Software Requirements Specification

for

Gaia-X Federation Services

Integration & Portal Orchestration

Published by

eco – Association of the Internet Industry (eco – Verband der Internetwirtschaft e.V.)
Lichtstrasse 43h
50825 Cologne
Germany

Table of Contents

List of Figures

List of Tables

1. Introduction

To get general information regarding Gaia-X and the Gaia-X Federation Services please refer to [TAD] and [PRD].

1.1. Definitions, Acronyms and Abbreviations

Deployment technology: The name of technologies that can be employed to make a Gaia-X Service run. The technology called “Kubernetes” would for example refer to deploying any workload on a Kubernetes cluster. “Ansible” would be the technology that supports Ansible playbooks. “OpenStack” would refer to the technology that can create any OpenStack resources (VM, volumes…). The name refers to the technology used to deploy the service, not the technology of the deployed service. For instance, the “Ansible” technology can deploy services with the “Ansible” technology on an OpenStack infrastructure, or Kubernetes. However, to deploy the service, Ansible is still needed.

Deployment instructions: The full set of commands to make a specific Gaia-X Service run. This is fetched from the PPR. It can be a Terraform template, a Kubernetes manifest file, an OpenStack Heat template.

Deployment settings: The default parameters of a Service set by its PPR in the service’s Self-Description.

PCR configuration: The parameters set by the PCR in the Portal. They will be merged with the deployment settings to create the deployment configuration.

Deployment configuration: The parameters for the final deployment, merged from the ones specified in the PCR configuration and the deployment settings. The former overrides if necessary the defaults set in the latter.

LCM: Life Cycle Management. For a resource, it consists of its creation, its update, its deletion and the reading of its current state.

LCM Engine: Service which belongs to the Portal. During the life cycle management of a Gaia-X service, it acts as the interface between the Portal, the LCM services and the PPR.

LCM Service: Gaia-X service that can be leveraged by the LCM to actually manage (CRUD) Gaia-X Services.

PCR: Participant Consumer Role

PPR: Participant Provider Role

PR: Participant Role

1.2. References

[TAD] Gaia-X European Association for Data and Cloud, AISBL (2021): Gaia-X Architecture Document. As of March 2021.
Please refer to annex “Gaia-X_Architecture_Document_2103”

[TDR] Gaia-X Federation Services Technical Development Requirements
Please refer to annex “GXFS_Technical_Development_Requirements”

[NF.SPBD] Gaia-X Federation Service Non-functional Requirements Security & Privacy by Design.
Please refer to annex “GXFS_Nonfunctional_Requirements_SPBD”

[PRD] Gaia-X European Association for Data and Cloud AISBL (2021): Gaia-X Policy Rules Document.
Please refer to annex “Gaia-X_Policy Rules_Document_2104”

1.3. Document Purpose

The Orchestration Services SHALL NOT be seen as a core service of Gaia-X. They represent together with the Gaia-X portal, the API Framework, Workflow Engine and the core GAIA-Services (Identity and Trust, Federated Catalogue, Sovereign Data Exchange, Compliance[1]) an implementation prototype. Main audience for this document are attendees of the public tender. The core feature of the Orchestration Services is to deal with the Life Cycle Management of Gaia-X Services.

1.4. Product Scope

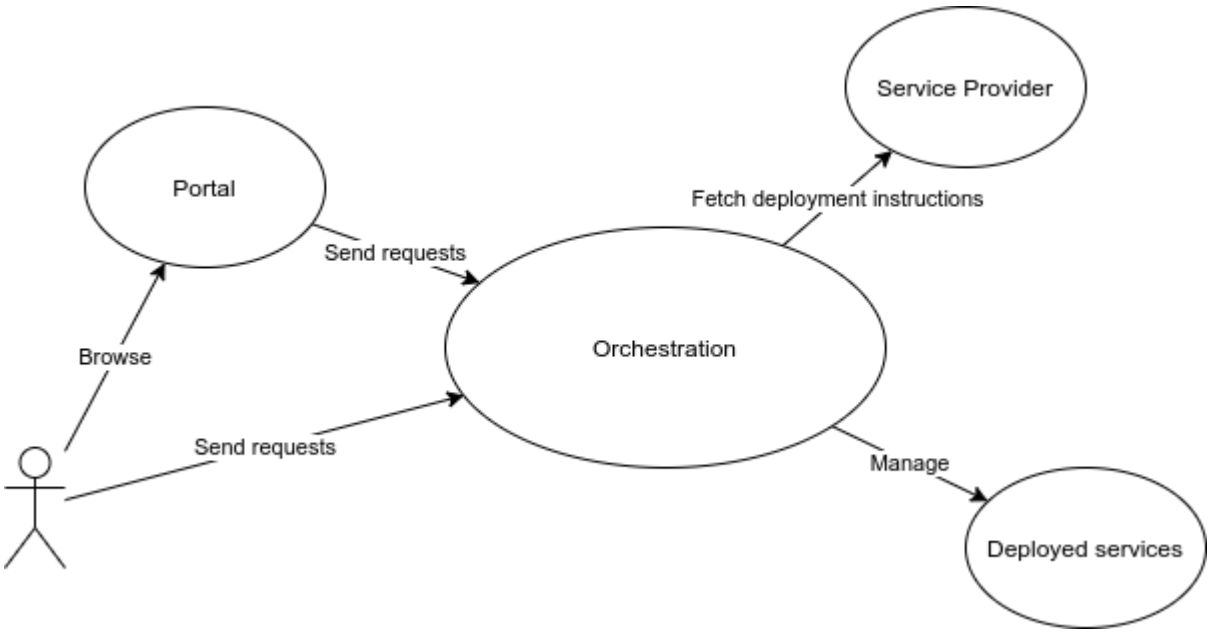


***Figure 1**: Scope of Orchestration*

The Orchestration is responsible for the instantiation and management of Services. It takes care of what is happening after a Gaia-X PR has selected a Service or created a Composite Service in the Portal

[1] Please refer to appendix B for an overview and explanation of the Work Packages (WP)

and wants to use this Service. The Orchestration can deploy, update, and delete Services. Those processes can be triggered by the PCR via the Portal (for instance the selection of a Service in the Portal), or by the PPR (for instance triggering an update of the Service to the latest version for security / compatibility reasons). The Orchestration also provides feedback to the Gaia-X PCR and to the PPR about the Service status (for instance after a successful deployment).

The Orchestration should be accessible through the Portal or by a PR directly.

The PPR must describe how to manage the provided Service. As a wide range of technology exists in order to deploy and manage Service (for instance, and depending on the use case, one could use Terraform, Ansible, Kubernetes, ...), the Orchestration does not provide an actual implementation of the deployment and management methods for each of these technologies. Instead, the Orchestration provides an API standard that has to be followed, in order to implement a "Life Cycle Management (LCM) Service". A LCM Service is a specialized component used to deploy and manage Services supporting the corresponding deployment technology. LCM Services are standard Gaia-X Services, that are provided by a PPR. The Orchestration also provides a central "LCM Engine", responsible for orchestrating the different LCM Services, interfacing with the Portal, and communicating with the LCM Services using the aforementioned API Standard.

1.5. Document Overview

The document describes the product perspective, functions, and constraints. It furthermore lists the functional and non-functional requirements and defines the system features in detail. The listed requirements are binding. The keywords MUST, MUST NOT, SHOULD, SHOULD NOT, MAY, corresponding to RFC 2119 [RFC 2119], are written in capital letters.

2. Orchestration

2.2. High-level overview

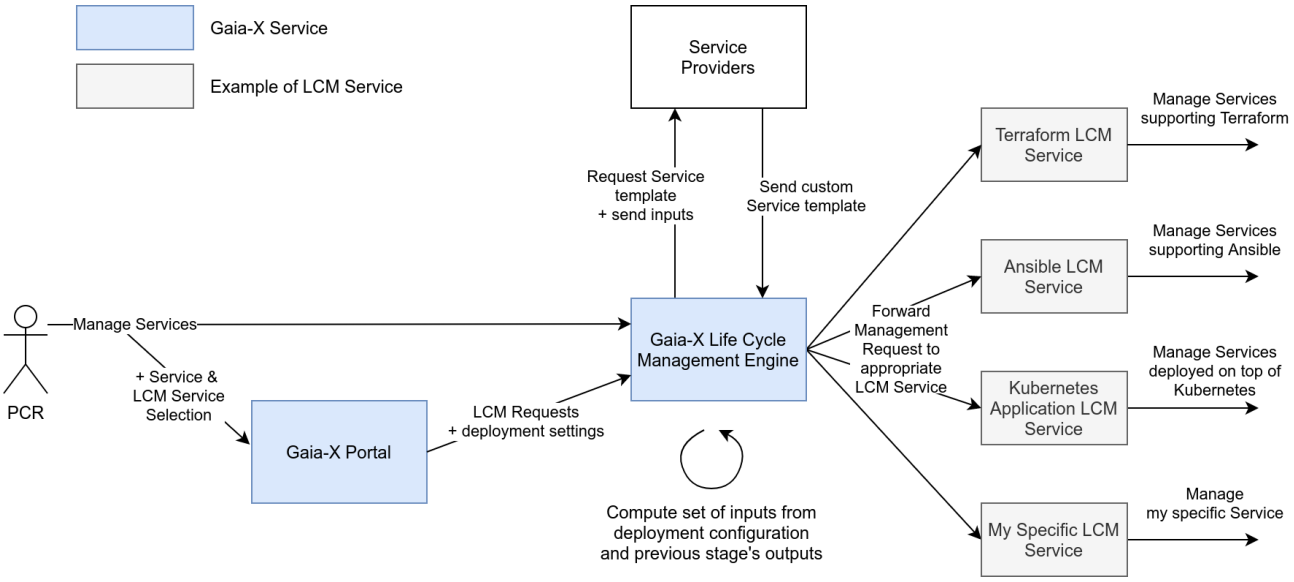


***Figure 2**: High level overview Orchestration*

The diagram above presents the communication between the different actors of the Orchestration context.

As explained in the Scope section, the PCR can manage its services. To do so, he/she has either the possibility to do it directly by communicating with the LCM Engine, or through the Gaia-X Portal, where the UI supports the communication with the Engine.

The LCM Engine can then fetch the deployment instructions from the PPR, prepare the deployment configuration from the parameters provided by the PCR and PPR, then hand it over to the LCM Services.

2.3. LCM Service matching

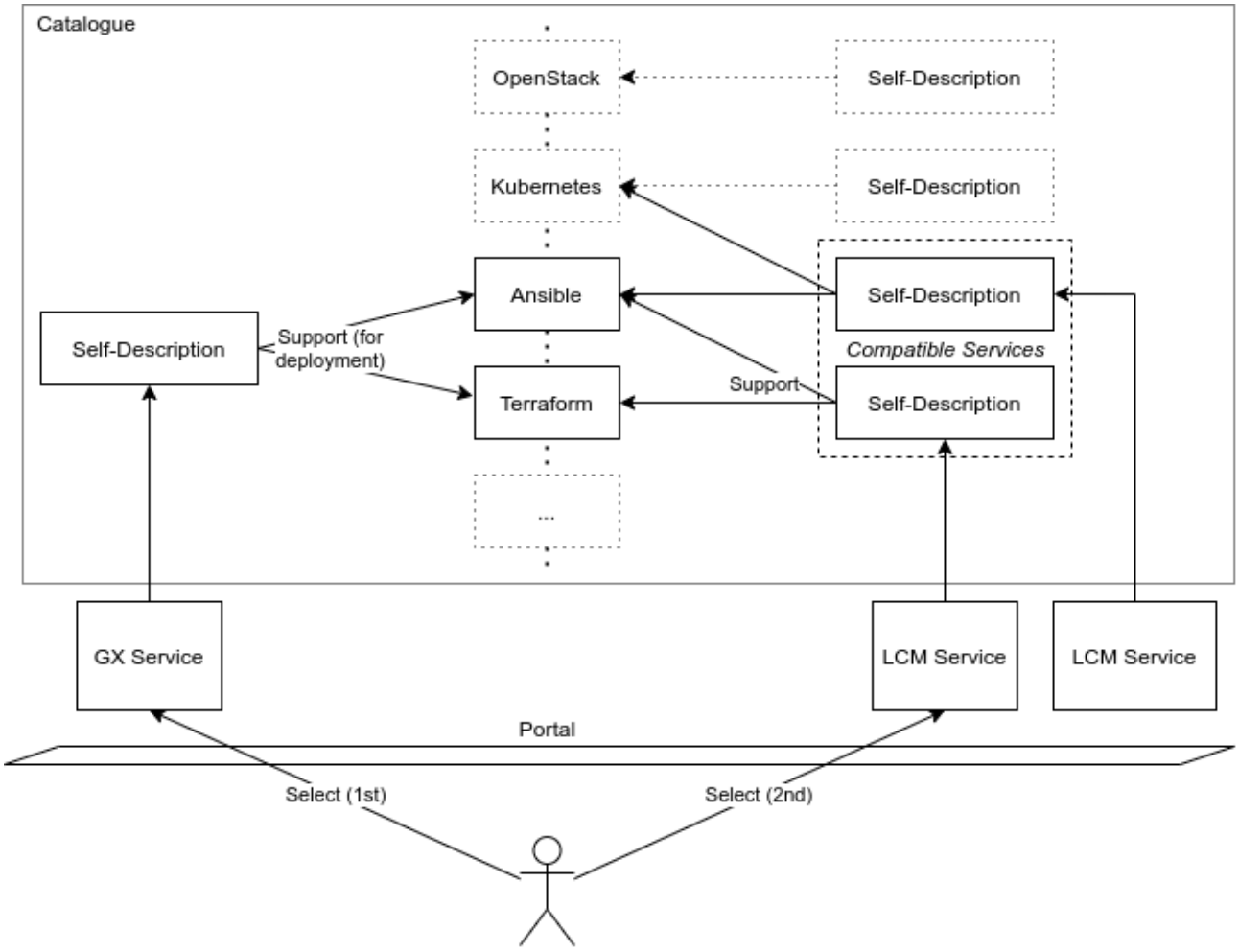


***Figure 3**: LCM Service matching*

During the selection of one or several Gaia-X Services (service composition), the PR selects the LCM Service responsible for the management of the Service. The Portal thus prepares a list of compatible LCM Services, and lets the PCR select the ones he/she wants to use.

This list is generated using the Deployment Technology as a filter: in the Self-Description of a Gaia-X service, a list must define all technologies which can manage the service (the so-called deployment technologies). In its Self-description, the LCM Service also specifies the list of supported Deployment Technologies. Thus, the Portal can generate a set of all LCM Services which have at least one match in their list. The PCR can finally select one element from this set.

2.4. Configuration merging with outputs

PRR must provide the list of necessary inputs, needed for the deployment of the Service. They may provide a default value. Before starting the deployment of a Service, the Deployment Configuration must be prepared. The LCM Engine provides enough information to allow the Portal to create a form. This allows the PCR to set the necessary parameters and overwrite the ones with a default value if necessary. The result will become the deployment configuration.

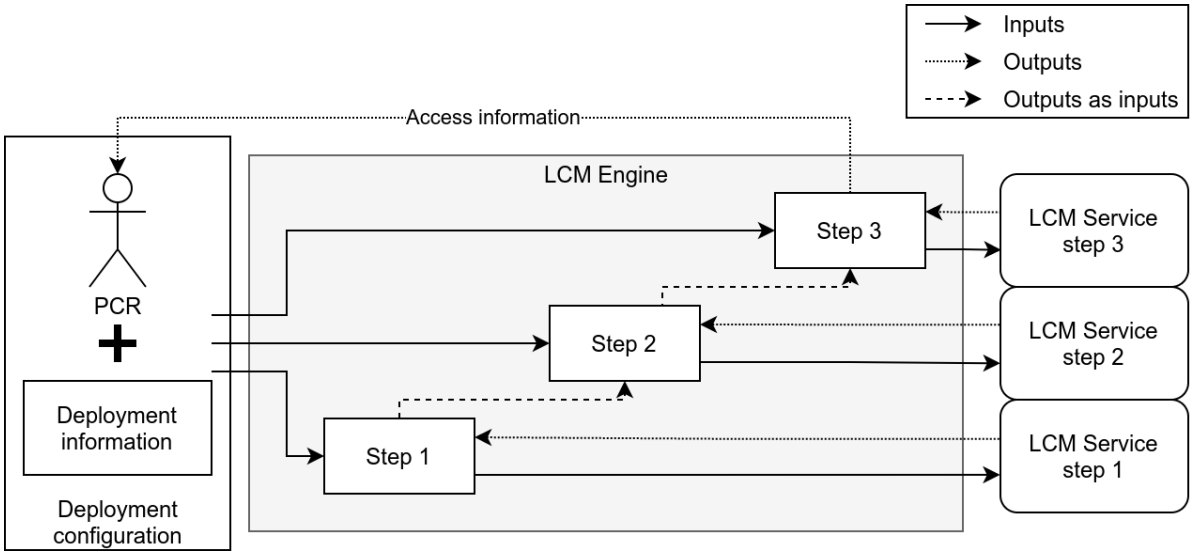


***Figure 4**: Deployment Configuration*

The Deployment Configuration is then passed to the LCM Engine. Consider a deployment separated into 3 parts, where step 2 depends on the step 1, and the step 3 depends on the step 2. The deployment proceeds as follow:

1. The first step needs parts of the configuration. The LCM Engine hands it over to the first LCM Service. This LCM outputs several elements, for instance some access information, to allow the second step to be deployed;
2. For the second step, the LCM Engine takes parts of the deployment configuration, adds the outputs of the first step, then hands over the result to the LCM Service for the second step;
3. The same process is used for the third step;
4. The output of the third step is used as final output from the process. It is given to the PCR. If additional outputs of the other steps have been defined, they can also be handed over to the PCR at the end of the process.

2.5. Responsibilities of the components

The responsibilities of several components in the context of Orchestration will be described next.

The LCM Engine responsibilities are as follow:

- accept requests from the Portal to deploy, update, delete a service or get its status. The LCM services selected are sent along;
- process the content of the requests to be suitable for the LCM services;
- send the processed contents to the necessary LCM services as requests;
- process the result of the deployment and the original configuration to create the deployment configuration;
- automatically send additional requests to LCM services if several steps for the deployment are necessary;
- store the deployment configuration and the deployment instructions of the managed services.

The responsibilities of the LCM services are as follow:

- process requests from the LCM Engine to deploy, update, delete a service or get its status;
- actually deploy the resources from the configuration inside a request to deploy;
- fetch the current state of the deployed services;
- send the current state to the LCM Engine;
- update services;
- delete services;

The responsibilities of the Portal for the Orchestration context:

- given the SD file of a Service, can provide a list of compatible LCM services;
- allow the user to select one from this list;
- provide an interface to deploy, update, delete or get the status of all deployed services;
- send requests to the LCM Engine to deploy, update, delete a service or get its status. The LCM services selected are sent along;
- display the current status of a service

2.6. APIs to develop

For the Orchestration, three different APIs must be developed to allow the LCM Engine to function as expected:

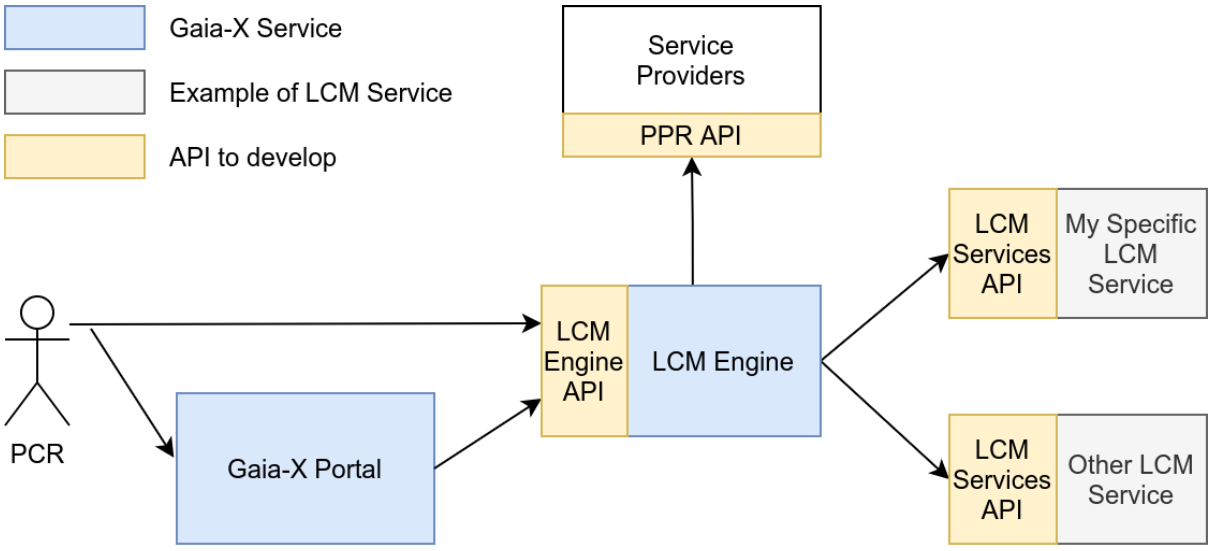


***Figure 5**: Orchestration APIs*

1. the **LCM Engine API**: responsible for the management of LCM Engine by the PCR, to manage the services of the latter. It can also be leveraged by the Portal;
2. the **PPR API**: responsible for the fetching of the deployment instructions from the PPR by the LCM Engine. Any PPR who wants to provide deployment instructions to the PCR must have a service that follows this API. An endpoint of this service will have to be available to the LCM Engine to fetch the deployment instructions. The latter will also only send requests following this API to the PPR.
3. the **LCM Services API**: responsible for the management of the LCM Services by the LCM Engine. Any PPR who wants to add an LCM Service must have this service follow the API to be

compatible with the LCM Engine. The latter will also only send requests following this API to the LCM Services.

2.7. User Story / Example scenarios PPR

1. Prepare an endpoint for fetching the deployment instructions (a TOSCA template or another format)
2. Add the SD file to the catalogue
3. (optional) create a LCM service dedicated to the provided service

2.7.1. Single-service deployment

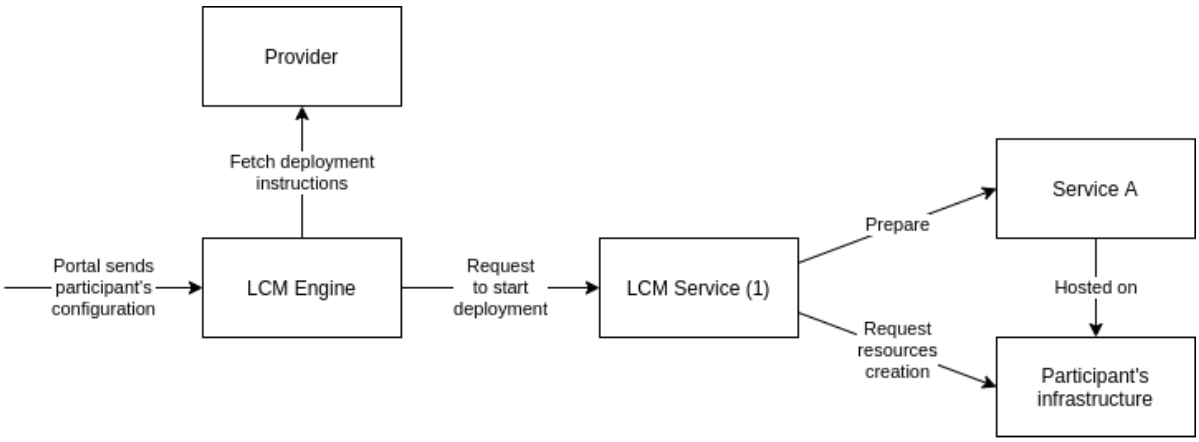


***Figure 6**: Single service deployment*

A simple deployment workflow for a Gaia-X service A, deployed by the LCM service (1) on the infrastructure of the Gaia-X PR would be:

1. the PCR selects the service **A**;
2. like all Services, **A** can only be deployed by some compatible LCM services, but not by all. A list of compatible LCM services is generated by the Portal;
3. the PCR selects the service **(1)**;
4. from an endpoint of the PPR itself, the deployment instructions are retrieved by the LCM Engine for the Service **A**;
5. the PCR is asked to provide additional information for the configuration of the deployed Service for **A**, for example:
    - credentials to the infrastructure of the PR,
    - name to give to the deployed Service;
6. the LCM Engine merges the deployment settings and the PCR configuration into the "deployment configuration";
7. the LCM Engine hands over the result to the LCM service **(1)**;
8. the LCM service **(1)** deploys the Service **A** using the deployment configuration and the deployment instructions;
9. an endpoint is given by the LCM service **(1)** to the LCM Engine, which makes it available to the PCR (on the portal/via mail/…)

2.7.1.1. Service-composition deployment

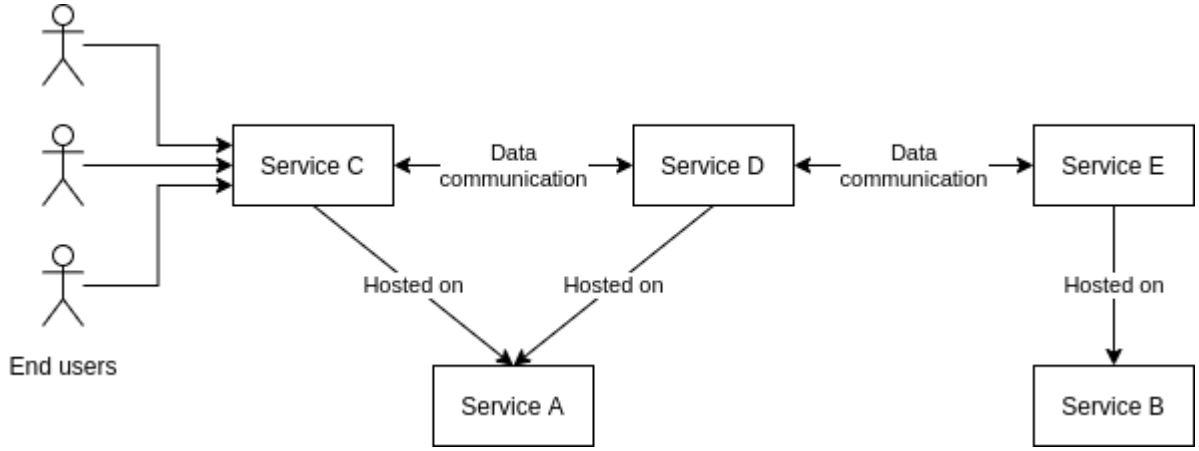


***Figure 7**: Service-composition deployment*

The following workflow is for a service composition. The solution package has the structure described in the above diagram. **C** is the only service accessible externally. Each Gaia-X service (letter: **A**, **B**, **C**...) will be deployed by a compatible LCM service (number: **(1), (2), (3)**...).

The deployment will be done in 2 phases: the first layer, which contains the **A** and **B**, and the second layer, with **C**, **D** and **E**

1. the PCR selects the solution package.
2. For each service:
   2.1. a list of compatible LCM services is generated;
   2.2. the PCR selects from the list the LCM service that will be leveraged (**(1)** for **A**, **(2)** for **B**...);
   2.3. from an endpoint of the PPR itself, the deployment settings is retrieved by the LCM;
3. the PCR is asked to provide additional information (credentials, name to give to the deployed Services, etc.) for the configuration of the deployed Services
4. the LCM merges the deployment settings and the PCR configuration into the new **deployment configuration**;
5. 1st layer:
   5.1. the LCM hands over the result to the LCM services **(1)** and **(2)**:
   5.2. the LCM service **(1)** and **(2)** deploy the service **A** and **B**;
6. the LCM Engine merges the output of deployment by **(1)** and **(2)** to the original **deployment configuration;**
7. 2nd layer:
   7.1. the LCM hands over the result to the LCM services **(3)**, **(4)** and **(5)**:
   7.2. the LCM service **(3)**, **(4)** and **(5)** deploy the service **C**, **D** and **E**;
8. the endpoints are given by the LCM service **(3)** to the LCM, which makes it available to the PCR (on the portal/via mail/...), as: "**C** is the only service accessible externally"

2.7.1.2. Deployment of new access

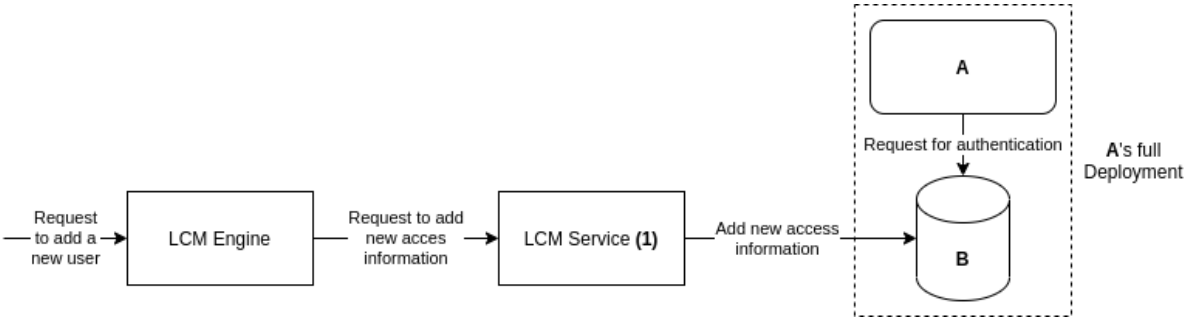


***Figure 8**: Deployment of new access*

The following user story considers a service which is already deployed and managed by its PPR. When selecting the corresponding Gaia-X service (**A**), the Orchestration will only add a new access to the service through its authentication service (**B**), and not deploy a new one. For this, the PPR also makes a LCM service available, **(1)**, which is dedicated to adding a new user inside the service.

1. the PCR selects the service **A**;
2. A list of compatible LCM services is generated by the Portal: only the one provided by the PPR can be used, **(1)**;
3. the PCR selects the LCM service **(1)**;
4. No endpoint is needed to retrieve deployment instructions;
5. the PCR is asked to provide additional information (credentials, contact information, ...) for the configuration of the added access;
6. the LCM Engine merges the deployment settings and the PCR configuration into the “deployment configuration”;
7. the LCM Engine hands over the result to the LCM service **(1)**;
8. the LCM service **(1)** creates new access information inside the internal service **B** using the deployment configuration;
9. the access information is given by the LCM Service **(1)** to the LCM Engine, which makes it available to the PCR (on the portal/via mail/...).

2.7.2. Examples for the user stories

2.7.2.1. Single-service deployment

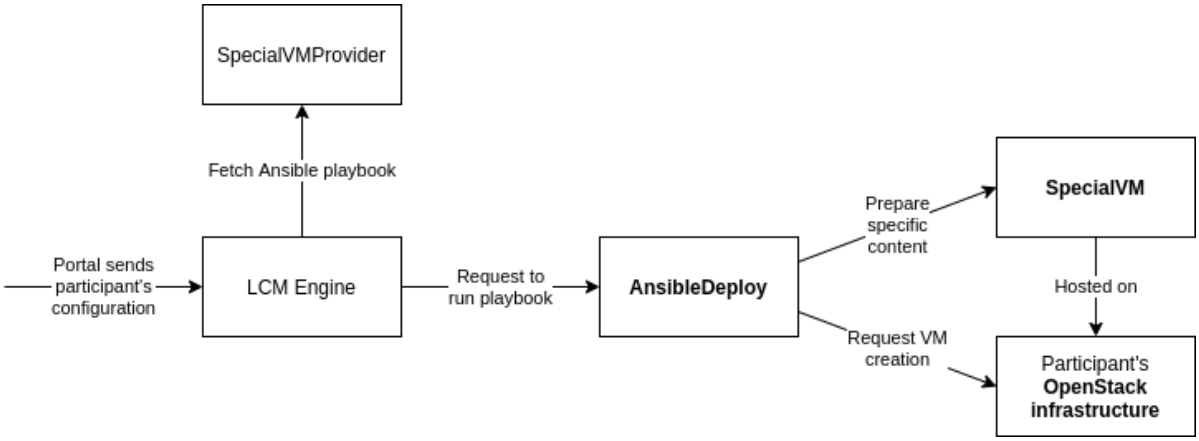


***Figure 9**: User story example single service deployment*

SpecialVM is a Gaia-X service which consists of the deployment of a VM with proprietary content, which is why a Gaia-X PR needs to book it. This VM can be deployed using Ansible, to create the machine itself, and download and prepare its content. So the LCM service in this case is a service that can support Ansible playbooks. Let's use one called **AnsibleDeploy**, provided by another entity. The Gaia-X PR wants to deploy it on its own **OpenStack infrastructure**.

A simple deployment workflow for **SpecialVM**, deployed by the **AnsibleDeploy** service would be:

1. the PCR selects the service **SpecialVM**;
2. like all Services, **SpecialVM** can only be deployed by some compatible LCM services, but not by all. A list of compatible LCM services is generated by the Portal;
3. the PCR selects the service **AnsibleDeploy** from the list;
4. from an endpoint of the PPR itself, the Ansible playbook are retrieved by the LCM Engine for the Service **SpecialVM**;
5. the PCR is asked to provide additional information for the configuration of the deployed **SpecialVM**:
    a. credentials to an OpenStack project,
    b. name to give to the deployed VM
6. the LCM Engine merges the deployment settings and the PCR configuration into the "deployment configuration";
7. the LCM Engine hands over the result to **AnsibleDeploy** service;
8. the AnsibleDeploy service deploys the **SpecialVM** using the deployment configuration and the deployment instructions (retrieved Ansible playbook);
9. an endpoint is given by the **AnsibleDeploy** service to the LCM Engine, which makes it available to the PCR (on the portal/via mail/...)

2.7.2.2. Service-composition deployment

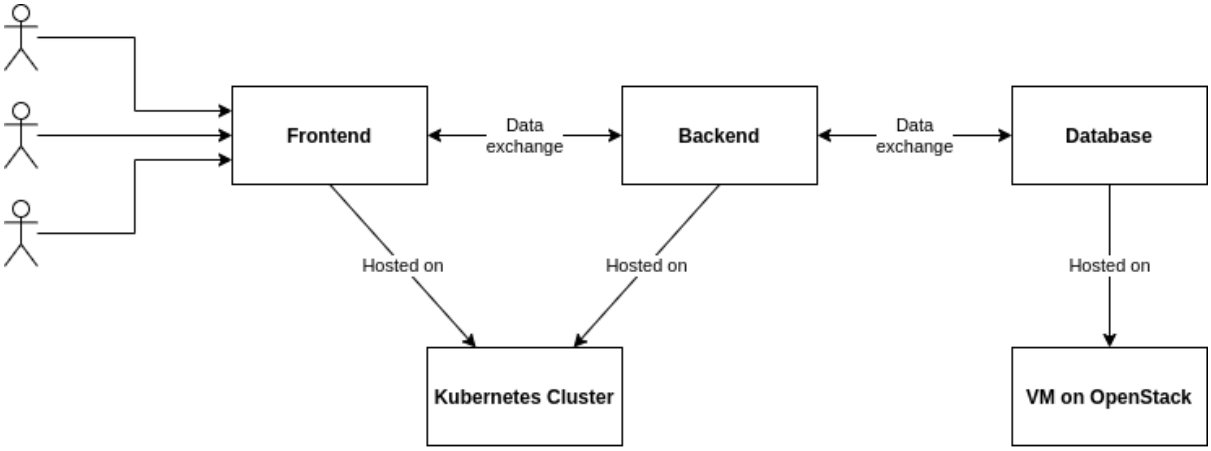


***Figure 10**: User story example service-composition deployment*

The following workflow is for a service composition. The solution package has the structure described in the above diagram. The **Frontend** is the only service accessible externally. Each Gaia-X service will be deployed by a compatible LCM service:

- **VM on OpenStack**: "**OSHeatDeploy**" LCM service
- **Kubernetes Cluster**: already existing, belongs to the PR
- **Frontend** and **Backend:** "**KubeDeploy**" LCM Service
- **Database**: "**SQLDeploy**" **LCM service**

The deployment will be done in 2 phases: the first layer, which contains the **VM on OpenStack**, and the second layer, with the **Frontend**, **Backend** and **Database**

1. the PCR selects the solution package.
2. For each service:
    a. a list of compatible LCM services is generated;
    b. the PCR selects from the list the LCM service that will be leveraged (see list above);
    c. from an endpoint of the PPR itself, the deployment settings is retrieved by the LCM;
3. the PCR is asked to provide additional information for the configuration of the deployed Services, for example:
    a. credentials for the **Kubernetes Cluster**
    b. name to give to the deployed **VM**
    c. **Database** credentials
    d. credentials for the **Backend** admin
    e. ...
4. the LCM Engine merges the deployment settings and the PCR configuration into the new **deployment configuration**;
5. 1st layer:
    a. the LCM hands over the result to the **OSHeatDeploy** LCM services:
    b. the LCM service **OSHeatDeploy** deploy the VM on **OpenStack**;

6. the LCM Engine merges the output of deployment by **OSHeatDeploy** and to the original **deployment configuration;**
7. 2nd layer:
    a. the LCM hands over the result to the LCM services **KubeDeploy** and **SQLDeploy**
    b. the LCM service **KubeDeploy** and **SQLDeploy** deploy the **Frontend**, **Backend** and **Database**
8. The endpoints are given by the **KubeDeploy** to the LCM, which makes it available to the PCR (on the portal/via mail/...), as the **Frontend** is the only service accessible externally.

2.7.2.3. Deployment of new access

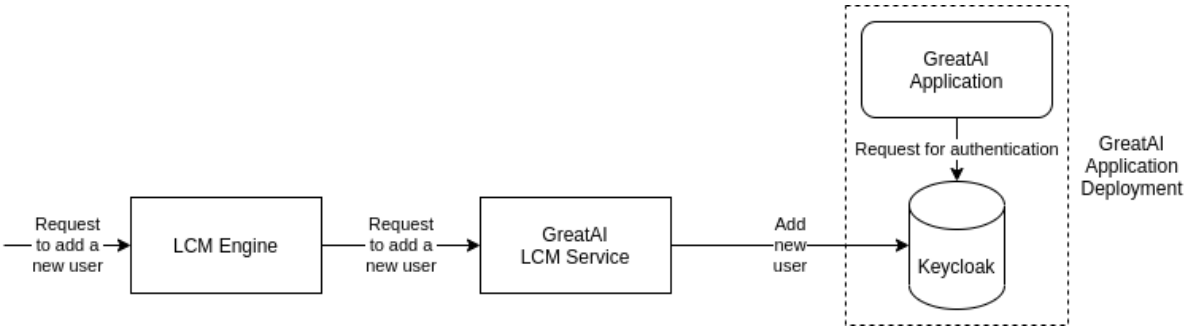


***Figure 11**: User story example deployment of new access user story example*

The following user story considers a service which is already deployed and managed by its PPR. When selecting the corresponding Gaia-X service (**GreatAI Application**), the Orchestration will only add a new access to the service through its authentication service (**keycloak** in this example), and not deploy a new one. For this, the PPR also makes a LCM service available, **GreatAI LCM Service**, which is dedicated to adding a new user inside the service.

1. the PCR selects the service **GreatAI Application**;
2. A list of compatible LCM services is generated by the Portal: only the one provided by the PPR can be used (**GreatAI LCM Service**);
3. the PCR selects the **GreatAI LCM Service** from the list;
4. No endpoint is needed to retrieve deployment instructions;
5. the PCR is asked to provide additional information (credentials,contact information,...) for the configuration of the added access;
6. the LCM Engine merges the deployment settings and the PCR configuration into the “deployment configuration”;
7. the LCM Engine hands over the result to the **GreatAI LCM Service**;
8. the **GreatAI LCM Service** creates new access information inside the **keycloak** service of the Service **GreatAI Application** using the deployment configuration;
9. the access information is given by the **GreatAI LCM Service** to the LCM Engine, which makes it available to the PCR (on the portal/via mail/…)

3. Requirements

Further binding requirements can be found in [TDR].

3.2. Functional requirements

3.2.1. LCM Engine

ID	Description
LE-Std-1	The LCM Engine MUST be able to communicate and deploy Services through any LCM Service, as long as it follows the standard API for the LCM Services.
LE-Std-2	For a given deployment technology, the LCM Engine MUST be able to use any LCM service which supports the technology interchangeably.
LE-CRUD-1	The LCM Engine MUST be able to fetch logs, current state, and access information of the GX Service from the LCM Services.
LE-CRUD-2	The LCM Engine MUST be able to create GX Services using the orchestration instructions provided by the PPR, if provided
LE-CRUD-3	The LCM Engine MUST be able to update GX Services using the orchestration instructions provided by the PPR, if provided
LE-CRUD-4	The LCM Engine MUST be able to delete GX Services
LE-TOSCA-1	The LCM Engine MUST support the deployment of GX Services with orchestration instructions that follow the TOSCA template format.
LE-TOSCA-2	The LCM Engine MAY support the deployment of GX Services with orchestration instructions that follow other template formats that TOSCA.
LE-Conf-1	The LCM Engine MUST be able to merge the default deployment parameters with the parameters set by the PCR.
LE-PPR-1	The LCM Engine MUST be able to fetch the deployment instructions from the PPR
LE-PCR-1	The LCM Engine MUST be reachable by the PCR also without the Portal.
LE-API-1	In the two cases of being managed by the Portal or by the PCR, the LCM Engine MUST be accessible through the same API.

***Table 1**: Functional requirements LCM Engine*

3.2.2. LCM Service API

ID	Description
LS-Std-1	Custom LCM Services MAY be added. Those LCM Services MUST follow the LCM Service API standard.

LS-CRUD-1	The LCM Services API MUST support sending logs, current state, and access information of the GX Service to the LCM Engine.
LS-CRUD-2	The LCM Services API MUST support the creation of GX Services using the orchestration instructions provided by the PPR, if provided
LS-CRUD-3	The LCM Services API MUST support the update of GX Services using the orchestration instructions provided by the PPR, if provided
LS-CRUD-4	The LCM Services API MUST support the deletion of GX Services

***Table 2**: Functional requirements LCM Service API*

3.2.3. PCR

ID	Description
PC-1	The PCR MUST be notified after the successful deployment of a Service.
PC-2	The PCR MAY manage its services through the Portal.
PC-3	The PCR MAY communicate with the LCM Engine through its API directly.

***Table 3**: Functional requirements PCR*

3.2.4. PPR

ID	Description
PP-1	The PPR MUST follow the standardized API if deployment instructions need to be fetched by the LCM Engine.
PP-2	A PPR MUST be able to add his/her own LCM Service to Gaia-X

***Table 4**: Functional requirements PPR*

3.3. Non-functional requirements

ID	Description
LS-Sel-1	A Gaia-X PCR MAY select a LCM Service when selecting a Gaia-X Service.
PP-Std-1	A standardized API MUST be developed to allow the LCM Engine to fetch the deployment instructions from any PPR interchangeably
IInp-1	A PPR MAY specify Default values for the inputs. If no default value is specified, the input MUST be specified by the Gaia-X PCR before triggering the service deployment.
Inp-2	A Gaia-X PCR MAY provide / override inputs for the deployment of Gaia-X Service
Inp-3	Inputs MUST possess metadata allowing the input to be validated.

	The information managed by the LCM Engine and LCM Services MUST only be accessible by the authorized entities (administrator, service owner, possibly PPR)
Scal-1	The LCM Engine MUST support a growing number of requests (scalability).
Scal-2	The LCM Engine MUST be able to support an arbitrary number of LCM services.
InOp-1	Any LCM Service that supports a deployment technology MUST be usable by the LCM Engine interchangeably. It means the LCM Services API MUST support this case.
REST-1	The standard API for the LCM Engine, for the LCM Services and for fetching the deployment instructions from the PPR MUST all be REST APIs.
REST-2	The standard API for the LCM Engine, for the LCM Services and for fetching the deployment instructions from the PPR MUST all support the OpenAPI specification. An OpenAPI file MUST be provided for each.
Doc-1	Documentation MUST be provided for the LCM Engine. It MUST allow developers to understand the content of the code easily and let them start further developing on the Engine. It MUST also allow a user to understand how to start and use the LCM Engine.
Doc-2	Documentation MUST be provided for each of the three different APIs (LCM Engine API, PPR API and LCM Service API).It MUST also allow a user to understand how to develop a client or a server using the respective API specifications.
Doc-3	The documentation MUST follow best practices in the software engineering field, such as keeping language simple, using plain English, explaining technical terms and jargon if they must be used, and making sure that individual needs are catered. Please refer to [TDR] for further requirements regarding documentation.

***Table 5**: Non-functional requirements*

3.4. General Security Requirements

Each Gaia-X Federation Service MUST meet the requirements stated in the document “Specification of non-functional Requirements Security and Privacy by Design” [NF.SPBD].

Appendix A: Glossary

The glossary is part of the Gaia-X Architecture Document [TAD].

Appendix B: Overview GXFS Work Packages

The project "Gaia-X Federation Services" (GXFS) is an initiative funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) to develop the first set of Gaia-X Federation Services, which form the technical basis for the operational implementation of Gaia-X.

The project is structured in five Working Groups, focusing on different functional areas as follows:

Work Package 1 (WP1): Identity & Trust
Identity &Trust covers authentication and authorization, credential management, decentral Identity management as well as the verification of analogue credentials.

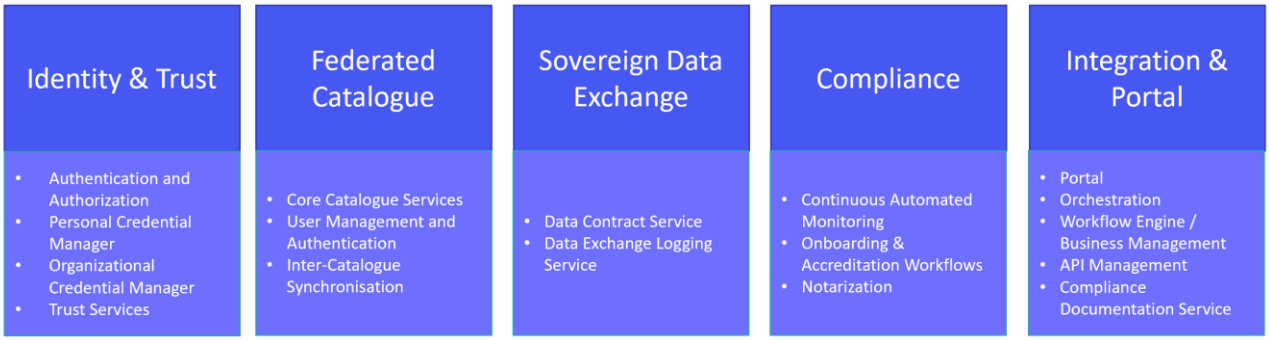
Work Package 2 (WP2): Federated Catalogue
The Federated Catalogue constitutes the central repository for Gaia-X Self-Descriptions to enable the discovery and selection of Providers and their Service Offerings. The Self-Description as expression of properties and Claims of Participants and Assets represents a key element for transparency and trust in Gaia-X.

Work Package 3 (WP3): Sovereign Data Exchange
Data Sovereignty Services enable the sovereign data exchange of Participants by providing a Data Agreement Service and a Data Logging Service to enable the enforcement of Policies. Further, usage constraints for data exchange can be expressed by Provider Policies as part of the Self-Description

Work Package 4 (WP4): Compliance
Compliance includes mechanisms to ensure a Participant's adherence to the Policy Rules in areas such as security, privacy transparency and interoperability during onboarding and service delivery.

Work Package 5 (WP5): Portal & Integration
Gaia-X Portals and API will support onboarding and Accreditation of Participants, demonstrate service discovery, orchestration and provisioning of sample services.

All together the deliverables of the first GXFS project phase are specifications for 17 lots, that are being awarded in EU-wide tenders:



Further general information on the Federation Services can be found in [TAD].