EXPERIMENTATION AND VALIDATION OPENNESS FOR LONGTERM EVOLUTION OF VERTICAL INDUSTRIES IN 5G ERA AND BEYOND

[H2020 - Grant Agreement No.101016608]

Deliverable D3.1

Implementations and integrations towards EVOLVED-5G framework realisation (intermediate)

Editor	R. Marco Alaez (ATOS)
Contributors	(TID), (ATOS), (NCSRD), (UMA), (UPV), (MAG), (INTRA), (LNV)
Version	1.0
Date	December 23rd, 2021
Distribution	PUBLIC (PU)

DISCLAIMER

REVISION HISTORY

Revision	*Date*	*Responsible*	*Comment*
0.1	*September, 9th, 2021*	*R. Marco (ATOS)*	*Edit ToC*
0.2	*October, 25th, 2021*	*R. Marco (ATOS)*	*New ToC*
0.2.1	*November, 8th, 2021*	*R. Marco (ATOS)*	*New document structure*
0.3	*November, 19th, 2021*	*R. Marco (ATOS)*	*Internal review*
0.4	*December, 2nd, 2021*	*R. Marco (ATOS)*	*SC review*
0.5	*December 8th, 2021*	*D. Tsolkas (FOG)*	*Final Review*
0.6	*December 13th, 2021*	*H. Koumaras (NCSRD)*	*Final Review*
0.7	*December 13th, 2021*	*R. Marco*	*Final Review*
1.0	*December 23rd, 2021*	*R. Marco*	*Final version*

LIST OF AUTHORS

Partner ACRONYM	*Partner FULL NAME*	*Name & Surname*
TID	*TELEFONICA INVESTIGACIÓN Y DESARROLLO*	*Javier Garcia* *David Artuñedo* *Alejandro Molina*
ATOS	*ATOS IT SOLUTIONS AND SERVICES IBERIA SL*	*Ricardo Marco* *Paula Encinar* *Sonia Castro*
NCSRD	*NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"*	*Stavros Kolometsos* *George Makropoulos* *Harilaos Koumaras* *Dimitris Fragkos* *Anastasios Gogos* *Dimitris Kyriazanos* *George Thanos*
UMA	*UNIVERSIDAD DE MALAGA*	*Bruno Garcia* *Francisco Luque Schempp* *Maria del Mar Moreno*
UPV	*UNIVERSITAT POLITECNICA DE VALENCIA*	*Regel G. Usach*
FOGUS	*FOGUS INNOVATIONS & SERVICES P.C*	*Dimitris Tsolkas*
MAG	*MAGGIOLI*	*Alexandros Tzoumas* *Yiannis Karadimas*
INTRA	*INTRASOFT INTERNATIONAL SA*	*Angela Dimitriou* *Manolis Falelakis*
LNV	*LENOVO DEUTSCHLAND GMBH*	*Apostolis Salkintzis* *Dimitris Dimopoulos*

GLOSSARY

Abbreviations/Acronym	*Description*
3GPP	*3rd Generation Partnership Project*
5GC	*5G Core*
5GS	*5G System*
AI	*Artificial Intelligence*
API	*Application Programming Interface*
APK	*Android Application Package*
CAPIF	*Common API Framework*
CI/CD	*Continuous Integration / Continuous Development*
CLI	*Command Line Interface*
COTS	*Commercial off-the-shelf*
CPE	*Customer Premises Equipment*
EC	European Commission
ELCM	*Experiment Life-Cycle Manager*
FoF	Factories of the Future
gNodeB / gNB	*Next Generation (5G) Base Station*
GUI	*Graphical User Interface*
HAT	*Hardware Attached on Top*
HTTP	*Hypertext Transfer Protocol*
IioT	Industrial Internet of Things
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
KPI	*Key Performance Indicator*
LDAP	*Lightweight Directory Access Protocol*
LTE	*Long Term Evolution*
LTE-A	*Long Term Evolution Advanced*
LTE-M	*Long Term Evolution for Machines*
MAC	*Mandatory Access Control*
MANO	*Management and Orchestration*
NB-IoT	*Narrow Band – Internet of Things*
NEF	*Network Exposure Function*
NetApp	*Network Application*
NFV	*Network Function Virtualization*
NSA	*Non StandAlone*
NSD	*Network Service Descriptor*
Oauth	*Open Authorization*
OEM	*Original Equipment Manufacturer*
ONAP	*Open Network Automation Platform*

OSM	*Open Source MANO*
PoP	*Platform to Platform*
PR	*Pull request*
QoS	*Quality of Service*
R&D	*Research and Development*
REST	*Representational State Transfer*
SA	*StandAlone*
SDK	*Software Development Kit*
SLA	*Service Level Agreement*
SME	Small Medium Companies
SSH	*Secure Shell*
SSL	*Secure Sockets Layer*
SSM	*Service Specific Manager*
TSL	*Transport Layer Security*
UE	*User Equipment*
UI	*User Interface*
vApp	*Vertical Application*
VDU	*Virtual Display Unit*
VNF	*Virtual Network Function*
VulnDB	*Vulnerability Database*
WiFi	*Wireless Fidelity*

EXECUTIVE SUMMARY

EVOLVED-5G responds to the *5G PPP ICT-41-2020 5G innovations for verticals with third party services* call [1], whose main goal is to deliver enhanced experimentation facilities on top of which third party experimenters (e.g., SMEs or any service provider and target vertical users) will have the opportunity to develop and test their applications.

The EVOLVED-5G project realises this vision by encouraging the creation of a NetApp ecosystem revolving around a 5G facility which will provide the tools and processes for the development, verification, validation and certification of NetApps as well as mechanisms for market releasing.

This document provides an overview of the current implementations and early prototypes of various parts of the EVOLVED-5G experimentation facility. More precisely, the deliverable describes the result of work conducted by the third Work Package (WP) of the project, *WP3-Overall framework development and integration activities*, and, in concrete, in Tasks *T3.1-Production of the workspace and the related verification tools*, *T3.2-Development of NetApps validation tools and open repository*, and *T3.3-Overall framework integration and 5G infrastructure evolution*. Indeed, the main goal of this WP is to implement all the core components of the reference architecture defined by *WP2-Realisation process, requirements, and reference architecture design*, related to the design of the environments and components of the EVOLVED-5G facility that will enable the development of NetApps and their lifecycle management.

The main contribution of this deliverable is to provide a detailed description of components from the EVOLVED-5G facility that enable the NetApp development, verification and validation processes. The spearhead for those processes is the *EVOLVED-5G Workspace environment* that provides the necessary tools for developers to ignite the development of NetApps, and hence set in motion verification and validation phases over which functional and non-functional testing activities take place. In more detail, this deliverable includes:

- The State of the Art of a variety of tools related to the different environments being implemented in the EVOLVED-5G facility, such as Software Development Kits (SDK), verification and validation tools, and potential open-source options for code and artifact repositories.
- An overview of the foreseen EVOLVED-5G facility, including the overall architecture and the environments implemented to support the NetApps lifecycle. For the deliverable's completeness, a brief recall from previous deliverables, regarding the NetApp conceptualization and implementation principles, is also provided.
- A detailed description of the Workspace and Validation environments along with the open repository, including security measures as well as their current implementation details.
- The evolutions conducted towards the 5G infrastructure of EVOLVED-5G facility, to enable the new required functionalities of the NetApp ecosystem.

The work presented in this deliverable will guide the project towards later implementation stages of the EVOLVED-5G facility, since it provides an initial implementation design and realization methodology that includes all the phases that any NetApp must undergo until it reaches the *EVOLVED-5G Certification environment*. Information will be expanded and updated in future outcoming WP3 deliverables, and especially in *D3.2-NetApp Certification Tools and*

Marketplace development (to be delivered at M18), which will cover the phases of NetApp certification and release to the *EVOLVED-5G Marketplace*.

TABLE OF CONTENTS

1 INTRODUCTION 1
1.1 Target audience 1
1.2 Purpose 2
1.3 Structure 2
2 State of the Art 4
2.1 SDK Tools 4
2.1.1 SONATA SDK tools 4
2.1.2 5GTANGO SDK tools 5
2.1.3 Cookiecutter 7
2.2 Verification Tools 8
2.3 Validation Tools 9
2.4 Open repositories 10
2.4.1 Code repositories 10
2.4.2 Artifact repositories 10
3 Evolved-5G Facility 12
3.1 General description 12
3.2 NetApps principles and implementation 14
3.3 High level architecture 15
4 Workspace 17
4.1 Main objective 17
4.2 Component description 17
4.2.1 Development Phase 20
4.2.2 Verification Phase 20
4.3 Implementation 20
4.3.1 Development Phase 20
4.3.2 Verification Phase 27
4.4 Security 29
4.4.1 Development Phase 29
4.4.2 Verification Phase 29
5 Validation environment 31
5.1 Main objective 31
5.2 Component description 31
5.3 Design 31
5.4 Security 32
5.5 Implementation 32
6 Open repository 35

6.1 Main objective 35
6.2 Component description 35
6.3 Design 35
6.3.1 Source Code open repository 36
6.3.2 Container images repository 36
6.4 Security 37
6.5 Implementation 38
7 EVOLVED-5G Infrastructure 40
7.1 Main Objective 40
7.2 Component Description 40
7.2.1 Athens platform 40
7.2.2 Málaga platform 43
7.2.3 CAPIF Core Function Tool 48
7.2.4 NEF Emulator 49
8 Conclusion and next steps 55
REFERENCES 56

LIST OF FIGURES

Figure 1 Cookiecutter usage example 8
Figure 2 Overall view of the current elements composing the different environments of the EVOLVED-5G facility. 14
Figure 3 EVOLVED-5G reference architecture........ 16
Figure 4 Workspace architectural components 18
Figure 5 Workflow of the functional blocks of the workspace 19
Figure 6 Template structure........ 21
Figure 7 Inputs to create and setup a NetApp. 22
Figure 8 Instructions repository 22
Figure 9 SDK-CLI repository........ 23
Figure 10 Inputs from generate command 24
Figure 11 Location Subscriber – High level usage 25
Figure 12 Pipelines repository........ 26
Figure 13 Run pipeline command 26
Figure 14 Build Pipeline execution example in Jenkins........ 27
Figure 15 Openshift Web Management Interface 28
Figure 16 NetApp interaction with NEF Emulator via RESTful APIs 28
Figure 17 *EVOLVED-5G validation environment using the Open5Genesis Framework*........ 33
Figure 18 GitHub Free Plan description 36
Figure 19 Artifactory selected plan “Pro X” by Telefónica........ 37
Figure 20 Github security packages included in Free plan........ 38
Figure 21 Evolved5G GitHub repository........ 39
Figure 22 EVOLVED-5G Validation and Certification areas in Artifactory........ 39
Figure 23 Logical Topology of the evolved Athens platform 41
Figure 24 Amarisoft Callbox Mini........ 41
Figure 25 SIM8200EA-M2 5G HAT UE 42
Figure 26 Qualcomm Snapdragon X55 CPE........ 43
Figure 27 Indoor small cell antennas 43
Figure 28 Outdoor antennas 44
Figure 29 Area covered by the outdoor deployment........ 44
Figure 30 From left to right - One Plus 9, Huawei P40 Samsung S20 Ultra 5G and Telit FN980m 5G M.2........ 44
Figure 31 Matrice 600 Pro........ 45
Figure 32 Husky Unmanned Ground Vehicle 45
Figure 33 Relyum TSN Bridge 46
Figure 34 Relyum TSN PCIe card 47
Figure 35 TSN architecture........ 48
Figure 36 CAPIF architecture........ 48
Figure 37 High-Level Architecture of NEF Emulator 50
Figure 38 Swagger UI........ 51
Figure 39 NEF emulator - Dashboard........ 51
Figure 40 NEF emulator - Map 52
Figure 41 Network scenario 53
Figure 42 Request/Notification details 53
Figure 43 NEF emulator - Authentication/Authorization........ 54

1 INTRODUCTION

1.1 TARGET AUDIENCE

The scope of the deliverable is to be public, intending to expose implementation details both for workspace and validation of EVOLVED-5G environments along with upgrades on the EVOLVED-5G infrastructure, namely Malaga and Athens, to a broad variety of research individuals and communities.

From specific to broader, different target audiences for D3.1 are identified as detailed below:

- **Project Consortium**: To validate that all objectives and proposed technological advancements have been analysed and to ensure that, through the proposed implementations, further work can be derived. Furthermore, the deliverable sets to establish a common understanding among the consortium with regards to:
  - The implementation details for each environment (development, verification, and validation) related to the NetApp lifecycle in the context of the EVOLVED-5G project, including tools and technologies to be employed.
  - The detailed evolution of the infrastructure, either physical components or new features, applied to the two platforms (Athens and Malaga) over which the experimentation will take place.
- **Industry 4.0/Industry 4.0 developers and FoF (Factories of the Future) vertical groups:** To set a joint understanding of the technologies and the design principles that underline the architectural design of the EVOLVED-5G facility, its implementation and the main phases that the NetApps undergo. A non-exhaustive list of Industry 4.0-related groups is as follows:
  - Manufacturing industries (including both large and SMEs) and Industrial Internet of Things (IIoT) technology providers.
  - European, national, and regional manufacturing initiatives, including funding programs, 5G-related research projects, public bodies and policy makers.
  - Technology transfer organizations and market-uptake experts and researchers.
  - Standardisation Bodies and Open-Source Communities.
  - Industry 4.0 professionals, researchers and developers with technical knowledge and expertise on this vertical and have al professional background on industry 4.0-related areas.
  - Industry 4.0 investors and business angels.
- **Other vertical industries and groups**: To seek impact on other 5G-enabled vertical industries and groups in the long run. Indeed, all the architectural components of the facility are designed to secure interoperability beyond vendor- specific implementation and across multiple domains. The same categorization can be applicable beyond the Industry 4.0 domain.
- **The scientific audience, general public and the funding EC Organisation**: To document the work performed by the project and justify the effort reported for all relevant activities. The scientific audience can also get an insight of the design approach and underlying components behind the EVOLVED-5G ecosystem implementation.

1.2 PURPOSE

This deliverable is the first of a series of four WP3 reports, to be delivered by the project consortium during its 30-month work plan. The primary objective of this first report, entitled *"D3.1 Implementations and integrations towards EVOLVED-5G framework realisation (intermediate)"*, is to present the actual implementation of tools, activities and methodologies that are materializing for the realisation of the EVOLVED-5G facility, targeting implementation, efficiency and security aspects of its architectural components that define NetApp processing environments.

This document describes the result of three out of the four different tasks that compose WP3: *T3.1-Production of the Workspace and the Related Verification Tools*, related mainly to the SDK tools developed for the creation and verification of NetApps; *T3.2-Development of NetApp Validation Tools and Open Repository*, focused on how to store and validate NetApps; and *T3.3-Overall Framework Integration and 5G Infrastructure Evolution*, devoted to how to link every environment with each other, on top of 5G connectivity infrastructures.

The second deliverable, *"D3.2- NetApp Certification Tools and Marketplace development (Intermediate)"*, to be submitted in M18 will report on the intermediate version of the Task *T3.4-Certification Tools and Marketplace Development* results with respect to the certification tools and marketplace development. The third deliverable, *D3.3-Implementations and integrations towards EVOLVED-5G framework realisation (final),* to be submitted in M28, and the fourth deliverable, *D3.4-NetApp Certification Tools and Marketplace development,* to be released in M30, will be extensions of D3.1 and D3.2 deliverables respectively.

WP3 deliverables, and especially this one, provide essential input to *WP4-NetApps Development and Verification,* devoted to the development of the NetApps being specifically created by the project, as well as to *WP5-Overall Evaluation process, NetApp Validation, Certification and Release,* where all the work related to the validation and certification of NetApps takes place.

1.3 STRUCTURE

The deliverable is organized in the following manner:

- **Section 1. Introduction**: This serves as an introduction to the deliverable.
- **Section 2. State of the Art:** This section provides the State of the Art of different tools related to NetApp development, verification and validation. Also, different open repository solutions are analysed.
- **Section 3. EVOLVED-5G Facility**. This section briefly describes the EVOLVED-5G facility, its architecture. It also outlines the EVOLVED-5G NetApp conceptualization and implementation principles.
- **Section 4. Workspace:** This section describes, in a thoroughly detailed way, the scope and status of implementation for those architectural components that are necessary for the realisation of two of the phases of the NetApp lifecycle: development and verification.
- **Section 5. Validation environment.** It addresses the validation of NetApps through a series of tests by making use of real 5G infrastructure. It assesses the different details about the implementation of the environment and the functionality of the facility.
- **Section 6. Open repository.** This section contains the solution chosen for the EVOLVED-5G project regarding the storage of NetApps images (artifacts). It is also described how

the NetApps are classified and stored, together with security and implementations details of the Open repository.

- **Section 7. 5G Infrastructure evolution.** This section focuses on describing all the improvements that have been implemented from M3 until the release of this deliverable in both platforms, Málaga and Athens. A description about new features and/or new functional blocks introduced within the context of the EVOLVED-5G project are explained in detail.

2 State of the Art

Prior the design and implementation of the different environments composing the EVOLVED-5G architecture, the EVOLVED-5G consortium members performed an exhaustive state-of-the-art research on the existing technologies and tools that fit with the current architectural vision of EVOLVED-5G, including those coming from the 5G-PPP associated projects and initiatives.

The identified utilities were deeply analysed by EVOLVED-5G partners in order to take advantage of existing knowledge and developments in the 5G area and to set the grounds for the design, the low-level definition and the implementation of the different environments that compose the EVOLVED-5G facility. This analysis allowed to leverage the fruit of the efforts performed during the last years towards the realisation of 5G programmability and to perform a wiser decision making on the choice of tools and technologies employed on the construction of the different environments composing the EVOLVED-5G architecture (Workspace, Validation, Certification and Marketplace).

In particular, in this section, it is shown the results of investigating potential elements regarding 5G programmability workspace utilities, namely **SDK** and **Verification tools**; 5G **Validation environments**; and moreover, **open repositories** which are key utilities employed on all phases of the NetApp lifecycle (development, verification, validation, certification and marketplace publication).

2.1 SDK Tools

This subsection overviews relevant open Software Development Kit tools (SDKs) for 5G network programmability. Some relevant initiatives for the creation of an SDK to this aim are 5G-PPP ICT-14 and ICT-8 projects SONATA [2] and 5GTANGO [3], respectively. With the aim of supporting the lifecycle of NFV-based services, the SONATA SDK provides software tools to assist in the development, debugging and validation of Virtual Network Functions (VNFs). 5GTANGO also provided its own SDK for NFV-Service Development by reusing and extending the open SONATA's kit [4].

Moreover, another remarkable SDK tool worth mentioning that assists on 5G-related programming is Cookiecutter [5], which eases notably the construction of software projects and repositories as well as the use of a predefined structure for a specific type of software development by providing a ready to use template that includes the basic scaffolding for the specific development, which helps homogenising the produced code and spare time creating and debugging the structure. Cookiecutter is an open-source, well-known and widely supported tool that very significantly eases the preparation of the software structure and considerably simplifies the development task as a result. Due to those benefits, Cookiecutter is a part of the NetApp Workspace Environment of EVOLVED-5G with the aim of providing a significant aid to developers in the construction of NetApps.
Those aforementioned SDK tools are described in the following subsections.

2.1.1 SONATA SDK tools

The SONATA project [2] aims to support the whole lifecycle of an NFV-based service, from development to production. In this regard, the SONATA SDK offers a set of software tools to assist developers in the programming, debugging, verification and validation of VNFs. This way, the SDK allows for a testing environment (i.e. sandbox), to develop and test NFV-based network services.

The SONATA SDK offers support for typical VNF-related actions such as: creation and test several forwarding chains through VNFs, injection of test traffic into an emulated network service, development and testing of APIs to interact with deployed VNFs and package the newly developed or modified services. Either during development or during its lifetime, service updates can be verified in the SDK; the SDK ensures that network services can be updated, if necessary, at any time, even if they are being employed in a production stage.

The software tools composing SONATA SDK are open-source [7] and can generally be deployed independently with the exception of three Command Line (CLI) tools that are integrated in the SONATA CLI utility (i.e. son-package, son-profile and son-validate). The SONATA SDK is made of:

- A **service graph editor** (*son-editor*) [8] that allows a visual creation and modification of VNF chains.
- An **NFV-based emulator** (*son-emu*) [9] that is capable of deploying container-based service chains in a virtual multi-platform-of-platforms (PoP) test environment, which is locally generated by the emulator.
- **VNF monitoring system**: that allows to collect monitoring data to debug, analyse and characterize a network service. *Son-monitor* [10] provides tools to easily monitor and generate metrics for debugging and analysing service performance. The SONATA monitoring system uses Prometheus to gather and store measured data and Grafana to visualize the metrics.
- **•VNF service analysis** (*son-analyse):* provides analysis tools to process the monitored service data.
- **Network service package** (*son-package*) [11]: that describes the network service in terms of resources, VNF images, networking, forwarding chains and other aspects that allow a successful deployment by the SONATA MANO platform on a virtualized infrastructure.
- **Support tool for VNF performance profiling** (*son-profile*) [11], to assist in capacity and resource planning for virtualized services, enabling service developers to automatically profile their network services and network functions with this CLI utility. By defining automated test and a set of KPIs to monitor, the performance of the VNFs can be characterized under different resource configurations.
- A **tool for a pre-deployment validation** of the generated network service package and forwarding graphs. *Son-validate* [11], a SONATA CLI tool, can be used to validate the syntax, integrity and topology of the service packages, projects, services and functions. This tool can be used via the CLI or as a micro-service running in a docker container.

As these tools are open source, they could potentially be reused, adapted or extended in the frame of the EVOLVED-5G project if it may be considered in the future useful to support the development and verification of NetApps and associated vertical Apps, or other NetApp lifecycle processes. The knowledge and ideas derived from the creation and use of this SDK, documented by SONATA, were useful as a starting point for the design of an SDK for the creation of NetApps in EVOLVED-5G project.

2.1.2 5GTANGO SDK tools

H2020 5GTANGO project developed an SDK for 5G network programmability (specifically, for NFV service development) by (i) leveraging the SDK provided by SONATA, extending the functionality of most utilities and adopting them, and (ii) developing and including in 5GTANGOS's kit a set of new tools. The aim of the SDK is to assist developers on the creation of NFV services and, furthermore, on other phases of the NFV-service lifecycle.

The final set of open tools offered in 5GTANGO SDK [12] are the following:

- **Schema** [13]: the schema definitions provided define all descriptor formats considered both in 5GTANGO and SONATA and support cloud native and physical deployment units as well as QoS requirements. Moreover, it also provides some examples and tests.
- **Developer portal** [14]**:** a Graphical User Interface to launch any of the 5GTANGO SDK tools, allowing its usage through the portal. This GUI aims to support a seamless development workflow employing the 5GTANGO SDK that can be deployed locally or be used remotely as a web-based graphical service.
- **Decision support engine** [15]: a component that proposes tests or VNFs based on a user profile and its similarity to other users.
- **Descriptor generator** [16]**:** a template generation tool to assist developers in the creation of descriptors. It provides a simple web-based GUI for the setup and generation of correct descriptors (VNFDs and NSDs). The generator uses high-level information provided and combines it with sensible default values to generate descriptors, both for 5GTANGO and for OSM. It exposes a REST API, and can be used also integrated with the project management tool or with SDK portal.
- **Packager** [18]**:** package creation tool for different MANO platforms, supporting several storage backends (e.g. tng-cat [17], OSM, ONAP), extended REST API and CLI.
- **Validator** [19]**:** tool for performing a syntactic and semantic checking of descriptors of VNFs and services. This component can be used to validate the syntax, integrity and topology of 5GTANGO file descriptors. The validator allows this 2-fold checking for cloud-native functions, SLAs, policies, slices and test descriptors, and provides support for custom rules. This tool can be used through the CLI or as a micro-service running inside a docker container.
- **sm tester** [20]**:** tool to assist testing of service-specific managers (SSMs) and enabling reusable patterns for state migration, merging and splitting.
- **Test creation framework** [24]**:** Python-based tool that enables functional testing of services and VNFs. Developed tests can be executed locally on the emulator.
- **Emulator** [25]: tool that supports the local deployment of NFV services under development without employing any 5G platform or 5G network. The emulator includes support for cloud-native NFs and multi-VDU deployments, for E-Line, E-LAN and E-Tree networks, for CI/CD with OSM, and support service function chaining. The aim of this tool is to assist the development process of NFV services, allowing to check and debug their performance and functionality.
- **Benchmarker** [27]: tool for enabling the automatic generation of performance metrics of the NFV services under a range of pre-defined resource configurations. The data output generated can be directly integrated with the analytics engine for its analysis. This component is based on the profiling tool (son-profile) from SONATA.
- **analytics engine** [28]: this tool enables the analysis of performance and correlation data and supports time-based selection of monitored metrics, time series analysis and extensive visualization of NFV services.

In a similar way as the SONATA SDK, the kit of utilities from 5GTANGO for the development of 5G applications provided valuable information, knowledge and ideas that served as an initial ground for the creation of the EVOLVED-5G SDK for the development and verification of NetApps. Although at this juncture the 5GTANGO SDK tools are not expected to be employed in the EVOLVED-5G project, they might be reused and adapted in the future if their adaptation is considered useful for the EVOLVED-5G objectives.

2.1.3 Cookiecutter

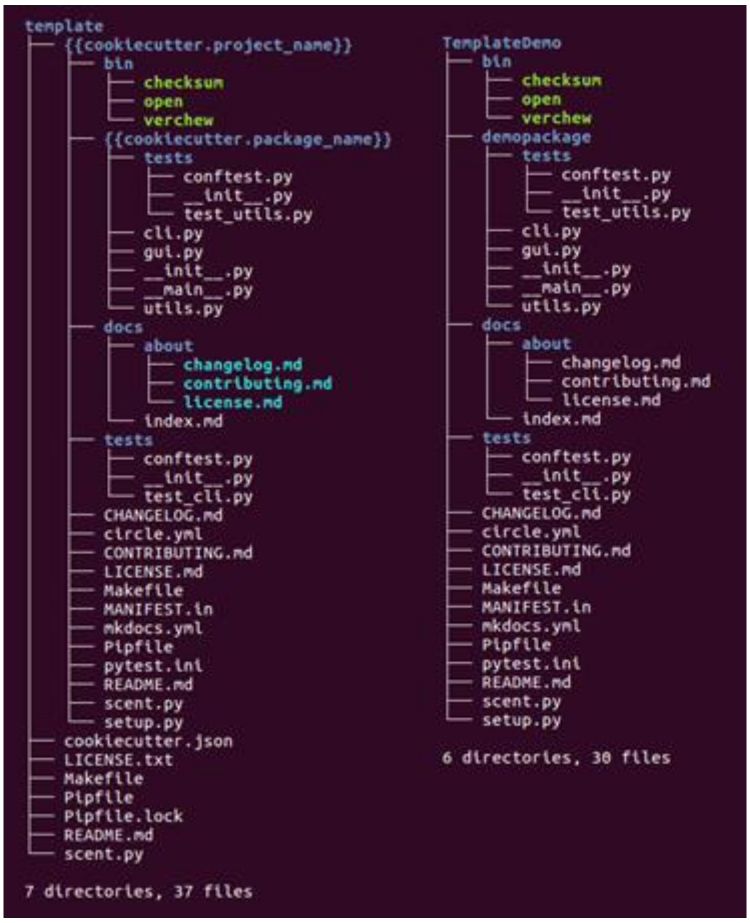
Cookiecutter is a CLI tool (Command Line Interface) based in Python that allows creating projects and repositories from specific software project templates (i.e., “cookiecutter”). Developers only have to modify a predefined template for a specific type of software project (previously prepared), introducing some required inputs, and run Cookiecutter command line utility to create a repository with the structure and files required. Thus, the use of Cookiecutter and its associated templates simplifies very significantly the effort of constructing the software repository structure, including folders and files.

Cookiecutter uses a specific templating system, Jinja2 [6] that allows to easily replace or customize folder and file names in the structure defined in a Jinja2 template, as well as file content. Jinja2 has a very simple high-level syntax that allows to define and modify templates specifying the folder structure of a software project in a very easy, friendly and intuitive manner.

Cookiecutter can be used either with a Git repository, with a folder or even with a Zip file. Although Cookiecutter is programmed in Python, the tool supports the creation of templates for software projects on another computer programmer languages.

The advantages of the use of this SDK tool are mainly time saving at constructing new repositories, prevention of any potential omission of mandatory files by mistake, such as Readme or Changelog, and to significantly ease the effort of development of code repositories, lowering the entry level to new collaborators on a software project. Furthermore, the use of Cookiecutter represents a way to enforce standards and best programming practices as invites developers to follow established rules (e.g., on testing and coverage checks, on the preparation of documentation or on the use of a specific naming syntax).

Cookiecutter is selected to be used at the EVOLVED-5G development phase in order to support developers on the NetApp production by generating a code repository that contains all the necessary folders and files for the creation of a NetApp.



```
template
├── {{cookiecutter.project_name}}
│   ├── bin
│   │   ├── checksum
│   │   ├── open
│   │   └── verchew
│   ├── {{cookiecutter.package_name}}
│   │   ├── tests
│   │   │   ├── conftest.py
│   │   │   ├── __init__.py
│   │   │   └── test_utils.py
│   │   ├── cli.py
│   │   ├── gui.py
│   │   ├── __init__.py
│   │   ├── __main__.py
│   │   └── utils.py
│   ├── docs
│   │   ├── about
│   │   │   ├── changelog.md
│   │   │   ├── contributing.md
│   │   │   └── license.md
│   │   └── index.md
│   ├── tests
│   │   ├── conftest.py
│   │   ├── __init__.py
│   │   └── test_cli.py
│   ├── CHANGELOG.md
│   ├── circle.yml
│   ├── CONTRIBUTING.md
│   ├── LICENSE.md
│   ├── Makefile
│   ├── MANIFEST.in
│   ├── mkdocs.yml
│   ├── Pipfile
│   ├── pytest.ini
│   ├── README.md
│   ├── scent.py
│   └── setup.py
├── cookiecutter.json
├── LICENSE.txt
├── Makefile
├── Pipfile
├── Pipfile.lock
├── README.md
└── scent.py

7 directories, 37 files
```

```
TemplateDemo
├── bin
│   ├── checksum
│   ├── open
│   └── verchew
├── demopackage
│   ├── tests
│   │   ├── conftest.py
│   │   ├── __init__.py
│   │   └── test_utils.py
│   ├── cli.py
│   ├── gui.py
│   ├── __init__.py
│   ├── __main__.py
│   └── utils.py
├── docs
│   ├── about
│   │   ├── changelog.md
│   │   ├── contributing.md
│   │   └── license.md
│   └── index.md
├── tests
│   ├── conftest.py
│   ├── __init__.py
│   └── test_cli.py
├── CHANGELOG.md
├── circle.yml
├── CONTRIBUTING.md
├── LICENSE.md
├── Makefile
├── MANIFEST.in
├── mkdocs.yml
├── Pipfile
├── pytest.ini
├── README.md
├── scent.py
└── setup.py

6 directories, 30 files
```

Figure 1 Cookiecutter usage example

An example of the use of Cookiecutter is depicted in Figure 1. On the left side it is shown the template structure, whereas on the right side it is displayed the resulting repository after running the Cookiecutter utility.

2.2 Verification Tools

Software verification process consists of checking in a testing environment if the software developed performs the functionality expected. In 5G programmability framework, and when the software under verification refers to NetApps, the verification process targets the functional testing of the NetApp interaction with a 5G network, with the aid of a 5G testbed or emulator. In this context, and given the NetApp definition, the verification consists on checking the correct interfacing with a 5G Core as a first testing step prior software validation on a real environment (i.e., a real 5G network).

Verification tools are frequently included as a part of Software Development Kits (SDKs). Some relevant verification tools for 5G programmability have already been described in Section 2.1 as a part of the SDKs of SONATA and 5GTANGO:

- **5GTANGO emulator** [25]
- **SONATA NFV-based emulator** (*son-emu*) [9]
- At some extent, other SDK tools that can be used for verification purposes in combination with an emulator (e.g. *son-monitor*)

In EVOLVED-5G project, it has been developed a NEF emulator (see section 7.2.4) and a CAPIF tool (see section 7.2.3) to assist the NetApp verification by providing the required 5G exposure capabilities. The NetApp verification process itself is conducted by the use of a CI/CD pipeline based in Jenkins [21] (see section 4.2) in conjunction with the open utility Robot Framework [22] to automatize test execution.

From one hand, the EVOLVED-5G NEF emulator enables the emulation of 5G Network Core functionality, allowing to test the interaction of a NetApp with 5GC northbound and southbound APIs. From the other hand, the CAPIF tool allows to verify this interaction following the APIs, calls and procedures specified for the Common API Framework proposed by 3GPP as an attempt to standardize 5GC exposed functionality and facilitate the realisation of 5G programmability. The CAPIF tool is an implementation of the CAPIF APIs from the 3GPP specification [23], and it is both employed in the EVOLVED-5G verification and certification processes to check the NetApp compliance with CAPIF.

The concept beyond the NEF emulator shares some similarities with the 5GTANGO and SONATA emulators. Though, despite of taking some ideas and inspiration on these preceding utilities, the EVOLVED-5G NEF emulator is significantly different and more potent, as considers different 5G platforms, incorporates diverse types of functionalities, encompasses functions from a wider set of 5GC and it is focused on a broader and more ambitious aim (i.e. NetApp verification –a more complex process than just 5G vApp verification).

Differently to the NEF emulator, the CAPIF tool concept, implemented in EVOLVED-5G, is completely novel and it has no precedents in the state of the art.

2.3 Validation Tools

The validation of 5G applications is generally understood as the process of testing and validating their characteristics and functionalities against a 5G environment.

Key initiatives to provide 5G end-to-end validation infrastructures have been endorsed in the frame of H2020 R&D programme. The most remarkable ones are the 5GENESIS, 5G EVE and 5G-VINNI projects, conducted under the 5GPPP umbrella. Each of these initiatives provided different large-scale end-to-end validation infrastructures in Europe with different characteristics, as well as its own validation methodology. These infrastructures provide an adequate level of openness in order to make it possible for vertical industries to test their innovative 5G business cases.

EVOLVED-5G takes advantage of the infrastructure and ideas coming from these projects, in particular by making use of two 5GENESIS validation platforms (the Athens and Málaga 5G platforms), as well as by expanding the 5GENESIS validation methodology.

Within the platforms, several tools are used for supporting the Validation process. These include general purpose tools developed by third parties, more specific applications created in the context of previous H2020 R&D projects or even custom scripts developed ad-hoc for controlling certain parts of the infrastructure.

The EVOLVED-5G Validation environment is based on tools created within the context of the 5GENESIS project, such as the Dispatcher and ELCM (Section 5) for the top-level component orchestration, while leveraging the use of general-purpose tools like OpenTAP [29] custom scripts and vendor specific management tools for controlling separate elements of the testbed during the validation.

2.4 Open repositories

The use of repositories for software development is a general practice, as it allows software storage, version control and distributed access to the development results. Repositories allow the storage of code and binaries, version tracking and collaborative work while programming projects of large complexity from a team of developers. Furthermore, the use of repositories facilitates the access and openness of projects and software results. There are two main types of repositories considering the type of software element stored:

- **Code repositories**, which store the source code of a software project.
- **Artifact repositories**, which contain binaries built from the source code developed in a software project.

These two main types of repositories[1] are employed in the EVOLVED-5G project: a source code repository (GitHub) for the development of NetApps, and an open repository for NetApp binaries (Artifactory). The Open Repository plays an important role on any of the NetApp lifecycle phases. On the one hand this repository stores the fruit of NetApp programming from the development phase. On the other hand, it is essential in the processes of NetApp verification, validation, certification and publication by allowing access to the NetApp binaries (in general Docker images) and to an associated NetApp status file.

2.4.1 Code repositories

Repositories integrate version control systems and allow access to authorized users. The most popular software for code version control is **Git** [30], created by Linus Torvalds with the aim of allowing collaborative and coordinated development of complex software projects (e.g Linux kernels). Git is a free and open-source distributed version control system that allows for tracking changes in any set of files. It allows collaborative development, enabling concurrent edits from multiple users while maintaining data integrity and supporting distributed, non-linear workflows (e.g., thousands of parallel branches running on different systems). Its main advantage over other control version systems is the possibility of creating branches of a software project at any version point, allowing to keep those branches isolated or to merge them with the main one. Git has many different implementations and variants.

Some Git repositories are offered as a free online service, such as GitHub [31] GitLab [26] or BitBucket [32]. Regarding local Git implementations, in addition to GitLab, which can also be self-hosted, a very popular one is Gogs [33], a software that should be installed, configured and managed by a local administrator. The main advantage of Gogs over other Git servers is the low complexity of its administrative management, considerably simplifying the Git administrator task.

2.4.2 Artifact repositories

Repositories for artifacts are typically specific for a concrete type of binary (e.g., Linux distros, Docker images, APKs) and they allow for cataloguing binary versions. An exception to that high degree of specificity is Artifactory [38], designed to store any type of binary as a universal repository with excellent scalability. Artifactory supports Kubernetes [39], which is currently

[1] A public open repository could be either, a code (GitHub) repository or an artifact (jfrog) repository. For the sake of clarification, EVOLVED-5G project considers, the Open Repository as an artifact repository where only NetApps binaries and its associated files are stored, and a code repository as a software development repository where the NetApps code, different tools and any documentation are stored.

considered the de-facto orchestration tool for automating deployment, scaling and management of containerized applications and microservices.

Regarding Docker environments, there are specific open repositories for the storage of Docker images, namely Docker Registries. A Docker Registry [34] can be installed on a server or a local machine to allow access to authorised users. DockerHub [35] is an example of an online free Docker Registry service, provided by Docker, which allows to share publicly or among a restricted team container images, search them, upload new versions and download them. Frequently is employed for performing a free public release of Docker images from a software project.

3 EVOLVED-5G FACILITY

3.1 GENERAL DESCRIPTION

The EVOLVED-5G facility has been designed to provide the necessary tools, processes and functionalities for the development, validation and certification of NetApps, as well as any supporting network infrastructure (e.g., a 5G-NPN) and mechanisms for market releasing and collaboration (e.g., a Marketplace). Thus, the design of the facility encompasses all the required interactions and connections among all the architectural components that support the realisation of the NetApp lifecycle, providing a set of tools that are imperative for the lifecycle to happen. The facility is organized around the idea of environments, which are Tier-1/Core architectural components of the facility and are specifically suited for supporting each one of the different phases of the NetApp lifecycle, along with other functional blocks (Tier-2 architectural component) and tools (Tier-3 architectural component) that provide environments with the expected functionalities and support their implementation, respectively.

As it has been mentioned the EVOLVED-5G project is composed of different functional blocks as well as different environments. Each environment has a variety of tools which enables the lifecycle of the NetApp. Figure 2 presents a mapping among the tools and functional blocks belonging to each environment, as well as to which WP3 task it belongs. The two main environments relative to Task 3.1 (Workspace), Task 3.2 (Validation) and Task 3.3 (integration and infrastructure connectivity) are presented. Certification and Marketplace is not represented since it will be detailed in Deliverable D3.2. Some tools have been developed and implemented specifically for the project, while some others are commercial solutions which are used to enable the lifecycle of the NetApp. Hereafter is briefly described the status of the implementation for each tool and functional block.

The first environment in the lifecycle of the NetApp is the Workspace, the functional blocks involved are SDK tools, repositories (GitHub and Open Repository), the CI/CD Services, Verification tools and tests, and an emulation environment (CAPIF and NEF). Within these functional blocks, some tools have been developed to serve as enabler during development, verification and validation of the NetApp. The status of such developments and implementations are the followings:

- SDK tools, it has been implemented a template, libraries for the 5G APIs and a Command Line (CLI) tool to create the NetApp based on such template and run the pipelines from the CI/CD.
- The CI/CD is based on a Jenkins solution, the implementation carried out pipelines such as build, deploy or destroy for NetApps, as well as a pipeline to perform a static code analysis of the NetApp source code. Is subject to further implementations, e.g., new pipelines for verification purposes.
- Robot framework, is the tool where the different verification tests are defined. Such verification tests are defined but subject to be implemented.
- GitHub repository, is a version control repository where actually the code of NetApps, instructions and tools such NEF and CAPIF are stored. Is subject to also store reports for the different phases of the NetApp lifecycle.
- Open Repository, after successfully pass the different phases of its lifecycle the NetApp binaries are stored, specific folders have been created to store them.

- CAPIF and NEF, are tools implemented for verification purposes but not developed within the scope of the Workspace.

The second environment is the Validation, the functional blocks involved are Validation tools, repositories (GitHub and Open Repository), Validation tests, the CI/CD services, within the Validation environment are Malaga and Athens platforms which, enable to deploy Time-Sensitive Networking (TSN), TSN controller, 5G exposure functions, 5G RAN and 5G core. The status is the following:

- Validation tools, such tools have been implemented in 5GENESIS and are subject to be used to validate the NetApp in both platforms.
- The CI/CD is not developed under the Validation but rather used to launch the pipeline to start the validation phase.
- The Open Repository, the Validation environment is using this repository to store the validated NetApp after the validation phase has been completed. To start the validation phase, the binaries of the verified NetApp are selected from this repository.
- Validation tests, different templates (Test cases) are in place but is subject to implement then after gathering feedback from the SMEs to define what to validate for each NetApp.
- Athens and Malaga platforms, are used to validate the NetApp along with the potential vApp in a real environment. As mentioned, on top of such platforms, some of the functional blocks deploy are the followings:
  - Time-Sensitive Networking (TSN), a current setup is already in place in the Málaga platform but is subject to finalise a full implementation and perform some tests. Also, a smart controller to ease monitoring and configuration of endpoints will be included.
  - 5G exposure functions, CAPIF and NEF will expose the APIs the NetApp must consume for verification, validation and certification phases.
  - 5G RAN and 5G Core, hardware and software required to deploy a fully functional 5G network. More details and improvements provided in Section 7.2.1 and Section 7.2.2.

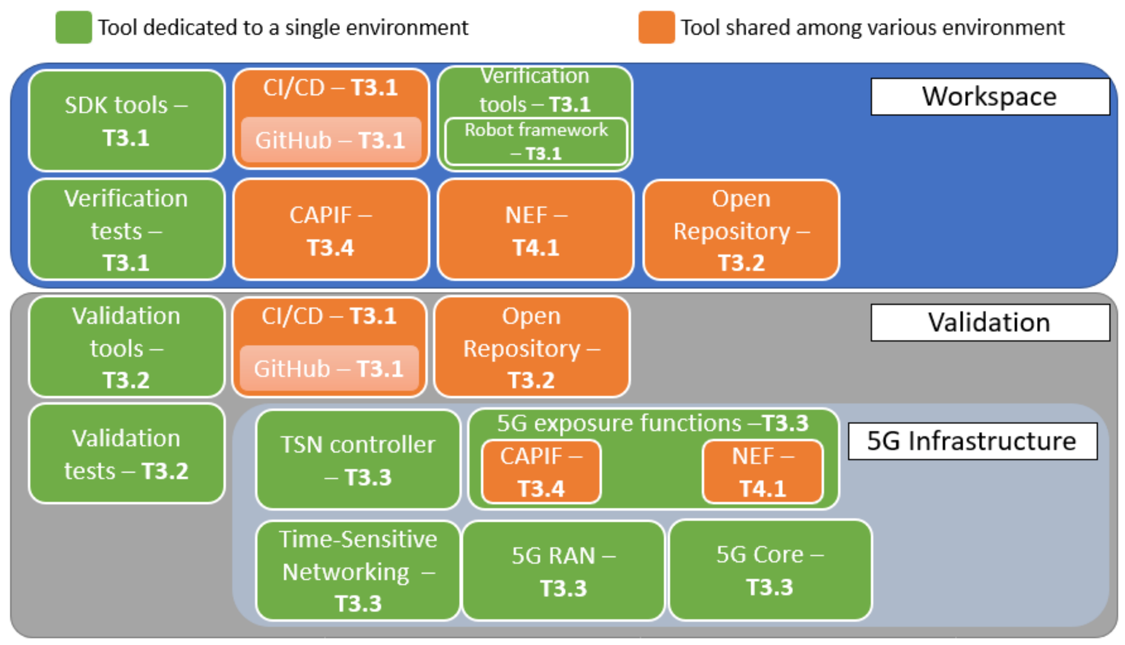


Figure 2 Overall view of the current elements composing the different environments of the EVOLVED-5G facility.

3.2 NETAPPS PRINCIPLES AND IMPLEMENTATION

The EVOLVED-5G NetApp concept has been described in previous deliverables D2.1 [37] and D2.2 [55], although a brief description is included for the benefit of the reader, before presenting the architecture.

A NetApp is a software component designed to interact with the control plane of a mobile network by consuming exposed APIs (e.g., Northbound APIs of 5G core) in a standardized and trusted way (i.e., for a 5G network a NetApp should be CAPIF [36] compliant) in order to compose services for the vertical industries. NetApps can provide services to a vertical system either as an integrated part of the Vertical Application (vApp) (Standalone) or expose APIs, which are referred to as business APIs in order to interface with vApps, increasing by this way the reusability of the NetApp by different vApps (Non-Standalone). A more in detail explanation will be given in further deliverables related to WP4.

In respect to the implementation of the NetApps the main approach refers to the request of the vApp to the NetApp which in turn is translated to one or more requests of the NetApp to the 5G API Core. The reply that comes from the 5GC includes the data that have been requested and are in turn forwarded to the vApp.

Narrowing down the different implementation approaches that have been identified by the consortium and are described in detail in D2.2 [55], the container-based REST API Deployment prevails in terms of advantages and flexibility. To that end the NetApps to be developed under the scope of EVOLVED-5G, will be based on a container image, offering REST API endpoints (a.k.a. Business APIs) to the vApp as well as to the 5G system for subscription-based events when applicable.

3.3 HIGH LEVEL ARCHITECTURE

The EVOLVED-5G reference architecture is separated in several functional blocks that encapsulate the necessary components and functionality required for each of the phases in the NetApp's lifecycle.

Very briefly, the **Development of the NetApp is the creation of its functionality by making use of the SDK**, while the **Verification is the process for checking compliance with the basic NetApp requirements, while making correct use of the interfaces in a synthetic environment**.

The NetApp development and verification phases are handled by the **Workspace Environment**. This environment contains the SDK, which includes the necessary tools for the NetApp development, and the CI/CD Services that provide specific *pipelines* for the execution of the Verification.

Successfully verified NetApps can be subject to the Validation process, which consist on the execution of functional and non-functional tests that measure the performance of the NetApp when it's used in conjunction with a particular Vertical Application. The components required for the execution of the Validation are contained in the **Validation Environment**. This environment is built on top of the **5G-NPN Infrastructure**, which is composed by all the equipment available in the Validation platforms and may be extended depending on the needs of the Vertical App used during the tests.

The **Certification** environment is in charge of providing the methodology and additional functionality for performing additional tests on the NetApp, with the goal of assuring a certain level of excellence and the compliance with the policies of the **Marketplace**, before a NetApp can be published and made available to end-users. This may include a manual, administrative process as well as additional automated tests that are also expected to be run using the validation **Infrastructure**.

The aforementioned **Marketplace** is the interface where end users have access to ready to use, certified NetApps. Encompassing all the environments, is located the **Open Repository**, where the NetApp is stored and made accessible to the different components (such as the CI/CD services, validation environment and certification environment among others) during their whole lifecycle. The Open Repository also contains the necessary metadata for tracking the status of the NetApp, as well as any additional reports generated during the execution of the different processes that a NetApp may undergo.

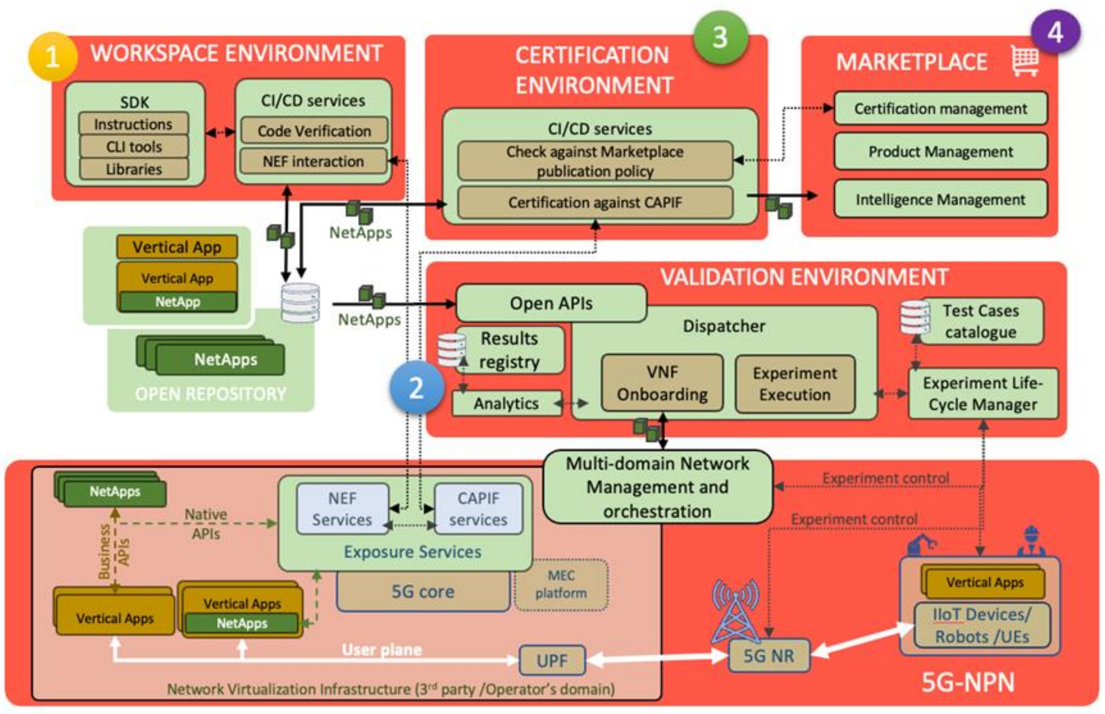


Figure 3 EVOLVED-5G reference architecture.

EVOLVED-5G will make use of existing solutions, either proprietary or open-source, for the implementation of the different components, as well as develop additional components for cases where ready to use solutions are not available. As an example, the following is a non-exhaustive list of solutions in each category:

- Commercial tools: GitHub (source control), Artifactory (artifact storage), Jenkins (continuous integration automation).
- Open-Source tools: Cookiecutter (project templating), Open5Genesis framework (testbed level automation).
- New tools: SDK-CLI, Validation reports generator

4 WORKSPACE

4.1 MAIN OBJECTIVE

The main objective of the Workspace environment is to provide developers with the necessary software tools and instructions for NetApp development while making this process easy, quick and convenient. To do so, it goes through two phases; namely, the NetApp development and verification phases.

This environment also provides developers with the appropriate code storage repository (GitHub) where tools and instructions are stored as well as the code of the NetApps generated during the development phase, and an artifact repository (Open Repository) where the binaries coming from the verification phase will be stored. As soon as a NetApp is developed, it can be uploaded to the repository and the developer can start the verification process, as described in Section 4.3.2.

4.2 COMPONENT DESCRIPTION

The workspace is one of the core architectural environments of the NetApp ecosystem. As defined in D2.1, three functional blocks (providing different functionalities depending on the phase of the NetApp) are conceived: The SDK, repositories and CI/CD (see Figure 3). The interconnection and communication of these blocks will enable the different functionalities of the Workspace, e.g., creating remote repositories, making use of a template, among others, such functionalities are explained in detail in Section 4.3. Each of these functional blocks will perform different actions, which are responsible to accomplish the development and verification phases of the NetApp lifecycle. The functional blocks of the workspace are as follows:

During the Development phase, the functional blocks involved are:

- **SDK**: it provides a toolkit to facilitate the development of NetApps. The toolkit is composed of: a set of instructions where developers can find all the information related to the creation of NetApps; a CLI tool, which allows to create the repository of each NetApp from the template; subsequently with the help of the CI/CD the developer is running the pipelines (build, deploy and destroy) to check that the NetApp works correctly. Each pipeline will return a unique ID of the executed pipeline, therefore the status can be checked, and a result is displayed, indicating to the developer whether it has been successful or unsuccessful; a template, where the developer can find the structure of folders and files where the NetApp code is stored; and the libraries, which will allow the interaction of the developers' NetApps with the 5G APIs, either from the emulator or a real 5G Core, providing an abstraction towards the API.
- **Repositories**: two types of repositories are provided:
  - GitHub, acts mainly as a code repository, where the NetApps are generated by developers and are stored along with their files. In addition, you can find all the tools, instructions and templates.
  - Open Repository, based on Artifactory, jfrog technology [38], where the NetApps images are stored.

During the Verification phase, the functional blocks involved are:

- **CI/CD**: it is a centralised server in charge of testing the operation of the NetApps from a series of pipelines in charge of building, deploying among others the NetApp to verify its correct operation.
- **Emulation:** The emulation environment makes part of the Verification phase since NEF emulator and CAPIF Core Function Tool (see Section 7.2.3) are the main tools of which the NetApp is verified against. Such tools are handled within the Workspace environment but are not deploy on it, please refer to Figure 3.

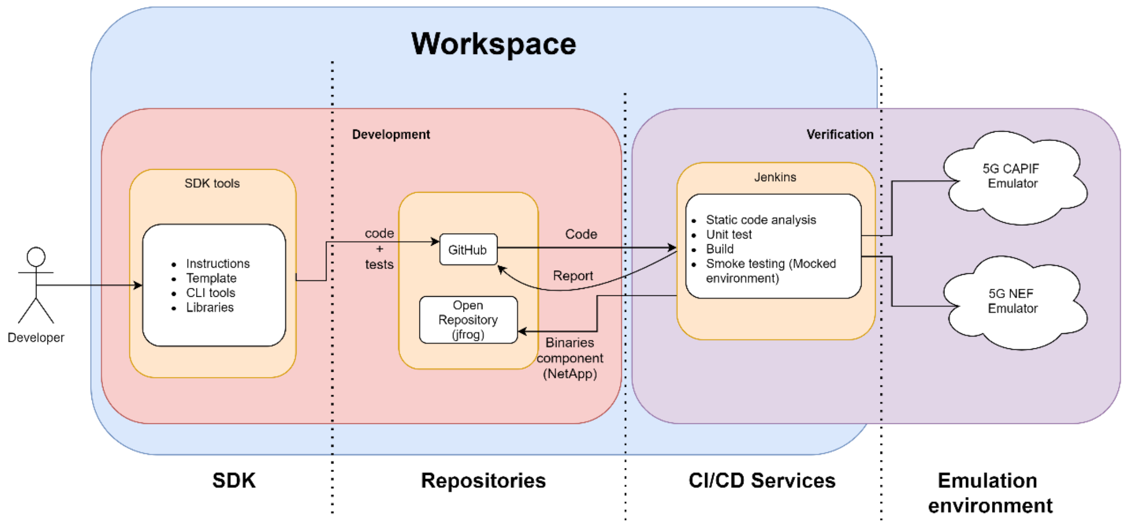


Figure 4 Workspace architectural components

Figure 4 shows the functional blocks of the Workspace previously described. While Figure 5 shows a workflow related to the different functional blocks in the Workspace Environment, from the first step of a NetApp development until the verification is finished and the NetApp is stored to the open repository.

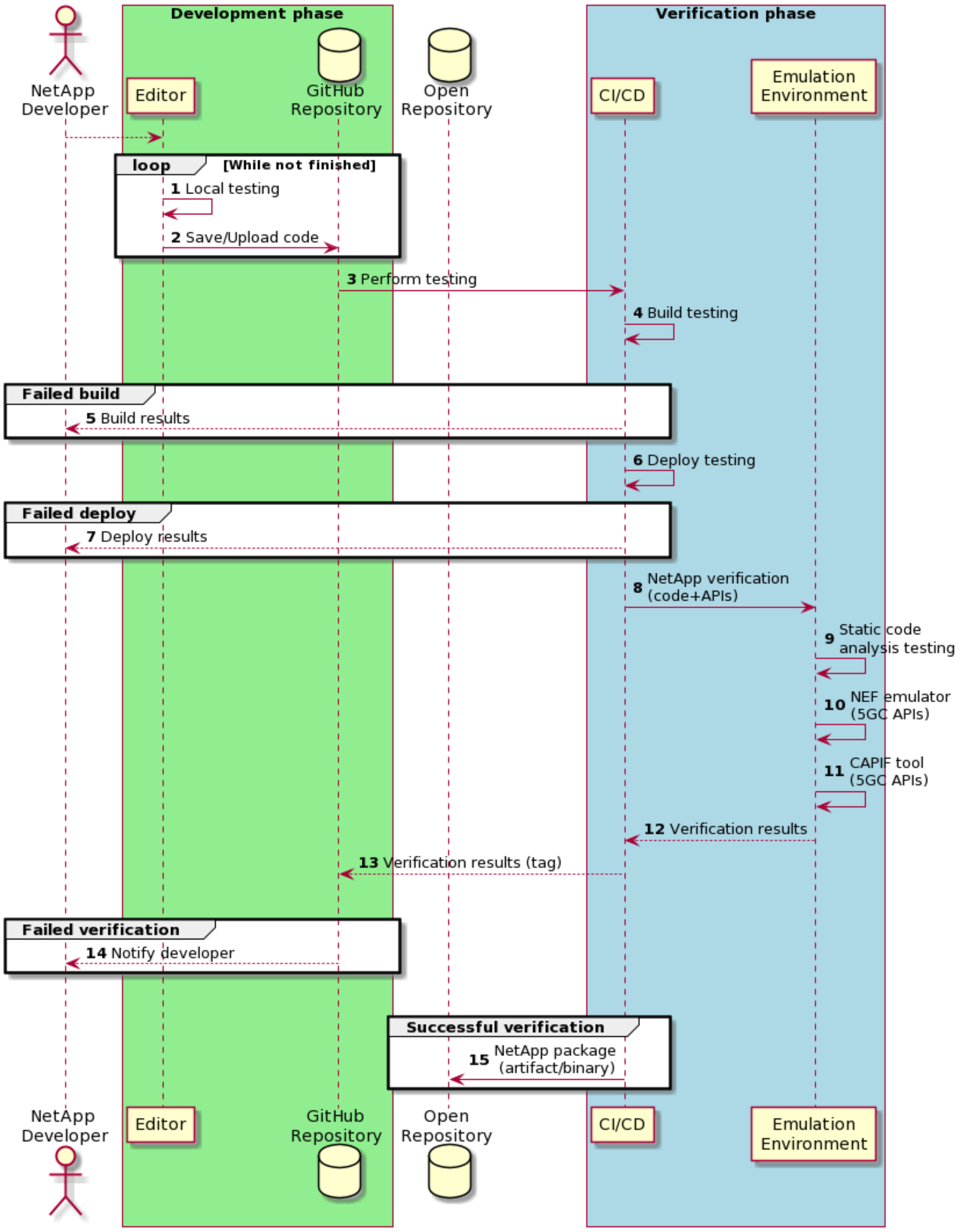


Figure 5 Workflow of the functional blocks of the workspace

All the functional blocks comprised within the Workspace., can be seen in Figure 5. First, a NetApp developer can use their own preferred editor, to start the process of developing the core functionalities of the NetApp (*step 1*). Once the NetApp development is finished, the NetApp developer uses the tools provided in the SDK to upload to GitHub and to provide the NetApp with 5G interaction capabilities (*step 2*).

Once this process is finished, the NetApp developer is able to start verifying the NetApp by making use of the CI/CD (*step 3*), which will be explained in more details in followings sections. To that end, the NetApp is built and deployed to verify its correct functioning (*step 4 & 6*). Both at the build or deploy the NetApp developer will receive a report (*step 5 & 7*) indicating whether the NetApp has worked as expected.

Then (*step 8*) a more exhaustive verification phase start, by analysing the code (*step 9*), and verifying the NetApp against the NEF emulator (*step 10*) and CAPIF tool (*step 11*), once the verification phase is finished the report will be stored on GitHub (*step 13*), and the NetApp Developer will be notified so that the result can be consulted it. If the NetApp is working as expected an image (artifact) of the NetApp will be stored in the Open Repository (*step 15*).

4.2.1 Development Phase

The Development phase is one of the Workspace main functional blocks. The objective of the Development phase is to help developers to develop the NetApp by providing the necessary means, e.g., within the scope of the SDK tools, the developers will be provided with some libraries to enhance the NetApp with the 5G capabilities which in following phases will be tested.

The purpose of the Development phase is to provide a fully NetApp with a readiness level to be sent to the next phase which will be verification.

4.2.2 Verification Phase

The Verification phase is formed by the CI/CD Services and the Emulation Environment functional blocks. The CI/CD services is in charge to run the pipeline; the Emulation Environment is responsible of performing the defined tests for verification.

As described in Figure 4, the Verification phase along with the Development phase is one of the first steps in the lifecycle of a NetApp. The verification phase are functional tests such as usability or smoke testing among others, which verifies the proper functionality of a software, in this case, the NetApp.

The objective of the Verification phase is to guarantee that the NetApp is able to operate correctly in a synthetic environment before releasing it to the Validation environment where functional and non-functional tests are provided. During the verification phase only the NetApp itself is considered and not any potential vApp that could be associated to it.

The primary objective is to guarantee a correct NetApp operation, the successful verification means the NetApp has passed all the tests proposed including NEF emulator and CAPIF tool and it is ready to pass to next step in its lifecycle. To realize this goal, a common base NetApp API enabling the communication of the 5G infrastructure with NetApps is required for the verification tests to be able to run automatically and horizontally for all the NetApps developed in the scope of the project.

4.3 IMPLEMENTATION

4.3.1 Development Phase

This section describes the implementation that has been done in the workspace environment to achieve the creation of a NetApp from scratch.

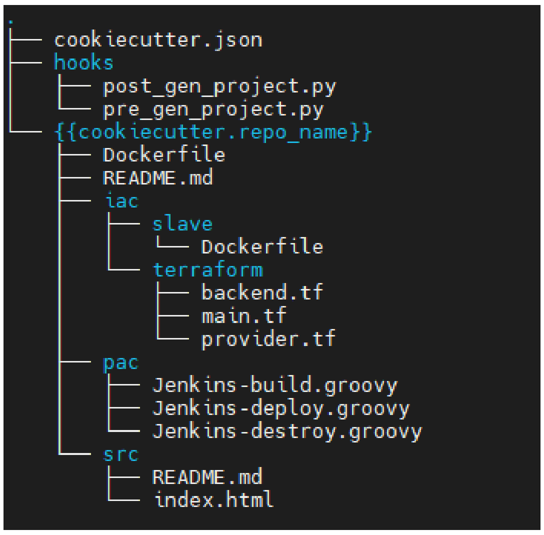
The goal of the Workspace is to provide means to developers for implementing a NetApp. It has been decided to create an organization in GitHub called EVOLVED-5G [40], where a series of tools will be hosted to achieve this goal. In such organization there will be multiple repositories, each with functions that will be discussed below. All repositories within this organization obey a branching model as follows:

- Master: Main branch of the project. This branch has to be labelled as protected and default branch.

- Develop: Branch from which the improvements/upgrades are made, comes from Master. This branch has to be labelled as protected.
- Feature: Branch that comes from Develop, used to develop new enhancements, once it is finished it will be merged into Develop, and it will be removed once it has been merged.
- Hotfix: Branch coming from Master, used for a bug that needs to be fixed urgently. It will be removed once it has been merged.
- Release: Branch used to make final changes before checking out a new version of the repository you are working on. It will be removed once it has been merged.

GitHub is a cloud-managed code version control platform using Git, enabling features that report issues or merge remote repositories (issues, pull request, etc.), as well as best practices provided by Git.

From here, this section describes every step to follow in order to create a NetApp. First, it is necessary to define a file structure, such file structure is inherited from a "Template", this is one of the SDK tools stored in a GitHub repository [42]. The template make use of Cookiecutter, a CLI tool described in Section 2.1.3, to perform the configuration of a NetApp, meaning, the file structure will be created locally in developers´ computer, as well as a new remote repository will be created in GitHub, and a push will be made to store the NetApp. The structure of the template repository is as follows:



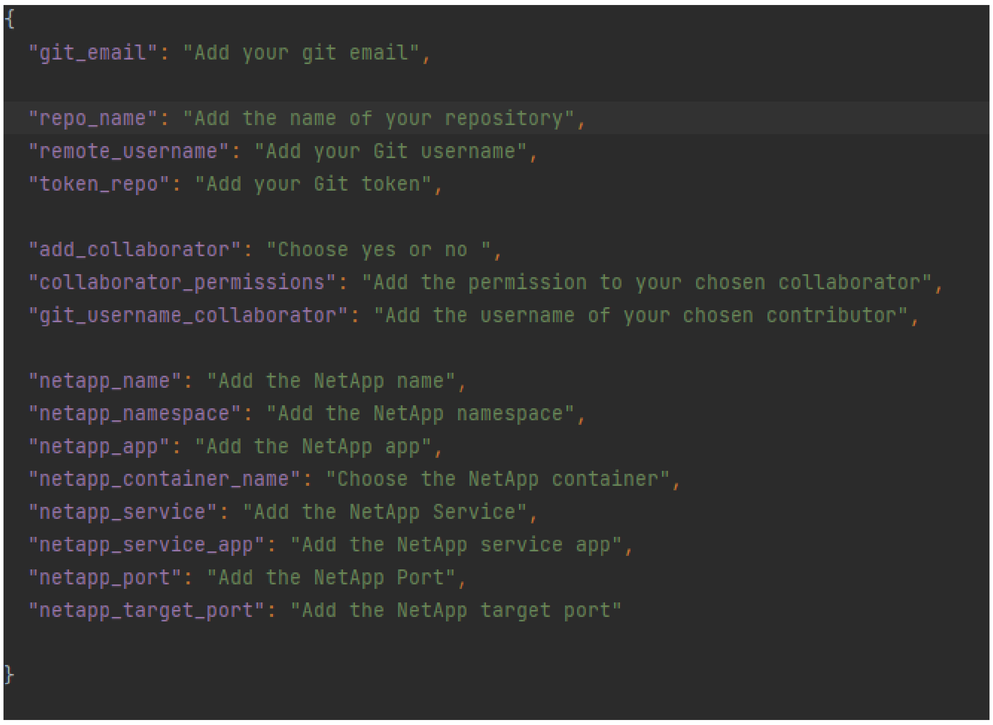
```
.
├── cookiecutter.json
├── hooks
│   ├── post_gen_project.py
│   └── pre_gen_project.py
└── {{cookiecutter.repo_name}}
    ├── Dockerfile
    ├── README.md
    ├── iac
    │   ├── slave
    │   │   └── Dockerfile
    │   └── terraform
    │       ├── backend.tf
    │       ├── main.tf
    │       └── provider.tf
    ├── pac
    │   ├── Jenkins-build.groovy
    │   ├── Jenkins-deploy.groovy
    │   └── Jenkins-destroy.groovy
    └── src
        ├── README.md
        └── index.html
```

Figure 6 Template structure

As you can see from Figure *6* there is a tree of folders and files, there are two main folders, which are *hooks* and *{{cookiecutter.repo_name}}*. The first one (*hooks*) contains two python files that take care of creating a remote repository on EVOLVED-5G's GitHub and pushing the configured NetApp. The second one (*{{cookiecutter.repo_name}}*) contains all the directories and files that will be configured through inputs, asked to the developer, to create and setup the NetApp. Within this second folder under the terraform name, the developer will find an infrastructure automation tool known as Terraform, which will build and version the NetApp infrastructure.

Terraform, is a tool for building, changing, and versioning infrastructure safely and efficiently. The terraform configuration files describe the components needed to create and run the NetApp in this case. Terraform generates an execution plan that describes what it will do to achieve the desired state, and then executes it to build the described infrastructure. These files will also be populated through the inputs that are requested when the SDK-CLI tool developed is executed.

In addition, to the main directories and their files, you can see in *Figure 7* a json file, called *cookiecutter.json*, where all the variables (inputs) that must be filled in when SDK-CLI tool is executed are configured. The variables are as follows:

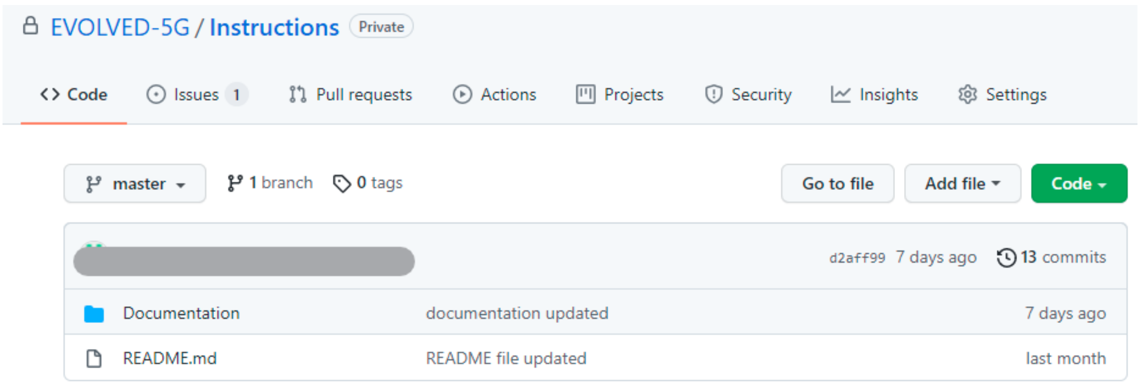


```
{
  "git_email": "Add your git email",

  "repo_name": "Add the name of your repository",
  "remote_username": "Add your Git username",
  "token_repo": "Add your Git token",

  "add_collaborator": "Choose yes or no ",
  "collaborator_permissions": "Add the permission to your chosen collaborator",
  "git_username_collaborator": "Add the username of your chosen contributor",

  "netapp_name": "Add the NetApp name",
  "netapp_namespace": "Add the NetApp namespace",
  "netapp_app": "Add the NetApp app",
  "netapp_container_name": "Choose the NetApp container",
  "netapp_service": "Add the NetApp Service",
  "netapp_service_app": "Add the NetApp service app",
  "netapp_port": "Add the NetApp Port",
  "netapp_target_port": "Add the NetApp target port"

}
```

Figure 7 Inputs to create and setup a NetApp.

Right after the previous steps, an empty NetApp will be ready, configured and set up to for the developer to start populating code.

To assist the developer, the Instructions repository [45] contains a guide with all the steps to make a good use of the Workspace and achieve the objectives of creating and verifying a NetApp. The repository look as follows:



EVOLVED-5G / Instructions Private

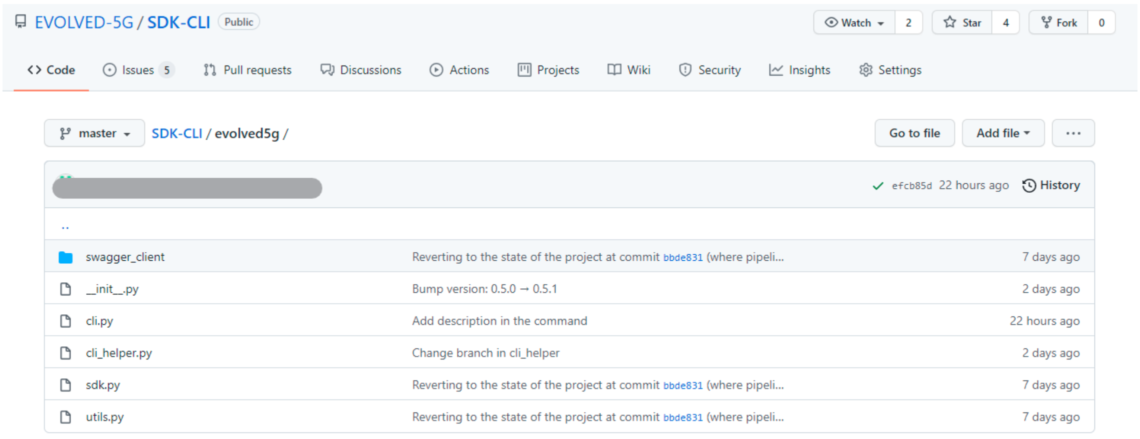
d2aff99 7 days ago 13 commits

Documentation	documentation updated	7 days ago
README.md	README file updated	last month

Figure 8 Instructions repository

Hereafter, are described the steps to install the package to create the NetApp by running some commands which make use of the repositories previously described. Along with it, some other important repositories will be described due to their involvement in the NetApp development.

The SDK-CLI repository [44] contains a generic command, followed by subcommands that can perform all the desired actions related to the NetApps. The directory "evolved5g" can be found within this repository, where the python files defining the commands for the creation of the NetApp and the verification of the NetApp can be found. In addition, there is a module called "sdk" where the content of the libraries can be found.


EVOLVED-5G / SDK-CLI Public

Watch 2 | Star 4 | Fork 0

Code | Issues 5 | Pull requests | Discussions | Actions | Projects | Wiki | Security | Insights | Settings

master | SDK-CLI / evolved5g / | Go to file | Add file | ···

[redacted] ✓ efcb85d 22 hours ago History

..

swagger_client	Reverting to the state of the project at commit bbde831 (where pipeli...	7 days ago
__init__.py	Bump version: 0.5.0 → 0.5.1	2 days ago
cli.py	Add description in the command	22 hours ago
cli_helper.py	Change branch in cli_helper	2 days ago
sdk.py	Reverting to the state of the project at commit bbde831 (where pipeli...	7 days ago
utils.py	Reverting to the state of the project at commit bbde831 (where pipeli...	7 days ago

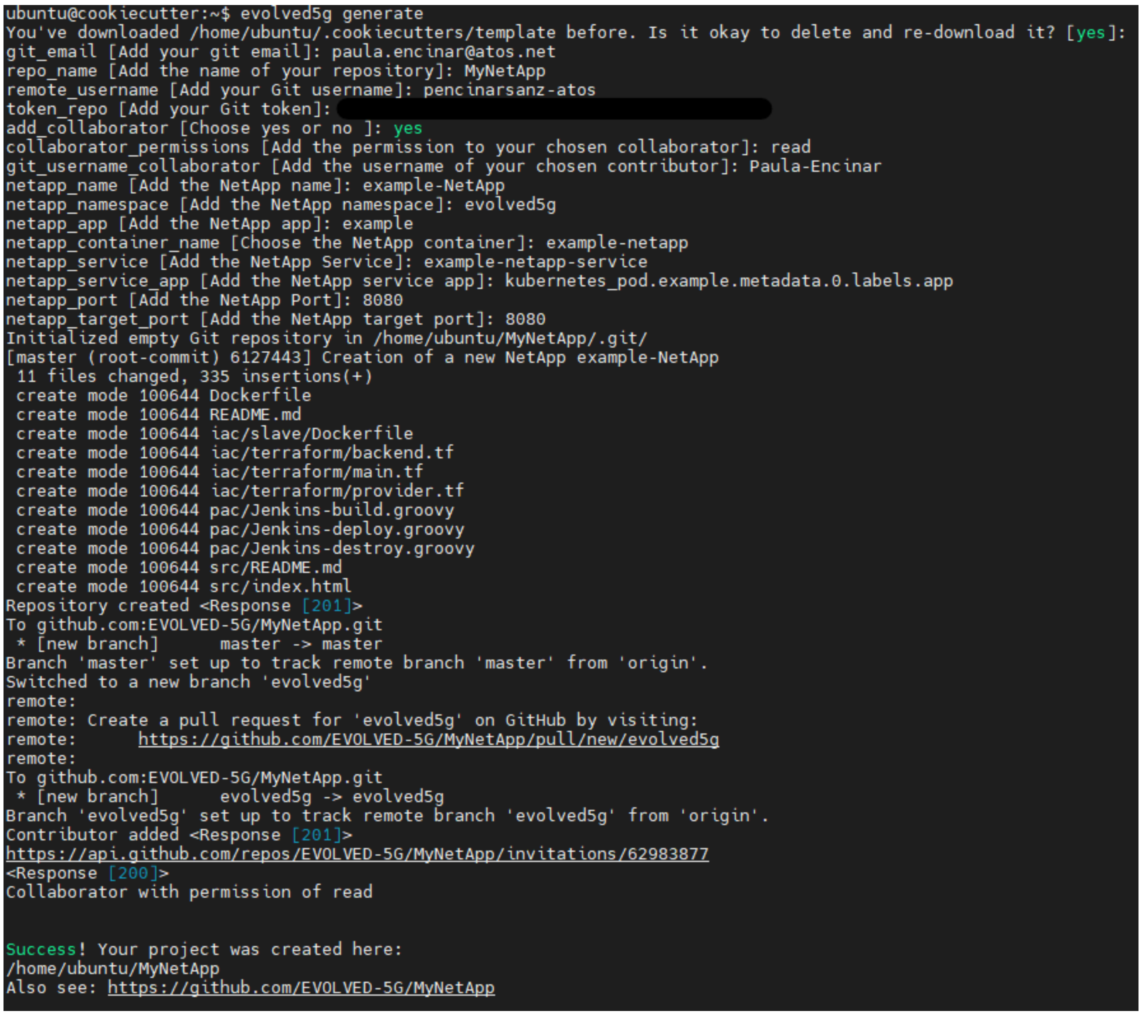
Figure 9 SDK-CLI repository

The SDK-CLI repository provides a CLI tool that can be used to manage NetApps via the terminal, by making use of commands such “*generate*” to launch the creation of the NetApp via the cookiecutter mechanism that was described above, and some other commands which will be explained in next paragraphs.

In this deliverable, the main installation process is described the main installation process and how to make use of the SDK-CLI tool. Two installation options are given to the developer, install the tool from source or via pip [43]. By typing “*pip install evolved5g*” the tool will be installed within seconds and ready to use. Some of the features already implemented in the tool are the followings:

- Creation of the NetApp by making use of the tools describe above.
- Run a verification pipeline (build, deploy or destroy), to verify the NetApp.
- Check a pipeline, to check the status of an executed pipeline.

By typing the command “*evolved5g generate*” the tool will start asking the developer some relevant information about the NetApp as displayed in the following picture:



```
ubuntu@cookiecutter:~$ evolved5g generate
You've downloaded /home/ubuntu/.cookiecutters/template before. Is it okay to delete and re-download it? [yes]:
git_email [Add your git email]: paula.encinar@atos.net
repo_name [Add the name of your repository]: MyNetApp
remote_username [Add your Git username]: pencinarsanz-atos
token_repo [Add your Git token]: [redacted]
add_collaborator [Choose yes or no ]: yes
collaborator_permissions [Add the permission to your chosen collaborator]: read
git_username_collaborator [Add the username of your chosen contributor]: Paula-Encinar
netapp_name [Add the NetApp name]: example-NetApp
netapp_namespace [Add the NetApp namespace]: evolved5g
netapp_app [Add the NetApp app]: example
netapp_container_name [Choose the NetApp container]: example-netapp
netapp_service [Add the NetApp Service]: example-netapp-service
netapp_service_app [Add the NetApp service app]: kubernetes_pod.example.metadata.0.labels.app
netapp_port [Add the NetApp Port]: 8080
netapp_target_port [Add the NetApp target port]: 8080
Initialized empty Git repository in /home/ubuntu/MyNetApp/.git/
[master (root-commit) 6127443] Creation of a new NetApp example-NetApp
 11 files changed, 335 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 README.md
 create mode 100644 iac/slave/Dockerfile
 create mode 100644 iac/terraform/backend.tf
 create mode 100644 iac/terraform/main.tf
 create mode 100644 iac/terraform/provider.tf
 create mode 100644 pac/Jenkins-build.groovy
 create mode 100644 pac/Jenkins-deploy.groovy
 create mode 100644 pac/Jenkins-destroy.groovy
 create mode 100644 src/README.md
 create mode 100644 src/index.html
Repository created <Response [201]>
To github.com:EVOLVED-5G/MyNetApp.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
Switched to a new branch 'evolved5g'
remote:
remote: Create a pull request for 'evolved5g' on GitHub by visiting:
remote:      https://github.com/EVOLVED-5G/MyNetApp/pull/new/evolved5g
remote:
To github.com:EVOLVED-5G/MyNetApp.git
 * [new branch]      evolved5g -> evolved5g
Branch 'evolved5g' set up to track remote branch 'evolved5g' from 'origin'.
Contributor added <Response [201]>
https://api.github.com/repos/EVOLVED-5G/MyNetApp/invitations/62983877
<Response [200]>
Collaborator with permission of read


Success! Your project was created here:
/home/ubuntu/MyNetApp
Also see: https://github.com/EVOLVED-5G/MyNetApp
```

Figure 10 Inputs from generate command

Figure 10 shows, how the tool creates the NetApp skeleton based on the responses given by the developer after all the inputs have been filled in.

The next step is to complete the NetApp development and upload it to the repository. In this step, the SDK libraries can facilitate the development.

The SDK libraries are a set of python classes that can be used to perform specific requests to the 5G APIs. Such libraries can be found in the following repository [44]. In order to use them, a developer has to import the evolved5G.sdk python module in its code. By Q4 December 2021 the following two libraries are provided, LocationSubscriber and QoSAwareness.

LocationSubscriber, it is implemented to monitor the device locations, allowing the NetApp developer to receive a notification every time the device is being monitored, via the NetApp, connects to a different cell. A high-level implementation can be seen in Figure 11. In the examples folder of the SDK repository a python script [46] is provided to demonstrate its usage. To implement the LocationSubscriber library in the Netapp, the developer must follow the subsequent steps:

I. As a first step, it is necessary to start the NEF emulator (described in Section 7.2.4) to simulate an environment where a user device is moving between different network cells. NEF emulator already provides a scenario with three different user devices and four 5G cells, simulating the movement of the user devices.
II. The second step is to initialize a web server, which can either resides in the NetApp or in an external web server, to retrieve Event notifications from the NEF Emulator (or the

5G API) every time the device that is monitored changes its location (moves to another cell).

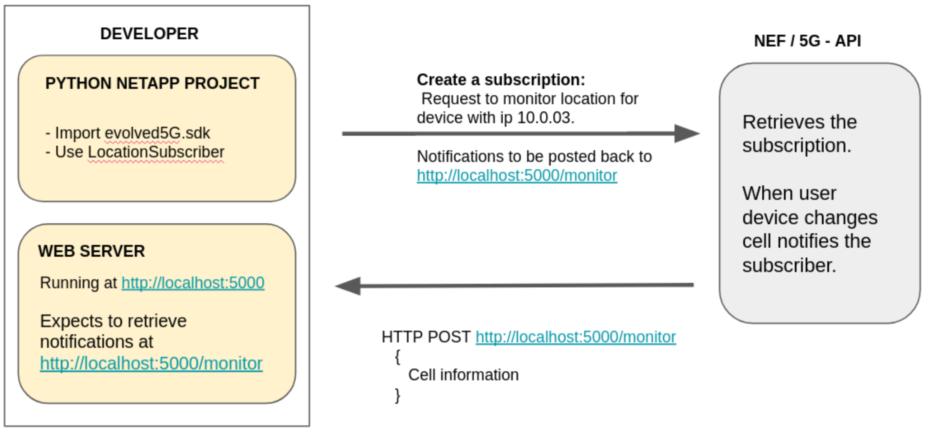

DEVELOPER
PYTHON NETAPP PROJECT
- Import evolved5G.sdk
- Use LocationSubscriber
WEB SERVER
Running at http://localhost:5000
Expects to retrieve notifications at http://localhost:5000/monitor
Create a subscription:
Request to monitor location for device with ip 10.0.03.
Notifications to be posted back to http://localhost:5000/monitor
NEF / 5G - API
Retrieves the subscription.
When user device changes cell notifies the subscriber.
HTTP POST http://localhost:5000/monitor
{
Cell information
}

Figure 11 Location Subscriber – High level usage

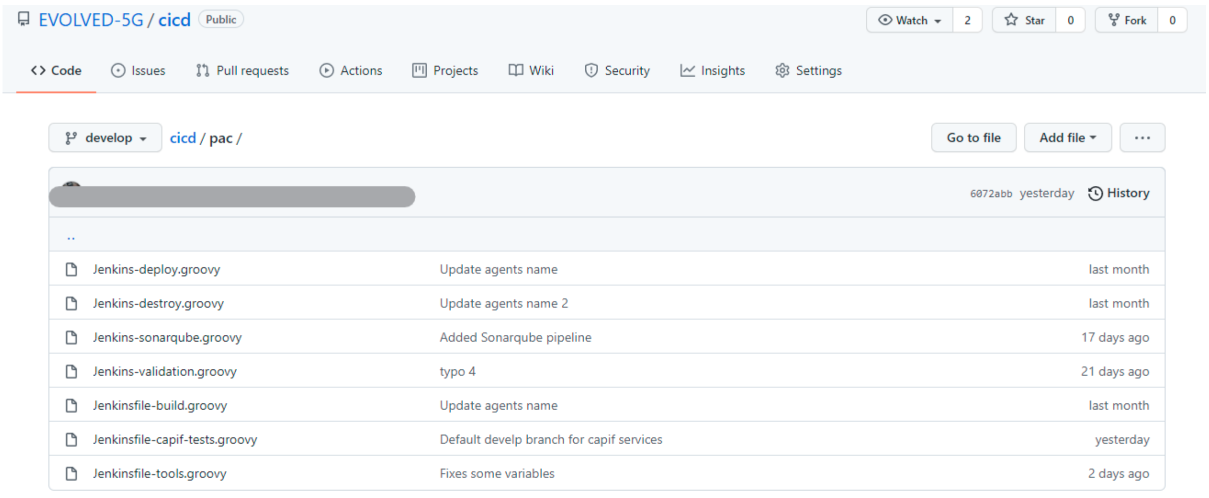
QoSAwareness, it is implemented to request Quality of Service (QoS) from a set of standardized values. This can result in better service experience for the NetApp consumers for scenarios like Live Streaming or Conversational Voice among others. A developer can use QoSAwareness to:

I. Establish a Guaranteed Bit Rate (GBR) QoS Flow in the NetApp. In this scenario the developer has to specify a) the QoS Flow Indicator (QFI) of interest: Conversational Voice or Conversational Video or Discrete automation b) the minimum allowed delay of data packages, during uplink or downlink or roundtrip (e.g., minimum 20ms for uplink) c) any data thresholds on volume (e.g., up to 5 GB for uplink). Once the QoS subscription is established, the NetApp developer receives notifications when a) thresholds (minimum delay) cannot be achieved b) thresholds (minimum delay) are restored or c) usage thresholds (data volume) are exceeded.
II. Establish a Non-Guaranteed Bit Rates (GBR) QoS in the NetApp. In this scenario the developer has to specify a) the QoS quality indicator of interest: Live Streaming or TCP Base db) the data thresholds on volume (ex. up to 5 GB for uplink). Once the QoS subscription is established, the NetApp developer receives notifications when usage thresholds (data volume) are exceeded.

In both scenarios the NetApp can use the received notifications to adjust the application's behavior in order to improve service experience. In the examples folder of the SDK repository a python script [47] is provided to demonstrate its usage. To implement the QoSAwareness library in the Netapp, the developer must follow the subsequent steps:

I. Initialize a python project and download evolved5G SDK as a python dependency. Use QoSAwareness class to create QoS subscriptions.
II. Initialize a web server to retrieve notifications from the NEF Emulator (or the 5G-API).
III. Start NEF emulator to simulate an environment where user devices are moving between / connecting to different network Cells. Then create a GBR subscription for these user devices via the NetApp. To facilitate testing, the NEF emulator supports the following scenario: If two user devices are connected to the same cell the NEF emulator will send notifications, informing the NetApp that the QoS targets cannot be achieved. If only one user device is connected to a cell, the NEF emulator sends notifications that the QoS targets are achieved.

Once the development of a NetApp is finished and has been pushed to the GitHub repository, the developer can launch the pipelines to verify the NetApp. This is performed by means of pipelines. Such pipelines can be found in the CI/CD repository [48], as it can be seen from Figure 12.



[redacted] 6072abb yesterday

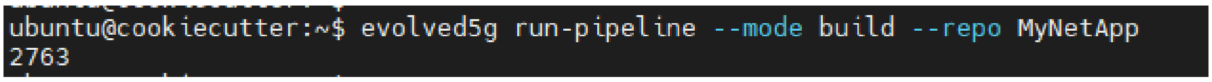
..		
Jenkins-deploy.groovy	Update agents name	last month
Jenkins-destroy.groovy	Update agents name 2	last month
Jenkins-sonarqube.groovy	Added Sonarqube pipeline	17 days ago
Jenkins-validation.groovy	typo 4	21 days ago
Jenkinsfile-build.groovy	Update agents name	last month
Jenkinsfile-capif-tests.groovy	Default develp branch for capif services	yesterday
Jenkinsfile-tools.groovy	Fixes some variables	2 days ago

Figure 12 Pipelines repository

These pipelines will be run by making use of Jenkins, which helps automate continuous integration and facilitate continuous delivery. These pipelines are Jenkins jobs and allow the lifecycle of the NetApp to be defined. These jobs are scripts based on the Groovy programming language, allowing to execute all kinds of tasks related to the verification of NetApps. How to run this pipeline will be shown below.

When all the steps related to the creation of the NetApp and the libraries to connect to the 5GC emulator or a real 5GC through the APIs have been done, the NetApp can go through three pipelines: build, deploy and destroy. In addition, the status of the pipeline can be checked to see if the result is satisfactory. To perform these steps, two commands are used, "*run-pipeline*" and "*check-pipeline*", which are described as follows.

The first one is used to launch the pipeline. Such command will return an ID so you can check the status, as it can be seen from Figure 13:



```
ubuntu@cookiecutter:~$ evolved5g run-pipeline --mode build --repo MyNetApp
2763
```

Figure 13 Run pipeline command

The second command is used to check the status of the NetApp pipeline, the developer can run the following command with the obtained pipeline ID during the execution:

```
Evolved5g check-pipeline –id 2763
```

When the pipeline is completed, the developer receives the following message:

```
Finished: SUCCESS
```

When all the pipelines have been executed and checked the NetApp is verified if the final message is satisfactory, binaries are generated, and these are stored in an Artifactory known as

Open Repository. Here besides finding the binaries of the verified NetApps, you can also see the images generated during the creation of the NetApps.

In case the NetApp is not capable of being built or deployed in OpenShift [49], in the command line is generated a report giving all the clues and information to the developer informing the errors found. Hence, the developer is able to debug through the NetApp code and fix the error, prior to re-run the pipeline again, the new NetApp code must be uploaded again to the corresponding GitHub repository.

In Section 4.3.2, it is described the verification process in more detail.

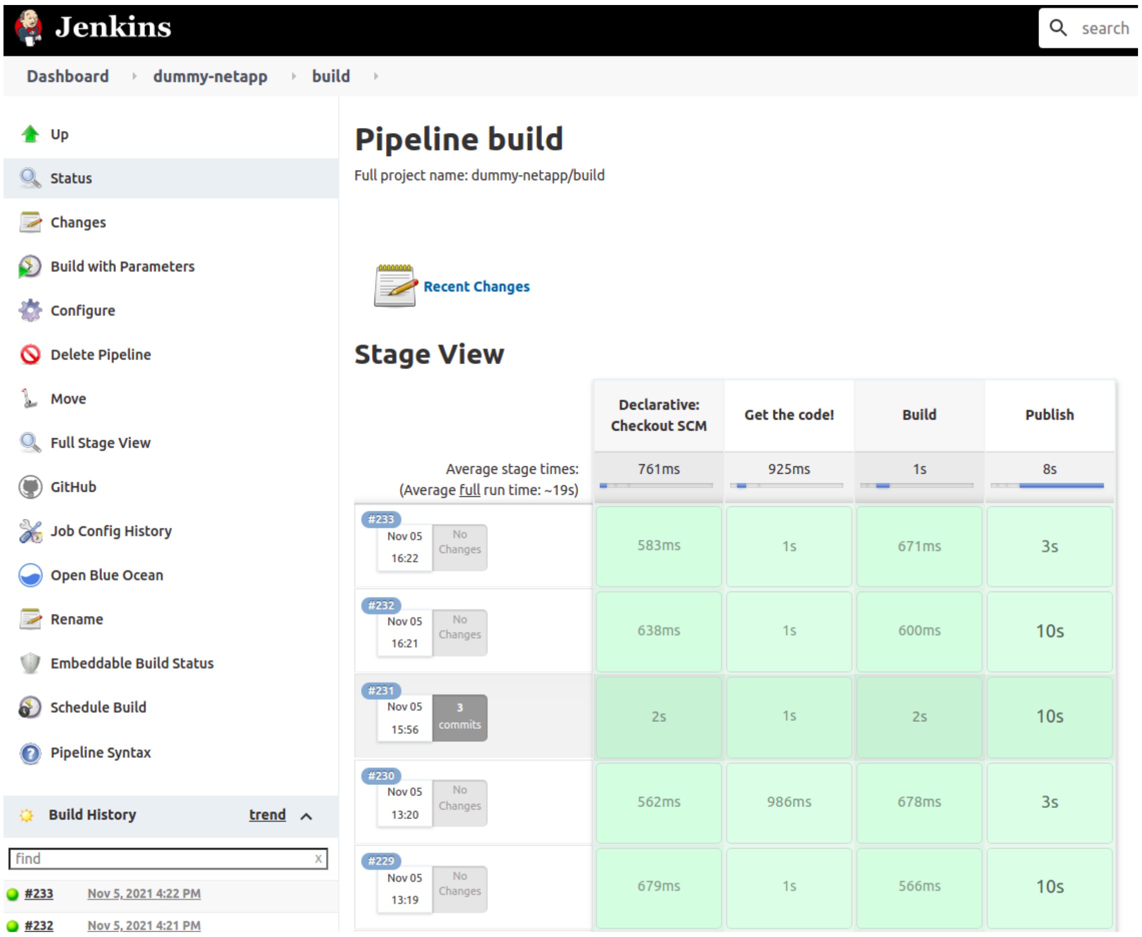
4.3.2 Verification Phase

Verification phase is compounded of several stages, where each stage is represented by a pipeline in Jenkins.

4.3.2.1 Pipelines

It has been developed several pipelines to automate software processes. These pipelines are described below.

- **Build.** Pipeline that builds the containers of the NetApp and generates necessary binaries to be deployed correctly in containerized infrastructure.



Pipeline build

Full project name: dummy-netapp/build

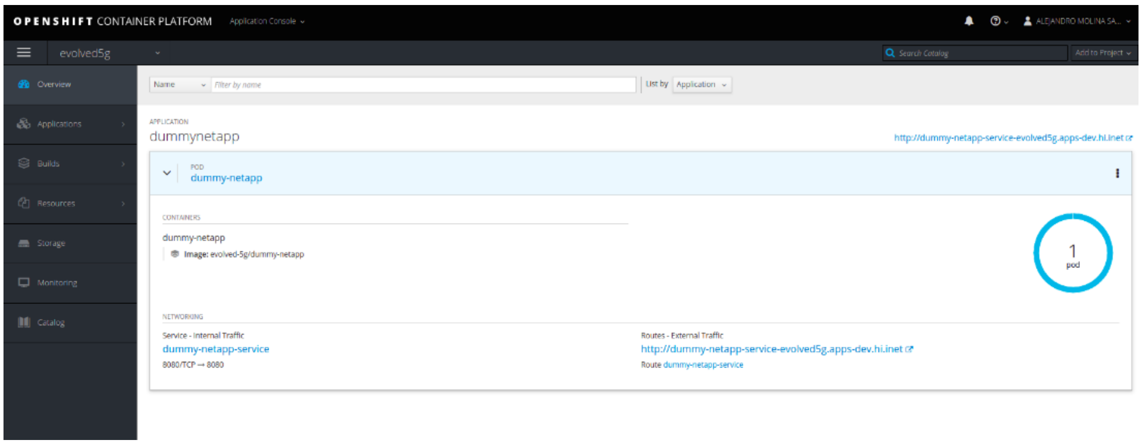
Stage View

	Declarative: Checkout SCM	Get the code!	Build	Publish
Average stage times: (Average full run time: ~19s)	761ms	925ms	1s	8s
#233 Nov 05 16:22 No Changes	583ms	1s	671ms	3s
#232 Nov 05 16:21 No Changes	638ms	1s	600ms	10s
#231 Nov 05 15:56 3 commits	2s	1s	2s	10s
#230 Nov 05 13:20 No Changes	562ms	986ms	678ms	3s
#229 Nov 05 13:19 No Changes	679ms	1s	566ms	10s

#233	Nov 5, 2021 4:22 PM
#232	Nov 5, 2021 4:21 PM

Figure 14 Build Pipeline execution example in Jenkins

- **Deploy.** Pipeline that deploys the NetApp in the verification infrastructure, which in this case is the OpenShift platform.



APPLICATION
dummynetapp
http://dummy-netapp-service-evolved5g.apps-dev.hi.inet

POD
dummy-netapp

CONTAINERS
dummy-netapp
Image: evolved-5g/dummy-netapp

1 pod

NETWORKING
Service - Internal Traffic
dummy-netapp-service
8080/TCP → 8080

Routes - External Traffic
http://dummy-netapp-service-evolved5g.apps-dev.hi.inet
Route dummy-netapp-service

Figure 15 Openshift Web Management Interface

- **Destroy.** Pipeline to clean and remove the NetApp from the execution infrastructure.

4.3.2.2 NetApp-NEF Emulator Interaction

The interaction between NetApps and the NEF emulator is established through RESTful APIs as presented in Figure 16. Initially, the NetApp registers to the available APIs that NEF supports and receives a reply with the created resource. When an event occurs in the simulated environment that the emulator implements, the emulator sends a callback notification to the NetApp asynchronously. In order to support the asynchronous notification, there is a need for the NetApp to implement its own RESTful API. As a result, a bidirectional communication is maintained, where both NetApps and NEF emulator play the role of client and server.

Consequently, after the successful establishment of the bidirectional communication, the NetApp processes the information retrieved from the emulator in the NetApp framework and traverses the results to the requested party (in this case, the Verification tests executors). The implementation aspects of the NEF emulator are analysed in Section 7.2.2.

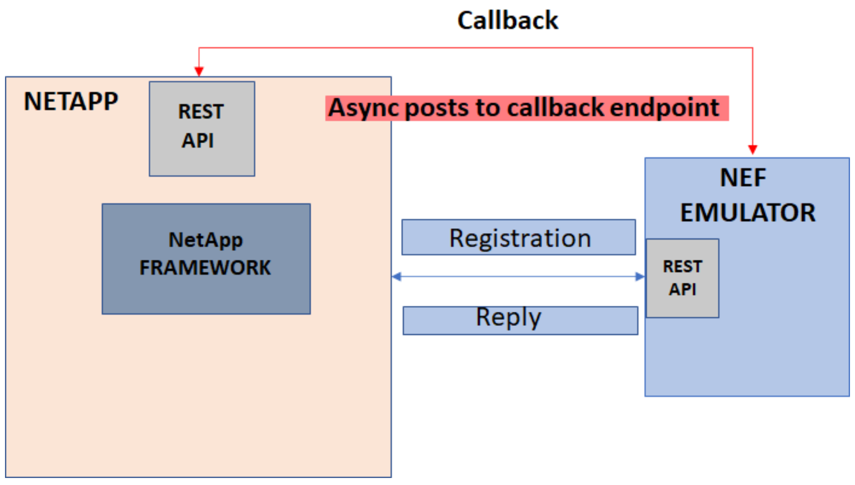


Figure 16 NetApp interaction with NEF Emulator via RESTful APIs

4.4 Security

This section focuses on security aspects related to the Workspace environment, development and verification phases, describing the measures adopted in order to avoid compromising the project data.

4.4.1 Development Phase

Regarding security within the development phase, the main focus is on the GitHub as it stores everything related to the workspace.

GitHub makes large investments related to platform security, incident response and anti-abuse [50]. In addition, GitHub has an R&D lab to create a foundation that developers and researchers can trust, and to contribute to making security easy for everyone who wants to secure open-source code. It has a bug testing and resolution system and partners with its customers' security and risk teams to encourage developer collaboration on GitHub. In the following, a more details about the security of this code control version platform in this project is presented.

A public organization called EVOLVED-5G has been created on GitHub where a set of repositories where all SDK tools as well as the CI/CD pipelines are stored. Anyone searching for this project will be able to access the organization and see the content of the public repositories. However, GitHub has a feature to keep the code that is stored in repositories and organizations safe, so in order to make any kind of modification in the public repositories, it is necessary for that user to be added to the organization. All organizations have a set of roles, to be able to make modifications in this organization, you must receive an admin invitation from this organization.

Each of the repositories found within the EVOLVED-5G organization can be configured with a different set of rules, giving the possibility of making changes to a reduced set of users. In addition, GitHub allows configuring Pull Requests, that is the process in which changes are integrated into the main branch, to be blocked until a specific set of administrators review and accept the changes.

To clone a repository locally or to create a NetApp repository, it is necessary to take security measures to make this possible. In order to both clone a repository and to put it in place, there are two options to do this securely, via HTTPs by providing or via username and password or via a user created SSH key on GitHub. Also, to create a remote repository with the SDK tools on GitHub, it is necessary for each user to have a token to be able to add it when prompted for input, otherwise GitHub will return an Authentication failure.

4.4.2 Verification Phase

Security has been one of the important things to take into consideration in EVOLVED-5G; as such, within the verification phase, it has been followed the recommended best practices for each one of the related technologies/tools.

4.4.2.1 Security in Jenkins

In Jenkins, two main security policies have been implemented:

I. **Access control**, which ensures users are authenticated when accessing Jenkins and their activities are authorized. In order to implement this Access control policy, we delegate all authentication to a configured Lightweight Directory Access Protocol (LDAP) server, including both users and groups.
II. **Protecting Jenkins against external threats.** Scan Jenkins server regularly in order to avoid vulnerabilities

4.4.2.2 Security in Open Repository

In the Open Repository, a Mandatory Access Control (MAC) through certifications have been implemented. The current configuration block access from clients that are not authenticated with an SSL/TLS certificate.

4.4.2.3 Security in OpenShift

There are security measures implemented in several layers of the OpenShift architecture.
These layers are the following:

I. **Container Hosts and Multi-tenancy.** Containers enable us to simplify multi-tenancy by deploying multiple applications on a single host or instance in OpenShift. This will allow us to deploy the different Netapps. In order to secure these container OpenShift has the following measures to ensure multi-tenancy.
    - **Linux namespaces.** Namespaces create an abstraction of a particular global system resource.
    - **SELinux.** This component enforces mandatory access control for every user providing additional security to containers,
    - **CGroups.** This component limits the usage of resources by each user.

II. **Registries.** It has been added a list of trusted sources to use public container registries. In this way, only containers coming from a trusted and verified repository can be used.

III. **Securing the container Platform.** OpenShift platform includes a built-in OAuth server that it will be used to authenticate ourselves to the API. As administrator of the platform, this OAuth server to use LDAP as Identity Server have been configured.

IV. **Network Security.** OpenShift platform uses software-defined networking to isolate different networks on the same host. This allows us to separate Netapps network from each other.

V. **Attached Storage.** Protect attached storage is fundamental in order to secure stateful services. OpenShift platform has installed plugin that allows us to create a persistent storage to each container.

5 VALIDATION ENVIRONMENT

5.1 MAIN OBJECTIVE

The goal of the Validation environment is to provide the tools and methodology required for the execution of the *validation process*. The validation process includes the evaluation of the functionality of the NetApps when they are used along with the vertical App (vApp). In the framework of EVOLVED-5G Industry 4.0 SME that participate to the project will provide the vApps. The target is to prove that the vApps reach the desired performance for a set of KPIs defined by the vertical/SME while making use of the NetApps. Additionally, the validation of a NetApp with a vApp guarantees that it can correctly operate in real network conditions (provided that the vApp makes use of all the capabilities exposed by the NetApp), as opposed to the emulated environment used during the *verification process*. As a result, the Industry 4.0 SME (vertical industry) receives a detailed report of the validation results, which can also be shared with the NetApp developer in order to help them improve the NetApp.

5.2 COMPONENT DESCRIPTION

The Validation environment is the functional block that is in charge of the validation of the NetApp. It is built on top of the EVOLVED-5G connectivity infrastructure (5G platforms) and communicates with the EVOLVED-5G Open Repository (See Section 62.4) for the retrieval of the NetApp artifacts to use during the Validation and the storage of the generated reports.

Within the lifecycle of a NetApp, the Validation environment is used after a successful completion of the verification process, i.e., only after a prove that the NetApp is functional and compatible with the 5G APIs. Similarly, the successful completion of the validation process is a pre-condition for a NetApp to enter the certification process. This is in order to ensure that the NetApp is able to operate in real network conditions before undertaking the Certification process.

5.3 DESIGN

The Validation environment is built on top of the Open5Genesis Framework [51], and is based on the methodology defined by the 5GENESIS project [52]. This methodology, which is detailed in Deliverable D2.4 [53] of the 5GENESIS project, describes the creation of a *Test Case* for each kind of experiment to perform, along with one or more *Scenarios* and *Slices*, which can be re-used for different Test Cases.

A Test Case is the description of an experiment, and includes information about the measurements to collect, the KPIs of interest and the methods for calculating them, any pre-conditions and the steps to perform during the experiment, among other details. On the other hand, the Scenarios and Slices describe the configuration and radio conditions to apply in the testbed during the execution of the experiment, and the set of end-to-end resources and services dedicated to the experiment, respectively.

In the context of EVOLVED-5G, the creation of new test cases devoted to the Validation of the NetApps would tackle details such as the requirements of the Vertical Application or the KPIs that are of interest to the Vertical. Given the variability on these matters, the Validation process is envisioned as a collaboration between the Verticals and the Platform Owners, who will agree on the set of KPIs to measure and the implementation of the tests, according to the specific needs of the Vertical and the capabilities exposed by the Platforms.

5.4 Security

Within the Open5Genesis Framework, security is handled by the Dispatcher [54]. This component is in charge of the security within the platforms and verifies that all the requests received are well-formed and authorized before redirecting them to the other components in the architecture. It features:

- TLS transport security based on Certificates
- User management (creation, activation, deletion, password recovery)
- Authentication
- Authorisation

The Dispatcher provides a front-end for many of the components in the Validation Framework, such as the ELCM and Analytics Module. Each request received by the Dispatcher is first validated, both in terms of user authorization and correct formatting, before redirecting it to the specific component in charge of the management of the request. The use of this approach yields two main benefits: all components are accessible from a single point of entry, while also allowing each component to delegate the authorization of the requests to the Dispatcher.

5.5 Implementation

The validation is a further step to assure the fully functionality of a NetApp. Such validation takes place in Malaga and Athens platforms. In order to start the validation process, a consultancy phase must be completed between the Vertical and the selected platform owner/administrator, where the scope and details of the validation are discussed. It is mandatory that the NetApp has successfully passed the verification phase and is stored in the Open Repository before a validation can be requested.

The validation process will select from the Open Repository a NetApp (artifact) already verified and will onboard it in the target platform to start the validation which comprises a series of test the NetApp must complete in order to get validated.

On the platforms, the orchestration of all the elements that take part in the execution of the Validation is handled by the Open5Genesis framework, implemented as part of the Coordination Layer of the 5GENESIS architecture [53].

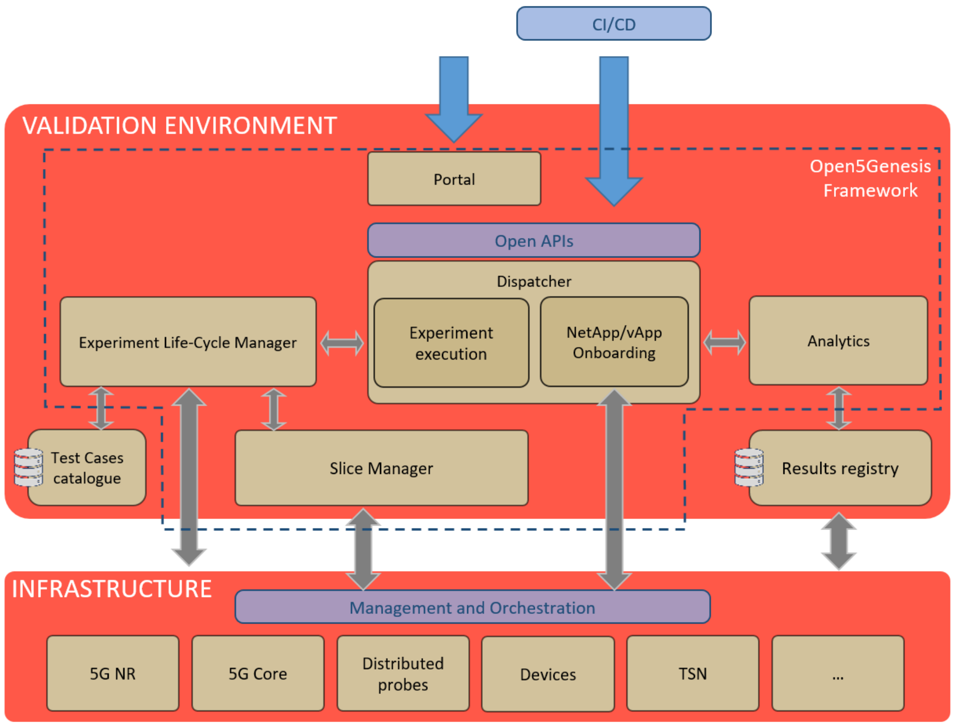


Figure 17 *EVOLVED-5G validation environment using the Open5Genesis Framework*

Figure *17* shows a high level overview of the components used during the validation phase.

The Open5Genesis Framework provides two points of entry for experimenters to the platforms: The Portal, which provides a graphical user interface for the definition and execution of experiments, and the Open APIs that expose such functionality via a REST API. In the context of EVOLVED-5G it is expected that the Open APIs will be the main access to the platforms, as interface with the other functional blocks in the architecture (the CI/CD framework described in Section 4.3.2 , is expected to be used as the starting point of the Validation or used to handle different steps during the process). For this reason, the Experiment Life-Cycle Manager (ELCM) has been modified in order to make the integration with the Portal an optional feature.

The access to all the functionality exposed by the Validation platforms is handled by the Dispatcher, which performs the authentication and validation of all the requests received.

The ELCM is able to coordinate the execution of concurrent experiments and orchestrates the usage of the resources in the platforms. This component provides a set of available Task types, which can be used for the implementation of the Test Cases. An initial implementation of new features and adaptations for the ELCM started during the fourth quarter of 2021 in the context of Task 3.2, with several results already available at time of writing.

The ELCM was developed using Python 3.7 as runtime environment. Given that this version has entered the Security Support phase it was decided to migrate to Python 3.10, effectively extending official support until October 2026. The first release of the ELCM based on Python 3.10 was released on October 11th, 2021 as version 3.0.0. This release also makes the usage of the Portal optional, which improves usability of this component within the context of EVOLVED-5G.

A second version (3.0.1) was released on November 5th 2021, which includes the following changes:

- The internal handling of each Task’s parameters has been modified in order to automatically perform a validation before the start of the Task’s execution, which provides additional insight in case of misconfiguration, and eases the creation of new kinds of Tasks.
- In order to better organize the configuration of the ELCM, an additional configuration file has been added (‘evolved5g.yml’), separated from the general configuration file (‘*config.yml*’). This file contains the settings of any component that is specific to EVOLVED-5G.
- A new task, that gives access to the internal REST client functionality has been added, allowing to perform simple REST requests during an experiment execution without the need of using external tools.
- Support for the existing CI/CD pipelines (Section 4.2.2) has been added to the ELCM, encapsulated as a new Jenkins API Client that is able to handle authentication, trigger jobs and retrieve the status of a pipeline. This client is used by two new Tasks that make use of these functionality. These Tasks are confirmed to work with the existing pipelines (which support building, deploying and deleting NetApp instances) and it is expected that the integration of new pipelines will be easy.
- In order to more cleanly integrate with the CI/CD framework, the handling of the experiment status REST endpoint has been slightly modified.

Finally, on November 11th 2021 version 3.1.0 was released, which includes the integration of the NEF emulator (Section 7.2.2) in the ELCM. In this version the general handling of the communication with the NEF emulator is implemented, as well as an additional task that can be used for starting and stopping the emulated movement of a device as part of a validation experiment.

Since the 5Genesis MANO Layer is focused on the management of VNFs, as opposed to containerized solutions, it is expected that this functionality will be handled, instead, by extensions performed to the Coordination Layer within the context of EVOLVED-5G. The 5Genesis MANO Layer will, in any case, remain available for use if needed, for example, for services required by any Vertical Application that are deployed as virtual machines.

Inherited from the Open5Genesis Framework there is also the Analytics Module, which provides access to the generated results and additional functionality for statistical analysis of the data. This component will be extended or complemented with the creation of new reporting capabilities in order to give Verticals the necessary information about the performance of the Vertical App and the NetApp, and to help them identify possible issues. The initial round of NetApp Validations, which is planned for the February to March 2022 period, as well as inputs received from Work Package 5 will drive the development of this reporting module.

Moving forward, the development of new features, as well as support for the initial round of NetApp Validations is planned. This includes the integration of the NEF emulator (Section 7.2.2) in the ELCM, which is expected to start during the month of November, while other components will be integrated as they are made available.

6 OPEN REPOSITORY

6.1 MAIN OBJECTIVE

The role of the open repository is twofold:

- It serves as Code Repository of the NetApps that SMEs are developing in the scope of EVOLVED-5G project, though it is OPEN for any SME to upload NetApps.
- It stores Verified, Validated and Certified versions of the NetApps so that those versions can be used by EVOLVED-5G tools till they are released to Marketplace.

6.2 COMPONENT DESCRIPTION

The open repository is a key component of EVOLVED-5G architecture. It contains the source code of all NetApps developed in EVOLVED-5G, as well as the Verificated, Validated and Certified binary versions, so that the NetApps can be used in the following EVOLVED-5G environments:

- Workspace, during NetApp development
- Verification, during NetApps testing

When NetApps have completed the development phase, NetApps images are built using Packer, and these images are stored in the Verificated, Validated and Certified NetApp areas:

- Verified area for NetApp verification.
- Validation area for NetApp Validation.
- Certification area for NetApp Certification.
- Marketplace for NetApp Release to Marketplace.

NetApp source code can be cloned from GitHub to compile, test, debug or modify the NetApps. To download the NetApp repository, the well-known command *clone* is performed as follows:

```
git clone https://github.com/EVOLVED-5G/<netapp_name>
```

NetApp container images can be downloaded from Artifactory to be deployed in Containerized infrastructure managers like OpenShift, Kubernetes, Rancher, among others.

In EVOLVED-5G, OpenShift will be used as a container platform in the verification phase. An example of the command line use of OpenShift to deploy a NetApp is the following:

```
oc deploy --latest dc/<netapp_name>
```

For deploying containers, terraform tool will be used. This tool will build all the infrastructure described in the terraform files. In this specific case, terraform will create the services and containers required for the correct and proper operation of the Netapp.

6.3 DESIGN

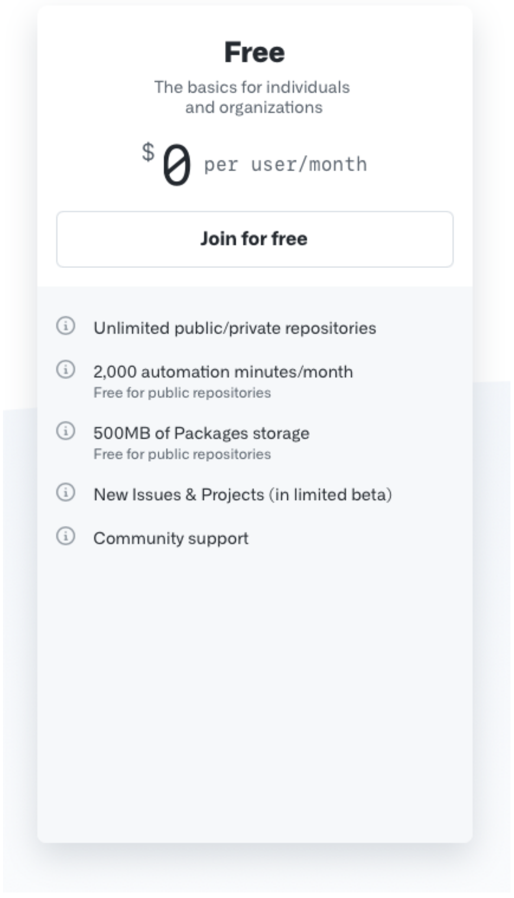
Design of the open repository has followed these design principles:

- Use industry standard tools instead of developing customs solutions
- Use open tools that requires no licenses for SMEs to use them
- Use Open-Source tools as preference

With this context, the following tools for implementing the open repository are selected.

6.3.1 Source Code open repository

Source Code open repository will be based in GitHub [31]. It has been selected the Free Plan as it fits in the activity volume expected during the project.



Free

The basics for individuals and organizations

$0 per user/month

Join for free

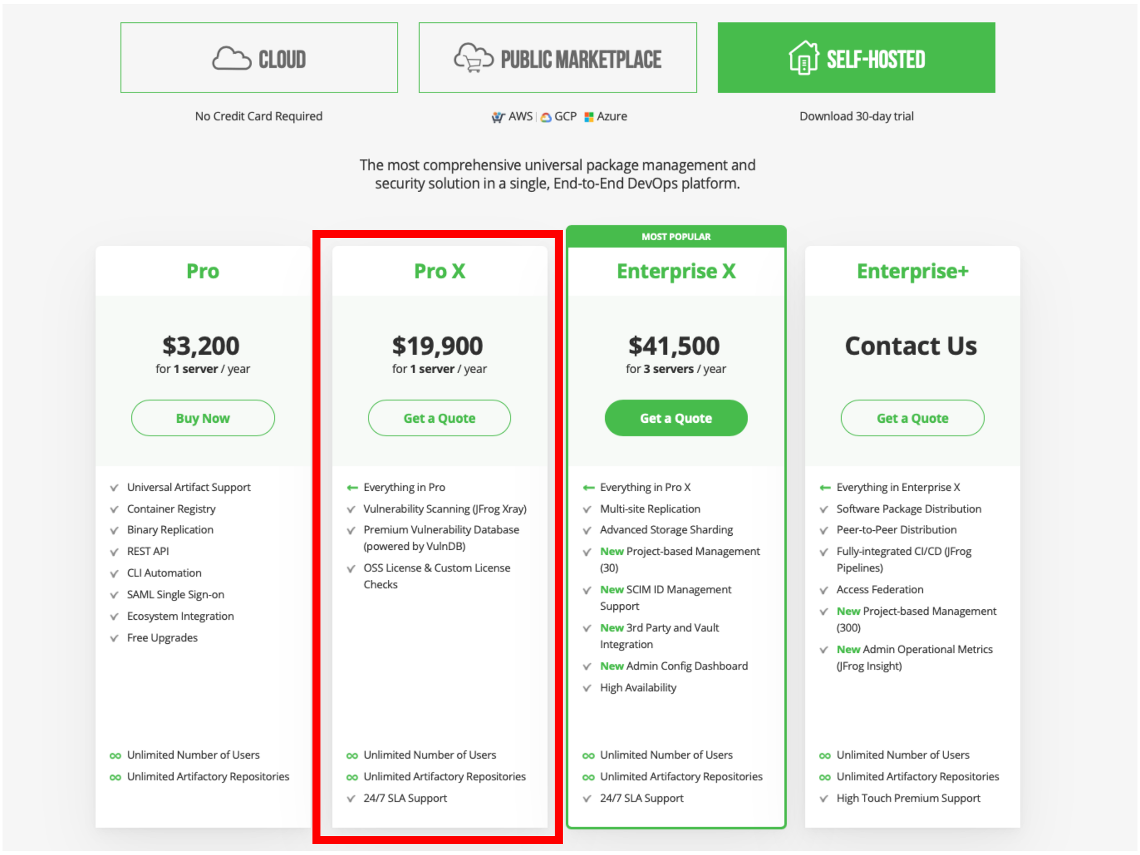
- Unlimited public/private repositories
- 2,000 automation minutes/month
  Free for public repositories
- 500MB of Packages storage
  Free for public repositories
- New Issues & Projects (in limited beta)
- Community support

Figure 18 GitHub Free Plan description

6.3.2 Container images repository

For storing Container images generated during Validation and Certification processes, the Artifactory [38] is selected.

The license model selected by Telefónica I+ D is "PRO X", which contains the features displayed in Figure *19*.



CLOUD	PUBLIC MARKETPLACE	SELF-HOSTED
No Credit Card Required	AWS | GCP Azure	Download 30-day trial

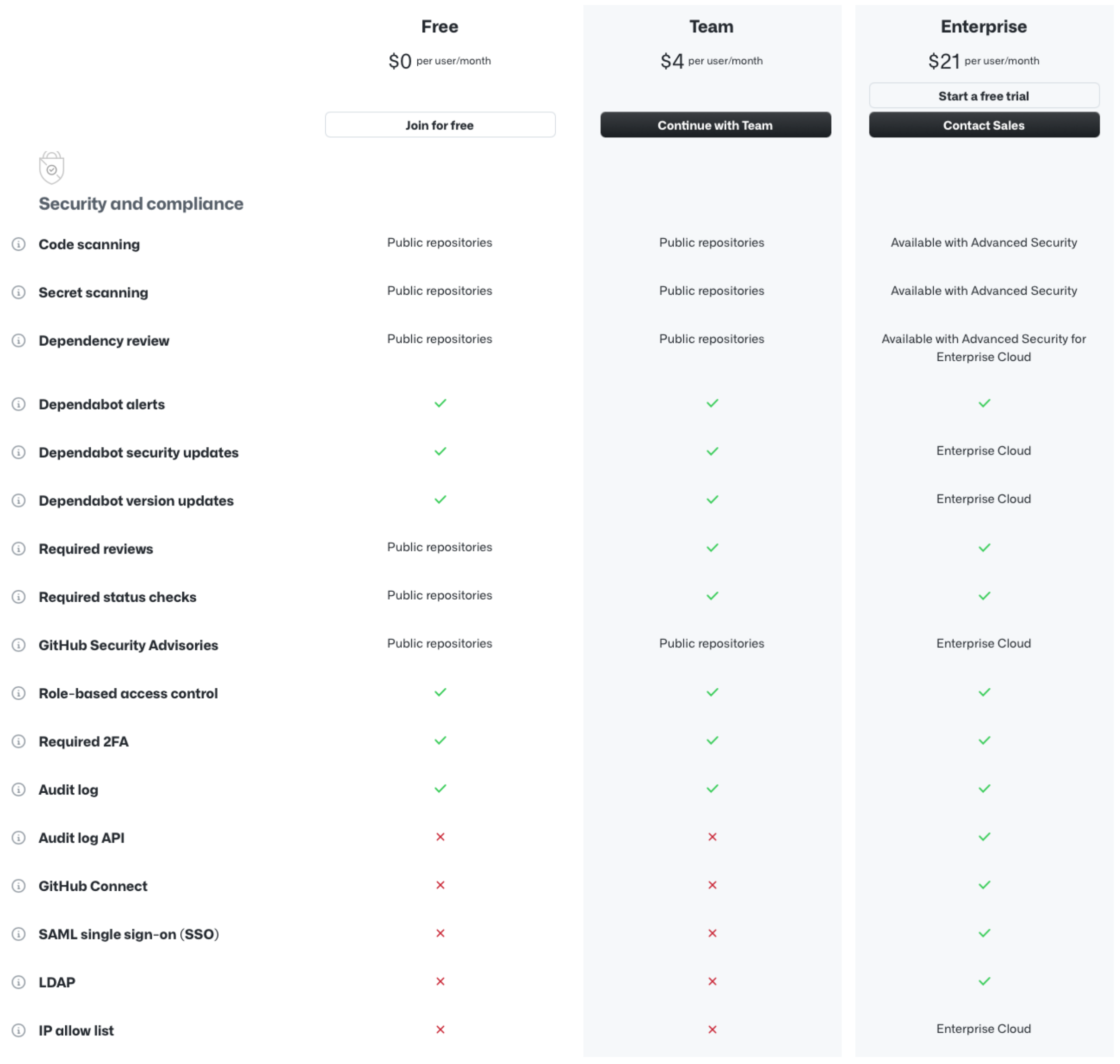
The most comprehensive universal package management and security solution in a single, End-to-End DevOps platform.

Pro	Pro X	Enterprise X (MOST POPULAR)	Enterprise+
$3,200 for **1 server** / year	$19,900 for **1 server** / year	$41,500 for **3 servers** / year	Contact Us
Buy Now	Get a Quote	Get a Quote	Get a Quote
Universal Artifact Support	Everything in Pro	Everything in Pro X	Everything in Enterprise X
Container Registry	Vulnerability Scanning (JFrog Xray)	Multi-site Replication	Software Package Distribution
Binary Replication	Premium Vulnerability Database (powered by VulnDB)	Advanced Storage Sharding	Peer-to-Peer Distribution
REST API	OSS License & Custom License Checks	New Project-based Management (30)	Fully-integrated CI/CD (JFrog Pipelines)
CLI Automation		New SCIM ID Management Support	Access Federation
SAML Single Sign-on		New 3rd Party and Vault Integration	New Project-based Management (300)
Ecosystem Integration		New Admin Config Dashboard	New Admin Operational Metrics (JFrog Insight)
Free Upgrades		High Availability	
Unlimited Number of Users	Unlimited Number of Users	Unlimited Number of Users	Unlimited Number of Users
Unlimited Artifactory Repositories	Unlimited Artifactory Repositories	Unlimited Artifactory Repositories	Unlimited Artifactory Repositories
	24/7 SLA Support	24/7 SLA Support	High Touch Premium Support

Figure 19 Artifactory selected plan “Pro X” by Telefónica

6.4 Security

GitHub security offer is based in package subscriptions. Figure *20* shows the security features per subscription type:



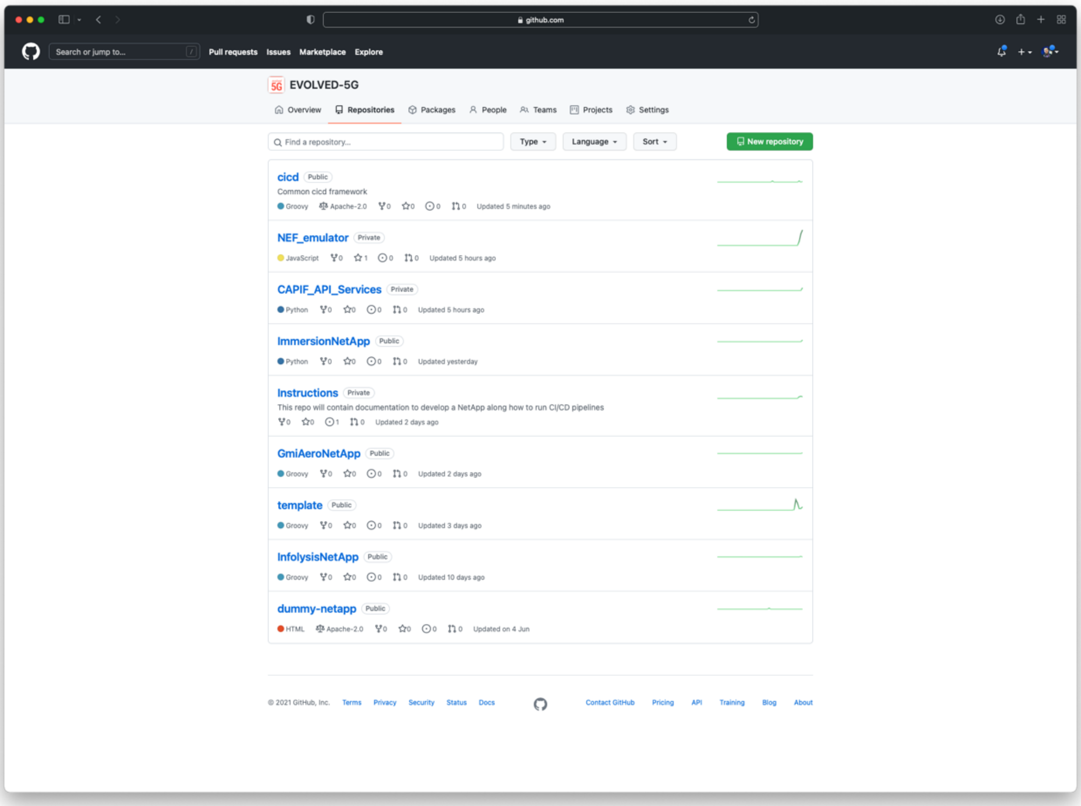
	Free	Team	Enterprise
	$0 per user/month	$4 per user/month	$21 per user/month
			Start a free trial
	Join for free	Continue with Team	Contact Sales
Security and compliance			
Code scanning	Public repositories	Public repositories	Available with Advanced Security
Secret scanning	Public repositories	Public repositories	Available with Advanced Security
Dependency review	Public repositories	Public repositories	Available with Advanced Security for Enterprise Cloud
Dependabot alerts	✓	✓	✓
Dependabot security updates	✓	✓	Enterprise Cloud
Dependabot version updates	✓	✓	Enterprise Cloud
Required reviews	Public repositories	✓	✓
Required status checks	Public repositories	✓	✓
GitHub Security Advisories	Public repositories	Public repositories	Enterprise Cloud
Role-based access control	✓	✓	✓
Required 2FA	✓	✓	✓
Audit log	✓	✓	✓
Audit log API	✗	✗	✓
GitHub Connect	✗	✗	✓
SAML single sign-on (SSO)	✗	✗	✓
LDAP	✗	✗	✓
IP allow list	✗	✗	Enterprise Cloud

Figure 20 *Github security packages included in Free plan*

Artifactory PRO X license has security built in features like Vulnerability Scanning and Premium Vulnerability Database (powered by VulnDB [41]). It is deployed in Telefónica internal Network, so it is not publicly exposed

6.5 Implementation

For storing source code, it is chosen the most common adopted source code manager tool in the development industry: GitHub. A repository has been created in GitHub for EVOLVED-5G project [40].



Repository	Visibility	Description	Language	License	Forks	Stars	Issues	Pull requests	Updated
cicd	Public	Common cicd framework	Groovy	Apache-2.0	0	0	0	0	Updated 5 minutes ago
NEF_emulator	Private		JavaScript		0	1	0	0	Updated 5 hours ago
CAPIF_API_Services	Private		Python		0	0	0	0	Updated 5 hours ago
ImmersionNetApp	Public		Python		0	0	0	0	Updated yesterday
Instructions	Private	This repo will contain documentation to develop a NetApp along how to run CI/CD pipelines			0	0	1	0	Updated 2 days ago
GmiAeroNetApp	Public		Groovy		0	0	0	0	Updated 2 days ago
template	Public		Groovy		0	0	0	0	Updated 3 days ago
InfolysisNetApp	Public		Groovy		0	0	0	0	Updated 10 days ago
dummy-netapp	Public		HTML	Apache-2.0	0	0	0	0	Updated on 4 Jun

Figure 21 *Evolved5G GitHub repository*

Artifactory tool has been chosen to keep Container images. As well, this tool is integrated with other CI/CD tools in Telefónica premises, which are supporting EVOLVED-5G Validation and Certification environments.

It has been created a repository in Artifactory for EVOLVED-5G project to upload container images. In this tool, two separate areas for Validated NetApp images and Certified NetApp images are created.

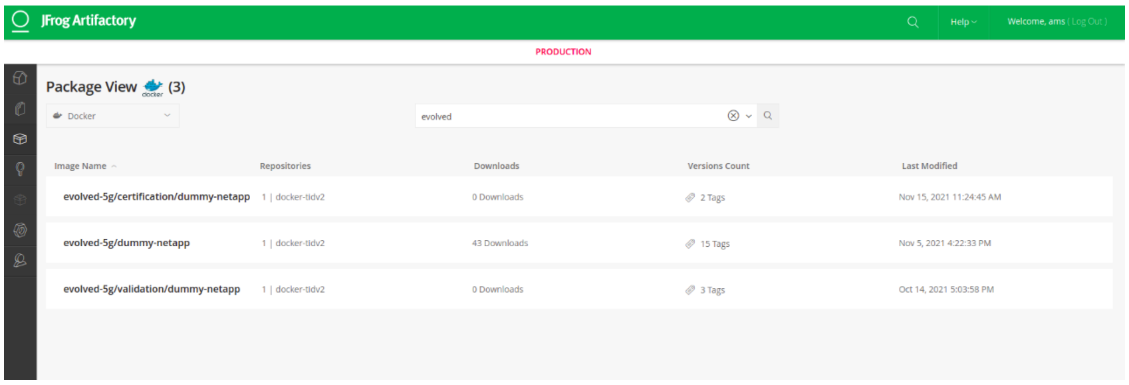


Image Name	Repositories	Downloads	Versions Count	Last Modified
evolved-5g/certification/dummy-netapp	1 | docker-tidv2	0 Downloads	2 Tags	Nov 15, 2021 11:24:45 AM
evolved-5g/dummy-netapp	1 | docker-tidv2	43 Downloads	15 Tags	Nov 5, 2021 4:22:33 PM
evolved-5g/validation/dummy-netapp	1 | docker-tidv2	0 Downloads	3 Tags	Oct 14, 2021 5:03:58 PM

Figure 22 *EVOLVED-5G* Validation and Certification *areas in Artifactory*

7 EVOLVED-5G INFRASTRUCTURE

7.1 MAIN OBJECTIVE

The focus of this section is to describe in detail main upgrades from the platforms used in EVOLVED-5G that compose the 5G-NPN Infrastructure, as well as NEF emulator and CAPIF Core Function tool.

7.2 COMPONENT DESCRIPTION

7.2.1 Athens platform

A key principle of the Athens platform is to provide multiple set-ups in a modular way to support and enhance a variety of test cases and experiments related to EVOLVED-5G. An initial description of the infrastructure regarding the Athens platform has been provided in D2.2 [55]. The already existing implementation will undergo architectural updates and specific features will be added during the lifecycle of the project, to serve the set of experiments under the scope of the validation phase.

Figure 23 depicts the logical topology which reflects the next iteration of updates within the Athens platform, including some network's functions updates within the platform's core network and the formation of three distinct edge regions, in order to introduce an advanced level in terms of flexibility and scalability.

The platform's core is supported by multiple Cloud computing systems that can be utilized as NFV Infrastructure for running NFV services. The instantiated VNFs implement some core network functions, such as the multiple virtual 5G Core functions. Additionally, the cloud computing solutions located at the core of the platform host various essential services that support and enrich the platform's capabilities, such as storage services, monitoring systems, CI/CD pipelines, network automation mechanisms, etc.

Each edge region to be developed, will stand as an autonomous ecosystem, connected to the core of the platform via either legacy routed layer three connections or SDN connections that enhance the overall network intelligence. A small factor NFVI system will be installed in each region to extend the virtualization capabilities of the platform and host VNFs that need to be instantiated closer to the user's premises. Finally, the edge regions will utilize RAN equipment that enables the 5G New Radio (NR), supporting the wide variation of 5G services, deployments, and spectrum.

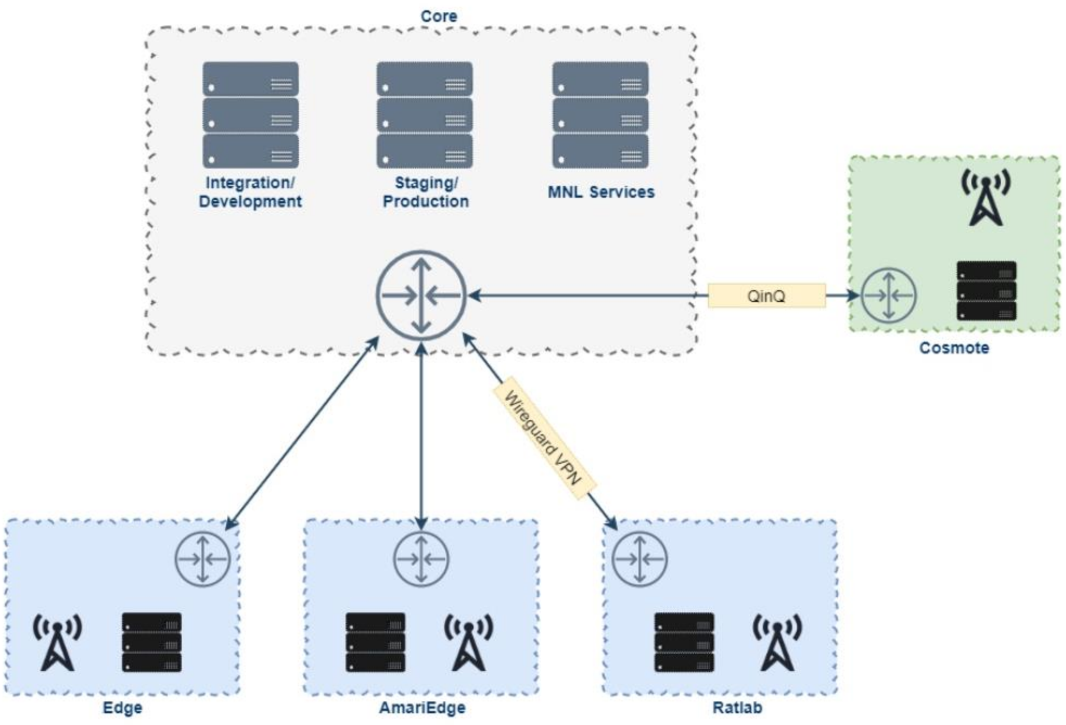


Figure 23 Logical Topology of the evolved Athens platform

The overall configuration and evolution of Athens platform, is being properly validated with both COTS UEs and open-source CPEs/UEs, that are described as follows:

7.2.1.1 Amarisoft Callbox mini

Packaged in a plug and play integrated PC, AMARI Callbox Mini is an ideal solution for 5G testing of all types of UEs with advanced configuration. It acts as a 3rd Generation Partnership Project (3GPP) compliant gNodeB and 5GC allowing functional and performance testing of NR (SA mode), LTE, LTE-A, LTE-M and NB-IoT devices. AMARI Callbox Mini is a very compact solution, designed to be used everywhere. From lab to exhibition show, this is the ideal product for NB-IoT, CATM1, CAT0 to CAT4 UEs testing.

Figure 24 Amarisoft Callbox Mini

7.2.1.2 5G Raspberry communication HAT

Moreover, as part of the evolution of the equipment there is a 5G Raspberry Pi communication HAT available, which adopts SIMCom 5G module SIM8200EA-M2, supporting 5G NSA and SA networking and data rates up to 2.4 Gbps (DL) / 500 Mbps (UL). The base board features standard M2 connector, allows connecting with different 4G or 5G communication module with M2 package. There are also USB3.1 port, audio jack and decoder, SIM card slot, etc. Allowing the deployment of configurations script and programming code, the "open" SA UE allows the demonstration and showcasing of various apps and modules with 5G connection. 5G HAT supports a wide range of the frequency bands, as follows:

Sub-6G (SA): n1, n2, n3, n5, n7, n8, n12, n20, n28, n38, n40, n41, n48, n66, n71, n77, n78, n79

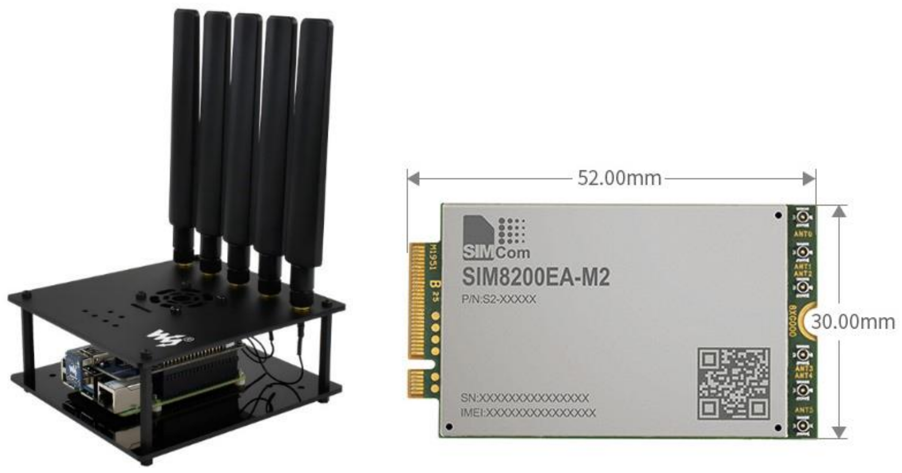


Figure 25 SIM8200EA-M2 5G HAT UE

7.2.1.3 5G CPE

The 5G CPE wireless router uses the Qualcomm Snapdragon X55 modem and converts 5G network into Gigabit Ethernet, USB 3.1, and WiFi 6, allowing the 5G connectivity with products that are not providing a 5G model as an option, but other network interfaces are available, such as ethernet, USB, wifi etc. The Qualcomm Snapdragon X55 5G Modem-RF System is a comprehensive modem-to-antenna solution designed to allow OEMs to build 5G multimode devices. The 5G CPE supports 5G SA and NSA and global multiple frequency band. Suitability for experimentation where high speed 5G network access is required, like: AI robot/drone, industrial IoT, outdoor live streaming, automotive devices.

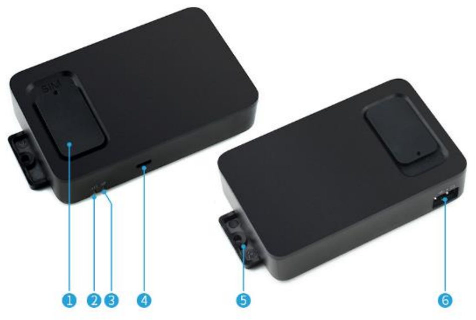

1
2
3
4
5
6
1 SIM card slot, supports 2FF standard SIM cards
2 Status indicator: showing operating/network statuses
3 Port indicator: showing WiFi/Ethernet statuses
4 USB-C Port
5 Mounting feet
6 Gigabit Ethernet/Power Supply

Figure 26 Qualcomm Snapdragon X55 CPE

7.2.2 Málaga platform

The evolution of the infrastructure in the Málaga platform can be organized in three different aspects: Extensions and upgrades to the radio interfaces, the acquisition of additional devices and the inclusion of Time Sensitive Networking (TSN) capabilities. The following sections provide details about each of these aspects.

7.2.2.1 Radio evolution

In UMA site the radio infrastructure has been upgraded with two new types of radio equipment.

One of them is an indoor small cell (Nokia Airscale ASiR - pico RRH) that includes two antennas in band n78 (3.5 GHz TDD band) operating in SA mode, with 100 MHz of bandwidth, up to 4x250mW Tx output power and 4 Tx/4 Rx. It covers the first floor of Module A of Ada Byron building that includes our laboratory and showcase area as you can see in Figure 27:

Figure 27 Indoor small cell antennas

Additionally, UMA has acquired some UEs for testing SA mode: One Plus 9, Huawei P40 and Telit FN980m.

The others are two outdoor millimeter antennas in n257 and n261 bands (26 GHz and 28 GHz TDD band), operating in NSA mode, with up to eight 100MHz 5G NR carriers and EIRP 60 dBm (Figure 28). They cover the parking area and the Faculty of Health Sciences in front, as well as the IHSM (Institute for Mediterranean and Subtropical Horticulture) building behind the Ada Byron building as can be seen in Figure 29:

Figure 28 Outdoor antennas

Figure 29 Area covered by the outdoor deployment

In order to test this equipment, the following devices will be used: Samsung S20 Ultra 5G (model SM-G988U1), Telit FN980m module.

7.2.2.2 User equipment and devices

The following UEs and devices have been acquired for use in the Málaga platform:

- **One Plus 9:** Supports 4×4 MIMO; 5G NSA and SA.
- **Huawei P40:** Supports SA & NSA network.
- **Samsung S20 Ultra 5G (model SM-G988U1)**: Supports mmWave bands 260(39 GHz), 261(28 GHz).
- **Telit FN980m 5G M.2:** Telit FN980m is a 5G M.2 module, that supports 5G sub-6 FDD and TDD + mmWave support and SA and NSA operations. 5G mmWave bands supported: n257 (28 GHz), n258 (26 GHz), n260 (39 GHz) and n261 (28 GHz).

Figure 30 From left to right - One Plus 9, Huawei P40 Samsung S20 Ultra 5G and Telit FN980m 5G M.2

- **MATRICE 600 PRO:** The Matrice 600 Pro aircraft is designed for cinematography and industrial applications; it belongs to the range of professional drones that can adapt to any project requiring a powerful flight tool with a range of up to 5 kilometers. The Matrice 600 Pro is distinguished by excellent payload performance, that is, it is designed to add up to a maximum of 6 kilograms of extra weight.

Figure 31 Matrice 600 Pro

- **HUSKY UNMANNED GROUND VEHICLE:** The Husky is a medium-sized robotic development platform. Its large payload capacity and power systems can accommodate an extensive variety of payloads, customized to meet research needs. Stereo cameras, LIDAR, GPS, IMUs, manipulators and more can be added to the UGV by an integration expert. The Husky is fully supported in ROS with community driven Open-Source code and examples.

Figure 32 Husky Unmanned Ground Vehicle

7.2.2.3 *Time-Sensitive Networking (TSN) over 5G*

TSN is a set of standards supporting deterministic communication based on Ethernet networks. The main aim is to add the TSN features to the 5G network, allowing an end-to-end deterministic communication.

The UMA infrastructure has evolved by adding the following components to the current setup:

- 1 Relyum TSN bridge [56]. A TSN switch that supports the largest TSN standards.

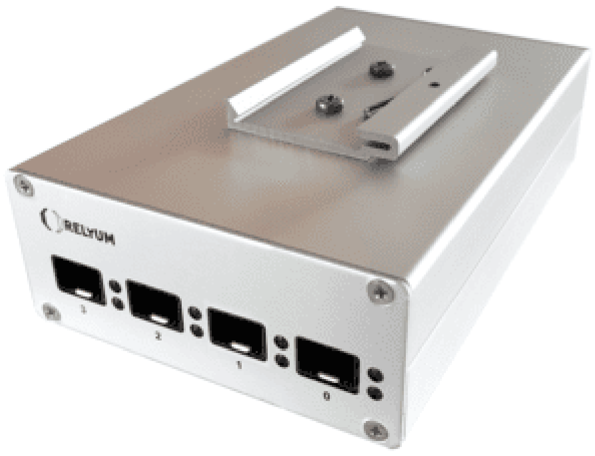


Figure 33 Relyum TSN Bridge

- 2 Relyum TSN PCIe cards [57] based on Intel I210 providing IEEE 802.1 TSN features. This device is able to act as a TSN bridge and a TSN endpoint offering the opportunity to include the TSN capabilities in the host device.

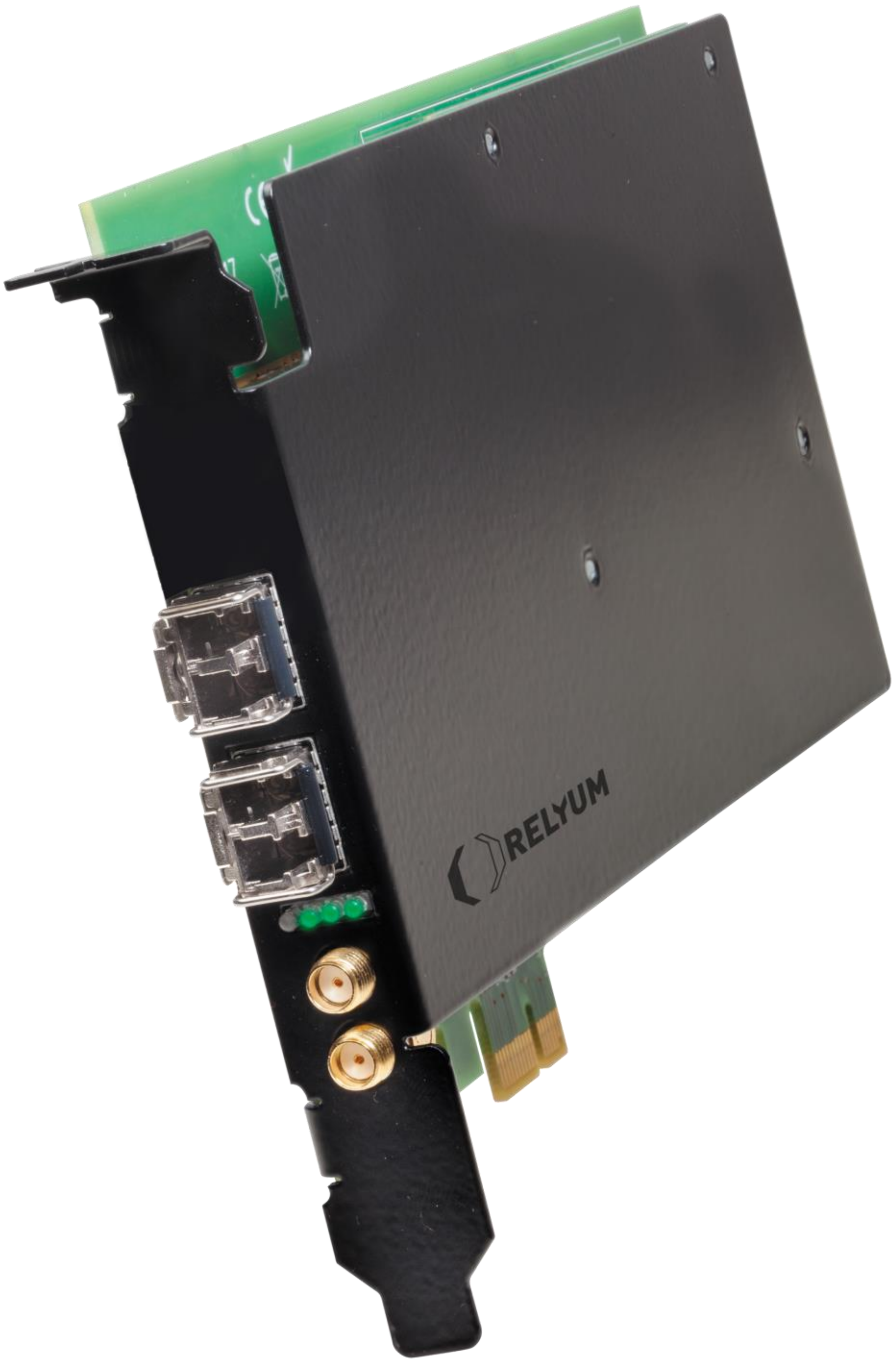


Figure 34 Relyum TSN PCIe card

The architecture is depicted in Figure 35. On the left and right side, the TSN endpoints, which will act as talker (send traffic) and listener (receive traffic), are represented. On the middle-top, the TSN bridge will allow to have a deterministic communication over Ethernet. On the opposite, middle bottom, the 5G bridge, which is the 5G network with the TSN key enablers elements (remarked in orange), will act as a TSN wireless bridge. The elements at the edge of the 5G network, Device-Side TSN translator (DS-TT) combined with 5G UE and Network TSN translator joint UPF, will be in charge of translation between both TSN and 5G domains. These TSN translators will take into account the traffic scheduling and prioritization and the time synchronization. The Application Function (AF) will be in charge of the 5G network configuration in order to maintain the TSN traffic requirements along the 5G network.

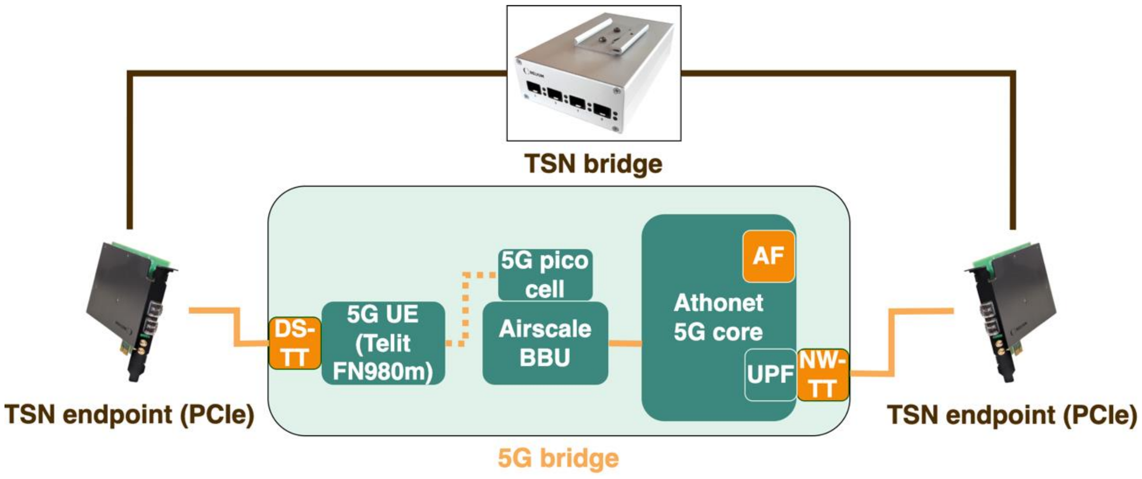


Figure 35 TSN architecture

7.2.3 CAPIF Core Function Tool

CAPIF Tool is a new tool developed in the context of EVOLVED-5G for the Verification, Validation and Certification process. CAPIF tool is an implementation of CAPIF APIs for the CAPIF Core Function entity as defined in 3GPP TS 29.222 [58]. This tool implements a REST API server that accepts API requests from CAPIF entities defined as API Invokers, API Exposing Function, API Publishing Function and API Management function.

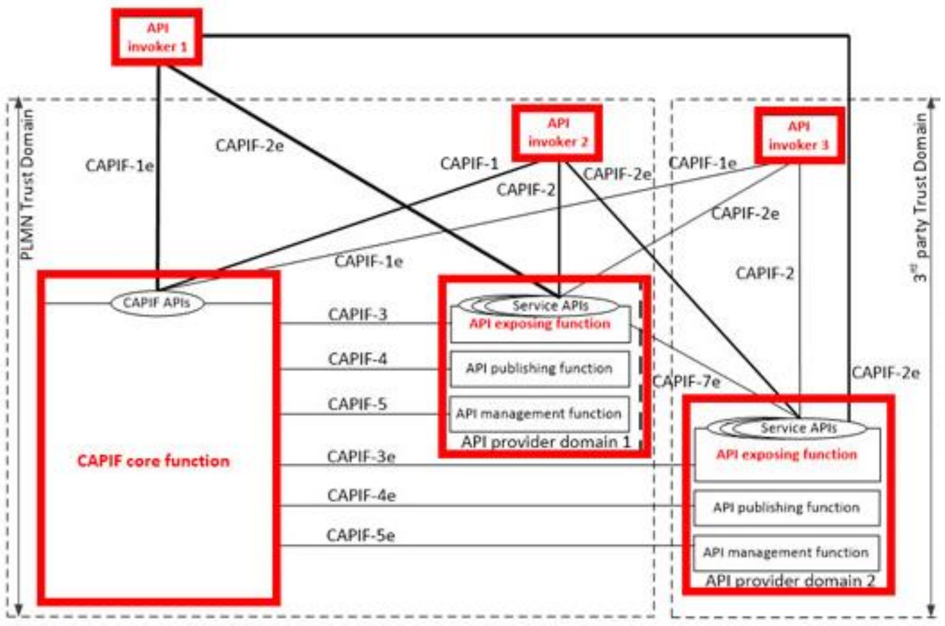


Figure 36 CAPIF architecture

As the CAPIF Core Function, the tool implements the following interfaces:

- CAPIF-1 to receive API requests from API Invokers.
- CAPIF-3 to receive API requests from API Publishers

- CAPIF-4 to receive API requests from API Publishers
- CAPIF-5 to receive API requests from API Management functions

CAPIF-1 will be the Interface consumed by NetApps that act as API Invokers. NetApps will be able to register in CAPIF and Discover published APIs. CAPIF-3, CAPIF-4 and CAPIF-5 interfaces are consumed by API publishing entities.

CAPIF Tool will also trigger Events API to notify CAPIF entities when required.

7.2.3.1 Implementation

CAPIF Tool uses Flask, a modern, high performance micro web framework for building APIs. It is fully compatible with the open standard for APIs, OpenAPI and also provides documentation based on Swagger UI. Its functional architecture is developed based on RESTful APIs.

For storing and preserving data, the emulator uses MongoDB which is a NoSQL database free and open-source relational database management system. Monitoring of the database is accomplished through mongo-express, an open-source administration and development platform for MongoDB.

The details of the CAPIF tool will be provided in D3.2, since it is mainly used in the certification process and it is being developed in the context of T3.4.

7.2.4 NEF Emulator

Is a software component that emulates the Northbound APIs of 3GPP’s Network Exposure Function (NEF). It mainly follows the RESTful paradigm and provides NetApps with some of the internal services and capabilities that the 5G System (5GS) offers. Considering that the Emulator is a standalone component (i.e., network function), and at this stage its internal communication with the 5GC is a challenging aspect, it provides tools for simulating events (e.g., mobility aware event where UEs are moving in predefined paths). To that end, NEF emulator can be described in three distinguishing parts depicted in *Figure 37*. The main features are described below:

- **Exposure Layer (NEF APIs)**: The principal idea of the emulator is the provision of the APIs that 5GC’s exposure function (i.e., NEF) defines. Therefore, the available APIs have been placed to the 5G Exposure layer. Currently, the available APIs include monitoring events (i.e., location), session establishment with QoS and network analytics. From NetApp’s perspective there are different APIs that will unlock the full potential of the vertical applications (vApps). As the project progresses, new APIs will be gradually added to the emulator and the existing ones will be enhanced.

- **Simulation Environment:** As mentioned above, currently the communication with the southbound interface (i.e., 5GC) is a demanding task. However, the emulator can surpass this challenge by creating simulated events. It provides an interactive geolocated environment where users can create different network scenarios. These scenarios are designed to simulate the basic aspects of a 5G network, required for testing the available service APIs. For example, in order to retrieve the location (i.e., cell level accuracy) of the UEs through the Monitoring Event API that 5GC exposes, the emulator provides a scenario where UEs are moving through 5G cells. NetApp developers will be able to alter data, allowing them to define specific scenarios according to their needs.

- **Common management Layer**: The emulator also provides common management functions such as token-based user authentication/authorization. Initially, to gain access to the emulator, there is the need for the user to create an account. After the creation of the account, an authorization step based on OAuth2.0 takes place, in order to make use of the available Northbound APIs or the Simulation Environment. Each NetApp developer will have a registered account/profile within the emulator that is considered as an isolated environment; thus, users will be able to configure the environment according to their needs.

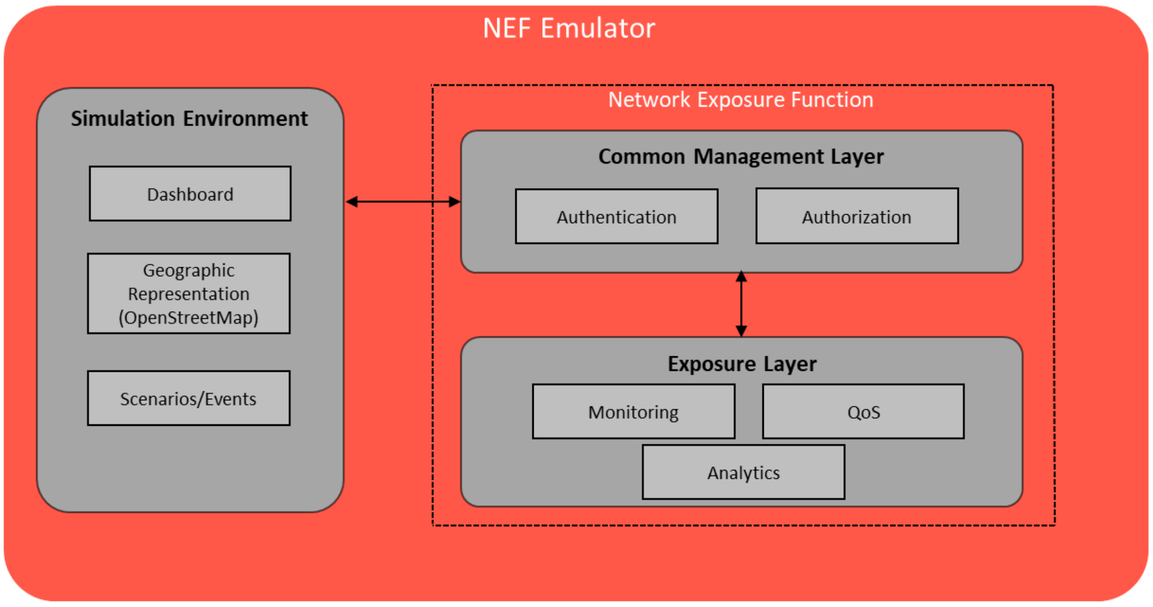


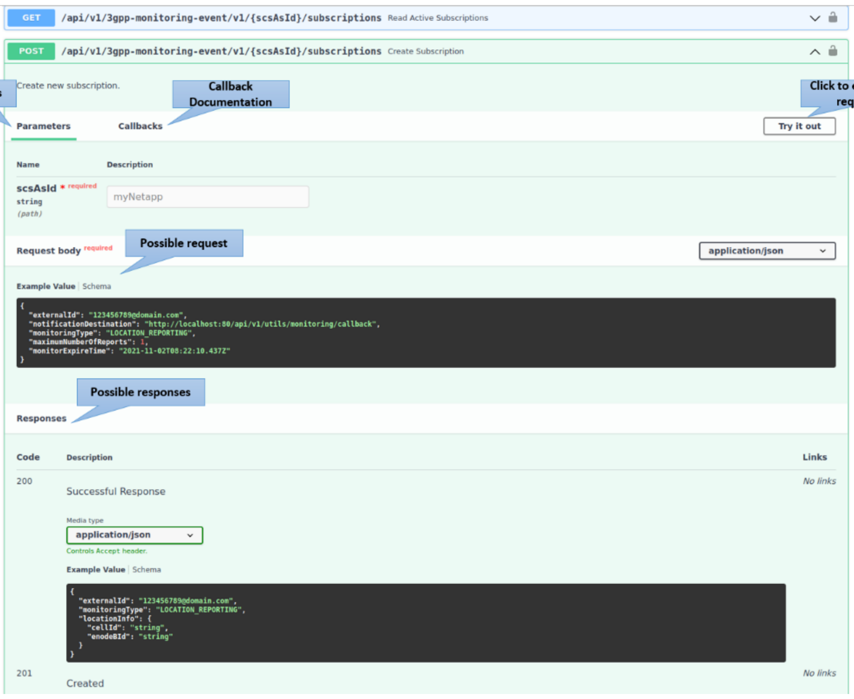
Figure 37 High-Level Architecture of NEF Emulator

Finally, it’s worth pointing out that user plane traffic is not supported. The traffic represents only the control plane flow between the NEF and a NetApp. NetApps will be tested under real traffic during the validation phase.

7.2.4.1 NEF Emulator Implementation

At its core functionality, NEF emulator uses FastAPI [59], a modern, high performance web framework for building APIs with Python 3.6+. It is fully compatible with the open standard for APIs, OpenAPI and also provides documentation based on Swagger UI [60] (i.e., interactive API documentation – Figure *38*) and Redoc [61]. Almost all of its functionality is developed based on RESTful APIs. Some preeminent examples are the NEF APIs, other APIs to create, configure and delete simulation events and user authentication / authorization process.

For storing and preserving data, the emulator uses PostgreSQL [62] which is a free and open-source relational database management system. The feasibility of administration and monitoring of the database is accomplished through pgAdmin [63], an open-source administration and development platform for PostgreSQL.



/api/v1/3gpp-monitoring-event/v1/{scsAsId}/subscriptions Read Active Subscriptions

/api/v1/3gpp-monitoring-event/v1/{scsAsId}/subscriptions Create Subscription

Create new subscription.

Name	Description
scsAsId * required string (path)	myNetapp

Request body required

Example Value | Schema

```
{
  "externalId": "123456789@domain.com",
  "notificationDestination": "http://localhost:80/api/v1/utils/monitoring/callback",
  "monitoringType": "LOCATION_REPORTING",
  "maximumNumberOfReports": 1,
  "monitorExpireTime": "2021-11-02T08:22:10.437Z"
}
```

Responses

Code	Description	Links
200	Successful Response	No links
201	Created	No links

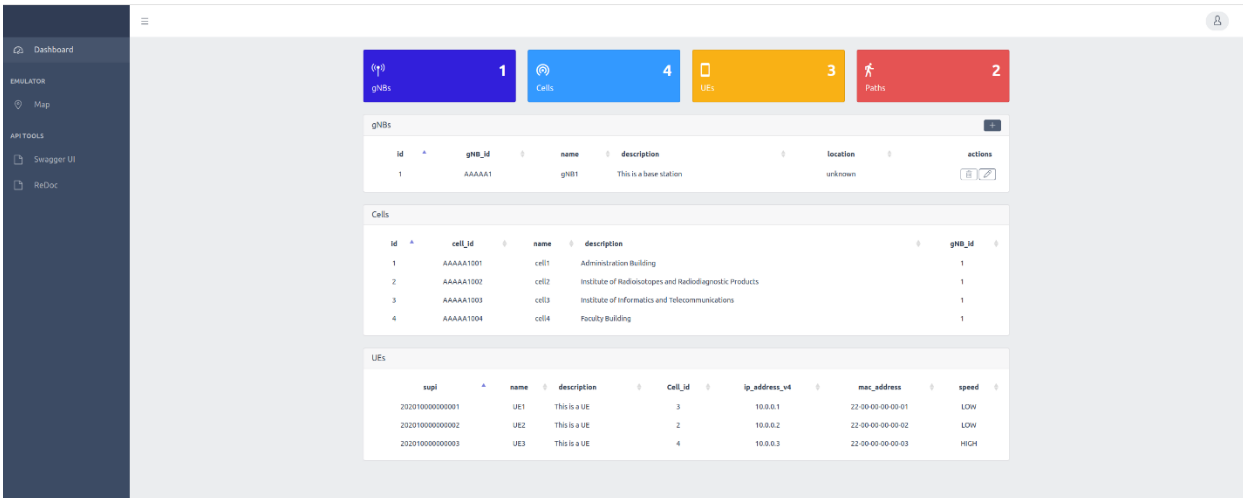
Example Value | Schema

```
{
  "externalId": "123456789@domain.com",
  "monitoringType": "LOCATION_REPORTING",
  "locationInfo": {
    "cellId": "string",
    "enodeBId": "string"
  }
}
```

Figure 38 Swagger UI

7.2.4.1.1 NEF Emulator Dashboard

When users sign in, they can monitor the services of the emulator through a front-end which is based on two open-source projects, CoreUI [64] and OpenStreetMap [65]. The former is an administrator template (Figure *39*) based on Bootstrap whereas the latter is used for distributing free geographic data.



gNBs

id	gNB_id	name	description	location	actions
1	AAAAA1	gNB1	This is a base station	unknown	

Cells

id	cell_id	name	description	gNB_id
1	AAAAA1001	cell1	Administration Building	1
2	AAAAA1002	cell2	Institute of Radioisotopes and Radiodiagnostic Products	1
3	AAAAA1003	cell3	Institute of Informatics and Telecommunications	1
4	AAAAA1004	cell4	Faculty Building	1

UEs

supi	name	description	Cell_id	ip_address_v4	mac_address	speed
202010000000001	UE1	This is a UE	3	10.0.0.1	22-00-00-00-00-01	LOW
202010000000002	UE2	This is a UE	2	10.0.0.2	22-00-00-00-00-02	LOW
202010000000003	UE3	This is a UE	4	10.0.0.3	22-00-00-00-00-03	HIGH

Figure 39 NEF emulator - Dashboard

The administrator template (i.e., Dashboard) contains information about the components of the simulation (i.e., UEs, Cells, predefined paths) that are created by a certain user. The front-end communicates with FastAPI and retrieves the essential information that is stored in the database. UEs are divided into three categories:

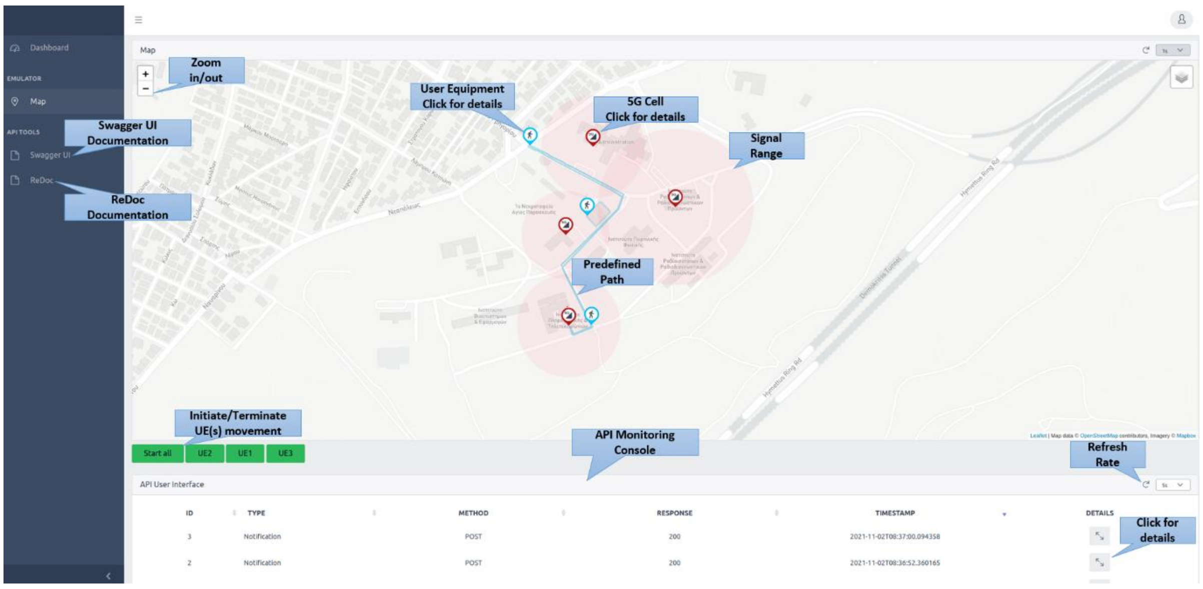
- Stationary UEs: They represent IoT connected devices (e.g., sensors, street cameras etc.). Also, they have a static location.

- Low velocity UEs: They represent pedestrians moving through the predefined paths and their velocity is relatively slow.
- High Velocity UEs: They represent vehicles moving through the predefined paths and they change location relatively high

For the simulation of the network a configurable set of 5G small cells are used. The details of the cell include cell's unique identifier, static geolocation and the coverage radius. Moreover, base stations (i.e., gNBs) are implemented to connect all the cells. It is important to note here that, the network does not reproduce a real operator deployment. To this end, the network scenario can be dynamically configured by any user. Specifically, users can create their own gNBs, cells, UEs and predefined paths through the Swagger UI.

7.2.4.1.2 NEF Emulator Map feature

After the successful creation of the scenario, the relevant information details are depicted in the Map page using OpenStreetMap (Figure *40*). In addition, Figure *41* presents some details regarding UEs and 5G small cells. At the bottom of Map page, there is an API User Interface that facilitates the monitoring of real time requests/notifications that the NEF APIs receive and generate. Users can experiment with NEF services either from the Swagger UI or directly through their application (i.e., NetApp). Figure *42* depicts the details of the requests/notifications provided in a form of Table. Some generic details include the name of the API used, the endpoint of this API, the type (request/notification) and a timestamp indicating the time that request occurred. Therefore, additional information regarding the request is provided, such as the method of the HTTP request, status code and last but not least, the request and response body (i.e., JSON format).



ID	TYPE	METHOD	RESPONSE	TIMESTAMP	DETAILS
3	Notification	POST	200	2021-11-02T08:37:00.094358	
2	Notification	POST	200	2021-11-02T08:36:52.360165	

Figure 40 NEF emulator - Map

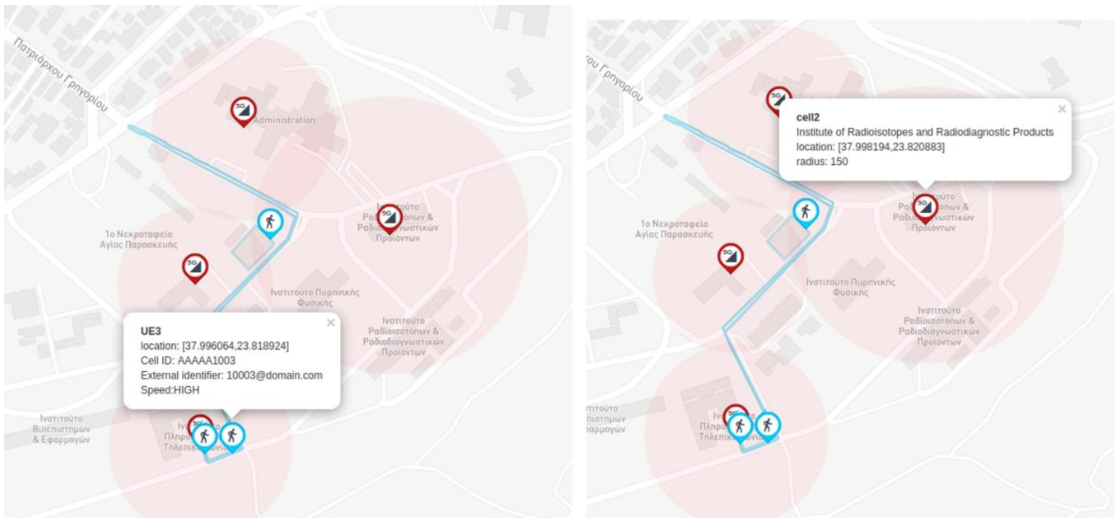
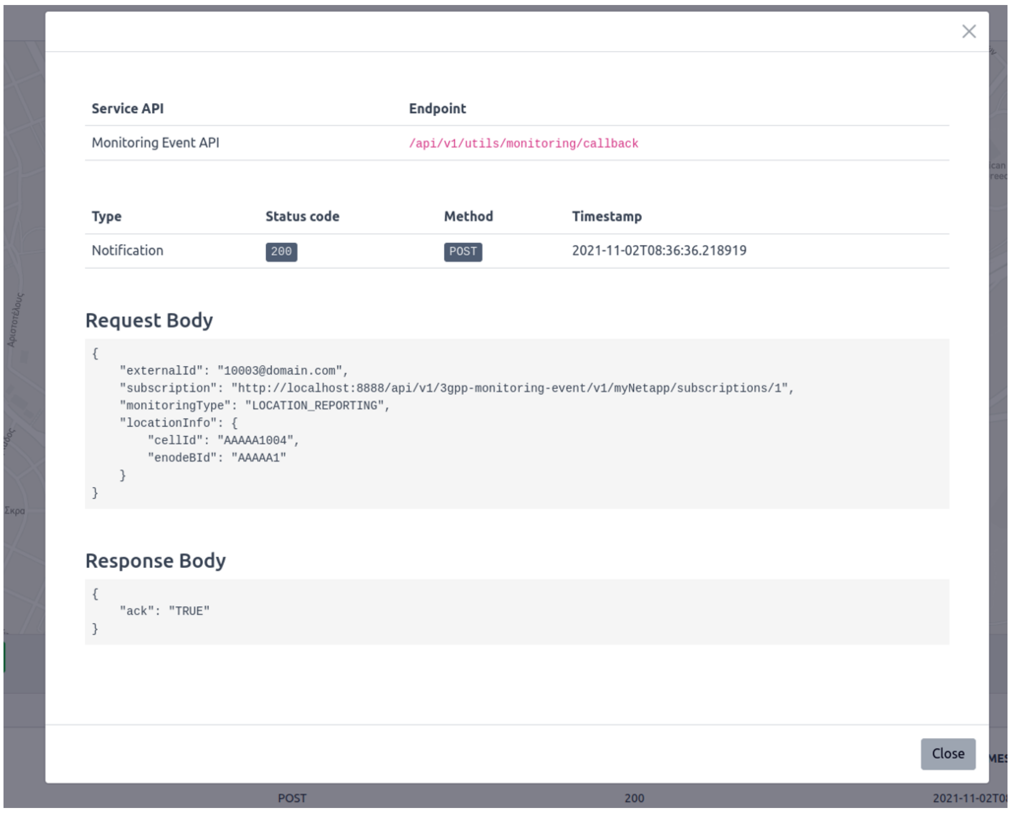

UE3
location: [37.996064,23.818924]
Cell ID: AAAAA1003
External identifier: 10003@domain.com
Speed:HIGH
Administration
cell2
Institute of Radioisotopes and Radiodiagnostic Products
location: [37.998194,23.820883]
radius: 150

Figure 41 Network scenario



Service API	Endpoint
Monitoring Event API	/api/v1/utils/monitoring/callback

Type	Status code	Method	Timestamp
Notification	200	POST	2021-11-02T08:36:36.218919

Request Body

```
{
    "externalId": "10003@domain.com",
    "subscription": "http://localhost:8888/api/v1/3gpp-monitoring-event/v1/myNetapp/subscriptions/1",
    "monitoringType": "LOCATION_REPORTING",
    "locationInfo": {
        "cellId": "AAAAA1004",
        "enodeBId": "AAAAA1"
    }
}
```

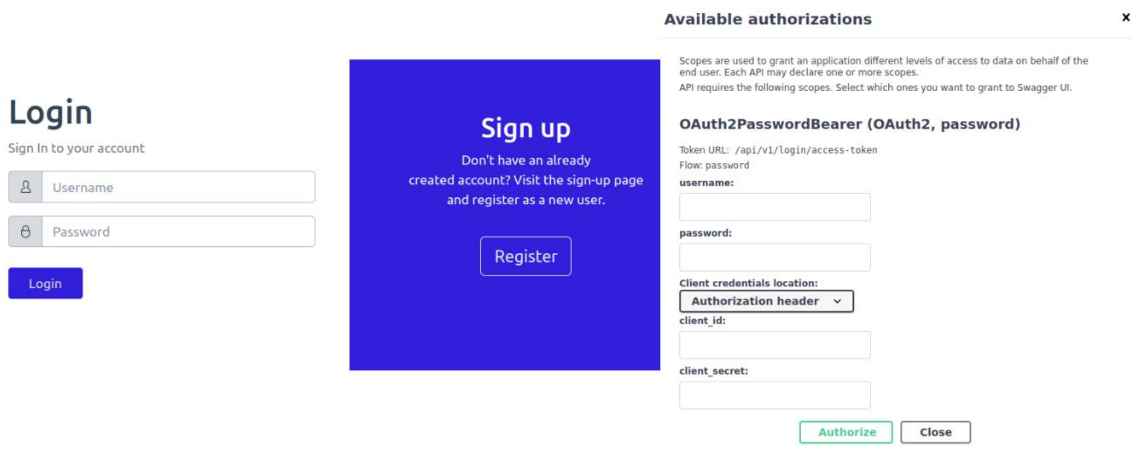
Response Body

```
{
    "ack": "TRUE"
}
```

Figure 42 Request/Notification details

7.2.4.1.3 NEF Emulator Security

The Emulator supports token-based authentication which is a one-time authentication (i.e., users enter their credentials once). Subsequently, whether the credentials are valid, users receive a unique access token. After the successful reception, they retain access by sending HTTP requests based on that token. Tokens are valid for a specific time, thus after expiration, new tokens must be generated. Therefore, authorization is supported through the OAuth2.0 open standard. Specifically, NEF emulator utilizes the OAuth2.0 with Password and JSON Web Tokens (JWT) [66]. While trying to access the APIs from the FastAPI framework, a user needs to send a request with the correct username and password, so the JWT can be generated. Authentication/authorization UIs are depicted in Figure 43.


Login
Sign In to your account
Username
Password
Login

Sign up
Don't have an already created account? Visit the sign-up page and register as a new user.
Register

Available authorizations
Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.
API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2PasswordBearer (OAuth2, password)
Token URL: `/api/v1/login/access-token`
Flow: password
username:
password:
Client credentials location: Authorization header
client_id:
client_secret:
Authorize Close

Figure 43 NEF emulator - Authentication/Authorization

7.2.4.1.4 Containers

The deployment of the NEF emulator is realized following the container-based approach. Specifically, containers are deployed using Docker [67], a ubiquitous open-source containerization platform that facilitates significantly the development efforts, as well as the technical expertise needed to deal with the programming tasks.

Having in mind that the NEF Emulator would eventually be deployed in different environments in order to serve different purposes (i.e., a workstation used for local development, CI/CD pipelines & testing, a potential live demo instance) the approach of shipping an application as containers seemed the best choice. By picking up Docker and containers in general we have managed:

- **Separation of concerns**: our application has discrete components which can now run independently in separate containers. This has the benefit of avoiding the "monolithic" approach and moving towards a more "microservices" one. As of version 1.0.0, the NEF Emulator consists of 3 containers, one for the NEF API service (backend), one for the PostgreSQL service (database) and one for visual database management (pgAdmin). As the project evolves, more services can be added, as independent containers.
- **Straightforward configuration**: Docker takes away repetitive, mundane configuration tasks and is used throughout the development lifecycle for fast, easy and portable application development - desktop and cloud. The developer can easily setup the environment by configuring a file with variables.
- **Shipping**: In order to facilitate the deployment of the NEF emulator, docker-compose is used to build and run our multi-container application. With just two commands, the developer can build (i.e., docker-compose build) and run (i.e., docker-compose up) the application as a composition of containers.
- **Developer velocity**: Docker provides fast build times since it's using a layered filesystem and can cache whole steps at the process of building a docker image. It also offers the possibility of quicky destroying and regenerating the application in case something breaks or is not working properly.

8 Conclusion and next steps

This deliverable presented the work performed in the context of WP3, and more specifically, as part of tasks T3.1, T3.2 and T3.3, during the first year of the project, concretely from M3 to M12.

The starting point for this WP was the input received from WP2, namely the EVOLVED-5G NetApp definition and the initial design of the EVOLVED-5G experimentation facility around which the project aims to create a NetApp ecosystem to foster innovation and collaboration in the Industry 4.0 vertical. This is the reason why, in order to provide some initial context to the reader, this deliverable includes some information about the current EVOLVED-5G architecture and the different environments and processes the NetApp will follow.

Based on those initial WP2 guidelines and definitions, T3.1 designed and started the development of the workspace environment, which offers a set of functionalities and tools to facilitate developers the creation and verification of NetApps. A detailed description of this environment is provided in this deliverable, including security aspects, as well as information about the current stage of its implementation.

T3.2 focuses on the creation of the validation environment, conceived as a set of validation tools for the non-functional testing of the previously developed and verified NetApps. This task is also responsible for the provision of a public open repository where the validated NetApps will be stored and made available for other developers. This report also introduces an extensive description of both the validation environment and the open repository, of their main components as well as information about security and current implementation details.

Task 3.3 supports the integrations between both the development and the validation environments, while guiding the required updates in the infrastructure. Feedback from WP4 and WP5 is continuously gathered to ensure that all new features required from developers are considered. All information related to the updates made in the Malaga and Athens platforms during the reporting period, from the integration of physical elements to the development of the new functional blocks, is presented in this deliverable.

It is important to highlight that before starting the design and development of the different environments, components and tools mentioned above, the project performed an extend analysis of what was available in the market. The State of the Art of a variety of tools related to the different environments being implemented in the EVOLVED-5G facility, such as the SDK, the verification and validation tools and the open repository is also provided in this report.

Regarding next steps, it must be taken into account that some other modules, that will also be part of the integrated EVOLVED-5G platform (i.e., the Certification environment and the Marketplace) are designed and developed as part of T3.4, and thus, they will be covered in deliverable D3.2, to be released in M18. In addition, the development work of all WP3 tasks is an ongoing process, thus this deliverable just provided information about the up-to-date work by the time this deliverable is being written. All the mentioned tools and environments are currently under development. For example, for the SDK tool, it is planned to implement new libraries coming from WP4, while in the verification side, new tests such as static code analysis are being defined. Work on the validation environment and the infrastructure also continues, with the integration of new components and the improvement of existing functionality. All this work will be documented in deliverable D3.3 to be submitted in M28.

REFERENCES

[1] 5G PPP – 5G innovations for verticals with third party services | Programme | H2020 | CORDIS | European Commission, from https://cordis.europa.eu/programme/id/H2020_ICT-41-2020
[2] Sonata project, from https://sonata-project.org
[3] 5G TANGO project, from https://www.5gtango.eu
[4] sonata-nfv/tng-sdk-project: The 5GTANGO SDK tool to manage network service projects, from https://github.com/sonata-nfv/tng-sdk-project
[5] Cookiecutter: A command-line utility that creates projects from cookiecutters (project templates), from https://github.com/cookiecutter/cookiecutter
[6] Jinja | The Pallets Projects, from https://palletsprojects.com/p/jinja/
[7] GitHub Sonata project, from https://github.com/sonata-project
[8] sonata-nfv/son-editor-backend: The backend of SONATA's web-based service and function descriptor editor, from https://github.com/sonata-nfv/son-editor-backend
[9] sonata-nfv/son-emu: Attention! Legacy! This repo will be replaced with https://github.com/containernet/vim-emu, from https://github.com/sonata-nfv/son-emu
[10] sonata-nfv/son-monitor: SONATA's monitoring repository, from https://github.com/sonata-nfv/son-monitor
[11] sonata-nfv/son-cli: SONATA SDK command line interface tools, from https://github.com/sonata-nfv/son-cli
[12] sonata-nfv, from https://github.com/sonata-nfv
[13] sonata-nfv/tng-schema: 5GTANGO descriptor, record, and package specifications and schemas (data models), from https://github.com/sonata-nfv/tng-schema
[14] sonata-nfv/tng-portal: The 5GTANGO (web) Portal, from https://github.com/sonata-nfv/tng-portal
[15] sonata-nfv/tng-vnv-dsm: 5GTANGO's V&V Platform recommendation engine repository, from https://github.com/sonata-nfv/tng-vnv-dsm
[16] sonata-nfv/tng-sdk-descriptorgen: 5GTANGO descriptor generator, from https://github.com/sonata-nfv/tng-sdk-descriptorgen
[17] SONATA releases - 5GTANGO, from https://5gtango.eu/software/49-sonata-releases.html
[18] sonata-nfv/tng-sdk-package: The 5GTANGO SDK tool to create and unpack 5GTANGO packages, from https://github.com/sonata-nfv/tng-sdk-package
[19] sonata-nfv/tng-sdk-validation: The 5GTANGO SDK validation repository, from https://github.com/sonata-nfv/tng-sdk-validation
[20] sonata-nfv/tng-sdk-sm: 5GTANGO SDK Specific Manager develop and test framework, from https://github.com/sonata-nfv/tng-sdk-sm
[21] Jenkins, from https://www.jenkins.io/
[22] Robot Framework, from https://robotframework.org/
[23] 3GPP 15-18 Release (CAPIF specification), from https://www.3gpp.org/specifications/67-releases
[24] sonata-nfv/tng-sdk-test: 5GTANGO SDK Tests repository, from https://github.com/sonata-nfv/tng-sdk-test
[25] sonata-nfv/tng-sp-ia-emu: 5GTANGO Service Platform Infrastructure Adapter emulator wrapper, from https://github.com/sonata-nfv/tng-sp-ia-emu
[26] Iterate faster, innovate together | GitLab, from https://about.gitlab.com/

[27] sonata-nfv/tng-sdk-benchmark: 5GTANGO SDK tool for fully automated VNF and network service benchmarking and profiling, from https://github.com/sonata-nfv/tng-sdk-benchmark

[28] sonata-nfv/tng-analytics-engine: 5GTANGO Analytics Engine repository, from https://github.com/sonata-nfv/tng-analytics-engine

[29] OpenTAP, An Open Source Project for Test Automation, from https://opentap.io/

[30] Git, from https://git-scm.com/

[31] GitHub, from https://github.com/

[32] Bitbucket | The Git solution for professional teams, from https://bitbucket.org/

[33] Gogs: A painless self-hosted Git service, from https://gogs.io/

[34] Docker Registry | Docker Documentation, from https://docs.docker.com/registry/

[35] Docker Hub Container Image Library | App Containerization, from https://hub.docker.com/

[36] 3GPP TS 23.222, "Common API Framework for 3GPP Northbound APIs", Release 17, V17.4.0, April 2021.

[37] EVOLVED-5G, Deliverable 2.1 "Overall Framework Design and Industry 4.0 Requirements"

[38] JFrog - Universal Artifact Management for DevOps Acceleration, from https://jfrog.com/

[39] Kubernetes, from https://kubernetes.io/

[40] EVOLVED-5G, from https://github.com/EVOLVED-5G/

[41] VulnDB, from https://vulndb.cyberriskanalytics.com/

[42] EVOLVED-5G/template, from https://github.com/EVOLVED-5G/template

[43] pip documentation v21.3.1, from https://pip.pypa.io/en/stable/

[44] EVOLVED-5G/SDK-CLI: SDK-CLI for EVOLVED-5G H2020 project, from https://github.com/EVOLVED-5G/SDK-CLI

[45] EVOLVED-5G/Instructions: This repo will contain documentation to develop a NetApp along how to run CI/CD pipelines, from https://github.com/EVOLVED-5G/Instructions

[46] EVOLVED-5G /SDK-CLI/location_subscriber_examples.py at master, from https://github.com/EVOLVED-5G/SDK-CLI/blob/master/examples/location_subscriber_examples.py

[47] EVOLVED-5G /SDK-CLI/qos_awereness_examples.py at master, from https://github.com/EVOLVED-5G/SDK-CLI/blob/master/examples/qos_awereness_examples.py

[48] EVOLVED-5G/cicd at develop cicd/pac, from https://github.com/EVOLVED-5G/cicd/tree/develop/pac

[49] Red Hat OpenShift makes container orchestration easier, from https://www.redhat.com/en/technologies/cloud-computing/openshift

[50] GitHub Security, from https://github.com/security

[51] 5GENESIS, from https://github.com/5genesis

[52] 5GENESIS – 5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing, from https://5genesis.eu/

[53] 5GENESIS – 5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing, D2.4 Final report on facility design and experimentation planning Final report on facility design and experimentation planning, from https://5genesis.eu/wp-content/uploads/2020/07/5GENESIS_D2.4_v1.0.pdf

[54] 5GENESIS – 5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing, D3.8 "Open API, service-level functions and interfaces for verticals",

from Open API, service-level functions and interfaces for verticals, from https://5genesis.eu/wp-content/uploads/2021/04/5GENESIS_D3.8_v1.0.pdf

[55] EVOLVED-5G, Deliverable 2.2 “Design of the NetApps development and evaluation environments”

[56] RELY-TSN-BRIDGE: TSN Switch | Relyum, from https://www.relyum.com/web/rely-tsn-bridge/

[57] RELY-TSN-PCIe | Relyum, from https://www.relyum.com/web/rely-tsn-pcie/

[58] TS 129 222 - V15.2.0 - 5G; Common API Framework for 3GPP Northbound APIs (3GPP TS 29.222 version 15.2.0 Release 15), from https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[59] FastAPI, from https://fastapi.tiangolo.com/

[60] Swagger, from https://swagger.io/

[61] Redoc, from https://redoc.ly/

[62] PostgreSQL: The world’s most advanced open source relational database, from https://www.postgresql.org/

[63] pgAdmin, from https://www.pgadmin.org/

[64] CoreUI: Free Bootstrap Admin Template, from https://coreui.io/

[65] OpenStreetMap, from https://www.openstreetmap.org/about

[66] JSON Web Tokens, from https://jwt.io/

[67] Docker, from https://www.docker.com/