UX DESIGN BRIEF

Amazon SageMaker Unified Studio Release Center

PRODUCT	FEATURE	DESIGNER	STATUS
Amazon SageMakerUnified Studio	**Release Center**	**Ofir Levy, Sr. UX**	**Shipped, 2024**

Executive Summary

Amazon SageMaker Unified Studio (SMUS) consolidated Amazon's fragmented ML tooling into a single collaborative environment for data engineers, data scientists, and ML engineers. The Release Center was a critical governance surface within SMUS that gave teams a structured, auditable mechanism to promote ML assets, datasets, and pipeline artifacts from development environments into production, while maintaining traceability across project boundaries.

This brief captures the design problem, user context, constraints, design decisions, and success criteria for the Release Center feature. It is intended to serve as a reference document for stakeholders, future designers, and portfolio reviewers.

Problem Statement

The Core Challenge

ML teams operating within AWS had no standardized, first-party workflow for releasing versioned assets from project workspaces into shared catalogs or production targets. Teams relied on ad-hoc scripts, Slack threads, and manual documentation to coordinate releases, which created audit gaps, introduced version conflicts, and made rollbacks difficult to trace.

The absence of a unified release workflow created compounding friction across the ML lifecycle: experimentation, validation, approval, and deployment each lived in separate tools, and handoffs between them were manual and error-prone.

Before Release Center

- Model versions tracked manually in wikis or spreadsheets

Target State

- Single pane of glass for all release activity within a project

• No in-product approval workflow for production promotion • Release history scattered across Slack, email, and Git commits • Rollback required engineering intervention with no UI support • Compliance teams had no auditable trail within AWS console	• Structured promotion workflow with configurable approval gates • Versioned, timestamped release history with author attribution • One-click rollback to any prior release with diff visibility • Exportable audit log for compliance and governance review

Users & Context

The Release Center was designed to serve three overlapping personas operating within a shared SMUS project. Each persona interacts with the release workflow at a different stage and with distinct priorities.

Persona 1 — The Data Scientist / ML Engineer

Initiates releases from within their development workspace. Their primary concern is speed and simplicity: they want to register a new release without context-switching out of their flow. They are not inherently interested in governance, but they need confidence that their submission will route correctly.

- Goal: Promote a trained model or processed dataset to a named release
- Frustration: Multi-step processes that require external documentation or ticketing
- Mental model: Git push analogy, expects tagging, commit message, and confirmation

Persona 2 — The Project Lead / Tech Lead

Reviews and approves incoming release requests. Has a clear picture of what is safe to promote and what requires additional validation. Often coordinates across multiple contributors and needs to see release state at a glance without reading detailed logs.

- Goal: Review release details, validate readiness, approve or reject with comments
- Frustration: Approval requests with insufficient context, no diff visibility
- Mental model: Code review workflow, expects structured diff, summary, and action button

Persona 3 — The Platform / Compliance Stakeholder

Monitors release history for governance and compliance purposes. Rarely initiates releases but needs reliable access to historical records. May operate across many projects simultaneously and values exportability and filter controls above all else.

- Goal: Audit who released what, when, from which environment, with which approval chain
- Frustration: Fragmented records, inability to export structured audit data
- Mental model: Admin dashboard, expects sortable, filterable log with export

Design Goals

The following goals were established in collaboration with the product manager and engineering leads during the discovery phase. They served as the criteria against which design decisions were validated throughout the project.

01	Make the release initiation path feel as lightweight as a Git commit. Remove every step that does not add meaningful signal.
02	Surface the right context at the right time. Approvers should never have to leave SMUS to get the information they need to make a decision.
03	Treat the release history as a first-class asset. It must be searchable, filterable, and exportable without engineering involvement.
04	Design for rollback confidence. Teams must trust that reverting a release is safe, fast, and traceable.
05	Reduce cognitive load through progressive disclosure. Advanced options and detailed metadata should be accessible but not prominent for the common case.
06	Support compliance requirements without compliance theater. Audit trails should be automatic, not an extra step imposed on contributors.

Scope & Constraints

In Scope

- Release initiation workflow from project workspace
- Release detail view with metadata, diff summary, and lineage
- Approval and rejection flow with comment threading
- Release history log with search, filter, and sort
- Rollback initiation with confirmation and downstream impact preview

Out of Scope (V1)

- Multi-project release coordination and cross-project dependencies
- Automated release scheduling and CI/CD pipeline integration
- Role-based approval policy configuration (delegated to platform team)
- Mobile or tablet-optimized layouts
- Real-time collaboration on approval decisions

- Audit log export (CSV / JSON)
- Status badge system (Draft, Pending Approval, Approved, Rejected, Rolled Back)

- Native notification system (email was the V1 channel)

Engineering Constraints

- SMUS used AWS CloudScape Design System exclusively. No custom component library was permitted.
- Release metadata was backed by DynamoDB with eventual consistency, meaning status updates required polling rather than websockets in V1.
- The diff engine was shared infrastructure owned by a separate team, limiting the visual control Ofir had over diff rendering.
- Page load performance targets: release list to render within 2 seconds at p95; detail view within 1.5 seconds.
- Accessibility: WCAG 2.1 AA compliance required throughout.

Key Design Decisions

Decision 1: Unified Status Bar over Tabbed Navigation

Early explorations used a tab-based layout (Pending, Approved, History) to organize releases. User testing revealed that reviewers frequently needed to cross-reference states, toggling back and forth between tabs to build context. The final design replaced tabs with a single filtered table using a persistent status badge column, allowing users to scan across all release states in one view.

Decision 2: Inline Approval with Contextual Metadata

The original approval flow directed approvers to a separate full-page form. This was simplified to a side panel that opened over the release list, preserving list context while surfacing just enough metadata (submitter, timestamp, changed assets, linked pipeline run) for a confident decision. This reduced the median approval task time significantly in usability testing.

Decision 3: Rollback as a Promoted Action, Not a Hidden Control

Initial designs buried rollback behind a kebab menu to de-emphasize it and avoid accidental use. Feedback from tech leads indicated the opposite need: they wanted rollback to be visually prominent so they could trust that the system supported recovery. The final design elevated rollback to a clearly labeled button within the release detail, protected by a confirmation dialog that surfaced downstream impact.

Decision 4: Progressive Disclosure on Release Initiation

The release initiation form was simplified to four required fields for the common case: release name, target environment, linked assets, and release notes. Advanced options (custom approval chain, notification overrides, metadata tags) were placed in a collapsible section accessed by fewer than 20% of users during testing, validating the progressive disclosure approach.

User Stories & Priorities

Priority	User Story	Outcome
P0 — Must Ship	As an ML engineer, I can initiate a release from my workspace with a name, notes, and target environment.	**Release submitted**
P0 — Must Ship	As a project lead, I can approve or reject a pending release with a required comment.	**Release approved/rejected**
P0 — Must Ship	As any team member, I can view a chronological history of all releases in a project.	**Full history visible**
P1 — Should Ship	As a project lead, I can roll back to a prior approved release from the history view.	**Rollback initiated**
P1 — Should Ship	As a compliance stakeholder, I can export the release history as a CSV.	**Audit log exported**
P1 — Should Ship	As any user, I can filter the release history by status, submitter, and date range.	**Filtered results displayed**
P2 — Nice to Have	As a project lead, I can view a diff of changed assets between two releases.	**Diff rendered**
P2 — Nice to Have	As an ML engineer, I can save a release as a draft before submitting for approval.	**Draft saved**

Information Architecture

The Release Center was structured as a secondary navigation destination within a SMUS project. The entry point lived in the left project navigation rail under the "Governance" section, alongside the Data Catalog and Access Controls surfaces.

Release Center — Primary Surfaces

- Release List (default view): Filterable, sortable table of all project releases with status badges, submitter, target environment, and timestamp
- Release Detail: Full metadata view with asset list, diff summary, approval history, and action controls (Approve, Reject, Rollback)
- New Release Modal: Step-by-step initiation form with progressive disclosure for advanced settings

- Audit Log Export: Triggered from Release List toolbar, produces CSV or JSON download of filtered results

Success Metrics

Success for the Release Center was defined across three dimensions: adoption, efficiency, and compliance impact.

Adoption	Efficiency	Quality
• 70%+ of SMUS projects using Release Center within 90 days of launch • Release Center as the primary release mechanism for >50% of production promotions	• Median release initiation time under 3 minutes • Median approval decision time under 10 minutes from notification	• CSAT score of 4.0+ (out of 5) from release submitters and approvers • Zero critical accessibility violations at launch

Open Questions & Known Risks

Open Questions at Launch

- How should the design scale when a project has hundreds of releases? Pagination vs. infinite scroll was unresolved at V1 launch.
- Should rejected releases be permanently hidden from the default view, or always accessible with a filter?
- What is the right model for releases that span multiple asset types (model + dataset + pipeline) simultaneously?
- How will the Release Center integrate with external CI/CD pipelines as SMUS matures?

Known Risks

- Eventual consistency in status updates could create a confusing experience if a user refreshes immediately after approving a release.
- The rollback confirmation dialog assumed users understood downstream impact. This assumption was not validated with compliance stakeholders before launch.
- Diff rendering relied on a shared internal service with no SLA guarantee, creating a potential gap in the approval experience.

Prepared by Ofir Levy, Senior UX Designer — ofirlevydesign.com

This brief reflects the design process and decisions for the Amazon SageMaker Unified Studio Release Center, shipped in 2024. Content is based on the designer's first-hand project knowledge.