Alma Cloud Apps

Fundamentals & Development

SLSP Developer Forum 2025

Agenda

1. **Introduction to Cloud Apps**
2. **Use Cases, Capabilities & Examples**
3. **Technical Foundation**
4. **Development Essentials**
    - 4a. Angular & RxJS
    - 4b. Working with the SDK
    - 4c. Configuration (i18n & manifest.json)
5. **Publishing & Lifecycle**
6. **Reference & Resources**
7. **Hands-on Time**

1. Introduction to Cloud Apps

What are Alma Cloud Apps?

Concept:

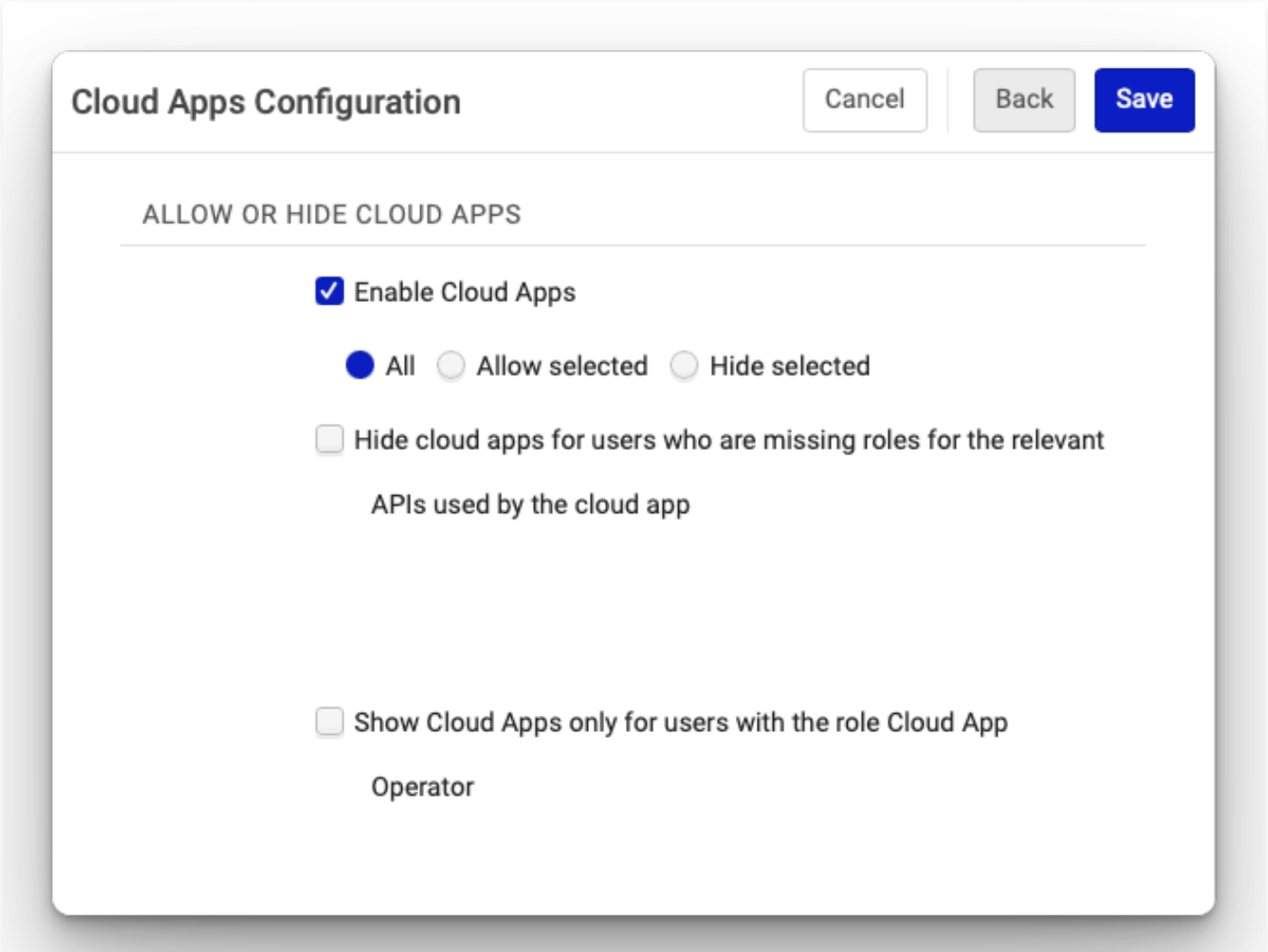
- Custom extensions for Ex Libris Alma platform
- Run directly in a sidebar in Alma
- Extend functionality via **Alma REST API**
- Connect to systems via **External APIs**

Key Benefits:

- Integrated user experience
- Can be shared across institutions via Cloud App Center
- No separate hosting or authentication needed

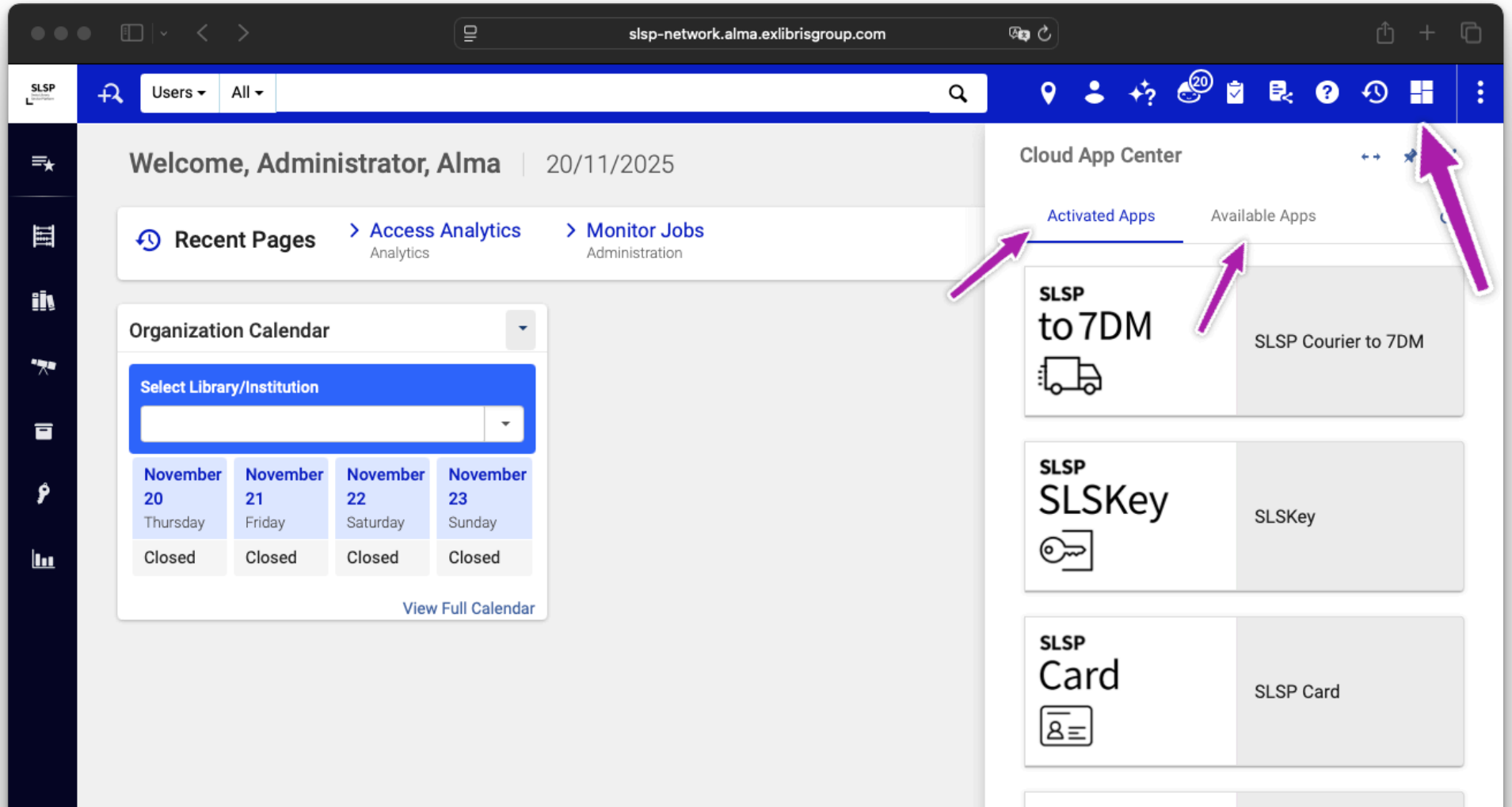
How to activate and use them?

- Activate them in the Institution Zone via: Configuration > General > Cloud Apps



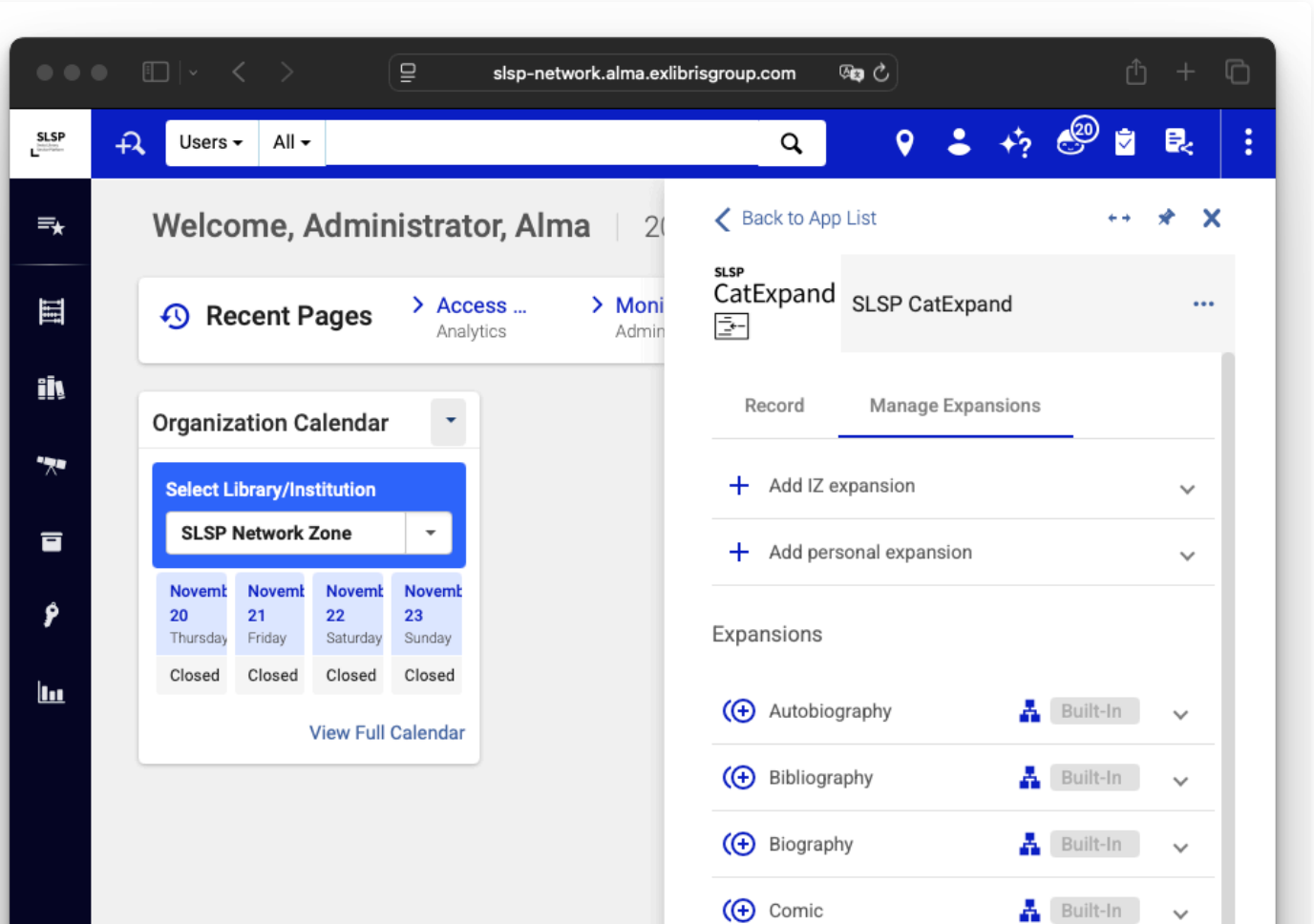
How to activate and use them?

- Use the Cloud App Store to install and configure apps

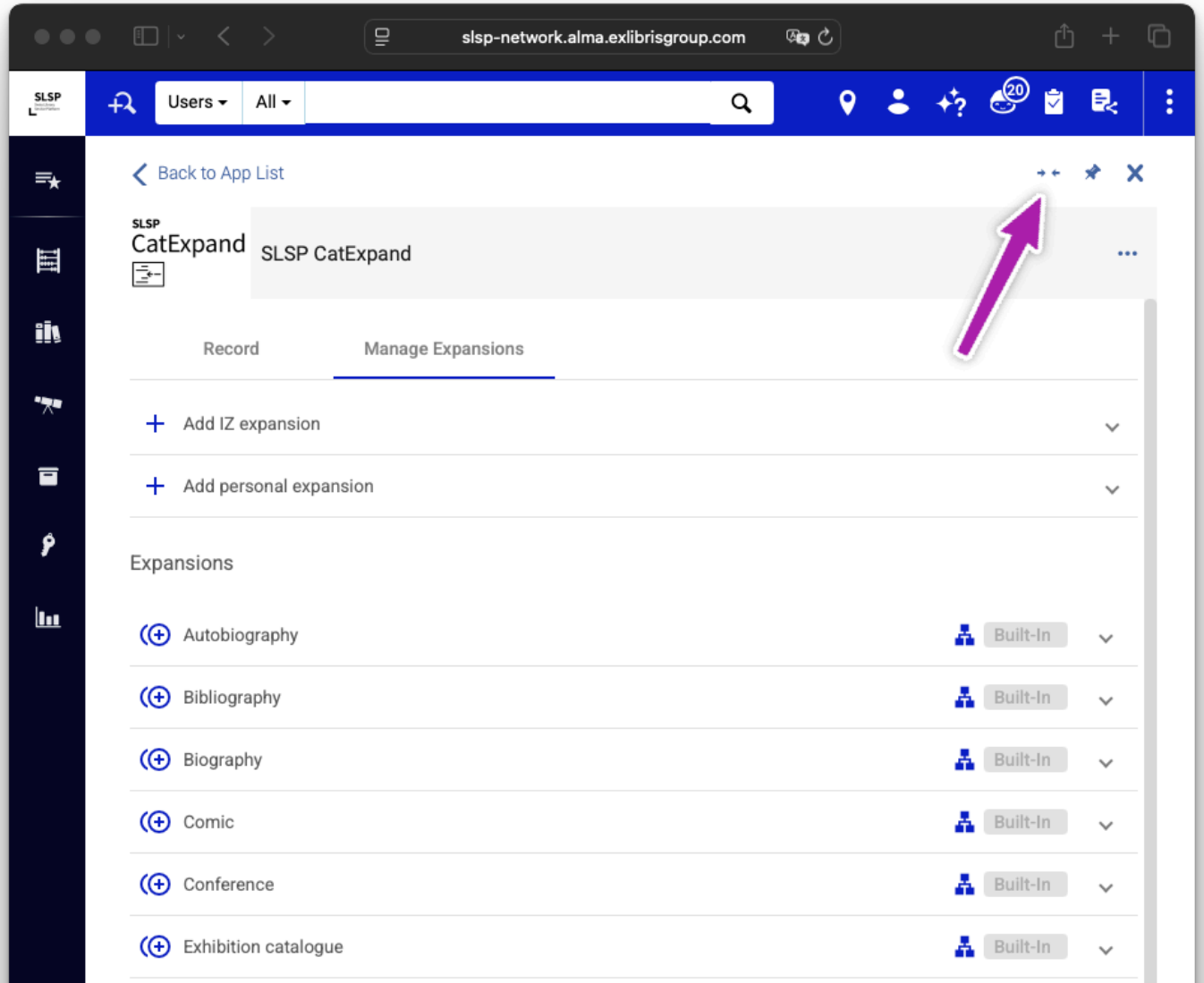


Two Types of Cloud Apps

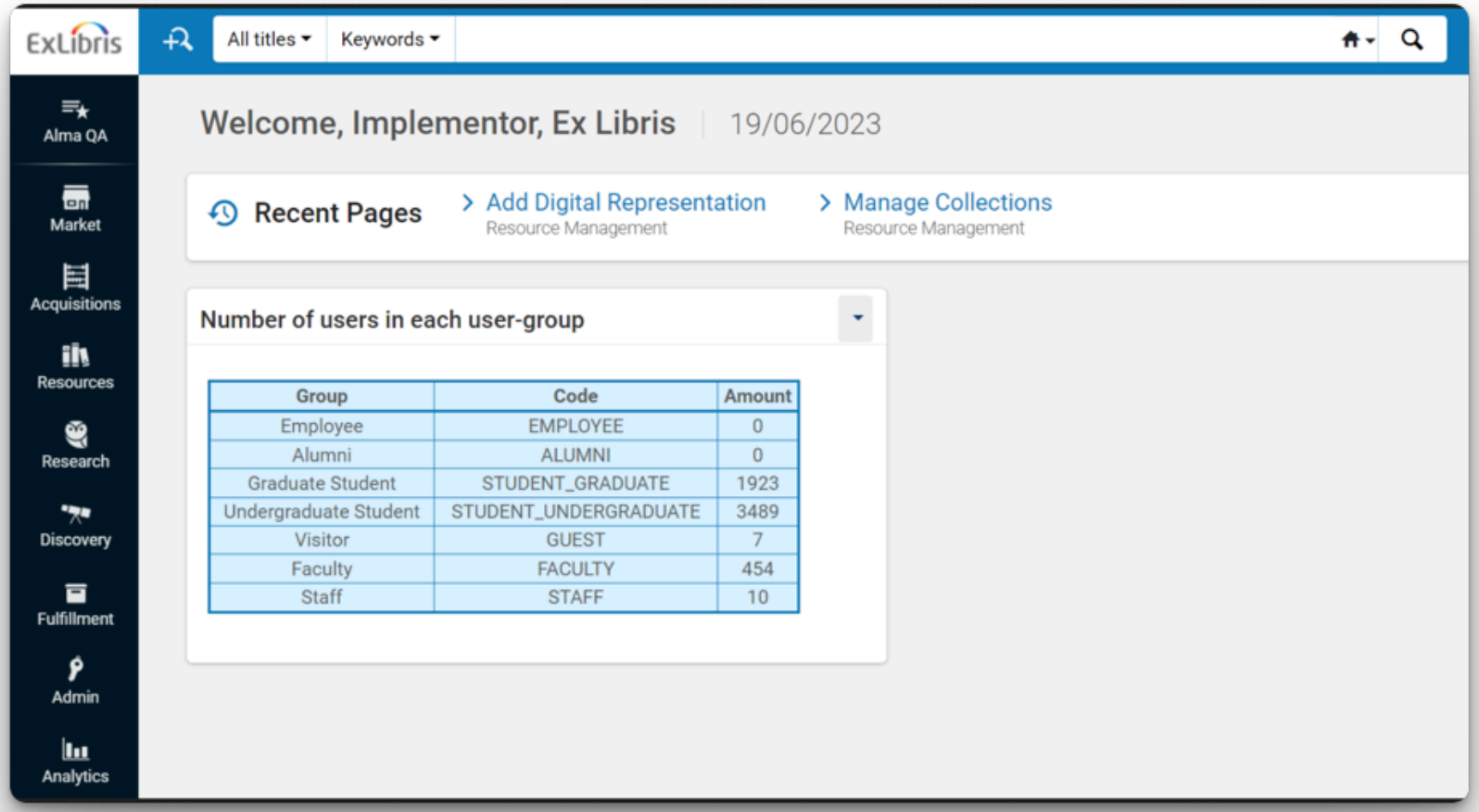
- **Full-page apps:** Standalone applications in Alma sidebar
  - sidebar can be resized to full width, if needed



- **Full-page apps:** Standalone applications in Alma sidebar
  - sidebar can be resized to full width, if needed



- **Dashboard widgets:** Small components on Alma dashboard
  - Quick access to important info or actions



2. Use Cases, Capabilities & Examples

Why Build Cloud Apps?

Key Benefits:

- ✅ **No separate frontend hosting** - Runs directly in Alma
- ✅ **Built-in authentication** - Users already logged in
- ✅ **Context-aware** - Knows current record/page user is viewing

What Cloud Apps CAN do:

- Access and manipulate data via Alma REST API
- Workflow shortcuts and automations
- Integration with external systems and APIs
- NZ API access via Cloud App Proxy

Real SLSP Examples

A few examples in our community:

- **Workflow automation** → SLSP CatExpand
- **Custom tools & reports** → Bib Hierarchy, Print Slip Report, Copy User Roles
- **External system integration** → SLSP to 7DM, SLSKey, SLSPmails (all their own backends)
- **Network Zone API access** → SLSP Card
- **Info from other IZs** → SLSP Rapido Cloud App

🔗 See all SLSP Cloud Apps: resources/links.md

Limitations to Keep in Mind

What Cloud Apps CANNOT do:

- ❌ Modify Alma's main UI (navigation, forms, MDE, etc.)
- ❌ Limited to data accessible via Alma REST API
- ❌ Perform batch operations (max 10 concurrent calls)
- ❌ Run background jobs or scheduled tasks

3. Technical Foundation

Technical Framework

Built on:

- **Angular 18** (HTML + TypeScript)
- **Material Components** as design components
- **RxJS** for reactive programming, async data streams
- **Cloud App SDK** library (`@exlibris/exl-cloudapp-angular-lib`)

Key Principle:

Apps interact with Alma through dedicated SDK services

Cloud App SDK & CLI

What is it?

- Official development toolkit for building Alma Cloud Apps
- CLI tool + Angular library (`@exlibris/exl-cloudapp-angular-lib`)
- Provides starter app, local dev server, and build tools

Maintained by:

- Ex Libris Group (official support)
- Open source on GitHub
- Officially described as receiving updates twice a year

We'll use it in the hands-on session!

📖 SDK on GitHub: github.com/ExLibrisGroup/exl-cloudapp-cli

4. Development Essentials

4a. Angular & RxJS

Angular Fundamentals

Application structures:

- **Components** - The fundamental building blocks (UI + logic)
  - **Template** - HTML
  - **Styles** - CSS/SCSS for appearance
  - **Class / Controller** - TypeScript logic
- **Services** - Reusable logic, shared state
  - Services are injected into components via Angular **Dependency Injection**
  - Dependency Injection: Angular's way of providing services to components automatically

Change Detection & Reactivity

- Angular automatically updates the UI when data changes
- Use **Observables** for async data streams (e.g., API calls) -> RxJS

RxJS & Asynchronous Patterns

- Frontend is inherently asynchronous (API calls, user interactions)
- RxJS makes this manageable with consistent patterns

Key Concepts:

- **Observables** - Asynchronous data streams
- **Operators** - Transform data (map, filter, tap, catchError, takeUntilDestroyed)
- **Subscribe** - Execute and get results

You'll see RxJS everywhere:

- All Cloud App SDK services return Observables
- 📖 Learn more: learnrxjs.io/learn-rxjs/concepts/rxjs-primer

RxJS: The Pattern You'll Use

Think of Observables like a stream of events over time:

```
// 1. Start with a stream (Observable)
this.restService.call('/users/12345')

  // 2. Transform data with operators (optional)
  .pipe(
    map(user => user.full_name),         // Extract just the name
    catchError(error => of('Unknown'))  // Handle errors, of() creates Observable with fallback value
  )

  // 3. Subscribe to execute and get results
  .subscribe(name => {
    console.log(name);  // "John Doe"
  });
```

- **Key pattern:**
  - `pipe()` - Chain operators together
  - Operators - Transform, filter, handle errors
  - `subscribe()` - Actually execute the Observable

4b. Working with the SDK

Cloud App SDK Services Overview

The SDK provides **6 core services** for interacting with Alma:

1. **Events Service** - Page context & navigation
2. **Settings Service** - User-specific settings
3. **Configuration Service** - Institution-wide configuration per app
4. **Alert Service** - User notifications
5. **Store Service** - Local data storage
6. **REST Service** - Alma API calls

Each service is injected via Angular Dependency Injection

Events Service

Purpose: Access page context and control navigation

Key Methods:

- `onPageLoad()` - Subscribe to page changes
- `getInitData()` - Get logged in user info, institution, language
- `entities$` - Observable of current records user is viewing (e.g., ITEM, USER, BIB_MDS)
- `refreshPage()` / `home()` / `back()` - Navigation of Alma main UI, but limited!

Event Service

$entities: Recommended approach with proper cleanup:

```
export class MyComponent {
  private eventsService = inject(CloudAppEventsService);
  private alert = inject(CloudAppAlertService);
  private destroyRef = inject(DestroyRef);

  entities: Entity[] = []; // Entity type from SDK, holds current entities

  // ngOnInit lifecycle hook (called on component initialization)
  ngOnInit() {
    this.eventsService.entities$.pipe(
      takeUntilDestroyed(this.destroyRef), // Auto-cleanup on destroy
      tap(entities => this.entities = entities), // Side effect: assign to local variable
      catchError(error => {
        this.alert.error(error.message); // Handle errors
        return of([]);
      })
    ).subscribe(); // Keep subscribe empty - logic in operators
  }
}
```

Settings Service

Purpose: Store per-user preferences

Key Methods:

- `get()` / `set()` / `remove()` - Store user preferences
- Persisted in Alma per user + per app, across sessions/devices

Examples:

- UI preferences, favorites, last search, filter settings

Configuration Service

Purpose: Store institution-wide settings

Key Methods:

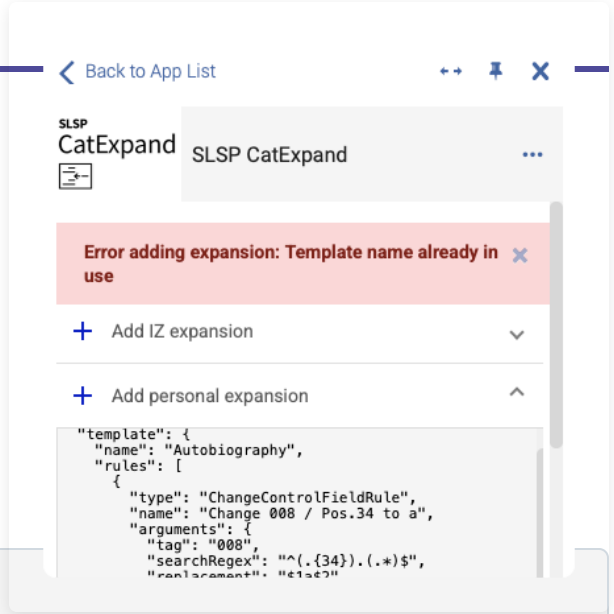
- `get()` / `set()` / `remove()` - Store app configuration
- Only users with admin roles can set, all users can read

Examples:

- API keys, default values, feature toggles

Alert Service

Purpose: Display messages to users



Methods:

- `success()`, `info()` , `warning()`, `error()`

  Example:

```
alertService.success('Item updated!');
alertService.error('Error adding expansion: ' + error.message,
  { autoClose: false });
```

📖 Alert Service: developers.exlibrisgroup.com/cloudapps/docs/api/alert-service

Store Service

Purpose: Local browser storage for temporary data

Key Features:

- Store temporary data in browser
- Not persisted across sessions/devices
- Useful for caching, temporary state

Remember:

- For persistent user data → use Settings Service
- For persistent config → use Configuration Service

REST Service

Purpose: Data retrieval and manipulation via Alma API

- Core functionality for most Cloud Apps
- Direct access to Alma data (items, users, loans, etc.)
- Enables CRUD operations on Alma resources

Key Features:

- **Automatic authentication** - Uses logged-in user's credentials
- **Permission-based** - User needs appropriate Alma roles
- **IZ API access only** - Accesses Institution Zone data
- **No governance impact** - Doesn't count toward limits

📖 REST Service: developers.exlibrisgroup.com/cloudapps/docs/api/rest-service
📖 Alma API: developers.exlibrisgroup.com/alma

Accessing Network Zone (NZ) API

Problem: REST Service only accesses **Institution Zone (IZ)** API

SLSP Use Case: Need access to **Network Zone (NZ)** data

- Examples: SLSP Card, SLSP CatExpand

Solution: Use of SLSP Cloud App **Proxy** for NZ API access

How it works:

- Proxy acts as external API endpoint
- App sends requests to proxy, which forwards to NZ API
- User roles are still checked by proxy

Using External APIs

Common Use Cases:

- External databases, web services, third-party integrations
- Data enrichment (covers, bibliographic data)
- **Custom backends** with database & scheduled jobs
  - Example: SLSP <> 7DM integration
  - Backend handles DB & batch operations, Cloud App provides UI

📖 External APIs: https://developers.exlibrisgroup.com/cloudapps/tutorials/outside/

Using External APIs: What You Need

Requirements:

- APIs must support **CORS** (Cross-Origin Resource Sharing)
  - May need backend proxy for CORS-restricted APIs
- You'll explicitly configure allowed domains in your app's configuration, it's a security requirement

We'll see how to configure this in the manifest.json section

4c. Configuration

Multi-Language Support

- Apps automatically use user's Alma language
- Translation files in `cloudapp/src/i18n/` (`en.json`, `de.json`, etc.)

In templates (HTML):

```
<button>{{ 'main.actionButtonLabel' | translate }}</button>
```

In TypeScript:

```
this.translate.instant('main.actionMessage')
```

Translation files:

```
// en.json: "actionButtonLabel": "Primary Action"
// de.json: "actionButtonLabel": "Primär-Aktion"
```

The manifest.json

What is it?

- Configuration file at the root of your Cloud App project
- Defines app metadata, behavior, and security settings

Key things you'll configure:

- **Basic metadata** - Title, subtitle, description, author
- **Entity types** - Which Alma pages your app appears on (ITEM, USER, BIB_MDS, etc.)
- **Security (CSP)** - External API connections, sandbox permissions
- **Widget settings** - Dashboard widget configuration
- **Institution restrictions** - Control which institutions can install (`relevantForInst`)

5. Publishing & Lifecycle

Cloud App Store & Publishing

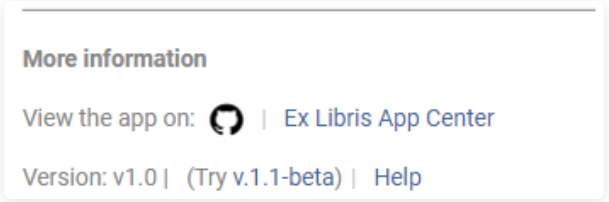
Process:

1. Build production version (`eca build`) and verify build is successful
2. Upload code to GitHub (public!) and create a release
3. Submit app to Ex Libris App Center (Developer Network)
4. Await review and approval
5. ... for updates, create new GitHub releases

Beta Versions & Testing

What are Beta versions?

- Pre-release versions for testing with real users
- Available alongside stable version
- Users can opt-in to beta testing



Benefits:

- Test new features before full release
- Gather feedback from real usage
- Safe rollback to stable version if issues arise

IZ Restrictions

What are IZ Restrictions?

- Control which institutions can install your app
- Set with `relevantForInst` field in `manifest.json`
- App won't appear in App Center for other institutions

Use Cases:

- **SLSP-specific apps** - Restrict to SLSP institutions only
- **Custom institutional apps** - Single institution only
- **Pilot programs** - Limit to participating institutions

Security Considerations

Understanding the Security Model:

- Cloud Apps introduce third-party code into Alma environment
- Apps run in sandboxed iframe with security restrictions
- Public apps reviewed by Ex Libris before initial publication
- Update review process is unclear - updates are deployed quickly

Transparency Requirements:

- Cloud Apps code must be open source (for public apps)
- External API connections defined in `manifest.json` and clearly visible in App Center

Security: Risks & Protection

What malicious apps could do:

- **Data exfiltration** - Steal patron data, circulation history, send externally
- **Data manipulation** - Alter records, loans, fines via API
- **Phishing** - Fake login forms inside Alma

How to Protect:

- ✅ Only allow apps from trusted sources in your IZ
- ✅ Review source code & manifest.json before installation and updates

6. Reference & Resources

Helpful Resources

Official Documentation:

- Cloud Apps Docs: developers.exlibrisgroup.com/cloudapps
- Alma API Docs: developers.exlibrisgroup.com/alma
- App Center Examples: developers.exlibrisgroup.com/appcenter

Our Workshop Repository:

- This presentation
- Development setup instructions
- Template Cloud App project (starter code)

Hands-on Time

Let's build something together!

Remember: This is collaborative - ask questions, share ideas!

Prerequisites & Setup

1. **IDE Setup:**
   - Use your preferred IDE or **recommended: VS Code**
2. **Get the Workshop Repository:**
   - **Option A:** Git (recommended)

```
git clone https://github.com/Swiss-Library-Service-Platform/slsp-cloudapps-resources
```

   - **Option B:** Download ZIP
     - Go to: https://github.com/Swiss-Library-Service-Platform/slsp-cloudapps-resources
     - Click "Code" → "Download ZIP"