Architectural Solution

Overview

U.Group's solution leverages only open source frameworks and data to provide rich insights, linkages and content management of automatically curate dossier content from publicly available data sets. To extract maximum value from these data sets, we leveraged a series of data driven AI/ML techniques. U.Group's evolutionary architecture philosophy helped evolve our many models and data sources while maintaining the security and confidentiality of the data under investigation and the users accessing the system.

In addition to the functional requirements that were provided as part of the challenge, we also spent time focused on the non-functional requirements. As the nature of a Content Management Solutions is one in which data storage requirements grow over time, we leveraged products that would maintain the security and sensitive data management policies as a part of their inherit design. These realities shaped our architecture into one that leveraged the capabilities of the cloud while being cognizant of the cost implication through leveraging containerization, serverless architecture and security, and ensuring uninterrupted service to our customers. This is coupled with providing telemetry on a single pane of glass through the use of the ElasticSearch Kibana and Logstash stack to aid in diagnosis while providing a highly available fault tolerant architecture.

While evaluating the concerns of security, we chose to model our approach around the various states of data to ensure completeness of our solution.

1. **Data at Rest,** stored to a persistable device such as a distributed storage mesh. Within our solution we elected to leverage S3 with **versioning disabled**, leveraging Amazon's data deletion and recoverability guarantees. In addition to the perimeter security provided through the use of IAM policies the system is also designed to pass all day through an encryption gateway, visible outside of our ecosystem.

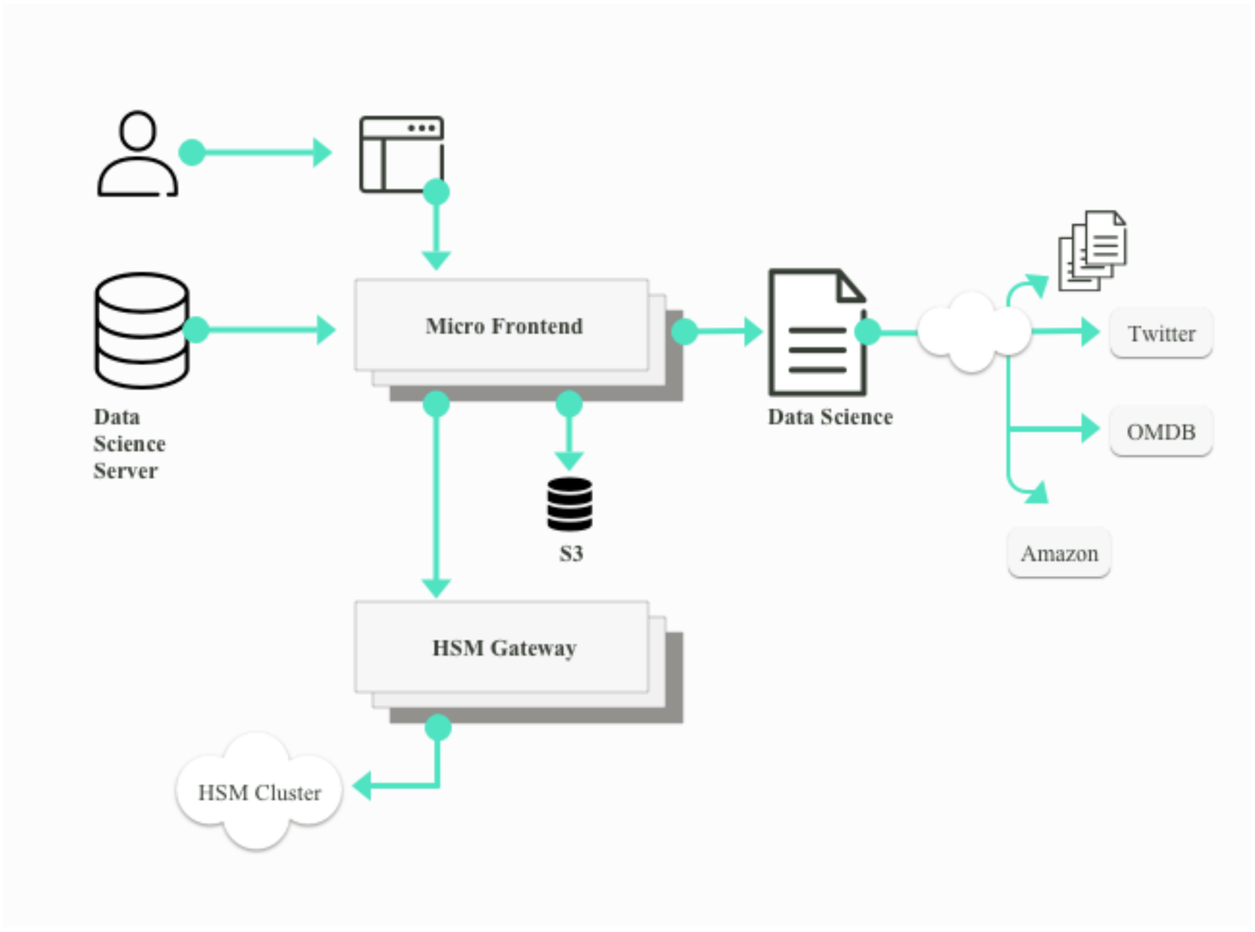
2. **Data in transit** captures the concern of data security while data is in motion between the data sources and the end location.

3. **Data in Use,** concern focuses around the means in which data can be accessed and the general security of the data in use.

For an added element of security we also chose to use a distributed encryption technique in which multiple keys were used (having a distinct key for every dossier) to reduce the surface area of attack. This design ensures that having access to a single user key would not compromise the entire system and limits the attack surface to individual documents rather than the entire store. This trade off was made consciously and supported by the strategic implementation of the AWS CloudHSM. U.Group's experience with Hardware Security Modules such as this product and the Luna HSM, aided in our design of this distributed key management that allows for an extended key space, minimizing the possibility of collisions while maintaining cryptographic performance.

High Level Architectural Approach

The solution architecture builds off of the architectural principles of Micro Service Design, while leveraging Cloud Centricity to achieve a highly scalable and testable system. These concerns enabled the creation of a pluggable abstraction that provides fault tolerance in a consistent and predictable fashion. These abstractions not only made the solution testable, they also added an additional level of flexibility to enable offline development through the use of Feature Toggles and Test Doubles.



Core User Personas

During our project kick off we conducted persona and feature mining sessions to better understand and prioritize the initiative. These sessions provided significant business and architectural insight and identified the following roles.

- **Business User** - SPIDER employee that would like to browse all of the dossiers and gather useful insights about movies and actors.
- **Business Supervisor** - SPIDER supervisor that has similar user experience needs to the Business User with additional capabilities around delete.
- **Data Scientist** - Individuals performing exploration and analysis of various data sets to better understand correlations, preferably in an interactive console such as Jupyter.
- **Developers** - Individuals creating the fully integrated production environment inclusive of all of the components and assets that have been created.
- **DevOps Engineers** - Individuals automating the deployment and High Availability characteristics of the environment.
- **Site Reliability & DevSecOps Engineers** - Individuals primarily concerned with the stability and security of the system and require instrumentation and telemetry.
- **Product Owners** - Individuals directing the delivery and development of Features and Prioritization.

Data Processing & Engineering Approach

We have created a horizontally scalable subsystem that is designed to ingest data sources as our data needs grow. In addition to practicing Agile Delivery from a business perspective, we have included the necessary capabilities and components into our solution to enable Agile Delivery from a Technical Perspective. As the scope of the challenge is fairly substantial, we elected to design flexible interfaces rather than rigid data structures that assumed the structure of the data to be ingested while maintaining flexibility.

During our data analysis, we found significant value leveraging a series of models for various functional requirements. The following items provide insights into the code techniques:

1. Entity Detection and Relationship analysis using Natural Language Processing, most specifically Parts of Speech Tagging.

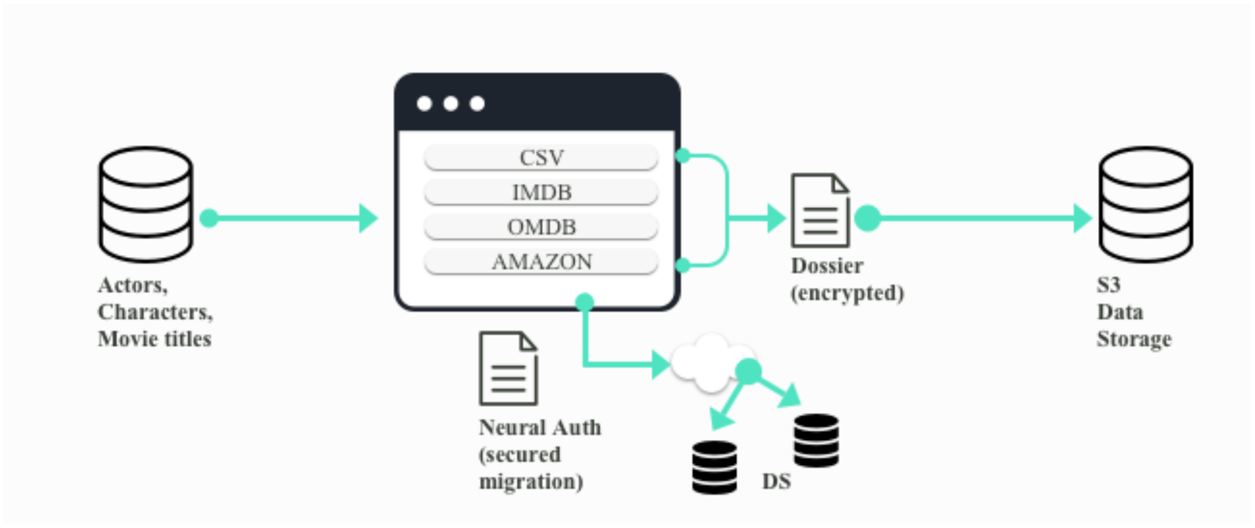
2. Genre fit sampling using various clustering mechanisms that leveraged reviewer sentiment in addition to information provided by various data sources.
3. Term Frequency Inverse Document Frequency (TF-IDF) to understand the significance of various terms in movies and better understand the similarities of given films to suggest additional dossiers that may be of interest to a given user.
4. Levenshtein Distance, to assist with entity resolution on items such as names and movie names.

Architectural Components
We have created a dynamic architecture comprised of 5 distinct components that can leverage the scale of the cloud and Amazon services to compute and classify large amounts of data. This capability is enabled by cloud native horizontal scaling architectures while promoting resiliency through a High Availability ecosystem. The strategy allows us to provide a level of fault tolerance and resilience leveraging the elastic nature of the cloud and our architectural selections.

- Content Management
- Data Orchestration and Ingestion
- Enhanced Search and Discovery
- End-User Security and Telemetry
- User Experience and Data Visualizations

Content Management

Responsible for Document Storage and Encryption. The system models document security as a first class citizen that leverages dependency injection to quickly swap encryption methods. Architecturally, we have designed our persistence as an abstractable component with an initial implementation of S3 because of the Object Security and Delete guarantees.



<u>**Core Technologies Used**</u>
Persistence - S3 for raw data persistence and Elastic Search for indexable document state management.
Search - ElasticSearch for dynamic multi-dimensional search with variable matching parameters.
Encryption - AWS CloudHSM for key management accompanied by TLS for transport security.
Client Presentation - ReactJS for client presentation and User Experience.

Data Orchestration and Unification Subsystem
Data Orchestration allows us to combine various data sources through a series of data science techniques. This subsystem is responsible for our intelligent rationalization of entity relationships as well as the basis for the AI/ML data sense making micro-services.

Enhanced Search and Discovery
After identifying elements of our data set that we deem sensitive, our system was automatically able to generate a searchable profile that retained multi-dimensional entity level relationships to allow easy navigability through our data graph.

End-User Security and Telemetry
Stateless Security Management through the use of industry standard JSON Web Token (JWT) signed with security materials provided at runtime at memory to each of the instances within a given cluster. This strategy was chosen to enable Role Based Access Control seamlessly and manage the user level audits throughout the system.

Data Ingestion and Transformation

This subsystem is responsible for the seamless integration of additional data sources in an efficient fashion. The system has been designed to support data from multiple data source so this abstraction allows us to create canonical models to standardize each entity in a similar fashion to techniques leveraged for NIEM compliance (https://www.niem.gov/).

CI/CD and Cloud Native Support

U.Group's solution makes use of Amazon web service through an automated pipeline orchestrated with Terraform as a configuration management tool. As we are provisioning Docker Containers, we have the ability to seamlessly provision workloads within the cloud leveraging AWS' Platform as a Service implementation (Elastic Container Service) to handle our process scheduling. As we have designed the architecture to be cloud native, we also have achieved uninterrupted deployments with our dynamically scalable and redundant architecture. Additionally, all data capture and model training is executed as part of our CI process to enable the consistent versioning and traceability of all assets that make their way to production. At no time will our team deploy assets into production without leveraging our CI process.

Finally, to ensure high availability each of our services are deployed in a minimum of 2 availability zones which enables uninterrupted service. This, along with Elastic Load Balancers, provides an unparalleled level of redundancy within the platform. All of our services, including our models, provide a series of telemetry to Cloud Watch which can be used to get insight into transactions that are inflight and additionally to measure model performance over time.

Technologies and Frameworks Overview

The majority of the applications components are written in a combination of Python and Java to handle our Data Science and Data Engineering needs respectively. This choice of technology was chosen because of their extensive toolkits for operationalizing data models along with providing the required telemetry to evaluate model performance overtime as well as integration into the AWS capabilities. Each of the respective sections have outlined the technologies that are significant to a specific tier within the application.

Significant Component List:
ElasticSearch
AWS CloudHSM

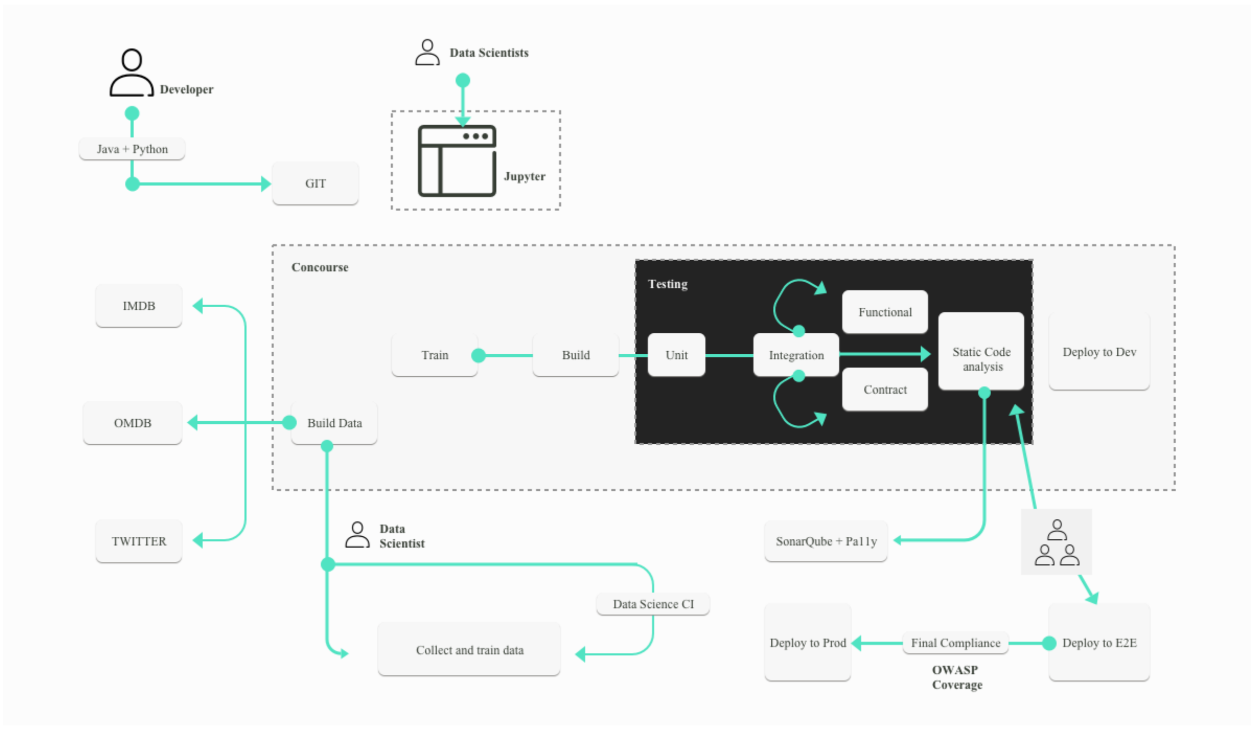
Elastic Container Service
ElasticSearch Logstash and Kibana
Terraform
Concourse
SpringBoot
Spacy

As the solution is designed to be cloud native which includes the criteria of supporting continuous delivery of binaries, we have included a few open source frameworks with the following capabilities.

Data Sources Used

- OMDB
- IMDB Public Data Sets
- Twitter
- Wikipedia
- Amazon

End to End Architecture and Engineering Culture



Pre-requisites for solution installation:

1. **AWS Account**

Although, most of the configuration will automatically be generated using our docker images, you will need to have access to an Amazon AWS account with the respective AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY to enable secured communication with Amazon. These keys will be required for provisioning an instance of Continuous Integration along with the dev, test, and production environments.

Note: The provided deployment script is a very large script that provisions the project CI/CD infrastructure, data, Jupyter notebook and results UI. It should only be executed on a clean slate account and environment. If you want to re-execute the script, please use a clean slate AWS account.

Note: For simplicity, we recommend using the terminal to execute each of the commands to recreate a consistent, reproducible environment where possible. Ensure that you have access to all AWS services, including S3.

2. **Clone Git Repo (use either Option 1 or Option 2)**

Option 1:

Install Git

Go to the following sites to download and install Git:

Git Mac installation
https://www.atlassian.com/git/tutorials/install-git#mac-os-x

Git Windows installation
https://www.atlassian.com/git/tutorials/install-git#windows

To clone our source repo, you will need a pre-configured destination Git repo.

Clone on Mac:

a. Make sure that git is installed and in the system's path
    · Open a Terminal, and run the following command: **git —version**
    · It should return something like **git version 2.15.0**
b. Run git clone https://github.com/ByteCubed/ugroup-records-submission
c. Enter the credentials for IRNSDD-Demo2 GitHub account
d. Run cd ugroup-records-submission to get into the project directory

Clone on Windows:

a. Make sure that git is installed and in the system's path

· Open a Command Prompt (or Git Bash if during installation you elected not to use Git from the Windows Command Prompt), and run the following command: **git —version**

· It should return something like **git version 2.15.0**

b. Run **git clone** https://github.com/ByteCubed/ugroup-records-submission

c. Enter the credentials for IRNSDD-Demo2 GitHub account

d. Run cd ugroup-records-submission to get into the project directory

Option 2:
Any platform (Mac or Windows, no need to install git):

a. Go to https://github.com

b. Enter the credentials for IRNSDD-Demo2 GitHub account

c. Go to https://github.com/ByteCubed/ugroup-records-submission and click on the green button: "clone or download" and click on the "download zip" option.

d. Extract the zip file.

3. Install Docker (Community Edition)

The deployment configuration is controlled through Terraform, to ensure seamless delivery we have provided a Dockerfile with a docker container that includes all necessary tools to handle Windows or Mac deployments. However, docker will need to be pre-installed by the developer. Docker is containerization software that allows for easy configuration of a production or development environment through the scriptable installation of tools or other system dependencies. Our containers will only provide the tools required for installation but will leverage only the scripts that are a part of this repo.

Go to the following sites to download and install Docker:

Docker for Mac: Install the Stable Channel. Wait for Docker to start. (It may take some time depending on your machine specifications. Docker Whale logo turns Green to indicate that Docker has started)

https://docs.docker.com/v17.12/docker-for-mac/install/

Docker for Windows:

https://docs.docker.com/v17.12/docker-for-windows/install/

Note: In order to use Docker on Windows 10 Pro, virtualization must be enabled so that Docker can function properly. (check Virtualization enabled status by

going to task manager and clicking on Performance Tab and seeing the status of Virtualization). If it is not enabled, please go to https://docs.docker.com/docker-for-windows/troubleshoot/ and follow instructions under Virtualization

4. Configure Development Environment

Development for this project requires the tools specified in Figure 1 of this document to be installed and configured on your path. If this is the first time you are configuring this machine for software delivery, we recommend that you implement the following steps:

Mac OS X

- Ensure machine is configured with Administrative Permissions
- Install necessary **x-code xcode-select --install tools** to ensure that the required development libraries have been properly configured.
- Install Docker (See above)
- Install **Homebrew** which will help with the installation of the following packages

Windows 10

- Ensure machine is configured with Administrative Permissions
- Install Docker (See above)

Note, these steps are meant to accelerate the path to providing all the appropriate CI infrastructure that will configure a build pipeline and promote the most recent commits to test.