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

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

Innovations for verticals with third party services

D2.2 Cloud-native infrastracture

Delivery date: April 2023

Dissemination level: Public

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

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

Document Information

Deliverable	D2.2: Cloud-native infrastructure
Work Package	WP2: Cloud-native 5G NFV
Task(s)	Task 2.1: Cloud-native infrastructure
Type	Other
Dissemination Level	Public
Due Date	M28, April 30, 2023
Submission Date	M28, April 27, 2023
Document Lead	Apostolos Siokis (IQU)
Contributors	Kostas Ramantas (IQU) Carlos Martins Marques (ALB) Manuel Requena (CTTC) Hamzeh Khalili CTTC) Josep Mangues (CTTC) Fatemehsadat Tabatabaeimeher (CTTC) Ankur Gupta (HHI) Kirsten Krüger (HHI) Holger Gäbler (HHI) Nicola di Pietro (ATH) Jorge Marquez Ortega (UMA) Almudena Diaz (UMA)
Internal Review	Jorge Carapinha (ALB) Bruno Garcia (UMA)

Document history

Version	Date	Changes	Contributor(s)
V0.1	05/12/2022	Initial deliverable structure	Apostolos Siokis (IQU)
V0.2	19/12/2022	Section 2	Apostolos Siokis (IQU) Kostas Ramantas (IQU)
V0.3	23/02/2023	Section 3.1	Jorge Marquez Ortega (UMA) Almudena Diaz (UMA) Nicola di Pietro (ATH)
V0.4	02/03/2023	Section 3.2	Carlos Martins Marques (ALB)
V0.5	03/03/2023	Sections 3.3, 3.4	Manuel Requena (CTTC) Ankur Gupta (HHI)
V1.0	28/03/2023	Integration of the material, Introduction, Conclusions, Executive Summary. Version preparation for internal review.	Apostolos Siokis (IQU)
V1.1	17/04/2023	Internal Review	Jorge Carapinha (ALB) Bruno Garcia (UMA)
V1.5	24/04/2023	Version with suggested revisions	Apostolos Siokis (IQU)
V1.6	26/04/2023	Quality review: copyediting; proofreading; formatting	Stefania Stamou (FORTH) Konstantinos C. Apostolakis (FORTH)
V2.0	27/04/2023	Final version for submission	Apostolos Siokis (IQU)

Project Partners

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

List of abbreviations

Abbreviation	Definition
3GPP	3rd Generation Partnership Project
5G-PPP	5G Infrastructure Public Private Partnership
AMF	Access and Mobility Management Function
API	Application Programming Interface
CNI	Container Network Interface
CNF	Cloud-Native Network Functions
CPU	Central Processing Unit
CU	Centralised Unit
DNS	Domain Name System
DU	Distributed Unit
EU	European Union
GA	Grant Agreement
gNB	gNodeB
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
IT	Information Technology
JSON	JavaScript Object Notation
K8s	Kubernetes
MANO	Management And Orchestration
MEC	Multi-Access Edge Computing
NFV	Network Function Virtualisation

NFS	Network File System
NR	New Radio
NSA	Non-Stand-Alone
OS	Operating System
OSM	Open Source MANO
PLMN	Public Land Mobile Network
PNF	Physical Network Functions
QoS	Quality of Service
R&D	Research and Development
(O-)RAN	(Open) Radio Access Network
OSM	Open-Source MANO
PLMN	Public Land Mobile Networks
RBAC	Role-Based Access Control
RESTful	Representational State Transfer
RRH	Remote Radio Head
RU	Radio hardware Unit
SA	Stand-Alone
SaS	Services as networkS
SMF	Session Management Function
TI	Testbed Instance
UC	Use Case
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function

URLLC	Ultra-Reliable Low Latency Communications
USRP	Universal Software Radio Peripheral
VE	Virtual Environment
VIM	Virtualised Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF(M)	Virtual Network Function (Manager)
VPN	Virtual Private Network
VR	Virtual Reality
YAML	YAML Ain't Markup Language

Executive summary

This document presents the 5G-EPICENTRE D2.2 “Cloud-native infrastracture”, and corresponds to Task T2.1 “Cloud-native Infrastructure”.

This deliverable stands as a public report, providing details regarding the cloud-native transformation of the infrastracture for each testbed used in the project. More specifically, details are provided about the approach followed by each testbed in order to cloud nativise their core and to support Kubernetes (orchestration, Openstack, bare metal approach, *etc.*), as well as the infrastructure software and hardware components. Information about the 5G network (as well as the used monitoring system) is also included.

A special emphasis is given to the presentation of guidelines for access to the Kubernetes platforms: the preliminary steps required to perform a deployment on the testbeds are presented separately for each testbed used in the project.

Table of Contents

List of Figures

List of Tables

1 Introduction

5G-EPICENTRE represents an attempt to federate 5G testbed infrastructures across the European Union (EU) with the specific purpose to address the needs and demands of the public safety and emergency and disaster management market needs. An important step to this end is the cloud-native information of the infrastructure offering benefits, such as flexibility supporting the accommodation of changing network requirements and scalability. The support of Kubernetes (K8s) as an orchestrator is a necessary condition in order to enable the federation of the four testbeds in the project using Karmada (work carried out within T4.3 "Cross-testbed federation and synchronization").

This deliverable aims at presenting details regarding the approach each testbed follows to support K8s, as well as the infrastructure software and hardware components used, and details about the monitoring system utilised in each case. Since there is not a common approach followed by all four testbeds to cloud nativise their infrastructure, a separate section is dedicated for every testbed in the project where the aforementioned details are presented for every case.

1.1 Mapping of Project's Outputs

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

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

5G-EPICENTRE Task	Respective Document Chapters	Justification
Task 2.1: Cloud-native infrastructure *"The purpose of this Task is to apply a cloud-native approach to the 5G Core, by smoothly evolving the existing NFV Infrastructure (NFVI) of the 5G-EPICENTRE testbeds […] to K8s, the de-facto container orchestration system for automating application deployment, scaling and management.*	Section 3.1 – Málaga testbed (UMA)	Section 3 presents details for the cloud-native transformation of the infrastructure, and the support of K8s in the four testbeds (in four different subsections, one dedicated for every testbed).
	Section 3.2 – Aveiro testbed (ALB)	
	Section 3.3 – Barcelona testbed (CTTC)	
	Section 3.4 – Berlin testbed (HHI)	
Task 2.1: Cloud-native infrastructure *"[…]To accommodate for the migration of existing VM-based VNFs and workloads that are traditionally difficult to containerize, and thus cater to both monolithic VMs and containers' integration to the MANO, appropriate virtualization APIs, such as KubeVirt, will be utilized, so as to run existing VMs inside containers."*	Section 3 – Cloud-native transformation of the infrastructure in the project	Section 3 presents details for the cloud-native transformation of the infrastructure of the four testbeds.

1.2 Structure of the Document

The structure of this deliverable has been divided into multiple Sections, in order to present the different approaches followed by the testbeds in Task 2.1. In Section 1 we introduced the current deliverable (D2.2) and explained what to expect from its content. It mapped each Section to specific need requested from the GA, specifically from T2.1. Section 2 then briefly presents the benefits of cloud-native transformation in 5G networks. Section 3 is the main section of this deliverable. It is divided in four sub-sections, one dedicated for each testbed in the project (respective sections for UMA, ALB, CTTC, HHI testbeds) A sub-section is dedicated to every testbed briefly describing the steps that have to be taken when an experimenter wants to perform a deployment on the specific testbed. Section 4 concludes the deliverable.

2 Benefits of Cloud-native transformation in 5G networks

"Cloud-Native" is the name of an approach for designing, building and running applications that fully exploit the benefits of the cloud computing model [1] [2]. This means that the apps can be built and changed more quickly, are more agile and scalable, and are more easily connected with other apps. The cloud-native approach is the way applications are created and deployed, not where they are executed. New operational tools and services, like continuous integration, container engines and orchestrators are pillars of this transformation.

Physical Network Functions (PNF) and Virtual Network Functions (VNF) [1] will continue to be used for at least another decade. The most promising approach is to allow the evolution of PNFs and VNFs to become Cloud-Native Network Functions (CNFs). Moving network functionality from physical hardware to encapsulating the software in a Virtual Machine (VM) is generally easier than containerising the software. There has been a similar journey on the software stack deployed currently in telco eco-system and at 5G Infrastructure Public Private Partnership (5G-PPP level). Most of the prototypes and the project realisation moved from a pure OpenStack ecosystem, derived by ETSI MANO, to include the capability to run K8s on top of either bare metal or any cloud where the intelligence is still centralised in the VNF Manager (VNFM). In the future version, most functions will be CNFs, while for VNFs that cannot, or have not yet been ported to CNFs, technologies such as KubeVirt or Virtlet can be used.

The expected benefits of the cloud-nativisation in 5G networks are the following:

- **Speed:** Companies of all sizes now see strategic advantage in being able to move quickly, and get ideas to market fast.
- **Scalability:** 5G networks require a highly scalable infrastructure, and cloud-native technology provides a flexible and scalable platform to meet the demands of 5G services.
- **Automation:** Cloud-native systems allow for automatic provisioning and orchestration of network functions and services, reducing manual processes and increasing efficiency.
- **Virtualisation:** 5G networks require VNFs to support dynamic network slicing, and cloud-native technology provides a platform for efficient and cost-effective deployment of VNFs.
- **Flexibility:** Cloud-native systems provide a flexible infrastructure that can accommodate changing network requirements, enabling rapid innovation and deployment of new services and applications.
- **Cost-effectiveness:** Cloud-native technology reduces the capital expenditures and operational costs associated with traditional network infrastructures, making it an attractive option for 5G network operators.

Overall, the use of cloud-native technologies in the 5G Core can help improve the performance, reliability, and agility of the 5G network, and to support the development and deployment of new 5G services and applications. By enabling organisations to more easily and efficiently develop, deploy, and manage their 5G services, cloud-native technologies can play a key role in driving the adoption and growth of 5G networks and services.

3 Cloud-native transformation of the infrastructure in the project

This Section provides details about the approach each testbed follows in order to cloud nativise their infrastructure and to support K8s. Sub-sections 3.1, 3.2, 3.3 and 3.4 are dedicated to UMA, ALB, CTTC and HHI testbeds respectively. There is a dedicated sub-section in these four sub-sections providing K8s support guidelines, briefly describing the steps that have to be taken when an experimenter wants to perform a deployment on the specific testbed.

3.1 Málaga testbed (UMA)

The K8s deployment in the Malaga testbed is based on a multi-master deployment to provide high availability. The deployment is composed of 3 masters that manage 3 nodes, with an additional node dedicated to the storage as shown in Figure 1.

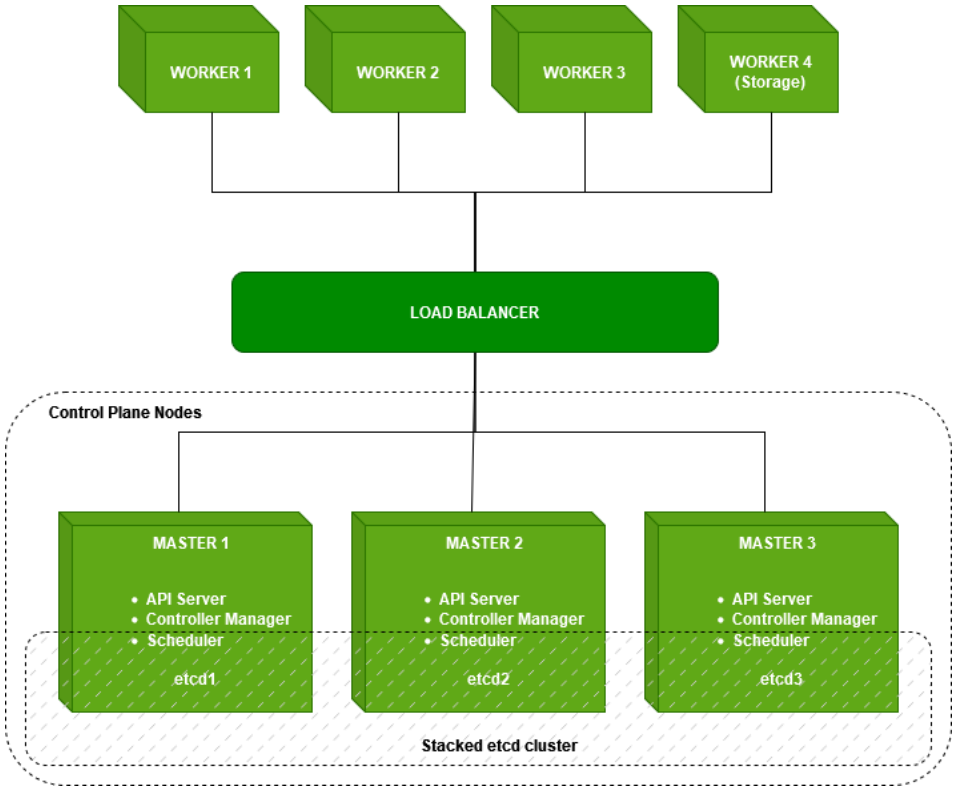

Figure 1: Representation of the K8s structure

Access to the nodes is implemented through a load balancer, to ensure the correct distribution of work among the nodes. Since the deployment is installed in a bare-metal structure, MetalLB has been used in conjunction with the NGINX Ingress Controller for the implementation of the network interfaces in the load balancer and the access control.

The management of the resources and deployments are done through kubectl, and the container runtime used is containerd. For security reasons, access to the functionalities is based on a role-based access scheme (RBAC), which provides isolation and prevents possible issues affecting each user's deployments. For the monitoring of measurements on the K8s infrastructure, Prometheus is used, together with a prometheus-node exporter, as well as an Alert Manager for the management of possible system alerts.

Finally, use has been made of the KuberVirt module to enable the deployment of VMs on top of the containerised infrastructure. The entire structure is virtualised using VMs created with Promox Virtual Environment (VE) and VMWare Vsphere hypervisors.

3.1.1 Infrastructure software components

This Section provides a more detailed explanation of the infrastructure software components.

Table 2: Infrastructure Software

Software	Description
MetalLB[1]	MetalLB is an open-source load balancer driver that allows external access to services in a K8s cluster running on a network insfrastructure that does not support Layer 2 load balancing. This tool is intended for the implementation of a load balancer in bare-metal systems.
NGINX-Ingress[2]	This is an ingress controller that runs as a deployment on the K8s cluster. It provides load balancing and custom traffic routes for services running on the cluster. This tool is intended to optimise traffic supported by the different services deployed in the cluster.
Kubectl[3]	This component provides command-line tools to interact with K8s clusters. It is used to manage K8s resource, such as pods, services and replicaset, and execute commands and tasks in the cluster. This tool enables cluster management and administration by users.
Calico[4]	Calico is an open-source project that provides a scalable networking and security solution for K8s clusters. It uses a flat, non-overlay network model to route traffic and provide network-level security for containers and applications. Thanks to this tool, it is possible to create Domain Name System (DNS) services and routing between the different pods deployed on the cluster nodes.
KubeVirt[5]	This is an open-source project that offers a VMs virtualisation option on K8s clusters.
GlusterFS[6] **+Heketi**[7]	GlusterFS is an open-source distributed file system used to store data on server clusters. Heketi is a tool that simplifies the administration of GlusterFS, allowing the creation and management of volumes in the K8s cluster. Thanks to Heketi access and management of volumes by system masters to designated storage nodes is possible.
Prometheus[8]	Prometheus is a highly scalable open-source monitoring framework. It provides out-of-the-box monitoring capabilities for the K8s container orchestration platform. Also, In the observability space, it is gaining huge popularity as it helps with metrics and alerts. Prometheus is used in conjunction with a node exporter to provide metrics on the different components of the system.

[1] https://metallb.universe.tf/
[2] https://docs.nginx.com/nginx-ingress-controller/
[3] https://kubernetes.io/docs/tasks/tools/
[4] https://docs.tigera.io/calico/latest/about/
[5] https://kubevirt.io/
[6] https://www.gluster.org/
[7] https://github.com/heketi/heketi
[8] https://prometheus.io/

3.1.2 Namespace management

Regarding the organisation of the namespaces, the monitoring tools have their own namespace called "monitoring", the rest of the components are scattered throughout the structure or contained in the "kube-system" namespace. Finally, each partner has its own namespace. In this way, there is an isolation between the functions of each partner and the cluster administration functions.

3.1.3 5G network

For the management of the 5G network, UMA has an instance of the 5G Core provided by Athonet (ATH)[9]. This 5G Core is software-based, and provides a network for both data and voice services.

The 5G Core software is fully virtualised running on VMs through an instance of the VMWare hypervisor.

3.1.4 Hardware approach

The hardware implementation of the platform consists of 3 physical servers, specifically two **Dell Poweredge R740** servers both virtualszed with Proxmox VE Hypervisor and a **Dell Poweregde R730** server virtualised with the VMWare VSphere ESXI hypervisor. The distribution of the structure between the different servers is shown in Figure 2.

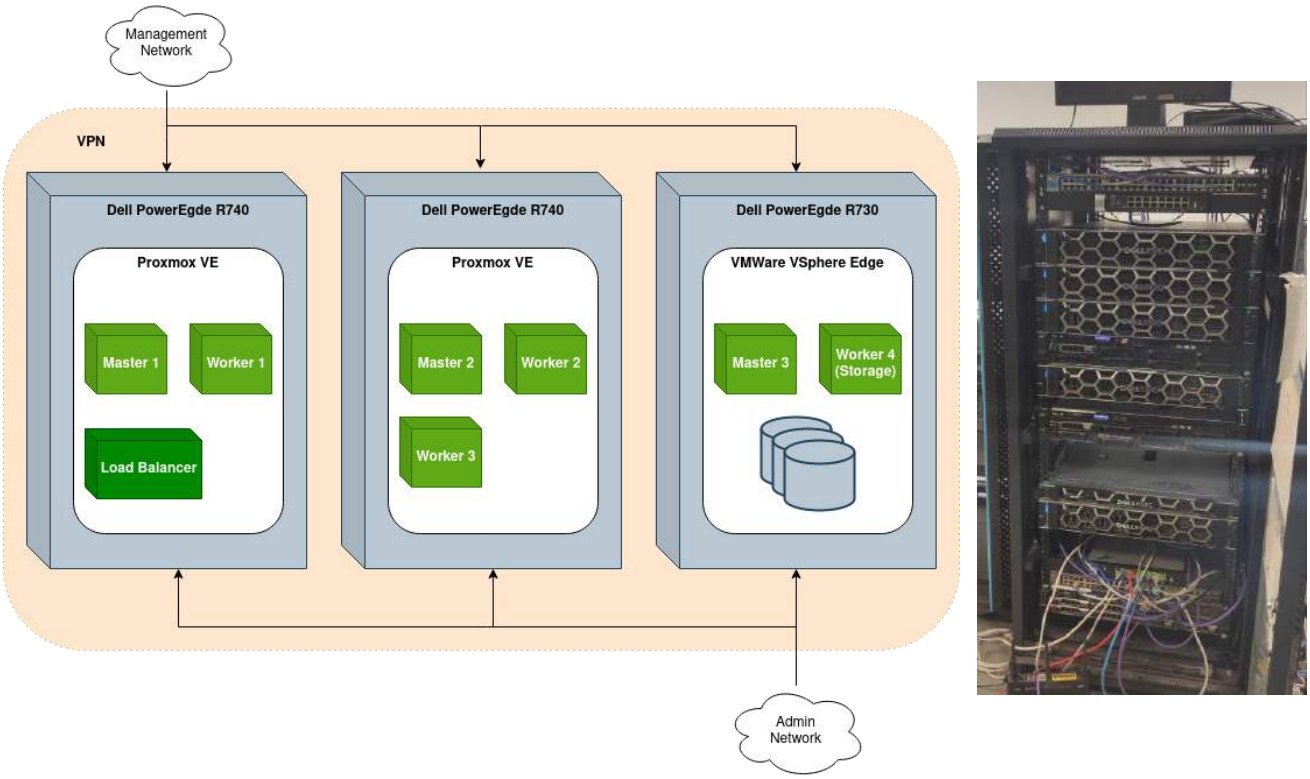

Figure 2: Arrangement of K8s elements in the infrastructure, and physical view of the 3 servers

For the management of the infrastructure, there is a management network for the administration of the hypervisors and a production network for the administration of the K8s framework they contain. The access to the infrastructure by the experimenters is done through a Virtual Private Network (VPN) implemented on the testbed.

[9] https://athonet.com/products/athonet-5g-core/

3.1.5 Example use case

A possible use case can be represented by a registered experimenter, who wants to deploy their application on the testbed.

To do so currently, they will apply their deployment using **kubectl**. The deployment will be handled by the Load-Balancer (**MetalLB**) for the correct distribution of resources. The communication between the elements of the deployment deployed along the different nodes is carried out by **Calico**. Finally, for the management of the traffic sent to the different elements of the deployment, **NGINX-Ingress** will be used. This behavior is illustrated in Figure 3.

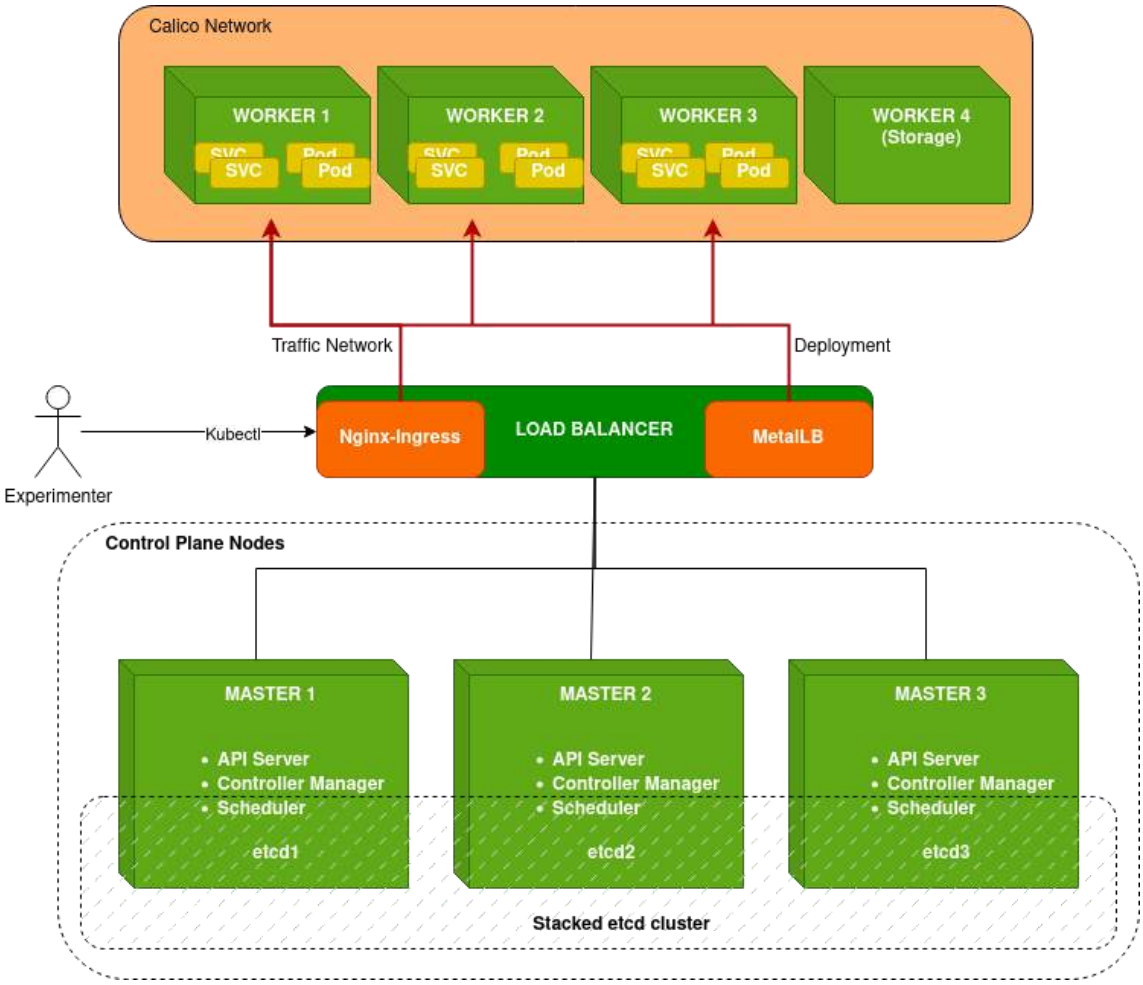

Figure 3: Deployment and network traffic use case

3.1.6 Dynamic persistent storage allocation

For dynamic allocation of storage volumes, **GlusterFS** is used in conjunction with **Heketi.** When a user wants a persistent storage volume, they request it through a persistent volume claim. This request is processed by the Heketi service installed on the master, which communicates with the GlusterFS server installed on the storage node. The storage node will allocate the persistent memory volume to the indicated deployment (Figure 4).

3.1.7 Kubernetes support guidelines

When an experimenter wants to perform a deployment on the cluster, a series of preliminary steps must be taken.

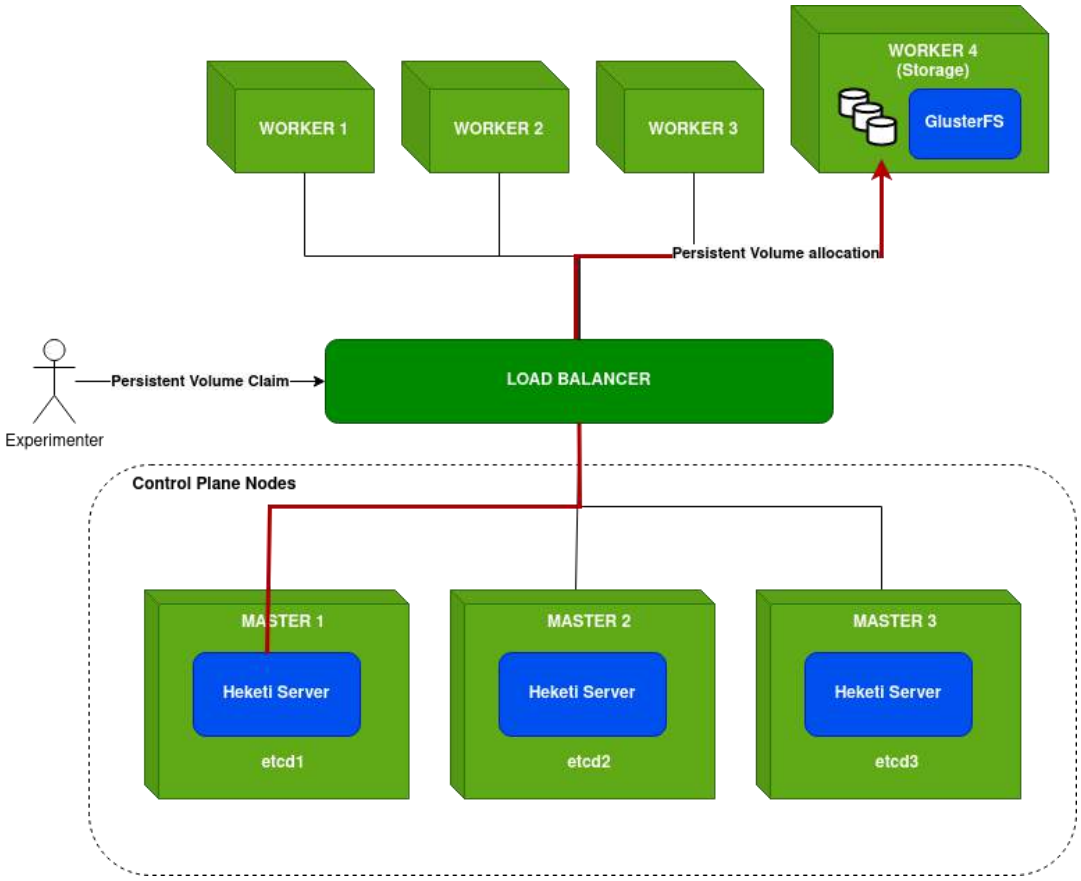

Figure 4: Dynamic memory allocation diagram

First, a preparation of configuration files is needed: to deploy an application on a K8s cluster, configuration files describing the different resources to be deployed must be created. These resources include pods, services, volumes and configurations.

Second, a deployment configuration file needs to be created: The deployment configuration file describes how the pods and associated resources are to be deployed in the cluster. This file should include information, such as the number of replicas to be deployed and the container image to be used, usually this file is expressed in a YAML format.

Thirdly, the application should be deployed to the cluster: once the configuration files have been prepared and the deployment configuration file has been created, the application can be deployed to the cluster using kubectl, the K8s command-line tool. Next, it is a good practice to check the deployment status. Once the application has been deployed, the status can be checked by using kubectl commands. For example, it can be checked if the pods have started correctly, and if the services are accessible.

If we want to update the application after the application is deployed, we can perform updates to fix bugs, or add new functionality. Updates are performed by creating a new version of the deployment configuration file, and updating the pods and other associated resources.

Overall, the deployment process in K8s is very flexible and scalable, allowing developers and system administrators to build and manage applications efficiently and effectively.

3.2 Aveiro testbed (ALB)

The Aveiro testbed includes a K8s-based infrastructure with a master and multi-node architecture, as shown in Figure 5. The cluster includes four worker nodes, two of them acting as the persistent storage server.

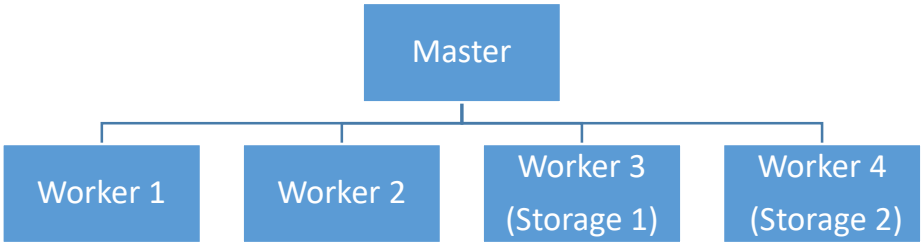

Figure 5: Representation of the Aveiro K8s structure

The entire structure is virtualised, using VMs created with Openstack. All the nodes are running Debian as the main operating system (OS), and the container runtime is containerd. Regarding the network specifications, the Container-Network Interface (CNI) in use is Flannel (Flannel is a simple and easy way to configure a layer 3 network fabric designed for K8s), combined with MetalLB for the load balancing and the NGINX Ingress Controller. Concerning persistent storage, we have installed GlusterFS and Heketi for its dynamic provision.

For security reasons, access to the functionalities is based on RBAC per use case / third-party company. For the monitoring of measurements on the K8s infrastructure, Prometheus is used together with a prometheus-node exporter.

Regarding the organisation of the namespaces, each use case / third-party company has its own namespace. This namespace organisation with the RBAC functionality allows the isolation between the functions of each partner and the cluster administration functions.

The system components use the default namespace (kube-system), and the monitoring tools use a specific namespace (monitoring).

The infrastructure software components are based in the following open-source K8s components: containerd, MetalLB, NGINX-Ingress, Kubectl, GlusterFS, Heketi, Prometheus and Flannel. More information can be found in the previous Section (Section 3.1.1).

The dynamic allocation of storage volumes was already been described in Section 3.1.6. Also, an example use case can be found in the previous Section 3.1.5.

3.2.1 5G system

For the 5G network, the Aveiro testbed uses two different 5G Cores:

- Fraunhofer FOKUS Open5GCore (fully virtualised, running on a VM).
- Druid (fully virtualised, running on containers).

The Fraunhofer FOKUS Open5GCore toolkit [3] is the worldwide first practical implementation of the 3rd Generation Partnership Project (3GPP) 5G Core network. It prototypes 3GPP Release 15 and 16 core network functionality, in a form suitable for research and development (R&D) activities. Open5GCore is interoperable with 5G New Radio (NR) base stations and user equipment (UE), as shown in Figure 6 (for more information, see D4.4 “5G-EPICENTRE experimentation facility”, Section 4.2.3).

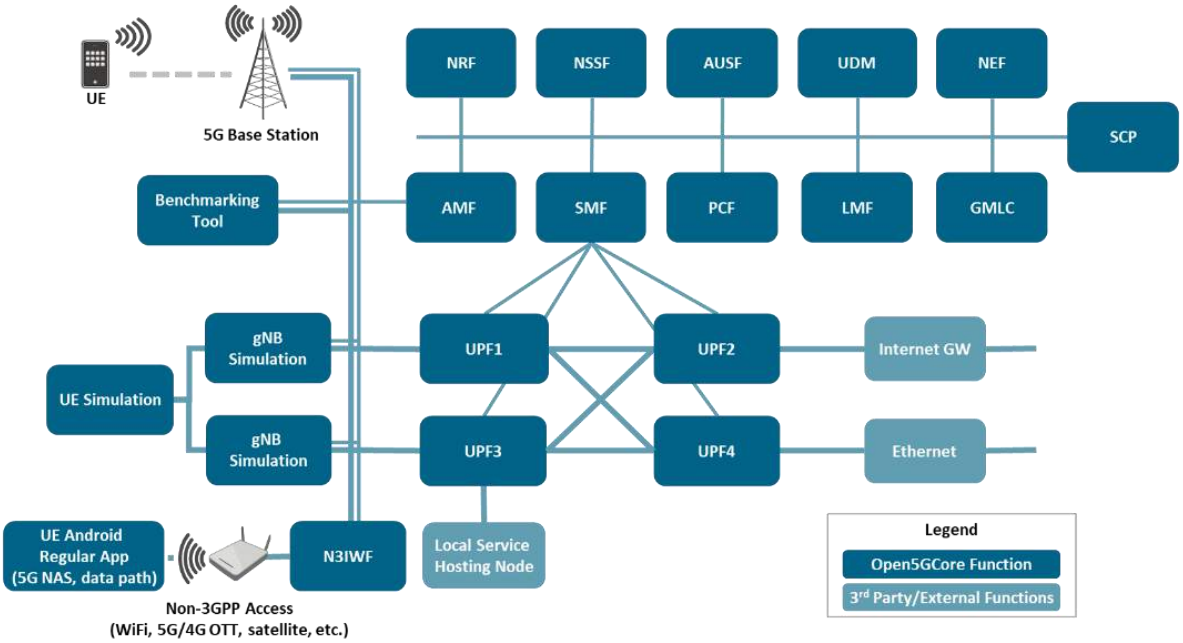

Figure 6: Open5GCore (Image is ©Fraunhofer FOKUS, retrieved from [3])

Druid's cellular solutions for business are built on its Raemis technology platform [4]. Raemis is a set of cellular software assets optimised for business use cases. The Raemis platform harnesses 5G, 4G, 3G, 2G and Wifi radios from any vendor, to implement standalone cellular core network solutions. It also integrates with mobile network operators, using standard interfaces for giving access to all of the radio resources of these operators, as shown in Figure 7.

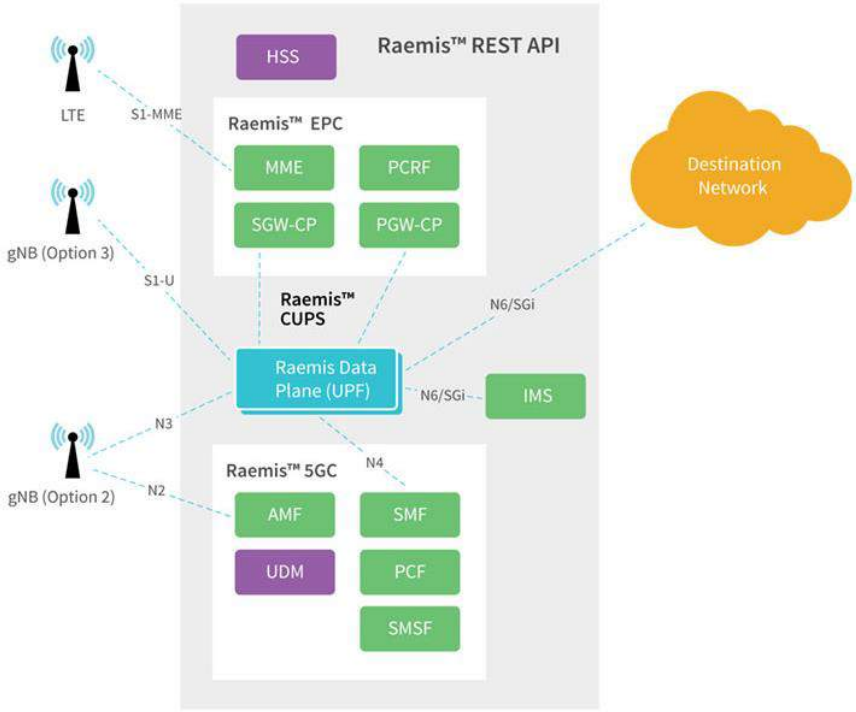

Figure 7: Druid Raemis (Image is ©Druid Software, retrieved from [4])

Druid Raemis 5G Core main features are:

- Solution specifically designed for private networks.
- 5G Ultra-Reliable Low Latency Communications (URLLC) data slicing.
- Configuration of Radio Quality of Service (QoS) and Radio Congestion Control per network.
- 5G Radio Network Slicing.
- Open Representational State Transfer (RESTful) Application Programming Interface (API) to enable Management and Orchestration (MANO) and third-party applications integration.
- Creation of multiple Packet Data Networks with QoS allocation.
- Definition of radio zones to logically group multiple gNodeB devices together.
- Resilience and redundancy options.
- Real-time System Monitoring.

For the 5G Radio Access Network (RAN), the Aveiro testbed is based on the ASOCS RAN solution.

ASOCS has built its fully virtualised CYRUS 2.0 5G vRAN solution [5] [6], that utilises the open fronthaul interface defined by Open RAN (O-RAN) and can support diversified use cases, offering flexible business models for operators, businesses, enterprises, and neutral host providers that need micro, metro, and macro scale networks.

This solution is split in multiple components, following the 3GPP 5G RAN architecture, namely Centralised Unit (CU), Distributed Unit (DU) and Radio Unit (RU), as shown in Figure 8 (for more information see D4.4 Section 4.2.3).

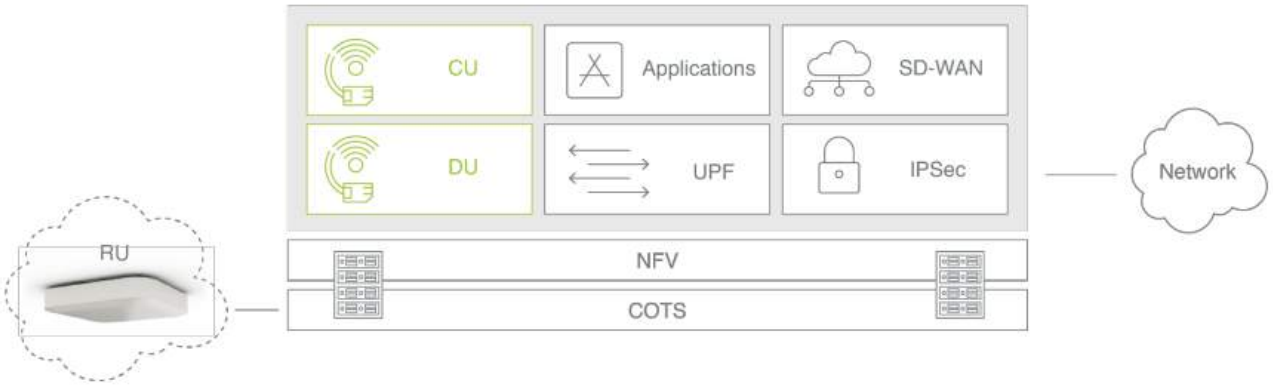

Figure 8: Cyrus 2.0 vRAN architecture (Image is ©Intel Corporation, retrieved from [6])

3.2.2 Hardware approach

The hardware implementation consists of different hardware components:

- The K8s and Open5GCore platform consists of 1 physical server Dell Power Edge R740xd, specifically virtualised with OpenStack.
- The ASOCS solution includes EVK Case, Servers, Switches, Indoor RU, POE, CPE, SIM Cards, cables.

The infrastructure hardware components can be seen in Figure 9.

Regarding the VMs running in the physical server (Dell Power Edge R740xd), they can be separated in different groups: the VMs that compose the K8s cluster, the VMs necessary for the 5G-EPICENTRE components and the 5G Core. A Graphical view can be seen in Figure 10.

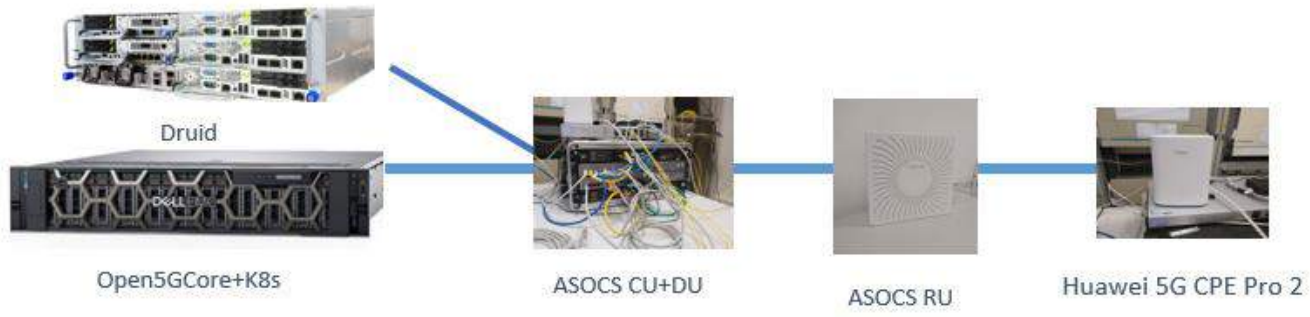

Figure 9: Infrastrutcture hardware components

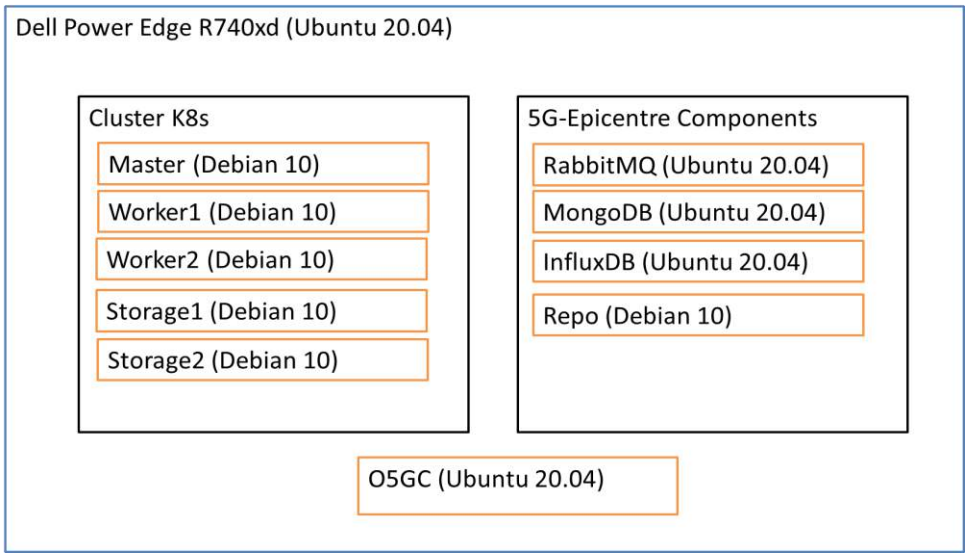

Figure 10: Virtual machines running in physical server

The Druid Raemis core is installed in a single-node K8s cluster, in one of the bare-metal sleds of the EP100.

3.2.3 Kubernetes support guidelines

For access to the Aveiro testbed K8s platform, the experimenters need to use a VPN (OpenVPN), and request the access to ALB IT department. This VPN provides access to the necessary VMs/containers.

Next, the experimenters need to access the repository VM, with the credentials (user/pass) generated by ALB (in the first access, the experimenters should change the password). In this VM they can use the K8s cluster through kubectl, or copy the K8s config file to use remotely. This config file is unique per experimenter company, and uses a specific namespace with RBAC rules for isolation/security purposes.

Experimenters are recommended to use the internal project GitLab[10] which includes a private Docker registry (registry.5gepicentre.eu), but remote registries can also be allowed, if needed.

At this moment the experimenters are able to use kubectl to deploy their applications on the K8s cluster, make updates, check the status and remove them.

Finally, experimenters are requested to follow the K8s best practices, and consult the official K8s documentation [7].

[10] 5GEPICENTRE GitLab: https://gitlab.5gepicentre.eu/

3.3 Barcelona testbed (CTTC)

The K8s infrastructure is deployed using Kubespray in 5G-EPICENTRE for hosting experimentation tools. Kubespray is an automated deployment tool that provides a highly available cluster, composable attributes, and support for most popular Linux distributions. It is a collection of Ansible playbooks, inventory, provisioning tools, and domain knowledge for deploying, configuring, and managing K8s clusters. The infrastructure consists of master nodes, each with a set of worker nodes, as shown in Figure 11. For each use case, there is a separate set of master and worker nodes with the number of worker nodes varying from one to three. An example of K8s infrastructure with a master and three worker nodes illustrated in Figure 11.

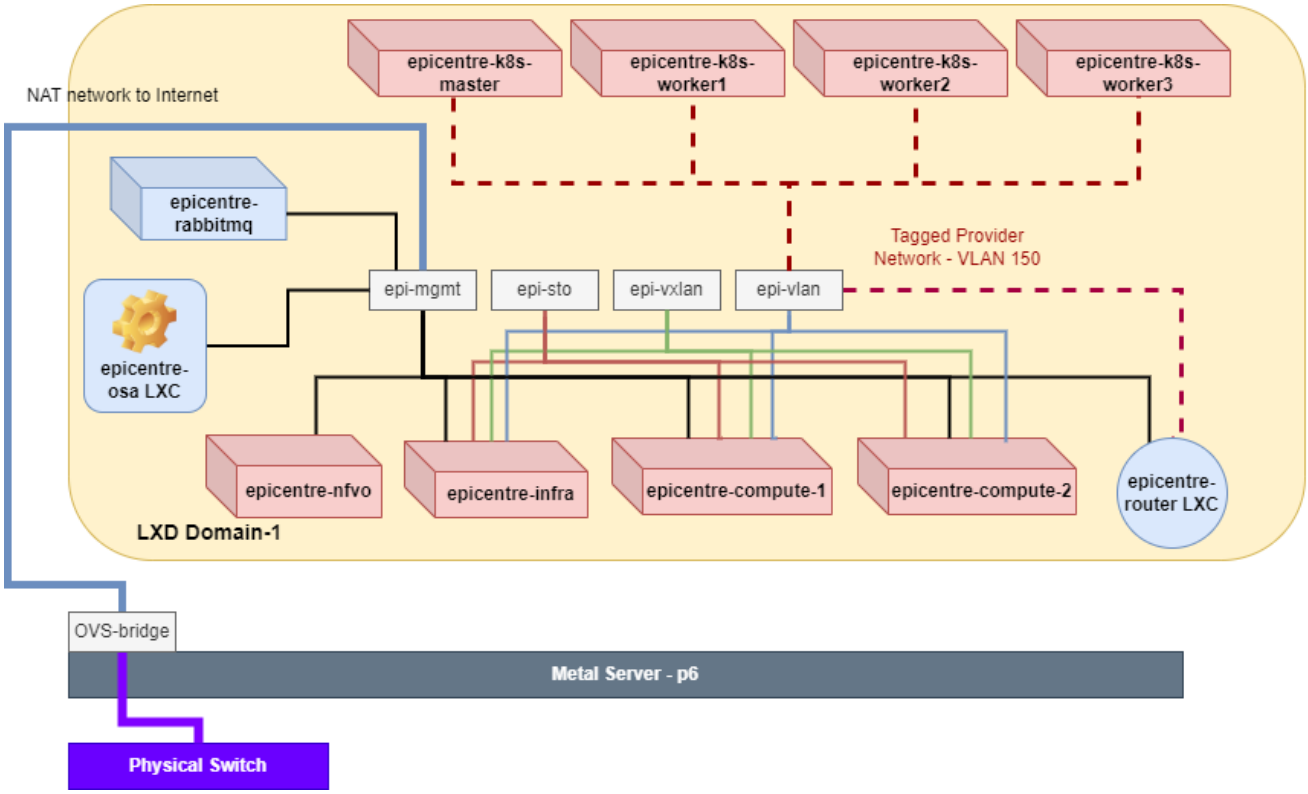

Figure 11: Representation of the CTTC K8s structure

The K8s cluster is created under the Testbed instance concept to provide isolated Cloud/Multi-Access Edge Computing (MEC) testing and validation environments. Virtualisation is essential, since this infrastructure is shared among multiple projects and researchers. The main virtualisation technology used is LXD, which enables the creation of a virtualisation experimental environment called Testbed Instance (TI). VNFs are generally orchestrated by Open-Source MANO (OSM), using K8s and/or OpenStack as the Virtualised Infrastructure Manager (VIM). However, this can be adapted depending on the testing needs. The computing resources managed by these VIMs are either VMs or Docker containers. Karmada is used to implement the multi-domain K8s clusters.

3.3.1 Infrastructure software components

This Section provides more details about the infrastructure software components. Implemented tools in 5G-EPICENTRE K8s include: (1) Kubectl; (2) Persistent Volume; (3) MetalLB LoadBalancer; and (4) NGINX Ingress Controller. Persistent Volume[11] is long-term storage in a K8s cluster. It exists beyond containers, pods, and nodes. A pod uses a persistent volume claim to get read and write access to the persistent volume. To create persistent

[11] https://kubernetes.io/docs/concepts/storage/persistent-volumes/

volumes, we use NFS, which is a shared filesystem accessed over the network. The NFS must already exist, and K8s does not run the Network File System (NFS); instead, pods access it. NFS is useful for two reasons: it allows data to persist beyond the life of a pod, and it enables multiple pods to access the same filesystem at the same time. In our K8s cluster, we have implemented an NFS server on the master node. The rest of the tools mentioned above, were described in detail in subsection 3.1.1.

3.3.2 5G network

The 5G Core (and RAN) can be deployed either through commercial products, *i.e.*, the Amarisoft CallBox, or through open-source solutions integrated with Universal Software Radio Peripherals (USRPs).

More specifically, the CTTC testbed consists of a variety of 5G Core, namely:

- Amarisoft 5G Core, as part of the Callbox.
- Containerised Open5GS.
- Containerised OpenAirInterface.

The FR1 band is supported in stand-alone (SA) and non-stand-alone (NSA) scenarios. Additionally, 4G scenarios can as well be deployed. The testbed features come from the Amarisoft RAN, as part of the Callbox equipment:

- One Amarisoft Callbox Ultimate.
- Two Amarisoft Callbox Mini.

3.3.3 Hardware approach

The CTTC 5G testbed allows the creation of multiple TIs. Each TI is an NFV ecosystem. This allows sharing the same testbed physical infrastructure and building different subtestbeds according to the experimentation needs of each use case. A TI may include:

- Computing capabilities (Central Processing Unit [CPU], Graphics Processing Unit [GPU], edge, cloud).
- 5G network capabilities (including UEs, RAN, and core).
- Other devices.

A global view of the CTTC 5G testbed is presented in Figure 12.

The left part corresponds to generic purpose servers over which the testbed instances deploy their VMs or containers. Though the configuration of this part is flexible, this component can be seen as the cloud data centre of the scenario under evaluation. The lower part corresponds to edge servers, which may be either generic purpose servers, or those equipped with GPUs for offering services to demanding URLLC applications (*e.g.*, Virtual Reality [VR]).

Since the goal is to create TIs on top of the shared infrastructure, LXD is exploited to create system containers and VMs that are part of the NFVI infrastructure of the TI. Inside each TI, the user can decide the MANO stack to deploy, which may include, for instance, OSM, K8s and Openstack.

There are multiple flavours of 5G mobile network that can be used, consisting of combinations of commercial and open-source hardware and software. Multiple real and emulated 5G devices can also be used in each testbed instance. The following sections describe in more detail each component. For monitoring purposes, Prometheus is used.

The basic computing capabilities consist of 10 servers with a total of 456 CPU cores, 2632 GB memory, 58 TB storage, and 6 GPUs distributed in two servers. For edge computing, there are 10 machines with a total of 40 cores, 147 GB memory, and 2.5 TB storage. It should be noted that the available GPUs can be virtualised for containers/VMs.

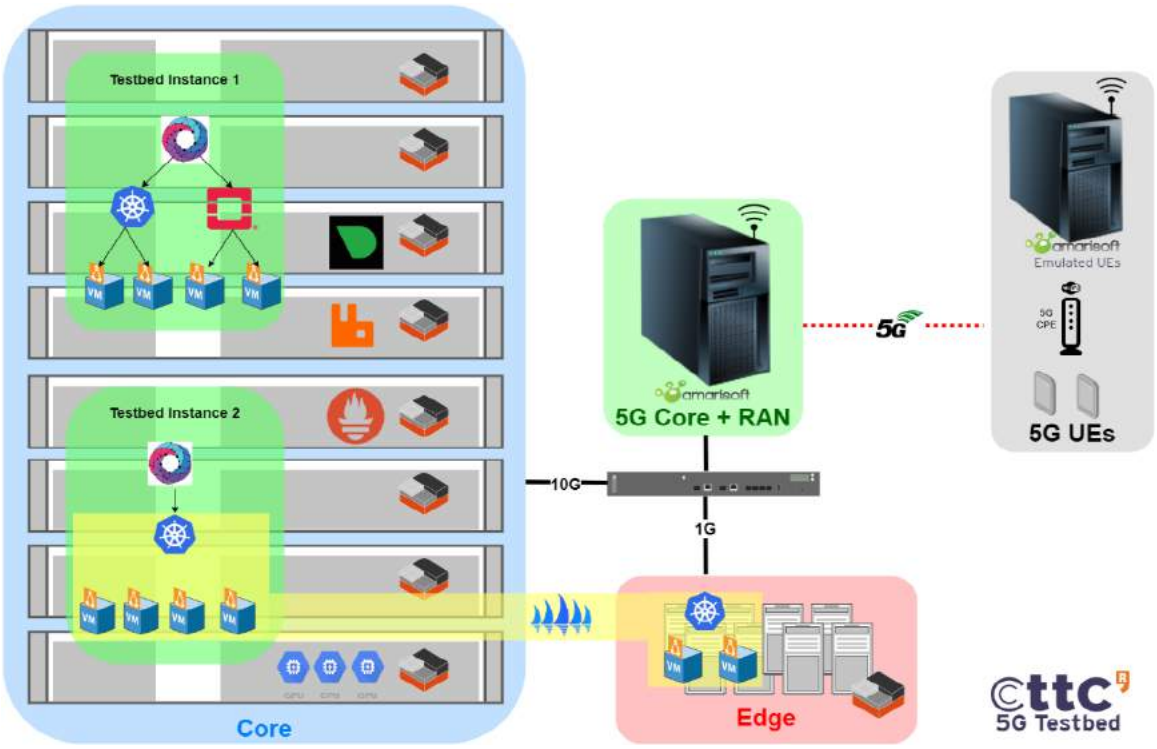

Figure 12: CTTC 5G testbed

Figure 13 shows the CTTC 5G testbed including the 5G network equipment. Additionally, more servers, networking, and measurement equipment can be made available from another testbed of the Services as Networks (SaS) group, namely the EXTREME Testbed® (Figure 14).

Figure 13: Part of the CTTC 5G testbed showing some of the servers and the Amarisoft 5G equipment

Figure 14: Part of the CTTC 5G testbed. EXTREME Testbed ®

3.3.4 Example use cases

Figure 15 shows two TIs for the use cases deployed by NEM (UC2) and ORAMA (UC8) in the CTTC 5G testbed, and two TIs used to deploy other components of the platform, namely the Publisher and RabbitMQ broker, and the Analytics Engine.

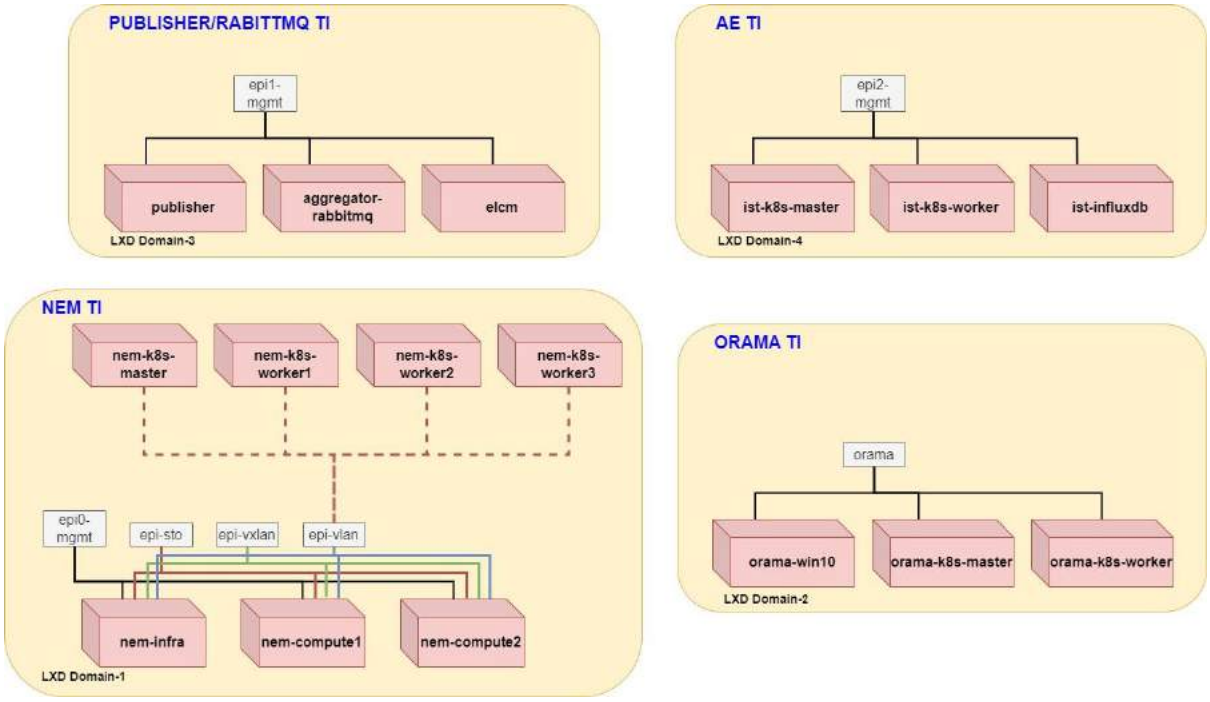

Figure 15: Testbed Instances for the use cases and other components of the CTTC 5G Barcelona testbed

3.3.5 Kubernetes support guidelines

The CTTC 5G testbed provides access to the different Tis, including the K8s clusters and other components with a VPN using an OpenVPN server. Once the user is connected to the VPN, they have administrative rights to their own K8s cluster.

The deployment of the use cases is done with the kubectl command-line tool, or with the Helm chart tool with the YAML configuration files. The testbed has access to Docker image repositories. The users can access the internal project GitLab repository (gitlab.5gepicentre.eu), the CTTC internal GitLab repository (gitlab.cttc.cat), or any external repository.

3.4 Berlin testbed (HHI)

A dedicated server is being installed in HHI testbed, which is in the same network infrastructure as the 5G Core. This enables the server to extract all the useful information regarding the deployed use cases directly from the core to process further. For processing, different VMs are deployed in the server, and are managed by VMWare. Docker CE is used to create Docker images, which are managed under K8s. A K8s cluster is composed of such 3 VMs, which are distributed as 1 master and 2 worker VMs. A message broker, namely RabbitMQ, is integrated in the cluster as a component which takes information (or message) in JavaScript Object Notation (JSON) format. This message is further sent to the publisher by the RabbitMQ. For monitoring purposes, Prometheus is utilised (Figure 16).

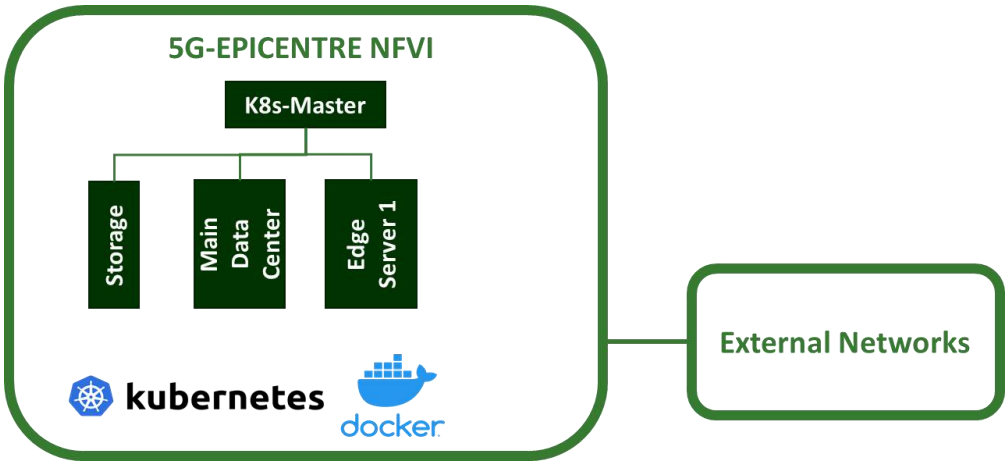

Figure 16: K8s structure at 5G Berlin testbed

It has proven successful that the names of the use case owners are used for the namespaces in the use cases we tested. This makes it easier to distinguish and manage the different use cases.

Used K8s namespases are listed in Table 3.

Table 3: Namespaces in the HHI testbed

namespace	Usage
default	components, services for RabbitMQ
rabbitmq-system	RabbitMQ cluster operator
hhi, youbiquo	components, services for use cases
kube-flannel, kube-node-lease, kube-public, kube-system	standard components, services of kubernetes
local-path-storage	persistent volume
metallb-system	Load Balancer, direct access to services
monitoring	Prometheus
nginx-ingress	Load Balancer, routing to different services

For more information about components and services see section 3.1.1.

3.4.1 5G network

In the HHI 5G testbed, a 5G Core from ng4T[12] is used in SA mode. The 5G Core runs on its own physical server, for which there is currently no container solution. The core is connected to the RAN system.

As can be seen in Figure 17, the topology of the 5G Core includes the Access and Mobility Management Function (AMF), Session Management Function (SMF), User Plane Function (UPF) and Unified Data Management (UDM).

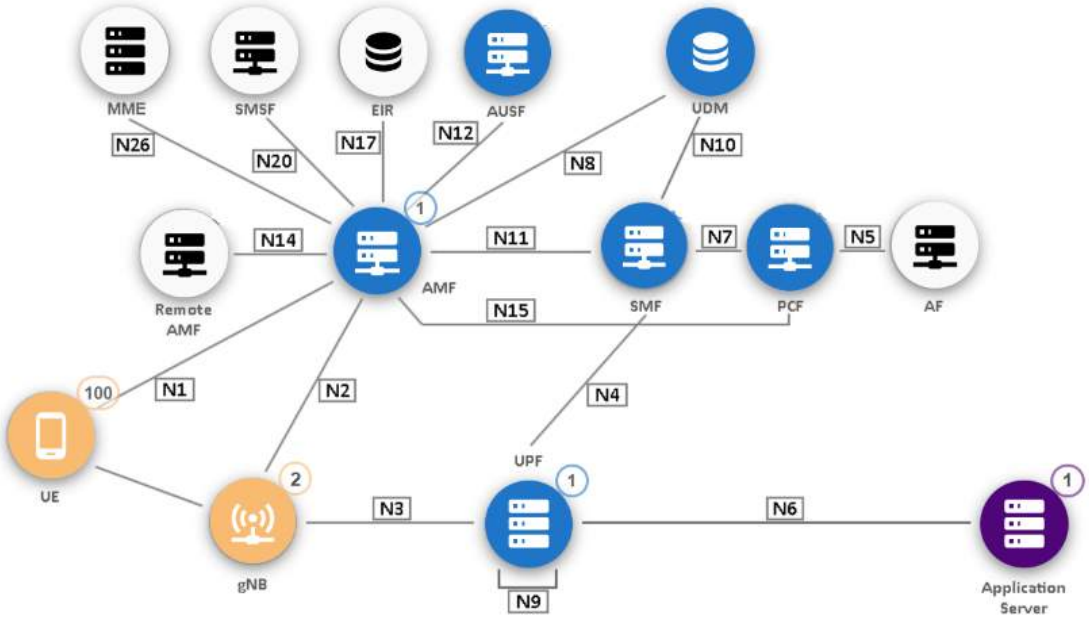

Figure 17: Topology of 5G core from NG4T

The gNB is connected to the core via the N2 control plane and N3 user plane interfaces. The connection to the application server and the Internet gateway is made via the N6 interface.

It is a 5G Core emulator licensed to connect up to 25 gNBs and supports up to 100 UEs. Currently, the core is set to support two Public Land Mobile Networks (PLMNs), each with its own network slice.

3.4.2 Hardware approach

The hardware implementation consists of different hardware components (Figure 18-Figure 20), namely:

- Macro Cell
  - RRH Nokia 5G SA, AirScale AEQE 64T64R (on the roof top).
  - gNB Nokia 5G SA, ABIL BBMOD-1.
- 5G Core from ng4T, running on Dell Server R440.
- Componet interconnection switch Dell EMC S5224F-ON Switch.
- FortiGate FG-200F.
  - Internet gateway.
  - VPN access.
  - Firewall.
  - Router.
- Application server Dell PowerEdge R440 Server.

[12] https://www.ng4t.com/

- managed by VMWare.
- K8s cluster running on VMs.

3.4.3 Kubernetes support guidelines

If an experimenter wants to deploy a use case on the K8s cluster of the 5G Berlin testbed, a number of preparatory steps need to be taken.

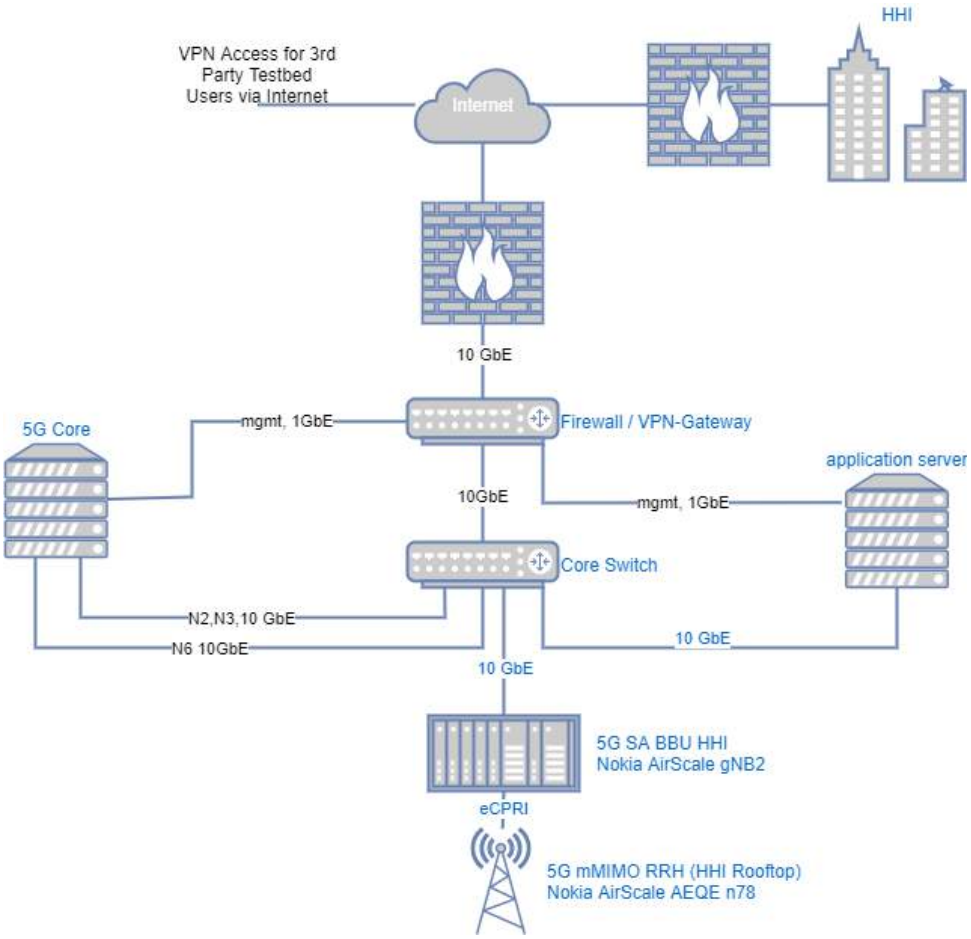

Figure 18: Network architecture of 5G Berlin testbed

Figure 19: Nokia gNB and Nokia RRH

Figure 20: 5G Core, Application server, FortiGate

First, the preparation of configuration files is required: In order to deploy an application on a K8s cluster, configuration files must be created, that describe the various resources that are to be deployed. These resources include pods, services such as LoadBalancer, volumes and secrets/tokens used. The resource requirements (like CPU and memory space) should be listed in the pods' configuration.

Secondly, a deployment configuration file must be created: The deployment configuration file describes how the pods and associated resources should be provisioned in the cluster. This file should contain information, such as the number of replicas to be provisioned and the container image to be used; usually this file is expressed in a YAML format.

Third, the application should be deployed in the cluster. To do this, the experimenter loads its container images and configuration file into the internal project GitLab, which contains a private Docker registry (registry.5gepicentre.eu) or provides access to its own repository.

For the deployment of the container image, there are two possibilities. First, the 5G Berlin testbed operator provides the application on the K8s cluster after consultation with the experimenter. Secondly, the experimenter receives external VPN access to the required VM, which they can access with the credentials (user/password) generated by HHI. The application can be deployed to the cluster with kubectl, the K8s command-line tool. It is then advisable to check the deployment status. Once the application has been deployed, the status can be checked with kubectl commands.

If the application is to be updated after deployment, updates can be performed to fix bugs or add new features. The updates are performed by creating a new version of the deployment configuration file and updating the pods and other related resources.

3.4.4 5G testbed management

The infrastructure is shared by several projects and researchers, so a coordinated agreement among the projects by a scheduler is necessary (planned within the project architecture, see D1.3 and D1.4). The individual network components are managed via a dedicated management Virtual Local Area Network (VLAN), which can be accessed via limited VPN access. The VMs on the application server managed by VMWare and works with the OS Ubuntu 20.04. In addition to the 3 VMs for the K8s cluster, further VMs areVM for an iPerf server, VM for the publisher, *etc*.

The components and services required for the use cases are only available to project partners of the 5G-EPICENTRE project.

For performance measurements with UC3, the 5G TrafficManager, remote-iPerf-agent and Publisher from UMA were used. For this purpose, measurement data was forwarded to the Publisher via the RabbitMQ broker and Prometheus. This can also be used for other use cases.

4 Conclusions

This document is the deliverable related to Task T2.1 “Cloud-native infrastracture”. This deliverable presents details regarding the cloud-native transformation of the infrastructure for each testbed used in the project.

The document starts by presenting the benefits of cloud-native transformation in 5G networks in general and then provides details regarding the approach followed by each testbed in order to cloud nativise their infrastructure and to support K8s. The infrastructure software and hardware components and details such as the used monitoring system are also presented. A special emphasis is given to the presentation of guidelines for access to the K8s platforms: a sub-section is dedicated to every testbed briefly describing the steps that have to be taken when an experimenter wants to perform a deployment on the specific testbed.

The cloud-native transformation of the testbeds will benefit the UCs and the PPDR verticals by providing better performance and faster instantiation times, better availability and reliability. It will improve the agility of the 5G network and will support the development and deployment of 5G services and applications.

References

[1] 5G-PPP Software Network Working Group. (2020). Cloud Native and 5G Verticals' services. https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-SN-WG-5G-and-Cloud-Native.pdf

[2] 5G-PPP Software Network Working Group. (2019). Cloud-Native and Verticals' services. https://5g-ppp.eu/wp-content/uploads/2019/09/5GPPP-Software-Network-WG-White-Paper-2019_FINAL.pdf

[3] Fraunhofer FOKUS. (n.d.). *Open5GCore*. Retrieved March 1, 2023, from https://www.open5gcore.org/

[4] DRUID. (n.d.). Raemis™ – Cellular Network Technology. Retrieved March 1, 2023, from https://www.druid-software.com/raemis-cellular-network-technology/

[5] ASOCS. (n.d.). *ASOCS*. Retrieved March 1, 2023, from www.asocscloud.com

[6] ASOCS. (n.d.). ASOCS' CYRUS 2.0 Delivers Flexible 4G/5G Indoor and Macro vRAN. Retrieved March 1, 2023, from https://asocscloud.com/wp-content/uploads/2019/10/asocs_intel-network-builders_solution-brief_10_2019.pdf

[7] Kubernetes. (2023, March 1). *Kubernetes Documentation*. Retrieved March 1, 2023, from https://kubernetes.io/docs/home/