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

[H2020 - Grant Agreement No.101016608]

Deliverable D5.3

NetApp Certification and Release to Marketplace (intermediate)

Editor Alejandro Molina (TID)

Contributors FOGUS, ATOS, NCSRD, INF, MAG, UMA

Version 1.0

Date January 31, 2023

Distribution PUBLIC (PU)

DISCLAIMER

REVISION HISTORY

Revision	*Date*	*Responsible*	*Comment*
0.1	*Oct 17th 2022*	*Alejandro Molina (TID)*	*Edit ToC*
0.2	*Nov 11th 2022*	*Alejandro Molina David Artuñedo (TID)*	*First Inputs TID*
0.3	*Nov 18th 2022*	*Ricardo Marco (ATOS) Bruno García (UMA) George Makropoulos (NCSRD)*	*Inputs ATOS, UMA, NCSRD*
0.4	*Nov 19th 2022*	*Yiannis Karadimas (MAG)*	*First inputs MAG*
0.5	*Dec 1st 2022*	*Bruno García (UMA)*	*Review UMA*
0.6	*Dec 15th 2022*	*Alejandro Molina David Artuñedo (TID)*	*Apply comments and add more content*
0.7	*Jan 15th 2022*	*George Makropoulos (NCSRD)*	*Review NCSRD*
0.8	*Jan 22nd 2022*	*Alejandro Molina David Artuñedo*	*Apply final comments*
0.9	*Jan 27th 2022*	*Alejandro Molina David Artuñedo (TID)*	*Final Version*
0.10	*Jan 31st 2022*	*Ines De Ibarguen (TID)*	*Final Quality Check*

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*	*Harilaos Koumaras* *George Makropoulos* *Dimitrios Fragkos* *Anastasios Gogos*
INF	*INFOLYSIS P.C.*	*Christos Sakkas* *George Theodoropoulos* *Antonios Varkas*
FOGUS	FOGUS INNOVATIONS & SERVICES P.C.	*Dimitris Tsolkas* *Georgios Krommydas* *Anastasia Papafotiou*
MAG	*MAGGIOLI*	*Yiannis Karadimas*
UMA	*UNIVERSITY OF MÁLAGA*	*Almudena Diaz* *Bruno García* *Francisco Luque* *Mª del Mar Moreno*

GLOSSARY

Abbreviations/Acronym	*Description*
3GPP	*3rd Generation Partnership Project*
API	*Application Programming Interface*
CA	*Certification Authority*
CAPIF	*Common Application Programming Interface Framework*
CI/CD	*Continuous Integration / Continuous Deployment*
IaC	*Infrastructure as Code*
K8s	*Kubernetes*
MD	*Markdown*
Network App	*Network Application*
RPA	*Robotic Process Automation*
TSN	*Time Sensitive Networking*
UI	*User Interface*
URL	*Uniform Resource Locator*
UX	*User Experience*
VPN	*Virtual Private Network*

EXECUTIVE SUMMARY

EVOLVED-5G responds to the *5G PPP ICT-41-2020 5G innovations for verticals with third party services* call, 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 test their applications.

The EVOLVED-5G project accomplishes this vision by encouraging the creation of a Network App ecosystem revolving around a 5G facility which will provide the tools and processes for the development, verification, validation, and certification of Network Apps as well as their validation on top of actual 5G network infrastructures, and mechanisms for market releasing.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 SCOPE

The scope of this deliverable is to expose implementation details for both the EVOLVED-5G Certification Environment and Certification Process and how the release of the Network Apps to the marketplace works. Its intention is to be accessible to a broad variety of research individuals and communities as described below:

- **Project Consortium**: To validate that all objectives and proposed technological advancements have been analyzed and to ensure that, through the proposed implementations, further work can be derived. Furthermore, the deliverable sets a common understanding among the consortium with regards to:
  - The implementation details of the Certification Environment related to the Network App lifecycle in the context of the EVOLVED-5G certification process, including tools and technologies to be used.
  - The implementation details of the Marketplace, the final step of the Network App lifecycle in the context of the EVOLVED-5G project.

- **Industry 4.0/Industry 4.0 developers and Factories of the Future (FoF) vertical groups**: To set a common understanding of the technologies and the design principles that underline the Certification Process design and its implementation, along with the Marketplace release process.

- **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 being 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 Certification Process and Marketplace implementation.

1.2 OBJECTIVES

This deliverable is the third deliverable of WP5 to be delivered by the project consortium during the 36-month work plan. This deliverable, titled *"D5.3 NetApp Certification and Release to Marketplace (intermediate)"*, presents the tools, activities and methodologies that materialize the EVOLVED-5G Certification Environment and Process, with an emphasis on the efficiency and security of the implementations, as well as the process by which certified Network Apps are released to the Marketplace. This document describes the results of task T5.3 *"EVOLVED-5G Network Apps Certification and Release to the Marketplace"*, providing a detailed description of a Network App going through the different steps of the Certification Process, which is described in section 2. In sections 3 and 4, there is a description about the mechanisms implemented in the Marketplace to

enable the integration with the CI/CD platform and how the Network Apps are released to the Marketplace. There will be an updated version of this deliverable, in D5.6 *"NetApps Certification and Release to Marketplace (Final)"* to be released at the end of the Project, with the final implementation details carried out during the rest of the Project.

1.3 STRUCTURE

This deliverable is organized in the following manner:

- **Section 1. Introduction:** This section describes the Deliverable target audience, objectives, and structure.

- **Section 2. Certification Process:** This section describes the Certification Environment focusing on the CI/CD toolset with a collection of software industry leading tools for automation.

- **Section 3. Marketplace and Open Repository Integration:** This section describes the approach used to integrate the Open Repository and the Marketplace, allowing Certified Network Apps to be published in the Marketplace.

- **Section 4. Release to Marketplace:** This section describes the Marketplace's internal processes as well as the platform's expected functionality such as versioning, creating, and removing Network Apps.

2 CERTIFICATION PROCESS

This section presents the description of the certification tools which are utilized during the Certification Process, the Certification Environment in which the Certification Process takes place, the Certification Pipeline built to automate the Certification tests, and finally, the Certification report produced at the end of the process. Moreover, it exemplifies an execution of the Certification process on top of a Network App, acting as an example of how a Network App is certified.

2.1 CERTIFICATION TOOLS

In Deliverable 3.2 [1] (and more specifically in Section 3) a grouping of the Certification requirements in three categories is presented. This Section described in detail the tools that fall under each category and are utilized towards the fulfillment of those requirements. More specifically the tools in each category are as follows:

- **Software product quality:** SonarQube, Robot Framework, Nmap
- **Security**: Trivy, SonarQube.
- **Licensing**: Debricked

2.1.1 Trivy

Trivy [2] is a security scanner towards vulnerability, misconfiguration for container and other artifacts. Trivy detects vulnerabilities in operating system packages (such as Alpine, RHEL, and CentOS) as well as language-specific packages (Bundler, Composer, npm, yarn, etc.). Furthermore, Trivy scans Infrastructure as Code (IaC) files (such as Terraform and Kubernetes descriptors) for potential configuration issues that expose your deployments to attack. Trivy also examines the containers for hardcoded secrets such as passwords, Application Programmable Interface (API) keys, and tokens.

Figure 1 Trivy logo

2.1.2 SonarQube

SonarQube [5] is an open-source platform developed by SonarSource for continuous code quality inspection. Sonarqube is a static code analysis tool that assesses a software project's overall quality. SonarQube can generate a detailed report on code quality, identifying issues such as code duplication, code smells, and vulnerabilities.

Sonarqube was selected as the quality assessment tool in EVOLVED-5G to measure the overall quality of Network Apps. SonarQube provides numerous benefits to Network App developers, including increased productivity, improved developer skills, and increased consistency for Network Apps deployed in the EVOLVED-5G ecosystem.

Figure 2 SonarQube logo

2.1.3 Robot Framework

The Robot Framework [3] is a free and open automation framework. It is suitable for test automation as well as robotic process automation (RPA). Robot Framework Foundation provides support for Robot Framework.

EVOLVED-5G utilizes Robot Framework to automate some of the certification tests. EVOLVED-5G includes all components needed to run Robot Framework tests in a Docker image, which EVOLVED-5G CI/CD deploys and runs using Jenkins to collect test results. The results of these tests are incorporated into the Jenkins pipeline flow.

Figure 3 RobotFramework logo

2.1.4 Nmap

Nmap (Network Mapper) [4] is a network scanner. Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap includes several features for probing computer networks, such as host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. During a scan, Nmap can adjust to network conditions such as latency and congestion.

EVOLVED-5G uses Nmap to ensure that connectivity is working in the ports specified by Network App for their application.

Figure 4 NMAP logo

2.1.5 Debriked

Debricked [7] assists software companies in reducing risk when making use of Open Source in product development. Approximately 90% of software development companies use Open Source but mitigating risks such as vulnerabilities is a costly and time-consuming task that is frequently overlooked. Debricked's tool allows for increased use of Open Source while minimizing risks, allowing for rapid development while remaining secure. The service is powered by cutting-edge machine learning, allowing for exceptional data quality as well as instant updates whenever a new vulnerability is discovered. Debricked is one of a kind in the world of cloud computing due to its high precision, flawless User Experience (UX), and unique abilities to customize the service to your company's needs.

Figure 5 Debricked logo

2.2 Certification Environment

As it has been described in detail in Deliverable 3.2, the EVOLVED-5G project utilizes two different 5G platforms located in Athens (composed by two sites: NCSR Demokritos and Cosmote) and Malaga. The two platforms provide multiple set-ups in a modular way so as to support and enhance a variety of test cases and experiments related to EVOLVED-5G. More specifically the capabilities vary in 5G radio access and exposure of the 5G APIs, a container orchestration system, measurement probes, and the possibility of integrating new devices as required for the vertical applications. On top of that, both platforms make use of the Open5Genesis framework for the coordination of the experiments. However, an important distinction between the two platforms is the fact that Malaga platform also provides Time-Sensitive Networking (TSN) capabilities based on a set of standards focused on improving the reliability of network communication and the transmission of data with very low and controlled latency.

Given the fact that among the 12 Network Apps targeted by EVOLVED-5G [8] there are use cases where very low latency is a critical factor, the allocation to the platform that each Network App will be validated and certified was deemed necessary, taking also into account the distinct characteristics that each use case requires. In the light of the above and following the categorization previously described in WP4 deliverables, the following table summarizes the selection of the platform for each Network App.

Network App	Athens Platform	Malaga Platform
Remote assistance in AR		X
Chatbot assistant	X	
Digital/physical twin	X	
CAFA-NetMapper		X
Smart irrigation and Agriculture		X
Industrial grade 5G connectivity	X	
Anomaly Detection	X	
Traffic Management	X	
ID Management and Access Control	X	
5G SIEM	X	
5G Teleoperation		X
Localisation		X

2.2.1 Malaga Platform Kubernetes

The Malaga platform contains a high-availability Kubernetes deployment that is accessible (via VPN access) during the Validation and Certification processes. The deployment is based on a multi-master architecture (three master nodes) with four worker nodes, one of them fine-tuned for storage. More information about the Kubernetes deployment can be seen in Section 2.4.3 of Deliverable D5.2 [9].

2.2.2 Athens Platform Kubernetes

On the other hand, Athen's platform integrates a Kubernetes (K8s) cluster that consists of three virtual machines (i.e., one master and two working nodes), to serve the purpose of certification of the Network Apps. Each node allocates 2x vCPUs and 4GB RAM and it is accessible via VPN connectivity from Telefónica's premises (i.e., CI/CD environment). It should be noted that the same Kubernetes cluster is used in the validation process [D5.2].

2.3 CERTIFICATION PIPELINE

The Certification process consists of a set of tests that certifies concrete aspects that need to be fulfilled by the Network App being certified (e.g., Vulnerability scan, Common API Framework (CAPIF) and Network Exposure Function (NEF) APIs compliance, …). These tests are implemented as individual steps. All steps are concatenated composing the Certification Process. This section will go over the certification steps that a Network App must pass in order to complete the certification.

2.3.1 CI/CD Environment

CI/CD methodology has been adopted in the EVOLVED-5G Project due to the advantages it offers (as described in D3.2 [1]). This CI/CD methodology also offers a way of establishing a central location where each project participant can benefit from the pipelines already in place to cut down on the time needed to test, build, or automatically deploy their Network Apps.

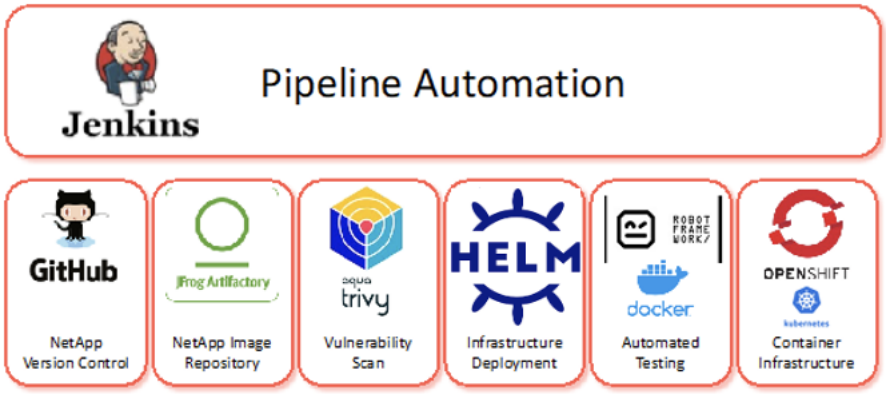


Figure 6 CI/CD Environment

The list of tools that will be used in the EVOLVED-5G project can be found in Figure 6 CI/CD Environment. As it can be seen, Jenkins is the tool selected to actually execute all the various steps and will be in charge of deciding whether or not to move on to the subsequent steps based on the success or failure of the previous steps, as defined in the certification pipeline. Jenkins, during the steps' execution, is invoking each of the tools described earlier, such as Trivy or Sonarqube. Jenkins is also responsible for the orchestration and management of the infrastructure for creating, deploying, and destroying artifact as needed to complete the steps. Finally, Jenkins collects all Steps results and generates the Certification Report that is sent to the Network App developer upon the completion of the process.

2.3.2 Certification steps

The Network App certification steps are the path a Network App must fulfil to obtain the Certification tag guarantying a full compatibility with the EVOLVED-5G ecosystem. A developer can start the certification path at any time, i.e., it does not need to follow the whole Network App lifecycle implemented and described in EVOLVED-5G [9]. However, it is strongly advised to follow the path defined to increase the chances of having a successful certificated Network App.

If a developer wishes to certify its Network App to be fully compatible with the EVOLVED-5G project, the first step from developer's view is to get in contact with the certification authority (CA) within the project to request starting the process. At this point, it is still open how this communication will be channelled.

Once the CA has all the requisites gathered from the developer, will then start the certification process. It should be noted that such process does involve the CI/CD pipelines to be executed in an automated manner.

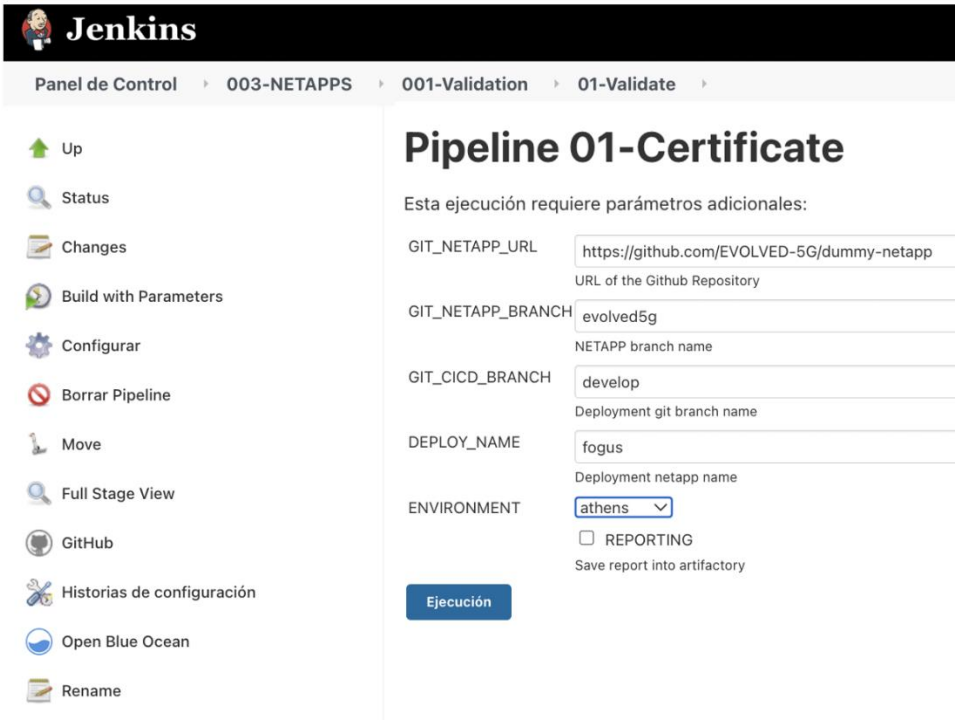


Figure 7 Input Parameters Certification pipeline

The Certification pipeline needs to provide the following parameters to start the Certification:

- GIT_NETAPP URL: GitHub repository URL of the Network App
- GIT_NETAPP_BRANCH: Selected branch of the GitHub repo to be certified
- GIT_CICD_BRANCH: Branch of the CI/CD that will be executed. This branch is important for the environment in order to have different configurations.
- DEPLOY_NAME: Name of the Network App.
- ENVIRONMENT: Whether the certification process will be using Málaga or Athens Kubernetes infrastructure
- REPORTING: Boolean parameter to create a report or not during the execution. This parameter is always True by default.

To prepare some Network App tests, the Certification Pipeline must initially interact with the Athens or Málaga 5G Platform environments. At a second stage, the Certification Pipeline must certify that the Athens and Málaga environments are available to execute the Certification process, and once completed, the environment must be cleaned up so that certification of other Network Apps can be executed properly.

The following steps have been identified and will be modelled as Jenkins steps in the Certification Pipeline:

Preconfiguration Step: Before carrying out any of the procedures to certify the Network App, Jenkins will first verify that the Jenkins slave is connected to the relevant platform. A network tool like ping to determine whether there is connectivity with the Kubernetes components is used. If this check fails, a message will be sent to the CI/CD's responsible party to re-establish connectivity between the Jenkins and the various platforms. Moreover, Experiments will be carried out in this step to assess the capabilities of the

EVOLVED-5G infrastructure and software components. Deliverable 5.1 [10] contains additional information about these experiments.

Step 1: The first step is the Source Code Quality Analysis and will be executed using SonarQube to assess the overall quality of a software project. SonarQube generates a detailed report on code quality, identifying issues such as code duplication, code smells, and vulnerabilities. A quality gate has been defined to specify what types of quality aspects must be met. In order to proceed with the pipeline execution, SonarQube analysis must pass this Quality Gate defined for Network Apps.

Sonarqube takes as input parameter the GitHub repository of the Network App and the Branch to be analyzed.

Upon completion of the SonarQube analysis, a JSON file is generated with the following results: number of blocker issues, number of critical issues, number of major issues and number of minor issues.

Summary

Severity	Number of vulnerabilities
blocker	0
critical	6
major	35
minor	14

Figure 8 SonarQube result example

The Quality Gate defined for Certification consists of having Zero blocking issues in order to have a SUCCESS status in this step.

Step 2: The second step is the Source Code Security Analysis that searches for vulnerabilities in the Network App source code through Trivy tool. If Trivy reports critical vulnerabilities with available patches, the pipeline will be halted. Otherwise, the pipeline will proceed to **Step 3** after the vulnerability scan is completed.

Trivy takes as input parameter the Network App GitHub repository to be analyzed.

Upon completion of the Trivy analysis, a JSON file is generated with the following results: number of Critical issues, number of high severity issues and number of medium severity issues.

Summary

Severity	Number of vulnerabilities
CRITICAL	1
HIGH	5
MEDIUM	3

Figure 9 Trivy Source Code Security Analysis result example

Along with the number of issues determined, Trivy includes information on each issue and specifies if there is a solution defined to solve the issue.

Vulnerabilities

Severity	ID	Title	PkgName	InstalledVersion	FixedVersion
CRITICAL	CVE-2021-35042 (https://nvd.nist.gov/vuln/detail/CVE-2021-35042)	django: potential SQL injection via unsanitized QuerySet.order_by() input	Django	3.1.7	3.1.13, 3.2.5

Figure 10 Trivy Source Code Security Analysis issue detail example

If the issue is fixed in a later version of the component where the issue has been detected, Trivy includes the information to help the developer fix the issue. The acceptance criteria defined for this step is having Zero critical issues in order to have a SUCCES status.

Step 3: The third step is the Source Code Secrete Leakage analysis to search for leaks of secrets in the source code. This step will also make use of the Trivy tool, which includes mechanisms for detecting raw credentials embedded in code. The pipeline will be halted if Trivy detects any critical information in the code. Otherwise, after the leak scan is completed, the pipeline will proceed to **Step 4**.

Trivy takes as input parameter the Network App's containerized image to be analyzed and reports the severity of the issues detected as well as the file and line in the source code where the issue has been detected.

Passwords detected in commit history

Severity	Description	Match	File	Author	Date
low	Dominios expuestos	image: dockerhub.hi.inet	fogus/templates/dbnetapp-deployment.yaml (https://github.com/Telefonica/Evolved5g-FogusNetApp/blob/699cd0dc9483f99e33e4340884d814dfb8bc0821/fogus/templates/dbnetapp-deployment.yaml#L35-L35)	Alejandro Molina Sanchez	2022-09-29 12:43
low	Dominios expuestos	image: dockerhub.hi.inet	fogus/templates/netappdjango-deployment.yaml (https://github.com/Telefonica/Evolved5g-FogusNetApp/blob/699cd0dc9483f99e33e4340884d814dfb8bc0821/fogus/templates/netappdjango-deployment.yaml#L34-L34)	Alejandro Molina Sanchez	2022-09-29 12:43
low	Dominios expuestos	- image: dockerhub.hi.inet	fogus/templates/netappfe-deployment.yaml (https://github.com/Telefonica/Evolved5g-FogusNetApp/blob/699cd0dc9483f99e33e4340884d814dfb8bc0821/fogus/templates/netappfe-deployment.yaml#L28-L28)	Alejandro Molina Sanchez	2022-09-29 12:43

Figure 11 Trivy secret leakage example

The acceptance criteria defined for this step is having zero high Severity issues to have a SUCCES status.

Step 4: This step will verify that the Network App has completed the Validation phase. To check this, Jenkins will search for the Network App image inside the Validation folder in Artifactory. If the Network App image exists in the folder, it will be grabbed from there and continue to **Step 5.** Otherwise, the pipeline will create a new image from the GitHub Repository. Furthermore, in this step Jenkins will execute a script based on NMAP to check and validate that the connectivity of the Network App works correctly.

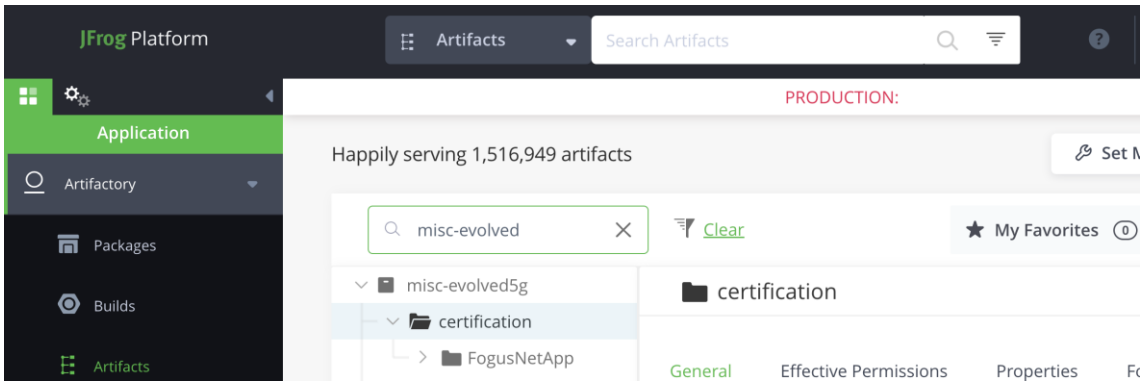


Figure 12 Open Repository Certification Folder

Step 5: Before deploying the Network App, Jenkins will make sure that Network App image is secure. For this, Jenkins will use again Trivy as external tool to validate the security vulnerabilities of the Network App image. If security critical issues are reported, the pipeline will stop. Otherwise, the pipeline will continue to **Step 6.**

Trivy takes as input parameter the Network App image to be analyzed. This image is generated in this step / grabbed from Validation folder in Artifactory.

Upon completion of the Trivy analysis, a JSON file is generated with the following results: number of Critical issues, number of High severity issues, number of medium severity issues, number of Low severity issues and number of unknown issues.

Severity	Number of vulnerabilities
CRITICAL	21
HIGH	356
MEDIUM	302
LOW	595
UNKNOWN	3

Figure 13 Trivy Security Image result example

Along with the number of issues determined, Trivy includes information on each issue and specifies if a solution is defined to solve the issue.

Vulnerabilities

Severity	ID	Title	PkgName	InstalledVersion	FixedVersion
CRITICAL	CVE-2022-32221 (https://nvd.nist.gov/vuln/detail/CVE-2022-32221)	curl: POST following PUT confusion	curl	7.74.0-1.3+deb11u3	
CRITICAL	CVE-2021-35042 (https://nvd.nist.gov/vuln/detail/CVE-2021-35042)	django: potential SQL injection via unsanitized QuerySet.order_by() input	Django	3.1.7	3.1.13, 3.2.5

Figure 14 Trivy Security Image issues detail example

The acceptance criteria defined for this step is having Zero Critical issues with a possible solution to have a SUCCES status.

Step 6: Once Jenkins has certified that the Network App is secure to be deployed, Jenkins need to prepare the environment for testing CAPIF and NEF APIs . In Step **6**, Jenkins will deploy a clean CAPIF instance to test CAPIF API usage from the Network Apps. Each time the certification pipeline is executed, a clean deployment of CAPIF will be created.

Step 7: After deploying CAPIF in Athens/Malaga Kubernetes infrastructure, Jenkins will run a series of RobotFramework tests to ensure that the functionality of CAPIF is working properly. This process can guarantee that any potential problem during CAPIF tests of the Network App is not related to CAPIF instance but to the Network App itself. If this step fails, the certification pipeline will be halted, and the environment cleaned for next Certification execution.

Step 8: After deploying CAPIF, Jenkins will deploy NEF in the selected environment (Athens or Malaga) along with the needed configurations to onboard this NEF instance in the CAPIF. NEF will register in CAPIF as an API Exposer and will publish NEF APIs to make them available to the Network Apps.

Step 9: Similarly, to Step 7, Jenkins will perform several RobotFramework tests after installing NEF to check that its functionality is operating properly. If this step fails, the certification pipeline will be interrupted, and the environment will be cleaned for next Certification execution.

Step 10: With CAPIF and NEF deployed and working properly, Jenkins will use nested Pipeline to deploy the Network App in the selected environment (Athens or Málaga). If the pipeline reports any error during the deployment, the certification pipeline will jump to **Step 19** to clean the environment and stop. If no errors are reported, the pipeline will continue to **Step 11**.

Step 11: With the Network App Deployed, the next step is to check that the Network App registers in CAPIF Core Function properly as an API Invoker. To certify the registration, Jenkins will parse the logs of CAPIF to check that the Network App has been registered properly by CAPIF Core Function.

Step 12: Once the Network App has registered in CAPIF Core Function as API Invoker, the next step is for the Network App to Discover APIs published in CAPIF. To certify that Network App is using Discover API from CAPIF Core Function, Jenkins will parse the logs from CAPIF Core Function Discover API to double check that the REST API has been consumed by the Network App.

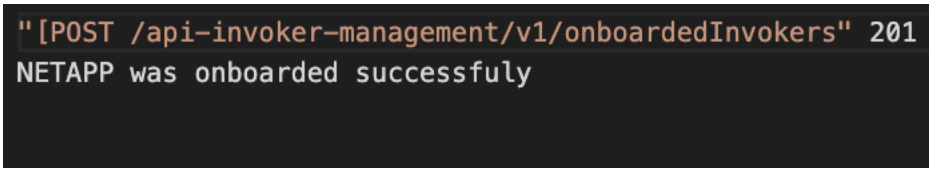


Figure 15 Output logs used for Checking CAPIF onboarding

The acceptance criteria defined for this step is having Zero Critical issues with a possible solution to have a SUCCES status.

Step 13: As part of the API dialogue between the Network App and CAPIF Core Function, the Network App will receive CAPIF Events. These events are notifications coming from CAPIF Core Function to the Network App, and the CAPIF Event callback URL is set by the Network App in the API Invoker Registration. Jenkins will certify that CAPIF Events are received by Network App properly by parsing the logs from CAPIF

Core Function Events API. In the logs, "200 OK" responses from the Network App must appear as the result of Events are sent.

```
"GET /service-apis/v1/allServiceAPIs?api-invoker-id=392bde5edc39437beb7c08841c7130 HTTP/1.0" 200
DISCOVER APIs work correctly
```

Figure 16 Output logs used for Checking Discovery Services

Step 14 and 15: The Network App Discover NEF API using CAPIF Core Function Discovery API. Two APIs are exposed by NEF: AsSessionWithQoS and MonitoringEvent. Jenkins will check which of these APIs are consumed by the Network App and add them to the Certification Report.

Step 16: As part of the API dialogue between the Network App and NEF Services, the Network App will receive NEF Events. These events are notifications coming from NEF Services to the Network App, and the NEF Event callback URL is set by the Network App in the NEF API Requests. Jenkins will certify that NEF Events are received by Network App properly parsing the logs from NEF Services. In the logs, "200 OK" responses from the Network App must appear as the result of NEF Events are sent.

Step 17: Network App might support scaling out horizontally by defining a ReplicaSet workload resource. A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical pods. ReplicaSets' are recommended to be used defining Deployments. Deployment provides declarative updates for Pods and ReplicaSets. Jenkins will increase the number of instances of Network App deployment and certify that Network App Pods scale properly.

Step 18: Jenkins will shrink the number of pods after scaling out the Network App by increasing the number of pods and certifying that the Network App adjusts to the scale defined.

Step 19: Once all the previous tests have been executed, Jenkins destroys the Network App and certifies is decommissioned properly. Jenkins will then remove the Network App from the container infrastructure, which will trigger Network App removal clean up.

Step 20: As part of the Network App removal clean up, the Network App needs to unregister from CAPIF Core Function. Jenkins will certify that the Network App has unregistered properly by querying CAPIF Core Function Database.

Step 21 & 22: Jenkins will remove NEF and CAPIF from the platform where they have been installed once the Network App has been destroyed and unregistered from CAPIF.

Step 23: After all functional tests have been completed, the Certification pipeline will finally collect information about Network App's open-source licenses. Jenkins will use the Debriked Compliance external tool [7] for this operation.

Debriked takes as input parameter the Network App GitHub repository and the branch to be analyzed.

Upon completion of the Debriked analysis, a JSON file is generated with the Licenses summary found.

Licenses Summary

License Name	Families	Dependencies
MIT	Permissive	710
Python-2.0	Permissive	1
ISC	Permissive	108
Debricked Unknown License	Unknown	140
Unlicense	Permissive	5
BSD-2-Clause	Permissive	38
BSD-2-Clause-NetBSD	Permissive	2
WTFPL	Permissive	2
BSD-3-Clause	Permissive	25
CC0-1.0	Permissive	31
Apache-2.0	Permissive	25
GPL-2.0-only	Strong copyleft	14
0BSD	Permissive	3
LGPL-3.0-only	Weak copyleft	1
CC-BY-4.0	Permissive	2
Beerware	Permissive	1
GPL-1.0-or-later	Strong copyleft	2
LGPL-2.0-or-later	Weak copyleft	1
ZPL-2.1	Unknown	1
LGPL-2.1-or-later	Weak copyleft	1
Zlib	Permissive	1
LGPL-2.1-only	Weak copyleft	1

Figure 17 Debriked License analysis result example

Step 24: The final step will compile all the information stemming from the previous steps and generate the final report with all of the important information for the certification. This report will indicate whether or not the Network App was successfully certified.

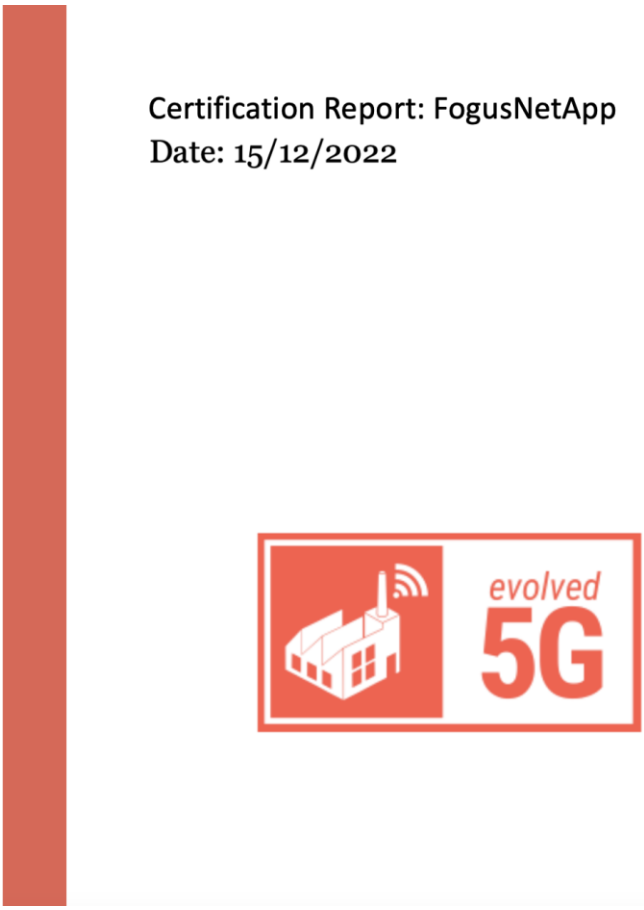


Figure 18 Report Cover

This report is sent via email to the Network App developers so as to get the complete picture of the Certification process over their Network Apps. In case of Successful Certification, the email with the report also contains a Fingerprint. This Fingerprint information will be required during the Marketplace release of the Network App.

At this stage, the developer will have a certified Network App which, can be released to the Marketplace and offered to the public. This step has been decoupled with the aim to provide more freedom to the developer, who might be interested in having a certified Network App but not exposing it publicly. This process is defined in Section 4 Release to Marketplace

Some steps require further explanation, and therefore, the following sections describe the most elaborated steps in more detail.

2.3.2.1 Network App Quality Analysis

In this section, a detailed description of how SonarQube is used and how the analysis mentioned in **Step 1** included in the certification report will be provided.

Sonarqube tool takes as input the Network App repository in GitHub to get access to Network App code. It performs several analyses on source code to detect bugs, vulnerabilities, etc. It reports the items found in each category with a detailed description per issue and guides the developer to the proposed fix in case there is one identified.

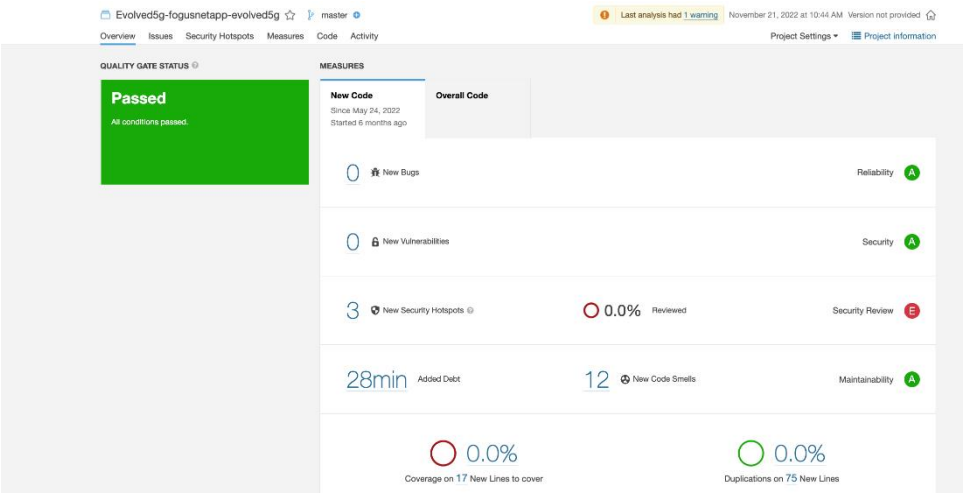


Figure 19 Sonarqube dashboard

In Figure 19 Sonarqube dashboard, there is a pass in the section that indicates that the Network App has successfully completed all of the verification steps defined in the quality gate. The quality gate is the project's minimum requirement for meeting the quality requirement.

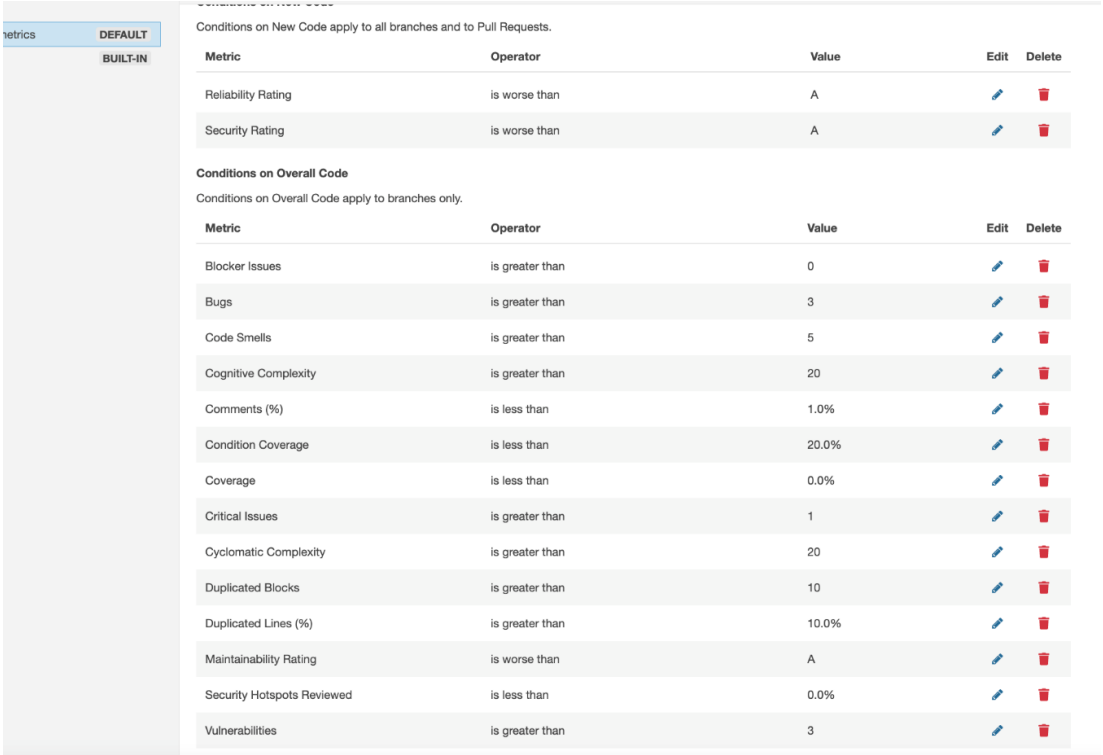
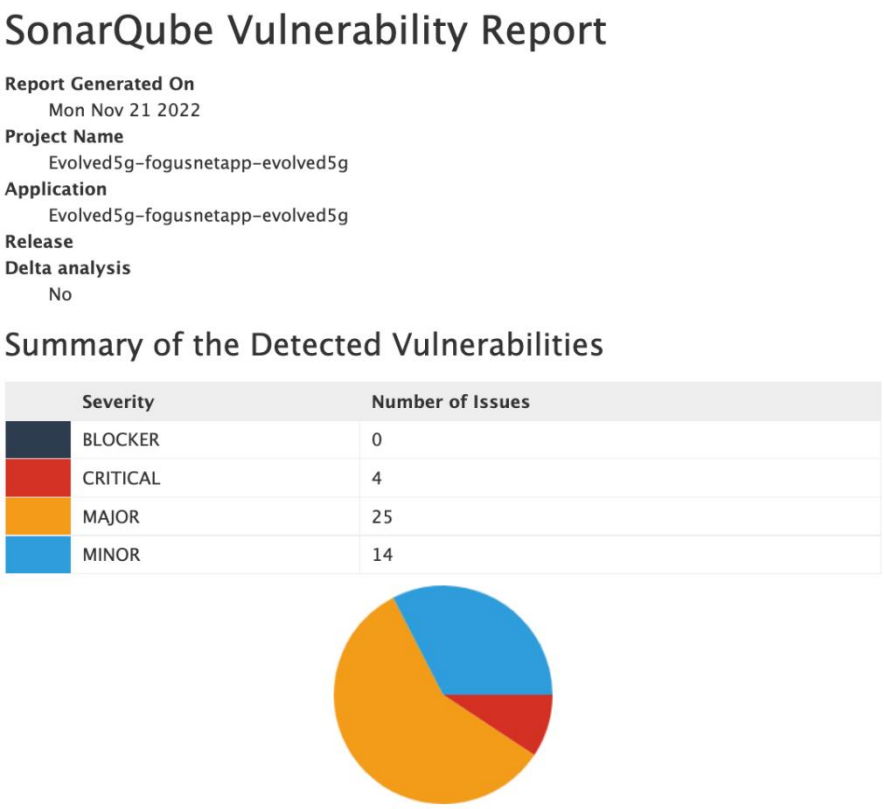


Figure 20 Sonarqube configuration

Figure 20 Sonarqube configuration illustrates several parameters defined in this quality gate. For example, bugs' metric cannot be greater than 3 in order to meet this criterion, otherwise the analysis will fail, and the pipeline will not continue with the execution of the certification. On such occasion a relevant report will be provided to the developer in order to fix this problem.

This analysis will provide some information about the Network App's quality aspects, which will be gathered in reports generated by the CI/CD. In the Figure 21 SonarQube Report, an example of the summary with all the problems stemming from the analysis, is presented



SonarQube Vulnerability Report

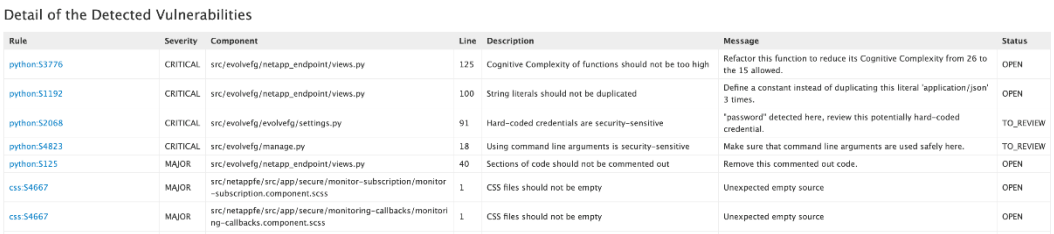
Report Generated On
Mon Nov 21 2022
Project Name
Evolved5g-fogusnetapp-evolved5g
Application
Evolved5g-fogusnetapp-evolved5g
Release
Delta analysis
No

Summary of the Detected Vulnerabilities

Severity	Number of Issues
BLOCKER	0
CRITICAL	4
MAJOR	25
MINOR	14

Figure 21 SonarQube Report

More information about these errors can be obtained by looking at the table generated below. In this table, a detailed description of each problem discovered during the analysis can be seen. As an example, in the second row of the table, there is an error related to line duplication. Instead of repeating the same line of code multiple times, Sonarqube recommends that the developer should create a constant to reduce the overall code size.



Detail of the Detected Vulnerabilities

Rule	Severity	Component	Line	Description	Message	Status
python:S3776	CRITICAL	src/evolvefg/netapp_endpoint/views.py	125	Cognitive Complexity of functions should not be too high	Refactor this function to reduce its Cognitive Complexity from 26 to the 15 allowed.	OPEN
python:S1192	CRITICAL	src/evolvefg/netapp_endpoint/views.py	100	String literals should not be duplicated	Define a constant instead of duplicating this literal 'application/json' 3 times.	OPEN
python:S2068	CRITICAL	src/evolvefg/evolvefg/settings.py	91	Hard-coded credentials are security-sensitive	"password" detected here, review this potentially hard-coded credential.	TO_REVIEW
python:S4823	CRITICAL	src/evolvefg/manage.py	18	Using command line arguments is security-sensitive	Make sure that command line arguments are used safely here.	TO_REVIEW
python:S125	MAJOR	src/evolvefg/netapp_endpoint/views.py	40	Sections of code should not be commented out	Remove this commented out code.	OPEN
css:S4667	MAJOR	src/netappfe/src/app/secure/monitor-subscription/monitor-subscription.component.scss	1	CSS files should not be empty	Unexpected empty source	OPEN
css:S4667	MAJOR	src/netappfe/src/app/secure/monitoring-callbacks/monitoring-callbacks.component.scss	1	CSS files should not be empty	Unexpected empty source	OPEN

Figure 22 SonarQube detailed report

Finally, this information is extremely useful for the developer in order to improve the quality of the code. This step sets a quality bar to ensure that all Network Apps uploaded to the Marketplace will go through a thorough quality analysis ensuring a high-quality application.

2.3.2.2 Network App Security Analysis

The Trivy tool will analyze the Network App in a variety of ways. These analyses will attempt to determine whether the Network App meets the security and privacy requirements of the EVOLVED-5G project. Trivy conducts three distinct analyses, which are presented further below:

- **Code Vulnerabilities:** Trivy does a static code analysis of the repository looking for vulnerabilities that are included in their database of vulnerabilities.

- **Secrets Leakage:** Trivy searches through all of the commits in the repository for any hidden leaks in the code. This capability is critical since it is likely that certain secrets may be hardcoded during the early stages of the software project's development, and once deleted, those secrets will be retained in the commit history.

- **Container vulnerabilities:** Trivy will show all the vulnerabilities found in the container images. Trivy displays a thorough description of the problem, including the name of the affected library and the version that resolves the issue (if exists)

2.3.2.3 Network App Network Analysis

During the certification process, the Network App follows a challenging course to obtain a successful result. There has been implemented different tests to analyse the behaviour of a Network App under specific conditions. Within such tests, the Network App network behaviour is one of the tests performed to the Network App to check a full compatibility with what it has been described in the docker compose file. Basically, all Network Apps in EVOLVED-5G follow the same pattern, this is, to have a docker compose file declaring all the ports to be used by the Network App. During the deploy pipeline, the Network App is challenged to verify that all the ports declared in the docker compose are listening for connections.

To perform the Network App network analysis the certification test is using the Nmap [12] tool to scan the ports gather from the docker compose file. Therefore, while the Network App is being deployed and before going further, it is tested the ports to ensure that all of them are listening.

2.3.2.4 Network App deployment

After the initial set of tests being successful, the Network App will be deployed in the platform where it will be certified. In the table below, the mapping between the Network App and the Platform is presented The mapping is based on the distinct characteristics that each Network App and the respective use case defines.

Netapp	PLATFORM
INF	ATHENS
UMS	MALAGA
ININ	ATHENS
PAL	MALAGA
QUCOMM	ATHENS
IQB	ATHENS
IMM	MALAGA
CAF	MALAGA
GMI-AERO	ATHENS
8BELLS	ATHENS
UMA-CSIC	MALAGA

Figure 23 Network App platform relation

In order to deploy each Network App, a YAML descriptors is produced with the information needed to instantiate Network App images into the container platform. These Helm descriptors are YAML files that contain the exact settings required for the Kubernetes platform to deploy Kubernetes resources like service, route, deployments, etc.

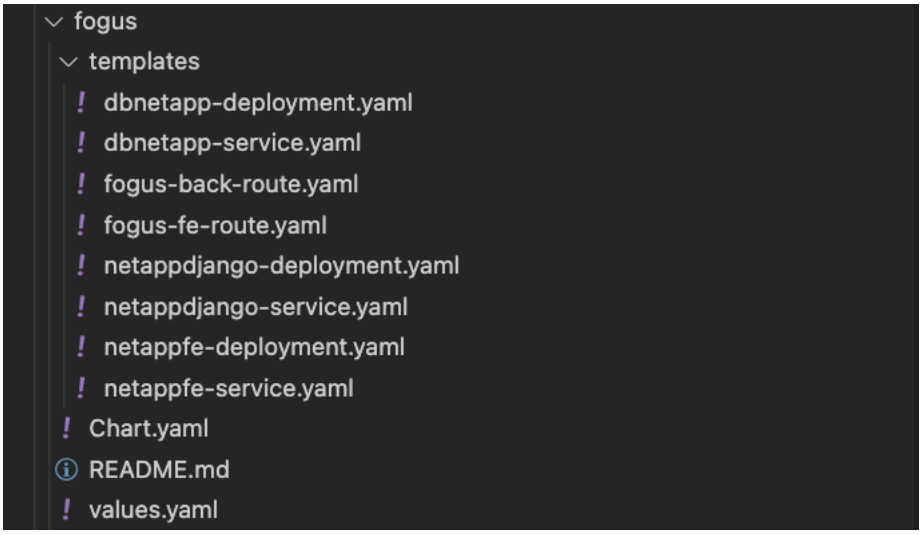


Figure 24 Helm files structure

In the Figure 24 Helm files structure the different resources that are part of the Fogus Network App can be seen.

2.3.2.5 CAPIF & NEF API Certification

CAPIF and NEF are tools have been explained in detail in deliverables [1] and [13] therefore, this section is devoted to explaining the tests implemented making use of NEF and CAPIF tools during the certification process.

These tests are based on calls that Network Apps make to NEF and CAPIF APIs, also, libraries, explained in detail in [13] are provided to developers facilitating their job and abstracting them on how NEF and CAPIF work so, using the libraries is as simple as invoking a method which will be in charge of doing the low-level work of understanding the communication behind NEF and CAPIF. These libraries offer support to both, Network Apps developers and certification authorities by allowing them to build tests based on the calls that the Network Apps make to a real 5G system. Basically, the certification tests are mean to certify communication between the Network App, NEF and CAPIF APIs deployed in a real 5G system by means of checking that the API calls from

the Network App are not causing any problem, as well as that the call-backs from the real 5G system are handle properly in the Network App side

2.3.2.6 Network App offboarding

Once all these steps are completed, the certification process will continue with the decommission of the Network App. First, the Network App will be deregistered from the CAPIF, and finally will be undeployed from the platform. In this way, it is certified that the Network App can deregister from CAPIF and be correctly undeployed with these actions.

2.3.3 Open-repository Onboarding

If everything works properly during the certification of the Network App, the uploading to the Open-Repository will continue. This will be explained further in the document, in the Section 3.1, but a fingerprint will be needed to verify the validity of the image that has been uploaded to the Open-Repository. This fingerprint will be sent to the developer of the Network App through email (only the Network App creators will know the fingerprint value).

2.3.4 Certification Report

2.3.4.1 Report summary structure

In this section, the structure of the generated report during the certification will be described. First of all, the certification pipeline will produce a report detailing the status of each step, as well as an error message if any steps fail during execution. There will be a summary of the results obtained during all the steps of the analysis. In the Figure 25, it can be seen that for each step there are only two outcomes: failure and success. The status of each step will indicate whether or not the result of that step was successful. This summary is an effective way to reflect at a glance how the Network App performed and what specific parts of the certification need to be improved in order to be certified.

The individual result of the validations test has been as show in the following table:

STEP 0: STATIC ANALYSIS status: SUCCESS

STEP 1: SECURITY ANALYSIS status: FAILURE

STEP 2: SECRETS ANALYSIS status: SUCCESS

STEP 3: OPENSOURCE LICENSE status: SUCCESS

STEP 4: BUILD status: SUCCESS

STEP 5: SCAN DOCKER IMAGES status: SUCCESS

STEP 6: DEPLOY NETAPP status: SUCCESS

STEP 7: NETWORK NETAPP status: SUCCESS

STEP 8: ONBOARD NETAPP status: SUCCESS

STEP 9: DISCOVER APIS status: SUCCESS

STEP 10: DESTROY NETAPP status: SUCCESS

STEP 11: DESTROY NEF status: FAILURE

STEP 12: DESTROY CAPIF status: SUCCESS

Please, take a look to those steps that are failing.

Figure 25 Steps summary Certification Report

The next sections of the report include more thorough information on each of the phases. Each phase contains information on the Network App that was analysed, such as the branch, repository, or commit. In addition, in case of failure, detailed information

regarding the cause of the failure will be presented providing hints for the resolution of the problem.

The Source Code Security Analysis scan has found the following CRITICAL ISSUES:

Vulnerabilities

Severity	ID	Title	PkgName	InstalledVersion	FixedVersion
CRITICAL	CVE-2021-35042 (https://nvd.nist.gov/vuln/detail/CVE-2021-35042)	django: potential SQL injection via unsanitized QuerySet.order_by() input	Django	3.1.7	3.1.13, 3.2.5

The Source Code Security Analysis scan stage has failed because we have found Critical Issues with a possible solution. Please try to solve this error.

Information about high, medium, low and unknown issues can be found in the following link: https://github.com/EVOLVED-5G/FogusNetApp/wiki/Telefonica-Evolved5g-FogusNetApp

Figure 26 Report critical vulnerability found

In Figure 26 is possible to observe that a vulnerability was found in the Network App. The idea of this report is that the developer will be able to correct the security problem and continue with the certification by solving the issues shown in the analyses. Links to more detailed information about the different analysis are also provided. These analyses can be found in the GitHub repository of the Network App or in the official SonarQube instance of the Project.

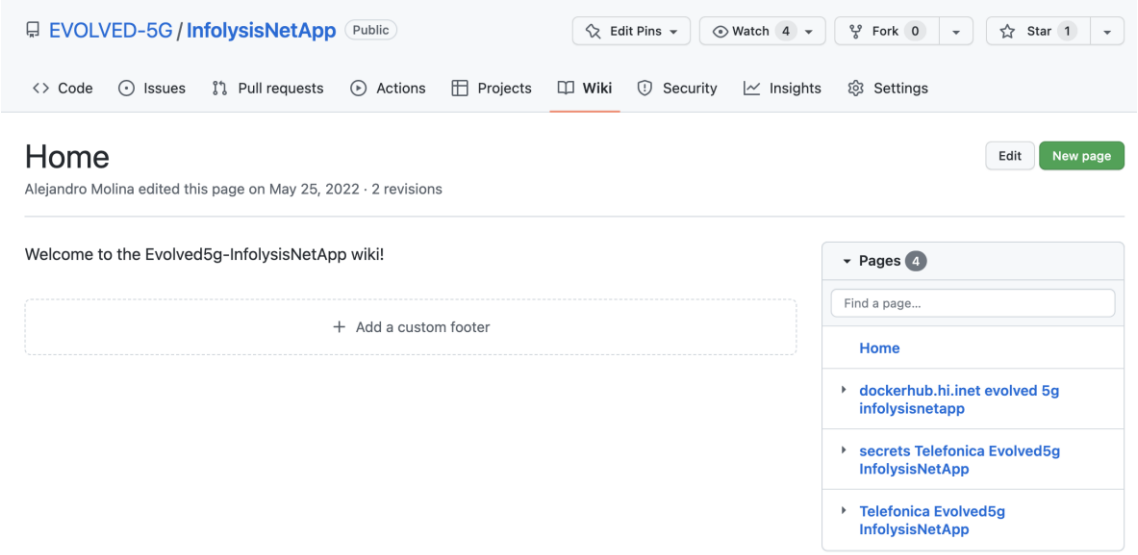


Figure 27 GitHub Wiki with all the security and quality static code analyses

2.3.4.2 Report Generation

The methods that have been used to generate and structure information into the various formats will be explained in this section. Ninja templates are used for this task because they are a fast, expressive, and extensible templating engine. This ninja tool allows us to create rich templates by inputting JSON files and receiving a file in Markdown (MD) format.

```
cicd > templates > scan-steps.md.j2
3   ***
4
5   This Certification Report contains the results of the Certification
6   Process executed over the Netapp {{version}}
7
8   The repo used for Certification has been: {{repo}}
9   The branch used for Certification has been: {{branch}}
10  The last commit ID has been: {{commit}}
11
12  The result of the Certification Process over the Netapp {{version}} has been:
13
14  The individual result of the Certifications test has been as show in the following table:
15
16  {% set vars = {'foo': False} %}
17  {% set counter = 0 -%}
18  {% for group in key.0 -%}
19    STEP {{ counter + loop.index0 }}: {{ group|upper|replace("-"," ") }} status: {{key.0[group]}}
20      {%- if key.0[group] == "FAILURE" %}
21        {% if vars.update({'foo': True}) %}
22        {% endif %}
23      {%- endif %}
24    {% set counter = counter + group|length %}
25  {% endfor -%}
26  {% if vars.foo %}
27    **Please, take a look to those steps that are failing.**
28  {% endif %}
29
30
31  In the following pages, we provide details of the tests executed
32  and the results.
```

Figure 28 Jinja file example

In Figure 28, it is possible to see an example of a ninja template. Jinja is a powerful templating engine for Python. It allows developers to define templates for their applications, which can include variables, logic, and other constructs that are replaced with actual values when the template is rendered. The syntax of Jinja templates is similar to that of Python, making it easy for developers who are already familiar with the language to learn. Additionally, Jinja includes a number of built-in filters and functions that can be used to manipulate data within templates. This file is used to generate the summary report; we can see how programmability can be included in the file. Figure 25 shows the outcome of running this template in Python.

The MD format is useful because it can be directly integrated with GitHub, making the information more accessible to the developer. Errors obtained during certification can be uploaded in their private repository without installing additional software or checking multiple sources.

Later, pdf generator will be used to convert the information in MD format into PDF format after receiving the MD format from Ninja. In Figure 29, the command used for generating the PDF can be seen:

```
pdf_generator markdown-pdf -f A4 -b 1cm -s style.css -i <NAME>.md -o <NAME>.pdf
```

Figure 29 Command for transforming MD into PDF

A PDF file is a readable format because it is designed to maintain the formatting and layout of a document regardless of the device or software used to open it. PDFs support various features such as hyperlinks, bookmarks, and annotations that make it easy to navigate and interact with the document.

3 MARKETPLACE AND OPEN REPOSITORY INTEGRATION

3.1 OPEN REPOSITORY AUTHENTICATION

During the Network App onboarding Network App creators are required to enter their Network App metadata in order to publicly release their Network App to the marketplace's Product Catalog. During this process, the Marketplace's component, namely "Certification report genuineness check" interacts with the Open Repository to make sure that the Network App has gone through the process of validation and certification and, that it has been published in the Open repository, thus making it ready to be consumed and published to the marketplace. The communication between the Marketplace and the Open Repository occurs under a private network. The Open Repository exposes an appropriately private http/https endpoint that allows the marketplace to retrieve related information about a specific Network App. The endpoint has the following form http://artifactory.hi.inet/artifactory/{parameter1}/certification/{parameter2} where parameter1 is the name of the Network App and parameter2 is the version of the Network App. Using this endpoint, the Marketplace is able to retrieve a file in JSON format that contains a code (from now on to be referred to as "fingerprint") that is related to that specific Network App.
Only the Network App creators know this fingerprint code. The Network App creators receive the fingerprint code via an email, while deploying the Network App to the Open Repository.
The Marketplace requires the user to paste this fingerprint code in the user interface and it automatically validates its existence by interacting with the Open Repository in the background.
The process is described in the following chapter.

3.2 METHODS IMPLEMENTATION - MARKETPLACE ONBOARDING & FINGERPRINT CODE

During Network App onboarding the Marketplace User Interface (UI) requires the Network App creator to enter metadata. In order to verify a Network App in the Open Repository, the Marketplace needs to receive:

a) The Network App name: The Network App name is distinguished and extracted from the GitHub URL entered by the Network App creator. For example, if the URL is https://github.com/EVOLVED-5G/dummy-netapp , the Network App name is extracted as "dummy-netapp".
b) The version of the Network App (Ex. 1.0)
c) The fingerprint code

Marketplace makes a call to the Open Repository in order to receive the existing fingerprint file for the given Network App. In our example, the URL will look like this: http://artifactory.hi.inet/artifactory/dummy-netapp/certification/1.0/fingerprint.json
If the fingerprint.json file is found and the fingerprint code in the file is the same as the fingerprint code entered by the user, then the Netqork App is considered as verified with the Open Repository. This process is depicted in the following diagram:

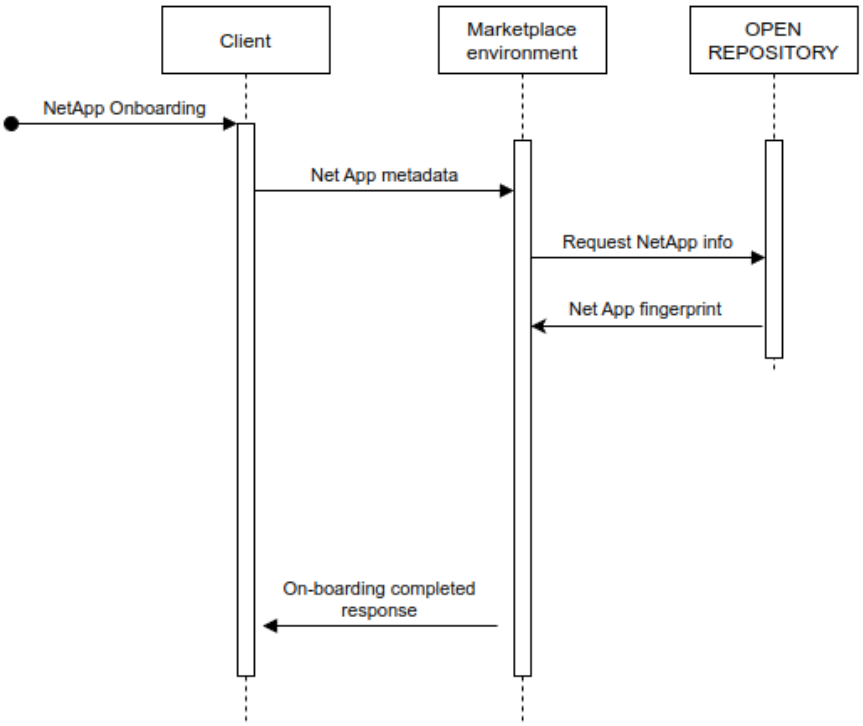


Figure 30 Verification OpenRepository - Marketplace

If the fingerprint.json file is not found, or if the fingerprint code is different than the one in fingerprint.json, then the platform informs the user accordingly, with an error message.

4 RELEASE TO MARKETPLACE

4.1 NETWORK APP RELEASE TO MARKETPLACE

The EVOLVED-5G Marketplace (https://marketplace.evolved-5g.eu/) is the main interaction point between Network App creators and Network App consumers.

It targets 3 different user profiles:

1) The Network App creators: Developers publishing their Network Apps to a public catalog.
2) The Network App consumers: Users purchasing and using the Network Apps.
3) The marketplace administrators: A group of administrators that have elevated access to view the platform's overall KPIs.

In this chapter focus is made on Network App creators as it describes step by step how the "Network App Onboarding" works.

The "Network App Onboarding" is the process of releasing an existing Network App to the marketplace. It is assumed that the Network App creator has already published the Network App to the Open Repository and has received a fingerprint code as described in the previous chapters.
Network App Onboarding is implemented as wizard and hence it contains a series of steps the Network App creators must follow in the user interface. It is available at the URL https://marketplace.evolved-5g.eu/create-netapp. Each of the steps are described below:

Step 1 | Service basic information/metadata

During this step, the user is prompted to enter Network App's basic details as well as some metadata. These details are:

1.Network App name
2.Network App about text
3.Type of the Network App (standalone or non-standalone)
4. Category of the Network App (currently, the categories offered are: Artificial Intelligence, Cyber security & cryptography, Identity and verification, Messaging services, Mobile carrier lending and advances, Mobile carrier subscriptions, Other
5.Version of the Network App.
6. Network App tag list (a list of tags that will help the users search for Network Apps with a corresponding tag)
7. URL slug. This describes the user-friendly URL via which the Network App will be accessed by. For example, in the following URL: https://marketplace.evolved-5g.eu/netapp-details/test-net-app , the slug part is "test-net-app"
8. Network App "view more" URL (a URL that the user will be able to visit in order to read more about the Network App, ex. a company page)
9.Network App logo image file
10.Network App publisher (either user or a Business/Organization).
In the case of Business/Organization, the user is then prompted to also enter the Business/Organization name and Social Number. If the Network App creator decides to create a Network App as a Business entity and not an individual

publisher, this will result as the Business being shown as the Network App publisher. These fields are also described in the following screenshot:

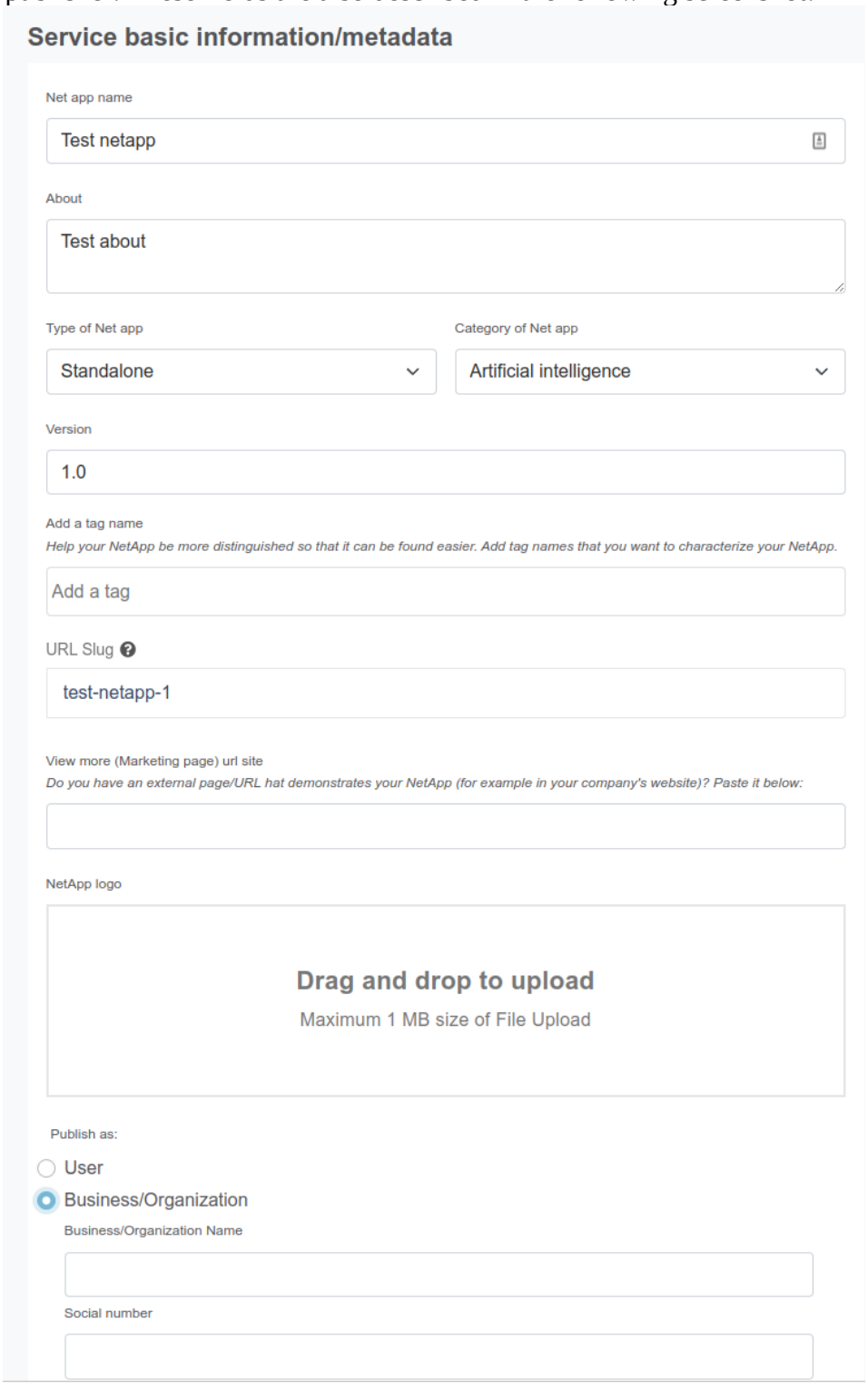


Figure 31 Service Basic Information Screen

After the user fills the fields and clicks on "Next", in order to proceed to the next step of the Onboarding Wizard, the app performs a first-level validation, ensuring that all the required fields have the correct input. For example, the URL slug should be unique in the platform.

Step 2 | Marketplace Policy
During the second step, the user sees a minimal form that asks them to ensure that they comply with the Marketplace policy. This is done by clicking the relevant checkbox:

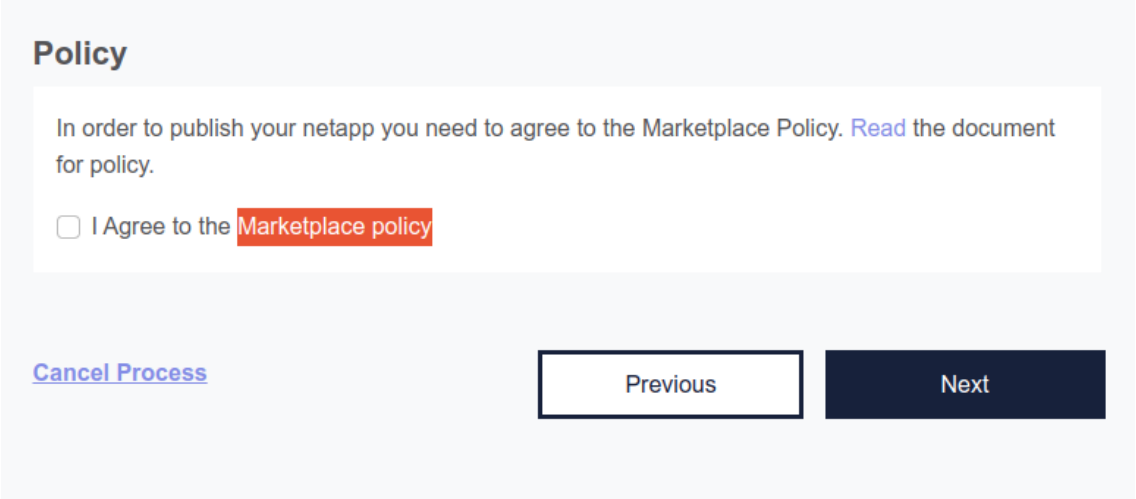


Figure 32 Marketplace policy

The checkbox should be filled in order to continue to the third step.

Step 3 | Deployment
In this step, the user needs to enter the following information:
1.The GitHub URL of the Network App

2.The Fingerprint Code

3.Upload the License file (optional)

After the user enters the GitHub URL and the Fingerprint code and clicks on "Next", the app performs the following steps in the background:

1.Checks whether the GitHub URL that was entered exists or not
2.With the fingerprint code and the Network App version (given in Step 1), the app checks whether the fingerprint code exists for this Network App in the Open Repository, as described in the previous chapter.

If the Network App is correctly found and confirmed with the Open Repository, then the related Open Repository Docker image URL is automatically generated and presented in the screen.

For example: https://dockerhub.hi.inet/evolved5g/certification/NetApp-1/images:1.0

This would be the docker image URL that will be presented to users that would like to use the Network App (Network App consumers).

Step 3 can be summarized in the following screenshot:

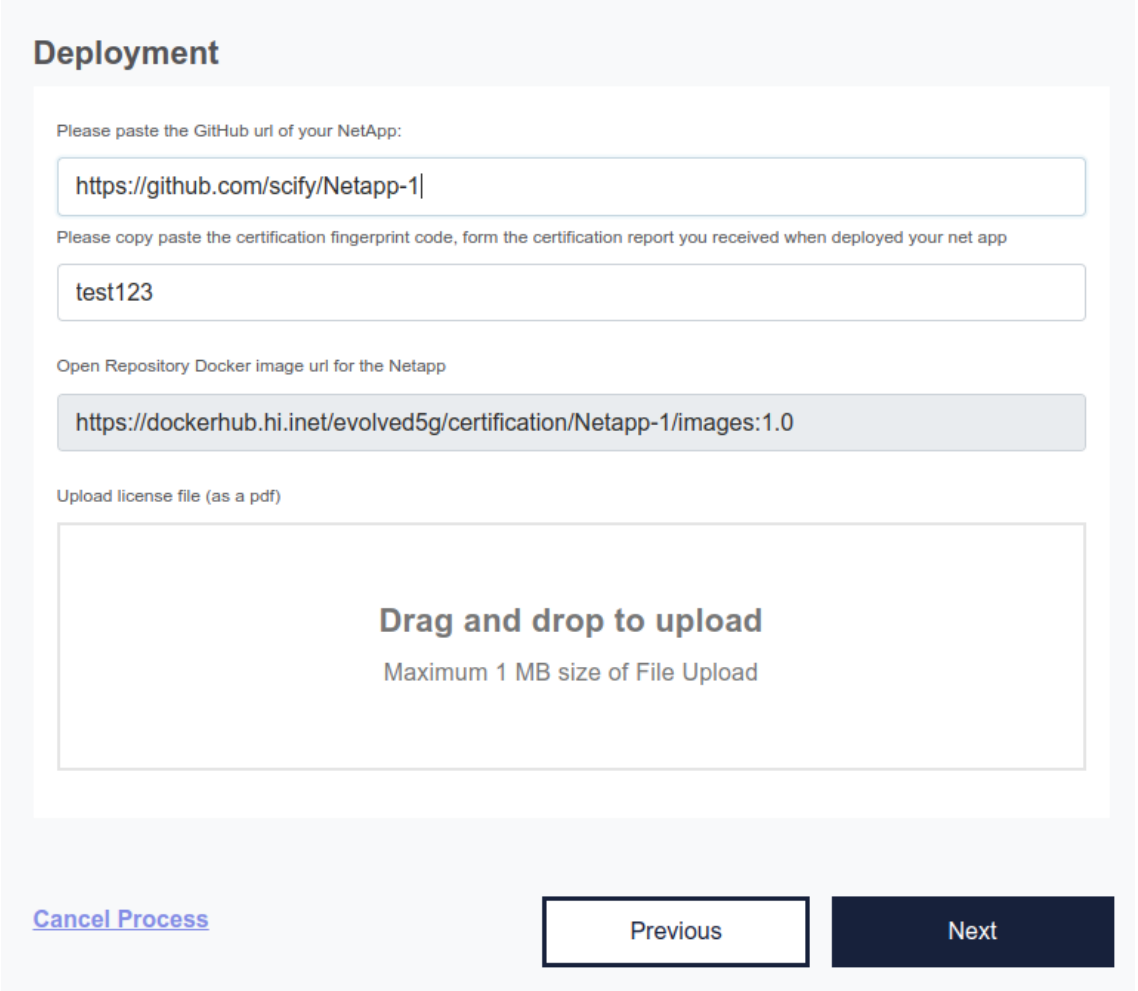


Figure 33 Deployment of the Network App in the Marketplace

Step 4 | Tutorial

In the next step, the user is prompted to enter the details for the tutorial of the Network App. This can be done in two ways, by entering the text in a text area or by uploading a file in PDF format for the tutorial (optional). The essence of the tutorial is very important. The Network App creator needs to describe exactly how the Network App and its corresponding services are deployed, used, and utilized. So, the tutorial is actually what the user who will use the Network App will need to read and understand, in order to use the Network App.
For example, the tutorial might include some practical examples with the required

commands that the user needs to follow in order to use the Network App Docker image. The fourth step looks like the following screenshot

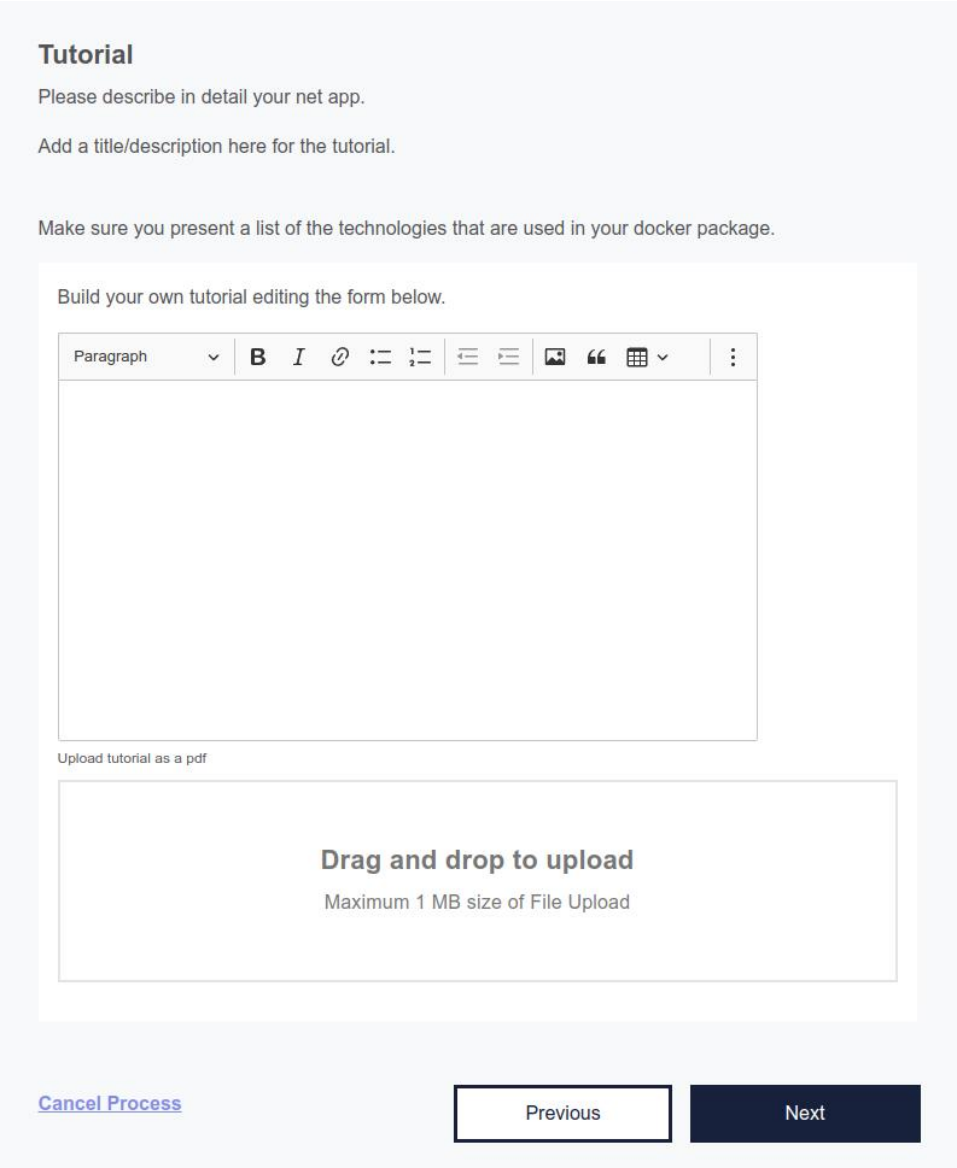


Figure 34 Tutorial Marketplace screen

Step 5 | Pricing Wizard

This is the last step of the Network App creation process. In this step, the user is prompted to enter the pricing details for their Network App. It is worth mentioning that during the EVOLVED-5G project no payments will take place in the platform. All of the Network Apps will have zero cost. Though the purpose of the pricing wizard is to demonstrate the available options a Network App creator could have under a production setting, where the Network App consumer is charged in order to use the Network App. Using Price wizard, Network App creators can help Network App consumers understand how they will be charged if they choose to use the Network App.

Marketplace supports two pricing modes: The pricing can be either “Once off”, meaning that the buyer will have to pay a one-time fee, or “Pay as you go”, meaning that the buyer

will have to periodically pay a variable amount of money depending on the use they make.

If the Network App creator selects “Once off”, they only need to enter the amount in euros as depicted below:

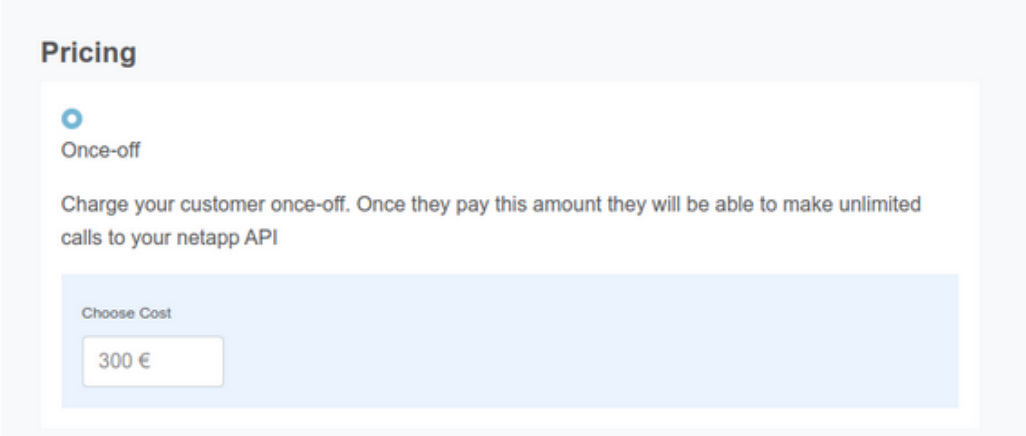


Figure 35 Pricing wizard marketplace

If the Network App creator selects “Pay as you go pricing”, they then have to describe the charge with either a) a fixed price for a specific number of calls, or b) fixed price per call. The Network App creator can then add specific endpoints of the Network App service and describe a payment scheme for the API calls per endpoint. Let’s consider a scenario where Network App can be used in order to track devices in a 5G network. The Network App exposes an API that allows Network App consumers to request for the coordinates of a specific device by making a call to an endpoint “/get-device-coordinates”. The Network App creators want to charge based on the number of requests that are made in this endpoint. Using the Price Wizard, they can make the following configuration:

“The first 500 requests to endpoint /get-device-coordinates is free. All subsequent requests have a fixed cost of 0.001€ per call”

The following picture illustrates this scenario:

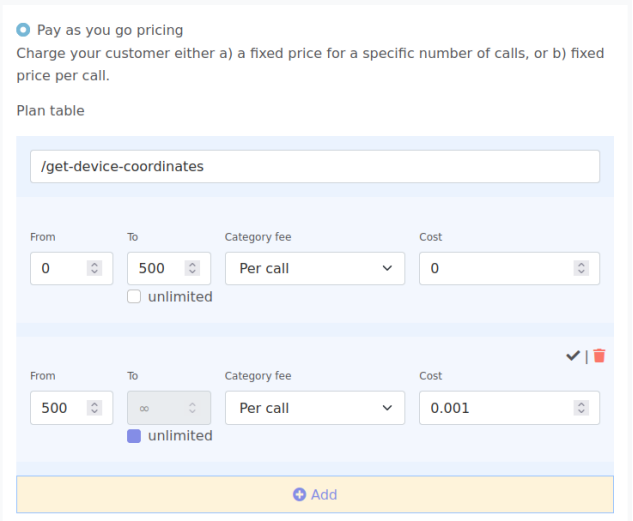


Figure 36 Network App price example

Final step | Release to the Marketplace

Once all of the above-mentioned steps are completed, the Network App is Onboarded to the marketplace, but it is still not publicly available to the Product Catalog.
Via the Edit Network App page a Network App creator can the Network App status from “private” to “public”, in order to make the Network App available to the Product catalog.

4.2 Purchase of a Network App in the Marketplace

The Product Catalog page consists of a list of published Network Apps which have been verified with the Open Repository and are available for purchase. For each Network App listed, the user can click and view all the relevant details, understand what the Network App does and which problem it solved, as well as review the pricing information:

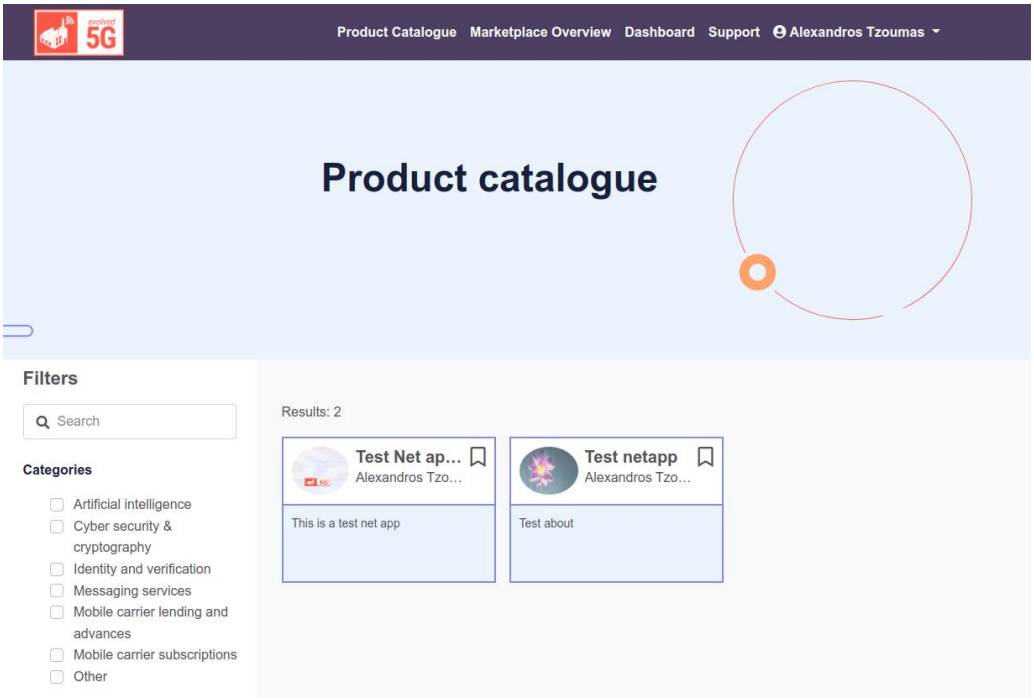


Figure 37 Marketplace Product Catalogue

In this page, registered users can search for Network Apps by using the filters on the left-side menu.

When the user clicks on one of the Network Apps, they are taken to the Network App public page, where they can read more about the Network App, as well as purchase it:

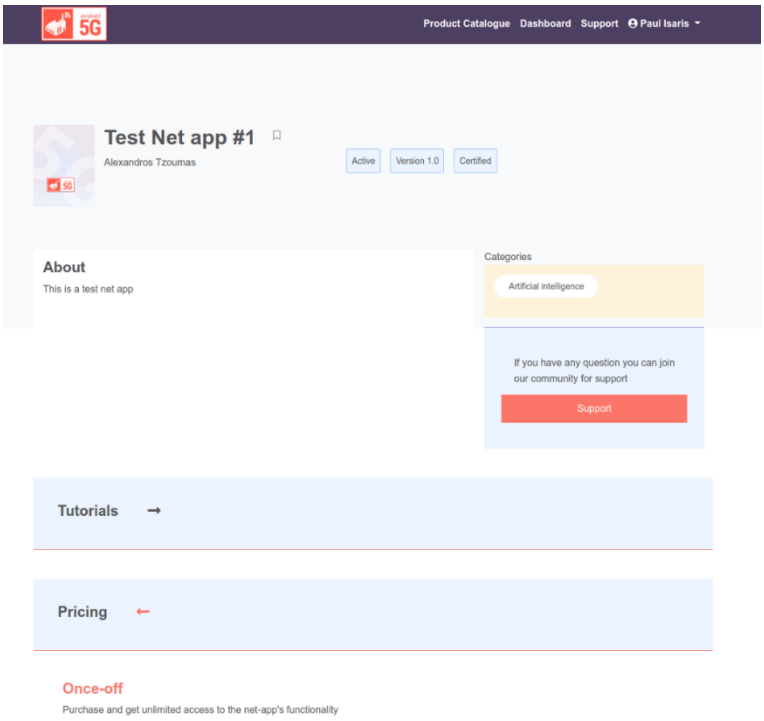


Figure 38 Purchase final screen

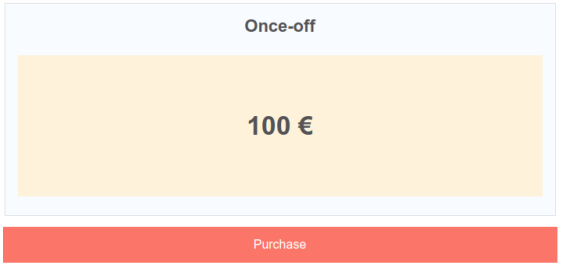


Figure 39 Network App purchase final screen

If the user clicks “Purchase”, the purchase is then completed in the background, and the user sees a confirmation message:

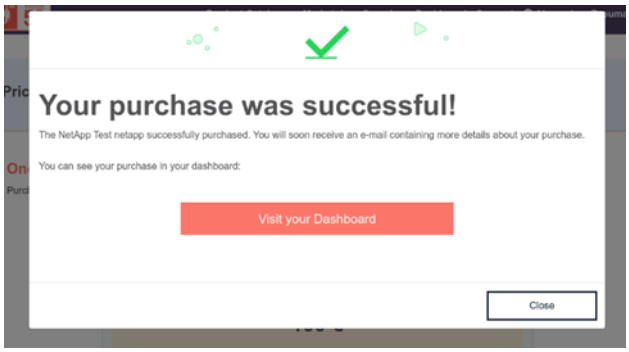


Figure 40 Purchase succeeded

Now the user has completed the Network App purchase. They will receive an automated email confirming the purchase:

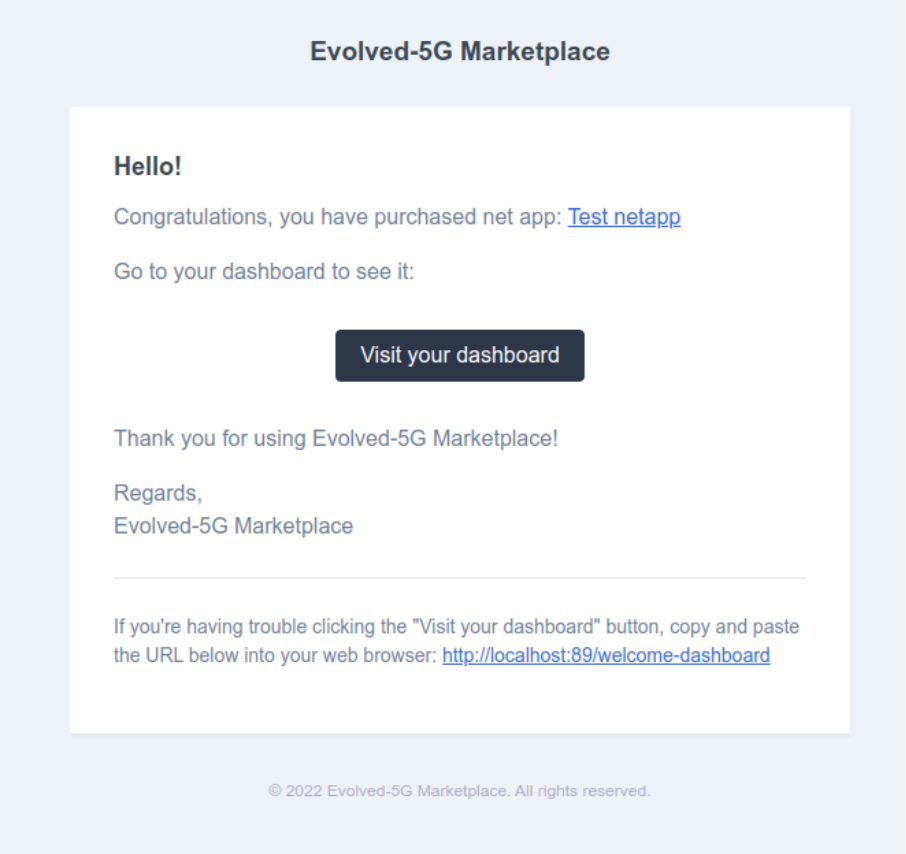


Figure 41 Dashboard Marketplace

Then, in the background, the app connects to the Ethereum Network, in order to log a digital signature of this purchase to the Blockchain Network. When the Blockchain transaction is completed, the user receives another automated email, notifying them about the transaction:

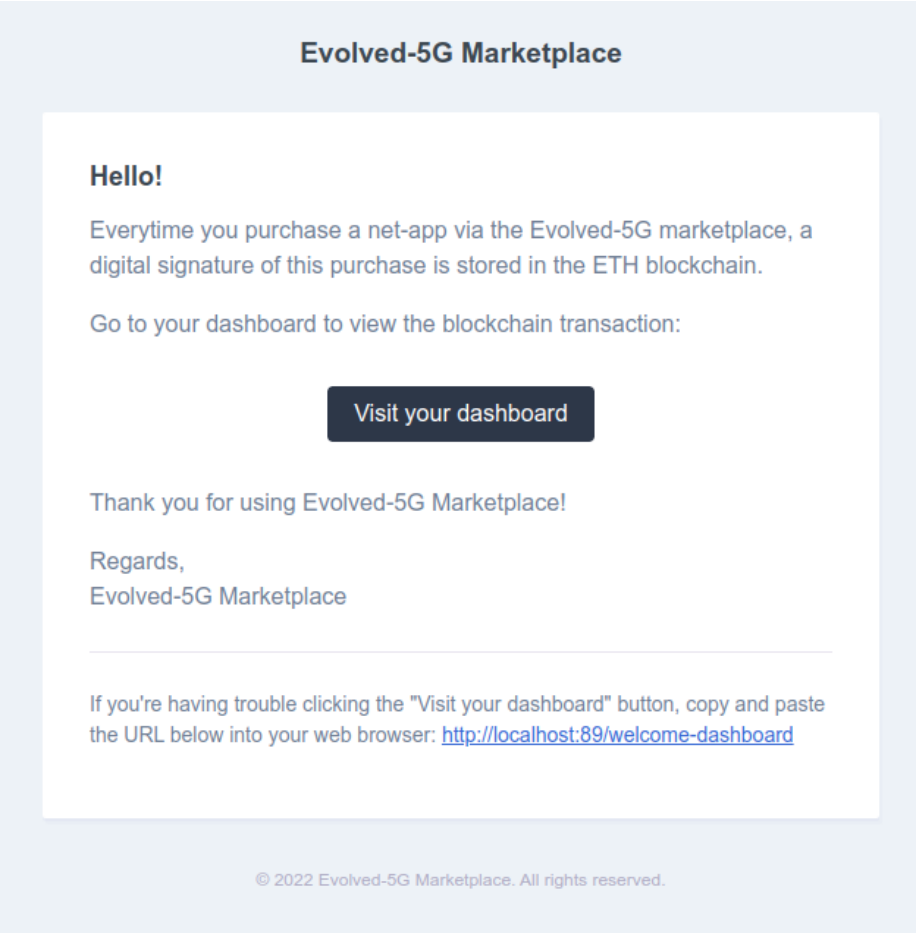


Figure 42 Purchase confirmation

This hash string is then shown as a **Digital Signature** in the "My purchased Network Apps" page, which is accessible via the "Dashboard" page:

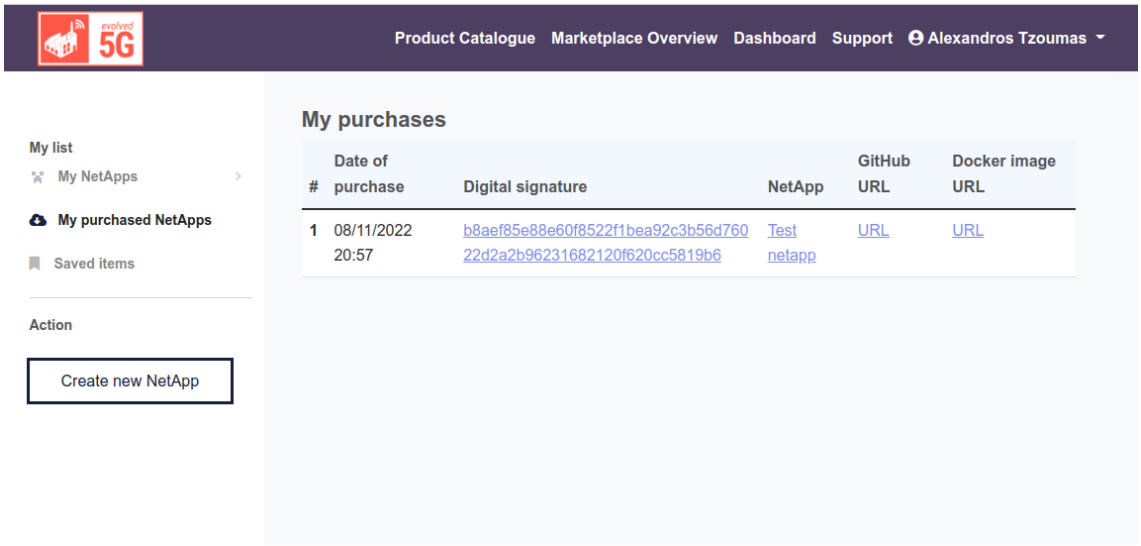


Figure 43 Marketplace Digital Signature

When the user clicks on the Digital Signature link, they are taken to an Etherscan page, where they can view the corresponding Blockchain transaction:

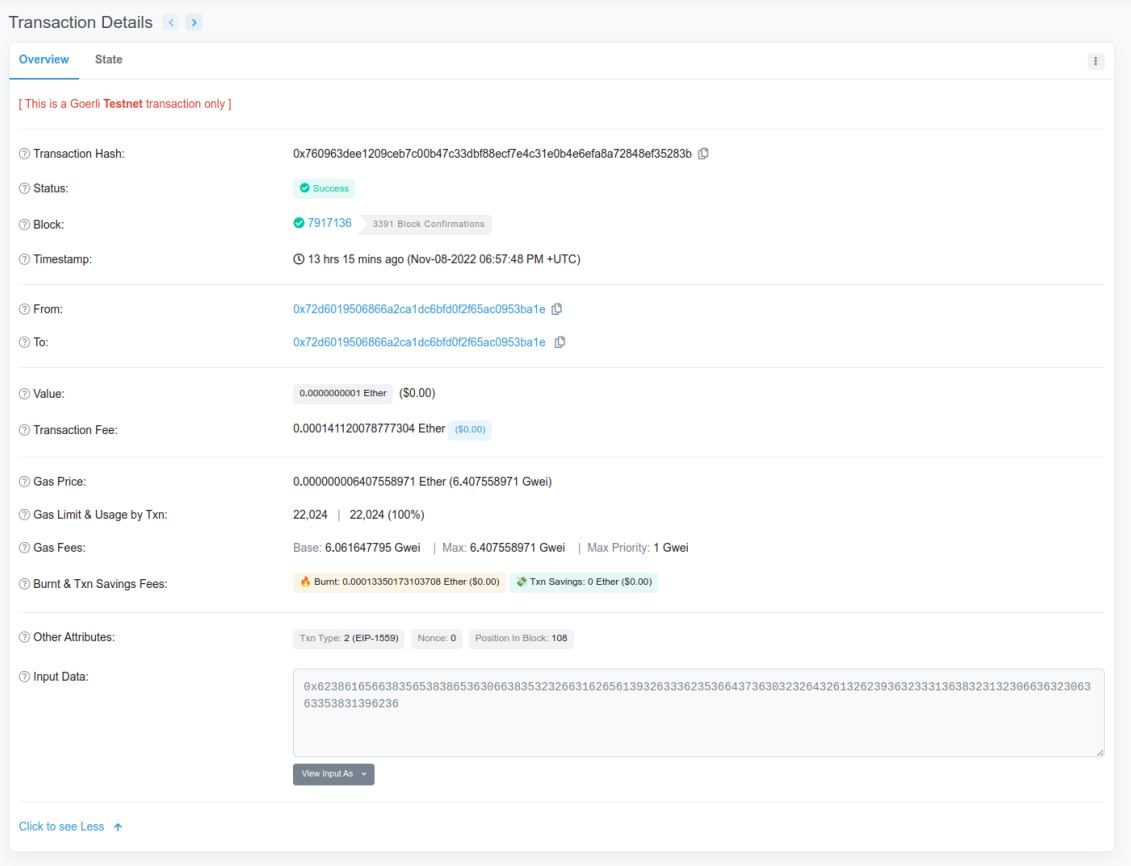


Figure 44 Marketplace Blockchain details

Finally, the same page, the user can also copy paste the Docker Image URL in order to start using the Network App.

4.3 Network App Versioning

Network App creators specify the Network App version during the deployment phase of the Network App. Once the Network App is deployed, the Network App creator has to copy and paste this version number during the Onboarding Process of the Network App.

The Onboarding Process in the Marketplace is always tightened to one specific version. So, if Network App creators want to support multiple versions of the same Network App, they will have to create separate entries in the Product Catalog. Each version will have a separate single page in the Marketplace and the differences between the supported versions should be explained in the tutorials.

4.4 Removing a Network App from the Marketplace

A Network App can take two statuses in the Marketplace:

a) Public (available for purchase and view by all platform users)
b) Private (available and shown only to the Network App creator).

When a Network App creator decides not to list their Network App anymore, they can select to make it “Private”, by editing the Network App status. When set to private, the Network App is not removed or deleted from the Marketplace’s database, but it is restricted to be shown only to its creator and is not available in the Marketplace.

This is useful in scenarios where multiple versions have been uploaded to the marketplace, but only the latest is visible to the product catalog. At the same time Network App consumers will have access to any older version they may have purchased.

5 CONCLUSION AND NEXT STEPS

This deliverable presented the work performed in the context of WP5, and more specifically, as part of task T5.3 from M14 to M25.

T5.3 is the task where the integration of the Marketplace and the development of Certification process takes place. Indeed, Network Apps will have to go through the Certification process in order to be certified and later uploaded to the Marketplace. A detailed description of this Certification process and status of the Marketplace development is provided in this deliverable as well as information about the current stage of its implementation.

Regarding next steps, it must be taken into account that some additional tools need to be tested in order to be integrated into the CI/CD as part of T5.3. This is an ongoing work under evaluation by the time this deliverable is being submitted and will be part of the work on the Certification environment. Besides, ongoing activities regarding the evolution of the platforms also continue, with the integration of existing components and the improvement of existing functionality based on the initial tests of running the Certification process over the Network Apps. All this work will be documented in deliverable D5.6 to be submitted in M36.

6 REFERENCES

[1] EVOLVED-5G, Deliverable 3.2 “NetApp Certification Tools and Marketplace development (intermediate) ”
[2] Trivy project, from https://aquasecurity.github.io/trivy/v0.28.1/
[3] Robo Framework, from https://robotframework.org/robotframework/
[4] lyon, g., 2005. Chapter 15. Nmap Reference Guide. [online] Nmap.org. Available at: https://nmap.org/book/man.html
[5] Sonarqube, from https://www.sonarqube.org/
[6] NMAP, from https://nmap.org/docs.html
[7] Debriked project, from https://debricked.com/tools/license-compliance/
[8] EVOLVED-5G, "Deliverable D4.1 “EVOLVED-5G Factory of the Future (FoF) NetApps," [Online]. https://evolved-5g.eu/wp-content/uploads/2022/09/EVOLVED-5G-D4.2_v_1.0.pdf
[9] EVOLVED-5G, Deliverable 5.2 “Deliverable D5.2NetApps Validation and onboarding to Open Repository (intermediate)”
[10] Deliverable 5.1 “System level evaluation and KPI analysis(Intermediate)” Accelerator, from https://evolved-5g.eu/community-accelerator/
[11] EVOLVED-5G, "D2.2 Design of NetApps development and evaluation environments"
[12] NMAP, from https://nmap.org/docs.html
[13] EVOLVED-5G, "D3.1 Implementations and integrations towards EVOLVED-5G framework realisation (intermediate)" https://evolved-5g.eu/wp-content/uploads/2022/01/EVOLVED-5G-D3.1-v1.0.pdf.