Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment

Zhao, Z.; Koulouzis, S.; Bianchi, R.; Farshidi, S.; Shi, Z.; Xin, R.; Wang, Y.; Li, N.; Shi, Y.; Timmermans, J.; Kissling, W.D.

DOI
10.1002/spe.3098

Publication date
2022

Document Version
Final published version

Published in
Software - Practice and Experience

License
CC BY-NC

Link to publication

Citation for published version (APA):
Zhao, Z., Koulouzis, S., Bianchi, R., Farshidi, S., Shi, Z., Xin, R., Wang, Y., Li, N., Shi, Y., Timmermans, J., & Kissling, W. D. (2022). Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment. *Software - Practice and Experience*, *52*(9), 1947-1966. https://doi.org/10.1002/spe.3098

Received: 24 November 2021 | Revised: 21 April 2022 | Accepted: 3 May 2022

DOI: 10.1002/spe.3098

RESEARCH ARTICLE

WILEY

Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment

Zhiming Zhao[1,2] | **Spiros Koulouzis**[1,2] | **Riccardo Bianchi**[1,2] | **Siamak Farshidi**[1] | **Zeshun Shi**[1] | **Ruyue Xin**[1] | **Yuandou Wang**[1] | **Na Li**[1] | **Yifang Shi**[2,3] | **Joris Timmermans**[2,3] | **W. Daniel Kissling**[2,3]

[1]Multiscale Networked Systems, University of Amsterdam, Amsterdam, The Netherlands

[2]LifeWatch ERIC, Virtual Lab & Innovation Center (VLIC), Amsterdam, Netherlands

[3]Institute for Biodiversity and Ecosystem Dynamics (IBED), University of Amsterdam, Amsterdam, The Netherlands

Correspondence
Zhiming Zhao, Multiscale Networked Systems, University of Amsterdam, Science Park 904, 1098XH Amsterdam, The Netherlands.
Email: z.zhao@uva.nl

Funding information
EU Horizon 2020 Marie Skłodowska-Curie Actions, Grant/Award Number: 860627; EU Horizon 2020 Framework Programme, Grant/Award Numbers: 824068, 862409, 825134; Microsoft, Grant/Award Number: AI4E-1111-Q3N7-20100806; Netherlands eScience Center, Grant/Award Number: ASDI.2016.014

Abstract

Virtual research environments (VREs) provide user-centric support in the lifecycle of research activities, for example, discovering and accessing research assets or composing and executing application workflows. A typical VRE is often implemented as an integrated environment, including a catalog of research assets, a workflow management system, a data management framework, and tools for enabling user collaboration. In contrast, notebook environments like Jupyter allow researchers to rapidly prototype scientific code and share their experiments as online accessible notebooks. Jupyter can support several popular languages used by data scientists, such as Python, R, and Julia. However, such notebook environments do not have seamless support for running heavy computations on remote infrastructure or finding and accessing collaborative software code inside notebooks. This article investigates the gap between a notebook environment and a VRE and proposes an embedded VRE solution for the Jupyter environment called Notebook-as-a-VRE (NaaVRE). The NaaVRE solution provides functional components via a component marketplace and allows users to create a customized VRE on top of the Jupyter environment. From the VRE, a user can search research assets (data, software, and algorithms), compose workflows, manage the lifecycle of an experiment, and share the results among users in the community. We demonstrate how such a solution can enhance a legacy workflow that uses Light Detection and Ranging (LiDAR) data from country-wide airborne laser scanning surveys for deriving geospatial data products of ecosystem structure at high resolution over broad spatial extents. This enables users to scale out the processing of multi-terabyte LiDAR point clouds for ecological applications

Abbreviations: AAI, authentication and authorization infrastructure; EBVs, essential biodiversity variables; LiDAR, Light Detection and Ranging; NaaVRE, notebook-as-a-VRE; PaaS, platform-as-a-service; SLA, service level agreement; VL, Virtual Lab; VREs, virtual research environments.

to more data sources in a distributed cloud environment. Similar applications could be developed for workflows producing other essential biodiversity variables.

KEYWORDS
cloud, Jupyter, virtual research environment

1 | INTRODUCTION

The study of many scientific problems, for example, big environmental challenges or cancer diagnosis, requires large data volumes, advanced modeling techniques, and distributed computing facilitates.[1,2] Researchers often have to reuse research assets, for example, observational data or medical images, AI models, workflows, and infrastructure services from different parties for building computational experiments to conduct such investigations. Specifically, researchers need effective collaborative support environments for conducting advanced data sciences research:[3] discovery access, interoperation, and reuse of the research assets, and integration of all resources into cohesive observational, experimental, and simulation investigations with replicable workflows.

Virtual research environments (VREs) support scientists in the life cycle of their research activities by enabling effective discovery and selection of data, software services, and other relevant research assets from different sources, construction of cohesive workflows, and collaboration with other scientists.[4] VREs are also called Virtual Laboratories or Science Gateways.[5,6] Examples include VRE4EIC[*], D4Science[†], EVER-EST[‡], and Galaxy[§].

Graphical environments, workflow management systems, and data analytics tools are typical components of such environments. The development of a VRE is often driven by practices from a specific user community, for example, for managing scientific workflows and sharing their research results, resulting in specific graphical environments, workflow management systems, and data analytics tools to be components of the VRE. Furthermore, a VRE is often implemented as an integrated environment that provides users with pre-configured sources of data and software tools and functional components for managing research activities. While providing many benefits to the scientific community, the adoption of such integrated VREs is often hampered by the high time investment for learning the new technologies and incompatible experiences, for example, managing scientific workflows.

A notebook environment, such as Jupyter[¶] (in contrast to a VRE), allows researchers to more quickly and effectively initiate their research activities, such as implement the experimental logic using scripting languages (such as Python, R, and Julia) to document and share the experiments with necessary inputs/outputs and parameters as self-contained documents (namely notebooks).[7] Compared to integrated VREs that require online access control (e.g., D4Science) or heavy client-side software on the local machine (e.g., for workflow management), notebook environments such as Jupyter demonstrate their advantages of light, portable, and easy to use. As such, it remains presently the most often used method for performing research, even though the approach faces challenges of utilizing remote infrastructure[8] and restricts collaborative research designs inherent to a VRE. There is a clear need for improving the Jupyter environment to meet the requirements for open sciences. In this article, we first review the current VRE support of the Jupyter environment and identify the gaps. After that, we present a lightweight solution to extend a Jupyter notebook environment as a collaborative VRE, which provides features for effectively managing research resources, remote cloud automation, and workflows. Finally, we demonstrate the key features of our solution, using a use case that bridges ecology and remote sensing, that is, processing multi-terabyte LiDAR point clouds from national airborne laser scanning (ALS) surveys to derive high-resolution geospatial layers of the ecosystem structure over broad spatial extents.

[*] https://vre4eic.ercim.eu/
[†] https://www.d4science.org
[‡] https://ever-est.eu/
[§] https://usegalaxy.org/
[¶] https://jupyter.org/

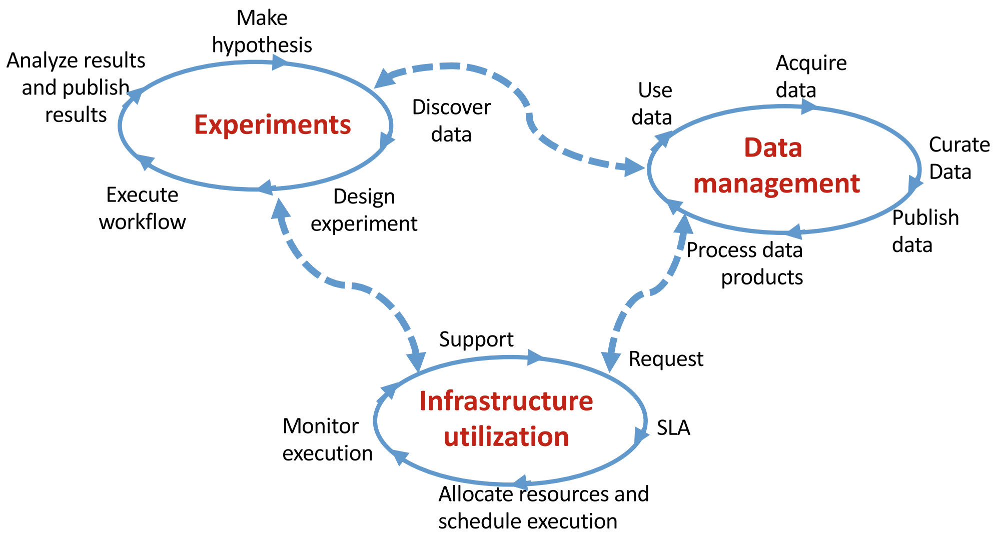


FIGURE 1 Different research activities that might be involved in data-intensive research

2 | PROBLEM DESCRIPTION AND RELATED WORK

Research activities in data-centric sciences are often a combination of many steps. We can often identify several closely related activities centered around domain-specific scientific research, data management, and infrastructure utilization for remote computation (Figure 1)[#].

For instance,

1. When doing **scientific research** on a specific problem, a scientist often starts with making a hypothesis for a scientific problem and then discovers and acquires data and software components that can be used to design an experiment or analysis for the study. After that, the scientist would process the data by executing a workflow with specific input data and configuration of parameters of the workflow steps (e.g., via algorithms or tools). When the data set is large, such an experiment or analysis often requires a large computing and storage capacity that remote computing facilities can only provide. After the execution, the scientist then analyzes and publishes the results. In general, this traditional scientific research is not a simple sequential flow but follows an iterative approach; for instance, a scientist can return to their original workflow design, execution, and analysis steps multiple times. Depending on specific practices, those activities can be structured in different orders or combinations, but the main components of this research life cycle remain the same.[9]
2. **Data management** involves a set of activities that make data a valuable resource for subsequent (re-)use. These activities are often conducted by dedicated data managers or support staff. For instance, they start by acquiring data from remotely deployed sensors, in-situ field measurements, observation networks, or other data infrastructures. Experts will curate those data via steps like quality control, metadata annotation, and archiving. Those curated data can then be published and made accessible to external users via metadata catalogs or data repositories. In this context, data can also be processed for different products (process data products) to serve dedicated users, for example, early warning platforms often need near-real-time data products from environmental observation networks to be quickly accessible in their big simulation platforms. Finally, there are services to make such data (either raw data or derived data products) adequate for new scientific experiments or analyses (use data). The activities in data management thus often differ among domains but can be generalized in typical steps of a data life cycle.[10] Those five basic activities illustrated here (Figure 1) are common ones we observed from research infrastructures in the environmental and earth science domains and modeled in the ENVRI reference model[‖].

[#]The lifecycle is derived from typical activities in workflow management, data management (based on the ENVRI reference model), and cloud resource management. There are undoubtedly other ways to describe them.

[‖]https://envri.eu/envri-rm-v2-0-released-today/

3. **Infrastructure utilization-related** activities can be typically seen in data centers or computing centers for scientific research. One essential issue in such environments is big data that every application encounters several performance challenges due to its unique context and requirements of data flow and processing.[11] The users who want to run scientific workflows on such infrastructures first have to request a certain capacity of resources from the infrastructure based on the requirement of the workflow (e.g., type of computation and volume of the data). If the requests can be handled, the infrastructure operator typically makes an agreement, also called service level agreement (SLA), with the user. After that, the infrastructure operator will allocate the resources and schedule the execution of the request on the infrastructure. During the execution, the operator will also actively monitor the status of the execution and provide support to the users when specific errors occur (e.g., failure of the workflow).

   The boundaries among those cycles, in many cases, are often blurred. For instance, domain researchers often spend lots of time on data acquisition, quality control, and processing during their daily activities, as ecologists observe. Data acquisition and curation are also seen as required steps in scientific research in the ecology domain. Moreover, some activities can be outsourced to third parties or largely automated, for example, infrastructure utilization or generating specific data products. Additionally, some activities are still challenging to support a single IT system because of their high dependencies on human actions. Nevertheless, we can see the complexity a scientist faces when managing IT-related activities in the research lifecycle; an effective research support system is needed.

2.1 | Virtual research environment and research lifecycle

To effectively support scientists to conduct those research activities, a VRE should provide tools and platforms to: manage research assets (data, software, and other types of objects): for example, hide the complexity of discovering research assets across different sources, provide interfaces to the other relevant infrastructures (e.g., research infrastructures in the ENVRI cluster[**]), and curate, publish, and share the newly created research assets. Manage scientific experiments: compose the logic (workflows) of the scientific experiments using available components, execute the workflow locally or using remote infrastructure, and analyze the final results. Manage collaboration with users from the community, for example, for sharing research assets and cooperative work on similar problems.

To enable those basic functionalities, a VRE needs to

1. Be flexible and efficient: the VRE should first allow users to conduct their research activities flexibly, preferably without changing much of their existing daily practices. It means the tools and functionality provided by the VRE should be customized to the specific domain and problems that researchers are working on.
2. Automate the standardized processes or the processes that scientists clearly understand but do not wish to iterate much time on, for example, data clearing pipelines and preprocessing tasks.
3. Being reproducible of the processes is crucial for the scientists to analyze the processes, particularly when an experiment is distributed on a large scale.
4. Provide trustworthiness on the quality and content of data products from an external source and on the services and model used for processing the data.

2.2 | Limits of the Jupyter environment

As an essential type of research asset, notebooks allow users to reproduce the experiments effectively and further develop them for new purposes. Still, there are several challenges to being an effective VRE for enabling open sciences and innovations across disciplines on a large scale:

1. Difficult to find and access a notebook at the granularity level of functions and components. Researchers often share their notebooks via version control systems, like Git; insufficient notebook metadata, particularly the monolithic nature of the notebooks, hampers the discovery of the latter, especially the components (namely cells) in the notebook.

[**]http://www.envri.eu

2. Lack of flexibility to reuse a notebook as part of a distributed workflow. When reusing an existing notebook, a researcher often needs part of the code fragments and integrates it into the other code to build logic for a new experiment. The often tightly coupled functions in the notebook, particularly the implicit dependencies on the libraries, make the reusability and extension of the code fragments difficult.
3. Difficult to scale the notebook to remote infrastructures. Current notebook environments, like Jupyter Hub[††], couple a pre-configured infrastructure (e.g., via Cloud IaaS) to dynamically load instances of a notebook and perform computing tasks.[12] When processing huge data volumes or computing complex tasks, dynamically allocated cloud resources other than that pre-configured capacity are often needed for parallelizing distributed computing tasks. Solutions to seamlessly automate the infrastructure services with scheduling tasks in the notebook are still lacking or not widely adopted.

2.3 | VRE development challenges

As a special kind of software, the commercial market of VREs is still relatively small; most of the developments still originate from the academic and research communities. This also brings lots of challenges to the sustainability of the VREs.

First, the development of the VRE interface is time-consuming and often customized to research assets in a specific domain or a specific type of computing infrastructure. It often creates gaps between the VRE and the daily practices of a scientific domain.

Second, the openness of the VRE is crucial for a scientist; not only a connection to the new infrastructure is needed for scaling experiments out, but also the connection to new communities for expanding collaboration to different scientific domains.

Moreover, the interface between VRE and the underlying research infrastructures (for accessing data, services, and other research assets) and computing infrastructure (for performing computing tasks) requires standardized metadata and APIs offered by those infrastructures. However, the low interoperability among current research infrastructures makes such interfaces non-trivial.

To tackle those challenges, we propose a VRE solution embedded in the daily practices that current scientists perform in their research activity and aim to expand their daily environment to a VRE with minimal effort. An embedded VRE solution called Notebook-as-a-VRE (NaaVRE) is proposed.

2.4 | Related work

During the past decades, many systems have been developed to support scientific research. Problem solving environments (PSE) are early examples proposed to support the computational research paradigm for complex problems. PSEs often provide simulation models or equation solvers, which allow researchers to construct computational solutions, for example, SciNaps[13] and NAREGI-PSE.[14] However, due to the limits of the early computing and software technologies, those PSEs are often developed as silos without extensive support for social interactions among researchers. Scientific workflow management systems are other examples of support systems; they aim to automate the execution of data pipelines or computing tasks across remotely distributed infrastructures.[15] VREs or Science Gateway, to a certain extent, can be seen as a new generation of those early systems with better support for data-intensive applications on remote infrastructures (e.g., cloud),[16] research activities across entire research lifecycles[‡‡] and collaboration and sharing within a community.[17] In many cases, the differences among those systems are not clearly cut. A VRE is often developed as a web-based online environment to support researchers to perform activities that they cannot do on their local computers. The development of functional components and user interfaces are heavily driven by the needs of the user community.[18] To improve the genericity of the online environment, many systems decouple the domain-specific working spaces (namely Virtual Labs) from the foundations of generic technology (also called VRE platforms), for example, D4Science hosts more than 150 virtual labs[§§].

[††]https://jupyter.org/hub
[‡‡]https://www.jisc.ac.uk/full-guide/implementing-a-virtual-research-environment-vre
[§§]https://www.d4science.org/

Many online VREs have a different user interface than the development environment that a researcher may use locally (e.g., R-Studio or Python). Seamless data exchange between online VREs and local computers is another gap that requires researchers to have a high learning curve to adapt their daily practices. In the meantime, the Jupyter environment provides an online interface, which can be deployed on either a local computer or a remote server. However, the native functionality of a Jupyter environment is currently limited to the backend engine, to the infrastructure where it is deployed, and to the pre-installed libraries. Its support for VRE features (as discussed in Section 2.1) is thus limited.

There are several Jupyter-based systems[19-21] aimed to improve the notebook environment for supporting scientific experiments. The Earth Observation Data Access Gateway (EODAG)[19] Jupyter extension[¶¶] enables end-users to query satellite imagery from various image providers, using the EODAG library via notebook environment. The query results can be transformed to code cells into the active Python notebook to process the dataset further. This extension focuses only on integrating a specific data resource with Jupyter. Binder[20] can automate the launch of a Jupyter notebook from a Git repository and thus simplify the reproduction of the Jupyter notebooks-based data experiments. As a highly interactive environment, cells in a notebook are often developed, tested, and executed in an arbitrary order by the user, resulting in less reusability and reproducibility of the notebook. Wang et al.[21] used code analysis techniques to analyze the dependencies among cells and to restore the reproducibility of the notebook. These existing examples address specific aspects of the scientific experiments, for example, data access, query, automation of the execution, or reproducibility of cells based on dependencies. These solutions do not aim to cover the entire lifecycle of an experiment from a VRE point of view.

3 | NOTEBOOK-AS-A-VRE (NAAVRE)

The Notebook-as-a-VRE (NaaVRE) solution focuses on building a VRE by extending the notebook environment of the end-users. Currently, we design it based on Jupyter,[7] a widely used tool for daily research activities in lots of data-centric sciences. In principle, other environments like R-Studio can also be considered. In Jupyter, popular programming languages used by data scientists including Python, R, Julia can be supported by specific execution kernels.

3.1 | Requirements

The proposed solution should meet the following requirements:

1. Should be integrated into the Jupyter environment, without forcing users to change their basic practices on conducting experiments.
2. Seamless support for adding VRE features, for example, publishing/sharing components with the community, running tasks on remote cloud infrastructure, and search tools for new data sources.
3. Open for new functional components. The NaaVRE should further be exploited as an open ecosystem to include new community contributed components.
4. The solution should be highly customizable and allow a user to reconfigure the environment based on the needs.
5. The solution should be efficient and fault-tolerant; users in the NaaVRE ecosystem are connected via a decentralized framework for sharing data and other research assets.

3.2 | System architecture

Based on the requirements, we design the NaaVRE as an ecosystem, which provides required VRE functional components as scalable services (e.g., microservices) and can be deployed as Jupyter extensions on the user environment (Figure 2).

The NaaVRE provides the functional components (services) via a repository (also called marketplace), allowing users to browse and select functional components to make a customized VRE experience. Typical functional components include:

[¶¶]https://github.com/CS-SI/eodag-labextension

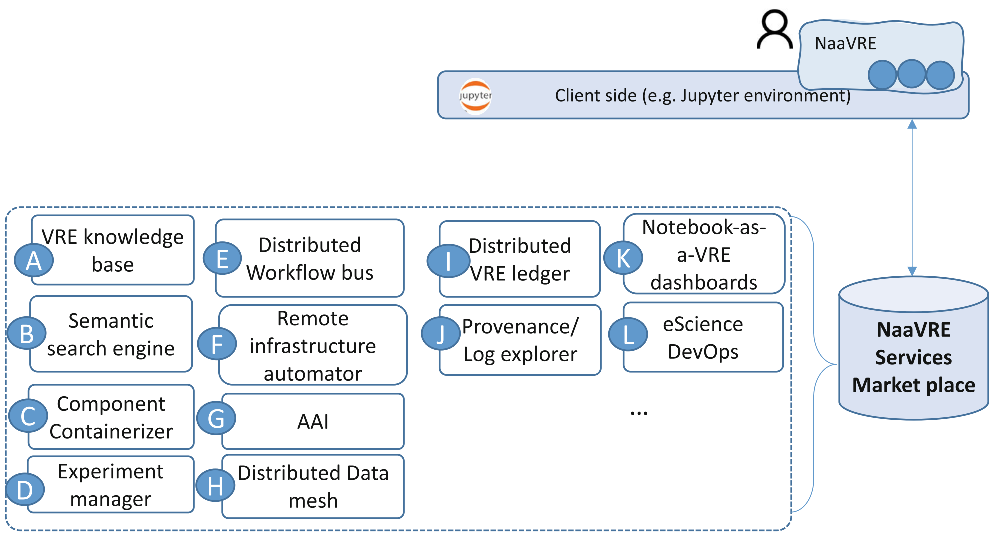

NaaVRE
Client side (e.g. Jupyter environment)
jupyter
A VRE knowledge base
B Semantic search engine
C Component Containerizer
D Experiment manager
E Distributed Workflow bus
F Remote infrastructure automator
G AAI
H Distributed Data mesh
I Distributed VRE ledger
J Provenance/ Log explorer
K Notebook-as-a-VRE dashboards
L eScience DevOps
...
NaaVRE Services Market place

FIGURE 2 NaaVRE and the Jupyter client

A **VRE knowledge base** with published research assets (data sets, software components, and tools) produced and shared by the VRE community.

B **The semantic search engine** provides a web interface to search external catalogs (e.g., for data sets, services, or other assets), and also API for the application to invoke via Jupyter.

C **Component containerizer** as a tool enabling users to select code fragments (cells) from the notebook effectively, generate reusable workflow building blocks (e.g., RESTful services), and encapsulate the building blocks as deployable containers (e.g., Docker).

D **Experiment manager** allows users to design experiments by selecting building blocks from the VRE knowledge base, composing the logical relations and dependencies among components, configuring the parameters, input, and output, executing the experiment via workflow bus, and monitoring the progress of the experiment.

E **Distributed workflow bus** uses a software bus concept[15] to manage runtime infrastructures for automating the deployment of workflow instances (namely experiments), scheduling the execution of each experiment, and managing information of different workflow instances over the distributed ledger and the distributed data mesh. The distributed workflow bus enables collaboration among workflows from different users, for example, publishing/subscribing experiment results and reproducing a specific result.

F **Remote infrastructure automator** that automates the execution of a workflow on the remote infrastructure, including (a) planning the cloud infrastructure (networked virtual machines) based on workflow description, providers, and execution requirements, (b) automates the provision of virtual machines, the configuration of Kubernetes clusters, and (c) schedules the execution of the workflow.

G **Authentication and authorization infrastructure (AAI)**[22] that enables the NaaVRE to (1) authenticate users and manages authentications when the users need to access remote infrastructures (e.g., cloud or HPC), and to (2) authorize users to access specific services and research assets of the VRE.

H **The distributed data mesh** provides an interface for applications to share data content in remote infrastructure, for example, via centralized storage or a peer-to-peer file system. It provides a transparent layer for workflow applications to exchange data.

I **A distributed VRE ledger** that provides an immutable logging and ledger management framework for NaaVRE users to publish assets and trace the evolution of digital objects. We assume user communities are organized as a consortium of institutions or individuals. The VRE ledger is thus based on a consortium blockchain environment.

J **Provenance and logs explorer** that allows the users to navigate the provenance information and system logs, analyze the specific events, and reproduce a workflow when necessary.

K **The NaaVRE dashboard** will be the interface to configure the components visualize the status of VRE components, for example, the progress of the workflow execution and utilization of the remote infrastructures.

L **eScience DevOps** (development and operation) provides an automated pipeline to handle the lifecycle of the scientific code from code in the notebook cell to a self-contained and independent cloud computational entity.

Those components aim to support scientific activities identified in Figure 1.

As shown in Figure 3, a domain researcher can use the NaaVRE knowledge base (A) to check existing solutions or similar problems in the community and search (B) existing data, software, and other relevant research assets for tackling the problem. The user can work on the experiment logic using the Jupyter environment or load existing code. The user can create reusable components by containerizing them as Dockers (C), assembling them as workflow (D), and configuring the input data to create an experiment. The experiment will be executed by the workflow bus (E) in several steps, including planning infrastructure based on the workflow, automating the infrastructure services, and deploying the workflow component (F). In this context, the NaaVRE will automate accessing different infrastructures (including data repositories and cloud) using (G). The NaaVRE will also customize the data mesh for provisioning data for the execution (H). The runtime processes will be logged, and key processes will be tracked and recorded via the distributed ledger system (I). The user can analyze the experiment results using provenance and data analytics tools (J). The entire processes of the research activities will be presented via a dashboard (K) and automated using an eScience DevOps pipeline where needed (L).

The NaaVRE framework relies on external catalogs to access research assists from specific communities, for example, the LifeWatch catalog for the ecology community and the ENVRI-Hub catalog for the ENVRI community. The NaaVRE also relies on the external infrastructures to execute the remote workflows, including e-Infrastructure, for example, EGI and EOSC, and cloud environments, for example, Azure and AWS.

3.3 | How can NaaVRE be operated?

The NaaVRE can be seen both as a framework of the VRE technologies and as a community to enable users to collaborate via the Jupyter environment (Figure 4).

An organization can operate the NaaVRE as a Platform-as-a-Service (PaaS). Through the PaaS:

1. One (e.g., a Virtual Lab administrator) can create dedicated virtual labs (step 1) for a group of users, for example, based on specific domain problems.

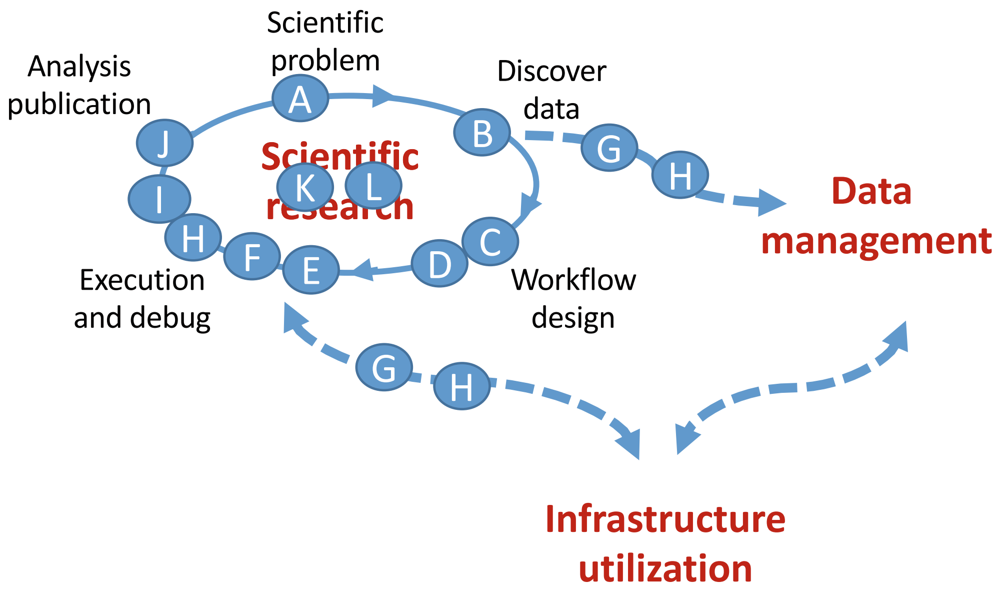


FIGURE 3 How NaaVRE supports scientific activities. The data management and infrastructure utilization parts are the same as Figure 1.

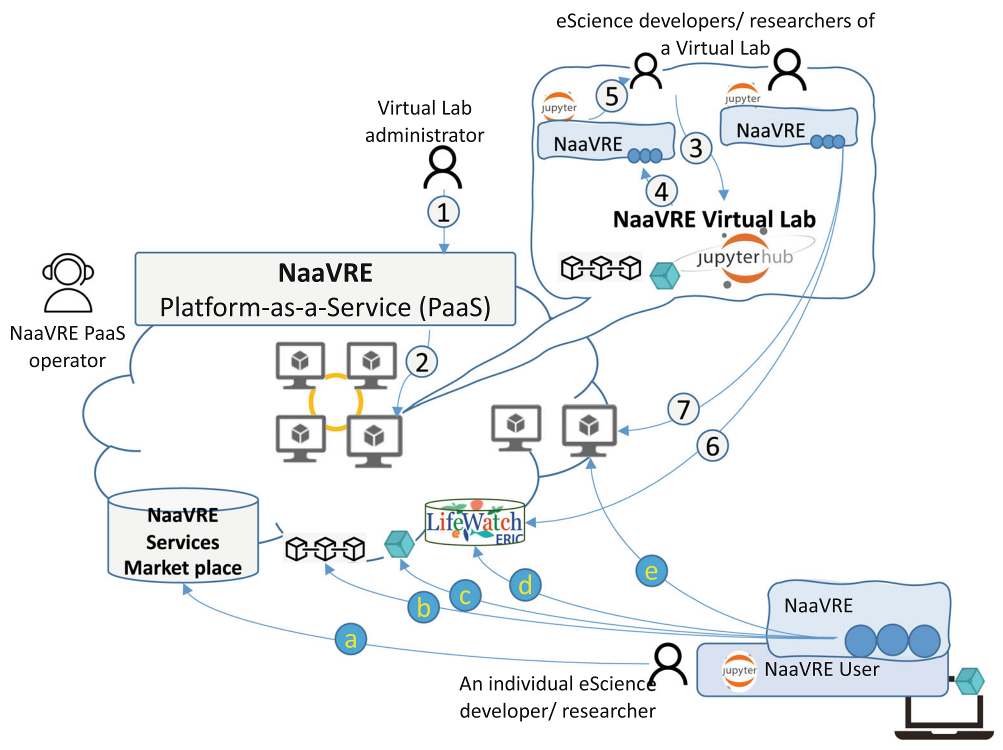

eScience developers/ researchers of a Virtual Lab
Virtual Lab administrator
NaaVRE
NaaVRE
NaaVRE Virtual Lab
jupyterhub
NaaVRE Platform-as-a-Service (PaaS)
NaaVRE PaaS operator
NaaVRE Services Market place
LifeWatch ERIC
NaaVRE
NaaVRE User
An individual eScience developer/ researcher
1
2
3
4
5
6
7
a
b
c
d
e

FIGURE 4 The distributed research community is enabled by NaaVRE.

2. The NaaVRE PaaS will instantiate a Virtual Lab (VL) based on required resource capacity and configurations of the NaaVRE components and deploy the VL in the infrastructure (step 2). A VL is typically based on the Jupyter Hub environment, with NaaVRE components installed.
3. VL users, for example, eScience developers or researchers, will create their Jupyter environment via the Virtual Lab (step 3 and 4) and perform the research activities (step5)
4. Via the NaaVRE functional components, VL users can access the customized catalog (step 6) create dynamic resources (step 7) for running workflows.
5. The VL will initialize a blockchain node and P2P file node needed by distributed ledger and data mesh.

An individual user can also customize their Jupyter environment as private VL using NaaVRE technology (see the bottom part of Figure 4)

1. A NaaVRE user can select relevant services from the NaaVRE marketplace and install them in the local environment (step a).
2. The users will also register themselves in the NaaVRE distributed ledger system (i.e., blockchain) to access the ledgers (step b).
3. The user will install a Peer 2 Peer file node on the local environment to connect with the NaaVRE distributed data mesh. The shared layer can also be used for sharing data objects at the runtime of the workflow (step c).
4. Using the NaaVRE components, the user can then search assets (data or services) from the external catalog (e.g., from LifeWatch or ENVRI) to design an experiment (step d).
5. The code prototyped by the user in Jupyter will be executed on remote infrastructure via the distributed workflow bus by the experiment manager (step e).

4 | IMPLEMENTATION AND CURRENT STATUS

The development of the NaaVRE follows the current industrial and community standards in cloud computing and scientific workflow management to develop the functional components.

4.1 | Development methodology

Some of the components are joint with the other projects, including SWITCH[##], VRE4EIC, ARTICONF[‖‖], ENVRI-FAIR[***], and BlueCloud[†††]. The development also follows the iterative and agile development software engineering practice, users are engaged in the development with different case studies. The technical development team closely works with the domain scientists (the last three co-authors): discuss the use cases, identify user stories, define implementation plan, select technologies, test implementation via use cases and continue.

4.2 | Current prototype

The NaaVRE is developed as an open-source project[‡‡‡]. Currently, the following components have been developed:

A **VRE knowledge base** is developed jointly with the ENVRI-FAIR project.[23] It is operated as a community service for the users of NaaVRE. The documents and metadata of the research assets will be indexed and captured in the knowledge base.

B **VRE search engine** is developed using information retrieval techniques.[6,24] It provides an interface for VRE users to search assets (e.g., data sets, Web API, or notebooks) from the community and select relevant ones in a basket. Additionally, the VRE search engine employs other information retrieval and extraction methods, such as textual content analysis such as topical coding, pattern clustering, and other topic models[25] to support the community. The search engine provides an API for the Jupyter users to retrieve research assets from the basket in the cells of a notebook (Figure 5).

C **Component containerizer**, also known as FAIR-Cells extension,[8] is an interactive tool for a researcher to (1) interactively encapsulate selected cells of a notebook as RESTful services based on input and output variables of the cells, (2) dockerize the services with customized base images (via Docker file), (3) publish the products (Docker images) in either a centralized way, namely on a remote repository (currently Docker Hub) and a catalog (e.g., LifeWatch catalog), or a decentralized way, namely using blockchain (for metadata) and IPFS (for dock images). In this way, the notebook's fragments can reach better findability and accessibility towards the FAIR digital objects. Figure 6 shows the screen snapshot of the FAIR-Cells extension. On the right side, the user can edit and select the relevant code fragments, and the tool will automatically capture the input and output of the cell and visualize the component on the left side. From the left side panel, the user can generate the RESTful representation of the code in the cell and dockerize it.

D **Experiment manager**. The NaaVRE provides an effective way to allow users to describe the logic of the experiments (Figure 7). The experiments can be based on the components created using the containerizer tool. We use Common Workflow Language syntax to describe the workflow logic, which can be mapped onto different flow languages executed by a specific engine, for example, ARGO flow[§§§] or TOSCA[¶¶¶]. Using the experiment manager, a user can use two built-in components "split" and "merge" to parallelize the execution of certain cells on large data collection, as shown in Figure 7.

E **Distributed workflow bus** is currently prototyped using ARGO framework. It interacts with the infrastructure automator to (1) prepare the cloud services and (2) automate the deployment and execution of an experiment. It also registers metadata and runtime information of the experiments on the distributed ledgers and enables collaborations among VRE users. Figure 8 shows a screen snapshot of the experiment information in the current workflow bus. When the workflow contains "split" and "merge" operations, the workflow bus will dynamically create instances of the required workflow component based on provided data sources and collect the output to a given data destination (in Figure 8).

F **Remote infrastructure automator**, also called the cloud-cells extension, automates cloud infrastructure provisioning and the deployment of Docker images on the cloud infrastructure. Cloud-cells uses SDIA (the Software Defined

[##]http://switch-project.eu/

[‖‖]https://articonf.eu/

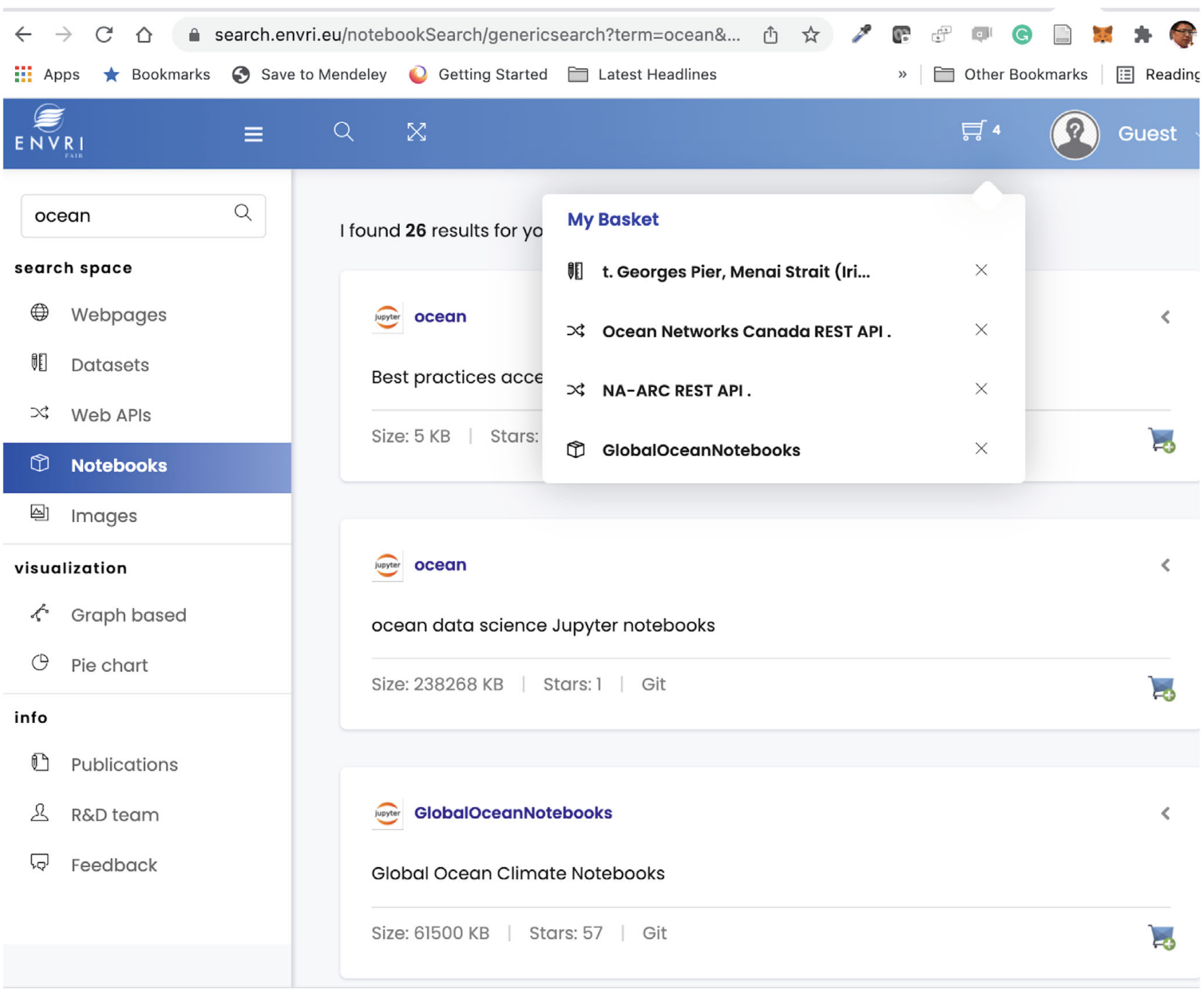
[***]http://envri.eu/envri-fair/

[†††]https://www.blue-cloud.org/

[‡‡‡]https://github.com/QCDIS/NaaVRE

[§§§]https://argoproj.github.io/

[¶¶¶]https://www.oasis-open.org/committees/tosca



I found 26 results for yo

ocean
Best practices acce
Size: 5 KB | Stars:

ocean
ocean data science Jupyter notebooks
Size: 238268 KB | Stars: 1 | Git

GlobalOceanNotebooks
Global Ocean Climate Notebooks
Size: 61500 KB | Stars: 57 | Git

FIGURE 5 The search engine prototyped using ElasticSearch.

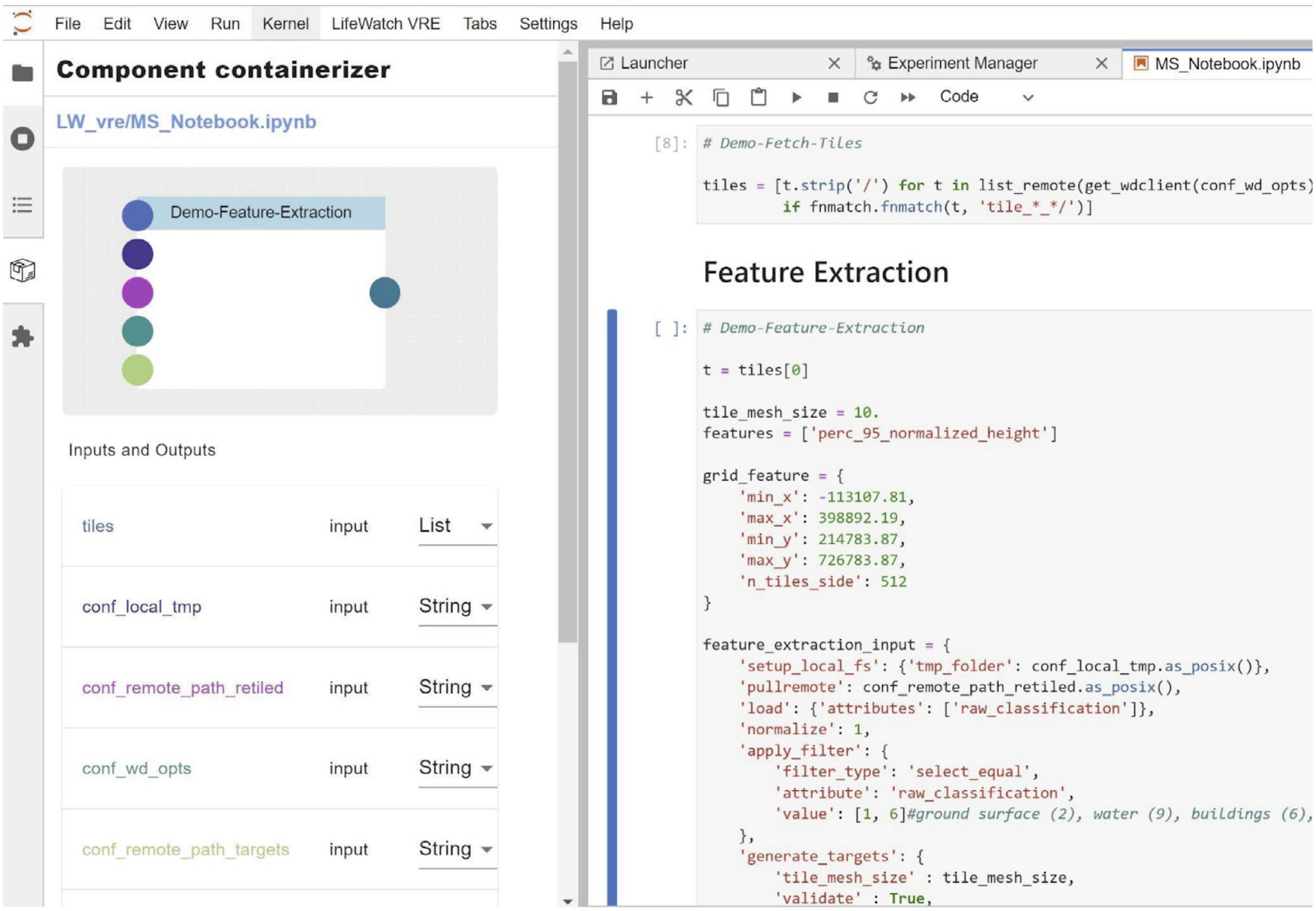
Infrastructure Automator), a framework developed by the same team for planning, provisioning, and deploying applications on multi-cloud environments[26] (Figure 9). A VRE can have a number of pre-provisioned virtual infrastructure (namely networked virtual machines); it can dynamically request and provision virtual infrastructure requested by a user for running specific workflows. A user should provide his/he own credential to obtain the dynamic resources. A dynamic infrastructure can be released after the workflow is finished.

G **Authentication and authorization infrastructure (AAI)** currently follows the best practices and guidelines from the initiatives like EOSC, ENVRI, and LifeWatch. It currently supports the public identity providers like GitHub. Virtual Lab administrators can log in to the NaaVRE platform using the identities.

H **The distributed data mesh** currently supports both centralized (WebDAV[###]) and decentralized (IPFS[‖‖‖]) paradigms.

I **The distributed VRE ledger** currently uses HyperLedger Fabric[****]. Instead of using a public blockchain, we design the ledger of a community as a consortium of a set of trusted organizations. A new organization of the NaaVRE can set up a new node and join the network of the VRE ledger. Individual users can join the network using the identity of their organization. Other identity providers will also be supported in the future.

[###] http://www.webdav.org/
[‖‖‖] https://ipfs.io/
[****] https://www.hyperledger.org/use/fabric



Inputs and Outputs

tiles	input	List
conf_local_tmp	input	String
conf_remote_path_retiled	input	String
conf_wd_opts	input	String
conf_remote_path_targets	input	String

```
[8]: # Demo-Fetch-Tiles

     tiles = [t.strip('/') for t in list_remote(get_wdclient(conf_wd_opts)
              if fnmatch.fnmatch(t, 'tile_*_*/')]
```

```
[ ]: # Demo-Feature-Extraction

     t = tiles[0]

     tile_mesh_size = 10.
     features = ['perc_95_normalized_height']

     grid_feature = {
         'min_x': -113107.81,
         'max_x': 398892.19,
         'min_y': 214783.87,
         'max_y': 726783.87,
         'n_tiles_side': 512
     }

     feature_extraction_input = {
         'setup_local_fs': {'tmp_folder': conf_local_tmp.as_posix()},
         'pullremote': conf_remote_path_retiled.as_posix(),
         'load': {'attributes': ['raw_classification']},
         'normalize': 1,
         'apply_filter': {
             'filter_type': 'select_equal',
             'attribute': 'raw_classification',
             'value': [1, 6]#ground surface (2), water (9), buildings (6),
         },
         'generate_targets': {
             'tile_mesh_size' : tile_mesh_size,
             'validate' : True,
```

FIGURE 6 Screen snapshot of the component containerizer extension.

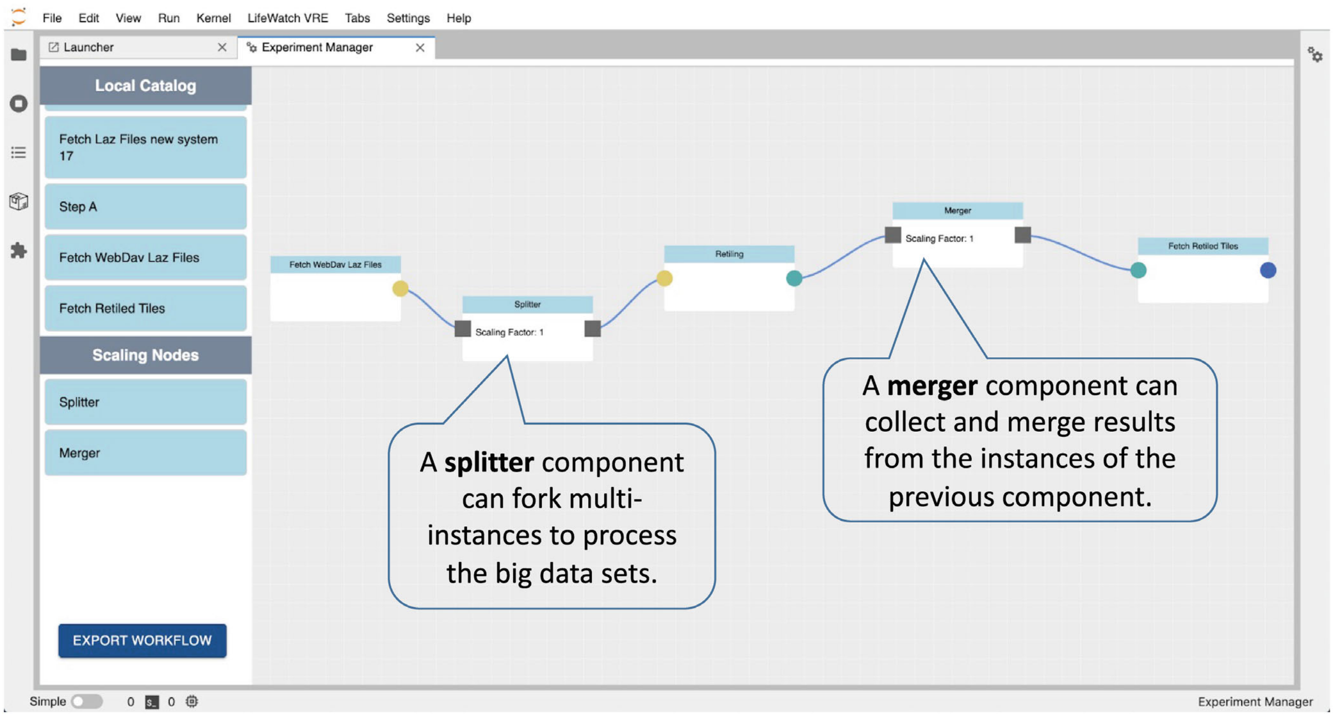


FIGURE 7 Screen snapshot of the workflow composer in the manager component.

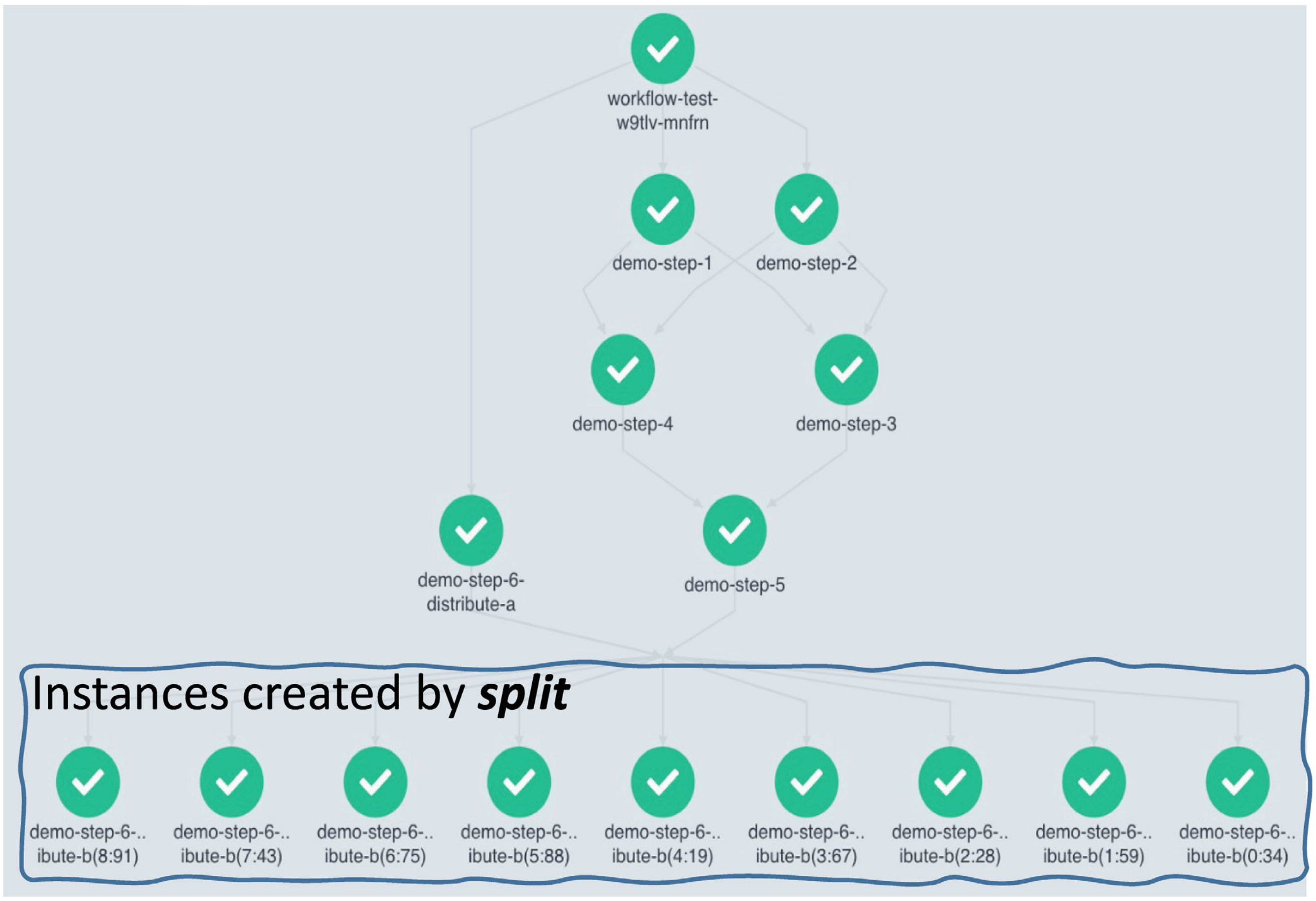


FIGURE 8 The runtime information of the distributed workflow bus

J **Provenance explorer and analysis**. The tool provides an interface for users to interactively explore the system logs and the provenance of the workflow and to identify the anomalies and reproduce the workflows or problems.[27] The provenance of the workflow is captured using PROV-O[††††]. Via the explorer, researchers will be able to interactively explore the provenance (see the upper part of Figure 10) of the workflow among cells (created using experiment manager) together with the workflow processes and system logs (see the middle and bottom part of Figure 10), and identify the anomalies during the execution.

K **The NaaVRE dashboard** visualizes the runtime status of the scientific experiments, infrastructure status, and distributed ledgers; currently, the ARGO framework is used for prototyping the dashboard.

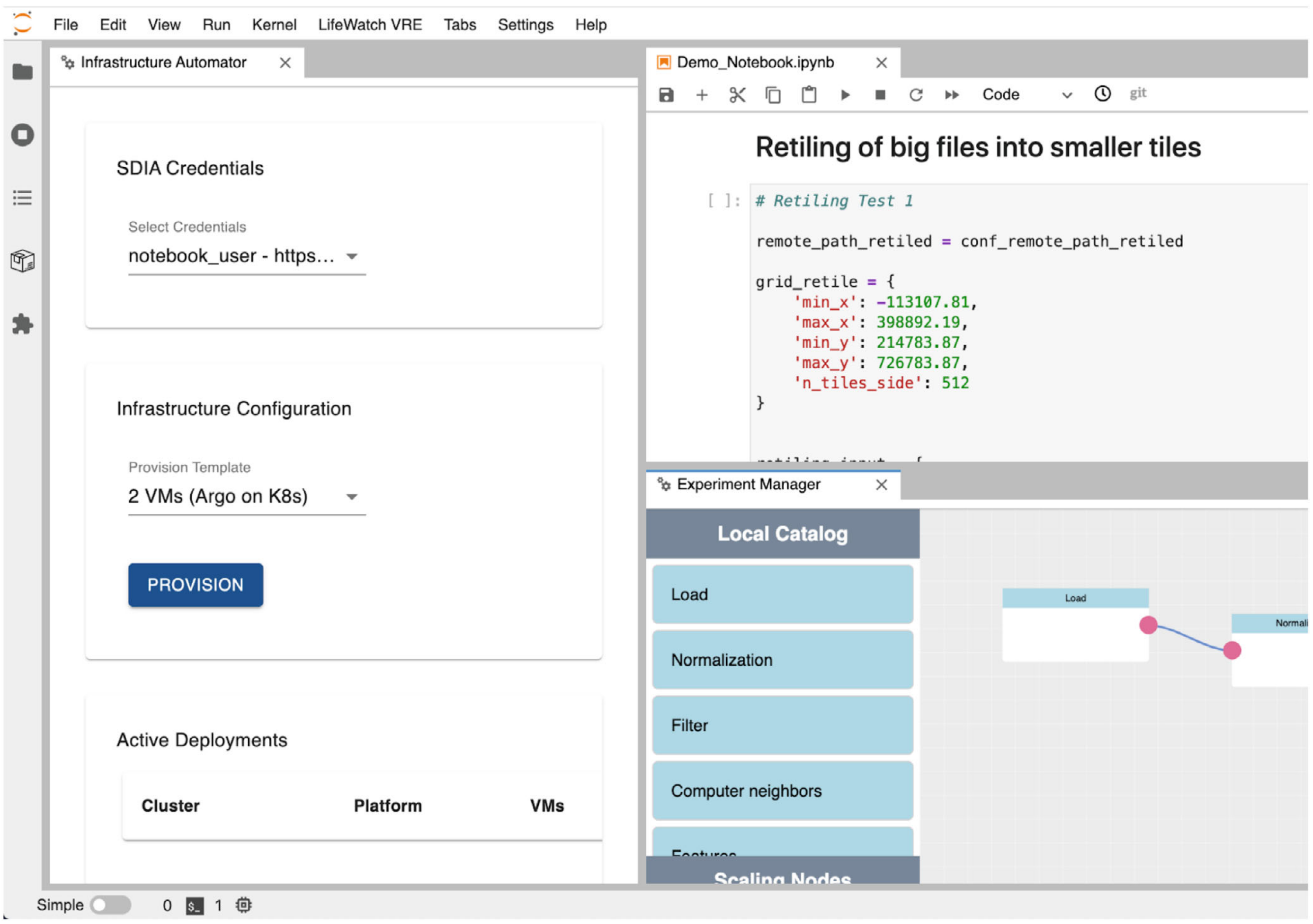
L **eScience DevOps** applies DevOps practices, including continuous integration (CI) and continuous deployment (CD) to automate the lifecycle of the experiments in the VRE. The NaaVRE uses Git to manage versions of the source of the cell and DockerHub or IPFS to store the images of containerized cells. The containerization and orchestration of the cells in a workflow on remote cloud infrastructure can be automated via the eScience DevOps pipeline when new changes are made on the source.

5 | CASE STUDIES: LIDAR VIRTUAL LAB

ALS[‡‡‡‡] data derived from Light Detection and Ranging (LiDAR) technology allow the construction of essential biodiversity variables (EBVs) of ecosystem structure with high resolution at the landscape, national and regional scale.[28]

[††††]https://www.w3.org/TR/prov-o/

[‡‡‡‡]The development of the use case is partially supported by the EOSC ENVRI-FAIR early adopter program (EAP) project, and the Azure AI on earth project.



Retiling of big files into smaller tiles

```
[ ]: # Retiling Test 1

     remote_path_retiled = conf_remote_path_retiled

     grid_retile = {
         'min_x': -113107.81,
         'max_x': 398892.19,
         'min_y': 214783.87,
         'max_y': 726783.87,
         'n_tiles_side': 512
     }
```

FIGURE 9 The screen snapshot of the infrastructure automator and experiment manager.

Researchers nowadays often process LiDAR data for various ecological applications and rapidly prototype using script languages like R or python and share their experiments via scripts or, more recently, via notebook environments, such as Jupyter.

A legacy program called "Laserchicken"[29] was developed in a previous project to process country-wide LiDAR point clouds in a local environment (e.g., the Dutch national ICT infrastructure called SURF). It is a Python tool that focuses on extracting features (i.e., statistical properties) of user-defined subsets of point cloud data.[29] The basic workflow of Laserchicken is shown in Figure 11. It consists of four core modules (load, compute neighbors, features, export) (as shown in Figure 11 in blue color), and two optional modules (filter, normalize) (as shown in Figure 11 in grey color), which provide a processing workflow for feature extraction from LiDAR point clouds, with file-based input and output.

However, a user currently can only use the Laserchicken application either on the dedicated infrastructure or on the local machine to process the LiDAR data. Running a specific part of the code on remote infrastructure is often challenging for domain scientists. The local machine's capacity or a given infrastructure also limits the volume of data. This use case (LidarAirCloud) demonstrates how the NaaVRE solution enables scientists to scale experiments to large data volumes, different data sources, or new models. Using the NaaVRE, the user first needs to set up the environment, for example, selecting the following components: experiment manager, component containerizer, distributed workflow bus, remote infrastructure automator, distributed data mesh, and distributed ledger as shown in the upper part of Figure 12. Since not all the components have been integrated, only a small subset has been used.

In this simple example, the user does not have to search for data from an extensive collection of the data sources; data sets from Table 1 have been considered, and country-wide LiDAR data from the Netherlands have been used below[§§§§].

[§§§§] https://www.ahn.nl/ahn-viewer

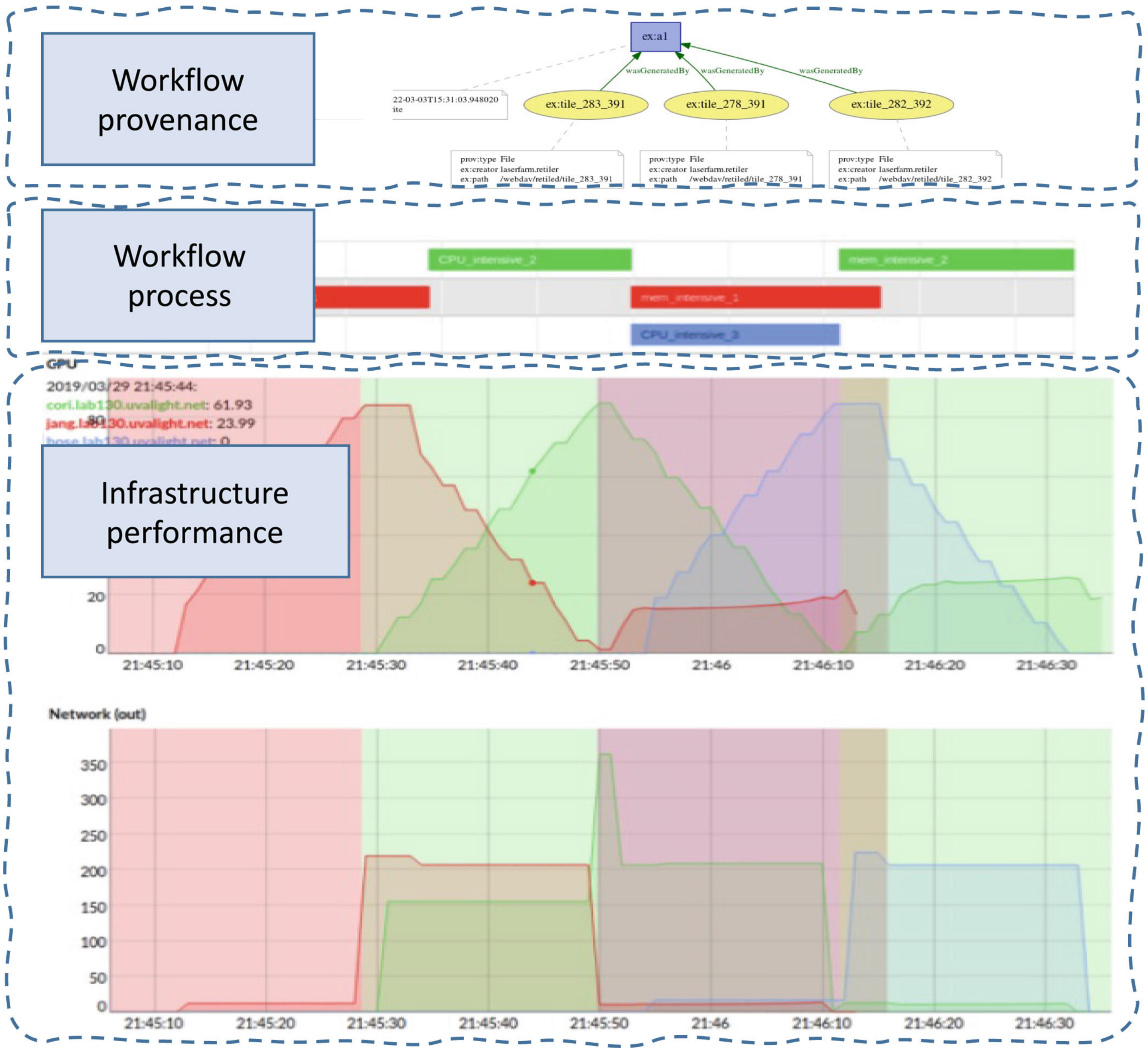


FIGURE 10 The provenance explorer and analysis interface. A user can visualize the provenance graph, workflow process, and infrastructure performance logs.

The sequences under the NaaVRE components in Figure 11 demonstrate the basic steps in the use case.

1. After preparing the LiDAR point clouds that need to be processed, the user can use experiment manager to design an experiment based on the "Laserchicken" workflow in the Jupyter environment. In this use case, the workflow was designed as the following steps: first, loading the raw LiDAR data (LAS files) covering a sample area in the Netherlands; second, normalizing the height of the point cloud; third, computing neighbors with a target volume of an infinite cylinder at 1-m radius; fourth, extracting selected features (e.g., 95th percentile of normalized height, coefficient of variance of normalized height); and finally, exporting the enhanced point cloud with added features (PLY files).
2. Via the experiment manager, the user will need to containerize the code as portable containers using the **containerizer** component provided in the Jupyter environment. Each step of the user's workflow described above can be containerized (and then published) as a container, providing a specific function of processing LiDAR data. This and

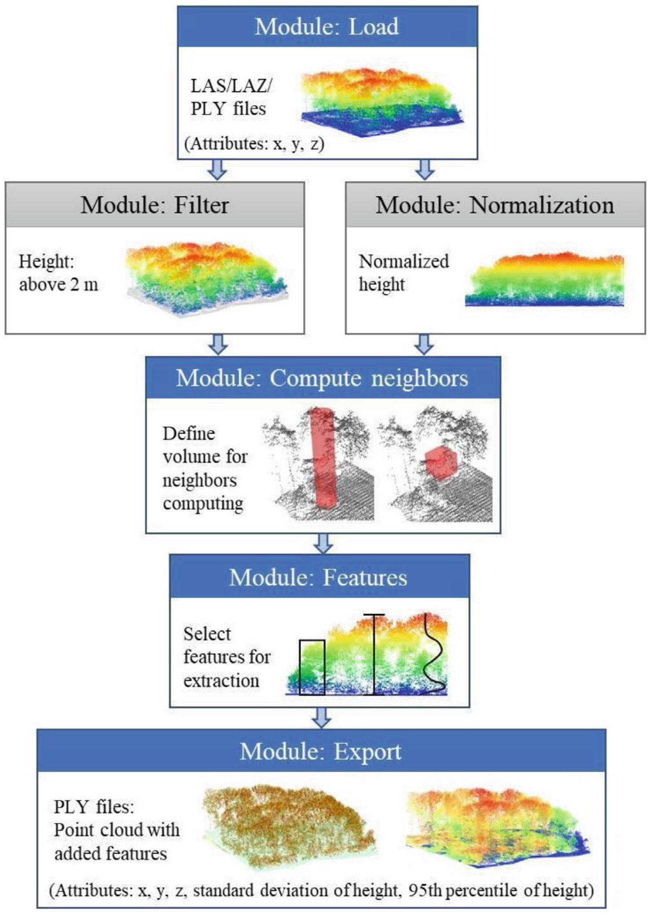

Module: Load
LAS/LAZ/ PLY files
(Attributes: x, y, z)
Module: Filter
Height: above 2 m
Module: Normalization
Normalized height
Module: Compute neighbors
Define volume for neighbors computing
Module: Features
Select features for extraction
Module: Export
PLY files: Point cloud with added features
(Attributes: x, y, z, standard deviation of height, 95th percentile of height)

FIGURE 11 Laserchicken application for processing LiDAR point clouds

TABLE 1 Details of LiDAR point clouds from example countries in Europe.

Country	Multi-year	Point density	Data volume
Finland	Partly	1–2 pt/m^2	4 TB
Netherlands	Yes	0.1–20 pt/m^2	16 TB
Spain	Partly	0.5–2 pt/m^2	5 TB
Denmark	Yes	4–5 pt/m^2	8 TB
Estonia	Yes	0.2–18 pt/m^2	-
Slovenia	Yes	2–5pt/m^2	2.5 TB
UK	Partly	0.5-16 pt/m^2	45 TB

Note: Data volume (compressed) represents how much data storage is needed based on the number of files available in each download portal and the average size of each file. The data used in this article is from the Netherlands (AHN3 dataset).

the previous step can be interactive until a concrete experiment is composed. An experiment can be seen as a concrete workflow with configurations on input data and other necessary parameters.

3. After the experiment is ready the user will submit the experiment to **the distributed workflow bus** for execution.
4. **The distributed workflow bus** will then plan cloud services (e.g., virtual machines and their topology) for running the workflow by using the **remote infrastructure automator**.
5. The **distributed workflow bus** will configure the **distributed data mesh** needed by the experiment, for example, centralized (based on WebDav) or decentralized (based on IPFS) option.
6. The **distributed workflow bus** will also register the experiment in the**VRE ledger**.

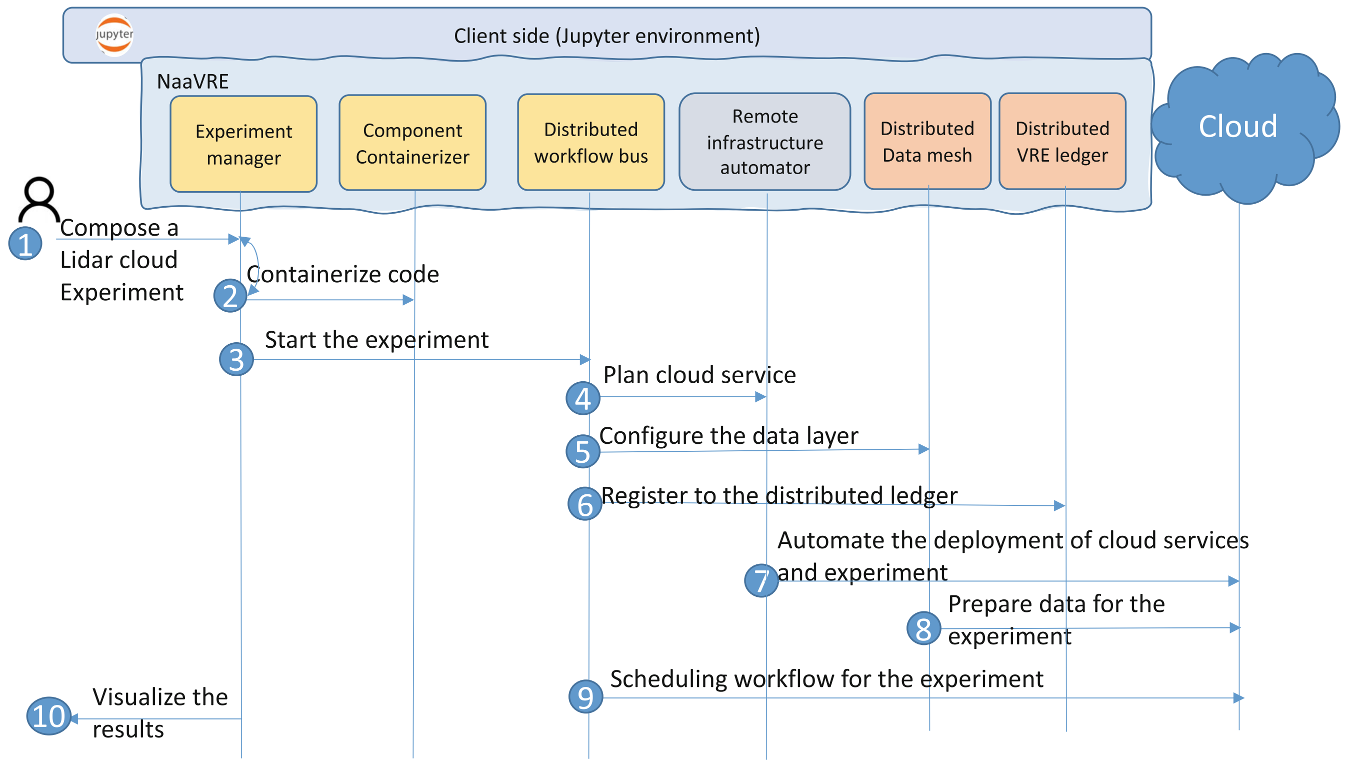

jupyter
Client side (Jupyter environment)
NaaVRE
Experiment manager
Component Containerizer
Distributed workflow bus
Remote infrastructure automator
Distributed Data mesh
Distributed VRE ledger
Cloud
1 Compose a Lidar cloud Experiment
2 Containerize code
3 Start the experiment
4 Plan cloud service
5 Configure the data layer
6 Register to the distributed ledger
7 Automate the deployment of cloud services and experiment
8 Prepare data for the experiment
9 Scheduling workflow for the experiment
10 Visualize the results

FIGURE 12 The workflow of the use case

7. The **remote infrastructure automator** will automate the provision of the cloud services on selected providers. At this stage, the component of AAI will be used.
8. The **distributed data mesh** will also provide the data sets needed by the workflow. In this use case, the pre-designed workflow can be upscaled and applied to other country-wide LiDAR datasets (with multi-terabyte data volume), as shown in Table 1. Ecosystem structure products covering whole countries or regions can be generated and used for further analysis.
9. The **distributed workflow bus** will schedule the workflow execution after components are being deployed (in step 4) and data is being prepared (in step 6).
10. After the execution, the user can visualize the results via the notebook. Figure 13 shows an example visualization of generated data products of the Netherlands[¶¶¶¶] from the Laserchicken workflow.

The initial version of the NaaVRE components has been tested on the cloud environment (offered by the EOSC early adopter program for ENVRI-FAIR, and Microsoft Azure through the AI for Earth program) to process a much bigger dataset than in a local environment.[30]

6 | DISCUSSION

In this use case, we demonstrate how a user can use NaaVRE to enhance the Jupyter lab to execute the legacy Laserchicken application in remote infrastructures. The article's target users are mainly developers or researchers using eScience approaches. They usually have interdisciplinary knowledge across scientific domains and computing techniques. They use the Jupyter environment for developing scientific code; the eScience researchers focus more on the scientific output, while the eScience developers are dedicated programmers for scientific code and often act as support staff for eScience or regular domain researchers.

¶¶¶¶https://doi.org/10.5281/zenodo.6364607

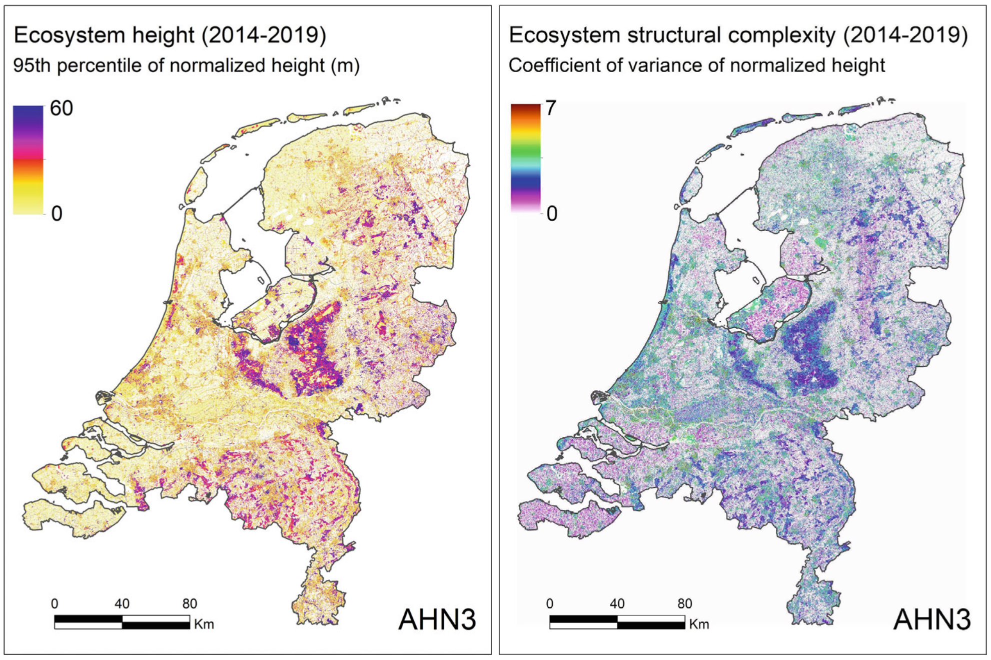

Ecosystem height (2014-2019)
95th percentile of normalized height (m)
60
0
0 40 80 Km
AHN3
Ecosystem structural complexity (2014-2019)
Coefficient of variance of normalized height
7
0
0 40 80 Km
AHN3

FIGURE 13 The visualization of two country-wide (10 m resolution) data products of ecosystem structure (height & structural complexity) of the Netherlands from applying the Laserchicken software. The LiDAR data were collected during the third Dutch national airborne laser scanning flight campaign (AHN3, Actueel Hoogtebestand Nederland).

The NaaVRE architecture aims to be open and flexible for other systems on workflow management. From the current prototype, we have specifically considered the following aspects.

Connect with other workflow management systems. NaaVRE aims to support other workflow systems besides the default one included in the marketplace as an open ecosystem. The NaaVRE can execute the workflows supported by an external workflow management system via the workflow bus, where the description will be wrapped and executed by invoking the external engine. The distributed workflow bus can dynamically launch the external engine when no such running engine instance is accessible. However, this integration also requires the external workflow management system to provide a backend engine with a well-defined interface for remote invocations.

On the other hand, the distributed workflow bus and the remote infrastructure automator are decoupled from the user interface of the Notebook. In principle, the backend can be integrated with the workflow composition interface of the third-party workflow management interface for designing and executing the workflow. However, this integration requires efforts to adapt the external composition interface to generate the required workflow description of the NaaVRE and invoke the workflow bus and remote infrastructure automator in NaaVRE. The above two options give opportunities for applying NaaVRE in the current service architecture in the LifeWatch ERIC, besides operating the NaaVRE as an online Platform-as-a-Service.

We envisage to expand the NaaVRE to other use cases in the ecology, biodiversity, and Earth science domain. For instance, application workflows in Jupyter notebooks can be developed for processing large volumes of satellite and airborne remote sensing data for generating EBVs.[31] This can contribute to developing a global observing system for biodiversity as envisaged by the Group on Earth Observations Biodiversity Observation Network (GEO BON).[32] It would further support newly emerging national and regional biodiversity observation networks such as those currently developed in Europe.[33]

7 | SUMMARY

The NaaVRE framework is jointly developed in the context of ENVRI-FAIR, LifeWatch, clarify, and several other projects. We closely interact with the users from ecology, marine, and other environmental and earth sciences domains. The framework is developed based on the Jupyter environment. The architecture aims to deliver an open ecosystem for the VRE components needed by the users. The framework is also going to be used as a framework for supporting distributed machine learning workflows in the medical domain[####], and for product generation of EBVs.

ACKNOWLEDGMENTS

This research was supported by the European Union's Horizon 2020 research and innovation program by the ENVRI-FAIR project Grant agreement No. 824068, by the BLUECLOUD project Grant agreement No. 862409, by the Life-Watch ERIC, by the ARTICONF project Grant agreement No. 825134, and by the project CLARIFY under the Marie Sklodowska-Curie Grant agreement No. 860627. The use case was also partially supported by the Earth Grant (AI4E-1111-Q3N7-20100806) from Microsoft, which provided access to the Azure Cloud in the context of the LidarAir-Cloud project. The Laserchicken software was developed in the eEcoLiDAR project funded by the Netherlands eScience Center (Grant number ASDI.2016.014).

AUTHOR CONTRIBUTIONS

All authors contributed to the conceptualization and writing of the paper; more specifically: Zhiming Zhao coordinated the development of the NaaVRE system, and led the writing and revision of the paper. Spiros Koulouzis developed NaaVRE components (C, F, G, H), and did system integration. Riccardo Bianchi developed NaaVRE components (D, E, K, L), and did system integration. Siamak Farshidi developed NaaVRE components (A, B). Zeshun Shi developed NaaVRE component (I). Ruyue Xin developed NaaVRE component (J). Yuandou Wang contributed to NaaVRE components (D, H). Na Li contributed to the NaaVRE components (A, B). Yifang Shi developed the use case and validated the NaaVRE system. Joris Timmermans validated the NaaVRE use case. W. Daniel Kissling coordinated the use case development and validation activities.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Data at https://www.ahn.nl/ahn-viewer, reference number Footnote z (page 16). The data product is also available via: https://doi.org/10.5281/zenodo.6364607.

ORCID

Zhiming Zhao https://orcid.org/0000-0002-6717-9418
Spiros Koulouzis https://orcid.org/0000-0001-8652-315X
Siamak Farshidi https://orcid.org/0000-0003-3270-4398
Zeshun Shi https://orcid.org/0000-0001-9163-8023
Ruyue Xin https://orcid.org/0000-0001-5821-4835
Yuandou Wang https://orcid.org/0000-0003-4694-9572
Na Li https://orcid.org/0000-0001-7799-876X
Yifang Shi https://orcid.org/0000-0001-8770-8906
Joris Timmermans https://orcid.org/0000-0003-0628-1803
W. Daniel Kissling https://orcid.org/0000-0002-7274-6755

REFERENCES

1. Vermeulen A, Glaves H, Pouliquen S, Kokkinaki A. Supporting cross-domain system-level environmental and earth science. In: Zhao Z, Hellström M, eds. *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences*. Series Title: Lecture Notes in Computer Science. Springer International Publishing; 2020:3-16.
2. Tuli S, Tuli S, Wander G, et al. Next generation technologies for smart healthcare: Challenges, vision, model, trends and future directions. *Internet Technol Lett*. 2020;3(2):e145.
3. Jeffery K, Pursula A, Zhao Z. ICT infrastructures for environmental and earth sciences. In: Zhao Z, Hellström M, eds. *Towards Interoperable Research Infrastructures for Environmental and Earth Sciences*. Series Title: Lecture Notes in Computer Science. Springer International Publishing; 2020:17-29.
4. Martin P, Remy L, Theodoridou M, Jeffery K, Zhao Z. Mapping heterogeneous research infrastructure metadata into a unified catalogue for use in a generic virtual research environment. *Future Gener Comput Syst*. 2019;101:1-13.
5. Miller MA, Pfeiffer W, Schwartz T. The CIPRES science gateway: enabling high-impact science for phylogenetics researchers with limited resources. Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the Extreme to the Campus and Beyond; 2012:1-8.

[####]http://www.clarify-project.eu

6. Kacsuk P. *Science Gateways for Distributed Computing Infrastructures*. Vol 10. Springer International Publishing; 2014:978-973.
7. Perkel JM. Why Jupyter is data scientists' computational notebook of choice. *Nature*. 2018;563(7732):145-147.
8. Kruijer W, Wang Y, Koulouzis S, Na L, Bianchi R, Zhao Z. FAIR-Cells: an interactive tool for enabling the FAIRness of code fragments in Jupyter notebooks. Proceedings of the International Conference of High-Performance Computing and Simulation (HPCS); 2020:1-8.
9. Tenopir C, Allard S, Douglass K, et al. Data sharing by scientists: practices and perceptions. *PloS One*. 2011;6(6):e21101.
10. Stodden V. The data science life cycle: a disciplined approach to advancing data science as a science. *Commun ACM*. 2020;63(7):58-66.
11. Alwasel K, Calheiros RN, Garg S, et al. BigDataSDNSim: a simulator for analyzing big data applications in software-defined cloud data centers. *Softw Pract Exp*. 2021;51(5):893-920.
12. Henderson ML, Krinsman W, Cholia S, Thomas R, Slaton T. Accelerating experimental science using Jupyter and NERSC HPC; 2019:145-163; Springer.
13. Akers RL, Kant E, Randall CJ, Steinberg S, Young RL. SciNapse: a problem-solving environment for partial differential equations. *IEEE Comput Sci Eng*. 1997;4(3):32-42. doi:10.1109/99.615429
14. Kanazawa H, Yamada M, Miyahara Y, Hayase Y, Kawata S, & Usami H. Problem solving environment based on grid services: NAREGI-PSE; 2005:456-463; IEEE, Pittsburg, PA.
15. Zhao Z, Belloum A, De Laat C, Adriaans P, Hertzberger B. Using jade agent framework to prototype an e-Science workflow bus. Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07); 2007:655-660.
16. Roth B, Hecht R, Volz B, Jablonski S. Towards a generic cloud-based virtual research environment; 2011:267-272; IEEE, Munich, Germany.
17. Roure DD, Goble C, Bhagat J, et al. myExperiment: defining the social virtual research environment; 2008:182-189; IEEE, Indianapolis, IN.
18. Calyam P, Wilkins-Diehr N, Miller M, et al. Measuring success for a future vision: Defining impact in science gateways/virtual research environments. *Concurr Comput Pract Exp*. 2021;33(19): 1-21. doi:10.1002/cpe.6099
19. Desconnets J-C, Giuliani G, Guigoz Y, et al. GEOCAB portal: a gateway for discovering and accessing capacity building resources in earth observation. *Int J Appl Earth Observ Geoinform*. 2017;54:95-104.
20. Ragan-Kelley B, Willing C. Binder 2.0-reproducible, interactive, sharable environments for science at scale. Proceedings of the 17th Python in Science Conference; 2018:113-120.
21. Wang J, Tzu-Yang KUO, Li L, Zeller A. Assessing and restoring reproducibility of Jupyter notebooks. Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE); 2020:138-149.
22. Paolini A, Scardaci D, Liampotis N, Spinoso V, Grenier B, Chen Y. Authentication, authorization, and accounting. Towards interoperable research infrastructures for environmental and earth sciences; 2020:247-271.
23. Zhao Z, Liao X, Martin P, et al. Knowledge-as-a-service: a community knowledge base for research infrastructures in environmental and earth sciences. Proceedings of the 2019 IEEE World Congress on Services (SERVICES); 2019:127-132.
24. Parodi A, Kranzlmüller D, Clematis A, et al. DRIHM (2US): an e-science environment for hydrometeorological research on high-impact weather events. *Bull Am Meteorol Soc*. 2017;98(10):2149-2166.
25. Hu G, Peng M, Zhang Y, Xie Q, Gao W, Yuan M. Unsupervised software repositories mining and its application to code search. *Softw Pract Exp*. 2020;50(3):299-322.
26. Koulouzis S, Martin P, Zhou H, et al. Time-critical data management in clouds: challenges and a dynamic real-time infrastructure planner (DRIP) solution. *Concurr Comput Pract Exp*. 2020;32(16):e5269.
27. Khaldi AE, Koulouzis S, Zhao Z. Contextual linking between workflow provenance and system performance logs. Proceedings of the 2019 15th International Conference on eScience (eScience); 2019:634-635.
28. Valbuena R, O'Connor B, Zellweger F, et al. Standardizing ecosystem morphological traits from 3D information sources. *TrendsEcol Evol*. 2020;35(8):656-667.
29. Meijer C, Grootes MW, Koma Z, et al. Laserchicken–A tool for distributed feature calculation from massive LiDAR point cloud datasets. *SoftwareX*. 2020;12:100626.
30. Koulouzis S, Shi Y, Wan Y, Bianchi R, Kissling D, Zhao Z. Enablingε LiDAR data processingε as a service in a Jupyter environment. Proceedings of the EGU General Assembly Conference Abstracts; 2021:EGU21-8294.
31. Skidmore AK, Coops NC, Neinavaz E, et al. Priority list of biodiversity metrics to observe from space. *Nature Ecol Evol*. 2021;5(7):896-906.
32. Navarro LM, Fernández N, Guerra C, et al. Monitoring biodiversity change through effective global coordination. *Current Opin Environ Sustain*. 2017;29:158-169.
33. Pereira HM, Junker J, Fernández N, et al. Europa biodiversity observation network: integrating data streams to support policy. ARPHA Preprints; Vol. 2, 2022:e81207.

How to cite this article: Zhao Z, Koulouzis S, Bianchi R, et al. Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment. *Softw Pract Exper*. 2022;52(9):1947-1966. doi: 10.1002/spe.3098