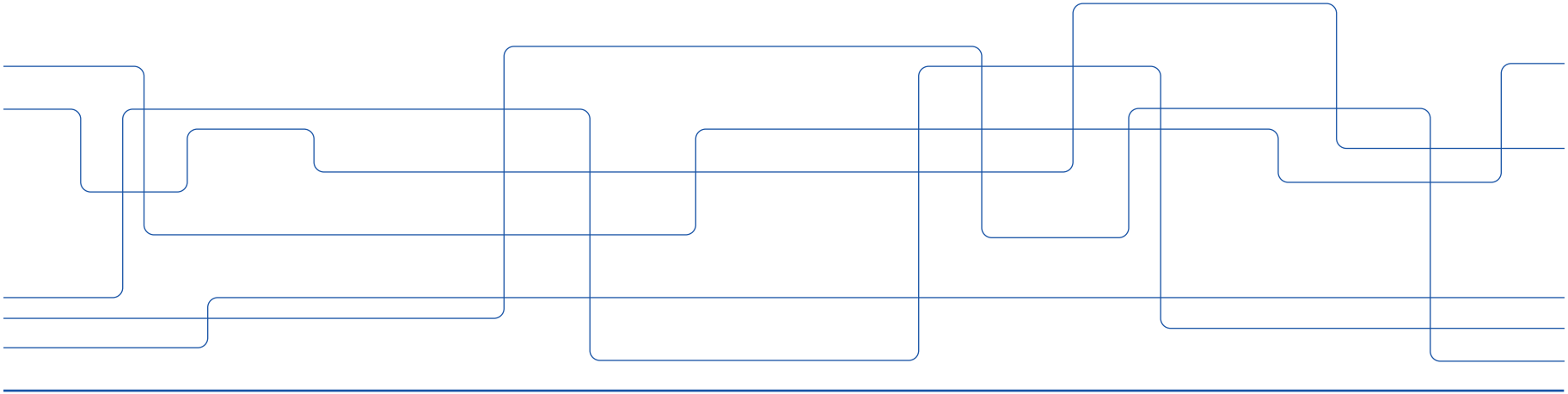
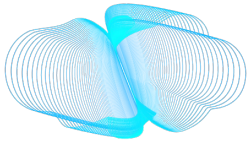
DataCloud project

Tutorial for students of ID2207

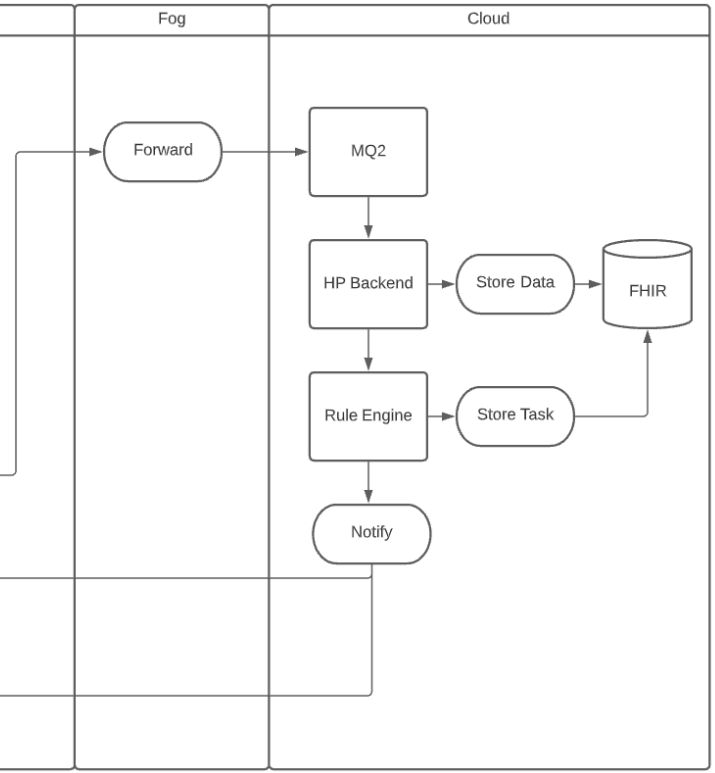


Content

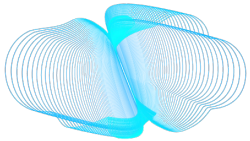


- Introduction
- DataCloud project and related work
- Requirement analysis
- DataCloud DSL language and visual tool
- Architecture
- Conclusion and futuer work

Data Processing in a eHealth application



Introduction

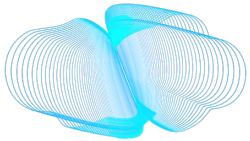


- The entire pipeline lifecycle should be supported by advance tools.

- Most of recent tools have focus mainly on runtime execution of pipelines rather than on pipeline definition.

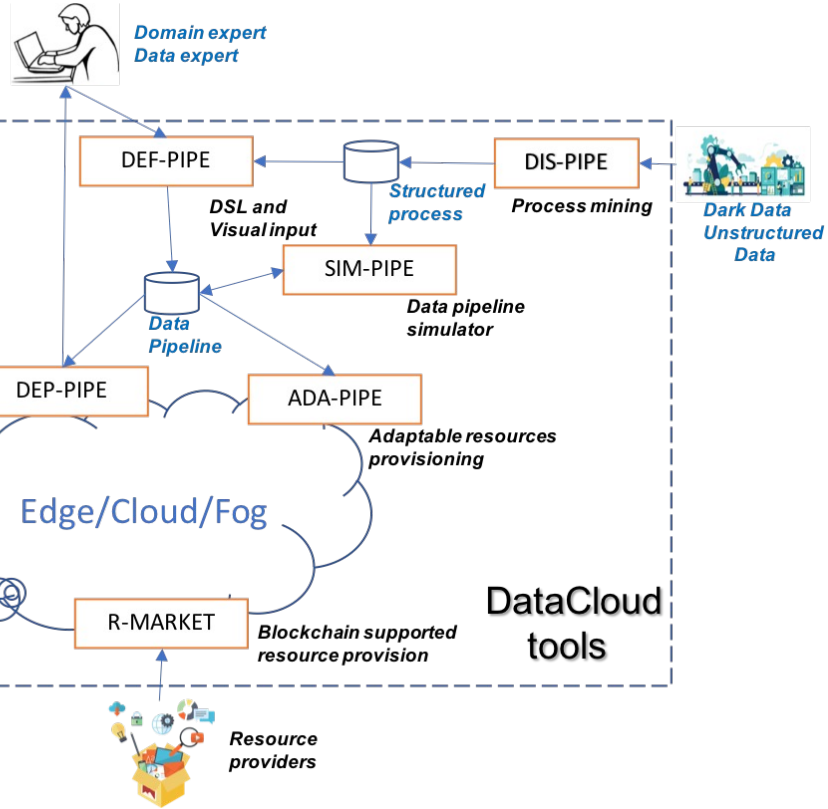
- The tools should allow their usage by domain experts

Challenges

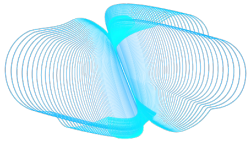


- You can hardcode the pipeline
- You can use some automated tools (Argo, Airflow…)
- You can use fully automated tool-box supporting easy definition of pipelines

The DataCloud project



Related work

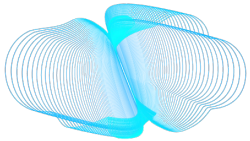


- For analysis of related solutions we refer to our comprehensive survey of related systems and solution in the framework of the DataCloud project:

> *M. Matskin, S. Tahmasebi, A. Layegh, A. H. Payberah, A. Thomas, N. Nikolov, and D. Roman, "A survey of big data pipeline orchestration tools from the perspective of the datacloud project," in Proceedings of the DAMDID Conference, 2021, pp. 63–78.*

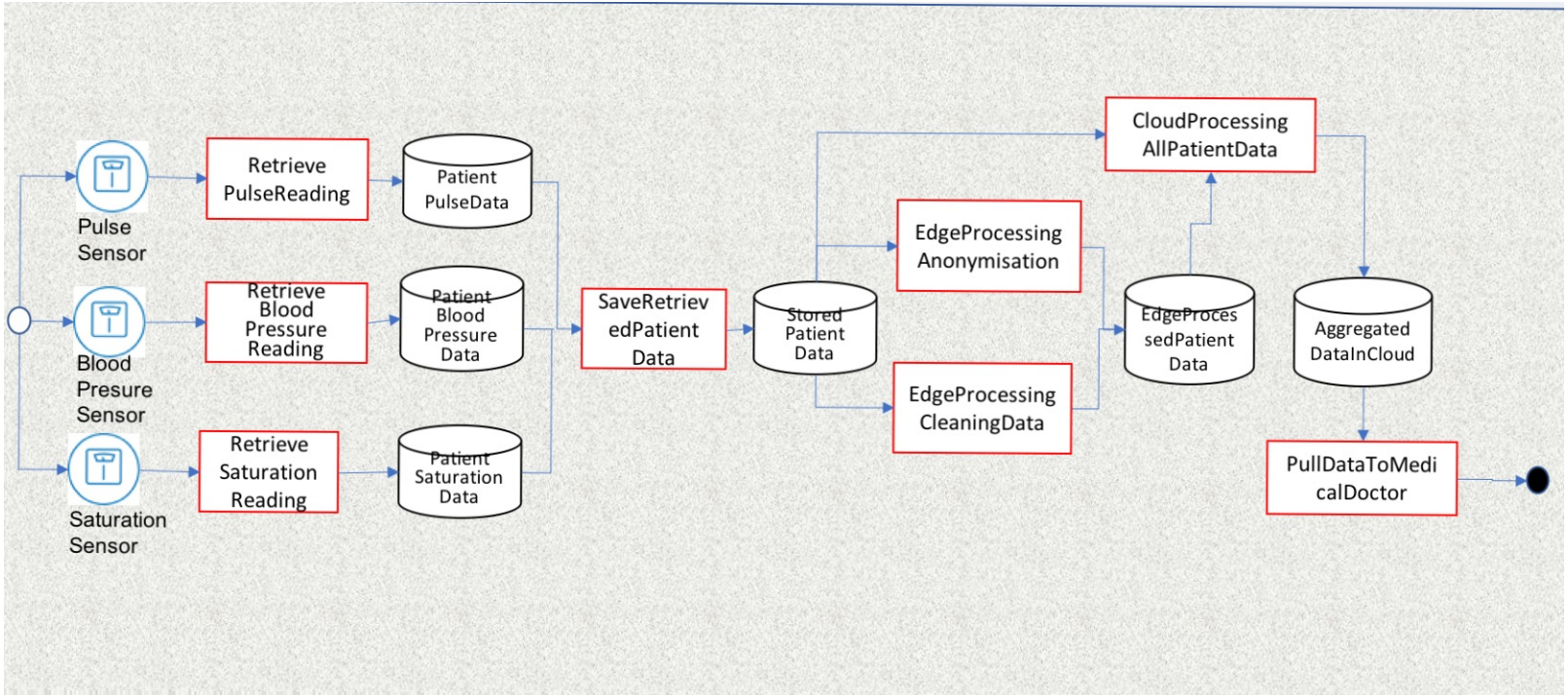
Available at http://ceur-ws.org/Vol-3036/paper05.pdf

General requirements for Big Data pipeline description components:



- Developing a DSL for a textual description
- Developing a visual/graphical form of DSL
- DSL and tools must support the separation of concerns between design and run-time issues
- Support of reuse of previously developed steps and pipelines in designing new pipelines
- Supporting a smooth data transfer between steps
- Applying containerization in pipeline descriptions
- Integrating the discovering and simulation components in the Big Data pipeline orchestration systems

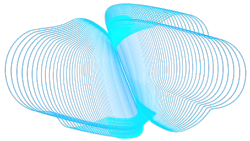
Example pipeline



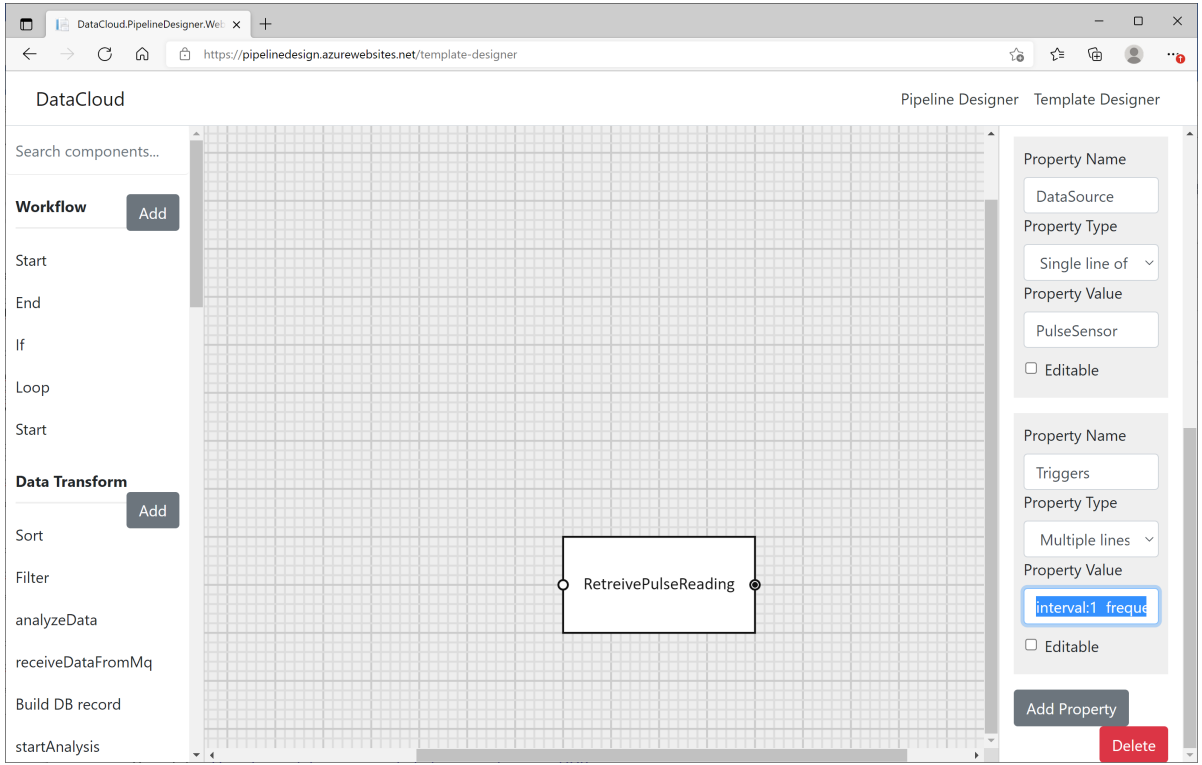
Model serialization (grammar)

```
Pipeline DigitalHealthUsecase {
   communicationMedium: MessageQueue,
   environmentParameters: {
      "MQ_HOST": kubeMQ,
      "MQ_USER": user1
   },
   steps:
      data-source step: RetreivePulseReading
      implementation: docker-implementation
      image: 'tellucare-api:latest'
      dataSource: PulseSensor,
      triggers: interval-schedule interval: 1
      frequency: MINUTE
      startTime: '30.10.2021 12:00:00'

      data-source step: RetrieveBloodPressure
      implementation: docker-implementation
      image: 'registry.sintef.cloud/tellucare-edge'
      dataSource: BloodPressureSensor,
      triggers: interval-schedule interval: 1
      frequency: MINUTE
      startTime: '30.10.2021 12:00:00'
...

      data-sink step: SaveRetrievedPatientData
      implementation: docker-implementation
      image: 'tellucare-application:latest'
      environmentParameters: {
            RABBITMQ_HOST=oslo.sintef.no:5672
            RABBITMQ_USERNAME=tellucareapi
            RABBITMQ_PASSWORD=???

            }
```

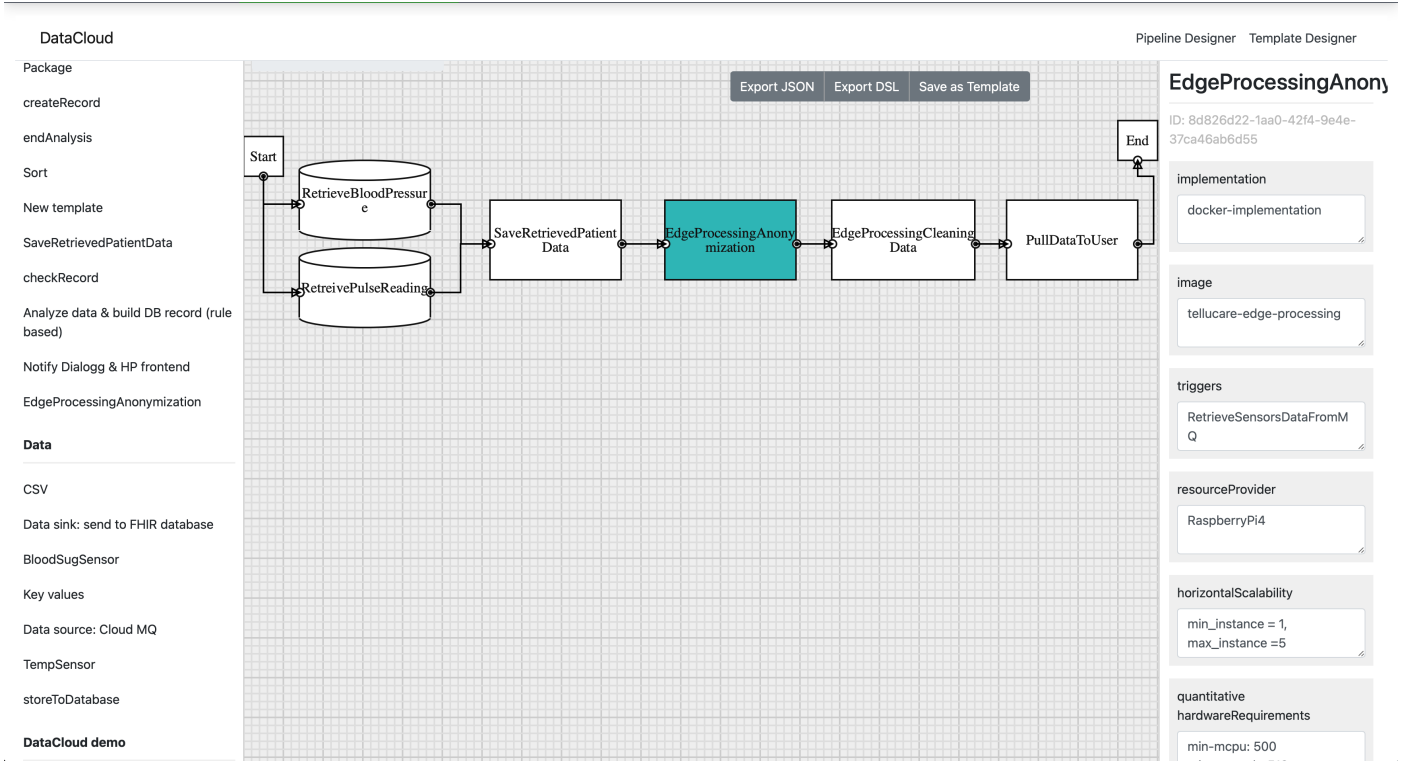
```
dataSource: StoredPatientData,
preCondition:
      [{condition: CheckInputINotEmpty,
       inputStep: retrieveDataOne},
       {condition: CheckInputIINotEmpty,
       inputStep: retrieveDataTwo},
       {condition: CheckInputIIINotEmpty,
       inputStep: retrieveDataThree},],
      operator: OR
...
data-processing step:EdgeProcessingAnonymization
implementation: docker-implementation
image: 'tellucare-edge-processing'
executionRequirements:
      horizontalScalability {
      min-instance : 1
      max-instance: 5
      },
      quantitative hardwareRequirements {
            min-mcpu: 500
            min-ram-mb: 512
            min-storage-mb: 1024
      }
triggers: RetrieveSensorsDataFromMQ
resourceProvider: RaspberryPi4

data-processing step: EdgeProcessingCleaningData
implementation: docker-implementation
image: 'tellucare-edge-cleaning'
triggers: RetrieveSensorsDataFromMQ
resourceProvider: RaspberryPi4

data-processing step: PullDataToUser
implementation: docker-implementation
image: 'tellucare-user'
resourceProvider: AWS
```



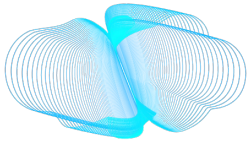
Visual tool (step description)



Visual tool (pipeline description)

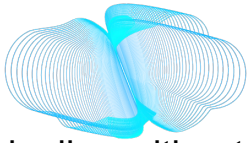


The Goal of Evaluation



- We would like measure efficiency of usage our visual solution to description of Big Data Pipelines wrt. manual coding and usage of existing tools.
- We would like measure efficiency of reusability of solutions developed with our toot wrt. manual coding and usage of existing tools.

Your tasks



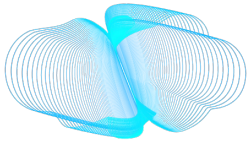
1. Your must manually describe components of the selected pipeline and the pipeline without using any orchestration tool.
   - You assume that each step in the pipeline is implemented as a container
   - You can use any data transfer mechanism you prefer

2. Your must describe the selected pipeline with using Argo-workflow tool (https://argoproj.github.io/argo-workflows/).
   - You assume that each step in the pipeline is implemented as a container
   - You can use any data transfer mechanism you prefer

3. You must describe the selected pipeline with using DEF-PIPE tool of the DataCloud projects (https://crowdserv.sys.kth).
   - You assume that each step in the pipeline is implemented as a container
   - You should use a specified data transfer mechanism

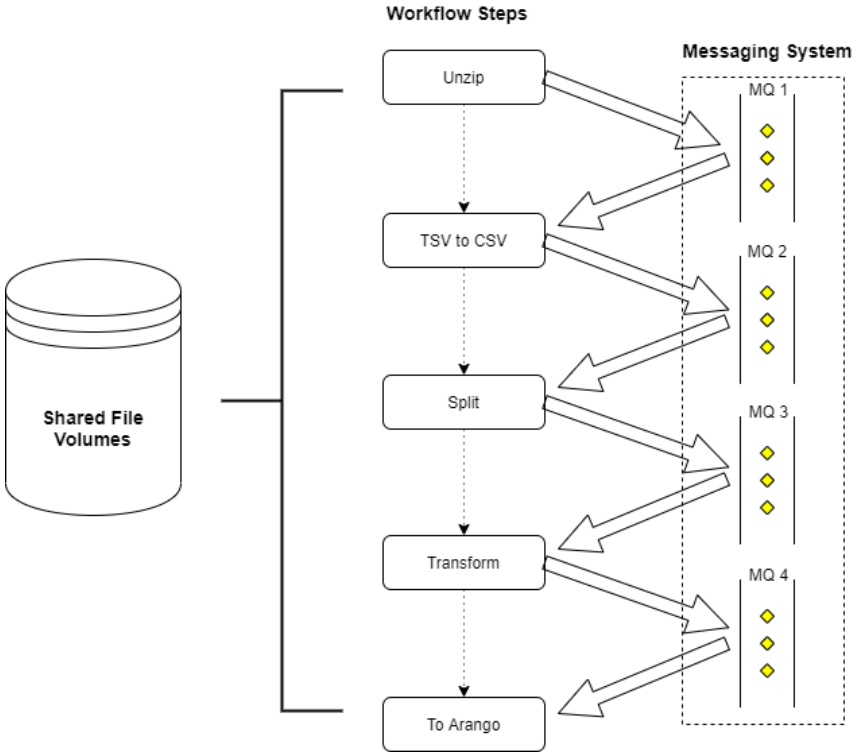
4. After completion tasks 1-3 you will be given a task to describe another pipeline that is quite similar to (but not exactly the same as) initial pipeline and you must repeat the Tasks 1-3 with this pipeline. In case of another pipeline you will be able to use previously described components (not only yours).

Your tasks

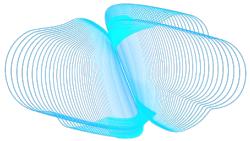


- We expect you only to describe (but not to execute) the pipeline.
- You must do the tasks exactly in the given order 1,2,3,4 but not to mix them by doing them in another order
- For each of the above-mentioned tasks you must make measurements (in hours) how much time you spend to complete the task. The measured time must include everything, including time to learn a new method or tool. Please be precise as much as possible. Acceptance of your results will not depend on how long it take in absolute time, but we are interested in relative time between completing all tasks.
- For performing all tasks 1,2,3 you will get 12 bonus points for performing all 1,2,3,4 tasks you will get 17 bonus points.
- For performing tasks 2,3,4 you get 10 bonus points
- For performing only tasks 3,4 you get 4 bonus points
- You will get bonus points only if you perform these tasks and report to us before October 25.
- Your report must contain description of the pipelines for all requested cases/tasks and measured time you spend on completion the tasks. We will ask you present your deliverables to briefly explain what you did in order to be sure that it is your own result.

Example pipeline

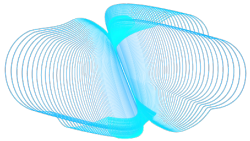


Example components of a pipeline



Step Name	Description	Input/output
Unzip	Extracts zip file	A zip file/ TSV files
TSV to CSV	Converts a TSV file to comma-separated values (CSV) file	A TSV file/ a CSV file
Split	Splits a CSV file into smaller pieces if the number of rows in the files is above a certain number	A CSV file/ CSV files
Transform	Cleans and preprocesses a CSV file using a stand-alone executable from Grafterizer	A CSV file/ a CSV file
To Arango	Converts a Datagraft CSV files to ArangoDB values based on external transformation JSON from Grafterizer	A CSV file / ArangoDB collection

Conclusion



- DataCloudDSL supports requirements obtained from SoTA and Business cases.
- A language has textual and visual forms.
- Visual form of the pipeline descriptions does not require deep technical knowledge from domain experts
- Support for re-use of previous solutions
- Future work is planned for applying the DataCloudDSL to partners use cases and for libraries of design solutions for pipelines design.

THANK YOU!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016835, the DataCloud Project.

https://datacloudproject.eu/