Software Architecture Project IV - Data Pipeline Storage Manager

Summary of the underlying technology stack

Frontend Application

Angular - The team decided to use the Angular framework for the user-facing web application, as it is an extensive framework that multiple of the team members already have experience with. It allows us to create a dynamic, responsive web page, and display information about the process of the data pipelines owned by the user.

Authorization

Keycloak - For authorization it was decided to use Keycloak, as it is open source and provides a good level of authentication.

Backend Application

Python and Flask - The backend will be built using Python and Flask. This decision was made as the Apache Airflow documentation indicated it could be easier to integrate with a Python backend than Java, as well as the fact that it is a widely used, simple backend language. Flask was chosen due to its simplicity.

MongoDB - For metadata storage we opted for MongoDB due to its document-style data storage, which allows flexible types of metadata to be stored.

Infrastructure Provisioning

Terraform - To provision infrastructure, we will use Terraform. Using Terraform ensures that we are working with the same infrastructure each time, as well as to easily deploy multiple instances of the data pipelines and required infrastructure, like s3 buckets and clusters.

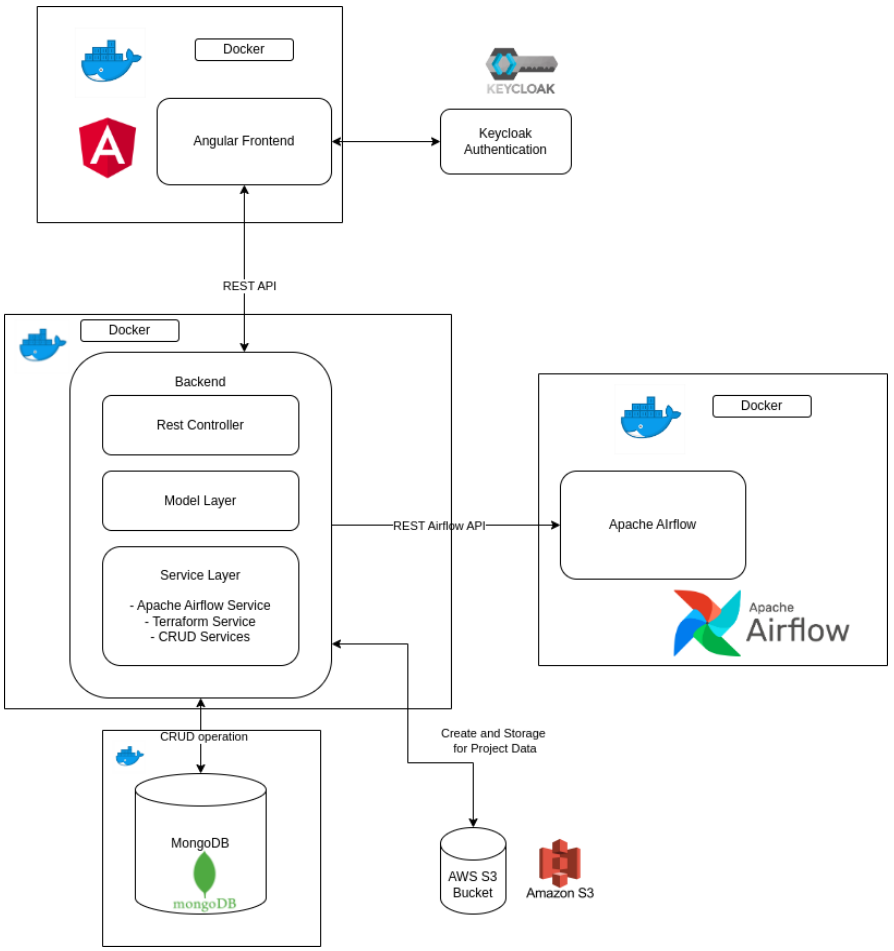
Data Pipeline Orchestration

Apache Airflow - Allows us to flexibly orchestrate data pipelines and schedule workflows by adding, editing or modifying tasks. Will allow the user to perform multiple types of data processing on the input data.

Containerization:

Docker, Kubernetes - In order to save deployment and run-up time for different technologies, it was decided to use containerization tool docker. It was kept in focus that each entity should be able to run independently. Hence, frontend, backend, data pipeline and database will be containerized as independent docker pods. For the application level of the whole software, Kubernetes will be used, as the main aim of the project is to deploy the whole application in EKS (*Elastic Kubernetes Service*) cluster on AWS cloud provider.

Overview Diagram of code components (static view)



Frontend

For frontend a TypeScript framework will be used called “Angular” using “npm” as the package manager.Frontend will be connected to the Keyclock for user authentication. Communication with the backend will be done over the REST-API.

Backend Application

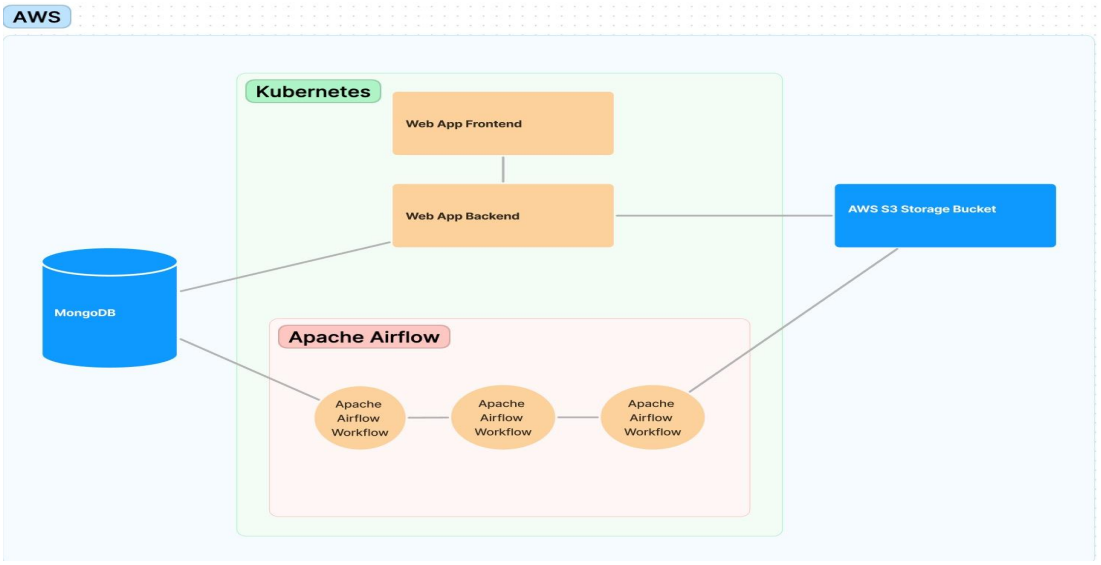
REST API for creation of entities. It has CRUD operations on project, data pipeline entities. And furthermore search and file upload functionality.

The service layer deals with the provisioning of the storage and controlling of the data pipelines. Also it manages the storage of the entities in the database. The model layer defines our work object entities.

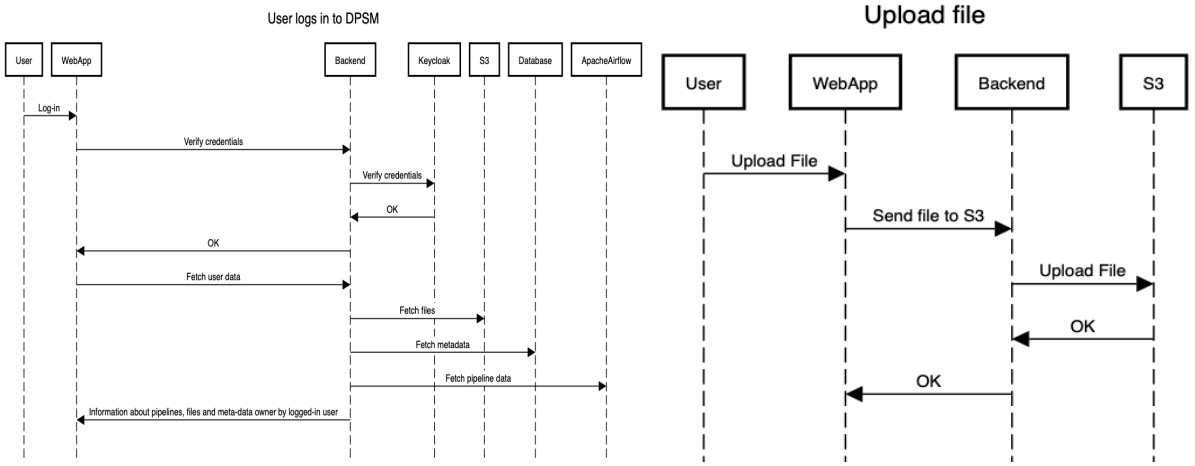
Data Pipeline

Apache airflow is the preferred technology for data pipelining as it offers a high level of extensibility, flexibility, and scalability. The Apache airflow is integrated through the backend database and S3 bucket on AWS. It allows tidying, filtering and processing data for the files uploaded. The results from data processing are then stored in MangoDB over the backend controller. The tasks or DAG (Direct acyclic graph) can be called with the help of Airflow API.

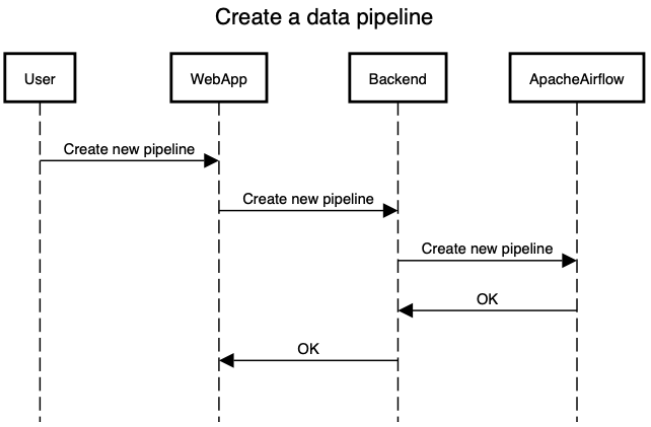
Overview Diagram of runtime components (static view)



Overview Diagram of runtime components (dynamic view)



Scenario 1 Scenario 2



Scenario 3

textual explanation of the diagrams and choices

Scenario 1 - File Upload

Description: In this scenario, a user initiates a file upload action through the WebApp. The WebApp communicates with the Backend, which in turn sends the file to the S3 storage service. Upon successful upload, S3 confirms the upload to the Backend, which responds with an "OK" status to the WebApp.

Scenario 2 - User Log-in

Description: When a user attempts to log in to the system via the WebApp, the WebApp sends the user's credentials to the Backend for verification. The Backend communicates with Keycloak, an identity and access management service, to verify the credentials. Upon successful verification, Keycloak sends confirmation back to the Backend, which responds to the WebApp with an "OK" status. Subsequently, the Backend fetches the user's data, retrieves files from S3, metadata from the Database, and pipeline data from Apache Airflow. The aggregated information about pipelines, files, and metadata belonging to the logged-in user is then sent back to the WebApp.

Scenario 3 - Create a Data Pipeline

Description: In this scenario, a user interacts with the WebApp to create a new data pipeline. The WebApp communicates with the Backend, which subsequently requests Apache Airflow to create the new pipeline. Upon successful creation, Apache Airflow responds with an "OK" status to the Backend, which, in turn, confirms the successful creation to WebApp.