Custom HR Identity Source and Okta Integration

A portfolio case study in approval controlled onboarding and continuous identity delivery

Project outcome

I deployed a custom HR portal in Docker and connected it to an Okta Custom Identity Source. The project progressed from offline previews and a controlled single-user pilot to a continuously running service that delivers eligible hires to Okta. This replaced the need to execute a separate provisioning command for every new employee.

I first populated Okta with six employees already staged in the HR platform, including two employees whose imports depended on their managers being present. I then used Ben Solo as a final end-to-end test of the GUI-driven workflow. The background service delivered the new hire without per-user scripts, manual uploads or terminal commands.

Why I built it

I wanted HR staff to initiate onboarding through a portal while a continuously running service handled identity delivery. The design separates HR submission and approval from backend processing, preserves effective dates and checks manager dependencies before sending employees to Okta.

Architecture and ownership

The application runs on a headless Ubuntu VM hosted in Proxmox. Docker Compose separates the Django web application, PostgreSQL database, effective-date worker and Okta sync service. PostgreSQL stores HR records and change requests; the sync service maintains durable delivery checkpoints.

The HR platform owns employment data and approved changes. A Python sync service reads eligible records and sends them through the Custom Identity Source import process. Okta handles imported profiles and user creation decisions. The OAuth service application provides API authorization and is distinct from the identity source application.

Scope of this milestone

Implemented: HR requests and reviewer approval, effective-date handling, key-based API authentication with DPoP, profile mappings, a verified pilot and continuous hires-only delivery. Planned for this integration: RBAC, automated mover and leaver workflows, downstream application provisioning and richer operational monitoring.

The HR platform

The JBC HR portal provides a graphical interface for employee records and onboarding approvals. This view shows the completed HR population: six initial employees plus Ben Solo, the final hire used to validate GUI-driven backend delivery.

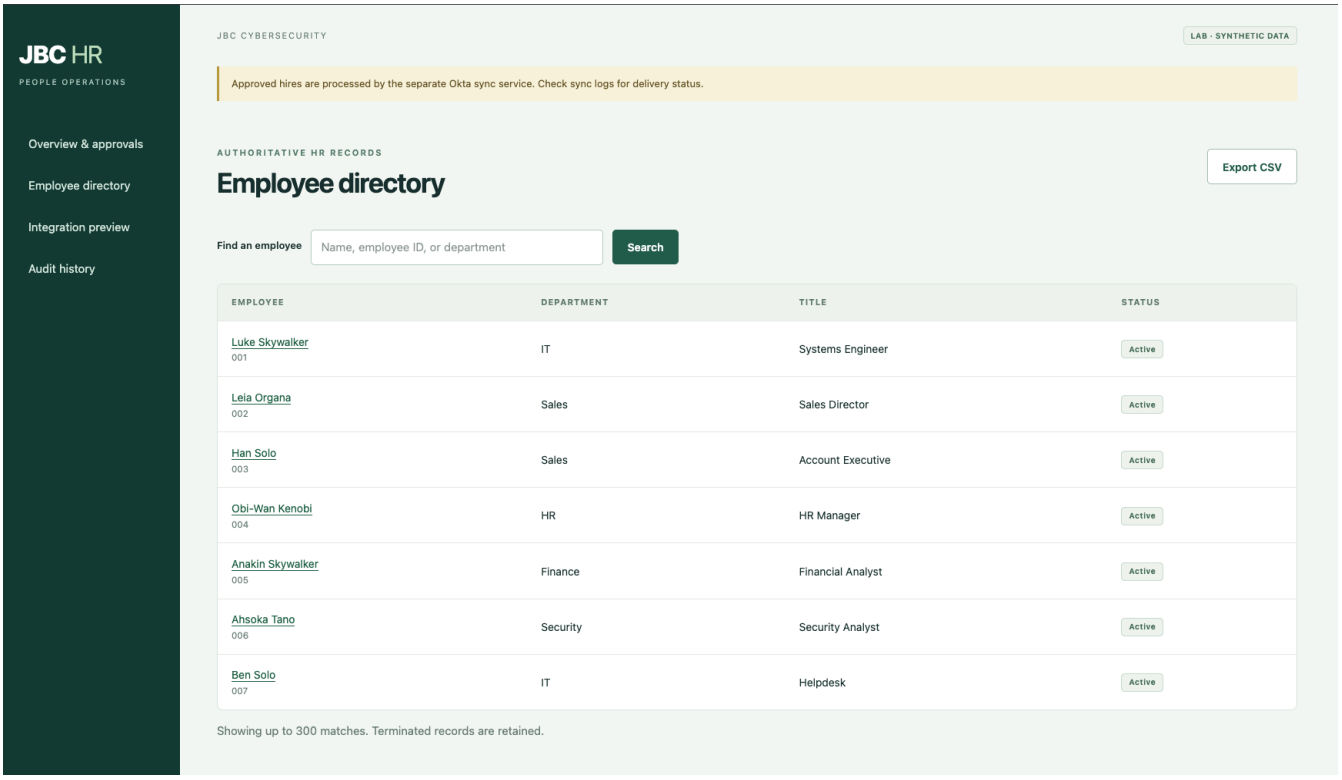


Figure 1. The HR employee directory shows seven synthetic employees, their IDs, departments, titles and HR status. The sidebar provides access to approvals, the directory, integration previews and audit history.

A central view of employment data

The directory brings the identity attributes used during onboarding into one searchable view. Employee IDs link each record to the integration workflow, while department and title describe the employment information supplied to Okta. The Active labels shown here describe HR record status, not Okta account activation.

HR operations and background delivery

Users work through the portal to submit and review hires. Once an approved hire is effective, the separate Okta sync service performs backend delivery. The banner identifies this separation and directs operators to the sync logs for delivery status; it is not a live service health indicator.

Securing and validating the integration

Key based authentication

I configured an Okta service application with a registered public key and a locally stored PEM private key. The private key is excluded from source and image builds. OAuth scopes and the service application administrative role must both authorize the requested API operations.

I implemented Demonstrating Proof of Possession, or DPoP, rather than disabling the requirement. Troubleshooting included nonce challenges and an invalid htu claim during user lookup. Correcting the proof and request URL handling allowed verification to proceed.

Read only checks before delivery

Before enabling the automatic worker, I ran an offline plan and a read-only preflight. The output identifies six candidate employees, validates read access and reports no active import sessions. It explicitly records zero employee changes and does not claim that write authorization or automatic confirmation has been tested.



Figure 2. AUTO_PREFLIGHT_PASS confirms the read checks while delivery remains disabled for this test.

Profile mapping

I mapped the HR employee ID to Okta employeeNumber as a string, preserving values such as 002. Department maps to department, job_title to title and employee_type to userType. A missing userType value was detected during verification and corrected so Leia's Okta profile displayed FTE.

Manager data required an explicit convention. The connector supplies the manager login in managerId and the display name in manager. Manager-dependent hires wait for the manager to be verified. Populating these fields is not, by itself, proof of every downstream manager relationship.

Controlled pilot and recovery

I first delivered a single employee with an explicit login and snapshot confirmation. An empty accepted upload response exposed a response-parsing issue. Recovery reused the recorded import session instead of creating a replacement session or re-uploading the profile. After the pilot succeeded, the automatic worker adopted that employee without another upload.

Importing the existing HR employees

The initial population used six synthetic employees already staged in HR. After approval and effective-date checks, the background service processed eligible records. Leia 002 was adopted from the successful pilot without another upload. Luke 001, Obi-Wan 004 and Anakin 005 were delivered through subsequent import sessions.

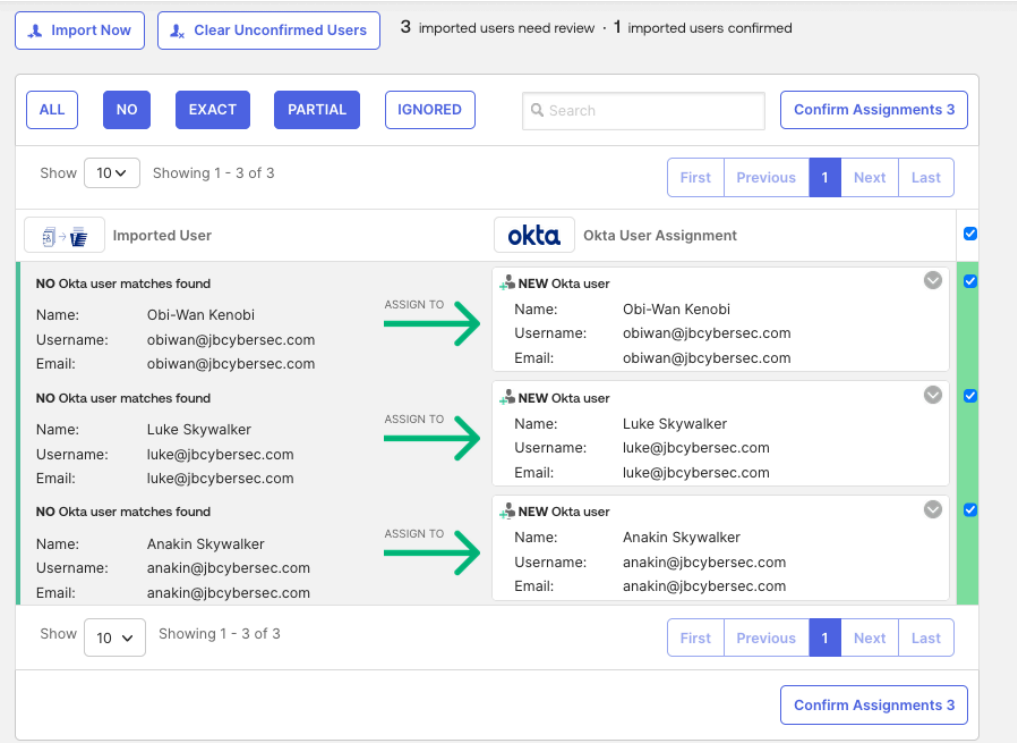


Figure 3. The initial import queue contains Obi-Wan, Luke and Anakin, with Leia already confirmed. The three proposed accounts are ready for review.

Verification after confirmation

After I confirmed these users in Okta, the worker verified their destination profiles. The log below records VERIFIED checkpoints for 001, 004 and 005, followed by delivery of employee 003. Persistent checkpoints allowed processing to continue without replaying completed imports.

```
okta-sync-1  | {"event": "AWAITING_OKTA_CONFIRMATION", "employee_id": "001"}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "001", "stage": "VERIFIED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "004", "stage": "VERIFIED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "005", "stage": "VERIFIED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "CREATE_INTENT", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "CREATED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "UPLOAD_INTENT", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "UPLOADED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "TRIGGER_INTENT", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "TRIGGERED", "detail": ""}
okta-sync-1  | {"event": "DELIVERY_CHECKPOINT", "employee_id": "003", "stage": "COMPLETED", "detail": ""}
okta-sync-1  | {"event": "AWAITING_OKTA_CONFIRMATION", "employee_id": "003"}
okta-sync-1  | {"event": "SESSION_SPACING_WAIT"}
okta-sync-1  | {"event": "AWAITING_OKTA_CONFIRMATION", "employee_id": "003"}
okta-sync-1  | {"event": "SESSION_SPACING_WAIT"}
okta-sync-1  | {"event": "AWAITING_OKTA_CONFIRMATION", "employee_id": "003"}
okta-sync-1  | {"event": "SESSION_SPACING_WAIT"}
```

Figure 4. Verified initial users are followed by creation, upload and completion of the next employee import.

Importing employees after their managers

Manager-dependent employees could not be delivered until their managers already existed in Okta and had been verified by the worker. This ordering ensured that the manager identity was available before the direct report was imported.

Employee	Manager required in Okta first
Han Solo 003	Leia Organa 002
Ahsoka Tano 006	Obi-Wan Kenobi 004

Once the manager prerequisites were satisfied, the service delivered Han and Ahsoka. I completed their Okta confirmation, bringing the initial HR population into the directory.

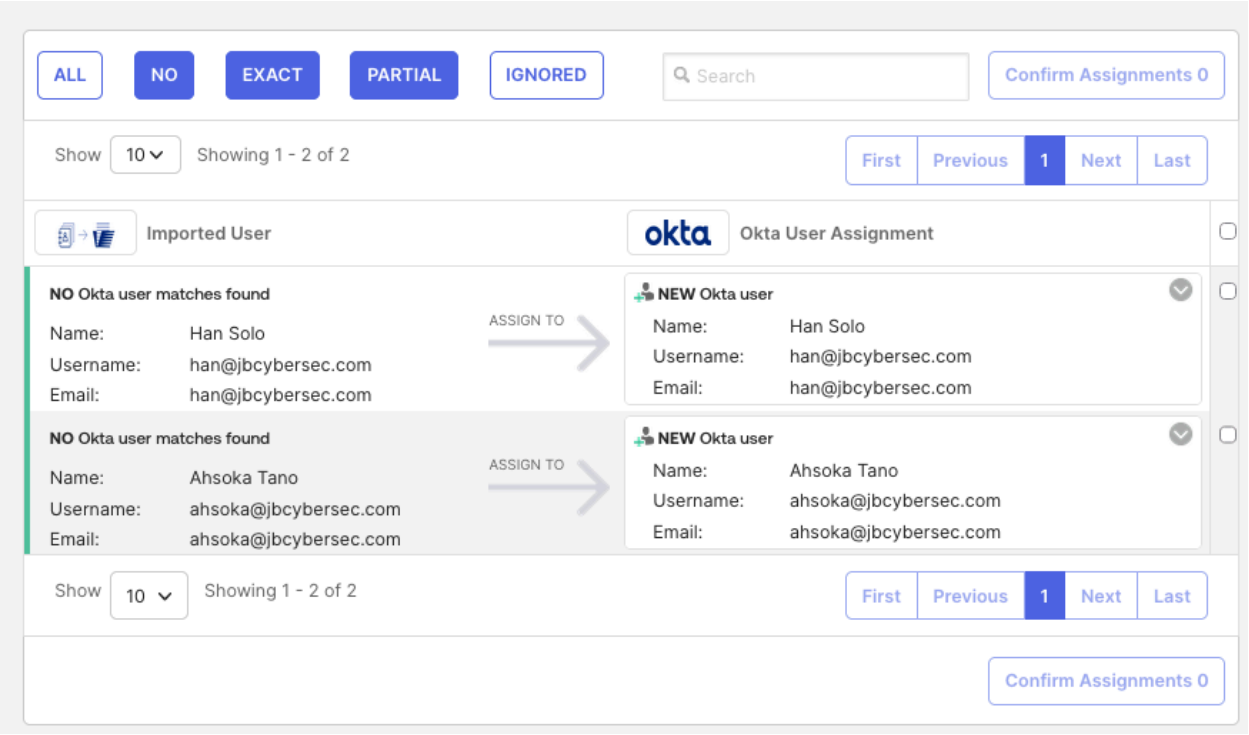


Figure 5. Han and Ahsoka appear in the later import group after their manager dependencies were satisfied.

Initial population completed

All six initial employees were successfully added to Okta. The final Ben Solo import screenshot later in this case study also records six previously confirmed imported users. Together, the staged import views, verification log and completed lab outcome show the progression from initial population to manager-dependent delivery.

Final GUI driven onboarding test

With the initial population complete, I tested the full workflow by creating a hire request in the HR GUI for employee 007, Ben Solo, with department IT, job title Helpdesk, employee type FTE and manager employee ID 001. The portal records the requester, business reason and effective date alongside the proposed profile values.

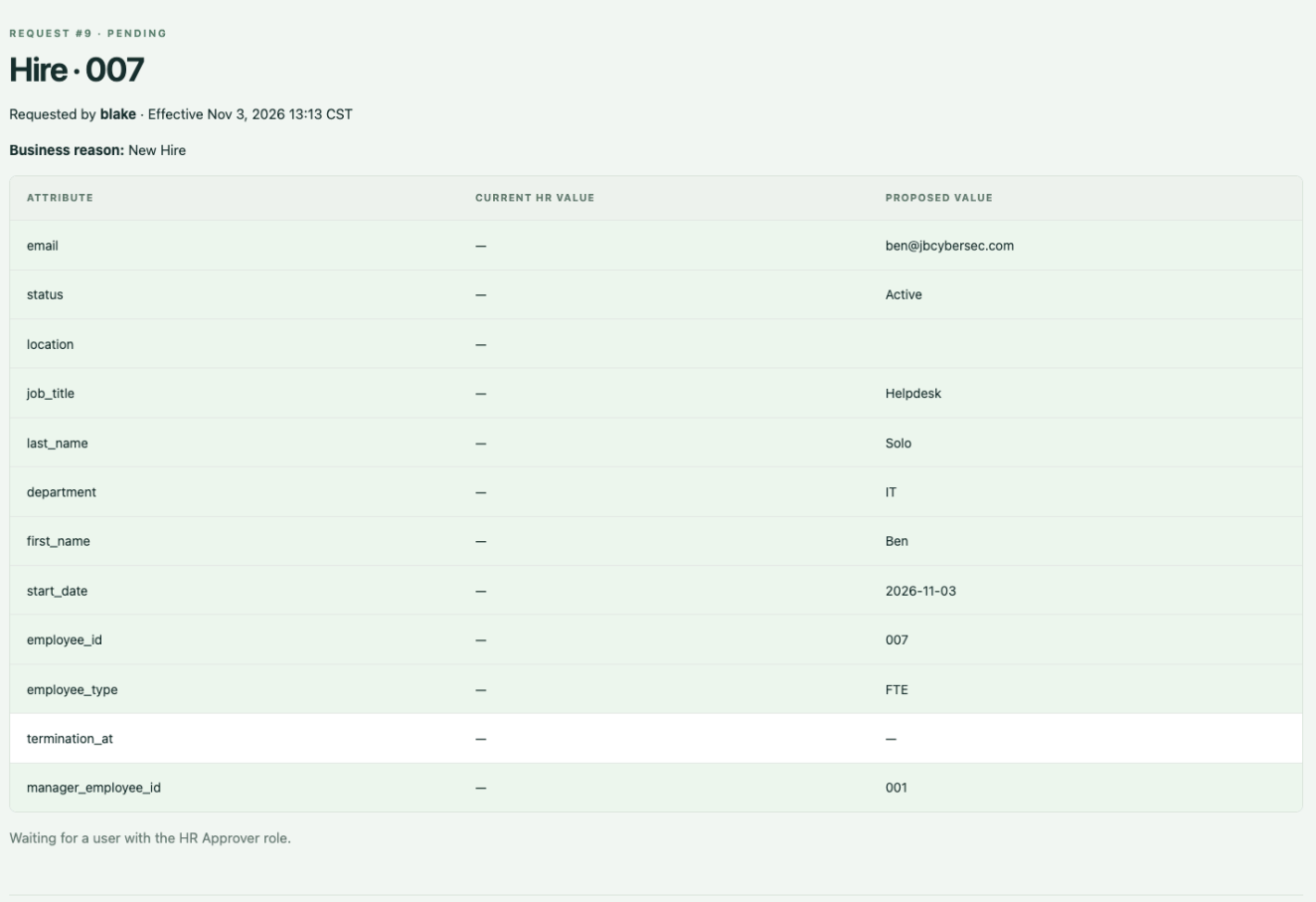


Figure 6. The pending HR request contains the employee attributes and explicitly waits for an HR Approver.

Approval before eligibility

A separate reviewer account can approve or reject the request. Approval and the effective date determine when an HR change can be applied. A future-dated hire therefore does not become eligible simply because the request was approved.

The reviewer screenshot in the evidence set shows the approval form with Approve selected, before the decision is recorded. It demonstrates the review interface, not by itself a completed approval event. The subsequent Okta import provides evidence that the record progressed to delivery.

Final test result in Okta

The Ben Solo test verified start-to-finish functionality from GUI submission and HR approval through backend delivery to Okta, followed by my final Okta confirmation. Once the approved hire became effective, the running service handled the import automatically. No per-user backend execution, manual data upload or service restart was required.

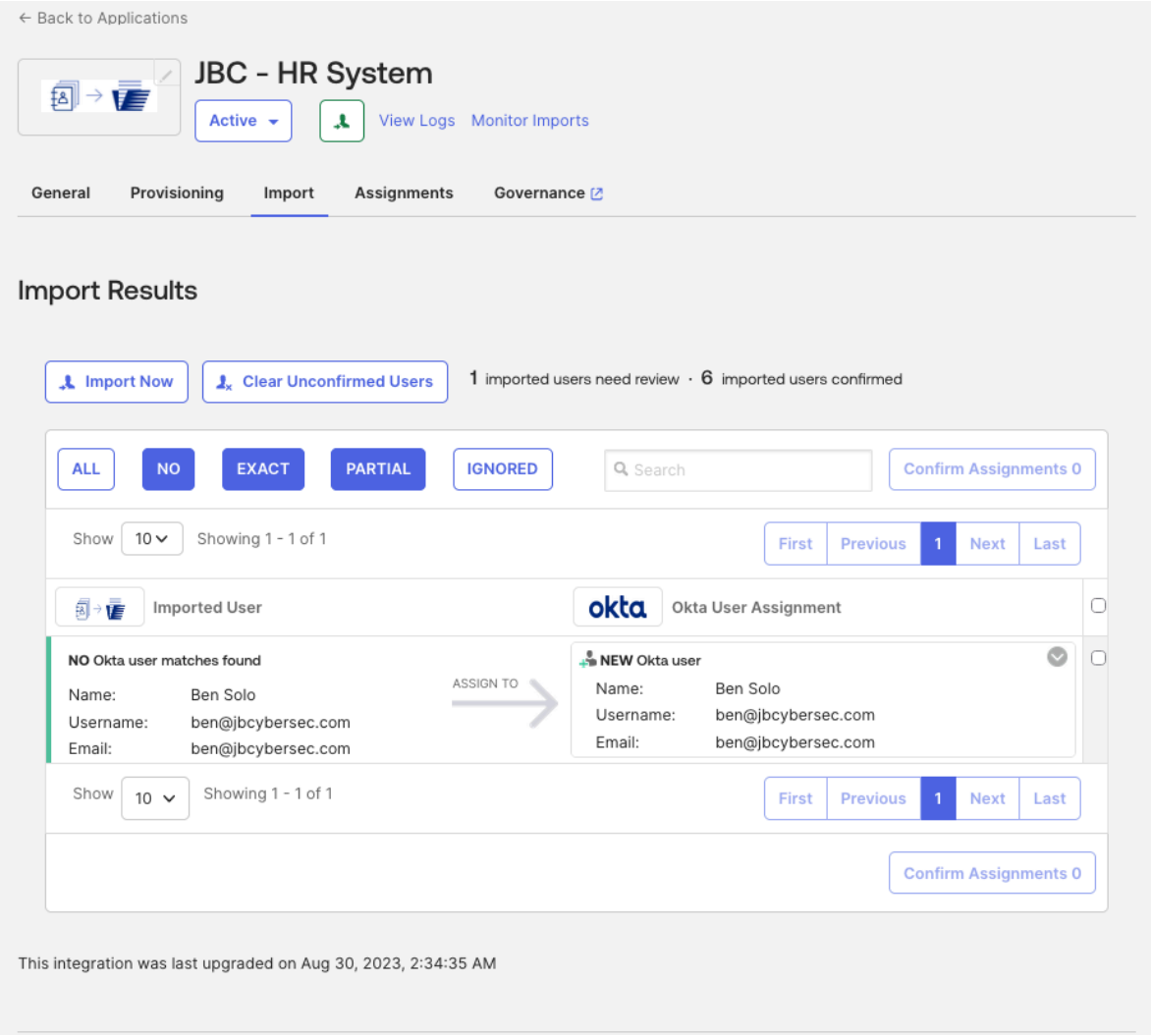


Figure 7. Ben Solo appears as an imported user with no existing Okta match. One user still needs review; six earlier imported users are confirmed.

The screenshot captures Ben awaiting Okta review; I subsequently confirmed and added him to the directory alongside the six initial users. This was a zero-touch backend delivery test, with HR approval and Okta confirmation remaining human workflow steps. The final directory addition is a subsequently confirmed lab outcome. Activation and sign-in were outside this test.

Operational lessons and next steps

Continuous service operation

The sync service runs independently of the HR web interface. Docker restart policies support recovery when the daemon starts, while persistent state retains delivery history. A temporary HTTPS token failure paused processing and later recovered, demonstrating why transient connectivity errors should not trigger a blind re-upload.

I documented startup order, health checks, log inspection and safe pause and resume procedures in a reboot recovery runbook. Database volumes, private keys, pilot receipts and the delivery ledger must survive restarts. The runbook is an operating procedure, not evidence of a completed disaster recovery exercise.

Keeping status messages accurate

The original portal banner said Okta delivery was disabled because it had been built for preview-only operation. That text was hardcoded and did not reflect the separate sync service. The updated banner directs operators to the sync logs, as shown in the HR platform screenshot. A live health indicator remains a future enhancement.

What this project demonstrates

This implementation gave me practical experience separating HR data ownership from identity delivery, configuring authenticated API access, handling asynchronous imports and verifying mappings against actual Okta profiles. It also reinforced the importance of durable checkpoints when an API response leaves the outcome uncertain.

The most important validation lesson was to avoid treating a successful HTTP request or completed import session as proof that onboarding was finished. Each stage needs its own evidence, especially when a confirmation step can still block account creation.

Next validation milestones

The GUI-driven hire test completed the backend delivery milestone. Next, implement RBAC and verify intended group assignments. Automatic Okta confirmation and account activation can be tested separately if those workflow steps are enabled. Extend this HR integration to mover and leaver scenarios with controlled tests, audit records and recovery behavior.

Evidence and publication scope

This case study uses the supplied HR and Okta screenshots and the implementation results recorded during the lab. Figures retain synthetic test identities and the portfolio domain. Private keys, access tokens, activation links and tenant-specific sign-in details are not included. The final addition of Ben to the directory is recorded here as a subsequent lab outcome, rather than represented as visible in the earlier screenshot.