5G-EPICENTRE

5G ExPerimentation Infrastructure hosting Cloud-nativE Netapps for public proTection and disaster RElief

Innovation Action – ICT-41-2020 - 5G PPP – 5G

Innovations for verticals with third party services

D4.3 Curated Network Application image repository

Delivery date: April 2023

Dissemination level: Public

Project Title:	**5G-EPICENTRE - 5G ExPerimentation Infrastructure hosting Cloud-nativE Netapps for public proTection and disaster RElief**
Duration:	**1 January 2021 – 31 December 2023**
Project URL	https://www.5gepicentre.eu/

This project has received funding from the European Union's Horizon 2020 Innovation Action programme under Grant Agreement No 101016521.

www.5gepicentre.eu

Document Information

Deliverable	D4.3: Curated Network Application image repository
Work Package	WP4: Platform integration and VNF development
Task(s)	Task 4.2: Container Network Functions and Network Application repositories
Type	Other
Dissemination Level	Public
Due Date	M28, April 30, 2023
Submission Date	M28, April 27, 2023
Document Lead	Apostolos Siokis (IQU)
Contributors	Kostas Ramantas (IQU) Marianthi Roumba (IQU) Daniel Del Teso (NEM)
Internal Review	Konstantinos C. Apostolakis (FORTH) Hamzeh Khalili (CTTC)

Disclaimer: This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. This material is the copyright of 5G-EPICENTRE consortium parties, and may not be reproduced or copied without permission. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Document history

Version	Date	Changes	Contributor(s)
V0.1	02/12/2012	Creation of JFrog private Helm repo	Marianthi Roumba (IQU)
V0.2	03/06/2022	Completion of implementation Open-API server	Marianthi Roumba (IQU) Apostolos Siokis (IQU)
V0.3	03/06/2022	Completion of implementation Open-API server	Marianthi Roumba (IQU) Apostolos Siokis (IQU)
V04	19/12/2022	Revising of the OpenAPI server code	Apostolos Siokis (IQU)
V0.5	21/02/2023	Section 2 and Appendix	Kostas Ramantas (IQU)
V1.0	31/03/2023	Introduction, Conclusions, Executive Summary. Version preparation for in-ternal review.	Apostolos Siokis (IQU)
V1.1	17/04/2023	Internal Review and Quality Review	Konstantinos C. Apostolakis (FORTH) Hamzeh Khalili (CTTC)
V2.0	27/04/2023	Final Version for submission	Apostolos Siokis (IQU)

Project Partners

Logo	Partner	Country	Short name
	AIRBUS DS SLC	France	**ADS**
	NOVA TELECOMMUNICATIONS SINGLE MEMBER S.A.	Greece	**NOVA**
	Altice Labs SA	Portugal	**ALB**
	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	Germany	**HHI**
	Foundation for Research and Technology Hellas	Greece	**FORTH**
	Universidad de Malaga	Spain	**UMA**
	Centre Tecnològic de Telecomunicacions de Catalunya	Spain	**CTTC**
	Istella SpA	Italy	**IST**
	One Source Consultoria Informatica LDA	Portugal	**ONE**
	Iquadrat Informatica SL	Spain	**IQU**
	Nemergent Solutions S.L.	Spain	**NEM**
	EBOS Technologies Limited	Cyprus	**EBOS**
	Athonet SRL	Italy	**ATH**
	RedZinc Services Limited	Ireland	**RZ**
	OptoPrecision GmbH	Germany	**OPTO**
	Youbiquo SRL	Italy	**YBQ**
	ORamaVR SA	Switzerland	**ORAMA**

List of abbreviations

Abbreviation	Definition
API	Application Programming Interface
CLI	Command Line Interface
CNF	Cloud Native Network Functions
cURL	client URL
DevOps	Development & Operations
FR	Functional Requirements
GA	Grant Agreement
HTTP	Hyper Text Transfer Protocol
IT	Information Technology
JSON	JavaScript Object Notation
K8s	Kubernetes
NFR	Non-Functional Requirements
NS	Network Service
OAS	OpenAPI Specification
QR	Quality Requirements
RBAC	Role-Based Access Control
REST	Representational State Transfer
SBD	Security-By-Design
UI	User Interface
UMLSec	Unified Modeling Language Secure
URL	Uniform Resource Locator

Executive summary

This document presents the 5G-EPICENTRE “Curated Network Application image repository” and corresponds to Task T4.2 “Container Network Functions and Network Application repositories”, and more specifically to sub-Task 4.2.2 “Network Applications provision”.

This deliverable stands as a public report, providing details regarding the implementation details of the Network Service Repository, developed in the project to host storage and manage Network Service and Application artefacts. These artefacts will be packaged and stored in the form of Helm Charts. More specifically, information is provided about the two sub-components that constitute the repository, namely the Private Helm Repo and the OpenAPI server. The latter is used for the communication of the Portal with the private Helm Repo. The former communicates with the Experiment Coordinator.

We first describe how the Network Service Repository adheres to the quality, functional and non-functional requirements. We then elaborate on the adherence to the Security-by-Design Framework and the security considerations of the project’s D1.5. The document then presents the internal architecture of the Network Service Repository and its place in the 5G-EPICENTRE platform architecture. We then proceed to describe how basic interactions, such as viewing the filenames of the Helm Charts stored in the repository, are realized using the OpenAPI server.

This deliverable serves as an accompanying report to the Network Service Repository software.

Table of Contents

List of Figures

List of Tables

1 Introduction

5G-EPICENTRE represents an attempt to federate 5G testbed infrastructures across the EU, with the specific purpose to address the needs and demands of the public safety and emergency and disaster management market needs. Part of the technologies and modules developed in the project is the development of a Network Service Repository, that will be used by application service providers in their experimentation activities. A repository of network application images is a centralized location that stores a collection of pre-configured and pre-packaged images of network applications, which can be easily deployed and run on various platforms or environments. These images contain all the necessary dependencies, libraries, and configurations needed to run the application, making it easier for developers and Information Technology (IT) professionals to deploy and manage complex network applications. The Network Service Repository in the project will host Network Service (NS) and Application artefacts in the form of Helm Charts and, in accordance with D1.4, it will communicate: (i) with the 5G-EPICENTRE Portal, in order to upload and manage Helm charts; and (ii) the Experiment Coordinator, in order to set up and instantiate clusters executing Cloud-native Network Functions (CNFs) over the testbeds.

This deliverable aims at presenting high-level information about the implementation of the Network Service Repository and the technologies used. The Repository is comprised of two sub-components, namely the Private Helm Repo and the OpenAPI server. Both sub-components are described briefly in this document.

1.1 Mapping of project's outputs

The purpose of this section is to map 5G-EPICENTRE Grant Agreement (GA) commitments within the formal Task description, against the project's respective outputs and work performed.

Table 1: Adherence to 5G-EPICENTRE's GA Task Description

5G-EPICENTRE Task	Respective Document Chapters	Justification
T4.2.2: Network Applications provision: *"This sub-task will […] deliver a curated repository of NetApp images that can be used by application service providers in their experimentation activities."*	Section 3 – Network Service Repository	Section 3 presents details about the two sub-components of the Network Service Repository, namely the Private Helm Repo and the OpenAPI server.

1.2 Structure of the Document

In the first Section, we introduced the current deliverable (D4.3), and explained what to expect from its content. In Section 2 we describe how the Network Service Repository adheres to project specifications (requirements and Security-by-Design considerations). Section 3 is the main Section of this deliverable, and it is devoted to a detailed description of the Network Service Repository. In separable sub-sections, each sub-component of the Repository (the Private Helm Repo and the OpenAPI server) is described. Section 4 concludes the deliverable. Finally, the document contains an appendix presenting example responses to the Application Programming Interface (API) calls to the OpenAPI Server.

2 Adherence to Project specifications

In this Section, we explore how the current implementation of the Network Service Repository supports project documentation with respect to its requirements (Section 2.1). We then provide details on the alignment to the provisions of the project Security-by-Design (SBD) Framework (Section 2.2).

2.1 5G-EPICENTRE Requirements adherence

The 5G-EPICENTRE Network Service Repository design has been informed by the requirements specified in deliverables D1.3 and D1.4. We briefly discuss adherence to the project elicited requirements in Table 2. In the Table below, requirements are identified using the codes QR, FR, NFR defined in D1.3. The definition of these codes is given below:

- QR: stakeholders' requirement
- FR: functional requirement
- NFR: non-functional requirement

Table 2: 5G-EPICENTRE Network Application Image Repository adherence to stakeholders' and platform requirements.

Req.	Description	Adherence	Implemented
QR3	The platform should provide network resource repository and the ability to use.	The Network Service Repository can be used to upload, delete, and view the contents of the repository, as well as view the content of the Helm charts stored therein.	Fully Implemented
QR4	The platform should provide VNF / Network Applications repository, and the ability to use.	The Network Service Repository can be used to upload, delete, and view the contents of the repository, as well as view the content of the Helm charts stored therein.	Fully Implemented
QR12	Avoid complexity and over-dimensioning.	The Network Service Repository has been implemented with the purpose of enabling developers and other partners in the project to use it, using simple API calls (implemented in the OpenAPI Server – see Section 3.2).	Fully Implemented
FR12	The system should expose easy-to-consume APIs toward the experimenter.	The OpenAPI server, which is part of the Network Service Repository, exposes APIs in order to upload, delete, view the contents of the repo, view the content of Helm charts. The Private Helm Repo offers APIs in order to communicate with the Experiment Coordinator.	Fully Implemented

FR16	The system should expose a requirements catalogue for the underlying network resources.	Using the OpenAPI server, which is part of the Network Service Repository, the 5G-EPICENTRE Portal user will be able to view the available Helm charts in the repository.	Fully Implemented
FR31	The system could support role-based access control (RBAC) policies.	The 5G-EPICENTRE Network Service Repository implements RBAC by means of user authentication.	Fully Implemented
NFR1	The system must be secure.	The 5G-EPICENTRE Network Service Repository implements RBAC by means of user authentication.	Fully Implemented
NFR6	The system should be user-friendly.	The Network Service Repository has been designed in order to enable ease of use.	Fully Implemented
NFR9	The system could be documented.	The present deliverable serves as accompanying documentation for the Network Service Repository.	Fully Implemented

2.2 Security-by-design considerations

In accordance to the guidelines set forth in D1.5, in this sub-Section we elaborate on the adherence to the Security-By-Design (SBD) Framework, and the security considerations integrated in the design of the Network Service Repository. UMLSec has been used to model security implications for the Network Service Repository. UMLSec provides the necessary stereotypes for Role-based Access Control (RBAC) [1], complete with tags and constraints, which we apply to the various activity diagrams describing the processes that the Network Service Repository should support. An indicative activity diagram for interaction with the Network Service Repository, shown carrying the <<rbac>> UMLSec stereotype is shown in Figure 1.

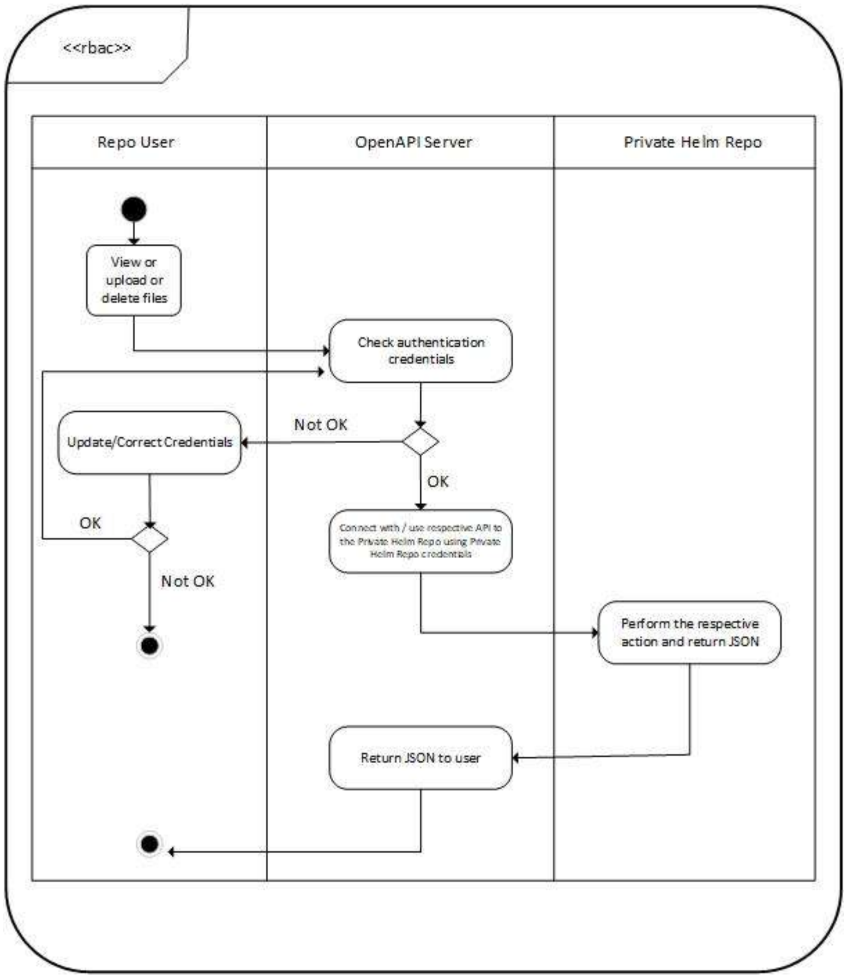

<<rbac>>
Repo User
OpenAPI Server
Private Helm Repo
View or upload or delete files
Check authentication credentials
Not OK
OK
Update/Correct Credentials
OK
Not OK
Connect with / use respective API to the Private Helm Repo using Private Helm Repo credentials
Perform the respective action and return JSON
Return JSON to user

Figure 1 UMLSec activity diagram for using the Network Service Repository.

3 Network Service Repository implementation

With respect to the 5G-EPICENTRE latest architecture (D1.4), the core Network Repository functional blocks are depicted in Figure 2. The Network Service Repository provides the 5G-EPICENTRE platform architecture with centralized support for storage and management of NS and Network Application artefacts. These will be packaged and stored in the form of Helm Charts. These Helm Charts aim at defining templates for properly installing the vertical application on a Kubernetes (K8s) cluster in an automated manner. They will be hence utilised in the southbound exchange, between the Repository and testbed K8s orchestrator (managed over the Karmada control plane block, see Task 4.3), for setting up and instantiating clusters executing CNFs over the testbeds.

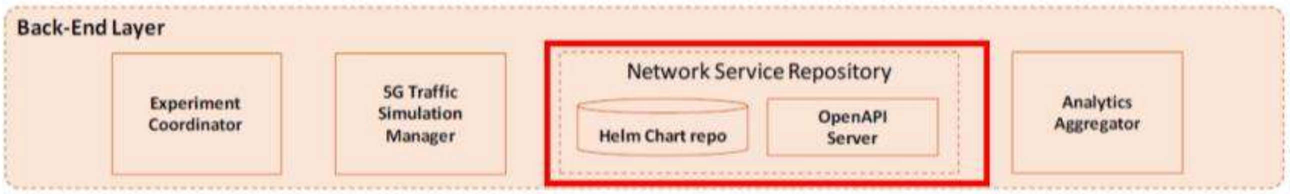


Figure 2: 5G-EPICENTRE Network Service Repository, with the contents covered in this deliverable highlighted in red (Image adapted from the 5G-EPICENTRE architecture functional view in D1.4).

The internal architecture of the Repository is shown in Figure 3. A JFrog[1] private Helm repository (see Section 3.1.2) is used to store the chart packages, providing methods for uploading, retrieving, updating and deleting Chart packages via Client URL (curl) commands. The JFrog Helm Repo exposes an extensive REST API, that supports a fully automated provisioning of Helm charts to a K8s cluster, that can be used for communication with the Experiment Coordinator (Task 2.4) to obtain chart packages and apply them over the Karmada API Server. An OpenAPI server is used as a proxy, for facilitating the interaction of the platform front-end components (Portal, Task 3.3) with the Private Helm Repo, for supporting these basic functions. Sub-sections 3.1 and 3.2 are devoted to the Private Helm Repo and the OpenAPI Server, respectively.

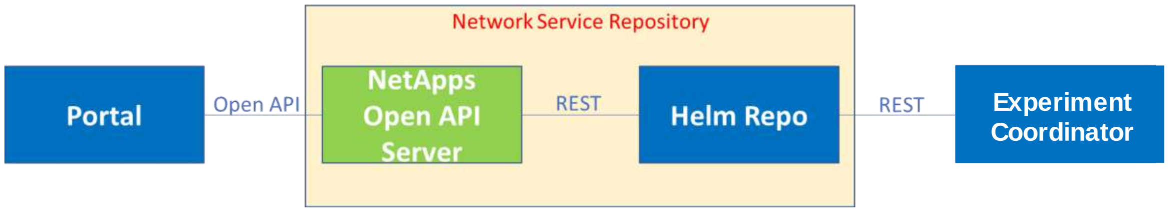


Figure 3: Network Service Repository functional elements and interfaces

According to D4.2, the UC-related Network Applications that are planned to be stored in the repository are listed in Table 3, below.

Table 3. List of UC related Network Applications and storage location

Network Application ID	Network Application Name	Storage location	UC
VS1.1	Push To Talk Critical Communication (MCPTT) Network Application	Repository	UC1

[1] https://jfrog.com/

VS1.2	Video Critical Communication (MCVideo) Network Application	Repository	UC1
VS1.3	Data Critical Communication (MCData) Network Application	Repository	UC1
VS2.1	Full Mission Critical Services (MCX) Server Network Application	Repository	UC2
VS2.2	MCX Dispatcher Network Application	Repository	UC2
VS2.3	MCX QoS management NS	Repository	UC2
VS3.1	Network Applications and Services for ultra-reliable drone navigation and remote control	Repository	UC3
VS3.2	Video splitting NS	Repository	UC3
VS4	Network Applications and Services for internet of Things (IoT) for improving first responders' situational awareness and safety	Repository	UC4
VS5	Network Applications and Services for wearable, mobile, point-of-view, wireless video service delivery	Repository	UC5
VS6.1	AI-Analyzer Network Application	Repository	UC6
VS6.2	Proxy NS	Repository	UC6
VS6.3	Dummy-Image-Sender NS	Repository	UC6
VS7	Network Applications and Services for augmented Reality (AR) and Artificial Intelligence (AI) wearable electronics for PPDR	Repository	UC7
VS8	Network Applications and Services for AR-assisted emergency surgical care	Repository	UC8

3.1 Private Helm Repo

3.1.1 Helm Charts

Helm [2] is an open-source package manager for K8s, a popular container orchestration platform. Helm allows developers and system administrators to easily package, deploy, and manage applications and services on K8s clusters. It is based on the concept of "charts," which are packages that contain all the necessary files, configuration, and dependencies needed to deploy an application on K8s. Charts are designed to be reusable and can be easily shared with other users or teams. Helm also supports versioning and rolling upgrades, making it easy to manage the lifecycle of applications running on K8s. It provides a command-line interface (CLI) for managing charts, including installation, upgrade, and deletion of charts. It also includes a web-based dashboard for browsing and searching available charts. Helm is highly extensible and supports plugins and customizations to meet specific use cases or requirements. Charts are created as files laid out in a particular directory tree. They can be packaged into versioned archives to be deployed. A chart is organized as a collection of files inside of a directory. The directory name is the name of the chart (without versioning information). Thus, a chart describing WordPress would be stored in a wordpress/ directory. Inside of this directory, Helm will expect a structure that is depicted below (as retrieve in https://helm.sh/docs/topics/charts/):

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  values.schema.json  # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/             # A directory containing any charts upon which this chart depends.
  crds/               # Custom Resource Definitions
  templates/          # A directory of templates that, when combined with values,
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

3.1.2 JFrog

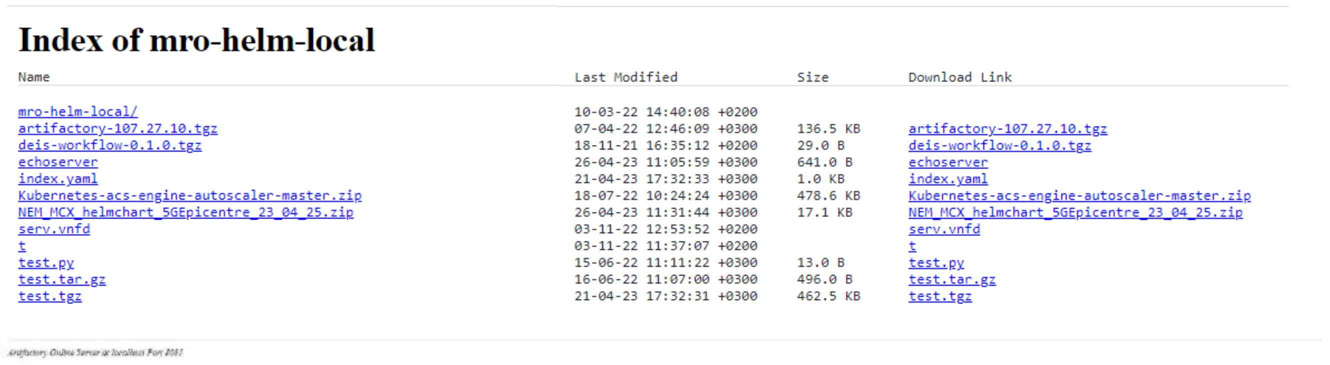
JFrog Artifactory [3] is a tool used by software developers to store, manage, and distribute binary artefacts, such as Java libraries, Docker images, and other software packages. It supports various package formats, including Maven, Gradle, npm, NuGet, and Docker, among others. It offers features such as metadata management, version control, access control, and artefact promotion. Artifactory is K8s-ready, supporting containers, Docker and Helm Charts, and is a K8s and Docker registry. It comes with full CLI and REST APIs, customizable to the applications' ecosystem. JFrog offers fully-featured operation with Helm through support for local, remote and virtual Helm chart repositories. The JFrog Helm Repo exposes an extensive REST API [4], that supports a fully automated provisioning of Helm charts to a K8s cluster. The Private Helm Repo was created using JFrog Artifactory. The Helm client used in the project is located in UMA premises, supported in the Experiment Coordinator. It will support basic Helm commands, such as *"helm install <chartname>".*

Figure 4 depicts a screenshot of the 5G-EPICENTRE JFrog Private Helm Repo index on 27/04/2023. *"NEM_MCX_helmchart_5GEpicentre_23_04_25.zip"* is the Helm chart for Use Case (UC) 2 and VS2.1 Network Application (see Table 3), developed by NEM. Currently, the rest of the UC owners are preparing the Helm charts for their applications in order to be stored in the Private Helm Repo.

3.2 OpenAPI Server

3.2.1 OpenAPI

OpenAPI [5], formerly known as Swagger, is a specification that defines a standard, language-agnostic interface for RESTful APIs. It provides a way to describe the structure of RESTful APIs using a JSON or YAML format, and includes information such as the available endpoints, HTTP methods, request and response schemas, authentication methods, and more. OpenAPI makes it easier for developers to understand and use APIs by providing a standard way to document them. This documentation can be used to generate client libraries, server stubs, and



Index of mro-helm-local

Name	Last Modified	Size	Download Link
mro-helm-local/	10-03-22 14:40:08 +0200		
artifactory-107.27.10.tgz	07-04-22 12:46:09 +0300	136.5 KB	artifactory-107.27.10.tgz
deis-workflow-0.1.0.tgz	18-11-21 16:35:12 +0200	29.0 B	deis-workflow-0.1.0.tgz
echoserver	26-04-23 11:05:59 +0300	641.0 B	echoserver
index.yaml	21-04-23 17:32:33 +0300	1.0 KB	index.yaml
Kubernetes-acs-engine-autoscaler-master.zip	18-07-22 10:24:24 +0300	478.6 KB	Kubernetes-acs-engine-autoscaler-master.zip
NEM_MCX_helmchart_5GEpicentre_23_04_25.zip	26-04-23 11:31:44 +0300	17.1 KB	NEM_MCX_helmchart_5GEpicentre_23_04_25.zip
serv.vnfd	03-11-22 12:53:52 +0200		serv.vnfd
t	03-11-22 11:37:07 +0200		t
test.py	15-06-22 11:11:22 +0300	13.0 B	test.py
test.tar.gz	16-06-22 11:07:00 +0300	496.0 B	test.tar.gz
test.tgz	21-04-23 17:32:31 +0300	462.5 KB	test.tgz

Artifactory Online Server at localhost Port 2081

Figure 4: Index of the Private Helm Repo

other tools, that can help developers interact with the API. It also enables automated testing and validation of API requests and responses.

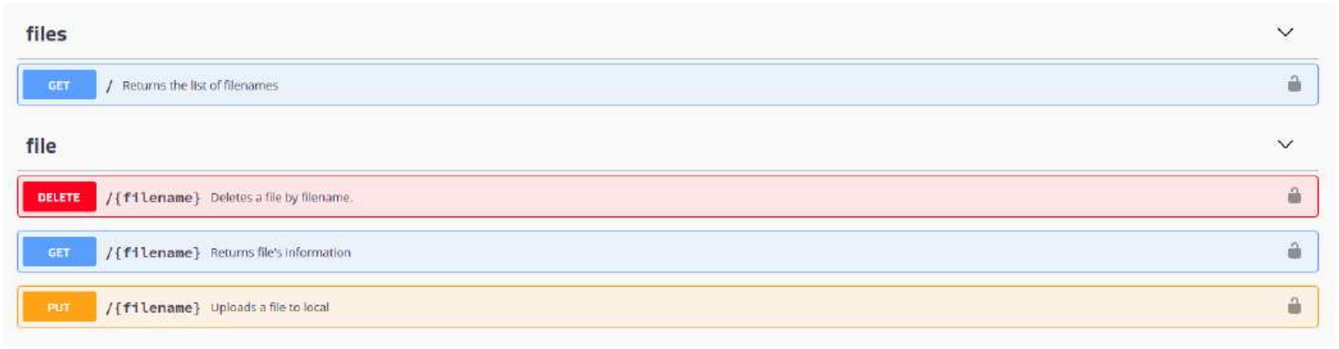
3.2.2 OpenAPI Server Details

The 5G-EPICENTRE OpenAPI server was generated and developed for Python Flask[2]. The APIs exposed (to be consumed by the Portal) support the following capabilities/operations:

Table 4: Network Service Repository API endpoints exposed to the 5G-EPICENTRE Portal

Method	Endpoint	Description
GET	/	Returns the list of filenames in the repository.
DELETE	/{filename}	Deletes a file by filename.
GET	/{filename}	Returns the specified file information and content. If the file is in an archive file format (implying Helm chart), the contents of "Chart.yaml" are returned.
PUT	/{filename}	Uploads a file to the repository.

Figure 5 shows a screenshot from the Swagger User Interface (UI) of the OpenAPI server. In the Appendix, some example responses to the above listed API calls can be found.



files

GET / Returns the list of filenames

file

DELETE /{filename} Deletes a file by filename.

GET /{filename} Returns file's information

PUT /{filename} Uploads a file to local

Figure 5: Network Service Repository API endpoints exposed to the 5G-EPICENTRE Portal

[2] https://flask.palletsprojects.com/en/2.2.x/

The following Figures show simple sequence diagrams for viewing of the filenames, for deletion of a file, for viewing Helm chart metadata and for file uploading in the Network Service Repository respectively.

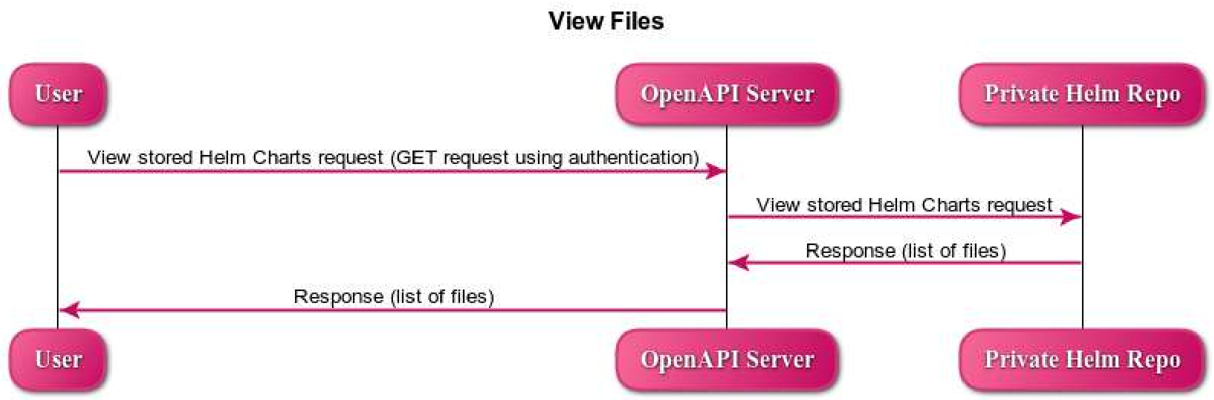

View Files
User
OpenAPI Server
Private Helm Repo
View stored Helm Charts request (GET request using authentication)
View stored Helm Charts request
Response (list of files)
Response (list of files)

Figure 6. Viewing of filenames in the Repository

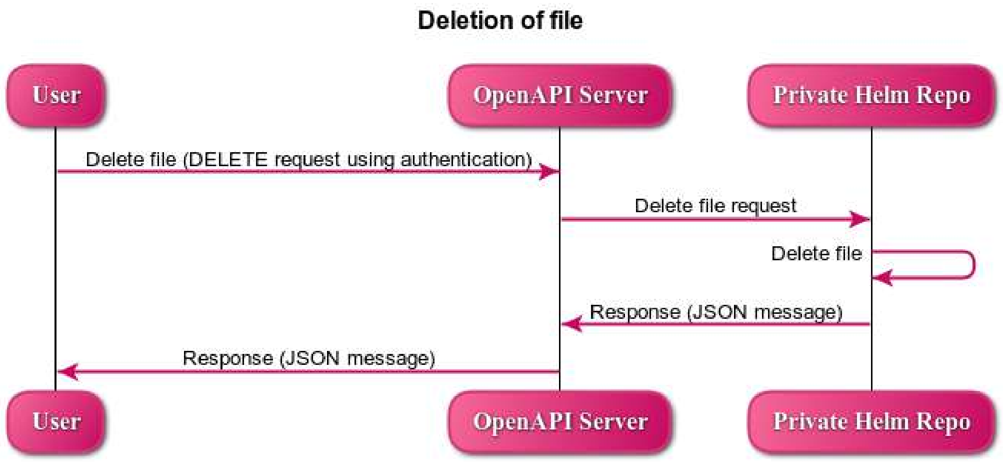


Figure 7. Deletion of file in the Repository

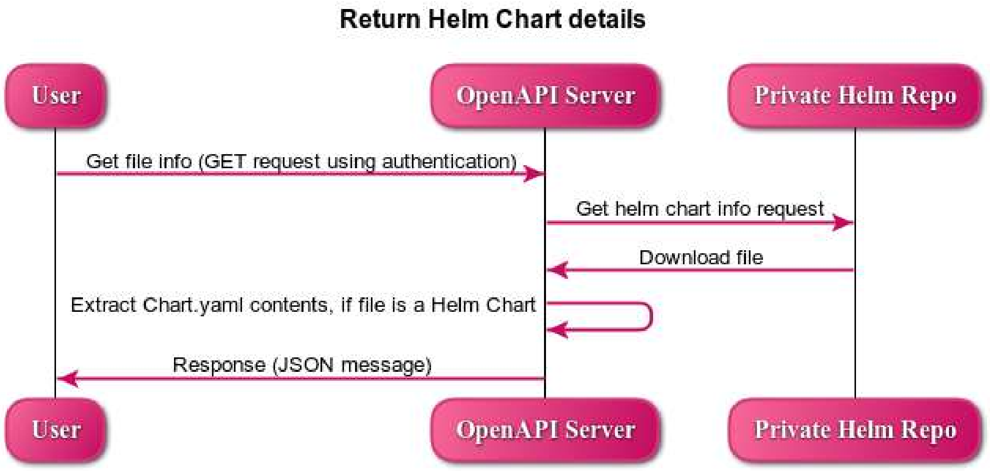

Return Helm Chart details
User
OpenAPI Server
Private Helm Repo
Get file info (GET request using authentication)
Get helm chart info request
Download file
Extract Chart.yaml contents, if file is a Helm Chart
Response (JSON message)

Figure 8. Viewing metadata about a Helm chart

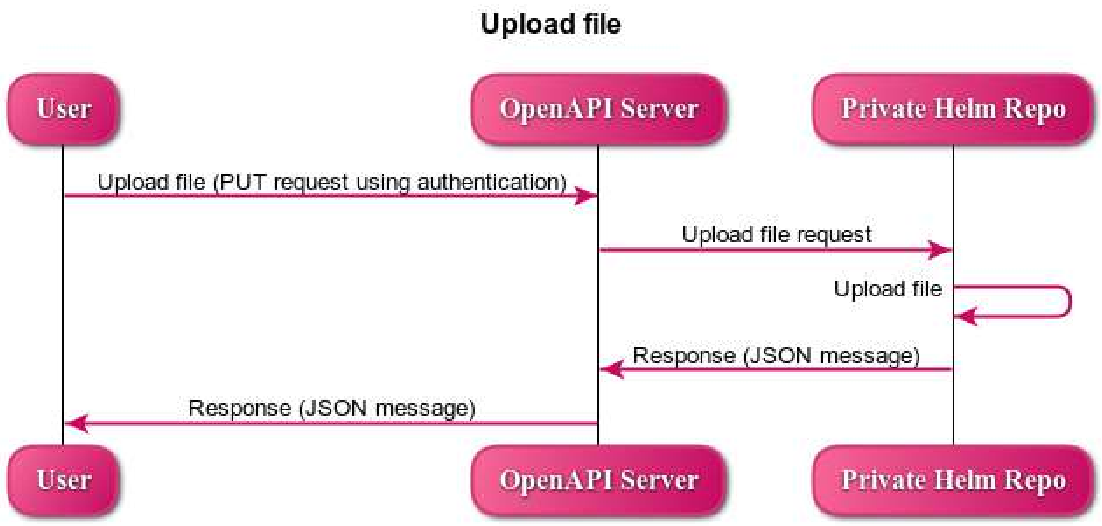


Figure 9. Uploading a Helm chart in the Repository

4 Conclusions

This document is the deliverable related to Task T4.2 “Container Network Functions and Network Application repositories” and more specifically to sub-task 4.2.2 “Network Application provision”. The document presents high-level details about the implementation of the Network Service Repository and the technologies used. Both sub-components of the Repository (Private Helm Repo and OpenAPI server) were described in respective sub-sections.

We first described how the Network Service Repository adheres to the QR, FR and NFR requirements of the project. We then elaborated on the adherence to the SBD Framework and the security considerations presented in D1.5. We illustrated the internal architecture of the Network Service Repository and its place in the 5G-EPI-CENTRE platform architecture. We then proceeded to describe how basic interactions, such as viewing the file-names of the Helm Charts stored in the repository, are realized using the OpenAPI server. This deliverable serves as an accompanying report to the Network Service Repository software, deployed at the UMA premises according to the D1.4 architecture deployment plan.

Currently the Repository hosts the NEM UC2 Network Application Helm Chart, while the rest of the UC owners are preparing their own Helm charts. JFrog Private Helm Repo credentials and the OpenAPI server code was given to UMA in order to be uploaded to their premises, in order to facilitate the communication of the Repository with the Portal and the Experiment Coordinator.

References

[1] Matulevičius R. (2017) Role-Based Access Control. In *Fundamentals of Secure System Modelling* (pp. 147-169). Springer, Cham. https://doi.org/10.1007/978-3-319-61717-6_10
[2] Helm Authors (2023),” Helm, The package manager for Kubernetes”, https://helm.sh/
[3] JFrog Authors (2023), https://jfrog.com/
[4] JFrog Authors (2023), “Kubernetes Helm Chart Repositories”, https://www.jfrog.com/confluence/display/JFROG/Kubernetes+Helm+Chart+Repositories
[5] OpenAPI authors (2023), “OpenAPI Specification”, https://swagger.io/specification/

Annex I: OpenAPI Server Response Examples

In this appendix the response bodies (JSON) examples for the four API calls of Table 4 are listed

---

- **GET: response----**

This is the body response of a GET request, in order to view the view the names of the files stored in the Repository. A 200 OK status code was returned.

Response body:

```
  "body": [
    "mro-helm-local/",
    "artifactory-107.27.10.tgz",
    "deis-workflow-0.1.0.tgz",
    "echoserver",
    "index.yaml",
    "Kubernetes-acs-engine-autoscaler-master.zip",
    "NEM_MCX_helmchart_5GEpicentre_23_04_25.zip",
    "serv.vnfd",
    "t",
    "test.py",
    "test.tar.gz",
    "test.tgz"
  ],
  "code": 200,
  "message": "The filenames are retrieved successfully."
}
```

- **DELETE: response---**

This is the body response of a DELETE request, in order to delete a file from the Repository. A 204 code was returned indicating that the server has successfully deleted the artifact and that there is no content to send in the response payload body.

Response body:

```
{
  "code": 204,
```

"message": "Artifact has been deleted successfully.",

"type": "{'Date': 'Wed, 15 Jun 2022 09:33:34 GMT', 'Connection': 'keep-alive', 'X-JFrog-Version': 'Artifactory/7.38.10 73810900', 'X-Artifactory-Id': '303c0646c50d8f99105c74a94c3195d500ca8f8e', 'X-Artifactory-Node-Id': 'a0srispgkkmfa-artifactory-primary-0', 'Strict-Transport-Security': 'max-age=15724800; includeSubDomains', 'Access-Control-Allow-Origin': 'https://connect.jfrog.io', 'Access-Control-Allow-Methods': 'PUT, GET, POST, OPTIONS', 'Access-Control-Allow-Headers': 'User-Agent,Content-Type,Authorization,X-Explode-Archive,X-Explode-Archive-Atomic', 'Access-Control-Allow-Credentials': 'false'}"

}

- **GET: response-----**

This is the body response of a GET request, in order to view the Chart.yaml content of "NEM_MCX_helm-chart_5GEpicentre_23_04_25.zip" Helm Chart stored in the Repository. A 200 OK status code was returned.

Response body:

{

"body": "apiVersion: v2\nname: nemergent_mcs\ndescription: A Helm chart to deploy Nemergent MCS Service \n\n# A chart can be either an 'application' or a 'library' chart.\n#\n# Applica-tion charts are a collection of templates that can be packaged into versioned archives\n# to be deployed.\n#\n# Library charts provide useful utilities or functions for the chart develop-er. They're included as\n# a dependency of application charts to in-ject those utilities and functions into the rendering\n# pipeline. Library charts do not define any templates and therefore cannot be deployed.\ntype: application\n\n# This is the chart version. This version number should be incremented each time you make changes\n# to the chart and its templates, including the app version.\n# Versions are expected to follow Semantic Versioning (https://semver.org/)\nversion: 2.0.0\n\n# This is the version number of the application being deployed. This version number should be\n# incremented each time you make changes to the ap-plication. Versions are not expected to\n# follow Semantic Versioning. They should reflect the version the application is using.\nappVersion: oinarri\n\nicon: https://nemergent-solutions.com/imagenes/main_logo_negativo.svg\n",

"code": 200,

"message": "Artifact has been downloaded successfully."

}

- **PUT: response-----**

This is the body response of a PUT request, in order to upload "NEM_MCX_helmchart_5GEpicentre_23_04_25.zip" Helm Chart in the Repository.

Response body:

"checksums": {

"md5": "6eb8293cec298eda815ee55adb5c840e",

"sha1": "54d9e833e97435a985e1feee2545ee22fa08fc42",

"sha256": "a1ba0a63a0bc5b2bb721301f36fdae71e0f02796555b74ca9acb36224dc622d9"

```
  },
  "created": "2023-04-26T08:15:18.992Z",
  "createdBy": "roumpam@gmail.com",
  "downloadUri": "https://jfrogmro.jfrog.io/artifactory/mro-helm-local/NEM_MCX_helmchart_5GEpicen-
tre_23_04_25.zip",
  "mimeType": "application/zip",
  "originalChecksums": {
    "sha256": "a1ba0a63a0bc5b2bb721301f36fdae71e0f02796555b74ca9acb36224dc622d9"
  },
  "path": "/NEM_MCX_helmchart_5GEpicentre_23_04_25.zip",
  "repo": "mro-helm-local",
  "size": "18024",
  "uri": "https://jfrogmro.jfrog.io/artifactory/mro-helm-local/NEM_MCX_helmchart_5GEpicen-
tre_23_04_25.zip"
}
```