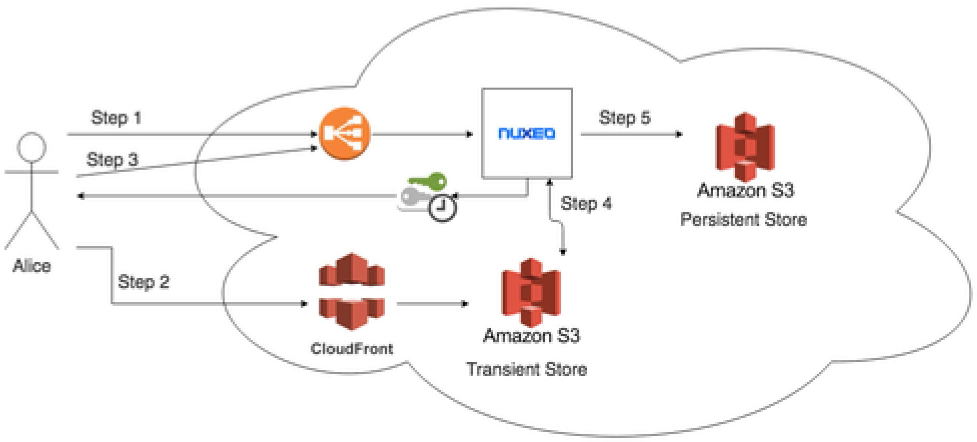
Solution - S3 Direct Upload

Architecture



Steps

1. Alice wants to upload a file
2. Nuxeo Server returns Temporary Credentials along with S3 Bucket information (Base Key, Bucket Name) to do the upload
3. Alice uploads file directly to S3
    a. Alice communicates to Nuxeo Server that the upload is completed in Bucket X with Key Y, File Size Z, Mimetype W
4. Nuxeo Server performs a consistency check on the upload (File Size, Bucket, Key, Mimetype) and renames the file name to match the ETag (Re-use S3BinaryManager)
5. On Blob.AttachOnDocument, Nuxeo Server copies the file over to the Persistent Store

Discovered Limitations

The Batch Upload mechanism doesn't support uploading to other sources than the Local File System.

Implementation

We applied the Fundamental Theorem of Software Engineering, so we added a new layer called Batch Upload Handler, which will allow to customise upload handling via a contribution on BatchManager, called **BatchUploadHandler**. To preserve the previous behaviour we encapsulated the former algorithm in a **DefaultBatchHandler** so that we still achieve backwards-compatibility.

We also added several endpoints to the `/upload` endpoint to support this new *"layer"*

New Endpoints

Method	Endpoint	Parameters	Description
GET	/upload/handlers		Lists all the registered Upload Handlers
POST	/upload/new/{provider}	Provider - The unique identifier of the handler to use for this batch upload	Created a new Batch bound to a certain Upload Handler
POST	/upload/{batchId}/{fileIndex}/complete	BatchId - The Id of the Batch File Index - Index of the file within the Batch	Notifies Upload Handler that an upload is completed. After that, the file consistency is double-checked by the handler and renamed to it's ETag so that we can re-use S3BinaryManager

GET	/upload/{batchId}/info	BatchId - The id of the Batch	This endpoint will now return a JSON Object along with custom Extra Information to be filled by the Handler. In S3 Handler case, it will return STS Credentials and Bucket Information on the **extraInfo** field

The /complete Endpoint will do a kind of a "hack", but it isn't a hack. It will simply check the existence of the file, and rename it to it's ETag so we can re-use the S3BinaryManager to fetch the Blobs from S3.

We needed to do a "little hack" where if the file was uploaded via MultipartUpload, the resulting ETag is an MD5 of each Part MD5 with a dash character `-` and the number of parts in the upload. After renaming (CopyObject) it will assume a different ETag if CopyMultiPartObject wasn't used.