Oneboard

databricks

Data Intelligence Platform Integration

ARCHITECTURE
DATA SECURITY
CONFIGURATION

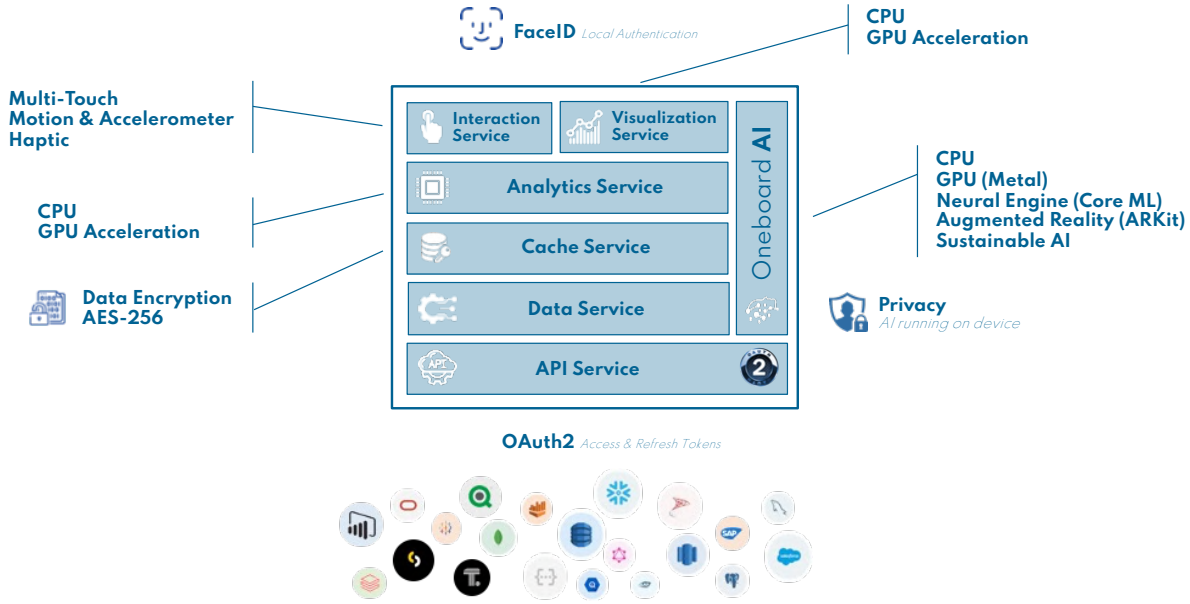
June 2024
Version 1.0

sweeft
REIMAGINING ANALYTICS

Oneboard x Databricks
Architecture Document

Overview

Oneboard is a native iOS application that connects to Databricks Platform through its API using OAuth2 for authentication. This setup ensures secure and auditable access to Databricks resources.



Components

- Oneboard iOS App: The native iOS application that users interact with.
- Databricks Platform: The cloud-based data intelligence platform where data is stored and analyzed.
- OAuth U2M Authentication: The method used to authenticate users securely.

Workflow

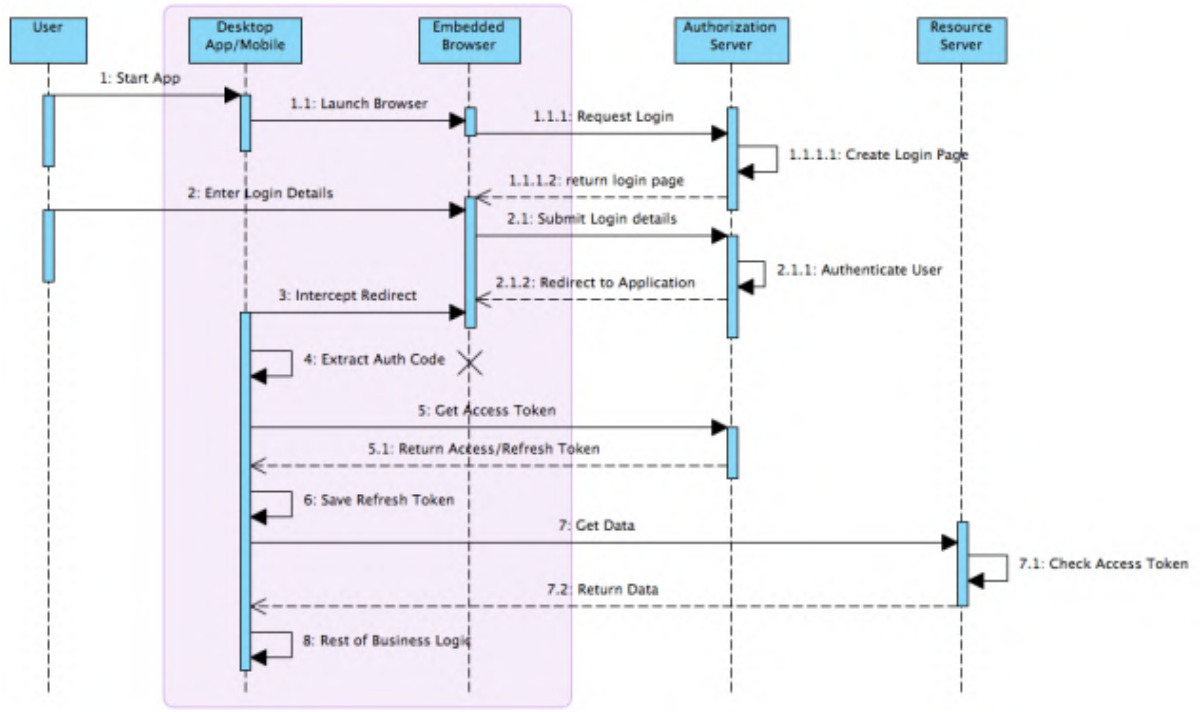
- User Authentication
  - Users authenticate via an OAuth flow, entering their Databricks credentials.
  - OAuth tokens are used for all interactions with Databricks Platform, ensuring secure access.

- Token Management
  - OAuth tokens are stored securely within the app.
  - Tokens expire periodically, prompting users to re-authenticate.

- Data Access
  - The app uses OAuth tokens to access Databricks Data.
  - Role-based access controls are enforced using Databricks permissions.

- Auditing and Security
  - Administrators can audit user connections to the Databricks Platform.
  - Authorization can be revoked for specific users through Databricks.

OAuth Workflow



1. The User starts the app. The app launches a browser component which is directed to the 'request login' endpoint of the Authorization Server. The Authorization Server generates and returns a login page.
2. The user enters their details and submits the page back to the Authorization Server. The Authorization Server authenticates the user, generates an authentication code and sends a redirects back to the embedded browser, instructing it to redirect to the application callback endpoint (which, of course, doesn't actually exist!)
3. The application intercepts the browser redirect (remember it is in an embedded browser component)
4. The application extracts the authentication code and terminates the browser component.
5. The application makes an HTTPS call to the Authorization Server, passing across the authentication code and gets back an Access Token and a Refresh Token.
6. The Refresh Token is saved locally (in the most secure way possible)
7. The application makes an HTTPS call to the Resource (FHIR) Server, passing across the Access Token in the 'Authorization' header. The Resource server validates the token and returns the data
8. The application performs any other required business logic.

One thing we haven't talked about is dealing with an expired Access Token.
Access Tokens have a limited life span (commonly an hour – but that's up to the Authorization Server) so what happens if the application is in the middle of doing something when it expires? Well, the flow is something like this:

1. The application makes a request (or an update) on the Resource Server.
2. The Resource Server recognizes that the token has expired and rejects the request.
3. The application retrieves the Refresh Token from its local store and sends that to the Authorization Server, which issues a new Access Token (assuming that the Refresh Token is valid of course).
4. The application re-tries the request to the Resource Server with the new Access Token.

Data Security and Encryption

- In-Transit Encryption
  - TLS Encryption: All data transmitted between the Oneboard app and Databricks Platform is encrypted using Transport Layer Security (TLS) to prevent eavesdropping and tampering.
  - It ensures privacy and data integrity between two communicating applications. TLS encrypts the data transmitted between the client (mobile app) and the server, preventing eavesdropping, tampering, and message forgery.
  - HTTPS uses TLS (or its predecessor, SSL) to encrypt the communication between the client and the server. When you see "https://" in a URL, it indicates that the connection is secured using TLS.
  - When an API call is made, it uses HTTPS, it means that the communication between the client and the API server is encrypted using TLS. This ensures that data transmitted via the API is secure from interception and tampering.

- At-Rest Encryption:
  - Databricks Platform Storage: Data stored in Databricks Platform is encrypted to protect against unauthorized access.
  - Device Storage: Data stored locally on the device, including cache data, is encrypted using AES-256 with SQLCipher 3 to ensure that even if the device is compromised, the data remains secure.
  - Tokens: They are stored locally on the device encrypted using AES-256 with SQLCipher 3 to ensure that even if the device is compromised, the data remains secure.

- Cache Data Encryption:
  - AES-256 Encryption: Cache data on the device is encrypted using AES-256 bits with SQLCipher 3. This ensures that any sensitive data temporarily stored on the device remains secure.
  - Secure Key Management: Encryption keys are securely managed and stored, with access restricted to authorized processes only.

- SQLCipher 3.4:
  - SQLCipher provides transparent 256-bit AES encryption of database files. It is designed to secure SQLite databases by encrypting the entire database file, making it a popular choice for mobile and embedded applications that require secure data storage.
  - Performance: SQLCipher is optimized for performance, with minimal overhead compared to standard SQLite operations.
  - SQLCipher is widely used in applications where data security is paramount, such as mobile apps that handle sensitive user information, financial data, or healthcare records. By encrypting the entire database, SQLCipher ensures that even if the database file is accessed by an unauthorized party, the data remains secure.

Databricks Platform Configuration

Prerequisites
- An active Databricks account and workspace
- The admin access to your Databricks account
- [OAuth U2M] IDP Integration is in place with the user either synced or created in Databricks

Steps to Configure OAuth in Databricks Platform

1. Create OAuth Application in Databricks:
   - Login to Databricks Account Console
     - AWS: https://accounts.cloud.databricks.com
     - Azure: https://accounts.azuredatabricks.net
     - GCP: https://accounts.gcp.databricks.com
   - Goto Settings | App connections
   - Click “Add connection”
   - Enter the application name : “Oneboard”
   - Enter the Redirect URLs : “https://sweeft.ai/oauth”
   - Select Access Scope : “SQL”
   - Select : Generate a client secret
   - Select a Value for Access Token TTL
   - Select a Value for Refresh Token TTL
   - Click “Add” to create your OAuth application

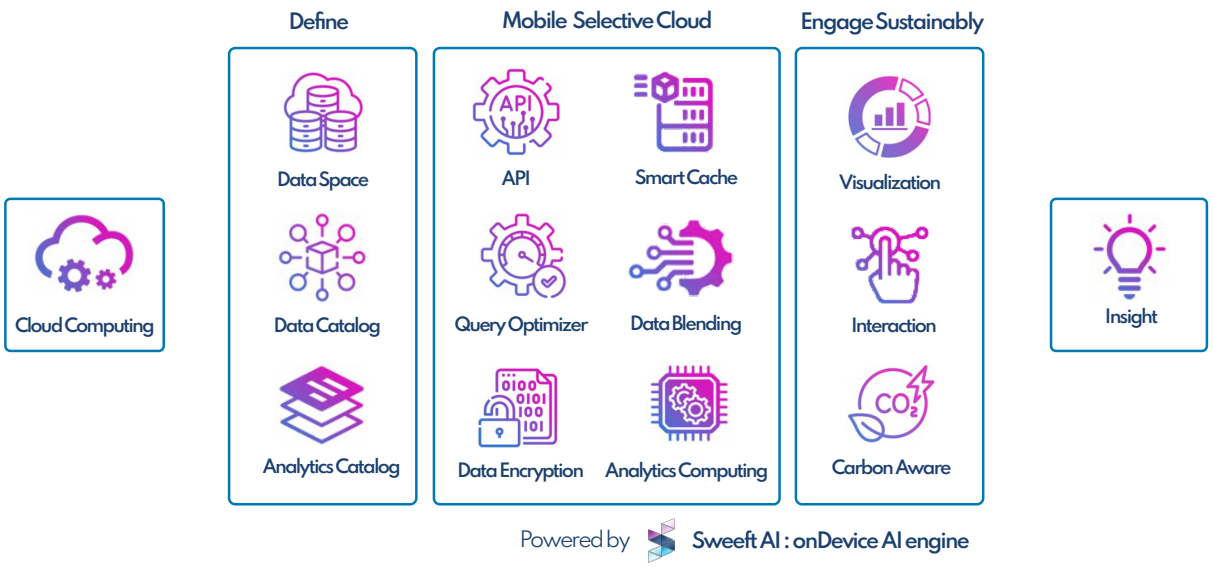
2. Retrieve OAuth Client ID and Client Secret :
   - After creating the configuration, retrieve the OAuth Client ID and the Client Secret
   - Make sure to copy the secret now. You won't be able to see it again.

Steps to Configure Oneboard Compute Warehouse in Databricks

1. Create SQL warehouse
   - Login to Databricks Platform
     - AWS : https://<deployement>.cloud.databricks.com
   - Goto Compute | SQL warehouses
   - Click “Create SQL warehouse”
   - Enter the SQL warehouse name : “Oneboard Warehouse”
   - Select the Cluster Size
   - Select Auto stop Value
   - Select Scaling Factor
   - Select Type : Serverless
   - Click “Create”

2. Retrieve SQL warehouse ID :
   - After creating the warehouse, retrieve the ID

Oneboard



Define Data Space : Connect Oneboard to Databricks Platform

1. Open Oneboard App:
   - Tap on the (+) icon in the Tab bar, then select "Connections".

2. Create New Databricks Connection:
   - In the connections screen, tap on "New Databricks Connection".
   - Enter the following details:
   - Display Name: Name of your Databricks Connection
   - Server URL: URL of your Databricks deployement
   - Client ID: The Client ID retrieved from the Databricks configuration
   - Client Secret: The Client Secret retrieved from the Databricks configuration
   - Warehouse: The SQL Warehouse ID retrieved from the Databricks configuration

3. Authenticate:
   - Follow the OAuth authentication flow to log in with your Databricks credentials.

By following these steps, you can securely connect your Oneboard iOS app to Databricks Platform using OAuth, ensuring robust access control and auditing capabilities.

Define Data Catalog : Set Databricks Table Metadata

1. Open Oneboard App:
   - Tap on the (+) icon in the Tab bar, then select "Data Analytics".

2. Select Databricks Table:
   - Tap on the (+) card on the top, then select "Databricks" Connection Name.
   - Select a Databricks Catalog
   - Select a Databricks Schema
   - Select a Databricks Table

3. Select Databricks Table Metadata:
   - Select Dimensions
   - Select Measures
   - Select Fields

4. Save Oneboard Data Analytics :
   - Click “OK”
   - Select a Name
   - Click “OK”

Define Analytics Catalog : Design a Dashboard on Databricks

The Workflow is not described as it’s more part of Oneboard Dashboard creation rather than Databricks integration.

To keep in mind, when users design a Dashboard, it’s only selecting the Oneboard Data Catalog Metadata.

When users interact with Live Databricks Data, it’s using the Oneboard Data Catalog Metadata to generate the SQL Statement in real time and optimized it through our Mobile Selective Cloud technology.