slices
DS

HORIZON 2020
H2020 - INFRADEV-2019-3

D2.4 SLICES as a Service, advanced

Acronym	SLICES-DS
Project Title	Scientific Large-scale Infrastructure for Computing/Communication Experimental Studies – Design Study
Grand Agreement	951850
Project Duration	24 Months (01/09/2020 – 31/08/2022)
Due Date	31 August 2022 (M24)
Submission Date	19 October 2022
Authors	Christian Perez (Inria), Walid Dabbous (Inria), Nathalie Mitton (Inria), Lucas Nussbaum (Inria), Adrien Lebre (IMT), Nikos Makris (UTH), Cedric Crettaz (MI), Bartosz Belter (PSNC), Carmen Guerrero (UC3M), Panayiotis Andreou (UCLAN), Nearchos Paspallis (UCLAN), Andreas Pamboris (UCLAN), Josephine Antoniou (UCLAN), Irene Polycarpou (UCLAN), Yuri Demchenko (UVA), Albert Su (SU).
Reviewers	All partners

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951850. The information, documentation and figures available in this deliverable, is written by the SLICES-DS project consortium and does not necessarily reflect the views of the European Commission. The European Commission is not responsible for any use that may be made of the information contained herein.

Scientific Large-scale Infrastructure for Computing Communication Experimental Studies **Design Study**

www.slices-ds.eu

Executive Summary

This document first provides an overview of the SLICES architecture and SLICES user-oriented services. Then, it focuses on a first analysis of the usage of such services with respect to three complex use cases. For each use case, a typical experiment is presented as well as the impact on SLICES services. It enables to confirm that identified SLICES services shall be able to cover complex use cases. It also enables to identify some points of attention that further refinement of these services should take care of. The last part of the document provides a preliminary list of some implementations of such services. While clearly theses implementations do not provide the level of functionality required by SLICES, it enables to show that there exist partial implementations of these services. Moreover, it can also be used as an input, which has to be extended, for the tasks of the next phases of SLICES that will specify and implement these services.

Table of contents

1 SLICES Overview

1.1 Architecture overview

SLICES aims at providing high quality experimentation services with emerging technologies around the area of digital sciences. The goal is to build a large-scale infrastructure for experimental research in computer science, and more precisely in networking and distributed systems, targeting scientific challenges in the fields including wireless networking, IoT, edge/fog/cloud computing and distributed systems.

SLICES will be a highly distributed infrastructure, to reflect the fact that the environments we aim to study are themselves distributed (e.g., Fog/Edge computing), towards supporting a large variety of viable topologies in distributed computing systems. SLICES shall be a coherent environment to perform large scale distributed experiments. The knowledge and experience gathered from previous efforts and initiatives have resulted in the design of several tools and platforms that shall manage this infrastructure in a coordinated way, providing users with a consistent environment that shall overcome the technical challenges of multi-sites experiments.

Deliverable D2.3 [D2.3] presents an architectural view by identifying the various layer of SLICES architectures as depicted in Figure 1.

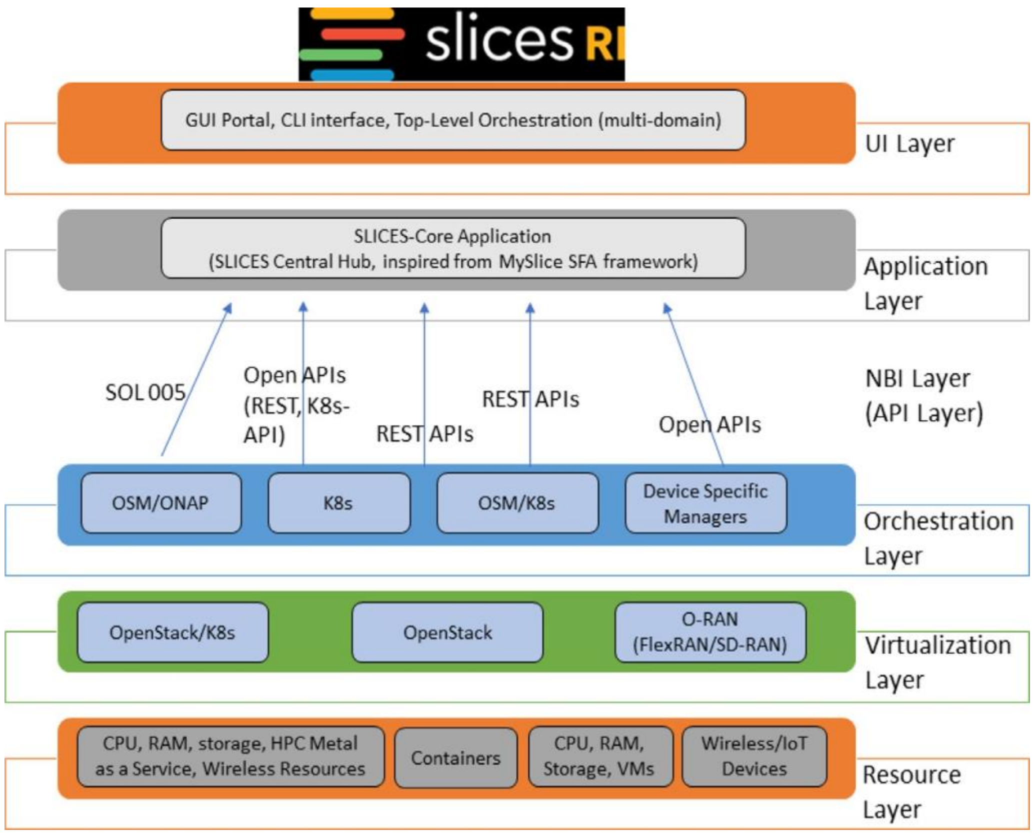

Figure 1: Layered architecture for SLICES

1. *Resource Layer*: It includes experimental resources such as CPU's, RAM, storage, containers, VM's, network, wireless, HPC and IoT devices;
2. *Virtualization Layer*: This layer includes cloud computing plat- forms (e.g., Openstack) that virtualize the underlying hardware resources and provide interfaces to the higher layers for programming/instantiating services over them. Examples of such programming interfaces are

the ones defined by the O-RAN alliance (e.g., A1/E2 interfaces), or the P4 programming abstractions for wired networks;

3. *Orchestration Layer*: It includes tools that orchestrate and instantiate services over the infrastructure equipment. Examples of such tools are OSM, ONAP and Kubernetes, mainly involved in NFV Management and Orchestration. It provides Network- Function-as-a-service and exposes northbound interfaces (NBI) APIs to be used by external entities;
4. *NBI Layer*: This Layer defines the Open APIs that can be used by the SLICES application framework. Examples of such inter- faces are the SOL005, the SOL004 from the ETSI NFV-MANO architecture[1] that can be found as the NBI interface of several MANO compliant tools, or even more generic ones, like TM-Forum based APIs for service lifecycle control;
5. *Application Layer*: This Layer will host the SLICES-Core application, located at the SLICES central hub. It is responsible for managing all experimental resources that are exposed by lower layers, saved in the database and is further exposed to experimenters as a Service-Catalog. It also exposes NBI API's that can be used by a 3rd party orchestrator. The architecture of SLICES-Core application will start from components similar to MySlice V2[2], and will be further enhanced at later stages;
6. *UI Layer*: This Layer defines the User Interface for the experimenters. It should abstract the experiments enough to make them more user friendly as possible.

This present document deals with the computer-based services envisioned to seamlessly access and make use of SLICES from a user point of view. These services aim at abstracting the layered view of the SLICES and are recalled in the next section.

1.2 SLICES Services

1.2.1 Overview

Deliverable D2.2 [D2.2] presents twelve services that have been identified during the SLICES-DS project as well as within the beginning of the SLICES-SC. These services are structured in the four following categories:

1. **User and platform management services**

   1.1: [USERS_MGT] User and group management

   1.2: [DOCUMENTATION] Documentation and Online Experiment Helpdesk

   1.3: [ACCOUNT] Accountability & billing

2. **Resource management services**

   2.1: [DISCOVERY] Resource discovery and description

   2.2: [RESERVATION] Resource reservation

   2.3: [CONFIGURATON] Resource configuration

   2.4: [MONITORING] Resource monitoring and profiling

[1] G. ETSI, 013: Network functions virtualisation, NFV; Management and orchestration; Os-Ma-Nfvo reference point–Interface and information model specification.

[2] L. Baron, R. Klacza, P. Gaudet-Chardonnet, A. Bradai, C. Scognamiglio, S. Fdida, Next generation portal for federated testbeds MySlice V2: From prototype to production, in: Fed4FIRE Engineering Conference - FEC2, 2017, poster URL https://hal.archives-ouvertes.fr/hal-01804013, [Last accessed 26 August 2022]

3. **Data oriented services**

   3.1: [DATA] Data Management Service

   3.2: [ANALYSIS] Experiment data validation and correlation with other experiments

4. **Experiment management services**

   4.1: [EXP_MGT] Experiment management

   4.2: [ORCHESTRATION] Experiment control and orchestration

   4.3: [DASHBOAD] Dashboard

D2.2 [D2.2] provides a detailed description of these services.

1.2.2 Experiment validation and correlation

Deliverable D2.2 [D2.2] does not provide an illustration of the usage of the experiment validation and correlation service. Therefore, this section provides an example of such a usage that is displayed in Figure 2 on the following page.

An experimenter (UserB in Figure 2) would like to replicate an experiment and compare her results to the previous experiment. To this end, she needs first to replicate the experiment, for example by retrieving the description of the experiment from the experiment management service provided UserA has allowed UserB to access it.
Then, the analysis service can be used to correlate her results (Data 2 in Figure 2) from previous results (Data 1).

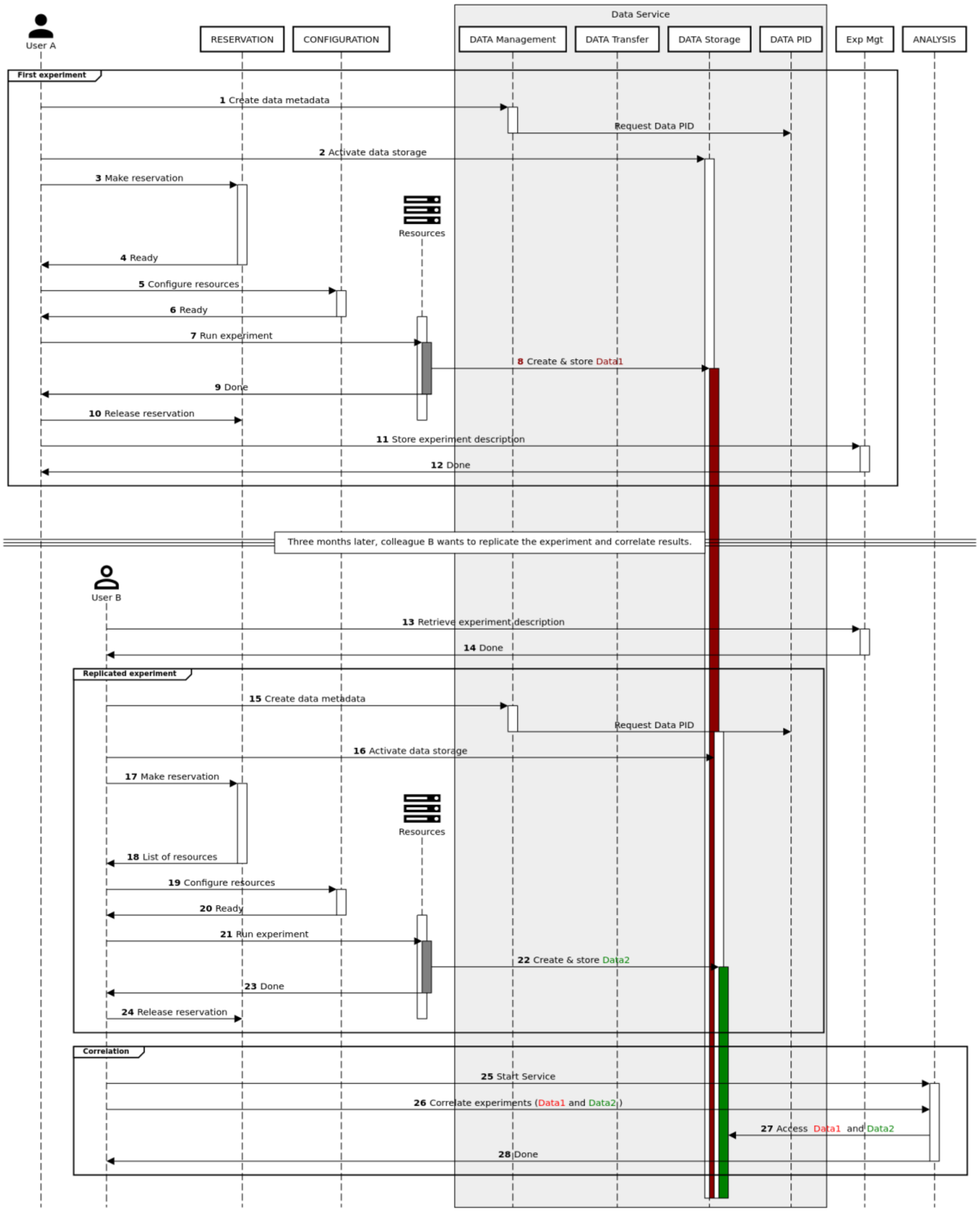

Figure 2: Replicating and correlating experiments

2 Analysis of the usage of SLICES Services

Deliverable D2.5 [D2.5] describes three use cases that deal with highly complex scenarios for Digital Science researchers, targeting three different scenarios with societal impact: physical disaster scenarios served by drones (Unmanned Aerial Vehicles or UAVs), a fully-automated smart city environment, and a digital-twin assisted recycling model for buildings.

This section briefly introduces each use case, infers an experiment that can needed, and discusses the services needed to be able to run it.

2.3 Rapid Resource Deployment for Physical Disaster Scenarios

2.3.1 Use Case Description

In this use case, the fixed communication infrastructure could be destroyed or unavailable due to high workload demand. For rescue operations, additional on-demand computing and network resources have to be deployed, typically mobile agents, such as robots or UAVs.

To serve the survivor devices as much as possible, there is a need to predict the kind and amount of resources these devices will request and the location of these resources. Some mobile edge resources may need to be deployed sporadically and temporarily at different locations based on IoT devices needs and mobility. Thus, there is a need to anticipate the deployment of edge services and to estimate the time they will be required at a given place to decide whether it is worth deploying durable edge resources, or instead mobile temporary resources could suffice. The trajectory of distributed mobile edge devices should be consciously planned accordingly, taking in consideration the time restrictions (robots should be deployed at the proper place before we need them).

Moreover, since the number of available robots may still be inadequate to serve all ground services, the prioritization of the applications, flows and devices is of paramount importance for the success of critical missions.

Even in the case of homogeneous mobile devices with identical computing and networking capabilities, their optimal allocation formulates a dynamic optimization problem, which depends on the size of the damaged area, the communication ranges, the propagation conditions, the data communication requirements (amount of data, frequency of collection, etc.) and the number and type of devices to serve for example.

This scenario illustrates the use and combination of the different control components of the framework, and in particular: 1) workload estimation in quantity, time and space, 2) resource allocation (tasks assignments to UAV and/or robots) and 3) path trajectory.

2.3.2 Experiments that can be run on SLICES

From all the issues that this scenario requires to be solved, let focus on the study of the impact of the deployed agents and their localization on the quality of services that can be achieved. From the experimentation point of view, it requires to deploy mobile agents, fix resources, and networks. In order to cover a large part of the problem space (quality of service, energy consumption, mobility, etc.), it is important to have access to a variety of type of hardware (computing and network). A goal of the experiment is to provide data to derive predictive models that could be applied in practice.

From a service point of view, the experiment workflow appears to be a classical one though the content of the experiment is complex: after selecting the needed resources thanks to the DISCOVERY service, the RESERVATION service is used to book the resources. The experiment is managed by the ORCHESTRATION service while the MONITORING service will be used to collect some data such as energy consumption for example. A critical part of the experiment is the deployment of the initial state of the experiment that involved in particular the CONFIGURATION service to correctly set up the mobile agent and the deployment of services on edge or cloud nodes. As this experiment may concern the addition or removal or degradation of resource to simulate various events that may occur during a disaster recovery scenario, the CONFIGURATION service should also be used during the execution of an experiment. Therefore, it is critical that the CONFIGURATION service does not add any overhead, or at least a very well described overhead, to minimize its impact on the experiment and/or to model its behavior.

In conclusion, this example of complex experiment could be set up with current identified SLICES. The critical parts for such experiments are the availability of diverse and numerous types of hardware (computing, networking, mobile agent) that will limit the parameter space that could be covered. The larger and diverse the SLICES platform is, the more cases could be tested. We have also identified that services that can used within an experiment, such as CONFIGURATION in this case, must be very well described so that their behavior can be taken into account in the analysis of the experiment results.

2.4 Smart-* applications: the smart cities' example

2.4.1 Use Case Description

This use case deals with how a usual event in a city such as a car accident could be managed tomorrow. First, the detection may involve the analysis and correlation of multiple sources of data (sensors within cars, by users' smartphones, and video streaming information such traffic light cameras, etc.). Second, a rapid and efficient reaction to the event requires coordinating multiple actors (Police, Emergency services, Public transport, district/municipal services...).

2.4.2 Experiments that can be run on SLICES

With respect to the detection phase, this scenario may need experiments with SLICES to observe, analyse, and model interactions between the IoT devices and the cloud resources where the applications are executed. It will help address the sizing challenge of edge resources in terms of computation, storage and network needs. In particular, such detection should involve distributed learning to reduce the amount of data to transfer as well as the coordination of various sources of data. For this sizing challenge, an experiment needs to access to varying size and type of resources (IoT, Cloud and network) to study where to deploy the various elements of the detection phase. Moreover, it can also study available algorithms used in the detection phase (typically distributed learning algorithms) to access their performance and behaviour under various conditions (number of events, size of the system, partial degradation, etc.).

With respect of service, the workflow of such experiments also appears to be well covered by identified SLICES services with respect to resource and experiment management service. However, it can be used to highlight the question of deploying complex user level software such as federated learning components. With current SLICES services, one possibility is to embed such components into a virtual image (or container) and to deploy them with classical cloud-oriented technologies. Though there are many works concerning the description of deployable application, SLICES may also require to be able to compose such description to ease the reuse of part of an experiment. With respect to the current scenario, we can imagine the re-use of federated learning components that could have been developed and tested into another context. Therefore, further studies are needed to define various levels of description of services such that Experiment management and Experiment control and orchestration to enable experiment description reuse as well as to be able to maintain under control the complexity of experiment description.

2.5 Automated Construction and Demolition Waste Management using digital twin for buildings

2.5.1 Use Case Description

This use case focuses on the establishment of a digital twin that will integrate the different stages of Construction and Demolition Waste CDW management, by providing an integrated digital approach to enable waste traceability and management, where built asset project information is managed through the whole life cycle. The digital twin will be established through the adoption and customization of a cloud-based collaboration solution, underpinned by existing standards, protocols, etc.

2.5.2 Experiments that can be run on SLICES

One experiment that may be needed by this use case is to build models to determine the type and localization of sensors and networks into the building. It is a typical case to where the solution requires to select a tradeoff between accuracy, energy consumption, resiliency, and cost for example. Therefore, an experiment may consist in characterizing the elements in various conditions, include external noise. A classical SLICES workflow experiment appears to be adapted to this experiment. A major criterion will be the availability of large and diverse type of hardware as well as the availability to collect accurate measure of their behavior, in terms of network parameter (latency, bandwidth, ...), computing (CPU and memory load, ...) as well as energy consumption. Therefore, the SLICES platform shall be able to collect and store all these data without affecting the experiment.
Once the data have been collecting, post-mortem analyzes will be needed to build the models. Depending of the amount of required computation, these analyzes could be handle by the SLICES platform or may require to be exported to large supercomputer for extreme cases.

3 Analysis of some existing services and technologies

This section aims at providing some examples of implementation of the SLICES services that are currently used in operational testbeds. Although these implementations do not provide all the features required by SLICES, their listing would help identify what is the existing experience and technology that could be leveraged in the context of SLICES, and what are the relevant ideas and design choices that have been made in similar contexts. This is an initial study that will be continued in the next phases of SLICES.

Table 1 hereafter provides a synthetic view of the analysis, which detailed in the remaining of this section. These services are in operation in the following research infrastructures: Grid '5000, FIT IoT Lab, IoT Lab (CH), PIONIER-LAB, PL-5G, FIT R2lab, FIT CorteXlab and NITOS.

Service	RI(s)	User and platform management services			Resource management services				Data oriented services				Experiment management services		
		USERS_MGT	DOCUMENTATION	ACCOUNT	DISCOVERY	RESERVATION	CONFIGURATION	MONITORING	DATA (storage)	DATA (transfer)	DATA (ID gen)	ANALYSIS	EXP_MGT	ORCHESTRATION	DASHBOARD
UMS	Grid'5000	X													
stats5k	Grid'5000			X											
Grid'5000 ref. API	Grid'5000				X										
OAR	Grid'5000					X									
OAR	FIT IoT-Lab					X									
OAR	FIT CorteXlab					X									
Kadeploy	Grid'5000						X								
KaVLAN	Grid'5000						X								
Testbed manager	FIT IoT-Lab			X	X	X	X	X							
EnOS	Grid'5000, Virtual Wall, Chameleon						X								
Kwollect	Grid'5000							X							
Jupyter interface	Grid'5000													X	
CKAN	IoT Lab		X						X						
GitLab	IoT Lab		X						X						
RI-MMS	PIONIER-LAB, PL-5G	X		X		X	X	X					X	X	X
R2lab API	FIT R2lab	X			X	X							X		
R2lab website	FIT R2lab	X	X	X	X	X		X							X
Rhubarbe	FIT R2lab				X	X	X	X							
nepi-ng	FIT R2lab						X						X	X	X
AccountMgt	FIT CorteXlab	X				X									
Minus	FIT CorteXlab						X							X	
CorteXlab-Dataset	FIT CorteXlab								X						
NITOS Portal	NITOS	X	X	X	X	X	X	X					X	X	X
OneLab Portal	OneLab	X	X		X	X									X
Openstack API	OneLab				X	X	X	X					X		

Table 1: Examples of existing services and technologies.

3.1 Grid’5000 User Management Service (USERS_MGT; Grid’5000)

Grid’5000’s *User Management Service* provides user and groups (projects) management with appropriate delegation mechanisms (for example, once a project is approved, the project manager can authorize additional users to use the testbed). Automatic account creation based on external identification mechanisms (such as identity federations) is possible, and implemented for users of Fed4FIRE. Additionally, this service also supports automatic accounts creation for students in the context of teaching.

3.2 stats5k (ACCOUNT; Grid’5000)

stats5k is Grid’5000 data warehouse system to gather data about the testbed (availability, usage, impact such as publications, ...), linked with the *user management service* and the HAL Open Archive for publications. It generates various reports to provide insight about top users/groups, and it is also used to track indicators.

3.3 Grid’5000 reference API (DISCOVERY; Grid’5000)

All Grid’5000 resources are described in the Grid’5000 reference API, with a collection of detailed and versioned JSON documents. It also serves as a basis for the generation of a set of documentation pages (such as https://www.grid5000.fr/w/Hardware). This service is described in depth in [TRI14].

3.4 OAR (RESERVATION; Grid’5000)

OAR is a resource management system, inspired from the HPC field. It supports selection of resources using SQL statements, batch scheduling (FIFO or priority based), time-based reservation in advance, etc. Its extensibility and flexibility are leveraged by Grid’5000 to build advanced reservation policies and resources selection capabilities.

3.5 OAR (RESERVATION, FIT IoT Lab)

FIT IoT-LAB reservation system is also based on OAR but only one central instance manages all remote IoT-LAB sites. Furthermore, the IoT-LAB OAR central instance is hidden behind the *testbed-manager* which provides a REST API for RESERVATION, CONFIGURATION and MONITORING.

3.6 OAR (RESERVATION, FIT CorteXlab)

FIT CorteXlab also make use of OAR to offer the service to book the CorteXlab room, to select radio nodes and to run experiments remotely.

3.7 Kadeploy (CONFIGURATION; Grid’5000)

Grid’5000 and FIT IoT Lab provide bare metal provisioning of system images using Kadeploy (described on https://www.grid5000.fr/w/Getting_Started for Grid’5000). Various system images are provided (with pre-installed software). Kadeploy is described in depth in [KA13].

3.8 KaVLAN (CONFIGURATION; Grid’5000)

KaVLAN is Grid’5000 network reconfiguration solution, described on https://www.grid5000.fr/w/KaVLAN. KaVLAN works by dynamically reconfiguring network switches to move nodes to specific VLANs. Network topologies can be created directly using KaVLAN or using higher-level third-party tools.

3.9 Testbed manager / FIT IoTLAB (ACCOUNT, DISCOVERY, RESERVATION, CONFIGURATION, MONITORING; FIT-IoT LAB)

The FIT IoT-Lab testbed management is a private software suite running at the master server of the distributed server and in charge of discovering the new resources to be opened to experimentation and synchronize the different sites to expose them to reservations. It manages the resource reservation at the frontend and is in close communication with OAR module that ensures the resource reservation at the backend. The testbed management also allows master server configuration and monitoring. Each remote site runs one *site-manager* REST server listening to orders from the central *testbed-manager*. At the end of the chain of command, each IoT resources are managed by the *iot-lab-gateway* REST server (https://github.com/iot-lab/iot-lab-gateway) waiting orders from the local *site-mana*ger. All user interactions are centralized by the *testbed-manager* REST API.

3.10 Enos (CONFIGURATION; Grid5000, Virtual Wall, Chameleon)

Enos aims at reproducible experiments of OpenStack. Enos relies on Kolla Ansible and helps users to easily deploy, customize and benchmark an OpenStack on several testbeds including Grid'5000, Chameleon and more generally any OpenStack cloud. Enos is described in depth in [ENOS].

3.11 Kwollect (MONITORING; Grid'5000)

Kwollect is a service to collect infrastructure metrics (including high-frequency wattmeters) and expose them to experimenters. Kwollect scales to high frequencies of metrics collection for hundreds of nodes. It can also be leveraged by the experimenter to collect custom metrics. Kwollect is described in depth in [KW21].

3.12 Jupyter interface (ORCHESTRATION; Grid'5000)

Grid'5000 provides a flexible Jupyter notebook interface, that provides access either outside the experiment context (if the notebook includes resources reservation), or inside it (to orchestrate data the experiment itself). It is described in [BN21].

3.13 CKAN (DATA, DOCUMENTATION; Mandat International, IoT Lab)

Mandat International and the IoT Lab testbed offer several services to manage the data related to the experiments done in the research infrastructure. Firstly, a CKAN server permits to published in an open manner the different documents such as scientific papers, journal articles, deliverables and data sets. The CKAN server allows to implement FAIR principles during the publication phase of data and documentation related to the research infrastructure.

3.14 GitLab (DATA, DOCUMENTATION; Mandat International, IoT Lab)

A GitLab server is available to store source code and the related technical documentation. The documentation can be done on several formats such README.md files or Wikis. The GitLab server provides also the tools to realise Continuous Integration/Continuous Delivery (CI/CD) pipelines to ease the deployment on virtual machines used during the experimentations.

3.15 RI-MMS (USERS_MGT, ACCOUNT, RESERVATION, CONFIGURATION, MONITORING, EXP_MGT, ORCHESTRATION, DASHBOARD; PL-5G and PIONIER-LAB)

The purpose of RI-MMS (Research Infrastructure - Management and Monitoring System) is to provide an IT system for managing experiments in a distributed heterogeneous research infrastructure. The system is designed on the basis of the experience gathered from the implementation of an analogous solution for the management of research infrastructure offered under the PL-LAB and PL-LAB2020 projects.

The resources of individual laboratories are available to a wide range of users, ranging from scientists, through representatives of small and medium-sized industries, to individual users. Each logged-in user gets an access to the catalog of services offered by individual research laboratories. Using a dedicated form, the user can request specific resources for the time needed to carry out his/her research work.

The use of laboratories and their resources requires a confirmation of the user's identity. The RI-MMS integrates different identity providers, in particular eduGAIN and the Polish-wide Id management platform – PIONIER.Id. The added value of compatibility with eduGAIN is the compliance with the EOSC (European Open Science Cloud) platform, which ensures the possibility of transparent publication and sharing the results of scientific research.

Users can select the location of the resources they intend to use and enter a detailed specification of resources (e.g., parameters of virtual machines to be deployed on demand) and indicate the topology of connections between the reserved resources. The system also provides a tool that allows the user to invite other participants who can obtain access rights to the shared resources in order to jointly conduct research work.

The RI-MMS software is based on a distributed architecture. The central module of the system cooperates with the heterogeneous infrastructure of laboratories in order to implement the following processes:

- configuration of devices in accordance with the reservation specification provided by the user
- configuration of user access to reserved resources
- collection of information about the availability of resources

Moreover, the RI-MMS supports the manual process of resource allocation, in case the automatic resource configuration is not possible (e.g., due to the nature or limitations of the resource).

Under the hood, RI-MMS uses the Netbox software as a way to store data about each device in each laboratory along with their access credentials. Additionally, resources modeled with Netbox can have relations or be a part of the hierarchy, which allows for creating complex network topologies.

Currently, the software is being deployed for the management of research infrastructures built in PL-5G and PIONIER-LAB projects.

3.16 FIT-R2lab API (USERS_MGT, ACCOUNT, DISCOVERY, FIT-R2lab)

Located at the core of R2lab's infrastructure, this component provides an xmlrpc API, so that programs can interact with the testbed in terms of, typically, account management and reservations; its full documentation can be found at https://r2labapi.inria.fr/db/doc/PLCAPI.php.

3.17 FIT-R2lab website and reservation service (RESERVATION, MONITORING; FIT-R2lab)

As a specific development around the basic API capabilities, R2lab users interact on a daily basis with the platform through the dedicated website at https://fit-r2lab.inria.fr/. It provides for one-click reservations and real-time status display of the testbed components.

3.18 Rhubarbe (DEPLOYMENT; FIT-R2lab)

A dedicated software layer called 'rhubarbe' takes care of the low-level management of R2lab nodes and phones; among others, it supports the ability to save, and to massively deploy, low-level disk images, and in this respect is one of the core pillars of R2lab's reproducibility capabilities. It documentation is available at R2lab's website, at https://fit-r2lab.inria.fr/tutorial.md.

3.19 nepi-ng (ORCHESTRATION; FIT-R2lab)

This component focuses on experiments synchronization, and allows R2lab users to design and program their experiments in a reproducible way, while taking into account synchronization needs between all the physical equipments that need to cooperate in order to fulfill the experiment's constraints. nepi-ng is described at https://nepi-ng.inria.fr/ ; it is made of 2 layers of asynchronous-Python ('asyncio') libraries, and is thus readily usable from any Python runtime, including Jupyter or colab notebooks.

3.20 AccountManagement/CorteXlab (USERS_MGT, RESERVATION; FIT-CorteXlab)

The *accountManagement* interface of CorteXlab offers user services to create and manage their accounts, and to access to their reservations through OAR.

3.21 Minus/CorteXlab (CONFIGURATION, ORCHESTRATION; FIT-CorteXlab)

Minus is a service allowing to deploy experimentations on the radio nodes of FIT-CorteXlab, to run these experimentations and to get the results. Its documentation is available at website https://wiki.cortexlab.fr/doku.php?id=get_started.

3.22 CorteXlab-Dataset (DATA; FIT-CorteXlab)

The repository (https://wiki.cortexlab.fr/doku.php?id=dl-datasets) provides access to the datasets generated on FIT-CorteXlab, for radio experimentations in a shielded area.

3.23 NITOS portal (USERS_MGT, ACCOUNT, RESERVATION, CONFIGURATION, MONITORING, EXP_MGT, ORCHESTRATION, DASHBOARD; NITOS testbed)

This component focuses on user management by employing a PostgreSQL component for storing user data with respect to accessing the underlying infrastructure. The component is able to allow a calendar-based reservation of wireless resources in the testbed (wireless nodes of different technologies, wireless spectrum allowed per experimenter) so as to allow isolation of users when multiple are running experiments concurrently. The NITOS Portal software is broken down to different components dedicated to different tasks; for example, configuration of a node to communicate with other testbeds over the GEANT network is accomplished by specifying the stitching points through the NITOS portal or configuration of programmable attenuation on the antenna outputs can be configured using some pre-defined routes in the city (mobility emulation framework). Experiment control is provided by invoking the respective OMF based framework commands (OMF – cOntrol and Management Framework [OMF]). Finally, the portal is providing documentation for accessing the resources or deploying experiments that are available in the NITOS experiment database. The framework is also providing monitoring of resources to the end-users, showing the health and current status of them, while it is also communicating through SFA with other testbeds. The portal has also been extended to support connections for orchestration with other frameworks, such as Open Source MANO and Kubernetes

3.24 OneLab portal (USERS_MGT, DOCUMENTATION, DISCOVERY, RESERVATION, DASHBOARD; OneLab testbed)

The portal https://portal.onelab.eu offers the possibility to reserve resources to run experiments combining multiple heterogeneous resources (IoT, Wireless and Cloud). It offers user management services to create and manage user accounts and also with the identity federation (GENI), it allows authentication from and to external source (CloudLab). It provides description of the reservable resources and documentation for accessing the resources and deploying experiments.

3.25 Openstack API (DISCOVERY, RESERVATION, CONFIGURATION, MONITORING, EXP_MGT, OneLab testbed)

This component provides a REST API based on the SFA (Slice-based Facility Architecture) and GENI AM API specification to advertise and allocate resources and also to monitor the resources and the status of the experiments. It allows Cloud-specific configuration to be pushed on the resources during the provisioning phase and provides automation using Ansible.

4 Conclusion

After providing an overview of the SLICES architecture and SLICES user-oriented services, this document focuses on a first analysis of the usage of such services with respect to three complex use cases. For each use case, a type of experiment has been presented as well as some impacts on SLICES services. It has enabled to confirm that the twelve identified services shall be able to cover complex use cases. It has also enabled to identify some points of attention that further refinement of services should take care of. The last part of the document provides a list of current implementations of such services. While clearly theses implementations do not provide the level of functionality required by SLICES, it enables to show that there exist partial implementations of these services. Moreover, it can also be used as an input, which has to be extended, for the tasks of the next phases of SLICES that will need to specify and implement these services.

5 Bibliography

[BN21] Luke Bertot, Lucas Nussbaum. Leveraging Notebooks on Testbeds: The Grid'5000 Case. In CNERT 2021.

[D2.2] SLICES-DS, D2.2 “SLICES as a Service, baseline”, 2022.

[D2.3] SLICES-DS, D2.3 “SLICES Reference Architecture”, 2022.

[D2.5] SLICES-DS, D2.5 “Use cases validated”, 2022.

[ENOS] Ronan-Alexandre Cherrueau, Dimitri Pertin, Anthony Simonet, Adrien Lebre, Matthieu Simonin. Toward a Holistic Framework for Conducting Scientific Evaluations of OpenStack. In the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), May 2017, Madrid, Spain.

[KA13] Emmanuel Jeanvoine, Luc Sarzyniec, Lucas Nussbaum. Kadeploy3: Efficient and Scalable Operating System Provisioning. USENIX; login: volume 38, number 1, pages 38-44 (February 2013).

[KW21] Simon Delamare, Lucas Nussbaum. Kwollect: Metrics Collection for Experiments at Scale. CNERT 2021 - Workshop on Computer and Networking Experimental Research using Testbeds, May 2021, Virtual, United States. pp.1-6.

[TRI14] David Margery, Emile Morel, Lucas Nussbaum, and Olivier Richard, Cyril Rohr. Resources Description, Selection, Reservation and Verification on a Large-scale Testbed. In TRIDENTCOM - 9th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities, 2014.

[OMF] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. 2010. OMF: a control and management framework for networking testbeds. SIGOPS Oper. Syst. Rev. 43, 4 (January 2010), 54–59. https://doi.org/10.1145/1713254.1713267

slicesDS