# Central Lancashire Online Knowledge (CLoK)

| Title | Experimental Research Reproducibility and Experiment Workflow Management |
|---|---|
| Type | Article |
| URL | https://clok.uclan.ac.uk/id/eprint/45676/ |
| DOI | https://doi.org/10.1109/COMSNETS56262.2023.10041378 |
| Date | 2023 |
| Citation | Demchenko, Yuri, Gallenmüller, Sebastian, Fdida, Serge, Andreou, Panayiotis, Crettaz, Cédric and Kirkeng, Mathias (2023) Experimental Research Reproducibility and Experiment Workflow Management. 2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS). ISSN 2155-2487 |
| Creators | Demchenko, Yuri, Gallenmüller, Sebastian, Fdida, Serge, Andreou, Panayiotis, Crettaz, Cédric and Kirkeng, Mathias |

It is advisable to refer to the publisher's version if you intend to cite from the work.
https://doi.org/10.1109/COMSNETS56262.2023.10041378

For information about Research at UCLan please go to http://www.uclan.ac.uk/research/

# Experimental Research Reproducibility and Experiment Workflow Management

Yuri Demchenko
*University of Amsterdam*
The Netherlands
y.demchenko@uva.nl

Sebastian Gallenmüller
*Technical University of Munich*
Germany
gallenmu@net.in.tum.de

Serge Fdida
*Sorbonne University*
France
serge.fdida@sorbonne-universite.fr

Panayiotis Andreou
*UCLan*
Cyprus
PGAndreou@uclan.ac.uk

Cédric Crettaz
*Mandat International*
Switzerland
ccrettaz@mandint.org

Mathias Kirkeng
*University of Amsterdam*
The Netherlands
m.kirkeng@gmail.com

*Abstract*—**Research reproducibility is an important factor to support the full research life cycle; this is especially important for experimental research, where it is required to also reproduce the whole experiment environment and equipment setup. This paper presents the methodology and solution proposed in the SLICES Research Infrastructure to enable research reproducibility in modern digital technologies for complex and large scale experimentation. The paper provides a short overview of existing research and approaches for experimental research reproducibility, generally including git based experiments deployment and operation, Jupyter Notebook and Common Workflow Language (CWL) for workflow management. The paper describes the framework and approaches taken in the SLICES-RI that also address research environment provisioning on demand with the Platform Research Infrastructure as a Service (PRIaaS) and data management infrastructure to ensure data quality and support effective data sharing.**

*Keywords—Reproducibility, Experimental Research, Experiment lifecycle, Experiment Automation, FAIR data principles, Data Management Infrastructure, SLICES Research Infrastructure.*

## I. INTRODUCTION

Modern research is increasingly multidisciplinary and data driven, emphasizing the need for effective communication between researchers and data sharing. The Open Science initiative and movement among research communities can address these needs and increase the overall efficiency of scientific and technology research [1]. Open Science is strongly supported by policy development and funding bodies in Europe and a mandatory requirement in the current Horizon Europe programme. In recent years, major initiatives and projects to create the foundation and e-Infrastructure for Open Science have been funded in Europe (in the framework of the past Horizon 2020 and current Horizon Europe programmes). This includes the currently operational OpenAIRE [2] and Zenodo [3] services, the development of the European Open Science Cloud (EOSC) [4], all support the FAIR (Findable, Accessible, Interoperable, Reusable) data principles [5]. The FAIR data principles were initially proposed for research data management primarily targeted at consistent metadata management. However, recent initiatives by the Research Data Alliance (RDA) [6] are extending FAIR principles to scientific software as a digital or data object.

Research reproducibility is one of the core principles of Open Science [7]. However, developments in this area are fragmented and lacking a common or widely used approach. One of the difficulties is that reproducible research requires the recreation or provisioning on demand research environment, even for general Data Science and Analytics tasks. Reproducibility of experimental research imposes additional requirements on the reproducible experiment setup, including resources provisioning, experiment environment setup, and experiment lifecycle management, which in its own turn include experimental data lifecycle management.

Reproducibility, according to ACM [9], is defined as a three-stage process. The first stage, repeatability, is achieved if the same research group can recreate experiments using the same equipment. The second stage, *reproducibility*, is reached if a different team can recreate experiments on the same equipment. If an independent team recreates experiments on their own equipment, the final stage is reached: *replicability*. To incentivize reproducible research, the ACM launched a multi-stage reproducibility award system for scientific papers based on their definition. A study considered the results of such a system helpful but time-consuming to implement [10] and therefore faces a weak adoption.

The focus of our paper is the investigation of a central component for the reproducible experimental research methodology - a reproducible workflow to execute experiments. It should also be supported by a consistent data management infrastructure. This research is a part of the SLICES Research Infrastructure to support research on digital infrastructures [8]. We aim to achieve the following goals:

- Analyze different existing approaches for experimental workflows
- Identify the benefits and disadvantages of these existing approaches
- Create guidelines for an ideal experimental workflow to validate hypotheses in computer science

The paper is structured as follows. Section II provides information about European Open Science projects and introduces the SLICES Research Infrastructure for digital infrastructures experimentation. Section III gives a short overview of existing approaches and practices on research reproducibility and solutions for experiment facilities provisioning and workflow description. Section IV discusses the experiment reproducibility in SLICES-RI. Section V discusses the Data Management Infrastructure requirements, an important component of consistent experimental research reproducibility. The paper concludes with the summary and suggestions for future research in Section VI.

## II. OPEN SCIENCE AND FAIR DATA PRINCIPLES IN EOSC

### A. European Open Science Cloud (EOSC)

EOSC is an initiative and programme by the European Union to provide European researchers, innovators, companies and citizens

with a federated and open multi-disciplinary environment where they can publish, find and re-use data, tools and services for research, innovation and educational purposes [4]. So far, the EOSC projects have created the foundation for research data interoperability and integration for European RIs. The EOSC Strategic Research and Innovation Agenda (SRIA) provides a roadmap to achieve the EOSC vision and objectives, namely to deliver an operational "Web of FAIR data and services" for science [11]. The Minimum Viable EOSC (MVE) achieved by the end of 2021 [11], created a starting point for future EOSC development generally coordinated by the EOSCFuture project [12]. MVE defines EOSC Core that is designed to provide a federated data exchange environment for research projects and communities where data comply with FAIR principles. Ongoing developments aim at providing a customisable research environment for researchers and research projects using services provided by the EOSC Portal Catalog and Marketplace [13].

The ongoing RELIANCE project intends to extend the EOSC with a set of services for Research Lifecycle Management in accordance with FAIR principles based on Research Objects (RO), Data Cubes and Text Mining [14]. ROHub is a service by RELIANCE for the storage, lifecycle management, and preservation of scientific research, campaigns and operational processes via research objects [15].

### B. SLICES-RI to support Digital Technologies Research and Experimentation

The Scientific Large-scale Infrastructure for Computing/ Communication Experimental Studies (SLICES) [8] is a distributed Digital Infrastructure designed to support large-scale experimental research focused on networking protocols, radio technologies, services, data collection, parallel and distributed computing and, in particular, cloud and edge-based computing architectures and services. This encompasses the full range of network, computing, and storage functions required for on-demand services across many verticals and addresses new complex research challenges, supporting disruptive science in IoT, networks and distributed systems. SLICES will integrate multiple experimental facilities and testbeds operated by partners providing a common services access and integration platform. SLICES will allow academics and industry to experiment and test the whole spectrum of digital technologies whereby the computing, network, storage, and IoT resources can be combined to design, experiment, operate, and automate the full research lifecycle management.

### III. RELATED WORK

This section analyzes existing approaches related to the creation of reproducible experiment workflows.

### A. Generaal Tools

Experiment results are highly dependent on a system state during the experiment. To allow for experiment repetition and replication, the documentation and recreation of this system state are essential. Bajpai et al. [16] provide several recommendations to simplify and ensure this process. Based on their recommendations and our own experience, we recommend the following tools: Configuration management and deployment frameworks, such as Ansible [17], help automate this task. At the same time, automation avoids any impact of the experimenter on the results. Version control systems such as git [18] help track the version of investigated source code.

A widely used tool to perform experimental evaluations is Jupyter [19]. Jupyter provides notebooks, a convenient way to combine documentation, code, and visual representations within a single file that can be accessed via a web browser and easily shared with other researchers.

### B. Experiment Control

Multiple initiatives maintain and provide testbeds for research, e.g., Fed4Fire (EU) [20], OneLab (EU) [21], Grid'5000 (France) [22], Planetlab (global) [23], or GENI (USA) [24]. Though not explicitly designed for reproducibility, Nussbaum [25] argues that testbeds, such as Grid'5000 or CloudLab, allow reproducible experiments if used correctly.

Whereas the previously mentioned approaches mainly focus on resource allocation, there are also more high-level approaches. These approaches define the experimental workflow. Examples of such solutions are OMF [26] or NEPI [27] that allow the definition and automated execution of experiment workflows. These controllers may use one of the previously mentioned testbeds as a backend to execute their experiments. Besides the previously mentioned solutions, new approaches emerged that combine resource allocation and experiment control. This integration allows full control over the entire experiment workflow laying the foundation to perform reproducible experiments:

#### 1) Chameleon platform for Computer Science Research

The Chameleon cloud platform [28] is a large-scale, deeply reconfigurable experimental platform built to support an experimental workflow for Computer Sciences systems research. The Chameleon Infrastructure (CHI - Cloud++) is a cloud platform powered by OpenStack with bare metal (re-)configuration (Ironic), using the OpenStack Blazar reservation service for experimental resource reservation.

Chameleon uses Jupyter notebooks as an experiment definition, execution, and sharing format. The testbed can be controlled through libraries in the Jupyter notebook. A collection of various experiments is provided via the Chameleon website [29, 30].

The Chameleon/CHI experimental workflow includes stages related to resource discovery, allocation, dynamic configuration, orchestration, and monitoring. The workflow management service is supported by a rich library of orchestration templates and images created by the research community.

Chameleon provides Jupyter integration for orchestration via JupyterLab service/portal [31], which allows creating and managing reproducible experiment workflows via Jupyter Notebooks that can be created and shared by researchers (or research teams).

#### 2) Plain orchestrating service (pos)

The plain orchestrating service (pos) [32, 33] provides two components, a testbed controller and an experiment workflow. The testbed controller takes care of the allocation and management of experimental resources. It provides bare-metal access to the experiment nodes. Images for the experiment nodes are provided in the form of live Linux images. Using live images for experiments has two benefits: First, rebooting an experiment node helps reset the system to a well-defined state. Second, testbed users are aware of the non-permanence of their configuration, gently pushing users towards documenting and automating experiment configuration.
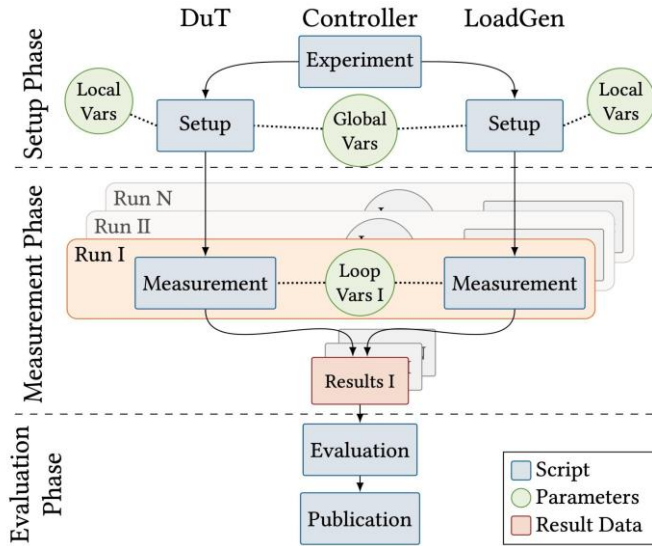
Figure 1: pos experimental workflow

Figure 1 demonstrates the pos experiment workflow, divided into the setup, measurement and evaluation phases. Three nodes are participating in the displayed experiment, a device under test (DuT), a load generator (LoadGen), and the controller. An experiment can be started by executing the experiment script on the controller. The controller reboots DuT and LoadGen. After the reboot, the two nodes can be configured using a setup script. The pos controller further provides variables (vars) to parameterize the setup. The measurement phase starts after the setup phase. There, different measurement scripts are executed on the experiment hosts. Measurements are typically parameterized using so-called loop variables. The number of loop variables determines how often the measurement script is run. For each run, a separate set of result files is generated and associated with a specific instance of loop variables acting as metadata to describe the result files. Result files are uploaded to the controller for further processing. After all result files are collected, the evaluation phase begins using an evaluation script to perform the evaluation. A publication script finally collects and prepares all experiment artifacts for release. The well-defined file structure of the experiment allows for an automated collection and processing of files. A possibility to present the results is a website thatcan be generated automatically and hosted on GitHub in conjunction with the other experiment artifacts.

A user can write the different script files in any language of his/her choice, however, the hosts must be able to execute it. The setup script, for instance, can be a simple shell script or a more complex Ansible configuration.

### C. Experiment lifecycle and experiment workflow description

#### 1) GitHub

GitHub is widely used for managing scientific code and data and, in particular, code for running experiments and processing experimental data. In this case, GitHub is used by scientific programmers in the same way as software developers, also benefiting from powerful functionality for code sharing, integration, and deployment (also referred to CI/CD process of Continuous Integration and Continuous Deployment).

However, using GitHub for experiment automation is limited by code portability, which depends on the individual scientific programmer style and may not provide sufficient code structuring

and formalised interfaces to infrastructure computational and storage services.

#### 2) Jupyter Notebook

Jupyter Notebooks are widely used in the scientific community for scientific data analysis and reporting; however, recent developments and uses are targeting the full scientific research cycle, including the full cycle of the experiment development and exploration.

The CHI uses Jupyter Notebooks for defining and running experimental workflows. However, in Chameleon, Jupyter Notebooks are used primarily for running experiments on already provisioned experimental infrastructure, which is provisioned in the infrastructure provisioning workflow. A Paper by Beg et al. [35] describes other cases of using Jupyter Notebooks for supporting reproducible experimental research.

SLICES-RI will review the Grid'5000 [36] experience to support using Jupyter Notebooks for different aspects/activities in the experiment automation and experimental research data management [37]:

- Notebooks as experiment drivers. These notebooks run the experiments from their beginning to their end, starting with resource reservations and going at least to data collection;
- Notebooks as experimental payload. The code contained within these notebooks is the core of experiments. These notebooks run on the reserved resources, and either contain or control the computation that is the subject matter of the experiment;
- Notebooks for post-processing. These notebooks are executed after an experiment to process the results. Supporting this usage will be dependent on a testbed infrastructure and the type of post-processing expected;
- Notebooks for exploratory programming are exploited by users as an enhanced interactive shell to create new code through trial and error;
- Notebooks as tutorials. Those are notebooks provided to the users by teachers that aim to present and explain to the users some specific concepts.

The usage of Jupyter Notebooks was already put in place in the context of the Fed4FIRE+ project by several testbeds offering Jupyter Notebooks to reproduce experiments. These testbeds are currently included as legacy testbeds in SLICES-RI via partners.

To achieve experiment reproducibility, the experimental platform must provide well-defined interfaces to experimental resources and data services that can be connected to the Jupyter programming environment. This is available in the Chameleon scientific cloud and provided by the major public cloud and Big Data infrastructure providers such as AWS with their SageMaker Studio Notebook [37] and Microsoft Azure Data Studio Notebook service [38] that is also supported by the Azure DevOps for Data Science platform [39].

#### 3) Common Workflow Language (CWL)

To achieve experiment workflow portability (in addition to experiment reproducibility), the scientific community uses scientific workflow languages. Succeeding a multitude of workflow languages introduced in the past, the CWL is gaining popularity in recent times.

The CWL [40] is a specification to describe digital workflows. It describes how multiple steps in a computational workflow and their connections should be defined. CWL itself is only a specification, so a user needs a program to execute workflows called

runner. A reference implementation of such a runner is cwltool [41], however several workflow management systems implement CWL support, for example, Apache Airflow [42], StreamFlow [43], and Toil [44]. Other workflow management systems offer partial or experimental support, such as Galaxy [45].

The standard defines CWL tools described in '.cwl' files using a subset of YAML. These tools can execute command line tools, evaluate javascript expressions, or define abstract operations to be implemented by a specific CWL runner. For each tool, the inputs and outputs need to be defined. Requirements can be listed, such as required software, the ability to process inline Javascript or certain files or directories to be present during a run time, among other requirements.

These steps can be combined into workflows, also defined in '.cwl' files using a subset of YAML. Each workflow contains a list of steps, with each step having defined inputs and outputs. These steps are not necessarily supposed to be executed in order but rather according to their dependencies on other steps. Independent steps can thus also be run in parallel. These steps execute CWL tools, either defined in the workflow itself or referencing a tool defined in a separate CWL file. The steps can also execute other CWL workflows, which allows for workflows to be nested. This works because, for each workflow, the inputs and outputs need to be defined. Finally, arbitrary metadata and metadata according to certain schemas can be defined in the workflows as well. The inputs of a workflow or tool are listed in a separate '.yml' file provided to the CWL runner at runtime. A tool or workflow step can also be set to scatter, meaning it runs multiple times for each element of an array of inputs.

*D. CWL for a sample experiment*

The data processing step in the sample experiment was implemented using CWL, specifically using the reference implementation of a CWL runner. The full workflow requires the AWS access credentials and the name of the DynamoDB table containing the sensor data. After execution, the workflow has produced the sensor data in CSV format sorted by date-time and a description as well as a line chart of the sensor data. The full details of the presented here example can be found in the project deliverable to be published after the project review [46].

The first step of the processing runs a CWL tool that retrieves the MQTT sensor data from the DynamoDB table using the boto3 python module. This tool uses AWS credentials to authenticate the client as well as the name of the table from which to retrieve the sensor data and outputs the sensor data in JSON format. The next step in the workflow converts the data from the JSON format to a CSV file using the JQ [47] command line tool. This CSV file is then sorted by the date-time of the sensor measurements in the next workflow step. The sorted CSV is one of the outputs of the workflow as well as the input to a CWL tool that uses the python pandas [48] library to create a description of the data, including information such as the mean and standard deviation. This description is the second output of the workflow. Finally, the sorted CSV is also used as input to a CWL tool, which uses the gnuplot [49] application to create a line chart of the sensor data.

The following code shows the contents of the CWL workflow used for the data processing step of the sample experiment with comments explaining the code. The tools used in each of the steps are defined in separate CWL files and referenced in the workflow code.

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow

# The inputs of the workflow as a whole
# These are referenced in the first workflow step
inputs:
    AWS_ACCESS_KEY_ID: string
    AWS_SECRET_ACCESS_KEY: string
    table_name: string

# In the following list the workflow steps are defined
steps:
    # the first step, called "get_data" gets the sensor data from the
DynamoDB table
    get_data:
        run: ../tools/get-dynamodb-data.cwl # the CWL tool is defined
in this file
        # the following list defines the inputs to the CWL tool
        in:
            AWS_ACCESS_KEY_ID: AWS_ACCESS_KEY_ID
            AWS_SECRET_ACCESS_KEY: AWS_SECRET_ACCESS_KEY
            table_name: table_name
        # the output of this workflow step is defined as "dynamodb_data"
        out: [dynamodb_data]

    # the second step of the workflow converts the sensor data from
JSON to CSV
    convert_to_csv:
        run: ../tools/json-to-csv.cwl
        in:
            #  the  input  is  the  output  of  the  previous  step,
"dynamodb_data"
            json_file: get_data/dynamodb_data
        out: [csv_file]

    # the third step sorts the sensor data in CSV format
    sort_csv:
        run: ../tools/sort.cwl
        in:
            file_to_sort: convert_to_csv/csv_file
            sort_field:
                default: 2 # which column to sort by
        out: [sorted_file]

    # the 4th step creates a description of the data
    describe_data:
        run: ../tools/describe-csv.cwl
        in:
            # the input is the sorted CSV file from the previous step
            csv_file: sort_csv/sorted_file
        out: [data_description]

    # the 5th step generates a line plot
    generate_graph:
        run: ../tools/graph-csv.cwl
        in:
            # the input is also the sorted CSV file from the 3rd step
            csv_to_plot: sort_csv/sorted_file
        out: [plot]

    # the outputs of the workflow as a whole are the sorted CSV file
from the third
    # step, the data description from the 4th step and the line chart
from the 5th
    # step
outputs:
    data_csv:
        type: File
        outputSource: sort_csv/sorted_file
    description:
        type: File
        outputSource: describe_data/data_description
    plot:
        type: File
        outputSource: generate_graph/plot
```

The deployment and execution of the experiment is done with Ansible playbooks and CloudFormation infrastructure component templates. The solution has been deployed and tested on the AWS cloud and proved that the use of templates both for cloud resources

and infrastructure and for experiment workflow provides an effective instrument and platform for the SLICES experiments automation for the whole experiment lifecycle.

## IV. EXPERIMENTAL RESEARCH REPRODUCIBILITY IN SLICESI

### A. Adopting pos and Chameleon in SLICES

In the following, we discuss how different approaches for experimental workflows can be integrated in SLICES.

Therefore, we selected the previously discussed approaches for reproducible experiment workflows offered by the Chameleon and pos testbeds. Chameleon achieves that by using Jupyter notebooks to provide a single file to document and describe the experiment workflow and evaluation. The collection of all experimental artifacts within a single file allows for easy sharing of experiments. The pos framework uses Linux live images and a structured collection of scripts to run and describe experiments. Both approaches offer enough flexibility to be combined. The previously separate scripts of the pos approach can be converted to code cells allowing the pos workflow structure within Jupyter notebooks. To demonstrate the integration of pos/chameleon, we created a Jupyter notebook [33] representing an experiment combining both approaches. We see the combined workflow as a prototype for future SLICES experiment workflow, providing reproducibility, easy sharing, and flexibility for researchers

### B. PRIaaS to support RI services provisioning for Experiments Reproducibility

The Platform Research Infrastructure as a Service (PRIaaS) proposed by the authors in the research paper [50] is proposed as an architectural solution to provide on-demand fully functional environment for experimental research on SLICES-RI to deliver specialised and community-oriented services. The main component of PRIaaS is the Actualisation platform that leverages the TeleManagement Forum Digital Platform Reference Architecture [51] and allows the composition and instantiation of a fully operational Virtual RI (VRI) configured for specific customer research purposes.

The VRI provisioning process is based on well-known and commonly used DevOps tools and is supported by the management and operation functions. As the PRIaaS platform progresses, the repository of the design patterns, templates and containerized applications and functions will grow. PRIaaS will allow natural integration with the EOSC Portal and Catalog services, sharing resources and experiment templates.

## V. DATA MANAGEMENT INFRASTRUCTURE AND DATA LIFECYCLE

### A. Experimental Data Management stages

Management of experimental data is an important aspect of SLICES-RI, and it includes several services that must support all stages of the experimental data lifecycle. As illustrated in Figure 2, SLICES-RI operates Data Storage and Management Infrastructure to support activities typical for experimental research, such as experiment planning and deployment (as explained in the previous sections), the discovery of data from internal data archives and external sources that are needed for correct experiment planning and setup as well as data publication and sharing. SLICES Data Management Infrastructure establishes policy for data governance and management, including data security and quality assurance, that are supported by corresponding infrastructure tools for data curation. Figure 2 indicates that the SLICES Data Management infrastructure should be inter-connected with the EOSC Scientific Data Infrastructure for data sharing and access.
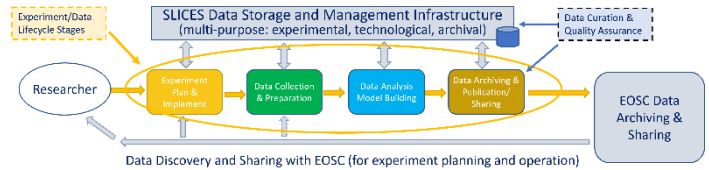


Figure 2 – SLICES Data Management stages and supporting infrastructure components.

Each Data Lifecycle stage, i.e., experiment setup, data collection, data analysis, and finally, data archiving - typically works with its own data sets, which are linked and their transformation must be recorded in the process that is called lineage (that can also be extended to provenance for complex linked scientific data). All staged datasets need to be stored for the purpose and possibly reused in later processes.

Many experiments require already existing datasets that will be available in the SLICES data repositories or can be obtained/discovered in EOSC data repositories

### B. Infrastructure components to support the experimental data management

The following are requirements for robust data management infrastructure (DMI) for experimental data that follows from best practices and use cases analysis in the SLICES-DS project [8, 45]:

RDM1. Distributed data storage and experimental data(set) repositories should support common data and metadata interoperability standards, in particular common data and metadata formats. Outsourcing of data storage to the cloud must be protected with appropriate access control and compliant with the SLICES Data Management policies.

RDM2. SLICES DMI should support the whole data lifecycle. It should provide interfaces to experiment workflow and staging

RDM3. SLICES DMI shall provide PID (Persistent IDentifier) and FDO (FAIR Digital Object) registration and resolution services to support linked data and data discovery that should be integrated with EOSC services.

RDM4. SLICES DMI must support (trusted) data exchange and transfer protocols that allow policy-based access control to comply with the data protection regulations.

RDM5. SLICES DMI must enforce user and application access control and identity management policies adopted by SLICES community that can be potentially federated with the EOSC Federated AAI

RDM6. Procedures and policies must be implemented for data curation and quality assurance.

RDM7. Certification of data and metadata repositories should be considered at some maturity level following certification and maturity recommendations by RDA

Strategy for practical SLICES DMI deployment must include well-defined procedures for distributed data storage integration and linking to ensure data is discoverable/findable and accessible across all SLICES-RI. This should also relate to using external community and cloud-based storage, and a clear procedure should be developed for data migration.

SLICES will consider connecting to and using EOSC community services to build a hybrid data management infrastructure that may include both its own data storage, as part of the private cloud, and external data storage offered by EOSC and EGI community. The use of public cloud storage and file sharing services will be regulated by data management policies.

## VI. Conclusion and Further Developments

Methodologies and tools for experimental research reproducibility still have a long way to go to achieve the maturity level to be widely adopted by different scientific disciplines. Following the experience and best practices in recent projects and ongoing research will facilitate the development of commonly accepted standards, specific to their respective field and will increase scholarly communication and research data sharing. Our paper presented ongoing research and developments in the SLICES-RI related projects and proposed solutions and important building blocks toward experimental research reproducibility and automation for digital technologies and computer science. The proposed solution brings together tools and practices used in DevOps, cloud native and platform design as well as research data management and Open Science.

A lot remains to be done in future work in order to make reproducibility practical and more widely adopted. Important developments will be undertaken in the course of the SLICES-RI to provide valuable services for the research community.

### References

[1] Open Science [online] https://www.fosteropenscience.eu/content/what-open-science-introduction
[2] OpenAIRE [online] https://www.openaire.eu/en/home
[3] Zenodo. [online] https://zenodo.org/
[4] EOSC Association [online] https://eosc.eu/about-eosc
[5] FAIR Data Principles [online] https://www.go-fair.org/fair-principles/
[6] Research Data Alliance [online] https://www.rd-alliance.org/
[7] What are reproducibility and replicability? [online] https://www.surrey.ac.uk/library/open-research/reproducibility
[8] S. Fdida, N. Makris, T. Korakis, R. Bruno, A. Passarella, P. Andreou, B. Belter, C. Crettaz, W. Dabbous, Y. Demchenko, and R. Knopp, "Slices, a scientific instrument for the networking community," *Comput. Commun.*, vol. 193, pp. 189–203, 2022.
[9] ACM, "Artifact Review and Badging Version 1.1," 2020.
[10] D. Saucez, L. Iannone, and O. Bonaventure, "Evaluating the artifacts of SIGCOMM papers," Comput. Commun. Rev., vol. 49, no. 2, pp. 44–47, 2019
[11] Strategic Research and Innovation Agenda (SRIA) of the European Open Science Cloud (EOSC), Version 1.0, 21 June 2021 [online] https://op.europa.eu/nl/publication-detail/-/publication/f9b12d1d-74ea-11ec-9136-01aa75ed71a1
[12] EOSC Future Project [online] https://eoscfuture.eu/
[13] EOSC Portal Catalog and Marketplace [online] https://marketplace.eosc-portal.eu/
[14] RELIANCE Project [online] https://www.reliance-project.eu/
[15] ROHub [online] http://reliance.rohub.org
[16] V. Bajpai, A. Brunstrom, A. Feldmann, W. Kellerer, A. Pras, H. Schulzrinne, G. Smaragdakis, M. Wählisch, and K. Wehrle, "The dagstuhl beginners guide to reproducibility for experimental networking research," *Comput. Commun. Rev.*, vol. 49, no. 1, pp. 24–30, 2019.
[17] Redhat Ansible, https://www.ansible.com/. Last accessed: 2022-11-16.
[18] GetHub [online] https://github.com/. Last accessed: 2022-11-16.
[19] Jupyter [online] https://jupyter.org/. Last accessed: 2022-11-16.
[20] Fed4Fire [online] https://www.fed4fire.eu/. Last accessed: 2022-11-16.

[21] OneLab [online] https://onelab.eu/. Last accessed: 2022-11-16.
[22] Grid'5000 [online] https://www.grid5000.fr/. Last accessed: 2022-11-16.
[23] Planetlab [online] https://www.planet.com/. Last accessed: 2022-11-16.
[24] Geni [online] https://portal.geni.net/. Last accessed: 2022-11-16.
[25] L. Nussbaum, "Testbeds support for reproducible research," in *Pro- ceedings of the Reproducibility Workshop, Reproducibility@SIGCOMM 2017, Los Angeles, CA, USA, August 25, 2017*, pp. 24–26, ACM, 2017.
[26] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking testbeds," *ACM SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, pp. 54–59, 2009.
[27] A. Quereilhac, M. Lacage, C. D. Freire, T. Turletti, and W. Dabbous, "NEPI: an integration framework for network experimentation," in *19th Int Conf on Software, Telecommunications and Computer Networks, SoftCOM 2011, Split, Croatia, September 15-17, 2011*, pp. 1–5, IEEE, 2011.
[28] Chameleon Cloud: Experiment in the Edge to Cloud continuum https://www.chameleoncloud.org/media/filer_public/8d/a8/8da8b517-fd99-46ce-94ea-61d4edd94531/cluster.pdf, [Last accessed 26 August 2022].
[29] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *2020 USENIX Annual Technical Conference, USENIX ATC 2020, July 15-17, 2020* (A. Gavrilovska and E. Zadok, eds.), pp. 219–233, USENIX Association, 2020.
[30] Chameleon Trovi Sharing Portal, https://chameleoncloud.readthedocs.io/en/latest/technical/sharing.html. Last accessed: 2022-11-16.
[31] JupyterLab Documentation, https://jupyterlab.readthedocs.io/en/stable/index.html,
[32] S. Gallenmüller, D. Scholz, H. Stubbe, and G. Carle, "The pos framework: a methodology and toolchain for reproducible network experiments," in *CoNEXT '21: The 17th Int Conf on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021* (G. Carle and J. Ott, eds.), pp. 259–266, ACM, 2021.
[33] S. Gallenmüller, A pos Experiment in a Jupyter-Notebook https://github.com/gallenmu/pos-jupyter/blob/main/pos-experiment.ipynb
[34] Marijan Beg, Juliette Taka, Thomas Kluyver, Alexander Konovalov, Min Ragan-Kelley, Nicolas M. Thiéry, Hans Fangohr, Using Jupyter for reproducible scientific workflows [online] https://arxiv.org/ftp/arxiv/papers/2102/2102.09562.pdf
[35] Grid'5000 a computer science testbed based in France - https://www.grid5000.fr/w/Grid5000:Home, [Last accessed 26 August 2022].
[36] Luke Bertot, Lucas Nussbaum, Leveraging Notebooks on Testbeds: the Grid'5000 Case, Proc IEEE INFOCOM 2021 - IEEE Conf on Computer Communications Workshops (INFOCOM WKSHPS), 2021 [online] https://ieeexplore.ieee.org/document/9484501
[37] AWS SageMaker Studio Notebook [online] https://docs.aws.amazon.com/sagemaker/latest/dg/notebooks.html
[38] Azure Data Studio [online] https://azure.microsoft.com/en-gb/services/developer-tools/data-studio/, [Last accessed 26 August 2022].
[39] Team Data Science Process for data scientists, Article, 2022, [online] https://docs.microsoft.com/en-us/azure/architecture/data-science-process/team-data-science-process-for-data-scientists
[40] CWL: Common Workflow Language [online] https://www.commonwl.org/v1.2/, [Last accessed 26 August 2022].
[41] CWL Tools [online] https://github.com/common-workflow-language/cwltool
[42] https://github.com/Barski-lab/cwl-airflow, [Last accessed 26 August 2022].
[43] https://streamflow.di.unito.it/, [Last accessed 26 August 2022].
[44] https://github.com/DataBiosphere/toil, [Last accessed 26 August 2022].
[45] https://galaxyproject.org/, [Last accessed 26 August 2022].
[46] SLICES-DS Deliverable D4.5: SLICES infrastructure and services integration with EOSC, Open Science and FAIR: Recommendations and design patterns (final report). August 2022. To be published Feb 2023
[47] jq JSON processor, https://stedolan.github.io/jq/, [online]
[48] Pandas, https://pandas.pydata.org/, [Last accessed 26 August 2022].
[49] Gnuplot, http://www.gnuplot.info/, [Last accessed 26 August 2022].
[50] Y. Demchenko, C. de Laat, W. Los, and L. Gommans, "Defining platform research infrastructure as a service (priaas) for future scientific data infrastructure," in Designing Data Spaces: The Ecosystem Approach to Competitive Advantage (B. Otto, M. ten Hompel, and S. Wrobel, eds.), pp. 241–260, Springer, 2022.
[51] IG1157 Digital Platform Reference Architecture Concepts and Principles v5.0.1, 21 July 2020 [online] https://www.tmforum.org/resources/reference/ig1157-digital-platform-reference-architecture-concepts-and-principles-v5-0-0/