ID2207 - Modern Methods in Software Engineering
DataCloud Project Report

Maragna Jacopo 20000305-7759

My background

I undertook this project with previous knowledge. In Particular, I had already worked extensively with Docker in my previous job as a junior Data/ML engineer. Regarding Argo, I've never worked with it before, but I've worked with KubeFlow which is a very similar tool.

The time I spent learning docker was roughly 16 hours, but it took me months to perfect it.

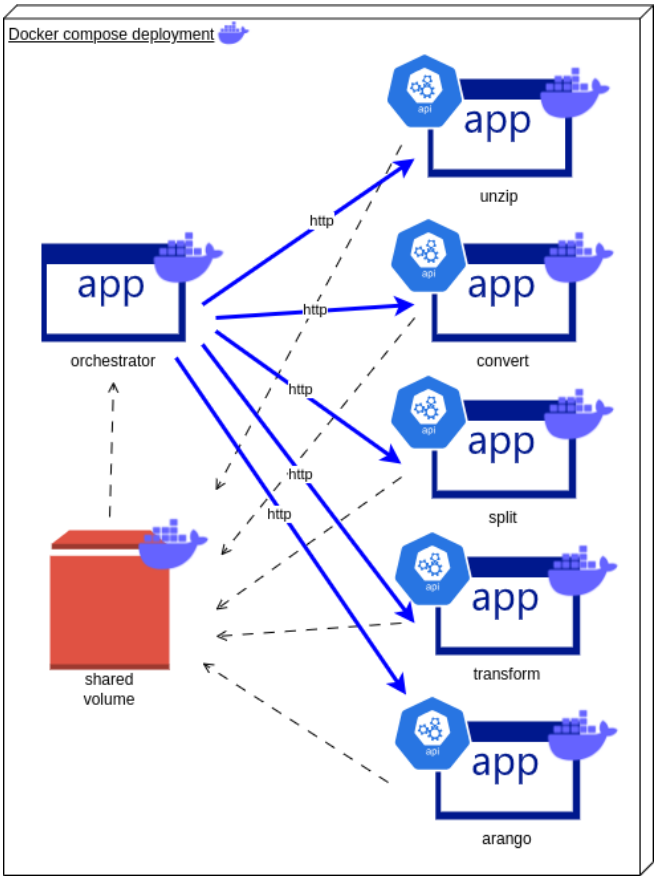
GitHub Repository

Reference to the GitHub repository where to find the DataCloud Project code:
https://github.com/jackma-00/DataCloudProject

Task 1

Manually describe components

Architecture



Basically solution number one has been implemented within a microservice paradigm and deployed in docker compose for the purpose of testing.

- Each step of the pipeline is a stand alone micro-service. The API receives http requests to process data. Once the service has done with data processing it saves the resulting data into the shared volume and returns the control to the caller.
- The orchestrator component coordinates the execution of the pipeline. It is responsible for invoking each microservice passing the correct data retrieved from the shared volume.

Pipeline description

1. Orchestrator invokes unzip service passing the compressed data. Unzip service extracts the data, saves the resulting artifact in the shared volume, and returns the control to the orchestrator.
2. Orchestrator retrieves the unzipped file from the shared volume and passes it to covert service. Convert service converts the data from TSV to CSV, saves the resulting artifact in the shared volume, and returns the control to the orchestrator.

3. Orchestrator retrieves the CSV file from the shared volume and passes it to split service. Split service splits the data, saves the resulting artifact in the shared volume, and returns the control to the orchestrator.
4. Orchestrator retrieves the splitted file from the shared volume and passes it to transform service. Transform service cleans the data, saves the resulting artifact in the shared volume, and returns the control to the orchestrator.
5. Orchestrator retrieves the transformed file from the shared volume and passes it to arango service. Arango service converts the data from CSV to Arango, saves the resulting artifact in the shared volume, and returns the control to the orchestrator.

Time assessment

Total time spent: 5 hours

- 30 minutes: system design.
- 1 hour: learning.
- 3:30 hours: evenly splitted between code writing and debugging.

Further improvements

Replace the Docker compose deployment with a AWS Lambda function deployment.
Transfer the shared volume to AWS S3 Bucket.
The enhanced solution would guarantee high scalability entirely provided by the serverless Lambda service.
Include CICD workflows to enable a DataOps system.

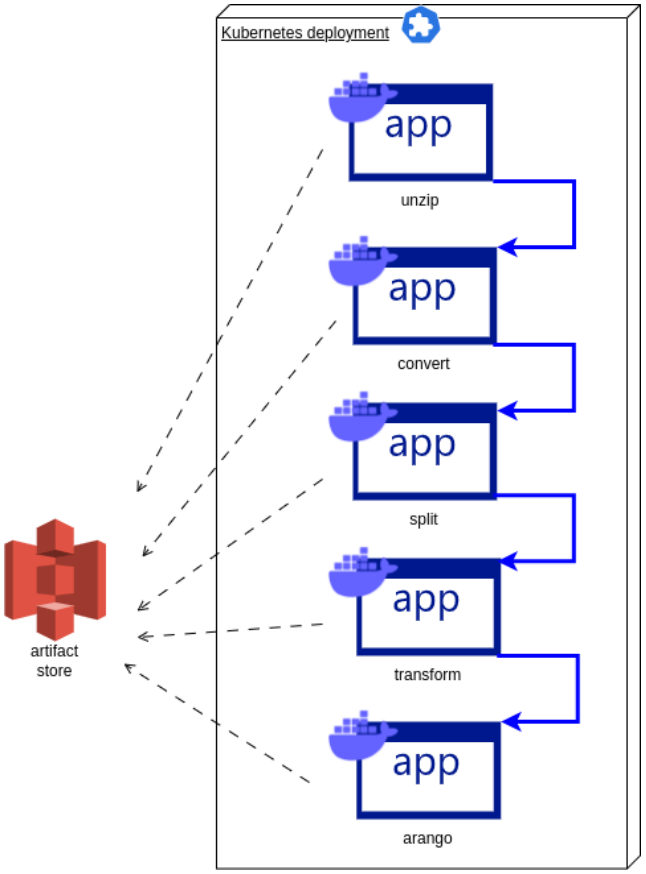
Estimated time: 5-8 hours including:

- Adapting the code to work within the AWS environment.
- Writing IAC infrastructure with AWS CDK (optional).
- Writing CICD workflows and setting up GitHub actions.

Task 2

Describe the pipeline using Argo Workflows

Architecture



Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes.

- Each step in the workflow is a container. The logic is the same. However, the API service is no more needed since here we leverage Argo's Artifact technology.
  - Docker images have been published to Docker Hub in order to be freely retrieved by Argo Workflows for execution.
- Each step generates an artifact (the processed file). The output artifact of one step is used as input artifacts to a subsequent step.
- To store artifacts between steps we leverage an external S3 compatible data store such as MinIO.
- An external artifact store is very convenient when it comes to trace all the intermediate artifacts especially in a DataOps paradigm. However, a simpler solution would be implementing a shared volume where to store intermediate files.

Pipeline description

Steps are the same as the previous pipeline, if not that here we don’t have any orchestrator component that needs to invoke the services but we leverage Argo itself as an orchestrator.

Time assessment

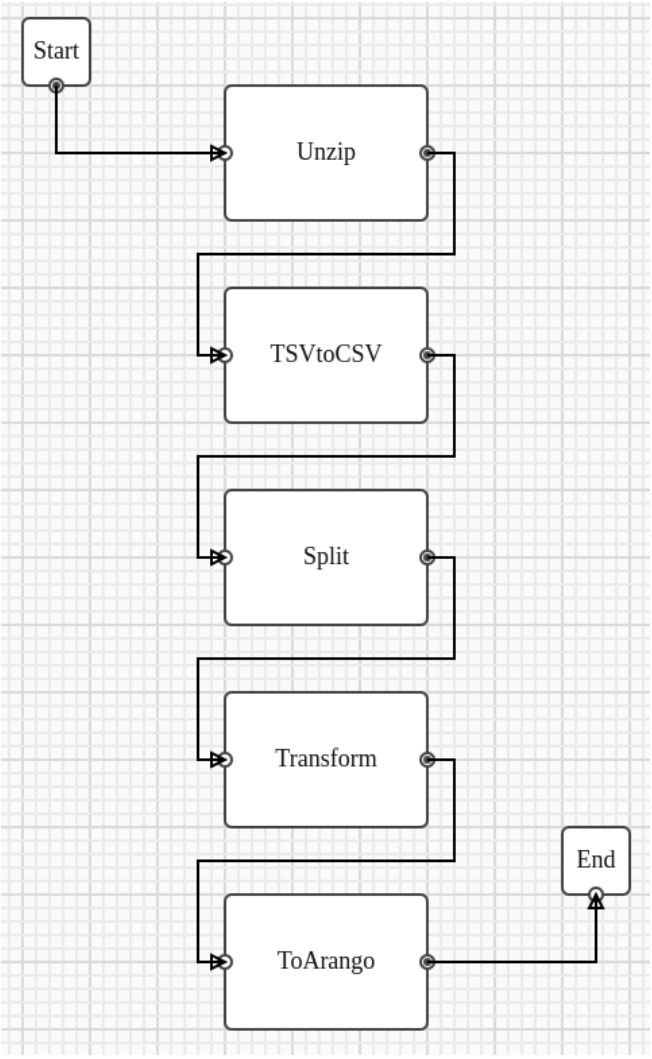
Total time spent: 4 hours

- 10 minutes: system design.
- 2:30 hours: learning.
- 1:20 hour: evenly splitted between code writing and debugging.

Task 3

Describe the pipeline using DEF-PIPE tool of the DataCloud Project

Architecture



The pipeline has been implemented using the DEF-PIPE tool. Each step implements the logic provided by the component's referenced Docker image.

Pipeline description

Steps are the same as the first pipeline, if not that here we don't have any orchestrator component that needs to invoke the services but we leverage DEF-PIPE itself as an orchestrator.

Time assessment

Total time spent: 1 hour

- 50 minutes: learning.
- 10 minutes: implementing the pipeline.