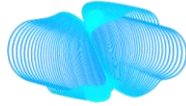


ICT-40-2020
H2020-ICT-2018-20



DataCloud

ENABLING THE BIG DATA PIPELINE LIFECYCLE ON THE COMPUTING CONTINUUM

D6.3: DATACLOUD INTEGRATED TOOLBOX V1

Document Metadata

Work package	WP6
Date	30.11.2022
Deliverable editor	Giannis Ledakis (UBI)
Version	1.0
Contributors	Narges Mehran (AAU), Dragi Kimovski (AAU), Souvik Sengupta (IEX), Brian Elvesæter (SINTEF), Nikolay Nikolov (SINTEF), Ioannis Plakas (UBI), Antoine Pultier (SINTEF), Amirhossein Layegh Kheirabadi (KTH), Mihhail Matskin (KTH), Simone Agostinelli (URO), Dario Benvenuti (URO), Andrea Marrella (URO)
Reviewers	Brian Elvesæter (SINTEF), Alexandre Ulisses (MOG)
Keywords	Big Data Pipeline Definition, Big Data Pipeline Simulation, Big Data Pipeline Discovery, Big Data Pipeline Execution, Toolbox, Integration
Dissemination Level	Public

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	14.09.2022	Table of Contents.	Giannis Ledakis (UBI)
V0.2	02.10.2022	Updates and assignments for section 3	Giannis Ledakis (UBI)
V0.3	26.10.2022	Section 2 updates, Installation material	Giannis Ledakis (UBI) with input by all partners
V0.4	04.11.2022	Input on section 3	Giannis Ledakis (UBI) with input by all partners
V0.5	12.11.2022	Updates on section 4	Giannis Ledakis (UBI) with input by all partners
V0.6	24.11.2022	Section 5, and updates in all sections	Giannis Ledakis (UBI)
V0.7	29.11.2022	Reviewer comments	Alexandre Ulisses (MOG), Brian Elvesæter (SINTEF)
V0.8	30.11.2022	Final fixes and updates for reviewer comments	Giannis Ledakis(UBI)
V1.0	30.11.2022	Final formatting and layout.	Dumitru Roman (SINTEF), Brian Elvesæter (SINTEF)

DataCloud Project Partners

SI	SINTEF AS
URO	SAPIENZA UNIVERSITY OF ROME
AAU	UNIVERSITY OF KLAGENFURT
KTH	ROYAL INSTITUTE OF TECHNOLOGY
IEX	IEXEC BLOCKCHAIN TECH



UBI	UBITECH
JOT	JOT INTERNET MEDIA
MOG	MOG TECHNOLOGIES SA
CER	CERAMICA CATALANO SRL
TLE	TELLU IOT AS
BOS	ROBERT BOSCH GMBH

DISCLAIMER

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

This document reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.



EXECUTIVE SUMMARY

The main objective of the DataCloud project is to develop an ecosystem consisting of tools, methods and software components that can efficiently exploit heterogeneous computing resources for managing Big Data pipelines on the Computing Continuum.

DataCloud suggests a Big Data pipeline lifecycle consisting of discovery, definition, simulation, adaptation, provisioning, and deployment stages, and provides not only the six interoperable tools to support such a lifecycle, but also an integrated DataCloud Toolbox.

This deliverable describes the first, early release of DataCloud Toolbox. This first release is composed by components loosely integrated and being accessible under a common landing page. The status of each tool is presented, covering development and integration aspects. Furthermore, we examine how well the envisioned flows of DataCloud are supported in this first release, while we also plan for the final release that will be more integrated, will focus on the user interactions and therefore easier to be utilized and exploited.

The document also contains instructions for the setup of the tools and also a walkthrough for the execution of the supported functionalities in this first release of the DataCloud integrated toolbox.



TABLE OF CONTENTS

1	INTRODUCTION	10
1.1	Objectives and scope	10
1.2	Relationship to other deliverables	10
1.3	Document structure	10
1.4	Applicable documents.....	10
1.5	Reference documents.....	11
1.6	Glossary	11
2	EARLY RELEASE OF DATA CLOUD INTEGRATED TOOLBOX	12
2.1	Flows of the platform in the First Release	13
2.2	Toolbox Overview	17
2.2.1	Identity Management Keycloak	17
2.2.2	Common UI.....	19
2.2.3	Deployment.....	22
2.2.4	Enabling technologies	22
3	OVERVIEW OF INTEGRATION AND TOOLBOX COMPONENTS.....	24
3.1	Pipeline Discovery	24
3.1.1	Features Implementation.....	25
3.1.2	Integration with the Toolbox Components	26
3.1.3	Deployment, Code and Documentation Availability	27
3.2	Pipeline Definition	28
3.2.1	Features Implementation.....	30
3.2.2	Integration with the Toolbox Components	30
3.2.3	Deployment, Code and Documentation Availability	31
3.3	Pipeline Simulation	32
3.3.1	Features Implementation.....	32
3.3.2	Integration to the Toolbox Components.....	34
3.3.3	Deployment, Code and Documentation Availability	34
3.4	Pipeline Scheduling and Adaptation	36
3.4.1	Features Implementation.....	36
3.4.2	Integration with the Toolbox Components	36
3.4.3	Deployment, Code and Documentation Availability	38
3.5	Decentralized Resource Marketplace (R-MARKET).....	39



3.5.1	Features Implementation.....	39
3.5.2	Integration with the Toolbox Components	40
3.5.3	Deployment, Code and Documentation Availability	41
3.6	PipeLine Deployment & Management.....	41
3.6.1	Features Implementation.....	42
3.6.2	Integration with the Toolbox Components	42
3.6.3	Deployment, Code and Documentation Availability	45
4	DATA CLOUD INTEGRATED TOOLBOX DOCUMENTATION	46
4.1	Setup Instructions.....	46
4.2	DataCloud Toolbox Usage.....	47
4.2.1	Landing in the Toolbox.....	47
4.2.2	Pipeline Discovery.....	48
4.2.3	Pipeline Design	54
4.2.4	Pipeline Simulation.....	57
4.2.5	Pipeline Execution & Runtime Management	60
5	PLANS FOR FUTURE RELEASES	66
	APPENDIX A – INSTALLATION INSTRUCTIONS	70
	DIS-PIPE.....	70
	DEF-PIPE.....	71
	SIM-PIPE	72
	SIM-PIPE Sandbox environment set up	72
	SIM-PIPE tool Interface and Simulation Controller	73
	ADA-PIPE	74
	R-MARKET	75
	DEP-PIPE	76



LIST OF FIGURES

Figure 1: DataCloud Conceptual architecture (source D1.2).....	12
Figure 2: DataCloud integrated tool interaction workflow (source D1.2)	13
Figure 3: DataCloud tool integration interfaces	16
Figure 4: Keycloak clients for the integration of DataCloud tools.....	17
Figure 5: Authentication/authorization in DataCloud	18
Figure 6: UserID usage across the tools of DataCloud	19
Figure 7: Common UI mock-up (initial proposition).....	20
Figure 8: Common UI mock-up (2 nd proposition)	20
Figure 9: DataCloud common UI	21
Figure 10: Tool accessed through the DataCloud common UI	21
Figure 11: Position in the GUI of the function importLog(xesLogID)	24
Figure 12: exportPipeline(pipelineID) REST-API	26
Figure 13: DIS-PIPE GitHub page	27
Figure 14: Deployment of DIS-PIPE Architecture and REST-APIs	28
Figure 15: Step Designer mode	29
Figure 16: Workflow Designer mode.....	29
Figure 17: DEF-PIPE API documentation web page.....	31
Figure 18: Screenshot of SIM-PIPE's user interface	32
Figure 19: Screenshots of software management organisation on GitHub	33
Figure 20: Screenshot of the SIM-PIPE Git repository hosted on GitHub	34
Figure 21: SIM-PIPE API documentation	35
Figure 22: ADA-PIPE inputs and output provided by/to other DataCloud tools	36
Figure 23: ADA-PIPE Swagger API.....	37
Figure 24: ADA-PIPE POST HTTP request & response	37
Figure 25: ADA-PIPE matching-based tool tutorial.....	38
Figure 26: ADA-PIPE Front page setup tutorial.....	39
Figure 27: Swagger Implementation of R-MARKET Node.JS API server.....	40
Figure 28: API for deployment of a pipeline chunk.....	43
Figure 29: API for deletion of a pipeline chunk	44
Figure 30: Scaling API.....	45
Figure 31: Repositories of DEP-PIPE.....	45
Figure 32: The toolbox page.....	46
Figure 33: Tools description in the landing page	47
Figure 34: Keycloak based login page.....	48



Figure 35: DIS-PIPE main page and its sections	49
Figure 36: The performance values related to the pipeline step “Data Merge”	50
Figure 37: Pipeline Map Visualization Options	50
Figure 38: Features of the Performance tab	51
Figure 39: An overview of the Event Log Analysis View	51
Figure 40: The Attribute filter	52
Figure 41: Mapping log events with pipeline steps	53
Figure 42: Alignment settings	53
Figure 43: Results of the trace alignment activity for the JOT pipeline	54
Figure 44: Login in DEF-PIPE	55
Figure 45: Left panel of DEF-PIPE	55
Figure 46: Step Designer in DEF-PIPE	56
Figure 47: Workflow designer in DEF-PIPE	57
Figure 48: Landing page of SIM-PIPE	58
Figure 49: Creating Simulation using a pipeline descriptor	58
Figure 50: Simulation overview	59
Figure 51: Configuring a simulation run	59
Figure 52: Simulation results	60
Figure 53: Landing page of the DEP-PIPE and the Runtime Management Dashboard ..	60
Figure 54: R-MARKET resource available at the DataCloud Workerpool	61
Figure 55: Onboarding a Kubernetes Cluster for pipeline deployment	61
Figure 56: User pipelines presented in DEP-PIPE	62
Figure 57: ADA-PIPE scheduling inputs	62
Figure 58: Pipeline Chunks provided by ADA-PIPE for deployment	62
Figure 59: Viewing a pipeline chunk before deployment	63
Figure 60: Enabling security	63
Figure 61: Pipeline Configuration	64
Figure 62: Pipeline Deployment	64
Figure 63: Pipeline Deployment Menu	64
Figure 64: Policy Editor for scaling	65
Figure 65: Pipeline Graph and Monitoring Data	65
Figure 66: Sequence Diagram presenting the updated DataCloud flows	67
Figure 67: Wireframe for retrieving pipelines from DEF-PIPE	68
Figure 68: Wireframe for the management of a pipeline and its chunks	68



LIST OF TABLES

Table 1: Applicable documents	10
Table 2: Reference documents	11
Table 3: Glossary table	11
Table 4: Overview of DataCloud tool deployments	22
Table 5: DIS-PIPE features implementation and integrations status	25
Table 6: DIS-PIPE Features Planned for second release of the toolbox	25
Table 7: DEF-PIPE features implementation and integrations status	30
Table 8: DEF-PIPE Features Planned for second release of the toolbox	30
Table 9: SIM-PIPE features implementation and integrations status	33
Table 10: SIM-PIPE Features Planned for second release of the toolbox	33
Table 11: ADA-PIPE features implementation and integrations status	36
Table 12: ADA-PIPE features planned for second release of the toolbox	36
Table 13: R-MARKET features implementation and integrations status	39
Table 14: R-MARKET features planned for second release of the toolbox	40
Table 15: DEP-PIPE features implementation and integrations status	42
Table 16: DEP-PIPE features planned for second release of the toolbox	42
Table 17: Aggregated DataCloud features planned for second release of the toolbox .	69



1 INTRODUCTION

1.1 OBJECTIVES AND SCOPE

This document describes an early release of the DataCloud integrated toolbox accompanied by a development, integration and use documentation.

1.2 RELATIONSHIP TO OTHER DELIVERABLES

The integration and the development of software for the Toolbox and this document are following the requirements analysis of deliverable [D1.1], the architecture as described in [D1.2] and the collaborative development methodology described in [D6.2].

But mainly, this document is based on the deliverables presenting the actual outcomes of the specific tools and provided in the scope of WP2, WP3, WP4 and WP5.

1.3 DOCUMENT STRUCTURE

Section 2 provides an overview of the Integrated DataCloud toolbox, explaining the architecture and presenting the current status. A more detailed, technical presentation of each component is provided in section 3.

Section 4 provides the documentation of the toolbox, covering the toolbox setup, and also a walkthrough of the Toolbox from the user perspective, as currently supported. Section 5, provides the plans for the future, final integrated toolbox release.

1.4 APPLICABLE DOCUMENTS

Table 1: Applicable documents

Reference	Title
[Gag]	DataCloud – Grant Agreement number 101016835
[DoA]	[GA] Annex-1 Description of the Action
[CA]	DataCloud Consortium Agreement



1.5 REFERENCE DOCUMENTS

Table 2: Reference documents

Reference	Title
[D1.1]	Technical and market requirements
[D1.2]	DataCloud architecture v1
[D2.1]	Techniques for Big Data extraction, processing and pipeline discovery
[D2.3]	Framework for Big Data pipeline discovery V1
[D3.1]	Design methodology for Big Data pipelines
[D3.2]	Big Data pipeline definition and simulation tool v1
[D3.3]	Big Data pipeline definition and simulation tool v2
[D4.2]	Blockchain-based decentralized marketplace v1
[D5.1]	Data-aware resource provisioning across the Computing Continuum
[D5.2]	Pipeline deployment and system orchestration
[D5.3]	Event-detection and infrastructure and deployment adaptation
[D6.2]	DataCloud toolbox testbed

1.6 GLOSSARY

Table 3: Glossary table

Acronym	Description
API	Application Programming Interface
CORS	Cross-origin resource sharing
DSL	Domain-specific language (specifically with respect to the DSL for Big Data Pipelines)
IAAS	Infrastructure as a Service
IP	Internet Protocol
JSON	JavaScript Object Notation
ML	Machine Learning
OIDC	OpenID Connect
REST	Representational state transfer
SoTA	State-of-the-Art
SDK	Software Development Kit
TCP	Transmission Control Protocol
UI	User Interface



2 EARLY RELEASE OF DATA CLOUD INTEGRATED TOOLBOX

DataCloud Integrated Toolbox is considered as the set of loosely integrated tools that allow to support all the phases related to the pipeline lifecycle, as explained in [D1.2].

- **Discovery** of pipelines from the deluge of collected and unused data;
- **Definition** of pipelines using a domain-specific language (DSL);
- **Simulation** and optimisation of pipelines deployment before actual operation;
- **Adaptation** of pipelines executions to performance fluctuations and faults;
- **Provisioning** of on-demand Computing Continuum resources for deployment and execution of pipelines;
- **Deployment** and execution monitoring of pipelines.

DataCloud automates and optimises the lifecycle of Big Data pipelines across these stages through the integration of six tools, illustrated in and grouped in three bundles: Discovery bundle, Design bundle and Runtime bundle.

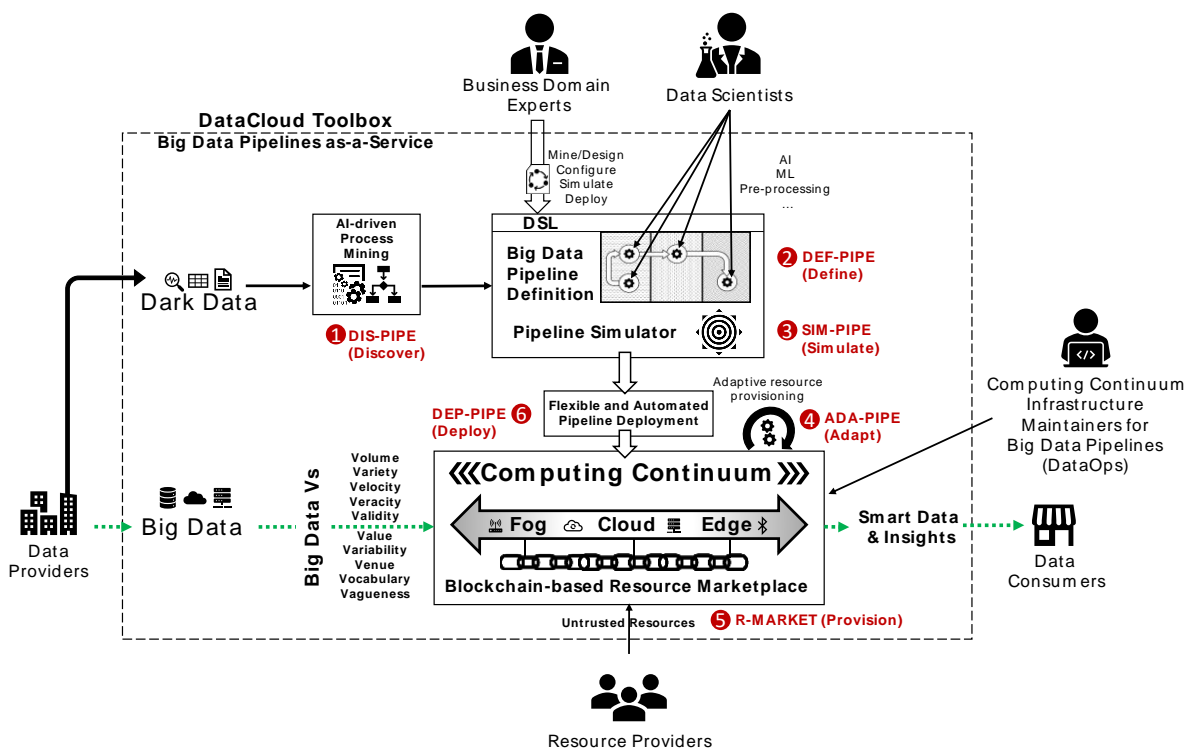


Figure 1: DataCloud Conceptual architecture (source D1.2)

The tools proposed by DataCloud are mapped to each of these phases and are developed in the scope of WP2-WP5 and their status is presented in section 3.

- **DIS-PIPE (Discovery pipeline tool)**
- **DEF-PIPE (Definition pipeline tool)**
- **SIM-PIPE (Simulation pipeline tool)**
- **R-MARKET (Provision through resource marketplace)**
- **ADA-PIPE (Scheduling and adaption pipeline tool)**
- **DEP-PIPE (Deployment pipeline tool)**

Based on the presented tools and phases of the suggested lifecycle, it is clear that in DataCloud, we focus not only on the runtime execution of the pipelines, but we also aim to leverage Dark Data (data that is collected but not used and turned into value) and make simulation part of the design. Therefore, the objective of DataCloud toolbox is to provide an integrated, general-purpose solution for Big Data pipelines usable by a broad audience.

For this first release of DataCloud toolbox the goal was to facilitate the testing of the components by the use cases, even if manual steps are required at some points. This approach allows us to collect feedback (testing and feedback collection is in progress, and being documented in DataCloud deliverable *D6.4 Testing and evaluation v1* that is due at M24. Main functionalities include the discovery, definition, design and deployment of a pipeline; these functionalities are presented in a user-oriented approach in section 4.2 of this document.

2.1 FLOWS OF THE PLATFORM IN THE FIRST RELEASE

Figure 2 presents a complete Big Data pipeline processing lifecycle in DataCloud as an integrated workflow with 23 interactions of the six tools across the three design stages. This workflow was presented in [D1.2].

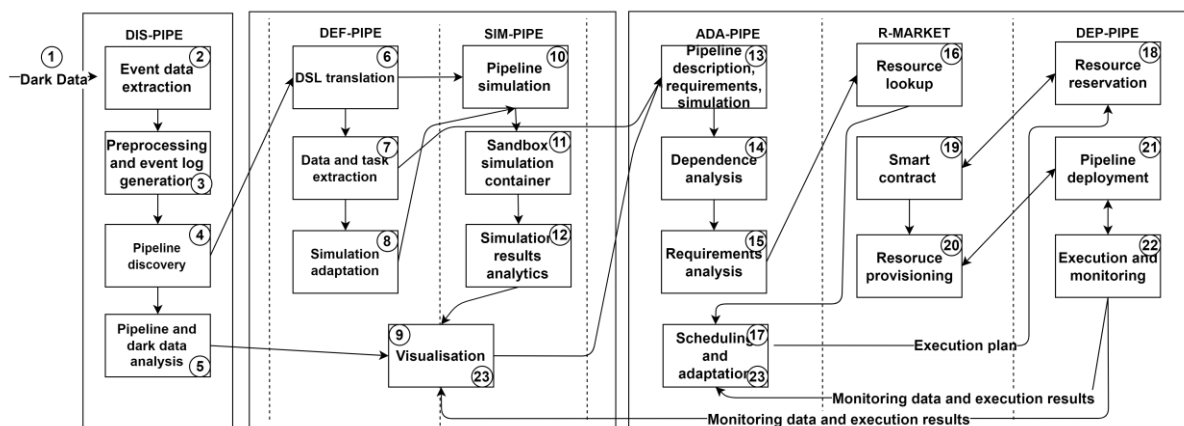


Figure 2: DataCloud integrated tool interaction workflow (source D1.2)

Below, we present each of the steps and their actual support in this first release

1. DIS-PIPE is the typical entry tool for collecting Dark Data from heterogeneous sources.

Step achieved, see sections 3.1 and 4.2.2.

2. DIS-PIPE analyses torrents of Dark Data and converts it into valuable data that reveal the concrete events that happened during daily operations. An event data contains knowledge about the execution of relevant pipelines.

Step achieved, see sections 3.1 and 4.2.2.

3. DIS-PIPE applies dedicated segmentation, abstraction and filtering techniques to the event data set and extracts the execution traces associated with a specific pipeline. DIS-PIPE automatically identifies and discards unnecessary events that do not contribute to the pipeline execution, representing noise or Dark Data. Finally, it stores a clean set of pipeline execution traces into an event log repository.

Step achieved, see sections 3.1 and 4.2.2.

4. DIS-PIPE employs well-established process mining algorithms to discover and learn the structure of a Big Data pipeline by interpreting the event log repository.

Step achieved, see sections 3.1 and 4.2.2.

5. DIS-PIPE supports a variety of analytics as plugins, integrated through a user-friendly visualisation workbench, comprising flowcharts, statistics, and dashboards.

Step achieved, see sections 3.1 and 4.2.2.

6. DEF-PIPE receives Big Data pipelines from DIS-PIPE, translates them into a proper DSL and provides a visual environment for their definition and customisation. Optionally, DEF-PIPE allows domain scientists to design new pipelines without the DIS-PIPE discovery functionality.

Integration step in progress, DIS-PIPE allows the export of the pipeline while DEF-PIPE has implemented the API for DSL uploading, see section 3.2.

7. DEF-PIPE extracts data and steps from the Big Data pipeline into an integrated library of reusable pipeline templates and execution steps based on container technology.

Step achieved, with defined pipeline templated and steps supported using containers . See sections 3.2 and 4.2.3.

8. DEF-PIPE sends the fully defined pipeline descriptions to SIM-PIPE for simulation and evaluation before committing to actual deployment and execution. A simulation adapter provides proper integration of the pipelines with the SIM-PIPE tool through (de-)serialisation (saving and loading) of pipeline definitions in a standardised format (i.e., JSON, XML, YAML).

Integration step in progress, SIM-PIPE allows the import of the pipeline while DEF-PIPE has implemented the API for DSL exporting, see sections 3.2 and 3.3.

9. DEF-PIPE and SIM-PIPE share a graphical tool supported by an open application programming interface (API), which allows the pipeline definition and modification and the visualisation of the simulation or execution results.



Two different UI views have been provided, while DEF-PIPE has implemented the API for DSL exporting, see sections 3.2 and 3.3.

10. SIM-PIPE extracts the pipeline steps from their definitions and simulates them by instantiating container templates deployed in a sandbox.

Step achieved. See sections 3.3 and 4.2.4.

11. SIM-PIPE automatically provides input to the sandboxed simulation steps, estimates their throughput and execution properties, and performance under configurable load conditions with high accuracy for the entire pipeline.

Step achieved. See sections 3.3 and 4.2.4.

12. SIM-PIPE analyses and shares the simulation results with DEF-PIPE for visualisation.

Simulation results are visualized in SIM-PIPE UI. See sections 3.3 and 4.2.4.

13. After confirming the pipeline optimisation by the user, DEF-PIPE and SIM-PIPE provide the pipeline structure, steps, and executable data to ADA-PIPE.

DEF-PIPE has implemented the API for DSL exporting that ADA-PIPE is using, see sections 3.2 and 3.4. Also DEF-PIPE is using the API for the DSL export of a pipeline.

14. After receiving the pipeline structure, ADA-PIPE performs dependence analysis between the steps to identify their correct sequence and identify those that can execute in parallel.

For the first release this transformation and analysis of the steps has been a manual procedure. For the final release this process is being automated.

15. Afterwards, ADA-PIPE analyses the requirements of each specific step, such as processing speed, memory, and storage sizes, and sends them to R-MARKET.

For the first release this transformation and analysis of the steps has been a manual procedure. For the final release this process is being automated.

16. R-MARKET searches for appropriate resources with associated performance metadata based on the identified step requirements. For this purpose, it deploys a decentralised marketplace based on a permissionless blockchain to federate a vast set of heterogeneous virtualised resources (i.e., virtual machines, containers) from various providers. R-MARKET returns the list of available resources to ADA-PIPE.

This step has been agreed to be made before deployment of a pipeline because the user must have an existing Kubernetes Cluster that could also contain resources from R-MARKET. In addition for the final release, we will support the adaptation of the Cluster dynamically through the execution of this step.

17. ADA-PIPE analyses the resources published in R-MARKET based on the pipeline steps' requirements, then it creates an execution schedule and sends it to DEP-PIPE.

Resources both in R-MARKET and from other cloud providers will be taken into account for adapting the underlying Kubernetes cluster. However the base scenario that is supported now the pipeline schedule is proved to be executed in a Kubernetes Cluster that can be hosted in various regions and devices.



18. ADA-PIPE signs the smart contracts with providers describing agreed SLO terms, like processing speed, memory, storage sizes, power consumption, etc.

This step is not supported in the first release.

19. DEP-PIPE receives the list of resources from the signed smart contract and checks their compliance with ADA-PIPE’s sheduling plan.

This step is not supported in the first release.

20. R-MARKET provisions the resources to DEP-PIPE according to the scheduling plan.

This step is not supported in the first release. Resources of R-MARKET are used only as pre-provisioned resources that have been added to the Kubernetes Cluster.

21. DEP-PIPE deploys the pipeline steps on the resources provisioned by R-MARKET following the schedule provided by ADA-PIPE.

Step achieved. Currently the schedule is provided by ADA-PIPE through the REST call. See sections 3.6 and 4.2.5. In the final release this will be executed seamlessly by the user through the Runtime Dashboard.

22. DEP-PIPE executes the pipeline as orchestrated by ADA-PIPE and continuously monitors and reports its agreed SLOs.

Step partially achieved. Policies for SLOs are created, but are not used yet by ADA-PIPE for providing updated schedule. See sections 3.6 and 4.2.5.

23. Finally, DEP-PIPE provides the pipeline monitoring data to DEF-PIPE for visualisation and ADA-PIPE for adaptation responding to fluctuations and faults.

Step partially achieved. Monitoring data collected and visualized, ADA-PIPE usage of the monitoring data is under integration. See sections 3.6 and 4.2.5.

These interactions were also the starting point for the integration of the first version of the toolbox, as it allowed us to identify the interaction interfaces among the DataCloud tools.

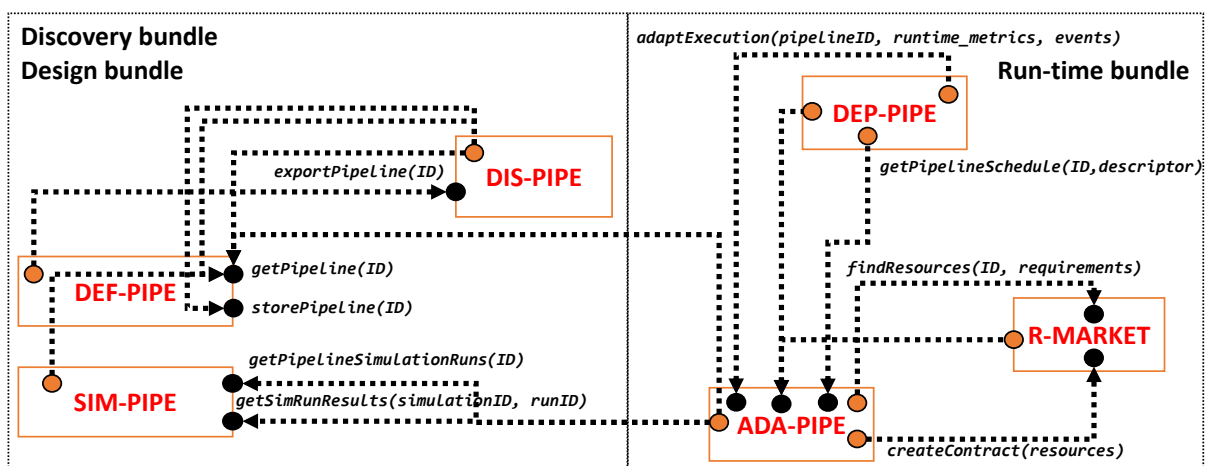


Figure 3: DataCloud tool integration interfaces

While these interfaces and flows were used for the first release, there was also the need to update them. These updates are presented in section 5 and cover a) user interactions with the platform, b) the need for possible pre-deployment or runtime configurations (e.g.,

configuration of credentials etc), c) the runtime adaptation part and d) having the overall goal of providing a tighter integration among platform components.

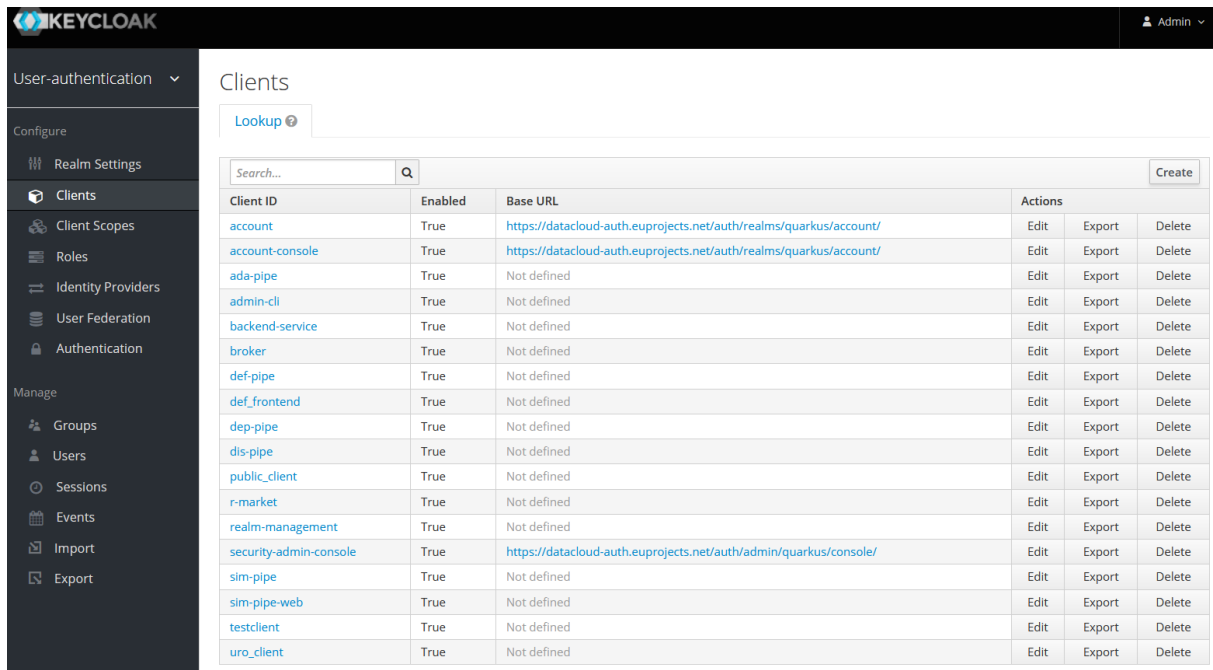
2.2 TOOLBOX OVERVIEW

At least in the first phase of the project, we considered that DataCloud will target in the creation of loosely couple components, that will be able to used as standalone services, and the same time will be able to be used together as part of a common “toolbox”.

For this integrated toolbox to be realized, additional components had to be provided; these components were be designed, developed, configured and deployed in the scope of WP6. The main components identified are the common Identity Management service and a common user interface. For the second release a common (relational) database and a Kafka queue will also be utilized.

2.2.1 Identity Management Keycloak

An Identity Management Service is essential for creating a well-integrated software solution. It is used to create the appropriate roles, allow users to register and create a personal account in DataCloud, and log into the system. Keycloak¹, an open-source solution, is used for this purpose and has been deployed for the scope of the project.



Client ID	Enabled	Base URL	Actions		
account	True	https://datacloud-auth.euprojects.net/auth/realms/quarkus/account/	Edit	Export	Delete
account-console	True	https://datacloud-auth.euprojects.net/auth/realms/quarkus/account/	Edit	Export	Delete
ada-pipe	True	Not defined	Edit	Export	Delete
admin-cli	True	Not defined	Edit	Export	Delete
backend-service	True	Not defined	Edit	Export	Delete
broker	True	Not defined	Edit	Export	Delete
def-pipe	True	Not defined	Edit	Export	Delete
def_frontend	True	Not defined	Edit	Export	Delete
dep-pipe	True	Not defined	Edit	Export	Delete
dis-pipe	True	Not defined	Edit	Export	Delete
public_client	True	Not defined	Edit	Export	Delete
r-market	True	Not defined	Edit	Export	Delete
realm-management	True	Not defined	Edit	Export	Delete
security-admin-console	True	https://datacloud-auth.euprojects.net/auth/admin/quarkus/console/	Edit	Export	Delete
sim-pipe	True	Not defined	Edit	Export	Delete
sim-pipe-web	True	Not defined	Edit	Export	Delete
testclient	True	Not defined	Edit	Export	Delete
uro_client	True	Not defined	Edit	Export	Delete

Figure 4: Keycloak clients for the integration of DataCloud tools

¹ <https://www.keycloak.org>

Through configured realms and clients, Keycloak can centralize the login process of various systems and components through the implementation of protocols such as OAuth2.0² and OpenID Connect (a.k.a. OIDC)³. Most tools have already integrated Keycloak, which for the public DataCloud toolbox, is deployed at <https://datacloud-auth.euprojects.net/>, and thus, the this first versions of the toolbox features secure communication among the tools and proper Identity Management.

Figure 5 shows an overview of the single sign-on integration in DataCloud, covering front and back-end services.

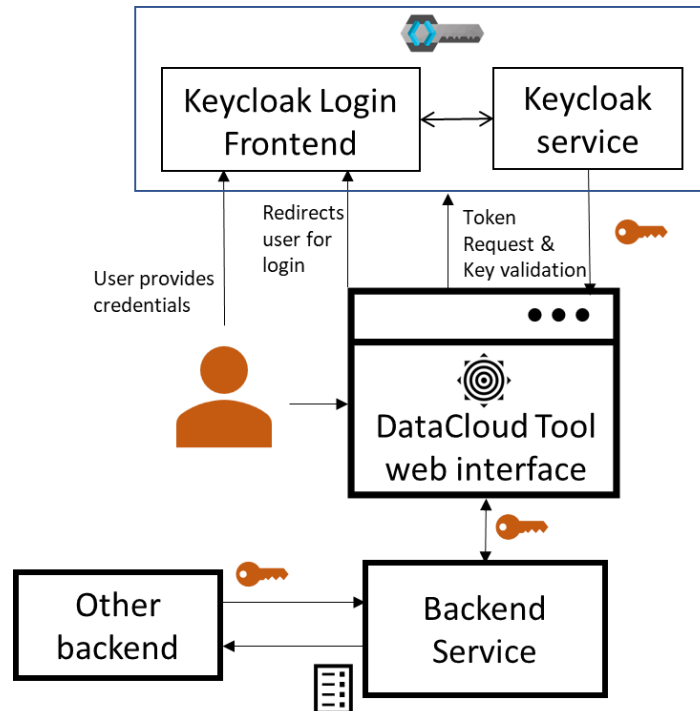


Figure 5: Authentication/authorization in DataCloud

A user loading the web interface of a tools is prompted to log in (unless they have a valid session already). Once the user enters their credentials, they receive a secure token that enables them to access the application. In addition, if a service requests an asset from another DataCloud service (e.g., results of a simulation run is requested by the adaptation engine), it needs to provide a valid user token for the specific asset as part of the API call. The specific way that the userID will be provided has been agreed (based on the Keycloak tokens), while each service is responsible for storing this userID locally so that can find the appropriate user resources (e.g., the defined pipelines, simulations or deployments). This flow is presented in Figure 6.

² <https://oauth.net/2>

³ <https://openid.net/connect>

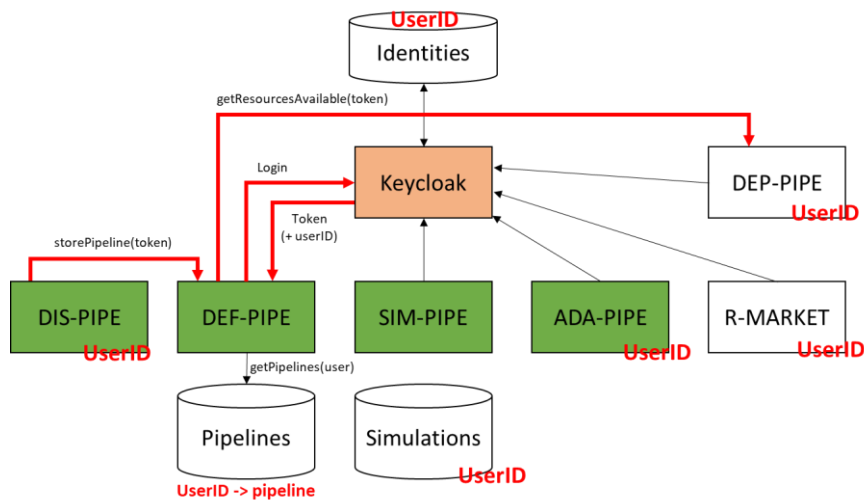


Figure 6: UserID usage across the tools of DataCloud

2.2.2 Common UI

Although in DataCloud, we consider the tools of each step of the lifecycle to be used as standalone services, we also decided that it would be beneficial for the overall user experience of DataCloud to provide a common view that aggregates the functionalities of all tools. Initially, this led to the design of mock-ups, that will allow such basic functionality. Figure 7 and Figure 8 depict the two main suggestions for the first release of the toolbox.



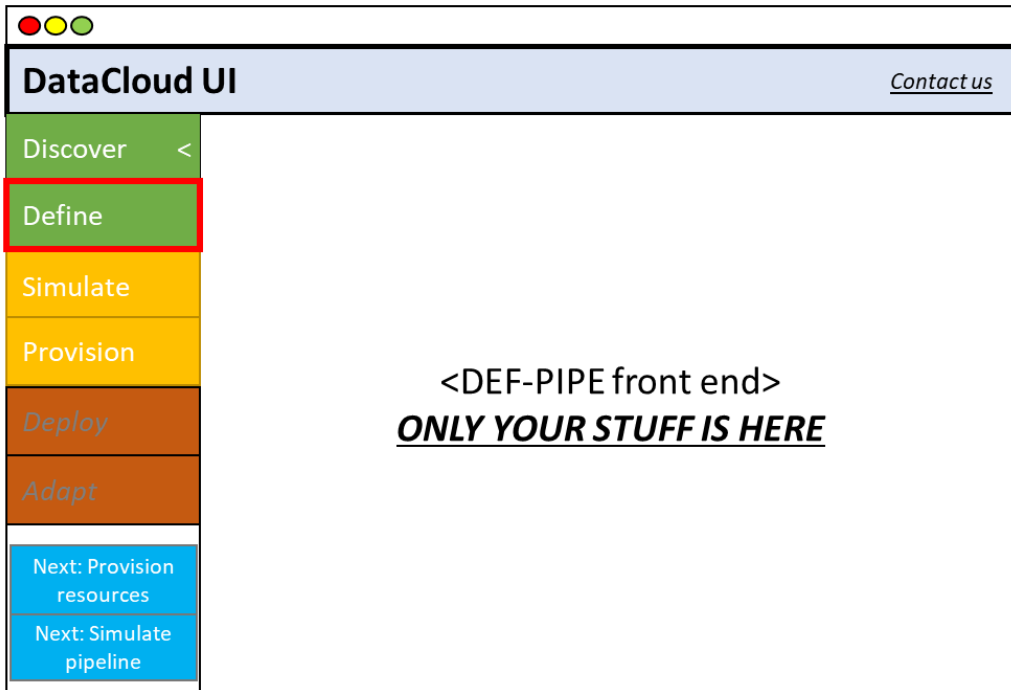


Figure 7: Common UI mock-up (initial proposition)

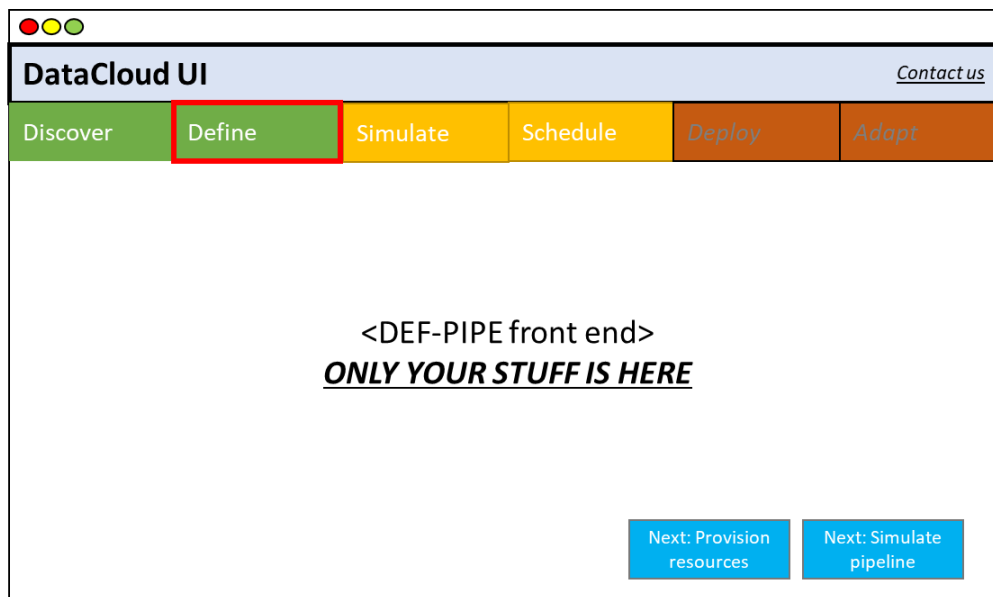


Figure 8: Common UI mock-up (2nd proposition)

With the selection the 2nd proposition, we proceeded with this approach to deliver the first PoC of the Toolbox, and since then, it has been used for accessing the tools deployed for demonstration and testing purposes. The UI of the toolbox is accessible at <https://datacloud-toolbox.euprojects.net>.



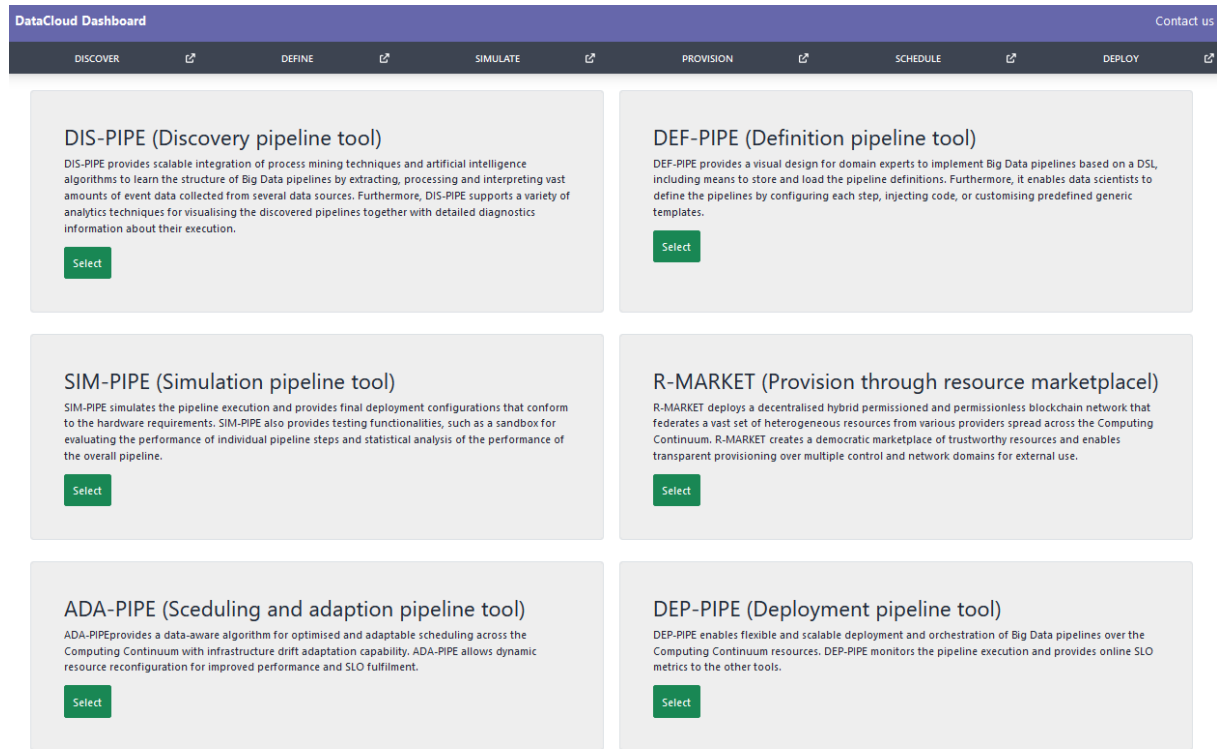


Figure 9: DataCloud common UI

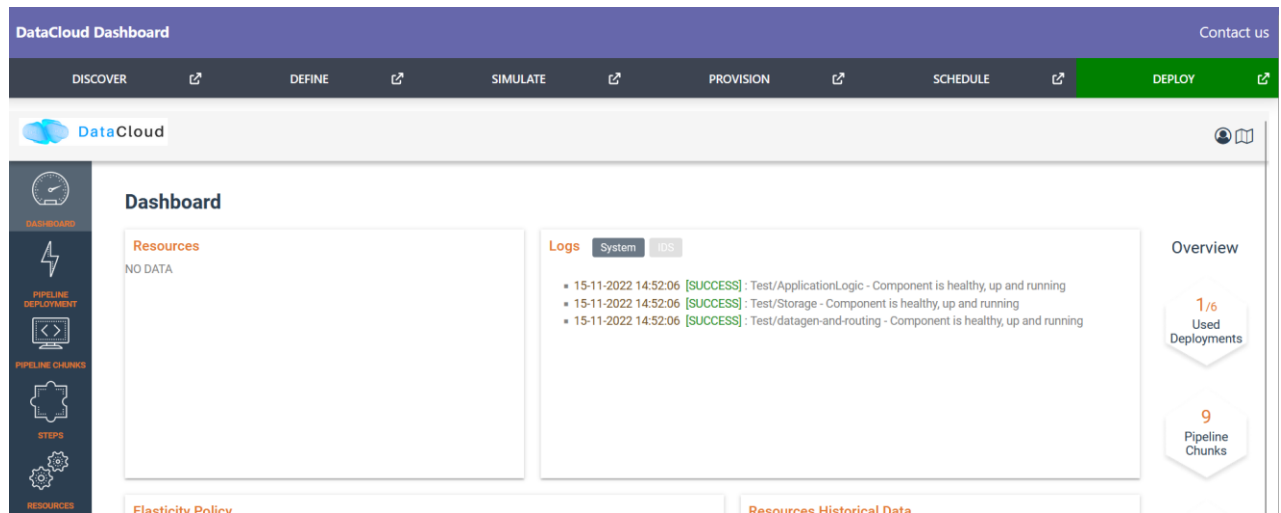


Figure 10: Tool accessed through the DataCloud common UI

The code for this common UI is available at: <https://github.com/DataCloud-project/ALL-PIPE>.

Although it was considered optional for this first integration stage, a relational database can help us to create common identifiers for the platform and to store things such as user-specific data, preferences, and other pipeline-related metadata that has to be moved across

components. If such a database is needed and the usage of Keycloak and MinIO⁴ is not enough, it will be deployed as part of the common UI.

Furthermore, using a message bus can allow the asynchronous communication of the components. This is considered especially in a platform where some actions can take a long time to be executed (e.g., data uploading or data anonymization), and asynchronous calls allow us to provide the user with a smoother user experience. Apache Kafka⁵ is considered for this purpose, and can be used as part of DEP-PIPE.

2.2.3 Deployment

All services are available for testing and are accessible through the toolbox landing page. For the final release all services will be provided with. datacloud subdomain, as this will allow to overcome some Cross-origin resource sharing (CORS) issues and assist on the tighter integration of the platform.

Table 4: Overview of DataCloud tool deployments

Component	IP, port or Current URL	Future common domain
Toolbox	https://datacloud-toolbox.euprojects.net 192.168.3.144:9085	toolbox.demo.datacloudproject.eu
DIS-PIPE	http://94.177.218.70:7778/	dis.demo.datacloudproject.eu
SIM-PIPE	https://simpipesct.sintef.no/graphql	sim.demo.datacloudproject.eu
DEF-PIPE	https://crowdserv.sys.kth.se	def.demo.datacloudproject.eu
ADA-PIPE	http://194.182.187.139:5000/pipelines .	ada.demo.datacloudproject.eu
DEP-PIPE	https://datacloud-dep.euprojects.net/ 192.168.3.144:80	dep.demo.datacloudproject.eu
R-MARKET	http://20.71.159.181/10000	market.demo.datacloudproject.eu
Authentication service	https://datacloud-auth.euprojects.net 192.168.3.120:8180	auth.demo.datacloudproject.eu
Monitoring	http://datacloud-prometheus.euprojects.net 212.101.173.86:30000	monitoring- db.demo.datacloudproject.eu

2.2.4 Enabling technologies

DataCloud Toolbox is a complex offering composed of different tools created by different development teams using different stacks and technologies. Apparently, this creates onboarding overhead for possible contributors, however this cannot be completely resolved in such a research project; we cannot enforce technological stack to the partners developing the components. Nevertheless, we have agreed among the consortium to use technologies

⁴ <https://min.io>

⁵ <https://kafka.apache.org>



and frameworks that can help on the integration and the maintainability of the project. These include the usage of containers for the deployment of all components (most have been using containers in this first release) and also using docker compose when need to deploy complex applications. In addition, for each tool release we have agreed on the release of a container image as well, and also to create a dockerhub organization for hosting all images of DataCloud tools. The usage of the OpenAPI standard⁶ for documenting all APIs is also an important step towards achieving a high platform maintainability goal, that is further enhanced by the usage of Keycloak.

Finally, guides for building, installing and deploying individual components are provided; for the final release this documentation will be consolidated as documentation of the entire toolkit. The common deployment and common domain usage is ongoing and will be achieved in the final release.

⁶ <https://swagger.io/specification/>



3 OVERVIEW OF INTEGRATION AND TOOLBOX COMPONENTS

In this section we present each tool of the toolbox, covering main functionalities, the current status and the integration of each tool, and also presenting where the tool code is hosted.

3.1 PIPELINE DISCOVERY

DIS-PIPE provides scalable integration of process mining techniques and artificially intelligent algorithms to learn pipelines' structure by extracting, processing and interpreting vast amounts of event data collected from several heterogeneous data sources. Furthermore, DIS-PIPE includes a graphical interface that supports various analytics techniques for visualising the discovered pipelines together with detailed diagnostics information about their execution, and a public API that enables external user interaction, including integration with the DEF-PIPE.

DIS-PIPE imports an event log in the XES format, containing traces associated with one or more pipeline executions. DIS-PIPE uses this event log to feed the preprocessing and discovery components for learning the pipeline models underlying the behaviors observed in the log. The `importLog` function can be invoked by clicking a dedicated button in the Map view of the GUI (cf. Figure 11):

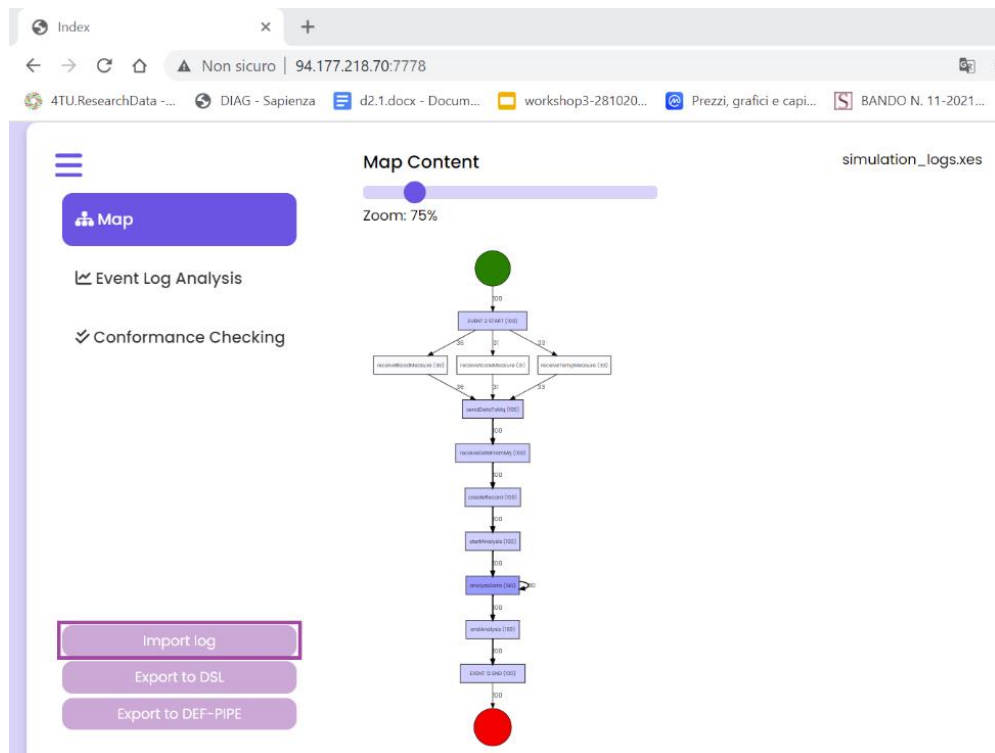


Figure 11: Position in the GUI of the function `importLog(xesLogID)`

More details about the DIS-PIPE implementation are available in the deliverable [D2.3].

3.1.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox.

Table 5: DIS-PIPE features implementation and integrations status

Features	Integration Status
Event Data Analysis: DIS-PIPE is able to analyse event data collected from various sources and related to a single (or multiple) data pipeline(s).	Already implemented in DIS-PIPE
Interpretation of Partially Ordered Event Data: DIS-PIPE interprets event data that are recorded in a partially ordered way, i.e., without a clear timestamp associated with them or with a timestamp having a coarse granularity (e.g., days).	Already implemented in DIS-PIPE
Low-frequency Trace Filtering: DIS-PIPE is able to recognise traces of event data with a low execution frequency and filter out them from the subsequent analysis.	Already implemented in DIS-PIPE
Rule Filter: DIS-PIPE enable to specify complex rules in the form of ordering constraint between pipeline steps.	Already implemented in DIS-PIPE
Event Logs Generation and Interpretation: DIS-PIPE is able to produce and interpret event logs formatted in XES, MXML, or CSV.	Already implemented in DIS-PIPE
Big Data Pipeline Discovery: DIS-PIPE is able to discover big data pipelines starting from event logs formatted in XES, MXML, or CSV	Already implemented in DIS-PIPE
Pipeline Expressiveness: DIS-PIPE is able to discover big data pipelines, including XOR, OR, AND, loops, and data association constructs.	Already implemented in DIS-PIPE
Pipeline Visualisation and Description: DIS-PIPE allows users to have a visualisation and a description of the running pipelines.	Already implemented in DIS-PIPE
Pipeline Analytics: DIS-PIPE computes analytics on the discovered pipelines.	Already implemented in DIS-PIPE
Conformance Checking: DIS-PIPE performs conformance checking on the discovered pipelines and the ones developed in the past.	Already implemented in DIS-PIPE
Anomaly Detection: DIS-PIPE interprets the compliance checking outcomes signaling nonconforming deviations and providing repair solutions	Already implemented in DIS-PIPE

Table 6: DIS-PIPE Features Planned for second release of the toolbox

Features	Integration Status
Interface Functions: Development of the missing interface functions to import a DSL pipeline from DEF-PIPE and to export an event log in XES format.	To be implemented within 30/11/2022
Dark Data Discovery and Exploitation: Development of a	To be implemented within 31/01/2023



querying algorithm to extract information related to the dark data manipulated by the data pipeline.	
Segmentation and Trace Preprocessing: Implementation and testing of the algorithm that perform segmentation and trace preprocessing, detailed in [D2.1].	To be implemented within 28/02/2023
Event Abstraction: Implementation and testing of the algorithm that perform event abstraction, detailed in [D2.1].	To be implemented within 30/03/2023

3.1.2 Integration with the Toolbox Components

DIS-PIPE interfaces with DEF-PIPE using the following two functions:

1. `importPipeline(pipelineID)` imports a pipeline represented in the DSL format, potentially modified by DEF-PIPE. DIS-PIPE analyses the model's conformance compared to the actual event logs, detects misalignments between the observed behavior stored in the log and the new expected behavior and suggests recovery or realignment strategies.
2. `exportPipeline(pipelineID)` exports a discovered pipeline model converting it into DSL format (as returned by DEF-PIPE). Specifically, DEF-PIPE invokes the `exportPipeline(pipelineID)` interface provided by DIS-PIPE to load a discovered DSL pipeline model and enhance/customize its description using its visual workbench. We implemented `exportPipeline(pipelineID)` as a Flask REST-API accessible via SwaggerUI⁷, as shown in Figure 12.

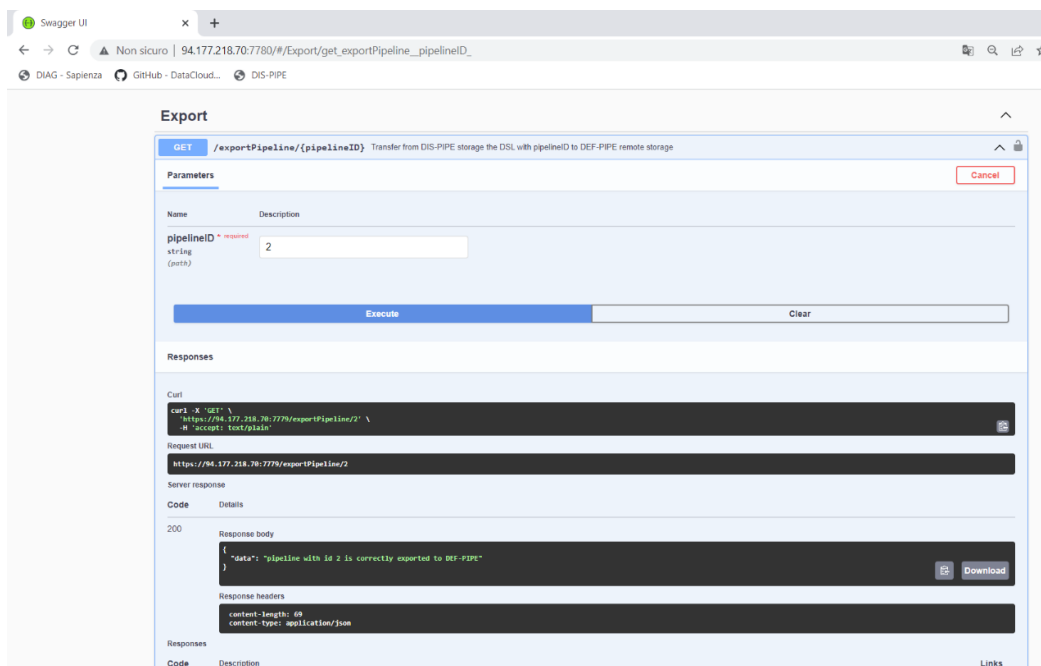
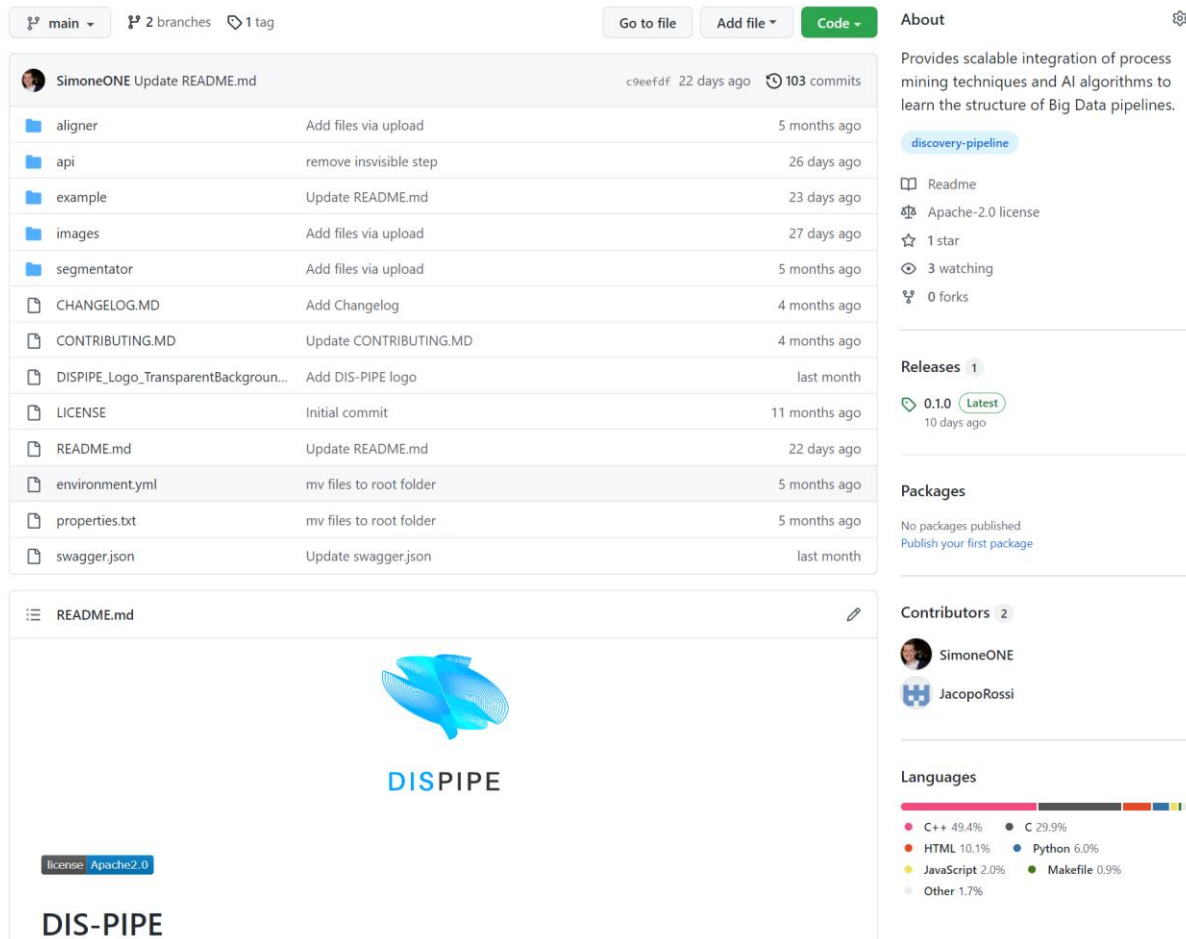


Figure 12: `exportPipeline(pipelineID)` REST-API

⁷ <http://94.177.218.70:7780/>

3.1.3 Deployment, Code and Documentation Availability

Concerning DIS-PIPE code and documentation, DIS-PIPE it is available on GitHub at the following link: <https://github.com/DataCloud-project/DIS-PIPE>. In Figure 13 it is reported an overview of the GitHub page.



The screenshot displays the GitHub repository for DIS-PIPE. The main content area shows a list of files and folders with their respective commit dates. The README.md file is selected and its content is visible, featuring the DIS-PIPE logo and the Apache 2.0 license. The right sidebar provides additional details about the repository, including a description, a 'discovery-pipeline' tag, 1 star, 3 watchers, 0 forks, 1 release (0.1.0, latest), and 2 contributors (SimoneONE and JacopoRossi). A language usage chart shows C++ at 49.4%, HTML at 10.1%, JavaScript at 2.0%, Other at 1.7%, C at 29.9%, Python at 6.0%, and Makefile at 0.9%.

Figure 13: DIS-PIPE GitHub page

As shown in Figure 14, DIS-PIPE is deployed on a single host accessible from the browser at following link: <http://94.177.218.70:7778/>. As an incoming request to access is triggered by the browser, it is managed and processed by the backend of the tool, which is in charge to load the GUI and allow users to interact with its provided functionalities.

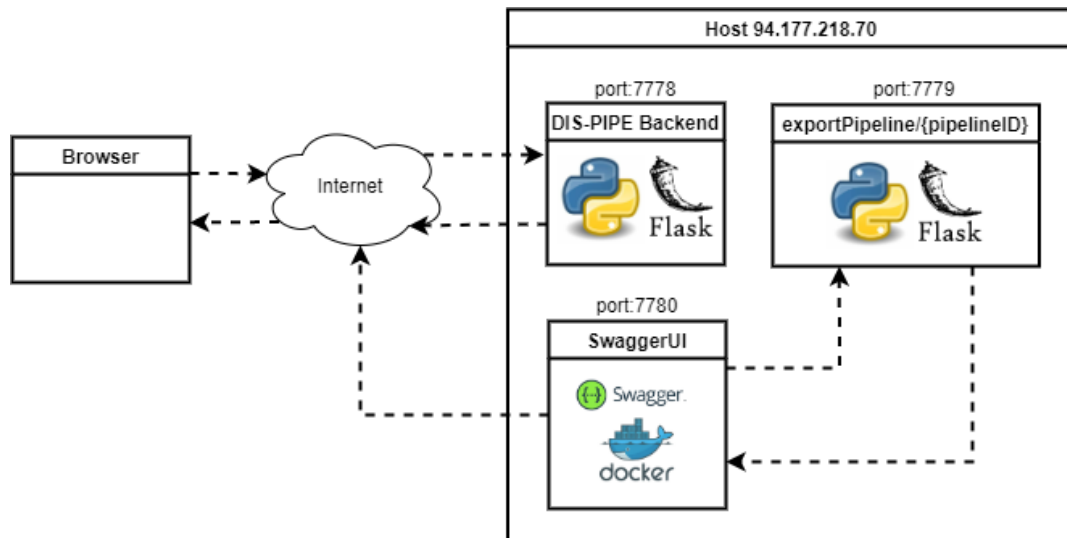


Figure 14: Deployment of DIS-PIPE Architecture and REST-APIs

3.2 PIPELINE DEFINITION

In this section we present the first part of the Big Data pipeline definition and simulation tooling, DEF-PIPE. DEF-PIPE provides a visual design for domain experts to implement Big Data pipelines based on a DSL, including means to store and load the pipeline definitions. Furthermore, it enables data scientists to define the pipelines by configuring each step, injecting code, or customising predefined generic templates.

The DEF-PIPE Frontend is a graphic pipeline designer tool for defining Big Data pipelines and transforming them to DSL. The main part of the application is the interface for designing big data pipelines. This interface is implemented as a single page application using ReactJS. The popularity and stability of ReactJS make it potentially more friendly with developers to continue with the project later on. The project also use Bootstrap, a framework providing basic UI components building blocks which are easy to customize.

Back-End is implemented in CSharp using Dot Net (.NET) framework from Microsoft. In particular, ASP.NET Core, which is the part of the NET framework for web application, is being used. It implements a web API providing a central interface for operations such as managing pipelines and templates data, transforming pipelines into DSL.

Database of Pipeline Designer is used to persist steps and workflow created by users. As the visual workflows are represented in JSON format, MongoDB is used.

DEF-PIPE has integrated with Keycloak as single sign on solution and offers user asset management for steps and pipelines including the ability to expose them as public for sharing with other users. The front end includes improved support for parametrization of steps.

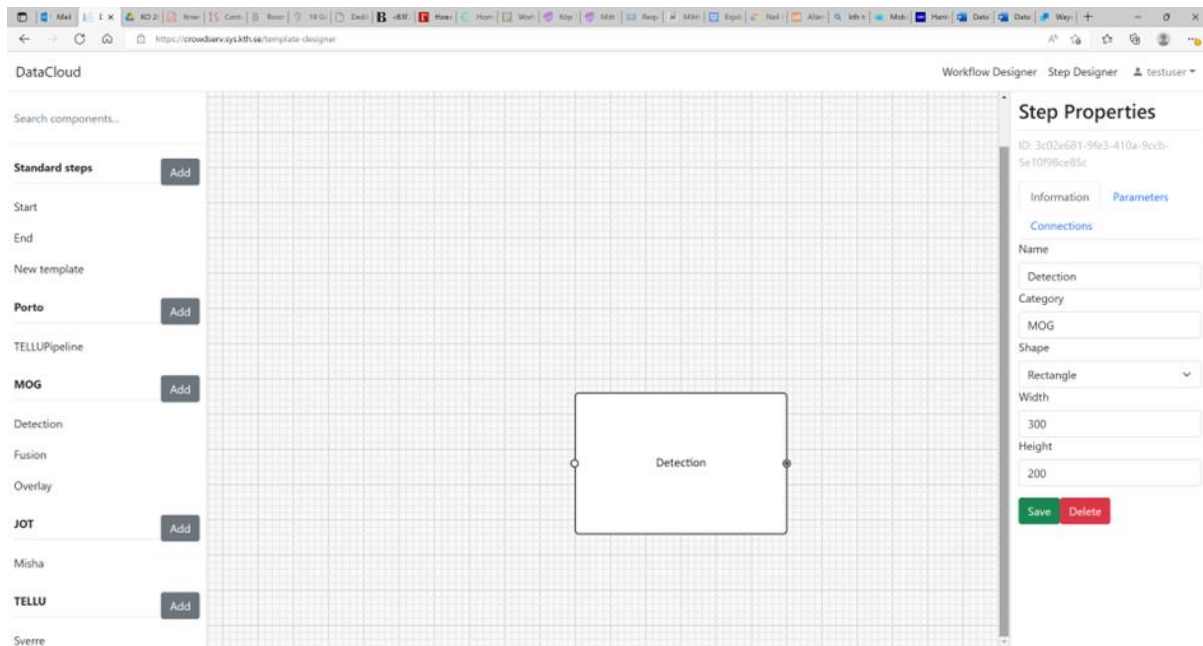


Figure 15: Step Designer mode

Regarding pipeline representation, the tooling provides various improvements of the domain-specific language (DSL), including a grammar for specifying DSL models and implementation of editors (autocomplete + validation) in the Eclipse environment [D3.3].

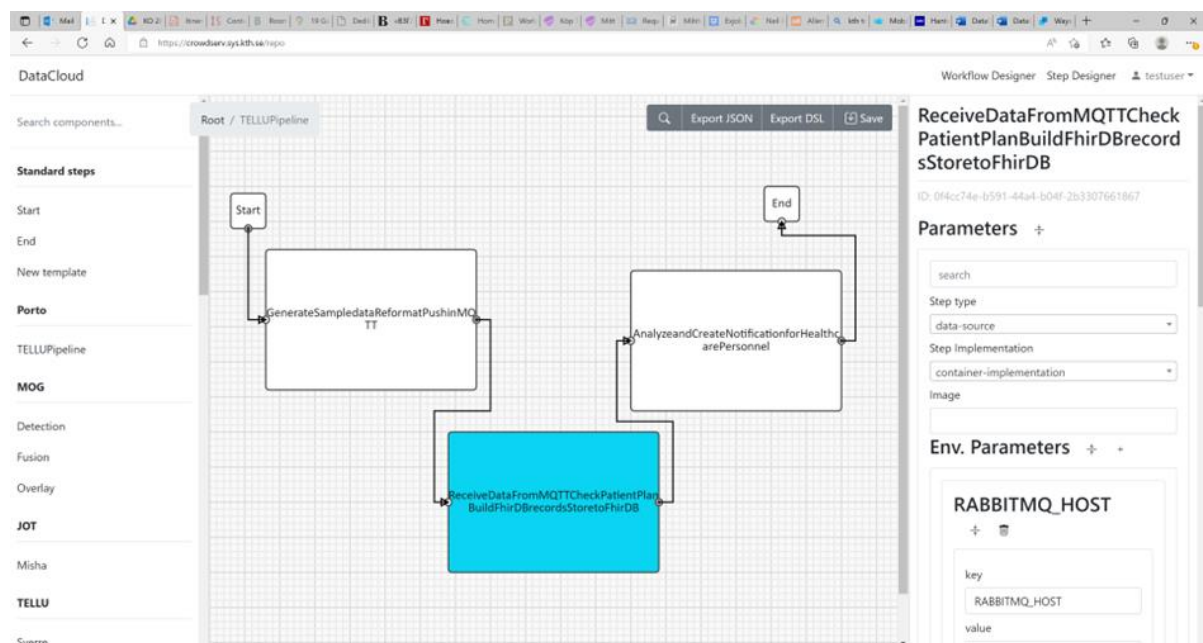


Figure 16: Workflow Designer mode

To start working with the DEF-PIPE graphical tool, use the following link:

- <https://crowdserv.sys.kth.se>

A quick start guide is available on GitHub:

- <https://github.com/DataCloud-project/DEF-PIPE-Frontend>

3.2.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox. The DEF-PIPE core features and architecture have been described and depicted in deliverable [D3.2], while improvements to the DEF-PIPE are also presented in [D3.3].

Table 7: DEF-PIPE features implementation and integrations status

Features	Status
Provide user based management of pipelines	Done, libraries of steps and pipelines associations with a user are implemented. each user has own list of steps and pipelines
Support private and public pipelines	Steps and Pipelines can be private or public. Public pipelines and steps can be used/copied by other users into their list of steps and pipelines. Private elements are not visible and can't be copied by other users
Integration with IAM for single sign-on	DEF-PIPE has been integrated with Keycloak system to support the authentication of users (single sign-on).
Improvements on the UI based on the feedback	Menu style for description of parameters is implemented. Parameters of steps are now presented in the right panel of the screen. User can choose which of the parameters are relevant to his/her description and insert or update them. This feature allows user describe steps and pipelines without knowledge of the DSL which is generated from the graphical representation
Integration with DEP-PIPE	Partially ready

Table 8: DEF-PIPE Features Planned for second release of the toolbox

Features	Status
Integration with SIM-PIPE	Ongoing
Integration with DIS-PIPE	Ongoing
Integration with ADA-PIPE	Ongoing

3.2.2 Integration with the Toolbox Components

Integration with other DataCloud components is currently under implementation via APIs. The APIs allow a DSL description of a discovered pipeline by the DIS-PIPE tool to be presented and edited in the graphical DEF-PIPE tool. DEP-PIPE and SIM-PIPE tools are already integrated and using the provided API.

DEF-PIPE APIs are described at:

- <https://crowdserv.sys.kth.se/docs>

Figure 3 shows a screenshot of the API documentation web page.



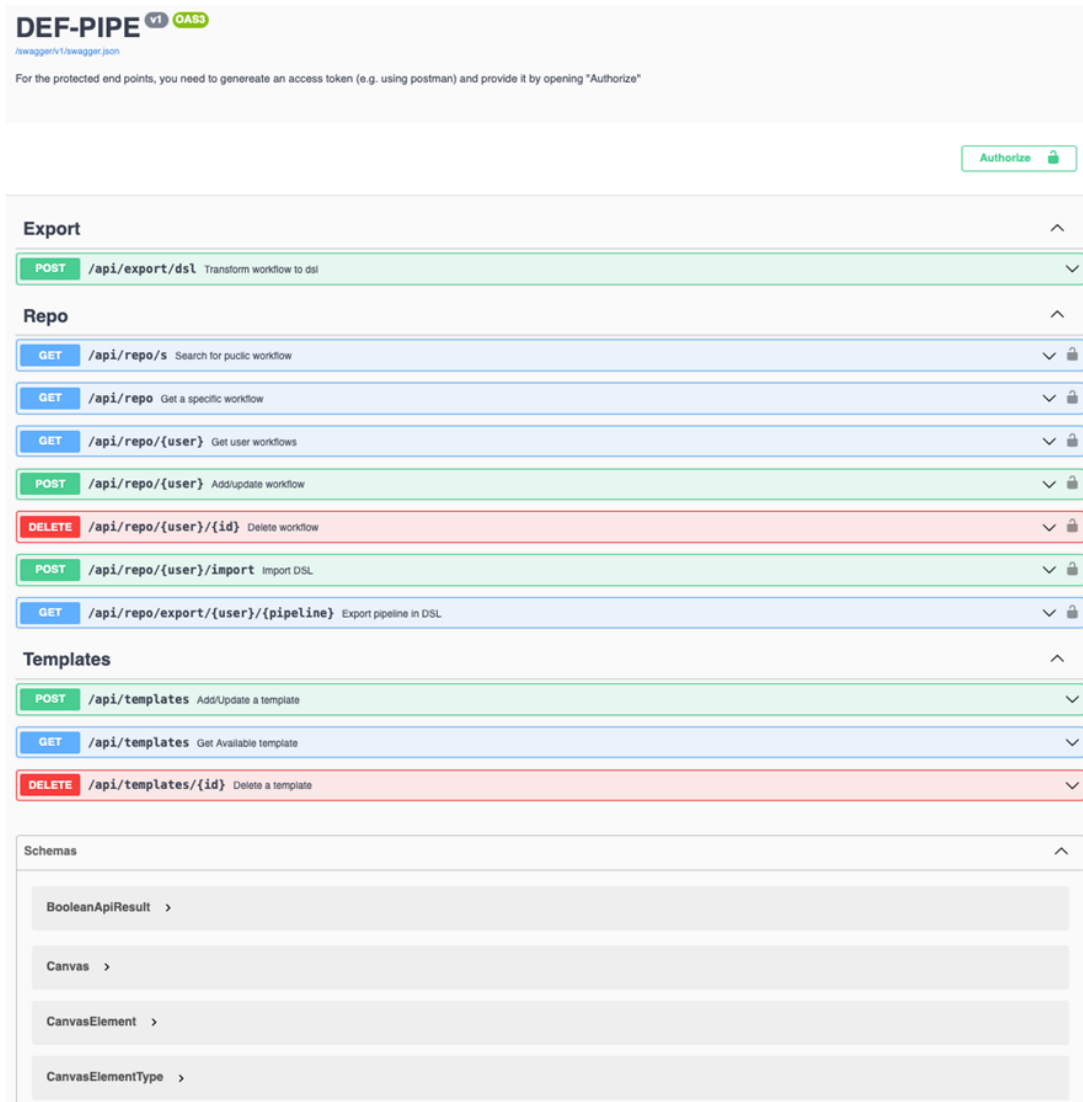


Figure 17: DEF-PIPE API documentation web page

3.2.3 Deployment, Code and Documentation Availability

DEF-PIPE is available on GitHub, along with dedicated instructions for usage

- <https://github.com/DataCloud-project/DEF-PIPE-Frontend>

The DEF-PIPE graphical tool is also deployed at

- <https://crowdserv.sys.kth.se>



3.3 PIPELINE SIMULATION

SIM-PIPE simulates the pipeline execution and provides final deployment configurations that conform to the hardware requirements. SIM-PIPE also provides testing functionalities, such as a sandbox for evaluating the performance of individual pipeline steps and statistical analysis of the overall pipeline performance.

SIM-PIPE takes a pipeline's definition as input and outputs runtime metrics, such as CPU usage, memory usage, energy consumption, run durations, and network bandwidth. SIM-PIPE can perform dry runs of pipelines by running small and brief versions of the pipelines in a sandbox. These dry runs improve the simulations' accuracy for deployments at larger scales and assert whether one pipeline executes successfully.

SIM-PIPE consists of the following components: a web graphical user interface, a controller with an API, a relational time-series database, a simulation engine, and a sandbox to execute the dry runs.

The key innovation implemented in SIM-PIPE leverages the container-based approach for data pipelines and is related to the following aspects:

- Novel means of leveraging container-based approach for data pipelines with sample data and different configurations to perform eventual simulations.
- A dry run approach for generating inputs for simulators.

More details about the SIM-PIPE implementation are available in the deliverable [D3.3].

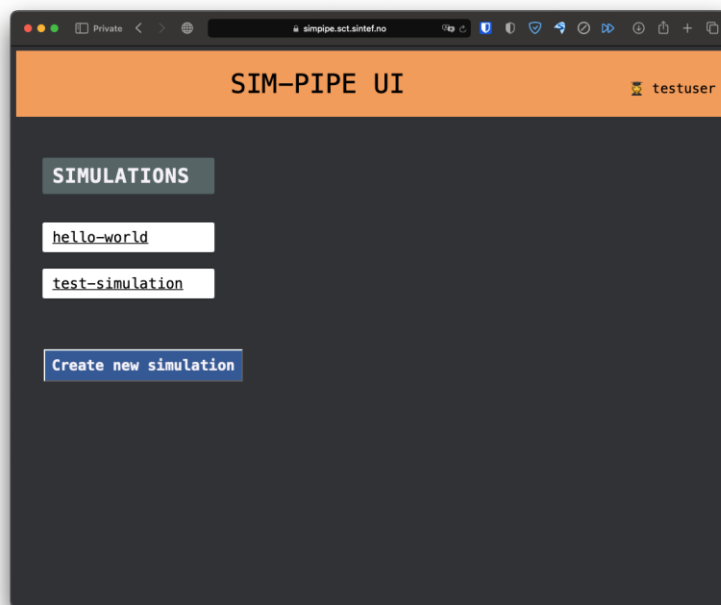


Figure 18: Screenshot of SIM-PIPE's user interface

3.3.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox. SIM-PIPE's features are

tracked in GitHub. The SIM-PIPE GitHub project has been configured according to the GitHub guidelines described in [D6.2].

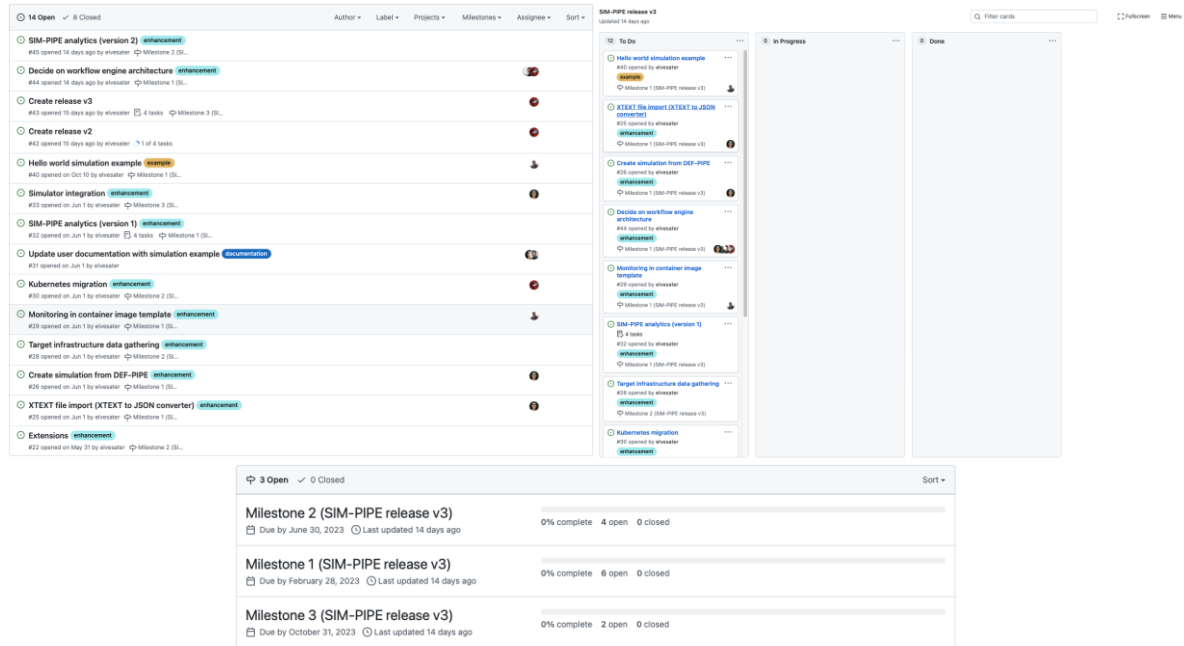


Figure 19: Screenshots of software management organisation on GitHub

Table 4 and table 5 present a summary of the features already implemented and going to be implemented and integrated with other tools.

Table 9: SIM-PIPE features implementation and integrations status

Features	Status
GraphQL API	Implemented and tested
Authentication and authorisations	Implemented and tested
Dry run execution	Implemented and tested, API available
Secure sandbox environment	Implemented and tested, API available
Metrics collection	Implemented and tested, API available

Table 10: SIM-PIPE Features Planned for second release of the toolbox

Features	Status
More data retrievable queries in GraphQL API	N/A
Access to DEF-PIPE catalogue for quick testing	Waiting.
Integration of simulation engine	To be implemented.
Support more advanced pipelines.	To be implemented.



3.3.2 Integration to the Toolbox Components

SIM-PIPE provides a GraphQL API. GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

We have not yet integrated the current version of SIM-PIPE with other components. We plan to integrate DEF-PIPE and SIM-PIPE to simulate a pipeline in one click from DEF-PIPE. We will implement this feature in 2023 after DEF-PIPE implements its catalogue of pipelines.

We authenticate the API using JWT tokens provided by the Keycloak instance of the toolbox. By the integration of such mechanism, users can only access their pipelines and simulation data and metadata.

3.3.3 Deployment, Code and Documentation Availability

The SIM-PIPE source code is available in a single Git repository hosted on GitHub: <https://github.com/DataCloud-project/SIM-PIPE> It is released as open-source software under the Apache License 2.0.

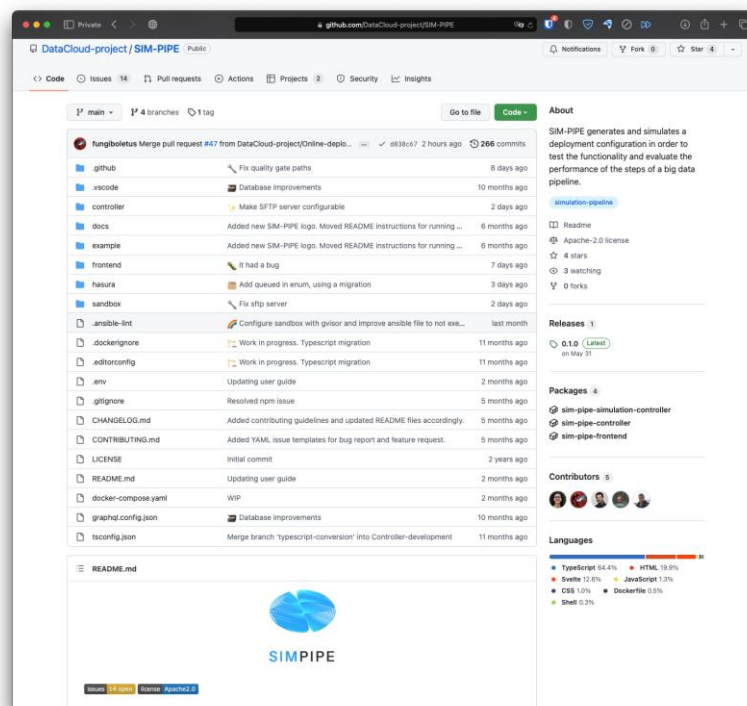


Figure 20: Screenshot of the SIM-PIPE Git repository hosted on GitHub

The SIM-PIPE technical documentation, architecture description, deployment and installation guides, and user guides with examples, and contributing guide are also available in the Git repository.

3.4 PIPELINE SCHEDULING AND ADAPTATION

ADA-PIPE provides a data-aware algorithm for optimised and adaptable scheduling across the Computing Continuum with infrastructure drift adaptation capability. ADA-PIPE allows dynamic resource reconfiguration for improved performance and SLO fulfilment.

3.4.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox.

Table 11: ADA-PIPE features implementation and integrations status

Features	Status
Scheduling tool implementation for QoS Guarantee for Tasks with Strict Deadlines and Data-Aware Pipeline Scheduling	Integrated
Adaptation algorithm with support of Limited Dynamic Scheduling; Avoidance of Highly Utilized Resources;	In integration phase

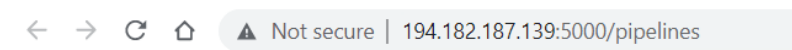
Table 12: ADA-PIPE features planned for second release of the toolbox

Features	Status
Adaptation policies, Scheduling and scaling up/down the data pipeline executions;	In progress
Monitoring policies, Utilization of resources and pipeline chunks.	In progress

3.4.2 Integration with the Toolbox Components

ADA-PIPE tool is running a European Exoscale Cloud virtual machine accessible at

- <http://194.182.187.139:5000/pipelines>



Inputs and output of ADA-PIPE



Figure 22: ADA-PIPE inputs and output provided by/to other DataCloud tools

Swagger UI and the API can be found in the following URL:

- <http://194.182.187.139:5000/swagger/>



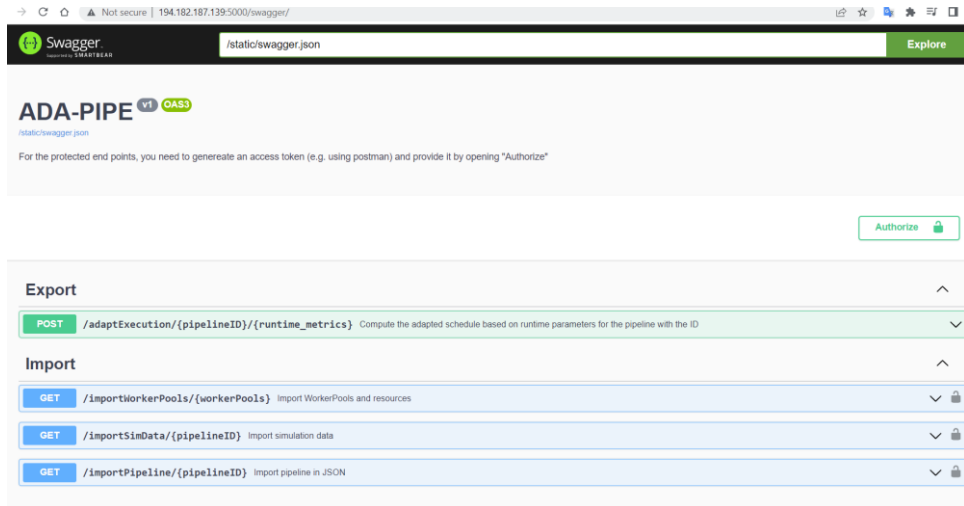


Figure 23: ADA-PIPE Swagger API

For instance, when ADA-PIPE plans to adapt the scheduling plan of a specific pipeline and the runtime metrics, the POST HTTP method response looks as follows:

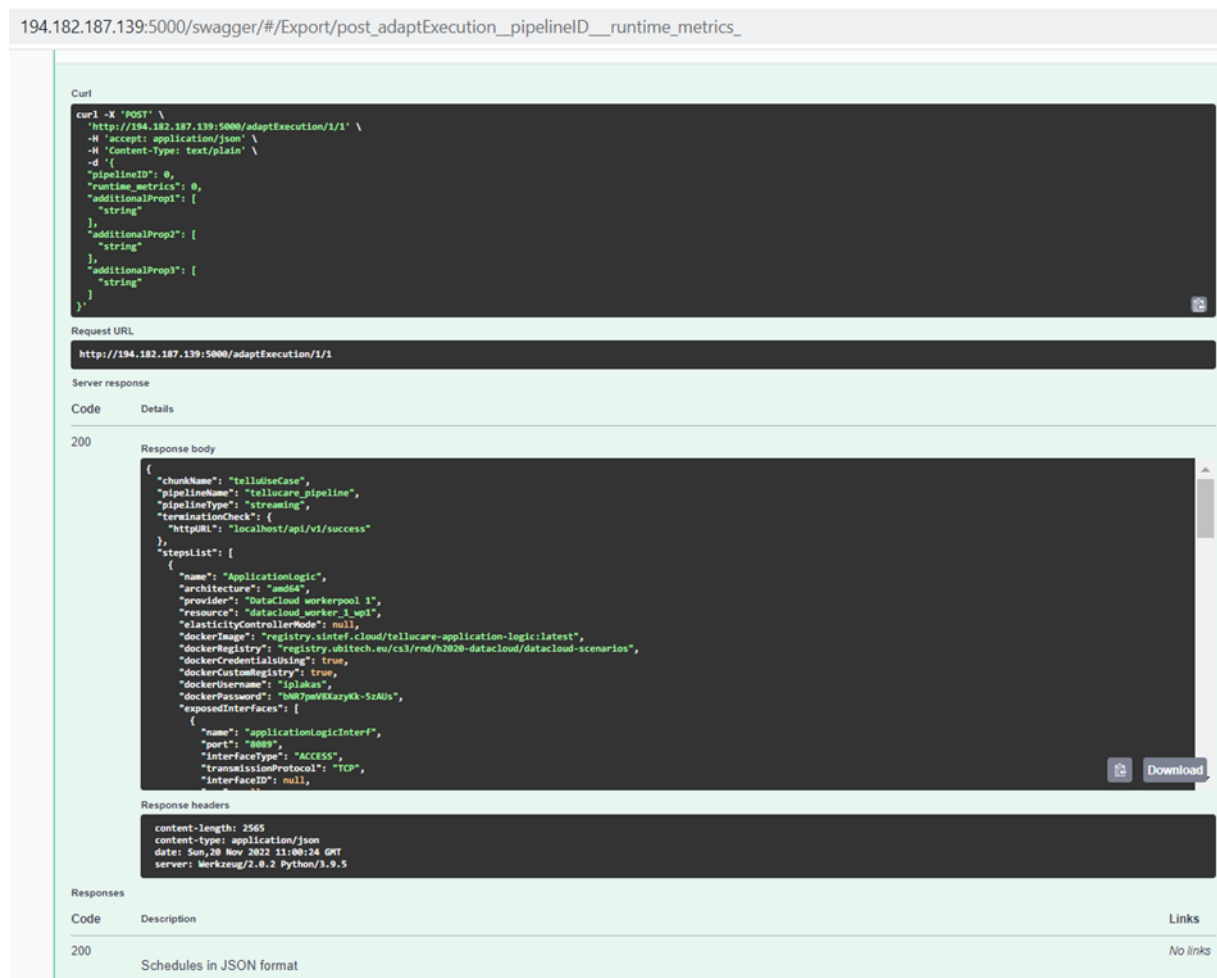


Figure 24: ADA-PIPE POST HTTP request & response

3.4.3 Deployment, Code and Documentation Availability

Mainly the source codes for the ADA-PIPE tool are available in the repository of <https://github.com/DataCloud-project/ADA-PIPE>. It is categorized based on the frontend, matching-based, and machine learning-based (recurrent neural network) source codes available. The user is also able to find other information regarding one data pipeline workflow and a testbed on which the pipeline execution was tested.

The inputs and output (imports and export) to/from the ADA-pipe tool is available in this figure. Currently, the tool works based on the matching-based scheduling strategy and the integration of its source code with the Kubernetes is available in the DataCloud public code repository:

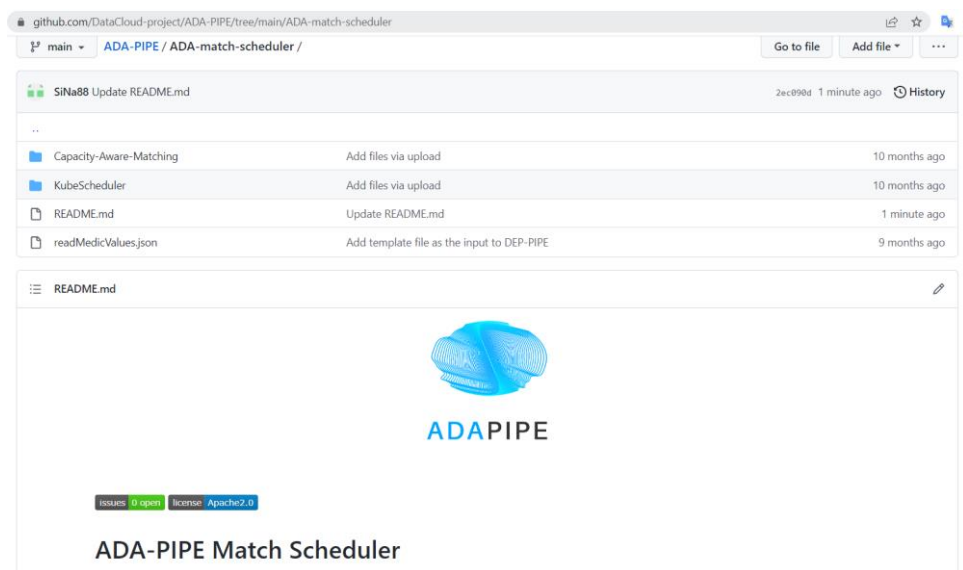


Figure 25: ADA-PIPE matching-based tool tutorial.

ADA-match-scheduler provides a capacity-aware matching-based scheduler for data pipeline processing across the computing continuum. The base model of the matching-based scheduler requires the following Python libraries: networkx, operator, numpy, yaml, json. The frontend part is hosted in the <https://github.com/DataCloud-project/ADA-PIPE-Frontend> repository, as shown in the figure below.

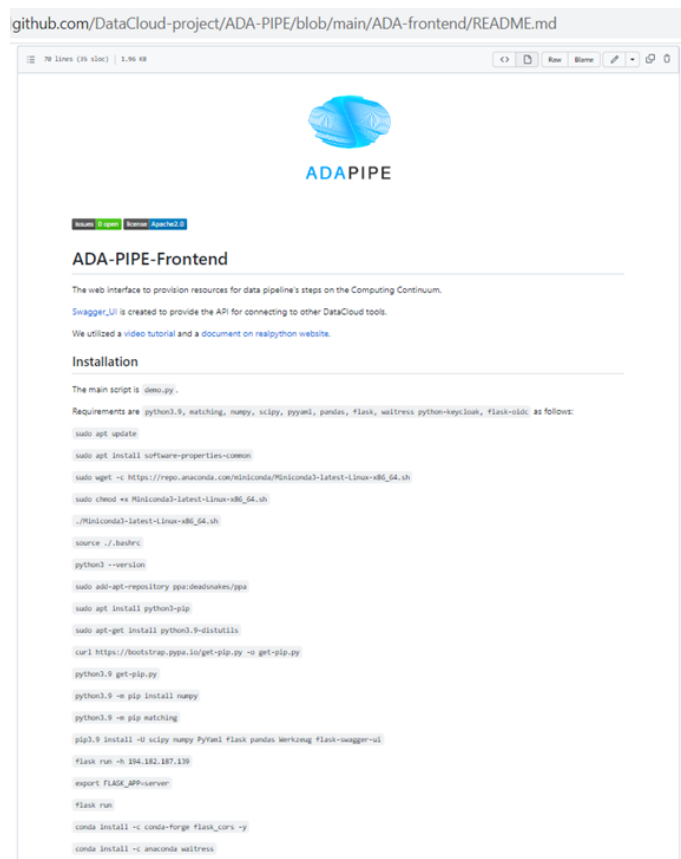


Figure 26: ADA-PIPE Front page setup tutorial

3.5 DECENTRALIZED RESOURCE MARKETPLACE (R-MARKET)

R-MARKET deploys a decentralised hybrid permissioned and permissionless blockchain network that federates a vast set of heterogeneous resources from various providers across the Computing Continuum. R-MARKET creates a democratic marketplace of trustworthy resources and enables transparent provisioning over multiple control and network domains for external use.

3.5.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox.

Table 13: R-MARKET features implementation and integrations status

Features	Status
R-MARKET APIs have been developed for hiding the Blockchain complexity	Deployed in AZURE testbed of the project and integration with R-MARKET component is on-going
Addition of Edge Server to Marketplace	Locally tested and waiting for adding the R-MARKET testbed
Support longer running task (Service-Task)	PoCo Smart Contract modified and tested on-chain computation part

Table 14: R-MARKET features planned for second release of the toolbox

Features	Status
Support complex DSL in R-MARKET SDK	Work-in-Progress
Support longer running task in off-chain	Waiting for completion the deployment of on-chain modification
Share IP addresses of provisioned worker	Working on the integration with Maestro agent

3.5.2 Integration with the Toolbox Components

R-MARKET is a composite tool, which developed by getting inspired of the iExec Marketplace⁸. All the related component of the R-MARKET are listed and available to the GitHub repository (<https://github.com/DataCloud-project/R-MARKET>). The main entry point of the R-MARKET tool is the R-MARKET API (https://github.com/DataCloud-project/R_MARKET_NodeJS), which is a Node.JS API server and it helps ease the interaction between other component tool and R-MARKET tool by hiding the complexity of Blockchain technology. This API server is deployed over the Microsoft Azure provisioned VM and available at: <http://20.71.159.181/10000>.

For authenticating the R-MARKET Node.JS API calls, the Keycloak authentication has been done and integrated with the R-MARKET Node.JS API server.

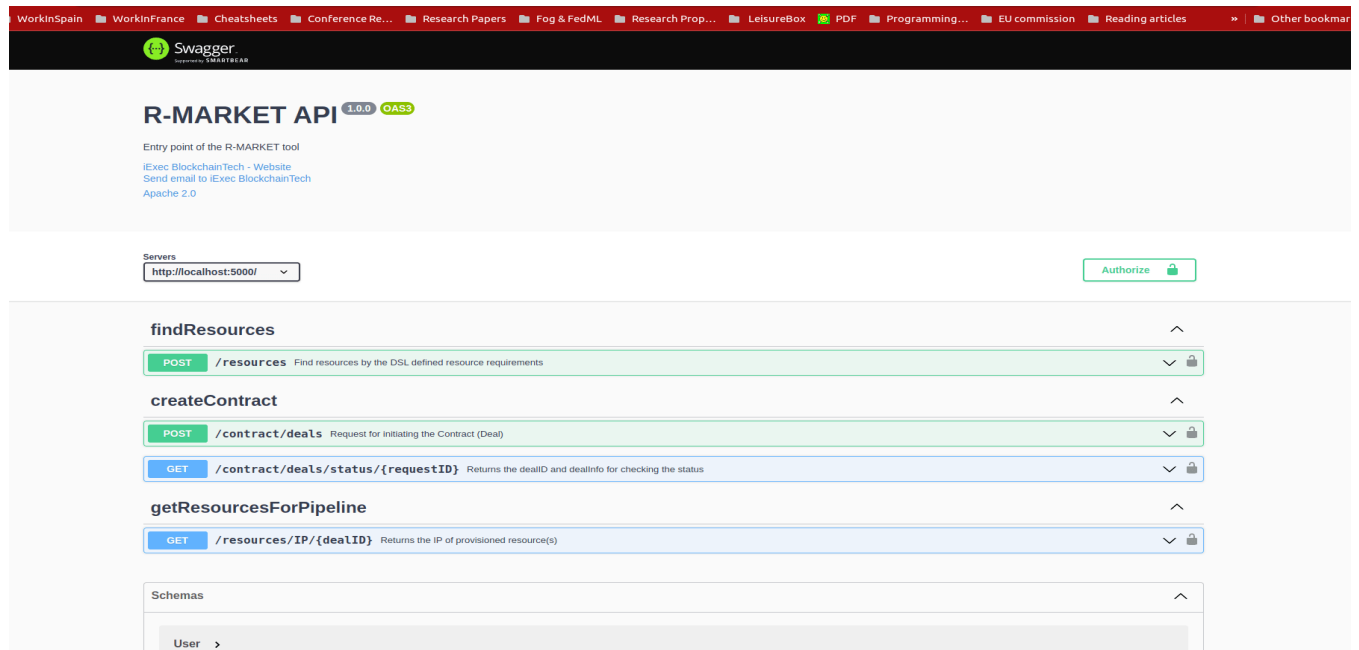


Figure 27: Swagger Implementation of R-MARKET Node.JS API server

⁸ <https://market.iex.ec>

Beside the R-MARKET Node.JS API, we have already deployed the All-in-One Blockchain node over the Microsoft Azure provisioned VM, for permanently keeping the deal information and transactions into the Blockchain. However, this Blockchain node is not publicly exposed for access. In addition, we have already deployed two workerpools along with scheduler and workers have been hosted over two Microsoft Azure provisioned VMs. The details of these two workpools are available to the following link:

- Workerpool 1: <http://20.71.159.181:30000/>
- Workerpool 2: <http://20.71.170.151:30000/>

3.5.3 Deployment, Code and Documentation Availability

In the main GitHub repository of the R-MARKET (<https://github.com/DataCloud-project/R-MARKET>), all the corresponding repositories for R-MARKET tool has been documented. Notably, R-MARKET has seven different repositories for the various R-MARKET components (e.g., Node.JS Server, Market-API, Scheduler, Worker, etc.) followed up with one repository for R-MARKET SDK. All the source code and user-instruction/guidelines for each individual tool can be found in the corresponding repositories.

For showcasing the workerpool details, the dashboard for workerpool has been integrated with the toolbox demo, which is accessible to the following link: <https://datacloud-toolbox.euprojects.net/#/provision>.

3.6 PIPELINE DEPLOYMENT & MANAGEMENT

DEP-PIPE enables flexible and scalable deployment and orchestration of Big Data pipelines over the Computing Continuum resources. DEP-PIPE monitors the pipeline execution and provides online SLO metrics to the other tools.

DEP-PIPE has been built to support a framework agnostic approach built on top of OS-level virtualisation (containers) techniques to allow the deployment of pipelines developed through different data analytics tools. DEP-PIPE also utilises resources offered by public cloud providers but will exploit the availability of edge resources to improve latency and cost. The Blockchain-based decentralised Marketplace will control the offering and the management of the edge resources for resources provided by R-MARKET.

DEP-PIPE will deal with the challenge of deployment and orchestration of services deployed across the compute continuum. Existing state-of-the-art approaches consider the edge and the cloud devices in isolation; therefore, the main challenge is the unification of cloud, fog, and edge resources for both the deployment and orchestration of distributed applications.

With DEP-PIPE, we extend existing orchestration concepts from Kubernetes⁹ and MAESTRO¹⁰ software platform of UBITECH to render them suitable for Edge and Fog

⁹ <https://kubernetes.io>

¹⁰ <https://themaestro.ubitech.eu>



environments. This results in novel orchestration concepts for highly distributed applications that will also include workflow orchestration. These concepts will unify the heterogeneous resources of different domains into a common orchestration engine, advancing over existing solutions by offering novel deployment and orchestration features with respect to workload colocation, workload distribution, and resource selection [D5.2].

3.6.1 Features Implementation

In this section we provide an overview of the features implemented and integrated with other tools, and also the plan for the second release of the toolbox.

Table 15: DEP-PIPE features implementation and integrations status

Features	Status
Graphical user interface provided	Integrated with DEF-PIPE and ADA-PIPE
Allow the initialization of a deployment from the UI	Integrated with DEF-PIPE and ADA-PIPE
R-MARKET resources usage	Integrated resources from R-MARKET in the initial deployment phase
Deployment of pipelines using containerized steps	The API provided to ADA-PIPE and was tested with use case pipelines
DEP-PIPE apply the deployment or adaptation based on the descriptors provided by ADA-PIPE.	The API provided to ADA-PIPE and was tested with use case pipelines
DEP-PIPE receive resource configurations from R-MARKET through the ADA-PIPE descriptor.	Integrated, also integration needed in order to setup the VMs properly

Table 16: DEP-PIPE features planned for second release of the toolbox

Features	Status
Multi-cloud support	Feature under implementation, and integration with ADA-PIPE to be achieved for the second release
Security Policies Enforcement	Feature under implementation, no further integration actions are needed
Scale cluster	Feature under implementation, and integration with ADA-PIPE to be achieved for the second release
Pre-deployment configuration	Feature under implementation, and integration with ADA-PIPE to be achieved for the second release
Adaptation based on pipeline chunks from ADA-PIPE	Feature implemented, integration and testing with complex pipelines is pending

3.6.2 Integration with the Toolbox Components

For performing a deployment, DEP-PIPE is provided by ADA-PIPE with the pipeline chunk to deploy in the specific resources selected by it.



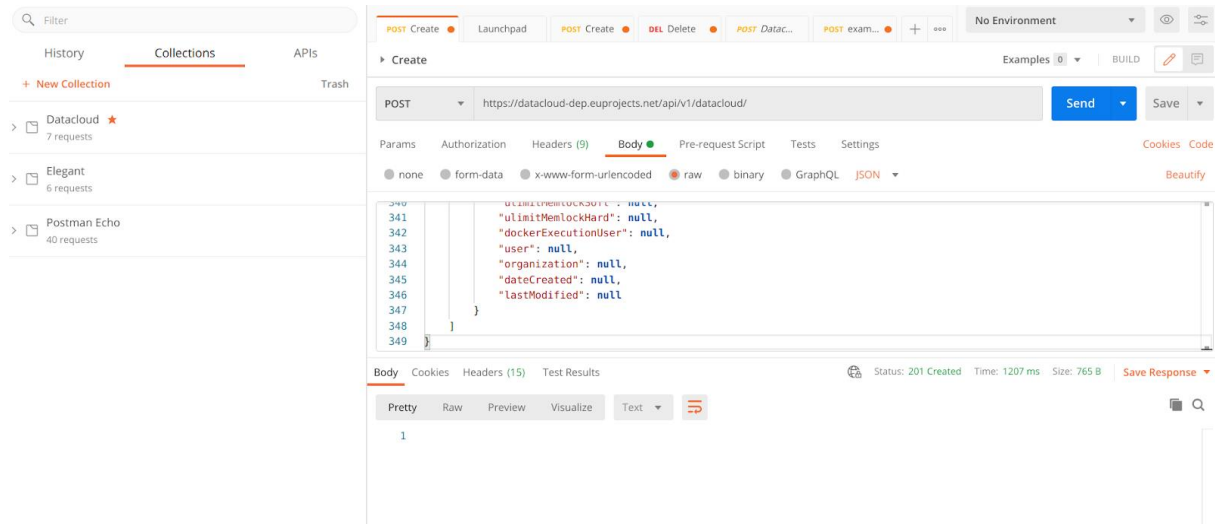


Figure 28: API for deployment of a pipeline chunk

As seen in the figure above, this is done through the call of an appropriate API. The JSON for this call is provided below, and based on the Jsn schema of the DEP-PIPE¹¹ descriptor.

```

{
  "pipelineName": "Heathcare_pipeline",
  "pipelineType": "simulation|production",
  "stepName": "readMedicValues",
  "terminationCheck": {
    "httpURL": "localhost/api/v1/success"
  },
  "time": {
    "EST": 0.0,
    "EFT": 2.92
  },
  "jobListjob": [
    {
      "order": 1,
      "name": "readMedicValues",
      "resource": "datacloud_worker_1_wp1",
      "provider": "R-MARKET",
      "architecture": "amd64",
      "elasticityControllerMode": "HORIZONTAL",
      "dockerImage": "xx/xxx:1.0",
      "dockerCredentialsUsing": "false",
      "dockerUsername": "",
      "dockerPassword": "",
      "dockerCustomRegistry": "true",
      "dockerRegistry": "",
      "requirement": {
        "vCPUs": 4,
        "ram": 4,
        "storage": 20
      },
      "healthCheck": {
        "httpURL": "localhost/api/v1/healthCheck",
        "args": "ps -e | grep java",
        "interval": 20
      },
      "terminationCheck": {
        "httpURL": "localhost/api/v1/success",
        "args": "ps -e | grep java"
      },
      "command": "-->Container Execution<--",
      "environmentalVariables": [
        {
          "key": ""
        }
      ]
    }
  ]
}

```

¹¹ <https://github.com/DataCloud-project/DEP-PIPE-Pipeline-Deployment-Controller/blob/main/samples/sample-file.json>

scaling-controller		^
POST	/api/v1/scaling/up	∨
POST	/api/v1/scaling/down	∨

Figure 30: Scaling API

In addition, following integration discussions with ADA-PIPE, DEP-PIPE also provides an API that provide the information of the Kubernetes Cluster(s).

3.6.3 Deployment, Code and Documentation Availability

DEP-PIPE is currently split in three different components, that are available in GitHub.

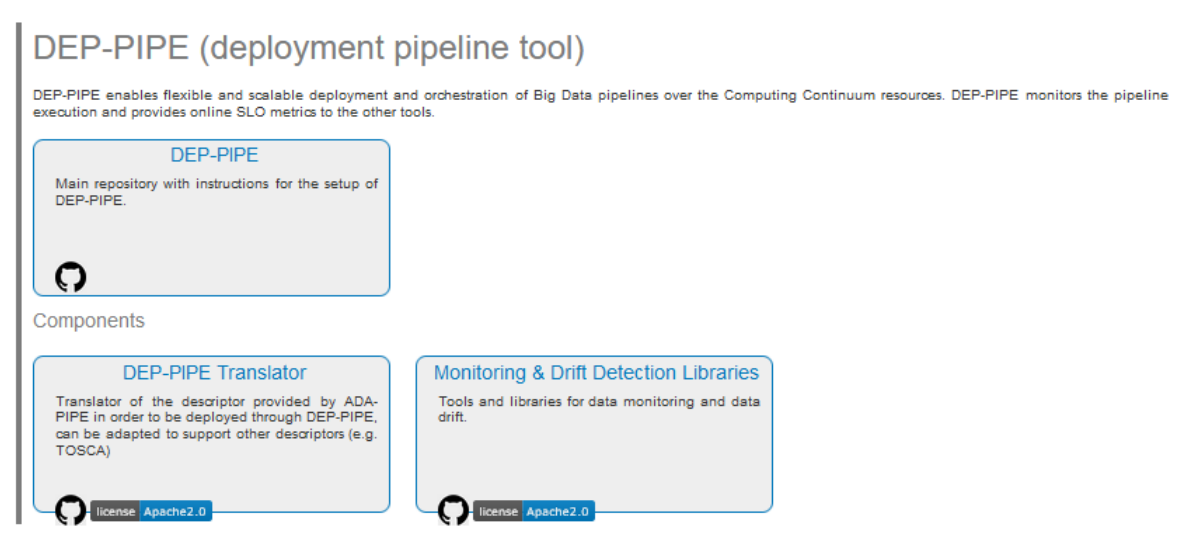


Figure 31: Repositories of DEP-PIPE

The main repository contains instructions for the DEP-PIPE setup (including MAESTRO tool setup instructions), with the DEP-PIPE translator providing the code used for the translation of ADA-PIPE provided JSON to and its deployment through DataCloud. It has to be mentioned that we are currently working in a separate, standalone version of DEP-PIPE that will be used as a standalone service that mediates all actions needed for DataCloud and provide then to the MAESTRO backed. This repo will be provided in the next months (by M26 of the project) as it is crucial for the newly designed flow presented in section 5.

DEP-PIPE has been deployed online and is accessible through the <https://datacloud-dep.europrojects.net> domain and is also part of the toolbox demo page (<https://datacloud-toolbox.europrojects.net/#/deploy>).

4 DATA CLOUD INTEGRATED TOOLBOX DOCUMENTATION

4.1 SETUP INSTRUCTIONS

Regarding the installation of the components, all component are providing instructions for the installation and usage as part of the documentation provided in the corresponding GitHub repositories. In addition, a central point of reference has been created for aggregating the information of all components and is provided in <https://datacloud-project.github.io/toolbox/>

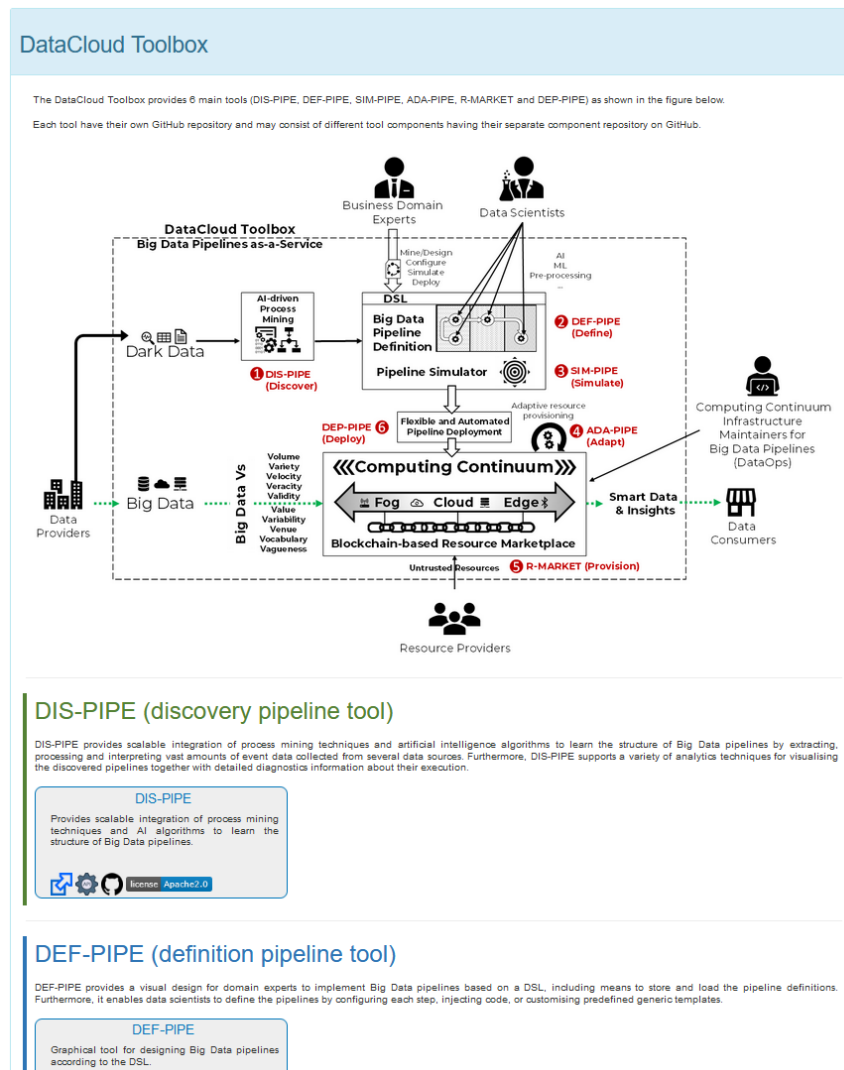


Figure 32: The toolbox page

Finally, for the completeness of the document, we have collected all technical instructions for the deployment of the components in the APPENDIX A – Installation Instructions at the end of this deliverable. For the final release we expect to have even more concise setup process with all components utilizing docker files for their deployment.



4.2 DATA CLOUD TOOLBOX USAGE

The goal of this section is to provide a walkthrough of the main functionalities of the toolbox, with the goal of explaining the flow of as supported in this first toolbox release. For this walkthrough we are going to use the demo deployment of DataCloud, accessible at <https://datacloud-toolbox.euprojects.net>.

4.2.1 Landing in the Toolbox

First, user can use the landing page of the toolbox to view an overview of the available tools, as seen in Figure 33.

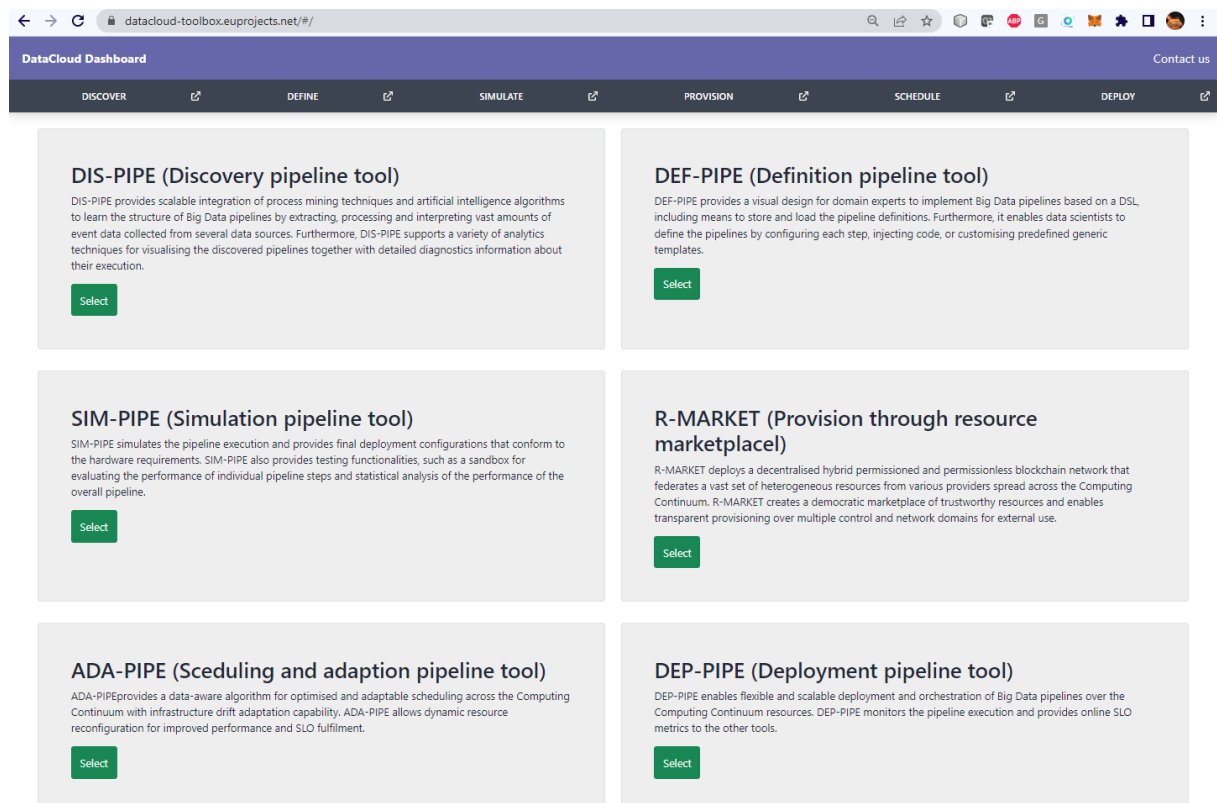


Figure 33: Tools description in the landing page

As for most tools Keycloak has been integrated, user should login with appropriate credentials in order to access tools such as DEF-PIPE and SIM-PIPE.

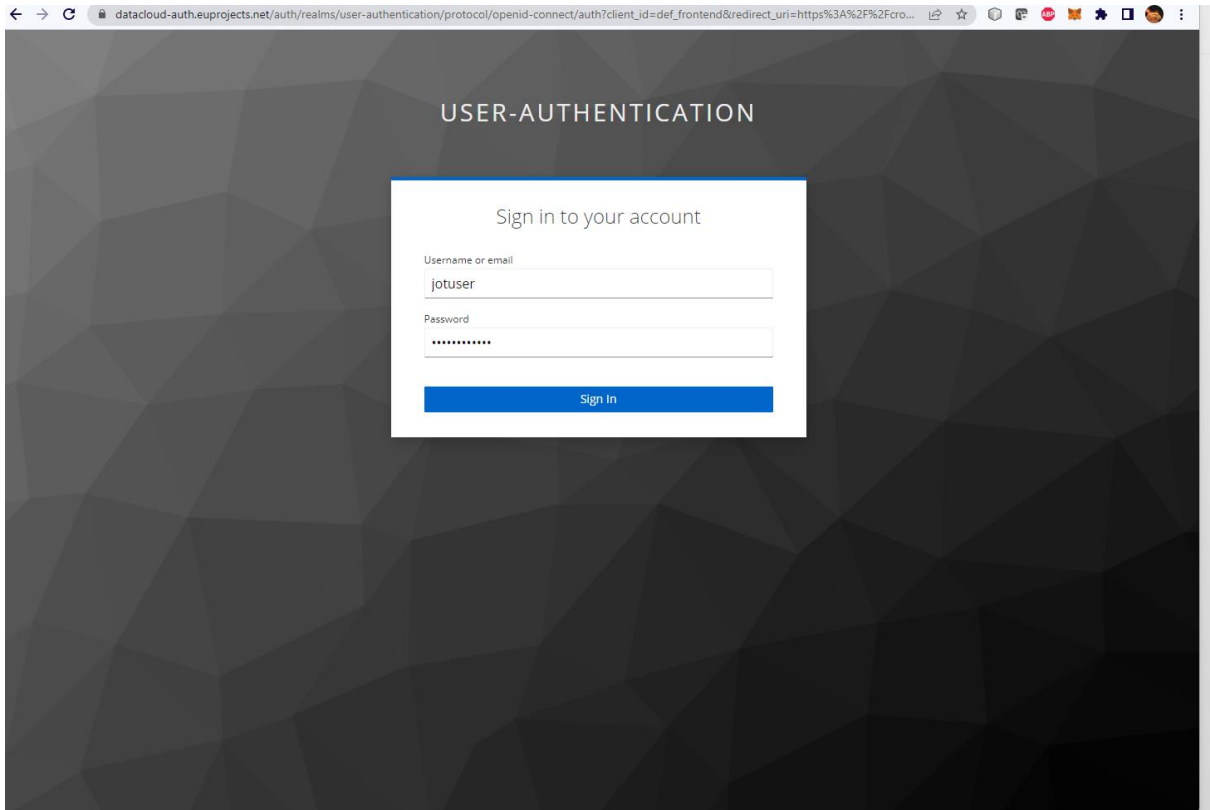


Figure 34: Keycloak based login page

4.2.2 Pipeline Discovery

First step of the DataCloud pipeline lifecycle is the discovery of a pipeline, so below we provide a walkthrough of the various steps required to execute DIS-PIPE and interact with its graphical user interface (GUI).

When a data scientist is interested in analyzing the behaviour of the data pipeline, can import in DIS-PIPE an event log in XES from the I/O Options panel, cf. the bottom-left part of the GUI in, box (vi).



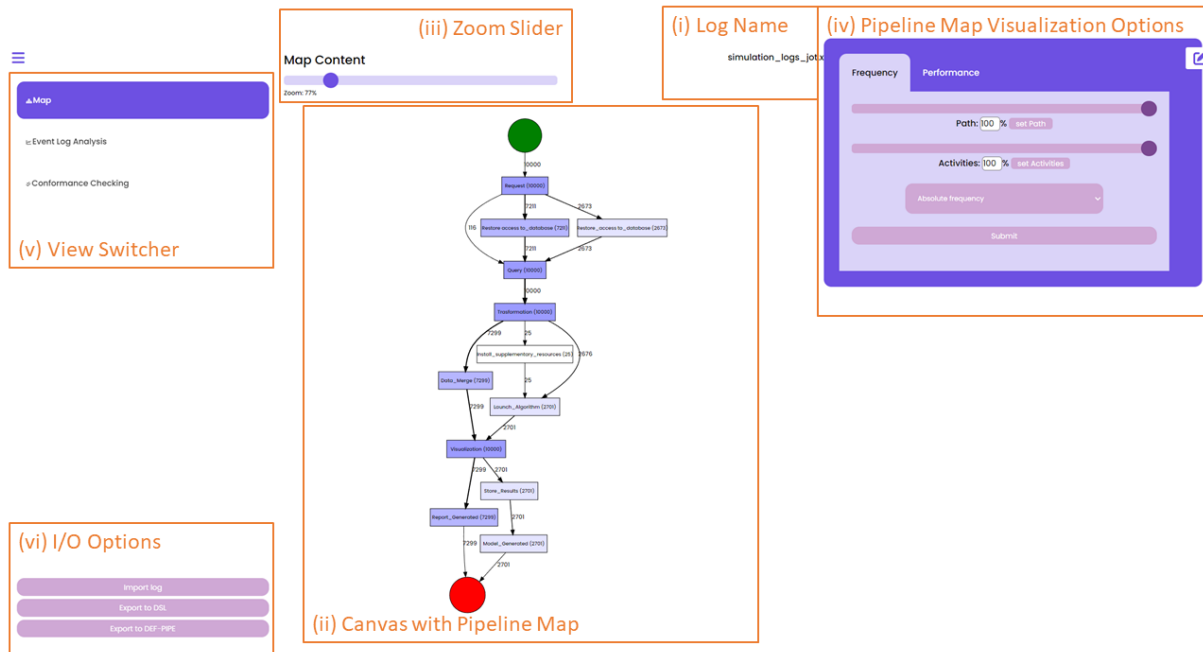


Figure 35: DIS-PIPE main page and its sections

Once an event log is imported, DIS-PIPE automatically runs the implemented discovery algorithm and produces an understandable flowchart view of the discovered pipeline in the form of a DFG, which is visualized in the Map View, cf. the central part of the GUI in Figure 35, box (ii). The name of the log file from which the pipeline model is extracted is shown in the top part of the GUI in box (i).

In the case the data scientist wants to obtain more targeted information on the frequency or performance values related to a node/edge of the DFG, she can directly click on the specific node/edge of interest. A pop-up appears in the GUI showing the requested information. For example, in Figure 36, the data about the frequency values of the step “Data Merge” are shown.

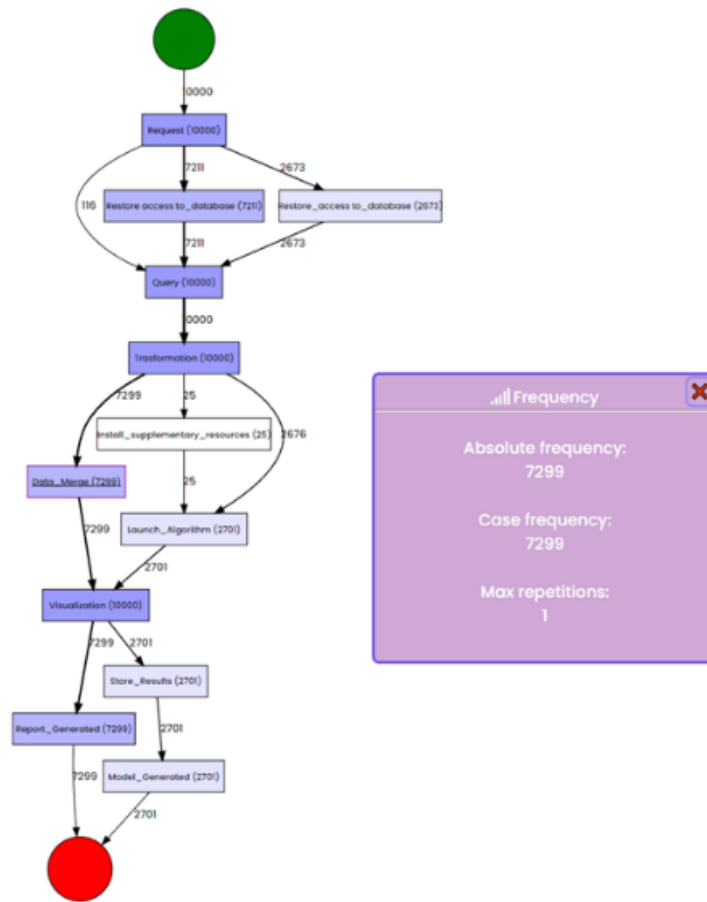


Figure 36: The performance values related to the pipeline step “Data Merge”

Suppose now that the data scientist is interested in gathering further insights about the performance of each step of the pipeline under investigation. To this aim, she clicks on the Performance Tab in the tab switcher called “Pipeline Map Visualization Options”, cf. the top-right part of the GUI in Figure 37.



Figure 37: Pipeline Map Visualization Options

To investigate that the mean duration of each step is aligned with the expectations, user can use the Metric Switcher (cf. Figure 38, box (iii)) to select the option “mean duration”, which

updates the pipeline model with the requested information, as shown in the DFG of Figure 38.

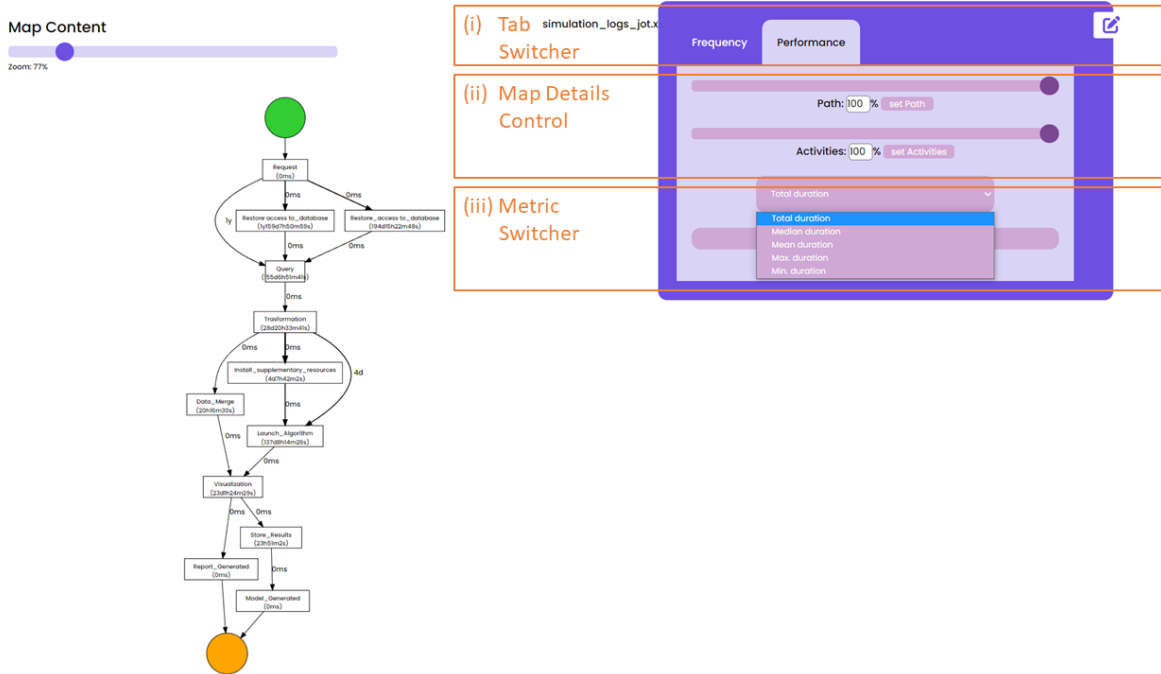


Figure 38: Features of the Performance tab

Let us suppose that the data scientist wants to visualize a version of the DFG that includes only the pipeline execution paths that did not need to restore access to databases. To achieve this objective, the data scientist should first switch from the “Map view” to the “Event Log Analysis View”, as depicted below.

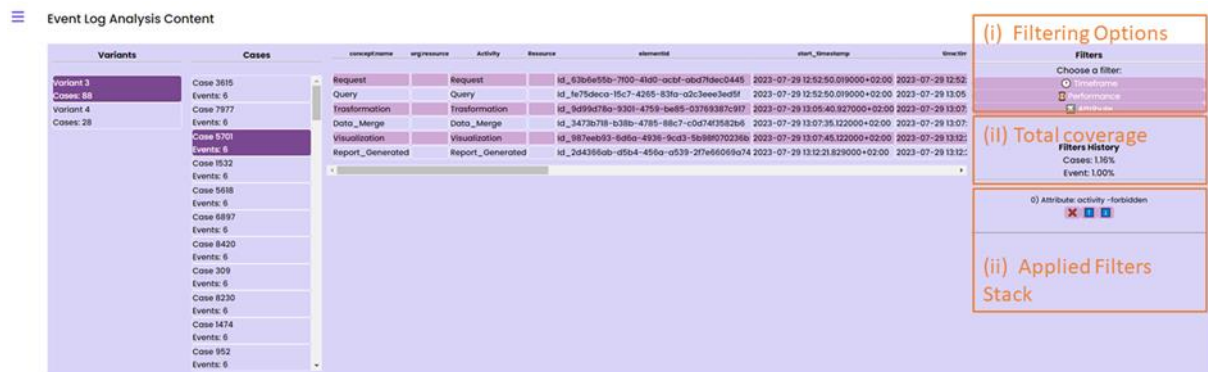


Figure 39: An overview of the Event Log Analysis View

As a result, the anatomy of the event log (i.e., the execution variants and the specific traces) will be visualized on the GUI. Then, to filter out the “restore access to database” steps from the analysis, the data scientist can select the “Attribute” filter from the “Filters” panel, as depicted below.



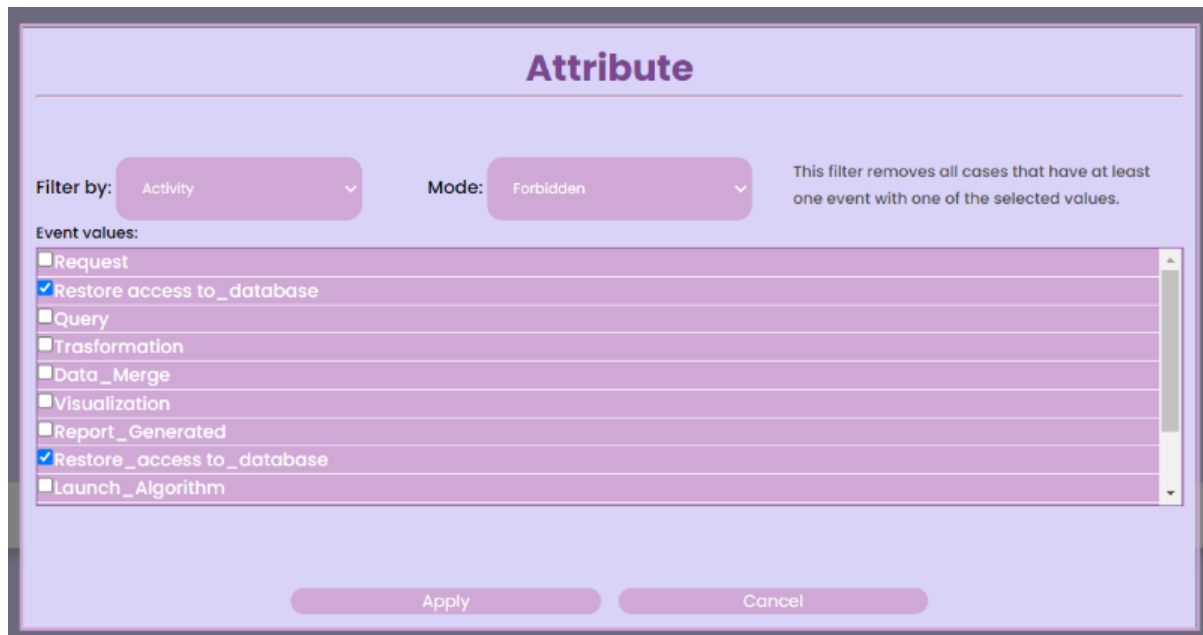
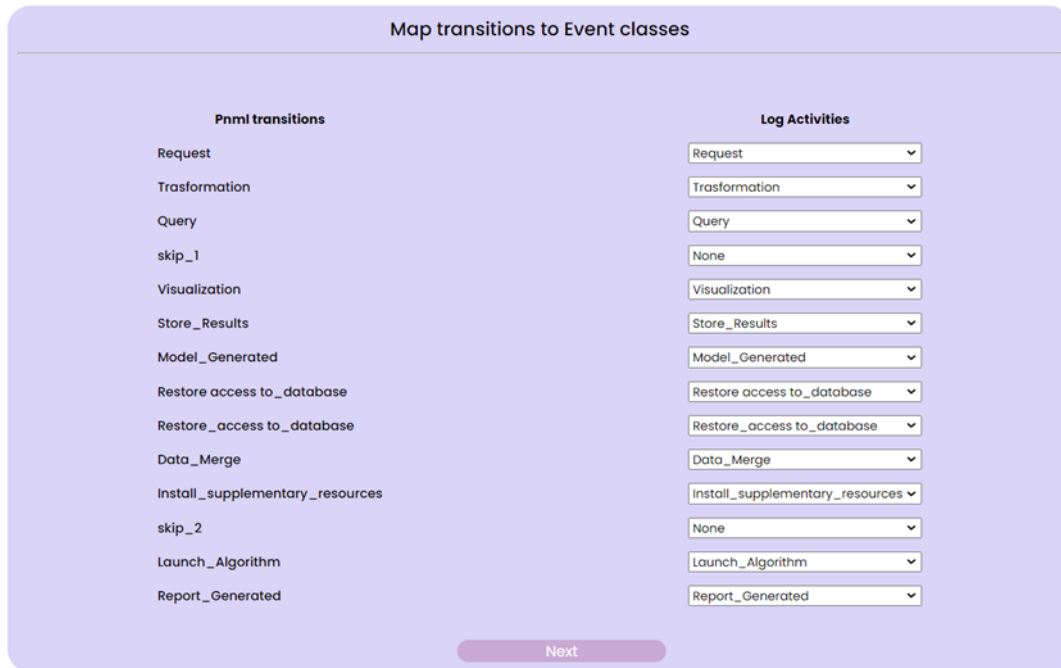


Figure 40: The Attribute filter

A pop-up will be shown on the GUI that enables the data scientist to activate the “forbidden mode” for the “*restore access to database*” steps, cf. Figure 40. By clicking on the “Apply” button, all the execution traces including (at least) one of these steps will be obscured in the event log, thus enabling the discovery algorithm to return an updated pipeline model that neglects the “*restore access to database*” steps.

When the application of the filtering is completed, the data scientist can look at the statistics associated to the filtering activities enacted up to that moment.

Finally, the data scientist may be interested to check the conformance between an event log and a pipeline model discovered from a different event log or modeled through DEF-PIPE. She starts by clicking on “*Conformance Checking*” in the View Switcher. This will ask her to choose a log file and a pipeline, which could either be the result of the process discovery that she previously did or a different one of which she has the DSL. After having chosen the input, user faces a menu in which is asked to map the steps found in the log, on the left-hand side, with the steps of the chosen model, on the right-hand side, as can be seen in Figure 41.

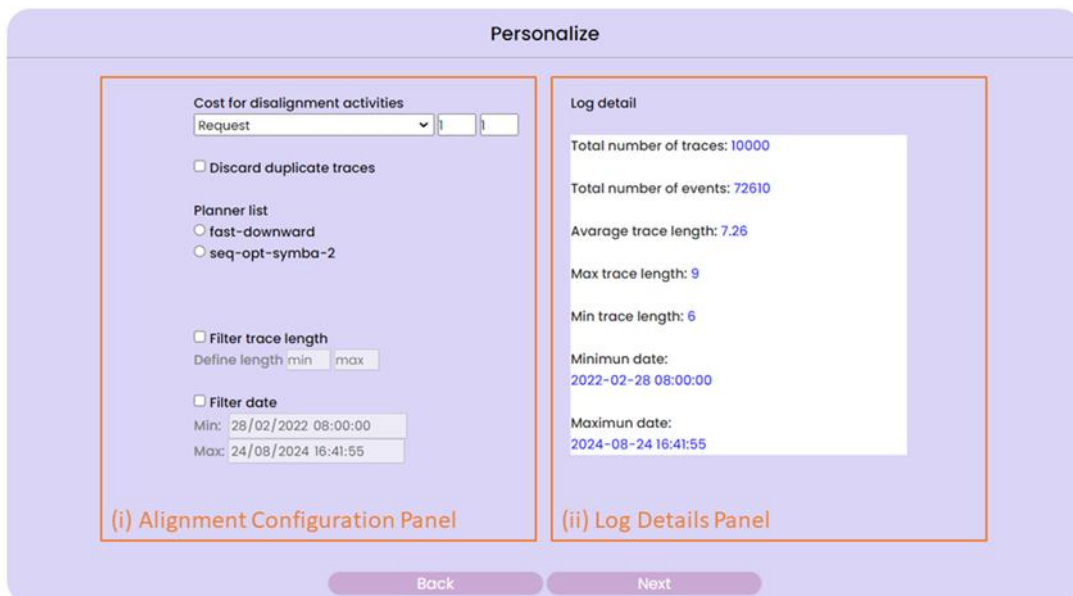


Pnml transitions	Log Activities
Request	Request
Trasformation	Trasformation
Query	Query
skip_1	None
Visualization	Visualization
Store_Results	Store_Results
Model_Generated	Model_Generated
Restore access to _database	Restore access to _database
Restore_access to _database	Restore_access to _database
Data_Merge	Data_Merge
Install_supplementary_resources	Install_supplementary_resources
skip_2	None
Launch_Algorithm	Launch_Algorithm
Report_Generated	Report_Generated

Next

Figure 41: Mapping log events with pipeline steps

When mapping is finished, user can click the “Next” button to go to the next menu, in which customization of the alignment in the “Alignment Configuration Panel” and look at details about the chosen log in the “Log Details Panel” can be performed, as can be seen in Figure 42.



Cost for disalignment activities

Request

Discard duplicate traces

Planner list

fast-downward

seq-opt-symba-2

Filter trace length

Define length

Filter date

Min: 28/02/2022 08:00:00

Max: 24/08/2024 16:41:55

(i) Alignment Configuration Panel

Log detail

Total number of traces: 10000

Total number of events: 72610

Avarage trace length: 7.26

Max trace length: 9

Min trace length: 6

Minimun date: 2022-02-28 08:00:00

Maximun date: 2024-08-24 16:41:55

(ii) Log Details Panel

Back Next

Figure 42: Alignment settings

When the configuration is completed, user can proceed by clicking the “Next” button and will be presented with the pipeline map in the “Map Panel”, with each step highlighted either in green, if there were no misalignments during the conformance checking, or in red, if some

misalignments were found. In the “Trace Panel” user can look at details about the alignment for each individual trace of the chosen log and can highlight in yellow the selected trace by setting “Highlight Trace” to “Yes”. By clicking on the box of a step in the pipeline map, user can see how many times it produced a misalignment, and the list of the traces in which that happened. An example of the interface showing the result of conformance checking can be seen in Figure 43.

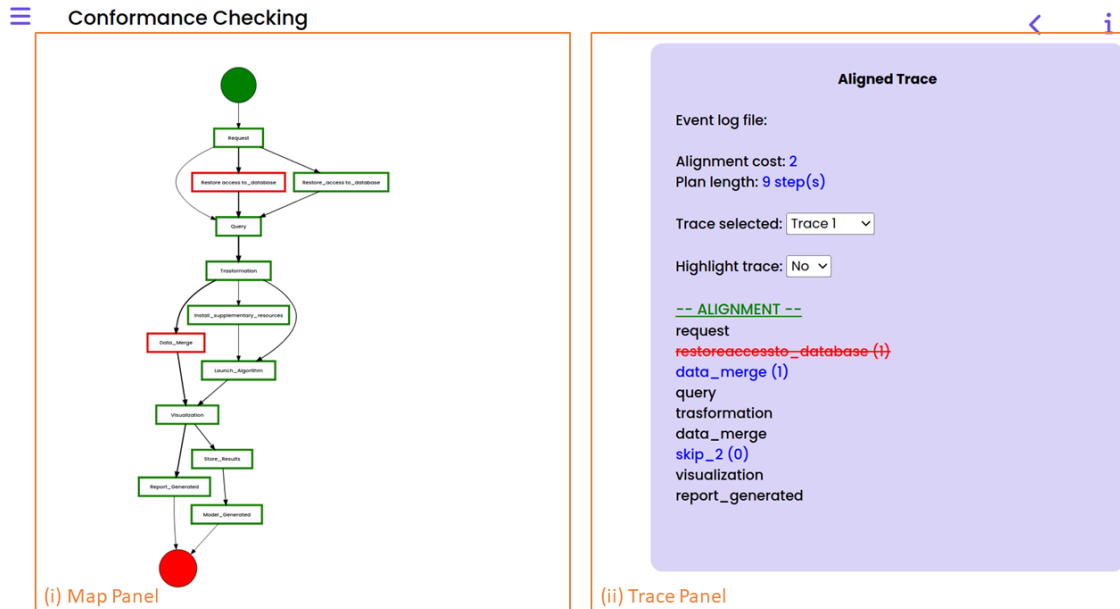


Figure 43: Results of the trace alignment activity for the JOT pipeline

4.2.3 Pipeline Design

When accessing DEF-PIPE, in the upper-right corner “login” button allows make login with the DataCloud Keycloak credentials, if user has not been signed-in already.

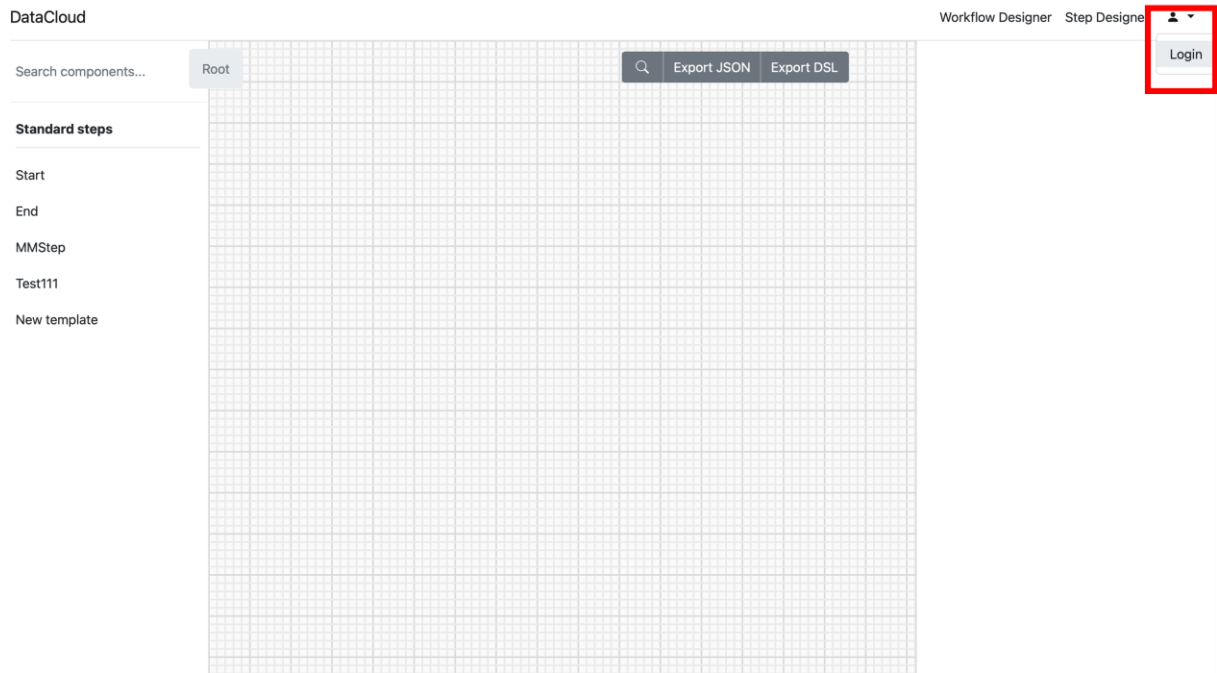


Figure 44: Login in DEF-PIPE

Left panel contains lists of steps and pipelines. Open lock icon near the steps and pipelines names means that the corresponding description is public (closed lock means private).

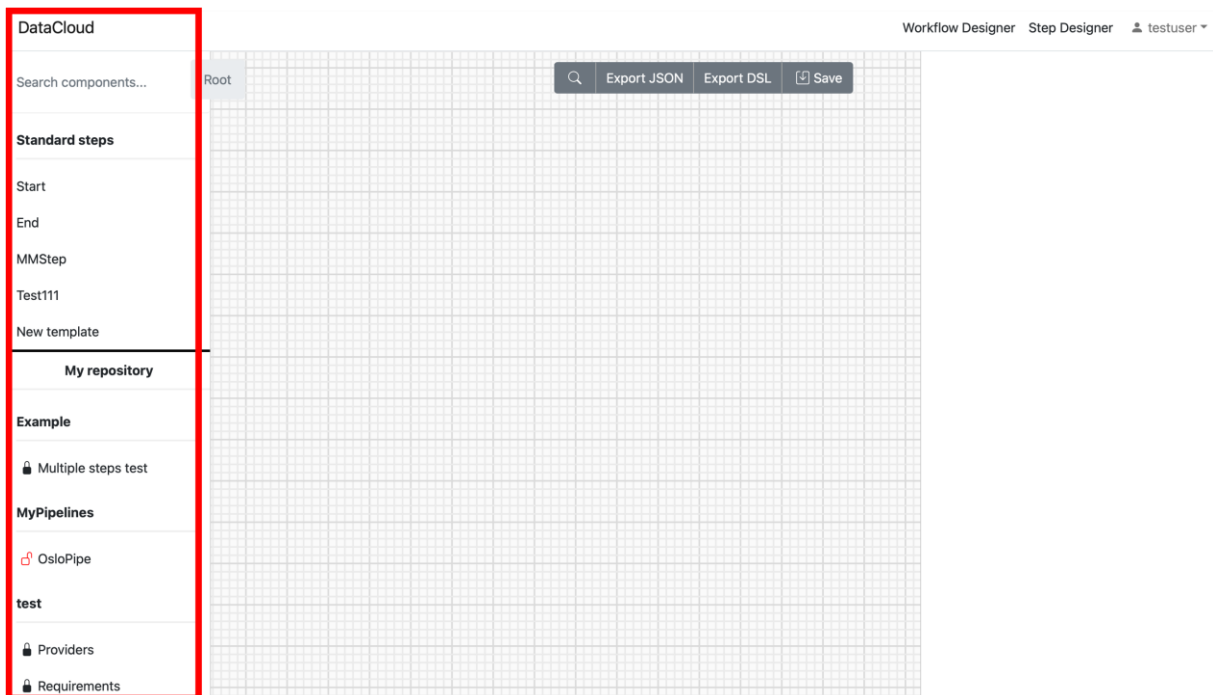


Figure 45: Left panel of DEF-PIPE



On the right of the upper menu line there two are options: “Workflow Designer” and “Step Designer”. They correspond to two regimes of work: creation/editing workflow /pipeline and creation /editing Steps.

When selecting “Step Designer” option new steps can be created after selection of “Add step” in the left panel. Otherwise clicking to existent Steps open it in the main (grid) part of the window.

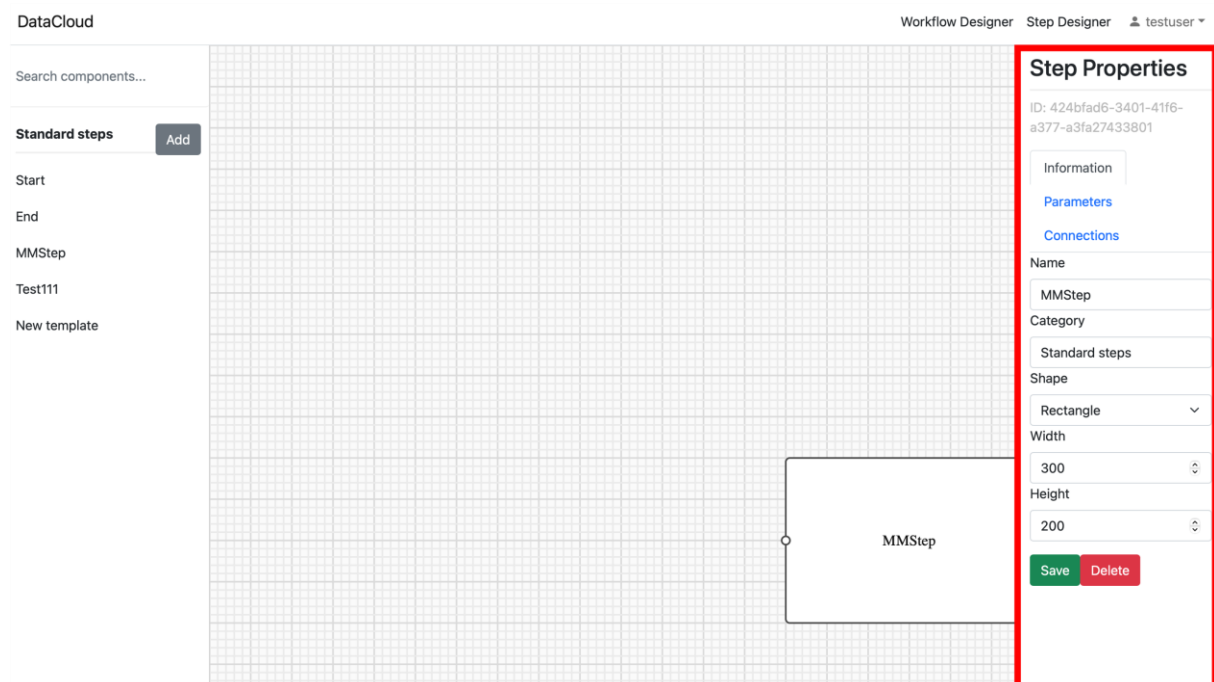


Figure 46: Step Designer in DEF-PIPE

When working with Steps right panel provides options: Information, Parameters and Connections.

- (a). **“Information”** allows add steps name and shape
- (b). **“Connection”** allows add inputs and outputs of the Step. Coordinates and types of the connections are entered in the corresponding boxes.
- (c). **“Parameters”** allow add Step parameters according to DSL description.
- (d). **Save and Delete** buttons to save or delete the step

When working with Workflow/pipelines (“Workflow Designer” menu) available steps are listed in the left panel. Clicking on corresponding step draws it in the main part of the window.

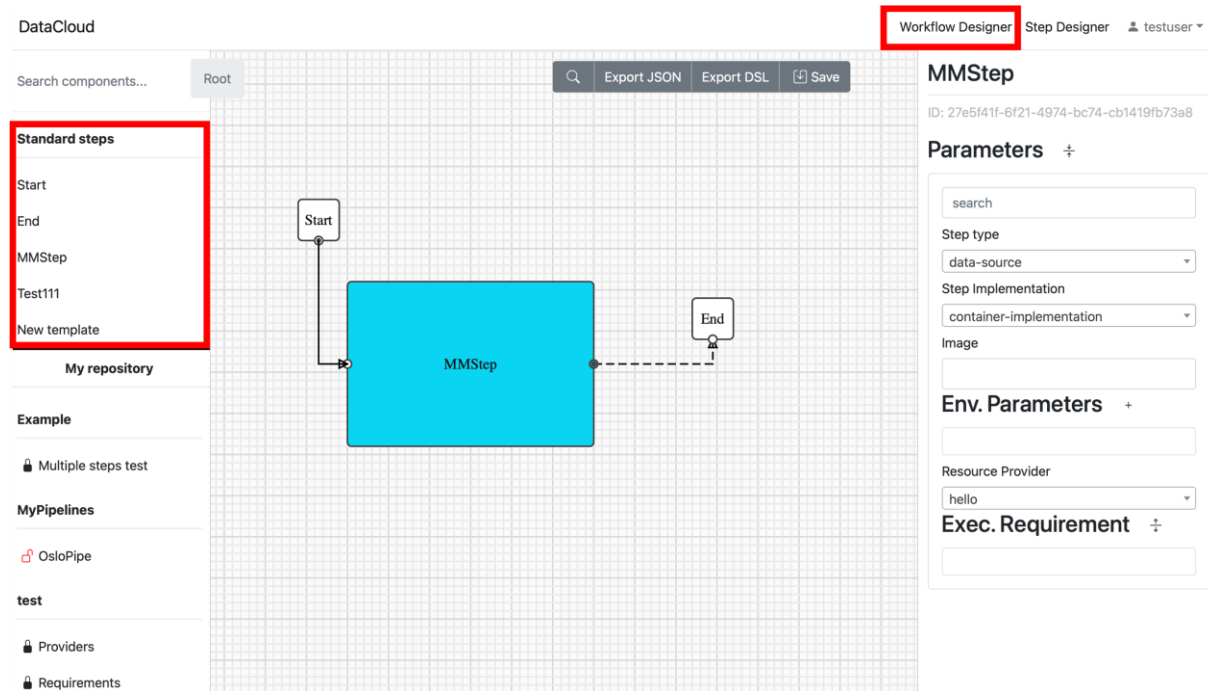


Figure 47: Workflow designer in DEF-PIPE

Connections between steps are drawn after clicking to corresponding connection points, as depicted in Figure 47. It has to be noted that all Workflows/pipelines must have Start and End points (taken from the list of steps). If they are not present, then generation of DSL is not possible.

Parameters of the pipelines can be described (or visualized) in the right panel.

“Save” button in the upper menu line allows to save the created pipeline. “Export DSL” button in the upper line menu allows to generated corresponding DSL text from the current graphical pipeline description. “Export JSON” button is currently used for internal purposes and users don’t need to use it.

4.2.4 Pipeline Simulation

When accessing SIM-PIPE, in the upper-right corner “login” button allows make login with the DataCloud Keycloak credentials, if user has not been singed-in already. Once logged in, user can view the simulations already available as depicted in Figure 48.



Figure 48: Landing page of SIM-PIPE

A new simulation can be created using the appropriate button and form, as seen in Figure 49. Currently, the user is instructed to upload a file that describes the pipeline and has been created using DEF-PIPE. In the future the two tools will integrate in order to allow easier onboarding of defined pipelines for simulation.

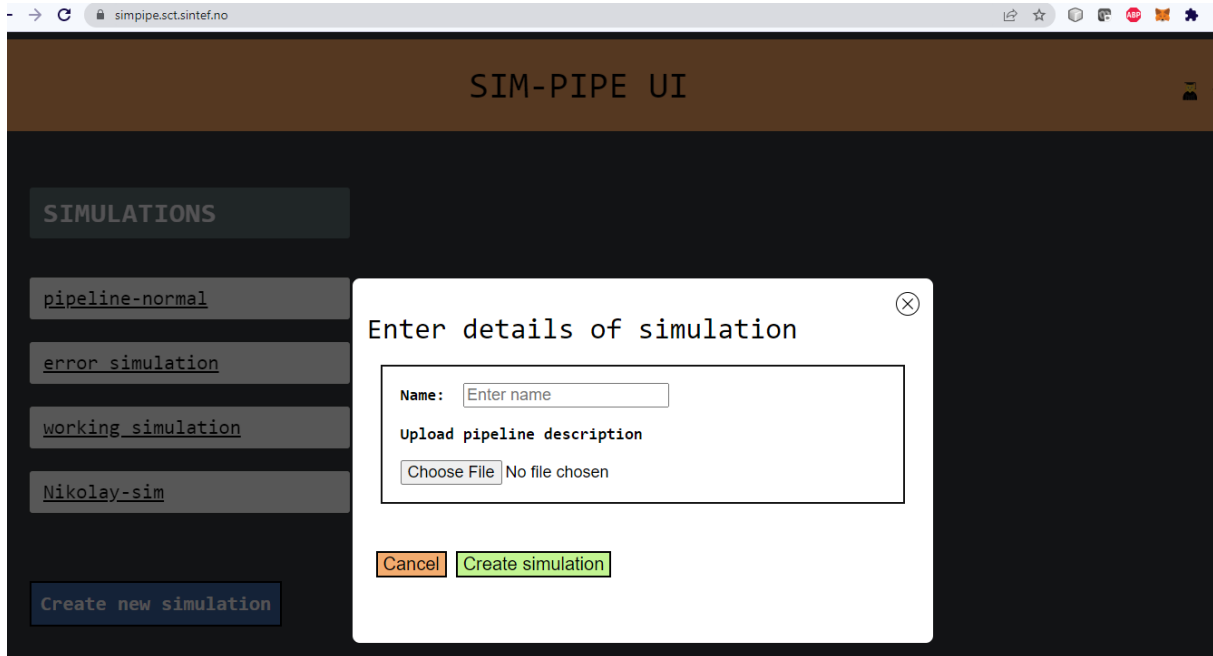


Figure 49: Creating Simulation using a pipeline descriptor

An overview of simulation is provided in Figure 50.



Figure 50: Simulation overview

Then the user is able to configure how the simulation run will be executed, by configuring the appropriate options (Figure 51).

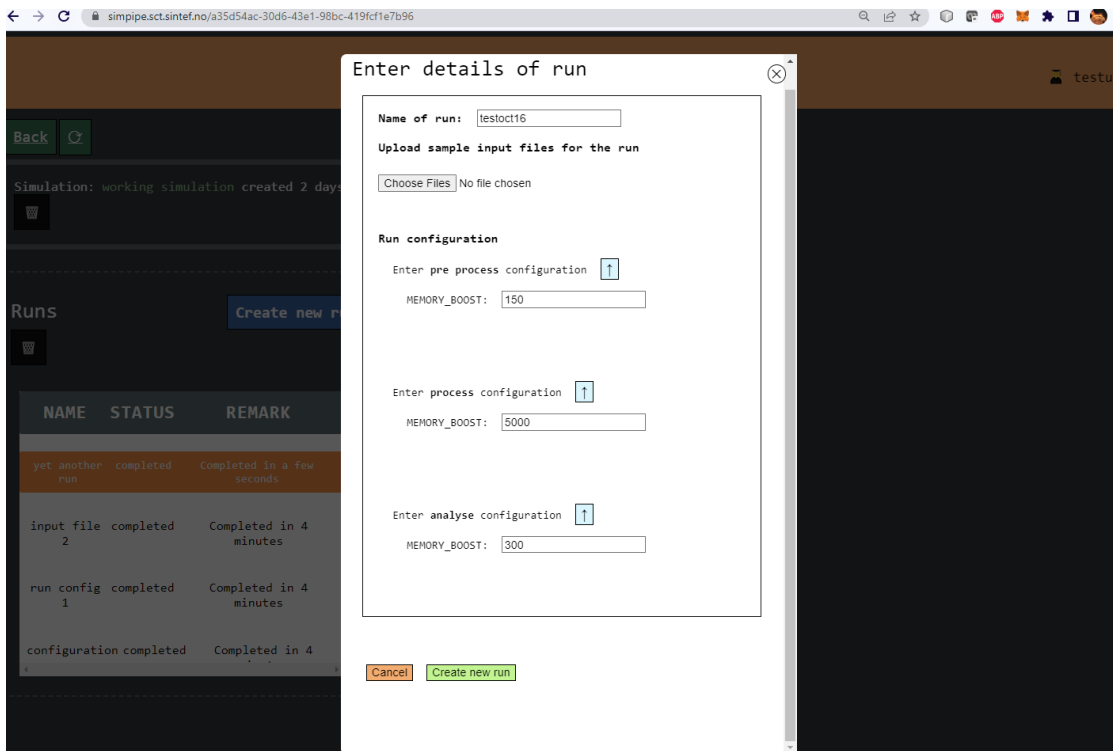


Figure 51: Configuring a simulation run



Finally, the user is presented with the results of the simulation that include various visualization graphs, as seen in Figure 52.

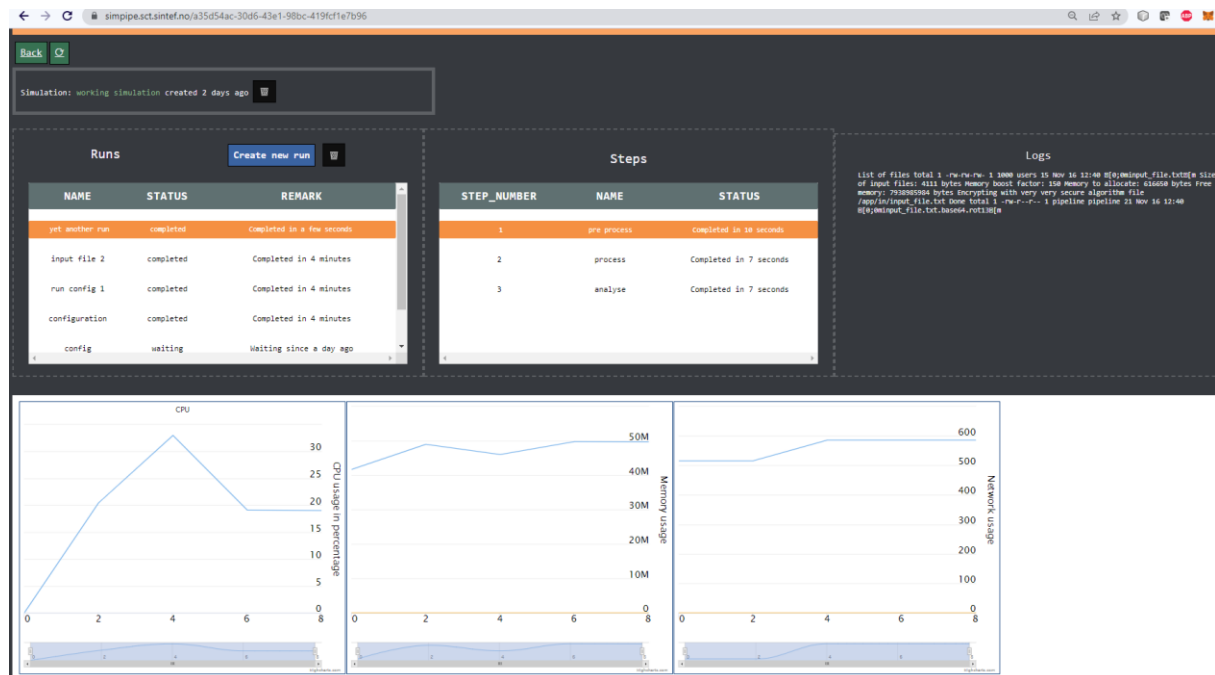


Figure 52: Simulation results

4.2.5 Pipeline Execution & Runtime Management

Regarding the execution and management of the pipeline at runtime, once the user logs in to the runtime dashboard that is provided by DEP-PIPE. This tool is actually the main point of interactions for both ADA-PIPE and R-MARKET.

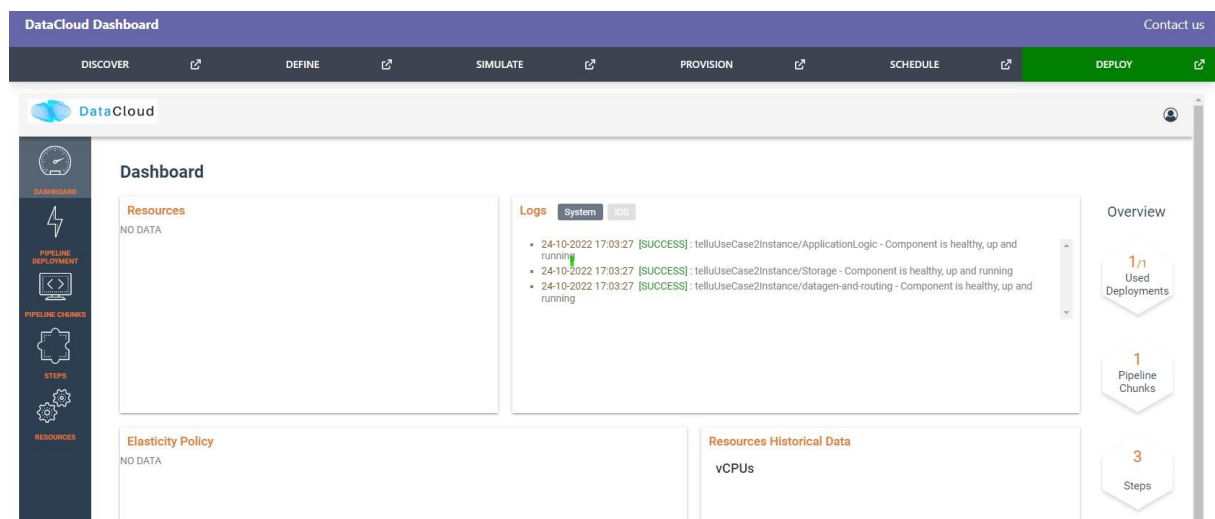


Figure 53: Landing page of the DEP-PIPE and the Runtime Management Dashboard

For the execution of the pipeline, an existing Kubernetes cluster must be provided. For the resources used in this cluster, it is also possible to use worker from R-MARKET and the DataCloud worker pool. For each worker a contract must be made.

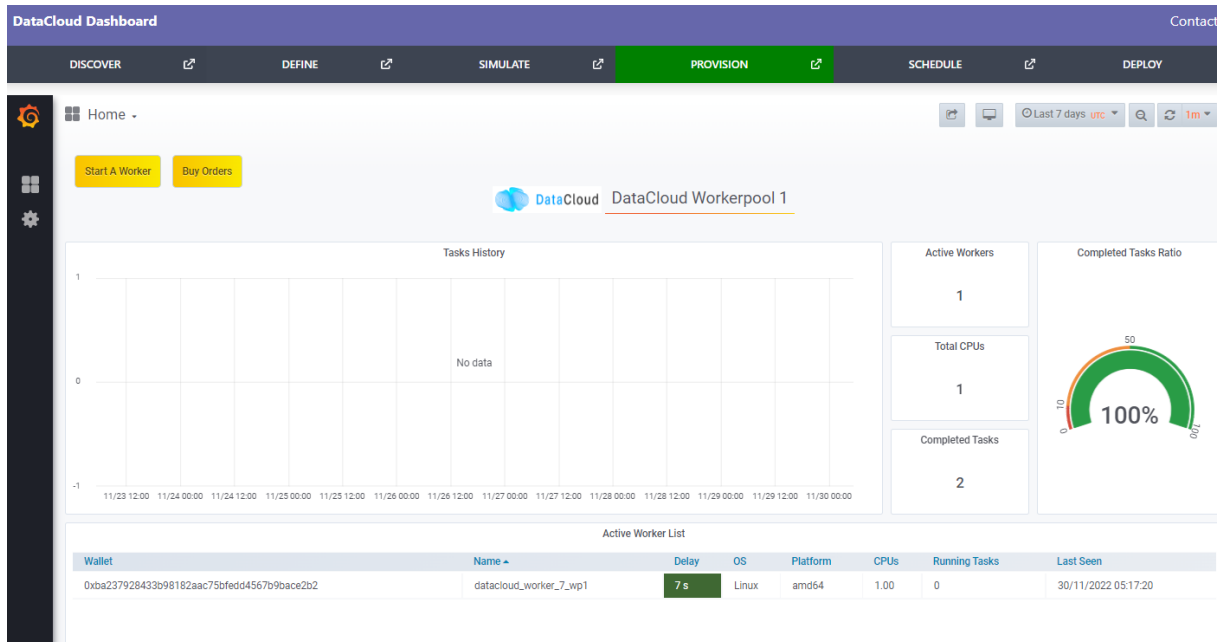


Figure 54: R-MARKET resource available at the DataCloud Workerpool

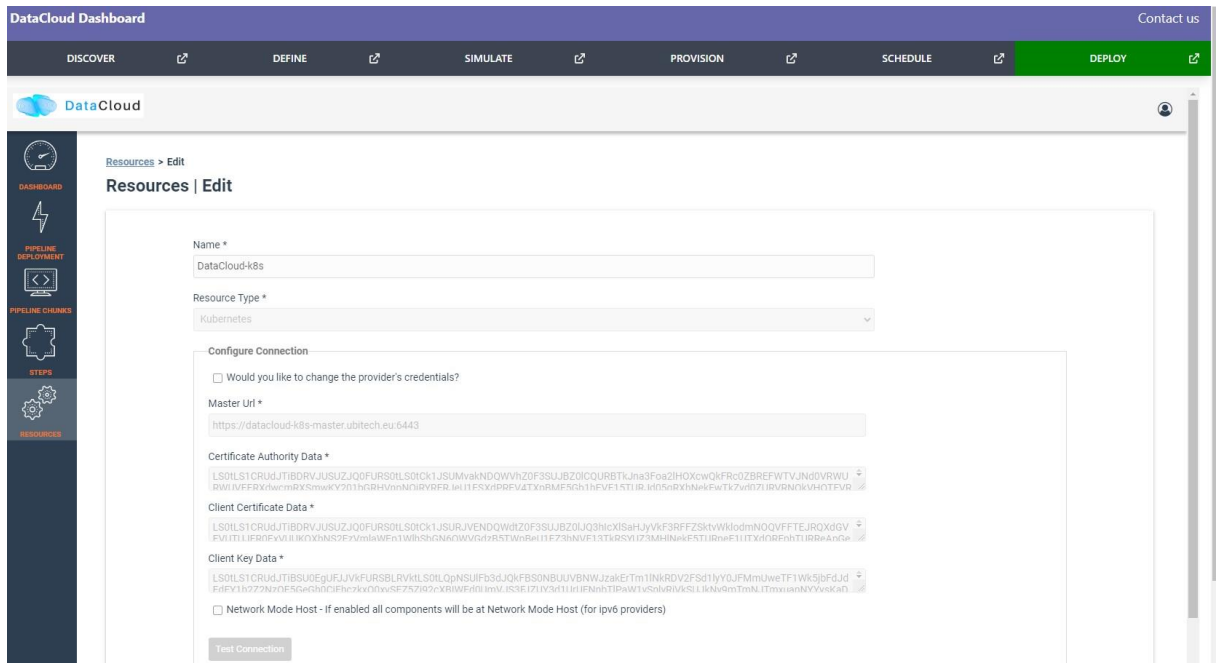


Figure 55: Onboarding a Kubernetes Cluster for pipeline deployment

The defined policies from the DEF-PIPE tool are fetched and displayed.

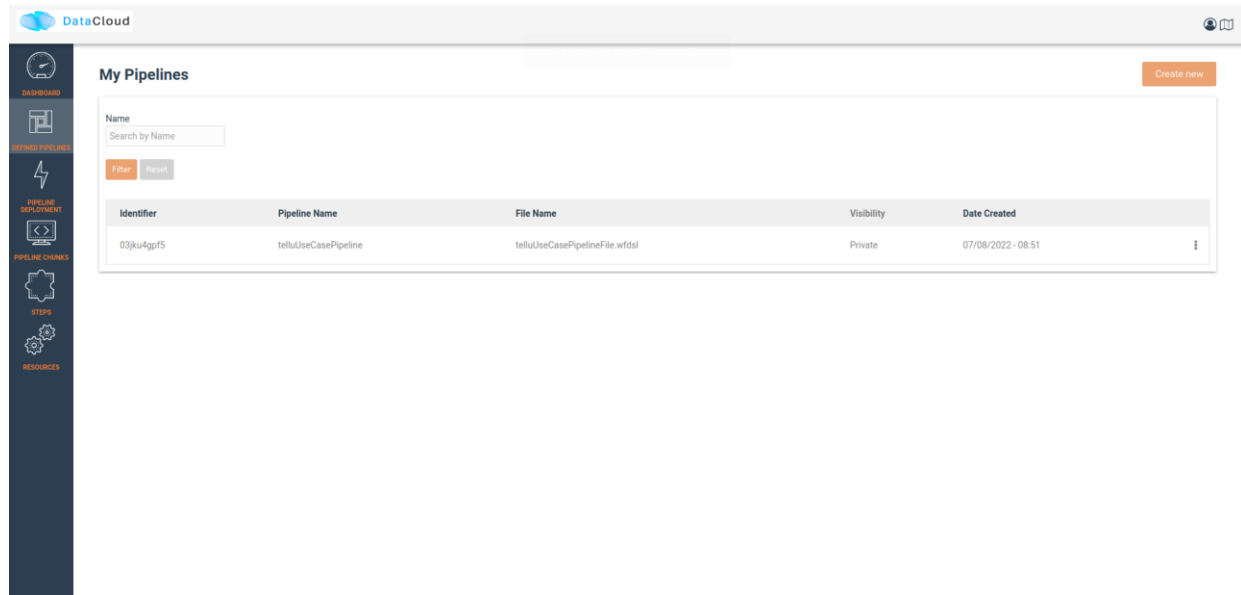


Figure 56: User pipelines presented in DEP-PIPE

For performing a deployment, DEP-PIPE is provided by ADA-PIPE with the pipeline chunk to deploy in the specific resources selected by it. Currently, this is done by API calls or the UI of ADA-PIPE presented in Figure 57, but the full automation of this step is in progress.

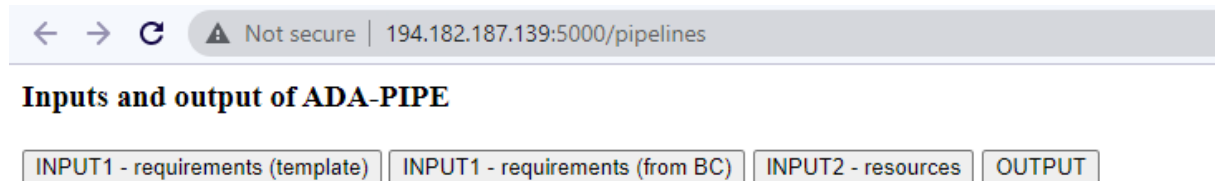


Figure 57: ADA-PIPE scheduling inputs

Once the pipeline has been analysed and chunks have been created, we can view them in the dashboard as seen in Figure 58 below.

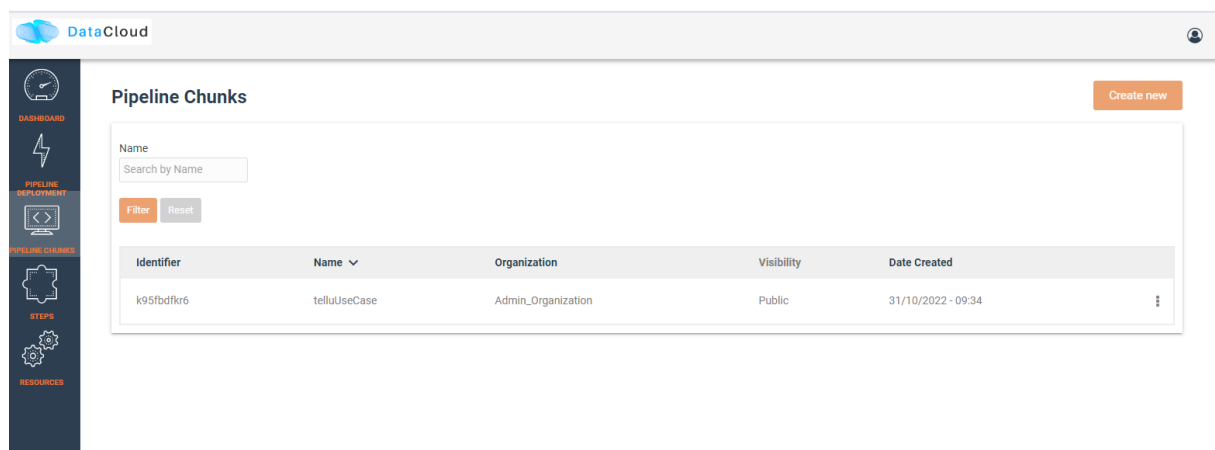


Figure 58: Pipeline Chunks provided by ADA-PIPE for deployment

Users can view the pipeline steps of the pipeline and if they need to configure them prior to deployment (integration is ongoing).

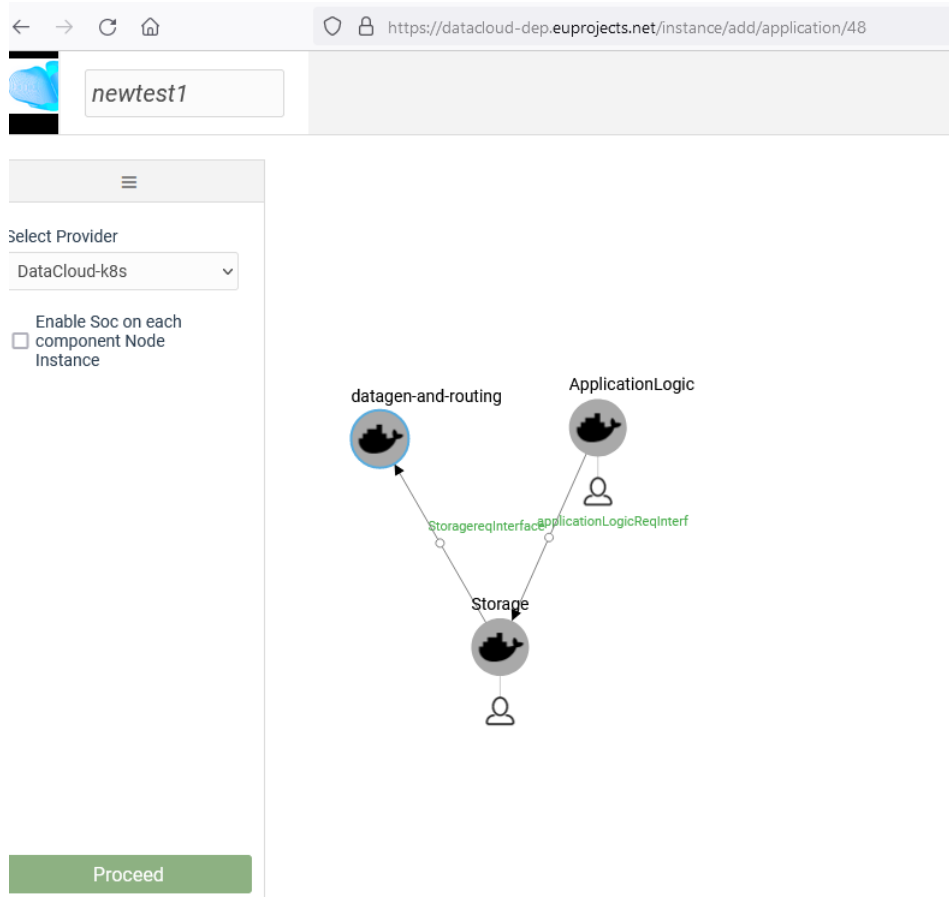


Figure 59: Viewing a pipeline chunk before deployment

Figure 60: Enabling security



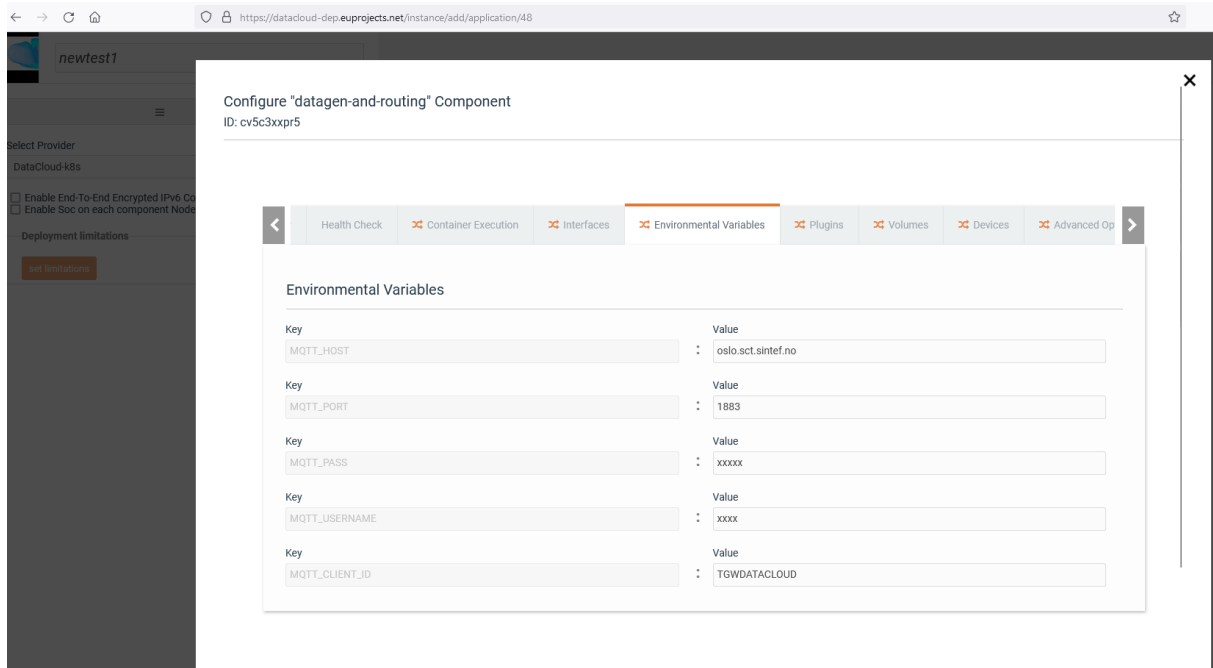


Figure 61: Pipeline Configuration

A successful deployment is then presented in the user dashboard.

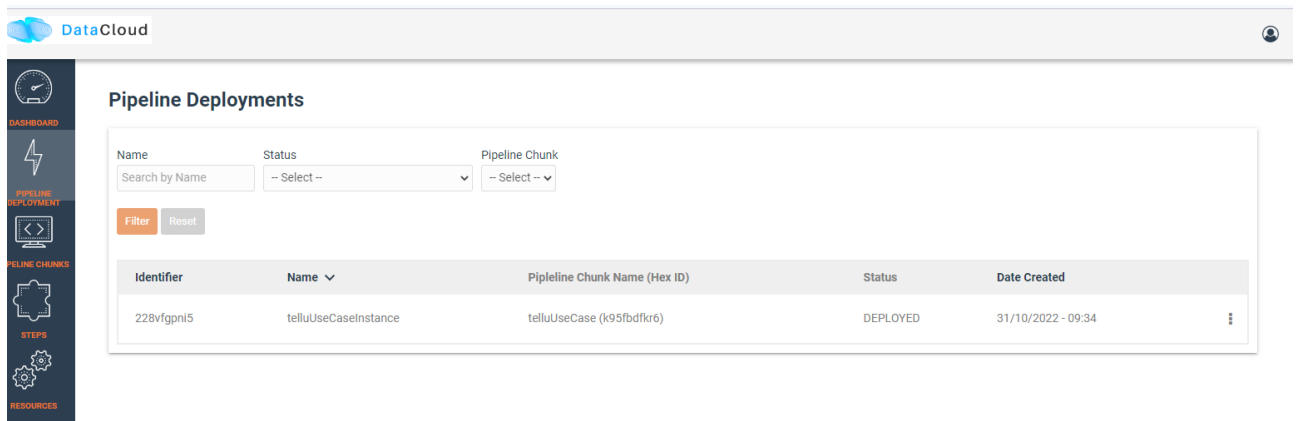


Figure 62: Pipeline Deployment

Menu options provided for viewing the graph, editing policies and un-deploy the pipeline.

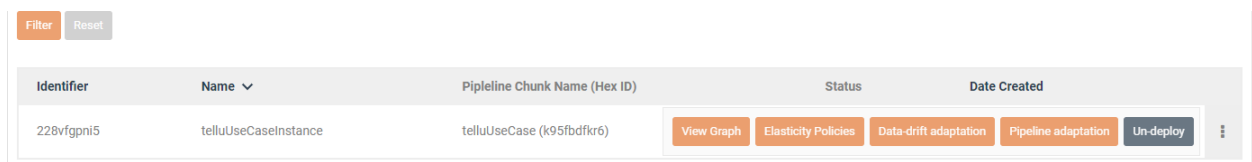


Figure 63: Pipeline Deployment Menu

Policy editor can be used for the creation of adaptation, scaling, and data drift policies.

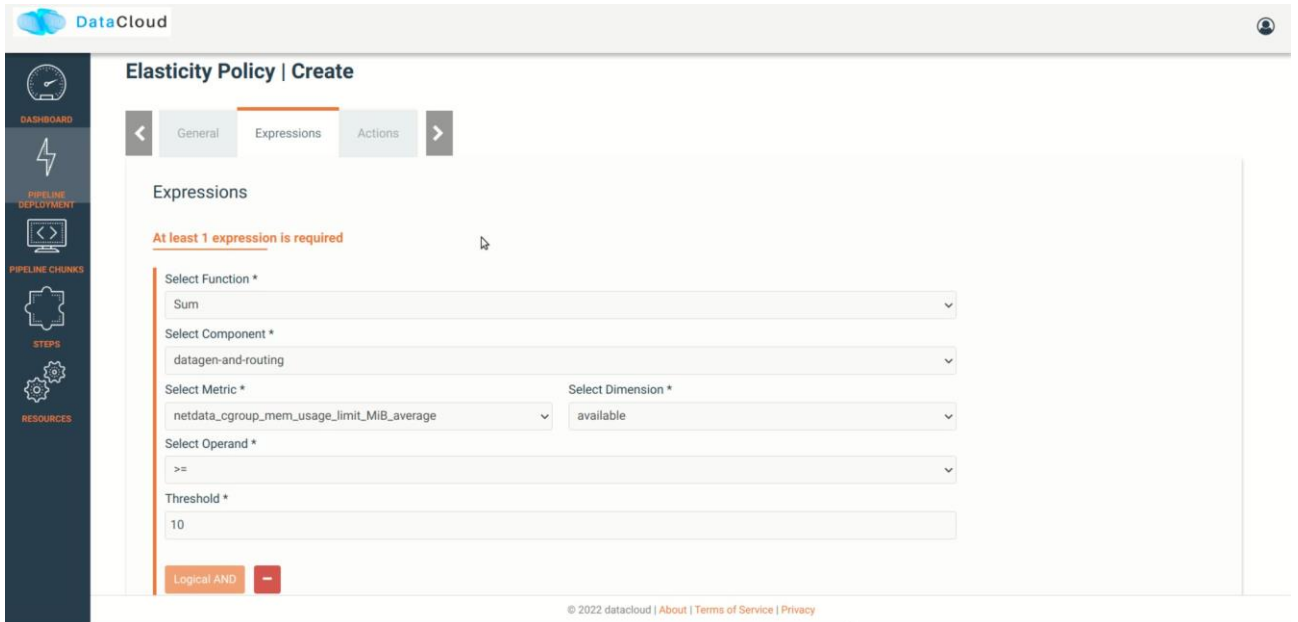


Figure 64: Policy Editor for scaling

The pipeline graph and monitoring data are provided to the user.

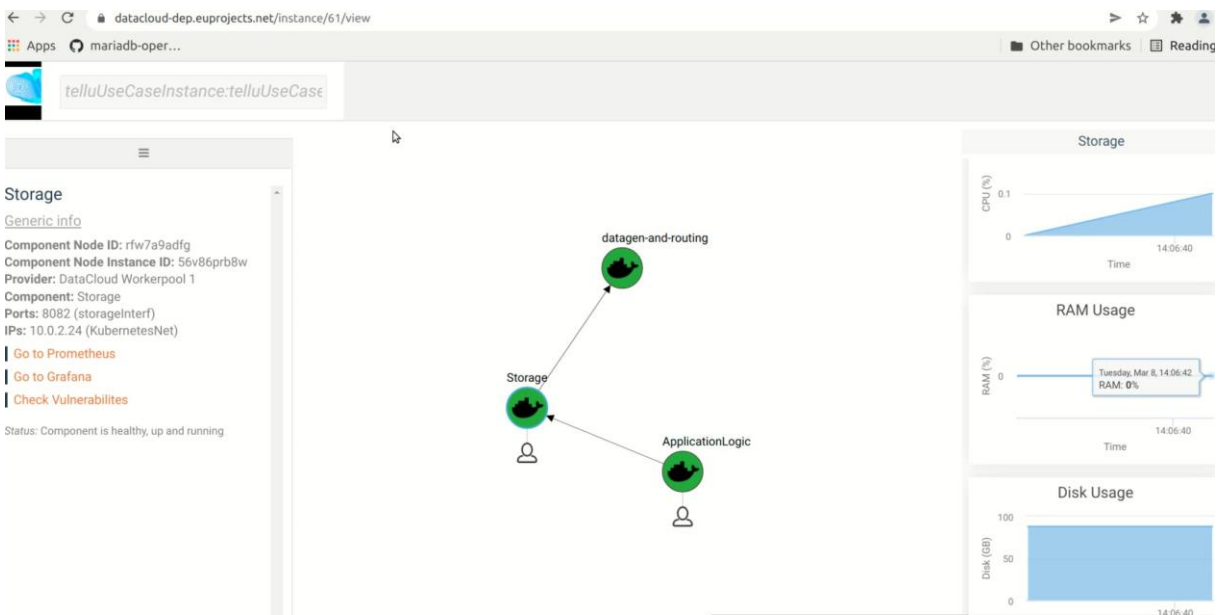


Figure 65: Pipeline Graph and Monitoring Data



5 PLANS FOR FUTURE RELEASES

Regarding the preparation for the final release of DataCloud toolbox two main aspects are analysed; the tool improvements and the integrated flow improvements. First is the improvements of the various tools; as explained already in section 3 each tool is planning updates and improvements that have to do with core functionalities that are promised, but also with the interaction with the other DataCloud components. For this interaction to be achieved, in most cases the proper usage of the DSL descriptor is a pending task, as the DSL descriptor created by DEF-PIPE is a central part of the flow.

In addition, as explained in section 2.1 we have started working towards a more tightly integrated platform for the DataCloud toolbox, with well-defined user interactions and therefore updated flows to be supported. One of the reasons for this plan is the valuable feedback from the use case evaluation that is currently performed; based on our interviews, we got mostly positive feedback for the runtime dashboard usage and we identified the need to integrate with the DEF-PIPE for providing the defined pipelines as a concept that is also used in the runtime. Also, it was made clear that pre-deployment or runtime configurations (e.g., configuration of credentials etc) should not be part of the pipeline design but instead a pre-deployment phase is needed.

For the redesign of the toolbox flow, we have developed the sequence diagram that is depicted in Figure 66, and also wireframes using collaborative design tool Figma¹². The sequence diagram has been created using PlantUML modelling tool. A high resolution SVG of the diagram is available online¹³.

¹² <https://www.figma.com>

¹³ <http://tinyurl.com/296hb5k2>



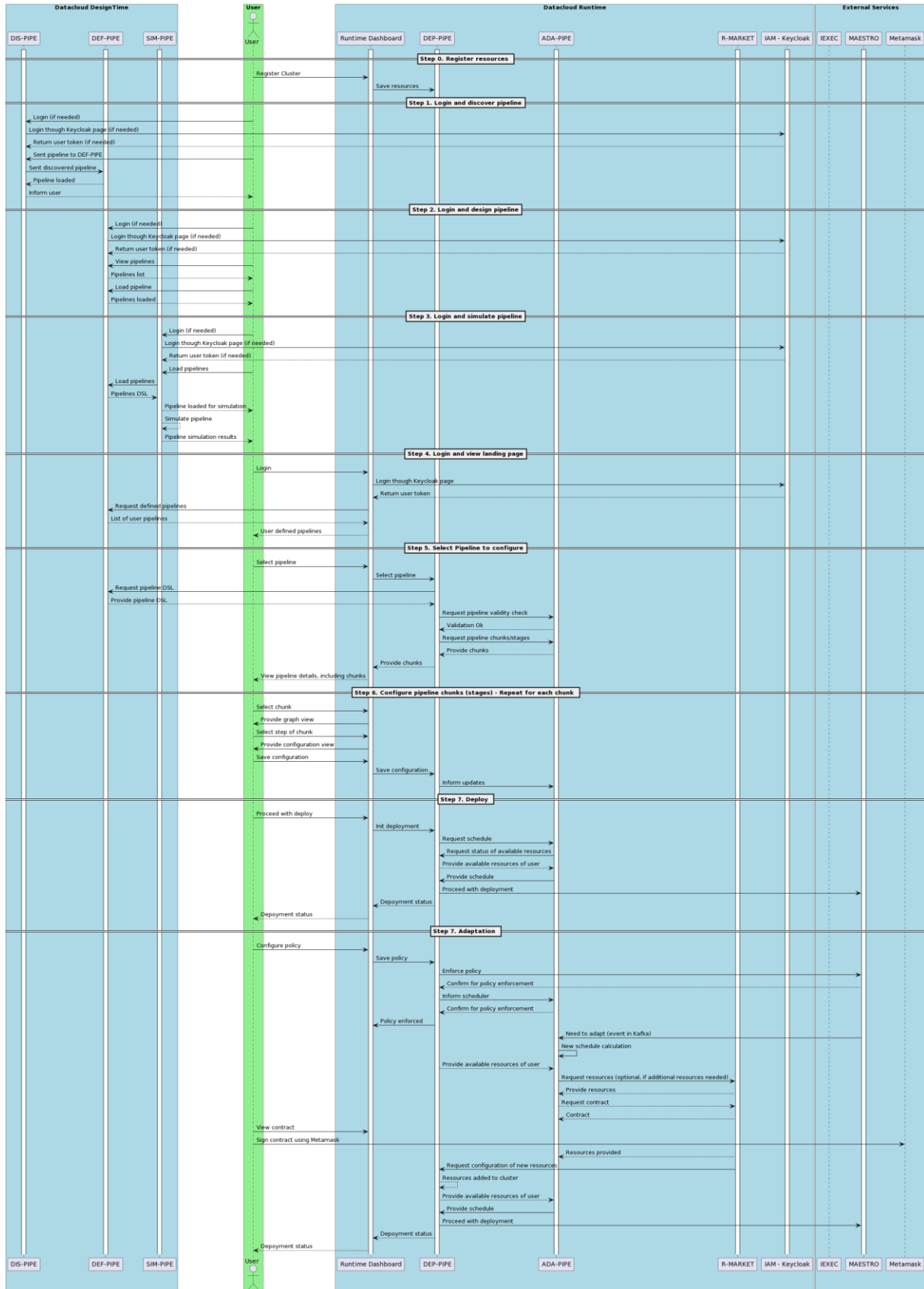


Figure 66: Sequence Diagram presenting the updated DataCloud flows



In addition to the sequence diagram the following wireframes have been created for integrating design time and runtime tools. The redesign process has been implemented using Figma (live) wireframes. The Figma project includes links to simulate the look and feel of the new wireframes to be created for the second release of the toolbox dashboard¹⁴, and also the updated wireframes for the SIM-PIPE tool¹⁵.

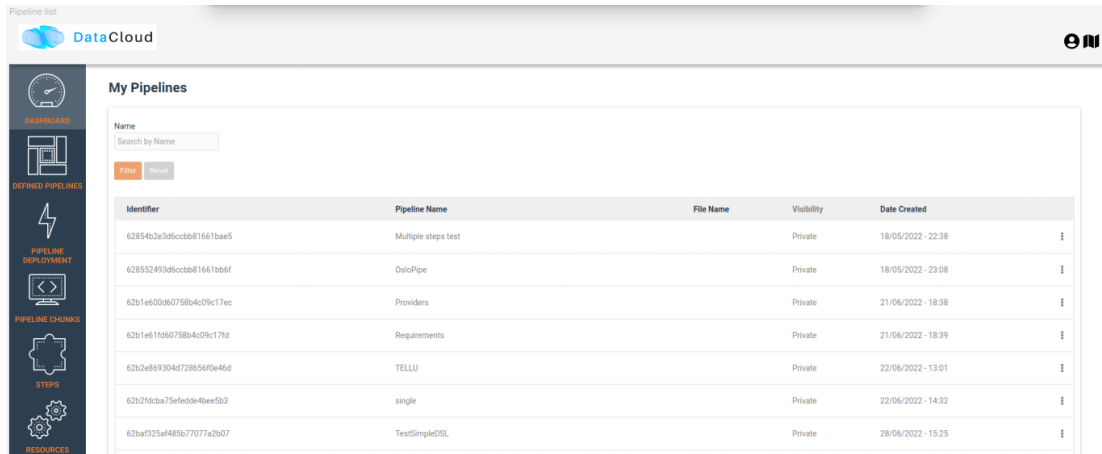


Figure 67: Wireframe for retrieving pipelines from DEF-PIPE

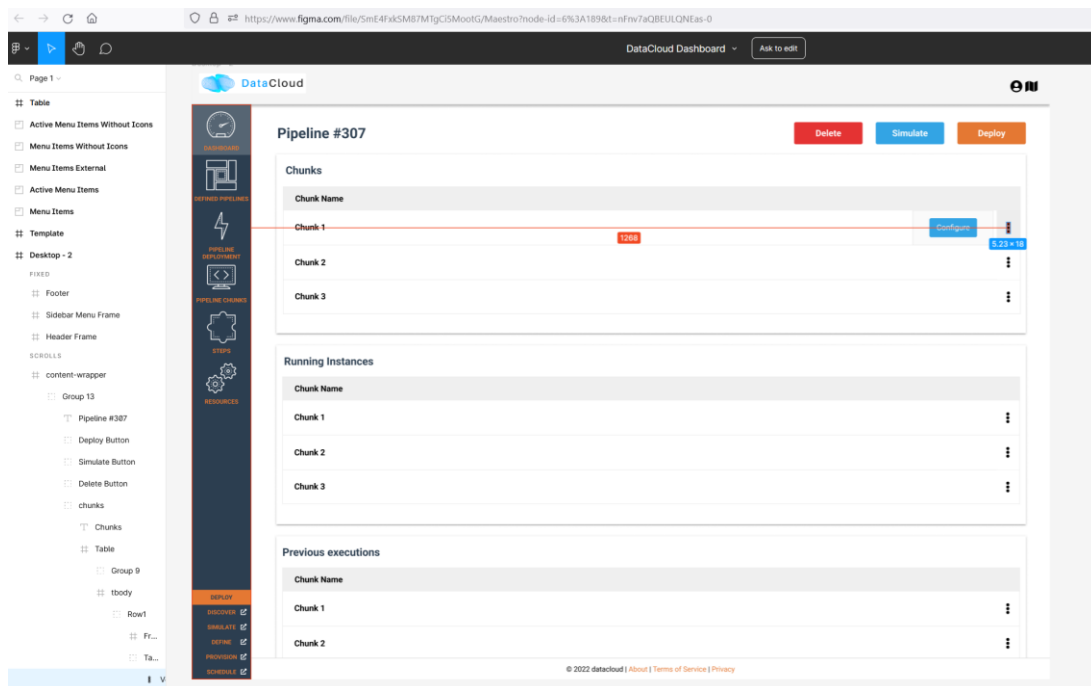


Figure 68: Wireframe for the management of a pipeline and its chunks

¹⁴ <https://www.figma.com/file/RuZvT5oBLhJdBvDCVvzzxy/Dashboard?node-id=0%3A1&t=3CxxsrhGvOzE0jOH-1>

¹⁵ <https://www.figma.com/proto/22s77SjuWsakkdivzdBsw/SIM-PIPE-UI%2FUX-design?node-id=8%3A14&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A2>



As seen in section 4.2.5, some of these views have been already created and are in the process of integration with the relative tools, while DEP-PIPE is re-designed to support these extra steps. For the second release a common (relational) database and a Kafka queue will also be utilized.

Table 17: Aggregated DataCloud features planned for second release of the toolbox

Features	Plan
Development of the missing interface functions to import a DSL pipeline from DEF-PIPE and to export an event log in XES format.	Q1 2023
Development of a querying algorithm to extract information related to the dark data manipulated by the data pipeline.	Q1 2023
Implementation and testing of the algorithm that perform segmentation and trace preprocessing, detailed in [D2.1].	Q1 2023
Implementation and testing of the algorithm that perform event abstraction, detailed in [D2.1].	Q1 2023
More data retrievable queries in GraphQL API	Q2 2023
Access to DEF-PIPE catalogue for quick testing	Q1 2023
Integration of simulation engine	Q2 2023
Support more advanced pipelines.	Q2 2023
Adaptation policies, Scheduling and scaling up/down the data pipeline executions;	Q1 2023
Monitoring policies, Utilization of resources and pipeline chunks.	Q1 2023
Support complex DSL in R-MARKET SDK	Q2 2023
Support longer running task in off-chain	Q2 2023
Share IP addresses of provisioned worker	Q1 2023
Multi-cloud support	Q2 2023
Security Policies Enforcement	Q2 2023
Scale cluster	Q2 2023
Pre-deployment configuration	Q1 2023
Adaptation based on pipeline chunks from ADA-PIPE	Q1 2023
Common database usage	Q1 2023
Kafka setup for async operations	Q2 2023



APPENDIX A – INSTALLATION INSTRUCTIONS

Here we provide instructions for the installation of the various components of DataCloud. More details for each tool can be found in the GitHub page of each tool.

DIS-PIPE

The code has been tested using Python 3.8.12 and conda 4.10.3. At the moment, DIS-PIPE can be deployed correctly on Windows operating systems only. DIS-PIPE will eventually work also for Linux/Mac operating systems.

To install coda follow these links:

<https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html>

Or go directly to Miniconda: <https://docs.conda.io/en/latest/miniconda.html>

It should work also on Anaconda: <https://docs.anaconda.com/anaconda/install/index.html>

Clone the project from <https://github.com/DataCloud-project/DIS-PIPE>

It is really suggested to use the already existing environment.

After downloading the file "environment.yml", open the terminal or an Anaconda Prompt, change directory to go to the location where the file is located and do the following steps:

Create the environment from the environment.yml file:

```
conda env create -f environment.yml
```

1. The first line of the yml file sets the new environment's name, in this case "pm4py_env"
2. Activate the new environment:

```
conda activate pm4py_env
```

3. Verify that the new environment was installed correctly:

```
conda env list OR conda info --envs.
```

4. Open terminal or Anaconda Prompt.
5. Activate "pm4py_env".
6. Change directory to go in the "api" folder downloaded.
7. Run: python backend.py in terminal or prompt;

```
python backend.py
```

8. Go to your browser on <http://127.0.0.1:8080/>

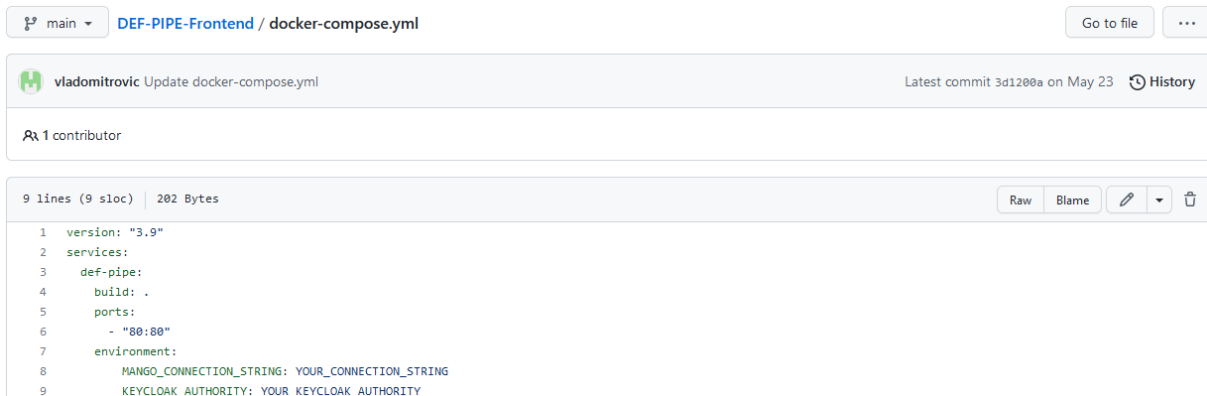


DEF-PIPE

The solution is configured to work in a docker container, the build configuration is located in the [Dockerfile](#) and a [docker-compose.yml](#) is available at the root of the repository.

Clone the repository from <https://github.com/DataCloud-project/DEF-PIPE-Frontend>

In the docker-compose.yml, change the following variables:



The screenshot shows a GitHub repository view for the file `docker-compose.yml` in the `DEF-PIPE-Frontend` repository. The file content is as follows:

```

1 version: "3.9"
2 services:
3   def-pipe:
4     build: .
5     ports:
6       - "80:80"
7     environment:
8       MANGO_CONNECTION_STRING: YOUR_CONNECTION_STRING
9       KEYCLOAK_AUTHORITY: YOUR_KEYCLOAK_AUTHORITY

```

Where, the following apply:

`MANGO_CONNECTION_STRING` → Connection string for MongoDB

`KEYCLOAK_AUTHORITY` → Url of the KeyCloak authorization server

For the frontend, to work properly the following variables should be set:

`REACT_APP_KEYCLOAK_URL` → KeyCloak base url

`REACT_APP_KEYCLOAK_REALM` → KeyCloak realm

`REACT_APP_KEYCLOAK_CLIENT_ID` → KeyCloak client id

After setting the environment variables required bellow, run `docker-compose up` to build the project and start the container.

```
run docker-compose up
```

The application will be reachable on the port 80.



SIM-PIPE

To install SIM-PIPE follow these steps:

SIM-PIPE Sandbox environment set up

The sandbox is required to be installed on a separate Ubuntu Server 20.04 LTS host.

To install Ansible on Ubuntu run the following commands:

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible
```

Run the following command to check that Ansible has been correctly installed:

```
ansible --version
```

Then, setting up the sandbox VM using Ansible:

Step 1. Clone the SIM-PIPE repository

```
git clone -b monorepo https://github.com/DataCloud-project/SIM-PIPE.git
cd SIM-PIPE/sandbox
```

Step 2. Configure the Ansible inventory file

The Ansible inventory file [inventory-datacloud.yaml](#) contains configuration settings for the sandbox VM host, such as the IP address and the SSH port and username.

- `ansible_host`: The IP address of the host. Default value is localhost.
- `ansible_port`: The SSH port of the host. Default value is 22.
- `ansible_user`: The SSH username to use. Default value is ubuntu.
- `ansible_become`: Enables to run tasks as sudo. Default value is true.

Use your favorite editor and change the inventory file settings accordingly

Step 3. Run the Ansible Playbook to setup the sandbox

Run the Ansible Playbook file [setup-sandbox-vm.yaml](#) to setup the sandbox. For more information about Ansible Playbooks see https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

Command line when using private key to authenticate: (Update the command line to point to the location of your keyfile.)

```
ansible-playbook -i inventory-datacloud.yaml setup-sandbox-vm.yaml --key-file ~/.ssh/mykey.pem
```



Command line when using password to authenticate:

```
ansible-playbook -i inventory-datacloud.yaml setup-sandbox-vm.yaml --ask-pass --ask-become-pass
```

The sandbox VM should now have the following services running:

SFTP service listening on port 22

Docker daemon listening on port 2375

SIM-PIPE tool Interface and Simulation Controller

After the sandbox is set up, proceed to install the other components of the SIM-PIPE tool (i.e., User Interface and Simulation Controller). This tool can be run on both Windows and Ubuntu. Below we provide the installation instructions for running the SIM-PIPE tool on Ubuntu.

First clone the project from <https://github.com/DataCloud-project/SIM-PIPE.git>

Then, build the simpipes-backend Docker image

```
sudo docker build -t simpipes-backend -f Dockerfile .
```

The [docker-compose.yaml](#) refers to environment settings that must be updated, such as the IP address of the host running the SIM-PIPE tool and the IP address of the host running the SIM-PIPE Sandbox. These settings are set in the [.env](#) file.

- `HASURA_URL`: URL to the host running hasura service. Default value is `http://192.168.1.6:9000`.
- `SANDBOX_IP`: IP address of the sandbox VM. Default value is `'192.168.56.1'`.
- `CONTAINER_STOP_TIMEOUT`: Number of seconds to wait after sending STOP signal to running container. Default value is 5 seconds.
- `POLLING_INTERVAL`: Polling interval (in seconds) for collecting resource usage statistics. Default value is 1 second.
- `HASURA_ADMIN_SECRET`: Admin password for hasura service. Default value is `hasuraadminsecret` second.
- `POSTGRES_PASSWORD`: Password for TimescaleDB database service. Default value is `postgrespassword` second.

Use your favorite editor and change the IP address in the `HASURA_URL` and the `SANDBOX_IP` settings accordingly.

Then to we run the application, execute the docker compose up command as shown below.

```
sudo docker-compose -f docker-compose.yaml --env-file .env up
```



ADA-PIPE

To integrate ADA-match-scheduler to Kubernetes scheduler, ADA-PIPE used the following software along with the tutorials of their setups:

- Package installer for Python 3.9 (pip3.9);
- [Docker Engine on Ubuntu](#) to install Docker engine along with [Kubernetes on Ubuntu](#) for the x86 instances, and [Kubernetes on \(vanilla\) Raspbian Lite](#) for the ARM-based devices;
- For GPU-enabled NVIDIA devices, we used the [NVIDIA Jetson Linux](#) operating system and the [Raspberry Pi OS](#) for RPi devices;
- [kube-latency](#) to measure bandwidth and latency between the devices;
- [Prometheus operator v0.45.0](#) to monitor the cluster;
- [KubeMQ v2.2.10](#) to transmit the data between pipeline steps through an asynchronous message queue platform;
- [matching library](#) for implementing the modified capacity-aware matching algorithm for ADA-PIPE scheduling tool.

After installing all the libraries and dependencies, you need to run the Python script of "scheduling.py", which collects the monitoring information by [Prometheus Python API](#) from the Kubernetes cluster, and then utilizes the [Python client library v17.17 for Kubernetes](#) to execute the customized ADA-match scheduler. As a result of this, schedules for the application's pods on the appropriate devices are provided for the users.

Moreover, the Frontend source codes of the ADA-PIPE tool is available in the following directory that includes the configuration of the KeyCloak api: <https://github.com/DataCloud-project/ADA-PIPE>



R-MARKET

First, we have to clone the nodejs server of R-MARKET tool from https://github.com/DataCloud-project/R_MARKET_NodeJS.

Then, to run the server, user executes run:

```
npm start
```

This way, the server that provides the API is spawned and the Swagger UI interface for the service is available to <http://localhost:5000/>



DEP-PIPE

1. Clone the DataCloud branch for the repository

```
git clone --branch datacloud https://github.com/ubitech/maestro.git
```

2. Switch to DataCloud branch

<https://gitlab.ubitech.eu/maestro/maestro/-/tree/datacloud>

3. Build project

```
copy /maestro/ext-deps/maven-libraries/settings.xml into your maven folder ./m2
```

```
cd /maestro
```

```
mvn clean install
```

4. Setup the environment

```
cd /maestro/framework
```

```
./setupEnvironment.sh
```

5. Start docker compose

```
cd /maestro/framework/<path>
```

```
docker-compose up -d --build
```

Instructions will be provided in the GitHub page of the tool: <https://github.com/DataCloud-project/DEP-PIPE>

