilities Scanning (React)** | Done
- 81543 **Vulnerability Scanning (Dataiku)** | New

We scan our repos for vulnerabilities using Dependabot periodically.

PR are created automatically by the tool with updated packages which are trigger Quality Checks and are then peer-reviewed by Developers.



Quality Checks

See: Quality Check Pipelines

Work items:

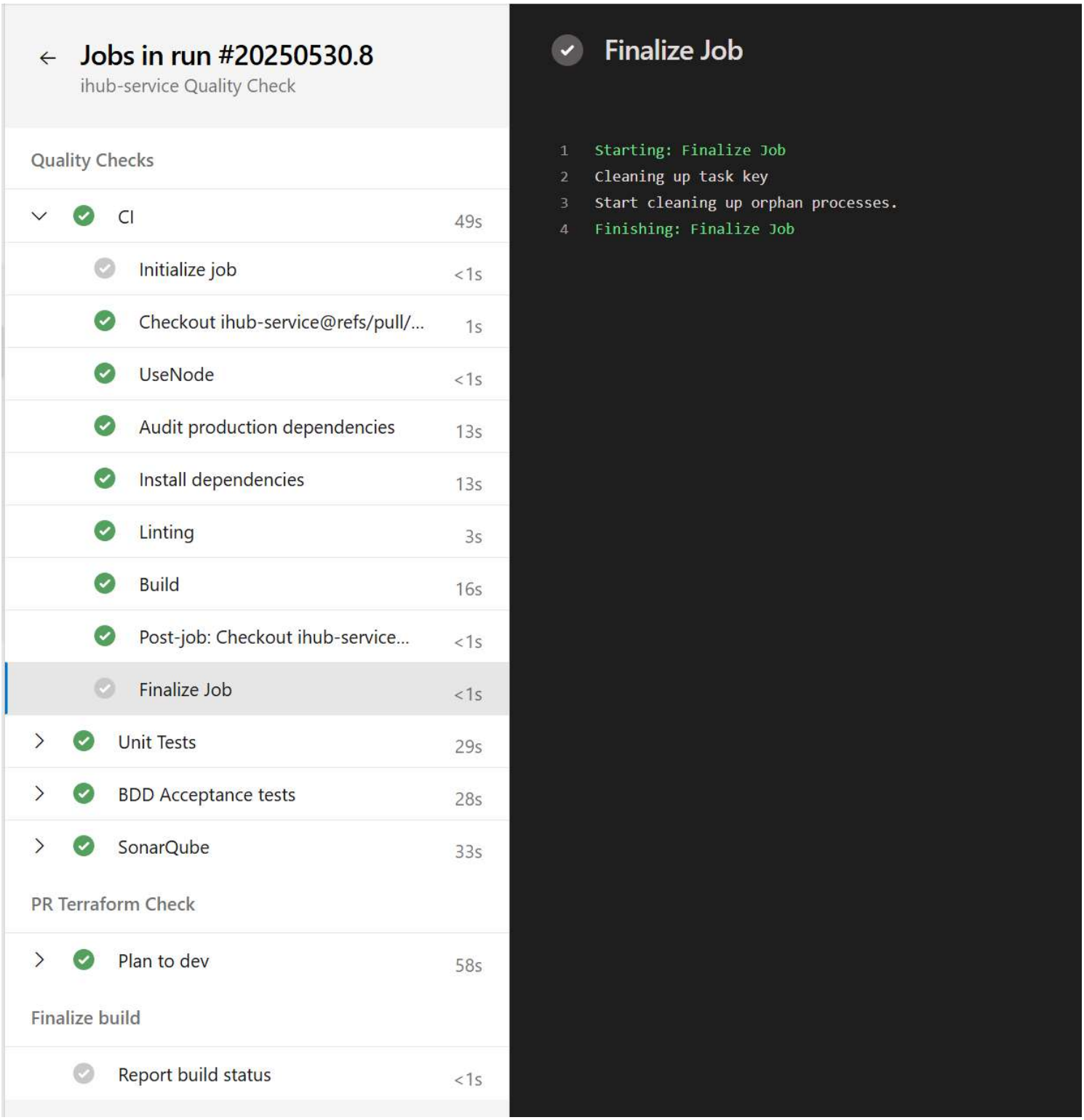
- 65980 **Add 100% test coverage for new code to all PRs (iHub)** | Done
- 69394 **Add 100% test coverage for new code to all PRs (Python)** | Done
- 70764 **Add 100% test coverage for new code to all PRs (Azure Functions)** | Done
- 70765 **Add code coverage reports and ensure 100% test coverage to all PRs (React)** | Done
- 81544 **Add 100% test coverage for new code to all PRs (Dataiku)** | New
- 55027 **Sonar Cloud pipeline scanning (iHub)** | Done
- 70759 **Sonar Cloud pipeline scanning (Python)** | Done
- 70760 **Sonar Cloud pipeline scanning (Azure Functions)** | Done

- 70761 **Sonar Cloud pipeline scanning (React)** | Done
- 81545 **Sonar Cloud pipeline scanning (Dataiku)** | New
- 70767 **Automated tests for React Apps (unit/integration test)** | Done

We check the code of our repos before and after merging PRs.

Quality checks include:

- Linting
- Building
- Unit Testing
- BDD Testing
- Code Coverage
- Code Scanning (SonarCloud)
- Terraform Coding Testing



Scheduled Health & E2E Tests

See: [Scheduled Health & E2E Test Pipelines]

Work items:

- 70862 **API Healthchecks (iHub Endpoints)** | Done
- 61700 **API Healthchecks (Dataiku Pricing Endpoints)** | Done
- 70861 **API Healthchecks (Python Endpoints)** | Done

We periodically run checks on our systems to ensure that they are working as expected.

Scheduled Health & E2E Checks include:

- Calling health endpoints on our APIs
- Running automated E2E tests on our front end applications

Deployment pipelines

See: Deployment Pipelines (iHub/Python/MEA)
See: Deployment Pipelines (Dataiku)
See: Deployment Pipelines (Q)

Work items:

- TBC

We have a diverse tech stack at 1856, with a variety of in-house and SaaS/IaaS applications.

Deployments for our internal applications and APIs are handled independently and co-ordinated across teams.

Release processes

iHub Release Process

iHub (Integration Hub) is deployed as follows:

1. Create a Semantic Versioning tag in the format `vMAJOR.MINOR.PATCH` where `MAJOR` is APIM API version (i.e. `v1.2.1`), on the `main` branch for a given microservice
2. A deployment pipeline will be triggered (see: Deployment Pipelines (iHub/Python/MEA))
3. All checks will need to pass and the deployment will need to be approved to be progressed to dev/test environments
4. Any deployment to production environments needs approval and have further checks (see: Change Control Process & Deployment Pipeline Controls)

MEA Release Process

Mea - Overview is deployed as follows:

1. Trigger deployment for Pipelines - Runs for mea-ingestion Deploy and/or Pipelines - Runs for mea-processing Deploy as required
2. All checks will need to pass and the deployment will need to be approved to be progressed to dev/test environments
3. Any deployment to production environments needs approval and have further checks (see: Change Control Process & Deployment Pipeline Controls)

Pricing APIs (Dataiku) Release Process

For the Dataiku Pricing API:

- All merges to `main` for a given Pricing API repo will deploy to the dev environment (for Dataiku Devs only)

To proceed to deploy to higher environment that proxied via iHub's Dataiku Gateway - Overview:

1. Trigger Dataiku Service Deployment Pipeline
2. All checks will need to pass and the deployment will need to be approved to be progressed to dev/test environments
3. Any deployment to production environments needs approval and have further checks (see: Change Control Process & Deployment Pipeline Controls)

See Dataiku Pipelines - Overview for more details

Python APIs Release Process

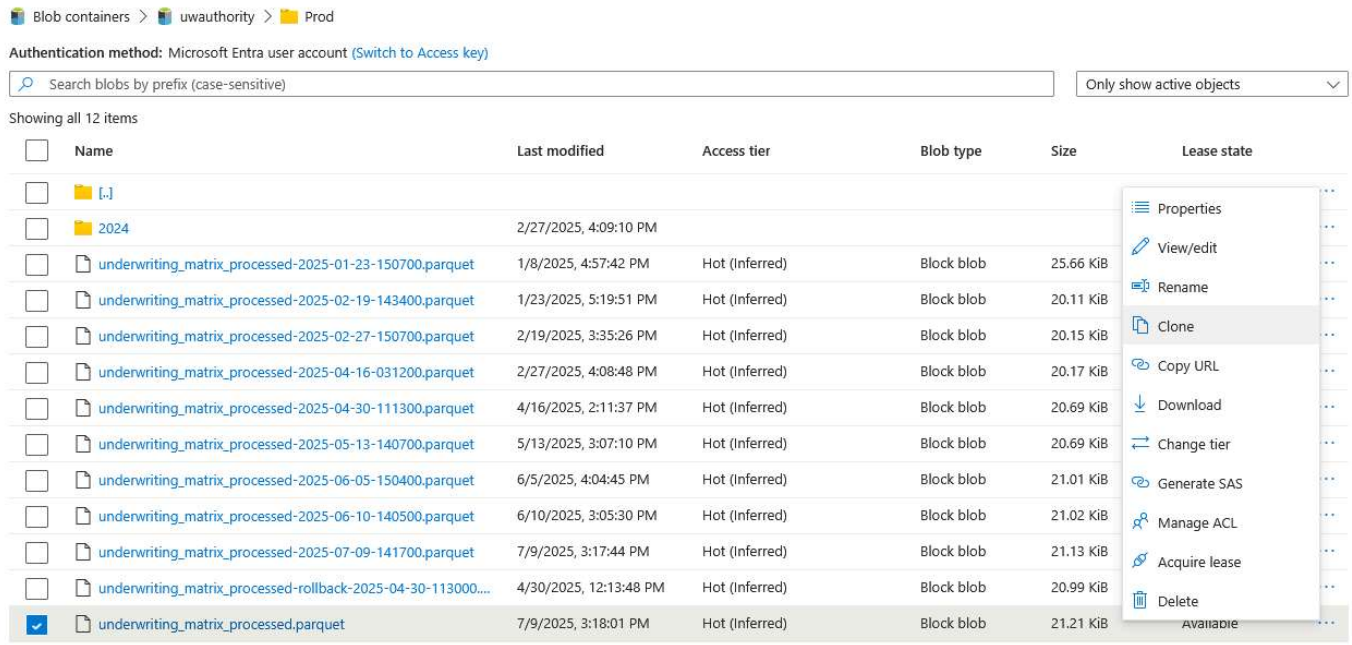
1. Create a Semantic Versioning tag in the format `vMAJOR.MINOR.PATCH` where `MAJOR` is APIM API version (i.e. `v1.2.1`), on the `main` branch for a given Python App built using the Python Starter Kit - Overview
2. A deployment pipeline will be triggered (see: Deployment Pipelines (iHub/Python/MEA))
3. All checks will need to pass and the deployment will need to be approved to be progressed to dev/test environments
4. Any deployment to production environments needs approval and have further checks (see: Change Control Process & Deployment Pipeline Controls)

Referrals Release Process

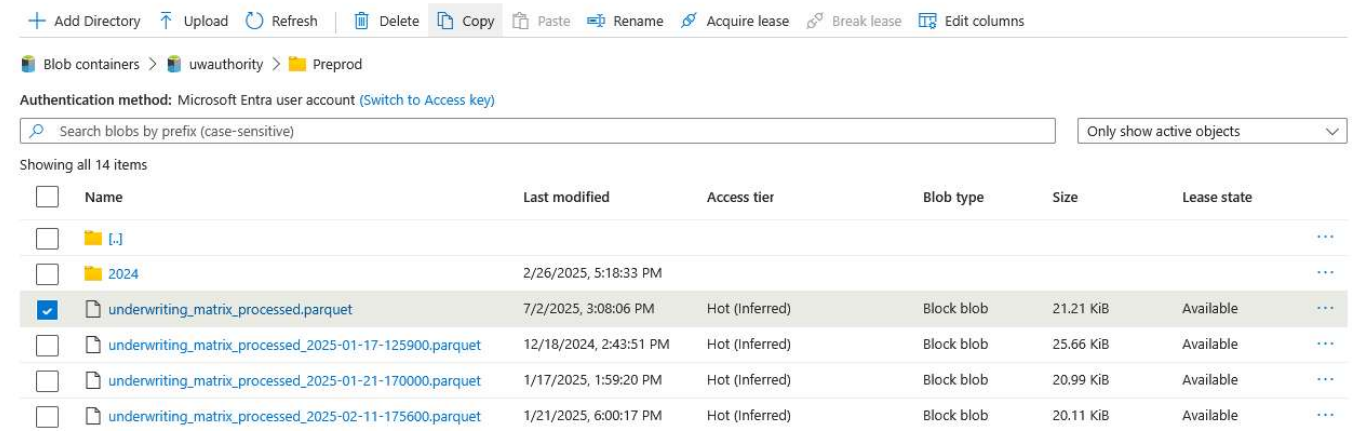
- Referrals code changes follow Dataiku Release Process
- Referrals dataset changes are released using the following procedure:

1. Non-prod releases
   - Releases to Dev and Staging envs are described here
2. Production releases
   1. In Azure navigate to Storage browser -> Subscription: `Service Model NonProd` -> Account: `stpricingdluksouthdev` -> Blob container: `uwauthority` (Make sure your ZScaler connection is authenticated).
   2. Go into the target environment folder (PreProd or Prod), select the master `underwriting_matrix_processed.parquet` file and create a backup, by clicking on the `...` menu and choosing the `clone` option. Backups are identified by a suffix containing a timestamp in the `-YY-MM-`

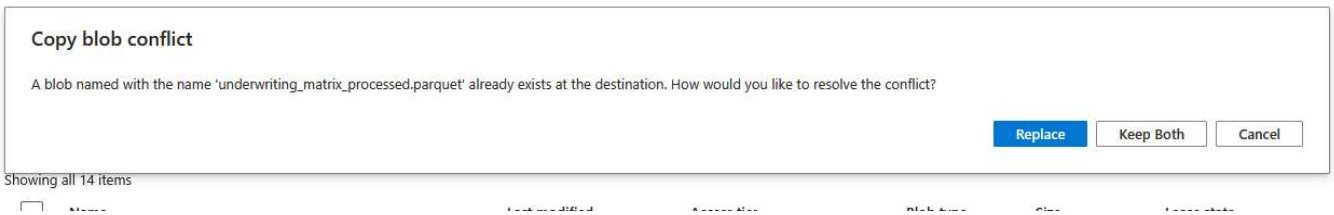
`DD-HHmmss` format in UTC time.



3. Go back to `uwauthority` blob storage and click into one environment lower (Staging -> PreProd, PreProd -> Prod). Make sure the lower environment has been deployed previously. Select the master `underwriting_matrix_processed.parquet` file and click the `Copy` button in the top bar.



4. Once again go into the target environment folder and click the `Paste` button in the top bar. When prompted, select `Replace` from the pop up dialog. The updated master file will by automatically picked up by the API code.



Notes:

- Code for release automation exists in several shapes and forms, but is not used for Prod releases due to authorization concerns, since the solution relies on Dataiku functionality, and the control mechanisms are not ideal. It shouldn't be too hard to replace the manual procedure with a ADO pipeline, if need be.

Q Web App / API Release Process

(TBC)

Workbench Release Process

Administrator->UDLC Toolkit-> Release Management
Click-> Create New Release
Version-> Enter the version of the release based on Sprint. Eg: 12.5
Name-> Enter release name. Eg: "release_12.5 Prod"
Notes-> This is optional.
Client-> Select "ers"
Environment-> Select "production"
Select Workspace applications modules. Full and partial selection can be done. In very rare cases partial releases are done.
After all selection of Workspace applications modules click on Create new Release
From Release dashboard select newly created release.
Click on Promote button.
Once all the Workspace applications status turned into green,click on Finish button.

More information on Release Managment can be read here

ODS Release Process

(TBC)