Media Cloud and Microservice Architecture

Jump start your media infrastructure in the cloud

https://tech.ebu.ch/groups/mcma
Why MCMA?

• Cloud infrastructure is complex and overwhelming
• Simple things are simple, scaling is difficult, management is challenging
• Cloud is not optimized for Media workflow
• Media services interactions are specialized
• Cloud providers services are structured differently
• Cloud underline processing services come with great power and require great responsibilities
• Cost of services becomes an issue for rigid infrastructures
• Cloud services functionalities and capabilities change rapidly
Why MCMA-AI?

• There is a lot of expectations around Artificial Intelligence
• Artificial Intelligence is directly associated to “micro-services”
• MCMA wants to take benefit of the AI hype to further promote the use of service-based architectures, building on earlier FIMS results

MCMA’s message is clear:

You may be rightly attracted by AI but it will be useless if not integrated in your operational workflows!
OPINIONATED TOOLING

• Several universities make the claim that, when deciding where to put sidewalks, they first let students wear paths through the grass. This told them where to pave and ensured the best use of their walkways.

• You can think of these well worn paths in cloud architecture as a procedure, or design, that gets repeated over and over again, to the point that it should just become boilerplate. Rather than everyone composing the same 99% of code, we can generate that code, and focus on the 1% that is unique.

• Opinionated tooling is designed to guide you down a path that is considered a best practice. Additionally, since best practice is the default, the amount of unique code we maintain is dramatically reduced.
Take control of your infrastructure

- Choose **where**:
  - Select your cloud provider(s)
  - Same design model can be applied to your on-premise infrastructure

- Choose **what**:
  - Select between different services to meet a cost or performance goal
  - Expand capabilities by easily leveraging new services

- Choose **how**:
  - Provide choice of development technology (Node.js, DotNet Core)
  - Leverage Open Source component for configuration and deployment

- Choose **who**:
  - Let non cloud experts start with enterprise ready infrastructure
  - Only implement what is specific to your scenario

- Choose **when**:
  - Enable you to simply expand and provision your infrastructure as needed
Design considerations

- Limit learning curve
- Share resources like reusable code
- Repeatable patterns
- Easily configure and deploy
- Developer friendly
- Follow opensource mentality
Best practices and implementation

• Job management and monitoring
• Service sequencing
• Long running transactions support
• Event based communication pattern
• Service registration and discovery
• Standardized REST based data repository
Sample AI-centered infrastructure

上传媒体 → 检查媒体 → 对齐媒体 (AWS 转码、FFMPEG、Lambda) → 或者 → 对齐媒体 (AWS 转码、FFMPEG、Docker EC2) → 分析媒体 (AWS 翻译、rekognition、Video 索引器)

分析媒体

AWS 翻译、rekognition、Video 索引器

MCMA 擅长多平台工作流整合！
MCMA Common Service Pattern

- REST Interface
- API Handler
- Worker
- MCMA Libraries

**Forward HTTP requests**
**Validate and route HTTP requests**
**Handle interactions with datastore and other components**
**Process requests**

Custom Service, API, library, exe
Basic Information on MCMA

FIMS > MCMA

- MCMA builds on, and continues, the services developed as part of the Framework for Interoperable Media Services (FIMS).
- MCMA is a project of the EBU Strategic Programme on Media Information Management and Artificial Intelligence.

Objectives

- Develop a set of simplified REST APIs with minimum payload to allow integrating workflows combining microservices in the cloud with other in-house services and processes.
- Share libraries containing glue code between the high level APIs and low level (e.g. cloud) platforms.

>> The demonstration using the REST APIs and the libraries will be shown here!
Cognito app – AWS speech to text

Credits: Eurovision Song Contest
Cognito app – Azure celebrity detection

Credits: Eurovision Song Contest
Cognito app – AWS translation to Japanese

Credits: World Economic Forum
AWS step functions, workflow and state machine
Media Services Environment (now)

Azure
- Face recog
- Object recog

Your organization
- Shot/Scene Detection
- Your system
- Media & metadata

AWS
- Extract Thumbnail
- Transform

GCP
- Face recog
- Speech to Text
- Translate En to Ja
- Translate Ja to Kr

Individuals
- Link to DPpedia
- Named Entity
Media Services Environment (with MCMA)

- **Azure**
  - Face recog
  - Object recog

- **Your organization**
  - Shot/Scene Detection
  - You can switch!

- **AWS**
  - Extract Thumbnail
  - Transform

- **GCP**
  - Face recog
  - Speech to Text
  - Translate En to Ja
  - Translate Ja to Kr

- **Individuals**
  - Link to DPpedia
  - Named Entity

You can switch!
MCMA Github Repository

**ebu/mcma-libraries**
https://github.com/ebu/mcma-libraries

**core libraries** (Node.js, .NET, Java etc.)
- Contains the model (e.g. classes, types, message contents) and helper functions and utilities to create and communicate properly formed messages
- mcma-core-nodejs (mcma-core.js)

**cloud provider specific libraries** (AWS, Azure, GCP etc.)
- Abstracts the particularities of a specific cloud to a predefined REST
- mcma-aws-nodejs (dynamo-db-table.js, rest-controllers.js)

**ebu/mcma-projects**
https://github.com/ebu/mcma-projects

**multi-cloud-ai-workflow**
- Demonstrates how you can leverage AI technologies from multiple cloud vendors in a single media workflow

Note that as the both are still work in progress, please checkout the development branch for the current status.
Toward to Cloud-native system

Conventional environment

Physical Server
- CMS
- HDD
- DB

CMS
- Speech 2 text
- Face Recog

Content

High cost!

Virtualize as it is

Pay as you go

Server less architecture

Switch easily

Cloud environment

VM Server
- Cloud provider specific library
- Core library

AWS Lambda
- AWS API Gateway
- AWS Step Functions

AWS S3
- AWS Dynamo

FaceA

FaceB

Content
Multi-cloud-ai workflow

How to start

1. Clone this repository to your local harddrive
2. Navigate to the multi-cloud-ai-workflow folder.
3. Create a file named gradle.properties
4. Add the AWS account information to the created file & save the file.
5. Open command line in multi-cloud-ai-workflow folder.
6. Execute “gradlew deploy” and let it run.
7. If no errors, you have successfully setup the infrastructure in your AWS cloud. Go to https://aws.amazon.com/console/ and sign in to see your cloud infrastructure.

file: gradle.properties
environmentName=com.your-domain.mcma
environmentType=dev
awsAccountId=<YOUR_AWS_ACCOUNT_ID>
awsAccessKey=<YOUR_AWS_ACCESS_KEY>
awsSecretKey=<YOUR_AWS_SECRET_KEY>
awsRegion=<YOUR_AWS_REGION>
What you will have after deployment

**AWS**
- S3:website
- S3:upload
- S3:repo
- S3:temp

**Step Functions**
- Step 1
- Step X
- Conform workflow

**API Gateway**
- Service Repository
- Job-repo
- Job-proc
- AME
- Transform
- Media-repo
- Workflow

**Lambda Function**

**DynamoDB**

**Azure**
- Transcript
- Celebrities

**V**
- Upload
- Kick

---

Celbrities

---

Transcript

---

Translate

---

Celebrities

---
Some keywords helping you to know more details...

1. Terraform
2. Gradle
3. Node.js
4. AWS: S3
5. AWS: Lambda
6. AWS: Step Functions
7. AWS: DynamoDB
8. AWS: API Gateway
9. AWS: Transcribe, Translate, Rekognition
10. EBU Core: BMContent, BMEssence
11. Azure: Videointerpreter
Service Registry

- Stores information about all available services on a REST endpoint
  - https://service-registry/services
  - Allows GET, POST, PUT and DELETE operations.
- Each other service only needs to know this URL to be able to find any other resource within the deployed infrastructure.
Job Repository

- Stores jobs that need to be executed on a REST endpoint
  - AIJob, AmeJob, CaptureJob, TransferJob, TransformJob, WorkflowJob
  - Allows GET, POST and DELETE operations
- On arrival of a new Job the service instantiates a JobProcess and sends it to the Job Processor Service
Job Processor Service

- Stores JobProcesses that manage the execution of jobs
- On arrival of a new JobProcess the service searches for a service that is capable of executing this job and sends a JobAssignment to this service
Message Flow Execution of AmeJob

Step Function 3
- Step Function Callback
  - POST Notification on update
  - POST AmeJob

Step Function 4
- Job Repository
  - POST Notification on update

Step Function 5
- Job Processor Service
  - POST Notification on update
- AME Service
  - POST Notification on update
Loadmap

It would be great if we can have a loadmap with timeline, but due to less resources, we cannot show it for the moment …

But the direction from now is:
1. Extend sharable libraries to new platforms like Google, IBM…
2. Further work on best practices on content management
   • HighRes vs. LowRes, thumbnails, audio only…
   • Continue work started with File Catalyst on content transfer between platforms
3. Test more AI tools directly relevant to business
4. …
Join us!

MCMA is a voluntary collaborative effort

We need more resources to go faster and share more useful resources

If you are an architect, you certainly want to know more about service-based architectures (if not already convinced)

If you are a developer, system integrator, you will certainly appreciate working with other competent and motivated colleagues eager to share knowledge. This will make the learning curve smoother using best practice development software platforms.

How can you join:

1. Visit https://tech.ebu.ch/groups/mcma
2. Click on “join this group”
3. Alternatively contact EBU -> evain@ebu.ch