NORDIC E-INFRASTRUCTURE COLLABORATION

PUHURI Deliverable D3

Project Title:	Puhuri
Public	This document is PUBLIC
Deliverable title:	LUMI Integration to Puhuri
Contractual delivery date:	20th of October 2021
Status	v1.0
Actual delivery date:	16th of November 2021
Updated	11th of November 2021
Partner(s) contributing to this deliverable:	FI, EE, SE

Authors and Contributors:

Marina Adomeit, Kent Engström, Kalle Happonen, Jarno Laitinen, Ilja Livenson, Juha Nyholm, Ahti Saar, Sergei Zaiaev and Anders Sjöström

Table of Contents

Introduction

This deliverable is a report of LUMI integration to Puhuri Services. EuroHPC Joint Undertaking (JU) is acquiring pre-exascale and petascale supercomputers (the EuroHPC supercomputers) which will be located at and operated by supercomputing centres in the Union. LUMI [LUM] is one of those. It is hosted at CSC in Kajaani data center. LUMI is the main use case for Puhuri. LUMI consortium countries may integrate their national portal to Puhuri or use Puhuri provided portal to manage the LUMI projects. The resource application review process of the call is out of Puhuri's scope. The LUMI integration is described in Chapter 5. As the previous deliverable 2 [D2], includes an explanation of planned CSC integration with LUMI so this deliverable partially repeats it.

The documentation in https://puhuri.neic.no [PUH] has been continuously extended. Please refer to that website for the latest technical details. The upcoming Deliverable 4 will report the accounting and reporting separately.

In addition to technical implementation of the integration, legal aspects have to be taken into account, in particular agreements and contracts regulating transferred and preserved data are needed. Due to Puhuri services operating with personal data of the users, GDPR needs to be applied. We have found a model for the GDPR roles for the LUMI piloting phase, where LUMI Consortium has taken a role of a Joint Data Controller in LUMI, Puhuri Core and Puhuri AAI (IdP Proxy) but it is subject to change in the future when more services join Puhuri.

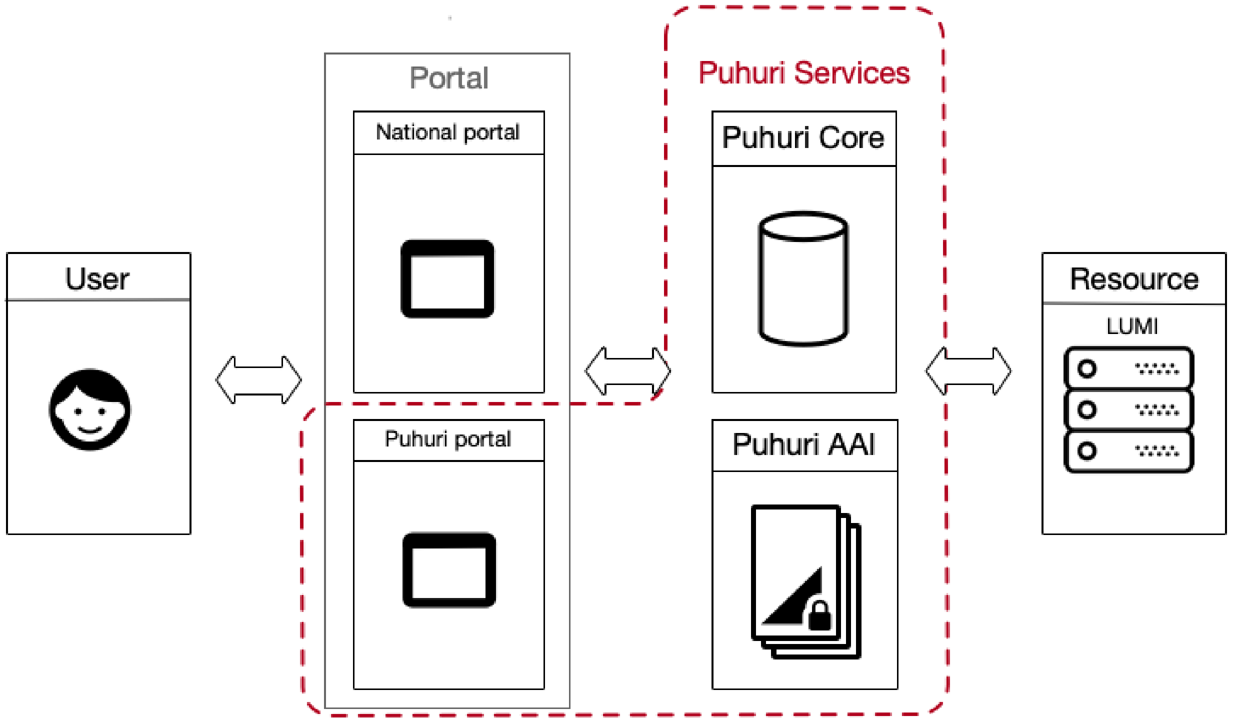

Figure 1. Puhuri's high level architecture. User uses either Puhuri Portal or National Portal.

Chapter 1 summarises the key processes regarding user registration and project allocation. The implementation of the processes are explained in the following chapters.

Chapter 2 describes the authentication and authorisation infrastructure services. User identity management is based on the MyAccessID AAI service operated by GEANT and technically based on eduTEAMS. Puhuri is an Infrastructure Service Domain (ISD) of MyAccessID.

Chapter 3 describes Puhuri Core, where information about resource allocation requests and fulfillment is kept. It provides programmatic interfaces (APIs) for the portals used by resource allocators, service providers like LUMI as well as support teams.

Chapter 4 reports how SNIC is integrating their national portal (SUPR). This is especially interesting for those who plan to integrate their existing national portals to Puhuri.

Chapter 5 focuses on how CSC as a Service Provider is integrating to Puhuri Core and Puhuri AAI.

1. Puhuri key processes

1.1. User Registration

All users must register in Puhuri AAI to use Resources connected with Puhuri, in particular LUMI. The Registration creates a unique identifier (Community Unique Identifier, CUID) for the User, which is used for referencing and linking user identity across the different components.

The identity provider releases the attributes about User's identity and affiliation, which is important for the resource providers to know. Furthemore, Level of Assurance of such data is provided by Puhuri AAI. Users can also provide additional information e.g. SSH public keys. Users can define alternative email addresses than what was returned from the Identity Provider used for registration in Puhuri AAI. That email will then be verified.

A user must also accept the Terms of Use agreement (ToU) as well as Acceptable Use Policy (AUP) document of the Puhuri AAI services. At the moment there are LUMI Terms of Use and we have to plan how to scale this for more services.

The national portal may implement identity linking of local accounts with a Puhuri AAI account. The recommended way for that is to use OIDC registration flow, which can be triggered from the national portal and redirect back with a created Puhuri User CUID

1.2. Project allocation

The Resource application review process happens outside of Puhuri. Once the project has been approved, Resource Allocator is responsible for pushing project information to Puhuri Core, from where it is then synchronised to the Service Provider. In addition to Project metainfo and membership data, allocation-specific attributes are included. The detailed description is in Chapter 3.

1.3. Other workflows

There are various other workflows related to the User and Project lifetime management including updating the User and Project related information.

Accounting information flows from the Resource to Puhuri Core and then to the portals. The Deliverable 4 describes that in more detail.

2. AAI implementation

2.1. AAI Architecture

Puhuri AAI is one of the essential Puhuri Service components as it enables Users to securely access to Resources such as LUMI. Draft of Puhuri AAI, which is compliant with AARC blueprint architecture, was presented in Deliverable 1 [D1]. In Deliverable 2 was discussed about Infrastructure Service domains. Since then, the central AAI has been branded as MyAccessID, as presented in Figure 2 that shows Puhuri AAI Architecture.

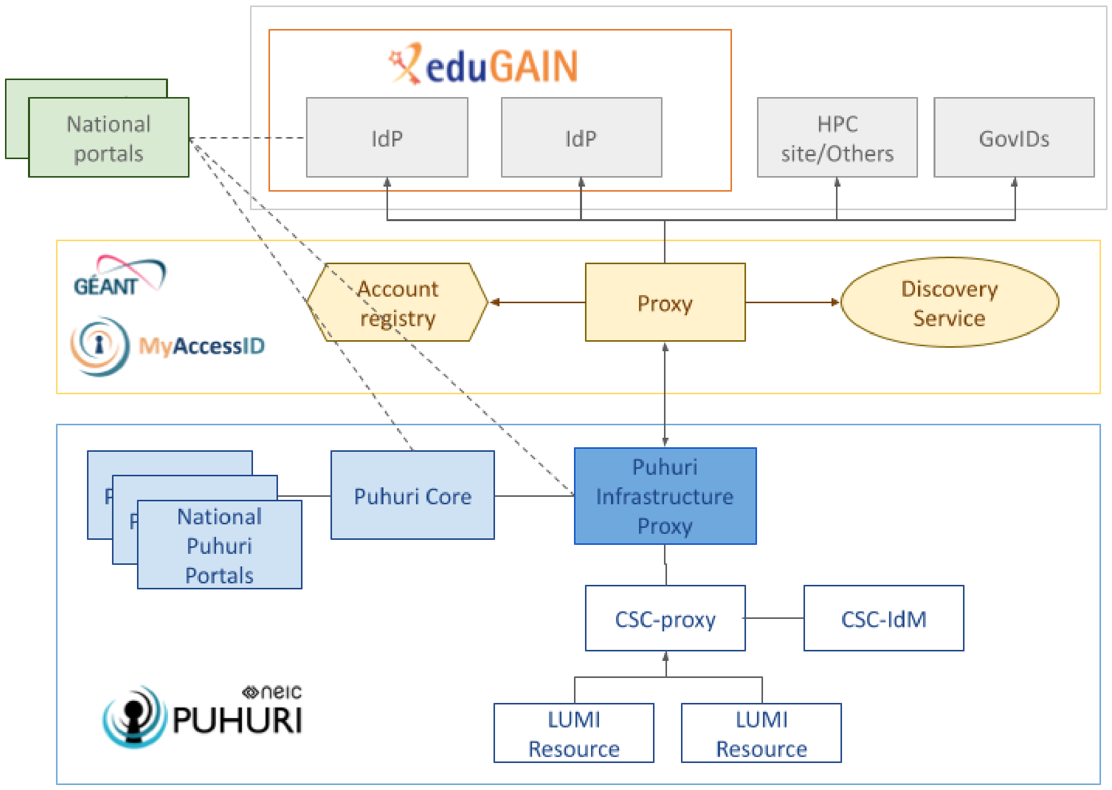

Figure 2: Puhuri AAI Architecture

GEANT delivers MyAccessID IAM platform (Identity and Access Management) that connects federated R&E IdPs via eduGAIN and HPC and Community AAI IdPs with the purpose of offering a common Identity solution for multiple HPC communities. With MyAccessID, users will be able to apply for and use HPC resources with their MyAccessID using their IdP account, without having to create and manage separate logins for each system. MyAccessID will be able to connect National eID systems as authentication sources as needed. GEANT also operates Puhuri Infrastructure Proxy, which connects the LUMI resources and allocation portals to MyAccessID.

Both MyAccessID and Puhuri Infrastructure Proxy are deployed in a production environment. GEANT has a service for testing attribute release by the Identity Providers. This have been used by LUMI countries integrators and LUST members to test attribute release from IdPs from their countries. Initial results are good, but Identity Providers will have to step up their capabilities to signal expected Level of Assurance.

In collaboration with LUMI countries integrators, their National Puhuri Portals are being connected to Puhuri Infrastructure Proxy as portals Terms of Use and Privacy Notices are being confirmed by the respective portal controller/owner.

2.2. Remaining future work

Following work is planned to further evolve Puhuri AAI:

- For users without a federated identity, e.g. users from the industry, needs to be further developed. This is of high importance as users from industry users.
- As a first measure, we are planning to enable usage of national eIDs solutions based on the eIDAS to eduGAIN connector operated by SUNET. For those users where national eID is not available either, a fall back solution to register and manually vet users will need to be put in place. These would be features supported by the MyAccessID platform, and conversations with GEANT about ways to support these are ongoing.
- Governance for Puhuri Infrastructure Proxy and policies for connecting services, security policy etc need to be defined.
- A date for when the required level of assurance described in Deliverable 2 will become mandatory needs to be defined so that AAI integration contacts can use this to influence implementation at IdP side. We will monitor the compliance with the level of assurance requirements on MyAccessID platform,

3. Resource allocation technical setup

In the Puhuri project, Waldur software[1] is being used for resource allocation and accounting. Waldur is an open-source cloud marketplace with a self-service environment for users to request and get access to resources. Waldur is deployed in two separate roles: as Puhuri Core and as a reference solution for Puhuri Portals. In the latter case, Waldur was extended to integrate with Puhuri Core APIs in the same way as National portal integration is done.

Puhuri Core is the heart of the resource allocation as it holds necessary information about users, group management, roles, resources, accounting etc. Puhuri Core is accessible using REST API with target users of API being:

- National allocation portal;
- Shared international allocation portal[2];
- Service Providers that want to use allocation information via Puhuri Core.

If the Resource Allocator does not have a national allocational portal or is not willing to integrate at once, Puhuri partner Estonian National Research Infrastructure (ETAIS) is providing an option for setting up a reference national portal either as a managed version or as a supported software package. There is an agreement framework in place including service description, pricing, Service Level Agreements and Data Protection Agreements. Alternatively, Resource Allocator can self-host the reference Puhuri Portal.

EuroHPC Joint Undertaking's resource allocation is handled through PRACE. The Puhuri project has an ongoing dialogue on how to enable PRACE to use the Puhuri system for the allocation of the EuroHPC JU portion of the LUMI resource.

In the Puhuri Core case, registered organizations are the Resource Allocation organizations, which allocate resources from service providers. In case of LUMI, a single service provider - LUMI - has been created with multiple offerings corresponding to the EuroHPC project types and Resource Allocators as nominated by LUMI Consortium.

3.1. Environments

In total, we operate three different deployments for Puhuri Core. Each environment[3] has its own intention and policy.

- Dev (https://puhuri-core-dev.neic.no/) runs the latest code and is updated frequently.
- Beta (https://puhuri-core-beta.neic.no/) is deployed with the versioned code, either the same or newer than on production and contains production structure for resource allocators.

[1] Waldur site - https://waldur.com/
[2] Puhuri Portal - https://puhuri-portal.neic.no/
[3] https://puhuri.neic.no/environments/

- Production (https://puhuri-core.neic.no/) is intended to be used for production data exchange.

The development environment is connected to the pre-production MyAccessID AAI (eduTEAMS-based AAI), Beta and Production environments are connected to the production MyAccessID AAI. All deployments implement the business processes we have decided so far in the Puhuri project. The whole deployment process is designed to allow for dynamic changes into the processes during the stabilization phase. In addition to OIDC flow, Puhuri Core implements a custom integration with MyAccessID for looking up user data based on provided unique references of the user, aka CUID.

3.2. Information stored in Puhuri Core

Puhuri Core stores information about projects, users, allocations and relations between those. Every project must contain members (PI, co-PI and regular members) and allocation (CPU, GPU or storage components).

Data from reference Puhuri Portal is synced with Puhuri Core only if the project has allocations and this project is approved by the Resource Allocator. If the data is present in the Core, then it will be synced with Service Provider. This synchronization interval is Service Provider specific.

Some of the project attributes are Service Provider specific and these existence depends on the requirements from the Service Provider. For example, the OECD science domain is required by LUMI and is mandatory for all allocations in LUMI.

More information can be found on Puhuri documentation website[4].

3.3. Puhuri Core integration with Puhuri AAI

Puhuri Core is integrated in with MyAccessID AAI in the following ways:

- Via standard OIDC protocol though Puhuri AAI proxy, where Puhuri Core acts as a client. This allows it to do permission grants as well as accept logins of Puhuri users into Core, for example representatives of Resource Allocators and support team members. While this is technologically possible, direct end user login to Puhuri Core is not foreseen in the general case.
- Access to user registry over a custom MyAccessID userinfo endpoint for retrieving user data from the registry based on the lookup. Such integration is needed to be able to present the National Portals API-endpoint for adding users to the resource allocation using the unique identity reference of the Puhuri AAI. Access is based on the offline access of OIDC.

[4] Puhuri documentation - https://puhuri.neic.no

3.4. Puhuri Core functionality for Resource Allocators

Resource Allocators are external parties that are entitled to manage projects and resource allocations in Puhuri Core via Puhuri Portal or national portal. In the scope of Puhuri, Resource Allocators are typically organizations operating National Portals or research communities.

Puhuri Core provides a rich set of APIs along with Python SDK for easy integration of Puhuri Core services with portals of Resource Allocators. The up-to-date documentation for the allocators is kept at https://puhuri.neic.no/resource-allocators/.

3.5. Puhuri Core functionality for Service Providers

Currently, the main service provider for Puhuri is CSC with its service LUMI. As such development of the functionality is largely driven by use cases that arise from LUMI, however we believe that they are generic enough to integrate a wide range of similar Service Providers.

A typical flow of operations from the Service Provider point of view is as follows:

1. Organization representing an allocation body creates allocation in a certain project. Allocations are created in the 'Creating' state.
2. Service Provider polls for new allocations and users with access to them, processes them and marks allocations as approved or rejected. Service Provider also reports to Puhuri Core local username created for the user accounts as well as local references to allocations.
3. Service Provider regularly reports back accounting data as well as can provide status report for allocation (a semi-structured report visible to end-users).

From the Service Provider side, Puhuri Core provides several groups of APIs. Functionality related to the setup of environment on the Service Provider side:

- Getting a list of active projects where Service Provider’s resources are available.
- Getting a list of active Allocations of Resources.
- Getting a list of members and their roles in each project.

Functionality related to reporting of accounting of resources

- Updating usage of each Resource Allocation.

Functionality in progress includes:

- Ability to report on the status of the environment provisioning to improve feedback to resource allocators.
- Ability to report locally generated usernames for each Puhuri AAI identity.

Service Provider documentation is available at https://puhuri.neic.no/service-providers/.

3.6. Puhuri Portal as an option to manage allocations

In order to validate Puhuri Core APIs as well as to provide a graphical tool for the resource allocators that do not have or do not want to integrate their own allocation portal, we developed Puhuri Portal as one option to manage projects and allocations. As of today, two slightly different portal options are available for Resource Allocators.

- Shared version[5] - this is a common version of the portal, meant for several Resource Allocators and with the fixed Privacy Policy and Terms of Service.
- Dedicated version - meant for typically one Resource Allocator with customizable Privacy Policy and Terms of Service.

Based on the same technology as Puhuri Core, dedicated Puhuri Portal users will have additional roles to reflect established processes of resource allocation. The roles are:

- Organization owner: represents a scientific organization or community, able to create new projects, manage project teams and manage requests for resources by approving or rejecting them.
- Principal Investigator (PI): able to request and use provisioned resources (with approval by Organization owner), can add project members from the same organization (i.e. pre-approved by the owner).
- co-Principal Investigator (co-PI): same rights as PI.
- Project member: can only participate in particular project work, able to use provisioned resources.

In the shared Puhuri Portal, the organization owner is renamed into Resource Allocator and all projects are affiliated with a Resource Allocator.

Puhuri Portal uses MyAccessID authentication for all of these mentioned roles. Organization owner roles are set to specific persons by the Puhuri team after they register in MyAccessID and in the portal. Further on, organization owners can manage other roles and persons.

Resource allocation process is designed to be flexible and generic, meaning that we standardize the main flows and actions, but allow for customisations, where we foresee changes. For example, we expect LUMI to provide a common schema for allocating resources for the initial LUMI services (cpu-hours, gpu-hours and storage-hours), but we

[5] Shared Puhuri Portal - https://puhuri-portal.neic.no

allow to provide own allocation components to other service providers or not yet clearly defined resources under LUMI umbrella. As such, Puhuri Core serves as a generic solution for sharing of resources by multiple allocators.

4. Integration of the Swedish national portal to Puhuri

4.1. Background

The SUPR project began in 2011 to provide SNIC (the Swedish National Infrastructure for Computing) and the multiple computing centres providing SNIC resources with a unified database of users and projects, as well as support for handling project proposals within different application rounds.

Users with different roles (PIs, project members, reviewers, staff etc.) use the portal at https://supr.snic.se/ to perform their various tasks. The primary authentication method is federated login via the Swedish SWAMID identity federation, but we also allow email/password login for those who do not have a SWAMID identity, and client certificate login for those who wish to use that. Staff users and PIs for sensitive data projects are required to use two factor authentication (2FA). TOTP[6] in SUPR itself on top of the primary method, and all users can enable 2FA if they wish.

The main reasons for integrating SUPR with Puhuri are:

- SUPR already has the functionality needed to accept project proposals of different kind, assign reviewers and let them register their evaluations, handle and document decisions, and then create projects from the approved proposals.
- PIs and project members know how to use SUPR for managing their proposals and projects and we expect most of them to have a mix of projects at LUMI and at national resources.

In the remainder of this part of the document, we describe how we have integrated SUPR with Puhuri, as this may be useful for other resource allocators that want to do the same for their national portals.

4.2. Puhuri AAI / MyAccessID Integration

The core user identity in the Puhuri system is the MyAccessID identity (also known as a CUID within Puhuri). For SUPR to be able to talk to the Puhuri Core system and add/remove users to projects, it needs to know the users' MyAccessID identities.

[6] https://en.wikipedia.org/wiki/Time-based_One-Time_Password

This is implemented using a function in SUPR that allows a user to link their MyAccessID identity to their SUPR user. Users who become members of LUMI projects are reminded that they need to do that.

The technical implementation uses an OIDC flow from SUPR to the Puhuri AAI proxy[7]. As we run Apache httpd and use other Apache modules for SAML and client certificate authentication, we use the `mod_auth_openidc` module to take care of the OIDC protocol. We enable it for the MyAccessID linking page in SUPR.

We have added a new field to our Person object in SUPR to store the linked MyAccessID identity. In addition, we also store the name and email address returned from MyAccessID, to be able to show them to the user, as the MyAccessID identity itself is a long opaque identifier.

4.3. Puhuri Core Integration

SUPR needs to talk to Puhuri Core to push information about projects, memberships and allocations. This is implemented using a sync script that is run regularly, and using the Puhuri Core APIs documented at https://puhuri.neic.no/resource-allocators/.

4.3.1. Creating and Updating Projects

Project creation is done using the API documented at https://puhuri.neic.no/resource-allocators/#project-creation. Here we need to map fields from our Project object to the Puhui Core fields:

Puhuri Core Field	**SUPR Field**	**Comment**
name	name	Formal name/code, for example "SNIC 2021/99-42"
description	title	For example "Monte-Carlo simulations of feline table-object interactions"
backend_id	id	Database ID on our side (an integer)
end_date	end_date	SUPR uses exclusive semantics while Puhuri Core uses inclusive, so 2022-01-01 will be translated to 2021-12-31.
oecd_fos_2007_code	classification1	SUPR has a five digit code that needs to be translated. For example, "21001" (for Nano-technology) maps to "2.10".

[7] https://puhuri.neic.no/idp_integration/use-cases/national-portal-integration/#overview

The concept of a start_date is not yet present in Puhuri Core. Projects become active as they are created. Therefore, we do not create the project in Puhuri Core before the start_date in SUPR has been reached.

We have added a field in the SUPR Project object to keep track of the Puhuri Core UUID for the project, for use in API calls referencing the project. We have also added a field to the SUPR Project object to keep track of the last time we updated the project information in Puhuri Core. As we already have a field for the last time the project's relevant fields were updated in SUPR, the sync script can compare those timestamps. If the project has been updated in SUPR since the last update done by us in Puhuri Core, we fetch the project object from Puhuri Core, compare field values, and use the API call documented at https://puhuri.neic.no/resource-allocators/#project-update to update any changed value.

4.3.2. Project Membership Management

SUPR projects have exactly one PI, zero or one proxy for the PI, and one or more members. Those concepts map to the Puhuri Core permission roles manager, admin and member respectively as described at https://puhuri.neic.no/resource-allocators/#membership-management

We need the Puhuri Core UUID for a user before we can reference that user in the permissions API. We have added a field to the SUPR Person object to store that. If that is empty, but we have the MyAccessID present, we first use the call documented at https://puhuri.neic.no/resource-allocators/#puhuri-aai-user-mapping-lookup to map the MyAccessID identity to the Puhuri Core UUID and store it for future use.

As part of project update handling, we compare the permissions present on the Puhuri Core project with the PI, proxy and members in SUPR, and issue the necessary calls to add and/or remove permissions from Puhuri Core to make it consistent with the SUPR project.

4.3.3. Resource Allocation Management

To communicate to Puhuri Core how many core hours, GPU hours and/or TB hours a project is allowed to use, we use the API calls documented at https://puhuri.neic.no/resource-allocators/#resource-allocation-management

We have added a field to the SUPR Project object to specify what Puhuri Core offering (for example, "LUMI SNIC / Regular Access") is to be used for the project.

We have also added a field to the SUPR Round object, that carries the default offering to be used for projects created from that application round.

We create a marketplace order with the limits taken from the allocations for the project in SUPR and store the resulting UUID for the order in a new field on the SUPR Project. When the sync script runs again, it will check if the order has been marked as done. When that is the case, it saves the resulting resource UUID in yet another new field on the SUPR Project object.

As part of project update handling, we fetch and compare the limits present on the project's resource object with the current allocations present in SUPR. If they differ, we use the update_limits API call to adjust them.

4.4. Future Work

LUMI (and therefore Puhuri Core) uses allocation units that are not used elsewhere in SUPR. LUMI uses core-hours and GPU-hours for the whole project duration, while SNIC resources use core-hours per month and GPU-hours per month. Also, LUMI uses TB hours (the time integral of the storage usage) while SNIC resources use TB or GB (the maximum storage usage at any time). We have added the LUMI units to SUPR, but still have work to do to make sure that we show the LUMI units in the correct way in allocation graphs, etc.

During the next phase of the Puhuri Project we will decide how usage accounting from LUMI will be propagated via Puhuri Core to national portals and Puhuri Portals. We will have to integrate this too with SUPR and decide if this will be done via the SGAS database that is used for other SNIC accounting, or if it will be stored in SUPR only. The amount and granularity of accounting data we can get will influence that decision.

There may also be other information, in addition to usage accounting, that needs to be sent from LUMI via Puhuri Core to national portals and Puhuri Portals, such as the actual user names used to login to LUMI and the actual project names (the Slurm account used) on LUMI. This will require changes to the way the sync script works, as it will then have to poll for changes from Puhuri Core too to fetch such data, in addition to looking for changes in SUPR to push data to Puhuri Core.

If it works policy-wise, we may want to enable MyAccessID as an authentication alternative for SUPR itself in the future. Also, SNIC centres may be interested in having MyAccessID authentication as a system login method if it turns out to be popular on LUMI.

5. Resource integration with the LUMI use case

CSC has implemented two scheduled scripts for data synchronization with Puhuri Core and its Identity Management (IdM) system. The first script is used to pull in User, Project and Allocation data from Puhuri Core to CSC IdM. The pull-script fetches all the relevant information from Puhuri Core's REST API endpoints and transforms the data into a format that is more easily consumed by the IdM system. The IdM system then consumes the pull-script generated pre-processed data and creates (or updates) the local instances of Puhuri Users and Puhuri Projects in CSC's IdM system.

During local instance creation, the imported objects are extended with auxiliary attributes, which are a part of CSC's IdM systems schema for Users and Projects. These auxiliary attributes include, among other things, the imported Puhuri Users local CSC username and the imported Puhuri Projects local CSC identifier. These local identifiers are needed by Puhuri Users and Puhuri Project members to access and use LUMI services.

After the local instances have been created in CSC's IdM system, the IdM system provisions the imported Puhuri Users and Puhuri Projects to CSC's LDAP cluster, from where the LUMI services consume the imported User, Project and Allocation data, by querying the relevant information using LDAP protocol. CSC's HPC team have implemented their own scheduled pull-scripts, which are used to pull in the relevant User and Project information from CSC's LDAP cluster to LUMI.

CSC's IdM system regards Puhuri Core as the authoritative source of information with regard to Puhuri imported Users, Projects and Allocation data, and updates the imported objects local instance in CSC IdM accordingly, when the imported objects state has changed in Puhuri Core.

Table of data that the pull-script reads from Puhuri Core

Puhuri User	**Description**
uuid	Users UUID in Puhuri Core
lastImportDate	Current date in ctime format, generated by import script
puhuriUserUniqueId	Users CUID in Puhuri Core
givenName	Users forname
surname	Users surname

mail	Users email address
telephoneNumber	Users telephone number (if available)
schacHomeOrganization	Users home organizations domain name (if available)
isActive	Accounts status in Puhuri Core (for future use)
sshPublicKeys	Multi-value, Users ssh public keys
eduPersonScopedAffiliations	Multi-value, Users scoped affiliations

Puhuri Project	**Description**
uuid	Projects UUID in Puhuri Core
lastImportDate	Current date in ctime format, generated by import script
title	Projects title
description	Projects description
types	Projects type(s), parsed from Allocation(s)
startTimestamp	Parsed from initial Allocation
endTimestamp	Parsed from Project, Allocation or generated based on Project type
allocatedCPUHours	Parsed from active Allocation(s) and summed up if there are multiple allocations
allocatedGPUHours	Parsed from active Allocation(s) and summed up if there are multiple allocations
allocatedStorageHours	Parsed from active Allocation(s) and summed up if there are multiple allocations
principalInvestigatorUniqueId	Single-value, PI’s CUID
proxyPrincipalInvestigatorUniqueIds	Multi-value, CO-PI’s CUID(s)
memberUniqueIds	Multi-value, PI + CO-PI(s) + Member(s) CUID(s)

scienceAreaCode	Science area code of Project, parsed from Project details or from Allocation details
allocatorName	Parsed from initial Allocation. Contains Allocators label e.g. SNIC, CSC
allocatorCountry	Parsed from initial Allocation. Contains Allocators country e.g. se, fi

The second scheduled script is used to publish information from CSC's IdM system to Puhuri Core. This push-script is used to publish Puhuri Users local CSC usernames and Puhuri Projects local CSC identifiers to Puhuri Core. Puhuri Core SDK provides ready-made methods[8] for publishing these local identifiers to Puhuri Core in an easy and consistent way. National Portals are then able to consume the local identifier data by querying it directly from Puhuri Core. This enables National Portals to display these local identifiers to their end-users in a way they see fit.

Puhuri Project local identifier at CSC	**Puhuri UUID for Project in Puhuri Core**
project_42600000	9ad4aad3c20c4f55a950775a0ce2b295
project_42600001	e41dc386d8884c798aba26039f588f30

Puhuri User local CSC username	**Puhuri UUID for User in Puhuri Core**
johndoe	91c9a4d3fa9e46197629abc392eff85f
janedoe	fd99ec56cbb94d8c905f8379b3628627

User and Project life-cycle management needs to be defined in policy level and then implement for example termination of the project process with data removal.

8 https://puhuri.neic.no/SDK%20guide/allocation-management-sp/#updating-resource-allocation-with-local-reference-setting-backend_id-field

5.1. User, Project and Allocation data import process

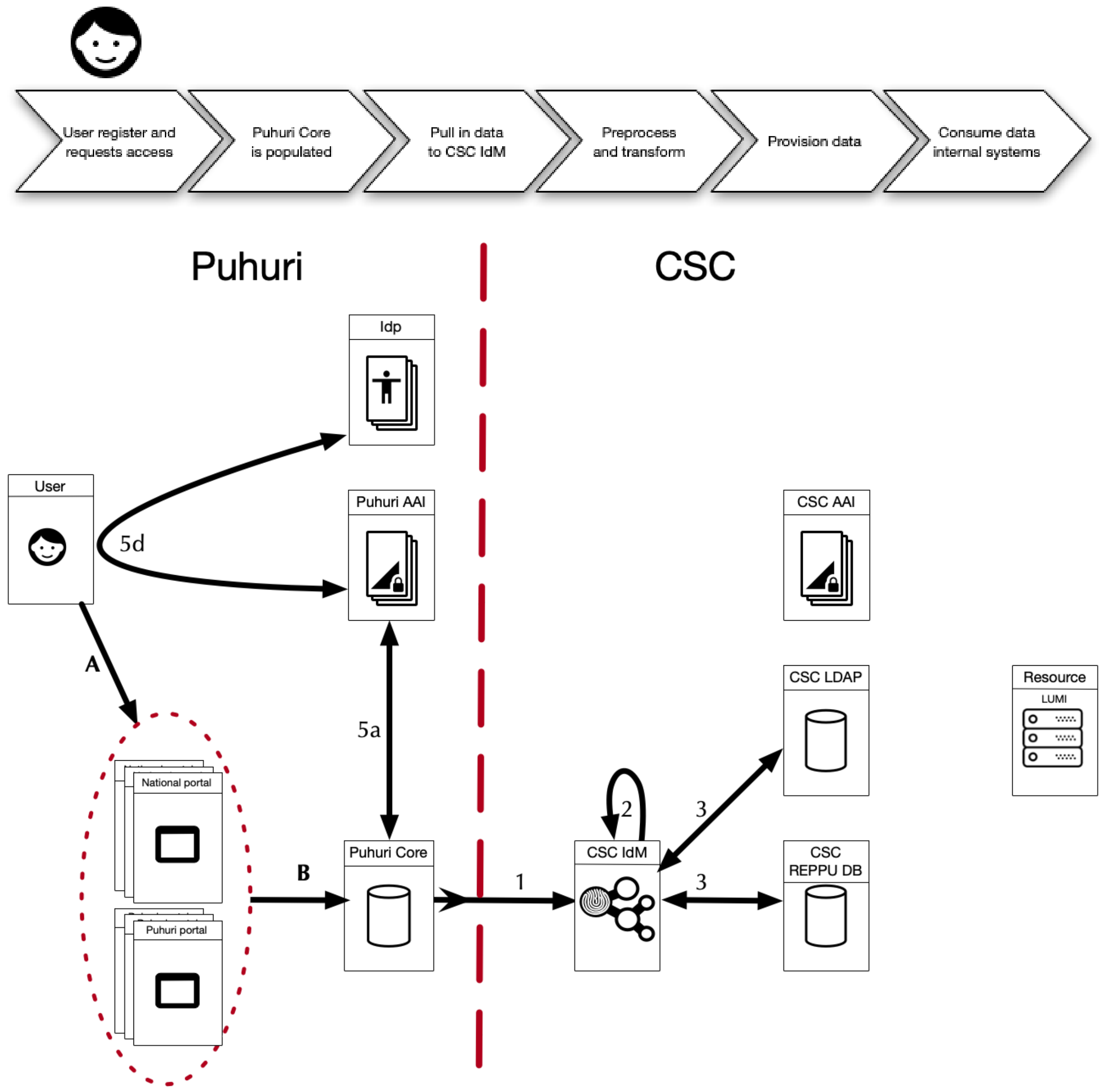

Figure 3. User registration process for LUMI. User, Project and Allocation data import procedure. When User has registered, Puhuri Core will get information from Puhuri AAI Registration Service (step 5a). This data, and possible additional data, such as User provided ssh public keys, are then propagated to CSC IdM (step 1).

The data import procedure outlined in Figure 3 consists of the following steps:

A) User registers, requests, and is granted resources
5)
  5a) Authenticate User (initiation of the authentication workflow)
  5d) User uses his/her home institution's IdP to authenticate (User attributes are returned to Puhuri Core)
    In addition to the IdP provided attributes, User can give additional attributes such as ssh public keys
    During the login process, the User accepts LUMI specific Terms of Use
B) Information on User, Project and Allocation is propagated to Puhuri Core
1) Users, Projects and Allocations are pulled in from Puhuri Core REST API to CSC IdM (script, cron job)
2) The pull script pre-process and transform imported data to comply with CSC's internal data model
  Local instances of User and Project are created in CSC IdM during CSC IdM import
  CSC IdM sends welcome email to User which contains information regarding Users local CSC user id
3) CSC IdM provisions imported local instances of User and Project to target systems

When interfacing with Puhuri Core REST APIs, the Waldur projects waldur_client.py[9] can be used as a good starting point, when developing programs or scripts, which publish or consume data from Puhuri Core.

[9] https://github.com/opennode/ansible-waldur-module/blob/6679b6b8f9ca21099eb3a6cb97e846d3e8dd1249/waldur_client.py

5.2. LUMI User authentication process

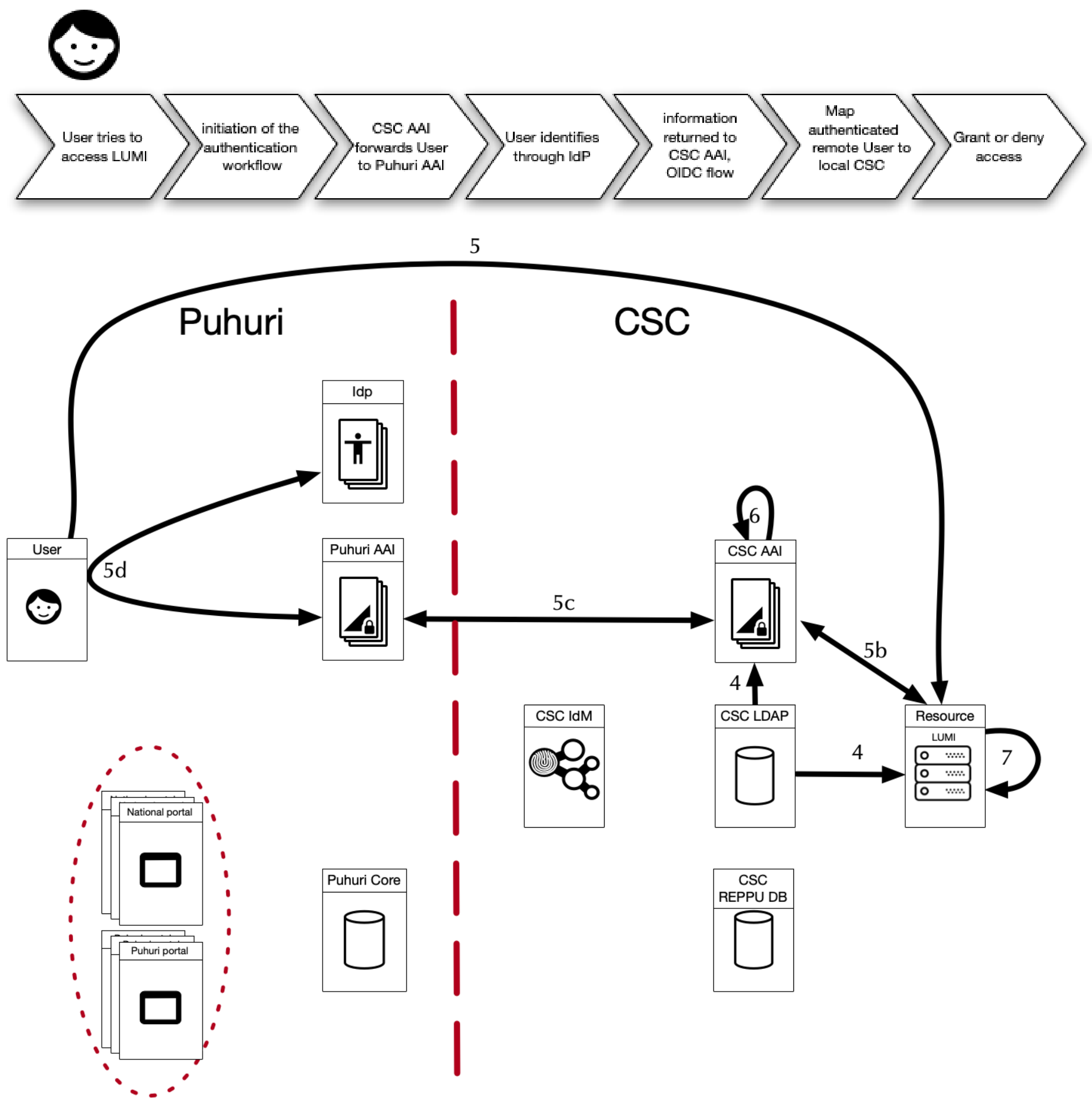
User tries to access LUMI
initiation of the authentication workflow
CSC AAI forwards User to Puhuri AAI
User identifies through IdP
information returned to CSC AAI, OIDC flow
Map authenticated remote User to local CSC
Grant or deny access
5
Puhuri
CSC
Idp
User
5d
Puhuri AAI
5c
6
CSC AAI
5b
4
CSC IdM
CSC LDAP
4
Resource
LUMI
7
National portal
Puhuri portal
Puhuri Core
CSC REPPU DB

Figure 4. User authentication flow for the LUMI services. The Users can authenticate to the LUMI services using the OIDC device flow or an ssh public key based authentication.

5.2.1. Authentication flow when using OIDC device flow

The authentication flows outlined in Figure 4 consists of the following steps when using OpenID Device Code workflow, which is being investigated as a login option :

4) CSC AAI and LUMI Services consume the imported local instances of User and Project from CSC LDAP

5) User tries to establish ssh session with LUMI using his/her CSC provided local user id

5b) User is instructed by LUMI ssh service to authenticate using web browser
User opens provided hyperlink in web browser or scans provided QR code
Users web browser is redirected to CSC AAI

5c) User selects Puhuri AAI login method in CSC AAI

5d) User uses his/her home institution's IdP to authenticate
Users attributes are returned to CSC AAI

6) CSC AAI attempts to map authenticated remote User to local CSC identity
CSC AAI queries CSC LDAP using Puhuri AAI provided unique User identifier (CUID)
If a match is found in CSC LDAP, and the matched local user id is not disabled in CSC LDAP, the web browser
prompts the User to Approve or Deny the ssh access request

7) LUMI grants or denies access based on CSC AAI authentication result and imported Users and Projects state in
CSC LDAP

5.2.2. Authentication flow when using ssh public keys

4) CSC AAI and LUMI Services consume the imported local instances of User and Project from CSC LDAP.

5) User tries to establish an ssh session with LUMI using his/her CSC provided local user id while requesting an ssh key based authentication.
Steps **5b), 5c), 5d)** and **6)** are not performed when using the ssh key based authentication with LUMI.

7) LUMI reads Users ssh public keys from LDAP and/or LUMI file system and performs ssh key based authentication.
LUMI grants or denies access based on ssh public key authentication result and imported Users and Projects state in CSC LDAP.

The Resource provider may have additional processes to authorise the first login, which are not feasible or wise to implement in earlier processes. For example, those can be very service specific processes, such as collecting complementary User information, that is required by the Resource provider, but is not directly available via the Puhuri integration. A Resource may also implement a multi factor authentication in addition to the authentication flow described above.

5.3. LUMI Accounting data export process

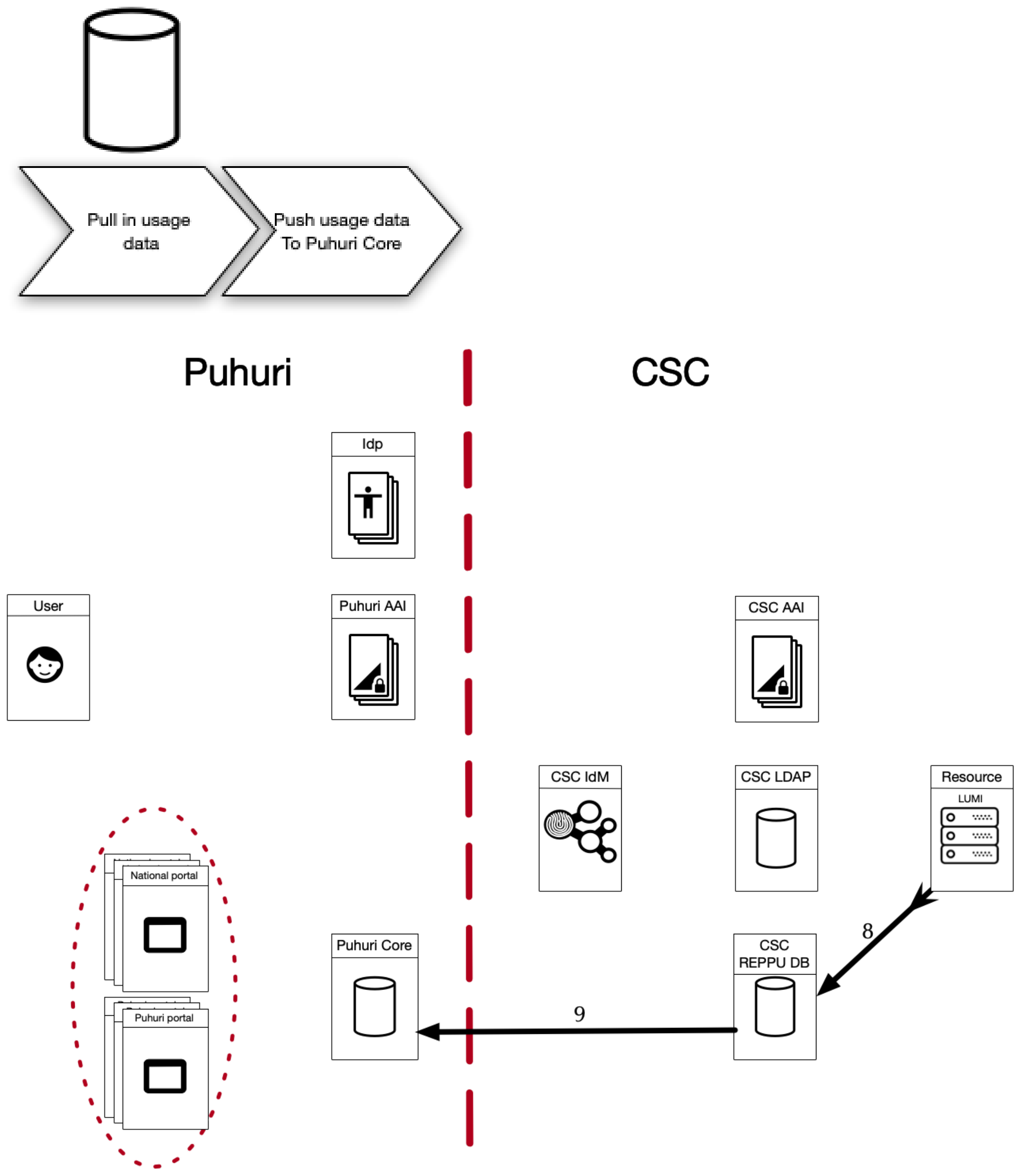

Figure 5. Accounting data export procedure from LUMI Resource to Puhuri Core.

The accounting reporting steps are:
8) Pull in Resource usage data from LUMI services to CSC's reporting database (REPPU)
9) Push Resource usage data from REPPU database to Puhuri Core (matches Projects in Puhuri Core using Puhuri Project UUID)

References

[D1] Requirements and architecture plan for Puhuri. Puhuri Deliverable 1
https://zenodo.org/record/4288776#.YFytZ9yxVhF

[D2] Implementation of Puhuri core functionalities. Puhuri Deliverable 2
https://zenodo.org/record/4727686

[LUM] LUMI EuroHPC supercomputer web page. https://www.lumi-supercomputer.eu/

[PUH] Puhuri Documentation Portal https://puhuri.neic.no