IBM DocumentHub

Lightweight content management system, which stores documents in GitHub, in a human friendly way

- Content Architecture
- Repository Structure
- Content Examples
- Catalog Structure
- Subcatalogs
- Access Control Lists (ACL)
- Attachments
- Documents Structure

Content Architecture

LIBRARIES

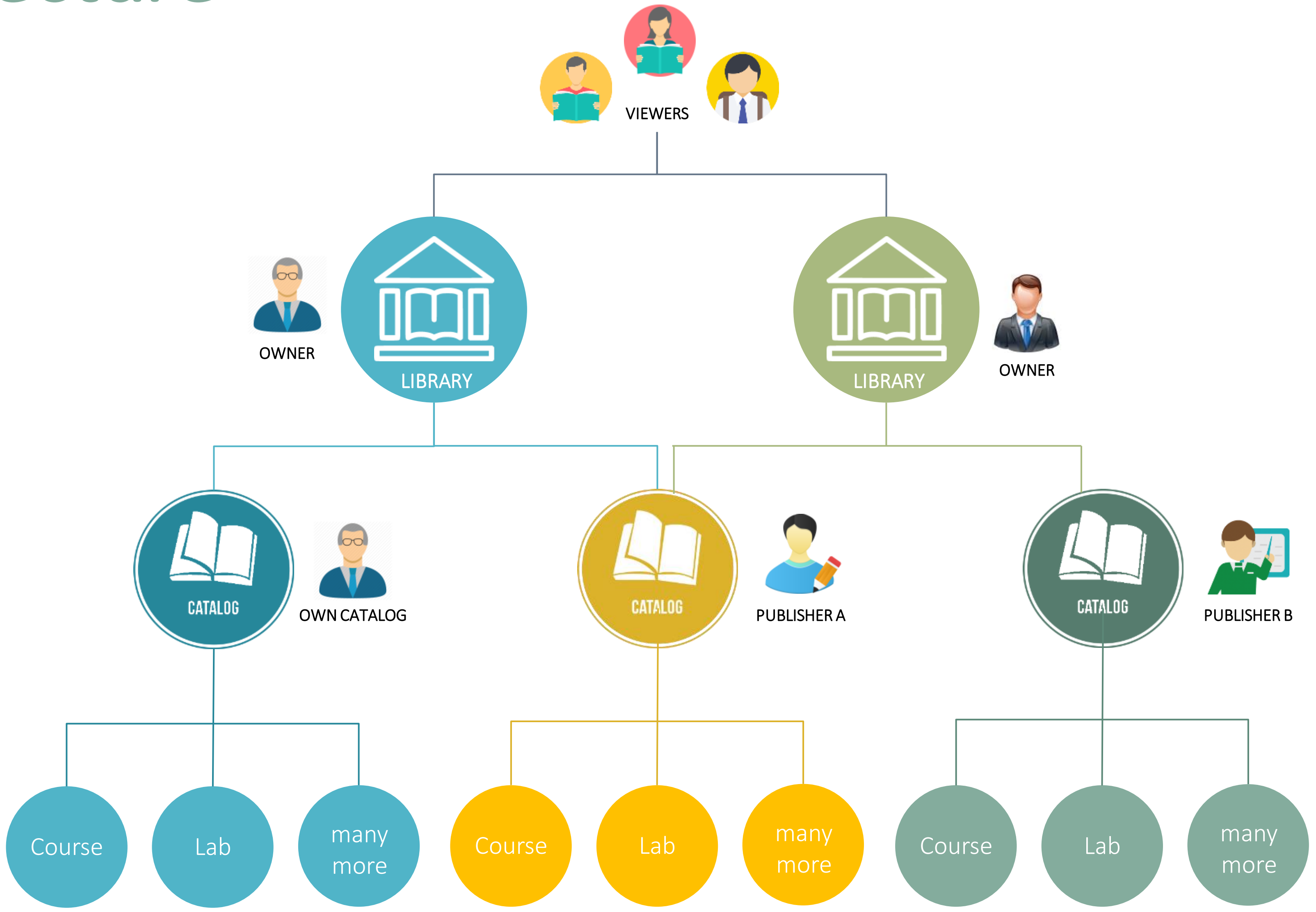
Libraries are showing catalogs from different publishers. A library can include your own catalogs or catalogs from other publishers. Usually, a library is a website.

CATALOGS

A catalog is a collection of documents. The publisher (catalog owner) decides the copyright for the catalog content.

DOCUMENTS

A document can be a course, lab, quiz or any other type of content.



Content architecture

Libraries

A library is a collection of catalogs. Catalogs in a library can be accessed by authorizing with the library id and secret. It can contain your own catalogs or catalogs from other publishers

Library credentials

Library ID and secret

Access permissions

- **Owners** and **Viewers**

Catalogs

- **Published** catalogs – can be seen by all viewers
- **Not published** catalogs – can be seen only by the owner

1-n

Repositories

GitHub repository

Live branch

Staging branch

Preview branch

A **branch** corresponds to a **physical catalog**

1-1

Catalogs

A catalog is a collection of documents. A physical catalog corresponds to a GitHub branch.

Catalog info

JSON data defined in the catalog

Catalog attachments

Attachments defined at the catalog level

Catalog files

Attachments defined at the catalog level

Access permissions

- Manage access roles: owners, creators, editors, viewers and previewers

Documents info

Documents data is **defined outside the documents**, at the catalog level

Physical or logical

A **physical catalog** corresponds to a GitHub branch. A **logical catalog** includes a selection of documents from the physical catalog

1-n

Documents

A document is a reusable piece of content. It corresponds to a folder in a GitHub branch.

Language

Language of the document

Document metadata

JSON data defined in the document

Content

Html, markdown or any other files.

Files can be organized in a flat or nested structure.

Attachments

Any **text or binary files**.

Are **automatically uploaded to Cloud Object Storage**.

Are **accessible on public links**, without authentication.

Akamai CDN is already configured for all public attachments.

Access permissions

- Manage access roles: owners, creators, editors, viewers and previewers

Repository structure

- GitHub repository
  - live
  - staging
  - preview
    - catalog.json
    - subcatalog a.json
    - subcatalog b.json
    - document1
    - document2 en
    - document3 es
    - document3 fr
    - _attachments
    - __files

branch = physical catalog

catalog.json = catalog metadata

subcatalog <name>.json = a logical catalog which contains a subset of documents from the physical catalog

folder = document

folders starting with _ = public catalog files (will be uploaded to Cloud Object Storage and CDN)
folders starting with __ = private catalog files (accessible only through API)

Content Examples

Catalog of documents

staging branch

- _static
- artificial-intelligence ar
- artificial-intelligence en
- artificial-intelligence es
- cloud-computing ar
- cloud-computing en
- cloud-computing es
- kubernetes
- machine-learning ar
- machine-learning en
- machine-learning es
- acl Admins.json
- acl Users.json
- catalog.json
- subcatalog default.json
- subcatalog Free.json
- subcatalog Middle East.json
- subcatalog US.json

Public static files are **automatically uploaded to Cloud Object Storage**. They are accessible on **public links**, without authorization. **Akamai CDN** is already configured for them.

Documents can have **multiple languages**

Access Control Lists can be used for the main catalog and subcatalogs.

Main catalog contains all documents from that branch

Subcatalogs contain a subset of documents

Document

artificial-intelligence folder

- _static
- 01 Getting Started.md
- 01.01 Introduction.md
- 01.02 Prerequisites.md
- 02 Machine Learning
- 02.01 Machine learning algorithms.md
- 02.02 Machine learning components.md
- 02.03 Epilogue.md
- 03 Assessment
- 03.01 Quiz.md
- document.json
- toc.json

Public static files, automatically uploaded to Cloud Object Storage, accessible on **public links** with **Akamai CDN** already configured.

Document content can be html, markdown or any other text format

Quizzes can be added as part of the content

Document json metadata

More Content Examples

Website config and content files

- _images
- config de
- config en
- config es
- config fr
- content de
- content en
- content es
- content fr
- catalog.json

Public static files are automatically **uploaded to Cloud Object Storage**. They are accessible on **public links**, without authorization. **Akamai CDN** is already configured for all public static files.

Documents containing **json** or **text** files, in **multiple languages**. Accessible on **public links** or with **authenticated API** requests.

Website json metadata

Advantages of keeping the website config and content files in GitHub & Edge CaaS:

- easy to be maintained by the content team
- changes are immediately available in the staging or production website
- don't need to redeploy application for every config or text change
- Akamai CDN is already configured for images and any other attachments

JSON only documents

- _static
- document A en
- document A fr
- document A pt
- document B en
- document B fr
- document B pt
- acl Managers.json
- catalog.json
- subcatalog Free.json
- subcatalog Paid.json

Public static files, automatically **uploaded to Cloud Object Storage**. **Akamai CDN** is already configured for them.

Documents with content in **multiple languages**

Access Control Lists

Subcatalogs with different access to content

GitHub & Edge CaaS advantages over a No SQL database:

- JSON files are managed in GitHub which is more friendly than a database UI
- Easy to create a content team with different roles in GitHub UI
- GitHub UI can be used instead of an admin module in your application
- The content is safe with a very good changes tracking
- Changes history, change differences and it's easy to revert mistakes
- Dev, staging and production branches for content
- Full text search and filtering by any JSON field
- Cloud Object Storage and Akamai CDN for static files
- Access Control Lists

Catalog structure

GitHub Catalog

A GitHub catalog corresponds to a GitHub branch.

Catalog parameters are defined in **catalog.json**

Access Control Lists (ACL)

Lists of permissions for libraries and users.

"**owners**": { "library1": ["admin@gmail.com"] },
"**editors**": { "library1": ["all"] },
"**viewers**": { "library2": ["*ibm.com", "somebody@gmail.com"] },
"**previewers**": { }

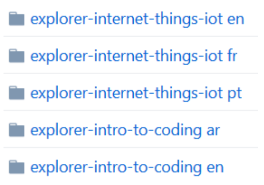
Metadata

Catalog metadata can be defined in the "info" field.

"**catalogInfo**": { }

Documents

A **document** corresponds to one **folder** from the branch.
Naming convention: **documentId language**



explorer-internet-things-iot en
explorer-internet-things-iot fr
explorer-internet-things-iot pt
explorer-intro-to-coding ar
explorer-intro-to-coding en

Documents info

JSON data located in the file: documents.json

```
"documentsInfo": {
  "doc1": {
    "image": "course.jpg"
  }
```

Catalog attachments

Folders starting with _

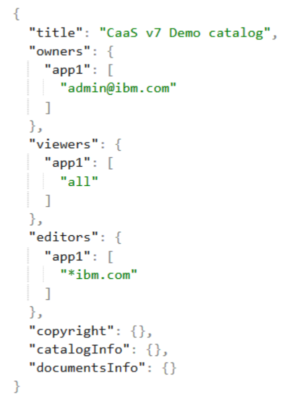
All files under folders starting with _ will be automatically uploaded to Cloud Object Storage and will be publicly available on https://developer.ibm.com/caas-storage/{catalogID}/{folder}/{file}

Catalog private files

Folders starting with __

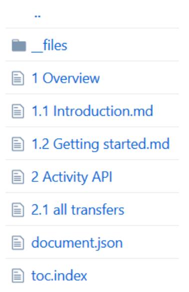
All files under folders starting with __ are accessible only through API

catalog.json example:



```
{
  "title": "CaaS v7 Demo catalog",
  "owners": {
    "app1": [
      "admin@ibm.com"
    ]
  },
  "viewers": {
    "app1": [
      "all"
    ]
  },
  "editors": {
    "app1": [
      "*ibm.com"
    ]
  },
  "copyright": {},
  "catalogInfo": {},
  "documentsInfo": {}
}
```

document example:



..
__files
1 Overview
1.1 Introduction.md
1.2 Getting started.md
2 Activity API
2.1 all transfers
document.json
toc.index

Subcatalogs

Subcatalogs are logical catalogs which **contain a subset of documents** from the physical catalog.

Subcatalog example:

subcatalog Test.json

```
{
  "title": "Subcatalog Abc",
  "owners": {
    "app1": [
      "all"
    ]
  },
  "viewers": {},
  "editors": {},
  "copyright": {},
  "catalogInfo": {},
  "documents": [
    "api",
    "artificial-intelligence-v2",
    "blockchain-fundamentals",
    "chatbots-v3"
  ],
  "documentsInfo": {
    "api": {
      "topic": "web_application",
      "journey": "new_collar"
    },
    "artificial-intelligence-v2": {
      "topic": "artificial_intelligence",
      "journey": "new_collar"
    }
  }
}
```

Same fields as a physical catalog (catalog.json)

List of the documents from the physical catalog which are included in the subcatalog

Documents info are inherited from the physical catalog, but can be overridden in subcatalog

Access Control Lists (ACL)

Access control lists can be defined for **catalogs** and **subcatalogs**. The access control lists contains the lists of libraries and the lists of users from each library which have access to documents. The list of users can contain emails, email patterns and access control lists.

There are **4 levels of access:**

- **owners** – Full access on the catalog - can view and edit the catalog and all it's documents
- **creators** – Can create documents
- **editors** – Can view and edit documents
- **viewers** – Can view documents
- **previewers** - Can view only documents metadata but not the content

ACL examples:

"**owners**": { "library1": ["admin@gmail.com"] },

User admin@gmail.com from library1 has owner level access

"**editors**": { "library1": ["john@ibm.com", "mary@ibm.com"] },

Users john@ibm.com and mary@ibm.com from library1 have editor level access

"**viewers**": { "library2": ["*ibm.com", "acl ClassA"] },

All ibm.com user and the users listed in acl ClassA.json, from library2 have view access

"**previewers**": { "library3": ["all"], "contentstore": [acl ContentStore"] }

All users from library3 have preview access

---

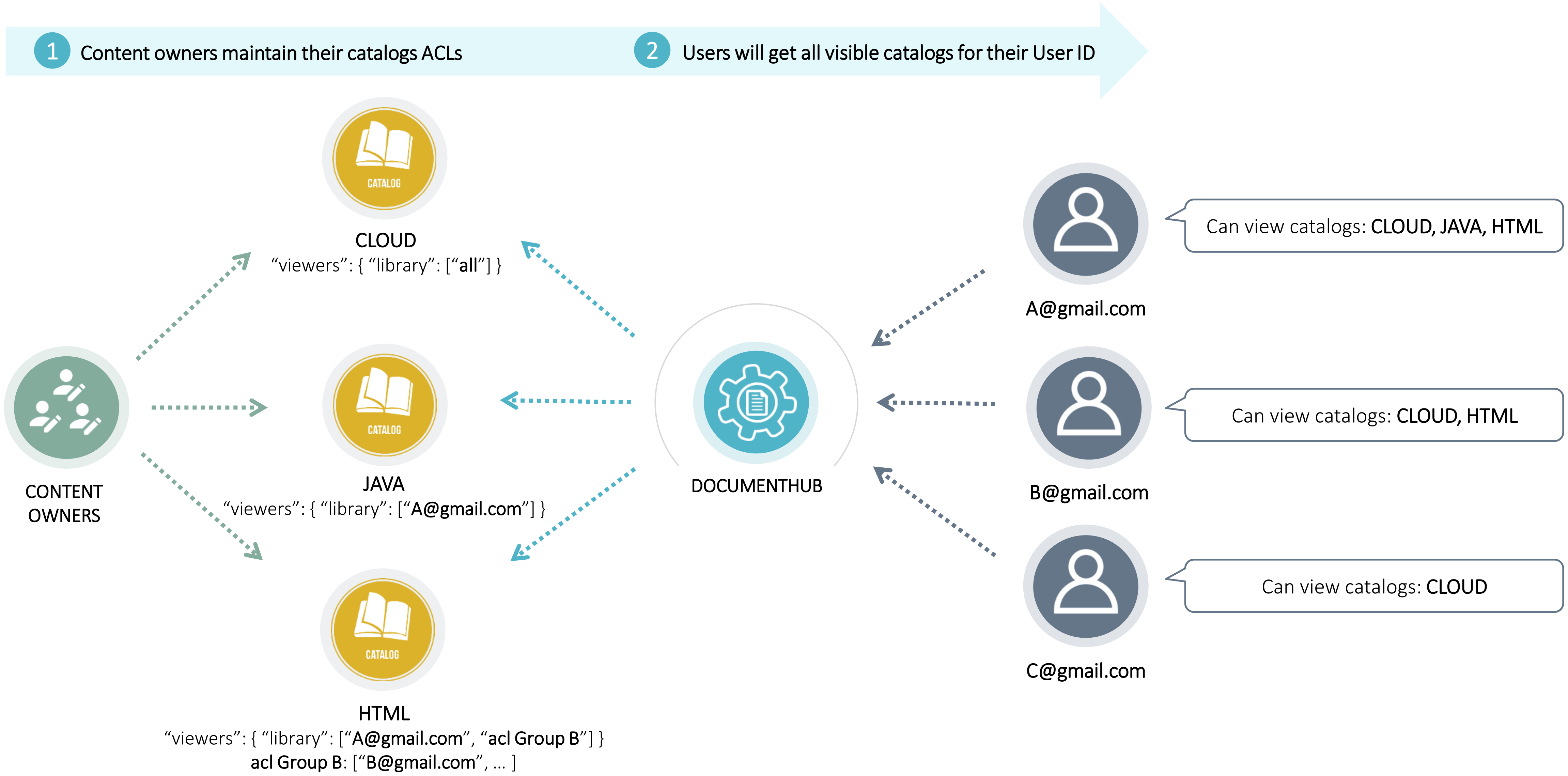
acl classA.json

["user1@gmail.com", "user2@gmail.com"]

acl ContentStore.json

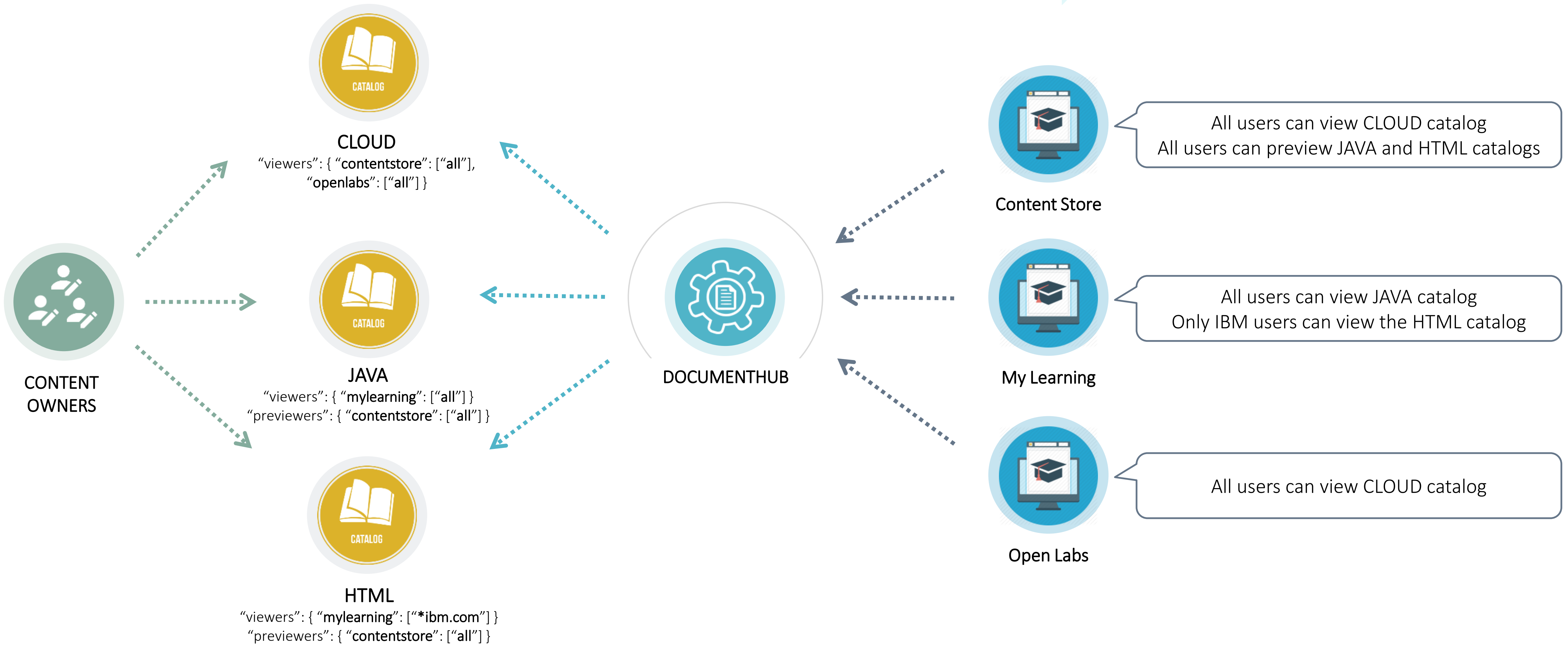
["userA@gmail.com", "userB@gmail.com"]

How ACL works for users


1 Content owners maintain their catalogs ACLs
2 Users will get all visible catalogs for their User ID
CATALOG
CLOUD
"viewers": { "library": ["all"] }
CATALOG
JAVA
"viewers": { "library": ["A@gmail.com"] }
CATALOG
HTML
"viewers": { "library": ["A@gmail.com", "acl Group B"] }
acl Group B: ["B@gmail.com", …]
CONTENT OWNERS
DOCUMENTHUB
A@gmail.com
Can view catalogs: CLOUD, JAVA, HTML
B@gmail.com
Can view catalogs: CLOUD, HTML
C@gmail.com
Can view catalogs: CLOUD

How ACL works for libraries

1. Content owners maintain their catalogs ACLs
2. Libraries will get all visible catalogs for their library ID



CLOUD
“viewers”: { “contentstore”: [“all”], “openlabs”: [“all”] }

JAVA
“viewers”: { “mylearning”: [“all”] }
“previewers”: { “contentstore”: [“all”] }

HTML
“viewers”: { “mylearning”: [“*ibm.com”] }
“previewers”: { “contentstore”: [“all”] }

Content Store:
All users can view CLOUD catalog
All users can preview JAVA and HTML catalogs

My Learning:
All users can view JAVA catalog
Only IBM users can view the HTML catalog

Open Labs:
All users can view CLOUD catalog

Attachments

Folders starting with _ (underscore) are **public attachments** folders

All files found in the public attachments folders are automatically uploaded to **Cloud Object Storage**.

Akamai CDN is already configured for all public attachments.

The files path to Cloud Object Storage will be:

- catalog attachments
  - CDN: https://dw1.s81c.com/caas-storage/{catalogID}/{attachmentsfolder}/{filename}
  - no cache: https://developer.ibm.com/caas-storage/{catalogID}/{attachmentsfolder}/{filename}
- document attachments
  - CDN: https://dw1.s81c.com/caas-storage/{catalogID}/{documentid}/{lang}/{attachmentsfolder}/{filename}
  - no cache: https://developer.ibm.com/caas-storage/{catalogID}/{documentid}/{lang}/{attachmentsfolder}/{filename}

Examples:

- https://dw1.s81c.com/caas-storage/008/_attachments/image1.jpg
- https://dw1.s81c.com/caas-storage/IcCVI6RO/documentation/en/_attachments/architecture5.png

The **attachmentsBaseUrl** and **cdnAttachmentsBaseUrl** are returned in catalogs and documents json response.

Folders starting with __ (two underscores) are **private attachments** folders

Files from the private attachments folders can be downloaded only with authorized API requests.

Document structure

Document free structure

- document.json
- Module A
  - Lesson A1.md
  - Lesson A2.html
- Module B.json
- Lesson B1.md
- _attachments

Document free structure with ToC

- document.json
- toc.json
- Module A
  - Lesson A1.md
- Module B.json
- Lesson B1.md
- _attachments

Document indexed flat structure

- document.json
- toc.index
- 1 Module A
- 1.1 Lesson A1.html
- 2 Module B
- 2.1 Lesson B1.md
- _attachments
- __files

Document indexed nested structure

- document.json
- toc.index
- 1 Module A
  - 1 Lesson A1.html
- 2 Module B
  - 1 Lesson B1.md
- _attachments
- __files

Documents

A **document** corresponds to one **folder** from the branch.
Naming convention is: **documentId language**

A document contains:

Metadata

- **document.json** file

Content

- **HTML, Markdown** or any other **text** files
- Supports 4 types of structure:

Free structure

- organized in a structure of files and folders (max 2 levels)
- no naming convention
- files can be accessed by the path

Free structure with ToC

- organized in a structure of files and folders (max 2 levels)
- no naming convention
- table of contents is automatically generated based on toc.json
- files can be accessed by the path or index

Indexed flat structure

- a single level structure of files
- naming convention: index name.ext
- table of contents is automatically generated (any number of levels)
- files can be accessed by path or index

Indexed nested structure

- organized in a structure of files and folders (max 2 levels)
- naming convention: index name.ext
- table of contents is automatically generated
- files can be accessed by path or index

Attachments

- Public attachments: any folder starting with _
- Private files: any folder starting with __